# View Package Documentation

Corresponding NYSOL Version: Ver. 1.2, 2.0

revise history:
October 6, 2014 : first release

August 3, 2015

# Contents

# Chapter 1

# Introduction

## 1.1 Abstract

This "View (眺)" package is composed of multiple data visualization commands. Visualization refers techniques to present data as a graph or chart or other forms of diagrams to communicate a message, when you examine the data, the information should play an important role.

This package contains 3 commands. The `mpie.rb` command to draw pie chart, `msankey.rb` command to draw sankey diagram, and `mgv.rb` command to convert graph data in generic form.

## 1.2 Install

The view package is installed as part of the the nysol package. For more information, please refer to the nysol installation instructions (http://www.nysol.jp/install).

# Chapter 2

# Visualization Commands

## 2.1 msankey.rb Draw sankey diagram

The sankey diagram is a technique to visualize Directed Acyclic Graph (DAG), in which the weight of edge is defined in proportion with the flow quantity between contact points. The diagram is typically used to visualize energy transfer between processes in power transmission network.

This visualization command calls to internal D3 library (Data-Driven Documents) to generate sankey diagrams (http://bost.ocks.org/mike/sankey/).

The input data is shown in Table 2.1, each edge and node pair and their corresponding value are shown in a row as CSV data. The sankey diagram is generated as one html file which can be displayed in the browser as shown in diagram (2.1).

The orientation of the graph starts from left to right, the first item specified at command parameter `f=` corresponds to the left, and the second item corresponds to the right. The color bar corresponds to the node, and the edge corresponds to the connecting points. Interval relaxation method is used to determine the output position of the node. For more details, please refer to the original URL above. Based on experiments, it consumes more time to draw 5 nodes $\times$ 10 level positions = 50 nodes.

In order to use this command, json library in nysol/mcmd library is required.

Table 2.1: Input data (Closed walk directed graph)

| node1 | node2 | val |
|---|---|---|
| a | b | 1 |
| a | c | 2 |
| a | d | 1 |
| b | c | 3 |
| b | d | 3 |
| c | f | 1 |
| c | e | 4 |
| d | e | 1 |
| e | f | 3 |

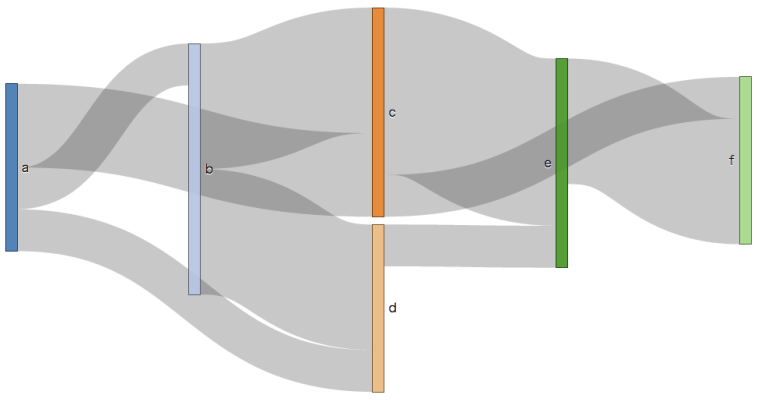



Figure 2.1: Sankey diagram

### 2.1.1 Format

```
msankey [i=] f= v= [o=] [t=] [T=] [--help]
```

| | |
|---|---|
| `i=` | : Edge data file |
| `f=` | : Two node field names on edge data |
| `v=` | : Field name of edge weight |
| `o=` | : Output file (HTML file) |
| `t=` | : Specify title |
| `-T` | : Working directory (default:/tmp) |
| `--help` | : Show help |

### 2.1.2 Example

**例 1: Basic Example**

Example used in the previous description.

```
$ more dat1.csv
node1,node2,val
a,b,1
a,c,2
a,d,1
b,c,3
b,d,3
c,f,1
c,e,4
d,e,1
e,f,3
$ msankey.rb i=dat1.csv f=node1,node2 v=val o=output.html
$ head output.html
<!DOCTYPE html>
<html class="ocks-org do-not-copy">
<meta charset="utf-8">
<!--
<title>Sankey Diagram</title>
-->
<title></title>
<style>
```

## 2.2 mpie.rb Draw pie graph

The command draws a pie chart. By specifying the attributes assigned to x axis and y axis, users can create 1 dimensional or 2 dimensional pie matrix. The output of the graph is saved as HTML file which can be displayed in popular browsers.

The input data in CSV format is shown in Table 2.2. The fields that construct the pie chart can be specified at `f=` parameter. The attributes assigned to x-axis and y-axis can be specified in `k=` parameter. When one field is specified at `k=` parameter, one-dimensional pie chart matrix is created, if two fields are specified at `k=` parameter, two-dimensional pie chart matrix is created. When `k=` parameter is not specified, one pie chart is created.

The pie chart is created with the JavaScript library D3.js (Data-Driven Documents). Please refer to the official website of D3.js (`http://d3js.org/`) for more details. Note that Nysol/mcmd library is required for this command.

Table 2.2: Number of individuals by age and prefecture

| Pref | Age | Population |
|---|---|---|
| Nara | 10 | 310504 |
| Nara | 20 | 552339 |
| Nara | 30 | 259034 |
| Nara | 40 | 450818 |
| Nara | 50 | 1231572 |
| Nara | 60 | 1215966 |
| Nara | 70 | 641667 |
| Hokkaido | 10 | 310504 |
| Hokkaido | 20 | 252339 |
| Hokkaido | 30 | 859034 |
| Hokkaido | 40 | 150818 |
| Hokkaido | 50 | 9231572 |
| Hokkaido | 60 | 4215966 |
| Hokkaido | 70 | 341667 |

### 2.2.1 Format

```
mpie.rb [i=] f= v= [o=] [k=] [title=] [pr=] [cc=] [--help]
```

| | |
|---|---|
| `i=` | Input data file name (CSV format) |
| `f=` | Specify the field name of component.<br>Null data is ignored. |
| `v=` | Specify the field name of component ratio (item that determines the arc length of the pie chart).<br>Data is treated as zero if it consists of null value.<br>Values started with 0 is ignored. Values except numeric characters returns error. |
| `o=` | Output file name (HTML file) |
| `k=` | Specify two or less field names assigned to x-axis and y-axis.<br>Create one pie chart when this parameter is not specified.<br>Create a one-dimensional pie chart when one field is specified,<br>create two-dimensional pie chart when two fields are specified. |
| `title=` | Specify the title of the graph |
| `pr=` | Specify the radius of the pie chart (default is 160). |
| `cc=` | Specify the maximum number of pie charts to be displayed in 1 row (default is 5).<br>Specify for one-dimension pie chart matrix. |
| `--help` | Show help |

