



# Node states

---

*Version 1.3.0, by Giorgio Bianchini*

**Description:** Draws node states based on attributes.

**Module type:** Plotting

**Module ID:** 0512b822-044d-4c13-b3bb-bca494c51daa

This module can be used to draw shapes at each node highlighting the "state" of the node. This can consist in a pie chart, a rectangle (useful for trees where *Rectangular* coordinates are being used) or a wedge (useful for trees where *Circular* coordinates are being used).

## Parameters

---

Show on

**Control type:** Drop-down list

**Default value:** Tips

**Possible values:**

- Tips
- Internal nodes
- All nodes

This parameter determines on which nodes the states are shown. If the value is `Leaves`, the states are only shown for terminal nodes (nodes with no child nodes). If the value is `Internal nodes` the states are shown only for internal nodes (nodes which have at least one child). If the value is `All nodes`, states are shown for both leaves and internal nodes.

Exclude cartoon nodes

**Control type:** Check box

**Default value:** Checked

This parameter determines whether states are shown for nodes which have been "cartooned" or collapsed. If the check box is checked, states are not shown for nodes that have been "cartooned".

## Anchor

**Control type:** Drop-down list

**Default value:** Node

**Possible values:**

- Node
- Mid-branch
- Origin

This parameter determines the anchor for the centre of the state shape. If the value is `Node`, the centre of the shape is anchored to the corresponding node. If the value is `Mid-branch`, the centre of the shape is aligned with the midpoint of the branch connecting the node to its parent. If the value is `Origin`, the alignment depends on the current Coordinates module:

Coordinates module	Origin
<i>Rectangular</i>	A point corresponding to the projection of the node on a line perpendicular to the direction in which the tree expands and passing through the root node. Usually (i.e. if the tree is horizontal), this means a point with the same horizontal coordinate as the root node and the same vertical coordinate as the current node.
<i>Radial</i>	The root node.
<i>Circular</i>	The root node.

## Orientation reference

**Control type:** Drop-down list

**Default value:** Branch

**Possible values:**

- Horizontal
- Branch

This parameter determines the direction along which the offset of the centre of the shape from the anchor is computed. If the value is `Horizontal`, the offset `X` coordinate of the offset corresponds to an horizontal displacement and the `Y` coordinate to a vertical displacement; if the value is `Branch`, the `X` coordinate corresponds to a shift in the direction of the branch, while the `Y` coordinate corresponds to a shift in a direction perpendicular to the branch.

## Position

**Control type:** Point

**Default value:** ( 0, 0 )

This parameter determines how shifted from the anchor point the shape is. The X coordinate corresponds to the line determined by the [Orientation reference](#); the Y coordinate corresponds to the line perpendicular to this.

## Type

**Control type:** Drop-down list

**Default value:** Pie chart

**Possible values:**

- Pie chart
- Rectangle
- Wedge

This parameter determines the kind of plot that is used to display the states for the nodes. If the value is `Pie chart`, a small pie chart is drawn at each node, with each slice of the pie corresponding to a possible state. If the value is `Rectangle`, a rectangle is drawn for each node; the different states are represented by differently coloured areas in the triangle. If the value is `Wedge`, the result is similar to the `Rectangle` case, except that the rectangle is cut in the shape of circular ring sector.

## Width

**Control type:** Number spin box (by node)

**Default value:** 8

**Range:** [ 0,  $+\infty$  )

**Default attribute:** `StateWidth`

This parameter determines the width of the shape. This can be determined on a node-by-node basis using an attribute.

## Suggested width

**Control type:** Number spin box

If the Coordinates module is set as `Circular`, the [Type](#) is set as `Wedge`, and the [Anchor](#)

is set as `Origin`, this read-only control shows the suggested width for the wedge, based on the number of taxa in the tree and the position.

## Height

**Control type:** Number spin box (by node)

**Default value:** 8

**Range:**  $[0, +\infty)$

**Default attribute:** `StateHeight`

This parameter determines the height of the shape. This can be determined on a node-by-node basis using an attribute.

## Angle

**Control type:** Slider

**Default value:**  $0^\circ$

**Range:**  $[0^\circ, 360^\circ]$

This parameter determines the angle by which the coloured areas representing the states of the node are rotated within the shape. E.g. if the `Type` is `Rectangle` and this angle is  $0^\circ$ , the different states will be represented as vertical bands; if the angle is  $90^\circ$ , the states will be represented as horizontal bands.

