TRABAJO FIN DE MÁSTER EN BIOESTADÍSTICA

Definición y estudio de redes bayesianas aplicadas a Ciencias de la Salud y de la Vida

Septiembre de 2021

Carlos Mateos Marcos

Tutoras: Julia Amador Pacheco y Rosario Susi García

ÍNDICE

GLOSARIO

AIC: Akaike Information Criterion

BDe: Bayesian Dirichlet equivalent

BGe: Bayesian Gaussian equivalent

BIC: Bayesian Information Criterion

CPDAG: Grafo Acíclico Completo Parcialmente Dirigido

DAG: Grafo Acíclico Dirigido

GS: Growth-Shrink

HC: Hill-Climbing

IAMB: Incremental-Association Markov Blanket

MMHC: Min-Max Hill Climbing

MMPC: Min-Max Parents and Children

SHD: Distancia Estructural de Hammond

RESUMEN

Las redes bayesianas son modelos gráficos probabilísticos que expresan las relaciones de dependencia condicional en un conjunto de variables. Desde su concepción, las redes bayesianas han estado profundamente ligadas a las Ciencias de la Salud y de la Vida, especialmente en el área clínica. Existe una bibliografía extensa sobre aplicaciones de las redes bayesianas a este ámbito. Sin embargo, el análisis de algoritmos de aprendizaje de redes y parámetros, y su aptitud en función de factores como la cantidad de variables, la naturaleza de los datos o la complejidad de la estructura de dependencia no es un tema común en la literatura. En este trabajo, analizamos la aplicación de estas técnicas a problemas descritos en la bibliografía, exploramos el software *bnlearn* disponible en el lenguaje de programación R documentando nuestro código y evaluamos las estrategias de aprendizaje que mejor se ajustan a cada tipo de datos. Esperamos con ello aportar conocimiento sobre las redes bayesianas y proporcionar un punto de partida para su estudio a profesionales sanitarios e investigadores.

Palabras clave: aprendizaje, dependencia, modelo gráfico probabilístico, red bayesiana, parámetros.

SUMMARY

Bayesian networks are graphical probabilistic models which represent conditional dependence relationships among a set of variables. Since their origin, Bayesian networks have been deeply linked to Health and Life Science, especially regarding the clinical field. There is an extensive bibliography about Bayesian networks applications in this field. However, the analysis of algorithms for network learning and their fitness according to factors like the amount of variables, the nature of the data or the complexity of the dependence structure is not a common subject in literature. In this project, we analyze the application of such techniques to problems characterized in the bibliography, we explore the software bnlearn available in the programming language R documenting our code and we evaluate the learning strategies which best fit each kind of data. With this project, we hope to contribute to the knowledge about Bayesian networks and provide a starting point for their study to health professionals and researchers.

Keywords: learning, dependence, graphical probabilistic model, Bayesian network, parameters.

1. INTRODUCCIÓN

Las Ciencias de la Salud y de la Vida constituyen un campo de estudio en el que determinar los fenómenos causales de un evento no suele ser un proceso sencillo. El profesional suele enfrentarse a numerosas fuentes de incertidumbre, datos ambiguos o incompletos y multitud de fenómenos multicausales en que resulta complicado establecer relaciones entre variables. En este sentido, los modelos gráficos probabilísticos son una herramienta útil para que el usuario pueda interpretar de forma intuitiva cómo cada uno de los factores afecta al resto del sistema.

Las redes bayesianas son un tipo de modelo gráfico probabilístico utilizado para representar relaciones de dependencia o de causalidad entre variables. Gracias a ello, obtenemos nuevo conocimiento bajo esquemas de incertidumbre, facilitando a su vez la toma de decisiones y el razonamiento por medio del uso de la teoría de probabilidad.

En este sentido, en la literatura sobre redes bayesianas se pueden encontrar trabajos como el de Scanagatta, Salmerón y Stella (2019), quienes realizan una revisión de los principales algoritmos implementados para el aprendizaje de redes bayesianas y, además, despliegan una exploración enfocada en listar las herramientas de software que se utilizan habitualmente para el modelado de redes bayesianas. Scutari y Denis (2014) presentan en su libro los paquetes más populares implementados para el modelado de redes bayesianas, haciendo especial énfasis en el paquete *bnlearn* desarrollado por Scutari (2010). De la misma manera, Nagarajan, Scutari y Lèbre (2013) muestran cómo se pueden ejecutar en R las diferentes tareas propias de la construcción y análisis de redes bayesianas a partir de código proveniente de diversos paquetes.

1.1. Motivación

Las redes bayesianas han estado íntimamente ligadas a la práctica clínica desde su concepción. La medicina suele enfocarse en los fenómenos causales de la enfermedad y, debido al gran número de variables con que se trabaja en biología, está sujeta a un alto grado de incertidumbre. Estas particularidades hicieron necesario un nuevo paradigma, que tuviera en cuenta para el diagnóstico tanto la evidencia disponible como el conocimiento previo en la materia.

Desde este punto, el desarrollo y expansión en el uso de redes bayesianas ha afectado a otras áreas de las Ciencias de la Salud y de la Vida, tales como la investigación bioquímica y genética o la conservación medioambiental. La naturaleza de las variables en cada una de estas ramas es notablemente distinta, y es por ello que la aplicación de redes bayesianas no puede ser puramente algorítmica. Se requiere una comprensión global de cómo funciona la estructura causal de cada problema, para así tratar de modelarla gráficamente.

1.2. Estructura

Estructuraremos el trabajo de la siguiente forma: en el apartado 2 se expondrá el Estado del Arte en cuanto a inferencia estadística y fundamentos de las redes bayesianas en Ciencias de la Salud y de la Vida. En el apartado 3, plantearemos el objetivo principal y objetivos secundarios de nuestro trabajo. En el apartado 4 se presentan los materiales, en forma de recursos de programación y bases de datos, y métodos, es decir, la algorítmica del trabajo. En el apartado 5 se describen los resultados obtenidos para cada una de las redes aprendidas. Finalmente, en el apartado 6 se presentan las conclusiones y algunas ideas para trabajo futuro. Incorporamos también el código utilizado para llevar a cabo el trabajo en el anexo.

2. ESTADO DEL ARTE

La inferencia estadística se centra en extraer conclusiones a partir de observaciones numéricas en muestras sobre magnitudes no observadas en poblaciones. En el contexto de las Ciencias de la Salud y de la Vida, la inferencia sobre las probabilidades en una población, así como sus diferencias, se basa en muestras de pacientes.

2.1. La inferencia frecuentista y sus limitaciones

La inferencia estadística es un procedimiento que permite extraer información de datos disponibles y generalizar los resultados más allá de nuestras observaciones (Lesaffre, 2012). Gracias a la inferencia, el investigador puede formular o verificar hipótesis, o tomar decisiones en fases experimentales. La inferencia depende tanto de los datos disponibles como del modelo probabilístico que se les imputa, es decir, de la distribución que asumimos que teóricamente siguen dichas observaciones.

Además, los resultados de la inferencia también pueden variar en función del enfoque que utilicemos para extrapolar los resultados a la población general. Podemos distinguir dos paradigmas mayoritarios para generar inferencia estadística: el enfoque frecuentista y el enfoque bayesiano.

Las técnicas de inferencia frecuentista, que se han utilizado en la mayor parte de trabajos científicos hasta la actualidad, han sido puestas en entredicho de forma creciente en las últimas décadas (Silva y Benavides, 2001). Se basan en magnitudes denominadas p-valores, que determinan la significación de la hipótesis nula (H_0); el p-valor se corresponde con la proporción de muestras aleatorias en las cuales encontraríamos resultados tan extremos como los que encontramos en nuestros datos, asumiendo que H_0 es cierta. P-valores bajos (comúnmente se estipulan valores de 0.05 o menores en trabajos de investigación) apuntan a que H_0 es inverosímil, por lo que rechazamos la hipótesis nula y aceptamos su contraparte, la hipótesis alternativa (H_1). El nivel de significación, denotado como α, expresa la probabilidad de rechazar la hipótesis nula cuando ésta es verdadera.

El intervalo de confianza (CI) de un parámetro de interés delimita el rango de posibles valores en que se encontraría dicho parámetro en el conjunto de la población. El CI se calcula para un nivel de confianza determinado (habitualmente 90%, 95% ó 99%). Este porcentaje se corresponde con la proporción de intervalos encontrados, cuando se toma un número elevado de muestras aleatorias, que contienen el verdadero valor del parámetro.

La perspectiva frecuentista a la hora de realizar el análisis estadístico de los resultados presenta varias limitaciones ineludibles (Lautner-Csorba, O., et al, 2012).

1. Los p-valores, de los cuales depende la decisión de rechazar o no rechazar la hipótesis nula, están fundamentalmente condicionados por el tamaño muestral. Es común que el investigador parta de evidencias muy reducidas de la realidad al construir sus hipótesis. En estos casos, será muy difícil obtener conclusiones. Por el contrario, es virtualmente seguro que, con muestras de tamaño masivo, se podrá argumentar el rechazo de la hipótesis nula, cualquiera que sea la verdadera naturaleza del problema. En última instancia, la validación de la hipótesis está en manos de un elemento ajeno a la realidad, y es completamente dependiente de los recursos de que dispone el investigador.
2. Los contrastes de hipótesis ignoran la evidencia externa al espacio experimental u observacional. Es imposible actualizar la inferencia a medida que aparece nueva información, ni corregirla con información preexistente. En las ciencias naturales, esto implica ignorar la plausibilidad de la teoría que se está analizando basándose en investigaciones previas. Si bien se alega que los contrastes de hipótesis permiten actuar con objetividad, tratar cada experimento u observación como un espacio estanco provoca que, cuando existen evidencias en conflicto, el investigador introduzca sesgos de interpretación para integrar sus resultados en el conocimiento general.
3. Los contrastes de hipótesis inducen a decisiones dicotómicas. Es imposible cuantificar la credibilidad de cada una de las hipótesis para decidir en consecuencia; se trata de un enfoque acientífico, porque las conclusiones han de ser representaciones provisionales de la realidad, y como tal han de contar con un grado de credibilidad que vaya variando con cada nueva evidencia. Es un error común en la investigación el tomar el p-valor como una medida cuantitativa de la fuerza de la hipótesis alternativa, y el intervalo de confianza como un intervalo de la probabilidad de que el valor real de un parámetro se encuentre delimitado dentro de un rango. Sin embargo, como hemos visto, el sentido de este enfoque es muy distinto.

2.2. La inferencia bayesiana y sus características

La estadística bayesiana supone un cambio de paradigma en el proceso de inferencia en auge en los últimos años. Los procedimientos bayesianos para el análisis de los datos constan de tres fases (Gelman et al, 2013):

1. Definir un modelo de probabilidad total, que recoja la distribución de probabilidad conjunta para todos los parámetros cuantificables en un problema. El modelo debe ser coherente con el conocimiento disponible sobre la materia y el proceso de recolección de datos.
2. Actualizar nuestro conocimiento con los datos observados; calcular e interpretar la distribución a posteriori, esto es, la distribución de probabilidad condicional de los parámetros no observados de interés, dados los datos observados. La distribución a posteriori de los parámetros es entonces proporcional al producto de la información que teníamos sobre su distribución a priori, que recoge

el conocimiento inicial acerca de los parámetros, por su función de verosimilitud dados los datos observados.

3. Evaluar la idoneidad del modelo y las implicaciones de la distribución a posteriori resultante. El investigador debe analizar si el modelo se ajusta adecuadamente a los datos, si se extraen conclusiones razonables, y cómo se integran los resultados en las premisas del paso 1.

En los últimos 40 años, se han producido grandes avances en la inferencia bayesiana. La motivación principal del pensamiento bayesiano es que facilita interpretaciones razonadas ante conclusiones estadísticas. De esta forma, un intervalo de probabilidad bayesiano para un valor desconocido de interés puede calificarse como un intervalo de alta probabilidad de contener dicho valor, en contraste con el intervalo de confianza, que sólo debe interpretarse de acuerdo a una proporción de resultados en una colección de inferencias con muestras equivalentes.

Recientemente, el interés de la estadística aplicada se ha ampliado desde el contraste de hipótesis a la estimación por intervalos de probabilidad, lo cual ha puesto de relieve nuevo modelo de inferencia estadística, puesto que, como hemos dicho, para calcular los intervalos de probabilidad se necesita implementar un razonamiento bayesiano. De este modo, la característica definitoria de la inferencia bayesiana es la cuantificación de la incertidumbre, que nos permite acotar parámetros a rangos de valores y, con base en dichos rangos, ajustar modelos con múltiples parámetros e hipótesis de distribución multinivel (con modelos de efectos fijos y aleatorios) de la probabilidad. En la realidad, las limitaciones llegan a la hora de calcular las distribuciones de probabilidad de los parámetros del modelo a posteriori.

Podemos distinguir dos tipos de estimandos. En primer lugar, están las cantidades no observables directamente; parámetros que gobiernan el proceso que hipotéticamente daría lugar a los datos observados (por ejemplo, coeficientes de regresión). En segundo lugar, tenemos las magnitudes potencialmente observables, como pueden ser observaciones a futuro de una medida determinada en una población, o un tratamiento no recibido en el ensayo clínico. La distinción entre ambos tipos no siempre es precisa, pero en general es útil para encajar modelos estadísticos en el mundo real.

En general, denotamos con $\boldsymbol{\theta}$ las magnitudes no observables o los parámetros de interés (por ejemplo, probabilidades de supervivencia bajo cada tratamiento para individuos aleatorios de una población en un ensayo clínico); *y* hace referencia a los datos observados (siguiendo con el ejemplo anterior, número de supervivientes y fallecidos en cada grupo), e $\tilde{\boldsymbol{y}}$ denota valores desconocidos, pero potencialmente observables (resultados en pacientes bajo el otro tratamiento, o resultado bajo cada tratamiento para un nuevo paciente de la misma población).

Las conclusiones que extraemos de un parámetro de interés θ, o dato no observado $\tilde{y}$, se expresan en términos de probabilidades condicionadas sobre los datos observados ***y***. Lo denotamos como $p(\theta|\boldsymbol{y})$ o $p(\tilde{y}|\boldsymbol{y})$, respectivamente. Al imponer este condicionamiento, la inferencia bayesiana se aleja del enfoque estadístico frecuentista, que se basa en una estimación de θ (o $\tilde{y}$) sobre la distribución de posibles valores de ***y***, condicionado sobre el verdadero valor desconocido del parámetro, $p(\boldsymbol{y}|\theta)$. Pese a esta diferencia en la filosofía de la estadística frecuentista y la estadística bayesiana, para muchos análisis simples las conclusiones de ambos métodos de inferencia son prácticamente intercambiables. Sin embargo, los resultados obtenidos por métodos bayesianos pueden extrapolarse a problemas más complejos.

Para realizar inferencias sobre θ dado ***y***, debemos empezar planteando un modelo que defina la distribución de probabilidad conjunta para θ e ***y***. La distribución de probabilidad conjunta puede escribirse como producto de dos distribuciones de probabilidad, comúnmente conocidas como distribución a priori $p(\theta)$ y distribución de las observaciones dado θ, $p(\boldsymbol{y}|\theta)$, respectivamente.

$$p(\theta, \boldsymbol{y}) = p(\theta)p(\boldsymbol{y}|\theta)$$

El teorema de Bayes permite obtener la distribución de probabilidad a posteriori de θ condicionada a ***y***:

$$p(\theta|\boldsymbol{y}) = \frac{p(\theta,\boldsymbol{y})}{p(\boldsymbol{y})} = \frac{p(\theta)p(\boldsymbol{y}|\theta)}{p(\boldsymbol{y})}; \text{con } p(\boldsymbol{y}) > 0$$

De este modo, la probabilidad de θ a posteriori es proporcional al producto de la probabilidad de θ a priori por la probabilidad de las observaciones ***y*** dado θ:

$$p(\theta|y) \propto p(\theta)p(\boldsymbol{y}|\theta)$$

El segundo término en la expresión, $p(\boldsymbol{y}|\theta)$, se toma como función de θ, no de ***y***. La probabilidad condicionada se denomina función de verosimilitud, y se denota como $L(\theta; \boldsymbol{y})$. Estas fórmulas encapsulan la idea de la inferencia bayesiana: la tarea principal de cualquier aplicación es desarrollar el modelo $p(\theta, y)$ y calcular $p(\theta|y)$.

2.3. Fundamentos de las redes bayesianas

Las redes bayesianas son fruto de la intersección de diversos campos de las ciencias formales, como la teoría de la probabilidad y la teoría de grafos, y el conocimiento y experiencia del investigador en la disciplina sobre la que se realiza la inferencia. Las redes bayesianas son modelos probabilísticos en forma de grafo, que plasman las relaciones de dependencia entre las variables y muestran la estructura causal de la realidad que se está estudiando.

2.3.1. Introducción a los grafos

Un grafo ***G** = (**V, A**)* es un modelo gráfico que consiste en un conjunto ***V*** de nodos o vértices, y un número determinado de uniones entre vértices llamadas arcos o aristas, que forman el conjunto ***A***. Cada enlace *A = (U, V)* se puede considerar una pareja ordenada o desordenada de nodos, los cuales están conectados por un arco, en caso de existir dirección en la relación, o una arista, en caso de que la relación no sea direccional. A los vértices conectados por un arco o arista se les llama nodos vecinos. Si el conjunto de nodos vecinos *(U, V)* es un par ordenado, se habla de *U* como la cola del arco y de *V* como la cabeza; el arco está dirigido de *U* a *V*, y normalmente se representa su direccionalidad con una flecha *(U→V)*. Si *(U, V)* es un par desordenado de nodos, *U* y *V* simplemente están conectados por una arista. Cuando nos encontramos con un arco no dirigido o arista, ésta suele representarse con una línea simple *(U–V)* (Nagarajan et al, 2013).

La clasificación de los arcos como dirigidos o no dirigidos puede generalizarse a los propios grafos, que se consideran dirigidos y se denotan como ***G** = (**V, A**)* cuando todos sus arcos son dirigidos, no dirigidos y denotados como ***G** = (**V, E**)* si todos los arcos son no dirigidos, y parcialmente dirigidos o mixtos y denotados como ***G** = (**V, A, E**)* si contienen arcos tanto dirigidos como no dirigidos.

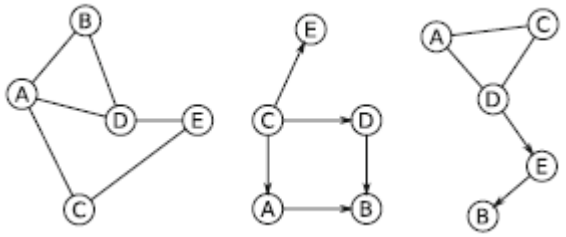


Fig. 1. Ejemplos de grafo no dirigido (izquierda), dirigido (centro) y parcialmente dirigido (derecha) (Nagarajan et al, 2013).

El esqueleto o grafo no dirigido subyacente se define como aquel que se construye a partir de un grafo dirigido o parcialmente dirigido al sustituir todos sus arcos por aristas. La configuración de los arcos es lo que define la estructura de un grafo. Se asume que los vértices *U* y *V* conectados por cada arco son distintos y sólo están unidos por un arco, de modo que *(U, V)* denota unívocamente un arco. Con esta nomenclatura excluimos la posibilidad de que *U = V* y, por tanto, desde un nodo salga un arco que se dirige hacia el mismo nodo formando un bucle.

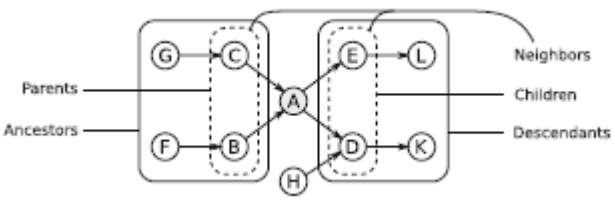


Fig. 2. Padres, hijos, ancestros, descendientes y vecinos del nodo *A* en un grafo dirigido (Nagarajan et al, 2013).

Al expresar la estructura de un grafo acíclico dirigido (DAG) estamos definiendo subconjuntos ordenados de nodos, como hicimos en el grafo dirigido de la figura 1. La dirección de los arcos es la que rige esta ordenación. Si un nodo V_i se encuentra antes que V_j, no puede haber ningún arco desde V_j a V_i, pero sí puede existir una ruta o conjunto de arcos que conduce desde V_i a V_j. En este caso, V_i sería un ancestro de V_j, y V_j es un descendiente de V_i. Si la ruta está compuesta por un solo arco, V_i es un nodo padre de V_j, y V_j es un nodo hijo de V_i.

2.3.2. Redes bayesianas: definición y propiedades

Las redes bayesianas estáticas, que a menudo se conocen simplemente como redes bayesianas, son modelos gráficos probabilísticos que toman la forma de DAGs. En las redes bayesianas se representan las relaciones de dependencia entre un conjunto de variables aleatorias. Cada arco está asociado a un parámetro en la red bayesiana, el cual recoge la distribución de probabilidad del nodo hijo en función del nodo padre. De este modo, las redes bayesianas se expresan como un par (***G***, ***P***), donde ***G*** es un DAG en que los nodos son las variables aleatorias ordenadas $\{X_1, \dots, X_n\}$ y los arcos recogen la estructura de dependencia entre ellas, y ***P*** recoge las distribuciones condicionales de probabilidad de cada variable X_i,es decir, $P_{X_i}(X_i|\Pi_{X_i})$ donde Π_{X_i} es el conjunto de padres de X_i.

Para que una red bayesiana sea apta a la hora de modelar un problema real, debe existir un correspondencia entre la separación gráfica ($\perp\!\!\!\perp_G$) de dos nodos y la independencia probabilística ($\perp\!\!\!\perp_P$) de las dos variables que se ven representadas por esos nodos, de modo que el estado de una variable no se ve afectado por el estado de la otra. De esta forma, la red bayesiana contiene todas las relaciones de independencia condicional que se dan en la realidad, y sólo las que se dan en la realidad. En una red bayesiana, la desconexión de dos nodos debido a un tercero sigue un criterio denominado d-separación (Pearl, 2014). Si *A*, *B*, y *C* son 3 nodos, o subconjuntos disjuntos de nodos, en un DAG, *C* provoca la d-separación de *A* respecto a *B* ($A \perp\!\!\!\perp_G B|C$), si existe un nodo *V* conectando *A* y *B* que satisfaga una de estas dos condiciones: existen al menos 2 arcos apuntando a *V* (conexión convergente) desde sus nodos vecinos y ni *V* ni sus descendientes están en el subconjunto *C*, o bien *V* está en el subconjunto *C* y no es el nodo central de una conexión convergente (Nagarajan, 2013).

Las redes bayesianas cumplen la propiedad de Markov; los estados de los nodos en la red en el futuro inmediato dependen exclusivamente de su estado actual, con independencia de su trayectoria en el pasado. Gracias a esta propiedad, podemos representar la distribución de la probabilidad conjunta de las variables aleatorias $\{X_1, \dots, X_n\}$ como producto de las distribuciones de probabilidad condicional de cada variable dados sus padres en el DAG, Π_{X_i}. Esta es una aplicación directa de la regla de la cadena (Korb y Nicholson, 2010). Cuando se trabaja con un número *n* de variables aleatorias discretas, la distribución de probabilidad conjunta P_X viene dada por:

$$P_X(X) = \prod_{i=1}^{n} P_{X_i}(X_i|\Pi_{X_i})$$

En el caso de variables aleatorias continuas, la función de densidad conjunta f_X para un conjunto de *n* variables viene dada por:

$$f_X(X) = \prod_{i=1}^{n} f_{X_i}(X_i|\Pi_{X_i})$$

Relacionado con lo anterior, la *manta de Markov* (Pearl, 2014) de un nodo es el conjunto de nodos que provocan su d-separación respecto al resto del grafo. La manta de Markov para un nodo $A \in \boldsymbol{V}$ es el mínimo subconjunto *S* de $\boldsymbol{V}$ tal que $A \perp\!\!\!\perp_P \boldsymbol{V} - S - A|S$ (Whittaker, 1990; Edwards, 2000). En cualquier red bayesiana, la manta de Markov de un nodo *A* es el conjunto de los padres de *A*, los hijos de *A*, y todos los nodos que compartan un hijo con *A*.

En la figura 3 se muestra lo que en DAGs se denomina conexiones fundamentales; son las tres posibles configuraciones dados tres nodos y dos arcos. En la conexión convergente, al nodo *C* llegan arcos que parten desde *A* y *B* respectivamente. El nodo *C* depende de las distribuciones conjuntas de *A* y *B*, por lo que al observar *C* los estados de *A* y B están relacionados. Es por ello que no se cumplen las condiciones para la d-separación entre *A* y *B* dado *C* y no se da independencia condicional.

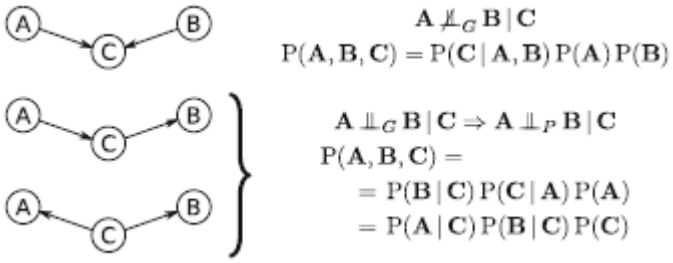


Fig. 3. Separación gráfica, independencia condicional y expresión probabilística para tres conexiones fundamentales (de arriba hacia abajo): conexión convergente, conexión en serie, conexión divergente (*Nagarajan et al, 2016)*.

Por otra parte, *A* y *B* son independientes dado *C* en las conexiones en serie o divergente. Cuando observamos *C* en la conexión en serie, el estado de *B* depende exclusivamente de dicha observación, y en el caso de la conexión divergente, el estado de A y el estado de *B* dependen exclusivamente de *C*. En ambos tipos de conexión, las distribuciones de probabilidad de *A* y *B* son condicionalmente independientes dado *C*.

Al comparar las distribuciones de probabilidad conjunta en la conexión en serie y la conexión divergente de la figura 3, se llega a expresiones equivalentes. Si se aplica el teorema de Bayes para definir las probabilidades de un nodo padre conociendo las de su nodo hijo, podría alternarse una expresión por la otra. Este conjunto de estructuras intercambiables (*estructuras equivalentes de Markov*) se denomina clase de equivalencia. Si extrapolamos este razonamiento al conjunto de una red, lo único que se necesita para diferenciar una clase de equivalencia de otra son sus conexiones convergentes (Chickernig, 1995), de modo

que las clases de equivalencia se pueden representar como CPDAGs, donde sólo se encuentran dirigidos los arcos que pertenecen a conexiones convergentes y aquellos que, en caso de dirigirse a la inversa, formarían otras conexiones convergentes o ciclos.

De manera relacionada, un DAG se puede transformar en el grafo no dirigido de las redes correspondientes siguiendo dos pasos: conectamos los nodos no adyacentes en cada conexión convergente con un arco no dirigido, e ignoramos la dirección del resto de arcos reemplazándolos por aristas. Esta transformación recibe el nombre de moralización, ya que une padres no adyacentes que tengan un hijo en común. El grafo resultante es el llamado *grafo moral* (Castillo et al., 1997).

Con lo que se ha descrito de las redes bayesianas se puede observar , al plantear un modelo de dependencia de nodos en un grafo, en realidad se analiza la distribución de probabilidad de un conjunto de variables en un entorno real.

2.3.3. Construcción de la red bayesiana

La construcción de las redes bayesianas es a menudo una tarea compleja, ya que la actualización de nuestras creencias sobre la distribución de probabilidad de un conjunto de variables no depende sólo de las observaciones de que se dispone, sino también de todo el conocimiento previo que exista sobre la materia. Toda esta información no siempre se puede integrar directamente en forma de modelo de probabilidad condicionada, y es por ello que se suele requerir la colaboración de expertos en el campo de estudio para construir redes bayesianas fiables.

El tipo de variables que componen la distribución de probabilidad también va a condicionar la forma en que se va a construir la red y cómo se van a estimar e interpretar sus parámetros. La literatura se ha enfocado mayormente en dos tipos de variables.

- Variables multinomiales: en este caso se trabaja con conjuntos de datos discretos o categóricos; se conoce como caso discreto. Este supuesto es el más común en la literatura, y las redes bayesianas correspondientes se conocen como *redes bayesianas discretas*.
- Variables normales multivariantes: en este supuesto se trabaja con conjuntos de datos continuos que siguen una distribución normal. Estas redes bayesianas se conocen como *redes bayesianas gaussianas* (Geiger and Heckerman, 1994; Neapolitan, 2003).

Aprendizaje de la red

La construcción de una red bayesiana mediante herramientas probabilísticas se conoce como aprendizaje, un término tomado de la teoría de sistemas e inteligencia artificial (Koller y Friedman, 2009). Consta de dos fases: selección del modelo gráfico y estimación de los parámetros.

La primera fase es el aprendizaje de la estructura, y consiste en identificar las relaciones de dependencia del grafo en la red bayesiana. Se han propuesto muchos algoritmos para el aprendizaje de la estructura, y pese a la variedad de terminología y marcos teóricos, suelen agruparse en tres categorías: algoritmos basados en restricciones, algoritmos basados en puntuaciones y algoritmos híbridos. El segundo paso es el aprendizaje de los parámetros. En esta fase, se estiman los parámetros del modelo.

Algoritmos basados en restricciones

Los algoritmos basados en restricciones se fundamentan en el algoritmo de causalidad inductiva (Verma y Pearl, 1991).

1. Para cada pareja de variables *A* y *B* en $\boldsymbol{V}$, se busca el conjunto $\boldsymbol{S}_{AB} \subset \boldsymbol{V}$ (incluyendo $S = \emptyset$, un conjunto vacío) tal que *A* y *B* son independientes dado $\boldsymbol{S}_{AB}$ y las variables *A* y *B* no pertenecen a S_{AB}. Si no existe el conjunto $\boldsymbol{S}_{AB}$, se coloca un arco no dirigido o arista entre los nodos correspondientes *A* y *B*. En este paso, se identifican parejas de variables en que se da una relación de dependencia, sin tener en cuenta la dirección. No existe entre ambas variables una independencia condicional respecto a una tercera variable o conjunto de variables, porque en la conexión entre *A* y *B* no hay intermediarios. En ocasiones este paso se interpreta como un proceso de podado o selección *backwards*: se comienza desde una red *saturada* (que incluya todos los arcos posibles entre todos los pares de nodos) y se descartan arcos espurios empleando tests de independencia condicional.
2. Para cada pareja de variables *A* y *B* que no sean vecinas pero tengan un vecino común *C*, se comprueba si se cumple que $C \in \boldsymbol{S}_{AB}$. Si esto no es cierto, se impone la dirección de los arcos *A*-*C* y *C*-*B* a $A \rightarrow C$ y $C \leftarrow B$, formando una estructura convergente. Las conexiones convergentes son el único tipo de conexión fundamental en que dos nodos no adyacentes no son condicionalmente independientes dado un tercero. Por tanto, mediante tests de independencia condicional, se puede determinar si los nodos A y B conforman una estructura convergente con cualquiera de sus vecinos, incluido *C*. Al finalizar este paso, conocemos la clase de equivalencia de la red, puesto que sólo es necesario conocer la dirección de los arcos en las estructuras convergentes.
3. Se impone la dirección de los arcos que aún no están dirigidos aplicando exhaustivamente las siguientes reglas:
    a. Si *A* es vecino de *B* y existe una ruta dirigida desde *A* hacia *B* (un conjunto de arcos que dirige de *A* a *B*), se impone la dirección de *A*-*B* como $A \rightarrow B$.
    b. Si *A* y *B* no son vecinos pero $A \rightarrow C$ y *C*-*B*, entonces se cambia esta última relación por $C \rightarrow B$.

    En esta fase, se define la orientación de los arcos no dirigidos de la clase de equivalencia construida recursivamente para obtener el CPDAG.
4. Se devuelve el DAG resultante, parcialmente completado.

Una limitación del algoritmo de causalidad inductiva es que las dos primeras fases del algoritmo no siempre se pueden aplicar a problemas reales, debido al número exponencial de potenciales relaciones de independencia condicional. Esto ha derivado en el desarrollo de algoritmos depurados como *PC* (Sprites et al., 2001), *Growth-Shrink* (Margaritis, 2003), *Incremental Association Markov Blanket* (IAMB, Tsamardins et al., 2003), y variaciones de este último (*Fast-IAMB*, Yaramakala and Margaritis, 2005, *Inter-IAMB*, Tsamardinos et al., 2003).

Estos algoritmos simplifican la identificación de los vecinos de cada nodo limitando la búsqueda a su manta de Markov. Como resultado, se reduce significativamente el número de tests de independencia condicional ejecutados por el algoritmo.

Los tests de independencia condicional para datos discretos son funciones de las *tablas de probabilidad condicional* (CPTs) de cada variable respecto a sus nodos padre. Las probabilidades condicionales de cada variable se estiman a partir de las frecuencias observadas en la muestra de cada nivel de la variable dado el nivel de sus padres.

En el caso gaussiano, los tests de independencia condicional y las puntuaciones de red son funciones de los coeficientes de correlación parcial $\rho_{XY|Z}$ de X e Y dado Z.

Algoritmos basados en puntuaciones

Los algoritmos basados en puntuaciones (también llamados *search-and-score*) están basados en técnicas exhaustivas de optimización para ajustar la estructura de una red bayesiana. Se genera un conjunto de redes candidatas, a cada una de las cuales se le asigna a una puntuación de red que refleja su verosimilitud. El algoritmo trata de maximizar esta puntuación. Algunos ejemplos de esta clase de algoritmos incluyen *Greedy Search* (Bouckaert, 1995), familias de *algoritmos genéticos* (Larrañaga et al, 1997) o *Simulated Annealing* (Bouckaert, 1995).

Una diferencia fundamental entre estos algoritmos es la forma en que se construye cada nueva red. Greedy Search comienza por una estructura de red (normalmente, el grafo vacío) y añade, elimina o invierte un arco en cada paso hasta que la puntuación no puede ser aumentada. Los algoritmos genéticos imitan la evolución natural seleccionando los modelos más aptos e integrando sus características mediante cruces (que combinan la estructura de dos redes) y mutaciones (que introducen alteraciones aleatorias en la estructura). Simulated Annealing realiza alteraciones locales (entre los vecinos de un único nodo), admitiendo tanto cambios que incrementen la puntuación de la red como cambios que la disminuyan, con una probabilidad de cambio inversamente proporcional al decrecimiento en la puntuación.

En todos los casos, los algoritmos basados en puntuaciones siguen un esquema común:

1. Se escoge una estructura de red ***G*** sobre el conjunto de nodos ***V***, normalmente (aunque no necesariamente) vacía.
2. Se calcula la puntuación de ***G***, denotada como $Score_G$.
3. Se declara *maxscore* = $Score_G$, de modo que la máxima puntuación conocida es la que se ha obtenido para la estructura ***G***.
4. Se repiten los siguientes pasos mientras *maxscore* aumente:
    a. Para cada posible adición, eliminación o inversión de un arco que no dé lugar a ciclos:
        i. Se calcula la puntuación de la red modificada *G**, $Score_{G^*} = Score(G^*)$
        ii. Si *ScoreG* > ScoreG*, se define *G = G** y *ScoreG = ScoreG**.
    b. Se actualiza *maxscore* con el nuevo valor de *ScoreG*.
5. Se devuelve el DAG ***G***.

Las diferencias fundamentales entre algoritmos se dan a la hora de calcular la puntuación de la red. Las puntuaciones de red más comunes en la literatura para redes bayesianas discretas son:

- Bayesian Dirichlet equivalent (BDe), la distribución de probabilidad a posteriori asociada a la estructura de red y los parámetros de cada distribución de probabilidad local, esto es, asociada a cada nodo (Heckerman et al, 1995). Se fundamenta en la distribución de Dirichlet, una generalización de la distribución Beta para el caso multivariante:

$$Dir(\alpha) = \frac{1}{B(\alpha)} \prod_{i=1}^{n} x_i^{\alpha_i - 1}$$

  Donde α es un vector de números reales denominados parámetros de concentración. La distribución de Dirichlet puede emplearse como conjugada a priori de la distribución multinomial, razón por la cual se utiliza en la puntuación de redes bayesianas discretas.

- Bayesian information criterion (BIC), una puntuación de verosimilitud penalizada definida como:

$$BIC = \sum_{i=1}^{n} \log P_{X_i}\left(X_i \middle| \Pi_{X_i}\right) - \frac{d}{2} \log n$$

  Donde *d* es el número de parámetros de la distribución global. BIC converge asintóticamente con la distribución de probabilidad a posteriori BDe.

Estas funciones de puntuación asignan la misma puntuación a redes que pertenezcan a la misma clase de equivalencia. Además, pueden descomponerse en las puntuaciones asociadas a cada nodo, lo cual es una ventaja significativa cuando se trata de aprender la estructura de la red (las únicas partes de la puntuación que deben estimarse son aquellas que difieren entre las redes comparadas).

En las redes bayesianas gaussianas, las puntuaciones de red más comunes son:

- Bayesian Gaussian equivalent (BGe): la densidad a posteriori asociada a la estructura de red y los parámetros de cada distribución de probabilidad local, esto es, asociada a cada nodo (Heckerman et al, 1995). Se fundamenta en la distribución de Wishart, una generalización de la distribución Gamma para datos multivariante. La distribución de Wishart puede emplearse como conjugada a priori de la inversa de la matriz de covarianzas para una distribución normal multivariante, razón por la cual se utiliza en la puntuación de redes bayesianas gaussianas.
- Bayesian information criterion (BIC), en este caso definida como:

$$BIC = \sum_{i=1}^{n} \log f_{X_i}\left(X_i \middle| \Pi_{X_i}\right) - \frac{d}{2}\log n$$

Algoritmos híbridos

Los algoritmos híbridos combinan características de algoritmos basados en restricciones y algoritmos basados en puntuaciones. Los dos algoritmos más conocidos de esta familia son *Sparse Candidate* (SC) de Friedman et al. (1999) y *Max-Min Hill-Climbing* (MMHC) de Tsamardinos et al. (2006). El algoritmo SC sigue esta serie de pasos:

1. Se elige una estructura de red ***G*** sobre el conjunto de nodos ***V***, normalmente (aunque no necesariamente) vacía.
2. Se repiten los siguientes pasos hasta alcanzar la convergencia:
    a. Restricción: se selecciona un subconjunto $\boldsymbol{C}_i$ de nodos que potencialmente serían padres para cada nodo X_i en el conjunto ***V***. De esta forma, se minimiza el espacio de búsqueda.
    b. Maximización: se identifica la estructura de red ***G**** que maximiza *Score(**G***)* entre las redes en que los padres de cada nodo X_i están incluidos en el correspondiente subconjunto $\boldsymbol{C}_i$.
    c. Se define ***G*** = ***G****.
3. Se devuelve el DAG ***G***.

En el algoritmo SC estos pasos se aplican iterativamente hasta que no se da un cambio en la red o una mejora en la puntuación; la forma en que se produce cada iteración depende del método implementado. Por el otro lado, en el algoritmo Min-Max Hill Climbing la restricción y maximización se aplican una única vez; los métodos de restricción de Max-Min Parents and Children (MMPC) permite ajustar los conjuntos candidatos C_i y el método Hill-Climbing Greedy Search permite hallar la red óptima en un solo paso.

Aprendizaje de los parámetros

Las distribuciones de probabilidad globales (de cada nodo condicionadas al resto de la red) y locales (de cada nodo condicionadas a sus vecinos) dependerán nuevamente de la naturaleza de los datos y los objetivos del análisis. En el caso de trabajar con variables multinomiales, tanto las distribuciones globales como locales son multinomiales, y estas últimas son representadas como CPTs. La distribución multinomial se expresa como un vector ***Y*** compuesto por *K* variables aleatorias, condicionada a los valores de un conjunto de observaciones **X** de las cuales son dependientes:

$$P(\boldsymbol{Y}|\boldsymbol{X}) = \frac{n!}{Y_1!\,Y_2!\,\dots\,Y_K!}\prod_{i=1}^{K} p(\boldsymbol{Y} = Y_i|\boldsymbol{X})^{Y_i}$$

Cuando se trabaja con variables normales multivariantes, la distribución global es una normal multivariante, mientras que las distribuciones locales son variables aleatorias univariante normales, representadas como modelos lineales donde los padres actúan como covariables a las que se asignan coeficientes de regresión β. Si suponemos una variable *Y* dependiente de un conjunto de variables ***X*** en una red bayesiana gaussiana, la imagen de *Y* se puede modelar como:

$$f(Y|\boldsymbol{X}) \sim N(X\beta, \sigma) = \sum_{j=1}^{L}(\beta_j * X_j) + \varepsilon$$

Donde el error estándar ε sigue una distribución normal centrada en 0 y desviación típica σ.

Después de aprender la estructura de la red, podemos estimar y actualizar los parámetros de la distribución global aplicando la propiedad de Markov sobre los datos muestrales.

Las distribuciones locales suelen involucrar un pequeño número de variables. Además, su dimensión normalmente no escala con el tamaño del conjunto ***V***, ya que un mayor número de variables no implica que la red sea más densa. Cuando se calcula la complejidad de los algoritmos para construir una red, se asume que el tamaño de una distribución local está acotado por un número máximo de variables. Esto contradice el aumento exponencial de la complejidad de la red en función del número de variables (la llamada maldición de la dimensión). Existen dos enfoques mayoritarios en la literatura para la estimación de estos parámetros: uno basado en la estimación de la máxima verosimilitud y otro basado en la estimación bayesiana (Nagarajan *et al*, 2013).

El número de parámetros necesarios para identificar la distribución global, que es la suma del número de parámetros de las distribuciones locales, también se ve reducido, debido a las relaciones de independencia condicional dentro de la red. Por ejemplo, en redes bayesianas gaussianas, los coeficientes de regresión β

que involucran variables condicionalmente independientes son iguales a 0 y, en redes bayesianas discretas, las probabilidades conjuntas se factorizan en probabilidades marginales cuando tenemos distribuciones condicionales.

Sin embargo, la estimación de parámetros sigue siendo problemática en muchas situaciones. Por ejemplo, es cada vez más común contar con tamaños muestrales mucho menores que el número de variables incluidas en el modelo. Esto es típico de equipos de análisis biológico de alto rendimiento, como los *microarrays*, que cuentan con unas pocas decenas o centenas de observaciones y miles de variables en forma de genes. Esta configuración de experimentos implica una gran variabilidad muestral.

Discretización

Un modo directo de aprender redes bayesianas desde datos mixtos (o datos puramente gaussianos cuyo ajuste resulta complicado) es convertir todas las variables continuas en variables discretas y, después, aplicar las técnicas descritas anteriormente sobre redes bayesianas discretas. Esta estrategia se denomina discretización o *binning*, y permite evitar la complejidad de formular una función de distribución continua que se adapte a los datos. La discretización también se aplica para tratar con datos continuos cuando una o más variables presentan desviaciones severas de la normalidad (asimetría, colas pesadas).

Los intervalos en que las variables serán discretizadas pueden elegirse de una de las siguientes formas:

- Utilizar conocimiento previo de los datos. Los límites del intervalo están definidos, para cada variable, como correspondientes a eventos reales significativamente distintos, como la concentración de un contaminante (ausente, peligroso, letal) o clasificaciones de edad (niño, adulto, anciano).
- Escoger el número de intervalos y sus límites para equilibrar precisión y pérdida de información (Kohavi y Sahami, 1996), de nuevo en una variable cada vez y antes de aprender la estructura de la red. Un enfoque similar, considerando pares de variables, se presenta en Hartemink (2001).
- Ejecutar discretizaciones y aprendizajes iterativamente hasta que no se obtienen mejoras (Friedman y Glodszmidt, 1996).

Estas estrategias implican distintas compensaciones entre la precisión de la representación discreta de los datos originales y la eficiencia computacional de la transformación.

