Introduction to Causal Reasoning on DAGS

A tutorials using R and the bnlearn package

Sara Taheri

Contents

Find the video tutorials from here.

Installing bnlearn

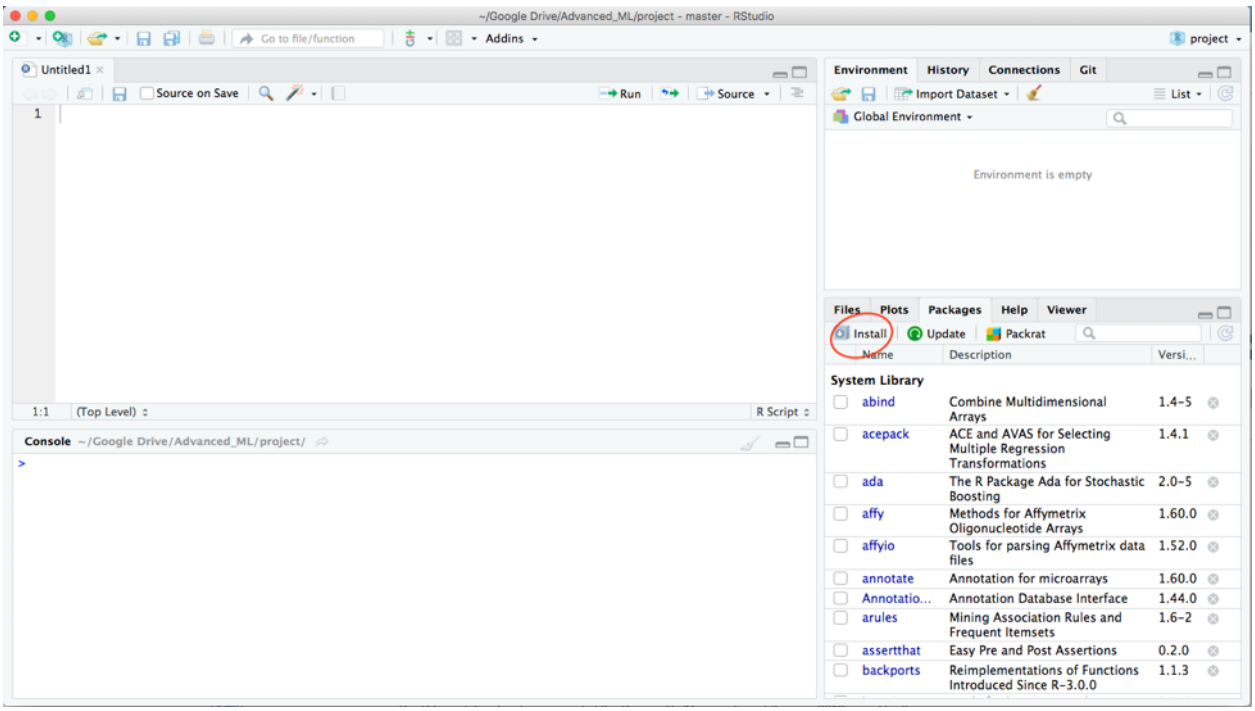
Open RStudio and in console type:

```
install.packages("bnlearn")
install.packages("Rgraphviz")
install.packages("png")
```

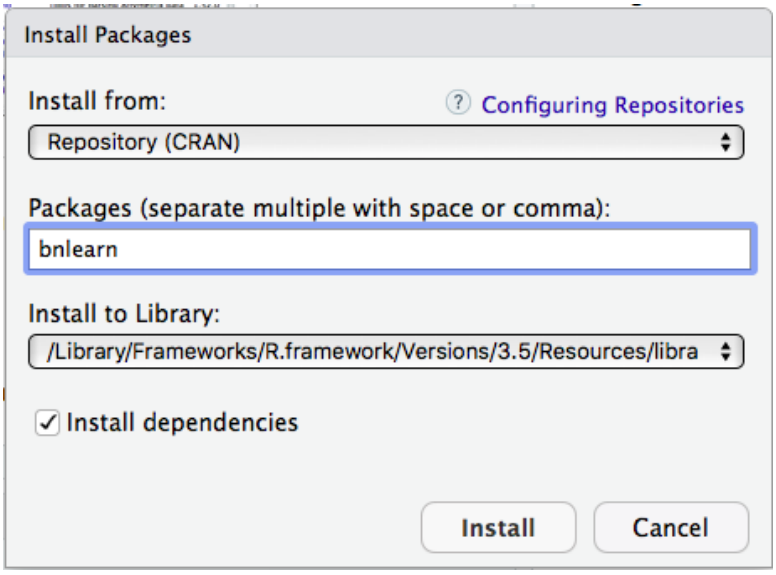
If you experience problems installing **Rgraphviz**, try the following script:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install("Rgraphviz")
```

Another way to install a package in R is to go to the "Packages" tab, and click on the "Install" button.



Then type “bnlearn” in the window that appears and click on the install button. Do the same thing for the other package.



Overview of a Transportation Survey Data Set

Data description

This data set contains information about usage of different transportation systems with a focus on cars and trains for different social groups. It is comprised of categorical variables with the following names and values:

- **Age (A):** The age of the individual, recorded as *young* (**young**) for individuals below 30 years, *adult* (**adult**) for individuals between 30 and 60 years old, and *old* (**old**) for people older than 60.
- **Sex (S):** The biological sex of individual, recorded as *male* (**M**) or *female* (**F**).
- **Education (E):** The highest level of education or training completed by the individual, recorded either *high school* (**high**) or *university degree* (**uni**).
- **Occupation (O):** It is recorded as an *employee* (**emp**) or a *self employed* (**self**) worker.
- **Residence (R):** The size of the city the individual lives in, recorded as *small* (**small**) or *big* (**big**).
- **Travel (T):** The means of transport favoured by the individual, recorded as *car* (**car**), *train* (**train**) or *other* (**other**)

Travel is the *target* of the survey, the quantity of interest whose behaviour is under investigation.

