Quebras de Barragens
Estimação Não-Paramétrica Completa

Rafael Rodrigues de Moraes
Verónica Andréa González-López

16 de Janeiro de 2021

Conteúdo

No artigo Larrauri e Lall (2018) os autores abordam metodologias de previsão da extensão dos resíduos a ser atingida em caso de quebras de barragens. Esse tema é de preocupação contemporânea por causa de derramamentos recentes em território brasileiro e todos os problemas ambientais, sociais e políticos decorrentes de tais tragédias.

1 Bibliotecas, Importação e Seleção de variáveis

1.1 Bibliotecas no GNU R necessárias para a análise

```
library(copula)      # diversas funções relacionadas à estimação de cópulas
library(kdecopula)   # estimação não paramétrica completa de cópulas
library(ggplot2)     # gráficos com qualidade de publicação
library(ggrepel)     # anotação nos gráficos
library(dplyr)       # %>%
library(snpar)       # kde
library(tidyr)       # pivot_longer
library(xtable)      # xtable
casas.decimais <- 3
```

1.2 Importação dos dados

1.2.1 Script de importação e tratamento dos dados no R

```
## Tabela de nomes das minas
names <- dt[ , c("No", "Mine") ]

## data.frame base para análises (ainda contém NAs !)
dt    <- dt[ , c("No", "Year", "H", "VT", "Dmax", "VF", "Mine") ]

## Novas variáveis
dt$R <- dt$H * dt$VF
dt$Rf <- round( dt$H * ( dt$VF/dt$VT) * dt$VF, 2)

## data.frame base para análises (ainda contém NAs !)
dt    <- dt[ , c("No", "Year", "H", "VT", "Dmax", "VF", "R", "Rf", "Mine") ]
```

1.3 Seleção de variáveis para modelagem externa ao artigo

```
## salva data.frame em '0_Dissertação/Aplicacoes/Barragens/artigo/full_nonpar/dados/barragens_dados.dat'
write.table(
    data.frame(
        dt
    )
   ,file='./dados/barragens_dados.dat'
   ,sep='|'
   ,row.names=FALSE
   ,quote = FALSE
)
```

2 Descrição dos dados

Os dados presentes no artigo em questão não necessitam de tratamento, exceto a exclusão de informações faltantes (*missing values*).

Abaixo são listadas 11 observações do conjunto de dados utilizado nas análises. Ao todo estão disponíveis dados para 35 quebras de barragens, dos quais 30 são registros completos, ou seja, não há informação faltante em qualquer das colunas da tabela.

```
set.seed(5)
amostra <- sample( 1:nrow(dt), size=11)
dt[sort(amostra),]
```

Tabela 1: Trecho da tabela de dados usada na análise.

No	Year	H	VT	Dmax	VF	R	Rf	Mine
2	2000	15	15	5.2	1.8	27	3.24	
3	1978	25	1.7	0.3	0.0211	0.5275	0.01	
6	1965	20	0.45	0.8	0.07	1.4	0.22	
7	1985	6	0.038	0.8	0.011	0.066	0.02	
9	1985	40	2	8	0.5	20	5	
11	1971	15	12.34	120	9	135	98.46	Cities Service, FL, USA
15	2015	90	55	637	32	2880	1675.64	Samarco Fundão-Santarém, MG, BRA
19	2014	40	74	7	23.6	944	301.06	Polley, BC, CAN
21	1965	15	nil	5	0.035	0.525	nil	
23	1965	15	0.043	5	0.021	0.315	0.15	
25	1978	28	0.48	8	0.08	2.24	0.37	

3 V_F em função de V_T

```
dt.sub <- na.omit(
  dt[, c("VF","VT","No") ]
)
dt.sub <- merge(
    dt.sub
   ,names
   ,by="No"
)
label      <- dt.sub$Mine

varX       <- dt.sub$VT
varX.pobs  <- rank( varX ) / ( length( varX ) + 1 )
varX.label <- "VT"

varY       <- dt.sub$VF
varY.pobs  <- rank( varY ) / ( length( varY ) + 1 )
varY.label <- "VF"
```

3.1 Estimação das marginais

A estimação das marginais VF e VT se dará não-parametricamente de duas formas: (1) via pseudo-observações e (2) via densidades de kernel.

Com base no pacote `snpar` do software `GNU R` estimou-se a função densidade de probabilidade com base em diferentes funções de kernel e, finalmente, obteve-se uma estimativa para a função distribuição acumulada, calculada a partir da função densidade de kernel.

3.1.1 VF

Na tabela abaixo é apresentada uma comparação entre as pseudo-observações e as funções de distribuição acumulada para os diferentes kernels.

```
kdeVarY <- data.frame(
  varY
 ,pobs = rank( varY ) / ( length( varY ) + 1 )
 ,epan = snpar::kde( x = varY, kernel='epan', plot=FALSE)$Fhat
 ,unif = snpar::kde( x = varY, kernel='unif', plot=FALSE)$Fhat
 ,tria = snpar::kde( x = varY, kernel='tria', plot=FALSE)$Fhat
 ,quar = snpar::kde( x = varY, kernel='quar', plot=FALSE)$Fhat
 ,triw = snpar::kde( x = varY, kernel='triw', plot=FALSE)$Fhat
 ,tric = snpar::kde( x = varY, kernel='tric', plot=FALSE)$Fhat
 ,gaus = snpar::kde( x = varY, kernel='gaus', plot=FALSE)$Fhat
 ,cos  = snpar::kde( x = varY, kernel='cos' , plot=FALSE)$Fhat
)
colnames(kdeVarY)[1] <- varY.label

kdeVarY[ order(kdeVarY[,1]) , ] %>% round(., casas.decimais)
```

VF	pobs	epan	unif	tria	quar	triw	tric	gaus	cos
0.011	0.032	0.293	0.309	0.284	0.284	0.276	0.287	0.333	0.292
0.021	0.065	0.296	0.31	0.287	0.287	0.28	0.29	0.334	0.294
0.021	0.097	0.296	0.31	0.287	0.287	0.28	0.29	0.334	0.294
0.025	0.129	0.297	0.311	0.288	0.288	0.281	0.291	0.335	0.295
0.038	0.177	0.3	0.314	0.292	0.292	0.286	0.295	0.337	0.299
0.038	0.177	0.3	0.314	0.292	0.292	0.286	0.295	0.337	0.299
0.07	0.226	0.308	0.32	0.302	0.302	0.297	0.304	0.342	0.307
0.08	0.258	0.311	0.322	0.306	0.305	0.301	0.307	0.343	0.31
0.085	0.306	0.312	0.323	0.307	0.307	0.302	0.308	0.344	0.311
0.085	0.306	0.312	0.323	0.307	0.307	0.302	0.308	0.344	0.311
0.09	0.355	0.313	0.324	0.309	0.308	0.304	0.31	0.345	0.312
0.17	0.387	0.334	0.339	0.334	0.333	0.333	0.333	0.357	0.334
0.19	0.419	0.339	0.343	0.341	0.34	0.34	0.339	0.36	0.339
0.2	0.452	0.342	0.345	0.344	0.343	0.344	0.342	0.362	0.342
0.22	0.484	0.347	0.349	0.35	0.349	0.351	0.348	0.365	0.348
0.28	0.516	0.363	0.361	0.369	0.368	0.373	0.365	0.374	0.364
0.35	0.548	0.381	0.374	0.391	0.39	0.398	0.386	0.385	0.383
0.37	0.581	0.387	0.378	0.398	0.396	0.405	0.392	0.388	0.388
0.5	0.613	0.421	0.403	0.437	0.436	0.45	0.43	0.408	0.424
0.6	0.645	0.447	0.423	0.466	0.467	0.483	0.46	0.423	0.451
1.8	0.677	0.699	0.664	0.702	0.707	0.709	0.709	0.597	0.701
1.9	0.71	0.711	0.685	0.713	0.716	0.717	0.717	0.61	0.712
2	0.742	0.721	0.706	0.722	0.723	0.724	0.723	0.622	0.721
3	0.774	0.785	0.78	0.785	0.787	0.787	0.787	0.726	0.785
3.5	0.806	0.812	0.805	0.812	0.813	0.814	0.813	0.765	0.812
4.2	0.839	0.838	0.833	0.84	0.842	0.844	0.841	0.806	0.839
6.8	0.871	0.883	0.883	0.883	0.883	0.883	0.883	0.881	0.883
9	0.903	0.917	0.917	0.917	0.917	0.917	0.917	0.912	0.917
23.6	0.935	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
32	0.968	0.983	0.983	0.983	0.983	0.983	0.983	0.983	0.983

Abaixo apresenta-se a distância de Kolmogorov-Smirnoff entre a FDA empírica e a FDA calculada com base nas diferentes funções de kernel, ou seja, $\max |\hat{F}(x) - F_k^*(x)|$, sendo $\hat{F}(x)$ a função distribuição acumulada empírica e $F_k^*(x)$ a função distribuição acumulada obtida a partir da função densidade do kernel k, com $k =$ epanechnikov, uniform, quartic, triagular, triweight, gaussian, cosine e tricube.

kernel_nome	kernel	max_distance_KS
Triweight	triw	0.244
Quartic	quar	0.251
Triangular	tria	0.251
Tricube	tric	0.254
Cosine	cos	0.259
Epanechnikov	epan	0.261
Uniform	unif	0.276
Gaussian	gaus	0.301

Como se pode notar, a função de kernel Triweight responde pela menor distância em relação à função distribuição acumulada empírica.

3.1.2 VT

Na tabela abaixo é apresentada uma comparação entre as pseudo-observações e as funções de distribuição acumulada para os diferentes kernels.

```
kdeVarX <- data.frame(
  varX
 ,pobs = rank( varX ) / ( length( varX ) + 1 )
 ,epan = snpar::kde( x = varX, kernel='epan', plot=FALSE)$Fhat
 ,unif = snpar::kde( x = varX, kernel='unif', plot=FALSE)$Fhat
 ,tria = snpar::kde( x = varX, kernel='tria', plot=FALSE)$Fhat
 ,quar = snpar::kde( x = varX, kernel='quar', plot=FALSE)$Fhat
 ,triw = snpar::kde( x = varX, kernel='triw', plot=FALSE)$Fhat
 ,tric = snpar::kde( x = varX, kernel='tric', plot=FALSE)$Fhat
 ,gaus = snpar::kde( x = varX, kernel='gaus', plot=FALSE)$Fhat
 ,cos  = snpar::kde( x = varX, kernel='cos' , plot=FALSE)$Fhat
)
colnames(kdeVarX)[1] <- varX.label

kdeVarX[ order(kdeVarX[,1]) , ] %>% round(., casas.decimais)
```

