Tarea sobre redes bayesianas

Cedric Prieels

2Diciembre 2016

Construcción del modelo

Primero vamos a leer los datos, instalar los paquetes y pintar por pantalla las primeras lineas del fichero de datos llamado *meteoro.txt*, que contienen datos meteorologicos de diferentes tipos.

```
rm(list=ls())
setwd('/Users/ced2718/Documents/Universite/Modelizacion/Redes_bayesianas')

if (!require(bnlearn)) install.packages("bnlearn")
```

```
## Loading required package: bnlearn
```

```
##
## Attaching package: 'bnlearn'
```

```
## The following object is masked from 'package:stats':
##
##     sigma
```

```
if (!require(RBGL)) {
      source("http://bioconductor.org/biocLite.R")
      biocLite() # Activa el conjunto de librerías que permiten realizar la instalacion remota.
      biocLite("RBGL")
}
```

```
## Loading required package: RBGL
```

```
## Loading required package: graph
```

```
## Loading required package: BiocGenerics
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
```

```
## The following object is masked from 'package:bnlearn':
##
##     score

## The following objects are masked from 'package:stats':
##
##     IQR, mad, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, cbind, colnames,
##     do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, lengths, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, setdiff,
##     sort, table, tapply, union, unique, unsplit, which, which.max,
##     which.min

##
## Attaching package: 'graph'

## The following objects are masked from 'package:bnlearn':
##
##     degree, nodes, nodes<-
```

```
# Paquete gRain
if (!require(gRain)) install.packages("gRain")
```

```
## Loading required package: gRain

## Loading required package: gRbase

##
## Attaching package: 'gRbase'

## The following objects are masked from 'package:bnlearn':
##
##     children, parents
```

```
library(bnlearn)

data <- read.table("meteoro.txt", header=T)
head(data)
```

```
##   lluvia nieve granizo tormenta niebla rocio escarcha nieveSuelo neblina
## 1      s     n       n        n      n     n        n          n       n
## 2      s     n       n        n      n     n        n          n       n
## 3      s     n       n        s      n     n        n          n       n
## 4      s     n       n        n      n     n        n          n       n
## 5      s     n       s        s      n     n        n          n       n
## 6      s     n       s        s      n     n        n          n       n
```

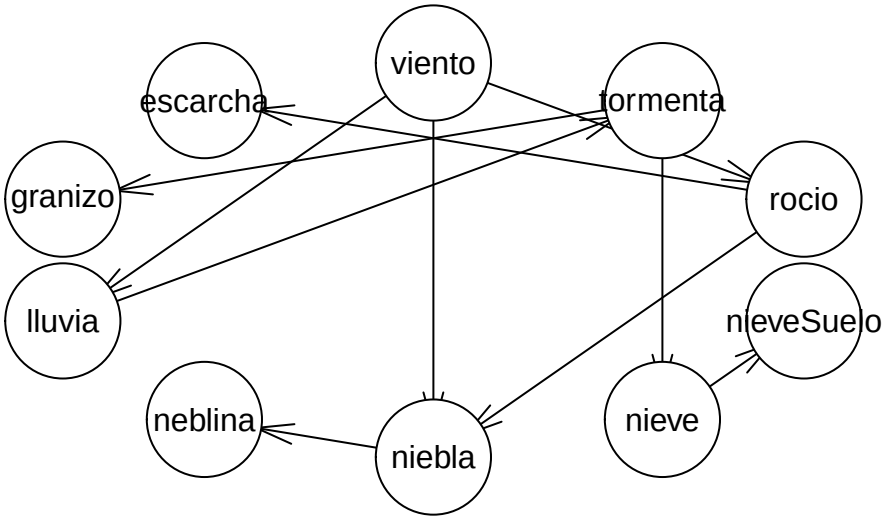
```
##   viento
## 1      s
## 2      s
## 3      s
## 4      s
## 5      s
## 6      s
```

A partir del grafo dado por el experto, podemos primero escribir a mano la factorización de este problema, mirando a las dependencias entre los nodos del grafo.

"[viento][lluvia|viento][rocio|viento][niebla|viento:rocio][escarcha|rocio][neblina|niebla][tormenta|lluvia][granizo|tormenta][nieve|t

Una vez escrita la factorización, creamos nuestro DAG fácilmente, usando la función *model2network*.

```
model.string <- "[viento][lluvia|viento][rocio|viento][niebla|viento:rocio]
[escarcha|rocio][neblina|niebla][tormenta|lluvia][granizo|tormenta]
[nieve|tormenta][nieveSuelo|nieve]"
dag <- model2network(model.string)
plot(dag)
```



Ahora podemos crear la red bayesiana que contiene todas las probabilidades de cada nodo de nuestra red. Podemos después de haber obtenido esta red calcular muchas probabilidades distintas : por ejemplo, la probabilidad de que haya viento o bien de que haya lluvia (y ya sabemos por los datos del experto que esta probabilidad viene condicionada por el resultado de si hay viento, o no).

```
bn <- bn.fit(dag, data = data, method = "mle")
#Probabilidad de que haya viento
bn$viento
```

```
##
##   Parameters of node viento (multinomial distribution)
##
## Conditional probability table:
##          n         s
## 0.8557517 0.1442483
```

```
#Probabilidad de que haya lluvia
bn$lluvia
```

```
##
##   Parameters of node lluvia (multinomial distribution)
##
## Conditional probability table:
##
##       viento
## lluvia         n         s
##      n 0.4960882 0.1371308
##      s 0.5039118 0.8628692
```