2.3.4. Inferencia en redes bayesianas

Uno de los principales propósitos de las redes bayesianas es realizar inferencias sobre las variables no observadas de un nuevo sujeto a partir de los valores observados para otras variables dentro del mismo

sujeto. En el contexto del modelo probabilístico, a este proceso se le llama propagación de la evidencia, ya que estimamos las variables desconocidas a posteriori con base en los valores conocidos.

Existen multitud de algoritmos para llevar a cabo dicha propagación. Entre los más simples encontramos los clasificadores bayesianos ingenuos o *naive Bayes*. Pese a que estos algoritmos resultan razonablemente eficientes y fáciles de implementar, en ocasiones es necesario formular clasificadores más sofisticados. En estas circunstancias, podemos mejorar la precisión de las estimaciones al considerar la red bayesiana como un árbol de decisiones en que la estimación de una variable se ve afectada por la evidencia tanto de los nodos padre como de los nodos hijo (Sucar y Tonantzintla, 2006).

Existen multitud de algoritmos clasificadores que implementan la propagación de la evidencia en árbol para realizar inferencias. De entre ellos, destaca por su simplicidad la familia de algoritmos *Tree-Augmented Naive Bayes* (TAN), que aplica la asignación de los niveles máximos a posteriori de cada variable utilizando la propagación en árbol.

2.3.5. Fusión de redes bayesianas

La agregación de redes bayesianas no es un proceso trivial, y hasta la fecha existen pocas propuestas en la literatura. Además, la mayoría de métodos que se han formulado exigen, hasta cierto grado, la participación de expertos en la materia para construir la estructura consenso. Pese a ello, el trabajo de Córdoba, Bielza y Larrañaga (2018) propone algunos algoritmos prometedores para esta tarea.

GBNFuseSVote

El método *GBNFuseSVote* está basado en votaciones. Cada modelo individual aporta su voto, y el voto mayoritario es elegido. En este caso, la votación se lleva a cabo sobre cada uno de los posibles arcos. El conjunto de arcos que articule la red fusionada estará determinado por el número de votos sobre cada arco.

Representamos las votaciones en una matriz que contiene la suma de matrices de adyacencia de cada red generada. Los elementos de una matriz de adyacencia M^G de dimensión $n \times n$ asociada a un grafo $\boldsymbol{G} = (\boldsymbol{V} = \{1,\dots,n\}, \boldsymbol{E})$ pueden definirse, para cada pareja de nodos $i,j \in \{1,\dots,n\}$, como:

$$M_{ij}^{G} = \begin{cases} 1 \text{ si } (i,j) \in \boldsymbol{E} \\ 0 \text{ en otro caso} \end{cases}$$

De esta manera, si denotamos como $\{\boldsymbol{G}_1,\dots,\boldsymbol{G}_k\}$ el conjunto de grafos que obtenemos de cada tabla de datos separadamente, la matriz de votaciones queda definida como:

$$M^{V} = \sum_{j=1}^{k} M^{\boldsymbol{G}_j}$$

Obtenida la matriz de votaciones M^V, cada arco es comparado con un parámetro umbral *t*. Aquellos arcos que lo superen o igualen, esto es, que aparezcan en al menos *t* grafos son incluidos. De este modo, la matriz de adyacencia para la red fusionada, M^{GF}, queda definida como:

$$M_{ij}^{G_F} = \begin{cases} 1 \text{ si } v_{ij} \geq t \\ 0 \text{ en otro caso} \end{cases}$$

Donde v_{ij} es el arco *(i,j)* de la matriz de votos M^V y *t* es el umbral especificado. En caso de que la adición de un arco cree un ciclo, este queda descartado. La dirección del arco mayoritaria en el conjunto de grafos es la que se impone en la red fusionada.

GBNFuseSInter

Este método se basa en la representación de la distribución normal multivariante en la forma factorizada, y en un teorema planteado por Sagrado y Moral (2003). Según este teorema, si $\boldsymbol{G}=(\boldsymbol{V},\boldsymbol{E}_G)$ y $\boldsymbol{H}=(\boldsymbol{V},\boldsymbol{E}_H)$ son dos DAGs con estructuras de independencia I_G e I_H, y se puede encontrar un orden ancestral compatible con $\boldsymbol{G}$ y $\boldsymbol{H}$, entonces $\boldsymbol{G}\cap\boldsymbol{H}=(\boldsymbol{V},\boldsymbol{E}_G\cap\boldsymbol{E}_H)$ es la red bayesiana correspondiente al modelo $I_G \cup I_H$, de acuerdo a los principios de independencia condicional.

Este teorema implica que, bajo ciertas condiciones, la intersección del conjunto de arcos de las redes individuales permite representar la unión de sus estructuras de independencia condicional. Por tanto, considerando el conjunto $\{\boldsymbol{G}_1,\ldots,\boldsymbol{G}_k\}$ de grafos iniciales, con $\boldsymbol{G}_j=(\boldsymbol{V},\boldsymbol{E}_j)$ para cada $j \in \{1,\ldots,k\}$, se define de forma inicial el conjunto de arcos para la red agregada como:

$$\widetilde{\boldsymbol{E}}_{G_F} = \cap_{j=1}^{k} \boldsymbol{E}_j$$

Sin embargo, esta expresión rara vez es aplicable en la realidad, puesto que es habitual que existan arcos presentes en la mayoría de las redes bayesianas, pero no en todas ellas, y que pertenecen al modelo teórico. Para mejorar el criterio de intersección, acudimos a la forma factorizada de la distribución gaussiana multivariante que vimos anteriormente:

$$f_X(X) = \prod_{i=1}^{n} f_{X_i}(X_i|\Pi_{X_i})$$

Definimos la imagen del nodo hijo de acuerdo a los padres como:

$$X_i|\Pi_{X_i} \sim N(\mu_i + \sum_{j\in\Pi_{X_i}} \beta_{ij}(x_j - \mu_j), V(X_i|\Pi_{X_i}))$$

Un coeficiente β_{ij} de 0 implica que el nodo *j* no es padre del nodo *i* en el DAG de la red gaussiana. Por tanto, la medida de la verosimilitud de los arcos se obtiene a partir de los coeficientes de regresión. Si el enlace $(i,j) \in \boldsymbol{E}_k$ para algún $k \in \{1,\ldots,K\}$ y *(i,j)* no se halla en el grafo condensado, el coeficiente de regresión β_{ij} es un

factor relevante en la inclusión de *(i,j)* en el modelo final. De esta forma, el conjunto de arcos de la red fusionada queda definido como:

$$\boldsymbol{E}_{G_F} = \widetilde{\boldsymbol{E}}_{G_F} \cup \left\{(i,j): (i,j) \in \cup_{k=1}^{K} \boldsymbol{E}_k \setminus \widetilde{\boldsymbol{E}}_{G_F}, \hat{\beta}_{ij} \geq t\right\}$$

Donde $t \in [0,1]$ es un parámetro de umbral, y el coeficiente de regresión se halla estanzarizado:

$$\hat{\beta}_{ij} = \beta_{ij} \frac{s_i}{s_j}$$

Siendo s_i y s_j las cuasidesviaciones típicas de las observaciones de los nodos *i* y *j*. La intersección de los grafos se efectúa de nuevo con base en la matriz de votos M^V; sin embargo, en este caso, los elementos de la matriz del grafo de intersección se definen como:

$$\widetilde{m}_{ij} = \begin{cases} 1 \text{ si } v_{ij} = k \\ 0 \text{ en otro caso} \end{cases}$$

2.4. Redes bayesianas en Ciencias de la Salud y de la Vida

La medicina frecuentemente se enfrenta a situaciones en que los modelos gráficos probabilísticos se ajustan bien para resolver problemas clínicos. En la práctica clínica se trata de comprender las estructuras de causalidad que dan lugar a un proceso fisiológico y, además, en medicina suele trabajarse con numerosas fuentes de incertidumbre, a menudo difíciles de minimizar por parte de los profesionales sanitarios. Por ello, desde su concepción, las redes bayesianas han estado vinculadas a la investigación médica.

El desarrollo de programas de diagnóstico basados en técnicas bayesianas comenzó en los años 60 en Estados Unidos y Gran Bretaña (Warner et al., 1961; Gorry y Barnett, 1968). Se trataba de sistemas que aplicaban el método probabilístico frecuentista, seleccionando una variable que recoge los posibles diagnósticos, y una serie de variables que en medicina suelen corresponderse con síntomas. En la década de los 80 se desarrollaron en medicina las redes bayesianas y los diagramas de influencia, desde su definición hasta el diseño de algoritmos eficientes para procesar la evidencia (Schwartz et al. 1988; Spiegelhalter y Knill-Jones, 1984). Los sistemas de diagnóstico probabilístico han tenido, desde entonces, una progresión exponencial en la literatura médica.

En todo modelo probabilístico aplicado al mundo real el investigador se enfrenta a la incertidumbre y la imprecisión, fundamentalmente por tres motivos: imprecisiones en la recogida de datos, indeterminismo del mundo real y deficiencias del propio modelo. En la práctica clínica, esto se traduce en historiales médicos incompletos, componentes subjetivos por parte del médico o el paciente, y mediciones erróneas o imprecisas (Díez, 1998).

Como hemos visto, existen dos paradigmas para la construcción de redes bayesiana: a partir de una base de datos, aplicando alguno de los métodos de aprendizaje de redes descritos anteriormente, o bien con la ayuda de expertos humanos, en el caso que nos ocupa profesionales sanitarios, que transmiten su conocimiento para introducir variables, relaciones y probabilidades condicionales en el modelo. También es posible combinar ambos enfoques, construyendo los modelos por métodos de aprendizaje pero eliminando o forzando la aparición de relaciones de acuerdo al criterio humano.

La forma más rápida de construir redes bayesianas en medicina consiste en tomar una base de datos que contenga un número suficientemente grande de observaciones y aplicar directamente métodos de aprendizaje de la red. Sin embargo, esta forma de proceder presenta limitaciones. A menudo, para cada observación se cuenta con unas pocas medidas además del diagnóstico. Estas variables pueden resultar insuficientes para determinar su independencia condicional, o incluso pueden no tener impacto en el diagnóstico desde el punto de vista médico.

Por lo anterior, es común contar con la opinión de expertos en el modelado. La construcción de una red bayesiana con criterio humano puede dividirse en dos fases. La primera de ellas consiste en recopilar la información cualitativa, identificando las variables de relevancia clínica y formando una red causal con las relaciones entre ellas. La segunda fase se ocupa de recoger la información cuantitativa: las probabilidades a priori, condicionadas o marginales.

En la actualidad, los modelos bayesianos en medicina se enfrentan a retos como el diseño de algoritmos que puedan abordar la complejidad exponencial de las redes bayesianas en tiempos y recursos computacionales razonables. Además, hay un interés creciente por ofrecer explicaciones intuitivas y comprensibles para la comunidad médica que utilizará los modelos a nivel usuario (Arora et al, 2019).

Sin embargo, las aplicaciones de las redes bayesianas en Ciencias de la Salud y de la Vida no se limitan al campo clínico. Con los avances en estudios genómicos en las tres últimas décadas, es cada vez más habitual utilizar modelos gráficos probabilísticos para predecir las relaciones de dependencia dentro de una red de genes; esto incluye distintas formas de influir en la expresión génica, tales como regulación positiva (estímulo positivo de la expresión de un gen), regulación negativa (estímulo negativo de la expresión de un gen) o supresión (Sachs et al, 2005). De forma paralela, también se han desarrollado redes bayesianas que tratan de explicar redes metabólicas por transformación e interacción de metabolitos (Correa y Goodacre, 2011), e incluso se han modelado relaciones de dependencia en tejidos celulares, lo que ha tenido amplias aplicaciones en el campo de la neurología (Bielza y Barranaga, 2014).

Por otro lado, también encontramos aplicaciones de redes bayesianas para modelar ecosistemas y en ciencias medioambientales (McCann et al, 2006). Son herramientas muy empleadas para representar

relaciones entre las especies y su hábitat, así como para predecir la viabilidad de las poblaciones, lo que ha definido parte de las políticas de protección medioambiental en Estados Unidos y Europa en las últimas décadas, aunque siempre en conjunción con pruebas de campo y simulaciones. Como en el resto de aplicaciones, la obtención de resultados en términos de probabilidad de hipótesis, y no de forma categórica, permite a los gestores barajar múltiples opciones a la hora de implementar medidas optimizando los recursos disponibles.

3. OBJETIVOS

El objetivo principal de nuestro trabajo es la construcción de redes bayesianas en ausencia de criterio experto a partir de conjuntos de datos para los cuales la bibliografía ha establecido, o realizado inferencias sólidas, sobre la estructura de dependencia de sus variables. Evaluaremos cómo abordar varios conjuntos de datos con distinta naturaleza y complejidad; la red ASIA, compuesta de variables binomiales, la red ALARM, compuesta por variables multinomiales (mayoritariamente con 3 ó 4 niveles) y la red SACHS, compuesta por variables gaussianas que, además, están presentadas en varios conjuntos de datos correspondientes a experimentos en los que se han realizado distintas perturbaciones por efectos externos, desplazando el gradiente de expresión génica a distintas regiones de la red.

Documentaremos nuestro uso de la herramienta *bnlearn* para el ajuste de las redes bayesianas. De esta forma, comprobaremos la aptitud de este paquete a la hora de representar fielmente la estructura causal subyacente, en función de la cantidad de variables y la naturaleza de los datos.

Respecto a los objetivos secundarios de este Trabajo de Fin de Máster, están:

- Evaluar cómo afectan al ajuste de estructura las diversas familias de algoritmos que incorpora *bnlearn*, observando potenciales diferencias según la cantidad y tipo de medidas. Asimismo, comprobar cómo la modificación de los parámetros de los algoritmos altera la precisión de la estructura de dependencia, tratando de hallar valores óptimos para los problemas planteados.
- Estudiar, en caso de trabajar con variables gaussianas, cómo afectan los distintos tipos de discretización a las estructuras causales propuestas, comparándolos además con las estructuras generadas con las variables sin discretizar.
- Plantear cómo afecta el remuestreo por metodología bootstrap para generar estructuras consenso a las redes generadas, y cómo construir estructuras consenso mediante inferencia bayesiana a partir de varias fuentes de información para nuestras variables, como es el caso de los 14 conjuntos de datos para la red SACHS.
- Obtener valores de puntuación de los parámetros para las redes ajustadas, valorando la variación entre sus magnitudes al realizar el ajuste por distintos métodos, además de su variación en función de la discretización a la hora de trabajar con variables a priori continuas.

En definitiva, realizaremos un análisis minucioso de los distintos métodos que ofrece *bnlearn* para la construcción de redes bayesianas y su aplicación a problemas reales englobados en Ciencias de la Salud y de la Vida. Exploraremos las limitaciones encontradas a la hora de representar fielmente la estructura de dependencia de las variables, así como las estrategias utilizadas para tratar de minimizarlas.

4. MATERIALES Y METODOLOGÍA

4.1. R y bnlearn

Existe un gran número de paquetes estadísticos en informática que incorporan funciones destinadas al aprendizaje de redes bayesianas. El software R es uno de los más utilizados para el modelado y tratamiento estadístico, y en el encontramos varios paquetes relacionados con la materia.

El paquete *bnlearn* (Scutari, 2010) es uno de los más empleados para el aprendizaje de la estructura causal, ajuste de parámetros, e inferencia a partir de los datos crudos. Este paquete implementa algoritmos basados en restricciones, basados en puntuaciones, e híbridos, para el aprendizaje de la estructura en redes discretas y gaussianas, así como una gama amplia de funciones de puntuación de estructuras y tests de independencia condicional.

Además, los clasificadores Naive Bayes y Tree-Augmented Naive Bayes (TAN) implementados en *bnlearn* permiten realizar predicciones de la estructura causal sobre variables de entrenamiento, y testear dichas predicciones sobre nuevas variables. Se incluyen algunas utlidades como comparación y manipulación de modelos, generación aleatoria de datos, tests de orientación de arcos y configuración de gráficos. Por último, *bnlearn* incorpora algoritmos para el ajuste de parámetros (ajuste bayesiano y máxima verosimilitud) e inferencia, consultas de probabilidad condicional, remuestreo bootstrap y consensuado del modelo.

4.2. Bases de datos y redes bayesianas

4.2.1. Red ASIA

El conjunto de datos ASIA, correspondiente al trabajo de Lauritzen y Spiegelhalter (1998), hace referencia a un problema teórico que los autores plantearon para ilustrar los métodos de cálculo local aplicados a estructuras gráficas probabilísticas.

El modelo parte de la premisa de que la disnea (dificultades respiratorias) puede deberse a tuberculosis, cáncer de pulmón o bronquitis, ninguna de ellas, o cualquier combinación entre las mismas. La estancia en Asia aumenta las probabilidades de contraer tuberculosis, mientras que el tabaquismo es un factor de riesgo para el cáncer de pulmón y la bronquitis. Los resultados puntuales de una radiografía pulmonar no logran discriminar entre cáncer de pulmón y tuberculosis. Tampoco lo hace la presencia o ausencia de disnea.

Lauritzen y Spiegelhalter analizaron la estructura causal desde el punto de vista de un médico que recibe en consulta a un paciente aquejado de disnea, habiendo visitado recientemente Asia. No es posible, por el momento, conocer el hábito tabáquico del paciente, ni someterlo a una prueba de rayos X.

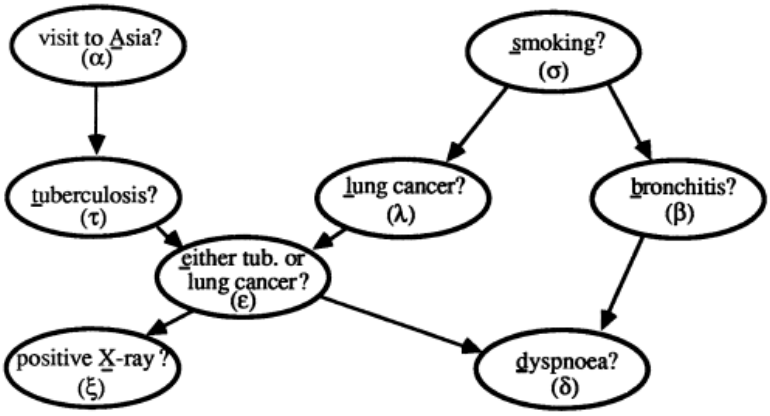


Fig. 4. Red causal en el ejemplo ficticio planteado por Lauritzen y Spiegelhalter (1998). Cada nodo tiene dos posibles estados: SÍ y NO.

El interés principal del doctor está en conocer las probabilidades de que el paciente haya contraído cada una de las enfermedades. Por otro lado, si la tuberculosis fuese descartada por otro test, sería conveniente analizar cómo cambiaría la confianza en el diagnóstico de cáncer de pulmón. Además, se pretende entender qué prueba diagnóstica contribuye más al conocimiento sobre la ausencia o presencia de cáncer, el historial de tabaquismo o la radiografía, teniendo en cuenta que la disnea podría venir por la bronquitis asociada al tabaco.

Pese a la sencillez de la estructura, el objetivo de los autores era plasmar las múltiples consultas que pueden surgir respecto a los elementos de la estructura de dependencia por parte de un especialista en el campo (Lauritzen y Spiegelhalter, 1998). Contamos con un conjunto de datos de 2000 observaciones y 8 variables.

4.2.2. Red ALARM

La red bayesiana ALARM (Beinlich et al., 1989) fue desarrollada en la Universidad de Stanford para ofrecer un sistema de alarma en pacientes ingresados en unidades de cuidado intensivo (UCIs). El mensaje de emergencia sobre el paciente monitorizado se basa en probabilidades para un diagnóstico diferencial, de acuerdo a la evidencia disponible.

Todo el conocimiento médico se encuentra comprendido en una estructura gráfica que interrelaciona 8 diagnósticos, 16 hallazgos clínicos y 13 variables intermedias. La red está compuesta de variables binomiales de tipo categórico (SÍ/NO) y, mayoritariamente, variables multinomiales consistentes en valores clínicos intervalizados (BAJO/MEDIO/ALTO, ocasionalmente también encontramos el nivel CERO).

La red ALARM contiene datos estadísticos basados en registros de pacientes, probabilidades fundamentadas en condiciones lógicas, y ciertos postulados subjetivos. Se trata de un sistema orientado a mediciones clínicas. Emulando un monitor de anestesia, ALARM registra una serie de medidas fisiológicas, las categoriza en intervalos de valores y genera mensajes de advertencia cuando alguna variable sale del rango normal.

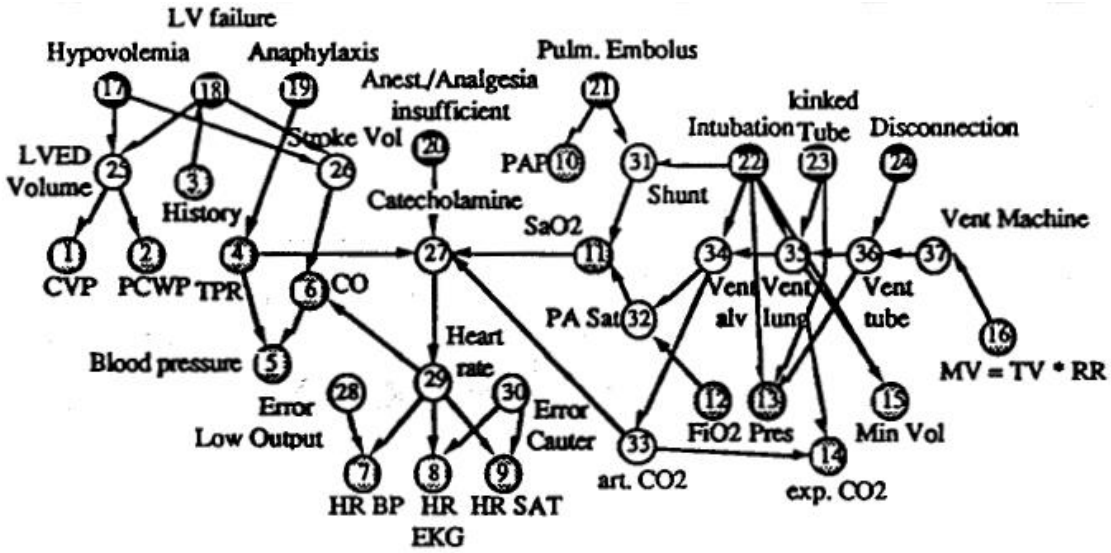


Fig. 5. La red ALARM representando la estructura causal de las variables. Se distinguen nodos diagnósticos, intermedios y de medidas clínicas.

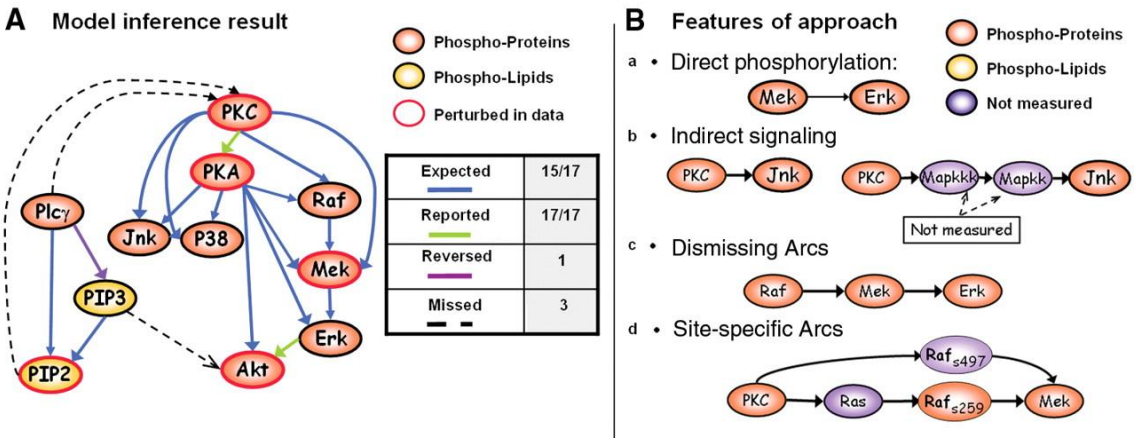
A partir de aquí, mediante el algoritmo de inferencia causal de Pearl, se implementa un sistema local de actualización de probabilidades a medida que se emiten mensajes. Cada nodo recibe mensajes sobre la información a priori de sus padres y de verosimilitud de sus hijos. Dichos mensajes se combinan para determinar la confianza en los niveles de cada nodo. Dicha confianza es enviada a nodos vecinos hasta actualizar la red completa.

La transmisión de información, cuando se dispone de observaciones o evidencias de variables determinadas, no se da de manera incontrolada; existe una serie de condiciones limitantes para la propagación: un nodo observado impide que se transmita información desde los hijos del nodo observado a los padres del nodo observado ni en el sentido inverso, un nodo observado no transmite información de un hijo a otro hijo, y un nodo no observado y sin ningún descendiente observado no transmite información de un padre a otro.

Nuestro conjunto de datos cuenta con 20000 observaciones y 37 variables.

4.2.3. Red SACHS

En el trabajo de Sachs et al. (2005), se medían simultáneamente 11 proteínas y fosfolípidos fosforilados derivados de miles de células del sistema inmune primario, bajo condiciones ambientales inespecíficas y ciertas intervenciones a nivel molecular. En el estudio observacional, se asegura que las rutas de señalización celular más importantes están activas, mientras que en los estudios experimentales, se realizan inferencias acerca de las relaciones de dependencia entre los metabolitos utilizando señales de estímulo o inhibición de compuestos.



Fig, 6. (A) Red inferida desde los datos de citometría de flujo en el trabajo de Sachs et al. (2005). Representa un modelo consenso de 500 resultados de alta puntuación. Se muestran los arcos de mayor confianza, que aparecen en, al menos, el 85% de las redes.

Los análisis realizados se pueden encontrar en el trabajo de Sachs et al. (2005). El estudio contaba con un conjunto de datos obtenido por observación, y 14 conjuntos registrados por inducción a estímulos específicos. Tanto para el conjunto de datos observacional como para los conjuntos de datos expermientales, contamos con 854 observaciones y 11 variables.

4.3. Construcción de redes

El paquete *bnlearn* nos permite crear redes bayesianas ad hoc en forma de objetos *bn*. Para ello, partimos de grafos vacíos a los que introducimos las relaciones de causalidad entre variables que la bibliografía respectiva ha determinado. Estas redes proporcionarán un estándar de referencia para comparar la bondad del ajuste de las redes bayesianas que construimos con los distintos algoritmos que incorpora *bnlearn*.

Como elemento descriptivo de la complejidad de la red, tomaremos la densidad (Δ) del grafo, que en grafos dirigidos queda definida como:

$$\Delta = \frac{|\boldsymbol{E}|}{|\boldsymbol{V}|(|\boldsymbol{V}|-1)}$$

Siendo $|\boldsymbol{E}|$ el número de arcos, y $|\boldsymbol{V}|$ el número de nodos.

Además, calcularemos el grado modal medio (número promedio de arcos por nodo, sean éstos efluentes o afluentes) de las redes.

El paquete auxiliar *Rgraphviz* (Hansen et al, 2020) provee de una interfaz gráfica en la que plasmar las estructuras causales, además de poder manipular el formato de la red. Utilizamos estas funcionalidades para denotar, en la red ASIA, las variables relacionadas con principios causales de la enfermedad, pruebas diagnósticas y potenciales enfermedades.

Al instanciar el objeto *bn*, comprobamos que la red bayesiana no sea cíclica. Además, generamos el correspondiente grafo moral y el CPDAG. Otros elementos descriptivos empleados para el análisis de la estructura son la ordenación de los nodos, y el vecindario y la manta de Markov para nodos en las regiones más densas de la red, además de las conexiones convergentes presentes en la red.

4.4. Preprocesado

4.4.1. Deduplicación

La deduplicación es un proceso de identificación de variables altamente correlacionadas dentro de un conjunto. Habitualmente, la deduplicación se emplea para detectar variables superfluas o redundantes que no aportan información sobre la estructura de dependencia, desechando una variable de cada par en que haya una alta correlación. En nuestro caso, la deduplicación tiene una función meramente descriptiva, puesto que al tomar nuestras variables de referencias bibliográficas que las incluyen por criterios clínicos o bioquímicos, partimos de la premisa de que ninguno de los nodos que conforman la red se encuentra incluido por asociaciones espúreas.

Imponemos un umbral de correlación, sobre el cual un par variables numéricas dentro de un conjunto de datos son consideradas solapantes. Utilizaremos la deduplicación sobre los conjuntos de datos SACHS observacionales y experimentales, imponiendo umbrales decrecientes de correlación. De esta manera, podemos observar en qué punto comienza a verse información solapante.

4.4.2. Discretización

La discretización proporciona una alternativa para trabajar con variables continuas; éstas son convertidas a variables multinomiales. La transformación a factores puede llevarse a cabo por diversos algoritmos: por defecto, la factorización se realiza por valores cuantiles, conteniendo los valores en intervalos de ancho de cuantiles equivalentes. Sin embargo, también es posible factorizar por intervalos de valores en términos absolutos o mediante el algoritmo de Hartemink; este método se fundamenta en generar intervalos que minimicen la pérdida de información mutua entre parejas de variables continuas:

$$I(X;Y) = \sum_x \sum_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

Utilizaremos los tres métodos de discretización sobre los datos observacionales de SACHS para observar las variaciones en cada uno de ellos respecto a los datos sin discretizar a la hora de ajustar la estructura y los parámetros. Emplearemos números decrecientes de intervalos para valorar cómo afecta la pérdida de información en las variables al ajuste de la red, en comparación con la red gaussiana.

4.5. Ajuste de estructura

Para el ajuste de las respectivas estructuras de red, emplearemos las familias de algoritmos descritas anteriormente. Los algoritmos utilizados devolverán la clase de equivalencia de las redes.

4.5.1. Algoritmos basados en restricciones

Los algoritmos basados en restricciones implementados en *bnlearn* y utilizados para el ajuste de estructura incluyen:

- Algoritmo *Growth-Shrink* (GS)
- Algoritmo *Incremental Association* (IAMB)
- Algoritmo *PC* (pc stable)
- Algoritmo *Min-Max Parents & Children* (MMPC)

Además de los diversos algoritmos, podemos realizar diversas modificaciones en los algoritmos para aproximar el resultado a la red original propuesta en el trabajo de Sachs.

En primer lugar, los tests de independencia condicional, sobre los cuales se valida la relación entre variables, pueden influir sobre el resultado de la estructura causal. Cuando trabajamos con variables multinomiales, contamos con los siguientes métodos:

- Tests *mutual information*: se emplean en redes bayesianas discretas o gaussianas. Se trata de medidas teóricas de la distancia entre variables. Son proporcionales a la razón de verosimilitud logarítmica, se basan en la devianza de los modelos analizados. Suponiendo que tenemos las variables *X* e *Y*, con *R* y *C* niveles respectivamente, condicionalmente dependientes de la variable *Z*, con *L* niveles. La información mutua de X e Y sigue la expresión:

$$MI(X,Y|Z) = \sum_{i=1}^{R}\sum_{j=1}^{C}\sum_{k=1}^{L}\frac{n_{ijk}}{n}\log\frac{n_{ijk}n_k}{n_{ik}n_{jk}}$$

  Siendo *n* el número de observaciones en cada nivel de las variables. Por defecto, el test utilizado es el de la χ^2 asintótica (*mi*, *mi-g* en el caso de gaussianas); otro método que utilizaremos es el de las permutaciones de Monte Carlo.
- Tests *shrinkage estimator*, que ajustan la χ^2 asintótica mediante estimadores de la información mutua.
- Tests *Pearson's* X^2, disponibles para redes discretas, se basan en tablas de contingencia para la X^2 de Pearson. Utilizaremos el test de la χ^2 asintótica y las permutaciones de Monte Carlo.

- Tests de correlación lineal de Pearson, disponibles para redes gaussianas. Utilizaremos el test de la t de Student (*cor*), además de las permutaciones de Monte Carlo (*mc-cor*).

$$X^2(X,Y|Z) = \sum_{i=1}^{R}\sum_{j=1}^{C}\sum_{k=1}^{L}\frac{(n_{ijk}-m_{ijk})^2}{m_{ijk}}$$

Donde:

$$m_{ijk} = \frac{n_{ik}n_{jk}}{n_k}$$

En ambos casos, contrastamos la hipótesis nula de independencia empleando la distribución asintótica $\chi^2_{(R-1)(C-1)L}$ o la permutación de Monte Carlo descrita en Edwards (2000).

Cuando trabajamos con variables normales, utilizamos los siguientes tests:

- El t-test exacto para el coeficiente de correlación de Pearson de *X* e *Y* dado un conjunto de variables $\boldsymbol{Z}$ ($\rho_{XY|\mathbf{Z}}$), definido como:

$$t(X,Y|\boldsymbol{Z}) = \rho_{XY|\boldsymbol{Z}}\sqrt{\frac{n-2}{1-\rho^2_{XY|\boldsymbol{Z}}}} \sim t_{n-|\boldsymbol{Z}|-2}$$

Sigue una distribución t de Student con n - $|\boldsymbol{Z}|$ - 2 grados de libertad.

- El test Z de Fisher, una transformación del coeficiente de correlación lineal $\rho_{XY|\mathbf{Z}}$ con una distribución asintótica normal. Definido como:

$$Z(X,Y|\boldsymbol{Z}) = \frac{\sqrt{n-|\boldsymbol{Z}|-3}}{2}\log\frac{1+\rho_{XY|\boldsymbol{Z}}}{1-\rho_{XY|\boldsymbol{Z}}}$$

Para ambos tests, también empleamos las permutaciones de Monte Carlo, como las descritas en Legendre (2000).

En aquellos tests que exijan determinar el número de permutaciones, utilizaremos cantidades crecientes para observar cómo varía la estructura de red, valorando en qué punto se alcanza la asíntota en la bondad del ajuste.

De entre los otros elementos que podemos modificar en el ajuste de la red, uno de los más destacables es el nivel de confianza 1-α, modificando el valor α, es decir, el nivel de significación, que se corresponde con la proporción asumida del error de tipo I (rechazar la hipótesis nula cuando esta es cierta, lo que en redes bayesianas se traduciría en declarar arcos cuando no existe una verdadera relación de dependencia entre los

dos nodos) en el total de análisis. Introduciremos valores crecientes de α para observar cómo afecta al ajuste de red una mayor permisividad a la hora de colocar arcos entre variables.

Se tratará de forzar la orientación de los arcos en aquellos indeterminados para mejorar la comparativa entre las redes ajustadas y la estructura de datos real.

4.5.2. Algoritmos basados en puntuaciones

El paquete *bnlearn* incorpora dos algoritmos basados en puntuaciones que utilizaremos para el ajuste de estructuras:

- Algoritmo *Hill-Climbing*, explora el espacio del DAG por adición, inversión y eliminación de arcos.
- Algoritmo *Tabu Search*: se trata de una modificación del algoritmo *Hill-Climbing* en que se trata de huir de optimizaciones locales penalizando lo menos posible la puntuación global.

Además, contamos con distintos criterios para calcular la puntuación de una red:

- *Bayesian Information Criterion*, es el que se aplica por defecto a redes tanto discretas como continuas.
- *Akaike Information Criterion*.
- Logaritmo del *Bayesian Dirichlet equivalent* o *Bayesian Gaussian equivalent*.
- El *log-likelihood* multinomial (loglik, loglik-g para redes gaussianas) para variables discretas.

Los valores de puntuación tienen un sentido orientativo para comprobar la aptitud de la red, pero no son un criterio principal a la hora de determinar con qué precision imita la estructura causal de la realidad.

4.5.3. Algoritmos híbridos

También contamos con algoritmos que combinan criterios de restricción y de puntuación, de los cuales emplearemos:

- *Max-Min Hill-Climbing* (MMHC), un algoritmo híbrido que combina los procedimientos MMPC para restringir el espacio de búsqueda y HC para optimizar la estructura dentro del espacio.

El algoritmo admite los parámetros de restricción y puntuación de los apartados anteriores. Además, admite, de forma adaptada, los criterios de puntuación expuestos en el apartado anterior.

4.5.4. Remuestreo bootstrap y red consenso

En casos en que los algoritmos de inferencia de estructura no permiten emular con precisión la estructura causal real del problema, la metodología bootstrap es una herramienta que nos permite generar redes con un alto nivel de confianza. Para ello, generamos muestras con reemplazo en las observaciones aplicando la

metodología bootstrap a las observaciones de nuestros conjuntos de datos, obteniendo una medida de fuerza (*strength*) que expone en qué proporción se encuentra un determinado arco en el conjunto de redes generadas por remuestreo, además de una medida de direccionalidad que describe en qué proporción se encuentra el arco en relación con el arco inverso.

A partir de las medidas que obtenemos del remuestreo, generamos una red consenso que contenga los arcos que superen un determinado umbral de fuerza. Si hubiera varios arcos que cumplieran este requisito y provocaran conflictos por generar ciclos en el grafo, descartaremos aquellos que posean un menor valor en la direccionalidad.

Imponemos una cantidad de 500 réplicas por remuestreo bootstrap con reemplazamiento en las observaciones para cada red, con el mismo tamaño que la red original. Evaluaremos la configuración de la red consenso en relación a la estructura causal real bajo distintos umbrales de fuerza para los arcos.

4.5.5. Fusión de redes

En el caso de la red SACHS, contamos con un conjunto de datos observacional y 14 conjuntos de datos correspondientes a estímulos específicos sobre la cascada de proteínas. Para integrar los resultados que obtengamos del ajuste de red para cada uno de estos conjuntos, implementaremos los algoritmos propuestos en el trabajo de Córdoba, Bielza y Larrañaga (2018), *GBNFuseSVote* y *GBNFuseSInter*, para redes bayesianas gaussianas y aplicaremos metodología propia para condensar redes bayesianas discretas.

4.5.6. Comparación con la estructura causal real

Para evaluar la validez de las redes generadas respecto a la estructura causal real, contamos con grafos comparativos entre ambos en que se ven los arcos faltantes o sobrantes de nuestra red. Además, las comparaciones contra la estructura causal construida ad hoc nos devuelven cuantificaciones de la semejanza en forma de número de verdaderos positivos (arcos coincidentes), falsos positivos (arcos espúreos que no están presentes en la red original) y falsos negativos (arcos no declarados que están presentes en la red original).

Para seleccionar el algoritmo más adecuado, elegimos aquel cuya estructura minimizaba la llamada SHD (distancia estructural de Hamming), esto es, el número de inserciones, eliminaciones o inversiones de arcos necesarias para convertir la red aprendida en la red ALARM original. La SHD es, por tanto, la suma de falsos positivos y falsos negativos de la red.

4.6. Ajuste de parámetros

Al realizar el ajuste de parámetros, en el caso de redes bayesianas discretas utilizamos el conocimiento previo acerca de la estructura de dependencia y los datos de que disponemos para generar, para cada nodo,

su tabla de contingencia de probabilidades para cada valor respecto de los valores de su nodo o nodos parentales. En el caso de redes bayesianas gaussianas, obtenemos los estimadores de los coeficientes de regresión que estiman el valor del nodo condicionado a los valores de sus nodos parentales.

La estimación paramétrica puede realizarse por dos métodos:

- Máxima verosimilitud (*mle*)
- Estimación bayesiana (*bayes*, sólo para datos discretos).

Respecto a la red fusionada en SACHS, para la agregación de parámetros en la red consenso utilizaremos el método expuesto por Córdoba *et al* (2015), según el cual se sintetiza un estimador de los coeficientes de regresión en cada una de las redes individuales. Para ello, si se desea agregar el coeficiente de regresión de una variable X_i sobre una variable X_j, se realiza una media ponderada de los coeficientes de regresión para cada red en el conjunto K con pesos w_{ijk}:

$$\tilde{\beta}_{ij} = \sum_{k=1}^{K} w_{ijk} \hat{\beta}_{ijk}$$

Donde el peso de cada red está determinado por la inversa de la varianza en los errores del nodo filial (r_{ijk}) en cada red entre el sumatorio de esta medida en todas las redes:

$$w_{ijk} = \frac{r_{ijk}^{-1}}{\sum_{k=1}^{K} r_{ijk}^{-1}}, r_{ijk} = \lambda_{ijk}\sigma_{jk}$$

5. RESULTADOS Y DISCUSIÓN

A continuación, se muestran los resultados de los procesos de aprendizaje de las redes ASIA, ALARM y SACHS. Esto incluye el proceso de construcción de la red, comparaciones con la estructura consensuada en la literatura previa y el aprendizaje de los parámetros.

5.1. Red ASIA

En este apartado mostramos los resultados del ajuste y evaluación de la red ASIA, así como del aprendizaje de los parámetros de la red.

5.1.1. Construcción de la red

La red construida a priori, de acuerdo a la estructura de dependencia definida por Lauritzen y Spiegelhalter (figura 4), fue sintetizada e instanciada como objeto de red bayesiana (figura 7):

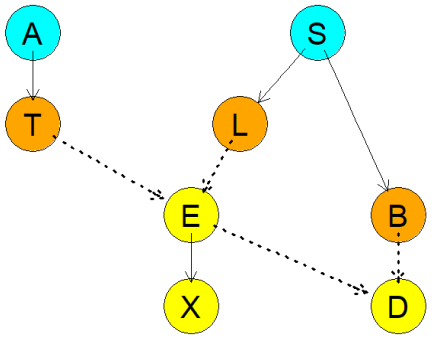


Fig. 7. Estructura ad hoc de la red ASIA, construida con base en la red propuesta por Laurintzen y Spiegelhalter (1988). Por colores, se clasifican las variables según su naturaleza. En azul celeste, las variables relacionadas con principios causales: estancia reciente en Asia (*A*) y hábito tabáquico (*S*). En naranja, las variables relacionadas con posibles enfermedades: tuberculosis (*T*), cáncer de pulmón (*L*) y bronquitis (*B*). En amarillo, las variables relacionadas con elementos diagnósticos: dicotomía tuberculosis/cáncer (*E*), rayos X (*X*) y disnea (*D*). Los arcos pertenecientes a estructuras convergentes se representan como líneas discontinuas.

Se trata de un grafo de baja densidad (Δ=0.14) y que integra un número muy reducido de nodos (8) y arcos (8). Únicamente encontramos un punto de mayor densidad en la variable correspondiente a la dicotomía tuberculosis/cáncer (*E*). A ella llegan dos arcos, formando una estructura convergente con centro en *E*, y de ella salen dos arcos, uno de los cuales es a su vez parte de otra estructura convergente. La importancia de este nodo en el ajuste de la red puede apreciarse al derivar el grafo moral y el CPDAG de la red ASIA.

En el grafo moral, el nodo *E* se encuentra conectado por hasta 5 aristas, 2 más que los siguientes nodos más poblados. Como vimos anteriormente, la estructura del grafo moral es de gran importancia en algoritmos de aprendizaje de red basados en restricciones, puesto que las relaciones de independencia condicional se establecen podando aquellos arcos entre nodos en que no se identifica una relación de dependencia.

A B

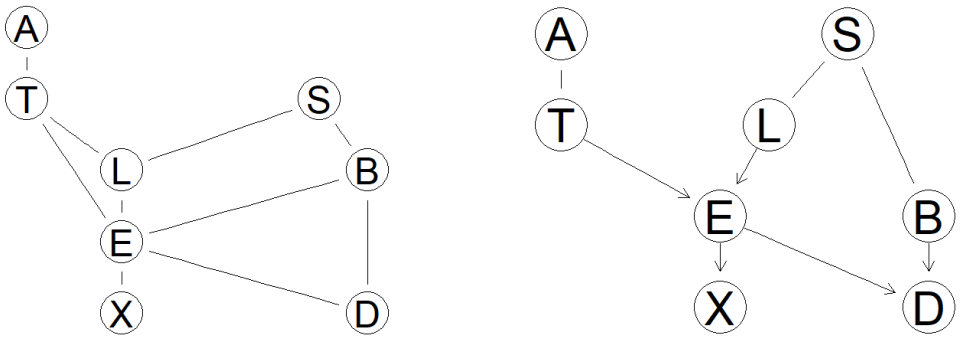


Fig. 8. Grafo moral (A) y CPDAG (B) derivados de la red ASIA.

