The **R** Package **stagedtrees** for Structural Learning of Stratified Staged Trees

Incontro di Statistica Matematica
27 - 28 January 2020
Sestri Levante

Federico Carli
Manuele Leonelli
Eva Riccomagno
Gherardo Varando

Motivation

- In the past twenty years graphical models have become popular for modelling (the absence of) conditional independences. **Bayesian networks (BNs)** are among the most used graphical models, with two main *R* packages: **bnlearn** by Scutari (2009) and **gRain** by Højsgaard (2012).

Motivation

- In the past twenty years graphical models have become popular for modelling (the absence of) conditional independences. **Bayesian networks (BNs)** are among the most used graphical models, with two main *R* packages: **bnlearn** by Scutari (2009) and **gRain** by Højsgaard (2012).
- BNs can only represent symmetric conditional independences which in practical applications may not be fully justified. A variety of models that can take into account the asymmetric nature of many real-world data have been proposed: **context-specific BNs**, **labeled directed acyclic graphs** and **probabilistic decision graphs**.

Motivation

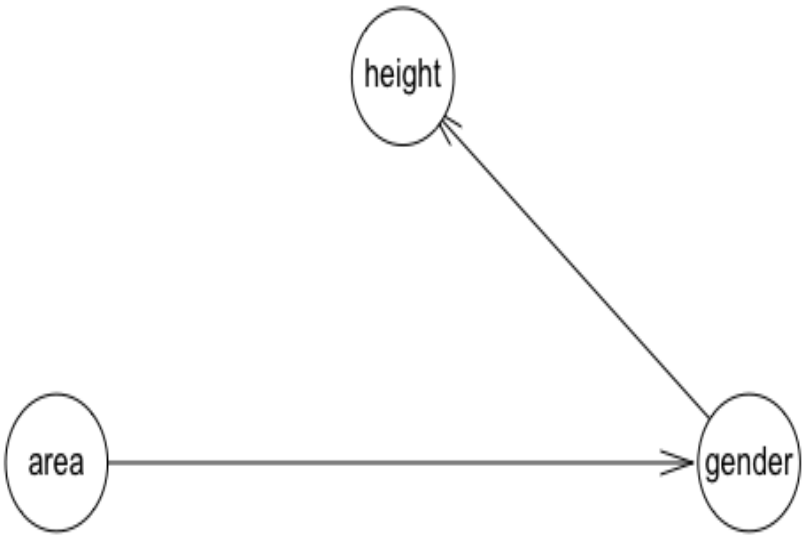
- In the past twenty years graphical models have become popular for modelling (the absence of) conditional independences. **Bayesian networks (BNs)** are among the most used graphical models, with two main *R* packages: **bnlearn** by Scutari (2009) and **gRain** by Højsgaard (2012).
- BNs can only represent symmetric conditional independences which in practical applications may not be fully justified. A variety of models that can take into account the asymmetric nature of many real-world data have been proposed: **context-specific BNs**, **labeled directed acyclic graphs** and **probabilistic decision graphs**.
- Unlike most of its competitors, the **chain event graph (CEG)** can capture all asymmetric conditional independences in a unique graph, obtained by a coalescence over the vertices of an appropriately constructed probability tree, called **staged tree**. Here we consider staged trees for modelling categorical variables.

Motivation

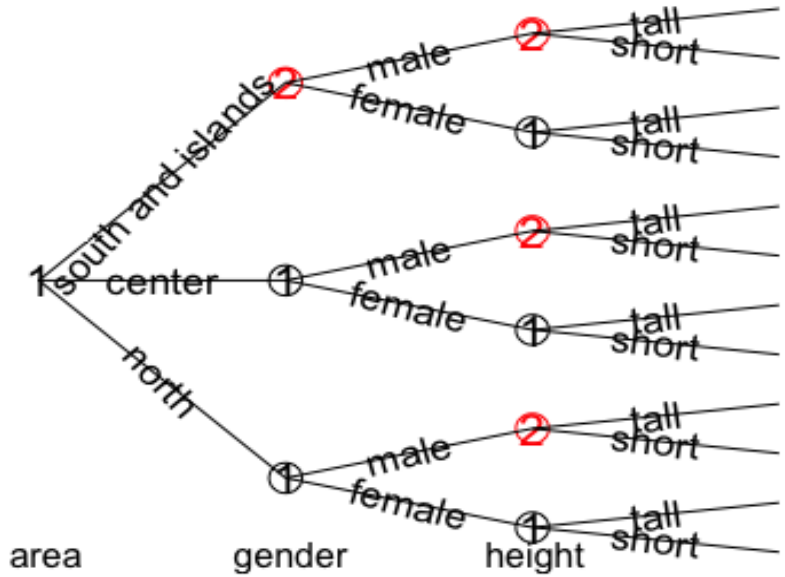
- In the past twenty years graphical models have become popular for modelling (the absence of) conditional independences. **Bayesian networks (BNs)** are among the most used graphical models, with two main *R* packages: **bnlearn** by Scutari (2009) and **gRain** by Højsgaard (2012).
- BNs can only represent symmetric conditional independences which in practical applications may not be fully justified. A variety of models that can take into account the asymmetric nature of many real-world data have been proposed: **context-specific BNs**, **labeled directed acyclic graphs** and **probabilistic decision graphs**.
- Unlike most of its competitors, the **chain event graph (CEG)** can capture all asymmetric conditional independences in a unique graph, obtained by a coalescence over the vertices of an appropriately constructed probability tree, called **staged tree**. Here we consider staged trees for modelling categorical variables.
- Only one algorithm and one *R* package (**ceg**) are available in the literature for fitting staged trees to a dataset, besides the **stagedtrees** package presented here.

A first example

Let $\mathtt{X} = (\mathtt{area}, \mathtt{gender}, \mathtt{height})$ be a random vector taking values in a product space $\mathbb{X} = \times_{i=1}^{3} \mathbb{X}_i$, where $\mathbb{X}_i$ is the sample space of the *i-th* component of $\mathtt{X}$. On a dataset with 2804 units we have:



(a) Bayesian Network



(b) Staged Tree

Main Definitions: *floret*

Directed Tree

A *directed tree* $\mathcal{T} = (V, E)$ is a tree with vertex set V and edge set E, where each vertex except for the root has one parent only, all non-leaf vertices have at least two children and all edges point away from the root. For $v, v' \in V$ let $e = (v, v') \in E$ be the edge pointing from v to v'.

Main Definitions: *floret*

Directed Tree

A *directed tree* $\mathcal{T} = (V, E)$ is a tree with vertex set V and edge set E, where each vertex except for the root has one parent only, all non-leaf vertices have at least two children and all edges point away from the root. For $v, v' \in V$ let $e = (v, v') \in E$ be the edge pointing from v to v'.

Floret

For a non-leaf v, let $E(v) = \{v' \in V : (v, v') \in E\}$ and call $\mathcal{F}(v) = (v, E(v))$ a *floret* of the tree.

Main Definitions: *floret*

Directed Tree

A *directed tree* $\mathcal{T} = (V, E)$ is a tree with vertex set V and edge set E, where each vertex except for the root has one parent only, all non-leaf vertices have at least two children and all edges point away from the root. For $v, v' \in V$ let $e = (v, v') \in E$ be the edge pointing from v to v'.

Floret

For a non-leaf v, let $E(v) = \{v' \in V : (v, v') \in E\}$ and call $\mathcal{F}(v) = (v, E(v))$ a *floret* of the tree.

