Joint seismic inversion of elastic and petrophysical properties using copula-based stochastic cosimulation models-Copy1

April 19, 2025

1 Joint geostatistical seismic inversion of elastic and petrophysical properties using stochastic cosimulation models based on parametric copulas

Authors: Daniel Vázquez-Ramírez[(1)] , Martín Díaz-Viera[(2)], R. Valle-García[(3)].

1) Earth Sciences Graduate Program (UNAM)
2) Mexican Petroleum Institute
3) Independent consultant

2 Abstract

Seismic properties play a fundamental role in the geological and petrophysical modeling of reservoirs due to their dependence on petrophysical properties. Most existing stochastic seismic inversion methods are based on Gaussian probability distribution functions and assume linear dependence. Examples include sequential Gaussian co-simulation (SGCS) and direct sequential simulation (DSS). In contrast, spatial stochastic co-simulation methods based on Bernstein copulas (BCCS) have recently been developed. These methods do not require a specific distribution type or linear dependence, thereby overcoming the limitations of traditional approaches.

In this context, a novel approach for joint seismic inversion of elastic and petrophysical properties based on copulas is proposed. Where the forward modeling process applies a parametric copula-based stochastic co-simulation (PCCS) model within a Bayesian inference framework. This approach uses well-scale petrophysical and seismic property data, fitted to parametric functions, as prior information. The model parameters are estimated *a posteriori* using petrophysical properties scaled by a moving window averaging method and seismic properties upscaled using the Backus averaging method as new information. The resulting posterior model is considered the forward model and is used to generate realizations of seismic attributes at the seismic scale.

The inverse modeling process involves generating joint realizations of petrophysical properties (e.g., porosity) and elastic properties (e.g., acoustic impedance) based on the forward model. Synthetic seismic traces are then produced by convolving the simulated acoustic impedance. An iterative optimization framework using simulated annealing is employed to minimize the root mean square error between the synthetic and observed seismic traces.

The proposed method was applied to a deepwater reservoir case study to estimate total porosity and acoustic impedance at the seismic scale. Results demonstrate that the use of parametric

copulas reduces computational cost and execution time. Furthermore, the effectiveness of the copula-based seismic inversion approach was confirmed, with synthetic traces exhibiting root mean square errors below 8%.

3 Introduction

The prediction of seismic and petrophysical properties is a key challenge in the geological and petrophysical modeling workflow (Cosentino, 2001). To predict the spatial distribution of petrophysical properties at the seismic scale, it is first necessary to estimate the spatial distribution of elastic properties. These elastic properties act as a bridge between seismic and petrophysical models, making it essential to establish a methodology that links them through an inverse modeling approach. The process of transforming seismic traces to estimate elastic and petrophysical properties is commonly known as seismic inversion. This technique is based on the concept of the inverse problem, which involves estimating model parameters from a set of observations. Typically, this process consists of three stages: (1) defining an objective function, (2) optimizing this function, and (3) solving the resulting system of equations to obtain a quantitative solution. The objective function, often called the forward model, must replicate the seismic trace under specific assumptions. Various mathematical models have been proposed for solving the inverse problem, including convolutional models and finite difference-based models. These models are commonly optimized using criteria such as the L1 norm or least squares, aiming to minimize the objective function error and obtain the most probable solution. A major limitation of these approaches is their tendency to linearize the problem, which may introduce uncertainty into the predictive model.

Seismic inversion in the petroleum industry has been an active area of research since the 1980s. Early works by Oldenburg et al. (1983), Cooke and Schneider (1983), and Gelfand and Larner (1984) presented methods to linearize the inverse problem using seismic surveys and well-log data. Their objective functions were designed to perform deconvolution based on maximum likelihood or L1 norm principles. During the 1990s, new inversion techniques emerged that attempted to integrate petrophysics, rock physics, and seismic data using Gaussian probability distributions, as seen in the works of Bortoli et al. (1993), Haas and Dubrule (1994), Buland and Omre (2003), Dubrule et al. (2003), Doyen (2007), and Grana (2014). However, this approach has notable disadvantages due to the limitations inherent to Gaussian simulations. For instance, Kriging equations are constrained to preserve low covariance between the original data and their estimates, and the variance of the random function is assumed to be stationary (Pyrcz and Deutsch, 2014). Moreover, these methods are not suitable when the number or spatial distribution of wells is insufficient, or when the correlation between petrophysical, seismic, and rock physics properties is too weak to meet the linearity assumptions of Gaussian-based models.

To address these issues, alternative simulation techniques have been explored. Copula-based stochastic simulation is a promising alternative for simulating properties such as petrophysical or seismic attributes, given their conditional dependence. A key advantage of this approach is its flexibility: it offers a nonparametric technique that avoids assumptions about specific functional forms or dependence structures. In the geological-petrophysical modeling context, copula-based simulation enables the estimation of petrophysical and elastic models that are consistent with well-log statistics. As shown by Diaz-Viera et al. (2017), the resulting realizations do not exhibit artificial smoothing away from wells. This methodology improves geostatistical seismic inversion by reducing uncertainty. However, existing applications of copulas have primarily used nonpara-

metric forms, which—while accurate—require significant computational resources. To address this limitation, this study proposes the use of parametric copulas. These are simpler to implement within a Bayesian framework and require fewer computational resources, although they may sacrifice some accuracy compared to their nonparametric counterparts.

This work introduces a novel copula-based joint seismic inversion method, developed in response to the limitations of existing stochastic seismic inversion approaches. Recent inversion methods using geostatistical approaches are still based on Gaussian or multi-Gaussian probability distribution functions (Azevedo and Soares, 2017; Pereira et al., 2020, 2023), or on mixed-Gaussian models (Grana et al., 2017). This is a significant constraint because it often requires transforming the data to satisfy distributional assumptions, potentially introducing bias during reverse transformation. To construct the forward model, we first define a joint probability distribution using a parametric copula fitted to petrophysical and elastic well-log data. This distribution is then updated using upscaled well-log data through a Bayesian framework. For the inverse problem, the posterior distribution serves as a search function to generate samples of seismic attributes via the simulated annealing method. These attributes are convolved with a wavelet to produce synthetic seismic traces, which are then compared to the observed trace using the least-squares criterion. Once the best-fitting synthetic trace is identified, the corresponding elastic property is used as a conditioning variable to simulate the petrophysical property using the same joint probability distribution.

To demonstrate the theoretical framework and advantages of the proposed method, this paper is organized into eight sections. Section 1 presents the introduction, offering a concise overview of seismic inversion. Section 2 outlines the theoretical basis of the copula-based joint seismic inversion approach. Section 3 describes the application workflow, offering a detailed, step-by-step implementation guide. Section 4 presents the case study, including geological context, statistical analysis of well logs, dependence modeling, and variogram estimation. Section 5 provides a validation of the method using the seismic trace closest to the well. Section 6 details the inversion process using the nearest seismic inline, applying the parameters defined earlier. Section 7 discusses the results, and Section 8 presents the conclusions.

3.1 Objective.

Generate petrophysical and elastic property distribution models using a stochastic copula-based simulation model to perform probabilistic data inversions using the resulting joint probability distribution functions.

4 Parametric Copula-Based Joint Geostatistical Seismic Inversion Method

5 Parametric Copula-Based Joint Geostatistical Seismic Inversion Method

Geostatistical seismic inversion is a well-established approach for solving the inverse problem using a trace-by-trace scheme, as proposed by Bortoli et al. (1993) and Haas and Dubrule (1994). This type of inversion employs stochastic simulation to generate multiple realizations of an elastic property. These realizations are convolved with a wavelet to produce synthetic seismic traces, which are then compared to real seismic traces using an optimization method. The process iter-

ates until a stopping criterion is satisfied, either based on the objective function or the maximum number of iterations.

The Copula-Based Joint Geostatistical Seismic Inversion Method integrates copula-based spatial stochastic co-simulation to jointly simulate elastic and petrophysical properties. This methodology has been presented in several works by Erdely and Díaz-Viera (2010), Hernández-Maldonado et al. (2012), Hernández-Maldonado et al. (2014), Erdely and Díaz-Viera (2015), Diaz-Viera et al. (2017), Vázquez-Ramírez (2018), Le et al. (2020) and Le (2021). Its main advantage lies in offering a nonparametric simulation approach through Bernstein Copula Co-simulation (BCCS), which eliminates the need to assume a specific marginal distribution or specific dependence structure. This method provides a robust alternative for simulating one property (e.g., elastic or petrophysical) conditioned on another, under a flexible dependence model. The resulting realizations are statistically consistent and preserve spatial distribution characteristics.

The copula-based spatial stochastic simulation method is a process which can be subdivided into two primary steps:

- A dependence model is established between a primary and a secondary variable using a Bernstein copula.
- The spatial dependence structure (variogram) of the primary variable is used to predict this variable conditioned on the secondary variable, using the Simulated Annealing method.

However, prior studies that apply this method to predict petrophysical properties at the seismic scale often follow a deterministic scheme. They typically subsample geophysical well-log data and rely on elastic property images, which may increase uncertainty. This is because the dependence model derived at the well scale can differ significantly from that obtained using subsampled logs. To address this issue, parametric copula co-simulation (PCCS) is proposed as an alternative to BCCS. The use of parametric functions facilitates lower computational cost while maintaining modeling flexibility (Vázquez-Ramírez et al., 2023).

The Parametric Copula-Based Joint Geostatistical Seismic Inversion Method involves the following three main steps:

- Estimate the dependence model using parametric copulas within a Bayesian framework.
- Generate realizations of the elastic property using its spatial dependence structure (variogram) until the synthetic trace with the lowest mean square error, relative to the observed seismic trace, is found—using the PCCS method.
- Predict the petrophysical property conditioned on the elastic property obtained in the previous step, using the same PCCS method.

The key innovation in this work is the use of Bayesian inference to update the dependence model, as demonstrated in the works by Kohn et al. (2006), Silva and Lopes (2008), Ausin and Lopes (2010), Min and Czado (2010) and Smith (2011). This approach allows the joint posterior probability distribution function to be updated using both upscaled and subsampled well-log data, ensuring consistency with all available information sources. Because the dependence model can incorporate multiple elastic and petrophysical properties estimated from well logs, it enables the joint generation of realizations for these properties.

As part of the inversion procedure, joint realizations of petrophysical properties (e.g., porosity) and elastic properties (e.g., acoustic impedance) derived from the forward model are generated. The simulated acoustic impedance is convolved with a wavelet to produce synthetic seismic traces, which are optimized using the global heuristic method of Simulated Annealing.

6 Joint Geostatistical Seismic Inversion Based on Copulas Workflow

The joint geostatistical seismic inversion method based on copulas is developed through the execution of eight steps, as illustrated in Figure~1. The following subsections provide a detailed explanation of each step and the requirements for their implementation.

6.1 Exploratory Data Analysis

This step involves univariate and bivariate exploratory analysis of well logs and their upscaled versions. The univariate analysis includes the computation of descriptive statistics (e.g., minimum and maximum values), as well as the generation of histograms and boxplots to assess the impact of upscaling. The bivariate analysis entails evaluating dependence measures and visualizing scatterplots before and after upscaling, to detect potential changes in relationships between variables.

6.2 Variographic Analysis

This step consists of estimating and modeling the spatial correlation function (variogram) from the upscaled samples (Díaz-Viera, 2002). The variogram model should be fitted to samples in the time domain, which facilitates the integration of seismic-scale realizations. Preprocessing is required to eliminate trends, and cross-validation should be employed to validate the chosen variogram model.

6.3 Estimation of the Joint Distribution Model via Bayesian Inference

To ensure consistency between well-log data and the upscaled values used at the seismic scale, a Bayesian approach is adopted for estimating the parameters of the joint distribution model. This approach allows the integration of prior information derived from well-scale observations and new data from upscaled logs.

6.3.1 Prior Information

The prior distributions for the marginal variables and the copula parameter are selected empirically by fitting well-log data to various parametric probability distributions. For each marginal variable α_i and the copula parameter θ, candidate distributions are fitted and evaluated using statistical criteria such as the log-likelihood, Akaike Information Criterion (AIC), and Bayesian Information Criterion (BIC). The model with the highest log-likelihood and lowest AIC/BIC is selected as the prior. Visual inspection using histograms, cumulative histograms, and cumulative distribution functions is also performed to validate the quality of the fit.

This process results in a joint prior distribution defined by:

$$\pi(\theta, \alpha_1, \ldots, \alpha_d) = \pi(\theta) \prod_{i=1}^{d} \pi(\alpha_i) \tag{1}$$

where $\pi(\alpha_i)$ represents the prior for the i-th marginal parameter, and $\pi(\theta)$ is the prior for the copula dependence parameter.

6.3.2 Posterior Model and Likelihood

The posterior distribution is computed using Bayes' theorem, incorporating the upscaled or subsampled data as new information. The likelihood function is defined as the product of the copula density function and the marginal densities evaluated at the observed data. Specifically, for a d-dimensional variable, the joint log-posterior is given by:

$$\begin{aligned}\log(\pi(\theta,\alpha_i|x_i)) \propto \log(\pi(\theta,\alpha_i)) + \sum_{j=1}^{n} \log\left(c\left(F_1\left(x_1^{(j)};\alpha_1\right),\ldots,F_d\left(x_d^{(j)};\alpha_d\right);\theta\right)\right) \\ + \sum_{i=1}^{d}\sum_{j=1}^{n} \log f_i\left(x_i^{(j)};\alpha_i\right)\end{aligned} \tag{2}$$

Here, $f_i(x_i;\alpha_i)$ is the marginal density of variable i parameterized by α_i, and $c(u_1,\ldots,u_d;\theta)$ is the copula density function governing the joint dependence, where $u_i = F_i(x_i;\alpha_i)$. The samples are denoted by x (Jaworski et al., 2010)

This framework allows for the flexible integration of information from different scales and supports updating the model as new data becomes available. The parameters of the posterior distribution are estimated using the Random Walk Metropolis-Hastings algorithm, as described in Section~**??**.

Here, $f_i(x_i;\alpha_i)$ is the marginal density of variable i parameterized by α_i, and $c(u_1,\ldots,u_d;\theta)$ is the copula density function governing the joint dependence, where $u_i = F_i(x_i;\alpha_i)$.

After the posterior model is estimated using upscaled logs, a second posterior model is computed using subsampled logs as prior information. This serves as a quality control step, ensuring that seismic-scale realizations reflect the statistical properties observed in the well and upscaled data. Subsampled logs are constructed assuming both the elastic and petrophysical properties are in the time domain, sampled at 4~ms intervals. For each target time (e.g., 3200~ms), samples from $\pm$2~ms (e.g., 3198–3202~ms) are used to compute the median.

This framework allows for the flexible integration of information from different scales and supports updating the model as new data becomes available. The estimated joint distribution is then used to generate synthetic samples during the inversion process.

6.4 Spatial Simulation of the Elastic Property Using the PCCS Method

To generate the synthetic trace that best matches the seismic trace, elastic properties are simulated using the Simulated Annealing method, which relies on two main components: the objective function and the search function.

The objective function includes the terms to be minimized. If it comprises multiple functions, their contributions should be normalized to balance the optimization process. Deutsch and Journel (1998) suggests generating a few realizations for each function separately to evaluate their convergence behavior. The first objective function (Eq. 3) computes the semivariogram error:

$$O = \sum_{h} \frac{[\gamma^*(h) - \gamma(h)]^2}{\gamma(h)^2} \tag{3}$$

where $\gamma^*(h)$ is the semivariogram of the perturbed elastic property and $\gamma(h)$ is the model variogram.

The search function introduces perturbations to the initial image of the elastic property as follows:

- Propose an initial image.
- Randomly select a position in the image.
- Generate a pair of values from the joint distribution model estimated in step~6.3.
- Accept the sample if its values lie within the observed range; otherwise, resample.
- Insert the new value into the selected position to generate a perturbed image, which is evaluated using Eq.~3.

This process is repeated until a maximum number of iterations is reached or the objective function reaches its minimum value.

6.5 Estimation of the Synthetic Trace Using the Convolutional Model

The convolutional model is used to compute the synthetic seismic trace from the perturbed elastic property. First, the reflectivity series is calculated using Eq. 4:

$$r_t = \frac{(\rho_2)(V_{p2}) - (\rho_1)(V_{p1})}{(\rho_2)(V_{p2}) + (\rho_1)(V_{p1})} \tag{4}$$

Then, the synthetic trace is obtained by convolving the reflectivity series r_t (Eq. 4) with a wavelet f_t (Eq. 5) estimated from the seismic image:

$$S_{st} = f_t * r_t \tag{5}$$

6.6 Estimation of the Difference Between the Synthetic and Real Seismic Traces Using RMS Error

The second objective function is the root mean square (RMS) error between the real and synthetic traces. If the RMS error is smaller than in the previous iteration, the proposal is accepted. Otherwise, a new realization of the elastic property must be generated.

6.7 Spatial Simulation of the Petrophysical Property Conditional on the Elastic Property Using the PCCS Method

After optimizing the synthetic trace, the petrophysical property is simulated using the elastic property as a conditioning variable. This simulation is also performed using the Simulated Annealing method. The variogram model for the petrophysical property must be derived from the subsampled well logs in the time domain. The same joint distribution model estimated in step 6.3 is employed in the search function to generate petrophysical realizations.

7 Case Study Description

7.1 Data Description

The computational implementation was carried out in the R programming language. The statistical analysis and joint inversion method used the following libraries: `copula` (Hofert et al., 2019), `fitdistrplus`, `LearnBayes` (Albert, 2009), `RSEIS`, `NMOF`, and `RGEOSTAD` (Díaz-Viera et al., 2021).

[2]:
```
library(Rcpp)
library(maps)
library(mapproj)
library(actuar)
library(fields)
library(fitdistrplus)
library(geoR)
library(gstat)
library(MASS)
library(moments)
library(poweRlaw)
library(spatstat)
library(ADGofTest)
library(reshape)
library(sp)
library(copula)
library(NMOF)
library(RSEIS)
library(Metrics)
library(EnvStats)
#library(plotly)
library(RFOC)
library(car)
library(modeest)
library(LearnBayes)
```

[2]:
```
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)
```

To implement seismic inversion, four sources of information are required: the file containing the geophysical well logs.

[3]:
```
Data_File <- read.csv(file='data/Lakach-1_3035.0-3404.5_25-05-2017.
↪csv',header=T,na.strings="-999.25")
```

upscaled logs

[4]:
```
Data_like_UPS <- read.csv(file='data/upscaling_TWT_3696_sample_Lakach-1.
↪csv',header=T,na.strings="-999.25")
```

subsampled logs

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

and the seismic image

[5]:
```
Data_like_CUBE <- read.csv(file='data/inline_trace.csv',header=T,na.
↪strings="-999.25")
```

From the geophysical well log file, the depth (Depth), total porosity (Phit), and acoustic impedance (Ip) vectors are extracted.

[7]:
```
Depth<-Data_File$DEPTH
Phit<-Data_File$PHIT          #Total porosity
Ip<-Data_File$Ip              #acoustic impedance
```

From the upscaled log file, the vectors of total porosity scaled using the moving averaging window method (Phit_like), acoustic impedance upscaled from the scaled P-wave velocity and density logs using the Backus averaging method (Ip_like), and the time vector in the well log (time_UPS) are extracted.

[8]:
```
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
```

From the subsampled log file, the total porosity (Phit_like_SUB), acoustic impedance (Ip_like_SUB), depth (Depth_SUB), and time (time_like_SUB) logs are extracted.

[9]:
```
Ip_like_SUB<-Data_like_SUB$Ip_ups_Bks
Phit_like_SUB<-Data_like_SUB$PHIT_MA
Depth_SUB<-Data_like_SUB$Depth
time_like_SUB<-Data_like_SUB$time
```

Finally, the values of the seismic trace closest to the well are extracted.

[7]:
```
neartrace<-Data_like_CUBE$q
```

Statistical descriptors are computed for all vectors.

[11]:
```
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”
```

[12]:
```
Phit_like_Stat<-Estadisticas(Phit_like)
Ip_like_Stat<-Estadisticas(Ip_like)
time_UPS_Stat<-Estadisticas(time_UPS)
```

```
Warning message in kurtosis(datos):
“'x' is not a numeric vector”
```

[13]:
```
Phit_like_SUB_Stat<-Estadisticas(Phit_like_SUB)
Ip_like_SUB_Stat<-Estadisticas(Ip_like_SUB)
```

```
Depth_SUB_Stat<-Estadisticas(Depth_SUB)
time_like_SUB_Stat<-Estadisticas(time_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”
```

To plot the histograms, the number of bins is estimated; in this case, Sturges' rule is applied. For both the geophysical well log and the scaled log, the number of bins is determined to be 13.

[14]:
```
ns = nclass.Sturges(Ip)
ns
```

13

Given the limited number of samples in the subsampled logs, an appropriate number of bins is determined.

[15]:
```
nsp = nclass.Sturges(Ip_like_SUB)
nsp
```

7

To implement the stochastic simulation using parametric copulas, geophysical well logs were obtained from a deepwater well drilled in the Gulf of Mexico. This well penetrates the Miocene turbidite system, a significant unit due to the presence of unassociated hydrocarbon fields. The Miocene is subdivided into Lower, Middle, and Upper intervals. This study focuses on the Lower Miocene, where flow direction is primarily from southwest to northeast, with dominant channel systems. The lithology consists of medium-grained sandstones with evidence of volcanic material, feldspars, quartz, and metamorphic fragments. The rock is poorly consolidated and mineralogically immature, with reported porosity values ranging from 12% to 28% (Arreguin-Lopez et al., 2011).

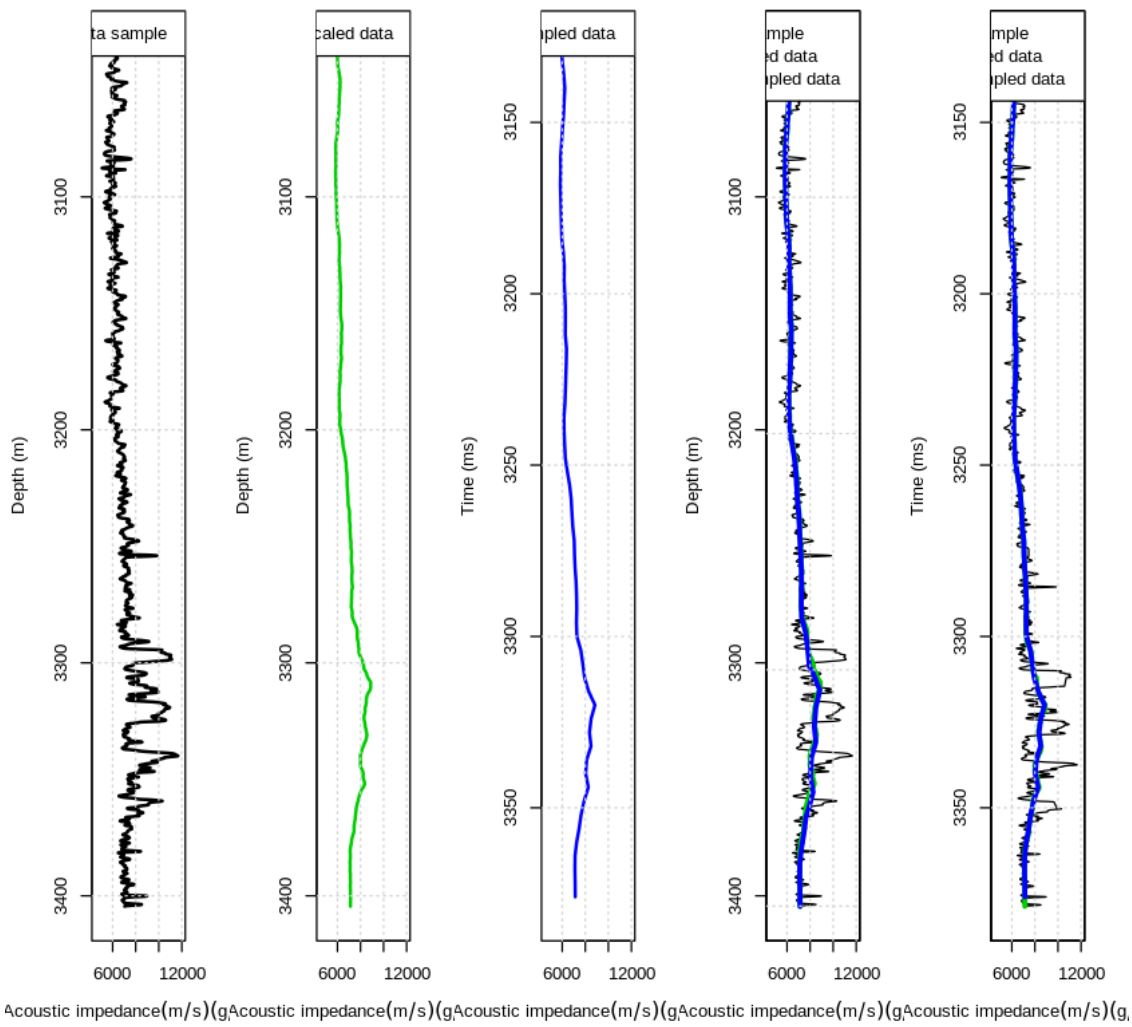
The well used in this study was selected based on the quality and completeness of its available data. It is the most informative well in the study area, containing interpreted logs for acoustic impedance and total porosity, along with other petrophysical properties. Its depth coverage, vertical resolution, and the availability of well-established interpretations make it representative of the stratigraphic interval targeted in this work and suitable for validating the proposed inversion methodology.

The interval analyzed spans from 3,035 to 3,404.5 meters, with a sampling interval of 0.1 meters. The available well logs are acoustic impedance (AI) and total porosity (ϕ_t), as shown in Figure **??**. These logs were upscaled using two methods: total porosity was upscaled using the moving average technique, while acoustic impedance was upscaled using the Backus method based on Vp (P-wave velocity) and ρ (density) logs. Both methods apply a vertical seismic resolution of 28.44 meters. Subsampled AI and ϕ_t values were extracted by computing the median within a $\pm$2~ms interval around each seismic grid node.

[16]:
```
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"), col = c("black"),lty=c(1))

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"), col = c( "green3"),lty=c(1))

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"), col = c("blue"),lty=c(1))

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"), col =␣
↪c("black", "green3","blue"),
       lty=c(1,1,1))

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)
```

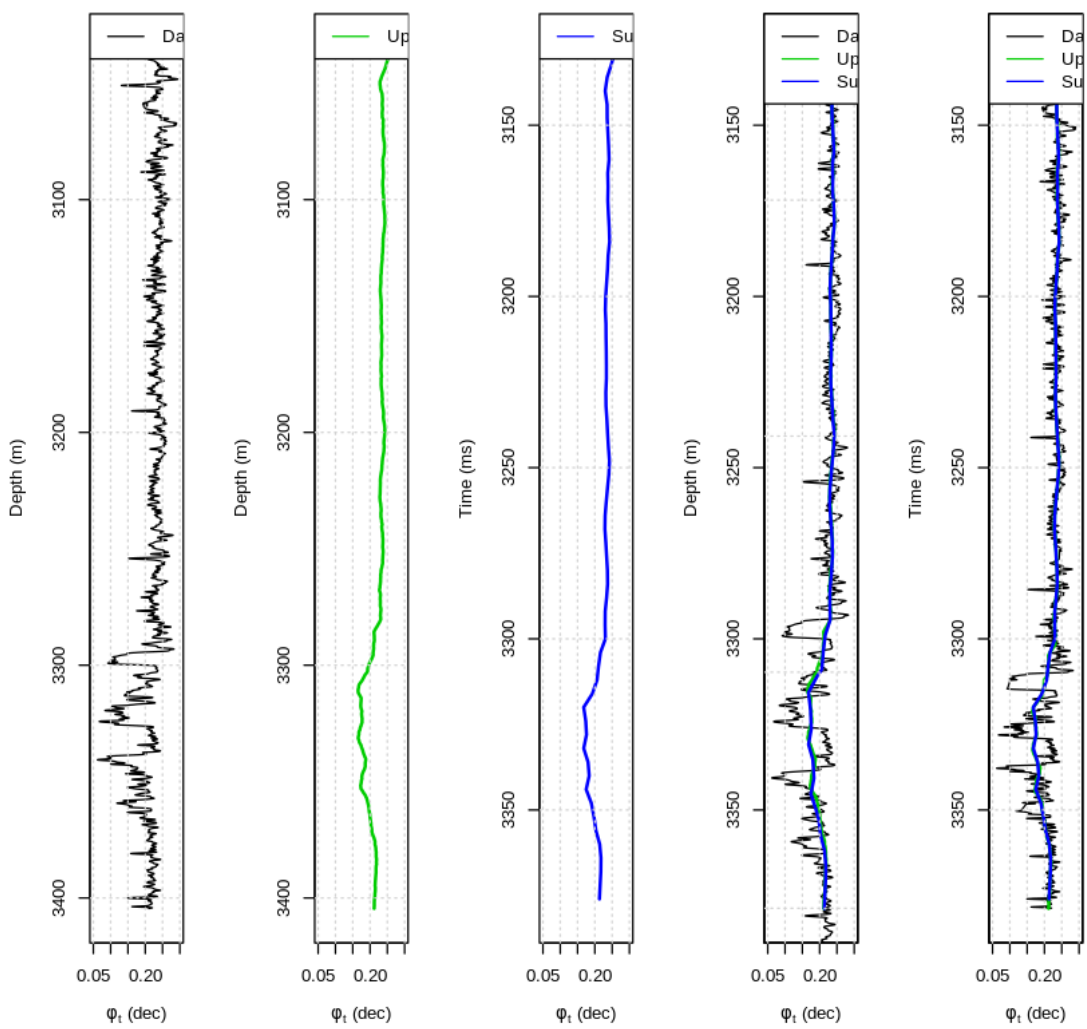
```
plot(Ip_like,time_UPS,xlim=c(4500,␣
↪12000),ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),type = "l",
     lwd=3,col=("green3"),
     xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Ip_like_SUB,time_like_SUB,xlim=c(4500,␣
↪12000),ylim=rev(c(time_UPS_Stat[2,2],time_UPS_Stat[7,2])),
     type = "l",lwd=3,col=("blue"),
     xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend = c("Data sample",
                                  "Upscaled data","Subsampled data"), col =␣
↪c("black", "green3","blue"),lty=c(1,1,1))
```



The same behavior is observed in the scaled and subsampled logs of total porosity.

[17]:
```
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"), col = c("black"),lty=c(1))

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"), col = c("green3"),lty=c(1))

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"), col = c("blue"),lty=c(1))

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"), col =␣
↪c("black", "green3","blue"),lty=c(1,1,1))

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"), col =␣
 ↪c("black", "green3","blue"),lty=c(1,1,1))
```



Descriptive statistics for the well, upscaled, and subsampled logs are provided in Table **??**. The well and upscaled logs contain 3,696 observations, while the subsampled log has 63 observations. For total porosity ϕ_t at the well scale, the difference between the mean and median is -0.008, with most outliers appearing on the left side of the histogram (Figure B). In contrast, the AI log has a larger difference between the mean and median is 158.726, indicating positive skewness due to high-value outliers (Figure A).

The upscaled total porosity values range from 0.165 to 0.261, and the histogram in Figure D is visually similar to that of the well-scale data. The upscaled AI values range from 5,835.096 to 8,891.212 $(m/s)(g/cm^3)$ and display apparent bimodal behavior in Figure C. Subsampled AI values range from 5,919.722 to 8,539.996, indicating a reduction of approximately 2,500 $(m/s)(g/cm^3)$ compared to the well-scale log. Subsampled ϕ_t values range from 0.167 to 0.257, with mean and median values similar to those of the original ϕ_t well log.

[18]:
```
Comparative_Stat<-cbind(Ip_Stat, Ip_like_Stat[,2],Ip_like_SUB_Stat[,2],
                                  ␣
 ↪Phit_Stat[,2],Phit_like_Stat[,2],Phit_like_SUB_Stat[,2])

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

A data.frame: 14 × 7

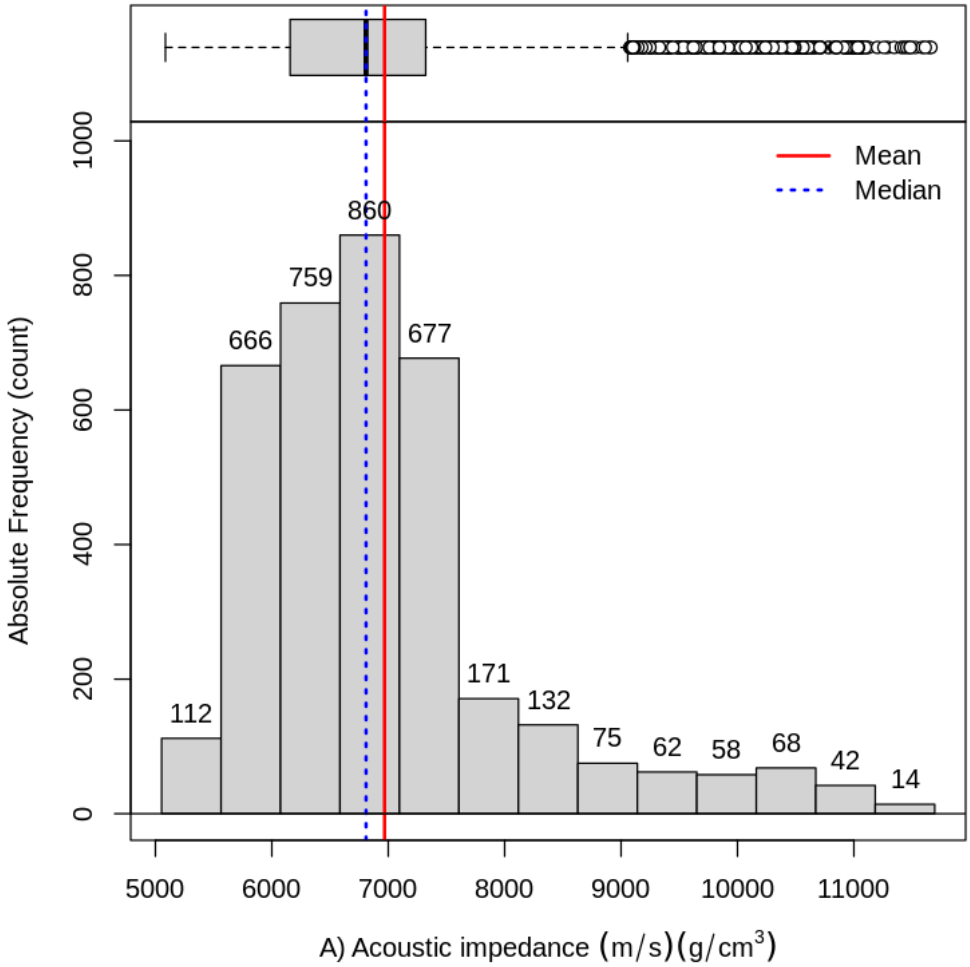
	Statistics <chr>	Acoustic impedance <dbl>	Acoustic impedance upscaled <dbl>	A <dbl>
muestras	n	3696.0000	3696.0000	6
minimos	Minimum	5086.0072	5835.0969	5
cuantiles1	1st. Quartile	6157.0052	6178.1181	6
medianas	Median	6809.5574	6831.4653	6
medias	Mean	6968.2839	6892.8985	6
cuantiles3	3rd. Quartile	7321.6170	7353.4905	7
maximos	Maximum	11661.4642	8891.2124	8
rangos	Rank	6575.4570	3056.1155	2
rangosInt	Interquartile Rank	1164.6118	1175.3724	1
varianzas	Variance	1310302.9359	708798.2641	6
desvs	Standard Deviation	1144.6846	841.9016	8
CVs	Variation Coeff.	0.1643	0.1221	0
simetrias	Skewness	1.5610	0.5967	0
curtosiss	Kurtosis	2.7711	-0.8097	-

[19]:
```
#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
```

A data.frame: 14 × 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	2.7711



The case of acoustic impedance processed using the Backus scaling method exhibits two peaks: the first at 6,178.1181 $(m/s)(g/cm^3)$ and the second around 7,050 $(m/s)(g/cm^3)$, indicating evidence of bimodal behavior. There is no indication of outliers.

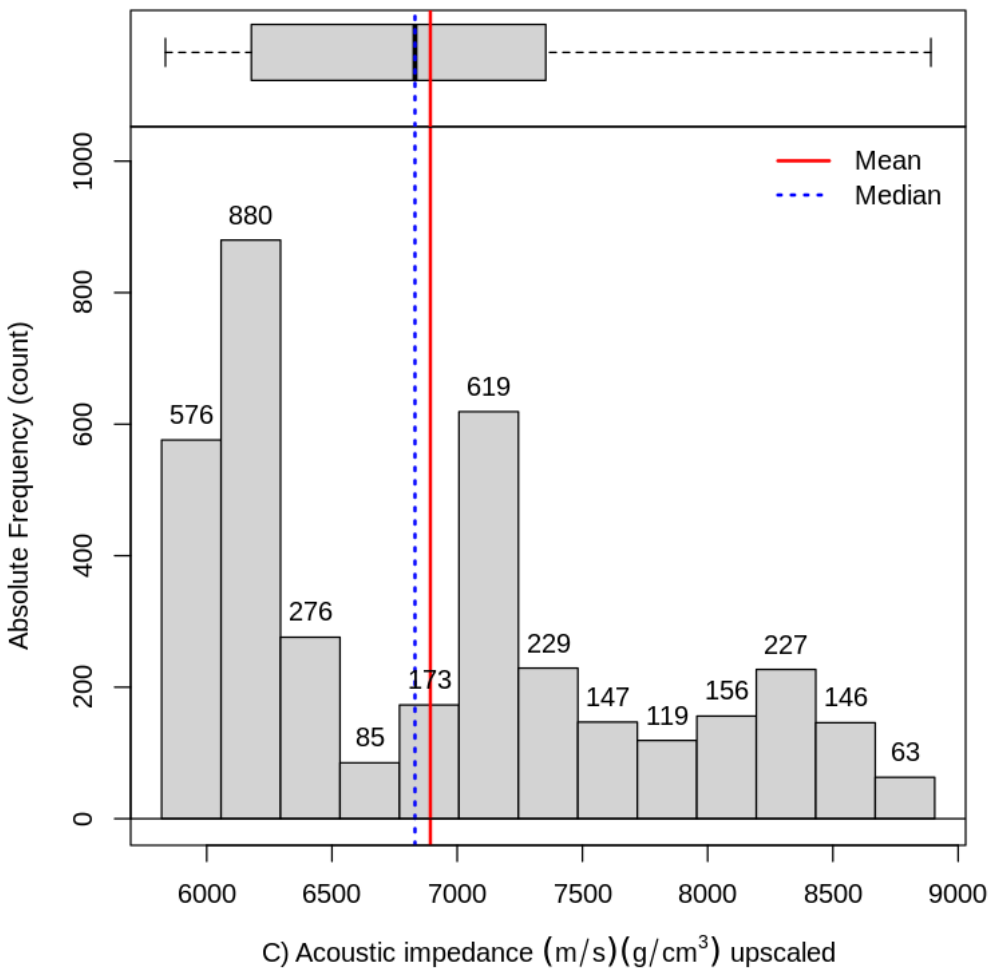
The minimum value is 5,835.0969, which is 749.0897 $(m/s)(g/cm^3)$ higher than the minimum recorded in the geophysical well log. The maximum is 8,891.2124, which is 2,770.2518 $(m/s)(g/cm^3)$ lower than the maximum reported in the geophysical well log. This suggests that the scaling significantly affected the representation of the high-impedance zone.

[20]:
```
#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
```

A data.frame: 14 × 2

	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	-0.8097



However, the subsampled acoustic impedance values do not reflect the bimodal behavior observed in the scaled log. The histogram does not show any outliers.

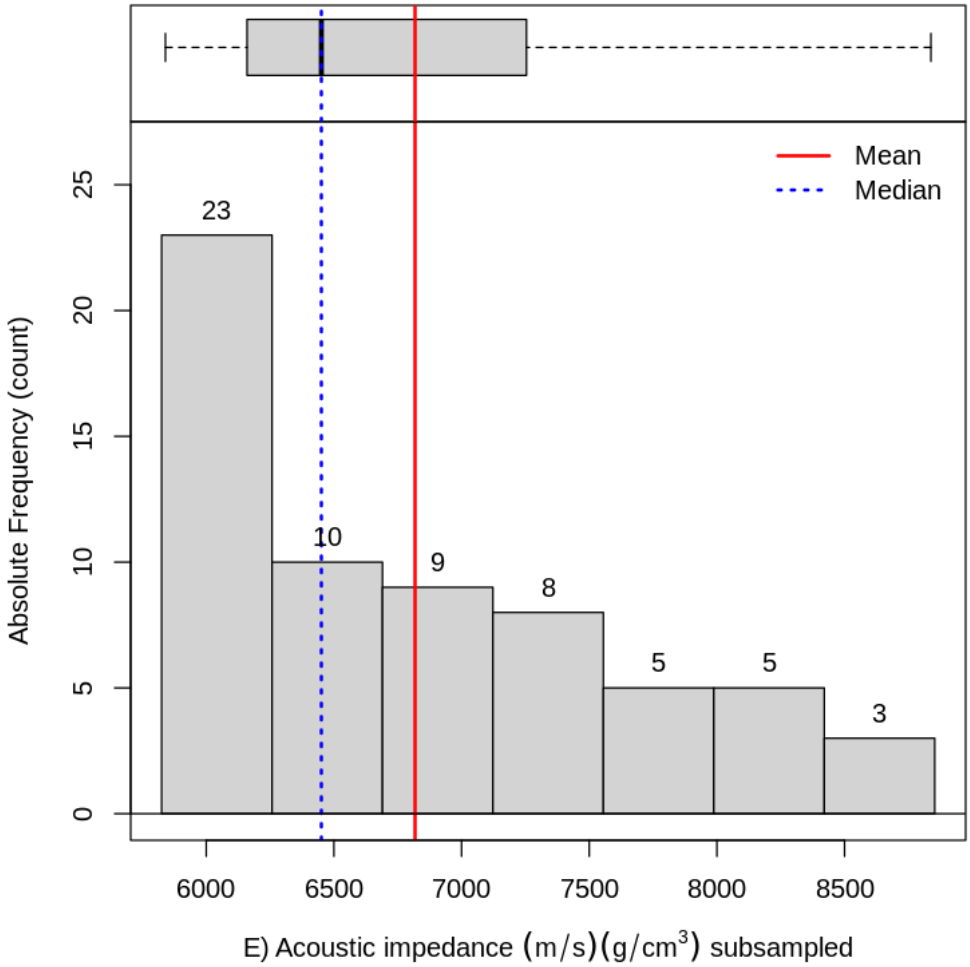
The minimum value is 5,840.1241 $(m/s)(g/cm^3)$, which is 5.0272 $(m/s)(g/cm^3)$ higher than that of the scaled log. The maximum value is 8,838.1850, which is 53.0274 $(m/s)(g/cm^3)$ lower than the scaled log.

[21]:
```
HistBoxplot(Ip_like_SUB,nbins = nsp, mean = Ip_like_SUB_Stat[5,2], median =␣
↪Ip_like_SUB_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_SUB_Stat
```

A data.frame: 14 × 2

	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	687258.8261
desvs	Standard Deviation	829.0108
CVs	Variation Coeff.	0.1216
simetrias	Skewness	0.6848
curtosiss	Kurtosis	-0.6026

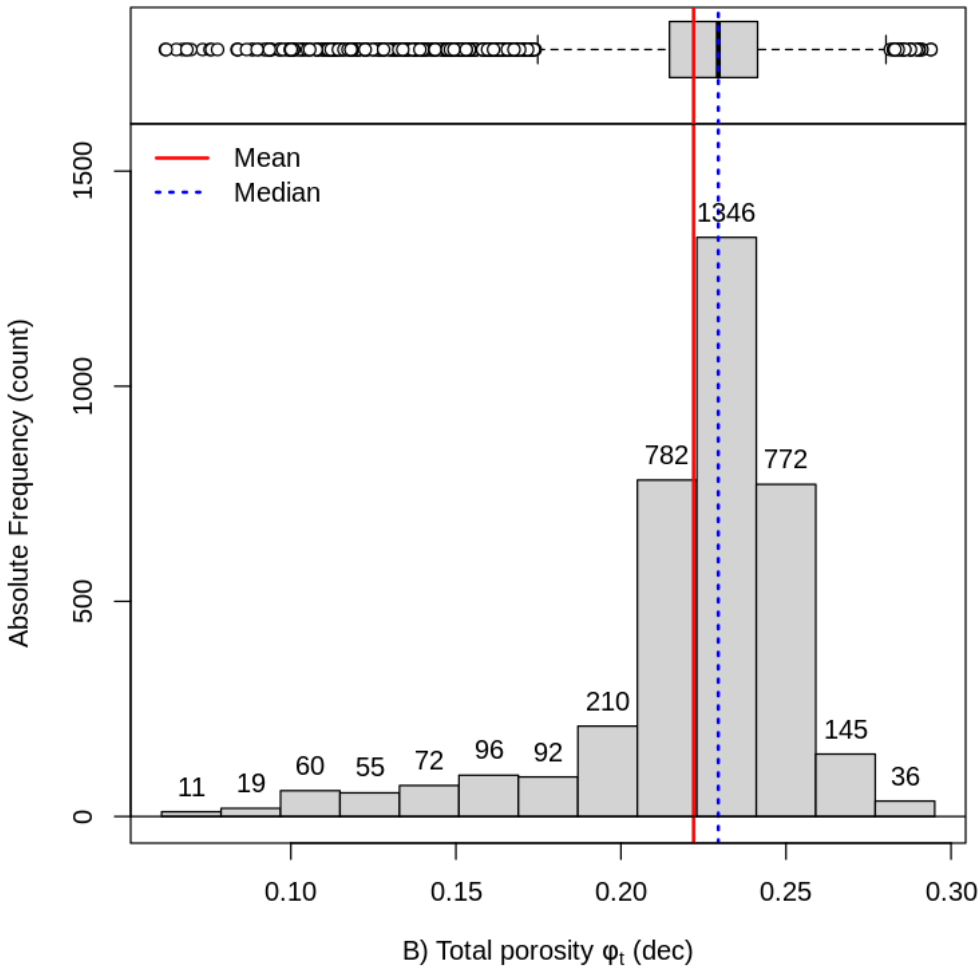


The geophysical well log of total porosity exhibits high negative skewness; most of the outliers are located on the left side and account for less than 10% of the total number of samples. The minimum value is 0.0620, the maximum is 0.2939, the mean is 0.2220, and the median is 0.2295.

[22]:
```
#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
```

A data.frame: 14 × 2

	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	4.0040

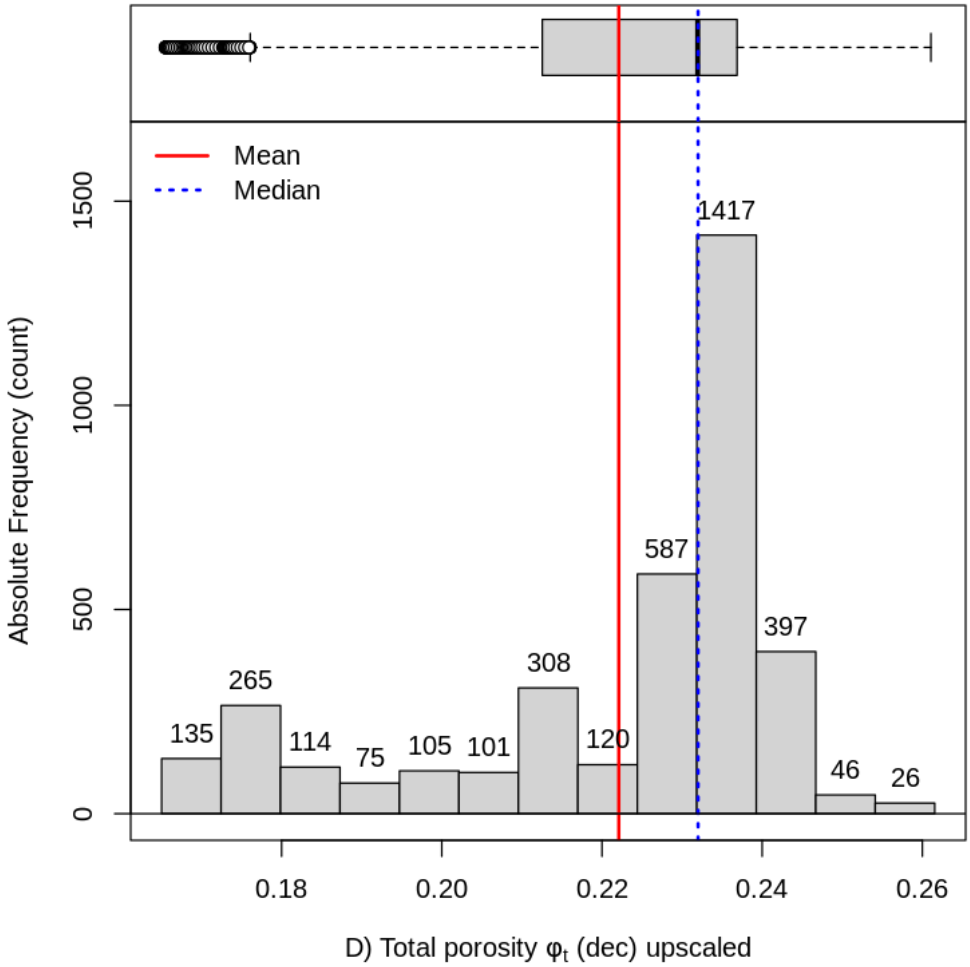


The total porosity log scaled using the moving window method exhibits negative skewness, although it is less pronounced than that observed in the geophysical well log of total porosity. Outliers are located on the left side of the boxplot. The minimum value is 0.1655, which is 0.1035 higher than that of the geophysical well log, and the maximum is 0.2611, which is 0.0328 lower than the well log. When comparing the effect of scaling on the geophysical well logs, the scaling had a greater impact on the high values of acoustic impedance and on the low values of total porosity.

[23]:
```
#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
```

A data.frame: 14 × 2

	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	0.2434

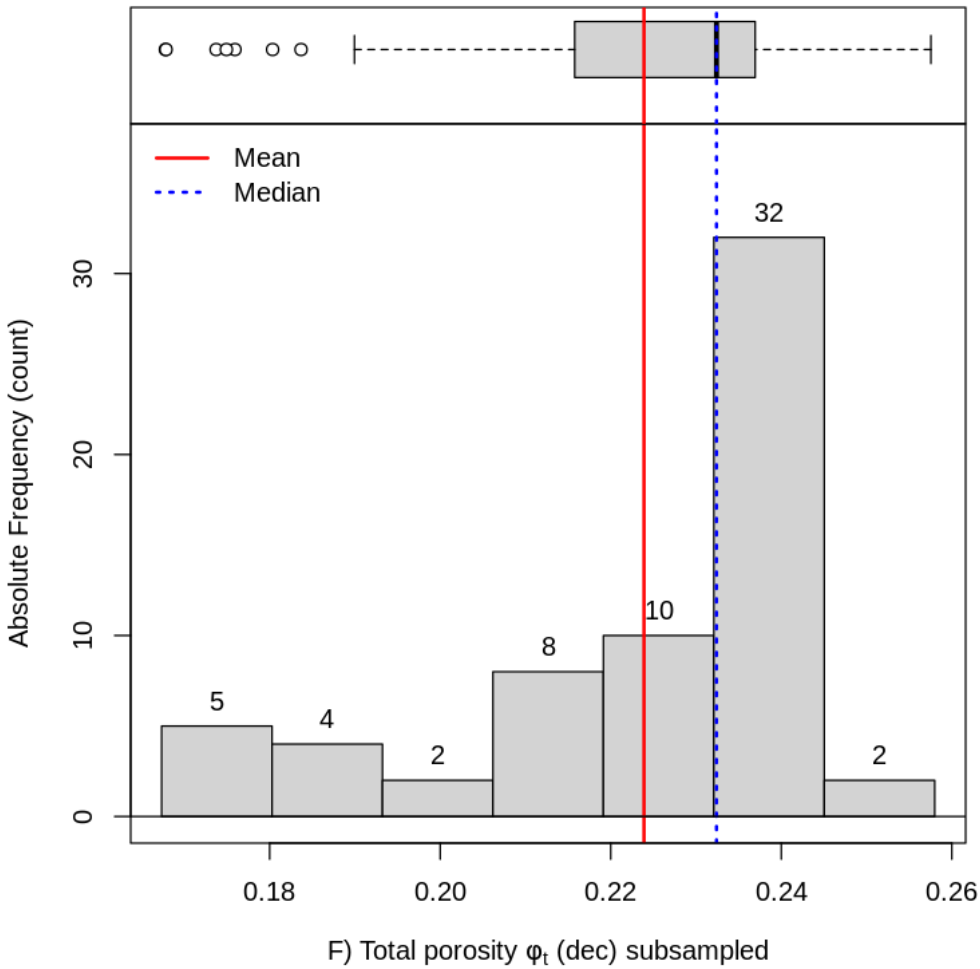


In the case of the subsampled total porosity log, the skewness is quite similar to that estimated in the scaled total porosity log. Outliers are located on the left side of the boxplot. The minimum value is 0.1678, which is 0.0023 higher than the scaled log, and the maximum is 0.2575, which is 0.0036 lower than the scaled log. The differences are very small.

[24]:
```
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
```

A data.frame: 14 × 2

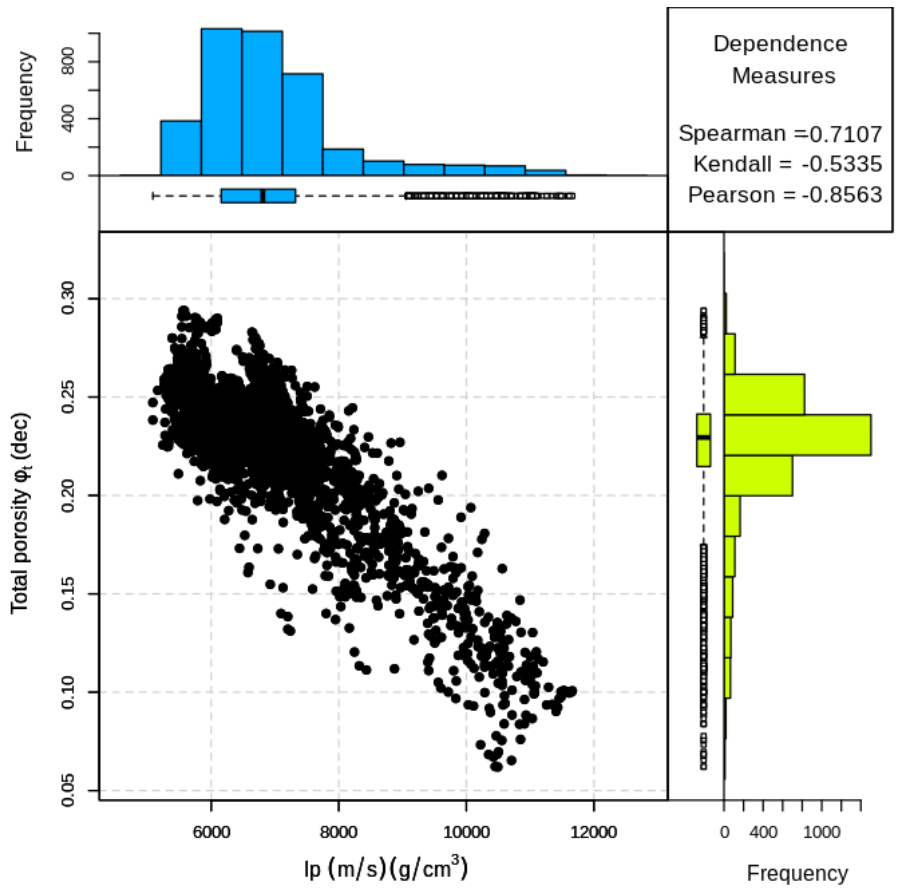
	Statistics <chr>	Values <dbl>
muestras	n	63.0000
minimos	Minimum	0.1678
cuantiles1	1st. Quartile	0.2158
medianas	Median	0.2324
medias	Mean	0.2239
cuantiles3	3rd. Quartile	0.2369
maximos	Maximum	0.2575
rangos	Rank	0.0898
rangosInt	Interquartile Rank	0.0212
varianzas	Variance	0.0005
desvs	Standard Deviation	0.0216
CVs	Variation Coeff.	0.0964
simetrias	Skewness	-1.3036
curtosiss	Kurtosis	0.8845



The dependence measures and corresponding scatterplots for (AI, ϕ_t) are presented. At the well scale, the Spearman, Kendall, and Pearson coefficients are -0.7107, -0.5335, and -0.8563, respectively. After upscaling and subsampling, the dependence measures increase, particularly for Spearman and Kendall, which increase by approximately 0.2. However, the differences between the upscaled and subsampled cases remain small, less than 0.1, suggesting that the overall dependence structure remains stable despite changes in the univariate statistics.

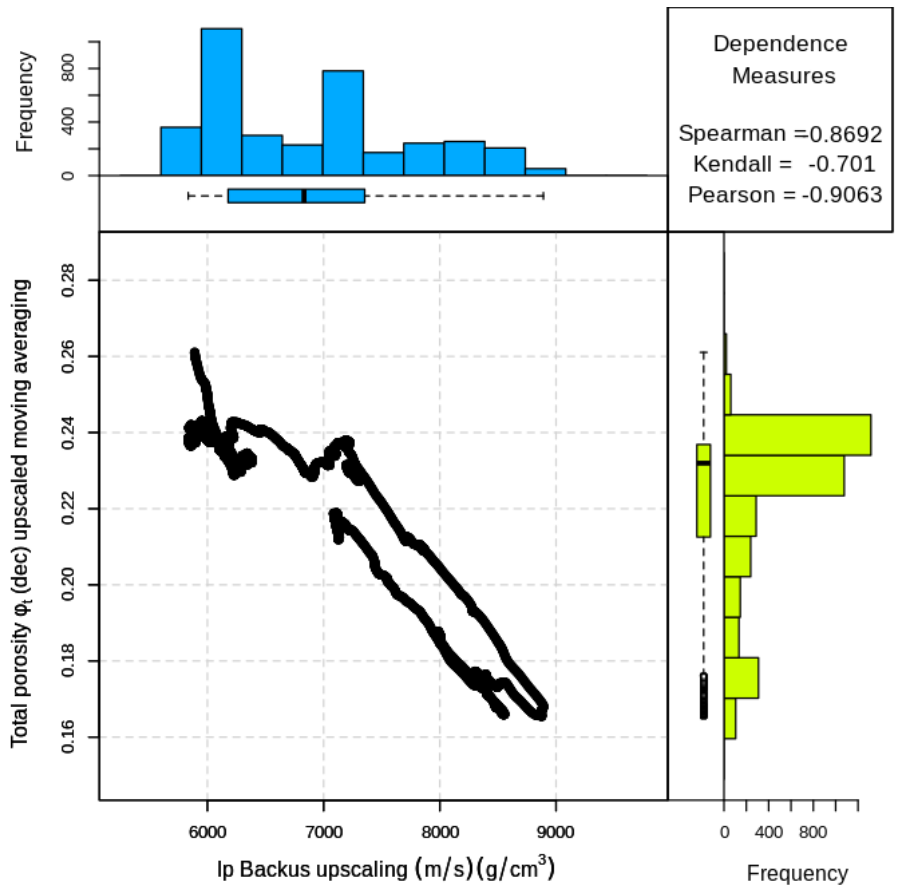
Measure of dependence	**(ϕ_t, Ip)**	**(ϕ_t, Ip) upscaled**	**(ϕ_t, Ip) subsampled**
Pearson	-0.8563	-0.9063	-0.9026
Spearman	-0.7107	-0.8692	-0.8673
Kendall	-0.8563	-0.7010	-0.6989

[25]:
```
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)")))
```



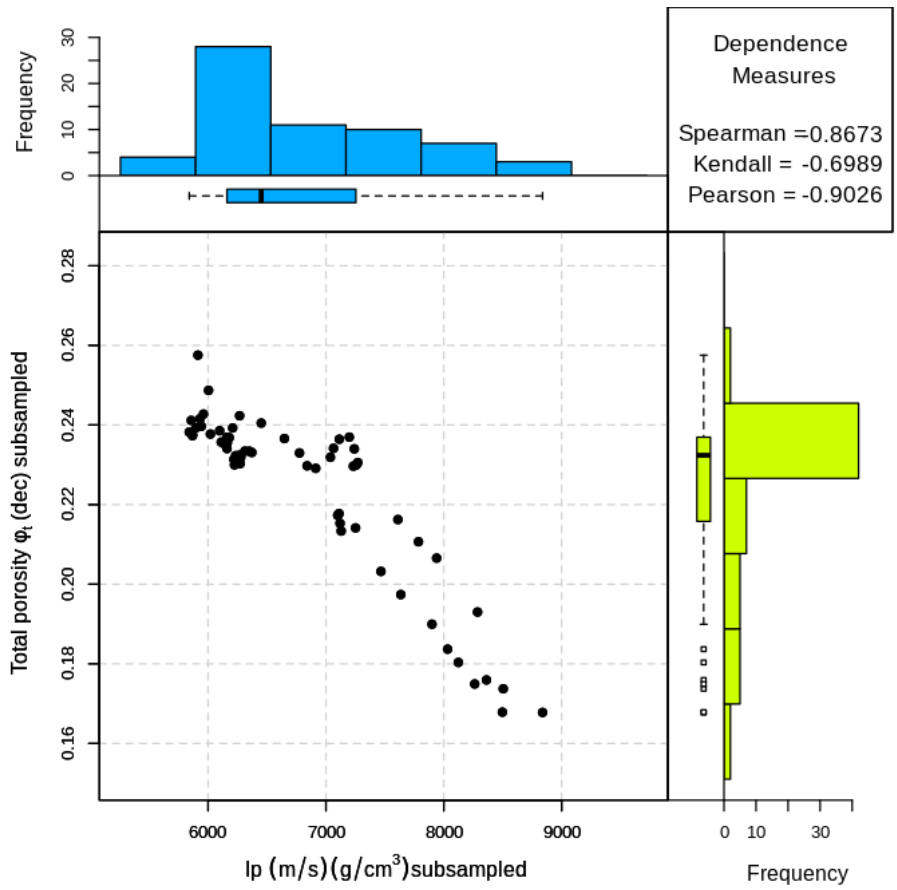
The dependency among the scaled geophysical logs

