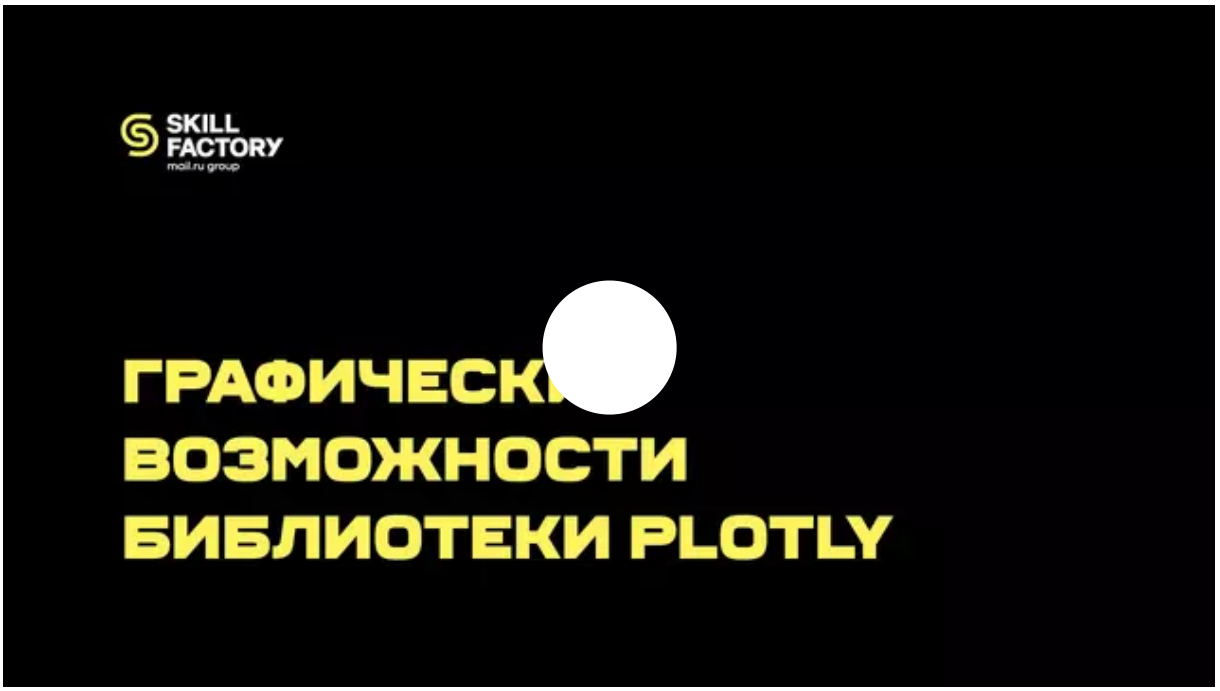
SkillFactory

7. Графические возможности библиотеки Plotly



→ **Скачать ноутбук из скринкаста**

✍ Последней (но не по значимости) библиотекой, которую мы рассмотрим в этом модуле, будет Plotly.

НЕМНОГО О БИБЛИОТЕКЕ

Библиотека *Plotly* является сравнительно новым коммерческим продуктом с бесплатной версией, который создавался специально для *Data Science*, в отличие от относительно старой библиотеки *Matplotlib*, которая изначально разрабатывалась для научных вычислений.

→ Библиотека *Plotly* позволяет строить интерактивные графики, которые можно приближать, отдалять, а также просматривать значения на графике в реальном времени. К тому же в библиотеке собрано огромнейшее количество красочных методов визуализации. У *Plotly* приятный дизайн, а способов работы с ней несколько.

→ С помощью *Plotly* можно делать сложные визуализации с элементами управления, например строить интерактивную *3D*-визуализацию, карту мира и многое другое.

Раньше дата-сайентисты использовали *Plotly* как «тяжёлую артиллерию» по визуализации данных для задач, в которых нужны крайне специфичные графики, которых нет в традиционных библиотеках (например, тепловой карты мира), или в задачах составления красивых дашбордов.

