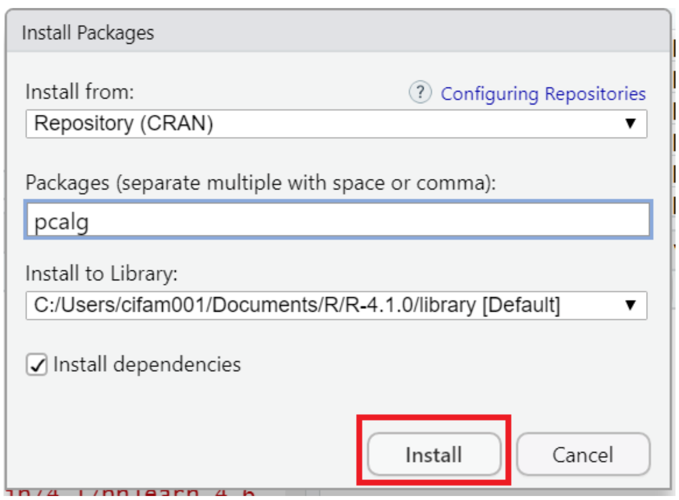
Practical 3: Bayesian Networks

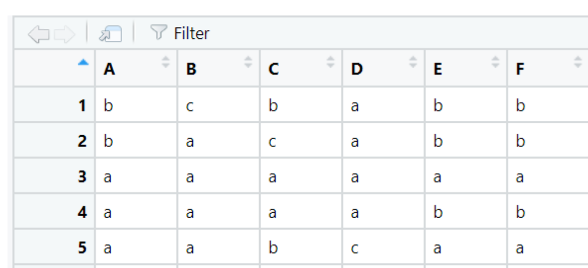
I. Learning Bayesian network structure from data – Search and Score approaches

1. Start R or Rstudio (Recommended).
2. Install `bnlearn` and `pcalg` packages.
   a. Select Tools → install packages → specify the name of the package you want to
      install.
   b. Tick the “Install dependencies” box to install all the dependent packages.
   c. Click “Install”.



3. Run the following codes to learn the Bayesian network structure for the dataset learning. Test using the Hill-Climbing algorithm.

```
library(bnlearn)
 #call an example built-in dataset
data(learning.test)
View(learning.test)
```



	A	B	C	D	E	F
1	b	c	b	a	b	b
2	b	a	c	a	b	b
3	a	a	a	a	a	a
4	a	a	a	a	b	b
5	a	a	b	c	a	a

train model by calling `hc()` function. You can change the way of scoring with the parameter `score`

```
network1=hc(learning.test
            ,score="bde")
network1
```

```
> network1

  Bayesian network learned via Score-based methods

  model:
   [A][C][F][B|A][D|A:C][E|B:F]
  nodes:                                 6
  arcs:                                  5
    undirected arcs:                     0
    directed arcs:                       5
  average markov blanket size:           2.33
  average neighbourhood size:            1.67
  average branching factor:              0.83

  learning algorithm:                    Hill-Climbing
  score:                                 Bayesian Dirichlet (BDe)
  graph prior:                           Uniform
  imaginary sample size:                 1
  tests used in the learning procedure:  40
  optimized:                             TRUE
```

you can also use tabu search instead of hc, by calling `tabu()` function. You can change the way of scoring with the parameter `score`

```
network2=tabu(learning.test
              ,score="bde")
network2
```

```
> network2

  Bayesian network learned via Score-based methods

  model:
   [A][C][F][B|A][D|A:C][E|B:F]
  nodes:                                 6
  arcs:                                  5
    undirected arcs:                     0
    directed arcs:                       5
  average markov blanket size:           2.33
  average neighbourhood size:            1.67
  average branching factor:              0.83

  learning algorithm:                    Tabu Search
  score:                                 Bayesian Dirichlet (BDe)
  graph prior:                           Uniform
  imaginary sample size:                 1
  tests used in the learning procedure:  135
  optimized:                             TRUE
```

More information of `bnlearn` package can be found at www.bnlearn.com

II. Learning Bayesian network structure from data – Constraint based approaches

In this section, we use the PC algorithm from the pcalg package to learn the Bayesian network structure from data. Please refer to the user manual of pcalg for more details https://cran.r-project.org/web/packages/pcalg/pcalg.pdf

1. **Using numeric data**

Load pcalg package and a numeric dataset.

```
library(pcalg)
## Load predefined data
data(gmG)
gmG8$x[1:5,]
```

