Université catholique de Louvain
Louvain School of Statistics

LSTAT2130 - Bayesian Statistics

Project - Group Q

Adrien Kinart - 0424-1400
Lionel Lamy - 1294-1700
Simon Legendre - 6433-2000

May 24, 2021

Contents

Introduction

In this work, the Net Monthly Income (HNI) of household older than 30 years is studied across the two Belgian regions. These regions are denoted by $k = \{1, 2\}$ with respect to Flanders and Wallonia, respectively. The 1228 households were listed with respect to 10 income intervals. The detailed frequency table is given below:

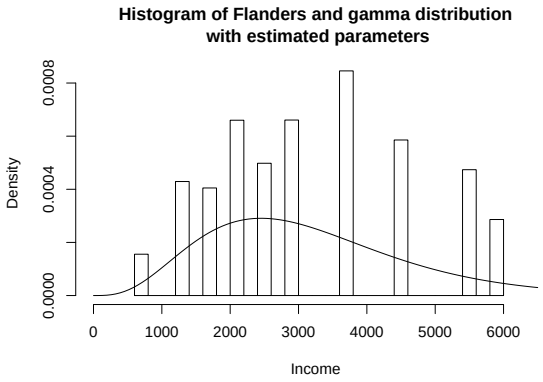
	Net monthly household income (in euros)										
Region	$\leq$1200	[1200, 1500)	[1500,1800)	[1800,2300)	[2300,2700)	[2700,3300)	[3300,4000)	[4000,4900)	[4900,6000)	$\geq$6000	Total
Flanders	25	69	65	106	80	106	136	94	76	46	803
Wallonia	17	36	47	58	47	53	59	54	33	21	425

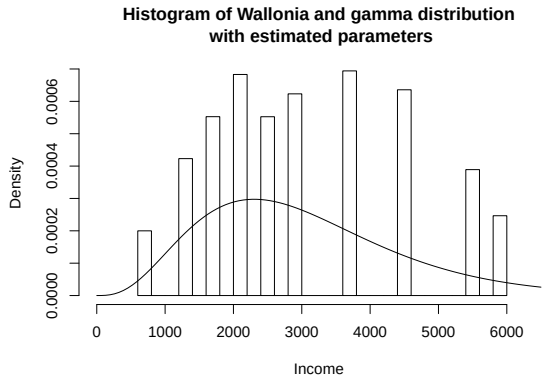
Table 1: Frequency table of the survey

By assuming the income to follow a gamma distribution, one can plot their experimental histogram and plot the theoretical density plot by estimating the shape and rate parameters with Maximum of Likelihood. This allows to get a first sight on the data behavior at hand.

	Flanders	Wallonia
$\hat{\mu}$	3182.756	3043.396
$\hat{\phi}$	0.229	0.243

Table 2: Estimated μ and ϕ for Flanders and Wallonia





1 Question 1

1.1 (a) Theoretical probability

Let X be the HNI regardless the 2 regions, it is assumed it follows a Gamma distribution with parameters $\theta_k := (\mu_k, \phi_k)$. It can be reparametrised in terms of its mean μ and dispersion parameter ϕ with the following trick:

$$\kappa = \frac{1}{\phi}$$
$$\lambda = \frac{1}{\phi\,\mu}$$

resulting in the following density $f(x)$

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

and the cumulative distribution function is such as

$$F(x) = \int_0^x f(u)\ \mathrm{d}u = \frac{\gamma(\phi^{-1}, \frac{x}{\phi\mu})}{\Gamma(\phi^{-1})} \tag{2}$$

where $\Gamma(a)$ and $\gamma(a, b)$ are the complete gamma and lower incomplete gamma functions, defined as:

$$\gamma(a, b) = \int_0^b t^{a-1} \exp(-t) \mathrm{d}t$$
$$\Gamma(a) = \gamma(a, \infty)$$

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

$$P(x \in I_j) = \int_{I_j} \frac{(\phi\mu)^{\frac{-1}{\phi}}}{\Gamma(\phi^{-1})} x^{\frac{1}{\phi}-1} \exp\left(\frac{-x}{\phi\mu}\right)\ \mathrm{d}x \tag{3}$$

Using CDF writing , the probability can be proposed as a difference of CDF :

$$p_j = \begin{cases} F(x_j) & \text{if} \quad j = 1 \\ F(x_{j+1}) - F(x_j) & \text{if} \quad j \in \{2, .., 9\} \\ 1 - F(x_j) & \text{if} \quad j = 10 \end{cases} \tag{4}$$

$$p_j = \begin{cases} \frac{\gamma(\phi^{-1}, \frac{x_j}{\phi\mu})}{\Gamma(\phi^{-1})} & \text{if} \quad j = 1 \\ \frac{1}{\Gamma(\phi^{-1})}\Big(\gamma(\phi^{-1}, \frac{x_{j+1}}{\phi\mu}) - \gamma(\phi^{-1}, \frac{x_j}{\phi\mu})\Big) & \text{if} \quad j \in \{2, .., 9\} \\ 1 - \frac{\gamma(\phi^{-1}, \frac{x_j}{\phi\mu})}{\Gamma(\phi^{-1})} & \text{if} \quad j = 10 \end{cases} \tag{5}$$

1.2 (b) Theoretical expression for the likelihood

Assuming the frequency distribution in a given region[1] to be multinomial, one has, writing $P := (p_1, .., p_{10})$ and $Y := (Y_1, ...Y_{10})$:

$$\begin{aligned} Y|P \sim \text{Mul}(Y, P) &= \frac{(\sum y_i)!}{y_1! \dots y_{10}!}\ p_1^{y_1} \times ... \times p_{10}^{y_{10}} \text{ when} \sum_{j=1}^{10} p_j = 1 \\ &= 0 \text{ otherwise} \end{aligned}$$

For such a distribution, the likelihood L is well known. Indeed, up to a multiplicative constant, for a region k, it is given by:

$$L(P_k, Y_k) = P(Y_k|P_k) \propto \prod_{j=1}^{10} p_{k,j}^{y_{k,j}}$$

To translate this likelihood in terms of μ_k and ϕ_k, one can use the definition of the probability referring to the difference of CDF (see equation 5). By substituting $p_{k,j}$ with it and writing $[0; x_1], [x_1; x_2], ..., [x_9; x_{10}], [x_{10}, +\infty]$ as $I_1, I_2, ..., I_{10}$, the likelihood function becomes, up to a multiplicative constant, as such:

$$L(\mu_k, \phi_k, Y_k) \propto \prod_{j=1}^{10} \left(\frac{1}{\Gamma(\phi^{-1})} \int\limits_{\frac{I_j}{\phi_k \mu_k}} x_j^{(\phi_k^{-1}-1)} e^{-x} \mathrm{d}x \right)^{y_{k,j}} \tag{6}$$

and then taking the logarithm, we obtain the log-likelihood ℓ for a certain region:

$$\ell(\mu_k, \phi_k, Y_k) = \sum_{j=1}^{10} y_{k,j} \times \ln(p_{k,j}) \tag{7}$$

where $y_{k,j}$ is the frequency of households in the interval j in the region k, and we recall that $p_{k,j}$ corresponds to the probability, or the area of the j^{th} interval in the region k. Again, by substituting with the difference of CDF, the log-likelihood becomes:

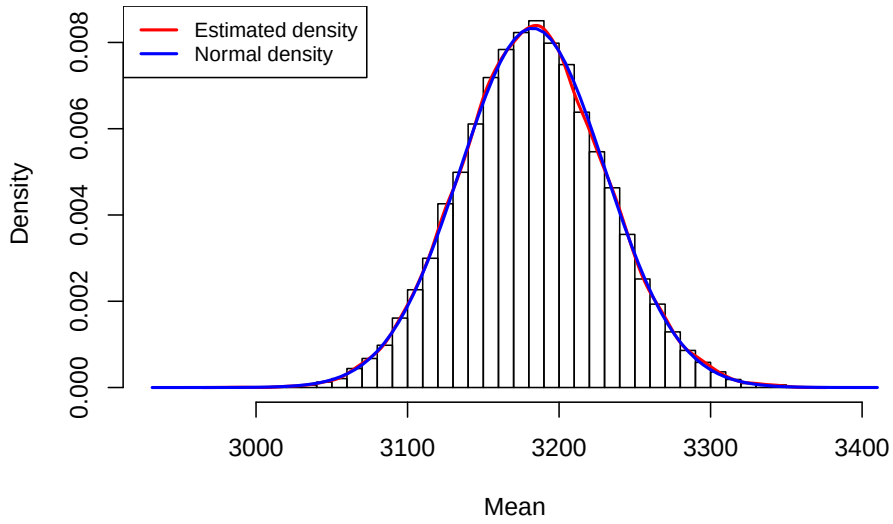
$$\ell(\mu_k, \phi_k, Y_k) = \sum_{j=1}^{10} y_{k,j} \times \ln \left(\frac{1}{\Gamma(\phi_k^{-1})} \int\limits_{\frac{I_j}{\phi_k \mu_k}} x^{(\phi_k^{-1}-1)} e^{-x} \mathrm{d}x \right) \tag{8}$$

[1] As such, the subscript k is dropped for expression of the multinomial

2 Question 2

2.1 Prior for the mean

The average net monthly household income, regardless the region, is believed to be within the interval $(2400, 3600)$ on a 95% confidence level. A convenient probability function that translates this belief is the Gaussian distribution with a mean at the center and standard deviation such that its 95% confidence interval corresponds to the mentioned bonds. This belief is confirmed by Monte Carlo simulations. Indeed, if one generates a large amount of gamma distributed samples and repeatedly take its mean, one can determine a density estimate for the mean. An example is shown below, using the data from Flanders. The estimated density and the estimated normal density are very similar.



