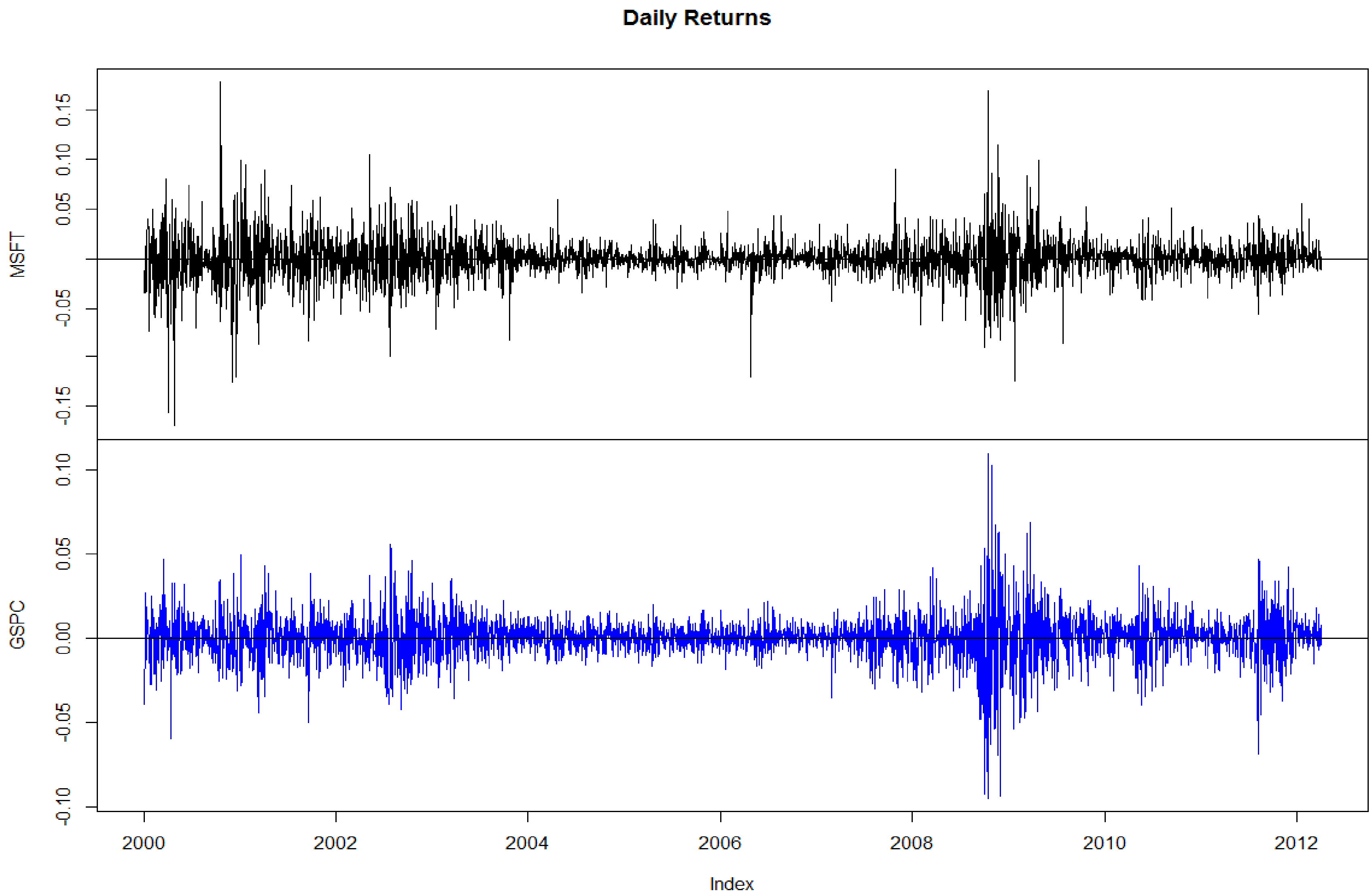
COMPUTATIONAL FINANCE & RISK MANAGEMENT
UNIVERSITY of WASHINGTON
Department of Applied Mathematics

