Bayesian Networks

Verena Brufatto

Dataset description

This contains data on coronary artery disease from a Danish heart clinic. The dataset contains 14 discrete variables recorded for 236 patients, 107 of whom actually had the disease. The dataset includes five background variables (sex, hypercholesterolemia, smoking, heridary disposition and workload), one recording whether or not the patient has coronary artery disease, four variables representing disease manifestation (hypertrophy, previous myocardial infarct, angina pectoris, other heartfailures), and four clinical measurements (Qwave, T-wave, Q-wave informative and T-wave informative). Angina pectoris has 3 levels and the remaining 13 variables are binary.

```
## cad1
##
##  14  Variables      236  Observations
## ---------------------------------------------------------------------------------
## Sex
##        n  missing distinct
##      236        0        2
##
## Value      Female   Male
## Frequency      47    189
## Proportion  0.199  0.801
## ---------------------------------------------------------------------------------
## AngPec
##        n  missing distinct
##      236        0        3
##
## Value      Atypical     None  Typical
## Frequency        30       85      121
## Proportion    0.127    0.360    0.513
## ---------------------------------------------------------------------------------
## AMI
##        n  missing distinct
##      236        0        2
##
## Value        Definite NotCertain
## Frequency          63        173
## Proportion      0.267      0.733
## ---------------------------------------------------------------------------------
## QWave
##        n  missing distinct
##      236        0        2
##
## Value         No   Yes
## Frequency    153    83
## Proportion 0.648 0.352
## ---------------------------------------------------------------------------
## QWavecode
##        n  missing distinct
##      236        0        2
##
## Value      Nonusable    Usable
## Frequency         13       223
## Proportion     0.055     0.945
## ---------------------------------------------------------------------------
## STcode
##        n  missing distinct
##      236        0        2
##
## Value      Nonusable    Usable
## Frequency         79       157
## Proportion     0.335     0.665
## ---------------------------------------------------------------------------
## STchange
##        n  missing distinct
##      236        0        2
##
## Value         No   Yes
## Frequency    133   103
## Proportion 0.564 0.436
## ---------------------------------------------------------------------------
## SuffHeartF
##        n  missing distinct
##      236        0        2
##
## Value         No   Yes
## Frequency    167    69
## Proportion 0.708 0.292
## ---------------------------------------------------------------------------
## Hypertrophi
##        n  missing distinct
##      236        0        2
##
## Value         No   Yes
## Frequency    172    64
## Proportion 0.729 0.271
## ---------------------------------------------------------------------------
## Hyperchol
##        n  missing distinct
##      236        0        2
##
## Value         No   Yes
## Frequency    108   128
## Proportion 0.458 0.542
## ---------------------------------------------------------------------------
## Smoker
##        n  missing distinct
##      236        0        2
##
```

```
## Value         No   Yes
## Frequency     51   185
## Proportion 0.216 0.784
## ---------------------------------------------------------------------------
## Inherit
##        n  missing distinct
##      236        0        2
##
## Value         No   Yes
## Frequency    162    74
## Proportion 0.686 0.314
## ---------------------------------------------------------------------------
## Heartfail
##        n  missing distinct
##      236        0        2
##
## Value        No  Yes
## Frequency   177   59
## Proportion 0.75 0.25
## ---------------------------------------------------------------------------
## CAD
##        n  missing distinct
##      236        0        2
##
## Value         No   Yes
## Frequency    129   107
## Proportion 0.547 0.453
## ---------------------------------------------------------------------------
```

Blacklist

A better approach is to incorporate our prior knowledge of the system under study into the model selection process.

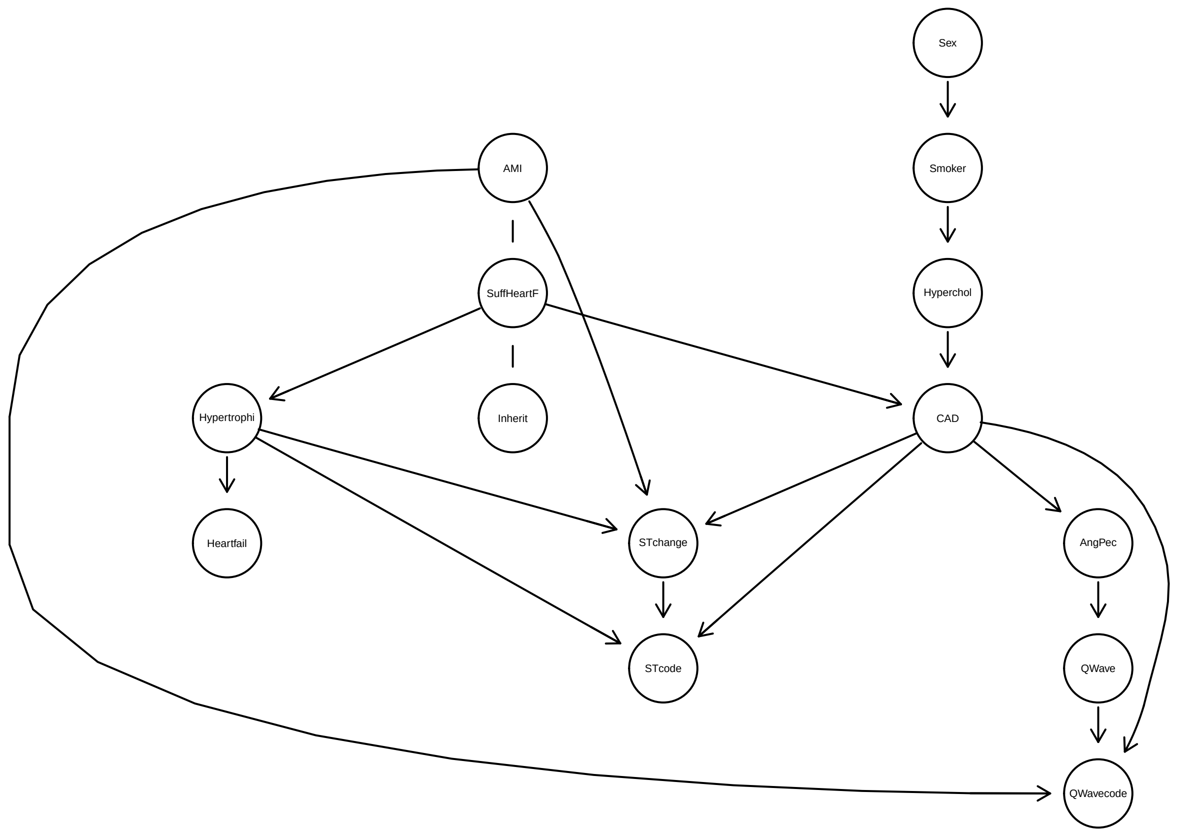
The variables are divided into four blocks, namely background variables, disease (which includes CAD and hypertrophy), disease manifestations and clinical measurements. We restrict the model selection process by blacklisting arcs that point from a later to an earlier block. We also blacklist arcs that point from being a smoker to having an hereditary disposition and from smoker, hypercholesterolemia, hereditary and workload to sex. In addition, we assume independence between different clincal tests, so that STchange and STcode cannot influence QWave and QWavecode and viceversa. Finally, we impose that the confidence on a test result (STcode and QWavecode) cannot influence whether or not a test is run (STchange, QWave).

```
##           from          to
## 1       AngPec         Sex
## 2       AngPec  SuffHeartF
## 3       AngPec Hypertrophi
## 4       AngPec   Hyperchol
## 5       AngPec      Smoker
## 6       AngPec     Inherit
## 7       AngPec         CAD
## 8          AMI         Sex
## 9          AMI  SuffHeartF
## 10         AMI Hypertrophi
## 11         AMI   Hyperchol
## 12         AMI      Smoker
## 13         AMI     Inherit
## 14         AMI         CAD
## 15       QWave         Sex
## 16       QWave      AngPec
## 17       QWave         AMI
## 18       QWave      STcode
## 19       QWave    STchange
## 20       QWave  SuffHeartF
## 21       QWave Hypertrophi
## 22       QWave   Hyperchol
## 23       QWave      Smoker
## 24       QWave     Inherit
## 25       QWave   Heartfail
## 26       QWave         CAD
## 27   QWavecode         Sex
## 28   QWavecode      AngPec
## 29   QWavecode         AMI
## 30   QWavecode       QWave
## 31   QWavecode      STcode
## 32   QWavecode    STchange
## 33   QWavecode  SuffHeartF
## 34   QWavecode Hypertrophi
## 35   QWavecode   Hyperchol
## 36   QWavecode      Smoker
## 37   QWavecode     Inherit
## 38   QWavecode   Heartfail
## 39   QWavecode         CAD
## 40      STcode         Sex
## 41      STcode      AngPec
## 42      STcode         AMI
## 43      STcode       QWave
## 44      STcode   QWavecode
## 45      STcode    STchange
## 46      STcode  SuffHeartF
## 47      STcode Hypertrophi
## 48      STcode   Hyperchol
## 49      STcode      Smoker
## 50      STcode     Inherit
## 51      STcode   Heartfail
## 52      STcode         CAD
## 53    STchange         Sex
## 54    STchange      AngPec
## 55    STchange         AMI
## 56    STchange       QWave
## 57    STchange   QWavecode
## 58    STchange  SuffHeartF
## 59    STchange Hypertrophi
## 60    STchange   Hyperchol
## 61    STchange      Smoker
## 62    STchange     Inherit
## 63    STchange   Heartfail
## 64    STchange         CAD
## 65  SuffHeartF         Sex
## 66 Hypertrophi         Sex
## 67 Hypertrophi  SuffHeartF
## 68 Hypertrophi   Hyperchol
## 69 Hypertrophi      Smoker
## 70 Hypertrophi     Inherit
## 71   Hyperchol         Sex
## 72      Smoker         Sex
## 73      Smoker     Inherit
## 74     Inherit         Sex
## 75   Heartfail         Sex
## 76   Heartfail  SuffHeartF
## 77   Heartfail Hypertrophi
## 78   Heartfail   Hyperchol
## 79   Heartfail      Smoker
## 80   Heartfail     Inherit
## 81   Heartfail         CAD
## 82         CAD         Sex
## 83         CAD  SuffHeartF
## 84         CAD   Hyperchol
## 85         CAD      Smoker
## 86         CAD     Inherit
```

Learn network structure

True CPDAG

We assume that the true CPDAG is the one described in Hojsgaard and Thiesson (1993). The true CPDAG has 19 arcs of which 2 undirected.

