Université catholique de Louvain
Louvain School of Statistics

LSTAT2130 - Bayesian Statistics

Project - Group Q

Lionel Lamy - 1294-1700
Adrien Kinart
Simon Lengendre

May 21, 2021

Contents

1 Introduction

```
library(EnvStats)
library(mnormt)
library(coda)
```

```
Table <- matrix(data = c(25, 69, 65, 106, 80, 106, 136, 94, 76, 46, 17,
    36, 47, 58, 47, 53, 59, 54, 33, 21), nrow = 2, byrow = T)
rownames(Table) <- c("Flanders", "Wallonia")
colnames(Table) <- c("<1200", "[1200-1500)", "1500-1800", "1800-2300",
    "2300-2700", "2700-3300", "3300-4000", "4000-4900", "4900-6000", ">6000")
Intervals <- c(0, 1200, 1500, 1800, 2300, 2700, 3300, 4000, 4900, 6000,
    1e+10)

NbFlemish <- sum(Table[1, ])
NbWaWalloons <- sum(Table[2, ])
```

```
kappaFct <- function(phi) {
    1/phi
}
lambdaFct <- function(phi, mu) {
    1/(phi * mu)
}
```

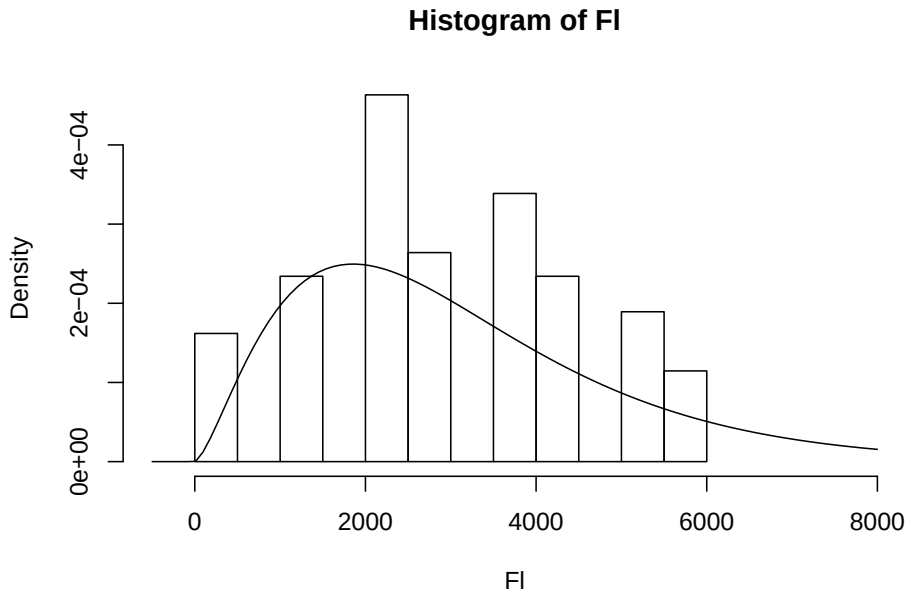
```
# Flanders
Fl <- c(rep(1200, 25), rep(1350, 69), rep((1500 + 1800)/12, 65), rep((1800 +
    2300)/2, 106), rep((2300 + 2700)/2, 80), rep(3000, 106), rep((3300 +
    4000)/2, 136), rep(4450, 94), rep((4900 + 6000)/2, 76), rep(6000, 46))

Estimation_Fl <- egamma(Fl)
Estimated_kappa_Fl <- Estimation_Fl$parameters["shape"]
Estimated_lambda_Fl <- 1/Estimation_Fl$parameters["scale"]
Estimated_mu_Fl <- Estimated_kappa_Fl/Estimated_lambda_Fl
Estimated_mu_Fl
```

```
##    shape
## 3089.944
```

```
Estimated_phi_Fl <- 1/Estimated_kappa_Fl
Estimated_phi_Fl
```

```
##    shape
## 0.399687
```

```
estimGamma_Fl <- rgamma(10000, Estimated_kappa_Fl, Estimated_lambda_Fl)
hist(Fl, probability = T, xlim = c(-500, 8000))
curve(dgamma(x, Estimated_kappa_Fl, Estimated_lambda_Fl), add = TRUE)
```



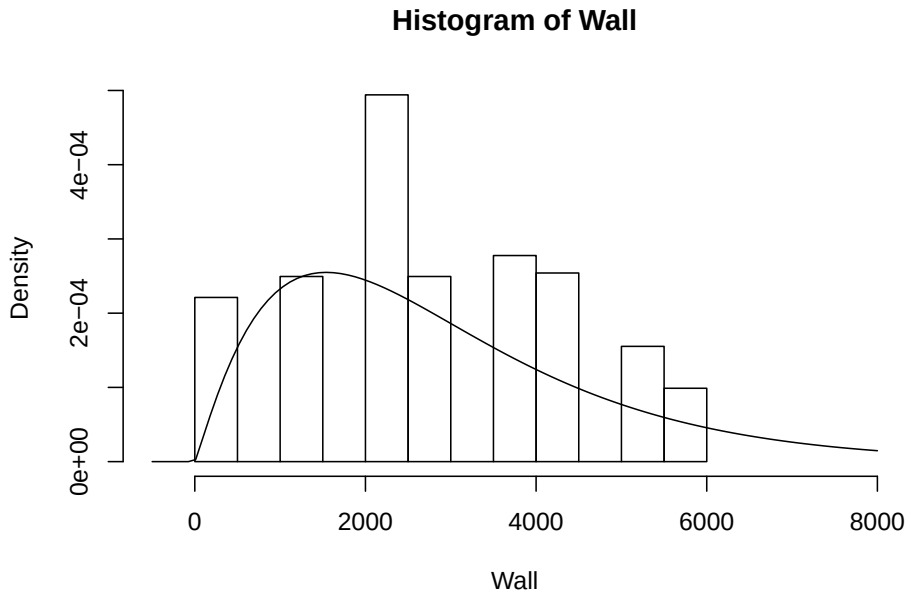

```
  # Wallonia
  Wall <- c(rep(1200, 17), rep(1350, 36), rep((1500 + 1800)/12, 47), rep((1800 +
      2300)/2, 58), rep((2300 + 2700)/2, 47), rep(3000, 53), rep((3300 +
      4000)/2, 59), rep(4450, 54), rep((4900 + 6000)/2, 33), rep(6000, 21))


  Estimation_Wall <- egamma(Wall)
  Estimated_kappa_Wal <- Estimation_Wall$parameters["shape"]
  Estimated_lambda_Wal <- 1/Estimation_Wall$parameters["scale"]
  Estimated_mu_Wal <- Estimated_kappa_Wal/Estimated_lambda_Wal
  Estimated_mu_Wal
```

```
##    shape
## 2914.882
```

```
  Estimated_phi_Wal <- 1/Estimated_kappa_Wal
  Estimated_phi_Wal
```

```
##    shape
## 0.471615
```

```
  estimGamma_Wal <- rgamma(10000, Estimated_kappa_Wal, Estimated_lambda_Wal)
  hist(Wall, probability = T, xlim = c(-500, 8000))
  curve(dgamma(x, Estimated_kappa_Wal, Estimated_lambda_Wal), add = TRUE)
```




```
  muExample <- 2500
  phiExample <- 1
  kappaExample <- kappaFct(phiExample)
  lambdaExample <- lambdaFct(muExample, phiExample)

  pgamma(2700, kappaExample, lambdaExample) - pgamma(2300, kappaExample,
      lambdaExample)
```

```
## [1] 0.05892352
```

```
  dgamma(2700, kappaExample, lambdaExample)
```

```
## [1] 0.0001358382
```

2 Question 1

Let $\theta_k := (\mu_k, \phi_k)$ be the set of parameters for a HNI with respect to region k.

