28462000 - Bourez Nicolas
25862000 - Loncour Romain
81862100 - Luyckx Matthieu

LSTAT2130
2023-2024

LSTAT2130 - Project report

Exercise 1

Consider a linear regression model for the price y_i of the ith rent:

$$\mu_i = \mathbb{E}(y_i|x_{i1}, x_{i2}) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}$$

where x_{i1} represents the recentered surface of the ith rent (adjusted to have mean 0) and x_{i2} represents the characteristics of its bathroom. Assuming a constant coefficient of variation (CV) and a Gamma conditional distribution for the rent price, one has $(y_i|\beta, x_{i1}, x_{i2}) \sim \mathrm{G}(\mu_i, \phi)$.

1. Provide an analytic form for the likelihood function $L(\theta|\mathcal{D})$ where $\theta = (\beta, \psi)$ and $\psi = \log \phi$.

   As we have $(y_i|\beta, x_{i1}, x_{i2}) \sim \mathrm{G}(\mu_i, \phi)$, we need to calculate the probability density function (pdf) of the Gamma distribution with shape parameter $\alpha = \frac{1}{\phi^2}$ and rate parameter $\beta = \frac{1}{\phi^2 \mu_i}$ in order to find the likelihood. This pdf is given by :

   $$f(y_i|\mu_i, \phi) = \frac{\left(\frac{1}{\phi^2 \mu_i}\right)^{\frac{1}{\phi^2}}}{\Gamma\left(\frac{1}{\phi^2}\right)} y_i^{\frac{1}{\phi^2}-1} e^{-\frac{1}{\phi^2 \mu_i} y_i}$$

   This simplifies to:

   $$f(y_i|\mu_i, \phi) = \frac{1}{\Gamma\left(\frac{1}{\phi^2}\right) \mu_i^{\frac{1}{\phi^2}} \phi^{\frac{2}{\phi^2}}} y_i^{\frac{1}{\phi^2}-1} e^{-\frac{1}{\phi^2 \mu_i} y_i}$$

   Verification

   In Cepeda-Cuervo (2001)[1] and Cepeda-Cuervo & Gammerman (2005) [2], given $\mathbb{E}(Y_i) = \mu_i$ and the shape parameter a of the corresponding Gamma distribution, the original Gamma distribution $\mathcal{G}(a, b)$ is written as :

   $$f(y_i|\mu_i, a) = \frac{1}{y_i \Gamma(a)} \left(\frac{a y_i}{\mu_i}\right)^a e^{-a y_i/\mu_i}$$

   replacing a by $1/\phi^2$, we find :

   $$\frac{1}{y_i \Gamma(1/\phi^2)} \left(\frac{y_i}{\mu_i \phi^2}\right)^{1/\phi^2} e^{-y_i/\mu_i \phi^2}$$

   Which is equivalent to the expression we found before.

   As the likelihood has to depend on ψ and β, we have to make the change of variable $\phi = e^{\psi}$ and $\mu_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}$, which leads to :

   $$L(\theta|y_i) = \frac{1}{\Gamma(e^{-2\psi})(\boldsymbol{x}^{\boldsymbol{t}}\boldsymbol{\beta})^{e^{-2\psi}} e^{\frac{2\psi}{e^{2\psi}}}} y_i^{e^{-2\psi}-1} e^{-\frac{1}{e^{2\psi}(\boldsymbol{x}^{\boldsymbol{t}}\boldsymbol{\beta})} y_i}$$

with $\theta = (\beta, \psi)$

The likelihood function for the entire dataset $\mathcal{D}$, assuming the observations are independent and identically distributed (iid), is the product of the likelihoods for each observation:

$$\begin{aligned} L(\theta|\mathcal{D}) &= \prod_{i=1}^{n} L(\theta|y_i) \\ &= \prod_{i=1}^{n} \frac{1}{\Gamma(e^{-2\psi})(\boldsymbol{x}^{\boldsymbol{t}}\boldsymbol{\beta})^{e^{-2\psi}} e^{\frac{2\psi}{e^{2\psi}}}} y_i^{e^{-2\psi}-1} e^{-\frac{1}{e^{2\psi}(\boldsymbol{x}^{\boldsymbol{t}}\boldsymbol{\beta})} y_i} \end{aligned} \tag{1}$$

2. Provide a R function enabling to compute the log-likelihood for given parameter values (β, ψ) and data $\mathcal{D}$.

   See Appendix 1.1

3. Provide a R function enabling to compute the log-posterior $\log p(\theta|\mathcal{D})$ for given θ, data $\mathcal{D}$, and assuming large variance priors for β and ψ.