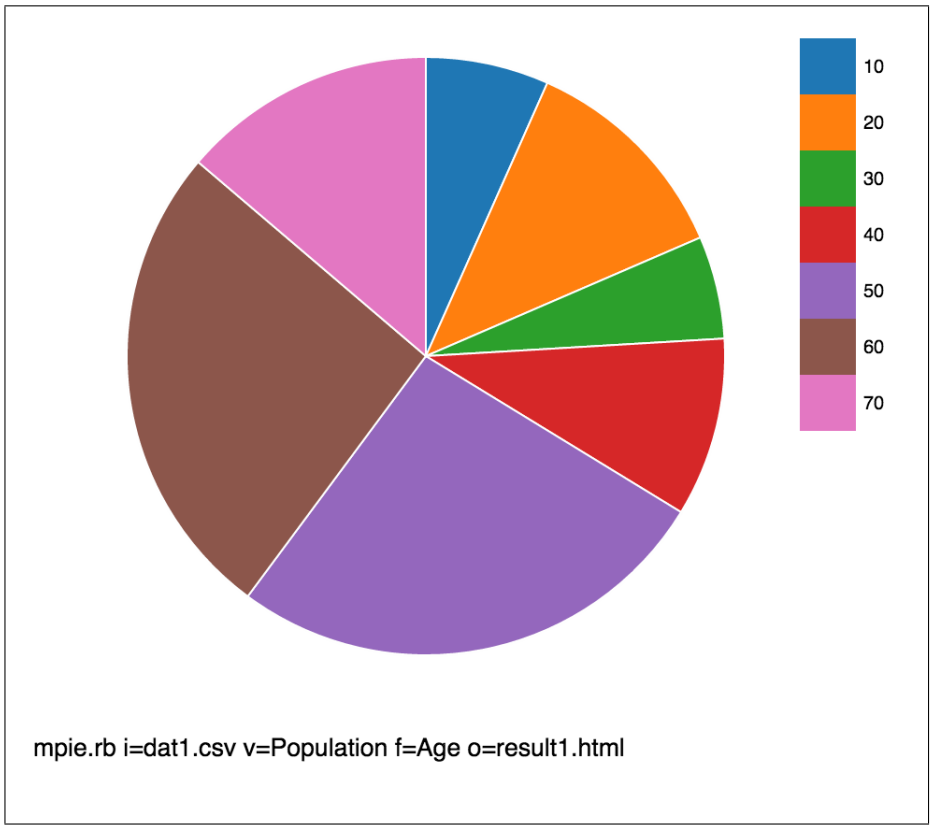
Noted that in `mpie.rb` command, the fields specified in `f=` parameter and `k=` parameter do not have automatic sorting function. It is necessary to sort the data prior to this command as needed.

### 2.2.2 Examples

**例 1: Draw a pie chart**

Using the data `dat1.csv`, set `Age` as the unit of composition, draw a one dimensional pie chart showing the population. The output of the pie chart can be viewed in the web browser. Note that in the pie chart, when you place the mouse cursor over the bar chart, details of the unit of composition is shown in a pop-up box. The graph can also be moved by dragging the mouse, and the size of the graph can be scaled by scrolling the mouse.

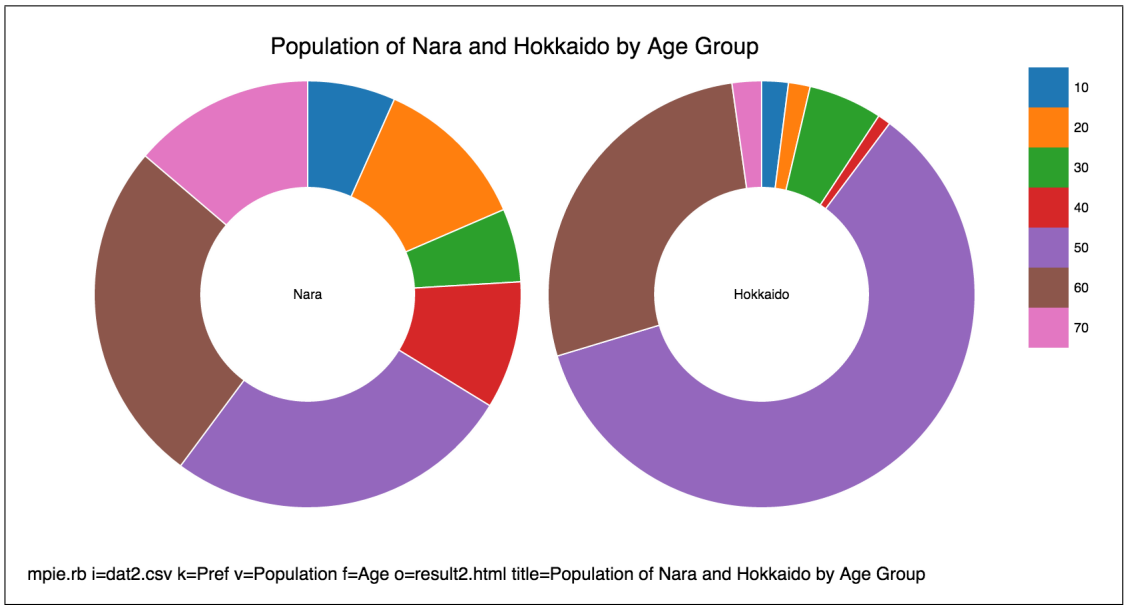
```
$ more dat1.csv
Age,Population
10,310504
20,552339
30,259034.5555
40,0450818
50,1231572
60,1215966
70,641667
$ mpie.rb i=dat1.csv v=Population f=Age o=result1.html
#END# /usr/bin/mpie.rb i=dat1.csv v=Population f=Age o=result1.html
$ head result1.html
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>
body {
  font: 10px sans-serif;
}
```




**例 2: Draw a one dimensional pie chart**

Using the data `dat2.csv`, set `Age` as the unit of composition, draw the pie chart showing the population by age. Specify `Pref` at `k=` parameter, which is designated on the x-axis as a one-dimensional pie chart. Specify the title of the graph at `title=` parameter.

