

# kpcalg: R Package tutorial

Petras Verbyla

November 15, 2024

## 1 Kernel Independence Test

Kernel PC algorithm is heavily based on the Independence Criteria. We use Hilbert-Schmidt Independence Criterion (*HSIC*) and Distance Covariance Criterion (*DCC*). We use these criteria to test the  $H_0 : x \perp y$  ( $x$  independent of  $y$ ) vs  $H_1 : x \not\perp y$  ( $x$  is not independent of  $y$ ). Under the null hypothesis  $H_0$  both *HSIC* and *DCC*  $\approx 0$ . We can find a probability of observing data as extreme as ours (the p-value) under the null hypothesis in two ways:

1. Permutation test
2. Gamma test

Permutation test permutes the  $y$  values to get  $y_{(i)}$  and recalculates the  $HSIC_{(i)}$  or  $DCC_{(i)}$  and compares it with the original *HSIC* or *DCC* respectively. Gamma test approximates the *HSIC* or *DCC* distribution under the null hypothesis  $H_0$  by gamma function.

*DCC* permutation test is implemented in the package "energy".

Here we present a small example of independent ( $x$  and  $y$ ) and dependent ( $z$  and  $w$ ) variables. We apply *HSIC* and *DCC* based tests to estimate the dependency between them:

```
library(energy)
library(kpcalg)
set.seed(10)
#independence
x <- runif(300)
y <- runif(300)

hsic.gamma(x,y)$p.value
## [1] 0.4950817

hsic.perm(x,y)$p.value
## [1] 0.5247525

dcov.gamma(x,y)$p.value
## [1] 0.7095448

dcov.test(x,y,R=100)$p.value
## [1] 0.7425743

#uncorelated but not dependent
z <- 10*(runif(300)-0.5)
w <- z^2 + 10*runif(300)

cor(z,w)
## [1] -0.07832659
```

```

hsic.gamma(z,w)$p.value
## [1] 0

hsic.perm(z,w)$p.value
## [1] 0.00990099

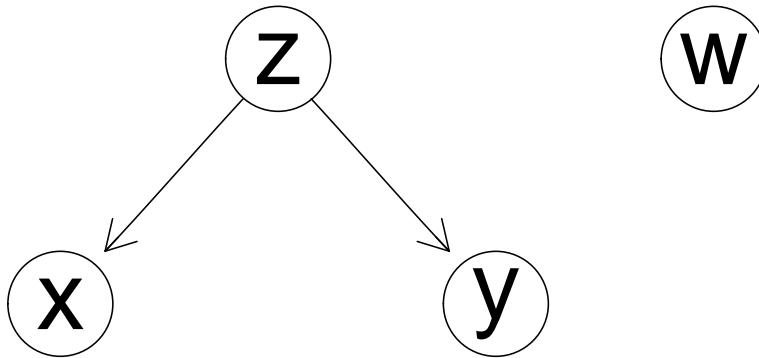
dcov.gamma(z,w)$p.value
## [1] 0

dcov.test(z,w,R=100)$p.value
## [1] 0.00990099

```

## 2 Kernel Conditional Independence Test

We consider a simple model with non-linear relationships and non-gaussian noise;  $x = \sin(z) + U(0, 1)$  and  $y = \cos(z) + U(0, 1)$



First we generate data from the model above. We generate 300 sample points.

```

set.seed(10)
z <- 10*runif(300)
w <- 10*runif(300)
x <- sin(z) + runif(300)
y <- cos(z) + runif(300)
data <- cbind(x,y,z,w)