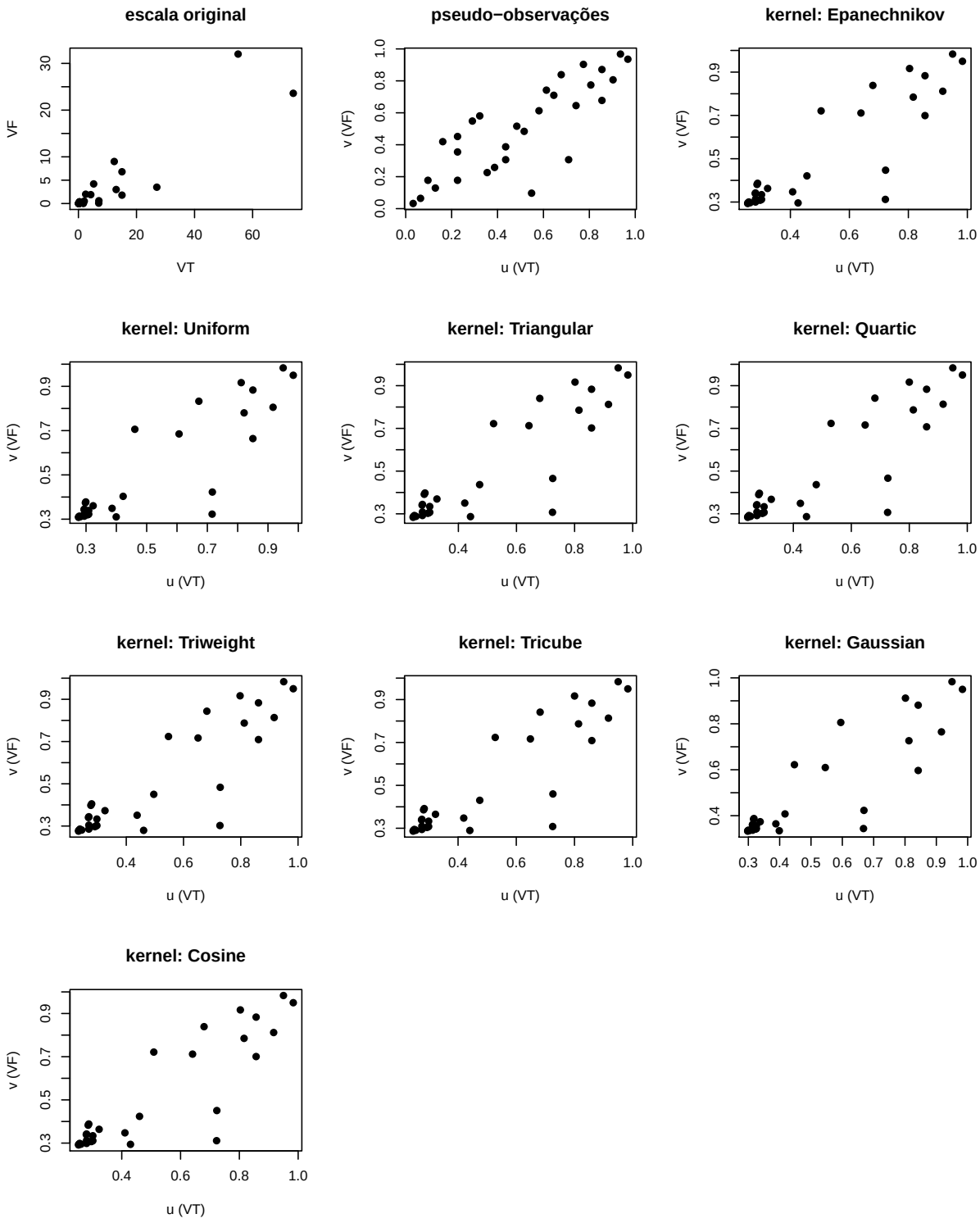
VT	pobs	epan	unif	tria	quar	triw	tric	gaus	cos
0.038	0.032	0.255	0.275	0.244	0.243	0.233	0.246	0.298	0.253
0.043	0.065	0.256	0.276	0.245	0.243	0.234	0.246	0.298	0.253
0.074	0.097	0.259	0.278	0.249	0.247	0.238	0.25	0.3	0.257
0.12	0.129	0.264	0.281	0.254	0.253	0.244	0.255	0.303	0.261
0.29	0.161	0.281	0.294	0.275	0.273	0.268	0.275	0.313	0.279
0.3	0.226	0.282	0.294	0.276	0.274	0.269	0.276	0.314	0.28
0.3	0.226	0.282	0.294	0.276	0.274	0.269	0.276	0.314	0.28
0.3	0.226	0.282	0.294	0.276	0.274	0.269	0.276	0.314	0.28
0.35	0.29	0.287	0.298	0.283	0.281	0.276	0.281	0.317	0.286
0.37	0.323	0.289	0.299	0.285	0.283	0.279	0.284	0.318	0.288
0.45	0.355	0.297	0.305	0.295	0.293	0.29	0.293	0.323	0.296
0.48	0.387	0.3	0.307	0.299	0.297	0.295	0.297	0.324	0.299
0.5	0.435	0.302	0.309	0.301	0.299	0.297	0.299	0.326	0.302
0.5	0.435	0.302	0.309	0.301	0.299	0.297	0.299	0.326	0.302
0.7	0.484	0.323	0.323	0.326	0.324	0.326	0.322	0.338	0.323
1.52	0.516	0.408	0.386	0.422	0.424	0.438	0.419	0.388	0.411
1.7	0.548	0.426	0.4	0.442	0.445	0.461	0.44	0.399	0.43
2	0.581	0.456	0.422	0.473	0.479	0.496	0.475	0.417	0.46
2.5	0.613	0.504	0.461	0.521	0.53	0.547	0.528	0.447	0.509
4.25	0.645	0.639	0.607	0.643	0.648	0.651	0.648	0.546	0.641
5.25	0.677	0.68	0.672	0.68	0.681	0.682	0.682	0.595	0.68
7	0.71	0.722	0.716	0.724	0.725	0.727	0.725	0.667	0.723
7.04	0.742	0.723	0.717	0.725	0.726	0.728	0.726	0.669	0.723
12.34	0.774	0.804	0.812	0.801	0.8	0.798	0.8	0.801	0.803
13	0.806	0.817	0.821	0.815	0.814	0.812	0.814	0.813	0.816
15	0.855	0.856	0.85	0.858	0.86	0.862	0.86	0.842	0.857
15	0.855	0.856	0.85	0.858	0.86	0.862	0.86	0.842	0.857
27	0.903	0.917	0.917	0.917	0.917	0.917	0.917	0.916	0.917
55	0.935	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
74	0.968	0.983	0.983	0.983	0.983	0.983	0.983	0.983	0.983

Abaixo apresenta-se a distância de Kolmogorov-Smirnoff entre a FDA empírica e a FDA calculada com base nas diferentes funções de kernel, ou seja, $\max |\hat{F}(x) - F_k^*(x)|$, sendo $\hat{F}(x)$ a função distribuição acumulada empírica e $F_k^*(x)$ a função distribuição acumulada obtida a partir da função densidade do kernel k, com k = epanechnikov, uniform, quartic, triagular, triweight, gaussian, cosine e tricube.

kernel_nome	kernel	max_distance_KS
Triweight	triw	0.201
Quartic	quar	0.21
Triangular	tria	0.212
Tricube	tric	0.213
Cosine	cos	0.221
Epanechnikov	epan	0.223
Uniform	unif	0.243
Gaussian	gaus	0.266

Como se pode notar, a função de kernel triw responde pela menor distância em relação à função distribuição acumulada empírica.

3.1.3 Comparação gráfica das funções de kernel



3.2 Estimação da dependência

3.2.1 Marginais em pseudo-observações

1. Cópula paramétrica (MPLE, pacote 'copula')

```
joe     <- fitCopula(     joeCopula(dim=2), data= as.matrix(data.frame(varX.pobs, varY.pobs)))
gumbel  <- fitCopula(  gumbelCopula(dim=2), data= as.matrix(data.frame(varX.pobs, varY.pobs)))
frank   <- fitCopula(   frankCopula(dim=2), data= as.matrix(data.frame(varX.pobs, varY.pobs)))
normal  <- fitCopula(  normalCopula(dim=2), data= as.matrix(data.frame(varX.pobs, varY.pobs)))
t       <- fitCopula(       tCopula(dim=2), data= as.matrix(data.frame(varX.pobs, varY.pobs)))
clayton <- fitCopula( claytonCopula(dim=2), data= as.matrix(data.frame(varX.pobs, varY.pobs)))

##  @loglik   --> log-verossimilhança
##  @estimate --> theta_hat
```

Tabela 2: Bondade do ajuste (por ordem decrescente do BIC)

copula	theta_mple	log_pseudolik	BIC
normal	0.875	19.32	17.619
gumbel	2.95	19.185	17.484
t	0.868 (df=3.529)	20.022	16.621
frank	9.385	16.968	15.268
joe	3.641	16.826	15.125
clayton	2.929	16.515	14.814

2. Cópula Não-paramétrica (Kernel, pacote 'kdecopula')

```r
## marginais em pseudo-observações
uv        <- data.frame( varX.pobs, varY.pobs)
names(uv) <- c( varX.label, varY.label )

## ajuste não-paramétrico por 3 métodos: T, TLL1 e TLL2
bic.method.T    <- round(
  BIC( fit.kdecop.T    <- kdecopula::kdecop( uv, method = 'T'    ) ) / (-2)
 ,digits=casas.decimais
)
bic.method.TLL1 <- round(
  BIC( fit.kdecop.TLL1 <- kdecopula::kdecop( uv, method = 'TLL1' ) ) / (-2)
 ,digits=casas.decimais
)
bic.method.TLL2 <- round(
  BIC( fit.kdecop.TLL2 <- kdecopula::kdecop( uv, method = 'TLL2' ) ) / (-2)
 ,digits=casas.decimais
)

nonpar_pobs_bestfit <- fit.kdecop.T
summary(nonpar_pobs_bestfit)

## plot(uv, pch=19)
## summary( fit <- kdecopula::kdecop( uv, method = 'TLL2' ) )
## paste( 'BIC_equiv_copula =', round( BIC(fit)/(-2) ,2) )
## subset( ajustes, copula == 'normal')
## plot(fit)
```

```
Kernel copula density estimate (tau = 0.62)
-------------------------------
Variables:    VT -- VF
Observations: 30
Method:       Transformation estimator ('T')
Bandwidth:    matrix(c(0.52, 0.44, 0, 0.28), 2, 2)
---
logLik: 23.26    AIC: -44.12    cAIC: -43.93    BIC: -42.44
Effective number of parameters: 1.2
```

3.2.2 Marginais em funções de kernel

```
varX.kern <- kdeVarX[ , varX_KS[ 1, 'kernel' ] ]
varY.kern <- kdeVarY[ , varY_KS[ 1, 'kernel' ] ]
```

1. Cópula paramétrica (MPLE, pacote 'copula')

```
joe     <- fitCopula(     joeCopula(dim=2), data= as.matrix(data.frame(varX.kern, varY.kern)))
gumbel  <- fitCopula(  gumbelCopula(dim=2), data= as.matrix(data.frame(varX.kern, varY.kern)))
frank   <- fitCopula(   frankCopula(dim=2), data= as.matrix(data.frame(varX.kern, varY.kern)))
normal  <- fitCopula(  normalCopula(dim=2), data= as.matrix(data.frame(varX.kern, varY.kern)))
t       <- fitCopula(       tCopula(dim=2), data= as.matrix(data.frame(varX.kern, varY.kern)))
clayton <- fitCopula( claytonCopula(dim=2), data= as.matrix(data.frame(varX.kern, varY.kern)))

##  @loglik   --> log-verossimilhança
##  @estimate --> theta_hat
```

Tabela 3: Bondade do ajuste (por ordem decrescente do BIC)

copula	theta_mple	log_pseudolik	BIC
clayton	6.339	27.783	26.083
normal	0.928	24.475	22.774
frank	14.716	24.204	22.504
t	0.929 (df=13.533)	24.639	21.238
gumbel	3.339	20.235	18.534
joe	3.359	15.214	13.514

2. Cópula Não-paramétrica (Kernel, pacote 'kdecopula')

```r
## marginais em kernel
uv        <- data.frame( varX.kern, varY.kern )
names(uv) <- c( varX.label, varY.label )

## ajuste não-paramétrico por 3 métodos: T, TLL1 e TLL2
bic.method.T    <- round(
  BIC( fit.kdecop.T    <- kdecopula::kdecop( uv, method = 'T'    ) ) / (-2)
 ,digits=casas.decimais
)
bic.method.TLL1 <- round(
  BIC( fit.kdecop.TLL1 <- kdecopula::kdecop( uv, method = 'TLL1' ) ) / (-2)
 ,digits=casas.decimais
)
bic.method.TLL2 <- round(
  BIC( fit.kdecop.TLL2 <- kdecopula::kdecop( uv, method = 'TLL2' ) ) / (-2)
 ,digits=casas.decimais
)

nonpar_kern_bestfit <- fit.kdecop.T
summary(nonpar_kern_bestfit)

## plot(uv, pch=19)
## summary( fit <- kdecopula::kdecop( uv, method = 'TLL2' ) )
## paste( 'BIC_equiv_copula =', round( BIC(fit)/(-2) ,2) )
## subset( ajustes, copula == 'normal')
## plot(fit)
```

```
Kernel copula density estimate (tau = 0.72)
-------------------------------
Variables:    VT -- VF
Observations: 30
Method:       Transformation estimator ('T')
Bandwidth:    matrix(c(0.47, 0.4, 0, 0.21), 2, 2)
---
logLik: 31.21    AIC: -60.62    cAIC: -60.5    BIC: -59.36
Effective number of parameters: 0.9
```