```
[26]: ScatterPlot(Ip_like , Phit_like, ns,
            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")))
```



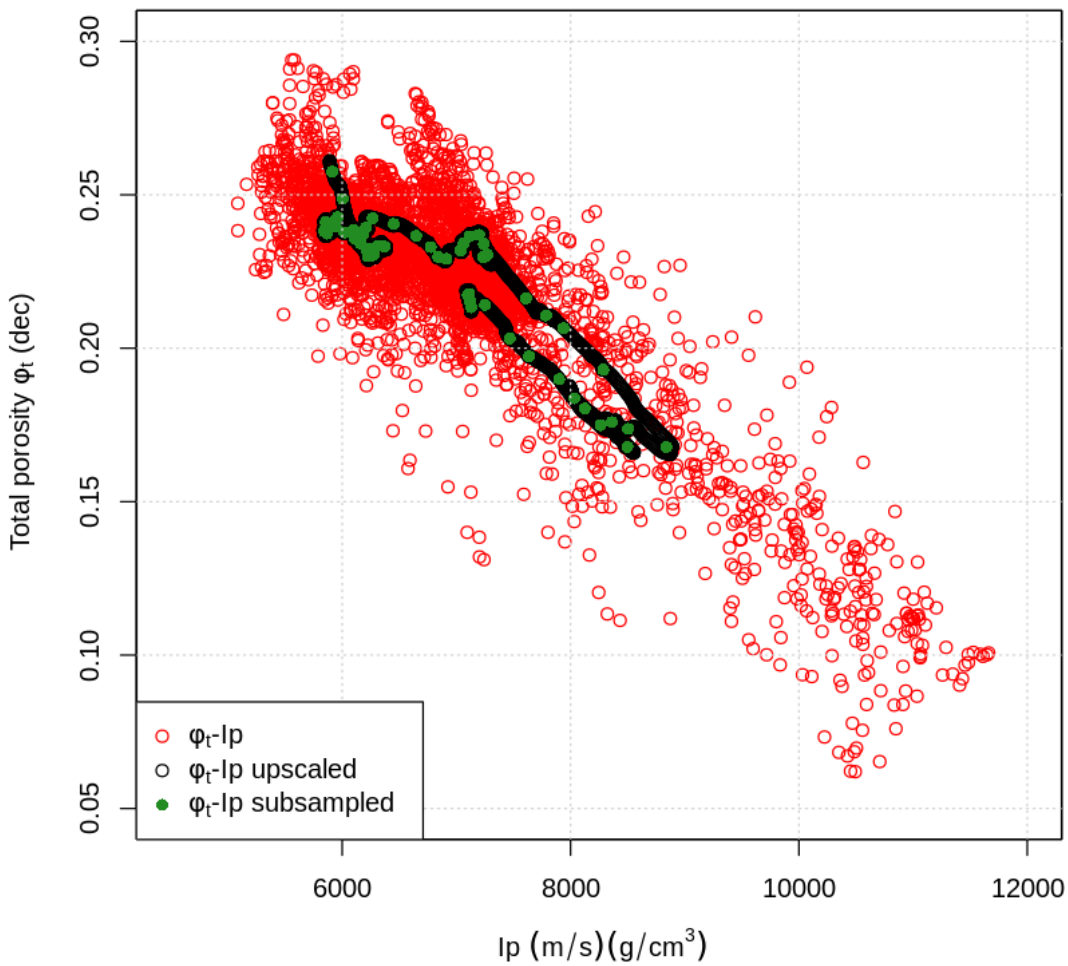
The dependency among the scaled geophysical logs and the subsampled geophysical logs.

[27]:
```
ScatterPlot(Ip_like_SUB , Phit_like_SUB,nsp,
            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")))
```



[28]:
```
plot(Ip , Phit, xlim=c(4500, 12000),ylim=c(0.05, 0.
↪30),col=("red"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Ip_like , Phit_like,xlim=c(4500, 12000),ylim=c(0.05, 0.
↪30),xaxt='n',yaxt='n',
     ann=FALSE)
par(new=TRUE)
plot(Ip_like_SUB , Phit_like_SUB, xlab = expression(paste(" Ip ",(m/s)(g/cm^3))),
        ylab = expression(paste(" Total porosity ",phi[t], "␣
↪(dec)")),xlim=c(4500, 12000),
     ylim=c(0.05, 0.30),
     col=("forestgreen"), pch=16)
```

```r
grid()
legend(x = "bottomleft", legend = c(expression(paste(phi[t],"-Ip ")),
                                    expression(paste(phi[t],"-Ip upscaled")),
                                   expression(paste(phi[t],"-Ip subsampled"))),
       col = c("red","black","forestgreen"),pch=c(1,1,16))
```



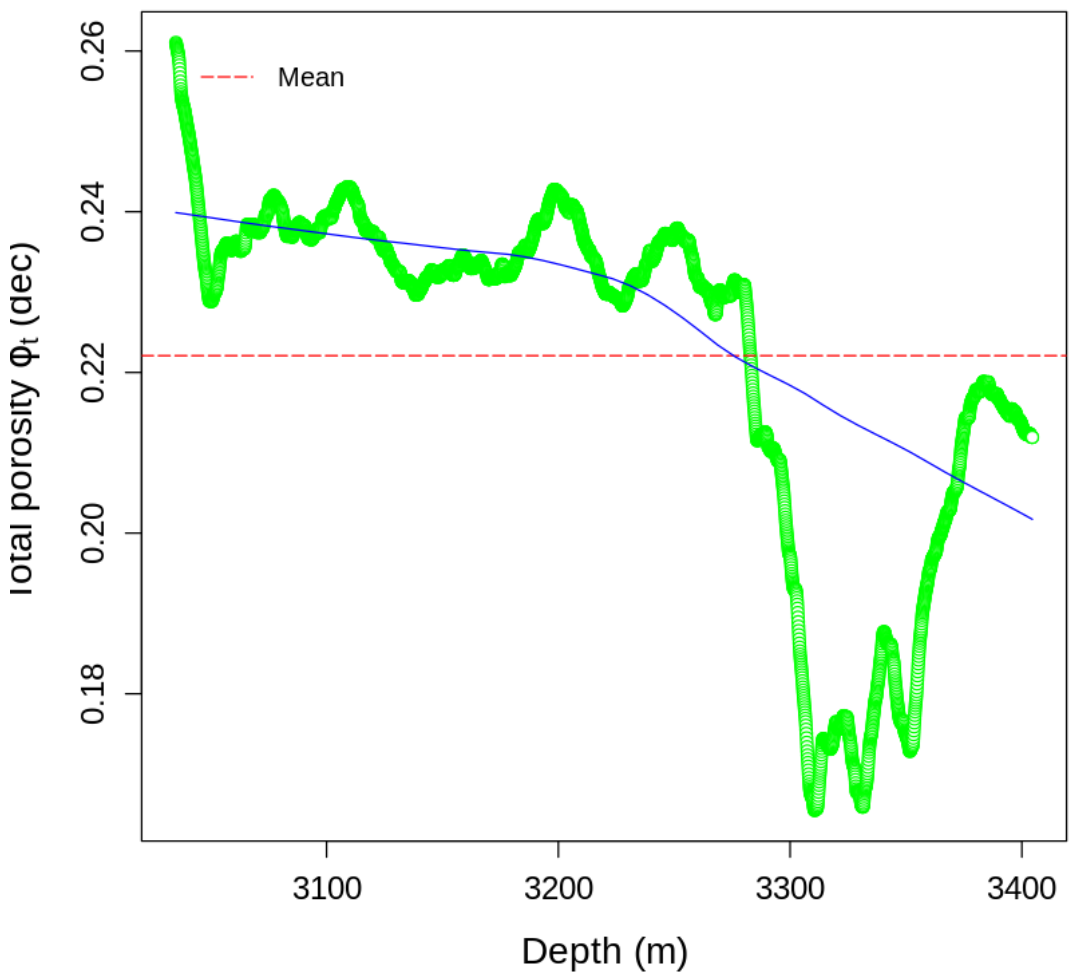
7.2 Variographic Analysis

In this step, the variogram model of the scaled and subsampled logs will be estimated, considering both the time and depth domains.

7.3 Variographic Analysis of Upscaled Total Porosity in the Depth Domain

Following the geostatistical analysis workflow, the process begins by assessing the presence of trend in the variable. The first test consists of estimating the median regression. For the scaled total porosity, the regression shows potential evidence of a trend throughout the entire log.

[29]:
```
MedianReg(Depth, Phit_like, Xlab= "Depth (m)", Varlab= expression(paste(" Total␣
↪porosity ",varphi[t], " (dec)")),
          deltay=500)
```

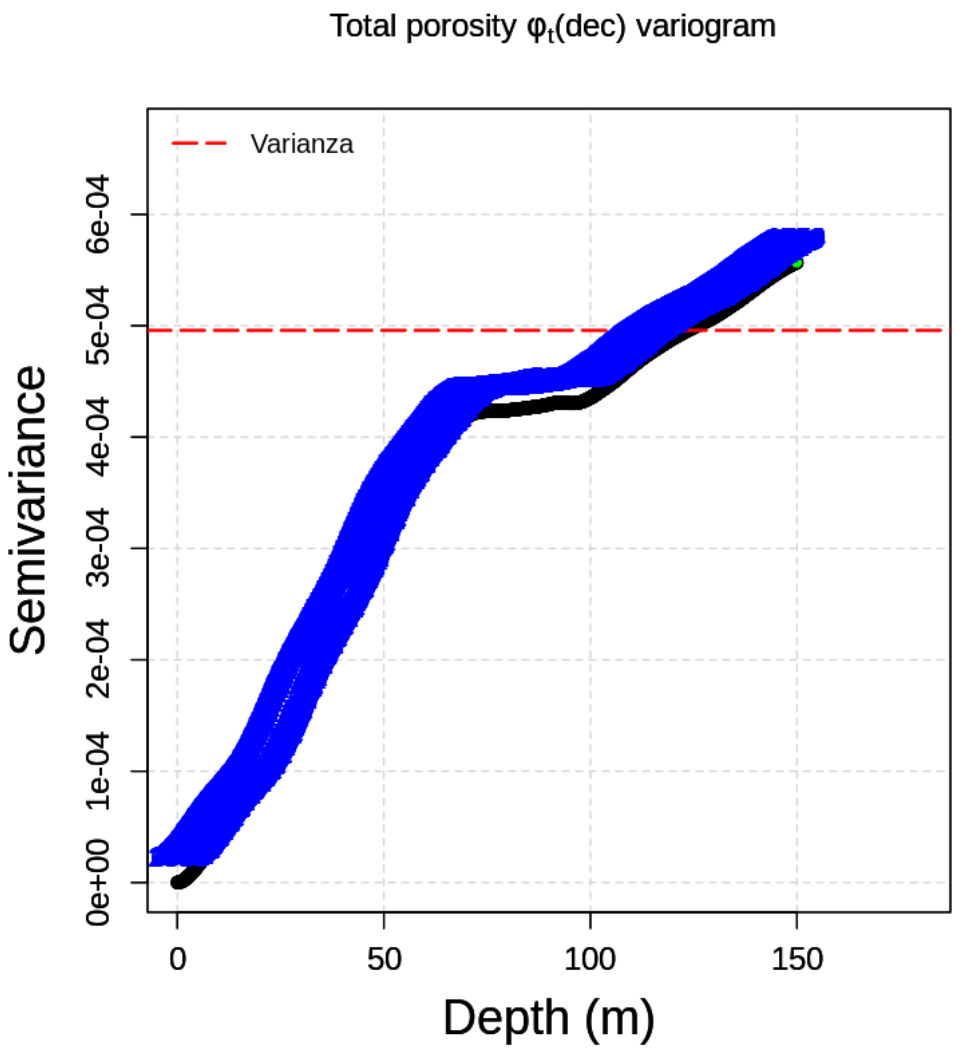


[30]:
```
N_lags<-1500
lag_value <- 0.1
YCoord=data.frame(matrix(numeric(Depth_Stat[1,2])))
```

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

The result of the semivariogram estimation indicates that the trend is not significant, as the semivariance appears to level off at approximately 70 meters, with a semivariance value of 4.2×10^{-4}.

[31]:
```
Phit_like_VarioEstimation<-Variograma(CoorX=Depth, CoorY=YCoord,␣
↪Variable=Phit_like,
                                Direccion=90, Tol=0, NIntervalos=N_lags,
                                Lags=lag_value,graph=TRUE,
                                MainTitle=expression(paste(" Total porosity␣
↪",phi[t], "(dec) variogram"))
                                  , xlab="Depth (m)",
                                ylab = "Semivariance")
```

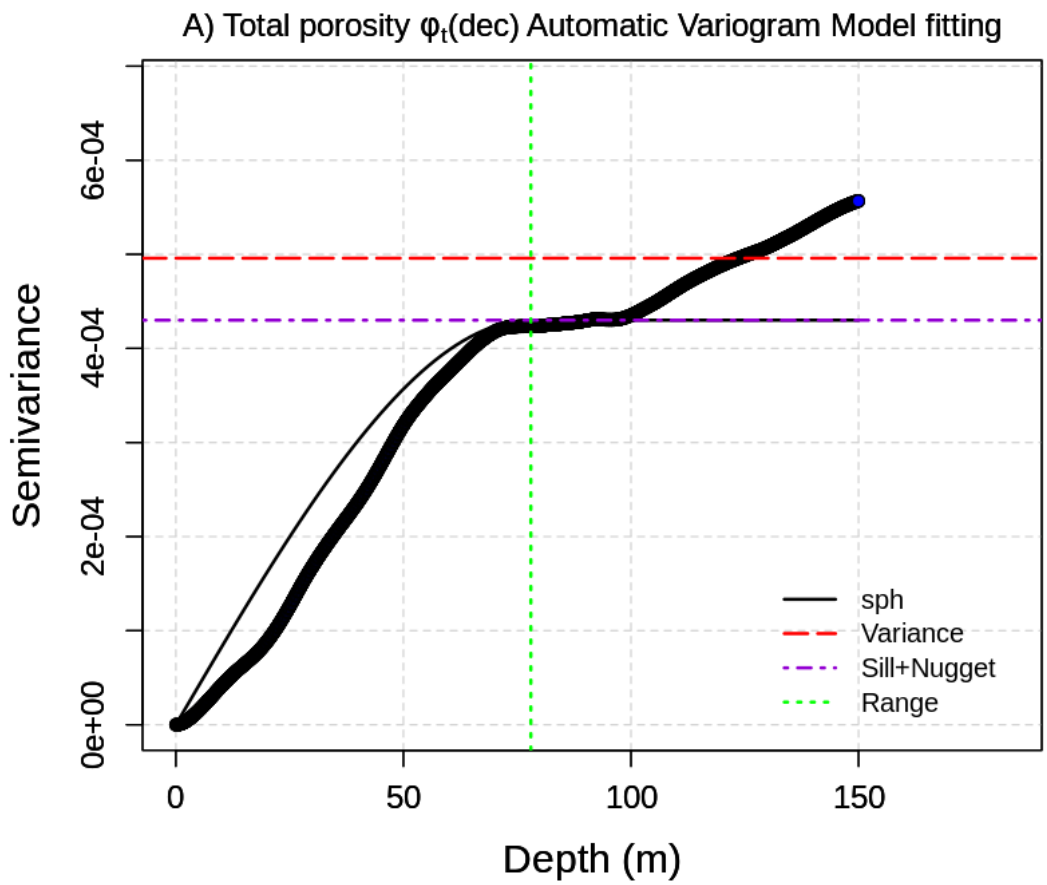


A manual fit was performed to select the variogram model that best matches the empirical semi-variogram. In this case, a spherical variogram model was used, with a nugget effect of 0, a sill plus nugget of 0.00043, and a range of 78 meters.

[32]:
```
vario_model<- 2
nugget<- 0
sill_with_nugget<- 0.00043
range <- 78
```

[33]:
```
Phit_like_EyeModelVarioFit<-EyeModel(Depth, YCoord,
                                     Phit_like, 0, 90, N_lags, lag_value, 1,
                                     vario_model, nugget, sill_with_nugget,␣
 ↪range,graph=TRUE,
                                     expression(paste("A) Total porosity ",phi[t],
                                                    "(dec) Automatic Variogram␣
 ↪Model fitting")),xlab="Depth (m)")
```

```
variog: computing omnidirectional variogram
```



Model	Nugget	Sill+Nugget	Range	MSE
sph	0.0000	0.0004	78.0000	0.0000

[34]:
```
Phit_like_CrossValid<- CrossValidation(Depth, YCoord,
                                       Phit_like, vario_model, nugget,␣
↪sill_with_nugget,
                                          range,MaxAnis=0, proporcion=1)
```

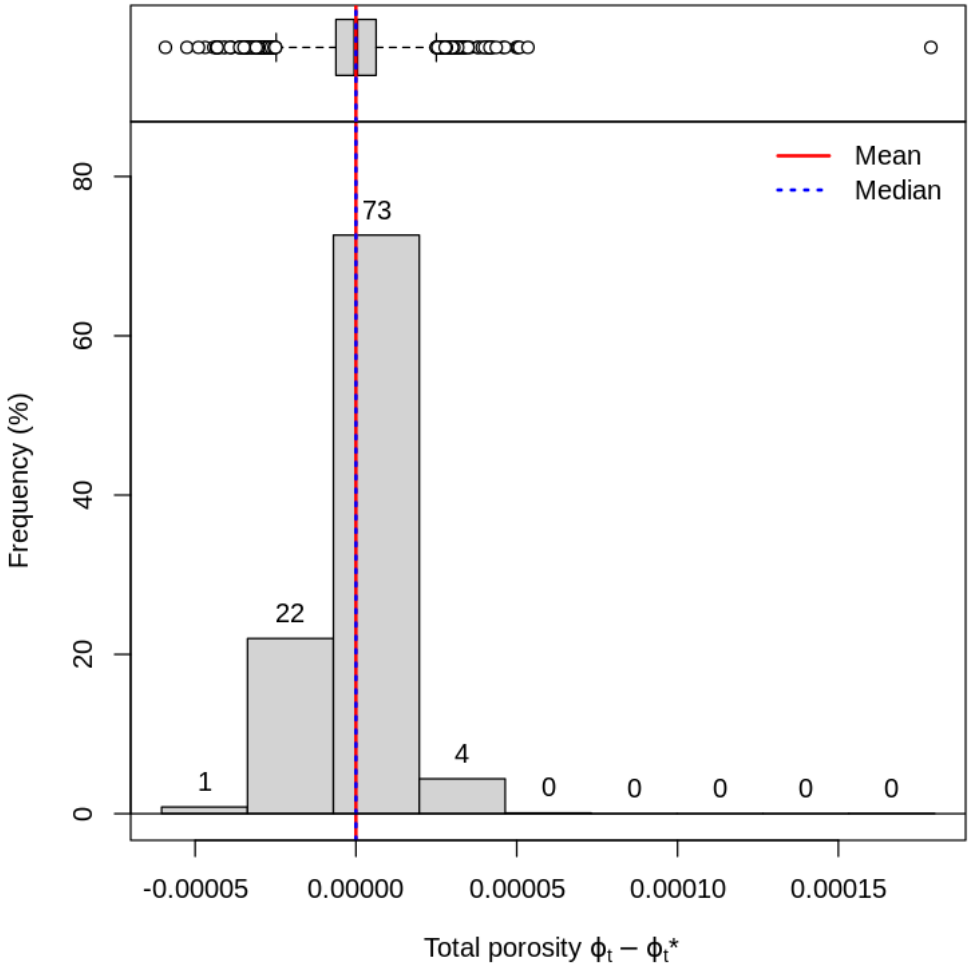
Upon validating the variogram model using the cross-validation approach, it can be observed that the proposed model is acceptable.

[35]:
```
Phit_like_CrossValid_Stat<- Val_Estadisticos(Phit_like_CrossValid[1:
↪length(Phit_like),c(3,4,5)])
Phit_like_CrossValid_Stat
```

A data.frame: 13 × 3

	Z <dbl>	Z* <dbl>	Z-Z* <dbl>
No_muestras	3696.00000	3696.00000	3696.00000
Minimo	0.16550	0.16552	-0.00006
Cuartil_1er	0.21255	0.21254	-0.00001
Mediana	0.23195	0.23196	0.00000
Media	0.22209	0.22209	0.00000
Cuartil_3er	0.23687	0.23687	0.00001
Maximo	0.26107	0.26090	0.00018
Rango	0.09557	0.09538	0.00024
Rango_Intercuartil	0.02431	0.02432	0.00001
Varianza	0.00050	0.00050	0.00000
Desv_Estandar	0.02227	0.02227	0.00001
Simetria	-1.21389	-1.21388	0.84588
Curtosis	0.24336	0.24310	14.59867

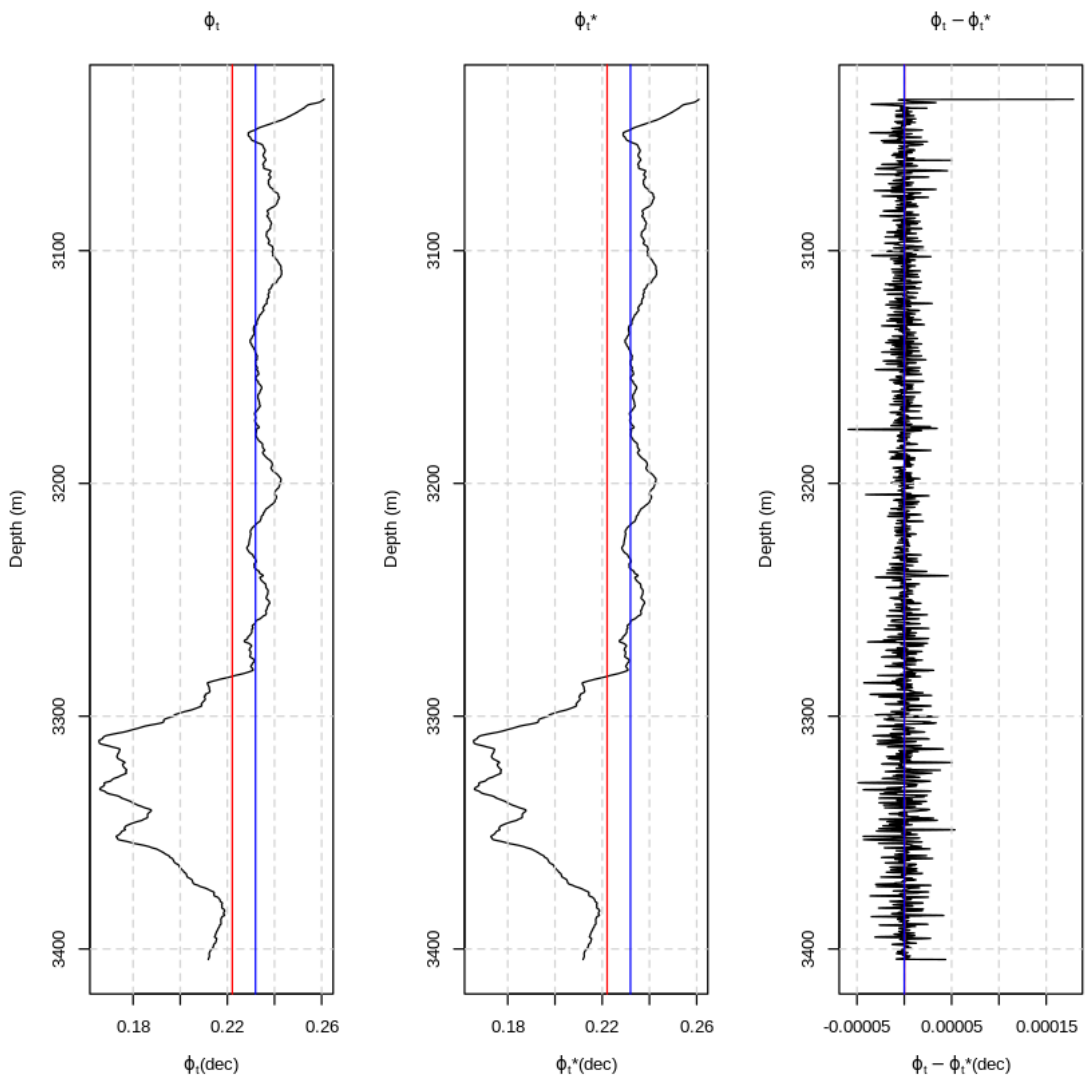
[36]:
```
HistBoxplot(x=Phit_like_CrossValid[,5], mean = Phit_like_CrossValid_Stat[5,3],
            median = Phit_like_CrossValid_Stat[4,3], main ="",
            xlab = expression(paste(" Total porosity␣
 ↪",varphi[t]-varphi[t],"*")), ylab = "Frequency (%)",
            AbsFreq = FALSE, PercentFreq = TRUE )
```



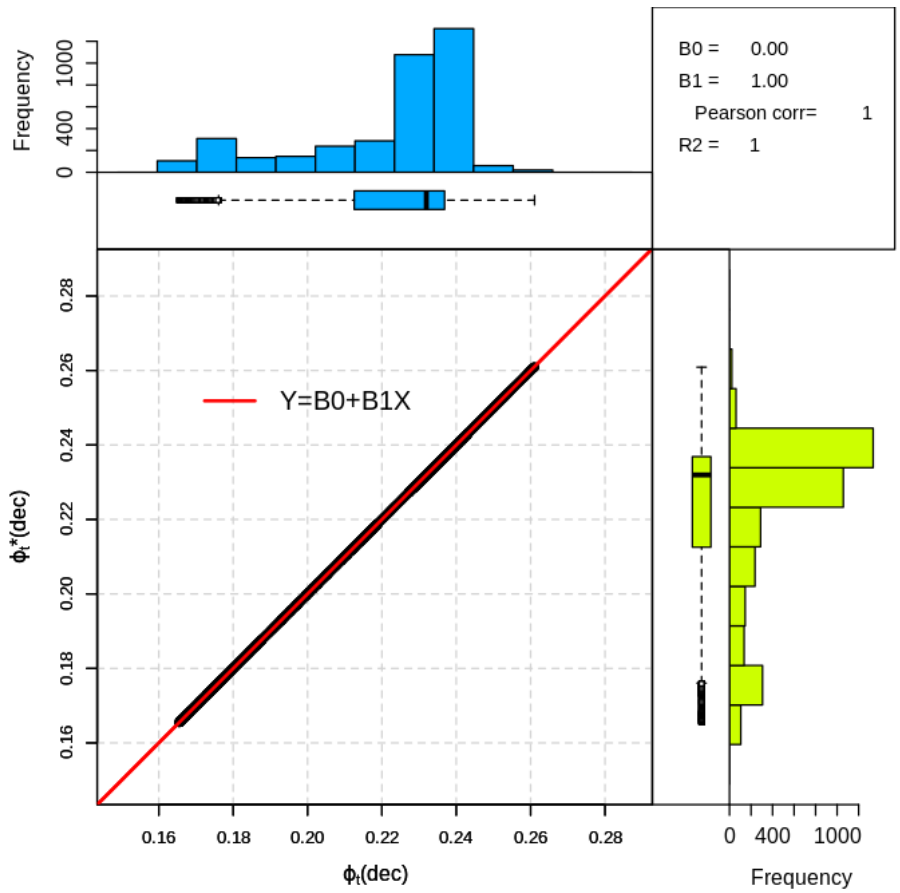
[37]:
```
Phit_like_est<-Phit_like_CrossValid[,4]
Phit_like_est_Stat<-Estadisticas(Phit_like_est)
Phit_like_diff<-Phit_like_CrossValid[,5]
Phit_like_diff_Stat<-Estadisticas(Phit_like_diff)
```

[38]:
```
LogPlot(depth=Depth, n_logs=3,
        log_list=list(Phit_like,Phit_like_est,Phit_like_diff),
      ␣
↪log_name_list=list(expression(paste(varphi[t])),expression(paste(varphi[t],"*")),
                          expression(paste(varphi[t]-varphi[t],"*"))),
      ␣
↪log_xlabel_list=list(expression(paste(varphi[t],"(dec)")),expression(paste(varphi[t],"*(dec)'
```

```
                                expression(paste(varphi[t]-varphi[t],"*(dec)"))),
        plot_mean=TRUE, plot_median=TRUE)
```



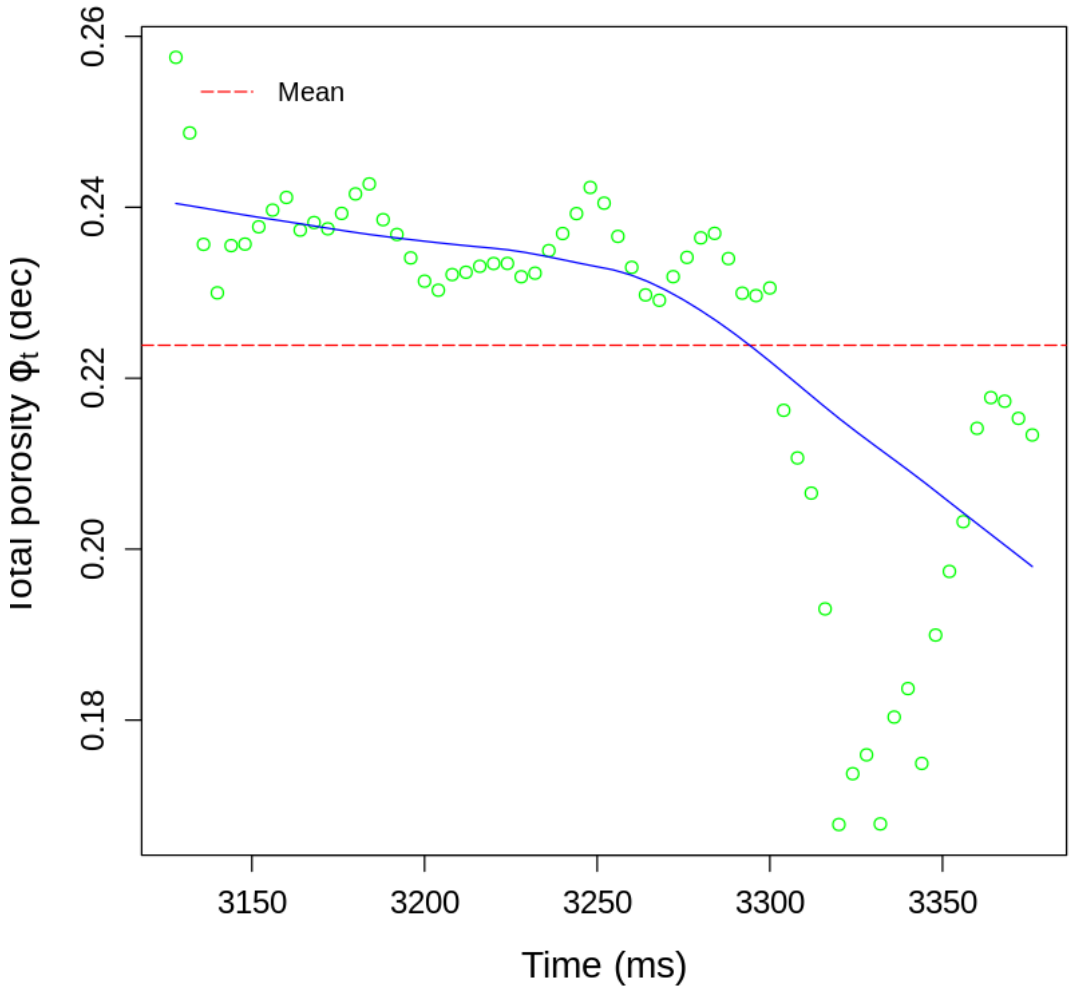
[39]:
```
scaterplotReg(Phit_like , Phit_like_est, ns,
              Xmin = Phit_like_Stat[2,2], Xmax = Phit_like_Stat[7,2],
              Ymin = Phit_like_est_Stat[2,2],Ymax = Phit_like_est_Stat[7,2],
              XLAB = expression(paste(varphi[t],"(dec)")), YLAB =␣
 ↪expression(paste(varphi[t],"*(dec)")))
```



7.3.1 Variographic analysis time domain

The variogram model is now estimated using the subsampled total porosity samples in the time domain. As with the variable in the depth domain, the median regression indicates potential evidence of a trend.

[40]:
```
MedianReg(time_like_SUB, Phit_like_SUB, Xlab= "Time (ms)",
          Varlab= expression(paste(" Total porosity ",varphi[t], " (dec)")),
          deltay=500)
```

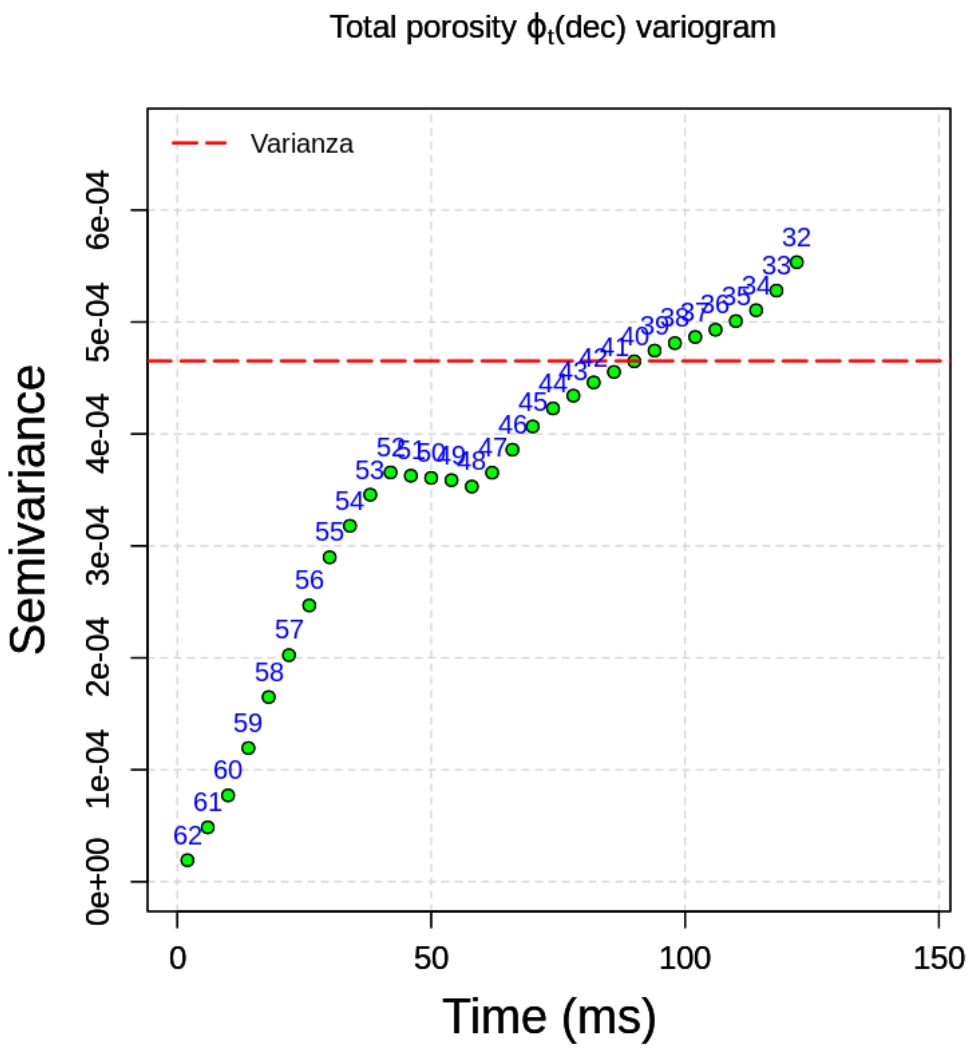


[41]:
```
N_lags_SUB<-31
lag_value_SUB <- 4
YCoord=numeric(time_like_SUB_Stat[1,2])
#YCoord_Stat<-Estadisticas(YCoord)
```

However, this trend is not apparent in the semivariance plot, where a potential sill appears to level off at approximately 3.8×10^{-4}.

[42]:
```
Phit_SUB_VarioEstimation<-Variograma(CoorX=time_like_SUB, CoorY=YCoord,␣
↪Variable=Phit_like_SUB,
                                Direccion=90, Tol=0, NIntervalos=N_lags_SUB,
                                Lags=lag_value_SUB,graph=TRUE,
```

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

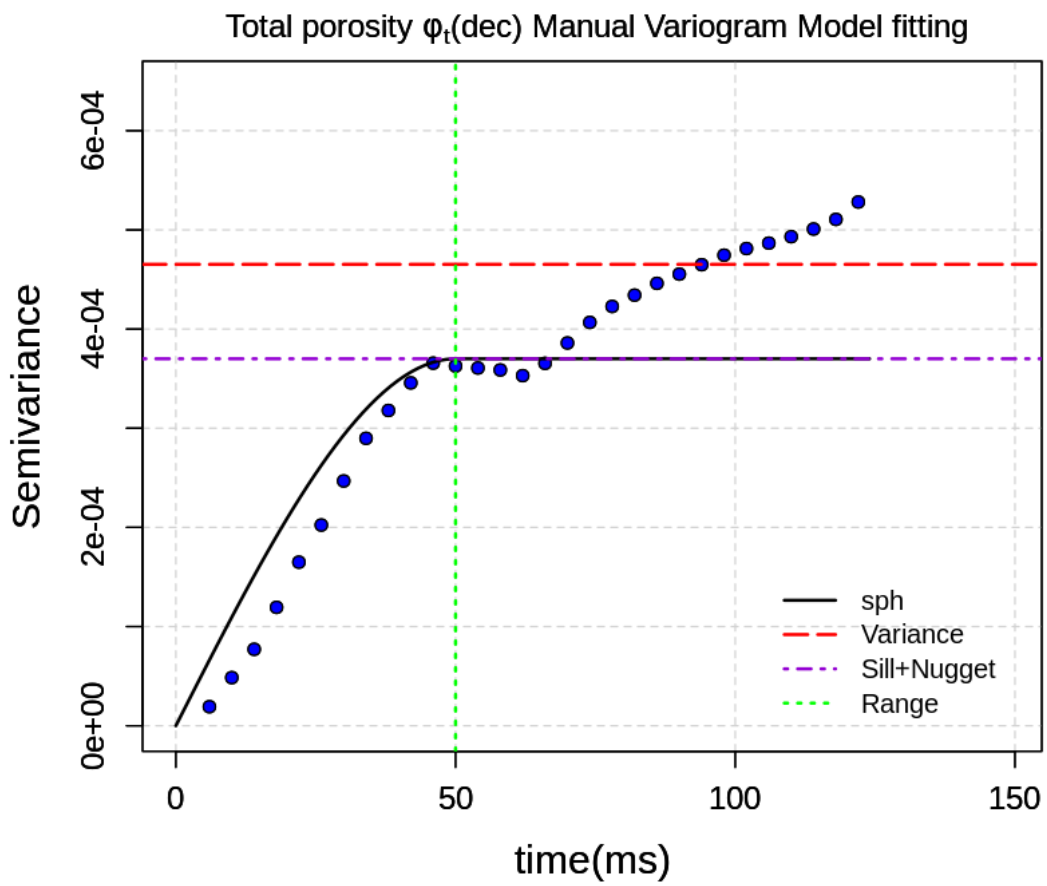


For the variogram model fitting, a spherical model is considered, with a nugget of 0, a sill plus nugget of 0.00037, and a range of 50 milliseconds. This model appears to level off appropriately with the proposed sill plus nugget value.

[43]:
```
vario_model_SUB<- 2
nugget_SUB<- 0
sill_with_nugget_SUB<- 0.00037
range_SUB <- 50
```

[44]:
```
Phit_SUB_EyeModelVarioFit<-EyeModel(time_like_SUB, YCoord,
                                    Phit_like_SUB, 0, 90, N_lags_SUB,␣
↪lag_value_SUB, 1,
                                    vario_model_SUB, nugget_SUB,␣
↪sill_with_nugget_SUB, range_SUB,graph=TRUE,
                                    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	0.0004	50.0000	0.0000

[45]:
```
Phit_SUB_CrossValid<- CrossValidation(time_like_SUB, YCoord,
                                      Phit_like_SUB, vario_model_SUB, nugget_SUB,
```

```
                                        sill_with_nugget_SUB,
                                    range_SUB, MaxAnis=0, proporcion=1)
```

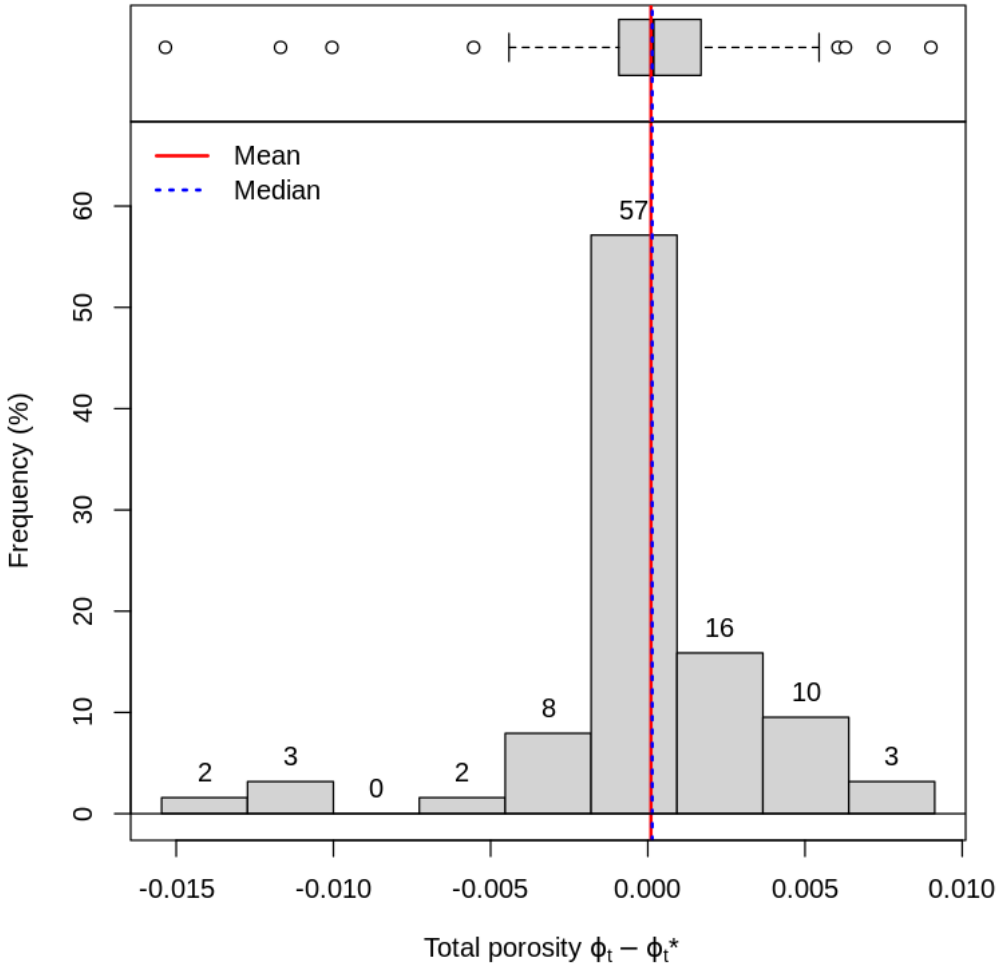
The variogram model fit, evaluated using the cross-validation method, indicates that the model is acceptable.

[46]:
```
Phit_SUB_CrossValid_Stat<- Val_Estadisticos(Phit_SUB_CrossValid[1:
 ↪length(Phit_like_SUB),c(3,4,5)])
Phit_SUB_CrossValid_Stat
```

A data.frame: 13 × 3

	Z <dbl>	Z* <dbl>	Z-Z* <dbl>
No_muestras	63.00000	63.00000	63.00000
Minimo	0.16778	0.17048	-0.01534
Cuartil_1er	0.21577	0.21571	-0.00092
Mediana	0.23240	0.23283	0.00014
Media	0.22386	0.22376	0.00010
Cuartil_3er	0.23694	0.23723	0.00169
Maximo	0.25754	0.24854	0.00900
Rango	0.08977	0.07806	0.02435
Rango_Intercuartil	0.02117	0.02151	0.00261
Varianza	0.00047	0.00044	0.00002
Desv_Estandar	0.02157	0.02099	0.00388
Simetria	-1.30363	-1.30248	-1.26324
Curtosis	0.88452	0.61788	4.94367

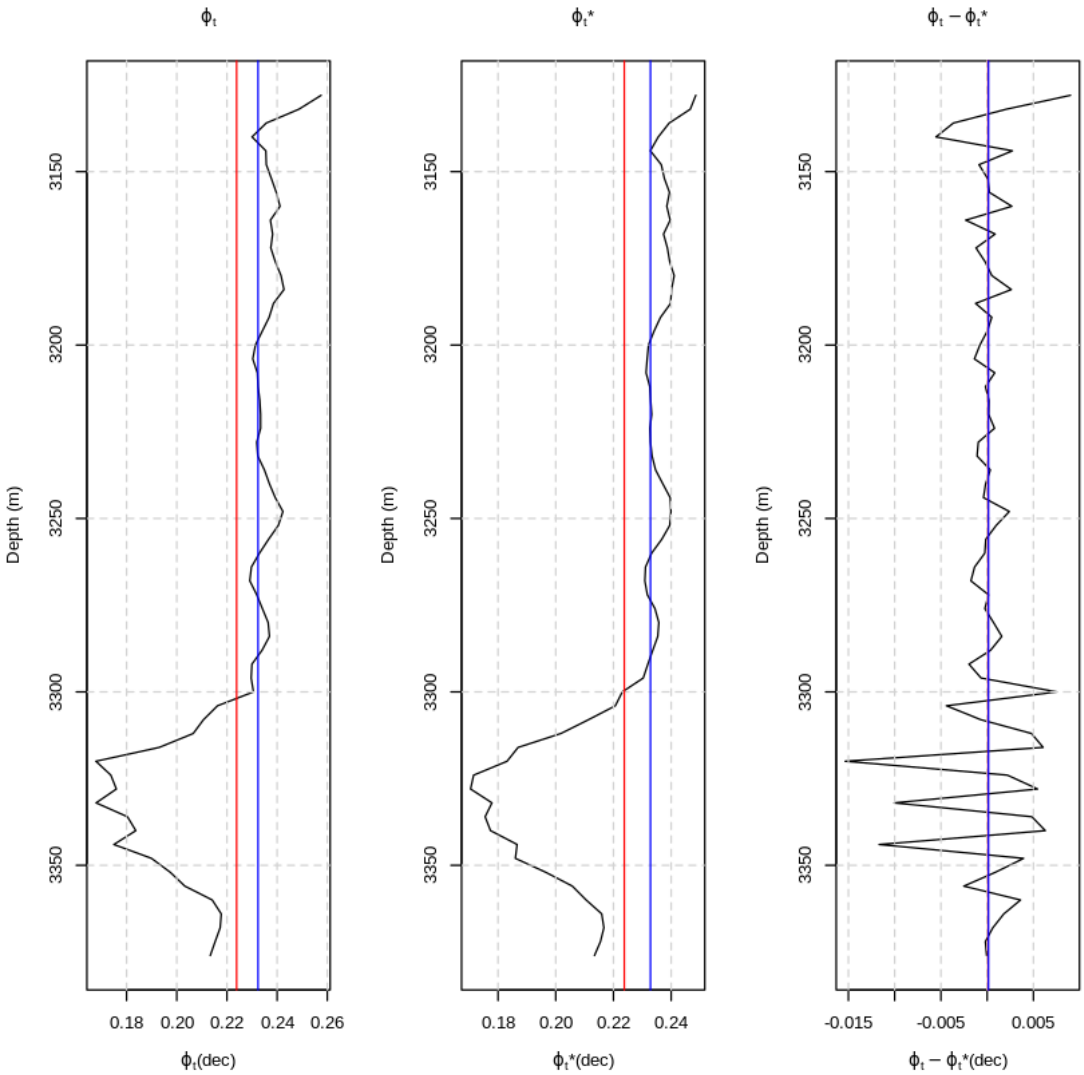
[47]:
```
HistBoxplot(x=Phit_SUB_CrossValid[,5], mean = Phit_SUB_CrossValid_Stat[5,3],
            median = Phit_SUB_CrossValid_Stat[4,3], main ="",
            xlab = expression(paste(" Total porosity␣
 ↪",varphi[t]-varphi[t],"*")), ylab = "Frequency (%)",
            AbsFreq = FALSE, PercentFreq = TRUE )
```



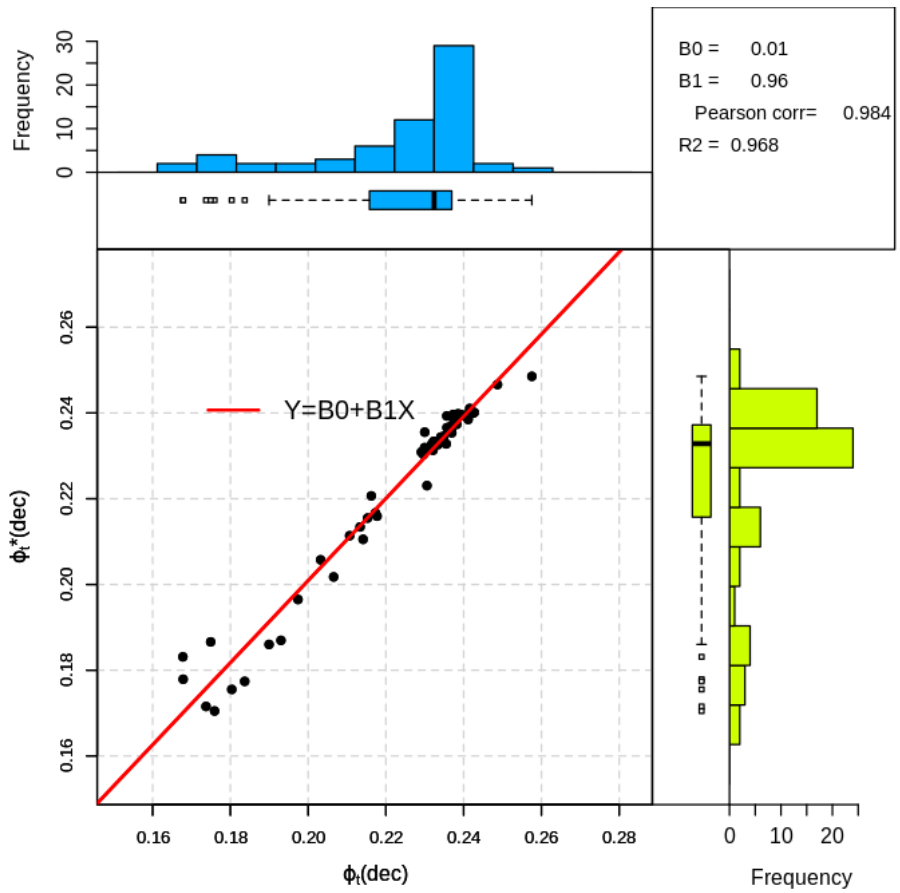
[48]:
```
Phit_SUB_est<-Phit_SUB_CrossValid[,4]
Phit_SUB_est_Stat<-Estadisticas(Phit_SUB_est)
Phit_SUB_diff<-Phit_SUB_CrossValid[,5]
Phit_SUB_diff_Stat<-Estadisticas(Phit_SUB_diff)
```

[49]:
```
LogPlot(depth=time_like_SUB, n_logs=3,
        log_list=list(Phit_like_SUB,Phit_SUB_est,Phit_SUB_diff),
      ␣
↪log_name_list=list(expression(paste(varphi[t])),expression(paste(varphi[t],"*")),
                          expression(paste(varphi[t]-varphi[t],"*"))),
      ␣
↪log_xlabel_list=list(expression(paste(varphi[t],"(dec)")),expression(paste(varphi[t],"*(dec)'
```

```
                                expression(paste(varphi[t]-varphi[t],"*(dec)"))),
        plot_mean=TRUE, plot_median=TRUE)
```



[50]:
```
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)")))
```



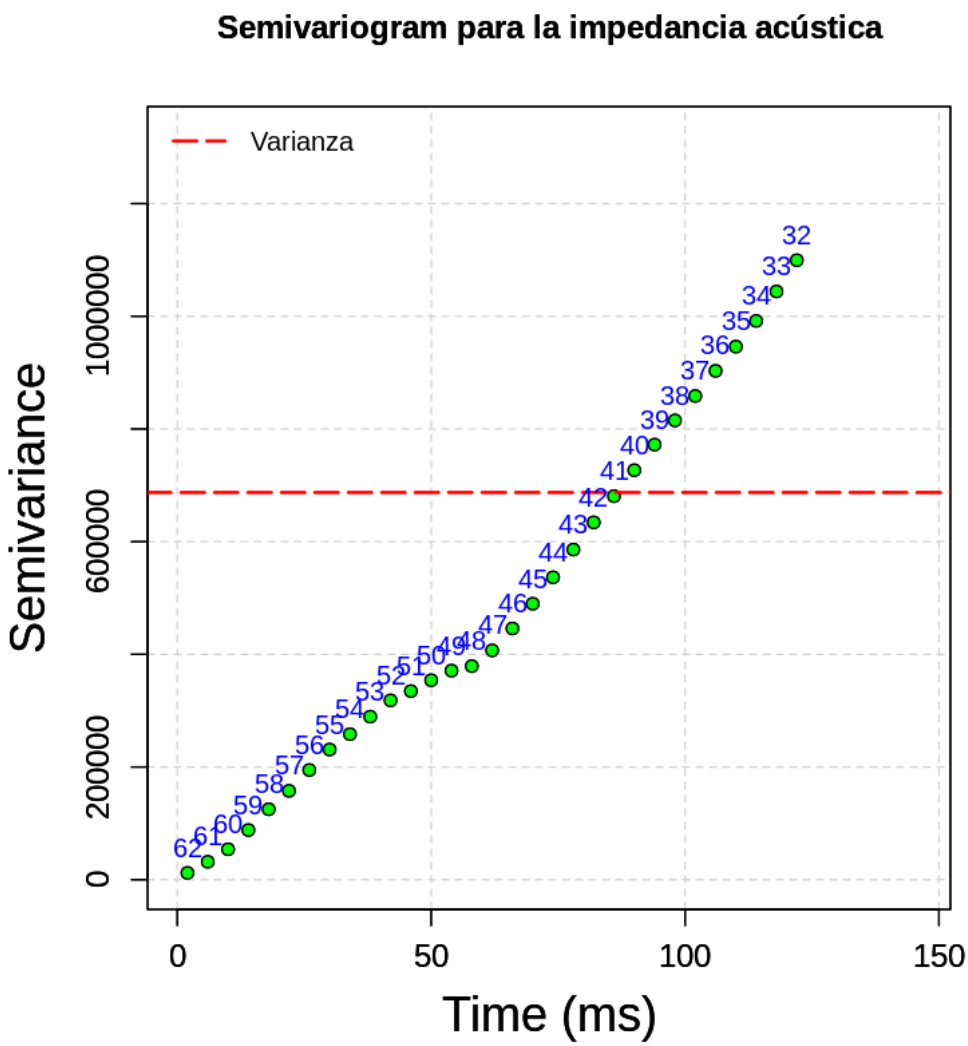
7.3.2 Variographyc analysis of acoustic impedance

As with total porosity, the variogram model must also be estimated for acoustic impedance in the time domain.