```

Now we test  $H_0 : x \perp y|z$  (x independent of y given z) vs  $H_1 : x \not\perp y|z$  (x is not independent of y given z).  $H_0$  is true, so we should get high p-values:

```

library(pcalg)
library(kpcalg)
#conditionally independent
test1a <- kernelCItest(x=1,y=2,S=c(3),suffStat = list(data=data,ic.method="dcc.gamma"))
test2a <- kernelCItest(x=1,y=2,S=c(3),suffStat = list(data=data,ic.method="dcc.perm"))
test3a <- kernelCItest(x=1,y=2,S=c(3),suffStat = list(data=data,ic.method="hsic.gamma"))
test4a <- kernelCItest(x=1,y=2,S=c(3),suffStat = list(data=data,ic.method="hsic.perm"))
test5a <- kernelCItest(x=1,y=2,S=c(3),suffStat = list(data=data,ic.method="hsic.clust"))
test6a <- gaussCItest(x=1,y=2,S=c(3),suffStat = list(C=cor(data),n=4))

cat("DCC (gamma test): \t\t\t",test1a,
    "\nDCC (permutation test): \t\t\t",test2a,

```

```

"\nHSIC-residuals (gamma test): \t\t",test3a,
"\nHSIC-residuals (permutation test): \t",test4a,
"\nHSIC-cluster: \t\t\t\t",test5a,
"\nFisher's Z test: \t\t\t",test6a)

## DCC (gamma test):      0.6455592
## DCC (permutation test):  0.5643564
## HSIC-residuals (gamma test):  0.4868527
## HSIC-residuals (permutation test):  0.4950495
## HSIC-cluster:      0.3168317
## Fisher's Z test:      1

```

All five tests find independence.

Now we test  $H_0 : x \perp y|w$  (x independent of y given w) vs  $H_1 : x \not\perp y|w$  (x is not independent of y given w).  $H_0$  is not true, so we should get very low p-values:

```

#dependent
test1b <- kernelCIttest(x=1,y=2,S=c(4),suffStat = list(data=data,ic.method="dcc.gamma"))
test2b <- kernelCIttest(x=1,y=2,S=c(4),suffStat = list(data=data,ic.method="dcc.perm"))
test3b <- kernelCIttest(x=1,y=2,S=c(4),suffStat = list(data=data,ic.method="hsic.gamma"))
test4b <- kernelCIttest(x=1,y=2,S=c(4),suffStat = list(data=data,ic.method="hsic.perm"))
test5b <- kernelCIttest(x=1,y=2,S=c(4),suffStat = list(data=data,ic.method="hsic.clust"))
test6b <- gaussCIttest( x=1,y=2,S=c(4),suffStat = list(C=cor(data),n=4))

cat("DCC (gamma test): \t\t\t",test1b,
    "\nDCC (permutation test): \t\t\t",test2b,
    "\nHSIC-residuals (gamma test): \t\t\t",test3b,
    "\nHSIC-residuals (permutation test): \t\t\t",test4b,
    "\nHSIC-cluster: \t\t\t\t\t",test5b,
    "\nFisher's Z test: \t\t\t\t\t",test6b)

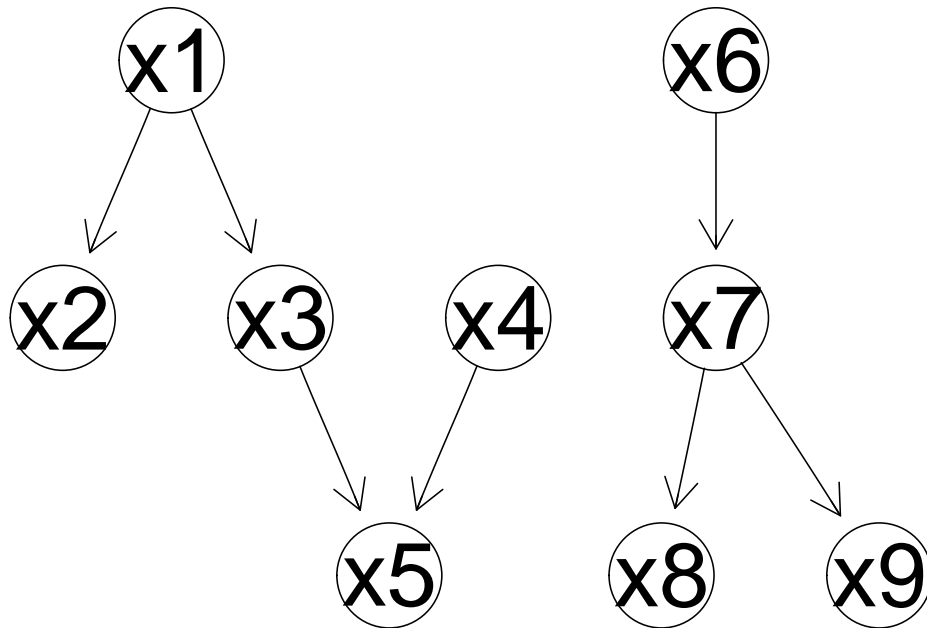
## DCC (gamma test):      0.03469079
## DCC (permutation test):  0.1188119
## HSIC-residuals (gamma test):  0.000292712
## HSIC-residuals (permutation test):  0.00990099
## HSIC-cluster:      0.00990099
## Fisher's Z test:      1

