CS&SS/STAT 566 Class Lab 6

February 12, 2016

Inferring structure and using the PC algorithm

Our mission here is to attempt to find the underlying causal structure.

```
source("http://bioconductor.org/biocLite.R")
```

```
## Bioconductor version 3.2 (BiocInstaller 1.20.1), ?biocLite for help
```

```
biocLite("RBGL")
```

```
## BioC_mirror: https://bioconductor.org

## Using Bioconductor 3.2 (BiocInstaller 1.20.1), R 3.2.3 (2015-12-10).

## Installing package(s) 'RBGL'

##
## The downloaded binary packages are in
##  /var/folders/9v/ty0pj_y578794sx7hjs4pdwm0000gn/T//RtmpEIPcPg/downloaded_packages
```

```
## It may ask you to update some dependent packages. For example,
# Old packages: 'manipulate'
# Update all/some/none? [a/s/n]:

## OR try this if the packages were not updated.
# update.packages(lib.loc = .libPaths()[1], repos="http://cran.fhcrc.org/")

# install.packages("pcalg", repos="http://cran.fhcrc.org/")
library("pcalg")
library("graph")

biocLite("Rgraphviz")  ## seems not to run on many systems
```

```
## BioC_mirror: https://bioconductor.org

## Using Bioconductor 3.2 (BiocInstaller 1.20.1), R 3.2.3 (2015-12-10).

## Installing package(s) 'Rgraphviz'

##
## The downloaded binary packages are in
##  /var/folders/9v/ty0pj_y578794sx7hjs4pdwm0000gn/T//RtmpEIPcPg/downloaded_packages
```

```
library("Rgraphviz")
```

```
## Loading required package: grid
```

```
# print(.packages(lib.loc = .libPaths()[1]))
plotcpdag <- "Rgraphviz" %in% print(.packages(lib.loc = .libPaths()[1]))
```

```
##  [1] "Rgraphviz"     "grid"          "graph"         "pcalg"
##  [5] "BiocInstaller" "knitr"         "stats"         "graphics"
##  [9] "grDevices"     "utils"         "datasets"      "methods"
## [13] "base"
```

```
## You may need to run these as well.
# install.packages(c("abind", "corpcor", "sfsmisc", "robustbase"),
#                  repos="http://cran.fhcrc.org/")

# library(abind)
# library(corpcor)
# library(sfsmisc)
# library(robustbase)

## Load some utility functions (showAmat; showEdgeList)
## (Thanks to Markus Kalisch!)
source("http://www.stat.washington.edu/tsr/s566/labs/pc-output-utilities.r")
```

(1) First dataset

```
#data <- read.table("lab4DAG1.dat")
data <- read.table("http://www.stat.washington.edu/tsr/s566/labs/lab4DAG1.dat")
attr(data,"names")
```

```
## [1] "x" "y" "z" "w"
```

```
x <- data$x
y <- data$y
z <- data$z
w <- data$w

names <- attr(data,"names")
```

Here are some basic tools:

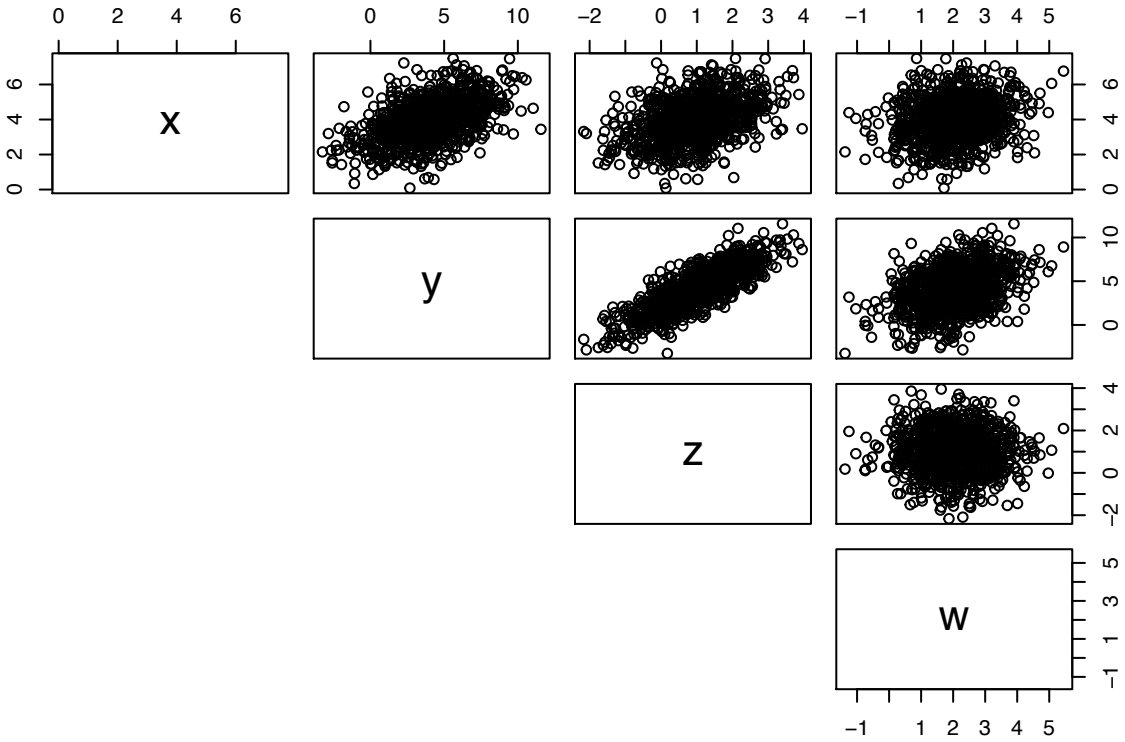
(i) the correlation matrix to check for independence (under normality) of the form

$$X \perp\!\!\!\perp Y.$$

```r
cor(data)
```

```
##       x     y        z        w
## x 1.000 0.507  0.41716  0.19579
## y 0.507 1.000  0.81459  0.40061
## z 0.417 0.815  1.00000 -0.00936
## w 0.196 0.401 -0.00936  1.00000
```

```r
# graphical display
pairs(data, lower.panel = NULL)
```




```r
#if anything looks close to zero we can test for this using:

# cor.test( put-var1-here, put-var2-here )
cor.test(y, z)
```

```
##
##  Pearson's product-moment correlation
##
## data:  y and z
## t = 40, df = 1000, p-value <2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.793 0.834
## sample estimates:
##   cor
## 0.815
```

```
# zeros correspond to marginal independence
```

(ii) To check for independence of the form

$$X \perp\!\!\!\perp Y \mid Z,$$

we can do:

```
tmp <- lm( x ~ y + z)
summary(tmp)$coef
```

```
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept)    2.976     0.0673  44.226 2.64e-237
## y              0.246     0.0233  10.531  1.16e-24
## z              0.016     0.0563   0.284  7.77e-01
```

```
tmp <- lm( x ~ w + z)
summary(tmp)$coef
```

```
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept)    2.995     0.0846   35.39 2.17e-178
## w              0.244     0.0343    7.11  2.18e-12
## z              0.501     0.0336   14.92  1.34e-45
```

```
tmp <- lm( y ~ w + x)
summary(tmp)$coef
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -1.107     0.2322   -4.77 2.17e-06
## w              0.773     0.0642   12.05 2.63e-31
## x              0.898     0.0525   17.11 9.13e-58
```

```
tmp <- lm( x ~ w + y)
summary(tmp)$coef
```

```
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept)   2.9843     0.0812  36.749 1.34e-187
## w            -0.0104     0.0365  -0.286  7.75e-01
## y             0.2529     0.0148  17.114  9.13e-58
```

```
# a coefficient of y close to zero indicates independence
# (again assuming normality!)
```

(iii) likewise to test for independence of the form

$$X \perp\!\!\!\perp Y \mid \{Z, W\},$$

we can do:

```
tmp <- lm( x ~ y + z + w)
summary(tmp)$coef
```

```
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept)  2.98253     0.0823  36.225 5.56e-184
## y            0.24896     0.0329   7.574  8.26e-14
## z            0.00965     0.0726   0.133  8.94e-01
## w           -0.00648     0.0470  -0.138  8.90e-01
```

```
tmp <- lm( z ~ y + x + w)
summary(tmp)$coef
```

```
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept)  0.17801    0.05438   3.274  1.10e-03
## y            0.40368    0.00733  55.046 2.32e-304
## x            0.00184    0.01382   0.133  8.94e-01
## w           -0.40898    0.01591 -25.711 3.19e-112
```

With these basic tools let's try to find the underlying DAG!

Hint: Begin by just looking for marginal independence relations

```
##### Using the PC Algorithm to do the job for us:
n <- nrow(data)
p <- ncol(data)
indepTest <- gaussCItest
suffStat <- list(C=cor(data), n = n)

## estimate CPDAG
alpha <- 0.05
pc.fit <- pc(suffStat, indepTest, p = p, alpha = alpha, verbose = TRUE)
```

```
## Order=0; remaining edges:12
## x= 1  y= 2  S=  : pval = 1.63e-69
## x= 1  y= 3  S=  : pval = 1.06e-44
## x= 1  y= 4  S=  : pval = 3.78e-10
## x= 2  y= 1  S=  : pval = 1.63e-69
## x= 2  y= 3  S=  : pval = 5.3e-284
## x= 2  y= 4  S=  : pval = 6.08e-41
## x= 3  y= 1  S=  : pval = 1.06e-44
## x= 3  y= 2  S=  : pval = 5.3e-284
## x= 3  y= 4  S=  : pval = 0.768
## x= 4  y= 1  S=  : pval = 3.78e-10
## x= 4  y= 2  S=  : pval = 6.08e-41
## Order=1; remaining edges:10
## x= 1  y= 2  S= 3 : pval = 4.68e-25
## x= 1  y= 2  S= 4 : pval = 3.53e-60
## x= 1  y= 3  S= 2 : pval = 0.777
## x= 1  y= 4  S= 2 : pval = 0.775
## x= 2  y= 1  S= 3 : pval = 4.68e-25
## x= 2  y= 1  S= 4 : pval = 3.53e-60
## x= 2  y= 3  S= 1 : pval = 2.32e-227
```

```
## x= 2  y= 3  S= 4 : pval = 0
## x= 2  y= 4  S= 1 : pval = 5.74e-32
## x= 2  y= 4  S= 3 : pval = 8.4e-168
## x= 3  y= 2  S= 1 : pval = 2.32e-227
## x= 4  y= 2  S= 1 : pval = 5.74e-32
## Order=2; remaining edges:6
## x= 2  y= 1  S= 3 4 : pval = 6.42e-14
## x= 2  y= 3  S= 1 4 : pval = 0
## x= 2  y= 4  S= 1 3 : pval = 1.97e-154
##
## Rule 1': 3 -> 2  and  2 - 1  where  3  and  1  not connected and  3 2 1  faithful triple:  2 -> 1
```

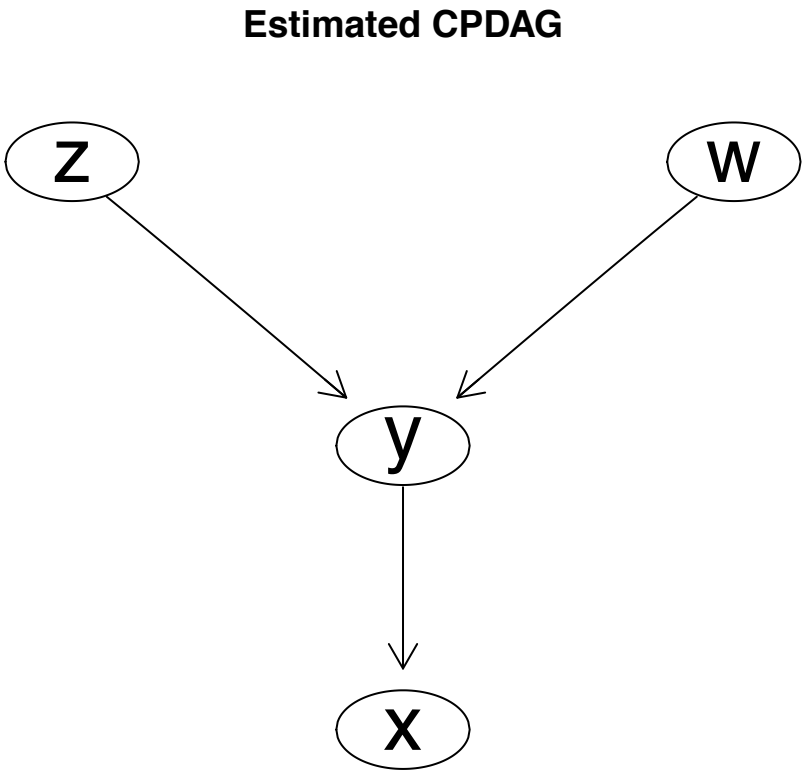
```
### Read through the output - can you see what it has done?

showAmat(pc.fit)
```

```
##
## Adjacency Matrix G:
## G[i,j] = 1/2 if edge mark of edge i-j at j is head/tail.

##   1 2 3 4
## 1 0 2 0 0
## 2 1 0 2 2
## 3 0 1 0 0
## 4 0 1 0 0
```

```
showEdgeList(pc.fit,names)
```

```
##
## Edge List:
##
## Undirected Edges:
##
## Directed Edges:
## y  -->  x
## z  -->  y
## w  -->  y
```

```
if (plotcpdag) {
  plot(pc.fit, main = "Estimated CPDAG",labels=c("x","y","z","w"))
}
```



(2) Second data set

```
#data <- read.table("lab4DAG2.dat")
data <- read.table("http://www.stat.washington.edu/tsr/s566/labs/lab4DAG2.dat")
attr(data,"names")
```

```
## [1] "x" "y" "z" "w"
```

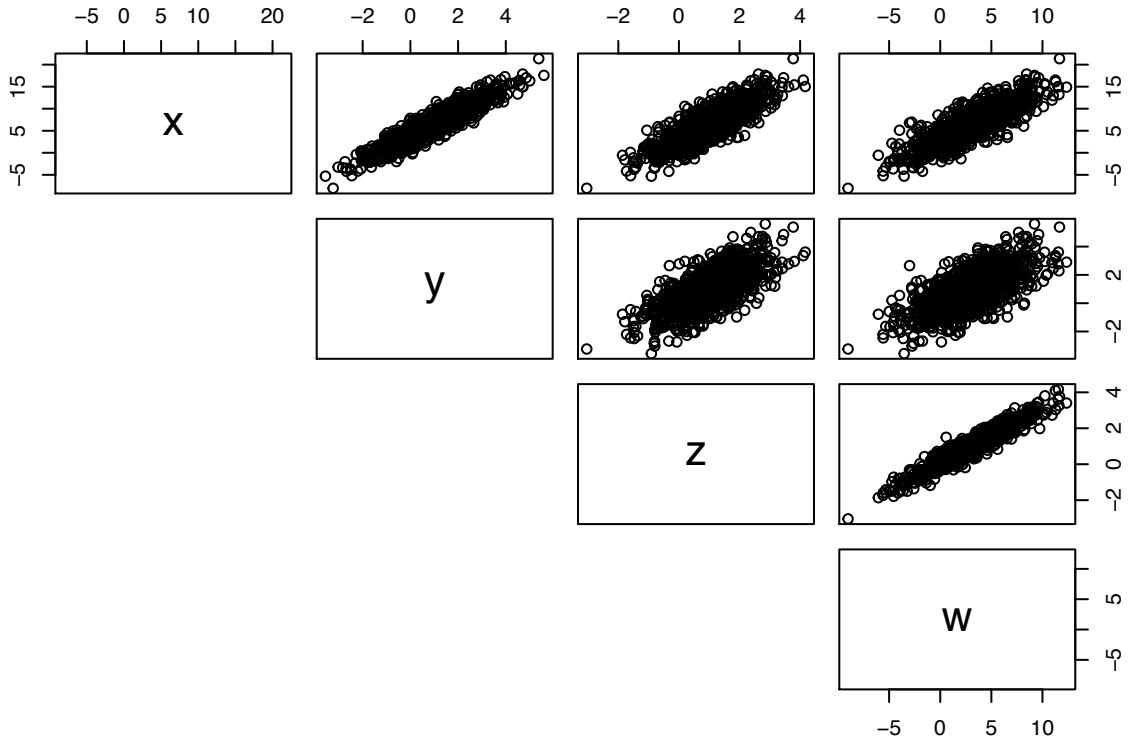
```
x <- data$x
y <- data$y
z <- data$z
w <- data$w

names <- attr(data,"names")
```

(i) the correlation matrix:

```
cor(data)
```

```
##       x     y     z     w
## x 1.000 0.931 0.836 0.830
## y 0.931 1.000 0.704 0.675
## z 0.836 0.704 1.000 0.951
## w 0.830 0.675 0.951 1.000
```

```
# graphical display
pairs(data, lower.panel = NULL)
```




```
#if anything looks close to zero we can test for this using

# cor.test( put-var1-here, put-var2-here )
cor.test(y, z)
```

```
##
##  Pearson's product-moment correlation
##
## data:  y and z
## t = 30, df = 1000, p-value <2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.672 0.734
## sample estimates:
##   cor
## 0.704
```

```
# zeros correspond to marginal independence (under normality)
```

(ii) To check for independence of the form

$$X \perp\!\!\!\perp Y \mid Z,$$

we can do:

```r
tmp <- lm( x ~ y + z)
summary(tmp)
```

```
##
## Call:
## lm(formula = x ~ y + z)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.384 -0.759 -0.011  0.735  4.018
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.0489     0.0479    63.6   <2e-16 ***
## y             1.9756     0.0341    58.0   <2e-16 ***
## z             1.4800     0.0483    30.7   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.09 on 997 degrees of freedom
## Multiple R-squared:  0.931,  Adjusted R-squared:  0.931
## F-statistic: 6.76e+03 on 2 and 997 DF,  p-value: <2e-16
```

```r
# a coefficient of y close to zero indicates independence
# (again assuming normality!)
```

(iii) likewise to test for independence of the form

$$X \perp\!\!\!\perp Y \mid \{Z, W\},$$

we can do:

```r
tmp <- lm( x ~ y + z + w)
summary(tmp)
```

```
##
## Call:
## lm(formula = x ~ y + z + w)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.172 -0.696  0.013  0.672  3.554
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.0383     0.0442   68.70   <2e-16 ***
## y             1.9658     0.0315   62.50   <2e-16 ***
## z             0.2015     0.1066    1.89    0.059 .
## w             0.4285     0.0325   13.20   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 1 on 996 degrees of freedom
## Multiple R-squared:  0.942,  Adjusted R-squared:  0.941
## F-statistic: 5.34e+03 on 3 and 996 DF,  p-value: <2e-16
```

```
##### Using the PC Algorithm to do the job for us:
n <- nrow(data)
p <- ncol(data)
indepTest <- gaussCItest
suffStat <- list(C=cor(data), n = n)
## estimate CPDAG
alpha <- 0.05
pc.fit <- pc(suffStat, indepTest, p = p, alpha = alpha, verbose = TRUE)
```

```
## Order=0; remaining edges:12
## x= 1  y= 2  S=  : pval = 0
## x= 1  y= 3  S=  : pval = 0
## x= 1  y= 4  S=  : pval = 0
## x= 2  y= 1  S=  : pval = 0
## x= 2  y= 3  S=  : pval = 2.7e-168
## x= 2  y= 4  S=  : pval = 7.44e-148
## x= 3  y= 1  S=  : pval = 0
## x= 3  y= 2  S=  : pval = 2.7e-168
## x= 3  y= 4  S=  : pval = 0
## x= 4  y= 1  S=  : pval = 0
## x= 4  y= 2  S=  : pval = 7.44e-148
## x= 4  y= 3  S=  : pval = 0
## Order=1; remaining edges:12
## x= 1  y= 2  S= 3 : pval = 0
## x= 1  y= 2  S= 4 : pval = 0
## x= 1  y= 3  S= 2 : pval = 1.38e-162
## x= 1  y= 3  S= 4 : pval = 2e-18
## x= 1  y= 4  S= 2 : pval = 8.97e-206
## x= 1  y= 4  S= 3 : pval = 5.11e-11
## x= 2  y= 1  S= 3 : pval = 0
## x= 2  y= 1  S= 4 : pval = 0
## x= 2  y= 3  S= 1 : pval = 1.27e-34
## x= 2  y= 3  S= 4 : pval = 1.05e-18
## x= 2  y= 4  S= 1 : pval = 4.21e-61
## x= 2  y= 4  S= 3 : pval = 0.457
## x= 3  y= 1  S= 2 : pval = 1.38e-162
## x= 3  y= 1  S= 4 : pval = 2e-18
## x= 3  y= 2  S= 1 : pval = 1.27e-34
## x= 3  y= 2  S= 4 : pval = 1.05e-18
## x= 3  y= 4  S= 1 : pval = 0
## x= 3  y= 4  S= 2 : pval = 0
## x= 4  y= 1  S= 2 : pval = 8.97e-206
## x= 4  y= 1  S= 3 : pval = 5.11e-11
## x= 4  y= 3  S= 1 : pval = 0
## x= 4  y= 3  S= 2 : pval = 0
## Order=2; remaining edges:10
## x= 1  y= 2  S= 3 4 : pval = 0
## x= 1  y= 3  S= 2 4 : pval = 0.0591
## x= 1  y= 4  S= 2 3 : pval = 1.05e-37
## x= 3  y= 2  S= 1 4 : pval = 0.0285
```

```
## x= 3  y= 4  S= 1 2 : pval = 4.43e-282
```

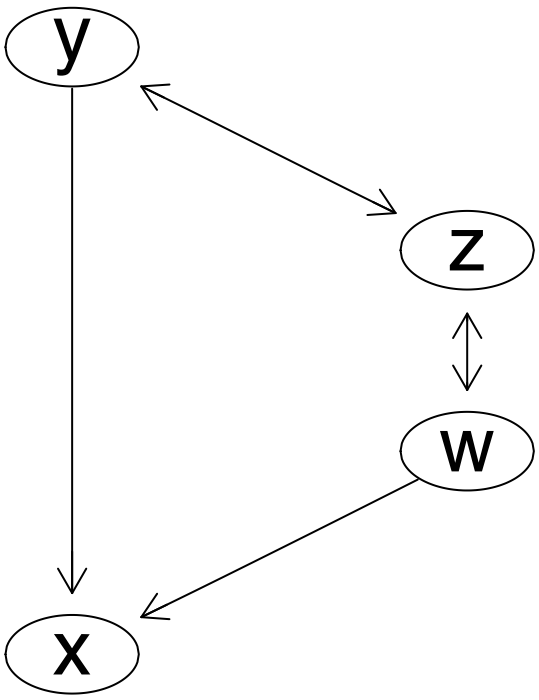
```
showAmat(pc.fit)
```

```
##
## Adjacency Matrix G:
## G[i,j] = 1/2 if edge mark of edge i-j at j is head/tail.

##   1 2 3 4
## 1 0 2 0 2
## 2 1 0 2 0
## 3 0 2 0 2
## 4 1 0 2 0
```

```
showEdgeList(pc.fit,names)
```

```
##
## Edge List:
##
## Undirected Edges:
## y  ---  z
## z  ---  w
##
## Directed Edges:
## y  -->  x
## w  -->  x
```

```
if (plotcpdag) {
  plot(pc.fit, main = "Estimated CPDAG",labels=c("x","y","z","w"))
  ## Note undirected edges are represented here as  <->
}
```



If there is an undirected edge, then you can have either orientation. But the choices are not independent! For example, if you choose $Y \to Z$, then you are not free to have $W \to Z$, because Z would then be an *unshielded collider* with respect to the path $Y \to Z \leftarrow W$.

The graph where Z is an unshielded collider on the path $Y \to Z \leftarrow W$ is not in the same Markov equivalence class to the graph where Z is not an unshielded collider on the path $Y - Z - W$.

So if you have an inferred graph $\mathcal{G}$ where $Y \to Z$, then you must have $Z \to W$ under the probability distribution that factorizes according to $\mathcal{G}$. (And conversely if you have $W \to Z$, then you must have $Z \to Y$.)

(3) Third data set

```
#data <- read.table("lab4DAG2.dat")
data <- read.table("http://www.stat.washington.edu/tsr/s566/labs/lab4DAG3.dat")
attr(data,"names")
```

```
## [1] "x" "y" "z" "w"
```

```
x <- data$x
y <- data$y
z <- data$z
w <- data$w

names <- attr(data,"names")

# Some linear regressions again
summary(lm(z~w, data))$coef
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.9694     0.0690  14.058 4.34e-41
## w             0.0117     0.0302   0.386 6.99e-01
```

```
summary(lm(z~w + x, data))$coef
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.827     0.0652   12.67 3.11e-34
## w              0.205     0.0322    6.37 2.84e-10
## x              0.322     0.0259   12.41 5.78e-33
```

```
summary(lm(z~w + y, data))$coef
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.6713     0.1112    6.04 2.22e-09
## w             0.1031     0.0403    2.56 1.07e-02
## y             0.0975     0.0286    3.40 6.88e-04
```

```
##### Using the PC Algorithm to do the job for us:
n <- nrow(data)
p <- ncol(data)
indepTest <- gaussCItest
suffStat <- list(C=cor(data), n = n)

## estimate CPDAG
alpha <- 0.05
pc.fit <- pc(suffStat, indepTest, p = p, alpha = alpha, verbose = TRUE)
```

```
## Order=0; remaining edges:12
## x= 1  y= 2  S=  : pval = 4.64e-80
## x= 1  y= 3  S=  : pval = 1.07e-24
## x= 1  y= 4  S=  : pval = 1.44e-62
## x= 2  y= 1  S=  : pval = 4.64e-80
## x= 2  y= 3  S=  : pval = 0.0231
## x= 2  y= 4  S=  : pval = 5.95e-142
## x= 3  y= 1  S=  : pval = 1.07e-24
## x= 3  y= 2  S=  : pval = 0.0231
## x= 3  y= 4  S=  : pval = 0.699
## x= 4  y= 1  S=  : pval = 1.44e-62
## x= 4  y= 2  S=  : pval = 5.95e-142
## Order=1; remaining edges:10
## x= 1  y= 2  S= 3 : pval = 2.69e-82
## x= 1  y= 2  S= 4 : pval = 4.59e-27
## x= 1  y= 3  S= 2 : pval = 7.86e-27
## x= 1  y= 3  S= 4 : pval = 1.05e-33
## x= 1  y= 4  S= 2 : pval = 1.16e-10
## x= 1  y= 4  S= 3 : pval = 5.49e-72
## x= 2  y= 1  S= 3 : pval = 2.69e-82
## x= 2  y= 1  S= 4 : pval = 4.59e-27
## x= 2  y= 3  S= 1 : pval = 0.000127
## x= 2  y= 3  S= 4 : pval = 0.000682
## x= 2  y= 4  S= 1 : pval = 9.47e-85
```

```
## x= 2  y= 4  S= 3 : pval = 1.75e-143
## x= 3  y= 1  S= 2 : pval = 7.86e-27
## x= 3  y= 2  S= 1 : pval = 0.000127
## x= 4  y= 1  S= 2 : pval = 1.16e-10
## x= 4  y= 2  S= 1 : pval = 9.47e-85
## Order=2; remaining edges:10
## x= 1  y= 2  S= 3 4 : pval = 1.78e-24
## x= 1  y= 3  S= 2 4 : pval = 4.34e-31
## x= 1  y= 4  S= 2 3 : pval = 7e-15
## x= 2  y= 1  S= 3 4 : pval = 1.78e-24
## x= 2  y= 3  S= 1 4 : pval = 0.641
## x= 2  y= 4  S= 1 3 : pval = 3.82e-81
##
## Rule 1': 3 -> 1  and  1 - 2  where  3  and  2  not connected and  3 1 2  faithful triple:  1 -> 2
##
## Rule 2: Chain  4 -> 1 -> 2 : 4 -> 2
```

```
showAmat(pc.fit)
```

```
##
## Adjacency Matrix G:
## G[i,j] = 1/2 if edge mark of edge i-j at j is head/tail.

##   1 2 3 4
## 1 0 1 2 2
## 2 2 0 0 2
## 3 1 0 0 0
## 4 1 1 0 0
```

```
showEdgeList(pc.fit,names)
```

```
##
## Edge List:
##
## Undirected Edges:
##
## Directed Edges:
## x  -->  y
## z  -->  x
## w  -->  x
## w  -->  y
```

```
if (plotcpdag) {
  plot(pc.fit, main = "Estimated CPDAG",labels=c("x","y","z","w"))
}
```

Estimated CPDAG