En cuanto al CPDAG, todos los arcos que conectan *E* con otros nodos son direccionales, y estos comprenden 4 de los 5 arcos direccionales presentes en la red.

5.1.2. Ajuste de la red

En la tabla 1, pueden verse los resultados de los algoritmos aplicados sobre nuestros conjuntos de datos al ser comparados con la red ASIA.

Tabla 1. Comparativa de redes ajustadas con la red ASIA. Algoritmos basados en restricciones: GS (Growth-ShrinK), IAMB (Incremental-Association Markov Blanket), PC, MMPC (Min-Max PC). Algoritmos basados en puntuaciones: HC (Hill-Climbing), Tabu (Tabu Search). Algoritmos híbridos: MMHC (Min-Max Hill-Climbing).

	Algoritmos basados en restricciones				**Algoritmos basados en puntuaciones**		**Algoritmos híbridos**
	GS	**IAMB**	**PC**	**MMPC**	**HC**	**Tabu**	**MMHC**
Verdaderos positivos	3	2	2	2	7	6	5
Falsos positivos	1	1	1	1	0	1	0
Falsos negativos	4	5	4	4	1	1	3

En primer lugar, el ajuste de la red ASIA empleando algoritmos basados en restricciones no demostró un rendimiento adecuado. Los resultados obtenidos a partir del algoritmo Growth-Shrink no lograron replicar con fidelidad la estructura de la red. Únicamente se generaron dos estructuras convergentes, una de las cuales incluyó además un arco invertido respecto a la estructura causal real (figura 9).

A B

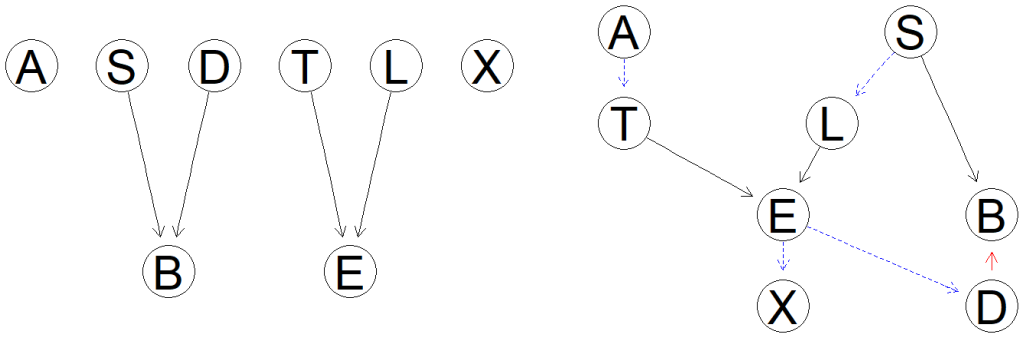


Fig. 9. Estructura ajustada mediante algoritmo Growth-Shrink (A) y comparación con la red causal real (B). Las flechas discontinuas azules indican arcos ausentes en la red ajustada, y las flechas rojas indican arcos espúreos.

El algoritmo Hill-Climbing resultó mucho más efectivo a la hora de replicar la estructura de la red que describe la bibliografía: únicamente encontramos una ausencia en el arco que conecta la estancia en Asia con la incubación de tuberculosis (figura 10).

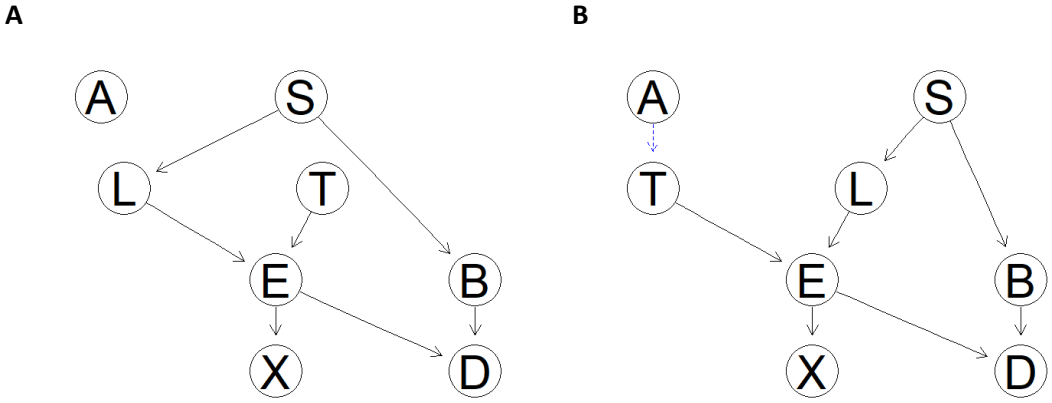


Fig. 10. Estructura ajustada mediante algoritmo Hill-Climbing (A) y comparación con la red causal real (B). Las flechas discontinuas azules indican arcos ausentes en la red ajustada, y las flechas rojas indican arcos espúreos.

En cuanto a la posibilidad de emplear algoritmos híbridos que combinen ambos enfoques, el algoritmo Min-Max Hill-Climbing no consiguió mejorar la predicción ofrecida por Hill-Climbing. Además de no detectar el arco conector de A con el resto de la red, se desprende de los arcos salientes del nodo E.

De estos resultados, podría inferirse que los algoritmos basados puramente en puntuaciones son los más efectivos a la hora de modelar la red ASIA. Los algoritmos basados en restricciones, que frecuentemente comienzan sintetizando redes locales, resultaron ineficientes a la hora de condensar éstas en redes globales, dando como resultado subconjuntos aislados de la estructura de dependencia como los vistos en la figura 9.

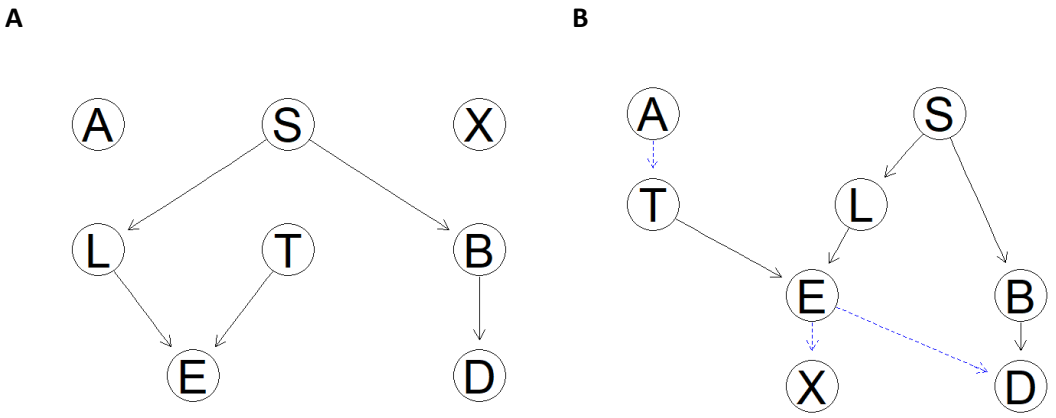


Fig. 11. Estructura ajustada mediante algoritmo Min-Max Hill-Climbing (A) y comparación con la red causal real (B). Las flechas discontinuas azules indican arcos ausentes en la red ajustada, y las flechas rojas indican arcos espúreos.

Respecto al arco faltante en la estructura obtenida por algoritmos de puntuación que conecta la estancia en Asia con el resto del grafo, la aplicación de la metodología bootstrap al método Hill-Climbing permitió la detección de la asociación entre la estancia en Asia y el desarrollo de tuberculosis. De acuerdo a referencias bibliográficas tales como el trabajo de Sachs et al (2005), impusimos un umbral en el ratio de aparición de un arco de 0.85 para que éste sea incluido en la red consenso. Sin embargo, no se debe obviar que la naturaleza de un problema de base clínica como es el caso de ASIA es lo suficientemente distinta de la investigación básica que planteaba el trabajo de Sachs como para que este umbral no resulte completamente extrapolable, si bien tiene un alto interés orientativo en ausencia de umbrales del mismo tipo en la bibliografía para casos clínicos.

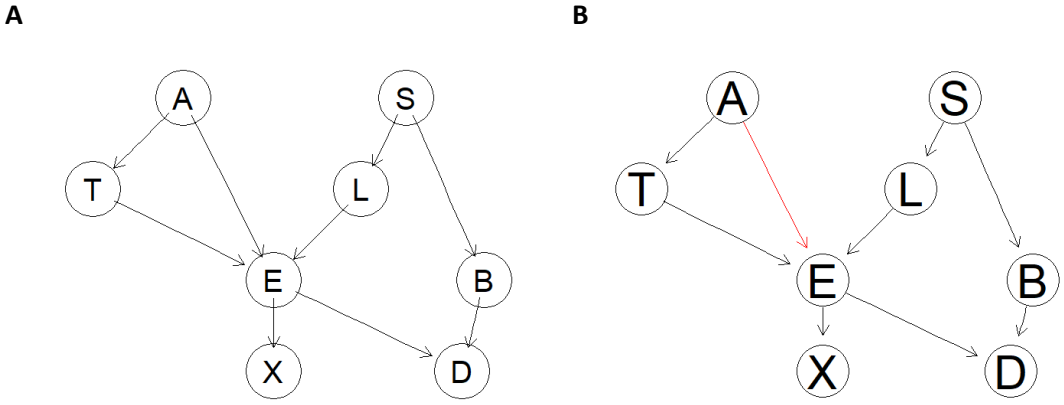


Fig. 12. Estructura consenso ajustada mediante algoritmo Hill-Climbing con metdología bootstrap y umbral de aparición de 0.85 (A) y comparación con la red causal real (B). Las flechas discontinuas azules indican arcos ausentes en la red ajustada, y las flechas rojas indican arcos espúreos.

La red consenso generada a partir del remuestreo bootstrap incluyó todos los arcos presentes en la estructura planteada por Lauritzen y Spiegelhalter. Sin embargo, introduce además un arco espúreo que une la estancia en Asia (*A*) con la dicotomía entre tuberculosis o cáncer de pulmón/bronquitis (figura 12). En este caso, *A* actúa como variable confusora del efecto de *T* sobre *E*.

Tabla 2. Fuerza y direccionalidad de los enlaces que integran la red consensuada por Hill Climbing con remuestreo bootstrap y umbral de fuerza de enlace superior a 0,85. En rojo, aparecen resaltados los arcos espúreos.

Desde	Hacia	Direccionalidad	Fuerza
Estancia en Asia (A)	Tuberculosis o cáncer/bronquitis (E)	0.988	0.958
Estancia en Asia (A)	Tuberculosis (T)	0.541	0.996
Tabaquismo (S)	Cáncer de pulmón (L)	0.683	0.996
Tabaquismo (S)	Bronquitis (B)	0.799	1.000
Tuberculosis o cáncer/bronquitis (E)	Rayos X (X)	0.922	1.000
Tuberculosis (T)	Tuberculosis o cáncer/bronquitis (E)	0.953	1.000
Cáncer de pulmón (L)	Tuberculosis o cáncer/bronquitis (E)	0.976	1.000
Bronquitis (B)	Disnea (D)	1.000	1.000
Tuberculosis o cáncer/bronquitis (E)	Disnea (D)	1.000	1.000

Cuando observamos las características de los arcos que conforman la red consenso en términos de fuerza de enlace (ratio de aparición del arco) y direccionalidad (proporción de arcos entre dos nodos que se encuentran en una dirección determinada), vemos que el arco espúreo que va desde *A* hacia *E* es el que tiene menos fuerza de enlace en la red consenso (tabla 2).

Por lo tanto, sería posible podar este arco de la red elevando el umbral de aparición. De esta forma, si impusiéramos una fuerza de enlace umbral $\theta_{strength} \in (0.958, 0.996]$ para la red consenso, la estructura coincidiría plenamente con la red ASIA planteada por Lauritzen y Spiegelhalter. Sin embargo, imponer un umbral ad hoc para equiparar nuestra red a la estructura original se desvía del objetivo de emplear criterios estrictamente estadísticos para construir la red bayesiana. Pueden existir discrepancias entre nuestros resultados y las redes presentes en la literatura, debidas a que no contamos con criterio experto ni utilizamos conjuntos de datos absolutamente fieles a los originales.

En la tabla 3 puede verse el valor orientativo de la puntuación de la red para evaluar su aptitud. Si se observa el valor de cada uno de los criterios de puntuación a lo largo de todos los algoritmos utilizados, vemos que la red ASIA presenta puntuaciones muy similares a las de la red obtenida por algoritmo Hill-Climbing (en que había un arco faltante de la red ASIA) y a la red consenso obtenida por remuestreo bootstrap mediante el mismo algoritmo con una fuerza de enlace umbral de 0.85 (en que existía un arco espúreo). Sin embargo, las disminuciones son apreciables cuando se compara la red ASIA con algoritmos basados en restricciones, ya sea total (GS) o parcialmente (MMHC). En consecuencia, presumiblemente la puntuación de red es un indicador aceptable de la bondad de ajuste de la estructura cuando se trabaja con un conjunto reducido de variables multinomiales.

Tabla 3. Puntuaciones de red bajo distintos criterios para la estructura original ASIA y las estructuras aprendidas mediante algoritmo Growth-Shrink (GS), Hill-Climbing (HC), Min-Max Hill Climbing (MMHC) y Hill-Climbing con remuestreo bootstrap (HC-Bs).

	Original / HC-Bs (0.995)	GS	HC	MMHC	HC-Bs (0.85)
BDe	-11095.82	-12517.55	-11095.79	-12156.47	-11097.07
Log-Likelihood	-11033.09	-12466.88	-11034.90	-12106.64	-11033.09
AIC	-11051.09	-12480.88	-11051.90	-12120.64	-11055.09
BIC	-11109.74	-12526.50	-11107.29	-12166.26	-11126.78

5.1.3. Ajuste de parámetros

Los parámetros de la red fueron ajustados mediante estimación de máxima verosimilitud. En la figura 13, se exhiben histogramas de las probabilidades marginales de cada uno de los niveles de las variables. Inicialmente, el conjunto de la red viene condicionado por un porcentaje muy reducido de pacientes que visitaron Asia recientemente. En la tabla anexa 1, se pueden observar las distribuciones condicionales de los nodos

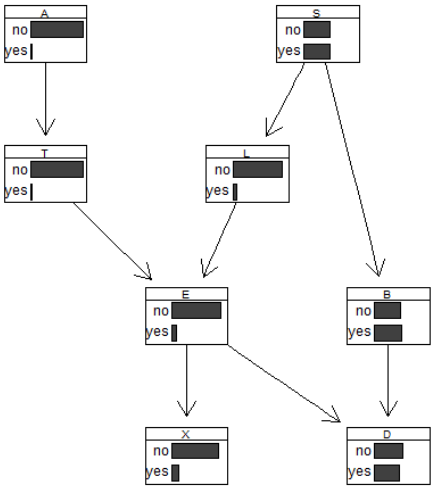


Fig. 13. Histogramas de probabilidad marginal para las variables de la red ASIA.

Por lo que vemos, es frecuente en la red que el nodo padre transmita sus distribuciones de probabilidad marginal a los nodos hijos. Sin embargo, esta tendencia se rompe en dos enlaces: respecto a la relación del tabaquismo (*S*) con el cáncer de pulmón (*L*), si bien es cierto que la ocurrencia de cáncer es muy superior en fumadores que en no fumadores ($OR_{S\text{-}L}$ =9.61, $IC_{95\%}(OR_{S\text{-}L})$: 6.69 – 14.21, p-value < 2.2e-16), es una ocurrencia relativamente poco frecuente en ambos casos (0.013 vs 0.118) por lo que, pese a que las subpoblaciones de fumadores y no fumadores están equilibradas, hay una incidencia baja de cáncer de pulmón.

Por otro lado, la variable dicotómica E toma el valor NO cuando se da la intersección de que el paciente no exhibe tuberculosis ni cáncer de pulmón, y el valor SÍ en caso contrario. Dado que son dos enfermedades de baja prevalencia, la distribución de *E* se ve desplazada mayoritariamente hacia el valor NO. Pese a ello, la distribución de probabilidades en la disnea (*D*) se encuentra más equilibrada. Esto se debe al efecto conjunto de *E* con la presencia de bronquitis (*B*), y a la interacción de efectos entre *E* y *B*. Mientras que, cuando existe bronquitis, la distribución de probabilidades de disnea se mantiene constante independientemente de la presencia de cáncer de pulmón o tuberculosis, en ausencia de bronquitis hay una diferencia notable de probabilidades en función de *E*; mientras que con un valor afirmativo de tuberculosis o cáncer las probabilidades de disnea son de 0.72, este valor baja a 0.10 cuando *E* es negativo.

5.2. Red ALARM

En este apartado mostramos los resultados del ajuste y evaluación de la red ALARM, así como del aprendizaje de los parámetros de la red.

5.2.1. Construcción de la red

En la figura 14 vemos la red bayesiana construida de acuerdo a los planteamientos de ALARM. La estructura consta de 46 arcos y 37 nodos, teniendo una densidad de Δ=0,03 y un grado modal medio de 2,49.

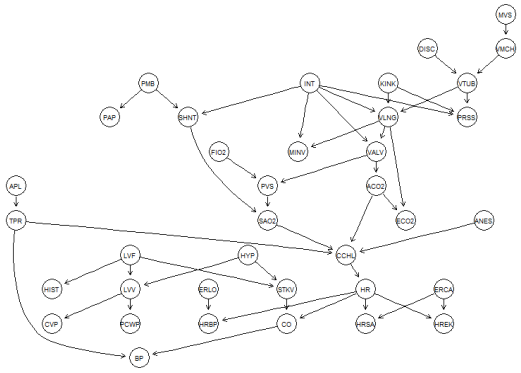


Fig. 14. Estructura ad hoc de la red ALARM, construida con base en la red descrita en Beinlich et al (1989).

Se trata de la red más extensa y compleja de cuantas se presentan en este trabajo, por lo que ofrece la oportunidad de evaluar el comportamiento de la estructura en función del tipo de algoritmo y los parámetros ajustados.

5.2.2. Ajuste de estructura

En la tabla 4 se puede ver el comportamiento de la red ajustada por diversos algoritmos basados en puntuaciones, basados en restricciones e híbridos.

Tabla 4. Comparativa de redes ajustadas con la red ALARM. Algoritmos basados en restricciones: GS (Growth-ShrinK), IAMB (Incremental-Association Markov Blanket), PC, MMPC (Min-Max PC). Algoritmos basados en puntuaciones: HC (Hill-Climbing), Tabu (Tabu Search). Algoritmos híbridos: MMHC (Min-Max Hill-Climbing).

	Algoritmos basados en restricciones				**Algoritmos basados en puntuaciones**		**Algoritmos híbridos**
	GS	**IAMB**	**PC**	**MMPC**	**HC**	**Tabu**	**MMHC**
Verdaderos positivos	5	16	30	0	22	26	15
Falsos positivos	14	18	12	32	31	25	17
Falsos negativos	41	30	16	46	24	20	31

Si bien en términos generales los algoritmos basados en puntuaciones parecen reflejar de forma más fiel la estructura de la red ALARM, el algoritmo PC resulta ser el más adecuado.

A diferencia del resto de algoritmos basados en restricciones utilizados, PC no se basa en detección de mantas de Markov, sino que parte del grafo saturado y elimina arcos sobrantes a medida que verifica las

hipótesis de probabilidad condicional. En consecuencia, no presenta las limitaciones que sí vimos en la red ASIA para aglutinar redes regionales, y refleja mejor la complejidad de la red (figura 15).

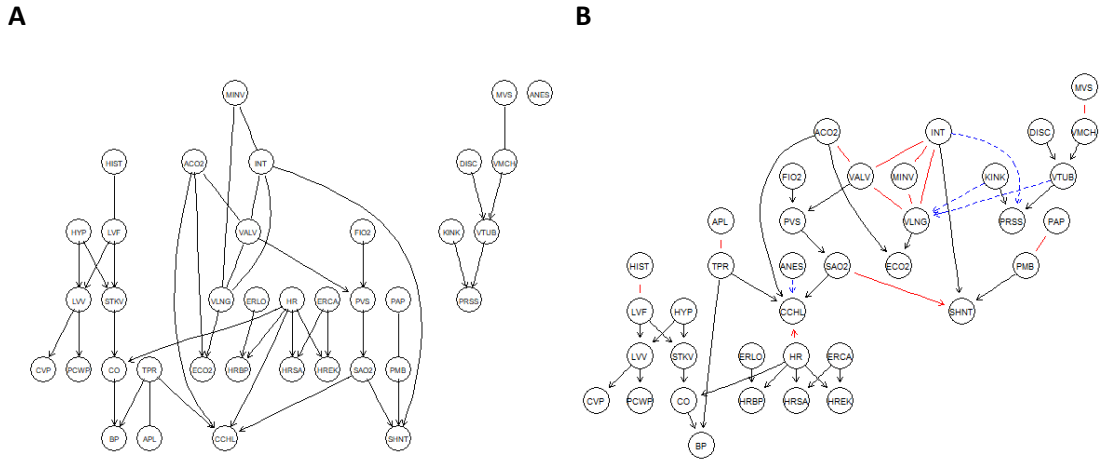


Fig. 15. Estructura ajustada mediante algoritmo PC (A) y comparación con la red causal real (B). Las flechas discontinuas azules indican arcos ausentes en la red ajustada, y las flechas rojas indican arcos espúreos.

A continuación, mostramos los resultados del ajuste de red aplicando el algoritmo PC en función del tipo de test de independencia condicional utilizado (tabla 5). El test que se había empleado anteriormente por defecto fue la prueba X^2 de Pearson asintótica, sobre la cual se evaluaron como alternativas la misma prueba empleando permutaciones de Monte-Carlo y el estimador por *mutual-information shrinkage*.

Tabla 5. Comparativa de redes ajustadas con la red ALARM por algoritmo PC según distintos tests de independencia condicional.

	X^2 de Pearson (Asintótica)	**X^2 de Pearson (Permutación de Monte Carlo)**	**Mutual-information shrinkage estimator**
Verdaderos positivos	30	38	30
Falsos positivos	12	6	12
Falsos negativos	16	8	16

Se puede observar que, tanto para el test X^2 de Pearson asintótico como para el estimador de información mutua el resultado en cuanto a SHD es el mismo. Sin embargo, las permutaciones de Monte Carlo aplicadas al test X^2 de Pearson resultaron en una estructura de red más fidedigna a la estructura original (figura 16). De acuerdo a estos resultados, cuando se trabaja con redes discretas con un gran número de variables las diferencias en el ajuste no vendrían dadas por el algoritmo en el test de independencia, sino por la asunción de resultados asintóticos o la búsqueda de métodos de remuestreo alternativos. La decisión de tomar cualquiera de estas vías se puede fundamentar en el número de observaciones (2000 en nuestro caso) y las distribuciones de probabilidad de cada una de las variables.

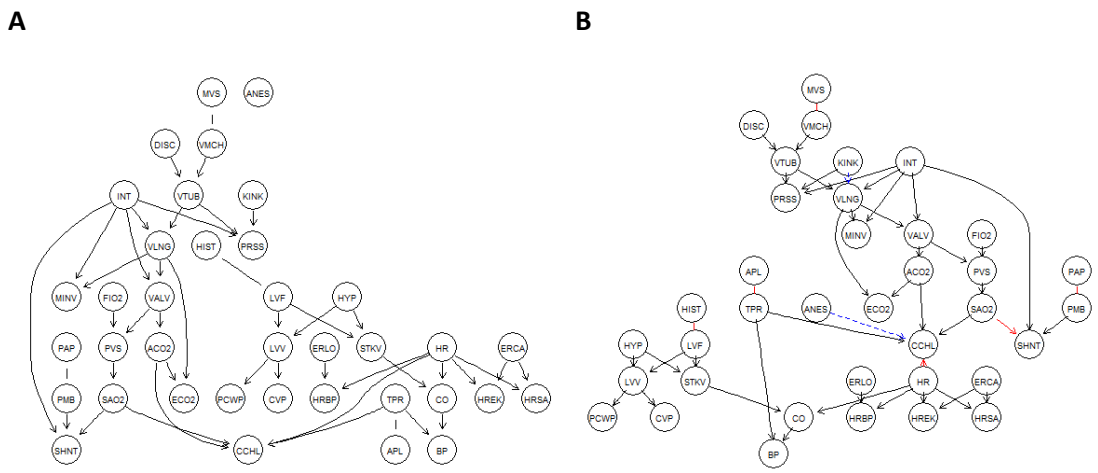


Fig. 16. Estructura ajustada mediante algoritmo PC con test X^2 de Pearson por permutaciones de Monte Carlo (A) y comparación con la red causal real (B). Las flechas discontinuas azules indican arcos ausentes en la red ajustada, y las flechas rojas indican arcos espúreos.

Los algoritmos de ajuste de redes bayesianas en *bnlearn* presentan por defecto un nivel de significación α=0,05. En la tabla 6, mostramos los resultados del aprendizaje de la red variando el nivel de significación.

Tabla 6. Comparativa de redes ajustadas con la red ALARM por algoritmo PC según valores para el nivel de significación.

	0.005	**0.01**	**0.05**	**0.1**	**0.2**
VP	30	36	38	30	39
FP	12	6	6	14	5
FN	16	10	8	16	7

La red aprendida alcanza su mínima SHD con la estructura real de ALARM con el nivel de significación α=0,2. El nivel de significación está relacionado con el grado de confianza de cada arco, por lo que alcanzamos la máxima verosimilitud en la red con una confianza de 0,8. El nivel de significación representa, además, el ratio asumido de errores de tipo I que se encontrará a la hora de declarar las relaciones de dependencia condicional entre variables.

Hay que tener en cuenta que las restricciones en las redes bayesianas, al ser por definición DAGs, provocan que la adición, eliminación o inversión de un arco en función del grado de confianza impuesto en el algoritmo condicione la aparición de otros arcos, por la condición de aciclicidad en el grafo, lo que puede resultar en una estructura notablemente distinta.

En la figura 17 aparece reflejada la estructura generada con α=0,2. Lo más notable en lo respectivo a los falsos positivos es que para algunos de los arcos en la red aprendida, si bien éstos existen en la red original, no se ha conseguido determinar su direccionalidad, apareciendo como aristas. Por otra parte, el nodo ANES se encuentra completamente desligado del resto de la red.

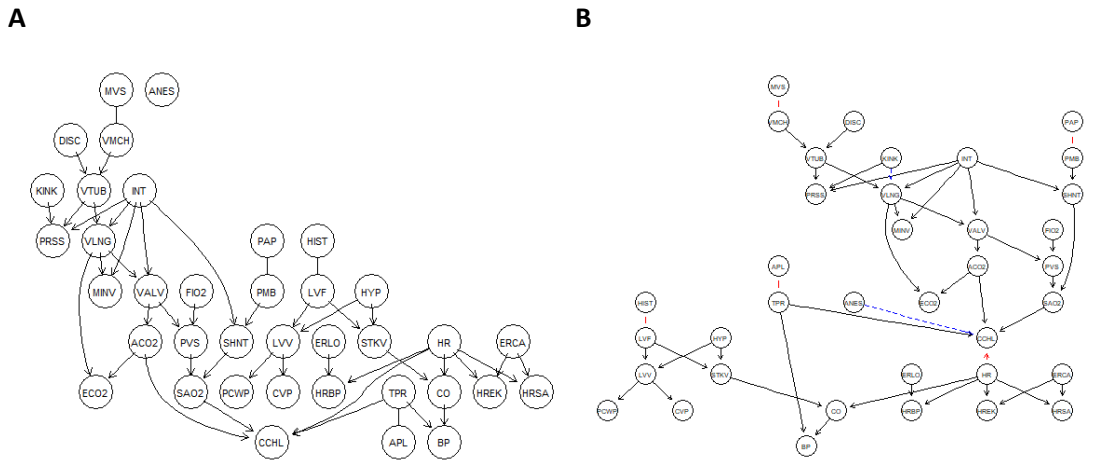


Fig. 17. Estructura ajustada mediante algoritmo PC con test X^2 de Pearson por permutaciones de Monte Carlo y α=0,2 (A) y comparación con la red causal real (B). Las flechas discontinuas azules indican arcos ausentes en la red ajustada, y las flechas rojas indican arcos espúreos.

En estas circunstancias, sería conveniente llevar a cabo un remuestro por metodología bootstrap para obtener una red consenso con los parámetros que hemos ajustado. Sin embargo, aplicar esta herramienta a algoritmos basados en restricciones con permutaciones de Monte-Carlo y números elevados de variables resulta muy exigente computacionalmente y sobrepasa los recursos y objetivos de este trabajo. En este sentido, hay que tener en cuenta que para el ajuste de redes bayesianas se deben tener en cuenta criterios de viabilidad además de la potencia estadística. La complejidad de las redes aumenta exponencialmente con el número de nodos y de niveles en cada variable, por lo que un algoritmo muy resolutivo desde el punto de vista estadístico puede ser inadecuado si requiere tiempos de ejecución o herramientas informáticas que ralentizan o imposibilitan la actividad del investigador.

5.2.3. Ajuste de parámetros

En la figura 18 se muestran las distribuciones de probabilidad marginal de cada variable ajustadas mediante estimadores de máxima verosimilitud. En este caso, la transmisión de las distribuciones de nodos padre a hijos no es tan directa como lo era en el caso de ASIA. Existen multitud de variables que, al presentar valores altos, inducen a los nodos hijos a valores bajos, además de variables que se encuentran mayoritariamente en valor nulo y cuya transmisión a los nodos hijos no es fácilmente descriptible.

En ocasiones, una mayor probabilidad de valor positivo o negativo de una variable dicotómica se transmite, para sus descendientes multinomiales, en una mayor probabilidad de valores centrales o valores extremos (altos y bajos).

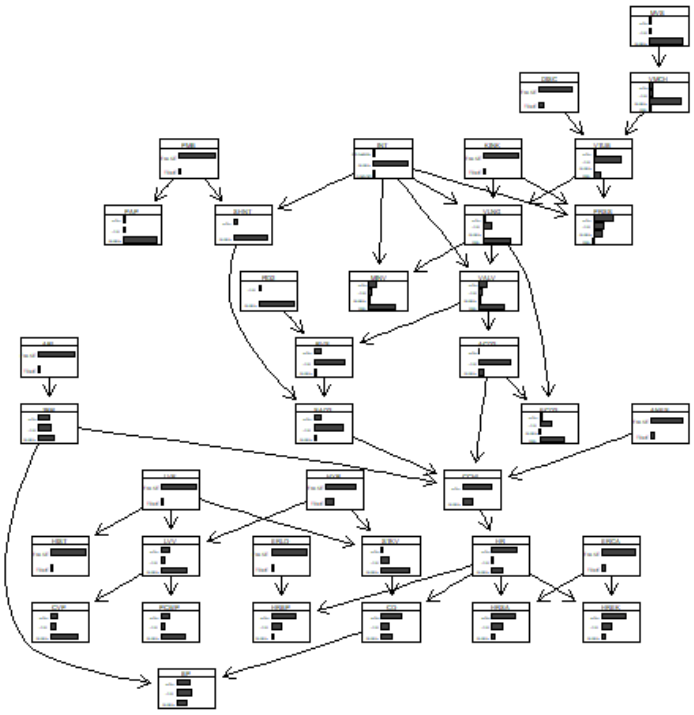


Fig. 18. Distribuciones de probabilidad marginal para las variables de la red ASIA representadas en forma de histograma.

5.3. Red SACHS

En este apartado mostramos los resultados del ajuste y evaluación de la red SACHS, así como del aprendizaje de los parámetros de la red.

5.3.1. Construcción de la red

Como expusimos anteriormente, la red de metabolitos que se predijo en el trabajo de Sachs et al presentaba algunas diferencias estructurales con la red predicha hasta el momento (figura 19).

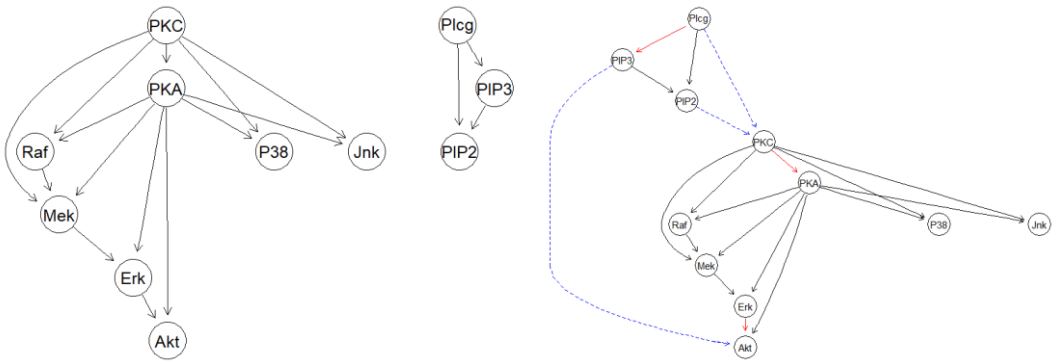


Fig. 19. Estructura ajustada en el trabajo de Sachs et al (A) y comparación con la red causal establecida en la bibliografía previa, también descrita en el trabajo de Sachs et al. (B). Las flechas discontinuas azules indican arcos ausentes en la red ajustada, y las flechas rojas indican arcos espúreos.

La red SACHS incorpora 2 arcos, elimina 3 arcos e invierte la dirección de 1 arco respecto a la red inferida en la bibliografía previa. No se producen cambios notables en la complejidad de la red; la densidad varía de Δ=0.16 a Δ=0.15, y el grado medio de los nodos pasa de 3,27 a 3,09. Sin embargo, a nivel topológico se dan

diferencias notables en la estructura de red. Tomamos la estructura inferida en Sachs et al como referencia para la verosimilitud de las redes que construyamos con nuestros algoritmos.

5.3.2. Preprocesado

Los resultados de los tests de deduplicación aplicados a los datos observacionales mostraron una única dupla de nodos con un alto nivel de correlación, compuesta por los nodos Akt y Erk ($IC_{95\%}(\rho_{Akt\text{-}Erk})$: 0.9905 – 0.9927, p-value < 2,2e-16). Ambas variables comparten un arco que sí estaba presente en la estructura inferida en el trabajo de Sachs et al, pero no así en la estructura consensuada por la bibliografía previa.

Para posteriores análisis, la discretización se llevó a cabo por el método de cuantiles, intervalos de valores e información mutua de Hartemink. Para cada uno de ellos se generaron, inicialmente, 3 intervalos.

5.3.3. Ajuste de la estructura discretizada

En la tabla 7, recogemos los resultados del ajuste de la estructura de red con distintos algoritmos en su comparativa con la red definida en el trabajo de Sachs et al. Las redes fueron ajustadas a partir del conjunto de datos gaussianos sin discretizar.

Tabla 7. Comparativa de redes ajustadas a partir de datos normales con la red inferida en el trabajo de Sachs et al. Algoritmos basados en restricciones: GS (Growth-ShrinK), IAMB (Incremental-Association Markov Blanket), PC, MMPC (Min-Max PC). Algoritmos basados en puntuaciones: HC (Hill-Climbing), Tabu (Tabu Search). Algoritmos híbridos: MMHC (Min-Max Hill-Climbing).

	Algoritmos basados en restricciones				**Algoritmos basados en puntuaciones**		**Algoritmos híbridos**
	GS	**IAMB**	**PC**	**MMPC**	**HC**	**Tabu**	**MMHC**
Verdaderos positivos	0	0	0	0	4	3	4
Falsos positivos	8	8	8	8	5	6	4
Falsos negativos	17	17	17	17	13	14	13

Se puede observar que los algoritmos basados en puntuaciones o híbridos devuelven un mejor resultado que los algoritmos basados en restricciones. Concretamente, obtenemos los mejores resultados con los algoritmos Hill-Climbing y Min-Max Hill-Climbing. Si bien con Hill-Climbing introducimos un arco espúreo más, las diferencias entre ambos son lo suficientemente pequeñas para plantear usar ambos en análisis posteriores ya que, como vimos en el caso de la red ALARM, la complejidad del proceso es un parámetro a tener en cuenta y Hill-Climbing es un algoritmo más parsimonioso.

A continuación, vemos el resultado del ajuste por los algoritmos seleccionados para nuestros datos normales y discretizados por los 3 métodos descritos anteriormente (tabla 8).

Tabla 8. Comparativa de redes ajustadas a partir de datos normales y discretizados por los algoritmos HC (Hill-Climbing) y MMHC (Min-Max Hill Climbing).

	HC				**MMHC**			
	Normal	**Intervalos**	**Cuantiles**	**Hartemink**	**Normal**	**Intervalos**	**Cuantiles**	**Hartemink**
Verdaderos positivos	4	2	6	6	4	2	6	6
Falsos positivos	5	0	2	3	4	0	2	3
Falsos negativos	13	15	11	11	13	15	11	11

Para cualquiera de los algoritmos planteados, observamos que el método de discretización por cuantiles de valores ofrece el mejor ajuste de acuerdo al trabajo de Sachs et al. Además, obtenemos los mismos valores para Hill-Climbing y Min-Max Hill-Climbing, con lo que preferentemente emplearíamos el primero por su menor complejidad.

El siguiente paso fue determinar el número de intervalos en la discretización por cuantiles que mejor se adecúa a la red de Sachs et al (tabla 9).

Tabla 9. Comparativa de redes ajustadas por el algoritmo Hill-Climbing a partir de datos discretizados por cuantiles en función del número de intervalos.

	2 intervalos	3 intervalos	5 intervalos	10 intervalos	20 intervalos
Verdaderos positivos	6	6	4	4	0
Falsos positivos	2	2	2	0	0
Falsos negativos	11	11	13	13	17

La mínima SHD se alcanza con 2 y 3 intervalos. Teniendo en cuenta la pauta de elegir siempre el modelo más parsimonioso, elegimos el modelo de discretización con 2 intervalos. Esto sería equivalente a elegir un modelo de variables dicotómicas con los niveles BAJO/ALTO. En la figura 20, se muestra el resultado obtenido al obtener la red consenso por metodología bootstrap con los parámetros que hemos definido, generando 500 muestras con reemplazo en las observaciones.

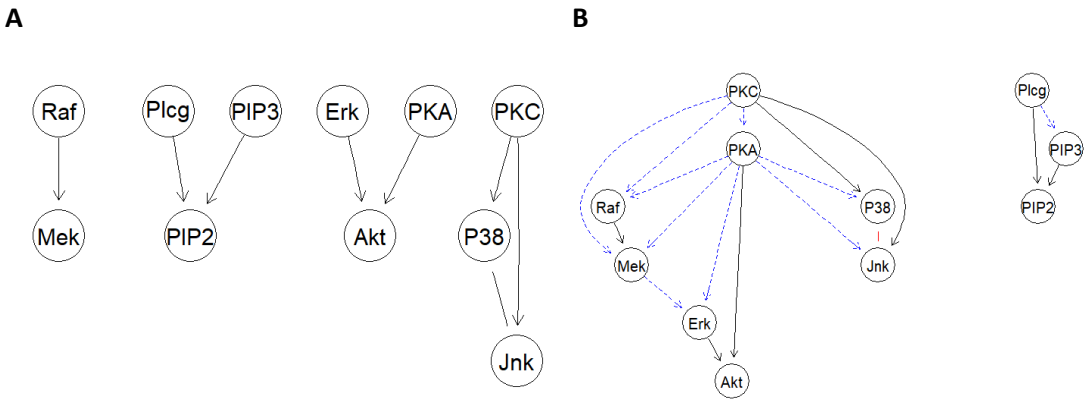


Fig. 20. Red consenso generada por metodología bootstrap con el algoritmo Hill-Climbing y los parámetros definidos previamente (A) y comparación con la red causal establecida en el trabajo de Sachs et al (B). Las flechas discontinuas azules indican arcos ausentes en la red ajustada, y las flechas rojas indican arcos espúreos.

Si bien el aprendizaje mejora al generar la red consenso, la SHD sigue siendo excesiva respecto a la red inferida en Sachs et al, con 10 falsos negativos y un falso positivo. Debido a ello, se plantearon los algoritmos de fusión de redes bayesianas gaussianas descritos anteriormente. Si bien sería posible aplicar el algoritmo de fusión por votos sobre redes construidas a partir de datos discretizados, la comparativa entre los métodos de fusión perdería coherencia al provenir las redes fusionadas de datos distintos, puesto que no existe forma de adaptar la fusión por coeficientes a variables multinomiales. Para las redes individuales que dan lugar a las redes fusionadas se utilizó el algoritmo Hill-Climbing.

5.3.4. Ajuste de la estructura gaussiana y fusión de redes

Aplicamos los algoritmos de fusión de redes a los 14 conjuntos de datos experimentales procedentes de Sachs et al.

Respecto al algoritmo *GBNFusesVote*, la red generada parece mejorar de forma sensible los resultados obtenidos a partir de los datos discretizados. De entre los valores umbral para incluir cada arco en la red fusionada, encontramos que una aparición mínima en 7 redes (50% del total). La SHD se reduce a 5, con 4 arcos no detectados y un arco espúreo (figura 21).

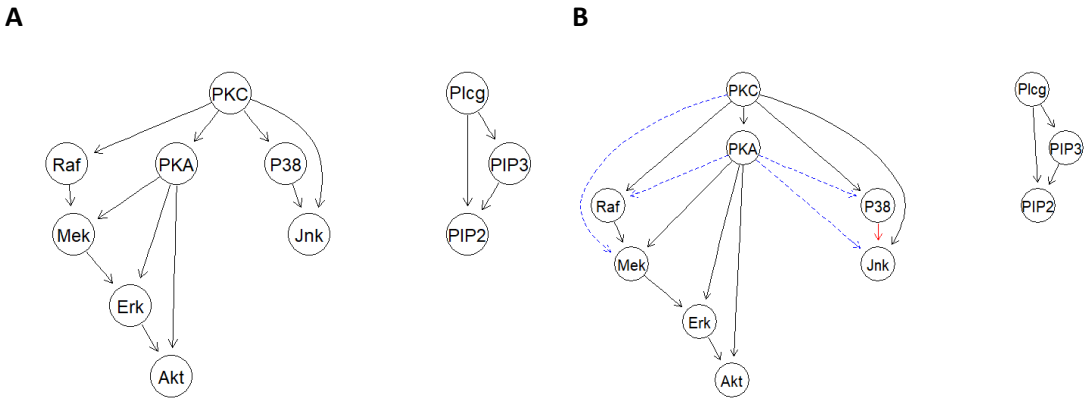


Fig. 21. Red fusionada generada por el algoritmo GBNFusesVote con redes individuales aprendidas por Hill-Climbing y los parámetros definidos previamente (A) y comparación con la red causal establecida en el trabajo de Sachs et al (B). Las flechas discontinuas azules indican arcos ausentes en la red ajustada, y las flechas rojas indican arcos espúreos.

En cuanto al algoritmo *GBNFusesInter*, encontramos un umbral óptimo para la inclusión de coeficientes no pertenecientes a la intersección de 0,7 para el valor del coeficiente normalizado. Con este valor umbral, encontramos una SHD de 3, debida enteramente a arcos espúreos (figura 22).