Sin utilizar el DAG, tendríamos 1023 parámetros en nuestro modelo, porque tenemos 10 variables que pueden tomar dos valores distintos, y una ligadura. Con el DAG, podemos calcular fácilmente este mismo número de parámetros, que tiene que ser menor (si no, usar una red bayesiana no tendría mucho sentido). En este caso, vemos que nuestro modelo tiene 21 parámetros.

```
nparams(bn)
```

```
## [1] 21
```

Podemos ahora obtener las tablas de probabilidades condicionales asociadas a los nodos granizo y niebla. Vemos que la tabla del nodo granizo es mucho más sencilla que la otra, porque el nodo granizo solo tiene una conexión con tormenta, mientras que el nodo niebla tiene como "padres" el rocio y el viento. Necesitamos entonces una matriz en 3 dimensiones para representar su tabla de probabilidad condicional.

```
bn$granizo
```

```
##
##   Parameters of node granizo (multinomial distribution)
##
## Conditional probability table:
##
##        tormenta
## granizo           n           s
##       n 0.990262902 0.746341463
##       s 0.009737098 0.253658537
```

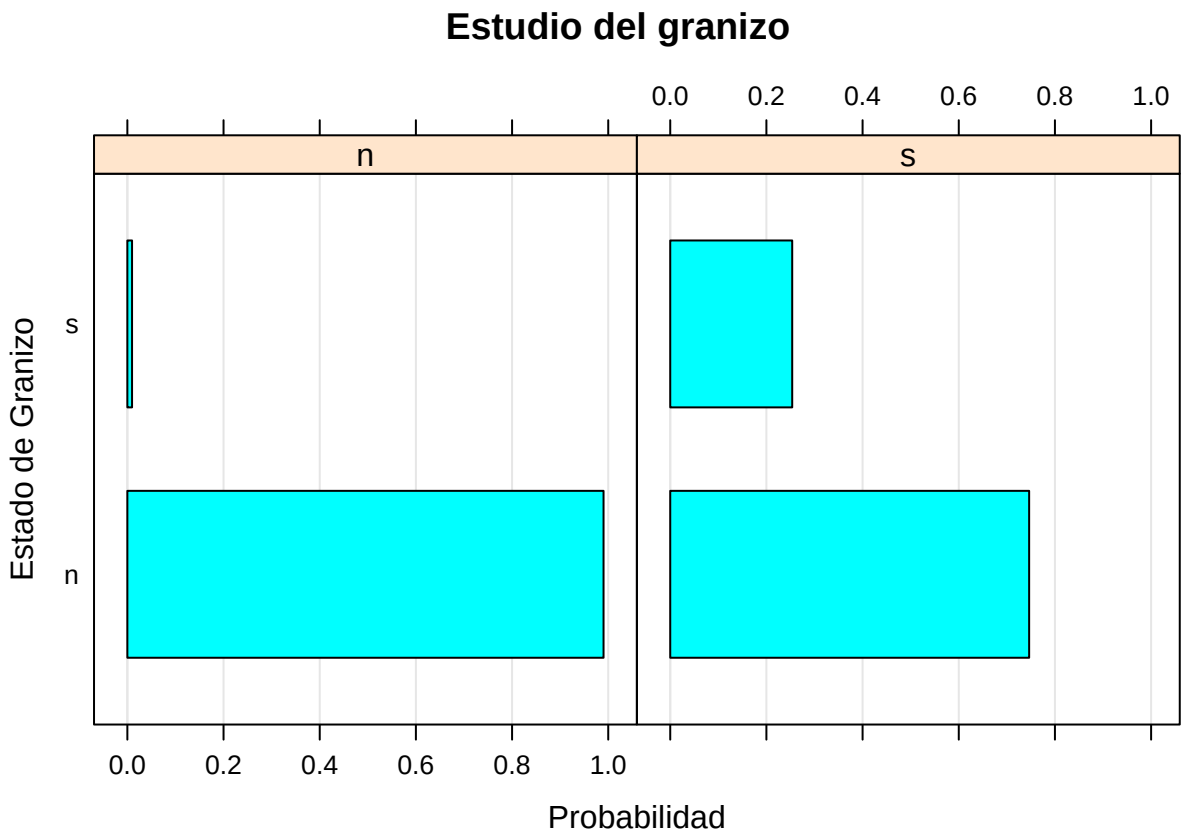
```
bn$niebla
```

```
##
##   Parameters of node niebla (multinomial distribution)
##
## Conditional probability table:
##
## , , viento = n
##
##       rocio
## niebla           n           s
##      n 0.946455131 0.855345912
```

