Petrophysical simulation based on parametric copula model under a Bayesian approach. Codes and notes

November 12, 2023

Daniel Vázquez-Ramírez [(1)], Van Huong Le [(2)], Martín Díaz-Viera [(3)], Raúl del Valle-García [(4)], Alfredo Garbuno-Iñigo[(5)]

(1) Posgrado de Ciencias de la Tierra, Universidad Nacional Autónoma de México, Circuito de Posgrado S/N, Coyoacán Ciudad Universitaria, 04510 Ciudad de México, Mexico, daniel.geofisico89@comunidad.unam.mx

(2) Department of Plant & Soil Sciences, University of Delaware, Newark, Delaware 19716, United States, levanhuong15011989@gmail.com

(3) Instituto Mexicano del Petróleo, Eje Central Lázaro Cárdenas No. 152, Col. San Bartolo Atepehuacan, México D.F. 07730, Mexico, mdiazv@imp.mx

(4) Independent consultant, Ciudad de México, México. raul.vontal@gmail.com

(5) Division of Actuarial Science, Statistics and Mathematics, Instituto Tecnológico Autónomo de México, Ciudad de México, México, alfredo.garbuno@itam.mx

1 Summary

The prediction of petrophysical properties from seismic data is one of the great challenges of geological-petrophysical modeling, since these models are used for decision-making in the reservoir development. However, a large part of the traditional methods uses the Gaussian-type simulation or requires linearizing the model. The following work will present a petrophysical simulation using the copula-based spatial stochastic co-simulation method modified under the Bayesian approach, separating the process into two parts: first, simulating the samples of the conditional petrophysical property on the seismic elastic attribute using the conditional parametric copula and second, doing the spatial simulation of the petrophysical property using the simulated annealing method. This perspective offers better results by using functions that allow a more realistic representation of the variable, instead of forcing it to be a linearized and/or a Gaussian representation.

2 Introduction

The prediction of petrophysical properties has been one of the great problems in the of reservoir geological-petrophysical modeling workflow [Cosentino, 2001], especially when considering the dependency relationship with other properties, such as seismic attributes. However, the approach of predicting a petrophysical property from a set of seismic attributes is sometimes called petrophysical inversion [Grana et al., 2022]. Most inversion and simulation approaches try to force using

normal probability distribution function because the method is based on Gaussian simulation as shown in the works by [Bortoli et al., 1993], [Haas and Dubrule, 1994], [Buland and Omre, 2003], [Dubrule et al., 2003], [Doyen, 2007], [Menke, 2012], [Grana, 2014] or direct sequential simulation proposed by [Azevedo and Soares, 2017].

The disadvantage raised in the aforementioned methodologies can be solved through the copula-based spatial stochastic co-simulation method [**?**]. This method can be basically divided into two steps. Firstly, a dependence model between the primary variable (petrophysical property) and secondary variable (seismic attribute) is established by estimating and modeling the joint cumulative probability distribution function (CPDF) using a copula. The CPDF model is used in conjunction with a variogram (spatial correlation) model to simulate the primary variable using the second one as a conditioning variable. This method has been presented in the works by [**?**], [Le, 2021], [Le et al., 2020], [Ramírez, 2018]. However, these works have not been tested under an environment by updating the information from nearby wells or seismic attributes at the site. Henceforth, by using the Bayesian inference approach on the joint probability distribution functions will allow preparing those models to relate the information from other sources within the area.

So the first step is to modify the copula-based spatial stochastic co-simulation method by using the estimated copula obtained from Bayesian inference. This new proposal seeks to generate a prior well-scale model between a petrophysical property and a seismic attribute using parametric functions. The parameters of the model will be estimated later using the same petrophysical property and the same seismic attribute scaled by the Backus method for the seismic attribute and a moving average window for the petrophysical property as new information, in order to evaluate the calibration. The posterior model will then be used to simulate samples of the petrophysical property conditioned by the seismic attribute. To assess the quality of the simulations, these samples will be compared in the second step to the estimated variogram of the well-scale petrophysical property. If the proposal offers good well-scale simulations, then this model is chosen to estimate new parameters values using, as new information, the seismic attributes under the trace-by-trace scheme, in order to predict the petrophysical property. Works related to Bayesian inference using copulas can be found in the economics area [Min and Czado, 2010], [Silva and Lopes, 2008], [Kohn et al., 2006], [Schamberger et al., 2017], market risk analysis [Ausin and Lopes, 2010], [Armstrong et al., 2004] and medical area [Saraiva et al., 2018]. Only the proposal by [Armstrong et al., 2004] is related to oil industry. The paper seeks to relate the well oil production and the oil prices in the market and then control the oil production using the new information about oil price.

The paper has the following structure. First, a brief description of Bayesian estimation using copulas will be given, in which the parametric, semi-parametric and non-parametric approaches are discussed. Then the use of parametric copulas in Bayesian inference is described, as well as how to build the prior, likelihood and posterior functions considering one-step or two-step approaches. A workflow is presented for the well-scale model calibration case and its comparison with the deterministic case, followed by the case where the petrophysical property is simulated at the seismic scale.

3 Bayesian copula estimation.

To implement the Bayesian inference in copula estimation, is neccessary to start from Bayes formula.

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(B|A)\mathbb{P}(A)}{\mathbb{P}(B)} \tag{1}$$

If A changes to $\mathbf{m}$, it is possible to use probability density function $\mathbb{P}(\mathbf{m})$, where $\mathbf{m}$ is a representation of the parameter space and B can be related to the observations $\mathbf{d}$ [Bernardo and Smith, 2009]. Gives the following relationship between the posterior density function $\mathbb{P}(\mathbf{m}|\mathbf{d})$ expressed in Bayes's theorem.

$$\mathbb{P}(\mathbf{m}|\mathbf{d}) = \frac{\mathbb{P}(\mathbf{d}|\mathbf{m})\mathbb{P}(\mathbf{m})}{\int \mathbb{P}(\mathbf{d}|\mathbf{m})\mathbb{P}(\mathbf{m})d\mathbf{m}}, \tag{2}$$

where $\mathbf{m} = m_1, ..., m_i$, $\mathbf{d} = d_1, ..., d_j$, $\mathbb{P}(\mathbf{m})$ corresponds to the set of prior distribution functions, $\mathbb{P}(\mathbf{d}|\mathbf{m})$ is the likelihood of the observations $\mathbf{d}$ given a model parameter $\mathbf{m}$. The denominator is known as the marginal likelihood, which shows how the observations should behave given the model before the observations are sampled; which is often not so easy to obtain. To solve the problem with the marginal likelihood, a proportionality constant $\mathbf{c}$ is proposed, with this (2) changes to

$$\mathbb{P}(\mathbf{m}|\mathbf{d}) = \frac{\mathbb{P}(\mathbf{d}|\mathbf{m})\mathbb{P}(\mathbf{m})}{\mathbf{c}}. \tag{3}$$

If $\mathbf{c}$ is removed in (3), the relationship changes from equal ($=$) to proportional ($\propto$).

$$\mathbb{P}(\mathbf{m}|\mathbf{d}) \propto \mathbb{P}(\mathbf{d}|\mathbf{m})\mathbb{P}(\mathbf{m}). \tag{4}$$

Now, to link the function (4) with the copula it is neccesary to use its joint probability density function. Under Sklar's theorem, the d-dimensional density function f with univariate marginals $F_1, ..., F_d$ and its corresponding univariate density functions $f_1, ..., f_d$ can be represented as [Jaworski et al., 2010]

$$f(x_1, ..., x_d) = c(F_1(x_1), .., F_d(x_d); \theta) \prod_{i=1}^{d} f(x_i). \tag{5}$$

where c is the density function of the d-dimensional copula. To solve the joint probability density function (5) in terms of Bayesian inference, three estimation methodologies can be used: non-parametric, semi-parametric and parametric.

The non-parametric approach uses a non-parametric estimator for the copula and the margins [Jaworski et al., 2010], the copula is estimated using its empirical copula and the margins are interpolated using Bernstein polynomials, splines or Gaussian mixtures as proposed by [Valle et al., 2017].

The parametric approach uses parametric marginals and parametric copulas, the advantage of using this approach is the low computational cost and easy implementation, the disadvantage is the limitation of the dependence measure. Parametric copulas are widely developed for positive dependences, just five families can be used with negative dependences: Gaussian, Plackett, Frank, reflected MTCJ and $t-$ copula [Joe, 2014].

The parametric copulas are classified as: elliptical, archimedean and extreme values.

The elliptical copulas are used when the marginal becomes from elliptical distribution functions,for example, the Gaussian and Student distribution; and the structure of dependence has elliptical

symmetry respect to the diagonal. Some examples of elliptical copulas are Gaussian and $t-$ copula [Shemyakin and Kniazev, 2017].

The extreme value copulas are used when the marginals and copula have been fitted with generalized extreme value distribution functions. For example, the marginal Gumbel, Frechet and Weibull belong to this classification, and the Galambos, Hüstle-Reis and Tawn are the joint distribution function [Hofert et al., 2019].

Archimedean copulas are the most used copulas because they are easier for implementation, they just need one parameter θ to control the dependence and the differentiation and integration to obtain the generator and inverse is simple [Shemyakin and Kniazev, 2017].

$$C(u,v) = \varphi^{[-1]}(\varphi(u) + \varphi(v)) \tag{6}$$

The function 6 is the Archimedean copula. Where φ function is called copula generator. If $\varphi(0) = \infty$ then φ is strictly generator, in that case $\varphi^{[-1]} = \varphi^{(-1)}$ and $C(u,v) = \varphi^{[-1]}(\varphi(u) + \varphi(v))$ and is said to be strictly an Archimedean copula. Another important property is the Archimedean density copula expressed in equation 7, this can be expressed through the generator and its derivatives as [Shemyakin and Kniazev, 2017]

$$c(u,v) = \frac{\partial^2 C(u,v)}{\partial u \partial v} = \frac{\varphi''(C(u,v))\varphi'(u)\varphi'(v)}{(\varphi''(C(u,v)))^3} \tag{7}$$

The semi-parametric approach is an option where the non-parametric estimator is applied in the margins or the copula, the most common case is the margins are estimated with the empirical function and the copula is estimated with the parametric estimator, examples of this approach are available in the works by [Al Labadi et al., 2019], [Rosen and Thompson, 2015]; one example where the copula generator φ is estimated using cuadratic splines is in [Hoyos-Argüelles and Nieto-Barajas, 2019].

4 Bayesian inference estimation in parametric copulas

As see in Eq. 4, Bayesian inference requires two elements: prior and likelihood

The advantage using the joint probability distribution function (5) under the Bayesian approach is the great flexibility with the information that can be considered as prior. For example, the work published by [Silva and Lopes, 2008] only selects two gamma distributions to represent the marginals and chooses between the Gaussian copula or Student copula, the model selection criteria is based almost in the data. The work of [Smith, 2011] proposes hierarchical priors for Gaussian copula correlation matrix. [Ausin and Lopes, 2010], [Min and Czado, 2010] and [Kohn et al., 2006] use Cauchy and uniform functions as prior, even non-parametric functions have been proposed as prior [Grazian and Liseo, 2016].

$$\mathbb{P}(\mathbf{m}) = \pi(\theta, \alpha_i) \tag{8}$$

Where θ is the copula parameter and α_i are the parameters of the marginals.

4.1 Likelihood function

To obtain the likelihood $\mathbb{P}(\mathbf{d}|\mathbf{m})$ from equation (5), it is possible to apply the logarithm and then obtain the log-lokelihood L_{log} following the next steps [Hofert et al., 2019].

$$L_{log} = \sum_{j=1}^{n} log\ f\left(x_1^{(j)}, ..., x_d^{(j)}\right) = L_c + \sum_{i=1}^{d} L_i \tag{9}$$

From (9), $\left(x_d^{(j)}\right)$ are the random samples represented as d-vectors, L_c is the log-likelihood of the dependency structure represented by the copula and its expressed as:

$$L_c = \sum_{j=1}^{n} log\ c\left(F_1\left(x_1^{(j)}\right), .., F_d\left(x_d^{(j)}\right)\right) \tag{10}$$

And L_i is the log-likelihood of each marginal, which are:

$$L_i = \sum_{j=1}^{n} log\ f_i\left(x_i^{(j)}\right) \tag{11}$$

The parameter model associated with Eq 10 and Eq 11 is composed of the parameters α_d of the univariate density functions f_i and the parameter of the associated copula θ is $(\alpha_1, \alpha_2, ..., \alpha_d, \theta)$, whereby (9) expands as:

$$L_{log}(x_i|\alpha_d, \theta) = \sum_{j=1}^{n} log\ c\left(F_1\left(x_1^{(j)}, \alpha_1\right), .., F_d\left(x_d^{(j)}, \alpha_d\right), \theta\right) + \sum_{i=1}^{d}\sum_{j=1}^{n} log\ f_i\left(x_i^{(j)}, \alpha_i\right) \tag{12}$$

To estimate the most probable values of each parameter, [Shemyakin and Kniazev, 2017] mention two approaches:

Estimation in one step. That is, calculate all the parameters in a single calculation. The adjustment can be made by maximizing the marginal parameter α_i and the copula parameter θ in the same process, that is, [Jaworski et al., 2010]

$$L_{log}(x_i|\alpha_d, \theta) = arg\ max_{(\alpha_1,...,\alpha_d,\theta)} \sum_{j=1}^{n} log\left(c\left(F_1\left(x_i^{(j)}; \alpha_1\right), ..., F_d\left(x_d^{(j)}; \alpha_d\right); \theta\right)\right) + \sum_{i=1}^{d}\sum_{j=1}^{n} log f_i\left(x_i^{(j)}; \alpha_i\right) \tag{13}$$

As can see, Eqn. 13 allows all the parameters to be estimated simultaneously, but advanced algorithms are required to maximize the likelihood; Among these methods, those of Monte Carlo stand out, be it Metropolis-Hastings, random walk, etc.

Two-step estimation. This consists of dividing the work; first estimate the parameters of the marginals and then estimate the copula, the best known method with this approach is the inference of functions for marginals (IFM) proposed by [**?**]. This uses the following likelihood L:

$$L(x_1,...,x_i|\alpha_1,...,\alpha_d,\theta) = c\left(F_1\left(x_i;\alpha_1\right),...,F_d\left(x_d;\alpha_d\right);\theta\right)\prod_{i=1}^{d} f_i\left(x_i;\alpha_i\right) \tag{14}$$

In the case of Eq 14 it can notice an advantage, the producer can be separated from the density function of the copula, therefore the marginal parameters can be estimated independently of the copula parameter, that is, first can get the log-likelihood of the marginals [Hofert et al., 2019]

$$L_i(\alpha_i) = \sum_{i=1}^{n} log\ f_i\left(x_i;\alpha_i\right) \quad i = 1,...,d \tag{15}$$

After obtaining the most probable values, this information is entered into the log-likelihood function of the joint distribution or copula:

$$L(\theta,\alpha_1,...,\alpha_d) = \sum_{i=1}^{n} log\ f_i\left(x_i;\alpha_1,...,\alpha_d,\theta\right) \tag{16}$$

In terms of computational efficiency, the IFM method turns out to be very efficient, since it allows the estimation to be separated into two parts.

4.2 Posterior function

Obtained the likelihood function (Eq 9) and the a priori functions, it is possible to define the posterior function as:

$$\mathbb{P}(\mathbf{m}|\mathbf{d}) \propto \pi(\theta,\alpha_i) L_{log}(x_i|\alpha_d,\theta) \tag{17}$$

If develop the Eq. 17 in terms of the logarithm to facilitate the computational implementation, it is possible to obtain the following equation:

$$log(f(\theta,\alpha_i|x_i)) \propto log(\pi(\theta,\alpha_i)) + \sum_{j=1}^{n} log\left(c\left(F_1\left(x_i^{(j)};\alpha_1\right),...,F_d\left(x_d^{(j)};\alpha_d\right);\theta\right)\right) + \sum_{i=1}^{d}\sum_{j=1}^{n} log f_i\left(x_i^{(j)};\alpha_i\right) \tag{18}$$

To solve the Eq. 18 under the traditional approach, conjugate distribution functions would have to be used, which results in a posterior distribution with the same functional as the initial distribution. For these cases, the use of the posterior maximum probability method can be proposed.

Another alternative is the Monte Carlo methods, especially those based on stochastic simulation. In particular, the random walk Metropolis-Hastings.

4.3 Metropolis-Hastings algorithm

Of all the varieties of algorithms based on the Monte Carlo method with Markov chains, we will analyze the Metropolis-Hastings algorithm. This algorithm works as follows [Kroese et al., 2013]: Suppose we want to generate samples from a multivariate distribution function.

$$f(x) = \frac{\mathbb{P}(x)}{Z} \tag{19}$$

Where $\mathbb{P}(x)$ is a known positive function and Z is a known or unknown normalized constant. Let $q(y|x)$ be a proposed conditional density function; to sample the known density function $f(x)$ to a normalization constant, it is proposed to start with some $\mathbf{X}_0$ such that $f(\mathbf{X}_0) > 0$. Then for each $t = 0, 1, 2, ..., T-1$ the following steps are followed:

- Given the Markov chain $\mathbf{X}_t$, generate $\mathbf{Y} \sim q(y|\mathbf{X}_t)$
- Generate $U \sim U(0,1)$ and deliver

$$\mathbf{X}_{t+1} = \begin{cases} \mathbf{Y} & with \quad probability \quad \alpha(\mathbf{X}_t, \mathbf{Y}) \\ \mathbf{X}_t & with \quad probability \quad 1 - \alpha(\mathbf{X}_t, \mathbf{Y}) \end{cases} \tag{20}$$

Where

$$\alpha(\mathbf{x}, \mathbf{y}) = min\left\{ \frac{f(y)q(x|y)}{f(x)q(y|x)}, 1 \right\} \tag{21}$$

The probability $\alpha(x, y)$ is called the **acceptance probability**. This probability is the most important property of the Metropolis-Hastings algorithm, since the sampling only depends on this acceptance range. The recommended range is within 20 to 50%, if the acceptance range is higher it means that the algorithm is not moving fast enough in the model space; if it is lower, we are wasting computational resources by testing models that are not accepted [Tarantola, 2005].

Random walk sampling

This particular case of the Metropolis-Hastings algorithm requires that the proposed density function be symmetric, that is, $q(y|x) = q(x|y)$, thus the acceptance probability in Eq. 22 changes to [Kroese et al., 2013]:

$$\alpha(\mathbf{x}, \mathbf{y}) = min\left\{ \frac{f(y)}{f(x)}, 1 \right\} \tag{22}$$

5 Method application workflow

The workflow follows the next steps:

- *Exploratory data analysis* this step includes univariate and bivariate analysis. The objective is to evaluate the properties of samples with statistics, histograms and scatter plots
- *Variographic analysis* In this step the samples are used to estimate the semivariogram, and then, the semivariogram is fitted to the spatial structure of dependence (variogram). This work considers two variogram models, the first model is obtained with the semivariogram of the upscaled samples in depth domain. The second model is obtained with the semivariogram

of subsampled petrophysical property in time domain. Both models are used to simulate the petrophysical property in their domain.

- *Parametric copula estimation with prior information.* In this step, the petrophysical property and seismic attribute well log are prior information. The samples are fitted to some probability distribution function, two belong to univariate samples and one is the joint distribution function between the univariate samples.
- *Build posterior function.* The posterior function is built with the prior parameter values obtained with fitted distribution function. The new information is the upscale well log for depth domain and subsampled well log for time domain.
- *Estimate the posterior parameter values using random walk Metropolis-Hastings.* The search function considers the parameter values are associated to a distribution function, some authors use normal or uniform distribution function. The probability of acceptance is between 20% to 30%
- *Simulate the petrophysical property using simulated annealing.* The simulation is performed with the posterior dependence model and variogram model. The simulations of petrophysical properties are conditioned to seismic attribute. In depth domain the upscaled seismic attribute is used, In time domain the subsampled seismic attribute is used, if the result is acceptable, the simulations are performed using seismic attribute from seismic model.

Statistical analysis was performed using the Rgeoestad2D package [Díaz-Viera et al., 2021], Bayesian analysis used the LearnBayes package [Albert, 2009].

6 Integration of upscaled data to the joint probability distribution function model

The data samples are the acoustic impedance (Ip) and total porosity (ϕ_t) obteined from a well dilled in the deep water gulf of México. This well crosses the Miocene turbiditic system. This system is important due to unassociated fields discovered in this unit; the unit is divided into three sections: Lower Miocene, Middle Miocene and Upper Miocene. The age of interest of this implementation is the Lower Miocene, in this zone the flow goes from southwest to northeast with some dominant channel systems. The rocks are medium-grained sandstones; evidence of volcanic material, feldspars, quartz and some metamorphic fragments were also found. The rock is poorly consolidated and mineralogically immature; the porosity ranges from 12 to 28% [Arreguin-Lopez et al., 2011]. This data samples were upscale using two methods: for total porosity (ϕ_t) the moving averaging technique was used, meanwhile the acoustic impedance Ip was estimated with the results obtained for P Velocity (Vp) and density (ρ) using Backus upscaling method. Both methods uses a vertical resolution of 28.44 meters.

[ ]:
```
install.packages("Rcpp")
install.packages("maps")
install.packages("mapproj")
install.packages("actuar")
install.packages("fields")
install.packages("fitdistrplus")
install.packages("geoR")
install.packages("gstat")
install.packages("MASS")
```

```
install.packages("moments")
install.packages("poweRlaw")
install.packages("RFOC")
install.packages("spatstat")
install.packages("ADGofTest")
install.packages("reshape")
install.packages("car")
install.packages("copula")
install.packages("LearnBayes")
install.packages("modeest")
install.packages("RSEIS")
install.packages("GenSA")
install.packages("NMOF")
```

[2]:
```
root_dir<-getwd()
setwd(root_dir)


#### Load Packages ####
library(Rcpp)
library(maps)
library(mapproj)
library(actuar)
library(fields)
library(fitdistrplus)
library(geoR)
library(gstat)
library(MASS)
library(moments)
library(poweRlaw)
library(RFOC)
library(spatstat)
library(ADGofTest)
library(reshape)
library(sp)
library(car)
library(copula)
library(modeest)
library(LearnBayes)
```

[31]:
```
root_dir<-getwd()

function_dir<-paste(root_dir,"/Functions",sep="")

setwd(function_dir)

source("AllModel.R", encoding='ISO-8859-1')
```

```
source("BestModel.R")
source("CDF.R")
source("CoKrigingOrd.R")
source("CoKrigingOrdAnis.R")
source("CrossValidation.R")
source("CrossValidation2.R")
source("CrossVariograma.R")
source("DEspacial.R", encoding='ISO-8859-1')
source("depurar.xy.R")
source("Estadisticas.R")
source("EyeModel.R", encoding='ISO-8859-1')
source("FitDistribution.R", encoding='ISO-8859-1')
source("GDirecciones.R", encoding='ISO-8859-1')
source("HistBoxplot.R")
source("HistModel.R")
source("hist2.R")
source("KrigingOrd.R", encoding='ISO-8859-1')
source("KrigingOrdAnis.R", encoding='ISO-8859-1')
source("LogPlot.R")
source("ModelVariogram.R")
source("MedianReg.R")
source("nscore.R")
source("OutliersPos.R")
source("PPplot.R")
source("QQplot.R")
source("ScatterPlot.R")
source("scaterplotReg.R")
source("simula.parametric.condicional1.R")
source("SGS.R")
source("Tendencia.R")
source("Trend.R")
source("Val_Estadisticos.R", encoding='ISO-8859-1')
source("Validacion.R", encoding='ISO-8859-1')
source("Variograma.R", encoding='ISO-8859-1')
source("Variograma4D.R", encoding='ISO-8859-1')

setwd(root_dir)
```

Load the data samples upscaled

[3]:
```
Data_like_UPS <- read.csv(file='data/upscaling_TWT_3696_sample_Lakach-1.
↪csv',header=T,na.strings="-999.25")
attach(Data_like_UPS)
```

```
The following objects are masked _by_ .GlobalEnv:

    Ip, Vp
```

and load the data samples obtained from well log

[4]:
```
Data_File <- read.csv(file='data/Lakach-1_3035.0-3404.5_25-05-2017.
 ↪csv',header=T,na.strings="-999.25")
attach(Data_File)
```

```
The following objects are masked _by_ .GlobalEnv:

    Ip, Vp


The following objects are masked from Data_like_UPS:

    DEPTH, DTCO, Ip, PHIT, RHOB, Vp, Vs
```

[5]:
```
Depth<-Data_File$DEPTH
Phit<-Data_File$PHIT          #Total porosity
Ip<-Data_File$Ip              #acoustic impedance
```

[6]:
```
Phit_like<-Data_like_UPS$PHIT_MA    #Total porosity upscaled
Ip_like<-Data_like_UPS$Ip_ups_Bks   #acoustic impedance upscaled
time_UPS<-Data_like_UPS$TWT

Data_like<-cbind(Ip_like,Phit_like)
```

calculate the statistics

[7]:
```
Depth_Stat<-Estadisticas(Depth)
Phit_Stat<-Estadisticas(Phit)  #total porosity
Ip_Stat<-Estadisticas(Ip)      #acoustic impedance
```

```
Warning message:
"encountered a tie, and the difference between minimal and
                  maximal value is > length('x') * 'tie.limit'
the distribution could be multimodal"
```

[8]:
```
Phit_like_Stat<-Estadisticas(Phit_like)
Ip_like_Stat<-Estadisticas(Ip_like)
```

The histograms use bin number estimated with Sturges method

[9]:
```
ns = nclass.Sturges(Ip)
ns
```

13

6.1 Exploratory data analysis

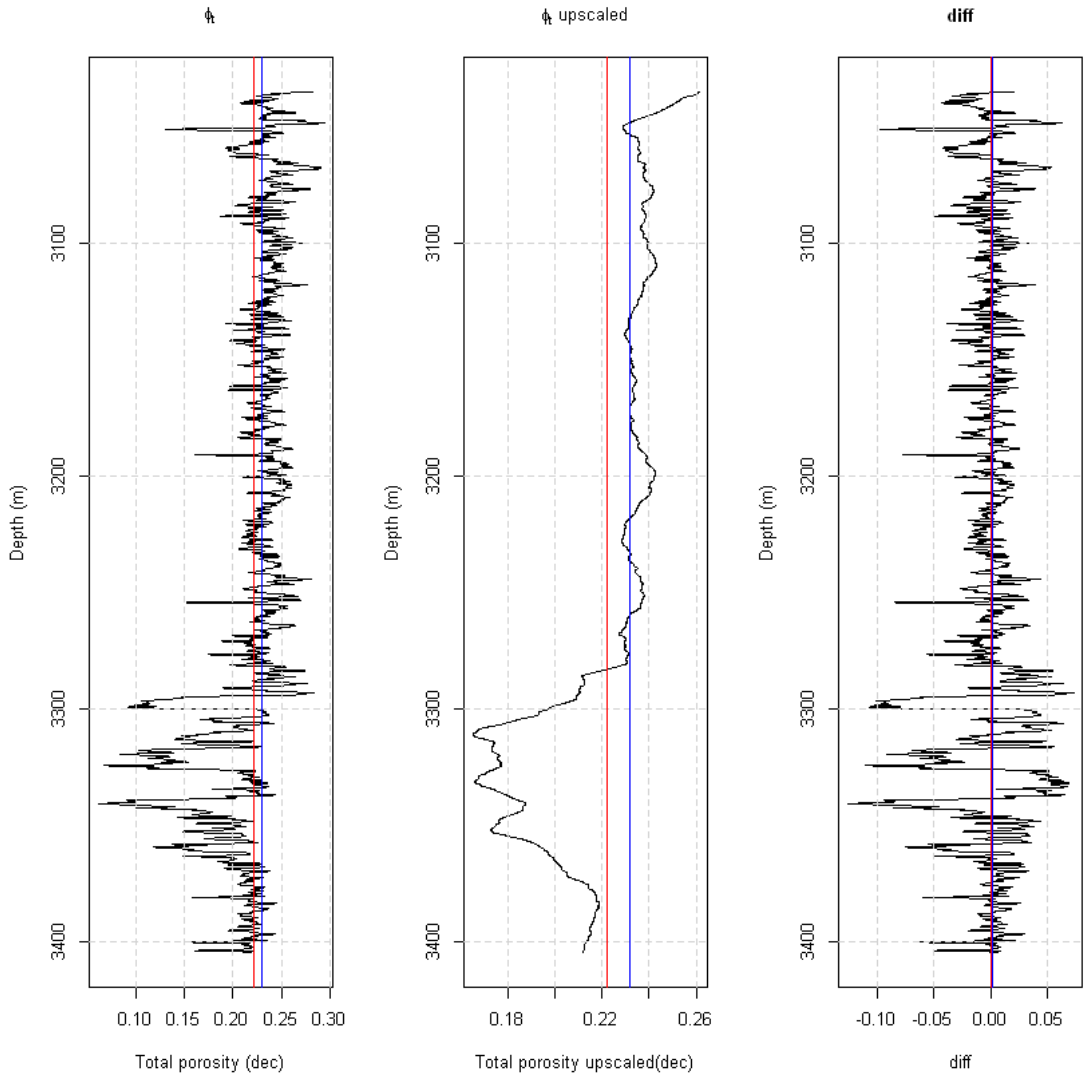
The table 1 shows the statistics of data samples and upscaled samples. Comparing both samples, the statistics show than all statistics has difference between 8%-12%, but the mean and median are relativity close. The range, minimum and maximum has a big difference this is due to upscaling.

[10]:
```
Comparative_like<-cbind(Ip,Ip_like,Phit,Phit_like)
Comparative_like_Stat<-Val_Estadisticos(Comparative_like)

colnames(Comparative_like_Stat)<-c("Acoustic impedance","Acoustic impedance␣
↪upscaled",
                                   "Total porosity","Total porosity upscaled")
Comparative_like_Stat
```

	Acoustic impedance <dbl>	Acoustic impedance upscaled <dbl>	Total porosity <dbl>	Total porosity upsca <dbl>
No_muestras	3.696000e+03	3696.00000	3696.00000	3696.00000
Minimo	5.086007e+03	5835.09689	0.06200	0.16550
Cuartil_1er	6.157005e+03	6178.11814	0.21470	0.21255
Mediana	6.809557e+03	6831.46526	0.22950	0.23195
Media	6.968284e+03	6892.89846	0.22197	0.22209
Cuartil_3er	7.321617e+03	7353.49049	0.24140	0.23687
Maximo	1.166146e+04	8891.21238	0.29390	0.26107
Rango	6.575457e+03	3056.11549	0.23190	0.09557
Rango_Intercuartil	1.164612e+03	1175.37236	0.02670	0.02431
Varianza	1.310303e+06	708798.26412	0.00114	0.00050
Desv_Estandar	1.144685e+03	841.90158	0.03376	0.02227
Simetria	1.561030e+00	0.59673	-1.81022	-1.21389
Curtosis	5.765750e+00	2.18977	6.99698	3.24141

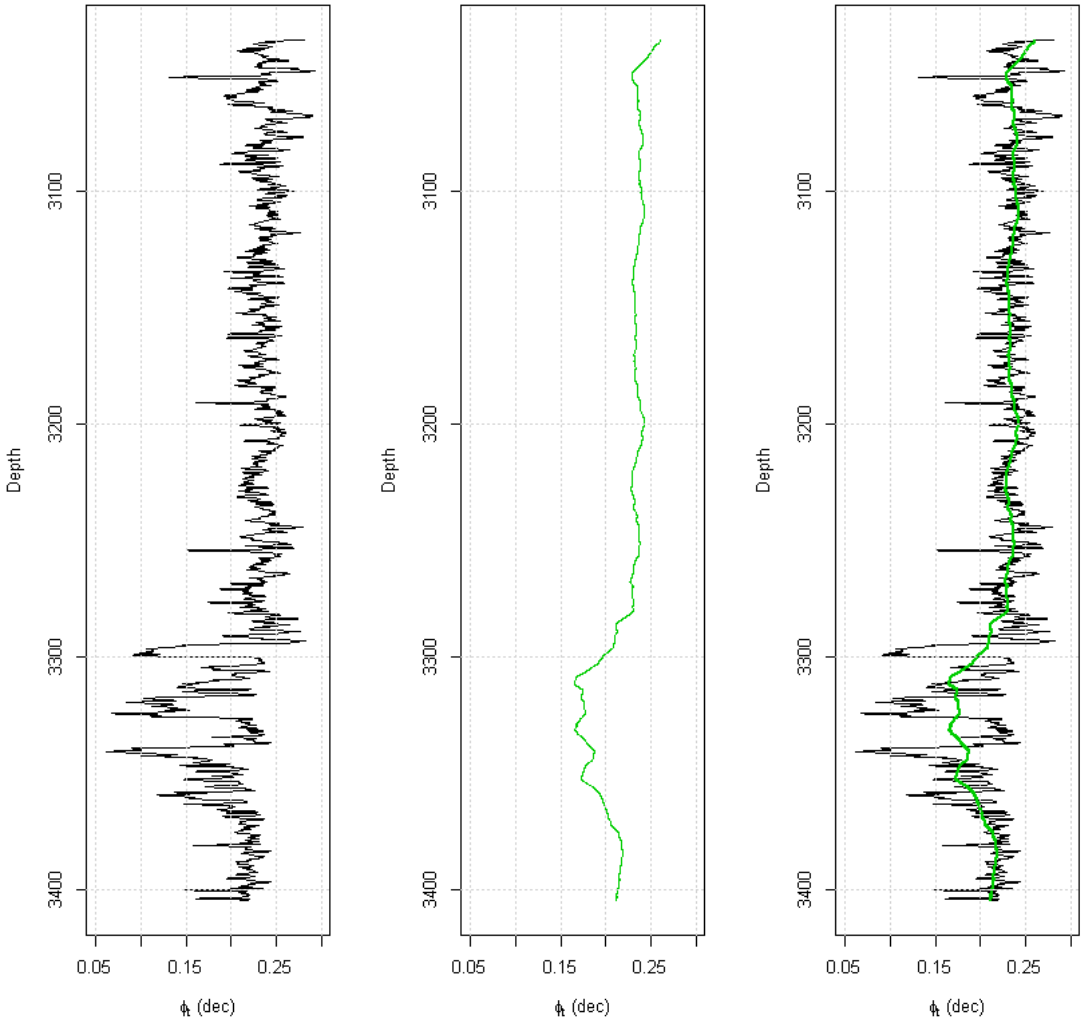
The difference between total porosity well log and total porosity upscaled is $\pm$ 0.05, specially before 3300 meters. Some small intervals has differences over 0.05.

[6]:
```
diff_Phit<-Phit-Phit_like
LogPlot(depth=Depth, n_logs=3,
        log_list=list(Phit,Phit_like,diff_Phit),
        log_name_list=list(expression(paste(phi[t])),expression(paste(phi[t],"␣
↪upscaled")),"diff"),
        log_xlabel_list=list("Total porosity (dec)","Total porosity␣
↪upscaled(dec)","diff"),
        plot_mean=TRUE, plot_median=TRUE)
```



[3]:
```
par(mfrow = c(1,3))
plot(Phit,Depth,xlim=c(0.05, 0.3),xlab = expression(paste(phi[t], "␣
↪(dec)")),ylab="Depth",ylim=rev(range(Depth)),
     type = "l",lwd=1,col=("black"))
grid()

plot(Phit_like,Depth,xlim=c(0.05, 0.3),xlab = expression(paste(phi[t], "␣
↪(dec)")),ylab="Depth",ylim=rev(range(Depth)),
     type = "l",lwd=1,col=("green3"))
grid()
```

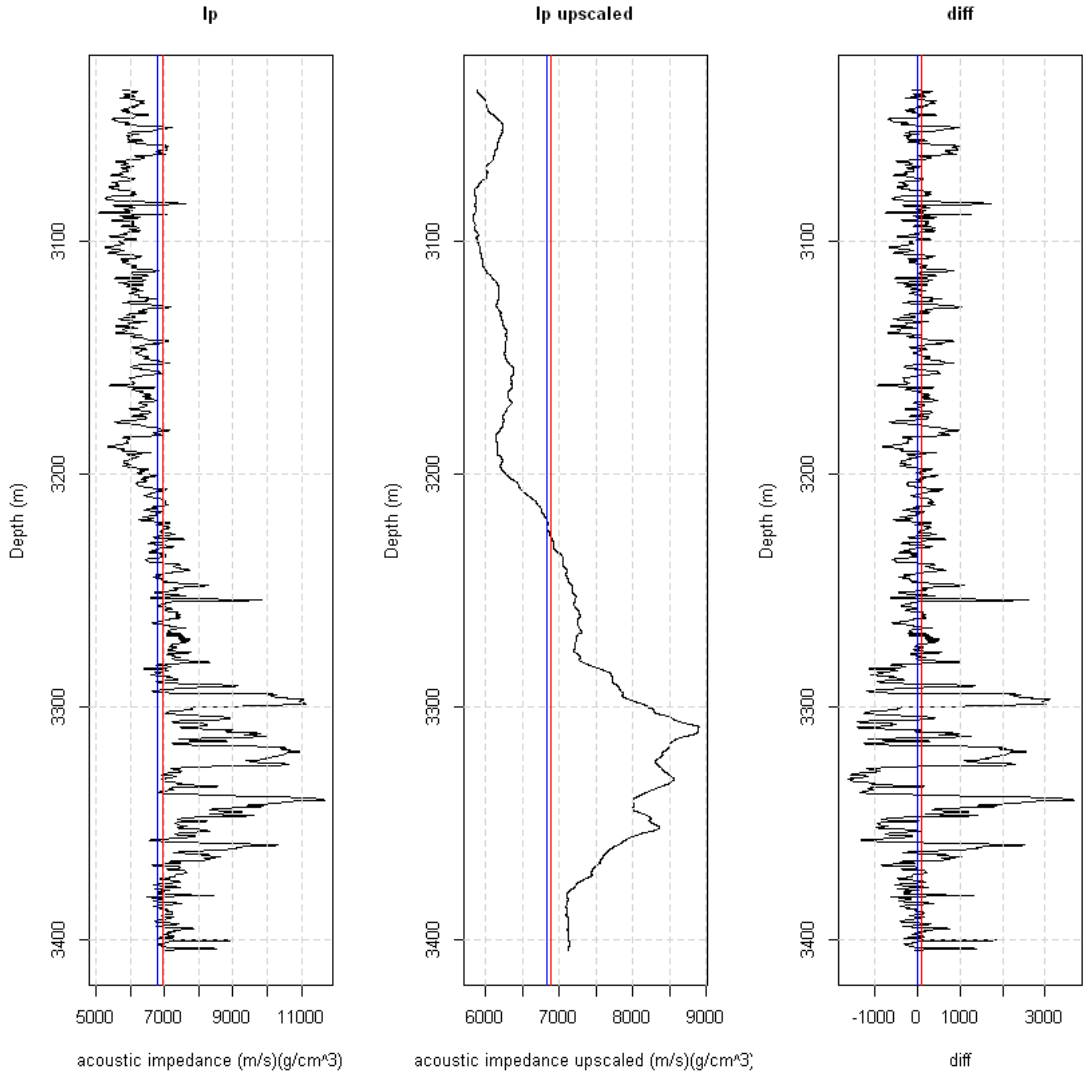
```
plot(Phit,Depth,xlim=c(0.05, 0.3),xlab = expression(paste(phi[t], "␣
↪(dec)")),ylab="Depth",ylim=rev(range(Depth)),
     type = "l",lwd=1,col=("black"))
par(new=TRUE)
plot(Phit_like,Depth,xlim=c(0.05, 0.3),ylim=rev(range(Depth)),type =␣
↪"l",lwd=2,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
grid()
```



The difference between acoustic impedance well log and upscaled is ± 1000, just a few parts have a difference over 1000.

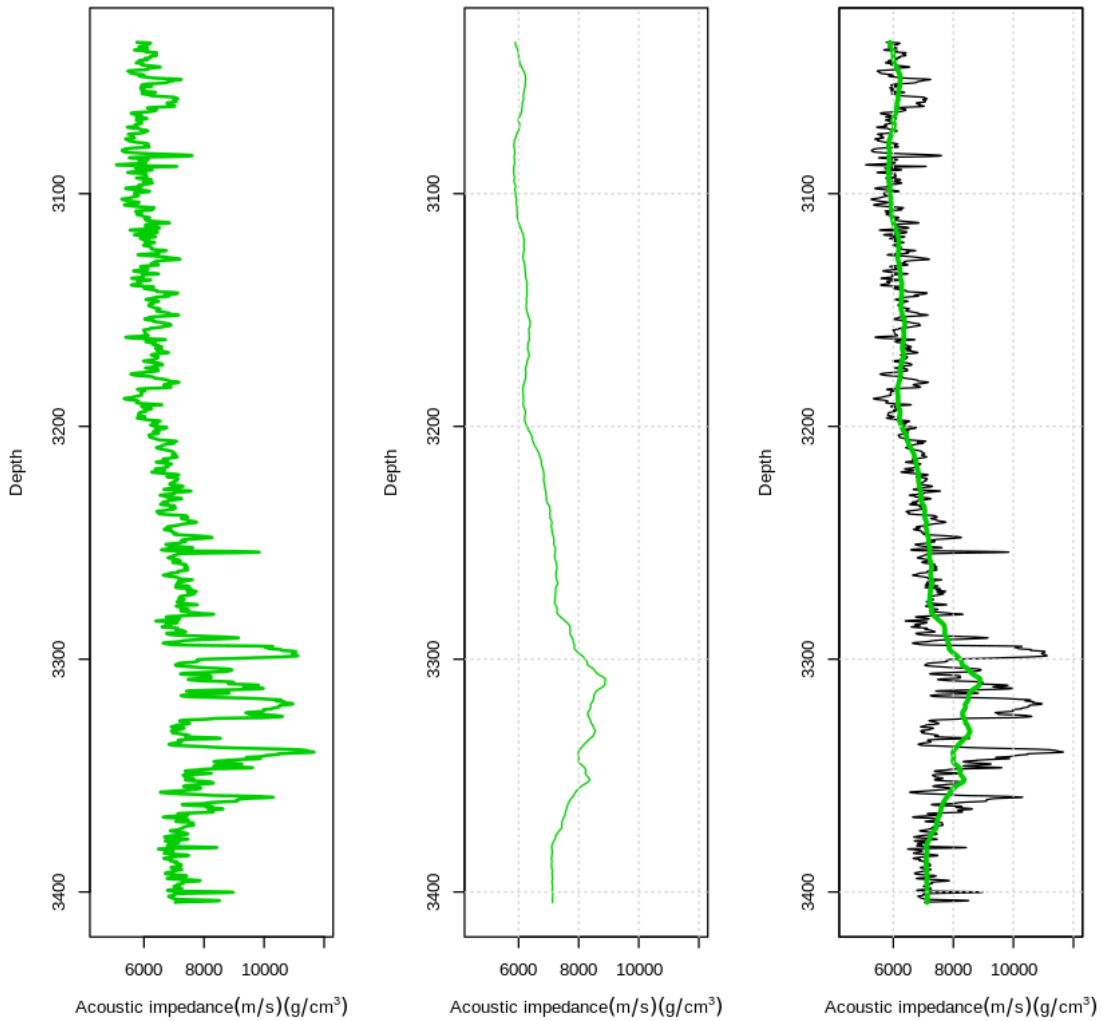
[17]:
```
diff_Ip<-Ip-Ip_like
LogPlot(depth=Depth, n_logs=3,
```

```
       log_list=list(Ip,Ip_like,diff_Ip),
       log_name_list=list("Ip","Ip upscaled","diff"),
       log_xlabel_list=list("acoustic impedance (m/s)(g/cm^3)","acoustic␣
↪impedance upscaled (m/s)(g/cm^3)","diff"),
       plot_mean=TRUE, plot_median=TRUE)
```



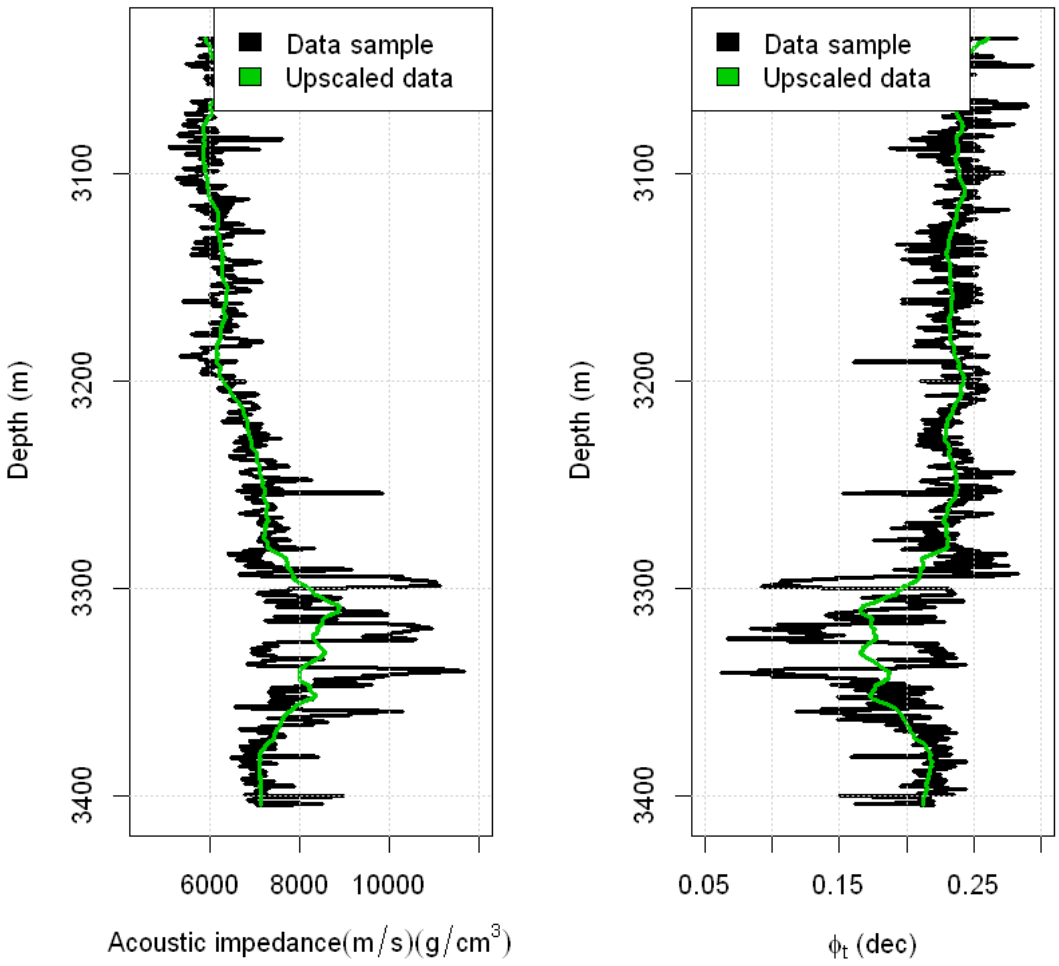
[3]:
```
par(mfrow = c(1,3))
plot(Ip,Depth,xlim=c(4500, 12000),xlab = expression(paste("Acoustic impedance",
                                                          (m/s)(g/
↪cm^3))),ylab="Depth",ylim=rev(range(Depth)),
    type = "l",lwd=2,col=("green3"))
```

```
plot(Ip_like,Depth,xlim=c(4500, 12000),xlab =
     expression(paste("Acoustic impedance", (m/s)(g/
 ↪cm^3))),ylab="Depth",ylim=rev(range(Depth)),
     type = "l",lwd=1,col=("green3"))
grid()

plot(Ip,Depth,xlim=c(4500, 12000),xlab = expression(paste("Acoustic impedance",␣
 ↪(m/s)(g/cm^3))),
     ylab="Depth",ylim=rev(range(Depth)),type = "l",lwd=1,col=("black"))
par(new=TRUE)
plot(Ip_like,Depth,xlim=c(4500, 12000),ylim=rev(range(Depth)),type = "l",lwd=3,
     col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
grid()
```



The data samples spatial distribution can be seen in the next figure (figure 1)

[11]:
```
#png("Results/wells_upscaled.png", bg = "white",  width=1500, height=2000,␣
↪res=200)
par(mfrow = c(1,2))

plot(Ip,Depth,xlim=c(4500, 12000),xlab = expression(paste("Acoustic impedance",␣
↪(m/s)(g/cm^3))),
     ylab="Depth (m)",ylim=rev(range(Depth)),type = "l",lwd=3,col=("black"))
par(new=TRUE)
plot(Ip_like,Depth,xlim=c(4500, 12000),ylim=rev(range(Depth)),type =␣
↪"l",lwd=3,col=("green3"),
     xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend = c("Data sample",
                                  "Upscaled data"), fill = c("black", "green3"))


plot(Phit,Depth,xlim=c(0.05, 0.3),xlab = expression(paste(phi[t], "␣
↪(dec)")),ylab="Depth (m)",
     ylim=rev(range(Depth)),type = "l",lwd=3,col=("black"))
par(new=TRUE)
plot(Phit_like,Depth,xlim=c(0.05, 0.3),ylim=rev(range(Depth)),type =␣
↪"l",lwd=3,col=("green3"),
     xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topleft", legend = c("Data sample",
                                  "Upscaled data"), fill = c("black", "green3"))
#dev.off()
```



The difference between the mean and the median in acoustic impedance (Ip) is 158.7265, this indicates positive skewness caused by the large number of outliers located to the right of the graph.

[8]:
```
png("Results/Ip_hist.png", bg = "white",  width=2000, height=2000, res=300)
HistBoxplot(Ip,nbins = ns, mean = Ip_Stat[5,2], median = Ip_Stat[4,2], main ="",
            xlab = expression(paste("A) Acoustic impedance ",(m/s)(g/cm^3))),␣
 ↪ylab = "Absolute Frequency (count)",
            AbsFreq = TRUE, PercentFreq = FALSE )
dev.off()
Ip_Stat
```

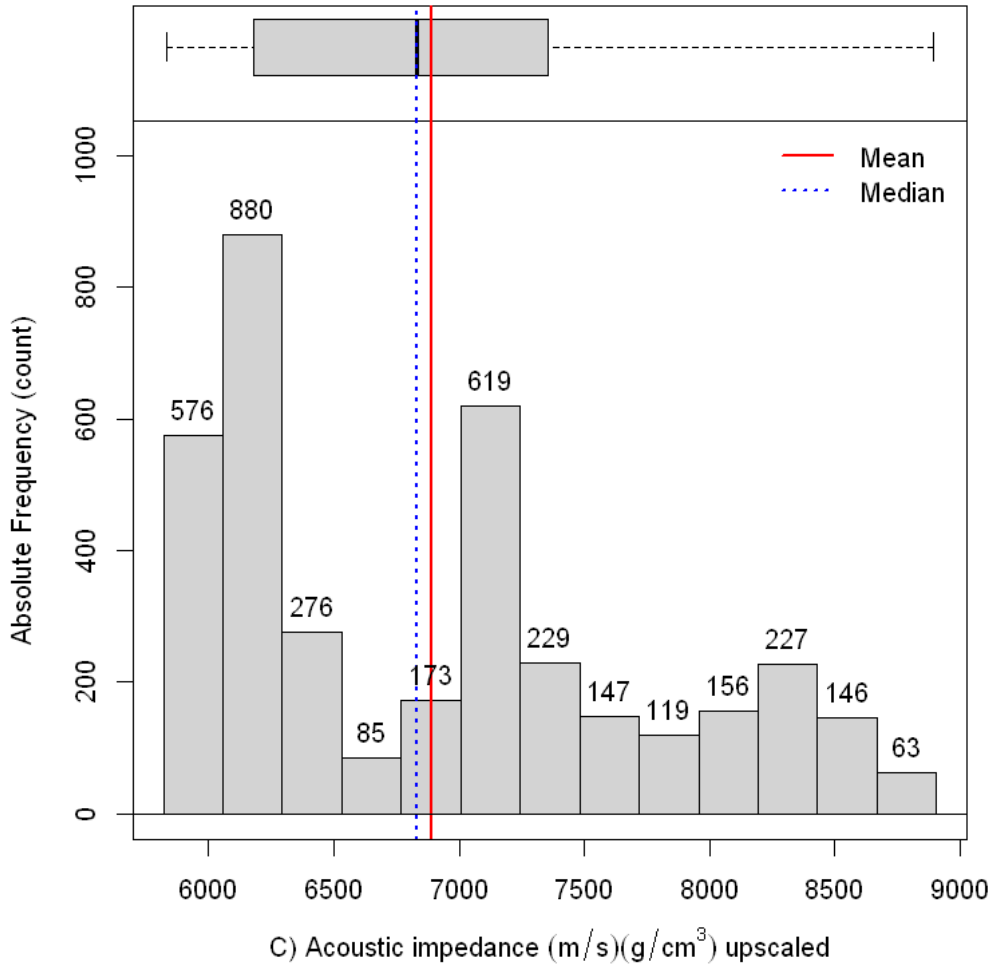
png: 2

	Statistics <chr>	Values <dbl>
muestras	n	3696.0000
minimos	Minimum	5086.0072
cuantiles1	1st. Quartile	6157.0052
medianas	Median	6809.5574
medias	Mean	6968.2839
cuantiles3	3rd. Quartile	7321.6170
maximos	Maximum	11661.4642
rangos	Rank	6575.4570
rangosInt	Interquartile Rank	1164.6118
varianzas	Variance	1310302.9359
desvs	Standard Deviation	1144.6846
CVs	Variation Coeff.	0.1643
simetrias	Skewness	1.5610
curtosiss	Kurtosis	5.7657

the same outliers behaviour in acoustic impedance upscaled histogram is observed. The histogram shows two bins with local maxima that could be a evidence of bimodal function.

[3]:
```
#png("Results/Ip_hist_upscaled.png", bg = "white",  width=2000, height=2000,␣
↪res=300)
HistBoxplot(Ip_like,nbins = ns, mean = Ip_like_Stat[5,2], median =␣
↪Ip_like_Stat[4,2], main ="",
            xlab = expression(paste("C) Acoustic impedance ",(m/s)(g/cm^3),"␣
↪upscaled")), ylab = "Absolute Frequency (count)",
            AbsFreq = TRUE, PercentFreq = FALSE )
#dev.off()
Ip_like_Stat
```

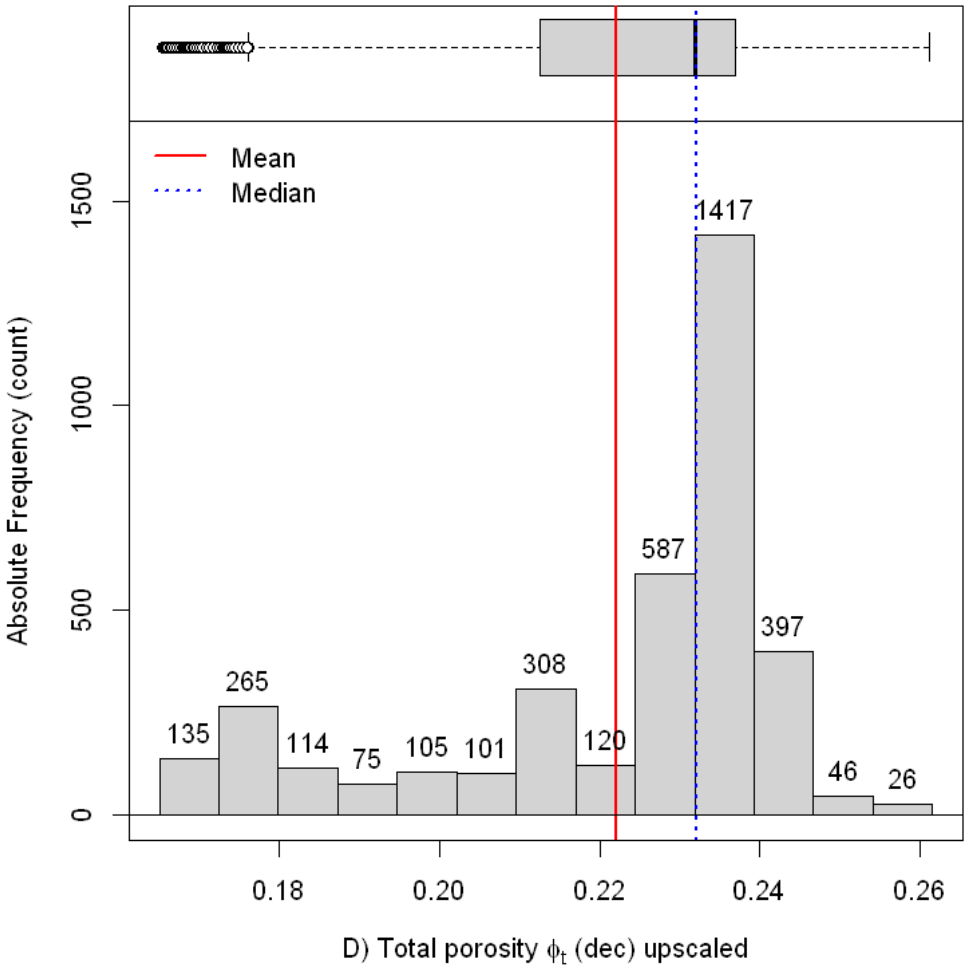
	Statistics <chr>	Values <dbl>
muestras	n	3696.0000
minimos	Minimum	5835.0969
cuantiles1	1st. Quartile	6178.1181
medianas	Median	6831.4653
medias	Mean	6892.8985
cuantiles3	3rd. Quartile	7353.4905
maximos	Maximum	8891.2124
rangos	Rank	3056.1155
rangosInt	Interquartile Rank	1175.3724
varianzas	Variance	708798.2641
desvs	Standard Deviation	841.9016
CVs	Variation Coeff.	0.1221
simetrias	Skewness	0.5967
curtosiss	Kurtosis	2.1898



For the case of total porosity log, the difference between the mean and median is -0.0075, which is very low. Regarding the boxplot, notice that most of the outliers are to the left of the graph.

[12]:
```
#png("Results/Phit_hist_upscaled.png", bg = "white",  width=2000, height=2000,␣
 ↪res=300)
HistBoxplot(Phit_like,nbins = ns, mean = Phit_like_Stat[5,2], median =␣
 ↪Phit_like_Stat[4,2], main ="",
            xlab = expression(paste("D) Total porosity ",phi[t], " (dec)␣
 ↪upscaled")),
            ylab = "Absolute Frequency (count)", AbsFreq = TRUE, PercentFreq =␣
 ↪FALSE )
#dev.off()
Phit_like_Stat
```

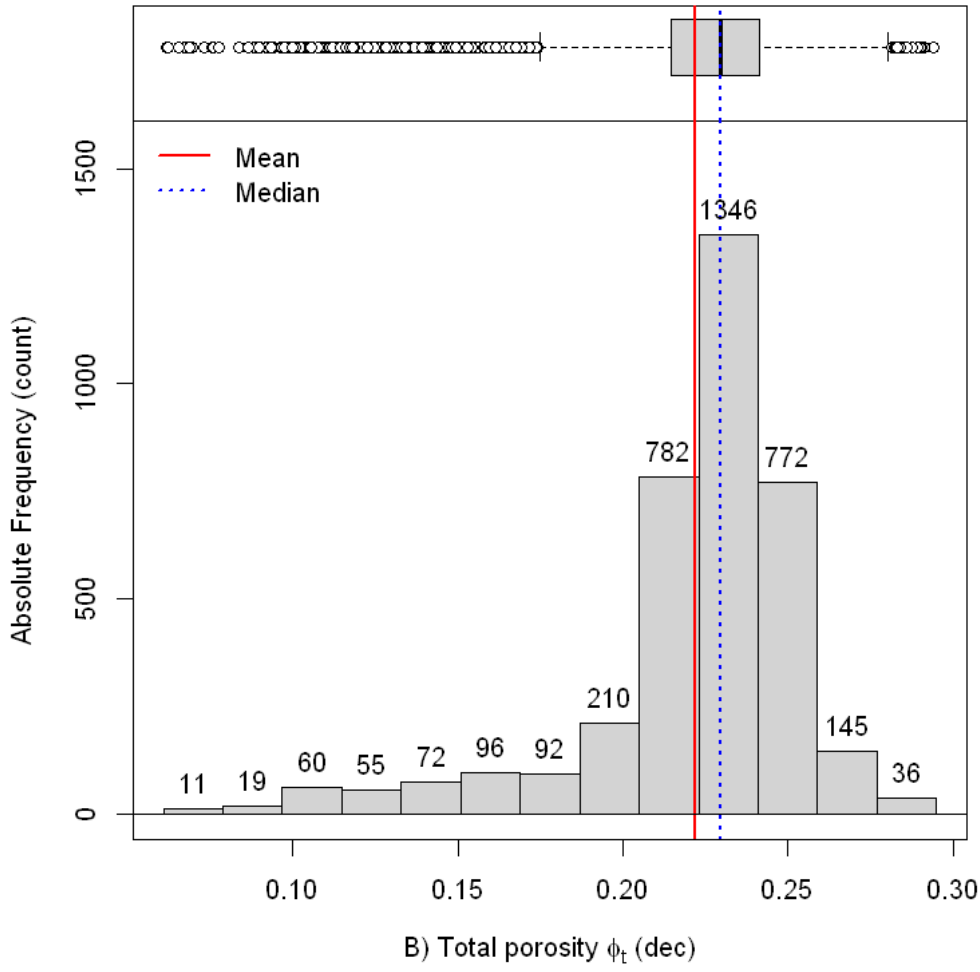
	Statistics <chr>	Values <dbl>
muestras	n	3696.0000
minimos	Minimum	0.1655
cuantiles1	1st. Quartile	0.2126
medianas	Median	0.2320
medias	Mean	0.2221
cuantiles3	3rd. Quartile	0.2369
maximos	Maximum	0.2611
rangos	Rank	0.0956
rangosInt	Interquartile Rank	0.0243
varianzas	Variance	0.0005
desvs	Standard Deviation	0.0223
CVs	Variation Coeff.	0.1003
simetrias	Skewness	-1.2139
curtosiss	Kurtosis	3.2414



The total porosity upscaled histogram shows outliers to the left of the graph. The histogram do not shows the bimodal effect observed in the acoustic impedance upscaled histogram.

[13]:
```
#png("Results/Phit_hist.png", bg = "white",  width=2000, height=2000, res=300)
HistBoxplot(Phit,nbins = ns, mean = Phit_Stat[5,2], median = Phit_Stat[4,2],␣
↪main ="",
            xlab = expression(paste("B) Total porosity ",phi[t], " (dec)")),
            ylab = "Absolute Frequency (count)", AbsFreq = TRUE, PercentFreq =␣
↪FALSE )
#dev.off()
Phit_Stat
```

	Statistics <chr>	Values <dbl>
muestras	n	3696.0000
minimos	Minimum	0.0620
cuantiles1	1st. Quartile	0.2147
medianas	Median	0.2295
medias	Mean	0.2220
cuantiles3	3rd. Quartile	0.2414
maximos	Maximum	0.2939
rangos	Rank	0.2319
rangosInt	Interquartile Rank	0.0267
varianzas	Variance	0.0011
desvs	Standard Deviation	0.0338
CVs	Variation Coeff.	0.1521
simetrias	Skewness	-1.8102
curtosiss	Kurtosis	6.9970



The comparative between dependence measures estimated in the scatter plots in the figures 2 and 3 shows low diferences, no more than 0.03. Apparently the structure of dependence is not different despite the differences in minimum and maximum observed in the univariate analysis.

