Hands-On Graphical Causal Modeling Using R

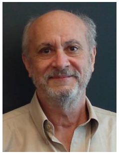
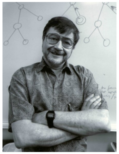
Johannes Textor

March 11, 2016

Causality theory

- A causality theory provides a language to encode causal relationships.
- A causality theory helps decide when, and how, causation can be inferred from domain knowledge and data.
- We use the theory based on structural causal models.

Some people with their own causality theories

Donald Rubin	Judea Pearl	Donald Campbell	Phil Dawid	Clive Granger
				

[. . .] all approaches to causation are variants or abstractions of [. . .] structural theory [. . .].

– Judea Pearl, "Causality"

Graphs and paths

- A graph consists of nodes (vertices) and edges (arrows or lines).

$$\begin{tikzcd}
 & Z \arrow[dl, leftrightarrow] \arrow[dd] \arrow[dr, dash] & \\
A \arrow[dr] & & D \\
 & E \arrow[ur] &
\end{tikzcd}$$

- We describe node relations using kinship terminology.
  - Z is a parent of E, D.
  - D is a child of Z, E.
  - D is a descendant of A.
  - A is an ancestor of D.

  (terms for $\leftrightarrow, -$ (spouse, neighbour) are less used)
- A path is a sequence of connected nodes (moving against arrows is allowed), e.g.$Z \leftrightarrow A \rightarrow E \leftarrow Z \rightarrow D$.

The dagitty package

I am going to show how to work with structural causal models in R using the package ‘dagitty‘. The package is not yet on CRAN, so it needs to be installed as follows:

```r
install.packages("devtools") #if you haven't already
library(devtools)
install_github("jtextor/dagitty/r")
```

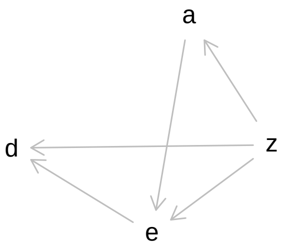
If you wish to follow this talk by copy-pasting code (encouraged): johannes-textor.name/leeds.pdf

Defining graphs in R

We define graphs using a textual syntax based on the program "graphviz". A simple command exists to plot graphs.

```
g <- dagitty( "dag {
a -> e
e -> d
z -> a
z -> e
z -> d
}" )
```

```
plot(graphLayout(g))
```



Or, more briefly:

```
g <- dagitty( "dag{{a->e->d}<-z}" )
```

Defining graphs in R

The function `graphLayout` generates automatic layouts. This works OK, but of course you can also specify the layout yourself.

```r
g <- dagitty( 'dag {
a [pos="0,0"]
e [pos="1,0"]
d [pos="2,0"]
z [pos="0,1"]
z->{a->e->d}}' )
```

```r
g <- dagitty( 'dag{
z->{a->e->d}}' )

coordinates(g) <- list(
X=c(a=0,b=1,c=2,z=1),
Y=c(a=1,b=1,c=1,z=0))
```

Or build it in the dagitty.net GUI and download it.

```r
g <- downloadGraph("dagitty.net/m331")
```

Ancestry relationships in R

The dagitty package contains functions for ancestral relationships.

```
parents( g, "Z" )

## [1] "W" "Y"
```

```
ancestors( g, "Z" )

## [1] "Z" "Y" "X" "W"
```

```
children( g, "W" )

## [1] "Y" "Z"
```

```
descendants( g, "W" )

## [1] "W" "Z" "T" "Y"
```

$$\begin{tikzcd}
 & W \arrow[d] \arrow[dr] & \\
X \arrow[ur] \arrow[r] & Y \arrow[r] \arrow[dr] & Z \arrow[d] \\
 & & T
\end{tikzcd}$$

Paths in R

For detailed inspection of paths and their statuses, dagitty provides the function ‘paths‘. This function returns a list with two components, ‘paths‘ and ‘open‘. I’ll explain later what the ‘open‘ is about.

```r
paths( g, "Y", "Z" )$paths
```

```
## Y -> T <- Z
## Y -> Z
## Y <- W -> Z
## Y <- X -> W -> Z
```

$$\begin{tikzcd}
 & W \arrow[d] \arrow[dr] & \\
X \arrow[ur] \arrow[r] & Y \arrow[r] \arrow[dr] & Z \arrow[d] \\
 & & T
\end{tikzcd}$$

Structural causal models (informally)

- A structural causal model is a graph whose nodes represent variables, and whose edges represent causal relations.
- A DAG is a directed acyclic graph.
- We draw $X \to Y$ if X might have causal influence on Y. Otherwise we omit the arrow.
- We draw $X \leftrightarrow Y$ if there is a hidden variable influencing both X and Y.