[51]:
```
var_ob=Ip_like_SUB
YCoord=numeric(time_like_SUB_Stat[1,2])
#YCoord_Stat<-Estadisticas(YCoord)

CoorY=YCoord
CoorX=time_like_SUB
```

[52]:
```
Direccion=90
Tol=0
NIntervalos=31
Lags=4
```

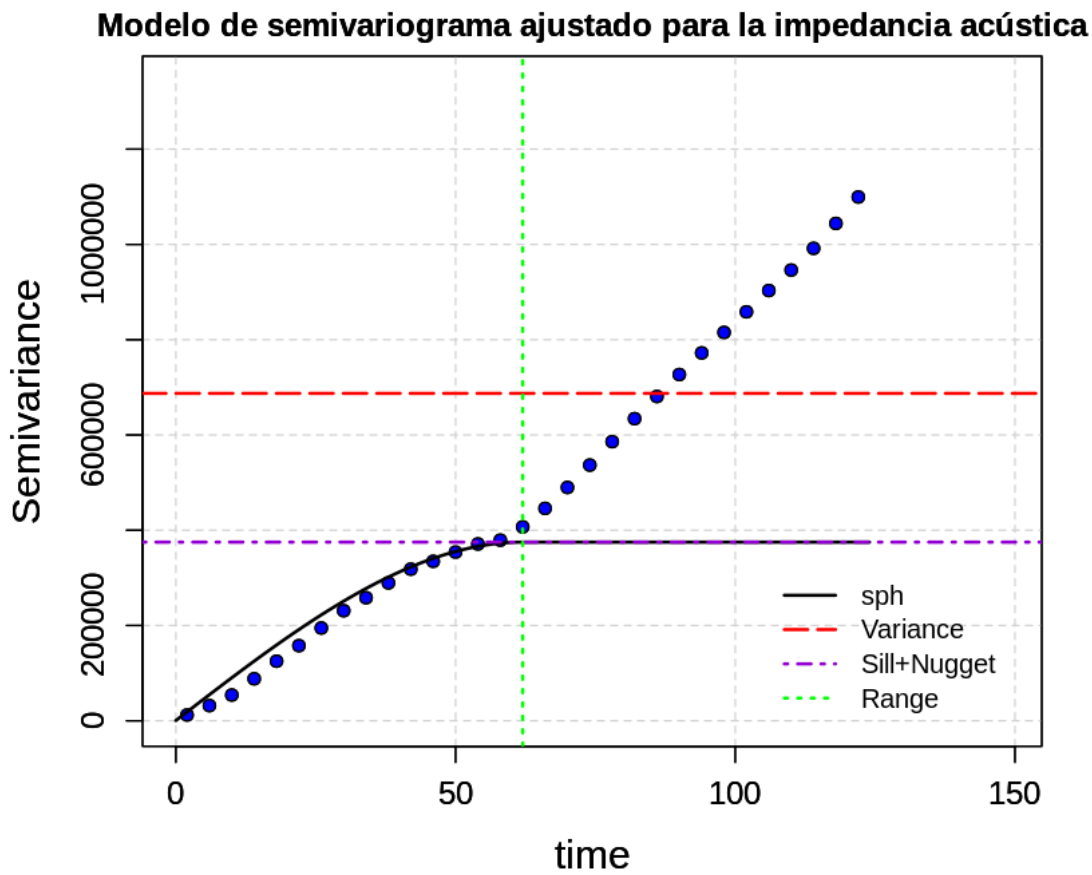
[53]:
```
Ip_SUB_VarioEstimation<-Variograma(CoorX=time_like_SUB, CoorY=YCoord,␣
↪Variable=var_ob,
                                   Direccion=90, Tol=0, NIntervalos=NIntervalos,
                                   Lags=Lags,graph=TRUE,
                                   MainTitle="Semivariogram para la impedancia␣
↪acústica",
                                   xlab="Time (ms)",
                                   ylab = "Semivariance")
```



[54]:
```
var_tr2<-Trend(CoorX, CoorY, var_ob, degree=1)
```

[55]:
```
Modelo=2
Nugget=500
SillYNugget=375000
Alcance=62
```

[56]:
```
EyeModelVarioFit_Ip<-EyeModel(CoorX, CoorY, var_ob,
                              Direccion=Direccion, Tol=Tol,␣
 ↪NIntervalos=NIntervalos,
                              Lags=Lags, 1,
                              Modelo=Modelo, Nugget, SillYNugget, Alcance,␣
 ↪graph=TRUE,
                              MainTitle="Modelo de semivariograma ajustado para␣
 ↪la impedancia acústica",
                              xlab="time")
```



Model	Nugget	Sill+Nugget	Range	MSE
sph	500.0000	375000.0000	62.0000	2925023074482.

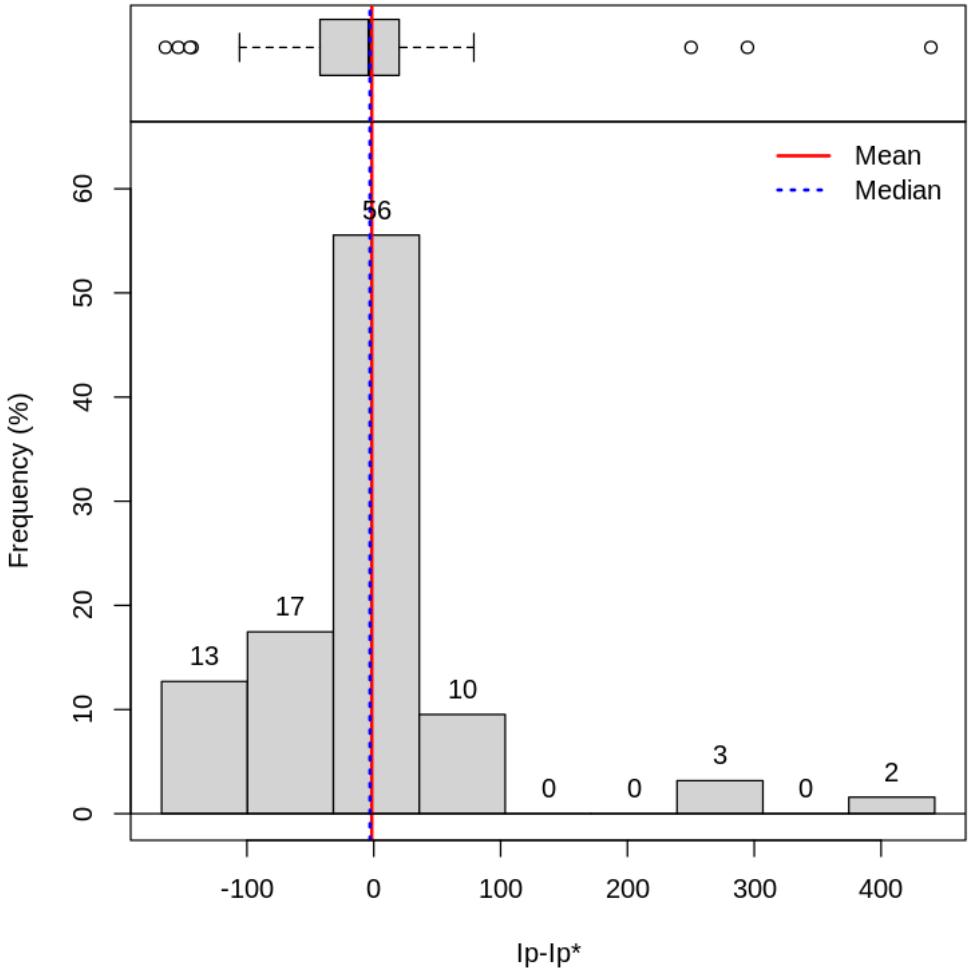
[57]:
```
Ip_CrossValid<- CrossValidation(CoorX, CoorY, var_ob,
                                Modelo, Nugget, SillYNugget, Alcance,␣
↪MaxAnis=0, proporcion=1)
```

[58]:
```
Ip_CrossValid_Stat<- Val_Estadisticos(Ip_CrossValid[1:
↪length(var_tr2[,3]),c(3,4,5)])
Ip_CrossValid_Stat
```

A data.frame: 13 × 3

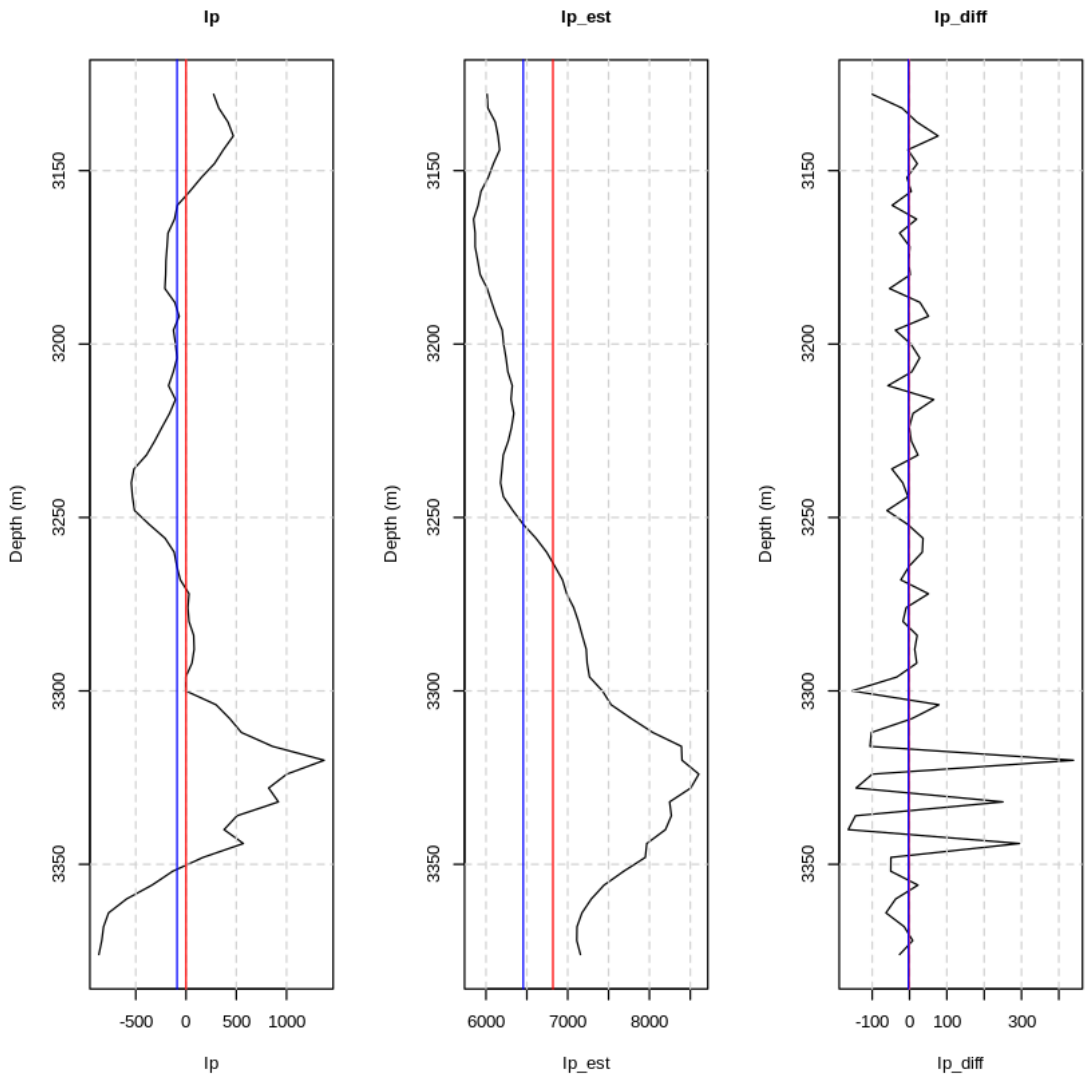
	Z <dbl>	Z* <dbl>	Z-Z* <dbl>
No_muestras	63.00000	63.00000	63.00000
Minimo	5840.12408	5847.88899	-164.27029
Cuartil_1er	6159.76602	6157.48678	-42.38749
Mediana	6450.82284	6455.85420	-3.00451
Media	6817.80232	6819.37196	-1.56964
Cuartil_3er	7254.24012	7277.71490	20.00870
Maximo	8838.18502	8604.37974	439.26289
Rango	2998.06095	2756.49075	603.53318
Rango_Intercuartil	1094.47410	1120.22812	62.39619
Varianza	687258.82606	672889.77012	8641.90687
Desv_Estandar	829.01075	820.29859	92.96186
Simetria	0.68479	0.64073	2.18025
Curtosis	-0.60264	-0.78486	9.25594

[59]:
```
HistBoxplot(x=Ip_CrossValid[,5], mean = Ip_CrossValid_Stat[5,3],
            median = Ip_CrossValid_Stat[4,3], main ="",
            xlab = "Ip-Ip*", ylab = "Frequency (%)",
            AbsFreq = FALSE, PercentFreq = TRUE )
```

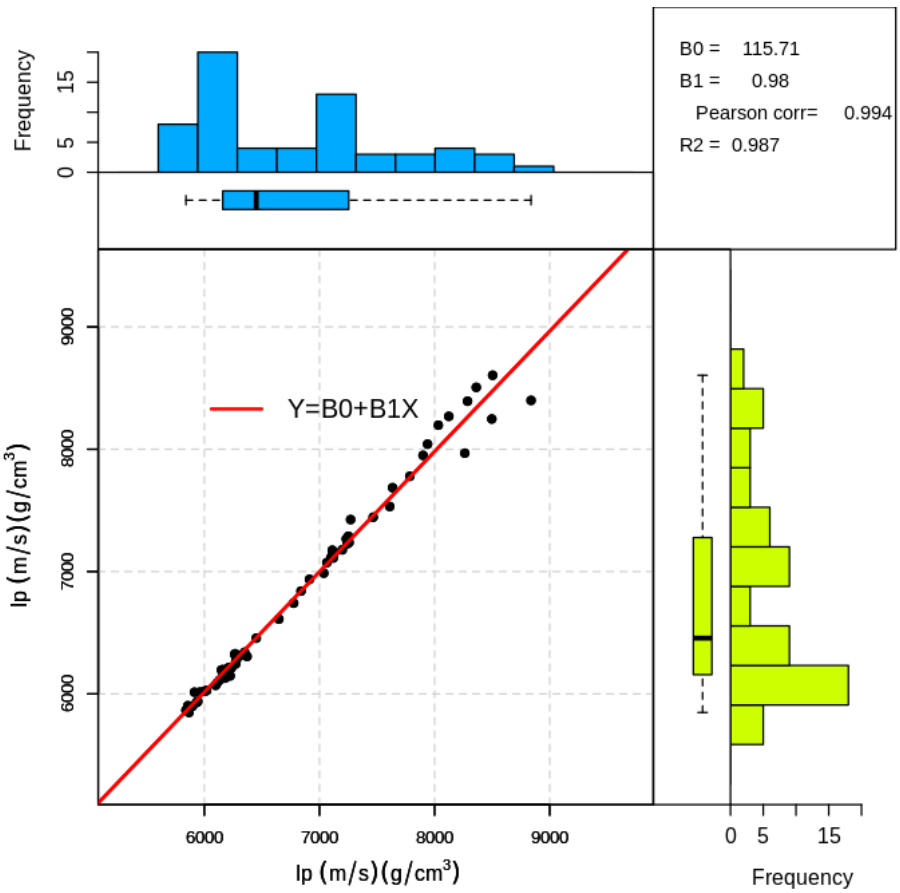


[60]:
```
Ip_est<-Ip_CrossValid[,4]
Ip_est_Stat<-Estadisticas(Ip_est)
Ip_diff<-Ip_CrossValid[,5]
Ip_diff_Stat<-Estadisticas(Ip_diff)
```

[61]:
```
LogPlot(depth=time_like_SUB, n_logs=3,
        log_list=list(var_tr2[,3],Ip_est,Ip_diff),
        log_name_list=list("Ip", "Ip_est","Ip_diff"),
        log_xlabel_list=list("Ip", "Ip_est","Ip_diff"),
        plot_mean=TRUE, plot_median=TRUE)
```



[62]:
```
scaterplotReg(Ip_like_SUB , Ip_est, ns,
              Xmin = Ip_like_SUB_Stat[2,2], Xmax = Ip_like_SUB_Stat[7,2],
              Ymin = Ip_est_Stat[2,2],Ymax = Ip_est_Stat[7,2],
              XLAB = expression(paste(" Ip ",(m/s)(g/cm^3))), YLAB =␣
 ↪expression(paste(" Ip ",(m/s)(g/cm^3)),"*"))
```



Finally, variogram models were estimated for the subsampled acoustic impedance and total porosity. Three models were tested: exponential, Gaussian, and spherical. The best-fit model for acoustic impedance was spherical, with a sill of 375,000, a range of 62 milliseconds, and a nugget of 500 . For total porosity, the spherical model was also selected, with a sill of 0.0004, a range of 50 milliseconds, and a nugget of zero.

8 Method Validation: Geostatistical Seismic Inversion Based on Copulas

8.1 Forward Model: Bayesian Estimation of the Dependence Model

This section applies the Bayesian inference framework described in Section~6.3 to estimate the joint distribution model that characterizes the dependence between acoustic impedance and total

porosity. The posterior parameters used in this section are obtained through the two-stage updating process, where well-log-based priors are first refined using upscaled logs and then validated with subsampled data. These posterior estimates form the core of the forward model used in the seismic inversion.

8.1.1 Prior information

Once the parameters of the different variogram models have been established, and the statistical descriptors of the various sources of information are known, the next step is to evaluate the changes in the dependence model as it is trained with the different sources of information obtained from scaling and subsampling. Therefore, the first step is to obtain the parameter values to construct the parametric dependence model using the well log data. This model will be updated with the information from the scaled and subsampled logs, so that the posterior dependence model updated with the subsampled logs becomes the model used for the simulations.

[63]:
```
fga <- fitdist(Phit, "gamma", method="mle")
summary(fga)
```

```
Fitting of the distribution ' gamma ' by maximum likelihood
Parameters :
       estimate Std. Error
shape  32.73952  0.7577396
rate  147.49648  3.4399633
Loglikelihood:  6804.604   AIC:  -13605.21   BIC:  -13592.78
Correlation matrix:
          shape      rate
shape 1.0000000 0.9923743
rate  0.9923743 1.0000000
```

[64]:
```
fb <- fitdist(Phit, "beta", method="mle")
summary(fb)
```

```
Fitting of the distribution ' beta ' by maximum likelihood
Parameters :
       estimate Std. Error
shape1 27.21242  0.6292995
shape2 95.50037  2.2230800
Loglikelihood:  6924.448   AIC:  -13844.9   BIC:  -13832.47
Correlation matrix:
          shape1    shape2
shape1 1.0000000 0.9882518
shape2 0.9882518 1.0000000
```

[65]:
```
fex <- fitdist(Phit, "exp", method="mle")
summary(fex)
```

```
Fitting of the distribution ' exp ' by maximum likelihood
```

```
Parameters :
     estimate Std. Error
rate 4.505154 0.07410434
Loglikelihood:  1867.301   AIC:  -3732.602   BIC:  -3726.387
```

[66]:
```r
fcb <- fitdist(Phit, "norm", method="mle")
summary(fcb)
```

```
Fitting of the distribution ' norm ' by maximum likelihood
Parameters :
       estimate   Std. Error
mean 0.22196799 0.0005552771
sd   0.03375792 0.0003910909
Loglikelihood:  7279.647   AIC:  -14555.29   BIC:  -14542.86
Correlation matrix:
     mean sd
mean    1  0
sd      0  1
```

[67]:
```r
fgu <- fitdist(Phit, "logis", method="mle")
summary(fgu)
```

```
Fitting of the distribution ' logis ' by maximum likelihood
Parameters :
          estimate   Std. Error
location 0.2268969 0.0004431505
scale    0.0161029 0.0002281026
Loglikelihood:  7660.835   AIC:  -15317.67   BIC:  -15305.24
Correlation matrix:
            location       scale
location  1.00000000 -0.08556198
scale    -0.08556198  1.00000000
```

[68]:
```r
fln <- fitdist(Phit, "lnorm", method="mle")
summary(fln)
```

```
Fitting of the distribution ' lnorm ' by maximum likelihood
Parameters :
          estimate  Std. Error
meanlog -1.5205644 0.003118662
sdlog    0.1895982 0.002204951
Loglikelihood:  6521.496   AIC:  -13038.99   BIC:  -13026.56
Correlation matrix:
        meanlog sdlog
meanlog       1     0
sdlog         0     1
```

[69]:
```r
fw <- fitdist(Phit, "weibull", method="mle")
summary(fw)
```

```
Fitting of the distribution ' weibull ' by maximum likelihood
Parameters :
       estimate   Std. Error
shape 9.3737893 0.1248476215
scale 0.2343424 0.0004270601
Loglikelihood:  7783.975   AIC:  -15563.95   BIC:  -15551.52
Correlation matrix:
         shape    scale
shape 1.000000 0.271627
scale 0.271627 1.000000
```

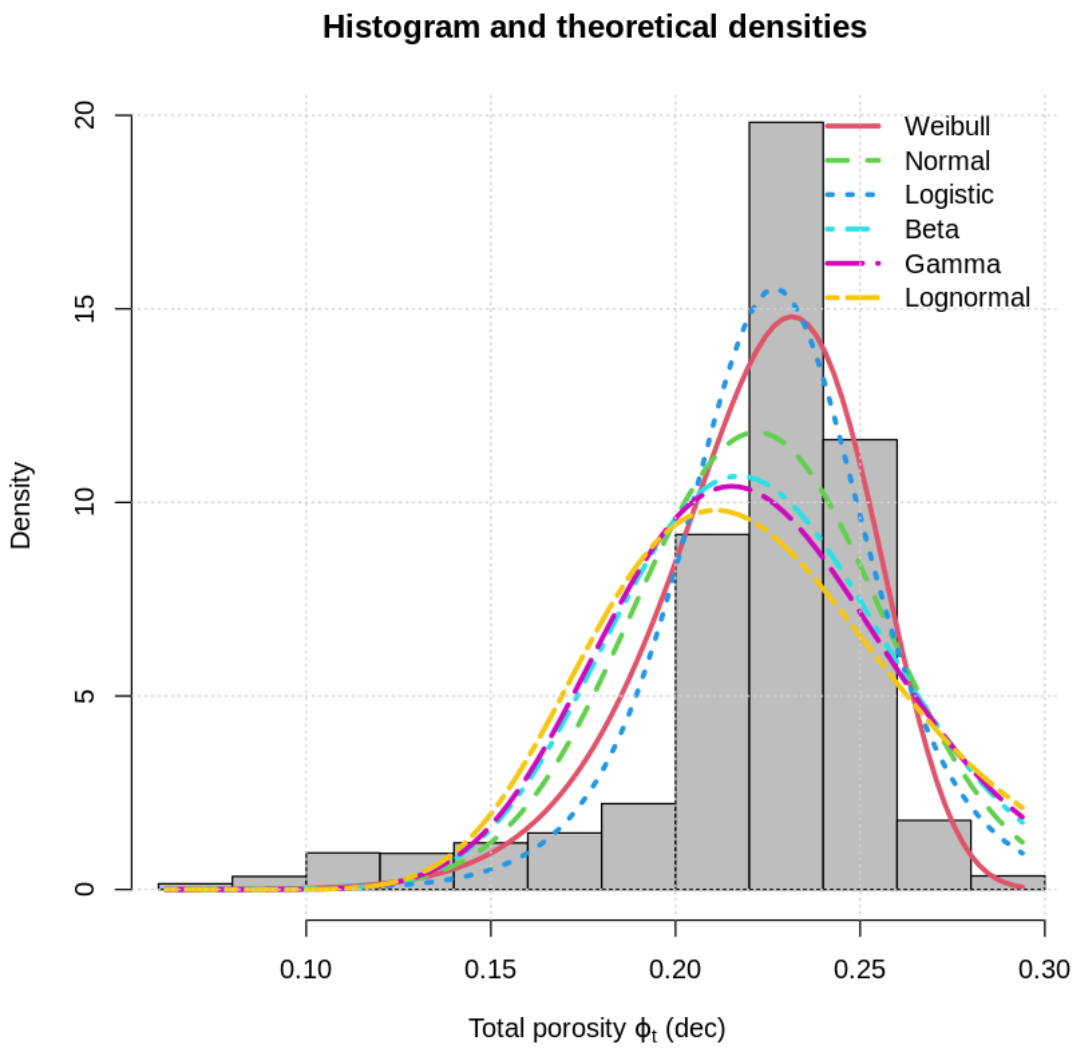
The goodness of fit results for ϕ_t are in next table shows the best option is Weibull function, followed by the normal function.

Function	Parameter 1	Error 1	Parameter 2	Error 2	Likelihood	AIC
Weibull	$\alpha = 9.3737893$	0.1248476215	$\lambda = 0.2343424$	0.0004270601	7783.975	-15563.95
Logistics	$\mu = 0.2268969$	0.0004431505	$\beta = 0.0161029$	0.0002281026	7660.835	-15317.67
Normal	$\mu = 0.22196799$	0.0005552771	$\sigma = 0.03375792$	0.0003910909	7279.647	-14555.29
Beta	$\alpha = 27.21242$	0.6292995	$\beta = 95.50037$	2.2230800	6924.448	-13844.90
Gamma	$k = 32.73952$	0.7577396	$\beta = 147.49648$	3.4399633	6804.604	-13605.21
Lognormal	$\log \mu = -1.5205644$	0.003118662	$\log \sigma = 0.1895982$	0.002204951	6521.496	-13038.99

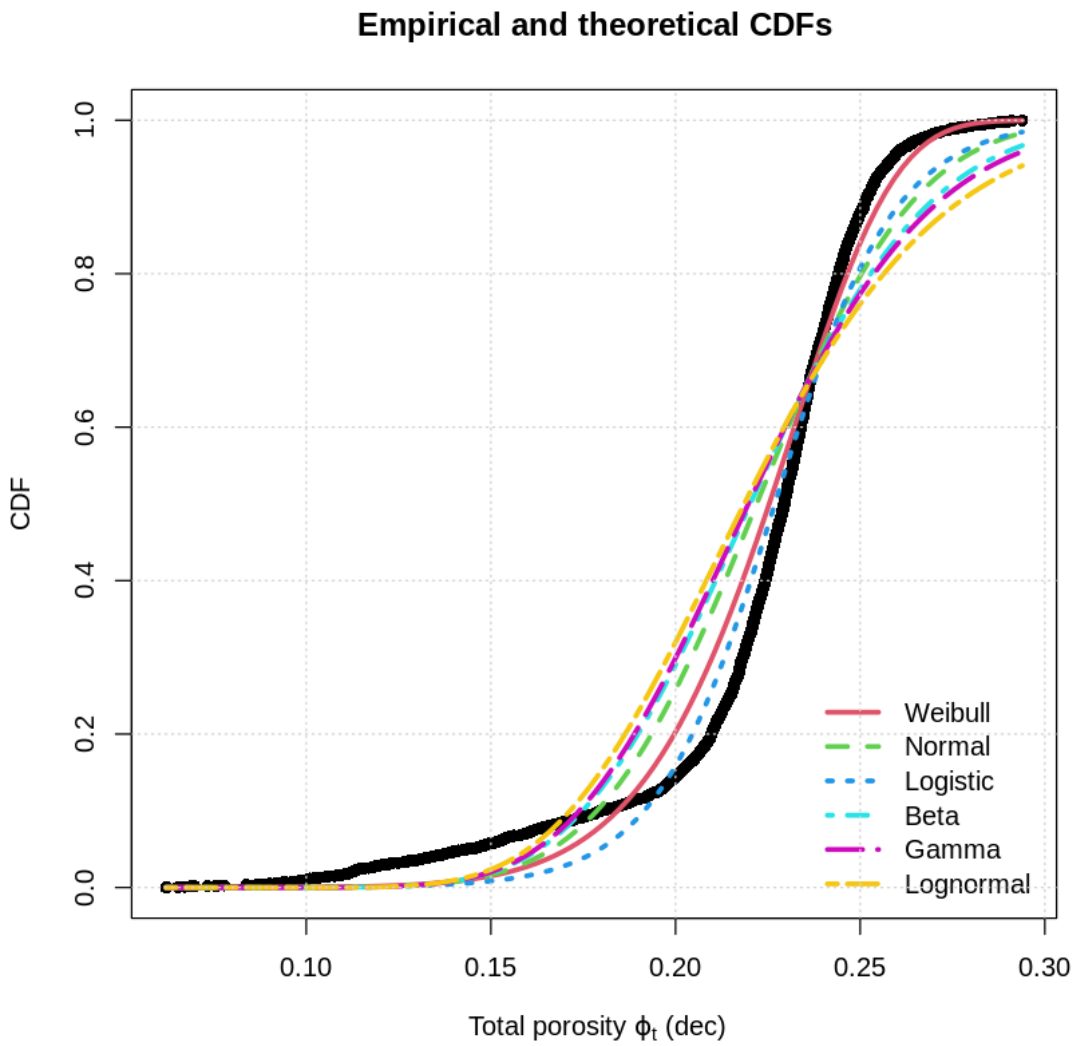
[70]:
```r
best_fit_Phit <- gofstat(list(fw,fgu,fcb,fb,fga,fln))
print("Goodness of fit results for density Phit")
best_fit_Phit
```

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

Goodness-of-fit statistics
                             1-mle-weibull 2-mle-logis 3-mle-norm  4-mle-beta
Kolmogorov-Smirnov statistic     0.1097527  0.07930123   0.167921   0.1934954
Cramer-von Mises statistic      12.8058933  9.32103949  33.443180  45.7591406
Anderson-Darling statistic      82.4283358 91.08782300 190.187892 252.2601466
                             5-mle-gamma 6-mle-lnorm
Kolmogorov-Smirnov statistic   0.2004381   0.2157027
Cramer-von Mises statistic    49.6588353  58.5242451
Anderson-Darling statistic   272.1908931 316.6594461

Goodness-of-fit criteria
                               1-mle-weibull 2-mle-logis 3-mle-norm 4-mle-beta
Akaike's Information Criterion     -15563.95   -15317.67  -14555.29  -13844.90
Bayesian Information Criterion     -15551.52   -15305.24  -14542.86  -13832.47
                               5-mle-gamma 6-mle-lnorm
Akaike's Information Criterion   -13605.21   -13038.99
Bayesian Information Criterion   -13592.78   -13026.56
```

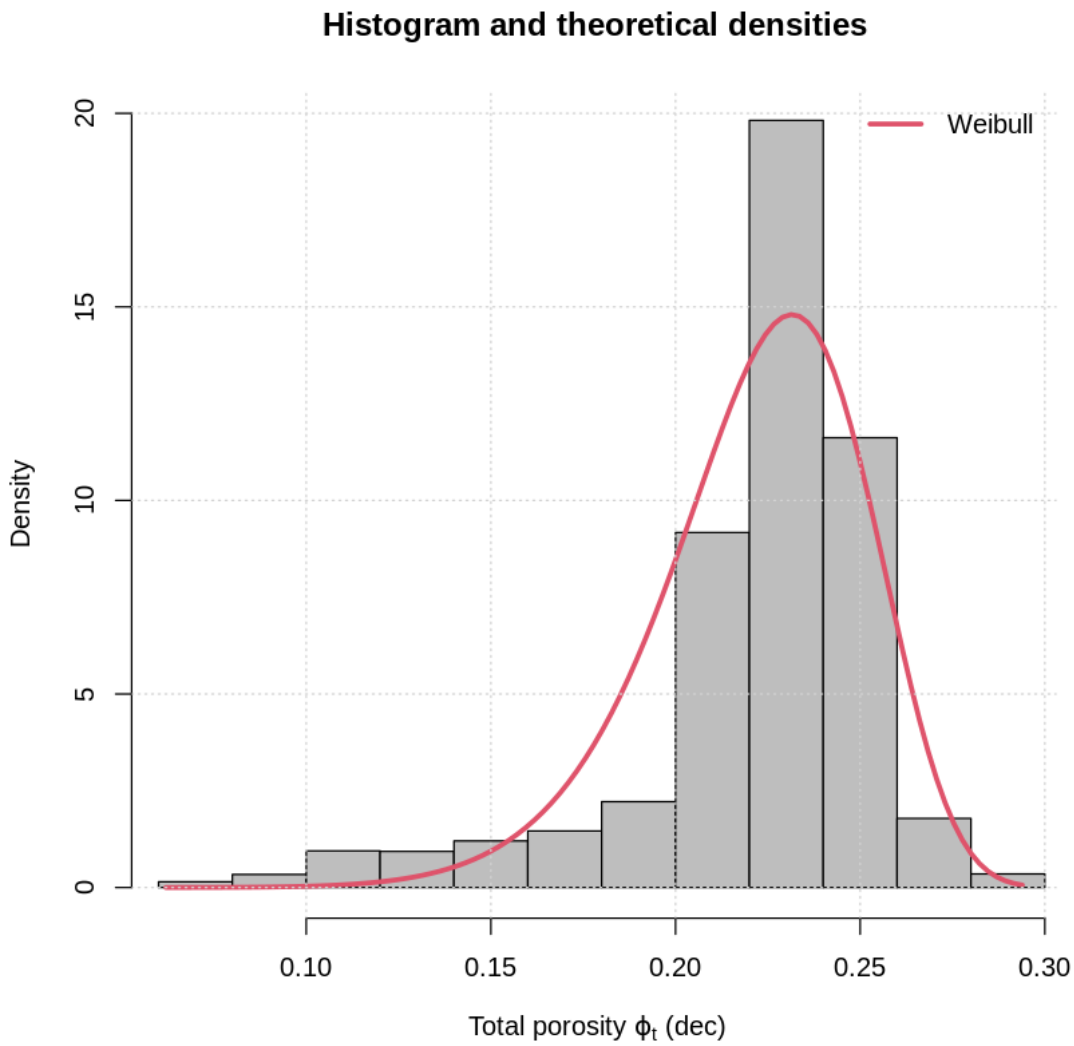
[71]:
```
plot.legend <- c("Weibull","Normal","Logistic","Beta","Gamma","Lognormal")
denscomp(list(fw,fcb,fgu,fb,fga,fln),fitlwd=3,datacol = "gray",
         xlab = expression(paste(" Total porosity ",varphi[t], " (dec)")),
         legendtext = plot.legend)
grid()
```



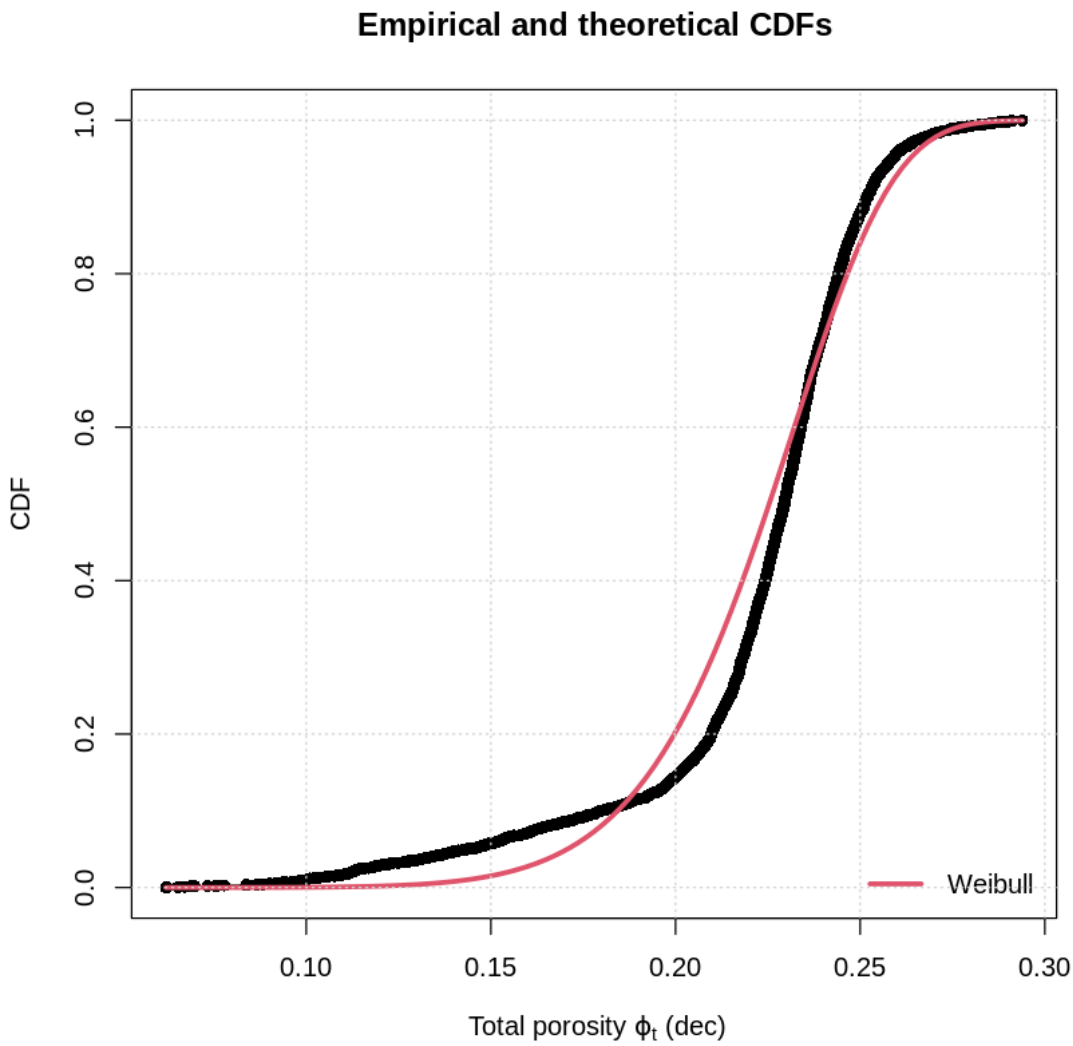
[72]:
```
cdfcomp(list(fw,fcb,fgu,fb,fga,fln),fitlwd=3,
        xlab = expression(paste(" Total porosity ",varphi[t], " (dec)")), ␣
 ↪legendtext = plot.legend)
grid()
```



[73]:
```
denscomp(list(fw),xlab = expression(paste(" Total porosity ",varphi[t], "␣
↪(dec)")),
         datacol = "gray",fitlwd=3 , legendtext = "Weibull")
grid()
```



[74]:
```
cdfcomp(list(fw),xlab = expression(paste(" Total porosity ",varphi[t], "␣
↪(dec)")),
        fitlwd=3,  legendtext = "Weibull")
grid()
```



8.2 Acoustic impedance CDF modeling

This estimation uses the following functions: gamma, normal, logistic, lognormal and Weibull.

[75]:
```
fga1 <- fitdist(Ip, "gamma", method="mle")
summary(fga1)
```

```
Fitting of the distribution ' gamma ' by maximum likelihood
Parameters :
          estimate   Std. Error
shape 42.009249278 4.748327e-01
rate   0.006028288 6.659491e-05
Loglikelihood:  -31014.55   AIC:  62033.11   BIC:  62045.54
Correlation matrix:
```

```
          shape      rate
shape 1.0000000 0.9747748
rate  0.9747748 1.0000000
```

[76]:
```r
fcb1 <- fitdist(Ip, "norm", method="mle")
summary(fcb1)
```

```
Fitting of the distribution ' norm ' by maximum likelihood
Parameters :
     estimate Std. Error
mean 6968.284   18.82691
sd   1144.530   13.31156
Loglikelihood:  -31274.4   AIC:  62552.8   BIC:  62565.23
Correlation matrix:
     mean sd
mean    1  0
sd      0  1
```

[77]:
```r
fgu1 <- fitdist(Ip, "logis", method="mle")
summary(fgu1)
```

```
Fitting of the distribution ' logis ' by maximum likelihood
Parameters :
          estimate Std. Error
location 6819.0032  16.178643
scale     578.6063   8.123053
Loglikelihood:  -31021.28   AIC:  62046.56   BIC:  62058.99
Correlation matrix:
           location      scale
location 1.00000000 0.07276249
scale    0.07276249 1.00000000
```

[78]:
```r
fln1 <- fitdist(Ip, "lnorm", method="mle")
summary(fln1)
```

```
Fitting of the distribution ' lnorm ' by maximum likelihood
Parameters :
         estimate  Std. Error
meanlog 8.8371689 0.002475473
sdlog   0.1504958 0.001750076
Loglikelihood:  -30907.01   AIC:  61818.03   BIC:  61830.46
Correlation matrix:
        meanlog sdlog
meanlog       1     0
sdlog         0     1
```

[79]:
```
fw1 <- fitdist(Ip, "weibull", method="mle")
summary(fw1)
```

```
Fitting of the distribution ' weibull ' by maximum likelihood
Parameters :
         estimate  Std. Error
shape    5.459444  0.05996752
scale 7475.132200 24.00575403
Loglikelihood:  -31742.46   AIC:  63488.92   BIC:  63501.35
Correlation matrix:
          shape     scale
shape 1.0000000 0.3457439
scale 0.3457439 1.0000000
```

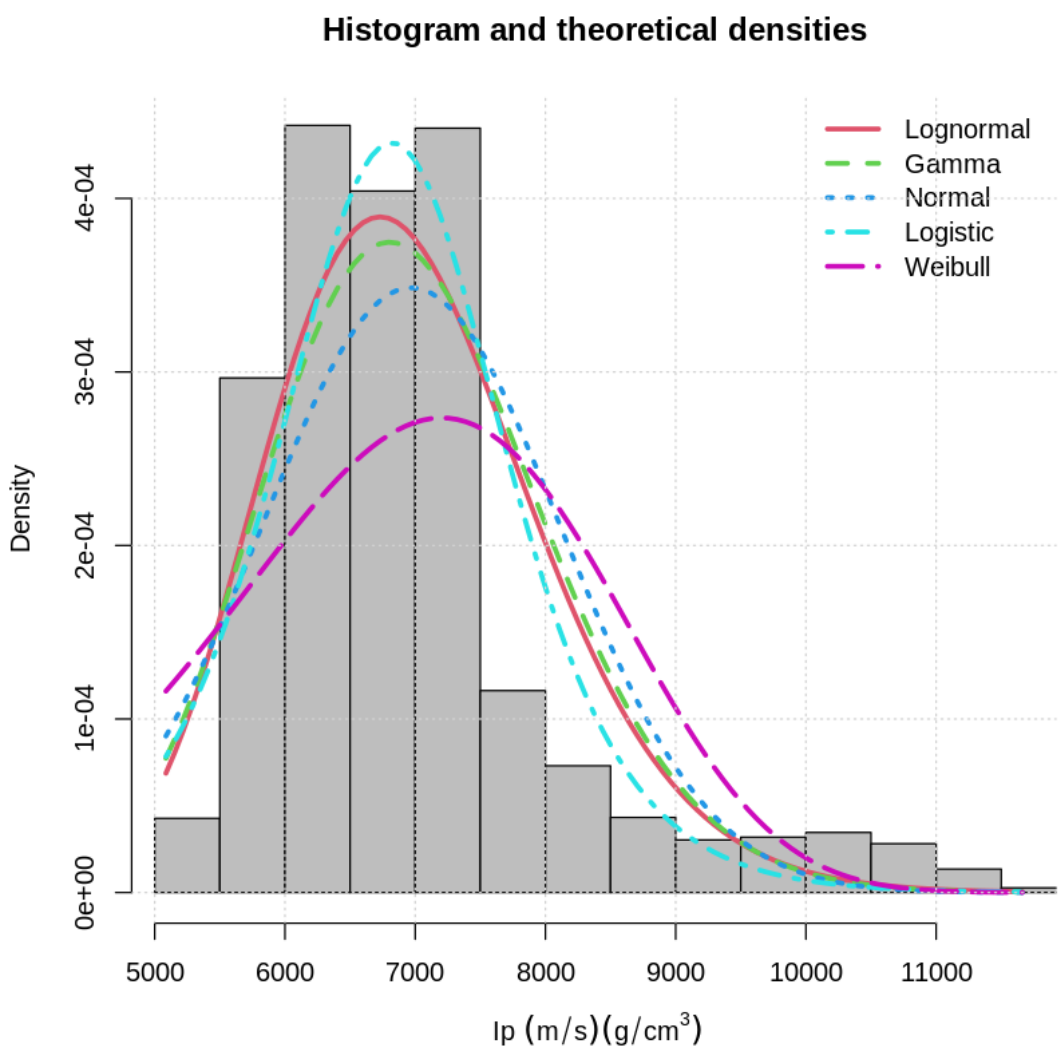
[80]:
```
best_fit_Vp <- gofstat(list(fln1,fga1,fgu1,fcb1,fw1))
print("Goodness of fit results for Ip")
best_fit_Vp
```

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

Goodness-of-fit statistics
                             1-mle-lnorm 2-mle-gamma 3-mle-logis  4-mle-norm
Kolmogorov-Smirnov statistic    0.100625   0.1127581  0.07269712   0.1371893
Cramer-von Mises statistic      9.307338  12.3790239  4.49666196  19.8380707
Anderson-Darling statistic     66.703203  85.2457585 61.39664016 128.9324776
                             5-mle-weibull
Kolmogorov-Smirnov statistic     0.1760765
Cramer-von Mises statistic      36.7735975
Anderson-Darling statistic     221.6392028

Goodness-of-fit criteria
                               1-mle-lnorm 2-mle-gamma 3-mle-logis 4-mle-norm
Akaike's Information Criterion    61818.03    62033.11    62046.56   62552.80
Bayesian Information Criterion    61830.46    62045.54    62058.99   62565.23
                               5-mle-weibull
Akaike's Information Criterion      63488.92
Bayesian Information Criterion      63501.35
```

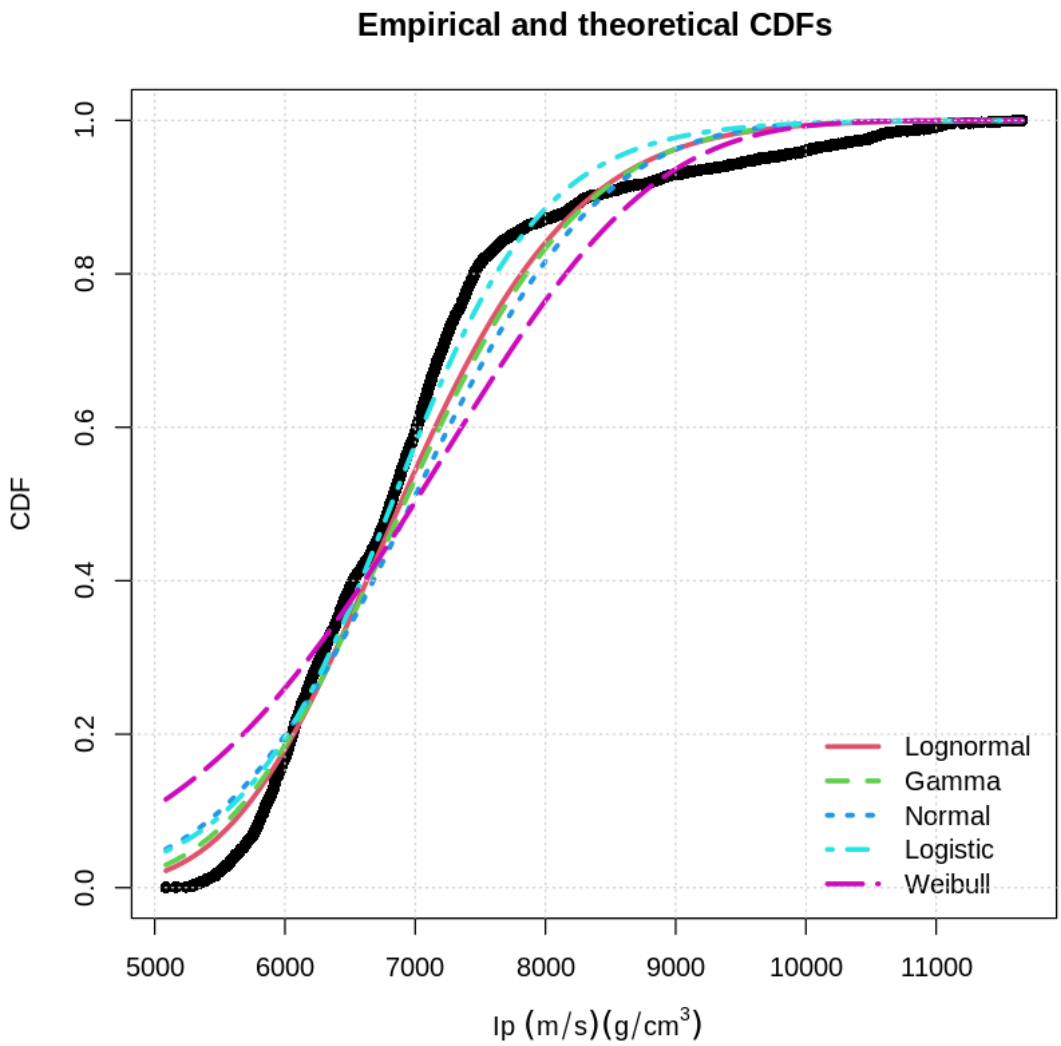
The results goodness of fit results for Ip is in next table (Table 3), the best function for Ip is lognormal, gamma function is very close to lognormal and maybe that function could be a good option, however, the lognormal function is taken.