[30]:
```
#png("Results/scatterplot.png", bg = "white",  width=2000, height=2000, res=300)
ScatterPlot(Ip , Phit,ns,
            Xmin = Ip_Stat[2,2], Xmax = Ip_Stat[7,2],
            Ymin = Phit_Stat[2,2],Ymax = Phit_Stat[7,2],
            XLAB = expression(paste(" Ip ",(m/s)(g/cm^3))),
            YLAB = expression(paste(" Total porosity ",phi[t], " (dec)")))
#dev.off()
```

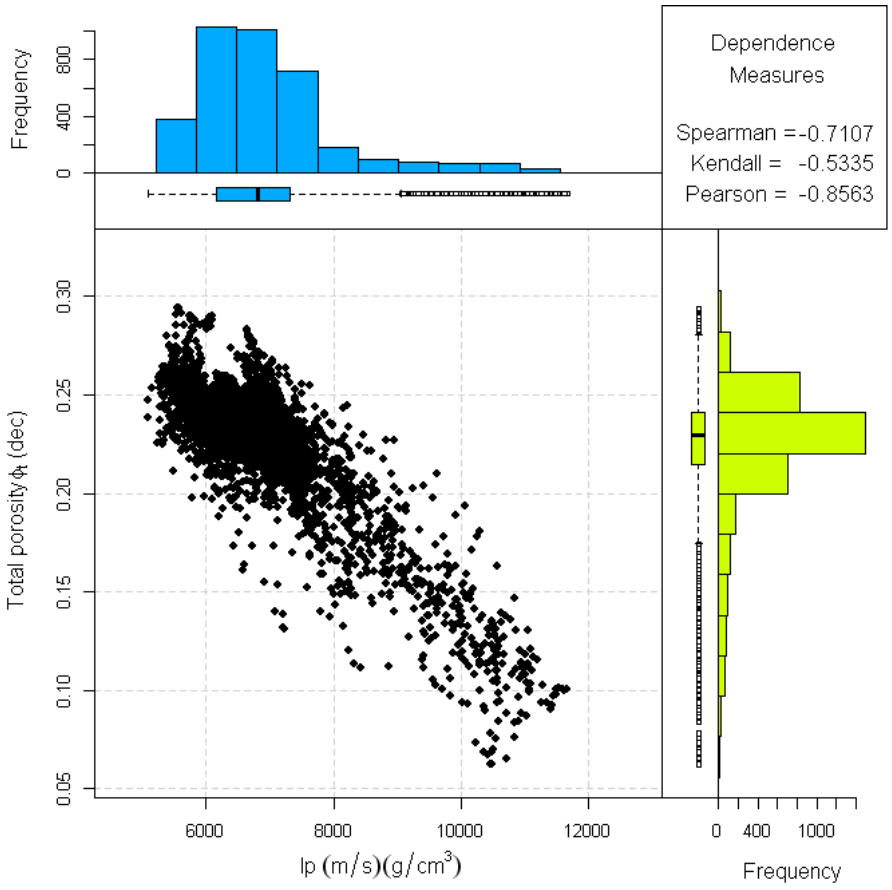


Figure 3

[12]:
```
png("Results/scatterplot_upscaled.png", bg = "white",  width=2000, height=2000,␣
 ↪res=300)
ScatterPlot(Ip_like , Phit_like, 9,
            Xmin = Ip_like_Stat[2,2], Xmax = Ip_like_Stat[7,2],
            Ymin = Phit_like_Stat[2,2],Ymax = Phit_like_Stat[7,2],
            XLAB = expression(paste(" Ip Backus upscaling ",(m/s)(g/cm^3))),
            YLAB = expression(paste(" Total porosity ",phi[t], " (dec) upscaled␣
 ↪moving averaging")))
dev.off()
```

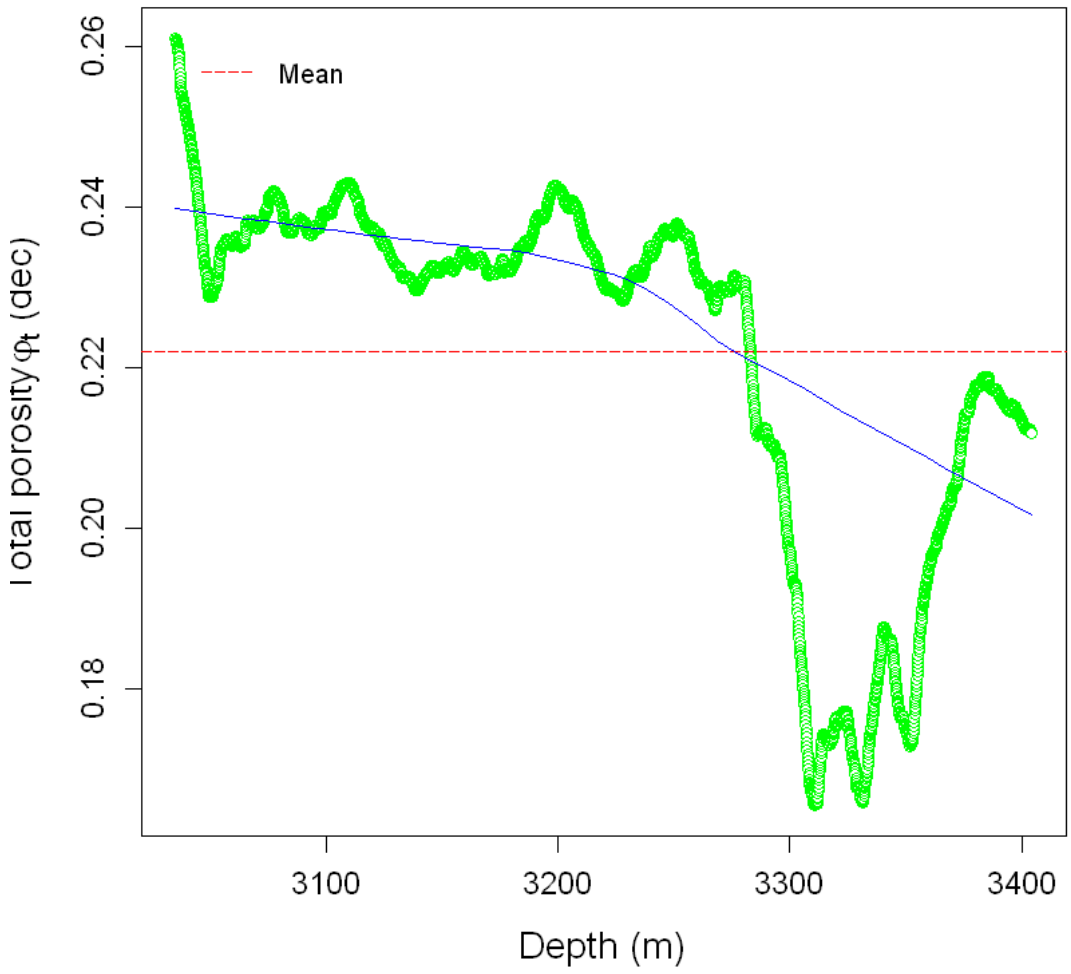
png: 2

6.2 Upscaled variogram estimation

To estimate the total porosity variogram, the upscaled total porosity and depth are taken. First plot the median regression and empirical variogram to evaluate if the variable has trend.

The median regresion plot (blue line) shows strong evidence of trend.

[7]:
```
MedianReg(Depth, Phit_like, Xlab= "Depth (m)", Varlab= expression(paste(" Total␣
↪porosity ",varphi[t], " (dec)")),
          deltay=500)
```



[8]:
```
N_lags<-1500
lag_value <- 0.1
YCoord=numeric(Depth_Stat[1,2])
```

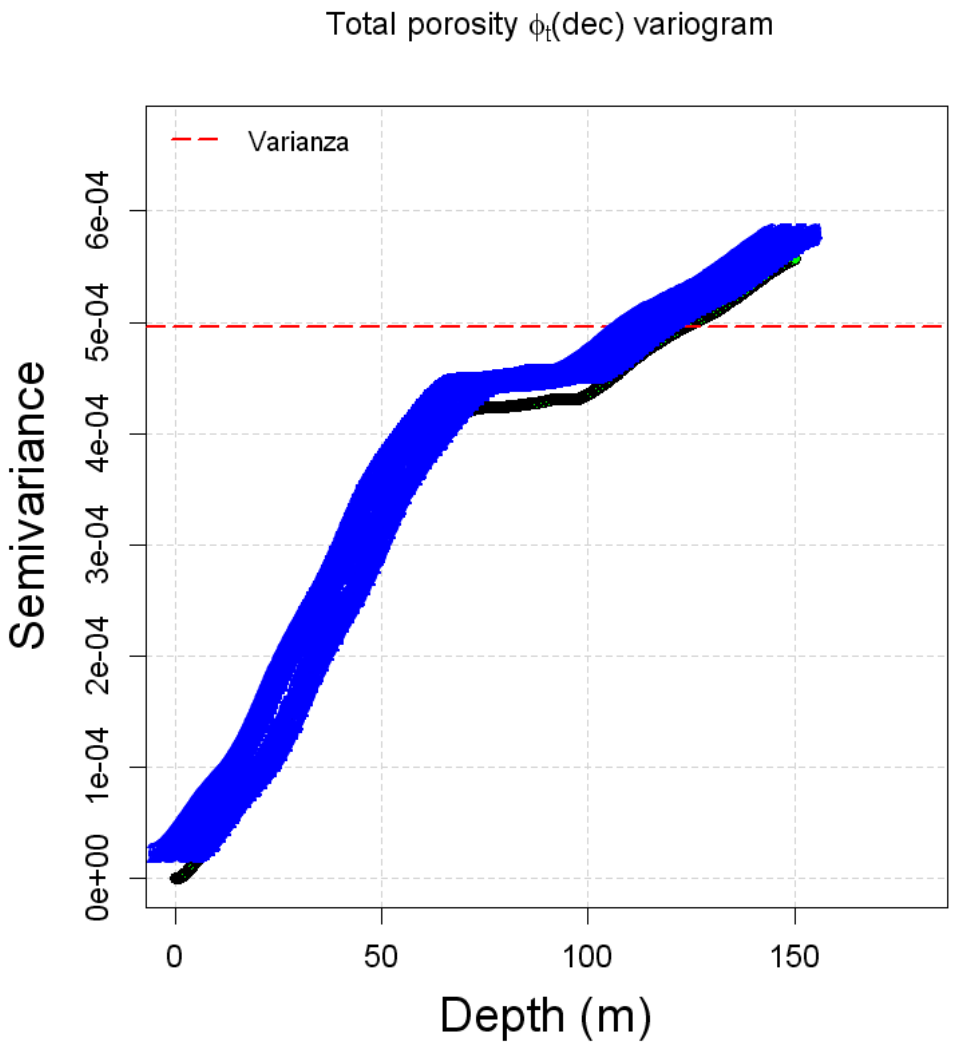
```
YCoord_Stat<-Estadisticas(YCoord)
```

```
Warning message:
"encountered a tie, and the difference between minimal and
                    maximal value is > length('x') * 'tie.limit'
the distribution could be multimodal"
```

The empirical variogram onfirm the evidence of trend, then, the variable is processed using polinomial function.

[10]:
```
Phit_like_VarioEstimation<-Variograma(CoorX=Depth, CoorY=YCoord,␣
↪Variable=Phit_like,
                                Direccion=90, Tol=0, NIntervalos=N_lags,
                                Lags=lag_value,
                                MainTitle=expression(paste(" Total porosity␣
↪",phi[t], "(dec) variogram"))
                                  , xlab="Depth (m)",
                                ylab = "Semivariance")
```

```
variog: computing variogram for direction = 90 degrees (1.571 radians)
        tolerance angle = 0 degrees (0 radians)
```



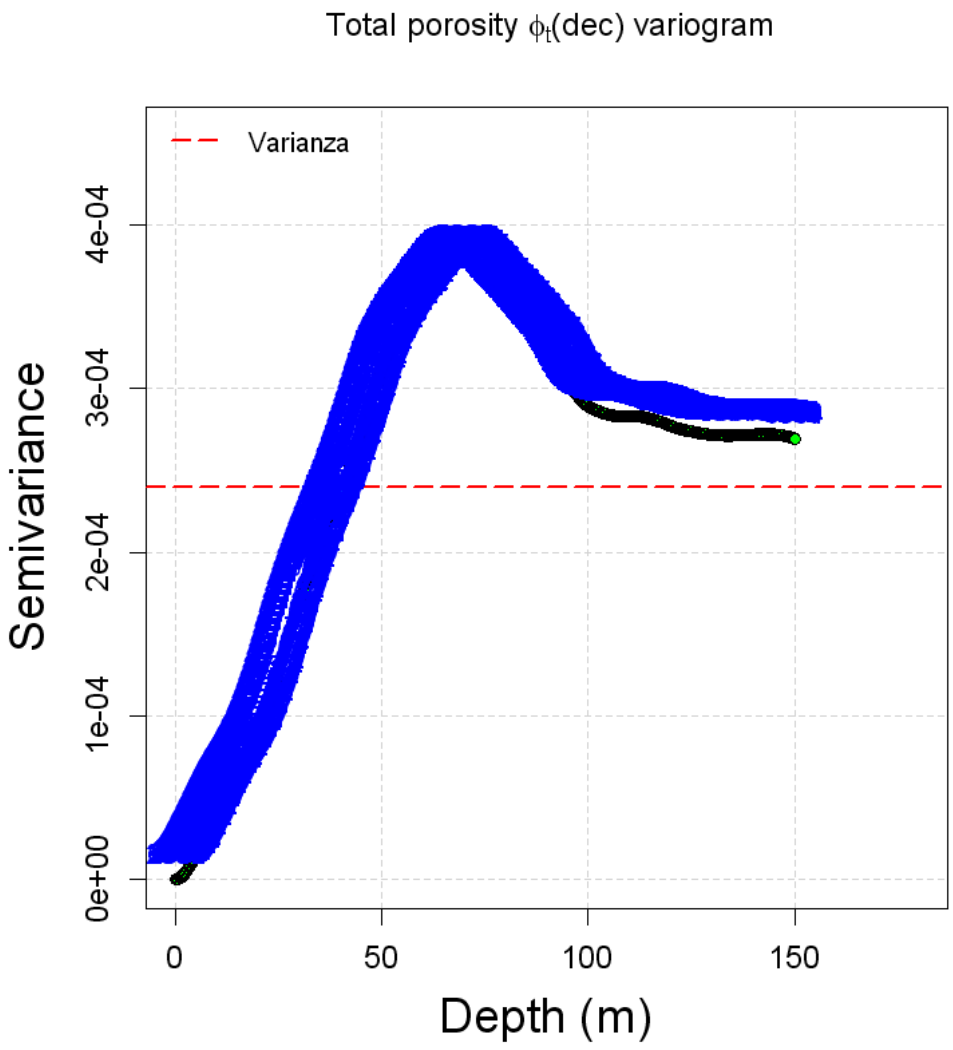
[11]:
```
var_tr1<-Trend(Depth, YCoord, Phit_like, degree=1)
```

The transform can reduce the trend, whith this semivariogram a variogram model is fitted.

[12]:
```
Phit_like_VarioEstimation_trend<-Variograma(CoorX=Depth, CoorY=YCoord,␣
↪Variable=var_tr1[,3],
                                Direccion=90, Tol=0, NIntervalos=N_lags,
                                Lags=lag_value,
                                MainTitle=expression(paste(" Total porosity␣
↪",phi[t], "(dec) variogram"))
                                  , xlab="Depth (m)",
                                ylab = "Semivariance")
```

```
variog: computing variogram for direction = 90 degrees (1.571 radians)
```

```
tolerance angle = 0 degrees (0 radians)
```

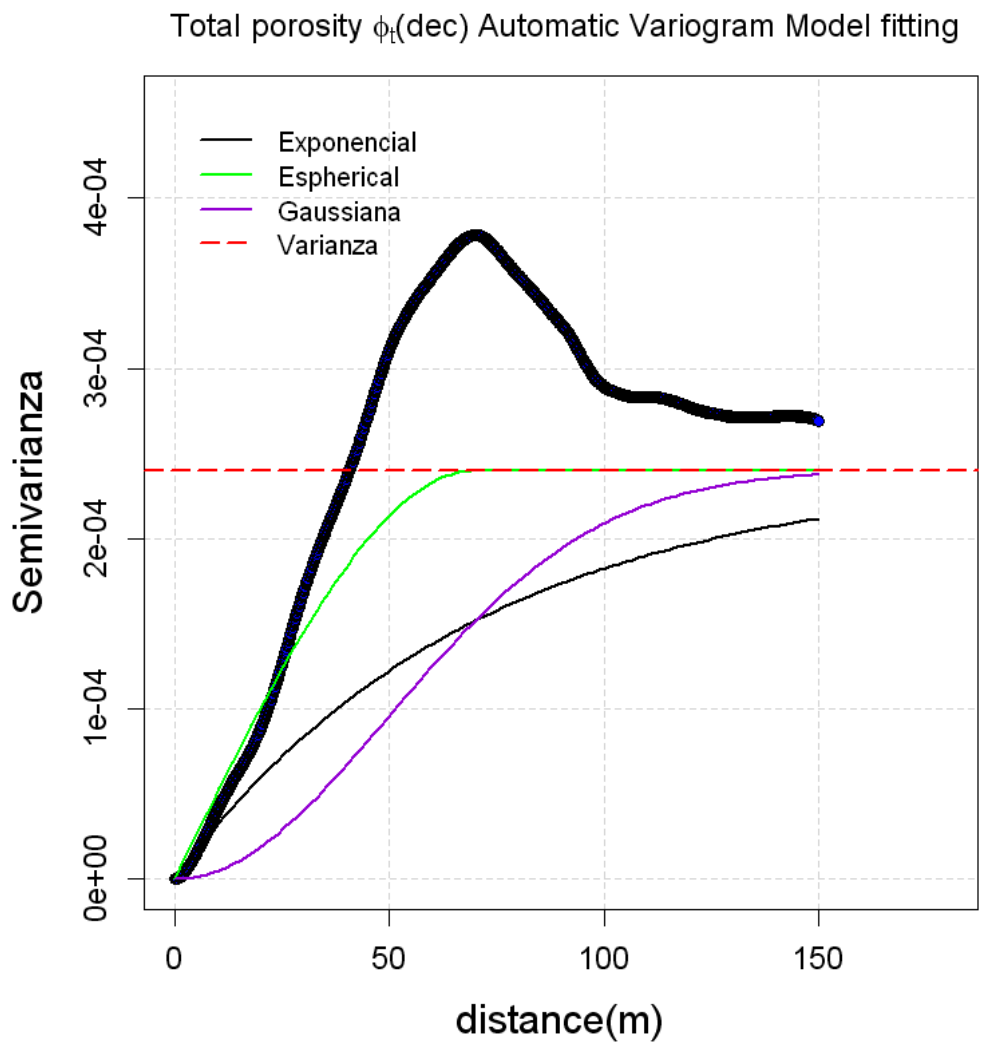


The empirical variogram is fitted using three models: exponential, spherical and Gaussian

[14]:
```
Phit_like_AllModelVarioFit<-AllModel(Depth, YCoord,
                                     var_tr1[,3], 0, 90, N_lags, lag_value, 1,
                                     expression(paste(" Total porosity ",phi[t],
                                                      "(dec) Automatic Variogram␣
↪Model fitting")),xlab="distance(m)")
```

```
variog: computing omnidirectional variogram
variofit: covariance model used is exponential
variofit: weights used: npairs
variofit: minimisation function used: optim
```

```
variofit: covariance model used is spherical
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is gaussian
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is exponential
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is spherical
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is gaussian
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is exponential
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is spherical
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is gaussian
variofit: weights used: npairs
variofit: minimisation function used: optim
```



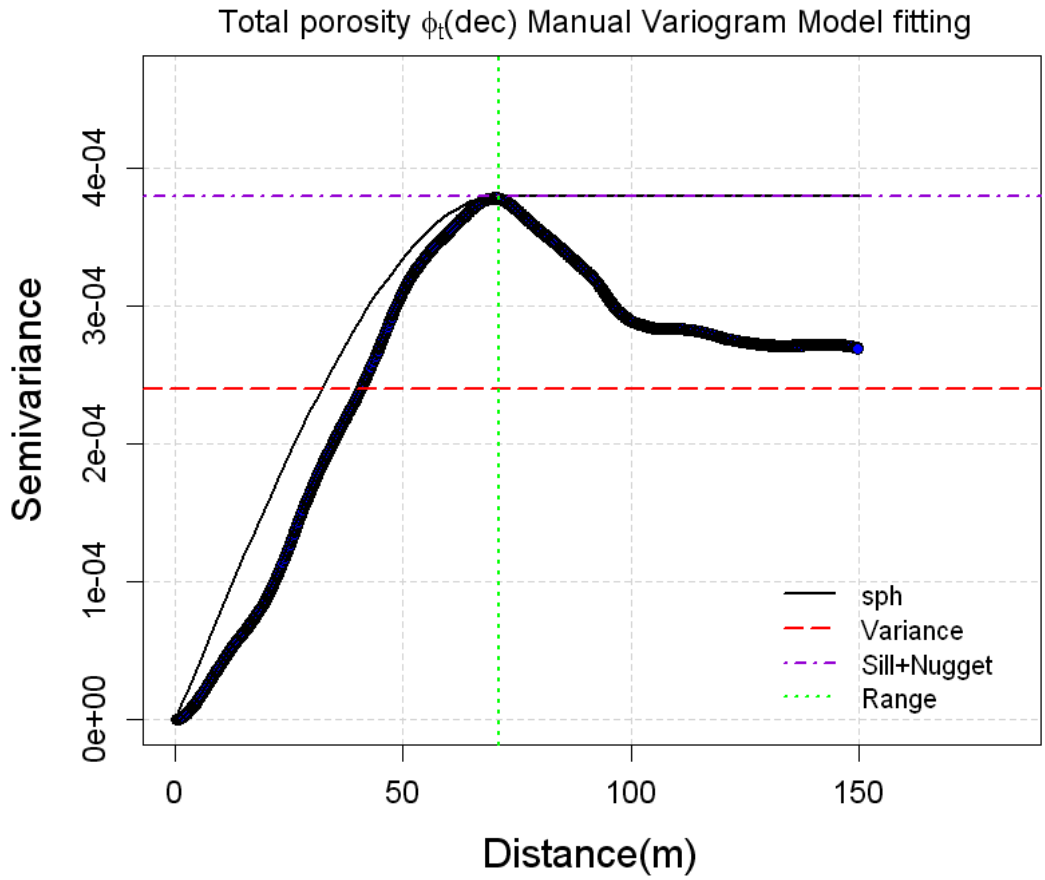
The automatic fitting shows the spherical model is the best option. Therefore, the spherical model is chosen. the ojective is obtain realizations with similar properties in both scales well log and upscaled log.

[15]:
```
vario_model<- 2
nugget<- 0
sill_with_nugget<- 0.00038
range <- 71
```

[18]:
```
#png("Results/Phit_variog_subsa.png", bg = "white",  width=2000, height=2000,␣
↪res=300)
Phit_like_EyeModelVarioFit<-EyeModel(Depth, YCoord,
                                     var_tr1[,3], 0, 90, N_lags, lag_value, 1,
```

```
                                    vario_model, nugget, sill_with_nugget, range,
                                    expression(paste(" Total porosity ",phi[t],
                                                     "(dec) Manual Variogram Model␣
 ↪fitting")),
                                  xlab="Distance(m)")
#dev.off()
```

```
variog: computing omnidirectional variogram
```



Model	Nugget	Sill+Nugget	Range	MSE
sph	0.0000	0.0004	71.0000	0.0000

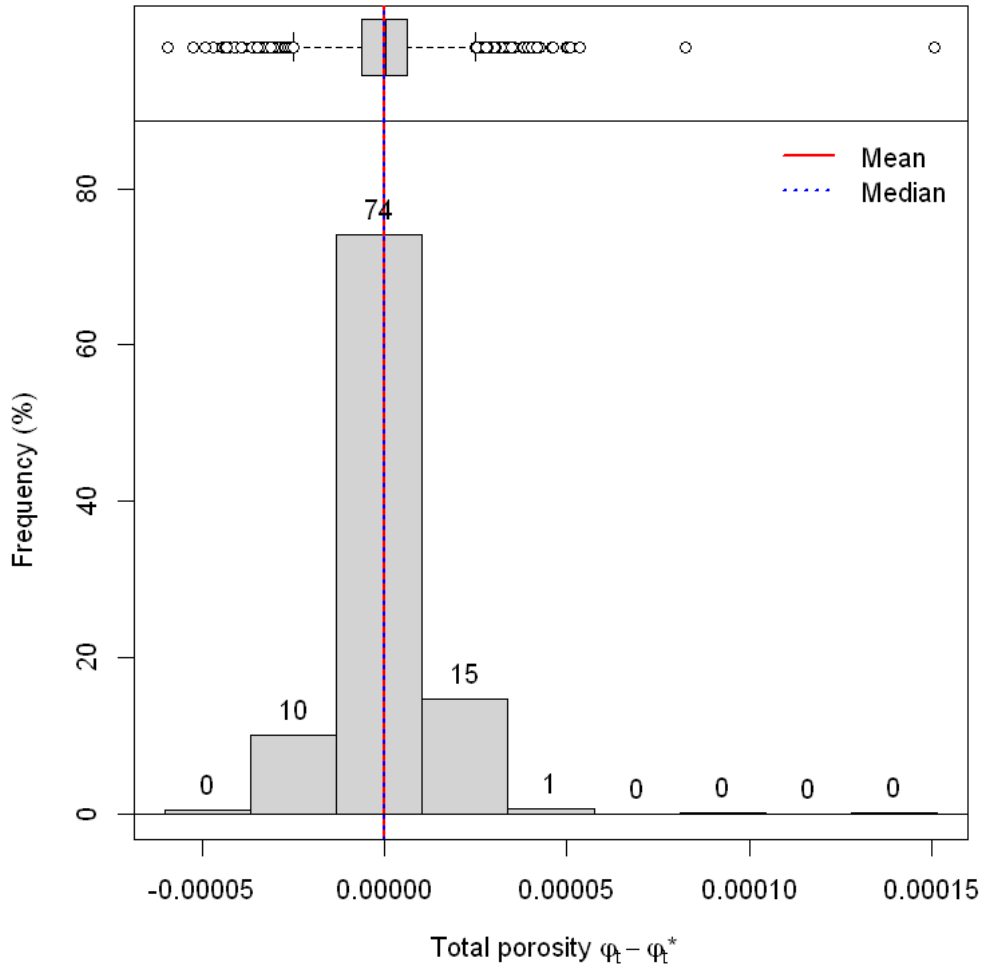
The variogram parameters for spherical model are nugget 0, sill plus nugget 0.00038 and range 71 meters.

[17]:
```
Phit_like_CrossValid<- CrossValidation(Depth, YCoord,
                                       var_tr1[,3], vario_model, nugget,␣
↪sill_with_nugget,
                                          range,MaxAnis=0, proporcion=1)
```

[19]:
```
Phit_like_CrossValid_Stat<- Val_Estadisticos(Phit_like_CrossValid[1:
↪length(Phit_like),c(3,4,5)])
Phit_like_CrossValid_Stat
```

	Z <dbl>	Z* <dbl>	Z-Z* <dbl>
No_muestras	3696.00000	3696.00000	3696.00000
Minimo	-0.04297	-0.04295	-0.00006
Cuartil_1er	-0.00558	-0.00557	-0.00001
Mediana	0.00114	0.00114	0.00000
Media	0.00000	0.00000	0.00000
Cuartil_3er	0.01279	0.01279	0.00001
Maximo	0.02156	0.02152	0.00015
Rango	0.06454	0.06447	0.00021
Rango_Intercuartil	0.01837	0.01836	0.00001
Varianza	0.00024	0.00024	0.00000
Desv_Estandar	0.01550	0.01549	0.00001
Simetria	-0.84339	-0.84332	0.56559
Curtosis	3.12535	3.12496	11.70648

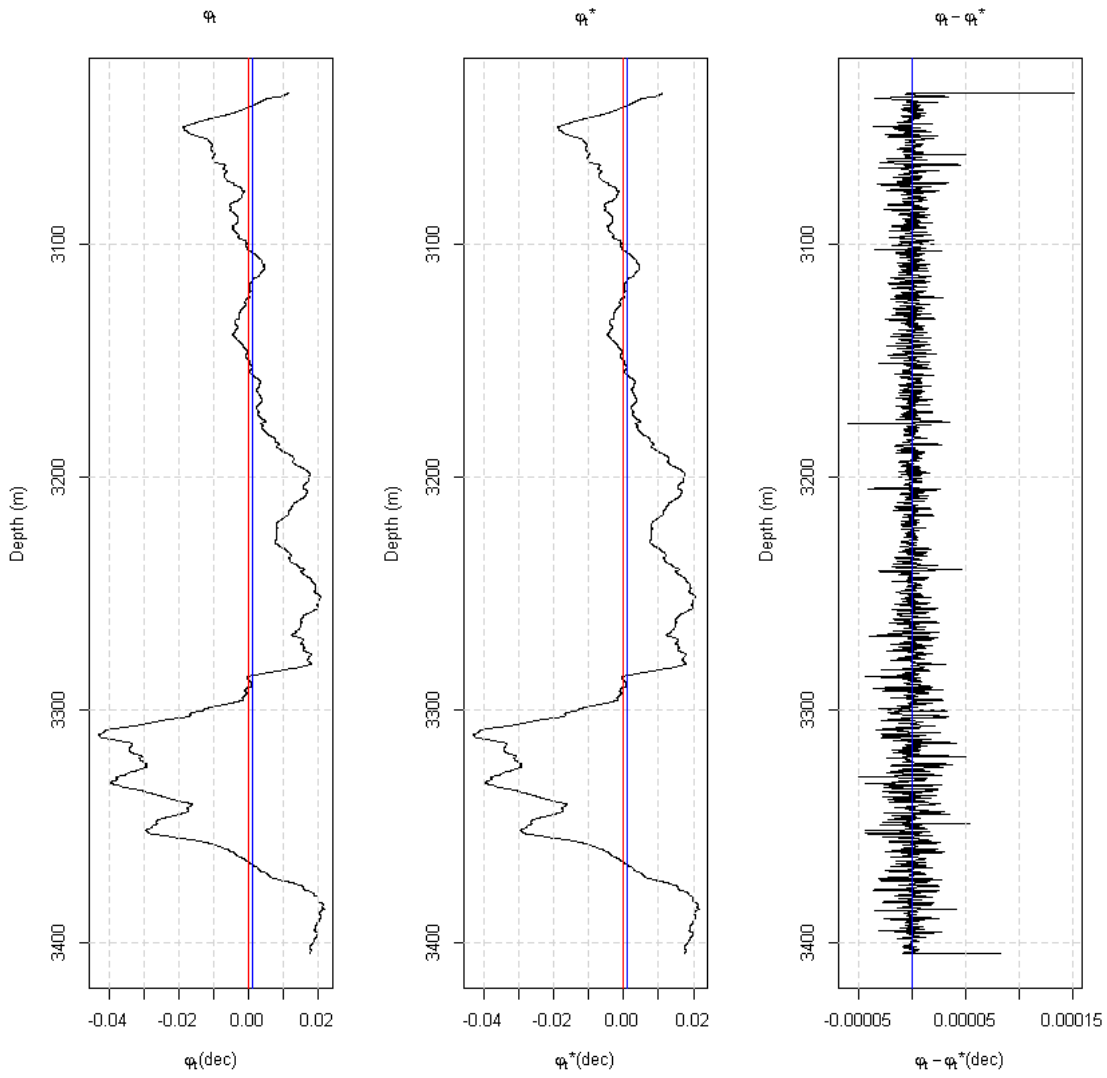
[20]:
```
HistBoxplot(x=Phit_like_CrossValid[,5], mean = Phit_like_CrossValid_Stat[5,3],
            median = Phit_like_CrossValid_Stat[4,3], main ="",
            xlab = expression(paste(" Total porosity␣
↪",varphi[t]-varphi[t],"*")), ylab = "Frequency (%)",
            AbsFreq = FALSE, PercentFreq = TRUE )
```



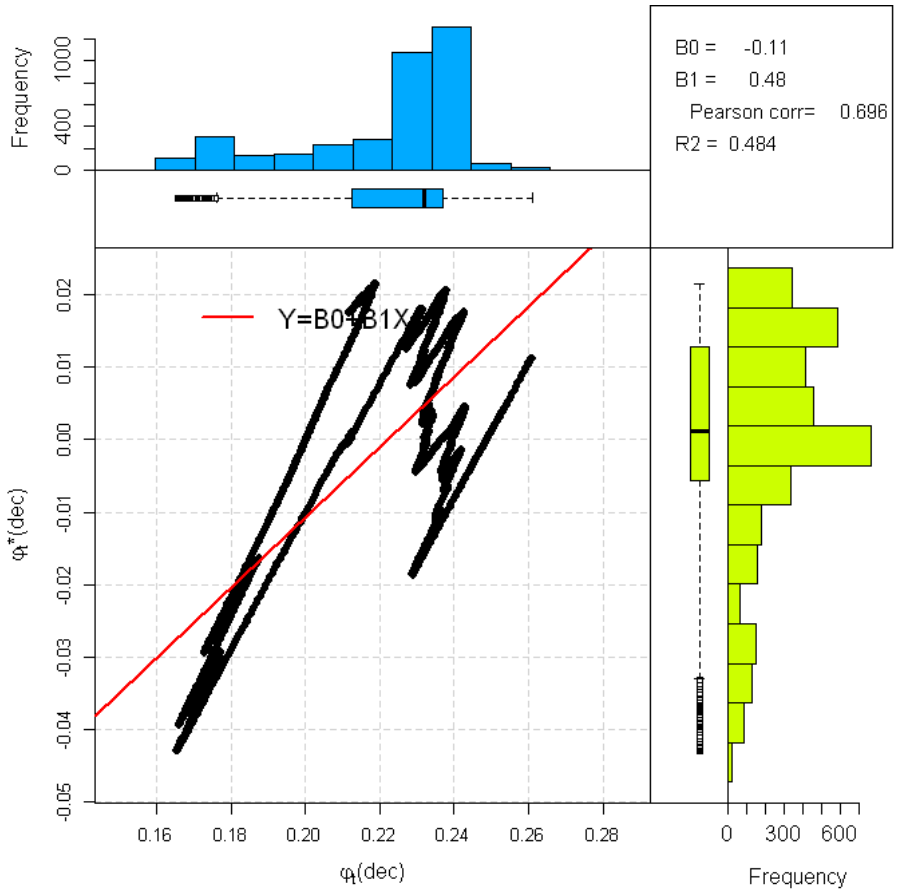
[21]:
```
Phit_like_est<-Phit_like_CrossValid[,4]
Phit_like_est_Stat<-Estadisticas(Phit_like_est)
Phit_like_diff<-Phit_like_CrossValid[,5]
Phit_like_diff_Stat<-Estadisticas(Phit_like_diff)
```

[22]:
```
LogPlot(depth=Depth, n_logs=3,
        log_list=list(var_tr1[,3],Phit_like_est,Phit_like_diff),
       ␣
↪log_name_list=list(expression(paste(varphi[t])),expression(paste(varphi[t],"*")),expression(p
↪
       ␣
↪log_xlabel_list=list(expression(paste(varphi[t],"(dec)")),expression(paste(varphi[t],"*(dec)'
```

```
        plot_mean=TRUE, plot_median=TRUE)
```



[23]:
```
scaterplotReg(Phit_like , Phit_like_est, ns,
              Xmin = Phit_like_Stat[2,2], Xmax = Phit_like_Stat[7,2],
              Ymin = Phit_like_est_Stat[2,2],Ymax = Phit_like_est_Stat[7,2],
              XLAB = expression(paste(varphi[t],"(dec)")), YLAB =␣
↪expression(paste(varphi[t],"*(dec)")))
```



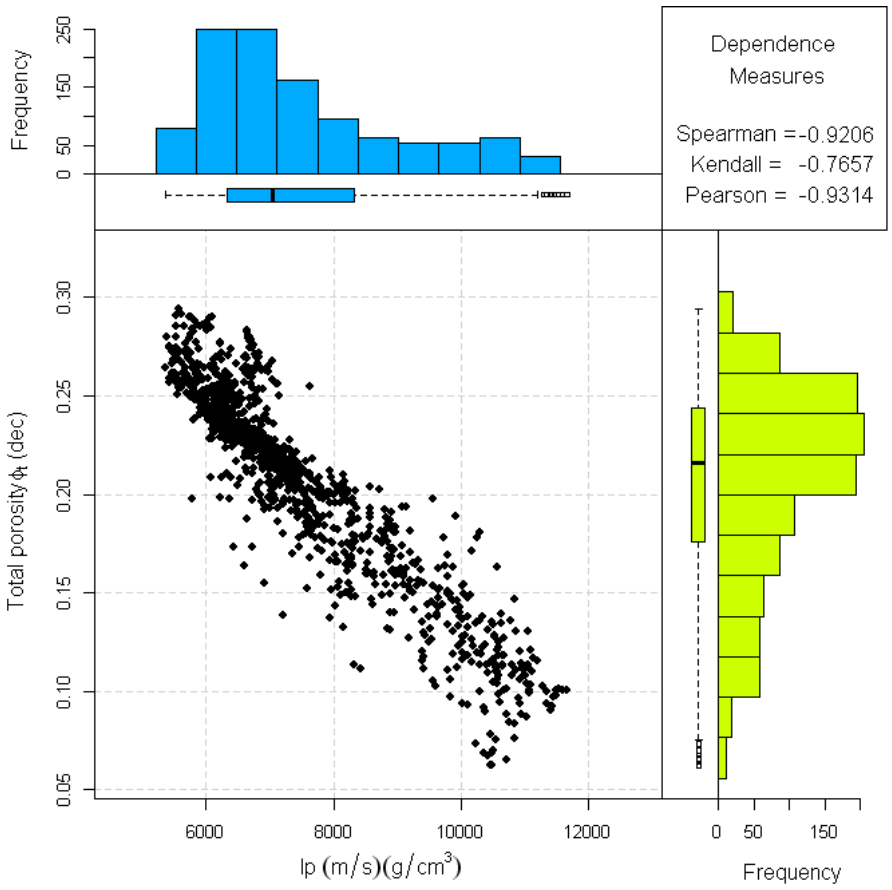
6.3 Prior information

The prior information comes from well logs. The joint distribution function is build using the best fit for both marginals and copula. First, depuring the data sample is neccesary to avoid repeat samples.

[26]:
```
muestra<-as.data.frame(cbind(Ip, Phit))
muestra_depurada <- depurar.xy(muestra)
Ip_d<-muestra_depurada[,1]
Phit_d<-muestra_depurada[,2]
```

[27]:
```
ScatterPlot(Ip_d , Phit_d,ns,
            Xmin = Ip_Stat[2,2], Xmax = Ip_Stat[7,2],
```

```
        Ymin = Phit_Stat[2,2],Ymax = Phit_Stat[7,2],
        XLAB = expression(paste(" Ip ",(m/s)(g/cm^3))),
        YLAB = expression(paste(" Total porosity ",phi[t], " (dec)")))
```

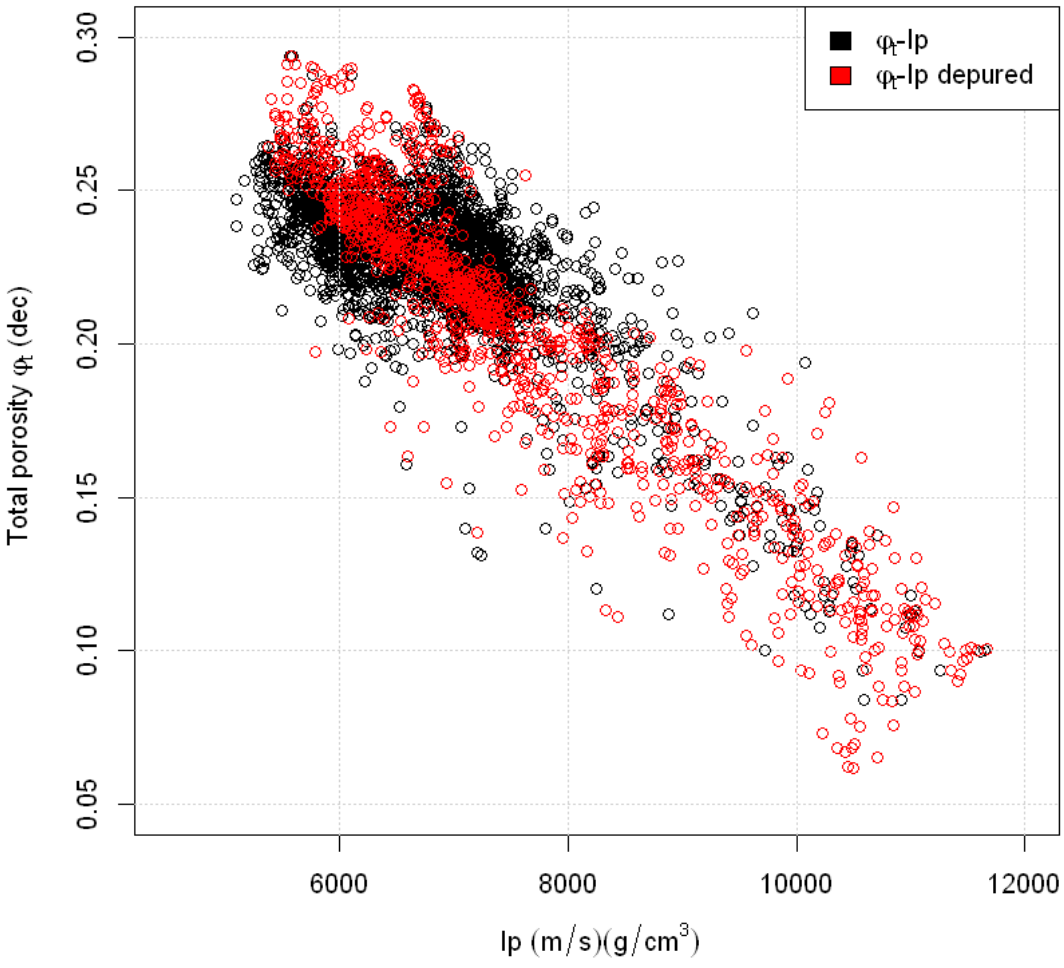


When the repeated values are removed the dependency is higher than original data samples, Pearson and Spearman dependency measures difference is low, but Kendall dependency measure has a considerable difference.

Dependence	Original	Depured
Pearson	-0.8563	-0.9206
Spearman	-0.7107	-0.7657
Kendall	-0.5335	-0.9314

When comparing the original and depured scatterplots, it can be seen that the repeated data were found in the range of total porosity (ϕ_t) 0.2-0.25 and acoustic impedance (Ip) 5500-8000.

[28]:
```
plot(muestra,xlab = expression(paste(" Ip ",(m/s)(g/cm^3))),
            ylab = expression(paste(" Total porosity ",varphi[t], "␣
 ↪(dec)")),xlim=c(4500, 12000),ylim=c(0.05, 0.3))
par(new=TRUE)
plot(muestra_depurada,xlim=c(4500, 12000),ylim=c(0.05, 0.
 ↪3),col=("red"),xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend =␣
 ↪c(expression(paste(varphi[t],"-Ip")),expression(paste(varphi[t],"-Ip␣
 ↪depured"))),
       fill = c("black", "red"))
```

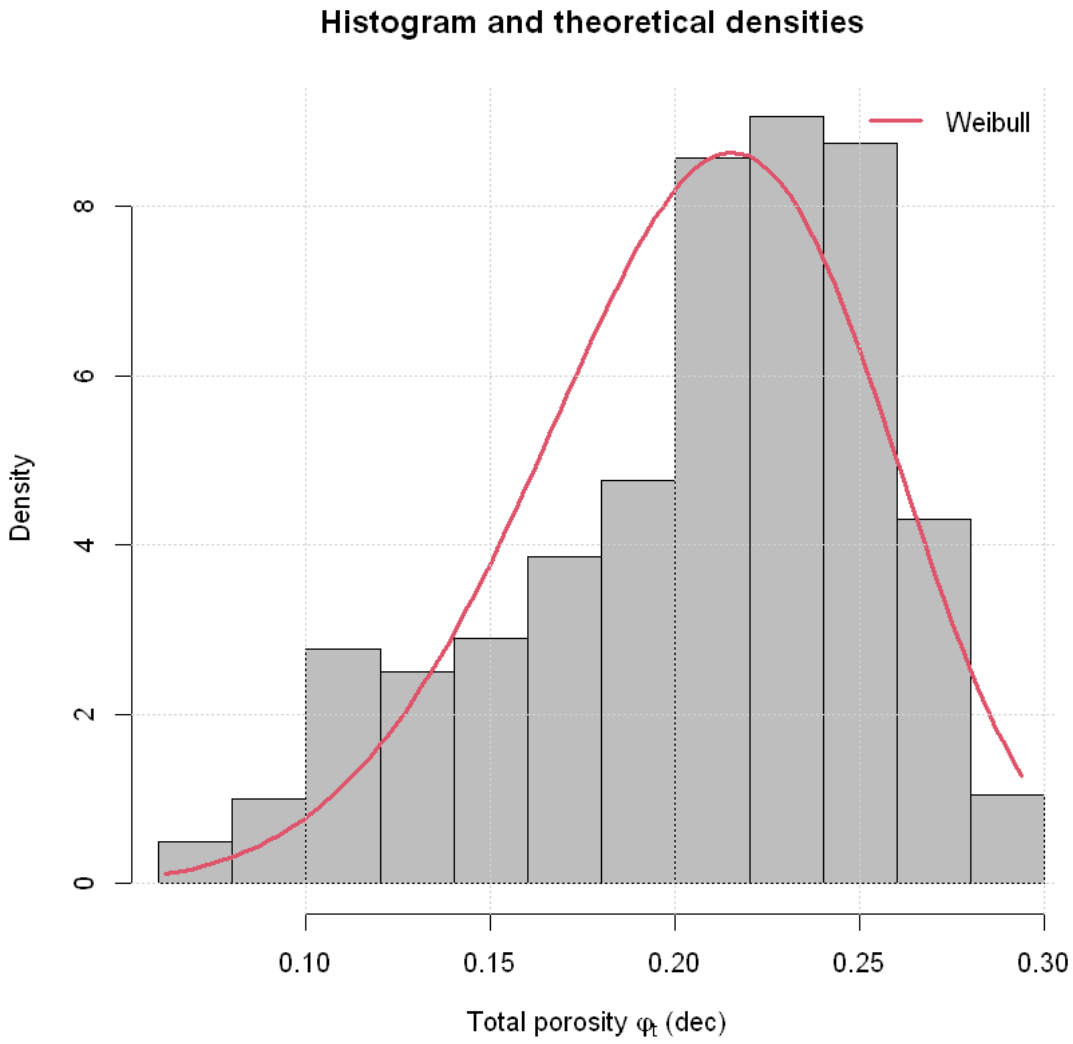


The total porosity (ϕ_t) prior estimation considers that Weibull function is the best fits, the parameters of the function are the following:

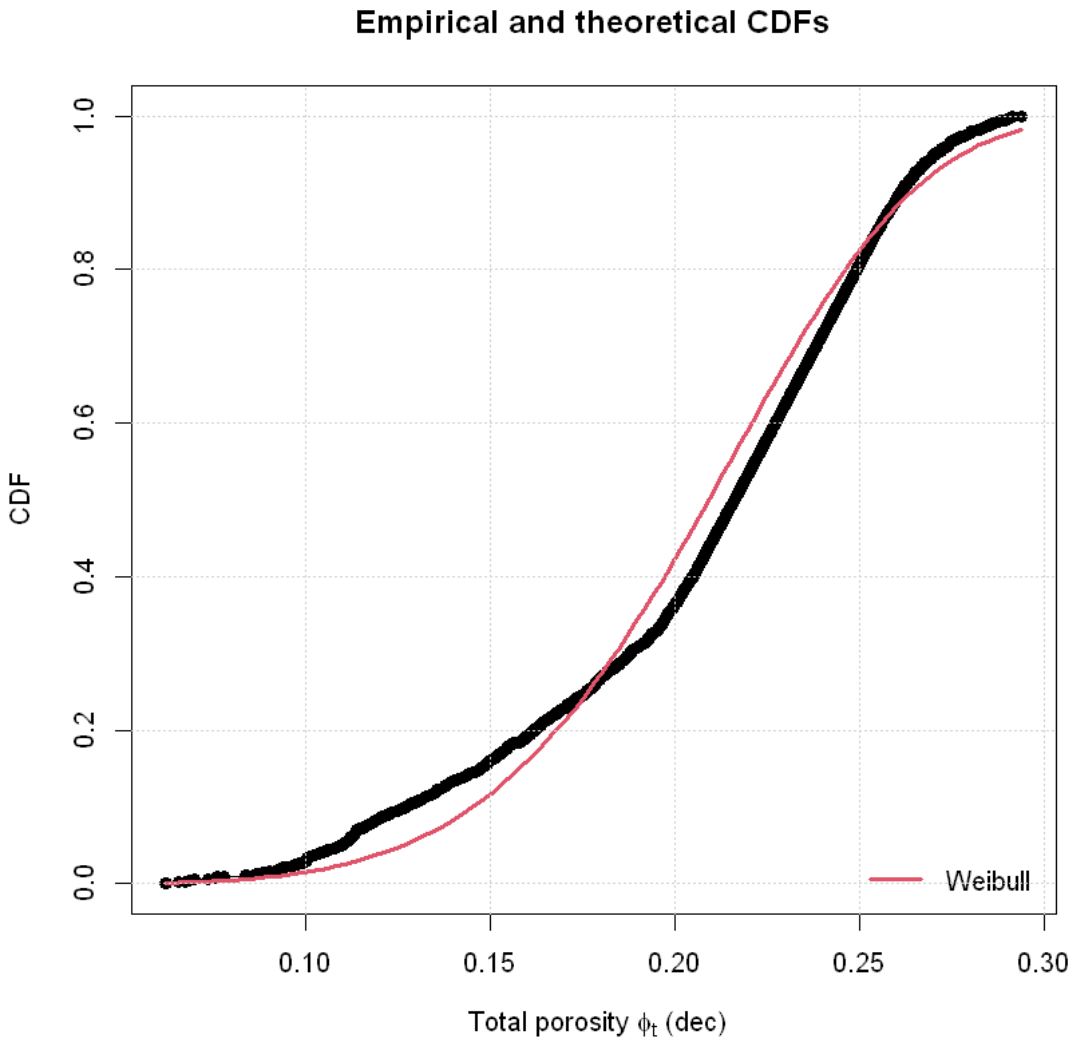
[29]:
```
fw <- fitdist(Phit_d, "weibull", method="mle")
summary(fw)
```

```
Fitting of the distribution ' weibull ' by maximum likelihood
Parameters :
       estimate  Std. Error
shape 5.1646292 0.129598464
scale 0.2244496 0.001367821
Loglikelihood:  1791.958   AIC:  -3579.917   BIC:  -3569.905
Correlation matrix:
          shape     scale
shape 1.0000000 0.2918398
scale 0.2918398 1.0000000
```

[30]:
```
denscomp(list(fw),xlab = expression(paste(" Total porosity ",varphi[t], "␣
↪(dec)")),
         datacol = "gray",fitlwd=3 , legendtext = "Weibull")
grid()
```



[24]:
```
cdfcomp(list(fw),xlab = expression(paste(" Total porosity ",phi[t], "␣
↪(dec)")),fitlwd=3,  legendtext = "Weibull")
grid()
```



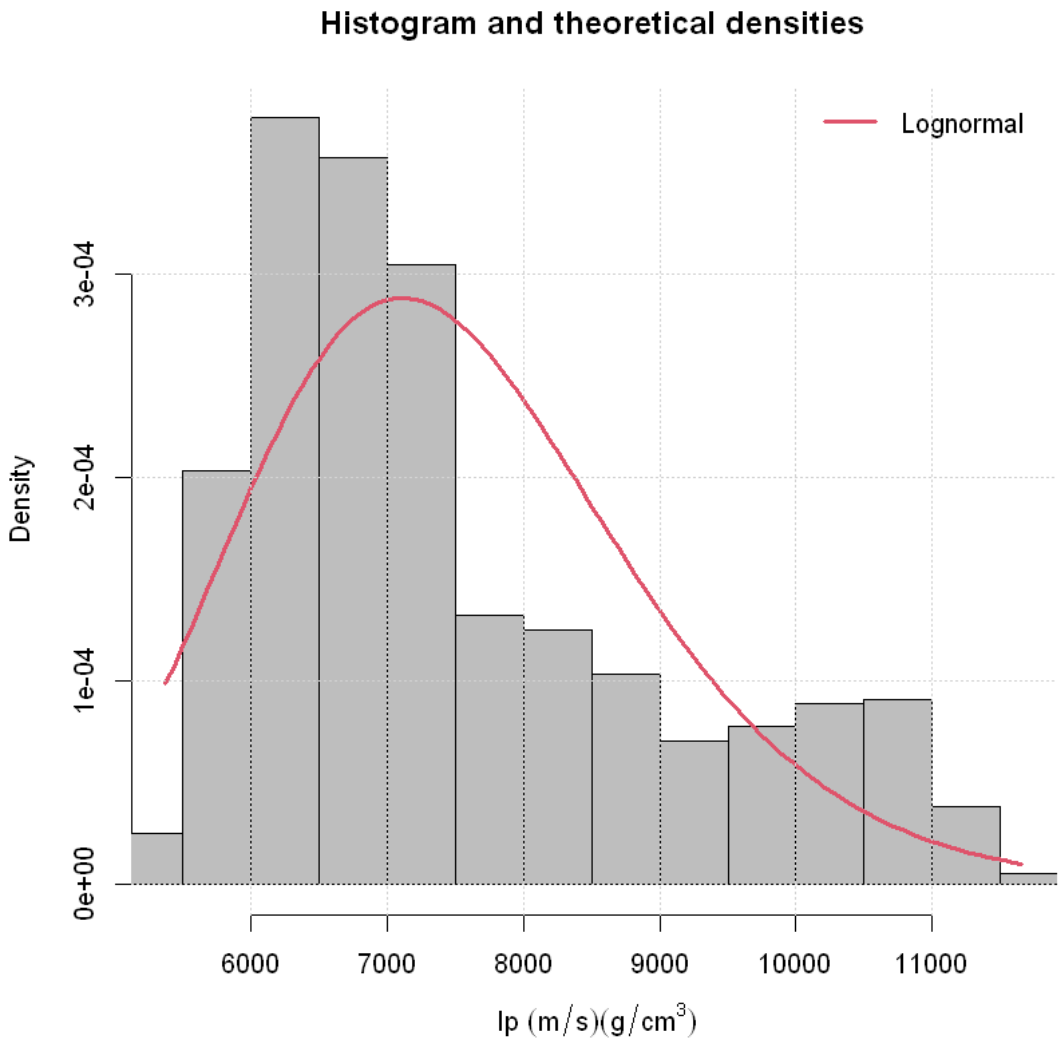
For the case of acoustic impedance, the lognormal function is the best option. The values of the parameters are follow

[32]:
```
fln1 <- fitdist(Ip_d, "lnorm", method="mle")
summary(fln1)
```

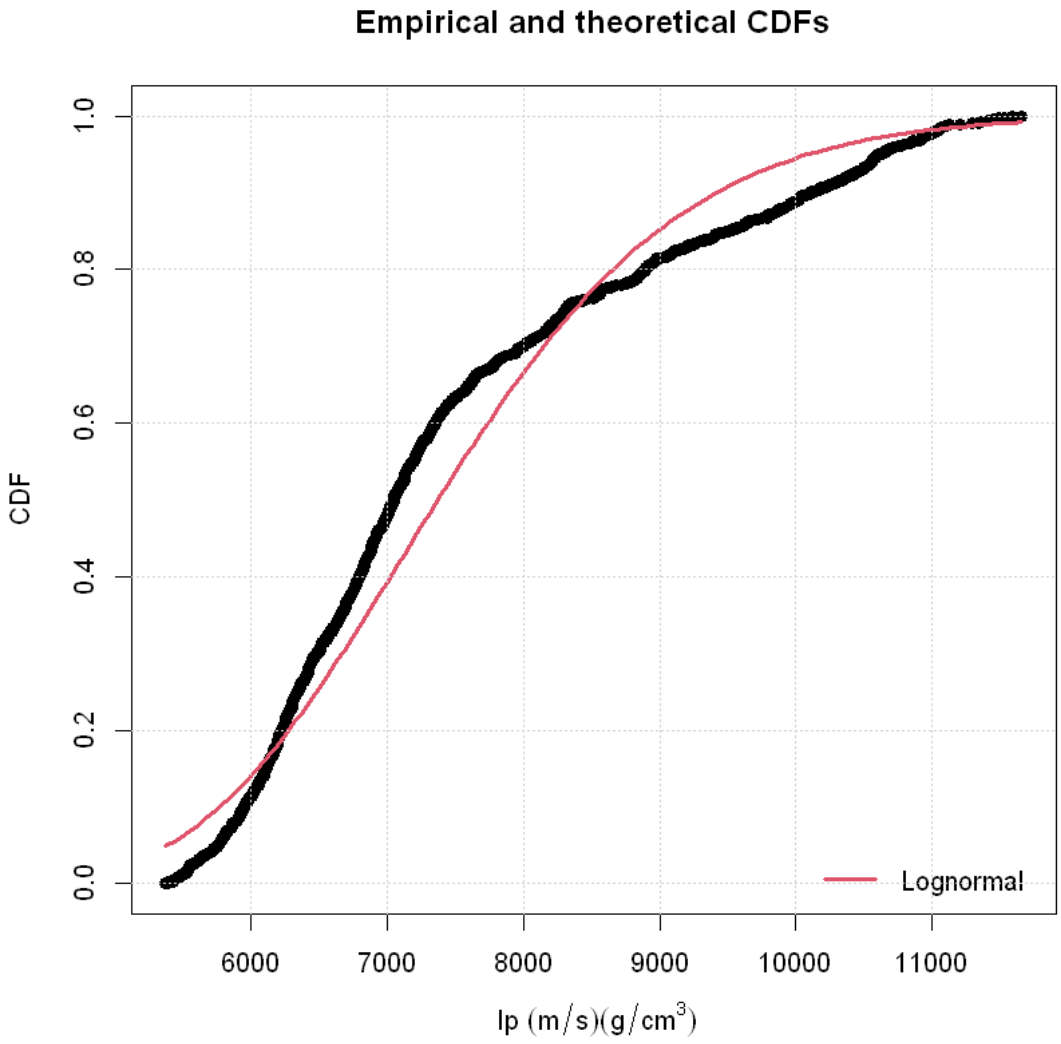
```
Fitting of the distribution ' lnorm ' by maximum likelihood
Parameters :
         estimate  Std. Error
meanlog 8.905543 0.005752746
sdlog   0.191057 0.004067304
Loglikelihood:  -9562.235   AIC:  19128.47   BIC:  19138.48
Correlation matrix:
         meanlog sdlog
```

```
meanlog        1      0
sdlog          0      1
```

[33]:
```
plot.legend <- c("Lognormal")
denscomp(list(fln1),datacol = "gray",xlab = expression(paste(" Ip ",(m/s)(g/
 ↪cm^3))),fitlwd=3 ,
         legendtext = plot.legend)
grid()
```



[34]:
```
cdfcomp(list(fln1),xlab = expression(paste(" Ip ",(m/s)(g/cm^3))),fitlwd=3, ␣
 ↪legendtext = plot.legend)
grid()
```



The parametric copula is

[35]:
```
data_pseudo=pobs(muestra_depurada)
n=1103
```

[36]:
```
ifme_frank <- fitCopula(frankCopula(), data = data_pseudo, method = "ml")
summary(ifme_frank)
```

```
Call: fitCopula(frankCopula(), data = data_pseudo, ... = pairlist(method = "ml"))
Fit based on "maximum likelihood" and 1103 2-dimensional observations.
Frank copula, dim. d = 2
      Estimate Std. Error
alpha   -15.04      0.443
The maximized loglikelihood is 1004
```

```
Optimization converged
Number of loglikelihood evaluations:
function gradient
       5        5
```

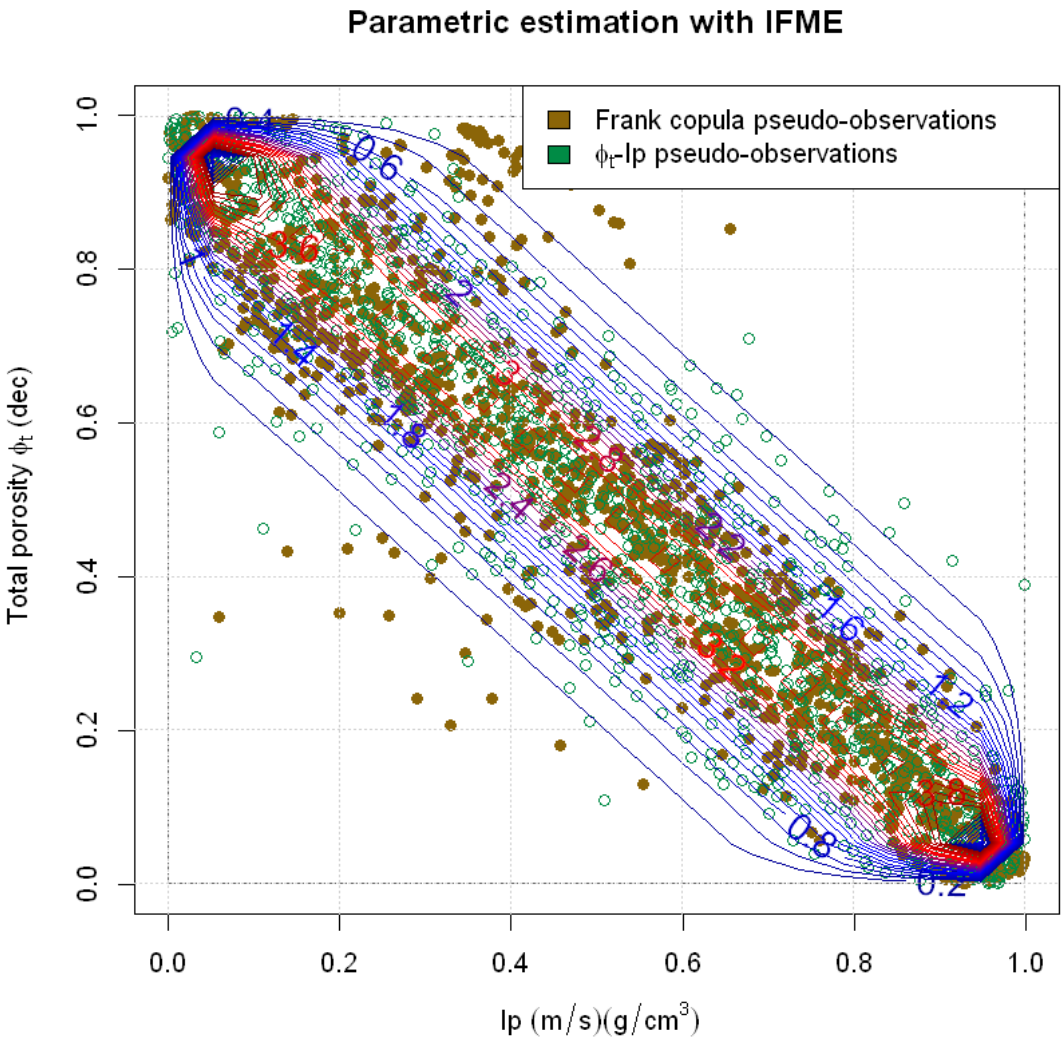
[37]:
```
fc <- frankCopula(ifme_frank@copula@parameters)
fc
```

```
Frank copula, dim. d = 2
Dimension:  2
Parameters:
  alpha   = -15.04194
```

[38]:
```
mfc <- mvdc(fc, margins = c("lnorm", "weibull"),
            paramMargins = list(list(meanlog = 8.905543, sdlog = 0.191057),
                                list(shape = 5.1646292, scale = 0.2244496)))
```

[39]:
```
Xf <- rMvdc(n, mvdc = mfc)
Xfpobs=pobs(Xf)
```

[26]:
```
cols = rev(colorRampPalette(c('darkred','red','blue','darkblue'))(25))


plot(data_pseudo, pch=16, col=("darkgoldenrod4"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Xfpobs, xlab=(""), ylab=(""), main = ("Parametric estimation with IFME"),␣
↪col=("springgreen4"))
par(new=TRUE)
contour(fc, dCopula,n=20,nlevels=25, xlab = expression(paste(" Ip ",(m/s)(g/
↪cm^3))),
    ylab = expression(paste(" Total porosity ",phi[t], " (dec)")) ,
       labcex=1.4, col=cols,xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend = c("Frank copula pseudo-observations",
                                  expression(paste(phi[t],"-Ip␣
↪pseudo-observations"))),
      fill = c("darkgoldenrod4", "springgreen4"))
```



With this results, the prior information is composed of five parameters: $(\log \mu, \log \sigma)$ for marginal u or acoustic impedance, (α, λ) for marginal v or total porosity and Frank copula (θ).

$$\mathbb{P}(\mathbf{m}) = \pi(\log \mu, \log \sigma, \alpha, \lambda, \theta) \tag{23}$$

Not all the prior parameters are used, so its neccesary to try with differents combinations. Each parameter is associated to a normal function except the parameter θ, this parameter is represented as uniform function.

The likelihoood function is composed by Frank copula density function

$$\mathbb{P}(\mathbf{d}|\mathbf{m}) = \frac{\theta\left(1-e^{-\theta}\right)e^{-\theta(u+v)}}{\left(e^{-\theta}-1+\left(e^{-\theta u}-1\right)\left(e^{-\theta v}-1\right)\right)^2} \cdot f_1(x_1; \log \mu, \log \sigma) \cdot f_2(x_2; \alpha, \lambda) \tag{24}$$

where f_1 is the lognormal density function with acoustic impedance samples (x_1) and f_2 is Weibull density function with total porosity samples (x_2).

6.4 Posterior model

The posterior function is

$$\mathbb{P}(\mathbf{m}|\mathbf{d}) = \frac{\theta\left(1-e^{-\theta}\right)e^{-\theta(u+v)}}{\left(e^{-\theta}-1+\left(e^{-\theta u}-1\right)\left(e^{-\theta v}-1\right)\right)^2} \cdot f_1(x_1;\log\mu,\log\sigma)\cdot f_2(x_2;\alpha,\lambda)\cdot\pi(\log\mu,\log\sigma,\alpha,\lambda,\theta) \tag{25}$$

To implement the Bayesian inference of the joint probability distribution function, the covariance matrix was constructed using the error values obtained from the prior estimation as shown in the following line

[41]:
```
var_theta_post<-(0.433)^2 #frank itau

#lognormal
var_mulog_post<-(0.005752746)^2
var_sigmalog_post<-(0.004067304)^2
#weibull
var_alpha_post<-(0.129598464)^2 #shape
var_lambda_post<- (0.001367821)^2 #scale
```

[42]:
```
v_tot_post_5<-diag(c(var_theta_post,var_mulog_post,var_sigmalog_post,var_alpha_post,var_lambda_
```

[43]:
```
start_5=array(c(-15,8.8,0.15,5,0.2),c(1,5))
```

For the search function, the use of the normal function is considered for all parameters. the mean value was established using the values of the a priori parameters and the standard deviation was selected as reference the percentage of the a priori value in such a way that its value allows the probability of acceptance of the random walk metropolis-Hastings method to be within the interval 20% to 30%, this interval is taken due to robustness in rate acceptances. The work published by [?] suggest the optimal acceptance rate is 23.4%. The number of iterations is 10,000.

