pheatmap

Release 0.1.0

Zongliang Hou

Jan 08, 2023

CONTENTS:

pheatmap is a python implementation for pheatmap of R!

You can use pheatmap with pure python and don't have to write data to file, then plot heatmap by R.

CHAPTER

ONE

BASIC USAGES

1.1 Introduction

`pheatmap` is a Python package for ploting heatmap with annotation bars. It just like `pheatmap` package of R.

1.2 Installation

You can install `pheatmap` by `pip`:

```
pip install pheatmap
```

Run the above command, `pip` will intall `pheatmap` and its dependencies, `numpy`, `pandas` and `matplotlib`.

1.3 Quick Start

Before create heatmap, we need to import some essential packages.

```
import pandas as pd
import numpy as np
from pheatmap import pheatmap
```

1.3.1 Simple Heatmap

First, we can create a DataFrame, each row means a gene, and columns are samples.

```
mat = pd.DataFrame(
    np.random.randint(0, 10, 600).reshape(30, 20),
    index=[f"G{i}" for i in np.arange(30)],
    columns=[f"sample{i}" for i in np.arange(20)]
)
# Show some expression values
mat.iloc[:, :5].head()
```

	sample0	sample1	sample2	sample3	sample4	sample5
G0	4	9	4	6	6	1
G1	1	4	6	3	2	1
G2	8	8	9	1	0	8
G3	2	5	9	3	2	3
G4	4	6	8	3	2	1

Then, we can plot a simple heatmap for these genes' expression on samples:

```
fig = pheatmap(mat, colnames_style={"rotation": 90})
```

1.3.2 Annotation Heatmap

Annotation columns

If we have some information for samples, such as experiment v.s. control or sample source, then we can and these information to the plot by `annotation_col` parameter.

```
# Create DataFrame, store samples' information
anno_col = pd.DataFrame({
    "source": np.repeat(["source1", "source2", "source3", "source4"], 5),
    "group": np.repeat(["experiment", "control"], 10)
})
fig = pheatmap(mat, annotation_col=anno_col, colnames_style={"rotation": 90}, legend_bar_
↪space=1.5)
```

We can point colors for every annotation bar by `annotation_col_cmaps` parameter. `annotation_col_cmaps` need a `dict`, the keys of `dict` are the anno_col's columns, and the values of dict are the colors. If the number of colors are not enough for the categories', `pheatmap` will cycle the colors to fullfill the number of categories.

```
fig = pheatmap(
    mat, annotation_col=anno_col,
    annotation_col_cmaps={"group": ["#7FC97F", "#FB9A99"]},
    colnames_style={"rotation": 90}, legend_bar_space=1.5
)
```

Annotation Rows

We can also add annotations for genes. For example, we want to check the degree of DNA methylation for genes. We can put the degree of DNA methylation on the left of heatmap. By the way, we have the information of clusters for genes and want to see it at the same time.

```
anno_row = pd.DataFrame({
    "DNA-methyl": np.linspace(0, 1, 30),
    "gene_clusters": ["CNS"[i % 3] for i in np.random.randint(0, 10, 30)]
})
fig = pheatmap(
    mat,
    annotation_col=anno_col,
    annotation_col_cmaps={"group": ["#7FC97F", "#FB9A99"]},
    annotation_row=anno_row,
    annotation_row_cmaps={"DNA-methyl": "Purples", "gene_clusters": "Set2"},
    colnames_style={"rotation": 90},
    legend_bar_space=1.5,
    wspace=0.15
)
fig.savefig("row_col_anno_heatmap.svg")
```

CHAPTER

TWO

ADVANCE USAGE

2.1 some_style

For the arguments ended with 'style', you can give a `dict`, which can be used to control text's style. For example, `rownames_style` can change the style of row's names.

First, we create a DataFrame represent a gene expresion matrix.

```
mat = pd.DataFrame(
    np.random.randint(0, 30, 20).reshape(5, 4),
    index=["ERBB2", "ERBB4", "EZH2", "FGFR2", "FGFR3"]
)
```

The default heatmap of the above DataFrame is as follow:

```
fig = pheatmap(mat=mat, show_colnames=False)
```

Now, we can change fontsize to 10pt(default: 6pt), fontstyle to "italic"(default: normal) and left align.