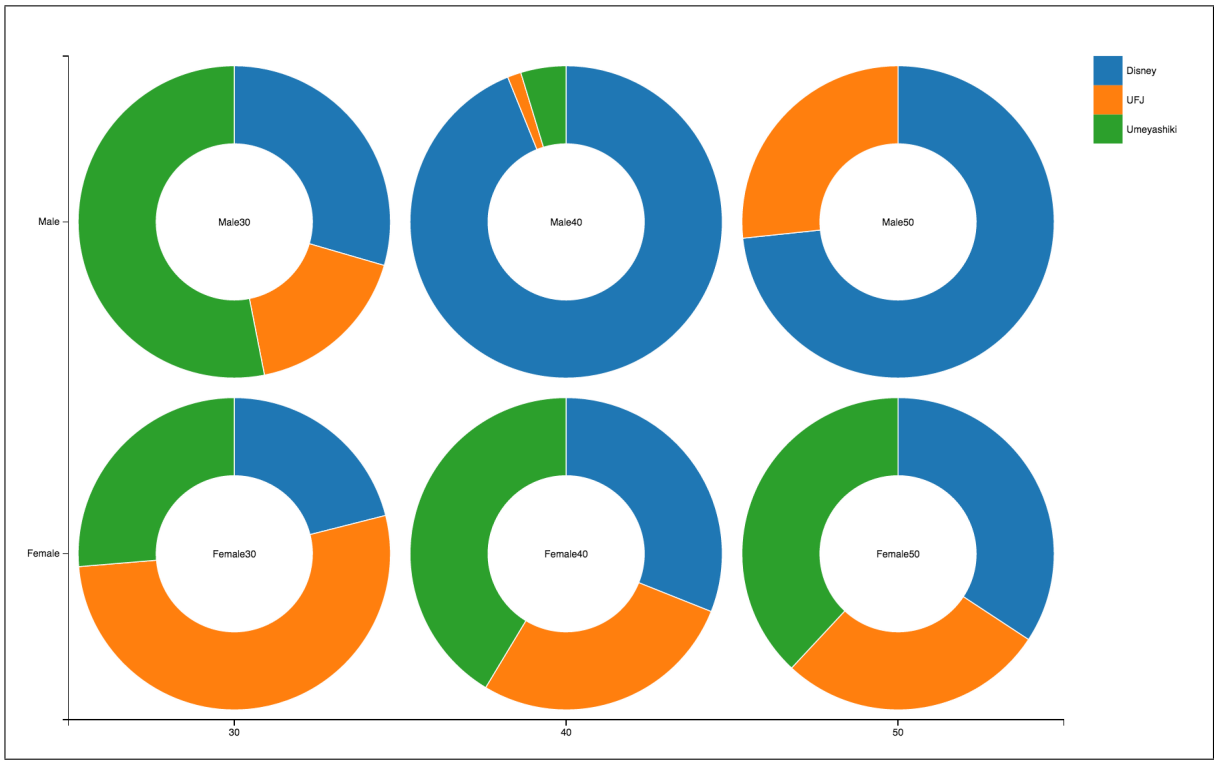
```
$ more dat2.csv
Pref,Age,Population
Nara,10,310504
Nara,20,552339
Nara,30,259034
Nara,40,450818
Nara,50,1231572
Nara,60,1215966
Nara,70,641667
Hokkaido,10,310504
Hokkaido,20,252339
Hokkaido,30,859034
Hokkaido,40,150818
Hokkaido,50,9231572
Hokkaido,60,4215966
Hokkaido,70,341667
$ mpie.rb i=dat2.csv k=Pref v=Population f=Age o=result2.html title='Population of Nara and
Hokkaido by Age Group'
#END# /usr/bin/mpie.rb i=dat2.csv k=Pref v=Population f=Age o=result2.html title=Population of Nara and H
$ head result2.html
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>
body {
  font: 10px sans-serif;
}
```




**例 3: Draw a two dimensional pie chart**

Using the data `dat3.csv`, set `ThemePark` as the unit of composition, draw the pie chart showing the theme parks by `Number` field, and specify the radius at 100 at the `pr=` parameter. Specify `Gender` and `Age` at `k=` parameter, which designates Gender on the x-axis, and Age on y-axis.

```
$ more dat3.csv
Gender,Age,ThemePark,Number
Male,30,Disney,100
Male,30,UFJ,59
Male,30,Umeyashiki,180
Male,40,Disney,200
Male,40,UFJ,3
Male,40,Umeyashiki,10
Male,50,Disney,110
Male,50,UFJ,40
Female,30,Umeyashiki,100
Female,30,Disney,80
Female,30,UFJ,200
Female,40,Disney,90
Female,40,UFJ,80
Female,40,Umeyashiki,120
Female,50,Disney,99
Female,50,UFJ,80
Female,50,Umeyashiki,110
$ mpie.rb i=dat3.csv k=Gender,Age v=Number f=ThemePark o=result3.html
#END# /usr/bin/mpie.rb i=dat3.csv k=Gender,Age v=Number f=ThemePark o=result3.html
$ head result3.html
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>
body {
  font: 10px sans-serif;
}
```


Disney
UFJ
Umeyashiki
Male
Female
Male30
Male40
Male50
Female30
Female40
Female50
30
40
50

## 2.3 mbar.rb Draw bar graph

The command creates a bar graph. By specifying the attributes assigned to x axis and y axis, users can create 1 dimensional or 2 dimensional bar matrix. The output of the graph is saved as HTML file which can be displayed in popular browsers.

The input data in CSV format is shown in Table 2.3. The fields that construct the bar graph can be specified at `f=` parameter. The attributes assigned to x-axis and y-axis can be specified in `k=` parameter. When one field is specified at `k=` parameter, one-dimensional bar graph matrix is created, if two fields are specified at `k=` parameter, two-dimensional bar graph matrix is created. When `k=` parameter is not specified, one bar graph is created.

The bar graph is created with the JavaScript library D3.js (Data-Driven Documents). Please refer to the official website of D3.js (`http://d3js.org/`) for more details. Note that Nysol/mcmd library is required for this command.

Table 2.3: Number of individuals by age and prefecture

| Pref | Age | Population |
|---|---|---|
| Nara | 10 | 310504 |
| Nara | 20 | 552339 |
| Nara | 30 | 259034 |
| Nara | 40 | 450818 |
| Nara | 50 | 1231572 |
| Nara | 60 | 1215966 |
| Nara | 70 | 641667 |
| Hokkaido | 10 | 310504 |
| Hokkaido | 20 | 252339 |
| Hokkaido | 30 | 859034 |
| Hokkaido | 40 | 150818 |
| Hokkaido | 50 | 9231572 |
| Hokkaido | 60 | 4215966 |
| Hokkaido | 70 | 341667 |

### 2.3.1 Format

```
mbar.rb [i=] f= v= [o=] [k=] [title=] [width=] [height=] [cc=] [--help]
```

| | |
|---|---|
| `i=` | Input data file name (CSV format) |
| `f=` | Specify the field name of component.<br>Null data is ignored. |
| `v=` | Specify the field name of component ratio (item that determines the height of the bar graph).<br>Data is treated as zero if it consists of null value.<br>Accepts minus, decimal values.<br>Values started with 0 is ignored. Values except numeric characters returns error. |
| `o=` | Output file name (HTML file) |
| `k=` | Specify two or less field names assigned to x-axis and y-axis.<br>Create one bar graph when this parameter is not specified.<br>Create a one-dimensional bar graph when one field is specified,<br>create two-dimensional bar graph when two fields are specified. |
| `title=` | Specify the title of the graph. |
| `width=` | Specify the width of drawing frame for the bar graph (default is 250, 1 bar graph is 600) . |
| `height=` | Specify the height of drawing frame for the bar graph (default is 250, 1 bar graph is 400). |
| `cc=` | Specify the maximum number of bars to be displayed in a row (default is 5) .<br>Specify for one-dimension pie chart matrix. |
| `--help` | Show help |