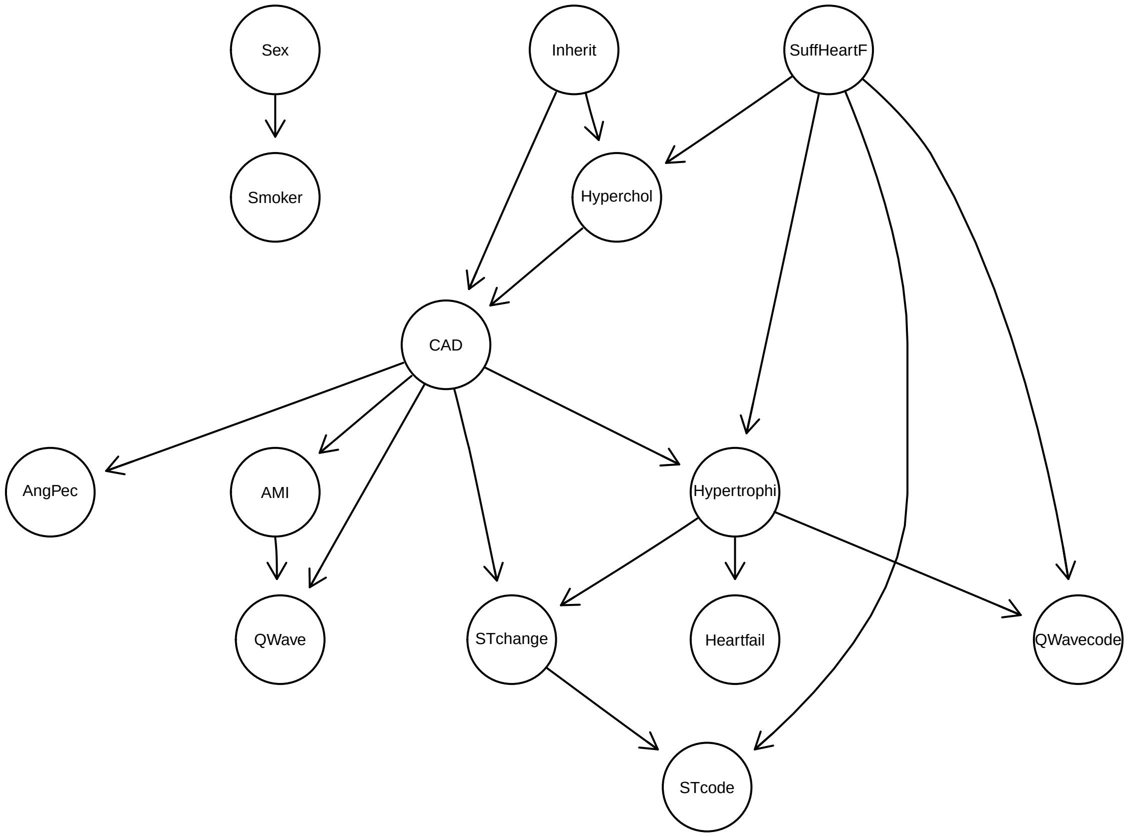


Score based algorithms

Hill climbing algorithm

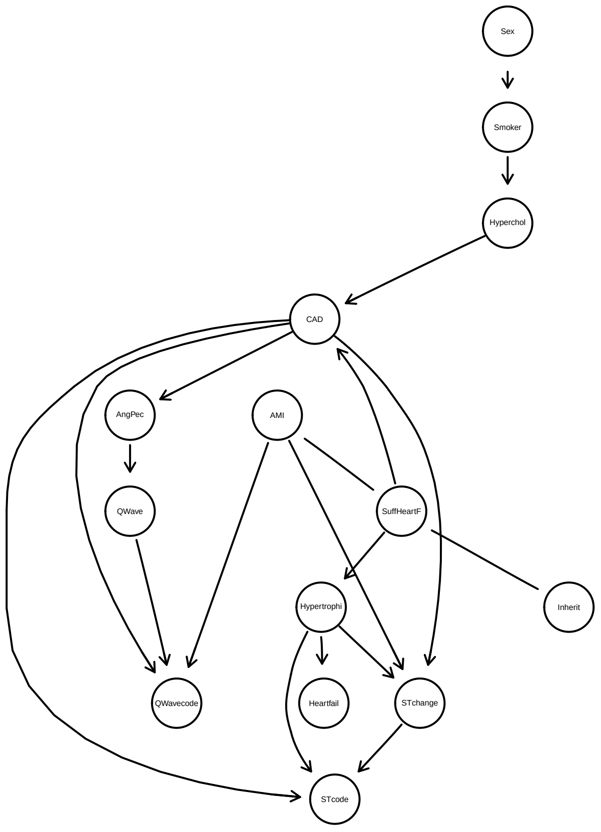
The hill climbing algorithm produces a network with 19 directed arcs.

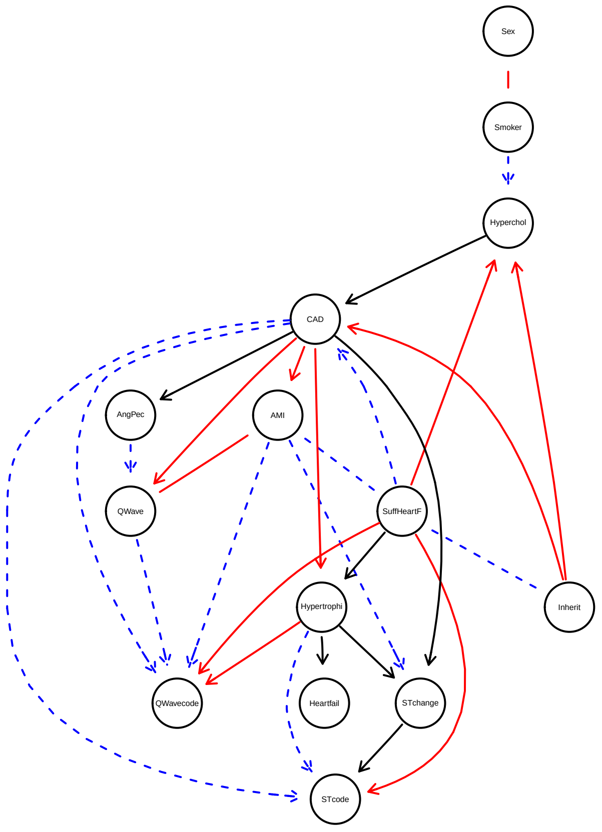
```
bn_hc <- hc(cad1, blacklist=blackL)
```



```r
unlist( bnlearn::compare(true_dag, cpdag(bn_hc)) )
```

```
## tp fp fn
##  7 11 12
```

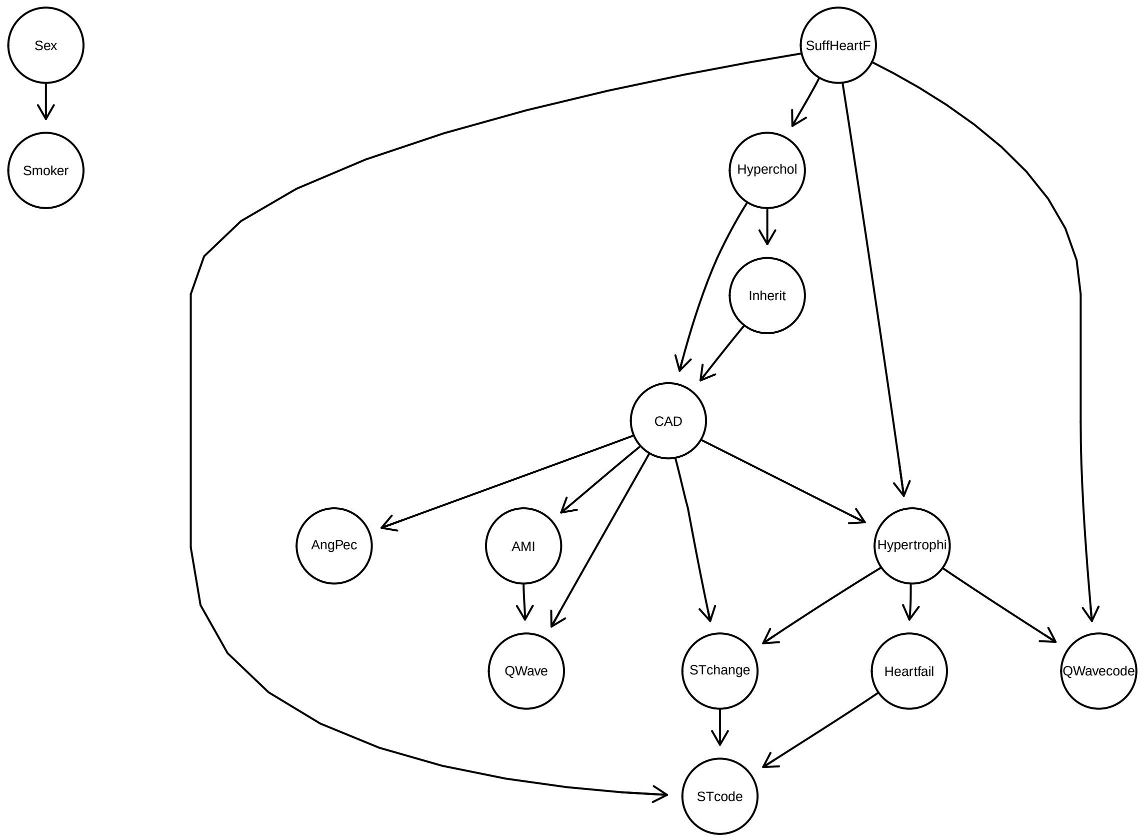




Hill climbing algorithm with random restarts

The algorithm produces a network with 21 directed arcs.

```
hc_restart = hc(cad1, score = "bde", iss = 1, restart = 10, perturb = 5, blacklist=blackL)
```