Hence, by assuming such a distribution for the prior, one can easily estimate its parameters. For both regions $\mu_0 = 3000$ (the center of the confidence interval). Then, to get the standard deviation, one can use the decomposition of a classical confidence interval for the mean.

$$3000 - t_{(n_k-1,1-\alpha/2)}\frac{s_k}{\sqrt{n_k}} = 2400$$
$$\rightarrow \hat{\sigma}_0 = \frac{s_k}{\sqrt{n_k}} = \frac{600}{t_{(n_k-1,1-\alpha/2)}}$$
$$\text{where: } t_{n_1-1,1-\alpha/2} \approx t_{n_2-1,1-\alpha/2} \approx 1.96$$

Therefore $\hat{\sigma}_{Fl} \approx \hat{\sigma}_{Wal} \approx 306.12$, then $\mu_k \sim N(\mu_{0,k} = 3000, \sigma_{0,k} = 306.12)$. So we get as a prior for the parameter μ, up to a multiplicative constant:

$$\pi(\mu_k) \propto \exp\big(-\frac{1}{2\sigma_0^2}(\mu_k - \mu_0)^2\big) \ \forall k \in \{1, 2\} \tag{9}$$

The issue that one might rise is the fact that the HNI cannot be negative while theoretically, the Gaussian support take negative values. In this case, the probability of having negative values is $5.6249147 \times 10^{-23}$, which can be considered as non-significant.

2.2 Prior for the dispersion parameter

It is certain that ϕ_k lies in the interval $(0.0, 10.0)$ in both regions, but there is no knowledge on how the probability mass is broken down. In order to reflect this prior knowledge, one can assume that the dispersion parameter is uniformly distributed with lower and upper bonds of 0 and 10, respectively, i.e. $\phi_k \sim U(0, 10)$. Hence, up to a multiplicative constant, the prior probability can be represented via the indicator function as below:

$$\pi(\phi_k) \propto \mathbf{1}_{0,10} \; \forall k \in \{1, 2\}$$

2.3 The conjugate prior

Since there is no prior information on the dependence structure of the priors, it is recommended to consider them as independent in order to remain as least-informative as possible (this follows the maximal entropy principle). By doing so, the conjugate prior becomes:

$$\begin{aligned} \pi(\mu_k, \phi_k) &= \pi(\mu_k)\ \pi(\phi_k) \\ &\propto \exp\big(-\frac{1}{2\sigma_0^2}(\mu - \mu_0)^2\big)\mathbf{1}_{0;10} \end{aligned} \tag{10}$$

3 Question 3

3.1 (a) Joint posterior for Flanders

Since the likelihood and prior for μ_k and ϕ_k have been defined, a joint posterior can be expressed as their product (see equation 6 and equation 10). The joint posterior for Flanders, i.e. of $\theta_1 = (\mu_1, \phi_1)$ is defined, up to a multiplicative constant, here below:

$$P(\mu_1, \phi_1|Y_1) \propto \left(\prod_{j=1}^{10}\left(\frac{1}{\Gamma(\phi_1^{-1})}\int\limits_{\frac{I_j}{\phi_1\mu_1}} x^{(\phi_1^{-1}-1)}e^{-x}\mathrm{d}x\right)^{y_{1,j}}\right)\exp\big(-\frac{1}{2\sigma_0^2}(\mu-\mu_0)^2\big)1_{0;10} \tag{11}$$

3.2 (b) Write function lpost

Computing the logarithm of this expression, we find the log likelihood posterior, which is, up to an additive constant:

$$h(\mu_1, \phi_1, Y_1) = C^t + \ell(\mu_1, \phi_1, Y_1) + \ln(\pi(\mu_1, \phi_1))$$

Replacing the log-likelihood and log of the prior by their expression (see Equations 8 and 10), the log-posterior can be theoretically written as below:

$$h(\mu_1, \phi_1, Y_1) = C^t + \sum_{j=1}^{10} y_{1,j} \times \ln\left(\frac{1}{\Gamma(\phi^{-1})}\int\limits_{\frac{I_j}{\phi_1\mu_1}} x^{(\phi_1^{-1}-1)}e^{-x}\mathrm{d}x\right) - \frac{1}{2\sigma_0^2}(\mu-\mu_0)^2 + \ln(\mathbf{1}_{0;10}) \tag{12}$$

This expression has been implemented in R by not considering the constant term.

```
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)
}
```

4 Question 4

4.1 Approximation of the joint posterior

Laplace approximation is a method that allows to approximate the joint posterior distribution $h(\mu_1, \phi_1, Y_1)$ by a multimodal normal distribution (bimodal in this case).

First, if the Gaussian assumption is sustainable, the modes should correspond to the mean of the functions. Hence, one just need to optimize the function in order to find an estimated mean for the parameters of interest. The standard deviation is found by assuming that the second order Taylor expansion of the distribution function is precise enough. If it's the case, then, at the mode $\tilde{\theta}$, the log-posterior could be written as follow:

$$h(\theta) \approx h(\tilde{\theta}) - \frac{1}{2}(\theta - \tilde{\theta})^t \, \mathcal{I}(\tilde{\theta}) \, (\theta - \tilde{\theta})$$

with $\mathcal{I}(\tilde{\theta})$ the opposite of the hessian matrix (i.e. the matrix composed of the second derivatives). By taking the exponential, one can easily see that it corresponds to the multivariate normal distribution with $\mathcal{I}(\tilde{\theta})$ being the variance-covariance matrix Σ. This is the reason why the optimization in the Laplace methodology is done by including the Hessian matrix. Accordingly, one has:

$$\begin{cases} \mu_{\theta_1}^t = \big(E(\mu_1), E(\phi_1)\big) \\ \mathbf{\Sigma_1} = \begin{pmatrix} \sigma_{\mu_1} & \sigma_{\mu_1,\phi_1} \\ \sigma_{\phi_1,\mu_1} & \sigma_{\phi_1} \end{pmatrix} \end{cases}$$

In this case, the optimization is done numerically, using starting values close the MLE's. Once done, 50000 bivariate Gaussian observations are generated.

```r
# Starting values
inits <- c(mu = mu_prior, phi = 0.01)
# Laplace approximation
laplace_approx <- function(inits, frequencies) {
  fit <- optim(inits, lpost, control = list(fnscale = -1), hessian = T, freq =
frequencies)
  params <- fit$par
  param_cov_mat <- solve(-fit$hessian)
  samples <- rmvnorm(50000, params, param_cov_mat)

  list(samples = samples, parameters = params, cov = param_cov_mat)
}

laplace_fl <- laplace_approx(inits, flanders)
laplace_fl.mcmc <- mcmc(laplace_fl$samples)
```

A summary of the data generated is available in appendix.

