
nxv

Release 0.1.3

Timothy Shields

Sep 27, 2020

CONTENTS

1	Installation	3
1.1	Dependencies	3
2	Quickstart	5
2.1	Using nxv inside of Jupyter	5
2.2	Using nxv outside of Jupyter	6
3	Reference	7
3.1	Rendering	7
3.2	Styling	8
3.3	Utilities	10
3.4	Errors	11
4	License	13
	Python Module Index	17
	Index	19

nxv renders [NetworkX](#) graphs using GraphViz.

```
import networkx as nx
import nxv

graph = nx.Graph()
graph.add_edge("A", "B")
graph.add_edge("B", "C")
graph.add_edge("C", "D")
graph.add_edge("B", "E")

style = nxv.Style(
    graph={"rankdir": "LR"},
    node=lambda u, d: {"shape": "circle" if u in "AEIOU" else "square"},
    edge=lambda u, v, d: {"style": "dashed", "label": u + v},
)

nxv.render(graph, style)
```


INSTALLATION

The nxv package is available on [PyPI](#).

To install nxv with pip:

```
pip install nxv
```

1.1 Dependencies

nxv requires a GraphViz installation. Instructions for how to [download and install GraphViz](#) can be found on the official GraphViz site.

QUICKSTART

`nxv` renders [NetworkX](#) graphs using [GraphViz](#).

- *Using `nxv` inside of Jupyter* is the easiest and recommended way to get started.
- *Using `nxv` outside of Jupyter* describes how to use `nxv` in other settings.

2.1 Using `nxv` inside of Jupyter

Start by importing `networkx` and `nxv`.

```
import networkx as nx
import nxv
```

Define a simple [NetworkX](#) graph.

```
graph = nx.Graph()
graph.add_edge("A", "B")
graph.add_edge("B", "C")
graph.add_edge("C", "D")
graph.add_edge("B", "E")
```

Render the graph with [GraphViz](#) using the `render()` function.

```
nxv.render(graph)
```

Use a [Style](#) to specify how to style the graph using [GraphViz](#).

```
style = nxv.Style(
    graph={"rankdir": "LR"},
    node={"shape": "square"},
    edge={"style": "dashed"},
)
```

See the [GraphViz attributes](#) documentation for information on what attributes are available to use.

Render the graph with the [Style](#) by passing it to the `render()` function.

```
nxv.render(graph, style)
```

The [Style](#) parameters can be functions that map the parts of a graph to different styles.

```
style = nxv.Style(  
    graph={"rankdir": "LR"},  
    node=lambda u, d: {"shape": "circle" if u in "AEIOU" else "square"},  
    edge=lambda u, v, d: {"style": "dashed", "label": u + v},  
)
```

```
nxv.render(graph, style)
```

2.2 Using nxv outside of Jupyter

Outside of `Jupyter`, the `format` parameter of the `render()` function is required. When the `format` parameter is provided, the behavior of the `render()` function is to return the `bytes` of the result in the specified format.

```
data = nxv.render(graph, style, format="svg")  
with open("graph.svg", "wb") as f:  
    f.write(data)
```

REFERENCE

3.1 Rendering

`nxv.render(graph, style=None, *, algorithm=None, format=None, graphviz_bin=None, subgraph_func=None)`

Render a [NetworkX](#) graph using [GraphViz](#).

In a Jupyter notebook, this will automatically display as an SVG.

Parameters

- **graph** (`Union[Graph, DiGraph, MultiGraph, MultiDiGraph]`) – A [NetworkX](#) graph.
- **style** (`Optional[Style]`) – A style specifying how graph nodes and edges should map to [GraphViz](#) attributes.
- **subgraph_func** – An optional function $f(u, d)$ that returns a subgraph key, where u is a [NetworkX](#) node and d is its attribute dict. If it returns `None` the node is not in any subgraph.
- **algorithm** (`Optional[str]`) – The [GraphViz](#) layout algorithm. Valid options include "circo", "dot", "fdp", "neato", "osage", "sfdp", "twopi". Defaults to "dot".
- **format** (`Optional[str]`) – The [GraphViz](#) output format. Valid options include "svg" and "raw". In a Jupyter notebook, prefixing the format with "ipython/" will automatically display the rendered output. When running in an interactive setting like a Jupyter notebook, the default is "ipython/svg". Otherwise, this parameter is required.
- **graphviz_bin** (`Optional[str]`) – The bin directory of the [GraphViz](#) installation. Defaults to the `GRAPHVIZ_BIN` environment variable. If neither this parameter nor the `GRAPHVIZ_BIN` environment variable is set, then `nxv` will try to autodetect the bin directory of the [GraphViz](#) installation. This behavior is for convenience and should not be relied on in production settings.