   The log-posterior is given by :

$$\log p(\theta|D) = \log L(\theta|D) + \log p(\beta) + \log p(\psi)$$

   See Appendix 1.2

4. Evaluate the posterior mode and information matrix at the mode. Using these results and the associated Laplace approximation to $p(\beta|\mathcal{D})$, obtain approximate 95% credible intervals for β_0, β_1, and β_2.

   - To find the mode of the posterior, we need to take the first order derivative with respect to θ and equate it to zero.
   - To find the information matrix at the mode, we need to calculate the hessian matrix according to θ and equate it to zero.

   These two operations can easily be found using an optimizer in `R`, available in Appendix 1.3 We then have the following values for the posterior mode and the information matrix :

β_0	β_1	β_2	ψ
462.369819	4.544781	85.753134	-1.098454

Table 1: Posterior mode

And for the information matrix, we have the following one:

31.632321552	0.6106593994	-36.794610701	-0.0013196149
0.610659399	0.0544462366	-1.214644269	-0.0000223131
-36.794610701	-1.2146442686	697.220711547	-0.0046337118
-0.001319615	-0.0000223131	-0.004633712	0.0005670554

Table 2: Information matrix

Here are the confidence intervals we found using Laplace approximation of the parameters :

Parameter	Lower Bound	Upper Bound
β_0	451.407507	473.465281
β_1	4.088653	5.003735
β_2	33.804692	137.282532

Table 3: Lower and upper bounds for the 95% confidence intervals of the parameters.

Exercise 2

1. Using a Metropolis algorithm (with componentwise proposals) and the R software (without using JAGS or specific packages), draw a random sample from the posterior distribution of $(\theta|\mathcal{D})$. Target an acceptance rate of 40% for each of the parameters.

   The Metropolis algorithm being a random walk, we have to choose a suitable starting value for θ^0. For that, we reused the optimizer from question 1.4 that returns also the best set of parameters found. In our case, the best set is :

β_0	β_1	β_2	ψ
462.436394	4.546194	85.543612	-1.098217

Table 4: Initial parameters of the Metropolis algorithm

   The *sd_prop* argument must also be tuned in order to converge to an acceptance rate of 40%. The best way to handle that is to fix some values, test them and then adjust them in order to converge to the 40% acceptance rate. We tried the following values :

sd_prop	Acceptance rate
(12.5, 0.75, 80, 0.1)	(0.43, 0.32, 0.35, 0.28)
(14, 0.5, 60, 0.05)	(0.38, 0.45, 0.44, 0.47)
(13.4, 0.57, 69, 0.068)	(0.4, 0.39, 0.4, 0.39)

Table 5: Acceptance rate according to *sd_prop*

   Then, with a try-and-adjust method, we will finally go for $sd\_prop = (13.4, 0.57, 69, 0.068)$ which are the values that get us closest to the acceptance rate of 40% for each parameter.

   The entire code can be found in Appendix 1.4

2. Evaluate the convergence of your algorithm. After a suitable burnin, evaluate the effective sample sizes for each of the model parameters and make sure that each of these values is at least 1000.

   In order to evaluate the convergence of the algorithm, the first (imprecise) method we tried is to look at the trace plots and see if the algorithm ends up fluctuating around the expected value with a pretty high frequency. Let's have a look at the plots :

