```r
library("scmamp")
```

```r
resultados <- read.csv("test_res.csv", header = TRUE, sep = '\t')
tablatst <- cbind(resultados[,1:dim(resultados)[2]])
colnames(tablatst) <- names(resultados)[1:dim(resultados)[2]]
```

```r
colnames(tablatst)
```

```
## [1] "SMOTE" "ROS"   "VAE"   "GAN"
```

```r
tablatst
```

```
##     SMOTE    ROS    VAE    GAN
## 1  0.9105 0.9105 0.9105 0.9182
## 2  0.9836 0.9781 0.9816 0.9881
## 3  0.8943 0.9025 0.9017 0.9070
## 4  1.0000 1.0000 1.0000 0.9998
## 5  0.7793 0.7800 0.8005 0.7951
## 6  0.9362 0.9542 0.9423 0.9436
## 7  0.9380 0.9331 0.9380 0.9380
## 8  0.9984 1.0000 0.9984 0.9969
## 9  0.9649 0.9639 0.9628 0.9618
## 10 0.8723 0.8869 0.8958 0.9039
## 11 0.6188 0.6724 0.6737 0.6709
## 12 0.7507 0.7405 0.7550 0.7556
## 13 0.8944 0.8944 0.8944 0.9925
## 14 0.9965 0.9943 0.9922 0.9936
## 15 0.6199 0.6311 0.6427 0.6583
## 16 1.0000 1.0000 1.0000 1.0000
## 17 0.6959 0.7364 0.7050 0.7208
## 18 0.8396 0.8596 0.8338 0.8425
## 19 1.0000 1.0000 1.0000 1.0000
## 20 0.8031 0.8252 0.8031 0.8092
## 21 0.9944 0.9914 0.9931 0.9955
## 22 0.8874 0.9030 0.9183 0.8755
## 23 0.9290 0.9460 0.9494 0.9472
## 24 0.7558 0.7271 0.7819 0.7819
## 25 0.9510 0.9611 0.9657 0.9487
## 26 0.9732 0.9716 0.9712 0.9662
## 27 0.8559 0.8281 0.8383 0.8657
```

```r
#Test de Fligner-Killeen para la homocedasticidad (no paramétrico, mediana, no normalidad)
fligner.test(x = list(tablatst$SMOTE, tablatst$ROS, tablatst$VAE, tablatst$GAN))
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  list(tablatst$SMOTE, tablatst$ROS, tablatst$VAE, tablatst$GAN)
## Fligner-Killeen:med chi-squared = 0.55363, df = 3, p-value = 0.907
```

```r
#Aplicación del test de Friedman Aligned Ranks
test_friedman_aligned_ranks <- friedmanAlignedRanksTest(as.matrix(tablatst))
test_friedman_aligned_ranks
```

```
##
##  Friedman's Aligned Rank Test for Multiple Comparisons
##
```

```
## data:  as.matrix(tablatst)
## T = 8.922, df = 3, p-value = 0.03035
```