Parameters	Prior value	Standard deviation (%)
Lognormal (μ)	8.905543	3.1
Lognormal (σ)	0.191057	3.1
Weibull (λ)	0.2244496	3.4
Frank copula (θ)	-15.04	1.5

[85]:
```
logpost_5 <- function(theta, data){
  par_cop <-theta[1]
  mulog <- theta[2]
  sigmalog <- theta[3]
  alpha <- theta[4]
  lambda <- theta[5]
```

```
  #theta_prior=dunif(par_cop, min=-15.5, max=-14.5, log=TRUE)
  theta_prior=dnorm(par_cop, mean=-15.04, sd=((15.04/100)*1.5), log=TRUE)
  mulog_prior=dnorm(mulog, mean=8.905543, sd=((8.905543/100)*3.1), log=TRUE)
  sigmalog_prior=dnorm(sigmalog, mean=0.191057, sd=((0.191057/100)*3.1),␣
↪log=TRUE)
  #alpha_prior=dnorm(alpha, mean=5.1646292, sd=((5.1646292/100)*3.5), log=TRUE)
  lambda_prior=dnorm(lambda, mean=0.2244496, sd=((0.2244496/100)*3.4), log=TRUE)



  u_like<-dlnorm(data[,1],mulog,sigmalog, log = TRUE)
  v_like<-dweibull(data[,2],alpha,lambda, log = TRUE)
  cop_like<-dCopula(pobs(data),frankCopula(param = par_cop), log = TRUE)

 ␣
↪#theta_prior+mulog_prior+sigmalog_prior+alpha_prior+lambda_prior+cop_like+u_like+v_like
    theta_prior+mulog_prior+sigmalog_prior+lambda_prior+cop_like+u_like+v_like
}
```

[ ]:
```
mcmc_5.fit <-rwmetrop(logpost_5,
                    list(var=v_tot_post_5,scale=2),
                    start_5,
                    10000,
                    Data_like)
```

[87]:
```
mcmc_5.fit$accept
```

0.2126

The acceptance probability obtained is 21.6%, now the posterior parameter value is selected in terms of mode observed in the histograms of each parameter.

[88]:
```
frank_theta_post<-Estadisticas(mcmc_5.fit$par[,1])
lnorm_mu_post<-Estadisticas(mcmc_5.fit$par[,2])
lnorm_sigma_post<-Estadisticas(mcmc_5.fit$par[,3])
weibull_alpha_post<-Estadisticas(mcmc_5.fit$par[,4])
weibull_lambda_post<-Estadisticas(mcmc_5.fit$par[,5])
```

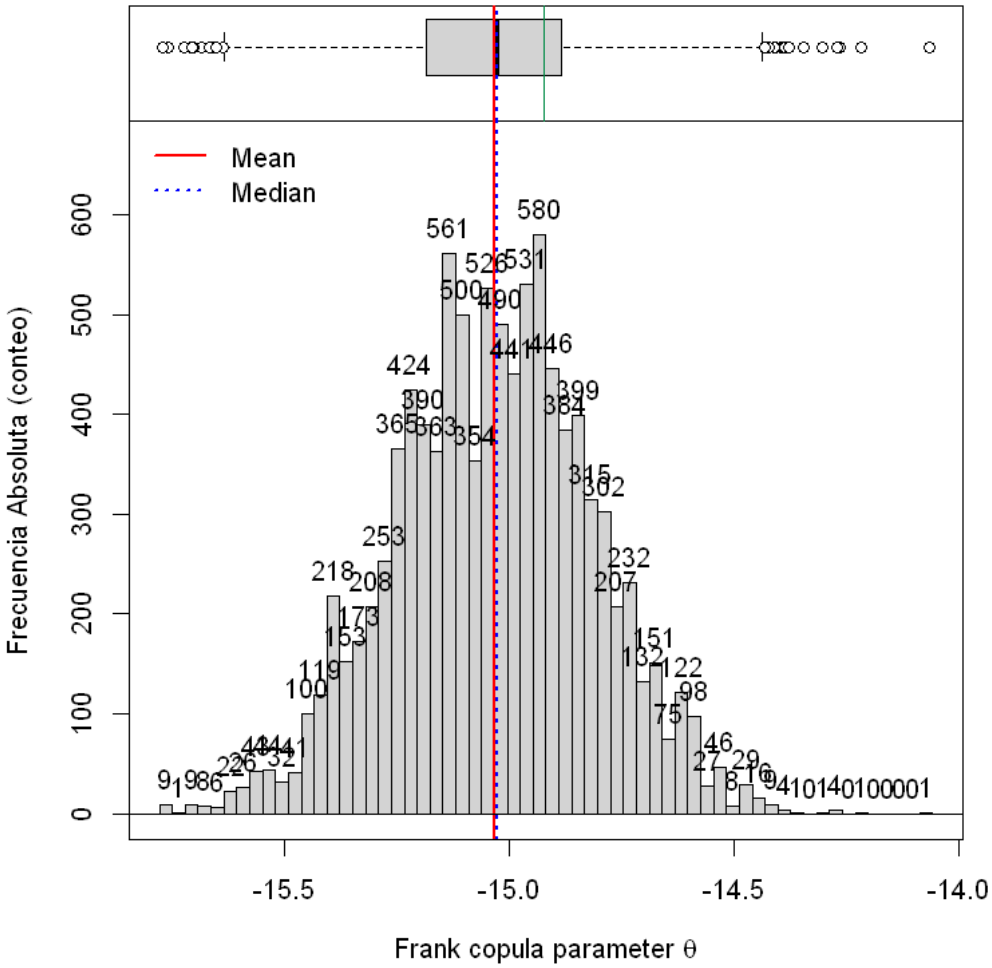
The the posterior parameter value θ has a value of -14.922, the difference with the prior value is 0.24, this value is accepted because the dependence measures between welllog values and upscaled values is very close.

[7]:
```
theta_best_post5=mfv(round(mcmc_5.fit$par[,1],3))
HistBoxplot(mcmc_5.fit$par[,1],nbins = 60, mean = frank_theta_post[5,2], median␣
↪= frank_theta_post[4,2], main ="",
            xlab = expression(paste("Frank copula parameter ",theta )), ylab =␣
↪"Frecuencia Absoluta (conteo)",
```

```
            AbsFreq = TRUE, PercentFreq = FALSE )
par(new=TRUE)
abline(v=theta_best_post5, col=("springgreen4"))
frank_theta_post
print(paste('The Frank copula parameter theta is',theta_best_post5))
```

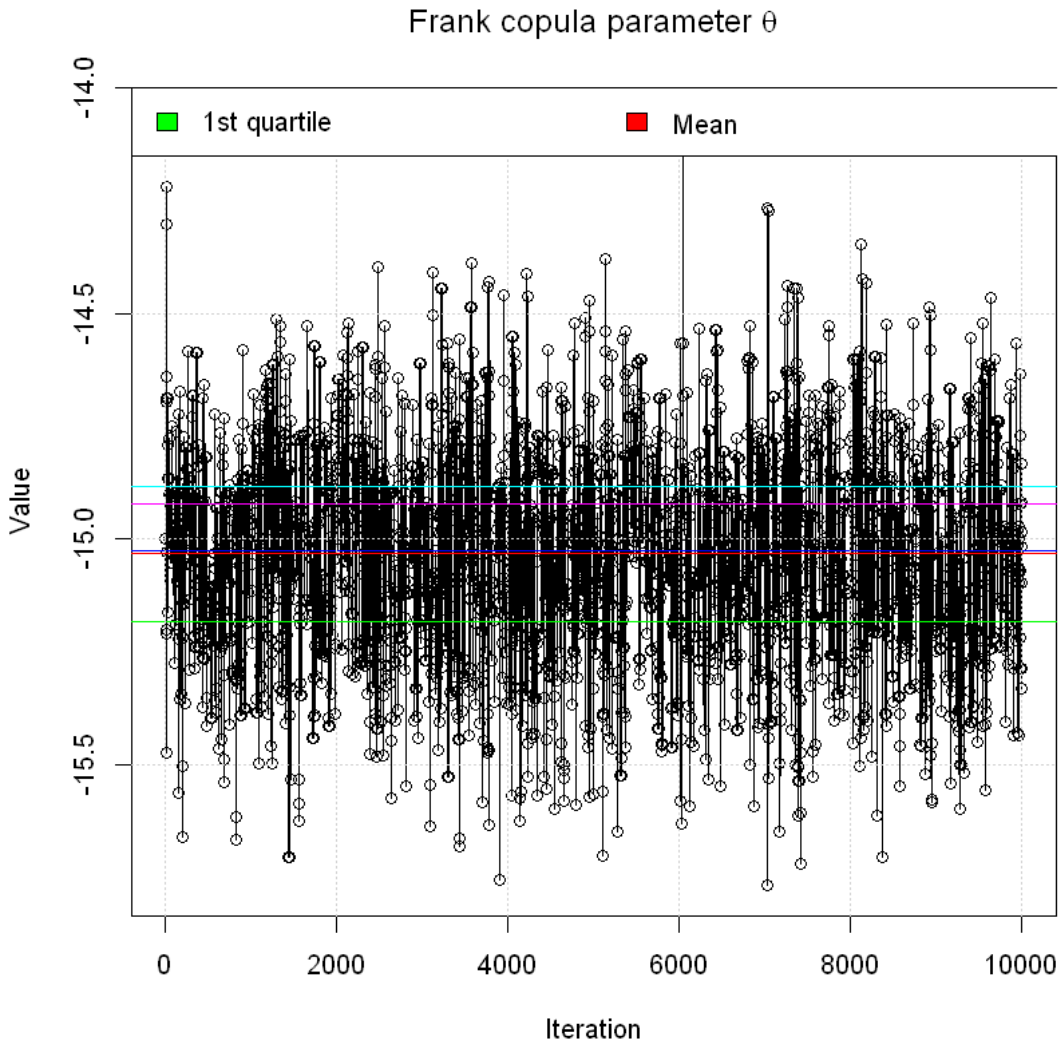
	Statistics <chr>	Values <dbl>
muestras	n	10000.0000
minimos	Minimum	-15.7686
cuantiles1	1st. Quartile	-15.1845
medianas	Median	-15.0263
medias	Mean	-15.0332
cuantiles3	3rd. Quartile	-14.8834
maximos	Maximum	-14.0668
rangos	Rank	1.7018
rangosInt	Interquartile Rank	0.3011
varianzas	Variance	0.0489
desvs	Standard Deviation	0.2212
CVs	Variation Coeff.	-0.0147
simetrias	Skewness	-0.0523
curtosiss	Kurtosis	2.9671

```
[1] "The Frank copula parameter theta is -14.922"
```



[8]:
```
plot(mcmc_5.fit$par[,1], xlab = "Iteration", ylab = "Value", type = "o",
     main =expression(paste("Frank copula parameter ",theta )))
par(new=TRUE)
abline(h=frank_theta_post[3,2],col=("green"))
par(new=TRUE)
abline(h=frank_theta_post[4,2],col=("blue"))
par(new=TRUE)
abline(h=frank_theta_post[5,2],col=("red"))
par(new=TRUE)
abline(h=frank_theta_post[6,2],col=("cyan"))
par(new=TRUE)
abline(h=theta_best_post5,col=("magenta"))
grid()
```

```
legend(x = "topleft", legend = c("1st quartile","Mean","Median","3rd quartile",
                                 expression(paste("Best Frank copula parameter␣
 ↪",theta ))),
       fill = c("green","red","blue","cyan","magenta"), horiz=TRUE)
```



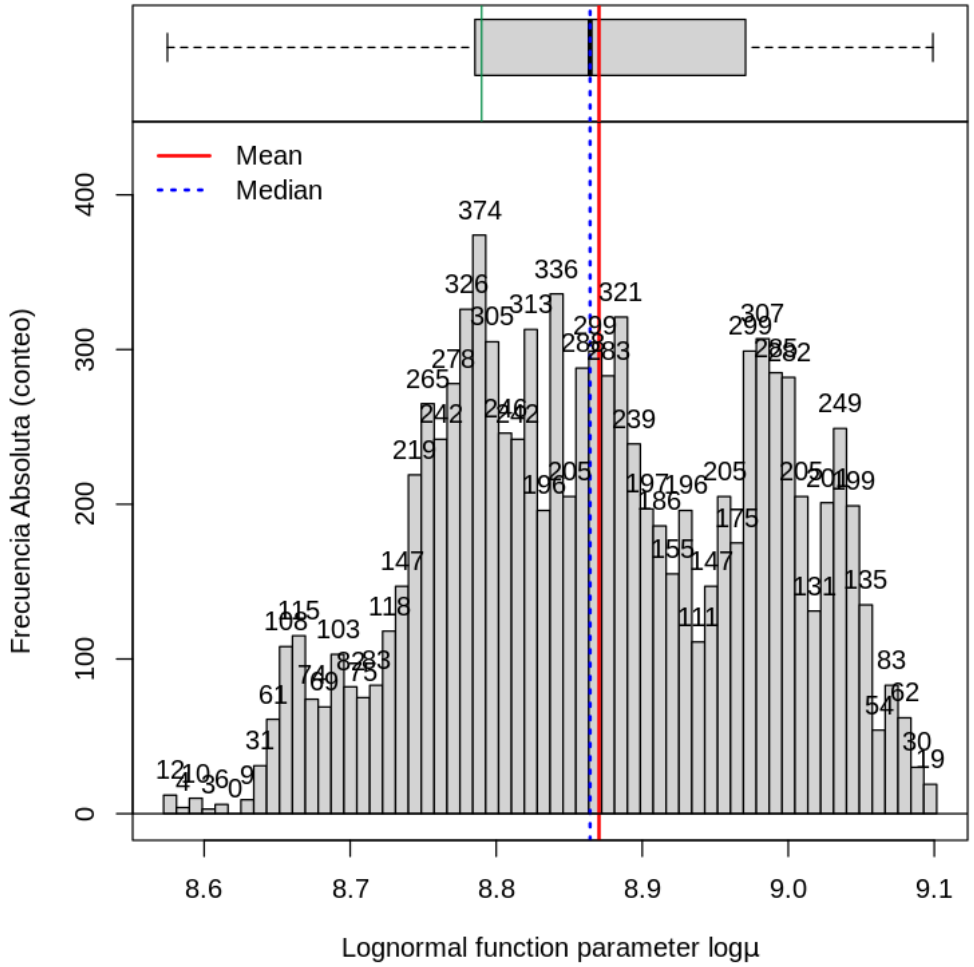
The posterior parameter value $Log\mu$ is 8.79, the prior parameter value is 8.9055, then the difference is close, near to 1.29%

[93]:
```
mu_best_post5=mfv(round(mcmc_5.fit$par[,2],2))
HistBoxplot(mcmc_5.fit$par[,2],nbins = 60, mean = lnorm_mu_post[5,2], median =␣
 ↪lnorm_mu_post[4,2], main ="",
            xlab = expression(paste("Lognormal function parameter log",mu )),
```

```
            ylab = "Frecuencia Absoluta (conteo)", AbsFreq = TRUE, PercentFreq =␣
 ↪FALSE )
par(new=TRUE)
abline(v=mu_best_post5, col=("springgreen4"))

lnorm_mu_post
print(paste('The Lognormal density function parameter mu is',mu_best_post5))
```

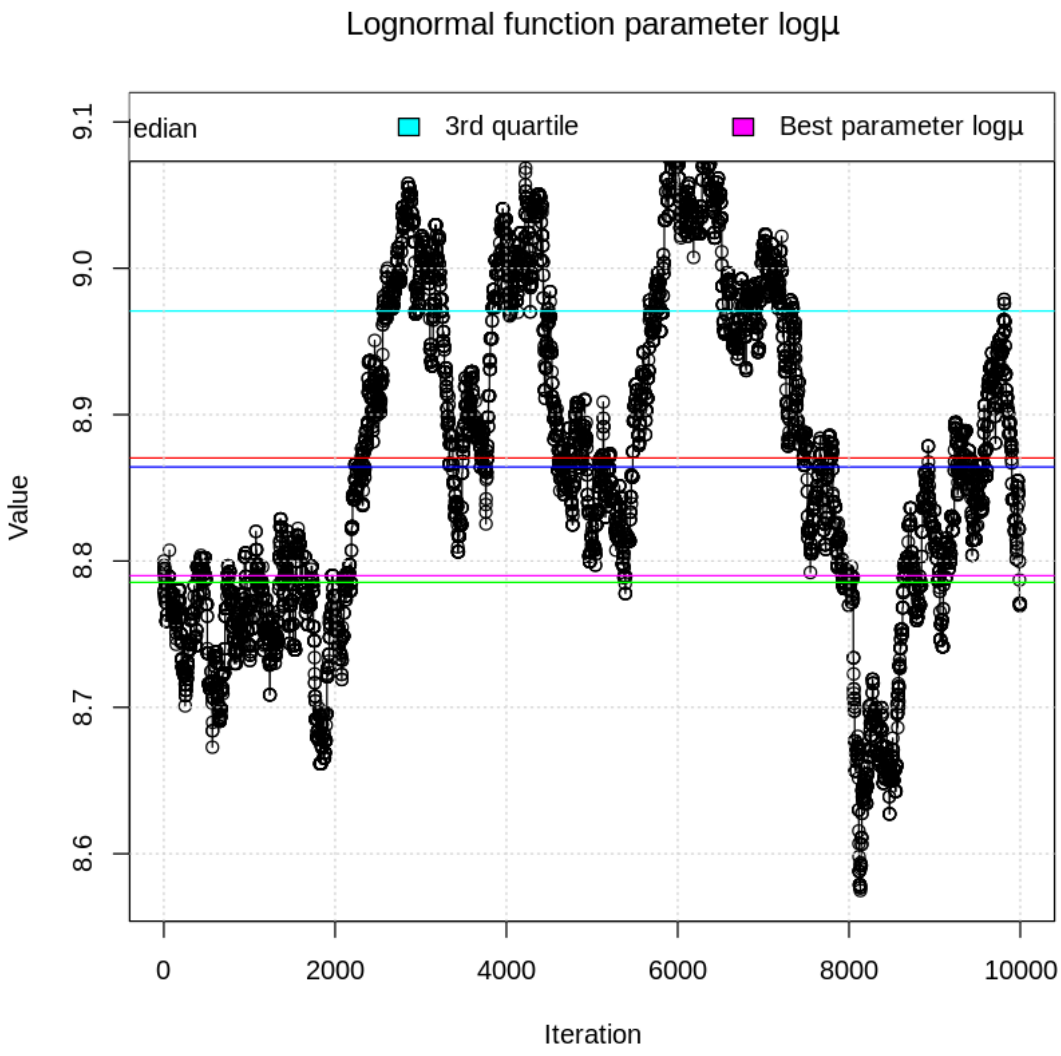
	Statistics <chr>	Values <dbl>
muestras	n	10000.0000
minimos	Minimum	8.5749
cuantiles1	1st. Quartile	8.7854
medianas	Median	8.8643
medias	Mean	8.8704
cuantiles3	3rd. Quartile	8.9708
maximos	Maximum	9.0990
rangos	Rank	0.5242
rangosInt	Interquartile Rank	0.1853
varianzas	Variance	0.0122
desvs	Standard Deviation	0.1104
CVs	Variation Coeff.	0.0125
simetrias	Skewness	-0.0242
curtosiss	Kurtosis	2.1356

```
[1] "The Lognormal density function parameter mu is 8.79"
```



[94]:
```
plot(mcmc_5.fit$par[,2], xlab = "Iteration", ylab = "Value", type = "o",
     main =expression(paste("Lognormal function parameter log",mu )))
par(new=TRUE)
abline(h=lnorm_mu_post[3,2],col=("green"))
par(new=TRUE)
abline(h=lnorm_mu_post[4,2],col=("blue"))
par(new=TRUE)
abline(h=lnorm_mu_post[5,2],col=("red"))
par(new=TRUE)
abline(h=lnorm_mu_post[6,2],col=("cyan"))
par(new=TRUE)
abline(h=mu_best_post5,col=("magenta"))
grid()
```

```
legend(x = "topright", legend = c("1st quartile","Mean","Median","3rd quartile",
                                  expression(paste("Best parameter log",mu ))),
       fill = c("green","red","blue","cyan","magenta"), horiz=TRUE)
```



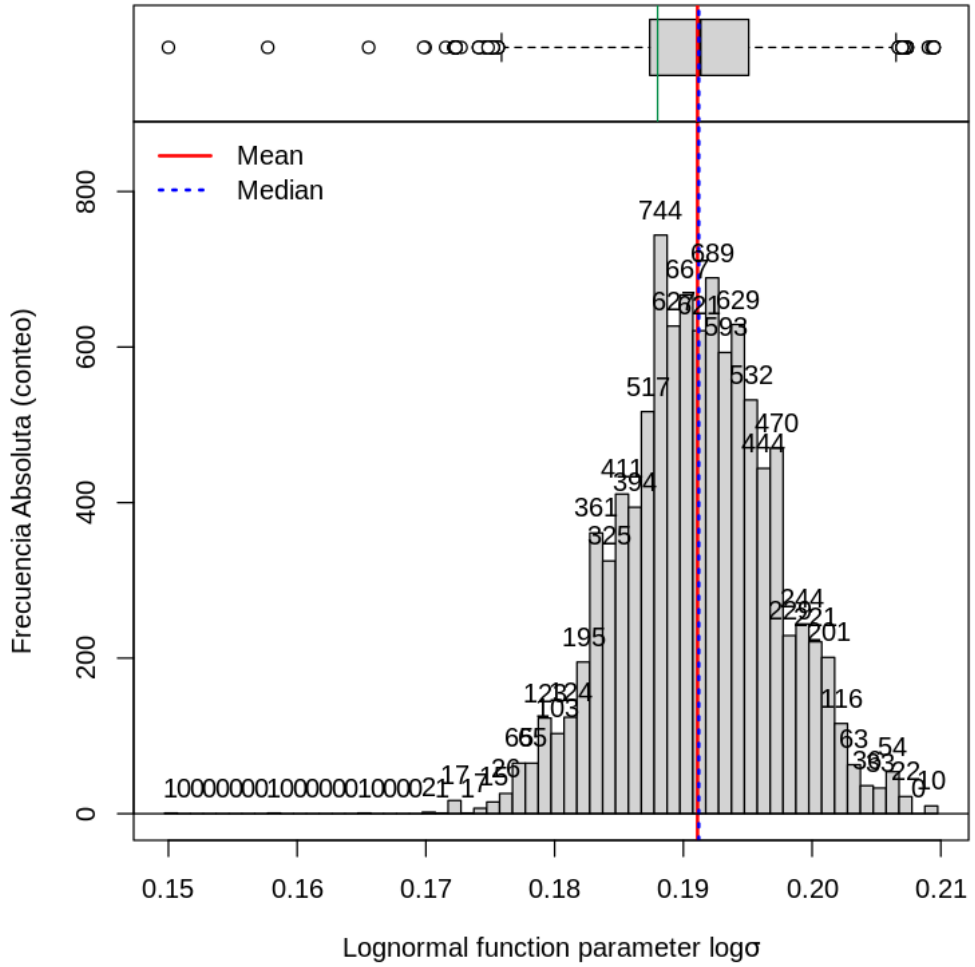
The posterior parameter value $Log\sigma$ is 0.188, the prior parameter value is 0.191057, then the difference is very low.

[96]:
```
sigma_best_post5=mfv(round(mcmc_5.fit$par[,3],3))
HistBoxplot(mcmc_5.fit$par[,3],nbins = 60, mean = lnorm_sigma_post[5,2], median␣
↪= lnorm_sigma_post[4,2],
            main ="",
            xlab = expression(paste("Lognormal function parameter log",sigma )),
            ylab = "Frecuencia Absoluta (conteo)",
```

```
            AbsFreq = TRUE, PercentFreq = FALSE )
par(new=TRUE)
abline(v=sigma_best_post5, col=("springgreen4"))
lnorm_sigma_post
print(paste('The Lognormal density function parameter sigma␣
 ↪is',sigma_best_post5))
```

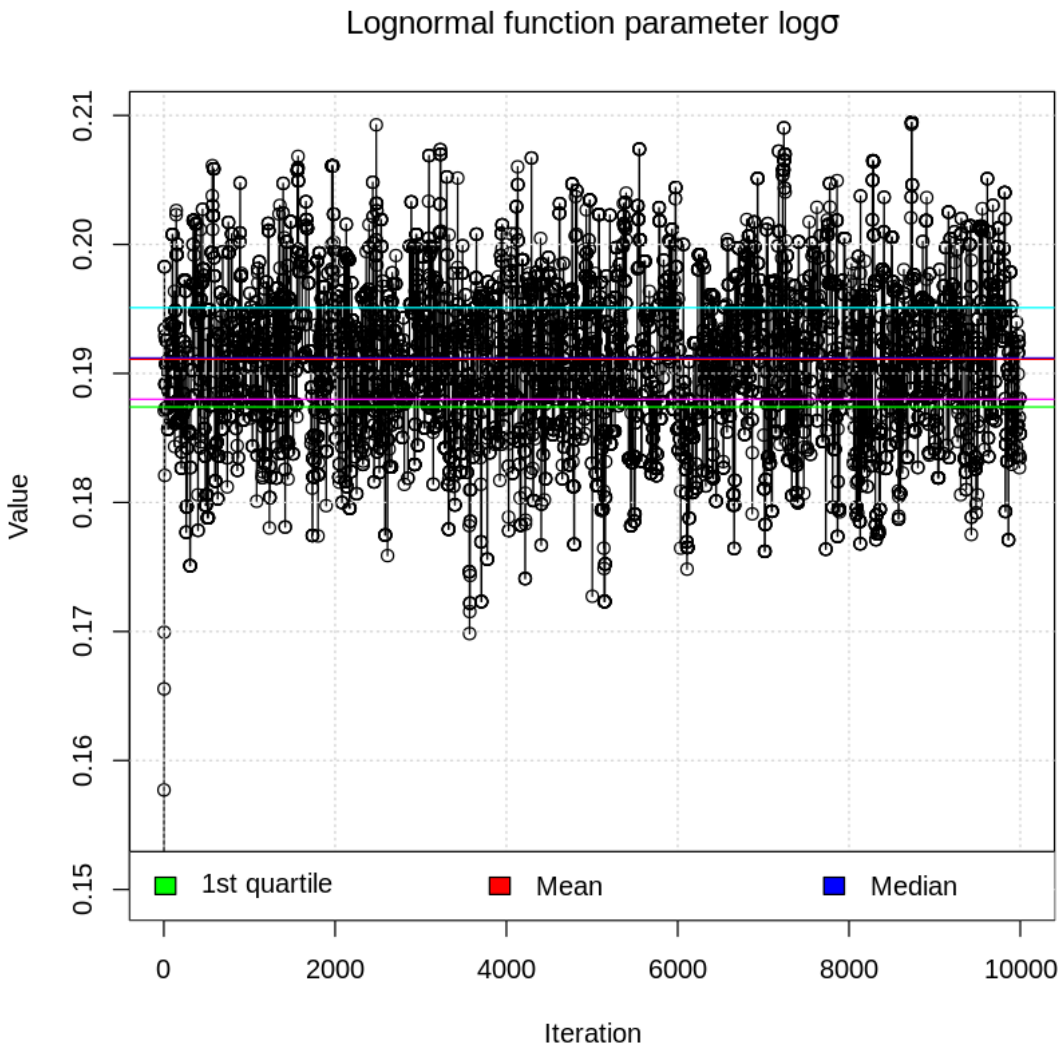
	Statistics <chr>	Values <dbl>
muestras	n	10000.0000
minimos	Minimum	0.1500
cuantiles1	1st. Quartile	0.1874
medianas	Median	0.1912
medias	Mean	0.1911
cuantiles3	3rd. Quartile	0.1951
maximos	Maximum	0.2095
rangos	Rank	0.0595
rangosInt	Interquartile Rank	0.0077
varianzas	Variance	0.0000
desvs	Standard Deviation	0.0059
CVs	Variation Coeff.	0.0310
simetrias	Skewness	-0.0887
curtosiss	Kurtosis	3.2900

```
[1] "The Lognormal density function parameter sigma is 0.188"
```



[97]:
```
plot(mcmc_5.fit$par[,3], xlab = "Iteration", ylab = "Value", type = "o",
     main =expression(paste("Lognormal function parameter log",sigma )))
par(new=TRUE)
abline(h=lnorm_sigma_post[3,2],col=("green"))
par(new=TRUE)
abline(h=lnorm_sigma_post[4,2],col=("blue"))
par(new=TRUE)
abline(h=lnorm_sigma_post[5,2],col=("red"))
par(new=TRUE)
abline(h=lnorm_sigma_post[6,2],col=("cyan"))
par(new=TRUE)
abline(h=sigma_best_post5,col=("magenta"))
grid()
```

```
legend(x = "bottomleft", legend = c("1st quartile","Mean","Median","3rd␣
 ↪quartile",
                                    expression(paste("Best parameter log",mu ))),
       fill = c("green","red","blue","cyan","magenta"), horiz=TRUE)
```



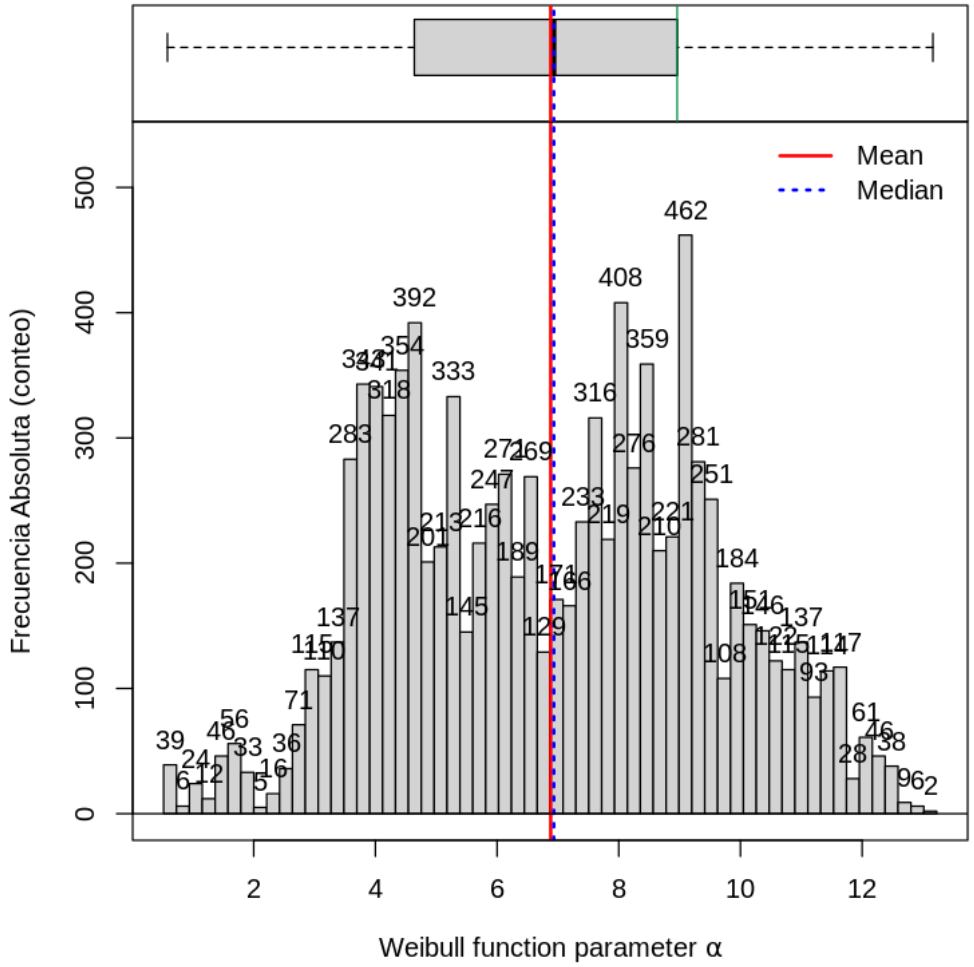
The posterior parameter value α is 8.96, the prior parameter value is 5.1646292, then the difference is 3.7954.

[100]:
```
alpha_best_post5=mfv(round(mcmc_5.fit$par[,4],2))#weibull_alpha_post[3,2]

HistBoxplot(mcmc_5.fit$par[,4],nbins = 60, mean = weibull_alpha_post[5,2],␣
 ↪median = weibull_alpha_post[4,2],
            main ="",
```

```
            xlab = expression(paste("Weibull function parameter ",alpha )),
            ylab = "Frecuencia Absoluta (conteo)",
            AbsFreq = TRUE, PercentFreq = FALSE )
par(new=TRUE)
abline(v=alpha_best_post5, col=("springgreen4"))
weibull_alpha_post
print(paste('The Weibull density function parameter alpha',alpha_best_post5))
```

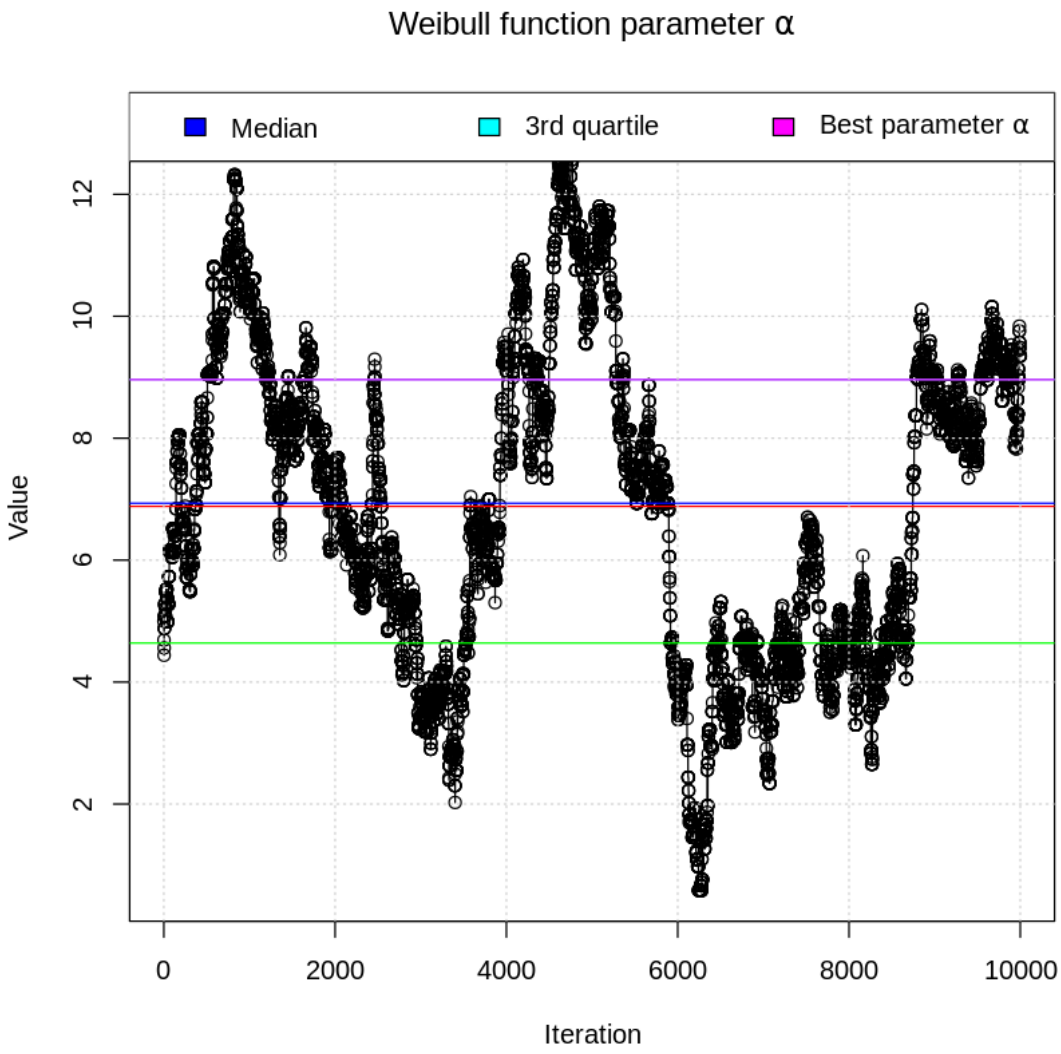
	Statistics <chr>	Values <dbl>
muestras	n	10000.0000
minimos	Minimum	0.5817
cuantiles1	1st. Quartile	4.6398
medianas	Median	6.9327
medias	Mean	6.8817
cuantiles3	3rd. Quartile	8.9641
maximos	Maximum	13.1630
rangos	Rank	12.5813
rangosInt	Interquartile Rank	4.3243
varianzas	Variance	6.6497
desvs	Standard Deviation	2.5787
CVs	Variation Coeff.	0.3747
simetrias	Skewness	0.0453
curtosiss	Kurtosis	2.1556

```
[1] "The Weibull density function parameter alpha 8.96"
```



[101]:
```
plot(mcmc_5.fit$par[,4], xlab = "Iteration", ylab = "Value", type = "o",
     main =expression(paste("Weibull function parameter ",alpha )))
par(new=TRUE)
abline(h=weibull_alpha_post[3,2],col=("green"))
par(new=TRUE)
abline(h=weibull_alpha_post[4,2],col=("blue"))
par(new=TRUE)
abline(h=weibull_alpha_post[5,2],col=("red"))
par(new=TRUE)
abline(h=weibull_alpha_post[6,2],col=("cyan"))
par(new=TRUE)
abline(h=alpha_best_post5,col=("magenta"))
grid()
```

```
legend(x = "topright", legend = c("1st quartile","Mean","Median","3rd quartile",
                                  expression(paste("Best parameter ",alpha ))),
       fill = c("green","red","blue","cyan","magenta"), horiz=TRUE)
```



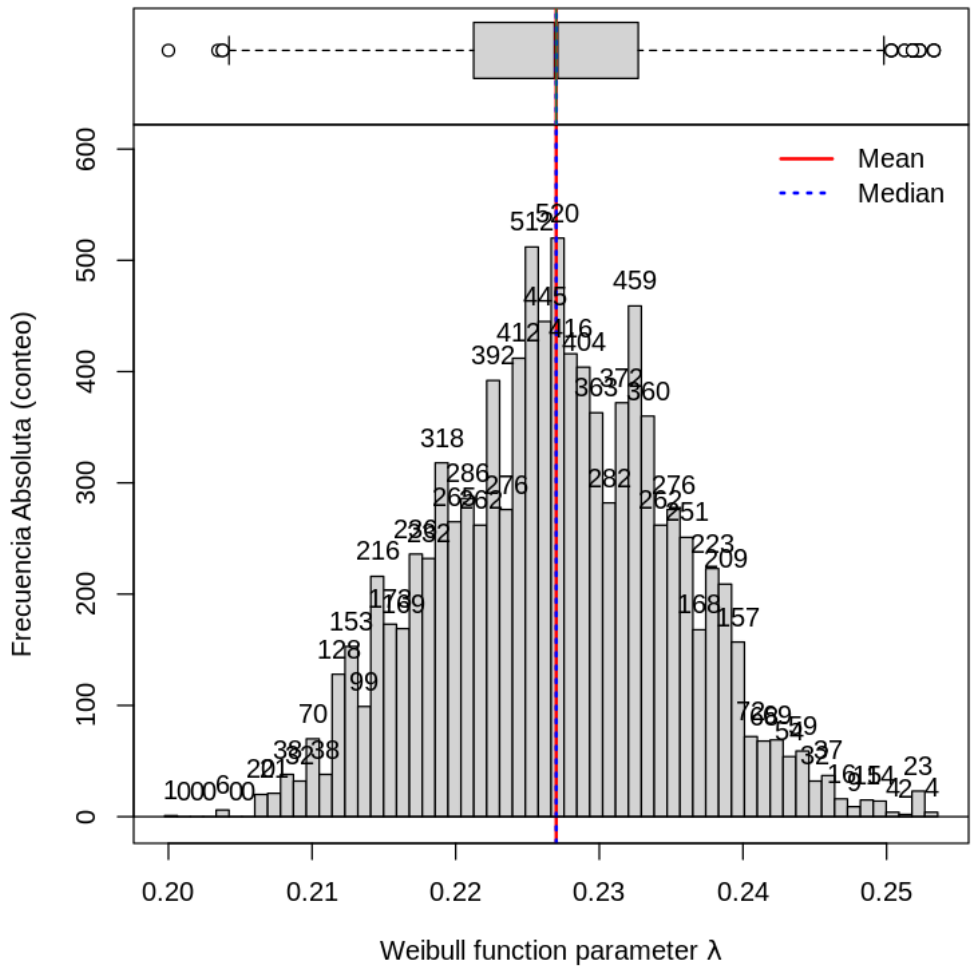
The posterior parameter value λ is 0.225, the prior parameter value is 0.2244496, then the difference is very low.

[102]:
```
lambda_best_post5=mfv(round(mcmc_5.fit$par[,5],3))
HistBoxplot(mcmc_5.fit$par[,5],nbins = 60, mean = weibull_lambda_post[5,2],␣
 ↪median = weibull_lambda_post[4,2],
            main ="",
            xlab = expression(paste("Weibull function parameter ",lambda )),
            ylab = "Frecuencia Absoluta (conteo)",
```

```
            AbsFreq = TRUE, PercentFreq = FALSE )
par(new=TRUE)
abline(v=lambda_best_post5, col=("springgreen4"))
weibull_lambda_post
print(paste('The Weibull density function parameter lambda',lambda_best_post5))
```

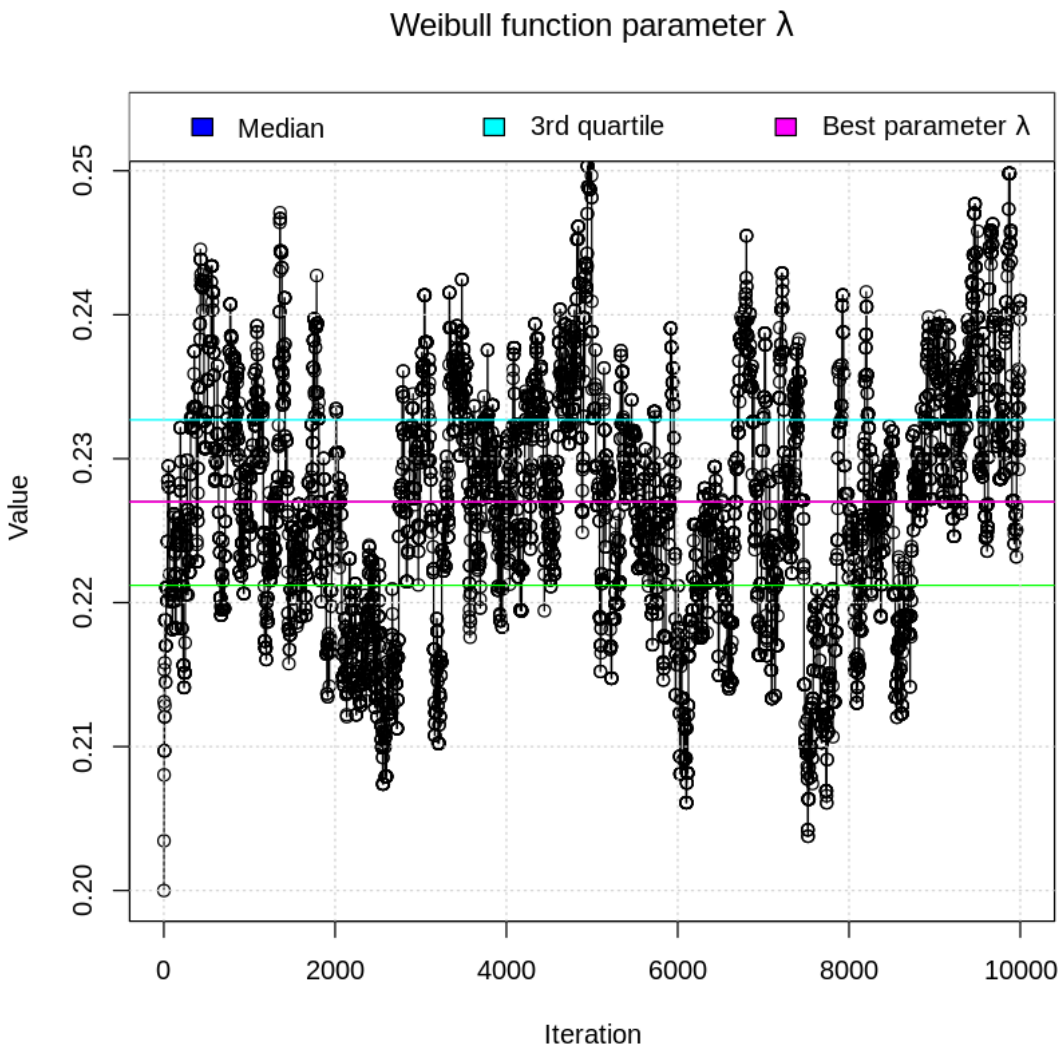
	Statistics <chr>	Values <dbl>
muestras	n	10000.0000
minimos	Minimum	0.2000
cuantiles1	1st. Quartile	0.2212
medianas	Median	0.2270
medias	Mean	0.2270
cuantiles3	3rd. Quartile	0.2327
maximos	Maximum	0.2533
rangos	Rank	0.0533
rangosInt	Interquartile Rank	0.0115
varianzas	Variance	0.0001
desvs	Standard Deviation	0.0082
CVs	Variation Coeff.	0.0362
simetrias	Skewness	0.0417
curtosiss	Kurtosis	2.7726

```
[1] "The Weibull density function parameter lambda 0.227"
```



[103]:
```
plot(mcmc_5.fit$par[,5], xlab = "Iteration", ylab = "Value", type = "o",
     main =expression(paste("Weibull function parameter ",lambda )))
par(new=TRUE)
abline(h=weibull_lambda_post[3,2],col=("green"))
par(new=TRUE)
abline(h=weibull_lambda_post[4,2],col=("blue"))
par(new=TRUE)
abline(h=weibull_lambda_post[5,2],col=("red"))
par(new=TRUE)
abline(h=weibull_lambda_post[6,2],col=("cyan"))
par(new=TRUE)
abline(h=lambda_best_post5,col=("magenta"))
grid()
```

```
legend(x = "topright", legend = c("1st quartile","Mean","Median","3rd quartile",
                                  expression(paste("Best parameter ",lambda ))),
       fill = c("green","red","blue","cyan","magenta"), horiz=TRUE)
```

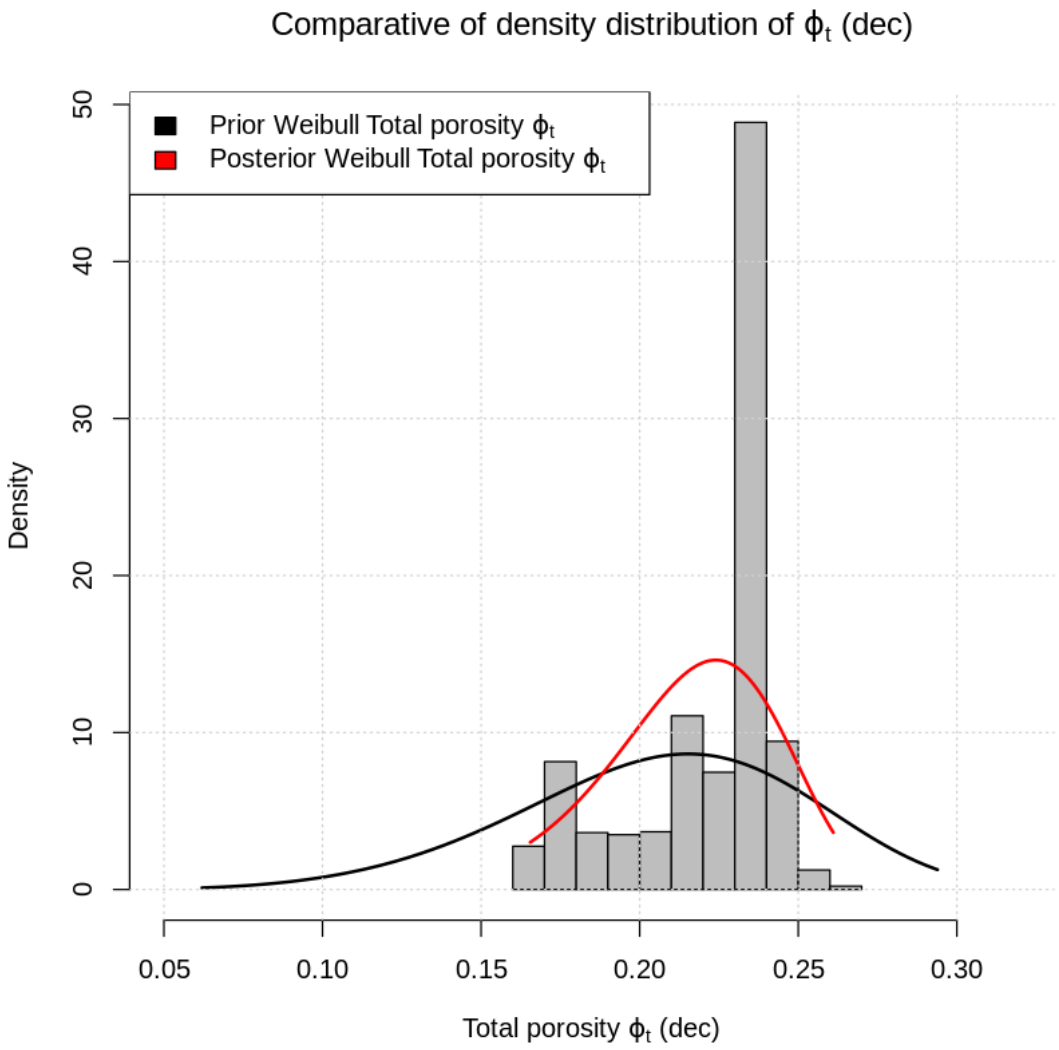


[105]:
```
cop_frank_postF <- mvdc(frankCopula(theta_best_post5), margins =␣
↪c("lnorm","weibull"),
                        paramMargins = list(list(meanlog = mu_best_post5, sdlog =␣
↪sigma_best_post5),
                                            list(shape = alpha_best_post5, scale␣
↪= lambda_best_post5)))

cop_frank_post_ran5F <- rMvdc(3696, mvdc = cop_frank_postF)
```

Comparing the prior and posterior parameter values for weibull density function, the posterior density function moves a little to the right and tries to cover the histogram.

parameters	Prior value	Posterior value	Difference
Lognormal (μ)	8.905543	8.79	0.115543
Lognormal (σ)	0.191057	0.188	0.003057
Weibull (α)	5.1646292	8.96	3.7953708
Weibull (λ)	0.2244496	0.227	0.0025504
Frank copula (θ)	-15.04	-14.922	0.118

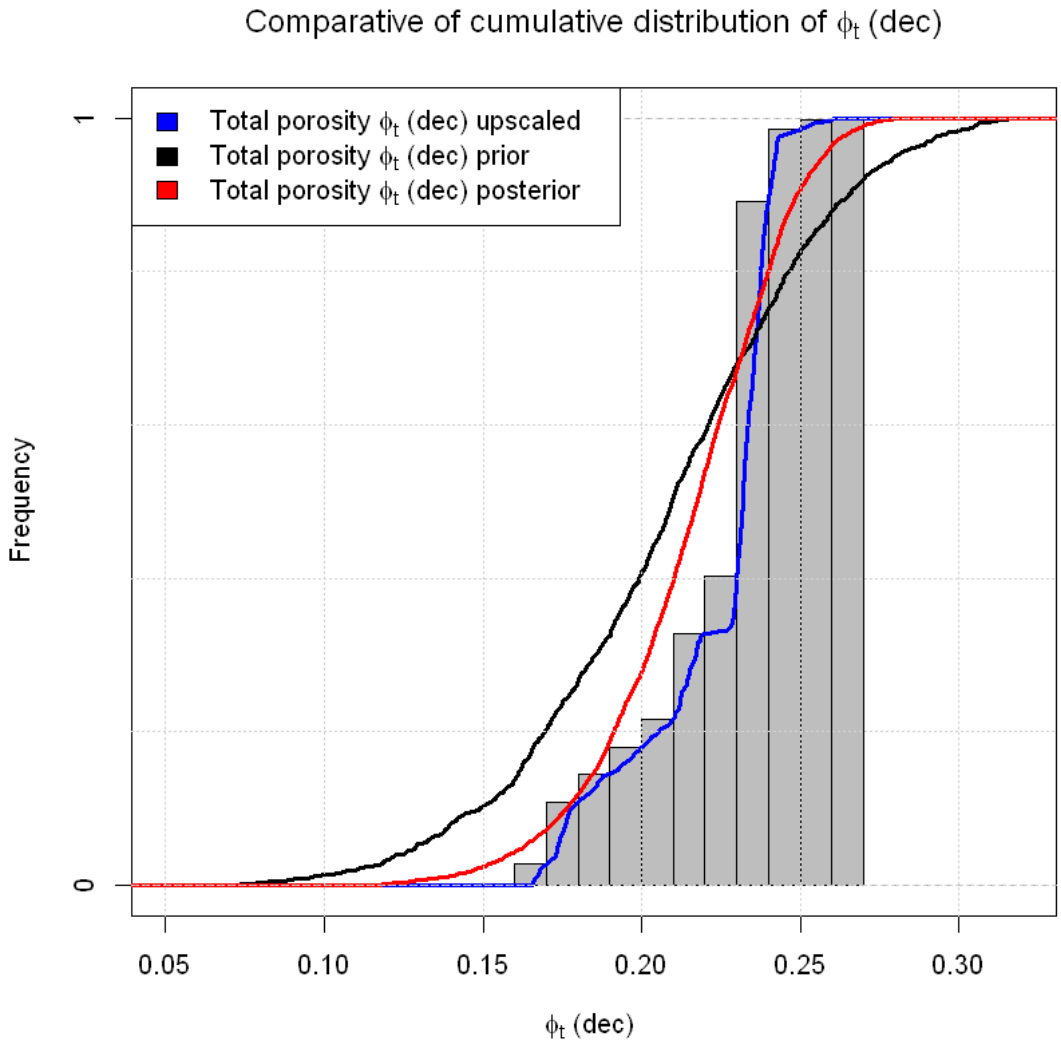
[106]:
```
den.x <- seq(min(Phit),max(Phit),length=3696)
den.y_prior <- dweibull(den.x,shape = 5.1646292, scale = 0.2244496)

den.x_post5F <- seq(min(Phit_like),max(Phit_like),length=3696)
den.y_post5F <- dweibull(den.x_post5F,shape = alpha_best_post5, scale =␣
↪lambda_best_post5)

hist(Phit_like, breaks=ns, col="gray", probability=TRUE,xlim=c(0.05, 0.32),
     main=(expression(paste(" Comparative of density distribution of␣
↪",varphi[t], " (dec) "))),
    xlab=(expression(paste(" Total porosity ",varphi[t], " (dec) "))))
lines(den.x, den.y_prior,xlim=c(0.05, 0.32), col="black", lwd=2)
lines(den.x_post5F, den.y_post5F,xlim=c(0.05, 0.32), col="red", lwd=2)
grid()
legend(x = "topleft", legend = c(expression(paste(" Prior Weibull Total porosity␣
↪", varphi[t])),
                                 expression(paste(" Posterior Weibull Total␣
↪porosity ", varphi[t]))),
       fill = c("black","red"))
```



Analizing Weibull cumulative distribution function with prior and posterior parameter values, the posterior Weibull cumulative distribution function moves a little to the right.

[12]:
```
hcum <- h <- hist(Phit_like, plot=FALSE, breaks=ns)
hcum$counts <- cumsum(hcum$counts)/sum(hcum$counts)
plot(hcum, xlim=c(0.05, 0.32), col=("gray"),main =expression(paste("Comparative␣
 ↪of cumulative distribution of ",phi[t]," (dec)")),
     xlab= expression(paste(phi[t], " (dec) ")))
par(new=TRUE)
plot(ecdf(Phit_like), col=("blue"),xlim=c(0.05, 0.
 ↪32),lwd=3,xaxt='n',yaxt='n',ann=FALSE )
par(new=TRUE)
```

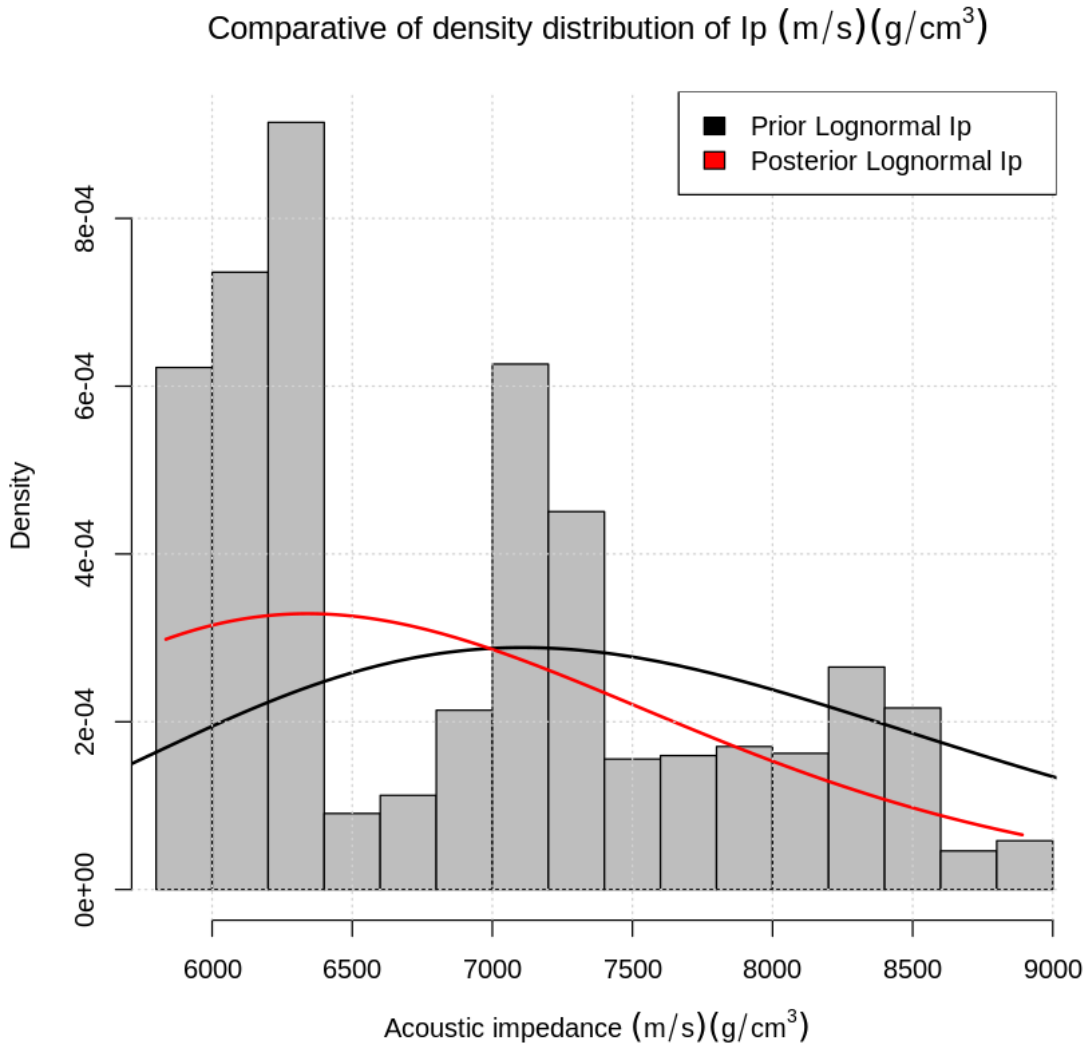
```
plot(ecdf(Xf[,2]), col=("black"),xlim=c(0.05, 0.
 ↪32),lwd=3,xaxt='n',yaxt='n',ann=FALSE) #IFME
par(new=TRUE)
plot(ecdf(cop_frank_post_ran5F[,2]), col=("red"),xlim=c(0.05, 0.
 ↪32),lwd=3,xaxt='n',yaxt='n',ann=FALSE) #Clayton Itau
grid()
legend(x = "topleft", legend = c(
expression(paste(" Total porosity ",phi[t], " (dec) upscaled")),
expression(paste(" Total porosity ",phi[t], " (dec) prior")),
expression(paste(" Total porosity ",phi[t], " (dec) posterior"))),
fill = c("blue","black","red"))
```



Meanwhile, Lognormal posterior density distribution function was moved to the left, the coverage

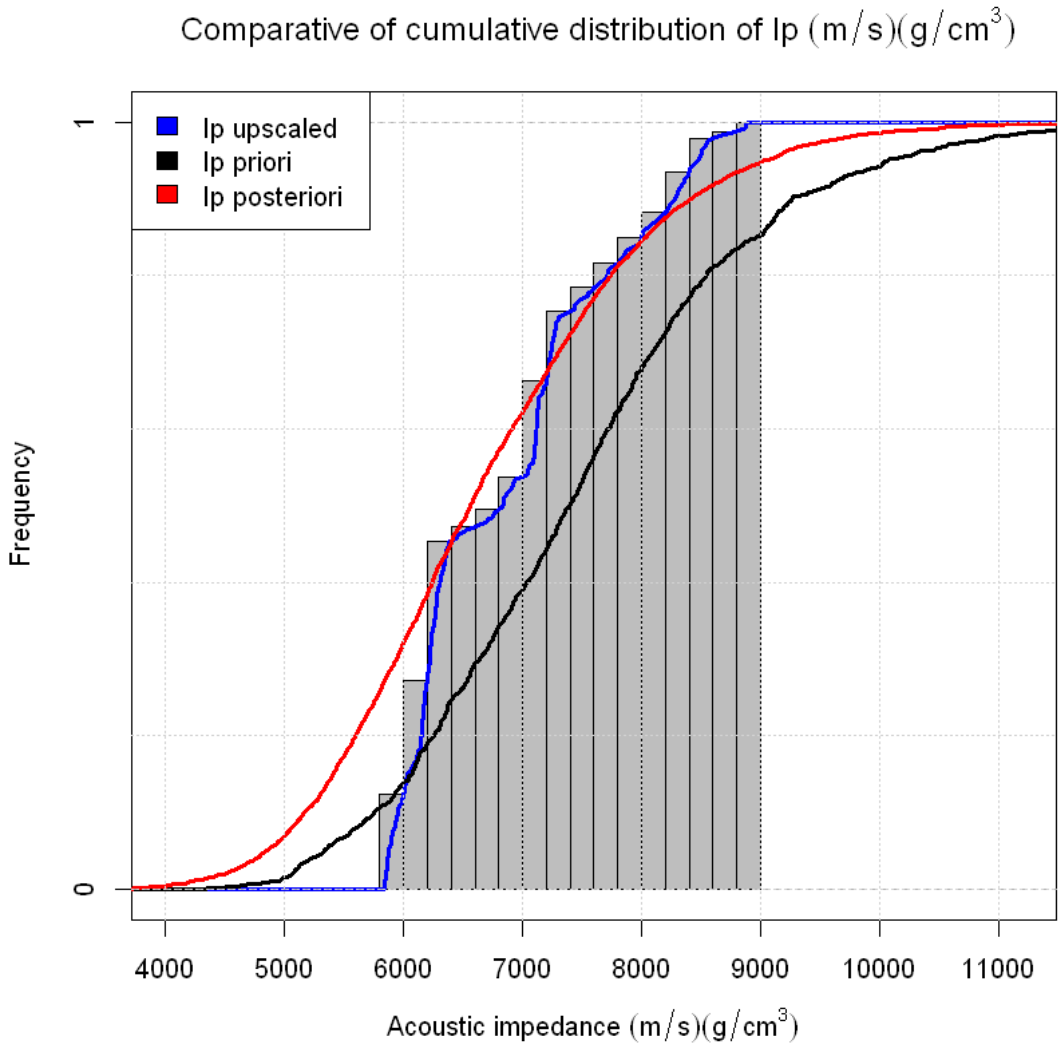
to the histogram is better.