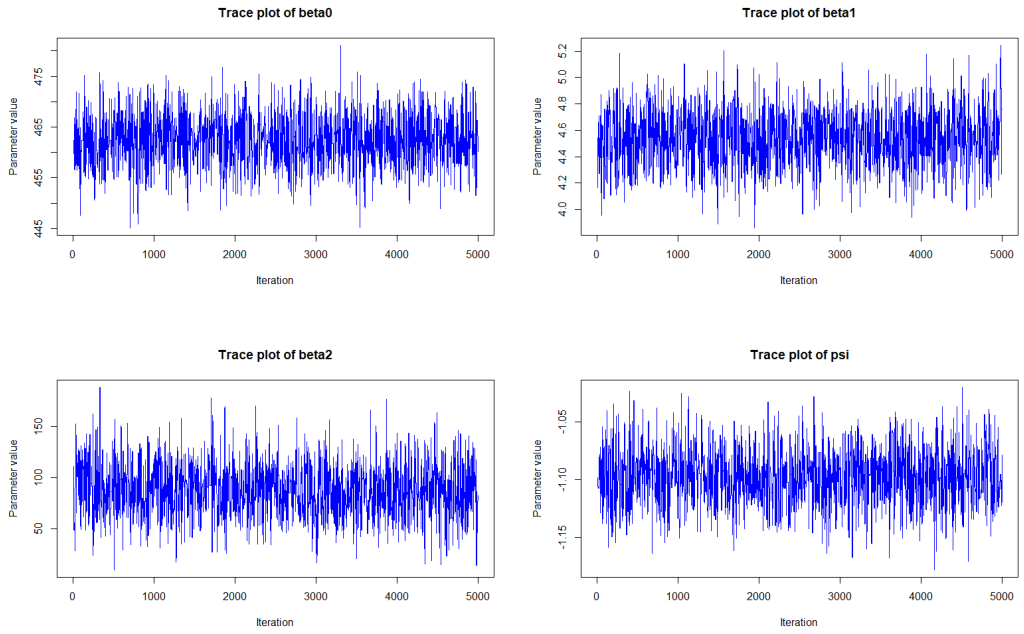


Figure 1: Traceplots of the difference parameter estimated by the Metropolis Algorithm

As we can see on Figure 1, they are indeed fluctuating as we want. However, let us now use a more precise method.

The Gelman-Rubin method evaluates the convergence by running multiple chains that start at different values. It is said that when $\hat{R}$ (see theory for formulas) is below 1.1 (and as close as possible to 1), it has converged. Let's try that by taking 3 different starting values, and computing the $\hat{R}$ plots.

As we can observe on Figure 2, the 4 parameters for which we compute this test converge all to 1 (almost), and are much smaller than 1.1. We can then assume that the algorithm has converged.

And, to make sure that the effective sample size has at least the size of 1000 values for each parameter, the function *effectiveSize(...)* will return the exact size for each parameter that are given in Table 6:

β_0	β_1	β_2	ψ
1161.257	1070.710	1114.043	1071.854

Table 6: Effective sample sizes

Both codes for running the imprecise (trace plots) and the precise (Gelman-Rubin) methods can be found in Appendix 1.5, with the line computing the effective sample size at the end.

3. Provide a point estimate and a 95% credible interval for each of the model parameters.

   For this part, only 4 lines are required and can be found in Appendix 1.6. These lines compute the means and the quantiles 2.5%, 50% and 97.5% that will be used to find the point estimates and the 95% credible intervals. The results are the showed in Table 7.

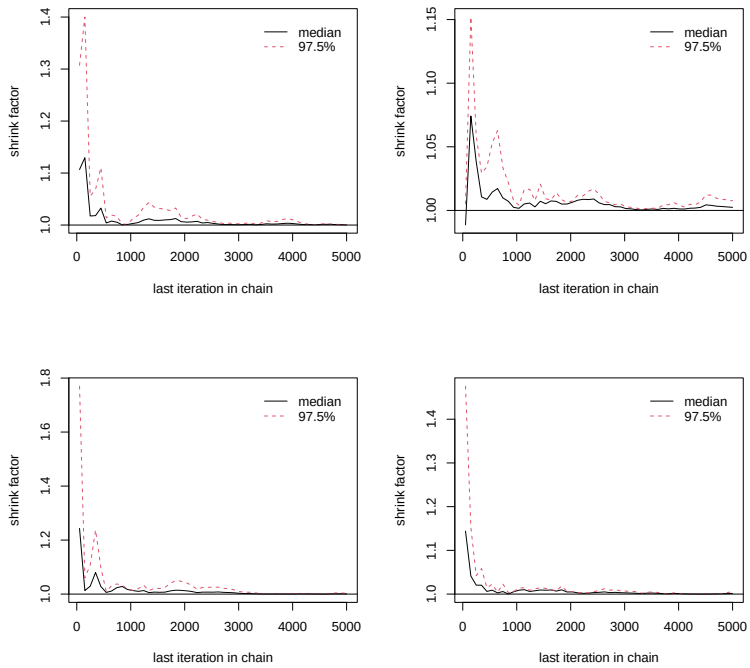


Figure 2: Result of the Gelman-Rubin test

Parameter	Mean	2.5%	50%	97.5%
β_0	462.44	453.094	462.332	472.073
β_1	4.541	4.132	4.542	4.924
β_2	87.825	41.717	86.759	137.665
ψ	-1.097	-1.143	-1.097	-1.052