## Stroke thickness

**Control type:** Number spin box


**Default value:** 0

**Range:**  $[0, +\infty)$

This parameter determines the thickness of the stroke surrounding the state shape.

## Stroke colour

**Control type:** Colour

**Default value:**  `#000000` (opacity: 100%)

This parameter determines the colour of the stroke surrounding the state shape.

## Total characters

**Control type:** Text box

This text box shows the total number of characters for which data is contained in the selected attribute.

## Enabled characters

**Control type:** Source code

This parameter contains a script that is used to determine which characters are enabled in the plot. Each character is associated to a boolean value - `true` means that the character is enabled, while `false` means that the character is not enabled. This can be changed more easily using the [Wizard edit enabled characters](#) button.

## Wizard edit enabled characters

**Control type:** Button

## State colours

**Control type:** Colour (by node)

**Default value:**  #DCDCDC (opacity: 100%)

**Default attribute:** (N/A)

This parameter is used to determine the colour associated to each state. While this uses a "Colour by node" control, the colours are actually determined based on the names of the states, rather than on attributes of the tree. The colours associated with each state can be changed (or additional states can be added) by modifying the formatter code for this parameter.

The default formatter used if the states are specified as strings is the following:

```
public static Colour? Format(object attribute)
{
    if (attribute is string state)
    {
        switch (state)
        {
            case "A":
                return Colour.FromRgb(213, 94, 0);
            case "B":
```

```

        return Colour.FromRgb(0, 114, 178);
    case "C":
        return Colour.FromRgb(0, 158, 115);
    case "0":
        return Colour.FromRgb(213, 94, 0);
    case "1":
        return Colour.FromRgb(0, 114, 178);
    case "2":
        return Colour.FromRgb(0, 158, 115);
    case "Y":
        return Colour.FromRgb(0, 158, 115);
    case "N":
        return Colour.FromRgb(220, 220, 220);
    default:
        return null;
    }
}
else
{
    return null;
}
}

```

The colour associated to each state can be changed by changing the RGB values in the `Colour.FromRgb` method calls. The possible states can be changed by modifying the case labels in the `switch` statement (e.g. replacing "A" with something else, or adding new case lines for additional states).

If the states are specified as numbers (e.g. in the case of continuous characters), a standard number formatter can be used, by setting an appropriate maximum and minimum value, and using a gradient to associate the colour to each state.

## Wizard edit state colours

**Control type:** Button

This button opens a window that can be used to specify the state colours using a graphical interface.

## Attribute

**Control type:** Attribute selector

**Default value:** Name

This parameter determines the attribute that contains the state associated to each node.

## Attribute type

**Control type:** Attribute type

**Default value:** String

**Possible values:**

- String
- Number

This parameter specifies the type of the attribute that contains the state associated to each node. You should use an attribute of type `String` for discrete character states, and an attribute of type `Number` for continuous character states.

## Missing data

**Control type:** Drop-down list

**Default value:** Ambiguous state

**Possible values:**

- Ambiguous state
- Unknown state

This parameter determines how nodes for which the specified attribute is not present are treated. If the value is `Ambiguous state`, in these nodes it is assumed that all the states are possible with equal probability. If the value is `Unknown state`, the state for the node is assumed to be unknown, and it is drawn using the default [State colour](#) (e.g. grey in the default settings).