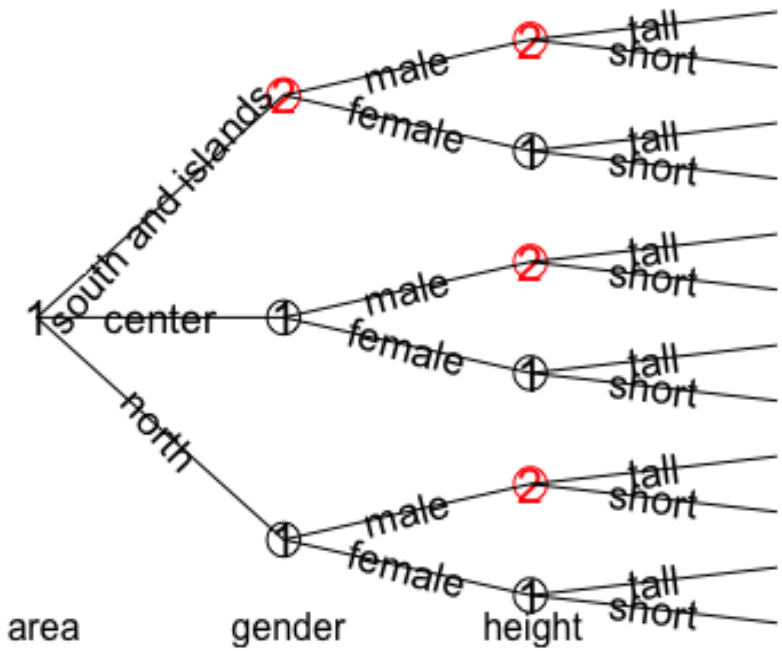
Floret Labels

Let Θ be a non-empty set of labels and $\theta : E \to \Theta$ be a function such that for any non-leaf $v \in V$ the labels in $\theta(E(v))$ are all distinct. The set $\theta(E(v))$ is denoted by $\boldsymbol{\theta}_v$ and is called the set of floret labels.

Main Definitions: *probability tree*

Probability Tree

Assume $\Theta \subseteq [0,1]$. If $\sum_{e \in E(v)} \theta(e) = 1$ for all non-leaves v, then $\mathcal{T}$ together with the $\boldsymbol{\theta}_v$'s is called a *probability tree* and $\theta_e = \theta(e)$ is the probability of the edge $e \in E$, i.e. the transition probability.



Main Definitions: *staged tree*

Staged Tree

A probability tree where for some $v, v' \in V$ the floret probability sets are equal, $\boldsymbol{\theta}_v = \boldsymbol{\theta}_{v'}$, is called *staged tree*. The vertices v and v' are said to be in the same stage and they have the same color in the graphical representation.

Main Definitions: *staged tree*

Staged Tree

A probability tree where for some $v, v' \in V$ the floret probability sets are equal, $\theta_v = \theta_{v'}$, is called *staged tree*. The vertices v and v' are said to be in the same stage and they have the same color in the graphical representation.

Statistical Model

The leaf vertices, equivalently the root-to-leaf paths, give the sample space. The product of the edge probabilities on a root-to-leaf path gives its associated probability.

Main Definitions: *stratified staged tree*

X-compatible Probability Tree

Let $\mathbf{X}$ be a discrete random vector with sample space $\mathbb{X}$. A probability tree is called $\mathbf{X}$-*compatible* if the sequential factorization

$$p(\boldsymbol{x}) = \prod_{i=2}^{k} p(x_i|\boldsymbol{x}^{i-1})p(x_1)$$

can be associated to the tree. Vertices associated to the same random variable are said to be in the same *stratum*.

Main Definitions: *stratified staged tree*

X-compatible Probability Tree

Let $\mathbf{X}$ be a discrete random vector with sample space $\mathbb{X}$. A probability tree is called $\mathbf{X}$-*compatible* if the sequential factorization

$$p(\boldsymbol{x}) = \prod_{i=2}^{k} p(x_i|\boldsymbol{x}^{i-1})p(x_1)$$

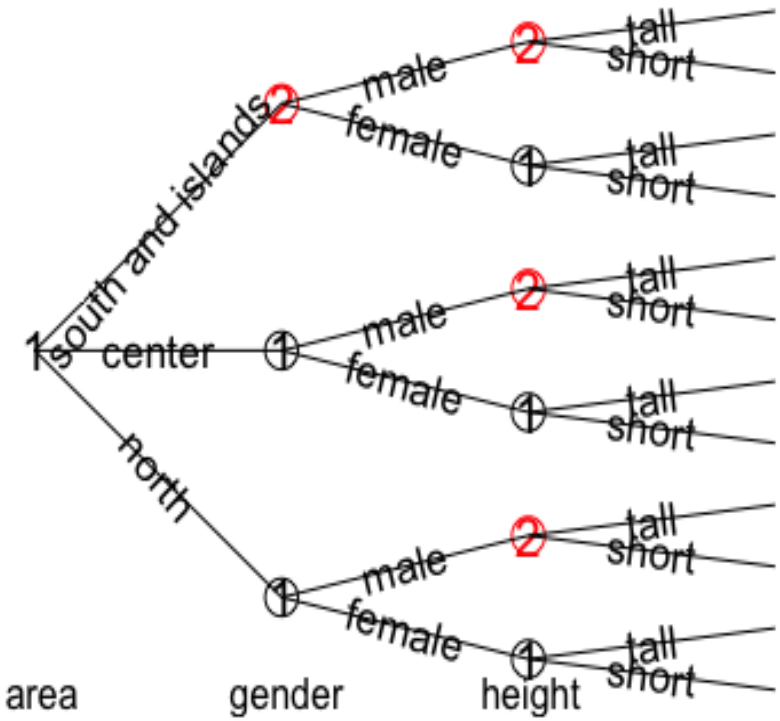
can be associated to the tree. Vertices associated to the same random variable are said to be in the same *stratum*.

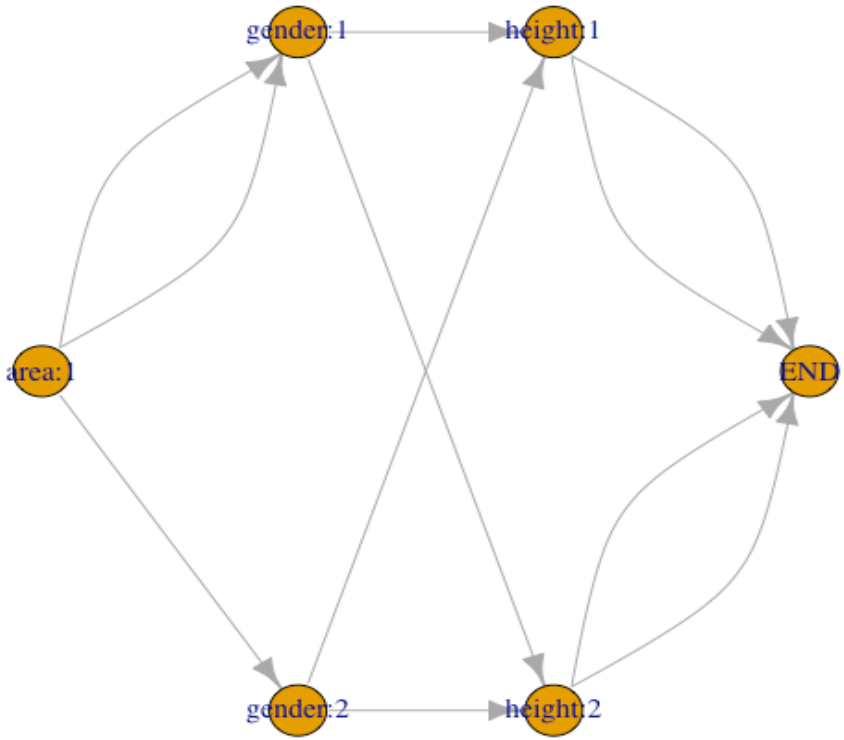
Stratified Staged Tree

A staged tree is called *stratified* if it is $\mathbf{X}$-compatible and if all non-leaf vertices in the same stage are in the same stratum.