2.1 (a) Theoretical probability

Let X be the monthly net income of 1123 Belgian households net income (HNI) older than 30 years. Regardless the 2 regions ($k = \{1, 2\}$ wrt Flanders and Wallonnia, respectively), is assumed it follows a Gamma distribution. It can be reparametrised in terms of its mean μ and dispersion parameter ϕ with the following trick:

$$\text{shape: } \kappa = \frac{1}{\phi}$$
$$\text{rate: } \lambda = \frac{1}{\phi\ \mu}$$

For both regions $k = \{1, 2\}$: This gives

$$f(x_k) = \frac{(\phi_k \mu_k)^{-1/\phi_k}}{\Gamma(\phi_k^{-1})}\ x_k^{1/\phi_k - 1} \exp\left(\frac{-x_k}{\phi_k \mu_k}\right)$$

Then, the probability to fall into a certain HNI interval I is:

$$P(x_k \in I_j) = \int_{I_j} \frac{(\phi_k \mu_k)^{-1/\phi_k}}{\Gamma(\phi_k^{-1})} x_k^{1/\phi_k - 1} \exp\left(\frac{-x_k}{\phi_k \mu_k}\right) \mathrm{d}x$$

Using CDF writing, for a region k ,

$$p_j = \begin{cases} F(x_j, \kappa, \lambda) \text{ if j } = 1 \\ F(x_{j+1}, \kappa, \lambda) - F(x_j, \kappa, \lambda) \text{ if } j \in \{2, ..9\} \\ 1 - F(x_j, \kappa, \lambda) \text{ if j } = 10 \end{cases}$$

$$p_j = \begin{cases} \frac{1}{\Gamma(\kappa)} \gamma(\kappa, \lambda x_j) \text{ if j } = 1 \\ \frac{1}{\Gamma(\kappa)} \Big(\gamma(\kappa, \lambda x_{j+1}) - \gamma(\kappa, \lambda x_j) \Big) \text{ if} j \in \{2, ..9\} \\ 1 - \frac{1}{\Gamma(\kappa)} \gamma(\kappa, \lambda x_j) \text{ if j } = 10 \end{cases}$$

In terms of μ and ϕ

$$p_j = \begin{cases} \frac{1}{\Gamma(\phi^{-1})} \gamma(\phi^{-1}, (\mu\phi)^{-1} x_j) \text{ if j } = 1 \\ \frac{1}{\Gamma(\phi^{-1})} \Big(\gamma(\phi^{-1}, (\mu\phi)^{-1} x_{j+1}) - \gamma(\kappa, (\mu\phi)^{-1} x_j) \Big) \text{ if} j \in \{2, ..9\} \\ 1 - \frac{1}{\Gamma(\phi^{-1})} \gamma(\phi^{-1}, (\mu\phi)^{-1} x_j) \text{ if j } = 10 \end{cases}$$

$$p_j = \begin{cases} \frac{1}{\Gamma(\phi^{-1})} \int_0^{(\mu\phi)^{-1} x_j} t^{\phi^{-1} - 1} e^{-t} \mathrm{d}t \text{ if j } = 1 \\ \frac{1}{\Gamma(\phi^{-1})} \int_{(\mu\phi)^{-1} x_j}^{(\mu\phi)^{-1} x_{j+1}} t^{\phi^{-1} - 1} e^{-t} \mathrm{d}t \text{ if} j \in \{2, ..9\} \\ \frac{1}{\Gamma(\phi^{-1})} \int_{(\mu\phi)^{-1} x_j}^{+\infty} t^{\phi^{-1} - 1} e^{-t} \mathrm{d}t \text{ if j } = 10 \end{cases}$$

2.2 (b) Theoretical expression for the likelihood

On behalf of writing simplicity, the region index is removed. Since the frequency distribution in a given region is multinomial, i.e.

We have, writing $P := (p1, .., p_{10})$ and $X := (X_1, ...X_{10})$:

$$\begin{aligned} X|P \sim \mathrm{Mul}(N, P) &= \frac{x!}{x_1! \dots x_{10}!} p_1^{x_1} \times ... \times p_{10}^{x_{10}} \text{ when} \sum_{j=1}^{10} p_j = 1 \\ &= 0 \text{ otherwise} \end{aligned}$$

Up to a multiplicative constant, the likelihood can be written as follow:

$$L(\theta_k, D_k) = P(D_k | \mu_k, \phi_k) \propto \prod_{j=1}^{10} p_{k,j}^{x_{k,j}}$$

2.3 Taking approximation

p_j corresponds to the area in the j^{th} interval. One can take the approximation mean the mean,e.g. $x_{Flanders,3} = (1500 + 1800)/2 = 1650$? On can approximate that with $f(x_i)\Delta$ where Δ is the unit of measurement.

$$\begin{aligned} p_j = P(x_j - \frac{\Delta_j}{2} < x_j < x_j + \frac{\Delta_j}{2}) &\approx f(x_j)\Delta_j \\ &\approx \frac{1}{\Gamma(\phi_k^{-1})}\big(\phi_k\mu\big)^{-1/\phi_k} x_j^{\frac{1}{\phi_k}-1} \exp\big(\frac{-x_j}{\phi_k\mu}\big)\Delta_j \end{aligned}$$

This gives for the likelihood:

$$\begin{aligned} P(D|\mu,\phi) &\propto \prod_{j=1}^{10} x_j^{\frac{1}{\phi_k}-1} \exp\big(\frac{-x_j}{\phi_k\mu}\big)\Delta_j \\ &\propto \exp\big(\frac{-\sum x_j}{\phi_k\mu}\big) \prod_{j=1}^{10} x_j^{\frac{1}{\phi_k}-1}\Delta_j \end{aligned}$$

2.4 Not taking approximation but the CDF differences

$$\begin{aligned} P(D|\kappa,\lambda) &\propto \big(\frac{1}{\Gamma(\kappa)}\big)^{\sum_{i=1} x_j} \gamma(x_1,\kappa,\lambda)^{x_1} \left[\prod_{j=2}^{9} \Big(\gamma(x_j,\kappa,\lambda) - \gamma(x_{j-1},\kappa,\lambda)\Big)^{x_j}\right] \big(1-\gamma(x_{10},\kappa,\lambda)\big)^{x_{10}} \\ &\propto \big(\frac{1}{\Gamma(\kappa)}\big)^{\sum_{i=1} x_j} \left(\int_0^{\lambda x_1} x^{\kappa-1}e^{-x}\mathrm{d}x\right)^{x_1} \left[\prod_{j=2}^{9} \Big(\int_{\lambda x_j-1}^{\lambda x_j} x^{\kappa-1}e^{-x}\mathrm{d}x\Big)^{x_j}\right] \left(\int_{\lambda x_{10}}^{+\infty} x^{\kappa-1}e^{-x}\mathrm{d}x\right)^{x_{10}} \end{aligned}$$

If we write $[0;x_1],[x_1;x_2],...,[x_9;x_{10}],[x_{10},+\infty]$ as $I_1, I_2, ..., I_9$ and I_{10}. The notation can be lightened. With respect to the region k, this gives:

$$P(D_k|\kappa_k,\lambda_k) \propto \big(\frac{1}{\Gamma(\kappa)}\big)^{\sum_{i=1} x_j} \prod_{j=1}^{10} \left(\int_{\lambda_k I_j} x^{\kappa_k-1}e^{-x}\mathrm{d}x\right)^{x_{k,j}}$$

Writing in terms of μ_k and ϕ_k:

$$P(D_k|\mu_k,\phi_k) \propto \prod_{j=1}^{10} \left(\frac{1}{\Gamma(\phi^{-1})} \int_{\frac{I_j}{\phi_k\mu_k}} x^{(\phi_k^{-1}-1)}e^{-x}\mathrm{d}x\right)^{x_{k,j}}$$

2.5 NOT taking approximation but with PDF definitions:

3 Question 2 : Priors

- Statement 1: we are at 95% convinced that the mean net monthly household income in a given region is in the interval (2400, 3600). If one assumes a normal distribution for the mean μ_k (à justifier), then it is possible to get a prior of the distriubtion for both regions:

For both regions $\mu_0 = 3000$. Then, to get the standard deviation:

$$3000 - t_{(n_k-1,1-\alpha/2)}\frac{s_k}{\sqrt{n_k}} = 2400$$
$$\rightarrow \hat{\sigma}_0 = \frac{s_k}{\sqrt{n_k}} = \frac{600}{t_{(n_k-1,1-\alpha/2)}}$$

```
mu_prior <- 3000
sigma_prior <- 306.12
```

$\hat{\sigma}_{Fl} = 306$

So we have

$$\mu \sim N(\mu_0 = 3000, \sigma_0 = 306)$$

$$\pi(\mu) \propto \sigma_0^{-1/2} \exp\big(-\frac{1}{2\sigma_0^2}(\mu - \mu_0)^2\big)$$

- dispersion parameter:

If one considers that the parameter can be in any point within the interval (0.0, 10.0) with the same probability, then one could say that it follows a uniform distribution between those to interval

$$\phi_k \sim U(a = 0, b = 10) \propto 1_{0,10}$$

Using the entropy theorem, the conjugate prior would be the product of the two last quantity:

$$\begin{aligned}\text{prior:} P(\mu_k, \phi_k) &= P(\mu_k)\ P(\phi_k) \\ &\propto \sigma_0^{-1/2} \exp\big(-\frac{1}{2\sigma_0^2}(\mu - \mu_0)^2\big) 1_{0;10}\end{aligned}$$