4.2 Credible intervals Laplace approximation

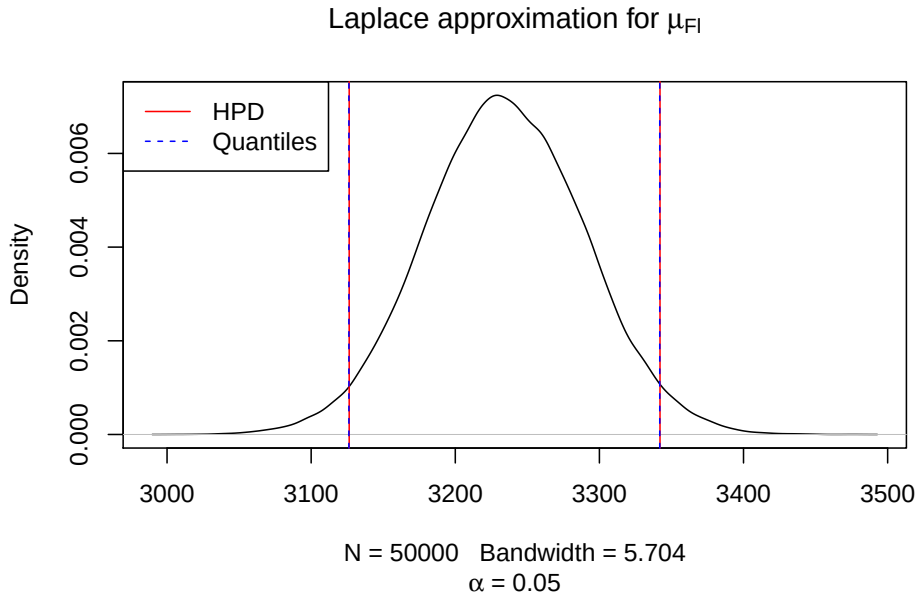
To get the credible interval for μ_1, the marginal posterior distribution of μ_1 is needed. It is given by :

$$P(\mu_1|Y_1) \propto \int p(\mu_1, \phi_1|Y_1)\ \mathrm{d}\phi_1$$

It can be shown that the marginal (univariate) distribution of the bivariate Gaussian distribution $N\big(\mu_{\theta_1}, \boldsymbol{\Sigma}_1\big)$ also follow a normal distribution:

$$\mu_1|Y_1 \sim N\big(E(\mu_1), \sigma_{\mu_1}\big)$$

Accordingly, a 95% quantile based or HPD based can be easily taken from the generated samples. The interval based on the quantile is [3126, 3342] , while the one based on the highest posterior density is [3126, 3342]. Those are shown in the following graph where the marginal distribution of the mean for Flanders is plotted.



5 Question 5

5.1 (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 $t = 2, \ldots, M$
   - 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 $\theta_t = (\mu_t, \phi_t)$

Which, in R gives the following code:

```
# Metropolis algorithm, mu and phi updated one at the time
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)
  }

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

burnin <- function(array, percent) {
  tail(array, -percent * nrow(array))
}

sd_hat_mu <- sqrt(laplace_fl$cov[1, 1])
sd_hat_phi <- sqrt(laplace_fl$cov[2, 2])
```

In order to reach an acceptance rate as close as possible to 40%, it was necessary to multiply by factors $f_1 = f_2 = 2.75$. The obtained rates, with $M = 50000$ iterations are such:

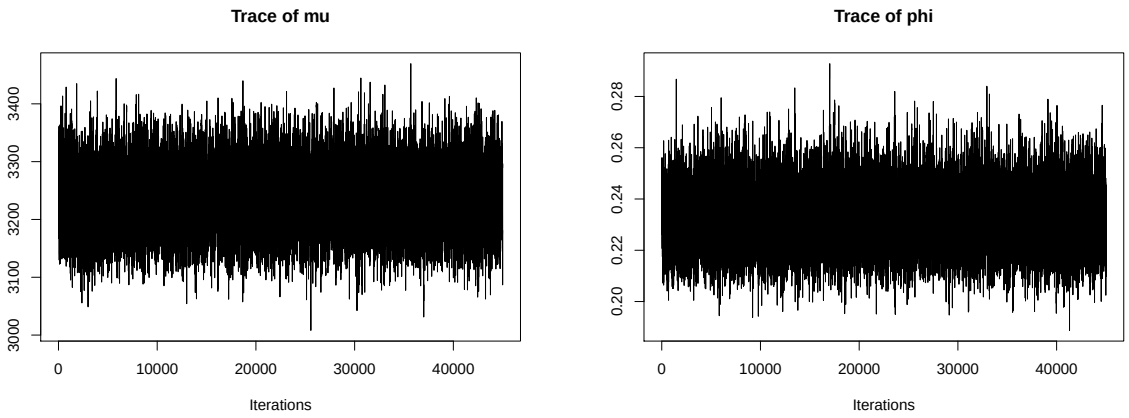
mu	phi
40.272	39.978

Table 3: Acceptance rates for the metropolis algorithm

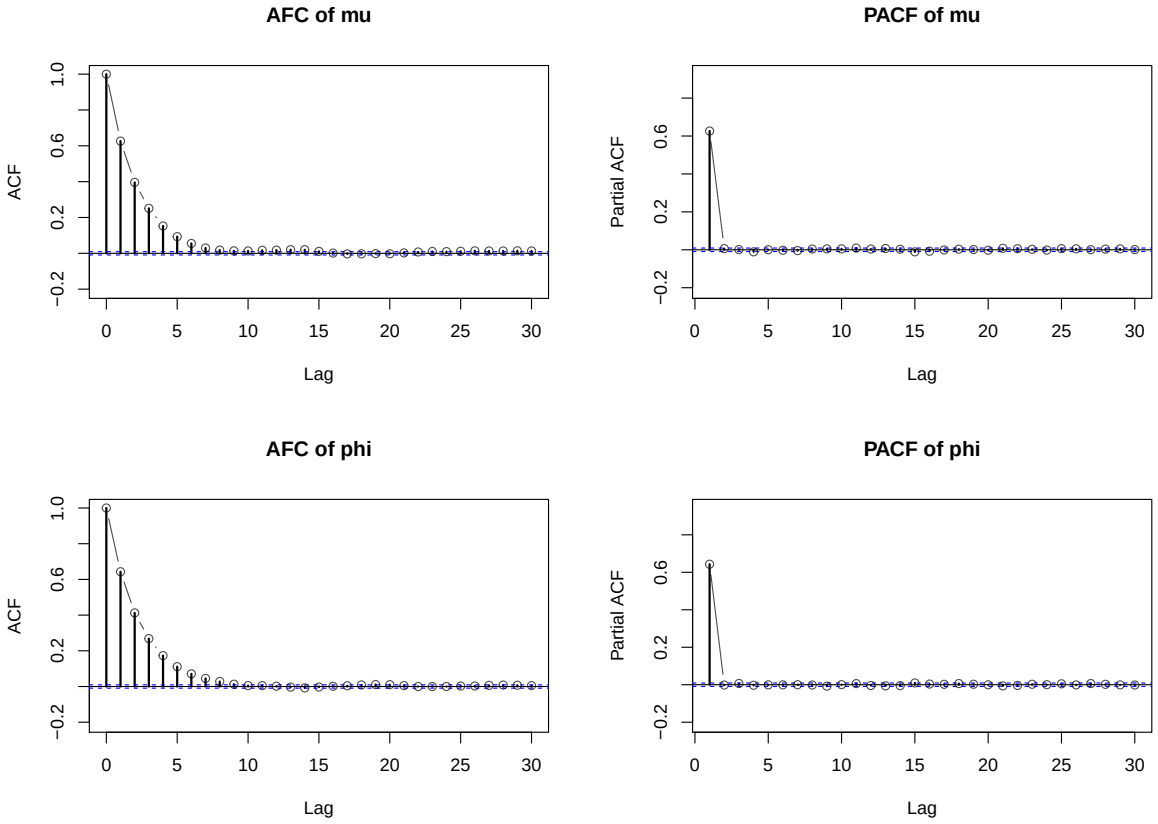
5.2 (b) diagnostic for convergence

Graphs analysis

The first, trivial, way of checking if the convergence occurred is by a visual analysis of the generated variables. One can look at the traceplot as well as the ACF and PACF.



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 θ does not seem to be visited too slowly.



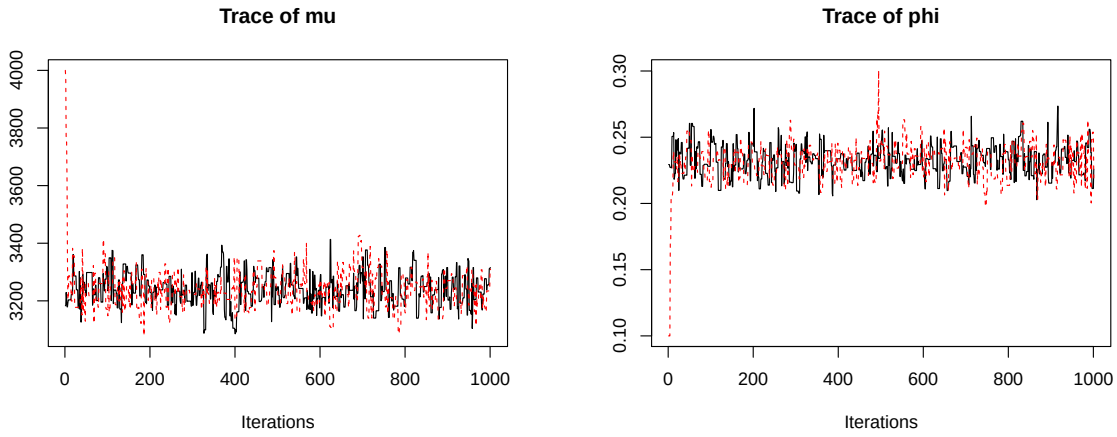
Since the chains clearly seem to be generated with respect to an Autoregressive process, one can compute effectively the effective sample size. It represents how many samples would be generated if there were no autocorrelation.

```
   mu   phi
10412  9778
```