From Stratified Staged Tree to CEG

- Staged trees are very expressive and flexible;
- as the number of variables increases, they cannot succinctly visualize their staging;
- partial solution: *chain event graph (CEG)*.





R package ***stagedtrees***

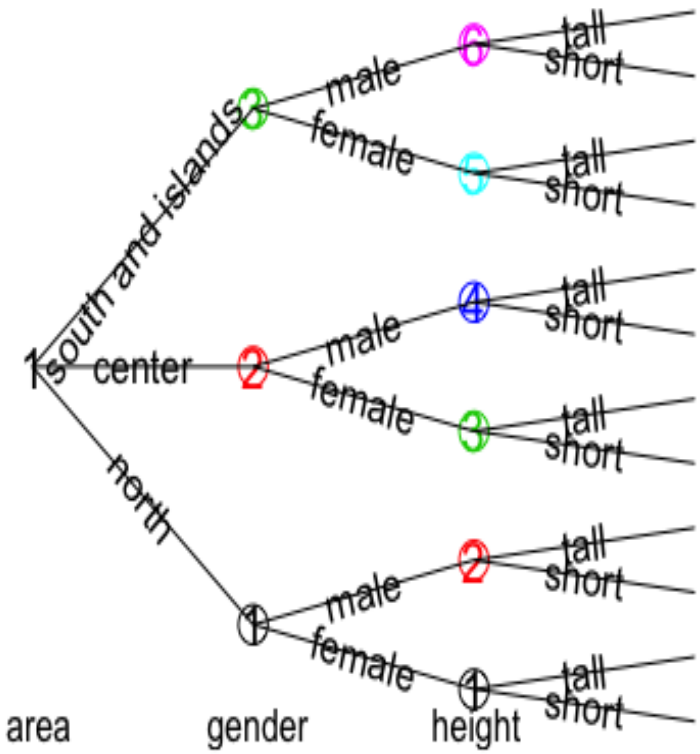
Various methods for fitting a stratified staged tree are implemented:

- hill-climbing,
- entropy,
- penalized log-likelihood (AIC and BIC index),
- distances and divergences among pair of discrete density distributions:
  - symmetrized Kullback-Leibler divergence,
  - symmetrized Renyi divergence,
  - L^p norm,
  - total variation,
  - Hellinger distance,
  - Bhattacharyya distance,
  - Chan-Darwiche distance,
  - user defined distance.

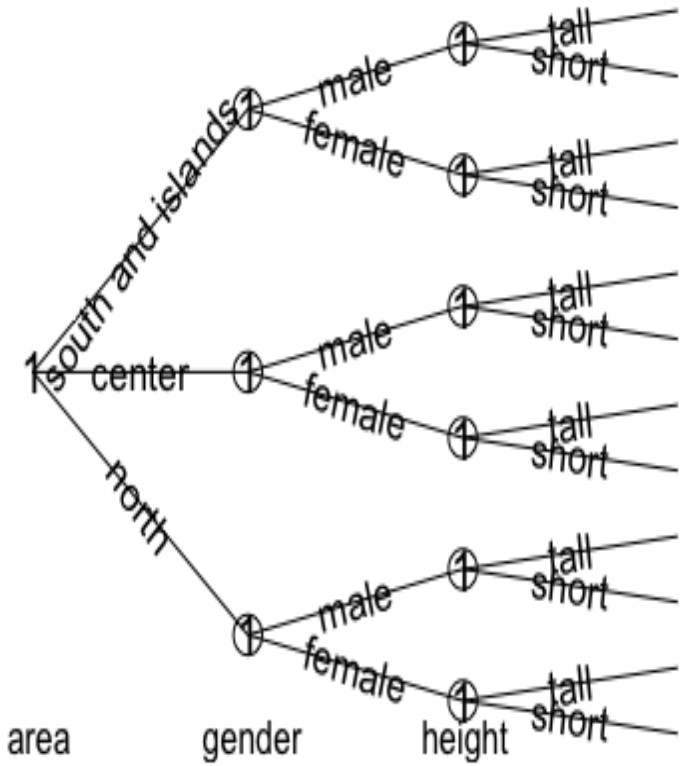
Stepwise Procedure

A backward and a forward procedure are implemented:

- **backward**: it starts from the saturated model, i.e. all edge labels are colored differently, and at each step it identifies the two nodes with the most marked independence relationship, until a stop criterion is reached;
- **forward**: it starts from the full-independence model, i.e. all edge labels have the same color, and at each step it identifies the two nodes with the most marked dependence relationship, until a stop criterion is reached.



(a) Saturated Model



(b) Full-Independence Model

R package **stagedtrees**

- Dataset format: *data.frame*, *table*, *list*.
- Initialize a *sevt* object: **full** or **indep**; order of variables can be specified with the *order* option;
  - **full** returns a sevt object where each vertex is in a different stage (a saturated model);
  - **indep** returns a sevt object where vertices in the same stratum are in the same stage (all variables are marginally independent of each other).

R package **stagedtrees**

Algorithms:

- hill-climbing:
  - **hc.sevt**
  - **bhc.sevt**
  - **fbhc.sevt**
  - **bhcr.sevt**

  for which we can specify the *score* to maximize (AIC, BIC, ...), the max number of iterations *max_iter* and if we want to display or not each step of algorithm (*trace*);

- distance-based: **bj.sevt**, where the *distance* and the threshold *thr* have to be specified.

Querying the Estimated Model

- **print** and **plot**;
- **summary**: returns for each stratum all the estimated stages, the number of paths and observations starting from the root that arrives to each stage and their corresponding probability distributions;
- **prob.sevt**: computes the probability (its logarithm if *log* = *TRUE*) of any event of interest (x);
- **compare.sevt**: checks if two staged trees are equal and returns a plot where nodes in different stages are colored in red;
- **sample.sevt**: generates observations according to the stage probability distributions;
- **predict**: returns a vector of labels predicting the class specified by the option *class* over the observations specified by *newdata*; If *prob* = *TRUE*, it returns the probabilities (their logarithm if *log* = *TRUE*) of observing all the class labels;
- **get_stage**: returns the stage of a given *path* in a staged tree (*object*);
- **get_path**: returns the paths that starting from the root arrive at a given *stage*, for the specified variable (*var*) in a staged tree (*object*);
- **ceg.sevt**: constructs the CEG corresponding to a given staged tree (*object*);
- **subtree.sevt**: it constructs a subtree having as root the explicited *path*.