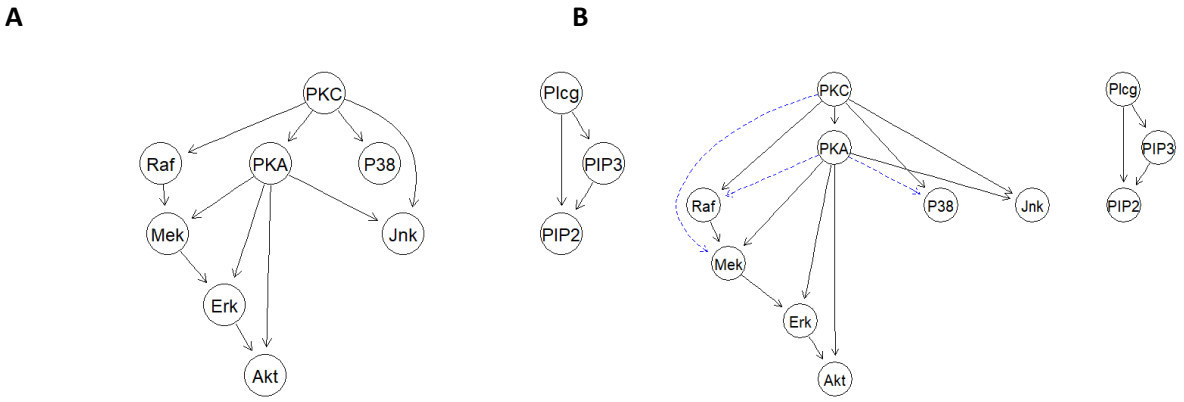


Fig. 22. Red fusionada generada por el algoritmo GBNFusesVote con redes individuales aprendidas por Hill-Climbing y los parámetros definidos previamente (A) y comparación con la red causal establecida en el trabajo de Sachs et al (B). Las flechas discontinuas azules indican arcos ausentes en la red ajustada, y las flechas rojas indican arcos espúreos.

Como hemos visto, las redes fusionadas a partir de datos observacionales y experimentales resultaron en un ajuste notablemente superior al que obtuvimos trabajando únicamente con datos observacionales. En este sentido, la fusión de redes demostró ser una herramienta efectiva para modelar las relaciones de dependencia entre variables.

5.3.5. Ajuste de parámetros

En la tabla 10 se observan los coeficientes de regresión para las variables de la red inferida en el trabajo de Sachs et al, de acuerdo al método de aprendizaje de parámetros ponderados propuesto en Córdoba et al (2015).

Tabla 10. Coeficientes de regresión ponderados para la red inferida en SACHS.

NODO	COEFICIENTES			
Raf	Intercept	PKA	PKC	
	62,1990459613	-0,0003792075	-0,1777450909	
Mek	Intercept	Raf	PKA	PKC
	-1,0902746221	0,5208447051	-0,0006520267	0,0396624572
Plcg	Intercept			
	19,49484			
PIP2	Intercept	Plcg	PIP3	
	52,3272605	0,3628681	0,7273050	
PIP3	Intercept	Plcg		
	24,340344	0,313834		
Erk	Intercept	Mek	PKA	
	-23,24874333	-0,02988581	0,08170673	
Akt	Intercept	Erk	PKA	
	1,87460049	1,36522290	0,01732143	
PKA	Intercept	PKC		
	554,3907306	0,8411526		
PKC	Intercept			
	15,01897			
P38	Intercept	PKA	PKC	
	15,14433	5,905402e-04	1,234783	
Jnk	Intercept	PKA	PKC	
	52,9553602851	-0,005306097	-0,764801188	

Los parámetros de la red tienen una interpretación diferente respecto a los dos casos anteriores, ya que en las redes Gaussianas no hablamos de distribuciones de probabilidad condicional, si no de densidades de probabilidad condicional. Este enfoque es particularmente útil en contextos en que el investigador está interesado en la cinética del proceso, como es el caso de la red de metabolitos estudiada en Sachs et al. De esta forma, predecimos cómo varía la concentración de cada fosfoproteína o fosfolípido en la red en función de la variación de las unidades de sus nodos padre.

6. CONCLUSIÓN

Pese a que las redes bayesianas han estado muy estrechamente ligadas a la práctica clínica desde su concepción, existen aún multitud de aspectos en su aplicación y desarrollo por analizar y optimizar cuando éstas se aplican al ámbito de las Ciencias de la Salud y de la Vida. En el presente trabajo, se han explorado las alternativas de que disponemos en el aprendizaje de redes bayesianas con datos provenientes de diversos campos de la medicina y la investigación. No hemos encontrado otras obras de la misma naturaleza, en que se analice el efecto que la elección de los algoritmos y la regulación de los valores de los parámetros tienen sobre el aprendizaje de redes bayesianas según el tipo de datos. Si bien es cierto que la construcción de redes bayesianas en la práctica está fuertemente condicionada por el conocimiento de los expertos en la materia y por la bibliografía previa al respecto, no se puede obviar la importancia que tiene un criterio adecuado en el uso de herramientas estadísticas para generar la estructura de la red. En nuestro caso, planteamos cómo afecta al aprendizaje el uso de algoritmos de puntuación o de restricción y los valores de parámetros como el grado de confianza o el tipo de test de independencia condicional, teniendo en cuenta el tipo de datos de que disponemos, la cantidad de variables y la densidad de la red.

En muchos aspectos técnicos, especialmente en lo que respecta a la fusión de redes bayesianas, encontramos escasas referencias en la literatura. Se trata de un recurso de las redes apenas explotado y que podría tener un impacto significativo en la investigación, ya que la integración de resultados provenientes de distintos experimentos es uno de los fundamentos de la estadística bayesiana y también es un procedimiento en auge en el mundo clínico en forma de revisiones sistemáticas y meta-análisis, que ofrecen a los investigadores la mejor evidencia disponible. Consideramos que es un área valiosa a la que dedicar futuros trabajos. Por otro lado, el *software* disponible para el aprendizaje de redes bayesianas resultó muy versátil para llevar a cabo nuestros análisis. Sin embargo, varias de las herramientas empleadas, especialmente en lo referente a la fusión de redes y de coeficientes de acuerdo al trabajo de Córdoba et al (2015) no estaban incorporadas en los paquetes estadísticos con que trabajamos, por lo que fueron programadas manualmente (ver código anexo). En este sentido, puede ser interesante añadir en futuros proyectos este tipo de prestaciones al software R, y concretamente al paquete *bnlearn*, a medida que la literatura al respecto se amplíe y se planteen nuevos métodos de fusión.

Finalmente, esperamos que nuestro trabajo sirva para aportar nuevo conocimiento sobre el uso de redes bayesianas en el ámbito biológico y sanitario, ofrecer un punto de partida a profesionales de estas disciplinas que busquen instrumentos estadísticos para detectar relaciones de dependencia entre variables en sus actividades profesionales e investigadoras, y contribuir al material existente en la programación estadística dedicada a este tipo de modelos.

7. BIBLIOGRAFÍA

Albert, J. (2009). Bayesian computation with R. *Springer*.

Arora, P., Boyne, D., Slater, J. J., Gupta, A., Brenner, D. R., & Druzdzel, M. J. (2019). Bayesian networks for risk prediction using real-world data: a tool for precision medicine. *Value in Health*, 22(4), 439-445.

Beinlich, I. A., Suermondt, H. J., Chavez, R. M., & Cooper, G. F. (1989). The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In *AIME 89* (pp. 247-256). Springer, Berlin, Heidelberg.

Bielza, C., & Larranaga, P. (2014). Bayesian networks in neuroscience: a survey. *Frontiers in computational neuroscience*, 8, 131.

Córdoba-Sánchez, I., Bielza, C., & Larrañaga, P. (2015). Towards Gaussian Bayesian Network Fusion. In European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (pp. 519-528). *Springer*.

Correa, E., & Goodacre, R. (2011). A genetic algorithm-Bayesian network approach for the analysis of metabolomics and spectroscopic data: application to the rapid identification of Bacillus spores and classification of Bacillus species. *BMC bioinformatics*, *12*(1), 1-17.

Cover, T. M. (1999). Elements of information theory. *John Wiley & Sons*.

Dıez, F. J. (1998). Aplicaciones de los modelos gráficos probabilistas en medicina. *Sistemas Expertos Probabilísticos*, 239-263.

Dimitrova, E. S., Licona, M. P. V., McGee, J., & Laubenbacher, R. (2010). Discretization of time series data. *Journal of Computational Biology*, 17(6), 853-868

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). Bayesian data analysis. *CRC press*.

Gorry, G. & G. O. Barnett, G. (1968). Experience with a model of sequential diagnosis. *Computers and Biomedical Research*, 1, 490-507.

Hansen, K.D., Gentry, J., Long, L., Gentleman, R., Falcon, S., Hahne, F., & Sarkar, D. (2020). Rgraphviz: Provides plotting capabilities for R graph objects. *R package version 2.32.0.*

Lauritzen S.L., Spiegelhalter D.J. (1988) Local computation with probabilities on graphical structures and their application to expert systems (with discussion). *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 50(2), 157–224

Lautner-Csorba, O., Gézsi, A., Semsei, Á. F., Antal, P., Erdélyi, D. J., Schermann, G., ... & Szalai, C. (2012). Candidate gene association study in pediatric acute lymphoblastic leukemia evaluated by Bayesian network based Bayesian multilevel analysis of relevance. *BMC medical genomics*, 5(1), 1-15.

Lesaffre, E., & Lawson, A. B. (2012). Bayesian biostatistics. *John Wiley & Sons*.

Mani, S., Valtorta, M., & McDermott, S. (2005). Building Bayesian network models in medicine: The MENTOR experience. *Applied Intelligence*, 22(2), 93-108.

McCann, R. K., Marcot, B. G., & Ellis, R. (2006). Bayesian belief networks: applications in ecology and natural resource management. Canadian Journal of Forest Research, 36(12), 3053-3062.

Nagarajan, R., Scutari, M., & Lèbre, S. (2013). Bayesian networks in r. *Springer*, 122, 125-127.

Sachs K, Perez O, Pe'er D, Lauffenburger DA, Nolan GP (2005) Causal protein-signaling networks derived from multiparameter single-cell data. *Science* 308(5721), 523–529

Scanagatta, M., Salmerón, A., Stella, P. (2019). A survey on Bayesian network structure learning from data. *Progress in Artificial Intelligence*, 8, 425-489.

Schwartz, S., Baron, J. Y Clarke, J. (1988). A causal Bayesian model for the diagnosis of appendicitis. *Uncertainty in Artificial Intelligence*, 2, 423-434.

Scutari, M. (2010). Learning Bayesian Networks with the bnlearn R Package. *Journal of Statistical Software*, 35(3), 1-22. URL http://www.jstatsoft.org/v35/i03/. [Fecha de consulta: 18/05/2021]

Scutari, M., Scutari, M. M., & MMPC, H. P. (2020). Package 'bnlearn'. Bayesian Network Structure Leaning, parameter learning and inference and inference, R package

Silva, L. C., & Benavides, A. (2001). El enfoque bayesiano: otra manera de inferir. *Gaceta Sanitaria*, 15(4), 341-346.

Sucar, L. E., & Tonantzintla, M. (2006). Redes bayesianas. *Aprendizaje Automático: conceptos básicos y avanzados*, 77, 100.

Spiegelhalter, D. & R. P. Knill-Jones, R. (1984). Statistical and knowledge-based approaches to clinical decision support systems, with an application to gastroenterology. *Journal of the Royal Statistical Society*, 147, 35-77.

Warner, H., Toronto, A., Veasy, L. & Stephenson, R (1961). A mathematical approach to medical diagnosis: Application to congenital heart disease. *Journal of the American Medical Association*, 177, 177-183.

8. ANEXOS

Tabla A1. Distribuciones condicionales de los nodos en la red ASIA.

Nodo	**Tablas de contingencia**		
A	*SÍ*	*NO*	
	0.0084	0.9916	
S	*SÍ*	*NO*	
	0.5030	0.4970	
T	*SÍ*	*NO*	
	0.0088	0.9912	
L		*SÍ*	*NO*
	S=SÍ	0.1177	0.8823
	S=NO	0.0137	0.9863
B		*SÍ*	*NO*
	S=SÍ	0.7177	0.2823
	S=NO	0.2994	0.7006
X		*SÍ*	*NO*
	E=SÍ	0.9946	0.0054
	E=NO	0.0434	0.9566
D	*E= SÍ*		
		SÍ	*NO*
	B=SÍ	0.8541	0.1459
	B=NO	0.7226	0.2774
	E= NO		
		SÍ	*NO*
	B=SÍ	0.7863	0.2137
	B=NO	0.0998	0.7862
E	*L= SÍ*		
		SÍ	*NO*
	T=SÍ	1	0
	T=NO	0	1
	L= NO		
		SÍ	*NO*
	T=SÍ	1	0
	T=NO	1	0

Tabla A2. Distribuciones condicionales de los nodos en la red ASIA.

Nodo	**Tablas de contingencia**			
APL	*SÍ*	*NO*		
	0.0101	0.9899		
ANES	*SÍ*	*NO*		
	0.1047	0.8953		
DISC	*SÍ*	*NO*		
	0.0987	0.9013		
ERCA	*SÍ*	*NO*		
	0.0994	0.9006		
ERLO	*SÍ*	*NO*		
	0.0501	0.9499		
HYP	*SÍ*	*NO*		
	0.2023	0.7977		
KINK	*SÍ*	*NO*		
	0.0364	0.9636		
LVF	*SÍ*	*NO*		
	0.0495	0.9505		
PMB	*SÍ*	*NO*		
	0.0103	0.9897		
FIO2	*ALTO*	*NORMAL*	*BAJO*	
	0.0000	0.9495	0.0505	
INT	*ESOFAGEAL*	*NORMAL*	*UNILATERAL*	
	0.0303	0.9198	0.0499	
MVS	*ALTO*	*NORMAL*	*BAJO*	
	0.0517	0.8963	0.0520	
HIST		*SÍ*	*NO*	
	LVF=SÍ	0.8930	0.1070	
	LVF=NO	0.0109	0.9891	
ACO2		*ALTO*	*NORMAL*	*BAJO*
	VALV=ALTO	0.0088	0.0851	0.9061
	VALV=NORMAL	0.0012	0.9248	0.0740
	VALV=BAJO	0.0150	0.1000	0.8850
	VALV=CERO	0.0100	0.0977	0.8923
CVP		*ALTO*	*NORMAL*	*BAJO*
	LVV=ALTO	0.7071	0.2806	0.0123
	LVV=NORMAL	0.2806	0.9532	0.0448
	LVV=BAJO	0.0067	0.0448	0.9484
HR		*ALTO*	*NORMAL*	*BAJO*
	CCHL=ALTO	0.8965	0.0937	0.0098
	CCHL=NORMAL	0.0439	0.9073	0.0488
PAP		*ALTO*	*NORMAL*	*BAJO*
	PMB=SÍ	0.7951	0.1856	0.0195
	PMB=NO	0.0499	0.8992	0.0509
PCWP		*ALTO*	*NORMAL*	*BAJO*
	LVV=ALTO	0.9493	0.4098	0.0097
	LVV=NORMAL	0.0091	0.9512	0.0397
	LVV=BAJO	0.0113	0.0346	0.9541

TPR		*ALTO*	*NORMAL*	*BAJO*
	APL=SÍ	0.0099	0.0050	0.9851
	APL=NO	0.3013	0.3989	0.2998

VMCH		*ALTO*	*NORMAL*	*BAJO*	*CERO*
	MVS=ALTO	0.9372	0.0116	0.0048	0.0464
	MVS=NORMAL	0.0103	0.9298	0.0103	0.0496
	MVS=BAJO	0.0096	0.0048	0.9462	0.0394

BP	*TPR=ALTO*			
		ALTO	*NORMAL*	*BAJO*
	CO=ALTO	0.8978	0.0898	0.0124
	CO=NORMAL	0.5422	0.4087	0.0491
	CO=BAJO	0.0925	0.6197	0.2878
	TPR=NORMAL			
		ALTO	*NORMAL*	*BAJO*
	CO=ALTO	0.7446	0.2070	0.0484
	CO=NORMAL	0.0520	0.8465	0.1015
	CO=BAJO	0.0172	0.0086	0.9742
	TPR=BAJO			
		ALTO	*NORMAL*	*BAJO*
	CO=ALTO	0.0108	0.0910	0.8982
	CO=NORMAL	0.0069	0.0069	0.9862
	CO=BAJO	0.0172	0.0086	0.9742

CCHL	*ANES=SÍ, SAO2=ALTO, TPR=ALTO*		
		ALTO	*NORMAL*
	ACO2=ALTO	1.0000	0.0000
	ACO2=NORMAL	0.0000	1.0000
	ACO2=BAJO	0.0698	0.9302
	ANES=NO, SAO2=ALTO, TPR=ALTO		
		ALTO	*NORMAL*
	ACO2=ALTO	0.6000	0.4000
	ACO2=NORMAL	0.0119	0.9881
	ACO2=BAJO	0.0516	0.9484
	ANES=SÍ, SAO2=NORMAL, TPR=ALTO		
		ALTO	*NORMAL*
	ACO2=NORMAL	1.0000	0.0000
	ACO2=BAJO	0.1739	0.8261
	ANES=SÍ, SAO2=ALTO, TPR=NORMAL		
		ALTO	*NORMAL*
	ACO2=ALTO	1.0000	0.0000
	ACO2=NORMAL	1.0000	0.0000
	ACO2=BAJO	0.9766	0.0234
	ANES=NO, SAO2=NORMAL, TPR=ALTO		
		ALTO	*NORMAL*
	ACO2=NORMAL	0.0000	1.0000
	ACO2=BAJO	0.0270	0.9730
	ANES=NO, SAO2=ALTO, TPR=NORMAL		
		ALTO	*NORMAL*
	ACO2=ALTO	1.0000	0.0000
	ACO2=NORMAL	0.9735	0.0265
	ACO2=BAJO	0.9482	0.0518

CCHL	*ANES=SÍ, SAO2=BAJO, TPR=ALTO*		
		ALTO	*NORMAL*
	ACO2=ALTO	1.0000	0.0000
	ACO2=NORMAL	0.2901	0.7099
	ACO2=BAJO	0.3500	0.6500
	ANES=SÍ, SAO2=ALTO, TPR=BAJO		
		ALTO	*NORMAL*
	ACO2=ALTO	1.0000	0.0000
	ACO2=NORMAL	1.0000	0.0000
	ACO2=BAJO	1.0000	0.0000
	ANES=NO, SAO2=BAJO, TPR=ALTO		
		ALTO	*NORMAL*
	ACO2=ALTO	0.9583	0.0417
	ACO2=NORMAL	0.2895	0.7105
	ACO2=BAJO	0.3024	0.6976
	ANES=NO, SAO2=ALTO, TPR=BAJO		
		ALTO	*NORMAL*
	ACO2=ALTO	1.0000	0.0000
	ACO2=NORMAL	0.9740	0.0260
	ACO2=BAJO	0.9595	0.0405
	ANES=SÍ, SAO2=NORMAL, TPR=NORMAL		
		ALTO	*NORMAL*
	ACO2=ALTO	1.0000	0.000
	ACO2=NORMAL	0.8571	0.1429
	ACO2=BAJO	0.9476	0.0524
	ANES=NO, SAO2=NORMAL, TPR=NORMAL		
		ALTO	*NORMAL*
	ACO2=ALTO	1.0000	0.0000
	ACO2=NORMAL	0.9817	0.0183
	ACO2=BAJO	0.9881	0.0119
	ANES=SÍ, SAO2=NORMAL, TPR=BAJO		
		ALTO	*NORMAL*
	ACO2=BAJO	0.8824	0.1176
	ANES=NO, SAO2=NORMAL, TPR=BAJO		
		ALTO	*NORMAL*
	ACO2=ALTO	1.0000	0.0000
	ACO2=NORMAL	1.0000	0.0000
	ACO2=BAJO	0.9884	0.0116
	ANES=SÍ, SAO2=BAJO, TPR=NORMAL		
		ALTO	*NORMAL*
	ACO2=ALTO	1.0000	0.0000
	ACO2=NORMAL	0.9817	0.0183
	ACO2=BAJO	0.9881	0.0119
	ANES=NO, SAO2=BAJO, TPR=NORMAL		
		ALTO	*NORMAL*
	ACO2=ALTO	1.0000	0.0000
	ACO2=NORMAL	0.9571	0.0429
	ACO2=BAJO	0.9441	0.0559

	ANES=SÍ, SAO2=BAJO, TPR=BAJO			
		ALTO	*NORMAL*	
	ACO2=ALTO	1.0000	0.0000	
	ACO2=NORMAL	1.0000	0.0000	
	ACO2=BAJO	0.9883	0.0117	
	ANES=NO, SAO2=BAJO, TPR=BAJO			
		ALTO	*NORMAL*	
	ACO2=ALTO	1.0000	0.0000	
	ACO2=NORMAL	0.9904	0.0096	
	ACO2=BAJO	0.9882	0.0118	
CO	*STVK=ALTO*			
		ALTO	*NORMAL*	*BAJO*
	HR=ALTO	0.9778	0.0102	0.0120
	HR=NORMAL	0.6878	0.3038	0.0084
	HR=BAJO	0.0000	0.7500	0.2500
	STVK=NORMAL			
		ALTO	*NORMAL*	*BAJO*
	HR=ALTO	0.9481	0.0424	0.0095
	HR=NORMAL	0.0106	0.9520	0.0375
	HR=BAJO	0.0000	0.0000	1.0000
	STVK=BAJO			
		ALTO	*NORMAL*	*BAJO*
	HR=ALTO	0.0078	0.7987	0.1935
	HR=NORMAL	0.0084	0.0338	0.9578
	HR=BAJO	0.0000	0.0000	1.0000

Las redes bayesianas cuentan con múltiples aplicaciones en el ámbito de Ciencias de la Salud y de la Vida. Aquí, mostraremos ejemplos de estructuras de datos biológicos incorporadas en el paquete bnlearn, así como las funcionalidades de bnlearn para ajustar las redes y los parámetros.

ASIA

Lauritzen, S. L., & Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society: Series B (Methodological), 50(2), 157-194.

El conjunto de datos *ASIA*, correspondiente al trabajo de Lauritzen y Spiegelhalter (1998), recoge el diagnóstico de distintos pacientes aquejados de dificultades respiratorias (disnea).

La afección puede deberse a tuberculosis, cáncer de pulmón o bronquitis, ninguna de ellas o cualquier combinación entre ellas. La estancia en Asia aumenta las probabilidades de contraer tuberculosis, mientras que el tabaquismo es un factor de riesgo para el cáncer de pulmón y la bronquitis.

Los resultados puntuales de una radiografía pulmonar no logran discriminar entre cáncer de pulmón y tuberculosis. Tampoco lo hace la presencia o ausencia de disnea.

Estamos interesados en conocer las probabilidades de que se de cada una de las enfermedades. Si la tuberculosis fuese descartada por otro test, ¿cómo cambiaría la confianza en el diagnóstico de cáncer de pulmón? Por otro lado, ¿qué contribuye más al conocimiento sobre la ausencia o presencia de cáncer, el historial de tabaquismo o la radiografía, teniendo en cuenta que la disnea podría venir por la bronquitis asociada al tabaco? Por último, cuando toda la información está recogida, ¿podemos identificar cuál fue más influyente en el diagnóstico?

CONSTRUCCIÓN DE LA RED

En primer lugar, descargamos el conjunto de datos del paquete *bnlearn*.

```
library(bnlearn)

data(asia)
```

A continuación, observamos las características de los datos:

```
str(asia)
```

```
## 'data.frame':    5000 obs. of  8 variables:
##  $ A: Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ S: Factor w/ 2 levels "no","yes": 2 2 1 1 1 2 1 2 2 2 ...
##  $ T: Factor w/ 2 levels "no","yes": 1 1 2 1 1 1 1 1 1 1 ...
##  $ L: Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ B: Factor w/ 2 levels "no","yes": 2 1 1 2 1 1 1 2 2 2 ...
##  $ E: Factor w/ 2 levels "no","yes": 1 1 2 1 1 1 1 1 1 1 ...
##  $ X: Factor w/ 2 levels "no","yes": 1 1 2 1 1 1 1 1 1 1 ...
##  $ D: Factor w/ 2 levels "no","yes": 2 1 2 2 2 2 1 2 2 2 ...
```

El conjunto de datos contiene las variables:

+D: disnea, un factor de dos niveles con valores *sí* o *no*. +T: tuberculosis, un factor de dos niveles con valores *sí* o *no*. +L: cáncer de pulmón, un factor de dos niveles con valores *sí* o *no*. +B: bronquitis, un factor de dos niveles con valores *sí* o *no*. +A: visita a Asia, un factor de dos niveles con valores *sí* o *no*. +S: tabaquismo, un factor de dos niveles con valores *sí* o *no*. +X: rayos X, un factor de dos niveles con valores *sí* o *no*. +E: tuberculosis vs cáncer de pulmón/bronquitis, un factor de dos niveles con valores *sí* o *no*.

Se trata de una red de variables binomiales, una simplificación de la distribución multinomial en que cada variable presenta dos niveles (SÍ/NO).

Respecto al tamaño de la muestra:

```
dim(asia)
```

```
## [1] 5000    8
```

El conjunto de datos cuenta con 5000 observaciones, con la siguiente distribución de niveles:

```
summary(asia)
```

```
##    A          S          T          L          B          E          X
##  no :4958   no :2485   no :4956   no :4670   no :2451   no :4630   no :4431
##  yes:  42   yes:2515   yes:  44   yes: 330   yes:2549   yes: 370   yes: 569
##    D
##  no :2650
##  yes:2350
```

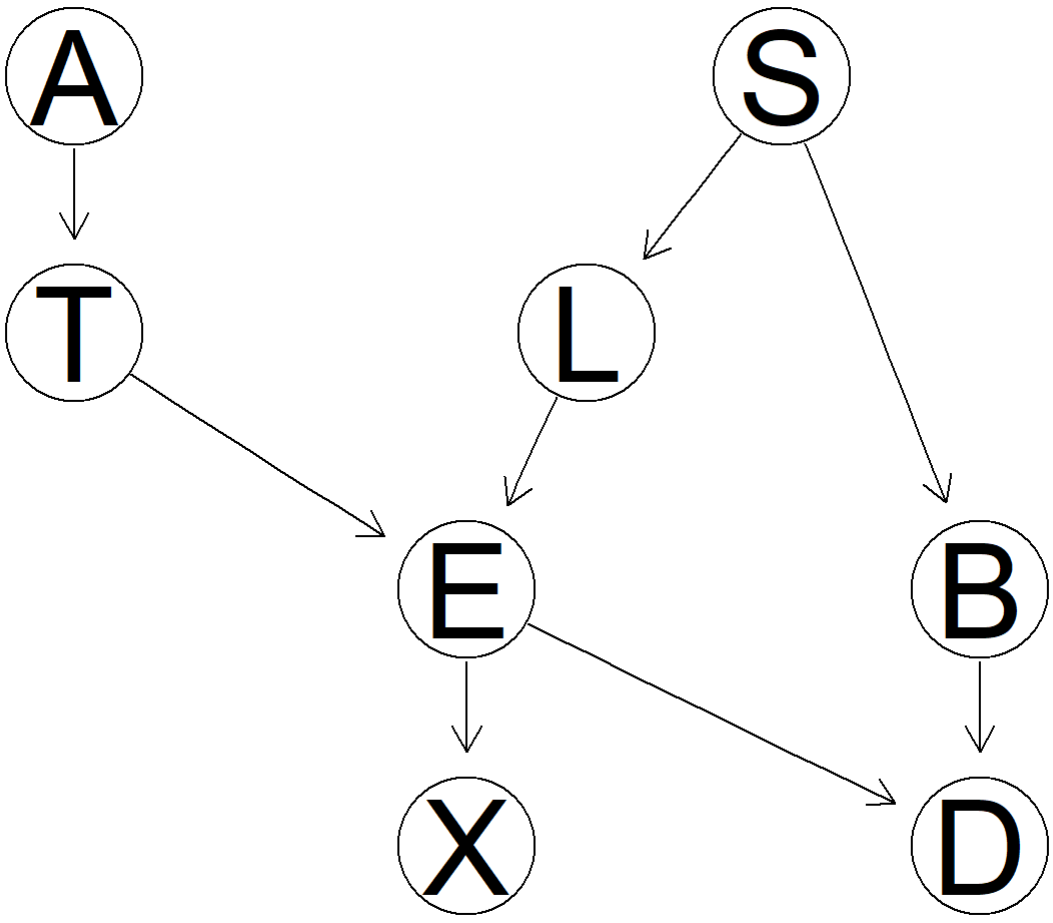
Como vemos, a excepción del hábito tabáquico y la disnea, el resto de campos se hallan muy desequilibrados: la ausencia es mucho mas común que la presencia del factor.

Podemos construir nuestra red bayesiana en forma de objeto *bn* implementado en el paquete *bnlearn*. Para ello, nos apoyamos en la estructura definida por Lauritzen y Spiegelhalter:

```
asia.bn <- empty.graph(names(asia))

arcs(asia.bn, check.cycles=FALSE) <- matrix(c("A", "T", "T", "E", "E", "X", "E", "D", "S", "L",
"L", "E","S", "B", "B", "D"), ncol = 2, byrow = TRUE, dimnames = list(c(), c("from", "to")))
```

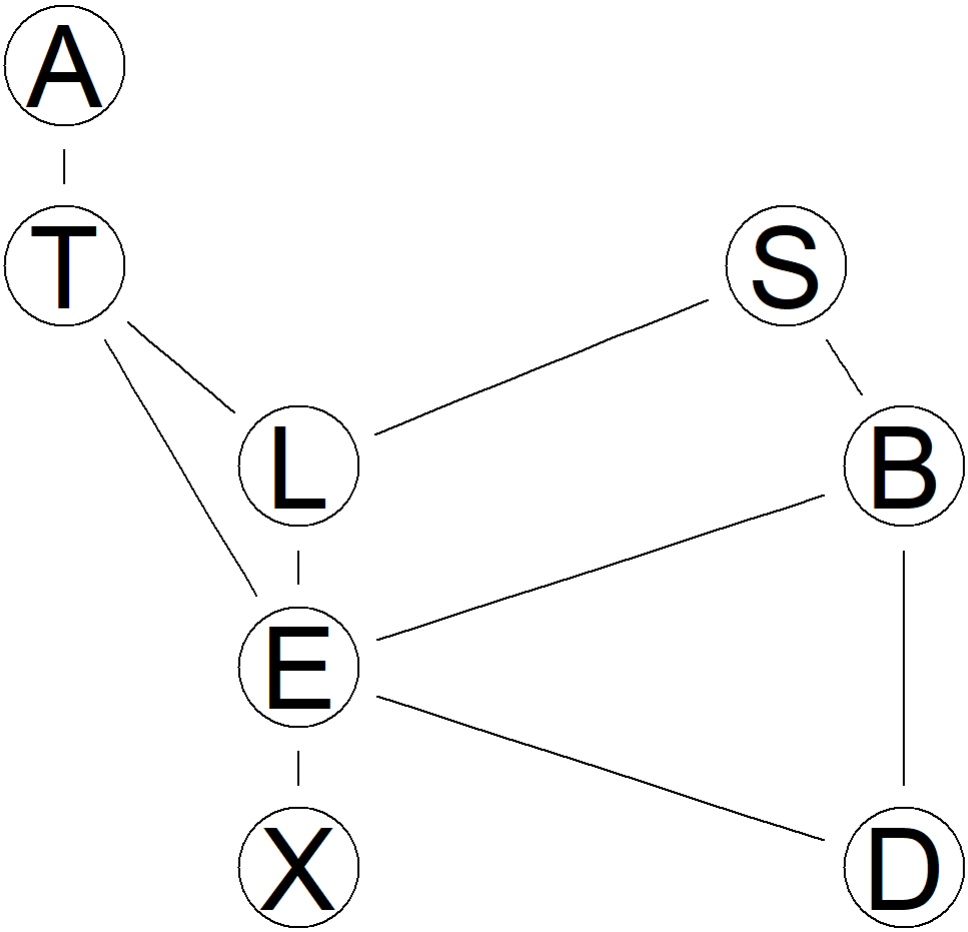
Nuestro grafo tendría la forma:

```
graphviz.plot(asia.bn)
```



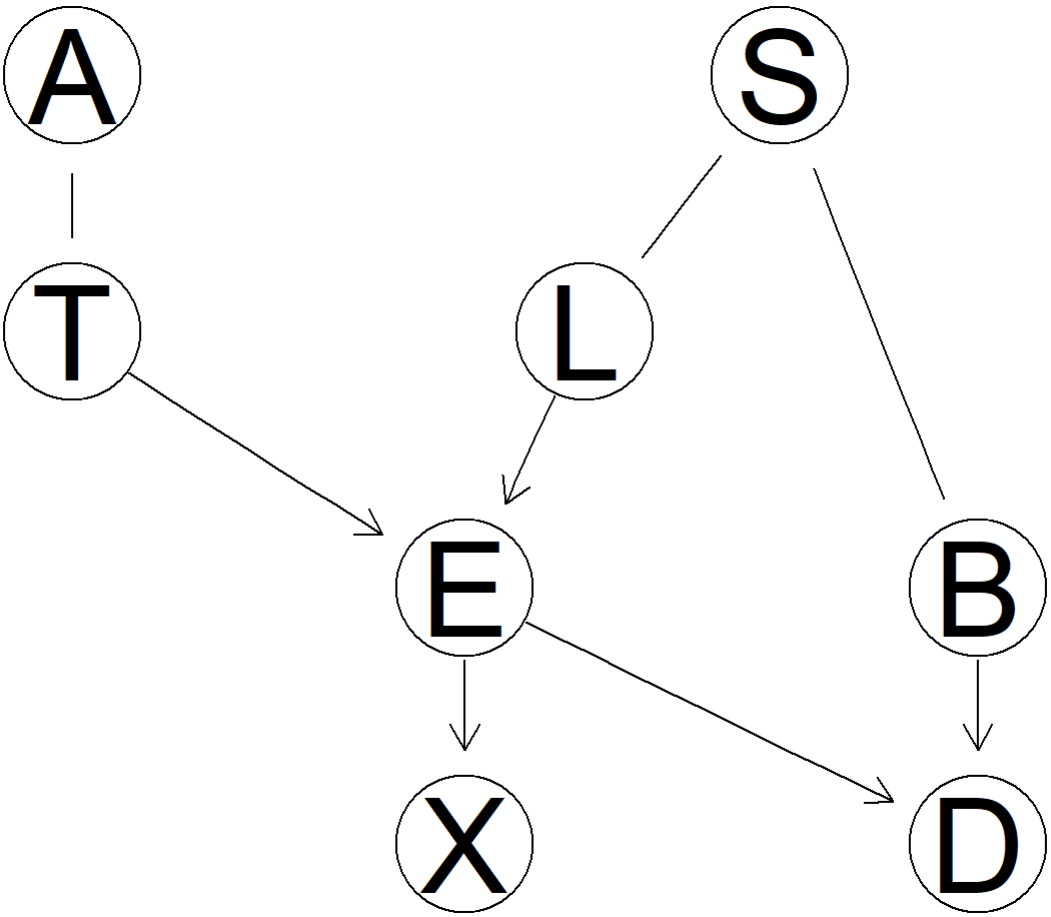
De lo que se deriva el grafo moral:

```
graphviz.plot(moral(asia.bn))
```



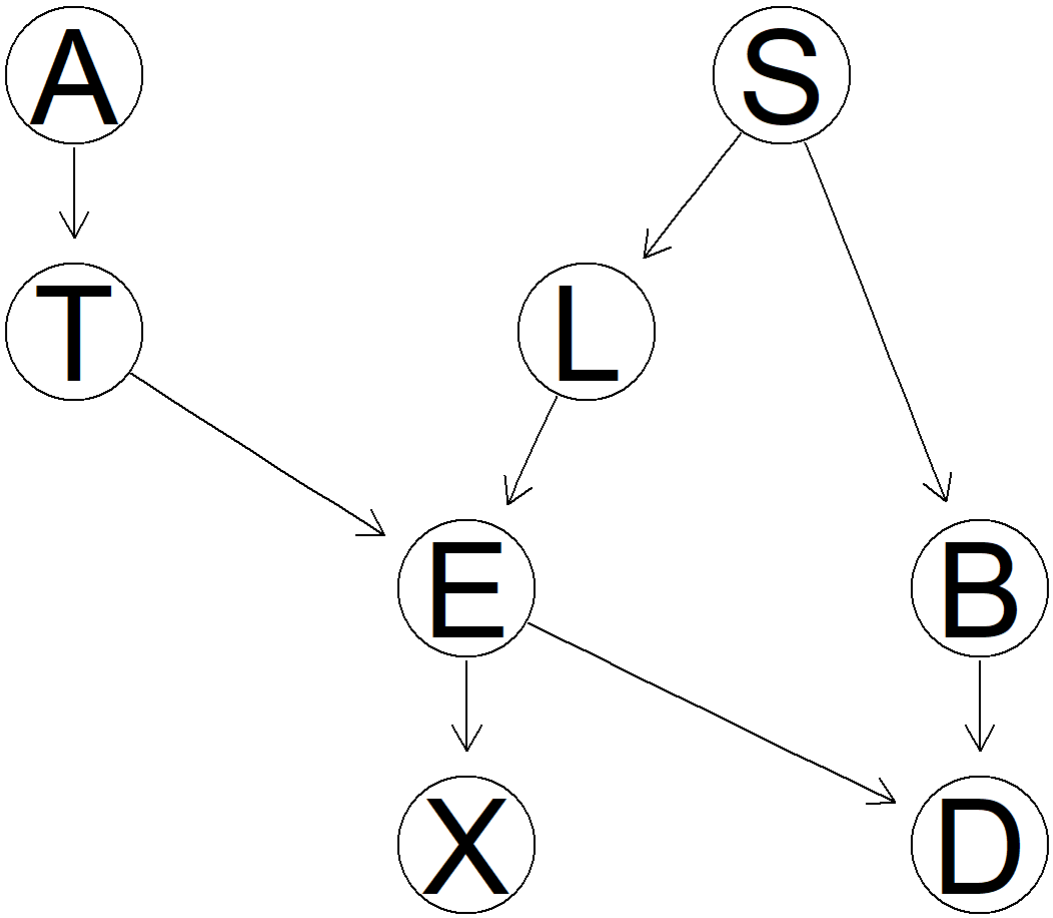
Y el CPDAG:

```
graphviz.plot(cpdag(asia.bn))
```

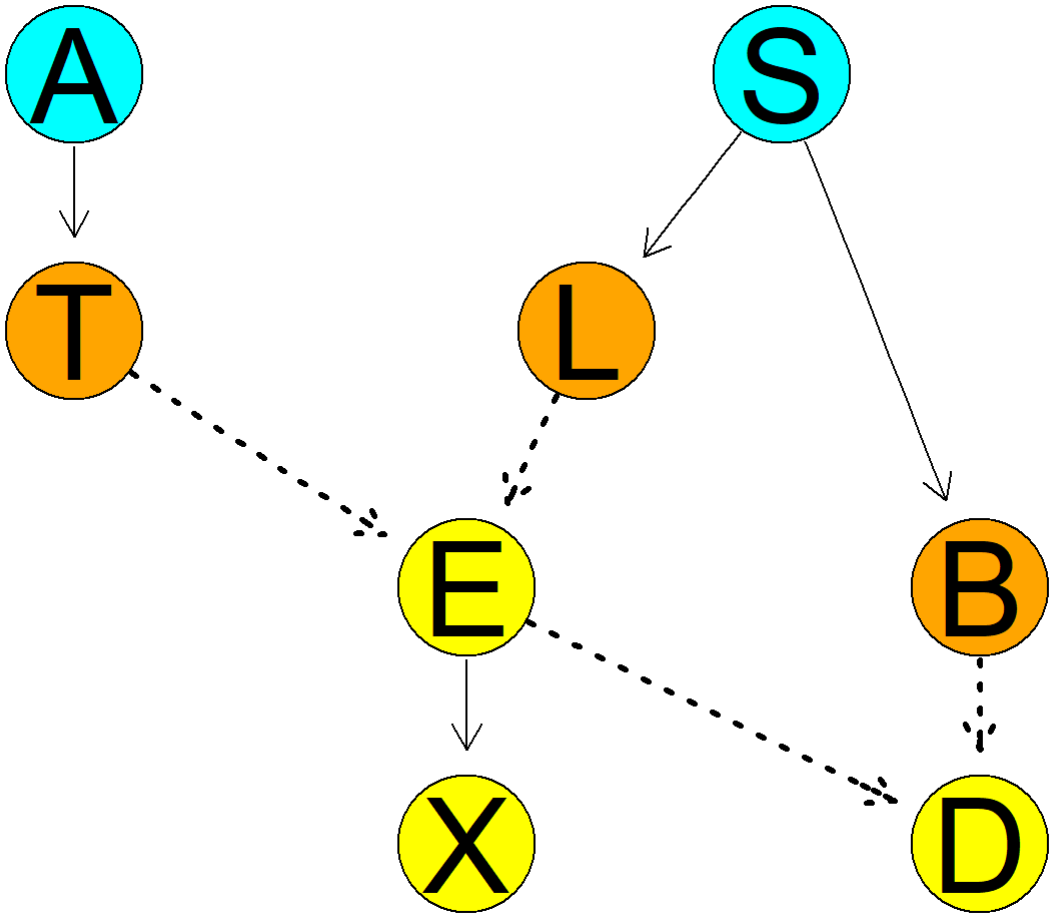


Podemos plasmar gráficamente la red discriminando entre nodos que se corresponden con principios causales (viaje a Asia, tabaquismo), posibles dolencias (cáncer de pulmón, bronquitis, tuberculosis) y pruebas diagnósticas derivadas de los síntomas de la enfermedad (historial de tabaquismo, radiografía, tuberculosis o cáncer de pulmón/bronquitis). También podemos resaltar las estructuras en V. Para ello, nos valemos de las funcionalidades de Rgraphviz.

```
library(Rgraphviz)

asia.graph <- graphviz.plot(asia.bn)
```




```
asia.attr <- nodeRenderInfo(asia.graph)
asia.attr.edge <- edgeRenderInfo(asia.graph)

asia.attr$fill[c("A","S")] <- "cyan"
asia.attr$fill[c("T","L","B")] <- "orange"
asia.attr$fill[c("E","X","D")] <- "yellow"

asia.attr.edge$lwd[c("T~E","L~E","E~D","B~D")] <- 3
asia.attr.edge$lty[c("T~E","L~E","E~D","B~D")] <- "dotted"

nodeRenderInfo(asia.graph) <- asia.attr
edgeRenderInfo(asia.graph) <- asia.attr.edge

renderGraph(asia.graph)
```



La densidad del grafo es:

```
asia.arcs <- narcs(asia.bn)
asia.nodes <- nnodes(asia.bn)

asia.density <- asia.arcs/(asia.nodes*(asia.nodes-1))
asia.density
```

```
## [1] 0.1428571
```

AJUSTE DE ESTRUCTURA

Una de las características más útiles de la estadística bayesiana es que podemos construir nuestros modelos con base en el conocimiento previo sobre el campo, e ir actualizándolos a medida que recogemos nueva evidencia.

En nuestro caso, *ASIA* es una red basada en procedimientos clínicos y en la que conocemos a priori principios causales, dolencias y pruebas diagnósticas, así como la relación entre ellas, por lo que podemos definir de antemano la estructura de la red.

