



Faculté des sciences

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	2
2	Question 1	4
2.1	(a) Theoretical probability	4
2.2	(b) Theoretical expression for the likelihood	5
2.3	Taking approximation	6
2.4	Not taking approximation but the CDF differences	6
2.5	NOT taking approximation but with PDF definitions:	7
3	Question 2 : Priors	7
4	Question 3	7
4.1	Question 3a: posterior	7
4.1.1	With approximation	7
4.1.2	Without approximation	8
4.2	3.b	8
5	4.	10
5.1	Estimation	10
5.1.1	Flanders	10
5.1.2	Wallonia	11
5.2	Credible intervals Laplace approximation	12
5.2.1	Flanders	12
6	Question 5	14
6.1	(a) Random walk component-wise Metropolis algorithm	14
6.2	(b) diagnostic for convergence	17
6.2.1	Graphs analysis	17
6.2.2	Geweke diagnostic	20
6.3	(c) Credible intervals for mu1	21
7	Question 6 : same question as 5 but with JAGS	22
8	Question 7: same but for Wallonia	25
9	Question 8: Credible interval for means difference	27
A	Appendix	28
A.1	Figures	28
A.2	Code	28

1 Introduction

```
library(EnvStats)
library(mnormmt)
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
F1 <- 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_F1 <- egamma(F1)
Estimated_kappa_F1 <- Estimation_F1$parameters["shape"]
Estimated_lambda_F1 <- 1/Estimination_F1$parameters["scale"]
Estimated_mu_F1 <- Estimated_kappa_F1/Estimated_lambda_F1
Estimated_mu_F1

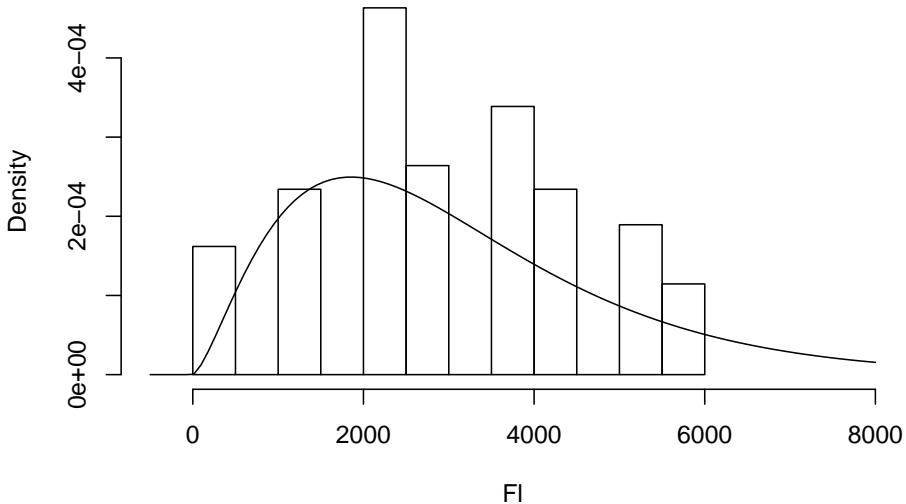
##      shape
## 3089.944

Estimated_phi_F1 <- 1/Estimated_kappa_F1
Estimated_phi_F1

##      shape
## 0.399687

estimGamma_F1 <- rgamma(10000, Estimated_kappa_F1, Estimated_lambda_F1)
hist(F1, probability = T, xlim = c(-500, 8000))
curve(dgamma(x, Estimated_kappa_F1, Estimated_lambda_F1), add = TRUE)
```

Histogram of FI



```
# 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$parameters["shape"]
Estimated_lambda_Wal <- 1/Estimination_Wall$parameters["scale"]
Estimated_mu_Wal <- Estimated_kappa_Wal/Estimated_lambda_Wal
Estimated_mu_Wal

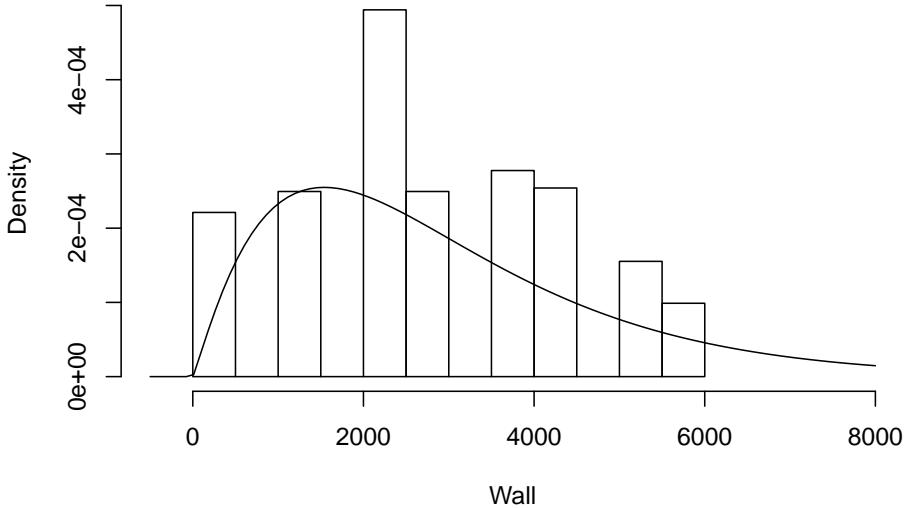
##      shape
## 2914.882

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)
```

Histogram of Wall



```

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 Wallonia, 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:

$$\begin{aligned} \text{shape: } \kappa &= \frac{1}{\phi} \\ \text{rate: } \lambda &= \frac{1}{\phi \mu} \end{aligned}$$

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:

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

Using CDF writing, for a region k ,

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

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

In terms of μ and ϕ

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

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

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 $P := (p_1, \dots, p_{10})$ and $X := (X_1, \dots, X_{10})$:

$$X|P \sim \text{Mul}(N, P) = \frac{x!}{x_1! \dots x_{10}!} p_1^{x_1} \times \dots \times p_{10}^{x_{10}} \text{ when } \sum_{j=1}^{10} p_j = 1$$

$$= 0 \text{ otherwise}$$

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. $x_{\text{Flanders},3} = (1500 + 1800)/2 = 1650$? One can approximate that with $f(x_i)\Delta$ where Δ is the unit of measurement.

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

This gives for the likelihood:

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

2.4 Not taking approximation but the CDF differences

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

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

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

Writing in terms of μ_k and ϕ_k :

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

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 distribution 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

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

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

- 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 two intervals

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

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

$$\text{prior: } P(\mu_k, \phi_k) = P(\mu_k) P(\phi_k)$$

$$\propto \sigma_0^{-1/2} \exp \left(-\frac{1}{2\sigma_0^2} (\mu - \mu_0)^2 \right) 1_{0,10}$$

4 Question 3

4.1 Question 3a: posterior

4.1.1 With approximation

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

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

4.1.2 Without approximation

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

4.2 3.b

the log likelihood is, up to an additive constant:

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

so the log-posterior is:

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

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