Creating a Staged Tree

```
devtools::install_github("gherardovarando/stagedtrees",
                         ref = "master", force = TRUE)
library(stagedtrees)

data("Titanic")
str(Titanic)

m.indep <- indep(Titanic, lambda = 0.5)
m.full <- full(Titanic, lambda = 0.5)
```

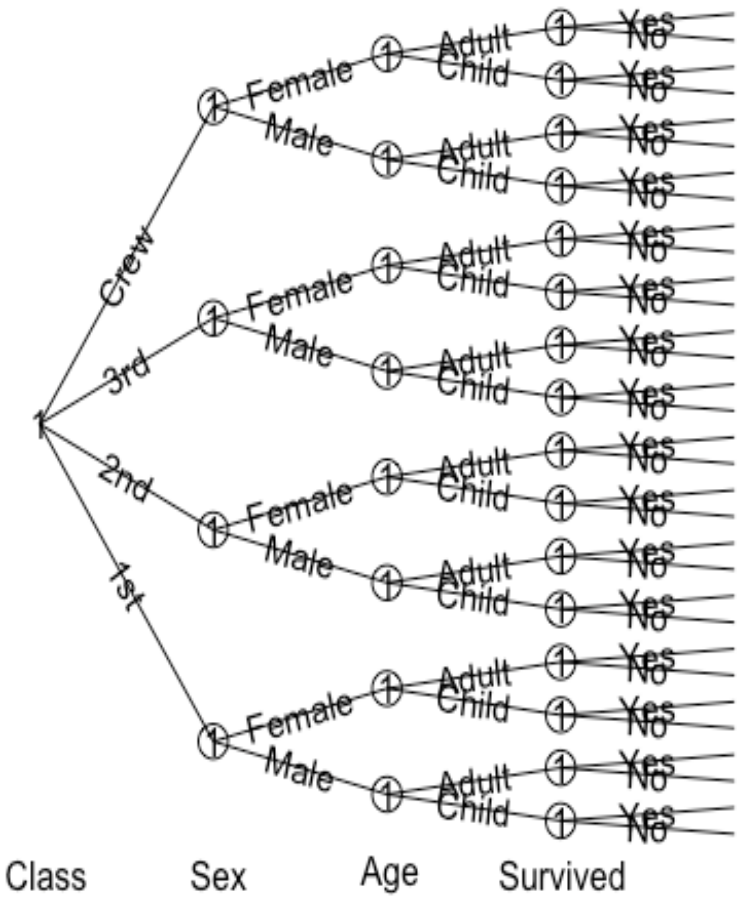
Structural Learning Algorithms

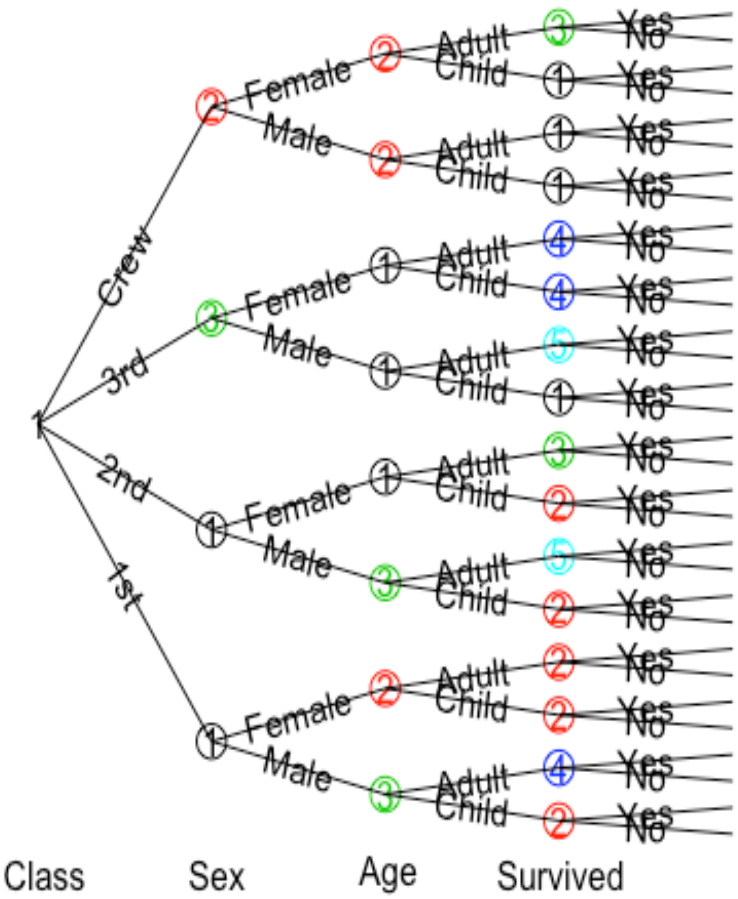
```
mod1 <- hc.sevt(m.indep)
mod2 <- bj.sevt(m.full, thr = 0.1)
mod2 <- stndnaming.sevt(mod2)

par(mfrow = c(1, 2))

plot(m.indep)
text(m.indep, y = -0.05, xlim = c(0.01, 1))
plot(mod1, col = "stages")
text(mod1, y = -0.05, xlim = c(0.01, 1))
```

Structural Learning Algorithms

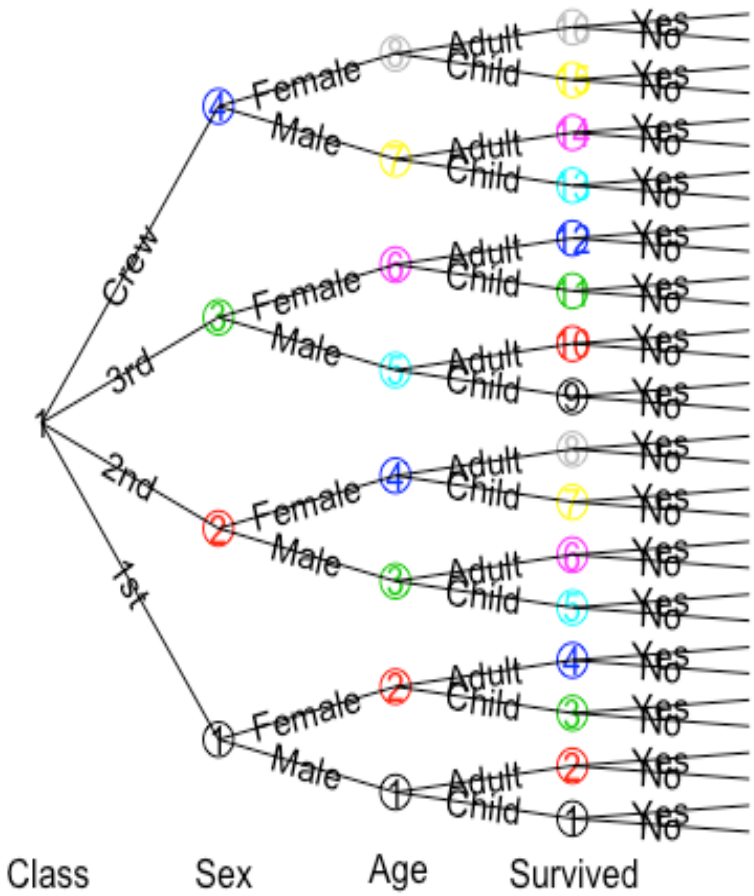
```
mod1 <- hc.sevt(m.indep)
mod2 <- bj.sevt(m.full, thr = 0.1)
mod2 <- stndnaming.sevt(mod2)

par(mfrow = c(1, 2))

plot(m.indep)
text(m.indep, y = -0.05, xlim = c(0.01, 1))
plot(mod1, col = "stages")
text(mod1, y = -0.05, xlim = c(0.01, 1))

plot(m.full)
text(m.full, y = -0.05, xlim = c(0.01, 1))
plot(mod2, col = "stages")
text(mod2, y = -0.05, xlim = c(0.01, 1))
```