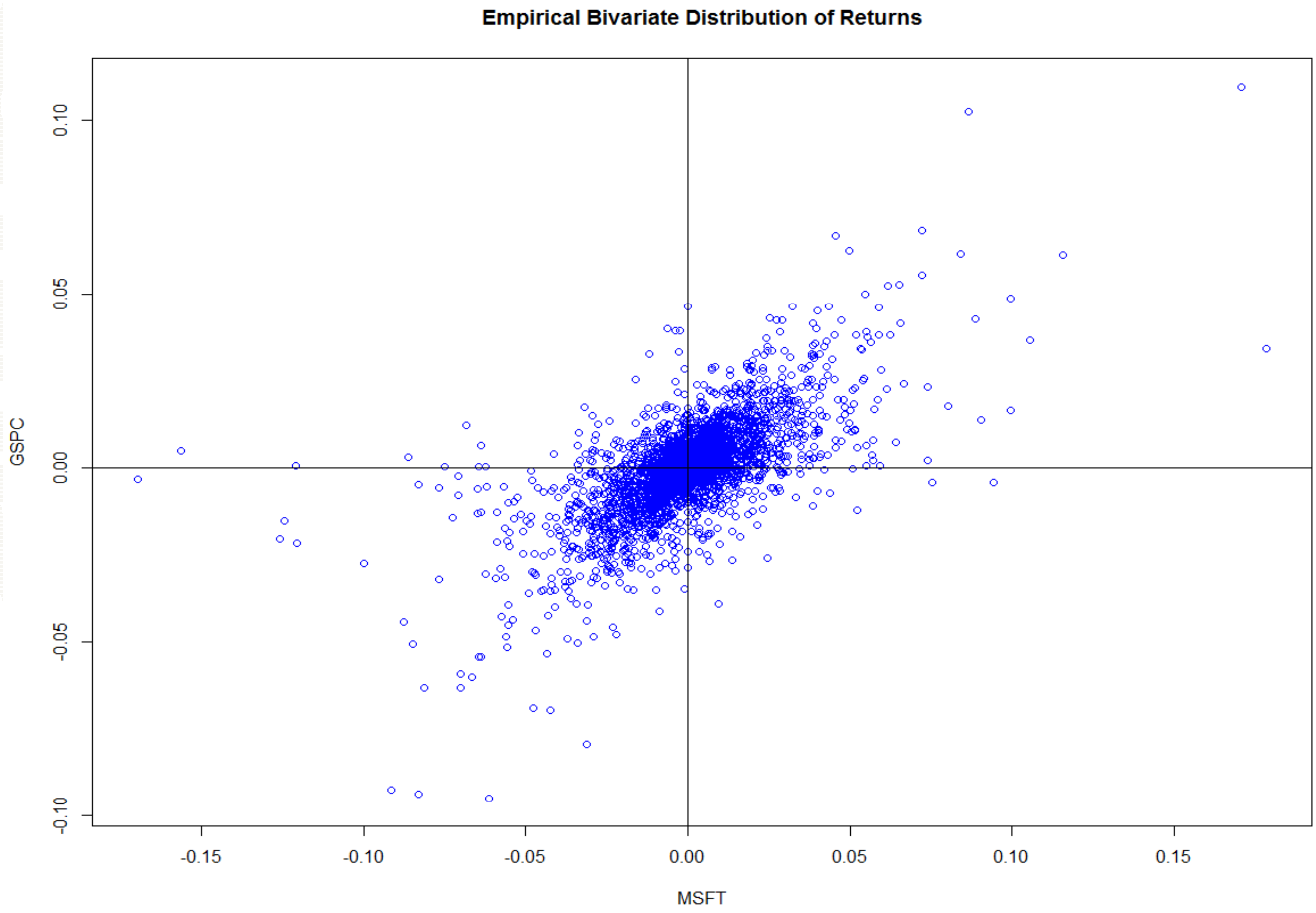
Copulas

Amath 546/Econ 589
Eric Zivot
Spring 2013
Updated: May 22, 2013

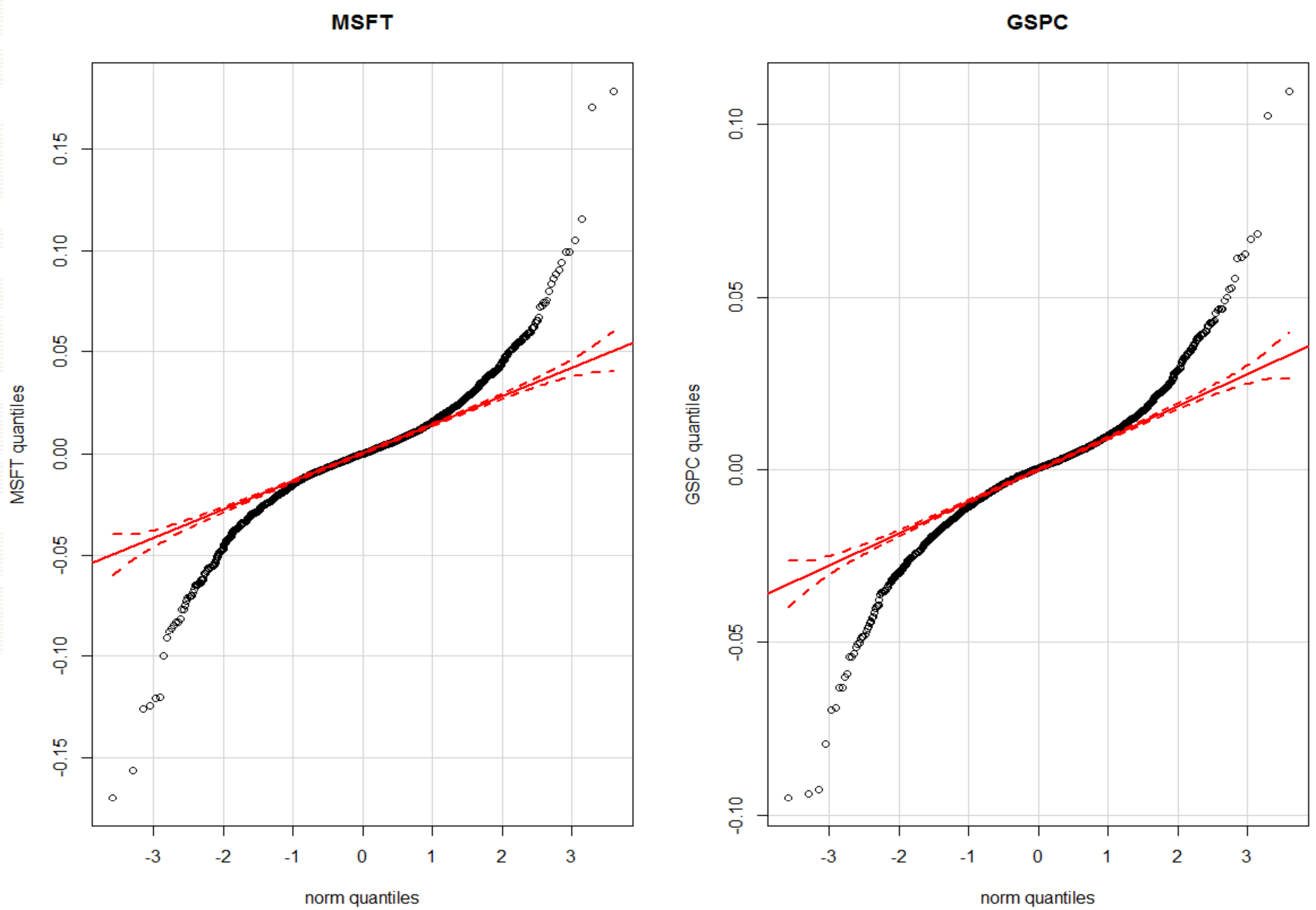
Example Data



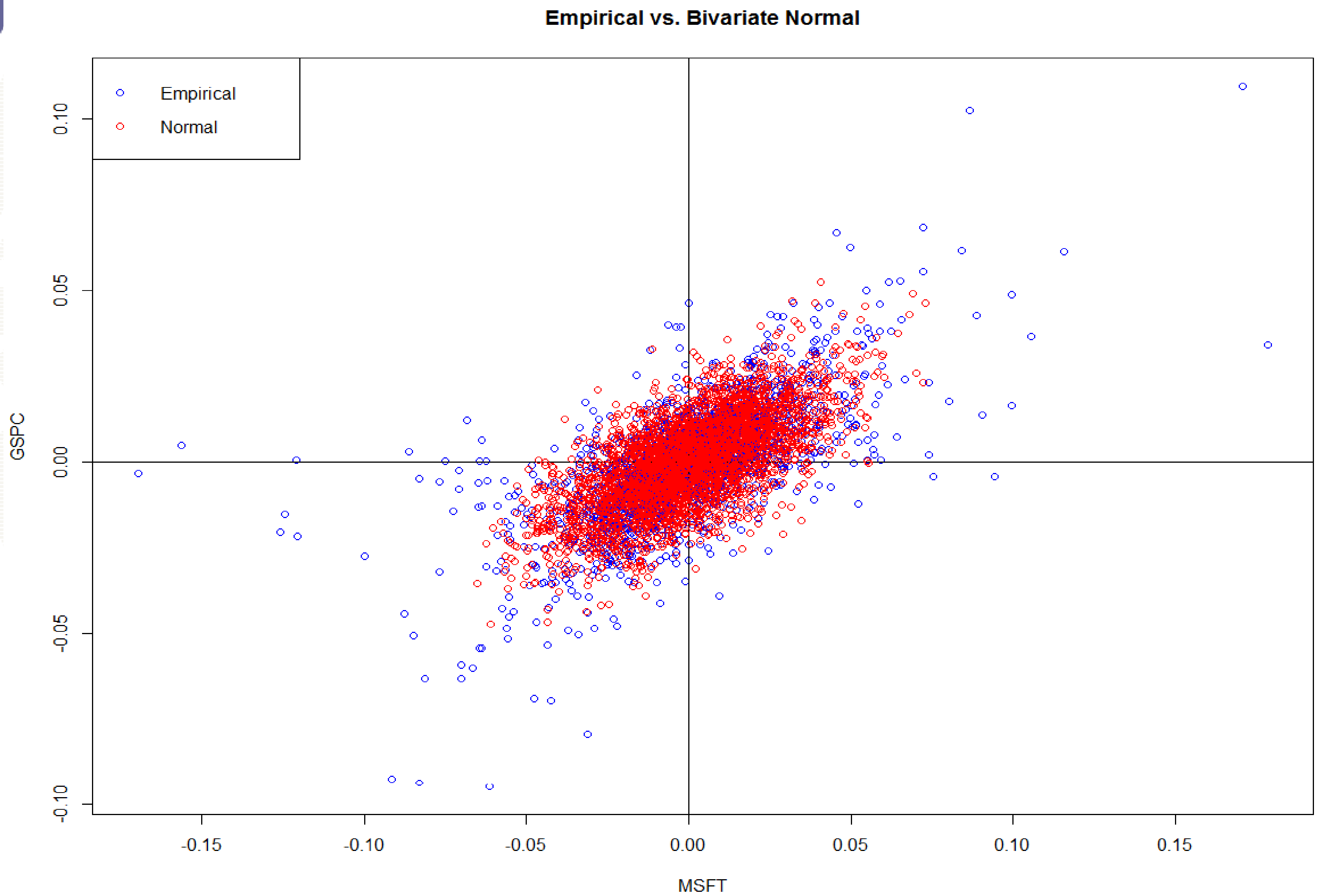
Non-Normal Bivariate Distribution



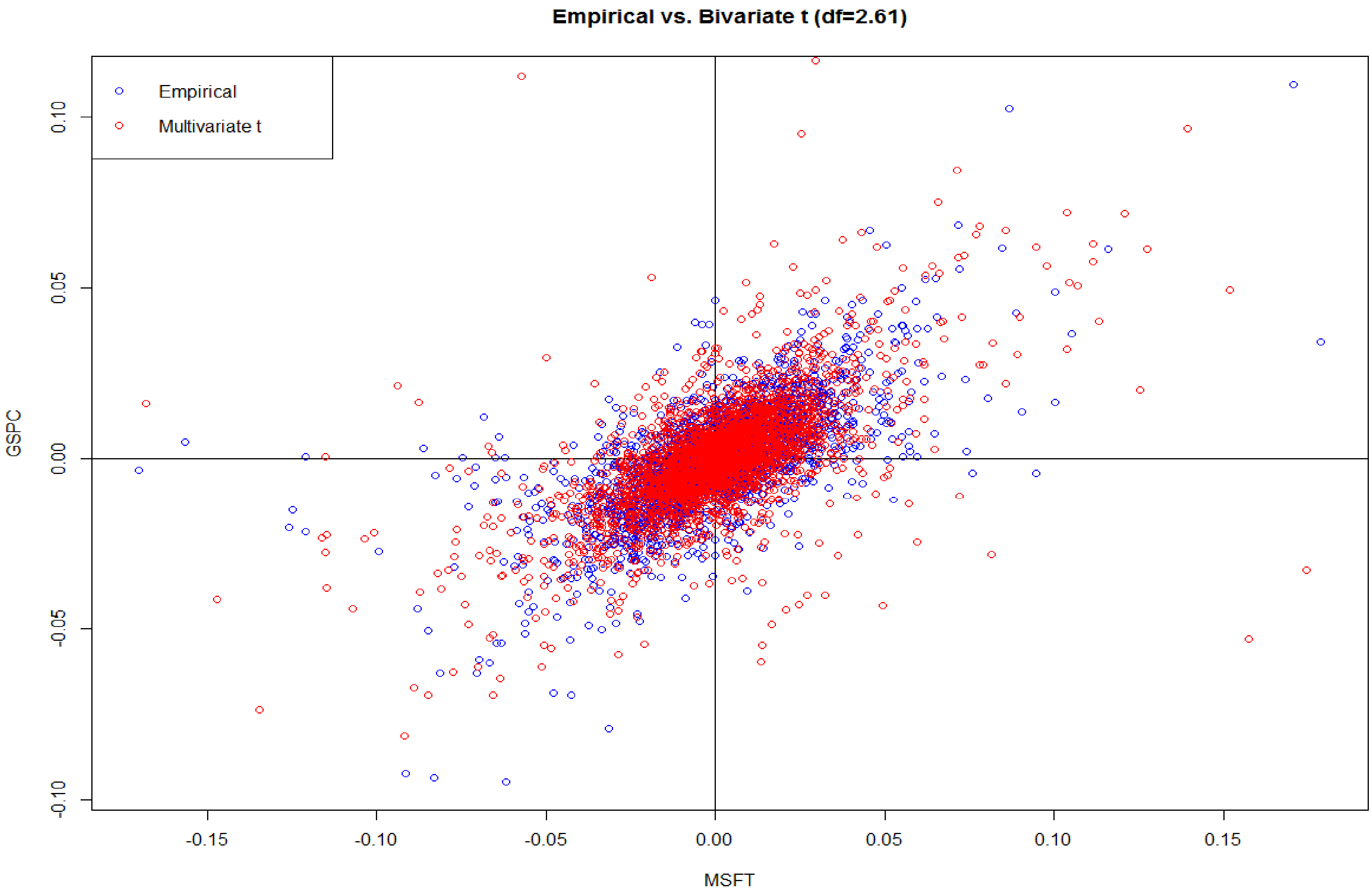
Non-Normal Marginal Distributions



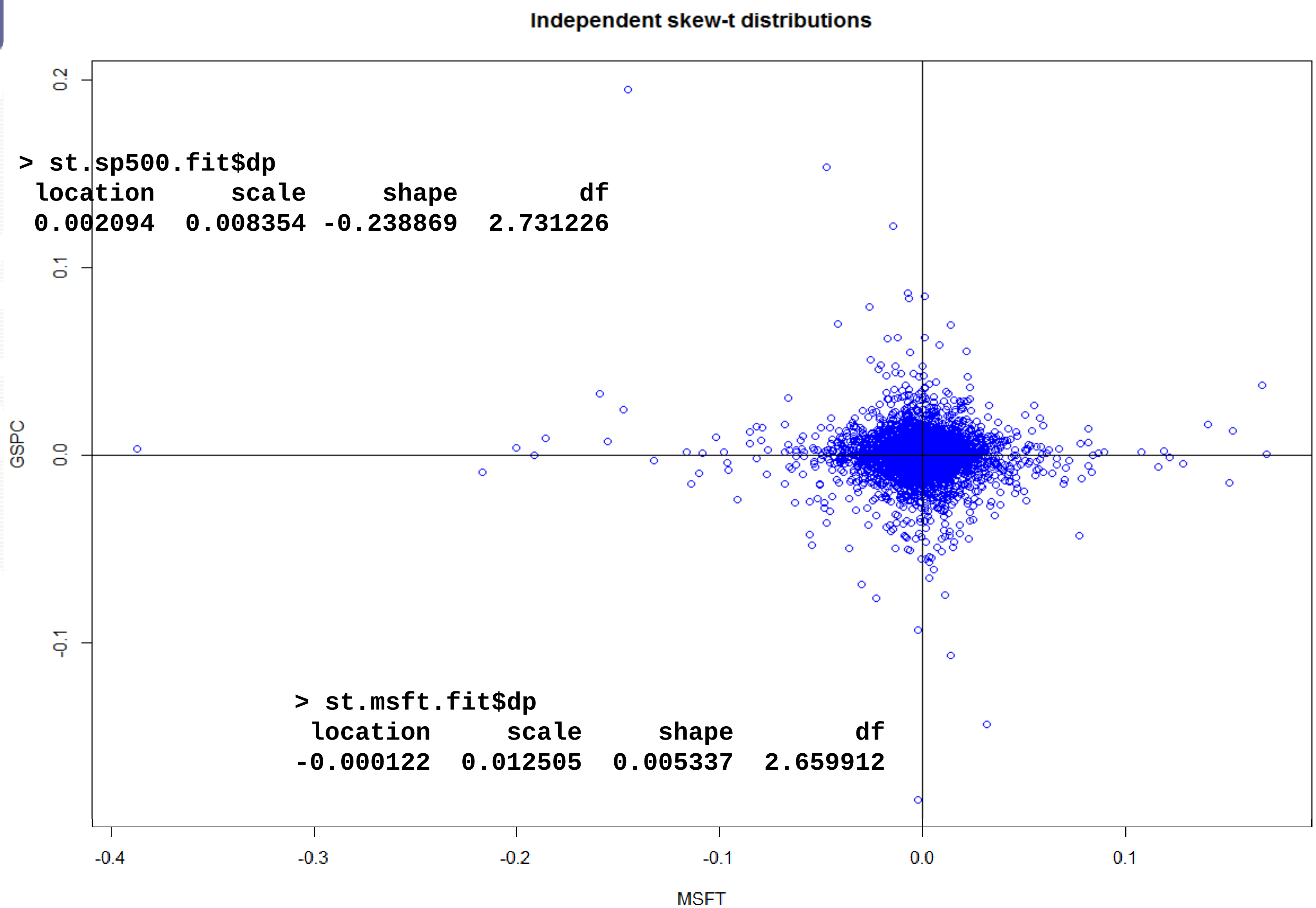
Bivariate Normal Misses Tail Dependence



Bivariate t Imposes Symmetry and Same df



Independent Skew-t Misses Dependence



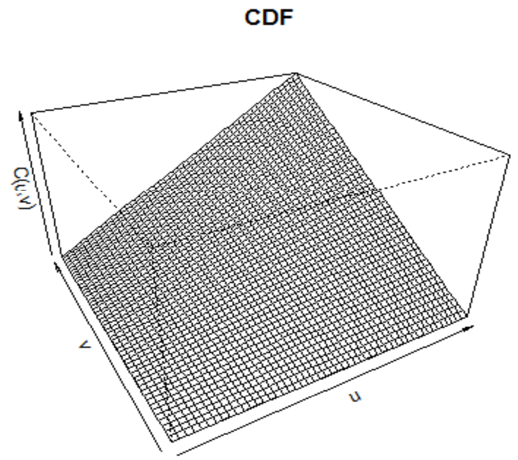
Independent Copula

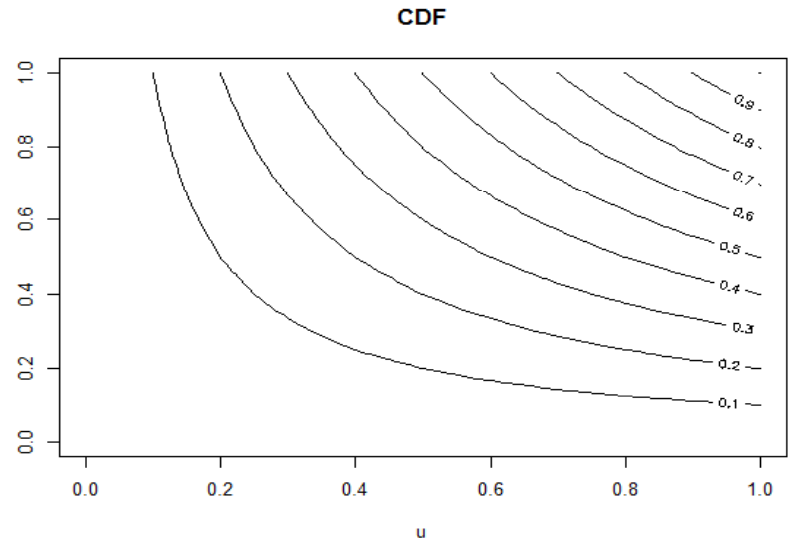
```
> indep.cop = indepCopula(2)
> class(indep.cop)
[1] "indepCopula"
attr(,"package")
[1] "copula“

> slotNames(indep.cop)
[[1] "dimension"    "parameters"   "param.names"   "param.lowbnd"
[5] "param.upbnd"  "fullname"     "exprdist"

> indep.cop
Independence copula
Dimension:  2

# simulate data from copula
> u = rcopula(indep.cop, 200)
> head(u)
        [,1]   [,2]
[1,] 0.16432 0.5323
[2,] 0.78620 0.2506
[3,] 0.39234 0.8470
[4,] 0.06112 0.7300
[5,] 0.36492 0.4187
[6,] 0.17866 0.4047
```

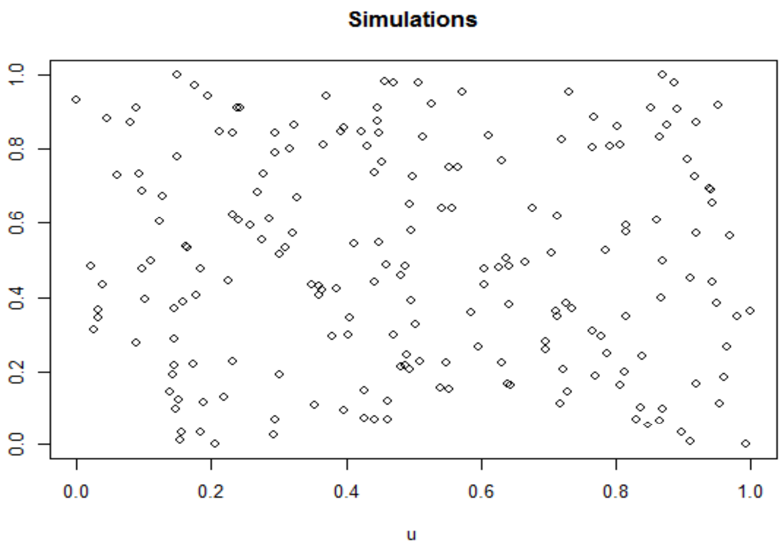
Independent Copula

```
# plots of CDF and simulated values
> par(mfrow=c(2,2))
>   persp(indep.cop, pcopula, main="CDF",
+         xlab="u", ylab="v", zlab="C(u,v)")
>   contour(indep.cop, pcopula, main="CDF",
+           xlab="u", ylab="v")
>   plot(u, main="Simulations",
+        xlab="u", ylab="v")
> par(mfrow=c(1,1))
```

Independent Copula







Simulations from the independent copula gives independent uniform values in the unit square

Note: bivariate density plot is a cube and does not plot nicely.

