Data Visualization with ggplot2 vs Matplotlib cheatsheet

Basics

ggplot2 is a system for declaratively creating graphics in R, based on The Grammar of Graphics.

Install:

```
install.packages("ggplot2")
library(ggplot2)
```

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

Install:

```
python -m pip install -U matplotlib
import matplotlib.pyplot as plt
```

Overview

ggplot2:

template:

```
ggplot (data =  ) +
        (aes(... ),
        stat =  ,
        position =  ) +
        <COORDINATE_FUNCTION> +
        <FACET_FUNCTION> +
        <SCALE_FUNCTION> +
        <THEME_FUNCTION>
```

save image:

```
ggsave("plot.png", width = 5, height = 5)
```

Matplotlib:

template:

```
initliaze data
fig, ax = plt.subplot()
ax.plot = (x, y, color = )
fig.show()
```

save image:

```
fig.savefig()
```

Basic plots

Barplot:

- **ggplot:** ggplot(data....) + geom_bar()
- **matplotlib:** plt.bar(names, values)

Histogram

- **ggplot**: ggplot(data...) + n geom_histogram()
- **matplotlib:** hist(x,bins...)

Scatter plot

- **ggplot**: ggplot (data...) + geom_point()
- **matplotlib:** scatter(x,y)

Hex plot

- **ggplot**: geom_hex()+ geom_line()
- **matplotlib:** hexbin(x,y,c)

Box plot

- **ggplot:** ggplot (data..) + geom_boxplot()
- **matplotlib:** boxplot(x)

Violin plot

- **ggplot:** ggplot (data..) + geom_violin()
- **matplotlib:** violinplot()

Contour plot

- **ggplot:** ggplot (data..) + geom_contour()
- **ggplot:** geom_contour_filled(aes(fill = z))
- **matplotlib:** contour[f]([x],[y],[z]

Scale, Labs and Legend

ggplot: change scale to log, square root, etc

```
ggplot(data....) +
        scale_x_log10()
        scale_x_sqrt()
```

matplotlib: change scale to linear, log, and other forms

```
ax.set_xscale(log)
ax.set_yscale('linear')
ax.set_yscale('symlog')
ax.set_yscale('logit')
```

ggplot: add title for x, y, caption, subtitle

```
ggplot(data....)  +
        labs(x = "...",
        y = "...",
        title ="...",
        subtitle = "..."
        caption = "...",
        alt = "...")
```

matplotlib: set title and x.y labels:

```
• plt.title()
• ax.set(title='...',
        ylabel='...',
        xlabel='...')
```

ggplot: change scale to continuous./discrete/indentity scale, also could add limit, breaks, etc

```
ggplot(data....)  +
        scale_x/y_continuous()
        scale_x/y_discrete()
        scale_x/y_indentity()
```

matplotlib:

- set axis margin/limit:

```
ax.margins(x=0.0,y=0.1)
```

- Set limits for x-and y-axis:

```
ax.set(xlim=[0,10.5],yli=[-1.5,1.5])
```

Example

ggplot: draw a scattter plot, we could adjust the paramter inside the geom_point function, such as alpha, size, stroke, etc

```
ggplot(data,
        aes(x = x, y = y)) +
geom_point(alpha = 0.3,
        size = 1.5,
        stroke = 0)  +
labs(title = "title",
        x = "colname",
        y = "count")
```

matplotlib: In matplotlib, we could also adjust the paramteres inside the plt.scatter function such as point size, colors of dots

```
plt.scatter(x, y, s=size, alpha=0.3,
c=colors, alpha=0.5)
plt.show()
```

For more detailed info of these functions, please refer to their official documentation.

Reference:

Rstudio cheatsheets. RStudio. (n.d.). Retrieved October 25, 2021, from https://www.rstudio.com/resources/cheatsheets/.

Visualization with python¶. Matplotlib. (n.d.). Retrieved October 25, 2021, from https://matplotlib.org/