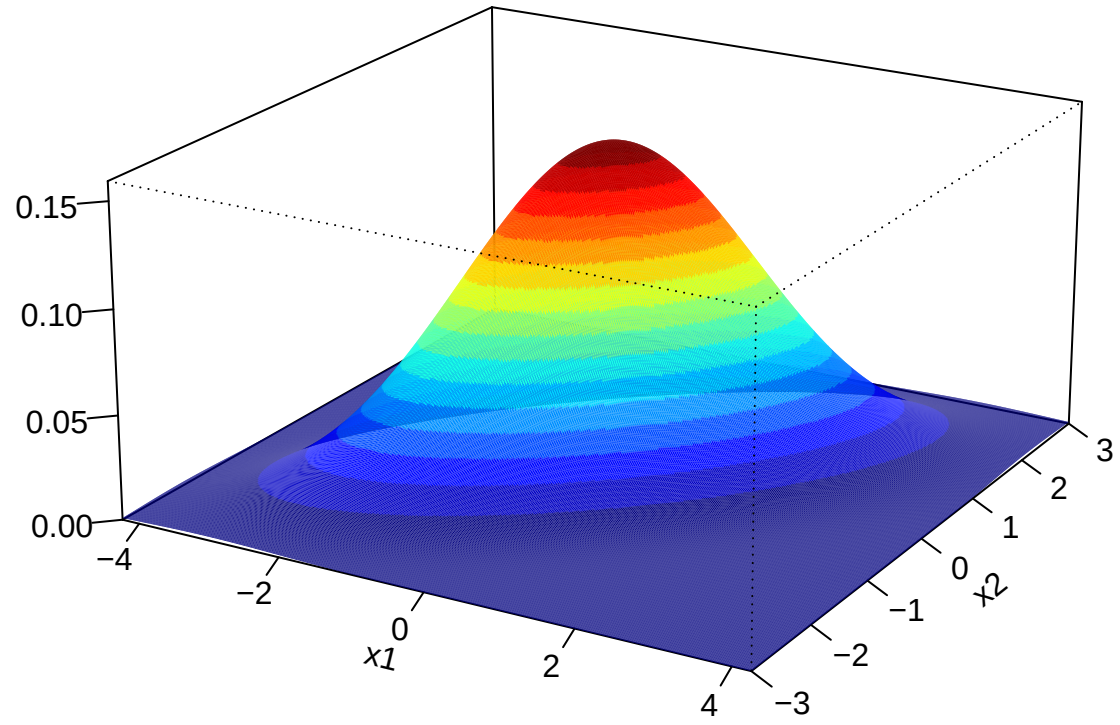
Multivariate Normal Distribution, Copula, and Nonparametric Density Estimation

Blake Pappas

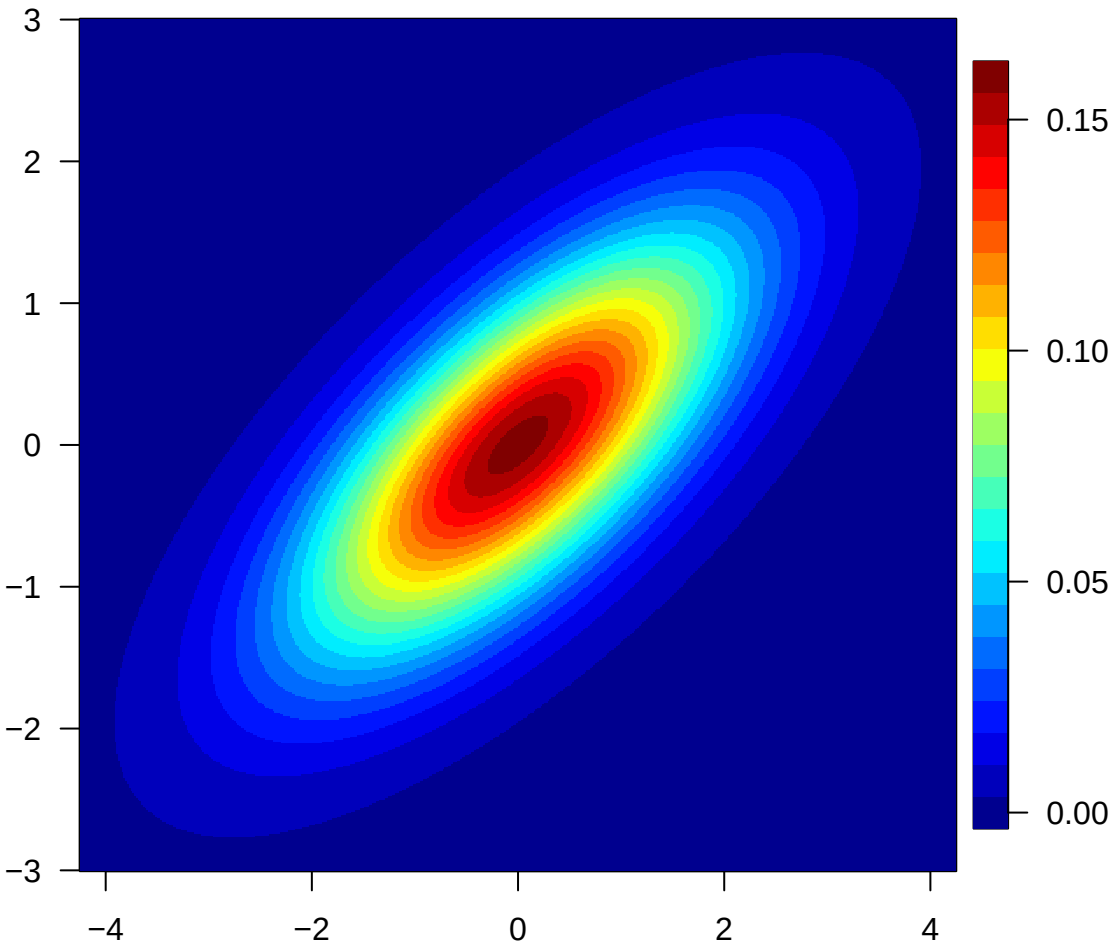
December 19, 2023

Bivariate Normal Density

```r
# Parameters
mu1 <- mu2 <- 0 # Means are both zero
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)

# Get the Grids for Plotting
x1 <- seq(mu1 - 3 * sqrt(sigma11), mu1 + 3 * sqrt(sigma11), length = 500) # 3 = 3 standard deviations;
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)) # dmvnorm() = density for the multi

par(mar = c(1, 2, 0.8, 0.6))
library(GA)
persp3D(x1, x2, out, theta = 30, phi = 20, expand = 0.5, zlab = "")
```