Dependence Measures

```
# pearson's linear correlation
> cor(MSFT.GSPC.ret, method="pearson")[1,2]
[1] 0.67

# Kendall's tau
> cor(MSFT.GSPC.ret, method="kendall")[1,2]
[1] 0.49

# Spearman's rho
> cor(MSFT.GSPC.ret, method="spearman")[1,2]
[1] 0.66
```

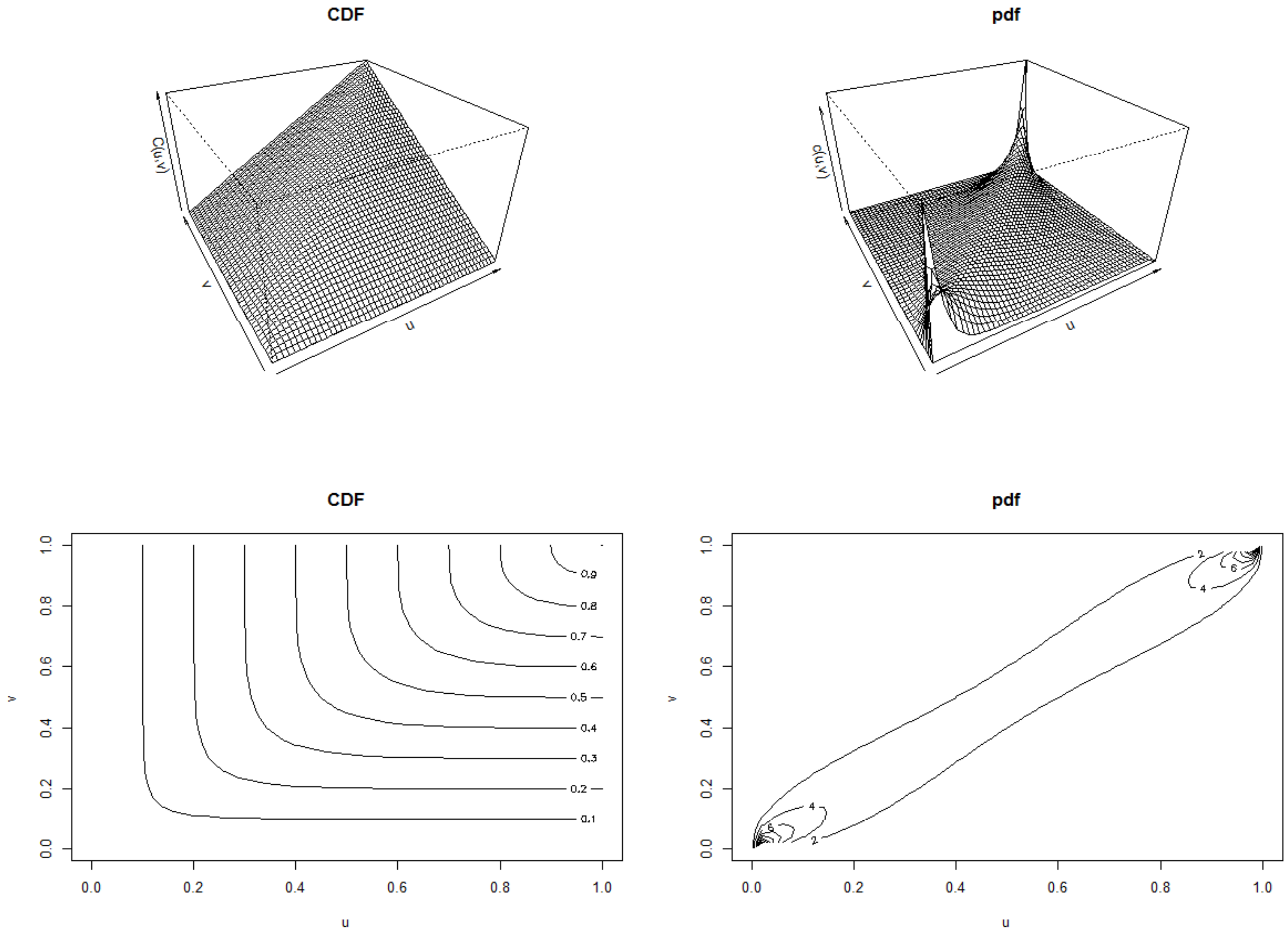
Bivariate Gaussian Copula

```
> norm.cop.9 = normalCopula(param=0.9, dim=2)
> class(norm.cop.9)
[1] "normalCopula"
attr(,"package")
[1] "copula“

> slotNames(norm.cop.9)
[1] "dispstr"      "getRho"       "dimension"    "parameters"
[5] "param.names"  "param.lowbnd" "param.upbnd"  "fullname"

> norm.cop.9
Normal copula family
Dimension:  2
Parameters:
   rho.1  =  0.9

# Method functions
dCopula, pCopula, rCopula, tau, rho, tailIndex
```

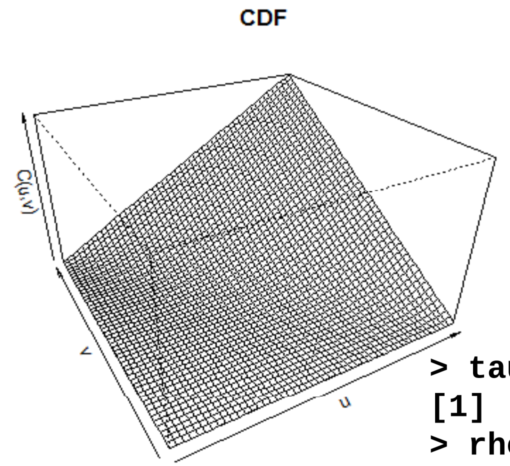
Bivariate Gaussian Copula

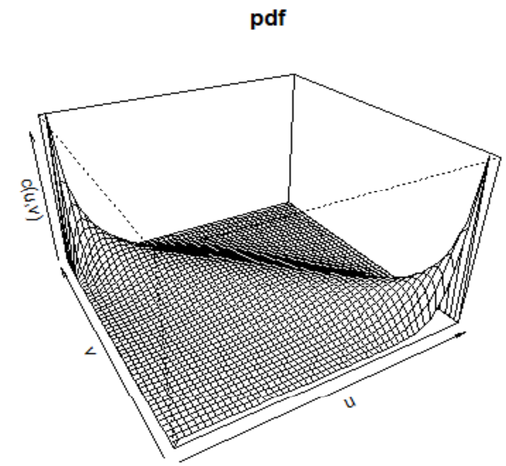
```
# plot copula CDF, pdf and contours
> par(mfrow=c(2,2))
>      persp(norm.cop.9, pcopula, main="CDF",
+             xlab="u", ylab="v", zlab="C(u,v)")
>      persp(norm.cop.9, dcopula, main="pdf",
+             xlab="u", ylab="v", zlab="c(u,v)")
>      contour(norm.cop.9, pcopula, main="CDF",
+             xlab="u", ylab="v")
>      contour(norm.cop.9, dcopula, main="pdf",
+             xlab="u", ylab="v")
> par(mfrow=c(1,1))

# compute Kendall's tau, Spreaman's rho and tailindex
> tau(norm.cop.9)
[1] 0.7129
> rho(norm.cop.9)
[1] 0.8915
> tailIndex(norm.cop.9)
lower upper
    0     0
```

Gaussian Copula: $\rho = 0.9$

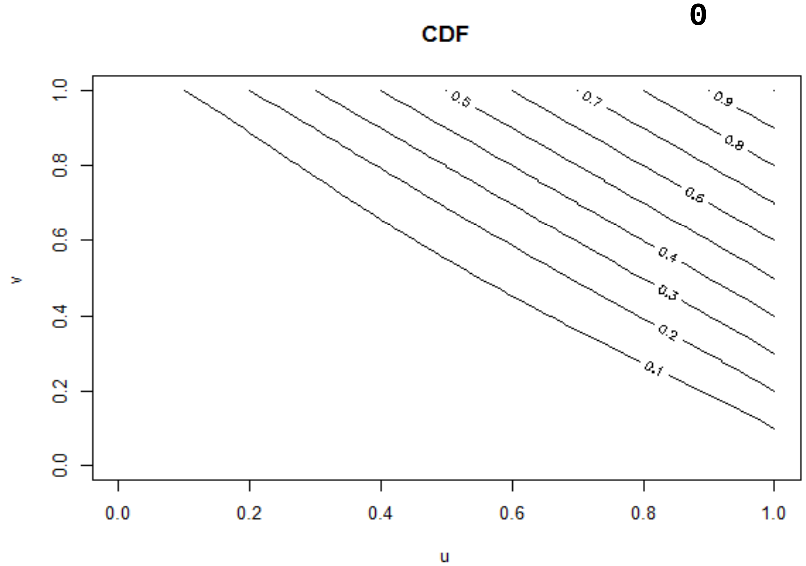


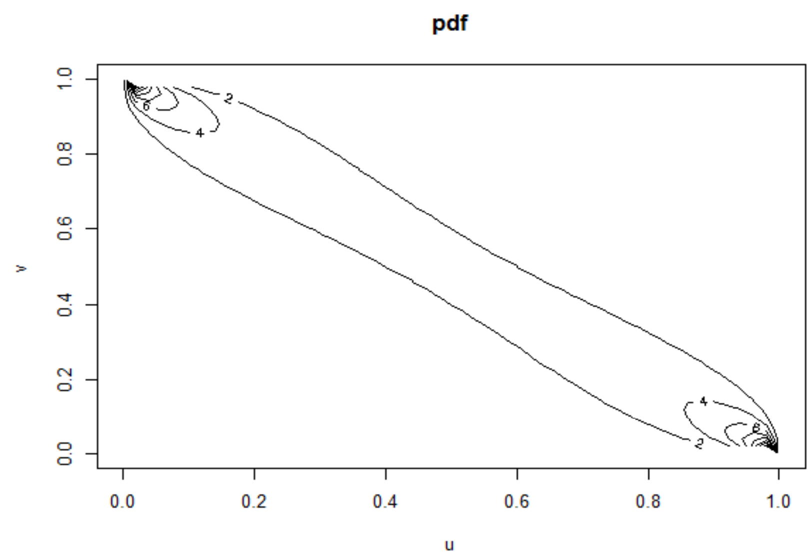
Gaussian Copula: ρ = -0.9



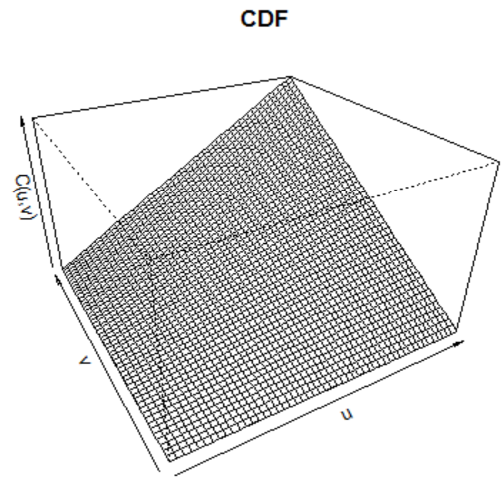



```
> tau(norm.cop.m9)
[1] -0.7129
> rho(norm.cop.m9)
[1] -0.8915
> tailIndex(norm.cop.m9)
lower upper
    0     0
```