[108]:
```
den.x <- seq(min(Ip),max(Ip),length=3696)
den.y_prior <- dlnorm(den.x,meanlog = 8.905543, sdlog = 0.191057)

den.x_post5F <- seq(min(Ip_like),max(Ip_like),length=3696)
den.y_post5F <- dlnorm(den.x_post5F,meanlog = mu_best_post5, sdlog =␣
↪sigma_best_post5)

hist(Ip_like, breaks=ns, col="gray", probability=TRUE,xlim =␣
↪c(min(Ip_like),max(Ip_like)),
    main=(expression(paste(" Comparative of density distribution of Ip ",(m/s)(g/
↪cm^3)))),
    xlab=(expression(paste(" Acoustic impedance ",(m/s)(g/cm^3)))))
lines(den.x, den.y_prior, col="black", lwd=2)
lines(den.x_post5F, den.y_post5F, col="red", lwd=2)
grid()
legend(x = "topright", legend = c("Prior Lognormal Ip", "Posterior Lognormal␣
↪Ip"),
       fill = c("black","red"))
```



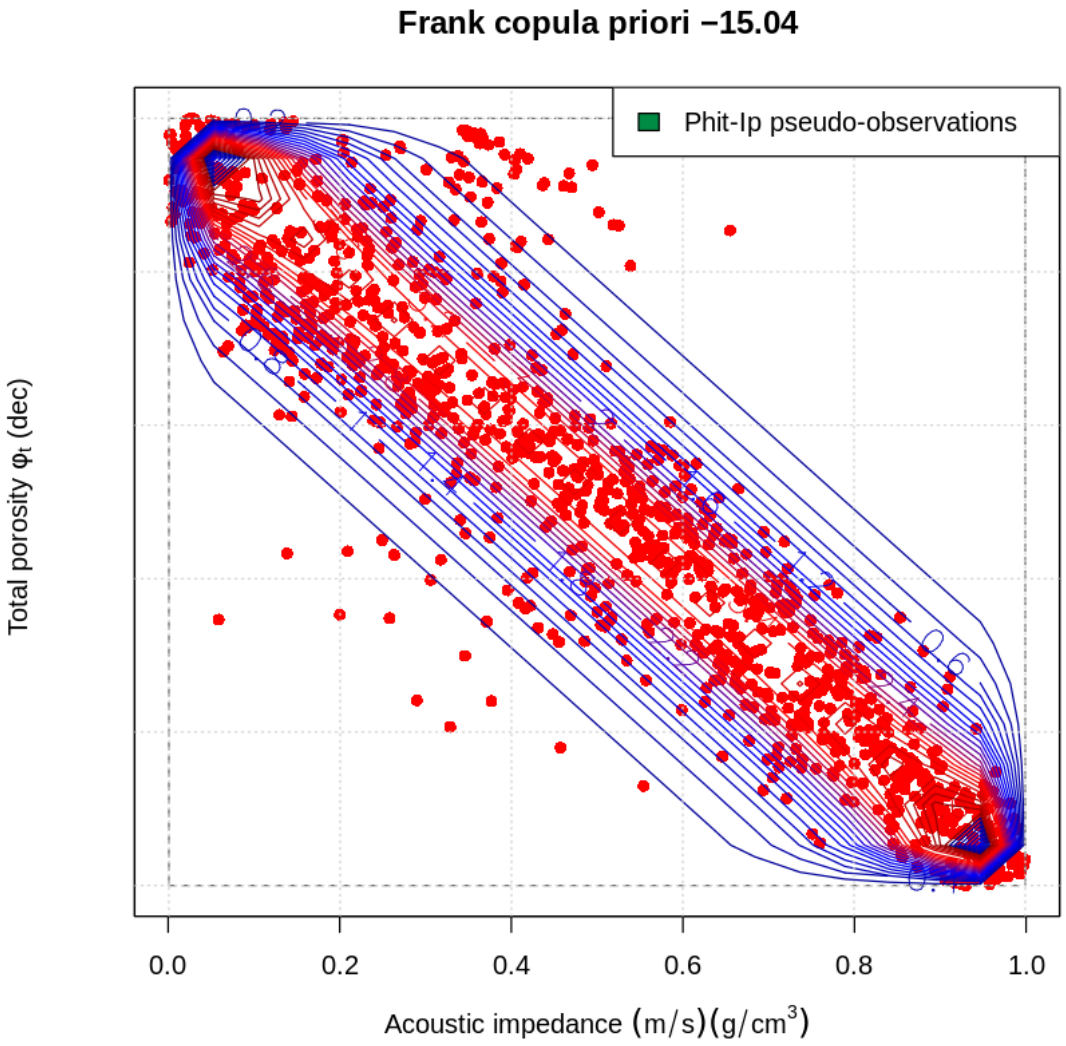
[13]:
```
hcum <- h <- hist(Ip_like, plot=FALSE, breaks=ns)
hcum$counts <- cumsum(hcum$counts)/sum(hcum$counts)
plot(hcum, xlim=c(4000, 11200), col=("gray"),main =(expression(paste("␣
 ↪Comparative of cumulative distribution of Ip ",(m/s)(g/cm^3)))),
     xlab= expression(paste("Acoustic impedance ",(m/s)(g/cm^3))))
par(new=TRUE)
plot(ecdf(Ip_like), col=("blue"),xlim=c(4000,␣
 ↪11200),lwd=3,xaxt='n',yaxt='n',ann=FALSE )
par(new=TRUE)
plot(ecdf(Xf[,1]), col=("black"),xlim=c(4000,␣
 ↪11200),lwd=3,xaxt='n',yaxt='n',ann=FALSE) #IFME
par(new=TRUE)
```

```
plot(ecdf(cop_frank_post_ran5F[,1]), col=("red"),xlim=c(4000,␣
 ↪11200),lwd=3,xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topleft", legend = c("Ip upscaled", "Ip priori","Ip posteriori"),␣
 ↪fill = c("blue","black","red"))
```



[110]:
```
plot(data_pseudo, pch=16, col=("red"),xlab=expression(paste("Acoustic impedance␣
 ↪",(m/s)(g/cm^3))),
     ylab=expression(paste("Total porosity ",phi[t], " (dec) ")),yaxt='n',main =␣
 ↪("Frank copula priori 15.04"))
par(new=TRUE)
```

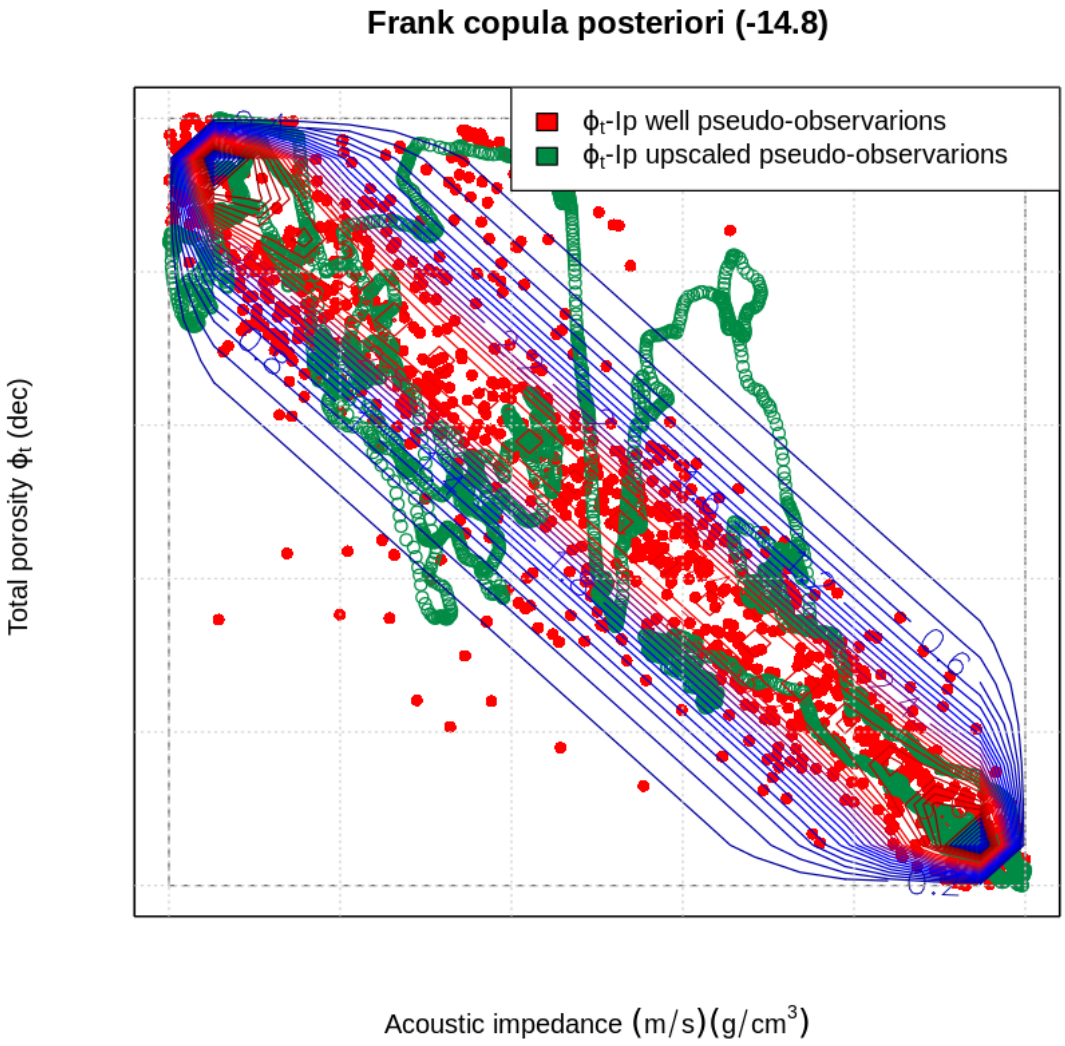
```
contour(frankCopula(-15.04), dCopula,n=20,nlevels=25, xlab=(""), ylab=(""),␣
↪,labcex=1.4, col=cols,xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend = c("Phit-Ip pseudo-observations"), fill = c(␣
↪"springgreen4"))
```



[124]:
```
plot(data_pseudo, pch=16, col=("red"), xlab=(""), ylab=(""),xaxt='n',yaxt='n',
     main = ("Frank copula posteriori (-14.8)"))
par(new=TRUE)
plot(pobs(Data_like), col=("springgreen4"), xlab=expression(paste("Acoustic␣
↪impedance ",(m/s)(g/cm^3))),
```

```
     ylab=expression(paste("Total porosity ",varphi[t], " (dec)␣
 ↪")),xaxt='n',yaxt='n',main = (""))
par(new=TRUE)
contour(frankCopula(param = theta_best_post5), dCopula,n=20,nlevels=25,␣
 ↪xlab=(""),
        ylab=("") ,labcex=1.4, col=cols,xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend = c(expression(paste(varphi[t], "-Ip well␣
 ↪pseudo-observarions")),
                                  expression(paste(varphi[t], "-Ip upscaled␣
 ↪pseudo-observarions"))),
       fill = c("red", "springgreen4"))

theta_best_post5
```

-15



6.5 Conditional joint simulation of total porosity with upscale acoustic impedance log.

Whith the posterior model is possible simulate the total porosity ϕ_t using the upscale acoustic impedance Ip as conditional data, for this case the posterior model was used to generate 36,960 pairs.

[146]:
```
Ip_like_df<-data.frame(Ip_like)
XFrF_Ip<-Ip_like_df[rep(seq_len(nrow(Ip_like_df)), each = 10), ]
```

[158]:
```
Ip_cond <- plnorm(XFrF_Ip,meanlog = mu_best_post5, sdlog = sigma_best_post5)

Fr_cond<-cCopula(cbind(Ip_cond,runif(length(Ip_cond))),
                 copula = frankCopula(theta_best_post5), inverse = TRUE)
```

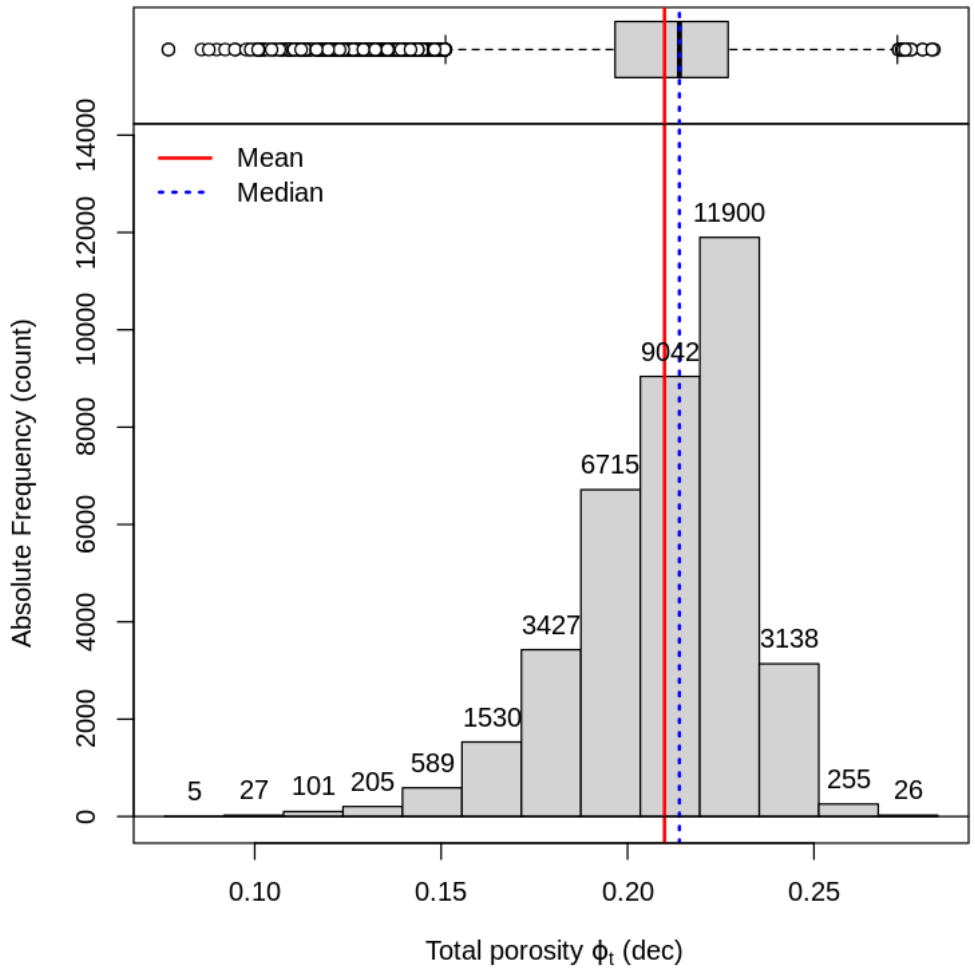
[159]:
```
XFrF_Phit <- qweibull(Fr_cond[,2],shape = alpha_best_post5, scale =␣
↪lambda_best_post5)
```

[160]:
```
XFrF_pair<-cbind(XFrF_Ip,XFrF_Phit)
write.csv(XFrF_pair, file = "Results/Frank_copula_40000_obs.csv")
```

[161]:
```
XFrF_Ip_Stat<-Estadisticas(XFrF_Ip)
XFrF_Phit_Stat<-Estadisticas(XFrF_Phit)
```

[162]:
```
HistBoxplot(XFrF_Phit,nbins = ns, mean = XFrF_Phit_Stat[5,2], median =␣
↪XFrF_Phit_Stat[4,2], main ="",
            xlab = expression(paste(" Total porosity ",varphi[t], " (dec)")),␣
↪ylab = "Absolute Frequency (count)",
            AbsFreq = TRUE, PercentFreq = FALSE )
XFrF_Phit_Stat
```

	Statistics <chr>	Values <dbl>
muestras	n	36960.0000
minimos	Minimum	0.0769
cuantiles1	1st. Quartile	0.1966
medianas	Median	0.2139
medias	Mean	0.2099
cuantiles3	3rd. Quartile	0.2270
maximos	Maximum	0.2822
rangos	Rank	0.2053
rangosInt	Interquartile Rank	0.0303
varianzas	Variance	0.0005
desvs	Standard Deviation	0.0231
CVs	Variation Coeff.	0.1102
simetrias	Skewness	-0.9216
curtosiss	Kurtosis	4.2166

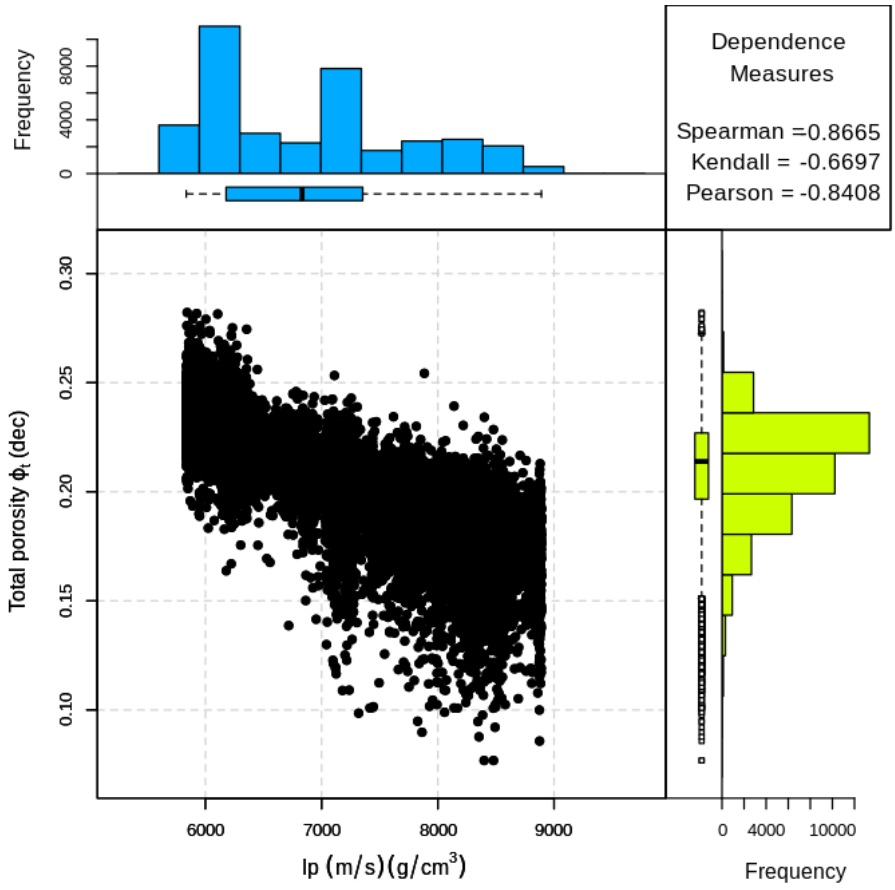


Comparing the statistics of ϕ_t simulated values, the minimum value is in the well log is 0.06, in the upscaled samples is 0.1655, but the simulated samples using the posterior model is 0.0769. The maximum in the well log is 0.29, in the upscaled samples is 0.2611 and the simulated samples using the posterior model is 0.2822. The values in median and mean in all cases are very close, with differences of 0.02.

[163]:
```
Comparative_post_Stat<-cbind(Ip_Stat, Ip_like_Stat[,2],XFrF_Ip_Stat[,2],
                             Phit_Stat[,2],Phit_like_Stat[,2],XFrF_Phit_Stat[,2])
#Comparative_post_Stat<-Val_Estadisticos(Comparative_post)
colnames(Comparative_post_Stat)<-c("Statistics","Ip","Ip upscaled",
                                   "Ip simulated","phit","phit upscaled", "phit␣
 ↪simulated")
Comparative_post_Stat
```

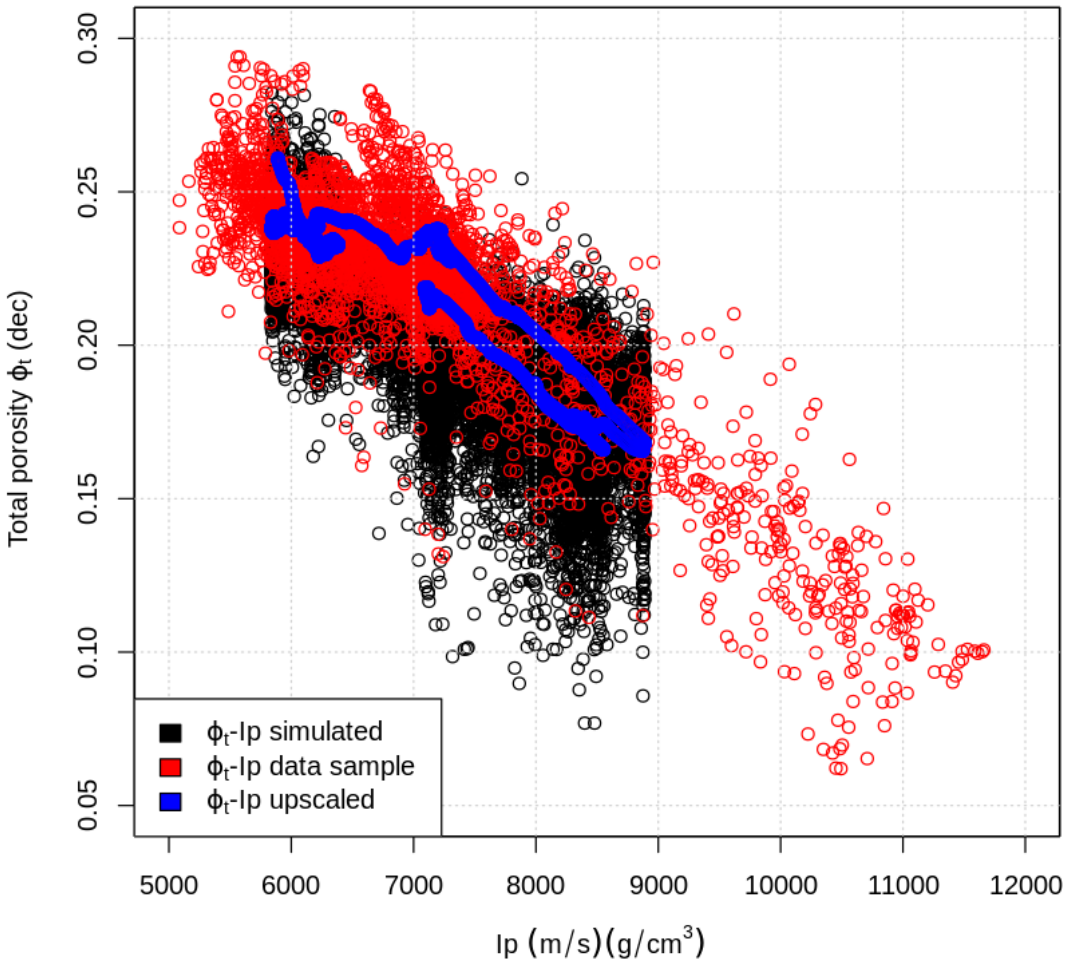
	Statistics <chr>	Ip <dbl>	Ip upscaled <dbl>	Ip simulated <dbl>	phit <dbl>	phit upscaled <dbl>	phit <dbl
muestras	n	3696.0000	3696.0000	36960.0000	3696.0000	3696.0000	3696
minimos	Minimum	5086.0072	5835.0969	5835.0969	0.0620	0.1655	0.076
cuantiles1	1st. Quartile	6157.0052	6178.1181	6178.1181	0.2147	0.2126	0.196
medianas	Median	6809.5574	6831.4653	6831.4653	0.2295	0.2320	0.213
medias	Mean	6968.2839	6892.8985	6892.8985	0.2220	0.2221	0.209
cuantiles3	3rd. Quartile	7321.6170	7353.4905	7353.4905	0.2414	0.2369	0.227
maximos	Maximum	11661.4642	8891.2124	8891.2124	0.2939	0.2611	0.282
rangos	Rank	6575.4570	3056.1155	3056.1155	0.2319	0.0956	0.205
rangosInt	Interquartile Rank	1164.6118	1175.3724	1175.3724	0.0267	0.0243	0.030
varianzas	Variance	1310302.9359	708798.2641	708625.6625	0.0011	0.0005	0.000
desvs	Standard Deviation	1144.6846	841.9016	841.7991	0.0338	0.0223	0.023
CVs	Variation Coeff.	0.1643	0.1221	0.1221	0.1521	0.1003	0.110
simetrias	Skewness	1.5610	0.5967	0.5969	-1.8102	-1.2139	-0.92
curtosiss	Kurtosis	5.7657	2.1898	2.1898	6.9970	3.2414	4.216

[164]:
```
ScatterPlot(XFrF_Ip , XFrF_Phit, ns,
            Xmin = XFrF_Ip_Stat[2,2], Xmax = XFrF_Ip_Stat[7,2],
            Ymin = XFrF_Phit_Stat[2,2],Ymax = XFrF_Phit_Stat[7,2],
            XLAB = expression(paste(" Ip ",(m/s)(g/cm^3))),
        YLAB = expression(paste(" Total porosity ",varphi[t], " (dec)")))
```



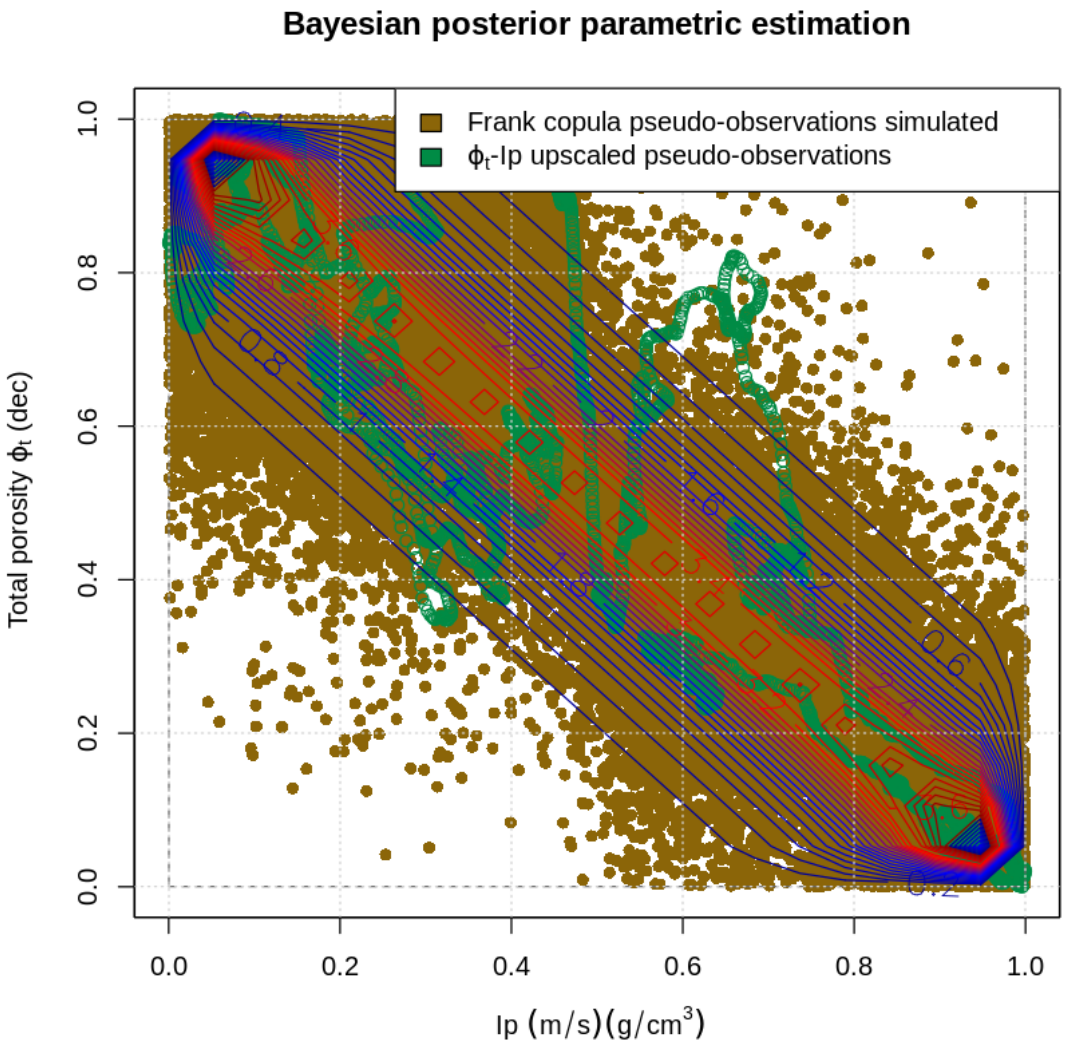
[165]:
```
plot(XFrF_Ip , XFrF_Phit,xlim=c(5000, 12000),ylim=c(0.05, 0.
↪3),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Ip , Phit, xlim=c(5000, 12000),ylim=c(0.05, 0.
↪3),col=("red"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Ip_like , Phit_like, xlab = expression(paste(" Ip ",(m/s)(g/cm^3))),
        ylab = expression(paste(" Total porosity ",varphi[t], "␣
↪(dec)")),xlim=c(5000, 12000),ylim=c(0.05, 0.3),
     col=("blue"))
grid()
legend(x = "bottomleft", legend = c(expression(paste(varphi[t],"-Ip simulated")),
```

```
                                        expression(paste(varphi[t],"-Ip data␣
↪sample")),
                                      expression(paste(varphi[t],"-Ip upscaled"))),
       fill = c("black", "red","blue"))
```



[166]:
```
plot(pobs(cbind(XFrF_Ip , XFrF_Phit)), pch=16,␣
 ↪col=("darkgoldenrod4"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(pobs(cbind(Ip_like , Phit_like)), xlab=(""), ylab=(""),
     main = ("Bayesian posterior parametric estimation"), col=("springgreen4"))
par(new=TRUE)
```

```
contour(frankCopula(theta_best_post5), dCopula,n=20,nlevels=25, xlab =␣
↪expression(paste(" Ip ",(m/s)(g/cm^3))),
    ylab = expression(paste(" Total porosity ",varphi[t], " (dec)")) ,
       labcex=1.4, col=cols,xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend = c("Frank copula pseudo-observations simulated",
                                  expression(paste(varphi[t],"-Ip upscaled␣
↪pseudo-observations"))),
      fill = c("darkgoldenrod4", "springgreen4"))
```



The dependence measures shows that all models have small differences in Pearson measure, but Spearman measure increases their dependence, this due to the amount of samples.

Dependence	Data sample	Upscaled data sample	Simulated Parametric
Pearson	-0.8563	-0.9063	-0.8408
Spearman	-0.7107	-0.8692	-0.8665
Kendall	-0.5335	-0.7010	-0.6697

After evaluate the simulated samples, the simulated annealing method is used to generate spatial distributions conditionated to acoustic impedance, the first example uses the acoustic impedance obtained from well log using the next variogram.

7 Total porosity spatial simulation

To obtain the total porosity realizations, the simulated annealing method is used. To compare the results 100 realizations are simulated.

[1]:
```
Data_like_FULL_100 <-  read.table(file='Results/100_sim_all_full.
 ↪txt',header=T,na.strings="-999.25")
#the select file is ip_Phie.csv
attach(Data_like_FULL_100)
```

```
The following objects are masked _by_ .GlobalEnv:

    Ip, Phit
```

[2]:
```
Ip_100<-Data_like_FULL_100$Ip
Phit_100<-Data_like_FULL_100$Phit

Ip_100_Stat<-Estadisticas(Ip_100)
Phit_100_Stat<-Estadisticas(Phit_100)
```

Comparing the statistics of total porosity ϕ_t obtained from well log and the total porosity ϕ_t simulated the differences in minimum and maximum is low, no more than 0.02, the same is observed in the median and mean. The variance is very close between them.

[3]:
```
Comparative_post_Stat<-cbind(Ip_Stat, Ip_like_Stat[,2],Ip_100_Stat[,2],
                             Phit_Stat[,2],Phit_like_Stat[,2],Phit_100_Stat[,2])
#Comparative_post_Stat<-Val_Estadisticos(Comparative_post)
colnames(Comparative_post_Stat)<-c("Statistics","Ip","Ip upscaled","Ip␣
 ↪simulated","phit","phit upscaled", "phit simulated")
Comparative_post_Stat
```

	Statistics <chr>	Ip <dbl>	Ip upscaled <dbl>	Ip simulated <dbl>	phit <dbl>	phit upscaled <dbl>	phit <dbl
muestras	n	3696.0000	3696.0000	369600.0000	3696.0000	3696.0000	3.696
minimos	Minimum	5086.0072	5835.0969	5835.0967	0.0620	0.1655	1.291
cuantiles1	1st. Quartile	6157.0052	6178.1181	6178.1183	0.2147	0.2126	2.117
medianas	Median	6809.5574	6831.4653	6831.4651	0.2295	0.2320	2.273
medias	Mean	6968.2839	6892.8985	6892.8985	0.2220	0.2221	2.235
cuantiles3	3rd. Quartile	7321.6170	7353.4905	7353.4907	0.2414	0.2369	2.382
maximos	Maximum	11661.4642	8891.2124	8891.2119	0.2939	0.2611	2.677
rangos	Rank	6575.4570	3056.1155	3056.1152	0.2319	0.0956	1.386
rangosInt	Interquartile Rank	1164.6118	1175.3724	1175.3724	0.0267	0.0243	2.650
varianzas	Variance	1310302.9359	708798.2641	708608.4035	0.0011	0.0005	4.000
desvs	Standard Deviation	1144.6846	841.9016	841.7888	0.0338	0.0223	1.920
CVs	Variation Coeff.	0.1643	0.1221	0.1221	0.1521	0.1003	8.610
simetrias	Skewness	1.5610	0.5967	0.5970	-1.8102	-1.2139	-8.03
curtosiss	Kurtosis	5.7657	2.1898	2.1898	6.9970	3.2414	3.338

[4]:
```
n=100 #numero de columnas
f=3696  #numero de filas
ff<-0
```

[5]:
```
sepf=matrix(0,n,2)
for (i in 1:n) {
  ff<-ff+f
 sepf[i,2]<-ff
}
sepf[,1]<-sepf[,2]-(f-1)
```

[6]:
```
Data_Phit_full=matrix(0,f,n)
for (i in 1:n) {
  Data_Phit_full[,i]<-Data_like_FULL_100[sepf[i,1]:sepf[i,2],5]
}
```

[7]:
```
XFrF_Phit_Stat<-Val_Estadisticos(Data_Phit_full)
XFrF_Phit_Stat
```

No_muestras	3696.00000	3696.00000	3696.00000	3696.00000	3696.00000	3696.00000	3696.0000
Minimo	0.14366	0.15124	0.15101	0.15489	0.15758	0.14366	0.15145
Cuartil_1er	0.21167	0.21164	0.21178	0.21158	0.21178	0.21172	0.21163
Mediana	0.22723	0.22738	0.22746	0.22736	0.22742	0.22744	0.22730
Media	0.22343	0.22344	0.22350	0.22349	0.22359	0.22351	0.22355
Cuartil_3er	0.23820	0.23814	0.23814	0.23821	0.23818	0.23813	0.23824
Maximo	0.26070	0.26342	0.26075	0.26556	0.26415	0.26486	0.26634
Rango	0.11704	0.11218	0.10974	0.11067	0.10657	0.12120	0.11489
Rango_Intercuartil	0.02653	0.02650	0.02636	0.02663	0.02640	0.02642	0.02661
Varianza	0.00037	0.00037	0.00037	0.00037	0.00037	0.00037	0.00036
Desv_Estandar	0.01934	0.01923	0.01919	0.01917	0.01913	0.01924	0.01907
Simetria	-0.84488	-0.79447	-0.80885	-0.75364	-0.74686	-0.78731	-0.74265
Curtosis	3.52449	3.24564	3.30442	3.13760	3.13060	3.26556	3.15520

[8]:
```
DF_Data_like_Phit<-data.frame(Data_Phit_full)

nc<-ncol(DF_Data_like_Phit)
nf<-nrow(DF_Data_like_Phit)
```

[9]:
```
P<-matrix(0, nc, 1)
```

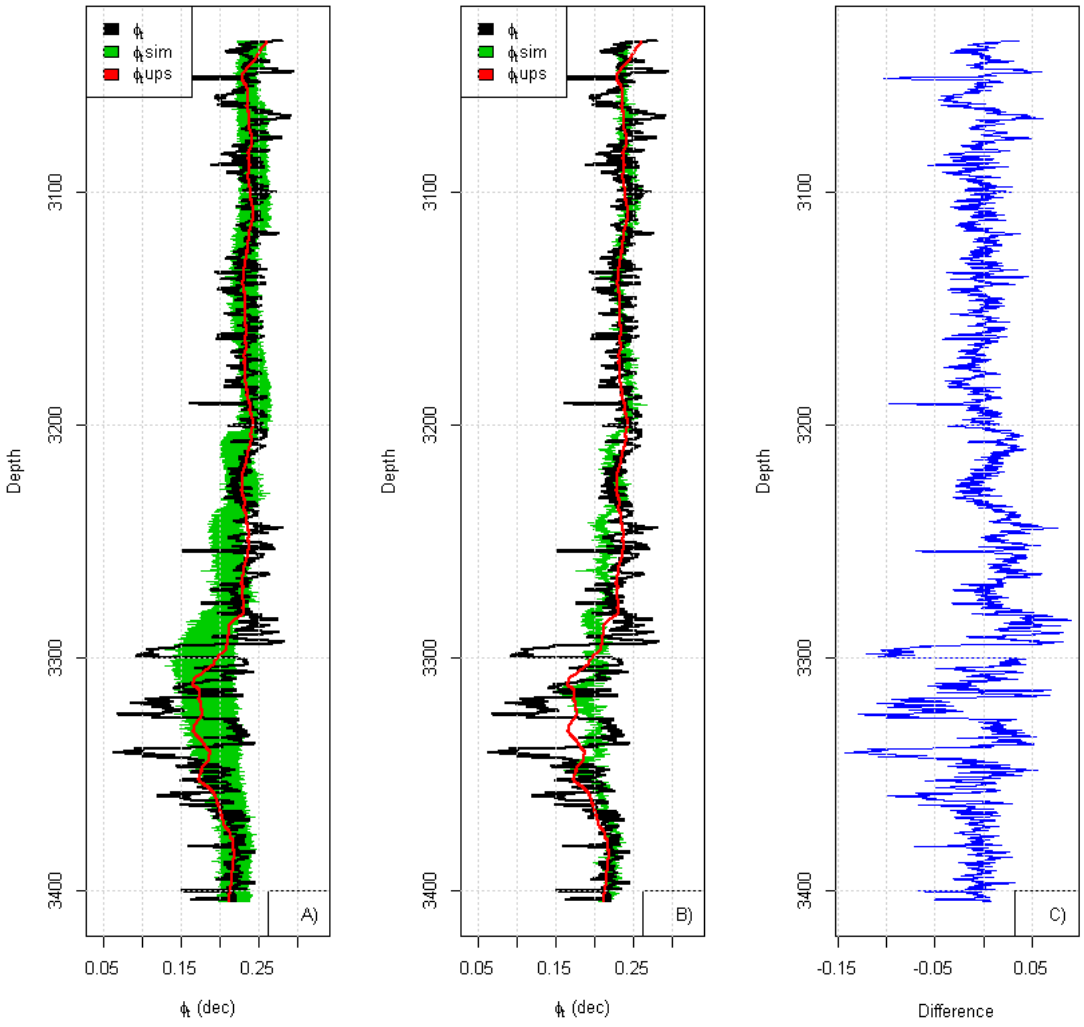
[10]:
```
for (i in 1:nc) {
   P[i,] <- (sum((Phit_like-DF_Data_like_Phit[,i])^2))/nf
}
```

[11]:
```
minP<-which.min(P)
minP
```

7

[6]:
```
#jpeg("Results/final/image8.jpeg", bg = "white", width = 2500, height = 2500,␣
↪res = 300)

par(mfrow = c(1,3))
for (i in 1:ncol(Data_Phit_full)) {
   plot(Data_Phit_full[,i],Depth,xlim=c(0.04, 0.33),ylim=rev(range(Depth)),type␣
↪= "l",lwd=1,col=("green3"),
        xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
}
plot(Phit,Depth,xlab = expression(paste(phi[t], " (dec)")),ylab="Depth",xlim=c(0.
↪04, 0.33),ylim=rev(range(Depth)),
     type = "l",lwd=2,col=("black"))
par(new=TRUE)
plot(Phit_like,Depth,xlab = expression(paste(phi[t], "␣
↪(dec)")),ylab="Depth",xlim=c(0.04, 0.33),
     ylim=rev(range(Depth)),
     type = "l",lwd=2,col=("red"))
```

```
legend(x = "topleft", legend =␣
 ↪c(expression(paste(phi[t])),expression(paste(phi[t], "sim")),
                                  expression(paste(phi[t], "ups"))),
       fill = c("black", "green3","red"))
legend(x="bottomright",legend="A)")
grid()

plot(Data_Phit_full[,minP],Depth,xlim=c(0.04, 0.33),ylim=rev(range(Depth)),type␣
 ↪= "l",
     lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Phit,Depth,xlab = expression(paste(phi[t], " (dec)")),ylab="Depth",xlim=c(0.
 ↪04, 0.33),ylim=rev(range(Depth)),
     type = "l",lwd=2,col=("black"))
par(new=TRUE)
plot(Phit_like,Depth,xlab = expression(paste(phi[t], "␣
 ↪(dec)")),ylab="Depth",xlim=c(0.04, 0.33),
     ylim=rev(range(Depth)),
     type = "l",lwd=2,col=("red"))
legend(x = "topleft", legend =␣
 ↪c(expression(paste(phi[t])),expression(paste(phi[t], "sim")),
                                  expression(paste(phi[t], "ups"))),
       fill = c("black", "green3","red"))
legend(x="bottomright",legend="B)")
grid()

plot((Phit-Data_Phit_full[,minP]),Depth,xlab="Difference",ylab="Depth",ylim=rev(range(Depth)),
     type = "l",lwd=1,col=("blue"))
legend(x="bottomright",legend="C)")
grid()

#dev.off()
```



the coverage of the 100 simulations is good until the 3200 meters, after this depth the simulations goes to left, this effect could be caused by the data in the interval 3300 to 3350, this zone has low porosity.

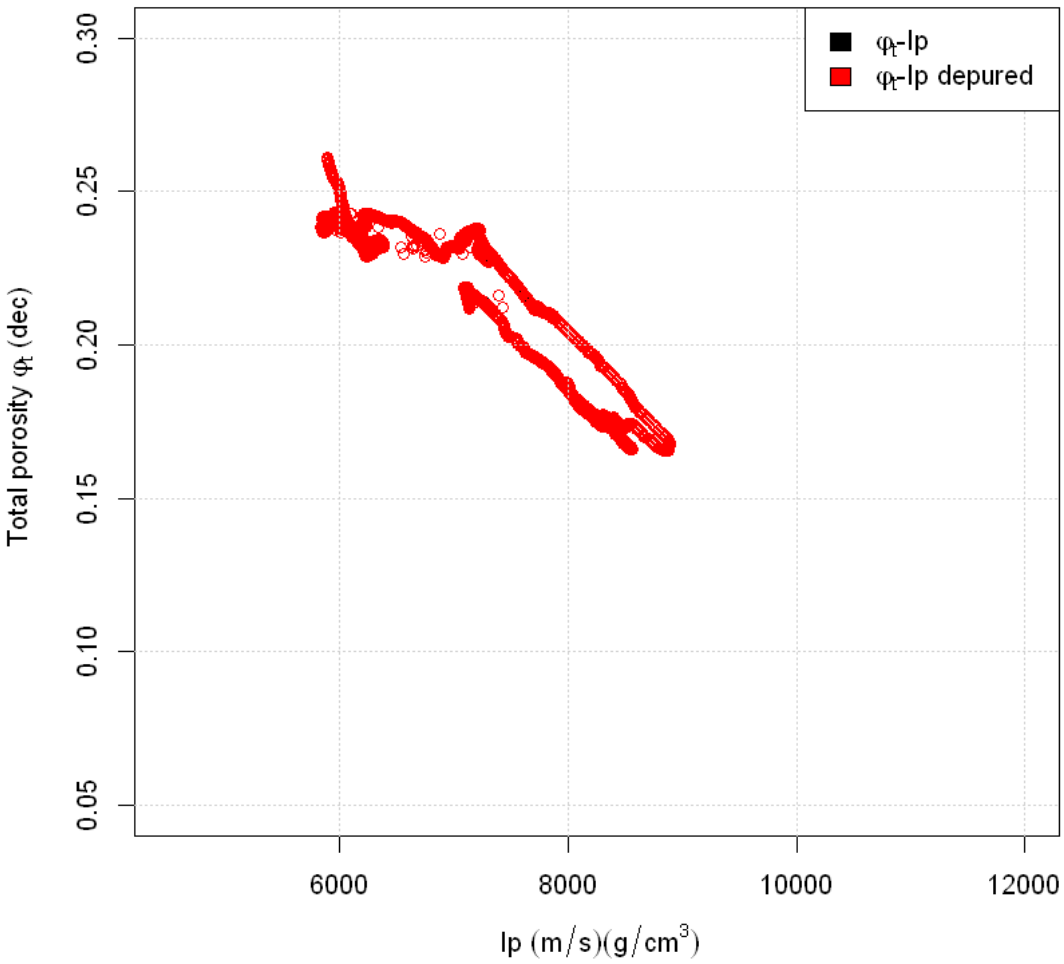
8 Comparison with deterministic parameter approach.

The results obtained with the bayesian copula model are good, valid for both scales. Now is time to compare the results if the bivariate copula model is estimated under deterministic approach. This example uses the same upscale data samples and the parametric functions for marginal and copula are estimated using Bayesian information criterion (BIC) and Akaike information criterion (AIC).

The same depuration process is applied in this approach.

[13]:
```
muestra_2<-as.data.frame(cbind(Ip_like, Phit_like))
muestra_depurada_2 <- depurar.xy(muestra_2)
Ip_like_d<-muestra_depurada_2[,1]
Phit_like_d<-muestra_depurada_2[,2]
```

[14]:
```
plot(muestra_2,xlab = expression(paste(" Ip ",(m/s)(g/cm^3))),
            ylab = expression(paste(" Total porosity ",varphi[t], "␣
 ↪(dec)")),xlim=c(4500, 12000),ylim=c(0.05, 0.3))
par(new=TRUE)
plot(muestra_depurada_2,xlim=c(4500, 12000),ylim=c(0.05, 0.
 ↪3),col=("red"),xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend =␣
 ↪c(expression(paste(varphi[t],"-Ip")),expression(paste(varphi[t],"-Ip␣
 ↪depured"))),
       fill = c("black", "red"))
```



[15]:
```
Ip_like_d_Stat<-Estadisticas(Ip_like_d)
Ip_like_d_Stat
```

	Statistics <chr>	Values <dbl>
muestras	n	3613.0000
minimos	Minimum	5835.0969
cuantiles1	1st. Quartile	6178.7922
medianas	Median	6836.2829
medias	Mean	6901.9772
cuantiles3	3rd. Quartile	7418.5621
maximos	Maximum	8891.2124
rangos	Rank	3056.1155
rangosInt	Interquartile Rank	1239.7700
varianzas	Variance	713239.8967
desvs	Standard Deviation	844.5353
CVs	Variation Coeff.	0.1224
simetrias	Skewness	0.5818
curtosiss	Kurtosis	2.1663

[16]:
```
Phit_like_d_Stat<-Estadisticas(Phit_like_d)
Phit_like_d_Stat
```

	Statistics <chr>	Values <dbl>
muestras	n	3613.0000
minimos	Minimum	0.1655
cuantiles1	1st. Quartile	0.2124
medianas	Median	0.2319
medias	Mean	0.2219
cuantiles3	3rd. Quartile	0.2368
maximos	Maximum	0.2611
rangos	Rank	0.0956
rangosInt	Interquartile Rank	0.0245
varianzas	Variance	0.0005
desvs	Standard Deviation	0.0224
CVs	Variation Coeff.	0.1010
simetrias	Skewness	-1.1878
curtosiss	Kurtosis	3.1691

8.0.1 Total porosity CDF deterministic modeling

As prior example, the total porosity uses gamma, beta, exponential, normal, logistic, lognormal and Weibull distribution funcion.

[20]:
```
fga_2 <- fitdist(Phit_like_d, "gamma", method="mle")
summary(fga_2)
```

```
Fitting of the distribution ' gamma ' by maximum likelihood
Parameters :
       estimate Std. Error
shape  89.36767   2.098672
```

```
rate  402.80599   9.485846
Loglikelihood:  8443.117   AIC:  -16882.23   BIC:  -16869.85
Correlation matrix:
          shape      rate
shape 1.0000000 0.9972038
rate  0.9972038 1.0000000
```

[21]:
```
fb_2 <- fitdist(Phit_like_d, "beta", method="mle")
summary(fb_2)
```

```
Fitting of the distribution ' beta ' by maximum likelihood
Parameters :
        estimate Std. Error
shape1  71.30153   1.673797
shape2 250.12276   5.886361
Loglikelihood:  8488.053   AIC:  -16972.11   BIC:  -16959.72
Correlation matrix:
          shape1    shape2
shape1 1.0000000 0.9955025
shape2 0.9955025 1.0000000
```

[22]:
```
fex_2 <- fitdist(Phit_like_d, "exp", method="mle")
summary(fex_2)
```

```
Fitting of the distribution ' exp ' by maximum likelihood
Parameters :
     estimate Std. Error
rate 4.507353 0.07498728
Loglikelihood:  1827.131   AIC:  -3652.261   BIC:  -3646.069
```

[23]:
```
fcb_2 <- fitdist(Phit_like_d, "norm", method="mle")
summary(fcb_2)
```

```
Fitting of the distribution ' norm ' by maximum likelihood
Parameters :
       estimate   Std. Error
mean 0.22185969 0.0003727499
sd   0.02240534 0.0002612152
Loglikelihood:  8597.196   AIC:  -17190.39   BIC:  -17178.01
Correlation matrix:
             mean           sd
mean 1.000000e+00 4.427781e-14
sd   4.427781e-14 1.000000e+00
```

[24]:
```
fgu_2 <- fitdist(Phit_like_d, "logis", method="mle")
summary(fgu_2)
```

```
Fitting of the distribution ' logis ' by maximum likelihood
Parameters :
           estimate   Std. Error
location 0.22579897 0.0003474766
scale    0.01208147 0.0001705655
Loglikelihood:  8654.399   AIC:  -17304.8   BIC:  -17292.41
Correlation matrix:
          location     scale
location  1.000000 -0.170157
scale    -0.170157  1.000000
```

[25]:
```
fln_2 <- fitdist(Phit_like_d, "lnorm", method="mle")
summary(fln_2)
```

```
Fitting of the distribution ' lnorm ' by maximum likelihood
Parameters :
          estimate  Std. Error
meanlog -1.5113156 0.001804200
sdlog    0.1084473 0.001275274
Loglikelihood:  8360.006   AIC:  -16716.01   BIC:  -16703.63
Correlation matrix:
        meanlog sdlog
meanlog       1     0
sdlog         0     1
```

[26]:
```
fw_2 <- fitdist(Phit_like_d, "weibull", method="mle")
summary(fw_2)
```

```
Fitting of the distribution ' weibull ' by maximum likelihood
Parameters :
        estimate   Std. Error
shape 14.7611041 0.2123744898
scale  0.2308911 0.0002703754
Loglikelihood:  9146.21   AIC:  -18288.42   BIC:  -18276.04
Correlation matrix:
          shape     scale
shape 1.0000000 0.2749465
scale 0.2749465 1.0000000
```

[27]:
```
best_fit_Phit_like_2 <- gofstat(list(fw_2,fgu_2,fcb_2,fb_2,fga_2,fln_2))
print("Goodness of fit results for density Phit_like")
best_fit_Phit_like_2
```

```
[1] "Goodness of fit results for density Phit_like"

Goodness-of-fit statistics
                             1-mle-weibull 2-mle-logis  3-mle-norm  4-mle-beta
```

```
Kolmogorov-Smirnov statistic     0.2311558   0.2099202   0.2702872   0.2755632
Cramer-von Mises statistic      37.1829435  36.7910685  53.3796580  56.5716932
Anderson-Darling statistic     217.7962219 248.6766036 291.0841653 308.0157889
                            5-mle-gamma 6-mle-lnorm
Kolmogorov-Smirnov statistic   0.2773456   0.2805854
Cramer-von Mises statistic    57.7403327  59.9434611
Anderson-Darling statistic   314.3729197 326.1119367

Goodness-of-fit criteria
                               1-mle-weibull 2-mle-logis 3-mle-norm 4-mle-beta
Akaike's Information Criterion     -18288.42   -17304.80  -17190.39  -16972.11
Bayesian Information Criterion     -18276.04   -17292.41  -17178.01  -16959.72
                               5-mle-gamma 6-mle-lnorm
Akaike's Information Criterion   -16882.23   -16716.01
Bayesian Information Criterion   -16869.85   -16703.63
```

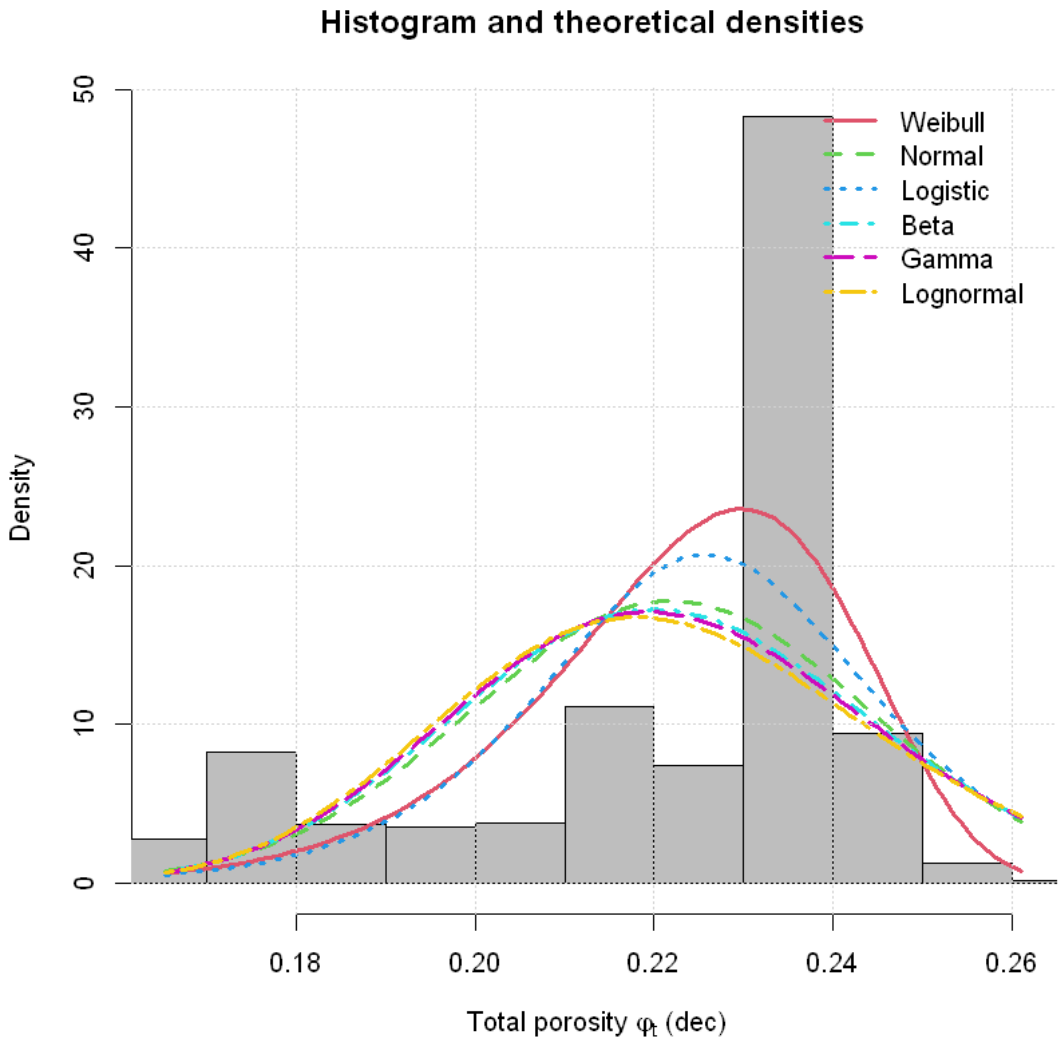
The best total porosity fit function is Weibull, is the same function selected in well log scale.

Function	Parameter	Error	Parameter	Error	Likelihood	AIC	BIC
Weibull	$\alpha =$ 14.7611041	0.2123744898	$\lambda =$ 0.2308911	0.000270974746	9216.21	-18288.42	-18276.04
Logistic	$\mu =$ 0.22579897	0.000347376	$\beta =$ 0.01208147	0.000170836	8654.399	-17304.8	-17292.41
Normal	$\mu =$ 0.22185969	0.000372749	$\sigma =$ 0.02240534	0.000261859	8597.196	-17190.39	-17178.01
Gamma	$k =$ 89.36767	2.098706	$\beta =$ 402.80599	9.485997	8443.117	-16882.23	-16869.85
Lognormal	$Log\mu =$ -1.5113156	0.001804206	$Log\sigma =$ 0.1084473	0.001275876	8360.006	-16716.01	-16703.63

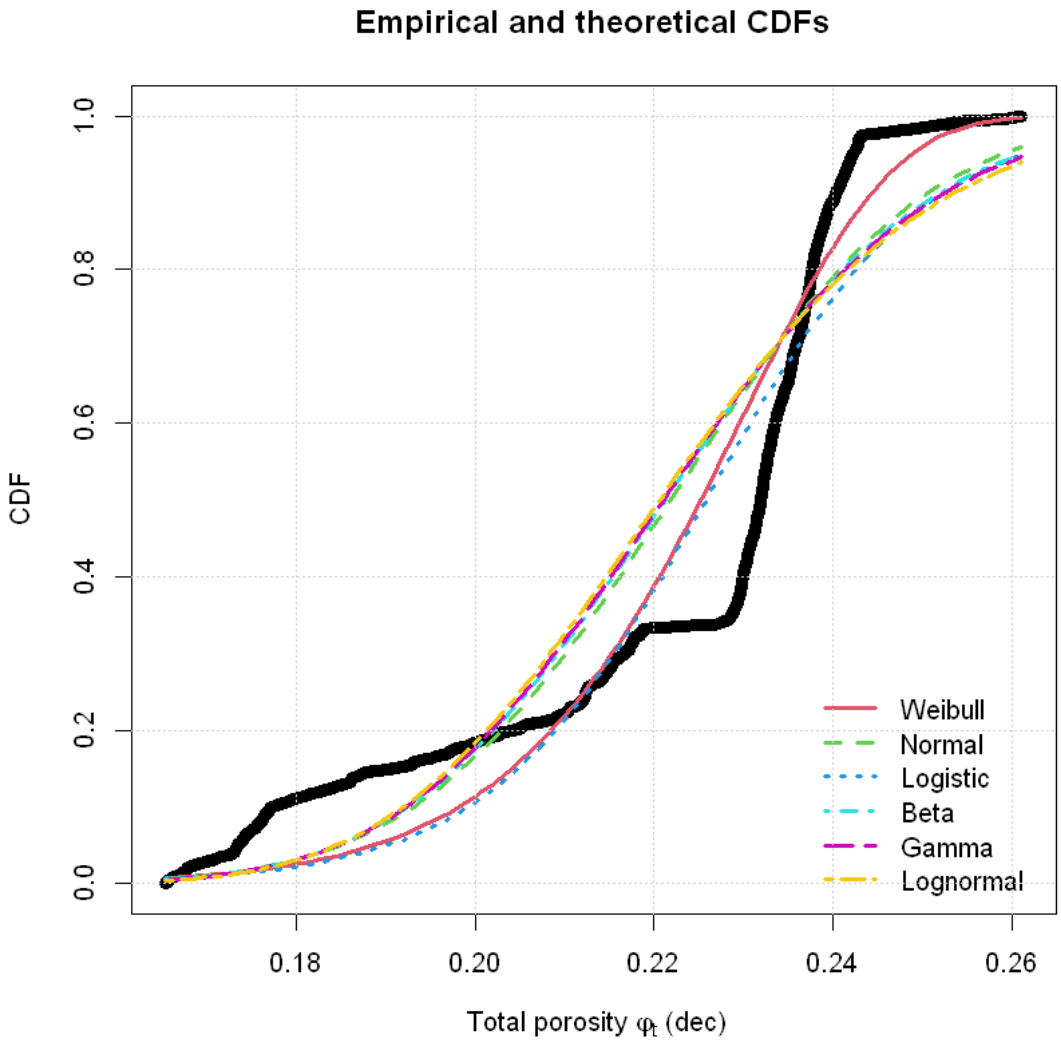
Comparing with the prior, posterior and deterministic parameter values, the patameter values α are different prior and posterior values are close, but deterministic value is almost double. Parameter λ is close in all cases

Function	Prior value	Posterior value	Deterministic value
Weibull (α)	5.1646292	8.96	14.7611041
Weibull (λ)	0.2244496	0.227	0.2308911

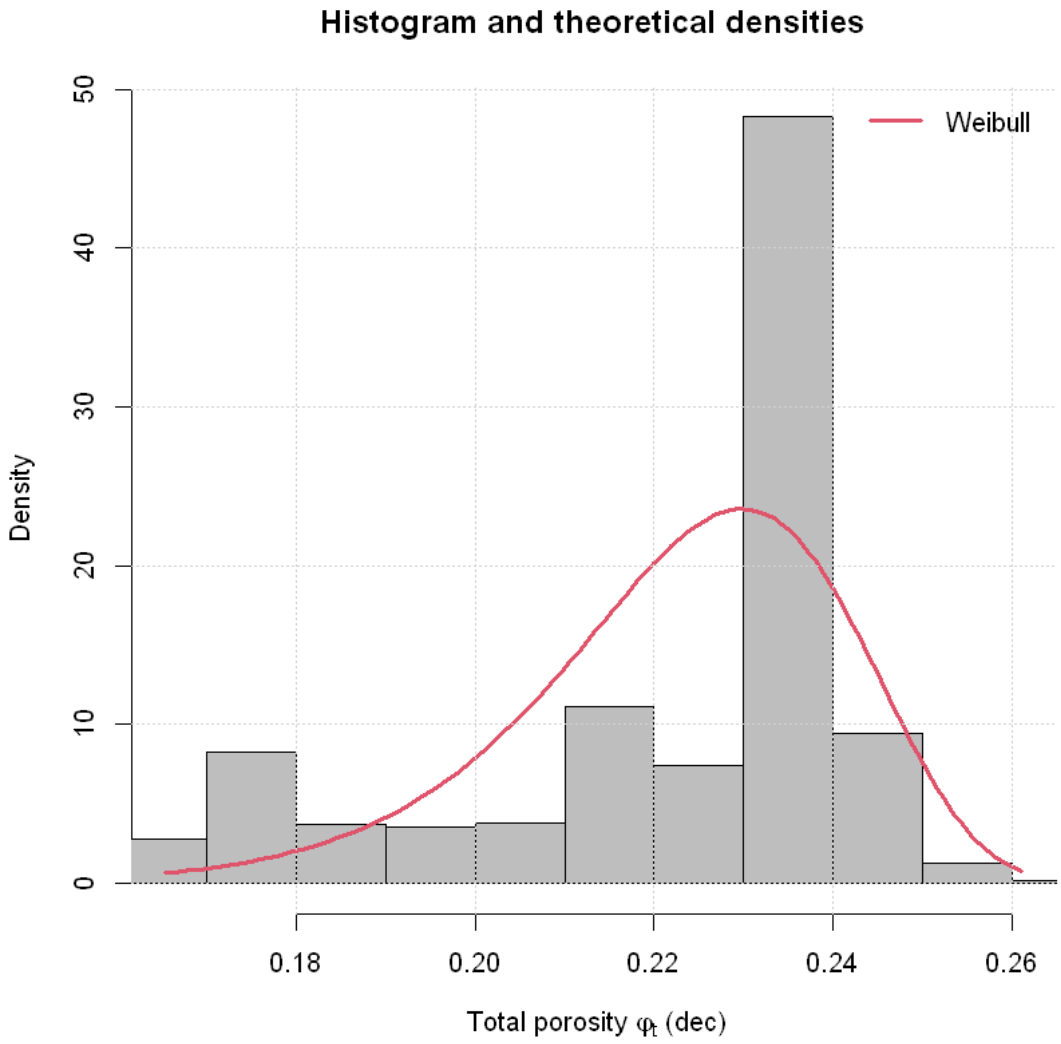
[28]:
```
plot.legend <- c("Weibull","Normal","Logistic","Beta","Gamma","Lognormal")
denscomp(list(fw_2,fcb_2,fgu_2,fb_2,fga_2,fln_2),fitlwd=3,datacol = "gray",
         xlab = expression(paste(" Total porosity ",varphi[t], " (dec)")),
         legendtext = plot.legend)
grid()
```



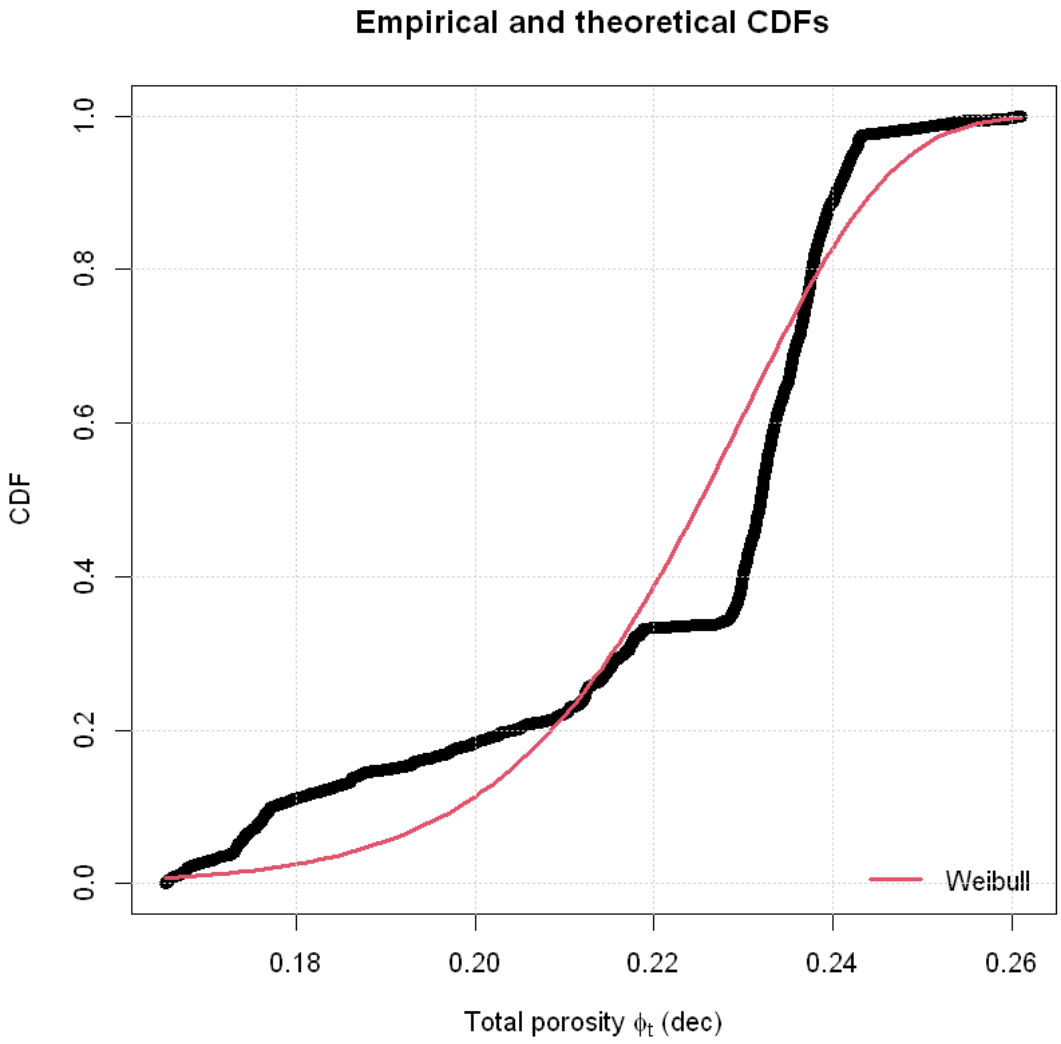
[29]:
```
cdfcomp(list(fw_2,fcb_2,fgu_2,fb_2,fga_2,fln_2),fitlwd=3,xlab =␣
↪expression(paste(" Total porosity ",varphi[t], " (dec)")),  legendtext = plot.
↪legend)
grid()
```



[30]:
```
denscomp(list(fw_2),xlab = expression(paste(" Total porosity ",varphi[t], "␣
↪(dec)")),
         datacol = "gray",fitlwd=3 , legendtext = "Weibull")
grid()
```



[3]:
```
cdfcomp(list(fw_2),xlab = expression(paste("Total porosity ",phi[t], "␣
↪(dec)")),fitlwd=3,
        legendtext = "Weibull")
grid()
```



8.0.2 Acoustic impedance CDF deterministic modeling

The parametric function tested are gamma, normal, lognormal, logistic and Weibull.