```

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)
  # initialisation
  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 être
  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_F1, Estimated_phi_F1), t(Table[1, ]))

##      shape
## -1755.997

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

##      shape
## -956.3546

```

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%20Approximation>

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)

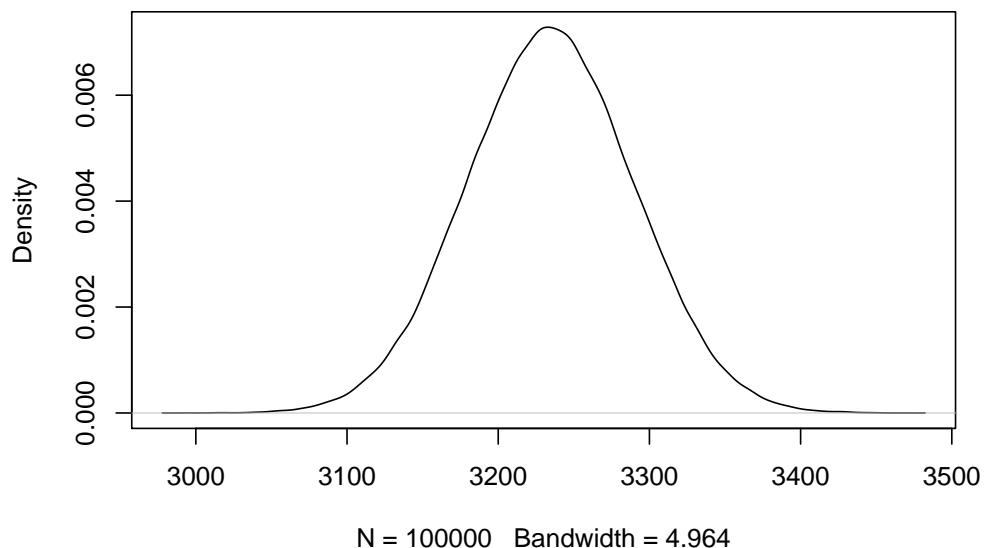
## mu.shape phi.shape
## 3234.50      0.23

round(param_cov_mat_Fl, 3)

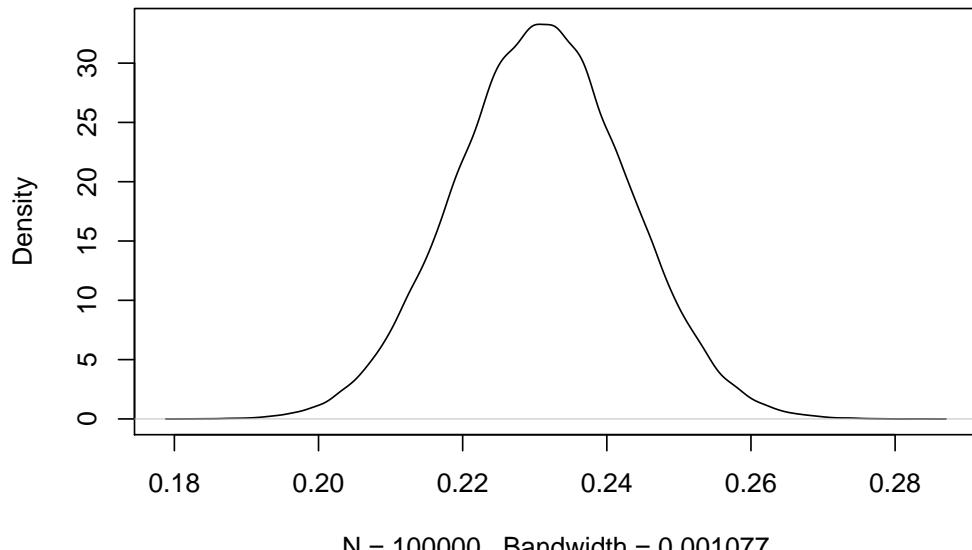
##          mu.shape phi.shape
## mu.shape 3023.270    0.025
## phi.shape     0.025    0.000

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

laplace approximation for mu in Flanders



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

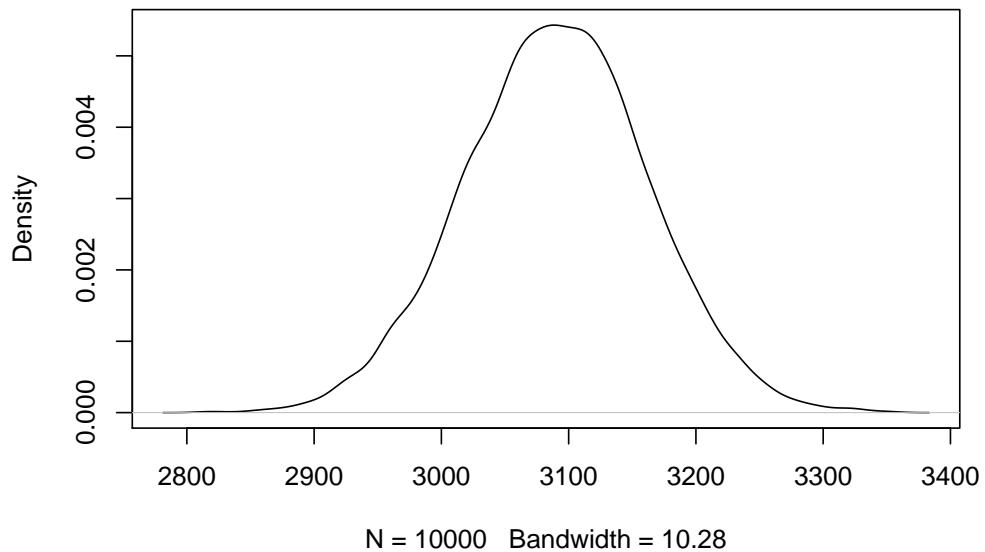
## mu.shape phi.shape
## 3089.32      0.24

round(param_cov_mat_Wal, 3)

##           mu.shape phi.shape
## mu.shape  5252.793   0.039
## phi.shape    0.039   0.000

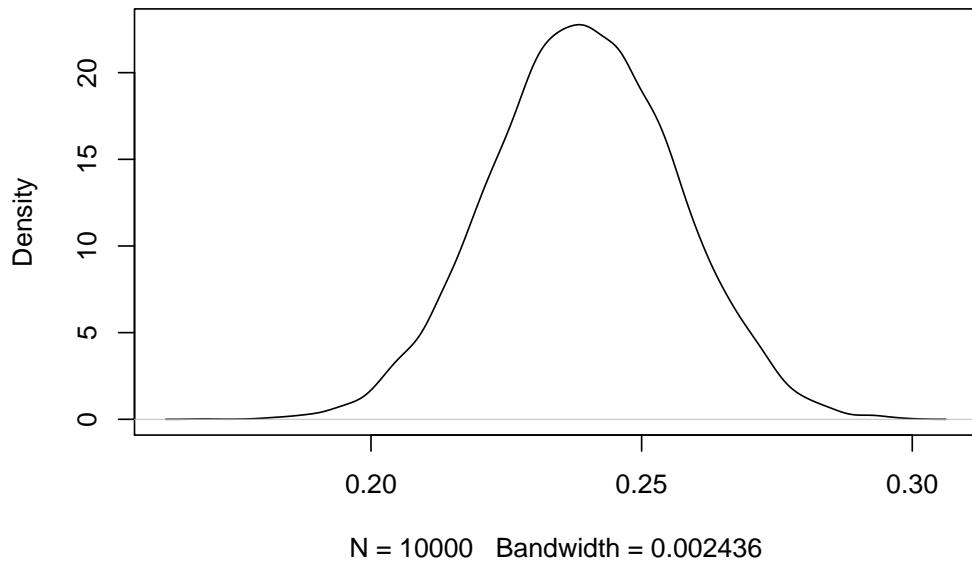
samples_Wal <- rmvnorm(10000, param_mean_Wal, param_cov_mat_Wal)
plot(density(samples_Wal[, 1]), main = "laplace approximation for mu in Wallonia")
```