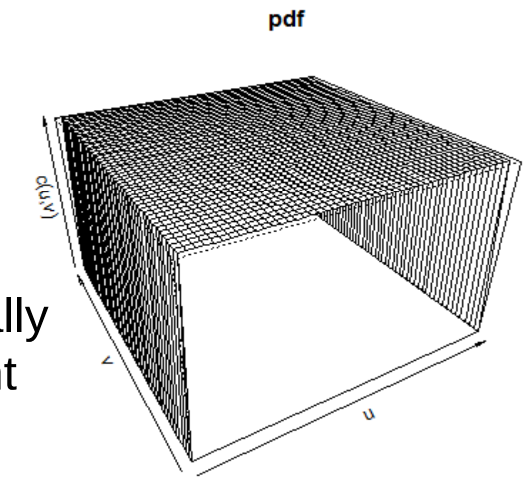


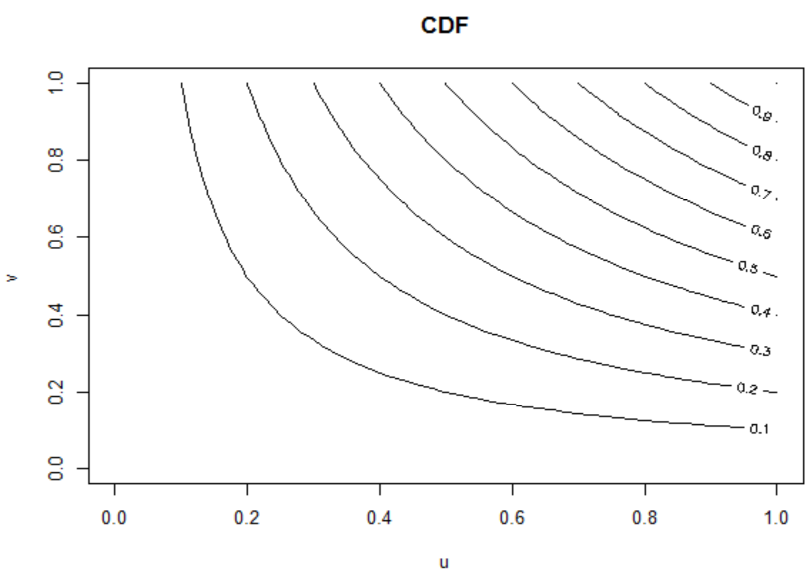


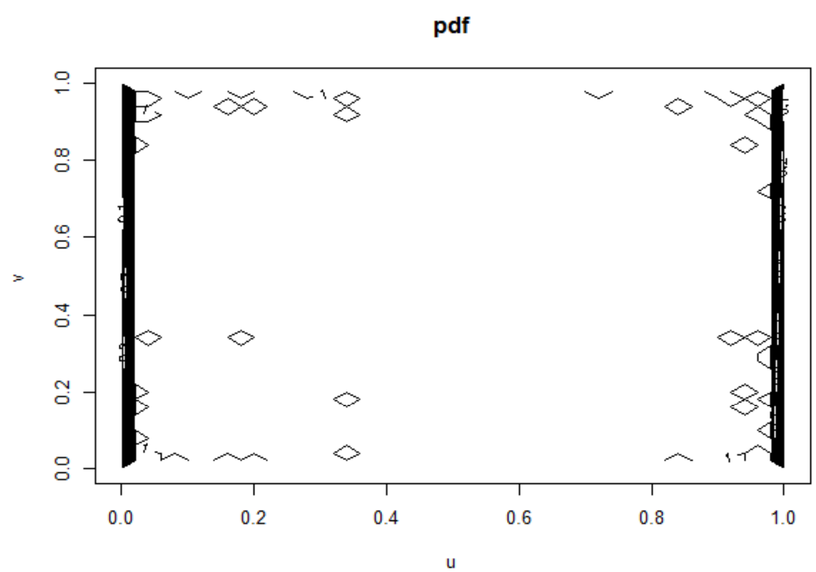
Gaussian Copula: $\rho = 0$



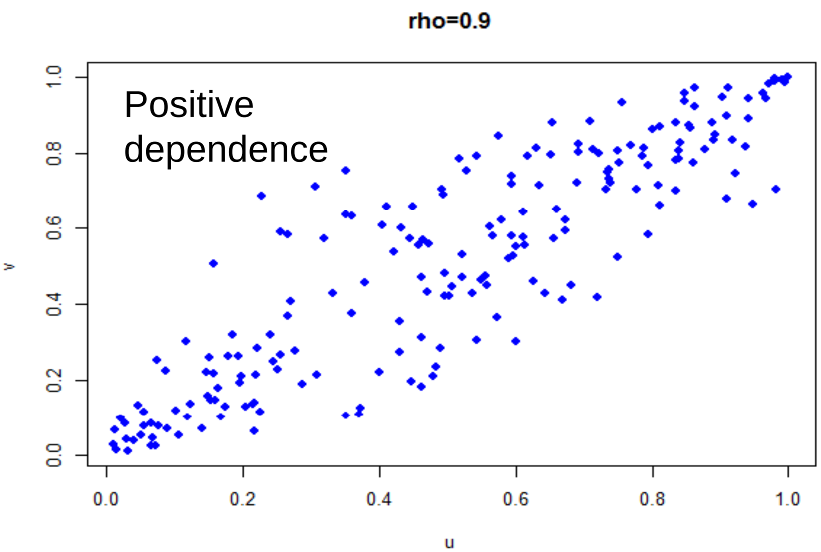
This is essentially the independent copula

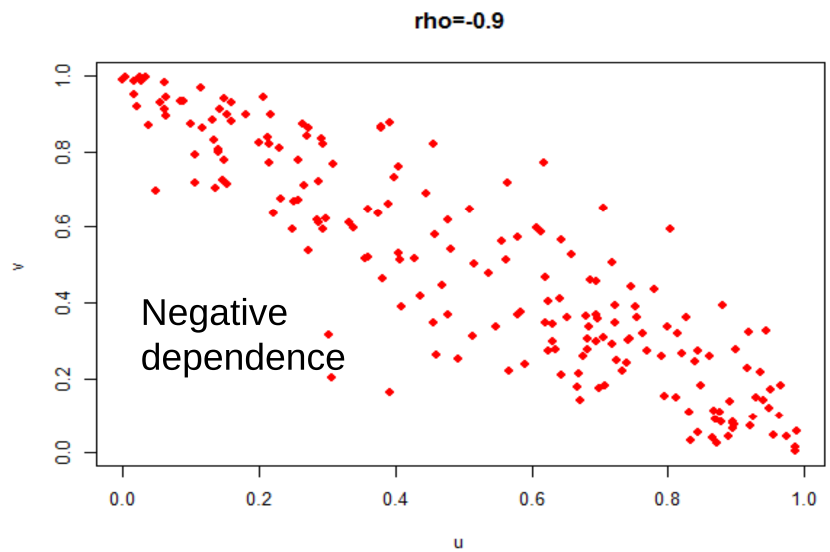


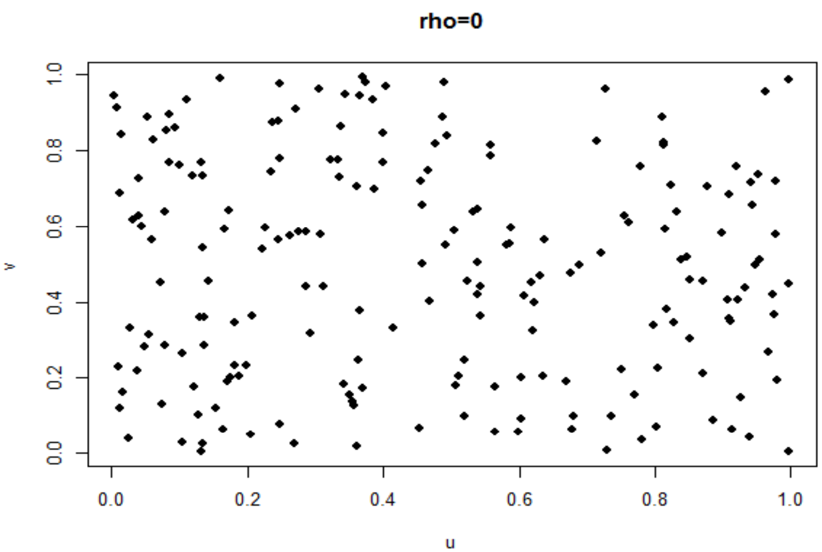




Simulations from Gaussian Copulas



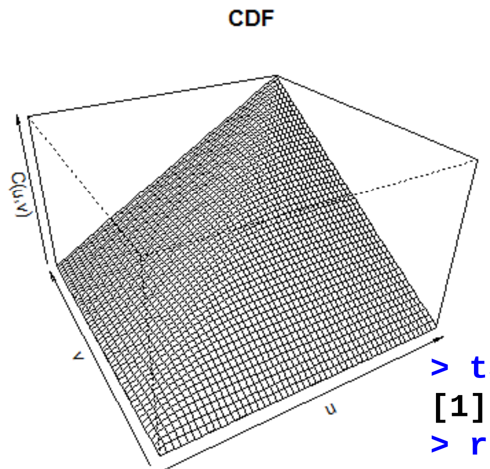


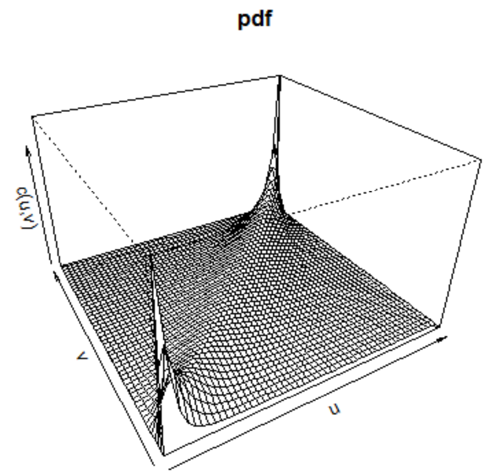


No dependence

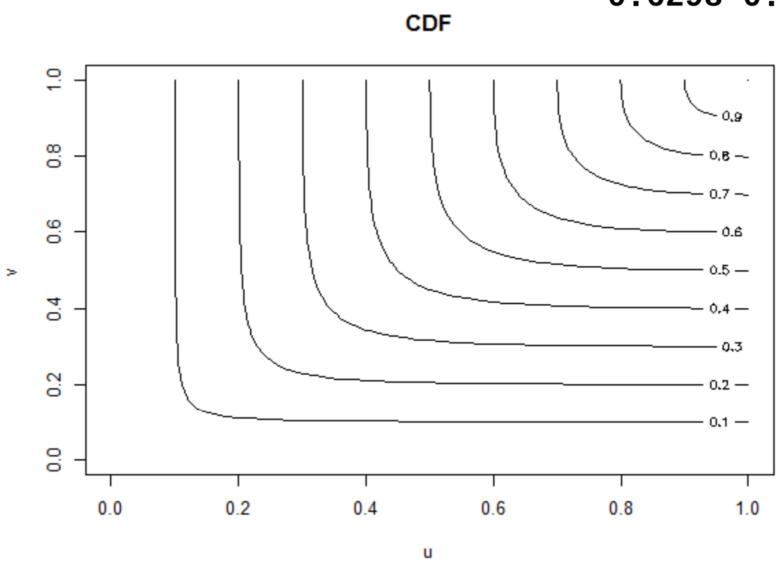
Student t Copula: ρ = 0.9, df=4

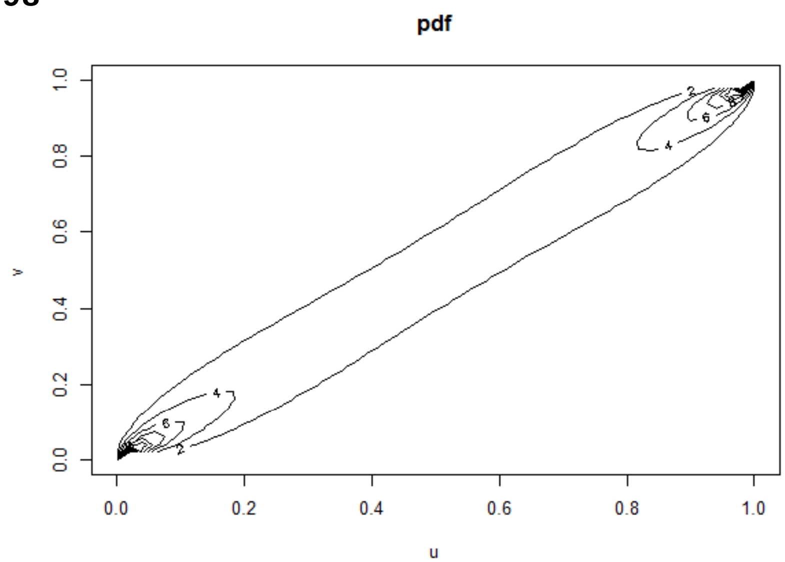
```
> t.cop.9 = tCopula(param=0.9, dim=2, df=4)
```



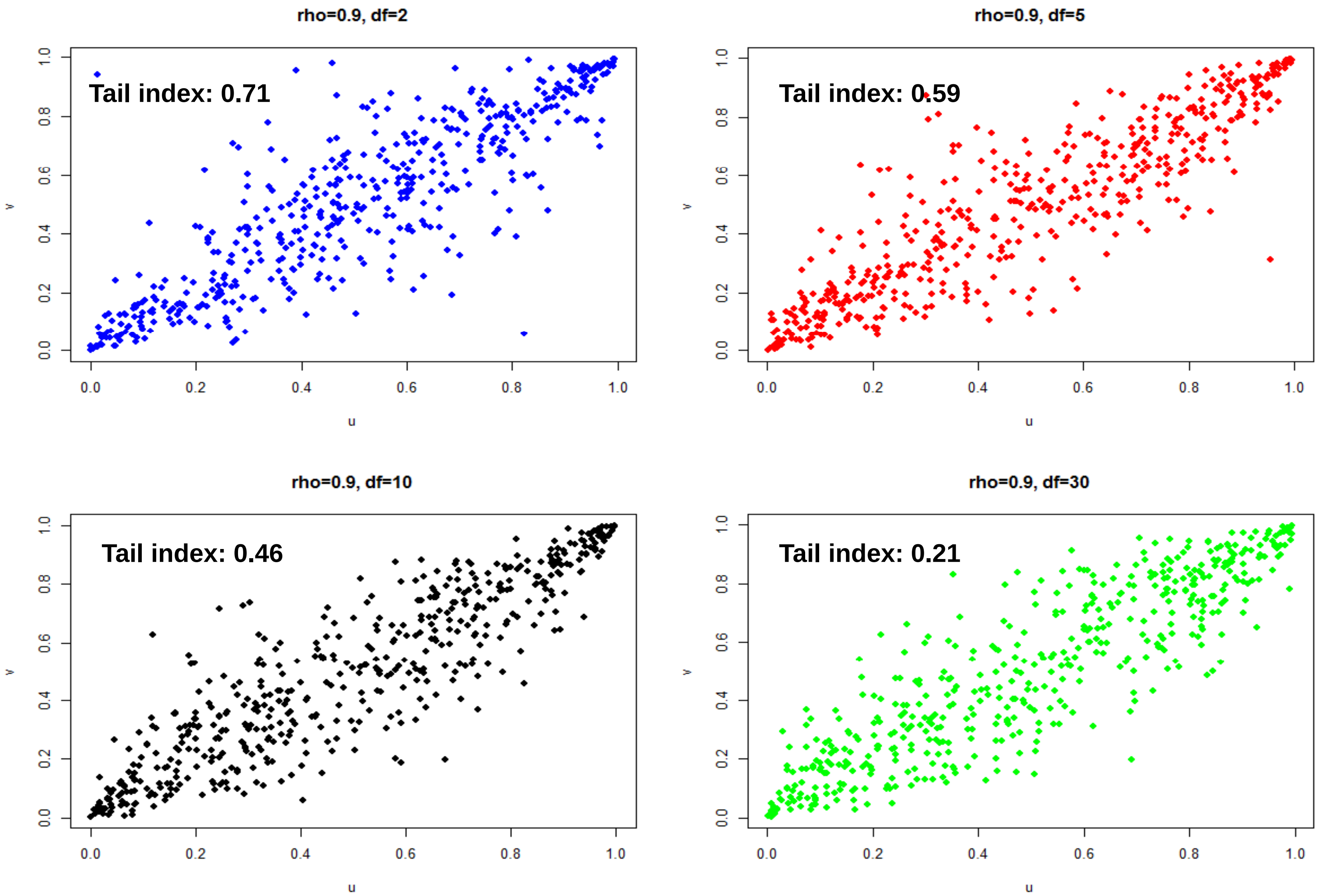



```
> tau(t.cop.9)
[1] 0.7129
> rho(t.cop.9)
[1] 0.8915
> tailIndex(t.cop.9)
 upper  lower
0.6298 0.6298
```