## Further information

---

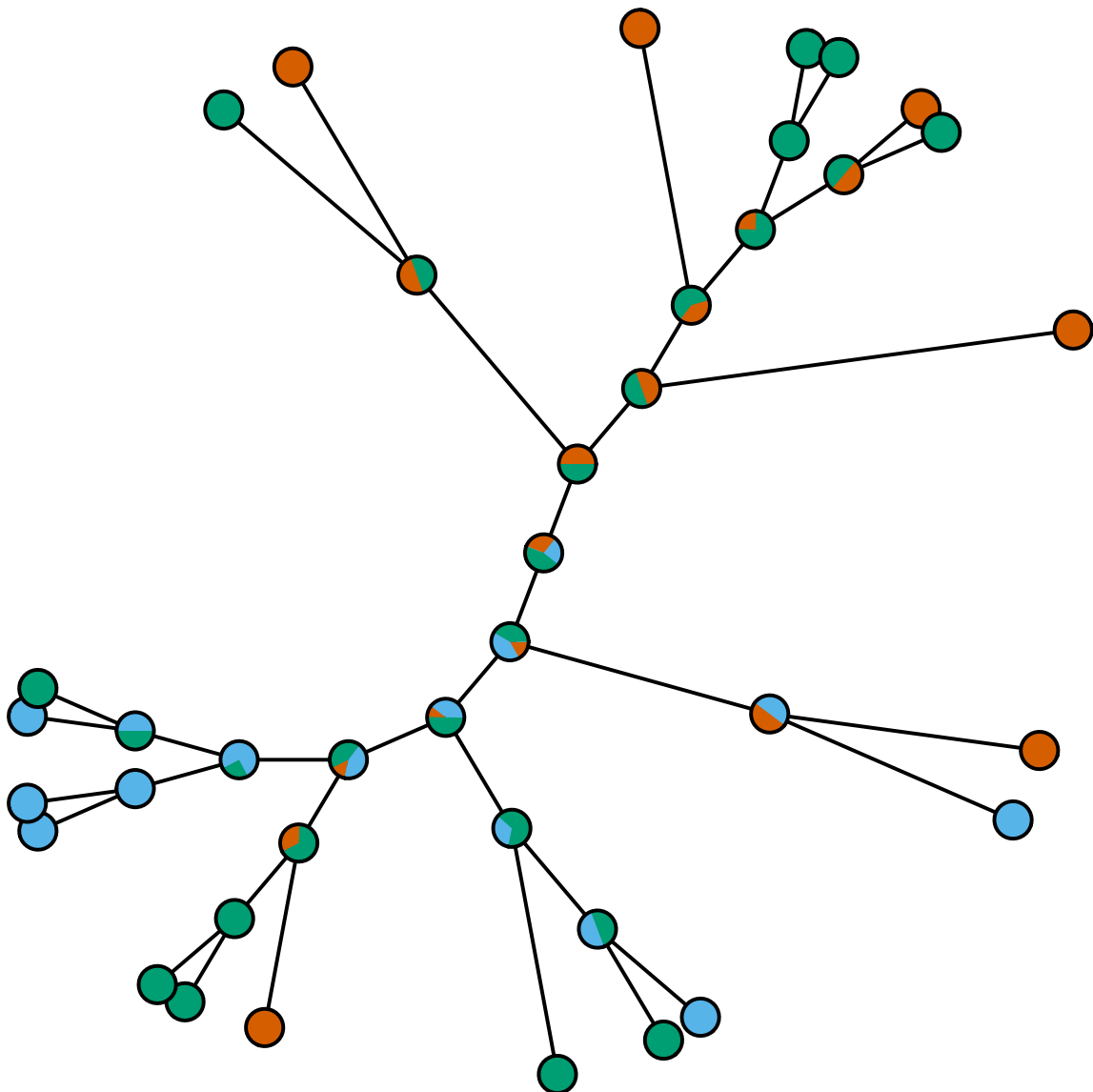
The format for the attribute that contains the node states should be as follows:

- If the attribute is a `Number`, the value of the attribute will be taken as the state of the node with weight 100%.
- If the attribute is a `String`, the value of the attribute can be:
  - a single alphanumerical character (e.g. `A`): in this case, the state of the node will correspond to this character with weight 100% and to any other state with weight 0%.
  - a list of `State:Weight` pairs, separated by commas and enclosed within curly brackets (e.g. `{A:0.5,B:0.5}`): in this case, the state of the node will be a mixture with the provided weights. Weights are normalised on a

per-node basis (e.g.  $\{A:0.66, B:0.33\}$  is equivalent to  $\{A:20, B:10\}$  ).

The most appropriate settings to use with this module to achieve the best results depend on the Coordinates module that is being used. Pie charts are appropriate to show states at each node of a tree drawn using any Coordinates module. Rectangles are most appropriate to show states at the leaf nodes for trees drawn with *Rectangular* coordinates. Wedges are most appropriate to show leaf states for trees drawn using *Circular* coordinates.