Это было связано с тем, что традиционный *Plotly* имеет огромную функциональность со множеством параметров, в которых легко запутаться. Графики в библиотеке строятся иногда десятками строк кода, отрисовываются медленно и очень сильно нагружают видеокарту.

Сейчас, с появлением новых оптимизированных модулей и надстроек в *Plotly* (express и cufflinks), которые позволили упростить работу с библиотекой, в отрасли наблюдается тенденция постепенного перехода бизнеса к использованию *Plotly* при работе с данными, ведь бесплатную версию библиотеки разрешено применять в коммерческих продуктах.

Библиотека нуждается в дополнительной установке. Для этого введите в командной строке (или командной строке *Anaconda*) следующее:

```
pip install plotly
```

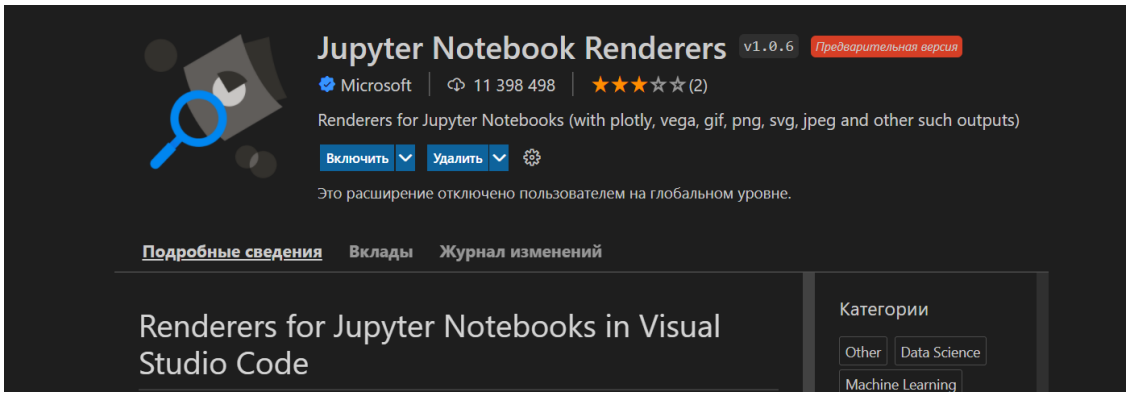
Из библиотеки нам понадобятся модуль `express`, он традиционно импортируется под псевдонимом px.

Также импортируем саму библиотеку, чтобы можно было вывести её версию:

```
import plotly
import plotly.express as px
print(plotly.__version__)
```

Если установка *Plotly* прошла верно, при выполнении кода выше отобразится версия библиотеки. Актуальные версии — *5.3* и выше.

Примечание. Для того чтобы графики *Plotly* отображались в *VS Code*, необходимо установить расширение **Jupyter Notebook Renderers**.



После установки не забудьте перезапустить *VS Code*.

ЭКСПРЕСС-РЕЖИМ

Plotly позволяет строить графики в нескольких режимах. Мы рассмотрим самый новый и подающий надежды — **экспресс-режим**. Его функциональность скромнее, чем у полного режима *Plotly*, но нам её будет более чем достаточно. Для работы в экспресс-режиме предназначен модуль *plotly.express*. Он был выпущен в марте 2019 года и находится в процессе активной разработки.

→ Работа с *plotly.express* напоминает работу с библиотекой *Seaborn*. Отличие лишь в том, что все настройки графика (размеры, подписи осей, текста на графике) прописываются в самом методе.

С помощью экспресс-режима (px) можно строить уже знакомые нам графики:

- line() — линейные графики;
- histogram() — гистограммы;
- scatter() — диаграммы рассеяния;
- box() — коробчатые диаграммы;
- bar() — столбчатые диаграммы;
- pie() — круговые диаграммы.

Также есть множество других графиков — их список вы можете посмотреть в документации.

Примечание. Дальнейшая работа будет вестись с таблицей covid_df — полными данными о статистике распространения вируса *Covid-19*, а также о вакцинации в разных странах.