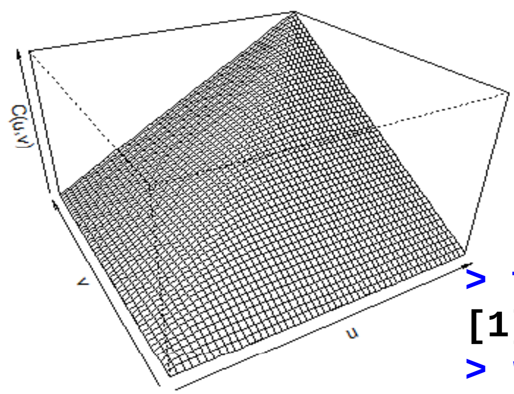
Simulations from t copulas



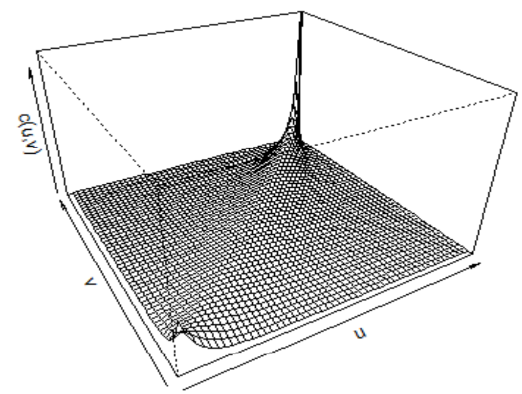
Gumbel Copula: δ = 4

```
> gum.cop.4 = archmCopula(family="gumbel", dim=2, param=4)
```

CDF

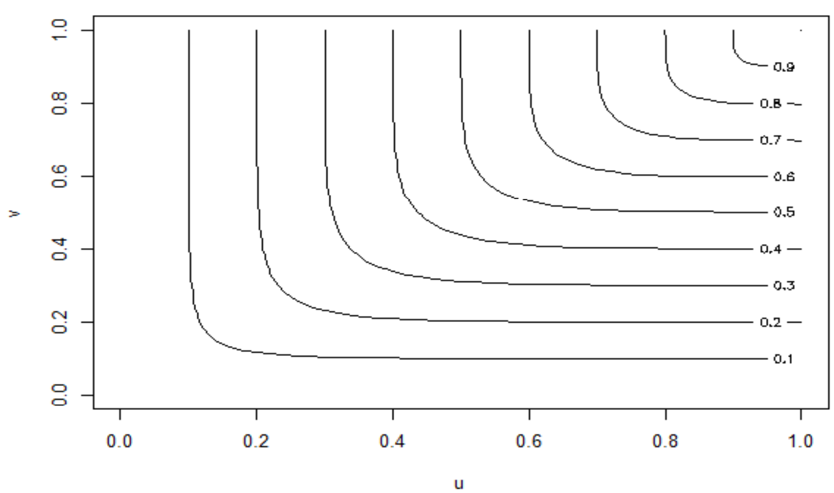


pdf

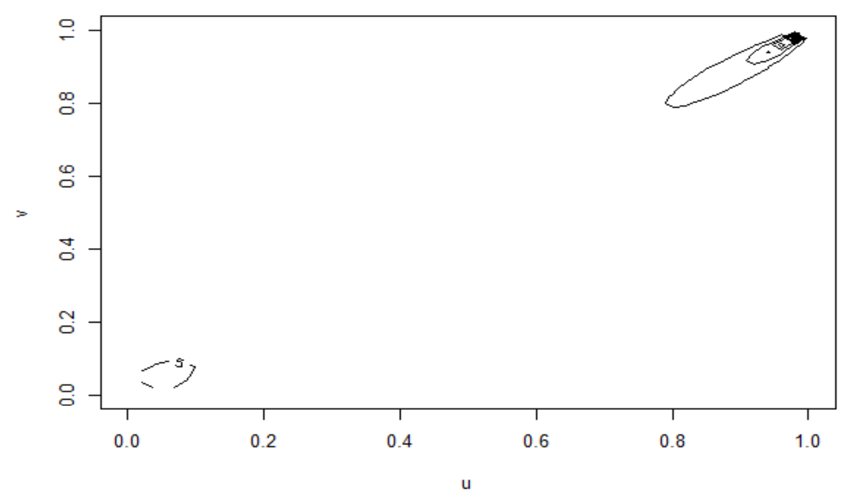



```
> tau(gum.cop.4)
[1] 0.75
> rho(gum.cop.4)
[1] 0.9127
> tailIndex(gum.cop.4)
 lower   upper
0.0000 0.8108
```

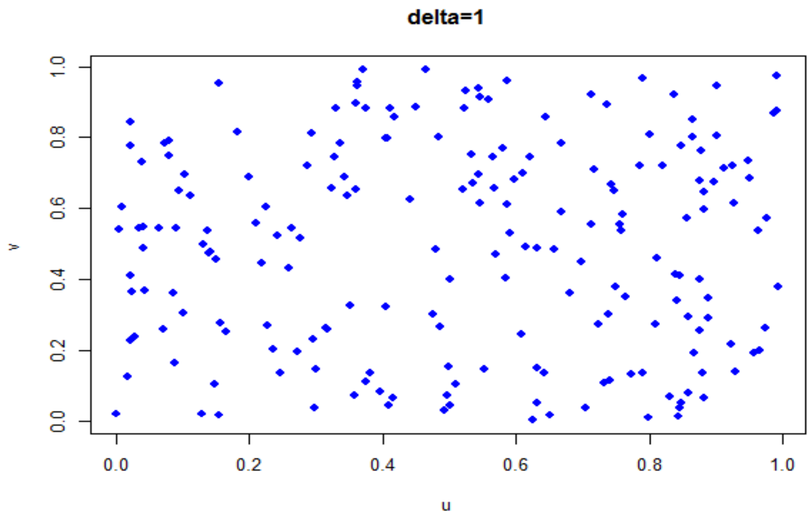
CDF

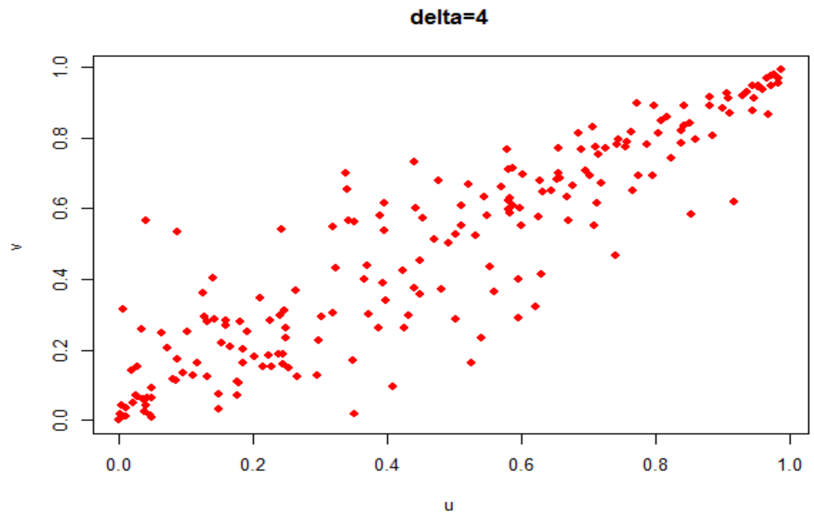


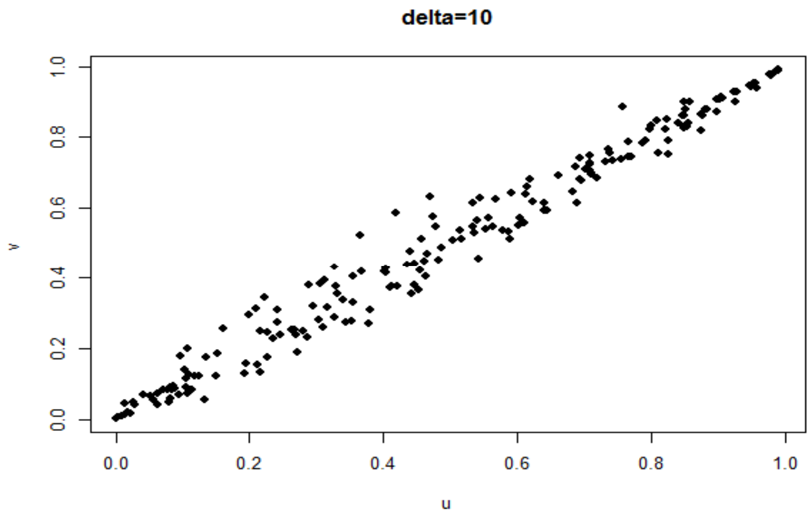
pdf



Simulations from Gumbel Copulas



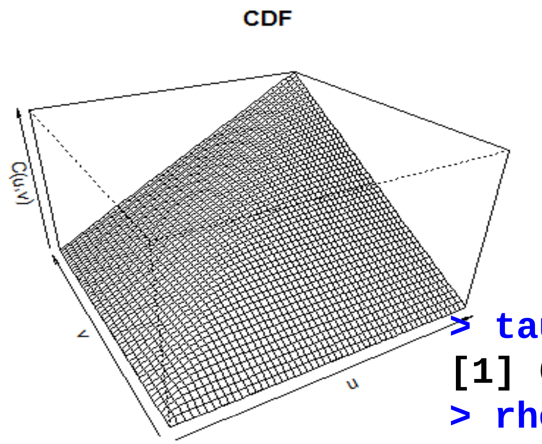


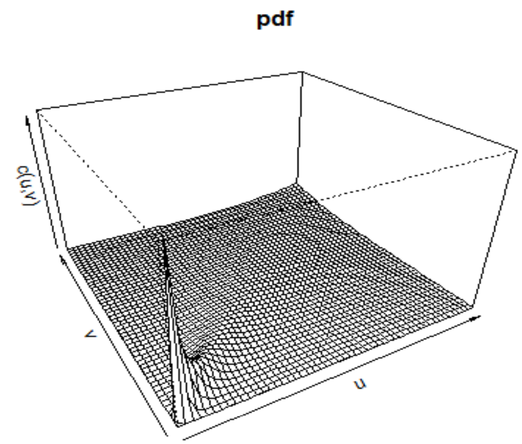



```
> tailIndex(gum.cop.1)
  lower   upper
0.00000 0.01368
> tailIndex(gum.cop.4)
 lower  upper
0.0000 0.8108
> tailIndex(gum.cop.10)
 lower  upper
0.0000 0.9282
```

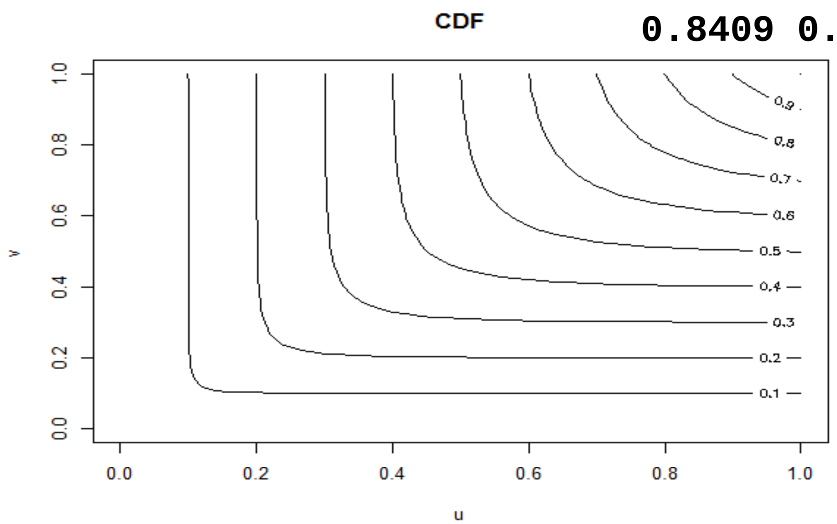
Clayton Copula: $\delta = 4$

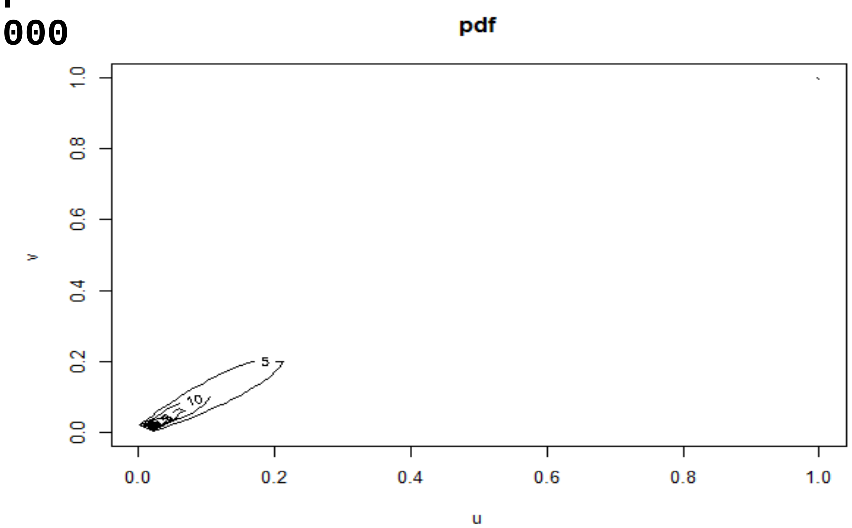
```
clay.cop.4 = archmCopula(family="clayton", dim=2, param=4)
```