Effective sample size of Metropolis

Gelman-Rubin

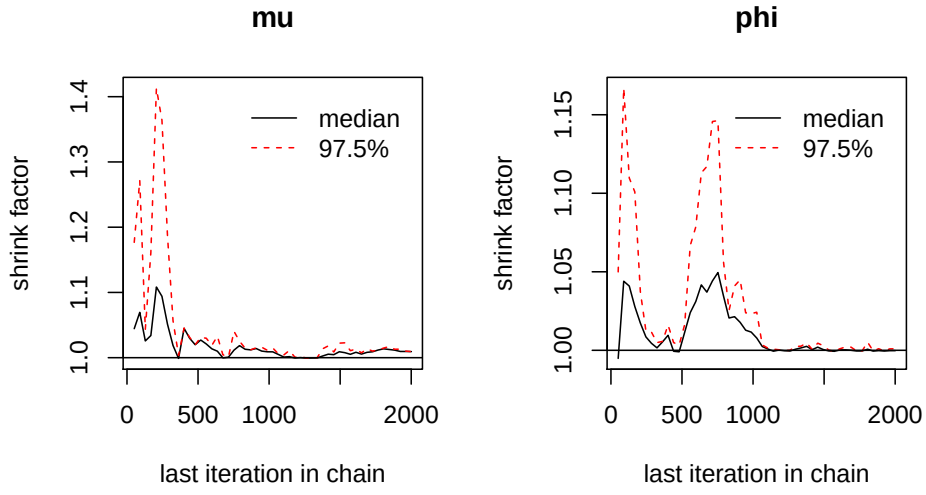
This method requires multiple chains. In this work, the diagnostic will be done with 2 generated chains with two different starting values. As a second chain, more exotic values (but still possible , a priori) are proposed for initialization: 4000 for μ_1 and 0.1 for ϕ_1. After having run the second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



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.

	Point est.	Upper C.I.
mu	1.014973	1.019435
phi	1.007675	1.028385

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:



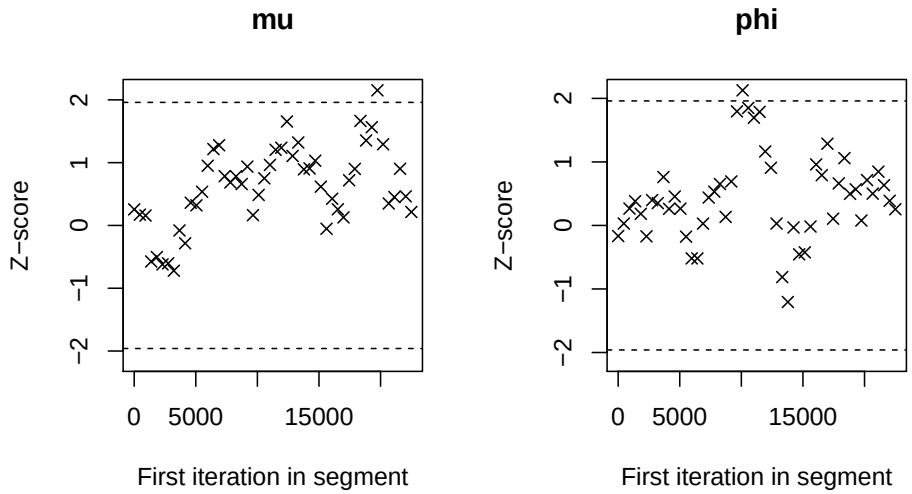
Accordingly, the convergence seems to occurs after roughly 1000 observations for both.

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:

```
Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5

    mu    phi
 0.256 -0.167
```

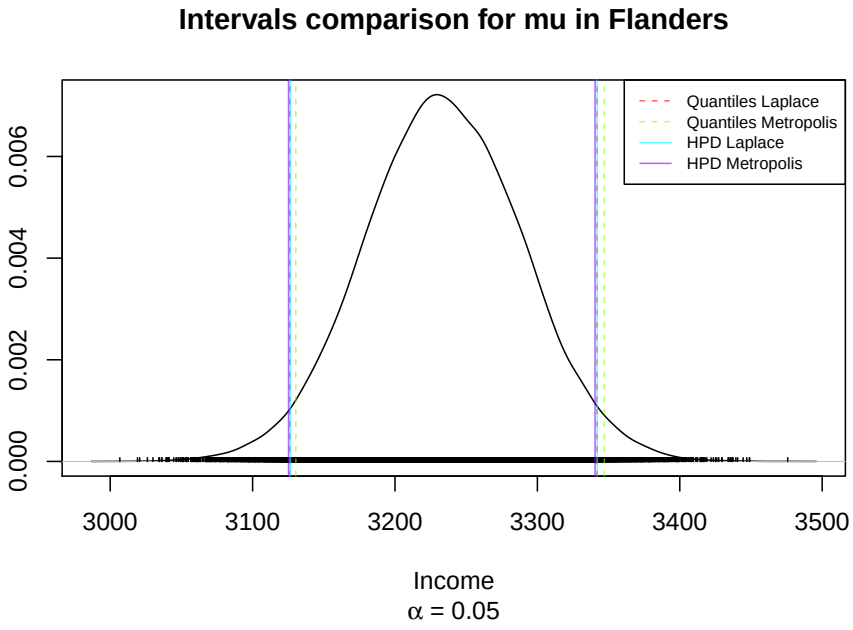
To strengthen the belief of convergence, one can successively discard larger numbers of iterations from the beginning of the chain. From there, the z-score is successively computed and presented here below:



Except for one value, the hypothesis of same mu and phi are never rejected. There is still no proof of non convergence of the generated chains.

5.3 (c) Credible intervals for μ_1

It would be interesting to calculate intervals and compare them to those previously found with the Laplace approximation. Using the same method, here are the results obtained.



The intervals being very close to each other, it is difficult to distinguish them visually. Thus, they are shown in the table below.

	Quantiles		HPD	
Laplace	3126.17	3341.87	3126.39	3342.05
Metropolis	3130.39	3347.01	3125.23	3340.49

Table 4: Comparison of credible intervals between Laplace and Metropolis

6 Question 6

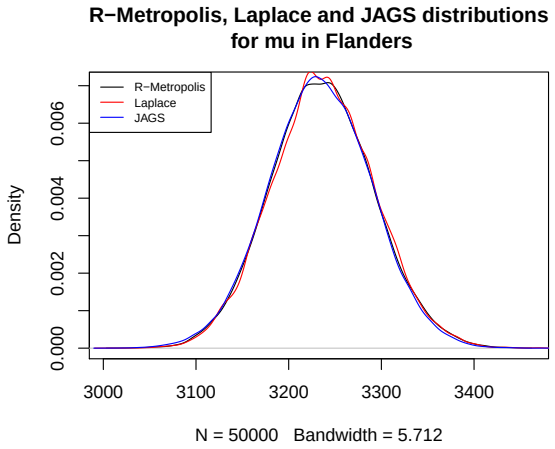
The use of JAGS is relatively simple. One just have to build a model by providing the likelihood and the statistical distributions so that the program can run and take care of the rest. In accordance with the information given in the previous sections, the model is defined as follows

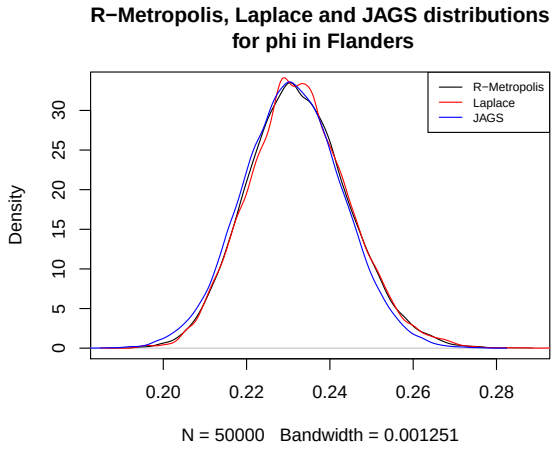
```
# JAGS Metropolis Model
metro_model <- function() {
  # Utils
  kappa <- 1 / phi
  lambda <- 1 / (phi * mu)

  # Probabilities
  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)
}
```

A summary of the results is available in appendix.

The graphical elements (traceplot, acf, ..) being very similar, only the comparison between the distributions estimated by the different models is presented. It is indeed also noticeable in the figures below that the estimated distributions are very close, either for mu or for phi. With the only small exception of JAGS, for phi which seems to be slightly lower.