3.3 $E(V|U \in (a,b])$

$$E(V|U \in (a,b]) = 1 - \frac{1}{b-a}\left[\int_0^1 C(b,x)\,dx - \int_0^1 C(a,x)\,dx\right]$$

Esperança condicional $E(V|U \in (a,b])$, utilizando a cópula paramétrica (estimação do parâmetro de associação via máxima pseudo-verossimilhança) calculada com base nas marginais reescaladas pelas pseudo-observações (ref. aos postos das observações originais).

```
## Cálculo da Esperança condicional E( V | U \in (a,b] )
##
## Estimação semi-paramétrica
##   marginais corrigidas pelas pseudo-observações
##   e cópula paramétrica estimada via MPLE
##
expect_V_given_UinInterval_semipar <- function( a, b, theta, family ) {
    ## Expression for E( V | U \in (a,b], \theta )
    if ( a<0 || a>1 ) stop('a in (a,b] outside domain [0,1]. ')
    if ( b<0 || b>1 ) stop('b in (a,b] outside domain [0,1]. ')
    if ( a>b )        stop('a greater then b in (a,b]        ')
    ## no control for correctness on the copula association parameter...
    ## --
    ## cast arguments to numeric
    a     <- as.numeric(a)
    b     <- as.numeric(b)
    theta <- as.numeric(theta)
    ## first integral: \int_0^1 C(b,x) dx
    first_integral <- integrate(
        Vectorize(
            function(x) { pCopula( c(b,x), family( param=theta, dim=2) ) }
        )
       ,lower=0
       ,upper=1
    )
    ## second integral: \int_0^1 C(a,x) dx
    second_integral <- integrate(
        Vectorize(
            function(x) { pCopula( c(a,x), family( param=theta, dim=2) ) }
        )
       ,lower=0
       ,upper=1
    )
    result <- 1 - ( first_integral$value - second_integral$value ) / ( b - a )
    return( round( result, digits = casas.decimais ) )
}
## check: expect_V_given_UinInterval_semipar( a=0.0, b=0.25, theta=3, family = gumbelCopula )
```

a	b	VT_a	VT_b	estimation	expect_V_given_U	expect_VF_given_VT
0	0.25	0.038	0.3	semipar_pobs	0.175	0.021
0.25	0.5	0.3	1.52	semipar_pobs	0.4	0.085
0.5	0.75	1.52	7.04	semipar_pobs	0.6	0.6
0.75	1	7.04	74	semipar_pobs	0.825	32

Esperança condicional $E(V|U \in (a,b])$, utilizando a cópula não-paramétrica (estimação do parâmetro de associação via funções de kernel) calculada com base nas marginais reescaladas pelas

funções de distribuição acumuladas provenientes da estimação das funções densidade de probabilidade obtidas via funções de kernel.

```
## Estimação não-paramétrica completa
##   marginais corrigidas pelas FDA baseadas na fdp de kernel
##   e cópula não-paramétrica estimada via pacote kdecopula
##
expect_V_given_UinInterval_nonpar <- function( a, b, nonpar_copula ) {
    ## Expression for E( V | U \in (a,b], \theta )
    if ( a<0 || a>1 ) stop('a in (a,b] outside domain [0,1]. ')
    if ( b<0 || b>1 ) stop('b in (a,b] outside domain [0,1]. ')
    if ( a>b )        stop('a greater then b in (a,b]        ')
    ## no control for correctness on the copula association parameter...
    ## --
    ## cast arguments to numeric
    a     <- as.numeric(a)
    b     <- as.numeric(b)
    ## first integral: \int_0^1 C(b,x) dx
    first_integral <- integrate(
        Vectorize(
            function(x) { pkdecop( c(b,x), nonpar_copula ) }
        )
       ,lower=0
       ,upper=1
    )
    ## second integral: \int_0^1 C(a,x) dx
    second_integral <- integrate(
        Vectorize(
            function(x) { pkdecop( c(a,x), nonpar_copula ) }
        )
       ,lower=0
       ,upper=1
    )
    result <- 1 - ( first_integral$value - second_integral$value ) / ( b - a )
    return( round( result, digits = casas.decimais ) )
}
## check: expect_V_given_UinInterval_nonpar(a= 0, b= 0.25, nonpar_copula = fit.kdecop.T)
```

a	b	VT_a	VT_b	estimation	expect_V_given_U	expect_VF_given_VT
0	0.25	0.038	0.12	nonpar_kern	0.122	0.011
0.25	0.5	0.12	2	nonpar_kern	0.442	0.17
0.5	0.75	2	7.04	nonpar_kern	0.626	2
0.75	1	7.04	74	nonpar_kern	0.818	23.6

```
## Compara expect_V_given_U (marginais padronizadas)
compara_cond_expct <- bind_rows(
  cond_expct_semipar %>%
  mutate(
    interval = paste0('(',a,';',b,']')
   ,cond_expct = 'E_1 (semipar_pobs)'
  ) %>%
  select( interval, cond_expct, expect_V_given_U)
 ,
  cond_expct_nonpar %>%
  mutate(
    interval = paste0('(',a,';',b,']')
   ,cond_expct = 'E_2 (nonpar_kernel)'
  ) %>%
```

```
  select( interval, cond_expct, expect_V_given_U)
)

## Opção 1 - formato largo
compara_cond_expct %>%
  pivot_wider(
    id_cols = cond_expct
    ,names_from = interval
   ,values_from = expect_V_given_U
  ) %>%
  xtable( digits= casas.decimais ) %>%
  print(
    only.contents = FALSE
   ,include.colnames = TRUE
   ,include.rownames = FALSE
   ,hline.after = NULL
   ,comment=FALSE
  ) %>%
  cat(
    file = './dados/cond_expect_compara_formato_largo.tex'
  )


## Opção 2 - formato longo
compara_cond_expct %>%
  pivot_wider(
    id_cols = interval
    ,names_from = cond_expct
   ,values_from = expect_V_given_U
  ) %>%
  xtable( digits= casas.decimais ) %>%
  print(
    only.contents = FALSE
   ,include.colnames = TRUE
   ,include.rownames = FALSE
   ,hline.after = NULL
   ,comment=FALSE
  ) %>%
  cat(
    file = './dados/cond_expect_compara_formato_longo.tex'
  )

## escala original, resultados da estimação não-paramétrica completa
##  formato longo
cond_expct_nonpar %>%
  mutate(
    interval = paste0('(',VT_a,';',VT_b,']')
  ) %>%
  select( interval, expect_VF_given_VT) %>%
  rename( expectVFgivenVT = expect_VF_given_VT ) %>%
  xtable( digits= casas.decimais ) %>%
  print(
    only.contents = FALSE
   ,include.colnames = TRUE
   ,include.rownames = FALSE
   ,hline.after = NULL
   ,comment=FALSE
  ) %>%
  cat(
    file = './dados/cond_expect_fullnonpar_escala_original_formato_longo.tex'
  )

```

```r
##  formato largo
cond_expct_nonpar %>%
  mutate(
    interval = paste0('(',VT_a,';',VT_b,']')
  ) %>%
  select( interval, expect_VF_given_VT) %>%
  rename( expectVFgivenVT = expect_VF_given_VT ) %>%
  t() %>%
  xtable( digits= casas.decimais ) %>%
  print(
    only.contents = FALSE
   ,include.colnames = TRUE
   ,include.rownames = TRUE
   ,hline.after = NULL
   ,comment=FALSE
  ) %>%
  cat(
    file = './dados/cond_expect_fullnonpar_escala_original_formato_largo.tex'
  )
```

4 $D_{\max}$ em função de R_f

```
dt.sub <- na.omit(
  dt[, c("Dmax","Rf","No") ]
)
dt.sub <- merge(
    dt.sub
   ,names
   ,by="No"
)
label      <- dt.sub$Mine

varX       <- dt.sub$Rf
varX.pobs  <- rank( varX ) / ( length( varX ) + 1 )
varX.label <- "Rf"

varY       <- dt.sub$Dmax
varY.pobs  <- rank( varY ) / ( length( varY ) + 1 )
varY.label <- "Dmax"
```

4.1 Estimação das marginais

A estimação das marginais Dmax e Rf se dará não-parametricamente de duas formas: (1) via pseudo-observações e (2) via densidades de kernel.

Com base no pacote `snpar` do software `GNU R` estimou-se a função densidade de probabilidade com base em diferentes funções de kernel e, finalmente, obteve-se uma estimativa para a função distribuição acumulada, calculada a partir da função densidade de kernel.

4.1.1 Dmax

Na tabela abaixo é apresentada uma comparação entre as pseudo-observações e as funções de distribuição acumulada para os diferentes kernels.

```
kdeVarY <- data.frame(
  varY
 ,pobs = rank( varY ) / ( length( varY ) + 1 )
 ,epan = snpar::kde( x = varY, kernel='epan', plot=FALSE)$Fhat
 ,unif = snpar::kde( x = varY, kernel='unif', plot=FALSE)$Fhat
 ,tria = snpar::kde( x = varY, kernel='tria', plot=FALSE)$Fhat
 ,quar = snpar::kde( x = varY, kernel='quar', plot=FALSE)$Fhat
 ,triw = snpar::kde( x = varY, kernel='triw', plot=FALSE)$Fhat
 ,tric = snpar::kde( x = varY, kernel='tric', plot=FALSE)$Fhat
 ,gaus = snpar::kde( x = varY, kernel='gaus', plot=FALSE)$Fhat
 ,cos  = snpar::kde( x = varY, kernel='cos' , plot=FALSE)$Fhat
)
colnames(kdeVarY)[1] <- varY.label

kdeVarY[ order(kdeVarY[,1]) , ] %>% round(., casas.decimais)
```