```
# empty graph
dag <- empty.graph(nodes = c("A","S","E","O","R","T"))
arc.set <- matrix(c("A", "E",
                    "S", "E",
```

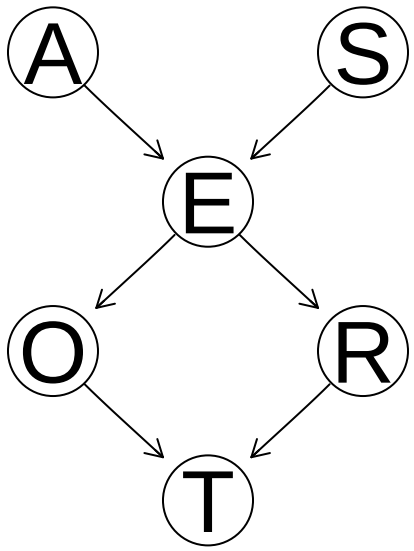
```
                    "E", "O",
                    "E", "R",
                    "O", "T",
                    "R", "T"),
                  byrow = TRUE, ncol = 2,
                  dimnames = list(NULL, c("from", "to")))
arcs(dag) <- arc.set
nodes(dag)
```

```
## [1] "A" "S" "E" "O" "R" "T"
```

```
arcs(dag)
```

```
##      from to
## [1,] "A"  "E"
## [2,] "S"  "E"
## [3,] "E"  "O"
## [4,] "E"  "R"
## [5,] "O"  "T"
## [6,] "R"  "T"
```

```
graphviz.plot(dag)
```



Learning a (causal) DAG from data

The strength of the `bnlearn` library is it's set of algorithsm for learning DAGs, including causal DAGs, from data.

However, this is a subject that warrants its own tutorial. We won't address the topic here. To learn more, I suggest visiting `bnlearn`'s website.

Key Graphical Concepts in DAGs

Whenever we use graphs in an analytical context, it is because it is useful to use graphical language and algorithms to reason about the problem.

The same is true in causal generative modeling. Here we introduce a few key concepts and some `bnlearn` abstractions applying them. In the next section we show how to use these concepts to reason about the data generating process we wish to model.

V-structure: A **collider** or a **V structure** is a V-motif formed by a node and two incoming parents. When the parents are connected, this is called a "moral" V-structure (because the parents are "married") and when they are not it is called an immoral V-structure. Less prudish terminology includes "open/closed V-structure" and "shielded/unshielded collider".

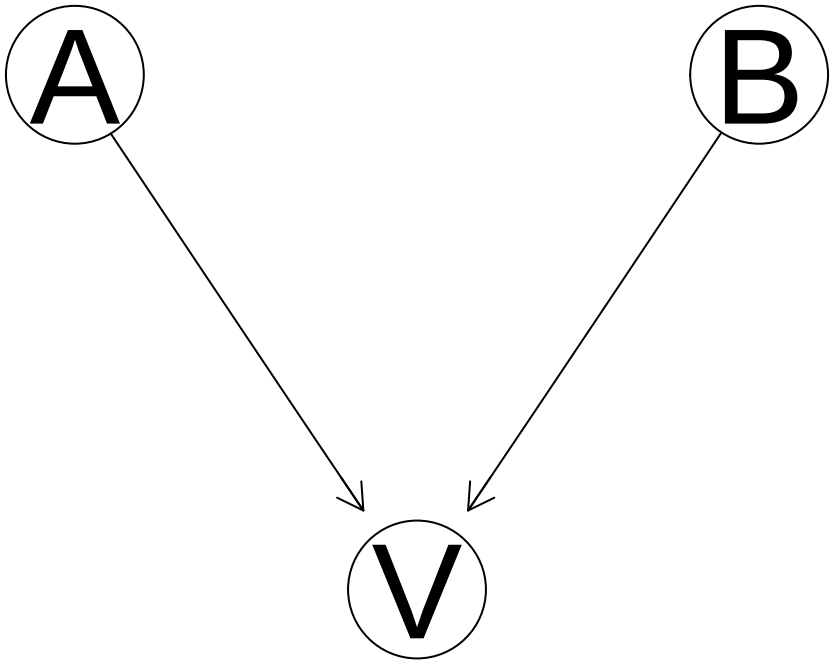


Figure 1: V is a collider

You can find all the **V structures** of a DAG:

```
vstructs(dag)
```

```
##      X   Z   Y
## [1,] "A" "E" "S"
## [2,] "O" "T" "R"
```

Note that conditioning on a collider induces dependence even though the parents aren't directly connected.

Markov blanket: The Markov blanket of a node is comprised of its parents, children and children's other parents. In other words, it is comprised of its parents, children, and any other members of V-structures to which it belongs. In the figure below, all the nodes inside the circle are Markov blanket of node A.

In R, we can find the Markov blankets of a node, by using the `mb` function from **bnlearn**.

```
#Markov blanket of node E in survey data set
mb(dag, node = "E")
```

```
## [1] "A" "S" "O" "R"
```

D-separation: The nodes in the DAG correspond to random variables, and we usually represent random variables with capital letters like X and Y. Let's put random variables aside for now an think purely in terms of nodes. Let's use bold capital letters to represent sets of nodes.

Let P be an undirected path (that is, a path which can go in either direction) from node U to V. Note that a path is just a set of nodes. Then the path P is said to be d-separated by a set of nodes **Z** if and only if (at least) one of the following holds:

- P contains a chain, I $\rightarrow$ M $\rightarrow$ J, such that the middle node M is in **Z**
- P contains a chain, I $\leftarrow$ M $\leftarrow$ J, such that the middle node M is in **Z**
- P contains a fork, I $\leftarrow$ M $\rightarrow$ J, such that the middle node M is in **Z**
- P contains an V-structure (or collider), I $\rightarrow$ M $\leftarrow$ J, such that the middle node M is not in **Z** and no descendant of M is in **Z**.

Thus U and V are said to be d-separated by **Z** if all undirected paths between them are d-separated. In Figure 2, U is d-separated from V given M and K (**Z** = {M,K}), because P (U - I - M - J - V) contains a chain (I $\rightarrow$ M $\rightarrow$ J) such that the middle node M in in **Z**.

In Figure 3, U is d-separated from V given l (Z = l), because P (U - I - M - J - V) contains a V-structure (or collider), I $\rightarrow$ M $\leftarrow$ J, such that the middle node M is not in **Z** and no descendant of M is in **Z**.

We can investigate whether two nodes in a **bn** object are d-separated using the **dsep** function. **dsep** takes three arguments, x, y and z; the first two must be the names of two nodes being tested for d-separation, while the latter is an optional d-separating set. So, for example,