Example

$$\begin{tikzcd}
& Z \arrow[dd] \arrow[dr] & \\
A \arrow[ur, leftrightarrow] \arrow[dr] & & D \\
& E \arrow[ur] &
\end{tikzcd}$$

Z and A influence E
Something hidden influences A and Z
Z and E influence D

What do we mean by "influences"?

Structural causal models (formally)

A structural causal model consists of a graph $G = (V, E)$ and a set of functions $\{f_X\}$ such that, for each variable X:

$$X := f_X(\mathrm{pa}_X, \epsilon_X)$$

- f_X is any deterministic function.
- pa_X is the set of all parents of X in G.
- ϵ_X is a random variable.

For $X \neq Y$, ϵ_X and ϵ_Y are independent unless $Xleftrightarrow Y$.

Example

$$\begin{tikzcd}
 & Z \arrow[dl] \arrow[dr] \arrow[dd] & \\
A \arrow[dr] & & D \\
 & E \arrow[ur] &
\end{tikzcd}$$

$$E := f_E(A, Z, \epsilon_E)$$
$$A := f_A(Z, \epsilon_A)$$
$$D := f_D(Z, E, \epsilon_D)$$

"Lady Nature looks up the values of Z and A when determining E"

Structural equation models

- A Structural Equation Model (SEM, SEmodel) is a causal diagram where the functions are restricted to linear functions with additive Gaussian noise.

Example

$$\begin{tikzcd}
 & Z \arrow[dl] \arrow[dd] \arrow[dr] & \\
A \arrow[dr] & & D \\
 & E \arrow[ur] &
\end{tikzcd}$$

$$\begin{aligned}
E &:= \beta_{AD} A + \beta_{ZD} Z + \epsilon_E \\
A &:= \beta_{ZA} Z + \epsilon_A \\
D &:= \beta_{EA} E + \beta_{ZD} Z + \epsilon_D
\end{aligned}$$

Simulating data

It is often useful to simulate data from a DAG.

```
g <- "dag{a<->b<-c}"
cor(simulateSEM(g,.5,.5))
```

```
g <- "dag{a<-b<->c}"
cor(simulateSEM(g,.5,.5))
```

```
##     a   b   c
## a 1.0 0.5 0.0
## b 0.5 1.0 0.5
## c 0.0 0.5 1.0
```

```
##      a   b    c
## a 1.00 0.5 0.25
## b 0.50 1.0 0.50
## c 0.25 0.5 1.00
```

‘simulateSEM‘ interprets the DAG as a structural equation model and samples path coefficients from the given interval.

Summary

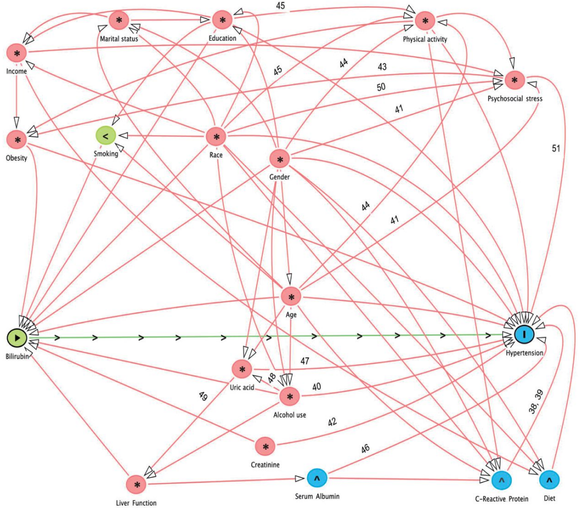
- Structural causal models consist of nodes (variables) and edges (functional relations).
- DAGs are models where all edges have arrowheads and that do not contain cycles.
- Structural equation models are DAGs where all edges represent linear relations with additive Gaussian noise.