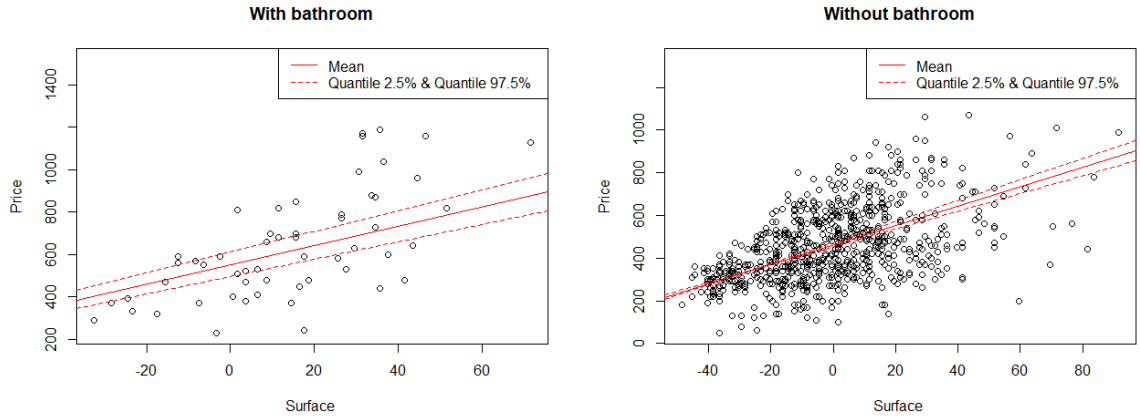
Table 7: Quantiles of the parameters

In this case, we could take the mean or the median as a point estimate, it does not matter a lot since they are very close (theoretically, they both should converge to the same value). And the 95% credible interval is defined as $[2.5\%; 97.5\%]$ for each parameter, which leads the result in Table 8:

Parameter	Point estimate	Credible interval
β_0	462.44	[453.094 ; 472.073]
β_1	4.541	[4.132 ; 4.924]
β_2	87.825	[41.717 ; 137.665]
ψ	-1.097	[-1.143 ; -1.052]

Table 8: Points estimate and credible intervals

4. Make 2 separate scatterplots $\{(x_{i1}, y_i) : i = 1, \ldots, n\}$ visualizing the association between the surface and the price of a rent with (plot 1) or without (plot 2) a premium bathroom. Superpose to each of them the corresponding estimated regression line and the associated 2.5%-97.5% conditional quantiles from the fitted Gamma distributions for a rent. Are the two fitted models satisfactory? Discuss.



This regression line and its 95% interval confidence are satisfactory. As both lines of 2.5% and 97.5% are very close to the regression line, it means that the majority (95% of them) of the samples are within this close interval. It is then well sampled. As we can see, because we have less data for the rents with a premium bathroom, the confidence interval is a bit wider and the samples are less precise. But as we can see, we can still be satisfied from this regression line.

Exercise 3

Consider now a more sophisticated regression model involving all the available covariates (including the kitchen status x_3) and enabling both the conditional mean and conditional CV to change with $x_i = (x_1, x_2, x_3)$:

$$\mu_i = E(y_i|x_i) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3}$$
$$\log \phi_i = \log \mathrm{CV}(y_i|x_i) = \psi_0 + \psi_1 x_{i1} + \psi_2 x_{i2} + \psi_3 x_{i3}$$

$(\beta, \psi, x_i) \sim G(\mu_i, \phi_i)$. Assuming large variance priors for the regression parameters β and ψ:

1. Sample $p(\theta|D)$ using JAGS with the same number of iterations (after burnin) as in Question 2. Detail your code, assess convergence of your chains and report your estimation results.

   The first thing we did was define the JAGS model we'll use to sample from the posterior. To do so, we first define the priors for every parameter we want to sample from, aka. the Betas and the Psis, for which we define a large variance prior using a Normal with mean 0 and precision $1e-4$ which guarantees a low uncertainty about the parameters we want to find. Afterwards, we define the likelihood relationships following what is expressed in the guidelines. We first define the mean μ, then the coefficient of variation ϕ and finally we express the form of the output y as a Gamma with parameters $\alpha = \frac{1}{\phi^2}$ and $\beta = \frac{1}{\phi^2 \mu}$.

   Figures 3 and 4 are the MCMC chains generated using this model. We can observe good convergence on every parameter. These good visual results are confirmed by Gelman-Rubin values close or equal to 1. Moreover, the values to which the parameters converge make sense and seem reasonable in our context. Using JAGS to sample gives us the following parameter values in Table 10.

Parameter	estimator JAGS model BETAS	estimator JAGS model PSIS
β_0 — ψ_0	458.1033	-1.096923
β_1 — ψ_1	4.494249	0.002364054
β_2 — ψ_2	79.10886	-0.08963886
β_3 — ψ_3	102.8424	-0.1192092

