
Daft Documentation

Release 0.0.5

Daft Developers

Mar 17, 2021

Contents

1	Summary	1
2	Installation	3
3	Issues	5
4	Authors & Contributions	7
5	License	9
6	API	11
6.1	API	11
	Python Module Index	17
	Index	19

CHAPTER 1

Summary

Daft is a Python package that uses `matplotlib` to render pixel-perfect *probabilistic graphical models* for publication in a journal or on the internet. With a short Python script and an intuitive model-building syntax you can design directed (Bayesian Networks, directed acyclic graphs) and undirected (Markov random fields) models and save them in any formats that `matplotlib` supports (including PDF, PNG, EPS and SVG).

CHAPTER 2

Installation

Installing the most recent stable version of Daft should be pretty easy if you use `pip`:

```
pip install daft
```

Otherwise, you can download the source (`tar`, `zip`) and run:

```
python setup.py install
```

in the root directory.

Daft only depends on `matplotlib` and `numpy`. These are standard components of the scientific Python stack but if you don't already have them installed `pip` will try to install them for you but sometimes it's easier to do that part yourself.

CHAPTER 3

Issues

If you have any problems or questions, open an [“issue”](#) on Github.

CHAPTER 4

Authors & Contributions

Daft is being developed and supported by [David S. Fulford](#), [Dan Foreman-Mackey](#) and [David W. Hogg](#).

For the hackers in the house, development happens on [Github](#) and we welcome pull requests. In particular, we'd love to see examples of how you're using Daft in your work.

CHAPTER 5

License

Copyright 2012-2019 Daft Developers.

Daft is free software made available under the *MIT License*. For details see [the LICENSE file](#).

If you use Daft in academic projects, acknowledgements are greatly appreciated.

6.1 API

6.1.1 The PGM Object

All daft scripts will start with the creation of a *PGM* object. This object contains a list of *Node* objects and *Edge* objects connecting them. You can also specify rendering parameters and other default parameters when you initialize your *PGM*.

```
class daft.PGM(shape=None, origin=None, grid_unit=2.0, node_unit=1.0, observed_style='shaded',  
               alternate_style='inner', line_width=1.0, node_ec='k', directed=True, aspect=1.0, label_params={}, dpi=None)
```

The base object for building a graphical model representation.

Parameters

- **shape** – (optional) The number of rows and columns in the grid. Will automatically determine if not provided.
- **origin** – (optional) The coordinates of the bottom left corner of the plot. Will automatically determine if not provided.
- **grid_unit** – (optional) The size of the grid spacing measured in centimeters.
- **node_unit** – (optional) The base unit for the node size. This is a number in centimeters that sets the default diameter of the nodes.
- **observed_style** – (optional) How should the “observed” nodes be indicated? This must be one of: "shaded", "inner" or "outer" where *inner* and *outer* nodes are shown as double circles with the second circle plotted inside or outside of the standard one, respectively.
- **alternate_style** – (optional) How should the “alternate” nodes be indicated? This must be one of: "shaded", "inner" or "outer" where *inner* and *outer* nodes are shown as double circles with the second circle plotted inside or outside of the standard one, respectively.

- **node_ec** – (optional) The default edge color for the nodes.
- **directed** – (optional) Should the edges be directed by default?
- **aspect** – (optional) The default aspect ratio for the nodes.
- **label_params** – (optional) Default node label parameters. See `PGM.Node` for details.
- **dpi** – (optional) Set DPI for display and saving files.

add_edge (*name1*, *name2*, *directed=None*, *xoffset=0.0*, *yoffset=0.1*, *label=None*, *plot_params={}*, *label_params={}*, ***kwargs*)

Construct an [Edge](#) between two named [Node](#) objects.

Parameters

- **name1** – The name identifying the first node.
- **name2** – The name identifying the second node. If the edge is directed, the arrow will point to this node.
- **directed** – (optional) Should the edge be directed from `node1` to `node2`? In other words: should it have an arrow?
- **label** – (optional) A string to annotate the edge.
- **xoffset** – (optional) The x-offset from the middle of the arrow to plot the label. Only takes effect if *label* is defined in *plot_params*.
- **yoffset** – (optional) The y-offset from the middle of the arrow to plot the label. Only takes effect if *label* is defined in *plot_params*.
- **plot_params** – (optional) A dictionary of parameters to pass to the `matplotlib.patches.FancyArrow` constructor.
- **label_params** – (optional) A dictionary of parameters to pass to the `matplotlib.axes.Axes.annotate` constructor.

add_node (*node*, *content=""*, *x=0*, *y=0*, *scale=1.0*, *aspect=None*, *observed=False*, *fixed=False*, *alternate=False*, *offset=[0.0, 0.0]*, *fontsize=None*, *plot_params={}*, *label_params=None*, *shape='ellipse'*)

Add a [Node](#) to the model.