Dmax	pobs	epan	unif	tria	quar	triw	tric	gaus	cos
0.03	0.032	0.144	0.167	0.136	0.134	0.127	0.135	0.213	0.142
0.1	0.065	0.147	0.17	0.14	0.137	0.131	0.138	0.216	0.146
0.15	0.097	0.15	0.172	0.143	0.14	0.134	0.141	0.218	0.148
0.3	0.145	0.157	0.178	0.151	0.148	0.143	0.148	0.223	0.155
0.3	0.145	0.157	0.178	0.151	0.148	0.143	0.148	0.223	0.155
0.8	0.21	0.183	0.2	0.18	0.176	0.173	0.175	0.242	0.181
0.8	0.21	0.183	0.2	0.18	0.176	0.173	0.175	0.242	0.181
1.3	0.258	0.209	0.222	0.209	0.205	0.204	0.203	0.261	0.209
1.5	0.29	0.22	0.231	0.221	0.218	0.217	0.215	0.269	0.22
2.5	0.323	0.279	0.285	0.282	0.281	0.283	0.279	0.309	0.279
4	0.355	0.375	0.369	0.375	0.378	0.379	0.379	0.371	0.376
5	0.419	0.438	0.425	0.438	0.441	0.44	0.442	0.413	0.438
5	0.419	0.438	0.425	0.438	0.441	0.44	0.442	0.413	0.438
5	0.419	0.438	0.425	0.438	0.441	0.44	0.442	0.413	0.438
5.2	0.484	0.45	0.436	0.45	0.453	0.452	0.454	0.421	0.451
6	0.516	0.495	0.482	0.497	0.498	0.499	0.499	0.453	0.496
7	0.548	0.546	0.536	0.549	0.551	0.555	0.55	0.493	0.547
8	0.629	0.591	0.577	0.597	0.6	0.606	0.599	0.531	0.593
8	0.629	0.591	0.577	0.597	0.6	0.606	0.599	0.531	0.593
8	0.629	0.591	0.577	0.597	0.6	0.606	0.599	0.531	0.593
8	0.629	0.591	0.577	0.597	0.6	0.606	0.599	0.531	0.593
12	0.726	0.721	0.705	0.724	0.727	0.73	0.728	0.653	0.722
12	0.726	0.721	0.705	0.724	0.727	0.73	0.728	0.653	0.722
25	0.774	0.783	0.783	0.783	0.783	0.783	0.783	0.782	0.783
41	0.806	0.82	0.823	0.819	0.818	0.817	0.818	0.825	0.819
45	0.839	0.847	0.844	0.848	0.849	0.849	0.849	0.841	0.847
80	0.871	0.883	0.883	0.883	0.883	0.883	0.883	0.883	0.883
110	0.903	0.917	0.917	0.917	0.917	0.917	0.917	0.918	0.917
120	0.935	0.95	0.95	0.95	0.95	0.95	0.95	0.948	0.95
637	0.968	0.983	0.983	0.983	0.983	0.983	0.983	0.983	0.983

Abaixo apresenta-se a distância de Kolmogorov-Smirnoff entre a FDA empírica e a FDA calculada com base nas diferentes funções de kernel, ou seja, $\max |\hat{F}(x) - F_k^*(x)|$, sendo $\hat{F}(x)$ a função distribuição acumulada empírica e $F_k^*(x)$ a função distribuição acumulada obtida a partir da função densidade do kernel k, com $k =$ epanechnikov, uniform, quartic, triagular, triweight, gaussian, cosine e tricube.

kernel_nome	kernel	max_distance_KS
Triweight	triw	0.095
Quartic	quar	0.101
Tricube	tric	0.103
Triangular	tria	0.104
Cosine	cos	0.11
Epanechnikov	epan	0.112
Uniform	unif	0.134
Gaussian	gaus	0.181

Como se pode notar, a função de kernel Triweight responde pela menor distância em relação à função distribuição acumulada empírica.

4.1.2 Rf

Na tabela abaixo é apresentada uma comparação entre as pseudo-observações e as funções de distribuição acumulada para os diferentes kernels.

```
kdeVarX <- data.frame(
  varX
 ,pobs = rank( varX ) / ( length( varX ) + 1 )
 ,epan = snpar::kde( x = varX, kernel='epan', plot=FALSE)$Fhat
 ,unif = snpar::kde( x = varX, kernel='unif', plot=FALSE)$Fhat
 ,tria = snpar::kde( x = varX, kernel='tria', plot=FALSE)$Fhat
 ,quar = snpar::kde( x = varX, kernel='quar', plot=FALSE)$Fhat
 ,triw = snpar::kde( x = varX, kernel='triw', plot=FALSE)$Fhat
 ,tric = snpar::kde( x = varX, kernel='tric', plot=FALSE)$Fhat
 ,gaus = snpar::kde( x = varX, kernel='gaus', plot=FALSE)$Fhat
 ,cos  = snpar::kde( x = varX, kernel='cos' , plot=FALSE)$Fhat
)
colnames(kdeVarX)[1] <- varX.label

kdeVarX[ order(kdeVarX[,1]) , ] %>% round(., casas.decimais)
```

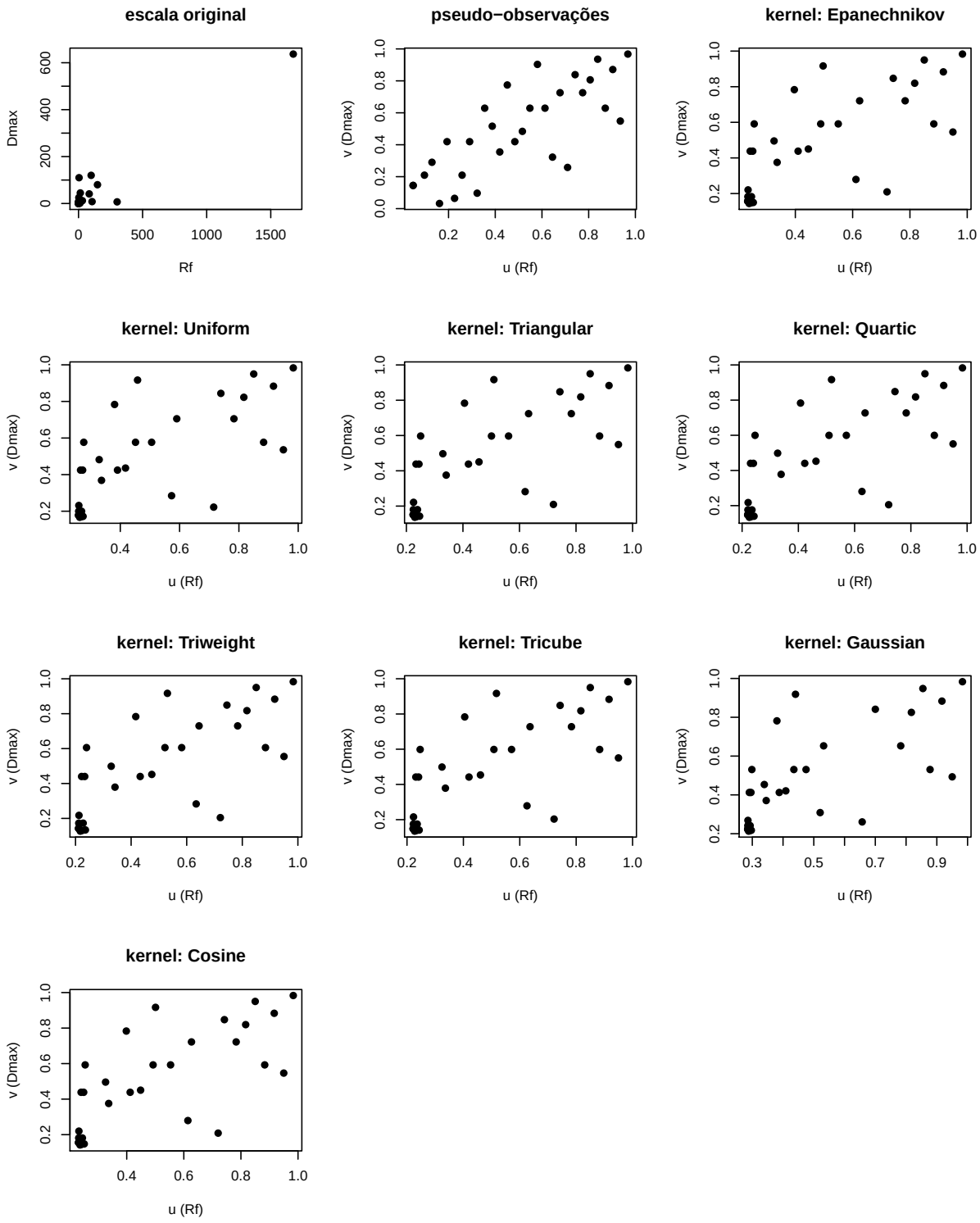
Rf	pobs	epan	unif	tria	quar	triw	tric	gaus	cos
0.01	0.048	0.234	0.259	0.224	0.22	0.211	0.222	0.285	0.231
0.01	0.048	0.234	0.259	0.224	0.22	0.211	0.222	0.285	0.231
0.02	0.097	0.234	0.26	0.225	0.221	0.212	0.222	0.286	0.232
0.03	0.129	0.235	0.26	0.225	0.221	0.212	0.223	0.286	0.232
0.09	0.161	0.239	0.263	0.23	0.226	0.217	0.227	0.288	0.236
0.15	0.194	0.242	0.266	0.234	0.23	0.222	0.231	0.29	0.24
0.18	0.226	0.244	0.268	0.236	0.232	0.224	0.233	0.292	0.242
0.22	0.258	0.247	0.27	0.239	0.235	0.227	0.236	0.293	0.245
0.29	0.29	0.251	0.273	0.244	0.24	0.233	0.241	0.296	0.249
0.32	0.323	0.253	0.274	0.247	0.242	0.236	0.243	0.297	0.251
0.37	0.355	0.256	0.277	0.25	0.246	0.24	0.247	0.299	0.254
1.43	0.387	0.326	0.329	0.329	0.326	0.328	0.324	0.339	0.326
1.59	0.419	0.336	0.337	0.341	0.339	0.342	0.336	0.345	0.337
2.49	0.452	0.396	0.381	0.405	0.408	0.417	0.405	0.38	0.399
2.69	0.484	0.41	0.391	0.419	0.423	0.433	0.42	0.388	0.412
3.24	0.516	0.445	0.418	0.457	0.463	0.474	0.461	0.409	0.449
3.93	0.548	0.489	0.451	0.501	0.509	0.522	0.508	0.435	0.492
4.07	0.581	0.497	0.458	0.509	0.518	0.531	0.517	0.44	0.501
5	0.613	0.55	0.506	0.561	0.571	0.582	0.571	0.475	0.554
6.31	0.645	0.612	0.573	0.62	0.626	0.634	0.626	0.521	0.614
6.65	0.677	0.624	0.59	0.632	0.637	0.644	0.636	0.532	0.627
11.34	0.71	0.72	0.715	0.72	0.721	0.721	0.722	0.657	0.72
13.85	0.742	0.742	0.739	0.743	0.743	0.745	0.743	0.7	0.742
29.73	0.774	0.783	0.783	0.783	0.783	0.783	0.783	0.783	0.783
83.23	0.806	0.817	0.817	0.817	0.817	0.817	0.817	0.817	0.817
98.46	0.839	0.85	0.85	0.85	0.85	0.85	0.85	0.855	0.85
105.6	0.871	0.883	0.883	0.883	0.883	0.883	0.883	0.878	0.883
147.84	0.903	0.917	0.917	0.917	0.917	0.917	0.917	0.917	0.917
301.06	0.935	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
1675.64	0.968	0.983	0.983	0.983	0.983	0.983	0.983	0.983	0.983

Abaixo apresenta-se a distância de Kolmogorov-Smirnoff entre a FDA empírica e a FDA calculada com base nas diferentes funções de kernel, ou seja, $\max |\hat{F}(x) - F_k^*(x)|$, sendo $\hat{F}(x)$ a função distribuição acumulada empírica e $F_k^*(x)$ a função distribuição acumulada obtida a partir da função densidade do kernel k, com k = epanechnikov, uniform, quartic, triagular, triweight, gaussian, cosine e tricube.

kernel_nome	kernel	max_distance_KS
Triweight	triw	0.162
Quartic	quar	0.172
Tricube	tric	0.173
Triangular	tria	0.175
Cosine	cos	0.183
Epanechnikov	epan	0.185
Uniform	unif	0.211
Gaussian	gaus	0.237

Como se pode notar, a função de kernel triw responde pela menor distância em relação à função distribuição acumulada empírica.

