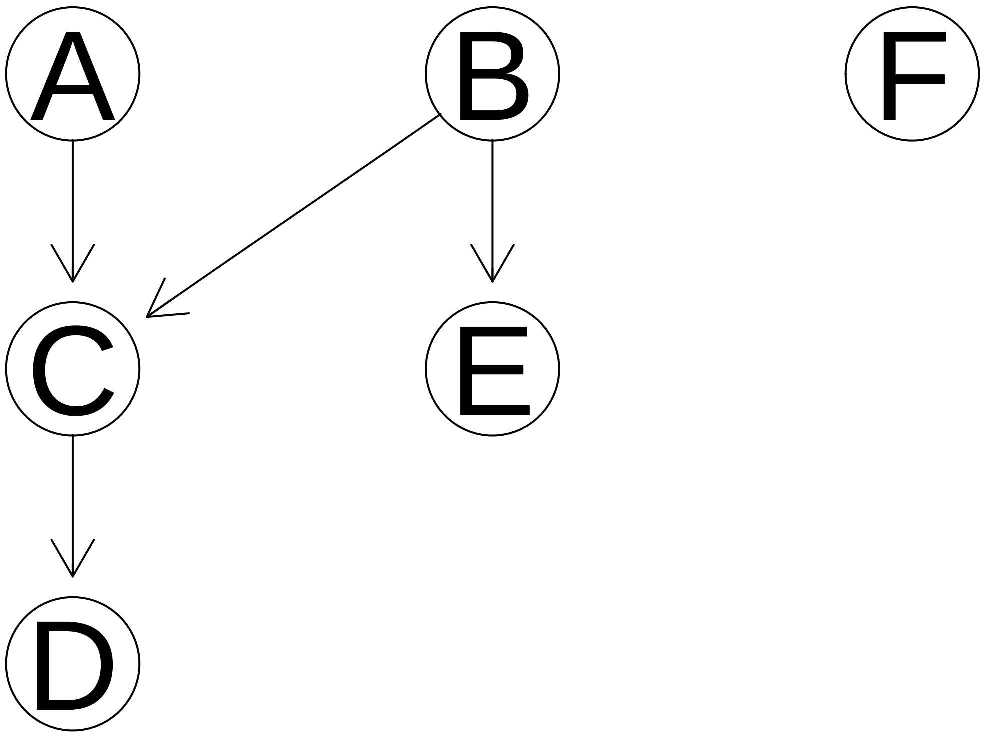
Structure learning of Bayesian networks with the PC (Peter–Clark) algorithm

Modestas Filipavicius

May 3, 2018

We're given a partially directed acyclic graph (pDAG), with 6 random variables:

```r
partial_dag = new("graphNEL", nodes=c("A", "B", "C", "D", "E", "F"), edgemode="directed")
partial_dag = addEdge("A", "C", partial_dag, 1)
partial_dag = addEdge("C", "D", partial_dag, 1)
partial_dag = addEdge("B", "C", partial_dag, 1)
partial_dag = addEdge("B", "E", partial_dag, 1)
plot(partial_dag)
```