This if how this dataset looks like:

```
> gmG8$x[1:5,]
         Author         Bar       Ctrl        Goal         V5         V6        V7         V8
[1,]  1.5763995 -0.20365553  0.9236034  1.43909630 -1.4088564 -1.9879970 0.1050979  0.6496531
[2,]  0.0271247  1.55034413  1.6974502  0.49585726  0.3799821  1.1915730 0.8068063  0.7353409
[3,] -0.5751062  0.03851787 -0.2696420 -0.79906964  0.7328964 -0.3046606 1.7930543  0.4269413
[4,]  0.6012083  0.17049269 -0.1608637 -0.09930314 -0.9849444  1.9901392 3.7865834 -0.9596913
[5,]  0.2756189 -0.99633800 -0.7162193 -1.32219298  0.9042090 -0.1422637 0.3165154  0.4685432
```

retrieve the number of observations and variable names

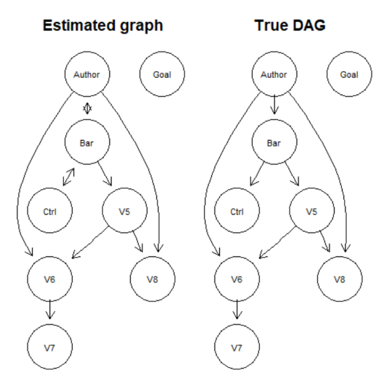
```
n <- nrow (gmG8$ x)
V <- colnames(gmG8$ x) # labels aka node names
```

Estimate CPDAG. Use `indepTest = gaussCItest` for numerical (continuous) data.

```
pc.fit <- pc(suffStat = list(C = cor(gmG8$x), n = n),
indepTest = gaussCItest, alpha=0.01, labels = V)
```

you can compare inferred graph and (provided) true graph

```
if (require(Rgraphviz)) {
## show estimated graph
par(mfrow=c(1,2))
plot(pc.fit
     , main = "Estimated graph")
plot(gmG8$g
     , main = "True DAG")
}
```



2. **Using discrete data**

Load data and retrieve variable's names

```
## Load data
data(gmD)
gmD$x[1:5,]
V <- colnames(gmD$x)
```

```
> gmD$x[1:5,]
  X1 X2 X3 X4 X5
1  2  0  0  2  1
2  2  1  1  2  1
3  1  0  1  3  0
4  1  0  2  2  1
5  1  0  0  2  1
```

Define sufficient statistics. In this case since data is discrete you need to include how many different values can take each variable. You do this by setting the parameter `nlev`

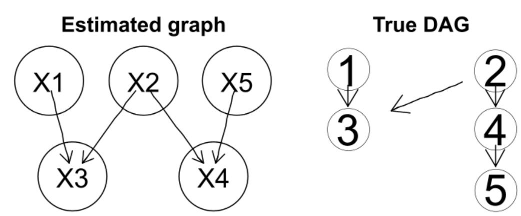
```
## define sufficient statistics
suffStat <- list(dm = gmD$x, nlev = c(3,2,3,4,2), adaptDF = FALSE)
```

Estimate the graph. Use `indepTest = disCItest` for discrete data.

```
pc.D <- pc(suffStat, indepTest = disCItest, alpha = 0.01, labels = V, verbose = TRUE)
```

Compare the graphs

```
#compare the graphs
if (require(Rgraphviz)) {
## show estimated CPDAG
par(mfrow = c(1,2))
plot(pc.D, main = "Estimated graph")
plot(gmD$g, main = "True DAG")}
```



3. **Using binary data**

Load data and retrieve variable's names

```
## Load binary data
data(gmB)
gmB$x[1:5,]
V <- colnames(gmB$x)
```

```
> gmB$x[1:5,]
     V1 V2 V3 V4 V5
[1,]  0  0  1  0  1
[2,]  1  0  0  0  0
[3,]  1  1  1  0  1
[4,]  0  0  0  0  0
[5,]  0  1  0  0  1
```

Estimate the graph. Use `indepTest = binCItest` for binary data. You can retrieve a summary of the inference by calling the variable.

```
## estimate the structure
pc.B <- pc(suffStat = list(dm = gmB$x, adaptDF = FALSE),
indepTest = binCItest, alpha = 0.01, labels = V, verbose = TRUE)
pc.B
```

```
> pc.B
Object of class 'pcAlgo', from Call:
pc(suffStat = list(dm = gmB$x, adaptDF = FALSE), indepTest = binCItest,
    alpha = 0.01, labels = V, verbose = TRUE)
Number of undirected edges:  2
Number of directed edges:    3
Total number of edges:       5
```

Compare the graphs

```
if (require(Rgraphviz)) {
## show estimated CPDAG
plot(pc.B, main = "Estimated CPDAG")
plot(gmB$g, main = "True DAG")
}
```