Joint stochastic simulation of petrophysical properties with elastic attributes based on parametric copula models: codes and notes

April 7, 2022

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

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

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

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

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

(5) Universidad Nacional Autónoma de México, FES Acatlán, Avenida Alcanfores y San Juan Totoltepec S/N, Santa Cruz Acatlán, 53150 Naucalpan de Juárez, Mexico, arturo.erdely@comunidad.unam.mx

1 Resumen

La co-simulación estocástica espacial basada en cópulas es un método general que permite simular variables con cualquier tipo de dependencia y funciones de distribución de probabilidad. Esta flexibilidad proviene del uso de un modelo de cópula para la representación de la función de distribución de probabilidad conjunta. El método se ha implementado principalmente a través de un enfoque no paramétrico utilizando cópulas de Bernstein y se ha aplicado con éxito para la simulación de propiedades petrofísicas usando atributos sísmicos elásticos como variables secundarias. En el presente trabajo este método se implementa mediante un enfoque paramétrico, específicamente, utilizando como alternativa las cópulas Arquimedianas. Primero, el enfoque paramétrico se valida con un caso publicado y luego se aplica a un caso de estudio. Ambos enfoques se comparan en términos de precisión y rendimiento. Los resultados mostraron que el enfoque paramétrico no reproduce las estadísticas de los datos y presenta más incertidumbre que el enfoque no paramétrico, mientras que el enfoque paramétrico tiene un costo computacional bajo en comparación con el enfoque no paramétrico.

Palabras clave: co-simulación, cópulas, Bernstein, Arquimedeanas, propiedades petrofísicas, atributos sísmicos elásticos.

2 Abstract

Copula-based spatial stochastic co-simulación is a general method that allows simulating variables with any type of dependency and probability distribution functions. This flexibility comes from the use of a copula model for the representation of the joint probability distribution function. The method has been mainly implemented through a non-parametric approach using Bernstein copulas and it has been successfully applied for the simulation of petrophysical properties using elastic seismic attributes as secondary variables. In the present work, as an alternative, it is implemented through a parametric approach, specifically, using the Archimedean copulas. First, the parametric approach is validated in a published case and then it is applied to a case study. Both approaches are compared in terms of accuracy and performance. The results showed the parametric approach does not reproduce data statistics and presents more uncertainty than the non-parametric approach, meanwhile the parametric approach has a low computational cost compared with the non-parametric approach.

Keywords: co-simulation, copulas, Bernstein, Archimedean, petrophysical properties, elastic seismic attributes.

3 Introduction.

One of the most common problems of reservoir geological-petrophysical modeling workflow [Cosentino, 2001] consists in predicting petrophysical properties considering the dependency relationships with the seismic attributes. For this purpose, different methods and approaches have been used, ranging from empirical relationships [Diaz-Viera et al., 2006], regression methods, neural networks [Parra et al., 2014], [Iturraran-Viveros and Parra, 2014] to spatial stochastic models. The latter being more flexible since they better reproduce the statistical and spatial behavior of petrophysical properties [Pyrcz and Deutsch, 2014].

In the last decade, various spatial stochastic simulation methods, also known as geostatistical simulation methods, have been developed. Among the most used are: Monte Carlo simulation [Bosch et al., 2007], [Oh and Kwon, 2001] and [Grana, 2014], sequential indicator simulation [Caers et al., 2000], direct simulation [Azevedo and Soares, 2017] or sequential Gaussian simulation whose methods are widely developed in the works of [Chiles and Delfiner, 1999], [Dubrule, 2003], [Bortoli et al., 1993] and [Pyrcz and Deutsch, 2014] . The disadvantage of most of these methods is that they assume that the data follow Gaussian distributions and that the dependency relationships between them are linear, which makes their application to real problems very limited. In the best case, it is forced to be applied by data transformation, which produces biased results when they are back transformed.

The copula-based spatial stochastic cosimulation method was proposed by [Diaz-Viera et al., 2017] as an alternative to the aforementioned simulation methods. This method can be basically divided into two steps. Firstly, a dependence model between the primary and secondary variables is established by estimating and modeling the joint cumulative probability distribution function (CPDF) using a copula. The CPDF model is used in conjunction with a variogram (spatial correlation) model to simulate the primary variable using the second one as a conditioning variable. This last step can be done in a global optimization framework using one method, such as simulated annealing, but other methods, such as genetic algorithms, can also be applied.

Recently, the copula-based simulation method was successfully applied for the prediction of petrophysical properties using seismic attributes as a secondary variable in [Le, 2021] , [Le et al., 2020] , [Díaz-Viera et al., 2018] , [Vázquez, 2018] . But in these works, the method was implemented using a non-parametric approach with Bernstein copulas. The non-parametric approach has the disadvantage that it requires a high computational cost, especially when calculating the joint probability distribution function estimated with the Bernstein copula.

In this work it is proposed to use a parametric copula model, through the implementation of Archimedean copula family. This approach is expected to reduce the computational cost. The paper aim is to compare the parametric and non-parametric approaches in terms of precision and performance. The comparative study is carried out through the application to two case studies in a marine reservoir in the Gulf of Mexico, where the total porosity and the density are simulated by means of the acoustic impedance and the P-wave velocity as secondary variables, respectively.

The paper has the following structure. First, the copula-based spatial stochastic co-simulación method is briefly described, distinguishing between non-parametric and parametric approaches. In particular, the members of Archimedean copula family are defined. An outline of the method application workflow is provided. Subsequently, the validation of the method is carried out by applying it to a previously published case study and the two approaches are compared. Finally, the parametric approach is applied to a new case study and the conclusions are given. In appendices, the definition of copula and its basic properties, as well as, the Bernstein copula simulation method description are shown.

4 Copula-based spatial stochastic co-simulation.

Copula-based spatial stochastic co-simulation had its origins in the works of [Erdely et al., 2012], [Díaz-Víera and Casar-González, 2005] and [Diaz-Viera et al., 2017]. The method has the advantages of not requiring linear dependence or a specific type of distribution and has been successfully applied to simulate petrophysical properties using seismic attributes as a secondary variable. The method has been implemented mainly by a non-parametric approach using Bernstein copulas, where the copula is estimated using the empirical copula and the variables are interpolated using Bernstein polynomial function (Appendix 10.2).

However, there are two more approach that have not been used in the dependency model estimation: semi-parametric and parametric approach. The semi-parametric approach estimates margins non-parametrically using empirical distribution functions and parametric copula is estimated [Jaworski et al., 2010] or the margins are estimated using parametric functions and the copula is estimated non-parametrically. The parametric approach use different parametric distribution functions are selected that best fit the petrophysical property or seismic attribute and then couple them within the parametric copula.

The difference between the non-parametric and parametric copula is the estimation, the non-parametric find to interpolate all the samples and reduce the variance in the empirical estimation; in parametric copula the samples are fitted into parametric distribution functions, the fit may skip some samples but will allow to represent the variable with a function that is easier to analyze. Therefore, if generate the joint distribution function that allows relating two or more petrophysical properties and seismic attributes in terms of their dependence is possible obtain more consistent simulations with their data sample statistics without having to assume a specific distribution or dependency. In the following subsections analyze the parametric joint distribution functions, es-

pecially the Archimedean copula family.

4.1 Parametric copula approach

The parametric copulas are a classification of copula whose advantage is use the parameters of the joint function to control the dependence. There are three great parametric copula families: elliptical, Archimedean and extreme values [Joe, 2014]. The elliptical copulas are recommended when the dependence model is derivate from elliptical distributions and the symmetry should be elliptical respect to the diagonal [Shemyakin and Kniazev, 2017]. Gaussian and t-copula are examples of elliptical copulas. The extreme value copulas are based on random vectors distributed according to multivariate extreme value distributions. Galambos, Hustle-Reis and Tawn are examples of extreme values copulas [Hofert et al., 2019].

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

4.1.1 Definition of Archimedean copulas.

Now is necessary study a class of copulas that will be useful for the development of Copula-based spatial stochastic co-simulation: Archimedean copulas. These copulas are very useful alternative because they are easy to build and there is a wide variety of families of copulas with interesting properties [Nelsen, 2006]:

Definition 4.1.1 Let φ be a continuous, strictly decreasing function of $\mathbf{I}$ on the interval $[0, \infty]$ such that $\varphi(1) = 0$, the **pseudo-inverse** of φ is the function $\varphi^{[-1]}$ with $Dom\varphi^{[-1]} = [0, \infty]$ y $Ran\varphi^{[-1]} = \mathbf{I}$ given by:

$$\varphi^{[-1]}(t) = \begin{cases} \varphi^{[-1]}(t) & Si \quad 0 \leq t \leq \varphi(0) \\ 0 & Si \quad \varphi(0) \leq t \leq \infty \end{cases}$$

Lemma 4.1.2 Let φ be a continuous function, strictly decreasing from $\mathbf{I}$ to $[0, \infty]$ such that $\varphi(1) = 0$ and $\varphi^{[-1]}$ the pseudo-inverse of (φ). Let C be the function from $(\mathbf{I}^2)$ to $\mathbf{I}$ given by

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

Then $C(u, v)$ satisfies the boundary conditions for a copula

$$C(u, 0) = 0 = C(0, v); \qquad C(u, 1) = u, C(1, v) = v$$

Lemma 4.1.3 Let φ, $\varphi^{[-1]}$ and C satisfy the hypothesis of lemma 4.1.2, then C is 2-increasing if and only if for all $\mathbf{I}$ when $u_1 \leq u_2$, therefore

$$C(u_2, v) - C(u_1, v) \leq u_2 - u_1$$

Theorem 4.1.4 Let φ be a continuous and strictly decreasing function from $\mathbf{I}$ to $[0, \infty]$ such that $\varphi(1) = 0$ and $\varphi^{[-1]}$ is the pseudo-inverse of (φ), so the function C $(\mathbf{I}^2)$ a $(\mathbf{I})$ given by lemma 4.1. 3 is a copula if and only if φ is convex.

Theorem 4.1.5 Let C be an Archimedean copula with generator φ, then:

- C is symmetric, for example C(u,v)=C(u,v) for all u, v in $\mathbf{I}$
- C is associative, for example $C(C(u,v),w) = C(u, C(v,w))$ for all u, v, w in $\mathbf{I}$
- If c> 0 is any constant, then $c\varphi$ is also a generator of C.

Properly said, the equation is the **Archimedean copula**. The function φ is called *copula generator*. Take into account that if $\varphi(0) = \infty$ it is said that φ is strictly generator, in that case $\varphi^{[-1]} = \varphi^{-1}$ and $C(u,v) = \varphi^{[-1]}(\varphi(u) + \varphi(v))$ and is said to be strictly an Archimedean copula.

Another important property is the Archimedean density copula, this can be expressed through the generator and its derivatives as [Shemyakin and Kniazev, 2017] :

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

Clayton copula. This family of copulas has been used when the dependence of the tails is strong on the left and weak on the right, as shown in the figure. Generally this type of copula does not accept negative dependencies, the copula parameter θ is restricted into the region $(0, \infty)$, if the value of θ is zero, then the marginals are they become independent [Trivedi and Zimmer, 2007].

Some members of Clayton copula can be implemented in models with negative dependencies, however, its use is not suggested under those conditions because the copula is not strict and it can violate the region restriction; reflecting the shape of the countermonotocity copula as the value of θ approaches -1, which itself is not an Archimedean copula. It cannot properly capture the negative dependency and to estimate the value of the parameter θ special conditions are needed to guarantee that the copula is strict [Cooray, 2018].

The general Clayton copula is [Trivedi and Zimmer, 2007]:

$$C_\theta(u,v) = (u^{-\theta} + v^{-\theta} - 1)^{\frac{-1}{\theta}} \qquad \theta \in (0;\infty)$$

The Clayton copula that can be evaluated in negative dependencies is defined as [Shemyakin and Kniazev, 2017]:

$$C_\theta(u,v) = max\left\{(u^{-\theta} + v^{-\theta} - 1)^{\frac{-1}{\theta}}, 0\right\} \qquad \theta \in [-1;0) \cup (0;\infty) \tag{3}$$

Whose generator is:

$$\varphi(t) = \frac{1}{\theta}\left(t^{-\theta} - 1\right) \tag{4}$$

Its pseudo-inverse is:

$$\varphi^{[-1]}(s) = max\left\{(1 + \theta s)^{\frac{-1}{\theta}}, 0\right\}$$

Frank copula. This family of copulas is very popular because it can be used in models with positive and negative dependencies between their marginals, it is a symmetric copula on both tails and includes both Frechet limits throughout the region $(-\infty, \infty)$. The disadvantage of Frank's copula is that the dependency tends to be weak in the tails and strong in the central part of the marginals; therefore, it is recommended to use this type of copula if the dependency model exhibits tail with low dependency [Trivedi and Zimmer, 2007].

The Frank copula is defined as [Shemyakin and Kniazev, 2017]:

$$C_\theta(u,v) = -\frac{1}{\theta} ln\left(1 + \frac{\left(e^{-\theta u} - 1\right)\left(e^{-\theta v} - 1\right)}{\left(e^{-\theta} - 1\right)}\right) \qquad \theta \neq 0 \tag{5}$$

The generator is :

$$\varphi(t) = -ln\frac{e^{-\theta t} - 1}{e^{-\theta} - 1} \tag{6}$$

Its pseudo-inverse is :

$$\varphi^{[-1]}(s) = -\frac{1}{\theta}ln\left[1 + e^{-s}\left(e^{-\theta} - 1\right)\right]$$

The density copula is:

$$c_\theta(u,v) = \frac{\theta\left(1 - e^{-\theta}\right)e^{-\theta(u+v)}}{\left(e^{-\theta} - 1 + \left(e^{-\theta u} - 1\right)\left(e^{-\theta v} - 1\right)\right)^2}$$

Gumbel-Hougaard copula. This family of copulas, like Clayton's copula, do not allow negative dependencies. The parameter θ is restricted to the interval $[1, \infty)$, if the value is 1 the copula is considered independent, if the value of θ is ∞ it is a countermonotocity copula or upper Frechet limit.

Gumbel copula exhibits strong dependence on the tails located on the right and weak on the left tails as shown in the figure, therefore, it is recommended when the highest values of the sample have strong dependence, while the low values have less dependency [Trivedi and Zimmer, 2007].

The Gumbel-Hougaard copula is defined as [Shemyakin and Kniazev, 2017]

$$C_\theta(u,v) = exp\left(-\left(\left(-ln(u)^\theta\right) + \left(-ln(v)^\theta\right)\right)^{\frac{1}{\theta}}\right), \qquad \theta \geq 1 \tag{7}$$

With generator:

$$\varphi(t) = (-ln(t))^{\theta} \tag{8}$$

Its pseudo-inverse :

$$\varphi^{[-1]}(s) = e^{-s^{\frac{1}{\theta}}}$$

The copula density function is:

$$c_{\theta}(u,v) = (uv)^{-1}(ln(u)\cdot ln(v))^{\theta-1}\left(w^{\frac{2}{\theta-2}} + (\theta-1)w^{\frac{1}{\theta-2}}\right)C_{\theta}(u,v), \qquad w = (-ln(u))^{\theta} + (-ln(v))^{\theta}$$

Joe copula. This copula family is based on Sibuya distribution LT [Joe, 2014]. Only works with positive dependence and the goodness of fit is reasonably well when is applied to small samples with moderate dependence [Hofert et al., 2019].

The Joe copula is

$$C_{\theta}(u,v) = 1 - \left((1-u)^{\theta} + (1-v)^{\theta} - (1-u)(1-v)^{\theta}\right)^{\frac{1}{\theta}} \tag{9}$$

And the generator is

$$\varphi(t) = -ln(1-(1-t)^{\theta}) \tag{10}$$

5 Method application workflow

The workflow follows the next steps:

- Exploratory data analysis: this step study the variable using statistical properties, histograms and scatterplot.
- Variographic analysis: this step consists in estimating and modelling a spatial correlation function, as the variogram, from a random function sample [Díaz-Viera, 2002].
- Marginal estimation and model fitting: for this step the variable is fitting using parametric probability functions, the best option is selected considering the result of loglikelihood, Akaike information criteria (AIC) and Bayesian information criteria (BIC). In case of (AIC) and (BIC) criteria, the parametric function with the lowest value will be the best fit, while the loglikelihood with the highest criterion value is the best fit. However, it is advisable to use graphical tools to assess whether the fit is adequate, therefore, the fit criteria must be confirmed using the histogram, the cumulative histogram and the cumulative distribution function.
- Parametric copula estimation and model fitting: This step using the results obtained in step 3 and the bivariate samples to estimate and select the best Archimedean copula model. The estimation is developed using maximum likelihood (MLE), Kendall inversion τ (ITau) and Spearman inversion ρ (IRho) [Hofert et al., 2019].
- Well-log simulation using the joint probability distribution function using simulated annealing.
- Compare the statistics and the spatial distribution between the original variables (step 1) and the simulated variables obtained in step 5.

This algorithm is implemented using the RGEOSTAD tools [Díaz-Viera et al., 2021] and GSLIB [Deutsch and Journel, 1998].

6 Method validation.

To implement the stochastic simulation using parametric copulas uses the lakach-1 geophysical well logs. This well crosses the Miocene turbiditic system. This system is important due to unassociated fields discovered in this unit, the unit is divided into three sections: Lower Miocene, Middle Miocene and Upper Miocene. The age of interest of this implementation is the Lower Miocene, in this zone the flow goes from southwest to northeast with some dominant channel systems. The rocks are medium-grained sandstones; Evidence of volcanic material, feldspars, quartz and some metamorphic fragments were also found. The rock is poorly consolidated and mineralogically immature, the porosity ranges from 12 to 28% [Arreguin-Lopez et al., 2011]. The interval to be evaluated is from 3035 to 3404.5 meters with a sampling interval of 0.1 meters.

The packages and functions that are used for the implementation are the following:

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


#### Load Packages ####
library(Rcpp)
library(maps)
```

```r
library(mapproj)
library(actuar)
library(fields)
library(fitdistrplus)
library(geoR)
library(gstat)
library(MASS)
library(moments)
library(poweRlaw)
library(RFOC)
library(spatstat)
library(ADGofTest)
library(reshape)
library(sp)
library(car)
library(copula)
```

[25]:

```r
#function_dir- functions directory
root_dir<-getwd()

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

setwd(function_dir)

source("AllModel.R", encoding='ISO-8859-1')
source("BasicStats.R", encoding='ISO-8859-1')
source("BestModel.R", encoding='ISO-8859-1')
source("BestModel.R")
source("BestModelName.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("Distance.R")
source("Estadisticas.R")
source("EyeModel.R", encoding='ISO-8859-1')
source("FitDistribution.R", encoding='ISO-8859-1')
source("GDEspacial.R", encoding='ISO-8859-1')
source("GDirecciones.R", encoding='ISO-8859-1')
source("GNormal.R", encoding='ISO-8859-1')
source("hist2.R")
source("HistBoxplot.R")
source("HistModel.R")
source("KrigingOrd.R", encoding='ISO-8859-1')
```

```r
source("KrigingOrdAnis.R", encoding='ISO-8859-1')
source("LogPlot.R")
source("MedianReg.R")
source("ModelVariogram.R")
source("Modelo.R")
source("Outliers.R")
source("OutliersCount.R")
source("OutliersCountTwo.R")
source("OutliersPos.R")
source("OutliersTwo.R")
source("PPplot.R")
source("QQplot.R")
source("RangoParams.R")
source("Regresion.R")
source("ScatterPlot.R")
source("Tendencia.R")
source("Transformacion.R")
source("Trend.R")
source("Val_Estadisticos.R", encoding='ISO-8859-1')
source("Validacion.R", encoding='ISO-8859-1')
source("ValidacionCross.R", encoding='ISO-8859-1')
source("Variograma.R")
source("Variograma4D.R", encoding='ISO-8859-1')
source("hist2.R")
source("scaterplot.R")
source("scaterplotReg.R")
source("Correlacion.R", encoding='ISO-8859-1')

source("copula.Bernshtein.emp.R")
source("cv.du.R")
source("dcopula.Bernshtein.emp.R")
source("depurar.xy.R")
source("dFn.inv.Bernshtein.R")
source("Fn.Bernshtein.aux.R")
source("Fn.Bernshtein.R")
source("Fn.emp.R")
source("Fn.inv.Bernshtein.R")
source("functions.aux.R")
source("genmat.copem.Bernshtein.R")
source("genmat.copem.R")
source("genmat.dcopem.Bernshtein.R")
source("genmat.densidad.Bernshtein.R")
source("simula.Bernshtein.condicional.R")
source("simula.Bernshtein.R")
source("simula.Bernshtein1.R")
source("simula.copula.Bernshtein.R")
source("transf.muestra.copula.R")
```

```
source("PLot.ScatterplotCBRegs.R")

setwd(root_dir)
```

6.1 Data selection.

We need to select the geophysical well logs: Density (RHOB) ρ (g/cm^3), sonic log (DTCO) (us/ft), total porosity (ϕ_t), effective porosity (ϕ_e), Acoustic impedance (Ip) $(m/s)(g/cm^3)$ and P velocity (Vp)

[3]:
```
Data_File <- read.csv(file='data/Lakach-1_3035.0-3404.5_25-05-2017.
 ↪csv',header=T,na.strings="-999.25")   #open file well_f03-4_502.5_1859.55.csv
#the select file is ip_Phie.csv
attach(Data_File)
```

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

    Ip, Vp
```

[4]:
```
Depth<-Data_File$DEPTH
rho<-Data_File$RHOB              #Density
VCL<-Data_File$VCL               #Clay volume
Phit<-Data_File$PHIT             #Total porosity
Phie<-Data_File$PHIE             #Effective porosity
Ip<-Data_File$Ip                 #Acoustic impedance
Vp<-Data_File$Vp                 #P velocity
```

The well-logs are:

[3]:
```
LogPlot(depth=Depth, n_logs=3,
        log_list=list(rho,VCL,Phit),
       ␣
 ↪log_name_list=list(expression(paste(rho)),"VCL",expression(paste(varphi[t]))),
        log_xlabel_list=list(expression(paste(" Density ",rho (g/
 ↪cm^3))),"VCL(dec)",expression(paste(" Total porosity ",varphi[t], " (dec)"))),
        plot_mean=TRUE, plot_median=TRUE)
```

[8]:
```
LogPlot(depth=Depth, n_logs=3,
        log_list=list(Phie,Ip,Vp),
        log_name_list=list(expression(paste(varphi[e])),"Ip","Vp"),
        log_xlabel_list=list(expression(paste(" Efective porosity ",varphi[e], "␣
↪(dec)")),expression(paste(" Ip ",(m/s)(g/cm^3))),"Vp (m/s)"),
        plot_mean=TRUE, plot_median=TRUE)
```

6.2 Exploratory data analysis.

As part of the process prior to selecting which are the best random variables, is needed to know the statistical and their dependency values.

[7]:
```
Depth_Stat<-Estadisticas(Depth)
rho_Stat<-Estadisticas(rho)              #density
VCL_Stat<-Estadisticas(VCL)              #Clay volume
Phit_Stat<-Estadisticas(Phit)            #total porosity
Phie_Stat<-Estadisticas(Phie)            #effective porosity
Ip_Stat<-Estadisticas(Ip)                #acoustic impedance
Vp_Stat<-Estadisticas(Vp)                #P velocity
```

Graph the histograms and analyze their results with their respective statisticians and estimate the Sturges value to determine the number of bins in the histogram.

[10]:
```
ns = nclass.Sturges(Ip)
ns
```

13

[5]:
```
HistBoxplot(rho,nbins = ns, mean = rho_Stat[5,2], median = rho_Stat[4,2], main␣
↪="",
            xlab = expression(paste(" Density ",rho (g/cm^3))), ylab = "Absolute␣
↪Frequency (count)", AbsFreq = TRUE, PercentFreq = FALSE )
rho_Stat
```

	Statistics <chr>	Values <dbl>
muestras	n	3696.0000
minimos	Minimum	2.0085
cuantiles1	1st. Quartile	2.1835
medianas	Median	2.2396
medias	Mean	2.2495
cuantiles3	3rd. Quartile	2.3062
maximos	Maximum	2.6031
rangos	Rank	0.5946
rangosInt	Interquartile Rank	0.1227
varianzas	Variance	0.0087
desvs	Standard Deviation	0.0934
CVs	Variation Coeff.	0.0415
simetrias	Skewness	0.6632
curtosiss	Kurtosis	3.5973

With the case of the density log (ρ), can see that the difference between the mean and the median is 0.0099, which apparently gives a slight positive asymmetry, however, the boxplot shows a strong amount of values outliers, especially to the right of the boxplot.

[6]:
```
HistBoxplot(VCL,nbins = ns, mean = VCL_Stat[5,2], median = VCL_Stat[4,2], main␣
↪="",
            xlab = "VCL (dec)", ylab = "Absolute Frequency (count)", AbsFreq =␣
↪TRUE, PercentFreq = FALSE )
sonic_Stat
```

	Statistics <chr>	Values <dbl>
muestras	n	3696.0000
minimos	Minimum	66.1380
cuantiles1	1st. Quartile	93.1207
medianas	Median	101.2140
medias	Mean	100.2340
cuantiles3	3rd. Quartile	109.4387
maximos	Maximum	124.5048
rangos	Rank	58.3668
rangosInt	Interquartile Rank	16.3180
varianzas	Variance	139.6124
desvs	Standard Deviation	11.8158
CVs	Variation Coeff.	0.1179
simetrias	Skewness	-0.5596
curtosiss	Kurtosis	2.9606

The statistical analysis of the clay volume log (VCL) shows that the difference between the mean and the median is -0.98, which indicates that have negative asymmetry. The boxplot shows a greater number of outliers to the left of the graph.

[7]:
```
HistBoxplot(Phit,nbins = ns, mean = Phit_Stat[5,2], median = Phit_Stat[4,2],␣
↪main ="",
            xlab = expression(paste(" Total porosity ",varphi[t], " (dec)")),␣
↪ylab = "Absolute Frequency (count)", AbsFreq = TRUE, PercentFreq = FALSE )
Phit_Stat
```

	Statistics <chr>	Values <dbl>
muestras	n	3696.0000
minimos	Minimum	0.0620
cuantiles1	1st. Quartile	0.2147
medianas	Median	0.2295
medias	Mean	0.2220
cuantiles3	3rd. Quartile	0.2414
maximos	Maximum	0.2939
rangos	Rank	0.2319
rangosInt	Interquartile Rank	0.0267
varianzas	Variance	0.0011
desvs	Standard Deviation	0.0338
CVs	Variation Coeff.	0.1521
simetrias	Skewness	-1.8110
curtosiss	Kurtosis	6.9970

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

[8]:
```
HistBoxplot(Phie,nbins = ns, mean = Phie_Stat[5,2], median = Phie_Stat[4,2],␣
↪main ="",
            xlab = expression(paste(" Effective porosity ",varphi[e], "␣
↪(dec)")), ylab = "Absolute Frequency (count)", AbsFreq = TRUE, PercentFreq =␣
↪FALSE )
Phie_Stat
```

	Statistics <chr>	Values <dbl>
muestras	n	3696.0000
minimos	Minimum	0.0379
cuantiles1	1st. Quartile	0.1395
medianas	Median	0.1650
medias	Mean	0.1631
cuantiles3	3rd. Quartile	0.1893
maximos	Maximum	0.2936
rangos	Rank	0.2557
rangosInt	Interquartile Rank	0.0498
varianzas	Variance	0.0013
desvs	Standard Deviation	0.0357
CVs	Variation Coeff.	0.2189
simetrias	Skewness	-0.3028
curtosiss	Kurtosis	3.3281

Analyzing the statistics of the effective porosity log, see that the difference between the mean and the median is -0.0019, which is very low. Looking at the histogram is notice an apparent balance in the number of outliers.

[9]:
```
HistBoxplot(Ip,nbins = ns, mean = Ip_Stat[5,2], median = Ip_Stat[4,2], main ="",
            xlab = expression(paste(" Ip ",(m/s)(g/cm^3))), ylab = "Absolute␣
 ↪Frequency (count)", AbsFreq = TRUE, PercentFreq = FALSE )
Ip_Stat
```

	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	1.5617
curtosiss	Kurtosis	5.7657

For the case of the variable Ip, the difference between the mean and the median is 158.7265, which indicates that it has positive skewness, which is possibly caused by the large number of outliers located to the right of the graph.

[10]:
```
HistBoxplot(Vp,nbins = ns, mean = Vp_Stat[5,2], median = Vp_Stat[4,2], main ="",
            xlab = "Vp (m/s)", ylab = "Absolute Frequency (count)", AbsFreq =␣
 ↪TRUE, PercentFreq = FALSE )
Vp_Stat
```

	Statistics <chr>	Values <dbl>
muestras	n	3696.0000
minimos	Minimum	2448.0984
cuantiles1	1st. Quartile	2785.1214
medianas	Median	3011.4426
medias	Mean	3088.0889
cuantiles3	3rd. Quartile	3273.1713
maximos	Maximum	4608.5458
rangos	Rank	2160.4474
rangosInt	Interquartile Rank	488.0498
varianzas	Variance	165004.4922
desvs	Standard Deviation	406.2074
CVs	Variation Coeff.	0.1315
simetrias	Skewness	1.2393
curtosiss	Kurtosis	4.5800

And finally, the variable Vp has a difference between the mean and the median of 76.6463, which indicates that the asymmetry is positive, the outliers are located to the right of the graph.

Now can see the full table of statistics (Table 1)

[11]:
```
par_Stat <- Val_Estadisticos(Data_File[,c(15,10,12,13,23,25)])
par_Stat
```

	VCL <dbl>	RHOB <dbl>	PHIT <dbl>	PHIE <dbl>	Vp <dbl>	Ip <dbl>
No_muestras	3696.00000	3696.00000	3696.00000	3696.00000	3696.00000	3.696000e+03
Minimo	0.00000	2.00850	0.06200	0.03790	2448.09839	5.086007e+03
Cuartil_1er	0.15820	2.18350	0.21470	0.13950	2785.12144	6.157005e+03
Mediana	0.21080	2.23955	0.22950	0.16500	3011.44261	6.809557e+03
Media	0.21911	2.24947	0.22197	0.16307	3088.08892	6.968284e+03
Cuartil_3er	0.28475	2.30620	0.24140	0.18930	3273.17128	7.321617e+03
Maximo	0.44420	2.60310	0.29390	0.29360	4608.54577	1.166146e+04
Rango	0.44420	0.59460	0.23190	0.25570	2160.44738	6.575457e+03
Rango_Intercuartil	0.12655	0.12270	0.02670	0.04980	488.04984	1.164612e+03
Varianza	0.00716	0.00873	0.00114	0.00127	165004.49215	1.310303e+06
Desv_Estandar	0.08459	0.09344	0.03376	0.03569	406.20745	1.144685e+03
Simetria	0.01678	0.66316	-1.81095	-0.30278	1.23926	1.561660e+00
Curtosis	2.57490	3.59734	6.99698	3.32805	4.57998	5.765750e+00

Now calculate the dependency measures of all the geophysical well-logs.

[10]:
```
matrix_all<-matrix(c(rho,VCL,Phit,Phie,Ip,Vp),nrow=length(rho))
colnames(matrix_all)<-c("rho","VCL","Phit","Phie","Ip","Vp")
var_labels_all<-c("rho","VCL","Phit","Phie","Ip","Vp")
pearson_table_all<-cor(matrix_all,use = "na.or.complete", method="pearson")
spearman_table_all<-cor(matrix_all,use = "na.or.complete", method="spearman")
kendall_table_all<-cor(matrix_all,use = "na.or.complete", method="kendall")
```

And the dependency values are:

[11]:
```
pearson_table_all
```

	rho	VCL	Phit	Phie	Ip	Vp
rho	1.0000000	0.2667316	-0.8652056	-0.9752970	0.7376025	0.5726283
VCL	0.2667316	1.0000000	0.2260381	-0.4191549	-0.1522825	-0.2510913
Phit	-0.8652056	0.2260381	1.0000000	0.7889350	-0.8563125	-0.7504898
Phie	-0.9752970	-0.4191549	0.7889350	1.0000000	-0.6936204	-0.5300353
Ip	0.7376025	-0.1522825	-0.8563125	-0.6936204	1.0000000	0.9748334
Vp	0.5726283	-0.2510913	-0.7504898	-0.5300353	0.9748334	1.0000000

[12]:
```
spearman_table_all
```

	rho	VCL	Phit	Phie	Ip	Vp
rho	1.0000000	0.36582027	-0.82942067	-0.9776952	0.60352140	0.3921829
VCL	0.3658203	1.00000000	0.08015421	-0.4767006	-0.04984034	-0.1988910
Phit	-0.8294207	0.08015421	1.00000000	0.7537217	-0.71071591	-0.5599701
Phie	-0.9776952	-0.47670064	0.75372169	1.0000000	-0.58433457	-0.3770040
Ip	0.6035214	-0.04984034	-0.71071591	-0.5843346	1.00000000	0.9618092
Vp	0.3921829	-0.19889096	-0.55997015	-0.3770040	0.96180919	1.0000000

[13]:
```
kendall_table_all
```

	rho	VCL	Phit	Phie	Ip	Vp
rho	1.0000000	0.28521146	-0.64486607	-0.8752851	0.43919316	0.2733108
VCL	0.2852115	1.00000000	0.04780274	-0.3863890	-0.03190214	-0.1400352
Phit	-0.6448661	0.04780274	1.00000000	0.5655579	-0.53354142	-0.4073982
Phie	-0.8752851	-0.38638898	0.56555792	1.0000000	-0.42213986	-0.2614406
Ip	0.4391932	-0.03190214	-0.53354142	-0.4221399	1.00000000	0.8341467
Vp	0.2733108	-0.14003521	-0.40739821	-0.2614406	0.83414672	1.0000000

And the scatter plot matrix is as follows:

[14]:
```
scatterplotMatrix(matrix_all,diagonal=var_labels_all)
```

```
Warning message in applyDefaults(diagonal, defaults = list(method =
"adaptiveDensity"), :
‘‘unnamed diag arguments, will be ignored’’
```

Of all the pair combinations, it was decided to use the pair (Ip, ϕ_t), since its dependency measures are:

Dependence	Original
Pearson	-0.8563
Spearman	-0.7107
Kendall	-0.5335

The (Ip, ϕ_t) geophysical well-logs are (Figure 1):

[13]:
```
LogPlot(depth=Depth, n_logs=2,
        log_list=list(Phit,Ip),
```

```
        log_name_list=list(expression(paste(varphi[t])),"Ip"),
        log_xlabel_list=list(expression(paste(" Total porosity ",varphi[t], "␣
↪(dec)")),expression(paste(" Ip ",(m/s)(g/cm^3)))),
        plot_mean=TRUE, plot_median=TRUE)
```

The (Ip, ϕ_t) scatterplot is (Figure 2):

[56]:
```
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)")))
```

	Ip (m/s)(g/cm^3)	Total porosity φ_t (dec)
Minimum	5086.00723	0.06200
1st Quartile	6157.00517	0.21470
Median	6809.55736	0.22950
Mean	6968.28386	0.22197
3rd Quartile	7321.61697	0.24140
Maximum	11661.46421	0.29390
Variance	1310302.93593	0.00114
Std. Dev.	1144.68464	0.03376

6.3 Variographic analysis

Now is must choose the variogram that best represents the variable (ϕ_t)

[12]:
```
MedianReg(Depth, Phit, Xlab= "Depth (m)", Varlab= expression(paste(" Total␣
↪porosity ",varphi[t], " (dec)")), deltay=500)
```

The first thing is to analyze the regression graph of the median, which indicates that there is a trend in the last 150 meters. So is have to check if this is consistent with the variogram.

[25]:
```
N_lags<-1500
lag_value <- 0.1
YCoord=numeric(Depth_Stat[1,2])
YCoord_Stat<-Estadisticas(YCoord)
```

[13]:
```
Phit_VarioEstimation<-Variograma(CoorX=Depth, CoorY=YCoord, Variable=Phit,
                                 Direccion=90, Tol=0, NIntervalos=N_lags,
                                 Lags=lag_value,
                                 MainTitle=expression(paste(" Total porosity␣
 ↪",varphi[t], "(dec) variogram")), xlab="Depth (m)",
```

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

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

The variogram was calculated with 1500 lags. When reviewing the variogram, no trend is noted.

[26]:
```
Phit_AllModelVarioFit<-AllModel(Depth, YCoord,
                                Phit, 0, 90, N_lags, lag_value, 1,
                                expression(paste(" Total porosity ",varphi[t],␣
 ↪"(dec) Automatic Variogram Model fitting")))
```

```
variog: computing omnidirectional variogram
variofit: covariance model used is exponential
```

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

[15]:
```
Phit_AllModelVarioFit
```

	Nugget	Sill+Nugget	Range	SCE
exponential	0.000000e+00	0.002330608	147	0.0001709277
spherical	0.000000e+00	0.001372492	147	0.0001685989
gaussian	3.964983e-05	0.002559377	147	0.0033588676

As can see in the variogram automatic adjustment; the best valid model is the spherical model.

[16]:
```
Phit_BestModelVarioFit<-BestModel(Depth, YCoord,
                                  Phit, 0, 90, N_lags, lag_value, 1,
                                  expression(paste(" Total porosity␣
 ↪",varphi[t], "(dec) Best Variogram Model fitting")))
```

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

[17]:
```
Phit_BestModelVarioFit
```

	Nugget	Sill+Nugget	Range	SCE	MaxY	MinY
spherical	0	0.001372496	147	0.0001685992	0.001181235	1.173639e-05

[18]:
```
vario_model<- 2
nugget<- 0
sill_with_nugget<- 0.0011
range <- 16
```

The autimatic model fitting is adjusted (Figure 3):

[19]:
```
Phit_EyeModelVarioFit<-EyeModel(Depth, YCoord,
                                Phit, 0, 90, N_lags, lag_value, 1,
                                vario_model, nugget, sill_with_nugget, range,
                                expression(paste(" Total porosity ",phi[t],␣
↪"(dec) Manual Variogram Model fitting")))
```

```
variog: computing omnidirectional variogram
```

Model	Nugget	Sill+Nugget	Range	MSE
sph	0.0000	0.0011	16.0000	0.0000

Validate the estimated variogram model using cross validation.

[66]:
```
Phit_CrossValid<- CrossValidation(Depth, YCoord,
                                  Phit, vario_model, nugget, sill_with_nugget,␣
↪range)
```

[67]:
```
Phit_CrossValid_Stat<- Val_Estadisticos(Phit_CrossValid[1:length(Phit),c(3,4,5)])
Phit_CrossValid_Stat
```

	Z	Z*	Z-Z*
	<dbl>	<dbl>	<dbl>
No_muestras	3696.00000	3696.00000	3696.00000
Minimo	0.06200	0.06373	-0.01216
Cuartil_1er	0.21470	0.21488	-0.00065
Mediana	0.22950	0.22930	0.00005
Media	0.22197	0.22197	0.00000
Cuartil_3er	0.24140	0.24119	0.00075
Maximo	0.29390	0.29255	0.01614
Rango	0.23190	0.22882	0.02830
Rango_Intercuartil	0.02670	0.02631	0.00140
Varianza	0.00114	0.00112	0.00000
Desv_Estandar	0.03376	0.03344	0.00134
Simetria	-1.81095	-1.83008	-0.15667
Curtosis	6.99698	7.05459	14.57014

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

[36]:
```
Phit_est<-Phit_CrossValid[,4]
Phit_est_Stat<-Estadisticas(Phit_est)
Phit_diff<-Phit_CrossValid[,5]
Phit_diff_Stat<-Estadisticas(Phit_diff)
```

[22]:
```
LogPlot(depth=Depth, n_logs=3,
        log_list=list(Phit,Phit_est,Phit_diff),
        ␣
↪log_name_list=list(expression(paste(varphi[t])),expression(paste(varphi[t],"*")),expression(p
        ␣
↪log_xlabel_list=list(expression(paste(varphi[t],"(dec)")),expression(paste(varphi[t],"*(dec)"
        plot_mean=TRUE, plot_median=TRUE)
```

[24]:
```
scaterplotReg(Phit , Phit_est, ns,
              Xmin = Phit_Stat[2,2], Xmax = Phit_Stat[7,2],
              Ymin = Phit_est_Stat[2,2],Ymax = Phit_est_Stat[7,2],
              XLAB = expression(paste(varphi[t],"(dec)")), YLAB =␣
↪expression(paste(varphi[t],"*(dec)")))
```

As see in the last Scatterplot, the estimated variogram model was able to estimate in good condition.

6.4 Parametric modeling of the marginal CDF

The (Ip, ϕ_t) samples are fitted in parametric probability distribution function that allows to represent it. To select the best option is use the Akaike information criteria (AIC), Bayesian information criteria (BIC) and loglikelihood.

[77]:
```
muestra<-as.data.frame(cbind(Ip, Phit))
muestra_depurada <- depurar.xy(muestra)
Ip_d<-muestra_depurada[,1]
Phit_d<-muestra_depurada[,2]
```

[82]:
```
spearman_original<-cor(Ip, Phit, method="spearman")
print("The Spearman coeficient of (Ip,Phit) is")
spearman_original
```

```
[1] "The Spearman coeficient of (Ip,Phit) is"
```

-0.710715909242442

[83]:
```
spearman_depurada<-cor(Ip_d, Phit_d, method="spearman")
print("The Spearman coeficient of (Ip,Phit) depurated is")
spearman_depurada
```

```
[1] "The Spearman coeficient of (Ip,Phit) depurated is"
```

-0.920629118565688

[26]:
```
plot(muestra,xlab = expression(paste(" Ip ",(m/s)(g/cm^3))),
            ylab = expression(paste(" Total porosity ",varphi[t], "␣
 ↪(dec)")),xlim=c(4500, 12000),ylim=c(0.05, 0.3))
par(new=TRUE)
plot(muestra_depurada,xlim=c(4500, 12000),ylim=c(0.05, 0.
 ↪3),col=("red"),xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend =␣
 ↪c(expression(paste(varphi[t],"-Ip")),expression(paste(varphi[t],"-Ip␣
 ↪depured"))), fill = c("black", "red"))
```

6.4.1 Total porosity CDF modeling

For this variable, the gamma, beta, normal, logistic, lognormal and weibull functions were tested.

[43]:
```
fga <- fitdist(Phit_d, "gamma", method="mle")
summary(fga)
```

```
Fitting of the distribution ' gamma ' by maximum likelihood
Parameters :
      estimate Std. Error
shape 13.99400  0.5889285
rate  67.98674  2.9130385
Loglikelihood:  1660.402   AIC:  -3316.805   BIC:  -3306.793
```

```
Correlation matrix:
          shape      rate
shape 1.0000000 0.9821971
rate  0.9821971 1.0000000
```

[44]:
```r
fb <- fitdist(Phit_d, "beta", method="mle")
summary(fb)
```

```
Fitting of the distribution ' beta ' by maximum likelihood
Parameters :
       estimate Std. Error
shape1 11.66202  0.4898396
shape2 45.08301  1.9241084
Loglikelihood:  1685.876   AIC:  -3367.752   BIC:  -3357.74
Correlation matrix:
          shape1    shape2
shape1 1.0000000 0.9733134
shape2 0.9733134 1.0000000
```

[45]:
```r
fex <- fitdist(Phit_d, "exp", method="mle")
summary(fex)
```

```
Fitting of the distribution ' exp ' by maximum likelihood
Parameters :
     estimate Std. Error
rate 4.858502    0.14629
Loglikelihood:  640.5454   AIC:  -1279.091   BIC:  -1274.085
```

[46]:
```r
fcb <- fitdist(Phit_d, "norm", method="mle")
summary(fcb)
```

```
Fitting of the distribution ' norm ' by maximum likelihood
Parameters :
      estimate  Std. Error
mean 0.2058248 0.001493919
sd   0.0496152 0.001054430
Loglikelihood:  1747.725   AIC:  -3491.45   BIC:  -3481.438
Correlation matrix:
     mean sd
mean    1  0
sd      0  1
```

[47]:
```r
fgu <- fitdist(Phit_d, "logis", method="mle")
summary(fgu)
```

```
Fitting of the distribution ' logis ' by maximum likelihood
Parameters :
           estimate   Std. Error
location 0.21049121 0.0014939208
scale    0.02833385 0.0007082399
Loglikelihood:  1735.726   AIC:  -3467.451   BIC:  -3457.44
Correlation matrix:
            location       scale
location  1.00000000 -0.08467743
scale    -0.08467743  1.00000000
```

[48]:
```
fln <- fitdist(Phit_d, "lnorm", method="mle")
summary(fln)
```

```
Fitting of the distribution ' lnorm ' by maximum likelihood
Parameters :
          estimate  Std. Error
meanlog -1.6168854 0.008615269
sdlog    0.2861255 0.006091580
Loglikelihood:  1598.547   AIC:  -3193.093   BIC:  -3183.082
Correlation matrix:
        meanlog sdlog
meanlog       1     0
sdlog         0     1
```

[49]:
```
fw <- fitdist(Phit_d, "weibull", method="mle")
summary(fw)
```

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

The goodness of fit results for ϕ_t are in next table (Table 2) shows the best option is Weibull function, followed by the normal function.

Function	Parameter	Error	Parameter	Error	Likelihood	AIC	BIC
Weibull	$\alpha = 5.1646$	0.1295	$\lambda = 0.2244$	0.0013	1791.958	-3579,917	-3569.905
Normal	$\mu = 0.2058$	0.0014	$\sigma = 0.0496$	0.0010	1747.725	-3491.45	-3481.438
Logistics	$\mu = 0.2104$	0.0014	$\beta = 0.0283$	0.0002	1735,726	-3467.451	-3457.44
Beta	$\alpha = 11.6620$	0.4898	$\beta = 45.0830$	1.9241	1685.876	-3367,752	-3357.74
Gamma	$k = 13.9940$	0.5889	$\beta = 67.9867$	2.9130	1660,402	-3316.805	-3306,793
Lognormal	$Log\mu = -1.6168$	0.0086	$Log\sigma = 0.2861$	0.0060	1598,547	-3193,093	-3183.082

[50]:
```
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    0.06386137  0.06162799  0.09194217  0.1222147
Cramer-von Mises statistic      1.89353829  1.73165175  3.16973848  5.2554615
Anderson-Darling statistic     12.77965259 15.29507992 18.90183328 30.2201784
                             5-mle-gamma 6-mle-lnorm
Kolmogorov-Smirnov statistic   0.1303951   0.1475888
Cramer-von Mises statistic     5.9748240   7.6772647
Anderson-Darling statistic    34.2348226  43.5736567

Goodness-of-fit criteria
                               1-mle-weibull 2-mle-logis 3-mle-norm 4-mle-beta
Akaike's Information Criterion     -3579.917   -3467.451  -3491.450  -3367.752
Bayesian Information Criterion     -3569.905   -3457.440  -3481.438  -3357.740
                               5-mle-gamma 6-mle-lnorm
Akaike's Information Criterion   -3316.805   -3193.093
Bayesian Information Criterion   -3306.793   -3183.082
```

As can see in the next figure (Figure 4(A)), the functions cannot cover the entire histogram, only a large part of the central area, the lower values located to the left of the histogram are omitted.

[27]:
```
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()
```

This is confirmed in the cumulative histogram in the next figure (Figure 4(B)), so it is possible that the results obtained in the simulation give greater importance to largest values and not so much to the minimum values reported during the univariate analysis.

[28]:
```
cdfcomp(list(fw,fcb,fgu,fb,fga,fln),fitlwd=3,xlab = expression(paste(" Total␣
↪porosity ",varphi[t], " (dec)")),  legendtext = plot.legend)
grid()
```

[29]:
```
denscomp(list(fw),xlab = expression(paste(" Total porosity ",varphi[t], "␣
↪(dec)")),datacol = "gray",fitlwd=3 , legendtext = "Weibull")
grid()
```

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

6.4.2 Acoustic impedance CDF modeling

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

[55]:
```
fga1 <- fitdist(Ip_d, "gamma", method="mle")
summary(fga1)
```

```
Fitting of the distribution ' gamma ' by maximum likelihood
Parameters :
          estimate   Std. Error
shape 26.558041607 4.114143e-01
rate   0.003534429 5.034362e-05
Loglikelihood:  -9586.137   AIC:  19176.27   BIC:  19186.28
```

```
Correlation matrix:
          shape      rate
shape 1.0000000 0.9275818
rate  0.9275818 1.0000000
```

[56]:
```r
fcb1 <- fitdist(Ip_d, "norm", method="mle")
summary(fcb1)
```

```
Fitting of the distribution ' norm ' by maximum likelihood
Parameters :
     estimate Std. Error
mean 7513.696   45.85103
sd   1521.673   32.40609
Loglikelihood:  -9647.394   AIC:  19298.79   BIC:  19308.8
Correlation matrix:
     mean sd
mean    1  0
sd      0  1
```

[57]:
```r
fgu1 <- fitdist(Ip_d, "logis", method="mle")
summary(fgu1)
```

```
Fitting of the distribution ' logis ' by maximum likelihood
Parameters :
          estimate Std. Error
location 7317.6550   45.17146
scale     858.5635   21.77019
Loglikelihood:  -9653.742   AIC:  19311.48   BIC:  19321.5
Correlation matrix:
          location     scale
location 1.0000000 0.1220283
scale    0.1220283 1.0000000
```

[58]:
```r
fln1 <- fitdist(Ip_d, "lnorm", method="mle")
summary(fln1)
```

```
Fitting of the distribution ' lnorm ' by maximum likelihood
Parameters :
        estimate  Std. Error
meanlog 8.905543 0.005752746
sdlog   0.191057 0.004067304
Loglikelihood:  -9562.235   AIC:  19128.47   BIC:  19138.48
Correlation matrix:
        meanlog sdlog
meanlog       1     0
```

```
sdlog         0     1
```

[59]:
```r
fw1 <- fitdist(Ip_d, "weibull", method="mle")
summary(fw1)
```

```
Fitting of the distribution ' weibull ' by maximum likelihood
Parameters :
         estimate Std. Error
shape    4.963831  0.1085294
scale 8153.505045 52.5928593
Loglikelihood:  -9710.083   AIC:  19424.17   BIC:  19434.18
Correlation matrix:
          shape     scale
shape 1.0000000 0.3415378
scale 0.3415378 1.0000000
```

[60]:
```r
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    0.110623   0.1228149   0.0969627  0.1467269
Cramer-von Mises statistic      3.864943   4.7432979   3.8754152  6.7094764
Anderson-Darling statistic     23.252379  28.0628476  31.6074815 39.0128512
                             5-mle-weibull
Kolmogorov-Smirnov statistic     0.1555284
Cramer-von Mises statistic       7.9486828
Anderson-Darling statistic      45.8826488

Goodness-of-fit criteria
                               1-mle-lnorm 2-mle-gamma 3-mle-logis 4-mle-norm
Akaike's Information Criterion    19128.47    19176.27    19311.48   19298.79
Bayesian Information Criterion    19138.48    19186.28    19321.50   19308.80
                               5-mle-weibull
Akaike's Information Criterion      19424.17
Bayesian Information Criterion      19434.18
```

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	Error	Parameter	Error	Likelihood	AIC	BIC
Lognormal	$Log\mu = 8.9055$	0.0057	$Log\sigma = 0.1910$	0.0040	-9562,235	19128.47	19138.48
Gamma	$k = 26.5580$	4.114e-01	$\beta = 0.0035$	5.034362e-05	-9586,137	19176.27	19186.28
Normal	$\mu = 7513.696$	45.8510	$\sigma = 1521.67$	32.4060	-9647,394	19298.79	19308.8
Logistic	$\mu = 7317.6550$	45.1714	$\beta = 858.5635$	21.7701	-9653,742	19311.48	19321.5
Weibull	$\alpha = 4.9638$	0.1085	$\lambda = 8153.5050$	52.5928	-9710.083	19424.17	19434.18

As ϕ_t variable, all functions fail to cover the entire histogram (Figure 5(A)). Notice that the highest values are not covered.

[93]:
```
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()
```

Notice that the highest values are not covered. If examinate the cumulative histogram (Figure 5(B)), the interval 6000 to 7000 has good approximation to empirical function.

[94]:
```
cdfcomp(list(fln1,fga1,fcb1,fgu1,fw1),xlab = expression(paste(" Ip ",(m/s)(g/
 ↪cm^3))),fitlwd=3,  legendtext = plot.legend)
grid()
```

[95]:
```
plot.legend <- c("Lognormal")
denscomp(list(fln1),datacol = "gray",xlab = expression(paste(" Ip ",(m/s)(g/
 ↪cm^3))),fitlwd=3 , legendtext = plot.legend)
grid()
```

[96]:
```
cdfcomp(list(fln1),xlab = expression(paste(" Ip ",(m/s)(g/cm^3))),fitlwd=3, ␣
 ↪legendtext = plot.legend)
grid()
```

6.5 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).