4 Question 3

4.1 Question 3a: posterior

4.1.1 With approximation

$\sigma^2 = \phi\mu^2$

$$P(\mu, \phi|D) \propto \exp\big(\frac{-\sum x_j}{\phi_k\mu}\big) \prod_{j=1}^{10} x_j^{\frac{1}{\phi_k}-1} \Delta_j \sigma^{-1/2} \exp\big(-\frac{1}{2\sigma^2}(\bar{x}_k - \mu)^2\big) 1_{0,10}$$
$$\propto \exp\big(\frac{-\sum x_j}{\phi_k\mu}\big) \prod_{j=1}^{10} x_j^{\frac{1}{\phi_k}-1} \Delta_j \big(\frac{1}{\sqrt{\phi}\mu}\big) \exp\big(-\frac{1}{2\phi\mu^2}(\bar{x}_k - \mu)^2\big) 1_{0,10}$$

4.1.2 Without approximation

$$P(\mu_k, \phi_k|D) \propto \left(\prod_{j=1}^{10} \left(\frac{1}{\Gamma(\phi^{-1})} \int_{\frac{I_j}{\phi_k \mu_k}} x^{(\phi_k^{-1}-1)} e^{-x} \mathrm{d}x \right)^{x_{k,j}} \right) \sigma_0^{-1/2} \exp\big(-\frac{1}{2\sigma_0^2}(\mu - \mu_0)^2 \big) 1_{0;10}$$

4.2 3.b

the log likelihood is, up to an additive constant:

$$l(\mu, \phi) \propto \sum_{j=1}^{10} x_j \ln \left(\frac{1}{\Gamma(\phi^{-1})} \int_{\frac{I_j}{\phi_k \mu_k}} x^{(\phi_k^{-1}-1)} e^{-x} \mathrm{d}x \right)$$

so the log-posterior is:

$$h(\mu, \phi) \propto l(\mu, \phi) + \ln(\pi(\mu, \phi))$$

$$h(\mu, \phi) = C^t + \sum_j x_j \ln\big(F(\mu, \phi, x_{j+1}) - F(\mu, \phi, x_j)\big) - \frac{1}{2\sigma_0^2}(\mu - \mu_0)^2$$

```
lpostold <- function(theta, freq) {
    Intervals <- c(0, 1200, 1500, 1800, 2300, 2700, 3300, 4000, 4900, 6000,
        Inf)  # length =11. J'ai rajouté INFINI
    mu <- theta[1]
    phi <- theta[2]
    kappa <- 1/phi
    lambda <- 1/(phi * mu)
    n <- length(freq)
    # intialisation
    LL <- 0

    for (i in 1:n) {
        LL <- LL + freq[i] * log(pgamma(Intervals[i + 1], kappa, lambda) -
            pgamma(Intervals[i], kappa, lambda))
    }

    logPost <- LL + dnorm(mu, mean = mu_prior, sd = sigma_prior, log = T) +
        dunif(phi, min = 0, max = 10, log = T)  # /!\ mu_prior et sigma_prior doivent avoir
    return(logPost)
}

# Intervals of the frequency table
intervals = c(0, 1200, 1500, 1800, 2300, 2700, 3300, 4000, 4900, 6000,
    Inf)

# Utility function to compute the prob of being between high and low
pgammadiff = function(low, high, kappa, lambda) {
    pgamma(high, kappa, lambda) - pgamma(low, kappa, lambda)
```

```
}

# Utility fonctions (already defined in the setup)
kappa = function(phi) {
    1/phi
}
lambda = function(phi, mu) {
    1/(phi * mu)
}

mu_prior = 3000
sigma_prior = 306.12
lpost = function(theta, freq) {
    # Transform mu and phi -> kappa and lambda
    kappa = kappa(theta[2])
    lambda = lambda(theta[2], theta[1])

    # Likelihood : sum_j^n[x_j * ln(probability_of_being_in_interval_j)]
    LL = sum(sapply(1:length(freq), function(j) {
        freq[j] * log(pgammadiff(low = intervals[j], high = intervals[j +
            1], kappa, lambda))
    }))

    # Log posterior
    lpi = dnorm(theta[1], mu_prior, sigma_prior, log = T) + dunif(theta[2],
        0, 10, log = T)
    lpost = LL + lpi

    names(lpost) = "lpost"
    return(lpost)
}
```

```
lpost(c(Estimated_mu_Fl, Estimated_phi_Fl), t(Table[1, ]))
```

```
##     lpost
## -1866.122
```

```
lpost(c(Estimated_mu_Wal, Estimated_phi_Wal), t(Table[2, ]))
```

```
##    lpost
## -1011.04
```

5 4.

5.1 Estimation

Start from the log-posterior $h(\mu_k, \phi_k)$. One assumes the distribution is unimodale then, if it is normally distributed, then the mode=mean, which means that the mean can be found by optimizing the log-likelihood with respect to its parameters. (aussi assumer μ et ϕ indépendant??), then blablabla (expliquez avec hessienne).

Based on http://www.sumsar.net/blog/2013/11/easy-laplace-approximation/#:~:text=Laplace%20Ap

5.1.1 Flanders

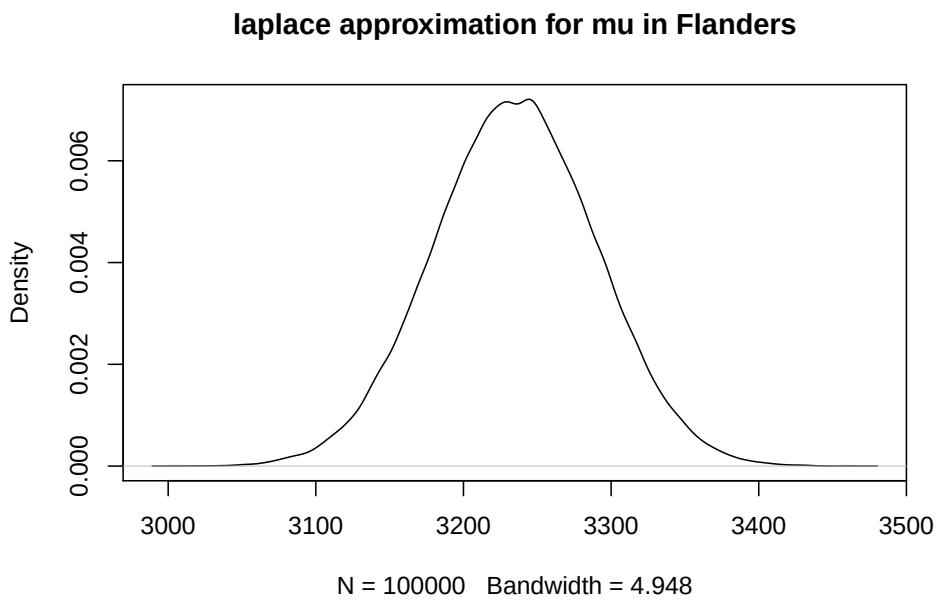
```
  inits_Fl <- c(mu = Estimated_mu_Fl, phi = Estimated_phi_Fl)
  fit_Fl <- optim(inits_Fl, lpost, control = list(fnscale = -1), hessian = TRUE,
      freq = Table[1, ])
  param_mean_Fl <- fit_Fl$par
  param_cov_mat_Fl <- solve(-fit_Fl$hessian)
  round(param_mean_Fl, 2)
```

```
##  mu.shape phi.shape
##   3234.50      0.23
```