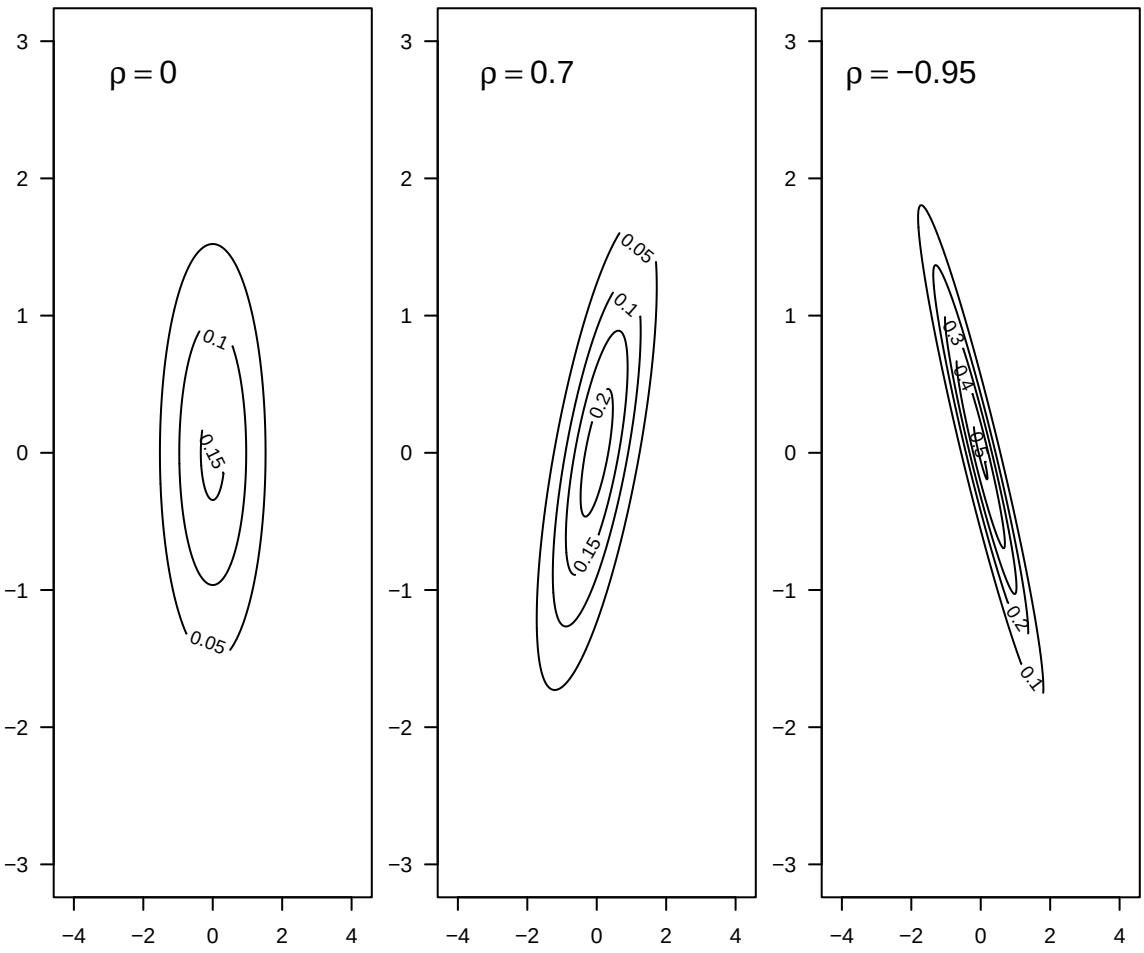



```
par(mar = c(2, 2, 0.8, 0.6), mgp = c(2, 1, 0))
library(fields)
image.plot(x1, x2, out, las = 1, col = tim.colors(24))
```



Bivariate Normal with Different ρ

```
# Creates Contour Plots
par(mfrow = c(1, 3), mar = c(2, 2, 0.8, 0.6))
mu <- c(0, 0)
rho <- c(0, 0.7, -0.95)

# Use a For Loop Because There Are 3 Different Correlation Coefficients That Need to Be Examined (0, 0.
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 = 4)
  text(-2, 2.75, bquote(rho == .(rho[i])), cex = 1.5)
}
```



Multivariate Normal QQ Plot

```
# Creates the QQ Plot
versicolor <- iris[51:100, 1:3] # Looks at the first 3 variables in the iris dataset
(mu <- apply(versicolor, 2, mean)) # Calculates the mean vector for each component
```