laplace approximation for mu in Wallonia



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

laplace approximation for phi in Wallonia



5.2 Credible intervals Laplace approximation

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

5.2.1 Flanders

The credible region is given here below:

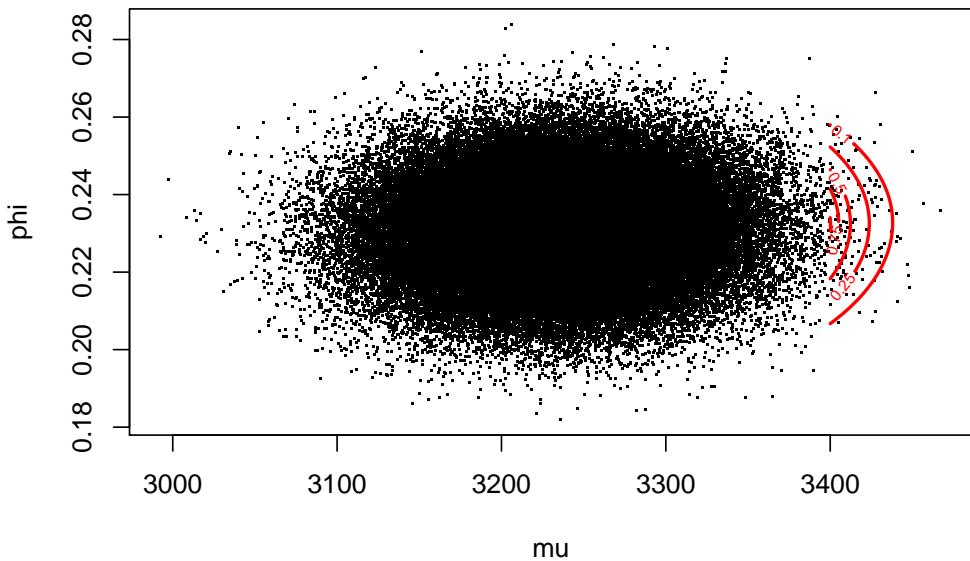
```
plot(x = laplace_samples_Fl[, 1], y = laplace_samples_Fl[, 2], xlab = "mu",
      ylab = "phi", pch = ".")
x1 <- seq(3400, 4000, length = 1000)
y1 <- seq(0.16, 0.26, length = 1000)
lpostFl_laplaceApprox <- function(MyMu, MyPhi) {
```

```

# theta_l = c(mu, phi)
MuPhi <- cbind(MyMu, MyPhi)
dmvnorm(MuPhi, colMeans(laplace_samples_F1), cov(laplace_samples_F1),
        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")

```



```

library(MASS)
library(plotly)
den3dLaplaceApprox <- kde2d(laplace_samples_F1[, 1], laplace_samples_F1[, 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:

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

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:

```

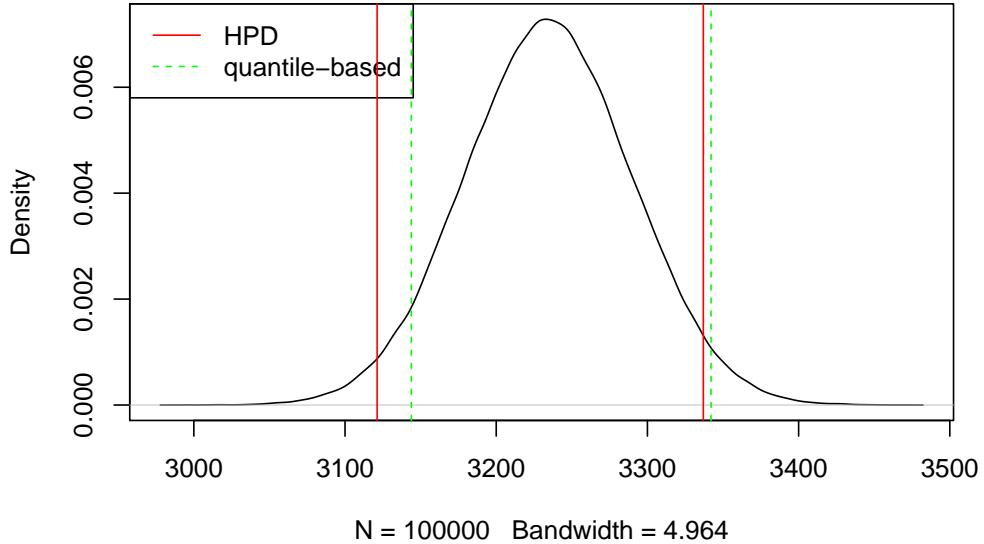
alpha <- 0.05
marginal_posterior_mu_Fl <- rnorm(10000, param_mean_Fl[1], sqrt(param_cov_mat_Fl[1,
  1]))
# Quantile based credible intervals
QuantileCI_Laplace_Fl <- quantile(marginal_posterior_mu_Fl, p = c(alpha/1,
  1 - alpha/2))
# HPD intervals
HPDIntervals_Laplace_Fl <- HPDinterval(as.mcmc(marginal_posterior_mu_Fl),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