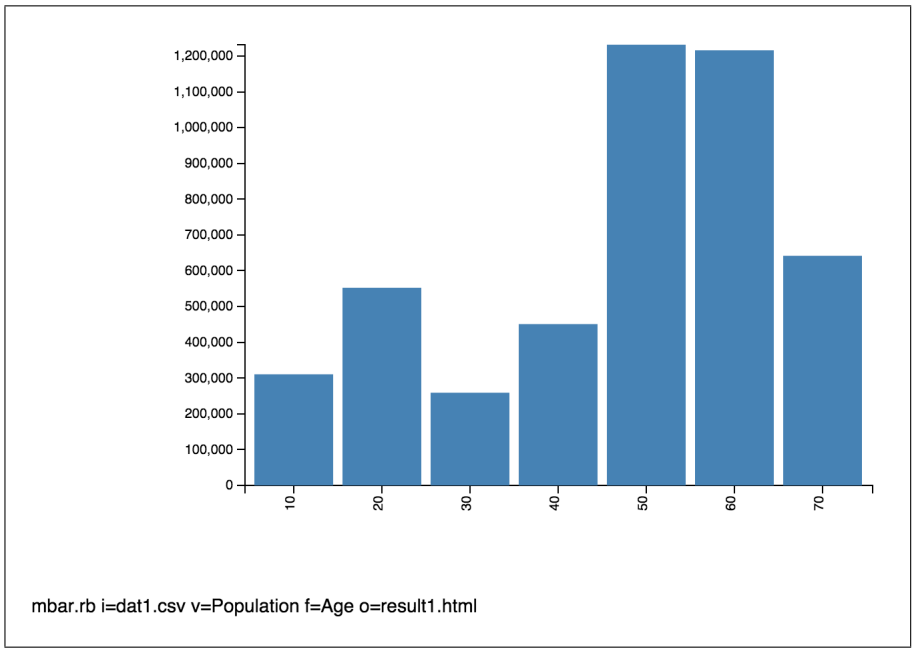
Noted that in `mbar.rb` command, the fields specified in `f=` parameter and `k=` parameter do not have automatic sorting function. It is necessary to sort the data prior to this command as needed.

### 2.3.2 Example

**例 1: Basic Example**

Using the data `dat1.csv`, set `Age` as the unit of composition, draw a one dimensional bar graph showing the population. The output of the bar chart can be viewed in the web browser. Note that in the bar chart, when you place the mouse cursor over the bar chart, details of the unit of composition is shown in a pop-up box. The graph can also be moved by dragging the mouse, and the size of the graph can be scaled by scrolling the mouse.

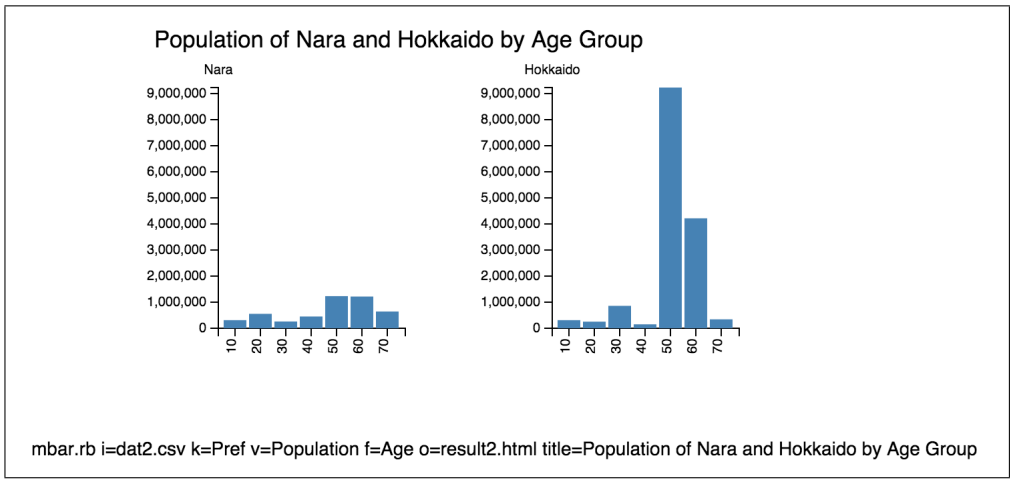
```
$ more dat1.csv
Age,Population
10,310504
20,552339
30,259034.5555
40,0450818
50,1231572
60,1215966
70,641667
$ mbar.rb i=dat1.csv v=Population f=Age o=result1.html
#END# /usr/bin/mbar.rb i=dat1.csv v=Population f=Age o=result1.html
$ head result1.html
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>
body {
  font: 10px sans-serif;
}
```




**例 2: Draw a one dimensional bar graph matrix**

Using the data `dat2.csv`, set `Age` as the unit of composition, draw the bar graph showing the population. Specify `Pref` at ¥verb—k=— parameter, which designates pref on the x-axis, extended horizontally as a one-dimensional bar graph. Specify the title of the graph at `title=` parameter.