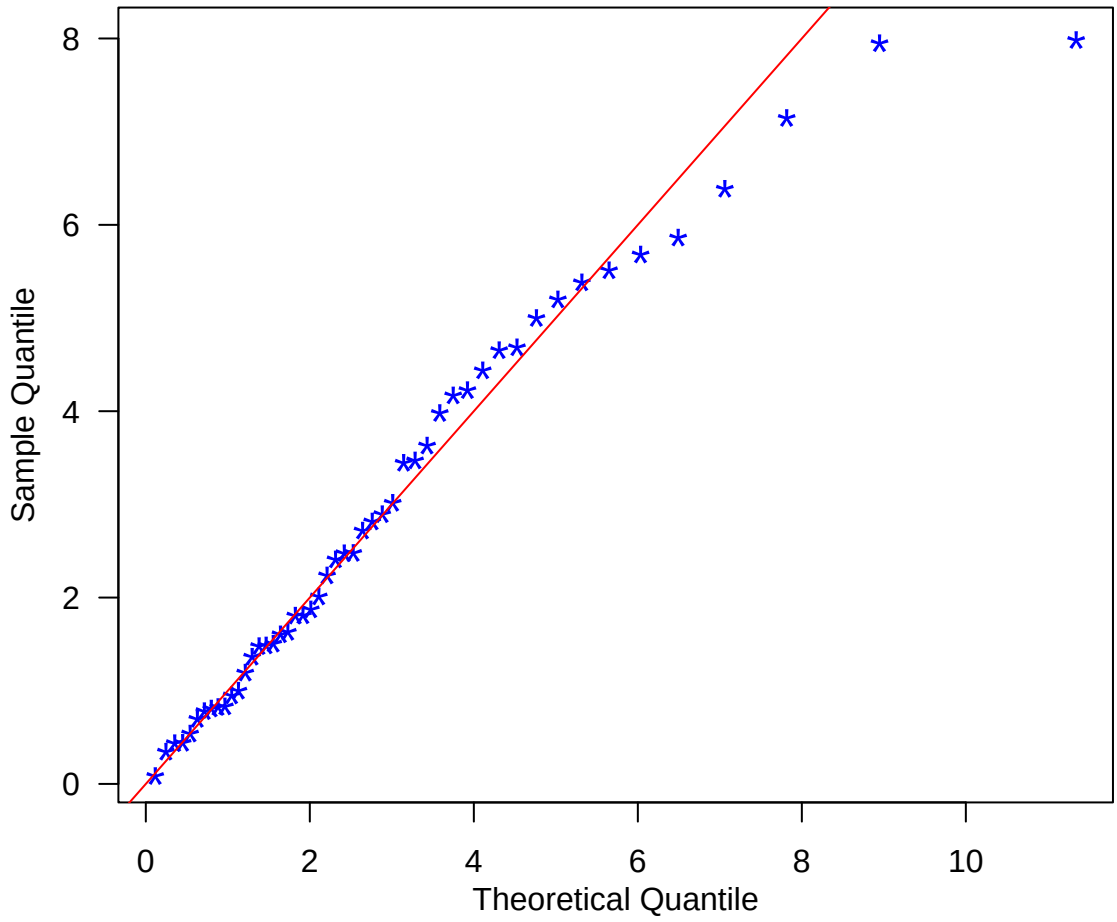
```
## Sepal.Length  Sepal.Width Petal.Length
##        5.936        2.770        4.260
```

```
(Sigma <- cov(versicolor)) # Calculates the covariance matrix
```

```
##              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
```

```
# Empirical Quantiles
d.square <- apply(versicolor, 1, function(x) t(x - mu) %*% solve(Sigma) %*% (x - mu)) # Pulls the squar
```

```r
# Sample Size
n <- dim(versicolor)[1]

# Theoretical Quantiles Under MVN
chiquant <- qchisq((1:n - 0.5) / n, 3) # 3 = degrees of freedom (3 variables)

# 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")
```

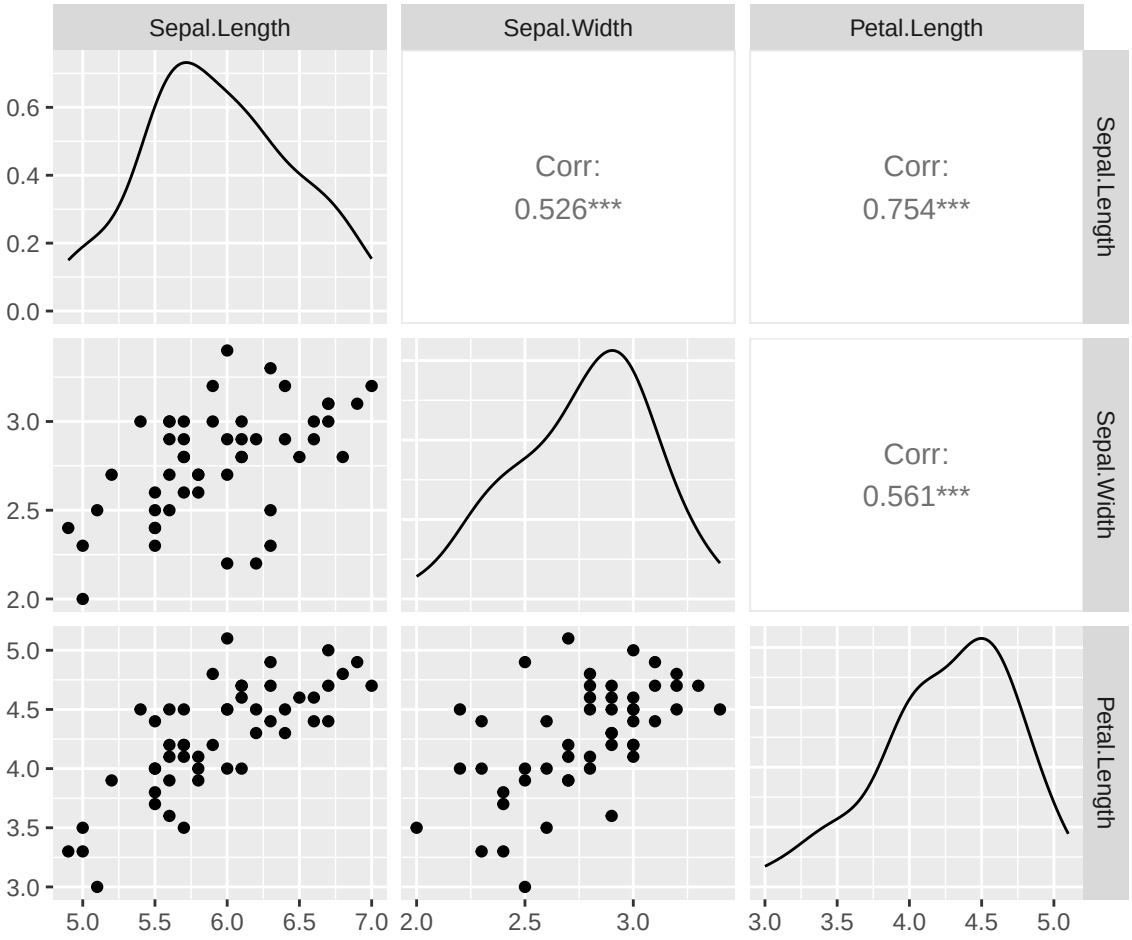



```r
library(GGally)
```

```
## Loading required package: ggplot2
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```r
ggpairs(versicolor)
```