```
#Aplicación del test de Friedman Aligned Ranks post-hoc
test_friedman_aligned_ranks_post <- friedmanAlignedRanksPost(as.matrix(tablatst))
test_friedman_aligned_ranks_post
```

```
##             SMOTE        ROS        VAE         GAN
## SMOTE          NA 0.07002071 0.02917723 0.001036912
## ROS   0.070020706         NA 0.71189891 0.141958025
## VAE   0.029177226 0.71189891         NA 0.271667133
## GAN   0.001036912 0.14195802 0.27166713          NA
```

```
#Bergmann and Hommel dynamic correction of p-values
adjustBergmannHommel (test_friedman_aligned_ranks_post)
```

```
##             SMOTE       ROS        VAE         GAN
## SMOTE          NA 0.1400414 0.08753168 0.006221472
## ROS   0.140041412        NA 0.71189891 0.425874074
## VAE   0.087531677 0.7118989         NA 0.425874074
## GAN   0.006221472 0.4258741 0.42587407          NA
```

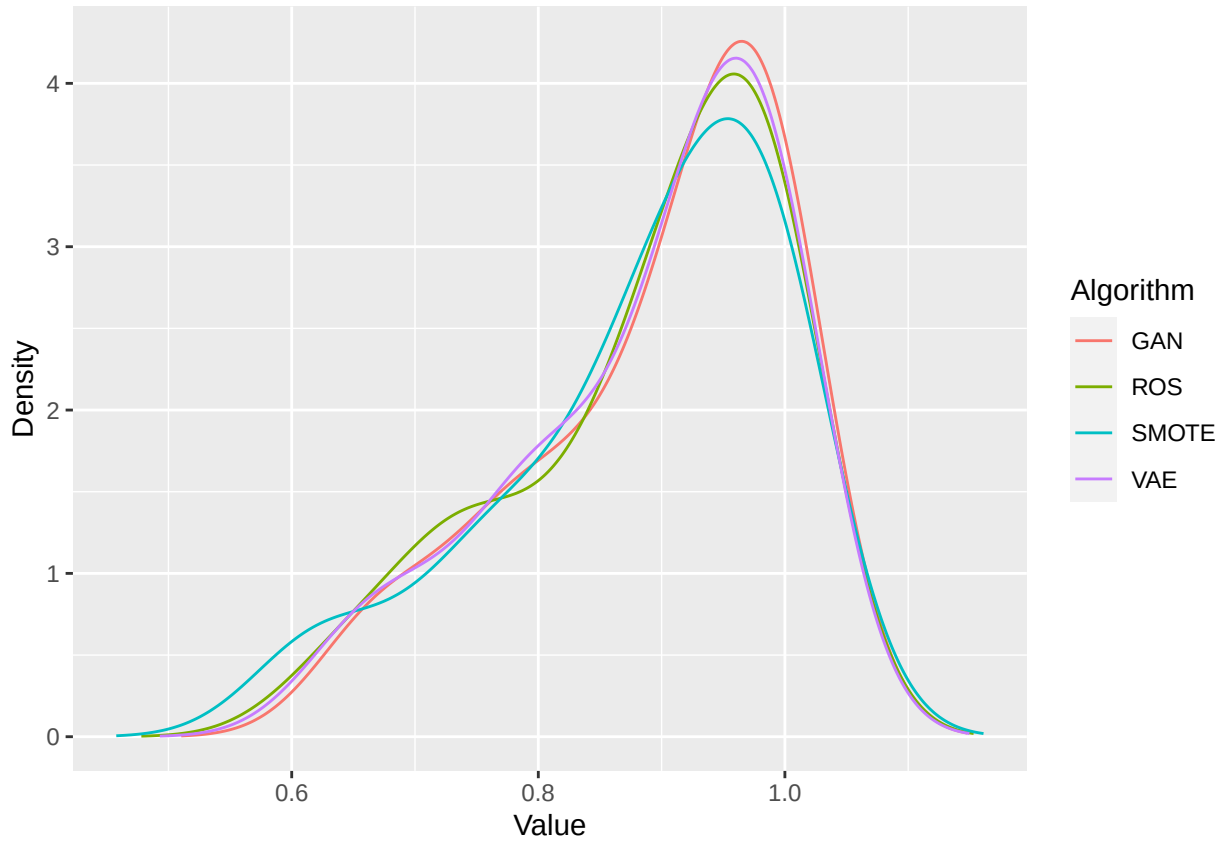
```
quadeTest(as.matrix(tablatst))
```

```
##
##  Quade for Multiple Comparisons
##
## data:  as.matrix(tablatst)
## T = 3.385, df = 3, p-value = 0.02223
```

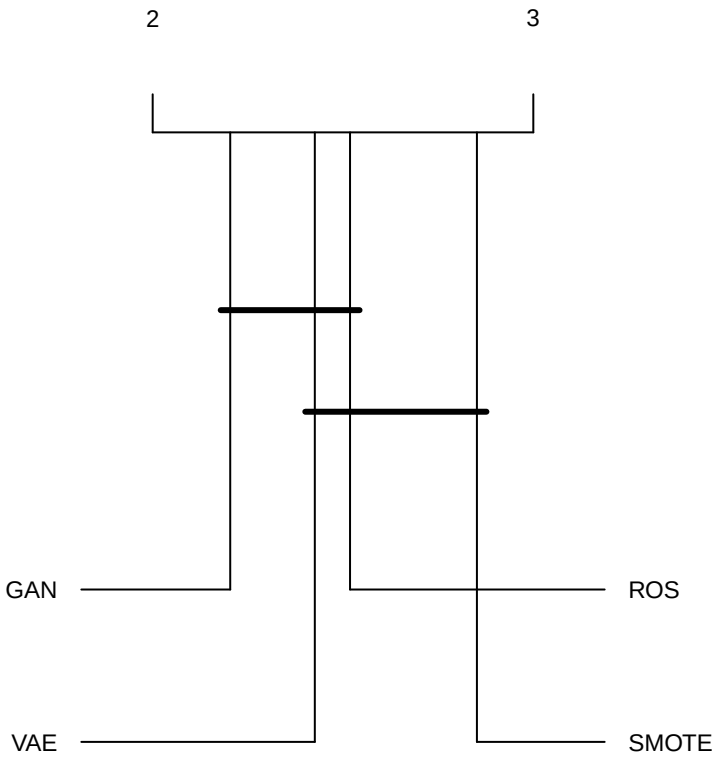
```
quadePost(as.matrix(tablatst))
```

```
##            SMOTE       ROS        VAE        GAN
## SMOTE         NA 0.1931202 0.07406026 0.01651534
## ROS   0.19312018        NA 0.62779210 0.27311412
## VAE   0.07406026 0.6277921         NA 0.54114502
## GAN   0.01651534 0.2731141 0.54114502         NA
```

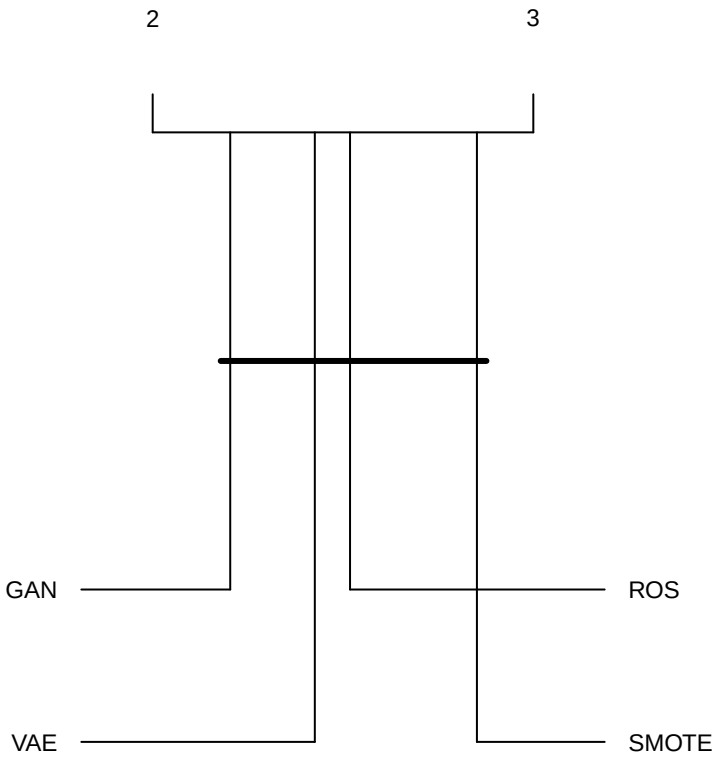
```
plotDensities(as.matrix(tablatst))
```




```
test <- postHocTest(as.matrix(tablatst), test="aligned ranks", correct="bergmann", use.rank=TRUE)
plotRanking(pvalues=test$corrected.pval, summary=test$summary, alpha=0.05)
```




```
test <- postHocTest(as.matrix(tablatst), test="quade", correct="bergmann", use.rank=TRUE)
plotRanking(pvalues=test$corrected.pval, summary=test$summary, alpha=0.05)
```




```
##TABLA NORMALIZADA - smote (other) vs vae (ref) para WILCOXON
