DSA 8070 R Session 3: Multivariate Normal Distribution, Copula, and Nonparametric Density Estimation

Whitney

Contents

Multivariate Normal Distribution

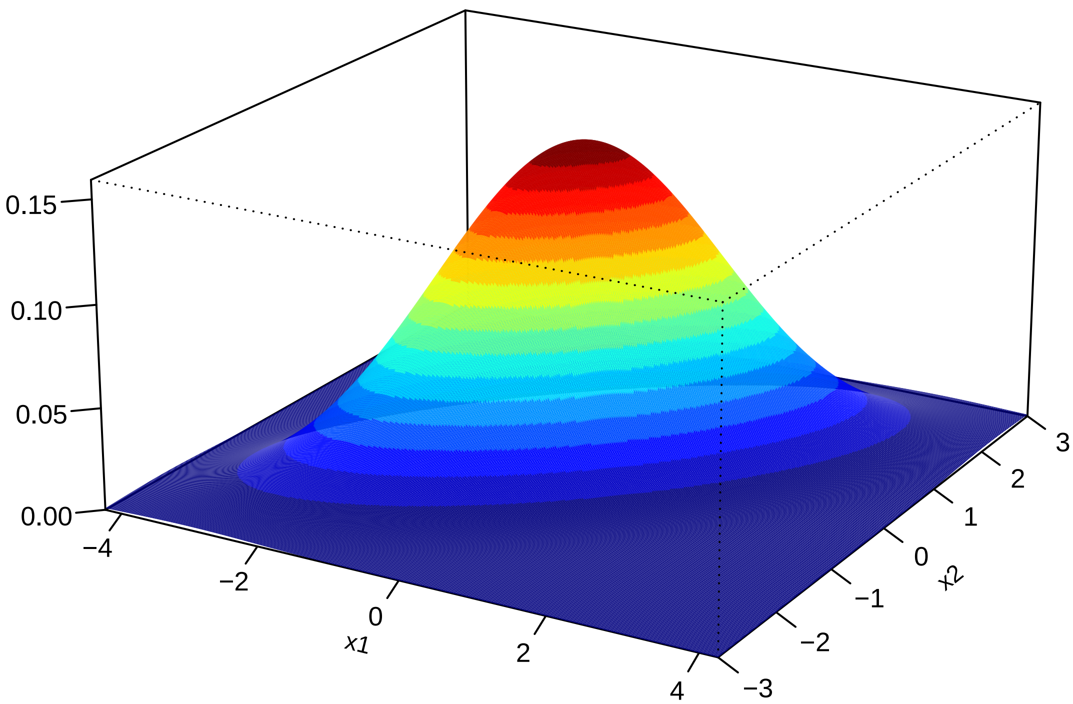
Bivariate normal density

If X_1 and X_2 jointly follows a bivariate normal distribution, then the probability density function takes the following form:

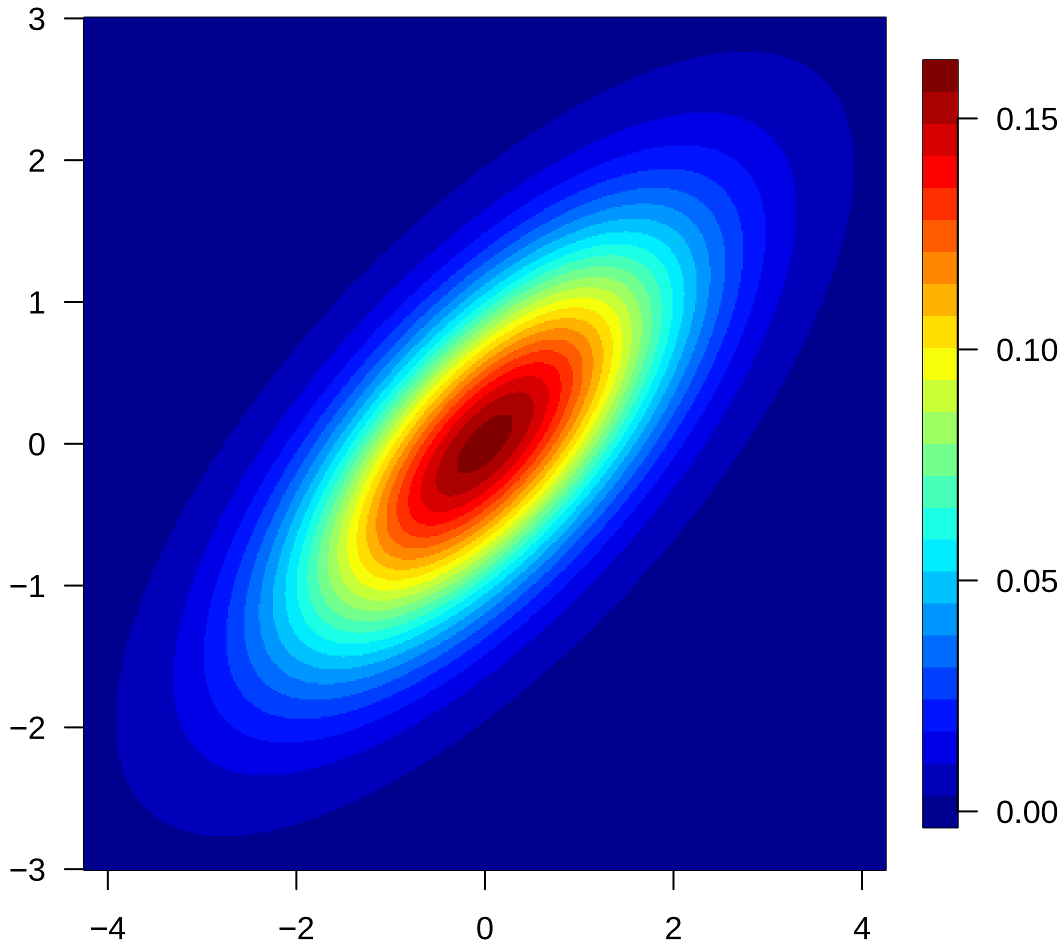
$$f(x_1, x_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2(1-\rho^2)}\left[\left(\frac{x_1-\mu_1}{\sigma_1}\right)^2 + \left(\frac{x_2-\mu_2}{\sigma_2}\right)^2 - 2\rho\frac{(x_1-\mu_1)(x_2-\mu_2)}{\sigma_1\sigma_2}\right]\right),$$

where $[\mu_1, \mu_2]^T$ is the mean vector, and $\begin{bmatrix}\sigma_{11} = \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_{22} = \sigma_2^2\end{bmatrix}$ is the covariance matrix.

```r
## parameters
mu1 <- mu2 <- 0
sigma11 <- 2 #variance of X1
sigma22 <- 1 #variance of X2
sigma12 <- 1 #covariance of X1 and X2
Sigma <- matrix(c(sigma11, sigma12, sigma12, sigma22), 2)
# create grids for plotting
x1 <- seq(mu1 - 3 * sqrt(sigma11), mu1 + 3 * sqrt(sigma11), length = 500)
x2 <- seq(mu2 - 3 * sqrt(sigma22), mu2 + 3 * sqrt(sigma22), length = 500)
library(mvtnorm)
grids <- expand.grid(x1, x2)
out <- array(dmvnorm(grids, c(mu1, mu2), Sigma), dim = c(500, 500))

par(mar = c(2, 2, 0.8, 1.3), mgp = c(2.2, 1, 0))
library(GA)
persp3D(x1, x2, out, theta = 30, phi = 20, expand = 0.5,
        zlab = "", cex.axis = 0.8, cex.lab = 0.75)
library(fields)
```

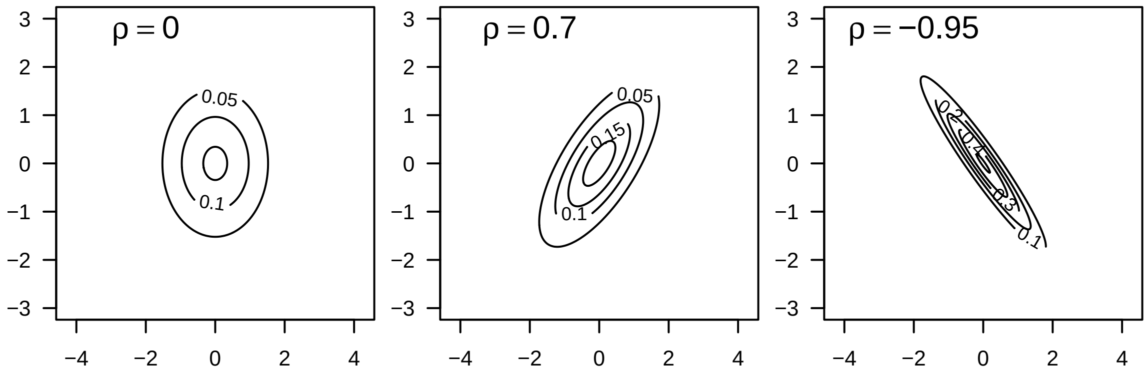



```r
image.plot(x1, x2, out, las = 1, col = tim.colors(24),
           legend.cex = 0.5, legend.mar = 7)
```



Bivariate normal with different ρ

```
par(mfrow = c(1, 3), mar = c(2, 2, 0.8, 0.6))
mu <- c(0, 0)
rho <- c(0, 0.7, -0.95)
for (i in 1:3){
  Sigma <- matrix(c(1, rho[i], rho[i], 1), 2)
  f <- array(dmvnorm(grids, mu, Sigma), dim = c(500, 500))
  contour(x1, x2, f, las = 1, nlevels = 5)
  text(-2, 2.75, bquote(rho == .(rho[i])), cex = 1.5)
}
```



Multivariate normal QQplot

1. Calculate the Hotelling's squared statistics

$$d^2 = (\boldsymbol{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu})$$

   for each multivariate data point $\boldsymbol{x}_i, \quad i = 1, \cdots, n.$

2. Sort $\{d_i\}_{i=1}^n$ in increasing order
3. Compute the theoretical quantiles under multivariate normal condition, which are the chi-squared quantiles with degrees of freedom equal to its dimension p (i.e., the number of variables)
4. Create a scatterplot with both empirical and theoretical quantiles.
5. Add a reference line to assess to straightness.