Not surprisingly, the intervals obtained via JAGS are very similar to those obtained previously, especially via the Metropolis algorithm in R. (See Table 4)

	Quantiles		HPD	
JAGS	3130.03	3347.28	3129.94	3346.99

Table 5: Credible intervals for the JAGS model

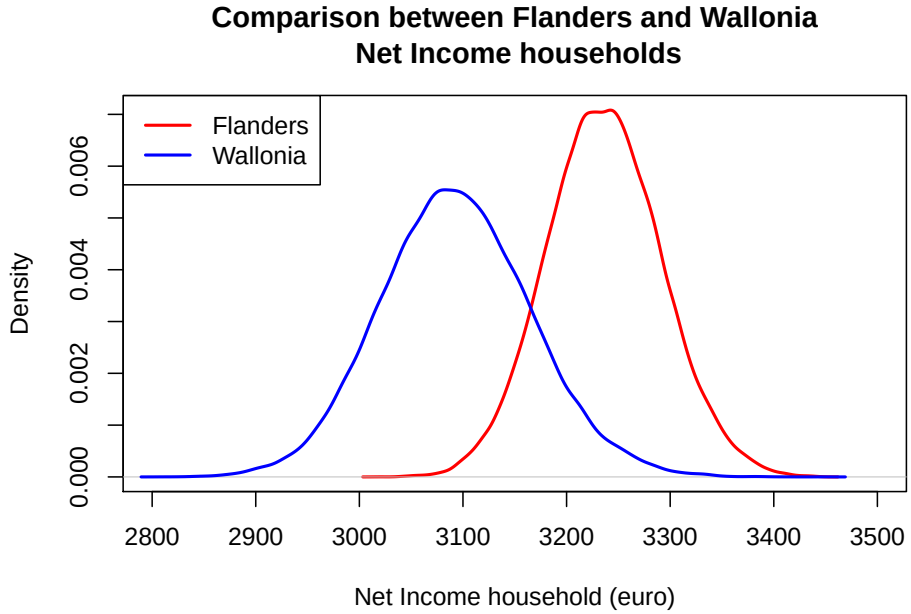
7 Question 7

Note 1. TODO

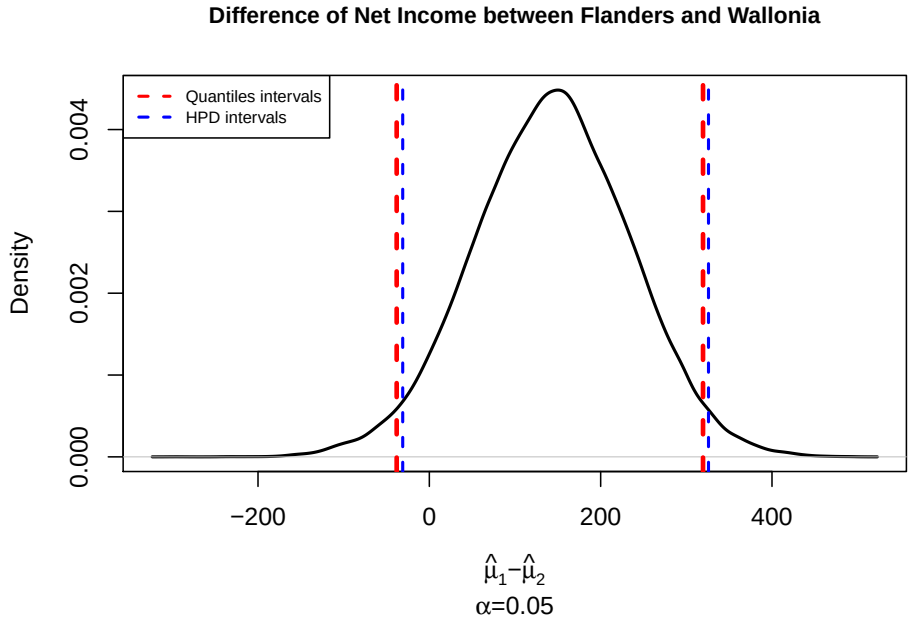
Voilà :)
Si vous faites cette partie ci je m'occupe de commenter tout le code <3
Et après on tryhard datamining !

8 Question 8

The ultimate goal of this work is comparing the households income divergence between Flanders and Wallonia given some a priori and after a experimental study through 1228 Belgian households. To do so, 50000 samples have been generated through the metropolis algorithm for Flanders and its French-talking counterpart. Their estimated density is jointly represented here below:



There is a 143.1€ difference on average. However, a large part of the density seems to overlap. The density of their difference is shown below along with a 95% quantile based and HPD credible intervals:



On a 95% confidence level, it is impossible to conclude that the average household income

level is significantly different between the regions. According to this Bayesian approach, one can only be 88.36 % sure that the income is different from one another. This is computed by looking from which confidence level one can reject the null hypothesis of same average income.

A Appendix

A.1 Output

A.1.1 Summary of the Laplace approximation for Flanders

```
Iterations = 1:50000
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  3234.199 55.1709 0.2467319      0.2467319
phi    0.231  0.0119 0.0000533      0.0000533

2. Quantiles for each variable:

        2.5%      25%      50%      75%    97.5%
mu  3126.373 3196.994 3233.745 3271.593 3342.630
phi    0.208    0.223    0.231    0.239    0.255
```

A.1.2 Summary of the JAGS model for Flanders

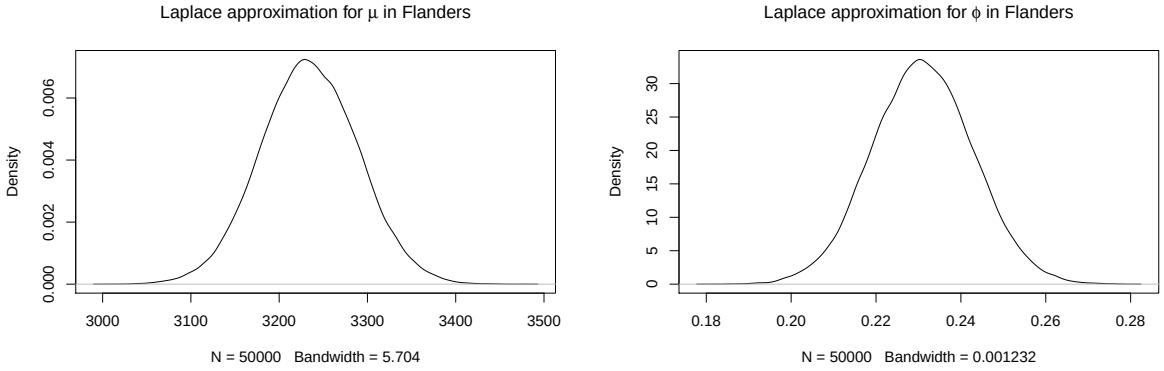
```
Iterations = 2001:52000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 50000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

        Mean      SD  Naive SE Time-series SE
mu  3236.235 55.2553 0.2471092      0.3120596
phi    0.232  0.0121 0.0000541      0.0000699

2. Quantiles for each variable:

       2.5%      25%      50%     75%    97.5%
mu  3130.03 3198.547 3235.560 3273.27 3347.277
phi    0.21    0.224    0.232    0.24    0.257
```

A.1.3 Summary of the JAGS model for Wallonia

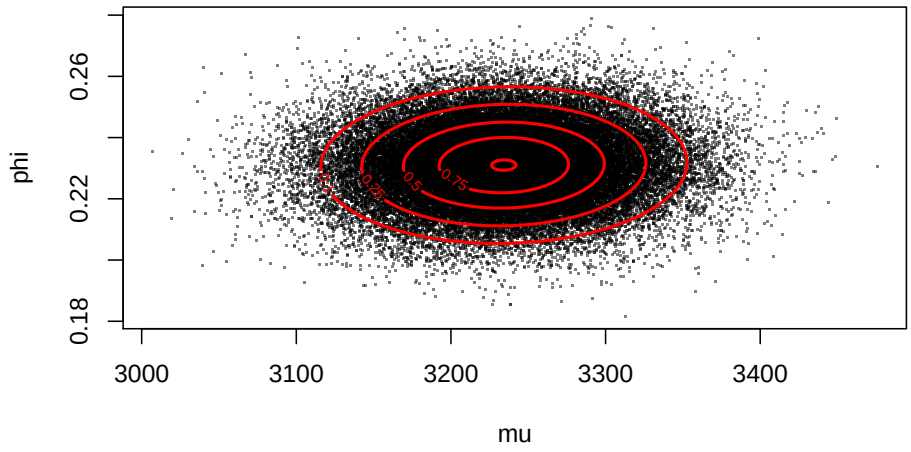
```
Iterations = 2001:52000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 50000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

        Mean      SD  Naive SE Time-series SE
mu  3093.131 72.4483 0.3239988       0.417904
phi    0.242  0.0175 0.0000782       0.000103

2. Quantiles for each variable:

       2.5%     25%      50%      75%    97.5%
mu  2955.09 3043.99 3091.687 3140.907 3240.777
phi    0.21    0.23    0.241    0.253    0.279
```