# + 0.1 porque wilcox R falla para valores == 0 en la tabla

difs <- (tablatst[,1] - tablatst[,3]) / tablatst[,1]
wilc_1_2 <- cbind(ifelse (difs<0, abs(difs)+0.1, 0+0.1), ifelse (difs>0,    abs(difs)+0.1, 0+0.1))
colnames(wilc_1_2) <- c(colnames(tablatst)[1], colnames(tablatst)[3])
head(wilc_1_2)
```

```
##          SMOTE       VAE
## [1,] 0.1000000 0.1000000
## [2,] 0.1000000 0.1020333
## [3,] 0.1082746 0.1000000
## [4,] 0.1000000 0.1000000
## [5,] 0.1272039 0.1000000
## [6,] 0.1065157 0.1000000
```

```
#Aplicación del test de WILCOXON
SMvsVAEtst <- wilcox.test(wilc_1_2[,1], wilc_1_2[,2], alternative = "two.sided", paired=TRUE)
```

```
## Warning in wilcox.test.default(wilc_1_2[, 1], wilc_1_2[, 2], alternative =
## "two.sided", : cannot compute exact p-value with zeroes
```

```
Rmas <- SMvsVAEtst$statistic
pvalue <- SMvsVAEtst$p.value
SMvsVAEtst <- wilcox.test(wilc_1_2[,2], wilc_1_2[,1], alternative = "two.sided", paired=TRUE)
```

```
## Warning in wilcox.test.default(wilc_1_2[, 2], wilc_1_2[, 1], alternative =
## "two.sided", : cannot compute exact p-value with zeroes
```

```
Rmenos <- SMvsVAEtst$statistic
Rmas
```

```
##   V
## 155
```

```
Rmenos
```

```
##  V
## 35
```

```
pvalue
```

```
## [1] 0.01664714
```

```
##TABLA NORMALIZADA - smote (other) vs gan (ref) para WILCOXON
# + 0.1 porque wilcox R falla para valores == 0 en la tabla

difs <- (tablatst[,1] - tablatst[,4]) / tablatst[,1]
wilc_1_2 <- cbind(ifelse (difs<0, abs(difs)+0.1, 0+0.1), ifelse (difs>0,    abs(difs)+0.1, 0+0.1))
colnames(wilc_1_2) <- c(colnames(tablatst)[1], colnames(tablatst)[4])
head(wilc_1_2)
```

```
##          SMOTE    GAN
## [1,] 0.1084569 0.1000
## [2,] 0.1045750 0.1000
## [3,] 0.1142011 0.1000
## [4,] 0.1000000 0.1002
## [5,] 0.1202746 0.1000
## [6,] 0.1079043 0.1000
```

```
#Aplicación del test de WILCOXON
SMvsGANtst <- wilcox.test(wilc_1_2[,1], wilc_1_2[,2], alternative = "two.sided", paired=TRUE)