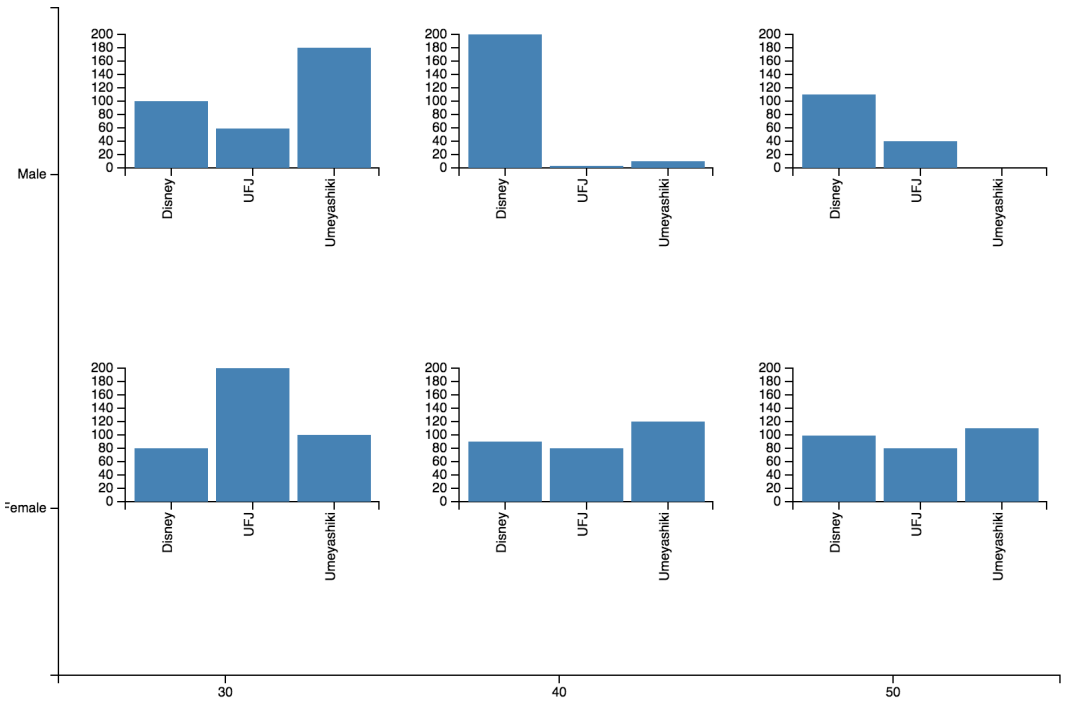
```
$ more dat2.csv
Pref,Age,Population
Nara,10,310504
Nara,20,552339
Nara,30,259034
Nara,40,450818
Nara,50,1231572
Nara,60,1215966
Nara,70,641667
Hokkaido,10,310504
Hokkaido,20,252339
Hokkaido,30,859034
Hokkaido,40,150818
Hokkaido,50,9231572
Hokkaido,60,4215966
Hokkaido,70,341667
$ mbar.rb i=dat2.csv k=Pref v=Population f=Age o=result2.html title='Population of Nara and Hokkaido by A
#END# /usr/bin/mbar.rb i=dat2.csv k=Pref v=Population f=Age o=result2.html title='Population of Nara and
$ head result2.html
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>
body {
  font: 10px sans-serif;
}
```




**例 3: Draw a two dimensional bar graph matrix**

Using the data `dat3.csv`, set `ThemePark` as the unit of composition, draw the bar graph showing the number item, and set the `width=` at 200, and `height=` at 150. Specify `Gender` and `Age` at `k=` parameter, which designates Gender on the x-axis extended horizontally, and Age on y-axis extended vertically.

```
$ more dat3.csv
Gender,Age,ThemePark,Number
Male,30,Disney,100
Male,30,UFJ,59
Male,30,Umeyashiki,180
Male,40,Disney,200
Male,40,UFJ,3
Male,40,Umeyashiki,10
Male,50,Disney,110
Male,50,UFJ,40
Female,30,Umeyashiki,100
Female,30,Disney,80
Female,30,UFJ,200
Female,40,Disney,90
Female,40,UFJ,80
Female,40,Umeyashiki,120
Female,50,Disney,99
Female,50,UFJ,80
Female,50,Umeyashiki,110
$ mbar.rb i=dat3.csv k=Gender,Age v=Number f=ThemePark o=result3.html
#END# /usr/bin/mbar.rb i=dat3.csv k=Gender,Age v=Number f=ThemePark o=result3.html
$ head result3.html
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>
body {
  font: 10px sans-serif;
}
```




```
mbar.rb i=dat3.csv k=Gender,Age v=Number f=ThemePark o=result3.html
```

## 2.4 mgv.rb Create graph data for Graphviz (DOT)

The graph data in CSV format is converted to a .dot file format for processing by Graphviz.

Graph processing commands (e.g. mpolishing) contained in Take package reads and generates graphs expressed in CSV format. Visualization (graphic image) helps users to understand attributes of the graph, such as the scale and density. By converting graph data into DOT plain text graph description language, the graphs can be visualized in softwares such as Graphviz (http://www.graphviz.org) and Gephi (http://www.gephi.org).

Graphviz is intended for the visualization of small graphs, thus, it may not be practical for large graphs with several hundred and thousands of vertices. It is recommended to use Gephi for drawing of large-scale graphs.

### 2.4.1 Format

```
mgv.rb [ni=] [nf=] [nv=] [nr=] [-nl] ei= ef= [ev=] [er=] [-el] [-d] [o=] [--help]
```

| | |
|---|---|
| `ni=` | : File name of vertex set |
| `nf=` | : Field name of vertex |
| `nv=` | : Field name of vertex attribute (size of vertex) |
| `nr=` | : Enlargement ratio when drawing graph. This parameter accepts integer values from 1 to 10. Default is set at 3 |
| `-nl` | : Specified value at `nv=` is added to the node name |
| `ei=` | : File name of edge set |
| `ef=` | : ID field name of beginning vertex and end vertex |
| `ev=` | : Field name of edge attribute (thickness of edge) |
| `er=` | : Enlargement ratio of edge when drawing graph. This parameter accepts integer values from 1 to 10. Default is set at 10. |
| `-el` | : Show width of edge specified at `ev=` |
| `-d` | : Specify directed graph |
| `o=` | : Output file name (.dot file) |
| `--help` | : Show help |

The input graph data may only be specified at `ei=` parameter as edge set CSV data. In order to include the attribute of vertices (size), the CSV data of the vertices set used can be specified at `ni=` parameter.

#### Display graph data in CSV format

One row represents one edge, which corresponds to the start of vertex and end of vertex stored in two fields.

```
node1,node2
A,B
B,C
C,A
C,D
E,D
```

#### Example of .dot format graph data

Contains information related to vertices and information of edges.

```
digraph G {
  edge [dir=none]

    n0 [label="A" height=0.5 width=0.75]
n1 [label="B" height=0.5 width=0.75]
n2 [label="C" height=0.5 width=0.75]
n3 [label="D" height=0.5 width=0.75]
n4 [label="E" height=0.5 width=0.75]


    n0 -> n1 [style="setlinewidth(1.0)"]
n1 -> n2 [style="setlinewidth(1.0)"]
n2 -> n0 [style="setlinewidth(1.0)"]
n2 -> n3 [style="setlinewidth(1.0)"]
n4 -> n3 [style="setlinewidth(1.0)"]

}
```

**Example to draw by Graphviz**

Graphviz GUI allows users to interact with graph data, the `dot` command installed with Graphviz allows for conversion to graphic (.png file). The visual of graph is shown in diagram 2.2.

```
$ dot -Tpng rsl1.dot > rsl1.png
$ open rsl1.png
```