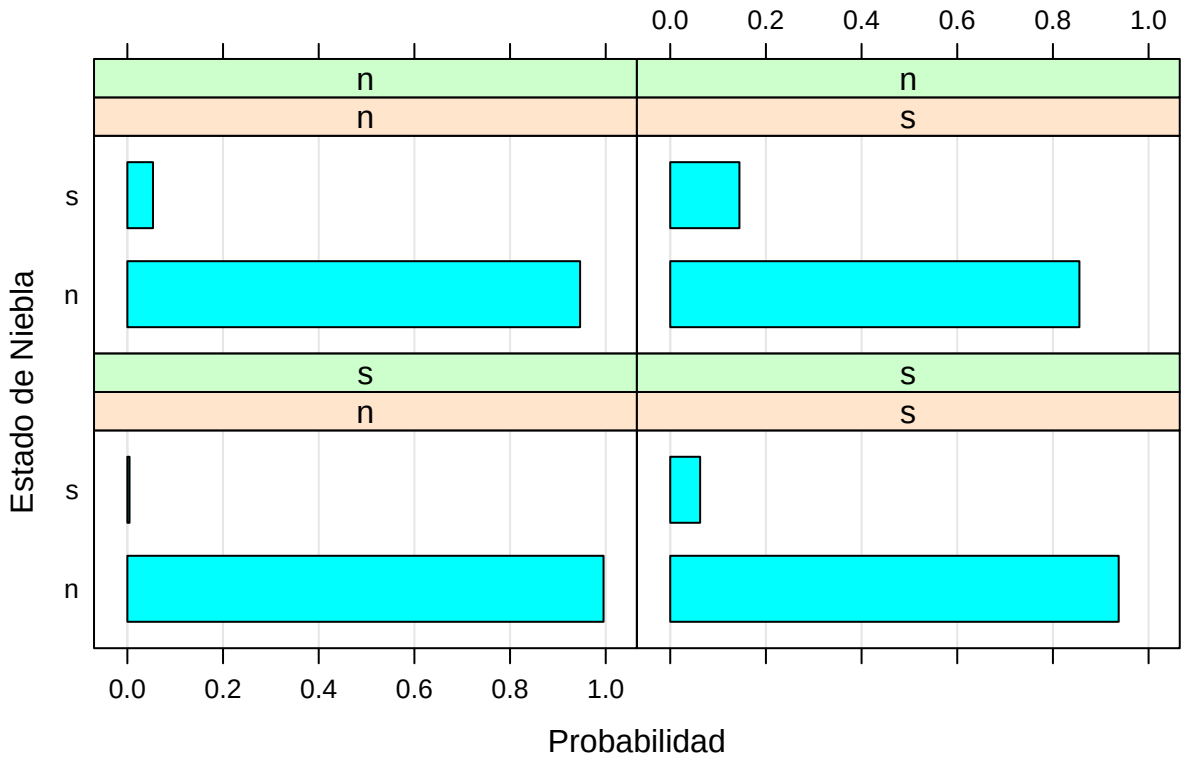
```
##       s 0.053544869 0.144654088
##
## , , viento = s
##
##       rocio
## niebla           n           s
##       n 0.995475113 0.937500000
##       s 0.004524887 0.062500000
```

Podemos ahora representa la información de otra forma, usando tabla en sendos gráficos multivariantes.

```
bn.fit.barchart(bn$granizo, main = "Estudio del granizo",
                xlab = "Probabilidad", ylab = "Estado de Granizo")
```




```
bn.fit.barchart(bn$niebla, main = "Estudio de la niebla",
                xlab = "Probabilidad", ylab = "Estado de Niebla")
```



Inferencia

Ahora, se puede realizar la inferencia : una vez creada nuestra red bayesiana, se puede calcular la probabilidad de cualquiera variable o conjunto de variables condicionadas, dada calquiera evidencia.

La d-separación es el estudio de la dependencia entre nodos de la red bayesiana que hemos obtenido. Podemos decir que dos nodos están d-separados (o solamente separados) si la información entre los dos nodos está bloqueada y condicionada por la información de otro nodo. Esto significa que dos nodos pueden estar independientes pero pasar a ser dependientes cuando una tercera variable entra en cuenta. Este tipo de dependencia se representa con estrcturas-V en el grafo de la red. Podemos usar este concepto para verificar algunas afirmaciones.

- La nieve y el granizo son fenómenos independientes : es falso. Podemos ver en el dag una estructura de tipo V clara formada por los tres nodos siguientes : granizo, nieve y tormenta. Esto significa que la nieve y el granizo son independientes, pero pasan a ser dependientes cuando la tormenta entra en cuenta. Lo veremos en la segunda afirmación.

```
dsep(dag, x = "nieve", y = "granizo")
```

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

- La nieve y el granizo son fenómenos independientes dado que haya habido tormenta : verdadero.

```r
dsep(dag, x = "nieve", y = "granizo", z="tormenta")
```