Table 9: Parameters obtained from JAGS

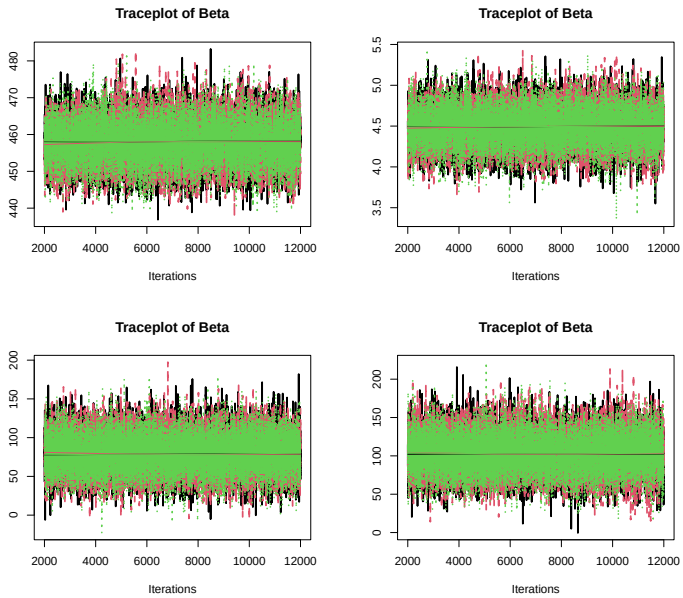


Figure 3: Traceplots of Betas — up-left : β_0, up-right : β_1, down-left : β_2, down-right : β_3

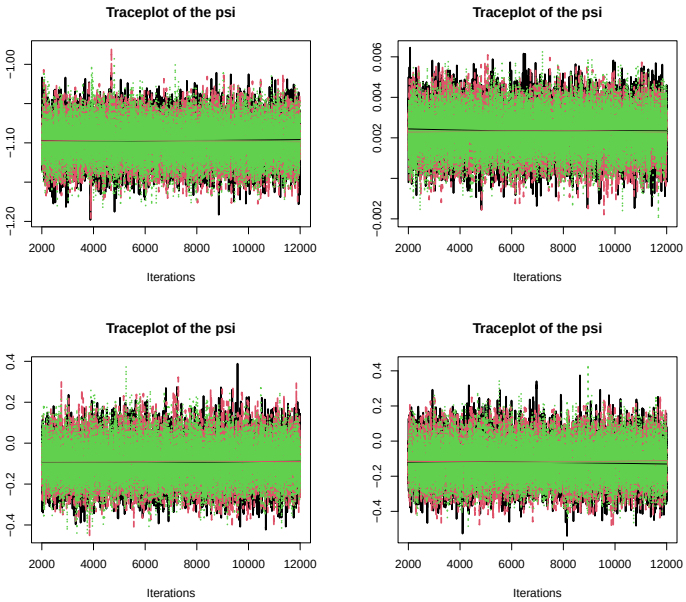


Figure 4: Traceplots of Psis — up-left : ψ_0, up-right : ψ_1, down-left : ψ_2, down-right : ψ_3

2. Give the 4 scatterplots of $(x_{i1}, y_i : i = 1, \ldots, n)$ corresponding to one of each of the 4 possible combinations for indicators x_2 and x_3. Superpose to each of them the estimated conditional means (corresponding to the regression lines in (2)) and the associated 2.5%-97.5% conditional quantiles of the fitted Gamma distributions for a rent. Are the fitted models satisfactory? Explain.