Рассмотрим процесс визуализации на примере. Посмотрим, как выглядит линейный график, построенный с помощью метода line() из модуля *express*. В документации к методу приведена пара десятков его параметров (они схожи с параметрами других методов) — мы приведём основные из них.

Кликните на плашку, чтобы увидеть информацию ↓

Основные параметры метода line()

Построим график роста зафиксированного числа случаев заражения (*confirmed*), смертей (*deaths*), выздоровлений (*recovered*) и активных случаев (*active*) за всё время. Для этого просуммируем статистику по дням и передадим полученный *DataFrame* в метод line().

Для отображения созданной методом line() фигуры используется метод fig.show():

```
line_data = covid_df.groupby('date', as_index=False).sum()
fig = px.line(
    data_frame=line_data, #DataFrame
    x='date', #ось абсцисс
    y=['confirmed', 'recovered', 'deaths', 'active'], #ось ординат
    height=500, #высота
    width=1000, #ширина
    title='Confirmed, Recovered, Deaths, Active cases over Time'  #заголовок
)
fig.show()
```

В результате мы получаем график, который позволяет:

- получать значения признака в отдельных точках в интерактивном режиме (наведите мышку на точку на графике, чтобы посмотреть информацию о её координатах);

- регулировать отображения конкретных признаков с помощью легенды (отключать и включать данные признаки без изменения кода);
- увеличивать фрагменты, выделяя их мышкой или приближая колёсиком мышки;
- сохранять полученный график в формате PNG напрямую из *Jupyter Notebook* (нажмите на иконку фотоаппарата рядом с графиком).

Давайте рассмотрим ещё один пример — построим столбчатую диаграмму, показывающую ТОП-10 стран по среднему проценту выздоравливающих пациентов (*recover_rate*). Для этого используем метод bar() модуля *express*. Добавим несколько параметров:

- color — группирующий признак, в соответствии с которым будут раскрашены столбцы диаграммы;
- text — текст, который будет подписан на столбцах диаграммы;
- orientation — ориентация графика ('v' — вертикальная, 'h' — горизонтальная).

```
#считаем средний процент выздоровлений для каждой страны
bar_data = covid_df.groupby(
    by='country',
    as_index=False
)[['recover_rate']].mean().round(2).nlargest(10, columns=['recover_rate'])

#строим график
fig = px.bar(
    data_frame=bar_data, #датафрейм
    x="country", #ось x
    y="recover_rate", #ось y
    color='country', #расцветка в зависимости от страны
    text = 'recover_rate', #текст на столбцах
    orientation='v', #ориентация графика
    height=500, #высота
    width=1000, #ширина
    title='Top 10 Countries for Recovery Rate'  #заголовок
)

#отображаем его
fig.show()
```

Примечание. В полученных данных вы можете увидеть «страну» *Diamond Princess*. Напомним, что это круизный лайнер, на котором в начале февраля 2020 года выявили заражённого *Covid-19*, после чего все пассажиры оказались изолированы на судне из-за карантинных мер. Последние пассажиры сошли на берег лишь 1 марта. Подробнее об этом инциденте вы можете прочитать здесь.

А теперь давайте построим что-нибудь, специфичное для библиотеки *Plotly*. Например, график treemap() (древесная, или иерархическая, диаграмма). Такой график используется для исследования показателя, когда число возможных категорий велико (например, число стран в таблице covid_df).

→ Данный вид диаграммы схож с круговой диаграммой и столбчатой диаграммой, однако, в отличие от них, позволяет вместить информацию с гораздо большем числе категорий, не потеряв при этом информативность.

При построении этого графика выделенное пространство разбивается на прямоугольники, соответствующие величине показателя для каждой из категорий: чем меньше прямоугольник, тем меньше в нём значение показателя.