Respecto a su puntuación, repasamos algunos de los algoritmos vistos hasta ahora, aplicables a conjuntos de datos con valores discretos:

```
# criterio de puntuación BDe
score(asia.bn, data=asia, type="bde")
```

```
## [1] -11095.82
```

```
# criterio de puntuación Log-Likelihood
score(asia.bn, data=asia, type="loglik")
```

```
## [1] -11033.09
```

```
# criterio de puntuación AIC
score(asia.bn, data=asia, type="aic")
```

```
## [1] -11051.09
```

```
# criterio de puntuación BIC
score(asia.bn, data=asia, type="bic")
```

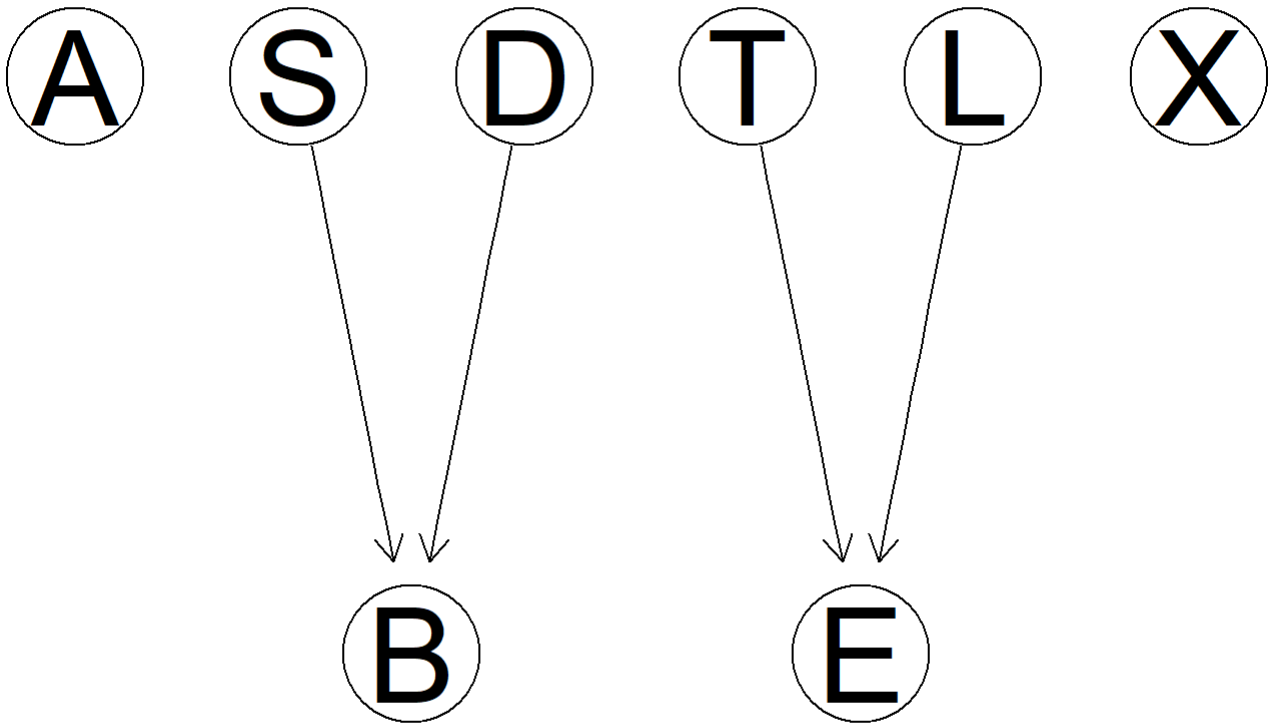
```
## [1] -11109.74
```

Vemos que los 4 criterios nos devuelven valores coherentes para la puntuación de la red bayesiana *ASIA*.

Sin embargo, podría darse el caso de que no contáramos con información previa sobre cómo están interrelacionadas las variables. En ese caso, podríamos preguntarnos qué método resulta más adecuado para predecir la estructura de dependencia.

A continuación, veremos distintos algoritmos para el ajuste de la estructura, evaluando su aptitud en función de la semejanza de la red con la original:

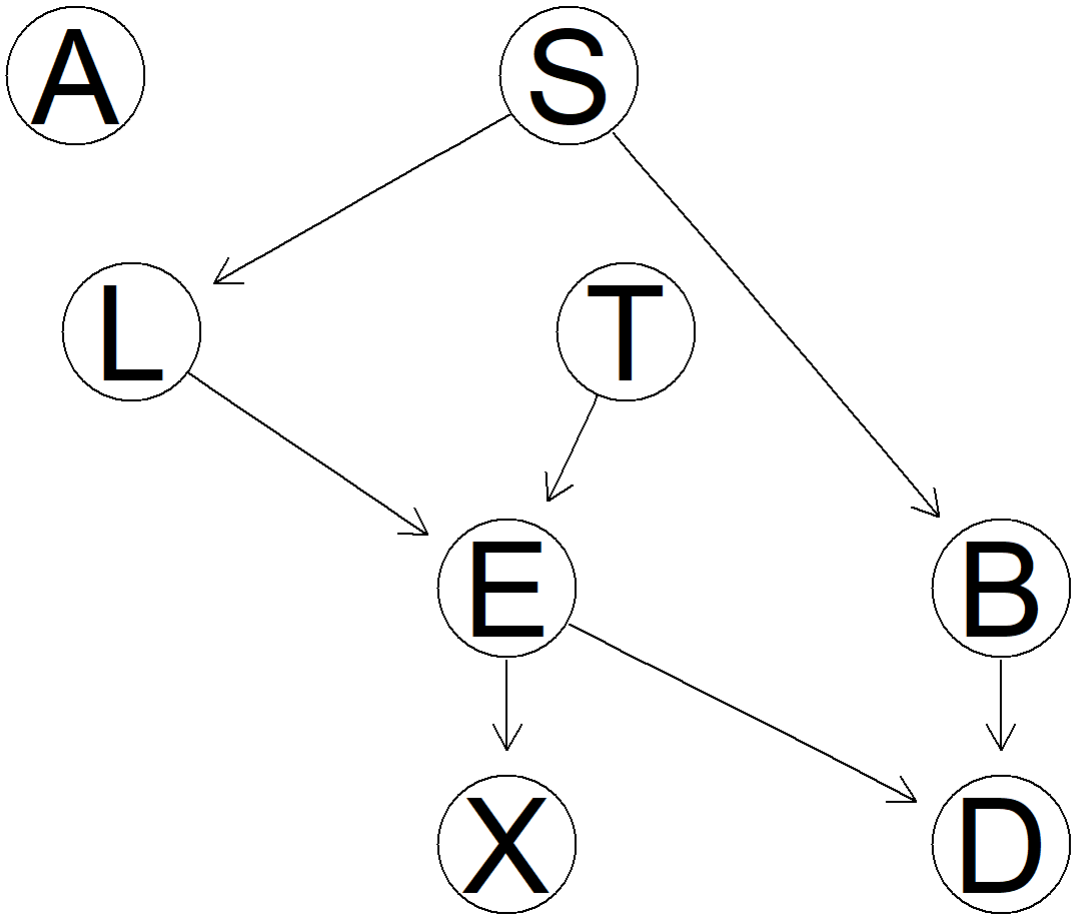
```
# Growth-Shrink
asia.gs <- gs(asia)

graphviz.plot(asia.gs)
```




```
score(asia.gs, data=asia, type="bde")
```

```
## [1] -12517.55
```

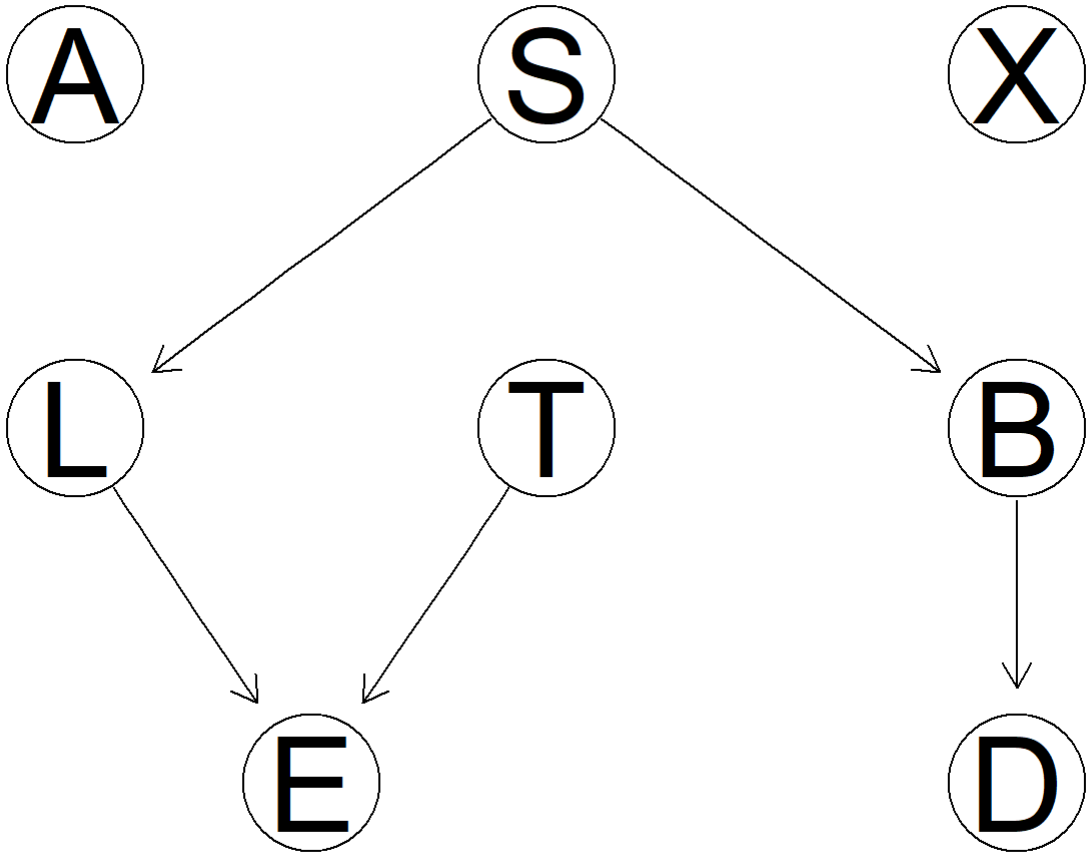
```
# Hill-Climbing
asia.hc <- hc(asia)

graphviz.plot(asia.hc)
```




```
score(asia.hc, data=asia, type="bde")
```

```
## [1] -11095.79
```

```
#Min-Max Hill-Climbing
asia.mmhc <- mmhc(asia)

graphviz.plot(asia.mmhc)
```

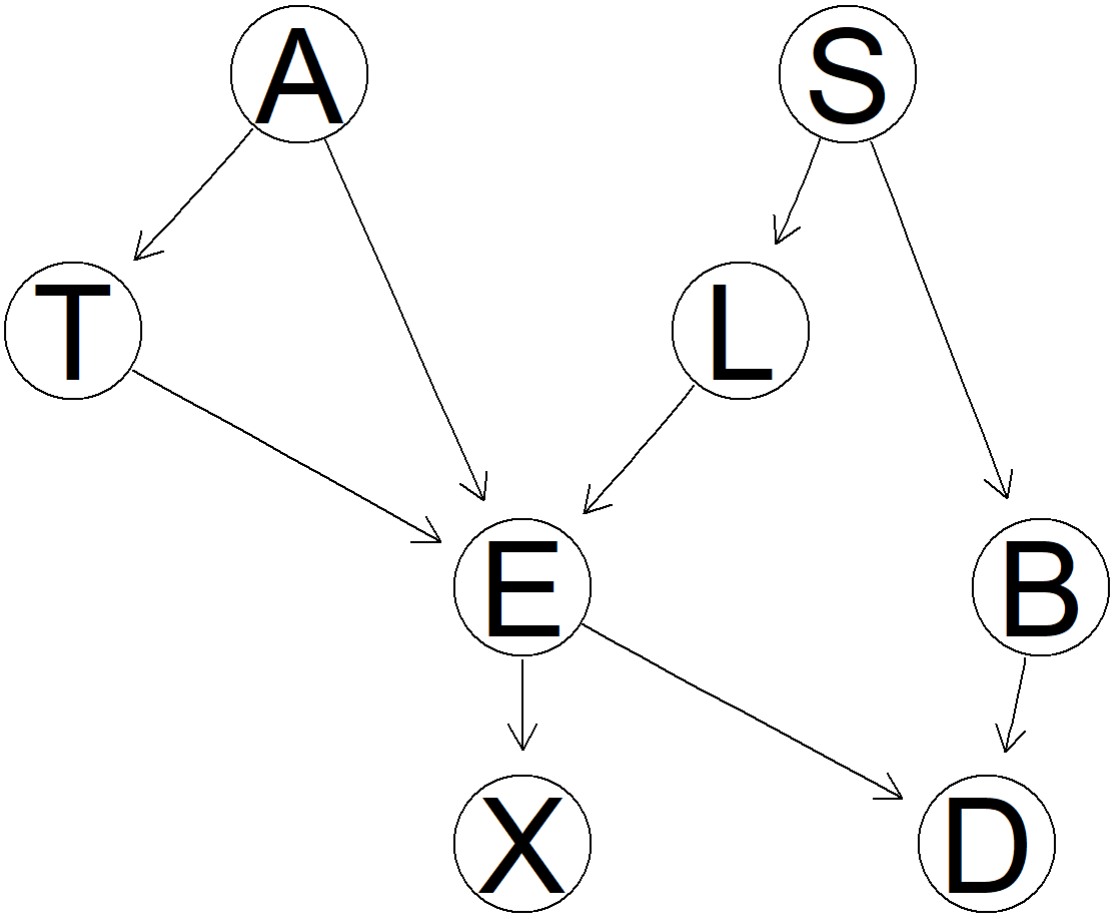



```
score(asia.mmhc, data=asia, type="bde")
```

```
## [1] -12156.47
```

Por lo que vemos, el algoritmo Hill-Climbing es el más fiel a la hora de reflejar las estructuras de dependencia que se dan en la realidad. Sin embargo, continúa siendo muy impreciso. Una forma de aumentar la fiabilidad de la red es obtener una red consenso empleando la metodología bootstrap, de modo que sólo los arcos que cuentan con alta verosimilitud entran en el modelo. En nuestro caso, empleando este método tras 500 iteraciones:

```
asia.boot <- boot.strength(data=asia, R=500,
                    algorithm="hc",
                    algorithm.args=list(score="bde",
                                        iss=10))

asia.avg <- averaged.network(asia.boot, threshold=0.95)

graphviz.plot(asia.avg)
```



La red es casi idéntica a la estructura del trabajo de Lauritzen y Spiegelhalter, a excepción de un arco espúreo que va desde la estancia en Asia (A) a la dicotomía bronquitis vs cáncer de pulmón/tuberculosis (E). Si observamos las características de este enlace:

```
asia.boot[asia.boot$strength > 0.95,]
```

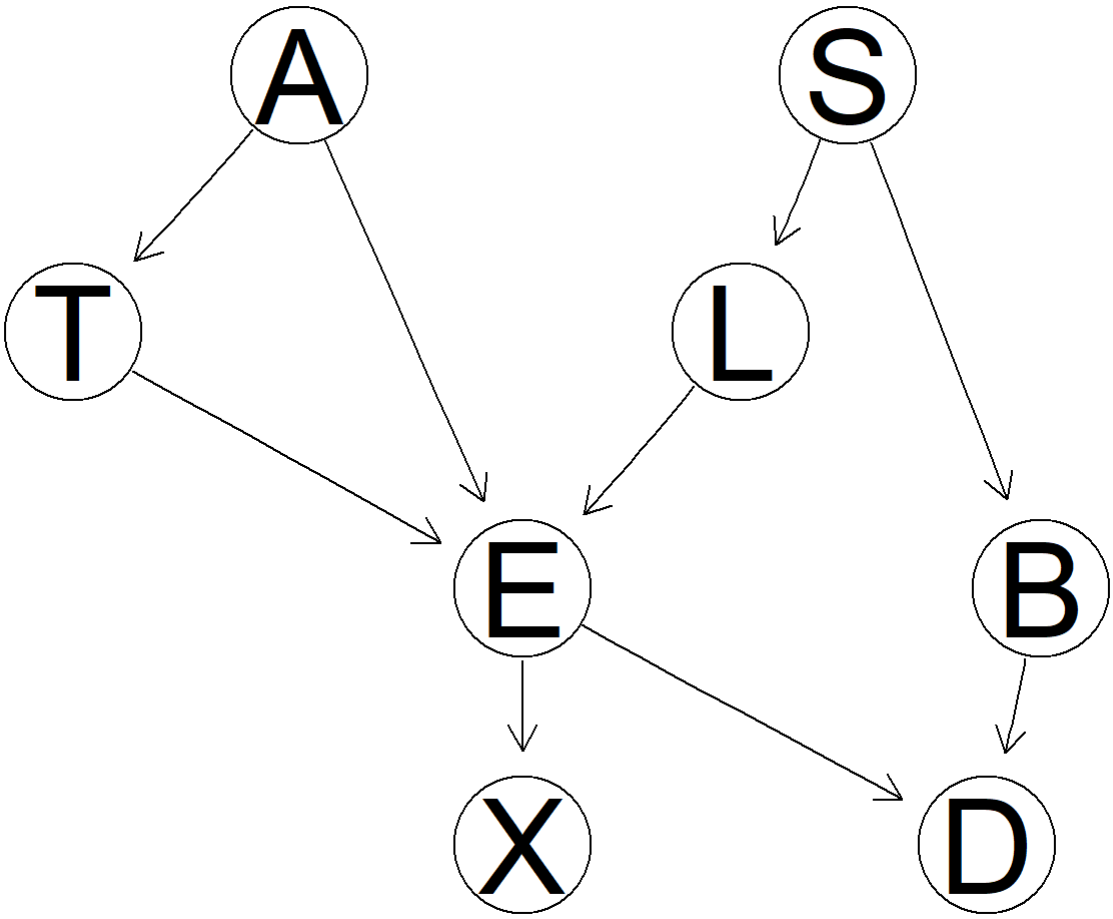
```
##    from to strength  direction
## 2     A  T    0.998 0.54208417
## 5     A  E    0.996 0.97590361
## 10    S  L    0.998 0.69338677
## 11    S  B    1.000 0.80900000
## 15    T  A    0.998 0.45791583
## 19    T  E    1.000 0.97300000
## 23    L  S    0.998 0.30661323
## 26    L  E    1.000 0.98700000
## 30    B  S    1.000 0.19100000
## 35    B  D    1.000 1.00000000
## 36    E  A    0.996 0.02409639
## 38    E  T    1.000 0.02700000
## 39    E  L    1.000 0.01300000
## 41    E  X    1.000 0.93600000
## 42    E  D    1.000 1.00000000
## 48    X  E    1.000 0.06400000
## 54    D  B    1.000 0.00000000
## 55    D  E    1.000 0.00000000
```

```
asia.boot[asia.boot$from == "A" & asia.boot$to=="E",]
```

```
##   from to strength direction
## 5    A  E    0.996 0.9759036
```

El campo strength define la confianza en la verosimilitud del arco según el modelo. Para tratar de eliminar el arco, podemos utilizar el argumento threshold, que impone un umbral mínimo de confianza para incluir los arcos en la red consenso.

```
asia.avg <- averaged.network(asia.boot, threshold=0.995)

graphviz.plot(asia.avg)
```



Ahora, la red es idéntica a la relación causal que se da en la realidad.

Vemos cómo se comportan los criterios de puntuación en las redes ajustadas hasta el momento.

Growth-Shrink

```
# criterio de puntuación BDe
score(asia.gs, data=asia, type="bde")
```

```
## [1] -12517.55
```

```
# criterio de puntuación Log-Likelihood
score(asia.gs, data=asia, type="loglik")
```

```
## [1] -12466.88
```

```
# criterio de puntuación AIC
score(asia.gs, data=asia, type="aic")
```

```
## [1] -12480.88
```

```
# criterio de puntuación BIC
score(asia.gs, data=asia, type="bic")
```

```
## [1] -12526.5
```

Hill-Climbing

```
# criterio de puntuación BDe
score(asia.hc, data=asia, type="bde")
```

```
## [1] -11095.79
```

```
# criterio de puntuación Log-Likelihood
score(asia.hc, data=asia, type="loglik")
```

```
## [1] -11034.9
```

```
# criterio de puntuación AIC
score(asia.hc, data=asia, type="aic")
```

```
## [1] -11051.9
```

```
# criterio de puntuación BIC
score(asia.hc, data=asia, type="bic")
```

```
## [1] -11107.29
```

Min-Max Hill Climbing

```
# criterio de puntuación BDe
score(asia.mmhc, data=asia, type="bde")
```

```
## [1] -12156.47
```

```
# criterio de puntuación Log-Likelihood
score(asia.mmhc, data=asia, type="loglik")
```

```
## [1] -12106.64
```

```
# criterio de puntuación AIC
score(asia.mmhc, data=asia, type="aic")
```

```
## [1] -12120.64
```

```
# criterio de puntuación BIC
score(asia.mmhc, data=asia, type="bic")
```

```
## [1] -12166.26
```

```
# criterio de puntuación BDe
score(asia.avg, data=asia, type="bde")
```

```
## [1] -11097.07
```

```
# criterio de puntuación Log-Likelihood
score(asia.avg, data=asia, type="loglik")
```

```
## [1] -11033.09
```

```
# criterio de puntuación AIC
score(asia.avg, data=asia, type="aic")
```

```
## [1] -11055.09
```

```
# criterio de puntuación BIC
score(asia.avg, data=asia, type="bic")
```

```
## [1] -11126.78
```

AJUSTE DE PARÁMETROS

Empleando la estructura consenso que acabamos de ajustar, podemos definir los parámetros de la red.

```
asia.fitted <- bn.fit(asia.avg, data=asia)
```

Empezando con los nodos *root*, aquellos que no tienen ningún ancestro:

```
asia.fitted
```

```
##
##   Bayesian network parameters
##
##   Parameters of node A (multinomial distribution)
##
## Conditional probability table:
##      no    yes
## 0.9916 0.0084
##
##   Parameters of node S (multinomial distribution)
##
## Conditional probability table:
##     no   yes
## 0.497 0.503
##
##   Parameters of node T (multinomial distribution)
##
## Conditional probability table:
##
##      A
## T              no         yes
##   no  0.991528842 0.952380952
##   yes 0.008471158 0.047619048
##
```

```
##   Parameters of node L (multinomial distribution)
##
## Conditional probability table:
##
##      S
## L             no        yes
##   no  0.98631791 0.88230616
##   yes 0.01368209 0.11769384
##
##   Parameters of node B (multinomial distribution)
##
## Conditional probability table:
##
##      S
## B            no       yes
##   no  0.7006036 0.2823062
##   yes 0.2993964 0.7176938
##
##   Parameters of node E (multinomial distribution)
##
## Conditional probability table:
##
## , , T = no, L = no
##
##      A
## E      no yes
##   no    1   1
##   yes   0   0
##
## , , T = yes, L = no
##
##      A
## E      no yes
##   no    0   0
##   yes   1   1
##
## , , T = no, L = yes
##
##      A
## E      no yes
##   no    0   0
##   yes   1   1
##
## , , T = yes, L = yes
##
##      A
## E      no yes
##   no    0
##   yes   1
##
##
##   Parameters of node X (multinomial distribution)
##
## Conditional probability table:
##
##      E
## X             no         yes
##   no  0.956587473 0.005405405
##   yes 0.043412527 0.994594595
##
##   Parameters of node D (multinomial distribution)
##
```

```
## Conditional probability table:
##
## , , E = no
##
##      B
## D             no        yes
##   no  0.90017286 0.21373057
##   yes 0.09982714 0.78626943
##
## , , E = yes
##
##      B
## D             no        yes
##   no  0.27737226 0.14592275
##   yes 0.72262774 0.85407725
```

```
asia.fitted$A
```

```
##
##   Parameters of node A (multinomial distribution)
##
## Conditional probability table:
##      no    yes
## 0.9916 0.0084
```

```
asia.fitted$S
```

```
##
##   Parameters of node S (multinomial distribution)
##
## Conditional probability table:
##     no   yes
## 0.497 0.503
```

Las tablas de probabilidad condicional de cada nodo son independendientes de otras variables, se fundamentan solo en las proporciones de cada nivel.

Considerando ahora los nodos que cuentan con un nodo parental:

```
asia.fitted$B
```

```
##
##   Parameters of node B (multinomial distribution)
##
## Conditional probability table:
##
##      S
## B            no       yes
##   no  0.7006036 0.2823062
##   yes 0.2993964 0.7176938
```

```
asia.fitted$L
```

```
##
##   Parameters of node L (multinomial distribution)
##
## Conditional probability table:
```

```
##
##      S
## L             no        yes
##   no  0.98631791 0.88230616
##   yes 0.01368209 0.11769384
```

```
asia.fitted$T
```

```
##
##   Parameters of node T (multinomial distribution)
##
## Conditional probability table:
##
##      A
## T             no         yes
##   no  0.991528842 0.952380952
##   yes 0.008471158 0.047619048
```

```
asia.fitted$X
```

```
##
##   Parameters of node X (multinomial distribution)
##
## Conditional probability table:
##
##      E
## X             no         yes
##   no  0.956587473 0.005405405
##   yes 0.043412527 0.994594595
```

Las tablas de probabilidad condicional de cada nodo expresan la distribución de niveles del nodo en función del nivel del nodo padre.

Por último, en los nodos centrales de una estructura-v:

```
asia.fitted$D
```

```
##
##   Parameters of node D (multinomial distribution)
##
## Conditional probability table:
##
## , , E = no
##
##      B
## D             no        yes
##   no  0.90017286 0.21373057
##   yes 0.09982714 0.78626943
##
## , , E = yes
##
##      B
## D             no        yes
##   no  0.27737226 0.14592275
##   yes 0.72262774 0.85407725
```

```
asia.fitted$E
```

```
##
##   Parameters of node E (multinomial distribution)
##
## Conditional probability table:
##
## , , T = no, L = no
##
##      A
## E      no yes
##   no    1   1
##   yes   0   0
##
## , , T = yes, L = no
##
##      A
## E      no yes
##   no    0   0
##   yes   1   1
##
## , , T = no, L = yes
##
##      A
## E      no yes
##   no    0   0
##   yes   1   1
##
## , , T = yes, L = yes
##
##      A
## E      no yes
##   no    0
##   yes   1
```

Cada tabla de probabilidad condicional recoge la distribución de niveles nodo en función del nivel de uno de sus padres, estando el nivel del otro padre predefinido.

```
asia.fitted
```

```
##
##   Bayesian network parameters
##
##   Parameters of node A (multinomial distribution)
##
## Conditional probability table:
##      no    yes
## 0.9916 0.0084
##
##   Parameters of node S (multinomial distribution)
##
## Conditional probability table:
##     no   yes
## 0.497 0.503
##
##   Parameters of node T (multinomial distribution)
##
## Conditional probability table:
##
##      A
## T              no         yes
##   no  0.991528842 0.952380952
```

```
##   yes 0.008471158 0.047619048
##
##   Parameters of node L (multinomial distribution)
##
## Conditional probability table:
##
##      S
## L             no        yes
##   no  0.98631791 0.88230616
##   yes 0.01368209 0.11769384
##
##   Parameters of node B (multinomial distribution)
##
## Conditional probability table:
##
##      S
## B            no       yes
##   no  0.7006036 0.2823062
##   yes 0.2993964 0.7176938
##
##   Parameters of node E (multinomial distribution)
##
## Conditional probability table:
##
## , , T = no, L = no
##
##      A
## E      no yes
##   no    1   1
##   yes   0   0
##
## , , T = yes, L = no
##
##      A
## E      no yes
##   no    0   0
##   yes   1   1
##
## , , T = no, L = yes
##
##      A
## E      no yes
##   no    0   0
##   yes   1   1
##
## , , T = yes, L = yes
##
##      A
## E      no yes
##   no    0
##   yes   1
##
##
##   Parameters of node X (multinomial distribution)
##
## Conditional probability table:
##
##      E
## X             no         yes
##   no  0.956587473 0.005405405
##   yes 0.043412527 0.994594595
##
```

```
##   Parameters of node D (multinomial distribution)
##
## Conditional probability table:
##
## , , E = no
##
##      B
## D             no        yes
##   no  0.90017286 0.21373057
##   yes 0.09982714 0.78626943
##
## , , E = yes
##
##      B
## D             no        yes
##   no  0.27737226 0.14592275
##   yes 0.72262774 0.85407725
```

```
graphviz.chart(asia.fitted)
```

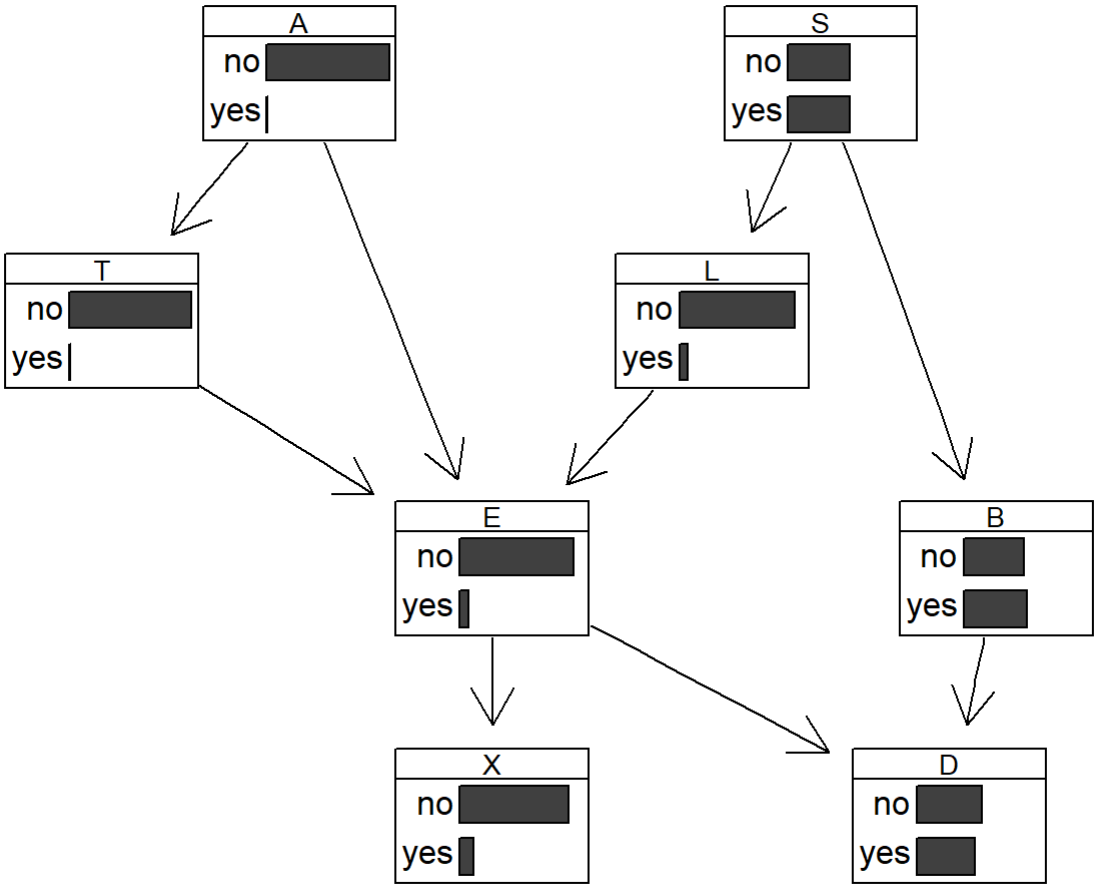
```
## Loading required namespace: gRain
```

```
## Warning in from.bn.fit.to.grain(x): NaN conditional probabilities in E, replaced
## with a uniform distribution.
```

```
## Warning: package 'gRbase' was built under R version 3.6.3
```

```
##
## Attaching package: 'gRbase'
```

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




```
library(questionr)

m1 <- matrix(0, 2, 2)

m1[1,1] <- nrow(asia[asia$S=='yes' & asia$L=='yes',])
m1[1,2] <- nrow(asia[asia$S=='yes' & asia$L=='no',])
m1[2,1] <- nrow(asia[asia$S=='no' & asia$L=='yes',])
m1[2,2] <- nrow(asia[asia$S=='no' & asia$L=='no',])

odds.ratio(m1)
```

```
##                   OR  2.5 % 97.5 %         p
## Fisher's test 9.6125 6.6903 14.212 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

ALARM

Beinlich IA, Suermondt HJ, Chavez RM, Cooper GF (1989) The ALARM monitoring system: a case study with two probabilistic inference techniques for belief networks. In: Proceedings of the 2nd European conference on artificial intelligence in medicine, Springer, pp 247–256

La red *ALARM* (Beinlich et al., 1989) fue diseñada para ofrecer un sistema de alarma en pacientes ingresados en unidades de cuidado intensivo (UCIs). Dado que *ALARM* es un hito muy recurrente en la literatura que se refiere a la estadística bayesiana, bnlearn incorpora un subconjunto de datos con 20000 observaciones de esta red, disponible bajo el nombre *alarm*.

CONSTRUCCIÓN DE LA RED

```
data(alarm)
```

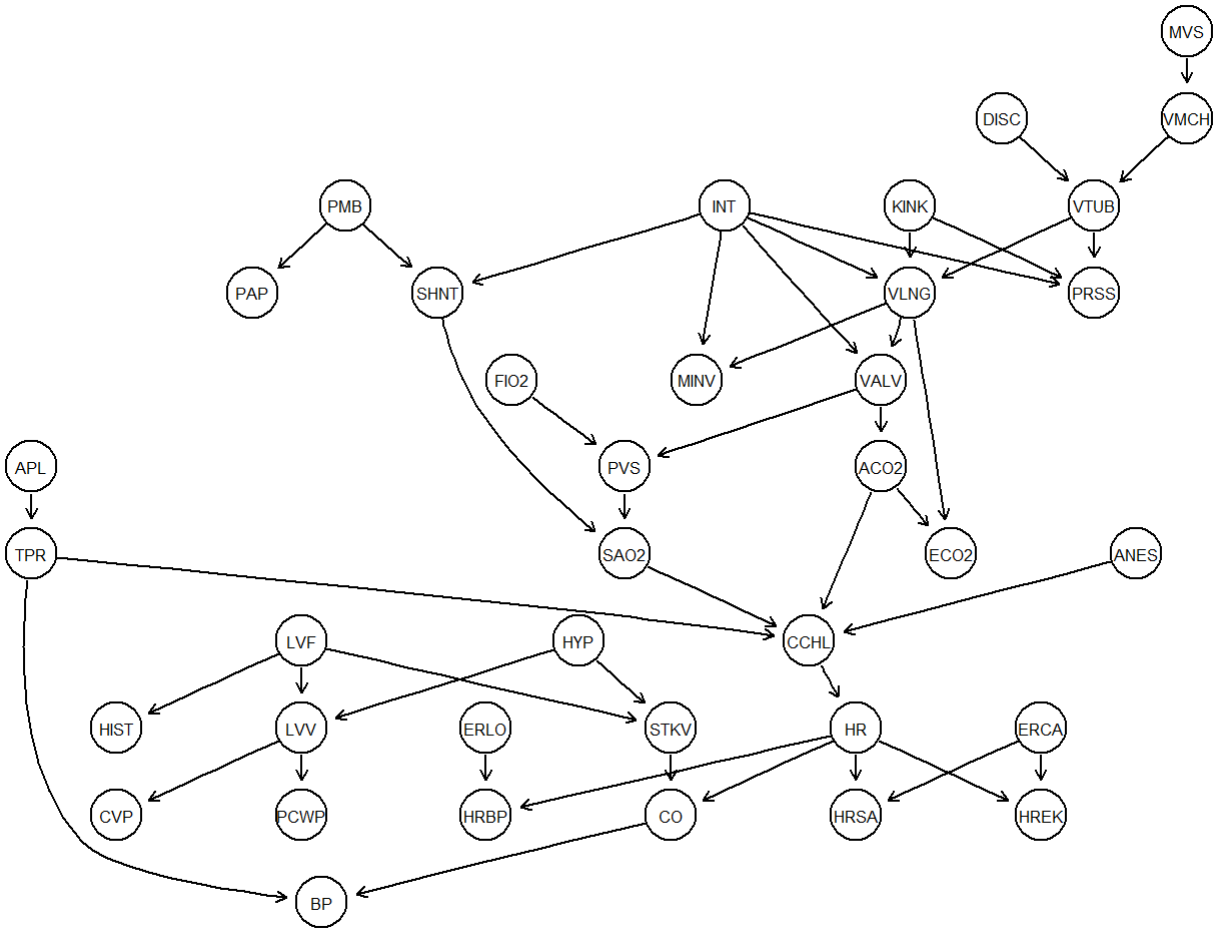
```
str(alarm)
```

```
## 'data.frame':    20000 obs. of  37 variables:
##  $ CVP : Factor w/ 3 levels "HIGH","LOW","NORMAL": 3 3 3 3 3 3 3 3 3 3 ...
##  $ PCWP: Factor w/ 3 levels "HIGH","LOW","NORMAL": 3 3 1 3 3 3 3 3 2 3 ...
##  $ HIST: Factor w/ 2 levels "FALSE","TRUE": 1 1 1 1 1 1 1 1 1 1 ...
##  $ TPR : Factor w/ 3 levels "HIGH","LOW","NORMAL": 2 3 3 2 2 2 2 3 2 3 ...
##  $ BP  : Factor w/ 3 levels "HIGH","LOW","NORMAL": 3 2 3 2 2 3 2 2 2 3 ...
##  $ CO  : Factor w/ 3 levels "HIGH","LOW","NORMAL": 1 2 1 1 3 1 1 2 1 1 ...
##  $ HRBP: Factor w/ 3 levels "HIGH","LOW","NORMAL": 1 1 1 1 1 1 1 1 1 1 ...
##  $ HREK: Factor w/ 3 levels "HIGH","LOW","NORMAL": 1 1 1 1 1 1 1 3 1 1 ...
##  $ HRSA: Factor w/ 3 levels "HIGH","LOW","NORMAL": 1 1 1 1 1 1 1 3 1 1 ...
##  $ PAP : Factor w/ 3 levels "HIGH","LOW","NORMAL": 3 3 3 3 3 3 3 2 3 2 ...
##  $ SAO2: Factor w/ 3 levels "HIGH","LOW","NORMAL": 3 2 2 3 2 2 2 2 2 2 ...
##  $ FIO2: Factor w/ 2 levels "LOW","NORMAL": 1 2 2 2 2 2 1 2 2 2 ...
##  $ PRSS: Factor w/ 4 levels "HIGH","LOW","NORMAL",..: 1 1 3 1 2 1 2 1 1 1 ...
##  $ ECO2: Factor w/ 4 levels "HIGH","LOW","NORMAL",..: 4 4 4 4 4 1 4 4 4 2 ...
##  $ MINV: Factor w/ 4 levels "HIGH","LOW","NORMAL",..: 1 4 4 4 4 4 4 4 4 4 ...
##  $ MVS : Factor w/ 3 levels "HIGH","LOW","NORMAL": 3 3 3 3 3 3 3 3 3 3 ...
##  $ HYP : Factor w/ 2 levels "FALSE","TRUE": 1 1 1 1 1 1 1 1 1 1 ...
##  $ LVF : Factor w/ 2 levels "FALSE","TRUE": 1 1 1 1 1 1 1 1 1 1 ...
##  $ APL : Factor w/ 2 levels "FALSE","TRUE": 1 1 1 1 1 1 1 1 1 1 ...
##  $ ANES: Factor w/ 2 levels "FALSE","TRUE": 1 1 1 1 1 2 1 1 1 1 ...
##  $ PMB : Factor w/ 2 levels "FALSE","TRUE": 1 1 1 1 1 1 1 1 1 1 ...
##  $ INT : Factor w/ 3 levels "ESOPHAGEAL","NORMAL",..: 2 2 2 2 2 2 2 2 2 1 ...
##  $ KINK: Factor w/ 2 levels "FALSE","TRUE": 1 1 1 1 1 1 1 1 2 1 ...
##  $ DISC: Factor w/ 2 levels "FALSE","TRUE": 2 1 1 1 1 1 1 1 1 1 ...
##  $ LVV : Factor w/ 3 levels "HIGH","LOW","NORMAL": 3 3 3 3 3 3 3 3 3 3 ...
##  $ STKV: Factor w/ 3 levels "HIGH","LOW","NORMAL": 3 2 3 3 3 3 3 2 3 3 ...
##  $ CCHL: Factor w/ 2 levels "HIGH","NORMAL": 1 1 1 1 1 1 1 1 1 1 ...
##  $ ERLO: Factor w/ 2 levels "FALSE","TRUE": 1 1 1 1 1 1 1 1 1 1 ...
##  $ HR  : Factor w/ 3 levels "HIGH","LOW","NORMAL": 1 1 1 1 1 1 1 1 1 1 ...
##  $ ERCA: Factor w/ 2 levels "FALSE","TRUE": 1 1 1 1 1 1 1 2 1 1 ...
##  $ SHNT: Factor w/ 2 levels "HIGH","NORMAL": 2 2 2 2 2 2 2 2 2 2 ...
##  $ PVS : Factor w/ 3 levels "HIGH","LOW","NORMAL": 3 2 2 3 2 2 2 2 2 2 ...
##  $ ACO2: Factor w/ 3 levels "HIGH","LOW","NORMAL": 3 2 2 2 2 2 2 2 2 2 ...
##  $ VALV: Factor w/ 4 levels "HIGH","LOW","NORMAL",..: 1 4 4 4 4 4 4 4 4 4 ...
##  $ VLNG: Factor w/ 4 levels "HIGH","LOW","NORMAL",..: 2 4 4 4 4 4 4 4 4 2 ...
##  $ VTUB: Factor w/ 4 levels "HIGH","LOW","NORMAL",..: 4 2 2 2 2 2 2 2 2 2 ...
##  $ VMCH: Factor w/ 4 levels "HIGH","LOW","NORMAL",..: 3 3 3 3 3 3 3 3 3 3 ...
```

Cada una de las variables se refiere a una medida con valor clínico, en formato factorial (p.ej: ACO2=CO2 arterial -> ALTO, BAJO, NORMAL). Se trata de un conjunto de datos que contiene 20000 observaciones con 37 campos:

Si atendemos al trabajo de *Beinlich et al.*, podemos obtener la estructura real de la red bayesiana:

```
alarm.model = paste0("[HIST|LVF][CVP|LVV][PCWP|LVV][HYP][LVV|HYP:LVF][LVF]",
  "[STKV|HYP:LVF][ERLO][HRBP|ERLO:HR][HREK|ERCA:HR][ERCA][HRSA|ERCA:HR][ANES]",
  "[APL][TPR|APL][ECO2|ACO2:VLNG][KINK][MINV|INT:VLNG][FIO2][PVS|FIO2:VALV]",
  "[SAO2|PVS:SHNT][PAP|PMB][PMB][SHNT|INT:PMB][INT][PRSS|INT:KINK:VTUB][DISC]",
  "[MVS][VMCH|MVS][VTUB|DISC:VMCH][VLNG|INT:KINK:VTUB][VALV|INT:VLNG]",
  "[ACO2|VALV][CCHL|ACO2:ANES:SAO2:TPR][HR|CCHL][CO|HR:STKV][BP|CO:TPR]")

alarm.bn <- model2network(alarm.model)

graphviz.plot(alarm.bn)
```




```
alarm.arcs <- narcs(alarm.bn)
alarm.nodes <- nnodes(alarm.bn)

alarm.arcs
```

```
## [1] 46
```

```
alarm.nodes
```

```
## [1] 37
```

```
alarm.density <- alarm.arcs/(alarm.nodes*(alarm.nodes-1))
alarm.density
```

```
## [1] 0.03453453
```

AJUSTE DE ESTRUCTURA

Podemos probar diversos algoritmos basados en restricciones para construir la red desde 0, suponiendo que no dispusiéramos de información previa:

```
# algoritmo PC
alarm.pc <- pc.stable(alarm)
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure ECO2 -> ACO2 <- CCHL is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure SHNT -> SAO2 <- PVS is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CCHL -> ACO2 <- VALV is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CCHL -> SAO2 <- SHNT is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
# algoritmo Grow-Shrink
alarm.gs <- gs(alarm)
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure ECO2 -> MINV <- INT is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure HREK -> HRBP <- ERLO is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure BP -> TPR <- CCHL is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CO -> HRBP <- ERLO is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CO -> HRBP <- HREK is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure SAO2 -> MINV <- ECO2 is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure BP -> CO <- STKV is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure FIO2 -> SAO2 <- MINV is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
# algoritmo Incrementa-Association
alarm.iamb <- iamb(alarm)
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure BP -> TPR <- CCHL is not applicable, because one or both arcs are
```