[32]:
```
fga1_2 <- fitdist(Ip_like_d, "gamma", method="mle")
summary(fga1_2)
```

```
Fitting of the distribution ' gamma ' by maximum likelihood
Parameters :
         estimate   Std. Error
shape 69.24375599 0.9617606290
rate   0.01003258 0.0001384462
Loglikelihood:  -29391.17   AIC:  58786.34   BIC:  58798.73
Correlation matrix:
```

```
          shape      rate
shape 1.0000000 0.9896614
rate  0.9896614 1.0000000
```

[33]:
```
fcb1_2 <- fitdist(Ip_like_d, "norm", method="mle")
summary(fcb1_2)
```

```
Fitting of the distribution ' norm ' by maximum likelihood
Parameters :
      estimate Std. Error
mean 6901.9772  14.047349
sd    844.4184   9.933867
Loglikelihood:  -29473.36   AIC:  58950.72   BIC:  58963.11
Correlation matrix:
     mean sd
mean    1  0
sd      0  1
```

[34]:
```
fgu1_2 <- fitdist(Ip_like_d, "logis", method="mle")
summary(fgu1_2)
```

```
Fitting of the distribution ' logis ' by maximum likelihood
Parameters :
          estimate Std. Error
location 6832.1271  14.850286
scale     501.5817   6.799732
Loglikelihood:  -29589.6   AIC:  59183.2   BIC:  59195.58
Correlation matrix:
          location     scale
location 1.0000000 0.0750662
scale    0.0750662 1.0000000
```

[35]:
```
fln1_2 <- fitdist(Ip_like_d, "lnorm", method="mle")
summary(fln1_2)
```

```
Fitting of the distribution ' lnorm ' by maximum likelihood
Parameters :
         estimate  Std. Error
meanlog 8.8323252 0.001985647
sdlog   0.1193537 0.001403621
Loglikelihood:  -29357.79   AIC:  58719.59   BIC:  58731.97
Correlation matrix:
              meanlog         sdlog
meanlog  1.000000e+00 -1.267424e-12
sdlog   -1.267424e-12  1.000000e+00
```

[36]:
```
fw1_2 <- fitdist(Ip_like_d, "weibull", method="mle")
summary(fw1_2)
```

```
Fitting of the distribution ' weibull ' by maximum likelihood
Parameters :
         estimate Std. Error
shape    8.348242  0.1030756
scale 7289.554866 15.4329433
Loglikelihood:  -29721.94   AIC:  59447.88   BIC:  59460.27
Correlation matrix:
          shape     scale
shape 1.0000000 0.3370412
scale 0.3370412 1.0000000
```

[37]:
```
best_fit_Ip_like_2 <- gofstat(list(fln1_2,fga1_2,fgu1_2,fcb1_2,fw1_2))
print("Goodness of fit results for Ip_like")
best_fit_Ip_like_2
```

```
[1] "Goodness of fit results for Ip_like"
```

```
Goodness-of-fit statistics
                             1-mle-lnorm 2-mle-gamma 3-mle-logis  4-mle-norm
Kolmogorov-Smirnov statistic    0.171736   0.1737814   0.1571515   0.1772675
Cramer-von Mises statistic     16.285163  16.6872924  15.5741111  17.8352280
Anderson-Darling statistic    100.851506 104.3561017 106.3598848 113.1408436
                             5-mle-weibull
Kolmogorov-Smirnov statistic     0.1657294
Cramer-von Mises statistic      20.3946082
Anderson-Darling statistic     129.3356849

Goodness-of-fit criteria
                               1-mle-lnorm 2-mle-gamma 3-mle-logis 4-mle-norm
Akaike's Information Criterion    58719.59    58786.34    59183.20   58950.72
Bayesian Information Criterion    58731.97    58798.73    59195.58   58963.11
                               5-mle-weibull
Akaike's Information Criterion       59447.88
Bayesian Information Criterion       59460.27
```

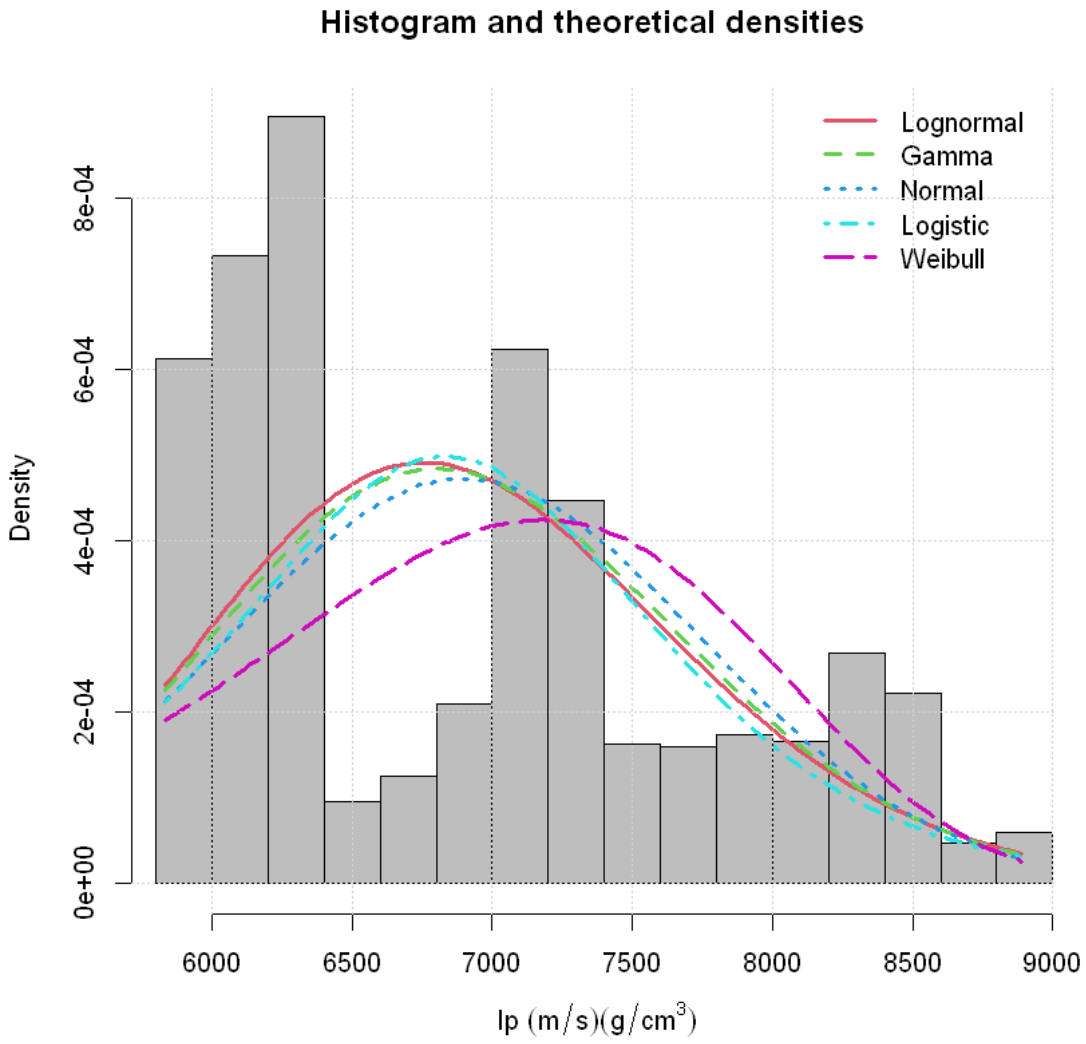
Based on fit results, Lognormal function is the best option, comparing with prior, posterior and deterministic values, parameter $\log \mu$ is close beneath them. Meanwhile, parameter $\log \mu$ is similar in prior and posterior, but deterministic value is different, 0.06, which is near to 30%

Function	Parameter	Error	Parameter	Error	Likelihood	AIC	BIC
Lognormal	$Log\mu =$ 8.8323252	0.0019856447	$Log\sigma =$ 0.1193537	0.0014036291	29357.79	58719.59	58731.97
Gamma	$k =$ 69.24375599	0.9617618428	$\theta =$ 0.01003258	0.0001384263	29391.17	58786.34	58798.73

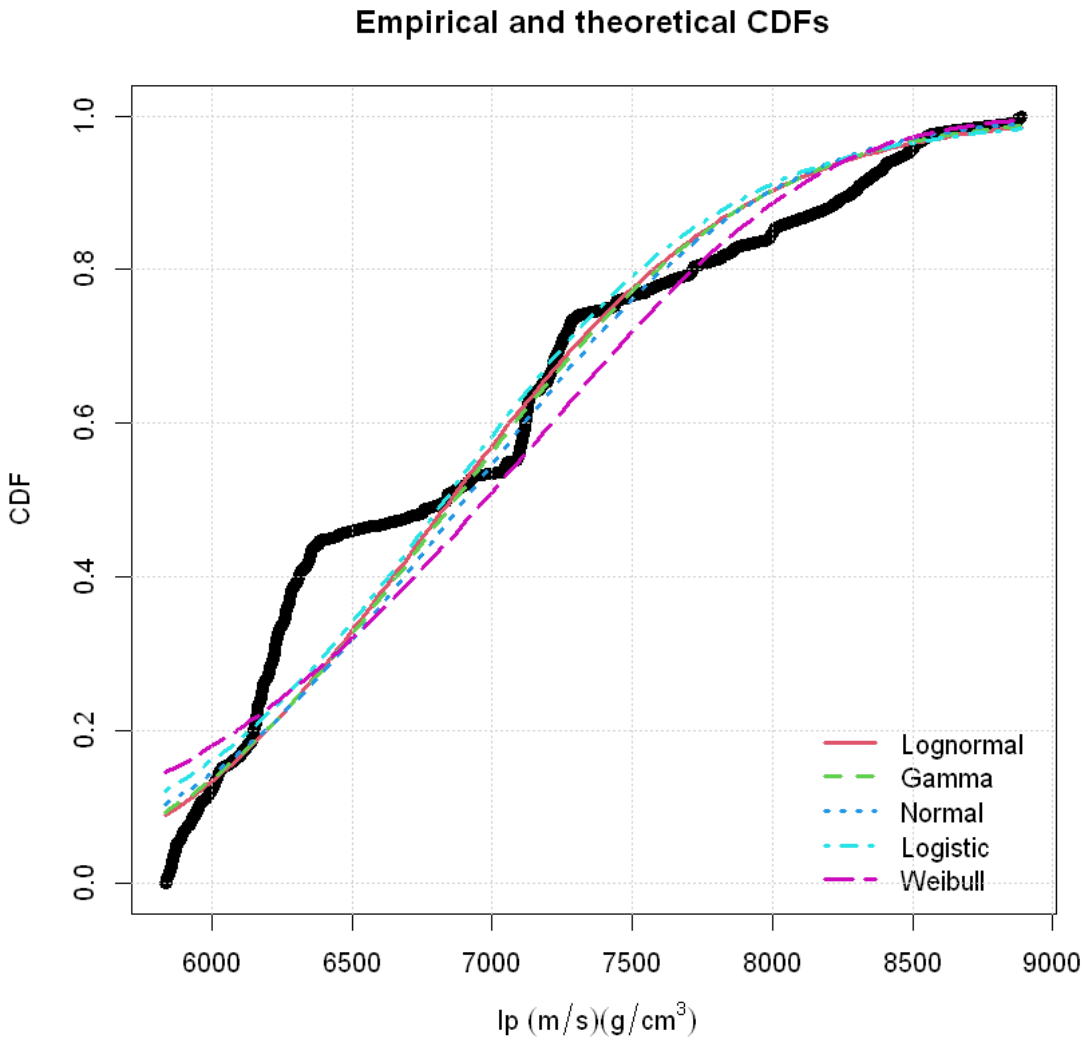
Function	Parameter	Error	Parameter	Error	Likelihood	AIC	BIC
Normal	$\mu =$ 6901.9772	14.047349	$\sigma =$ 844.4184	9.933867	-29473.36	58950.72	58963.11
Logistic	$\mu =$ 6832.1271	14.850286	$\beta =$ 501.5817	6.799732	-29589.6	59183.2	59195.58
Weibull	$\alpha =$ 8.348242	0.1030750	$\lambda =$ 7289.554866	15.43294	-29721.94	59447.88	59460.27

Function	Prior value	Posterior value	Deterministic value
Lognormal (μ)	8.905543	8.79	8.8323
Lognormal (σ)	0.191057	0.188	0.1193

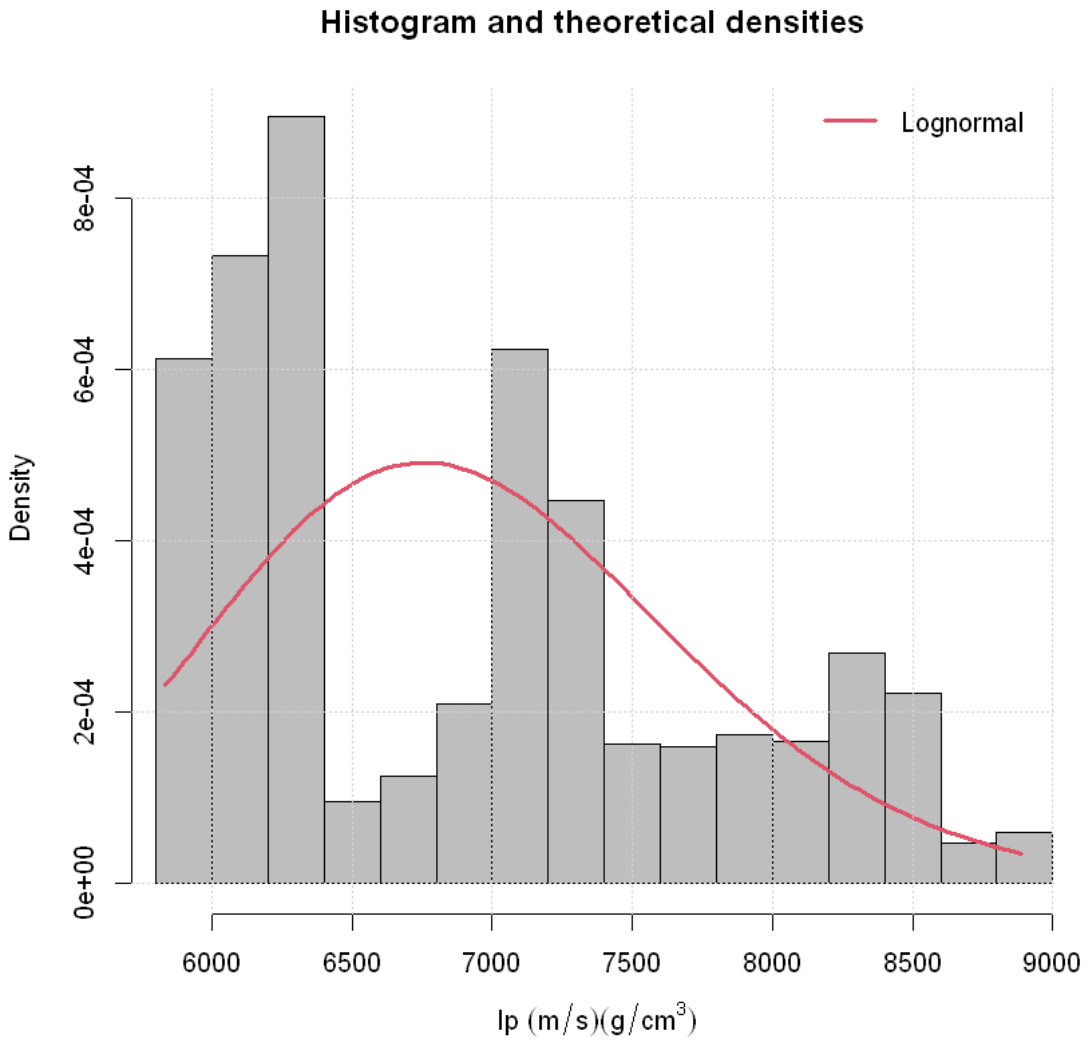
[38]:
```
plot.legend <- c("Lognormal","Gamma", "Normal", "Logistic",    "Weibull")
denscomp(list(fln1_2,fga1_2,fcb1_2,fgu1_2,fw1_2),datacol = "gray",
         xlab = expression(paste(" Ip ",(m/s)(g/cm^3))),fitlwd=3 , legendtext =␣
 ↪plot.legend)
grid()
```



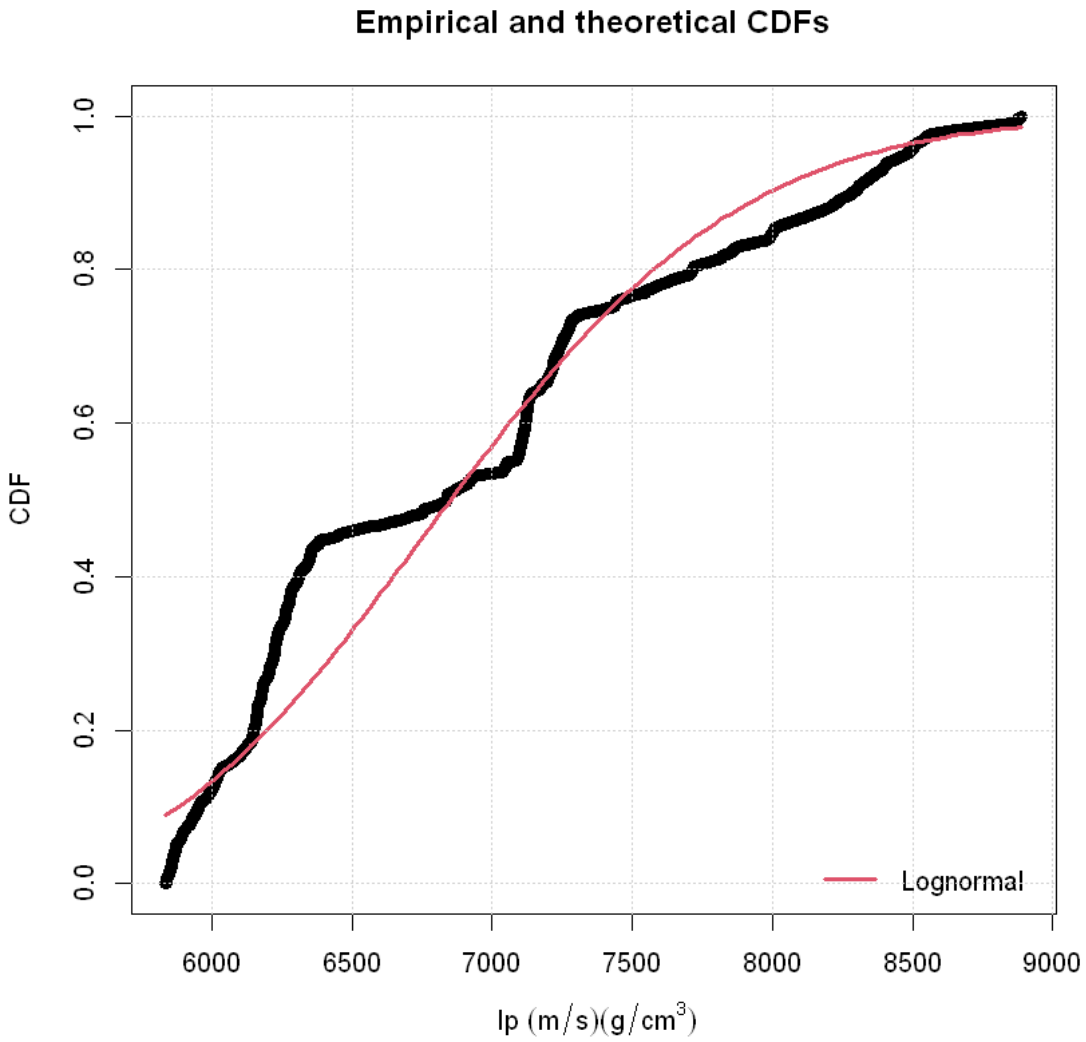
[39]:
```
cdfcomp(list(fln1_2,fga1_2,fcb1_2,fgu1_2,fw1_2),xlab = expression(paste(" Ip␣
↪",(m/s)(g/cm^3))),
        fitlwd=3,  legendtext = plot.legend)
grid()
```



[40]:
```
plot.legend <- c("Lognormal")
denscomp(list(fln1_2),datacol = "gray",xlab = expression(paste(" Ip ",(m/s)(g/
 ↪cm^3))),fitlwd=3 ,
         legendtext = plot.legend)
grid()
```



[4]:
```
cdfcomp(list(fln1_2),xlab = expression(paste(" Ip ",(m/s)(g/cm^3))),fitlwd=3, ␣
↪legendtext = plot.legend)
grid()
```



8.1 Copula parametric deterministic modeling

The copula selection consider Gumbel and Frank copula, the best option is Frank copula with parameter θ with value -7.003, this value is really diferent. the prior and posterior value is almost double tan deterministic value estimated although the measure of dependence is almost the same.

parameters	Prior value	Posterior value	Deterministic
Frank copula (θ)	-15.04	-14.8	-10.78

[42]:
```
data_pseudo_2=pobs(muestra_depurada_2)
n_2=3490
```

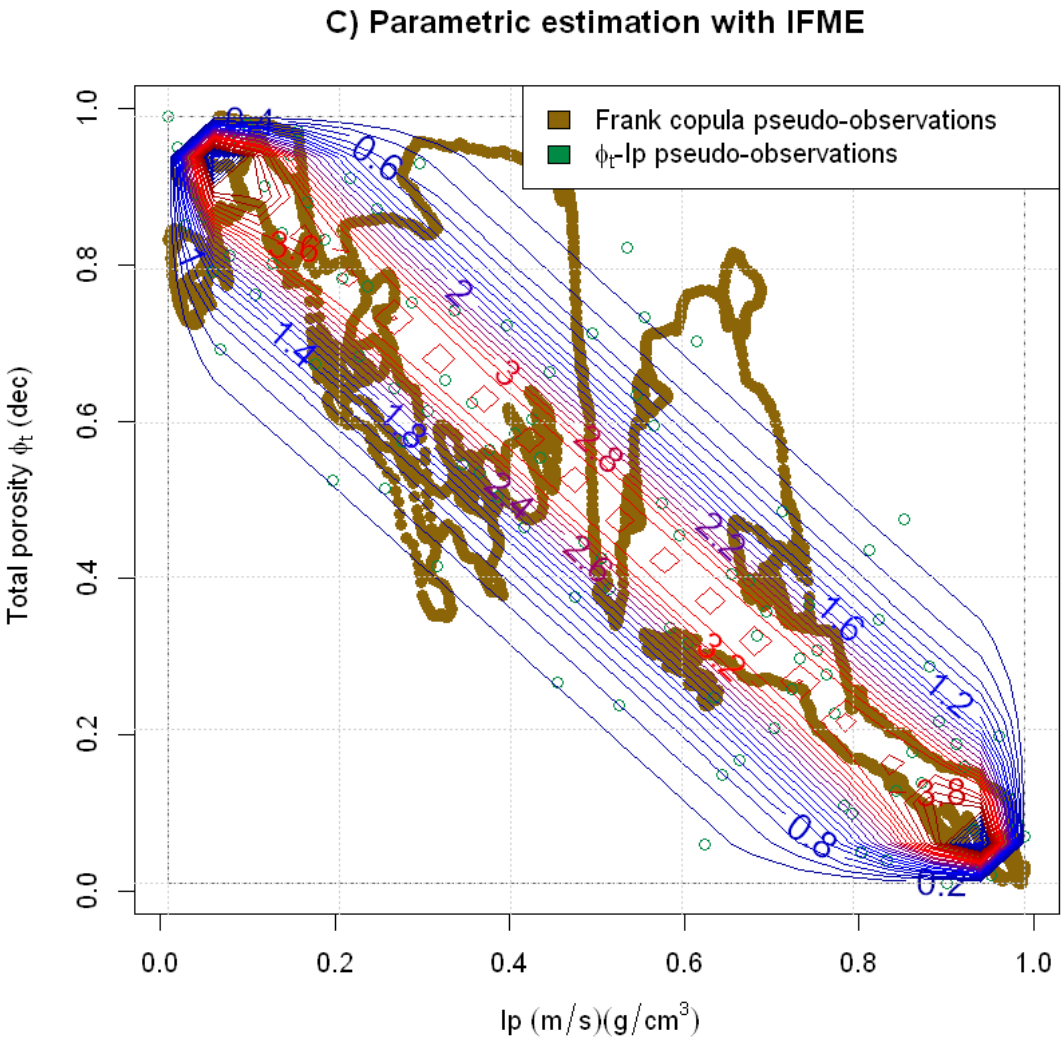
[43]:
```
ifme_frank_2 <- fitCopula(frankCopula(), data = data_pseudo_2, method = "ml")
summary(ifme_frank_2)
```

```
Call: fitCopula(frankCopula(), data = data_pseudo_2, ... = pairlist(method = "ml"))
Fit based on "maximum likelihood" and 3613 2-dimensional observations.
Frank copula, dim. d = 2
      Estimate Std. Error
alpha   -10.78      0.186
The maximized loglikelihood is 2482
Optimization converged
Number of loglikelihood evaluations:
function gradient
       6        6
```

[44]:
```
fc_2 <- frankCopula(ifme_frank_2@copula@parameters)

mfc_2 <- mvdc(fc_2, margins = c("lnorm", "weibull"),
            paramMargins = list(list(meanlog = 8.8323, sdlog = 0.1193),
                                list(shape = 14.7611, scale = 0.2308)))
```

[45]:
```
Xf_2 <- rMvdc(n, mvdc = mfc_2)

Xf_2_stat <- Val_Estadisticos(Xf_2[])

Xfpobs_2=pobs(Xf_2)
```

[8]:
```
plot(pobs(cbind(Ip_like , Phit_like)), pch=16,␣
↪col=("darkgoldenrod4"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Xfpobs_2, xlab=(""), ylab=(""), main = ("C) Parametric estimation with␣
↪IFME"), col=("springgreen4"))
par(new=TRUE)
contour(fc, dCopula,n=20,nlevels=25, xlab = expression(paste(" Ip ",(m/s)(g/
↪cm^3))),
    ylab = expression(paste(" Total porosity ",phi[t], " (dec)")) ,labcex=1.4,␣
↪col=cols,xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend = c("Frank copula pseudo-observations",
                                  expression(paste(phi[t],"-Ip␣
↪pseudo-observations"))), fill = c("darkgoldenrod4", "springgreen4"))
```



8.2 Conditional joint simulation of total porosity with acoustic impedance using deterministic approach model.

With the joint distribution function estimated with deterministic approah, 36,960 samples are simulated. These samples are condicionated with the acoustic impedance.

[81]:
```
Ip_like_df<-data.frame(Ip_like)
XFrF_Ip_2<-Ip_like_df[rep(seq_len(nrow(Ip_like_df)), each = 10), ]
```

[82]:
```
Ip_cond_2 <- plnorm(XFrF_Ip_2,meanlog = 8.8323, sdlog = 0.1193)
```

[83]:
```

```

```
Fr_cond_2<-cCopula(cbind(Ip_cond_2,runif(length(Ip_cond_2))),
                   copula = frankCopula(ifme_frank_2@copula@parameters), inverse =␣
↪TRUE)
```

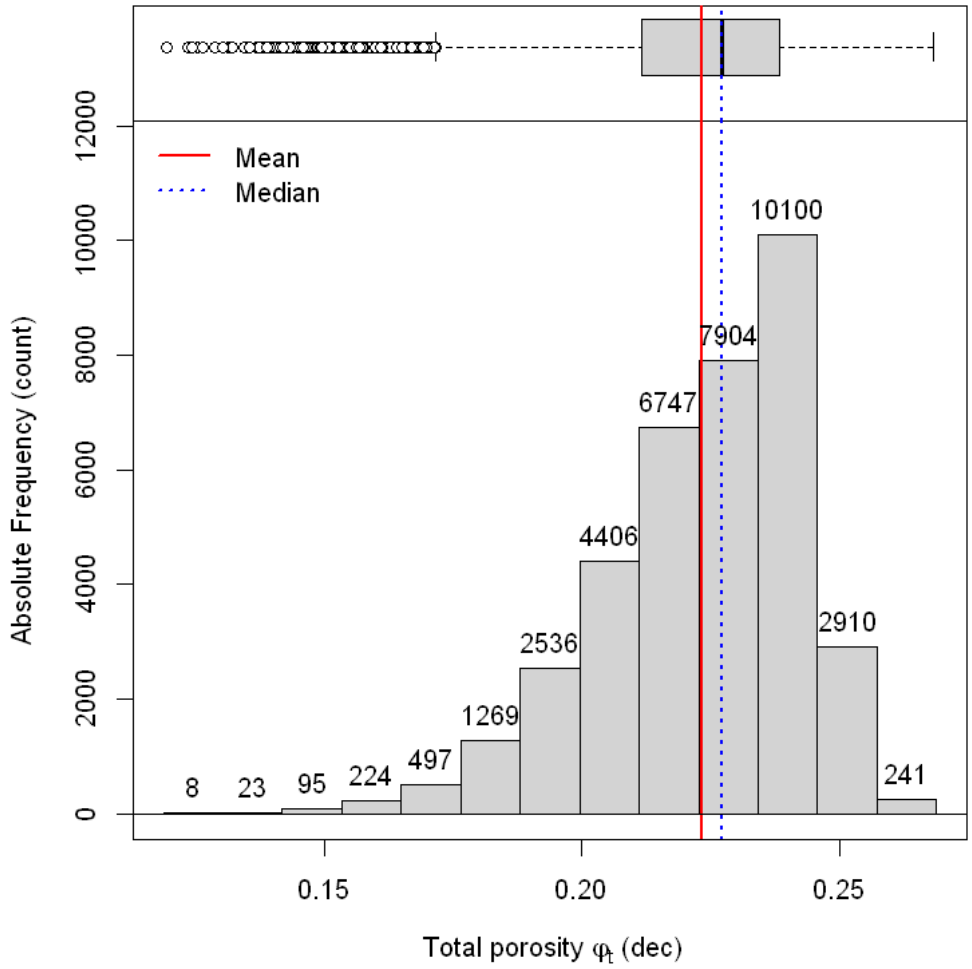
[84]:
```
XFrF_Phit_2 <- qweibull(Fr_cond_2[,2],shape = 14.7611, scale = 0.2308)
```

[85]:
```
XFrF_pair_2<-cbind(XFrF_Ip,XFrF_Phit_2)
write.csv(XFrF_pair_2, file = "Results/Frank_copula_40000_obs_det.csv")
```

[86]:
```
XFrF_Ip_2_Stat<-Estadisticas(XFrF_Ip_2)
XFrF_Phit_2_Stat<-Estadisticas(XFrF_Phit_2)
```

[87]:
```
HistBoxplot(XFrF_Phit_2,nbins = ns, mean = XFrF_Phit_2_Stat[5,2], median =␣
↪XFrF_Phit_2_Stat[4,2], main ="",
            xlab = expression(paste(" Total porosity ",varphi[t], " (dec)")),␣
↪ylab = "Absolute Frequency (count)",
            AbsFreq = TRUE, PercentFreq = FALSE )
XFrF_Phit_2_Stat
```

	Statistics <chr>	Values <dbl>
muestras	n	36960.0000
minimos	Minimum	0.1196
cuantiles1	1st. Quartile	0.2115
medianas	Median	0.2270
medias	Mean	0.2233
cuantiles3	3rd. Quartile	0.2382
maximos	Maximum	0.2680
rangos	Rank	0.1484
rangosInt	Interquartile Rank	0.0267
varianzas	Variance	0.0004
desvs	Standard Deviation	0.0197
CVs	Variation Coeff.	0.0883
simetrias	Skewness	-0.8752
curtosiss	Kurtosis	3.7571

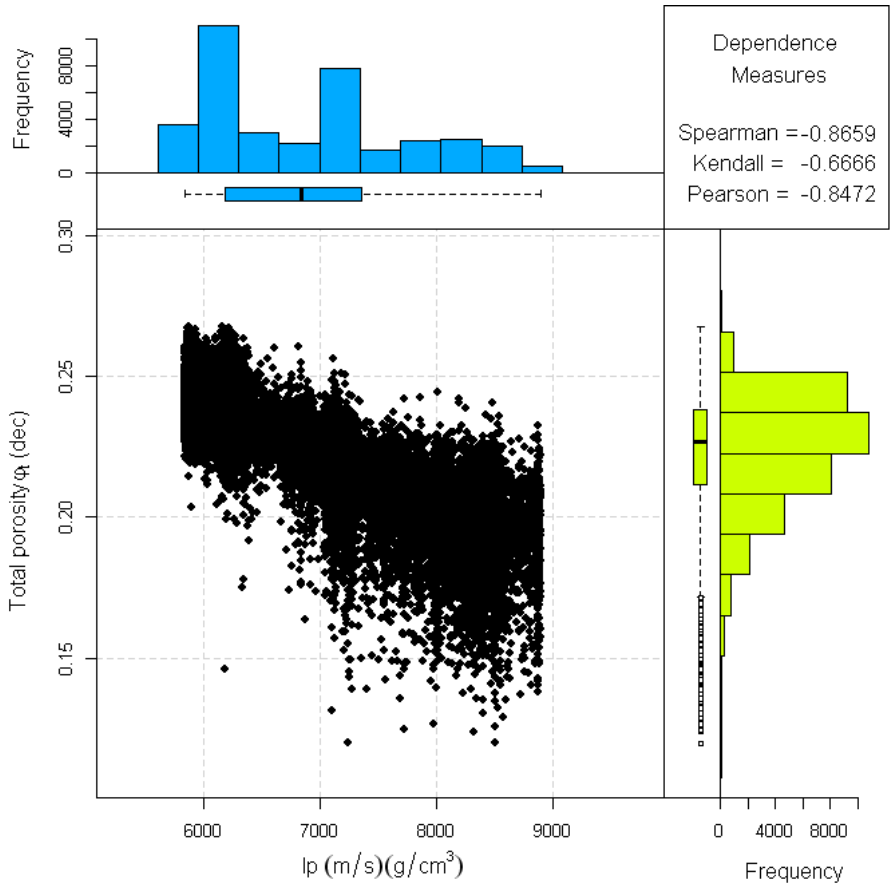


The results in the next comparative table, the total porosity upscaled and simulated has values closed, but the scatter plot shows a different perspective.

[88]:
```
Comparative_post_Stat_2<-cbind(Ip_Stat, Ip_like_Stat[,2],XFrF_Ip_2_Stat[,2],
                               ␣
 ↪Phit_Stat[,2],Phit_like_Stat[,2],XFrF_Phit_2_Stat[,2])
#Comparative_post_Stat<-Val_Estadisticos(Comparative_post)
colnames(Comparative_post_Stat_2)<-c("Statistics","Ip","Ip upscaled","Ip␣
 ↪simulated","phit","phit upscaled", "phit simulated")
Comparative_post_Stat_2
```

	Statistics <chr>	Ip <dbl>	Ip upscaled <dbl>	Ip simulated <dbl>	phit <dbl>	phit upscaled <dbl>	phit <dbl
muestras	n	3696.0000	3696.0000	36960.0000	3696.0000	3696.0000	3696
minimos	Minimum	5086.0072	5835.0969	5835.0969	0.0620	0.1655	0.119
cuantiles1	1st. Quartile	6157.0052	6178.1181	6178.1181	0.2147	0.2126	0.211
medianas	Median	6809.5574	6831.4653	6831.4653	0.2295	0.2320	0.227
medias	Mean	6968.2839	6892.8985	6892.8985	0.2220	0.2221	0.223
cuantiles3	3rd. Quartile	7321.6170	7353.4905	7353.4905	0.2414	0.2369	0.238
maximos	Maximum	11661.4642	8891.2124	8891.2124	0.2939	0.2611	0.268
rangos	Rank	6575.4570	3056.1155	3056.1155	0.2319	0.0956	0.148
rangosInt	Interquartile Rank	1164.6118	1175.3724	1175.3724	0.0267	0.0243	0.026
varianzas	Variance	1310302.9359	708798.2641	708625.6625	0.0011	0.0005	0.000
desvs	Standard Deviation	1144.6846	841.9016	841.7991	0.0338	0.0223	0.019
CVs	Variation Coeff.	0.1643	0.1221	0.1221	0.1521	0.1003	0.088
simetrias	Skewness	1.5610	0.5967	0.5969	-1.8102	-1.2139	-0.87
curtosiss	Kurtosis	5.7657	2.1898	2.1898	6.9970	3.2414	3.757

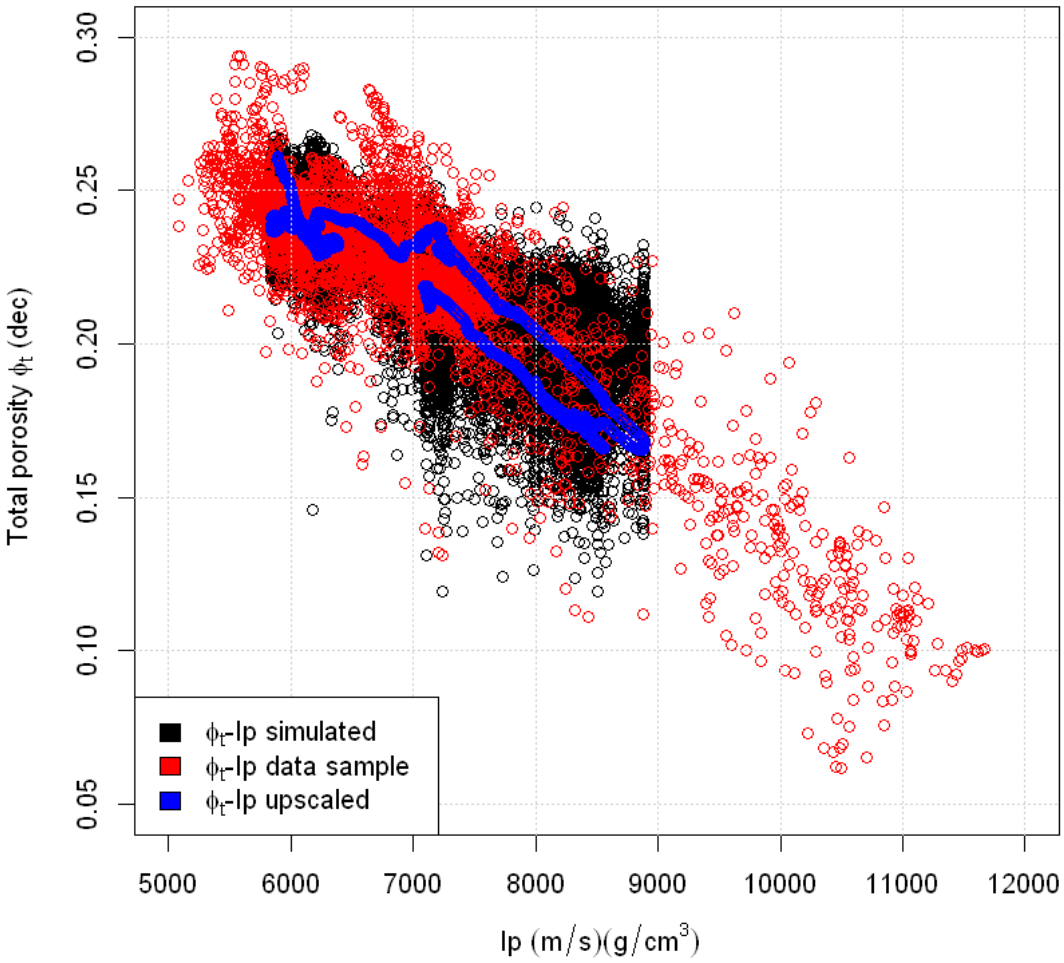
[89]:
```
ScatterPlot(XFrF_Ip_2 , XFrF_Phit_2, ns,
            Xmin = XFrF_Ip_2_Stat[2,2], Xmax = XFrF_Ip_2_Stat[7,2],
            Ymin = XFrF_Phit_2_Stat[2,2],Ymax = XFrF_Phit_2_Stat[7,2],
            XLAB = expression(paste(" Ip ",(m/s)(g/cm^3))),
        YLAB = expression(paste(" Total porosity ",varphi[t], " (dec)")))
```



Comparing the scatter plot data sample (red), upscaled sample (blue) and simulate data values (black), the simulate data values are concentrated in the center of the model, even the scatter plot looks like with less slope.

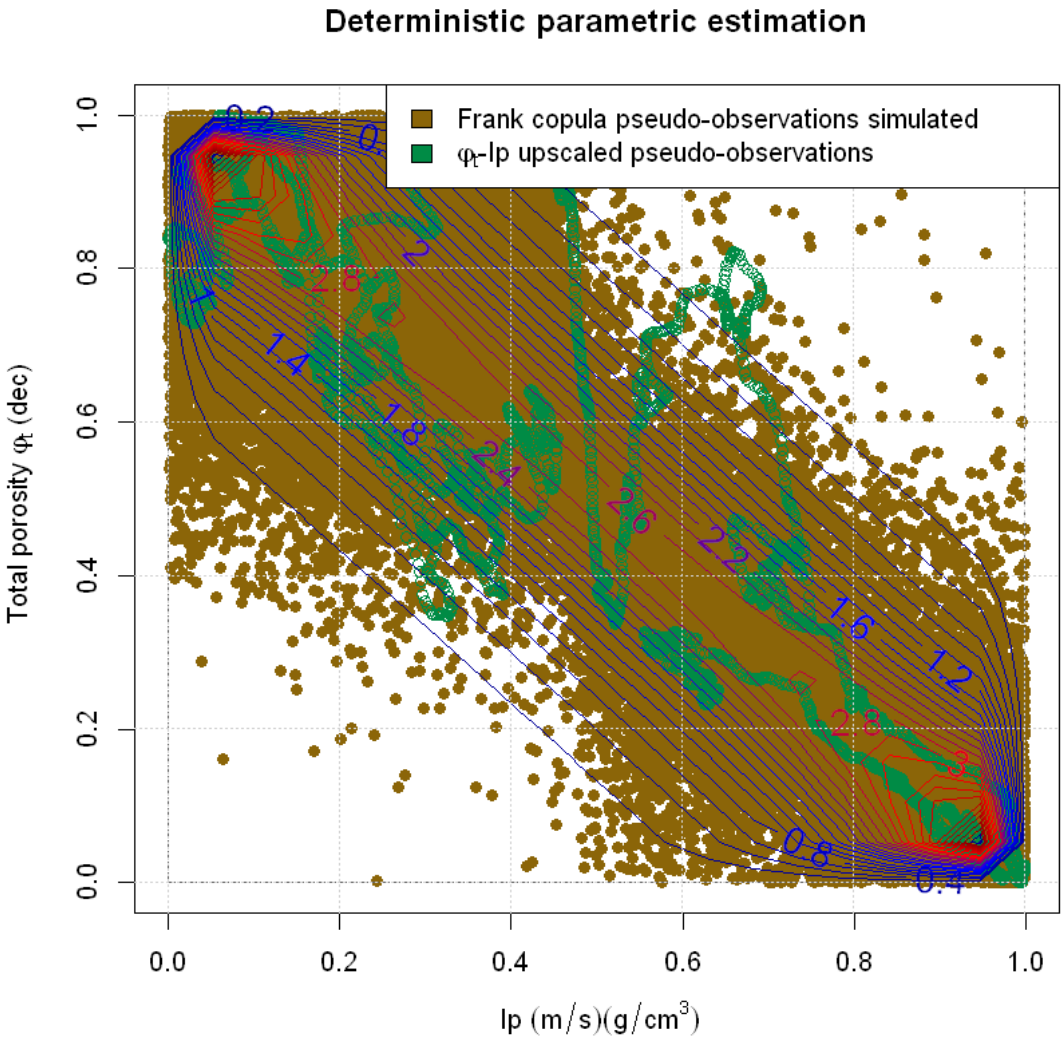
[10]:
```
plot(XFrF_Ip_2 , XFrF_Phit_2,xlim=c(5000, 12000),ylim=c(0.05, 0.
↪3),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Ip , Phit, xlim=c(5000, 12000),ylim=c(0.05, 0.
↪3),col=("red"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Ip_like , Phit_like, xlab = expression(paste(" Ip ",(m/s)(g/cm^3))),
        ylab = expression(paste(" Total porosity ",phi[t], "␣
↪(dec)")),xlim=c(5000, 12000),ylim=c(0.05, 0.3),
```

```
     col=("blue"))
grid()
legend(x = "bottomleft", legend = c(expression(paste(phi[t],"-Ip simulated")),
                                    expression(paste(phi[t],"-Ip data sample")),
                                   expression(paste(phi[t],"-Ip upscaled"))),
       fill = c("black", "red","blue"))
```



[91]:
```
plot(pobs(cbind(XFrF_Ip_2 , XFrF_Phit_2)), pch=16,␣
 ↪col=("darkgoldenrod4"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(pobs(cbind(Ip_like , Phit_like)), xlab=(""), ylab=(""),
     main = ("Deterministic parametric estimation"), col=("springgreen4"))
```

```
par(new=TRUE)
contour(frankCopula(ifme_frank_2@copula@parameters), dCopula,n=20,nlevels=25,␣
↪xlab = expression(paste(" Ip ",(m/s)(g/cm^3))),
    ylab = expression(paste(" Total porosity ",varphi[t], " (dec)")) ,
       labcex=1.4, col=cols,xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend = c("Frank copula pseudo-observations simulated",
                                  expression(paste(varphi[t],"-Ip upscaled␣
↪pseudo-observations"))),
      fill = c("darkgoldenrod4", "springgreen4"))
```



8.3 Total porosity simulation using the deterministic approach model.

With the samples simulated in the last step, the total porosity is simulated. As Bayesian approach, the simulations are genetared whit simulated annealing method, the objective function is the same variogram. The total porosity statistics in the next comparative table shows values really closed. If these values are compared with the total porosity simulation using the bayesian approach model, the differences are similar in each statistic parameter but minimun value in bayesian approach case.

[92]:
```
Data_like_FULL_100_det <- read.table(file='Results/100_sim_all_full_DET.
↪txt',header=T,na.strings="-999.25")
```

[93]:
```
Ip_100_det<-Data_like_FULL_100_det$Ip
Phit_100_det<-Data_like_FULL_100_det$Phit
```

[94]:
```
Ip_100_det_Stat<-Estadisticas(Ip_100_det)
Phit_100_det_Stat<-Estadisticas(Phit_100_det)
```

[95]:
```
Comparative_det_Stat_2<-cbind(Ip_Stat, Ip_like_Stat[,2],Ip_100_det_Stat[,2],
                              ␣
↪Phit_Stat[,2],Phit_like_Stat[,2],Phit_100_det_Stat[,2])
#Comparative_post_Stat<-Val_Estadisticos(Comparative_post)
colnames(Comparative_det_Stat_2)<-c("Statistics","Ip","Ip upscaled","Ip␣
↪simulated","phit","phit upscaled", "phit simulated")
Comparative_det_Stat_2
```

	Statistics <chr>	Ip <dbl>	Ip upscaled <dbl>	Ip simulated <dbl>	phit <dbl>	phit upscaled <dbl>	phit <dbl
muestras	n	3696.0000	3696.0000	369600.0000	3696.0000	3696.0000	3696
minimos	Minimum	5086.0072	5835.0969	5835.0967	0.0620	0.1655	0.119
cuantiles1	1st. Quartile	6157.0052	6178.1181	6178.1183	0.2147	0.2126	0.210
medianas	Median	6809.5574	6831.4653	6831.4651	0.2295	0.2320	0.227
medias	Mean	6968.2839	6892.8985	6892.8985	0.2220	0.2221	0.222
cuantiles3	3rd. Quartile	7321.6170	7353.4905	7353.4907	0.2414	0.2369	0.238
maximos	Maximum	11661.4642	8891.2124	8891.2119	0.2939	0.2611	0.267
rangos	Rank	6575.4570	3056.1155	3056.1152	0.2319	0.0956	0.148
rangosInt	Interquartile Rank	1164.6118	1175.3724	1175.3724	0.0267	0.0243	0.027
varianzas	Variance	1310302.9359	708798.2641	708608.4035	0.0011	0.0005	0.000
desvs	Standard Deviation	1144.6846	841.9016	841.7888	0.0338	0.0223	0.023
CVs	Variation Coeff.	0.1643	0.1221	0.1221	0.1521	0.1003	0.106
simetrias	Skewness	1.5610	0.5967	0.5970	-1.8102	-1.2139	-1.34
curtosiss	Kurtosis	5.7657	2.1898	2.1898	6.9970	3.2414	5.383

[96]:
```
n=100 #numero de columnas
f=3696  #numero de filas
ff<-0
```

[97]:
```
sepf=matrix(0,n,2)
for (i in 1:n) {
```

```
  ff<-ff+f
 sepf[i,2]<-ff
}
sepf[,1]<-sepf[,2]-(f-1)
```

[98]:
```
Data_Phit_full_det=matrix(0,f,n)
for (i in 1:n) {
  Data_Phit_full_det[,i]<-Data_like_FULL_100_det[sepf[i,1]:sepf[i,2],5]
}
```

[99]:
```
XFrF_Phit_det_Stat<-Val_Estadisticos(Data_Phit_full_det)
XFrF_Phit_det_Stat
```

No_muestras	3696.00000	3696.00000	3696.00000	3696.00000	3696.00000	3696.00000	3696.0000
Minimo	0.12043	0.12168	0.12415	0.12134	0.12271	0.12186	0.12037
Cuartil_1er	0.21094	0.21116	0.21090	0.21098	0.21097	0.21109	0.21097
Mediana	0.22720	0.22718	0.22700	0.22707	0.22717	0.22730	0.22748
Media	0.22209	0.22219	0.22218	0.22218	0.22222	0.22219	0.22191
Cuartil_3er	0.23838	0.23839	0.23844	0.23849	0.23861	0.23846	0.23841
Maximo	0.26759	0.26655	0.26698	0.26744	0.26592	0.26751	0.26781
Rango	0.14716	0.14487	0.14283	0.14610	0.14321	0.14565	0.14744
Rango_Intercuartil	0.02744	0.02724	0.02754	0.02751	0.02765	0.02737	0.02744
Varianza	0.00055	0.00055	0.00054	0.00055	0.00055	0.00055	0.00058
Desv_Estandar	0.02351	0.02341	0.02328	0.02354	0.02338	0.02340	0.02405
Simetria	-1.33814	-1.33702	-1.30471	-1.36502	-1.28194	-1.33048	-1.44520
Curtosis	5.35084	5.39907	5.30201	5.47844	5.10136	5.38698	5.71584

[100]:
```
DF_Data_Phit_full_det<-data.frame(Data_Phit_full_det)

nc<-ncol(DF_Data_Phit_full_det)
nf<-nrow(DF_Data_Phit_full_det)
```

[101]:
```
P_det<-matrix(0, nc, 1)
```

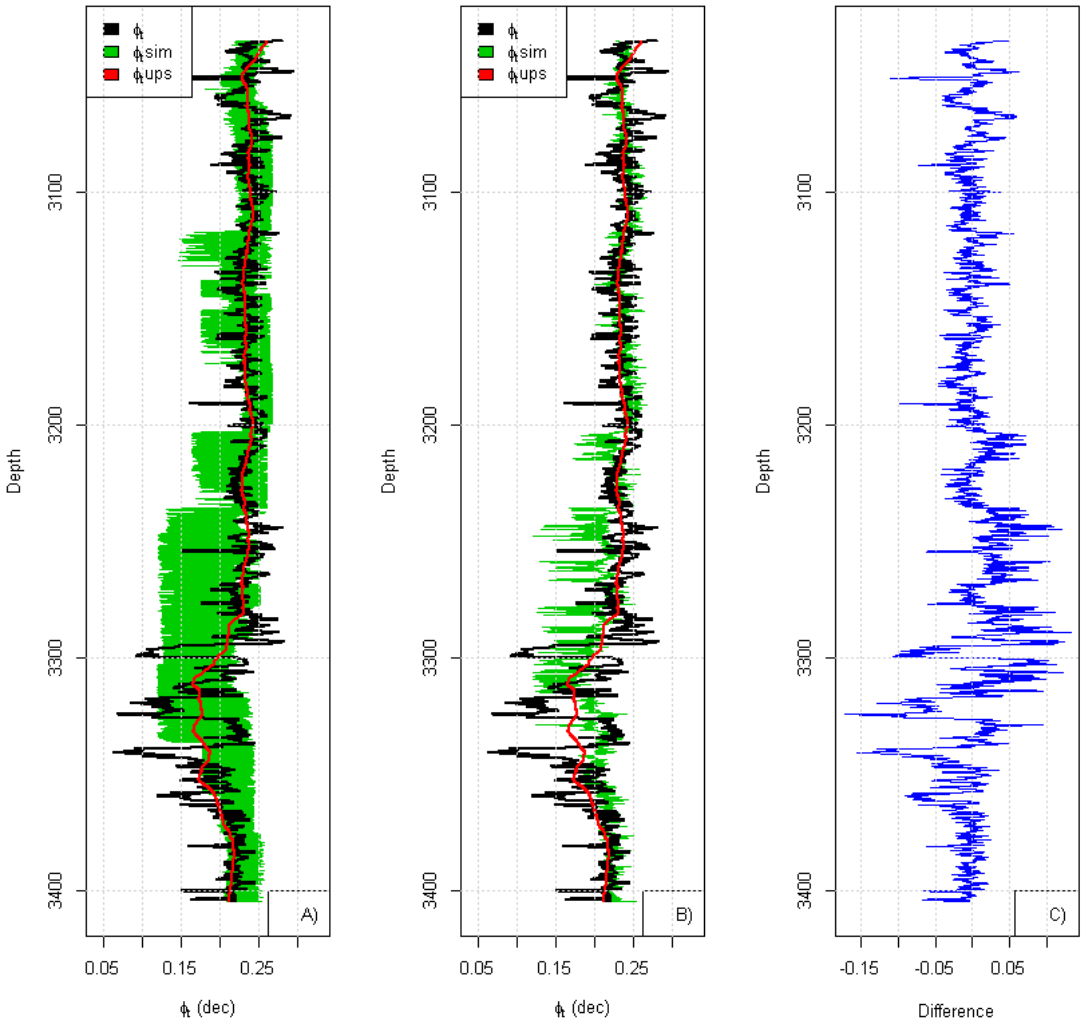
[102]:
```
for (i in 1:nc) {
   P_det[i,] <- (sum((Phit_like-DF_Data_Phit_full_det[,i])^2))/nf
}
```

[103]:
```
minP_det<-which.min(P_det)
minP_det
```

91

[8]:
```
#jpeg("Results/final/image8a.jpeg", bg = "white", width = 2500, height = 2500,␣
↪res = 300)

par(mfrow = c(1,3))
for (i in 1:ncol(Data_Phit_full_det)) {
```

```
   plot(Data_Phit_full_det[,i],Depth,xlim=c(0.04, 0.
↪33),ylim=rev(range(Depth)),type = "l",lwd=1,col=("green3"),
        xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
}
plot(Phit,Depth,xlab = expression(paste(phi[t], " (dec)")),ylab="Depth",xlim=c(0.
↪04, 0.33),ylim=rev(range(Depth)),
     type = "l",lwd=2,col=("black"))
par(new=TRUE)
plot(Phit_like,Depth,xlab = expression(paste(phi[t], "␣
↪(dec)")),ylab="Depth",xlim=c(0.04, 0.33),
     ylim=rev(range(Depth)),
     type = "l",lwd=2,col=("red"))
legend(x = "topleft", legend =␣
↪c(expression(paste(phi[t])),expression(paste(phi[t], "sim")),
                                expression(paste(phi[t], "ups"))),
       fill = c("black", "green3","red"))
legend(x="bottomright",legend="A)")
grid()

plot(Data_Phit_full_det[,minP],Depth,xlim=c(0.04, 0.
↪33),ylim=rev(range(Depth)),type = "l",
     lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Phit,Depth,xlab = expression(paste(phi[t], " (dec)")),ylab="Depth",xlim=c(0.
↪04, 0.33),ylim=rev(range(Depth)),
     type = "l",lwd=2,col=("black"))
par(new=TRUE)
plot(Phit_like,Depth,xlab = expression(paste(phi[t], "␣
↪(dec)")),ylab="Depth",xlim=c(0.04, 0.33),
     ylim=rev(range(Depth)),
     type = "l",lwd=2,col=("red"))
legend(x = "topleft", legend =␣
↪c(expression(paste(phi[t])),expression(paste(phi[t], "sim")),
                                expression(paste(phi[t], "ups"))),
       fill = c("black", "green3","red"))
legend(x="bottomright",legend="B)")
grid()

plot((Phit-Data_Phit_full_det[,minP]),Depth,xlab="Difference",ylab="Depth",ylim=rev(range(Depth
     type = "l",lwd=1,col=("blue"))
legend(x="bottomright",legend="C)")
grid()

#dev.off()
```



Comparing the spatial distributions between both approaches, deterministic and Bayesian, shows that the coverage is not as good as that obtained from the Bayesian case, there are even considerable sections that deviate from the upscale log. Therefore, it can be considered that the deterministic method could not satisfactorily cover the requirements of the model to take it to the seismic scale case.

9 Integration of seismic-scale acoustic impedance as new information to the joint probability distribution function model

The results obtained in previous tests demonstrated that Bayesian approach allows consistent results at well-scale observations and dependency model, which is not obtained with deterministic approach.

Now the acoustic impedance at seismic scale is used as new information, the acoustic impedance

was obtained from Sparse-Spike inversion, this inversion has a sampling of 4 milliseconds (ms). The total porosity is obtained subsampling the scaled log, to obtain the subsampling the following restrictions were followed:

- The acoustic impedance at the seismic scale is in time domain, the scaled total porosity log must be transformed to the time domain, for that a time-depth model is used.
- To obtain the sample corresponding at node of seismic grid, the median value is estimated with an interval ± 2 ms from the node, for example, if the node is at 3200 ms, then the samples that are used to estimate the median should be in the range 3198 to 3202 ms.