```
dsep(dag, x = "S", y = "R")
```

```
## [1] FALSE
```

```
dsep(dag, x = "O", y = "R")
```

```
## [1] FALSE
```

```
dsep(dag, x = "S", y = "R", z = "E")
```

```
## [1] TRUE
```

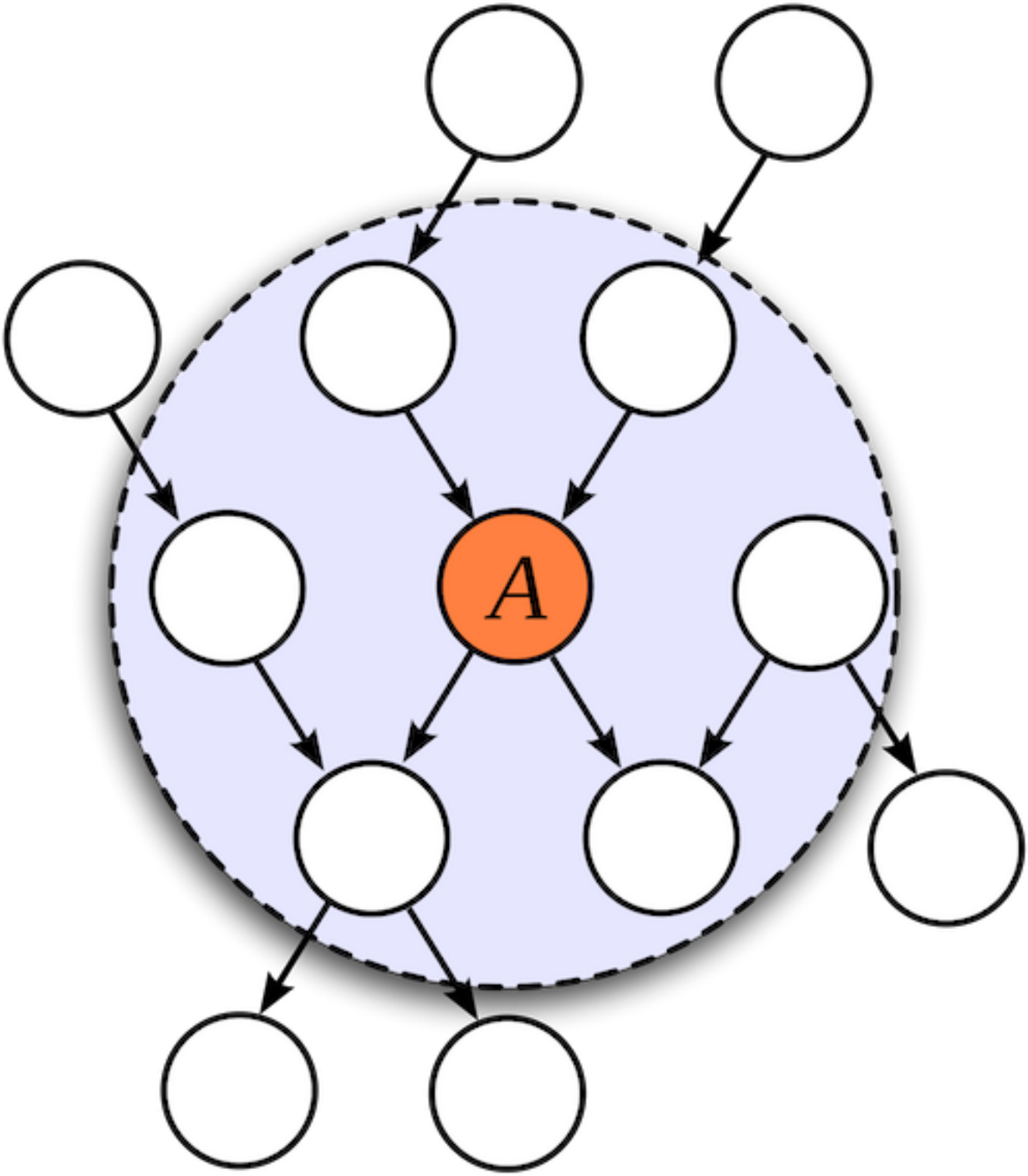


Figure 2: Figure 1: Markov blanket of node A are the nodes inside the circle.

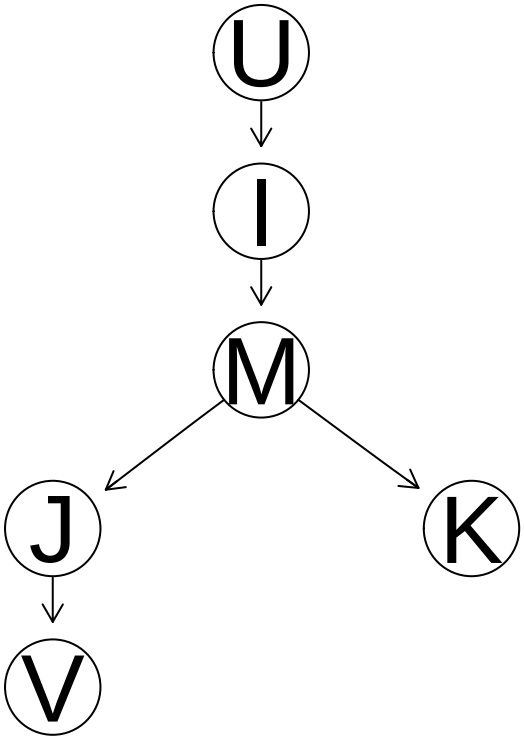


Figure 3: Figure 2: u is d-separated from v given m and k

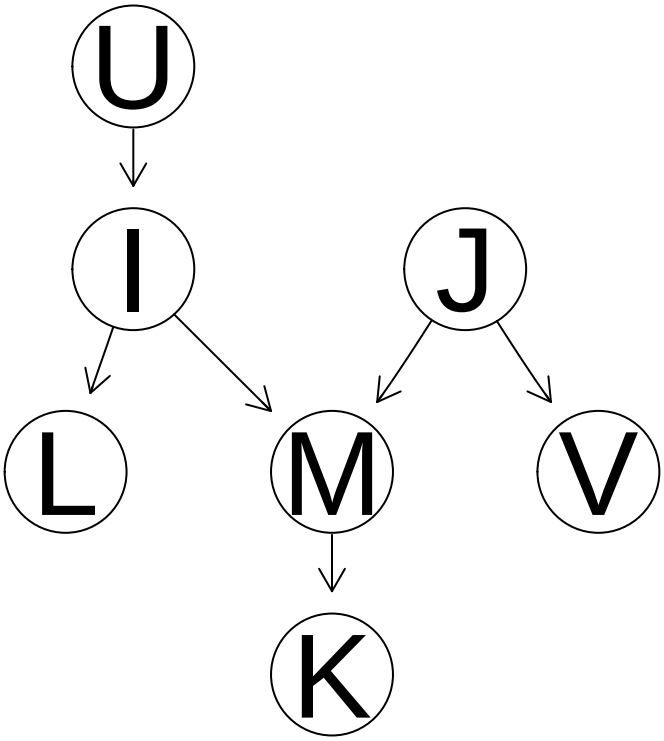


Figure 4: Figure 3: u is d-separated from v given l