Here is an example of a tree that uses pie charts:



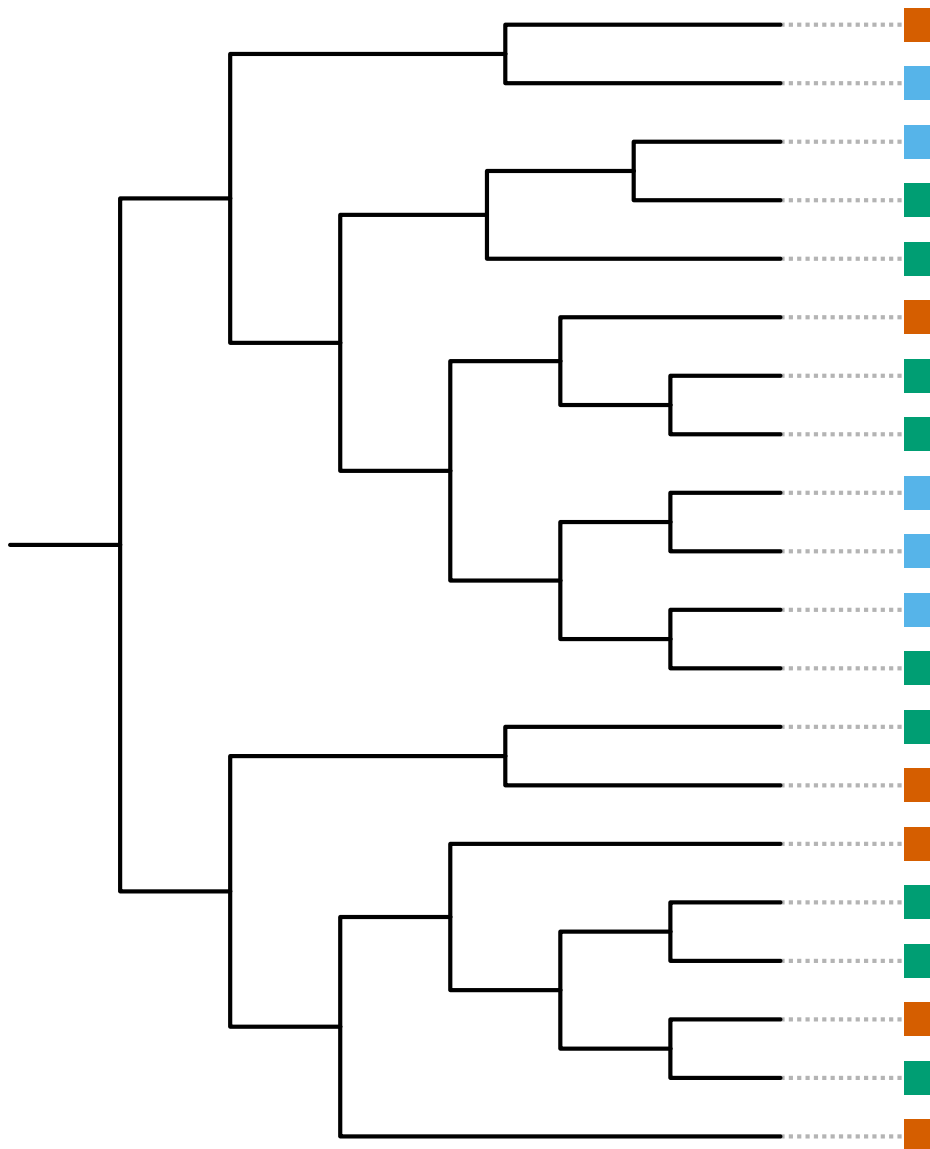
To obtain the best results for trees with *Rectangular* coordinates, you should:

- Set [Show on](#) to [Tips](#) .
- Set the [Anchor](#) to [Origin](#) .



- Set the [Orientation reference](#) to `Branch`.
- Set the [Position](#) to a sufficient displacement on the `X` axis (e.g. `(1000, 0)`).
- Set the [Type](#) to `Rectangle`.

Here is an example of a tree obtained with these settings:



To obtain the best results for trees with *Circular* coordinates, you should:

- Set [Show on](#) to `Tips`.
- Set the [Anchor](#) to `Origin`.
- Set the [Orientation reference](#) to `Branch`.
- Set the [Position](#) to a displacement on the `X` axis that is slightly larger than the outer radius of the circular coordinates (e.g. `(1100, 0)` if the outer radius is `1000`).
- Set the [Type](#) to `Wedge`.

- Set the [Width](#) to the [Suggested width](#).

Here is an example of a tree obtained with these settings:

