Конспект по теме "Библиотека plotly"

Интерактивные графики

Интерактивный график предполагает возможность взаимодействия с ним. Это значит, что пользователь может выделить элементы, навести на них курсор, получить дополнительную информацию.

Интерактивные графики применять лучше всего:

- когда у вас действительно сложный линейный график или гистограмма, с большим количеством данных. Интерактивность поможет детально изучить значения основных узлов;
- когда для работы с графиком нужна возможность детализации отдельных его частей;
- когда на одном графике нужно разместить большое количество информации — её можно спрятать во всплывающих элементах.

Интерактивный график — своего рода прототип другого варианта визуализации — **дашборда**. Дашборд — сгруппированная по смыслу визуализация данных, которая позволяет быстро ответить на вопросы о продукте и бизнесе. Данные обновляются в реальном времени, все графики — интерактивные. Вы узнаете, как строить дашборды в курсе «Автоматизация».

Библиотека **plotly** позволяет строить интерактивные графики в *Python*. Её основа — **ploty.js.** Она в свою очередь базируется на знаменитой **d3.js** — библиотеке языка веб-разработки *JavaScript* для визуализации данных. Познакомьтесь с ней подробнее здесь.

Базовые графики plotly

Если график создают на основе данных из датафрейма, обращаются к **plotly express** — API Plotly для быстрого доступа к основным методам библиотеки. Так её импортируют:

```
import plotly.express as px
```

Как у *seaborn*, в *plotly* есть встроенные наборы данных.

```
data = px.data.election()
```

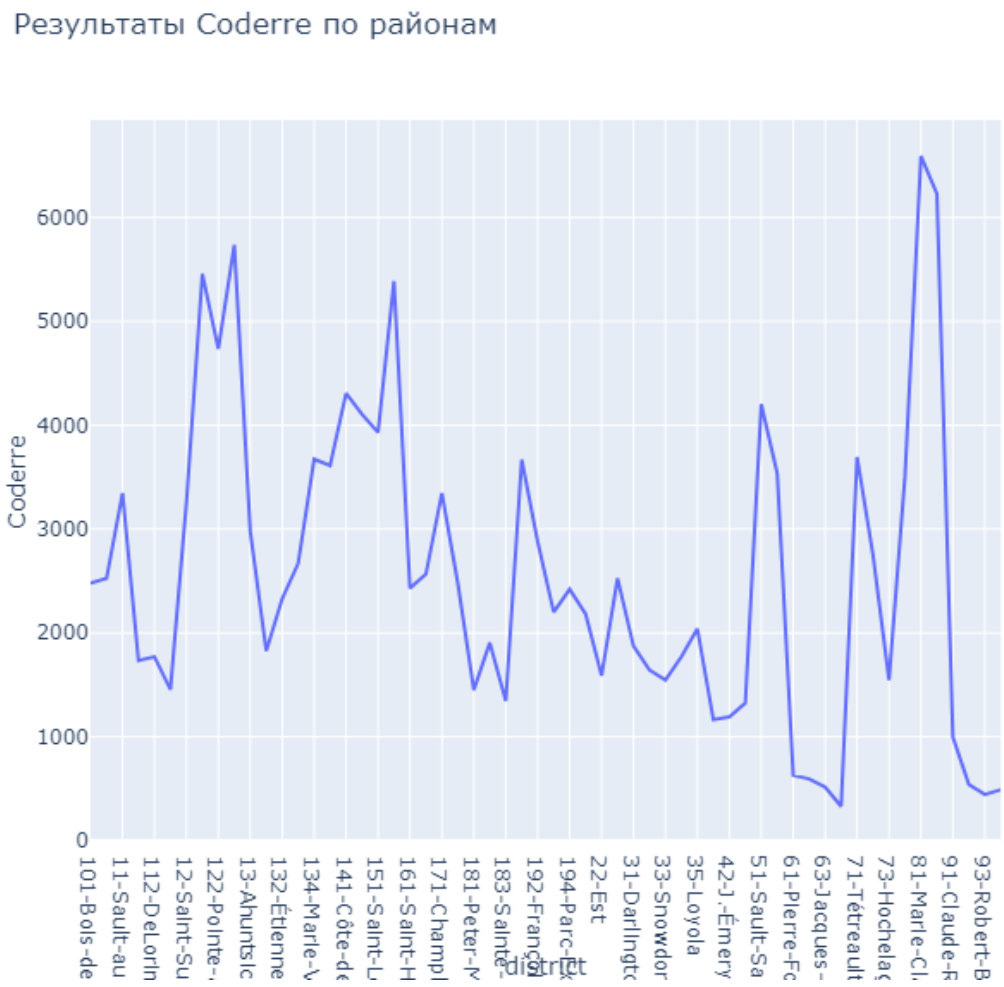
Линейный график

Отличительная особенность интерактивного линейного графика — возможность просмотреть значения прямой при наведении на неё. Чтобы построить линейный график, вызывают метод **line()** с аргументами:

- `data` — данные;
- `X` — данные по оси X;
- `Y` — данные по оси Y;
- `title` — заголовок графика.

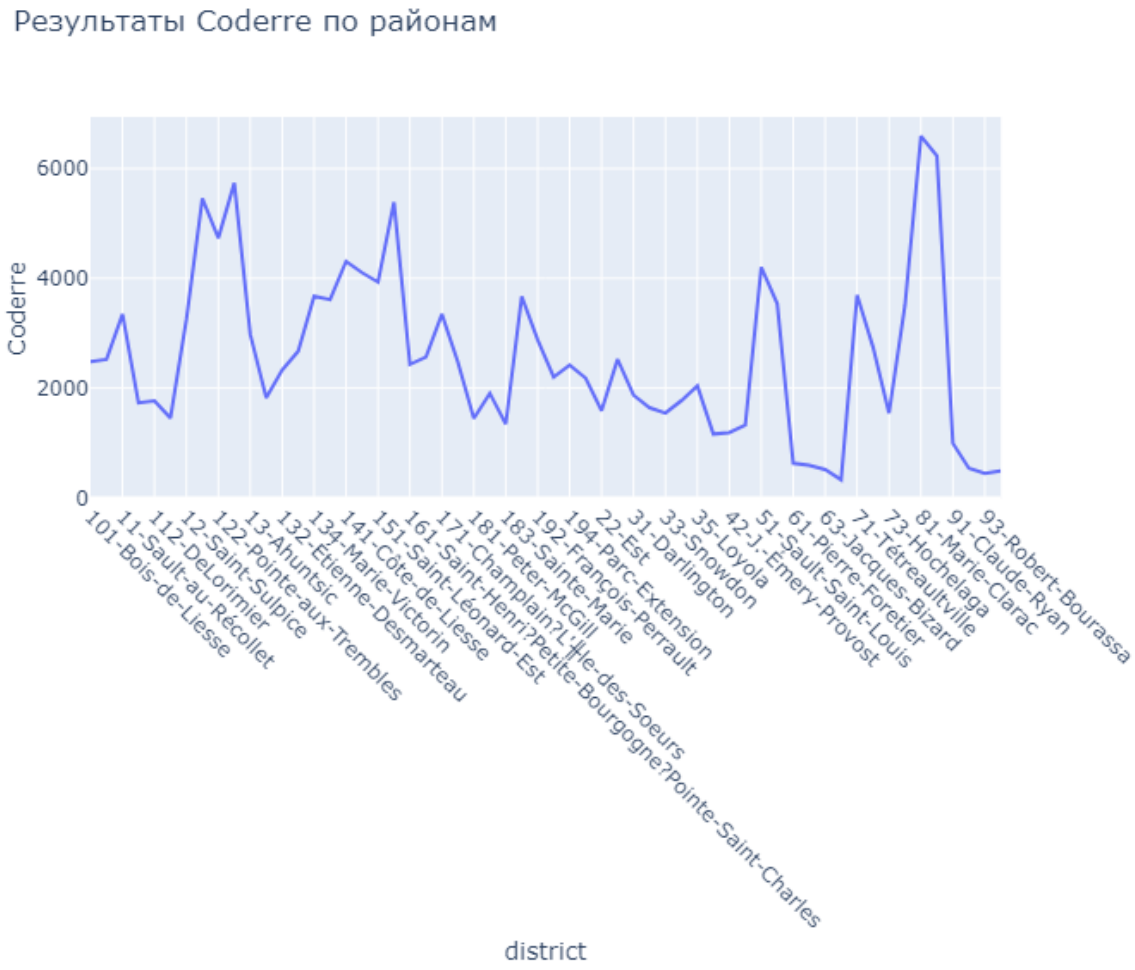
```
import plotly.express as px

fig = px.line(data, x='column1', y='column2', title='Plot title')
fig.show()
```



Повернуть подписи по оси X можно методом **update_xaxes()**, для этого передадим аргументу **tickangle** угол поворота в градусах:

```
fig.update_xaxes(tickangle=45) # повернём подписи по оси X на 45 градусов
fig.show()
```