[65]:
```
data_pseudo=pobs(muestra_depurada)
n=1103
```

[66]:
```
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 1103 2-dimensional observations.
Frank copula, dim. d = 2
      Estimate Std. Error
alpha   -15.04      0.443
The maximized loglikelihood is 1004
Optimization converged
Number of loglikelihood evaluations:
function gradient
       5        5
```

[67]:
```
fc <- frankCopula(ifme_frank@copula@parameters)
```

[68]:
```
mfc <- mvdc(fc, margins = c("lnorm", "weibull"),
            paramMargins = list(list(meanlog = 8.905543, sdlog = 0.191057),
                                list(shape = 5.1646292, scale = 0.2244496)))
```

[69]:
```
Xf <- rMvdc(n, mvdc = mfc)

Xf_stat <- Val_Estadisticos(Xf[])
write.csv(Xf_stat, file = "Results/Priori_estimation/Tables/
 ↪basic_statistics_frank_copula_IFME.csv")
```

[70]:
```
Xfpobs=pobs(Xf)
```

To select the best joint distribution function, the error or standard deviation and the graphs are evaluated, the Frank's copula with the parameter estimated by maximum likelihood method (MLE)(Figure 6(B)) is the best option.

[31]:
```
cols = rev(colorRampPalette(c('darkred','red','blue','darkblue'))(25))


plot(data_pseudo, pch=16, col=("darkgoldenrod4"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Xfpobs, xlab=(""), ylab=(""), main = ("Parametric estimation with IFME"),␣
 ↪col=("springgreen4"))
par(new=TRUE)
contour(fc, dCopula,n=20,nlevels=25, xlab = expression(paste(" Ip ",(m/s)(g/
 ↪cm^3))),
     ylab = expression(paste(" Total porosity ",varphi[t], " (dec)")) ,labcex=1.
 ↪4, col=cols,xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend = c("Frank copula pseudo-observations",
                                  expression(paste(varphi[t],"-Ip␣
 ↪pseudo-observations"))), fill = c("darkgoldenrod4", "springgreen4"))
```

The parameter is estimated using the Itau moment method, which is a semi-parametric method.

[72]:
```
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 1103 2-dimensional observations.
Clayton copula, dim. d = 2
      Estimate Std. Error
alpha  -0.8673      0.006
```

[73]:
```
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 1103 2-dimensional observations.
Frank copula, dim. d = 2
      Estimate Std. Error
alpha   -15.22       0.03
```

Plot the estimated copulas.

[74]:
```
ccC <- claytonCopula(itauclay@copula@parameters)
ccFr <- frankCopula(itauFrank@copula@parameters)
```

[75]:
```
mccC <- mvdc(ccC, margins = c("lnorm", "weibull"),
             paramMargins = list(list(meanlog = 8.905543, sdlog = 0.191057),
                                 list(shape = 5.1646292, scale = 0.2244496)))
```

[76]:
```
XC <- rMvdc(n, mvdc = mccC)
XpobsC=pobs(XC)
```

Clayton's probability distribution function (Figure 6(A)) is the second option.

[32]:
```
plot(data_pseudo, pch=16, col=("springgreen4"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(XpobsC, 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(ccC, dCopula,n=20,nlevels=25, xlab=(""), ylab=("") ,labcex=1.4,␣
 ↪col=cols,xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend = c("Clayton copula pseudo-observations",
                                  expression(paste(varphi[t],"-Ip␣
 ↪pseudo-observations"))), fill = c("darkgoldenrod4", "springgreen4"))
```

For Frank's copula

[78]:
```
mccFr <- mvdc(ccFr, margins = c("lnorm", "weibull"),
            paramMargins = list(list(meanlog = 8.905543, sdlog = 0.191057),
                                list(shape = 5.1646292, scale = 0.2244496)))
```

[79]:
```
XFr <- rMvdc(n, mvdc = mccFr)
XpobsFr=pobs(XFr)
```

[33]:
```
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 = "topright", legend = c("Frank copula pseudo-observations",
                                  expression(paste(varphi[t],"-Ip␣
 ↪pseudo-observations"))), fill = c("darkgoldenrod4", "springgreen4"))
```

The parameter is estimated using the IRho moment method, which is a semi-parametric method.

[81]:
```
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 1103 2-dimensional observations.
Clayton copula, dim. d = 2
      Estimate Std. Error
alpha   -0.922      0.006
```

[82]:
```
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 1103 2-dimensional observations.
Frank copula, dim. d = 2
      Estimate Std. Error
alpha   -14.03      0.664
```

[83]:
```
crhoC <- claytonCopula(irhoClay@copula@parameters)
crhoFr <- frankCopula(irhoFrank@copula@parameters)
```

Now we graph the Clayton copula

[84]:
```
mcrhoC <- mvdc(crhoC, margins = c("lnorm", "weibull"),
            paramMargins = list(list(meanlog = 8.905543, sdlog = 0.191057),
                                list(shape = 5.1646292, scale = 0.2244496)))
```

[85]:
```
XrhoC <- rMvdc(n, mvdc = mcrhoC)
XpobsrhoC=pobs(XrhoC)
```

[34]:
```
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 = "topright", legend = c("Clayton copula pseudo-observations",
                                  expression(paste(varphi[t],"-Ip␣
 ↪pseudo-observations"))), fill = c("darkgoldenrod4", "springgreen4"))
```

For the Frank copula

[87]:
```
mcrhoFr <- mvdc(crhoFr, margins = c("lnorm", "weibull"),
            paramMargins = list(list(meanlog = 8.905543, sdlog = 0.191057),
                                list(shape = 5.1646292, scale = 0.2244496)))
```

[88]:
```
XrhoFr <- rMvdc(n, mvdc = mcrhoFr)
XpobsrhoFr=pobs(XrhoFr)
```

[35]:
```
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 ",varphi[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 = "topright", legend = c("Frank copula pseudo-observations",
                                  expression(paste(varphi[t],"-Ip␣
 ↪pseudo-observations"))), fill = c("darkgoldenrod4", "springgreen4"))
```

As can see in the following table (Table 4), 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	$\theta = -0.8673$	0.006
Clayton	Irho	$\theta = -0.922$	0.006
Frank	Itau	$\theta = -15.22$	0.03
Frank	MLE	$\theta = -15.04$	0.443
Frank	Irho	$\theta = -14.03$	0.664

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

Function	Parameter	Error	Parameter	Error	Likelihood	AIC	BIC
Weibull	$\alpha = 5.1646$	0.1295	$\lambda = 0.2244$	0.0013	1791.958	-3579,917	-3569.905

For the variable Ip.

Function	Parameter	Error	Parameter	Error	Likelihood	AIC	BIC
Lognormal	$Log\mu = 8.9055$	0.0057	$Log\sigma = 0.1910$	0.0040	-9562,235	19128.47	19138.48

The best copula is:

Copula	Method	Parameter	Error
Frank	MLE	$\theta = -15.04$	0.443

6.6 Conditional joint simulation of total porosity with acoustic impedance.

After select the best parameters for the joint probability distribution function, is simulate a considerable number of pairs (Ip,ϕ_t), it is proposed to simulate 40,000 observations based on the best estimate selected. The simulation samples under the parametric approach are very fast, less than one second.

[37]:
```
system.time(XFrF <- rMvdc(40000, mvdc = mfc))
XpobsFrF=pobs(XFrF)

write.csv(XFrF, file = "Results/Likelihood/Tables/Frank_copula_40000_obs.csv")
```

```
   user  system elapsed
  0.018   0.000   0.018
```

Considering the information in next comparative table (Table 4), the case of ϕ_t simulated and ϕ_t variable has difference in the minimum values of 0.01 and maximum 0.007; in the case of mean and median their differences are even lower. Therefore, it can be considered that the probability distribution function is representative. With the case of Ip, the difference in the minimum and maximum values have considerable differences, however, the mean and median values are very close to each other.

[38]:
```
XFrF_Ip_Stat<-Estadisticas(XFrF[,1])
XFrF_Phit_Stat<-Estadisticas(XFrF[,2])

XFrF_Stat<-Val_Estadisticos(XFrF)
```

```
write.csv(XFrF_Stat, file = "Results/Likelihood/Tables/
 ↪Frank_copula_40000_obs_Stat.csv")
```

[39]:
```
Comparative<-cbind(XFrF_Stat[1],par_Stat[6],XFrF_Stat[2],par_Stat[3])
Comparative
```

	XFrF_Stat[1] <dbl>	Ip <dbl>	XFrF_Stat[2] <dbl>	PHIT <dbl>
No_muestras	4.000000e+04	3.696000e+03	40000.00000	3696.00000
Minimo	3.000827e+03	5.086007e+03	0.02649	0.06200
Cuartil_1er	6.482617e+03	6.157005e+03	0.17626	0.21470
Mediana	7.364988e+03	6.809557e+03	0.20918	0.22950
Media	7.510479e+03	6.968284e+03	0.20661	0.22197
Cuartil_3er	8.401775e+03	7.321617e+03	0.23950	0.24140
Maximo	1.682304e+04	1.166146e+04	0.36246	0.29390
Rango	1.382221e+04	6.575457e+03	0.33596	0.23190
Rango_Intercuartil	1.919158e+03	1.164612e+03	0.06324	0.02670
Varianza	2.134060e+06	1.310303e+06	0.00211	0.00114
Desv_Estandar	1.460842e+03	1.144685e+03	0.04596	0.03376
Simetria	6.064500e-01	1.561660e+00	-0.27632	-1.81095
Curtosis	3.677430e+00	5.765750e+00	2.92066	6.99698

[41]:
```
plot(XpobsFrF, xlab = expression(paste(" Ip ",(m/s)(g/cm^3))),
        ylab = expression(paste(" Total porosity ",varphi[t], " (dec)")), main =␣
 ↪("Parametric estimation with maximum likelihood (MLE)"),␣
 ↪col=("darkgoldenrod4"))
par(new=TRUE)
contour(fc, dCopula,n=20,nlevels=25, xlab=(""), ylab=(""),labcex=1.4, col=cols␣
 ↪,xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(data_pseudo, pch=16, col=("springgreen4"),xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend = c("Frank copula pseudo-observations",
                                  expression(paste(varphi[t],"-Ip␣
 ↪pseudo-observations"))), fill = c("darkgoldenrod4", "springgreen4"))
```

[42]:
```
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 ",varphi[t], " (dec)")))
```

[43]:
```
ScatterPlot(XFrF[,1] , XFrF[,2], ns,
            Xmin = XFrF_Ip_Stat[2,2], Xmax = XFrF_Ip_Stat[7,2],
            Ymin = XFrF_Phit_Stat[2,2],Ymax = XFrF_Phit_Stat[7,2],
            XLAB = expression(paste(" Ip ",(m/s)(g/cm^3))),
        YLAB = expression(paste(" Total porosity ",varphi[t], " (dec)")))
```

In comparison, the dependency measures obtained from the simulated samples appear to be quasi linear, however this may be due to the large number of samples. Regarding the comparison in the scatterplots (Figure 7), is note that the joint probability distribution function fails to adequately sample the area located in the interval Ip (10000,12000) ϕ_t (0.05,0.15). Sampling is abundant in the central part of the original sample (red dots), which is to be expected given the properties of Frank's copula (black dots).

[48]:
```
plot(XFrF[,1] , XFrF[,2],xlim=c(3800, 16500),ylim=c(0, 0.
↪35),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Ip , Phit, xlab = expression(paste(" Ip ",(m/s)(g/cm^3))),
        ylab = expression(paste(" Total porosity ",varphi[t], "␣
↪(dec)")),xlim=c(3800, 16500),ylim=c(0, 0.35),col=("red"))
```

```
grid()
legend(x = "bottomleft", legend = c(expression(paste(varphi[t],"-Ip Frank␣
 ↪copula")),expression(paste(varphi[t],"-Ip"))), fill = c("black", "red"))
legend(
  'topright', ncol = 3L, title = 'Measure of dependences',
  legend = c(
    '','Pearson', 'Spearman','Kendall',
    'Original','-0.8563','-0.7107', '-0.5335',
    'Frank copula','-0.8888','-0.9306','-0.7661'
  )
)
```

As we can see in comparison, the dependency values are the following:

Dependence	Data sample	Simulated Parametric
Pearson	-0.8563	-0.8888
Spearman	-0.7107	-0.9306
Kendall	-0.5335	-0.7661

The dependency measures obtained from the simulated samples appear to be quasi linear, however this may be due to the large number of samples.

Regarding the comparison in the scatterplots, we can note that the joint probability distribution function fails to adequately sample the area located in the interval Ip (10000,12000) ϕ_t (0.05,0.15). Sampling is abundant in the central part of the original sample (Red dots), which is to be expected given the properties of Frank's copula.

6.7 Total porosity spatial simulation

Now is simulated the total porosity well-log ϕ_t using the simulated annealing method, for the implementation the GSLIB package [Deutsch and Journel, 1998] was used. The variogram estimated for ϕ_t (Figure 3) was used as objective function and considered 100 realizations. The results are the following: in the comparative Table 6 with all the samples selected from 100 realizations and the best realization. The minimum, maximum, mean and median values have low differences, not greater than 0.01.

[12]:
```
Data_like_FULL_100 <- read.csv(file='Results/Likelihood/Tables/100 simulaciones/
 ↪100_sim_all.csv',header=T,na.strings="-999.25")    #open file well_f03-4_502.
 ↪5_1859.55.csv
#the select file is ip_Phie.csv
attach(Data_like_FULL_100)
Data_like_FULL_100_Stat<-Val_Estadisticos(Data_like_FULL_100)

write.csv(Data_like_FULL_100_Stat, file = "Results/Likelihood/Tables/
 ↪Phit_ip_sim_Stat.csv")

Comparative_100<-cbind(Data_like_FULL_100_Stat[1],par_Stat[6],Data_like_FULL_100_Stat[2],par_Sta
Comparative_100
```

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

    Ip, Phit


The following objects are masked from Data_like_FULL_100 (pos = 3):

    Ip, Phit


The following objects are masked from Data_like_FULL_100 (pos = 4):

    Ip, Phit
```

```
The following objects are masked from Data_like_FULL_100 (pos = 6):

    Ip, Phit
```

	Ip <dbl>	Ip <dbl>	Phit <dbl>	PHIT <dbl>
No_muestras	3.696000e+05	3.696000e+03	3.6960e+05	3696.00000
Minimo	5.086007e+03	5.086007e+03	5.4070e-02	0.06200
Cuartil_1er	6.157005e+03	6.157005e+03	2.0710e-01	0.21470
Mediana	6.809557e+03	6.809557e+03	2.2844e-01	0.22950
Media	6.968284e+03	6.968284e+03	2.2442e-01	0.22197
Cuartil_3er	7.321617e+03	7.321617e+03	2.4885e-01	0.24140
Maximo	1.166146e+04	1.166146e+04	3.3382e-01	0.29390
Rango	6.575457e+03	6.575457e+03	2.7975e-01	0.23190
Rango_Intercuartil	1.164612e+03	1.164612e+03	4.1750e-02	0.02670
Varianza	1.309952e+06	1.310303e+06	1.4100e-03	0.00114
Desv_Estandar	1.144531e+03	1.144685e+03	3.7540e-02	0.03376
Simetria	1.561660e+00	1.561660e+00	-8.6440e-01	-1.81095
Curtosis	5.765750e+00	5.765750e+00	4.2437e+00	6.99698

Superimposing all realizations (Figure 8 (A)) there is dispersion in the interval 3200 to 3330 meters.

[45]:
```
Data_like_Phit <- read.csv(file='Results/Likelihood/Tables/100 simulaciones/
 ↪100_sim_Phit.csv',header=T,na.strings="-999.25")   #open file well_f03-4_502.
 ↪5_1859.55.csv
#the select file is ip_Phie.csv
attach(Data_like_Phit)
```

```
The following object is masked _by_ .GlobalEnv:

    Depth
```

[46]:
```
XFrF_Phit_Stat<-Val_Estadisticos(Data_like_Phit)
XFrF_Phit_Stat
write.csv(XFrF_Phit_Stat, file = "Results/Likelihood/Tables/Phit_100_sim_Stat.
 ↪csv")
```

	Depth <dbl>	Phit_like_1 <dbl>	Phit_like_2 <dbl>	Phit_like_3 <dbl>	Phit_like_4 <dbl>	Phit_like_5 <dbl>	Phit_like <dbl>
No_muestras	3696.0000	3696.00000	3696.00000	3696.00000	3696.00000	3696.00000	3696.000
Minimo	3035.0000	0.07481	0.07562	0.07503	0.07587	0.06753	0.06705
Cuartil_1er	3127.3749	0.20690	0.20690	0.20709	0.20689	0.20706	0.20730
Mediana	3219.7500	0.22846	0.22829	0.22836	0.22824	0.22845	0.22841
Media	3219.7500	0.22447	0.22435	0.22445	0.22434	0.22440	0.22442
Cuartil_3er	3312.1251	0.24878	0.24853	0.24906	0.24880	0.24891	0.24871
Maximo	3404.5000	0.32489	0.32740	0.32517	0.31143	0.31508	0.31843
Rango	369.5000	0.25008	0.25178	0.25014	0.23556	0.24755	0.25138
Rango_Intercuartil	184.7501	0.04189	0.04163	0.04197	0.04191	0.04185	0.04141
Varianza	11386.7600	0.00141	0.00142	0.00141	0.00140	0.00142	0.00142
Desv_Estandar	106.7088	0.03757	0.03765	0.03750	0.03745	0.03763	0.03771
Simetria	0.0000	-0.85602	-0.87085	-0.86267	-0.86860	-0.87916	-0.88625
Curtosis	1.8000	4.21385	4.26662	4.24496	4.22096	4.29005	4.33760

[47]:
```
par(mfrow = c(1,3))
plot(Data_like_Phit$Phit_like_1,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_2,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_3,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_4,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_5,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_6,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_7,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_8,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_9,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_10,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
```

```
plot(Data_like_Phit$Phit_like_11,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_12,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_13,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_14,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_15,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_16,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_17,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_18,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_19,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_20,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_21,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_22,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_23,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_24,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_25,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
```