plot(density(laplace_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)

```

laplace approximation for mu in Flanders



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 $t = 2, 3, \dots, 55000$
 - Draw μ_{prop} from a normal distribution centered on value derived at $t - 1$, i.e. $\mu_{prop} \sim N(\mu_{t-1}, f_1\hat{\sigma}_\mu)$. The candidate is then $\theta_{prop}^\mu := (\mu_{prop}, \phi_{t-1})$
 - Compute $prob_\mu = \min(1, \exp(h(\theta_{prop}^\mu) - h(\theta_{t-1}))$
 - Set $\mu_t = \mu_{prop}$ 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. $\phi_{prop} \sim N(\phi_{t-1}, f_2\hat{\sigma}_\phi)$. The candidate is then $\theta_{prop}^\phi := (\mu_{t-1}, \phi_{prop})$
 - Compute $prob = \min(1, \exp(h(\theta_{prop}^\phi) - h(\theta_{t-1}))$
 - Set $\phi_t = \phi_{prop}$ 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(lpost(theta_mu, frequencies) - lpost(this_theta,
      frequencies)), 1), min(exp(lpost(theta_phi, frequencies) -
      lpost(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(laplace_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_f1 = 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_f1$accept_rate["mu"] * 100
cat("Acceptance rate for mu: ", acceptance_rate_mu_Metrop1, "% \n")

## Acceptance rate for mu: 40.39 %

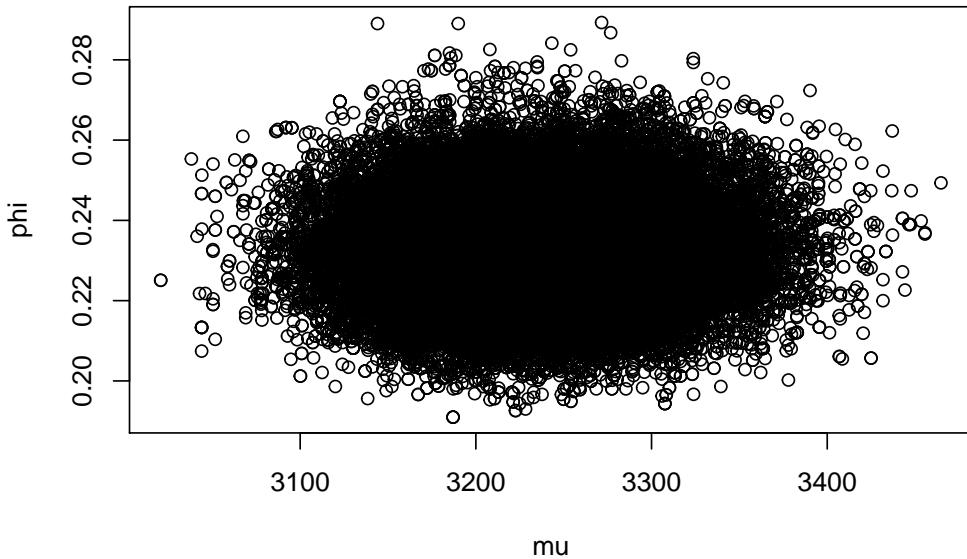
acceptance_rate_phi_Metrop1 <- metro_f1$accept_rate["phi"] * 100
cat("Acceptance rate for phi: ", acceptance_rate_phi_Metrop1, "% \n")

## Acceptance rate for phi: 40.13 %

# Remove bur-in samples
thetasM1 <- tail(metro_f1$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

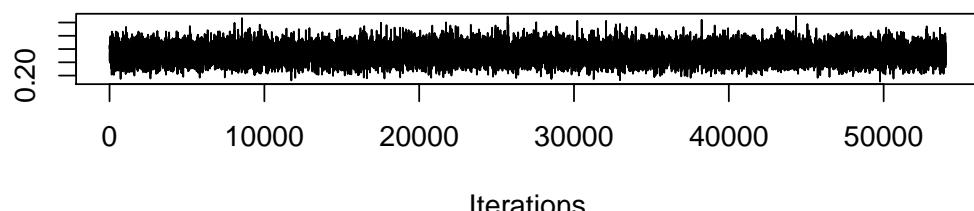
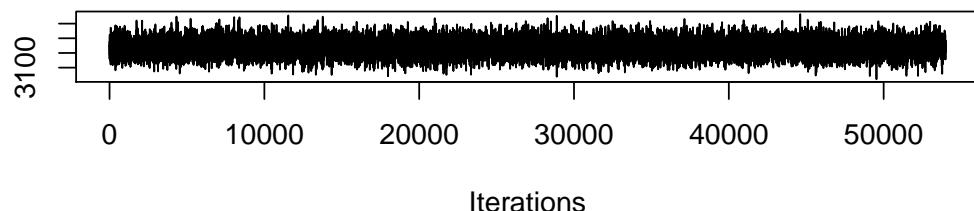
6.2.1 Graphs analysis

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))

```



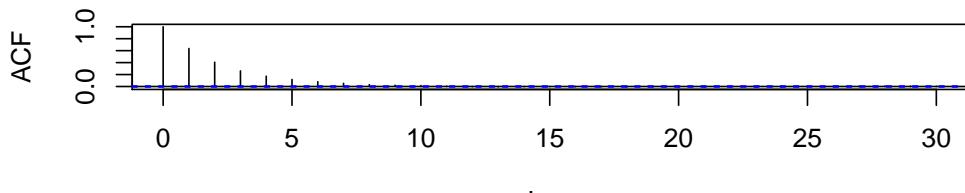
By doing so, one sees that the mixing seems pretty good. For example, the autocorrelation is significative up to not a too high order. As such, the parameter space of *theta* does not seem to be visited to slowly.

```

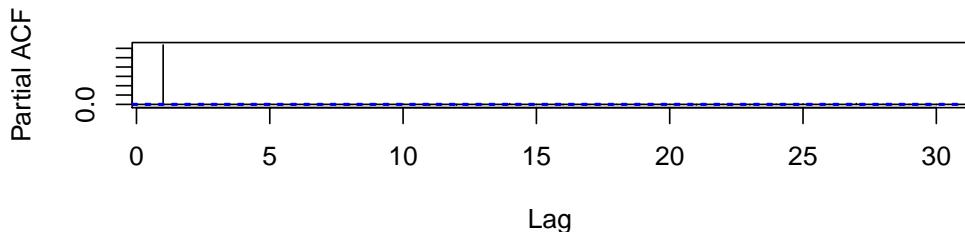
par(mfrow = c(2, 1))
acf((metro_f11$theta[, 1]), lag.max = 30)
pacf((metro_f11$theta[, 1]), lag.max = 30)

```

Series (metro_f1\$theta[, 1])