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

Lo mismo pasa con las dos observaciones siguientes, aunque el fenomeno esté un poco menos claro en este caso porque existen muchos nodos intermedios entre los nodos que consideramos. Las afirmaciones siguientes se pueden deducir más facilmente una vez pintada el arból de cliques de la red (punto siguiente), donde se va claramente que el nodo tormenta separa la red en dos partes distintas. Tendrá por lo tanto por supuesto una influencia en la d-separación de dos nodos, si cada nodo está en un lado de esta diagrama de cliques. – La nieve en el suelo y la neblina son fenómenos independientes : falso.

```r
dsep(dag, x = "nieveSuelo", y = "neblina")
```

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

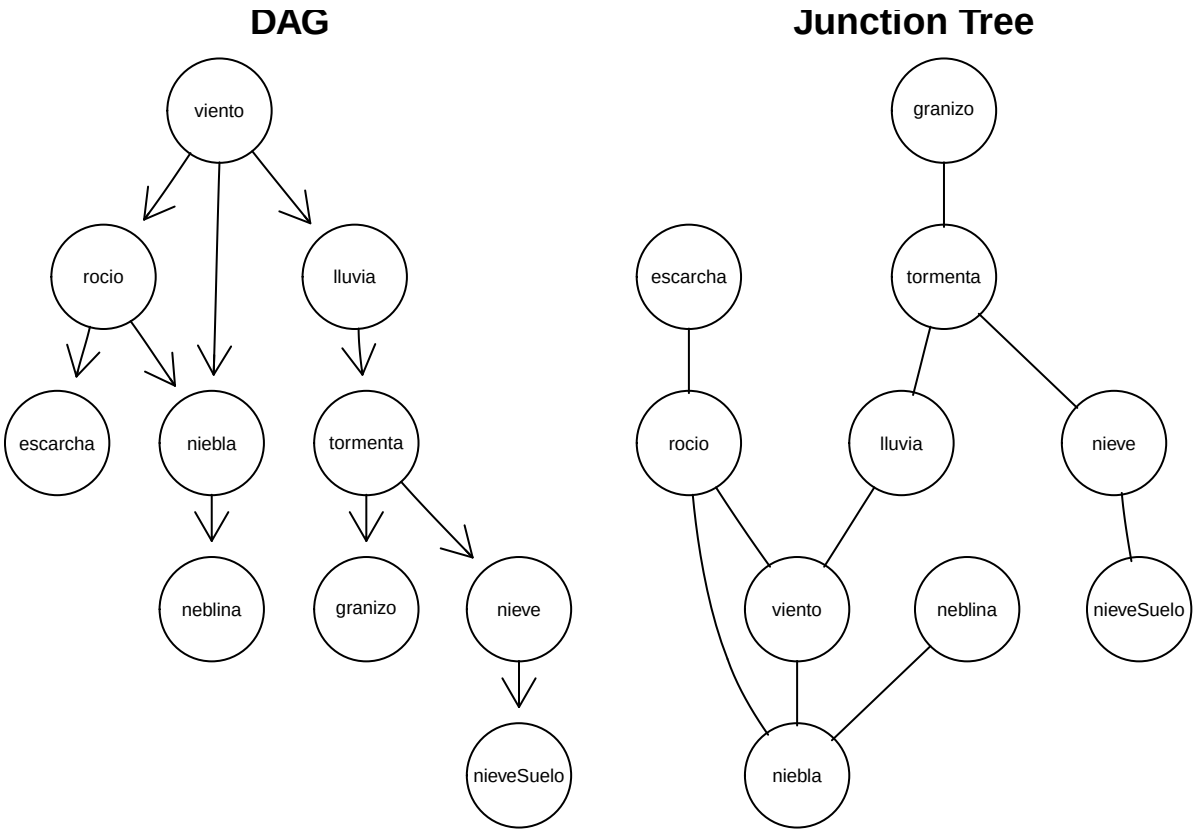
– La nieve en el suelo y la neblina son fenómenos independientes dado que haya habido tormenta : verdadero.

```r
dsep(dag, x = "nieveSuelo", y = "neblina", z="tormenta")
```

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

Con las dos primeras y las dos últimas afirmaciones vemos claramente que dos nodos pueden ser independientes pero dependientes cuando conocemos el valor de un tercer nodo de la red. Esto ilustra perfectamente el fenómeno d-separación.

```r
#Setup de la segunda parte de inferencia para calcular probabilidades condicionadas
#Creamos primero el "junction tree" de nuestro problema
library(gRain)
junction <- compile(as.grain(bn))
par(mfrow = c(1,2))
plot(as.grain(bn))
title("DAG")
plot(compile(junction))
title("Junction Tree")
```



Crear este diagrama de cliques (cunjontos maximales de variables dependientes) es un paso previo necesario para poder propagar una evidencia en la red bayesiana. Ahora, si se supone que sabemos que en un día dado se han producido tormentas, podemos calcular cómo afecta este hecho a la probabilidad de que se produzcan algunos fenómenos meteorológicos.