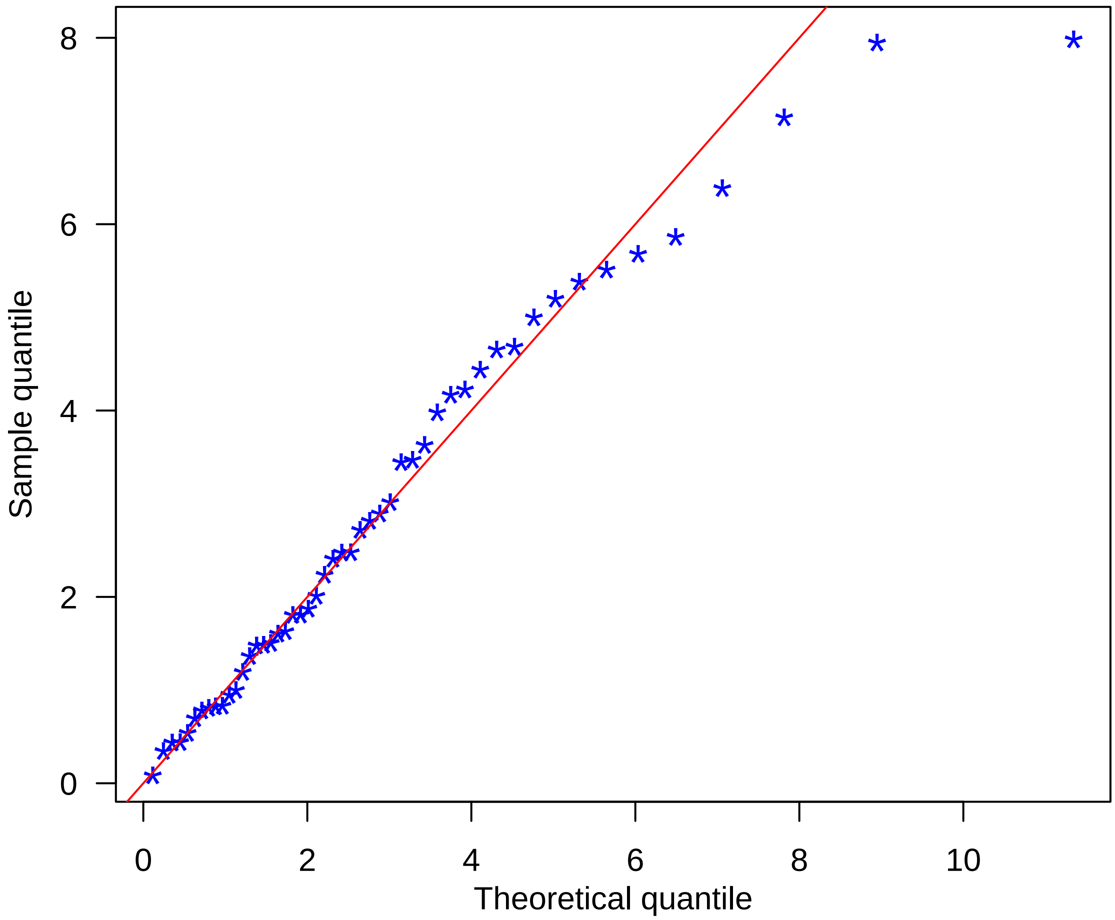
```r
versicolor <- iris[51:100, 1:3]
(mu <- apply(versicolor, 2, mean))
```

```
## Sepal.Length  Sepal.Width Petal.Length
##        5.936        2.770        4.260
```

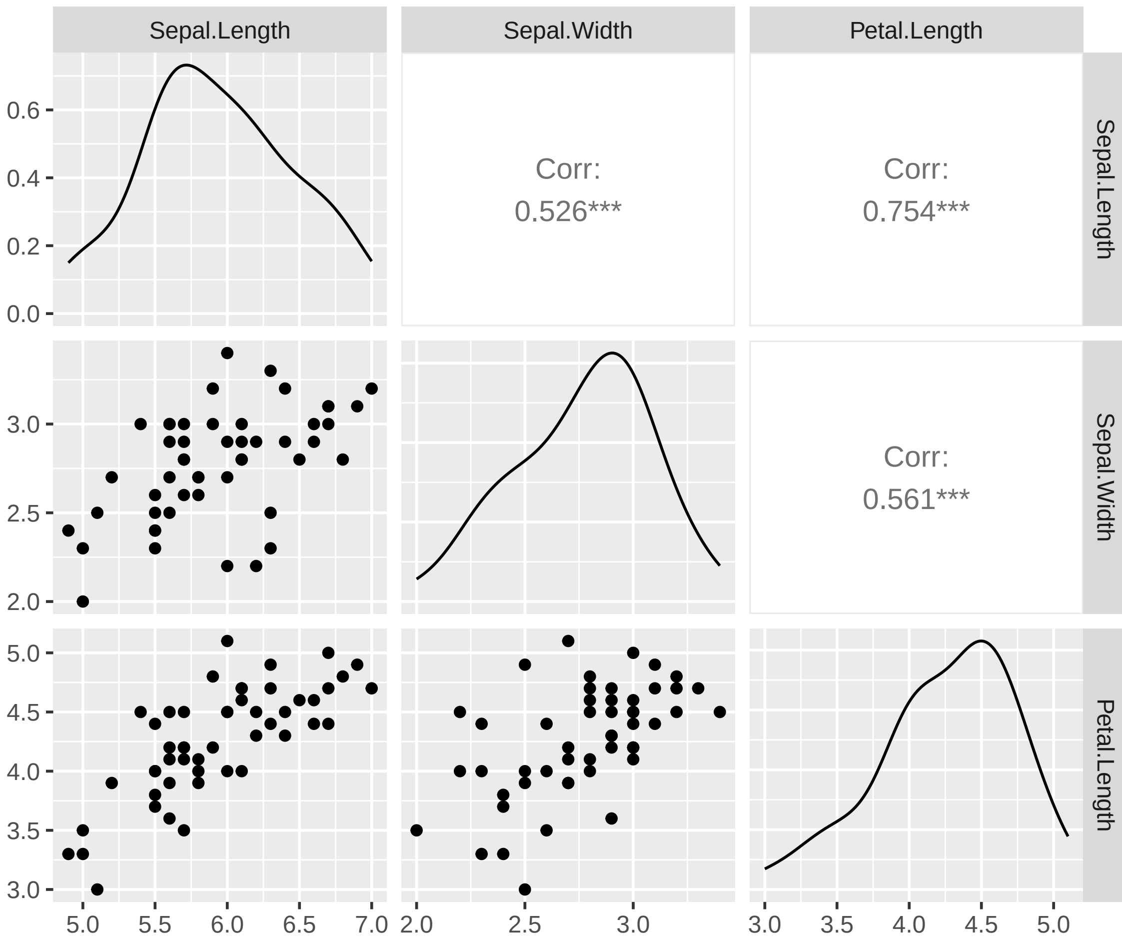
```r
(Sigma <- cov(versicolor))
```

```
##              Sepal.Length Sepal.Width Petal.Length
## Sepal.Length   0.26643265  0.08518367   0.18289796
## Sepal.Width    0.08518367  0.09846939   0.08265306
## Petal.Length   0.18289796  0.08265306   0.22081633
```

```r
# empirical quantiles
d.square <- apply(versicolor, 1, function(x) t(x - mu) %*% solve(Sigma) %*% (x - mu))
# sample size
n <- dim(versicolor)[1]
# theoretical quantiles under MVN
chiquant <- qchisq((1:n - 0.5) / n, 3)
# QQ plot
par(las = 1, mgp = c(2, 1, 0), mar = c(3.5, 3.5, 0.8, 0.6))
plot(chiquant, sort(d.square), pch = "*", col = "blue",
     xlab = "Theoretical quantile", ylab = "Sample quantile", cex = 1.6)
abline(0, 1, col = "red")
```



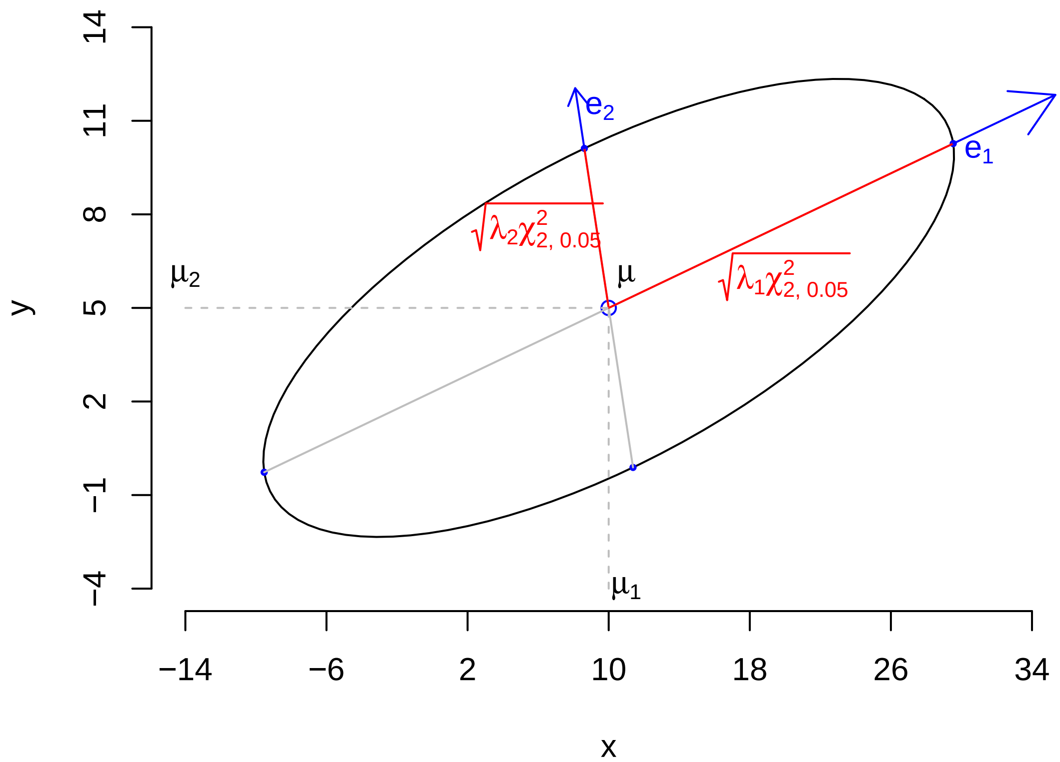

```r
library(GGally)
ggpairs(versicolor)
```