```r
dsep(dag, x = "O", y = "E", z = c("A", "S"))
```

```
## [1] FALSE
```

Conditional Independence and the Markov Property

We use the graphical concepts in the previous section to reason about the data generating process we want to model. Specifically, the graphical concepts represent assumptions of **conditional independence** in the joint distribution over all the variables in the process.

Why we care about conditional independence in machine learning and causality

1. In machine learning, if a prediction target T is conditionally independent of some set of variables B given a set of variables A, then you should only add variables A to your predictive model. Adding B along with A would be overfitting.
2. Conditional independence gives us more compact representations of the probability distribution captured by our causal generative model. The more conditional independence statements are true, less parameter values need to be trained and stored.
3. Conditional independence is why causal models are most robust and transfer more easily across domains. Changes that affect one part of the model do not affect other parts.

Let's revisit the concepts that we learned in the previous section:

V-structure: We will start by talking about a simple example. Look at the figure below with three nodes. There is a grass that is either wet or dry, the sprinkler is either on or off and the weather is either rainy or not rainy. This network has a V-structure. In general, weather and sprinkler are independent from each other but when we condition on the child (grass), rain and sprinkler become dependent. If we know that the grass is wet, finding out it is not raining tells you the sprinkler must be on.

Markov blanket: A node is conditionally independent of all the other nodes in the system given its Markov blanket. This means that if you predict that node, given the nodes in the Markov blanket, adding any other predictors to your model is overfitting, because the Markov blanket contains all the information available about the node.

D-separation and the Markov property: We have a data generating process with a joint distribution P_X over all the variables ($\{X_1, X_2, ..., X_P\}$) in our DAG G. The global Markov property is a property of G. It means that all of the d-separations in G, correspond to conditional independence assumptions in P_X. When we build a causal model that has an underlying causal DAG, we generally assumes the DAG meets the global Markov property. Let's define Markov property: Given a DAG G and a joint distribution P_X, this distribution is said to satisfy,

(a) The **global Markov property** with respect to G, if,

$$X_i \perp\!\!\!\perp_G X_j | X_k \Rightarrow X_i \perp\!\!\!\perp X_j | X_k$$

This means that if two node are conditionaly d-separated in G, they are conditionaly independent with respect to their joint probability distribution.

(b) The **local Markov property** with respect to G, if each variable X_i is independent of its non-descendants given its parents, and

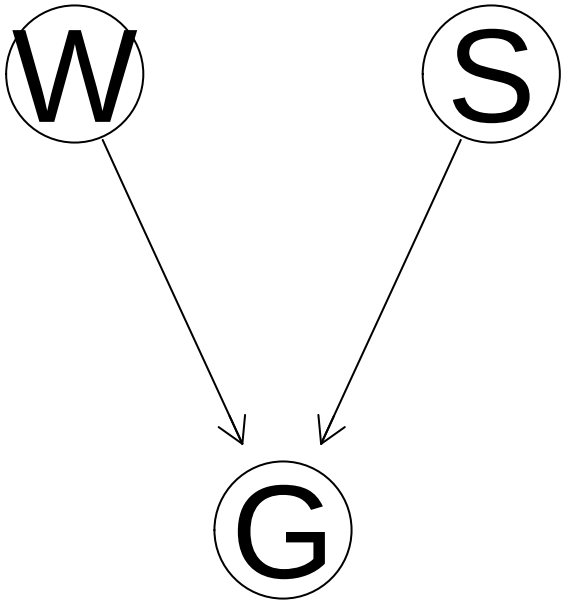


Figure 5: Figure 4: weather is dependent on sprinkler given grass

(c) The **Markov factorization property** with respect to G, if,

$$p(X) = p(X_1, X_2, ..., X_p) = \prod_{j=1}^{p} p(X_j | Pa_j^G)$$

Where Pa_j^G is the set of parents of node (variable) X_j. If a DAG has the Markov factorization property with respect to a distribution, then the distribution factorizes according to the ordering of the DAG.

In R, we can use the `dsep` function from bnlearn package to find whether two variables are dseparated given a set of variables. Here is an example from survey data. We want to check whether S is independent of T given O and R:

```
dsep(dag, "S", "T",c("O","R"))
```

```
## [1] TRUE
```

Now, let's check whether the global Markov property is satisfied in this example. We will use the survey data to check the Null hypothesis that S is independent of T given O and R. In R, we can use `ci.test` function from bnlearn to find conditional dependence.

```
surveyData <- read.table("../data/survey.txt", header = TRUE)
surveyData[] <- lapply(surveyData, function(x) as.factor(x)) #convert column types from character to fa
head(surveyData)
```

```
##       A     R    E   O S     T
## 1 adult   big high emp F   car
## 2 adult small  uni emp M   car
## 3 adult   big  uni emp F train
## 4 adult   big high emp M   car
## 5 adult   big high emp M   car
## 6 adult small high emp F train
```

```
ci.test("S","T",c("O","R"), data = surveyData)
```

```
##
##  Mutual Information (disc.)
##
## data:  S ~ T | O + R
## mi = 13.207, df = 8, p-value = 0.1049
## alternative hypothesis: true value is greater than 0
```

As you can see, the p-value is greater that 0.05 threshold, hence we do not reject the Null hypothesis and conclude that in fact, S is independent of T given O and R.

Faithfulness assumption: A distribution P_X is faithful to the DAG G, if,

$X_i \perp\!\!\!\perp X_j | X_k \Rightarrow X_i \perp\!\!\!\perp_G X_j | X_k$

In the simple example above, we can see that from the data distribution, S is independent of T given O and R (we checket it with `ci.test`) and hence in the network S is d-separated of T given O and R. So we have a faithful distribution.

Finally, let's compute the factorization of the survey data:

$p(X) = p(A, S, E, O, R, T) = p(A)p(S)p(E|A, S)p(O|E)p(R|E)p(T|O, R)$

In R, we can use the `modelstring` function from bnlearn package to find the factorization of a graphical model.

```
modelstring(dag)
```

```
## [1] "[A][S][E|A:S][O|E][R|E][T|O:R]"
```

The causal DAG as a representation of joint probability

Any DAG we might specify for this data represents a factorization of the joint probability distribution of the variables in this data. The DAG that aligns with our causal assumptions is just one of such factorizations That said, it is the most useful factorization because the factors correspond to independent causal mechanisms we assume to be invariant across data sets.

In this section we show how to add custom probability distributions to a DAG, as well as how to estimate the parameters of the conditional probability distribution using maximum likelihood estimation or Bayesian estimation.

Specifying the probability distributions on your own

Given the causal DAG, the joint probability distribution of the survey data variables factorizes as follows:

$Pr(A, S, E, O, R, T) = Pr(A)Pr(S)Pr(E|A, S)Pr(O|E)Pr(R|E)Pr(T|O, R)$.

```r
A.lv <- c("young", "adult", "old")
S.lv <- c("M", "F")
E.lv <- c("high", "uni")
O.lv <- c("emp", "self")
R.lv <- c("small", "big")
T.lv <- c("car", "train", "other")

A.prob <- array(c(0.3,0.5,0.2), dim = 3, dimnames = list(A = A.lv))
S.prob <- array(c(0.6,0.4), dim = 2, dimnames = list(S = S.lv))
E.prob <- array(c(0.75,0.25,0.72,0.28,0.88,0.12,0.64,0.36,0.70,0.30,0.90,0.10), dim = c(2,3,2), dimname
O.prob <- array(c(0.96,0.04,0.92,0.08), dim = c(2,2), dimnames = list(O = O.lv, E = E.lv))
R.prob <- array(c(0.25,0.75,0.2,0.8), dim = c(2,2), dimnames = list(R = R.lv, E = E.lv))
T.prob <- array(c(0.48,0.42,0.10,0.56,0.36,0.08,0.58,0.24,0.18,0.70,0.21,0.09), dim = c(3,2,2), dimname
cpt <- list(A = A.prob, S = S.prob, E = E.prob, O = O.prob, R = R.prob, T = T.prob)
```