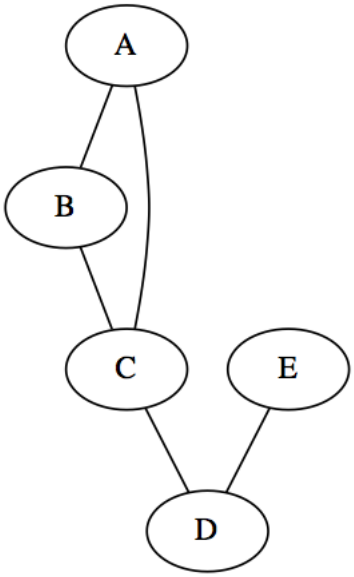


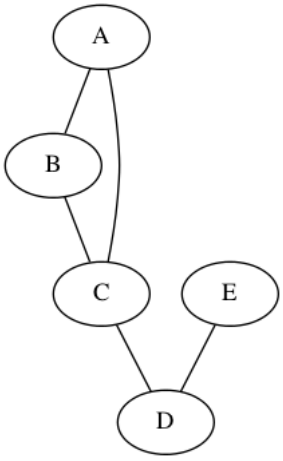

Figure 2.2: Example of drawing with Graphviz

### 2.4.2 Examples

**例 1: 基本例**

開始頂点と終了頂点からなる枝集合ファイルのみを与える。

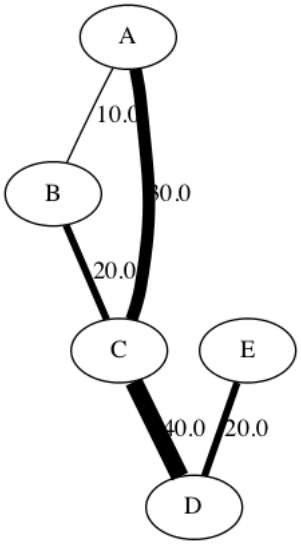
```
$ more edge1.csv
node1,node2
A,B
B,C
C,A
C,D
E,D
$ mgv.rb ei=edge1.csv ef=node1,node2 o=rsl1.dot
```




**例 2: 枝に属性 (太さ) を指定する例**

ev=パラメータで val 項目を属性 (太さ) として指定している。同時に-el オプションを付けることで、属性値もグラフに描画される。

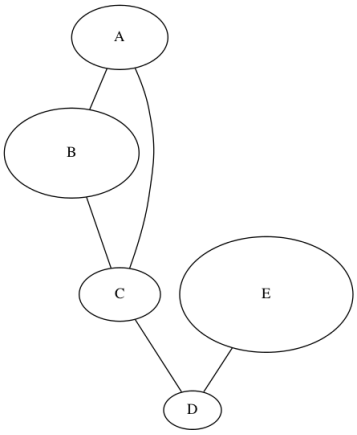
```
$ more edge2.csv
node1,node2,val
A,B,10
B,C,20
C,A,30
C,D,40
E,D,20
$ mgv.rb ei=edge2.csv ef=node1,node2 ev=val -el o=rsl2.dot
```



**例 3: 頂点に属性 (大きさ) を指定する例**

ni=パラメータで頂点集合ファイルを指定する。nv=パラメータで、val 項目を属性 (大きさ) として指定している。

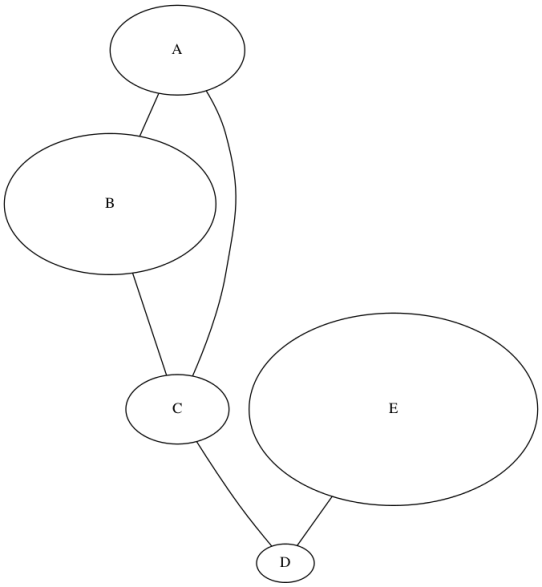
```
$ more node1.csv
node,val
A,10
B,15
C,8
D,5
E,20
$ more edge1.csv
node1,node2
A,B
B,C
C,A
C,D
E,D
$ mgv.rb ei=edge1.csv ef=node1,node2 ni=node1.csv nf=node nv=val o=rsl3.dot
```




**例 4: 頂点に属性 (大きさ) と拡大率を指定する例**

nr=パラメータで、ノードの拡大率を指定している。

```
$ more node1.csv
node,val
A,10
B,15
C,8
D,5
E,20
$ more edge1.csv
node1,node2
A,B
B,C
C,A
C,D
E,D
$ mgv.rb ei=edge1.csv ef=node1,node2 ni=node1.csv nf=node nv=val nr=5 o=rsl4.dot
```


A
B
C
E
D

## 2.5 mdtree.rb Draw decision tree model by PMML

The command visualize the decision tree model described in PMML (Predictive Model Markup Language) and generate a visual diagram in HTML format with D3 library. The command is created to visualize the output generated from `mbonsai` command, and it can be applied to decision tree models in PMML format generated by other software.

PMML defines description of values and categories of branch rules, however, it does not record the presence of sequence pattern in `mbonsai`. Therefore, `mbonsai` defines the extension tag to record the branching of sequence pattern, and mdtree.rb allow users to visual the expansion tag of the decision tree.

The following examples illustrates the series of decision trees constructed with `mbonsai`.

Table 2.4: Input data dat1.csv. Refer to examples for all data.

| Gender | VisitGap | PurchasePattern | Hospitalized |
|---|---|---|---|
| Male | 1.2 | ABCAAA | Yes |
| Male | 10.5 | BCDADD | Yes |
| Male | 0.5 | AAAA | No |
| Male | 2.0 | BBCC | No |
| Male | 3.1 | DEDDA | Yes |
| Female | 0.7 | CCCAA | No |
| Female | 1.5 | DDDEEE | Yes |
| Female | 2.6 | BACD | Yes |
| Female | 3.5 | ABBB | Yes |
| Female | 4.0 | DDDD | Yes |
| Female | 2.1 | DEDE | No |
| : | : | : | : |

Table 2.4 shows the training data for the construction of decision tree model with `mbonsai` command. The decision tree is saved as PMML file `model.pmml` in the directory specified at `O=`.

```
$ mbonsai c=Hospitalized n=VisitGap p=PurchasePattern d=Gender i=dat1.csv O=outdat
#END# kgbonsai O=outdat c=Hospitalized d=Gender i=dat1.csv n=VisitGap p=PurchasePattern; IN=81;
$ ls outdat
alpha_list.csv model.pmml     model.txt      model_info.csv param.csv      predict.csv
```

The following command can be used to visualize `model.pmml`. The output is rendered as `model.html` as shown in Figure 2.3.

```
$ mdtree.rb i=outdat/model.pmml o=model.html
#END# mdtree.rb i=outdat/model.pmml o=model.html;
$ open model.html
```

The maximum tree built in `mbonsai` is stored, the level of pruning of the decision tree can be controlled by specifying pruning degree at `alpha=`. When an integer value that is greater than 0 is specified at `alpha=`, when the decision tree is large, a lot of branches will be pruned. When `alpha=` is not specified and cross validation is not specified for `mbonsai`, `alpha=0.01` is specified. Yet if cross validation is specified, minimum misclassification rate is rendered.

