Манипулирование данными с помощью Pandas

Агрегация фреймов данных

Среднее и медиана

Статистика обзора - это именно то, что звучит - она подводит итог многим числам в одной статистике. Например, среднее, медиана, минимум, максимум и стандартное отклонение - это статистика обзора. Рассчет статистики обзора позволяет лучше понять ваши данные, даже если их много.

`sales` доступен, и pandas загружен как pd.

Инструкции

- Изучите свой новый DataFrame, сначала распечатав первые несколько строк DataFrame sales.
- Выведите информацию о столбцах в `sales`.
- Выведите среднее значение столбца `weekly_sales`.
- Выведите медиану столбца `weekly_sales`.

```
import pandas as pd
sales = pd.read_csv('datasets/sales_subset.csv')

# Распечатать первые строки DataFrame sales
print(sales.head())

# Вывести информацию о DataFrame sales
print(sales.info())

# Вывести среднее значение weekly_sales
print(sales["weekly_sales"].mean())

# Вывести медиану weekly_sales
print(sales["weekly_sales"].median())
```

```
   Unnamed: 0  store type  department        date  weekly_sales  is_holiday  \
0           0      1    A           1  2010-02-05      24924.50       False
1           1      1    A           1  2010-03-05      21827.90       False
2           2      1    A           1  2010-04-02      57258.43       False
3           3      1    A           1  2010-05-07      17413.94       False
4           4      1    A           1  2010-06-04      17558.09       False

   temperature_c  fuel_price_usd_per_l  unemployment
0       5.727778              0.679451         8.106
1       8.055556              0.693452         8.106
2      16.816667              0.718284         7.808
3      22.527778              0.748928         7.808
4      27.050000              0.714586         7.808
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10774 entries, 0 to 10773
Data columns (total 10 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
```

```
 0   Unnamed: 0            10774 non-null  int64
 1   store                 10774 non-null  int64
 2   type                  10774 non-null  object
 3   department            10774 non-null  int64
 4   date                  10774 non-null  object
 5   weekly_sales          10774 non-null  float64
 6   is_holiday            10774 non-null  bool
 7   temperature_c         10774 non-null  float64
 8   fuel_price_usd_per_l  10774 non-null  float64
 9   unemployment          10774 non-null  float64
dtypes: bool(1), float64(4), int64(3), object(2)
memory usage: 768.2+ KB
None
23843.95014850566
12049.064999999999
```

Суммирование дат

Сводная статистика также может быть рассчитана для столбцов с датами, значения которых имеют тип данных `datetime64`. Некоторые сводные статистики, например, среднее значение, не имеют особого смысла для дат, но другие очень полезны, например, минимум и максимум, которые позволяют увидеть диапазон времени, охватываемый вашими данными.

`sales` доступен, и pandas загружен как pd.

Инструкции

- Выведите максимум столбца даты (date).
- Выведите минимум столбца даты (date).

```
# Вывести максимум столбца даты
print(sales["date"].max())

# Вывести минимум столбца даты
print(sales["date"].min())
```

```
2012-10-26
2010-02-05
```

Эффективные сводки

Хотя у `pandas` и `NumPy` есть множество функций, иногда вам может понадобиться другая функция для сводки ваших данных.

Метод `.agg()` позволяет применять ваши собственные пользовательские функции к DataFrame, а также применять функции к более чем одному столбцу DataFrame одновременно, делая ваши агрегации очень эффективными. Например,

```
df['столбец'].agg(функция)
```

В пользовательской функции для этого упражнения "IQR" означает интерквартильный размах, который представляет собой разницу между 75-м и 25-м процентилем. Это альтернатива стандартному отклонению, полезная, если в ваших данных есть выбросы.

`sales` доступен, а pandas загружен как pd.

Инструкции

1. Используйте определенную для вас пользовательскую функцию `iqr` вместе с `.agg()`, чтобы вывести интерквартильный размах (IQR) столбца `temperature_c` из `sales`.

```
# Пользовательская функция IQR
def iqr(столбец):
    return столбец.quantile(0.75) - столбец.quantile(0.25)

# Вывести IQR столбца temperature_c
print(sales["temperature_c"].agg(iqr))
```

```
16.583333333333336
```