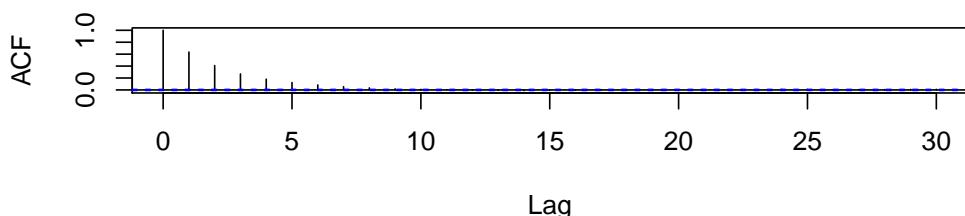


Series (metro_f1\$theta[, 1])

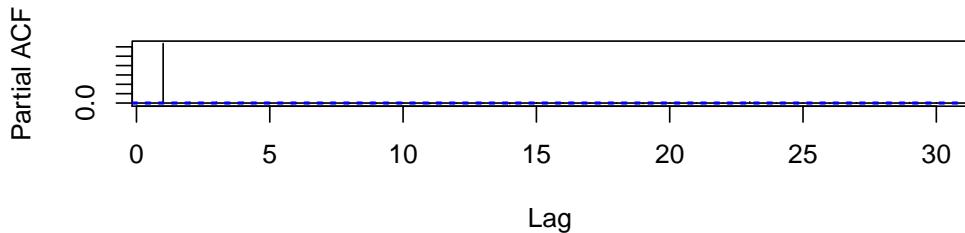


```
acf((metro_f1$theta[, 2]), lag.max = 30)
pacf((metro_f1$theta[, 2]), lag.max = 30)
```

Series (metro_f1\$theta[, 2])



Series (metro_f1\$theta[, 2])



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.

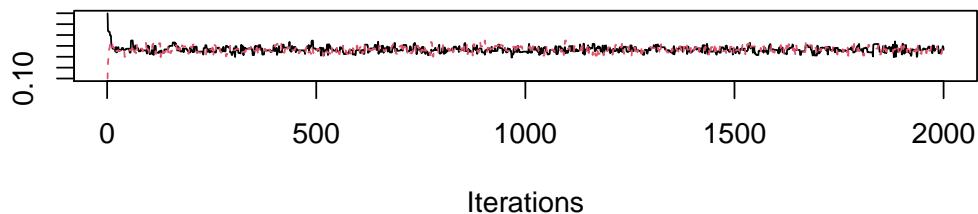
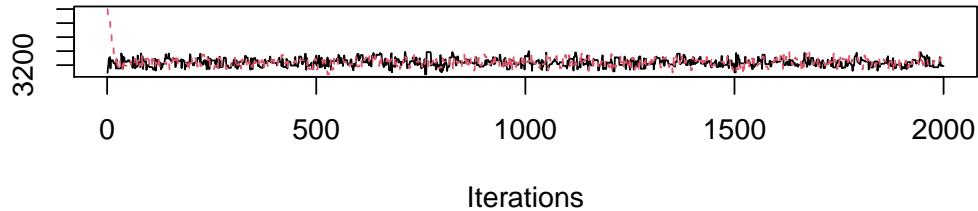
```
# out of 48 000 observations
effectiveSize(thetasM1)

##      var1      var2
## 12080.57 11544.19
```

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_f12 = 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_f11$theta[1:2000, ]), mcmc(metro_f12$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 than, say, 1.1. Here below is shown the estimated statistic along with its upper bond.

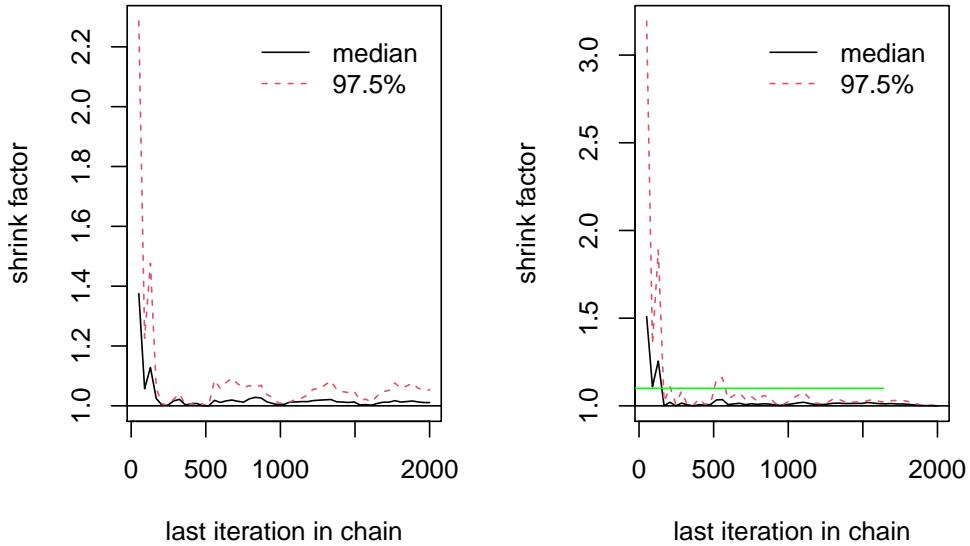
```
gelman.diag(list(mcmc(metro_f11$theta[1:2000, ]), mcmc(metro_f12$theta[1:2000,
])))

## Potential scale reduction factors:
## 
##      Point est. Upper C.I.
## [1,]     1.01     1.05
## [2,]     1.00     1.00
## 
##      Multivariate psrf
## 
##     1.01
```

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_f11$theta[1:2000, ]), mcmc(metro_f12$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