Figure 2.4l the pruned decision tree with `alpha=0.1`.

```
$ mdtree.rb alpha=0.1 i=outdat/model.pmml o=model2.html
#END# mdtree.rb alpha=0.1 i=outdat/model.pmml o=model.html;
$ open model2.html
```

### 2.5.1 Collaboration with R

Many decision tree construction packages are available in the R statistical analysis package. The following section explains how to draw a decision tree with the `rpart` library.

We will build a decision tree with R script using two data sets - Iris data set (`iris`) and prostate cancer data set (`stagec`). The decision tree is built using `rpart` libraries and the model saved in PMML output.

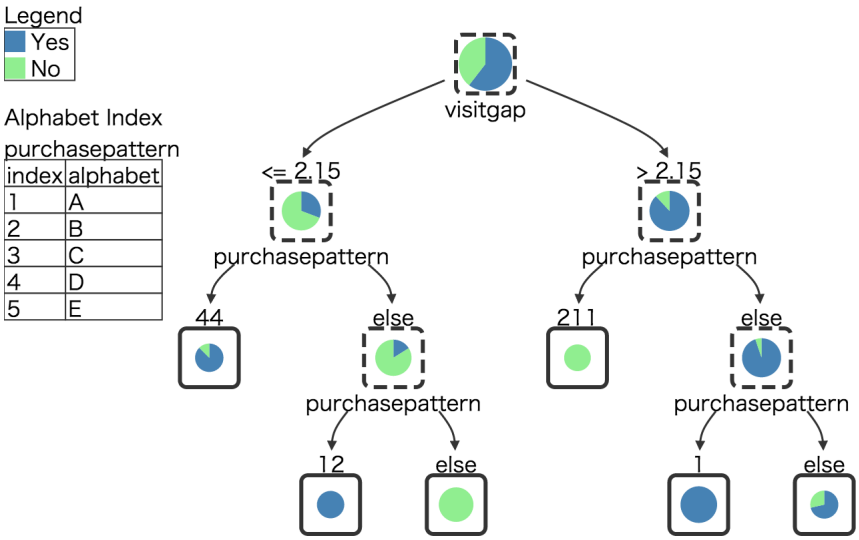



Figure 2.3: Draw decision tree with this command. The pie chart inside each node shows the class distribution (color for each class is shown in legend). The node with dotted line represents an intermediate node, the node with solid line represents a leaf node. The item name for the branch is shown below the node, and the branch rule is shown above the node. For example, on the first child level, if the Visit gap is below 2.15 on the left, and more than 2.15 on the right branch with no overlaps on both. When the node contains a sequence pattern, the branch which contained the pattern will be shown on the left, and the branch without the pattern is shown on the right. For instance, the left node in the second level from the top shows the purchase pattern containing "44". Furthermore, the characters of sequence patterns and each corresponding index are shown in the alphabet-index table in the upper left of the figure.

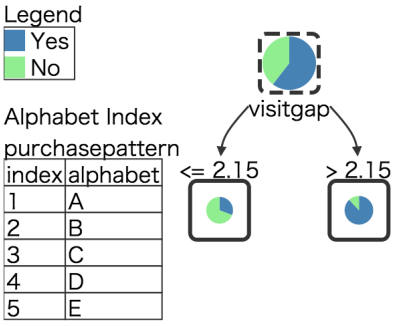



Figure 2.4: Decision tree created when pruning degree is set as alpha=0.1

This manual will not go into details on how to build decision tree model from the content in the data set. Note that PMML, XML, and rpart R libraries must be installed before proceeding with the following examples.

The program will generate a output from the decision tree of iris and prostate cancer will be saved as PMML file `model_r1.pmml` and `model_r1.pmm2`.

```
library(pmml)
library(rpart)
iris.rp=rpart(Species~.,data=iris)
sink("model_r1.pmml")
pmml(iris.rp)
sink()

stagec$progstat <- factor(stagec$pgstat, levels = 0:1, labels = c("No", "Prog"))
cfit <- rpart(progstat ~ age + eet + g2 + grade + gleason + ploidy, data = stagec, method = "class")
sink("model_r2.pmml")
pmml(cfit)
sink()
```

After obtaining two PMML files, we will follow the procedure for drawing a decision tree. The decision trees are shown in diagrams 2.5 and 2.5.

```
$ mdtree.rb i=model_r1.pmml o=out_r1.html
#END# mdtree.rb i=model_r1.pmml o=outl_r1.html;
$ mdtree.rb i=model_r2.pmml o=/out_r2.html
#END# mdtree.rb i=model_r2.pmml o=out_r2.html;
$ open model1_r1.html
$ open model1_r2.html
```

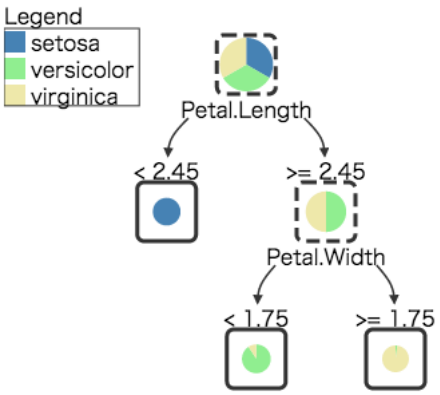



Figure 2.5: Decision tree of Iris dataset

### 2.5.2 Format

`mdtree.rb i= o= [alpha=] [--help]`

| | |
|---|---|
| `i=` | : PMML file of decision tree model |
| `o=` | : Output file (HTML file) |
| `alpha=` | : Specify the pruning degree (more branches are pruned when pruning degree is a integer greater than 0). |
| | : when this is not specified, and cross validation is not specified for `mbonsai`, |
| | : the value will be set as 0.01. If cross validation is specified, model with the minimum misclassification rate is rendered. |
| | : This parameter is only valid for building decision trees with `mbonsai`. |
| `--help` | : Show help |