Return type `Optional[bytes]`

Returns If `format` is not an "ipython/*" format, the render output; otherwise, `None`.

Raises

- [`GraphVizInstallationNotFoundError`](#) – If `nxv` cannot find a [GraphViz](#) installation.
- [`GraphVizAlgorithmNotFoundError`](#) – If `nxv` cannot find the specified algorithm in a [GraphViz](#) installation.

- **GraphVizError** – If **GraphViz** failed to run on the given inputs.

3.2 Styling

class `nxv.Style` (*, *graph=None*, *node=None*, *edge=None*, *subgraph=None*)

A specification for how to style a **NetworkX** graph using **GraphViz**.

See the **GraphViz attributes** documentation for information on what attributes are available to use with the *graph*, *node*, *edge*, and *subgraph* parameters.

Parameters

- **graph** – An optional dict of **GraphViz graph attributes**, or a function $f(g, d)$ that returns it, in which g is the **NetworkX** graph and d is its attribute dict.
- **node** – An optional dict of **GraphViz node attributes**, or a function $f(u, d)$ that returns it, in which u is a **NetworkX** node and d is its attribute dict.
- **edge** – An optional dict of **GraphViz edge attributes**, or a function $f(u, v, d)$ that returns it, in which (u, v) is a **NetworkX** edge and d is its attribute dict. If styling a graph with multi-edges, the signature should be $f(u, v, k, d)$ instead, where k is the edge key.
- **subgraph** – An optional dict of **GraphViz subgraph attributes**, or a function $f(s)$ that returns it, in which s is a subgraph key. This only applies when calling `nxv.render` with a *subgraph_func*.

`nxv.compose` (*styles*)

Compose a sequence of *Style* objects as a single *Style*.

Parameters *styles* (`Iterable[Optional[Style]]`) – An iterable of *Style* objects.

Return type *Style*

Returns The composed *Style*.

`nxv.chain` (*funcs*)

Chain a sequence of dict-returning functions together to form a new dict-returning function.

The result is a function $f(*args, **kwargs)$ that returns `{**apply(funcs[0], *args, **kwargs), **apply(funcs[1], *args, **kwargs), ...}`.

Parameters *funcs* – An iterable of functions that return dicts.

Returns A function $f(*args, **kwargs)$ that returns `{**apply(funcs[0], *args, **kwargs), **apply(funcs[1], *args, **kwargs), ...}`.

`nxv.switch` (*key*, *funcs*, *, *default=None*)

Combine a dict of keyed functions to form a new function.

The result is a function $f(*args, **kwargs)$ that returns `apply(funcs[key(*args, **kwargs)], *args, **kwargs)`.

If $key(*args, **kwargs)$ is not in *funcs* but *default* is present, `apply(default, *args, **kwargs)` will be returned instead.

Parameters

- **key** – The key selector function.
- **funcs** – The mapping from keys to functions.
- **default** – An optional default function for keys that do not appear in *funcs*.

Returns The function `f(*args, **kwargs)` that returns `apply(funcs[key(*args, **kwargs)], *args, **kwargs)`.

`nxv.styles.verbose()`

Get a verbose *Style* that shows all of the data in a graph.

Return type *Style*

Returns A verbose *Style*.

`nxv.styles.font(fontname=None, fontsize=None)`

Styles text in a graph using the given font.

Parameters

- **fontname** (*Optional[str]*) – Optional font name.
- **fontsize** (*Union[int, float, None]*) – Optional font size.

Return type *Style*

Returns A *Style* object that applies this font.

3.2.1 HTML-Like Labels

The `nxv.html_like` subpackage provides functions for building GraphViz HTML-like labels.

The idiomatic import for this subpackage is:

```
import nxv.html_like as H
```

`nxv.html_like.join(children)`

`nxv.html_like.line_break(attributes=None)`

`nxv.html_like.font(content, attributes=None)`

`nxv.html_like.italic(content)`

`nxv.html_like.bold(content)`

`nxv.html_like.underline(content)`

`nxv.html_like.overline(content)`

`nxv.html_like.subscript(content)`

`nxv.html_like.superscript(content)`

`nxv.html_like.strikethrough(content)`

`nxv.html_like.table(rows, attributes=None)`

`nxv.html_like.table_row(cells)`

`nxv.html_like.horizontal_rule()`

`nxv.html_like.table_cell(content, attributes=None)`

`nxv.html_like.vertical_rule()`

`nxv.html_like.image(attributes=None)`

3.3 Utilities

`nxv.neighborhood` (*graph*, *nodes*, *, *radius=None*, *cost=None*)

Get the subgraph in the neighborhood of the specified nodes.