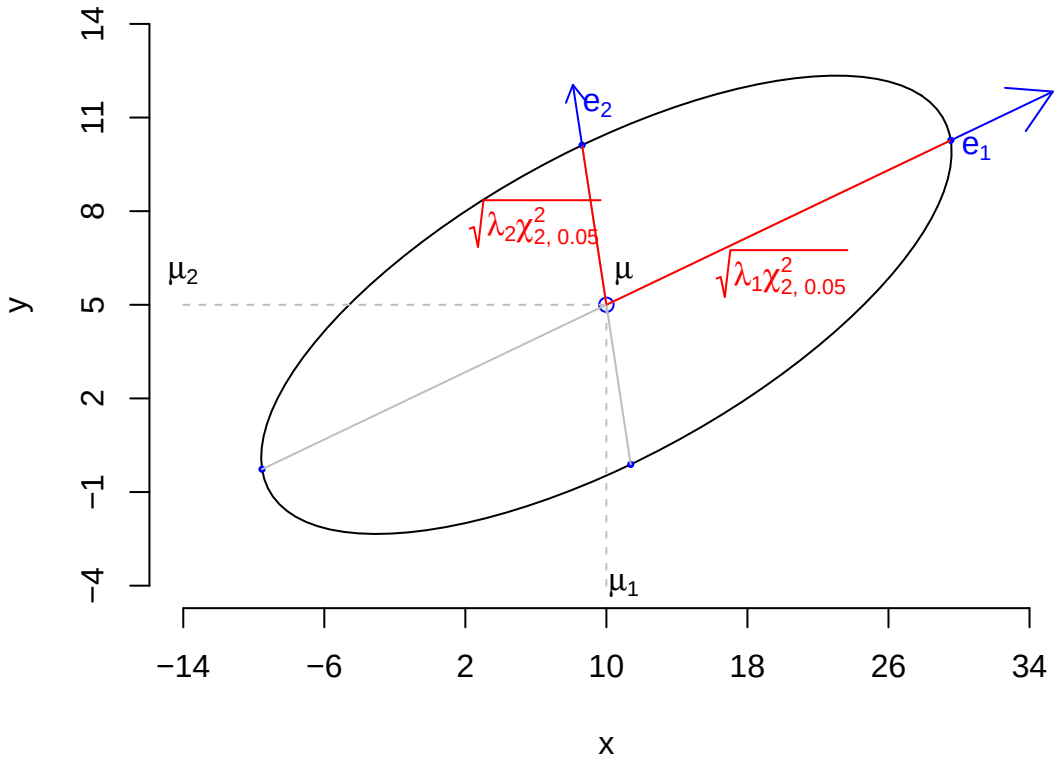


Probability Contour

```r
# Creates the Contour Plot
mu <- c(10, 5) # True mean vector
Sigma <- matrix(c(64, 16, 16, 9), 2) # Covariance matrix
sd1 <- sqrt(diag(Sigma)[1]); sd2 <- sqrt(diag(Sigma)[2]) # Calculates the standard deviation of each di
xg <- seq(mu[1] - 3 * sd1, mu[1] + 3 * sd1, len = 500) # Grid point sequence. 3 standard deviations wit
yg <- seq(mu[2] - 3 * sd2, mu[2] + 3 * sd2, len = 500) # Grid point sequence. 3 standard deviations wit
grids <- expand.grid(xg, yg)
f <- array(dmvnorm(grids, mu, Sigma), dim = c(500, 500))

spetralDecomp <- eigen(Sigma) # Calculates the eigenvalues and eigenvectors
lambda1 <- spetralDecomp$values[1]
lambda2 <- spetralDecomp$values[2]
e1 <- spetralDecomp$vectors[, 1] # Eigenvalue for lambda1 (e1)
e2 <- spetralDecomp$vectors[, 2] # Eigenvalue for lambda2 (e2)
c <- sqrt(qchisq(0.95, 2)) # Chi-square with two degrees of freedom and a 95% confidence level
library(ellipse)
```

```
##
## Attaching package: 'ellipse'

## The following object is masked from 'package:graphics':
##
##     pairs
```

```
cor <- Sigma[1, 2] / (sd1 * sd2)
plot(ellipse(cor, scale = sqrt(diag(Sigma)), centre = mu), type = 'l',
xlim = range(xg), ylim = range(yg), las = 1, bty = "n", yaxt = "n", xaxt = "n")
points(mu[1], mu[2], col = "blue") # Creates the center of the contour by plotting the means
axis(1, at = seq(-14, 34, 8)); axis(2, at = seq(-4, 14, 3))
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

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

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