4.1.3 Comparação gráfica das funções de kernel



4.2 Estimação da dependência

4.2.1 Marginais em pseudo-observações

1. Cópula paramétrica (MPLE, pacote 'copula')

```
joe     <- fitCopula(     joeCopula(dim=2), data= as.matrix(data.frame(varX.pobs, varY.pobs)))
gumbel  <- fitCopula(  gumbelCopula(dim=2), data= as.matrix(data.frame(varX.pobs, varY.pobs)))
frank   <- fitCopula(   frankCopula(dim=2), data= as.matrix(data.frame(varX.pobs, varY.pobs)))
normal  <- fitCopula(  normalCopula(dim=2), data= as.matrix(data.frame(varX.pobs, varY.pobs)))
t       <- fitCopula(       tCopula(dim=2), data= as.matrix(data.frame(varX.pobs, varY.pobs)))
clayton <- fitCopula( claytonCopula(dim=2), data= as.matrix(data.frame(varX.pobs, varY.pobs)))

##  @loglik   --> log-verossimilhança
##  @estimate --> theta_hat
```

Tabela 4: Bondade do ajuste (por ordem decrescente do BIC)

copula	theta_mple	log_pseudolik	BIC
normal	0.799	13.007	11.306
gumbel	2.329	12.716	11.015
frank	7.237	12.409	10.709
t	0.799 (df=1479.849)	13.004	9.603
joe	2.888	11.136	9.435
clayton	1.558	9.088	7.388

2. Cópula Não-paramétrica (Kernel, pacote 'kdecopula')

```r
## marginais em pseudo-observações
uv        <- data.frame( varX.pobs, varY.pobs)
names(uv) <- c( varX.label, varY.label )

## ajuste não-paramétrico por 3 métodos: T, TLL1 e TLL2
bic.method.T    <- round(
  BIC( fit.kdecop.T    <- kdecopula::kdecop( uv, method = 'T'    ) ) / (-2)
 ,digits=casas.decimais
)
bic.method.TLL1 <- round(
  BIC( fit.kdecop.TLL1 <- kdecopula::kdecop( uv, method = 'TLL1' ) ) / (-2)
 ,digits=casas.decimais
)
bic.method.TLL2 <- round(
  BIC( fit.kdecop.TLL2 <- kdecopula::kdecop( uv, method = 'TLL2' ) ) / (-2)
 ,digits=casas.decimais
)

nonpar_pobs_bestfit <- fit.kdecop.T
summary(nonpar_pobs_bestfit)

## plot(uv, pch=19)
## summary( fit <- kdecopula::kdecop( uv, method = 'TLL2' ) )
## paste( 'BIC_equiv_copula =', round( BIC(fit)/(-2) ,2) )
## subset( ajustes, copula == 'normal')
## plot(fit)
```

```
Kernel copula density estimate (tau = 0.53)
-------------------------------
Variables:    Rf -- Dmax
Observations: 30
Method:       Transformation estimator ('T')
Bandwidth:    matrix(c(0.52, 0.39, 0, 0.34), 2, 2)
---
logLik: 16.9    AIC: -30.85    cAIC: -30.58    BIC: -28.78
Effective number of parameters: 1.47
```

4.2.2 Marginais em funções de kernel

```
varX.kern <- kdeVarX[ , varX_KS[ 1, 'kernel' ] ]
varY.kern <- kdeVarY[ , varY_KS[ 1, 'kernel' ] ]
```

1. Cópula paramétrica (MPLE, pacote 'copula')

```
joe     <- fitCopula(     joeCopula(dim=2), data= as.matrix(data.frame(varX.kern, varY.kern)))
gumbel  <- fitCopula(  gumbelCopula(dim=2), data= as.matrix(data.frame(varX.kern, varY.kern)))
frank   <- fitCopula(   frankCopula(dim=2), data= as.matrix(data.frame(varX.kern, varY.kern)))
normal  <- fitCopula(  normalCopula(dim=2), data= as.matrix(data.frame(varX.kern, varY.kern)))
t       <- fitCopula(       tCopula(dim=2), data= as.matrix(data.frame(varX.kern, varY.kern)))
clayton <- fitCopula( claytonCopula(dim=2), data= as.matrix(data.frame(varX.kern, varY.kern)))

##  @loglik   --> log-verossimilhança
##  @estimate --> theta_hat
```

Tabela 5: Bondade do ajuste (por ordem decrescente do BIC)

copula	theta_mple	log_pseudolik	BIC
clayton	2.694	13.065	11.365
normal	0.797	12.155	10.454
t	0.781 (df=1.876)	13.245	9.844
gumbel	2.242	11.434	9.734
frank	7.007	10.895	9.194
joe	2.475	9.452	7.752

2. Cópula Não-paramétrica (Kernel, pacote 'kdecopula')

```r
## marginais em kernel
uv        <- data.frame( varX.kern, varY.kern )
names(uv) <- c( varX.label, varY.label )

## ajuste não-paramétrico por 3 métodos: T, TLL1 e TLL2
bic.method.T    <- round(
  BIC( fit.kdecop.T    <- kdecopula::kdecop( uv, method = 'T'    ) ) / (-2)
 ,digits=casas.decimais
)
bic.method.TLL1 <- round(
  BIC( fit.kdecop.TLL1 <- kdecopula::kdecop( uv, method = 'TLL1' ) ) / (-2)
 ,digits=casas.decimais
)
bic.method.TLL2 <- round(
  BIC( fit.kdecop.TLL2 <- kdecopula::kdecop( uv, method = 'TLL2' ) ) / (-2)
 ,digits=casas.decimais
)

nonpar_kern_bestfit <- fit.kdecop.T
summary(nonpar_kern_bestfit)

## plot(uv, pch=19)
## summary( fit <- kdecopula::kdecop( uv, method = 'TLL2' ) )
## paste( 'BIC_equiv_copula =', round( BIC(fit)/(-2) ,2) )
## subset( ajustes, copula == 'normal')
## plot(fit)
```

```
Kernel copula density estimate (tau = 0.52)
-------------------------------
Variables:    Rf -- Dmax
Observations: 30
Method:       Transformation estimator ('T')
Bandwidth:    matrix(c(0.48, 0.38, 0, 0.35), 2, 2)
---
logLik: 16.27    AIC: -29.47    cAIC: -29.19    BIC: -27.33
Effective number of parameters: 1.53
```

4.3 $Prob(V > v|U \in (a,b])$

$$Prob(V > v|U \in (a,b]) = 1 - \frac{C(b,x)}{b-a} + \frac{C(a,x)}{b-a}$$

Probabilidade condicional $Prob(V > v|U \in (a,b])$, utilizando a cópula paramétrica (estimação do parâmetro de associação via máxima pseudo-verossimilhança) calculada com base nas marginais reescaladas pelas pseudo-observações (ref. aos postos das observações originais).

```
## Cálculo da Probabilidade Condicional Prob( V > v | U \in (a,b] )
##
## Estimação semi-paramétrica
##   marginais corrigidas pelas pseudo-observações
##   e cópula paramétrica estimada via MPLE
##
## Análise para (1) Dmax ~ R and (2) Dmax ~ Rf
prob_Vgtv_given_UinInterval_semipar <- function( v, a, b, theta, family ) {
    ## Expression for P( V>v | U \in (a,b], \theta )
    if ( v<0 || v>1 ) stop('v outside domain [0,1].          ')
    if ( a<0 || a>1 ) stop('a in (a,b] outside domain [0,1]. ')
    if ( b<0 || b>1 ) stop('b in (a,b] outside domain [0,1]. ')
    if ( a>b )        stop('a greater then b in (a,b]        ')
    aux_b    <- pCopula( c(b,v), family( param=theta, dim=2) ) / ( b - a )
    aux_a    <- pCopula( c(a,v), family( param=theta, dim=2) ) / ( b - a )
    result   <- 1 - ( (aux_b) - (aux_a) )
    return( round( result, digits = casas.decimais ) )
}
## check: prob_Vgtv_given_UinInterval_semipar( v=0.9, a=0.75, b=1, theta=3, family = gumbelCopula )
```

a	b	ranks	prob_Vgtv_given_U	estimation	Rf_a	Rf_b	Dmax
0	0.25	0.7	0.012	semipar_pobs	0.01	0.22	12
0	0.25	0.75	0.006	semipar_pobs	0.01	0.22	25
0	0.25	0.8	0.003	semipar_pobs	0.01	0.22	41
0	0.25	0.85	0.001	semipar_pobs	0.01	0.22	45
0	0.25	0.9	0	semipar_pobs	0.01	0.22	110
0	0.25	0.95	0	semipar_pobs	0.01	0.22	120
0.25	0.5	0.7	0.104	semipar_pobs	0.22	3.24	12
0.25	0.5	0.75	0.066	semipar_pobs	0.22	3.24	25
0.25	0.5	0.8	0.038	semipar_pobs	0.22	3.24	41
0.25	0.5	0.85	0.018	semipar_pobs	0.22	3.24	45
0.25	0.5	0.9	0.007	semipar_pobs	0.22	3.24	110
0.25	0.5	0.95	0.001	semipar_pobs	0.22	3.24	120
0.5	0.75	0.7	0.335	semipar_pobs	3.24	13.85	12
0.5	0.75	0.75	0.252	semipar_pobs	3.24	13.85	25
0.5	0.75	0.8	0.174	semipar_pobs	3.24	13.85	41
0.5	0.75	0.85	0.106	semipar_pobs	3.24	13.85	45
0.5	0.75	0.9	0.05	semipar_pobs	3.24	13.85	110
0.5	0.75	0.95	0.013	semipar_pobs	3.24	13.85	120
0.75	1	0.7	0.75	semipar_pobs	13.85	1675.64	12
0.75	1	0.75	0.675	semipar_pobs	13.85	1675.64	25
0.75	1	0.8	0.585	semipar_pobs	13.85	1675.64	41
0.75	1	0.85	0.475	semipar_pobs	13.85	1675.64	45
0.75	1	0.9	0.343	semipar_pobs	13.85	1675.64	110
0.75	1	0.95	0.186	semipar_pobs	13.85	1675.64	120

Probabilidade condicional $Prob(V > v|U \in (a, b])$, utilizando a cópula não-paramétrica (estimação do parâmetro de associação via funções de kernel) calculada com base nas marginais reescaladas pelas funções de distribuição acumuladas provenientes da estimação das funções densidade de probabilidade obtidas via funções de kernel.

```
## Estimação não-paramétrica completa
##   marginais corrigidas pelas FDA baseadas na fdp de kernel
##   e cópula não-paramétrica estimada via pacote kdecopula
##
prob_Vgtv_given_UinInterval_nonpar <- function( v, a, b, nonpar_copula ) {
    ## Expression for P( V>v | U \in (a,b], \theta )
    if ( v<0 || v>1 ) stop('v outside domain [0,1].          ')
    if ( a<0 || a>1 ) stop('a in (a,b] outside domain [0,1]. ')
    if ( b<0 || b>1 ) stop('b in (a,b] outside domain [0,1]. ')
    if ( a>b )        stop('a greater then b in (a,b]        ')
    aux_b    <- pkdecop( c(b,v), nonpar_copula ) / ( b - a )
    aux_a    <- pkdecop( c(a,v), nonpar_copula ) / ( b - a )
    result   <- 1 - ( (aux_b) - (aux_a) )
    return( round(result, digits = casas.decimais) )
}
## check: prob_Vgtv_given_UinInterval_nonpar( v=0.9, a=0.75, b=1, nonpar_copula = fit.kdecop.T )
```