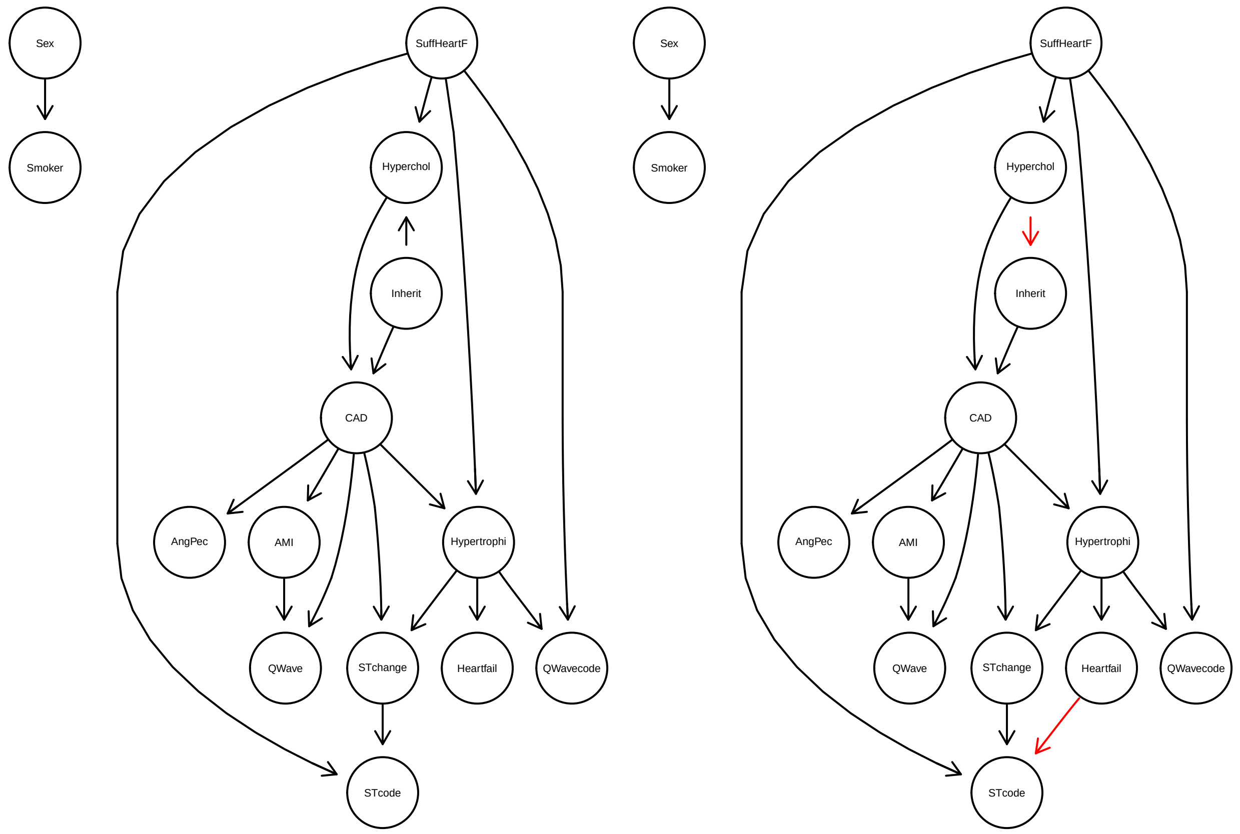
```r
# compare score based algorithms

all.equal(bn_hc, hc_restart)
```

```
## [1] "Different number of directed/undirected arcs"
```

```r
unlist( bnlearn::compare(bn_hc, hc_restart) )
```

```
## tp fp fn
## 17  2  1
```

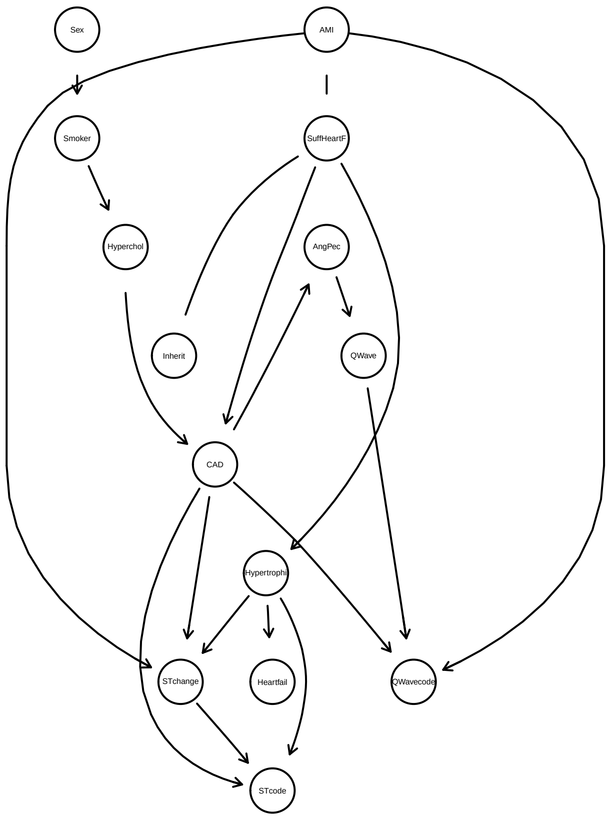


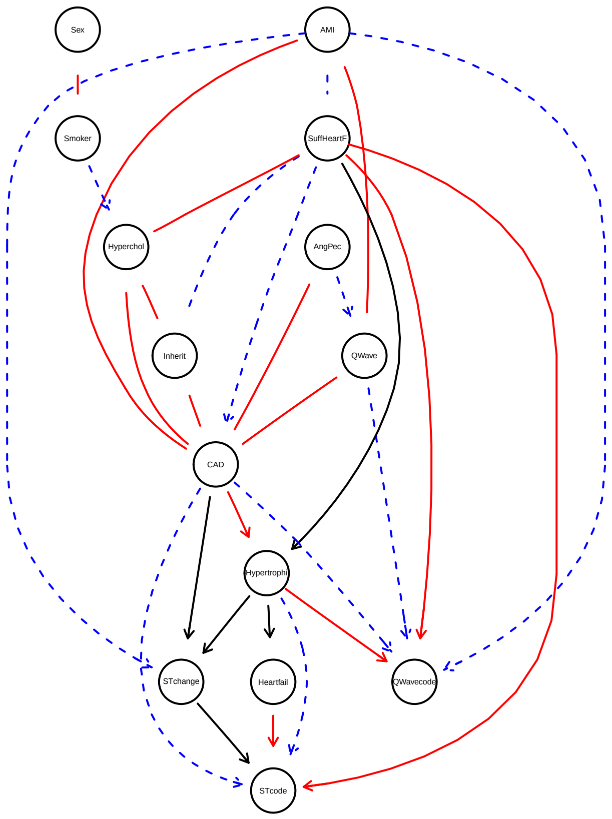
Based on the network score, the DAG produced with the hill climbing algorithm with random restarts provides a slightly worse fit to the data.

```
## [1] -1801.804 -1806.889
```

```
# compare with true dag
unlist( bnlearn::compare(true_dag, cpdag(hc_restart)) )
```

```
## tp fp fn
##  5 14 14
```



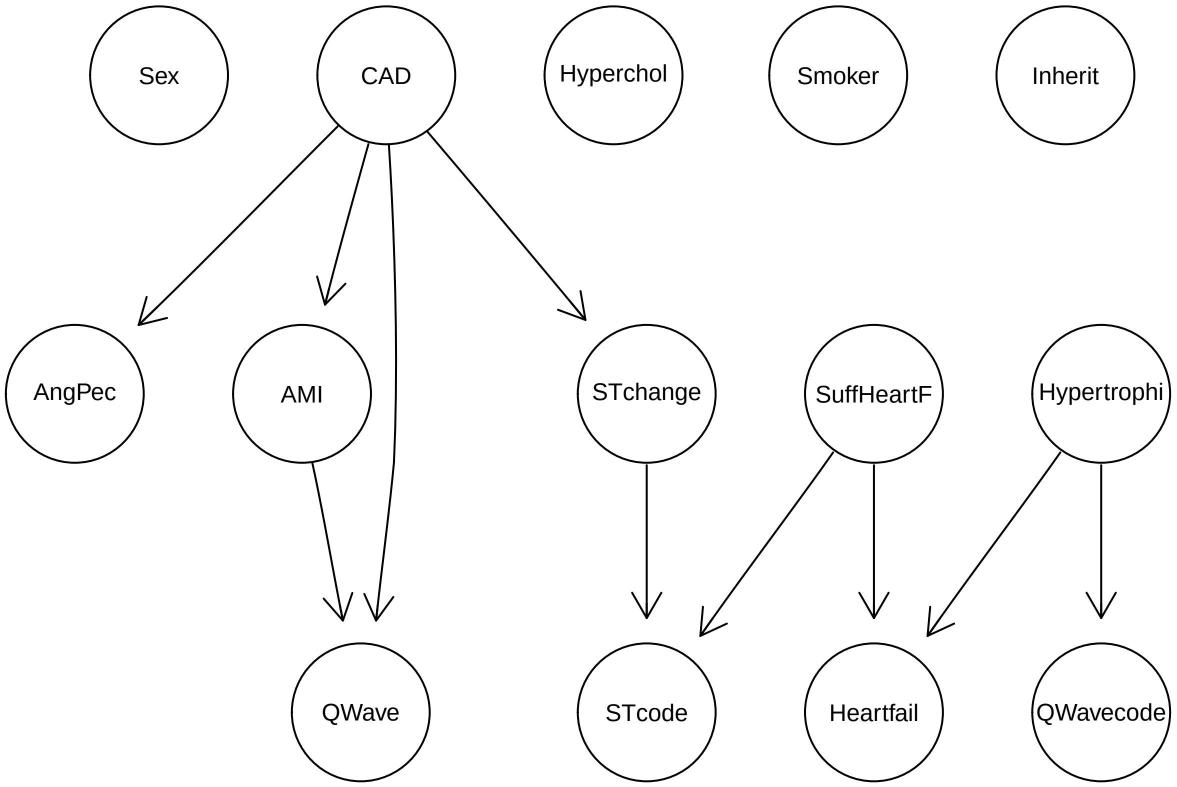


Constraint based algorithms

Semi-Interleaved Hiton-PC

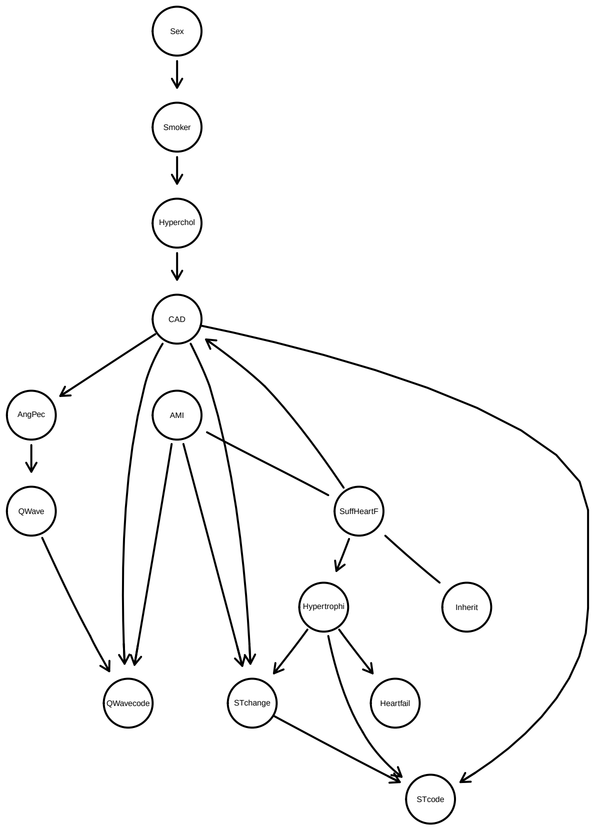
The algorihm produces a network with 10 arcs, 1 of which undirected.