Function	Parameter 1	Error 1	Parameter 2	Error 2	Likelihood	AIC	B
Lognormal	$\log \mu = 8.8371689$	0.002475473	$\log \sigma = 0.1504958$	0.001750076	-30907.01	61818.03	618
Gamma	$k = 42.0092$	0.4748327	$\beta = 0.0060283$	0.000066595	-31014.55	62033.11	620
Logistic	$\mu = 6819.0032$	16.178643	$\beta = 578.6063$	8.123053	-31021.28	62046.56	620
Normal	$\mu = 6968.284$	18.82691	$\sigma = 1144.530$	13.31156	-31274.40	62552.80	625
Weibull	$\alpha = 5.459444$	0.05996752	$\lambda = 7475.1322$	24.00575403	-31742.46	63488.92	635

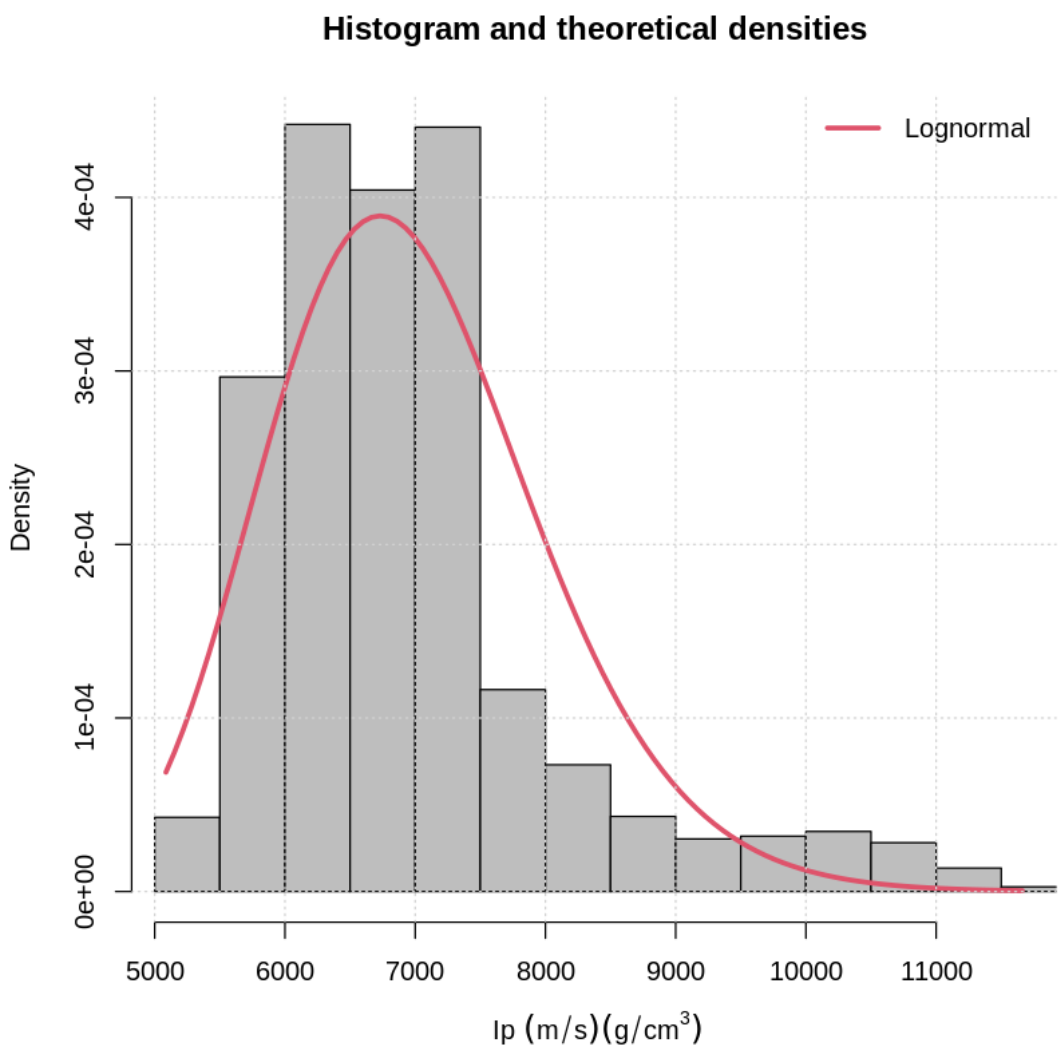
[81]:
```
plot.legend <- c("Lognormal","Gamma", "Normal", "Logistic",    "Weibull")
denscomp(list(fln1,fga1,fcb1,fgu1,fw1),datacol = "gray",
         xlab = expression(paste(" Ip ",(m/s)(g/cm^3))),fitlwd=3 , legendtext =␣
 ↪plot.legend)
grid()
```



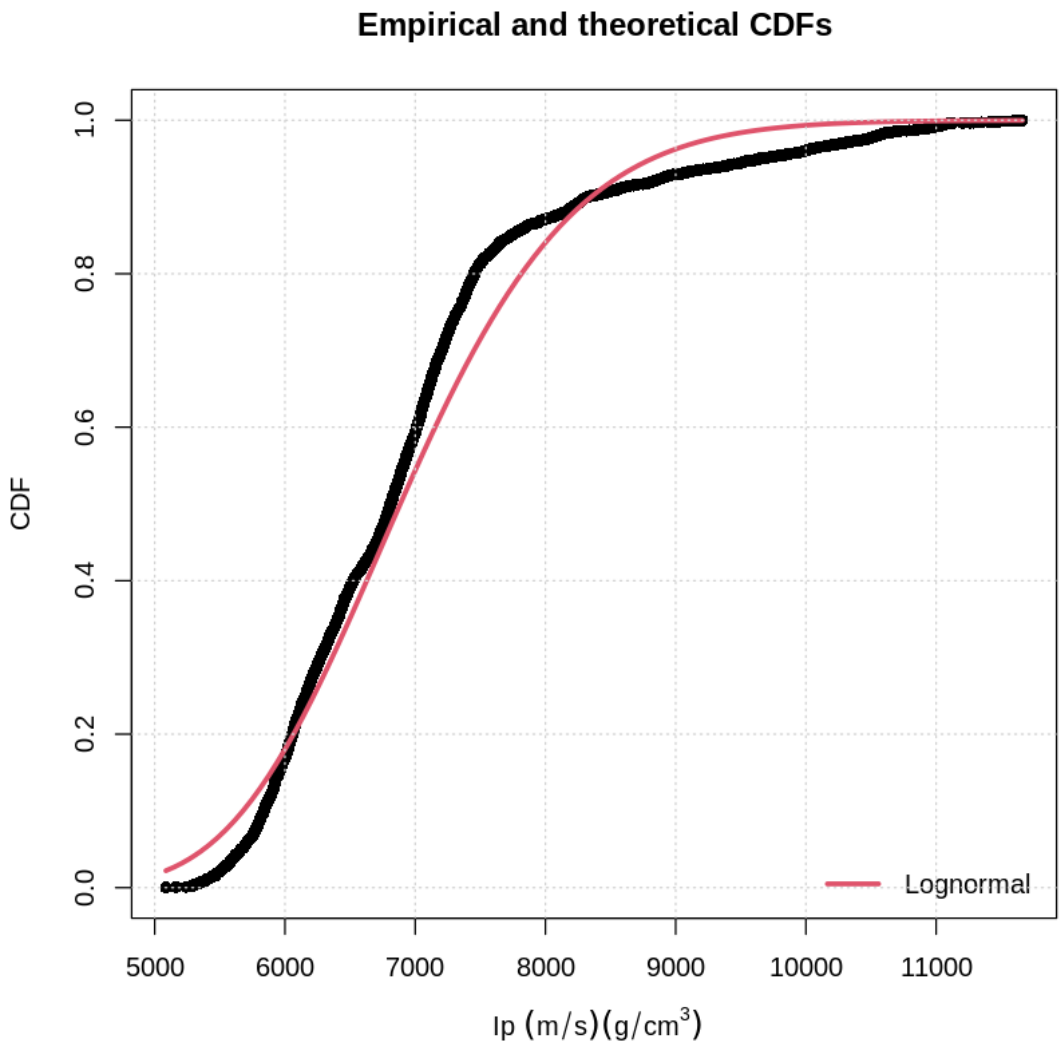
[82]:
```
cdfcomp(list(fln1,fga1,fcb1,fgu1,fw1),xlab = expression(paste(" Ip ",(m/s)(g/
 ↪cm^3))),
        fitlwd=3,  legendtext = plot.legend)
grid()
```



[83]:
```
plot.legend <- c("Lognormal")
denscomp(list(fln1),datacol = "gray",xlab = expression(paste(" Ip ",(m/s)(g/
↪cm^3))),
         fitlwd=3 , legendtext = plot.legend)
grid()
```



[84]:
```
cdfcomp(list(fln1),xlab = expression(paste(" Ip ",(m/s)(g/cm^3))),fitlwd=3, ␣
↪legendtext = plot.legend)
grid()
```



8.3 Copula parametric modeling

After selecting the variable best functions, is estimate the best parametric copula. Considering the dependence model has negative dependencies, is necessary omit the Gumbel-Hougaard copula because it's not valid on negative dependencies, Clayton copula can be calculated in negative dependences only if the estimation method is semi-parametric, that's the reason of use the Kendall inversion τ (ITau) and Spearman inversion ρ (IRho).

[85]:
```
muestra<-as.data.frame(cbind(Ip, Phit))
data_pseudo=pobs(muestra)
n=3696
```

[86]:
```
ifme_frank <- fitCopula(frankCopula(), data = data_pseudo, method = "ml")
summary(ifme_frank)
```

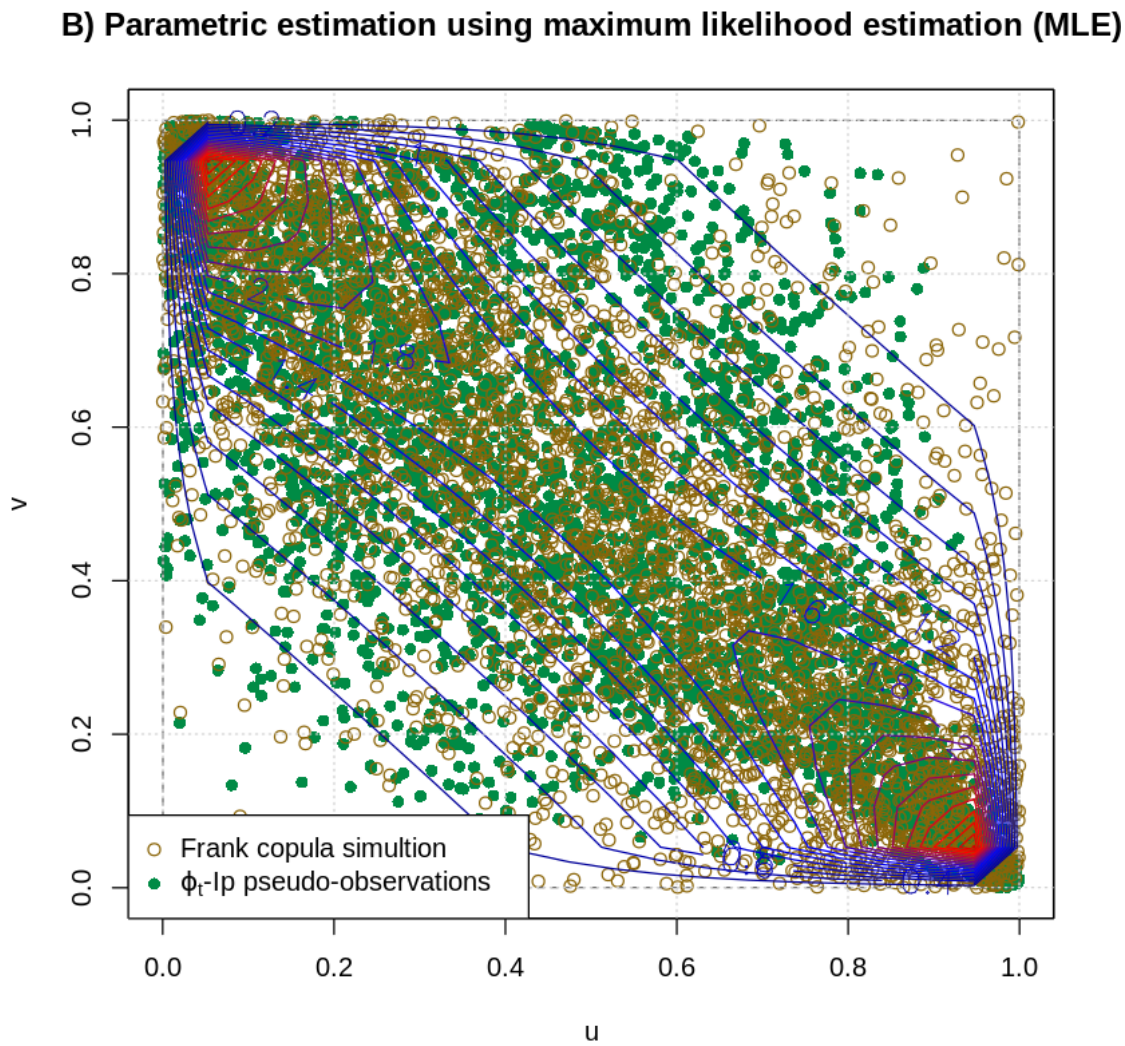
```
Call: fitCopula(copula, data = data, method = "ml")
Fit based on "maximum likelihood" and 3696 2-dimensional observations.
Frank copula, dim. d = 2
      Estimate Std. Error
alpha   -6.265      0.132
The maximized loglikelihood is 1312
Optimization converged
Number of loglikelihood evaluations:
function gradient
       5        5
```

[87]:
```
fc <- frankCopula(ifme_frank@copula@parameters)
```

[88]:
```
mfc <- mvdc(fc, margins = c("lnorm", "weibull"),
            paramMargins = list(list(meanlog = 8.8371689, sdlog = 0.1504958),
                                list(shape = 9.3737893, scale = 0.2343424)))
```

[89]:
```
Xf <- rMvdc(n, mvdc = mfc)
Xfpobs=pobs(Xf)
```

[90]:
```
cols = rev(colorRampPalette(c('darkred','red','blue','darkblue'))(25))

#png("Results/Frank_MLE.png", bg = "white",  width=2000, height=2000, res=250)
plot(data_pseudo, pch=16, col=("springgreen4"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Xfpobs, xlab=(""), ylab=(""), main = ("B) Parametric estimation using␣
 ↪maximum likelihood estimation (MLE)"),
     col=("darkgoldenrod4"))
par(new=TRUE)
contour(fc, dCopula,n=20,nlevels=25, xlab = "u",
     ylab = "v" ,labcex=1.4, col=cols,xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "bottomleft", legend = c("Frank copula simultion",
                                    expression(paste(varphi[t],"-Ip␣
 ↪pseudo-observations"))),
                                    col = c("darkgoldenrod4", "springgreen4"),
                                    pch=c(1,16))
#dev.off()
```



[91]:
```
itauclay<-fitCopula(claytonCopula(), data = data_pseudo, method = "itau")
summary(itauclay)
```

```
Call: fitCopula(copula, data = data, method = "itau")
Fit based on "inversion of Kendall's tau" and 3696 2-dimensional observations.
Clayton copula, dim. d = 2
      Estimate Std. Error
alpha  -0.6958      0.008
```

[92]:
```
itauFrank<-fitCopula(frankCopula(), data = data_pseudo, method = "itau")
summary(itauFrank)
```

```
Call: fitCopula(copula, data = data, method = "itau")
Fit based on "inversion of Kendall's tau" and 3696 2-dimensional observations.
```

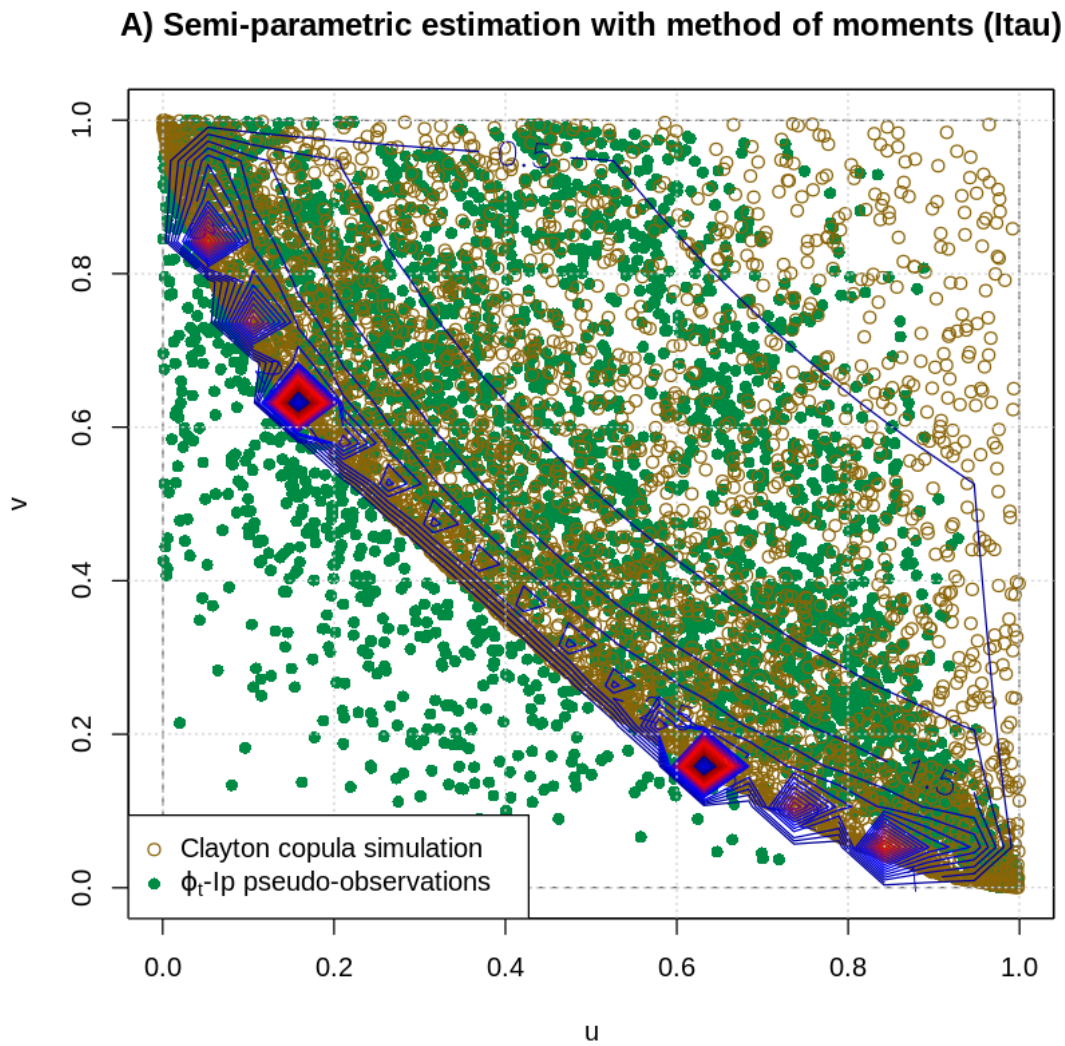
```
Frank copula, dim. d = 2
      Estimate Std. Error
alpha   -6.382      0.011
```

[93]:
```
ccC <- claytonCopula(itauclay@copula@parameters)
ccFr <- frankCopula(itauFrank@copula@parameters)
```

[94]:
```
mccC <- mvdc(ccC, margins = c("lnorm", "weibull"),
             paramMargins = list(list(meanlog = 8.8371689, sdlog = 0.1504958),
                                 list(shape = 9.3737893, scale = 0.2343424)))
```

[95]:
```
XC <- rMvdc(n, mvdc = mccC)
XpobsC=pobs(XC)
```

[96]:
```
plot(data_pseudo, pch=16, col=("springgreen4"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(XpobsC, xlab = 'u',
     ylab = 'v',
     main = ("A) Semi-parametric estimation with method of moments (Itau)"),␣
 ↪col=("darkgoldenrod4"))
par(new=TRUE)
contour(ccC, dCopula,n=20,nlevels=25, xlab=(""), ylab=("") ,labcex=1.4,␣
 ↪col=cols,xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "bottomleft", legend = c("Clayton copula simulation",
                                    expression(paste(varphi[t],"-Ip␣
 ↪pseudo-observations"))),
                                    col = c("darkgoldenrod4", "springgreen4"),
                                    pch=c(1,16))
```

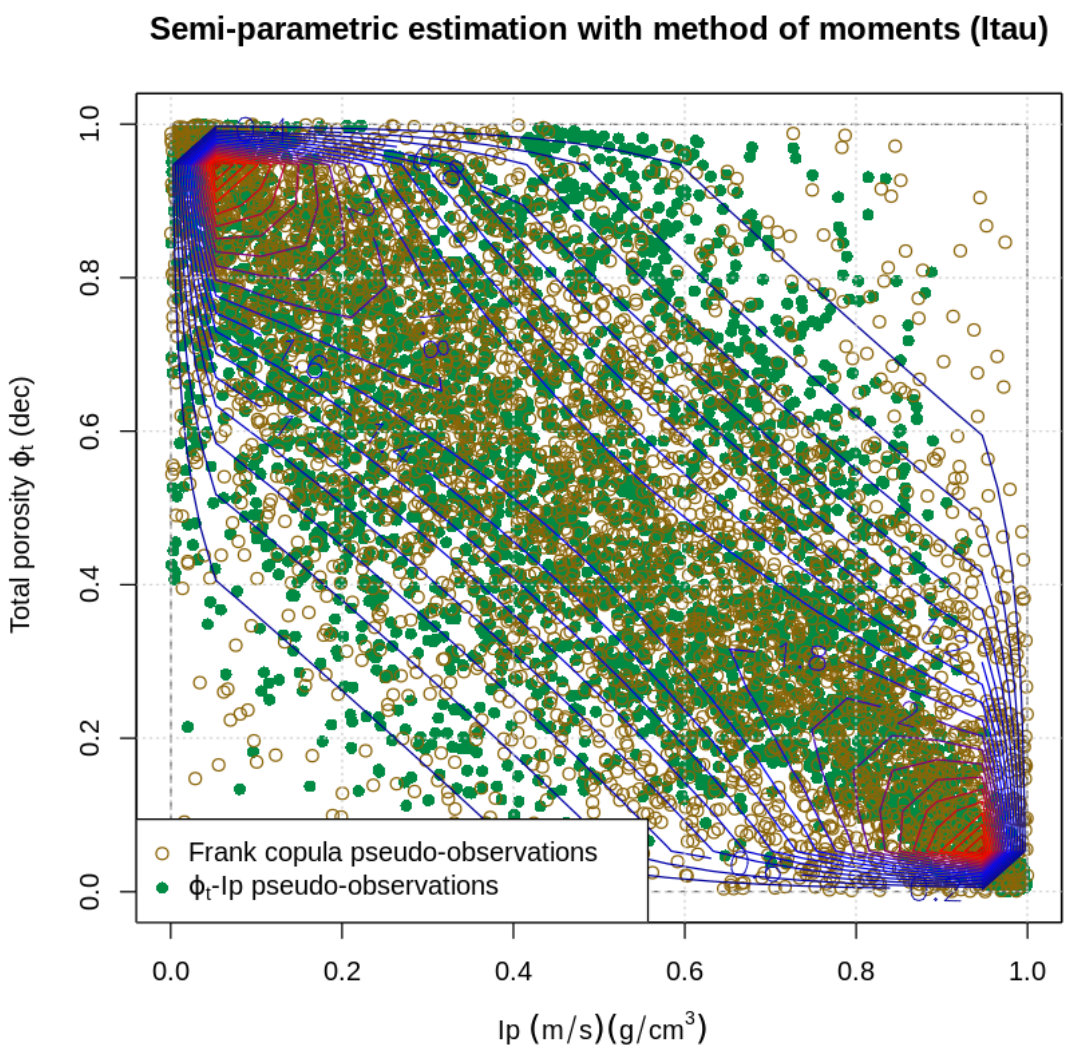


[97]:
```
mccFr <- mvdc(ccFr, margins = c("lnorm", "weibull"),
              paramMargins = list(list(meanlog = 8.8371689, sdlog = 0.1504958),
                                  list(shape = 9.3737893, scale = 0.2343424)))
```

[98]:
```
XFr <- rMvdc(n, mvdc = mccFr)
XpobsFr=pobs(XFr)
```

[99]:
```
plot(data_pseudo, pch=16, col=("springgreen4"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(XpobsFr, xlab = expression(paste(" Ip ",(m/s)(g/cm^3))),
     ylab = expression(paste(" Total porosity ",varphi[t], " (dec)")),
     main = ("Semi-parametric estimation with method of moments (Itau)"),␣
 ↪col=("darkgoldenrod4"))
```

```
par(new=TRUE)
contour(ccFr, dCopula,n=20,nlevels=25, xlab=(""), ylab=(""),labcex=1.4, col=cols␣
↪,xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "bottomleft", legend = c("Frank copula pseudo-observations",
                                    expression(paste(varphi[t],"-Ip␣
↪pseudo-observations"))),
                                    col = c("darkgoldenrod4", "springgreen4"),
                                    pch=c(1,16))
```



[100]:
```
irhoClay<-fitCopula(claytonCopula(), data = data_pseudo, method = "irho")
summary(irhoClay)
```

```
Call: fitCopula(copula, data = data, method = "irho")
Fit based on "inversion of Spearman's rho" and 3696 2-dimensional observations.
Clayton copula, dim. d = 2
      Estimate Std. Error
alpha  -0.7243      0.009
```

[101]:
```
irhoFrank<-fitCopula(frankCopula(), data = data_pseudo, method = "irho")
summary(irhoFrank)
```

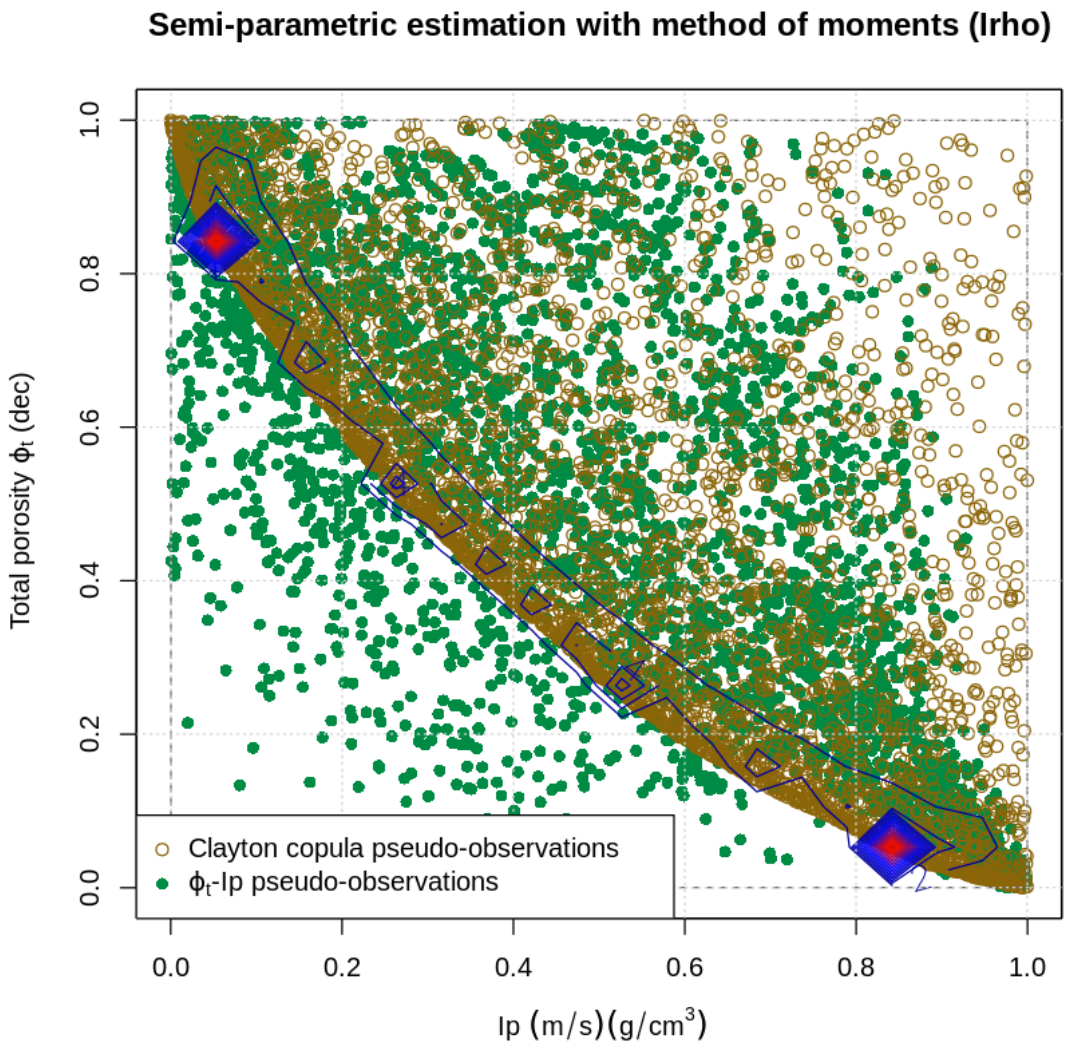
```
Call: fitCopula(copula, data = data, method = "irho")
Fit based on "inversion of Spearman's rho" and 3696 2-dimensional observations.
Frank copula, dim. d = 2
      Estimate Std. Error
alpha   -5.998      0.165
```

[102]:
```
crhoC <- claytonCopula(irhoClay@copula@parameters)
crhoFr <- frankCopula(irhoFrank@copula@parameters)
```

[103]:
```
mcrhoC <- mvdc(crhoC, margins = c("lnorm", "weibull"),
            paramMargins = list(list(meanlog = 8.8371689, sdlog = 0.1504958),
                                list(shape = 9.3737893, scale = 0.2343424)))
```

[104]:
```
XrhoC <- rMvdc(n, mvdc = mcrhoC)
XpobsrhoC=pobs(XrhoC)
```

[105]:
```
plot(data_pseudo, pch=16, col=("springgreen4"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(XpobsrhoC, xlab = expression(paste(" Ip ",(m/s)(g/cm^3))),
     ylab = expression(paste(" Total porosity ",varphi[t], " (dec)")),
     main = ("Semi-parametric estimation with method of moments (Irho)"),␣
 ↪col=("darkgoldenrod4"))
par(new=TRUE)
contour(crhoC, dCopula,n=20,nlevels=25, xlab=(""), ylab=("") ,labcex=1.4,␣
 ↪col=cols,xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "bottomleft", legend = c("Clayton copula pseudo-observations",
                                    expression(paste(varphi[t],"-Ip␣
 ↪pseudo-observations"))),
                                    col = c("darkgoldenrod4", "springgreen4"),
                                    pch=c(1,16))
```

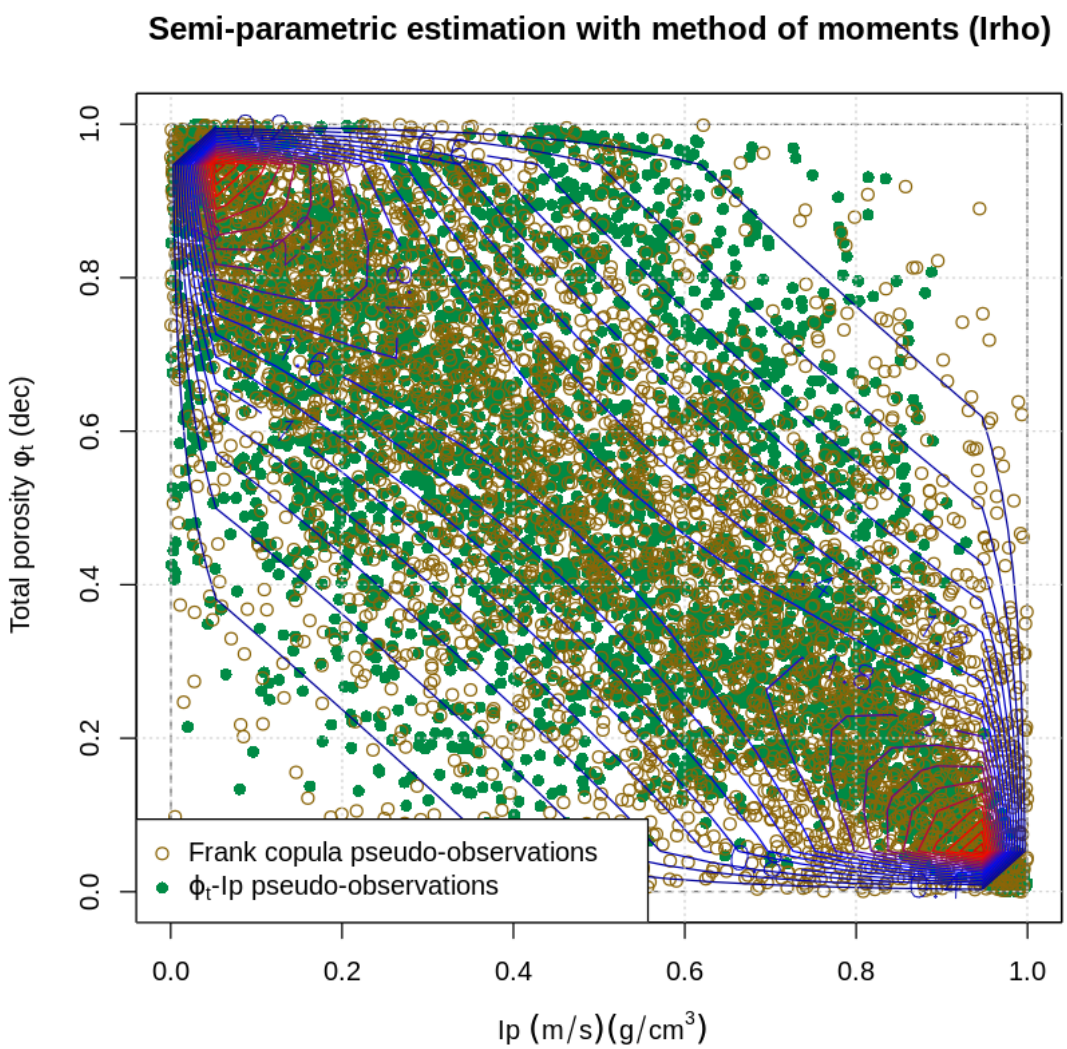


[106]:
```
mcrhoFr <- mvdc(crhoFr, margins = c("lnorm", "weibull"),
            paramMargins = list(list(meanlog = 8.8371689, sdlog = 0.1504958),
                                list(shape = 9.3737893, scale = 0.2343424)))
```

[107]:
```
XrhoFr <- rMvdc(n, mvdc = mcrhoFr)
XpobsrhoFr=pobs(XrhoFr)
```

[108]:
```
plot(data_pseudo, pch=16, col=("springgreen4"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(XpobsrhoFr, xlab = expression(paste(" Ip ",(m/s)(g/cm^3))),
     ylab = expression(paste(" Total porosity ",phi[t], " (dec)")),
     main = ("Semi-parametric estimation with method of moments (Irho)"),␣
 ↪col=("darkgoldenrod4"))
```

```
par(new=TRUE)
contour(crhoFr, dCopula,n=20,nlevels=25, xlab=(""), ylab=("") ,labcex=1.4,␣
 ↪col=cols,xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "bottomleft", legend = c("Frank copula pseudo-observations",
                                    expression(paste(varphi[t],"-Ip␣
 ↪pseudo-observations"))),
                                    col = c("darkgoldenrod4", "springgreen4"),
                                    pch=c(1,16))
```



As can see in the following table, the best option is to use Clayton's probability distribution function. However, using this function could omit the values that are outside its coverage area, therefore we will use Frank's copula with the parameter estimated by the Kendall inversion method.

Also, the difference in the value of their respective errors is very low.

Copula	Method	Parameter	Error
Clayton	Itau	$\theta = - - 0.6958$	0.008
Clayton	Irho	$\theta = -0.7243$	0.009
Frank	Itau	$\theta = -6.382$	0.011
Frank	MLE	$\theta = -6.265$	0.132
Frank	Irho	$\theta = -5.998$	0.165

Therefore, the best settings are as follows, for the variable ϕ_t:

Function	**Parameter 1**	**Error 1**	**Parameter 2**	**Error 2**	**Likelihood**	**AIC**	**BIC**
Weibull	$\alpha = 9.3737893$	0.1248476215	$\lambda = 0.2343424$	0.0004270601	7783.975	-15563.95	-15551.52

For the variable Ip.

Function	**Parameter 1**	**Error 1**	**Parameter 2**	**Error 2**	**Likelihood**	**AIC**	**BIC**
Lognormal	$\log \mu = 8.8371689$	0.002475473	$\log \sigma = 0.1504958$	0.001750076	-30907.01	61818.03	—

The best copula is:

Copula	Method	Parameter	Error
Frank	MLE	$\theta = -6.265$	0.132

The results of the parametric analysis are presented in the work by Vázquez-Ramírez et al. (2023). That study demonstrated that the functions most compatible with geophysical well logs are, in order of suitability: the lognormal distribution is optimal for acoustic impedance, the Weibull distribution is optimal for total porosity, and the Frank copula model is optimal for the dependence structure. Thus, the prior model consists of five parameters: the lognormal parameters $(\log \mu, \log \sigma)$ and its normalized marginal u representing acoustic impedance; the Weibull parameters (α, λ) and its normalized marginal v representing total porosity; and the Frank copula dependence parameter (θ). The prior function is expressed as:

$$\mathbb{P}(\mathbf{m}) = \pi(\log \mu, \log \sigma, \alpha, \lambda, \theta) \tag{6}$$

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 evaluated with acoustic impedance samples (x_1), f_2 is the Weibull density function evaluated with total porosity samples (x_2), and $c_\alpha(u,v)$ is the Frank copula density function (Eq.~**??**). The posterior distribution $\mathbb{P}(\mathbf{m}|\mathbf{d})$ is obtained by combining Eqs.~6 and~**??**.

8.4 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

[109]:
```
Data_like<-cbind(Ip_like,Phit_like)
```

[110]:
```
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
```

[111]:
```
v_tot_post_5<-diag(c(var_theta_post,var_mulog_post,var_sigmalog_post,var_alpha_post,var_lambda_
```

[112]:
```
start_5=array(c(-15,8.8,0.15,9,0.2),c(1,5))
start_5
```

A matrix: 1 × 5 of type dbl -15 8.8 0.15 9 0.2

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

[3]:
```
logpost_5 <- 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=-(6.265), sd=((-(6.265)/100)*1.5), log=TRUE)
  mulog_prior=dnorm(mulog, mean=8.8371689, sd=((8.8371689/100)*1.8), log=TRUE)
  sigmalog_prior=dnorm(sigmalog, mean=0.1504958, sd=((0.1504958/100)*1.8),␣
↪log=TRUE)
  alpha_prior=dnorm(alpha, mean=9.3737893, sd=((9.3737893/100)*1.8), log=TRUE)
  lambda_prior=dnorm(lambda, mean=0.2343424, sd=((0.2343424/100)*1.8), 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)

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

}
```

[31]:
```
suppressWarnings({mcmc_5.fit <-rwmetrop(logpost_5,
                    list(var=v_tot_post_5,scale=2),
                    start_5,
                    10000,
                    Data_like)})
```

[32]:
```
mcmc_5.fit$accept
```

0.2352

[33]:
```
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])
```

[34]:
```
nb = nclass.Sturges(mcmc_5.fit$par[,1])
nb
```

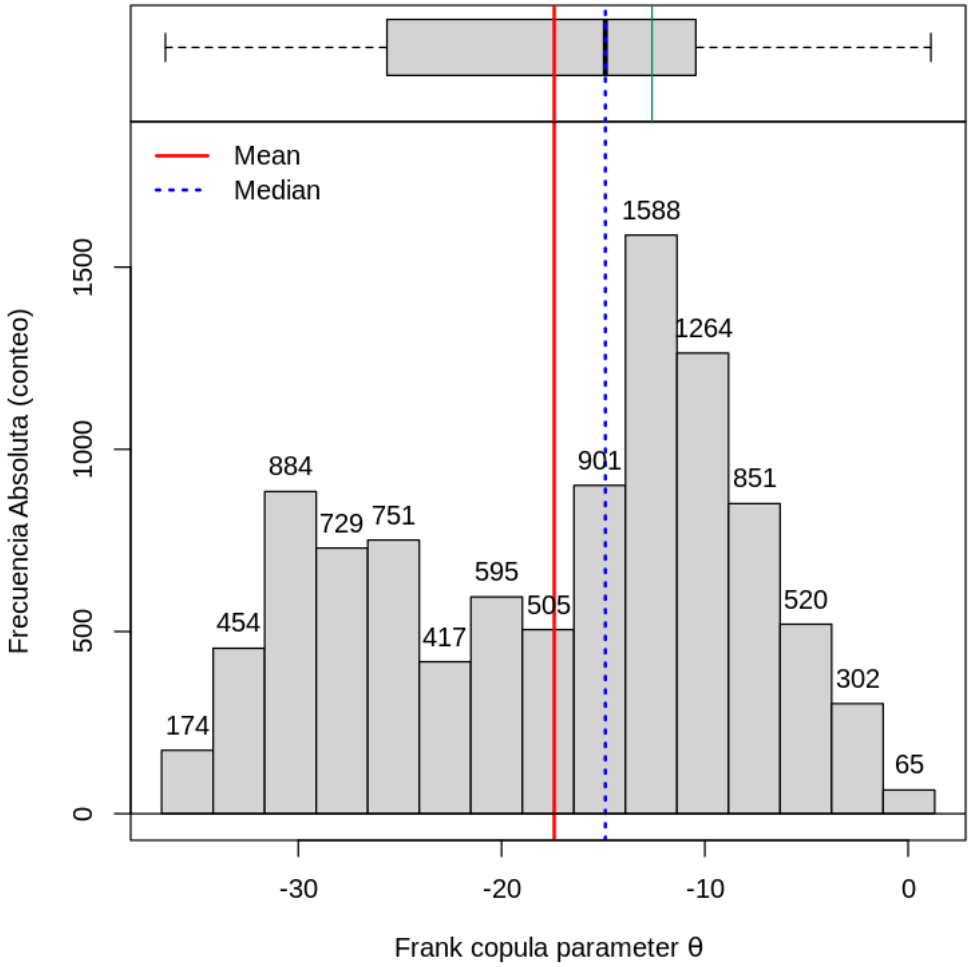
15

[35]:
```
theta_best_post5=mfv(round(mcmc_5.fit$par[,1],1))
HistBoxplot(mcmc_5.fit$par[,1],nbins = nb, 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))
```

A data.frame: 14 × 2

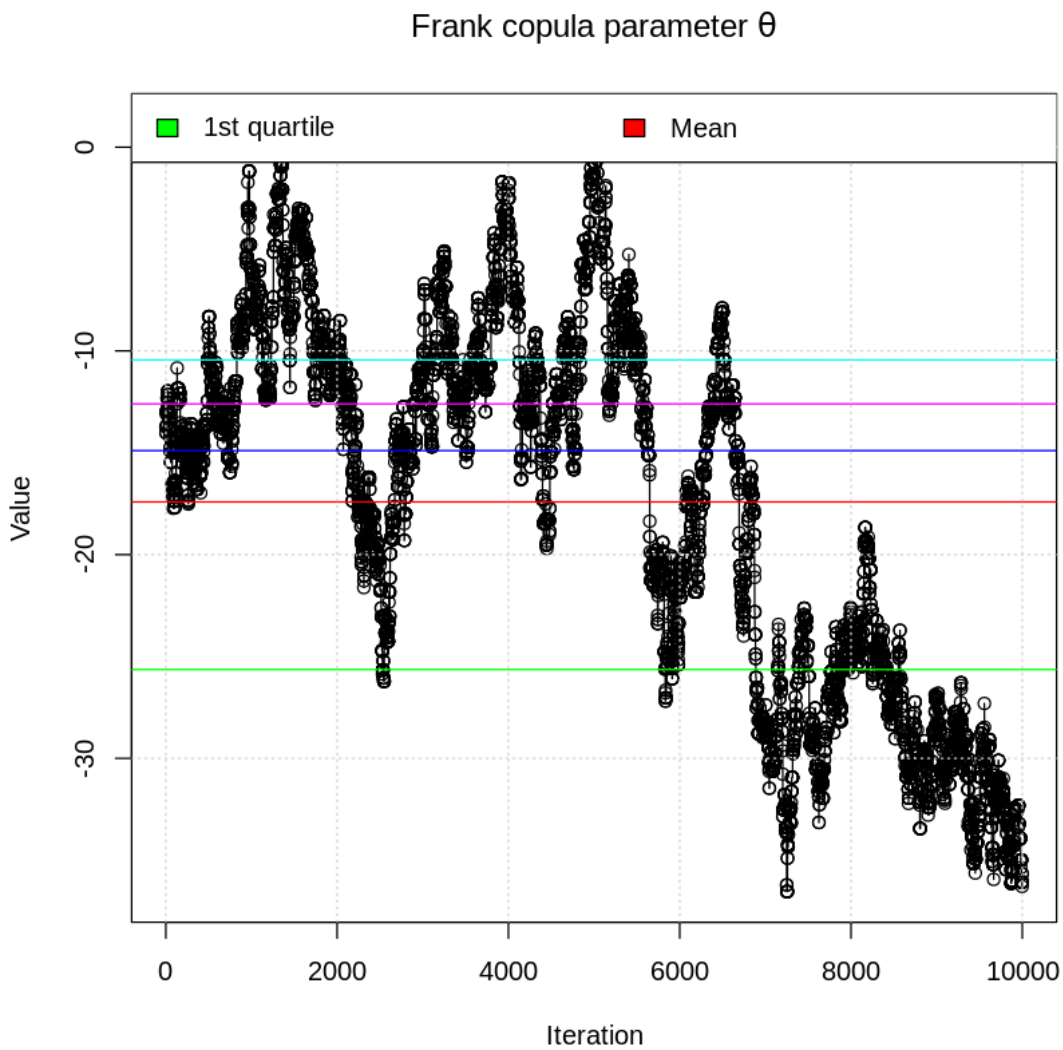
	Statistics <chr>	Values <dbl>
muestras	n	10000.0000
minimos	Minimum	-36.5374
cuantiles1	1st. Quartile	-25.6391
medianas	Median	-14.8952
medias	Mean	-17.4113
cuantiles3	3rd. Quartile	-10.4431
maximos	Maximum	1.1208
rangos	Rank	37.6581
rangosInt	Interquartile Rank	15.1960
varianzas	Variance	81.0383
desvs	Standard Deviation	9.0021
CVs	Variation Coeff.	-0.5170
simetrias	Skewness	-0.2896
curtosiss	Kurtosis	-1.0988

```
[1] "The Frank copula parameter theta is -12.6"
```



[36]:
```
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)
```



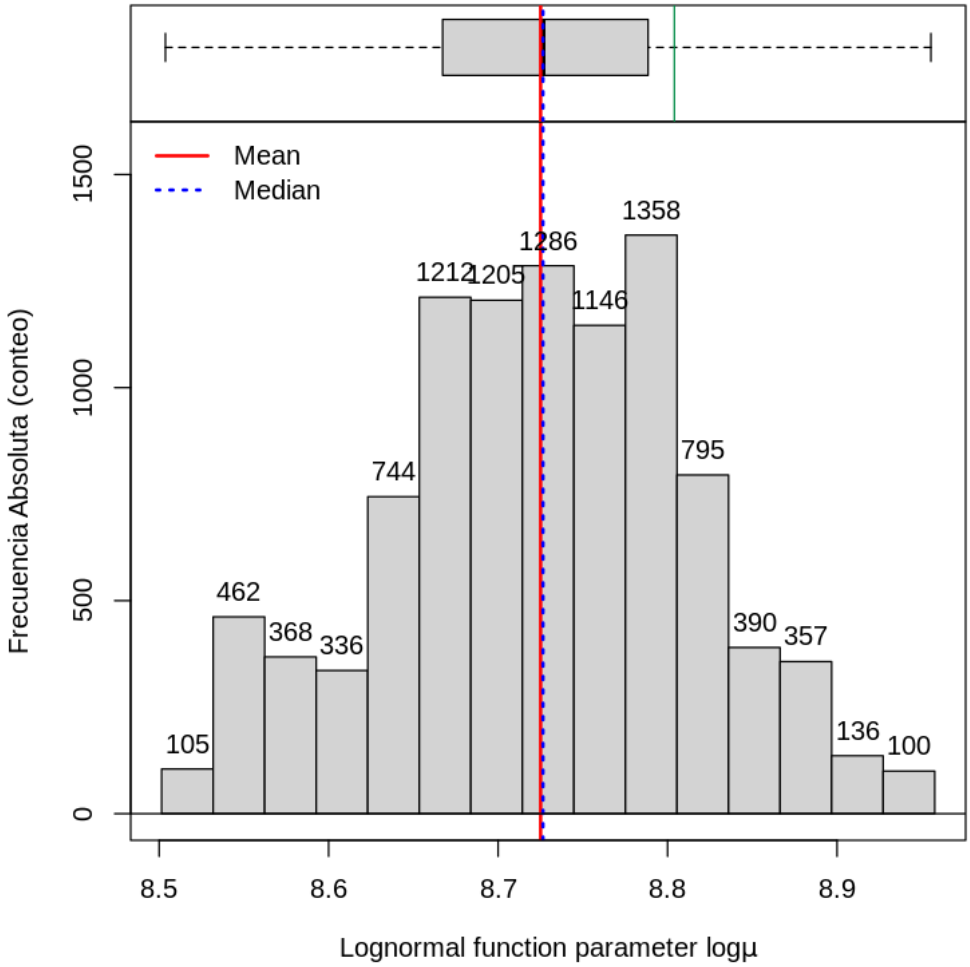
[37]:
```
mu_best_post5=mfv(round(mcmc_5.fit$par[,2],3))
HistBoxplot(mcmc_5.fit$par[,2],nbins = nb, mean = lnorm_mu_post[5,2], median =␣
 ↪lnorm_mu_post[4,2], main ="",
            xlab = expression(paste("Lognormal function parameter log",mu )),
            ylab = "Frecuencia Absoluta (conteo)", AbsFreq = TRUE, PercentFreq =␣
 ↪FALSE )
par(new=TRUE)
```

```r
abline(v=mu_best_post5, col=("springgreen4"))

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

A data.frame: 14 × 2

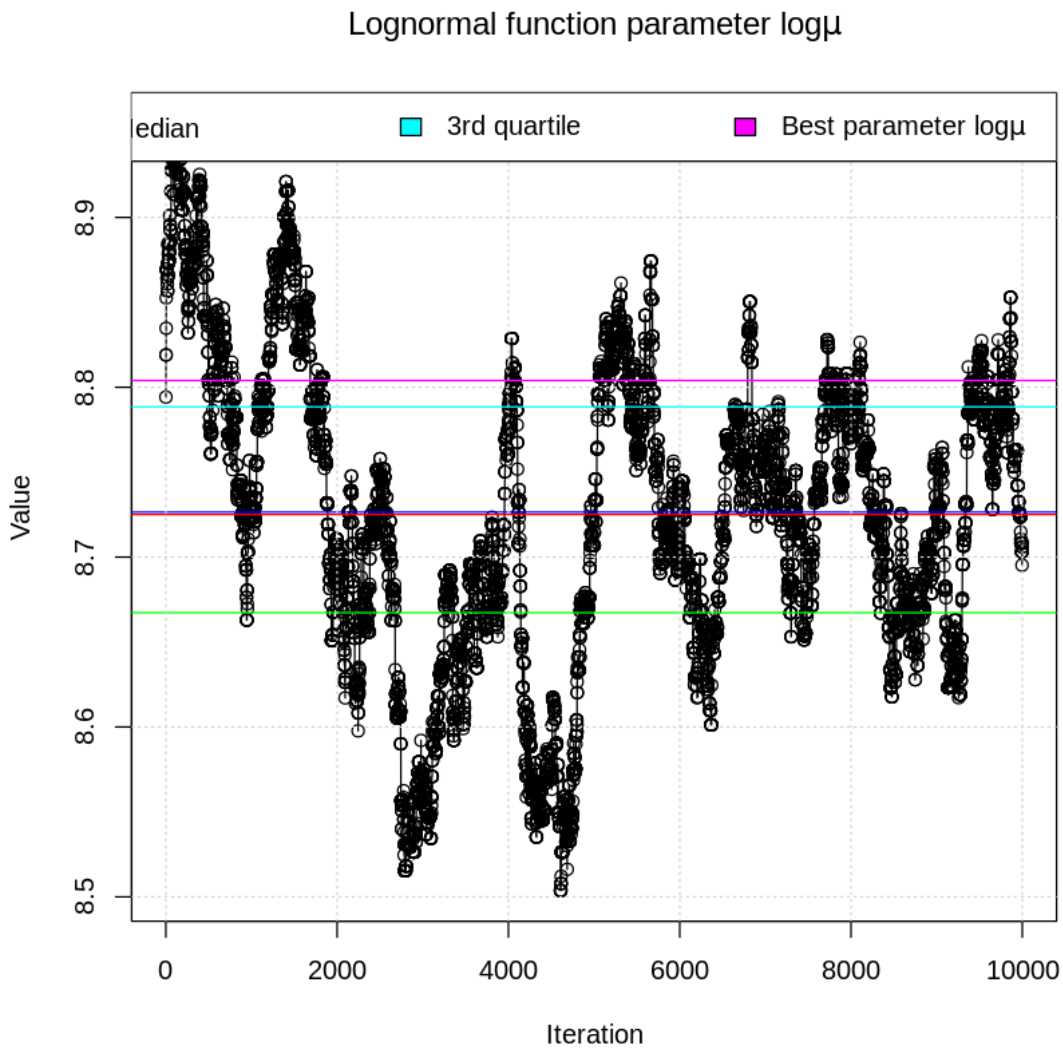
	Statistics <chr>	Values <dbl>
muestras	n	10000.0000
minimos	Minimum	8.5037
cuantiles1	1st. Quartile	8.6673
medianas	Median	8.7265
medias	Mean	8.7251
cuantiles3	3rd. Quartile	8.7885
maximos	Maximum	8.9554
rangos	Rank	0.4516
rangosInt	Interquartile Rank	0.1212
varianzas	Variance	0.0080
desvs	Standard Deviation	0.0897
CVs	Variation Coeff.	0.0103
simetrias	Skewness	-0.1156
curtosiss	Kurtosis	-0.3539

```
[1] "The Lognormal density function parameter mu is 8.804"
```



[38]:
```
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)
```



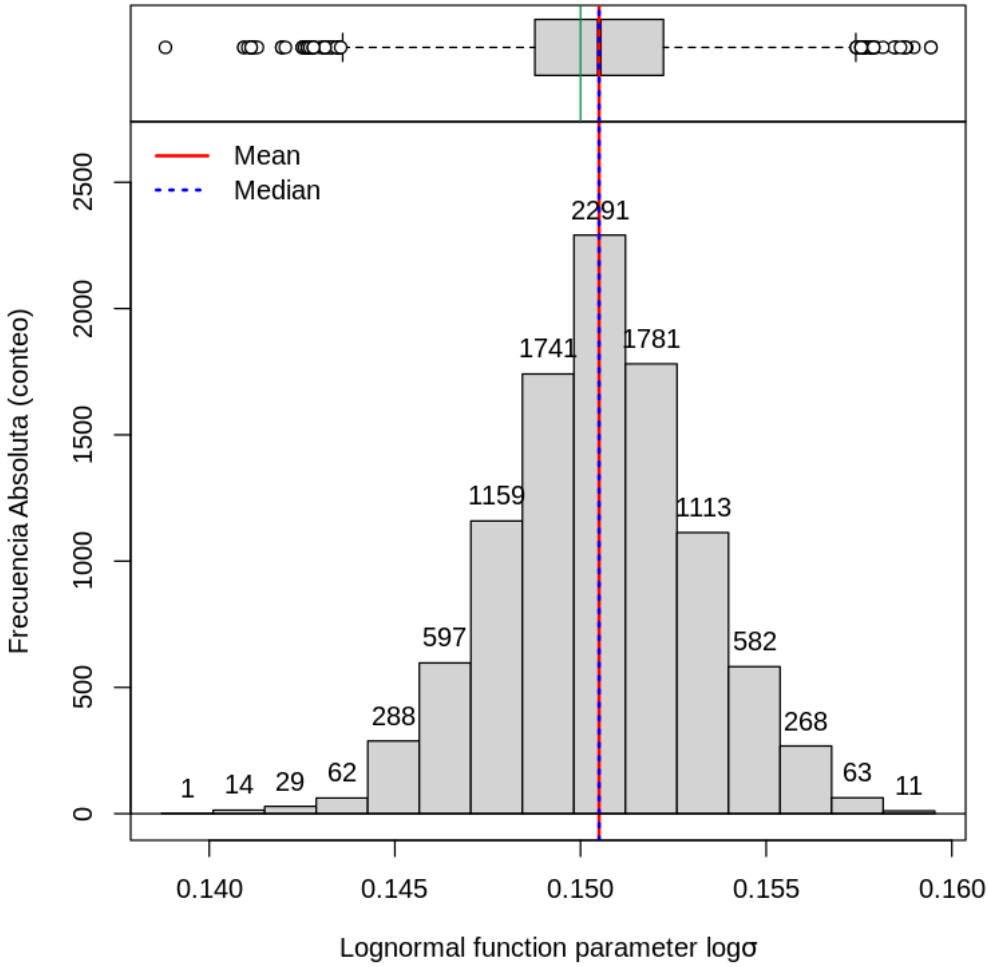
[39]:
```
sigma_best_post5=mfv(round(mcmc_5.fit$par[,3],3))
HistBoxplot(mcmc_5.fit$par[,3],nbins = nb, mean = lnorm_sigma_post[5,2], median␣
 ↪= lnorm_sigma_post[4,2],
            main ="",
            xlab = expression(paste("Lognormal function parameter log",sigma )),
            ylab = "Frecuencia Absoluta (conteo)",
            AbsFreq = TRUE, PercentFreq = FALSE )
par(new=TRUE)
```