[4]:
```
Data_like_TRA <- read.csv(file='data/inline_trace.csv',header=T,na.strings="-999.
↪25")   #open file well_f03-4_502.5_1859.55.csv
#the select file is ip_Phie.csv
```

[5]:
```
Data_like_SUB <- read.csv(file='data/subsampled_median_Lakach-1.csv',header=T,na.
↪strings="-999.25")
#the select file is ip_Phie.csv
```

[6]:
```
#Phit_like_TRA<-Data_like_SUB$PHIT_MA
Ip_like_TRA<-Data_like_TRA$q

Phit_like_SUB<-Data_like_SUB$PHIT_MA
time_like_SUB<-Data_like_SUB$time

time_like_SUB_Stat<-Estadisticas(time_like_SUB)

Data_like_TRA<-cbind(Ip_like_TRA,Phit_like_SUB)
```

```
Warning message:
"encountered a tie, and the difference between minimal and
                  maximal value is > length('x') * 'tie.limit'
the distribution could be multimodal"
```

[11]:
```
Ip_like_SUB<-Data_like_SUB$Ip_ups_Bks
Depth_SUB<-Data_like_SUB$Depth
```

[12]:
```
Phit_like_SUB_Stat<-Estadisticas(Phit_like_SUB)
Ip_like_SUB_Stat<-Estadisticas(Ip_like_SUB)
```

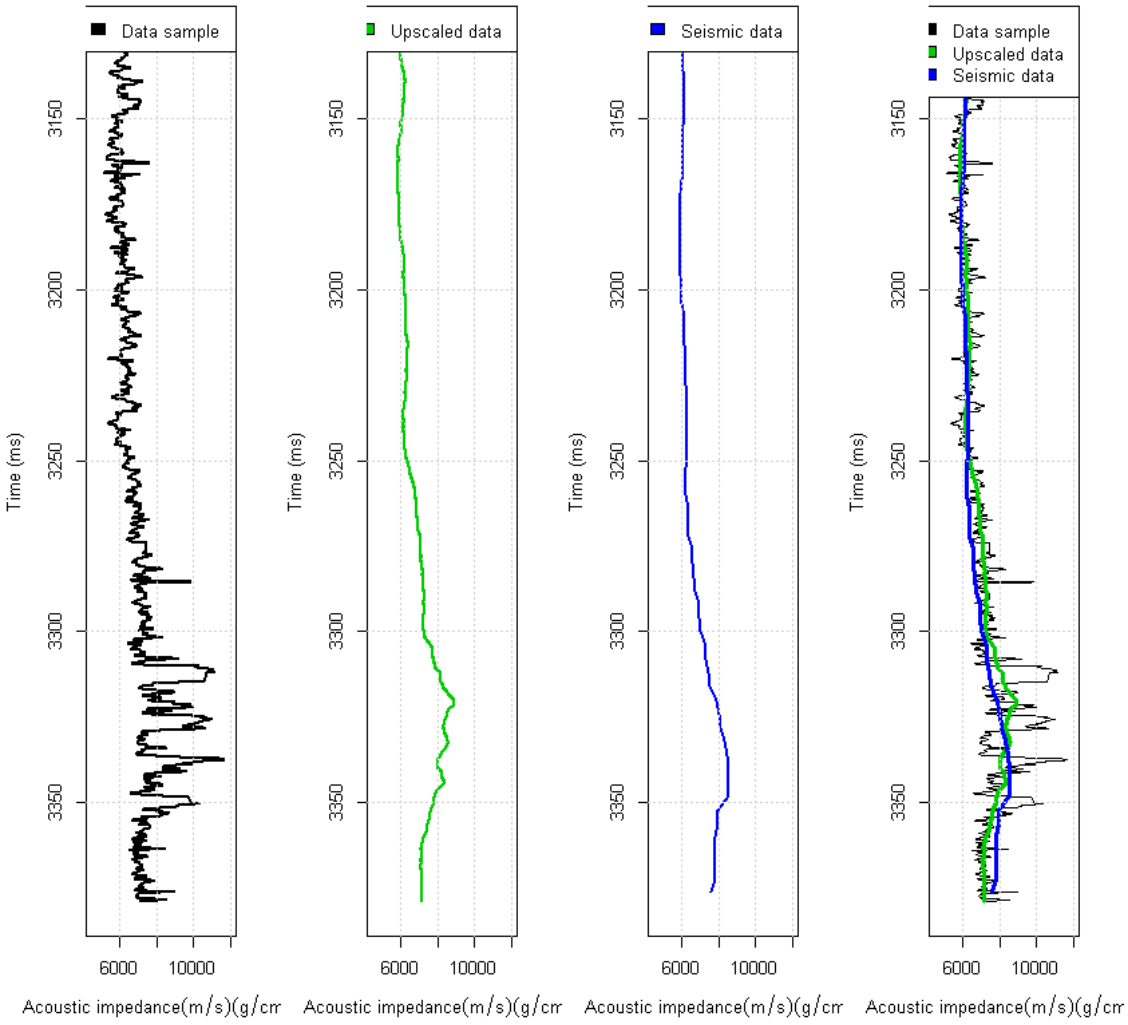
[7]:
```
Depth_SUB_Stat<-Estadisticas(Depth_SUB)
```

[10]:
```
nsp = nclass.Sturges(Ip_like_TRA)
nsp
```

7

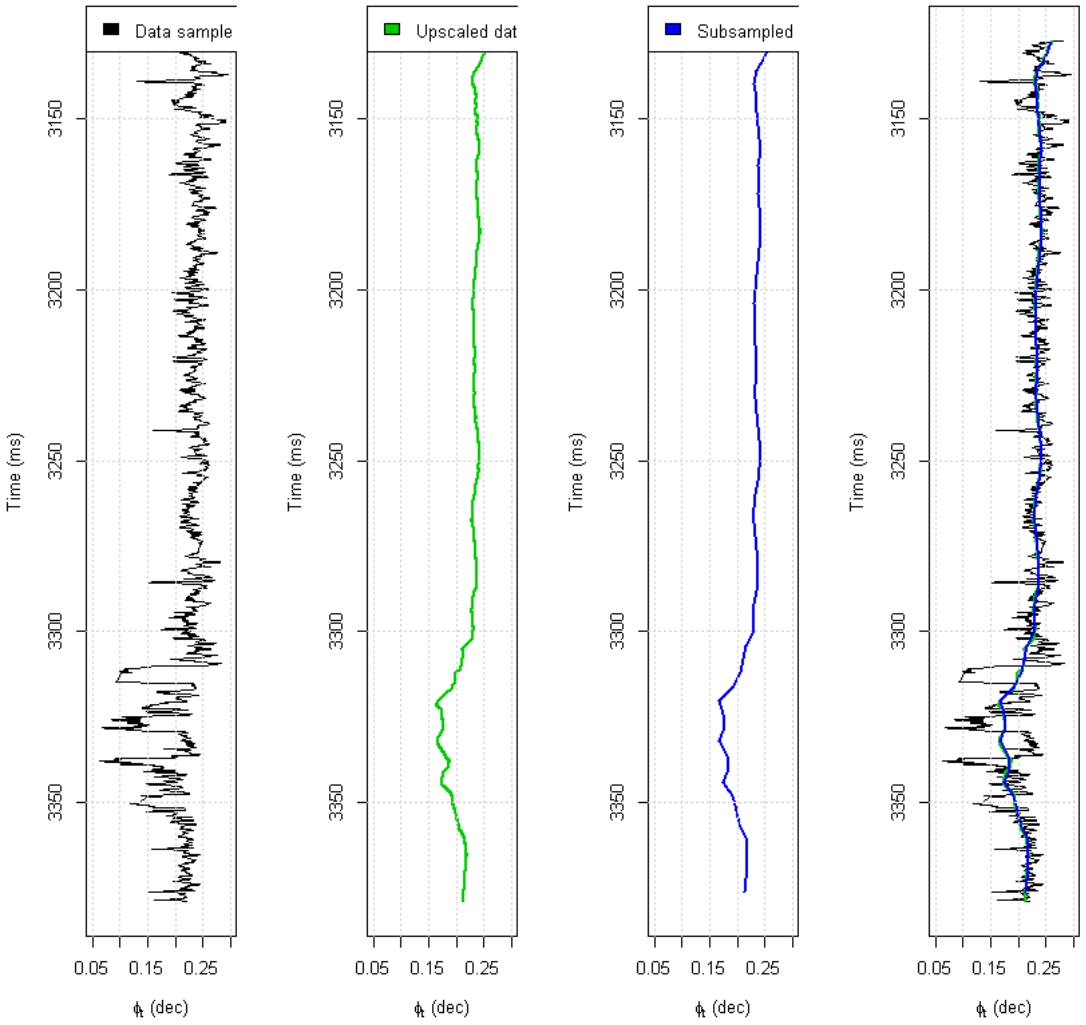
[11]:
```
time_UPS_Stat<-Estadisticas(time_UPS)
```

[111]:
```
#png("Results/Ip_all.png", bg = "white",  width=3500, height=2500, res=300)
par(mfrow = c(1,4))

plot(Ip,time_UPS,xlim=c(4500, 12000),xlab = expression(paste("Acoustic␣
↪impedance", (m/s)(g/cm^3))),
     ylab="Time (ms)",ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =␣
↪"l",lwd=2,col=("black"))
grid()
legend(x = "topright", legend = c("Data sample"), fill = c("black"))

plot(Ip_like,time_UPS,xlim=c(4500, 12000),xlab = expression(paste("Acoustic␣
↪impedance", (m/s)(g/cm^3))),
     ylab="Time (ms)",ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =␣
↪"l",lwd=2,col=("green3"))
grid()
legend(x = "topright", legend = c("Upscaled data"), fill = c( "green3"))

plot(Ip_like_TRA,time_like_SUB,xlim=c(4500, 12000),xlab =␣
↪expression(paste("Acoustic impedance", (m/s)(g/cm^3))),
     ylab="Time (ms)",ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =␣
↪"l",lwd=2,col=("blue"))
grid()
legend(x = "topright", legend = c("Seismic data"), fill = c("blue"))

plot(Ip,time_UPS,xlim=c(4500, 12000),xlab = expression(paste("Acoustic␣
↪impedance", (m/s)(g/cm^3))),
     ylab="Time (ms)",ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =␣
↪"l",lwd=1,col=("black"))
par(new=TRUE)
plot(Ip_like,time_UPS,xlim=c(4500,␣
↪12000),ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =␣
↪"l",lwd=3,col=("green3"),
     xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Ip_like_TRA,time_like_SUB,xlim=c(4500,␣
↪12000),ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =␣
↪"l",lwd=3,col=("blue"),
     xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend = c("Data sample",
                                  "Upscaled data","Seismic data"), fill =␣
↪c("black", "green3","blue"))
#dev.off()
```



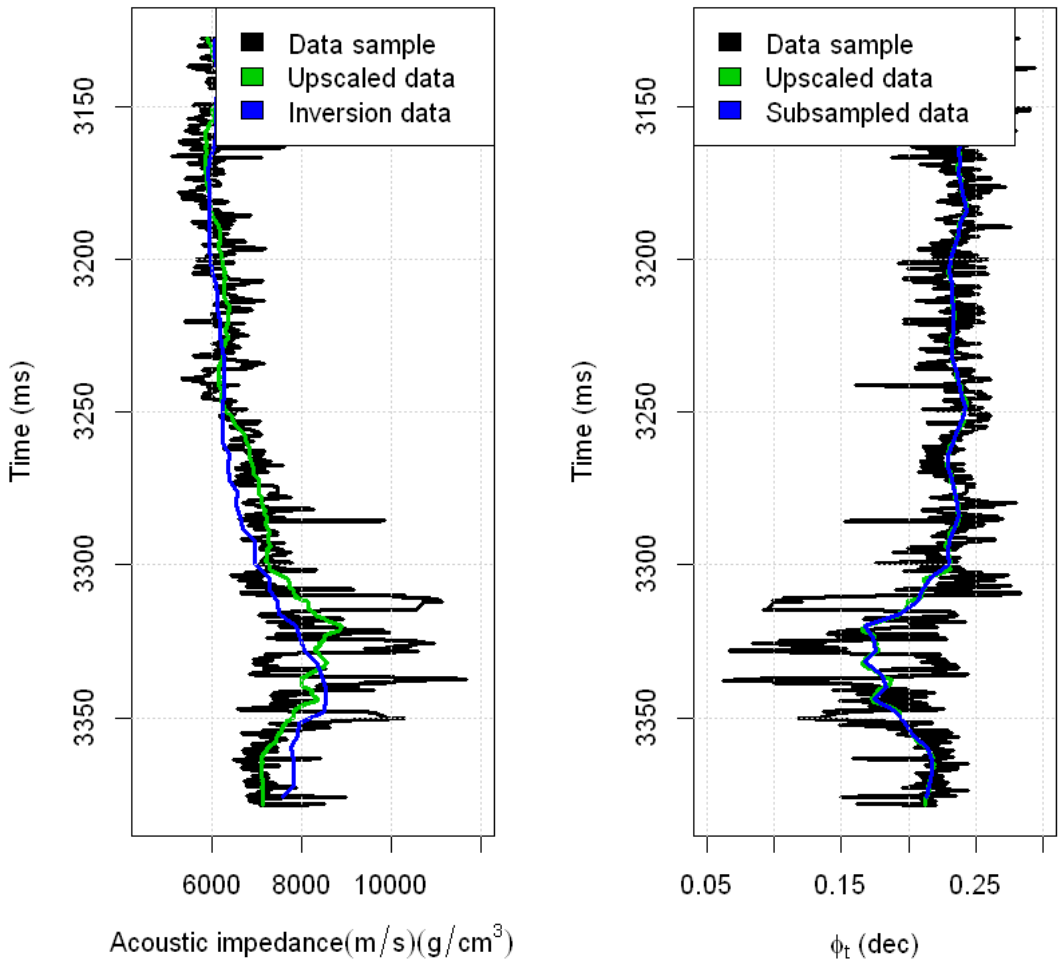
[113]:
```
#png("Results/Phit_all.png", bg = "white",  width=3500, height=2500, res=300)
par(mfrow = c(1,4))

plot(Phit,time_UPS,xlim=c(0.05, 0.3),xlab = expression(paste(phi[t], "␣
↪(dec)")),ylab="Time (ms)",
     ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =␣
↪"l",lwd=1,col=("black"))
grid()
legend(x = "topleft", legend = c("Data sample"), fill = c("black"))
```

```
plot(Phit_like,time_UPS,xlim=c(0.05, 0.
 ↪3),ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =␣
 ↪"l",lwd=2,col=("green3"),
     xlab = expression(paste(phi[t], " (dec)")),ylab="Time (ms)")
grid()
legend(x = "topleft", legend = c("Upscaled data"), fill = c("green3"))

plot(Phit_like_SUB,time_like_SUB,xlim=c(0.05, 0.
 ↪3),ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =␣
 ↪"l",lwd=2,col=("blue"),
     xlab = expression(paste(phi[t], " (dec)")),ylab="Time (ms)")
grid()
legend(x = "topleft", legend = c("Subsampled data"), fill = c("blue"))


plot(Phit,time_UPS,xlim=c(0.05, 0.3),xlab = expression(paste(phi[t], "␣
 ↪(dec)")),ylab="Time (ms)",
     ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =␣
 ↪"l",lwd=1,col=("black"))
par(new=TRUE)
plot(Phit_like,time_UPS,xlim=c(0.05, 0.
 ↪3),ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =␣
 ↪"l",lwd=2,col=("green3"),
     xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Phit_like_SUB,time_like_SUB,xlim=c(0.05, 0.
 ↪3),ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =␣
 ↪"l",lwd=2,col=("blue"),
     xaxt='n',yaxt='n',ann=FALSE)
grid()
#legend(x = "topleft", legend = c("Data sample",
#                                  "Upscaled data","Subsampled data"), fill =␣
 ↪c("black", "green3","blue"))

#dev.off()
```



[3]:
```
#png("Results/wells_upscaled_time.png", bg = "white",  width=1700, height=2100,
 ↪res=200)
par(mfrow = c(1,2))

plot(Ip,time_UPS,xlim=c(4500, 12000),xlab = expression(paste("Acoustic
 ↪impedance", (m/s)(g/cm^3))),
     ylab="Time (ms)",ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =
 ↪"l",lwd=3,col=("black"))
par(new=TRUE)
plot(Ip_like,time_UPS,xlim=c(4500,
 ↪12000),ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =
 ↪"l",lwd=3,col=("green3"),
```

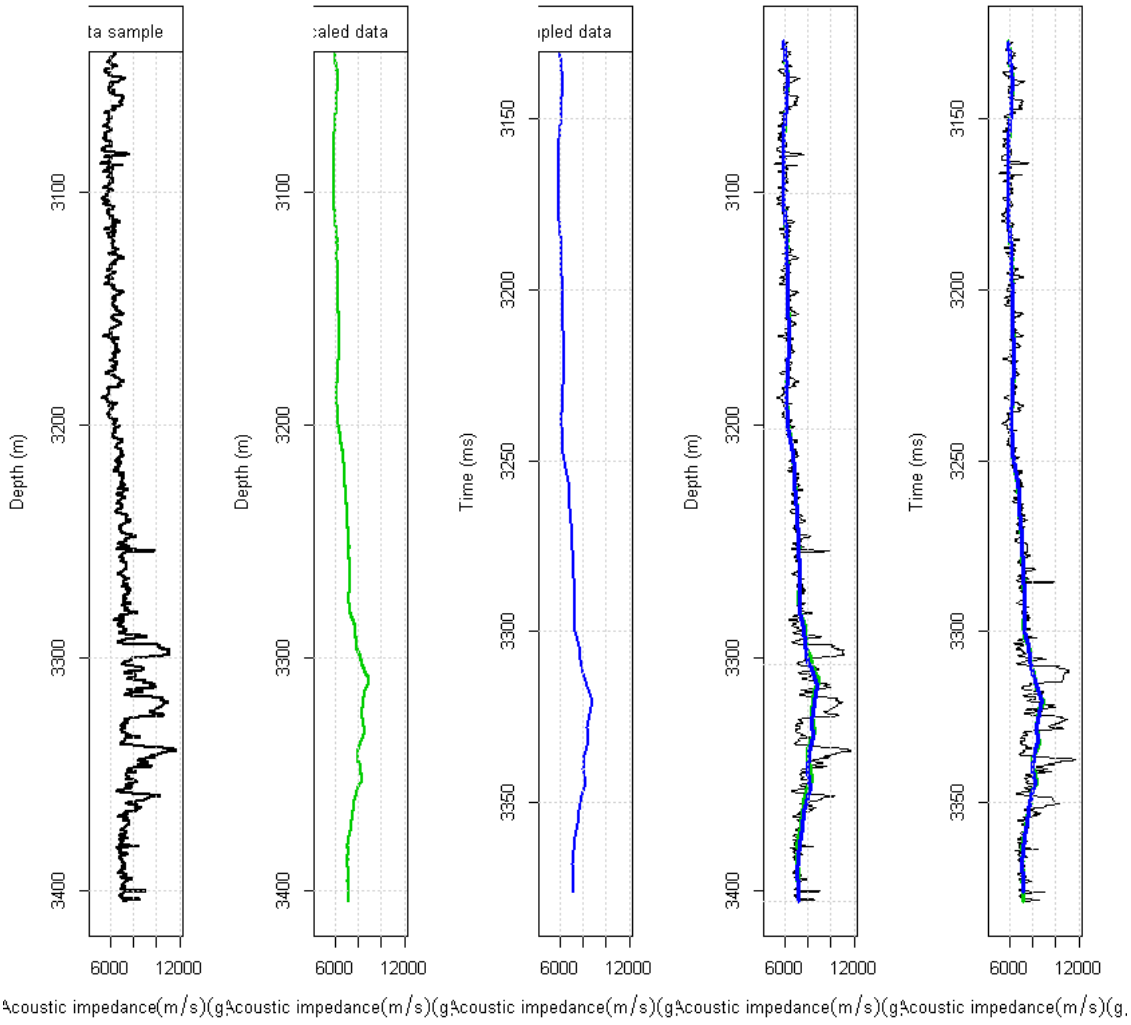
```
     xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Ip_like_TRA,time_like_SUB,xlim=c(4500,␣
 ↪12000),ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type = "l",lwd=3,
     col=("blue"),
     xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend = c("Data sample",
                                  "Upscaled data","Inversion data"), fill =␣
 ↪c("black", "green3","blue"))


plot(Phit,time_UPS,xlim=c(0.05, 0.3),xlab = expression(paste(phi[t], "␣
 ↪(dec)")),ylab="Time (ms)",
     ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =␣
 ↪"l",lwd=3,col=("black"))
par(new=TRUE)
plot(Phit_like,time_UPS,xlim=c(0.05, 0.
 ↪3),ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =␣
 ↪"l",lwd=3,col=("green3"),
     xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Phit_like_SUB,time_like_SUB,xlim=c(0.05, 0.
 ↪3),ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type = "l",lwd=3,
     col=("blue"),
     xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topleft", legend = c("Data sample",
                                  "Upscaled data","Subsampled data"), fill =␣
 ↪c("black", "green3","blue"))
#dev.off()
```



[8]:
```
#png("Results/Ip_all.png", bg = "white",  width=3500, height=2500, res=300)
par(mfrow = c(1,5))

plot(Ip,Depth,xlim=c(4500, 12000),xlab = expression(paste("Acoustic impedance",␣
 ↪(m/s)(g/cm^3))),
     ylab="Depth (m)",ylim=rev(c(Depth_Stat[2,2],Depth_Stat[7,2])),type =␣
 ↪"l",lwd=2,col=("black"))
grid()
legend(x = "topright", legend = c("Data sample"), fill = c("black"))

plot(Ip_like,Depth,xlim=c(4500, 12000),xlab = expression(paste("Acoustic␣
 ↪impedance", (m/s)(g/cm^3))),
```

```
     ylab="Depth (m)",ylim=rev(c(Depth_Stat[2,2],Depth_Stat[7,2])),type =␣
 ↪"l",lwd=2,col=("green3"))
grid()
legend(x = "topright", legend = c("Upscaled data"), fill = c( "green3"))

plot(Ip_like_SUB,time_like_SUB,xlim=c(4500, 12000),xlab =␣
 ↪expression(paste("Acoustic impedance", (m/s)(g/cm^3))),
     ylab="Time (ms)",ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =␣
 ↪"l",lwd=2,col=("blue"))
grid()
legend(x = "topright", legend = c("Subsampled data"), fill = c("blue"))

plot(Ip,Depth,xlim=c(4500, 12000),xlab = expression(paste("Acoustic impedance",␣
 ↪(m/s)(g/cm^3))),
     ylab="Depth (m)",ylim=rev(c(Depth_Stat[2,2],Depth_Stat[7,2])),type =␣
 ↪"l",lwd=1,col=("black"))
par(new=TRUE)
plot(Ip_like,Depth,xlim=c(4500,␣
 ↪12000),ylim=rev(c(Depth_Stat[2,2],Depth_Stat[7,2])),type =␣
 ↪"l",lwd=3,col=("green3"),
     xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Ip_like_SUB,Depth_SUB,xlim=c(4500,␣
 ↪12000),ylim=rev(c(Depth_SUB_Stat[2,2],Depth_SUB_Stat[7,2])),type =␣
 ↪"l",lwd=3,col=("blue"),
     xaxt='n',yaxt='n',ann=FALSE)
grid()
#legend(x = "topright", legend = c("Data sample",
#                                  "Upscaled data","Subsampled data"), fill =␣
 ↪c("black", "green3","blue"))

plot(Ip,time_UPS,xlim=c(4500, 12000),xlab = expression(paste("Acoustic␣
 ↪impedance", (m/s)(g/cm^3))),
     ylab="Time (ms)",ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =␣
 ↪"l",lwd=1,col=("black"))
par(new=TRUE)
plot(Ip_like,time_UPS,xlim=c(4500,␣
 ↪12000),ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =␣
 ↪"l",lwd=3,col=("green3"),
     xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Ip_like_SUB,time_like_SUB,xlim=c(4500,␣
 ↪12000),ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =␣
 ↪"l",lwd=3,col=("blue"),
     xaxt='n',yaxt='n',ann=FALSE)
grid()
```

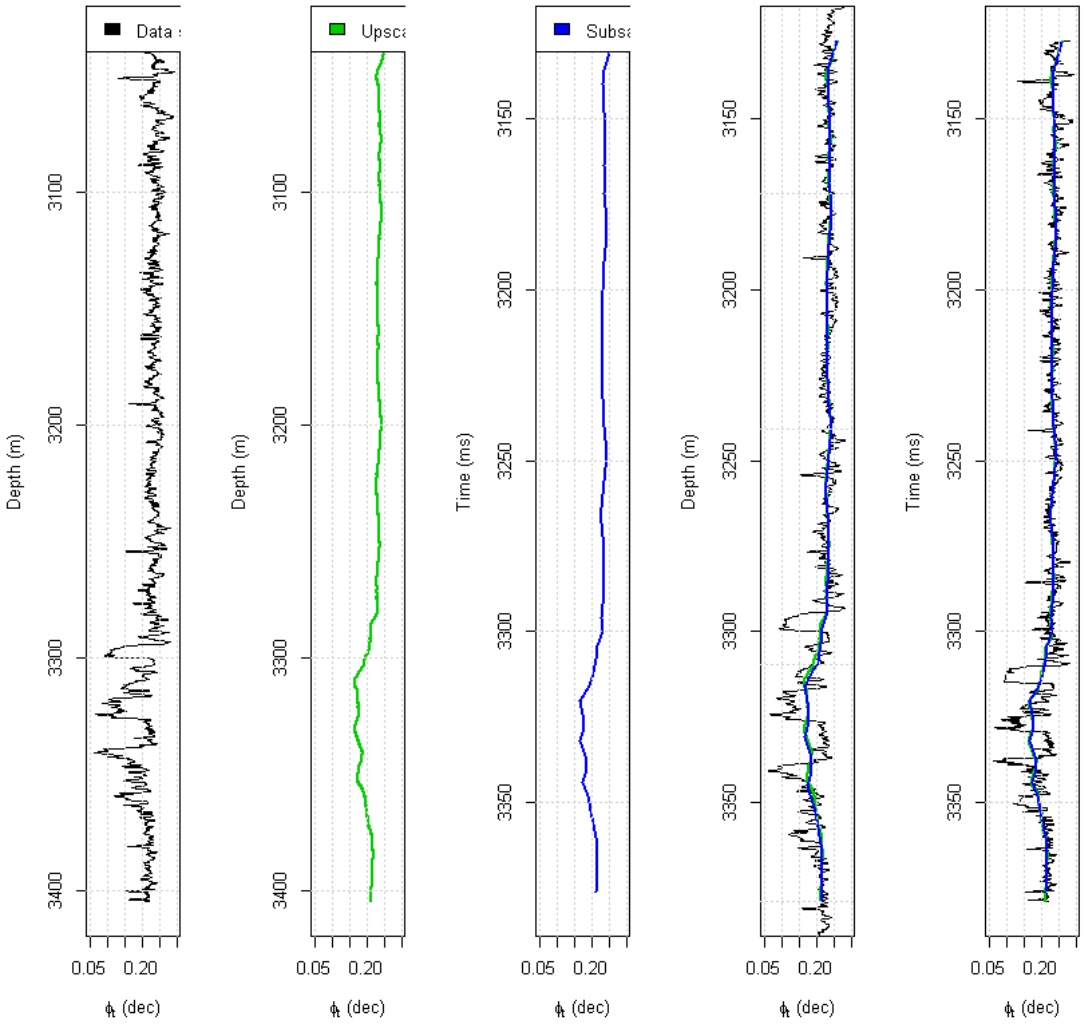
```
#legend(x = "topright", legend = c("Data sample",
#                                  "Upscaled data","Subsampled data"), fill =␣
 ↪c("black", "green3","blue"))
#dev.off()
```



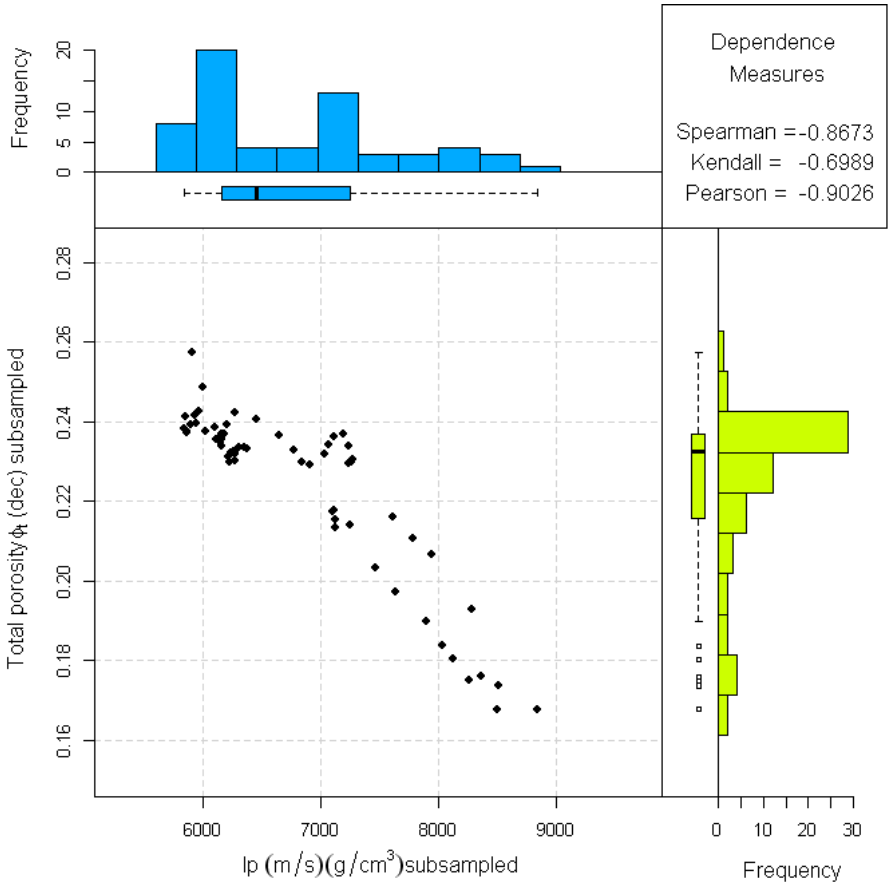
[9]:
```
#png("Results/Phit_all.png", bg = "white",  width=3500, height=2500, res=300)
par(mfrow = c(1,5))

plot(Phit,Depth,xlim=c(0.05, 0.3),xlab = expression(paste(phi[t], "␣
 ↪(dec)")),ylab="Depth (m)",
     ylim=rev(c(Depth_Stat[2,2],Depth_Stat[7,2])),type = "l",lwd=1,col=("black"))
grid()
```

```
legend(x = "topleft", legend = c("Data sample"), fill = c("black"))

plot(Phit_like,Depth,xlim=c(0.05, 0.
 ↪3),ylim=rev(c(Depth_Stat[2,2],Depth_Stat[7,2])),type =␣
 ↪"l",lwd=2,col=("green3"),
     xlab = expression(paste(phi[t], " (dec)")),ylab="Depth (m)")
grid()
legend(x = "topleft", legend = c("Upscaled data"), fill = c("green3"))

plot(Phit_like_SUB,time_like_SUB,xlim=c(0.05, 0.
 ↪3),ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =␣
 ↪"l",lwd=2,col=("blue"),
     xlab = expression(paste(phi[t], " (dec)")),ylab="Time (ms)")
grid()
legend(x = "topleft", legend = c("Subsampled data"), fill = c("blue"))

plot(Phit,Depth,xlim=c(0.05, 0.3),xlab = expression(paste(phi[t], "␣
 ↪(dec)")),ylab="Depth (m)",
     ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =␣
 ↪"l",lwd=1,col=("black"))
par(new=TRUE)
plot(Phit_like,Depth,xlim=c(0.05, 0.
 ↪3),ylim=rev(c(Depth_Stat[2,2],Depth_Stat[7,2])),type =␣
 ↪"l",lwd=2,col=("green3"),
     xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Phit_like_SUB,Depth_SUB,xlim=c(0.05, 0.
 ↪3),ylim=rev(c(Depth_SUB_Stat[2,2],Depth_SUB_Stat[7,2])),type =␣
 ↪"l",lwd=2,col=("blue"),
     xaxt='n',yaxt='n',ann=FALSE)
grid()
#legend(x = "topleft", legend = c("Data sample",
#                                  "Upscaled data","Subsampled data"), fill =␣
 ↪c("black", "green3","blue"))

plot(Phit,time_UPS,xlim=c(0.05, 0.3),xlab = expression(paste(phi[t], "␣
 ↪(dec)")),ylab="Time (ms)",
     ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =␣
 ↪"l",lwd=1,col=("black"))
par(new=TRUE)
plot(Phit_like,time_UPS,xlim=c(0.05, 0.
 ↪3),ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =␣
 ↪"l",lwd=2,col=("green3"),
     xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
```

```
plot(Phit_like_SUB,time_like_SUB,xlim=c(0.05, 0.
 ↪3),ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =␣
 ↪"l",lwd=2,col=("blue"),
     xaxt='n',yaxt='n',ann=FALSE)
grid()
#legend(x = "topleft", legend = c("Data sample",
#                                  "Upscaled data","Subsampled data"), fill =␣
 ↪c("black", "green3","blue"))

#dev.off()
```



[13]:
```
#png("Results/scatterplot_subsampled.png", bg = "white",  width=2000,␣
 ↪height=2000, res=300)
ScatterPlot(Ip_like_SUB , Phit_like_SUB,ns,
            Xmin = Ip_like_SUB_Stat[2,2], Xmax = Ip_like_SUB_Stat[7,2],
            Ymin = Phit_like_SUB_Stat[2,2],Ymax = Phit_like_SUB_Stat[7,2],
            XLAB = expression(paste(" Ip ",(m/s)(g/cm^3),"subsampled")),
            YLAB = expression(paste(" Total porosity ",phi[t], " (dec)␣
 ↪subsampled")))
#dev.off()
```

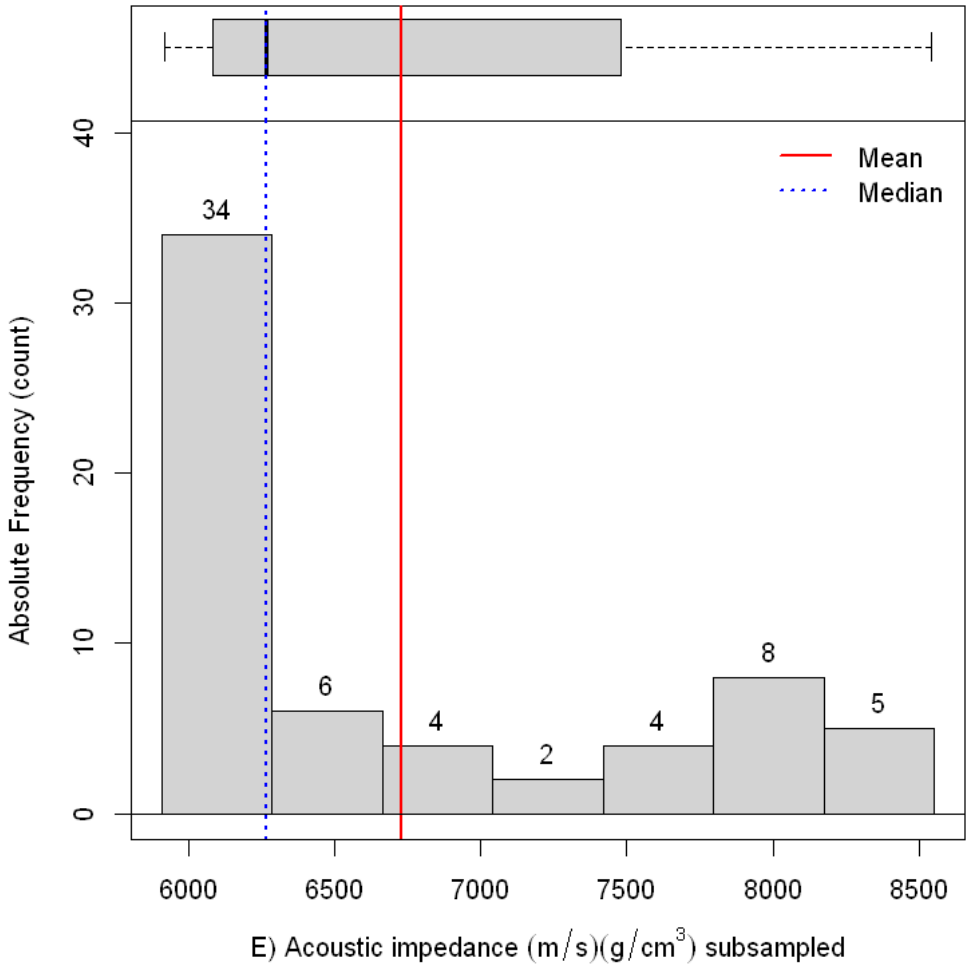


[12]:
```
Ip_like_SUB_Stat
```

	Statistics <chr>	Values <dbl>
muestras	n	63.0000
minimos	Minimum	5840.1241
cuantiles1	1st. Quartile	6159.7660
medianas	Median	6450.8228
medias	Mean	6817.8023
cuantiles3	3rd. Quartile	7254.2401
maximos	Maximum	8838.1850
rangos	Rank	2998.0609
rangosInt	Interquartile Rank	1094.4741
varianzas	Variance	687258.8260
desvs	Standard Deviation	829.0108
CVs	Variation Coeff.	0.1216
simetrias	Skewness	0.6848
curtosiss	Kurtosis	2.3504

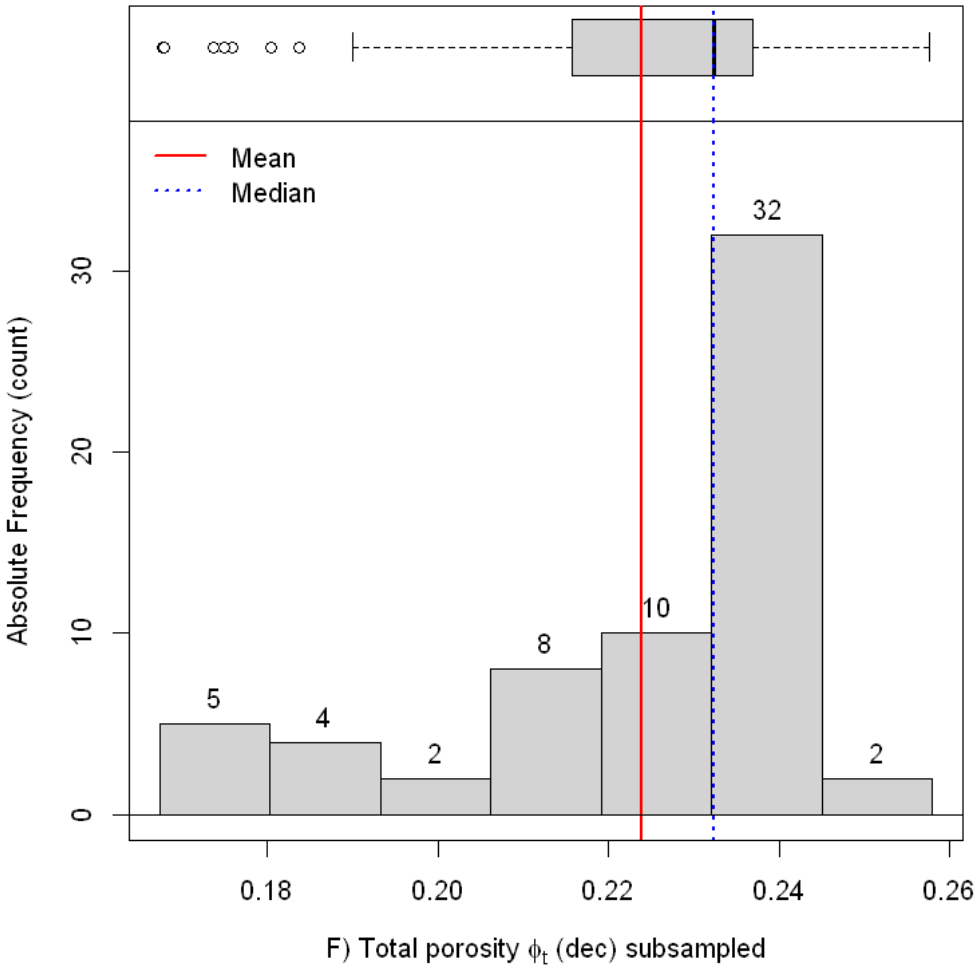
[16]:
```
Ip_like_TRA_Stat<-Estadisticas(Ip_like_TRA)
HistBoxplot(Ip_like_TRA,nbins = nsp, mean = Ip_like_TRA_Stat[5,2], median =␣
↪Ip_like_TRA_Stat[4,2], main ="",
            xlab = expression(paste("E) Acoustic impedance ",(m/s)(g/cm^3), "␣
↪subsampled")), ylab = "Absolute Frequency (count)",
            AbsFreq = TRUE, PercentFreq = FALSE )
Ip_like_TRA_Stat
```

	Statistics <chr>	Values <dbl>
muestras	n	63.0000
minimos	Minimum	5919.7222
cuantiles1	1st. Quartile	6084.1604
medianas	Median	6264.1689
medias	Mean	6727.7487
cuantiles3	3rd. Quartile	7481.5164
maximos	Maximum	8539.9961
rangos	Rank	2620.2739
rangosInt	Interquartile Rank	1397.3560
varianzas	Variance	720107.0427
desvs	Standard Deviation	848.5912
CVs	Variation Coeff.	0.1261
simetrias	Skewness	0.8545
curtosiss	Kurtosis	2.2712



[117]:
```
Phit_like_SUB_Stat<-Estadisticas(Phit_like_SUB)
HistBoxplot(Phit_like_SUB,nbins = nsp, mean = Phit_like_SUB_Stat[5,2], median =␣
↪Phit_like_SUB_Stat[4,2], main ="",
            xlab = expression(paste("F) Total porosity ",phi[t], " (dec)␣
↪subsampled")), ylab = "Absolute Frequency (count)",
            AbsFreq = TRUE, PercentFreq = FALSE )
Phit_like_SUB_Stat
```

	Statistics <chr>	Values <dbl>
muestras	n	63.0000
minimos	Minimum	0.1678
cuantiles1	1st. Quartile	0.2158
medianas	Median	0.2324
medias	Mean	0.2239
cuantiles3	3rd. Quartile	0.2369
maximos	Maximum	0.2575
rangos	Rank	0.0898
rangosInt	Interquartile Rank	0.0212
varianzas	Variance	0.0005
desvs	Standard Deviation	0.0216
CVs	Variation Coeff.	0.0964
simetrias	Skewness	-1.3036
curtosiss	Kurtosis	3.7221

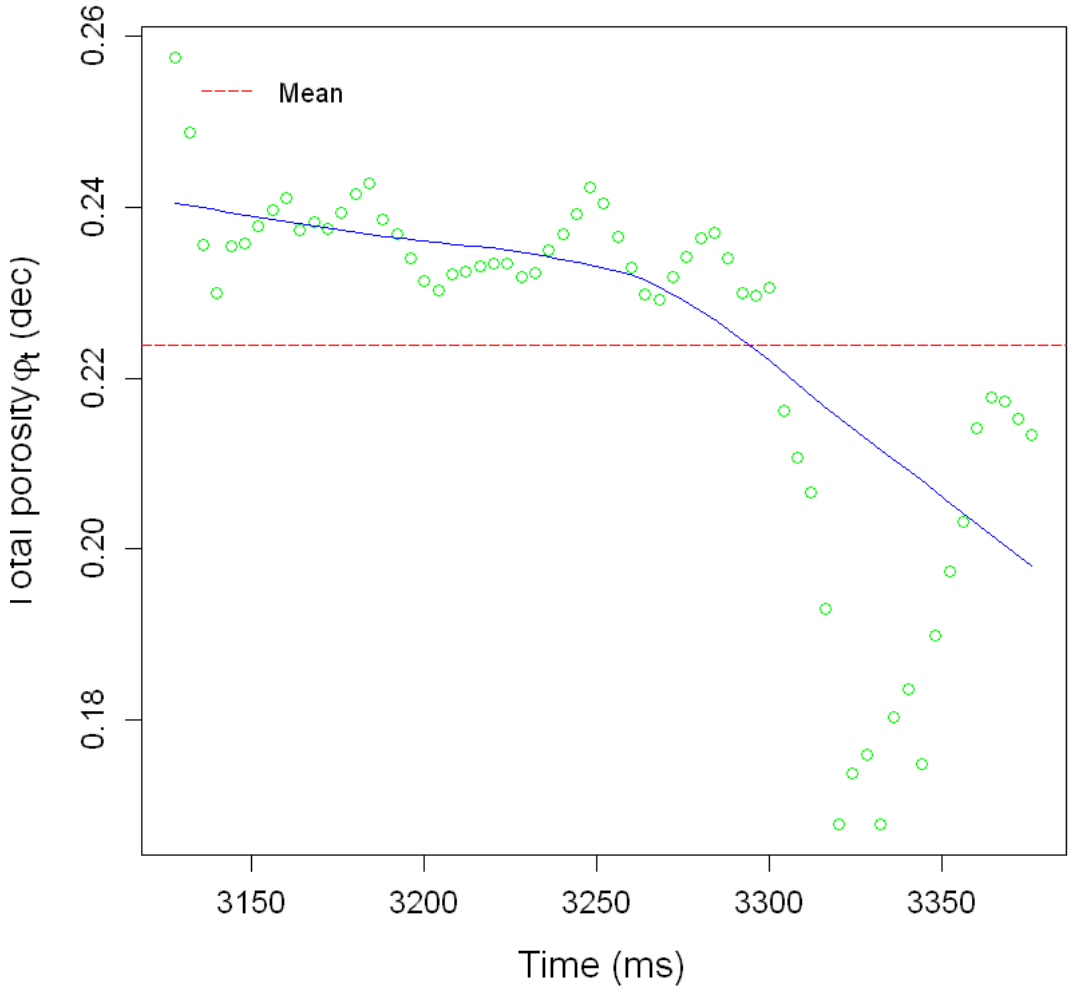


Analysing the statistics, the acoustic impedance has a interval of 5919.7222 to 8539.9961, this is a reduction of 2500 $(m/s)(g/cm^3)$ if compared to well log acoustic impedance. Meanwhile the total porosity has a interval of 0.1322 to 0.2606, this implies that minimum increases 0.07 and maximum decreases 0.03 but the mean and median are very close than the obtained from well log total porosity. Due to the acoustic impedance is obtained in the time domain, a new variogram for the total porosity is proposed.

9.1 Variographic analysis

Due to chagne to time domain, a new variogram should be proposed.

[119]:
```
MedianReg(time_like_SUB, Phit_like_SUB, Xlab= "Time (ms)",
          Varlab= expression(paste(" Total porosity ",varphi[t], " (dec)")),
          deltay=500)
```



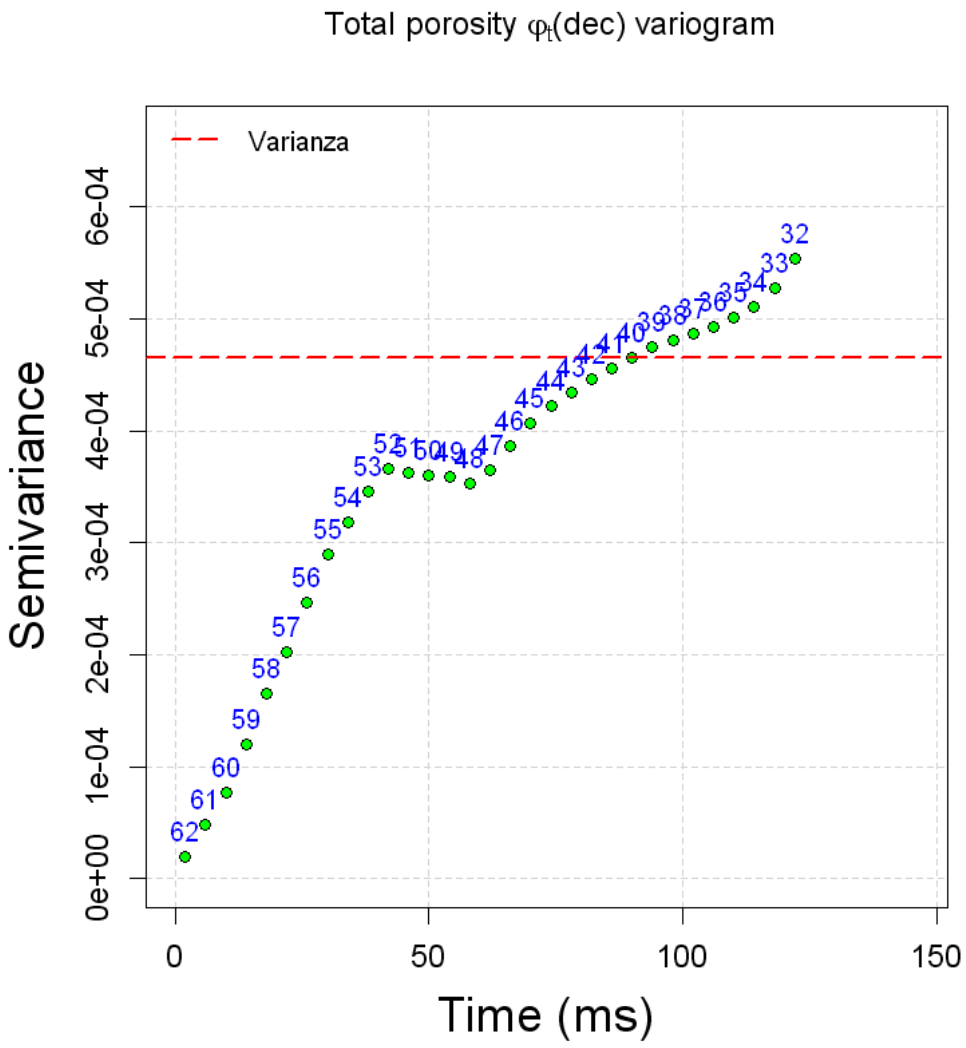
The variogram parameters are 31 lags with lag value of 4 ms, variogram direction 90 degrees and tolerance 0.

[120]:
```
N_lags<-31
lag_value <- 4
YCoord=numeric(time_like_SUB_Stat[1,2])
YCoord_Stat<-Estadisticas(YCoord)
```

```
Warning message:
"encountered a tie, and the difference between minimal and
                  maximal value is > length('x') * 'tie.limit'
the distribution could be multimodal"
```

[121]:
```
Phit_SUB_VarioEstimation<-Variograma(CoorX=time_like_SUB, CoorY=YCoord,␣
↪Variable=Phit_like_SUB,
                                Direccion=90, Tol=0, NIntervalos=N_lags,
                                Lags=lag_value,
                                MainTitle=expression(paste(" Total porosity␣
↪",varphi[t], "(dec) variogram")),
                                      xlab="Time (ms)",
                                ylab = "Semivariance")
```

```
variog: computing variogram for direction = 90 degrees (1.571 radians)
        tolerance angle = 0 degrees (0 radians)
```

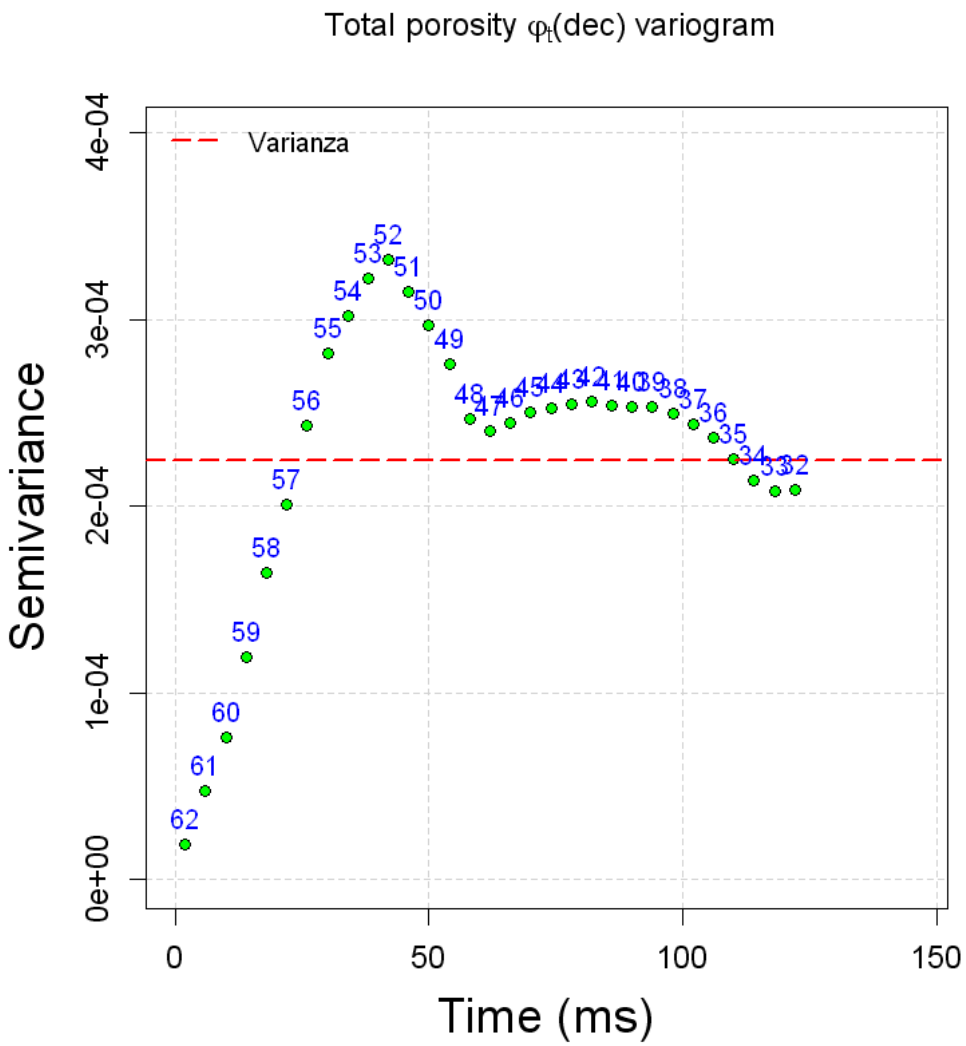


[122]:
```
pol_degree=1
Phit_like_SUB_Detrended_1<-Trend(time_like_SUB, YCoord,Phit_like_SUB, pol_degree)
```

[123]:
```
Phit_SUB_VarioEstimation_Detrended_1<-Variograma(CoorX=time_like_SUB,␣
↪CoorY=YCoord,
                                                 ␣
↪Variable=Phit_like_SUB_Detrended_1[,3],
                              Direccion=90, Tol=0, NIntervalos=N_lags,
                              Lags=lag_value,
                              MainTitle=expression(paste(" Total porosity␣
↪",varphi[t], "(dec) variogram")),
                                    xlab="Time (ms)",
```

```
                                    ylab = "Semivariance")
```

```
variog: computing variogram for direction = 90 degrees (1.571 radians)
        tolerance angle = 0 degrees (0 radians)
```

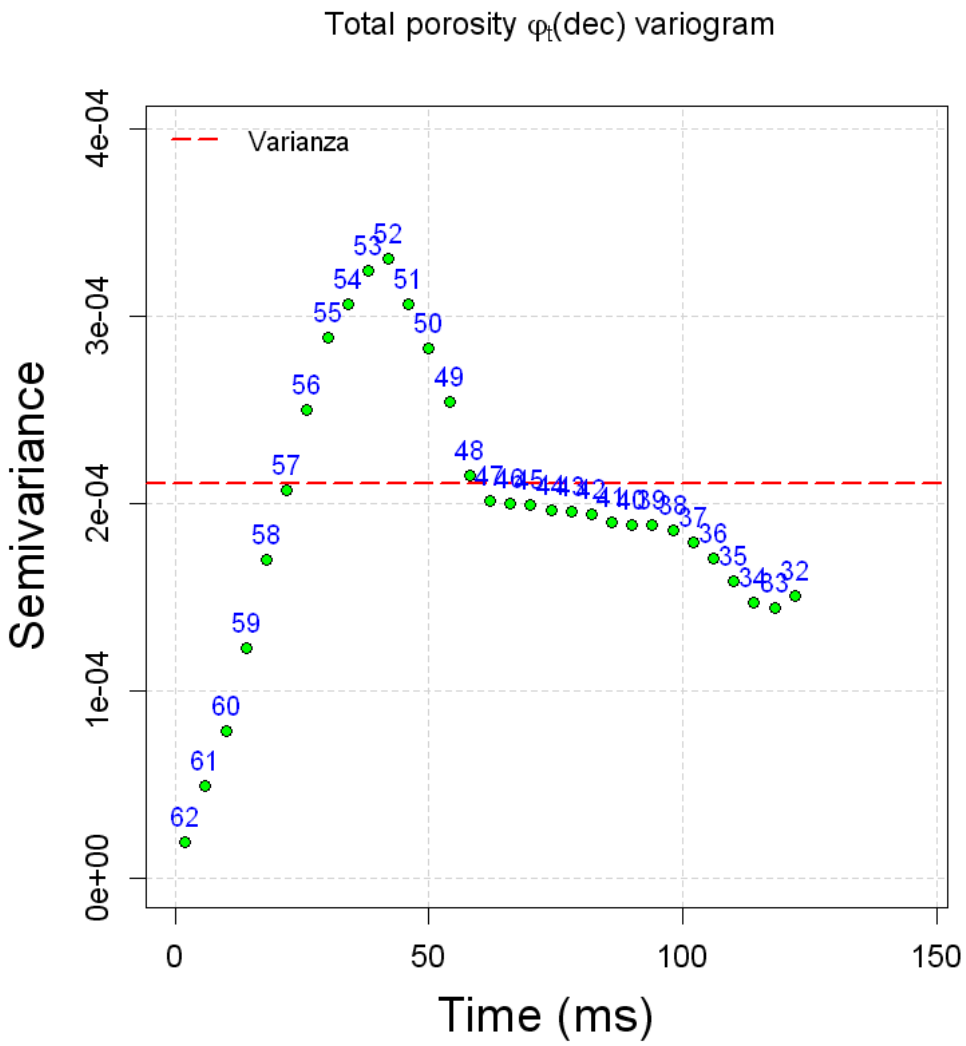


[124]:
```
pol_degree=2
Phit_like_SUB_Detrended_2<-Trend(time_like_SUB, YCoord,Phit_like_SUB, pol_degree)
```

[125]:
```
Phit_SUB_VarioEstimation_Detrended_2<-Variograma(CoorX=time_like_SUB,␣
↪CoorY=YCoord,

↪Variable=Phit_like_SUB_Detrended_2[,3],
                                Direccion=90, Tol=0, NIntervalos=N_lags,
```

```
                                Lags=lag_value,
                                MainTitle=expression(paste(" Total porosity␣
↪",varphi[t], "(dec) variogram")),
                                      xlab="Time (ms)",
                                ylab = "Semivariance")
```

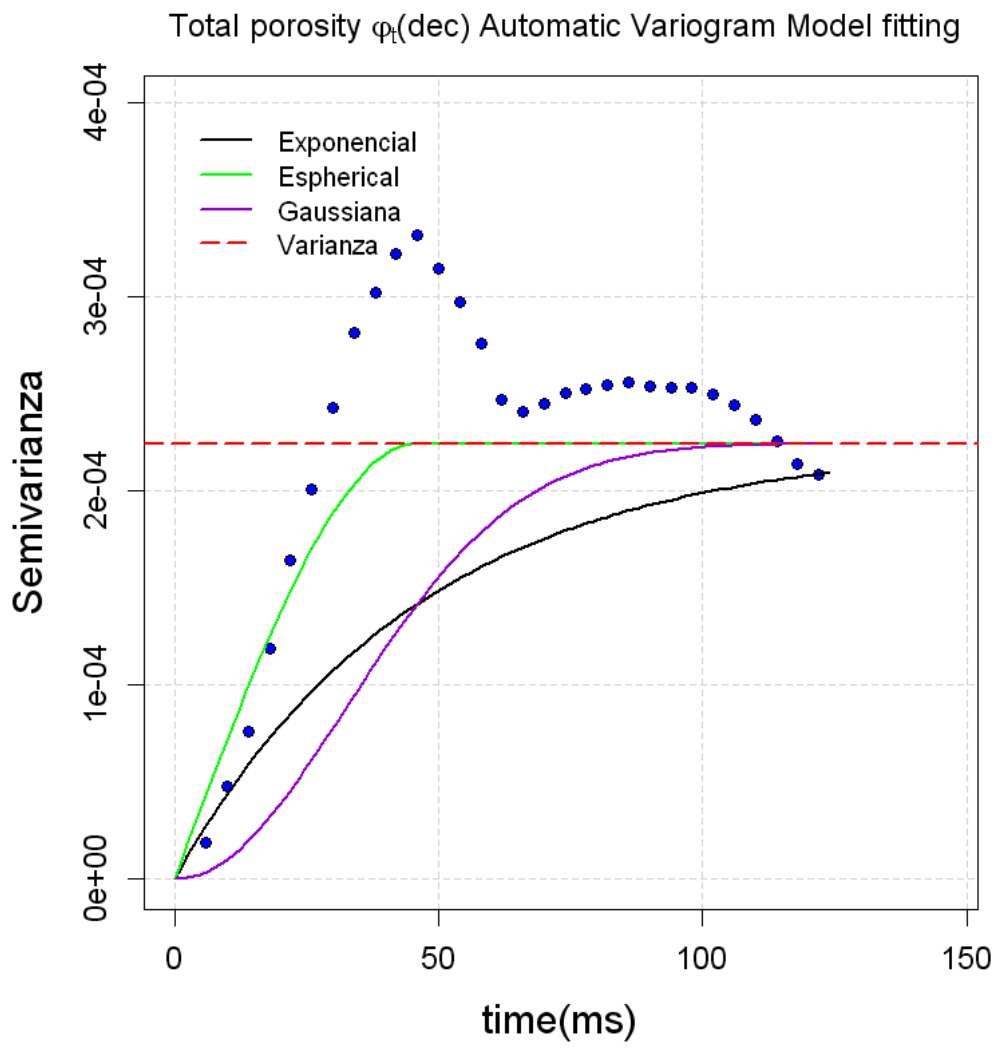
```
variog: computing variogram for direction = 90 degrees (1.571 radians)
        tolerance angle = 0 degrees (0 radians)
```



The best fit variogram is estimated, the variogram models tested are exponential, spherical and Gaussian.

[126]:
```
Phit_SUB_AllModelVarioFit<-AllModel(time_like_SUB, YCoord,
                                    Phit_like_SUB_Detrended_1[,3], 0, 90, N_lags,␣
↪lag_value, 1,
                                    expression(paste(" Total porosity ",varphi[t],
                                                     "(dec) Automatic Variogram␣
↪Model fitting")),xlab="time(ms)")
```

```
variog: computing omnidirectional variogram
variofit: covariance model used is exponential
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is spherical
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is gaussian
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is exponential
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is spherical
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is gaussian
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is exponential
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is spherical
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is gaussian
variofit: weights used: npairs
variofit: minimisation function used: optim
```



[127]:
```
Phit_SUB_AllModelVarioFit
```

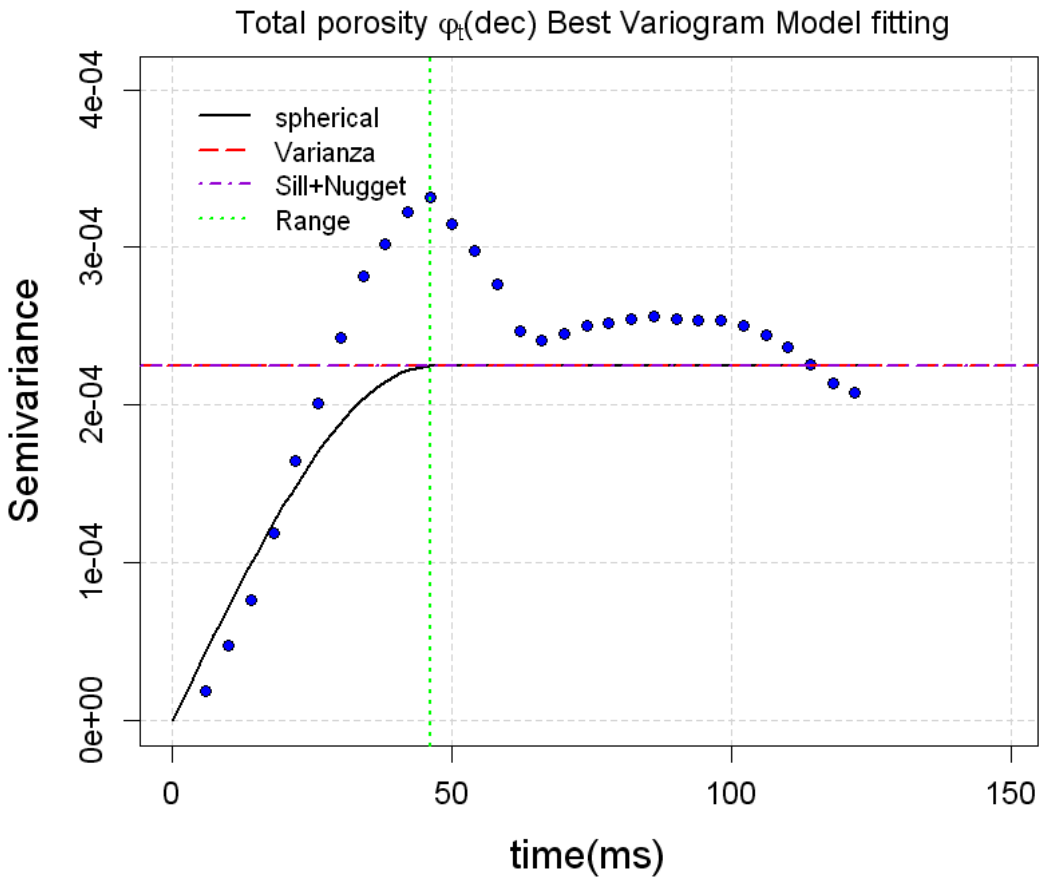
A matrix: 3 × 4 of type dbl

	Nugget	Meseta+Nugget	Alcance	SCE
exponential	0	0.0002245566	46	2.791987e-07
spherical	0	0.0002245566	46	6.591958e-08
gaussian	0	0.0002245566	46	2.891484e-07

[128]:
```
Phit_SUB_BestModelVarioFit<-BestModel(time_like_SUB, YCoord,
                                      Phit_like_SUB_Detrended_1[,3], 0, 90, N_lags,␣
↪lag_value, 1,
                                      expression(paste(" Total porosity␣
↪",varphi[t],
```

```
                                                    "(dec) Best Variogram Model␣
↪fitting")),xlab="time(ms)")
```

```
variog: computing omnidirectional variogram
variofit: covariance model used is exponential
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is spherical
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is gaussian
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is exponential
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is spherical
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is gaussian
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is exponential
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is spherical
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is gaussian
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is spherical
variofit: weights used: npairs
variofit: minimisation function used: optim
```



Model	Nugget	Sill+Nugget	Range	MSE
spherical	0.0000	0.0002	46.0000	0.0000

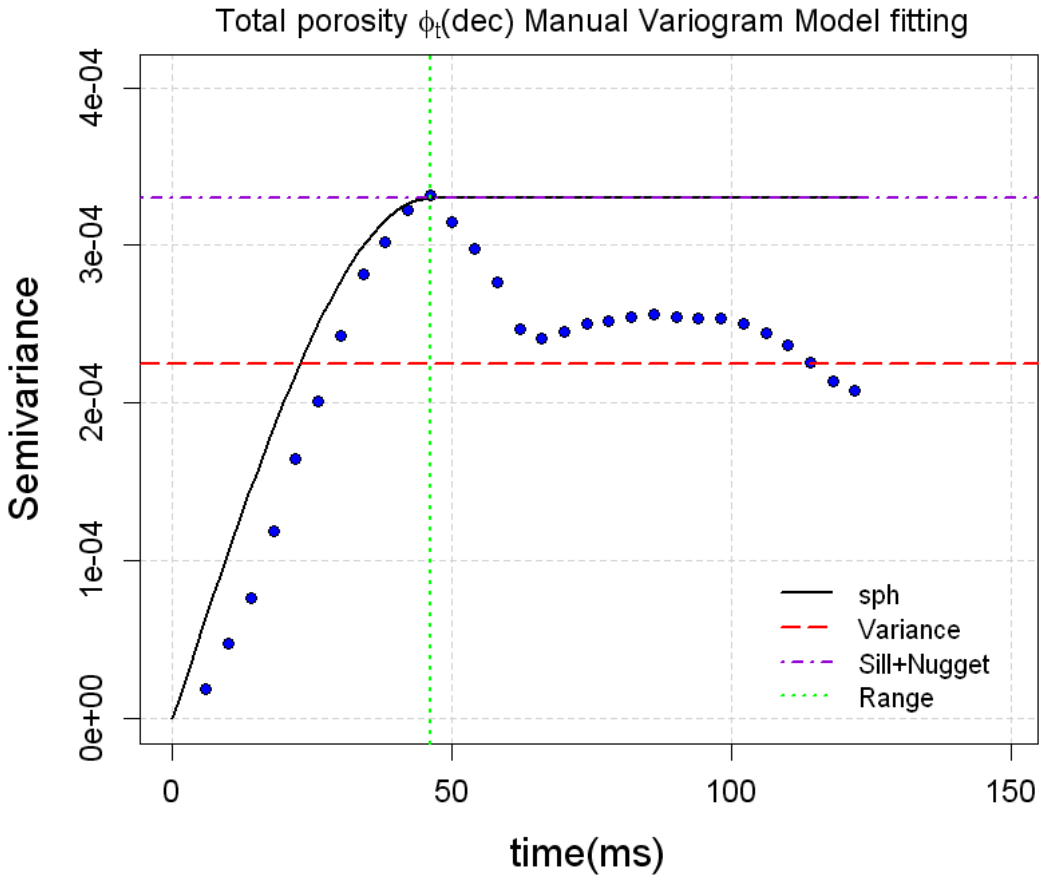
The best fit is expoential model, but the variogram model in the last case uses spherical model. for this case the variogram model is spherical due to comparatives betweeen both cases. The manual fit parameters are range 60 ms, nugget 0.0001 and sill with nugget 0.0006

[129]:
```
vario_model<- 2
nugget<- 0
sill_with_nugget<- 0.00033
range <- 46
```

[9]:
```
Phit_SUB_EyeModelVarioFit<-EyeModel(time_like_SUB, YCoord,
                                    Phit_like_SUB_Detrended_1[,3], 0, 90, N_lags,␣
↪lag_value, 1,
                                    vario_model, nugget, sill_with_nugget, range,
```

```
                                    expression(paste(" Total porosity ",phi[t],
                                                     "(dec) Manual Variogram Model␣
↪fitting")),xlab="time(ms)")
```

```
variog: computing omnidirectional variogram
```



Model	Nugget	Sill+Nugget	Range	MSE
sph	0.0000	0.0003	46.0000	0.0000

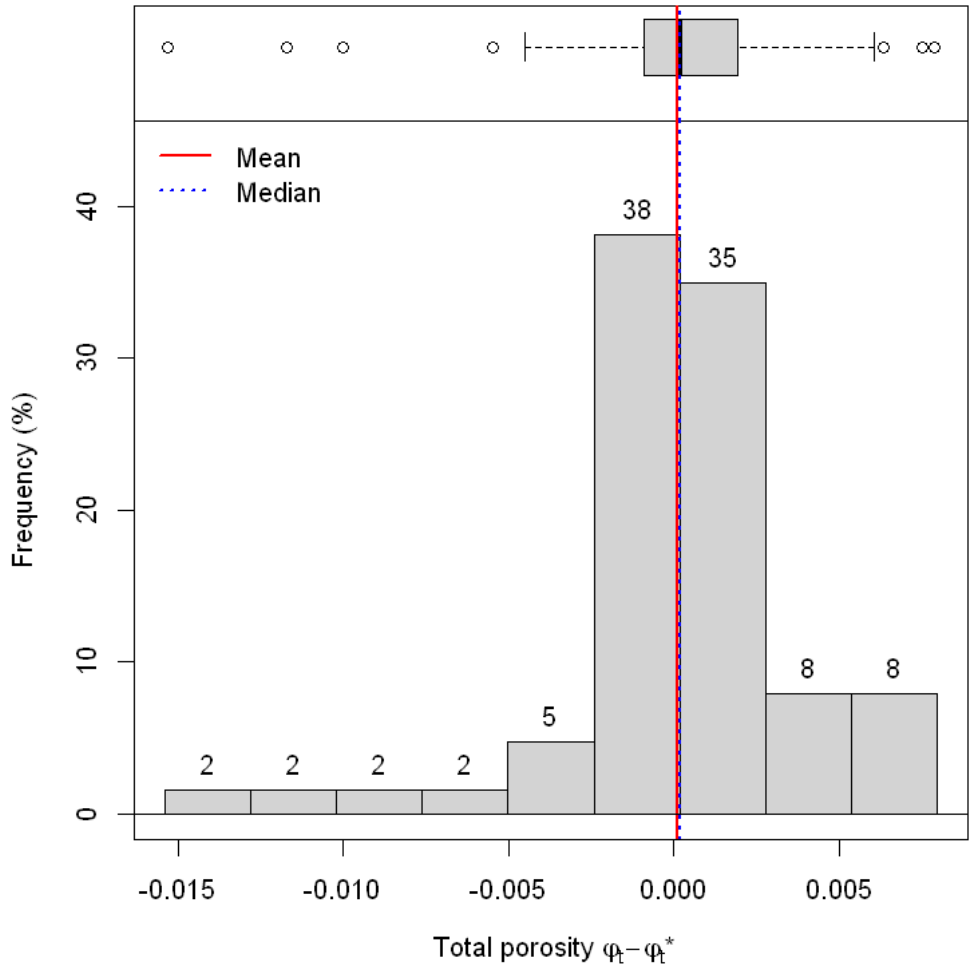
[131]:
```
Phit_SUB_CrossValid<- CrossValidation(time_like_SUB, YCoord,
                                      Phit_like_SUB_Detrended_1[,3], vario_model,␣
↪nugget, sill_with_nugget,
                                    range, MaxAnis=0, proporcion=1)
```

[132]:
```
Phit_SUB_CrossValid_Stat<- Val_Estadisticos(Phit_SUB_CrossValid[1:
↪length(Phit_like_SUB),c(3,4,5)])
```

```
Phit_SUB_CrossValid_Stat
```

	Z <dbl>	Z* <dbl>	Z-Z* <dbl>
No_muestras	63.00000	63.00000	63.00000
Minimo	-0.04169	-0.03735	-0.01530
Cuartil_1er	-0.00488	-0.00552	-0.00091
Mediana	0.00136	0.00141	0.00018
Media	0.00000	-0.00012	0.00012
Cuartil_3er	0.01269	0.01086	0.00194
Maximo	0.01986	0.01832	0.00786
Rango	0.06155	0.05567	0.02316
Rango_Intercuartil	0.01757	0.01638	0.00285
Varianza	0.00022	0.00020	0.00001
Desv_Estandar	0.01499	0.01425	0.00385
Simetria	-0.94133	-0.85494	-1.35406
Curtosis	3.52252	3.22126	7.46389

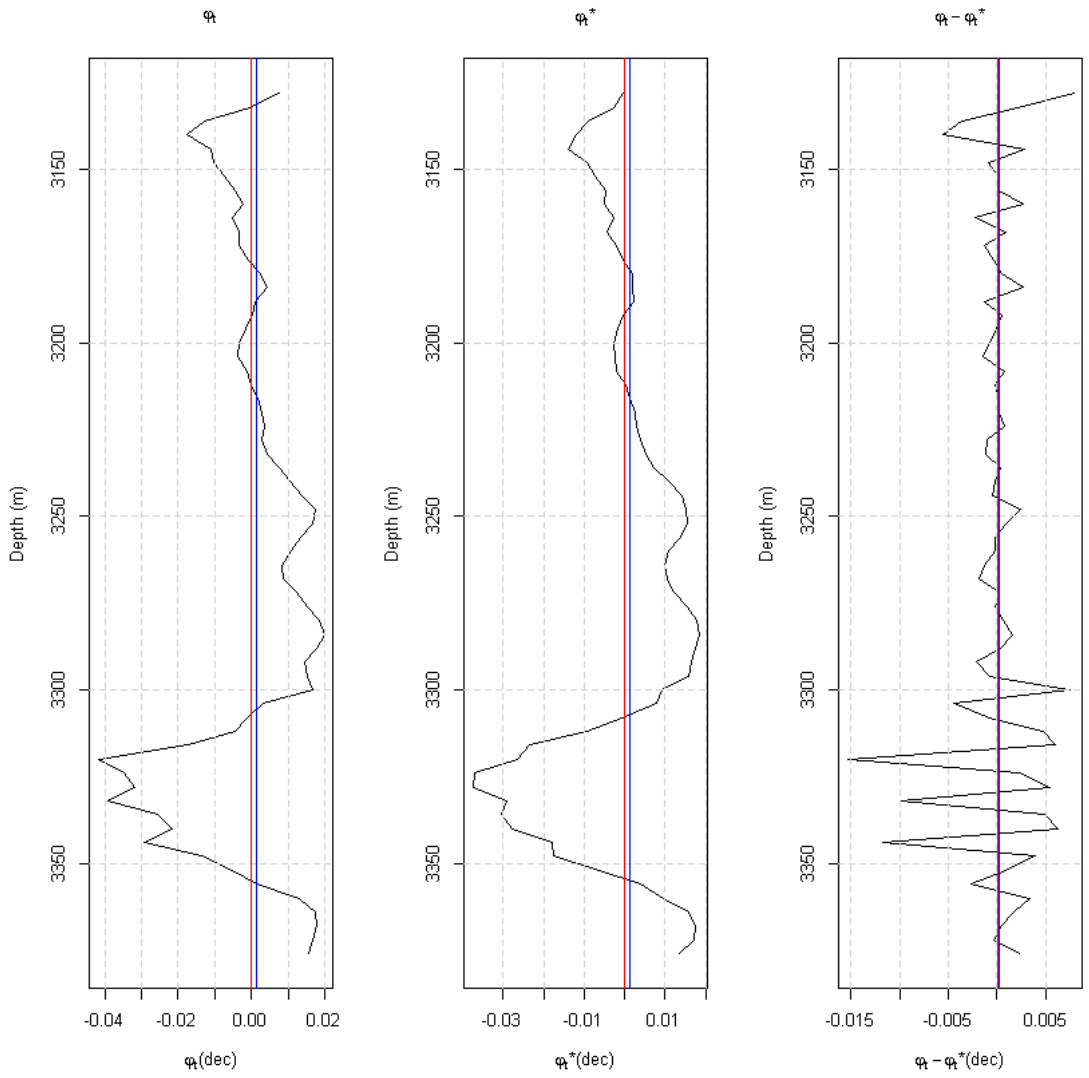
[133]:
```
HistBoxplot(x=Phit_SUB_CrossValid[,5], mean = Phit_SUB_CrossValid_Stat[5,3],
            median = Phit_SUB_CrossValid_Stat[4,3], main ="",
            xlab = expression(paste(" Total porosity␣
 ↪",varphi[t]-varphi[t],"*")), ylab = "Frequency (%)",
            AbsFreq = FALSE, PercentFreq = TRUE )
```



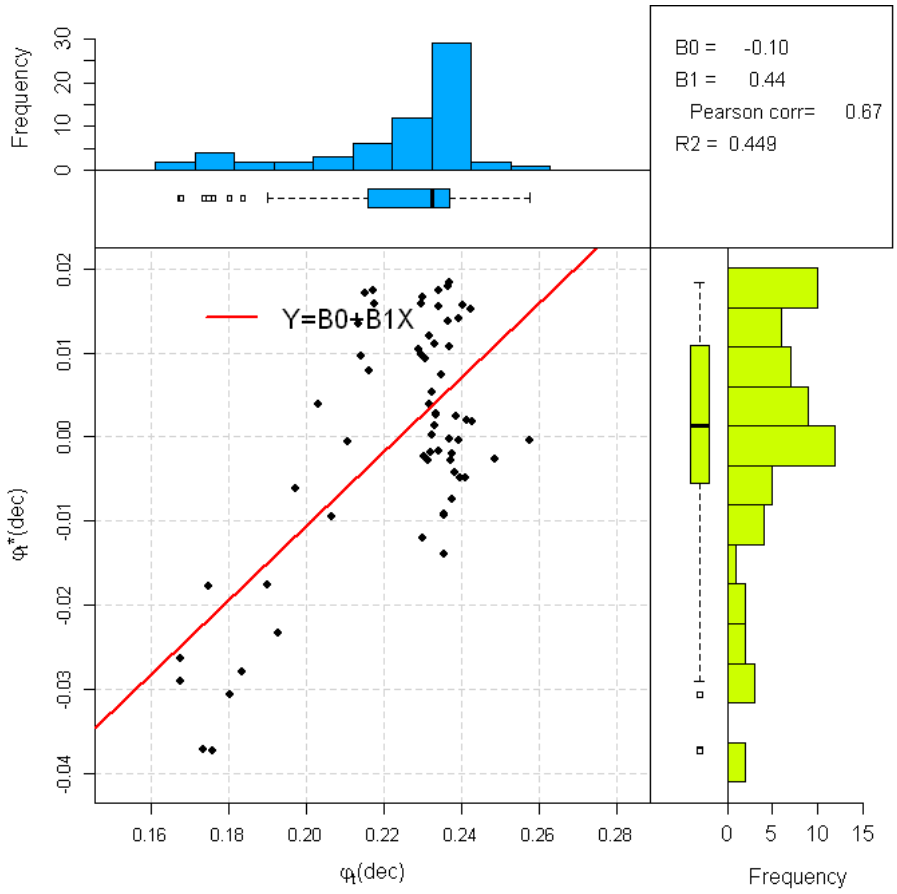
[134]:
```
Phit_SUB_est<-Phit_SUB_CrossValid[,4]
Phit_SUB_est_Stat<-Estadisticas(Phit_SUB_est)
Phit_SUB_diff<-Phit_SUB_CrossValid[,5]
Phit_SUB_diff_Stat<-Estadisticas(Phit_SUB_diff)
```

[137]:
```
LogPlot(depth=time_like_SUB, n_logs=3,
        log_list=list(Phit_like_SUB_Detrended_1[,3],Phit_SUB_est,Phit_SUB_diff),
       ␣
↪log_name_list=list(expression(paste(varphi[t])),expression(paste(varphi[t],"*")),expression(p
↪
       ␣
↪log_xlabel_list=list(expression(paste(varphi[t],"(dec)")),expression(paste(varphi[t],"*(dec)'
```

```
        plot_mean=TRUE, plot_median=TRUE)
```



[136]:
```
scaterplotReg(Phit_like_SUB , Phit_SUB_est, ns,
              Xmin = Phit_like_SUB_Stat[2,2], Xmax = Phit_like_SUB_Stat[7,2],
              Ymin = Phit_SUB_est_Stat[2,2],Ymax = Phit_SUB_est_Stat[7,2],
              XLAB = expression(paste(varphi[t],"(dec)")), YLAB =␣
↪expression(paste(varphi[t],"*(dec)")))
```



9.2 Parameter estimation using acoustic impedance at seismic scale

Whit this new information is necessary to estimate the posterior parameter values, the conditions to make this estimation are the same: For the search function, the use of the normal function is considered for all parameters. the mean value was established using the values of the a priori parameters and the standard deviation was selected using as reference the percentage of the a priori value in such a way that its value allows the probability of acceptance of the random walk metropolis-Hastings method to be within the interval 20% to 30%. The number of iterations is 10,000.

parameters	Prior value	Standard deviation (%)
Lognormal (μ)	8.79	1.7
Lognormal (σ)	0.188	1.7

parameters	Prior value	Standard deviation (%)
Weibull (α)	8.96	1.7
Weibull (λ)	0.227	1.7
Frank copula (θ)	-14.992	2.3

[125]:
```
#lognormal
var_mulog_post_5TRA<-(0.005752746)^2
var_sigmalog_post_5TRA<-(0.004067304)^2
#weibull
var_alpha_post_5TRA<-(0.129598464)^2 #shape
var_lambda_post_5TRA<- (0.001367821)^2 #scale

var_theta_post_5TRA<-(0.433)^2 #frank itau
```

[126]:
```
v_tot_post_5TRA<-diag(c(var_theta_post_5TRA,var_mulog_post_5TRA,var_sigmalog_post_5TRA,
                        var_alpha_post_5TRA,var_lambda_post_5TRA))

start_5TRA=array(c(-15,8.8,0.15,5,0.2),c(1,5))
```

[72]:
```
logpost_5TRA <- function(theta, data){
  par_cop <-theta[1]
  mulog <- theta[2]
  sigmalog <- theta[3]
  alpha <- theta[4]
  lambda <- theta[5]

  theta_prior=dnorm(par_cop, mean=-14.992, sd=((14.992/100)*2.3), log=TRUE)
  mulog_prior=dnorm(mulog, mean=8.79, sd=((8.79/100)*1.7), log=TRUE)
  sigmalog_prior=dnorm(sigmalog, mean=0.188, sd=((0.188/100)*1.7), log=TRUE)
  alpha_prior=dnorm(alpha, mean=8.96, sd=((8.96/100)*1.7), log=TRUE)
  lambda_prior=dnorm(lambda, mean=0.227, sd=((0.227/100)*1.7), log=TRUE)



  u_like<-dlnorm(data[,1],mulog,sigmalog, log = TRUE)
  v_like<-dweibull(data[,2],alpha,lambda, log = TRUE)
  cop_like<-dCopula(pobs(data),frankCopula(param = par_cop), log = TRUE)

  #␣
 ↪theta_prior+mulog_prior+sigmalog_prior+alpha_prior+lambda_prior+cop_like+u_like+v_like
  mulog_prior+sigmalog_prior+alpha_prior+lambda_prior+cop_like+u_like+v_like
  #mulog_prior+sigmalog_prior+lambda_prior+theta_prior+cop_like+u_like+v_like
}
```

[ ]:
```
mcmc_5TRA.fit <-rwmetrop(logpost_5TRA,
                    list(var=v_tot_post_5TRA,scale=2),
```

```
                    start_5TRA,
                    10000,
                    Data_like_TRA)
```

[74]:
```
mcmc_5TRA.fit$accept
```

0.2523

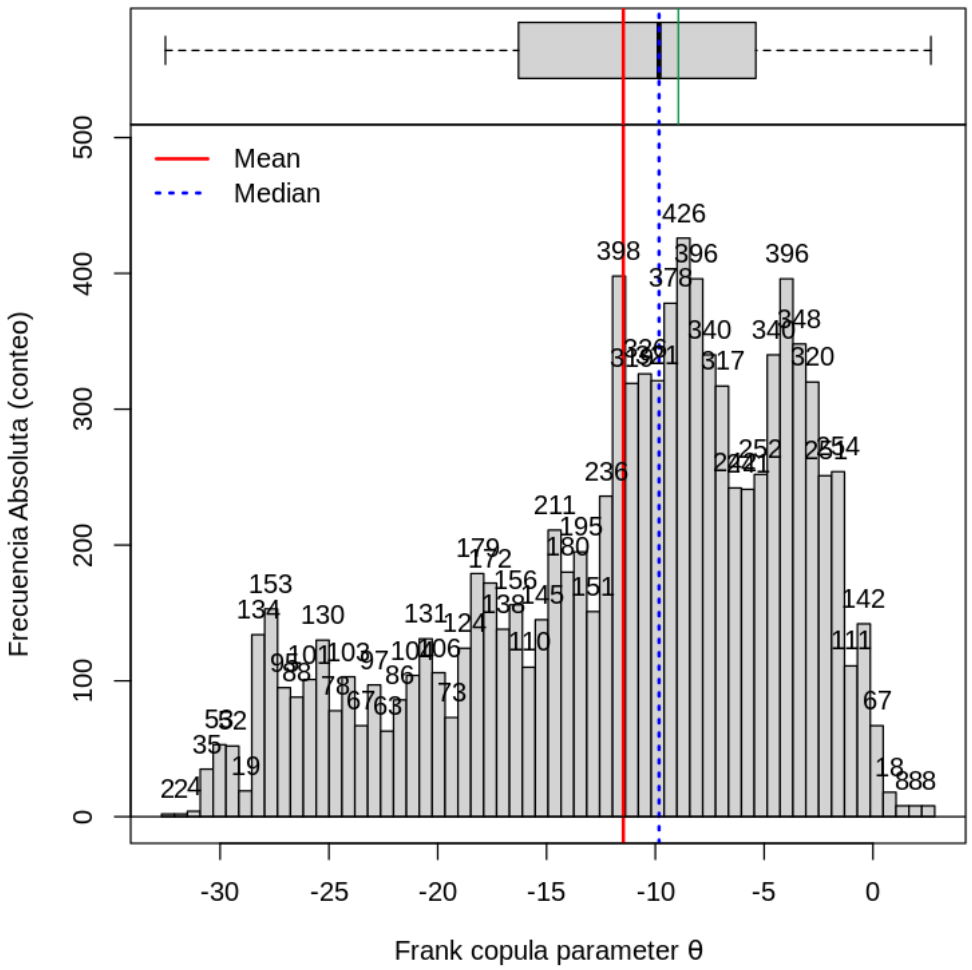
[75]:
```
frank_theta_post_5TRA<-Estadisticas(mcmc_5TRA.fit$par[,1])
lnorm_mu_post_5TRA<-Estadisticas(mcmc_5TRA.fit$par[,2])
lnorm_sigma_post_5TRA<-Estadisticas(mcmc_5TRA.fit$par[,3])
weibull_alpha_post_5TRA<-Estadisticas(mcmc_5TRA.fit$par[,4])
weibull_lambda_post_5TRA<-Estadisticas(mcmc_5TRA.fit$par[,5])
```

The acceptance probability is 25.23%. The posterior value for parameter θ is -8.944

[77]:
```
theta_best_post_5TRA=mfv(round(mcmc_5TRA.fit$par[,1],3))
HistBoxplot(mcmc_5TRA.fit$par[,1],nbins = 60, mean = frank_theta_post_5TRA[5,2],
            median = frank_theta_post_5TRA[4,2], main ="",
            xlab = expression(paste("Frank copula parameter ",theta )), ylab =␣
 ↪"Frecuencia Absoluta (conteo)",
            AbsFreq = TRUE, PercentFreq = FALSE )
par(new=TRUE)
abline(v=theta_best_post_5TRA, col=("springgreen4"))
frank_theta_post_5TRA
print(paste('The Frank copula parameter theta is',theta_best_post_5TRA))
```

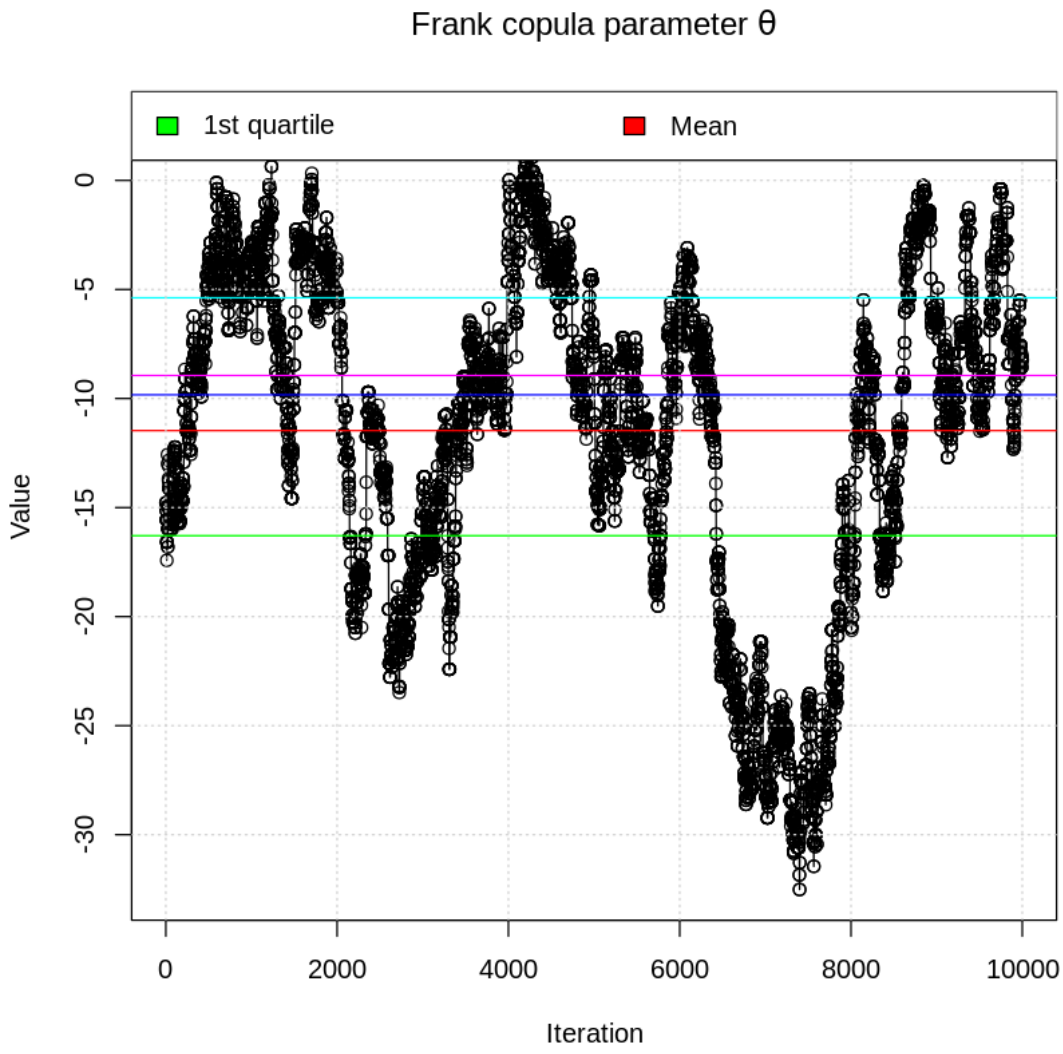
	Statistics <chr>	Values <dbl>
muestras	n	10000.0000
minimos	Minimum	-32.5133
cuantiles1	1st. Quartile	-16.2892
medianas	Median	-9.8286
medias	Mean	-11.4699
cuantiles3	3rd. Quartile	-5.3819
maximos	Maximum	2.6623
rangos	Rank	35.1756
rangosInt	Interquartile Rank	10.9074
varianzas	Variance	58.6180
desvs	Standard Deviation	7.6562
CVs	Variation Coeff.	-0.6675
simetrias	Skewness	-0.6932
curtosiss	Kurtosis	2.6184

```
[1] "The Frank copula parameter theta is -8.944"
```



[78]:
```
plot(mcmc_5TRA.fit$par[,1], xlab = "Iteration", ylab = "Value", type = "o",
     main =expression(paste("Frank copula parameter ",theta )))
par(new=TRUE)
abline(h=frank_theta_post_5TRA[3,2],col=("green"))
par(new=TRUE)
abline(h=frank_theta_post_5TRA[4,2],col=("blue"))
par(new=TRUE)
abline(h=frank_theta_post_5TRA[5,2],col=("red"))
par(new=TRUE)
abline(h=frank_theta_post_5TRA[6,2],col=("cyan"))
par(new=TRUE)
abline(h=theta_best_post_5TRA,col=("magenta"))
grid()
```

```
legend(x = "topleft", legend = c("1st quartile","Mean","Median","3rd quartile",
                                 expression(paste("Best Frank copula parameter␣
 ↪",theta ))),
       fill = c("green","red","blue","cyan","magenta"), horiz=TRUE)
```



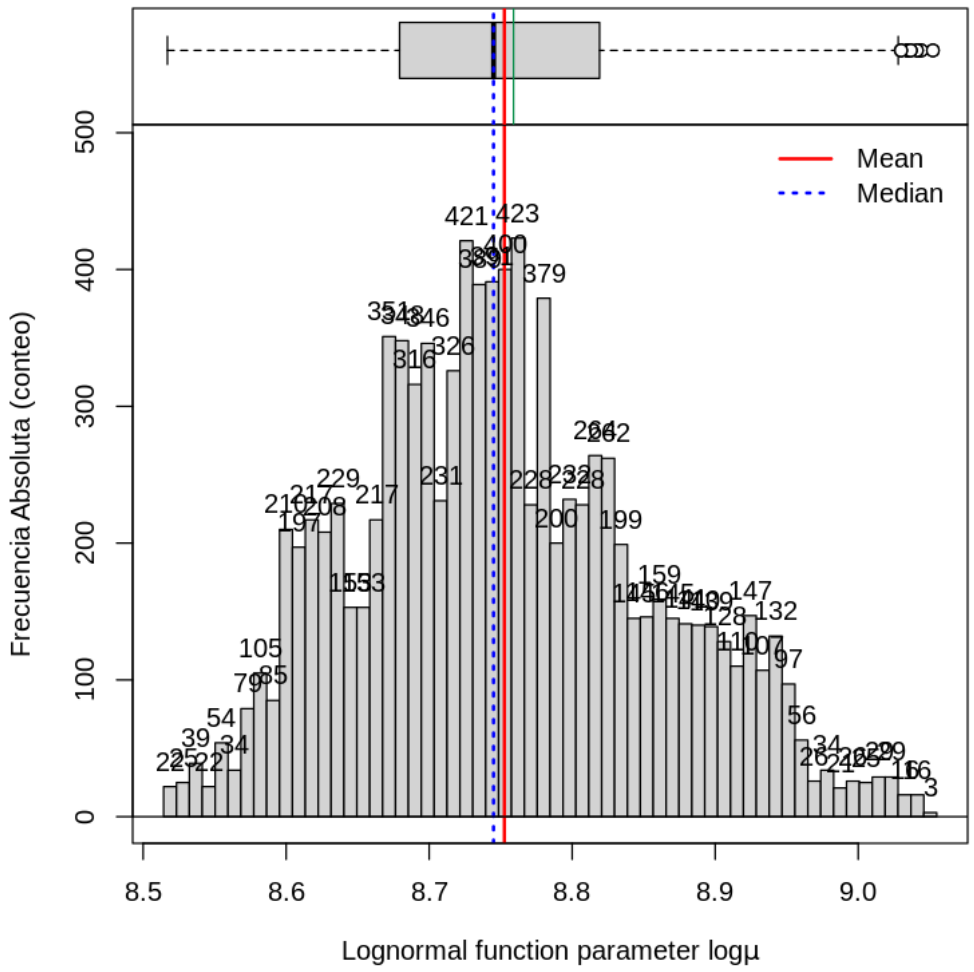
The posterior lognormal parameter μ is 8.759

[110]:
```
mu_best_post_5TRA=8.759#mfv(round(mcmc_5TRA.fit$par[,2],3))
HistBoxplot(mcmc_5TRA.fit$par[,2],nbins = 60, mean = lnorm_mu_post_5TRA[5,2],
            median = lnorm_mu_post_5TRA[4,2], main ="",
            xlab = expression(paste("Lognormal function parameter log",mu )),
            ylab = "Frecuencia Absoluta (conteo)", AbsFreq = TRUE, PercentFreq =␣
 ↪FALSE )
```

```
par(new=TRUE)
abline(v=mu_best_post_5TRA, col=("springgreen4"))

lnorm_mu_post_5TRA
print(paste('The Lognormal density function parameter mu is',mu_best_post_5TRA))
```

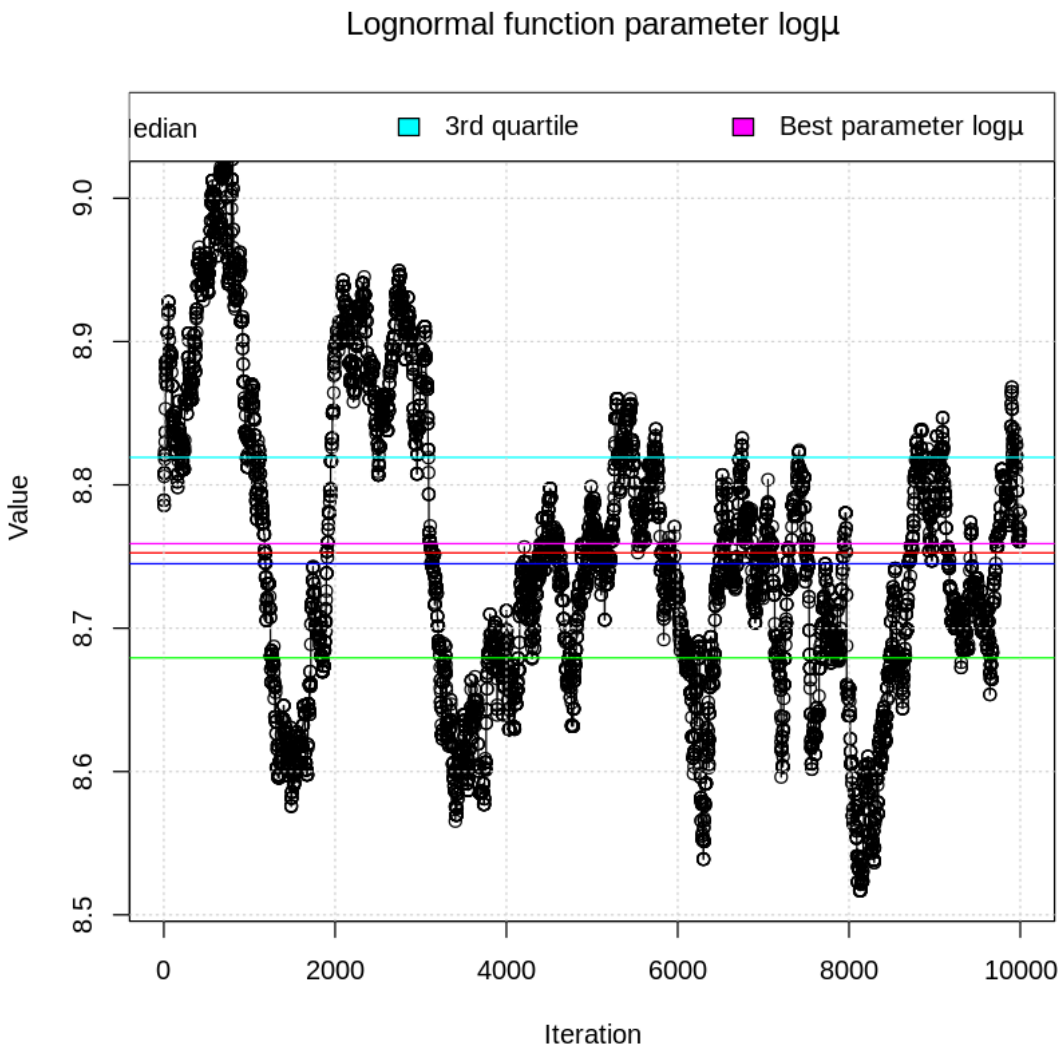
	Statistics <chr>	Values <dbl>
muestras	n	10000.0000
minimos	Minimum	8.5170
cuantiles1	1st. Quartile	8.6793
medianas	Median	8.7450
medias	Mean	8.7526
cuantiles3	3rd. Quartile	8.8192
maximos	Maximum	9.0522
rangos	Rank	0.5353
rangosInt	Interquartile Rank	0.1398
varianzas	Variance	0.0107
desvs	Standard Deviation	0.1035
CVs	Variation Coeff.	0.0118
simetrias	Skewness	0.3119
curtosiss	Kurtosis	2.6445

```
[1] "The Lognormal density function parameter mu is 8.759"
```



[111]:
```
plot(mcmc_5TRA.fit$par[,2], xlab = "Iteration", ylab = "Value", type = "o",
     main =expression(paste("Lognormal function parameter log",mu )))
par(new=TRUE)
abline(h=lnorm_mu_post_5TRA[3,2],col=("green"))
par(new=TRUE)
abline(h=lnorm_mu_post_5TRA[4,2],col=("blue"))
par(new=TRUE)
abline(h=lnorm_mu_post_5TRA[5,2],col=("red"))
par(new=TRUE)
abline(h=lnorm_mu_post_5TRA[6,2],col=("cyan"))
par(new=TRUE)
abline(h=mu_best_post_5TRA,col=("magenta"))
grid()
```

```
legend(x = "topright", legend = c("1st quartile","Mean","Median","3rd quartile",
                                  expression(paste("Best parameter log",mu ))),
       fill = c("green","red","blue","cyan","magenta"), horiz=TRUE)
```



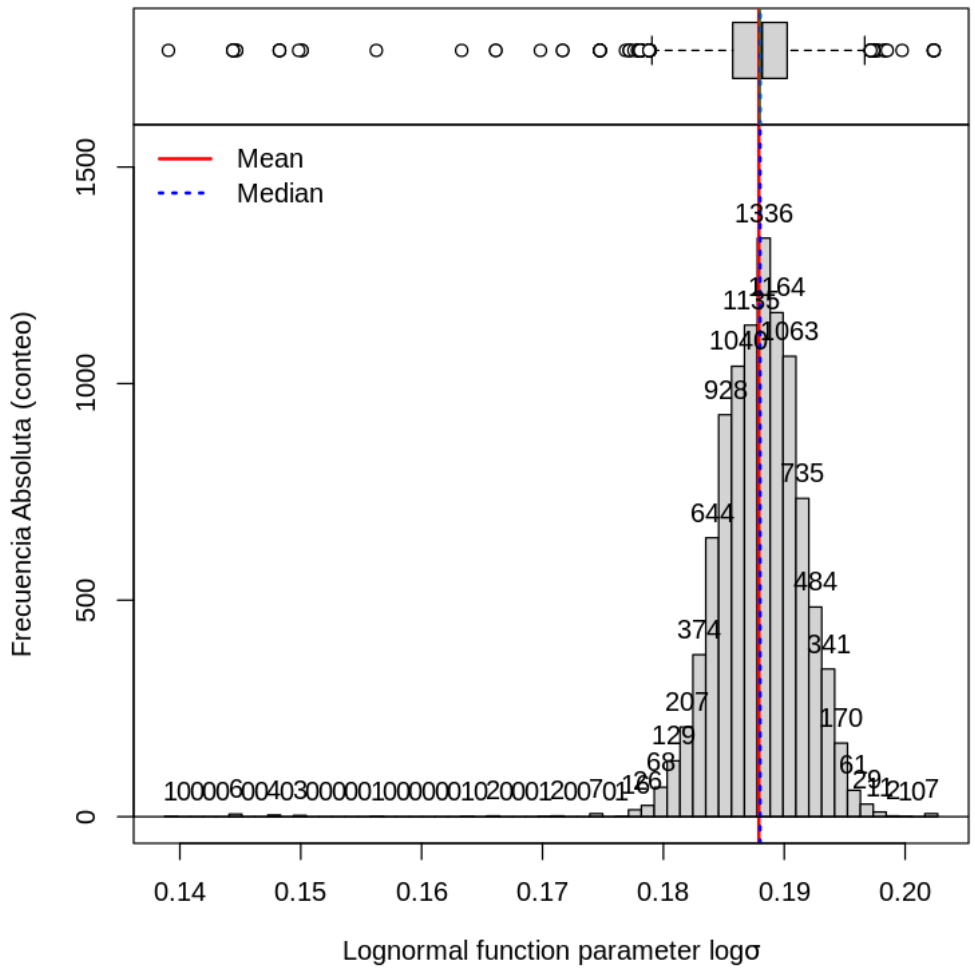
The posterior lognormal parameter σ is 0.188

[81]:
```
sigma_best_post_5TRA=mfv(round(mcmc_5TRA.fit$par[,3],3))
HistBoxplot(mcmc_5TRA.fit$par[,3],nbins = 60, mean = lnorm_sigma_post_5TRA[5,2],␣
↪median = lnorm_sigma_post_5TRA[4,2],
            main ="",
            xlab = expression(paste("Lognormal function parameter log",sigma )),
            ylab = "Frecuencia Absoluta (conteo)",
            AbsFreq = TRUE, PercentFreq = FALSE )
```

```
par(new=TRUE)
abline(v=sigma_best_post_5TRA, col=("springgreen4"))
lnorm_sigma_post_5TRA
print(paste('The Lognormal density function parameter sigma␣
 ↪is',sigma_best_post_5TRA))
```

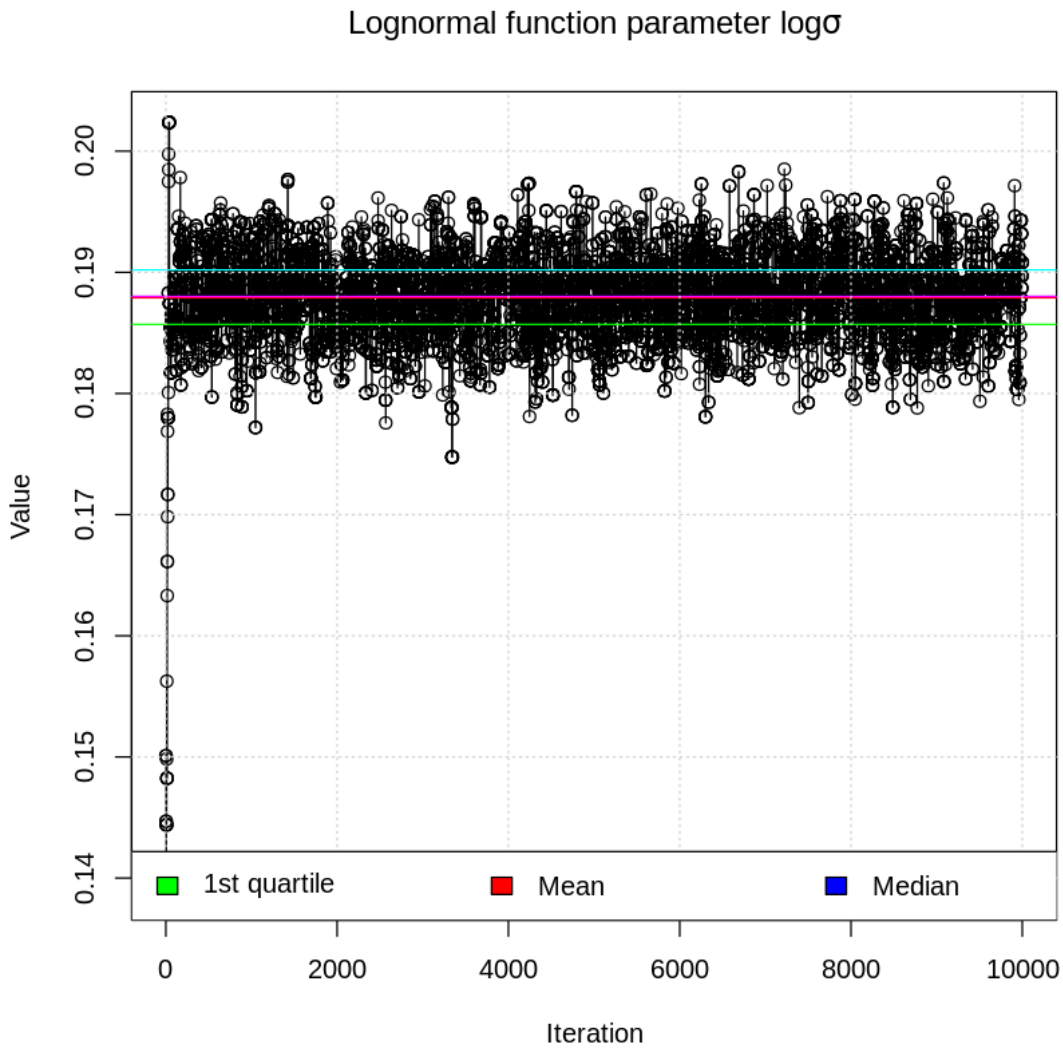
	Statistics <chr>	Values <dbl>
muestras	n	10000.0000
minimos	Minimum	0.1391
cuantiles1	1st. Quartile	0.1857
medianas	Median	0.1880
medias	Mean	0.1879
cuantiles3	3rd. Quartile	0.1902
maximos	Maximum	0.2024
rangos	Rank	0.0633
rangosInt	Interquartile Rank	0.0045
varianzas	Variance	0.0000
desvs	Standard Deviation	0.0037
CVs	Variation Coeff.	0.0197
simetrias	Skewness	-2.1504
curtosiss	Kurtosis	25.9154

```
[1] "The Lognormal density function parameter sigma is 0.188"
```



[82]:
```
plot(mcmc_5TRA.fit$par[,3], xlab = "Iteration", ylab = "Value", type = "o",
     main =expression(paste("Lognormal function parameter log",sigma )))
par(new=TRUE)
abline(h=lnorm_sigma_post_5TRA[3,2],col=("green"))
par(new=TRUE)
abline(h=lnorm_sigma_post_5TRA[4,2],col=("blue"))
par(new=TRUE)
abline(h=lnorm_sigma_post_5TRA[5,2],col=("red"))
par(new=TRUE)
abline(h=lnorm_sigma_post_5TRA[6,2],col=("cyan"))
par(new=TRUE)
abline(h=sigma_best_post_5TRA,col=("magenta"))
grid()
```

```
legend(x = "bottomleft", legend = c("1st quartile","Mean","Median","3rd␣
 ↪quartile",
                                    expression(paste("Best parameter log",mu ))),
       fill = c("green","red","blue","cyan","magenta"), horiz=TRUE)
```



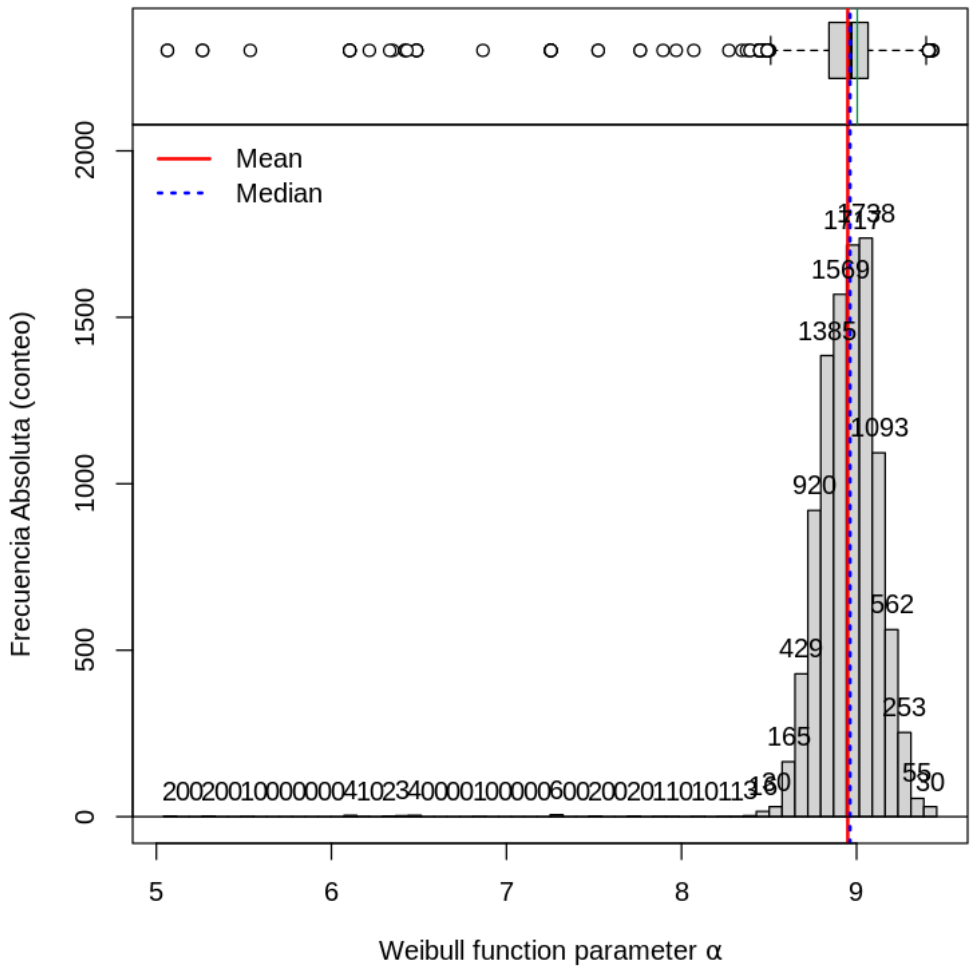
The posterior Weibull parameter α is 9.004

[83]:
```
alpha_best_post_5TRA=mfv(round(mcmc_5TRA.fit$par[,4],3))

HistBoxplot(mcmc_5TRA.fit$par[,4],nbins = 60, mean =␣
 ↪weibull_alpha_post_5TRA[5,2],
            median = weibull_alpha_post_5TRA[4,2],
            main ="",
```

```
            xlab = expression(paste("Weibull function parameter ",alpha )),
            ylab = "Frecuencia Absoluta (conteo)",
            AbsFreq = TRUE, PercentFreq = FALSE )
par(new=TRUE)
abline(v=alpha_best_post_5TRA, col=("springgreen4"))
weibull_alpha_post_5TRA
print(paste('The Weibull density function parameter alpha',alpha_best_post_5TRA))
```

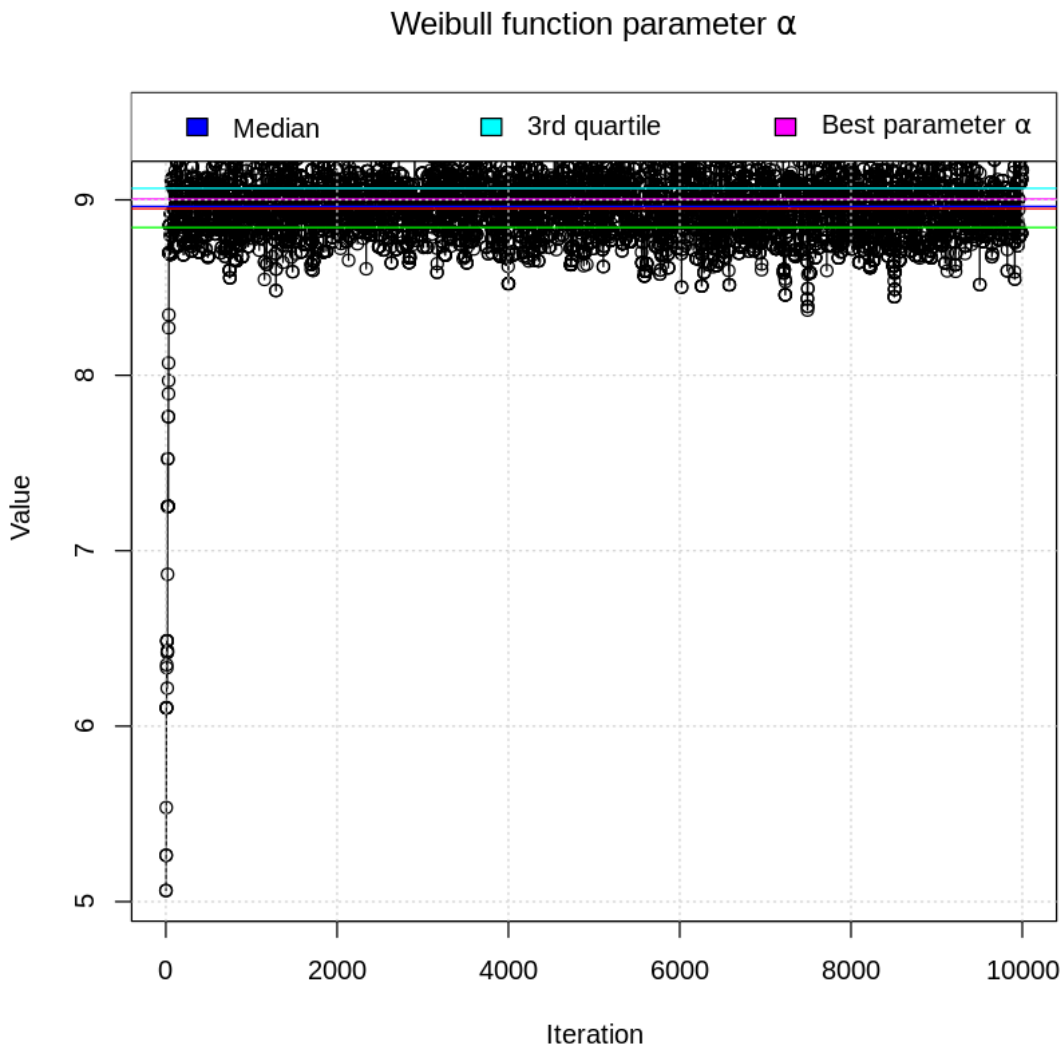
	Statistics <chr>	Values <dbl>
muestras	n	10000.0000
minimos	Minimum	5.0633
cuantiles1	1st. Quartile	8.8425
medianas	Median	8.9628
medias	Mean	8.9497
cuantiles3	3rd. Quartile	9.0655
maximos	Maximum	9.4360
rangos	Rank	4.3727
rangosInt	Interquartile Rank	0.2230
varianzas	Variance	0.0440
desvs	Standard Deviation	0.2097
CVs	Variation Coeff.	0.0234
simetrias	Skewness	-6.0786
curtosiss	Kurtosis	90.0645

```
[1] "The Weibull density function parameter alpha 9.004"
```



[84]:
```
plot(mcmc_5TRA.fit$par[,4], xlab = "Iteration", ylab = "Value", type = "o",
     main =expression(paste("Weibull function parameter ",alpha )))
par(new=TRUE)
abline(h=weibull_alpha_post_5TRA[3,2],col=("green"))
par(new=TRUE)
abline(h=weibull_alpha_post_5TRA[4,2],col=("blue"))
par(new=TRUE)
abline(h=weibull_alpha_post_5TRA[5,2],col=("red"))
par(new=TRUE)
abline(h=weibull_alpha_post_5TRA[6,2],col=("cyan"))
par(new=TRUE)
abline(h=alpha_best_post_5TRA,col=("magenta"))
grid()
```

```
legend(x = "topright", legend = c("1st quartile","Mean","Median","3rd quartile",
                                  expression(paste("Best parameter ",alpha ))),
       fill = c("green","red","blue","cyan","magenta"), horiz=TRUE)
```



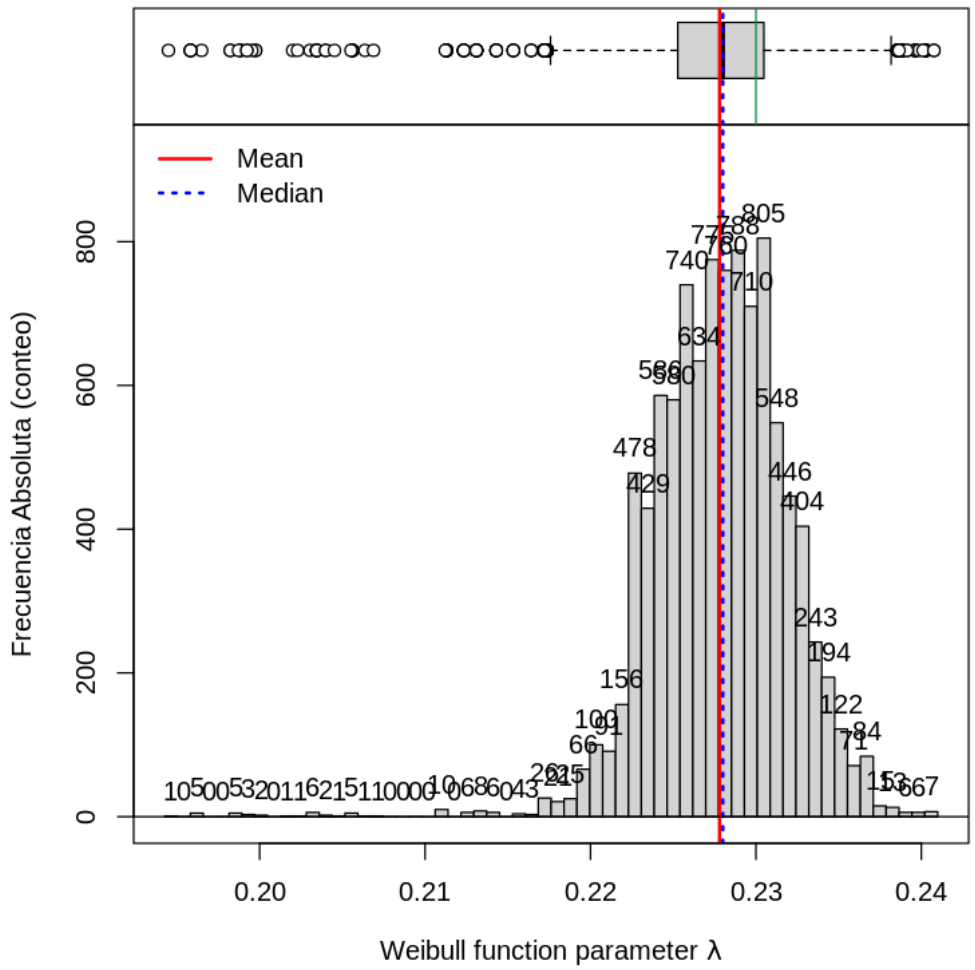
The posterior Weibull parameter λ is 0.23

[86]:
```
lambda_best_post_5TRA=mfv(round(mcmc_5TRA.fit$par[,5],3))
HistBoxplot(mcmc_5TRA.fit$par[,5],nbins = 60, mean =␣
↪weibull_lambda_post_5TRA[5,2],
            median = weibull_lambda_post_5TRA[4,2],
            main ="",
            xlab = expression(paste("Weibull function parameter ",lambda )),
            ylab = "Frecuencia Absoluta (conteo)",
```

```
            AbsFreq = TRUE, PercentFreq = FALSE )
par(new=TRUE)
abline(v=lambda_best_post_5TRA, col=("springgreen4"))
weibull_lambda_post_5TRA
print(paste('The Weibull density function parameter␣
 ↪lambda',lambda_best_post_5TRA))
```

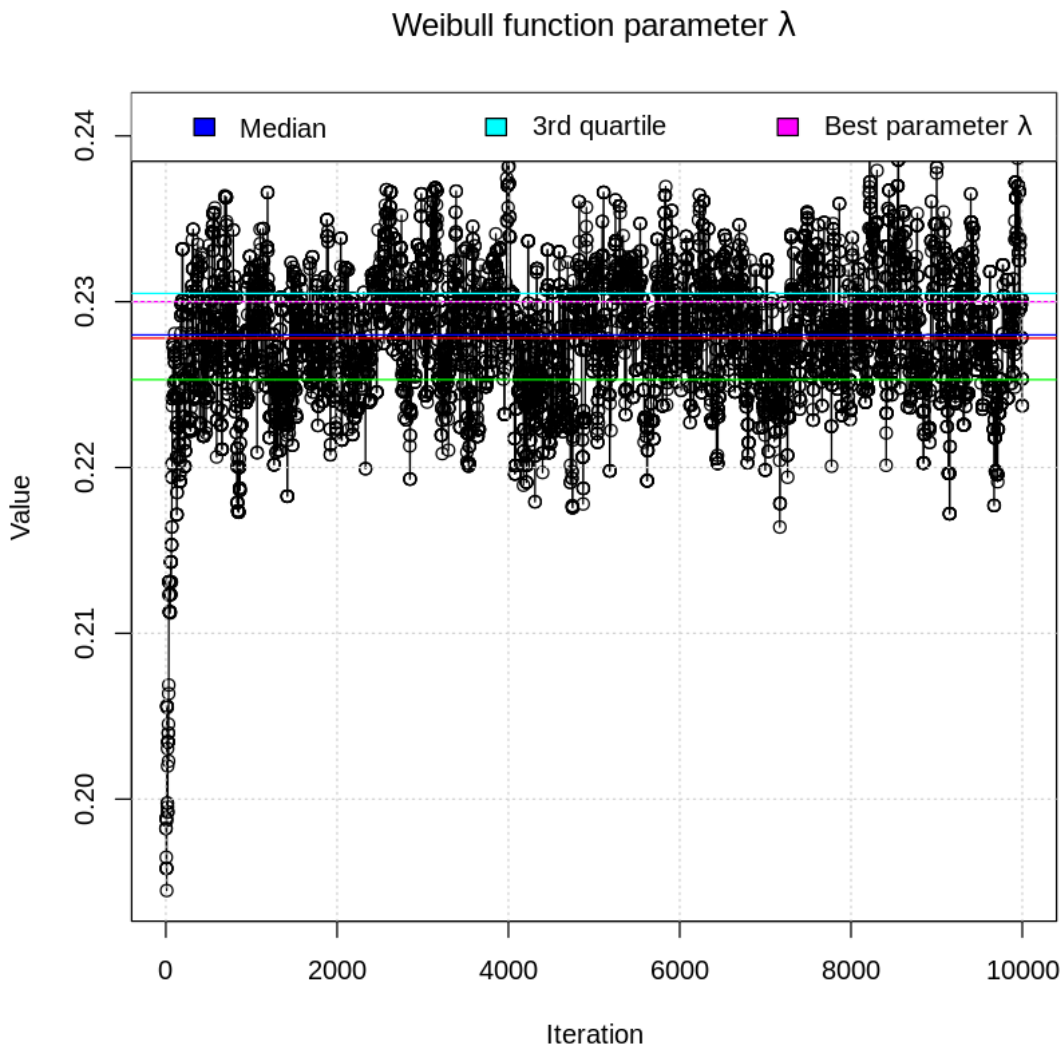
	Statistics <chr>	Values <dbl>
muestras	n	10000.0000
minimos	Minimum	0.1945
cuantiles1	1st. Quartile	0.2253
medianas	Median	0.2280
medias	Mean	0.2278
cuantiles3	3rd. Quartile	0.2305
maximos	Maximum	0.2408
rangos	Rank	0.0463
rangosInt	Interquartile Rank	0.0052
varianzas	Variance	0.0000
desvs	Standard Deviation	0.0041
CVs	Variation Coeff.	0.0180
simetrias	Skewness	-1.0346
curtosiss	Kurtosis	9.1167

```
[1] "The Weibull density function parameter lambda 0.23"
```



[87]:
```
plot(mcmc_5TRA.fit$par[,5], xlab = "Iteration", ylab = "Value", type = "o",
     main =expression(paste("Weibull function parameter ",lambda )))
par(new=TRUE)
abline(h=weibull_lambda_post_5TRA[3,2],col=("green"))
par(new=TRUE)
abline(h=weibull_lambda_post_5TRA[4,2],col=("blue"))
par(new=TRUE)
abline(h=weibull_lambda_post_5TRA[5,2],col=("red"))
par(new=TRUE)
abline(h=weibull_lambda_post_5TRA[6,2],col=("cyan"))
par(new=TRUE)
abline(h=lambda_best_post_5TRA,col=("magenta"))
grid()
```

```
legend(x = "topright", legend = c("1st quartile","Mean","Median","3rd quartile",
                                  expression(paste("Best parameter ",lambda ))),
       fill = c("green","red","blue","cyan","magenta"), horiz=TRUE)
```



[112]:
```
cop_frank_post_5TRA <- mvdc(frankCopula(theta_best_post_5TRA), margins =␣
↪c("lnorm","weibull"),
                        paramMargins = list(list(meanlog = mu_best_post_5TRA,␣
↪sdlog = sigma_best_post_5TRA),
                                            list(shape = alpha_best_post_5TRA,␣
↪scale = lambda_best_post_5TRA)))

cop_frank_post_ran_5TRA <- rMvdc(3696, mvdc = cop_frank_post_5TRA)
```

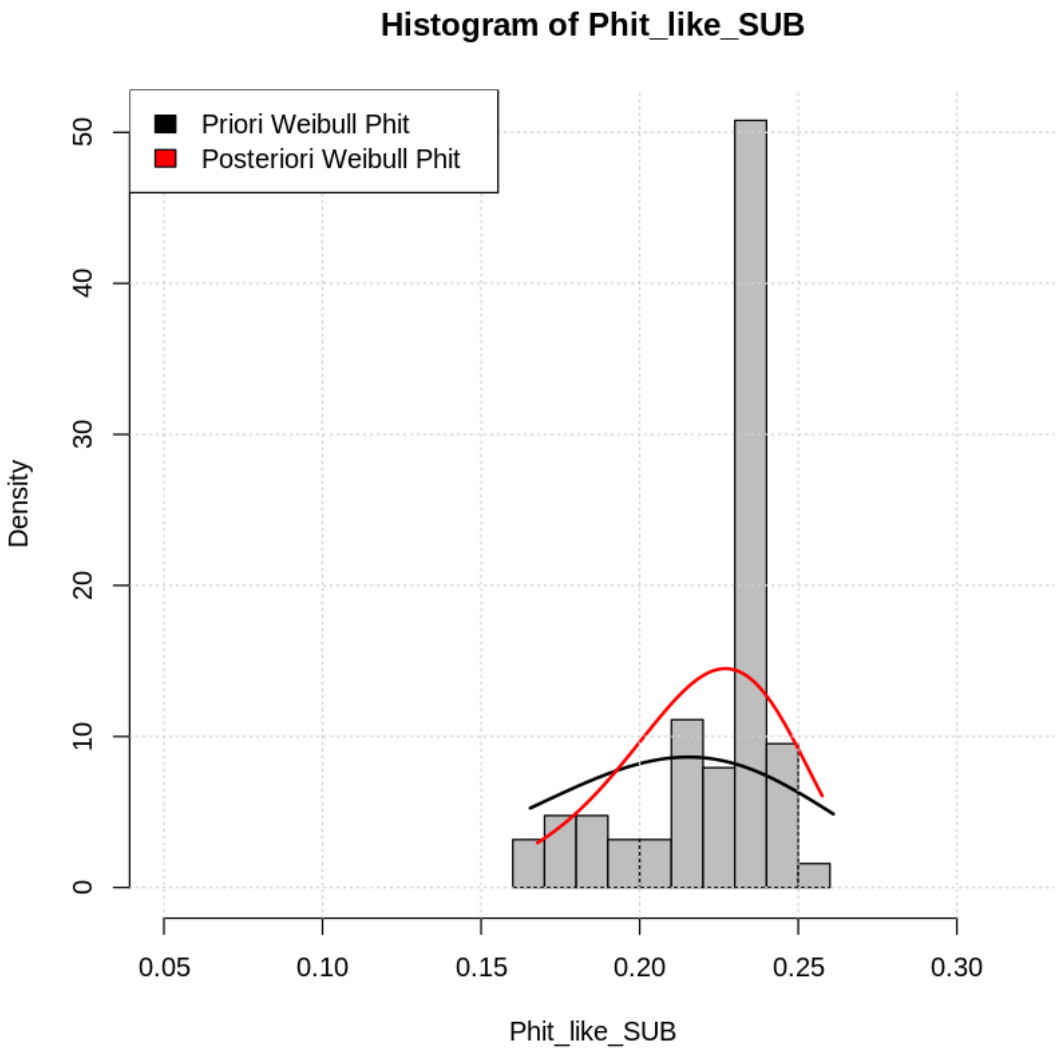
Comparing the prior and posterior values, all of them are very close but the parameter θ, the difference is 6.048.

parameters	Prior value	Posterior value	Difference
Lognormal (μ)	8.79	8.8759	0.0859
Lognormal (σ)	0.188	0.188	0
Weibull (α)	8.96	9.004	0.044
Weibull (λ)	0.227	0.23	0.003
Frank copula (θ)	-14.992	-8.944	6.048

[113]:
```
den.x <- seq(min(Phit_like),max(Phit_like),length=3696)
den.y_prior <- dweibull(den.x,shape = 5.1646292, scale = 0.2244496)

den.x_post_5TRA <- seq(min(Phit_like_SUB),max(Phit_like_SUB),length=3696)
den.y_post_5TRA <- dweibull(den.x_post_5TRA,shape = alpha_best_post_5TRA, scale␣
↪= lambda_best_post_5TRA)
```

Comparing the prior and posterior parameter values for weibull density function, the posterior density function moves to right.