> Note: left align, the rownames will be overlap with heatmap, therefore, need to adjust rownames to left by `position`

```
fig = pheatmap(
    mat=mat, show_colnames=False,
    rownames_style={"fontsize": 10, "fontstyle": "italic", "ha": "left", "position": (-0.
↪1, 0)}
)
```

2.2 Legend

For legend, you can adjust every legend by point the legend's name as keys to `legend_*`. For example, we can normalize the genes' expression to [0, 1] by max-min normalize.

```
mat = (mat - mat.min()) / mat.max()
fig = pheatmap(
    mat=mat, show_colnames=False,
    rownames_style={"fontsize": 10, "fontstyle": "italic", "ha": "left", "position": (-0.
```

(continues on next page)

(continued from previous page)

```
↪1, 0)},
)
```

But the legend's labels are not beautiful, you can modify it by youself.

```
fig = pheatmap(
    mat=mat, show_colnames=False,
    rownames_style={"fontsize": 10, "fontstyle": "italic", "ha": "left", "position": (-0.
↪1, 0)},
    legend_tick_locs={"heatmap": np.arange(0, 1, 0.2)},
    legend_tick_labels={"heatmap": [f"{i:.3f}" for i in np.arange(0, 1, 0.2)]}
)
```

You can also change label style for legend tick labels, such as: fontsize, rotation, fontstyle, etc.

```
fig = pheatmap(
    mat=mat, show_colnames=False,
    rownames_style={"fontsize": 10, "fontstyle": "italic", "ha": "left", "position": (-0.
↪1, 0)},
    legend_tick_locs={"heatmap": np.arange(0, 1.2, 0.2)},
    legend_tick_labels={"heatmap": [f"{i:.3f}" for i in np.arange(0, 1.2, 0.2)]},
    legend_tick_labels_styles={"fontsize": 8, "color": "gray"}
)
```

2.3 Annotation Advance

Annotations can be continuous or discrete for rows or columns. If the annotation is discrete, you can specify the order of the categories. For example, we want to visualize some gene expression dynamic along with the embryo development.

```
mat = pd.DataFrame(
    np.hstack([np.random.randint(0, 15, 20 * 10).reshape(20, 10),
               np.random.randint(5, 20, 20 * 20).reshape(20, 20),
               np.random.randint(10, 25, 20 * 30).reshape(20, 30)]),
    index=[f"G{i}" for i in np.arange(20)]
)
anno_col = pd.DataFrame({
    "embryo-developing": np.hstack([np.repeat(["Morula"], 10), np.repeat(["Blastula"],␣
↪20), np.repeat(["ESC"], 30)])
})
fig = pheatmap(
    mat, cmap="Reds",
    annotation_col=anno_col,
    annotation_col_cmaps={"embryo-developing": "Set1"},
    show_colnames=False
)
```

The legend's of embryo-developing annotation is not right order of embryo development. You can specify the order by the follow method:

```
anno_col["embryo-developing_right_order"] = anno_col["embryo-developing"].astype(
    pd.CategoricalDtype(["Morula", "Blastula", "ESC"]))
fig = pheatmap(
    mat, cmap="Reds", annotation_col=anno_col,
    annotation_col_cmaps={"embryo-developing_right_order": "Set2", "embryo-developing":
↪"Set1"},
    show_colnames=False, legend_bar_space=2.5
)
```

More information to see *pheatmap API*.

CHAPTER

THREE

PHEATMAP API

pheatmap.**pheatmap**(*mat: DataFrame*, *cmap: Union[str, Colormap, list] = 'bwr'*, *vmin: Optional[float] = None*, *vmax: Optional[float] = None*, *name: Optional[str] = None*, *rownames: Optional[ndarray] = None*, *colnames: Optional[ndarray] = None*, *rownames_side: str = 'left'*, *colnames_side: str = 'bottom'*, *show_rownames: bool = True*, *show_colnames: bool = True*, *rownames_style: dict = {'rotation': 0, 'size': 6}*, *colnames_style: dict = {'rotation': 0, 'size': 6}*, *edgecolor: str = 'none'*, *edgewidth: float = 1*, *annotation_row: Optional[DataFrame] = None*, *annotation_col: Optional[DataFrame] = None*, *annotation_row_cmaps: Optional[Dict[str, Union[str, Colormap, list]]] = None*, *annotation_col_cmaps: Optional[Dict[str, Union[str, Colormap, list]]] = None*, *annotation_row_names_style: Dict = {'size': 6}*, *annotation_col_names_style: Dict = {'size': 6}*, *show_annotation_row_names: bool = True*, *show_annotation_col_names: bool = True*, *legend_tick_locs: Optional[Dict[str, Sequence]] = None*, *legend_tick_labels: Optional[Dict[str, Sequence]] = None*, *legend_tick_labels_styles: Dict = {'size': 6}*, *legend_titles: Optional[Dict[str, bool]] = None*, *legend_title_styles: Dict = {'size': 6}*, *width: float = 8*, *height: float = 6*, *wspace: float = 0.1*, *hspace: float = 0.1*, *annotation_bar_width: float = 0.03*, *legend_bar_width: float = 0.045*, *annotation_bar_space: float = 0.2*, *legend_bar_space: float = 1*) → Figure