Parameters

- **node** – The plain-text identifier for the `nodeself`. Can also be the [Node](#) to retain backward compatibility.
- **content** – The display form of the variable.
- **x** – The x-coordinate of the node in *model units*.
- **y** – The y-coordinate of the node.
- **scale** – (optional) The diameter (or height) of the node measured in multiples of `node_unit` as defined by the [PGM](#) object.
- **aspect** – (optional) The aspect ratio width/height for elliptical nodes; default 1.
- **observed** – (optional) Should this be a conditioned variable?
- **fixed** – (optional) Should this be a fixed (not permitted to vary) variable? If *True*, modifies or over-rides `diameter`, `offset`, `facecolor`, and a few other `plot_params` settings. This setting conflicts with `observed`.
- **alternate** – (optional) Should this use the alternate style?

- **offset** – (optional) The (dx, dy) offset of the label (in points) from the default centered position.
- **fontsize** – (optional) The fontsize to use.
- **plot_params** – (optional) A dictionary of parameters to pass to the `matplotlib.patches.Ellipse` constructor.
- **label_params** – (optional) A dictionary of parameters to pass to the `matplotlib.text.Annotation` constructor. Any kwargs not used by Annotation get passed to `matplotlib.text.Text`.
- **shape** – (optional) String in {ellipse (default), rectangle} If rectangle, aspect and scale holds for rectangle

add_plate (*plate*, *label=None*, *label_offset=[5, 5]*, *shift=0*, *position='bottom left'*, *fontsize=None*, *rect_params=None*, *bbox=None*)
Add a *Plate* object to the model.

Parameters

- **plate** – The rectangle describing the plate bounds in model coordinates. Can also be the *Plate* to retain backward compatibility.
- **label** – (optional) A string to annotate the plate.
- **label_offset** – (optional) The x and y offsets of the label text measured in points.
- **shift** – (optional) The vertical “shift” of the plate measured in model units. This will move the bottom of the panel by *shift* units.
- **position** – (optional) One of "{vertical} {horizontal}" where vertical is "bottom" or "middle" or "top" and horizontal is "left" or "center" or "right".
- **fontsize** – (optional) The fontsize to use.
- **rect_params** – (optional) A dictionary of parameters to pass to the `matplotlib.patches.Rectangle` constructor.

add_text (*x*, *y*, *label*, *fontsize=None*)

A subclass of plate to writing text using grid coordinates. Any **kwargs** are passed through to `PGM.Plate`.

Parameters

- **x** – The x-coordinate of the text in *model units*.
- **y** – The y-coordinate of the text.
- **label** – A string to write.
- **fontsize** – (optional) The fontsize to use.

render (*dpi=None*)

Render the *Plate*, *Edge* and *Node* objects in the model. This will create a new figure with the correct dimensions and plot the model in this area.

Parameters **dpi** – (optional) The DPI value to use for rendering.

savefig (*fname*, **args*, ***kwargs*)

Wrapper on `matplotlib.Figure.savefig()` that sets default image padding using `bbox_inches = tight.*args` and **kwargs** are passed to `matplotlib.Figure.savefig()`.

Parameters

- **fname** – The filename to save as.
- **dpi** – (optional) The DPI value to use for saving.

show (*dpi=None, *args, **kwargs*)

Wrapper on [PGM.render\(\)](#) that calls `matplotlib.show()` immediately after.

Parameters **dpi** – (optional) The DPI value to use for rendering.

6.1.2 Nodes

class `daft.Node` (*name, content, x, y, scale=1.0, aspect=None, observed=False, fixed=False, alternate=False, offset=[0.0, 0.0], fontsize=None, plot_params={}, label_params=None, shape='ellipse'*)

The representation of a random variable in a [PGM](#).

Parameters

- **name** – The plain-text identifier for the node.
- **content** – The display form of the variable.
- **x** – The x-coordinate of the node in *model units*.
- **y** – The y-coordinate of the node.
- **scale** – (optional) The diameter (or height) of the node measured in multiples of `node_unit` as defined by the [PGM](#) object.
- **aspect** – (optional) The aspect ratio width/height for elliptical nodes; default 1.
- **observed** – (optional) Should this be a conditioned variable?
- **fixed** – (optional) Should this be a fixed (not permitted to vary) variable? If *True*, modifies or over-rides `diameter`, `offset`, `facecolor`, and a few other `plot_params` settings. This setting conflicts with `observed`.
- **alternate** – (optional) Should this use the alternate style?
- **offset** – (optional) The (`dx`, `dy`) offset of the label (in points) from the default centered position.
- **fontsize** – (optional) The fontsize to use.
- **plot_params** – (optional) A dictionary of parameters to pass to the `matplotlib.patches.Ellipse` constructor.
- **label_params** – (optional) A dictionary of parameters to pass to the `matplotlib.text.Annotation` constructor. Any kwargs not used by `Annotation` get passed to `matplotlib.text.Text`.
- **shape** – (optional) String in {`ellipse` (default), `rectangle`} If `rectangle`, `aspect` and `scale` holds for `rectangle`.

get_frontier_coord (*target_xy, ctx, edge*)

Get the coordinates of the point of intersection between the shape of the node and a line starting from the center of the node to an arbitrary point. Will throw a `SameLocationError` if the nodes contain the same x and y coordinates. See the example of rectangle below:

```
| _____ |      ____--X (target_node)
|           |      ____--X-----
```

(continues on next page)

(continued from previous page)

	X--	(return coordinate of this point)

Target_xy (x float, y float) A tuple of coordinate of target node

render (*ctx*)

Render the node.