```
(Sigma <- cov(data[, -1])) # Calculates the covariance matrix
```

```
##           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
```

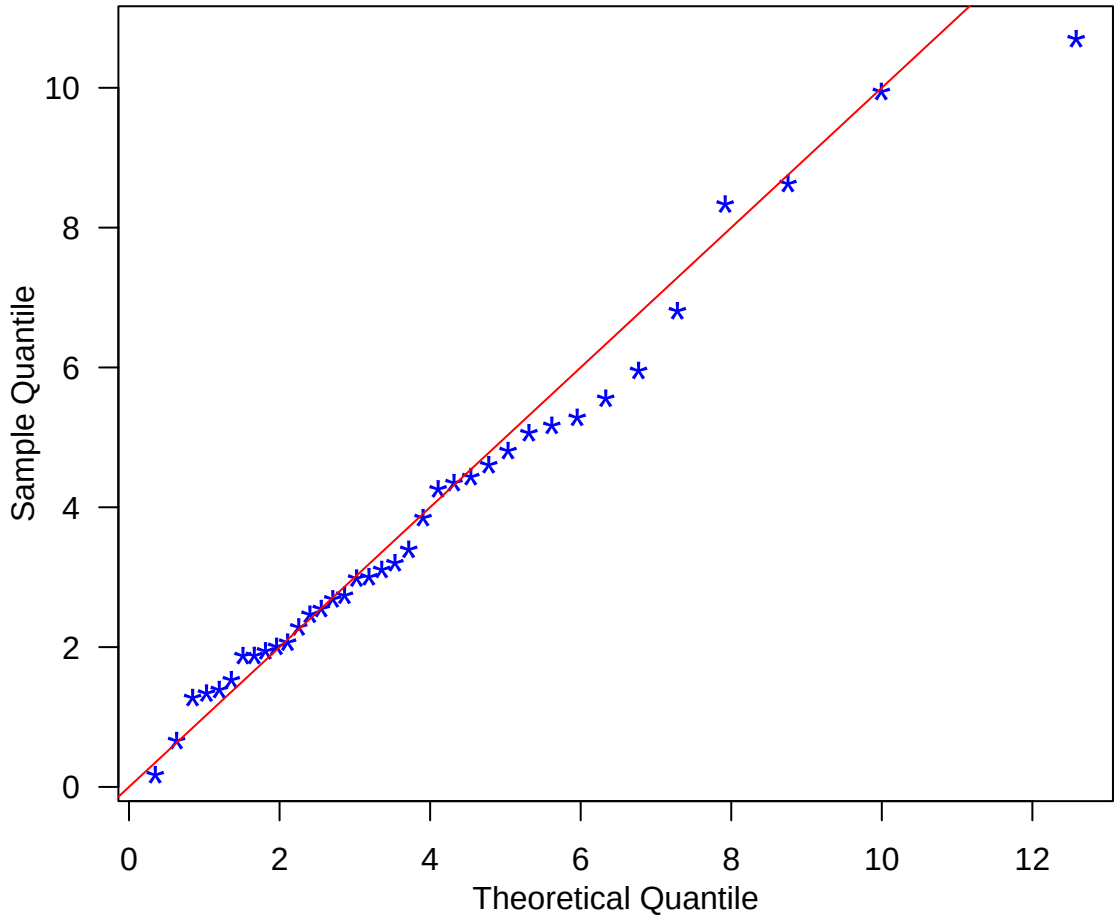
```
out <- eigen(Sigma) # Calculates the eigenvalues and eigenvectors
out$values
```

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

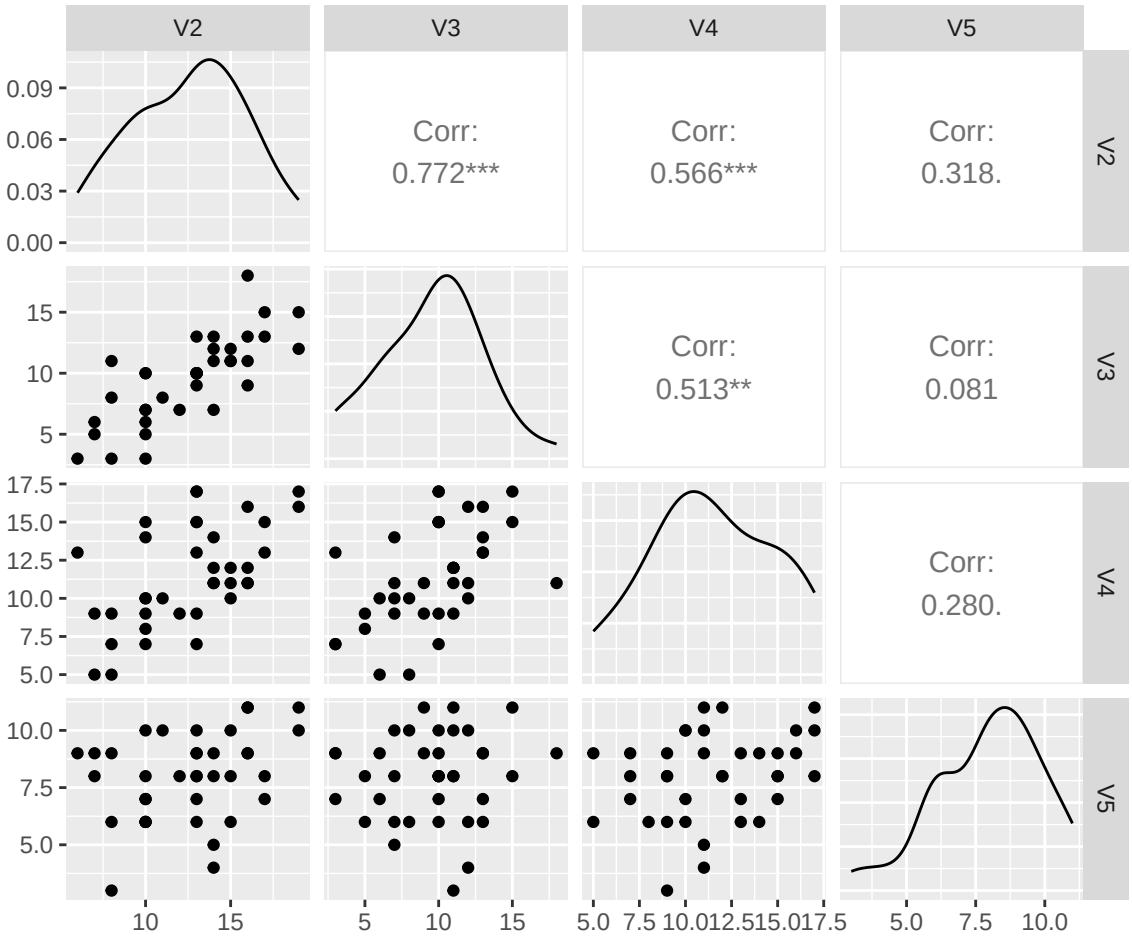
```r
out$vectors
```