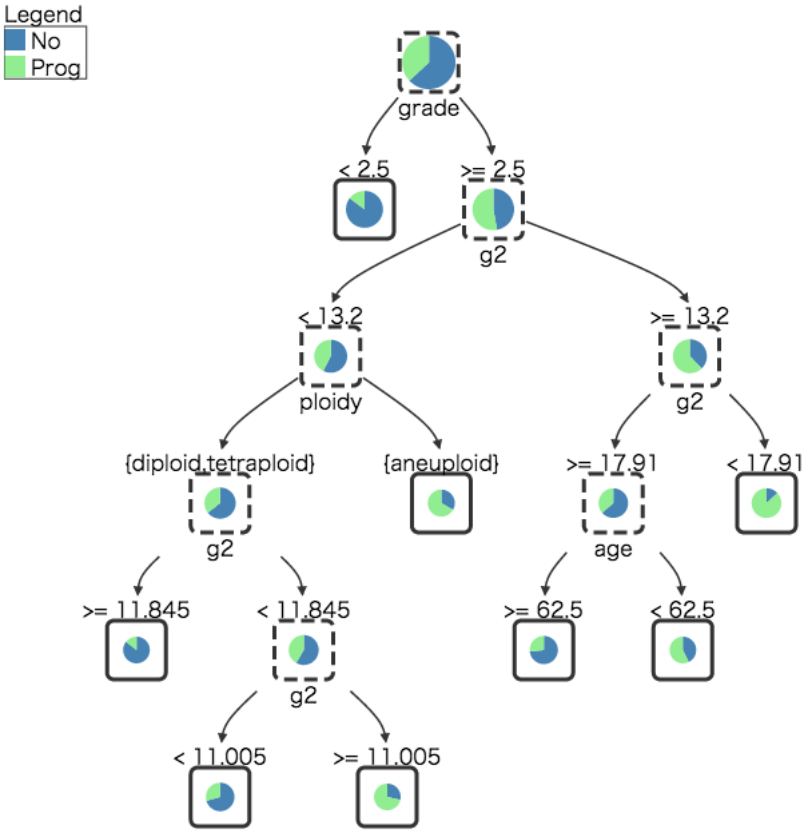



Figure 2.6: Decision tree of prostate cancer

### 2.5.3 Example

**例 1: Basic Example**

Example from the above section.

```
$ cat dat1.csv
gender,visitgap,purchasepattern,hospitalized
Male,1.2,ABCAAA,Yes
Male,10.5,BCDADD,Yes
Male,0.5,AAAA,No
Male,2.0,BBCC,No
Male,3.1,DEDDA,Yes
Female,0.7,CCCAA,No
Female,1.5,DDDEEE,Yes
Female,2.6,BACD,Yes
Female,3.5,ABBB,Yes
Female,4.0,DDDD,Yes
Female,2.1,DEDE,No
Male,1.2,ABCAAA,Yes
Male,10.5,BCDADD,Yes
Male,0.5,AAAA,No
Male,2.0,BBCC,No
Male,3.1,DEDDA,Yes
Male,0.7,CCCAA,No
Male,1.5,DDDEEE,No
Male,2.6,BACD,Yes
Male,3.5,ABBB,Yes
Male,4.0,DDDD,Yes
Male,2.1,DEDE,No
Male,1.2,ABCAAA,Yes
Male,10.5,BCDADDA,Yes
Male,0.5,AAAAA,No
Male,2.0,BBCCA,No
Male,3.1,DEDDA,Yes
Male,0.7,CCCAA,No
Male,1.5,ADDDEEE,Yes
Male,2.6,BACD,Yes
```

```
Male,3.5,ABBB,Yes
Male,4.0,DDDD,Yes
Female,2.1,DEDE,No
Female,1.2,ABCAAA,Yes
Female,10.5,BCDADD,Yes
Female,0.5,AAAA,No
Female,2.0,BBCC,No
Female,3.1,DEDDA,Yes
Female,0.7,CCCAA,No
Female,1.5,DDDEEE,Yes
Female,2.6,BACD,Yes
Female,3.5,ABBB,Yes
Female,4.0,DDDD,Yes
Female,2.1,DEDE,No
Female,1.2,ABCAAA,Yes
Female,10.5,BCDADD,Yes
Female,0.5,AAAA,No
Female,2.0,BBCC,No
Female,3.1,DEDDA,Yes
Female,0.7,CCCAA,No
Female,1.5,DDDEEE,Yes
Female,2.6,BACD,Yes
Female,3.5,ABBB,Yes
Female,1.0,DDDD,Yes
Female,2.5,DEDE,No
Female,2.5,ABBB,Yes
Female,1.0,DDDD,Yes
Female,1.1,DEDE,No
Female,2.2,ABCAAA,Yes
Female,10.5,BCDADD,Yes
Female,1.5,AAAA,No
Female,2.6,BBCC,No
Female,3.3,DEDDA,Yes
Female,1.7,CCCAA,No
Female,1.5,DDDEEE,Yes
Female,2.6,BACD,Yes
Female,3.9,ABBB,Yes
Female,4.5,DDDD,Yes
Female,2.1,DEDE,No
Female,3.9,BABB,Yes
Male,4.5,BAA,No
Female,2.1,DEDE,No
Male,3.9,BABB,Yes
Female,3.9,BABB,Yes
Male,4.5,BAA,No
Female,2.1,DEDE,No
Male,3.9,BABB,Yes
Female,3.9,BABB,Yes
Male,4.5,BAA,No
Female,2.1,DEDE,No
Male,3.9,BABB,Yes
$ mbonsai c=hospitalized n=visitgap p=purchasepattern d=gender i=dat1.csv O=outdat
ABCDE = 12345  *improved(errev:0.037037 *improved(errMin:0,leaf:1)
#END# kgbonsai O=outdat c=hospitalized d=gender i=dat1.csv n=visitgap p=purchasepattern
N=81
$ mdtree.rb i=outdat/model.pmml o=model1.html
#END# /usr/bin/mdtree.rb i=outdat/model.pmml o=model1.html
$ mdtree.rb alpha=0.1 i=outdat/model.pmml o=model2.html
#END# /usr/bin/mdtree.rb alpha=0.1 i=outdat/model.pmml o=model2.html
$ head model1.html
<html lang="ja">
<head>
  <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <style type="text/css">
          p.title { border-bottom: 1px solid gray
                g > .type-node > rect { stroke-dasharray: 10,5
hite
                g > .type-leaf > rect { stroke-width: 3px
                .edge path {  fill: none
                svg >.legend > rect { stroke-width: 1px
$ head model2.html
<html lang="ja">
<head>
  <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <style type="text/css">
          p.title { border-bottom: 1px solid gray
                g > .type-node > rect { stroke-dasharray: 10,5
hite
```

```
g > .type-leaf > rect { stroke-width: 3px
.edge path {  fill: none
svg >.legend > rect { stroke-width: 1px
```