```
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure BP -> CO <- STKV is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure BP -> CO <- HR is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure HYP -> LVV <- LVF is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
# algoritmo Max-Min Parents and Children
alarm.mmpc <- mmpc(alarm)
```

```
# algoritmo Hill-Climbing
alarm.hc <- hc(alarm)
# algoritmo Tabu
alarm.tabu <- tabu(alarm)
```

```
# algoritmo Min-Max Hill-Climbing
alarm.mmhc <- mmhc(alarm)
```

```
compare(alarm.bn, alarm.pc)
```

```
## $tp
## [1] 30
##
## $fp
## [1] 12
##
## $fn
## [1] 16
```

```
compare(alarm.bn, alarm.iamb)
```

```
## $tp
## [1] 16
##
## $fp
## [1] 18
##
## $fn
## [1] 30
```

```
compare(alarm.bn, alarm.mmpc)
```

```
## $tp
## [1] 0
##
## $fp
```

```
## [1] 32
##
## $fn
## [1] 46
```

```
compare(alarm.bn, alarm.gs)
```

```
## $tp
## [1] 5
##
## $fp
## [1] 14
##
## $fn
## [1] 41
```

```
compare(alarm.bn, alarm.hc)
```

```
## $tp
## [1] 22
##
## $fp
## [1] 31
##
## $fn
## [1] 24
```

```
compare(alarm.bn, alarm.tabu)
```

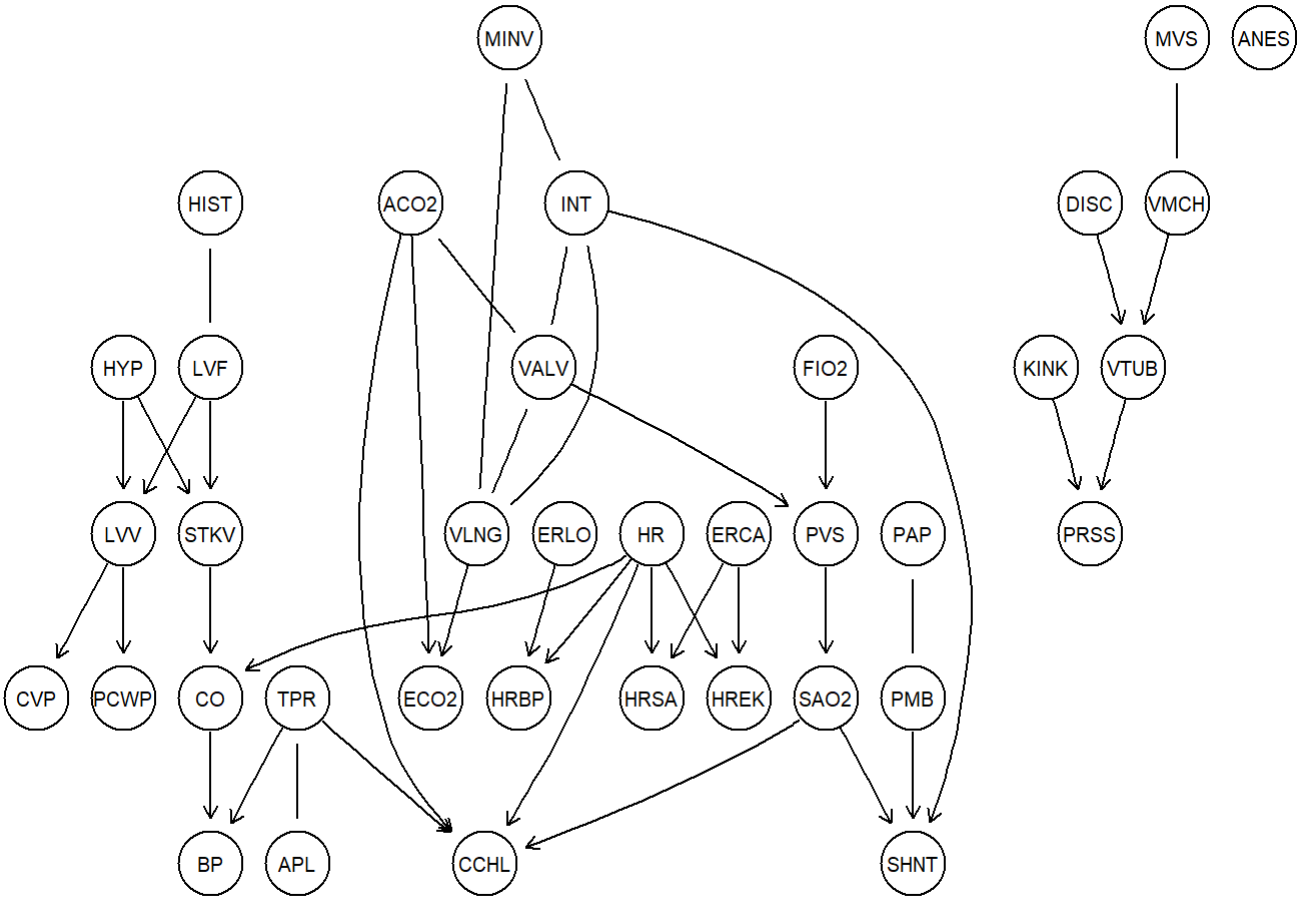
```
## $tp
## [1] 26
##
## $fp
## [1] 25
##
## $fn
## [1] 20
```

```
compare(alarm.bn, alarm.mmhc)
```

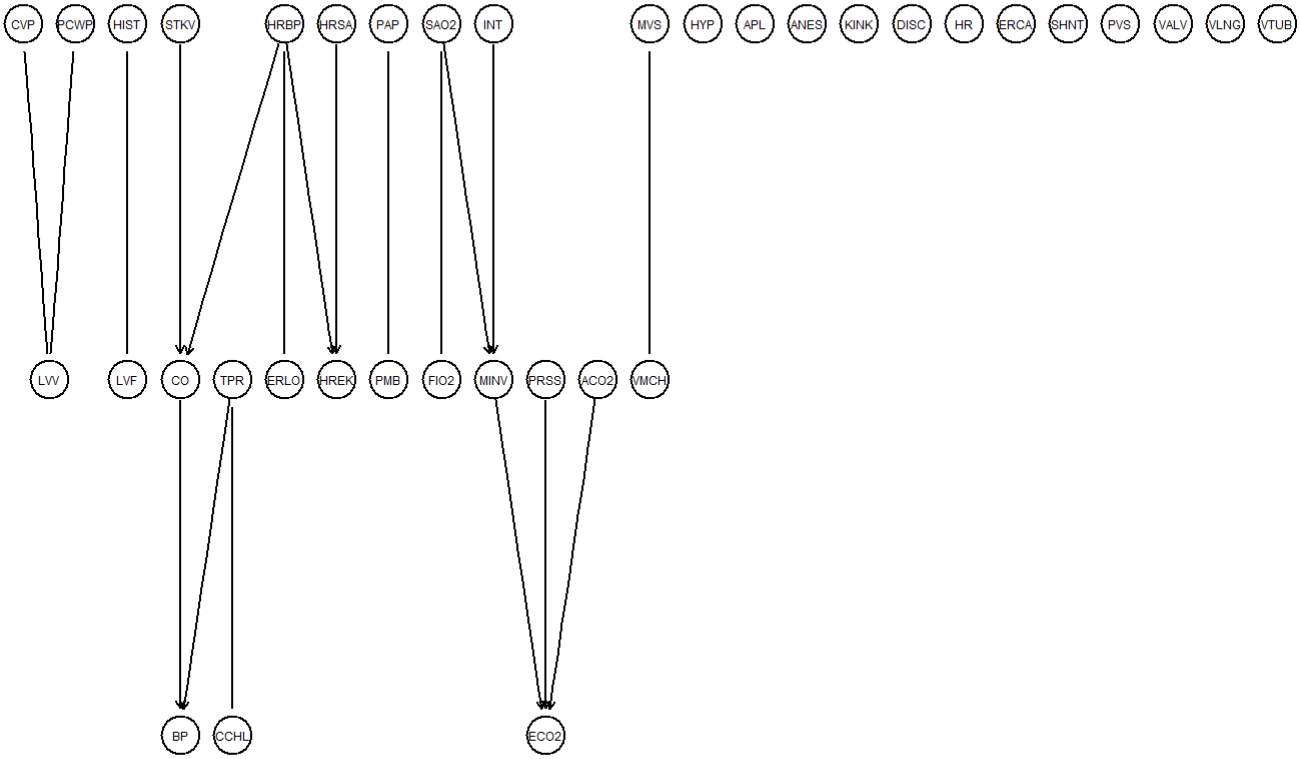
```
## $tp
## [1] 15
##
## $fp
## [1] 17
##
## $fn
## [1] 31
```

Si observamos los resultados gráficamente:

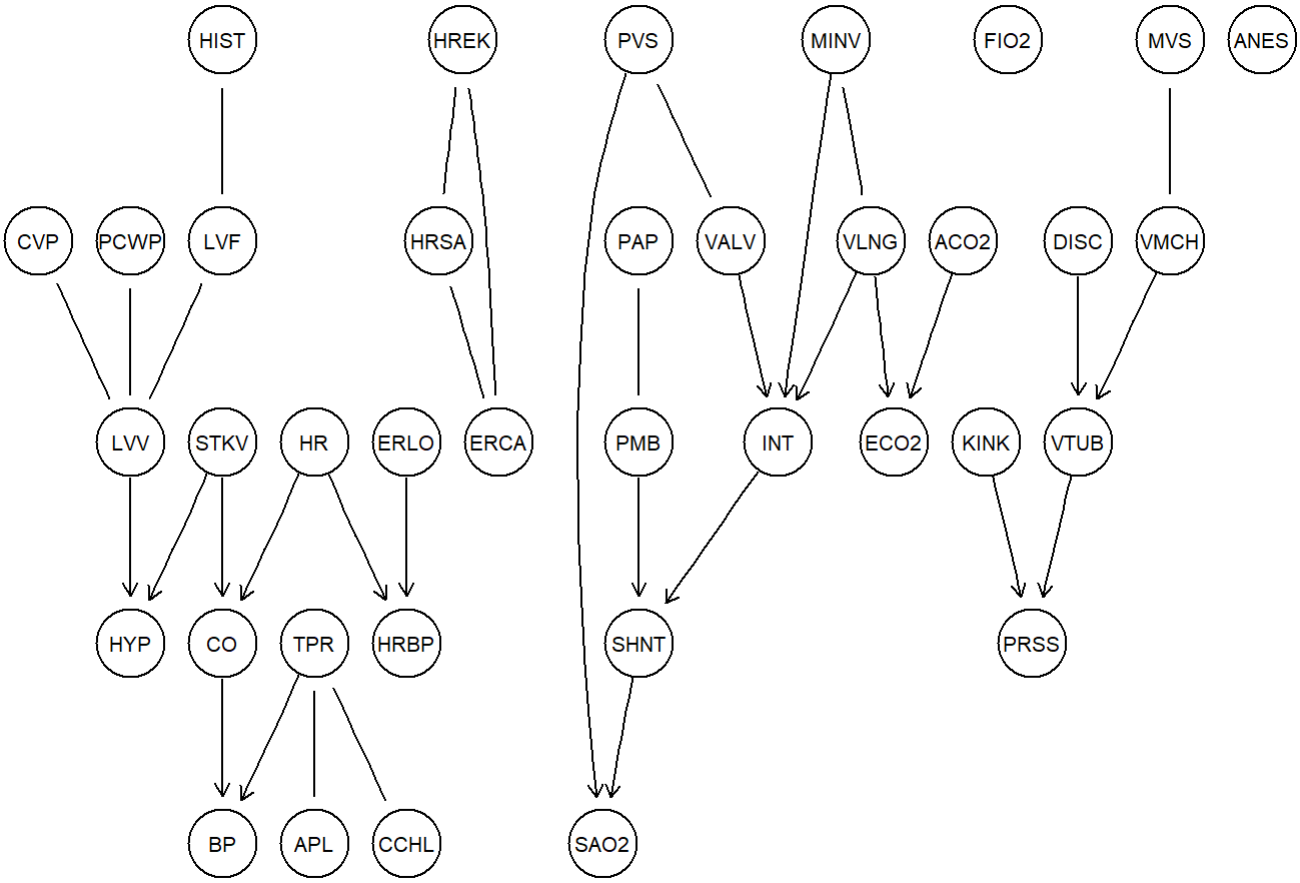
```
graphviz.plot(alarm.pc)
```



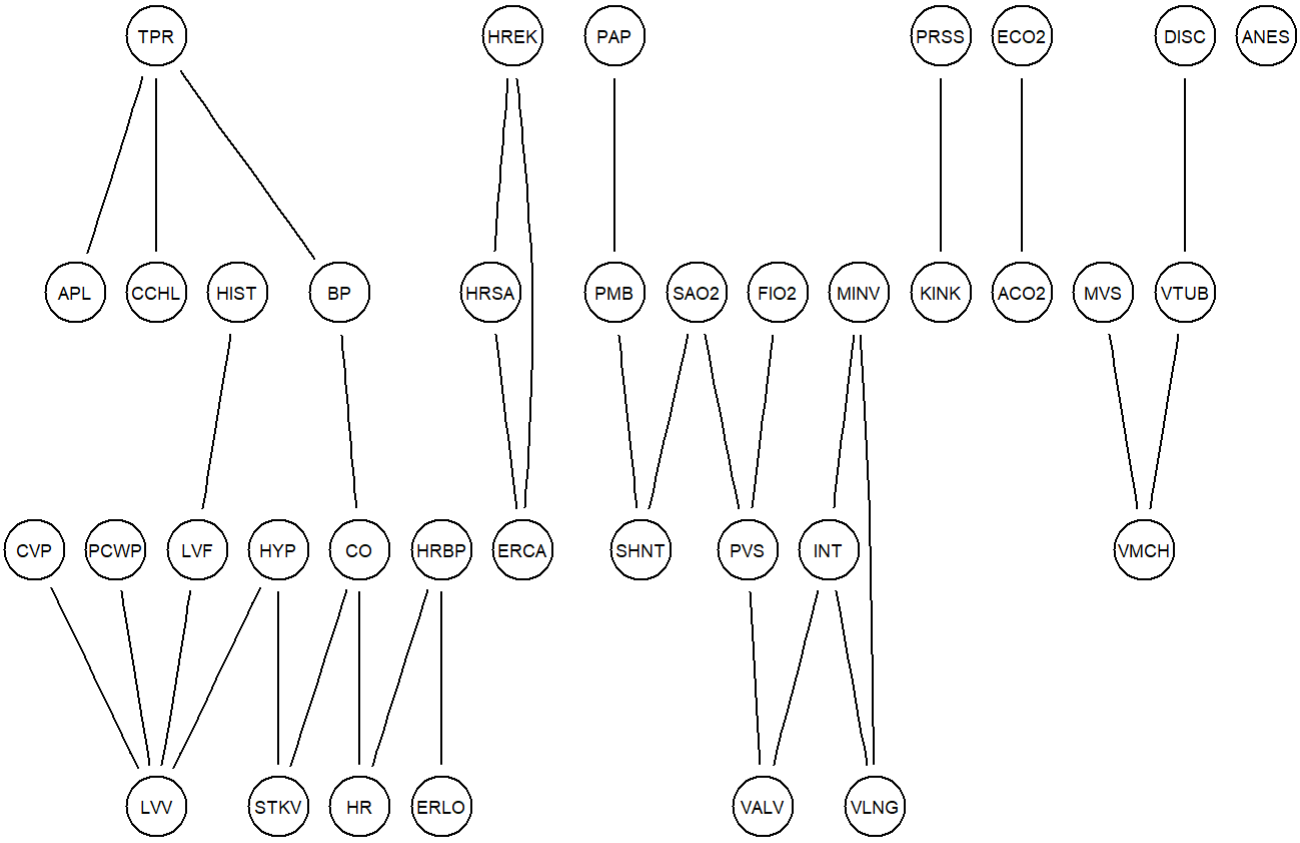

```
graphviz.plot(alarm.gs)
```




```
graphviz.plot(alarm.iamb)
```




```
graphviz.plot(alarm.mmpc)
```



Vemos que todos estos algoritmos originan redes de muy baja complejidad en comparación a la estructura original de los datos; este es un problema típico en redes con un gran número de nodos y tiene que ver con la escala exponencial de la dimensionalidad. En cualquier caso, de entre todos ellos el que presenta una relación de dependencia más cercana a la realidad es el algoritmo PC.

Por defecto, los algoritmos basados en restricciones utilizan tests de independencia condicional (Chi-cuadrado asintótica). ¿Tendríamos resultados más coherentes con la realidad si utilizásemos tests de permutación o de encogimiento (shrinkage)?

The overall performance of constraint-based algorithms suggests that the asymptotic χ^2 conditional independence tests may not be appropriate for analyzing alarm. Are permutation or shrinkage tests better choices?

```
# permutación: Monte-Carlo Permutation Test
alarm.pc.per <- pc.stable(alarm, test="mc-x2")
```

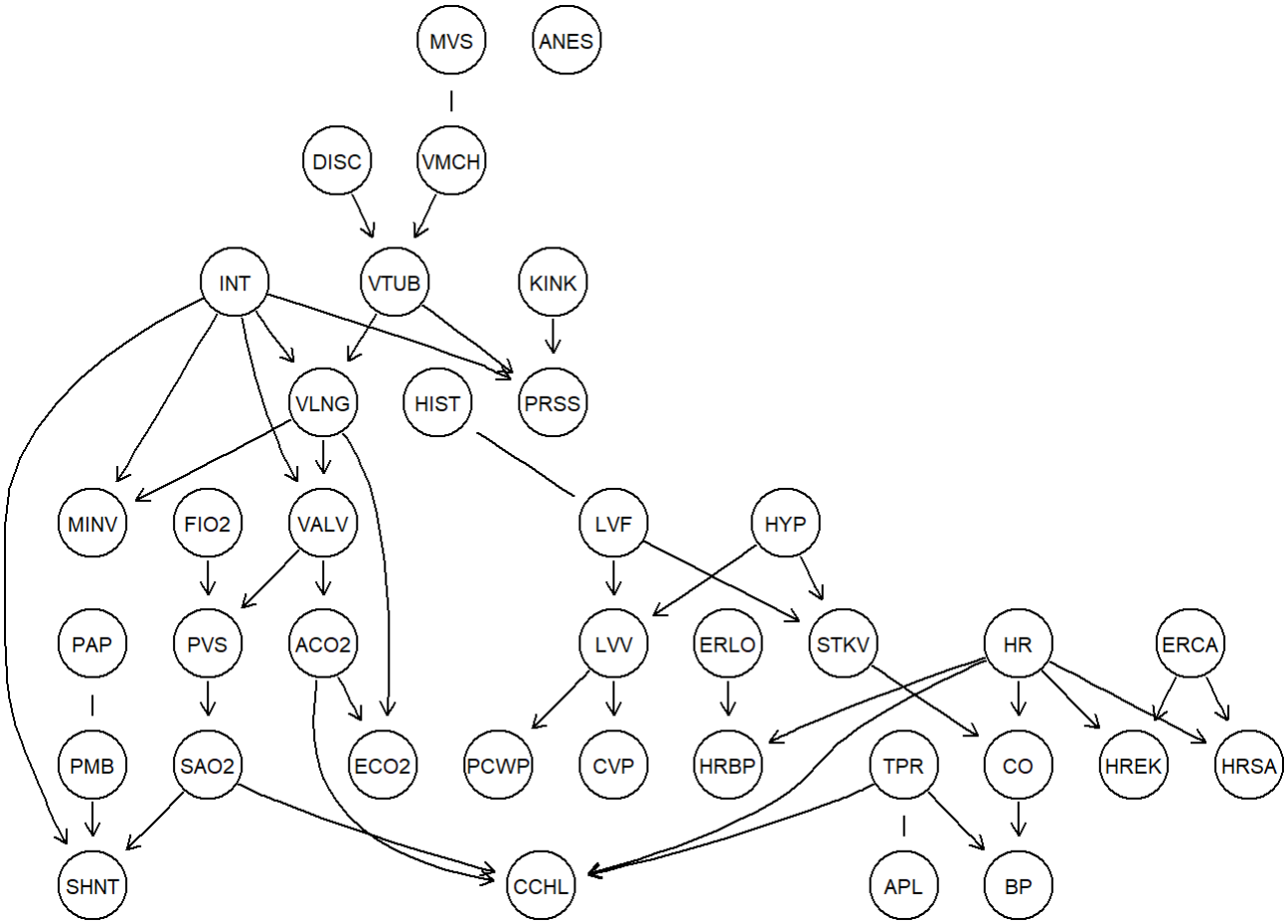
```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure ECO2 -> ACO2 <- CCHL is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CCHL -> ACO2 <- VALV is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure SHNT -> SAO2 <- PVS is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CCHL -> SAO2 <- SHNT is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
graphviz.plot(alarm.pc.per)
```




```
# encogimiento: mutual information shrinkage estimator
alarm.pc.shr <- pc.stable(alarm, test="mi-sh")
```

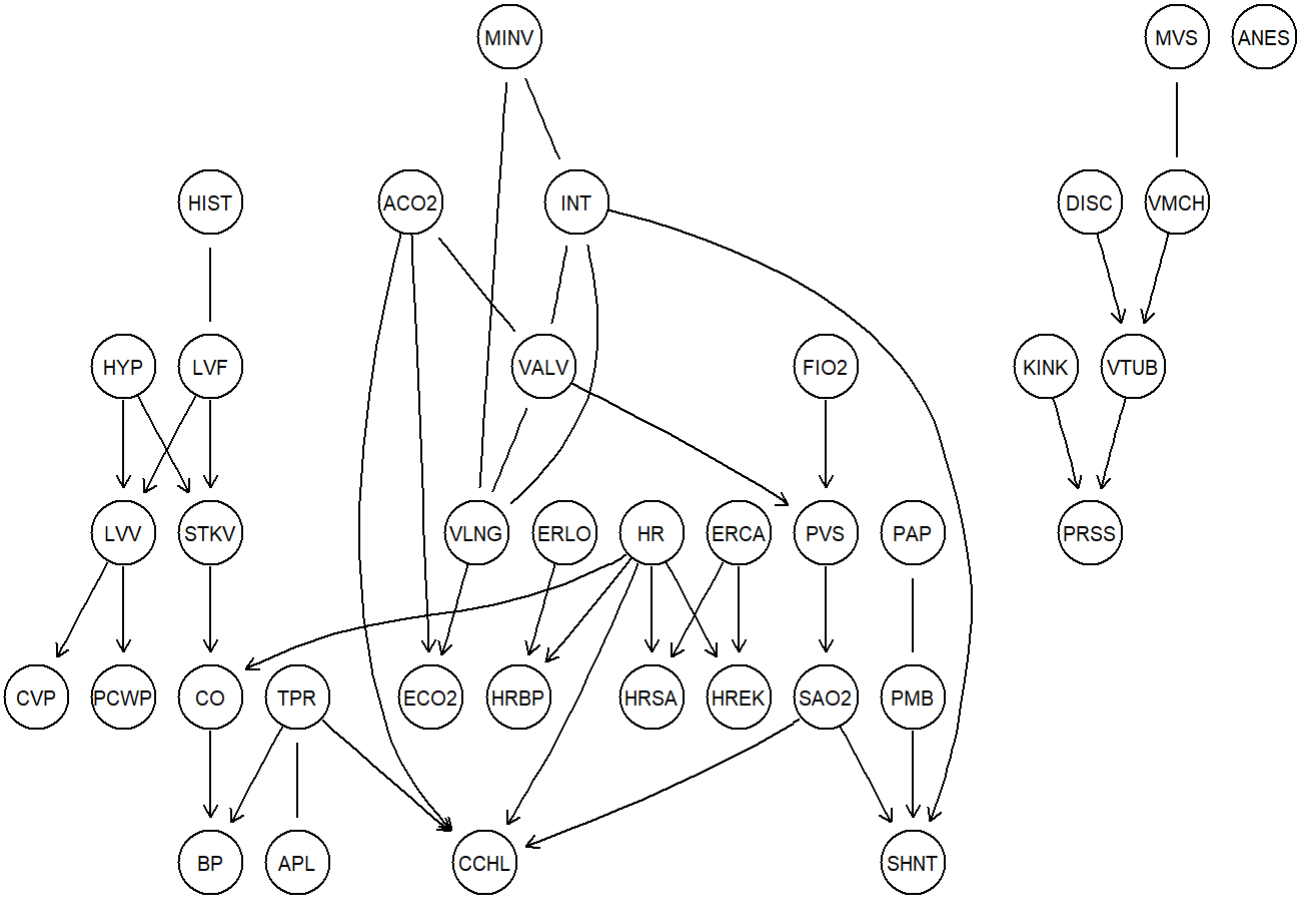
```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure ECO2 -> ACO2 <- CCHL is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure SHNT -> SAO2 <- PVS is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CCHL -> ACO2 <- VALV is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

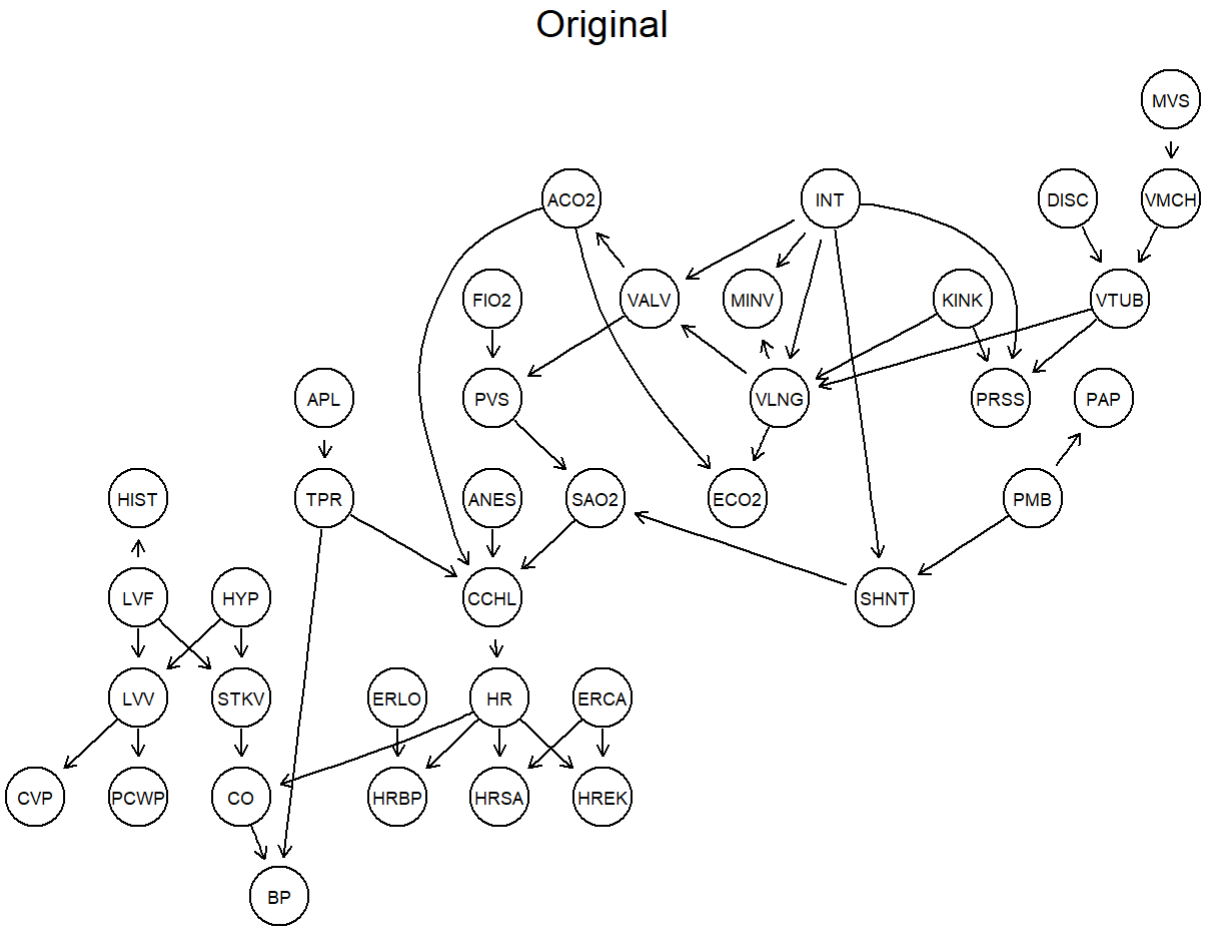
```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CCHL -> SAO2 <- SHNT is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

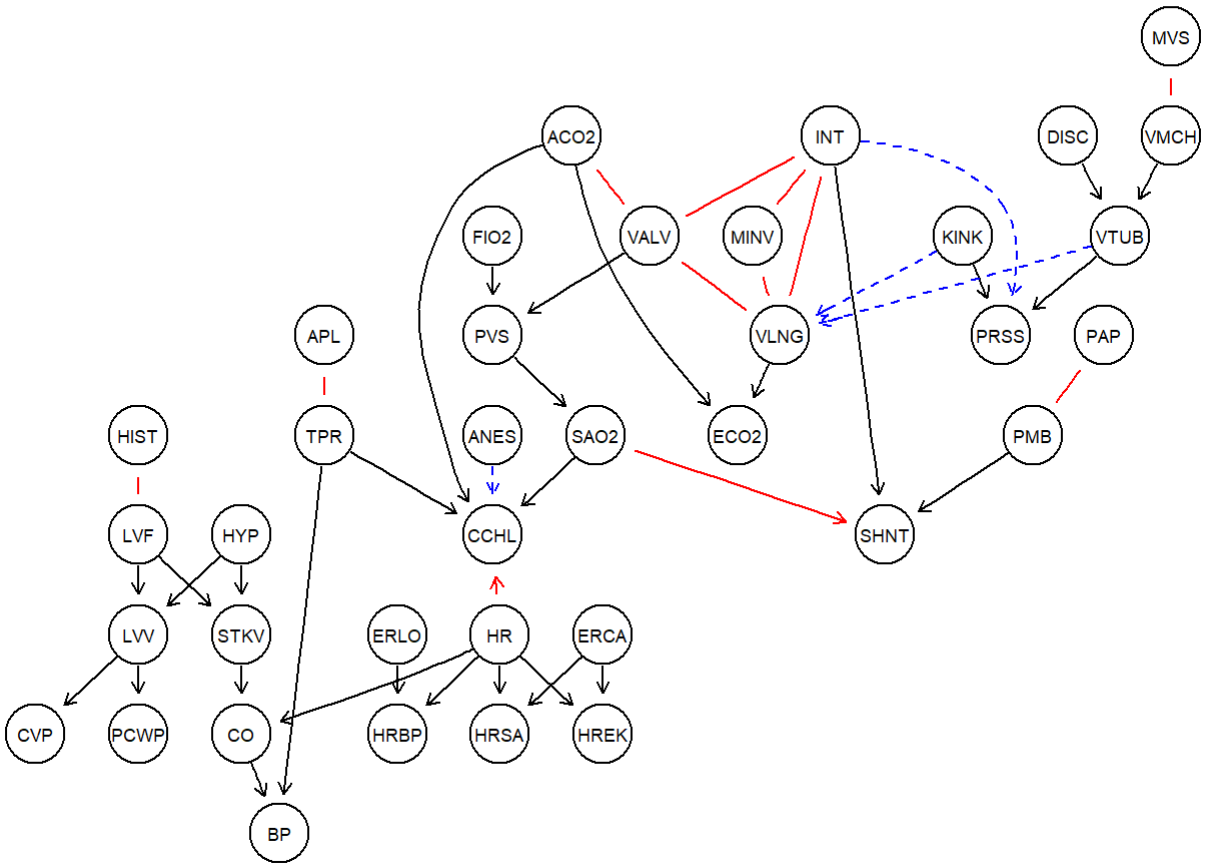
```
graphviz.plot(alarm.pc.shr)
```



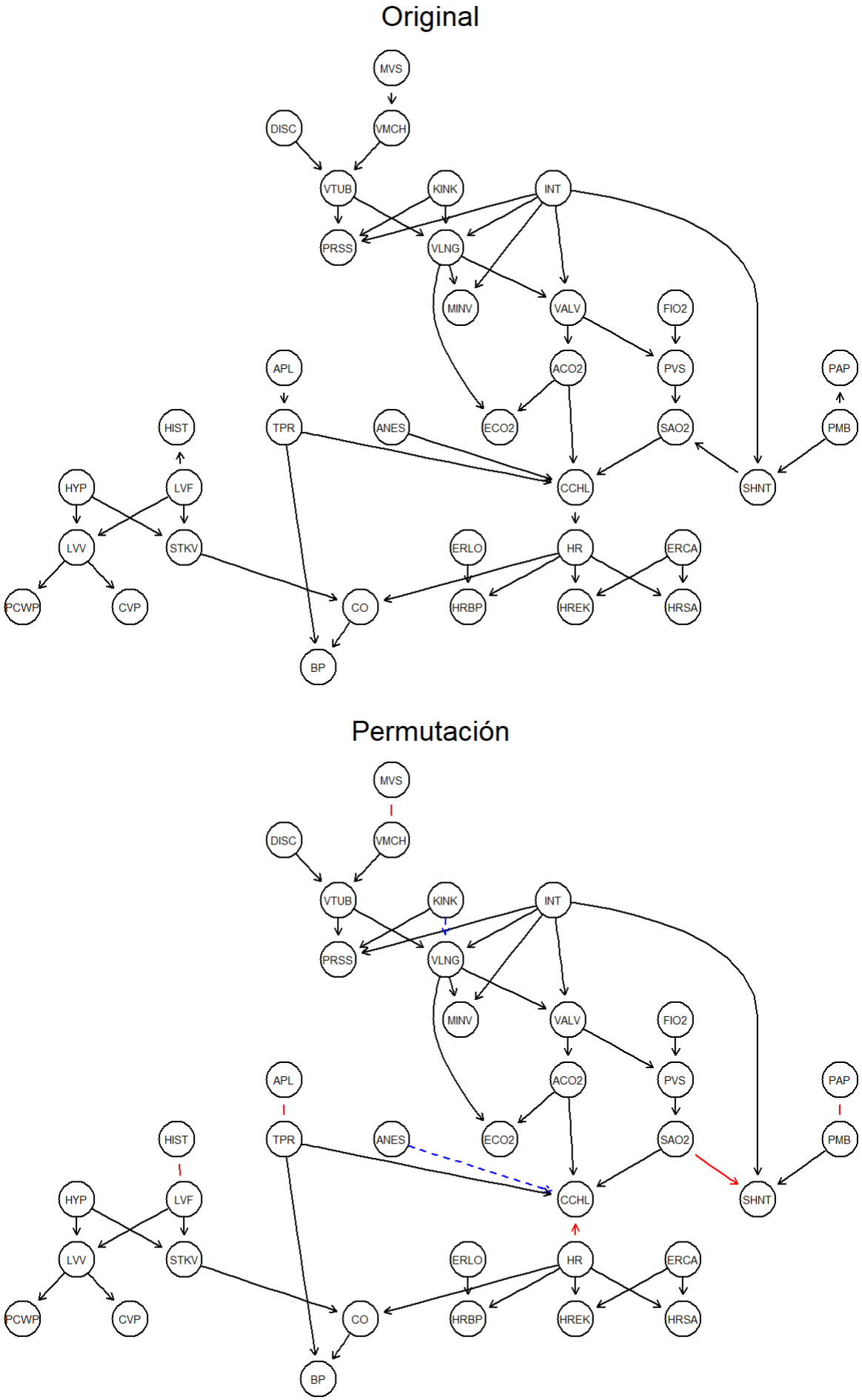
Veamos gráficamente cómo se diferencia cada uno de estos modelos de la estructura real:

```
# test de independencia condicional
graphviz.compare(alarm.bn, alarm.pc, main=c("Original", "Independencia condicional"))
```



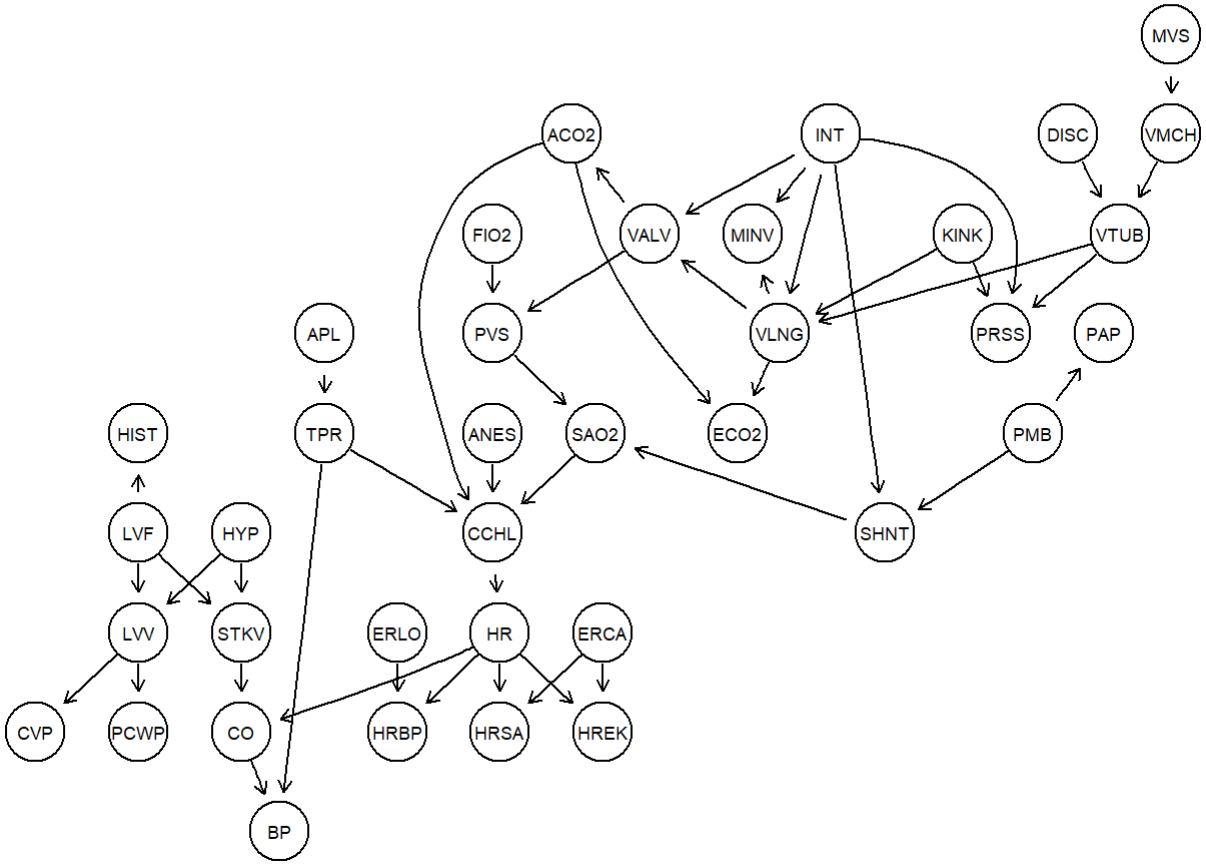



```
# test de permutación
graphviz.compare(alarm.bn, alarm.pc.per, main=c("Original", "Permutación"))
```

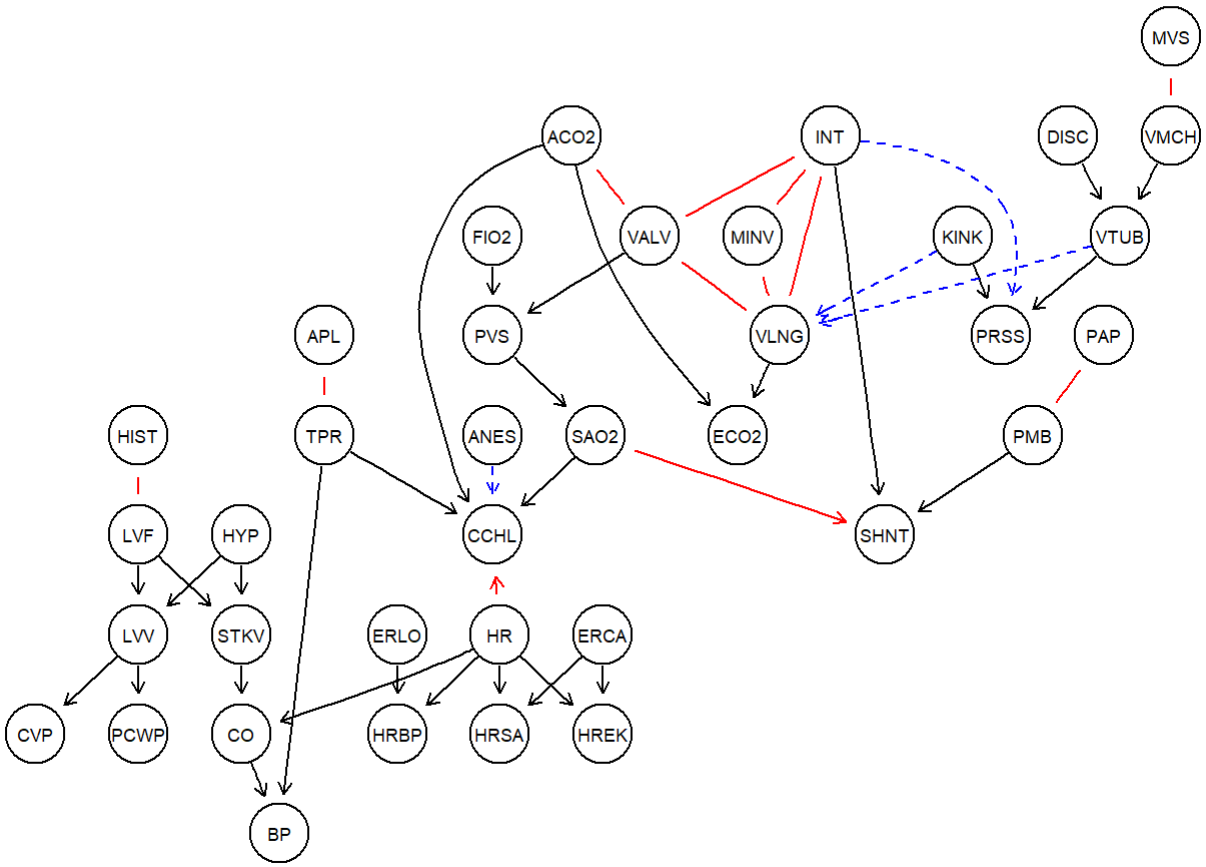



```
# test de encogimiento
graphviz.compare(alarm.bn, alarm.pc.shr, main=c("Original", "Encogimiento"))
```

Original



Encogimiento



Al parecer, el algoritmo PC que incorpora el test de permutación es el que mejor se ajusta a la naturaleza de los datos. Podríamos ver esto matemáticamente en forma de número de nodos coincidentes (TP: verdaderos positivos) y discrepantes (FP: falsos positivos; FN: falsos negativos):

```
# test de independencia condicional
compare(alarm.bn, alarm.pc)
```

```
## $tp
## [1] 30
##
## $fp
## [1] 12
##
## $fn
## [1] 16
```

```
# test de permutación
compare(alarm.bn, alarm.pc.per)
```

```
## $tp
## [1] 38
##
## $fp
## [1] 6
##
## $fn
## [1] 8
```

```
# test de encogimiento
compare(alarm.bn, alarm.pc.shr)
```

```
## $tp
## [1] 30
##
## $fp
## [1] 12
##
## $fn
## [1] 16
```

De entre los otros parámetros que podemos modificar en el auste de la red, uno de los más destacables es el grado de confianza, concretamente modificando su inverso α, es decir, el ratio de errores de tipo I (rechazar la hipótesis nula cuando esta es cierta, lo que en redes bayesianas se traduciría en declarar arcos cuando no existe una verdadera relación de dependencia entre los dos nodos). Asumiendo nuevamente el algoritmo PC basado en tests de permutación, ¿Cómo afecta el cambio en α a la estructura de red?