```

As expected the HSIC and DCC based independence tests reject  $H_0$ , while the Fisher's Z test does not.

### 3 Kernel PC algorithm

This establishes the effectiveness of the kernel and distance based independence tests for testing conditional dependence in data with non-linear relationships and non-gaussian noise. Now we will test the Kernel PC algorithm on a toy network on 9 nodes.



First we generate data from this DAG. We again generate 300 data points.

```

set.seed(4)
n <- 300
data <- NULL
x1 <- 2*(runif(n)-0.5)
x2 <- x1 + runif(n)-0.5
x3 <- x1^2 + 0.6*runif(n)
x4 <- rnorm(n)
x5 <- x3 + x4^2 + 2*runif(n)
x6 <- 10*(runif(n)-0.5)
x7 <- x6^2 + 5*runif(n)
x8 <- 2*x7^2 + 1.5*rnorm(n)
x9 <- x7 + 4*runif(n)
data <- cbind(x1,x2,x3,x4,x5,x6,x7,x8,x9)

```

Now we apply PC, Kernel PC residuals (with gamma and permutation independence tests), Kernel PC cluster and Distance PC algorithms to infer the underlying network.

```

library(pcalg)
library(kpcalg)
pc <- pc(suffStat = list(C = cor(data), n = 9),
        indepTest = gaussCItest,
        alpha = 0.9,
        labels = colnames(data),
        u2pd = "relaxed",
        skel.method = "stable",
        verbose = F)
kpc1 <- kpc(suffStat = list(data=data, ic.method="dcc.perm"),
          indepTest = kernelCItest,
          alpha = 0.1,
          labels = colnames(data),
          u2pd = "relaxed",
          skel.method = "stable",
          verbose = F)