– Que llueva : hay una probabilidad de 0,96 de que haya lluvía, sabiendo que se ha producido una tormenta.

```
#Queremos calcular P(lluvia = s|tormenta = s)
jsex <- setEvidence(junction, nodes = "tormenta", states = "s")
querygrain(jsex, nodes = "lluvia")$lluvia
```

```
## lluvia
##          n          s
## 0.03902439 0.96097561
```

– Que haya rachas de viento superiores a 50 km/h : dado que se haya producido una tormenta, hay una probabilidad igual a 0,22 de ver vientos superiores a 50 km/h.

```
querygrain(jsex, nodes = "viento")$viento
```

```
## viento
##         n         s
## 0.7830167 0.2169833
```

– Que llueva y que además las rachas de viento superen los 50 Km/h : vemos en la tabla creada que la probabilidad de observar a la vez lluvia y viento cuando hay una tormenta vale 0.21.

```r
querygrain(jsex, nodes = c("lluvia", "viento"), type = "joint")
```

```
##       lluvia
## viento          n         s
##      n 0.037287003 0.7457297
##      s 0.001737387 0.2152459
```

– A partir de la información revelada por la red bayesiana, sabiendo que un día se producen tormentas: ¿Hay mayor probabilidad de que llueva cuando se producen tormentas que cualquier otro día?

```r
querygrain(junction, nodes = "lluvia")$lluvia
```

```
## lluvia
##         n         s
## 0.4443092 0.5556908
```

Se ve claramente que en un día calquiera, la probabilidad de tener lluvía vale 0.56 mientras que en un día de tormenta, esta probabilidad sube al 0.96. Esto confirma la intuición.

¿Aumenta o disminuye la probabilidad de tener rachas de viento mayores de 50 Km/h cuando se produce tormenta? ¿Cuánto?

```r
querygrain(junction, nodes = "viento")$viento
```

```
## viento
##         n         s
## 0.8557517 0.1442483
```

En este caso, vemos que cuando hay una tormenta, la probabilidad de ver viento es igual al 0.21 mientras que en calquier día, la probabilidad vale 0.14.

Aprendizaje estructural

También existen métodos para crear directamente el DAG a partir de los datos (métodos datadriven), sin tener que definir nosotros la factorisación de nuestro problema. En particular, en clase vimos el método de hill-climbing y sabemos que existe el algoritmo denominado Tabu search, que nos permite llegar al mismo resultado usando otro método. También existen métodos "mixtos", que permiten crear los DAG solos pero con informaciones complementaria que elegimos nosotros. Esta información complementaria que damos puede tener dos formatos : whitelist (cuando sabemos que un enlace entre nodos tiene que existir) y blacklist (el caso contrario, sabemos que un enlace no puede existir). Estudiamos también lo que son los scores y como nos pueden ayudar, porque penalizan el número de parámetros del modelo y tienen en cuenta las relaciones entre parámetros. Nos permiten entonces comparar modelos directamente.

Vamos primero a evaluar la significancia de los arcos dibujados por nosotros en el DAG precediente, utilizando el estadístico chi cuadrado. Esto nos permite determinar si nuestra factorización es buena, o bien si hay por ejemplo algún arco no-significativo que el experto nos dio pero que no parece significativo con nuestros datos. Vamos también a ver los pares de nodos que presentan una relación de dependencia más fuerte.

```r
#Estudio de la significancia de los arcos
arc.strength(dag, data = data, criterion = "x2")
```

```
##        from         to      strength
## 1    viento     lluvia  5.887107e-48
## 2    viento      rocio  1.730098e-23
## 3    viento     niebla  1.591451e-05
## 4     rocio     niebla  2.087399e-17
## 5     rocio   escarcha  2.106559e-05
## 6    niebla    neblina  1.300969e-27
## 7    lluvia   tormenta  1.707081e-33
## 8  tormenta    granizo 3.189229e-104
## 9  tormenta      nieve  1.051550e-13
## 10    nieve nieveSuelo 2.515328e-101
```

Vemos directamente con esta función la fuerza de los diferentes arcos de nuestra red. Por ejemplo, vemos que el arco entre tormenta y lluvia tiene una "fuerza" muy pequeña, mientras que el enlace entre el viento y la niebla es muy significativo. Hay que estar cuidado : una fuerza pequeña no quiere decir que el enlace no tiene que existir, también puede significar que falta algo de estadística.

Ahora podemos comparar el score global (llamado el BIC) del DAG creado usando los dos métodos datadriven diferentes que vimos : el hill-climbing y el Tabu search, para ver que DAG construido es el mejor.

```r
hillClimbing <- hc(data, debug = F)
hcScore <- bnlearn::score(hillClimbing, data = data, type = "bic")
hcScore
```

```
## [1] -9366.75
```

```r
tabuSearch <- tabu(data, debug = F)
tabuScore <- bnlearn::score(tabuSearch, data = data, type = "bic")
tabuScore
```

```
## [1] -9365.175
```

