

---

# **dppd**<sub>*p*</sub>*lotnine*Documentation

**Florian Finkernagel**

**Aug 17, 2023**



---

## Contents

---

|          |                                 |          |
|----------|---------------------------------|----------|
| <b>1</b> | <b>Quickstart</b>               | <b>3</b> |
| <b>2</b> | <b>Contents</b>                 | <b>5</b> |
| 2.1      | Changes from plotnine . . . . . | 5        |
| 2.2      | License . . . . .               | 6        |
| 2.3      | Contributors . . . . .          | 6        |
| 2.4      | Changelog . . . . .             | 7        |
| 2.5      | dppd_plotnine . . . . .         | 7        |
| <b>3</b> | <b>Indices and tables</b>       | <b>9</b> |



Welcome to **dppd\_plotnine**, which converts [plotnine](#) to work with [dppd](#) and pythonifies it's api.  
It's source lives at [github](#).



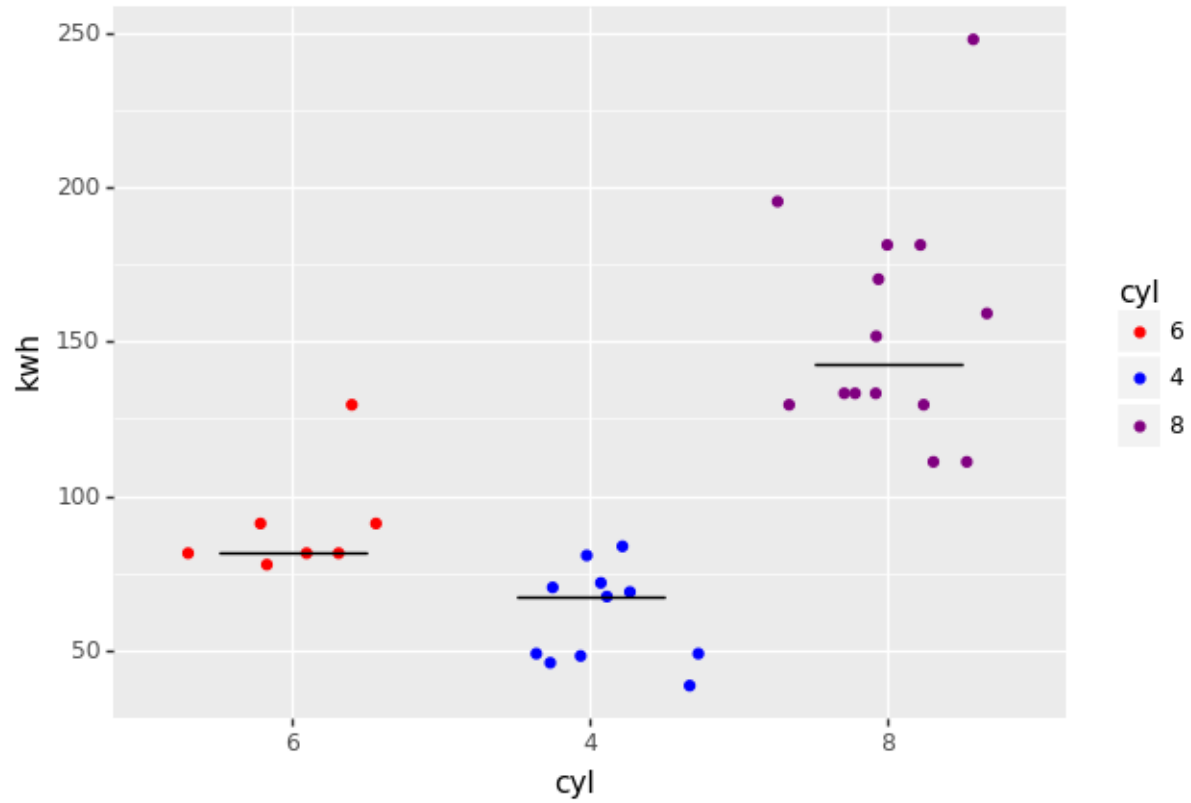
# CHAPTER 1

## Quickstart

```
import numpy as np
from dppd import dppd
import dppd_plotnine
from plotnine.data import mtcars
import plotnine as p9

dp, X = dppd()

plot = (
    dp(mtcars)
    .assign(kwh=X.hp * 0.74)
    .categorize("cyl")
    .p9()
    .add_point(
        "cyl",
        "kwh",
        color="cyl",
        position=p9.position_jitter(height=0, random_state=500),
    )
    .add_errorbar(
        x="cyl",
        y="kwh_median",
        ymin="kwh_median",
        ymax="kwh_median",
        data=dp(X.data)
        .groupby("cyl")
        .summarize(("kwh", np.median, "kwh_median"))
        .pd,
    )
    .scale_color_manual(
        ["red", "blue", "purple"]
    ) # after pd, X is what it was before
    .pd
)
plot.save("test.png")
```



`dppd_plotnine` supports two different call conventions, one matching `plotnine` (and `ggplot`) and another, perhaps more convinient one, see [call convention](#).