```
plot(Data_like_Phit$Phit_like_26,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_27,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_28,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_29,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_30,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_31,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_32,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_33,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_34,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_35,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_36,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_37,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_38,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_39,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_40,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
```

```
plot(Data_like_Phit$Phit_like_41,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_42,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_43,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_44,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_45,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_46,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_47,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_48,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_49,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_50,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_51,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_52,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_53,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_54,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_55,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
```

```
plot(Data_like_Phit$Phit_like_56,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_57,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_58,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_59,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_60,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_61,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_62,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_63,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_64,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_65,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_66,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_67,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_68,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_69,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_70,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
```

```
plot(Data_like_Phit$Phit_like_71,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_72,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_73,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_74,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_75,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_76,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_77,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_78,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_79,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_80,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_81,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_82,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_83,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_84,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_85,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
```

```
plot(Data_like_Phit$Phit_like_86,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_87,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_88,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_89,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_90,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_91,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_92,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_93,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_94,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_95,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_96,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_97,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_98,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_99,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit$Phit_like_100,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
```

```
plot(Phit,Depth,xlab = expression(paste(phi[t], "␣
↪(dec)")),ylab="Depth",ylim=rev(range(Depth)),type = "l",lwd=1,col=("black"))
legend(x = "topleft", legend =␣
↪c(expression(paste(phi[t])),expression(paste(phi[t], "sim"))), fill =␣
↪c("black", "green3"))
grid()

plot(Data_like_Phit$Phit_like_88,Depth,ylim=rev(range(Depth)),type =␣
↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Phit,Depth,xlab = expression(paste(phi[t], "␣
↪(dec)")),ylab="Depth",ylim=rev(range(Depth)),type = "l",lwd=1,col=("black"))
grid()

plot((Phit-Data_like_Phit$Phit_like_89),Depth,xlab="Difference",ylab="Depth",ylim=rev(range(Dep
↪= "l",lwd=1,col=("red"))
grid()
```

7 Comparison between Frank (parametric) and Bernstein (non-parametric) based copula simulations.

Now it's time to make the comparison using the estimation Bernstein copula, this joint probability distribution function is classified as non-parametric. For its implementation is used the same pair (Ip, ϕ_t). As parametric case, also simulate 40,000 samples using the Bernstein joint distribution function (Figure 9). Each step takes around 10 hours, which compared to the time of parametric copula estimate is too much. However, the variance that obtain from this estimate is very close to original samples, this is confirmed in the Figure 9, the samples simulated using Bernstein copula (green dots) are close to data sample.

[114]:
```
muestra<-as.data.frame(cbind(Ip, Phit))
```

To reduce the computational cost, is neccesary refined the sample to be left with only non-repeating representative samples.

[115]:
```
muestra<-as.data.frame(cbind(Ip, Phit))
muestra_depurada <- depurar.xy(muestra)
X<-muestra_depurada[,1]
Y<-muestra_depurada[,2]
```

[116]:
```
tolerancia <- 1e-05
```

[117]:
```
matriz.copem <- genmat.copem(muestra)
```

[118]:
```
system.time(UV.cop<-simula.copula.Bernshtein(tam.muestra=length(X)))
```

```
    user   system  elapsed
2156.754 4436.298  981.027
```

[119]:
```
system.time(XY.sim_list<-simula.Bernshtein(tam.muestra=length(X)))
```

```
    user   system  elapsed
2169.339 4466.243  989.450
```

[120]:
```
system.time(XY.sim<-simula.Bernshtein1(tam.muestra=length(X)))
```

```
    user   system  elapsed
2152.394 4440.823  980.202
```

[121]:
```
# X simulated
X_sim<-XY.sim[,1]
# Y simulated
Y_sim<-XY.sim[,2]
```

[122]:
```
# Basic Statistics of XY simulated values
X_sim_Stat<-Estadisticas(X_sim)
Y_sim_Stat<-Estadisticas(Y_sim)
```

As can see in the following table to compare the simulation of 1103 samples, see that their statistics are very similar to the statistics of the original variables. Which shows that the model can reproduce the phenomenon even when remove the repeated samples.

[123]:
```
Comparative_be<-cbind(X_sim_Stat,Ip_Stat[,2],Y_sim_Stat[,2],Phit_Stat[,2])
Comparative_be
```

	Statistics <chr>	Values <dbl>	Ip_Stat[, 2] <dbl>	Y_sim_Stat[, 2] <dbl>	Phit_Stat[, 2] <dbl>
muestras	n	1103.0000	3696.0000	1103.0000	3696.0000
minimos	Minimum	5235.0911	5086.0072	0.0624	0.0620
cuantiles1	1st. Quartile	6176.9147	6157.0052	0.2155	0.2147
medianas	Median	6814.7321	6809.5574	0.2290	0.2295
medias	Mean	6961.9539	6968.2839	0.2222	0.2220
cuantiles3	3rd. Quartile	7350.2267	7321.6170	0.2396	0.2414
maximos	Maximum	11660.6592	11661.4642	0.2917	0.2939
rangos	Rank	6425.5681	6575.4570	0.2293	0.2319
rangosInt	Interquartile Rank	1173.3121	1164.6118	0.0240	0.0267
varianzas	Variance	1253163.4730	1310302.9359	0.0010	0.0011
desvs	Standard Deviation	1119.4478	1144.6846	0.0324	0.0338
CVs	Variation Coeff.	0.1608	0.1643	0.1458	0.1521
simetrias	Skewness	1.5790	1.5617	-1.8879	-1.8110
curtosiss	Kurtosis	5.9843	5.7657	7.6832	6.9970

In terms of the bivariate analysis the dependency measures are very close to the original ones, even the model is quite well replicated.

[125]:
```
ScatterPlot(X_sim , Y_sim, ns,
            Xmin = X_sim_Stat[2,2], Xmax = X_sim_Stat[7,2],
            Ymin = Y_sim_Stat[2,2], Ymax = Y_sim_Stat[7,2],
            XLAB = expression(paste(" Ip ",(m/s)(g/cm^3))),
        YLAB = expression(paste(" Total porosity ",phi[t], " (dec)")))
```

[112]:
```
num.repetidos <- function(x) sum(duplicated(x))
indice.repetidos <- function(x) which(duplicated(x))
valores.repetidos <- function(x) sort(unique(x[indice.repetidos(x)]))
estandarizar <- function(muestra.xy) apply(muestra.xy, 2, rank)/nrow(muestra.xy)␣
↪# calculate standardized sample
```

[113]:
```
Ip_sim<-X_sim
PhiT_sim<-Y_sim
```

[114]:
```
(Spearman.Ip_simPhiT_sim <- cor(Ip_sim, PhiT_sim, method = "spearman"))
plot(estandarizar(cbind( Ip_sim,PhiT_sim)), main = "( Ip_sim, PhiT_sim ) copula",
     sub = paste("Spearman =", round(Spearman.Ip_simPhiT_sim, 3)), xlab = "u",␣
↪ylab = "v")
```

-0.686585311152221

Now simulate 40,000 samples using the Bernstein joint distribution function.

[115]:
```
sim_size=40000
```

[3]:
```
system.time(UV.cop<-simula.copula.Bernshtein(tam.muestra=sim_size))
```

```
     user    system   elapsed
 78839.69 180977.26  33310.42
```

[4]:
```
system.time(XY.sim_list<-simula.Bernshtein(tam.muestra=sim_size))
```

```
     user    system   elapsed
```

```
 78869.27 183043.34  81790.98
```

[5]:
```
system.time(XY.sim<-simula.Bernshtein1(tam.muestra=sim_size))
```

```
    user    system   elapsed
 80854.17 186531.60  91497.40
```

As can see, each step takes around 10 hours of computation, which compared to the computation that is done by making a parametric copula estimate is too much. However, the variance obtain from this estimate is equal to the original samples.

[119]:
```
# X simulated
X_sim2<-XY.sim[,1]
# Y simulated
Y_sim2<-XY.sim[,2]
```

[120]:
```
X_sim2_Stat<-Estadisticas(X_sim2)
Y_sim2_Stat<-Estadisticas(Y_sim2)
```

Like the first simulation using the Bernstein copula, the result is satisfactory and the difference between the statistic of the original samples and the simulated samples is very low.

[121]:
```
Comparative_be2<-cbind(X_sim2_Stat,Ip_Stat[,2],Y_sim2_Stat[,2],Phit_Stat[,2])
Comparative_be2
```

	Statistics <chr>	Values <dbl>	Ip_Stat[, 2] <dbl>	Y_sim2_Stat[, 2] <dbl>	Phit_Stat[, 2] <dbl>
muestras	n	40000.0000	3696.0000	40000.0000	3696.0000
minimos	Minimum	5086.4759	5086.0072	0.0620	0.0620
cuantiles1	1st. Quartile	6158.6817	6157.0052	0.2144	0.2147
medianas	Median	6818.2062	6809.5574	0.2293	0.2295
medias	Mean	6978.2748	6968.2839	0.2217	0.2220
cuantiles3	3rd. Quartile	7337.8782	7321.6170	0.2412	0.2414
maximos	Maximum	11661.4121	11661.4642	0.2938	0.2939
rangos	Rank	6574.9362	6575.4570	0.2318	0.2319
rangosInt	Interquartile Rank	1179.1965	1164.6118	0.0268	0.0267
varianzas	Variance	1318703.2847	1310302.9359	0.0011	0.0011
desvs	Standard Deviation	1148.3481	1144.6846	0.0337	0.0338
CVs	Variation Coeff.	0.1646	0.1643	0.1521	0.1521
simetrias	Skewness	1.5525	1.5617	-1.7948	-1.8110
curtosiss	Kurtosis	5.7346	5.7657	6.8923	6.9970

[127]:
```
ScatterPlot(X_sim2 , Y_sim2, ns,
            Xmin = X_sim2_Stat[2,2], Xmax = X_sim2_Stat[7,2], Ymin =␣
↪Y_sim2_Stat[2,2],
            Ymax = Y_sim2_Stat[7,2], XLAB = expression(paste(" Ip ",(m/s)(g/
↪cm^3))),
       YLAB = expression(paste(" Total porosity ",phi[t], " (dec)")))
```

[123]:
```
Ip_sim<-X_sim2
PhiT_sim<-Y_sim2

Spearman.Ip_simPhiT_sim <- cor(Ip_sim, PhiT_sim, method = "spearman")
Pearson.Ip_simPhiT_sim <- cor(Ip_sim, PhiT_sim, method = "pearson")
plot(cbind( Ip_sim,PhiT_sim), main = "( Ip_sim, PhiT_sim )",
     sub = paste("Pearson =", round(Pearson.Ip_simPhiT_sim, 3), ", Spearman =",␣
 ↪round(Spearman.Ip_simPhiT_sim, 3)), xlab = "Ip_sim", ylab = "PhiT_sim",
     pch = 21, col = "black", bg = "Orange")
```

[124]:
```
Sim_out_File<- as.data.frame(cbind(X_sim2,Y_sim2))

write.csv(Sim_out_File, file = "Ip-PHIT_BCsim40000.csv")
```

Now do 100 simulations using the samples obtained from the Bernstein copula.

[125]:
```
Data_like_FULL_100_B <- read.csv(file='Results/non_parametric/Ip_Phit_100_all.
 ↪csv',header=T,na.strings="-999.25")    #open file well_f03-4_502.5_1859.55.csv
#the select file is ip_Phie.csv
attach(Data_like_FULL_100_B)
Data_like_FULL_100_B_Stat<-Val_Estadisticos(Data_like_FULL_100_B)

Comparative_100_B<-cbind(Data_like_FULL_100_B_Stat[1],par_Stat[6],Data_like_FULL_100_B_Stat[2],
```

```
Comparative_100_B
```

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

    Ip, Phit


The following objects are masked from Data_like_FULL_100:

    Ip, Phit


The following object is masked from Data_File:

    Ip
```

A data.frame: 13 × 4

	Ip <dbl>	Ip <dbl>	Phit <dbl>	PHIT <dbl>
No_muestras	3.696000e+05	3.696000e+03	369600.00000	3696.00000
Minimo	5.086007e+03	5.086007e+03	0.06200	0.06200
Cuartil_1er	6.157005e+03	6.157005e+03	0.21537	0.21470
Mediana	6.809557e+03	6.809557e+03	0.22950	0.22950
Media	6.968284e+03	6.968284e+03	0.22203	0.22197
Cuartil_3er	7.321617e+03	7.321617e+03	0.24127	0.24140
Maximo	1.166146e+04	1.166146e+04	0.29379	0.29390
Rango	6.575457e+03	6.575457e+03	0.23179	0.23190
Rango_Intercuartil	1.164612e+03	1.164612e+03	0.02590	0.02670
Varianza	1.309952e+06	1.310303e+06	0.00109	0.00114
Desv_Estandar	1.144531e+03	1.144685e+03	0.03297	0.03376
Simetria	1.561660e+00	1.561660e+00	-1.94593	-1.81095
Curtosis	5.765750e+00	5.765750e+00	7.43614	6.99698

[126]:
```
Data_like_Phit_B <- read.csv(file='Results/non_parametric/Phit_100_all.
 ↪csv',header=T,na.strings="-999.25")   #open file well_f03-4_502.5_1859.55.csv
#the select file is ip_Phie.csv
attach(Data_like_Phit_B)
```

```
The following object is masked _by_ .GlobalEnv:

    Depth


The following objects are masked from Data_like_Phit:

    Depth, Phit_like_1, Phit_like_10, Phit_like_100, Phit_like_11,
    Phit_like_12, Phit_like_13, Phit_like_14, Phit_like_15,
```

```
Phit_like_16, Phit_like_17, Phit_like_18, Phit_like_19,
Phit_like_2, Phit_like_20, Phit_like_21, Phit_like_22,
Phit_like_23, Phit_like_24, Phit_like_25, Phit_like_26,
Phit_like_27, Phit_like_28, Phit_like_29, Phit_like_3,
Phit_like_30, Phit_like_31, Phit_like_32, Phit_like_33,
Phit_like_34, Phit_like_35, Phit_like_36, Phit_like_37,
Phit_like_38, Phit_like_39, Phit_like_4, Phit_like_40,
Phit_like_41, Phit_like_42, Phit_like_43, Phit_like_44,
Phit_like_45, Phit_like_46, Phit_like_47, Phit_like_48,
Phit_like_49, Phit_like_5, Phit_like_50, Phit_like_51,
Phit_like_52, Phit_like_53, Phit_like_54, Phit_like_55,
Phit_like_56, Phit_like_57, Phit_like_58, Phit_like_59,
Phit_like_6, Phit_like_60, Phit_like_61, Phit_like_62,
Phit_like_63, Phit_like_64, Phit_like_65, Phit_like_66,
Phit_like_67, Phit_like_68, Phit_like_69, Phit_like_7,
Phit_like_70, Phit_like_71, Phit_like_72, Phit_like_73,
Phit_like_74, Phit_like_75, Phit_like_76, Phit_like_77,
Phit_like_78, Phit_like_79, Phit_like_8, Phit_like_80,
Phit_like_81, Phit_like_82, Phit_like_83, Phit_like_84,
Phit_like_85, Phit_like_86, Phit_like_87, Phit_like_88,
Phit_like_89, Phit_like_9, Phit_like_90, Phit_like_91,
Phit_like_92, Phit_like_93, Phit_like_94, Phit_like_95,
Phit_like_96, Phit_like_97, Phit_like_98, Phit_like_99
```

[49]:
```
XFrF_Phit_B_Stat<-Val_Estadisticos(Data_like_Phit_B)
XFrF_Phit_B_Stat
write.csv(XFrF_Phit_B_Stat, file = "Results/non_parametric/Phit_100_all_Stat.
↪csv")
```

A data.frame: 13 × 101

	Depth <dbl>	Phit_like_1 <dbl>	Phit_like_2 <dbl>	Phit_like_3 <dbl>	Phit_like_4 <dbl>
No_muestras	3696.0000	3696.00000	3696.00000	3696.00000	3696.00000
Minimo	3035.0000	0.06205	0.06210	0.06426	0.07274
Cuartil_1er	3127.3749	0.21548	0.21538	0.21546	0.21536
Mediana	3219.7500	0.22936	0.22938	0.22948	0.22962
Media	3219.7500	0.22201	0.22198	0.22206	0.22205
Cuartil_3er	3312.1251	0.24126	0.24139	0.24123	0.24127
Maximo	3404.5000	0.28728	0.29167	0.28830	0.29378
Rango	369.5000	0.22523	0.22957	0.22404	0.22104
Rango_Intercuartil	184.7501	0.02578	0.02602	0.02577	0.02591
Varianza	11386.7600	0.00110	0.00109	0.00109	0.00108
Desv_Estandar	106.7088	0.03312	0.03305	0.03296	0.03286
Simetria	0.0000	-1.95334	-1.95159	-1.93347	-1.93703
Curtosis	1.8000	7.51028	7.46146	7.33388	7.29827

[1]:
```
par(mfrow = c(1,3))
plot(Data_like_Phit_B$Phit_like_1,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_2,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_3,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_4,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_5,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_6,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_7,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_8,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_9,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_10,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_11,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_12,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_13,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_14,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_15,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
```

```
plot(Data_like_Phit_B$Phit_like_16,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_17,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_18,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_19,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_20,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_21,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_22,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_23,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_24,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_25,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_26,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_27,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_28,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_29,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_30,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
```

```
plot(Data_like_Phit_B$Phit_like_31,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_32,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_33,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_34,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_35,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_36,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_37,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_38,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_39,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_40,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_41,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_42,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_43,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_44,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_45,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
```

```
plot(Data_like_Phit_B$Phit_like_46,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_47,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_48,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_49,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_50,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_51,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_52,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_53,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_54,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_55,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_56,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_57,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_58,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_59,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_60,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
```

```
plot(Data_like_Phit_B$Phit_like_61,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_62,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_63,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_64,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_65,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_66,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_67,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_68,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_69,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_70,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_71,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_72,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_73,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_74,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_75,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
```

```
plot(Data_like_Phit_B$Phit_like_76,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_77,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_78,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_79,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_80,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_81,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_82,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_83,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_84,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_85,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_86,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_87,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_88,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_89,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_90,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
```

```
plot(Data_like_Phit_B$Phit_like_91,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_92,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_93,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_94,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_95,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_96,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_97,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_98,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_99,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_Phit_B$Phit_like_100,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Phit,Depth,xlab=expression(paste(phi[t], "␣
 ↪(dec)")),ylab="Depth",ylim=rev(range(Depth)),type = "l",lwd=1,col=("black"))
legend(x = "topleft", legend =␣
 ↪c(expression(paste(phi[t])),expression(paste(phi[t], "sim"))), fill =␣
 ↪c("black", "green3"))
grid()

plot(Data_like_Phit_B$Phit_like_89,Depth,xlab=expression(paste(phi[t], "␣
 ↪(dec)")),ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Phit,Depth,xlab=expression(paste(phi[t], "␣
 ↪(dec)")),ylab="Depth",ylim=rev(range(Depth)),type = "l",lwd=1,col=("black"))
grid()
```

```
plot((Phit-Data_like_Phit_B$Phit_like_89),Depth,xlab="Difference",ylab="Depth",ylim=rev(range(D
↪= "l",lwd=1,col=("red"))
grid()
```

Figure 9

[158]:
```
plot(XFrF[,1] , XFrF[,2],pch=20,xlim=c(3800, 16500),ylim=c(0, 0.
↪35),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(X_sim2 , Y_sim2,pch=20,xlim=c(3800, 16500),ylim=c(0, 0.
↪35),col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Ip , Phit,pch=20,xlim=c(3800, 16500),ylim=c(0, 0.35),col=("red"))
```

```r
grid()
legend(x = "bottomleft", legend = c("Phit-Ip Frank copula","Phit-Ip Bernstein␣
↪copula","Phit-Ip Original"),
       fill = c("black","green3", "red"))
legend(
  'topright', ncol = 4L, title = 'Measure of dependences',
  legend = c(
    '','Pearson', 'Spearman','Kendall',
    'Original','-0.8563','-0.7107', '-0.5335',
    'Frank copula','-0.8886','-0.9306','-0.7649',
    'Bernstein copula','-0.8481','-0.7073','-0.5303'
  )
)
```

8 Application to case study: density vs. P- wave velocity

The next example uses (Vp, ρ) well logs. The statistics (Table 12) and the histograms of Vp (Figure 11, blue histogram) and ρ (Figure 11, green histogram) show high asymmetry although the difference between median and median is low in both variables. This case has positive dependence (Figure 11), but the dependence is less compared with the (Ip,_t) case. So, use Frank, Gumbel, Clayton and Joe copula is allowed.