a	b	ranks	prob_Vgtv_given_U	estimation	Rf_a	Rf_b	Dmax
0	0.25	0.7	0.034	nonpar_kern	0.01	0.37	12
0	0.25	0.75	0.027	nonpar_kern	0.01	0.37	12
0	0.25	0.8	0.023	nonpar_kern	0.01	0.37	25
0	0.25	0.85	0.02	nonpar_kern	0.01	0.37	45
0	0.25	0.9	0.019	nonpar_kern	0.01	0.37	80
0	0.25	0.95	0.018	nonpar_kern	0.01	0.37	120
0.25	0.5	0.7	0.144	nonpar_kern	0.37	3.93	12
0.25	0.5	0.75	0.102	nonpar_kern	0.37	3.93	12
0.25	0.5	0.8	0.067	nonpar_kern	0.37	3.93	25
0.25	0.5	0.85	0.038	nonpar_kern	0.37	3.93	45
0.25	0.5	0.9	0.015	nonpar_kern	0.37	3.93	80
0.25	0.5	0.95	0	nonpar_kern	0.37	3.93	120
0.5	0.75	0.7	0.383	nonpar_kern	3.93	13.85	12
0.5	0.75	0.75	0.297	nonpar_kern	3.93	13.85	12
0.5	0.75	0.8	0.215	nonpar_kern	3.93	13.85	25
0.5	0.75	0.85	0.141	nonpar_kern	3.93	13.85	45
0.5	0.75	0.9	0.078	nonpar_kern	3.93	13.85	80
0.5	0.75	0.95	0.024	nonpar_kern	3.93	13.85	120
0.75	1	0.7	0.645	nonpar_kern	13.85	1675.64	12
0.75	1	0.75	0.579	nonpar_kern	13.85	1675.64	12
0.75	1	0.8	0.501	nonpar_kern	13.85	1675.64	25
0.75	1	0.85	0.406	nonpar_kern	13.85	1675.64	45
0.75	1	0.9	0.294	nonpar_kern	13.85	1675.64	80
0.75	1	0.95	0.163	nonpar_kern	13.85	1675.64	120

```r
## Comparação da probabilidade condicional usando marginais
##   em escala [0,1]
##
##
cond_prob_compara <- bind_rows(
  ## estimação semiparamétrica usando pseudo-observações
  cond_prob_semipar %>%
  rename( limiar = ranks) %>%
  mutate(
    intervalo = paste0('(',a,', ',b,']')
   ,modelo = paste0('01_',varY.label,'_',varX.label)
   ,marginais = '01_pobs'
  ) %>%
  select( intervalo, limiar, modelo, marginais, prob_Vgtv_given_U)
 ,
  ## estimação não-parametrica usando marginais corrigidas pelo kernel
  cond_prob_nonpar %>%
  rename( limiar = ranks) %>%
  mutate(
    intervalo = paste0('(',a,', ',b,']')
   ,modelo = paste0('01_',varY.label,'_',varX.label)
   ,marginais = '02_kernel'
  ) %>%
  select( intervalo, limiar, modelo, marginais, prob_Vgtv_given_U)
)

## Probabilidade condicional usando marginais
##   em escala original e estimação não-paramétrica completa
##
##
cond_prob_escala_original_Dmax_Rf <- cond_prob_nonpar %>%
  rename(
    prob_cond = prob_Vgtv_given_U
    ,limiar = Dmax
  ) %>%
  mutate(
    intervalo = paste0('(', Rf_a,', ',Rf_b,']')
   ,intervalo = ifelse( ranks == 0.70, intervalo, '')
  ) %>%
  select( intervalo, limiar, prob_cond )

## exporta resultados da comparação entre semipar e nonpar
cond_prob_escala_original_Dmax_Rf %>%
  xtable( digits= c(0,0,1,casas.decimais) ) %>%
  print(
    only.contents = FALSE
   ,include.colnames = TRUE
   ,include.rownames = FALSE
   ,hline.after = NULL
   ,comment=FALSE
  ) %>%
  cat(
    file = './dados/cond_prob_escala_original_Dmax_Rf_formato_longo.tex'
  )
```

5 $D_{\max}$ em função de R

```
dt.sub <- na.omit(
  dt[, c("Dmax","R","No") ]
)
dt.sub <- merge(
    dt.sub
   ,names
   ,by="No"
)
label      <- dt.sub$Mine

varX       <- dt.sub$R
varX.pobs  <- rank( varX ) / ( length( varX ) + 1 )
varX.label <- "R"

varY       <- dt.sub$Dmax
varY.pobs  <- rank( varY ) / ( length( varY ) + 1 )
varY.label <- "Dmax"
```

5.1 Estimação das marginais

A estimação das marginais Dmax e R se dará não-parametricamente de duas formas: (1) via pseudo-observações e (2) via densidades de kernel.

Com base no pacote `snpar` do software `GNU R` estimou-se a função densidade de probabilidade com base em diferentes funções de kernel e, finalmente, obteve-se uma estimativa para a função distribuição acumulada, calculada a partir da função densidade de kernel.

5.1.1 Dmax

Na tabela abaixo é apresentada uma comparação entre as pseudo-observações e as funções de distribuição acumulada para os diferentes kernels.

```
kdeVarY <- data.frame(
  varY
 ,pobs = rank( varY ) / ( length( varY ) + 1 )
 ,epan = snpar::kde( x = varY, kernel='epan', plot=FALSE)$Fhat
 ,unif = snpar::kde( x = varY, kernel='unif', plot=FALSE)$Fhat
 ,tria = snpar::kde( x = varY, kernel='tria', plot=FALSE)$Fhat
 ,quar = snpar::kde( x = varY, kernel='quar', plot=FALSE)$Fhat
 ,triw = snpar::kde( x = varY, kernel='triw', plot=FALSE)$Fhat
 ,tric = snpar::kde( x = varY, kernel='tric', plot=FALSE)$Fhat
 ,gaus = snpar::kde( x = varY, kernel='gaus', plot=FALSE)$Fhat
 ,cos  = snpar::kde( x = varY, kernel='cos' , plot=FALSE)$Fhat
)
colnames(kdeVarY)[1] <- varY.label

kdeVarY[ order(kdeVarY[,1]) , ] %>% round(., casas.decimais)
```

Dmax	pobs	epan	unif	tria	quar	triw	tric	gaus	cos
0.03	0.029	0.155	0.178	0.147	0.145	0.138	0.146	0.224	0.153
0.1	0.057	0.159	0.181	0.151	0.148	0.142	0.15	0.226	0.157
0.15	0.1	0.161	0.183	0.154	0.151	0.145	0.152	0.228	0.159
0.15	0.1	0.161	0.183	0.154	0.151	0.145	0.152	0.228	0.159
0.3	0.157	0.169	0.19	0.163	0.16	0.155	0.16	0.234	0.167
0.3	0.157	0.169	0.19	0.163	0.16	0.155	0.16	0.234	0.167
0.61	0.2	0.185	0.204	0.182	0.178	0.174	0.177	0.245	0.184
0.8	0.243	0.196	0.213	0.194	0.189	0.187	0.188	0.253	0.194
0.8	0.243	0.196	0.213	0.194	0.189	0.187	0.188	0.253	0.194
1.3	0.286	0.224	0.236	0.225	0.22	0.22	0.218	0.273	0.223
1.5	0.314	0.235	0.245	0.237	0.233	0.234	0.231	0.281	0.235
2.5	0.343	0.296	0.3	0.3	0.299	0.302	0.297	0.322	0.297
4	0.371	0.394	0.384	0.395	0.399	0.401	0.4	0.385	0.395
5	0.443	0.457	0.44	0.458	0.461	0.461	0.463	0.428	0.458
5	0.443	0.457	0.44	0.458	0.461	0.461	0.463	0.428	0.458
5	0.443	0.457	0.44	0.458	0.461	0.461	0.463	0.428	0.458
5	0.443	0.457	0.44	0.458	0.461	0.461	0.463	0.428	0.458
5.2	0.514	0.469	0.452	0.47	0.473	0.473	0.474	0.436	0.47
6	0.543	0.514	0.499	0.516	0.517	0.519	0.518	0.47	0.514
7	0.571	0.563	0.554	0.566	0.568	0.572	0.567	0.51	0.564
8	0.643	0.607	0.595	0.613	0.615	0.621	0.614	0.548	0.609
8	0.643	0.607	0.595	0.613	0.615	0.621	0.614	0.548	0.609
8	0.643	0.607	0.595	0.613	0.615	0.621	0.614	0.548	0.609
8	0.643	0.607	0.595	0.613	0.615	0.621	0.614	0.548	0.609
12	0.743	0.739	0.725	0.742	0.745	0.747	0.745	0.673	0.74
12	0.743	0.739	0.725	0.742	0.745	0.747	0.745	0.673	0.74
12	0.743	0.739	0.725	0.742	0.745	0.747	0.745	0.673	0.74
25	0.8	0.809	0.809	0.809	0.809	0.809	0.809	0.807	0.809
41	0.829	0.841	0.844	0.84	0.84	0.839	0.839	0.846	0.841
45	0.857	0.865	0.862	0.866	0.866	0.867	0.866	0.86	0.865
80	0.886	0.897	0.897	0.897	0.897	0.897	0.897	0.897	0.897
110	0.914	0.926	0.926	0.926	0.926	0.926	0.926	0.928	0.926
120	0.943	0.956	0.956	0.956	0.956	0.956	0.956	0.954	0.956
637	0.971	0.985	0.985	0.985	0.985	0.985	0.985	0.985	0.985

Abaixo apresenta-se a distância de Kolmogorov-Smirnoff entre a FDA empírica e a FDA calculada com base nas diferentes funções de kernel, ou seja, $\max |\hat{F}(x) - F_k^*(x)|$, sendo $\hat{F}(x)$ a função distribuição acumulada empírica e $F_k^*(x)$ a função distribuição acumulada obtida a partir da função densidade do kernel k, com $k =$ epanechnikov, uniform, quartic, triagular, triweight, gaussian, cosine e tricube.

kernel_nome	kernel	max_distance_KS
Triweight	triw	0.11
Quartic	quar	0.116
Tricube	tric	0.117
Triangular	tria	0.119
Cosine	cos	0.125
Epanechnikov	epan	0.126
Uniform	unif	0.149
Gaussian	gaus	0.195

Como se pode notar, a função de kernel Triweight responde pela menor distância em relação à função distribuição acumulada empírica.

5.1.2 R

Na tabela abaixo é apresentada uma comparação entre as pseudo-observações e as funções de distribuição acumulada para os diferentes kernels.

```r
kdeVarX <- data.frame(
  varX
 ,pobs = rank( varX ) / ( length( varX ) + 1 )
 ,epan = snpar::kde( x = varX, kernel='epan', plot=FALSE)$Fhat
 ,unif = snpar::kde( x = varX, kernel='unif', plot=FALSE)$Fhat
 ,tria = snpar::kde( x = varX, kernel='tria', plot=FALSE)$Fhat
 ,quar = snpar::kde( x = varX, kernel='quar', plot=FALSE)$Fhat
 ,triw = snpar::kde( x = varX, kernel='triw', plot=FALSE)$Fhat
 ,tric = snpar::kde( x = varX, kernel='tric', plot=FALSE)$Fhat
 ,gaus = snpar::kde( x = varX, kernel='gaus', plot=FALSE)$Fhat
 ,cos  = snpar::kde( x = varX, kernel='cos' , plot=FALSE)$Fhat
)
colnames(kdeVarX)[1] <- varX.label

kdeVarX[ order(kdeVarX[,1]) , ] %>% round(., casas.decimais)
```