```r
# custom cpt table
cpt
```

```
## $A
## A
## young adult   old
##   0.3   0.5   0.2
##
## $S
## S
##   M   F
## 0.6 0.4
##
## $E
## , , S = M
##
##       A
## E      young adult  old
##   high  0.75  0.72 0.88
##   uni   0.25  0.28 0.12
##
## , , S = F
##
##       A
## E      young adult old
##   high  0.64   0.7 0.9
##   uni   0.36   0.3 0.1
##
##
## $O
##       E
## O      high  uni
##   emp  0.96 0.92
##   self 0.04 0.08
##
## $R
##       E
## R      high uni
```

```
##   small 0.25 0.2
##   big   0.75 0.8
##
## $T
## , , R = small
##
##        O
## T        emp self
##   car   0.48 0.56
##   train 0.42 0.36
##   other 0.10 0.08
##
## , , R = big
##
##        O
## T        emp self
##   car   0.58 0.70
##   train 0.24 0.21
##   other 0.18 0.09
```

Now that we have defined both the causal DAG and the local distribution corresponding to each variable, we can combine them to form a fully-specified causal Bayesian network. We combine the DAG we stored in `dag` and a list containing the local distributions, which we will call `cpt`, into an object of class **bn.fit** called bn.

```
# fit cpt table to network
bn <- custom.fit(dag, cpt)
```

Estimating parameters of conditional probability tables

So far, we have assumed to know both the causal DAG and the parameters of the local distributions defining the causal BN.

This is a plausible scenario if we indeed assume the DAG is causal. In the causal case, each CPT represents an independent mechanism that we assume is fairly invariant. So we could encode our prior knowledge about these mechanisms directly in the form of parameter values.

However, in the context of machine learning, most of the time we are going to learn these parameter values from data.

Let's read the survey data:

```
survey <- read.table("../data/survey.txt", header = TRUE)
survey[] <- lapply(survey, function(x) as.factor(x)) #convert column types from character to factors
head(survey)
```

```
##       A     R    E   O S     T
## 1 adult   big high emp F   car
## 2 adult small  uni emp M   car
## 3 adult   big  uni emp F train
## 4 adult   big high emp M   car
## 5 adult   big high emp M   car
## 6 adult small high emp F train
```

In the case of this survey, and of discrete causal BNs in general, the parameters to estimate are the conditional probabilities in the local distributions. They can be estimated, for example, by the corresponding empirical frequencies in the data set, e.g.,

$$\hat{Pr}(O = emp|E = high) = \frac{\hat{Pr}(O{=}emp, E{=}high)}{\hat{Pr}(E{=}high)} = \frac{\text{number of observations for which O = emp and E = high}}{\text{number of observations for which E = high}}$$

This yields the classic frequentist and maximum likelihood estimates. In bnlearn, we can compute them with the **bn.fit** function. **bn.fit** complements the **custom.fit** function we used in the previous section; the latter constructs a BN using a set of custom parameters specified by the user, while the former estimates the same from the data.

```
bn.mle <- bn.fit(dag, data = survey, method = "mle")
bn.mle
```

```
##
##   Bayesian network parameters
##
##   Parameters of node A (multinomial distribution)
##
## Conditional probability table:
##  adult   old young
## 0.472 0.208 0.320
##
##   Parameters of node S (multinomial distribution)
##
## Conditional probability table:
##      F     M
## 0.402 0.598
##
##   Parameters of node E (multinomial distribution)
##
## Conditional probability table:
##
## , , S = F
##
##       A
## E          adult       old     young
##   high 0.6391753 0.8461538 0.5384615
##   uni  0.3608247 0.1538462 0.4615385
##
## , , S = M
##
##       A
## E          adult       old     young
##   high 0.7194245 0.8923077 0.8105263
##   uni  0.2805755 0.1076923 0.1894737
##
##
##   Parameters of node O (multinomial distribution)
##
## Conditional probability table:
##
##       E
## O            high        uni
```

```
##   emp  0.98082192 0.92592593
##   self 0.01917808 0.07407407
##
##   Parameters of node R (multinomial distribution)
##
## Conditional probability table:
##
##        E
## R            high       uni
##   big   0.7178082 0.8666667
##   small 0.2821918 0.1333333
##
##   Parameters of node T (multinomial distribution)
##
## Conditional probability table:
##
## , , R = big
##
##        O
## T              emp       self
##   car   0.58469945 0.69230769
##   other 0.19945355 0.15384615
##   train 0.21584699 0.15384615
##
## , , R = small
##
##        O
## T              emp       self
##   car   0.54700855 0.75000000
##   other 0.07692308 0.25000000
##   train 0.37606838 0.00000000
```

Note that we assume we know the structure of the DAG, so `dag` is an input of **bn.fit** function.

As an alternative, we can also do Bayesian estimation of the parameters. This will provide the maximum a posterior point values of the posterior. The Bayesian modeling depends on the data type, in the discrete case, it makes use of the Dirichlet conjugate prior.

To use Bayesian estimation, set the `method` argument of `bn.fit` must be set to `"bayes"`.