[15]:
```
ScatterPlot(Vp , rho, ns,
            Xmin = Vp_Stat[2,2], Xmax = Vp_Stat[7,2],
            Ymin = rho_Stat[2,2],Ymax = rho_Stat[7,2],
            XLAB = "Vp (m/s)", YLAB = expression(paste(" Density ",rho (g/
↪cm^3))))
```

	Vp (m/s)	Density ρ(g/cm³)
Minimum	2448.09839	2.00850
1st Quartile	2785.12144	2.18350
Median	3011.44261	2.23955
Mean	3088.08892	2.24947
3rd Quartile	3273.17128	2.30620
Maximum	4608.54577	2.60310
Variance	165004.49215	0.00873
Std. Dev.	406.20745	0.09344

8.1 Variographic analysis

Now calculate the variogram model for Density ρ

[16]:
```
MedianReg(Depth, rho, Xlab= "Depth (m)", Varlab= expression(paste(" Density␣
↪",rho (g/cm^3))), deltay=500)
```

[17]:
```
N_lags<-1500
lag_value <- 0.1
YCoord=numeric(Depth_Stat[1,2])
YCoord_Stat<-Estadisticas(YCoord)
```

The experimental variogram calculated has not trend evidence, now estimate the variogram model for density ρ

[21]:
```
rho_VarioEstimation<-Variograma(CoorX=Depth, CoorY=YCoord, Variable=rho,
                                Direccion=90, Tol=0, NIntervalos=N_lags,
                                Lags=lag_value,
                                MainTitle=expression(paste(" Density ",rho (g/
↪cm^3),"variogram")), xlab="Depth (m)",
                                ylab = "Semivariance")
```

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

[27]:
```
rho_AllModelVarioFit<-AllModel(Depth, YCoord,
                               rho, 0, 90, N_lags, lag_value, 1,
```

```r
                                expression(paste(" Density ",rho (g/cm^3),␣
↪"Automatic Variogram Model fitting")))
```

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

[23]:
```
rho_AllModelVarioFit
```

	Nugget	Sill+Nugget	Range	SCE
exponential	0.005695236	0.010758952	133	0.0007587733
spherical	0.005604617	0.008776371	133	0.0006681314
gaussian	0.006636447	0.010168663	133	0.0013857786

The best variogram model estimated for ρ is spherical.

[24]:
```
rho_BestModelVarioFit<-BestModel(Depth, YCoord,
                                 rho, 0, 90, N_lags, lag_value, 1,
                                 expression(paste(" Density ",rho (g/cm^3),"␣
 ↪Best Variogram Model fitting")))
```

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

[28]:
```
rho_BestModelVarioFit
```

	Nugget	Sill+Nugget	Range	SCE	MaxY	MinY
spherical	0.005604617	0.008776355	133	0.0006681301	0.009127765	0.0001285253

[36]:
```
vario_model<- 2
nugget<- 0
sill_with_nugget<- 0.0083
range <- 15
```

[37]:
```
rho_EyeModelVarioFit<-EyeModel(Depth, YCoord,
                               rho, 0, 90, N_lags, lag_value, 1,
                               vario_model, nugget, sill_with_nugget, range,
```

```
                                    expression(paste(" Density ",rho (g/cm^3),␣
↪"Manual Variogram Model fitting")))
```

```
variog: computing omnidirectional variogram
```

Model	Nugget	Sill+Nugget	Range	MSE
sph	0.0000	0.0083	15.0000	0.0014

After estimate the variogram, the parameters are: sill plus nugget 0.0083, range 15 meter and nugget 0. this model is validated using cross validation.

[38]:
```
rho_CrossValid<- CrossValidation(Depth, YCoord,
                                 rho, vario_model, nugget, sill_with_nugget,␣
↪range)
```

[39]:
```
rho_CrossValid_Stat<- Val_Estadisticos(rho_CrossValid[1:length(rho),c(3,4,5)])
rho_CrossValid_Stat
```

	Z	Z*	Z-Z*
	<dbl>	<dbl>	<dbl>
No_muestras	3696.00000	3696.00000	3696.00000
Minimo	2.00850	2.01050	-0.05433
Cuartil_1er	2.18350	2.18389	-0.00235
Mediana	2.23955	2.23962	-0.00010
Media	2.24947	2.24947	0.00000
Cuartil_3er	2.30620	2.30429	0.00227
Maximo	2.60310	2.59863	0.02186
Rango	0.59460	0.58813	0.07620
Rango_Intercuartil	0.12270	0.12040	0.00462
Varianza	0.00873	0.00849	0.00002
Desv_Estandar	0.09344	0.09213	0.00436
Simetria	0.66316	0.68313	-0.22529
Curtosis	3.59734	3.62900	10.86730

[29]:
```
HistBoxplot(x=rho_CrossValid[,5], mean = rho_CrossValid_Stat[5,3],
            median = rho_CrossValid_Stat[4,3], main ="",
            xlab =expression(paste(" Density ",rho (g/cm^3)-rho ,"*",(g/cm^3))),␣
 ↪ylab = "Frequency (%)",
            AbsFreq = FALSE, PercentFreq = TRUE )
```

[41]:
```
rho_est<-rho_CrossValid[,4]
rho_est_Stat<-Estadisticas(rho_est)
rho_diff<-rho_CrossValid[,5]
rho_diff_Stat<-Estadisticas(rho_diff)
```

[42]:
```
LogPlot(depth=Depth, n_logs=3,
        log_list=list(rho,rho_est,rho_diff),
        ␣
↪log_name_list=list(expression(paste(rho)),expression(paste(rho,"*")),expression(paste(rho-rho
        log_xlabel_list=list(expression(paste(rho (g/
↪cm^3))),expression(paste(rho (g/cm^3),"*")),expression(paste(rho (g/cm^3)-rho␣
↪(g/cm^3),"*"))),
        plot_mean=TRUE, plot_median=TRUE)
```

[44]:
```
scaterplotReg(rho , rho_est, ns,
              Xmin = rho_Stat[2,2], Xmax = rho_Stat[7,2],
              Ymin = rho_est_Stat[2,2],Ymax = rho_est_Stat[7,2],
              XLAB = expression(paste("Density",rho (g/cm^3))), YLAB =␣
↪expression(paste("Density estimated", rho (g/cm^3),"*")))
```

Evaluate de cross validation results, the variogram model is accepted. the scatterplot $\rho - \rho*$ shows measures of dependence cuasi-lineals.

8.2 Parametric modeling of the marginal CDF

This process is the same as (Ip, ϕ_t), the variables are depured and fit the results into probability functions.

[45]:
```
muestra_2<-as.data.frame(cbind(Vp, rho))
muestra_depurada_2 <- depurar.xy(muestra_2)
Vp_d<-muestra_depurada_2[,1]
rho_d<-muestra_depurada_2[,2]
```

[46]:
```
plot(muestra_2,xlab = expression(paste(" Vp ",(m/s))),
     ylab = expression(paste(" density ", rho (g/cm^3))),xlim=c(2300,␣
 ↪4800),ylim=c(2,2.7))
par(new=TRUE)
plot(muestra_depurada_2,xlim=c(2300, 4800),ylim=c(2,2.
 ↪7),col=("red"),xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend =␣
 ↪c(expression(paste(rho,"-Vp")),expression(paste(rho,"-Vp depured"))), fill =␣
 ↪c("black", "red"))
```

8.2.1 Density CDF modeling

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

[47]:
```
fga_2 <- fitdist(rho_d, "gamma", method="mle")
summary(fga_2)
```

```
Fitting of the distribution ' gamma ' by maximum likelihood
Parameters :
      estimate Std. Error
shape 455.2516  13.656299
rate  201.6026   6.050847
Loglikelihood:  1838.346   AIC:  -3672.692   BIC:  -3661.28
Correlation matrix:
          shape      rate
shape 1.0000000 0.9994509
rate  0.9994509 1.0000000
```

[48]:
```
fcb_2 <- fitdist(rho_d, "norm", method="mle")
summary(fcb_2)
```

```
Fitting of the distribution ' norm ' by maximum likelihood
Parameters :
      estimate  Std. Error
mean 2.2581655 0.002260229
sd   0.1065189 0.001597590
Loglikelihood:  1822.317   AIC:  -3640.635   BIC:  -3629.223
Correlation matrix:
     mean sd
mean    1  0
sd      0  1
```

[49]:
```
fgu_2 <- fitdist(rho_d, "logis", method="mle")
summary(fgu_2)
```

```
Fitting of the distribution ' logis ' by maximum likelihood
Parameters :
           estimate  Std. Error
location 2.25241734 0.002253402
scale    0.06072549 0.001067205
Loglikelihood:  1804.607   AIC:  -3605.213   BIC:  -3593.802
Correlation matrix:
           location      scale
location 1.00000000 0.04004445
scale    0.04004445 1.00000000
```

[50]:
```
fln_2 <- fitdist(rho_d, "lnorm", method="mle")
summary(fln_2)
```

```
Fitting of the distribution ' lnorm ' by maximum likelihood
Parameters :
         estimate   Std. Error
meanlog 0.8134542 0.0009917163
sdlog   0.0467371 0.0006998052
Loglikelihood:  1845.261   AIC:  -3686.522   BIC:  -3675.11
Correlation matrix:
        meanlog sdlog
meanlog       1     0
sdlog         0     1
```

[51]:
```
fw_2 <- fitdist(rho_d, "weibull", method="mle")
summary(fw_2)
```

```
Fitting of the distribution ' weibull ' by maximum likelihood
Parameters :
       estimate  Std. Error
shape 20.278796 0.303743496
scale  2.310839 0.002569914
Loglikelihood:  1568.417   AIC:  -3132.834   BIC:  -3121.422
Correlation matrix:
          shape     scale
shape 1.0000000 0.3388889
scale 0.3388889 1.0000000
```

[52]:
```
best_fit_rho <- gofstat(list(fw_2,fgu_2,fcb_2,fga_2,fln_2))
print("Goodness of fit results for density rho")
best_fit_rho
```

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

Goodness-of-fit statistics
                             1-mle-weibull 2-mle-logis  3-mle-norm 4-mle-gamma
Kolmogorov-Smirnov statistic    0.09347628  0.03457589  0.04390336  0.03948611
Cramer-von Mises statistic      7.77963213  0.92915634  1.57641552  1.18558499
Anderson-Darling statistic     51.77823439  8.98197425 10.91091476  8.30950236
                             5-mle-lnorm
Kolmogorov-Smirnov statistic  0.03732116
Cramer-von Mises statistic    1.01450994
Anderson-Darling statistic    7.16228947

Goodness-of-fit criteria
                               1-mle-weibull 2-mle-logis 3-mle-norm 4-mle-gamma
```

```
Akaike's Information Criterion     -3132.834   -3605.213  -3640.635   -3672.692
Bayesian Information Criterion     -3121.422   -3593.802  -3629.223   -3661.280
                                 5-mle-lnorm
Akaike's Information Criterion   -3686.522
Bayesian Information Criterion   -3675.110
```

The parametric estimation for ρ (Table 9) suggests the lognormal function, but the cumulative distribution functions in Figure 13 (B) shows all functions except Weibull function has good fit. Gamma and normal functions have likelihood, AIC and BIC very close to lognormal.

Function	Parameter	Error	Parameter	Error	Likelihood	AIC	BIC
Lognormal	$Log\mu = 0.813$	0.009	$Log\sigma = 0.04$	0.006	1845.26	-3686.52	-3675.11
Gamma	$k = 455.25$	13.65	$\beta = 201.6$	6.05	1838.346	-3672.69	-3661.28
Normal	$\mu = 2.2581$	0.002	$\sigma = 0.106$	0.0015	1822.31	-3640.63	-3629.22
Logistics	$\sigma = 2.2524$	0.002	$\beta = 0.0607$	0.001	1804.6	-3605.21	-3593.80
Weibull	$\alpha = 20.278$	0.303	$\lambda = 2.31$	0.002	1568.41	-3132.83	-3121.42

Figure 13(A)

[53]:
```
plot.legend <- c("Weibull","Logistic","Normal","Gamma","Lognormal")
denscomp(list(fw_2,fgu_2,fcb_2,fga_2,fln_2),fitlwd=3,datacol = "gray",xlab =␣
 ↪expression(paste(" Density ",rho (g/cm^3))),
         legendtext = plot.legend)
grid()
```

Figure 13(B)

```
[54]: cdfcomp(list(fw_2,fgu_2,fcb_2,fga_2,fln_2),fitlwd=3,xlab = expression(paste("␣
      ↪Density ",rho (g/cm^3))),  legendtext = plot.legend)
      grid()
```

[55]:
```
denscomp(list(fln_2),xlab = expression(paste(" Density ",rho (g/cm^3))),datacol␣
↪= "gray",fitlwd=3 , legendtext = "Lognormal")
grid()
```

[57]:
```
cdfcomp(list(fln_2),xlab = expression(paste(" Density ",rho (g/cm^3))),fitlwd=3,␣
↪ legendtext = "Lognormal")
grid()
```

8.2.2 P-wave velocity CDF modeling

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

[58]:
```
fga1_2 <- fitdist(Vp_d, "gamma", method="mle")
summary(fga1_2)
```

```
Fitting of the distribution ' gamma ' by maximum likelihood
Parameters :
         estimate   Std. Error
shape 58.24492538 1.4484855310
rate   0.01861559 0.0004635967
Loglikelihood:  -16500   AIC:  33004.01   BIC:  33015.42
```

```
Correlation matrix:
          shape      rate
shape 1.0000000 0.9937846
rate  0.9937846 1.0000000
```

[59]:
```r
fcb1_2 <- fitdist(Vp_d, "norm", method="mle")
summary(fcb1_2)
```

```
Fitting of the distribution ' norm ' by maximum likelihood
Parameters :
      estimate Std. Error
mean 3128.7366   9.117707
sd    429.6975   6.447193
Loglikelihood:  -16617.57   AIC:  33239.13   BIC:  33250.54
Correlation matrix:
     mean sd
mean    1  0
sd      0  1
```

[60]:
```r
fgu1_2 <- fitdist(Vp_d, "logis", method="mle")
summary(fgu1_2)
```

```
Fitting of the distribution ' logis ' by maximum likelihood
Parameters :
          estimate Std. Error
location 3069.3054   8.146719
scale     224.6357   4.081603
Loglikelihood:  -16523.25   AIC:  33050.51   BIC:  33061.92
Correlation matrix:
           location      scale
location 1.00000000 0.09791403
scale    0.09791403 1.00000000
```

[61]:
```r
fln1_2 <- fitdist(Vp_d, "lnorm", method="mle")
summary(fln1_2)
```

```
Fitting of the distribution ' lnorm ' by maximum likelihood
Parameters :
         estimate  Std. Error
meanlog 8.0397783 0.002725099
sdlog   0.1284271 0.001926411
Loglikelihood:  -16449.44   AIC:  32902.89   BIC:  32914.3
Correlation matrix:
        meanlog sdlog
meanlog       1     0
```

```
sdlog         0     1
```

[62]:
```r
fw1_2 <- fitdist(Vp_d, "weibull", method="mle")
summary(fw1_2)
```

```
Fitting of the distribution ' weibull ' by maximum likelihood
Parameters :
         estimate  Std. Error
shape    6.628033  0.09627245
scale 3325.878783 11.34603858
Loglikelihood:  -16903.62   AIC:  33811.24   BIC:  33822.65
Correlation matrix:
          shape     scale
shape 1.0000000 0.3459202
scale 0.3459202 1.0000000
```

[63]:
```r
best_fit_Vp <- gofstat(list(fln1_2,fga1_2,fgu1_2,fcb1_2,fw1_2))
print("Goodness of fit results for Vp")
best_fit_Vp
```

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

Goodness-of-fit statistics
                             1-mle-lnorm 2-mle-gamma 3-mle-logis 4-mle-norm
Kolmogorov-Smirnov statistic  0.09305146   0.1026936  0.08328729  0.1228673
Cramer-von Mises statistic    7.49265885   9.1495407  4.06464122 12.8334804
Anderson-Darling statistic   48.47743891  57.9904176 46.22712526 78.9965792
                             5-mle-weibull
Kolmogorov-Smirnov statistic     0.1631692
Cramer-von Mises statistic      22.0412307
Anderson-Darling statistic     128.6791267

Goodness-of-fit criteria
                               1-mle-lnorm 2-mle-gamma 3-mle-logis 4-mle-norm
Akaike's Information Criterion    32902.89    33004.01    33050.51   33239.13
Bayesian Information Criterion    32914.30    33015.42    33061.92   33250.54
                               5-mle-weibull
Akaike's Information Criterion      33811.24
Bayesian Information Criterion      33822.65
```

Now, the Vp variable goodness of fit results in Table 10 shows gamma function as best option, Logistic and lognormal are close to gamma function results, but the graphs in Figure 14 demonstrate that gamma and logistic functions have similar fitted.