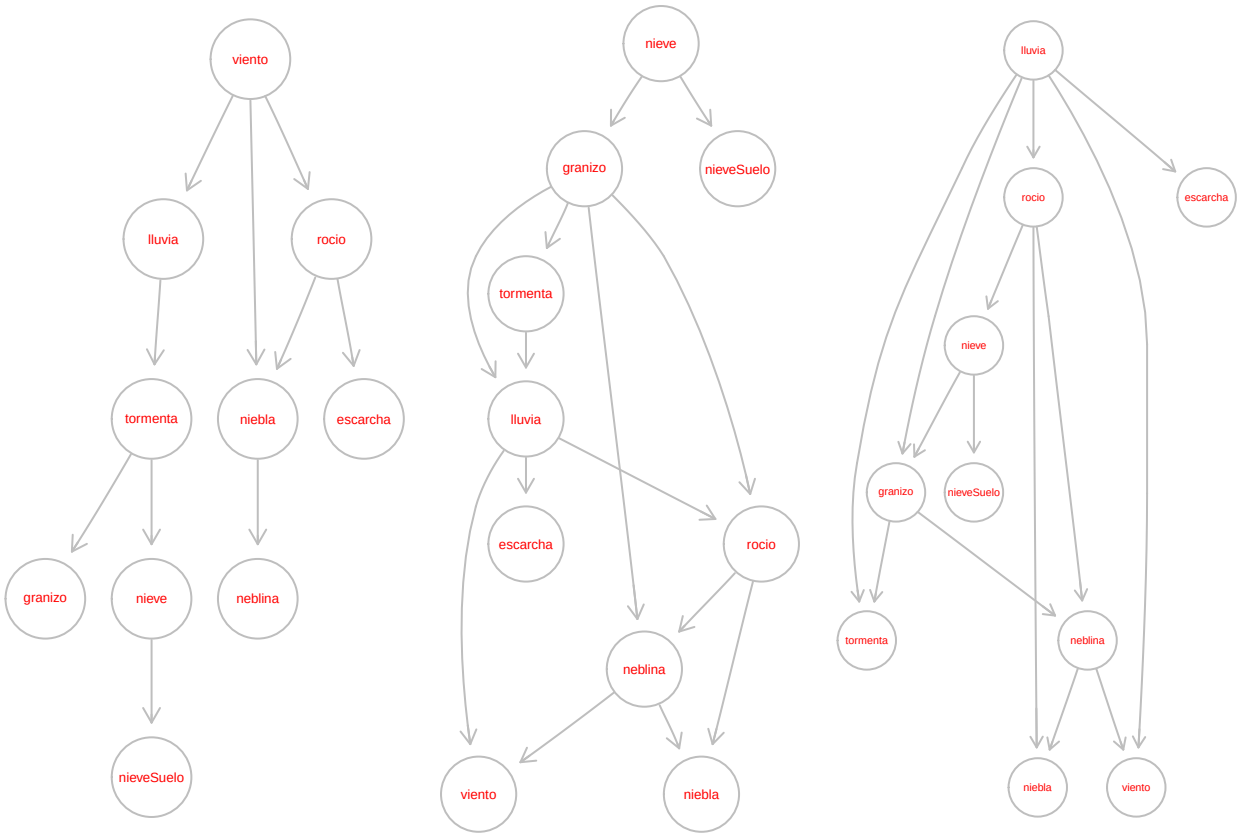
Vemos que los dos métodos devuelven un score BIC más o menos similar, pero que el método tabuSearch es un poco mejor, porque tiene un score un poco más alto. Podemos comparar los dos DAG que acabamos de obtener con el DAG que creamos siguiendo las recomendaciones del experto, pintando los tres utilizando la función graphviz.plot

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

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

```r
par(mfrow=c(1,3))

#Primer DAG
hlight <- list(nodes = nodes(dag), arcs = arcs(dag), col = "grey", textCol = "red")
pp <- graphviz.plot(dag, highlight = hlight)
#DAG obtenido por hillClimbing
hlight <- list(nodes = nodes(hillClimbing), arcs = arcs(hillClimbing), col = "grey", textCol = "red")
```

```
pp <- graphviz.plot(hillClimbing, highlight = hlight)
#DAG obtenido por tabuSearch
hlight <- list(nodes = nodes(tabuSearch), arcs = arcs(tabuSearch), col = "grey", textCol = "red")
pp <- graphviz.plot(tabuSearch, highlight = hlight)
```



A la vista de estos últimos plots, vemos algunas diferencias entre los diferentes métodos. Por ejemplo, el DAG obtenido por el experto no tiene enlace entre la lluvia y la escarcha, mientras que los otros dos DAG lo tienen. Como los dos últimos métodos son datadriven, se puede en general fiar más de ellos (pero siempre está bien comparar los resultados datadriven con el conocimiento de un experto).