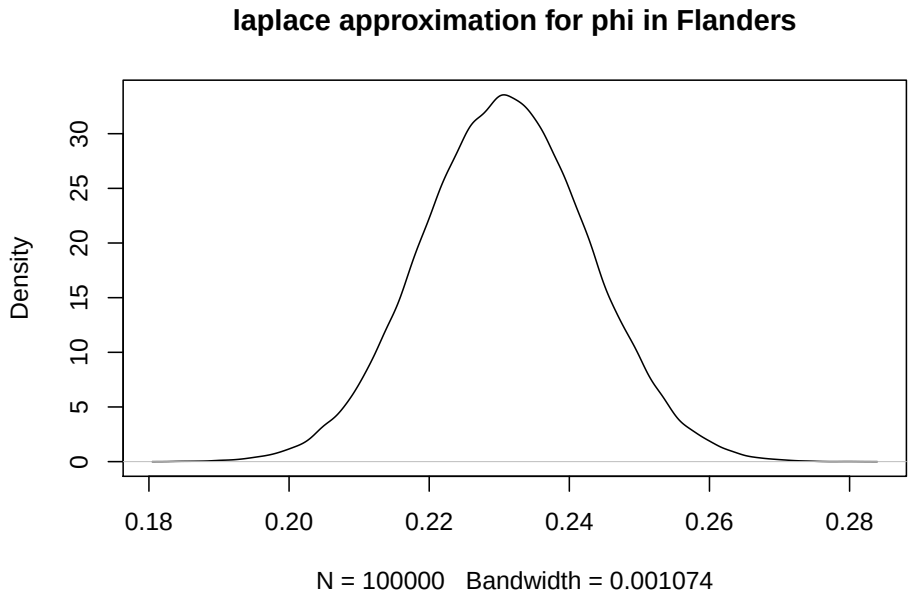
```
  round(param_cov_mat_Fl, 3)
```

```
##           mu.shape phi.shape
## mu.shape  3024.310     0.025
## phi.shape    0.025     0.000
```

```
  library(mvtnorm)
  samples_Fl <- rmvnorm(1e+05, param_mean_Fl, param_cov_mat_Fl)
  plot(density(samples_Fl[, 1]), main = "laplace approximation for mu in Flanders")
```




```
  plot(density(samples_Fl[, 2]), main = "laplace approximation for phi in Flanders")
```



5.1.2 Wallonia

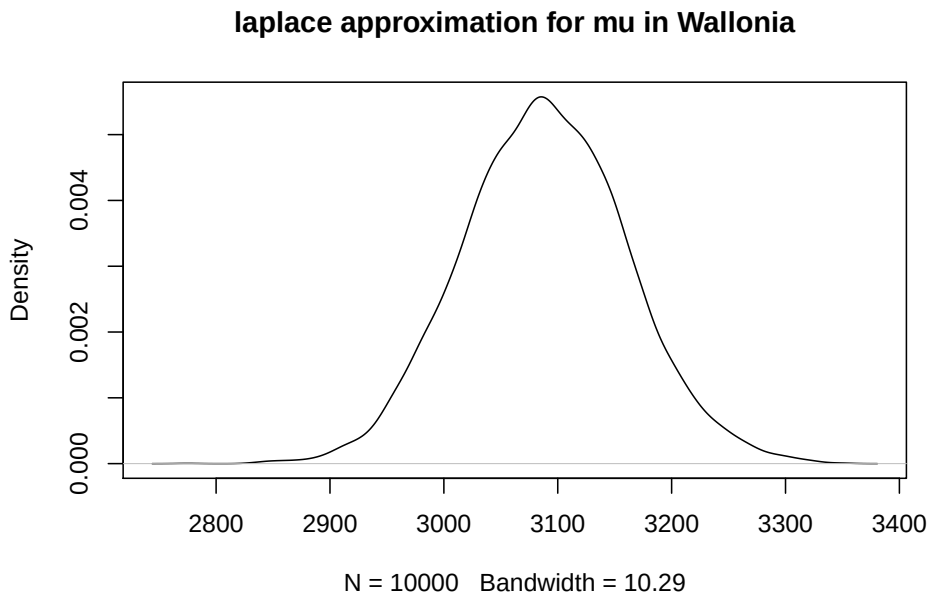
```
  inits_Wal <- c(mu = Estimated_mu_Wal, phi = Estimated_phi_Wal)
  fit_Wal <- optim(inits_Wal, lpost, control = list(fnscale = -1), hessian = TRUE,
      freq = Table[2, ])
  param_mean_Wal <- fit_Wal$par
  param_cov_mat_Wal <- solve(-fit_Wal$hessian)
  round(param_mean_Wal, 2)
```

```
##  mu.shape phi.shape
##   3089.32      0.24
```

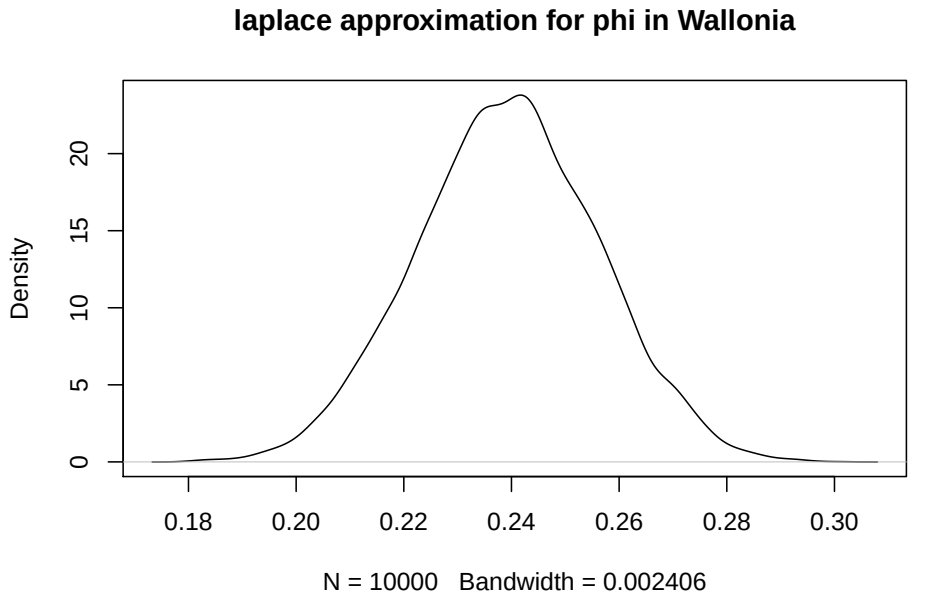
```
  round(param_cov_mat_Wal, 3)
```

```
##           mu.shape phi.shape
## mu.shape  5254.362     0.039
## phi.shape    0.039     0.000
```

```
  samples_Wal <- rmvnorm(10000, param_mean_Wal, param_cov_mat_Wal)
  plot(density(samples_Wal[, 1]), main = "laplace approximation for mu in Wallonia")
```




```
plot(density(samples_Wal[, 2]), main = "laplace approximation for phi in Wallonia")
```



5.2 Credible intervals Laplce approximation

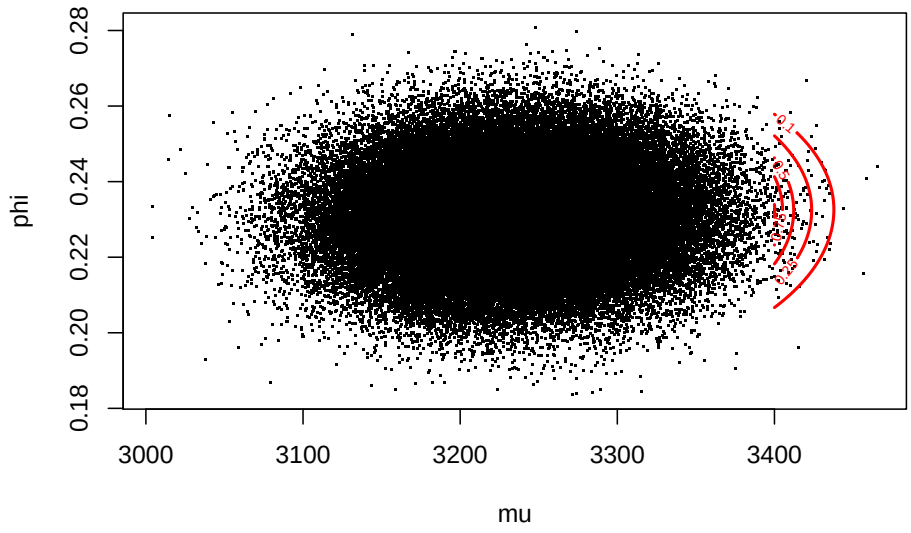
```
levels <- c(0.1, 0.25)
```

5.2.1 Flanders

The credible region is given here below:

```
plot(x = samples_Fl[, 1], y = samples_Fl[, 2], xlab = "mu", ylab = "phi",
    pch = ".")
```

```r
x1 <- seq(3400, 4000, length = 1000)
y1 <- seq(0.16, 0.26, length = 1000)
lpostFl_laplaceApprox <- function(MyMu, MyPhi) {
    # theta_l = c(mu,phi)
    MuPhi <- cbind(MyMu, MyPhi)
    dmvnorm(MuPhi, colMeans(samples_Fl), cov(samples_Fl), log = TRUE)
}

z1 <- outer(x1, y1, lpostFl_laplaceApprox)
R <- exp(z1 - max(z1))
lvls <- c(0.01, 0.25, 0.5, 0.75, 0.9)
contour(x1, y1, R, levels = exp(-0.5 * qchisq(lvls, 2)), add = T, lwd = 2,
    labels = (1 - lvls), col = "red")
```




```r
library(MASS)
library(plotly)
den3dLaplaceApprox <- kde2d(samples_Fl[, 1], samples_Fl[, 2])
plot_ly(x = den3dLaplaceApprox$x, y = den3dLaplaceApprox$y, z = den3dLaplaceApprox$z) %>%
    add_surface()
```