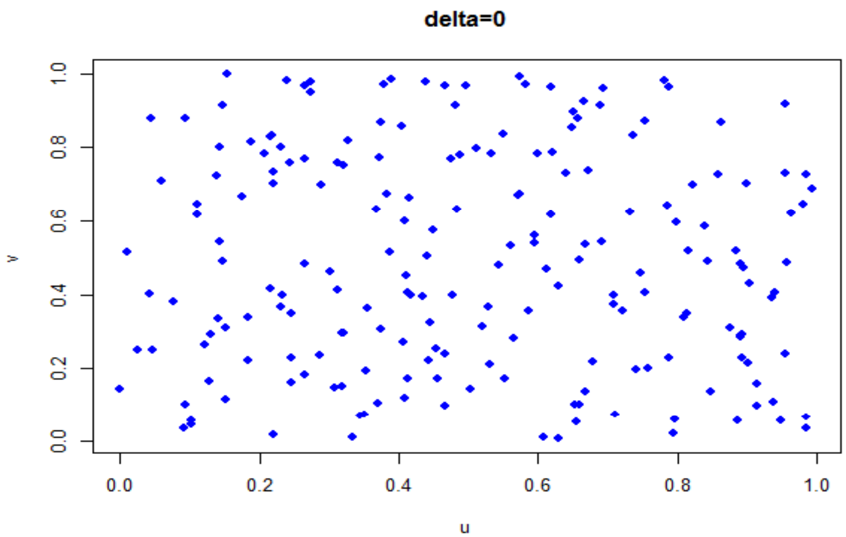


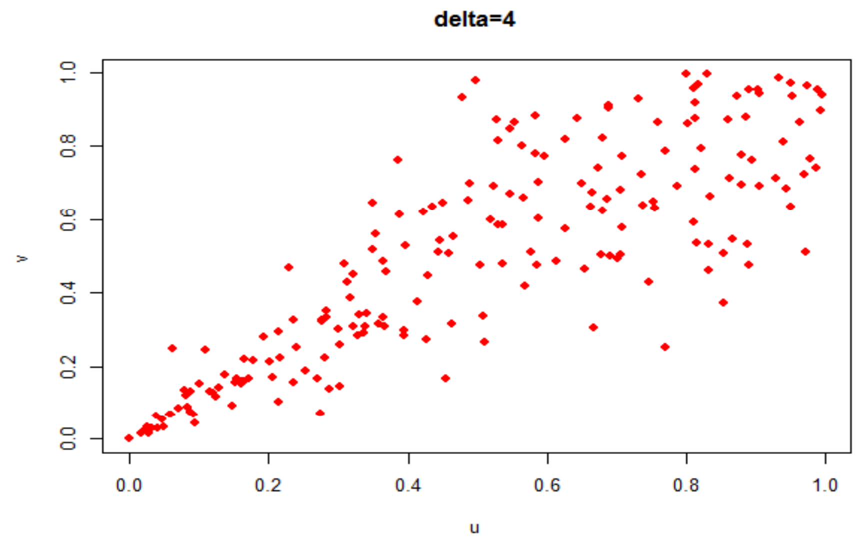

```
> tau(clay.cop.4)
[1] 0.6667
> rho(clay.cop.4)
[1] 0.8465
> tailIndex(clay.cop.4)
 lower  upper
0.8409 0.0000
```

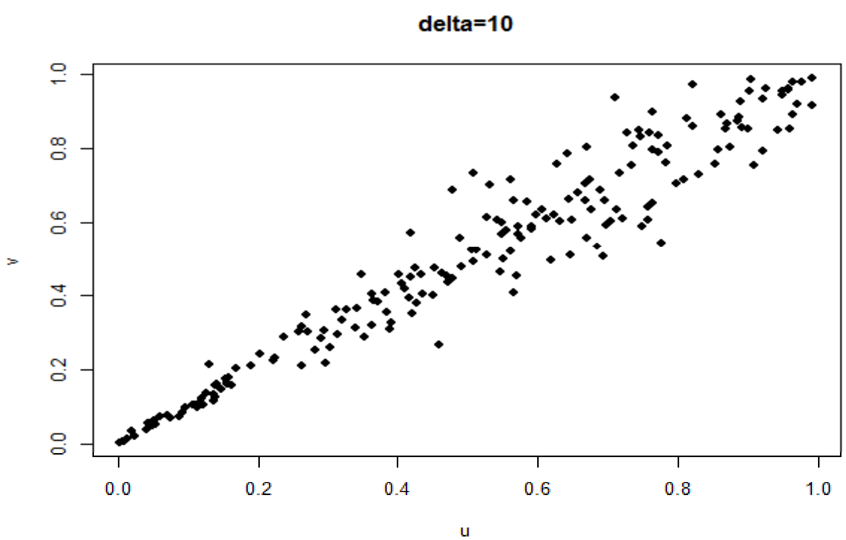




Simulations from Clayton Copulas








```
> tailIndex(clay.cop.0)
    lower     upper
7.889e-31 0.000e+00
> tailIndex(clay.cop.4)
 lower  upper
0.8409 0.0000
> tailIndex(clay.cop.10)
lower upper
0.933 0.000
```

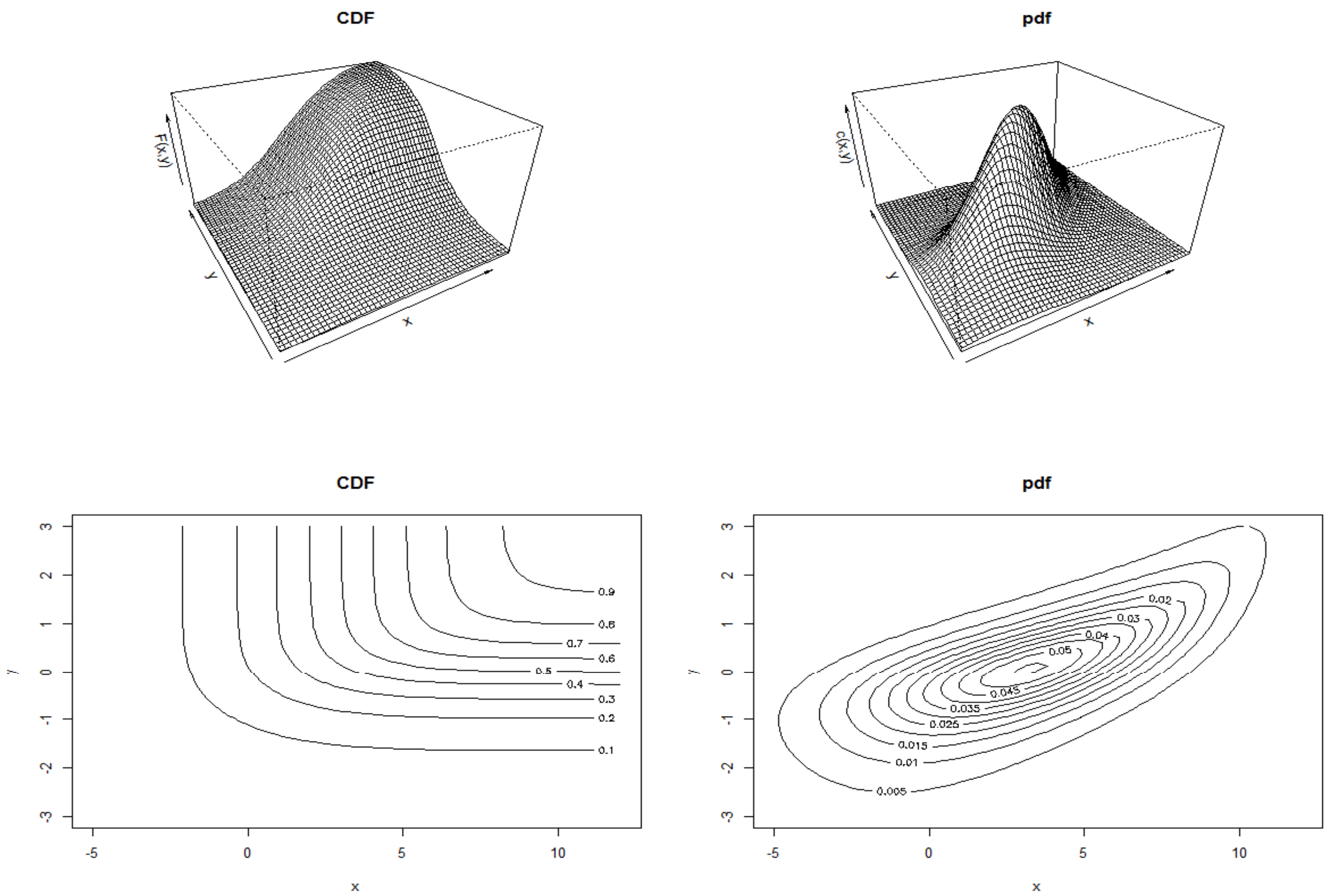
Create Custom Bivariate Distribution

```
> args(mvdc)
function (copula, margins, paramMargins, marginsIdentical = FALSE,
    check = TRUE, fixupNames = TRUE)

# bivariate distribution with N(3, 4^2) and t3 margins, and gumbel
# copula with d = 2
> my.cop = archmCopula(family="gumbel", dim=2, param=2)
> my.margins = c("norm", "t")
> my.parms = list(list(mean=3, sd=4), list(df=3))
> myBvd = mvdc(copula=my.cop,
+              margins=my.margins,
+              paramMargins=my.parms)

> class(myBvd)
[1] "mvdc"
attr(,"package")
[1] "copula"

> slotNames(myBvd)
[1] "copula"           "margins"           "paramMargins"
[4] "marginsIdentical"
```

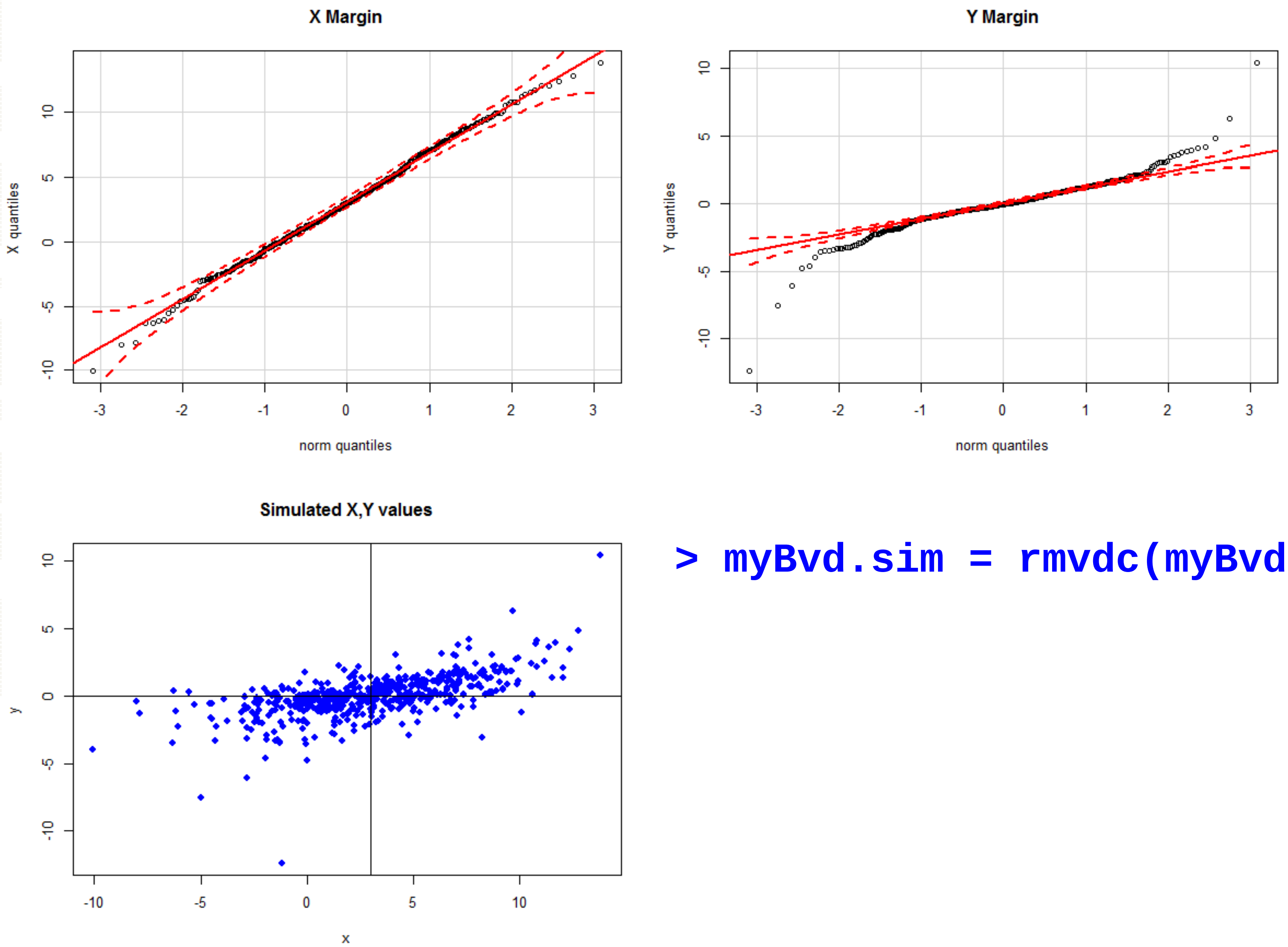
mvdc object

```
> myBvd
Multivariate Distribution Copula based ("mvdc")
 @ copula:
Gumbel copula family; Archimedean copula; Extreme value copula
Dimension:  2
Parameters:
   param  =  2
 @ margins:
[1] "norm" "t"
   with 2 (not identical)  margins; with parameters (@ paramMargins)
List of 2
 $ :List of 2
  ..$ mean: num 3
  ..$ sd  : num 4
 $ :List of 1
  ..$ df: num 3
```