Probability contour

```
# mean, covariance, and data range
mu <- c(10, 5)
Sigma <- matrix(c(64, 16, 16, 9), 2)
sd1 <- sqrt(diag(Sigma)[1]); sd2 <- sqrt(diag(Sigma)[2])
xr <- c(mu[1] - 3 * sd1, mu[1] + 3 * sd1)
yr <- c(mu[2] - 3 * sd2, mu[2] + 3 * sd2)
# Spectral decomposition
spetralDecomp <- eigen(Sigma)
lambda1 <- spetralDecomp$values[1]
lambda2 <- spetralDecomp$values[2]
e1 <- spetralDecomp$vectors[, 1]
e2 <- spetralDecomp$vectors[, 2]
# plot the ellipse
c <- sqrt(qchisq(0.95, 2))
library(ellipse)
cor <- Sigma[1, 2] / (sd1 * sd2)
plot(ellipse(cor, scale = sqrt(diag(Sigma)), centre = mu), type = 'l',
xlim = xr, ylim = yr, las = 1, bty = "n", yaxt = "n", xaxt = "n", las = 1)
points(mu[1], mu[2], col = "blue")
axis(1, at = seq(mu[1] - 3 * sd1, mu[1] + 3 * sd1, sd1))
```

```r
axis(2, at = seq(mu[2] - 3 * sd2, mu[2] + 3 * sd2, sd2))
# add axes and labels
majorX <- c * sqrt(lambda1) * -e1[1]
majorY <- c * sqrt(lambda1) * -e1[2]
minorX <- c * sqrt(lambda2) * -e2[1]
minorY <- c * sqrt(lambda2) * -e2[2]
points(mu[1] + majorX, mu[2] + majorY, pch = 16, cex = 0.5, col = "blue")
points(mu[1] - majorX, mu[2] - majorY, pch = 16, cex = 0.5, col = "blue")
points(mu[1] + minorX, mu[2] + minorY, pch = 16, cex = 0.5, col = "blue")
points(mu[1] - minorX, mu[2] - minorY, pch = 16, cex = 0.5, col = "blue")
segments(mu[1] + majorX, mu[2] + majorY,
         mu[1] - majorX, mu[2] - majorY, col = "gray")
segments(mu[1], mu[2], mu[1] + majorX, mu[2] + majorY, col = "red")
text(20, 6, expression(sqrt(lambda[1]*chi["2, 0.05"]^2)), col = "red")
segments(mu[1] + minorX, mu[2] + minorY,
         mu[1] - minorX, mu[2] - minorY, col = "gray")
segments(mu[1], mu[2], mu[1] + minorX, mu[2] + minorY, col = "red")
text(6, 7.6, expression(sqrt(lambda[2]*chi["2, 0.05"]^2)), col = "red")
arrows(mu[1] + majorX, mu[2] + majorY,
       mu[1] + majorX + (-6) * e1[1], mu[2] + majorY + (-6) * e1[2],
       col = "blue")
arrows(mu[1] + minorX, mu[2] + minorY, mu[1] + minorX + (-2) * e2[1],
       mu[2] + minorY + (-2) * e2[2], length = 0.1, col = "blue")
segments(10, -4, 10, 5, col = "gray", lty = 2)
text(11, -4, expression(mu["1"]))
segments(-14, 5, 10, 5, col = "gray", lty = 2)
text(-14, 6, expression(mu["2"]))
text(11, 6, expression(bolditalic(mu)))
text(31, 10, expression(e["1"]), col = "blue")
text(9.5, 11.4, expression(e["2"]), col = "blue")
```



Wechsler Adult Intelligence Scale Example

```r
data <- read.table("wechsler.txt")
(mu <- apply(data[, -1], 2, mean))
```

```
##        V2        V3        V4        V5
## 12.567568  9.567568 11.486486  7.972973
```

```r
(Sigma <- cov(data[, -1]))
```

```
##           V2         V3        V4        V5
## V2 11.474474  9.0855856  6.382883 2.0713213
## V3  9.085586 12.0855856  5.938438 0.5435435
## V4  6.382883  5.9384384 11.090090 1.7912913
## V5  2.071321  0.5435435  1.791291 3.6936937
```

```r
out <- eigen(Sigma)
out$values
```

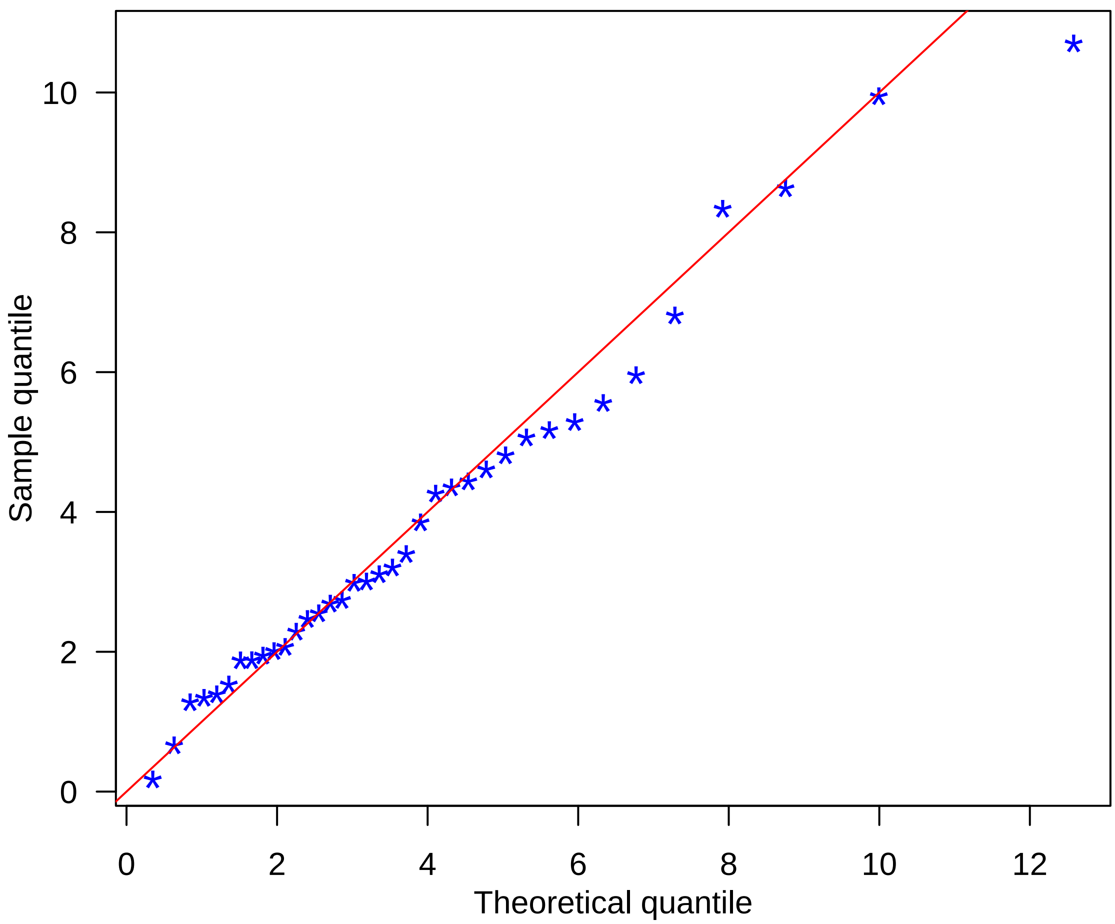
```
## [1] 26.245278  6.255366  3.931553  1.911647
```

```r
out$vectors
```

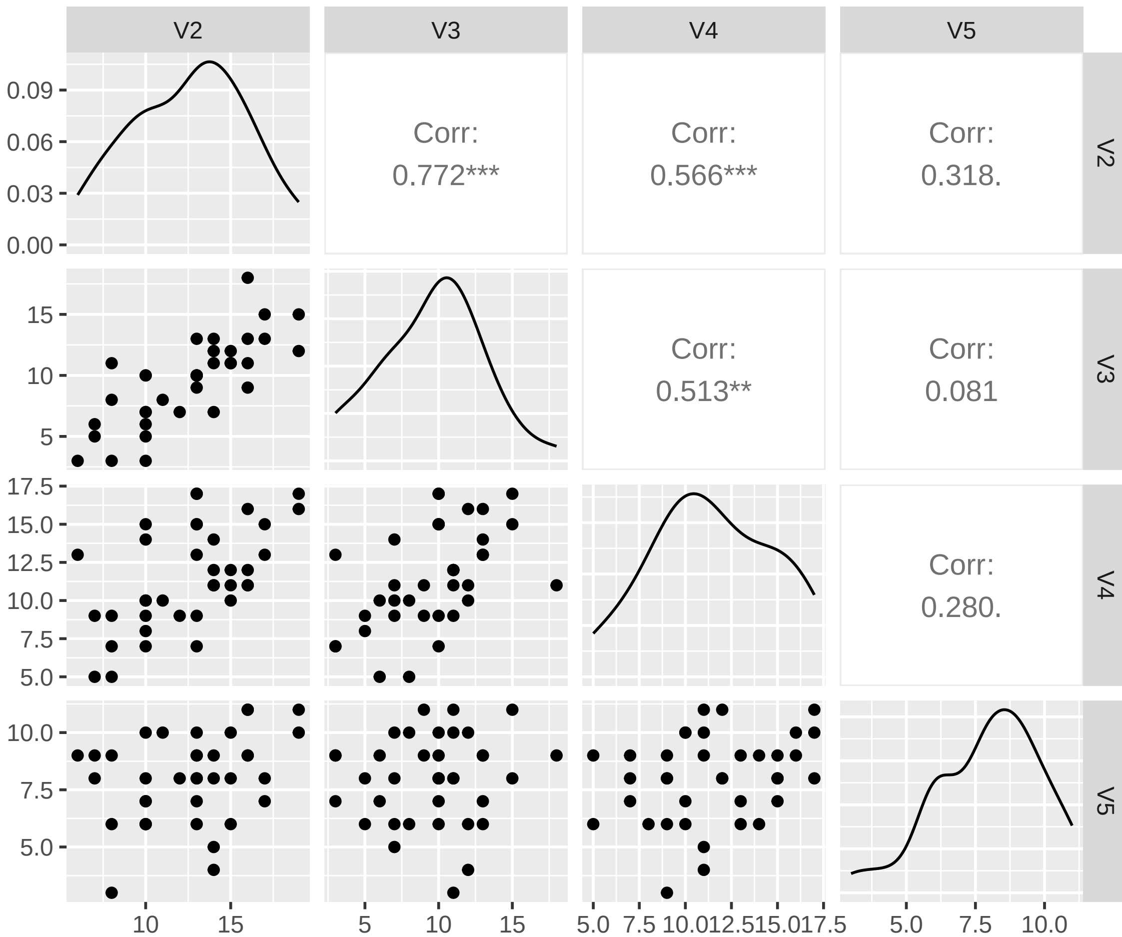
```
##            [,1]       [,2]       [,3]        [,4]
## [1,] -0.6057467  0.2176473  0.4605028  0.61125912
```

```
## [2,] -0.6047618  0.4958117 -0.3196759 -0.53501516
## [3,] -0.5051337 -0.7946452 -0.3349263  0.03468877
## [4,] -0.1103360 -0.2744802  0.7573433 -0.58216643
```

```
d.square <- apply(data[, -1], 1, function(x) t(x - mu) %*% solve(Sigma) %*% (x - mu))
n <- dim(data)[1]; p <- dim(data[, - 1])[2]
chiquant <- qchisq((1:n - 0.5) / n, p)
par(las = 1, mgp = c(2, 1, 0), mar = c(3.5, 3.5, 0.8, 0.6))
plot(chiquant, sort(d.square), pch = "*", col = "blue",
     xlab = "Theoretical quantile", ylab = "Sample quantile", cex = 1.6)
abline(0, 1, col = "red")
```