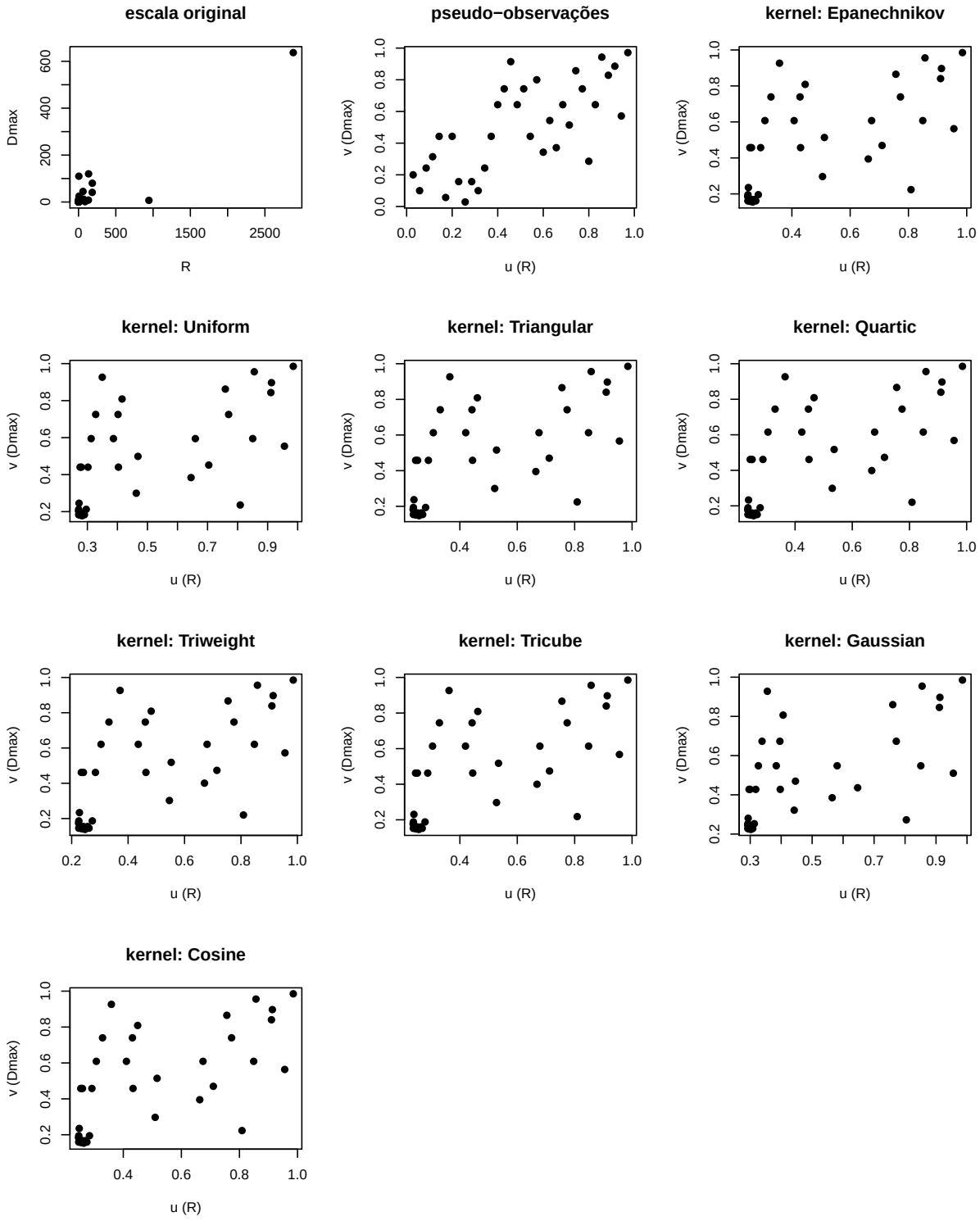
R	pobs	epan	unif	tria	quar	triw	tric	gaus	cos
0.034	0.029	0.248	0.271	0.237	0.234	0.224	0.237	0.292	0.246
0.057	0.057	0.249	0.271	0.238	0.235	0.225	0.238	0.292	0.246
0.066	0.086	0.249	0.271	0.238	0.235	0.225	0.238	0.292	0.247
0.125	0.114	0.251	0.272	0.24	0.237	0.227	0.24	0.293	0.248
0.315	0.143	0.256	0.276	0.246	0.243	0.234	0.245	0.296	0.253
0.342	0.171	0.257	0.276	0.247	0.244	0.235	0.246	0.297	0.254
0.525	0.2	0.261	0.28	0.253	0.25	0.242	0.252	0.3	0.259
0.528	0.229	0.261	0.28	0.253	0.25	0.242	0.252	0.3	0.259
0.684	0.257	0.266	0.283	0.258	0.255	0.247	0.257	0.302	0.264
0.935	0.286	0.272	0.287	0.266	0.263	0.256	0.264	0.306	0.271
1.08	0.314	0.276	0.29	0.27	0.268	0.262	0.269	0.308	0.275
1.4	0.343	0.285	0.296	0.281	0.278	0.273	0.279	0.313	0.283
1.7	0.371	0.293	0.302	0.29	0.288	0.284	0.288	0.318	0.292
2.24	0.4	0.307	0.313	0.308	0.305	0.304	0.305	0.326	0.307
3	0.429	0.328	0.328	0.332	0.33	0.332	0.329	0.338	0.329
4.07	0.457	0.358	0.349	0.365	0.365	0.372	0.362	0.355	0.359
5.9	0.486	0.408	0.387	0.42	0.424	0.436	0.42	0.384	0.411
6.65	0.514	0.428	0.402	0.442	0.447	0.461	0.443	0.396	0.431
6.72	0.543	0.43	0.403	0.444	0.449	0.463	0.445	0.397	0.433
7.31	0.571	0.446	0.415	0.461	0.466	0.482	0.462	0.406	0.449
9.633	0.6	0.505	0.463	0.521	0.53	0.546	0.528	0.442	0.51
9.9	0.629	0.512	0.468	0.528	0.537	0.553	0.535	0.446	0.516
18.6	0.657	0.662	0.645	0.665	0.668	0.671	0.669	0.565	0.663
20	0.686	0.674	0.659	0.676	0.678	0.68	0.679	0.581	0.674
27	0.714	0.709	0.704	0.711	0.713	0.715	0.712	0.647	0.71
60	0.743	0.757	0.759	0.756	0.755	0.754	0.756	0.76	0.757
66.5	0.771	0.773	0.77	0.774	0.774	0.775	0.774	0.772	0.773
87.5	0.8	0.809	0.809	0.809	0.809	0.809	0.809	0.804	0.809
132	0.829	0.849	0.85	0.848	0.848	0.848	0.849	0.851	0.849
135	0.857	0.857	0.855	0.858	0.858	0.858	0.857	0.855	0.857
183.6	0.886	0.91	0.911	0.91	0.91	0.91	0.91	0.911	0.91
184.8	0.914	0.913	0.913	0.914	0.914	0.914	0.914	0.912	0.913
944	0.943	0.956	0.956	0.956	0.956	0.956	0.956	0.956	0.956
2880	0.971	0.985	0.985	0.985	0.985	0.985	0.985	0.985	0.985

Abaixo apresenta-se a distância de Kolmogorov-Smirnoff entre a FDA empírica e a FDA calculada com base nas diferentes funções de kernel, ou seja, $\max |\hat{F}(x) - F_k^*(x)|$, sendo $\hat{F}(x)$ a função distribuição acumulada empírica e $F_k^*(x)$ a função distribuição acumulada obtida a partir da função densidade do kernel k, com $k =$ epanechnikov, uniform, quartic, triagular, triweight, gaussian, cosine e tricube.

kernel_nome	kernel	max_distance_KS
Triweight	triw	0.196
Quartic	quar	0.206
Tricube	tric	0.208
Triangular	tria	0.209
Cosine	cos	0.217
Epanechnikov	epan	0.22
Uniform	unif	0.242
Gaussian	gaus	0.263

Como se pode notar, a função de kernel triw responde pela menor distância em relação à função distribuição acumulada empírica.

5.1.3 Comparação gráfica das funções de kernel



5.2 Estimação da dependência

5.2.1 Marginais em pseudo-observações

1. Cópula paramétrica (MPLE, pacote 'copula')

```
joe     <- fitCopula(     joeCopula(dim=2), data= as.matrix(data.frame(varX.pobs, varY.pobs)))
gumbel  <- fitCopula(  gumbelCopula(dim=2), data= as.matrix(data.frame(varX.pobs, varY.pobs)))
frank   <- fitCopula(   frankCopula(dim=2), data= as.matrix(data.frame(varX.pobs, varY.pobs)))
normal  <- fitCopula(  normalCopula(dim=2), data= as.matrix(data.frame(varX.pobs, varY.pobs)))
t       <- fitCopula(       tCopula(dim=2), data= as.matrix(data.frame(varX.pobs, varY.pobs)))
clayton <- fitCopula( claytonCopula(dim=2), data= as.matrix(data.frame(varX.pobs, varY.pobs)))

##  @loglik   --> log-verossimilhança
##  @estimate --> theta_hat
```

Tabela 6: Bondade do ajuste (por ordem decrescente do BIC)

copula	theta_mple	log_pseudolik	BIC
normal	0.744	11.603	9.84
gumbel	2.055	11.57	9.807
frank	5.826	10.777	9.014
joe	2.548	10.451	8.688
t	0.744 (df=12378.75)	11.603	8.077
clayton	1.254	7.694	5.93

2. Cópula Não-paramétrica (Kernel, pacote 'kdecopula')

```r
## marginais em pseudo-observações
uv        <- data.frame( varX.pobs, varY.pobs)
names(uv) <- c( varX.label, varY.label )

## ajuste não-paramétrico por 3 métodos: T, TLL1 e TLL2
bic.method.T    <- round(
  BIC( fit.kdecop.T    <- kdecopula::kdecop( uv, method = 'T'    ) ) / (-2)
 ,digits=casas.decimais
)
bic.method.TLL1 <- round(
  BIC( fit.kdecop.TLL1 <- kdecopula::kdecop( uv, method = 'TLL1' ) ) / (-2)
 ,digits=casas.decimais
)
bic.method.TLL2 <- round(
  BIC( fit.kdecop.TLL2 <- kdecopula::kdecop( uv, method = 'TLL2' ) ) / (-2)
 ,digits=casas.decimais
)

nonpar_pobs_bestfit <- fit.kdecop.T
summary(nonpar_pobs_bestfit)

## plot(uv, pch=19)
## summary( fit <- kdecopula::kdecop( uv, method = 'TLL2' ) )
## paste( 'BIC_equiv_copula =', round( BIC(fit)/(-2) ,2) )
## subset( ajustes, copula == 'normal')
## plot(fit)
```

```
Kernel copula density estimate (tau = 0.47)
-------------------------------
Variables:    R -- Dmax
Observations: 34
Method:       Transformation estimator ('T')
Bandwidth:    matrix(c(0.51, 0.36, 0, 0.37), 2, 2)
---
logLik: 14.49    AIC: -25.45    cAIC: -25.14    BIC: -22.76
Effective number of parameters: 1.77
```

5.2.2 Marginais em funções de kernel

```
varX.kern <- kdeVarX[ , varX_KS[ 1, 'kernel' ] ]
varY.kern <- kdeVarY[ , varY_KS[ 1, 'kernel' ] ]
```

1. Cópula paramétrica (MPLE, pacote 'copula')

```
joe     <- fitCopula(     joeCopula(dim=2), data= as.matrix(data.frame(varX.kern, varY.kern)))
gumbel  <- fitCopula(  gumbelCopula(dim=2), data= as.matrix(data.frame(varX.kern, varY.kern)))
frank   <- fitCopula(   frankCopula(dim=2), data= as.matrix(data.frame(varX.kern, varY.kern)))
normal  <- fitCopula(  normalCopula(dim=2), data= as.matrix(data.frame(varX.kern, varY.kern)))
t       <- fitCopula(       tCopula(dim=2), data= as.matrix(data.frame(varX.kern, varY.kern)))
clayton <- fitCopula( claytonCopula(dim=2), data= as.matrix(data.frame(varX.kern, varY.kern)))

##  @loglik   --> log-verossimilhança
##  @estimate --> theta_hat
```

Tabela 7: Bondade do ajuste (por ordem decrescente do BIC)

copula	theta_mple	log_pseudolik	BIC
clayton	2.386	12.266	10.503
normal	0.752	10.834	9.071
gumbel	1.993	10.231	8.468
t	0.714 (df=2.087)	11.951	8.425
frank	5.711	9.176	7.413
joe	2.177	8.547	6.784