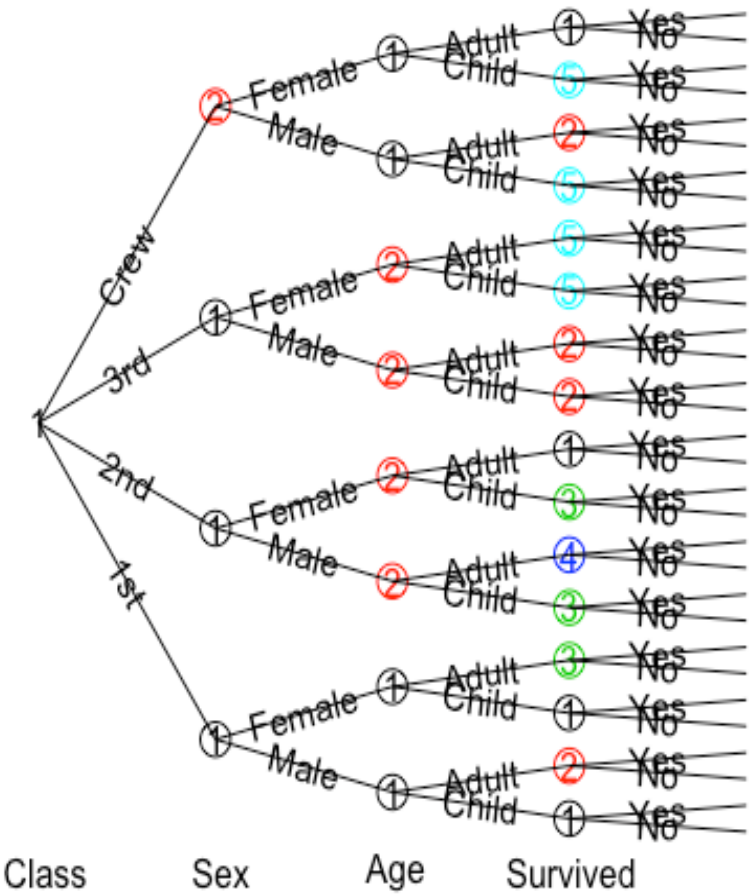
m.indep and *mod1*





m.full and *mod2*

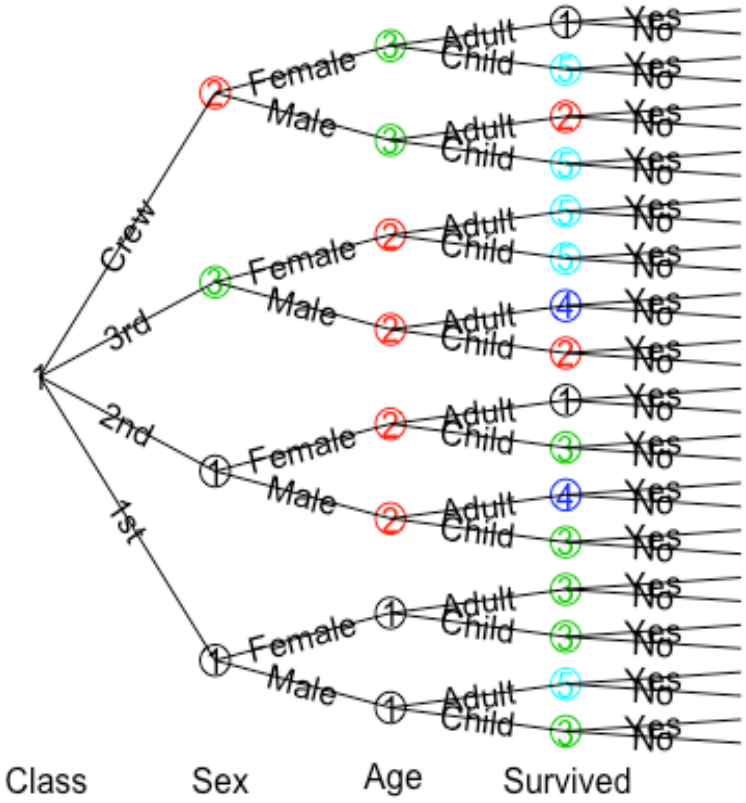


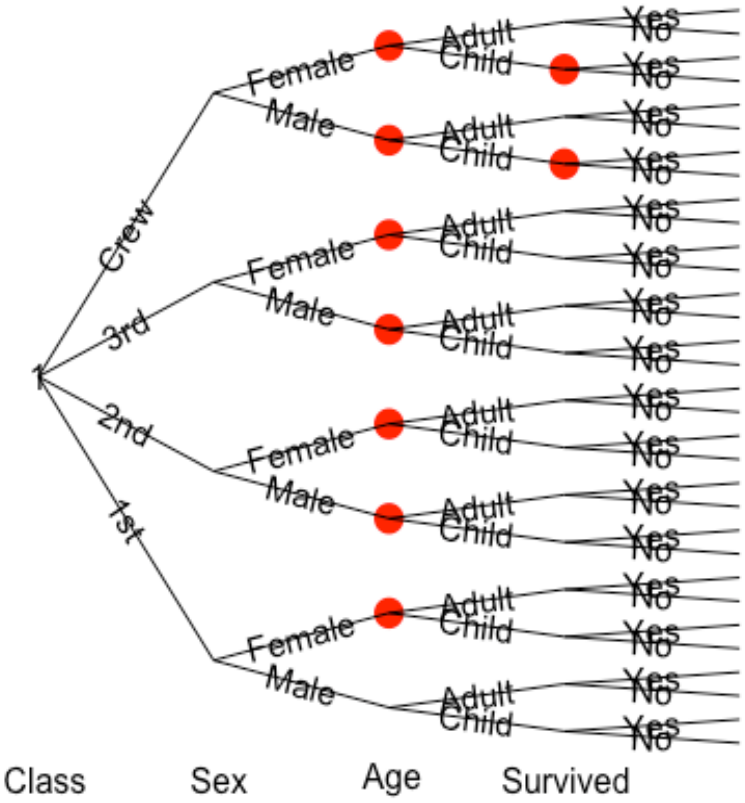


Re-fitting a Staged Tree

```
mod3 <- hc.sevt(mod2)

par(mfrow = c(1, 1))
plot(mod3, col = "stages")
text(mod3, y = -0.015, xlim = c(0.01, 1))

compare.sevt(mod1, mod3, plot = TRUE)
text(mod3, y = -0.015, xlim = c(0.01, 1))
```