Custom Bivariate Distribution



Custom Bivariate Distribution




```
> myBvd.sim = rmvdc(myBvd, 500)
```

Estimate Custom Bivariate Distn by MLE

```
# use copula function fitMvdc
> args(fitMvdc)
function (data, mvdc, start, optim.control = list(), method = "BFGS",
estimate.variance = TRUE, hideWarnings = TRUE)

> start.vals = c(1, 2, 4, 5)
> names(start.vals) = c("mu", "sigma", "df", "delta")
> myBvd.fitMvdc = fitMvdc(myBvd.sim, myBvd, start.vals)

> class(myBvd.fitMvdc)
[1] "fitMvdc"
attr(,"package")
[1] "copula“

> slotNames(myBvd.fitMvdc)
[1] "mvdc"          "estimate"       "var.est"   "loglik"   “nsample"
[6] "method"        "fitting.stats“

# there are show() and summary() methods and coefficients() extractor
```

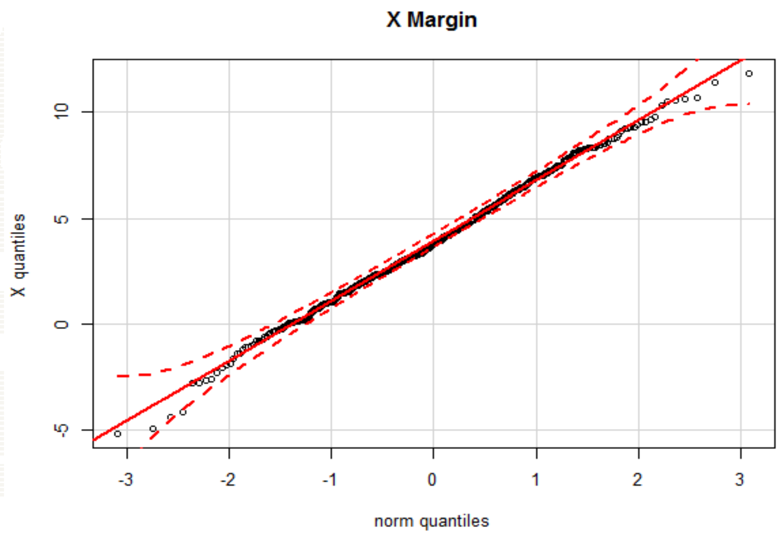
Estimate Custom Bivariate Distn by MLE

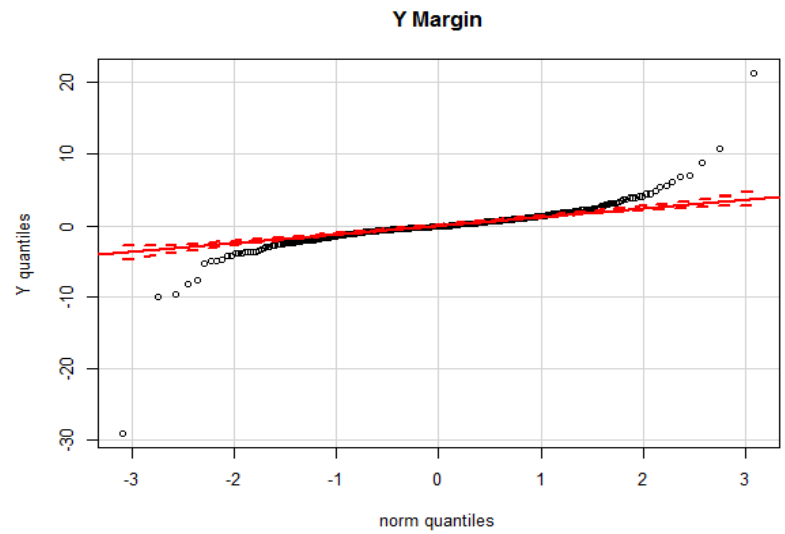
```
# estimation results
> myBvd.fitMvdc
The Maximum Likelihood estimation is based on  500  observations.
Margin  1 :
        Estimate Std. Error
m1.mean     3.09       0.13             # true value is 3
m1.sd       3.83       0.11             # true value is 4
Margin  2 :
      Estimate Std. Error
m2.df     2.98       0.31               # true value is 3
Copula:
      Estimate Std. Error
param     1.85       0.08               # true value is 2
The maximized loglikelihood is  -2111
Optimization converged
Number of loglikelihood evaluations:
function gradient
      67       28
```

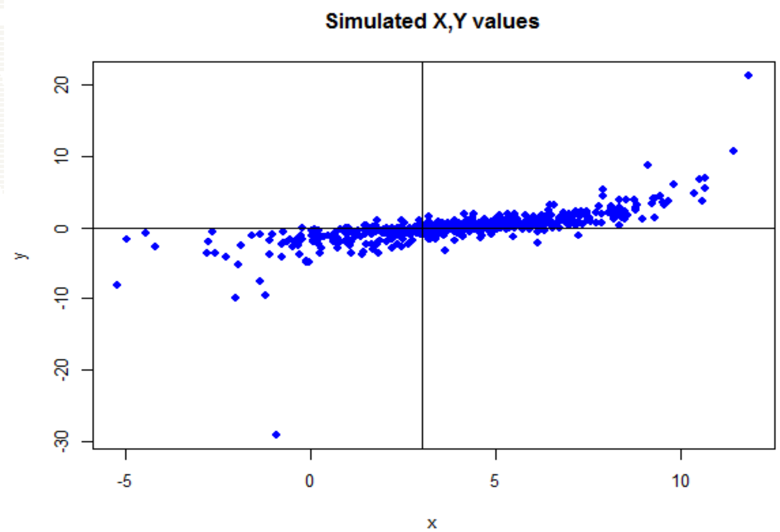
Simulate from Fitted Distribution

```
> param.hat = myBvd.fitMvdc@estimate
> param.hat
   mu sigma    df delta
3.093 3.832 2.977 1.853

> my.cop.fit = archmCopula(family="gumbel", dim=2,
+                          param=param.hat[1])
> my.margins.fit = c("norm", "t")
> my.parms.fit = list(list(mean=param.hat[2], sd=param.hat[3]),
+                     list(df=param.hat[4]))
> myBvd.fit = mvdc(copula=my.cop.fit,
+                  margins=my.margins.fit,
+                  paramMargins=my.parms.fit)

> myBvd.fit.sim = rMvdc(500, myBvd.fit)
```

Simulate from Fitted Distribution







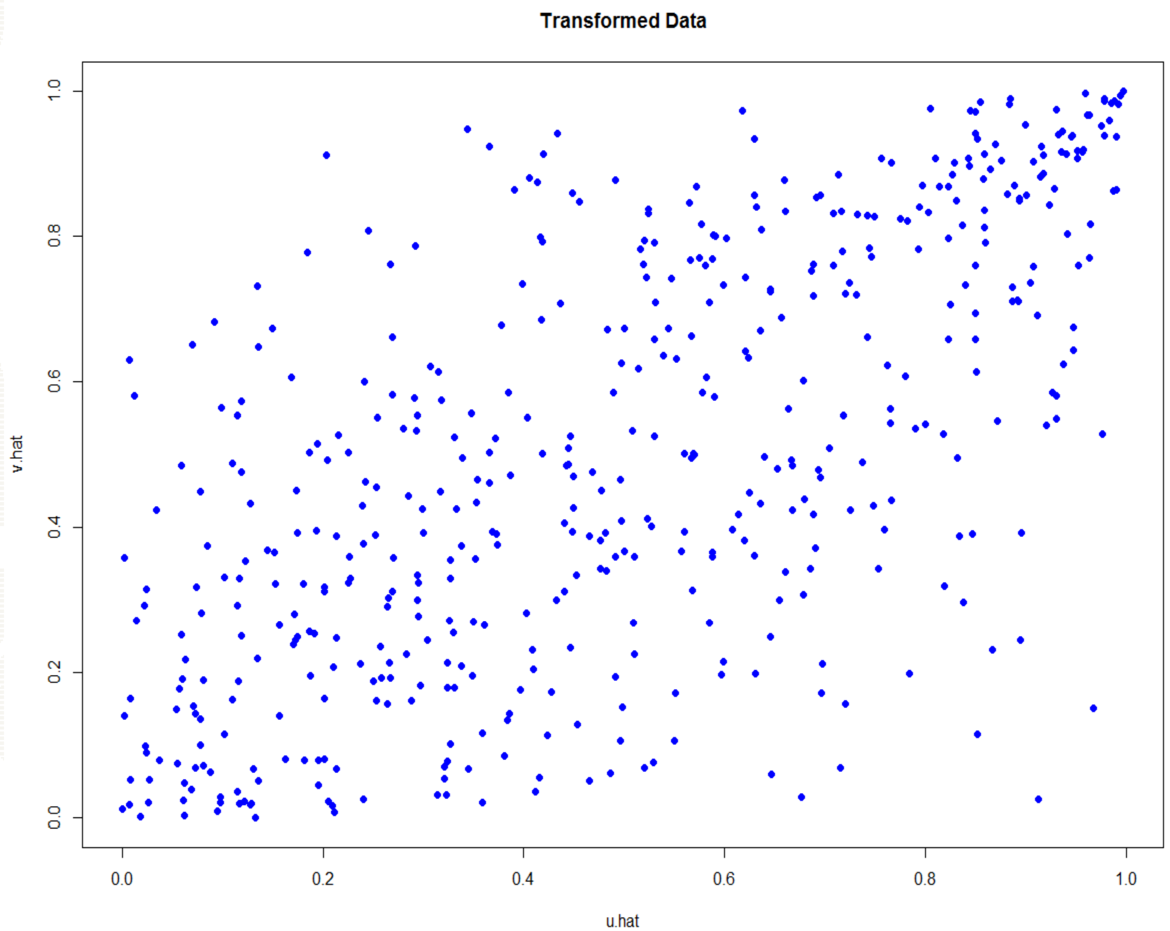

```
> set.seed(123)
> myBvd.sim = rMvdc(500, myBvd)
```

Estimate Custom Bivariate Distn by IFM

```
> x = myBvd.sim[,1]
> y = myBvd.sim[,2]

# step 1: estimate marginal distributions
# X ~ N(mu, sigma^2)
> mu.hat = mean(x)
> sigma.hat = sd(x)
> mu.hat
[1] 3.065
> sigma.hat
[1] 3.811

# Y ~ t(df)
> fit.t = fitdistr(y, densfun="t")
> df.hat = coef(fit.t)["df"]
> df.hat
   df
3.199
```

Estimate Custom Bivariate Distn by IFM

```
# transform data to uniform using estimated CDF function
> u.hat = pnorm(x, mu.hat, sigma.hat)
> v.hat = pt(y, df=df.hat)
```



Looks like there is some upper tail dependence here.

Estimate Custom Bivariate Distn by IFM

```
# step 2: estimate copula on transformed uniform
# observations using fitCopula()
> args(fitCopula)
function (copula, data,
          method = c("mpl", "ml", "itau", "irho"),
          start = NULL, lower = NULL, upper = NULL,
          optim.method = "BFGS",
          optim.control = list(maxit = 1000),
          estimate.variance = TRUE, hideWarnings = FALSE)

> fit.ifm = fitCopula(copula=myBvd@copula,
+                     data=cbind(u.hat,v.hat), start=2)
> class(fit.ifm)
[1] "fitCopula"
attr(,"package")
[1] "copula"

> slotNames(fit.ifm)
[[1] "copula"        "estimate"      "var.est"       "loglik"
[5] "nsample"       "method"        "fitting.stats"
```

Estimate Custom Bivariate Distn by IFM

```
> fit.ifm
fitCopula() estimation based on 'maximum pseudo-likelihood'
and a sample of size 500.
      Estimate Std. Error z value Pr(>|z|)
param    1.835      0.079    23.2   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
The maximized loglikelihood is  153
Optimization converged
Number of loglikelihood evaluations:
function gradient
      18        6

# compare MLE and IFM fits
> myBvd.fitMvdc@estimate
   mu sigma    df delta
3.093 3.832 2.977 1.853
> ifm.parms
   mu sigma    df delta
3.065 3.811 3.199 1.835
```