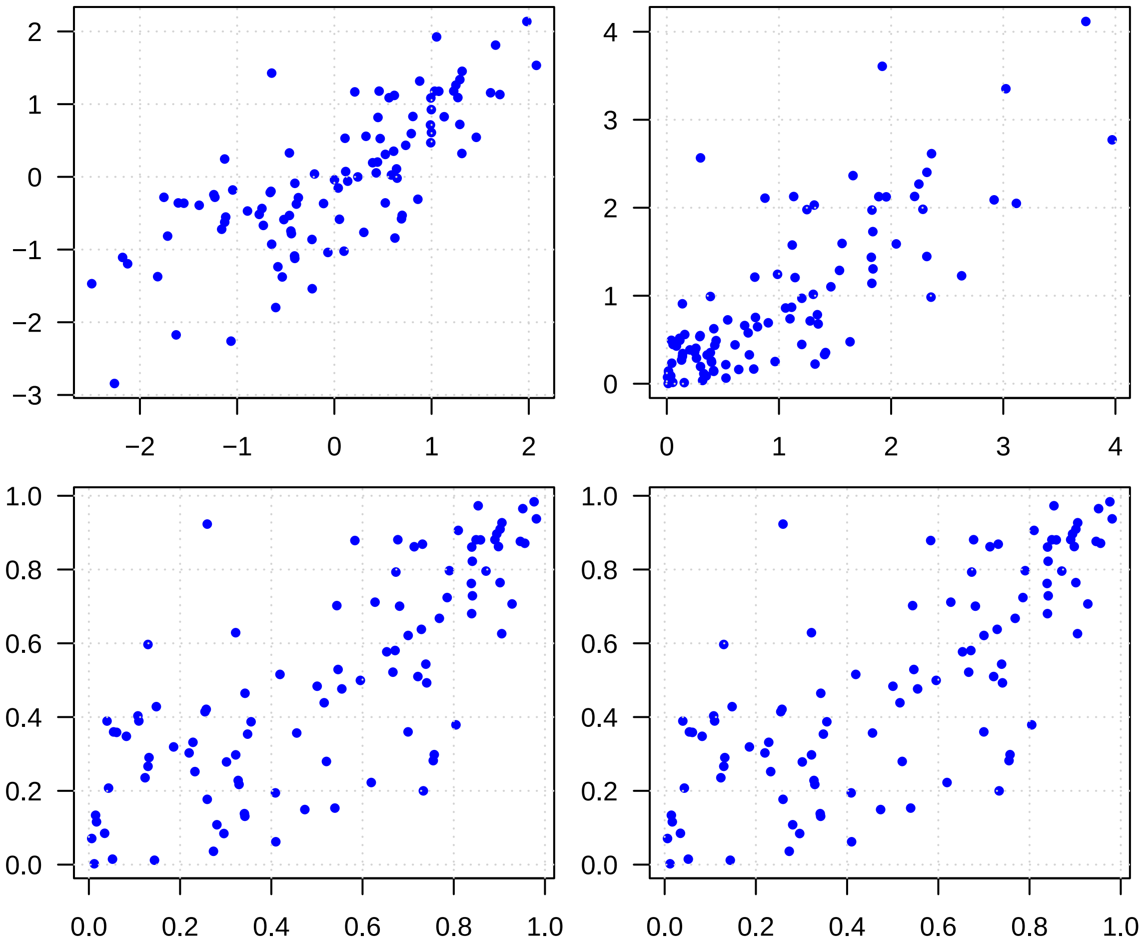

```
library(GGally)
ggpairs(data[, -1])
```



Copula

An Illustration of Bivariate Gaussian Copula

```
mu <- c(0, 0)
Sigma <- matrix(c(1, 0.7, 0.7, 1), 2)
library(MASS)
normal <- mvrnorm(n = 100, mu, Sigma)
par(las = 1, mgp = c(2, 1, 0), mfrow = c(2, 2), mar = c(2, 2.4, 0.8, 0.6))
plot(normal, pch = 16, col = "blue", cex = 0.8, xlab = "", ylab = "")
grid()
U <- apply(normal, 2, pnorm)
plot(qexp(U[, 1]), qexp(U[, 2]), pch = 16, col = "blue", cex = 0.8, xlab = "", ylab = "")
grid()
plot(U[, 1], U[, 2], pch = 16, col = "blue", cex = 0.8, xlab = "", ylab = "")
grid()
plot(U[, 1], U[, 2], pch = 16, col = "blue", cex = 0.8, xlab = "", ylab = "")
grid()
```

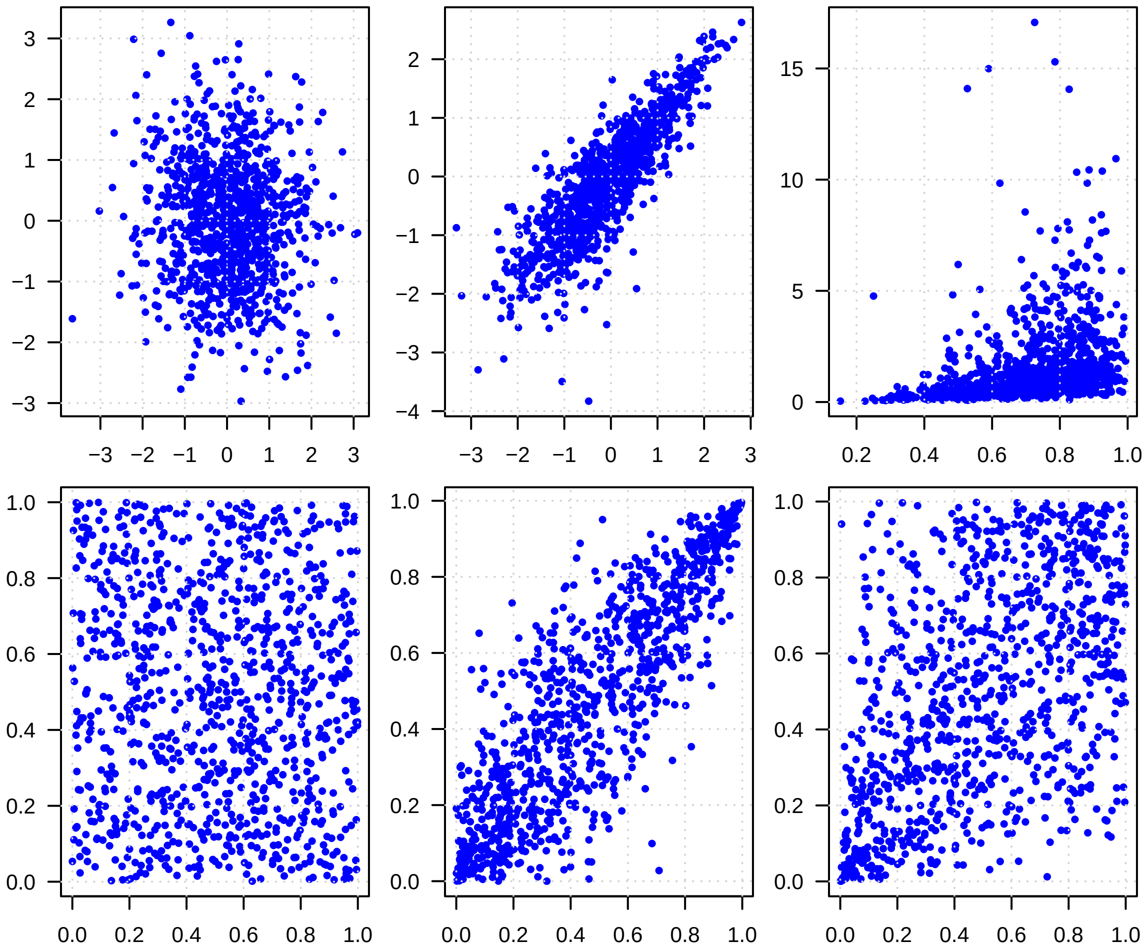


More Examples

- Left: normal + Gaussian copula ($\rho = 0$)
- Middle: normal + Gumbel copula
- Right: Beta + Log-normal + Clayton copula

```
library(VineCopula)
par(las = 1, mgp = c(2, 1, 0), mfrow = c(2, 3),
    mar = c(2, 2.4, 0.8, 0.6))
case1 <- cbind(rnorm(1000), rnorm(1000))
plot(case1, pch = 16, col = "blue", cex = 0.8, xlab = "", ylab = "")
grid()

GumCop <- BiCop(family = 4, par = 3); UGum <- BiCopSim(1000, GumCop)
plot(qnorm(UGum[, 1]), qnorm(UGum[, 2]), pch = 16, col = "blue", cex = 0.8,
     xlab = "", ylab = "")
grid()

ClayCop <- BiCop(family = 3, par = 0.95); UClay <- BiCopSim(1000, ClayCop)
plot(qbeta(UClay[, 1], 5, 2), qlnorm(UClay[, 2]), pch = 16, col = "blue",
     cex = 0.8, xlab = "", ylab = "")
grid()
```

```r
U <- apply(case1, 2, pnorm)
plot(U[, 1], U[, 2], pch = 16, col = "blue", cex = 0.8, xlab = "", ylab = "")
grid()

plot(UGum[, 1], UGum[, 2], pch = 16, col = "blue", cex = 0.8, xlab = "", ylab = "")
grid()

plot(UClay[, 1], UClay[, 2], pch = 16, col = "blue", cex = 0.8, xlab = "", ylab = "")
grid()
```