Кликните на плашку, чтобы увидеть информацию ↓

Основные параметры метода treemap()

Но, как говорится, лучше один раз увидеть, чем сто раз прочитать. Построим иерархическую диаграмму для среднего ежедневного показателя выздоровевших пациентов (*daily_recovered*) во всех странах.

```
#считаем среднее ежедневно фиксируемое количество выздоровевших по странам
treemap_data = covid_df.groupby(
    by='country',
    as_index=False
)[['daily_recovered']].mean()

#строим график
fig = px.treemap(
    data_frame=treemap_data, #DataFrame
    path=['country'], #категориальный признак, для которого строится график
    values='daily_recovered', #параметр, который сравнивается
    height=500, #высота
    width=1000, #ширина
    title='Daily Recovered Cases by Country'  #заголовок
)

#отображаем график
fig.show()
```

АНИМАЦИЯ ГРАФИКОВ ВО ВРЕМЕНИ

С помощью *plotly.express* можно строить даже анимированные графики. Мы рассмотрим только базовые приёмы анимации, но на самом деле это очень интересная и глубокая тема. Если вы захотите, то сможете ознакомиться с ней более детально здесь и здесь.

Для нашей задачи отлично подойдёт график под названием choropleth() (тепловая картограмма) — это тепловая карта, которая строится на географической карте мира. Чтобы увидеть, как изменяется значение показателя на карте во времени, можно добавить в график анимацию.

Кликните на плашку, чтобы увидеть информацию ↓

Основные параметры метода choropleth()

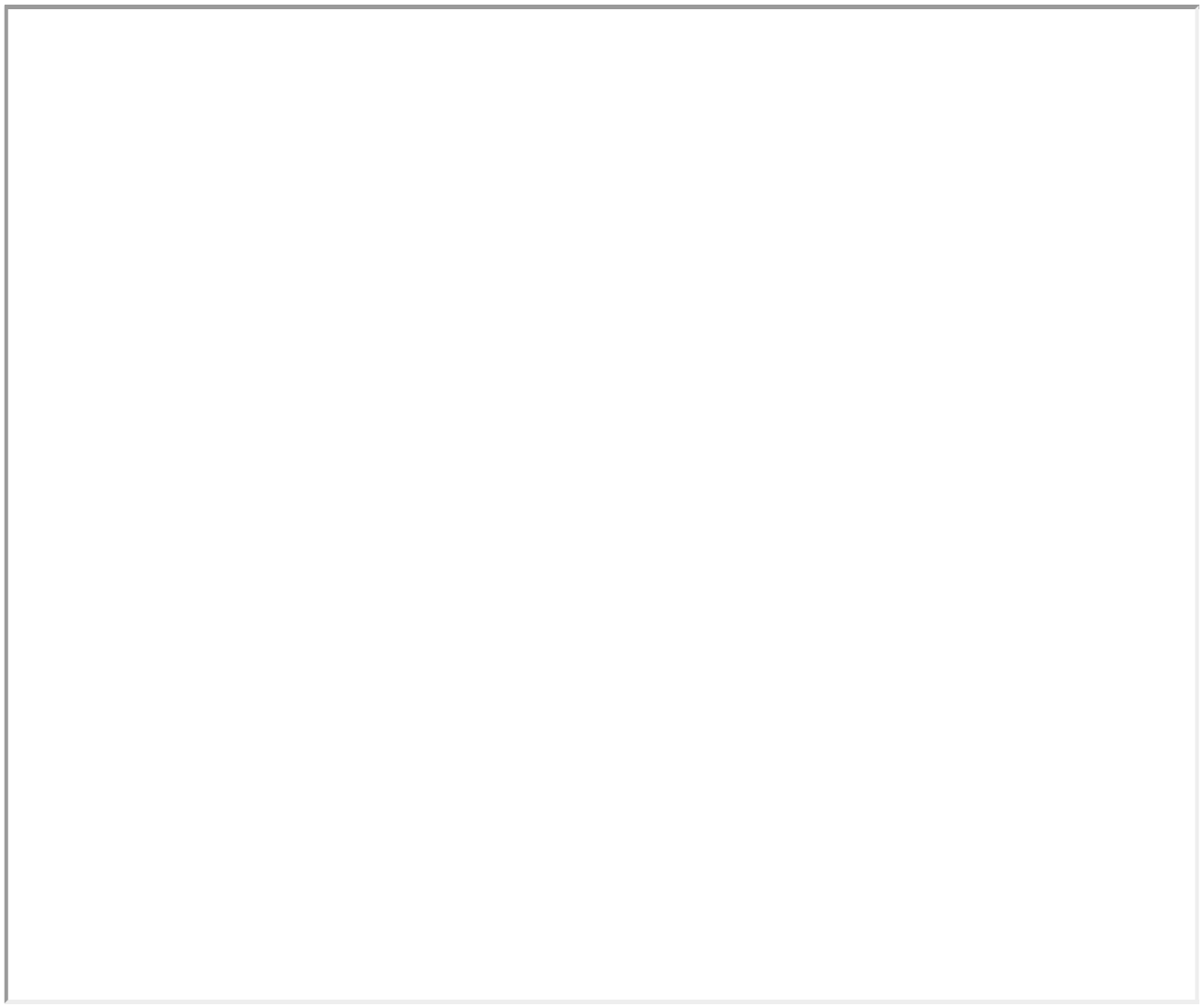
Итак, построим фоновую картограмму, которая покажет распространение (*confirmed*) коронавируса в мире во времени.