Testing structural causal models

- Structural causal models are used for inference (e.g., computing covariate adjustment sets for regression).
- Such inferences depend on the validity of the model (frequent criticism: how do you know the model is correct?)
- Structural causal models can be tested.
- The key idea behind graphical model testing is consistency.

An example

Wang and Bautista, Int. J. Epidemiol. (2014)

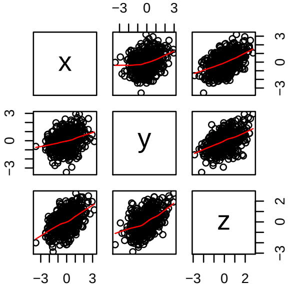


● Exposure; ● Outcome; ● Ancestor of exposure; ● Ancestor of outcome; ● Ancestor of exposure *and* outcome; 🟩 Causal path; 🟥 Biasing path

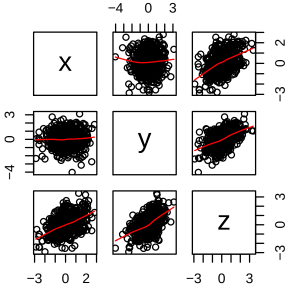
Consistency

Given a probability distribution P and a graph G for some variables V, we say that P is consistent with G if there exists any set of functions and error terms by which G generates P.

$$\begin{tikzcd} x & z \arrow[l] \arrow[r] & y \end{tikzcd}$$



$$\begin{tikzcd} x \arrow[r] & z & y \arrow[l] \end{tikzcd}$$



Consistency is intimately related to how variables correlate. We will now formalize this intuition.

Conditional independence and vanishing covariance

A few definitions first:

- $\mathrm{Cov}(X, Y \mid Z)$: partial covariance between X and Y given Z.
- $X \perp\!\!\!\perp Y$ means: X and Y are independent.
- $X \perp\!\!\!\perp Y \mid Z$ means: X and Y are independent given Z.
  Z and Y tell us no more about X than Z alone does.
- If $X \perp\!\!\!\perp Y$, then $\mathrm{Cov}(X, Y) = 0$.
- If $X \perp\!\!\!\perp Y \mid Z$, then $\forall z : \mathrm{Cov}(X, Y \mid Z = z) = 0$.

d-Separation

- A collider is a path of length 3 that looks like $X \to M \leftarrow Y$.
- All other paths of length 3 are called non-colliders: $X \to M \to Z$, $X \leftarrow M \leftarrow Y$, $X \leftarrow M \to Z$

d-separation

A set Z d-separates (blocks) a path, if

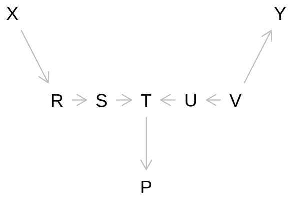
- The midpoint M of some non-collider is in Z; or
- The midpoint M of some collider is not an ancestor of any variable in Z.

Theorem (Verma & Pearl, 1984)

If all paths between X and Y are closed by the set Z, then $X \perp\!\!\!\perp Y \mid Z$ in every consistent probability distribution.

Path inspection in R

```
g <- dagitty("dag{ X -> R -> S -> T <- U <- V -> Y
T -> P }")
```

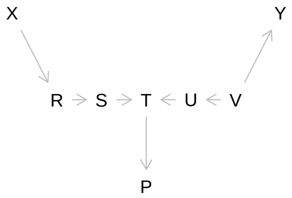


```
paths(g,"X","Y")

## $paths
## [1] "X -> R -> S -> T <- U <- V -> Y"
##
## $open
## [1] FALSE
```

Path inspection in R

```r
g <- dagitty("dag{ X -> R -> S -> T <- U <- V -> Y
T -> P }")
```

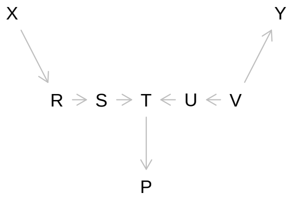


```r
paths(g,"X","Y","T")

## $paths
## [1] "X -> R -> S -> T <- U <- V -> Y"
##
## $open
## [1] TRUE
```