A.2 Figures

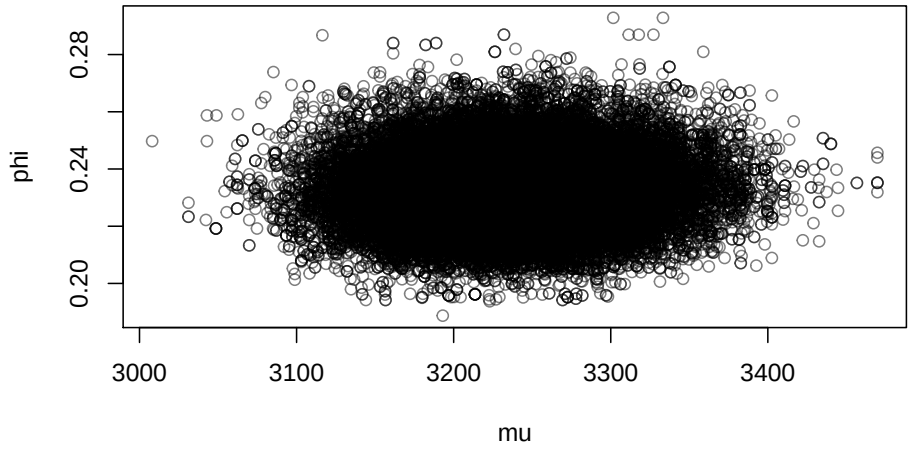
A.2.1 Laplace approximation density plots



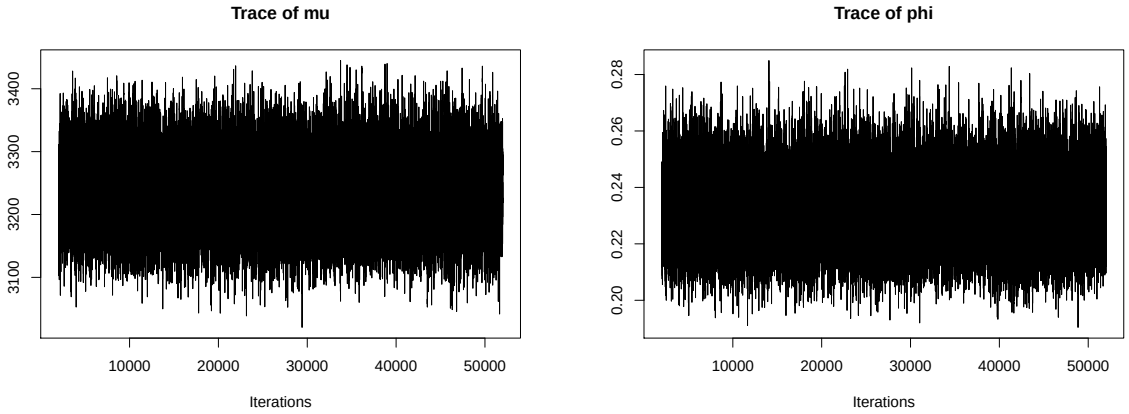
A.2.2 Laplace approximation contour plot