Предварительно для правильного отображения на анимационном бегунке даты в таблице covid_df необходимо перевести обратно в строковый тип данных.

```
#преобразуем даты в строковый формат
choropleth_data = covid_df.sort_values(by='date')
choropleth_data['date'] = choropleth_data['date'].astype('string')

#строим график
fig = px.choropleth(
    data_frame=choropleth_data, #DataFrame
    locations="country", #столбец с локациями
    locationmode = "country names", #режим сопоставления локаций с базой Plotly
    color="confirmed", #от чего зависит цвет
    animation_frame="date", #анимационный бегунок
    range_color=[0, 30e6], #диапазон цвета
    title='Global Spread of COVID-19', #заголовок
    width=800, #ширина
    height=500, #высота
    color_continuous_scale='Reds'  #палитра цветов
)

#отображаем график
fig.show()
```



Чтобы воспроизвести анимацию, нажмите на кнопку ***Play*** *или потяните за бегунок.*

ТРЁХМЕРНАЯ ВИЗУАЛИЗАЦИЯ

Настало время познакомиться с *3D*-визуализацией.

> На самом деле общий принцип построения *3D*-графиков ничем не отличается от построения обычных. Просто добавляется ещё один параметр — ось *z* (ось аппликат).

Давайте рассмотрим пример.

Построим *3D*-диаграмму рассеяния, которая покажет, как число ежедневно обнаруживаемых случаев и число ежедневных смертей влияют на желание людей вакцинироваться. Для того чтобы нам было проще рассматривать

диаграмму (точки будут более сгруппированными), построим её в логарифмическом масштабе по осям абсцисс и ординат.

Для построения такой диаграммы используем метод scatter_3d(). Добавим несколько параметров:

- z — параметр по оси аппликат;
- log_x — установка логарифмического масштаба по оси *x*;
- log_y — установка логарифмического масштаба по оси *y*.

Чтобы не перегрузить график, будем строить зависимость только в нескольких странах: США, России, Великобритании, Бразилии и Франции. Наблюдения для каждой страны окрасим разными цветами.

```
#фильтруем таблицу по странам
countries=['United States', 'Russia', 'United Kingdom', 'Brazil', 'France']
scatter_data = covid_df[covid_df['country'].isin(countries)]

#строим график
fig = px.scatter_3d(
    data_frame=scatter_data, #DataFrame
    x = 'daily_confirmed', #ось абсцисс
    y = 'daily_deaths', #ось ординат
    z = 'daily_vaccinations', #ось аппликат
    color='country', #расцветка в зависимости от страны
    log_x=True,
    log_y=True,
    width=1000,
    height=700
)

#отображаем график
fig.show()
fig.write_html("plotly/scatter_3d.html")
```

СОХРАНЕНИЕ ГРАФИКА PLOTLY

Чтобы сохранить интерактивный график, построенный в библиотеке *Plotly*, чаще всего используется метод фигуры fig.write_html('path/to/file.html'), который сохраняет график в формате HTML, после чего вы можете вставлять его на свой сайт, в веб-приложение или просто делиться им с коллегами.