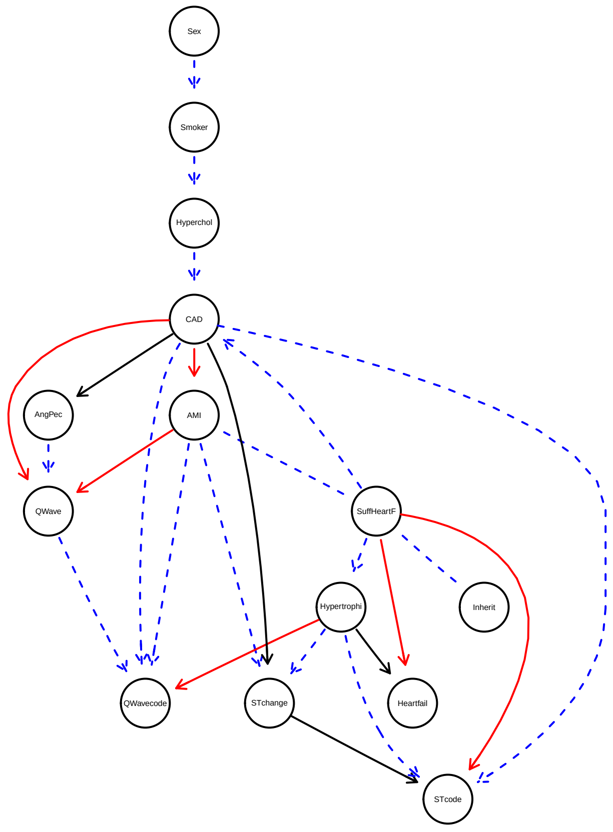
```
bn_hit <- si.hiton.pc(cad1, undirected = F,
                      blacklist = blackL)
```



```
unlist( bnlearn::compare(true_dag, bn_hit) )
```

```
## tp fp fn
##  4  6 15
```

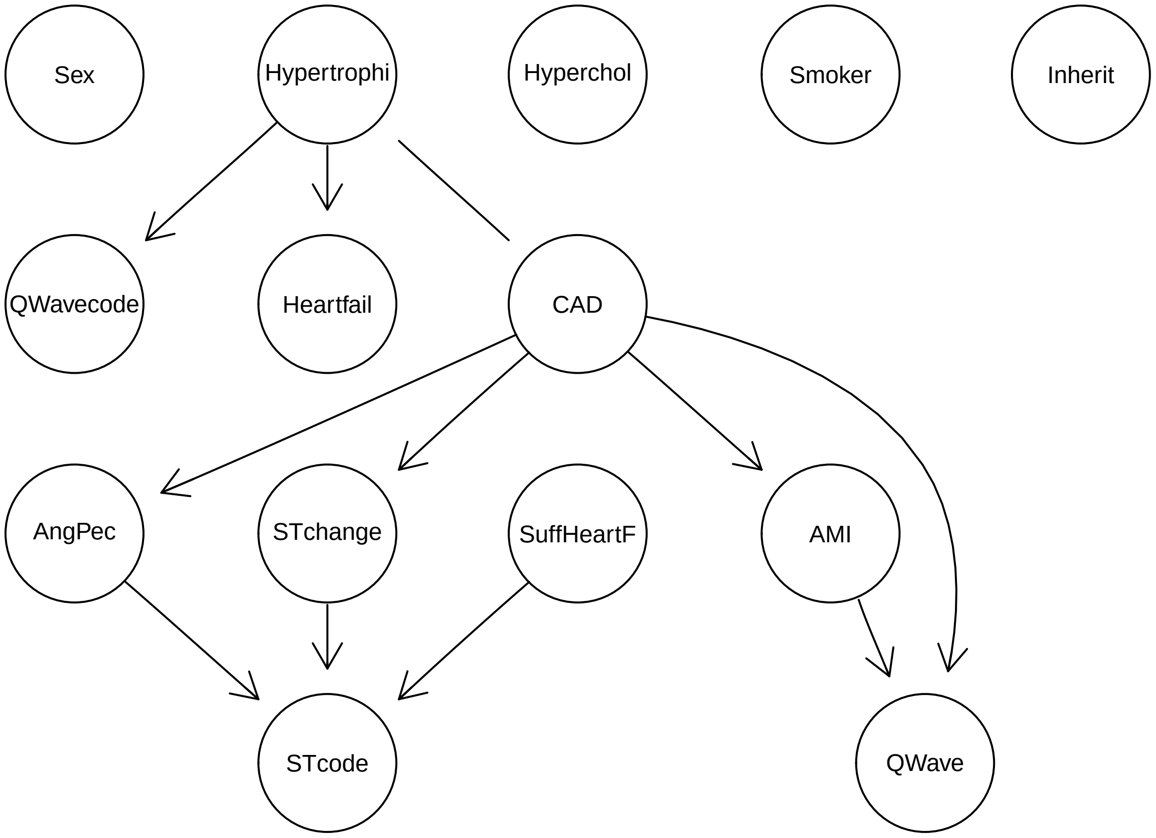




Hiton-PC with permutation tests

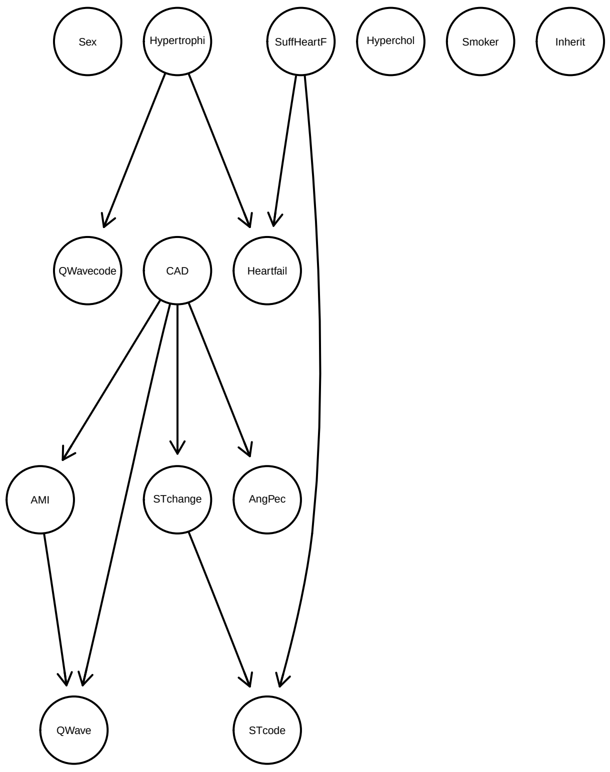
The network learned has the same number of directed and undirected arcs as the previous one but a different arc set (2 false positives and 1 false negative).

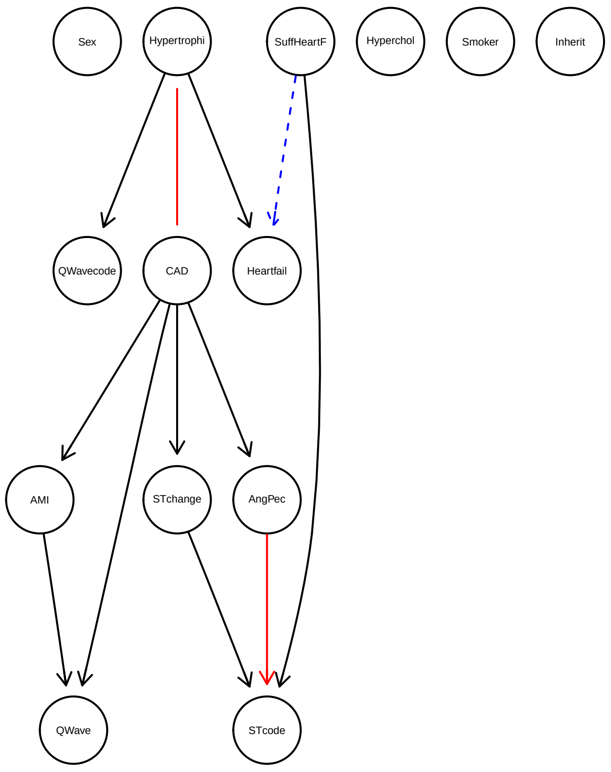
```
bn_hit2 = si.hiton.pc(cad1, test = "mc-mi", undirected = FALSE,
                      blacklist = blackL)
```



```r
unlist( bnlearn::compare(bn_hit, bn_hit2) )
```

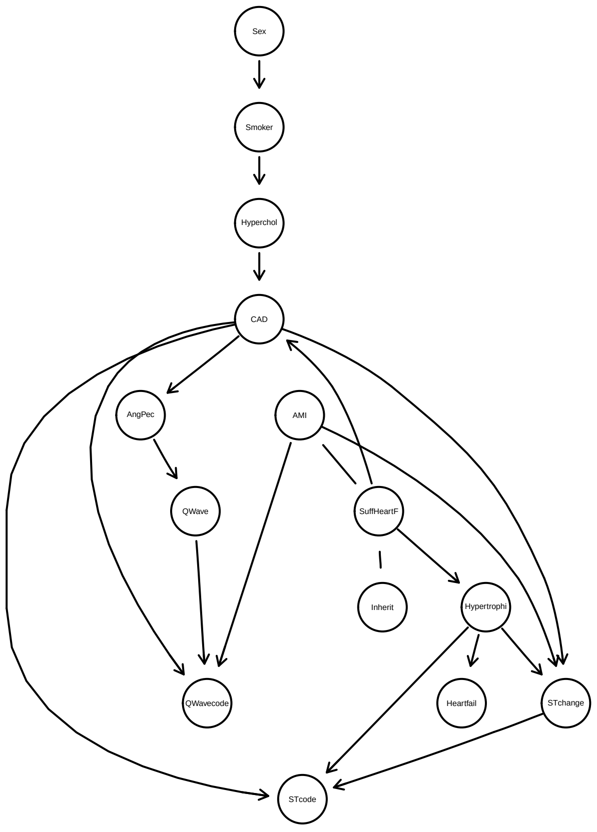
```
## tp fp fn
##  9  2  1
```

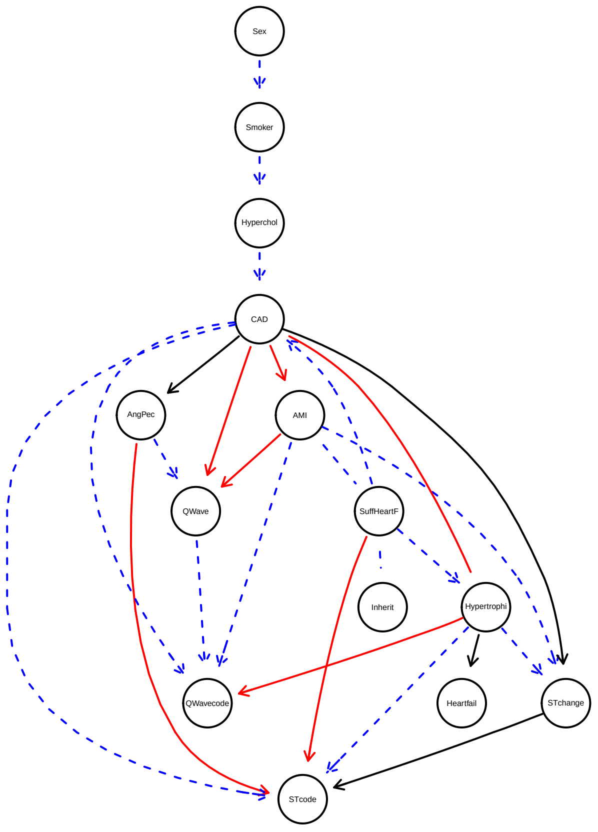




```
unlist( bnlearn::compare(true_dag, bn_hit2) )
```

