Язык Python. Часть 2

Лекция 9

"Сцепление" таблиц по общему столбцу

- Есть таблица с измерениями
- Есть колонка с местом измерений
- Выражается как уникальное имя
- Или числовой идентификатор

"Сцепление" таблиц по общему столбцу

- Есть отдельная таблица с данными о местах измерений
- Есть колонка с уникальныим именем или идентификатором
- И другие колонки - географические координаты и т.п.
- Хотим соединить две таблицы

Как соединяем

- Берем строку первой таблицы
- Берем значение в колонке, по которой соединяем
- Берем вторую таблицу
- Берем все ее строки с таким же значением в колонке, по которой соединяем
- Каждую соединяем со строкой из первой
- Помещаем в результат

Как соединяем

- Переходим к следующей строке в первой таблице
- Повторяем
- Перебираем все строки
- Получаем таблицу с колонками из обоих таблиц
- Только соединяющую не дублируем

Варианты

- А если взяли строку первой таблицы
- И ей ничего не соответствует
- Есть измерения по какому-то месту
- Но нет данных о месте

Варианты

- Или наоборот
- Взяли строку второй таблицы
- И ей ничего не соответствует
- Есть данные о месте
- Но нет измерений по нему

Варианты

- Можем отбросить такие варианты с обоих сторон
- Это называется inner
- Можем строку из первой таблицы ("слева") по-любому включить в результат
- Если "справа" не нашлось соответствия, добавить одну строку
- Поля правой таблицы заполнить "пустотами" (NaN)
- Это называется left

Варианты

- Например, есть какое-то измерение
- Есть место измерения
- Нет данных об этом месте
- Добавим все равно измерение
- Только оставим координаты и т.п. пустыми

Варианты

- Можем применить аналогичный принцип в "обратную" сторону
- Например, если есть данные о месте измерения
- Но нет измерений
- Добавим одну строчку с местом в результат
- А конкретные измерения оставим пустыми (NaN)
- Это называется right

Варианты

- Можно применить left и right одновременно
- У такого режима есть свое отдельное название - outer
- По смыслу - противоположность inner
- Есть еще пятый вариант

Варианты

- Берем строку первой таблицы
- Соединяем со всеми строками второй
- Добавляем в результат
- Повторяем для всех строк первой
- Называется cross
- В математическом смысле - декартово произведение

Пример на merge

```
import pandas as pd
import matplotlib.pyplot as plt

air_quality_no2 = pd.read_csv("air_quality_no2_long.csv",
                              parse_dates=True)
air_quality_no2 = air_quality_no2[["date.utc", "location",
                                   "parameter", "value"]]
air_quality_pm25 = pd.read_csv("air_quality_pm25_long.csv",
                               parse_dates=True)
air_quality_pm25 = air_quality_pm25[["date.utc", "location"
                                     "parameter", "value"]]
stations_coord = pd.read_csv("air_quality_stations.csv")

air_quality = pd.concat([air_quality_pm25, air_quality_no2]
air_quality = pd.merge(air_quality, stations_coord, how="le
```

По разным колонкам

- Часто мы сцепляемся по колонкам с одним именем
- Часто, но не всегда
- Иногда колонка в одной таблице называется по-одному
- А в другой - по-другому
- Но содержат одно по сути
- Есть два параметра merge: left_on, right_on

Пример на merge

```
import pandas as pd
import matplotlib.pyplot as plt

air_quality_no2 = pd.read_csv("air_quality_no2_long.csv",
                              parse_dates=True)
air_quality_no2 = air_quality_no2[["date.utc", "location",
                                   "parameter", "value"]]
air_quality_pm25 = pd.read_csv("air_quality_pm25_long.csv",
                               parse_dates=True)
air_quality_pm25 = air_quality_pm25[["date.utc", "location"
                                     "parameter", "value"]]
air_quality = pd.concat([air_quality_pm25, air_quality_no2]
air_quality_parameters = pd.read_csv("air_quality_parameter
air_quality = pd.merge(air_quality, air_quality_parameters,
                       how="left", left_on='parameter', rig
```

Работа с датой/временем

- Колонки с числами pandas умеет распознавать сам
- Колонки с датами распознавать сам не берется
- Но можно помочь
- Функцией to_datetime
- Или именованным параметром parse_dates в read_-методах

Пример на to_datetime

```
import pandas as pd

air_quality = pd.read_csv("air_quality_no2_long.csv")
air_quality = air_quality.rename(columns={"date.utc": "datet
air_quality["datetime"] = pd.to_datetime(air_quality["dateti
print(air_quality["datetime"])
```

Пример на parse_dates

```
1 import pandas as pd
2
3 air_quality = pd.read_csv("air_quality_no2_long.csv", parse_
4 air_quality = air_quality.rename(columns={"date.utc": "datet
5 print(air_quality["datetime"])
```

Операции с датой/временем

- Можем найти максимальное в колонке
- Или минимальное
- Можем вычесть из одного значения другое
- Результат - отдельного типа (Timedelta)