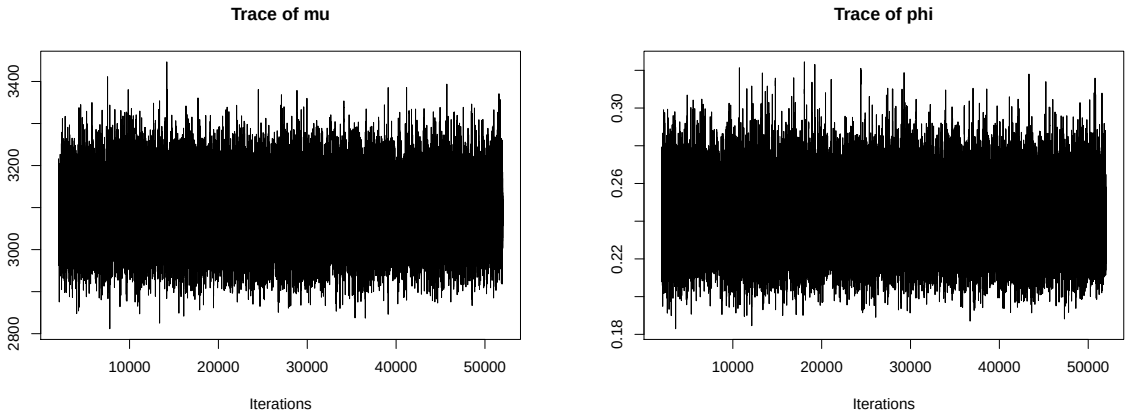
A.2.3 Metropolis mu vs phi



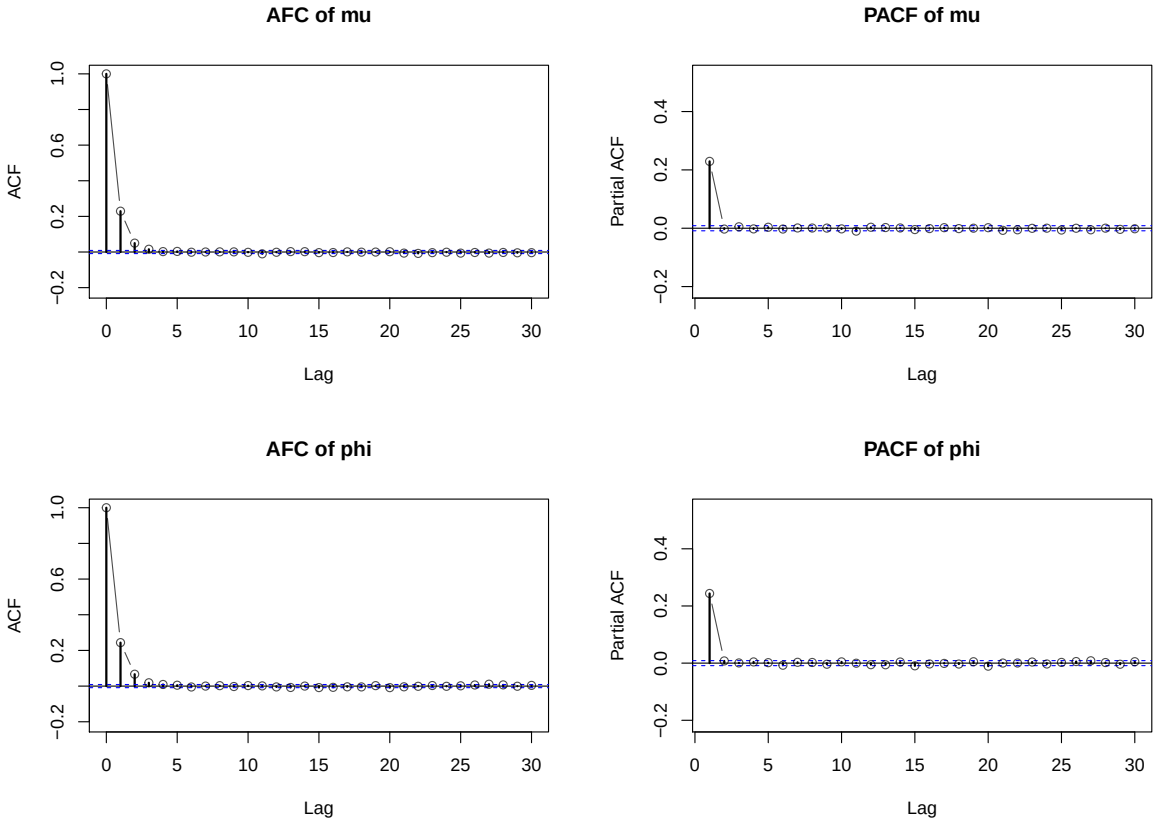
A.2.4 JAGS traceplot for Flanders



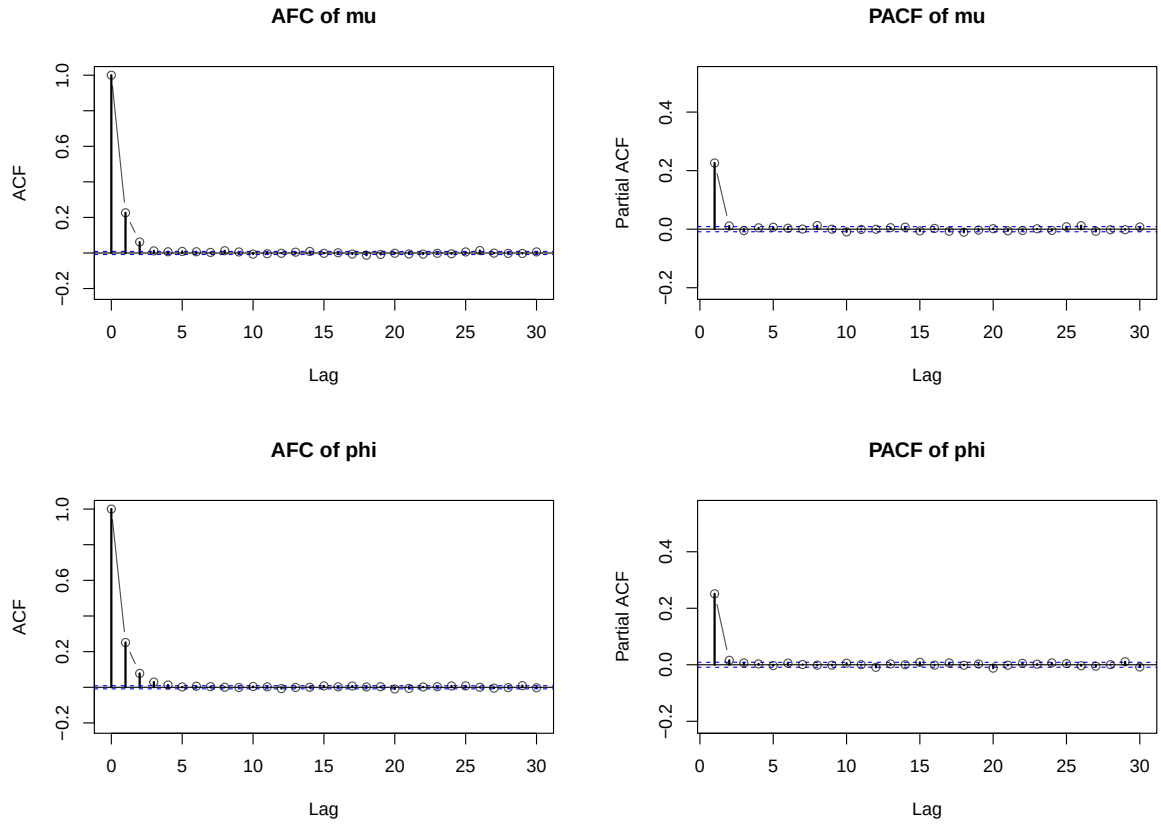
A.2.5 JAGS traceplot for Wallonia



A.2.6 JAGS ACF/PACF for Flanders



A.2.7 JAGS ACF/PACF for Wallonia



A.3 Code

```
# Require the necessary packages

if (!require(mvtnorm)) {install.packages("mvtnorm");require(mvtnorm)}
if (!require(EnvStats)) {install.packages("EnvStats");require(EnvStats)}
if (!require(R2WinBUGS)) {install.packages("R2WinBUGS");require(R2WinBUGS)}
if (!require(coda)) {install.packages("coda");require(coda)}
if (!require(rjags)) {install.packages("rjags");require(rjags)}
if (!require(latex2exp)) {install.packages("latex2exp");require(latex2exp)}

# Setup of the table
data <- 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(data) <- c("Flanders", "Wallonia")
colnames(data) <- c(
  "<1200", "[1200-1500)", "[1500-1800)", "[1800-2300)", "[2300-2700)",
  "[2700-3300)", "[3300-4000)", "[4000-4900)", "[4900-6000)", ">=6000"
```

```
)

flanders <- data[1, ]
wallonia <- data[2, ]
flanders.n <- sum(flanders)
wallonia.n <- sum(wallonia)
flanders.prop <- prop.table(flanders)
wallonia.prop <- prop.table(wallonia)

# Intervals of the frequency table
intervals <- c(0, 1200, 1500, 1800, 2300, 2700, 3300, 4000, 4900, 6000, Inf)
intervals.mean <- c(600, 1350, 1650, 2050, 2500, 3000, 3650, 4450, 5450, 6000)

# 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
kappa <- function(phi) {
  1 / phi
}
lambda <- function(phi, mu) {
  1 / (phi * mu)
}

alpha <- .05

# Estimate the parameters by simulation

obtain_the_estimated_parameters <- function(row, boot = F, simulations = 10000)
{
  if (boot) {
    sim_prop <- rowSums(rmultinom(simulations, sum(data), prop.table(data)[row,
]))
  } else {
    sim_prop <- data[row, ]
  }

  sim_data <- unlist(sapply(1:ncol(data), function(i) {
    rep(intervals.mean[i], sim_prop[i])
  }))
  est_gam <- egamma(sim_data)

  est_kappa <- est_gam$parameters["shape"]
  est_lambda <- 1 / est_gam$parameters["scale"]

  est_mu <- est_kappa / est_lambda
  est_phi <- 1 / est_kappa

  return(list(mu = est_mu, phi = est_phi, kappa = est_kappa, lambda =
  est_lambda, sim = sim_data))
```

```
}

estim_params_fl <- obtain_the_estimated_parameters(1, T)
estim_params_wal <- obtain_the_estimated_parameters(2, T)

estim_mu_fl <- estim_params_fl$mu
estim_phi_fl <- estim_params_fl$phi
estim_fl <- estim_params_fl$sim
estim_mu_wal <- estim_params_wal$mu
estim_phi_wal <- estim_params_wal$phi
estim_wal <- estim_params_wal$sim


hist(estim_fl, freq = F, xlim = c(0, 6500), main = "Histogram of Flanders and
gamma distribution\n with estimated parameters", xlab = "Income")
curve(dgamma(x, estim_params_fl$kappa, estim_params_fl$lambda), xlim = c(0,
6500), add = T)

hist(estim_wal, freq = F, xlim = c(0, 6500), main = "Histogram of Wallonia and
gamma distribution\n with estimated parameters", xlab = "Income")
curve(dgamma(x, estim_params_wal$kappa, estim_params_wal$lambda), xlim = c(0,
6500), add = T)

# Set the priors
mu_prior <- 3000
sigma_prior <- 306.12

(function(N) {
  moyenne <- sapply(1:N, function(i) {
    mean(rgamma(1000, estim_params_fl$kappa, estim_params_fl$lambda))
  })

  hist(moyenne, probability = T, breaks = 30, main = "", xlab = "Mean", xlim =
  c(min(moyenne) - 50, 50 + max(moyenne)))
  lines(density(moyenne), col = "red", lwd = 2)
  curve(dnorm(x, mean = mean(moyenne), sd = sd(moyenne)), add = T, col =
  "blue", lwd = 2)
  legend("topleft", legend = c("Estimated density", "Normal density"), col =
  c(2, 4), lty = 1, cex = 0.8, lwd = 2)
})(30000)

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)
}

# Starting values
inits <- c(mu = mu_prior, phi = 0.01)
# Laplace approximation
laplace_approx <- function(inits, frequencies) {
  fit <- optim(inits, lpost, control = list(fnscale = -1), hessian = T, freq =
frequencies)
  params <- fit$par
  param_cov_mat <- solve(-fit$hessian)
  samples <- rmvnorm(50000, params, param_cov_mat)

  list(samples = samples, parameters = params, cov = param_cov_mat)
}

laplace_fl <- laplace_approx(inits, flanders)
laplace_fl.mcmc <- mcmc(laplace_fl$samples)

# Credible intervals of Laplace (HPD vs QB)
marginal_post_Fl <- rnorm(1e6, laplace_fl$parameters["mu"],
sqrt(laplace_fl$cov[1, 1]))

QB_LP <- quantile(marginal_post_Fl, probs = c(alpha / 2, 1 - alpha / 2))
HPD_LP <- HPDinterval(as.mcmc(marginal_post_Fl), prob = 1 - alpha)

plot(density(laplace_fl$samples[, 1]), main = expression(paste("Laplace
approximation for ", mu[Fl])), sub = expression(paste(alpha, " = ", 0.05)))
legend("topleft",
  legend = c("HPD", "Quantiles"),
  col = c("red", "blue"), lty = 1:2
)

abline(v = c(HPD_LP[1], HPD_LP[2]), col = "red", lty = 1)
abline(v = c(QB_LP[1], QB_LP[2]), col = "blue", lty = 2)

# Metropolis algorithm, mu and phi updated one at the time
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)
  }

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

burnin <- function(array, percent) {
  tail(array, -percent * nrow(array))
}

sd_hat_mu <- sqrt(laplace_fl$cov[1, 1])
sd_hat_phi <- sqrt(laplace_fl$cov[2, 2])

# Run the algorithm
metro_fl1 <- metropolis_algorithm(5e4,
  theta = c(mu = estim_mu_fl, phi = estim_phi_fl),
  sd.propositions = c(
    mu = sd_hat_mu,
    phi = sd_hat_phi
  ),
  factors = c(mu = 2.75, phi = 2.75),
  frequencies = flanders
)
```

```r
# Remove burnin samples
metro_fl1.theta <- burnin(metro_fl1$theta, 0.1)
metro_fl1.theta.mcmc <- as.mcmc(metro_fl1.theta)

traceplot(metro_fl1.theta.mcmc)

# Plot slightly modified acf and pacf
plot.acf.pacf <- function(data, lag.max = 30, linked_by_line = T, titles =
c("ACF", "PACF"), ...) {

  ts.acf <- acf(x = data, lag = lag.max, plot = F)
  ts.pacf <- pacf(x = data, lag = lag.max, plot = F)

  # Add lines to improve the visualization
  plot(ts.acf, ylim = c(min(ts.acf$acf) - 0.2, min(max(ts.acf$acf + 0.3), 1)),
  main = titles[1], lwd = 2, ...)
  if (linked_by_line) {
    lines(ts.acf$lag, ts.acf$acf, type = "b", col = rgb(0, 0, 0, .7))
  }

  plot(ts.pacf, ylim = c(min(ts.pacf$acf) - 0.2, min(max(ts.pacf$acf + 0.3),