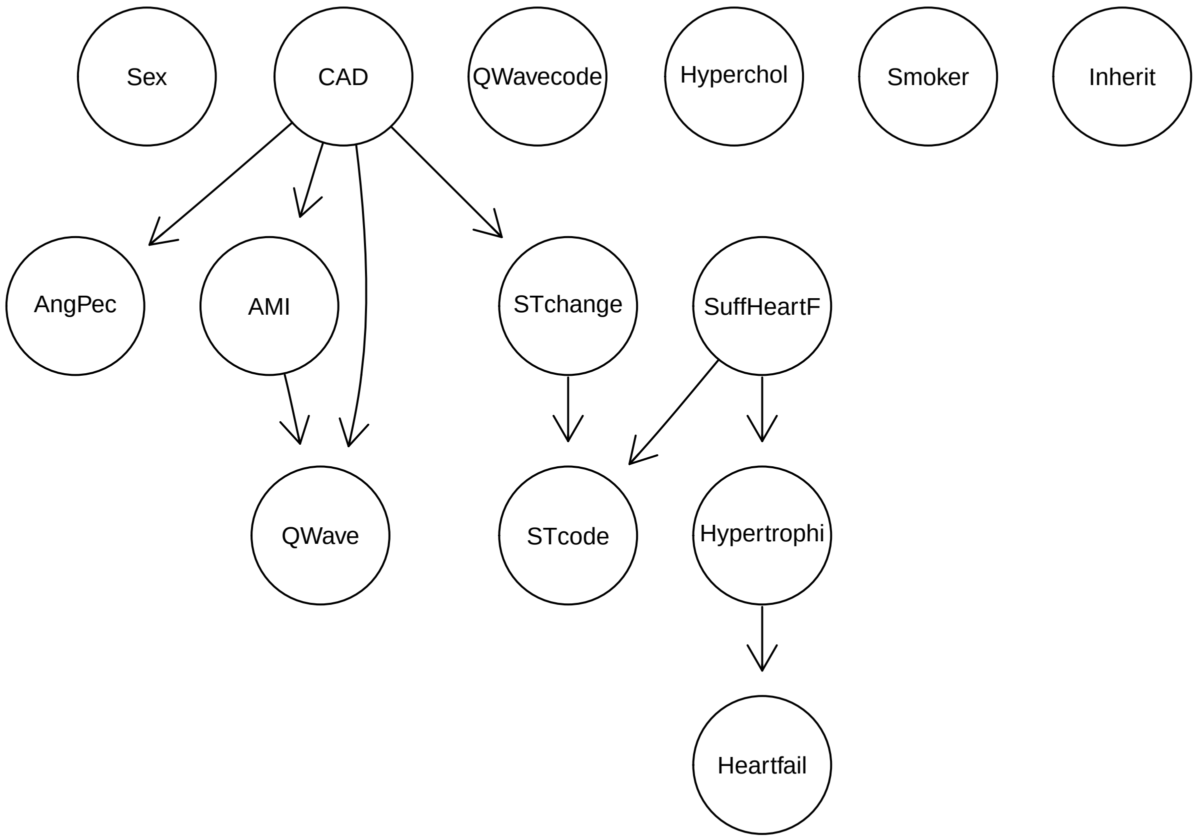
```
## tp fp fn
##  4  7 15
```





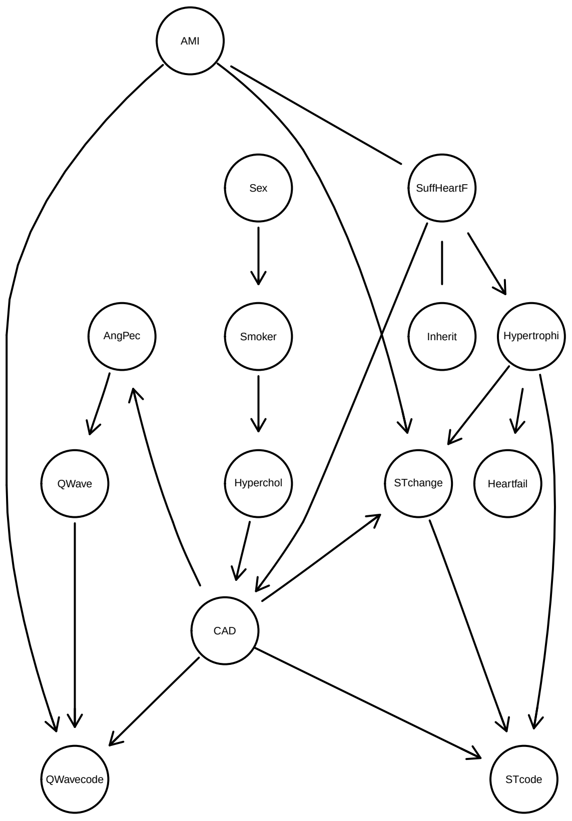
Hybrid algorithms

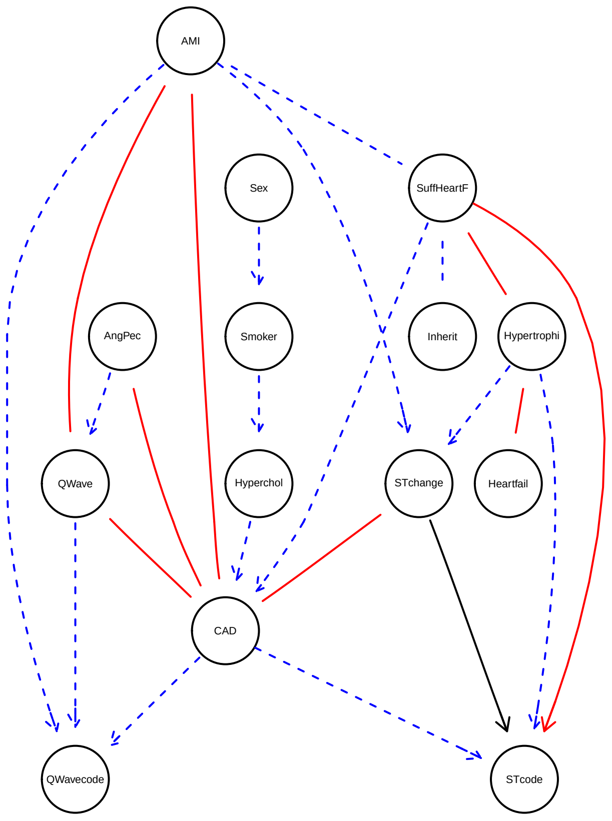
```
bn_rsmax <- rsmax2(cad1, restrict="mmpc", maximize="hc", blacklist = blackL)
```



```
unlist( bnlearn::compare(true_dag, cpdag( bn_rsmax)) )
```

```
## tp fp fn
##  1  8 18
```





Model averaging

Frequentist approach

We average multiple networks to improve the structure learned. Bootstrap resampling is applied to the dataset to learn a set of 500 CPDAGS with the hill climbing algorithm. Arcs are considered significant if they appear in at least 85% of the networks, and in the direction that appears most frequently.

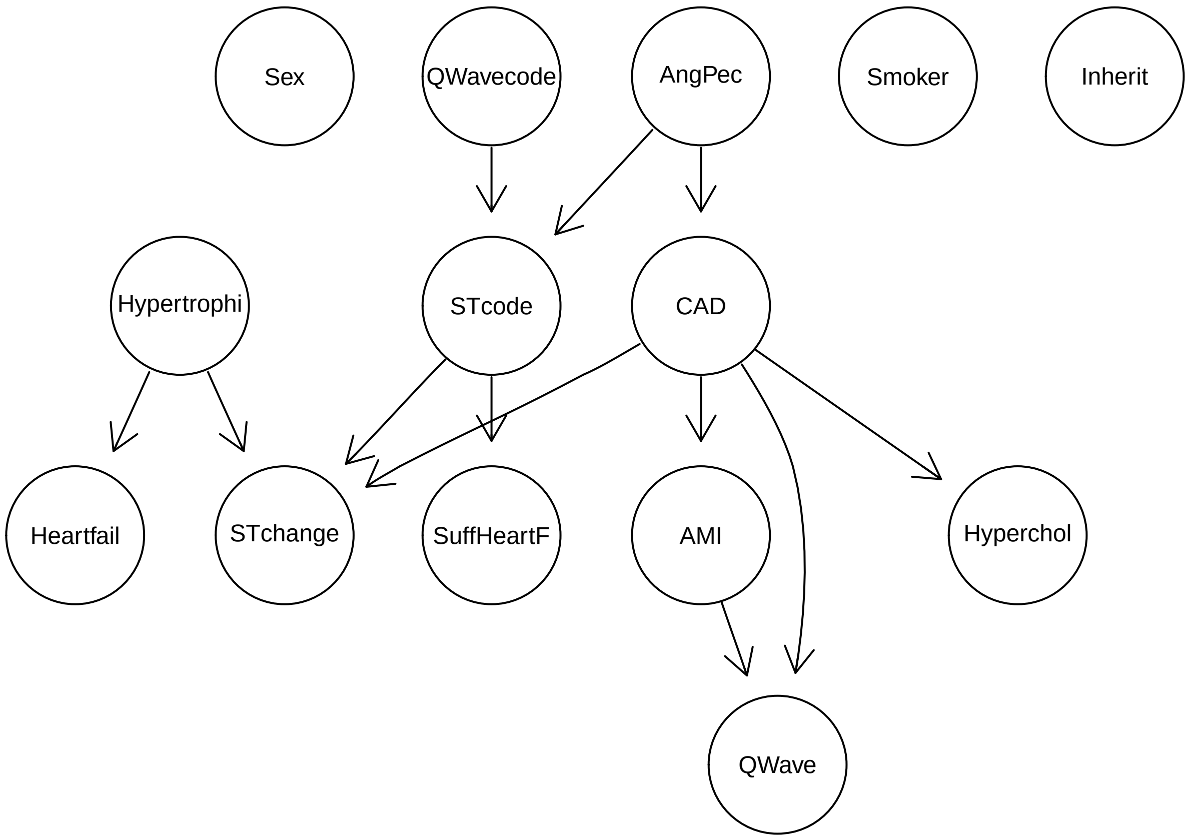
```
boot <- boot.strength(cad1, R = 500, algorithm = "hc",
                      algorithm.args = list(score = "bde", iss = 10))

boot[(boot$strength > 0.85) & (boot$direction >= 0.5), ]
```

```
##              from          to strength direction
## 18         AngPec      STcode    0.890 0.7382022
## 26         AngPec         CAD    1.000 0.8580000
## 29            AMI       QWave    0.958 0.7004175
## 57      QWavecode      STcode    0.990 0.7747475
## 71         STcode    STchange    1.000 0.8900000
## 72         STcode  SuffHeartF    1.000 0.8950000
## 111 Hypertrophi    STchange    0.860 0.8011628
## 116 Hypertrophi   Heartfail    1.000 0.7480000
## 172           CAD         AMI    0.992 0.5584677
## 173           CAD       QWave    0.984 0.6808943
## 176           CAD    STchange    0.956 0.5878661
## 179           CAD   Hyperchol    0.962 0.9729730
```

Since all the values in the direction column are well above 0.5, we can infer that the direction of the arcs is well established and that they are not score equivalent. Lowering the threshold from 85% to 50% does not change the results of the analysis, which seems to indicate that in this case the results are not sensitive to its value.

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



```
unlist( bnlearn::compare(true_dag, cpdag( avg.boot)) )
```

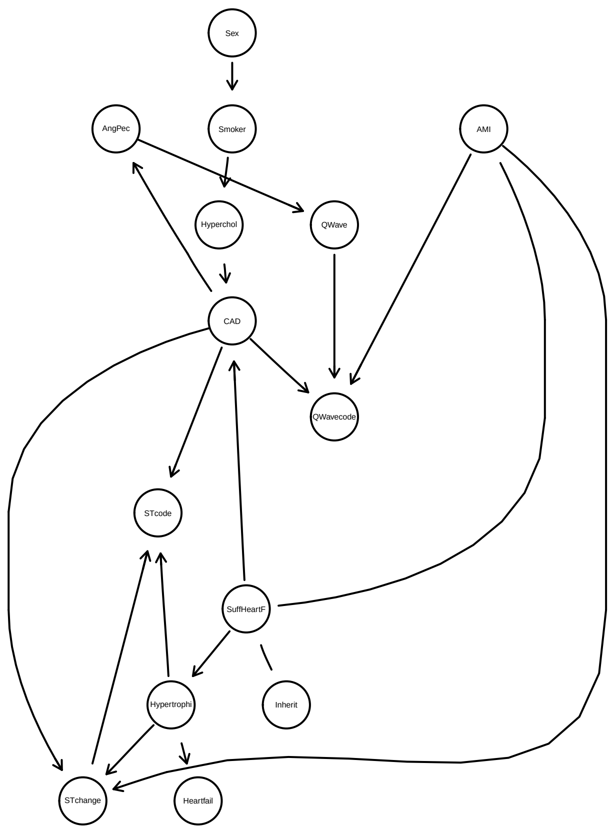
```
## tp fp fn
##  2 10 17
```

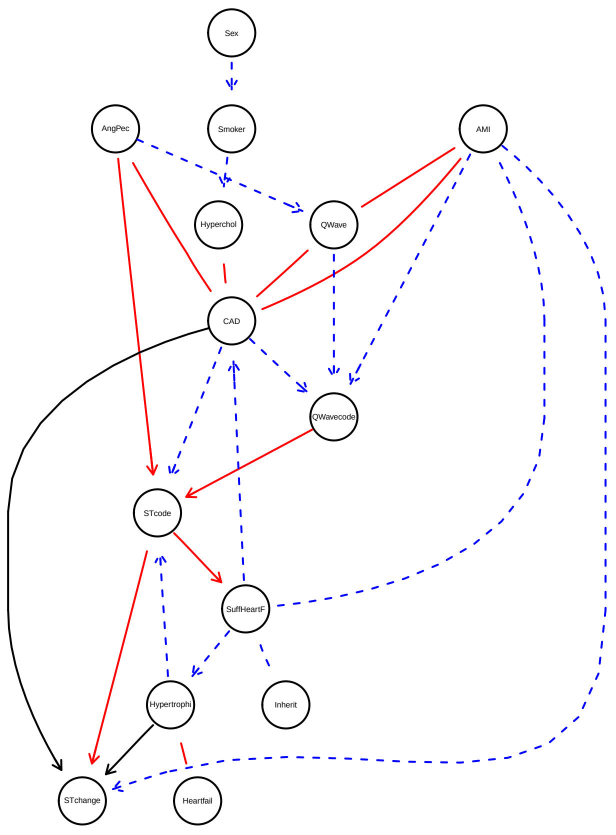
Table 1: Compare CPDAGS

	tp	fp	fn
HC restarts	5	14	14
HC	**7**	**11**	**12**
Hiton	4	6	15
Hiton permutations	4	7	15
Hybrid	1	8	18
Averaging	2	10	17

Table 2: Compare skeletons

	tp	fp	fn
HC restarts	8	11	11
HC	8	10	11
Hiton	4	6	15
Hiton permutations	4	7	15
Hybrid	5	4	14
Averaging	6	6	13





Model comparison

There are 5 archs that appear both in the true DAG and in the DAG learned by the Hill Climbing algorithm and 14 arcs that appear in the current DAG but not in the true DAG and viceversa.

Table 3: Hamming distance

	Hamming distance	SHD
HC restarts	22	26
HC	21	23
Hiton	21	24
Hiton permutations	22	24
Hybrid	18	20
Averaging	19	22

The skeleton of the network learned by the HC algorithm differs from the skeleton of the true DAG by 21 arcs, while its CPDAG differs from the true CPDAG by 23 arcs.

Parameter estimate

We choose to use the Bayesian network produced by the hill climbing algorithm for the purpose of statistical inference. The graph is chosen because it has the largest number of true positive arcs with respect to the true CPDAG among the estimated graphs. The networks produced by the constraint based algorithms and by the hybrid algorithms are not considered eligible for selection because the node CAD has no parents, making it impossible to inquire about the underlying causes of the disease.

The joint probability distribution of the model obtained with the hill climbing algorithm factorizes according to:

P(Sex, AngPec, AMI, QWave, QWavecode,STcode, STchange, SuffHeartF, Hypertrophi, Hyperchol, Smoker, Inherit,
Heartfail, CAD) = P(Sex)P(Inherit)P(SuffHeartF)P(Smoker|Sex)P(AMI|CAD)P(CAD| Inherit, Hyperchol) P(Hyperchol|Inherit, SuffHeartF) P(QWave|AMI, CAD) P(STchange| CAD, Hypertrophi) P(AngPec|CAD) P(Hypertrophi| CAD, SuffHeartF) P(Heartfail|Hypertrophi) P(QWavecode|Hypertrophi, SuffHeartF) P(STcode|STchange, SuffHeartF)

The MLE and the Bayes estimator yield similar results for the model coefficients.

```
fit_mle = bn.fit(bn_hc, cad1, method ="mle")

coef(fit_mle$CAD)
```

```
## , , Inherit = No
##
##      Hyperchol
## CAD          No       Yes
##   No  0.8214286 0.4487179
##   Yes 0.1785714 0.5512821
##
## , , Inherit = Yes
##
##      Hyperchol
## CAD          No       Yes
##   No  0.5000000 0.2600000
##   Yes 0.5000000 0.7400000
```

```
fit_bay = bn.fit(bn_hc, cad1, method = "bayes", iss = 10)

coef(fit_bay$CAD)
```

```
## , , Inherit = No
##
##      Hyperchol
## CAD          No       Yes
##   No  0.8121387 0.4503106
##   Yes 0.1878613 0.5496894
##
## , , Inherit = Yes
##
##      Hyperchol
## CAD          No       Yes
##   No  0.5000000 0.2714286
##   Yes 0.5000000 0.7285714
```

Inference

Exact inference

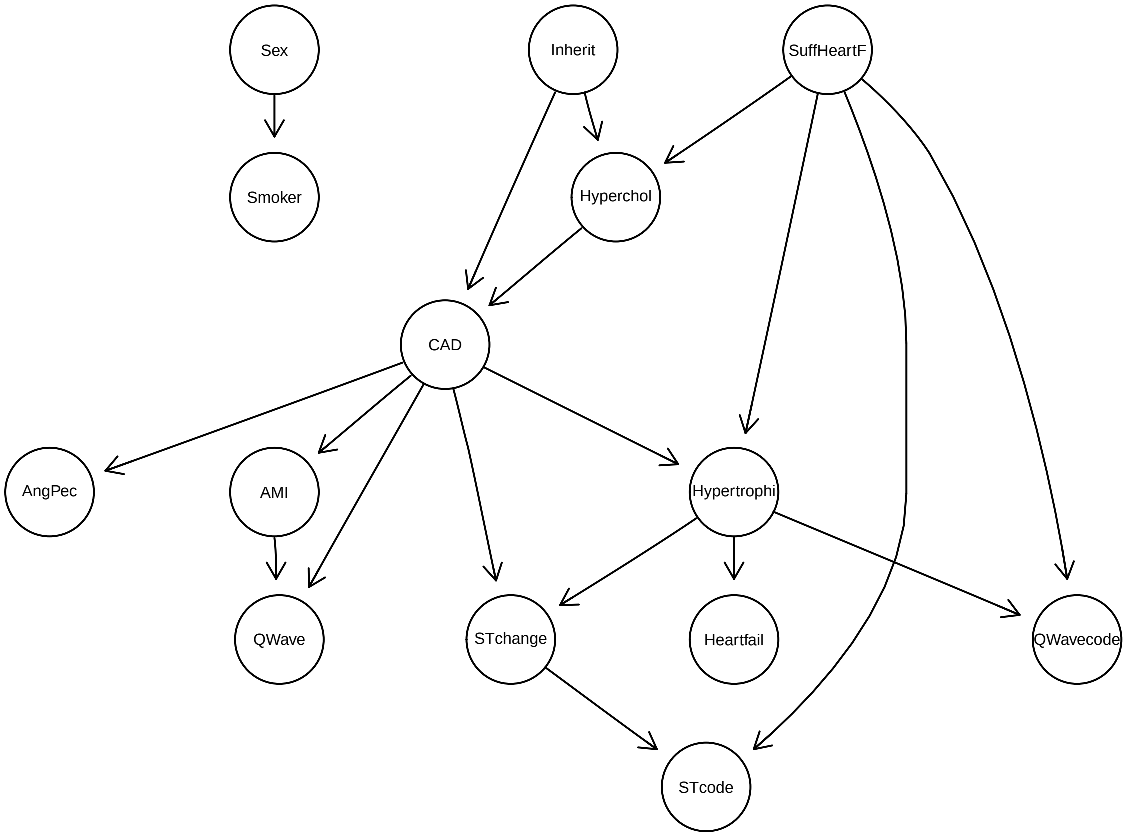


Figure 1: Bayesian network

```r
querygrain(junction, nodes = "CAD")
```

```
## $CAD
## CAD
##        No       Yes
## 0.5464106 0.4535894
```