Parameters **ctx** – The `_rendering_context` object.

6.1.3 Edges

class daft.**Edge** (*node1*, *node2*, *directed=True*, *label=None*, *xoffset=0*, *yoffset=0.1*, *plot_params={}*, *label_params={}*)

An edge between two `Node` objects.

Parameters

- **node1** – The first `Node`.
- **node2** – The second `Node`. The arrow will point towards this node.
- **directed** – (optional) Should the edge be directed from `node1` to `node2`? In other words: should it have an arrow?
- **label** – (optional) A string to annotate the edge.
- **xoffset** – (optional) The x-offset from the middle of the arrow to plot the label. Only takes effect if `label` is defined in `plot_params`.
- **yoffset** – (optional) The y-offset from the middle of the arrow to plot the label. Only takes effect if `label` is defined in `plot_params`.
- **plot_params** – (optional) A dictionary of parameters to pass to the `matplotlib.patches.FancyArrow` constructor to adjust edge behavior.
- **label_params** – (optional) A dictionary of parameters to pass to the `matplotlib.axes.Axes.annotate` constructor to adjust label behavior.

render (*ctx*)

Render the edge in the given axes.

Parameters **ctx** – The `_rendering_context` object.

6.1.4 Plates

class daft.**Plate** (*rect*, *label=None*, *label_offset=[5, 5]*, *shift=0*, *position='bottom left'*, *fontsize=None*, *rect_params=None*, *bbox=None*)

A plate to encapsulate repeated independent processes in the model.

Parameters

- **rect** – The rectangle describing the plate bounds in model coordinates. This is [x-start, y-start, x-length, y-length].
- **label** – (optional) A string to annotate the plate.
- **label_offset** – (optional) The x- and y- offsets of the label text measured in points.

- **shift** – (optional) The vertical “shift” of the plate measured in model units. This will move the bottom of the panel by `shift` units.
- **position** – (optional) One of "{vertical} {horizontal}" where vertical is "bottom" or "middle" or "top" and horizontal is "left" or "center" or "right".
- **fontsize** – (optional) The fontsize to use.
- **rect_params** – (optional) A dictionary of parameters to pass to the `matplotlib.patches.Rectangle` constructor, which defines the properties of the plate.
- **bbox** – (optional) A dictionary of parameters to pass to the `matplotlib.axes.Axes.annotate` constructor, which defines the box drawn around the text.

render (*ctx*)

Render the plate in the given axes.

Parameters **ctx** – The `__rendering_context` object.

6.1.5 The Rendering Context

class `daft.__rendering_context` (**kwargs)

Parameters

- **shape** – The number of rows and columns in the grid.
- **origin** – The coordinates of the bottom left corner of the plot.
- **grid_unit** – The size of the grid spacing measured in centimeters.
- **node_unit** – The base unit for the node size. This is a number in centimeters that sets the default diameter of the nodes.
- **observed_style** – How should the “observed” nodes be indicated? This must be one of: "shaded", "inner" or "outer" where inner and outer nodes are shown as double circles with the second circle plotted inside or outside of the standard one, respectively.
- **alternate_style** – (optional) How should the “alternate” nodes be indicated? This must be one of: "shaded", "inner" or "outer" where inner and outer nodes are shown as double circles with the second circle plotted inside or outside of the standard one, respectively.
- **node_ec** – The default edge color for the nodes.
- **directed** – Should the edges be directed by default?
- **aspect** – The default aspect ratio for the nodes.
- **label_params** – Default node label parameters.
- **dpi** – (optional) The DPI value to use for rendering.

convert (*xy)

Convert from model coordinates to plot coordinates.

d

daft, [11](#)

Symbols

`_rendering_context` (*class in daft*), 16

A

`add_edge()` (*daft.PGM method*), 12

`add_node()` (*daft.PGM method*), 12

`add_plate()` (*daft.PGM method*), 13

`add_text()` (*daft.PGM method*), 13

C

`convert()` (*daft._rendering_context method*), 16

D

`daft` (*module*), 11

E

`Edge` (*class in daft*), 15

G

`get_frontier_coord()` (*daft.Node method*), 14

N

`Node` (*class in daft*), 14

P

`PGM` (*class in daft*), 11

`Plate` (*class in daft*), 15

R

`render()` (*daft.Edge method*), 15

`render()` (*daft.Node method*), 15

`render()` (*daft.PGM method*), 13

`render()` (*daft.Plate method*), 16

S

`savefig()` (*daft.PGM method*), 13

`show()` (*daft.PGM method*), 14