Path inspection in R

```r
g <- dagitty("dag{ X -> R -> S -> T <- U <- V -> Y
T -> P }")
```


$$\begin{tikzcd}
X \arrow[dr] & & & & & Y \\
& R \arrow[r] & S \arrow[r] & T \arrow[d] & U \arrow[l] & V \arrow[l] \arrow[u] \\
& & & P & &
\end{tikzcd}$$

```r
paths(g,"X","Y","P")

## $paths
## [1] "X -> R -> S -> T <- U <- V -> Y"
##
## $open
## [1] TRUE
```

Verifying d-separation in R

```
g <- "dag { z -> m -> y } "
impliedConditionalIndependencies(g)

## y _||_ z | m

confint( lm( y ~ z + m, data=simulateSEM( g ) ) )

##                2.5 % 97.5 %
## (Intercept) -0.045  0.104
## z           -0.109  0.062
## m           -0.558 -0.385
```

Discovering model misspecifications

d-Separation is a useful tool to check whether the DAGs that we postulate are in fact correct.

$$\begin{tikzcd}
 & \boxed{M_1} \arrow[dr] & \\
\boxed{X} \arrow[ur] \arrow[dr] & & \boxed{Y} \\
 & \boxed{M_2} \arrow[ur] &
\end{tikzcd}$$

assumed model
(perfect measurement)
$M_1 \perp\!\!\!\perp M_2 \mid X$
$X \perp\!\!\!\perp Y \mid \{M_1, M_2\}$

$$\begin{tikzcd}
 & & \boxed{M_1} \\
 & U_1 \arrow[ur] \arrow[dr] & \\
\boxed{X} \arrow[ur] \arrow[dr] & & \boxed{Y} \\
 & \boxed{M_2} \arrow[ur] &
\end{tikzcd}$$

true model
(measurement error)
$M_1 \perp\!\!\!\perp M_2 \mid X$

Model testing in R

```r
# Simulate data from true model
d <- simulateSEM("dag{X->{U1 M2}->Y U1->M1}",.6,.6)
```

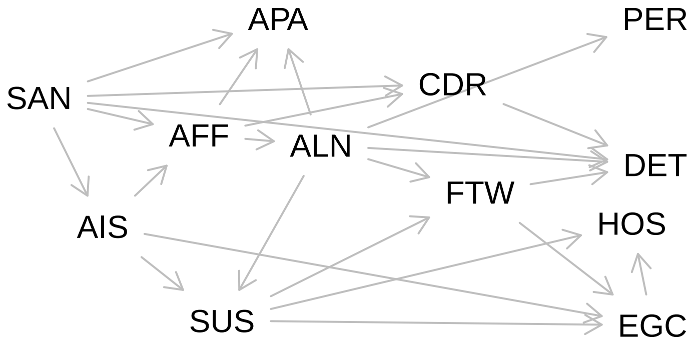
```r
# Test postulated model
localTests( "dag{ X -> {M1 M2} -> Y }", d, "cis" )
```

```
##                    estimate std.error p.value  2.5% 97.5%
## M1 _||_ M2 | X       -0.004     0.052 9.4e-01 -0.11 0.098
## X _||_ Y | M1, M2     0.650     0.066 3.0e-21  0.52 0.779
```

Indeed, we find that the second implication is violated.

A real-world example

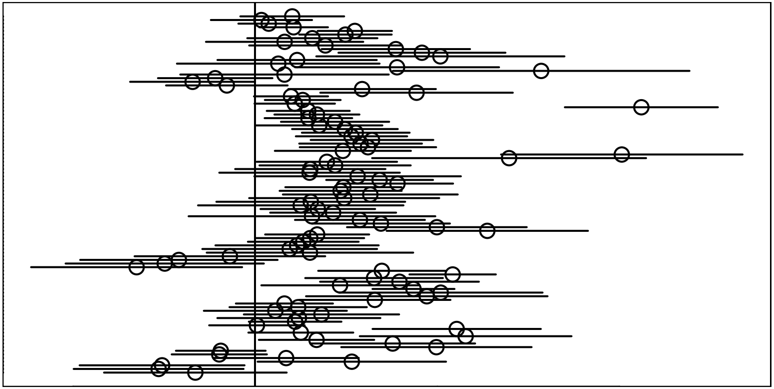
```
plot( getExample("Kampen") )
```



SSQ model of schizophrenic prodromal unfolding
van Kampen, European Psychiatry, 2014

A real-world example

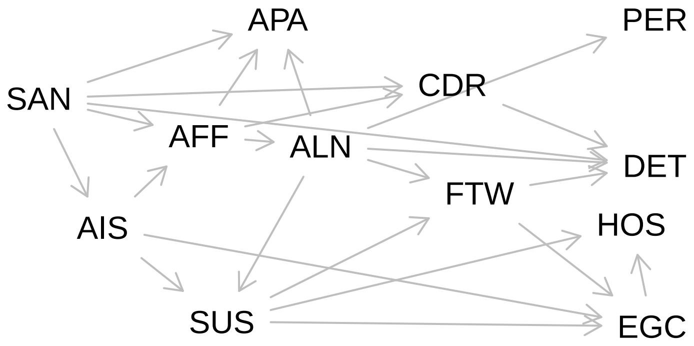
```
tests <- localTests( getExample("Kampen"), d, "cis" )
plotLocalTestResults( tests )
```



A real-world example

```r
head(tests[order(tests$p.value),])
```

```
##                          estimate std.error p.value  2.5% 97.5%
## DET _||_ PER | ALN           0.42     0.043 7.9e-22 0.340  0.51
## ALN _||_ SAN | AFF           0.22     0.024 2.2e-18 0.170  0.26
## ALN _||_ CDR | AFF           0.17     0.023 9.7e-14 0.129  0.22
## CDR _||_ PER | AFF           0.40     0.068 3.9e-09 0.270  0.54
## PER _||_ SAN | AFF           0.11     0.021 1.7e-07 0.069  0.15
## CDR _||_ FTW | AFF, AIS      0.16     0.031 6.0e-07 0.095  0.22
```



Summary

- Please test your models!
- 1 line of code using the daitty package . . .

Equivalent models

- Sometimes two models have exactly the same testable implications.
- Such models are called equivalent – they cannot be distinguished by statistical means alone.

```
impliedConditionalIndependencies("dag{ x -> m -> y }")

## x _||_ y | m
```

```
impliedConditionalIndependencies("dag{ x <- m <- y }")

## x _||_ y | m
```

Markov equivalence

- Models with the same testable implications are called Markov equivalent.

$$\begin{tikzcd}
 & Z \arrow[dr] \arrow[dd] & \\
A \arrow[dr] & & D \arrow[dl] \\
 & E &
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
 & Z \arrow[dd] & \\
A \arrow[dr] & & D \arrow[ul] \arrow[dl] \\
 & E &
\end{tikzcd}$$

$$D \perp\!\!\!\perp A \quad ; \quad A \perp\!\!\!\perp Z \qquad\qquad D \perp\!\!\!\perp A \quad ; \quad A \perp\!\!\!\perp Z$$

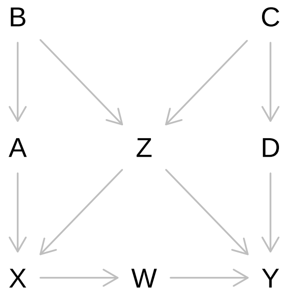
- Two models are Markov equivalent iff they have the same "skeleton" (edges w/o arrowheads) and the same "immoralities" (children of "unmarried" = unlinked parents).

$$\begin{tikzcd}
 & Z \arrow[dr, dash] \arrow[dd, dash] & \\
A \arrow[dr, dash] & & D \arrow[dl, dash] \\
 & E &
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
 & Z \arrow[dr, dash] \arrow[dd] & \\
A \arrow[ur, dash, dashed, bend left] \arrow[dr] & & D \arrow[dl, dash] \\
 & E &
\end{tikzcd}$$

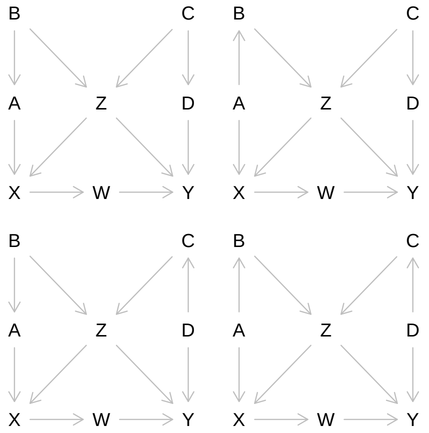
Equivalent models in R

```r
g <- dagitty( "dag{
B->{A Z}->X->W->Y
C->{Z D}->Y}" )
```

```r
plot(g)
```