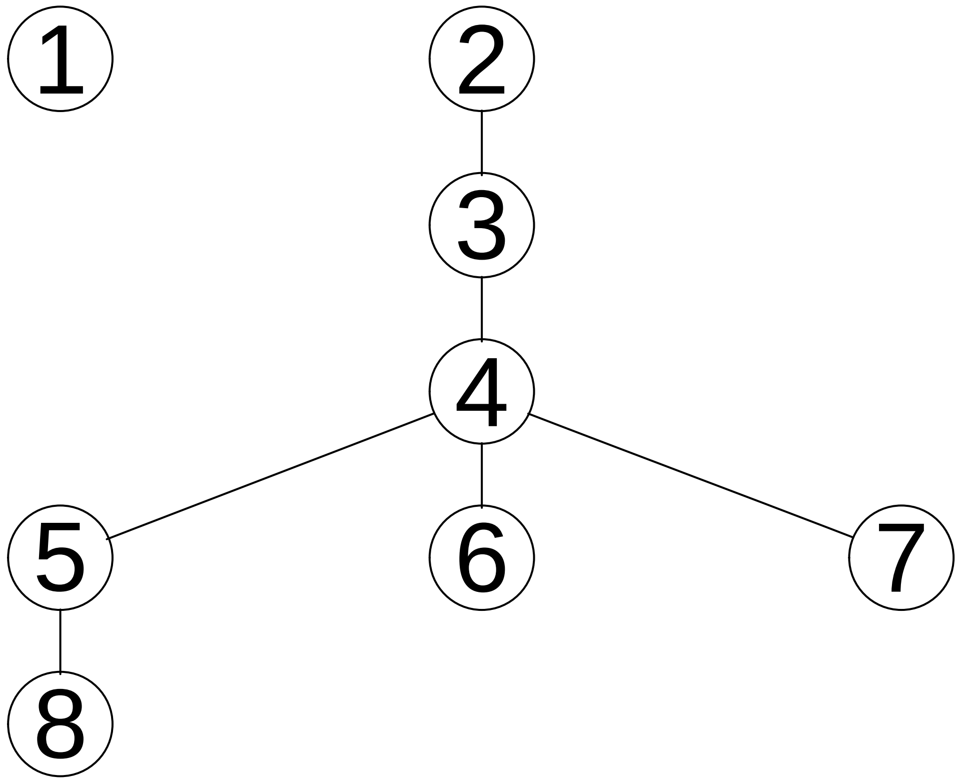


Figure 2: Junction tree

Since smoker is not directly connected to CAD in the graph, we find that $P(CAD|Smoker = Yes) = P(CAD)$.

```
gr1.ev1 <- setFinding(junction,nodes=c("Smoker"),states=list(c("Yes")))
querygrain(gr1.ev1,nodes=c("CAD"),type = "marginal")
```

```
## $CAD
## CAD
##        No       Yes
## 0.5464106 0.4535894
```

In fact, the two nodes are d-separated in the original graph.

```
bnlearn::dsep(bn_hc, x = "CAD", y = "Smoker")
```

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

The probability of observing $CAD = Yes$ is higher for the evidence $Inherit = Yes$ (about 66%) since the two nodes are connected by an arc.

```
gr1.ev1 <- setFinding(junction,nodes=c("Inherit"),states=list(c("Yes")))
querygrain(gr1.ev1,nodes=c("CAD"),type = "marginal")
```

```
## $CAD
## CAD
##        No       Yes
## 0.3377717 0.6622283
```

If both Inherit and Hypercol are present, the probability of CAD is about 74%

```
gr1.ev2 <- setFinding(junction, nodes=c("Inherit", "Hyperchol"),states=list(c("Yes"), c( "Yes" )))
querygrain(gr1.ev2,nodes=c("CAD"),type = "marginal")
```

```
## $CAD
## CAD
##   No  Yes
## 0.26 0.74
```

Adding SuffHeart to the evidence does not change the probability distribution of CAD

```
gr1.ev3 <- setFinding(junction, nodes=c("Inherit", "Hyperchol", "SuffHeartF"),states=list(c("Yes"), c(
querygrain(gr1.ev3,nodes=c("CAD"),type = "marginal")
```

```
## $CAD
## CAD
##   No  Yes
## 0.26 0.74
```

```
gr1.ev4 <- setFinding(junction, nodes=c( "SuffHeartF"),states=list(c("Yes")))
querygrain(gr1.ev4,nodes=c("CAD"),type = "marginal")
```

```
## $CAD
## CAD
##        No       Yes
## 0.4931176 0.5068824
```

However, the two nodes are not d-separated in the orignal graph, since there is a path going from SuffHeartF to CAD.

```
bnlearn::dsep(bn_hc, x = "SuffHeartF", y = "CAD")
```

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

```
bnlearn::path(bn_hc, from = "SuffHeartF", to = "CAD")
```

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

SuffHeartF and CAD are not conditionally independent even if we condition on Hyperchol, since there exist other paths between the nodes.

```
bnlearn::dsep(bn_hc, x = "SuffHeartF", y = "CAD", z = "Hyperchol")
```

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

The presence of CAD makes AngPec and QWave more likely and Hypertrophi less likely

```
gr1.ev5 <- setFinding(junction, nodes=c( "CAD"),states=list(c("Yes")))
querygrain(gr1.ev5,nodes=c("AngPec", "Hypertrophi", "QWave"),type = "marginal")
```

```
## $QWave
## QWave
##        No       Yes
## 0.4205607 0.5794393
##
## $AngPec
## AngPec
##   Atypical       None    Typical
## 0.05607477 0.12149533 0.82242991
##
## $Hypertrophi
## Hypertrophi
##        No       Yes
## 0.8589799 0.1410201
```

Next, we investigate the joint distribution of CAD and Hyperchol given that the patient has an hereditary predisposition.

```
gr1.ev6 <- setFinding(junction, nodes=c(  "Inherit"), states=list(c("Yes")))
querygrain(gr1.ev6, nodes=c("CAD", "Hyperchol"), type = "joint")
```

```
##      Hyperchol
## CAD          No       Yes
##   No  0.1620244 0.1757473
##   Yes 0.1620244 0.5002039
```

Finally, we compute the probability of CAD conditional on the evidence and the values of Hyperchol $P(CAD|Inherit = Yes, Hypercol = h), h \in \{Yes, No\}$.

```
querygrain(gr1.ev6, nodes = c("CAD", "Hyperchol"), type = "conditional")
```

```
##      Hyperchol
## CAD    No  Yes
##   No  0.5 0.26
##   Yes 0.5 0.74
```

Approximate inference

Logic sampling

By generating 10^6 observations from the fitted Bayesian network we find that $P(CAD = Yes|Inherit = Yes) = 0.65$ and $P(CAD = Yes|Hyperchol = Yes, Inherit = Yes) = 0.73$, in line with the previous findings obtained via exact inference. In both queries, $P(E)$ is large enough for the logic sampling algorithm to estimate the probability with sufficient precision, which is confirmed by the fact that the likelihood weighting approach yields similar results.

```
cpquery(fit_bay, event=(CAD=="Yes"),
evidence = (Inherit=="Yes"), method="ls", n=10^6)
```

```
## [1] 0.6516906
```

```
cpquery(fit_bay, event=(CAD=="Yes"),
evidence = (Inherit=="Yes") & (Hyperchol == "Yes"), method="ls", n=10^6)
```

```
## [1] 0.7299633
```

```
cpquery(fit_bay, event=(CAD=="Yes"),
evidence = list(Inherit ="Yes", Hyperchol = "Yes"), method="lw")
```

```
## [1] 0.7375207
```