```
##            [,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
```

```r
d.square <- apply(data[, -1], 1, function(x) t(x - mu) %*% solve(Sigma) %*% (x - mu)) # Calculates the
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")
```

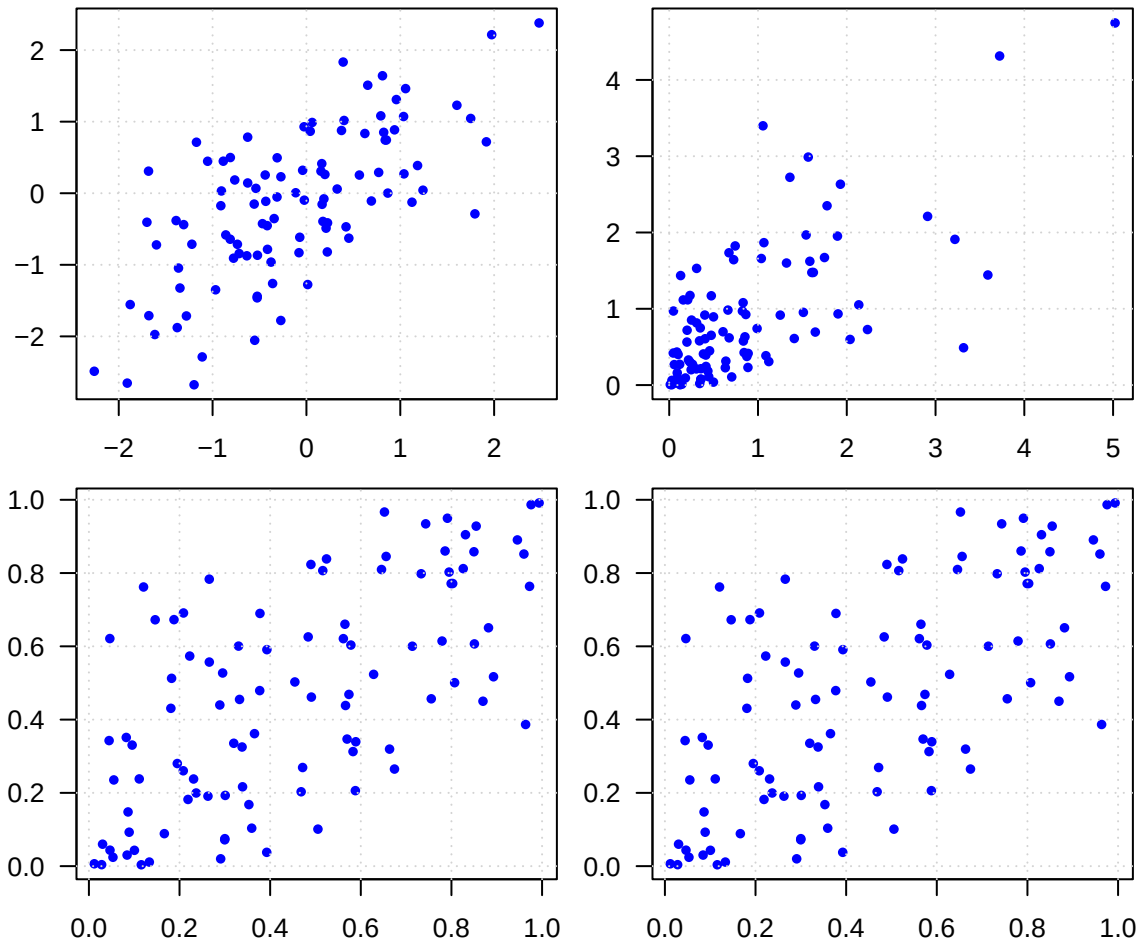



```r
library(GGally)
ggpairs(data[, -1])
```



An Illustration of Bivariate Gaussian Copula

```r
# Simulates a Multivariate Normal Distribution with a Correlation of 0.7
mu <- c(0, 0) # Mean vector
Sigma <- matrix(c(1, 0.7, 0.7, 1), 2) # Covariance matrix
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()

# Translates Into a Copula Scale
U <- apply(normal, 2, pnorm)
plot(qexp(U[, 1]), qexp(U[, 2]), pch = 16, col = "blue", cex = 0.8, xlab = "", ylab = "")
grid()