2. Cópula Não-paramétrica (Kernel, pacote 'kdecopula')

```r
## marginais em kernel
uv        <- data.frame( varX.kern, varY.kern )
names(uv) <- c( varX.label, varY.label )

## ajuste não-paramétrico por 3 métodos: T, TLL1 e TLL2
bic.method.T    <- round(
  BIC( fit.kdecop.T    <- kdecopula::kdecop( uv, method = 'T'    ) ) / (-2)
 ,digits=casas.decimais
)
bic.method.TLL1 <- round(
  BIC( fit.kdecop.TLL1 <- kdecopula::kdecop( uv, method = 'TLL1' ) ) / (-2)
 ,digits=casas.decimais
)
bic.method.TLL2 <- round(
  BIC( fit.kdecop.TLL2 <- kdecopula::kdecop( uv, method = 'TLL2' ) ) / (-2)
 ,digits=casas.decimais
)

nonpar_kern_bestfit <- fit.kdecop.T
summary(nonpar_kern_bestfit)

## plot(uv, pch=19)
## summary( fit <- kdecopula::kdecop( uv, method = 'TLL2' ) )
## paste( 'BIC_equiv_copula =', round( BIC(fit)/(-2) ,2) )
## subset( ajustes, copula == 'normal')
## plot(fit)
```

```
Kernel copula density estimate (tau = 0.48)
-------------------------------
Variables:    R -- Dmax
Observations: 34
Method:       Transformation estimator ('T')
Bandwidth:    matrix(c(0.46, 0.34, 0, 0.37), 2, 2)
---
logLik: 15.1    AIC: -26.8    cAIC: -26.51    BIC: -24.2
Effective number of parameters: 1.7
```

5.3 $Prob(V > v|U \in (a,b])$

$$Prob(V > v|U \in (a,b]) = 1 - \frac{C(b,x)}{b-a} + \frac{C(a,x)}{b-a}$$

Probabilidade condicional $Prob(V > v|U \in (a,b])$, utilizando a cópula paramétrica (estimação do parâmetro de associação via máxima pseudo-verossimilhança) calculada com base nas marginais reescaladas pelas pseudo-observações (ref. aos postos das observações originais).

```
## Cálculo da Probabilidade Condicional Prob( V > v | U \in (a,b] )
##
## Estimação semi-paramétrica
##   marginais corrigidas pelas pseudo-observações
##   e cópula paramétrica estimada via MPLE
##
## Análise para (1) Dmax ~ R and (2) Dmax ~ Rf
prob_Vgtv_given_UinInterval_semipar <- function( v, a, b, theta, family ) {
    ## Expression for P( V>v | U \in (a,b], \theta )
    if ( v<0 || v>1 ) stop('v outside domain [0,1].          ')
    if ( a<0 || a>1 ) stop('a in (a,b] outside domain [0,1]. ')
    if ( b<0 || b>1 ) stop('b in (a,b] outside domain [0,1]. ')
    if ( a>b )        stop('a greater then b in (a,b]        ')
    aux_b    <- pCopula( c(b,v), family( param=theta, dim=2) ) / ( b - a )
    aux_a    <- pCopula( c(a,v), family( param=theta, dim=2) ) / ( b - a )
    result   <- 1 - ( (aux_b) - (aux_a) )
    return( round( result, digits = casas.decimais ) )
}
## check: prob_Vgtv_given_UinInterval_semipar( v=0.9, a=0.75, b=1, theta=3, family = gumbelCopula )
```

a	b	ranks	prob_Vgtv_given_U	estimation	R_a	R_b	Dmax
0	0.25	0.7	0.023	semipar_pobs	0.0342	0.684	12
0	0.25	0.75	0.014	semipar_pobs	0.0342	0.684	12
0	0.25	0.8	0.007	semipar_pobs	0.0342	0.684	25
0	0.25	0.85	0.003	semipar_pobs	0.0342	0.684	45
0	0.25	0.9	0.001	semipar_pobs	0.0342	0.684	80
0	0.25	0.95	0	semipar_pobs	0.0342	0.684	120
0.25	0.5	0.7	0.131	semipar_pobs	0.684	6.65	12
0.25	0.5	0.75	0.09	semipar_pobs	0.684	6.65	12
0.25	0.5	0.8	0.057	semipar_pobs	0.684	6.65	25
0.25	0.5	0.85	0.031	semipar_pobs	0.684	6.65	45
0.25	0.5	0.9	0.013	semipar_pobs	0.684	6.65	80
0.25	0.5	0.95	0.003	semipar_pobs	0.684	6.65	120
0.5	0.75	0.7	0.34	semipar_pobs	6.65	60	12
0.5	0.75	0.75	0.263	semipar_pobs	6.65	60	12
0.5	0.75	0.8	0.19	semipar_pobs	6.65	60	25
0.5	0.75	0.85	0.122	semipar_pobs	6.65	60	45
0.5	0.75	0.9	0.064	semipar_pobs	6.65	60	80
0.5	0.75	0.95	0.02	semipar_pobs	6.65	60	120
0.75	1	0.7	0.706	semipar_pobs	60	2880	12
0.75	1	0.75	0.633	semipar_pobs	60	2880	12
0.75	1	0.8	0.546	semipar_pobs	60	2880	25
0.75	1	0.85	0.443	semipar_pobs	60	2880	45
0.75	1	0.9	0.322	semipar_pobs	60	2880	80
0.75	1	0.95	0.177	semipar_pobs	60	2880	120

Probabilidade condicional $Prob(V > v|U \in (a, b])$, utilizando a cópula não-paramétrica (estimação do parâmetro de associação via funções de kernel) calculada com base nas marginais reescaladas pelas funções de distribuição acumuladas provenientes da estimação das funções densidade de probabilidade obtidas via funções de kernel.

```
## Estimação não-paramétrica completa
##   marginais corrigidas pelas FDA baseadas na fdp de kernel
##   e cópula não-paramétrica estimada via pacote kdecopula
##
prob_Vgtv_given_UinInterval_nonpar <- function( v, a, b, nonpar_copula ) {
    ## Expression for P( V>v | U \in (a,b], \theta )
    if ( v<0 || v>1 ) stop('v outside domain [0,1].          ')
    if ( a<0 || a>1 ) stop('a in (a,b] outside domain [0,1]. ')
    if ( b<0 || b>1 ) stop('b in (a,b] outside domain [0,1]. ')
    if ( a>b )        stop('a greater then b in (a,b]        ')
    aux_b    <- pkdecop( c(b,v), nonpar_copula ) / ( b - a )
    aux_a    <- pkdecop( c(a,v), nonpar_copula ) / ( b - a )
    result   <- 1 - ( (aux_b) - (aux_a) )
    return( round( result, digits = casas.decimais) )
}
## check: prob_Vgtv_given_UinInterval_nonpar( v=0.9, a=0.75, b=1, nonpar_copula = fit.kdecop.T )
```

a	b	ranks	prob_Vgtv_given_U	estimation	R_a	R_b	Dmax
0	0.25	0.7	0.057	nonpar_kern	0.0342	0.684	12
0	0.25	0.75	0.046	nonpar_kern	0.0342	0.684	12
0	0.25	0.8	0.037	nonpar_kern	0.0342	0.684	25
0	0.25	0.85	0.03	nonpar_kern	0.0342	0.684	41
0	0.25	0.9	0.023	nonpar_kern	0.0342	0.684	80
0	0.25	0.95	0.019	nonpar_kern	0.0342	0.684	120
0.25	0.5	0.7	0.202	nonpar_kern	0.684	7.31	12
0.25	0.5	0.75	0.152	nonpar_kern	0.684	7.31	12
0.25	0.5	0.8	0.107	nonpar_kern	0.684	7.31	25
0.25	0.5	0.85	0.069	nonpar_kern	0.684	7.31	41
0.25	0.5	0.9	0.038	nonpar_kern	0.684	7.31	80
0.25	0.5	0.95	0.012	nonpar_kern	0.684	7.31	120
0.5	0.75	0.7	0.346	nonpar_kern	7.31	60	12
0.5	0.75	0.75	0.277	nonpar_kern	7.31	60	12
0.5	0.75	0.8	0.208	nonpar_kern	7.31	60	25
0.5	0.75	0.85	0.139	nonpar_kern	7.31	60	41
0.5	0.75	0.9	0.075	nonpar_kern	7.31	60	80
0.5	0.75	0.95	0.024	nonpar_kern	7.31	60	120
0.75	1	0.7	0.601	nonpar_kern	60	2880	12
0.75	1	0.75	0.531	nonpar_kern	60	2880	12
0.75	1	0.8	0.454	nonpar_kern	60	2880	25
0.75	1	0.85	0.368	nonpar_kern	60	2880	41
0.75	1	0.9	0.269	nonpar_kern	60	2880	80
0.75	1	0.95	0.151	nonpar_kern	60	2880	120

```
## Comparação da probabilidade condicional usando marginais
##   em escala [0,1]
##
##
cond_prob_compara_Dmax_Rf <- cond_prob_compara

```

```r
cond_prob_compara_Dmax_R <- bind_rows(
  ## estimação semiparamétrica usando pseudo-observações
  cond_prob_semipar %>%
  rename( limiar = ranks) %>%
  mutate(
    intervalo = paste0('(',a,', ',b,']')
   ,modelo = paste0('02_',varY.label,'_',varX.label)
   ,marginais = '03_pobs'
  ) %>%
  select( intervalo, limiar, modelo, marginais, prob_Vgtv_given_U)
 ,
  ## estimação não-parametrica usando marginais corrigidas pelo kernel
  cond_prob_nonpar %>%
  rename( limiar = ranks) %>%
  mutate(
    intervalo = paste0('(',a,', ',b,']')
   ,modelo = paste0('02_',varY.label,'_',varX.label)
   ,marginais = '04_kernel'
  ) %>%
  select( intervalo, limiar, modelo, marginais, prob_Vgtv_given_U)
)

## consolida resultados dos modelos Dmax_Rf e Dmax_R
cond_prob_compara <- bind_rows(
  cond_prob_compara_Dmax_Rf
 ,cond_prob_compara_Dmax_R
)

## Resultados para o paper/livro
cond_prob_compara_resultados <- cond_prob_compara %>%
  rename(
    prob_cond = prob_Vgtv_given_U
  ) %>%
  pivot_wider(
    id_cols = c('intervalo','limiar')
    ,names_from = c('modelo','marginais')
    ,values_from = prob_cond
  ) %>%
  as.data.frame

for(i in 2:nrow(cond_prob_compara_resultados)){
  cond_prob_compara_resultados[i, 'intervalo'] <- ifelse( cond_prob_compara_resultados[i, 'limiar'] == 0.7
}

## exporta resultados da comparação entre semipar e nonpar
cond_prob_compara_resultados %>%
  xtable( digits= casas.decimais ) %>%
  print(
    only.contents = FALSE
   ,include.colnames = TRUE
   ,include.rownames = FALSE
   ,hline.after = NULL
   ,comment=FALSE
  ) %>%
  cat(
    file = './dados/cond_prob_compara_formato_longo.tex'
  )


## Probabilidade condicional usando marginais
##   em escala original e estimação não-paramétrica completa
##
```

```r
##
cond_prob_escala_original_Dmax_R <- cond_prob_nonpar %>%
  rename(
    prob_cond = prob_Vgtv_given_U
    ,limiar = Dmax
  ) %>%
  mutate(
    intervalo = paste0('(', R_a,', ',R_b,']')
   ,intervalo = ifelse( ranks == 0.70, intervalo, '')
  ) %>%
  select( intervalo, limiar, prob_cond )

## exporta resultados da comparação entre semipar e nonpar
cond_prob_escala_original_Dmax_R %>%
  xtable( digits= c(0,0,1,casas.decimais) ) %>%
  print(
    only.contents = FALSE
   ,include.colnames = TRUE
   ,include.rownames = FALSE
   ,hline.after = NULL
   ,comment=FALSE
  ) %>%
  cat(
    file = './dados/cond_prob_escala_original_Dmax_R_formato_longo.tex'
  )
```

6 Referências

LARRAURI, P. C.; LALL, U. Tailings dams failures: updated statistical model for discharge volume and runout. **Environments**, Multidisciplinary Digital Publishing Institute, v. 5, n. 2, p. 28, 2018.