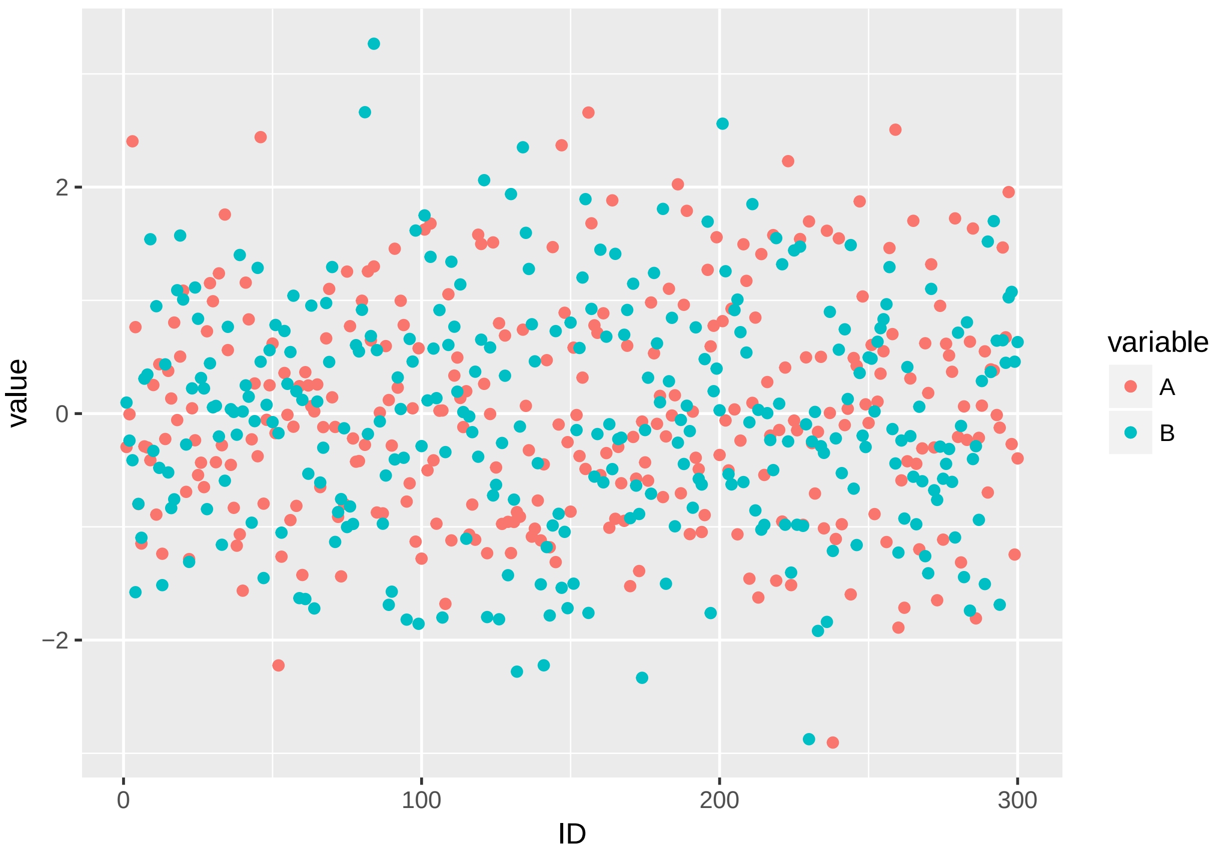


Within a Bayes Net with 6 random variables, are two variables, A and B, causally related?

We will run the PC algorithm to learn correct graph structure and causal relationships.

Plot the A and B observations:

```r
data = readRDS("MVN_DAG.rds")
data$ID = seq.int(nrow(data))

library(ggplot2)
ggplot(data, aes(ID, y = value, color = variable)) +
    geom_point(aes(y = A, col = "A")) +
    geom_point(aes(y = B, col = "B"))
```



What does the plot suggest about their (marginal) correlation?

```
# calculate Pearson Corerrelation Coef (PCC) between A and B
cor_AB = cor.test(data[, 1], data[, 2])
cor_AB$p.value
```

```
## [1] 0.840103
```

```
# Null hypothesis, that A and B are not related, holds with high confidence.

# Just for fun, test for correlations between A and C, and B and C
cor_AC = cor.test(data[, 1], data[, 3])
cor_BC = cor.test(data[, 2], data[, 3])
cor_AC$p.value
```

```
## [1] 2.261867e-18
```

```
cor_BC$p.value
```