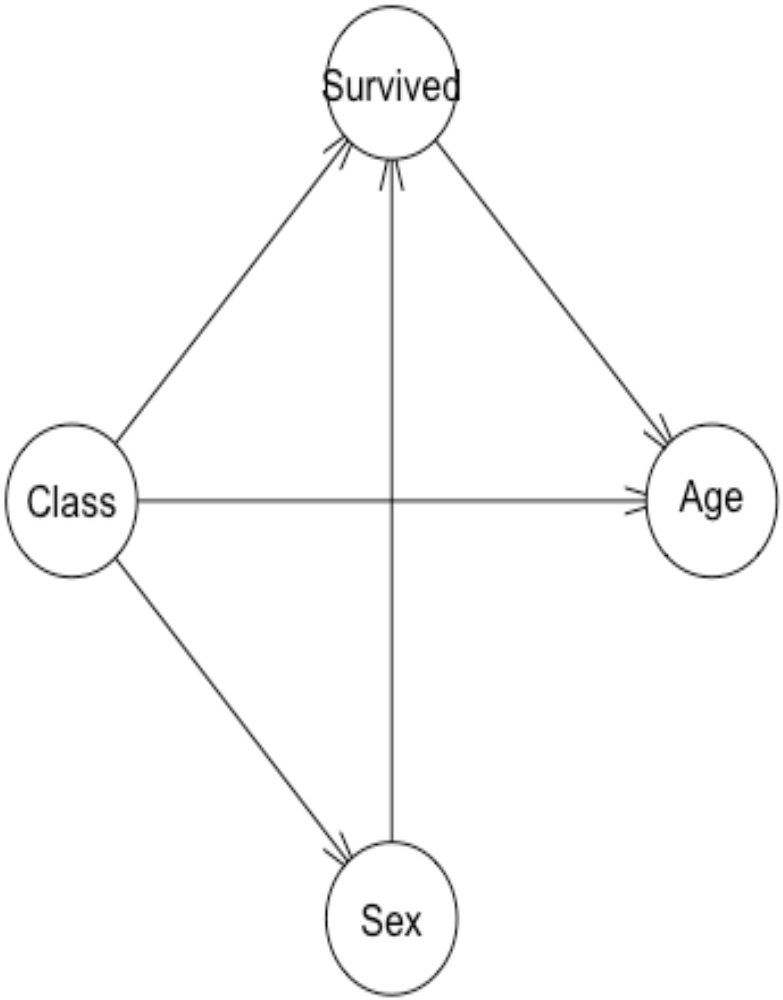
mod3 and *compare.sevt* between *mod1* and *mod3*

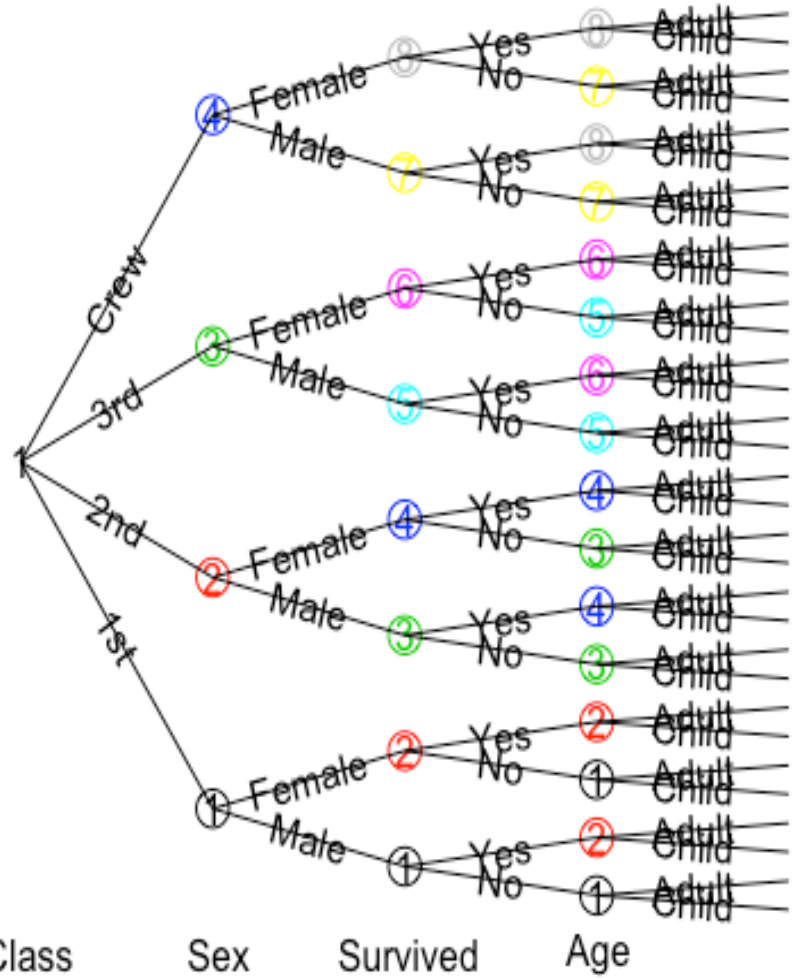




Bayesian Network as Staged Tree

```
tit <- as.data.frame(Titanic)
tit <- tit[rep(row.names(tit), tit$Freq), 1:4]

library(bnlearn)
mod.bn <- bnlearn::hc(tit)
plot(mod.bn)

mod.bn <- bn.fit(mod.bn, tit)
bn.tree <- fit.sevt(full(mod.bn, lambda = 0.5), data = tit)
plot(bn.tree)
text(bn.tree, y = -0.05, xlim = c(0.01, 1))
```