Function	Parameter	Error	Parameter	Error	Likelihood	AIC	BIC
Gamma	$k = 58.244$	1.44	$\beta = 0.0186$	0.0004	-16500	33004.01	33015.42
Logistic	$\mu = 3069.30$	8.14	$\beta = 224.63$	4.081	-16523.25	33050.51	33061.92
Lognormal	$Log\mu = 8.039$	0.002	$Log\sigma = 0.12$	4.08	-16523.44	32902.89	32914.3
Normal	$\mu = 3128.73$	9.117	$\sigma = 429.69$	6.447	-16617.57	33239.13	33250.54
Weibull	$\alpha = 6.628$	0.096	$\sigma = 3325.87$	11.34	-16903.62	33811.24	33822.65

Figure 14(A)

[64]:
```
plot.legend <- c("Lognormal","Gamma", "Logistic", "Normal",   "Weibull")
denscomp(list(fln1_2,fga1_2,fgu1_2,fcb1_2,fw1_2),datacol = "gray",xlab =␣
 ↪expression(paste(" Vp ",(m/s))),fitlwd=3 , legendtext = plot.legend)
grid()
```

Figure 14(B)

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

```
[67]: plot.legend <- c("Gamma")
denscomp(list(fga1_2),datacol = "gray",xlab = expression(paste(" Vp ",(m/
 ↪s))),fitlwd=3 , legendtext = plot.legend)
grid()
```

[68]:
```
cdfcomp(list(fga1_2),xlab = expression(paste(" Vp ",(m/s))),fitlwd=3, ␣
↪legendtext = plot.legend)
grid()
```

8.3 Copula parametric modeling

Now fit the samples using the best functions obtained and four archimedean copulas: Joe, Gumbel, Frank and Clayton, in this case, the measure of dependence is positive and the estimation method is maximum likelihood.

[80]:
```
data_pseudo_2=pobs(muestra_depurada_2)
n=length(data_pseudo_2[,1])
```

[81]:
```
summary(ifme_frank_2 <- fitCopula(frankCopula(), data = data_pseudo_2, method =␣
↪"ml"))
```

```
Call: fitCopula(copula, data = data, method = "ml")
Fit based on "maximum likelihood" and 2221 2-dimensional observations.
Frank copula, dim. d = 2
      Estimate Std. Error
alpha    3.671      0.141
The maximized loglikelihood is 342.4
Optimization converged
Number of loglikelihood evaluations:
function gradient
       4        4
```

[82]:
```
summary(ifme_clayton_2 <- fitCopula(claytonCopula(), data = data_pseudo_2,␣
↪method = "ml"))
```

```
Call: fitCopula(copula, data = data, method = "ml")
Fit based on "maximum likelihood" and 2221 2-dimensional observations.
Clayton copula, dim. d = 2
      Estimate Std. Error
alpha   0.5704      0.035
The maximized loglikelihood is 170.2
Optimization converged
Number of loglikelihood evaluations:
function gradient
       7        7
```

[83]:
```
summary(ifme_gumbel_2 <- fitCopula(gumbelCopula(), data = data_pseudo_2, method␣
↪= "ml"))
```

```
Call: fitCopula(copula, data = data, method = "ml")
Fit based on "maximum likelihood" and 2221 2-dimensional observations.
Gumbel copula, dim. d = 2
      Estimate Std. Error
alpha    1.643      0.028
The maximized loglikelihood is 522.6
Optimization converged
Number of loglikelihood evaluations:
function gradient
       6        6
```

[84]:
```
summary(ifme_joe_2 <- fitCopula(joeCopula(), data = data_pseudo_2, method =␣
↪"ml"))
```

```
Call: fitCopula(copula, data = data, method = "ml")
Fit based on "maximum likelihood" and 2221 2-dimensional observations.
Joe copula, dim. d = 2
      Estimate Std. Error
alpha    2.038      0.044
The maximized loglikelihood is 550.6
```

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

[85]:
```
fc_2 <- frankCopula(ifme_frank_2@copula@parameters)
```

[86]:
```
mfc_2 <- mvdc(fc_2, margins = c("gamma","lnorm"),
            paramMargins = list(list(shape = 58.24493, rate = 0.01861559),
                                list(meanlog = 0.8134542, sdlog = 0.0467371)))
```

[87]:
```
Xf_2 <- rMvdc(n, mvdc = mfc_2)
```

[88]:
```
Xf_stat_2 <- Val_Estadisticos(Xf_2[])
Xf_stat_2
```

No_muestras	2221.00000	2221.00000
Minimo	1935.86324	1.92683
Cuartil_1er	2844.30675	2.18678
Mediana	3103.81057	2.25869
Media	3130.78532	2.26050
Cuartil_3er	3393.66446	2.33300
Maximo	4768.66855	2.57664
Rango	2832.80532	0.64981
Rango_Intercuartil	549.35772	0.14622
Varianza	163093.98885	0.01129
Desv_Estandar	403.84897	0.10626
Simetria	0.30719	0.06598
Curtosis	3.13016	2.86523

[89]:
```
Xfpobs_2=pobs(Xf_2)
```

[92]:
```
plot(data_pseudo_2, pch=16, col=("darkgoldenrod4"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Xfpobs_2, xlab=(""), ylab=(""), main = ("Parametric estimation with IFME"),␣
 ↪col=("springgreen4"))
par(new=TRUE)
contour(fc_2, dCopula,n=20,nlevels=25, xlab = expression(paste(" Vp ",(m/s))),
        ylab = expression(paste(" Density ", rho (g/cm^3))) ,labcex=1.4,␣
 ↪col=cols,xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend = c("Frank copula pseudo-observations",
                                  expression(paste(rho,"-Vp"))), fill =␣
 ↪c("darkgoldenrod4", "springgreen4"))
```

[93]:
```
cc_2 <- claytonCopula(ifme_clayton_2@copula@parameters)
```

[94]:
```
mcc_2 <- mvdc(cc_2, margins = c("gamma","lnorm"),
              paramMargins = list(list(shape = 58.24493, rate = 0.01861559),
                                  list(meanlog = 0.8134542, sdlog = 0.0467371)))
```

[95]:
```
Xc_2 <- rMvdc(n, mvdc = mcc_2)

Xc_stat_2 <- Val_Estadisticos(Xc_2[])
Xc_stat_2
```

No_muestras	2221.00000	2221.00000
Minimo	2033.31135	1.92473
Cuartil_1er	2842.23154	2.18296
Mediana	3090.11967	2.25282
Media	3116.73689	2.25591
Cuartil_3er	3384.32415	2.32724
Maximo	4587.63713	2.59145
Rango	2554.32578	0.66673
Rango_Intercuartil	542.09262	0.14428
Varianza	169437.73987	0.01126
Desv_Estandar	411.62816	0.10611
Simetria	0.29720	0.17567
Curtosis	3.05206	2.81326

[96]:
```
Xcpobs_2=pobs(Xc_2)
```

[97]:
```
plot(data_pseudo_2, pch=16, col=("darkgoldenrod4"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Xcpobs_2, xlab=(""), ylab=(""), main = ("Parametric estimation with IFME"),␣
 ↪col=("springgreen4"))
par(new=TRUE)
contour(cc_2, dCopula,n=20,nlevels=25, xlab = expression(paste(" Vp ",(m/s))),
        ylab = expression(paste(" Density ", rho (g/cm^3))) ,labcex=1.4,␣
 ↪col=cols,xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend = c("Clayton copula pseudo-observations",
                                  expression(paste(rho,"-Vp"))), fill =␣
 ↪c("darkgoldenrod4", "springgreen4"))
```

[98]:
```
cg_2 <- gumbelCopula(ifme_gumbel_2@copula@parameters)
```

[99]:
```
mcg_2 <- mvdc(cg_2, margins = c("gamma","lnorm"),
              paramMargins = list(list(shape = 58.24493, rate = 0.01861559),
                                  list(meanlog = 0.8134542, sdlog = 0.0467371)))
```

[100]:
```
Xcg_2 <- rMvdc(n, mvdc = mcg_2)
```

[101]:
```
Xcg_stat_2 <- Val_Estadisticos(Xcg_2[])
Xcg_stat_2
```

No_muestras	2221.00000	2221.00000
Minimo	1694.89354	1.88350
Cuartil_1er	2857.00677	2.19145
Mediana	3116.00137	2.25607
Media	3138.95882	2.26181
Cuartil_3er	3406.10263	2.32909
Maximo	4670.71580	2.69273
Rango	2975.82226	0.80923
Rango_Intercuartil	549.09586	0.13763
Varianza	172926.17336	0.01080
Desv_Estandar	415.84393	0.10393
Simetria	0.28483	0.24929
Curtosis	3.09933	3.47881

[102]:
```
Xcgpobs_2=pobs(Xcg_2)
```

[2]:
```
plot(data_pseudo_2, pch=16, col=("darkgoldenrod4"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Xcgpobs_2, xlab=(""), ylab=(""), main = ("Parametric estimation with␣
 ↪IFME"), col=("springgreen4"))
par(new=TRUE)
contour(cg_2, dCopula,n=20,nlevels=25, xlab = expression(paste(" Vp ",(m/s))),
        ylab = expression(paste(" Density ", rho (g/cm^3))) ,labcex=1.4,␣
 ↪col=cols,xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend = c("Gumbel copula pseudo-observations",
                                  expression(paste(rho,"-Vp"))), fill =␣
 ↪c("darkgoldenrod4", "springgreen4"))
```

[3]:
```
cj_2 <- joeCopula(ifme_joe_2@copula@parameters)
```

[4]:
```
mcj_2 <- mvdc(cj_2, margins = c("gamma","lnorm"),
              paramMargins = list(list(shape = 58.24493, rate = 0.01861559),
                                  list(meanlog = 0.8134542, sdlog = 0.0467371)))
```

[5]:
```
Xcj_2 <- rMvdc(n, mvdc = mcj_2)

Xcj_stat_2 <- Val_Estadisticos(Xcj_2[])
Xcj_stat_2
```

No_muestras	2221.00000	2221.00000
Minimo	1834.25898	1.84604
Cuartil_1er	2856.20024	2.19003
Mediana	3110.51134	2.25838
Media	3130.55739	2.25993
Cuartil_3er	3400.36947	2.33055
Maximo	4623.13901	2.61725
Rango	2788.88003	0.77122
Rango_Intercuartil	544.16923	0.14051
Varianza	162973.13711	0.01080
Desv_Estandar	403.69932	0.10392
Simetria	0.20604	0.07419
Curtosis	3.05348	3.08164

[6]:
```
Xcjpobs_2=pobs(Xcj_2)
```

Figure 15

[7]:
```
plot(data_pseudo_2, pch=16, col=("darkgoldenrod4"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Xcjpobs_2, xlab=(""), ylab=(""), main = ("Parametric estimation with␣
 ↪IFME"), col=("springgreen4"))
par(new=TRUE)
contour(cj_2, dCopula,n=20,nlevels=25, xlab = expression(paste(" Vp ",(m/s))),
        ylab = expression(paste(" Density ", rho (g/cm^3))) ,labcex=1.4,␣
 ↪col=cols,xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend = c("Joe copula pseudo-observations",
                                  expression(paste(rho,"-Vp"))), fill =␣
 ↪c("darkgoldenrod4", "springgreen4"))
```

The fit copula results show Joe copula model as best option (Table 11). Gumbel copula could be a good option, But Joe copula can cover the pseudo-observations, especially the samples in the extremes (Figure 15).

Copula	Method	Parameter	Error	Loglikelihood
Joe	MLE	$\theta = 2.038$	0.044	550.6
Gumbel	MLE	$\theta = 1.643$	0.028	522.6
Frank	MLE	$\theta = 3.671$	0.041	342.4
Clayton	MLE	$\theta = 0.5704$	0.035	170.2

8.4 Conditional joint simulation of density with P-wave velocity

Simulating 40,000 samples from Joe copula, superimposing the results (Figure 16) is evident the drop form of Joe copula samples. The measures of dependences show Pearson values have differ-

ence of 0.02, but Spearman and Kendall measures are high. Other characteristic is the scatter of Joe copula samples, simulated samples are near to original samples. Comparing the variance and standard deviation in Table 12, the values are very close, this is not the same with the validation case, this could be an advantage of positive dependence.

[13]:
```
system.time(XjF <- rMvdc(40000, mvdc = mcj_2))
XpobsjF=pobs(XjF)

write.csv(XjF, file = "Results/2case/Tables/joe_copula_40000_obs.csv")
```

```
   user  system elapsed
  0.053   0.000   0.052
```

[4]:
```
XjF_Vp_Stat<-Estadisticas(XjF[,1])
XjF_rho_Stat<-Estadisticas(XjF[,2])
```

[6]:
```
XjF_Stat<-Val_Estadisticos(XjF)
write.csv(XjF_Stat, file = "Results/2case/Tables/joe_copula_40000_obs_Stat.csv")
```

Table 12

[9]:
```
Comparative_2<-cbind(XjF_Stat[1],par_Stat[5],XjF_Stat[2],par_Stat[2])
Comparative_2
write.csv(Comparative_2, file = "Results/2case/Tables/rho_Vp_40000_comp_Stat.
 ↪csv")
```

	XjF_Stat[1] <dbl>	Vp <dbl>	XjF_Stat[2] <dbl>	RHOB <dbl>
No_muestras	40000.00000	3696.00000	40000.00000	3696.00000
Minimo	1619.79905	2448.09839	1.86308	2.00850
Cuartil_1er	2843.14011	2785.12144	2.18734	2.18350
Mediana	3112.66684	3011.44261	2.25683	2.23955
Media	3129.29699	3088.08892	2.25907	2.24947
Cuartil_3er	3398.51891	3273.17128	2.32763	2.30620
Maximo	5038.71431	4608.54577	2.73280	2.60310
Rango	3418.91526	2160.44738	0.86972	0.59460
Rango_Intercuartil	555.37880	488.04984	0.14029	0.12270
Varianza	168551.63693	165004.49215	0.01103	0.00873
Desv_Estandar	410.55041	406.20745	0.10502	0.09344
Simetria	0.24601	1.23926	0.13022	0.66316
Curtosis	3.04567	4.57998	3.04621	3.59734

[10]:
```
plot(XpobsjF, xlab = expression(paste(" Vp ",(m/s))),
     ylab = expression(paste(" Density ", rho (g/cm^3))), main = ("Parametric␣
 ↪estimation with IFME"), col=("darkgoldenrod4"))
par(new=TRUE)
contour(cj_2, dCopula,n=20,nlevels=25, xlab=(""), ylab=(""),labcex=1.4, col=cols␣
 ↪,xaxt='n',yaxt='n',ann=FALSE)
```

```
par(new=TRUE)
plot(data_pseudo_2, pch=16, col=("springgreen4"),xaxt='n',yaxt='n',ann=FALSE)
grid()
legend(x = "topright", legend = c("Joe copula pseudo-observations",
                                  expression(paste(rho,"-Vp"))), fill =␣
 ↪c("darkgoldenrod4", "springgreen4"))
```

[11]:
```
ScatterPlot(XjF[,1] , XjF[,2], ns,
            Xmin = XjF_Vp_Stat[2,2], Xmax = XjF_Vp_Stat[7,2],
            Ymin = XjF_rho_Stat[2,2],Ymax = XjF_rho_Stat[7,2],
            XLAB = expression(paste(" Vp ",(m/s))),
            YLAB = expression(paste(" Density ", rho (g/cm^3))))
```

[12]:
```
plot(XjF[,1] , XjF[,2],xlim=c(1700, 5600),ylim=c(1.8, 2.
↪9),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Vp , rho,pch=20, xlab = expression(paste(" Vp ",(m/s))),
     ylab = expression(paste(" Density ", rho (g/cm^3))),xlim=c(1700,␣
↪5600),ylim=c(1.8, 2.9),col=("red"))
grid()
legend(x = "bottomleft", legend = c(expression(paste(rho,"-Vp Frank␣
↪copula")),expression(paste(rho,"-Ip"))), fill = c("black", "red"))
legend(
  'topleft', ncol = 3L, title = 'Measure of dependences',
  legend = c(
    '','Pearson', 'Spearman','Kendall',
```

```
    'Original','0.5726','0.3922', '0.2733',
    'Joe copula','0.5557','0.5145','0.3631'
  )
)
```