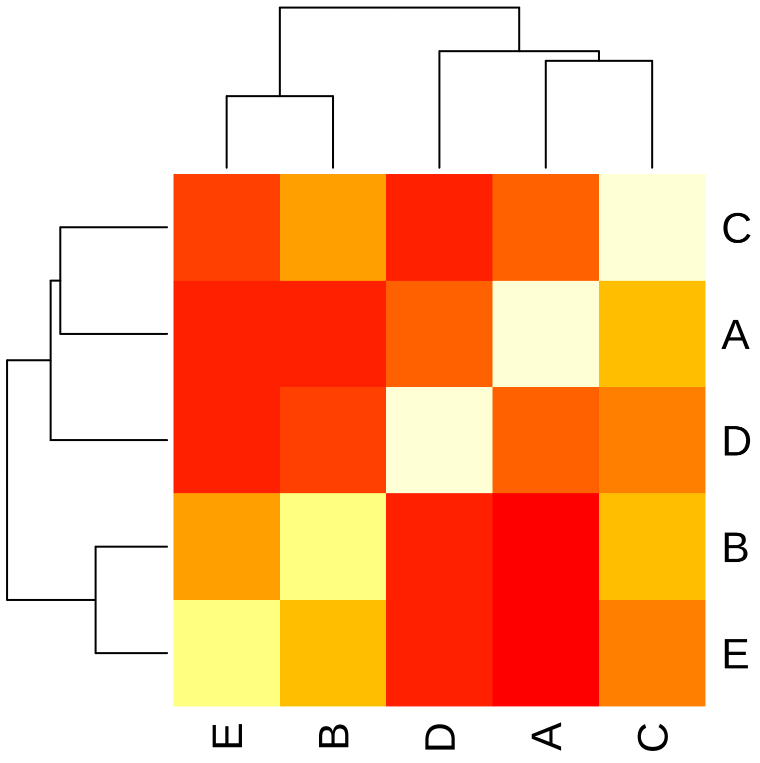
```
## [1] 6.901592e-32
```

```
# these p-values are extremely small, and we can reject
# null hypothesis with high confidence!


# "negative" control: unconnected node F to B
# expect very low correlation
```

```r
cor_BF = cor.test(data[, 2], data[, 6])
cor_BF
```

```
##
##  Pearson's product-moment correlation
##
## data:  data[, 2] and data[, 6]
## t = -1.4007, df = 298, p-value = 0.1624
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.19235247  0.03266679
## sample estimates:
##         cor
## -0.08087322
```

```r
# calculate all pair-wise PCCs
cor_all = round(cor(data[, 1:5]), 2)
heatmap(cor_all, xlab = 'All pair-wise PCCs for 6 nodes')
```

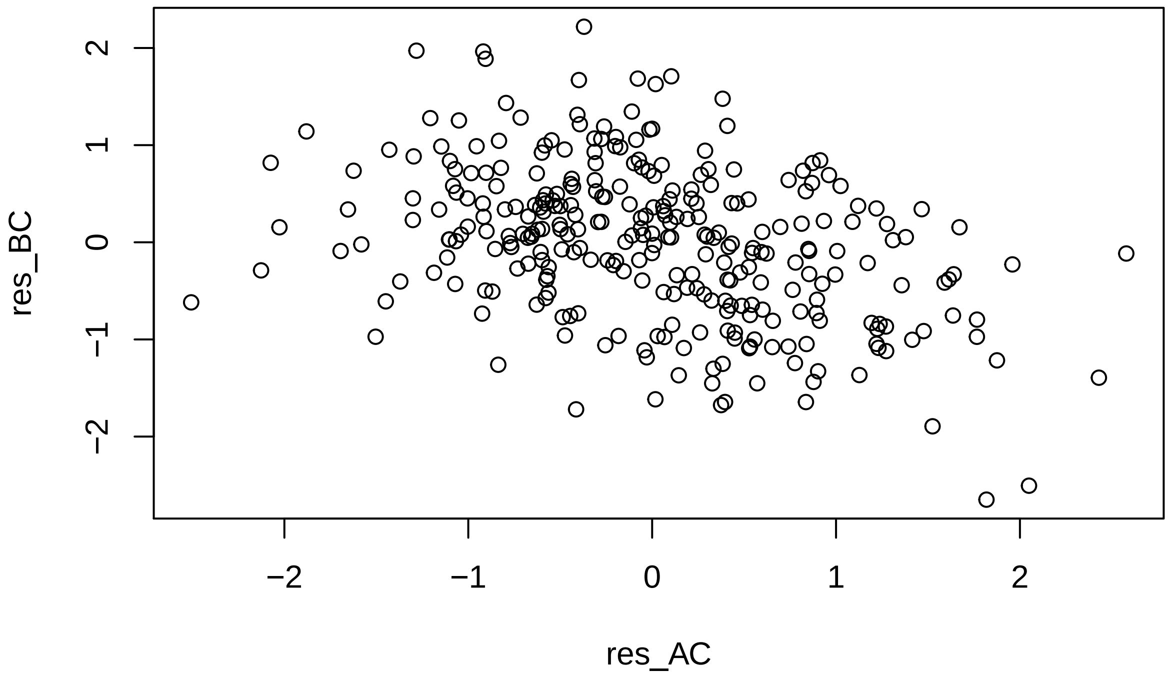


All pair−wise PCCs for 6 nodes

Testing for partial correlation

```r
# Linearly regress A on C (that is, with A as the response variable
# and C as the explanatory variable).
# Compute and store the residuals.
lm_AC = lm(data[, 1] ~ data[, 3], data = data)
```

```r
res_AC = residuals(lm_AC)

# Linearly regress B on C. Compute and store the residuals.
lm_BC = lm(data[, 2] ~ data[, 3], data = data)
res_BC = residuals(lm_BC)

# Plot the residuals of A (regressed on C) against
# the residuals of B (regressed on C). What do you see?
plot(x = res_AC, y = res_BC)
```



In the plot above, I can see a slight negative correlation between residuals test for Pearson correlation between A and B residuals regressed on C.