  1)), main = titles[2], lwd = 2, ...)
  if (linked_by_line) {
    lines(ts.pacf$lag, ts.pacf$acf, type = "b", col = rgb(0, 0, 0, .7))
  }
}

plot.acf.pacf(metro_fl1.theta[, 1], titles = c("AFC of mu", "PACF of mu"))
plot.acf.pacf(metro_fl1.theta[, 2], titles = c("AFC of phi", "PACF of phi"))

# out of 48 000 observations
effectiveSize(metro_fl1.theta.mcmc)

# Run the algorithm with another initial theta
metro_fl2 <- metropolis_algorithm(50000,
  theta = c(mu = 4000, phi = 0.1),
  sd.propositions = c(
    mu = sd_hat_mu,
    phi = sd_hat_phi
  ),
  factors = c(mu = 2.75, phi = 2.75),
  frequencies = flanders
)

metro_fl2.theta <- metro_fl2$theta
metro_fl2.theta.mcmc <- as.mcmc(metro_fl2.theta)

# Traceplot of the two metropolis, comparison
traceplot(list(
  head(as.mcmc(metro_fl1$theta), 1000),
```

```
  head(as.mcmc(metro_fl2$theta), 1000)
))

gelman.diag(list(head(metro_fl1.theta.mcmc, 1000), head(metro_fl2.theta.mcmc,
1000)))$psrf

gelman.plot(list(
  head(as.mcmc(metro_fl1$theta), 2000),
  head(as.mcmc(metro_fl2$theta), 2000)
))

# Z-Score plot
geweke.plot(metro_fl1.theta.mcmc, nbins = 50)

# Credible intervals of the Metropolis
QB_ME <- quantile(metro_fl1.theta[, 1], probs = c(alpha / 2, 1 - alpha / 2))
HPD_ME <- HPDinterval(as.mcmc(metro_fl1.theta[, 1]), prob = 1 - alpha)

# Comparison of Laplace vs Metropolis
credible_intervals_comparison <- function(alpha) {
  colors <- rainbow(n = 4, alpha = .6)

  densplot(laplace_fl.mcmc[, 1], main = "Intervals comparison for mu in
  Flanders", xlab="Income", sub = expression(paste(alpha, " = ", 0.05)))
  legend("topright",
    legend = c("Quantiles Laplace", "Quantiles Metropolis", "HPD Laplace", "HPD
    Metropolis"), col = colors, lty = c(2, 2, 1, 1), cex = .7
  )

  abline(v = c(QB_LP[1], QB_LP[2]), col = colors[1], lty = 2)
  abline(v = c(QB_ME[1], QB_ME[2]), col = colors[2], lty = 2)
  abline(v = c(HPD_LP[1], HPD_LP[2]), col = colors[3], lty = 1)
  abline(v = c(HPD_ME[1], HPD_ME[2]), col = colors[4], lty = 1)

  # return(list(QB=QB,HPD=HPD))
}

credible_intervals_comparison(.05)

# JAGS Metropolis Model
metro_model <- function() {
  # Utils
  kappa <- 1 / phi
  lambda <- 1 / (phi * mu)

  # Probabilities
  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)

# Test with Flanders
jags_fl <- jags.model(
  file = model.file,
  inits = list(list(mu = estim_mu_fl, phi = estim_phi_fl)),
  data = list(n = 803, intervals = intervals, y = flanders),
  n.chains = 1,
  quiet = T
)

update(jags_fl, 1000)

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

plot(density(out_fl.matrix[, "mu"]), main = "R-Metropolis, Laplace and JAGS
distributions\n for mu in Flanders")
lines(density(metro_fl1.theta[, 1]), col = "red")
lines(density(laplace_fl$samples[, 1]), col = "blue")
legend("topleft", legend=c("R-Metropolis", "Laplace", "JAGS"), col=c(1,2,4),
lty=1, cex=.7)

plot(density(out_fl.matrix[, "phi"]), main = "R-Metropolis, Laplace and JAGS
distributions\n for phi in Flanders")
lines(density(metro_fl1.theta[, 2]), col = "red")
lines(density(laplace_fl$samples[, 2]), col = "blue")
legend("topright", legend=c("R-Metropolis", "Laplace", "JAGS"), col=c(1,2,4),
lty=1, cex=.7)

# Credible intervals of JAGS model
QB_JA <- quantile(out_fl.matrix[, 1], probs = c(alpha / 2, 1 - alpha / 2))
HPD_JA <- HPDinterval(mcmc(out_fl.matrix[, 1]), prob = 1 - alpha)

# Test with Wallonia
jags_wal <- jags.model(
  file = model.file,
  inits = list(list(mu = estim_mu_wal, phi = estim_phi_wal)),
  data = list(n = 425, intervals = intervals, y = wallonia),
  n.chains = 1,
  quiet = T
)
```

```r
update(jags_wal, 1000)

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

# Compare the income between region
plot(density(out_fl.matrix[, "mu"]),
  main = "Comparison between Flanders and Wallonia\n Net Income households",
  lty = 1, xlim = c(2800, 3500), xlab = "Net Income household (euro)", col = 2,
  lwd = 2
)
legend("topleft", legend = c("Flanders", "Wallonia"), col = c(2, 4), lwd = 2,
lty = 1)
lines(density(out_wal.matrix[, "mu"]), col = 4, lwd = 2)

# Credible intervals of the difference
diff_of_mu <- out_fl.matrix[, 1] - out_wal.matrix[, 1]
plot(density(diff_of_mu),
  main = "Difference of Net Income between Flanders and Wallonia",
  xlab = expression(paste(hat(mu)[1], "-", hat(mu)[2])), cex.main = 0.9, lwd =
  2,
  sub = expression(paste(alpha, "=", 0.05))
)
legend("topleft", legend = c("Quantiles intervals", "HPD intervals"), col =
c("red", "blue"), lty = 2, cex = 0.7, lwd = 2)
abline(v = quantile(diff_of_mu, probs = c(0.025, 0.975)), col = "red", lty = 2,
lwd = 3)
abline(v = HPDinterval(as.mcmc(diff_of_mu)), col = "blue", lty = 2, lwd = 2)

# ============ APPENDIX =========== #
summary(laplace_fl.mcmc)

summary(out_fl)

summary(out_wal)

plot(density(laplace_fl$samples[, 1]),
  main = expression(paste("Laplace approximation for ", mu, " in Flanders"))
)
plot(density(laplace_fl$samples[, 2]),
  main = expression(paste("Laplace approximation for ", phi, " in Flanders"))
)

# Contour plot of the Laplace approximation
(function() {
  lvls <- c(0.01, 0.25, 0.5, 0.75, 0.9)

  x1 <- seq(2400, 3600, length = 3000)
  y1 <- seq(0.1, 0.4, length = 3000)
```

```
  z1 <- outer(x1, y1, function(x, y) {
    dmvnorm(cbind(x, y), colMeans(laplace_fl$samples), cov(laplace_fl$samples),
    log = T)
  })

  R <- exp(z1 - max(z1))

  plot(
    x = laplace_fl$samples[, 1], y = laplace_fl$samples[, 2],
    xlab = "mu", ylab = "phi", pch = ".", col = rgb(0, 0, 0, .5)
  )

  contour(x1, y1, R,
    lwd = 2, add = T,
    levels = exp(-0.5 * qchisq(lvls, 2)),
    labels = (1 - lvls), col = "red", method = "edge"
  )
})()
plot(metro_fl1.theta[, 1], metro_fl1.theta[, 2], xlab = "mu", ylab = "phi", col
= rgb(0, 0, 0, .5))

traceplot(out_fl)

traceplot(out_wal)

plot.acf.pacf(out_fl.matrix[, 1], titles = c("AFC of mu", "PACF of mu"))
plot.acf.pacf(out_fl.matrix[, 2], titles = c("AFC of phi", "PACF of phi"))

plot.acf.pacf(out_wal.matrix[, 1], titles = c("AFC of mu", "PACF of mu"))
plot.acf.pacf(out_wal.matrix[, 2], titles = c("AFC of phi", "PACF of phi"))
```