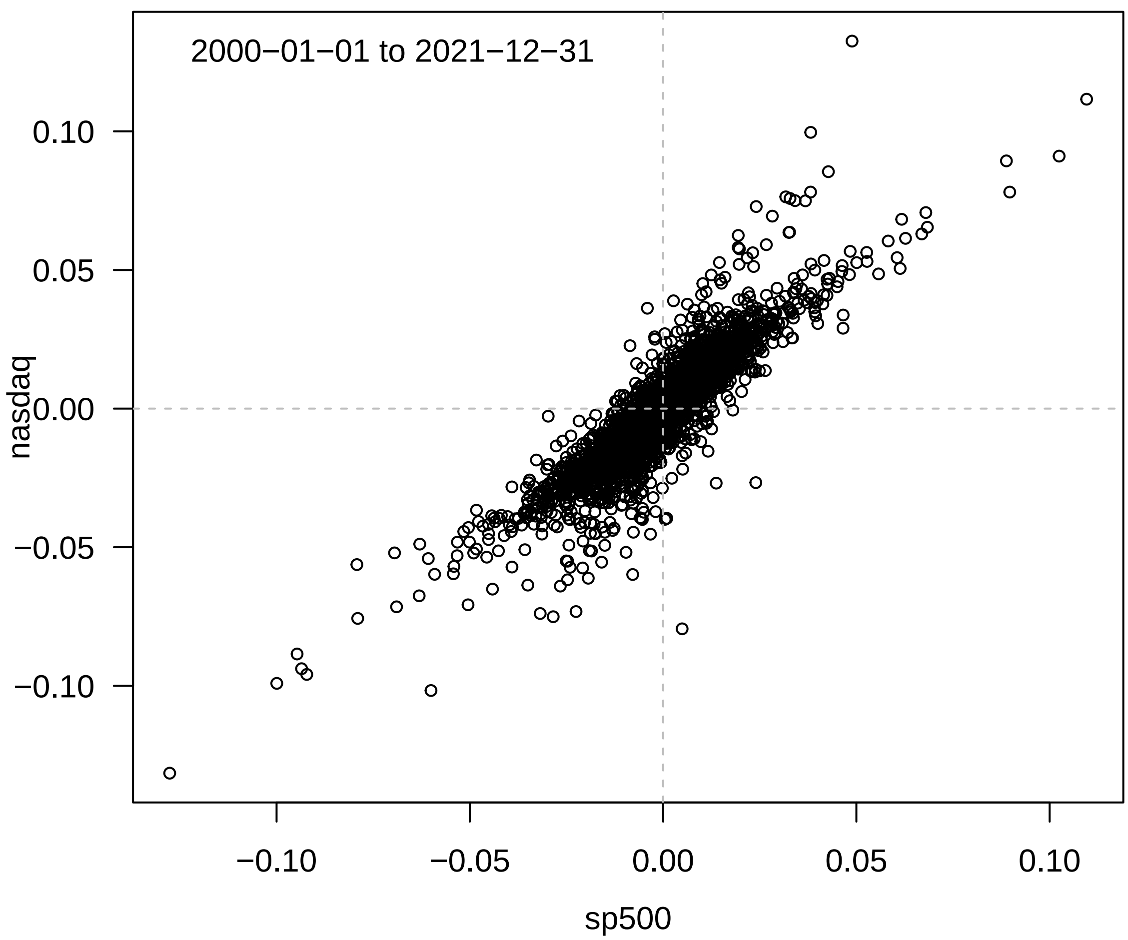
Real Data Example

Here we illustrate how to use a copula to model the bivariate joint distribution of S&P 500 and Nasdaq (log) returns, where a normal distribution is assumed for each variable, while a Student-t copula is used to capture the dependence structure.

```r
library(quantmod)
# Download historical data for S&P 500 and NASDAQ Composite Index
getSymbols(c("^GSPC", "^IXIC"), from = "2000-01-01", to = "2021-12-31")
```

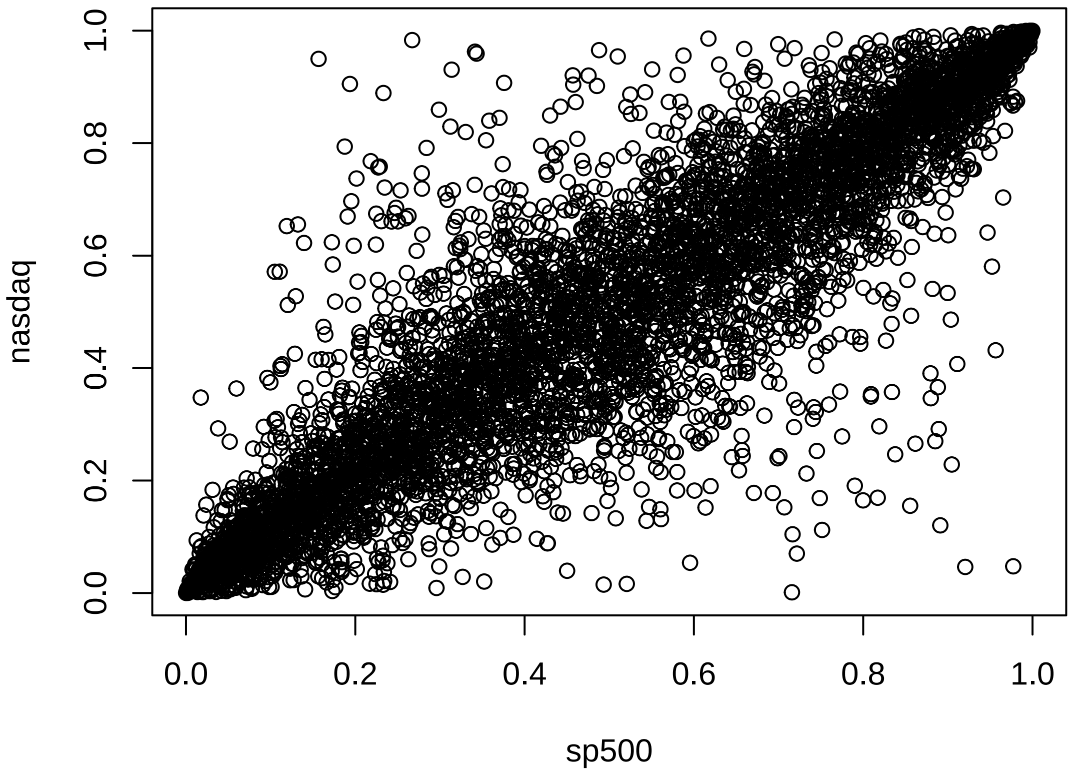
```
## [1] "^GSPC" "^IXIC"
```

```r
sp500 <- as.numeric(GSPC[, "GSPC.Close"])
nasdaq <- as.numeric(IXIC[, "IXIC.Close"])
# Create a data object with both series
data <- cbind(sp500, nasdaq)
# Calculate returns
returns <- diff(log(data), lag = 1)
# plot the returns
par(las = 1, mar = c(3.6, 3.6, 0.8, 0.6), mgp = c(2.5, 1, 0))
plot(returns, cex = 0.75)
abline(v = 0, lty = 2, col = "gray")
abline(h = 0, lty = 2, col = "gray")
legend("topleft", legend = "2000-01-01 to 2021-12-31", bty = "n")
```



Let's fit a copula model

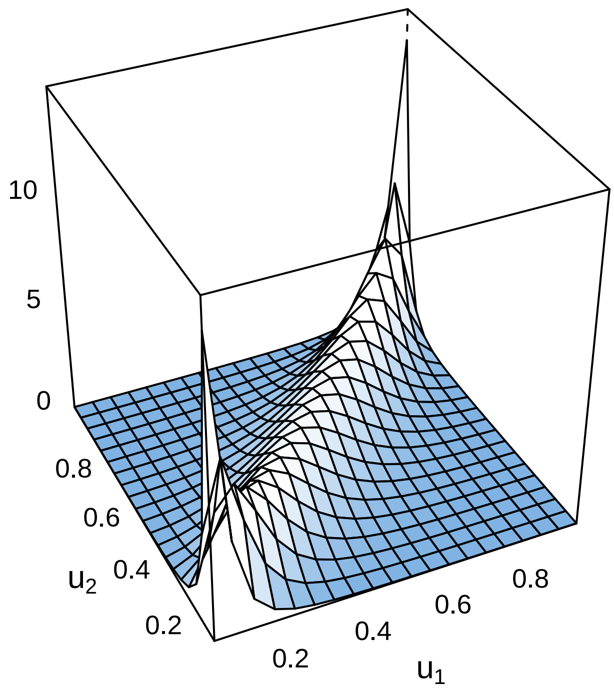
```r
# Compute the empirical percentiles.
u <- apply(returns, 2, function(x) rank(x) / (length(x) + 1))
plot(u)
```




```r
# Select a copula family
library(VineCopula)
selectedCopula <- BiCopSelect(u[, 1], u[, 2], familyset = NA)
selectedCopula
```

```
## Bivariate copula: t (par = 0.92, par2 = 3.45, tau = 0.74)
```

```r
# Fit the selected copula model to the data
copula_fit <- BiCopEst(u[, 1], u[, 2], family = 2)
# Plot the fitted copula
plot(copula_fit)
```



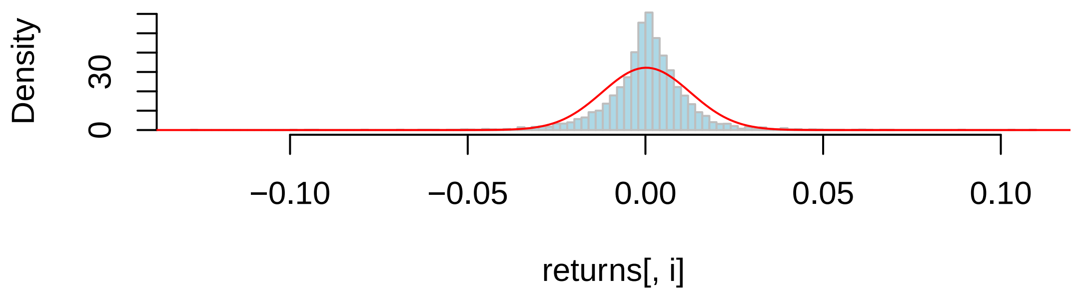

```r
# Summarize the copula fit
summary(copula_fit)
```