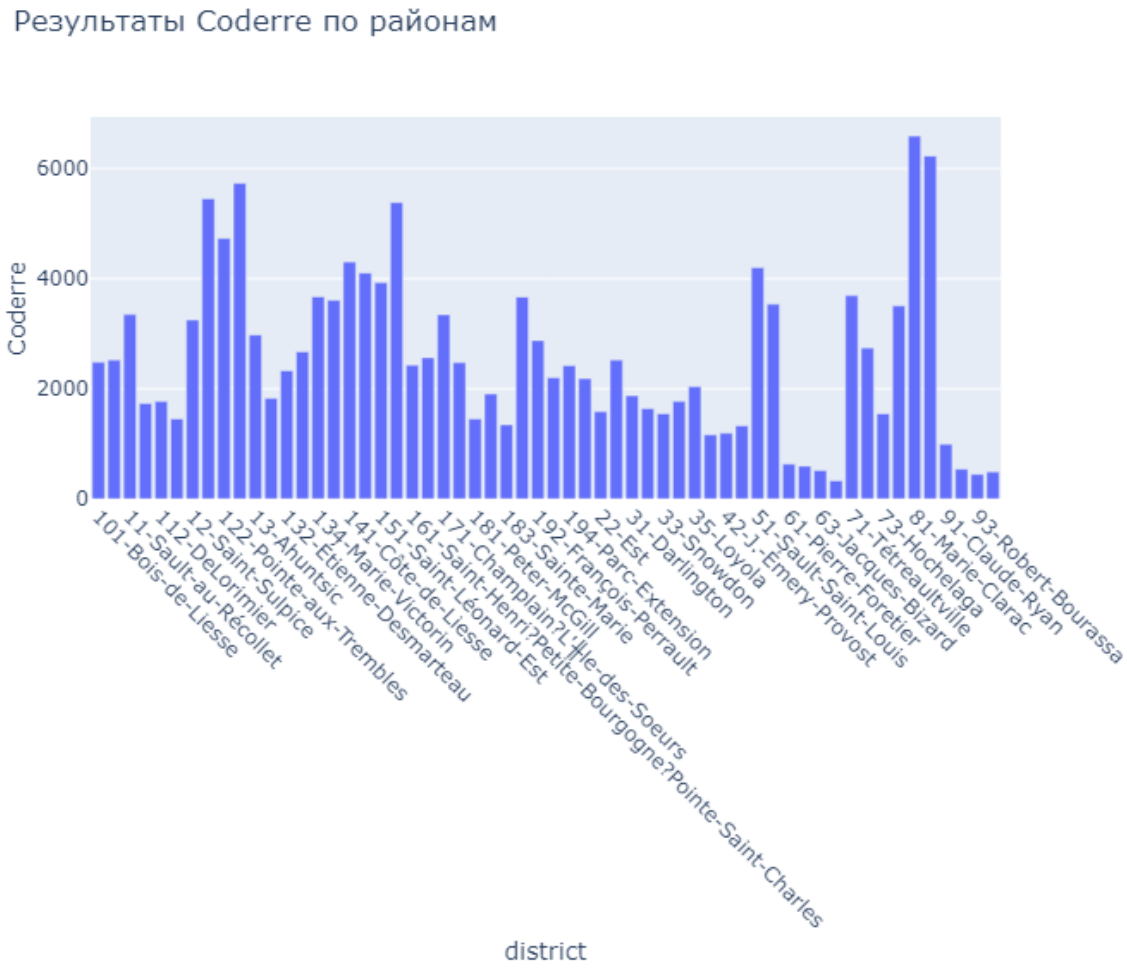


Гистограмма

Гистограмму строят методом `bar()` с теми же аргументами, как и у линейного графика:

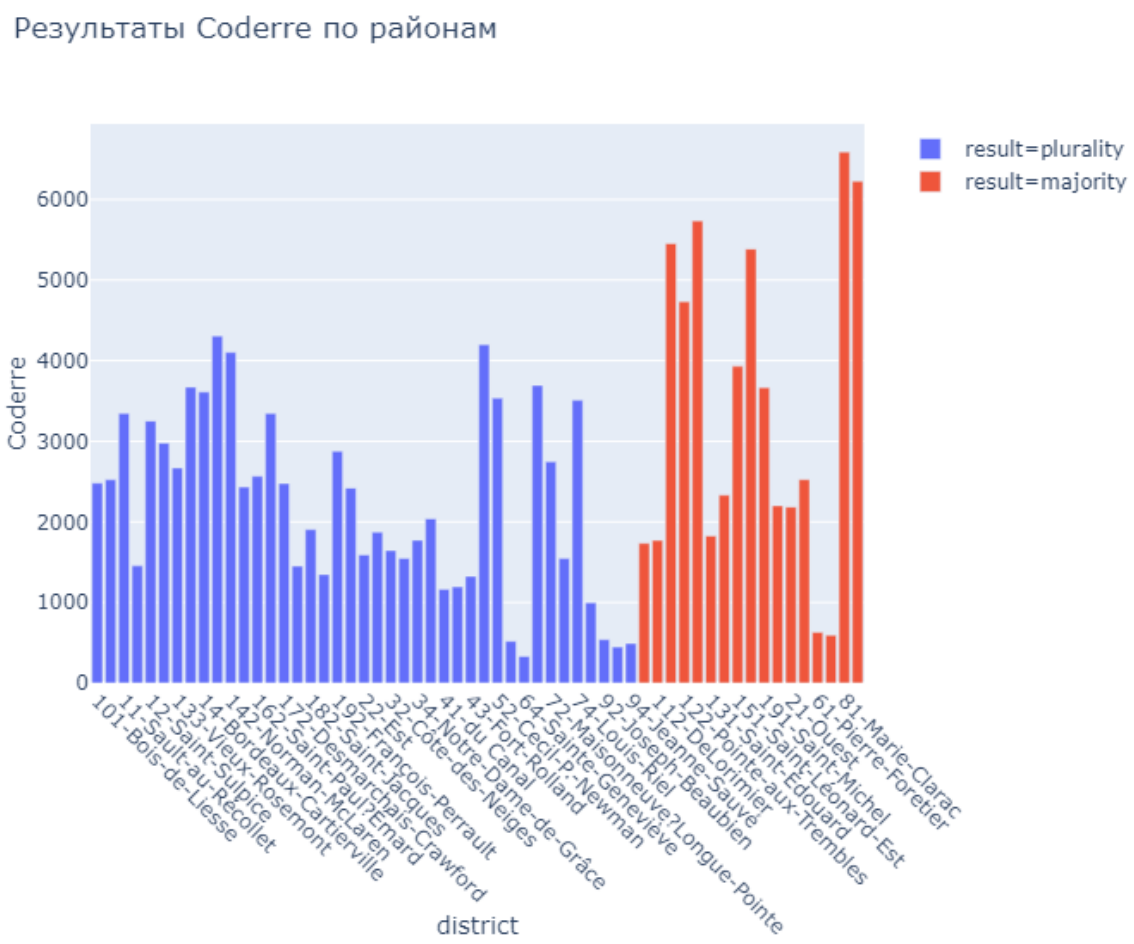
```
import plotly.express as px

fig = px.bar(data, x='column1', y='column2', title='Plot title')
fig.update_xaxes(tickangle=45)
fig.show()
```



При добавлении параметра `color`, наш график будет разделён по значениям указанного столбца:

```
fig = px.bar(data, x='column1', y='column2', color='column3', title='Grouped plot title')
fig.update_xaxes(tickangle=45)
fig.show()
```



Круговая диаграмма

В *plotly* круговую диаграмму строят методом **Pie()** с аргументами **labels** — названиями долей и **values** — значениями долей:

```
from plotly import graph_objects as go

fig = go.Figure(data=[go.Pie(labels=labels_series, values=values_series)])
fig.show()
```

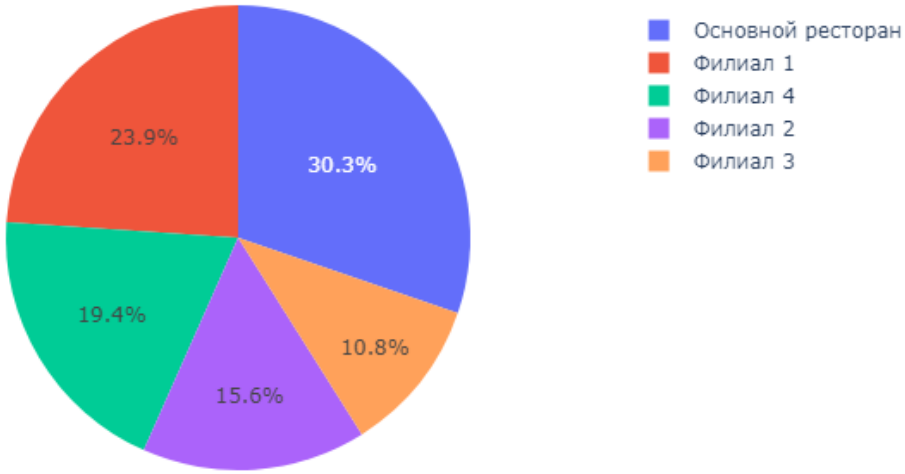


График воронки

За построение воронки отвечает метод **Funnel()**. Его аргументы:

- y — названия шагов воронки;
- x — количество пользователей на шаге.

```
from plotly import graph_objects as go

fig = go.Figure(go.Funnel(y = y_series, x = x_series))
fig.show()
```

Если навести мышкой на каждый шаг воронки, увидим подробную информацию об этом шаге. А именно:

- Сколько процентов составляет число пользователей от первого шага на этом шаге;
- Сколько процентов составляет число пользователей от предыдущего шага;
- Сколько процентов составляет число пользователей от общего числа;

Всё это не нужно рассчитывать дополнительно, достаточно посмотреть на график.