Let's calculate an exact value of Pearson Correlation:

```r
cor_ABgC = cor.test(res_AC, res_BC)

# PCC is in fact negative:
cor_ABgC
```

```
##
##  Pearson's product-moment correlation
##
## data:  res_AC and res_BC
## t = -7.5173, df = 298, p-value = 6.6e-13
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.4903245 -0.2995546
## sample estimates:
##        cor
## -0.3992521
```

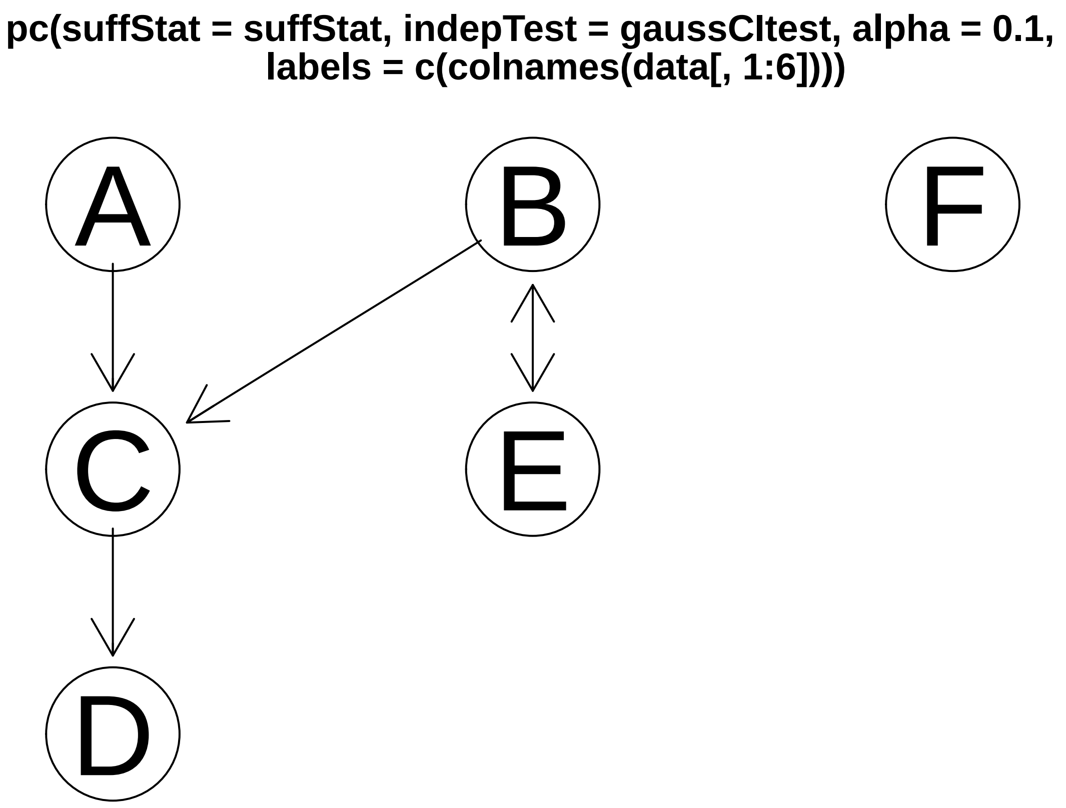
```r
# P-value
cor_ABgC$p.value
```

```
## [1] 6.599955e-13
```