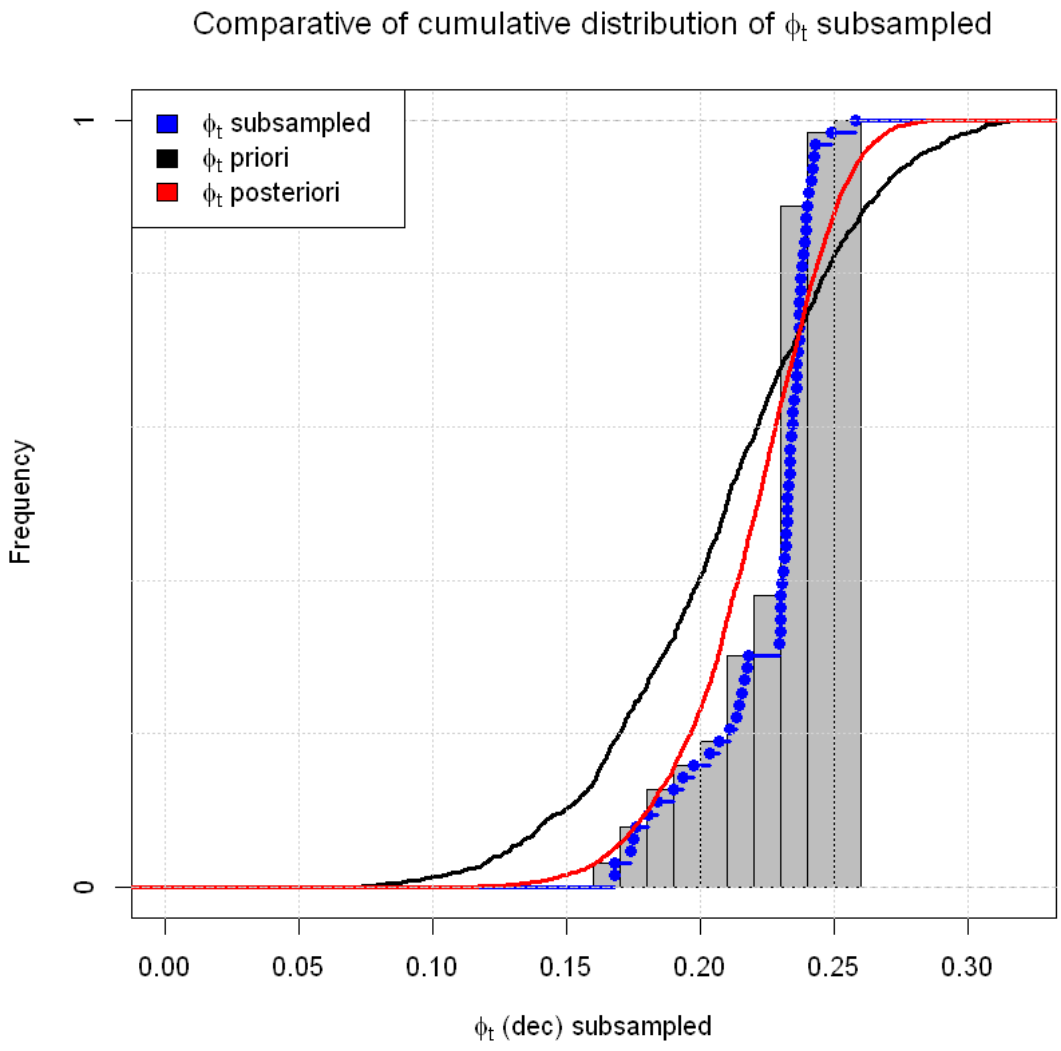
[114]:
```
hist(Phit_like_SUB, breaks=nsp, col="gray", probability=TRUE,xlim=c(0.05, 0.32))
lines(den.x, den.y_prior,xlim=c(0.05, 0.32), col="black", lwd=2)
lines(den.x_post_5TRA, den.y_post_5TRA,xlim=c(0.05, 0.32), col="red", lwd=2)
grid()
legend(x = "topleft", legend = c("Priori Weibull Phit", "Posteriori Weibull␣
↪Phit"),
       fill = c("black","red"))
```



The prior and posterior cumulative distribution function shows the posterior function is more close to subsample data.

[6]:
```
hcum <- h <- hist(Phit_like_SUB, plot=FALSE, breaks=nsp)
hcum$counts <- cumsum(hcum$counts)/sum(hcum$counts)
plot(hcum, xlim=c(0, 0.32), col=("gray"),main = expression(paste("Comparative of␣
 ↪cumulative distribution of ",phi[t]," subsampled")),
     xlab= expression(paste(phi[t]," (dec)"," subsampled")))
par(new=TRUE)
plot(ecdf(Phit_like_SUB), col=("blue"),xlim=c(0, 0.
 ↪32),lwd=3,xaxt='n',yaxt='n',ann=FALSE )
par(new=TRUE)
```

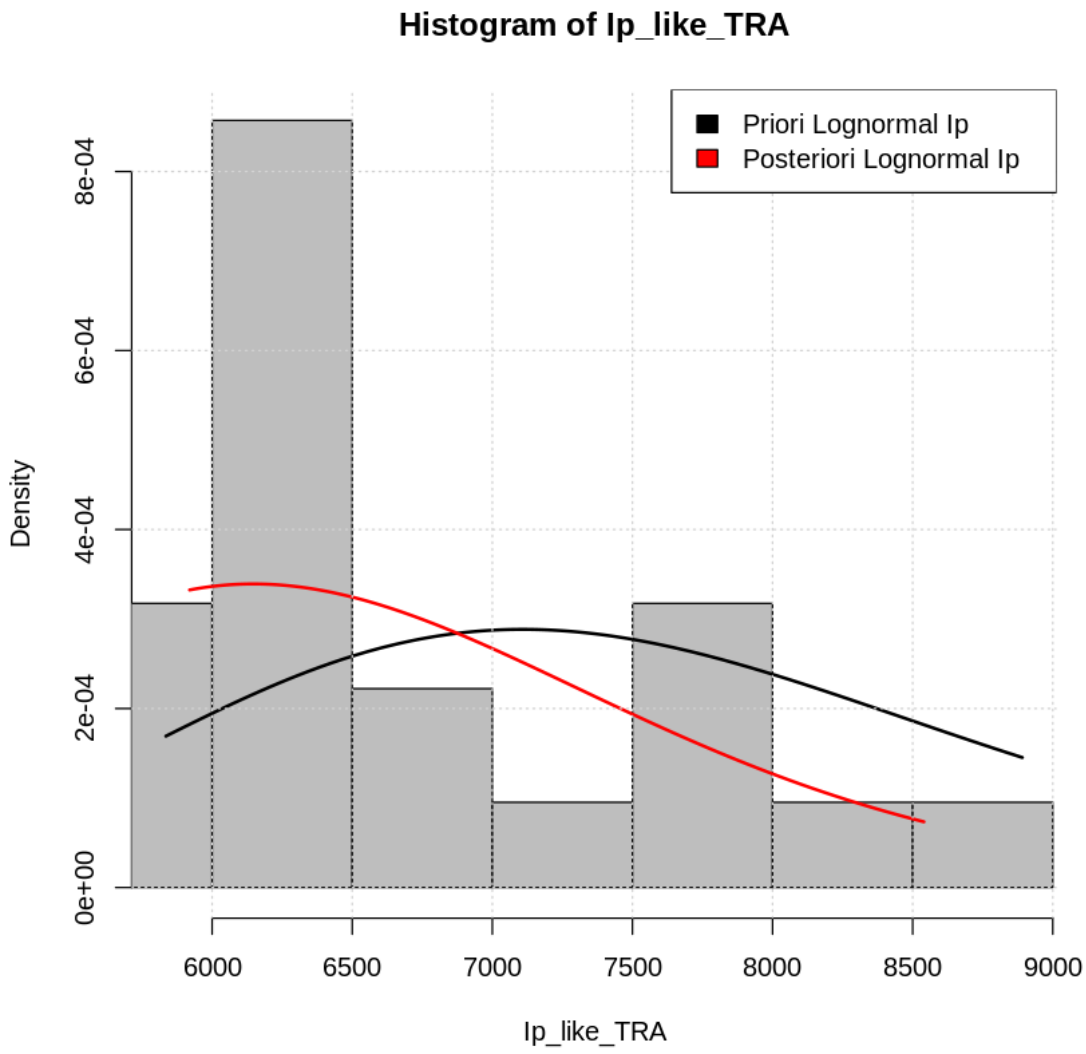
```r
plot(ecdf(Xf[,2]), col=("black"),xlim=c(0, 0.
 ↪32),lwd=3,xaxt='n',yaxt='n',ann=FALSE) #IFME
par(new=TRUE)
plot(ecdf(cop_frank_post_ran_5TRA[,2]), col=("red"),xlim=c(0, 0.
 ↪32),lwd=3,xaxt='n',yaxt='n',ann=FALSE) #Clayton Itau
grid()
legend(x = "topleft", legend = c(expression(paste(phi[t]," subsampled")),
                                 expression(paste(phi[t]," priori")),
                                 expression(paste(phi[t]," posteriori"))), fill␣
 ↪= c("blue","black","red"))
```



[116]:
```
den.x <- seq(min(Ip_like),max(Ip_like),length=3696)
den.y_prior <- dlnorm(den.x,meanlog = 8.905543, sdlog = 0.191057)

den.x_post_5TRA <- seq(min(Ip_like_TRA),max(Ip_like_TRA),length=3696)
den.y_post_5TRA <- dlnorm(den.x_post_5TRA,meanlog = mu_best_post_5TRA, sdlog =␣
↪sigma_best_post_5TRA)
```

The prior and posterior density distribution function for Ip changes, the posterior density moves to the left, but the coverage is better than the prior density function.

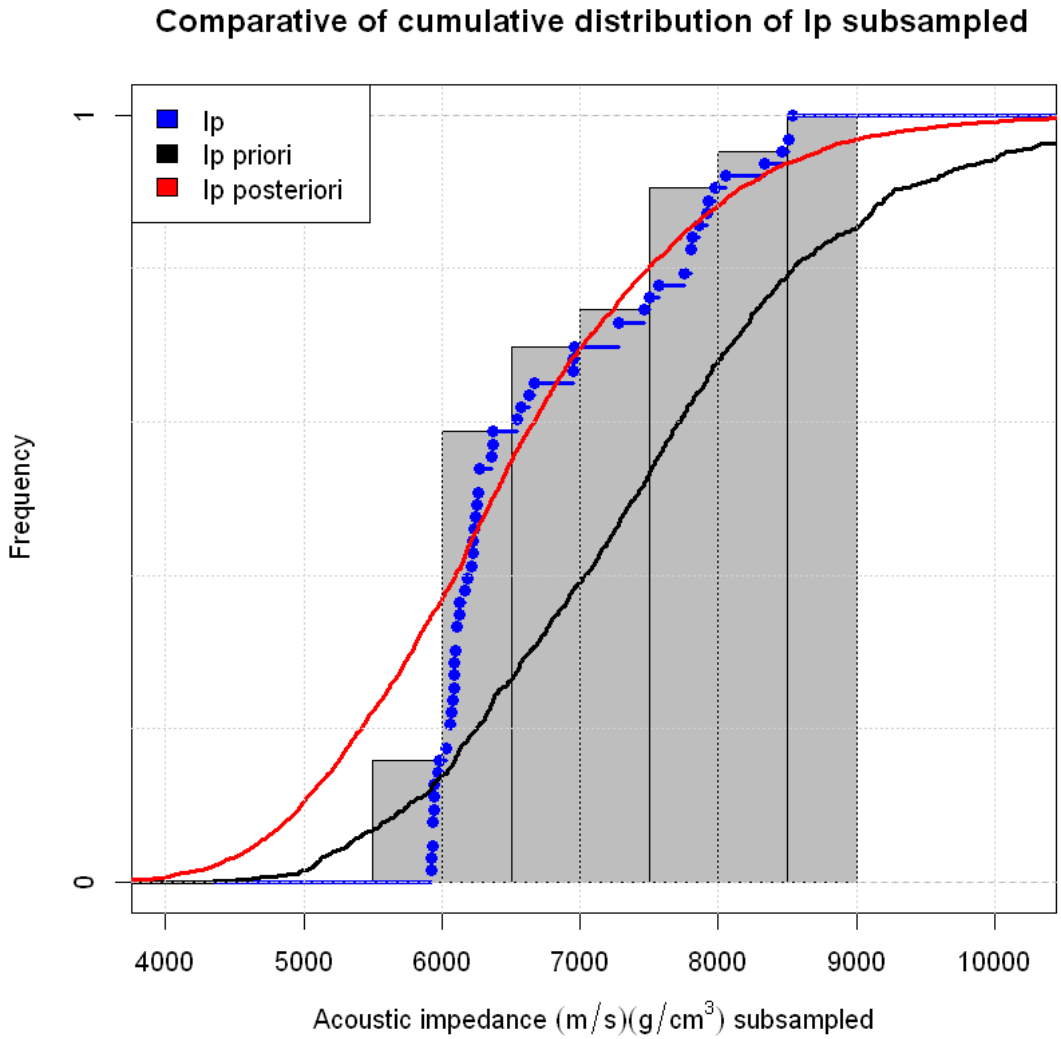
[117]:
```
hist(Ip_like_TRA, breaks=nsp, col="gray", probability=TRUE,xlim =␣
↪c(min(Ip_like),max(Ip_like)))
lines(den.x, den.y_prior, col="black", lwd=2,xlim =␣
↪c(min(Ip_like),max(Ip_like)))
lines(den.x_post_5TRA, den.y_post_5TRA, col="red", lwd=2,xlim =␣
↪c(min(Ip_like),max(Ip_like)))
grid()
legend(x = "topright", legend = c("Priori Lognormal Ip", "Posteriori Lognormal␣
↪Ip"),
       fill = c("black","red"))
```



The prior and posterior cumulative distribution function confirm the information observed in the density distribution function. The function is really close to subsample data.

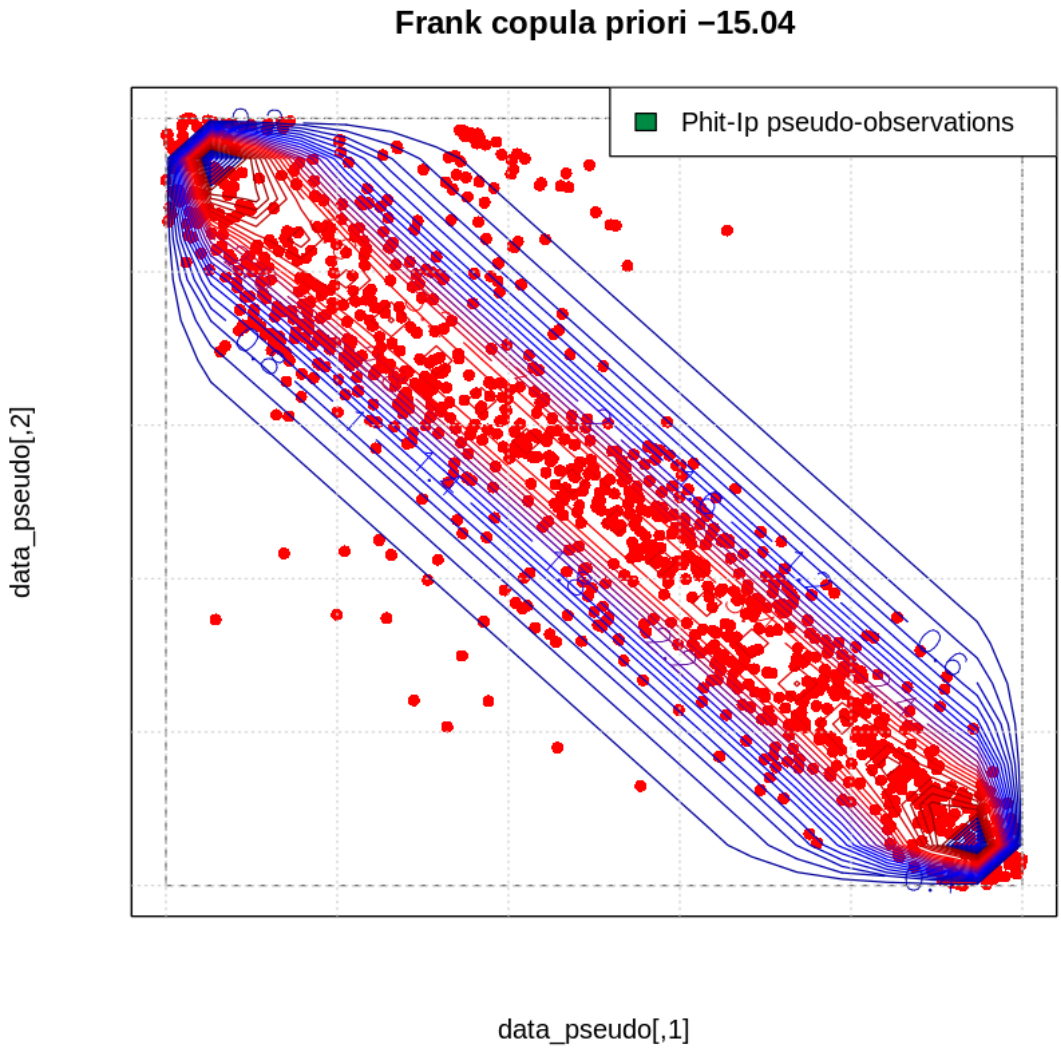
[7]:
```
hcum <- h <- hist(Ip_like_TRA, plot=FALSE, breaks=nsp)
hcum$counts <- cumsum(hcum$counts)/sum(hcum$counts)
plot(hcum, xlim=c(4000, 10200), col=("gray"),main = ("Comparative of cumulative␣
 ↪distribution of Ip subsampled"),
     xlab= expression(paste("Acoustic impedance ",(m/s)(g/cm^3), " subsampled")))
par(new=TRUE)
plot(ecdf(Ip_like_TRA), col=("blue"),xlim=c(4000,␣
 ↪10200),lwd=3,xaxt='n',yaxt='n',ann=FALSE )
par(new=TRUE)
```

```
plot(ecdf(Xf[,1]), col=("black"),xlim=c(4000,␣
↪10200),lwd=3,xaxt='n',yaxt='n',ann=FALSE) #IFME
par(new=TRUE)
plot(ecdf(cop_frank_post_ran_5TRA[,1]), col=("red"),xlim=c(4000,␣
↪10200),lwd=3,xaxt='n',yaxt='n',ann=FALSE) #Clayton Itau
grid()
legend(x = "topleft", legend = c("Ip", "Ip priori","Ip posteriori"), fill =␣
↪c("blue","black","red"))
```



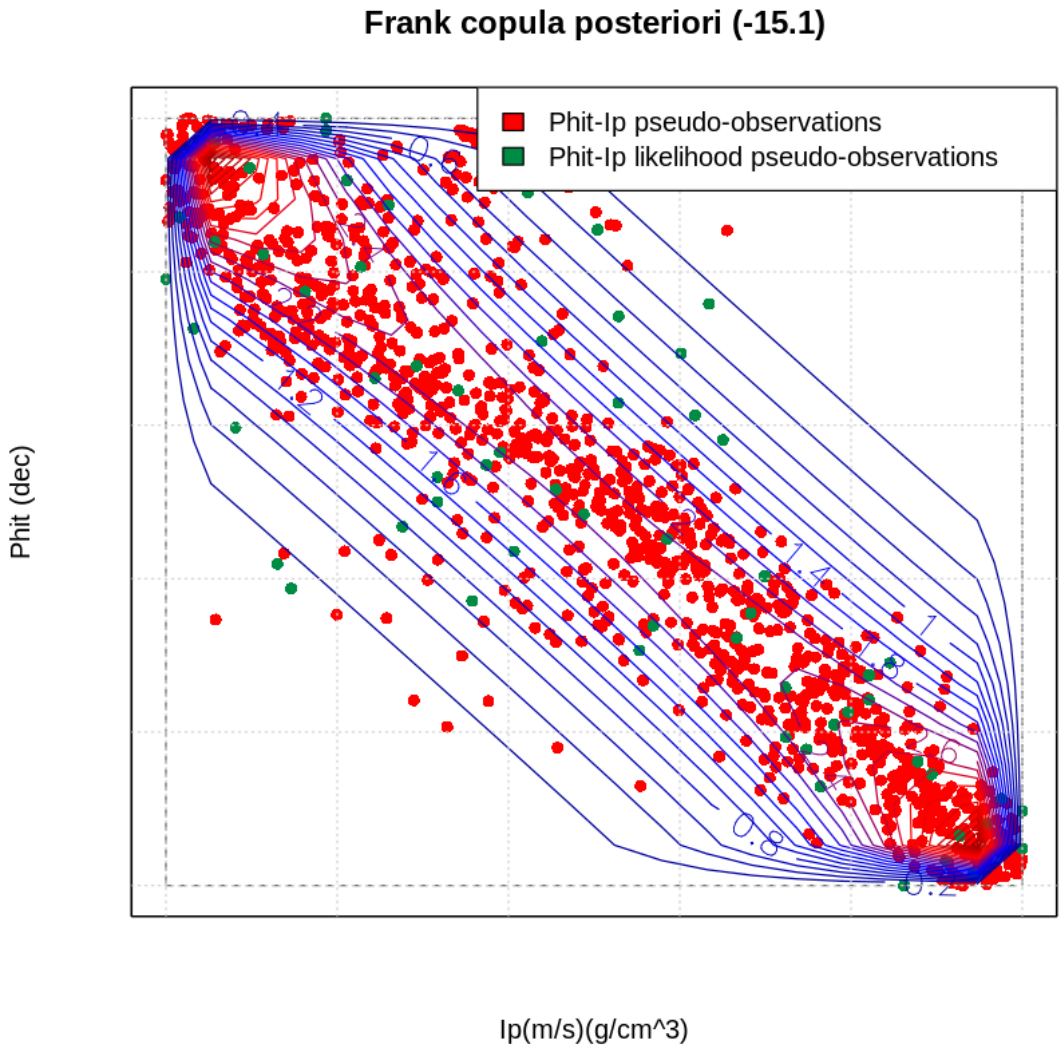
[119]:
```
plot(data_pseudo, pch=16, col=("red"),xaxt='n',yaxt='n',main = ("Frank copula␣
↪priori 15.04"))
par(new=TRUE)
```

```
contour(frankCopula(-15.04), dCopula,n=20,nlevels=25, xlab=(""), ylab=(""),␣
↪,labcex=1.4, col=cols,xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend = c("Phit-Ip pseudo-observations"), fill = c(␣
↪"springgreen4"))
```



[120]:
```
plot(data_pseudo, pch=16, col=("red"), xlab=(""),␣
↪ylab=(""),xaxt='n',yaxt='n',main = ("Frank copula posteriori (-15.1)"))
par(new=TRUE)
plot(pobs(Data_like_TRA), col=("springgreen4"), xlab=("Ip(m/s)(g/cm^3)"),␣
↪pch=16, ylab=("Phit (dec)"),xaxt='n',yaxt='n',main = (""))
par(new=TRUE)
```

```
contour(frankCopula(param = theta_best_post_5TRA), dCopula,n=20,nlevels=25,␣
 ↪xlab=(""), ylab=("") ,labcex=1.4, col=cols,xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend = c("Phit-Ip pseudo-observations","Phit-Ip␣
 ↪likelihood pseudo-observations"), fill = c("red", "springgreen4"))
```



9.3 Conditional joint simulation of total porosity with seismic scale acoustic impedance.

Whith the posterior model simulate the total porosity ϕ_t using the seismic scale acoustic impedance Ip as conditional data, for this case the posterior model was used to generate 39,942 pairs.

[121]:
```
Ip_trace_df<-data.frame(Ip_like_TRA)
XFrF_Ip_trace<-Ip_trace_df[rep(seq_len(nrow(Ip_trace_df)), each = 634), ]
```

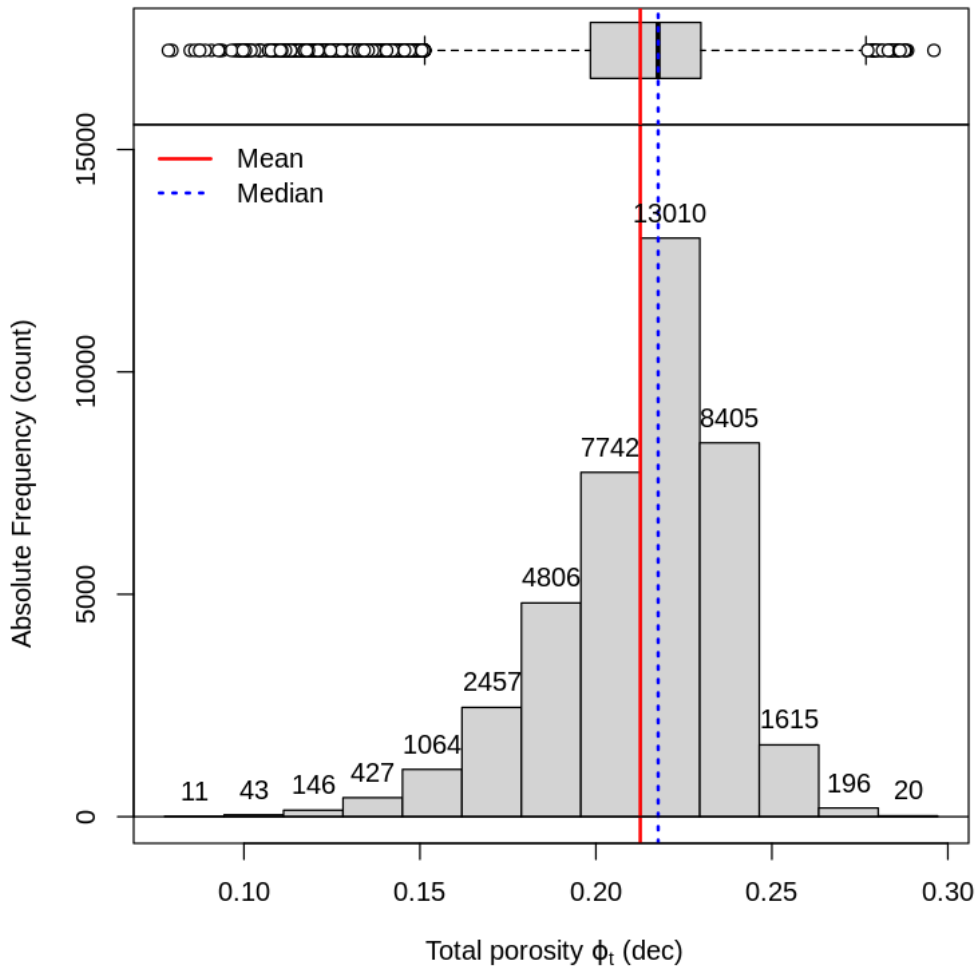
[126]:
```
Ip_cond_trace <- plnorm(XFrF_Ip_trace,meanlog = mu_best_post_5TRA, sdlog =␣
↪sigma_best_post_5TRA)

Fr_cond_trace<-cCopula(cbind(Ip_cond_trace,runif(length(Ip_cond_trace))),
                copula = frankCopula(theta_best_post_5TRA), inverse = TRUE)
```

[127]:
```
XFrF_Phit_trace <- qweibull(Fr_cond_trace[,2],shape = alpha_best_post_5TRA,␣
↪scale = lambda_best_post_5TRA)


XFrF_pair_trace<-cbind(XFrF_Ip_trace,XFrF_Phit_trace)
write.csv(XFrF_pair_trace, file = "Results/Frank_copula_trace_obs.csv")
```

[128]:
```
XFrF_Ip_trace_Stat<-Estadisticas(XFrF_Ip_trace)
XFrF_Phit_trace_Stat<-Estadisticas(XFrF_Phit_trace)
```

[129]:
```
HistBoxplot(XFrF_Phit_trace,nbins = ns, mean = XFrF_Phit_trace_Stat[5,2], median␣
↪= XFrF_Phit_trace_Stat[4,2],
            main ="",
            xlab = expression(paste(" Total porosity ",varphi[t], " (dec)")),␣
↪ylab = "Absolute Frequency (count)",
            AbsFreq = TRUE, PercentFreq = FALSE )
XFrF_Phit_trace_Stat
```

	Statistics <chr>	Values <dbl>
muestras	n	39942.0000
minimos	Minimum	0.0785
cuantiles1	1st. Quartile	0.1984
medianas	Median	0.2177
medias	Mean	0.2126
cuantiles3	3rd. Quartile	0.2298
maximos	Maximum	0.2960
rangos	Rank	0.2175
rangosInt	Interquartile Rank	0.0314
varianzas	Variance	0.0006
desvs	Standard Deviation	0.0252
CVs	Variation Coeff.	0.1185
simetrias	Skewness	-0.8812
curtosiss	Kurtosis	4.1132

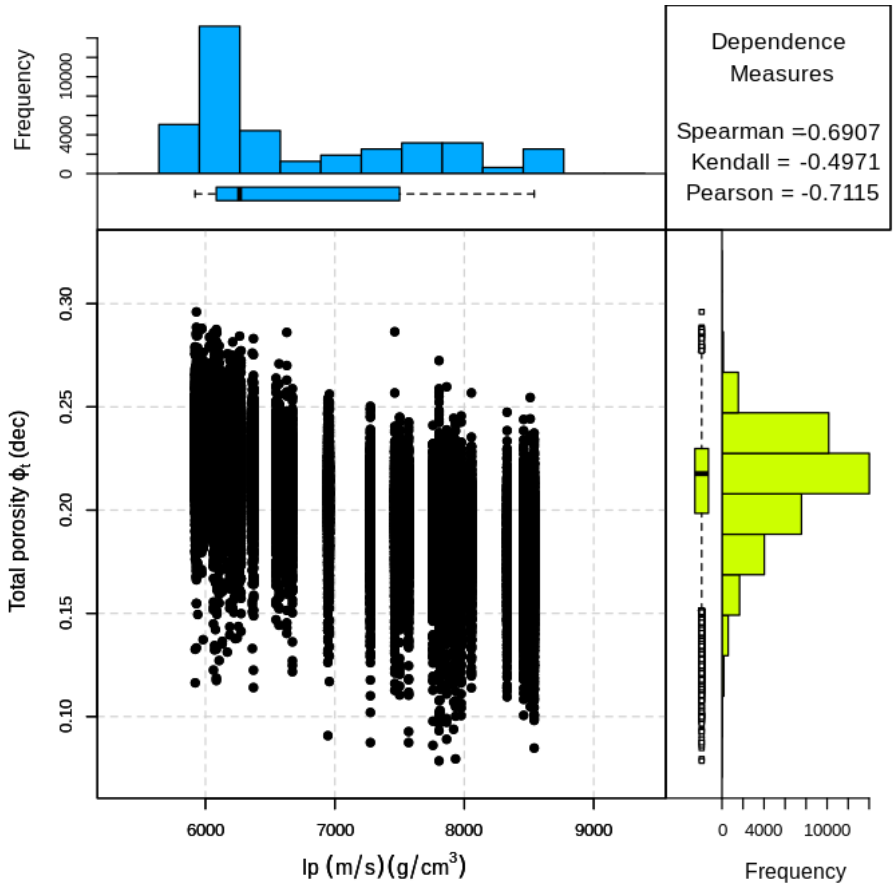


Comparing the statistics, the behaviour is the same as observed in the last example. The results are also consistents in both scales, well log and seismis.

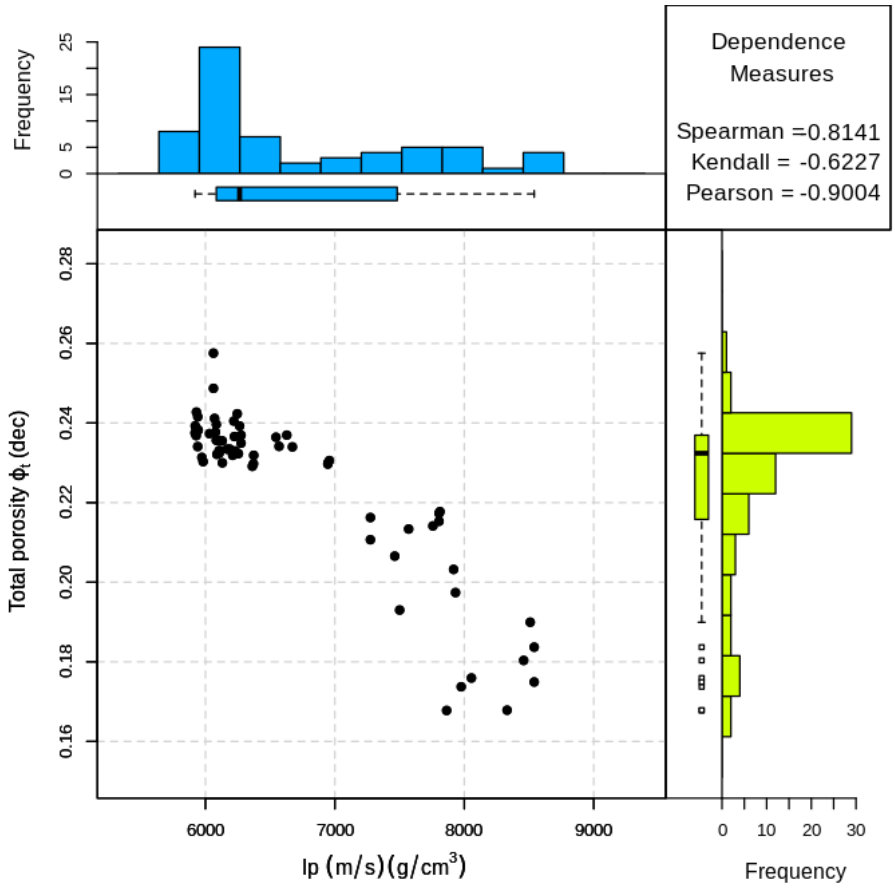
[9]:
```
Comparative_post_trace_Stat<-cbind(Ip_Stat,␣
 ↪Ip_like_TRA_Stat[,2],XFrF_Ip_trace_Stat[,2],
                                ␣
 ↪Phit_Stat[,2],Phit_like_SUB_Stat[,2],XFrF_Phit_trace_Stat[,2])
#Comparative_post_Stat<-Val_Estadisticos(Comparative_post)
colnames(Comparative_post_trace_Stat)<-c("Statistics","Ip","Ip subsampled","Ip␣
 ↪simulated","phit","phit subsampled", "phit simulated")
Comparative_post_trace_Stat
```

	Statistics <chr>	Ip <dbl>	Ip subsampled <dbl>	Ip simulated <dbl>	phit <dbl>	phit subsampled <dbl>
muestras	n	3696.0000	63.0000	39942.0000	3696.0000	63.0000
minimos	Minimum	5086.0072	5919.7222	5919.7222	0.0620	0.1678
cuantiles1	1st. Quartile	6157.0052	6084.1604	6084.0015	0.2147	0.2158
medianas	Median	6809.5574	6264.1689	6264.1689	0.2295	0.2324
medias	Mean	6968.2839	6727.7487	6727.7487	0.2220	0.2239
cuantiles3	3rd. Quartile	7321.6170	7481.5164	7500.5479	0.2414	0.2369
maximos	Maximum	11661.4642	8539.9961	8539.9961	0.2939	0.2575
rangos	Rank	6575.4570	2620.2739	2620.2739	0.2319	0.0898
rangosInt	Interquartile Rank	1164.6118	1397.3560	1416.5464	0.0267	0.0212
varianzas	Variance	1310302.9359	720107.0427	708694.5152	0.0011	0.0005
desvs	Standard Deviation	1144.6846	848.5912	841.8400	0.0338	0.0216
CVs	Variation Coeff.	0.1643	0.1261	0.1251	0.1521	0.0964
simetrias	Skewness	1.5610	0.8545	0.8752	-1.8102	-1.3036
curtosiss	Kurtosis	5.7657	2.2712	2.2712	6.9970	3.7221

[131]:
```
ScatterPlot(XFrF_Ip_trace , XFrF_Phit_trace, ns,
            Xmin = XFrF_Ip_trace_Stat[2,2], Xmax = XFrF_Ip_trace_Stat[7,2],
            Ymin = XFrF_Phit_trace_Stat[2,2],Ymax = XFrF_Phit_trace_Stat[7,2],
            XLAB = expression(paste(" Ip ",(m/s)(g/cm^3))),
        YLAB = expression(paste(" Total porosity ",varphi[t], " (dec)")))
```

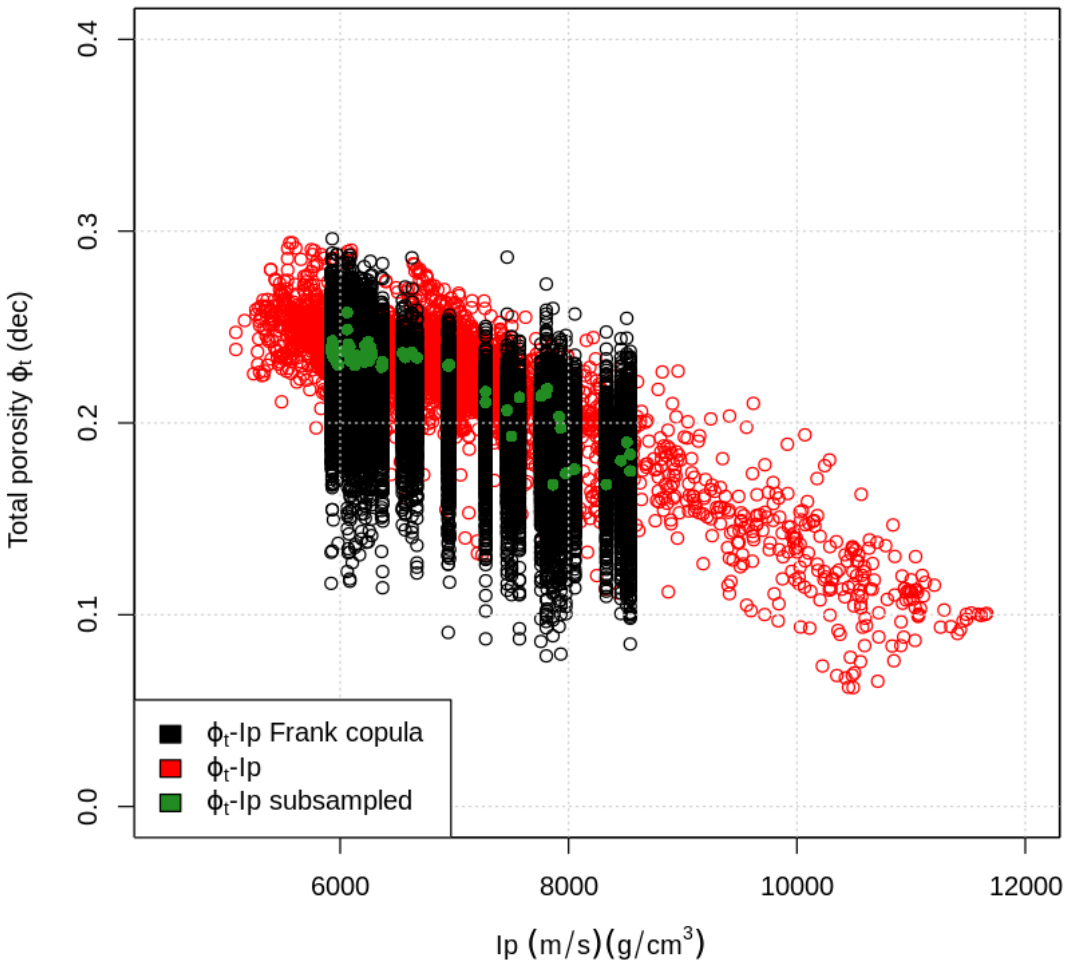


[132]:
```
ScatterPlot(Ip_like_TRA , Phit_like_SUB, ns,
            Xmin = Ip_like_TRA_Stat[2,2], Xmax = Ip_like_TRA_Stat[7,2],
            Ymin = Phit_like_SUB_Stat[2,2],Ymax = Phit_like_SUB_Stat[7,2],
            XLAB = expression(paste(" Ip ",(m/s)(g/cm^3))),
          YLAB = expression(paste(" Total porosity ",varphi[t], " (dec)")))
```



[133]:
```
plot(Ip , Phit, xlim=c(4500, 12000),ylim=c(0, 0.
 ↪40),col=("red"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(XFrF_Ip_trace , XFrF_Phit_trace,xlim=c(4500, 12000),ylim=c(0, 0.
 ↪40),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Ip_like_TRA , Phit_like_SUB, xlab = expression(paste(" Ip ",(m/s)(g/
 ↪cm^3))),
       ylab = expression(paste(" Total porosity ",varphi[t], "␣
 ↪(dec)")),xlim=c(4500, 12000),ylim=c(0, 0.40),
    col=("forestgreen"), pch=16)
grid()
```

```
legend(x = "bottomleft", legend = c(expression(paste(varphi[t],"-Ip Frank␣
↪copula")),
                                    expression(paste(varphi[t],"-Ip")),
                                   expression(paste(varphi[t],"-Ip␣
↪subsampled"))),
       fill = c("black", "red","forestgreen"))
```



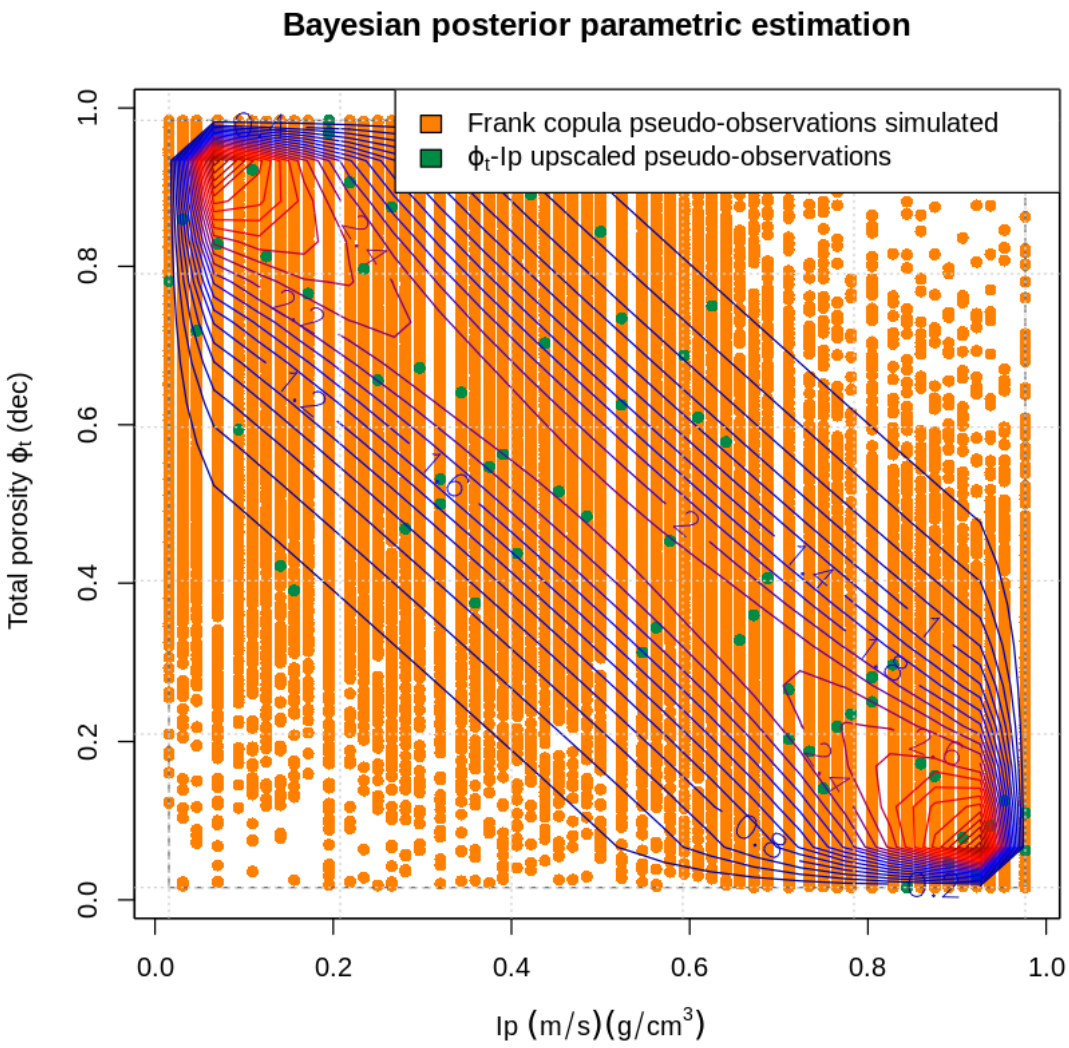
The dependence measures shows that the data sample at well log scale and the simulated samples in pearson measure are close. The other measures have big differences.

Dependence	Data sample	Seismic data sample	Simulated Parametric
Pearson	-0.8563	-0.7478	-0.8419
Spearman	-0.7107	-0.6751	-0.8089

Dependence	Data sample	Seismic data sample	Simulated Parametric
Kendall	-0.5335	-0.4894	-0.6152

After evaluate the simulated samples, the simulated annealing method is used to generate spatial distributions conditionated to acoustic impedance, the first example uses the acoustic impedance obtained from well log using the variogram estimated from seismic scale.

[134]:
```
plot(pobs(cbind(XFrF_Ip_trace , XFrF_Phit_trace)), pch=16,␣
↪col=("darkorange1"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(pobs(cbind(Ip_like_TRA , Phit_like_SUB)), xlab=(""), ylab=(""),
     main = ("Bayesian posterior parametric estimation"), col=("springgreen4"),␣
↪pch=16)
par(new=TRUE)
contour(frankCopula(theta_best_post_5TRA), dCopula,n=20,nlevels=25, xlab =␣
↪expression(paste(" Ip ",(m/s)(g/cm^3))),
     ylab = expression(paste(" Total porosity ",varphi[t], " (dec)")) ,
        labcex=1.4, col=cols,xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend = c("Frank copula pseudo-observations simulated",
                                  expression(paste(varphi[t],"-Ip upscaled␣
↪pseudo-observations"))),
       fill = c("darkorange1", "springgreen4"))
```



[136]:
```
Data_like_FULL_100_full <- read.table(file='Results/100_sim_all_TRACE.
↪txt',header=T,na.strings="-999.25")    #open file well_f03-4_502.5_1859.55.csv
#the select file is ip_Phie.csv
attach(Data_like_FULL_100_full)
```

```
The following objects are masked _by_ .GlobalEnv:

    Ip, Phit
```

[137]:
```
Ip_100_full<-Data_like_FULL_100_full$Ip
Phit_100_full<-Data_like_FULL_100_full$Phit
```

```
Ip_100_full_Stat<-Estadisticas(Ip_100_full)
Phit_100_full_Stat<-Estadisticas(Phit_100_full)
```

In this example the well log acoustic impedance is used as conditional. The stiatistics between ϕ_t well log and ϕ_t simulated values, the minimum and maximum have a difference of 0.02. The values in median and mean the the three cases are very close. The variance obtained between well log and simulated is the same, but is the double if compare the seismic scale with the simulated.

[138]:
```
Comparative_post_Stat_full<-cbind(Ip_Stat,␣
↪Ip_like_TRA_Stat[,2],Ip_100_full_Stat[,2],
                                  ␣
↪Phit_Stat[,2],Phit_like_SUB_Stat[,2],Phit_100_full_Stat[,2])
#Comparative_post_Stat<-Val_Estadisticos(Comparative_post)
colnames(Comparative_post_Stat_full)<-c("Statistics","Ip","Ip Seismic","Ip␣
↪simulated","phit","phit Seismic",
                                        "phit simulated")
Comparative_post_Stat_full
```

	Statistics <chr>	Ip <dbl>	Ip Seismic <dbl>	Ip simulated <dbl>	phit <dbl>	phit Seismic <dbl>	phit s <dbl>
muestras	n	3696.0000	63.0000	6300.0000	3696.0000	63.0000	6300.0
minimos	Minimum	5086.0072	5919.7222	5919.7222	0.0620	0.1678	0.1458
cuantiles1	1st. Quartile	6157.0052	6084.1604	6084.0015	0.2147	0.2158	0.2031
medianas	Median	6809.5574	6264.1689	6264.1690	0.2295	0.2324	0.2177
medias	Mean	6968.2839	6727.7487	6727.7487	0.2220	0.2239	0.2141
cuantiles3	3rd. Quartile	7321.6170	7481.5164	7500.5478	0.2414	0.2369	0.2276
maximos	Maximum	11661.4642	8539.9961	8539.9961	0.2939	0.2575	0.2718
rangos	Rank	6575.4570	2620.2739	2620.2739	0.2319	0.0898	0.1260
rangosInt	Interquartile Rank	1164.6118	1397.3560	1416.5464	0.0267	0.0212	0.0245
varianzas	Variance	1310302.9359	720107.0427	708789.2777	0.0011	0.0005	0.0004
desvs	Standard Deviation	1144.6846	848.5912	841.8962	0.0338	0.0216	0.0190
CVs	Variation Coeff.	0.1643	0.1261	0.1251	0.1521	0.0964	0.0888
simetrias	Skewness	1.5610	0.8545	0.8750	-1.8102	-1.3036	-0.662
curtosiss	Kurtosis	5.7657	2.2712	2.2712	6.9970	3.7221	3.0684

Now graph the simulations

[139]:
```
n=100 #numero de columnas
f=63  #numero de filas
ff<-0
```

[140]:
```
sepf=matrix(0,n,2)
for (i in 1:n) {
  ff<-ff+f
 sepf[i,2]<-ff
}
sepf[,1]<-sepf[,2]-(f-1)
```

[ ]:
```
Data_Phit_TRA=matrix(0,f,n)
for (i in 1:n) {
  Data_Phit_TRA[,i]<-Data_like_FULL_100_full[sepf[i,1]:sepf[i,2],5]
}
```

[142]:
```
DF_Phit_sim_TRA<-data.frame(Data_Phit_TRA)

nc<-ncol(DF_Phit_sim_TRA)
nf<-nrow(DF_Phit_sim_TRA)
```

[143]:
```
P_TRA<-matrix(0, nc, 1)
```

[144]:
```
for (i in 1:ncol(DF_Ip_sim_TRA)) {
   P_TRA[i,] <- (sum((Phit_like_SUB-DF_Phit_sim_TRA[,i])^2))/nf
}
```

[145]:
```
minP_TRA<-which.min(P_TRA)
minP_TRA
```

4

[146]:
```
Estadisticas(Data_Phit_TRA[,minP_TRA])
```

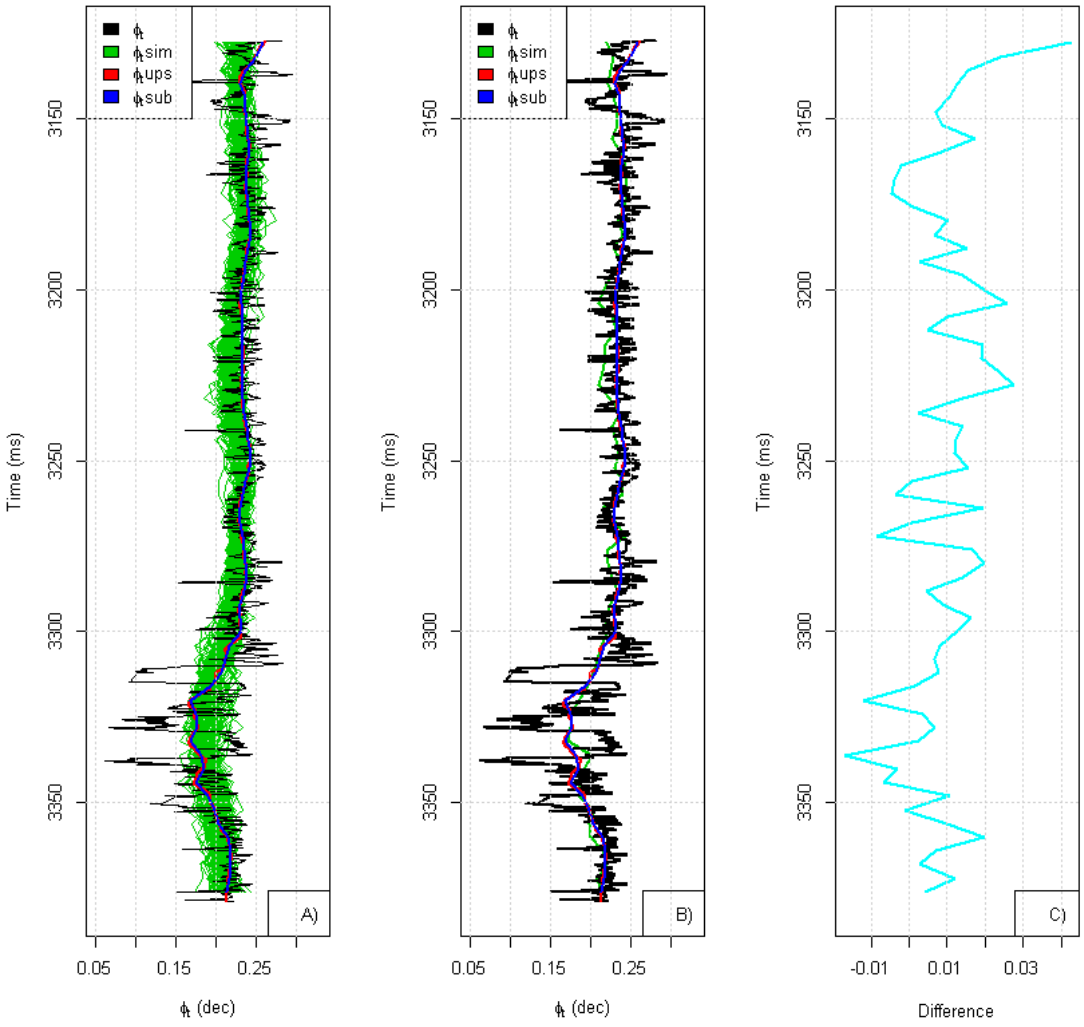
	Statistics <chr>	Values <dbl>
muestras	n	63.0000
minimos	Minimum	0.1653
cuantiles1	1st. Quartile	0.2045
medianas	Median	0.2178
medias	Mean	0.2148
cuantiles3	3rd. Quartile	0.2285
maximos	Maximum	0.2420
rangos	Rank	0.0767
rangosInt	Interquartile Rank	0.0240
varianzas	Variance	0.0004
desvs	Standard Deviation	0.0187
CVs	Variation Coeff.	0.0872
simetrias	Skewness	-0.8187
curtosiss	Kurtosis	3.1494

The spatial distribution of 100 realizations shows good results, amost all the samples are under the interval observed, just a few samples are far, this is observed in the interval from 3275 to 3300 meters.

[17]:
```
#jpeg("Results/final/image8tra.jpeg", bg = "white", width = 2500, height = 2500,␣
↪res = 300)

par(mfrow = c(1,3))
for (i in 1:ncol(Data_Phit_TRA)) {
```

```
   plot(Data_Phit_TRA[,i],time_like_SUB,xlim=c(0.04, 0.33),
        ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =␣
↪"l",lwd=1,col=("green3"),
        xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
}
plot(Phit,time_UPS,xlim=c(0.05, 0.33),xlab = expression(paste(phi[t], "␣
↪(dec)")),ylab="Time (ms)",
     ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =␣
↪"l",lwd=1,col=("black"))
par(new=TRUE)
plot(Phit_like,time_UPS,xlim=c(0.05, 0.
↪33),ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2]))
     ,type = "l",lwd=2,col=("red"),
     xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Phit_like_SUB,time_like_SUB,xlim=c(0.05, 0.
↪33),ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2]))
     ,type = "l",lwd=2,col=("blue"),
     xaxt='n',yaxt='n',ann=FALSE)
legend(x = "topleft", legend =␣
↪c(expression(paste(phi[t])),expression(paste(phi[t], "sim")),
                                  expression(paste(phi[t],␣
↪"ups")),expression(paste(phi[t], "sub"))),
       fill = c("black", "green3","red","blue"))
legend(x="bottomright",legend="A)")
grid()

plot(Data_Phit_TRA[,minP_TRA],time_like_SUB,xlim=c(0.04, 0.
↪33),ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2]))
     ,type = "l",
     lwd=2,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Phit,time_UPS,xlim=c(0.05, 0.33),xlab = expression(paste(phi[t], "␣
↪(dec)")),ylab="Time (ms)",
     ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type =␣
↪"l",lwd=2,col=("black"))
par(new=TRUE)
plot(Phit_like,time_UPS,xlim=c(0.05, 0.
↪33),ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2]))
     ,type = "l",lwd=2,col=("red"),
     xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Phit_like_SUB,time_like_SUB,xlim=c(0.05, 0.
↪33),ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2]))
     ,type = "l",lwd=2,col=("blue"),
```

```
     xaxt='n',yaxt='n',ann=FALSE)
legend(x = "topleft", legend =␣
 ↪c(expression(paste(phi[t])),expression(paste(phi[t], "sim")),
                                 expression(paste(phi[t],␣
 ↪"ups")),expression(paste(phi[t], "sub"))),
       fill = c("black", "green3","red","blue"))
legend(x="bottomright",legend="B)")
grid()

plot((Phit_like_SUB-Data_Phit_TRA[,minP_TRA]),time_like_SUB,xlab="Difference",ylab="Time␣
 ↪(ms)",
     ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),
     type = "l",lwd=2,col=("cyan"))
legend(x="bottomright",legend="C)")
grid()

#dev.off()
```



[18]:
```
save.image() #important line, it's for save the R workspace, include variables␣
↪and results
```

The joint distribution moel can be used in the neares traces, the simulated annealing method can simulate the total porosity using the same data obtained from copula simulation, the acoustic impedance at seismic scale is used as conditional.

10 Conclutions

The methodology proposed here through the use of a Bayesian approach for petrophysical simulation using a parametric dependence model based on copulas offers adequate results, since it reproduces with greater precision the data statistics and the posterior estimated model improves the prediction in comparison with the prior parametric model. This can be achieved using the Bayesian approach

by obtaining an optimal set of model parameters.

Regarding the comparison of the Bayesian method with the deterministic method, it was noted that the deterministic model was able to reproduce the statistics of the scaled samples, but the dependency model was not satisfactory, which led to obtaining simulations of the scaled register with high variations in several sections. On the other hand, the dependency model obtained from the Bayesian method did not manage to approach all the statistics of the scaled samples, but the dependency model did allow obtaining the simulations of both the scaled registry and the original registry. This represents a great advantage to bring the model to the seismic scale.

Given the advantages found with this analysis, it is possible to make a prediction of the total porosity on a seismic scale using the acoustic impedance estimated by the sparse spilke method. For this, 20 realizations are simulated using the same set of simulated samples with the a posteriori dependence model of the previous case, the same variogram model and each seismic scale acoustic impedance trace is used as a conditional variable. As can be seen, the inline has the minimum at 0.1350 and the maximum is 0.2552 with a mean of 0.2057, median of 0.2122 and standard deviation of 0.0259. Therefore, the prediction shows that the low and high porosity intervals are consistent with those observed in the well log. Only in the interval 3350 to 3376 ms does a discrepancy seem to be observed between the high porosity shown by the total porosity records and the low porosity of the prediction. This indicates that the seismic-scale prediction of total porosity is consistent in the vast majority of the inline, but there are other areas that require more work or a greater number of realizations.

As future work, this proposal should be extended to the case of joint petropysical inversion, using a copula model to linking seismic traces, petrophysical properties and seismic attributes. To this end could be considered trivariate parametric copulas or a hierarchical copula scheme.

11 Aknowlenment

The data were provided by the National Hydrocarbons Commission of Mexico, according to appendix C of the license to use the information in favor of Universidad Nacional Autónoma de México, dated December 11, 2017, under the nomenclature CNIH-C-00417.305. This reference information is the property of Mexico and its collection, safekeeping, use, administration and updating, as well as its publication is only authorized to the National Hydrocarbons Commission.

12 Supplementary material

A pdf version of Jupyter notebook in R language is available in https://github.com/esmg-mx/Copula-Base-modelling

References

[Al Labadi et al., 2019] Al Labadi, L., Fazeli Asl, F., and Saberi, Z. (2019). A bayesian semiparametric gaussian copula approach to a multivariate normality test.

[Albert, 2009] Albert, J. (2009). *Bayesian Computation with R*. Use R! Springer New York.

[Armstrong et al., 2004] Armstrong, M., Galli, A., Bailey, W., and Couët, B. (2004). Incorporating technical uncertainty in real option valuation of oil projects. *Journal of Petroleum Science and Engineering*, 44:67–82.

[Arreguin-Lopez et al., 2011] Arreguin-Lopez, M. A., Reyna-Martinez, G., Sánchez-Hernández, H., Escamilla-Herrera, A., and Gutierrez-Araiza, A. (2011). Tertiary turbidite systems in the south-western gulf of mexico. *Geology*.

[Ausin and Lopes, 2010] Ausin, M. C. and Lopes, H. F. (2010). Time-varying joint distribution through copulas. *Computational Statistics & Data Analysis*, 54(11):2383–2399. The Fifth Special Issue on Computational Econometrics.

[Azevedo and Soares, 2017] Azevedo, L. and Soares, A. (2017). *Geostatistical Methods for Reservoir Geophysics*. Springer International Publishing.

[Bernardo and Smith, 2009] Bernardo, J. M. and Smith, A. F. M. (2009). *Bayesian Theory*. Wiley.

[Bortoli et al., 1993] Bortoli, L.-J., Alabert, F., Haas, A., and Journel, A. (1993). *Constraining stochastic images to seismic data*. Springer.

[Buland and Omre, 2003] Buland, A. and Omre, H. (2003). Bayesian linearized avo inversion. *GEOPHYSICS*, 68:185–198.

[Cosentino, 2001] Cosentino, L. (2001). *Integrated Reservoir Studies*, volume 53.

[Díaz-Viera et al., 2021] Díaz-Viera, M. A., Hernández-Maldonado, V., Méndez-Venegas, J., Mendoza-Torres, F., Le, V. H., and Vázquez-Ramírez, D. (2021). *RGEOESTAD: Un programa de código abierto para aplicaciones geoestadísticas basado en R-Project*.

[Doyen, 2007] Doyen, P. (2007). *Seismic reservoir characterization: an earth modelling perspective*. EAGE publications.

[Dubrule et al., 2003] Dubrule, O., of Exploration Geophysicists, S., of Geoscientists, E. A., and Engineers (2003). *Geostatistics for Seismic Data Integration in Earth Models: 2003 Distinguished Instructor Short Course*. Society of Exploration Geophysicists.

[Grana, 2014] Grana, D. (2014). Probabilistic approach to rock physics modeling. *Geophysics*, 79.

[Grana et al., 2022] Grana, D., Azevedo, L., Figueiredo, L., and Mukerji, T. (2022). Probabilistic inversion of seismic data for reservoir petrophysical characterization: Review and examples. *GEOPHYSICS*, 87:1–99.

[Grazian and Liseo, 2016] Grazian, C. and Liseo, B. (2016). Approximate bayesian inference in semiparametric copula models. *Bayesian Analysis*, 12.

[Haas and Dubrule, 1994] Haas, A. and Dubrule, O. (1994). Geostatistical inversion: a sequential method of stochastic reservoir modeling constrained by seismic data. *First Break*, 12:561–569.

[Hofert et al., 2019] Hofert, M., Kojadinovic, I., Mächler, M., and Yan, J. (2019). *Elements of Copula Modeling with R*. Springer International Publishing.

[Hoyos-Argüelles and Nieto-Barajas, 2019] Hoyos-Argüelles, R. and Nieto-Barajas, L. (2019). A bayesian semiparametric archimedean copula. *Journal of Statistical Planning and Inference*, 206.

[Jaworski et al., 2010] Jaworski, P., Durante, F., Härdle, W. K., and Rychlik, T. (2010). *Copula Theory and Its Applications: Proceedings of the Workshop Held in Warsaw, 25-26 September 2009*. Springer Berlin Heidelberg.

[Joe, 2014] Joe, H. (2014). *Dependence Modeling with Copulas*. Taylor & Francis.

[Kohn et al., 2006] Kohn, R., Pitt, M., and Chan, D. (2006). Efficient bayesian inference for gaussian copula regression models. *Biometrika*, 93:537–554.

[Kroese et al., 2013] Kroese, D. P., Taimre, T., and Botev, Z. I. (2013). *Handbook of Monte Carlo Methods*. Wiley.

[Le, 2021] Le, V. H. (2021). Copula-based modeling for petrophysical property prediction using seismic attributes as secondary variables.

[Le et al., 2020] Le, V. H., Díaz-Viera, M. A., Vázquez-Ramírez, D., del Valle-García, R., Erdely, A., and Grana, D. (2020). Bernstein copula-based spatial cosimulation for petrophysical property prediction conditioned to elastic attributes. *Journal of Petroleum Science and Engineering*, 193:107382.

[Menke, 2012] Menke, W. (2012). *Geophysical Data Analysis: Discrete Inverse Theory: MATLAB Edition*. Elsevier Science.

[Min and Czado, 2010] Min, A. and Czado, C. (2010). Bayesian inference for multivariate copulas using pair-copula constructions. *Journal of Financial Econometrics*, 8:511–546.

[Ramírez, 2018] Ramírez, D. V. (2018). *Simulación estocástica conjunta de propiedades petrofísicas con cópulas de Bernstein usando atributos sísmicos como variables secundarias a escala de registros de pozo*. Universidad Nacional Autónoma de México.

[Rosen and Thompson, 2015] Rosen, O. and Thompson, W. (2015). Bayesian semiparametric copula estimation with application to psychiatric genetics: Bayesian semiparametric copula estimation. *Biometrical journal. Biometrische Zeitschrift*, 57.

[Saraiva et al., 2018] Saraiva, E., Suzuki, A., and Milan, L. (2018). Bayesian computational methods for sampling from the posterior distribution of a bivariate survival model, based on amh copula in the presence of right-censored data. *Entropy*, 20:642.

[Schamberger et al., 2017] Schamberger, B., Gruber, L., and Czado, C. (2017). Bayesian inference for latent factor copulas and application to financial risk forecasting. *Econometrics*, 5:21.

[Shemyakin and Kniazev, 2017] Shemyakin, A. and Kniazev, A. (2017). *Introduction to Bayesian Estimation and Copula Models of Dependence*. Wiley.

[Silva and Lopes, 2008] Silva, R. and Lopes, H. (2008). Copula, marginal distributions and model selection: A bayesian note. *Statistics and Computing*, 18:313–320.

[Smith, 2011] Smith, M. (2011). Bayesian approaches to copula modelling. *ERN: Bayesian Analysis (Topic)*.

[Tarantola, 2005] Tarantola, A. (2005). *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics.

[Valle et al., 2017] Valle, L., Leisen, F., and Rossini, L. (2017). Bayesian non-parametric conditional copula estimation of twin data. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 67.