# Plots the Data Onto a Copula Scale
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()
```

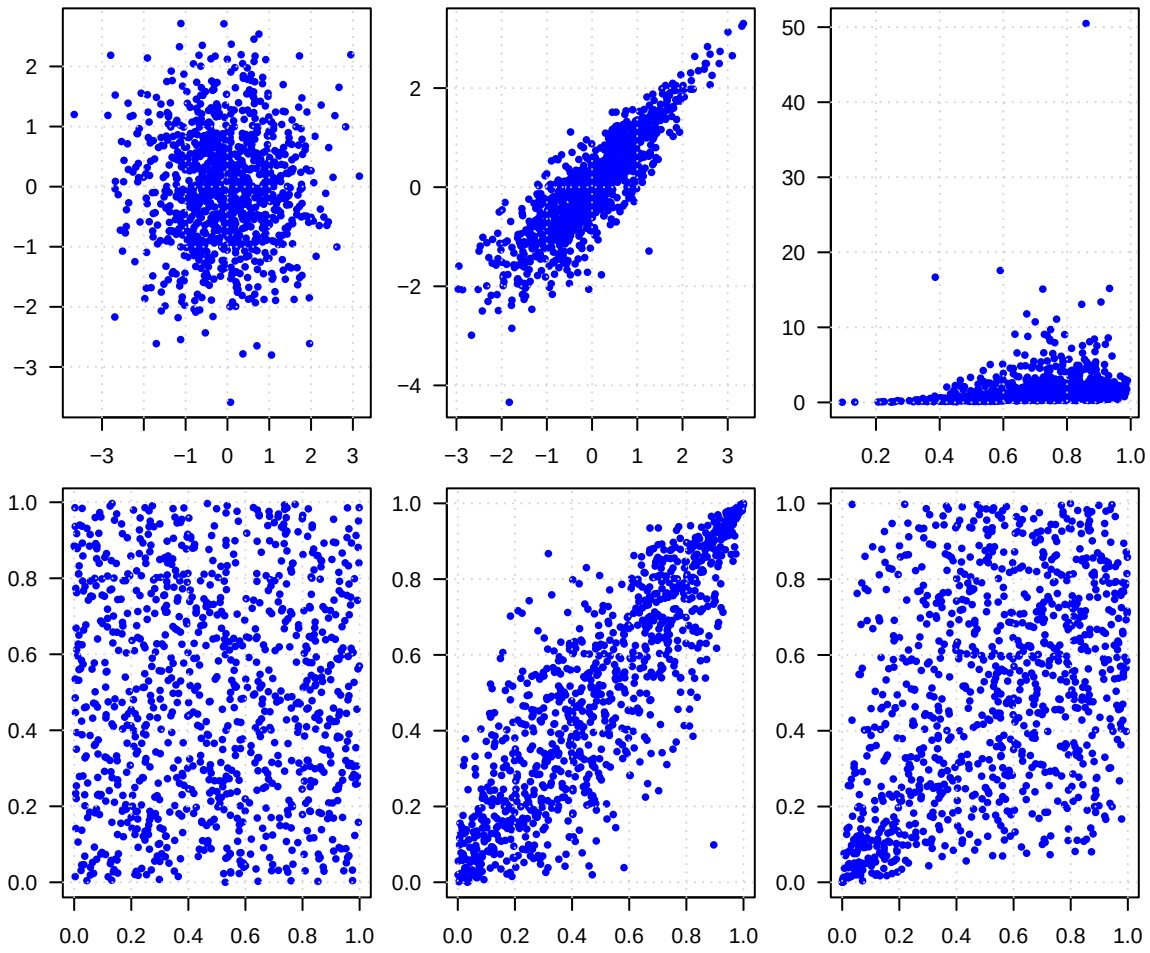


More Examples

- Left: Normals + Gaussian Copula ($\rho = 0$)
- Middle: Normals + Gumbel Copula
- Right: Beta + Lognormal + Clayton Copula

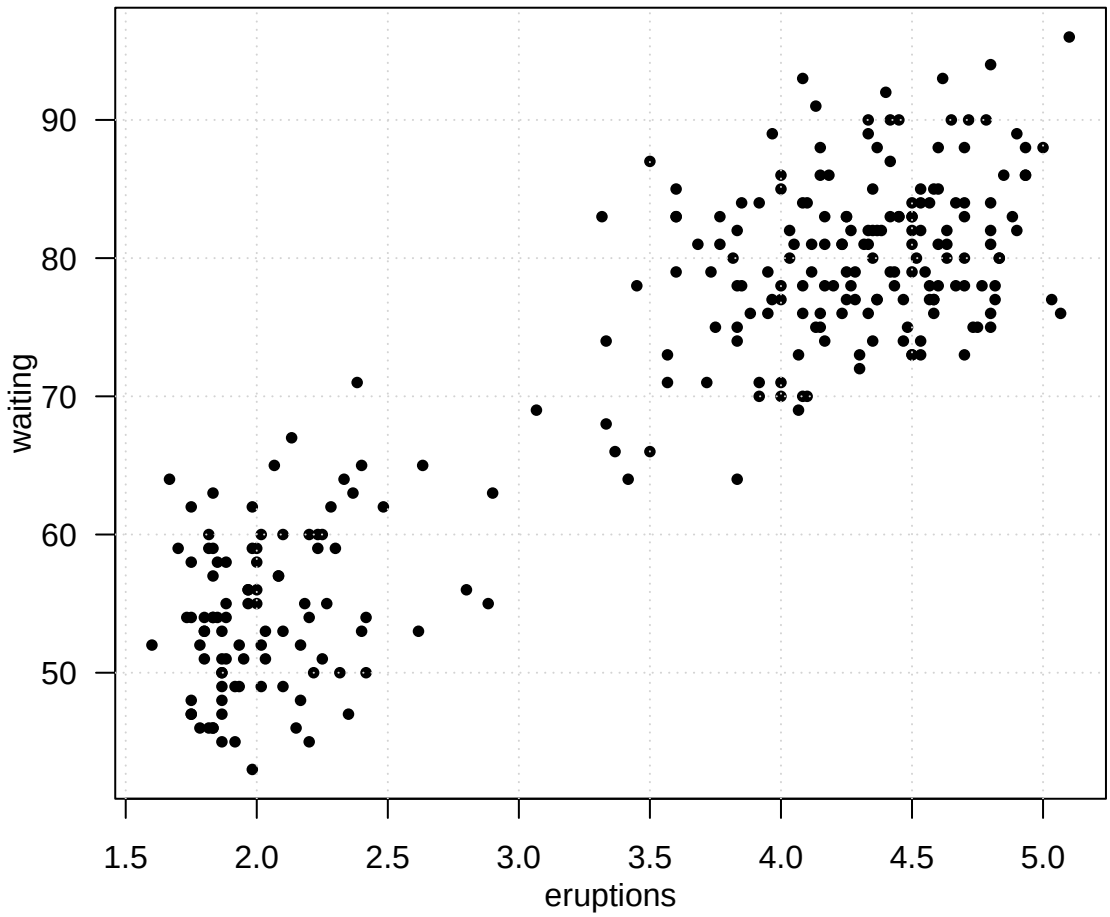
```
library(VineCopula)
par(las = 1, mgp = c(2, 1, 0), mfrow = c(2, 3),
    mar = c(2, 2.4, 0.8, 0.6))
case1 <- cbind(rnorm(1000), rnorm(1000))
plot(case1, pch = 16, col = "blue", cex = 0.8, xlab = "", ylab = "")
grid()