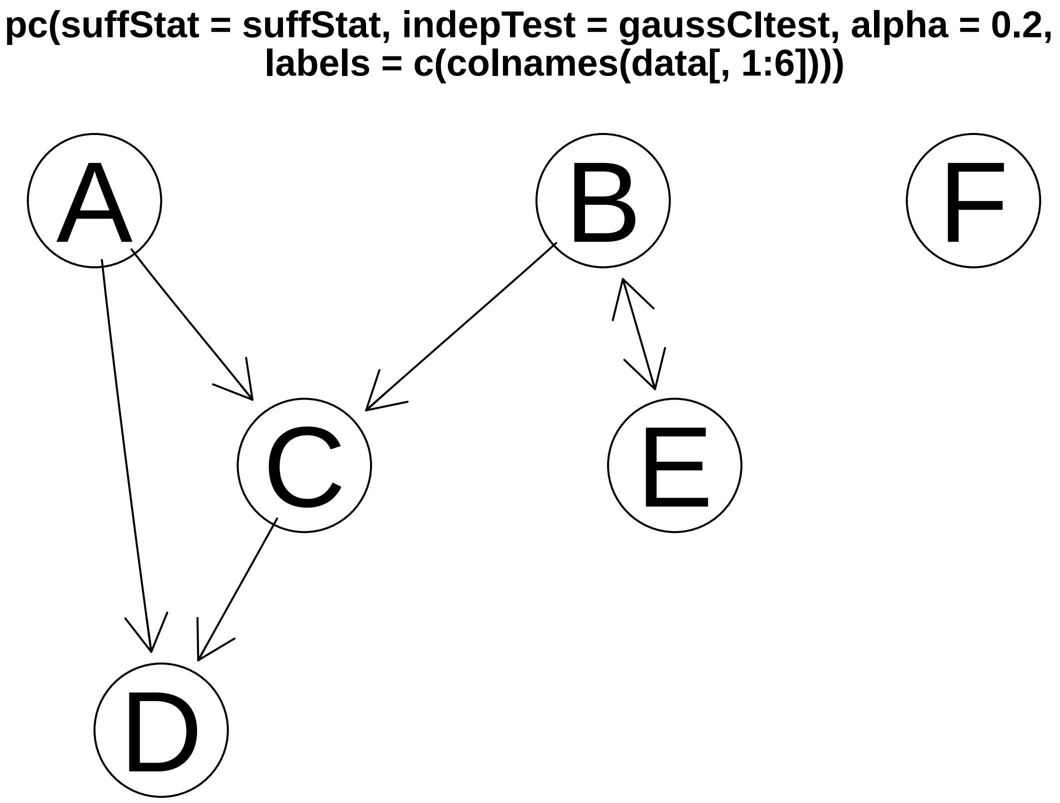
The p-value is extremely small, thus we can reject the null hypothesis, namely, that there's no correlation between residuals, with high confidence.

Running the PC algorithm to learn correct graph structure and causal relationships

```
# install.packages("pcalg")
# source("https://bioconductor.org/biocLite.R")
# biocLite("graph")
# biocLite("RBGL")
# biocLite("Rgraphviz")


library(pcalg)

# calculate covariance matrix
data_cor = cor(data[, 1:6])
data_n = nrow(data)
suffStat = list(C = data_cor, n = data_n)

# alpha: significance level (number in (0, 1) for the individual
# conditional independence tests.
a = pc(alpha = 0.10, suffStat = suffStat, indepTest = gaussCItest, labels = c(colnames(data[, 1:6])))
plot(a)
```