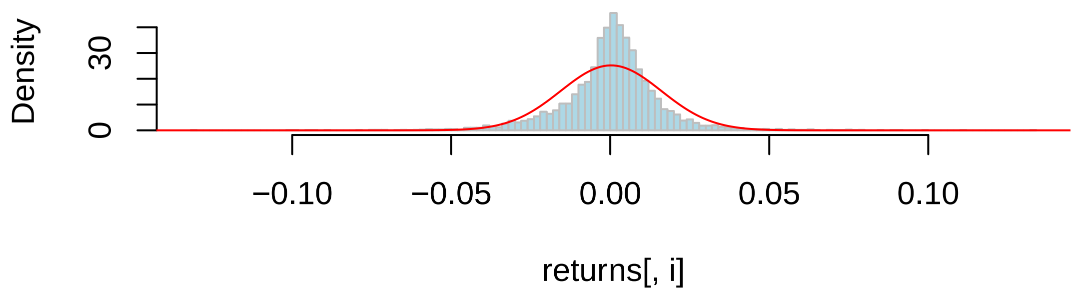
```
## Family
## ------
## No:    2
## Name:  t
##
## Parameter(s)
## ------------
## par:  0.92
## par2: 3.45
## Dependence measures
## -------------------
## Kendall's tau:    0.74 (empirical = 0.74, p value < 0.01)
## Upper TD:         0.68
## Lower TD:         0.68
##
## Fit statistics
## --------------
## logLik:  5228.59
## AIC:    -10453.19
## BIC:    -10439.95
```

Taking care of marginal distributions

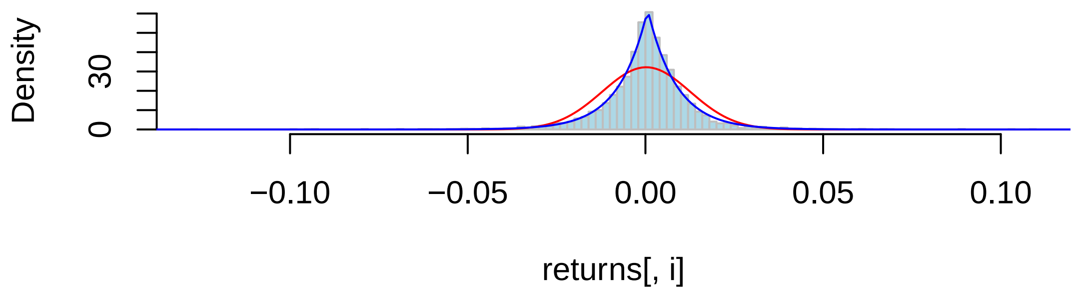
```r
# Taking care of marginal distributions
## Normal
library(fitdistrplus)
norm_fit <- apply(returns, 2, fitdist, "norm")
par(mfrow = c(2, 1))
```

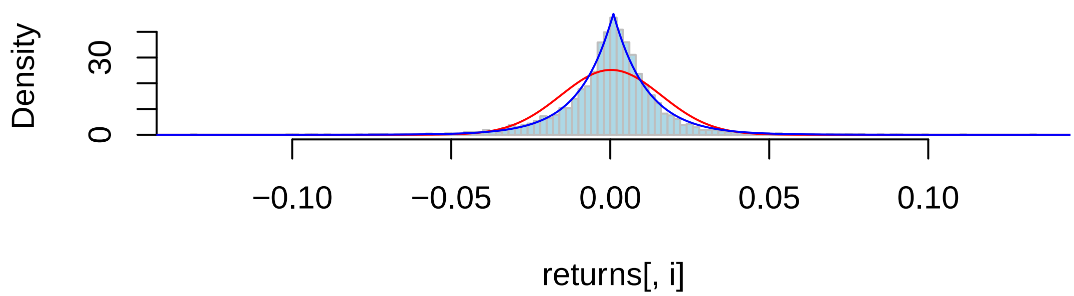
```r
for (i in 1:2){
  hist(returns[, i], 100, col = "lightblue", border = "gray", prob = T,
       main = "")
  lines(seq(-0.2, 0.2, 0.001), dnorm(seq(-0.2, 0.2, 0.001),
                                     norm_fit[[i]]$estimate[1],
                                     norm_fit[[i]]$estimate[2]), col = "red")
}
```






```r
## Laplace
Laplace_est <- apply(returns, 2, function(x) c(median(x), mean(abs(x - median(x)))))
library(rmutil)

for (i in 1:2){
  hist(returns[, i], 100, col = "lightblue", border = "gray", prob = T,
       main = "")
  lines(seq(-0.2, 0.2, 0.001), dnorm(seq(-0.2, 0.2, 0.001),
                                     norm_fit[[i]]$estimate[1],
                                     norm_fit[[i]]$estimate[2]), col = "red")
  lines(seq(-0.2, 0.2, 0.001), dlaplace(seq(-0.2, 0.2, 0.001),
                                        Laplace_est[1, i], Laplace_est[2, i]),
        col = "blue")
}
```



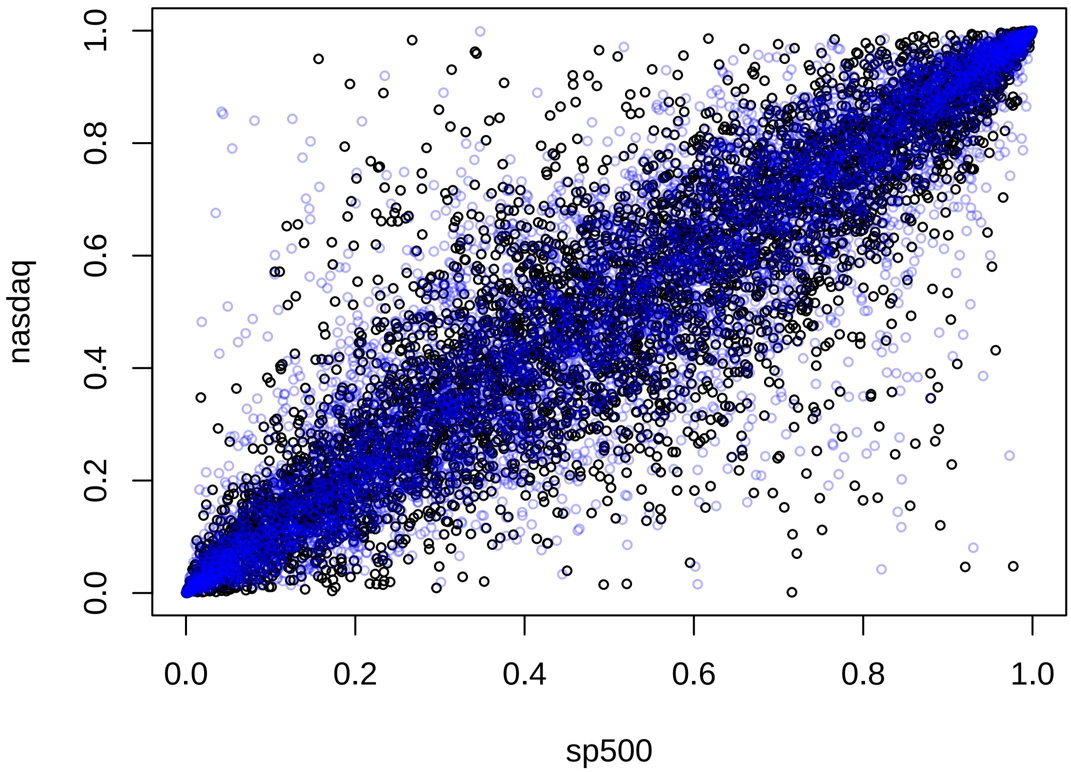


Let's put everything together to simulate new 'data'

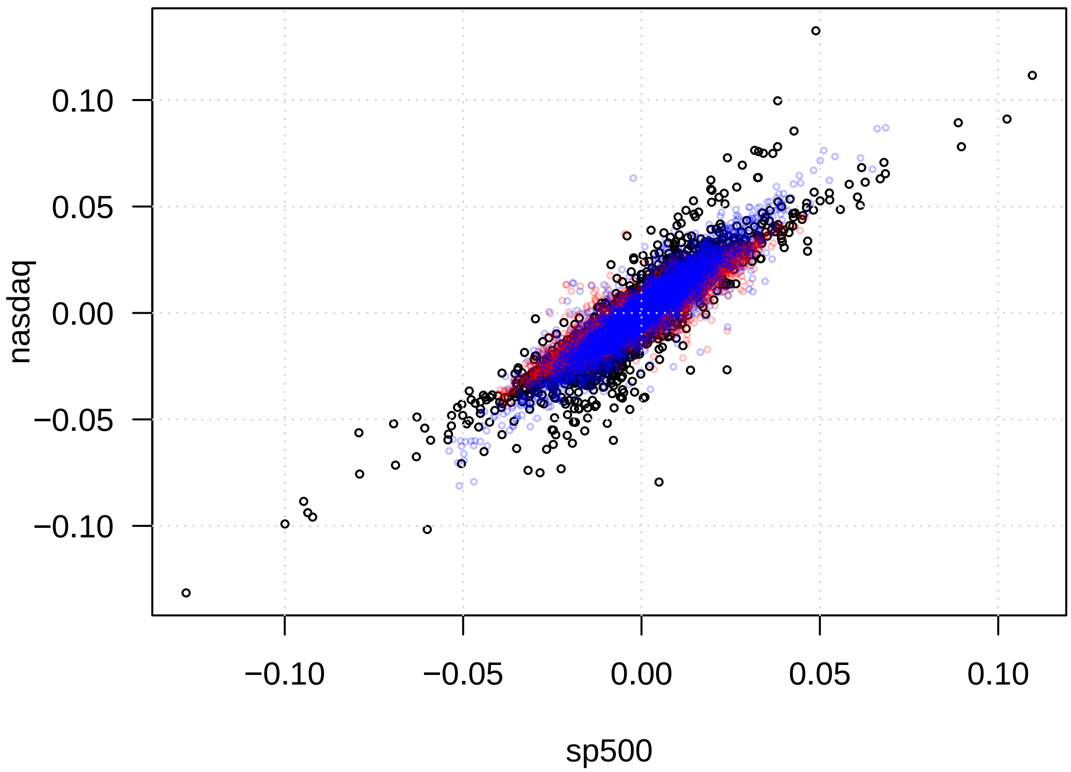
```
library(scales)
```

```
##
## Attaching package: 'scales'

## The following object is masked from 'package:viridis':
##
##     viridis_pal
```