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()
```

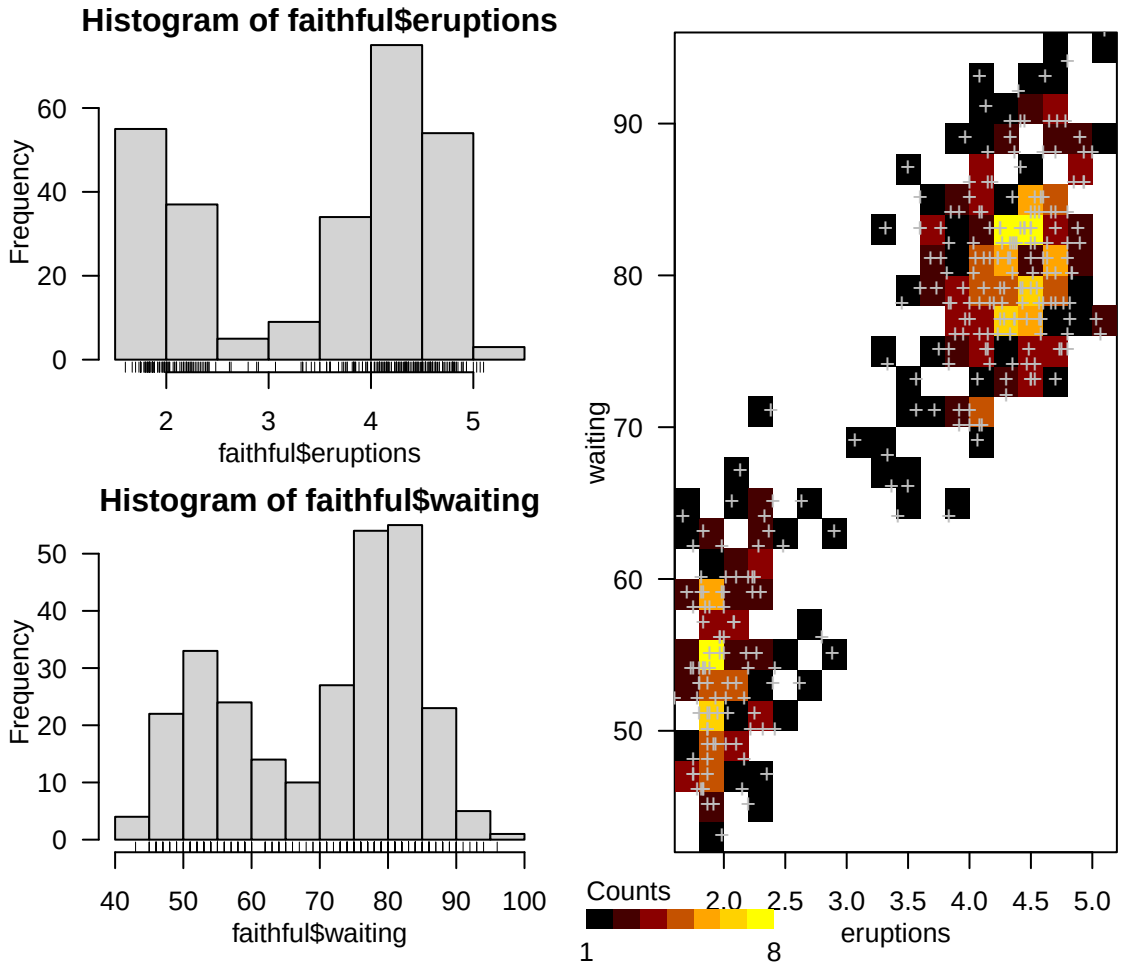


Non-Parametric Density Estimation

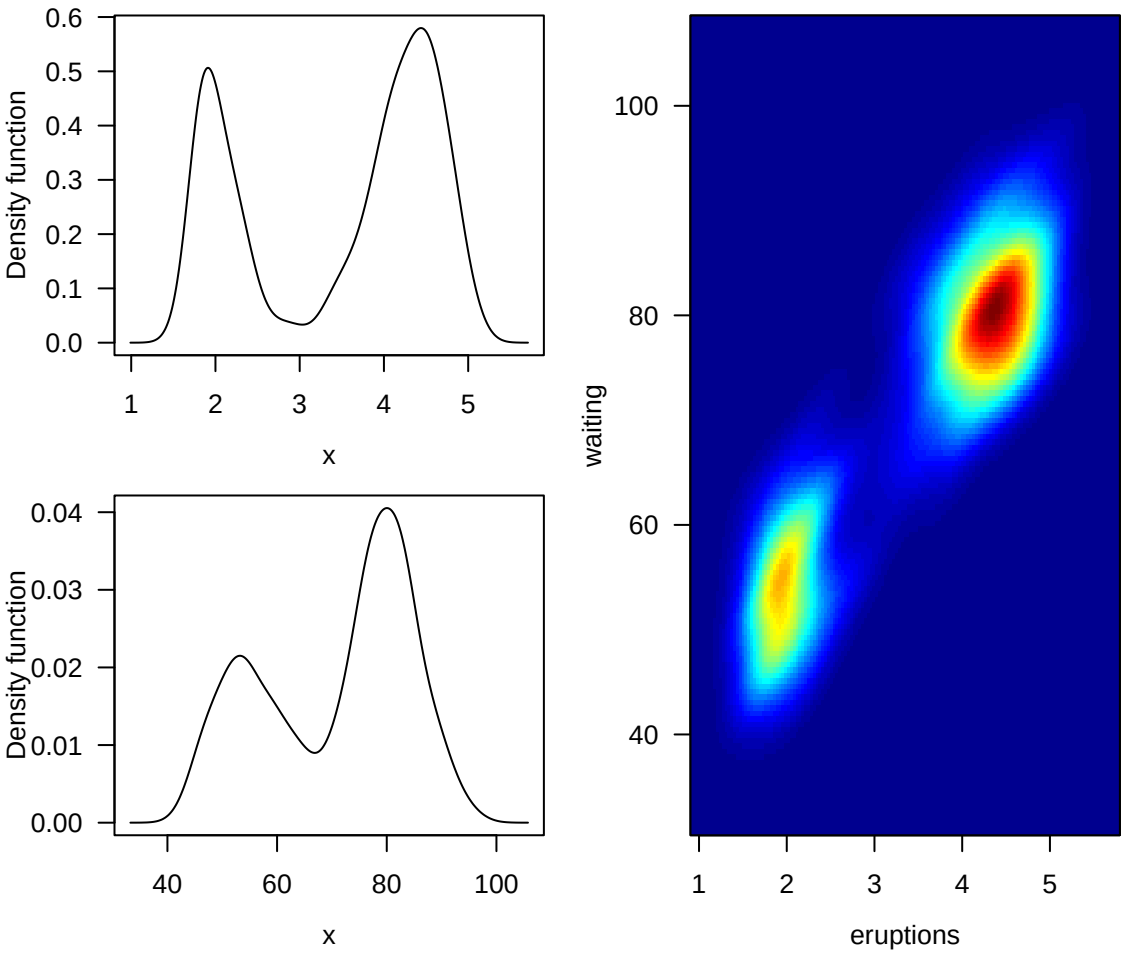
```r
data(faithful)
par(las = 1, mgp = c(2, 1, 0), mar = c(3.6, 3.6, 0.8, 0.6))
plot(faithful, las = 1, cex = 0.8, pch = 16)
grid()
```




```
# Creates the One-Dimensional Histograms and Two-Dimensional Histogram (Heat Map)
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")
```




```
# Creates the Kernel Density Estimates
library(ks)
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$eruptions)) # kde = Kernel Density Estimates
plot(kde(faithful$waiting))
Hpi1 <- Hpi(x = faithful)
kdeHpi1 <- kde(x = faithful, H = Hpi1)
plot(kdeHpi1, display = "image",
     col = tim.colors())
```


Density function
0.6
0.5
0.4
0.3
0.2
0.1
0.0
1
2
3
4
5
x
Density function
0.04
0.03
0.02
0.01
0.00
40
60
80
100
x
waiting
100
80
60
40
1
2
3
4
5
eruptions