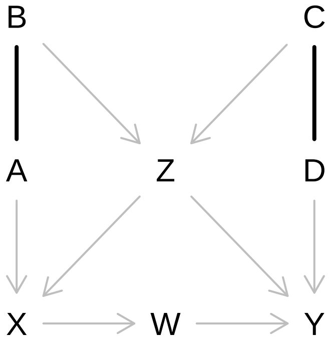
```r
for( g2 in equivalentDAGs(g) ){
        plot(g2)
}
```



Equivalent models

```
plot(equivalenceClass(g))
```

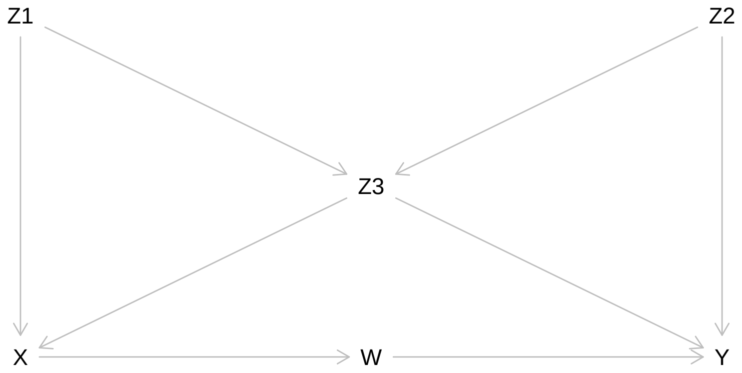
The equivalence class is a graph that contains both arrows and lines (undirected edges). The lines can be oriented in either direction, as long as no immorality and no cycle is created.



$$\begin{tikzcd}
B \arrow[d, dash] \arrow[dr] & & C \arrow[d, dash] \arrow[dl] \\
A \arrow[d] & Z \arrow[dl] \arrow[dr] & D \arrow[d] \\
X \arrow[r] & W \arrow[r] & Y
\end{tikzcd}$$

Equivalence classes

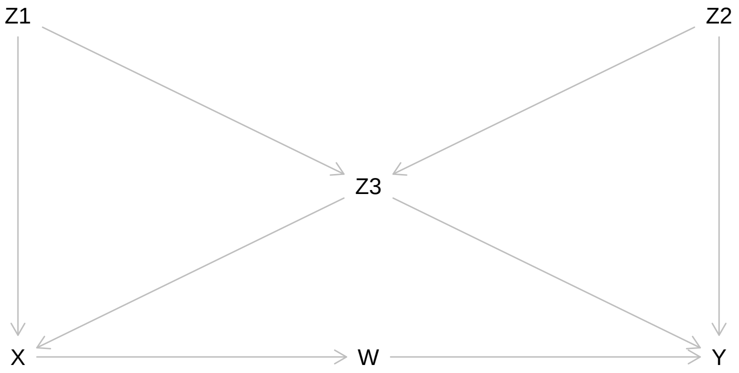
How many equivalent models do you think this DAG has?



$$\begin{tikzcd}
Z1 \arrow[d] \arrow[dr] & & Z2 \arrow[d] \arrow[dl] \\
 & Z3 \arrow[dl] \arrow[dr] & \\
X \arrow[r] & W \arrow[r] & Y
\end{tikzcd}$$

Equivalence classes

How many equivalent models do you think this DAG has?