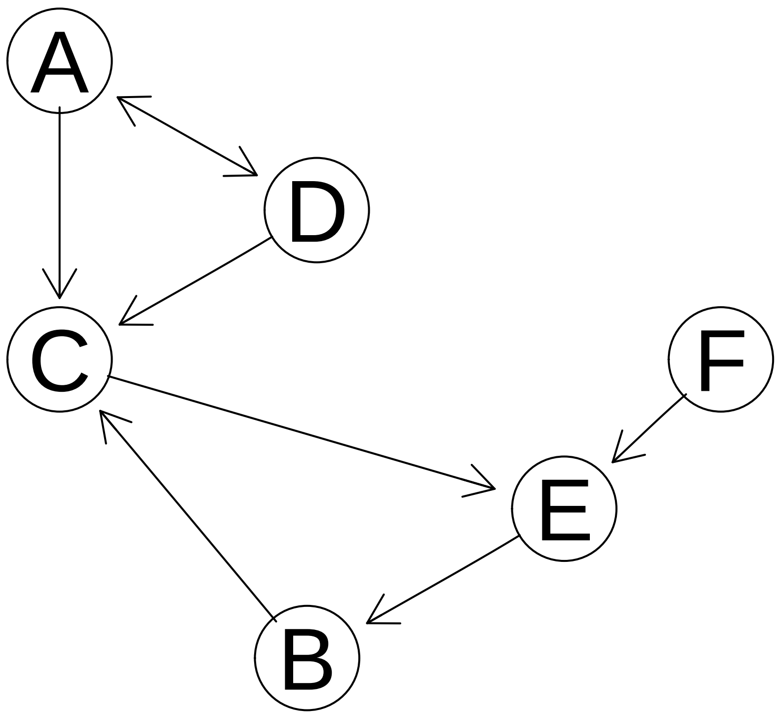

pc(suffStat = suffStat, indepTest = gaussCItest, alpha = 0.1,
labels = c(colnames(data[, 1:6])))
A
B
F
C
E
D

```r
a = pc(alpha = 0.20, suffStat = suffStat, indepTest = gaussCItest, labels = c(colnames(data[, 1:6])))
plot(a)
```


pc(suffStat = suffStat, indepTest = gaussCItest, alpha = 0.2,
labels = c(colnames(data[, 1:6])))
A
B
F
C
E
D

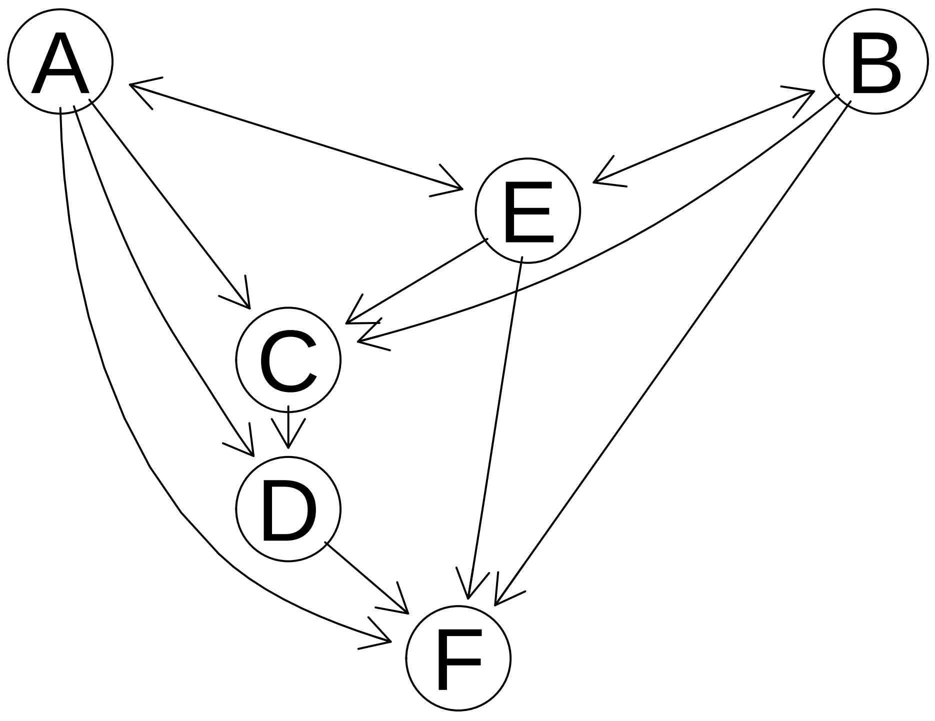
```r
a = pc(alpha = 0.40, suffStat = suffStat, indepTest = gaussCItest, labels = c(colnames(data[, 1:6])))
plot(a)
```

pc(suffStat = suffStat, indepTest = gaussCItest, alpha = 0.4, labels = c(colnames(data[, 1:6])))



```
a = pc(alpha = 0.9, suffStat = suffStat, indepTest = gaussCItest, labels = c(colnames(data[, 1:6])))
plot(a)
```

pc(suffStat = suffStat, indepTest = gaussCItest, alpha = 0.9, labels = c(colnames(data[, 1:6])))



We observe that the closest structure to figure 2 is obtained by setting $alpha = 0.1$. Then the equivalence class joint distribution looks like $p(a, b, c, d) = p(a)p(b)p(c|a, b)p(d|c)$ and is only missing E and F nodes, which are completely disconnected. Increasing $alpha$ to $alpha = 0.2$ connects E and F via undirected edge. As soon as $alpha = 0.25$, every node is connected to at least one other node. However, I was not able to reproduce the figure 2 with any $alpha$ values.

Here alpha param is defined as "significance level (number in (0, 1) for the individual conditional independence", and roughly corresponds to reducing the dependancy threshold needed to connect two graphs. In other words, it's a regularization coefficient that we must optimize for a given dataset.