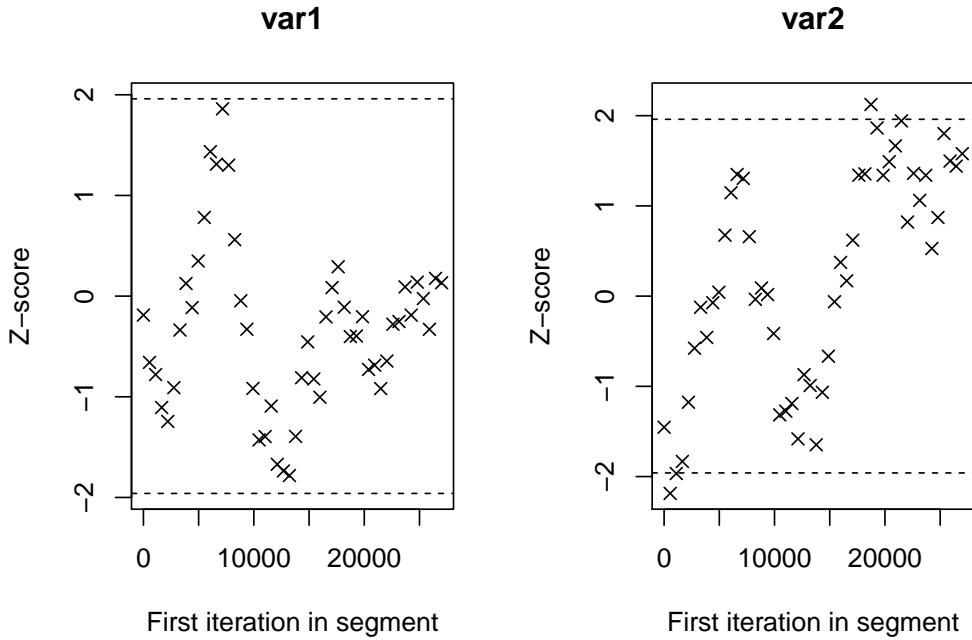
The Geweke diagnostic is useful in the sense that it allows for only one chain to be generated. The chain will be separated into two part: the first accounts for the first 10% of the data while the other accounts for the last 50% of the chain. Under the hypothesis of convergence, the mean of both subsets 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
##
##      var1      var2
## -0.1884 -1.4532
```

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)
```



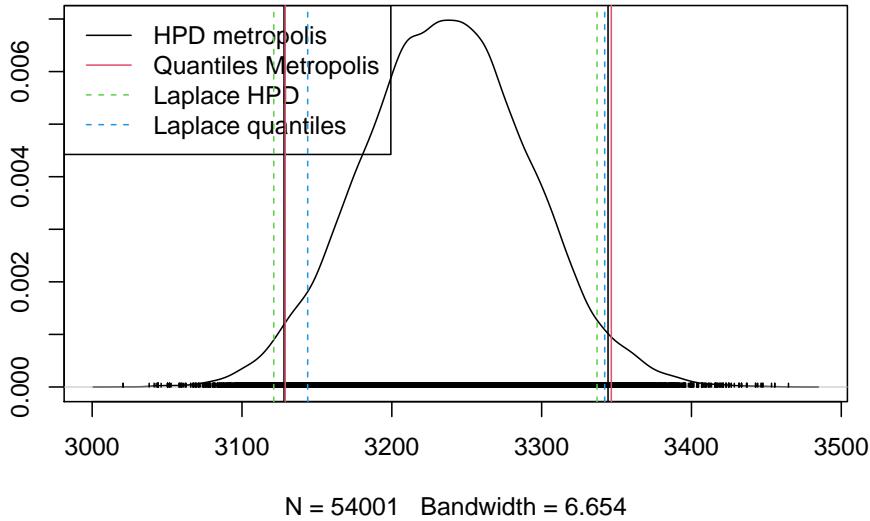
Except for one in ϕ , the hypothesis of same mean is never rejected. There is still no proof of non convergence of the generated chains.

6.3 (c) Credible intervals for mu1

```
## FOR mu in Flanders

HPDmu_metrop_F1 <- HPDinterval(as.mcmc((thetasM1)))[1, ]
CImu_F1 <- quantile(thetasM1[, 1], probs = c(alpha/2, 1 - alpha/2))
densplot(as.mcmc((thetasM1[, 1])), main = "Comparison between Metropolis and Laplace ap-
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_F1[1], HPDmu_metrop_F1[2]), col = 1)
abline(v = CImu_F1, col = 2)
abline(v = c(HPDIntervals_Laplace_F1[1], HPDIntervals_Laplace_F1[2]), col = 3,
  lty = 2)
abline(v = c(QuantileCI_Laplace_F1[1], QuantileCI_Laplace_F1[2]), col = 4,
  lty = 2)
```

Comparison between Metropolis and Laplace approximation for mu in Fl



```
# 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 = 3089.94396,
    phi = 0.399687)), 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)

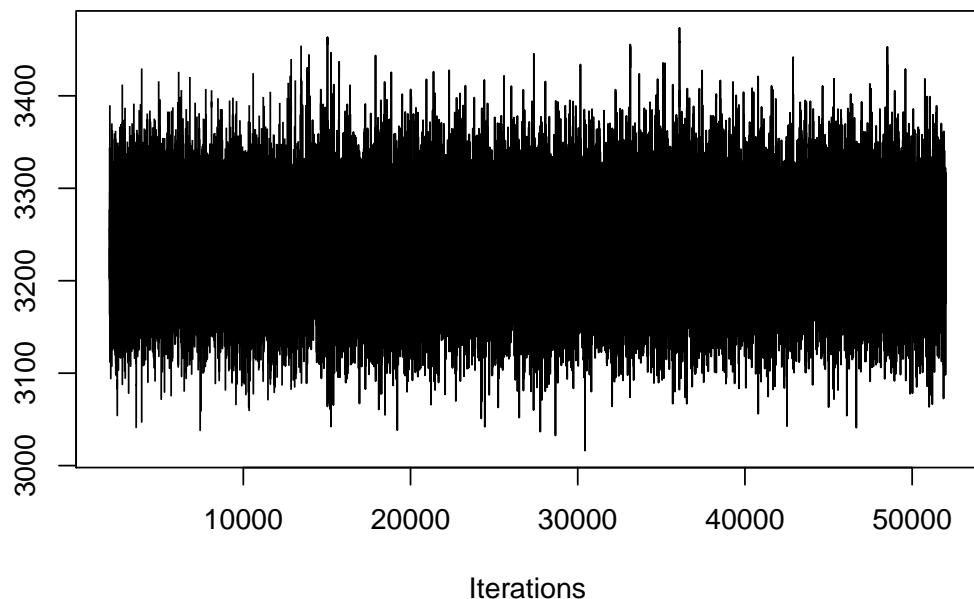
out_fl = coda.samples(model = jags_fl, c("mu", "phi"), n.iter = 50000)
out_fl.matrix = as.matrix(out_fl)
summary(out_fl.matrix)

##          mu            phi
##  Min. :3016  Min. :0.1872
##  1st Qu.:3199  1st Qu.:0.2242
##  Median :3236  Median :0.2322
##  Mean   :3236  Mean   :0.2325
##  3rd Qu.:3274  3rd Qu.:0.2403
##  Max.   :3474  Max.   :0.2894

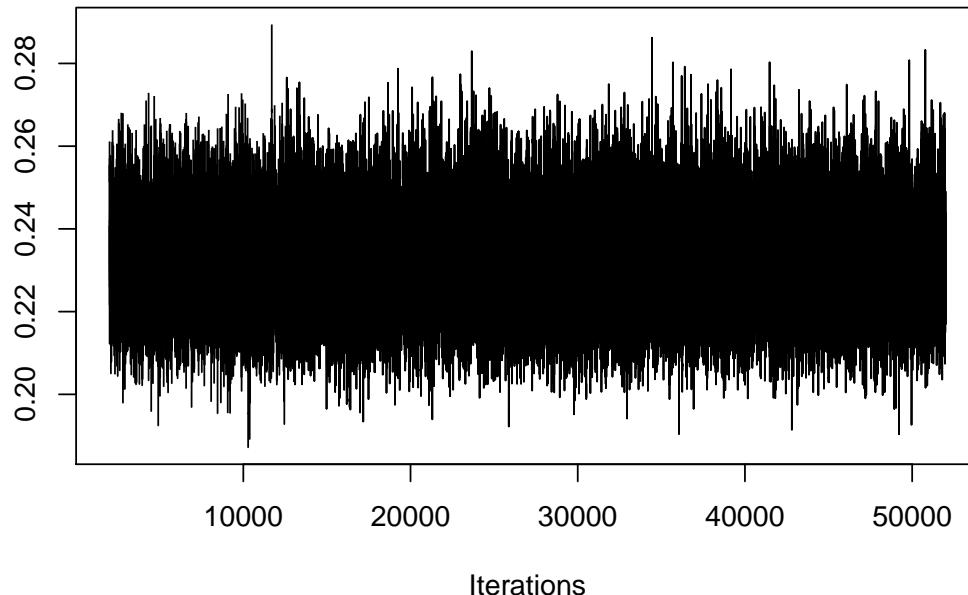
```

```
traceplot(out_fl)
```