This is useful for viewing a small portion of a large graph.

Parameters

- **graph** – A graph.
- **nodes** – An iterable of nodes.
- **radius** – The size of the neighborhood.
- **cost** – A function $f(u, v)$ specifying the cost of traversing from u to v .

Returns The neighborhood subgraph.

`nxv.boundary` (*graph*, *subgraph*)

Get the nodes in the subgraph that have neighbors in the graph but not in the subgraph.

This is useful for conditionally styling nodes at the boundary of a subgraph. For example:

```
boundary = nxv.boundary(graph, subgraph)
style = nxv.Style(node=lambda u, d: {
    'style': 'dashed' if u in boundary else 'solid',
})
nxv.render(subgraph, style)
```

Parameters

- **graph** – A graph.
- **subgraph** – A subgraph of the graph.

Returns The nodes in the subgraph that have neighbors in the graph but not in the subgraph.

`nxv.to_ordered_graph` (*graph*, *node_key=None*, *edge_key=None*, *attr_key=None*)

Create an ordered copy of the specified graph, with nodes and edges ordered by the specified key functions.

Parameters

- **graph** – The graph to order.
- **node_key** – The node key function, `node_key(u, d)`. Defaults to the identity function.
- **edge_key** – The edge key function, `edge_key(u, v, d)`. If the graph has multi-edges, the signature should be `edge_key(u, v, k, d)` instead, where k is the edge key. Defaults to the identity function.
- **attr_key** – The attribute key function, `attr_key(k, v)`. Defaults to the identity function.

Returns A copy of the graph with the nodes and edges ordered by the specified key functions.

`nxv.contrasting_color` (*channels*, *, *options=None*)

Get a color that most contrasts with a specified color.

Parameters

- **channels** – The RGB or RGBA color channels. Values should be in the range $[0, 1]$.

- **options** – The possible contrasting colors. Defaults to black and white.

Returns The color option that most contrasts the input color.

3.4 Errors

class `nxv.GraphVizInstallationNotFoundError`

Raised when a `GraphViz` installation is not found.

class `nxv.GraphVizAlgorithmNotFoundError`

Raised when a `GraphViz` algorithm is not found.

class `nxv.GraphVizError`

Raised when a `GraphViz` run fails.

LICENSE

Apache License
Version 2.0, January 2004
<http://www.apache.org/licenses/>

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION**1. Definitions.**

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain

(continues on next page)

(continued from previous page)

separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.
4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:
 - (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and
 - (b) You must cause any modified files to carry prominent notices stating that You changed the files; and
 - (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and

(continues on next page)

(continued from previous page)

- attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
- (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.
6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.
8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a

(continues on next page)

(continued from previous page)

result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "{}" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright {yyyy} {name of copyright owner}

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

PYTHON MODULE INDEX

n

`nxv.html_like`, 9

B

`bold()` (in module `nxv.html_like`), 9
`boundary()` (in module `nxv`), 10

C

`chain()` (in module `nxv`), 8
`compose()` (in module `nxv`), 8
`contrasting_color()` (in module `nxv`), 10

F

`font()` (in module `nxv.html_like`), 9
`font()` (in module `nxv.styles`), 9

G

`GraphVizAlgorithmNotFoundError` (class in `nxv`), 11
`GraphVizError` (class in `nxv`), 11
`GraphVizInstallationNotFoundError` (class in `nxv`), 11

H

`horizontal_rule()` (in module `nxv.html_like`), 9

I

`image()` (in module `nxv.html_like`), 9
`italic()` (in module `nxv.html_like`), 9

J

`join()` (in module `nxv.html_like`), 9

L

`line_break()` (in module `nxv.html_like`), 9

M

module
`nxv.html_like`, 9

N

`neighborhood()` (in module `nxv`), 10
`nxv.html_like`
module, 9

O

`overline()` (in module `nxv.html_like`), 9

R

`render()` (in module `nxv`), 7

S

`strikethrough()` (in module `nxv.html_like`), 9
`Style` (class in `nxv`), 8
`subscript()` (in module `nxv.html_like`), 9
`superscript()` (in module `nxv.html_like`), 9
`switch()` (in module `nxv`), 8

T

`table()` (in module `nxv.html_like`), 9
`table_cell()` (in module `nxv.html_like`), 9
`table_row()` (in module `nxv.html_like`), 9
`to_ordered_graph()` (in module `nxv`), 10

U

`underline()` (in module `nxv.html_like`), 9

V

`verbose()` (in module `nxv.styles`), 9
`vertical_rule()` (in module `nxv.html_like`), 9