## 2.1 Changes from plotnine

### 2.1.1 Call convention

The `dppd_plotnine` interface differs in two aspects from plotnine (and R's `ggplot2`) API:

Method-chaining replaces addition

```
#plotnine
p = p9.ggplot(df)
p += p9.geom_point(p9.aes('x', 'y'), color='red')

#dppd_plotnine
dp(df).p9.geom_point(p9.aes('x', 'y'), color='red').pd
```

And optionally, aes mapping and kwargs are replaced by kwargs starting with and without underscore:

```
dp(df).p9.add_point('x', y='y', _color='red').pd
```

All `geom_*` functions can be called as `add_*` with the following changes to the calling api:

- all `REQUIRED_AES` can be passed by position. The order is x,y, then alphabetically
- kwargs that name an aes are mapped (ie. as if they were passed to `mapping=p9.aes(...)`).
- kwargs starting with a `'_'` are treated as unmapped (ie. `_color=` get's passed as `color=` kwarg to the geom)
- data, position and stat are left alone.
- `data=None` turns the geom into an annotation.

This pythonifies the interface a bit and get's rid of the `p9.aes` boilerplate, while maintaining full expressiveness.

Note that this also allows you to use `geom_hline(300)` without having to type `y_intercept=...`

### 2.1.2 Ways to pass in data

1. As an expression evaluated in the context of the DataFrame, using patsy, mapped: `add_point(x='df_column * 5', ...)`. Note that column named 'x \* 5' takes precedence over a column 'x' multiplied by 5.
2. As a mapped list, the size of the DataFrame: `add_point(x=['a', 'b', 'c'], ...)`
3. As an unmapped list, the size of the DataFrame: `add_point(_color=['red', 'blue', 'red', 'blue'])`
4. As a mapped scalar: `add_point(x='"a"' ...)`, note the inner quoting!
5. As a unmapped scalar: `add_point(..., _color='red')`
6. A mapping may refer to stat derived variables, `add_bar(x='x', y='stat(count)', stat=p9.stat_count)`
7. If `data=None` is passed, the geom is treated as an annotation and a DataFrame is constructed from the values of the mappend and unmapped arguments: `add_point(data=None, x=5, y=10, _color='red')`

### 2.1.3 Other changes from plotnine

- the default stat on `geom_bar` is `stat_identity`.
- `save` is `verbose=False` by default and returns the plot object
- `save` is aliased to `render`
- `add_scatter` is an alias for `add_point`
- there is a small set of convenience wrappers - see `dppd_plotnine.plotnine_extensions`

## 2.2 License

The MIT License (MIT)

Copyright (c) 2018 Florian Finkernagel

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## 2.3 Contributors

- Florian Finkernagel <[finkernagel@imt.uni-marburg.de](mailto:finkernagel@imt.uni-marburg.de)>

## 2.4 Changelog

### 2.4.1 Version 0.2.4

- `annotation_strips` -> `annotation_strips_dppd` (we still have some improvements from the plotnine default)
- experimental `.cyberpunk()` aesthetics
- `scale_x/y_continuous` aliases (`sxc`, `sxc10` for `log10`, `sxc2` for `log()`)
- `.hide_legend`
- `.figure_size/fig_size/size` for adjusting size in jupyter notebooks
- `.add_cumulative`

### 2.4.2 Version 0.2.1

- version no longer managed by git tags
- tests no longer need 'mbf\_qualitycontrol'

### 2.4.3 Version 0.2

- fixed package name
- plots now understand 'render\_args' which are passed to save later on.
- turn axis labels
- improved tests

### 2.4.4 Version 0.1

- Feature A added
- FIX: nasty bug #1729 fixed
- add your changes here!

## 2.5 dppd\_plotnine

### 2.5.1 dppd\_plotnine package

#### Subpackages

`dppd_plotnine.geoms` package

#### Submodules

`dppd_plotnine.geoms.annotation_strips_dppd` module

## Module contents

### Submodules

**dppd\_plotnine.dppd\_plotnine** module

**dppd\_plotnine.plotnine\_extensions** module

### Module contents

## CHAPTER 3

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`