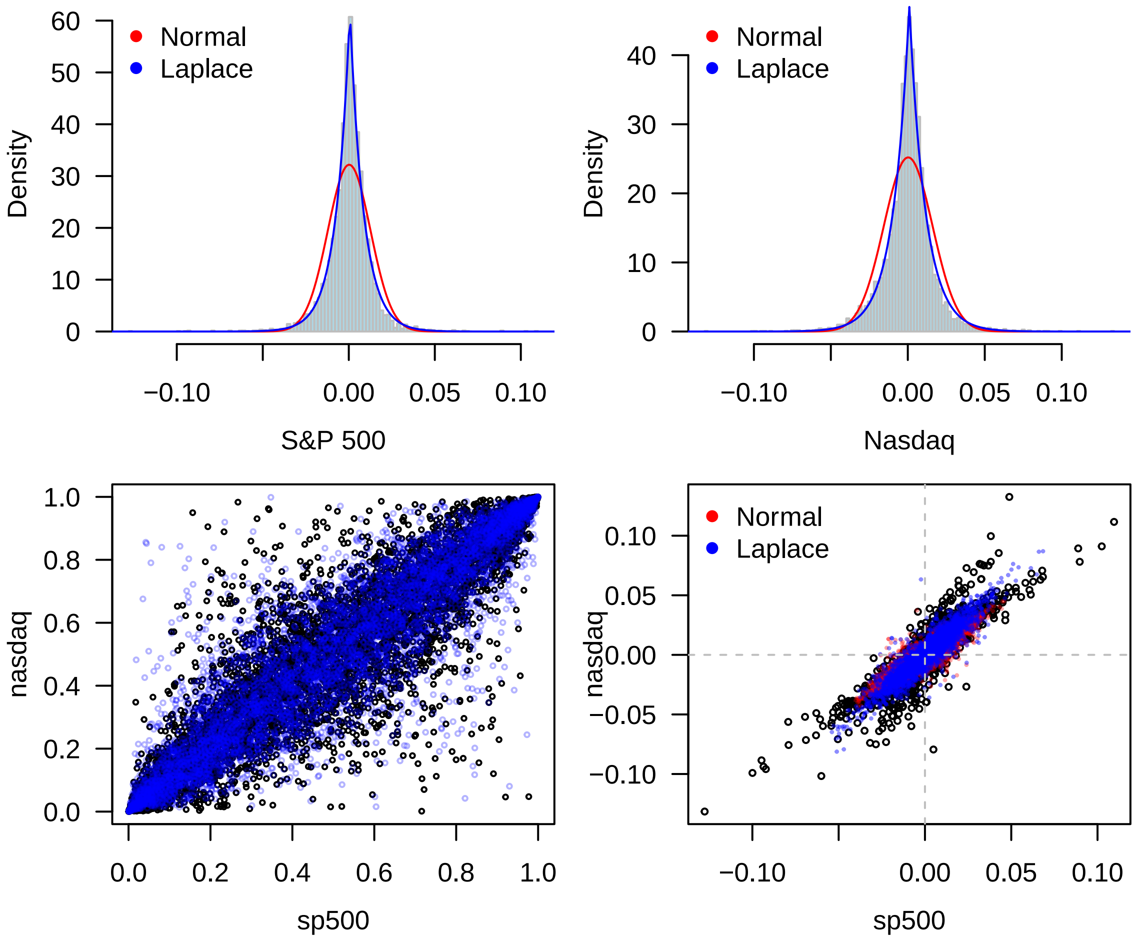
```
n <- dim(returns)[1]
# Simulate new (pseudo) observations
u_sim <- BiCopSim(n, selectedCopula)
plot(u, cex = 0.6)
points(u_sim, cex = 0.6, col = alpha("blue", 0.3))
```




```r
# Convert back to the original scale using the fitted marginals
x_sim1 <- qnorm(u_sim[, 1], norm_fit[[1]]$estimate[1], norm_fit[[1]]$estimate[2])
y_sim1 <- qnorm(u_sim[, 2], norm_fit[[1]]$estimate[1], norm_fit[[1]]$estimate[2])

x_sim2 <- qlaplace(u_sim[, 1], Laplace_est[1, 1], Laplace_est[2, 1])
y_sim2 <- qlaplace(u_sim[, 2], Laplace_est[1, 2], Laplace_est[2, 2])

plot(returns, cex = 0.5, las = 1)
points(x_sim1, y_sim1, cex = 0.4, col = alpha("red", 0.25))
points(x_sim2, y_sim2, cex = 0.4, col = alpha("blue", 0.25))
grid()
```



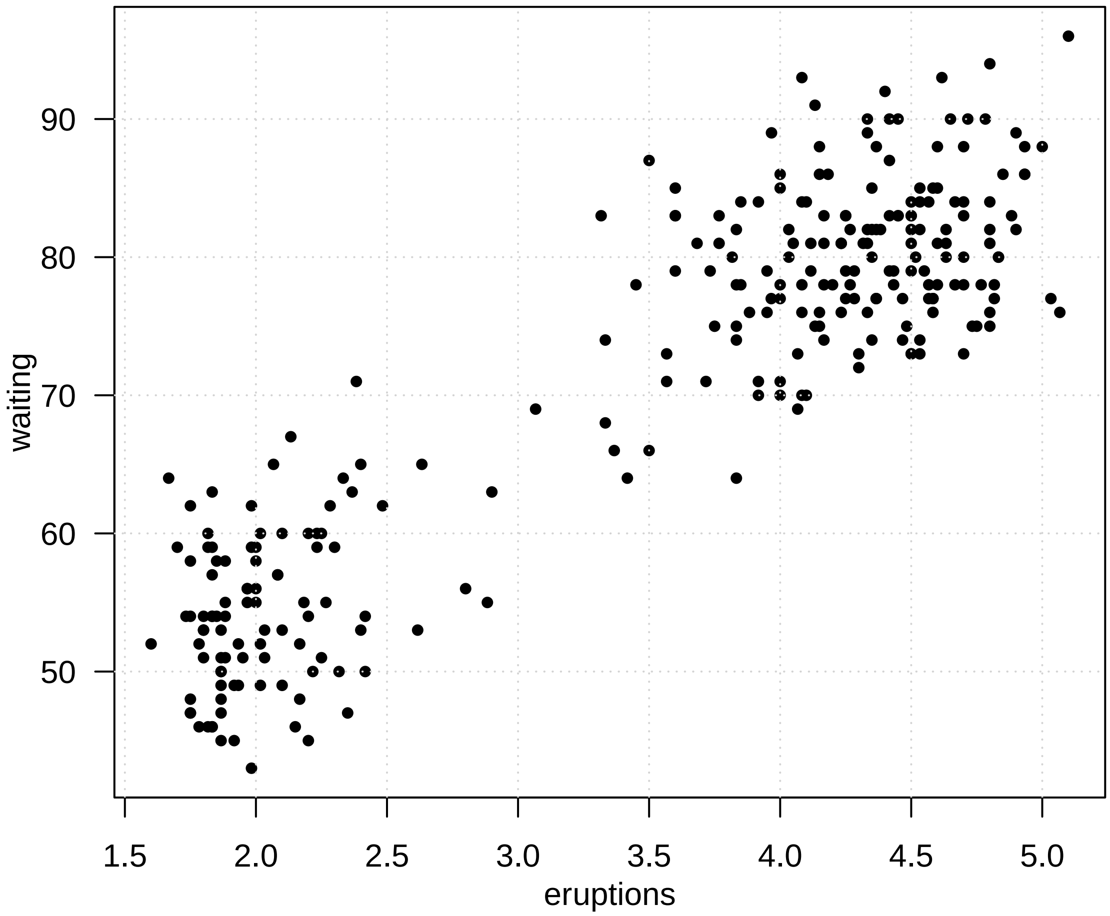

```r
par(las = 1, mar = c(3.6, 3.6, 0.8, 0.6), mgp = c(2.5, 1, 0), mfrow = c(2, 2))

stock <- c("S&P 500", "Nasdaq")
for (i in 1:2){
  hist(returns[, i], 100, col = "lightblue", border = "gray", prob = T,
       main = "", xlab = stock[i])
  legend("toplef", legend = c("Normal", "Laplace"), pch = 16,
         col = c("red", "blue"), bty = "n")
  lines(seq(-0.2, 0.2, 0.001), dnorm(seq(-0.2, 0.2, 0.001),
                                     norm_fit[[i]]$estimate[1],
                                     norm_fit[[i]]$estimate[2]), col = "red")
  lines(seq(-0.2, 0.2, 0.001), dlaplace(seq(-0.2, 0.2, 0.001),
                                        Laplace_est[1, i], Laplace_est[2, i]),
        col = "blue")
}

plot(u, cex = 0.4)
points(u_sim, cex = 0.4, col = alpha("blue", 0.3))

plot(returns, cex = 0.5)

points(x_sim1, y_sim1, cex = 0.2, col = alpha("red", 0.35))
points(x_sim2, y_sim2, cex = 0.2, col = alpha("blue", 0.45))
abline(v = 0, lty = 2, col = "gray")
abline(h = 0, lty = 2, col = "gray")
legend("toplef", legend = c("Normal", "Laplace"), pch = 16,
         col = c("red", "blue"), bty = "n")
```