Ahora volvemos a crear los DAG usando los métodos de hill-climbing y tabu, poniendo algunos parámetros en la whitelist y en la blacklist. Por ejemplo, imponemos que los arcos viento –> lluvia, tormenta –> granizo y nieve –> nieveSuelo existen mientras que empedimos la existencia de los enlaces entre neblina y granizo, y entre niebla y tormenta.

```
whitelist <- matrix(c("viento", "lluvia",
                      "tormenta", "granizo",
                      "nieve", "nieveSuelo"),
                    ncol = 2,byrow = TRUE,
                    dimnames = list(NULL, c("from", "to")))
blacklist <- matrix(c("neblina", "granizo",
                      "granizo", "neblina",
                      "niebla", "tormenta",
                      "tormenta", "niebla"),
                    ncol = 2, byrow = TRUE, dimnames = list(NULL, c("from", "to")))

#Hill-climbing
```

```r
hillClimbing2 <- hc(data, whitelist = whitelist, blacklist = blacklist)
hcScore2 <- bnlearn::score(hillClimbing2, data = data, type = "bic")

hcScore2
```

```
## [1] -9361.871
```

```r
fuerzaHillClimbing2 <- arc.strength(hillClimbing2, data = data, criterion = "x2")
fuerzaHillClimbing2
```

```
##        from         to      strength
## 1     nieve nieveSuelo 2.515328e-101
## 2  tormenta    granizo  1.375034e-67
## 3    viento     lluvia  5.887107e-48
## 4    lluvia      rocio  1.588620e-78
## 5     rocio    neblina  9.575702e-44
## 6    lluvia   tormenta  9.350705e-27
## 7   neblina     niebla  8.140958e-19
## 8    viento    neblina  1.401866e-29
## 9    lluvia   escarcha  8.743909e-12
## 10   viento      rocio  1.009701e-08
## 11  granizo      nieve  1.243591e-28
## 12    rocio     niebla  4.868168e-09
## 13   viento    granizo  3.039235e-08
## 14   viento   tormenta  1.624833e-06
```

```r
#Tabu search
tabuSearch2 <- tabu(data, whitelist = whitelist, blacklist = blacklist)
tabuScore2 <- bnlearn::score(tabuSearch2, data = data, type = "bic")
tabuScore2
```

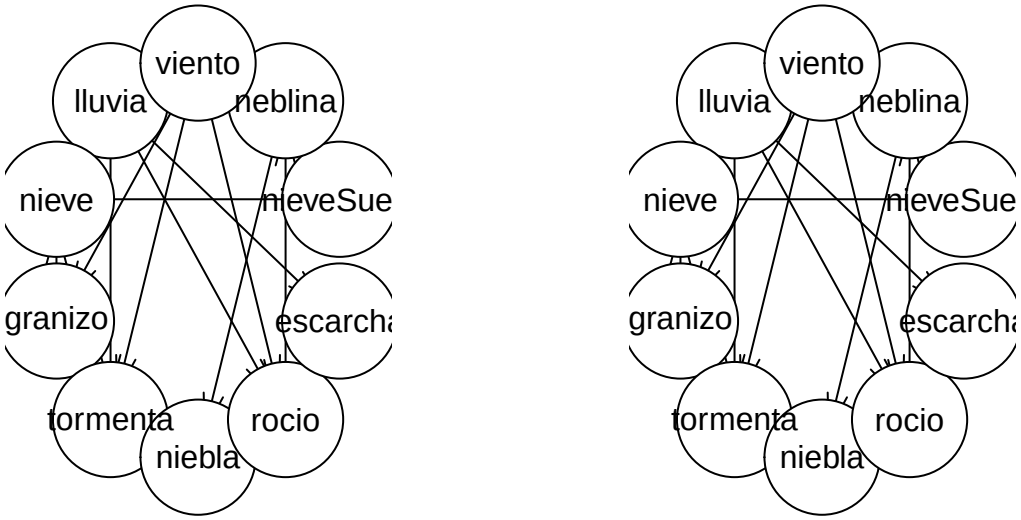
```
## [1] -9361.871
```

```r
fuerzaTabuSearch2 <- arc.strength(tabuSearch2, data = data, criterion = "x2")
fuerzaTabuSearch2
```

```
##        from         to      strength
## 1     nieve nieveSuelo 2.515328e-101
## 2  tormenta    granizo  1.375034e-67
## 3    viento     lluvia  5.887107e-48
## 4    lluvia      rocio  1.588620e-78
## 5     rocio    neblina  9.575702e-44
## 6    lluvia   tormenta  9.350705e-27
## 7   neblina     niebla  8.140958e-19
## 8    viento    neblina  1.401866e-29
## 9    lluvia   escarcha  8.743909e-12
## 10   viento      rocio  1.009701e-08
## 11  granizo      nieve  1.243591e-28
## 12    rocio     niebla  4.868168e-09
## 13   viento    granizo  3.039235e-08
## 14   viento   tormenta  1.624833e-06
```

Con estas restricciones, vemos que los dos métodos devuelven exactamente el mismo score. Además, podemos observar también que el score devuelto es más grande en este caso (lo que está bien, significa que las restricciones aplicadas ayudan en algo la construcción del DAG). Vemos también que los enlaces tienen la misma fuerza en los dos DAG (hemos creado la misma red neuronal, con dos métodos diferentes) y que el enlace que tiene la fuerza más pequeña es el que existe entre nieve y nieveSuelo. Podemos dibujar los DAG así creados y resumir los resultados obtenidos hasta ahora en una tabla. Por fin, podemos observar que los enlaces que hemos puesto en la *whitelist* no tienen porque ser muy fuertes : existen, pero no parecen muy verificados por los datos que tenemos.

```
par(mfrow=c(1,2))
plot(hillClimbing2)
plot(tabuSearch2)
```




```
par(mfrow=c(1,1))
resultadosDAG <- matrix(c(hcScore, hcScore2, tabuScore, tabuScore2), ncol=2)
colnames(resultadosDAG) <- c("Score del hill-climbing", "Score del tabu search")
rownames(resultadosDAG) <- c("Sin restricciones","Con restricciones")
rtab <- as.table(resultadosDAG)
head(rtab)
```

```
##                   Score del hill-climbing Score del tabu search
## Sin restricciones               -9366.750             -9365.175
## Con restricciones               -9361.871             -9361.871
```