Trace of mu

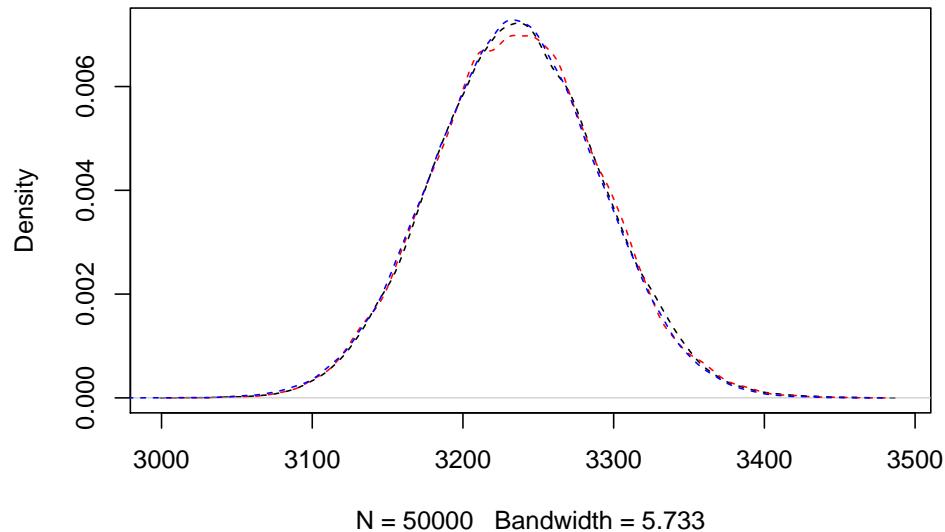


Trace of phi



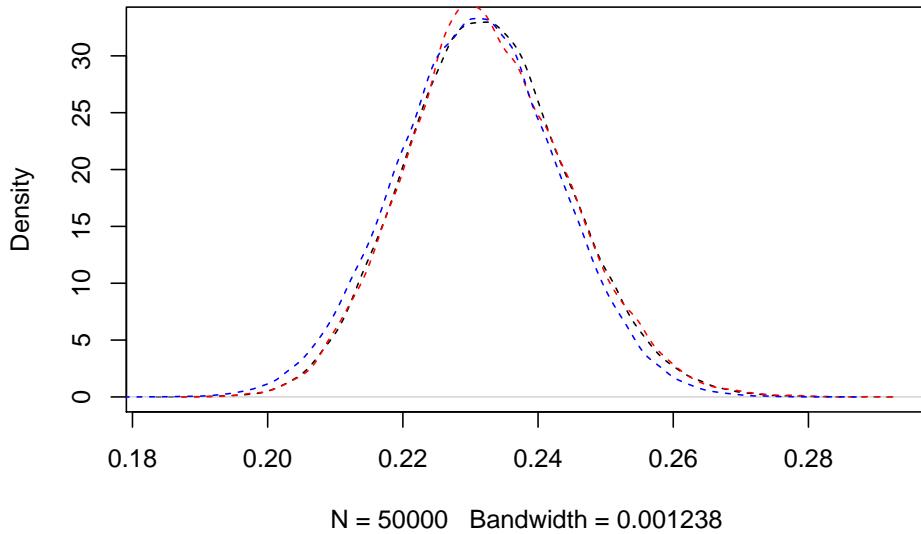
```
plot(density(out_fl.matrix[, "mu"]), main = "Comparison R metropolis, Laplace and JAGS
lty = 2)
lines(density(thetasM1[, 1]), col = "red", lty = 2)
lines(density(laplace_samples_F1[, 1]), col = "blue", lty = 2)
```

Comparison R metropolis, Laplace and JAGS distribution for mu



```
plot(density(out_fl.matrix[, "phi"]), main = "Comparison R metropolis, Laplace and JAGS
lty = 2)
lines(density(thetasM1[, 2]), col = "red", lty = 2)
lines(density(laplace_samples_F1[, 2]), col = "blue", lty = 2)
```

Comparison R metropolis, Laplace and JAGS distribution for phi



8 Question 7: same but for Wallonia

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

# Test with Wallonia
jags_wal = jags.model(file = model.file, inits = list(list(mu = 2914.882,
    phi = 0.471615)), data = list(n = 425, intervals = intervals, y = Table[2,
]), 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_wal, 1000)

out_wal = coda.samples(model = jags_wal, c("mu", "phi"), n.iter = 50000)
out_wal.matrix = as.matrix(out_wal)
summary(out_wal)