```
alpha.values <- c(0.005, 0.01, 0.05, 0.1, 0.2)

comparisons <- lapply(alpha.values, function(alpha){
  model <- pc.stable(alarm, test="mc-x2", alpha=alpha)
  comparison <- compare(alarm.bn, model)
  return(comparison)
  })
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure FIO2 -> PVS <- VALV is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure SHNT -> SAO2 <- PVS is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CCHL -> SAO2 <- SHNT is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure SHNT -> SAO2 <- PVS is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CCHL -> SAO2 <- SHNT is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure ECO2 -> ACO2 <- CCHL is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CCHL -> ACO2 <- VALV is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure SHNT -> SAO2 <- PVS is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CCHL -> SAO2 <- SHNT is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure ECO2 -> ACO2 <- CCHL is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure KINK -> PRSS <- VTUB is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure INT -> PRSS <- VTUB is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure INT -> VLNG <- VTUB is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure SHNT -> SAO2 <- PVS is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CCHL -> ACO2 <- VALV is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CCHL -> SAO2 <- SHNT is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure ECO2 -> ACO2 <- CCHL is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CCHL -> ACO2 <- VALV is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CCHL -> SAO2 <- SHNT is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
names(comparisons) <- as.character(alpha.values)
```

Lo esperable es que valores de α más restrictivos den lugar a un menor número de enlaces espúreos, aunque a expensas de disminuir también los verdaderos positivos. Sin embargo, viendo los resultados:

```
comparisons
```

```
## $`0.005`
## $`0.005`$tp
## [1] 30
##
## $`0.005`$fp
## [1] 12
##
## $`0.005`$fn
## [1] 16
##
##
## $`0.01`
## $`0.01`$tp
## [1] 36
##
## $`0.01`$fp
## [1] 6
##
## $`0.01`$fn
## [1] 10
##
##
## $`0.05`
## $`0.05`$tp
## [1] 38
##
## $`0.05`$fp
## [1] 6
##
## $`0.05`$fn
## [1] 8
##
##
## $`0.1`
```

```
## $`0.1`$tp
## [1] 30
##
## $`0.1`$fp
## [1] 14
##
## $`0.1`$fn
## [1] 16
##
##
## $`0.2`
## $`0.2`$tp
## [1] 39
##
## $`0.2`$fp
## [1] 5
##
## $`0.2`$fn
## [1] 7
```

$\alpha = 0.2$ ofrece el número de verdaderos positivos más alto, sin que haya penalización en forma de más falsos positivos y negativos. Parece que, en redes complejas, es poco recomendable introducir parámetros demasiado restrictivos, porque contribuyen a limitar la potencia de R para trabajar con múltipes dimensiones. Probando con un nuevo conjunto de valores más altos para α:

```
alpha.values.2 <- c(0.3, 0.4, 0.5, 0.75)

comparisons.2 <- lapply(alpha.values.2, function(alpha){
  model <- pc.stable(alarm, test="mc-x2", alpha=alpha)
  comparison <- compare(alarm.bn, model)
  return(comparison)
  })
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure ACO2 -> ECO2 <- VLNG is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure HYP -> STKV <- LVF is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure SAO2 -> CCHL <- ACO2 is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure HR -> CCHL <- ACO2 is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure HIST -> PCWP <- LVV is not applicable, because one or both arcs
## introduce cycles in the graph.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CCHL -> SAO2 <- SHNT is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure TPR -> CCHL <- ACO2 is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure TPR -> BP <- CO is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure HIST -> APL <- TPR is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CO -> HREK <- ERCA is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CCHL -> ACO2 <- VALV is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CCHL -> SAO2 <- SHNT is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure PRSS -> FIO2 <- PVS is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure HIST -> APL <- TPR is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CCHL -> ACO2 <- VALV is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CCHL -> SAO2 <- SHNT is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure PRSS -> MINV <- VALV is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure HYP -> STKV <- LVF is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure APL -> HIST <- STKV is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CVP -> HYP <- PCWP is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CCHL -> ACO2 <- VLNG is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure PCWP -> STKV <- BP is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure SAO2 -> FIO2 <- PRSS is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure PRSS -> ACO2 <- ECO2 is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure PRSS -> VTUB <- MINV is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure HRBP -> HRSA <- HREK is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure BP -> APL <- DISC is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure FIO2 -> PRSS <- VTUB is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure LVF -> HIST <- APL is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure PRSS -> MVS <- DISC is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure INT -> PRSS <- VTUB is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure PVS -> ACO2 <- VLNG is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure MINV -> ECO2 <- ACO2 is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure PRSS -> MVS <- VMCH is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CVP -> HYP <- STKV is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CCHL -> ACO2 <- PVS is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure PRSS -> ACO2 <- PVS is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure ACO2 -> PRSS <- VTUB is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure ACO2 -> ECO2 <- VTUB is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure PCWP -> LVV <- PMB is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure KINK -> PRSS <- VTUB is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure KINK -> ECO2 <- VTUB is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure HR -> CCHL <- ACO2 is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure PCWP -> HIST <- APL is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure SHNT -> SAO2 <- PVS is not applicable, because one or both arcs
## introduce cycles in the graph.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure ECO2 -> ACO2 <- PVS is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure PRSS -> ACO2 <- CCHL is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure HREK -> CO <- STKV is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure MINV -> VTUB <- DISC is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure KINK -> PRSS <- ACO2 is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure KINK -> ECO2 <- ACO2 is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure HIST -> APL <- BP is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure HIST -> STKV <- BP is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure MVS -> DISC <- APL is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure FIO2 -> SAO2 <- CCHL is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure HYP -> LVV <- PMB is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure PRSS -> ACO2 <- VALV is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CCHL -> SAO2 <- SHNT is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure HRSA -> HRBP <- ERLO is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure MVS -> PRSS <- VTUB is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure MVS -> DISC <- VTUB is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure MVS -> PRSS <- KINK is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure TPR -> CCHL <- ACO2 is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CVP -> LVV <- PMB is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure INT -> PRSS <- KINK is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure PMB -> SHNT <- INT is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
names(comparisons.2) <- as.character(alpha.values.2)

comparisons.2
```

```
## $`0.3`
## $`0.3`$tp
## [1] 34
##
## $`0.3`$fp
## [1] 13
##
## $`0.3`$fn
## [1] 12
##
##
## $`0.4`
## $`0.4`$tp
## [1] 39
##
## $`0.4`$fp
## [1] 9
##
## $`0.4`$fn
```

```
## [1] 7
##
##
## $`0.5`
## $`0.5`$tp
## [1] 37
##
## $`0.5`$fp
## [1] 15
##
## $`0.5`$fn
## [1] 9
##
##
## $`0.75`
## $`0.75`$tp
## [1] 30
##
## $`0.75`$fp
## [1] 42
##
## $`0.75`$fn
## [1] 16
```

El rendimiento decae por encima de $\alpha = 0.2$. La red ajustada, comparada con la original, sería:

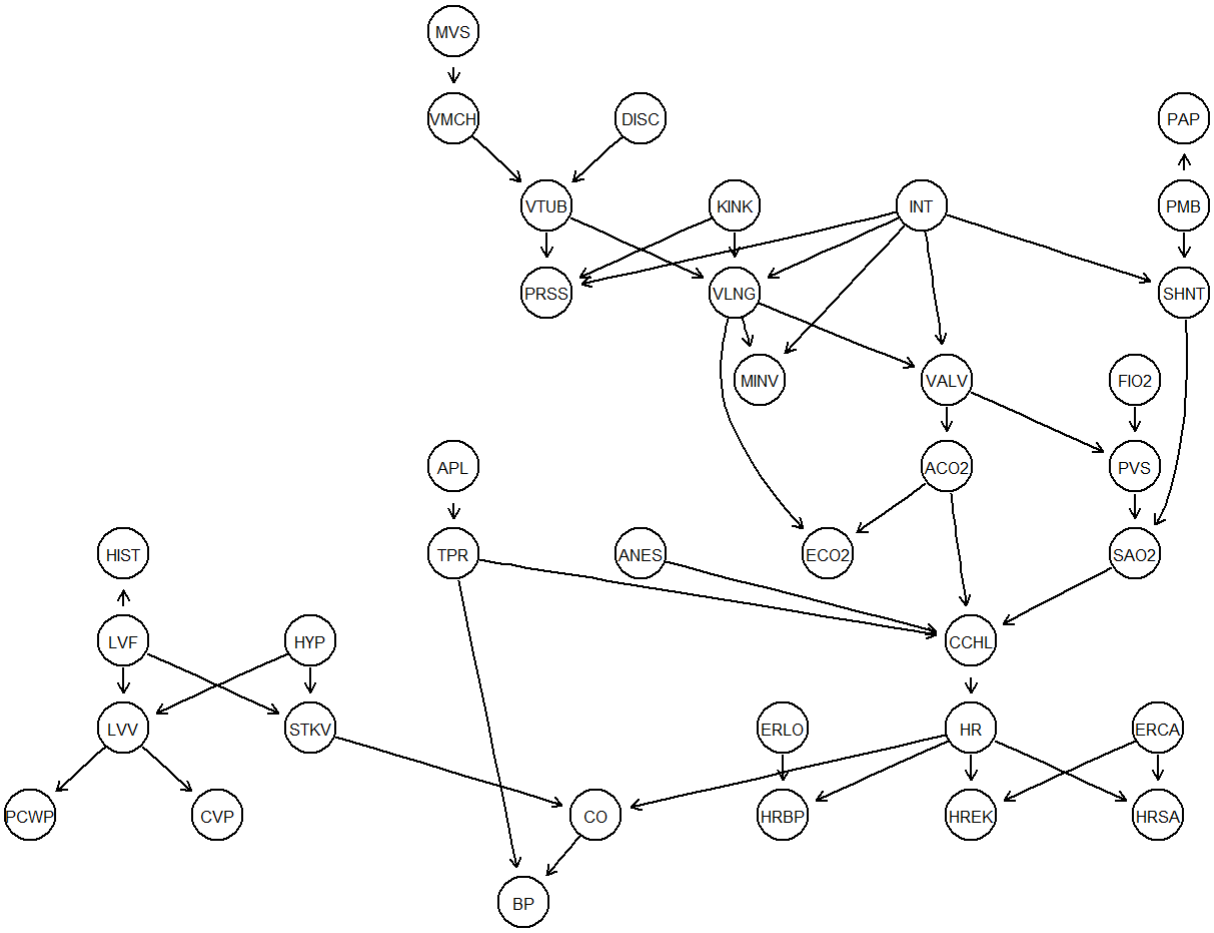
```
alarm.model.pc <- pc.stable(alarm, test="mc-x2", alpha=0.2)
```

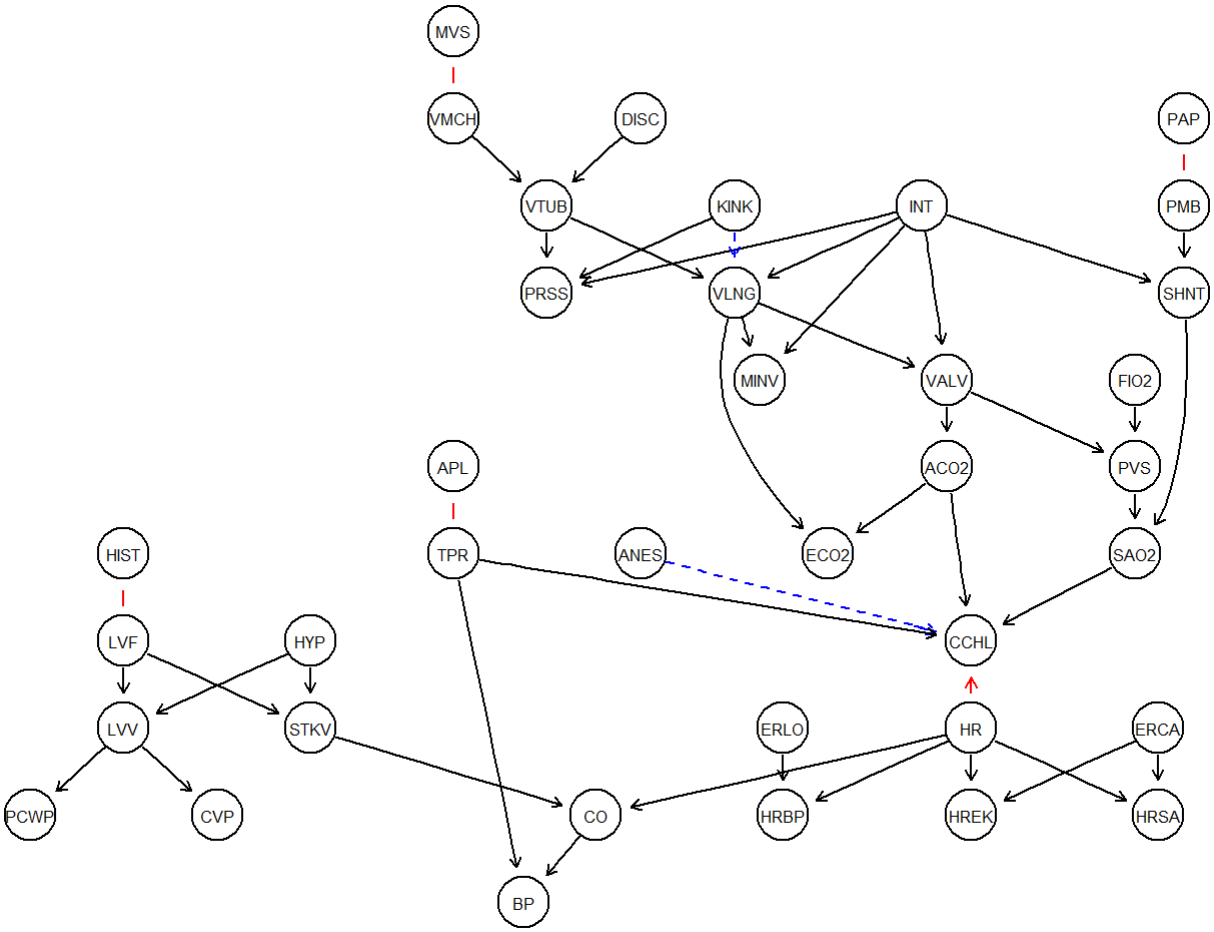
```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure ECO2 -> ACO2 <- CCHL is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CCHL -> ACO2 <- VALV is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
## Warning in vstruct.apply(arcs = arcs, vs = vs, nodes = nodes, debug = debug):
## vstructure CCHL -> SAO2 <- SHNT is not applicable, because one or both arcs are
## oriented in the opposite direction.
```

```
graphviz.compare(alarm.bn, alarm.model.pc)
```





Tenemos otras alternativas para el ajuste de la red, basadas en puntuación: algoritmo Hill-Climbing y búsqueda tabú:

```
alarm.model.hc <- hc(alarm)
alarm.model.tabu <- tabu(alarm)

compare(alarm.bn, alarm.model.hc)
```

```
## $tp
## [1] 22
##
## $fp
## [1] 31
##
## $fn
## [1] 24
```

```
compare(alarm.bn, alarm.model.tabu)
```

```
## $tp
## [1] 26
##
## $fp
## [1] 25
##
## $fn
## [1] 20
```

En principio, no mejoran los resultados que obtuvimos con algoritmos basados en restricciones. Alterando el valor de tabu:

```
taboos <- c(5,10,20,50,100,200)

comparisons.3 <- lapply(taboos, function(tabu){
  model <- tabu(alarm, test="mc-x2", tabu=tabu)
  comparison <- compare(alarm.bn, model)
  return(comparison)
  })
```

```
## Warning in check.unused.args(extra, c(method.extra.args[[heuristic]],
## score.extra.args[[score]])): unused argument(s): 'test'.

## Warning in check.unused.args(extra, c(method.extra.args[[heuristic]],
## score.extra.args[[score]])): unused argument(s): 'test'.

## Warning in check.unused.args(extra, c(method.extra.args[[heuristic]],
## score.extra.args[[score]])): unused argument(s): 'test'.

## Warning in check.unused.args(extra, c(method.extra.args[[heuristic]],
## score.extra.args[[score]])): unused argument(s): 'test'.

## Warning in check.unused.args(extra, c(method.extra.args[[heuristic]],
## score.extra.args[[score]])): unused argument(s): 'test'.

## Warning in check.unused.args(extra, c(method.extra.args[[heuristic]],
## score.extra.args[[score]])): unused argument(s): 'test'.
```

```
names(comparisons.3) <- as.character(taboos)

comparisons.3
```

```
## $`5`
## $`5`$tp
## [1] 26
##
## $`5`$fp
```

```
## [1] 25
##
## $`5`$fn
## [1] 20
##
##
## $`10`
## $`10`$tp
## [1] 26
##
## $`10`$fp
## [1] 25
##
## $`10`$fn
## [1] 20
##
##
## $`20`
## $`20`$tp
## [1] 26
##
## $`20`$fp
## [1] 25
##
## $`20`$fn
## [1] 20
##
##
## $`50`
## $`50`$tp
## [1] 27
##
## $`50`$fp
## [1] 23
##
## $`50`$fn
## [1] 19
##
##
## $`100`
## $`100`$tp
## [1] 27
##
## $`100`$fp
## [1] 23
##
## $`100`$fn
## [1] 19
##
##
## $`200`
## $`200`$tp
## [1] 27
##
## $`200`$fp
## [1] 23
##
## $`200`$fn
## [1] 19
```

En ningún caso parece mejorar el valor explicativo, por lo que el número de búsquedas tabú no parece un parámetro significativo para la precisión del modelo.

AJUSTE DE PARÁMETROS

Comenzamos ajustando las parámetros

```
alarm.fitted <- bn.fit(alarm.bn, alarm)
```

Tenemos una red con distribución multinomial; según la naturaleza de las variables, pueden adoptar 2 (TRUE, FALSE), 3 (HIGH, LOW, NORMAL) o 4 (HIGH,LOW,NORMAL,ZERO) valores. Por ello, las tablas de probabilidad condicional son potencialmente más complejas que en el caso de ASIA, en que teníamos variables binomiales.

Tomemos el ejemplo de la variable ECO2. Presenta 4 posibles valores (HIGH,LOW,NORMAL,ZERO), y tiene dos nodos parentales; las variables VLNG, con los mismos 4 estados, y ACO2, con 3 (HIGH,LOW,NORMAL). Los parámetros para ECO2 son:

```
alarm.fitted$ECO2
```

```
##
##   Parameters of node ECO2 (multinomial distribution)
##
## Conditional probability table:
##
## , , VLNG = HIGH
##
##         ACO2
## ECO2            HIGH         LOW      NORMAL
##   HIGH   0.875000000 0.967078189 0.931034483
##   LOW    0.000000000 0.002057613 0.017241379
##   NORMAL 0.125000000 0.024691358 0.034482759
##   ZERO   0.000000000 0.006172840 0.017241379
##
## , , VLNG = LOW
##
##         ACO2
## ECO2            HIGH         LOW      NORMAL
##   HIGH   0.000000000 0.009240246 0.005012531
##   LOW    0.000000000 0.969199179 0.012531328
##   NORMAL 0.977777778 0.008983573 0.005012531
##   ZERO   0.022222222 0.012577002 0.977443609
##
## , , VLNG = NORMAL
##
##         ACO2
## ECO2     HIGH         LOW      NORMAL
##   HIGH        0.000000000 0.005780347
##   LOW         0.000000000 0.011560694
##   NORMAL      0.983606557 0.965317919
##   ZERO        0.016393443 0.017341040
##
## , , VLNG = ZERO
##
##         ACO2
## ECO2            HIGH         LOW      NORMAL
##   HIGH   0.000000000 0.009163534 0.010188487
##   LOW    0.944055944 0.008771930 0.962812022
##   NORMAL 0.027972028 0.010886591 0.013245033
##   ZERO   0.027972028 0.971177945 0.013754457
```

Es decir, fijamos la tabla para cada uno de los 4 estados del nodo parental VLNG, y presentamos las distribuciones de cada uno de los 4 estados de ECO2 en función de cada uno de los 3 estados de ACO2.

SACHS

En el trabajo de Sachs et al. (2005), se medían simultáneamente 11 proteínas y fosfolípidos fosforilados derivados de miles de células del sistema inmune primario, bajo condiciones ambientales inespecíficas y ciertas intervenciones a nivel molecular. En el primer caso, aseguramos que las rutas de señalización más importantes están activas, mientras que en el segundo, realizamos inferencias causales elucidando las direcciones de los arcos mediante señales de estímulo o inhibición.

El análisis realizado en Sachs et al. (2005) puede esquematizarse en:

1. Los valores atípicos fueron eliminados y los datos fueron discretizados empleando el enfoque descrito en Hartemink (2001), ya que no era probable que el conjunto de datos original se ajustara a las premisas de distribución que requieren las redes bayesianas gaussianas.
2. El ajuste de la estructura se repitió exhaustivamente. De este modo, se exploró un gran número de estructuras de red, de modo que se redujo, para pasos posteriores de inferencia, la influencia de redes óptimas desde el enfoque local pero subóptimas globalmente.
3. Las redes ajustadas en el paso previo se pusieron en común en una red consenso para dar lugar a un modelo más robusto. Esta práctica, conocida como model averaging (Claeskens and Hjort, 2009), se aplica para obtener un resultado más predictivo del que daría una sola red de alta puntuación. La estructura de red consensuada se sintetizó incluyendo los arcos presentes en, al menos, el 85% de las redes. Esta proporción mide el peso de cada arco y permite extraer u significación dado el umbral especificado.
4. La validez de la red consensuada se evaluó tomando como referencia rutas y conexiones bien caracterizadas en la literatura científica.

Todos estos pasos pueden llevarse a cabo usando el paquete bnlearn. Por el momento, consideraremos solamente los datos manipulados con intervenciones generales.

Recogemos los datos observacionales:

```
sachs.obs <- read.csv(file="sachs.data.txt", sep="\t")
```

CONSTRUCCIÓN DE LA RED

El trabajo de Sachs contenía el mapa genético que convencionalmente se aceptaba para la ruta de fosforilación:

```
library(bnlearn)

sachs.previous.bn <- empty.graph(names(sachs.obs))

arcs(sachs.previous.bn, check.cycles=FALSE) <- matrix(c("PKC", "Jnk",
                                                       "PKC", "P38",
                                                       "PKC", "Raf",
                                                       "PKC", "Mek",
                                                       "PKA", "Jnk",
                                                       "PKA", "P38",
                                                       "PKA", "Akt",
                                                       "PKA", "Erk",
                                                       "PKA", "Mek",
                                                       "PKA", "Raf",
                                                       "Raf", "Mek",
                                                       "Mek", "Erk",
                                                       "Plcg", "PKC",
                                                       "Plcg", "PIP2",
                                                       "PIP3", "Plcg",
                                                       "PIP3", "PIP2",
```

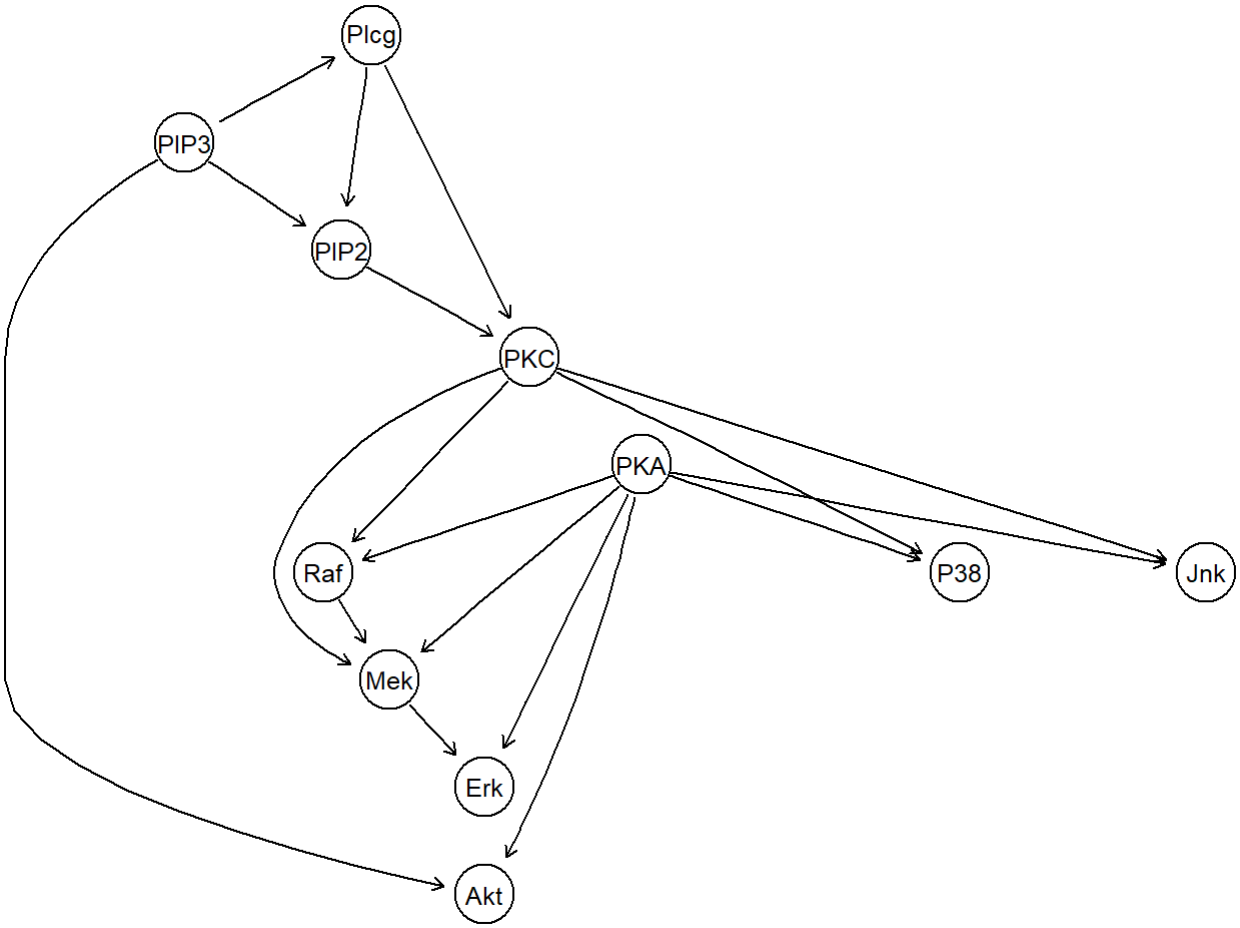
```
                                                  "PIP3", "Akt",
                                                  "PIP2", "PKC"),
                                                ncol = 2,
                                                byrow = TRUE,
                                                dimnames = list(c(),
                                                                c("from", "to")))
```

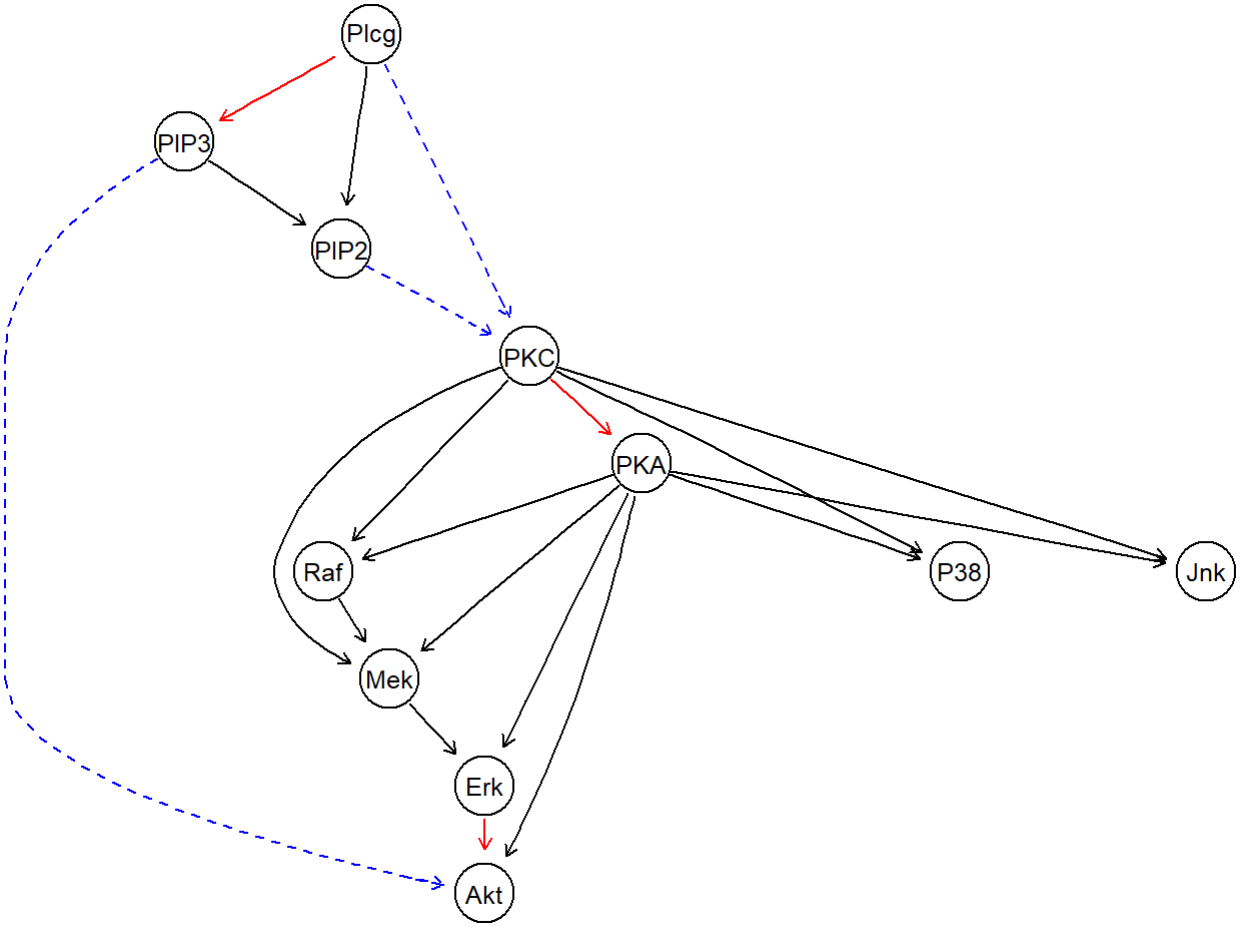
Por otro lado, mostraron la red que obtuvieron por inferencia a partir de los datos de expresión génica:

```
sachs.new.bn <- empty.graph(names(sachs.obs))

arcs(sachs.new.bn, check.cycles=FALSE) <- matrix(c("PKC", "Jnk",
                                                   "PKC", "P38",
                                                   "PKC", "Raf",
                                                   "PKC", "Mek",
                                                   "PKC", "PKA",
                                                   "PKA", "Jnk",
                                                   "PKA", "P38",
                                                   "PKA", "Akt",
                                                   "PKA", "Erk",
                                                   "PKA", "Mek",
                                                   "PKA", "Raf",
                                                   "Raf", "Mek",
                                                   "Mek", "Erk",
                                                   "Erk", "Akt",
                                                   "Plcg", "PIP2",
                                                   "Plcg", "PIP3",
                                                   "PIP3", "PIP2"),
                                                 ncol = 2,
                                                 byrow = TRUE,
                                                 dimnames = list(c(), c("from", "to")))
```

Aquí pueden apreciarse las diferencias:

```
graphviz.compare(sachs.previous.bn, sachs.new.bn)
```





La diferencia más notable es que, según la inferencia realizada por Sachs et al, el subconjunto [PIP2, PIP3, Plcg] queda desvinculado del resto de la red. Encontramos, además, dos arcos espúreos y una inversión.

La densidad de la red a priori:

```
sachs.previous.arcs <- narcs(sachs.previous.bn)
sachs.previous.nodes <- nnodes(sachs.previous.bn)
```

```
sachs.previous.arcs
```

```
## [1] 18
```

```
sachs.previous.nodes
```

```
## [1] 11
```

```
sachs.previous.density <- sachs.previous.arcs/(sachs.previous.nodes*(sachs.previous.nodes-1))
sachs.previous.density
```

```
## [1] 0.1636364
```

```
sachs.new.arcs <- narcs(sachs.new.bn)
sachs.new.nodes <- nnodes(sachs.new.bn)

sachs.new.arcs
```

```
## [1] 17
```

```
sachs.new.nodes
```

```
## [1] 11
```

```
sachs.new.density <- sachs.new.arcs/(sachs.new.nodes*(sachs.new.nodes-1))
sachs.new.density
```

```
## [1] 0.1545455
```

PREPROCESADO

Utilizamos la deduplicación únicamente de forma descriptiva para identificar genes altamente correlacionados con otros:

```
# definir los umbrales de correlación
thresholds <- c(0.98, 0.95, 0.9, 0.8, 0.6)

# para cada umbral
dedup.variables <- lapply(thresholds, function(threshold){
  # identificar las variables excluidas del conjunto al superar la correlación
  dedup.vars <- names(sachs.obs)[which(!names(sachs.obs) %in% names(dedup(sachs.obs, threshold
 = threshold)))]})

# etiquetar cada resultado con el valor de la variable
names(dedup.variables) <- as.character(thresholds)

dedup.variables
```

```
## $`0.98`
## [1] "Akt"
##
## $`0.95`
```

```
## [1] "Akt"
##
## $`0.9`
## [1] "Akt"
##
## $`0.8`
## [1] "Akt"
##
## $`0.6`
## [1] "Mek" "Akt" "P38"
```

Como podemos observar, la proteína quinasa Akt presenta un enorme grado de correlación, superior al 98%, con alguna de las otras variables. Las variables candidatas para dicha correlación lineal de Pearson serían, de acuerdo a los mapas de dependencia construidos anteriormente, Erk, PKA y PIP3:

```
cor.test(sachs.obs$Akt, sachs.obs$Erk)
```

```
##
##  Pearson's product-moment correlation
##
## data:  sachs.obs$Akt and sachs.obs$Erk
## t = 224.54, df = 851, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9904725 0.9927107
## sample estimates:
##       cor
## 0.9916661
```

```
cor.test(sachs.obs$Akt, sachs.obs$PKA)
```

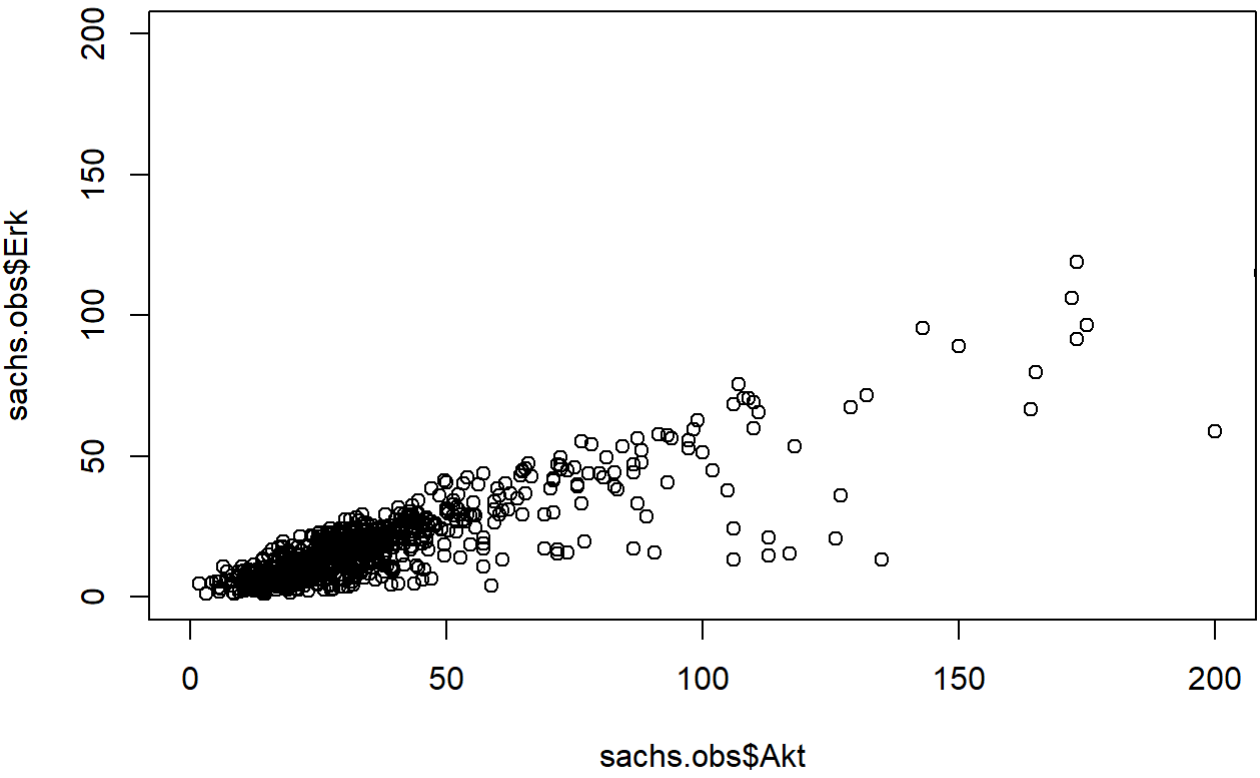
```
##
##  Pearson's product-moment correlation
##
## data:  sachs.obs$Akt and sachs.obs$PKA
## t = 14.108, df = 851, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3793438 0.4882389
## sample estimates:
##       cor
## 0.4353826
```

```
cor.test(sachs.obs$Akt, sachs.obs$PIP3)
```

```
##
##  Pearson's product-moment correlation
##
## data:  sachs.obs$Akt and sachs.obs$PIP3
## t = -0.99404, df = 851, p-value = 0.3205
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.10094972  0.03314552
## sample estimates:
##         cor
## -0.03405537
```

La variable de alta correlación con Akt es ERK, lo que podría explicar la presencia de un arco espúreo entre ambas en el trabajo de Sachs, de acuerdo a la bibliografía previa. En cualquier caso, no parece pertinente desechar variables de nuestro conjunto de datos, ya que sabemos que todas las fosfoproteínas y fosfolípidos están englobadas en el ciclo de señalización celular.

```
plot(sachs.obs$Akt, sachs.obs$Erk, xlim=c(0,200), ylim=c(0,200))
```



Al margen de esta correlación, no encontramos ninguna otra pareja de variables con valores por encima del 70%.

Utilizaremos los métodos de discretización disponibles para gaussianas multivariante. Estos son: por cuantiles, por intervalos de valores, y por el método de información mutua de Hartemink. Mantendremos todos los parámetros por defecto.

```
dsachs.q <- discretize(sachs.obs, method="quantile", breaks=3)

dsachs.i <- discretize(sachs.obs, method="interval", breaks=3)

dsachs.h <- discretize(sachs.obs, method = "hartemink",
                    breaks = 3, ibreaks = 60,
                    idisc = "quantile")
```

AJUSTE DE ESTRUCTURA

Familia de algoritmos basados en restricciones:

```
sachs.bn.pc <- pc.stable(sachs.obs)
sachs.bn.gs <- gs(sachs.obs)
sachs.bn.iamb <- iamb(sachs.obs)
sachs.bn.mmpc <- mmpc(sachs.obs)
```

Familia de algoritmos basados en puntuaciones:

```
sachs.bn.hc <- hc(sachs.obs)
sachs.bn.tabu <- tabu(sachs.obs)
```

Familia de algoritmos híbridos:

```
sachs.bn.mmhc <- mmhc(sachs.obs)
```

Comparativas entre redes aprendidas por algoritmos basados en restricciones y la red inferida en Sachs et al:

```
compare(sachs.new.bn, sachs.bn.pc)
```

```
## $tp
## [1] 0
##
## $fp
## [1] 8
##
## $fn
## [1] 17
```

```
compare(sachs.new.bn, sachs.bn.gs)
```

```
## $tp
## [1] 0
##
## $fp
## [1] 8
##
## $fn
## [1] 17
```

```
compare(sachs.new.bn, sachs.bn.iamb)
```

```
## $tp
## [1] 0
##
## $fp
## [1] 8
##
## $fn
## [1] 17
```

```
compare(sachs.new.bn, sachs.bn.mmpc)
```

```
## $tp
## [1] 0
##
## $fp
## [1] 8
##
## $fn
## [1] 17
```

Los algoritmos basados en restricciones no devuelven ajustes apropiados (ningún verdadero positivo).

Comparativas entre redes aprendidas por algoritmos basados en puntuaciones y la red inferida en Sachs et al:

```
compare(sachs.new.bn, sachs.bn.hc)
```

```
## $tp
## [1] 4
##
## $fp
## [1] 5
##
## $fn
## [1] 13
```

```
compare(sachs.new.bn, sachs.bn.tabu)
```

```
## $tp
## [1] 3
##
## $fp
## [1] 6
##
## $fn
## [1] 14
```

El aprendizaje mejora sustancialmente, pero aún está lejos de resultar aceptable. El método Hill-Climbing mejora en una unidad a la búsqueda tabú en los tres parámetros (verdaderos positivos, falsos positivos y falsos negativos).

Comparativas entre redes aprendidas por algoritmos basados en restricciones y la red inferida en Sachs et al:

```
compare(sachs.new.bn, sachs.bn.mmhc)
```

```
## $tp
## [1] 4
##
## $fp
## [1] 4
##
## $fn
## [1] 13
```

Min-Max Hill-Climbing elimina un arco espúreo respecto a Hill-Climbing. Sin embargo, la diferencia es tan escasa que no parece conveniente descartar el algoritmo Hill-Climbing para el aprendizaje de la red.

A continuación, aplicamos los algoritmos HC y MMHC a los distintos tipos de datos discretizados que generamos anteriormente.

```
dsachs.q.bn.hc <- hc(dsachs.q)
dsachs.i.bn.hc <- hc(dsachs.i)
```

```
## Warning in check.data(x): variable PIP3 has levels that are not observed in the
## data.
```

```
## Warning in check.data(x): variable Erk has levels that are not observed in the
## data.
```

```
## Warning in check.data(x): variable Akt has levels that are not observed in the
## data.
```

```
dsachs.h.bn.hc <- hc(dsachs.h)

dsachs.q.bn.mmhc <- mmhc(dsachs.q)
dsachs.i.bn.mmhc <- mmhc(dsachs.i)
```

```
## Warning in check.data(x, allow.missing = TRUE): variable PIP3 has levels that
## are not observed in the data.
```

```
## Warning in check.data(x, allow.missing = TRUE): variable Erk has levels that are
## not observed in the data.
```

```
## Warning in check.data(x, allow.missing = TRUE): variable Akt has levels that are
## not observed in the data.
```

```
## Warning in check.data(x): variable PIP3 has levels that are not observed in the
## data.
```

```
## Warning in check.data(x): variable Erk has levels that are not observed in the
## data.
```

```
## Warning in check.data(x): variable Akt has levels that are not observed in the
## data.
```

```
dsachs.h.bn.mmhc <- mmhc(dsachs.h)
```

Comparamos las redes aprendidas con la red inferida en el trabajo de Sachs et al.