```r
abline(v=sigma_best_post5, col=("springgreen4"))
lnorm_sigma_post
print(paste('The Lognormal density function parameter sigma␣
↪is',sigma_best_post5))
```

A data.frame: 14 × 2

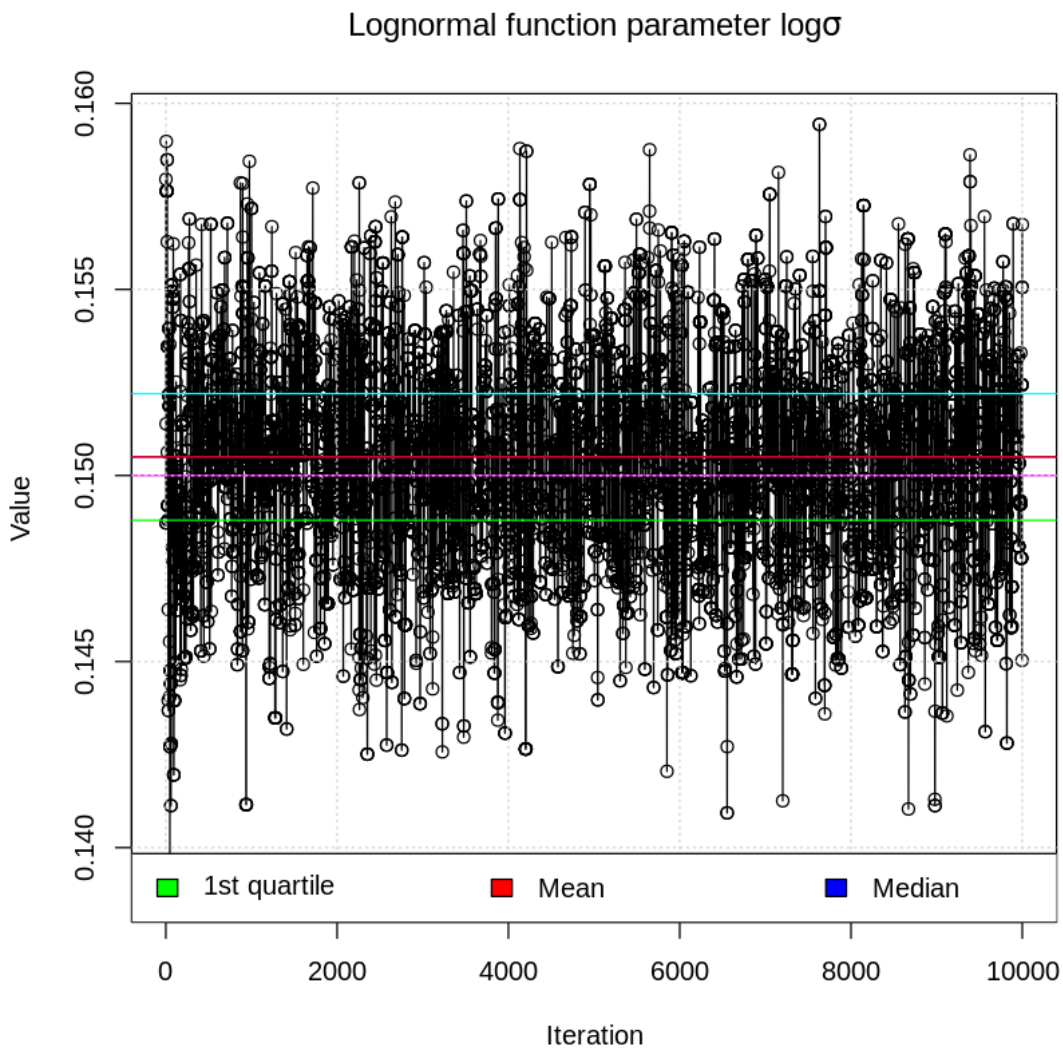
	Statistics <chr>	Values <dbl>
muestras	n	10000.0000
minimos	Minimum	0.1388
cuantiles1	1st. Quartile	0.1488
medianas	Median	0.1505
medias	Mean	0.1505
cuantiles3	3rd. Quartile	0.1522
maximos	Maximum	0.1594
rangos	Rank	0.0206
rangosInt	Interquartile Rank	0.0035
varianzas	Variance	0.0000
desvs	Standard Deviation	0.0027
CVs	Variation Coeff.	0.0178
simetrias	Skewness	-0.0829
curtosiss	Kurtosis	0.1321

```
[1] "The Lognormal density function parameter sigma is 0.15"
```



[40]:
```
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)
```



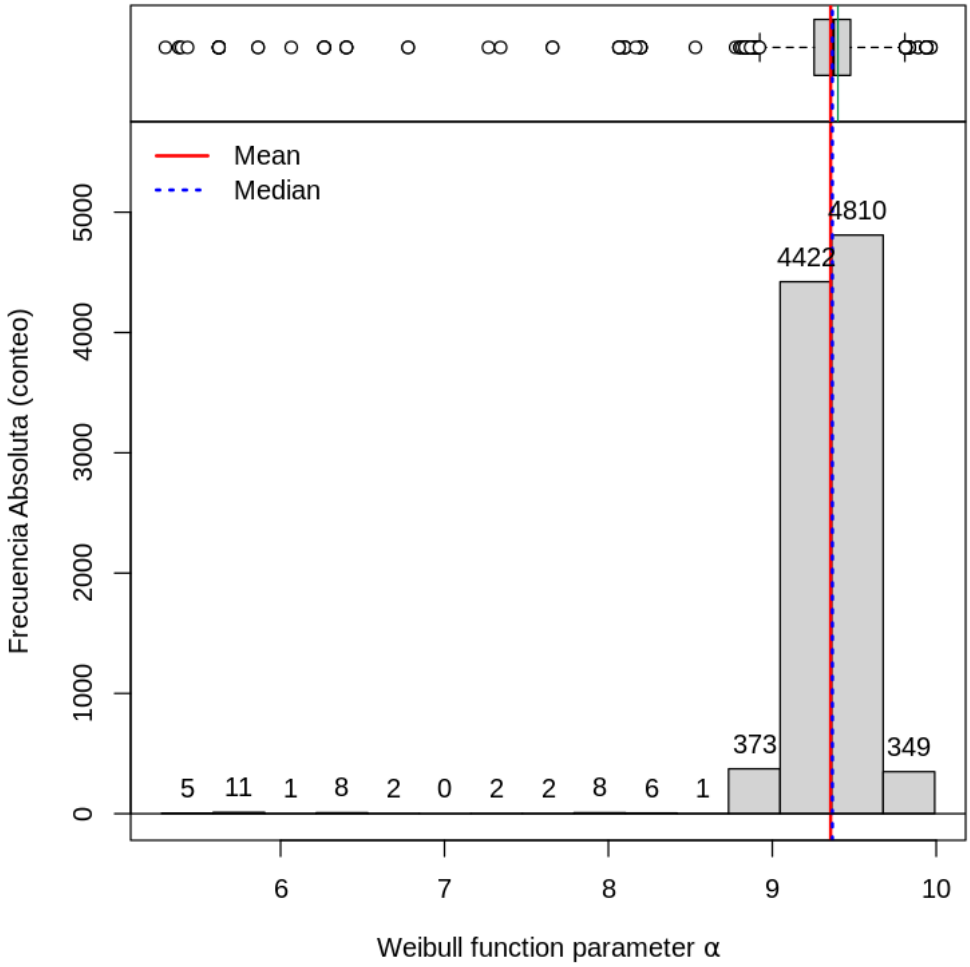
[41]:
```
alpha_best_post5=mfv(round(mcmc_5.fit$par[,4],1))

HistBoxplot(mcmc_5.fit$par[,4],nbins = nb, 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 is',alpha_best_post5))
```

A data.frame: 14 × 2

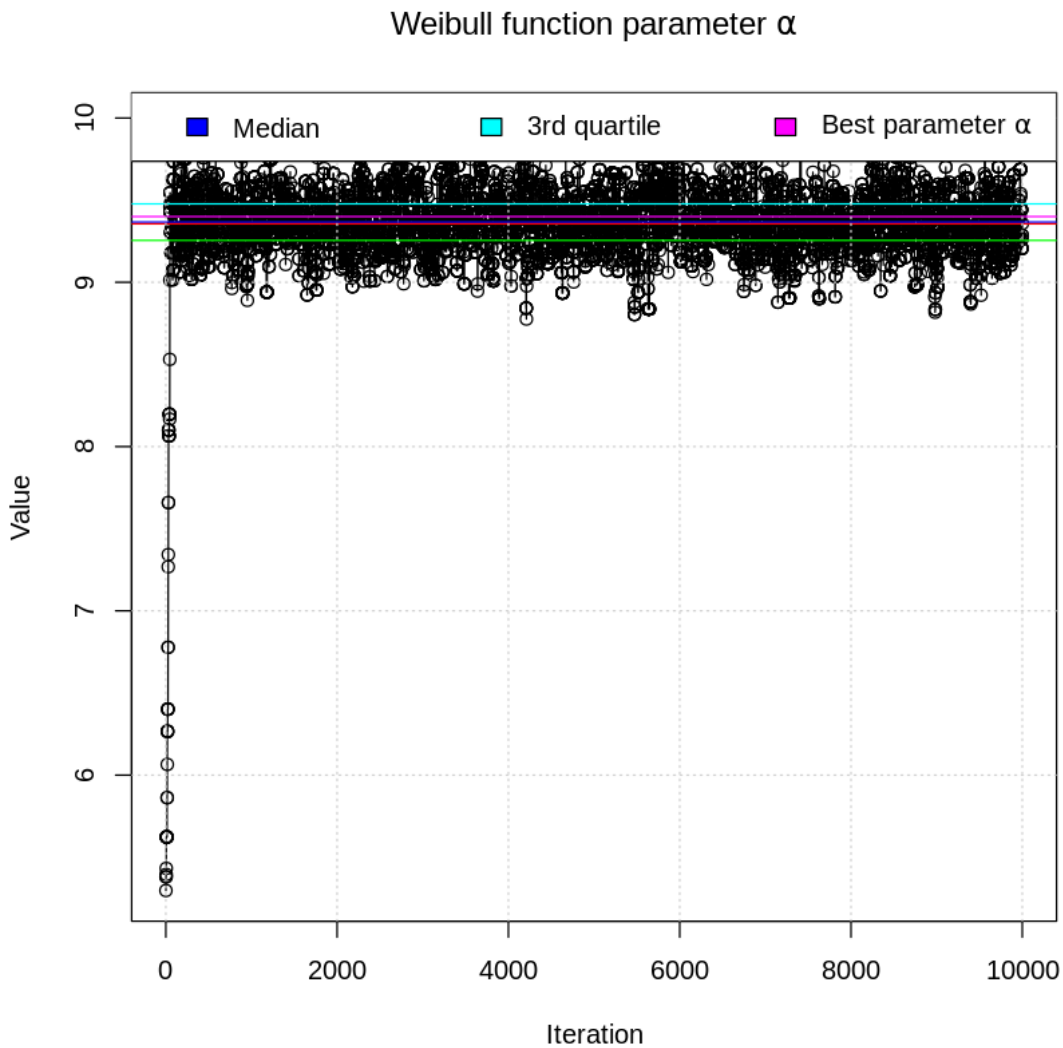
	Statistics <chr>	Values <dbl>
muestras	n	10000.0000
minimos	Minimum	5.2976
cuantiles1	1st. Quartile	9.2550
medianas	Median	9.3664
medias	Mean	9.3558
cuantiles3	3rd. Quartile	9.4772
maximos	Maximum	9.9674
rangos	Rank	4.6698
rangosInt	Interquartile Rank	0.2222
varianzas	Variance	0.0661
desvs	Standard Deviation	0.2572
CVs	Variation Coeff.	0.0275
simetrias	Skewness	-7.0816
curtosiss	Kurtosis	94.9841

```
[1] "The Weibull density function parameter alpha is 9.4"
```




```
[42]: 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)
```



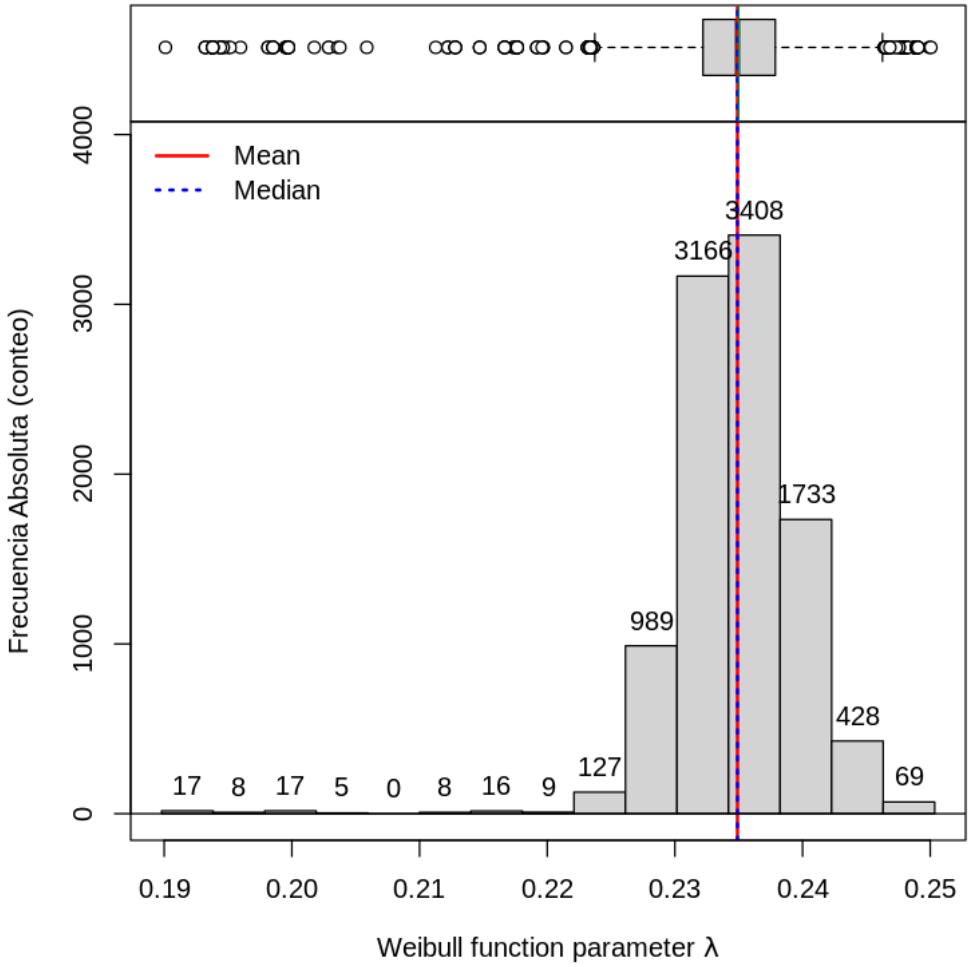
[43]:
```
lambda_best_post5=mfv(round(mcmc_5.fit$par[,5],3))
HistBoxplot(mcmc_5.fit$par[,5],nbins = nb, mean = weibull_lambda_post[5,2],␣
 ↪median = weibull_lambda_post[4,2],
            main ="",
            xlab = expression(paste("Weibull function parameter ",lambda )),
            ylab = "Frecuencia Absoluta (conteo)",
            AbsFreq = TRUE, PercentFreq = FALSE )
par(new=TRUE)
```

```r
abline(v=lambda_best_post5, col=("springgreen4"))
weibull_lambda_post
print(paste('The Weibull density function parameter lambda␣
↪is',lambda_best_post5))
```

A data.frame: 14 × 2

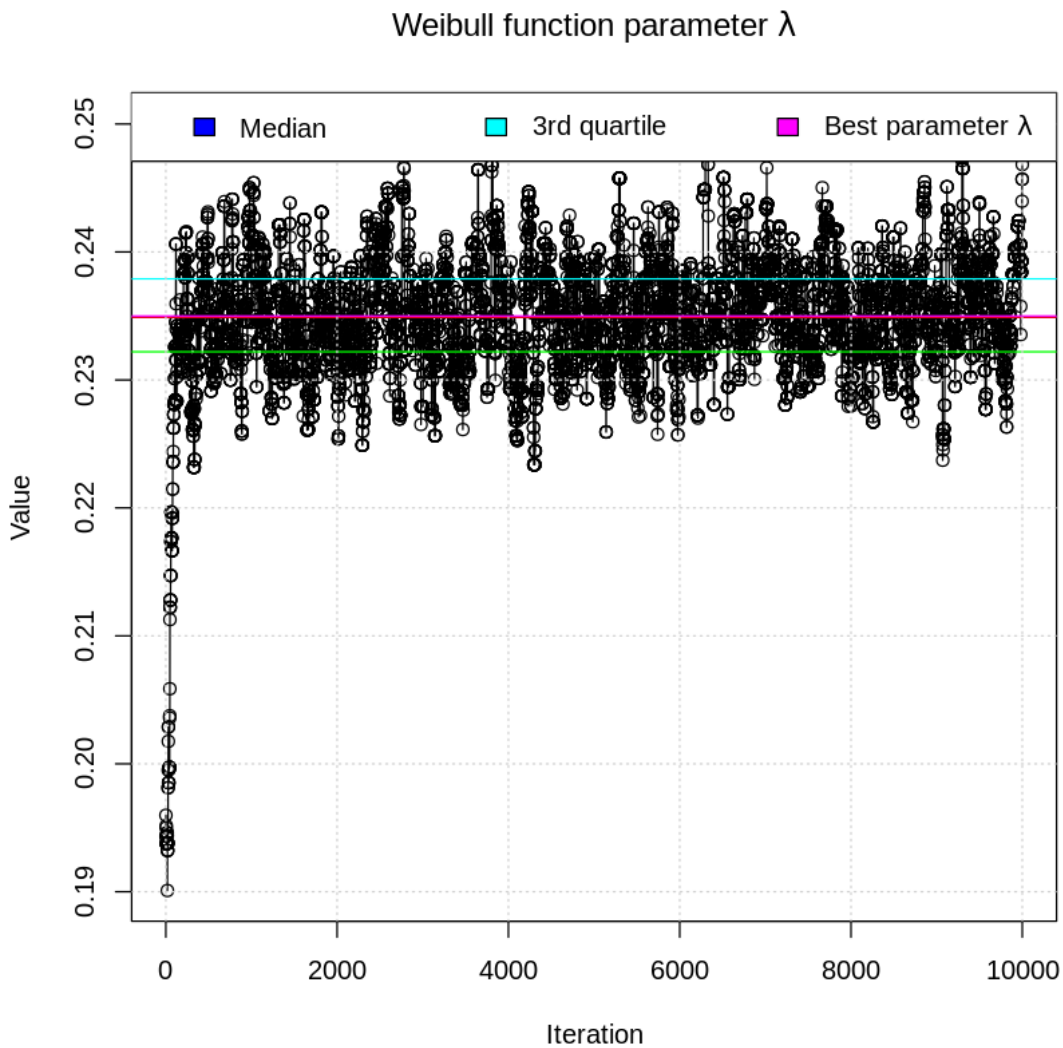
	Statistics <chr>	Values <dbl>
muestras	n	10000.0000
minimos	Minimum	0.1901
cuantiles1	1st. Quartile	0.2322
medianas	Median	0.2349
medias	Mean	0.2349
cuantiles3	3rd. Quartile	0.2379
maximos	Maximum	0.2501
rangos	Rank	0.0600
rangosInt	Interquartile Rank	0.0057
varianzas	Variance	0.0000
desvs	Standard Deviation	0.0051
CVs	Variation Coeff.	0.0215
simetrias	Skewness	-2.0063
curtosiss	Kurtosis	14.9353

```
[1] "The Weibull density function parameter lambda is 0.235"
```



[44]:
```
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)
```

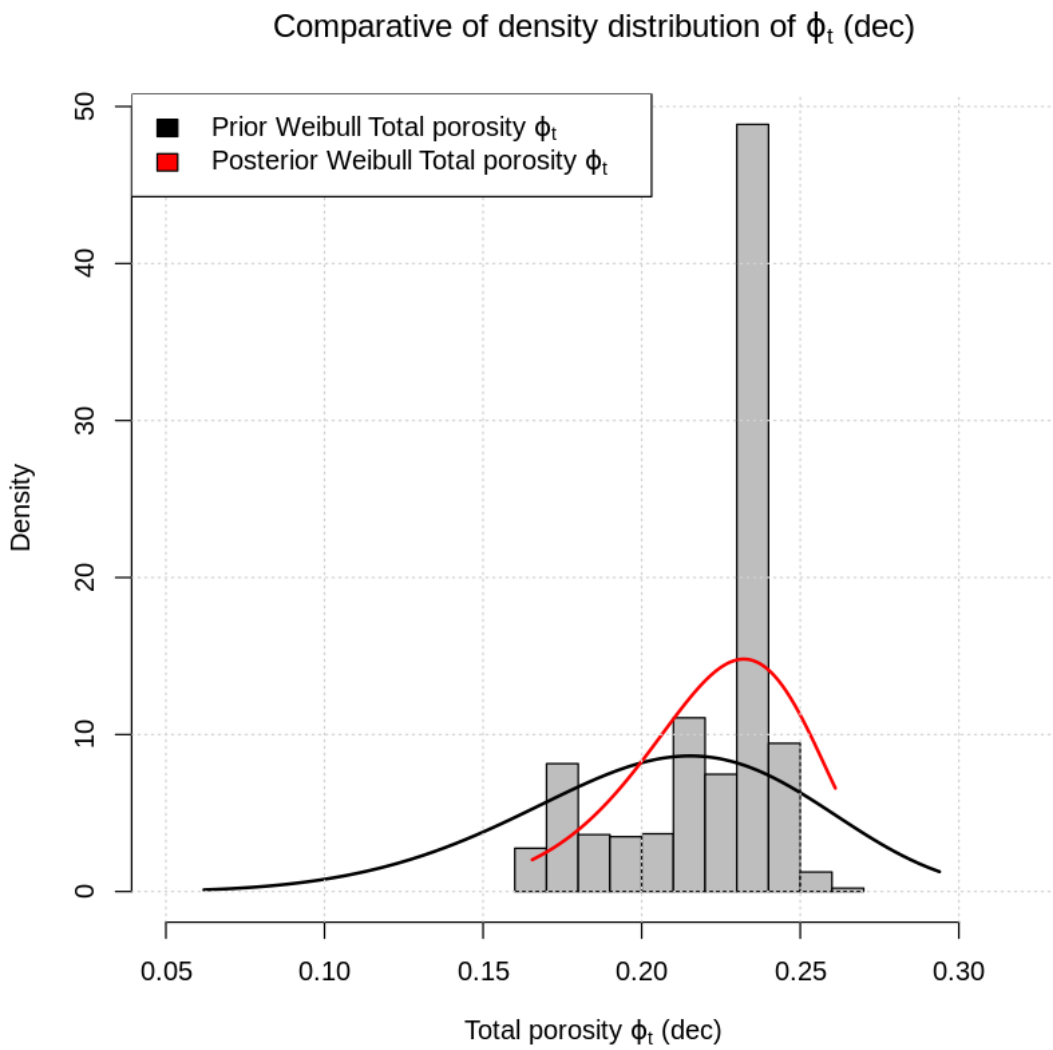


In the first iteration, the prior parameter values were initialized using the optimal fits to well-log samples based on the Akaike Information Criterion (AIC). These values are listed in Table **??**, second column. The parameters were then updated using the upscaled geophysical well logs. To maintain the desired acceptance rate of 23.5%, only four of the five parameters were used in the update, as the initial θ parameter restricted convergence. As shown in Table **??**, fourth column, the parameters of the lognormal and Weibull distributions exhibit minimal change. However, the comparison of prior and posterior cumulative distributions in Figures **??**A and **??**C shows significant shifts. The posterior distribution is considered acceptable given its proximity to the empirical distribution of the upscaled logs. Although the posterior θ value increased to -12.6,

this aligns with the increased dependence observed in Table **??**, and is also visible in Figures **??**E and **??**F, which show how the model captures the distribution of pseudo-observations.

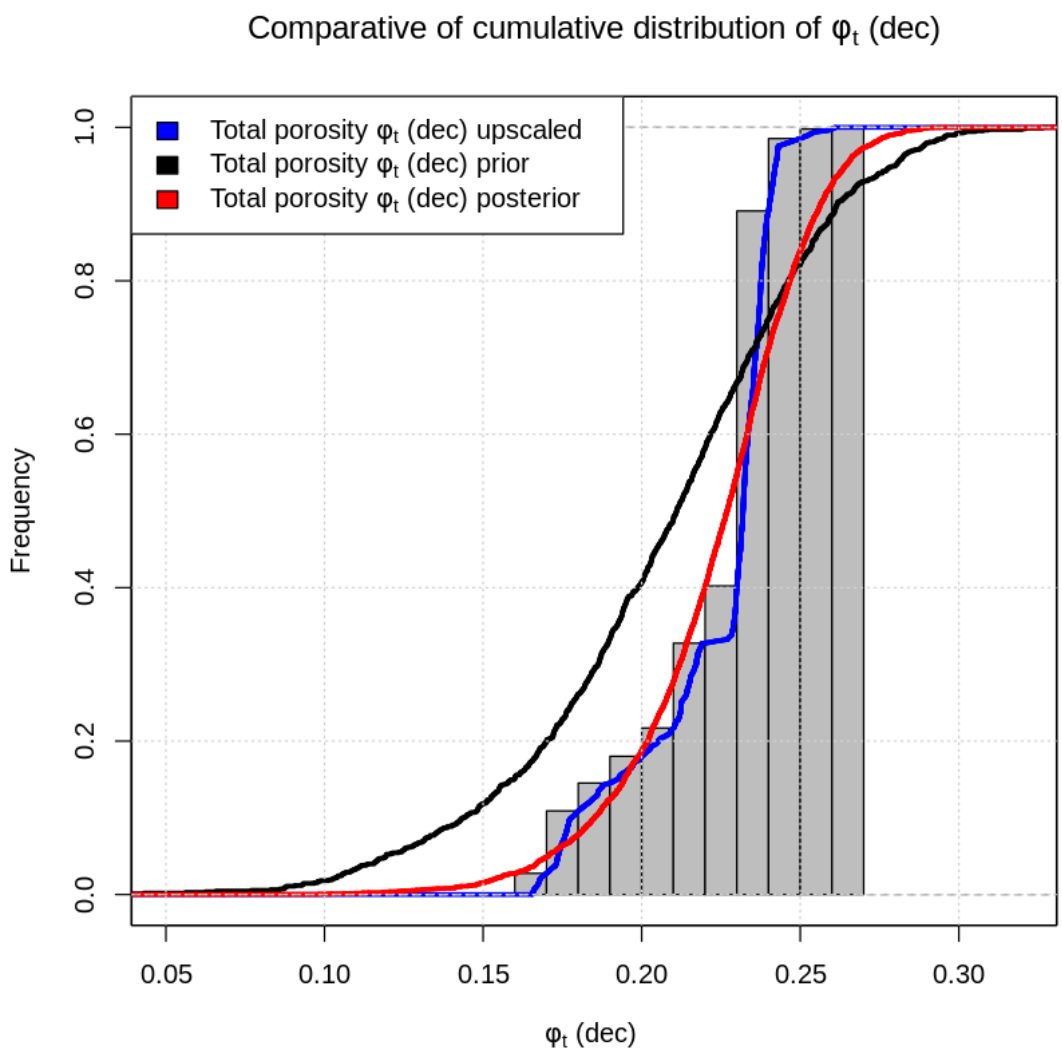
[45]:
```
cop_frank_postF <- mvdc(frankCopula(theta_best_post5), margins =␣
↪c("lnorm","weibull"),
                        paramMargins = list(list(meanlog = mu_best_post5, sdlog =␣
↪sigma_best_post5),
                                            list(shape = alpha_best_post5, scale␣
↪= lambda_best_post5)))

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

[46]:
```
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"))
```



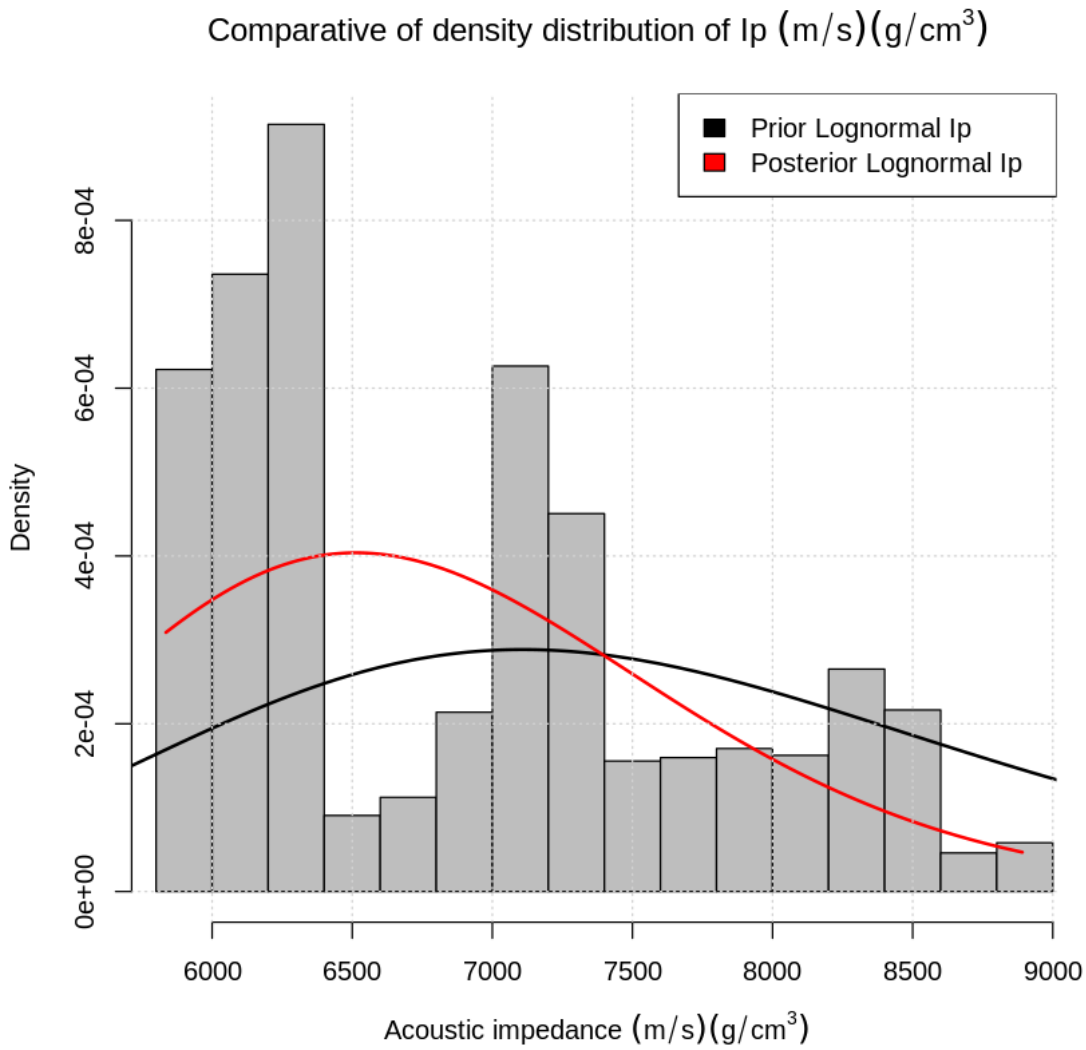
[47]:
```
hcum <- h <- hist(Phit_like, plot=FALSE, breaks=ns)
hcum$counts <- cumsum(hcum$counts)/sum(hcum$counts)
plot(hcum, xlim=c(0.05, 0.32), col=("gray"),
     main =expression(paste("Comparative of cumulative distribution of␣
 ↪",phi[t]," (dec)")),
     xlab= expression(paste(phi[t], " (dec) ")))
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"))
```



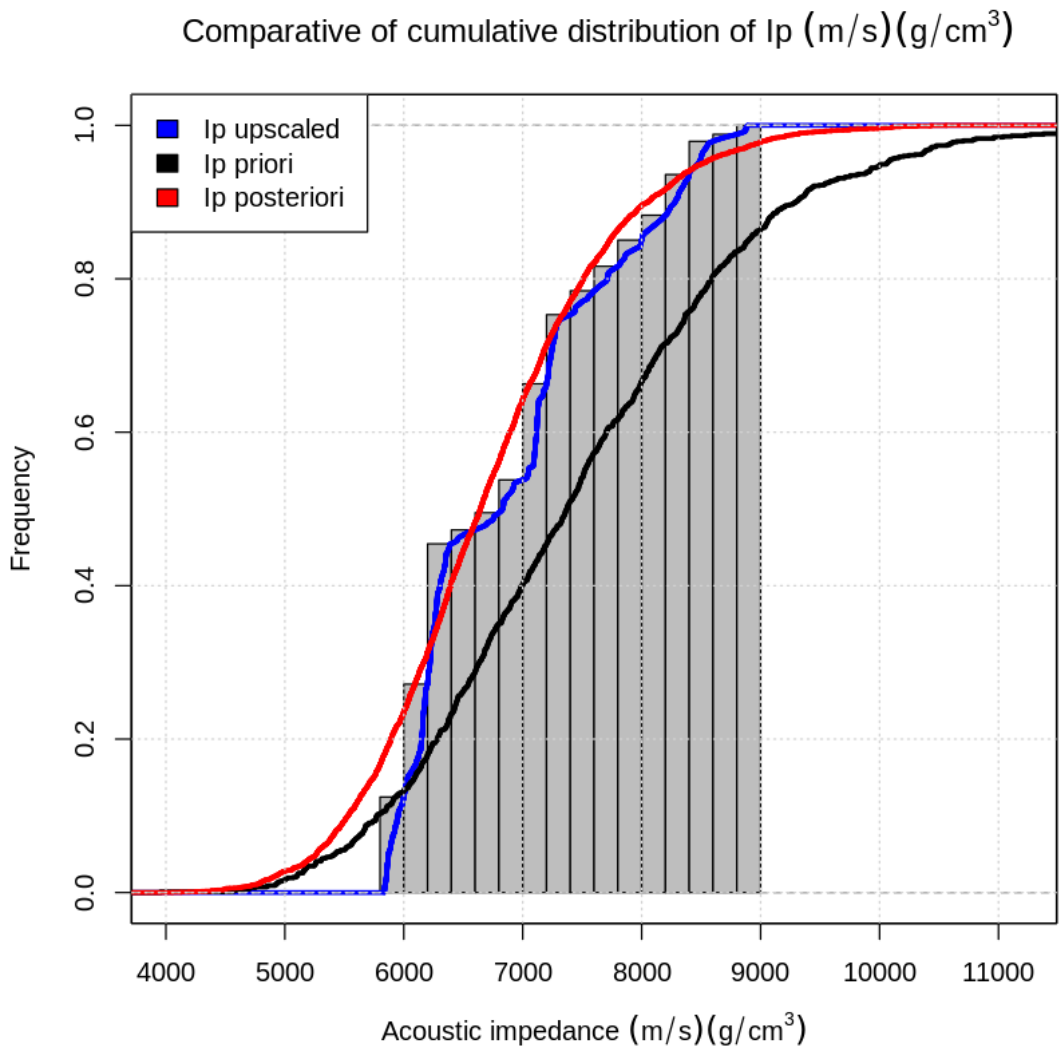
[48]:
```
den.x <- seq(min(Ip),max(Ip),length=3696)
den.y_prior <- dlnorm(den.x,meanlog = 8.905543, sdlog = 0.191057)
```

```
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"))
```



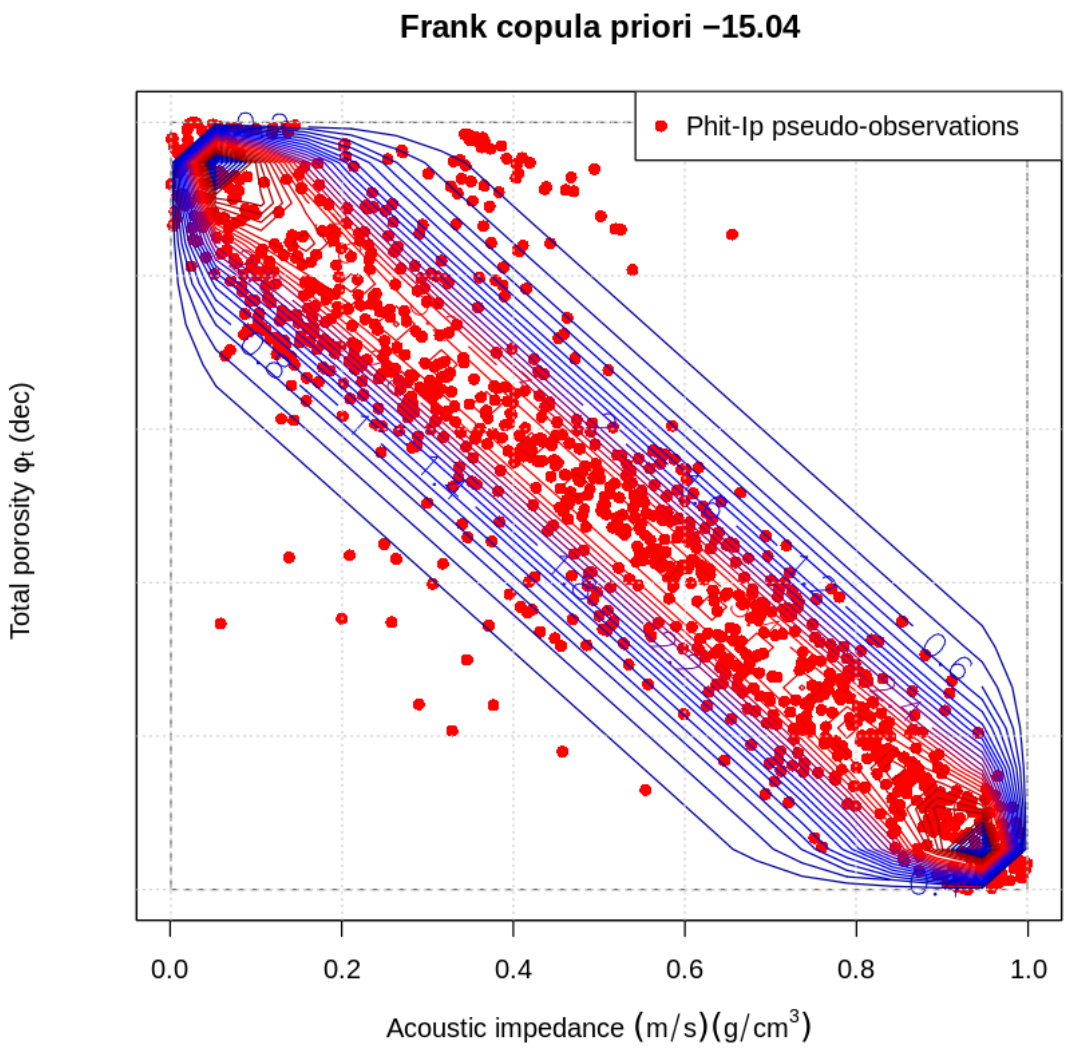
[49]:
```
hcum <- h <- hist(Ip_like, plot=FALSE, breaks=ns)
hcum$counts <- cumsum(hcum$counts)/sum(hcum$counts)
plot(hcum, xlim=c(4000, 11200), col=("gray"),
     main =(expression(paste(" Comparative of cumulative distribution of Ip ",(m/
 ↪s)(g/cm^3)))),
     xlab= expression(paste("Acoustic impedance ",(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"))
```



[50]:
```
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)
```

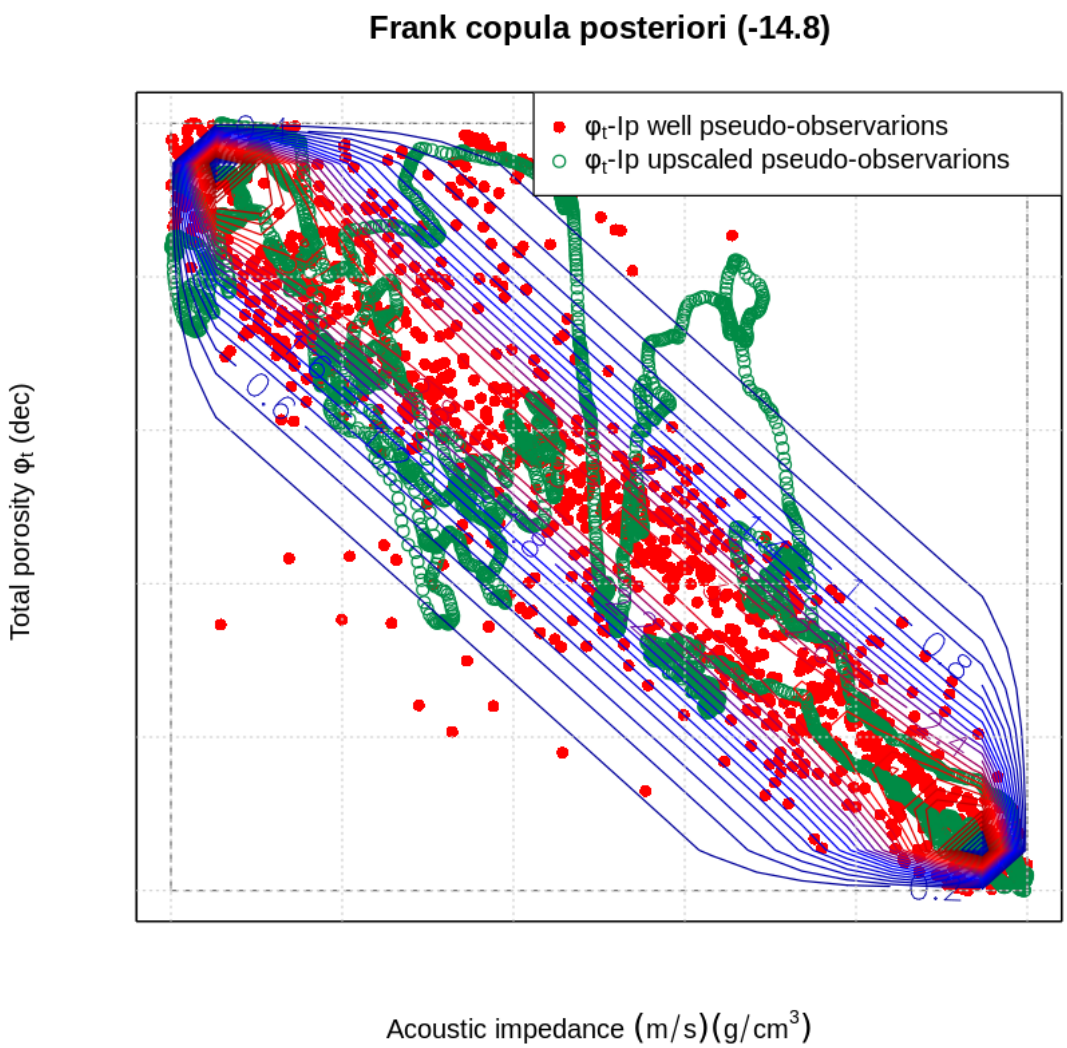
```
contour(frankCopula(-15.04), dCopula,n=20,nlevels=25, xlab=("“), ylab=("“)␣
↪,labcex=1.4, col=cols,xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend = c("Phit-Ip pseudo-observations"), col = c(␣
↪"red"),pch=16)
```



[51]:
```
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 ",phi[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(phi[t], "-Ip well␣
 ↪pseudo-observarions")),
                                  expression(paste(phi[t], "-Ip upscaled␣
 ↪pseudo-observarions"))),
       col = c("red", "springgreen4"),pch=c(16,1))

theta_best_post5
```

-12.6



8.5 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)	8.8	1.7
Lognormal (σ)	0.15	1.7
Weibull (α)	9.4	1.7
Weibull (λ)	0.23	1.7
Frank copula (θ)	-12.6	2.3

[52]:
```
Data_like_TRA<-cbind(Ip_like_SUB,Phit_like_SUB)
```

[53]:
```
#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
```

[54]:
```
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(theta_best_post5,mu_best_post5,sigma_best_post5,alpha_best_post5,lambda_best
```

[55]:
```
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=theta_best_post5, sd=((theta_best_post5/100)*2.
↪3), log=TRUE)
  mulog_prior=dnorm(mulog, mean=mu_best_post5, sd=((mu_best_post5/100)*1.7),␣
↪log=TRUE)
```

```
  sigmalog_prior=dnorm(sigmalog, mean=sigma_best_post5, sd=((sigma_best_post5/
↪100)*1.7), log=TRUE)
  alpha_prior=dnorm(alpha, mean=alpha_best_post5, sd=((alpha_best_post5/100)*1.
↪7), log=TRUE)
  lambda_prior=dnorm(lambda, mean=lambda_best_post5, sd=((lambda_best_post5/
↪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
}
```

[56]:
```
suppressWarnings({mcmc_5TRA.fit <-rwmetrop(logpost_5TRA,
                    list(var=v_tot_post_5TRA,scale=2),
                    start_5TRA,
                    10000,
                    Data_like_TRA)})
```

[57]:
```
mcmc_5TRA.fit$accept
```

0.2153

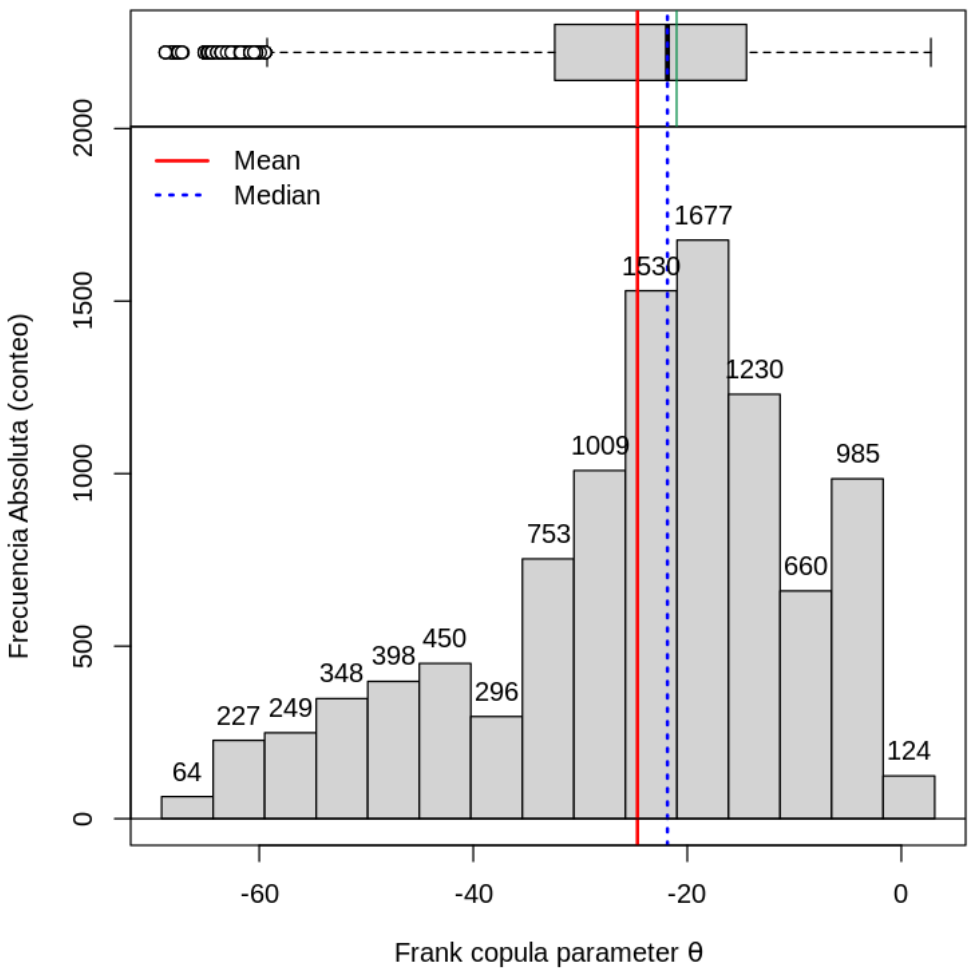
[58]:
```
frank_theta_post_5TRA<-Estadisticas(mcmc_5TRA.fit$par[,1])
lnorm_mu_post_5TRA<-Estadisticas(mcmc_5TRA.fit$par[,2])
lnorm_sigma_post_5TRA<-Estadisticas(mcmc_5TRA.fit$par[,3])
weibull_alpha_post_5TRA<-Estadisticas(mcmc_5TRA.fit$par[,4])
weibull_lambda_post_5TRA<-Estadisticas(mcmc_5TRA.fit$par[,5])
```

[59]:
```
theta_best_post_5TRA=mfv(round(mcmc_5TRA.fit$par[,1],0))
HistBoxplot(mcmc_5TRA.fit$par[,1],nbins = nb, 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))
```

A data.frame: 14 × 2

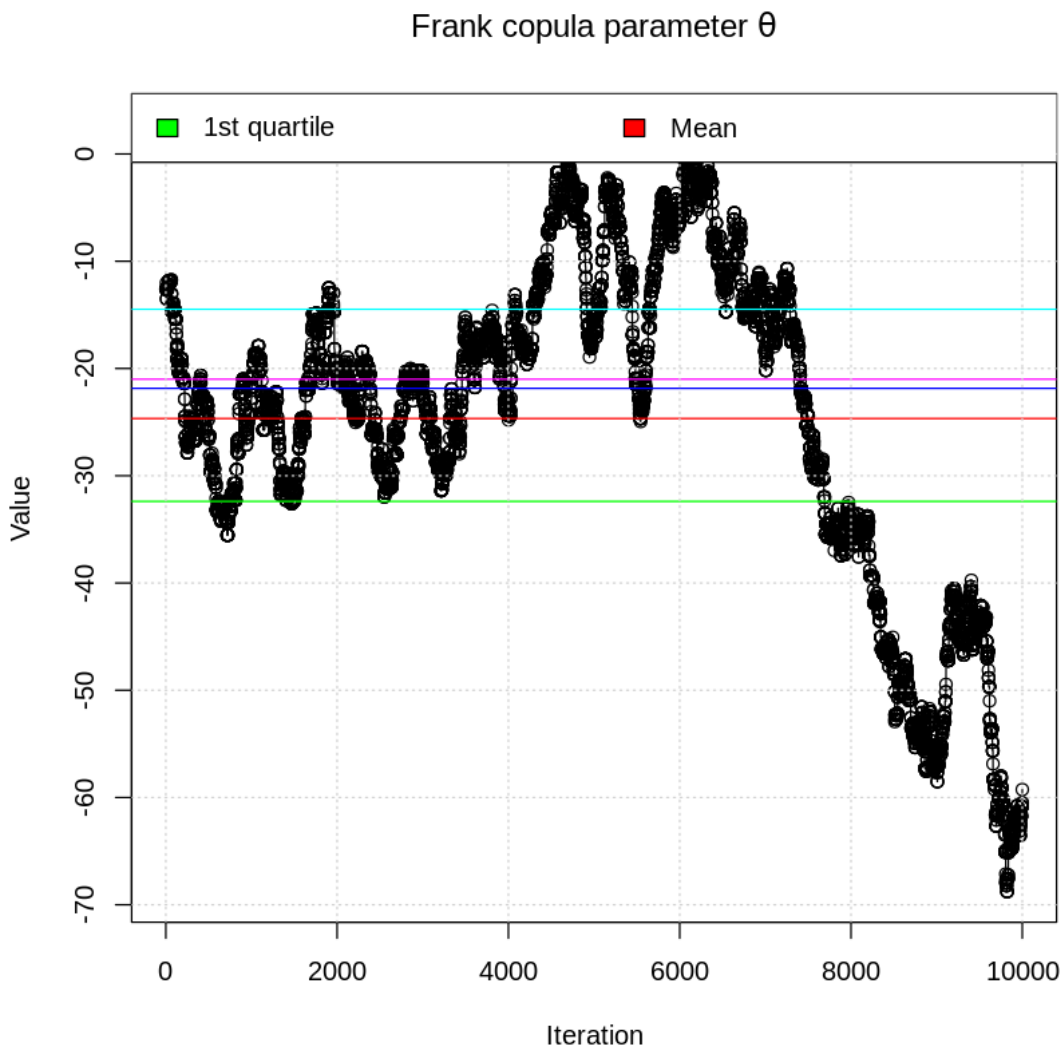
	Statistics <chr>	Values <dbl>
muestras	n	10000.0000
minimos	Minimum	-68.7668
cuantiles1	1st. Quartile	-32.3889
medianas	Median	-21.8565
medias	Mean	-24.6627
cuantiles3	3rd. Quartile	-14.4804
maximos	Maximum	2.7612
rangos	Rank	71.5280
rangosInt	Interquartile Rank	17.9086
varianzas	Variance	228.7837
desvs	Standard Deviation	15.1256
CVs	Variation Coeff.	-0.6133
simetrias	Skewness	-0.7364
curtosiss	Kurtosis	-0.0114

```
[1] "The Frank copula parameter theta is -21"
```



[60]:
```
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)
```



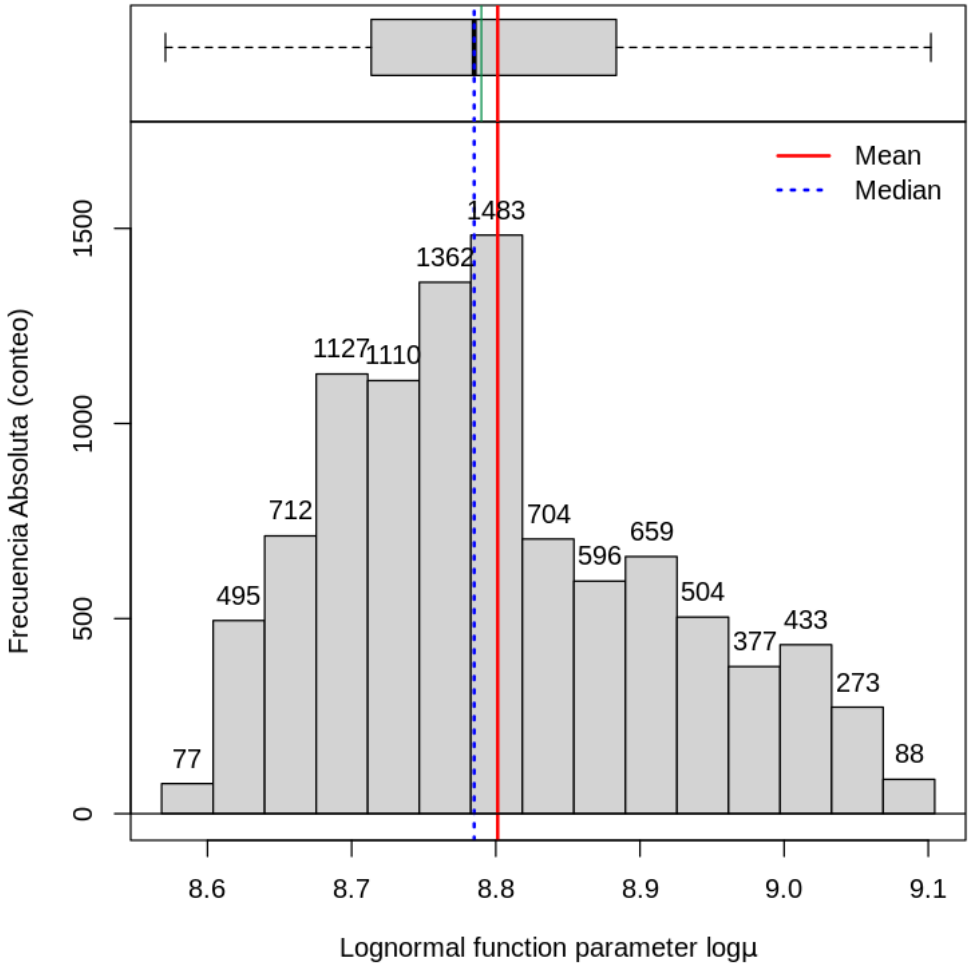
[62]:
```
mu_best_post_5TRA=mfv(round(mcmc_5TRA.fit$par[,2],2))
HistBoxplot(mcmc_5TRA.fit$par[,2],nbins = nb, mean = lnorm_mu_post_5TRA[5,2],
            median = lnorm_mu_post_5TRA[4,2], main ="",
            xlab = expression(paste("Lognormal function parameter log",mu )),
            ylab = "Frecuencia Absoluta (conteo)", AbsFreq = TRUE, PercentFreq =␣
 ↪FALSE )
par(new=TRUE)
```

```r
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))
```