8.5 Density spatial simulation

The spatial distributions obtained to 100 simulations (Figure 17) shows good covering except in the interval 3350-3400, the variance in that range is high

[15]:
```
Data_like_FULL_100_rho <- read.csv(file='Results/2case/Tables/Vp_rho_100_all.
 ↪csv',header=T,na.strings="-999.25")    #open file well_f03-4_502.5_1859.55.csv
#the select file is ip_Phie.csv
```

```
attach(Data_like_FULL_100_rho)
Data_like_FULL_100_rho_Stat<-Val_Estadisticos(Data_like_FULL_100_rho)
```

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

    rho, Vp
```

[17]:
```
Comparative_100_rho<-cbind(Data_like_FULL_100_rho_Stat[1],par_Stat[5],Data_like_FULL_100_rho_Sta
Comparative_100_rho
```

	Vp <dbl>	Vp <dbl>	rho <dbl>	RHOB <dbl>
No_muestras	369600.00000	3696.00000	3.69600e+05	3696.00000
Minimo	2448.09839	2448.09839	1.93698e+00	2.00850
Cuartil_1er	2785.12152	2785.12144	2.18104e+00	2.18350
Mediana	3011.44262	3011.44261	2.24579e+00	2.23955
Media	3088.08892	3088.08892	2.24975e+00	2.24947
Cuartil_3er	3273.17126	3273.17128	2.31275e+00	2.30620
Maximo	4608.54590	4608.54577	2.64828e+00	2.60310
Rango	2160.44751	2160.44738	7.11300e-01	0.59460
Rango_Intercuartil	488.04974	488.04984	1.31710e-01	0.12270
Varianza	164960.29236	165004.49215	9.18000e-03	0.00873
Desv_Estandar	406.15304	406.20745	9.58200e-02	0.09344
Simetria	1.23926	1.23926	3.48230e-01	0.66316
Curtosis	4.57998	4.57998	2.92202e+00	3.59734

[18]:
```
Data_like_rho_B <- read.csv(file='Results/2case/Tables/100_sim_rho.
 ↪csv',header=T,na.strings="-999.25")   #open file well_f03-4_502.5_1859.55.csv
#the select file is ip_Phie.csv
attach(Data_like_rho_B)
```

```
The following object is masked _by_ .GlobalEnv:

    Depth
```

[19]:
```
XFrF_rho_B_Stat<-Val_Estadisticos(Data_like_rho_B)
XFrF_rho_B_Stat
write.csv(XFrF_rho_B_Stat, file = "Results/2case/Tables/rho_100_all_Stat.csv")
```

	Depth <dbl>	rho_like_1 <dbl>	rho_like_2 <dbl>	rho_like_3 <dbl>	rho_like_4 <dbl>	rho_like_5 <dbl>	rho_like_6 <dbl>
No_muestras	3696.0000	3696.00000	3696.00000	3696.00000	3696.00000	3696.00000	3696.00000
Minimo	3035.0000	1.98700	2.02261	2.02436	1.99708	1.97996	2.01386
Cuartil_1er	3127.3749	2.18020	2.18083	2.18134	2.18127	2.18069	2.18034
Mediana	3219.7500	2.24628	2.24567	2.24547	2.24560	2.24554	2.24583
Media	3219.7500	2.24955	2.24982	2.24977	2.24954	2.24991	2.24977
Cuartil_3er	3312.1251	2.31321	2.31213	2.31180	2.31184	2.31269	2.31235
Maximo	3404.5000	2.56602	2.55162	2.57644	2.56351	2.57659	2.57605
Rango	369.5000	0.57902	0.52901	0.55208	0.56643	0.59663	0.56219
Rango_Intercuartil	184.7501	0.13301	0.13130	0.13046	0.13057	0.13200	0.13201
Varianza	11386.7600	0.00926	0.00920	0.00915	0.00924	0.00921	0.00924
Desv_Estandar	106.7088	0.09620	0.09592	0.09568	0.09613	0.09597	0.09611
Simetria	0.0000	0.33241	0.33826	0.36538	0.34298	0.35341	0.33328
Curtosis	1.8000	2.95354	2.89606	2.94998	2.93101	2.94795	2.91084

[21]:
```
par(mfrow = c(1,3))
plot(Data_like_rho_B$rho_like_1,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_2,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_3,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_4,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_5,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_6,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_7,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_8,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_9,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_10,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
```

```
plot(Data_like_rho_B$rho_like_11,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_12,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_13,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_14,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_15,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_16,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_17,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_18,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_19,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_20,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_21,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_22,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_23,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_24,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_25,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
```

```
plot(Data_like_rho_B$rho_like_26,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_27,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_28,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_29,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_30,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_31,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_32,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_33,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_34,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_35,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_36,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_37,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_38,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_39,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_40,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
```

```
plot(Data_like_rho_B$rho_like_41,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_42,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_43,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_44,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_45,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_46,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_47,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_48,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_49,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_50,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_51,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_52,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_53,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_54,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_55,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
```

```
plot(Data_like_rho_B$rho_like_56,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_57,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_58,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_59,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_60,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_61,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_62,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_63,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_64,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_65,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_66,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_67,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_68,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_69,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_70,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
```

```
plot(Data_like_rho_B$rho_like_71,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_72,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_73,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_74,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_75,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_76,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_77,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_78,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_79,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_80,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_81,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_82,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_83,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_84,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_85,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
```

```
plot(Data_like_rho_B$rho_like_86,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_87,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_88,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_89,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_90,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_91,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_92,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_93,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_94,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_95,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_96,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_97,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_98,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_99,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(Data_like_rho_B$rho_like_100,Depth,ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
```

```
plot(rho,Depth,xlab=expression(paste(rho (g/
 ↪cm^3))),ylab="Depth",ylim=rev(range(Depth)),type = "l",lwd=1,col=("black"))
legend(x = "topleft", legend = c(expression(paste(rho)),expression(paste(rho,␣
 ↪"sim"))), fill = c("black", "green3"))
grid()

plot(Data_like_rho_B$rho_like_96,Depth,xlab=expression(paste(rho (g/
 ↪cm^3))),ylim=rev(range(Depth)),type =␣
 ↪"l",lwd=1,col=("green3"),xaxt='n',yaxt='n',ann=FALSE)
par(new=TRUE)
plot(rho,Depth,xlab=expression(paste(rho (g/
 ↪cm^3))),ylab="Depth",ylim=rev(range(Depth)),type = "l",lwd=1,col=("black"))
grid()

plot((rho-Data_like_rho_B$rho_like_89),Depth,xlab="Difference",ylab="Depth",ylim=rev(range(Dept
 ↪= "l",lwd=1,col=("red"))
grid()
```

[30]:
```
save.image() #important line, this is for save the R workspace, this include␣
↪variables and results
```

As we can see, the spatial distributions obtained using the Bernstein copula are even closer to the original distribution. However, the computational cost is high, as the simulation of 10,000 samples can take hours.

9 Conclusions.

In the comparison between parametric and non-parametric approaches of the spatial stochastic simulation method based on copulas, it turned out that the parametric approach has a very low computational cost, since its execution lasts only a few seconds. While the non-parametric approach can take up to hours. This is because the first approach involves analytic functions in

the calculation, while the second one requires the calculation of the joint probability distribution function through the numerical approximation of the copula and its marginals with Bernstein polynomials.

When comparing the numerical results, it can be verified that the parametric approach does not reproduce data statistics such as minimum, maximum, skewness and kurtosis with the same precision as the non-parametric approach. The parametric approach presents a greater dispersion expressed by its variance, which means more uncertainty. This fact is due to it is practically impossible to be able to represent the underlying dependence complexity between data with a parametric model, since the data is forced to fitting an analytical model depending of a single parameter, where as the objective is to represent the real dependence complexity existing in the data, the latter only could be done by the non-parametric approach.

This indicates that it is advisable to use the parametric approach in cases with large data samples and limited computing capabilities. This problem would be expected to occur when the method is applied in three spatial dimensions and when more than two variables are included in the multivariate joint estimation as well. However, the non-parametric approach could be applied if the appropriate computational capabilities are available and predictions with less uncertainty are required.

10 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.

11 Appendix

11.1 Appendix A: Copula definition and basic properties

According to the definition of [Nelsen, 2006], a copula is *a function that joins or couples multivariate distribution functions to their one-dimensional marginal distribution functions* If we want to use copulas in the field of statistics, use Sklar's theorem. Sklar's theorem is defined as:

$*Let H be a joint distribution function with margins F and G. Then there exists a copula C such that for all (x,y) in* $\bar{R}$*

$$H(x, y) = C(F(x), G(y)) \tag{A1}$$

If $u = F(x)$ and $v = F(y)$ are marginal distribution functions and $C(u, v) = C(F(x), F(y))$ is a distribution function bivariate valid, then for any copula $C(u, v)$ its partial derivatives are $\frac{\partial C}{\partial u}$ and $\frac{\partial C}{\partial v}$ exist for almost all (u, v) in $[0; 1]$. Then $\frac{\partial^2 C}{\partial u \partial v}$ and $\frac{\partial^2 C}{\partial v \partial u}$ exist and are continuous on I^2. If this is true, the density function of the copula is [Shemyakin and Kniazev, 2017].

$$c(u,v) = \frac{\partial^2 C}{\partial u \partial v} \tag{A2}$$

And the joint probability density function of x and y is:

$$f(x,y) = \frac{\partial^2 C}{\partial u \partial v} \cdot \frac{dF}{dx} \cdot \frac{dG}{dy} \tag{A3}$$

11.2 Appendix B: Bernstein copula simulation method

The Bernstein copula provides an estimate to the copula using the empirical copula and the Bernstein polynomials. If $F(x)$ and $G(y)$ are continuous, by elementary probability we know that $U = F(X)$ and $V = G(Y)$ are continuous Uniform (0,1) random variables, and the underlying copula C for the random vector (U,V) is the same copula corresponding to (X,Y), and by Sklar's Theorem we have that the joint probability distribution function for (U,V) is equal to $F_{UV}(u,v) = C(F(u),G(v)) = C(u,v)$. Therefore, in case $F(x)$ and $G(y)$ are known and F_{XY} is unknown, if $\{(x_1,y_1),\ldots,(x_n,y_n)\}$ is an observed random sample from (X,Y), the set $\{(u_k,v_k) = (F_X(x_k),F_Y(y_k))k = 1,\ldots,n\}$ would be an observed random sample from (U,V) with the same underlying copula C as (X,Y), and since $C = F_{UV}$ we may use the (u_k,v_k) values (called copula observations) to estimate C as a joint empirical distribution:

$$\hat{C}(u,v) = \frac{1}{n}\sum_{k=1}^{n} 1_{\{u_k \le u, v_k \le v\}} \tag{B1}$$

Strictly speaking, the estimation $\hat{C}$ is not a copula since it is discontinuous and copulas are always continuous. If F_X, F_Y, and F_{XY} are all unknown (the usual case) we estimate F_X and F_Y by univariate empirical distribution functions:

$$\widehat{F_X}(x) = \frac{1}{n}\sum_{k=1}^{n} 1_{\{x_k \le x\}} \qquad \widehat{F_Y}(y) = \frac{1}{n}\sum_{k=1}^{n} 1_{\{y_k \le y\}} \tag{B2}$$

Now the set of pairs $\left\{(u_k, v_k) = \left(\widehat{F_X}(x_k), \widehat{F_Y}(y_k)\right) \quad : \quad k = 1,\ldots,n\right\}$ is referred to as copula pseudo-observations. It is straightforward to verify that $\widehat{F_X}(x_k) = \frac{1}{n}rank(x_k)$ and $\widehat{F_Y}(y_k) = \frac{1}{n}rank(y_k)$. In this case the concept of empirical copula, see (Nelsen, 2006), is defined as the following function $C_n : I_n^2$ß$[0,1]$, where $I_n = \left\{\frac{i}{n} i = 0,\ldots,n\right\}$, given by:

$$C_n\left(\frac{i}{n},\frac{j}{n}\right) = \frac{1}{n}\sum_{k=1}^{n} 1_{\{rank(x_k) \le i, rank(y_k) \le j\}} \tag{B3}$$

Again, C_n is not a copula but it is an estimation of the underlying copula C on the grid I_n^2 that may be extended to a copula on $[0,1]^2$ by means of, for example, Bernstein polynomials, as proposed and studied in Sancetta & Satchell (2004), which leads to what is known as a Bernstein copula non-parametric estimation $\tilde{C} : [0,1]^2$ß$[0,1]$ given by:

$$\tilde{C}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{n} C_n\left(\frac{i}{n},\frac{j}{n}\right)\binom{n}{i} u^i(1-u)^{n-i}\binom{n}{j} v^j(1-v)^{n-j} \tag{B5}$$

As summarized in Erdely & Diaz-Viera (2010) in order to simulate replications from the random vector (X,Y) with the dependence structure inferred from the observed data $\{(x_1,y_1),\ldots,(x_n,y_n)\}$ we have the following:

Algorithm 1 * Generate two independent and continuous Uniform $(0,1)$ random variates u and t. * Set $v = c_u^{-1}(t)$ where $c_u(v) = \frac{\partial \tilde{C}(u,v)}{\partial(u)}$. * The desired pair is $(x,y) = \left(\tilde{Q}_n(u), \tilde{R}_n(v)\right)$ where $\tilde{Q}_n$ and $\tilde{R}_n$ are empirical quantile functions for X and Y, respectively.

For a value x in the range of the random variable X and a given $0 < \alpha < 1$ let $y = \varphi_\alpha(x)$ denote the solution to the equation $P(Y \leq yX = x) = \alpha$. Then the graph of $y = \varphi_\alpha(x)$ is the α-quantile regression curve of Y conditional on $X = x$. In Nelsen (2006) is proven that:

$$P(Y \leq yX = x) = c_u(v)|_{(u=F_X(x),v=F_Y(y))} \tag{B4}$$

This result leads to the following algorithm to obtain the α-quantile regression curve of Y conditional on $X = x$:

Algorithm 2 * Set $c_u(v) = \alpha$. * Solve for v the regression curve, say $v = g_\alpha(u)$. * Replace u by $\widetilde{Q_n^{-1}}(x)$ and v by $\widetilde{R_n^{-1}}(y)$. * Solve for y the regression curve, say $y = \varphi_\alpha(x)$.

References

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

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

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

[Bosch et al., 2007] Bosch, M., Cara, L., Rodrigues, J., Navarro, A., and Díaz, M. (2007). A monte carlo approach to the joint estimation of reservoir and elastic parameters from seismic amplitudes. *Geophysics*, 72.

[Caers et al., 2000] Caers, J., Srinivasan, S., and Journel, A. G. (2000). Geostatistical quantification of geological information for a fluvial-type north sea reservoir. *SPE Reservoir Evaluation and Engineering - SPE RESERV EVAL ENG*, 3.

[Chiles and Delfiner, 1999] Chiles, J.-P. and Delfiner, P. (1999). Geostatistics: Modeling spatial uncertainty. *Wiley series in probability and statistics. Applied probability and statistics section.*, pages xi, 695.

[Cooray, 2018] Cooray, K. (2018). Strictly archimedean copulas with complete association for multivariate dependence based on the clayton family. *Dependence Modeling*, 6:1–18.

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

[Deutsch and Journel, 1998] Deutsch, C. V. and Journel, A. G. (1998). *GSLIB: Geostatistical software library and user's guide*, volume 369.

[Diaz-Viera et al., 2006] Diaz-Viera, M., Anguiano-Rojas, P., Mousatov, A., Kazatchenko, E., and Markov, M. (2006). Stochastic modeling of permeability in double porosity carbonates applying a monte-carlo simulation method with t-copulas;.

[Díaz-Víera and Casar-González, 2005] Díaz-Víera, M. and Casar-González, R. (2005). Stochastic simulation of complex dependency pattern of petrophysical properties using t-copulas. page 7.

[Diaz-Viera et al., 2017] 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*.

[Díaz-Viera et al., 2018] Díaz-Viera, M., Le, V., and Vázquez-Ramírez, D. (2018). A prediction of the spatial distribution of petrophysical properties with bernstein copula using seismic attributes as secondary variables. InterPore.

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

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

[Dubrule, 2003] Dubrule, O. (2003). *Geostatistics for Seismic Data Integration in Earth Models*. Society of Exploration Geophysicists and European Association of Geoscientists and Engineers.

[Erdely et al., 2012] Erdely, A., Diaz-Viera, M., and Hernandez-Maldonado, V. (2012). *Trivariate nonparametric dependence modeling of petrophysical properties*.

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

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

[Iturraran-Viveros and Parra, 2014] Iturraran-Viveros, U. and Parra, J. (2014). Artificial neural networks applied to estimate permeability, porosity and intrinsic attenuation using seismic attributes and well-log data. *Journal of Applied Geophysics*, pages 45–54.

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

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

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

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

[Nelsen, 2006] Nelsen, R. B. (2006). *An Introduction to Copulas*. Springer New York.

[Oh and Kwon, 2001] Oh, S. and Kwon, B.-D. (2001). Geostatistical approach to bayesian inversion of geophysical data: Markov chain monte carlo method. *Earth Planets and Space*, 53:777–791.

[Parra et al., 2014] Parra, J., Iturraran-Viveros, U., Parra, J., and Xu, P.-C. (2014). Neural network and rock physics for predicting and modeling attenuation logs.

[Pyrcz and Deutsch, 2014] Pyrcz, M. J. and Deutsch, C. V. (2014). *Geostatistical Reservoir Modeling*. OUP USA.

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

[Trivedi and Zimmer, 2007] Trivedi, P. and Zimmer, D. (2007). Copula modeling: An introduction for practitioners. *Foundations and Trends(R) in Econometrics*, 1:1–111.

[Vázquez, 2018] Vázquez, D. (2018). Simulación estocástica conjunta de propiedades petrofísicas con cópulas de bernstein usando atributos sísmicos como variables secundarias a escala de registros de pozo.