```
compare(sachs.new.bn, dsachs.q.bn.hc)
```

```
## $tp
## [1] 6
##
## $fp
## [1] 2
##
## $fn
## [1] 11
```

```
compare(sachs.new.bn, dsachs.i.bn.hc)
```

```
## $tp
## [1] 2
##
## $fp
## [1] 0
##
## $fn
## [1] 15
```

```
compare(sachs.new.bn, dsachs.h.bn.hc)
```

```
## $tp
## [1] 6
##
## $fp
## [1] 3
##
## $fn
## [1] 11
```

```
compare(sachs.new.bn, dsachs.q.bn.mmhc)
```

```
## $tp
## [1] 6
##
## $fp
## [1] 2
##
## $fn
## [1] 11
```

```
compare(sachs.new.bn, dsachs.i.bn.mmhc)
```

```
## $tp
## [1] 2
##
## $fp
## [1] 0
##
## $fn
## [1] 15
```

```
compare(sachs.new.bn, dsachs.h.bn.mmhc)
```

```
## $tp
## [1] 6
##
## $fp
## [1] 3
##
## $fn
## [1] 11
```

El método de discretización por cuantiles resulta más efectivo a la hora de ajustar la red, y devuelve el mismo resultado tanto por el algoritmo Hill-Climbing como por Min-Max Hill Climbing, por lo que optamos por el algoritmo HC por su menor complejidad.

A continuación, proponemos una serie de números de intervalos para observar en cuál de ellos se optimiza el aprendizaje de la red.

```
# definir el número de intervalos
n.breaks <- c(2, 3, 5, 10, 20)

# para cada número de intervalos
discrete.results <- lapply(n.breaks, function(breaks){
  # ajustar la red por el método HC con los datos discretizados por el número de intervalos imp
uesto
  discrete.network <- hc(discretize(sachs.obs, method="quantile", breaks=breaks))
```

```
  # comparar la red aprendida con la red inferida en el trabajo de Sachs et al
  discrete.result <- compare(sachs.new.bn, discrete.network)
  })

# etiquetar cada resultado con el valor de la variable
names(discrete.results) <- as.character(n.breaks)

discrete.results
```

```
## $`2`
## $`2`$tp
## [1] 6
##
## $`2`$fp
## [1] 2
##
## $`2`$fn
## [1] 11
##
##
## $`3`
## $`3`$tp
## [1] 6
##
## $`3`$fp
## [1] 2
##
## $`3`$fn
## [1] 11
##
##
## $`5`
## $`5`$tp
## [1] 4
##
## $`5`$fp
## [1] 2
##
## $`5`$fn
## [1] 13
##
##
## $`10`
## $`10`$tp
## [1] 4
##
## $`10`$fp
## [1] 0
##
## $`10`$fn
## [1] 13
##
##
## $`20`
## $`20`$tp
## [1] 0
##
## $`20`$fp
## [1] 0
##
```

```
## $`20`$fn
## [1] 17
```

2 ó 3 intervalos da el mejor resultado.

Para que nuestra red refleje de forma más fiel la estructura inferida por Sachs et al, procedemos a realizar un remuestreo bootstrap sobre el algoritmo HC con datos discretizados por cuantiles. La muestra será de tamaño 500.

```
set.seed(68703258)

dsachs.boot <- boot.strength(data=dsachs.q, R=500, algorithm = "hc")
```

Para generar una red consenso a partir del remuestreo bootstrap, incluimos todos los arcos que superen un determinado umbral de presencia (en proporción de redes que presentan dicho arco), en su dirección mayoritaria.

Nos basaremos en la bibliografía existente sobre construcción de redes de dependencia en genes (referencias en el TFM), que típicamente consideran una proporción de presencia del 80-85% para incluir un arco en la red consenso. Impondremos un umbral de 0.85:

```
avg.dsachs.bn <- averaged.network(dsachs.boot, threshold = 0.85)

compare(sachs.new.bn, avg.dsachs.bn)
```

```
## $tp
## [1] 7
##
## $fp
## [1] 1
##
## $fn
## [1] 10
```

FUSIÓN DE REDES

Aún quedan muchos arcos de la estructura real no detectados al tratar con los datos observacionales. Un método para tratar de extraer estos arcos es sintetizar la información de los datos observacionales con las redes que obtengamos por metodología bootstrap para cada uno de los 14 conjuntos de datos en que se realizaron perturbaciones sobre la expresión de determinados genes.

GBNFUSESVOTE

```
library(readxl)
```

```
## Warning: package 'readxl' was built under R version 3.6.3
```

```
set.seed(68703258)

# recoger los ficheros con datos de expresión de genes
files <- list.files(path="C:/Users/carlm/OneDrive/Escritorio/Biotecnología/Máster/Trabajo de Fi
n de Máster/Documentos/sachs datasets/Data Files", pattern="*.xls", full.names=TRUE, recursive=
FALSE)

# para cada fichero
sachs.networks <- lapply(files, function(file) {
  # leer los datos
  sachs.file <- read_excel(file) # load file
  # renombrar las variables a los nombres en el fichero de datos observacionales
  colnames(sachs.file) <- colnames(sachs.obs)
  # generar una muestra bootstrap de tamaño 500 de la red ajustada por el método HC
```

```
  sachs.boot <- boot.strength(data=sachs.file, R=500,
                              algorithm = "hc")
  # consensuar el bootstrap con un umbral de aparición de 0.85
  sachs.network <- averaged.network(sachs.boot, threshold = 0.85)
})
```

```
## Warning in averaged.network.backend(strength = strength, nodes = nodes, : arc
## PIP2 -> PIP3 would introduce cycles in the graph, ignoring.
```

Creamos la matriz de votaciones:

```
nodes.sachs <- colnames(sachs.obs)
links.sachs <- matrix(0, nrow=length(nodes.sachs), ncol=length(nodes.sachs))

dimnames(links.sachs) <- list(nodes.sachs, nodes.sachs)

links.sachs
```

```
##      Raf Mek Plcg PIP2 PIP3 Erk Akt PKA PKC P38 Jnk
## Raf    0   0    0    0    0   0   0   0   0   0   0
## Mek    0   0    0    0    0   0   0   0   0   0   0
## Plcg   0   0    0    0    0   0   0   0   0   0   0
## PIP2   0   0    0    0    0   0   0   0   0   0   0
## PIP3   0   0    0    0    0   0   0   0   0   0   0
## Erk    0   0    0    0    0   0   0   0   0   0   0
## Akt    0   0    0    0    0   0   0   0   0   0   0
## PKA    0   0    0    0    0   0   0   0   0   0   0
## PKC    0   0    0    0    0   0   0   0   0   0   0
## P38    0   0    0    0    0   0   0   0   0   0   0
## Jnk    0   0    0    0    0   0   0   0   0   0   0
```

Generamos el número de votaciones, esto es, el número de apariciones de cada arco en el conjunto de redes.

```
for (index in 1:(length(sachs.networks)-1)){
  sachs.network <- sachs.networks[[index]]
  arcs.network <- arcs(sachs.network)
  for (arc.index in 1:ncol(arcs.network)){
    arc <- arcs.network[arc.index,]
    links.sachs[arc[1],arc[2]] <- links.sachs[arc[1],arc[2]] + 1
  }
}
```

```
links.sachs
```

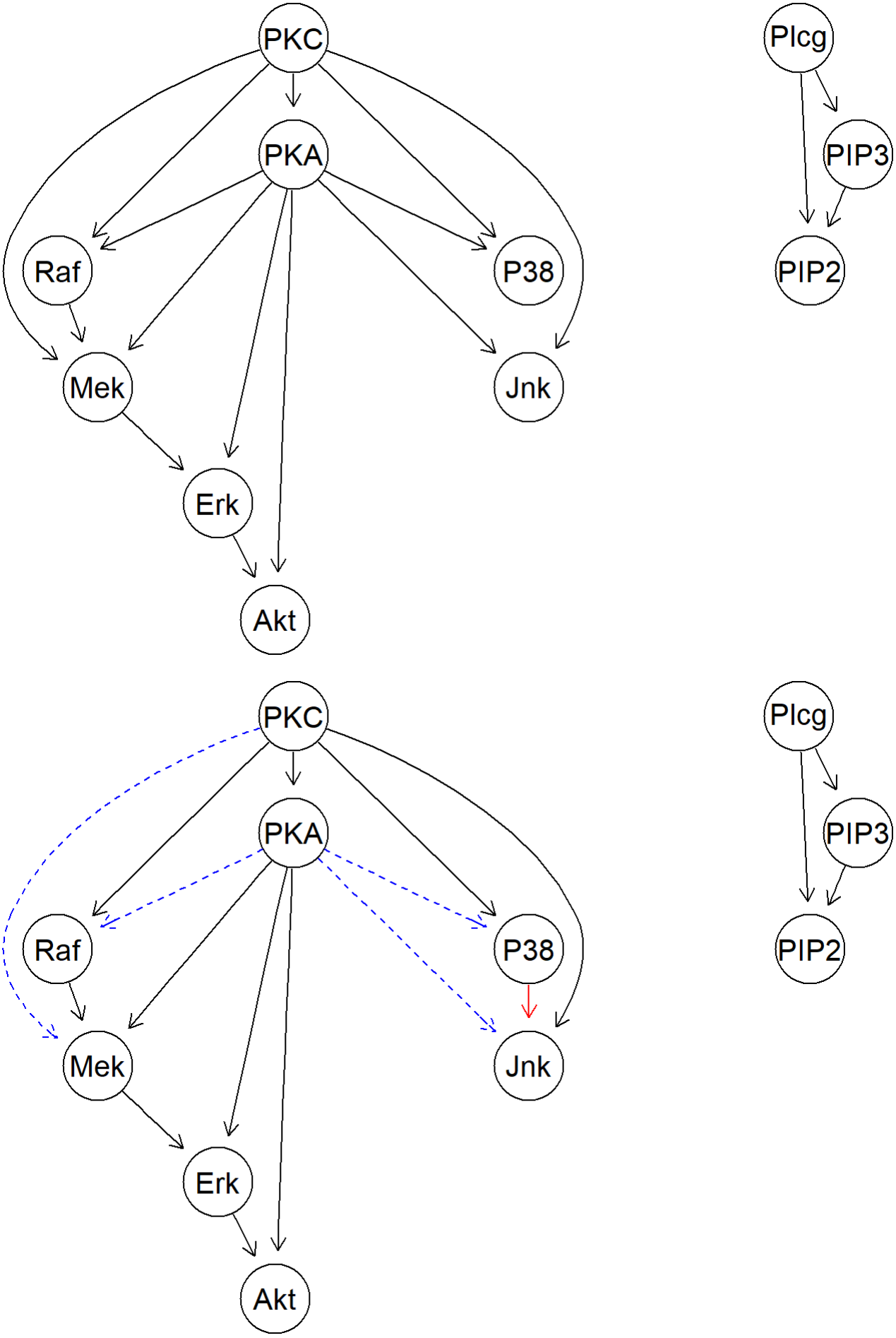
```
##      Raf Mek Plcg PIP2 PIP3 Erk Akt PKA PKC P38 Jnk
## Raf    0  13    0    0    0   0   0   0   0   0   0
## Mek    0   0    0    0    0   9   0   0   0   0   0
## Plcg   0   0    0   10   12   0   0   0   0   0   0
## PIP2   0   0    0    0    5   0   0   0   0   0   0
## PIP3   0   0    1    8    0   0   0   0   0   0   0
## Erk    0   0    0    0    0   0  13   0   0   0   0
## Akt    0   0    0    0    0   0   0   0   0   0   0
## PKA    0   9    0    0    0  10   8   0   0   0   0
## PKC   11   0    0    0    0   0   0   8   0  11   9
## P38    0   0    0    0    0   0   0   0   0   0  10
## Jnk    0   0    0    0    0   0   0   0   0   0   0
```

```
gbnfusesvote.bn <- empty.graph(names(sachs.obs))

for (rowname in rownames(links.sachs)){
  for (colname in colnames(links.sachs)){
    if (links.sachs[rowname,colname] > 7){
      gbnfusesvote.bn <- set.arc(gbnfusesvote.bn, from=rowname, to=colname)
    }
  }
}
```

```
compare(sachs.new.bn, gbnfusesvote.bn)
```

```
## $tp
## [1] 13
##
## $fp
## [1] 1
##
## $fn
## [1] 4
```

```
graphviz.compare(sachs.new.bn, gbnfusesvote.bn)
```



GBNFUSESINTER

```
sachs.params <- list()

for (index in 1:length(sachs.networks)){
  sachs.file <- read_excel(files[index])
  sachs.network <- sachs.networks[[index]]
  colnames(sachs.file) <- colnames(sachs.obs)
```

```
  sachs.params[[index]] <- bn.fit(sachs.network, sachs.file)
}
```

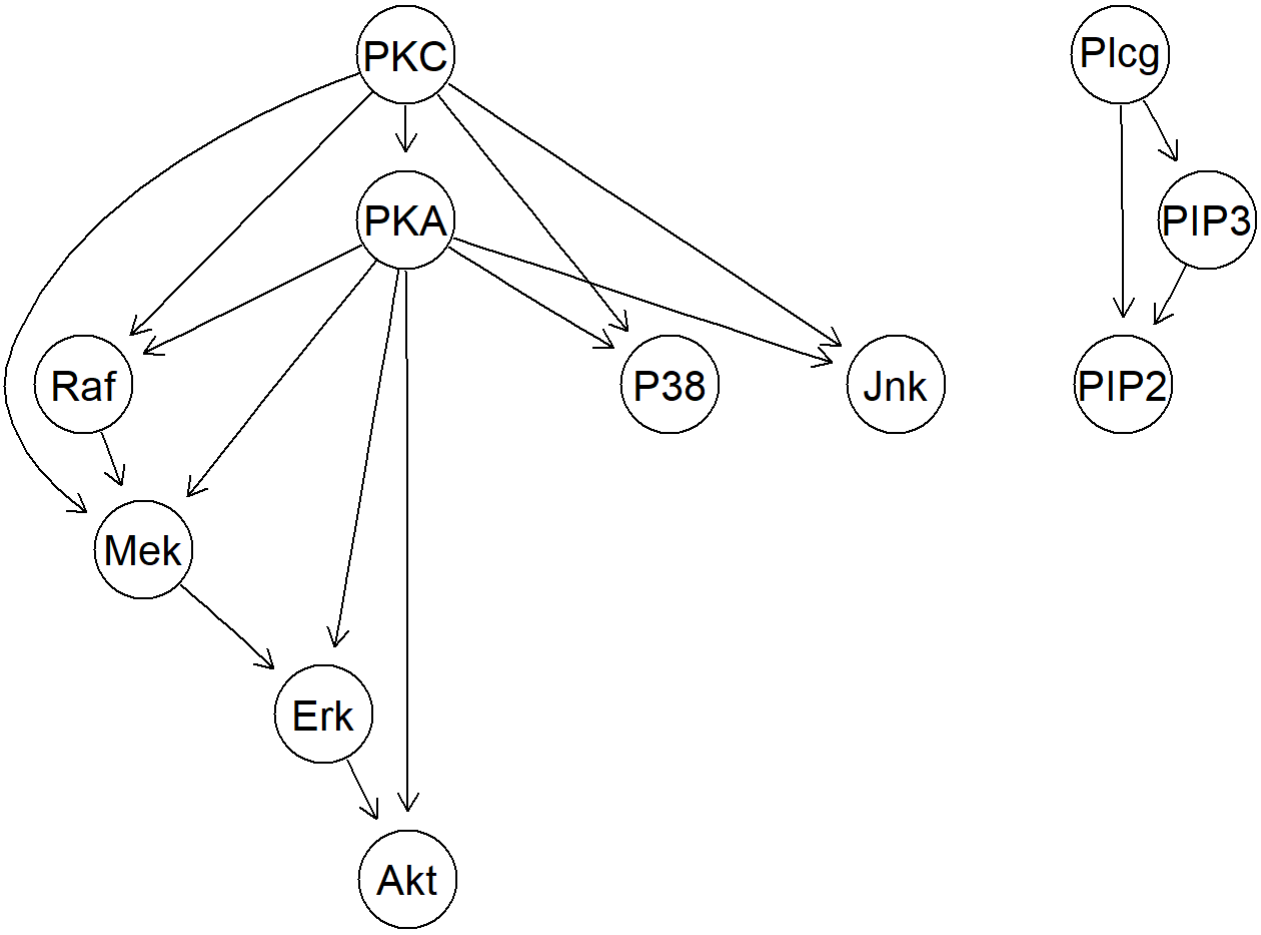
```
nodes.sachs <- colnames(sachs.obs)
regression.sachs <- matrix(0, nrow=length(nodes.sachs), ncol=length(nodes.sachs))

dimnames(regression.sachs) <- list(nodes.sachs, nodes.sachs)

for (index in 1:length(sachs.params)){
  nodes.names <- c()
  nodes.sd <- c()
  for (node in sachs.params[[index]]){
    node.name <- `node`$node
    nodes.names <- c(nodes.names, node.name)
    node.sd <- `node`$sd
    nodes.sd <- c(nodes.sd, node.sd)
    node.coefficients <- `node`$coefficients
  }
  for (node in sachs.params[[index]]){
    node.name <- `node`$node
    node.sd <- `node`$sd
    node.coefficients <- `node`$coefficients
    if (length(node.coefficients) > 1){
      for (coefficient.index in (2:length(node.coefficients))){
        coefficient <- node.coefficients[coefficient.index]
        parent.name <- names(coefficient)
        parent.sd <- nodes.sd[parent.name==nodes.names]
        normal.coefficient <- coefficient*parent.sd/node.sd
        if (normal.coefficient > regression.sachs[parent.name, node.name]){
          regression.sachs[parent.name, node.name] <- normal.coefficient
        }
      }
    }
  }
}
```

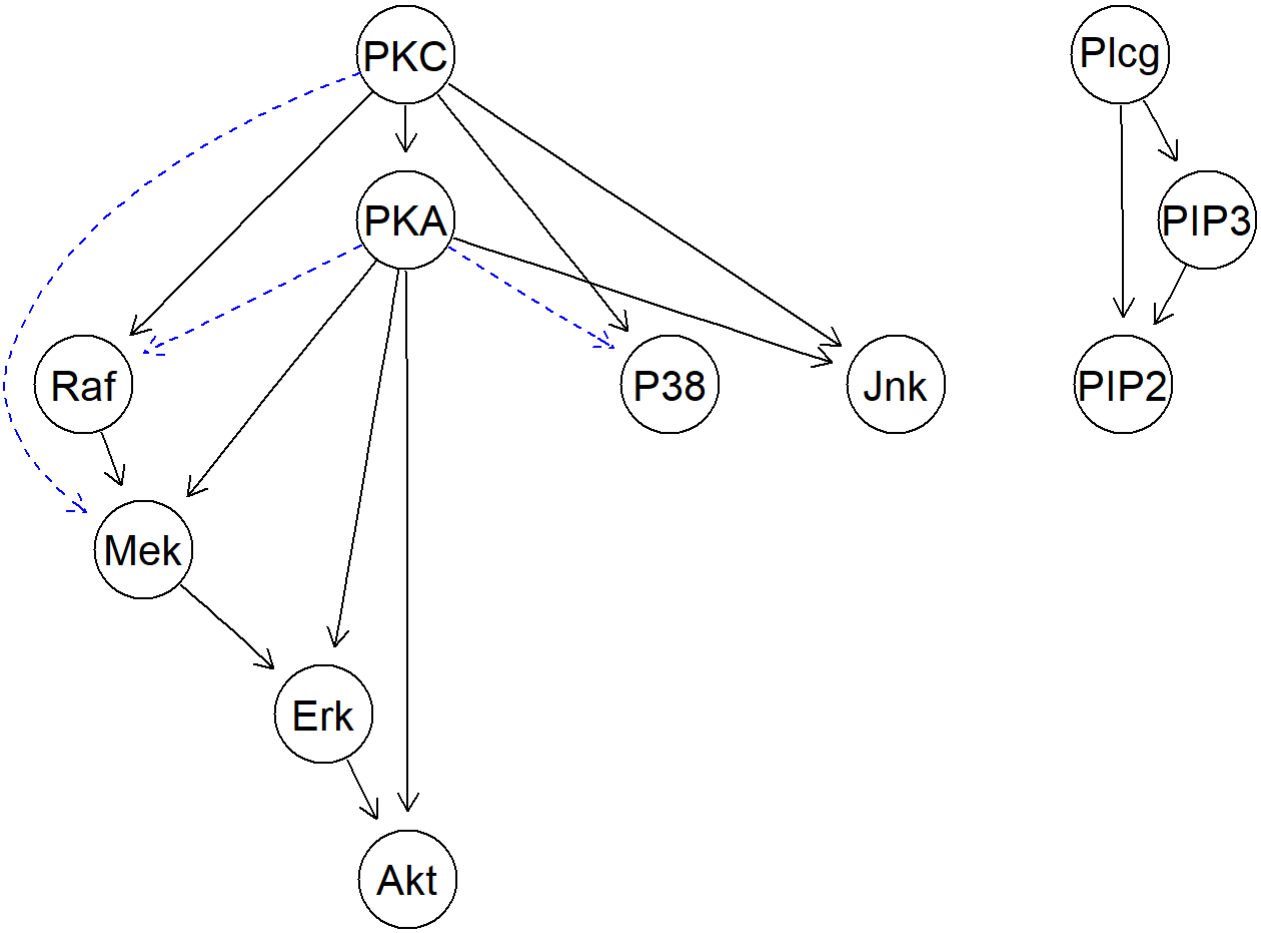
```
gbnfusesinter.bn <- empty.graph(names(sachs.obs))

for (rowname in rownames(links.sachs)){
  for (colname in colnames(links.sachs)){
    if (regression.sachs[rowname,colname] > 0.65){
      gbnfusesinter.bn <- set.arc(gbnfusesinter.bn, from=rowname, to=colname)
    }
  }
}
```

```
compare(sachs.new.bn, gbnfusesinter.bn)
```

```
## $tp
## [1] 14
##
## $fp
## [1] 0
##
## $fn
## [1] 3
```

```
graphviz.compare(sachs.new.bn, gbnfusesinter.bn)
```





AJUSTE DE PARÁMETROS

```
sachs.params[[1]]$Mek$residuals
```

```
##    [1]   0.297257535  -1.345934846  14.026168213  45.649598069   3.098804443
##    [6]  -5.198188561  13.971040794  -8.829799306   8.219180827  11.355038547
```

```
##  [11]   8.700617935 -14.926838784  -3.846468016  10.885762718   9.740316623
##  [16] -11.382673010  13.901324743  18.428821806   4.763879771  -8.214370574
##  [21] -10.298808550   2.191510127   7.400351350  18.402471773   6.402471773
##  [26]  -9.353095869   3.536956222 -15.085326603   4.974401194   4.456452162
##  [31]  -7.245228038   7.295444042  39.859279392 -11.368924479 -10.261630163
##  [36] -14.459949962  -6.401462143 -54.319637416 -13.347093622   0.771747601
##  [41] -16.546468016 -12.182499373   2.761492763   2.657158970 -17.373124979
##  [46] -12.438907115   9.203311873  19.608925988  16.549024554 -20.340720608
##  [51] -10.038907115  -0.350975446  -7.691074012  -4.773831787 -55.446694257
##  [56]  -2.385326603  -9.881126103 -11.749695124   6.962199570 -13.204955836
##  [61] -12.263843534   6.303138236  -3.381699618   0.594030427 -51.400981576
##  [66]  -2.141294123 -18.908316236  -6.498808550  10.888856534   1.937089515
##  [71]  -7.308756459  -0.297528227   5.438369837   0.326875021  11.514673397
##  [76]   6.098844787  20.421260905  -4.791074012   5.319273775  -4.437933722
##  [81]  22.966000193   3.686469526  -4.871311486   8.135409315 -20.625425169
##  [86]  -9.635680007  44.222714864   4.544517123 -10.973965079   6.062199570
##  [91]  18.886069648  -6.944590685   5.736249415 168.692644897  -3.635413422
##  [96]   1.040316623   2.546637546  -4.879445902  11.993763842  38.189296756
## [101] -12.655309240  -7.601462143   5.138369837 -29.843507493  -4.672284879
## [106]   4.128821806  -2.658136470   6.657292262   0.384082518 -13.378739095
## [111]  -6.961989696   0.451411561  28.669800816  -5.294167827  -7.081126103
## [116]  -6.119544468  69.621701127  -1.398808550  13.039916745  21.656492507
## [121]  -1.321664890  10.056759092 -10.141294123 -16.040853900 -18.709156337
## [126] -18.041827293  15.595044164   2.771040794   5.376921495  10.091510127
## [131]   7.020554098  -9.073965079   4.579175210 -10.426838784  -1.912690374
## [136]   1.411712874 -13.924451776 -15.475778572  -4.651108739   7.016753475
## [141] -12.585326603   1.496417435 -21.440147093  -4.773831787 -14.592047404
## [146] -70.133079018 -13.378739095   0.546637546   8.404059025  11.703138236
## [151] -15.882499373   6.426034921 -10.408889751   3.359279392  -8.546468016
## [156]  61.848265142  -1.898808550  18.224488013 -27.094567705 -11.924451776
## [161] -29.521491253  -8.119544468 -24.562163333   7.700218058   5.616753475
## [166]  -9.285326603   2.952099423  -5.914370574  -5.632186314  15.148584331
## [171]   9.280188947  -8.433999807  -6.140147093  27.946411305 -13.966537471
## [176]  -7.228252399  -5.942841030  -3.557296369  -5.006236158 -10.497528227
## [181]  -3.845228038  -6.860216548   3.076214687  11.936249415 -11.263310363
## [186]  37.717847901  -2.975511987 -13.941294123 -14.491074012   0.493763842
## [191]  22.393630549  -3.278298873  -1.428252399 -21.365523734   6.821701127
## [196]   7.324221428  26.169627179  -3.278298873   3.161092885  11.311979459
## [201]  -0.747881631  -2.446468016 -15.740720608  -0.924051898 -11.806369451
## [206]   2.628248291   4.141863530   8.137222808  16.232315499 -21.151108739
## [211]  -1.481699618  -1.381126103   3.637929615   5.214673397  -1.926838784
## [216] -16.219811053  -3.507783066  24.266573708  -6.026972076   0.727390828
## [221]   6.407952596  -8.394567705   0.537089515 -24.175778572  -9.660083255
## [226] -12.989220174  -0.263043778   6.425727991  -3.645228038 -10.978739095
## [231]  -3.609729851   6.417326990 -20.640147093  -7.451108739 -54.304168340
## [236]   4.693057034  -6.162870141   5.193057034 -15.243240908 -17.055309240
## [241] -21.840720608  -7.139613922 -15.363310363  -9.941827293 -11.673831787
## [246]  19.978508747   4.590270149   2.746904131 -25.295940975  -0.342574445
## [251]  15.624488013  -6.702035658   0.528688514  -2.887163077 -12.125425169
## [256]  -8.168924479 -17.460083255   3.472854287   8.685629426 -28.958629296
## [261] -30.287539974  -2.486606926   2.762906378   2.619354464   1.961092885
## [266]   2.397257535   3.320554098 -16.669191064 -10.709996437 -11.340720608
## [271]   6.432315499  11.988856534 -11.898808550  26.660386078  -6.424585068
## [276]  -4.686606926  -9.088420419   2.479175210  -5.797528227   6.670200694
## [281] -10.663310363  31.506232051  13.212771503   8.689256412  -8.844237282
## [286] -24.494567705   4.452651539   8.979175210  18.803311873  -1.973831787
## [291]   4.557425555  12.820687390  -5.794167827  -0.842841030   3.451678147
## [296]  12.906365344   8.836956222  -9.686166703 -15.378739095   7.125861284
## [301]  -7.074538594  -9.038907115   0.544499760 -11.426838784 -15.450401931
## [306]   1.542466156  -8.988420419 -14.768924479  11.705432295  -3.237093622
## [311]  -6.237093622 142.790791060   4.498537857 -11.343507493   7.508752351
## [316]  -2.584486503  -3.763043778 -10.408889751   9.666000193  22.240490260
```

```
## [321]   5.217726868  -0.231172577  -8.299781942 -20.451908495  -2.520824790
## [326] -46.051561221  -2.888420419  -3.226838784 -10.945187694   4.079175210
## [331]  10.056759092  -1.020691497  -4.022371698  -1.250975446   5.936249415
## [336] -16.868101742   1.964319993   7.667106879   4.097257535  -1.485326603
## [341]  -4.645228038 -30.129892254 -16.121491253  -6.708889751  -3.373831787
## [346]  -6.621664890   7.231075521  -1.343414545   1.618300382   0.478335110
## [351] -13.151108739 -10.638907115   5.145357224  -6.459949962  -7.208889751
## [356]   3.314140226  -9.815917482  -0.828959206   2.113833297   7.700218058
## [361]   3.135409315  -0.645187694  10.062906378   6.586469526  -6.136120229
## [366]  -8.915917482 -14.040147093   9.811006067  11.287309626   7.772987579
## [371] -12.510703244  11.023207691  -8.328252399   3.720687390  -1.658136470
## [376]  23.360386078  20.030808936  23.809592452  21.128248291   6.857292262
## [381] -27.080819173  -6.422371698  14.620554098 -15.604955836 -75.082152099
## [386] -12.095094745   0.235409315 -12.214637160  27.811579581   3.221527490
## [391]   4.325727991  -2.697528227   4.851678147  -9.729799306   4.873027924
## [396] -16.306195814  11.988856534  12.672854287  -4.324051898  12.479308503
## [401]   5.057292262  -1.248588439   1.670200694  -3.143547838   9.418566968
## [406] -16.940853900  -3.845094745  -4.698507237   0.448891261  -4.347881631
## [411]  33.488856534  15.052118369  14.613659660 -18.884179574  -7.033252655
## [416]  -2.604555958  13.446445319  32.990843663   2.917326990 -28.892047404
## [421]   6.984349103   4.795444042   6.395710627 -40.231398818 -11.359509740
## [426]  -9.940853900  -7.871178194 -11.109156337  -4.588020541  -9.351108739
## [431]  -8.104555958  13.567373464   3.941156723   2.657158970  -8.430199184
## [436]  -7.638907115 -10.663310363   0.003578458 -14.824451776  10.214499760
## [441] -17.860216548   2.791683764 102.608634884   4.256452162 -17.894167827
## [446]  -4.342841030 -21.145187694  -9.206369451  -4.940853900   4.420432552
## [451]  -1.022997816  -5.468084379  14.916046667   4.374401194  71.693977823
## [456]   6.086069648 -13.875778572   8.138369837  -4.364590685  -3.913530474
## [461]  -8.286606926 -17.150401931  -3.224051898  12.656759092  -3.499648650
## [466]   9.549024554  14.514499760   9.104018680  -7.326838784  35.356452162
## [471] -18.831746092  -1.186676380   7.528688514   7.554905255   8.408925988
## [476]  11.396387968  -5.399781942 -11.246468016   2.949024554  10.257425555
## [481]  18.289296756 -10.941694001  -1.579312610   6.868253908   8.452518247
## [486]   9.059146100   4.237222808  -9.468084379   4.280148603   8.551678147
## [491] -17.091074012   1.611712874 -15.349695124  34.170975930   6.031915621
## [496] -25.559509740  15.481962096   9.444731105  -5.661630163 -11.998101742
## [501]  -4.841827293  81.259279392  -3.031746092   8.520554098 -12.385326603
## [506]  -9.330199184   0.166000193   2.278335110 -21.326838784  15.825746581
## [511]  20.239916745  -0.521664890  16.006232051   0.813995700   3.112819559
## [516]  15.155038547   4.425861284   6.229088392  -6.573124979  -1.137113794
## [521]  -0.404555958   5.579308503  -6.586606926   9.401898258   0.911006067
## [526]   5.954771962  -4.885297493  -3.979312610   4.851678147 -13.294567705
## [531]   8.135409315  -5.142707738  -0.211143466  -8.608756459 -11.791074012
## [536]  -4.248588439   5.949024554 -15.713930352  -6.096688127  -2.281126103
## [541]  37.374401194  -9.072284879  25.528821806   4.070200694   1.956452162
## [546]  -1.379445902   5.999777835   2.600351350  10.078335110   2.911006067
## [551]   5.823340983  -5.991074012  -1.474272009  10.935857249  -0.623095869
## [556]  -4.399781942  -3.433999807  -4.068084379  -7.408316236  -6.950401931
## [561]  -4.506942966   2.346904131  13.664319993   4.833022307  -1.841294123
## [566]  -6.363043778   4.456452162   1.891510127  -5.459509740   2.568827423
## [571] -28.424585068   2.917326990   6.578335110  -5.059949962   1.679308503
## [576]  10.227715121  -5.535413422   1.823323620   6.518873897   6.518730953
## [581]  -4.452273132 -15.809996437   1.141863530 -20.524585068   5.602471773
## [586]  -8.114637160  33.946411305  -3.557296369  -7.049695124  -3.259949962
## [591]  11.561092885   1.367813686  -0.942574445   6.757158970   1.708018885
## [596]   6.335409315   6.735409315   0.003578458   6.491243541  10.862066278
## [601]  15.693323620   7.891510127 -10.661630163   3.342703631  13.913393074
## [606]   6.091510127  -2.240147093   2.169627179  38.085189204  -0.007783066
## [611]  -0.956827037  18.968920371   6.491243541  12.506232051 -23.414637160
## [616]  -7.188420419   5.379308503   5.252118369  33.352518247   2.296417435
## [621]   3.365559971   6.777628302 -12.140720608  -6.196421542   2.697964342
## [626]   0.404018680 -21.075778572 -31.784179574 -10.263310363  -7.422371698
```

```
## [631]   6.308925988 -12.908889751   0.914366467   4.911579581   2.337894888
## [636]  27.659852907 -11.840720608 -11.513930352  -0.404955836  16.598537857
## [641]  -3.006369451 -10.650975446   9.572987579  -0.114370574   3.111712874
## [646]   1.818873897  33.989256412 -26.592047404 -12.161630163  38.507419425
## [651]   0.241863530  10.551811439  -2.745934846   1.285762718  -4.138907115
## [656]   1.760386078  -1.771311486  16.247696175  -1.822371698   2.026875021
## [661]   4.033862407  -0.988287126 -24.430199184  12.501898258  -2.748321853
## [666]  12.421527490   6.672854287  -5.186606926 -21.516277015  -3.411143466
## [671]  -0.098808550   5.148891261   6.637089515   1.537089515 -13.606369451
## [676]  -2.671311486  -2.669497993 -21.394567705  18.676825382  27.928248291
## [681]   3.368408243   4.301898258  11.340050038 -10.778739095  -4.966537471
## [686]   5.761092885 -14.860483133 -10.953547838 -26.941827293   0.878508747
## [691]   7.900351350   1.999777835   6.018300382  -2.471178194  -1.281583689
## [696] -15.840853900   6.465559971  -3.763310363  33.626568091  -5.106942966
## [701]  -3.042759829  -5.192047404  13.317500627  -6.478739095   0.821527490
## [706] -10.375511987  10.024047791  -1.381699618 -15.094167827  14.274401194
## [711]  10.770200694  20.616046667   6.120554098  -8.960083255  20.355878647
## [716]   3.893057034 -30.440853900 -14.792580575 -10.640853900  -9.460216548
## [721]  12.098097635 -16.340853900  10.254065154 -14.575511987 -13.733426292
## [726]  -7.488420419   6.636956222  -0.160083255  -7.838907115 -12.159949962
## [731]   7.228248291  -2.799781942   0.611006067 -10.573965079   2.236249415
## [736]  -7.078739095  -3.806236158  -7.250975446  -1.560239528   0.257158970
## [741]  -3.398808550  22.975948102 -20.480819173  -2.602835413  10.042703631
## [746]  -7.524585068  19.862199570 -13.788420419  -2.278472510   1.775948102
## [751]  10.485629426  -8.596688127   9.526168213   4.886602819 -43.500981576
## [756]  -9.338136470   8.293763842  -6.719544468   7.961492763   8.290096512
## [761] -38.643507493  13.866573708  -5.521664890  -8.633999807  -3.148321853
## [766]   1.321527490 -11.348588439  14.247696175 -11.273831787  -4.985326603
## [771]  -4.635413422  -9.586942966   8.047610939 -19.586166703  -8.798101742
## [776]   1.455878647 -13.668084379  -2.945094745  -2.122371698  -5.296421542
## [781]  -0.735680007  -1.760216548  -5.542707738   5.167240171  18.731915621
## [786]  -3.937093622   0.800351350   4.746637546  10.130635299   5.745357224
## [791]   6.162066278  -5.601195557 -10.586166703  -2.950975446  -5.928212054
## [796]  -0.281699618   3.097257535   9.014673397   5.494030427  12.800092477
## [801]  -4.568924479  19.220953975   0.346637546  -2.399648650   9.792216934
## [806]   0.452118369   3.405832173 -18.531746092  -8.663310363 -15.080819173
## [811]   3.102471773  39.504018680 -11.360216548  27.031475399  -1.508756459
## [816]  64.285924096  -2.113530474   3.954905255 -14.209156337  -7.823171453
## [821]   0.454905255  12.511006067  -9.482673010  -5.939613922 -12.908316236
## [826] -28.426972076   2.364586578   1.351811439   7.605525244  52.905832173
## [831] -12.097528227 -10.063246525   7.262066278  11.436249415 -19.826972076
## [836]  40.271787946  -0.468084379  -3.369497993  -9.178739095  -8.494167827
## [841] -19.930199184   0.403578458  -5.160083255   0.765559971  -3.632893121
## [846]  40.287576211  -1.313530474  -9.007783066   6.475948102  -9.778739095
## [851] -20.560483133 -15.895981320  -8.413530474
```

Ajustamos los parámetros de la red en función de los datos observacionales.

```
fitted.sachs <- bn.fit(sachs.new.bn, sachs.obs)

fitted.sachs
```

```
##
##   Bayesian network parameters
##
##   Parameters of node Raf (Gaussian distribution)
##
## Conditional density: Raf | PKA + PKC
## Coefficients:
##   (Intercept)            PKA            PKC
## 62.1990459613  -0.0003792075  -0.1777450909
```

```
## Standard deviation of the residuals: 41.84243
##
##   Parameters of node Mek (Gaussian distribution)
##
## Conditional density: Mek | Raf + PKA + PKC
## Coefficients:
##   (Intercept)            Raf            PKA            PKC
## -1.0902746221   0.5208447051  -0.0006520267   0.0396624572
## Standard deviation of the residuals: 16.73515
##
##   Parameters of node Plcg (Gaussian distribution)
##
## Conditional density: Plcg
## Coefficients:
## (Intercept)
##    19.49484
## Standard deviation of the residuals: 14.67289
##
##   Parameters of node PIP2 (Gaussian distribution)
##
## Conditional density: PIP2 | Plcg + PIP3
## Coefficients:
## (Intercept)         Plcg         PIP3
##  52.3272605    0.3628681    0.7273050
## Standard deviation of the residuals: 89.89495
##
##   Parameters of node PIP3 (Gaussian distribution)
##
## Conditional density: PIP3 | Plcg
## Coefficients:
## (Intercept)         Plcg
##   24.340344     0.313834
## Standard deviation of the residuals: 33.90992
##
##   Parameters of node Erk (Gaussian distribution)
##
## Conditional density: Erk | Mek + PKA
## Coefficients:
##  (Intercept)           Mek           PKA
## -23.24874333   -0.02988581    0.08170673
## Standard deviation of the residuals: 82.90612
##
##   Parameters of node Akt (Gaussian distribution)
##
## Conditional density: Akt | Erk + PKA
## Coefficients:
## (Intercept)          Erk          PKA
##  1.87460049   1.36522290   0.01732143
## Standard deviation of the residuals: 14.83913
##
##   Parameters of node PKA (Gaussian distribution)
##
## Conditional density: PKA | PKC
## Coefficients:
## (Intercept)          PKC
## 554.3907306    0.8411526
## Standard deviation of the residuals: 427.94
##
##   Parameters of node PKC (Gaussian distribution)
##
## Conditional density: PKC
## Coefficients:
```

```
## (Intercept)
##    15.01897
## Standard deviation of the residuals: 11.59194
##
##   Parameters of node P38 (Gaussian distribution)
##
## Conditional density: P38 | PKA + PKC
## Coefficients:
##  (Intercept)           PKA           PKC
## 1.514433e+01  5.905402e-04  1.234783e+00
## Standard deviation of the residuals: 13.13428
##
##   Parameters of node Jnk (Gaussian distribution)
##
## Conditional density: Jnk | PKA + PKC
## Coefficients:
##  (Intercept)           PKA           PKC
## 52.953602851  -0.005306097  -0.764801188
## Standard deviation of the residuals: 42.1489
```

Con los datos discretizados por cuantiles:

```
fitted.sachs.q <- bn.fit(sachs.new.bn, sachs.obs)

fitted.sachs.q
```

```
##
##   Bayesian network parameters
##
##   Parameters of node Raf (Gaussian distribution)
##
## Conditional density: Raf | PKA + PKC
## Coefficients:
##   (Intercept)            PKA            PKC
## 62.1990459613  -0.0003792075  -0.1777450909
## Standard deviation of the residuals: 41.84243
##
##   Parameters of node Mek (Gaussian distribution)
##
## Conditional density: Mek | Raf + PKA + PKC
## Coefficients:
##   (Intercept)            Raf            PKA            PKC
## -1.0902746221   0.5208447051  -0.0006520267   0.0396624572
## Standard deviation of the residuals: 16.73515
##
##   Parameters of node Plcg (Gaussian distribution)
##
## Conditional density: Plcg
## Coefficients:
## (Intercept)
##    19.49484
## Standard deviation of the residuals: 14.67289
##
##   Parameters of node PIP2 (Gaussian distribution)
##
## Conditional density: PIP2 | Plcg + PIP3
## Coefficients:
## (Intercept)         Plcg         PIP3
##  52.3272605    0.3628681    0.7273050
## Standard deviation of the residuals: 89.89495
##
```

```
##   Parameters of node PIP3 (Gaussian distribution)
##
## Conditional density: PIP3 | Plcg
## Coefficients:
## (Intercept)         Plcg
##   24.340344     0.313834
## Standard deviation of the residuals: 33.90992
##
##   Parameters of node Erk (Gaussian distribution)
##
## Conditional density: Erk | Mek + PKA
## Coefficients:
##  (Intercept)           Mek           PKA
## -23.24874333   -0.02988581    0.08170673
## Standard deviation of the residuals: 82.90612
##
##   Parameters of node Akt (Gaussian distribution)
##
## Conditional density: Akt | Erk + PKA
## Coefficients:
## (Intercept)          Erk          PKA
##  1.87460049   1.36522290   0.01732143
## Standard deviation of the residuals: 14.83913
##
##   Parameters of node PKA (Gaussian distribution)
##
## Conditional density: PKA | PKC
## Coefficients:
## (Intercept)          PKC
## 554.3907306    0.8411526
## Standard deviation of the residuals: 427.94
##
##   Parameters of node PKC (Gaussian distribution)
##
## Conditional density: PKC
## Coefficients:
## (Intercept)
##    15.01897
## Standard deviation of the residuals: 11.59194
##
##   Parameters of node P38 (Gaussian distribution)
##
## Conditional density: P38 | PKA + PKC
## Coefficients:
##  (Intercept)           PKA           PKC
## 1.514433e+01  5.905402e-04  1.234783e+00
## Standard deviation of the residuals: 13.13428
##
##   Parameters of node Jnk (Gaussian distribution)
##
## Conditional density: Jnk | PKA + PKC
## Coefficients:
##  (Intercept)           PKA           PKC
## 52.953602851  -0.005306097  -0.764801188
## Standard deviation of the residuals: 42.1489
```

```
citation("Rgraphviz")
```

```
##
## To cite package 'Rgraphviz' in publications use:
```

```
##
##   Kasper Daniel Hansen, Jeff Gentry, Li Long, Robert Gentleman, Seth
##   Falcon, Florian Hahne and Deepayan Sarkar (2019). Rgraphviz: Provides
##   plotting capabilities for R graph objects. R package version 2.30.0.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {Rgraphviz: Provides plotting capabilities for R graph objects},
##     author = {Kasper Daniel Hansen and Jeff Gentry and Li Long and Robert Gentleman and Seth
Falcon and Florian Hahne and Deepayan Sarkar},
##     year = {2019},
##     note = {R package version 2.30.0},
##   }
```