We needs to get the credible interval for μ. This means that we needs the marginal posterior distribution:

$$P(\mu|D) \propto \int p(\mu, \phi|D) \mathrm{d}\phi$$

It can be shown that the marginal (univariate) distribution of the bivariate Gaussian distribution $N(\mu_\theta, \Sigma)$ with $\mu_\theta = (E(\mu), E(\phi))$ and

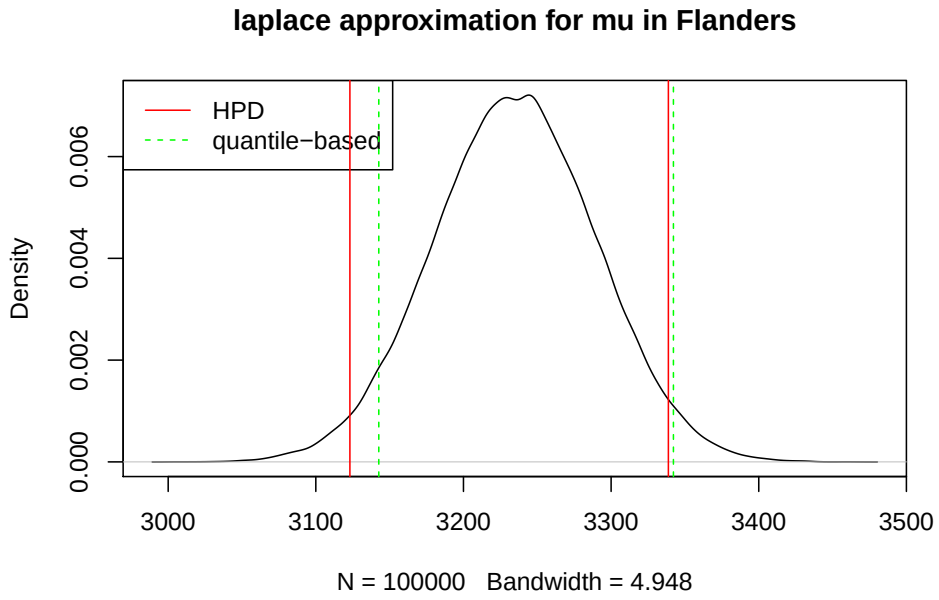
$$\Sigma = \begin{pmatrix} \sigma_\mu & \sigma_{\mu,\phi} \\ \sigma_{\phi,\mu} & \sigma_\phi \end{pmatrix}$$

also follows a normal distribution. See here. Hence, for μ, one has:

$$\mu|D{=}N(E(\mu), \sigma_\mu)$$

We needs that 95% of the marginal posterior to fall into the interval for Flanders:

```
alpha <- 0.05
marginal_posterior_mu_Fl <- rnorm(10000, param_mean_Fl[1], sqrt(param_cov_mat_Fl[1,
    1]))
# Quantile based credible intervals
QuantileCI_Laplace_Fl <- quantile(marginal_posterior_mu_Fl, p = c(alpha/1,
    1 - alpha/2))
# HPD intervals
HPDIntervals_Laplace_Fl <- HPDinterval(as.mcmc(marginal_posterior_mu_Fl),
    prob = 1 - alpha)

plot(density(samples_Fl[, 1]), main = "laplace approximation for mu in Flanders")
legend("topleft", legend = c("HPD", "quantile-based"), col = c("red", "green"),
    lty = 1:2)

abline(v = c(HPDIntervals_Laplace_Fl[1], HPDIntervals_Laplace_Fl[2]), col = "red",
    lty = 1)
abline(v = c(QuantileCI_Laplace_Fl[1], QuantileCI_Laplace_Fl[2]), col = "green",
    lty = 2)
```



6 Question 5

6.1 (a) Random walk component-wise Metropololis algorithm

One first needs starting values for $\theta_t := (\mu_t, \phi_t)$. One can take advantage of the MLE estimations of κ and λ and compute μ_0 and ϕ_0 using the relations described earlier. At

each iteration, a new candidate will be proposed. To build it, it is a good idea to use the variance-covariance matrix computed into the Laplace approximation section since it should, more or less, describe the standard deviations of the parameters of interest $\hat{\sigma}_\mu$ and$\hat{\sigma}_\phi$. A factor, f_1 and f_2 should multiply their standard deviation in order to optimize the acceptance rate (goal is 40%). Moreover, since the posterior probability is on a log-scale, the comparison of probabilities should be made taking the exponential of the difference of the candidate with the current state. In this work, 10% of the generated data is considered as burn-in.

That being, this gives the following algorithm:

1. $t = 0$: Set starting values of the parameters of interest to their MLE: $\theta_0 := (\hat{\mu}_{MLE}, \hat{\phi}_{MLE})$
2. for $t = 2, 3, ..55000$
   - Draw μ_{prop} from a normal distribution centered on value derived at $t-1$, i.e. $\mu_{\text{prop}} \sim N(\mu_{t-1}, f_1\hat{\sigma}_\mu)$. The candidate is then $\theta^{\mu}_{prop} := (\mu_{\text{prop}}, \phi_{t-1})$
   - Compute $prob_\mu = \min\left(1, \exp\left(h(\theta^{\mu}_{prop}) - h(\theta_{t-1})\right)\right)$
   - Set $\mu_t = \mu_{\text{prop}}$ with probability $prob_\mu$, $\mu_t = \mu_{t-1}$ otherwise.
   - Draw ϕ_{prop} from a normal distribution centered on value derived at $t-1$, i.e. $\phi_{\text{prop}} \sim N(\phi_{t-1}, f_2\hat{\sigma}_\phi)$. The candidate is then $\theta^{\phi}_{prop} := (\mu_{t-1}, \phi_{\text{prop}})$
   - Compute $prob = \min\left(1, \exp\left(h(\theta^{\phi}_{prop}) - h(\theta_{t-1})\right)\right)$
   - Set $\phi_t = \phi_{\text{prop}}$ with probability $prob_\phi$, $\phi_t = \phi_{t-1}$ otherwise.
   - Set $\theta_t = (\mu_t, \phi_t)$

```
metropolis_algorithm = function(M, theta, sd.propositions, factors, frequencies) {

    theta = as.vector(theta)
    sd.propositions = as.vector(sd.propositions)
    factors = as.vector(factors)
    frequencies = as.vector(frequencies)

    thetas = array(dim = c(M + 1, 2))
    thetas[1, ] = theta

    accepted = c(0, 0)

    sigma_mu = factors[2] * sd.propositions[1]
    sigma_phi = factors[2] * sd.propositions[2]

    for (i in 2:(M + 1)) {
        theta_mu = theta_phi = this_theta = thetas[i - 1, ]

        theta_mu[1] = this_theta[1] + rnorm(1, 0, sigma_mu)
        theta_phi[2] = this_theta[2] + rnorm(1, 0, sigma_phi)

        thresholds = c(min(exp(lpost2(theta_mu, frequencies) - lpost2(this_theta,
            frequencies)), 1), min(exp(lpost2(theta_phi, frequencies) -
            lpost2(this_theta, frequencies)), 1))

        accepts = runif(2) <= thresholds
```

```
            thetas[i, ] = this_theta
            if (accepts[1])
                thetas[i, 1] = theta_mu[1]
            if (accepts[2])
                thetas[i, 2] = theta_phi[2]

            accepted = accepted + as.integer(accepts)
        }

        thetas = thetas[, -0.1 * M]

        names(thetas) = c("mu", "phi")
        names(accepted) = c("mu", "phi")
        return(list(theta = thetas, accept_rate = accepted/M))
    }
```

```
    # Sigma_hat to be used: the proposed one is the one from the Laplace
    # approximation: cov(samples_Fl)
    sd_hat_of_mu <- sqrt(param_cov_mat_Fl[1, 1])
    sd_hat_of_phi <- sqrt(param_cov_mat_Fl[2, 2])
    M <- 60000
```

```
    # using starting values as Estimated_mu_Fl and Estimated_phi_Fl that
    # are chosen via MLE
    metro_fl1 = metropolis_algorithm(M, theta = c(mu = 3089.94396, phi = 0.399687),
        sd.propositions = c(mu = sd_hat_of_mu, phi = sd_hat_of_phi), factors = c(mu = 2.75,
            phi = 2.75), frequencies = Table[1, ])
```

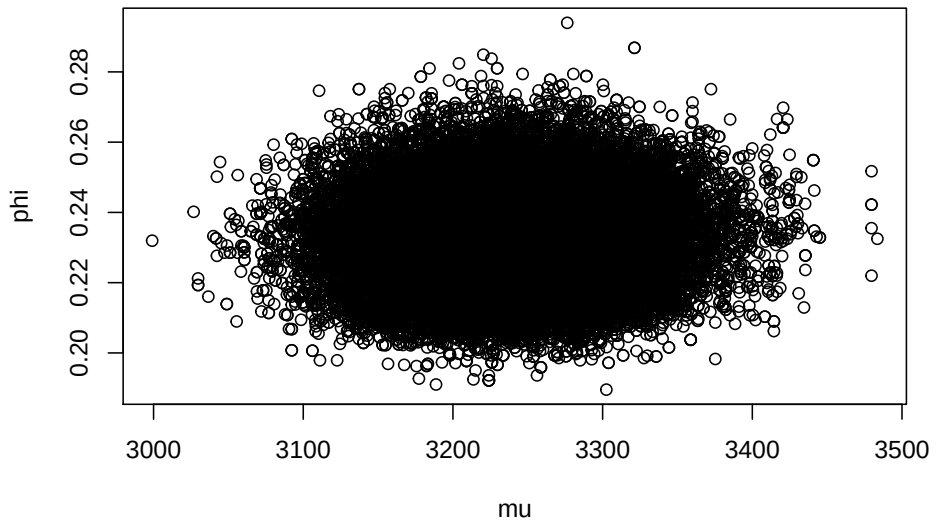
```
    acceptance_rate_mu_Metrop1 <- metro_fl1$accept_rate["mu"] * 100
    cat("Acceptance rate for mu: ", acceptance_rate_mu_Metrop1, "% \n")
```

```
## Acceptance rate for mu:  40.16333 %
```

```
    acceptance_rate_phi_Metrop1 <- metro_fl1$accept_rate["phi"] * 100
    cat("Acceptance rate for phi: ", acceptance_rate_phi_Metrop1, "% \n")
```

```
## Acceptance rate for phi:  40.23833 %
```

```
    # Remove bur-in samples
    thetasM1 <- tail(metro_fl1$theta, (-0.1 * M))

    plot(thetasM1[, 1], thetasM1[, 2], xlab = "mu", ylab = "phi")
```



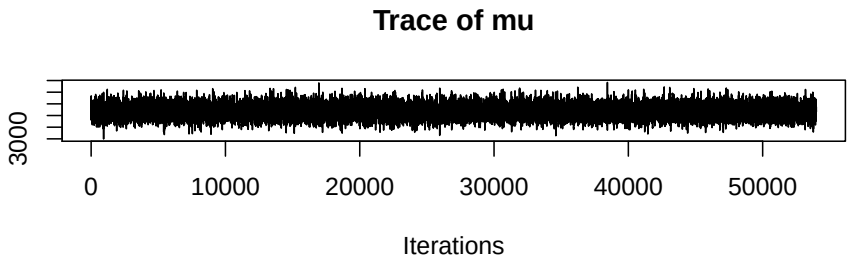

```
library(MASS)
library(plotly)
den3d <- kde2d(thetasM1[, 1], thetasM1[, 2])
plot_ly(x = den3d$x, y = den3d$y, z = den3d$z) %>%
    add_surface()
```

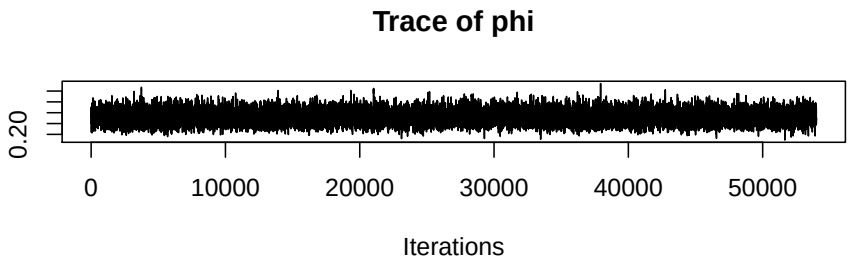
6.2 (b) diagnostic for convergence

6.2.1 Graphs analysis

The first, trivial, way of checking if the convergence occurred is looking at the plot of the generated variables:

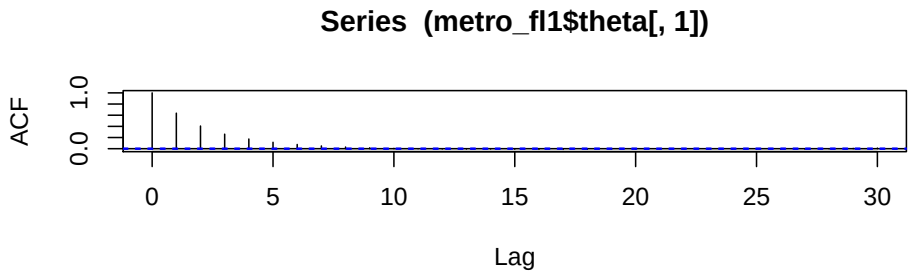
```
par(mfrow = c(2, 1))
traceplot(as.mcmc(thetasM1))
```

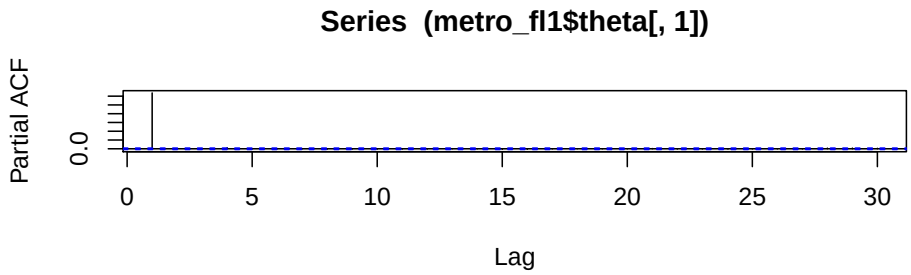




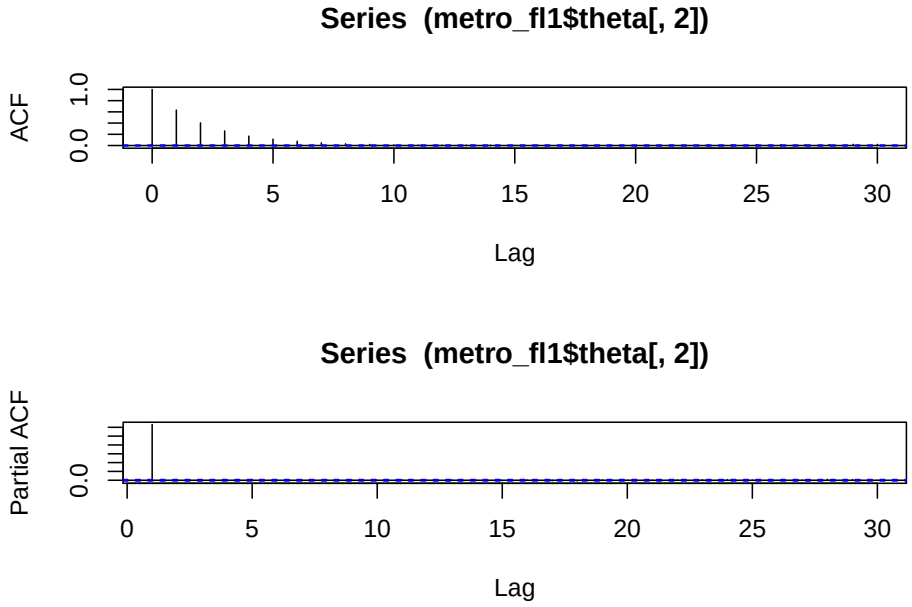
By doing so, one sees that the mixing seems pretty good. For example, the autocorrelation is significative up to not a too high order. As such, the parameter space of *theta* does not seem to be visited to slowly.

```
par(mfrow = c(2, 1))
acf((metro_fl1$theta[, 1]), lag.max = 30)
pacf((metro_fl1$theta[, 1]), lag.max = 30)
```






```
acf((metro_fl1$theta[, 2]), lag.max = 30)
pacf((metro_fl1$theta[, 2]), lag.max = 30)
```



Since the chains clearly seem to be generated with respect to an Autoregressive process (see ACF/PACF plots), one can compute effectively the effective sample size. It represents how many samples would be generated if there were no autocorrelation.