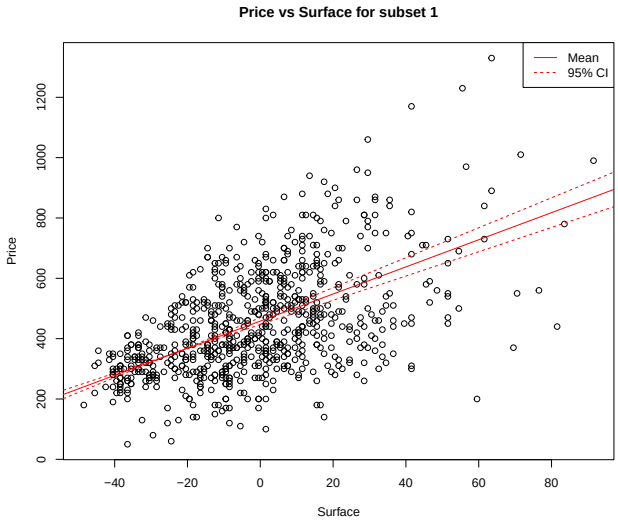


Figure 5: Without bathroom & Without kitchen

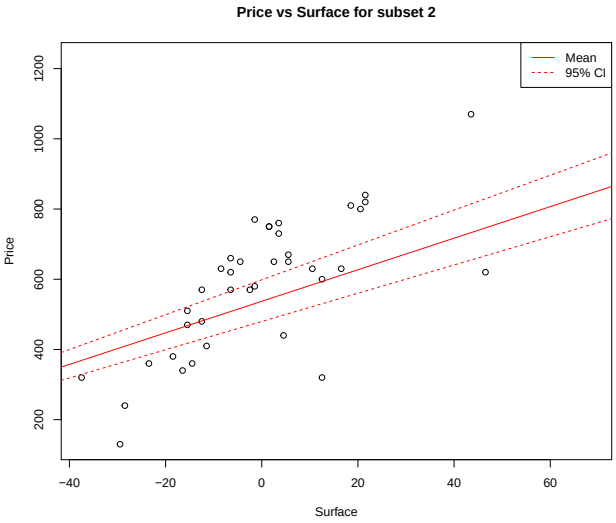


Figure 6: Without bathroom & With kitchen

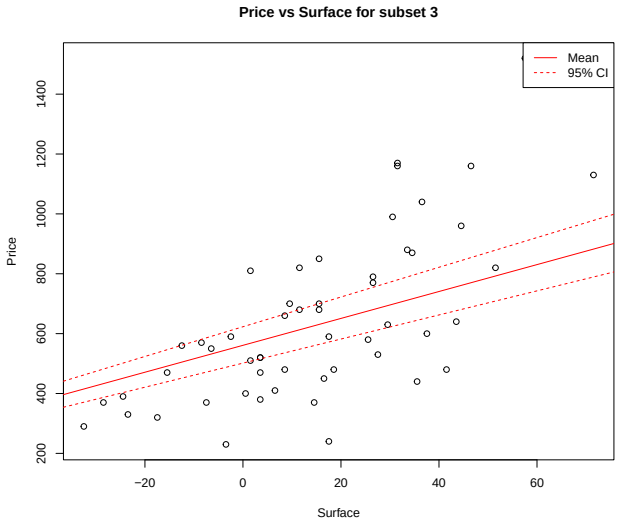


Figure 7: With bathroom & Without kitchen

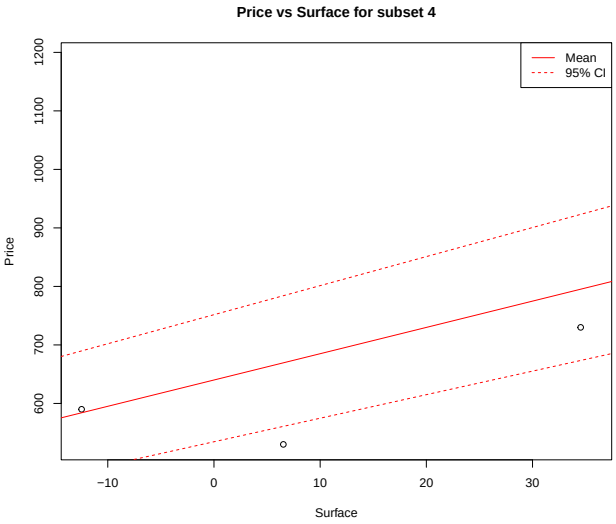


Figure 8: With bathroom & With kitchen

The fitted models are satisfactory and fit the data pretty well. They seem to encompass the relationships between price and surface well for the different assignments of the other parameters, namely the kitchen and bathroom booleans. We can observe that, when the number of training points decreases, the width of the confidence interval increases since we have more uncertainty added to the model.

3. Compare your results with those from Question 2.

- **Comparison of the parameters**
  In Table 10, we can see that the value of the parameters have changed between the first and the second model. While the β_0 and β_1 are still close, the third parameter is changing drastically. The main reason is the introduction of the fourth parameter into the model, which is also a boolean parameter. The addition of this parameter necessitates a reevaluation of the other parameters. The second reason is that JAGS is probably more optimized than the simple algorithm we implemented in section 2.

Parameter	estimator first model	estimator second model
β_0	462.47	458.10
β_1	4.55	4.49
β_2	85.54	79.11
β_3	-	102.84

Table 10: Comparison of the parameters of the two models

- **Comparison of the predictions**
  When we compare the predictions of the two models (cfr. Table 11), we can see that the first model is better at predicting the price of the house if we split the dataset into two parts : the house with and without bathroom. But, when we decide to use this model on the entire dataset, we can see that the performances are below those of the second model. The addition of the fourth parameter allows our model to have more degrees of freedom and to learn better. This more efficient learning allows the second model to have a better prediction of the renting prices.

  The metric used to compare the model is the *Root Mean Squared Error*. This score is a measure of the differences between values predicted by a model and the values actually observed. The formula for RMSE is:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

  Where:

  - n is the number of observations.
  - y_i is the actual value for the i-th observation.
  - $\hat{y}_i$ is the predicted value for the i-th observation.

  In words, RMSE is the square root of the average of the squared differences between the actual and predicted values.