## Iterations = 2001:52000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 50000
##
```

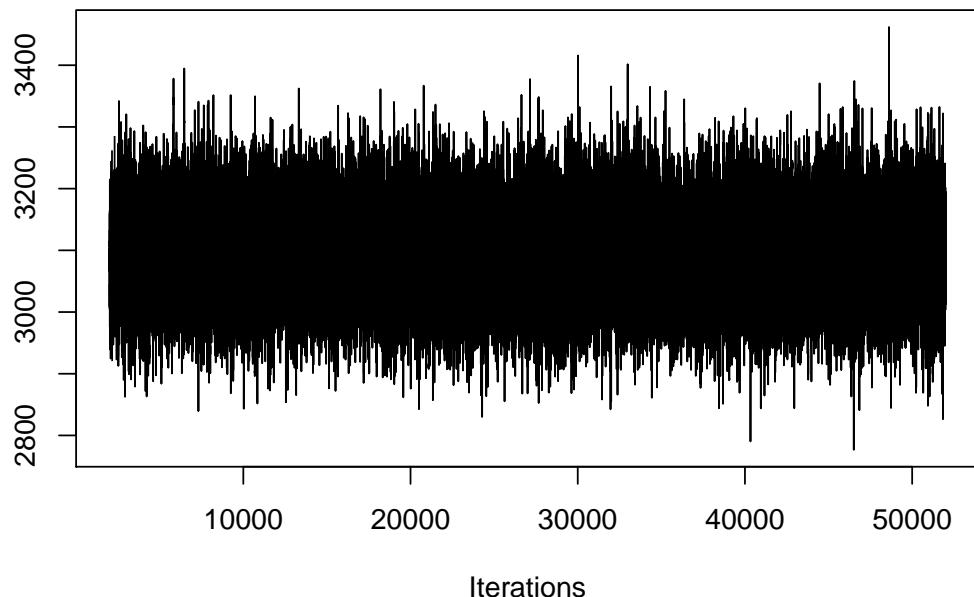
```

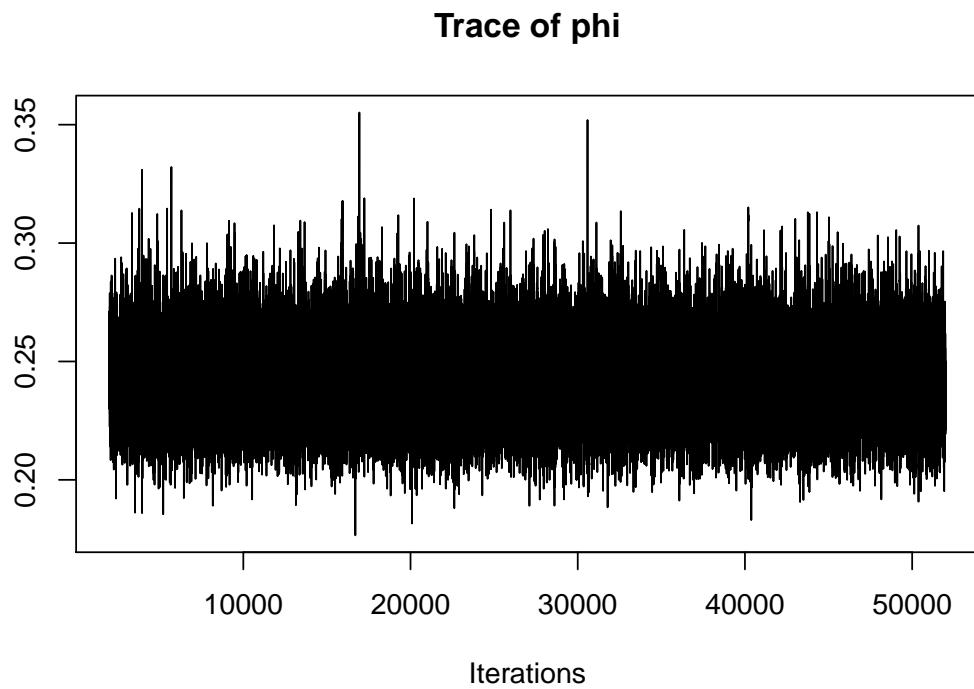
## 1. Empirical mean and standard deviation for each variable,
##     plus standard error of the mean:
##
##          Mean      SD  Naive SE Time-series SE
## mu   3092.5741 73.32658 3.279e-01       4.346e-01
## phi    0.2421  0.01741 7.786e-05       9.957e-05
##
## 2. Quantiles for each variable:
##
##          2.5%     25%     50%     75%    97.5%
## mu   2951.7895 3042.51 3091.2834 3141.0315 3240.3404
## phi    0.2103    0.23    0.2413    0.2532    0.2784

```

```
traceplot(out_wal)
```

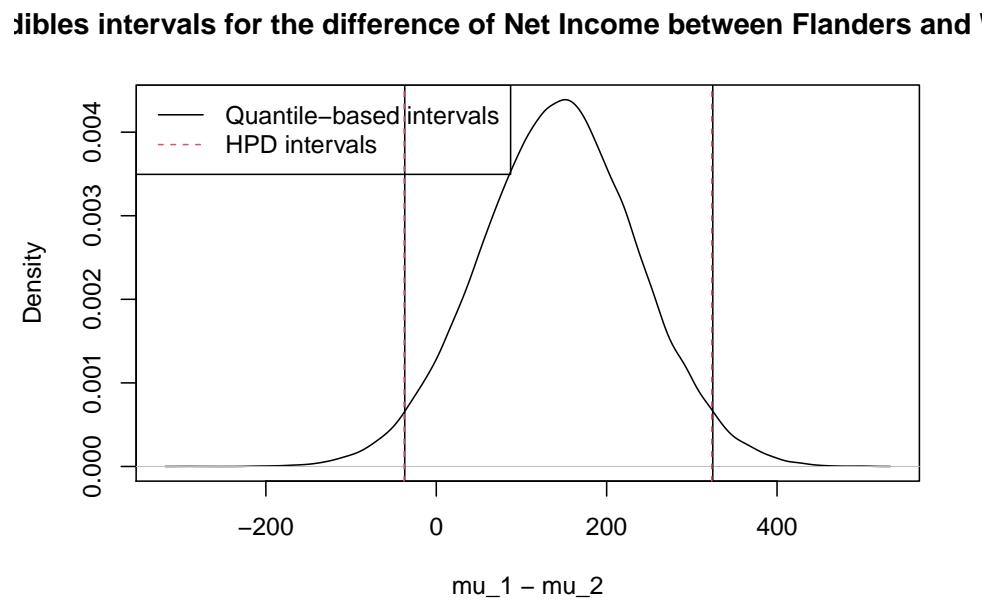
Trace of mu





9 Question 8: Credible interval for means difference

```
diff_of_mu <- out_f1.matrix[, 1] - out_wal.matrix[, 1]
plot(density(diff_of_mu), main = "Credibles intervals for the difference of Net Income between Flanders and Wallonia",
     xlab = "mu_1 - mu_2")
legend("topleft", legend = c("Quantile-based intervals", "HPD intervals"),
       col = 1:2, lty = 1:2)
abline(v = quantile(diff_of_mu, probs = c(0.025, 0.975), col = "red", lty = 2))
abline(v = HPDinterval(as.mcmc(diff_of_mu)), col = 2, lty = 2)
```



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)
```