Сохраним график трёхмерной диаграммы рассеяния:

```
fig.write_html("plotly/scatter_3d.html")
```

В результате в папке plotly (которую необходимо создать заранее) у вас появится файл с графиком.

✍ В библиотеке *Plotly* насчитываются десятки различных графиков и сотни возможных настроек к ним. В этом юните мы лишь прикоснулись к этой сокровищнице, погружаться ли в неё с головой — решать вам.

> Принципы построения других базовых типов графиков вы можете найти здесь, а все способы визуализации статистических показателей и зависимостей перечислены здесь.

А теперь предлагаем вам самим **попрактиковаться в использовании библиотеки Plotly** ↓

Задание 7.1

1/1 point (graded)

Выберите режимы, в которых можно работать с библиотекой *Plotly*:

- [x] **A** Основной режим (имеет наибольшую функциональность, позволяет вручную настраивать графики, но имеет более сложную структуру)
- [ ] **B** Умный режим (позволяет библиотеке самой выбирать график)
- [x] **C** Экспресс-режим (позволяет строить графики в упрощённом варианте)
- [x] **D** *Cufflinks* (сторонняя библиотека, которая позволяет строить графики «от лица» *DataFrame*)

✔

Ответ

Верно:
A Верно.
C Верно.
D Верно.

Отправить

Задание 7.2

1/1 point (graded)
Какой параметр в методах модуля `plotly.express` позволяет сгруппировать данные по категориям и отобразить их разными цветами?

- `hue`
- `color` ✔
- `colormap`
- `locationmode`

Ответ
Верно:
Данный параметр отвечает за цветую группировку в методах модуля `plotly.express` .

Отправить

Задание 7.3

1/1 point (graded)
В документации по методу scatter() найдите параметр, который позволяет регулировать размер точек в зависимости от какого-либо признака:

- `size` ✔

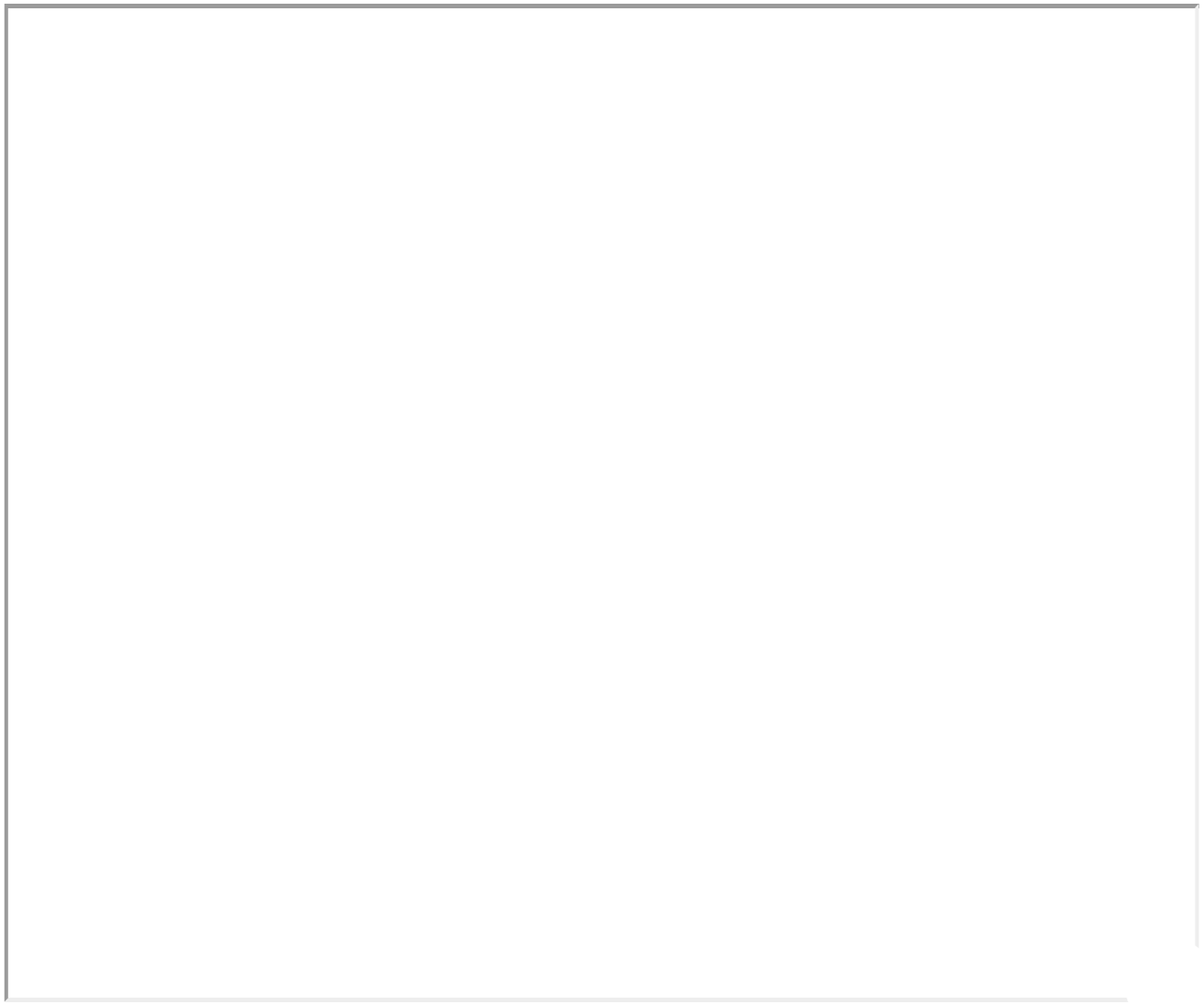
- `color`
- `symbol`
- `size_point`

Отправить

Задание 7.4

1/1 point (graded)

С помощью какого варианта кода можно построить график, представленный ниже?



Прежде чем заносить варианты кода в ноутбук, попробуйте определить верный вариант ответа логически.

- ```
  countries=['United States', 'Russia', 'United Kingdom', 'Brazil', 'France']
  box_data = covid_df[covid_df['country'].isin(countries)]
  fig = px.box(data_frame=box_data, x = 'total_vaccinations', color='country')
  fig.show()
  ```

- ```
  countries=['United States', 'Russia', 'United Kingdom', 'Brazil', 'France']
  bar_data = covid_df[covid_df['country'].isin(countries)]
  fig = px.bar(data_frame=bar_data , x = 'daily_vaccinations_per_million',
  color='country')
  fig.show()
  ```

- ```
  countries=['United States', 'Russia', 'United Kingdom', 'Brazil', 'France']
  box_data = covid_df[covid_df['country'].isin(countries)]
  fig = px.box(data_frame=box_data, x = 'daily_vaccinations_per_million',
  color='country')
  fig.show()
  ```

  ✔

- ```
  countries=['United States', 'Russia', 'United Kingdom', 'Brazil', 'France']
  box_data = covid_df[covid_df['country'].isin(countries)]
  fig = px.box(data_frame=box_data, x = 'daily_vaccinations_per_million')
  fig.show()
  ```

Отправить

Задание 7.5

1/1 point (graded)

Постройте линейный график, который отображает, как изменялось ежедневное количество вакцинированных (*daily_vaccinations*) в мире во времени. Из графика найдите, чему равно количество вакцинированных (в миллионах) 28 февраля 2021 года (2021-02-28). Ответ **округлите до целого числа**.

6

Отправить

Задание 7.6

1/1 point (graded)

Постройте анимированную тепловую картограмму для числа поставленных вакцин во всём мире (*total_vaccinations*). На полученной карте найдите, чему равно количество вакцинированных в Японии (*Japan*) на 24 марта 2021 года (2021-03-24). Ответ приведите **в тысячах (без нулей) и округлите до целого числа**.

> **Примечание.** Если в *jupyter notebook* в *VS Code* не запускается анимация тепловой карты, попробуйте отобразить график командой fig.show(renderer='notebook').

741

Отправить