A data.frame: 14 × 2

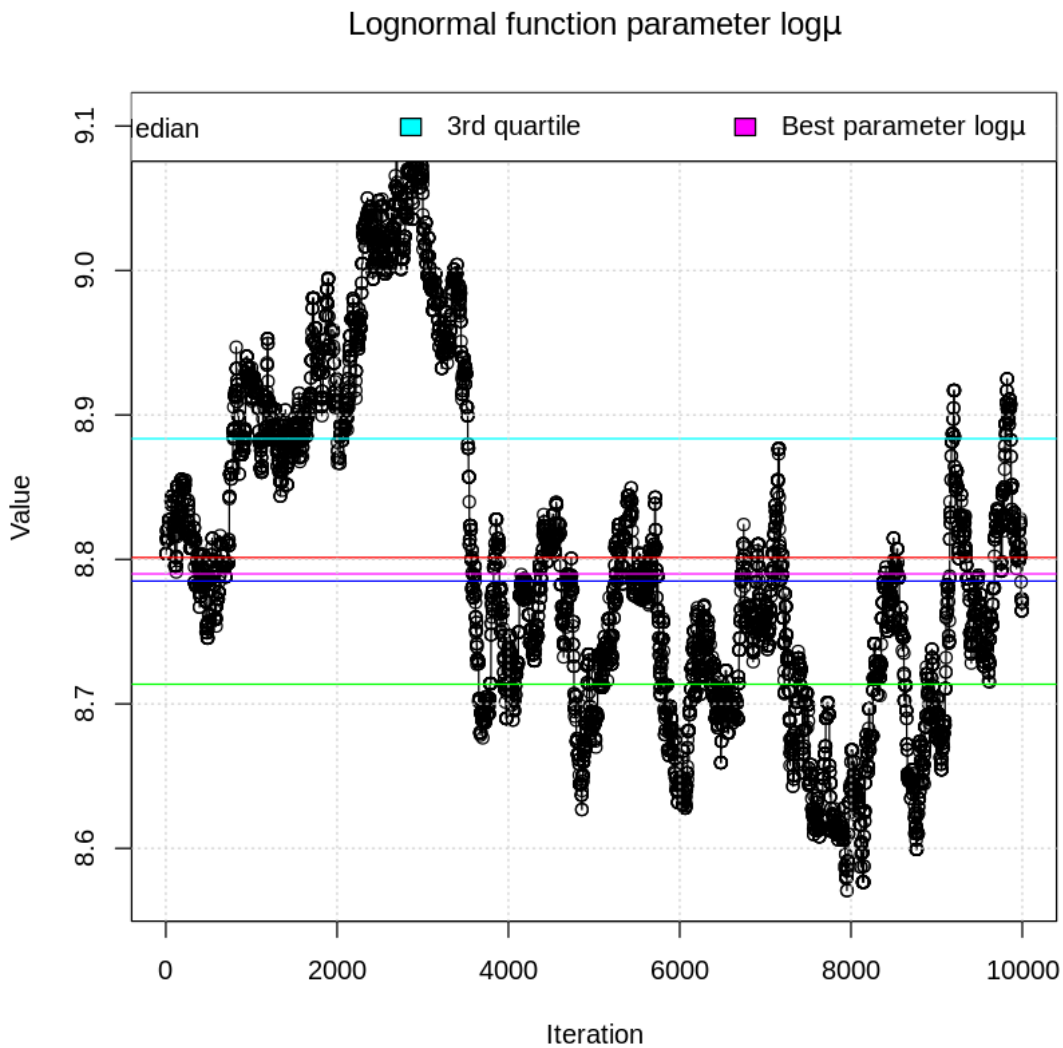
	Statistics <chr>	Values <dbl>
muestras	n	10000.0000
minimos	Minimum	8.5708
cuantiles1	1st. Quartile	8.7136
medianas	Median	8.7850
medias	Mean	8.8013
cuantiles3	3rd. Quartile	8.8836
maximos	Maximum	9.1018
rangos	Rank	0.5310
rangosInt	Interquartile Rank	0.1700
varianzas	Variance	0.0133
desvs	Standard Deviation	0.1152
CVs	Variation Coeff.	0.0131
simetrias	Skewness	0.4746
curtosiss	Kurtosis	-0.5041

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



[63]:
```
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)
```



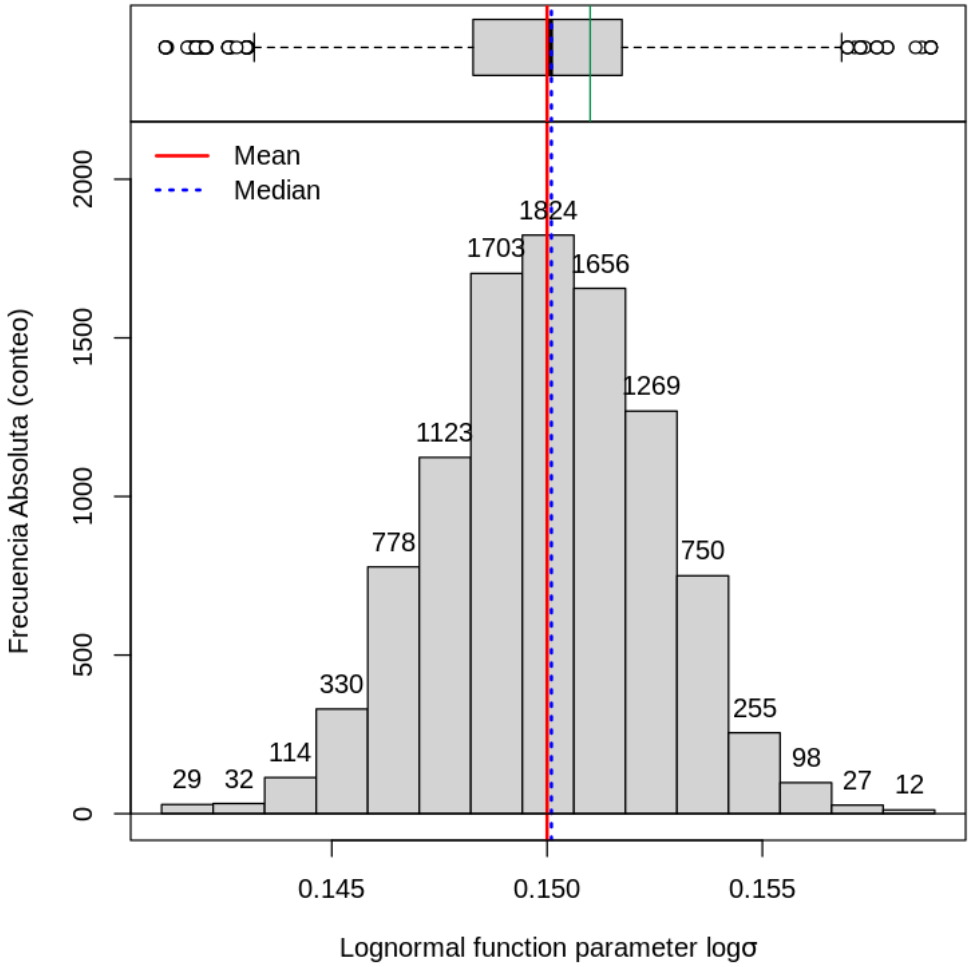
[64]:
```
sigma_best_post_5TRA=mfv(round(mcmc_5TRA.fit$par[,3],3))
HistBoxplot(mcmc_5TRA.fit$par[,3],nbins = nb, mean = lnorm_sigma_post_5TRA[5,2],␣
 ↪median = lnorm_sigma_post_5TRA[4,2],
            main ="",
            xlab = expression(paste("Lognormal function parameter log",sigma )),
            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))
```

A data.frame: 14 × 2

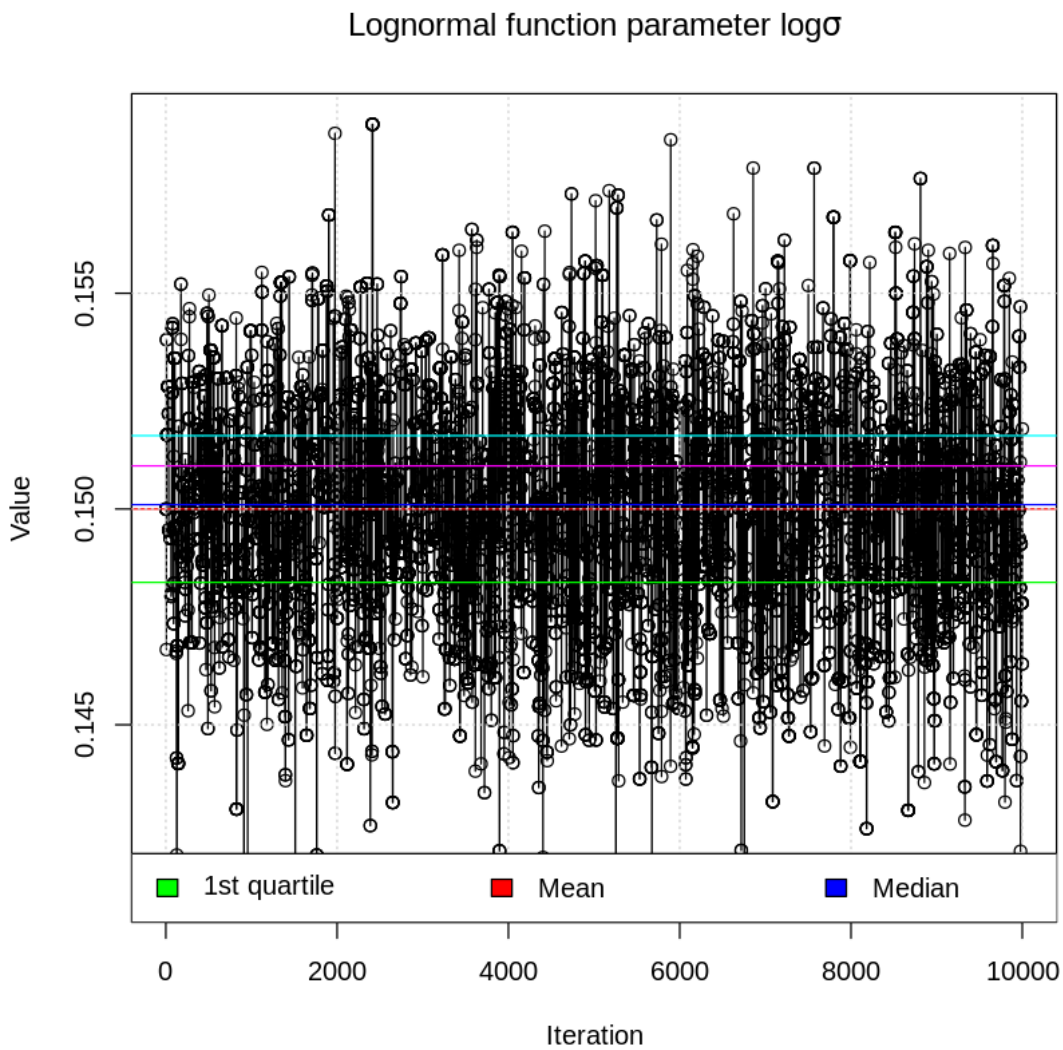
	Statistics <chr>	Values <dbl>
muestras	n	10000.0000
minimos	Minimum	0.1411
cuantiles1	1st. Quartile	0.1483
medianas	Median	0.1501
medias	Mean	0.1500
cuantiles3	3rd. Quartile	0.1517
maximos	Maximum	0.1589
rangos	Rank	0.0178
rangosInt	Interquartile Rank	0.0035
varianzas	Variance	0.0000
desvs	Standard Deviation	0.0025
CVs	Variation Coeff.	0.0169
simetrias	Skewness	-0.0901
curtosiss	Kurtosis	0.0514

```
[1] "The Lognormal density function parameter sigma is 0.151"
```



[65]:
```
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)
```



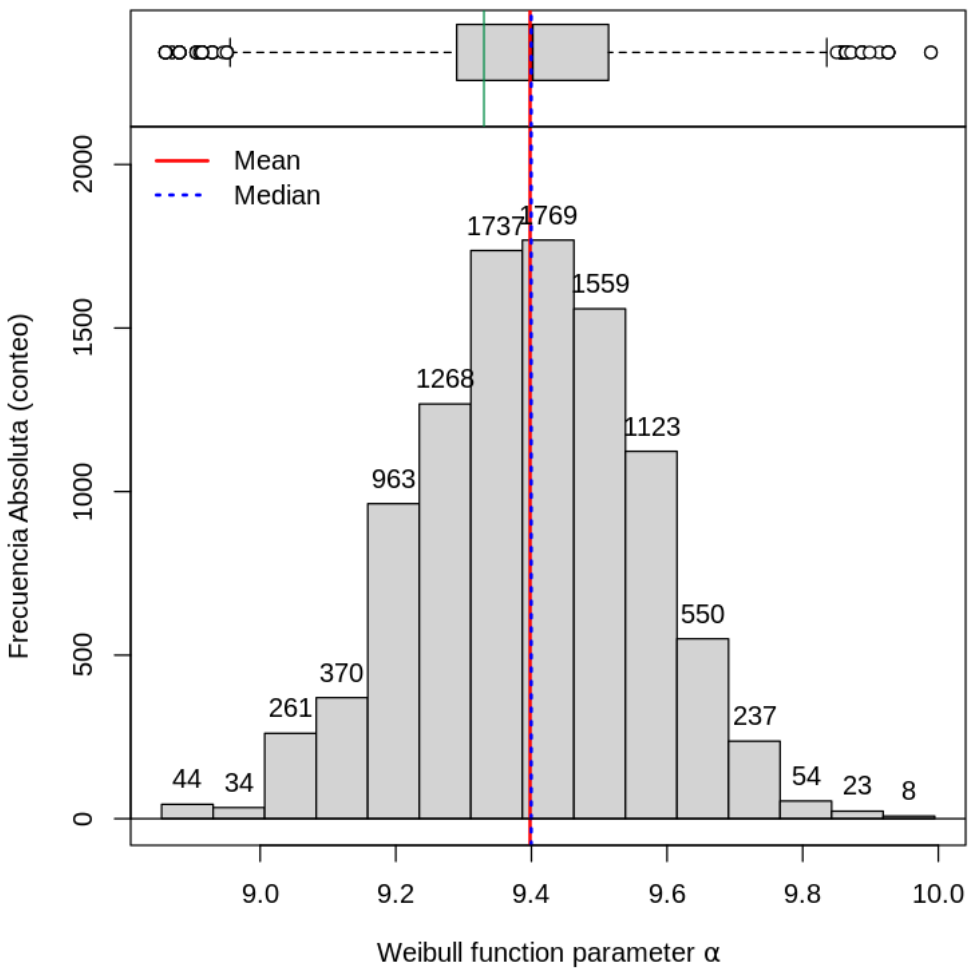
[67]:
```
alpha_best_post_5TRA=mfv(round(mcmc_5TRA.fit$par[,4],2))

HistBoxplot(mcmc_5TRA.fit$par[,4],nbins = nb, 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))
```

A data.frame: 14 × 2

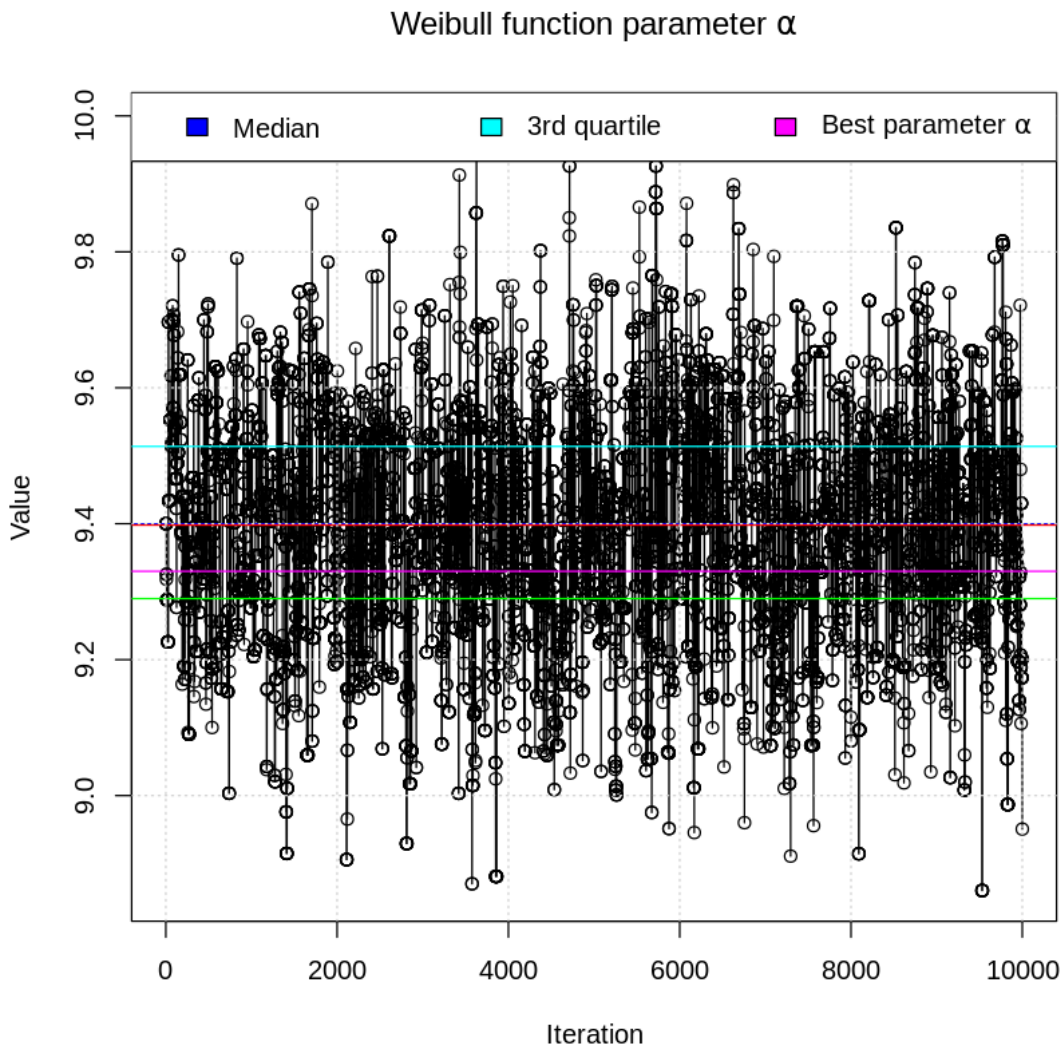
	Statistics <chr>	Values <dbl>
muestras	n	10000.0000
minimos	Minimum	8.8603
cuantiles1	1st. Quartile	9.2898
medianas	Median	9.3997
medias	Mean	9.3980
cuantiles3	3rd. Quartile	9.5137
maximos	Maximum	9.9890
rangos	Rank	1.1287
rangosInt	Interquartile Rank	0.2239
varianzas	Variance	0.0267
desvs	Standard Deviation	0.1633
CVs	Variation Coeff.	0.0174
simetrias	Skewness	-0.0922
curtosiss	Kurtosis	-0.0154

```
[1] "The Weibull density function parameter alpha 9.33"
```



[68]:
```
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)
```



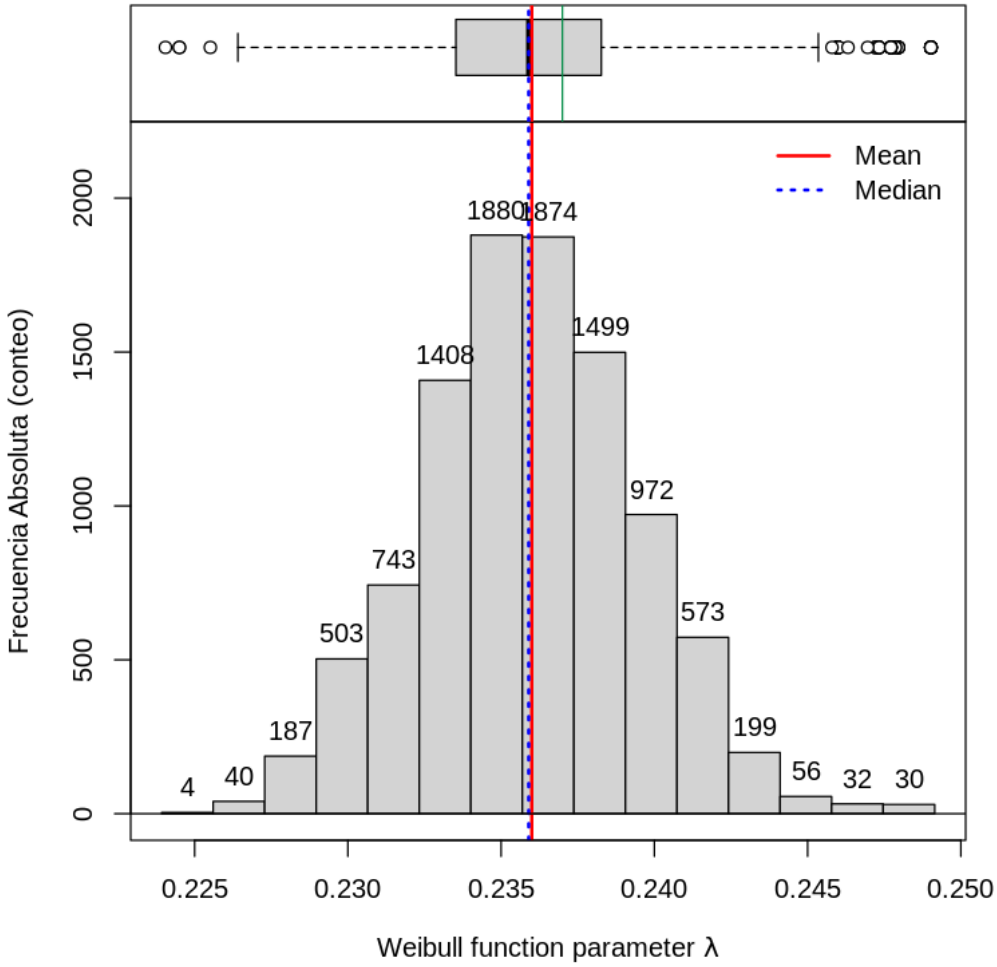
[69]:
```
lambda_best_post_5TRA=mfv(round(mcmc_5TRA.fit$par[,5],3))
HistBoxplot(mcmc_5TRA.fit$par[,5],nbins = nb, mean =␣
 ↪weibull_lambda_post_5TRA[5,2],
            median = weibull_lambda_post_5TRA[4,2],
            main ="",
            xlab = expression(paste("Weibull function parameter ",lambda )),
            ylab = "Frecuencia Absoluta (conteo)",
            AbsFreq = TRUE, PercentFreq = FALSE )
```

```r
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))
```

A data.frame: 14 × 2

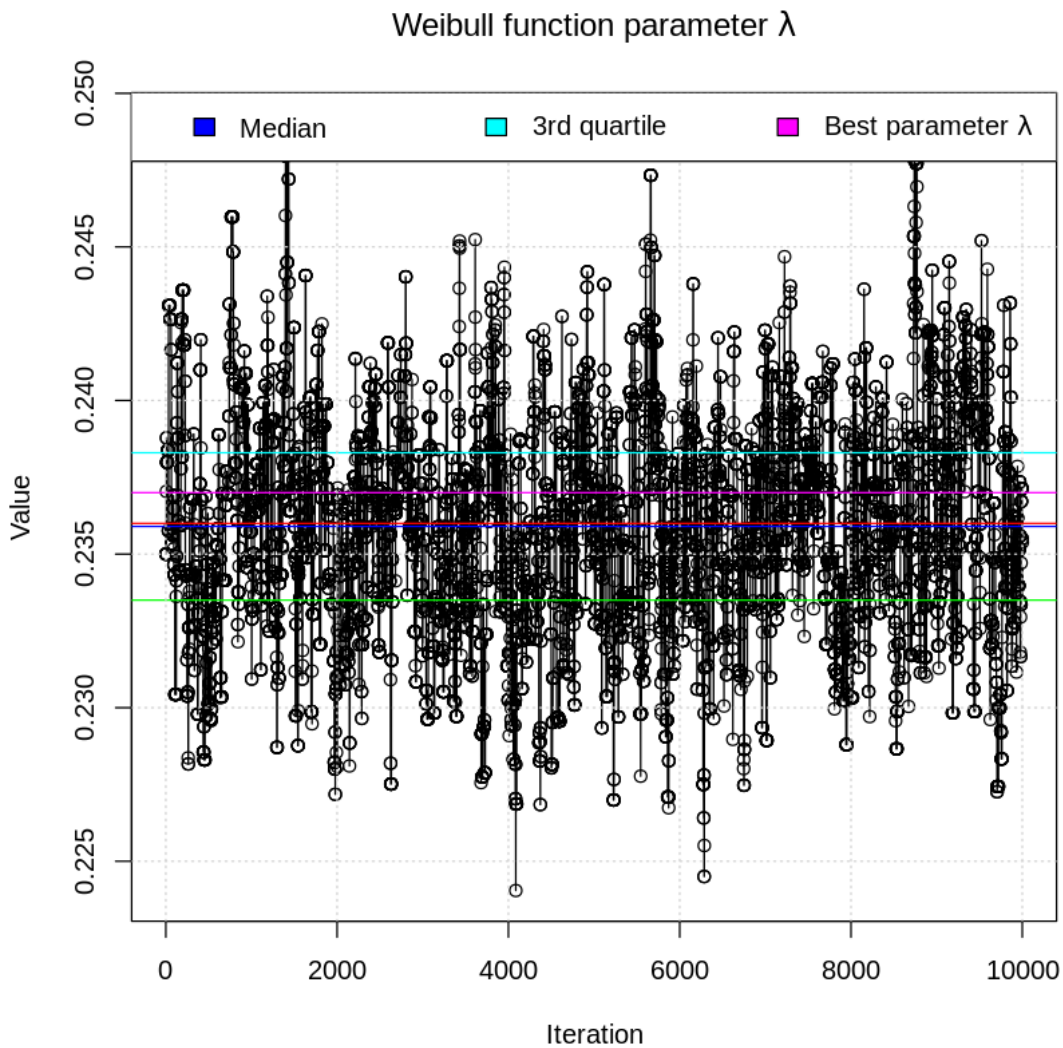
	Statistics <chr>	Values <dbl>
muestras	n	1.000e+04
minimos	Minimum	2.240e-01
cuantiles1	1st. Quartile	2.335e-01
medianas	Median	2.359e-01
medias	Mean	2.360e-01
cuantiles3	3rd. Quartile	2.383e-01
maximos	Maximum	2.490e-01
rangos	Rank	2.500e-02
rangosInt	Interquartile Rank	4.700e-03
varianzas	Variance	0.000e+00
desvs	Standard Deviation	3.500e-03
CVs	Variation Coeff.	1.500e-02
simetrias	Skewness	1.202e-01
curtosiss	Kurtosis	1.225e-01

```
[1] "The Weibull density function parameter lambda 0.237"
```



[70]:
```
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)
```



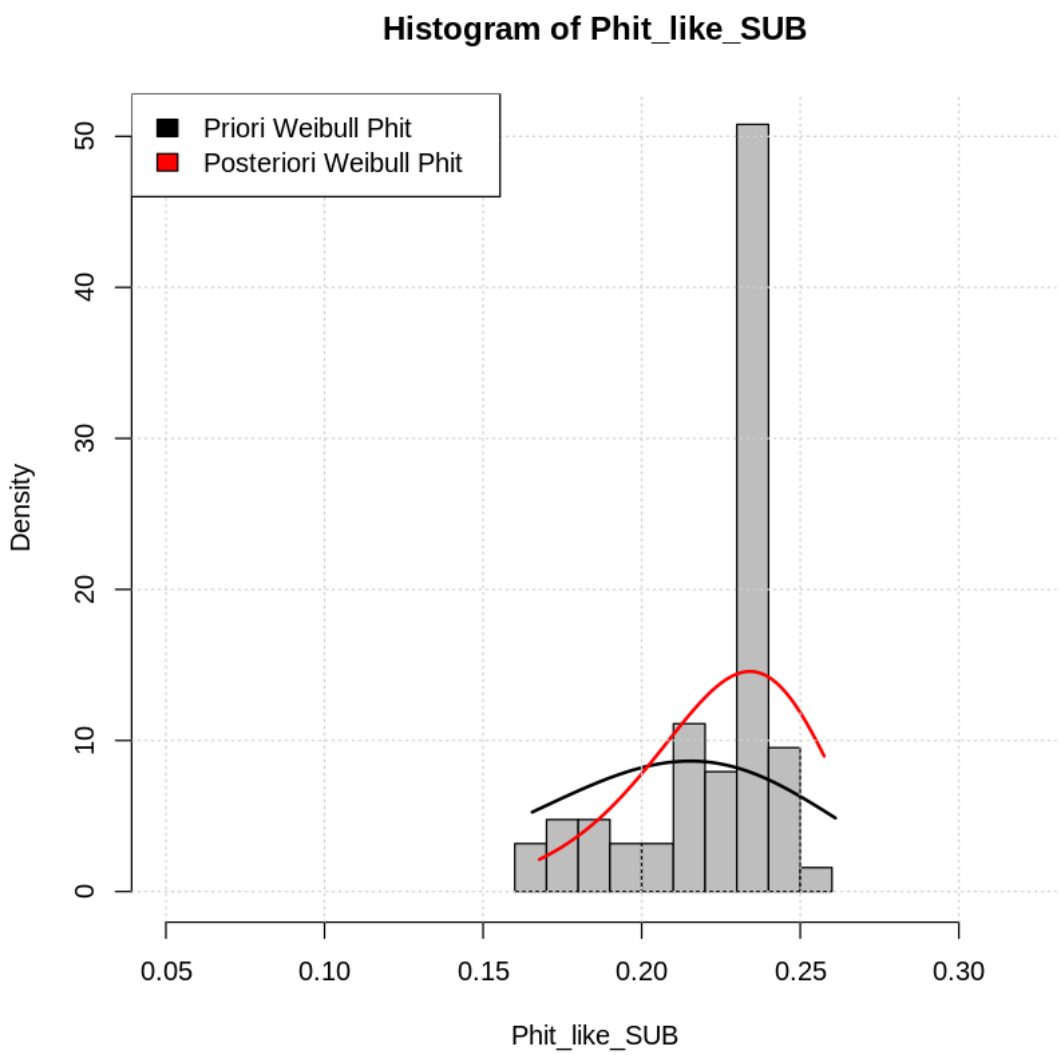
[71]:
```
cop_frank_post_5TRA <- mvdc(frankCopula(theta_best_post_5TRA), margins =␣
↪c("lnorm","weibull"),
                        paramMargins = list(list(meanlog = mu_best_post_5TRA,␣
↪sdlog = sigma_best_post_5TRA),
                                            list(shape = alpha_best_post_5TRA,␣
↪scale = lambda_best_post_5TRA)))

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

In a second iteration, the posterior parameter values from the first search were used as priors, and the new information was the subsampled logs. This time, all five parameters were included, and the acceptance rate was 21.5%. As seen in Table **??**, sixth column, most parameters changed only slightly—except for the dependence parameter θ, which shifted from -12.6 to -21. While this is a substantial change, it reflects the increase in dependence documented in Table **??**. It is worth noting that the Frank copula parameter can range between -100 and 100 (Joe, 2014).

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

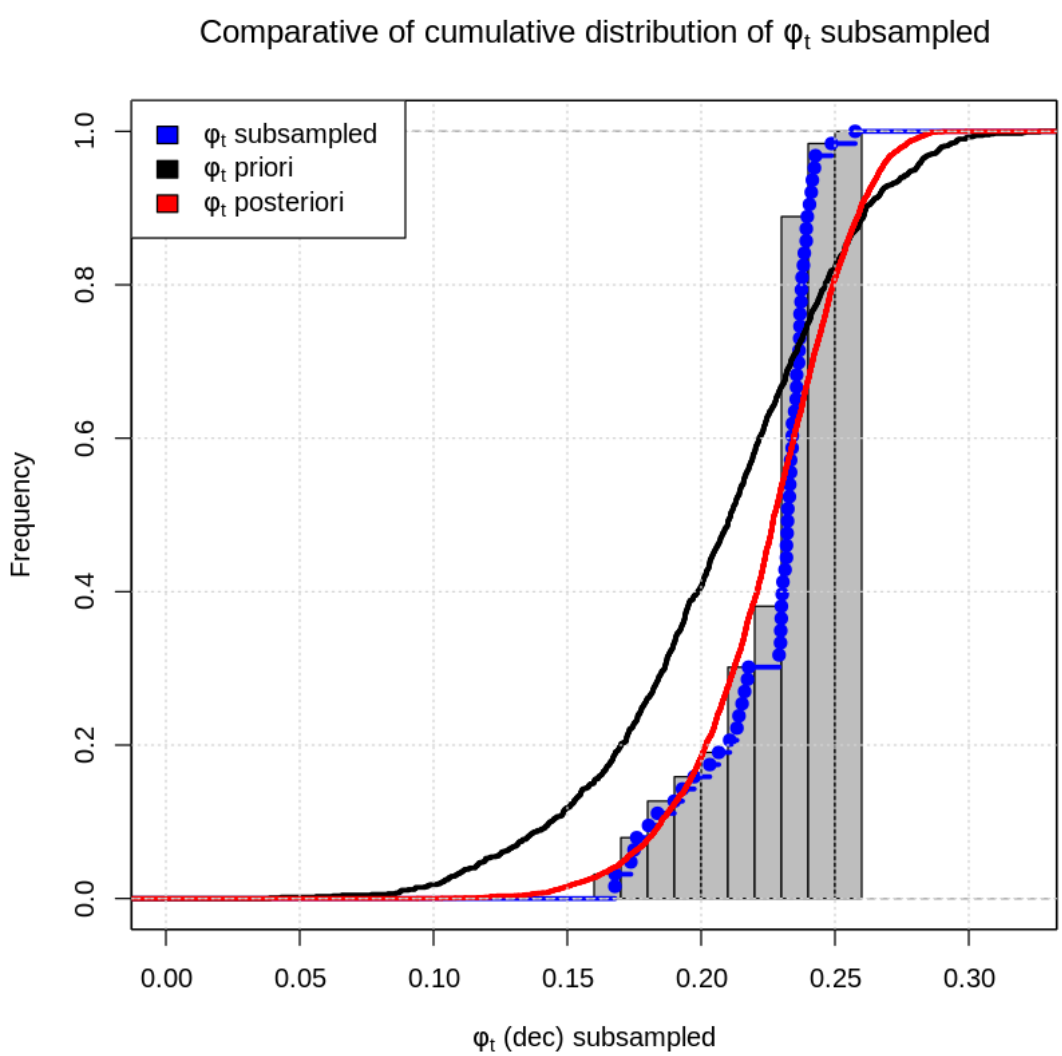
[72]:
```
den.x <- seq(min(Phit_like),max(Phit_like),length=3696)
den.y_prior <- dweibull(den.x,shape = 5.1646292, scale = 0.2244496)

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)
```

[73]:
```
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"))
```



[74]:
```
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)
plot(ecdf(Xf[,2]), col=("black"),xlim=c(0, 0.
↪32),lwd=3,xaxt='n',yaxt='n',ann=FALSE) #IFME
par(new=TRUE)
```

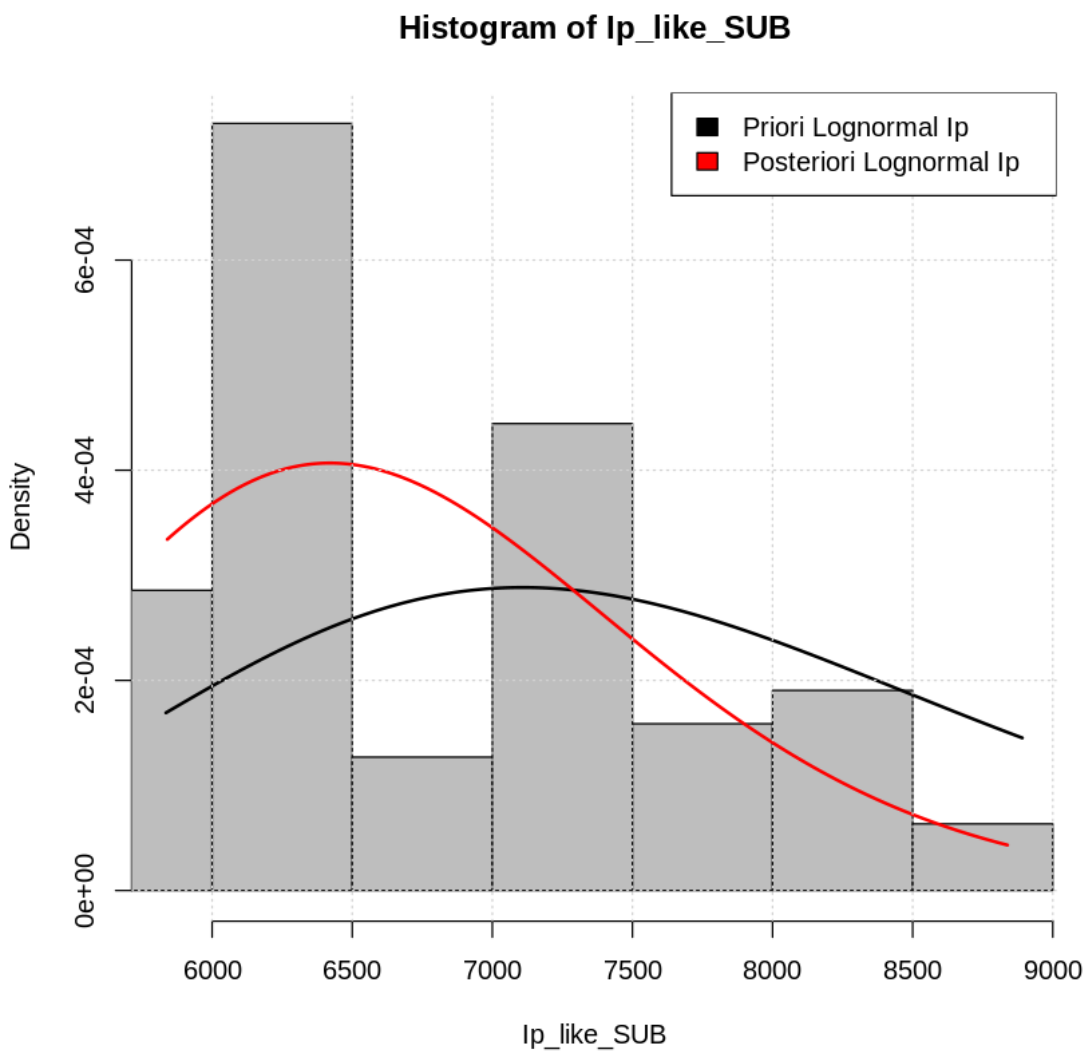
```
plot(ecdf(cop_frank_post_ran_5TRA[,2]), col=("red"),xlim=c(0, 0.
↪32),lwd=3,xaxt='n',yaxt='n',ann=FALSE) #Clayton Itau
grid()
legend(x = "topleft", legend = c(expression(paste(phi[t]," subsampled")),
                                 expression(paste(phi[t]," priori")),
                          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



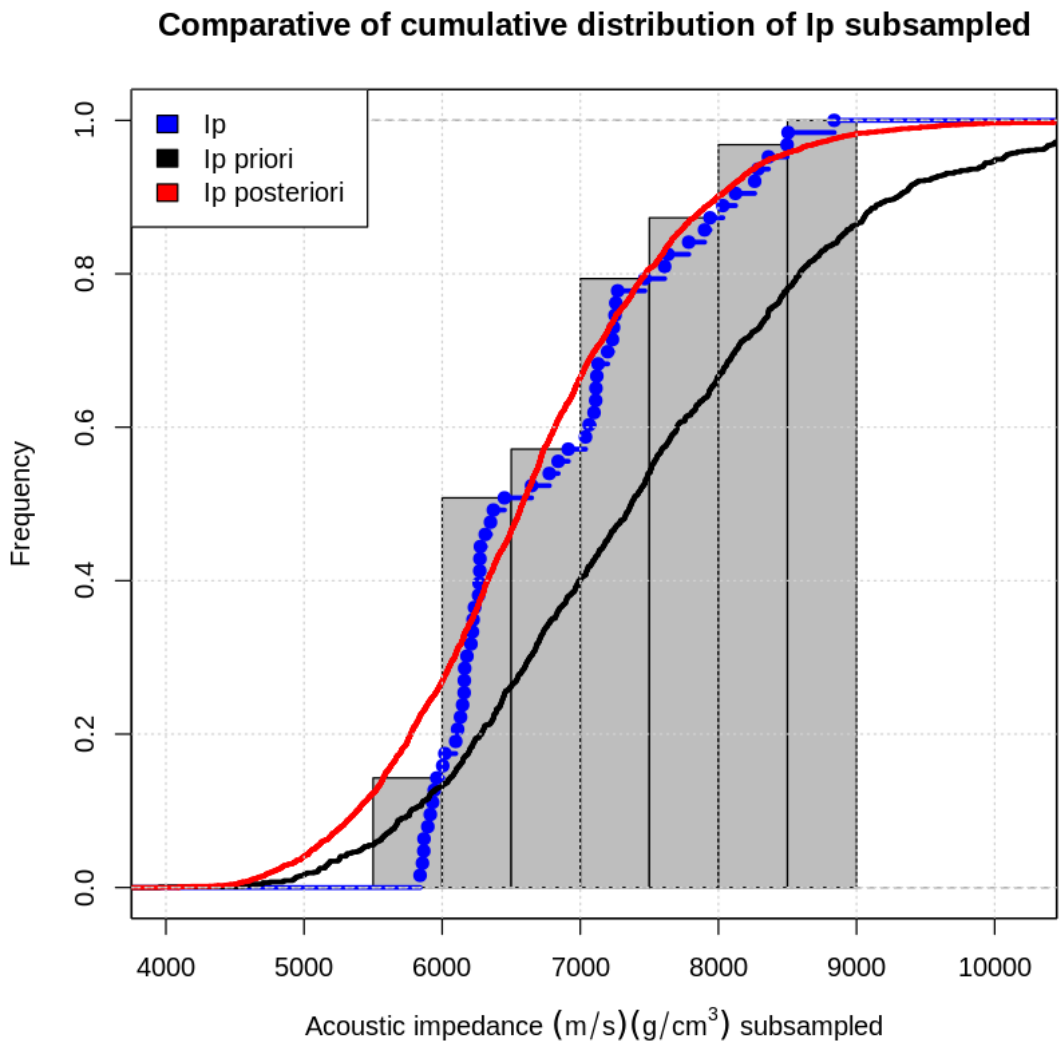
[75]:
```
den.x <- seq(min(Ip_like),max(Ip_like),length=3696)
den.y_prior <- dlnorm(den.x,meanlog = 8.905543, sdlog = 0.191057)

den.x_post_5TRA <- seq(min(Ip_like_SUB),max(Ip_like_SUB),length=3696)
```

```
den.y_post_5TRA <- dlnorm(den.x_post_5TRA,meanlog = mu_best_post_5TRA, sdlog =␣
↪sigma_best_post_5TRA)
```

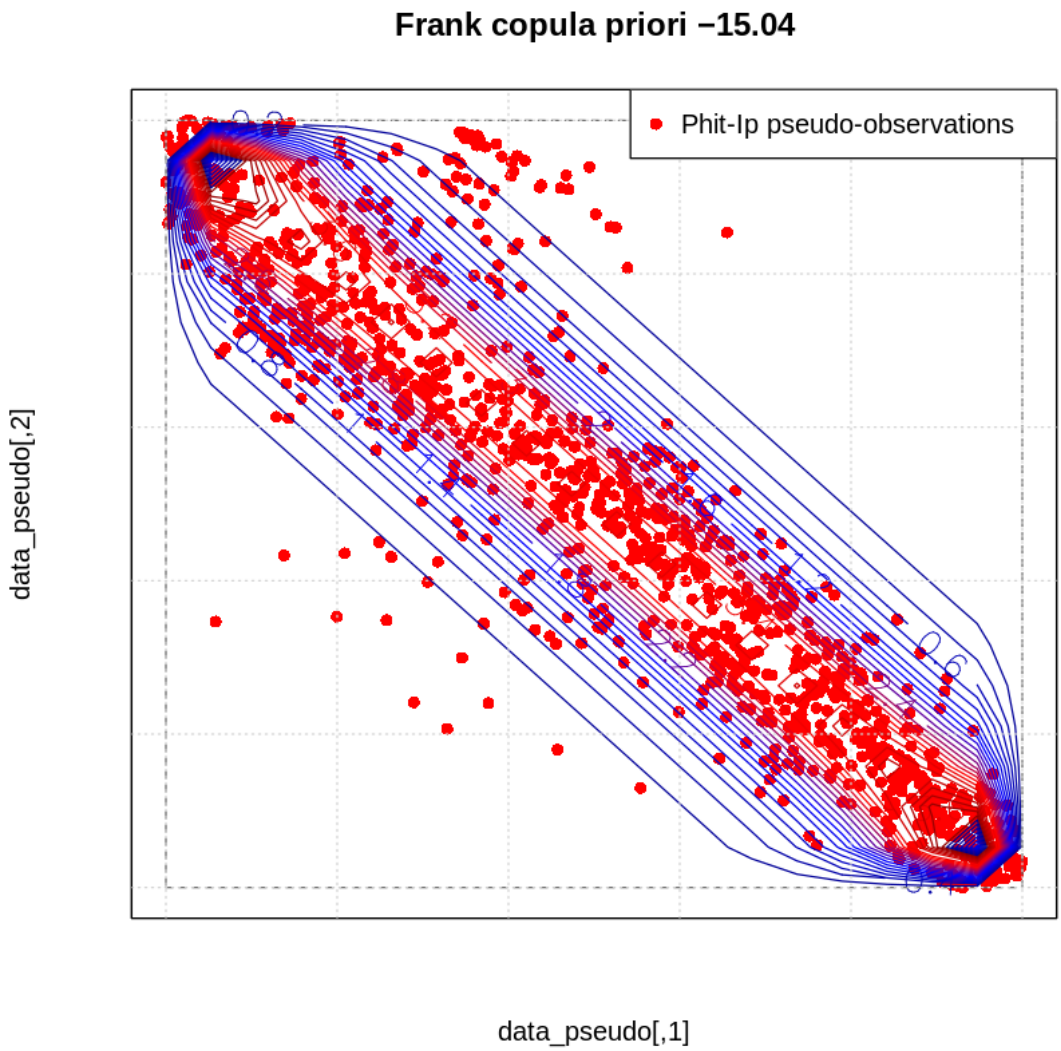
[76]:
```
hist(Ip_like_SUB, breaks=nsp, col="gray", probability=TRUE,xlim =␣
↪c(min(Ip_like),max(Ip_like)))
lines(den.x, den.y_prior, col="black", lwd=2,xlim = c(min(Ip_like),max(Ip_like)))
lines(den.x_post_5TRA, den.y_post_5TRA, col="red", lwd=2,xlim =␣
↪c(min(Ip_like),max(Ip_like)))
grid()
legend(x = "topright", legend = c("Priori Lognormal Ip", "Posteriori Lognormal␣
↪Ip"),
       fill = c("black","red"))
```



[77]:
```
hcum <- h <- hist(Ip_like_SUB, 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_SUB), 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"))
```

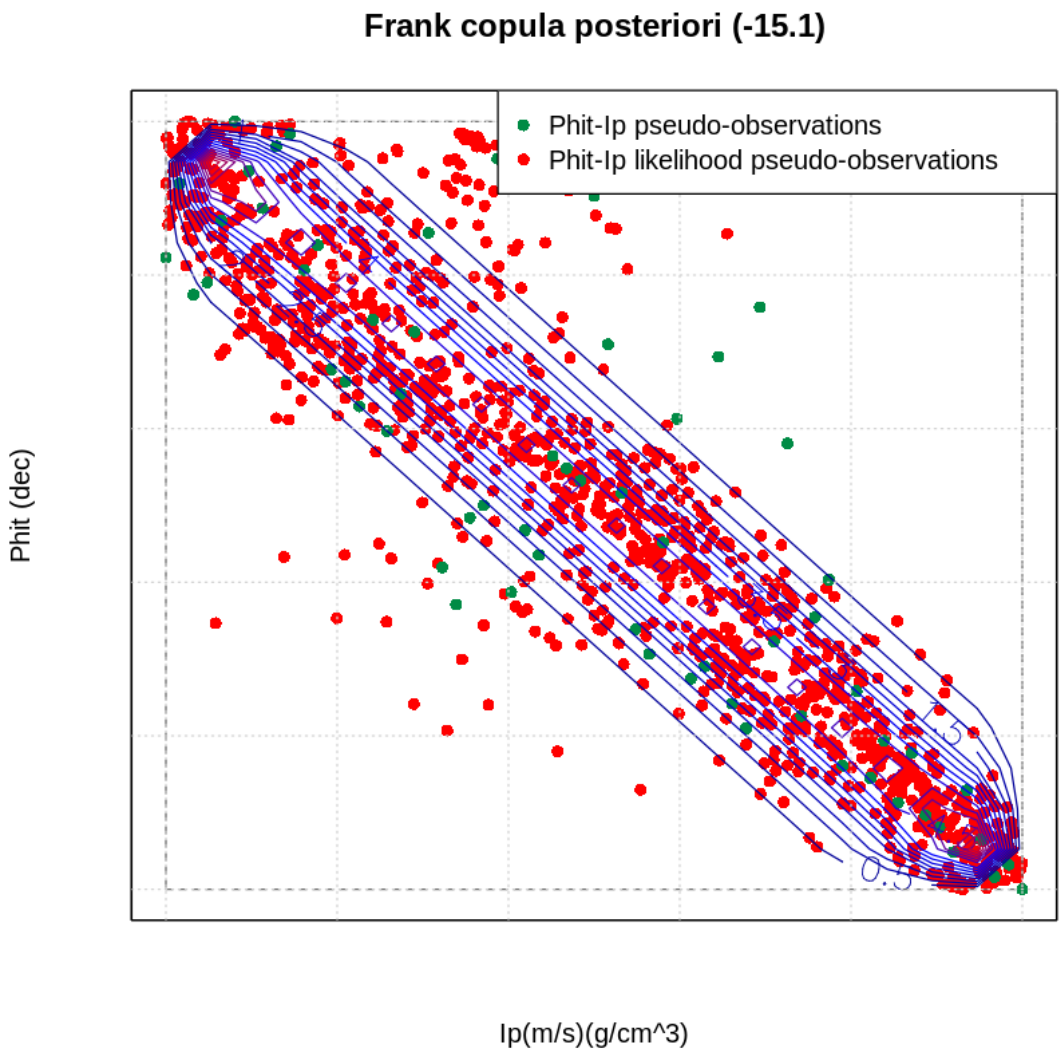


[78]:
```
plot(data_pseudo, pch=16, col=("red"),xaxt='n',yaxt='n',main = ("Frank copula␣
↪priori 15.04"))
par(new=TRUE)
contour(frankCopula(-15.04), dCopula,n=20,nlevels=25, xlab=(""), ylab=("")␣
↪,labcex=1.4,
        col=cols,xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend = c("Phit-Ip pseudo-observations"), col = c(␣
↪"red"),pch=16)
```



[79]:
```
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"),
       col = c("springgreen4","red"),pch=c(16,16))
```



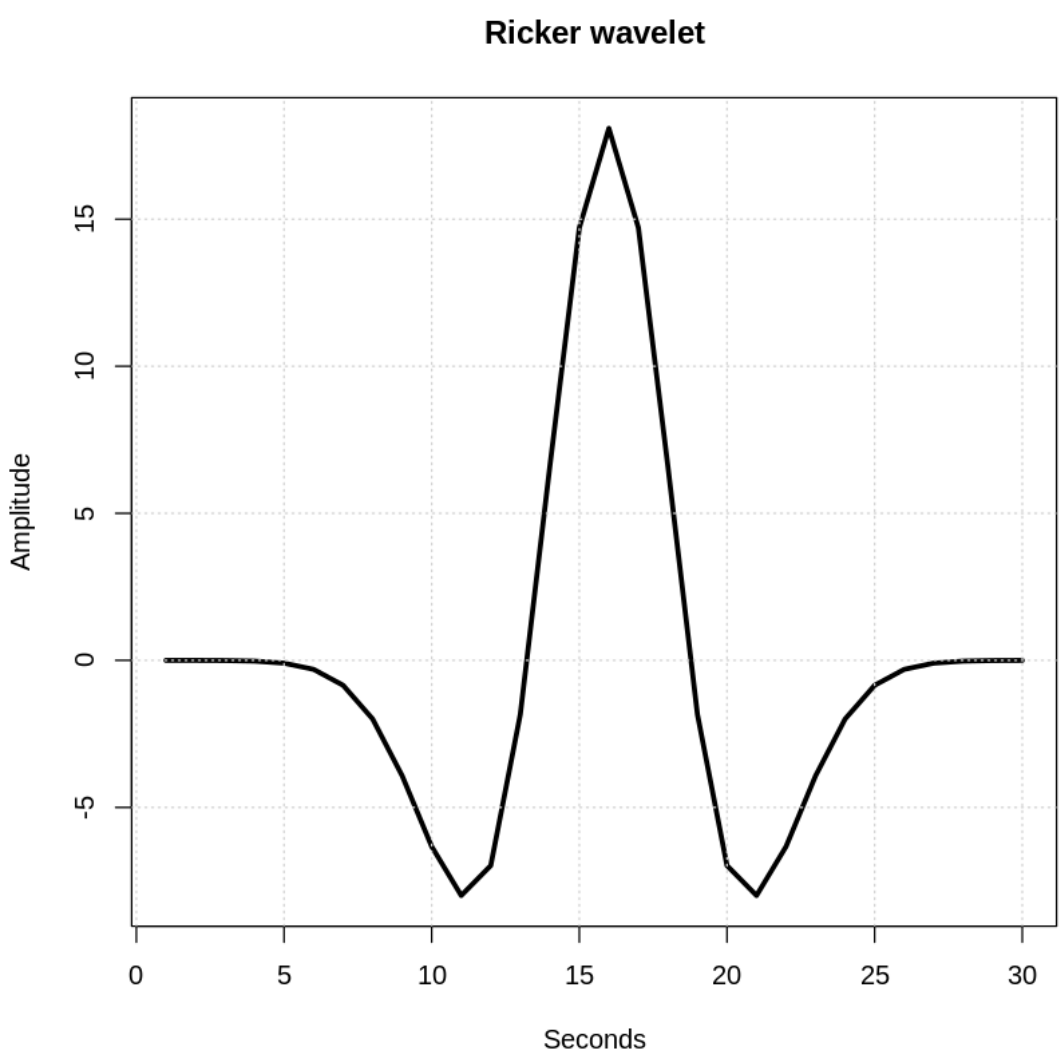
8.6 Seismic Inversion Using the Nearest Borehole Trace

The objective of geostatistical seismic inversion is to solve the inverse problem by simulating elastic properties that reproduce the observed seismic trace under a given forward model. In this study, the convolutional model is used, requiring two components: a wavelet and a reflectivity series. The wavelet is the 20.16 Hz Ricker wavelet, and the reflectivity series is computed from acoustic impedance as described in Section 7.1. Multiple realizations of acoustic impedance are generated to minimize the squared error between synthetic and observed seismic traces. The seismic trace closest to the well, located at 18.5 meters, is used for validation.