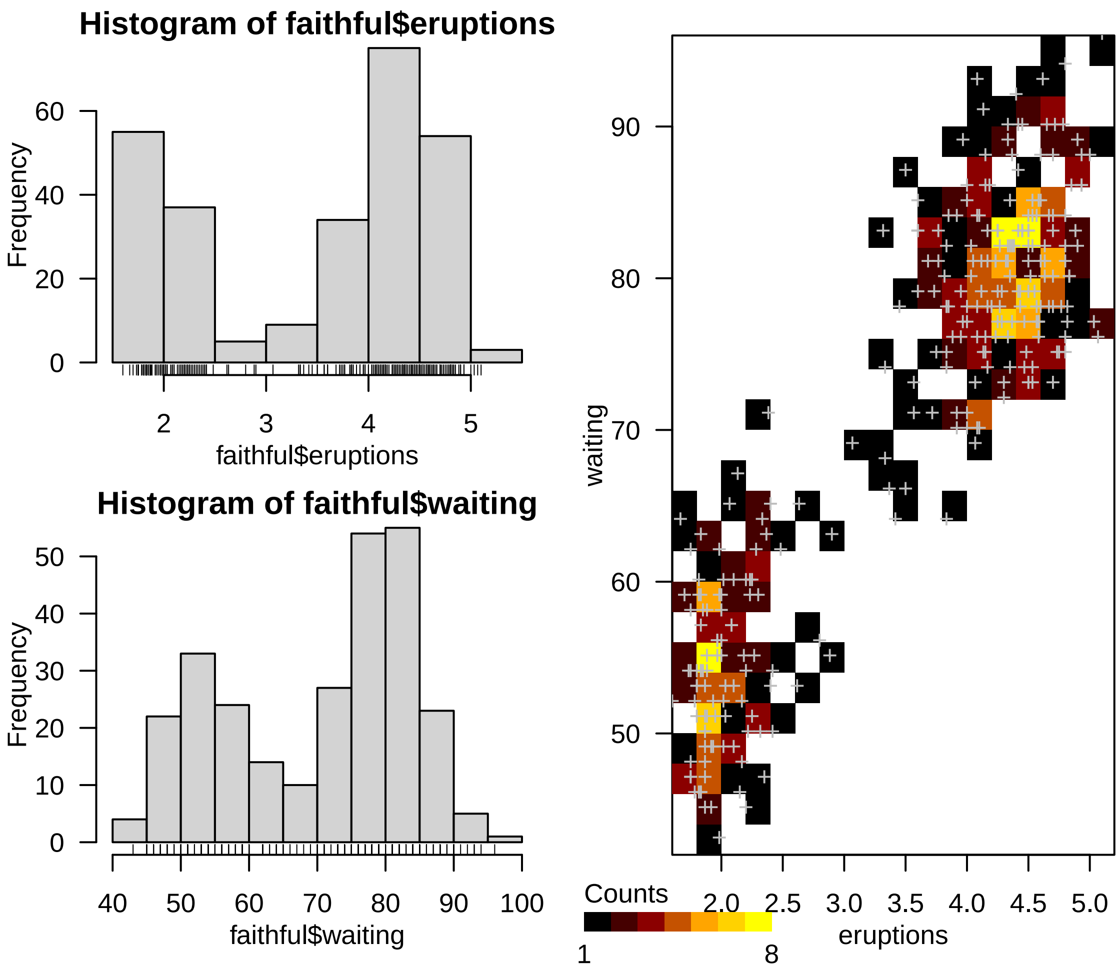
Nonparametric Density Estimation

Histogram

```r
data(faithful)
par(las = 1, mgp = c(2, 1, 0), mar = c(3.6, 3.6, 0.8, 0.6))
plot(faithful, las = 1, cex = 0.8, pch = 16)
grid()
```




```r
par(las = 1, mgp = c(2, 1, 0), mar = c(3.6, 3.6, 0.8, 0.6))
layout(matrix(c(1, 2, 3, 3), nrow = 2, ncol = 2))
hist(faithful$eruptions)
rug(faithful$eruptions)
hist(faithful$waiting, las = 1)
rug(faithful$waiting)
library(squash)
hist2(faithful, nx = 20, ny = 20, las = 1)
points(faithful, pch = "+", col = "gray")
```

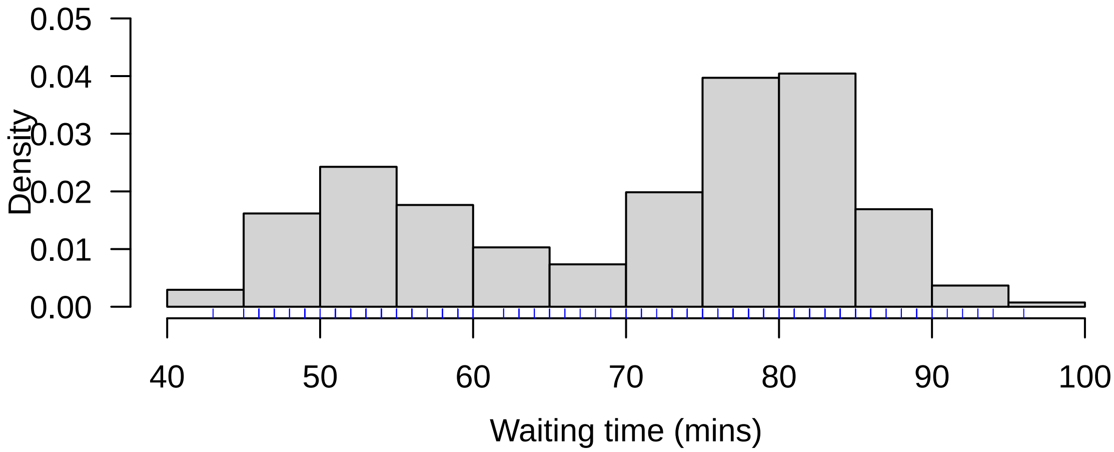


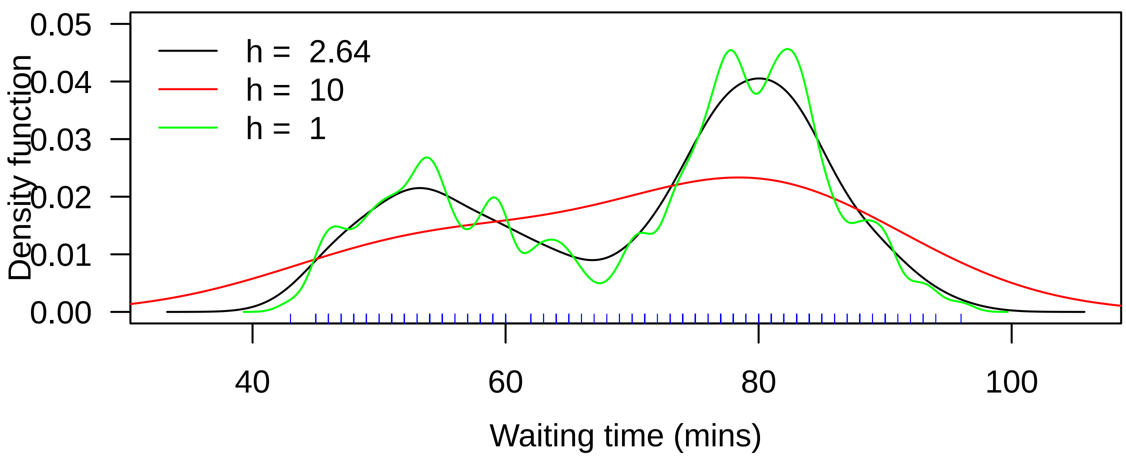
Transition from Histogram to Kernel Density

```
library(ks)
par(las = 1, mfrow = c(2, 1), mar = c(3.6, 3.6, 0.8, 0.6), mgp = c(2.4, 1, 0))
hist(faithful$waiting, xlab = "Waiting time (mins)", prob = T, main = "",
     ylim = c(0, 0.05))
rug(faithful$waiting, col = "blue")

plot(kde(faithful$waiting), xlab = "Waiting time (mins)", ylim = c(0, 0.05))
```

```
## Warning in fun(libname, pkgname): couldn't connect to display
## "/private/tmp/com.apple.launchd.E90mwpBJGj/org.xquartz:0"
```

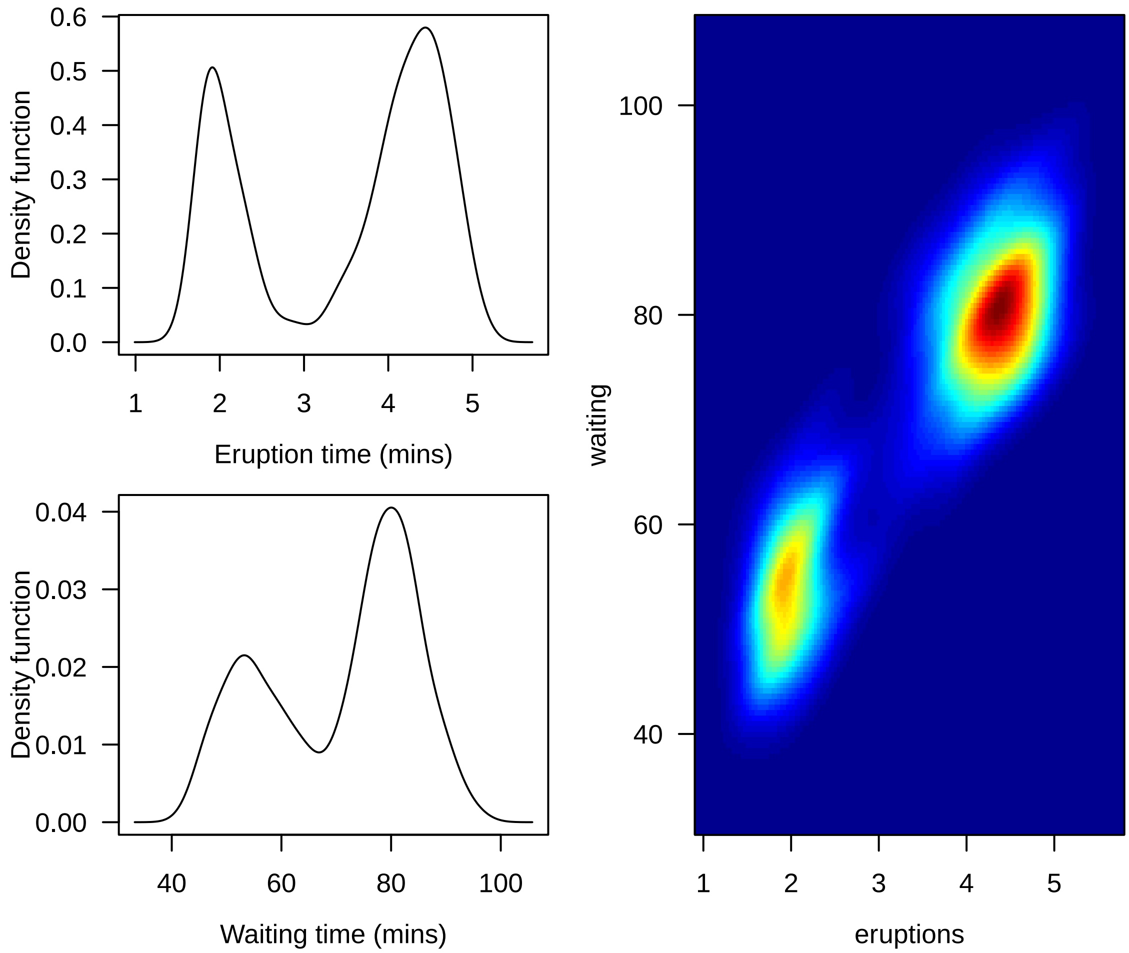
```
plot(kde(faithful$waiting, h = 10), col = "red", add = T)
plot(kde(faithful$waiting, h = 1), col = "green", add = T)
legend("topleft", legend = paste("h = ", c(2.64, 10, 1)),
       col = c("black", "red", "green"), lty = 1, bty = "n")
rug(faithful$waiting, col = "blue")
```





Kernel Density Estimation (KDE)

```
par(las = 1, mgp = c(2.6, 1, 0), mar = c(3.6, 4, 0.8, 0.6))
layout(matrix(c(1, 2, 3, 3), nrow = 2, ncol = 2))
plot(kde(faithful$eruptions), xlab = "Eruption time (mins)")
plot(kde(faithful$waiting), xlab = "Waiting time (mins)")
Hpi1 <- Hpi(x = faithful)
kdeHpi1 <- kde(x = faithful, H = Hpi1)
plot(kdeHpi1, display = "image", col = tim.colors())
```



Density function
Eruption time (mins)
Density function
Waiting time (mins)
waiting
eruptions