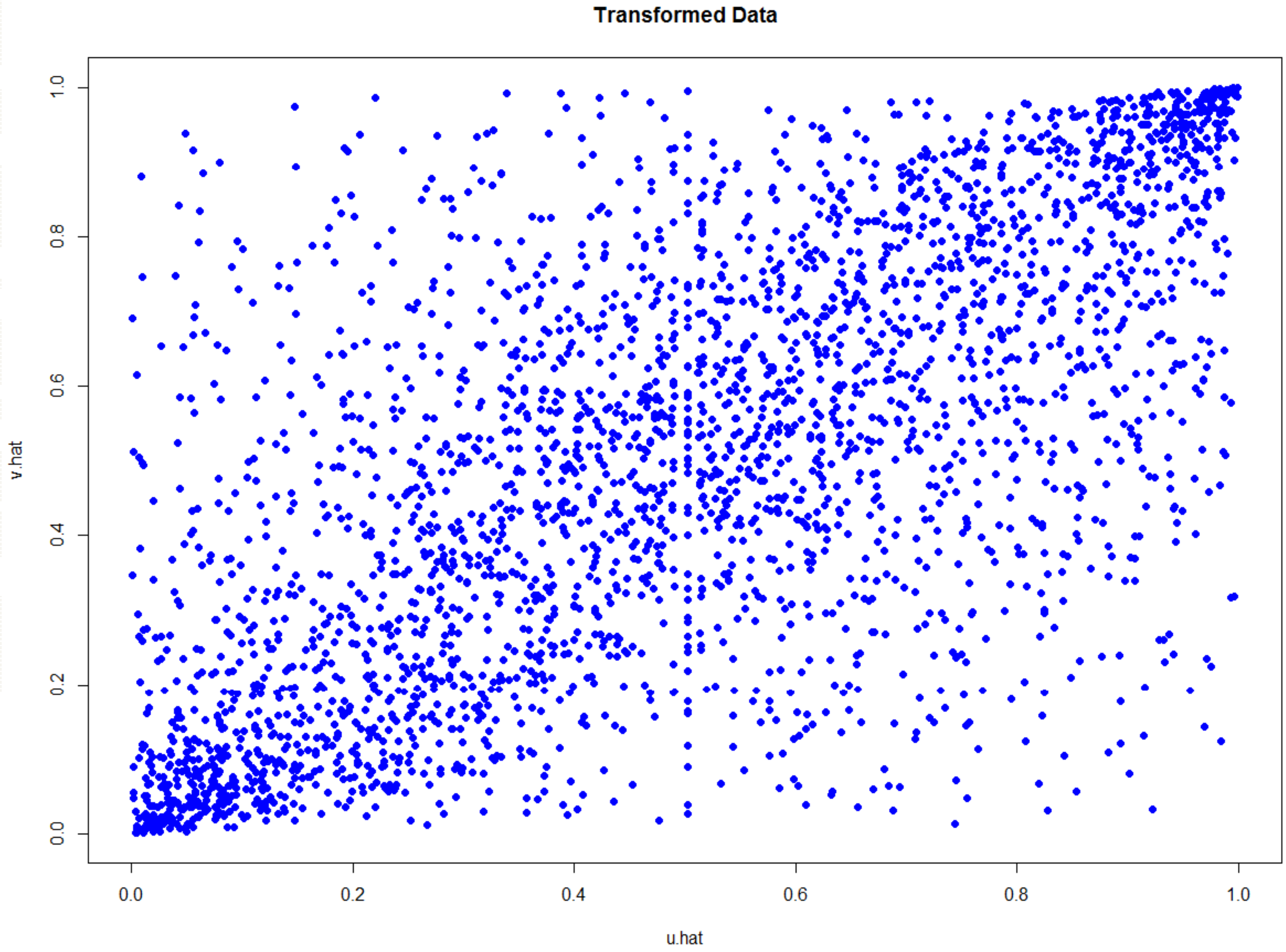
Evaluating Goodness of Fit

```
# compute Cramer-von
> gof.test = gofCopula(myBvd@copula, cbind(u.hat,v.hat),
+                      estim.method="mpl")
===============================================================

> gof.test

        Parametric bootstrap based GOF test with 'method'="Sn",
'estim.method'="mpl"

data:  x
statistic = 0.0108, parameter = 1.86, p-value = 0.9016

# Here we do not reject the hypothesis that our copula is the
# true copula.
```

Estimate Custom Distribution for MSFT & GSPC

```
# fit univariate skew-t distributions
> st.msft.fit = st.mle(y=coredata(MSFT.ret))
> st.gspc.fit = st.mle(y=coredata(GSPC.ret))

# transform data to uniform
> u.hat = pst(coredata(MSFT.ret), dp=st.msft.fit$dp)
> v.hat = pst(coredata(GSPC.ret), dp=st.gspc.fit$dp)

# create normal and t copula objects for fitting
# Note: correlation and df parameters will be estimated
# by the fitCopula function
> n.cop = normalCopula(param=0.5, dim=2)
> t.cop = tCopula(param=0.5, dim=2, df=3)

# plot estimated uniform data
> plot(u.hat,v.hat, main="Transformed Data",
+      xlab="u.hat", ylab="v.hat", pch=16, col="blue")
```

Estimated Uniform Values



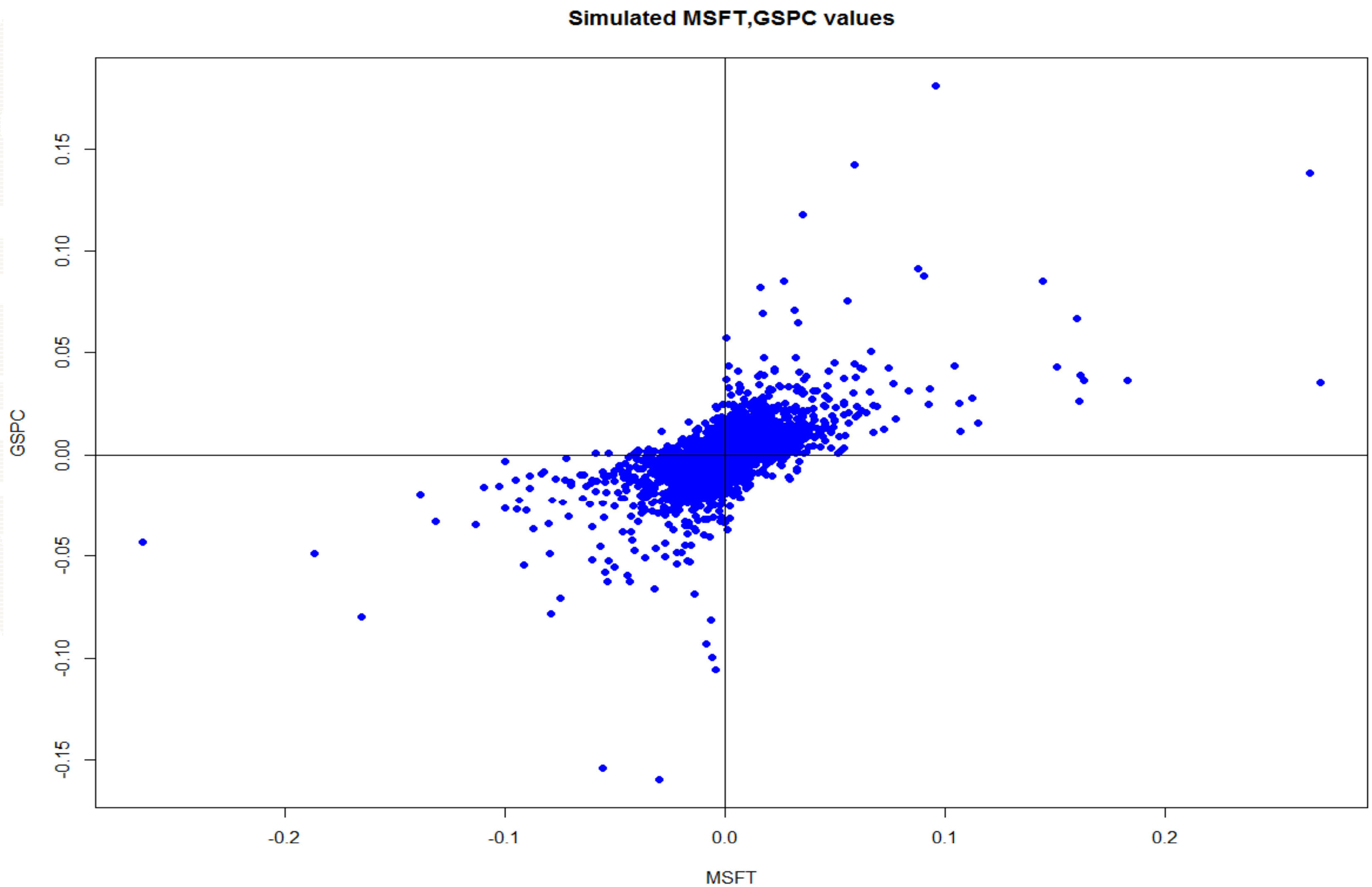
Estimate Custom Distribution for MSFT & GSPC

```
# fit normal copula model by IFM
> start.vals = 0.5
> fit.ifm = fitCopula(copula=n.cop,
                      data=cbind(u.hat,v.hat),
                      start=start.vals)

fitCopula() estimation based on 'maximum pseudo-
likelihood' and a sample of size 3082.
      Estimate Std. Error z value Pr(>|z|)
rho.1  0.68944    0.00744    92.6   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
' ' 1
The maximized loglikelihood is  993
Optimization converged
Number of loglikelihood evaluations:
function gradient
      36        8
```

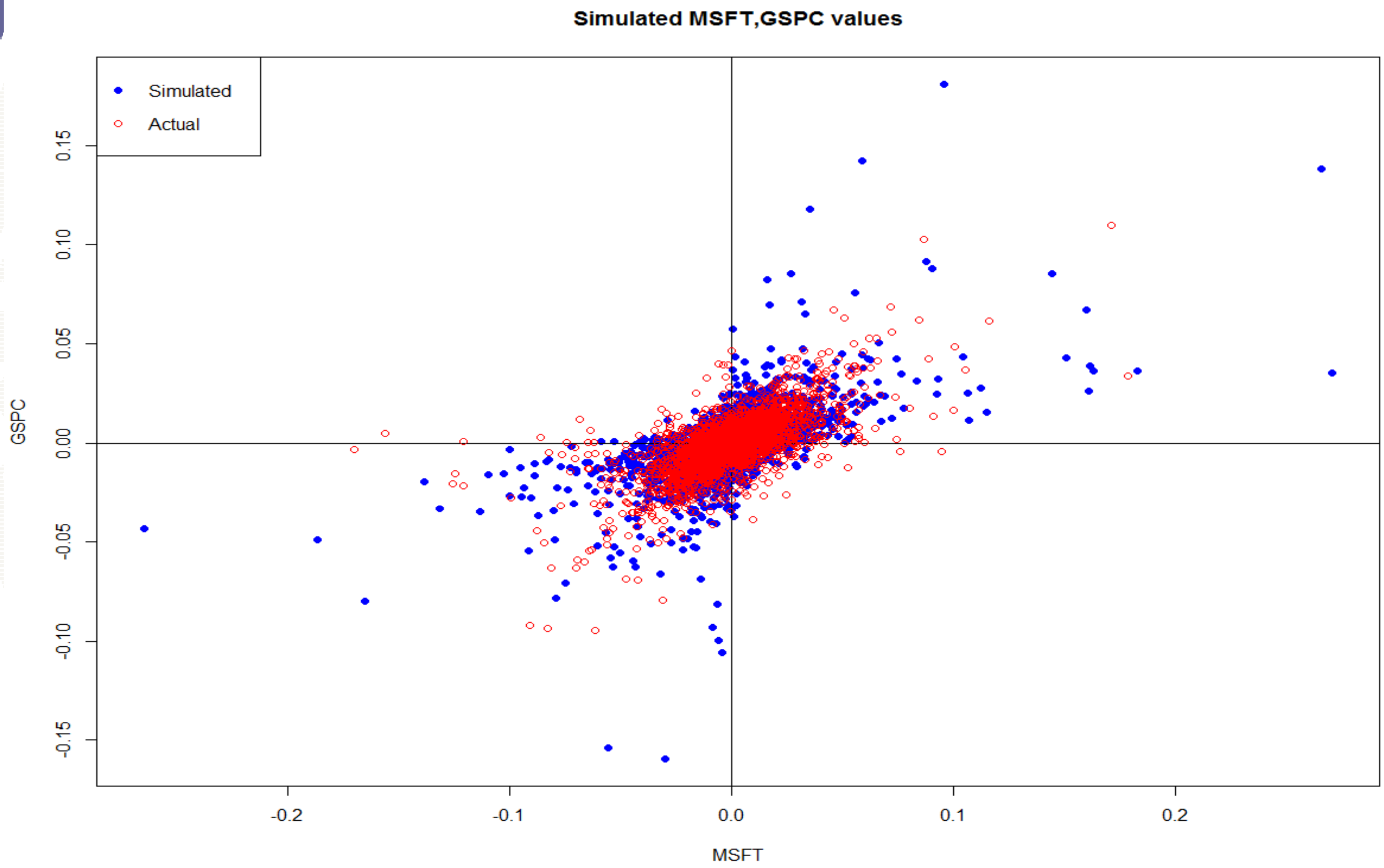
Estimate Custom Distribution for MSFT & GSPC

```
# fitted skew-t margins
> MSFT.GSPC.margins = c("st", "st")
> MSFT.GSPC.parms = list(list(dp=st.msft.fit$dp),
+                         list(dp=st.gspc.fit$dp))

# fitted normal copula
> MSFT.GSPC.n.cop = normalCopula(param=fit.ncop.ifm@estimate,
dim=2)
> myBvd.MSFT.GPSC.fit = mvdc(copula=MSFT.GSPC.n.cop,
+                             margins=MSFT.GSPC.margins,
+                             paramMargins=MSFT.GSPC.parms)

# simulate from fitted bivariate distn
> myBvd.MSFT.GSPC.fit.sim = rMvdc(nrow(MSFT.ret),
+                                 myBvd.MSFT.GPSC.fit)
```

Simulations from Custom Distribution



Simulated vs Actual: Normal Copula



Estimate Custom Distribution for MSFT & GSPC

```
> start.vals = c(0.5, 3)
> names(start.vals) = c("rho.1","df")
> fit.tcop.ifm = fitCopula(copula=t.cop,
+                           data=cbind(u.hat,v.hat),
+                           method="mpl",
+                           start=start.vals, optim.method="L-BFGS-B",
+                           lower=c(-0.99, 2),
+                           upper=c(0.99, 10))
> fit.tcop.ifm
fitCopula() estimation based on 'maximum pseudo-likelihood'
and a sample of size 3082.
      Estimate Std. Error z value Pr(>|z|)
rho.1   0.7009     0.0129    54.1   <2e-16 ***
df      3.5657         NA      NA       NA
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
The maximized loglikelihood is  1113
Optimization converged
Number of loglikelihood evaluations:
function gradient
      14       14
```

Simulate from Fitted Custom Distribution

```
# create fitted t-copula object
> t.cop.fit = tCopula(param=fit.tcop.ifm@estimate[1], dim=2,
+                     df=fit.tcop.ifm@estimate[2])

# created fitted custom distribution object
> myBvd.tcop.fit = mvdc(copula=t.cop.fit,
+                       margins=MSFT.GSPC.margins,
+                       paramMargins=MSFT.GSPC.parms)

# simulate from fitted bivariate distn
> myBvd.fit.sim = rMvdc(nrow(MSFT.ret), myBvd.tcop.fit)
```

Simulated vs Actual: t Copula