Extra

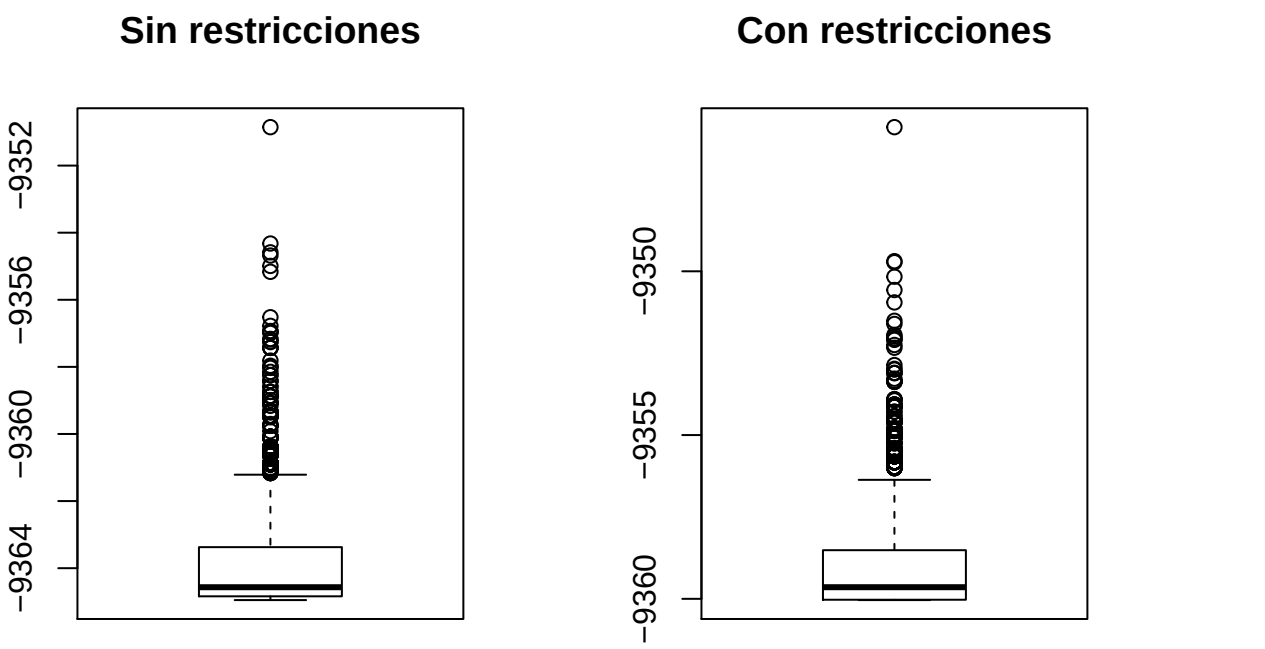
Ahora usamos el método leave-one-out que estudiamos sobre la tabla de datos para estudiar la dispersión del score obtenido con el método hill-climbing, con y sin las restricciones.

```
n <- nrow(data)
train <- 1:n
hcScore <- rep(NA, n)
hcScore2 <- rep(NA, n)
```

```
par(mfrow = c(1,2))
#Sin restricciones
for (i in train){

  #Ejecutamos hc para todos los datos excepto el elemento -i
  hillClimbing <- hc(data[train[-i],])
  hcScore[i] <- bnlearn::score(hillClimbing, data = data[train[-i],], type = "bic")

}
boxplot(hcScore, main="Sin restricciones")
summary(hcScore, digits=7)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -9364.951 -9364.839 -9364.568 -9363.924 -9363.373 -9350.855
```

```
#Con restricciones
for (i in train){

  #Ejecutamos hc para todos los datos excepto el elemento -i
  hillClimbing2 <- hc(data[train[-i],], whitelist = whitelist, blacklist = blacklist)
  hcScore2[i] <- bnlearn::score(hillClimbing2, data = data[train[-i],], type = "bic")

}
boxplot(hcScore2, main="Con restricciones")
```




```
summary(hcScore2, digits=7)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -9360.038 -9360.028 -9359.645 -9359.049 -9358.515 -9345.600
```

Se ve que globalmente, los dos boxplots parecen similares. El comando *summary* nos permite quantificar las informaciones que vemos en los boxplots. Se ve por ejemplo que el valor de la media en el caso con

restricciones (-9359.049) es un poco más alto que el valor de la media sin restricciones (-9363.924), lo que significa que con los datos que tenemos, aplicar las restricciones que estamos considerando es una buena idea, nos devuelve un score un poco más alto. Observamos que el valor minimo del score del caso con restricciones es incluso bastante mayor que el valor del tercer cuartil del caso sin restricciones.