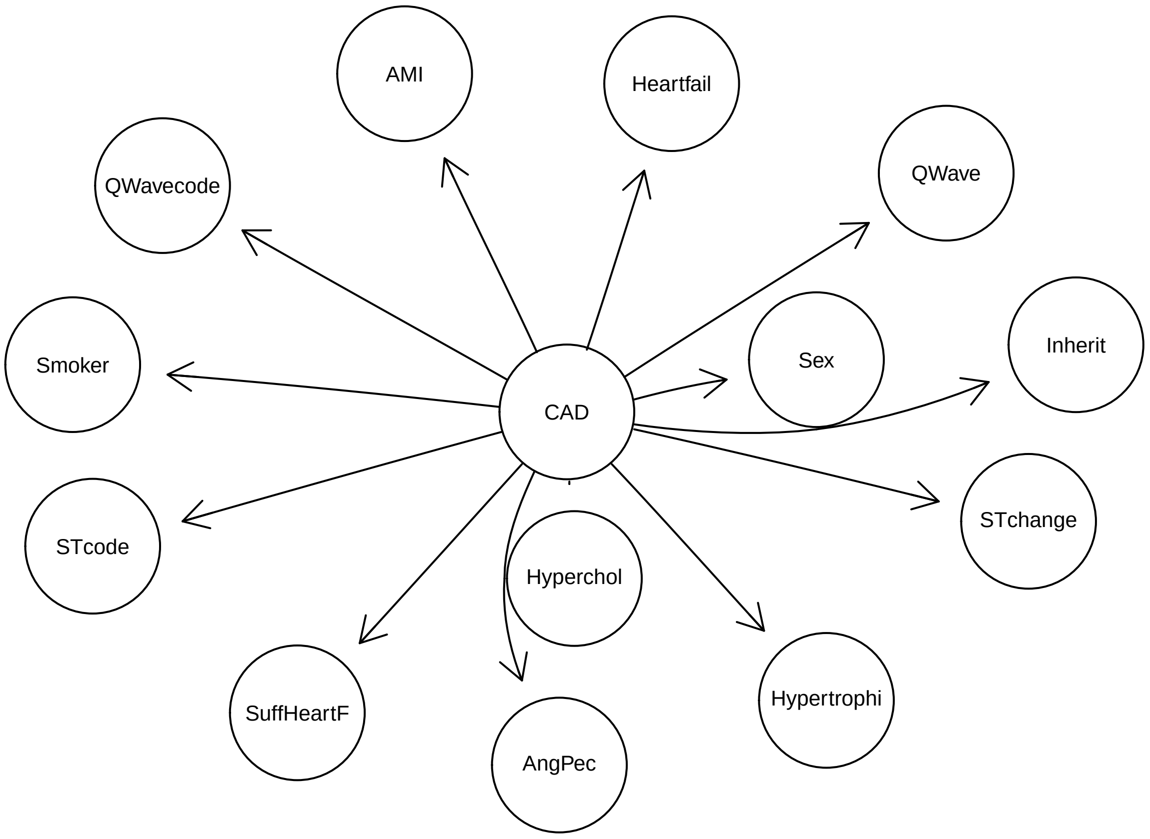
Classification

Naive Bayes Classifier

The naive Bayes classifier predicts the correct class about 85% of the time, whereas the original Bayesian network predicts the correct class only 67% of the time. Generating a random sample, as we did in this example, is optional since the database is large enough for the classifier to yield robust estimates even if applied to the original data.

```
set.seed(123)

survey <- bnlearn::rbn(fit_bay,1000)
nbcl <- naive.bayes (survey, training="CAD")
graphviz.plot(nbcl,layout="fdp")
```



```
nbcl.trained <- bn.fit(nbcl,survey)

coef(nbcl.trained$CAD)
```

```
##    No   Yes
## 0.548 0.452
```

```
coef(nbcl.trained$Hypertrophi)
```

```
##            CAD
## Hypertrophi         No       Yes
##         No  0.6478102 0.8539823
##         Yes 0.3521898 0.1460177
```

```
set.seed(123)

cv.nb <- bn.cv(nbcl,data=survey,runs=10,
method="k-fold",folds=10)
cv.nb
```

```
##
##   k-fold cross-validation for Bayesian networks
##
```

```
##   target network structure:
##    [Naive Bayes Classifier]
##   number of folds:                       10
##   loss function:                         Classification Error (Posterior, exact)
##   training node:                         CAD
##   number of runs:                        10
##   average loss over the runs:            0.1465
##   standard deviation of the loss:        0.001900292
```

```
set.seed(123)

cv.orig <- bn.cv(bn_hc, data = survey, runs=10, method ="k-fold", folds=10,
                 loss="pred", loss.args = list(target="CAD"))
cv.orig
```

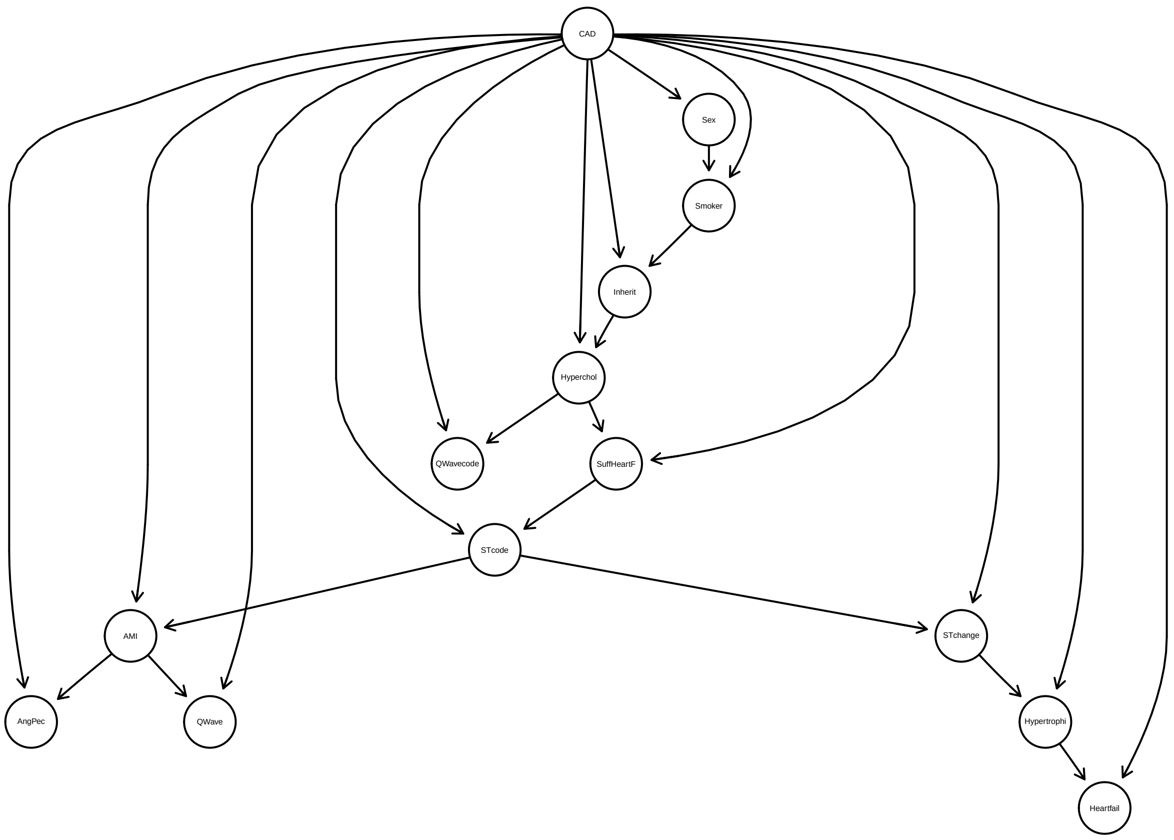
```
##
##   k-fold cross-validation for Bayesian networks
##
##   target network structure:
##    [Sex][SuffHeartF][Inherit][Hyperchol|SuffHeartF:Inherit][Smoker|Sex]
##    [CAD|Hyperchol:Inherit][AngPec|CAD][AMI|CAD][Hypertrophi|SuffHeartF:CAD]
##    [QWave|AMI:CAD][QWavecode|SuffHeartF:Hypertrophi][STchange|Hypertrophi:CAD]
##    [Heartfail|Hypertrophi][STcode|STchange:SuffHeartF]
##   number of folds:                       10
##   loss function:                         Classification Error
##   training node:                         CAD
##   number of runs:                        10
##   average loss over the runs:            0.334
##   standard deviation of the loss:        0.004242641
```

Tree-Augumented Naive Bayes Classifier

The TAN classifier has a slightly higher predictive accuracy than the Naive Bayes classifier (86%).

```
set.seed(123)

tan.cl <- tree.bayes(survey,training="CAD")
graphviz.plot(tan.cl)
```



```
tancl.trained <- bn.fit(tan.cl,survey)
coef(tancl.trained$CAD)
```

```
##    No   Yes
## 0.548 0.452
```

```
tancl.trained <- bn.fit(tan.cl,survey)
coef(tancl.trained$Hypertrophi)
```

```
## , , STchange = No
##
##              CAD
## Hypertrophi          No        Yes
##         No  0.76354680 0.68604651
##         Yes 0.23645320 0.31395349
##
## , , STchange = Yes
##
##              CAD
## Hypertrophi          No        Yes
##         No  0.31690141 0.95714286
##         Yes 0.68309859 0.04285714
```

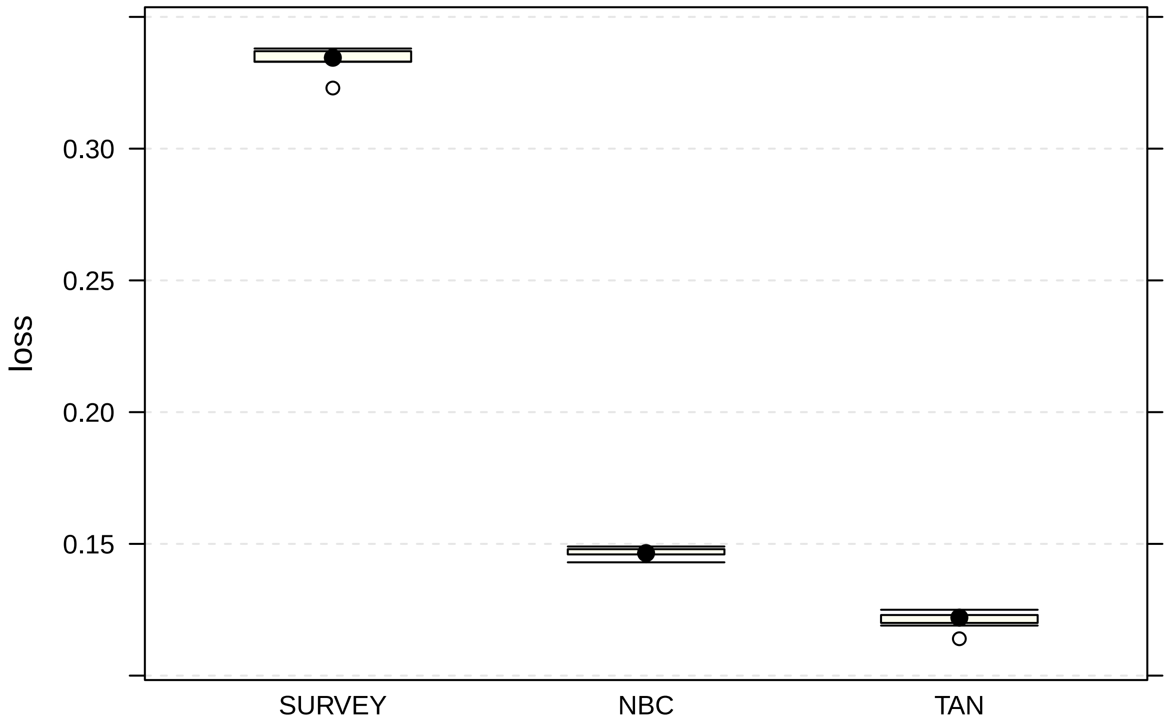
```
set.seed(123)

cv.tan <- bn.cv("tree.bayes",data=survey,
                runs=10,
                method="k-fold",folds=10,algorithm.args = list(training ="CAD"))
cv.tan
```

```
##
##   k-fold cross-validation for Bayesian networks
##
##   target learning algorithm:              TAN Bayes Classifier
##   number of folds:                        10
##   loss function:                          Classification Error (Posterior, exact)
##   training node:                          CAD
##   number of runs:                         10
##   average loss over the runs:             0.1213
##   standard deviation of the loss:         0.003093003
```

By generating 1000 samples from the bayesian network we obtain a distribution with very low variance.

```
plot(cv.orig,cv.nb,cv.tan, xlab=c("SURVEY","NBC","TAN"))
```


loss
SURVEY
NBC
TAN