```

```
## Warning in wilcox.test.default(wilc_1_2[, 1], wilc_1_2[, 2], alternative =
## "two.sided", : cannot compute exact p-value with zeroes
```

```
Rmas <- SMvsGANtst$statistic
pvalue <- SMvsGANtst$p.value
SMvsGANtst <- wilcox.test(wilc_1_2[,2], wilc_1_2[,1], alternative = "two.sided", paired=TRUE)
```

```
## Warning in wilcox.test.default(wilc_1_2[, 2], wilc_1_2[, 1], alternative =
## "two.sided", : cannot compute exact p-value with zeroes
```

```
Rmenos <- SMvsGANtst$statistic
Rmas
```

```
##   V
## 256
```

```
Rmenos
```

```
##  V
## 44
```

```
pvalue
```

```
## [1] 0.002575851
```

```
##TABLA NORMALIZADA - vae (other) vs gan (ref) para WILCOXON
# + 0.1 porque wilcox R falla para valores == 0 en la tabla
```

```r
difs <- (tablatst[,3] - tablatst[,4]) / tablatst[,3]
wilc_1_2 <- cbind(ifelse (difs<0, abs(difs)+0.1, 0+0.1), ifelse (difs>0,    abs(difs)+0.1, 0+0.1))
colnames(wilc_1_2) <- c(colnames(tablatst)[3], colnames(tablatst)[4])
head(wilc_1_2)
```

```
##            VAE       GAN
## [1,] 0.1084569 0.1000000
## [2,] 0.1066218 0.1000000
## [3,] 0.1058778 0.1000000
## [4,] 0.1000000 0.1002000
## [5,] 0.1000000 0.1067458
## [6,] 0.1013796 0.1000000
```

```r
#Aplicación del test de WILCOXON
VAEvsGANtst <- wilcox.test(wilc_1_2[,1], wilc_1_2[,2], alternative = "two.sided", paired=TRUE)
```

```
## Warning in wilcox.test.default(wilc_1_2[, 1], wilc_1_2[, 2], alternative =
## "two.sided", : cannot compute exact p-value with zeroes
```

```r
Rmas <- VAEvsGANtst$statistic
pvalue <- VAEvsGANtst$p.value
VAEvsGANtst <- wilcox.test(wilc_1_2[,2], wilc_1_2[,1], alternative = "two.sided", paired=TRUE)
```

```
## Warning in wilcox.test.default(wilc_1_2[, 2], wilc_1_2[, 1], alternative =
## "two.sided", : cannot compute exact p-value with zeroes
```

```r
Rmenos <- VAEvsGANtst$statistic
Rmas
```

```
##   V
## 187
```

```r
Rmenos
```

```
##  V
## 89
```

```r
pvalue
```

```
## [1] 0.1401789
```