Пример на методы Timestamp

```
1 import pandas as pd
2
3 air_quality = pd.read_csv("air_quality_no2_long.csv", parse_
4 air_quality = air_quality.rename(columns={"date.utc": "datet
5 print(air_quality["datetime"].min())
6 print(air_quality["datetime"].max())
7 print(air_quality["datetime"].max() - air_quality["datetime"
```

Хотим столбец с месяцем измерений

- Возьмем поле dt
- От него поле month
- Выполним эти операции над колонкой "datetime"
- Присвоим колонке "month"

Пример на извлечение месяца

```
1 import pandas as pd
2
3 air_quality = pd.read_csv("air_quality_no2_long.csv", parse_
4 air_quality = air_quality.rename(columns={"date.utc": "datet
5 air_quality = air_quality["datetime"].dt.month
6 print(air_quality.head())
```

Группировка по дню недели

- Хотим получить среднюю концертрацию NO2
- По каждому городу
- На каждый день недели
- Все для этого есть
- Напишем код

Пример на группировку

```
1 import pandas as pd
2
3 air_quality = pd.read_csv("air_quality_no2_long.csv", parse_
4 air_quality = air_quality.rename(columns={"date.utc": "datet
5
6 result = air_quality.groupby(
7     [air_quality["datetime"].dt.weekday, "location"])["value
8 print(result)
```

Визуализация по времени

- Хотим получить почасовой график выбросов NO2
- Сгруппируем по часам
- Сделаем агрегацию по value
- И нарисуем график
- Не забудем plt.show()

Пример на визуализацию

```
 1 import pandas as pd
 2 import matplotlib.pyplot as plt
 3
 4 air_quality = pd.read_csv("air_quality_no2_long.csv", parse
 5 air_quality = air_quality.rename(columns={"date.utc": "date
 6
 7 fig, axs = plt.subplots(figsize=(12, 4))
 8 air_quality.groupby(air_quality["datetime"].dt.hour)["value
 9     kind='bar', rot=0, ax=axs
10 )
11 plt.xlabel("Hour of the day");  # custom x label using Matp
12 plt.ylabel("$NO_2 (mg/m^3)$");
13 plt.show()
```

Время/дата в качестве индекса

- Можем обращаться к индексной колонке через свойство index
- Если это дата/время, можно обращаться к соответствующим полям
- Можно использовать диапазон при индексации DataFrame

Пример на поля индекса

```
 1 import pandas as pd
 2 import matplotlib.pyplot as plt
 3
 4 air_quality = pd.read_csv("air_quality_long.csv", index_col
 5 print(air_quality.pivot_table(
 6     values="value", index="location", columns="parameter",
 7 ))
 8 print(air_quality.pivot_table(
 9     values="value",
10     index="location",
11     columns="parameter",
12     aggfunc="mean",
13     margins=True,
14 ))
```

Пример на диапазон

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 air_quality = pd.read_csv("air_quality_no2_long.csv", parse_
5 air_quality = air_quality.rename(columns={"date.utc": "datet
6
7 no_2 = air_quality.pivot(index="datetime", columns="location
8 no_2["2019-05-20":"2019-05-21"].plot()
9 plt.show()
```

Работа с текстом

- Пусть есть таблица со строковой колонкой
- Можно выполнять поэлементные строковые операции
- Нужно использовать атрибут str
- И вызывать привычные строковые методы

Пример на строковые методы

```
1 import pandas as pd
2
3 titanic = pd.read_csv("titanic.csv")
4 print(titanic["Name"].str.lower())
5 print(titanic["Name"].str.split(","))
6 print(titanic["Name"].str.contains("Countess"))
```

Подведем итог

- pandas - библиотека для табличных форматов данных
- В первую очередь - csv, excel
- Именованные колонки
- Массированные операции
- Выборка данных по критериям

Близкая к pandas библиотека

- Есть близкая к pandas библиотека
- Тоже про таблицы
- Но без именованные колонок
- И с акцентом на числа
- И быстрые вычисления

numpy

- python3 -m pip install numpy
- https://www.numpy.org
- Импортируем numpy
- Идиоматичный псевдоним - np

ndarray

- Базовый тип данных - ndarray
- Многомерный массив
- Нет отдельного типа для столбца
- ndarray может иметь любую размерность

shape

- Что-то типа кортежа положительных чисел
- Описывает размерности экземпляров ndarray
- Например, двумерный ndarray может иметь shape (3, 5)
- Можно понимать как 3 строки и 5 столбцов

dtype

- Тип данных
- Идея близка pandas
- Набор типов отличается
- Более разнообразные числовые типы

Создание ndarray

- Из нулей - np.zeros(...)
- Из единиц - np.ones(...)
- Случайные значения - np.ndarray()
- Как минимум, надо задать shape

Простейший пример

```
1 import numpy as np
2
3 data = np.ones(shape=(3, 2))
4 print(data)
5 data = np.zeros(shape=(3, 2, 4))
6 print(data)
7 data = np.ndarray(shape=(5, )) # случайный вектор
8 print(data)
```