Parameter	RMSE Without bathroom	RMSE With bathroom	Total RMSE
Model 1	0.63	31.81	1.54
Model 2	3.70	44.15	1.09

Table 11: Comparison of the errors of the predictions of the models on the dataset with bathroom, without bathroom and the entire dataset

4. Sample the predictive distribution of the price of a 80m² rent with a premium bathroom and a premium kitchen. Provide the histogram of the sampled values and report a point estimate and a credible interval for that prediction.

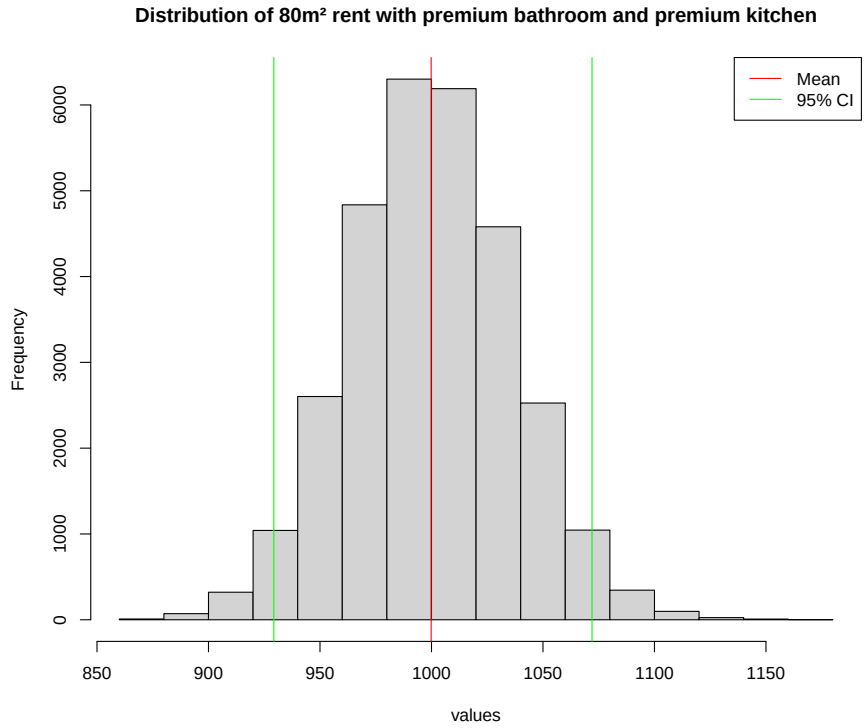


Figure 9: Predictive Distribution of 80m² rent

The mean of this distribution is 999.8504 and the 95% credible interval is $[929.2954, 1071.998]$.

1 Appendix

1.1 Annex 1.1

```r
log_likelihood <- function(params, data) {
  beta0 <- params[1]; beta1 <- params[2]; beta2 <- params[3]; psi <- params[4]

  phi <- exp(psi); phi_2 <- exp(2*psi); phi_minus_2 <- exp(-2*psi)
  mu <- beta0 + beta1 * data$surface + beta2 * data$bathroom

  prod1 <- gamma(phi_minus_2) * mu^(phi_minus_2) * phi^(2/phi_2)
  prod2 <- data$price^(phi_minus_2 - 1)
  prod3 <- exp(-data$price / (phi_2 * mu))

  likelihoods <- (1/prod1) * prod2 * prod3
  log_likelihood <- sum(log(likelihoods))

  return(log_likelihood)
}
```

1.2 Annex 1.2

```r
# Define the log-likelihood function for the gamma distribution
log_prior <- function(params) {
  beta0 <- params[1]
  beta1 <- params[2]
  beta2 <- params[3]
  psi <- params[4]

  prior_beta0 <- dnorm(beta0, mean = 0, sd = 100, log = TRUE)
  prior_beta1 <- dnorm(beta1, mean = 0, sd = 100, log = TRUE)
  prior_beta2 <- dnorm(beta2, mean = 0, sd = 100, log = TRUE)
  prior_psi <- dnorm(psi, mean = 0, sd = 100, log = TRUE)

  log_prior <- prior_beta0 + prior_beta1 + prior_beta2 + prior_psi

  return(log_prior)
}

# Define the log-posterior function for the gamma distribution
log_posterior <- function(params, data) {
  log_likelihood_value <- log_likelihood(params, data)
  log_prior_value <- log_prior(params)

  log_posterior <- log_likelihood_value + log_prior_value

  return(log_posterior)
}
```

1.3 Annex 1.3

```r
# Minimize the negative log-posterior to find the mode
init_params <- c(120, 0.3, 0.4, 0.2)
# random initial values for the parameters between 0 and 1

result <- optim(init_params, function(params) -log_posterior(params, data),
    hessian = TRUE)

# The mode of the posterior distribution is the parameter values that
    minimize the negative log-posterior
posterior_mode <- result$par

# The Hessian matrix at the mode is the inverse of the information matrix
information_matrix <- solve(result$hessian)
```