Survived
Class
Age
Sex

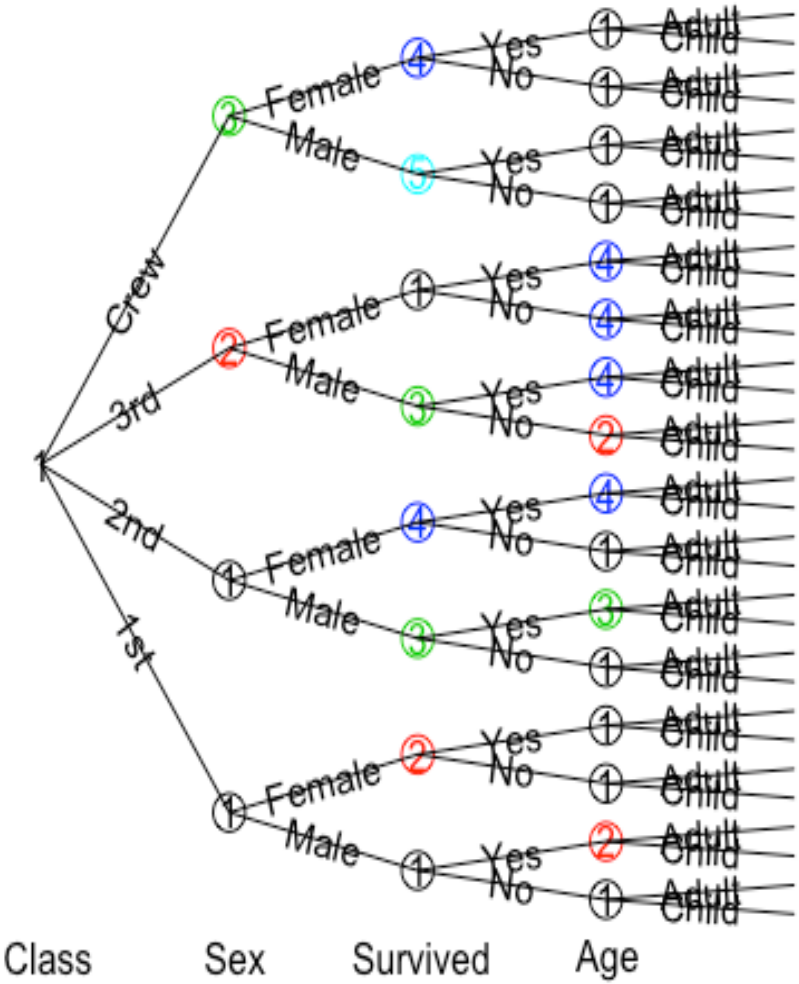

Crew
3rd
2nd
1st
Female
Male
Yes
No
Adult
Child
Class
Sex
Survived
Age

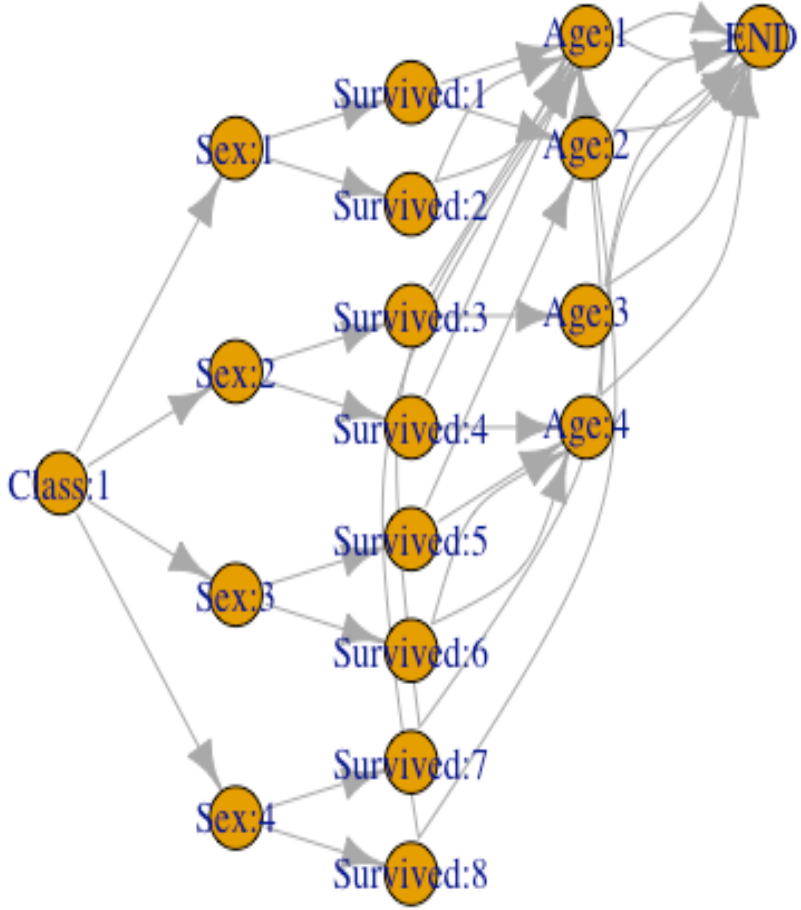
Refining a BN to a Staged Tree & Model Comparison

```
mod4 <- hc.sevt(bn.tree)
mod4 <- stndnaming.sevt(mod4)

results <- cbind(df = AIC(mod1, mod2, mod3, bn.tree, mod4)[, 1],
                 logLik = c(mod1$ll, mod2$ll, mod3$ll, bn.tree$ll, mod4$ll),
                 AIC = AIC(mod1, mod2, mod3, bn.tree, mod4)[, 2],
                 BIC = BIC(mod1, mod2, mod3, bn.tree, mod4)[, 2])

rownames(results) <- c("mod1", "mod2", "mod3", "bn.tree", "mod4")
#             df     logLik       AIC       BIC
# mod1        14   -5167.385   10362.77   10442.52
# mod2        12   -5192.397   10408.79   10477.15
# mod3        14   -5168.066   10364.13   10443.89
# bn.tree     23   -5162.628   10371.26   10502.28
# mod4        15   -5158.697   10347.39   10432.84
```

From Staged Tree to CEG

```
library(igraph)
ceg <- ceg.sevt(mod4)
A <- ceg2adjmat(ceg)
gr <- graph_from_adjacency_matrix(A)
lay <- layout.reingold.tilford(gr)

par(mfrow = c(1, 2))

plot(mod4)
text(mod4, y = -0.015, xlim = c(0.01, 1))
plot.igraph(gr, layout = -lay[, 2:1])
```


Crew
3rd
2nd
1st
Female
Male
Yes
No
Adult
Child
Class
Sex
Survived
Age


Class:1
Sex:1
Sex:2
Sex:3
Sex:4
Survived:1
Survived:2
Survived:3
Survived:4
Survived:5
Survived:6
Survived:7
Survived:8
Age:1
Age:2
Age:3
Age:4
END