[80]:
```
semi_prior_sis<-EyeModel(CoorX=time_like_SUB, CoorY=YCoord1, Variable=var_sis,␣
↪Direccion=Direccion1, Tol=Tol1, NIntervalos=NIntervalos1,
                    Lags=Lags1, Npares=1, Modelo=Modelo1, Nugget=Nugget1,␣
↪SillYNugget=SillYNugget1,
                    Alcance=Alcance1,graph=FALSE,
                    MainTitle="Variograma",xlab="time(m)")
```

[81]:
```
dt <- 0.004 # sampling interval in seconds
freq <- 20.41  #frequency
nlen <- 31  # number of samples
G <- genrick(freq, dt, nlen)
```

[82]:
```
plot(G,lwd=3,type="l",xlab="Seconds",ylab="Amplitude",main="Ricker wavelet")
grid()
```



[83]:
```
semi_prior<-EyeModel(CoorX, CoorY, Variable=Ip_like_SUB, Direccion=Direccion,␣
↪Tol=Tol, NIntervalos=NIntervalos,
                     Lags=Lags, Npares=1, Modelo=Modelo, Nugget=Nugget,␣
↪SillYNugget=SillYNugget,
                     Alcance=Alcance,graph=FALSE,
                     MainTitle="Variograma",xlab="Distancia(m)")
```

[3]:
```
semi_priorp<-EyeModel(time_like_SUB, YCoord,
                                  Phit_like_SUB, 0, 90, N_lags_SUB,␣
↪lag_value_SUB, 1,
                                  vario_model_SUB, nugget_SUB,␣
↪sill_with_nugget_SUB, range_SUB,graph=FALSE,
                                  expression(paste(" Total porosity ",phi[t],
                                                   "(dec) Manual Variogram Model␣
↪fitting")),xlab="time(ms)")
semi_priorp<-append(semi_priorp,0.00033)
```

```
variog: computing omnidirectional variogram
```

[87]:
```
mfc <- 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)))
```

Simulated annealing is used to generate realizations of acoustic impedance and total porosity. The initial image is the subsampled log data, and the search function employs the dependence model from Section **??**.

[88]:
```
Var_dim<-length(var_ob)
x<-rMvdc(Var_dim, mvdc =mfc)
x0<-cbind(Ip_like_SUB,Phit_like_SUB)
lower_ip=Ip_like_SUB_Stat[2,2]
upper_ip=Ip_like_SUB_Stat[7,2]
lower_phit=Phit_like_SUB_Stat[2,2]
upper_phit=Phit_like_SUB_Stat[7,2]
```

The objective functions are the variogram model of Figure **??**A and the root mean square error (RMSE) between the synthetic and real seismic traces. Once the best-fitting acoustic impedance realization is found, it is used as a conditional variable to simulate total porosity using the same dependence model. The variogram model for porosity (Figure~**??**B) is used as the objective function.

[89]:
```
hchange<-function(x0)
{
  i=0
  sus<-round(runif(1,1,Var_dim),0)
```

```
  while (i==0) {
    x<-round(rMvdc(1, mvdc =mfc),4)
    if(x[,1]>=lower_ip & x[,1]<=upper_ip & x[,2]>=lower_phit &␣
↪x[,2]<=upper_phit){
      i=1}
    else{
      i=0}
  }
  x0[sus,]<-x
  return(x0)
}
```

[194]:
```
Func_syn  <- function(var_sim) {
  DF_var_sim<-data.frame(var_sim[,1])
  CRi<-matrix(0, Var_dim, 1)
  for (i in 1:(Var_dim-1)) {
    CRi[i,1] <- (DF_var_sim[(i+1),1]-DF_var_sim[i,1])/
↪(DF_var_sim[(i+1),1]+DF_var_sim[i,1])
  }
  synthetic<-sigconv(CRi,G)
  synthetic<-(synthetic*2.4e4)
  #----------------Función O1 variograma␣
↪Ip-------------------------------------------#
  Var_emp <- Variograma(CoorX, CoorY, Variable=var_sim[,1], Direccion=Direccion,␣
↪Tol=Tol, NIntervalos=NIntervalos,
                        Lags=Lags, Npares=1,graph=FALSE ,␣
↪MainTitle="Variograma", xlab="Distancia", ylab = "Semivarianza")
  semi_sim = Var_emp[,3]
  Fun1<-(sum(((semi_sim-semi_prior)/semi_prior)^2))/21.40191
␣
↪#----------------------------------------simulation-----------------------------------#
  #Var_emp1 <- Variograma(CoorX=time_like_SUB, CoorY=YCoord1,␣
↪Variable=synthetic, Direccion=Direccion1, Tol=Tol1, NIntervalos=NIntervalos1,
  #                       Lags=Lags1, Npares=1,graph=FALSE ,␣
↪MainTitle="Variograma", xlab="Distancia", ylab = "Semivarianza")
  #semi_sim1 = Var_emp[,3]
  #----------------Función O1-------------------------------------------#
  #Fun2=sum(((semi_sim1-semi_prior_sis)/semi_prior_sis)**2)
  #----------------Función O2-------------------------------------------#
  Fun3=((rmse(var_sis,synthetic))/neartrace_Stat[8,2])/0.2987836
  #return(Fun1+Fun2+Fun3)
  return(0.1*Fun1+0.9*Fun3)

}
```

For all simulations, simulated annealing is initialized with a temperature of 0.001, a cooling parameter of 0.22, and stops at a convergence value of 0.0001 or after 20,000 iterations.

[195]:
```
algo=list(nS=10,initT=0.001,alpha=0.22,nI=20000,x0=x0,neighbour=hchange,
          printDetail=2000,storeF=TRUE,OF.target=0.01)
```

[196]:
```
system.time(s4<-SAopt(OF=Func_syn, algo = algo))
```

```
Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999999
Best solution (iteration 2000/20000): 0.2210986
Best solution (iteration 4000/20000): 0.1907199
Best solution (iteration 6000/20000): 0.1723611
Best solution (iteration 8000/20000): 0.167979
Best solution (iteration 10000/20000): 0.159993
Best solution (iteration 12000/20000): 0.1541784
Best solution (iteration 14000/20000): 0.1510771
Best solution (iteration 16000/20000): 0.1470233
Best solution (iteration 18000/20000): 0.1431847
Best solution (iteration 20000/20000): 0.1335844
Finished.
Best solution overall: 0.1335844

   user  system elapsed
140.395  88.830 268.695
```

[197]:
```
DF_var_f<-data.frame(s4$xbest[,1])
CRf<-matrix(0, Var_dim, 1)
for (i in 1:(Var_dim-1)) {
  CRf[i,1] <- (DF_var_f[(i+1),1]-DF_var_f[i,1])/(DF_var_f[(i+1),1]+DF_var_f[i,1])
}
```

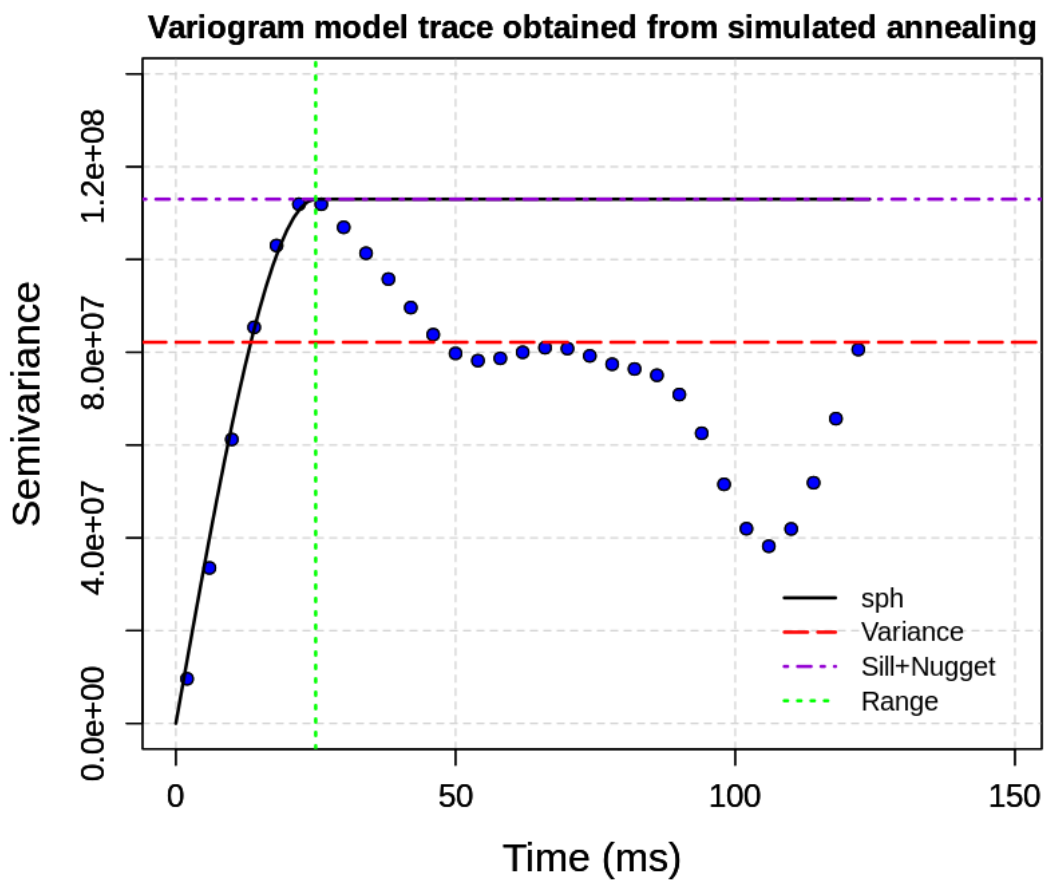
[198]:
```
synthetic_f<-sigconv(CRf,G)
synthetic_f<-(synthetic_f*2.4e4)
synthetic_f<-data.frame(synthetic_f)
```

[199]:
```
EyeModel(CoorX=time_like_SUB, CoorY=YCoord1, synthetic_f,
                                  Direccion=Direccion1, Tol=Tol1,␣
 ↪NIntervalos=NIntervalos1,
                                  Lags=Lags1, 1,
                                  Modelo=Modelo1, Nugget1, SillYNugget1,␣
 ↪Alcance1, graph=TRUE,
                                  MainTitle = "Variogram model trace obtained␣
 ↪from simulated annealing",
                                  xlab="Time (ms)")

EyeModel(CoorX=time_like_SUB, CoorY=YCoord1, var_sis,
```

```
                                        Direccion=Direccion1, Tol=Tol1,␣
↪NIntervalos=NIntervalos1,
                                        Lags=Lags1, 1,
                                        Modelo=Modelo1, Nugget1, SillYNugget1,␣
↪Alcance1, graph=TRUE,
                                        MainTitle="Modelo de semivariogram ajustado␣
↪para la traza",
                                        xlab="time")
```

A matrix: 1 × 3 of type dbl

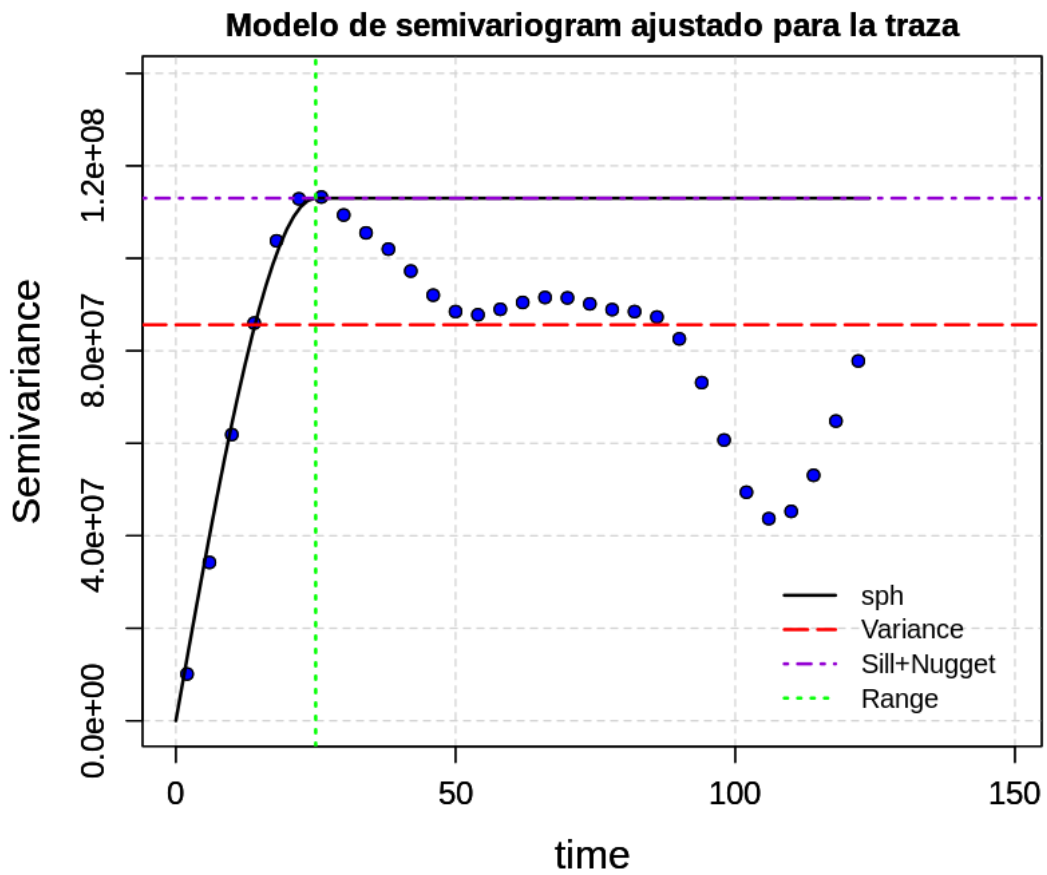
	Nugget	Meseta+Nugget	Alcance
sph	700	1.13e+08	25



Model	Nugget	Sill+Nugget	Range	MSE
sph	700.0000	113000000.0000	25.0045	45866073541899

A matrix: 1 × 3 of type dbl

	Nugget	Meseta+Nugget	Alcance
sph	700	1.13e+08	25



Model	Nugget	Sill+Nugget	Range	MSE
sph	700.0000	113000000.0000	25.0000	6238863303510702

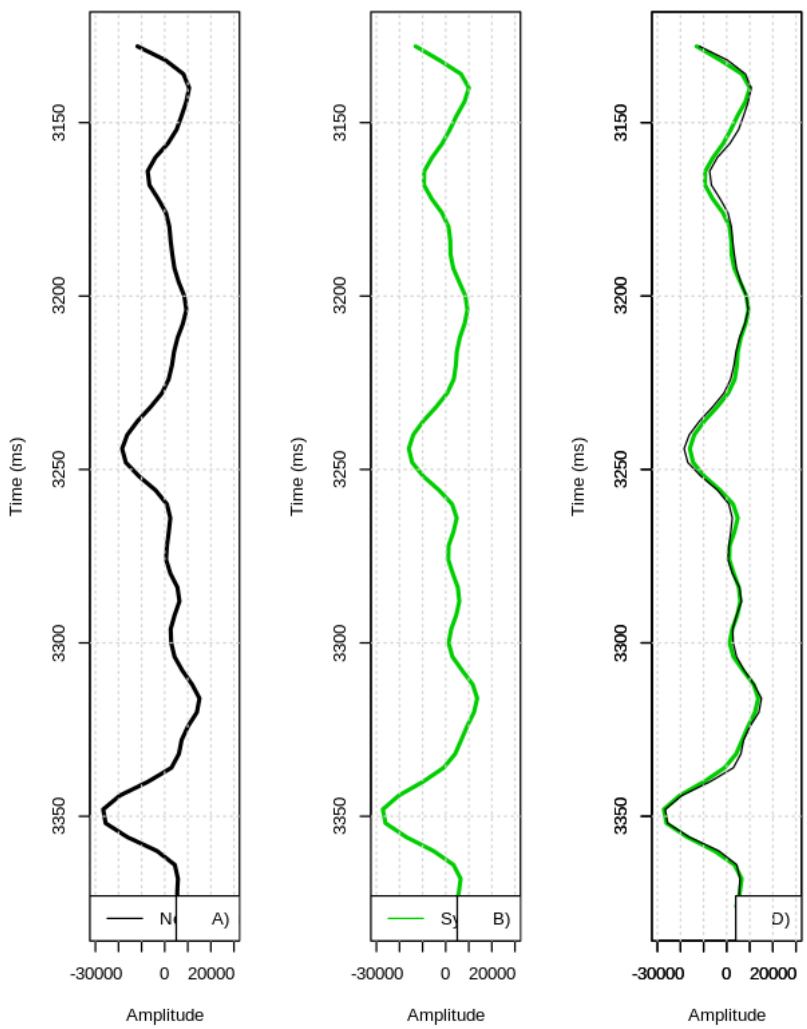
[200]:
```
var_sim4t_Stat<-Estadisticas(synthetic_f[,1])


Comparative_4t_Stat<-cbind(neartrace_Stat,var_sim4t_Stat[,2])
#Comparative_post_Stat<-Val_Estadisticos(Comparative_post)
colnames(Comparative_4t_Stat)<-c("Statistics"," nearest trace",
                                 " synthetic trace SAopt")
Comparative_4t_Stat
```

A data.frame: 14 × 3

	Statistics <chr>	nearest trace <dbl>	synthetic trace SAopt <dbl>
muestras	n	63.0000	63.0000
minimos	Minimum	-26641.8281	-27232.7247
cuantiles1	1st. Quartile	-3642.2930	-4717.6634
medianas	Median	2596.1748	2779.1219
medias	Mean	258.0798	-187.8174
cuantiles3	3rd. Quartile	5913.7070	5788.8181
maximos	Maximum	15073.0547	13657.6984
rangos	Rank	41714.8828	40890.4231
rangosInt	Interquartile Rank	9556.0000	10506.4814
varianzas	Variance	85621835.6709	82159627.6589
desvs	Standard Deviation	9253.2068	9064.1948
CVs	Variation Coeff.	35.8541	-48.2607
simetrias	Skewness	-1.1455	-1.1089
curtosiss	Kurtosis	3.8210	1.0154

[201]:
```
par(mfrow = c(1,4))

plot(neartrace,time_like_SUB,xlab = "Amplitude",ylab="Time (ms)",
     ylim=rev(range(time_like_SUB)),xlim=c(-30000, 30000),type = "l",lwd=2.
 ↪5,col=("black"))
grid()
legend(x = "bottomleft", legend = c("Nearest trace"), col = c("black"),lty=c(1))
legend(x="bottomright",legend="A)")

plot(synthetic_f[,1],time_like_SUB,xlab = "Amplitude",ylab="Time (ms)",
     ylim=rev(range(time_like_SUB)),xlim=c(-30000, 30000),type = "l",lwd=2.
 ↪5,col=("green3"))
grid()
legend(x = "bottomleft", legend = c("Synthetic SA trace"), col = c(␣
 ↪"green3"),lty=c(1))
legend(x="bottomright",legend="B)")

#plot(synthetic_nt,time_like_SUB,xlab = "Amplitude",ylab="Time (ms)",
#     ylim=rev(range(time_like_SUB)),xlim=c(-30000, 30000),type =␣
 ↪"l",lwd=1,col=("blue"))
#grid()
#legend(x = "bottomleft", legend = c("Synthetic SS trace"), col =␣
 ↪c("blue"),lty=c(1))
#legend(x="bottomright",legend="C)")

plot(synthetic_f[,1],time_like_SUB,xlab = "",ylab="",
     ylim=rev(range(time_like_SUB)),xlim=c(-30000, 30000),type = "l",lwd=2.
 ↪5,col=("green3"))
par(new=TRUE)
```

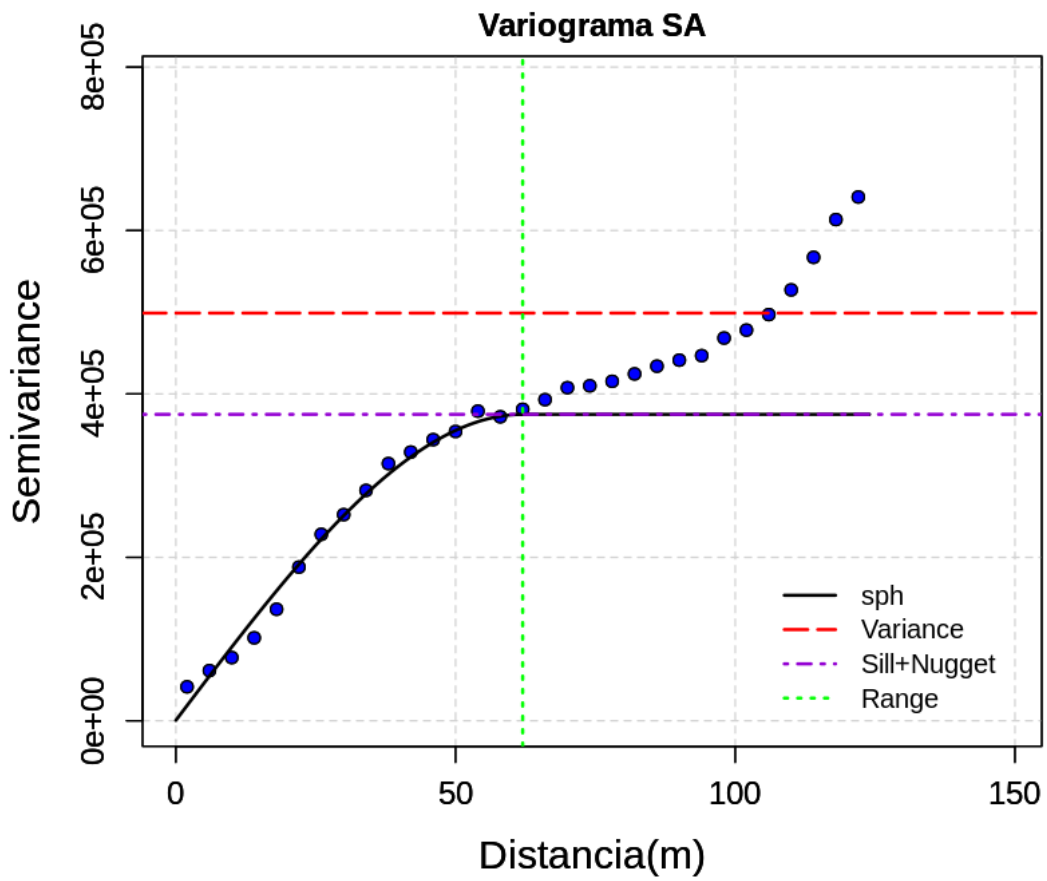
```r
plot(neartrace,time_like_SUB,xlab = "Amplitude",ylab="Time␣
↪(ms)",ylim=rev(range(time_like_SUB)),
     xlim=c(-30000, 30000),type = "l",lwd=1,col=("black"))
par(new=TRUE)
#plot(synthetic_nt,time_like_SUB,xlab = "Amplitude",ylab="Time (ms)",
#     ylim=rev(range(time_like_SUB)),xlim=c(-30000, 30000),type =␣
↪"l",lwd=1,col=("blue"))
#legend(x = "bottomleft", legend = c("trace","synthetic SA","synthetic SS"),␣
↪fill = c("black","green3","blue"))
#legend(x="bottomright",legend="C)")
legend(x="bottomright",legend="D)")
grid()
```



[202]:
```
EyeModel(CoorX, CoorY, Variable=s4$xbest[,1], Direccion=Direccion, Tol=Tol,␣
↪NIntervalos=NIntervalos,
                    Lags=Lags, Npares=1, Modelo=Modelo, Nugget=Nugget,␣
↪SillYNugget=SillYNugget,
                    Alcance=Alcance,graph=TRUE,
                    MainTitle="Variograma SA",xlab="Distancia(m)")

EyeModel(CoorX, CoorY, Variable=Ip_like_SUB, Direccion=Direccion, Tol=Tol,␣
↪NIntervalos=NIntervalos,
                    Lags=Lags, Npares=1, Modelo=Modelo, Nugget=Nugget,␣
↪SillYNugget=SillYNugget,
                    Alcance=Alcance,graph=TRUE,
                    MainTitle="Variograma prior",xlab="Distancia(m)")
```

A matrix: 1 × 3 of type dbl

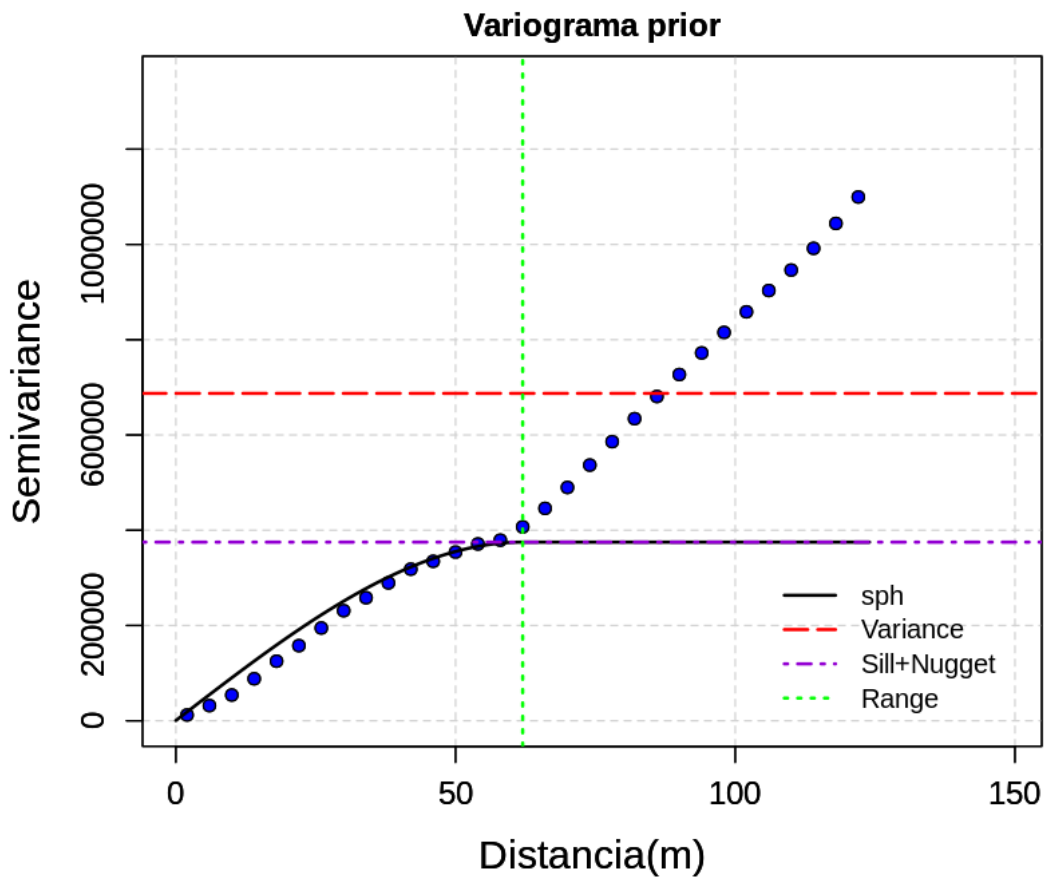
	Nugget	Meseta+Nugget	Alcance
sph	500	375000	62



Model	Nugget	Sill+Nugget	Range	MSE
sph	500.0000	375000.0000	62.0000243987115738.(	

A matrix: 1 × 3 of type dbl

	Nugget	Meseta+Nugget	Alcance
sph	500	375000	62



Model	Nugget	Sill+Nugget	Range	MSE
sph	500.0000	375000.0000	62.0000	2925023074482.

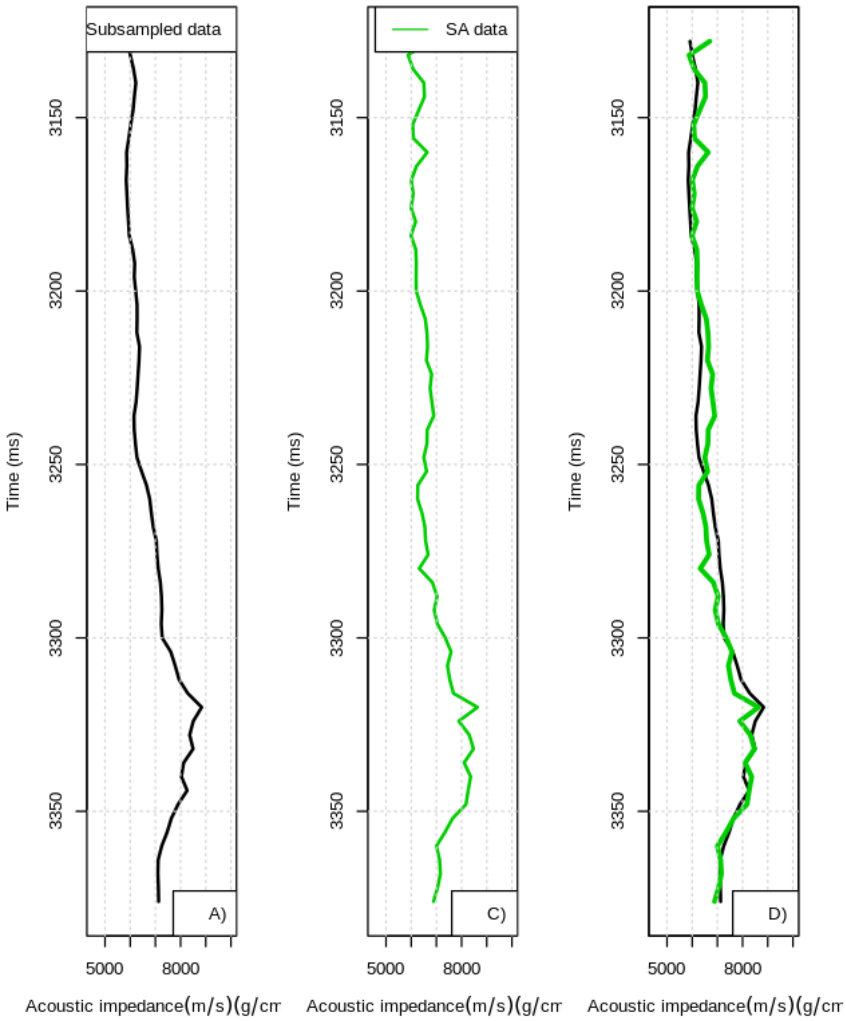
[208]:
```
var_sim4_Stat<-Estadisticas(s4$xbest[,1])

Comparative_4_Stat<-cbind(Ip_like_SUB_Stat,var_sim4_Stat[,2])
#Comparative_post_Stat<-Val_Estadisticos(Comparative_post)
colnames(Comparative_4_Stat)<-c("Statistics"," Acoustic impedance subsampled",
                                " Acoustic impedance SAopt")
Comparative_4_Stat
```

A data.frame: 14 × 3

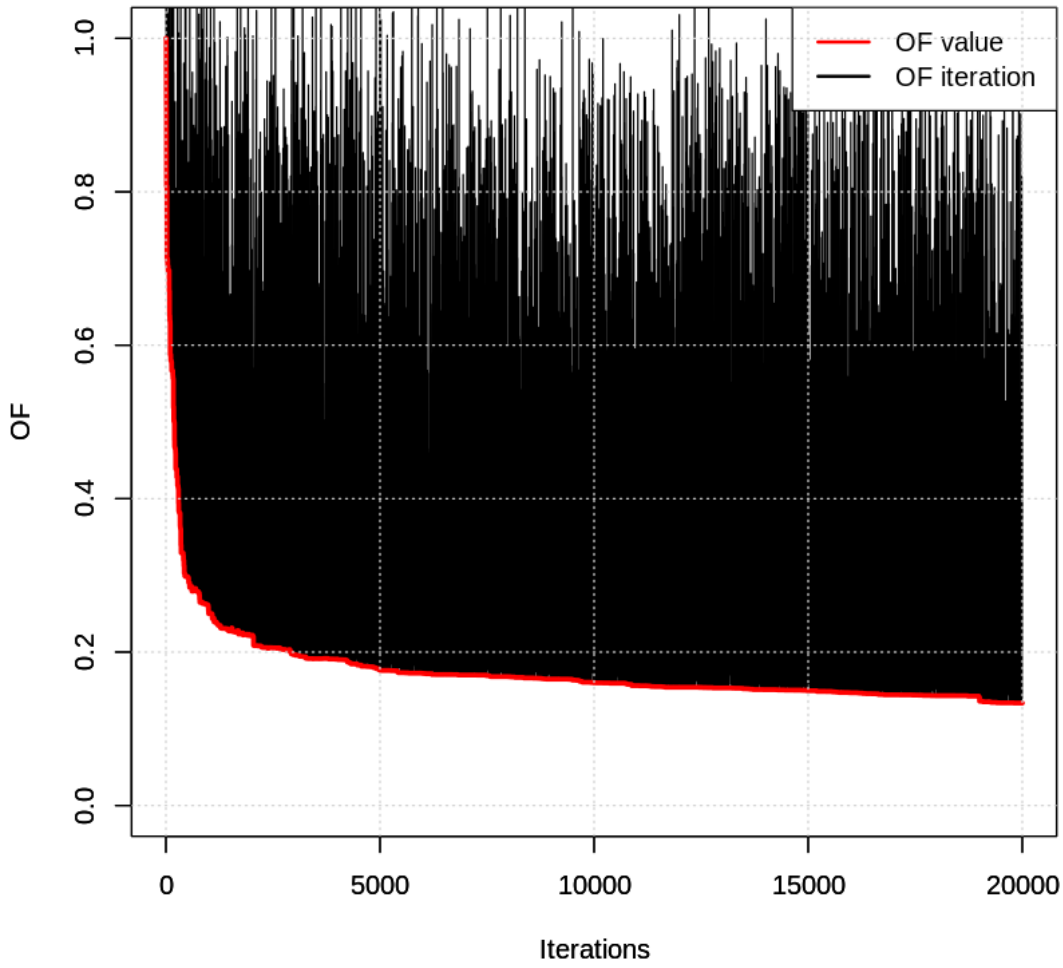
	Statistics <chr>	Acoustic impedance subsampled <dbl>	Acoustic impedance <dbl>
muestras	n	63.0000	63.0000
minimos	Minimum	5840.1241	5876.2758
cuantiles1	1st. Quartile	6159.7660	6276.9646
medianas	Median	6450.8228	6636.5546
medias	Mean	6817.8023	6846.3676
cuantiles3	3rd. Quartile	7254.2401	7144.2550
maximos	Maximum	8838.1850	8632.7570
rangos	Rank	2998.0609	2756.4812
rangosInt	Interquartile Rank	1094.4741	867.2904
varianzas	Variance	687258.8261	498938.5227
desvs	Standard Deviation	829.0108	706.3558
CVs	Variation Coeff.	0.1216	0.1032
simetrias	Skewness	0.6848	0.8917
curtosiss	Kurtosis	-0.6026	0.0610

[204]:
```
par(mfrow = c(1,4))

plot(Ip_like_SUB,time_like_SUB,xlim=c(4500, 10000),xlab =␣
 ↪expression(paste("Acoustic impedance", (m/s)(g/cm^3))),
     ylab="Time (ms)",ylim=rev(range(time_like_SUB)),type =␣
 ↪"l",lwd=2,col=("black"))
grid()
legend(x = "topright", legend = c("Subsampled data"), col = c("black"),lty=c(1))
legend(x="bottomright",legend="A)")

#plot(Ip_seismic,time_like_SUB,xlim=c(4500, 10000),xlab =␣
 ↪expression(paste("Acoustic impedance", (m/s)(g/cm^3))),
#     ylab="Time (ms)",ylim=rev(range(time_like_SUB)),type =␣
 ↪"l",lwd=2,col=("blue"))
#grid()
#legend(x = "topright", legend = c("Sparse spike data"), col =␣
 ↪c("blue"),lty=c(1))
#legend(x="bottomright",legend="B)")

plot(s4$xbest[,1],time_like_SUB,xlim=c(4500, 10000),xlab =␣
 ↪expression(paste("Acoustic impedance", (m/s)(g/cm^3))),
     ylab="Time (ms)",ylim=rev(range(time_like_SUB)),type =␣
 ↪"l",lwd=2,col=("green3"))
grid()
legend(x = "topright", legend = c("SA data"), col = c( "green3"),lty=c(1))
legend(x="bottomright",legend="C)")

plot(Ip_like_SUB,time_like_SUB,xlim=c(4500, 10000),xlab =␣
 ↪expression(paste("Acoustic impedance", (m/s)(g/cm^3))),
```

```
     ylab="Time (ms)",ylim=rev(range(time_like_SUB)),type =␣
 ↪"l",lwd=2,col=("black"))
par(new=TRUE)
#plot(Ip_seismic,time_like_SUB,xlim=c(4500, 10000),xlab =␣
 ↪expression(paste("Acoustic impedance", (m/s)(g/cm^3))),
#     ylab="Time (ms)",ylim=rev(range(time_like_SUB)),type =␣
 ↪"l",lwd=2,col=("blue"))
#par(new=TRUE)
plot(s4$xbest[,1],time_like_SUB,xlim=c(4500,␣
 ↪10000),ylim=rev(range(time_like_SUB)),type = "l",lwd=3,col=("green3"),
     xaxt='n',yaxt='n',ann=FALSE)
grid()
#legend(x = "topright", legend = c("Subsampled data","Sparse spike data",
#                                   "SA data"), fill = c("black","blue",␣
 ↪"green3"))
legend(x="bottomright",legend="D)")
```



[207]:
```
plot(s4$Fmat[ , 1L],type = "l",
     ylab = "OF", xlab = "Iterations",xlim=c(0,20000),ylim=c(0,1),col=("black"))
par(new=TRUE)
plot(s4$Fmat[ , 2L],type = "l",
     ylab = "OF", xlab =␣
 ↪"Iterations",xlim=c(0,20000),ylim=c(0,1),col=("red"),lwd=3)
legend(x = "topright", legend = c("OF value", "OF iteration"),
       lty = 1, lwd = 2, col = c("red","black"))
grid()
```



8.7 simulacion de la porosidad totl condicionada a la impedancia acustica

[11]:
```
hchange_cond_po<-function(x0)
{
  i=0
  sus<-round(runif(1,1,Var_dim),0)
  while (i==0) {
        Ip_sam<-s4$xbest[sus,1]
        Ip_cond <- plnorm(Ip_sam,meanlog = mu_best_post_5TRA, sdlog =␣
 ↪sigma_best_post_5TRA)
        Fr_cond<-cCopula(cbind(Ip_cond,runif(length(Ip_cond))),
                    copula = frankCopula(theta_best_post_5TRA), inverse = TRUE)
```

```
        samcond<-qweibull(Fr_cond[,2],shape = alpha_best_post_5TRA, scale =␣
↪lambda_best_post_5TRA)
    if(samcond>=lower_phit & samcond<=upper_phit){
      i=1}
    else{
      i=0}
  }
  x0[sus]<-samcond
  return(x0)
}
```

[12]:
```
Func_po  <- function(var_sim) {
 ␣
 ↪#----------------------------------------simulation-----------------------------------#
     Var_emp1 <- Variograma(CoorX, CoorY, Variable=var_sim, Direccion=Direccion,␣
 ↪Tol=Tol, NIntervalos=N_lags_SUB,
                                Lags=lag_value_SUB, Npares=1,graph=FALSE ,␣
 ↪MainTitle="Variograma", xlab="Distancia",
                            ylab = "Semivarianza")
  semi_sim1 = Var_emp1[,3]


  #----------------Función O1-----------------------------------------#
  Fun2=sum(((semi_sim1-semi_priorp)/semi_priorp)**2)/2.883024
  #----------------Función O2-----------------------------------------#
  #Fun3=(rmse(var_sis,synthetic)/neartrace_Stat[8,2])
  return(Fun2)
}
```

[31]:
```
algo_po=list(nS=10,initT=0.01,alpha=0.
↪22,nI=20000,x0=Phit_like_SUB,neighbour=hchange_cond_po,
         printDetail=2000,storeF=TRUE,OF.target=0.0001)
```

[32]:
```
system.time(s3<-SAopt(OF=Func_po, algo = algo_po))
```

```
Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 2000/20000): 0.03728396
Best solution (iteration 4000/20000): 0.01746571
Best solution (iteration 6000/20000): 0.00636807
Best solution (iteration 8000/20000): 0.003051821
Best solution (iteration 10000/20000): 0.001578936
Best solution (iteration 12000/20000): 0.001223112
Best solution (iteration 14000/20000): 0.001019607
```

```
Best solution (iteration 16000/20000): 0.0009097961
Best solution (iteration 18000/20000): 0.0008645148
Best solution (iteration 20000/20000): 0.0008176768
Finished.
Best solution overall: 0.0008176768
```

```
   user  system elapsed
 161.81   13.48  172.92
```

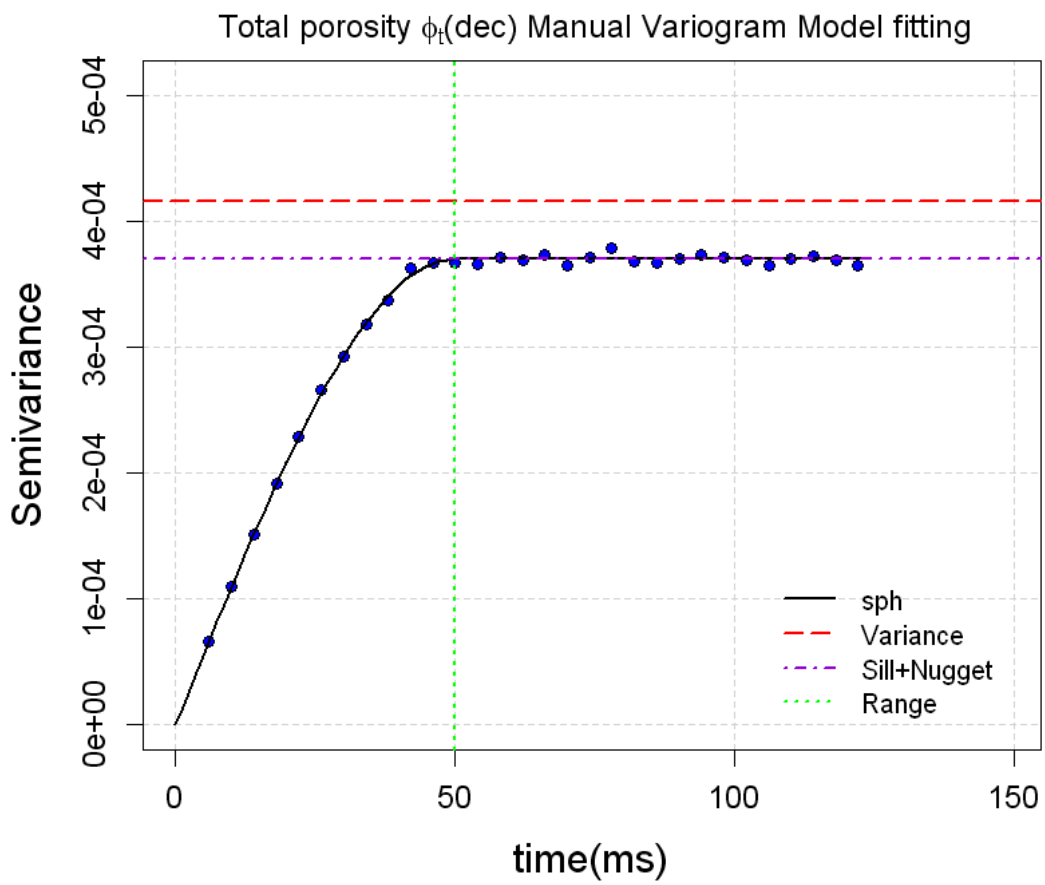
[33]:
```
EyeModel(time_like_SUB, YCoord,
                                    s3$xbest, 0, 90, N_lags_SUB, lag_value_SUB, 1,
                                    vario_model_SUB, nugget_SUB,␣
 ↪sill_with_nugget_SUB, range_SUB,graph=TRUE,
                                    expression(paste(" Total porosity ",phi[t],
                                                     "(dec) Manual Variogram Model␣
 ↪fitting")),xlab="time(ms)")

EyeModel(time_like_SUB, YCoord,
                                    Phit_like_SUB, 0, 90, N_lags_SUB,␣
 ↪lag_value_SUB, 1,
                                    vario_model_SUB, nugget_SUB,␣
 ↪sill_with_nugget_SUB, range_SUB,graph=TRUE,
                                    expression(paste(" Total porosity ",phi[t],
                                                     "(dec) Manual Variogram Model␣
 ↪fitting")),xlab="time(ms)")
```

```
variog: computing omnidirectional variogram
```

A matrix: 1 × 3 of type dbl

	Nugget	Meseta+Nugget	Alcance
sph	0	0.00037	50

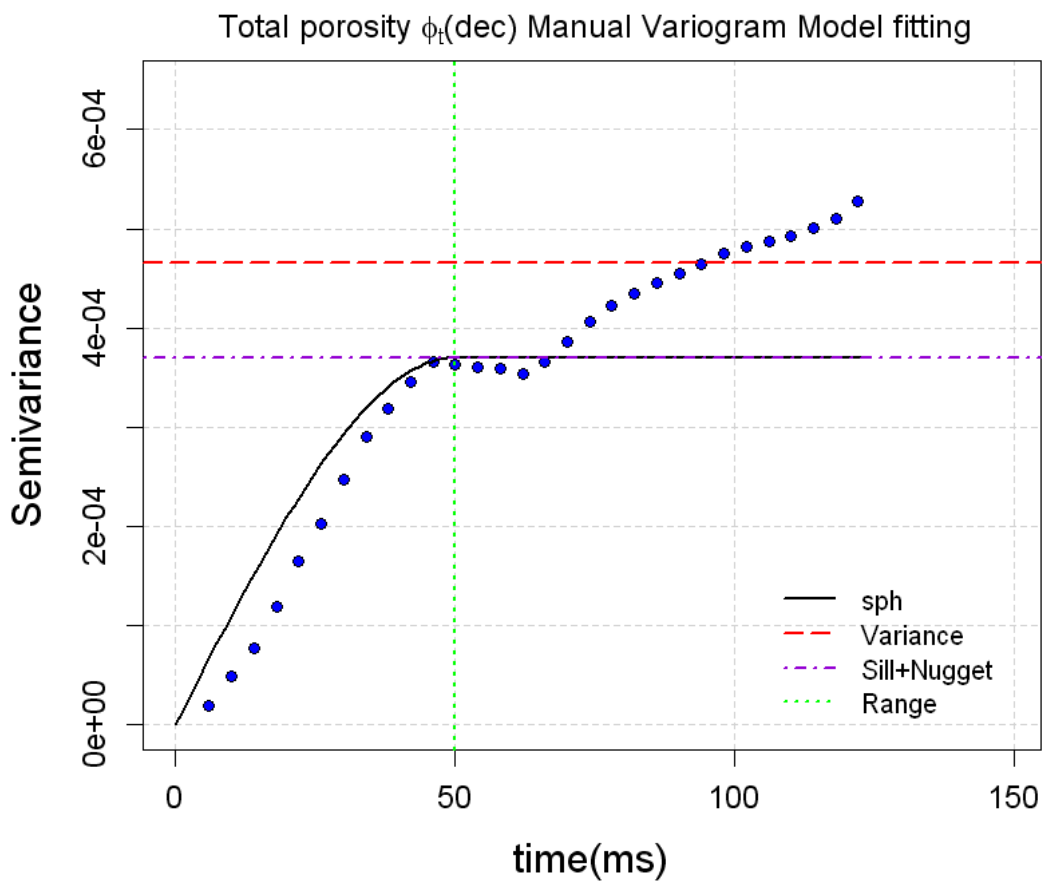
```
variog: computing omnidirectional variogram
```



Model	Nugget	Sill+Nugget	Range	MSE
sph	0.0000	0.0004	50.0000	0.0000

A matrix: 1 × 3 of type dbl

	Nugget	Meseta+Nugget	Alcance
sph	0	0.00037	50



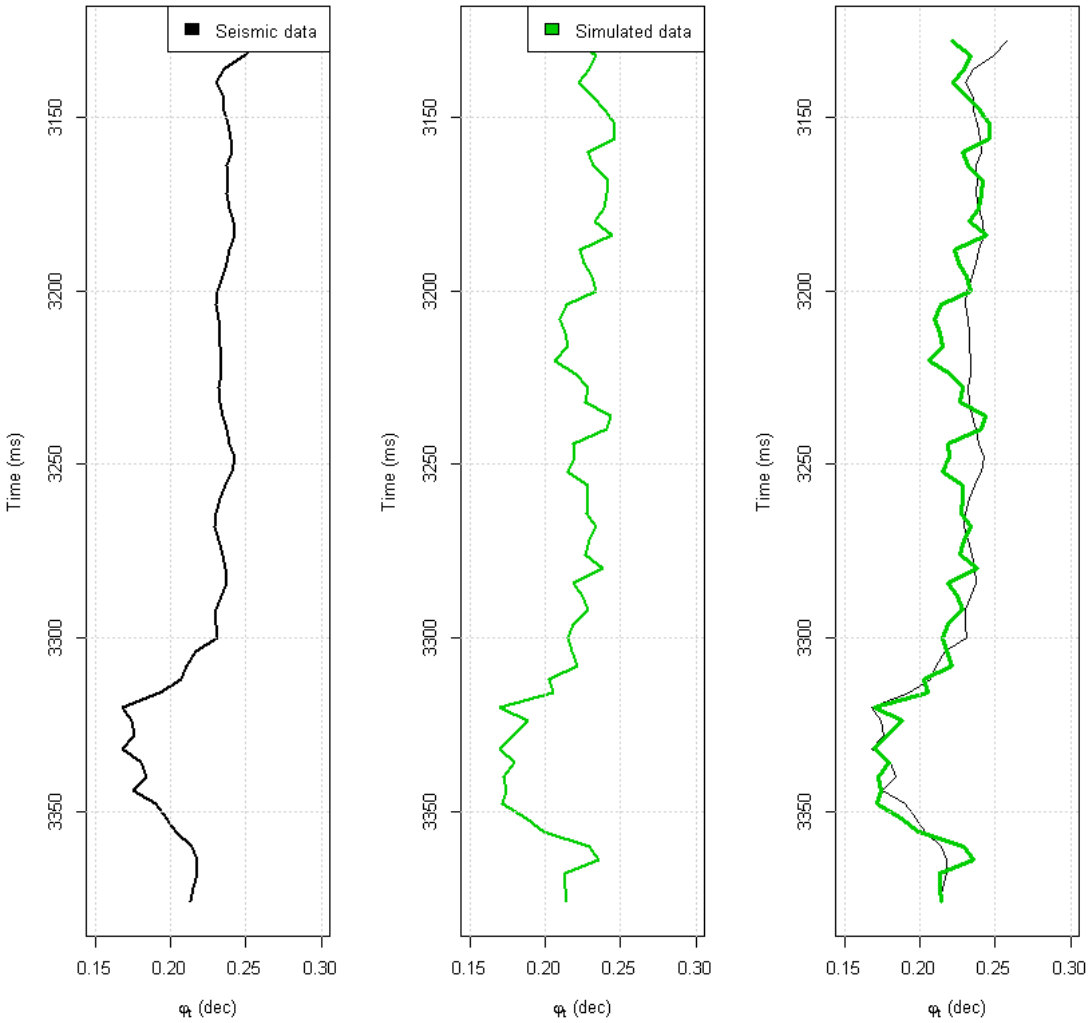
Model	Nugget	Sill+Nugget	Range	MSE
sph	0.0000	0.0004	50.0000	0.0000

[34]:
```
var_sim3p_Stat<-Estadisticas(s3$xbest)


Comparative_3p_Stat<-cbind(Phit_like_SUB_Stat,var_sim3p_Stat[,2])
#Comparative_post_Stat<-Val_Estadisticos(Comparative_post)
colnames(Comparative_3p_Stat)<-c("Statistics"," Total porosity subsampled",
                                 " Total porosity SAopt")
Comparative_3p_Stat
```

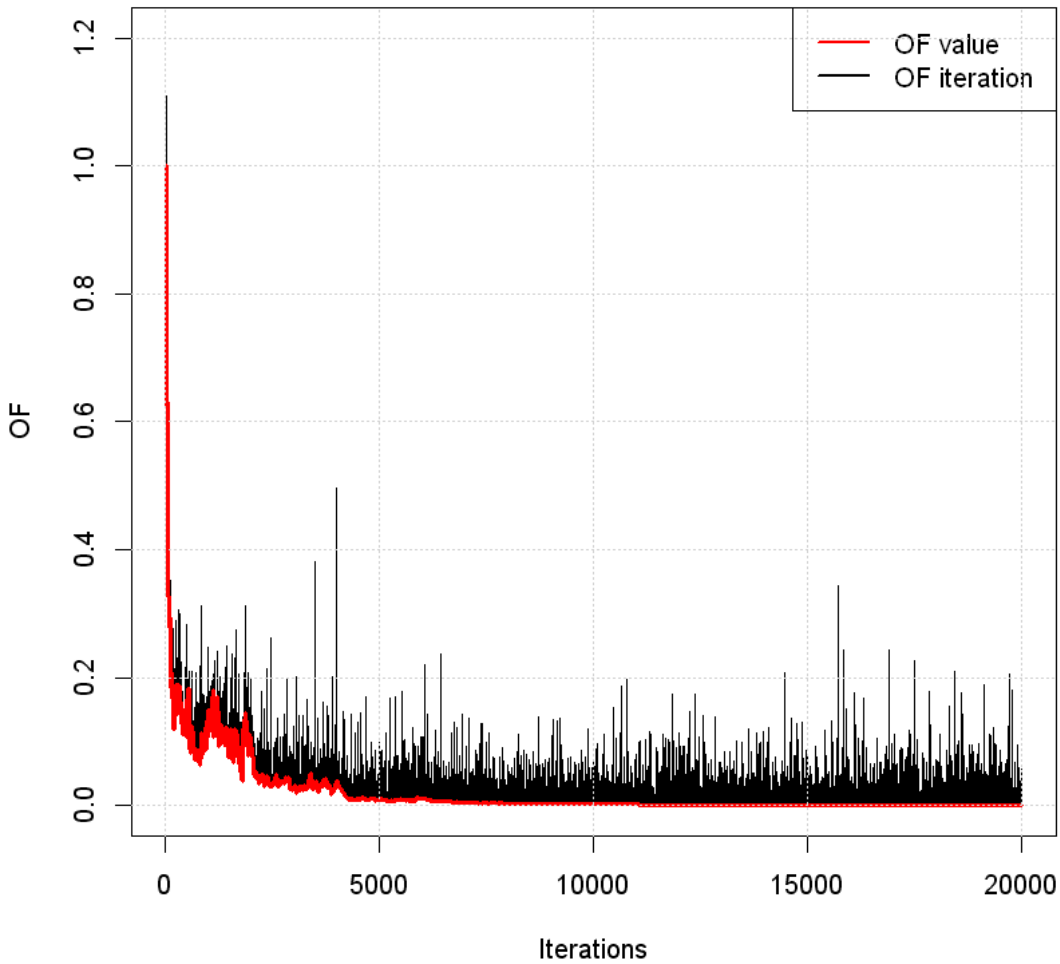
A data.frame: 14 × 3

	Statistics <chr>	Total porosity subsampled <dbl>	Total porosity SAopt <dbl>
muestras	n	63.0000	63.0000
minimos	Minimum	0.1678	0.1692
cuantiles1	1st. Quartile	0.2158	0.2130
medianas	Median	0.2324	0.2227
medias	Mean	0.2239	0.2185
cuantiles3	3rd. Quartile	0.2369	0.2320
maximos	Maximum	0.2575	0.2463
rangos	Rank	0.0898	0.0771
rangosInt	Interquartile Rank	0.0212	0.0189
varianzas	Variance	0.0005	0.0004
desvs	Standard Deviation	0.0216	0.0204
CVs	Variation Coeff.	0.0964	0.0934
simetrias	Skewness	-1.3036	-1.0468
curtosiss	Kurtosis	0.8845	0.5208