Plot heatmap with annotation bars

mat
: [DataFrame] the main heatmap DataFrame

cmap
: [Union[str, Colormap, list], optional] the colormap of heatmap, by default “bwr”

vmin
: [float, optional] the minemum value scaled. by default None, use the minemum value of *mat*

vmax
: [float, optional] the maximum value scaled. by default None, use the maximum value of *mat*

name
: [str, optional] the name of heatmap, by default None, use “heatmap” as the name of heatmap

rownames
: [ndarray, optional] the row names provided, by default None, use the row names of main heatmap DataFrame

colnames
: [ndarray, optional] the column names provided, by default None, use the column names of main heatmap DataFrame

rownames_side
: [str, optional] show row names on which side(“left” or “right”)? by default “left”

colnames_side
[str, optional] show column names on which side("top" or "bottom")? by default "bottom"

show_rownames
[bool, optional] whether show row names? by default True

show_colnames
[bool, optional] whether show column names? by default True

rownames_style
[dict, optional] modify row names' style, such as, fontsize, fontstyle, etc. See more information on [*matplotlib.text.Text*](https://matplotlib.org/stable/api/text_api.html#matplotlib.text.Text). by default dict(rotation=0, size=6)

colnames_style
[dict, optional] see *rownames_style*, by default dict(rotation=0, size=6)

edgecolor
[str, optional] the color of heatmap's cell edge, by default "none", no edge. !Note: If provide *None*, will use the *rcParams["patch.edgecolor"]*, it default as "black".

edgewidth
[float, optional] the width of heatmap's cell edge, by default 1

annotation_row
[DataFrame, optional] DataFrame used to create row Annotationbar, by default None

annotation_col
[DataFrame, optional] DataFrame used to create column Annotationbar, by default None

annotation_row_cmaps
[Dict[str, Union[str, Colormap, list]], optional] Colormaps for each Annotationbar, keys are the DataFrame's columns, by default None, use "viridis" for continuous and "tab20" for discrete

annotation_col_cmaps
[Dict[str, Union[str, Colormap, list]], optional] see *annotation_row_cmaps*, by default None

annotation_row_names_style: Dict[str, Dict], optional
modify the style of row AnnotationBar's name. Keys are the DataFrame's columns, by default None

annotation_col_names_style: Dict[str, Dict], optional
modify the style of column AnnotationBar's name. Keys are the DataFrame's columns, by default None

show_annotation_row_names
[bool, optional] whether show row Annotationbar's name, by default True

show_annotation_col_names
[bool, optional] whether show column Annotationbar's name, by default True

legend_tick_locs
[Dict[str, Sequence], optional] modify the tick locations of legend, keys are the name of heatmap or the column names of annotation DataFrame, by default None

legend_tick_labels
[Dict[str, Sequence], optional] modify the tick labels of legend, keys are the name of heatmap or the column names of annotation DataFrame. The length of each legend tick labels must be matched with its tick locations. by default None

legend_tick_labels_styles
[Dict, optional] modify the each legend tick labels' style. Keys are the name of heatmap or the column names of annotation DataFrame. More informations see *rownames_style*. by default dict(size=6)

legend_titles
[Dict[str, bool], optional] modify the each legend title. Others are the same as *legend_tick_labels*. by default None

legend_title_styles
[Dict, optional] modify the each legend title's style. Others are the same as *legend_tick_labels_styles*. by default dict(size=6)

width
[float, optional] the whole figure width, by default 8

height
[float, optional] the whole figure height, by default 6

wspace
[float, optional] the space of whole figure at the width direction, by default 0.1. It's the fraction of the average length of width's axises

hspace
[float, optional] see wspace, by default 0.1

annotation_bar_width
[float, optional] the width of Annotationbar, by default 0.03. It's the fraction of the whole figure width

legend_bar_width
[float, optional] the width of legend bar, by default 1.5*0.03. It's the fraction of the whole figure width. And it's better, set it as 1.5 * annotation_bar_width.

annotation_bar_space
[float, optional] the space between Annotationbars, by default 0.2. It's the fraction of the real Annotationbar width.

legend_bar_space
[float, optional] the space between legend bars, by default 1. It's the fraction of the real legend bar width

Figure

INDEX

P