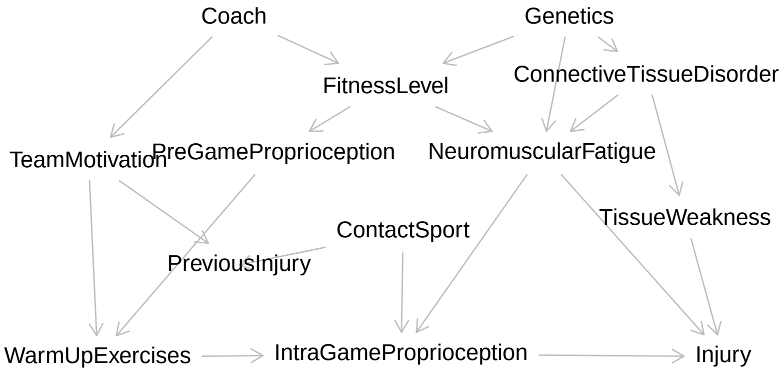

$$\begin{tikzcd}
Z1 \arrow[dd] \arrow[dr] & & Z2 \arrow[dd] \arrow[dl] \\
& Z3 \arrow[dl] \arrow[dr] & \\
X \arrow[r] & W \arrow[r] & Y
\end{tikzcd}$$

```
length(equivalentDAGs( g ))

## [1] 1
```

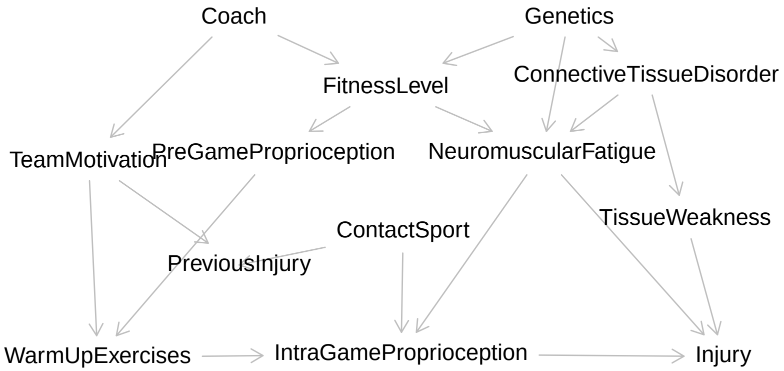
Equivalence classes

How many equivalent models do you think this DAG has?



Equivalence classes

How many equivalent models do you think this DAG has?

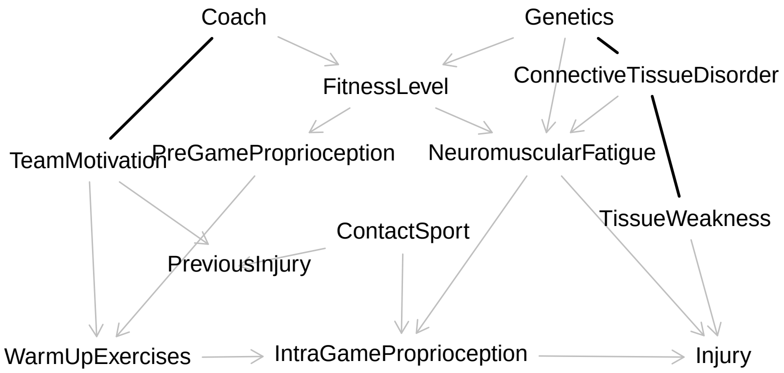


```
length(equivalentDAGs( g ))

## [1] 6
```

Equivalence classes

```
plot(equivalenceClass(g))
```



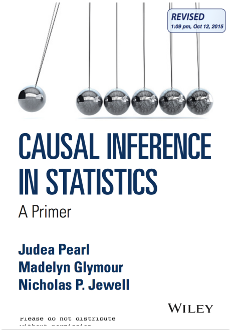
Only 3 edges can be reversed!

Summary

- Markov equivalent models are statistically indistinguishable (unless parametric assumptions are made).
- Immoralities are key to break Markov equivalence.
- There can be surprisingly few equivalent model if there are many immoralities.

Conclusion

- dagitty is known as a web-based user interface for drawing and analyzing graphs.
- An R package is now available that does everything the GUI can, and much more.
- A dagitty-based companion for Judea Pearl's new book is available at dagitty.net/primer.


REVISED
1:09 pm, Oct 12, 2015

CAUSAL INFERENCE IN STATISTICS

A Primer

Judea Pearl
Madelyn Glymour
Nicholas P. Jewell

WILEY

Please do not distribute without permission

Acknowledgements

- George Ellison and Mark Gilthorpe, Leeds :)
- Bela van der Zander and Maciej Liśkiewicz, Lübeck, Germany
- Felix Thoemmes, Cornell
- Judea Pearl and Andrew Forney, UCLA