```
bn.bayes <- bn.fit(dag, data = survey, method = "bayes", iss = 10)
```

The estimated posterior probabilities are computed from an uniformed prior over each conditional probability table. `iss` is an optional argument, whose name stands for imaginary sample size (also known as equivalent sample size). It determines how much weight is assigned to the prior distribution compared to the data when computing the posterior. The weight is specified as the size of an imaginary sample supporting the prior distribution.

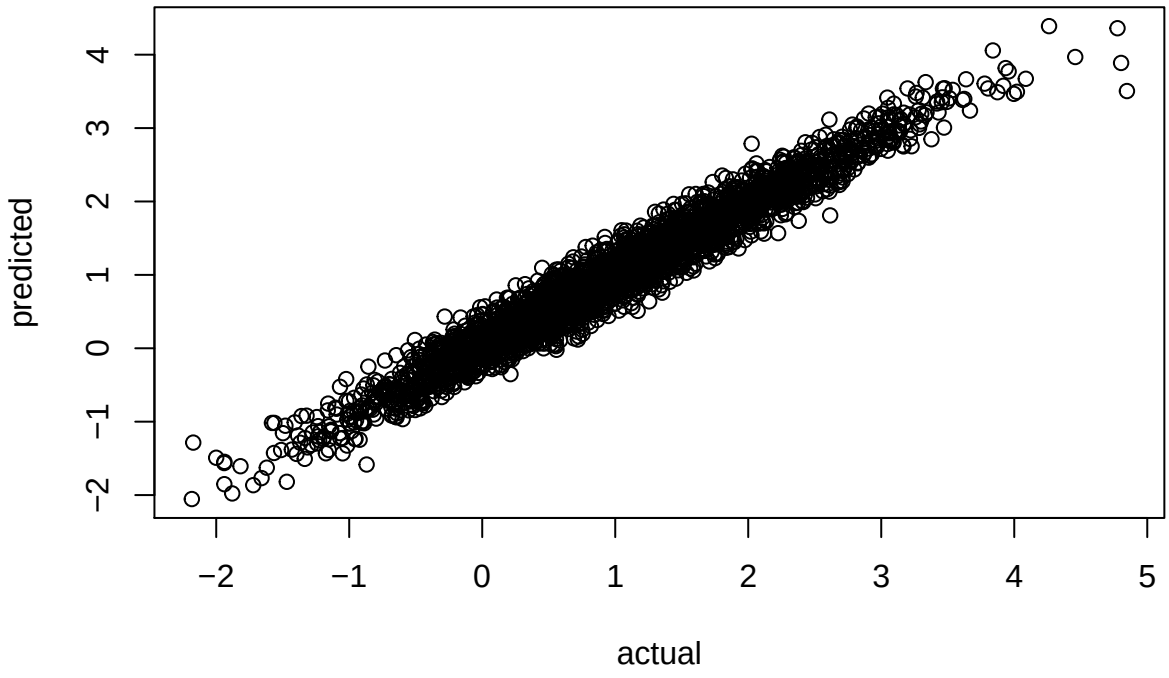
Predicting the value of a latent variable

After we've train the model, there will often be cases where we need to apply it to data with unobserved (latent) variables.

The code below demonstrates fitting a model on the first 2000 points of a sample data set, and then predicting the value of the variable "A" on the remaining values of the data set.

```r
# predicting a variable in the test set.
model <- bn.fit(
    model2network("[A][B][E][G][C|A:B][D|B][F|A:D:E:G]"),
    gaussian.test[1:2000, ]
)
test <- gaussian.test[2001:nrow(gaussian.test),]
predicted <- predict(
    model,
    node = "A",
    data = test,
    method = "bayes-lw")

plot(test$A, predicted, xlab = "actual", ylab = "predicted")
```



`bayes-lw` is the inference method used to infer the latent value. Specifically, `bnlearn` uses a fairly light-weight inference technique called likelihood weighting.

Note that `bnlearn` can be combined with R libraries that provide better inference algorithms for graphical models, such as gRain. However, for more powerful infernce, you might consider reimplementing your model in a probabilistic programming language such as Stan.

Contact me

If you have any questions about the bnlearn tutorial please email me at `mohammadtaheri.s@northeastern.edu` (Sara Taheri) . I will be happy to schedule a zoom call with you to answer your questions.

Some Useful References

1. Scutari, Marco, and Jean-Baptiste Denis. Bayesian networks: with examples in R. Chapman and Hall/CRC, 2014.

2. Peters, Jonas, Dominik Janzing, and Bernhard Schölkopf. Elements of causal inference: foundations and learning algorithms. MIT press, 2017.