[35]:
```
par(mfrow = c(1,3))

plot(Phit_like_SUB,time_like_SUB,xlim=c(0.15,0.3),xlab =␣
 ↪expression(paste(varphi[t], " (dec)")),
     ylab="Time (ms)",ylim=rev(range(time_like_SUB)),type =␣
 ↪"l",lwd=2,col=("black"))
grid()
legend(x = "topright", legend = c("Seismic data"), fill = c("black"))

plot(s3$xbest,time_like_SUB,xlim=c(0.15,0.3),xlab = expression(paste(varphi[t],␣
 ↪" (dec)")),
     ylab="Time (ms)",ylim=rev(range(time_like_SUB)),type =␣
 ↪"l",lwd=2,col=("green3"))
grid()
legend(x = "topright", legend = c("Simulated data"), fill = c( "green3"))

plot(Phit_like_SUB,time_like_SUB,xlim=c(0.15,0.3),xlab =␣
 ↪expression(paste(varphi[t], " (dec)")),
     ylab="Time (ms)",ylim=rev(range(time_like_SUB)),type =␣
 ↪"l",lwd=1,col=("black"))
par(new=TRUE)
plot(s3$xbest,time_like_SUB,xlim=c(0.15,0.3),ylim=rev(range(time_like_SUB)),type␣
 ↪= "l",lwd=3,col=("green3"),
     xaxt='n',yaxt='n',ann=FALSE)
grid()
#legend(x = "topright", legend = c("Subsampled data",
#                                   "Simulated data"), fill = c("black",␣
 ↪"green3"))
```



[36]:
```
plot(s3$Fmat[ , 1L],type = "l",
     ylab = "OF", xlab = "Iterations",xlim=c(0,20000),ylim=c(0,1.
 ↪2),col=("black"))
par(new=TRUE)
plot(s3$Fmat[ , 2L],type = "l",
     ylab = "OF", xlab = "Iterations",xlim=c(0,20000),ylim=c(0,1.
 ↪2),col=("red"),lwd=3)
legend(x = "topright", legend = c("OF value", "OF iteration"),
       lty = 1, lwd = 2, col = c("red","black"))
grid()
```



[ ]:

```
```

[9]:

```
all_traces<-data.frame(Data_like_CUBE[,c(2:34)])
```

[6]:

```
Func_image  <- function(var_sim) {
  DF_var_sim<-data.frame(var_sim[,1])
  CRi<-matrix(0, Var_dim, 1)
  for (i in 1:(Var_dim-1)) {
    CRi[i,1] <- (DF_var_sim[(i+1),1]-DF_var_sim[i,1])/
↪(DF_var_sim[(i+1),1]+DF_var_sim[i,1])
  }
  synthetic<-sigconv(CRi,G)
  synthetic<-(synthetic*2.4e4)
```

```
  #----------------Función O1 variograma␣
↪Ip-------------------------------------------#
  Var_emp <- Variograma(CoorX, CoorY, Variable=var_sim[,1], Direccion=Direccion,␣
↪Tol=Tol, NIntervalos=NIntervalos,
                        Lags=Lags, Npares=1,graph=FALSE ,␣
↪MainTitle="Variograma", xlab="Distancia", ylab = "Semivarianza")
  semi_sim = Var_emp[,3]
  Fun1<-(sum(((semi_sim-semi_prior)/semi_prior)^2))/21.40191
␣
↪#---------------------------------------simulation-----------------------------------#
  #Var_emp1 <- Variograma(CoorX=time_like_SUB, CoorY=YCoord1,␣
↪Variable=synthetic, Direccion=Direccion1, Tol=Tol1, NIntervalos=NIntervalos1,
  #                       Lags=Lags1, Npares=1,graph=FALSE ,␣
↪MainTitle="Variograma", xlab="Distancia", ylab = "Semivarianza")
  #semi_sim1 = Var_emp[,3]
  #----------------Función O1-------------------------------------------#
  #Fun2=sum(((semi_sim1-semi_prior_sis)/semi_prior_sis)**2)
  #----------------Función O2-------------------------------------------#
  Fun3=((rmse(var_all,synthetic))/range_all)/0.2987836
  #return(Fun1+Fun2+Fun3)
  return(0.1*Fun1+0.9*Fun3)
}
```

[10]:
```
algo_image=list(nS=10,initT=0.01,alpha=0.22,nI=20000,x0=x0,neighbour=hchange,
          printDetail=5000,storeF=TRUE,OF.target=0.01)
```

[11]:
```
ncTRA<-ncol(all_traces)
nfTRA<-nrow(all_traces)

syn_tra<-matrix(0, nfTRA, ncTRA)
syn_tra_ip<-matrix(0, nfTRA, ncTRA)
syn_tra_phit<-matrix(0, nfTRA, ncTRA)
syn_tra_OFV<-matrix(0, algo$nI, ncTRA)
syn_tra_OFI<-matrix(0, algo$nI, ncTRA)
```

[12]:
```
system.time(for (j in 1:ncol(all_traces)) {
    # se selecciona una traza
    var_all<-all_traces[,j]
    #se estiman sus estadigrafos
    var_all_Stat<-Estadisticas(var_all)
    range_all<-var_all_Stat[8,2]
    # se ejecuta el recocido simulado
    simage<-SAopt(OF=Func_image, algo = algo_image)
    # se calcula la trza sintetica
    DF_var_fimage<-data.frame(simage$xbest[,1])
    CRfimage<-matrix(0, Var_dim, 1)
     for (i in 1:(Var_dim-1)) {
```

```
      CRfimage[i,1] <- (DF_var_fimage[(i+1),1]-DF_var_fimage[i,1])/
↪(DF_var_fimage[(i+1),1]+DF_var_fimage[i,1])
     }
    synthetic_fimage<-sigconv(CRfimage,G)
    syn_tra[,j]<-(synthetic_fimage*2.4e4)
    # se guarda la impedancia
    syn_tra_ip[,j]<-simage$xbest[,1]
    # se guarda la porosidad
    syn_tra_phit[,j]<-simage$xbest[,2]
    # se guarda el error de iteracion
    syn_tra_OFV[,j]<-simage$Fmat[ , 2L]
    # se guarda el mejor error encontrado
    syn_tra_OFI[,j]<-simage$Fmat[ , 1L]
})
```

```
Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  1.086603
Best solution (iteration 5000/20000): 0.2027595
Best solution (iteration 10000/20000): 0.1754705
Best solution (iteration 15000/20000): 0.1507783
Best solution (iteration 20000/20000): 0.1398416
Finished.
Best solution overall: 0.1398416

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  1.099514
Best solution (iteration 5000/20000): 0.2339936
Best solution (iteration 10000/20000): 0.1906578
Best solution (iteration 15000/20000): 0.1738689
Best solution (iteration 20000/20000): 0.1622726
Finished.
Best solution overall: 0.1622726

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  1.101554
Best solution (iteration 5000/20000): 0.164505
Best solution (iteration 10000/20000): 0.1272381
Best solution (iteration 15000/20000): 0.1211119
Best solution (iteration 20000/20000): 0.1161102
Finished.
Best solution overall: 0.1161102
```

```
Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  1.1726
Best solution (iteration 5000/20000): 0.1734233
Best solution (iteration 10000/20000): 0.1433405
Best solution (iteration 15000/20000): 0.1321892
Best solution (iteration 20000/20000): 0.1231802
Finished.
Best solution overall: 0.1231802

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  1.271048
Best solution (iteration 5000/20000): 0.1472722
Best solution (iteration 10000/20000): 0.1287823
Best solution (iteration 15000/20000): 0.1200484
Best solution (iteration 20000/20000): 0.1143222
Finished.
Best solution overall: 0.1143222

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  1.303002
Best solution (iteration 5000/20000): 0.1073593
Best solution (iteration 10000/20000): 0.0971449
Best solution (iteration 15000/20000): 0.0924319
Best solution (iteration 20000/20000): 0.08986099
Finished.
Best solution overall: 0.08986099

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  1.307303
Best solution (iteration 5000/20000): 0.1282936
Best solution (iteration 10000/20000): 0.1189477
Best solution (iteration 15000/20000): 0.1120985
Best solution (iteration 20000/20000): 0.1088248
Finished.
Best solution overall: 0.1088248

Simulated Annealing.

Running Simulated Annealing ...
```

```
Initial solution:  1.224206
Best solution (iteration 5000/20000): 0.1588758
Best solution (iteration 10000/20000): 0.1391467
Best solution (iteration 15000/20000): 0.1345955
Best solution (iteration 20000/20000): 0.1325661
Finished.
Best solution overall: 0.1325661

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  1.22623
Best solution (iteration 5000/20000): 0.1420203
Best solution (iteration 10000/20000): 0.1236285
Best solution (iteration 15000/20000): 0.1205484
Best solution (iteration 20000/20000): 0.1184217
Finished.
Best solution overall: 0.1184217

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  1.184076
Best solution (iteration 5000/20000): 0.1735447
Best solution (iteration 10000/20000): 0.1362942
Best solution (iteration 15000/20000): 0.1280966
Best solution (iteration 20000/20000): 0.1229682
Finished.
Best solution overall: 0.1229682

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  1.164885
Best solution (iteration 5000/20000): 0.1394551
Best solution (iteration 10000/20000): 0.1161254
Best solution (iteration 15000/20000): 0.1092077
Best solution (iteration 20000/20000): 0.1035217
Finished.
Best solution overall: 0.1035217

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  1.221343
Best solution (iteration 5000/20000): 0.1576475
Best solution (iteration 10000/20000): 0.1180545
Best solution (iteration 15000/20000): 0.1097086
```

```
Best solution (iteration 20000/20000): 0.103491
Finished.
Best solution overall: 0.103491

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  1.243541
Best solution (iteration 5000/20000): 0.1340072
Best solution (iteration 10000/20000): 0.1146268
Best solution (iteration 15000/20000): 0.1062559
Best solution (iteration 20000/20000): 0.1022814
Finished.
Best solution overall: 0.1022814

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  1.260704
Best solution (iteration 5000/20000): 0.1662741
Best solution (iteration 10000/20000): 0.1343311
Best solution (iteration 15000/20000): 0.1231419
Best solution (iteration 20000/20000): 0.1146737
Finished.
Best solution overall: 0.1146737

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  1.263353
Best solution (iteration 5000/20000): 0.1750725
Best solution (iteration 10000/20000): 0.1386461
Best solution (iteration 15000/20000): 0.1312957
Best solution (iteration 20000/20000): 0.1284869
Finished.
Best solution overall: 0.1284869

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  1.189477
Best solution (iteration 5000/20000): 0.1626385
Best solution (iteration 10000/20000): 0.1371091
Best solution (iteration 15000/20000): 0.1233492
Best solution (iteration 20000/20000): 0.1159012
Finished.
Best solution overall: 0.1159012
```

```
Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999999
Best solution (iteration 5000/20000): 0.1506562
Best solution (iteration 10000/20000): 0.1324342
Best solution (iteration 15000/20000): 0.1269562
Best solution (iteration 20000/20000): 0.1247656
Finished.
Best solution overall: 0.1247656

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.8985223
Best solution (iteration 5000/20000): 0.1779653
Best solution (iteration 10000/20000): 0.1382124
Best solution (iteration 15000/20000): 0.1266331
Best solution (iteration 20000/20000): 0.122929
Finished.
Best solution overall: 0.122929

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.8501846
Best solution (iteration 5000/20000): 0.1282081
Best solution (iteration 10000/20000): 0.107878
Best solution (iteration 15000/20000): 0.1008893
Best solution (iteration 20000/20000): 0.09814626
Finished.
Best solution overall: 0.09814626

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.8587307
Best solution (iteration 5000/20000): 0.1455799
Best solution (iteration 10000/20000): 0.09354027
Best solution (iteration 15000/20000): 0.07794519
Best solution (iteration 20000/20000): 0.07544504
Finished.
Best solution overall: 0.07544504

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.8581468
```

```
Best solution (iteration 5000/20000): 0.1279409
Best solution (iteration 10000/20000): 0.1137189
Best solution (iteration 15000/20000): 0.1069615
Best solution (iteration 20000/20000): 0.1043358
Finished.
Best solution overall: 0.1043358

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.8329171
Best solution (iteration 5000/20000): 0.1670529
Best solution (iteration 10000/20000): 0.1075993
Best solution (iteration 15000/20000): 0.09720076
Best solution (iteration 20000/20000): 0.0916074
Finished.
Best solution overall: 0.0916074

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.7706124
Best solution (iteration 5000/20000): 0.1408686
Best solution (iteration 10000/20000): 0.1017711
Best solution (iteration 15000/20000): 0.09567763
Best solution (iteration 20000/20000): 0.09321395
Finished.
Best solution overall: 0.09321395

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.732954
Best solution (iteration 5000/20000): 0.129487
Best solution (iteration 10000/20000): 0.1145326
Best solution (iteration 15000/20000): 0.1079648
Best solution (iteration 20000/20000): 0.1004648
Finished.
Best solution overall: 0.1004648

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.7347542
Best solution (iteration 5000/20000): 0.1300264
Best solution (iteration 10000/20000): 0.0976257
Best solution (iteration 15000/20000): 0.08442875
Best solution (iteration 20000/20000): 0.0810279
```

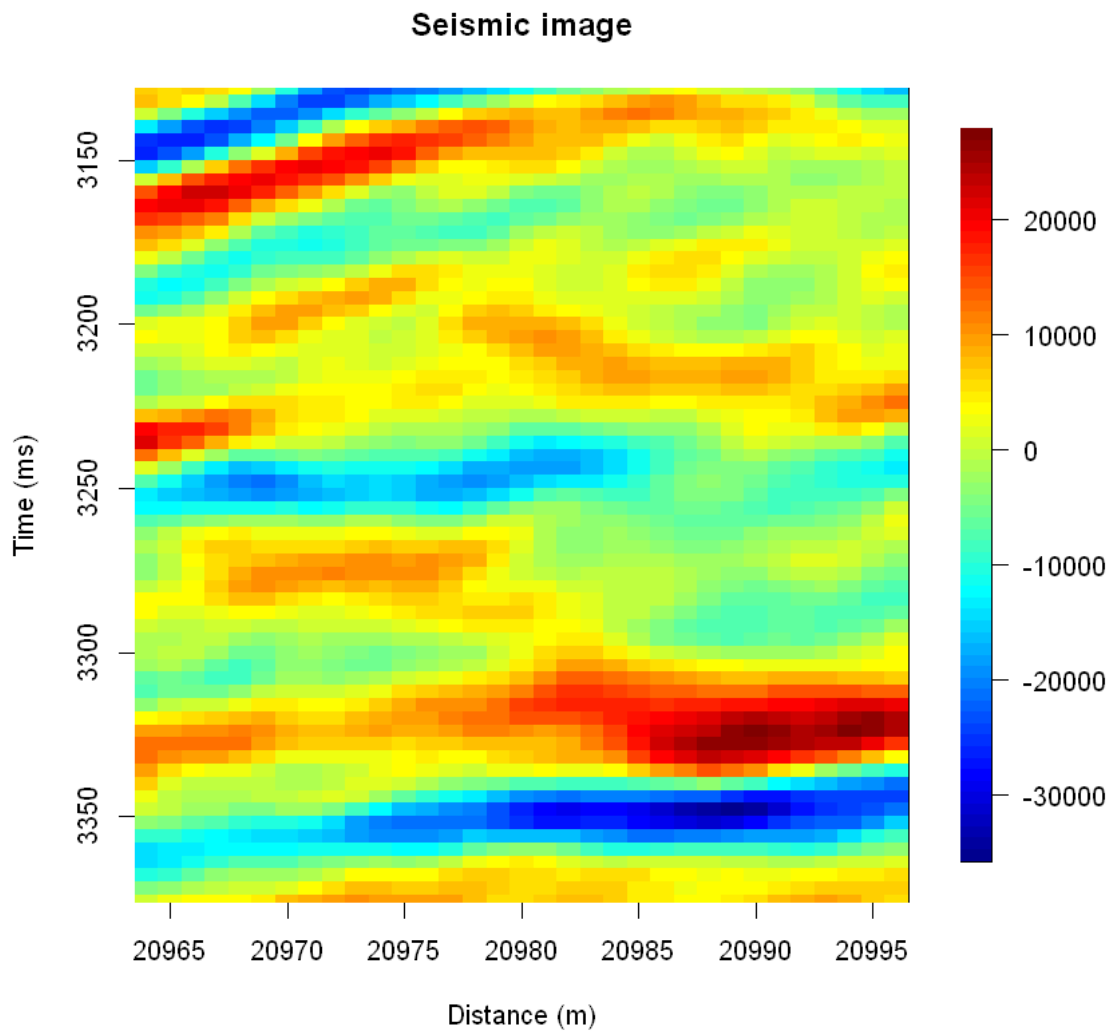
```
Finished.
Best solution overall: 0.0810279

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.7404661
Best solution (iteration 5000/20000): 0.1180387
Best solution (iteration 10000/20000): 0.09927496
Best solution (iteration 15000/20000): 0.09536117
Best solution (iteration 20000/20000): 0.09096158
Finished.
Best solution overall: 0.09096158

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.7507766
Best solution (iteration 5000/20000): 0.09227201
Best solution (iteration 10000/20000): 0.07902847
Best solution (iteration 15000/20000): 0.07393698
Best solution (iteration 20000/20000): 0.07122505
Finished.
Best solution overall: 0.07122505

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.7849893
Best solution (iteration 5000/20000): 0.08743842
Best solution (iteration 10000/20000): 0.06732982
Best solution (iteration 15000/20000): 0.05803329
Best solution (iteration 20000/20000): 0.05258736
Finished.
Best solution overall: 0.05258736

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.8368067
Best solution (iteration 5000/20000): 0.1171931
Best solution (iteration 10000/20000): 0.08510293
Best solution (iteration 15000/20000): 0.07683005
Best solution (iteration 20000/20000): 0.07394873
Finished.
Best solution overall: 0.07394873

Simulated Annealing.
```

```
Running Simulated Annealing ...
Initial solution:  0.8662806
Best solution (iteration 5000/20000): 0.1059566
Best solution (iteration 10000/20000): 0.07898983
Best solution (iteration 15000/20000): 0.06981955
Best solution (iteration 20000/20000): 0.06489449
Finished.
Best solution overall: 0.06489449

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.8548054
Best solution (iteration 5000/20000): 0.120673
Best solution (iteration 10000/20000): 0.08636684
Best solution (iteration 15000/20000): 0.0798092
Best solution (iteration 20000/20000): 0.07728832
Finished.
Best solution overall: 0.07728832

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.8593831
Best solution (iteration 5000/20000): 0.1584229
Best solution (iteration 10000/20000): 0.09809439
Best solution (iteration 15000/20000): 0.08508547
Best solution (iteration 20000/20000): 0.07493608
Finished.
Best solution overall: 0.07493608

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.8579167
Best solution (iteration 5000/20000): 0.1333081
Best solution (iteration 10000/20000): 0.1070291
Best solution (iteration 15000/20000): 0.102027
Best solution (iteration 20000/20000): 0.09529375
Finished.
Best solution overall: 0.09529375

   user  system elapsed
1740.49  168.66 6045.71
```

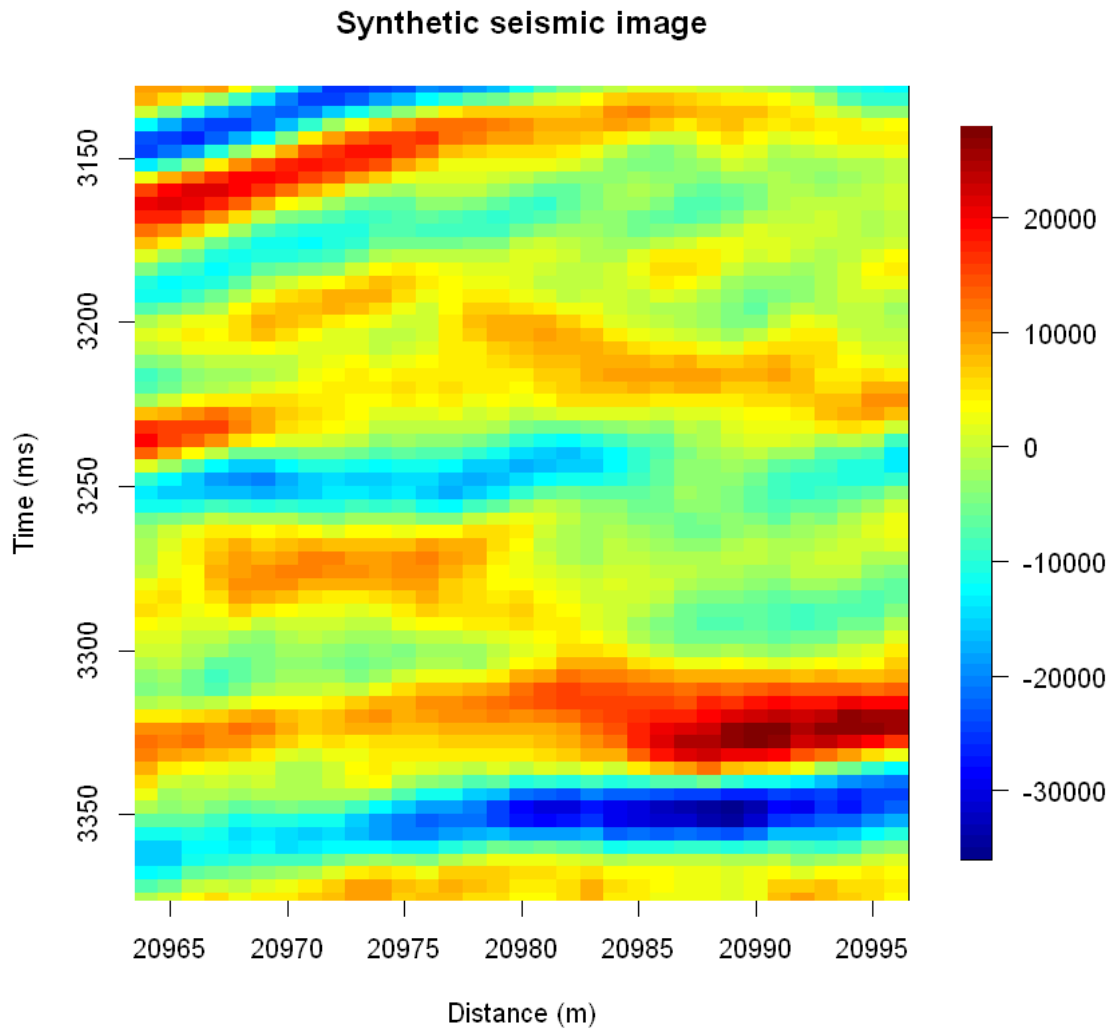
[13]:
```r
write.csv(syn_tra, file = "Results/syn_tra.csv")
write.csv(syn_tra_ip, file = "Results/syn_tra_ip.csv")
```

```
write.csv(syn_tra_phit, file = "Results/syn_tra_phit.csv")
write.csv(syn_tra_OFV, file = "Results/syn_tra_OFV.csv")
write.csv(syn_tra_OFI, file = "Results/syn_tra_OFI.csv")
```

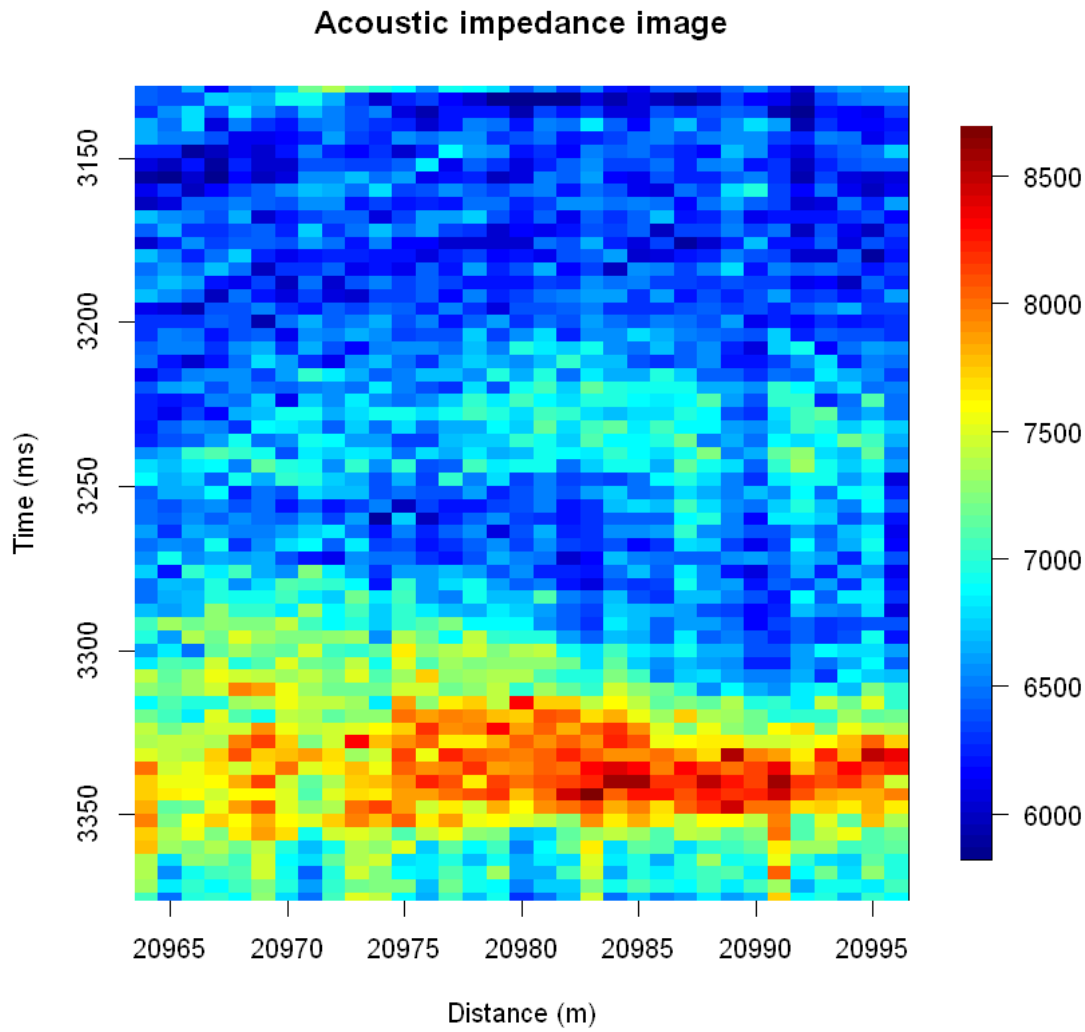
[31]:
```
image.plot(x=seq(from=20964, to=20996,by=1),␣
↪y=time_like_SUB,z=t(all_traces),xlab = "Distance (m)",ylab="Time␣
↪(ms)",main="Seismic image",
    ylim=rev(range(time_like_SUB)))
```



[28]:
```
image.plot(x=seq(from=20964, to=20996,by=1), y=time_like_SUB,z=t(syn_tra),xlab =␣
↪"Distance (m)",ylab="Time (ms)",main="Synthetic seismic image",
    ylim=rev(range(time_like_SUB)))
```



[27]:
```
image.plot(x=seq(from=20964, to=20996,by=1),␣
↪y=time_like_SUB,z=t(syn_tra_ip),xlab = "Distance (m)",ylab="Time␣
↪(ms)",main="Acoustic impedance image",
    ylim=rev(range(time_like_SUB)))
```



8.8 simulacion de la porosidd total condicionada a la impedancia acustica.

[5]:
```
syn_tra_phit_cond<-matrix(0, nfTRA, ncTRA)
```

[37]:
```
hchange_cond_phi_image<-function(x0)
{
  i=0
  sus<-round(runif(1,1,Var_dim),0)
  while (i==0) {
        Ip_sam<-Ip_SA[sus]
        Ip_cond <- plnorm(Ip_sam,meanlog = mu_best_post_5TRA, sdlog =␣
 ↪sigma_best_post_5TRA)
```

```
        Fr_cond<-cCopula(cbind(Ip_cond,runif(length(Ip_cond))),
                    copula = frankCopula(theta_best_post_5TRA), inverse = TRUE)
        samcond<-qweibull(Fr_cond[,2],shape = alpha_best_post_5TRA, scale =␣
↪lambda_best_post_5TRA)
    if(samcond>=lower_phit & samcond<=upper_phit){
      i=1}
    else{
      i=0}
  }
  x0[sus]<-samcond
  return(x0)
}
```

[38]:
```
FuncO2  <- function(var_sim) {
 ␣
 ↪#----------------------------------------simulation----------------------------------#
     Var_emp1 <- Variograma(CoorX, CoorY, Variable=var_sim, Direccion=Direccion,␣
 ↪Tol=Tol, NIntervalos=N_lagsp,
                                Lags=lag_valuep, Npares=1,graph=FALSE ,␣
 ↪MainTitle="Variograma", xlab="Distancia",
                             ylab = "Semivarianza")
  semi_sim1 = Var_emp1[,3]
  #----------------Función O1-----------------------------------------#
  Fun2=sum(((semi_sim1-semi_priorp)/semi_priorp)**2)/2.883024
  #----------------Función O2-----------------------------------------#
  #Fun3=(rmse(var_sis,synthetic)/neartrace_Stat[8,2])
  return(Fun2)
}
```

[39]:
```
algo_cond=list(nS=10,initT=0.01,alpha=0.22,nI=20000,x0=Phit_like_SUB,␣
 ↪neighbour=hchange_cond_phi_image,
          printDetail=5000,storeF=TRUE,OF.target=0.0001)
```

[40]:
```
system.time(for (j in 1:ncol(syn_tra_ip)) {
    Ip_SA<-syn_tra_ip[,j]
    simage_phit<-SAopt(OF=FuncO2, algo = algo_cond)
    # se calcula la trza sintetica
    syn_tra_phit_cond[,j]<-simage_phit$xbest
})
```

```
Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.003430361
Best solution (iteration 10000/20000): 0.0009087393
```

```
Best solution (iteration 15000/20000): 0.0003956973
Best solution (iteration 20000/20000): 0.0002259367
Finished.
Best solution overall: 0.0002259367

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.004538372
Best solution (iteration 10000/20000): 0.001095237
Best solution (iteration 15000/20000): 0.0007464057
Best solution (iteration 20000/20000): 0.0006479557
Finished.
Best solution overall: 0.0006479557

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.005278423
Best solution (iteration 10000/20000): 0.0009184856
Best solution (iteration 15000/20000): 0.000516885
Best solution (iteration 20000/20000): 0.0004230307
Finished.
Best solution overall: 0.0004230307

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.007613355
Best solution (iteration 10000/20000): 0.002195125
Best solution (iteration 15000/20000): 0.00124699
Best solution (iteration 20000/20000): 0.0008789052
Finished.
Best solution overall: 0.0008789052

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.003300202
Best solution (iteration 10000/20000): 0.001345822
Best solution (iteration 15000/20000): 0.0004489071
Best solution (iteration 20000/20000): 0.0002750415
Finished.
Best solution overall: 0.0002750415
```

```
Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.006658977
Best solution (iteration 10000/20000): 0.001490109
Best solution (iteration 15000/20000): 0.0007368267
Best solution (iteration 20000/20000): 0.000595001
Finished.
Best solution overall: 0.000595001

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.01068971
Best solution (iteration 10000/20000): 0.001662817
Best solution (iteration 15000/20000): 0.00130514
Best solution (iteration 20000/20000): 0.0009777259
Finished.
Best solution overall: 0.0009777259

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.006190449
Best solution (iteration 10000/20000): 0.0008613674
Best solution (iteration 15000/20000): 0.0003048977
Best solution (iteration 20000/20000): 0.000241919
Finished.
Best solution overall: 0.000241919

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.006041321
Best solution (iteration 10000/20000): 0.002020248
Best solution (iteration 15000/20000): 0.001159703
Best solution (iteration 20000/20000): 0.0008229681
Finished.
Best solution overall: 0.0008229681

Simulated Annealing.

Running Simulated Annealing ...
```

```
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.01180125
Best solution (iteration 10000/20000): 0.001798671
Best solution (iteration 15000/20000): 0.001049966
Best solution (iteration 20000/20000): 0.0008508856
Finished.
Best solution overall: 0.0008508856

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.005329889
Best solution (iteration 10000/20000): 0.001513485
Best solution (iteration 15000/20000): 0.0008387644
Best solution (iteration 20000/20000): 0.0007242293
Finished.
Best solution overall: 0.0007242293

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.004943067
Best solution (iteration 10000/20000): 0.001258418
Best solution (iteration 15000/20000): 0.0008000311
Best solution (iteration 20000/20000): 0.0006763884
Finished.
Best solution overall: 0.0006763884

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.01100033
Best solution (iteration 10000/20000): 0.0006115398
Best solution (iteration 15000/20000): 0.0004159762
Best solution (iteration 20000/20000): 0.0003097348
Finished.
Best solution overall: 0.0003097348

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.005053058
Best solution (iteration 10000/20000): 0.001772766
Best solution (iteration 15000/20000): 0.0008254074
```

```
Best solution (iteration 20000/20000): 0.0006018938
Finished.
Best solution overall: 0.0006018938

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.006062857
Best solution (iteration 10000/20000): 0.001267729
Best solution (iteration 15000/20000): 0.0008758533
Best solution (iteration 20000/20000): 0.0006361795
Finished.
Best solution overall: 0.0006361795

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.0078339
Best solution (iteration 10000/20000): 0.002592269
Best solution (iteration 15000/20000): 0.001784483
Best solution (iteration 20000/20000): 0.001357899
Finished.
Best solution overall: 0.001357899

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.006472707
Best solution (iteration 10000/20000): 0.001768216
Best solution (iteration 15000/20000): 0.001099869
Best solution (iteration 20000/20000): 0.0009684949
Finished.
Best solution overall: 0.0009684949

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.009369927
Best solution (iteration 10000/20000): 0.001678925
Best solution (iteration 15000/20000): 0.0009845587
Best solution (iteration 20000/20000): 0.0006897791
Finished.
Best solution overall: 0.0006897791
```

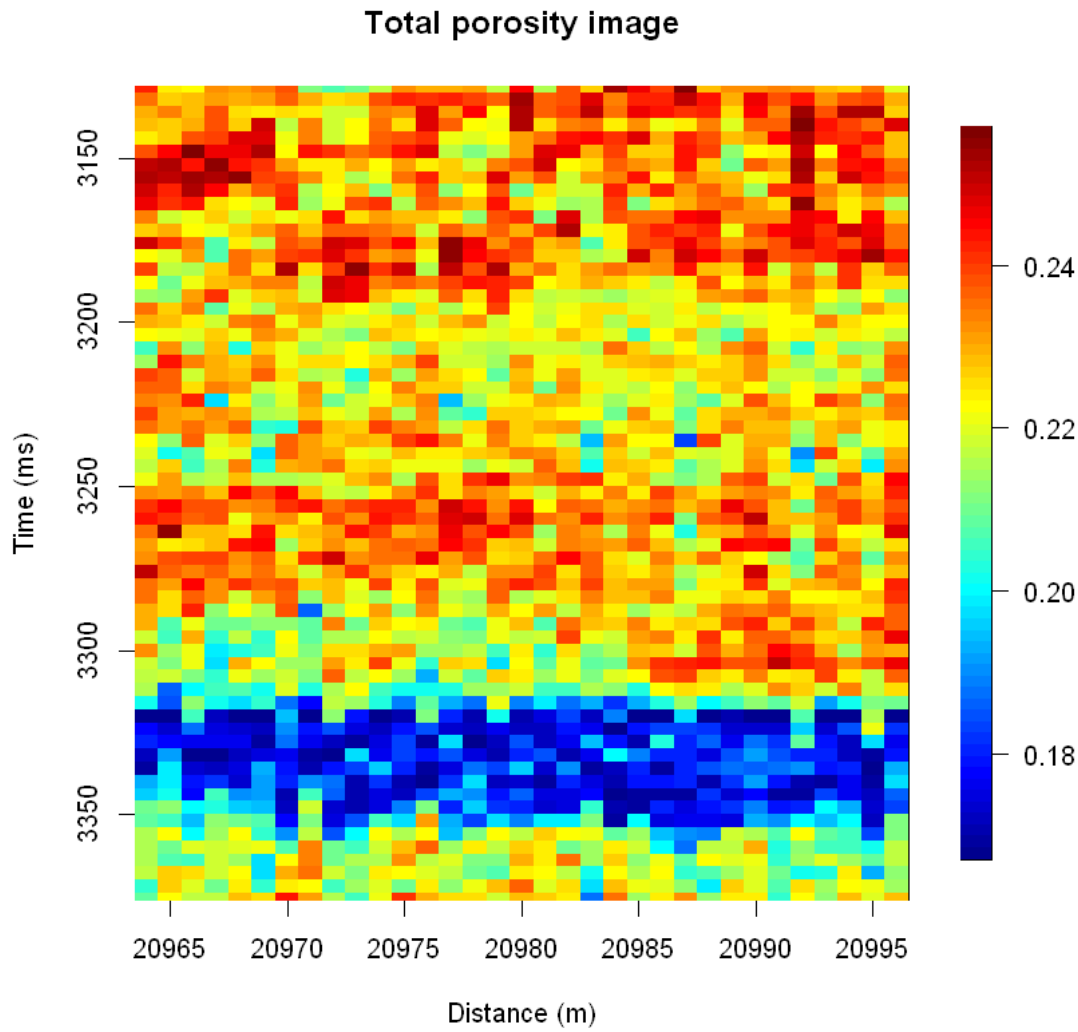
```
Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.005371472
Best solution (iteration 10000/20000): 0.001127552
Best solution (iteration 15000/20000): 0.0006512418
Best solution (iteration 20000/20000): 0.0004715899
Finished.
Best solution overall: 0.0004715899

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.01096189
Best solution (iteration 10000/20000): 0.0008159279
Best solution (iteration 15000/20000): 0.0005682376
Best solution (iteration 20000/20000): 0.0004803474
Finished.
Best solution overall: 0.0004803474

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.01026939
Best solution (iteration 10000/20000): 0.002239716
Best solution (iteration 15000/20000): 0.0009275862
Best solution (iteration 20000/20000): 0.0007026971
Finished.
Best solution overall: 0.0007026971

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.006148316
Best solution (iteration 10000/20000): 0.001542629
Best solution (iteration 15000/20000): 0.0006164477
Best solution (iteration 20000/20000): 0.0003905188
Finished.
Best solution overall: 0.0003905188

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
```

```
Best solution (iteration 5000/20000): 0.003664296
Best solution (iteration 10000/20000): 0.001103609
Best solution (iteration 15000/20000): 0.0005502844
Best solution (iteration 20000/20000): 0.0004279522
Finished.
Best solution overall: 0.0004279522

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.01022456
Best solution (iteration 10000/20000): 0.001771089
Best solution (iteration 15000/20000): 0.001399973
Best solution (iteration 20000/20000): 0.001247252
Finished.
Best solution overall: 0.001247252

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.002898541
Best solution (iteration 10000/20000): 0.0007071105
Best solution (iteration 15000/20000): 0.0003544919
Best solution (iteration 20000/20000): 0.0002964637
Finished.
Best solution overall: 0.0002964637

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.003861262
Best solution (iteration 10000/20000): 0.0008139247
Best solution (iteration 15000/20000): 0.0004233951
Best solution (iteration 20000/20000): 0.0003351822
Finished.
Best solution overall: 0.0003351822

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.003964015
Best solution (iteration 10000/20000): 0.0007504278
Best solution (iteration 15000/20000): 0.0004217768
Best solution (iteration 20000/20000): 0.0003628453
```

```
Finished.
Best solution overall: 0.0003628453

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.007153903
Best solution (iteration 10000/20000): 0.0008379232
Best solution (iteration 15000/20000): 0.0004868206
Best solution (iteration 20000/20000): 0.0003374156
Finished.
Best solution overall: 0.0003374156

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.006500713
Best solution (iteration 10000/20000): 0.001140187
Best solution (iteration 15000/20000): 0.0007102804
Best solution (iteration 20000/20000): 0.0005076963
Finished.
Best solution overall: 0.0005076963

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.00675184
Best solution (iteration 10000/20000): 0.001279232
Best solution (iteration 15000/20000): 0.0004277623
Best solution (iteration 20000/20000): 0.0003500021
Finished.
Best solution overall: 0.0003500021

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.006658142
Best solution (iteration 10000/20000): 0.0008847014
Best solution (iteration 15000/20000): 0.0004912151
Best solution (iteration 20000/20000): 0.0004137818
Finished.
Best solution overall: 0.0004137818

Simulated Annealing.
```

```
Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.01032566
Best solution (iteration 10000/20000): 0.002172096
Best solution (iteration 15000/20000): 0.001095244
Best solution (iteration 20000/20000): 0.0008407176
Finished.
Best solution overall: 0.0008407176

Simulated Annealing.

Running Simulated Annealing ...
Initial solution:  0.9999998
Best solution (iteration 5000/20000): 0.005413915
Best solution (iteration 10000/20000): 0.000799921
Best solution (iteration 15000/20000): 0.000458787
Best solution (iteration 20000/20000): 0.0003056161
Finished.
Best solution overall: 0.0003056161

   user  system elapsed
5214.92  401.22 5582.56
```

[26]:
```
image.plot(x=seq(from=20964, to=20996,by=1),␣
 ↪y=time_like_SUB,z=t(syn_tra_phit_cond),xlab = "Distance (m)",ylab="Time (ms)",
          main="Total porosity image",ylim=rev(range(time_like_SUB)))
```



[32]:
```
save.image()
```

9 Discussion

As shown in Section~**??**, the Bayesian approach yielded a posterior dependence model that is consistent with the well-log information. While the marginal distributions remained relatively stable despite scaling and subsampling, the dependence parameter θ experienced a significant change—from -6.265 to -21. Interestingly, the Spearman dependence measure did not exhibit a corresponding large change, highlighting the sensitivity of the copula parameter to the joint structure.

When comparing the cumulative distribution function (CDF) generated using the posterior parameter values against total porosity ϕ_t, Figure~**??**C shows good agreement with the scaled poros-

ity values, particularly in the range of 0.17 to 0.23. At higher porosity values, however, some deviation is observed. For acoustic impedance (Ip), Figure~**??**A reveals that the posterior cumulative distribution closely follows the scaled CDF, with only minor discrepancies between 5,950 and 6,200~$(m/s)(g/cm^3)$. The posterior fits shown in Figures~**??**C and D represent improvements over the scaled distributions.

The acoustic impedance results obtained via the simulated annealing method also demonstrate high consistency with the subsampled logs. As shown in Table~**??**, most statistical descriptors are comparable between the simulated and subsampled acoustic impedance, with the main exception being variance. Similarly, the total porosity simulation yields statistics that closely match those of the subsampled data.

Another notable observation is the use of subsampled logs as the initial image in the simulation process. This contributes to consistency with the borehole structure and facilitates alignment with seismic features. For instance, the degradation pattern in the amplitude value between 20,963 and 20,970 reveals a decrease in acoustic impedance that corresponds to a polarity reversal in the seismic trace—from positive to negative—highlighting the inversion method's sensitivity to seismic features.

The current implementation assumes uniform vertical resolution across the seismic inline, and the well-log data were upscaled accordingly. Total porosity was averaged using a moving window equivalent to the vertical seismic resolution, and acoustic impedance was computed using the Backus averaging method applied to Vp and ρ logs. The dominant frequency of the seismic signal, estimated at 20.16~Hz (see Figure~**??**), was used to define the resolution scale. While this approach mitigates resolution mismatch, seismic data often exhibit variable resolution due to acquisition and processing differences. Future work will incorporate adaptive upscaling strategies or multiresolution inversion frameworks (Bunks et al., 2012), enabling the model to better adapt to spatially variable resolution and improve inversion accuracy.

A key limitation of the current implementation is the absence of lateral variogram modeling. The inversion was carried out using a trace-by-trace scheme with vertical variograms estimated in the time domain, restricting spatial continuity to the vertical direction. Although the inverted sections display structural coherence that aligns with the well data and reflectivity patterns, the lack of lateral spatial constraints may limit the model's ability to capture horizontally extensive geological features. Future efforts will focus on integrating three-dimensional spatial models by including lateral variograms derived from seismic attributes or well-log correlations, thereby improving the realism and consistency of the simulated volumes.

While the proposed method demonstrates stable and consistent results using parametric copulas, it is important to acknowledge the trade-offs associated with this choice. Parametric copulas, such as the Frank copula used here, are less flexible than non-parametric alternatives like Bernstein copulas. Their analytical structure imposes assumptions that may limit accuracy in datasets with tail asymmetry or multimodal joint behavior. Nonetheless, parametric copulas offer computational efficiency, interpretability, and ease of integration into a Bayesian framework, making them suitable for moderately complex dependencies like the negatively correlated acoustic impedance and porosity in this study. Users are advised to evaluate the dependence structure in their data before selecting a copula model. Future work may explore hybrid approaches that combine the efficiency of parametric forms with the flexibility of non-parametric models to handle more complex geological scenarios.

Scalability to large-scale datasets is another consideration. Although parametric copulas reduce

the computational cost, the current inversion framework is constrained by the trace-by-trace simulation scheme and reliance on MCMC for Bayesian estimation. As such, the method is best suited for 2D lines or small-to-moderate 3D volumes. To improve scalability, future developments will explore parallelized simulations across traces, as well as more efficient inference techniques (e.g., variational Bayes, Hamiltonian Monte Carlo). Global optimization strategies may also reduce dependence on repeated forward modeling, enabling the method's application to high-resolution 3D seismic datasets.

Another limitation of this study is the use of a single well to define marginal distributions and dependence structure. This choice was driven by data availability: while neighboring wells exist, most lack total porosity or acoustic impedance logs. Nevertheless, multi-well integration would increase the statistical robustness and spatial representativeness of the joint model. In future applications, this could be achieved by jointly estimating copula parameters across multiple wells or allowing spatially varying copula models. Additional wells would also improve variogram calibration and simulation conditioning, especially in areas far from the reference well.

10 Conclusions

The joint seismic inversion of elastic and petrophysical properties using copula-based stochastic co-simulation models presents a novel approach that enables the simultaneous simulation of both property types using a flexible dependence structure, without assuming a predefined joint probability distribution. The method is effective for generating spatially consistent realizations, particularly when the principal elastic attribute (acoustic impedance) is used as a conditioning variable. Furthermore, new data from additional wells or seismic surveys can be integrated into the model via a Bayesian inference framework.

By embedding the copula-based dependence model into the forward modeling process, the Bayesian approach enables the estimation of posterior parameters using both scaled and subsampled log samples. The resulting joint posterior distribution supports robust simulations through simulated annealing, yielding stable elastic and petrophysical models that honor both geophysical and statistical constraints.

The inversion results demonstrate that, when acoustic impedance is available at the seismic scale, the copula-based method can generate predictive realizations of petrophysical properties—such as total porosity—that are consistent with upscaled well-log data. These realizations preserve key spectral characteristics, including the dominant frequency and bandwidth of the seismic signal, and align well with observed reflectivity patterns.

While the method currently employs vertical variograms and a trace-by-trace inversion scheme, it produced coherent spatial distributions in both acoustic impedance and porosity. Future research will incorporate lateral variogram modeling to enhance horizontal continuity and structural realism.

The use of parametric copulas, particularly the Frank copula, provides a balance between modeling capacity and computational efficiency. Although less flexible than non-parametric alternatives, parametric copulas are well-suited for moderate complexity datasets, such as the negatively correlated acoustic impedance and porosity observed in this case study.

This framework is most applicable to 2D seismic lines or small-to-moderate 3D volumes. Future extensions will focus on improving scalability through parallelized inversion routines and more

efficient Bayesian inference techniques. Additionally, incorporating data from multiple wells and extending the framework to simulate multiple elastic and petrophysical properties such as those required in amplitude versus offset (AVO) inversion are promising directions for continued development.

11 Acknowledgments

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. Data will be shared on request to the corresponding author with permission of the National Hydrocarbons Commission.

References

Albert, J. (2009). *Bayesian computation with r*. Use R! Springer New York.

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

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

Azevedo, L. and Soares, A. (2017). *Geostatistical methods for reservoir geophysics*. Springer International Publishing.

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

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

Bunks, C., Saleck, F. M., Zaleski, S., and Chavent, G. (2012). Multiscale seismic waveform inversion. *https://doi.org/10.1190/1.1443880*, 60:1457–1473.

Cooke, D. A. and Schneider, W. A. (1983). Generalized linear inversion of reflection seismic data. *Geophysics*, 48(6):665–676.

Cosentino, L. (2001). *Integrated reservoir studies*, volume 53. Journal of Chemical Information and Modeling.

Deutsch, C. V. and Journel, A. G. (1998). *Gslib: geostatistical software library and user's guide*, volume 369. Springer New York.

Diaz-Viera, M., Erdely, A., Kerdan, T., Valle, R. D., and Mendoza-Torres, F. (2017). Bernstein copula-based spatial stochastic simulation of petrophysical properties using seismic attributes as secondary variable. *Geostatistics Valencia 2016*, pages 487–504.

Díaz-Viera, M. A., Le, V. H., and Vázquez-Ramírez, D. (2021). *Rgeoestad 2.0: un programa de código abierto para aplicaciones geoestadísticas basado en r-project.*

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

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.

Díaz-Viera, M. A. (2002). Geoestadística aplicada. *Instituto de Geofísica*, page 144.

Erdely, A. and Díaz-Viera, M. A. (2010). Nonparametric and semiparametric bivariate modeling of petrophysical porosity-permeability dependence from well log data. *In: Jaworski P., Durante F., Härdle W., Rychlik T. (eds) Copula Theory and Its Applications. Lecture Notes in Statistics. Springer, Berlin, Heidelberg*, 198:267–278.

Erdely, A. and Díaz-Viera, M. A. (2015). A vine and gluing copula model for permeability stochastic simulation. In *Stochastic and data analysis methods and applications in statistics and demography*, pages 273–281.

Gelfand, V. A. and Larner, K. L. (1984). Seismic lithologic modeling. *The Leading Edge*, 3(11):30–34.

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

Grana, D., Fjeldstad, T., and Omre, H. (2017). Bayesian gaussian mixture linear inversion for geophysical inverse problems. *Mathematical Geosciences*, 49.

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

Hernández-Maldonado, V., Díaz-Viera, M. A., and Erdely, A. (2012). A joint stochastic simulation method using the bernstein copula as a flexible tool for modeling nonlinear dependence structures between petrophysical properties. *Journal of Petroleum Science and Engineering*, 90-91:112–123.

Hernández-Maldonado, V., Díaz-Viera, M. A., and Erdely, A. (2014). A multivariate bernstein copula model for permeability stochastic simulation. *Geofísica Internacional*, 53:163–181.

Hofert, M., Kojadinovic, I., Mächler, M., and Yan, J. (2019). *Elements of copula modeling with r*. Springer International Publishing.

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, H. (2014). *Dependence modeling with copulas*. Taylor and Francis.

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

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

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.

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

Oldenburg, D. W., Scheuer, T., and Levy, S. (1983). Recovery of the acoustic impedance from reflection seismograms. *GEOPHYSICS*, 48(10):1318–1337.

Pereira, A., Azevedo, L., and Soares, A. (2023). Updating local anisotropies with template matching during geostatistical seismic inversion. *Mathematical Geosciences*, 55:497–519.

Pereira, P., Calçôa, I., Azevedo, L., Nunes, R., and Soares, A. (2020). Iterative geostatistical seismic inversion incorporating local anisotropies. *Computational Geosciences*, 24:1589–1604.

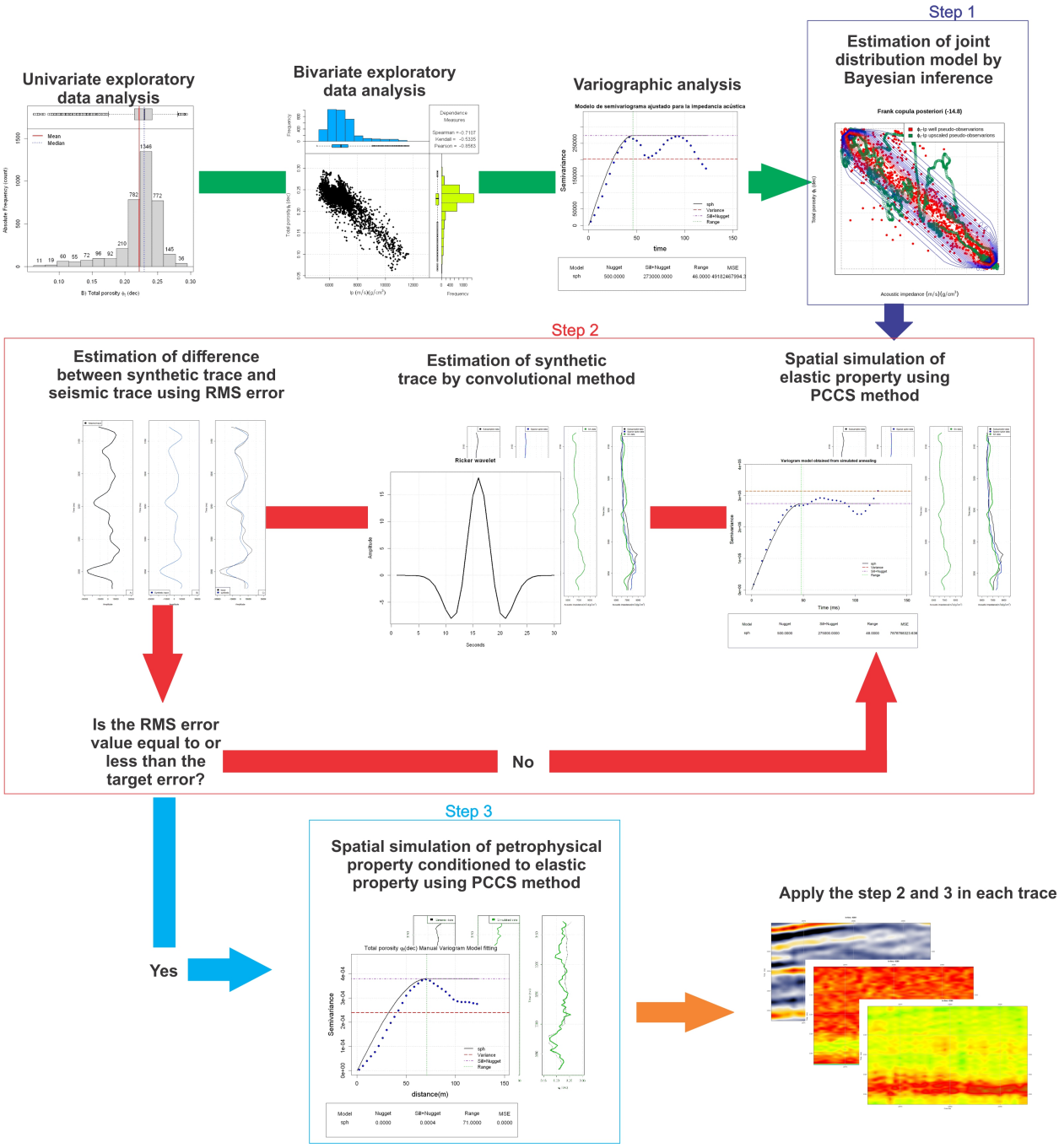
Pyrcz, M. J. and Deutsch, C. V. (2014). *Geostatistical reservoir modeling*. OUP USA.

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

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

Vázquez-Ramírez, D. (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.

Vázquez-Ramírez, D., Huong Le, V., Díaz-Viera, M. A., del Valle-García, R., and Erdely, A. (2023). Joint stochastic simulation of petrophysical properties with elastic attributes based on parametric copula models. *Geofísica Internacional*, 62(2):487–506.


Univariate exploratory data analysis
Bivariate exploratory data analysis
Variographic analysis
Step 1
Estimation of joint distribution model by Bayesian inference
Step 2
Estimation of difference between synthetic trace and seismic trace using RMS error
Estimation of synthetic trace by convolutional method
Spatial simulation of elastic property using PCCS method
Is the RMS error value equal to or less than the target error?
No
Yes
Step 3
Spatial simulation of petrophysical property conditioned to elastic property using PCCS method
Apply the step 2 and 3 in each trace