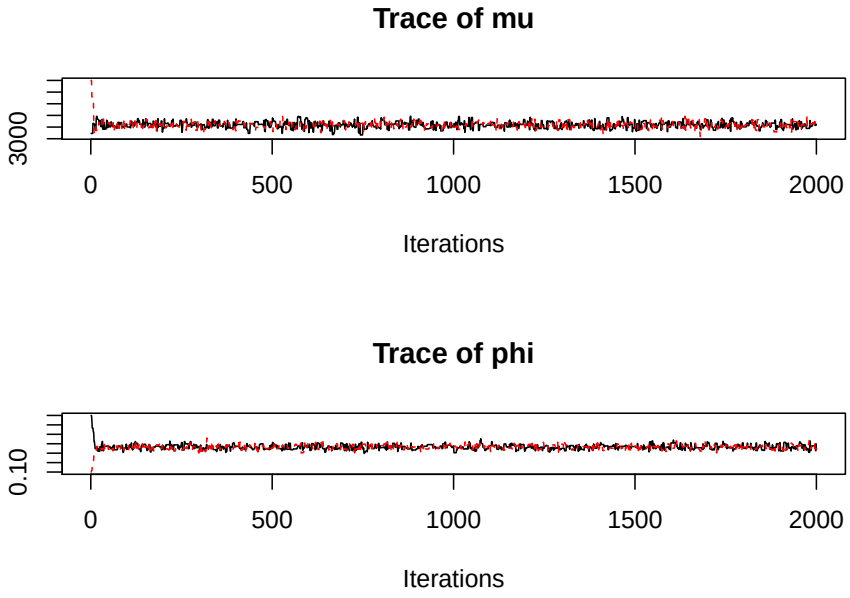
```
# out of 48 000 observations
effectiveSize(thetasM1)
```

```
##       mu      phi
## 11583.38 11943.99
```

Gelman-Rubin This method requires multiple chains. In this work, the diagnostic will be done with 2 generated chains with two different starting values. As a second chain, more exotic values (but still possible , a priori) are proposed for initialization: 4000 for μ_1 and 0.1 for ϕ_1. After having run the second chain, the traceplot of both chain is plotted with respect to the first 1000 generations.

```
metro_fl2 = metropolis_algorithm(60000, theta = c(mu = 4000, phi = 0.1),
    sd.propositions = c(mu = sd_hat_of_mu, phi = sd_hat_of_phi), factors = c(mu = 2.75,
        phi = 2.75), frequencies = Table[1, ])
```

```
par(mfrow = c(2, 1))
traceplot(list(mcmc(metro_fl1$theta[1:2000, ]), mcmc(metro_fl2$theta[1:2000,
    ])))
```



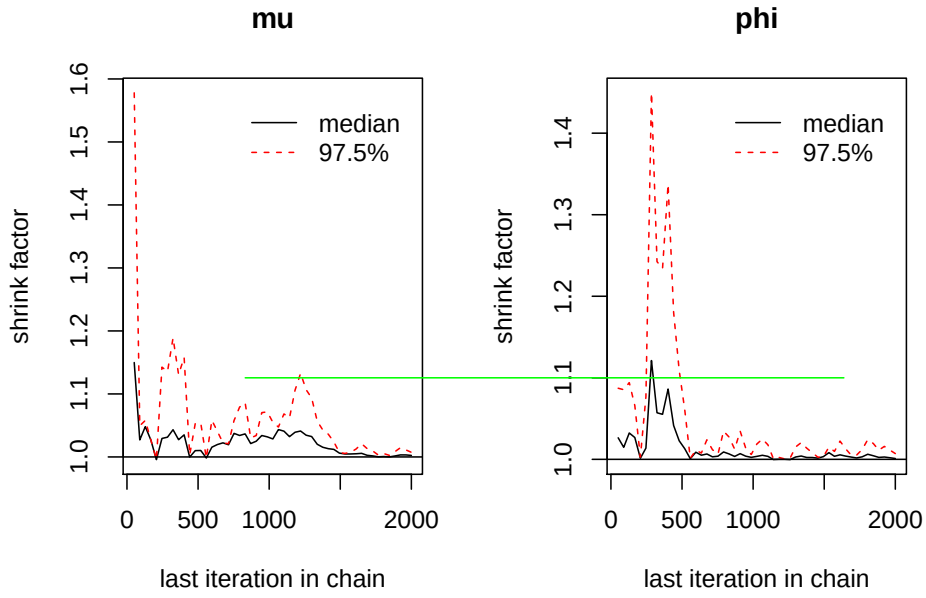
It seems that convergence occurs rather quickly. After a early generations, the chains seem to have similar mean variance. This can be more formally assessed with the Gelman-Rubin statistic that one wishes to be smaller that, say, 1.1. Here below is shown the estimated statistic along with its upper bond.

```
  gelman.diag(list(mcmc(metro_fl1$theta[1:2000, ]), mcmc(metro_fl2$theta[1:2000,
      ])))
```

```
## Potential scale reduction factors:
##
##     Point est. Upper C.I.
## mu           1       1.01
## phi          1       1.01
##
## Multivariate psrf
##
## 1
```

Those are clearly below, which is a good sign of convergence. To see when convergence may have occurred, one could look at the statistic with respect to the number of generations. Here below are presented the statistics, up to 2000 samples:

```
  gelman.plot(list(mcmc(metro_fl1$theta[1:2000, ]), mcmc(metro_fl2$theta[1:2000,
      ])))
  abline(h = 1.1, col = "green")
```



Accordingly, the convergence seems to occurs after roughly 1500 observations for both parameters.

6.2.2 Geweke diagnostic

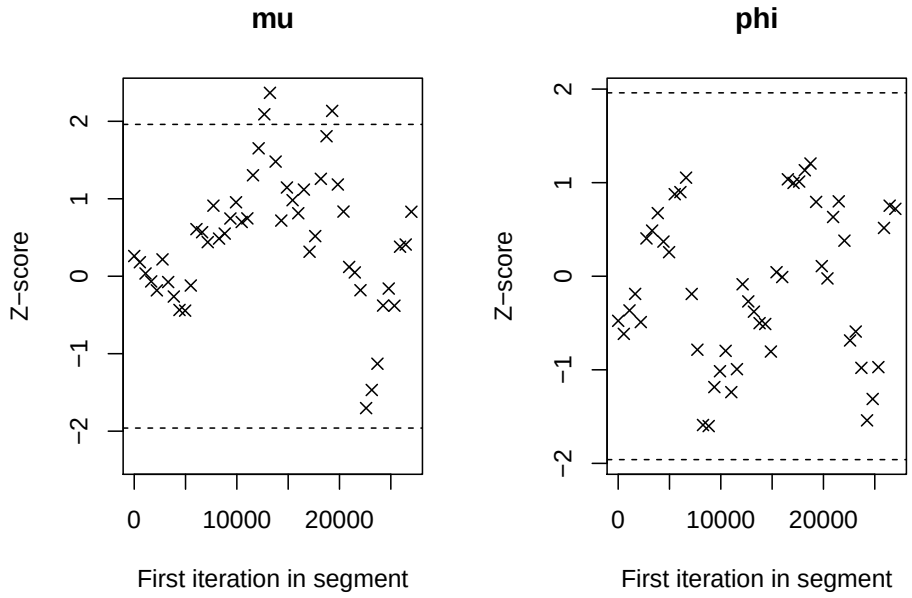
The Geweke diagnostic is useful in the sense that it allows for only one chain to be generated. The chain will be separated into two part: the first accounts for the first 10% of the data while the other accounts for the last 50% of the chain. Under the hypothesis of convergence, the mean of both subsets should be similar. Hence, one can construct a corresponding two means test. The Z-scores are given given here below for μ and ϕ, respectively:

```
geweke.diag(mcmc((thetasM1)))
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##      mu     phi
##  0.2610 -0.4767
```

To strengthen the belief of convergence, one can successively discard larger numbers of iterations from the beginning of the chain. From there, the z-score is successively computed and presented here below:

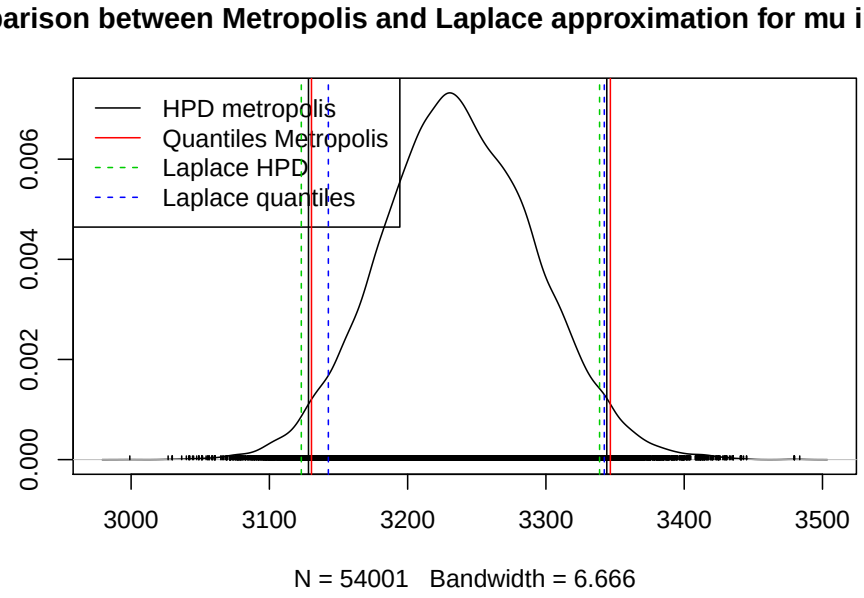
```
par(mfrow = c(2, 1))
geweke.plot(mcmc((thetasM1)), nbins = 50)
```



Except for one in ϕ, the hypothesis of same mean in never rejected. There is still no proof of non convergence of the generated chains.

6.3 (c) Credible intervals for mu1

```
## FOR mu in Flanders

HPDmu_metrop_Fl <- HPDinterval(as.mcmc((thetasM1)))[1, ]
CImu_Fl <- quantile(thetasM1[, 1], probs = c(alpha/2, 1 - alpha/2))
densplot(as.mcmc((thetasM1[, 1])), main = "Comparison between Metropolis and Laplace approx
legend("topleft", legend = c("HPD metropolis", "Quantiles Metropolis",
    "Laplace HPD", "Laplace quantiles"), col = 1:4, lty = c(1, 1, 2, 2))
abline(v = c(HPDmu_metrop_Fl[1], HPDmu_metrop_Fl[2]), col = 1)
abline(v = CImu_Fl, col = 2)
abline(v = c(HPDIntervals_Laplace_Fl[1], HPDIntervals_Laplace_Fl[2]), col = 3,
    lty = 2)
abline(v = c(QuantileCI_Laplace_Fl[1], QuantileCI_Laplace_Fl[2]), col = 4,
    lty = 2)
```




```
# Encore à faire, là ça fait de la merde
library(kableExtra)
# kable(as.data.frame(matrix(data=cbind(HPDmu_metrop_Fl,
# CImu_Fl,HPDIntervals_Laplace_Fl, QuantileCI_Laplace_Fl) , nrow = 2)))
```

7 Question 6 : same question as 5 but with JAGS

```
require(R2WinBUGS)
require(coda)
require(rjags)
```

```
metro_model = function() {

    kappa <- 1/phi
    lambda <- 1/(phi * mu)

    for (i in 1:10) {
        pi[i] <- pgamma(intervals[i + 1], kappa, lambda) - pgamma(intervals[i],
            kappa, lambda)
    }

    # Likelihood
    y ~ dmulti(pi, n)

    # Priors
    mu ~ dnorm(3000, pow(306.12, -2))
    phi ~ dunif(0, 10)

}

model.file = "resources/metropolis.bug"
```

```
  write.model(metro_model, model.file)
```

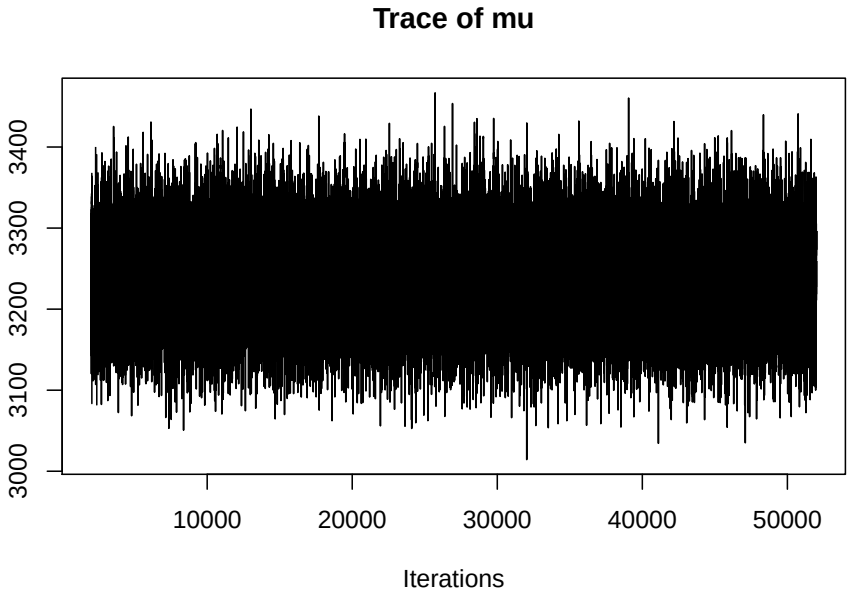
```
  model.file = "resources/metropolis.bug"

  # Test with Flanders
  jags_fl = jags.model(file = model.file, inits = list(list(mu = 3090, phi = 0.4)),
      data = list(n = 803, intervals = intervals, y = Table[1, ]), n.chains = 1)
```

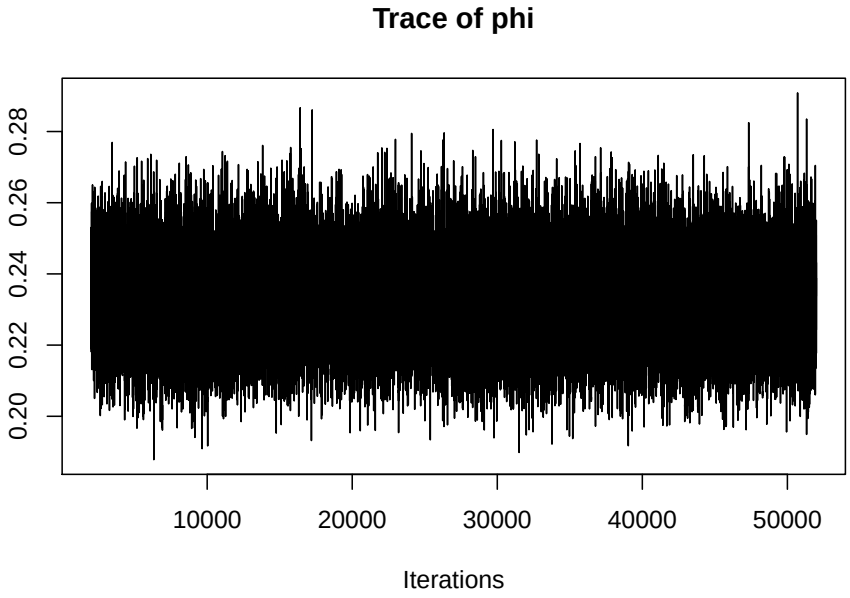
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1
##    Unobserved stochastic nodes: 2
##    Total graph size: 48
##
## Initializing model
```

```
  update(jags_fl, 1000)

  out_fl = coda.samples(model = jags_fl, c("mu", "phi"), n.iter = 50000)
  out_fl.matrix = as.matrix(out_fl)
  summary(out_fl.matrix)
```

```
##        mu             phi
##  Min.   :3014   Min.   :0.1878
##  1st Qu.:3199   1st Qu.:0.2240
##  Median :3235   Median :0.2321
##  Mean   :3236   Mean   :0.2325
##  3rd Qu.:3274   3rd Qu.:0.2404
##  Max.   :3467   Max.   :0.2909
```

```
  traceplot(out_fl)
```





A Appendix

A.1 Figures

A.2 Code

Note

For reproducibility purposes, the complete R project containing the source code and the results is available on github.com.

```
# remotes::install_github('yihui/formatR')
library(formatR)
```