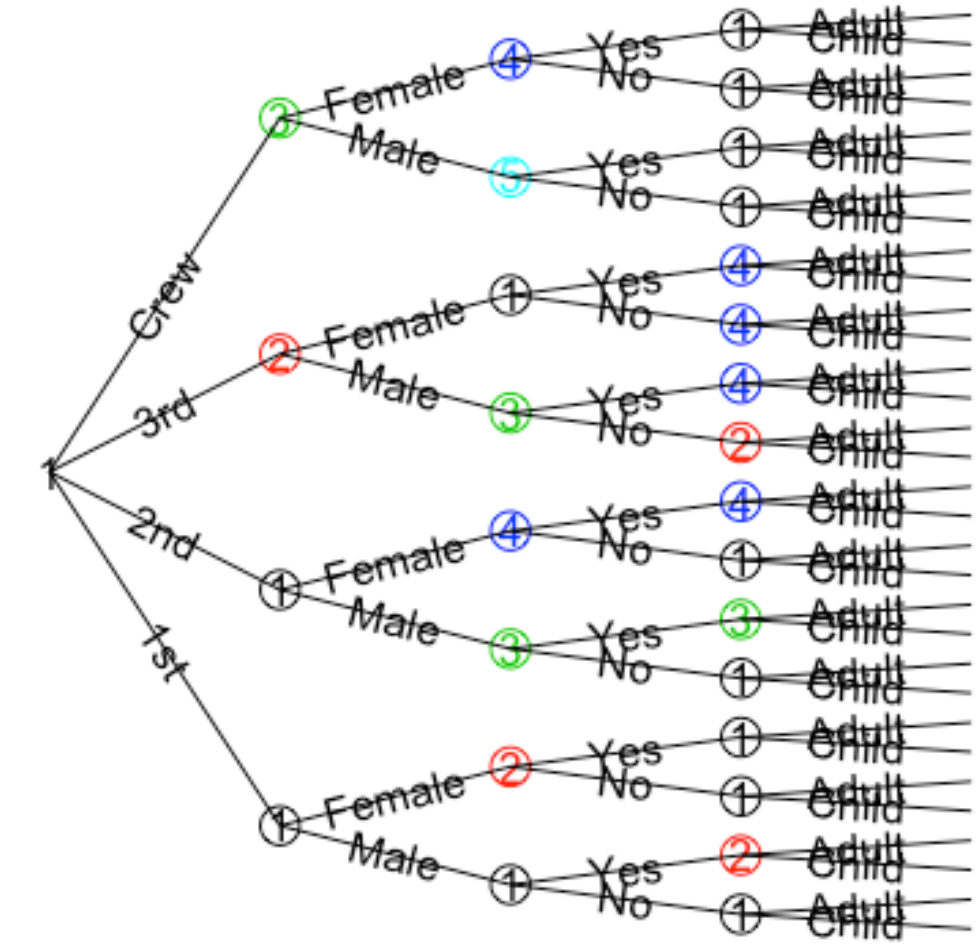
Querying the Model: *subtree.sevt*

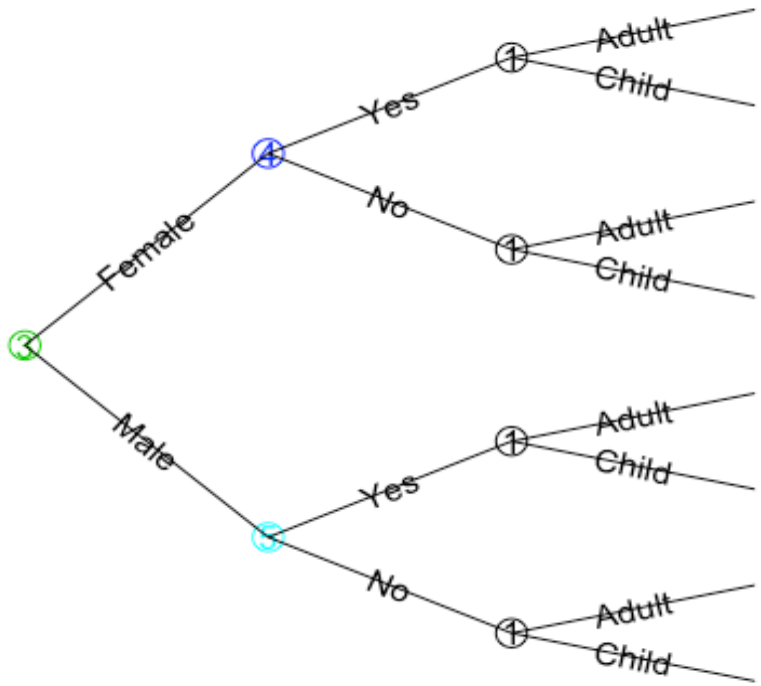
```
subtree.crew <- subtree.sevt(mod4, path = c("Crew"))
```

```
plot(mod4)
text(mod4, y = -0.05, xlim = c(0.01, 1))

plot(subtree.crew, col = "stages")
text(subtree.crew, y = -0.05, xlim = c(0.01, 1))
```

```
summary(mod4)
```

mod4 and *subtree.crew*





Querying the Model: *get_path* and *get_stage*

```
get_path(mod4, var = "Survived", stage = "3")
#    Class  Sex
# 3    2nd   Male
# 5    3rd   Male

get_path(mod4, var = "Survived", stage = "4")
#   Class    Sex
# 4    2nd   Female
# 8   Crew   Female

get_path(mod4, var = "Age", stage = "1")
#     Class      Sex     Survived
# 1     1st     Male         No
# 3     1st   Female         No
# 4     1st   Female        Yes
# 5     2nd     Male         No
# 7     2nd   Female         No
# 13   Crew     Male         No
# 14   Crew     Male        Yes
# 15   Crew   Female         No
# 16   Crew   Female        Yes

get_stage(mod4, path = c("Crew", "Female"))
# [1] "4"
```

Querying the Model: *prob.sevt*

```
prob.sevt(mod4, x = c(Class = "Crew", Sex = "Female"))
# [1] 0.0104498

paths_grid <- expand.grid(mod4$tree[2:1])[, 2:1]
cbind(paths_grid,
      Stage = get_stage(mod4, paths_grid),
      Prob = round(prob.sevt(mod4, paths_grid), 4))
#    Class      Sex    Stage   Prob
# 1   1st      Male      1    0.0869
# 2   1st    Female      2    0.0608
# 3   2nd      Male      3    0.0762
# 4   2nd    Female      4    0.0533
# 5   3rd      Male      3    0.2317
# 6   3rd    Female      1    0.0891
# 7  Crew      Male      5    0.3916
# 8  Crew    Female      4    0.0104
```

Querying the Model: *sample.sevt*

```
set.seed(1234)
bootstrap <- sample.sevt(mod4, n = NROW(tit))

summary(tit[, names(mod4$tree)])
# Class          Sex        Survived      Age
# 1st :325   Male  :1731   No :1490   Child: 109
# 2nd :285   Female: 470   Yes: 711   Adult:2092
# 3rd :706
# Crew:885
summary(bootstrap)
# Class          Sex        Survived      Age
# 1st :312   Female: 487   No :1482   Adult:2087
# 2nd :268   Male  :1714   Yes: 719   Child: 114
# 3rd :726
# Crew:895
```

Querying the Model: *predict*

```
p1 <- predict(mod4, newdata = bootstrap,
              class = "Survived", prob = F)
# [1] No  No  No  No  Yes No
# Levels: No Yes
p2 <- predict(mod4, newdata = bootstrap,
              class = "Survived", prob = T)
#  No       Yes
# [1,] 0.77726218 0.2227378
# [2,] 0.84636664 0.1536334
# [3,] 0.84636664 0.1536334
# [4,] 0.84636664 0.1536334
# [5,] 0.02758621 0.9724138
# [6,] 0.84636664 0.1536334
p3 <- predict(mod4, newdata = bootstrap,
              class = "Survived", prob = T, log = T)
#  No         Yes
# [1,] -0.2519776 -1.50175990
# [2,] -0.1668026 -1.87318632
# [3,] -0.1668026 -1.87318632
# [4,] -0.1668026 -1.87318632
# [5,] -3.5904394 -0.02797385
# [6,] -0.1668026 -1.87318632
table(p1, bootstrap$Survived)
# p1      No   Yes
# No    1465   451
# Yes     17   268
```

Conclusions

- **stagedtrees** is an *R* package which provides a free implementation of staged trees and CEGs structures.
- **stagedtrees** is designed to support users in staged tree modelling of categorical experimental data and analizying the learned models to untagle complex dependence structures. It provides a set of utility functions to perform descriptive statistics and basic inference procedures.
- Various structure learning algorithms for *stratified staged trees* are currently implemented.
- Exploring the model space of non-stratified trees lies in the exponential explosion of its size with the number of variables. Fast euristic model search procedures are currently investigated.

References

- Federico Carli, Manuele Leonelli, Eva Riccomagno and Gherardo Varando. *https://CRAN.R-project.org/package=stagedtrees*. (2019).
- Collazo, R. A., Görgen, C., & Smith, J. Q. (2018). *Chain event graphs*. CRC Press.
- Smith, Jim Q., and Paul E. Anderson. *Conditional independence and chain event graphs*. Artificial Intelligence 172.1 (2008): 42-68.
- Barclay, Lorna M., Jane L. Hutton, and Jim Q. Smith. *Refining a Bayesian network using a chain event graph*. International Journal of Approximate Reasoning 54.9 (2013): 1300-1309.
- Thwaites, Peter A., and Jim Q. Smith. *A new method for tackling asymmetric decision problems*. International Journal of Approximate Reasoning 88 (2017): 624-639.
- Görgen, Christiane, et al. *Discovery of statistical equivalence classes using computer algebra*. International Journal of Approximate Reasoning 95 (2018): 167-184.
- Højsgaard, Søren, David Edwards, and Steffen Lauritzen. *Graphical models with R*. Springer Science & Business Media. (2012).