```

```

kpc2 <- kpc(suffStat = list(data=data, ic.method="hsic.gamma"),
            indepTest = kernelCIttest,
            alpha = 0.1,
            labels = colnames(data),
            u2pd = "relaxed",
            skel.method = "stable",
            verbose = F)
kpc3 <- kpc(suffStat = list(data=data, ic.method="hsic.perm"),
            indepTest = kernelCIttest,
            alpha = 0.1,
            labels = colnames(data),
            u2pd = "relaxed",
            skel.method = "stable",
            verbose = F)
kpc4 <- kpc(suffStat = list(data=data, ic.method="hsic.clust"),
            indepTest = kernelCIttest,
            alpha = 0.1,
            labels = colnames(data),
            u2pd = "relaxed",
            skel.method = "stable",
            verbose = F)

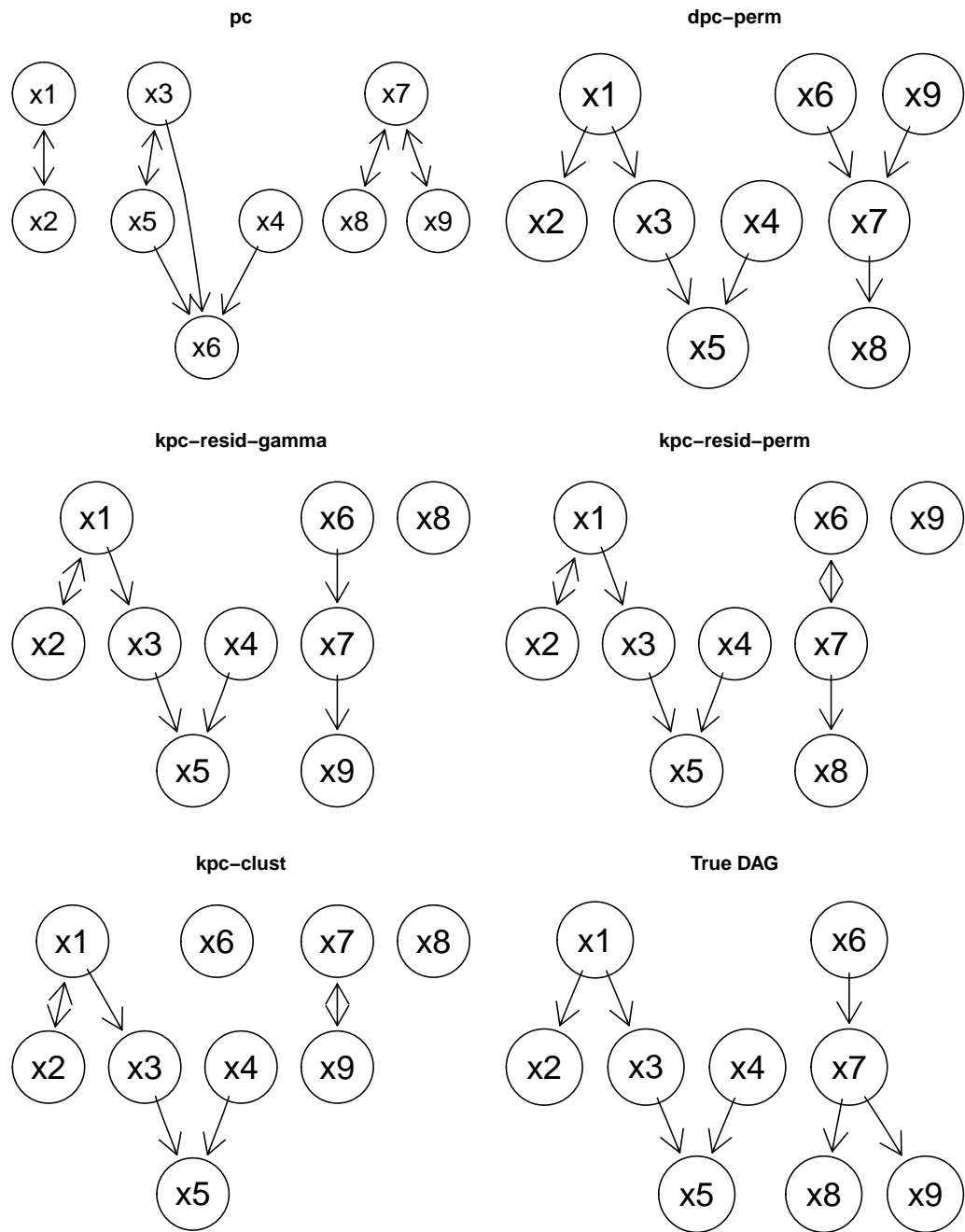
```

We plot all five outputs as well as the true underlying graph structure:

```

par(mfrow=c(3,2))
plot(pc@graph, attrs=list(node=list(fontsize=5)), main="pc")
plot(kpc1@graph, attrs=list(node=list(fontsize=5)), main="dpc-perm")
plot(kpc2@graph, attrs=list(node=list(fontsize=5)), main="kpc-resid-gamma")
plot(kpc3@graph, attrs=list(node=list(fontsize=5)), main="kpc-resid-perm")
plot(kpc4@graph, attrs=list(node=list(fontsize=5)), main="kpc-clust")
plot(as(true, "graphNEL"), attrs=list(node=list(fontsize=5)), main="True DAG")

```



We observe that all independence based PC versions significantly outperform the original PC algorithm.