1.4 Annex 2.1

```r
# Load your data
data <-read.csv("./Renting.csv")
# Center the surface variable
min_surface <- max(data$surface)
data$surface <- data$surface - mean(data$surface)
# Minimize the negative log-posterior to find the mode
init_params <- c(min_surface, 0.5, 0.5, 0.2)
result <- optim(init_params, function(params) -log_posterior(params, data),
    hessian = TRUE)
# The mode of the posterior distribution is the parameter values that
    minimize the negative log-posterior
theta.hat <- result$par
```

```r
# The Hessian matrix at the mode is the inverse of the information matrix
Sigma <- solve(result$hessian)
M <- 10000
# Initial values
sd.prop = c(13.4, 0.57, 69, 0.068)
n.accept = c(0, 0, 0, 0)

cat(Sigma , "\n")
cat(theta.hat , "\n")
cat("Starting Metropolis algorithm...\n")

metropolis_algorithm <- function(params) {
  theta_result <- matrix(nrow = M, ncol = 4)
  for (i in 1:4) {
    theta <- matrix(nrow = M, ncol = 4)
    # theta[1, ] <- theta.hat
    theta[1, ] <- params
    theta.prop <- theta.hat
    theta.prop[i] <- 0
    for (m in 2:M) {
      # need to update the components one at a time

        theta.prop[i] <- theta[m - 1, i] + rnorm(1, 0, sd.prop[i])
        alpha <- log_posterior(theta.prop, data) - log_posterior(theta[m - 1, ], data)
        prob <- min(1, exp(alpha))

        # If the probability is NaN, set it to 0 (if the exponent is too large, the probability will be NaN)
        if(is.nan(prob)){
          prob <- 1
        }

        # Draw a random number
        u <- runif(1)

        # Update the parameter
        if (u <= prob) {

          theta[m, ] <- theta.prop
          n.accept[i] <- n.accept[i] + 1
        } else {
          theta[m, ] <- theta[m - 1, ]
        }
      }
    theta_result[, i] <- theta[, i]
  }
  # concatenate theta_result and accept
  acceptance_rate <- round(n.accept / (M-1), 2)
  cat("Acceptance rate:", acceptance_rate, "\n")
  return(theta_result)
}
```

1.5 Annex 2.2

```r
theta_result <- metropolis_algorithm(theta.hat)
```

```
plot(theta_result[, 1], type = "l", col = "blue", xlab = "Iteration", ylab =
    "Parameter value", main = "Trace plot of beta0")
plot(theta_result[, 2], type = "l", col = "blue", xlab = "Iteration", ylab =
    "Parameter value", main = "Trace plot of beta1")
plot(theta_result[, 3], type = "l", col = "blue", xlab = "Iteration", ylab =
    "Parameter value", main = "Trace plot of beta2")
plot(theta_result[, 4], type = "l", col = "blue", xlab = "Iteration", ylab =
    "Parameter value", main = "Trace plot of psi")

# Define chains with different starting values
theta_result1 <- metropolis_algorithm(c(500, 5.5, 200, -0.8))
theta_result2 <- metropolis_algorithm(c(410, 3.5, 0, -1.3))
theta_result3 <- metropolis_algorithm(c(460, 4.5, 100, -1.1))

gelman.diag(
  list(
    mcmc(theta_result1),
    mcmc(theta_result2),
    mcmc(theta_result3)
  )
)

gelman.plot(
  list(
    mcmc(theta_result1),
    mcmc(theta_result2),
    mcmc(theta_result3)
  )
)

# Compute effective sample size
effectiveSize(mcmc(theta_result))
```

1.6 Annex 2.3

```
# Point estimate and 95\% confidence interval
round(mean (theta_result[, 1]),3) ; round(quantile(theta_result[, 1],c
    (.025,.5,.975)),3)
round(mean (theta_result[, 2]),3) ; round(quantile(theta_result[, 2],c
    (.025,.5,.975)),3)
round(mean (theta_result[, 3]),3) ; round(quantile(theta_result[, 3],c
    (.025,.5,.975)),3)
round(mean (theta_result[, 4]),3) ; round(quantile(theta_result[, 4],c
    (.025,.5,.975)),3)
```

References

[1] E. Cepeda-Cuervo. “Modelagem de variabilidade em modelos lineares generalizados”. PhD thesis. Mathematics Institute, Universidade Federal Rio de Janeiro, 2001.

[2] Edilberto Cepeda-Cuervo et al. “On Gamma Regression Residuals”. In: *Journal of the Iranian Statistical Society* 15 (Jan. 2016), pp. 29–44. DOI: 10.7508/jirss.2016.01.002.