1. Обновите выбор столбцов, используя пользовательскую функцию `iqr` с `.agg()`, чтобы вывести интерквартильный размах (IQR) `temperature_c`, `fuel_price_usd_per_l` и `unemployment`, в таком порядке.

```
# Пользовательская функция IQR
def iqr(столбец):
    return столбец.quantile(0.75) - столбец.quantile(0.25)

# Обновить для вывода IQR temperature_c, fuel_price_usd_per_l и unemployment
print(sales[["temperature_c", "fuel_price_usd_per_l", "unemployment"]].agg(iqr))
```

```
temperature_c           16.583333
fuel_price_usd_per_l     0.073176
unemployment             0.565000
dtype: float64
```

1. Обновите функции агрегации, вызываемые `.agg()`: включите `iqr` и `np.median` в таком порядке.

```
# Импортировать NumPy и создать пользовательскую функцию IQR
import numpy as np
def iqr(столбец):
    return столбец.quantile(0.75) - столбец.quantile(0.25)

# Обновить для вывода IQR и медианы temperature_c, fuel_price_usd_per_l и
unemployment
print(sales[["temperature_c", "fuel_price_usd_per_l", "unemployment"]].agg([iqr,
'median']))
```

```
        temperature_c  fuel_price_usd_per_l  unemployment
iqr         16.583333              0.073176         0.565
median      16.966667              0.743381         8.099
```

Статистика по накоплению

Кумулятивная статистика также может быть полезной для отслеживания суммарной статистики с течением времени. В этом упражнении вы будете вычислять кумулятивную сумму и кумулятивный максимум еженедельных продаж отдела, что позволит вам определить, какова была общая сумма продаж на текущий момент, а также каковы были максимальные еженедельные продажи на данный момент.

Создан DataFrame под названием `sales_1_1`, который содержит данные о продажах для отдела 1 магазина 1. pandas загружен как pd.

Инструкции

- Отсортировать строки `sales_1_1` по столбцу `date` в порядке возрастания.
- Получить накопительную сумму `weekly_sales` и добавить ее как новый столбец `sales_1_1` под названием `cum_weekly_sales`.
- Получить накопительный максимум `weekly_sales` и добавить его как столбец под названием `cum_max_sales`.
- Вывести столбцы `date`, `weekly_sales`, `cum_weekly_sales` и `cum_max_sales`.

```
# Сортировка sales_1_1 по date
sales_1_1 = sales.sort_values("date")

# Получение накопительной суммы weekly_sales, добавление в качестве столбца
cum_weekly_sales
sales_1_1["cum_weekly_sales"] = sales_1_1["weekly_sales"].cumsum()

# Получение накопительного максимума weekly_sales, добавление в качестве столбца
cum_max_sales
sales_1_1["cum_max_sales"] = sales_1_1["weekly_sales"].cummax()

# Просмотр столбцов, которые вы вычислили
print(sales_1_1[["date", "weekly_sales", "cum_weekly_sales", "cum_max_sales"]])
```

```
            date  weekly_sales  cum_weekly_sales  cum_max_sales
0     2010-02-05      24924.50      2.492450e+04       24924.50
6437  2010-02-05      38597.52      6.352202e+04       38597.52
1249  2010-02-05       3840.21      6.736223e+04       38597.52
6449  2010-02-05      17590.59      8.495282e+04       38597.52
6461  2010-02-05       4929.87      8.988269e+04       38597.52
...          ...           ...               ...            ...
3592  2012-10-05        440.00      2.568932e+08      293966.05
8108  2012-10-05        660.00      2.568938e+08      293966.05
10773 2012-10-05        915.00      2.568947e+08      293966.05
6257  2012-10-12          3.00      2.568947e+08      293966.05
3384  2012-10-26        -21.63      2.568947e+08      293966.05

[10774 rows x 4 columns]
```

Удаление дубликатов

Удаление дубликатов - это важный навык для получения точных подсчетов, потому что часто необходимо избежать учета одного и того же элемента несколько раз. В этом упражнении вы создадите несколько новых DataFrame, используя уникальные значения из `sales`.

Доступен DataFrame `sales`, и pandas импортирован как pd.

Инструкции

- Удалите строки из `sales` с повторяющимися парами `store` и `type`, сохраните как `store_types` и выведите `head`.
- Удалите строки из `sales` с повторяющимися парами `store` и `department`, сохраните как `store_depts` и выведите `head`.
- Выберите строки, соответствующие неделям праздников, используя столбец `is_holiday`, и удалите повторяющиеся `dates`, сохраните как `holiday_dates`.
- Выберите столбец `date` из `holiday_dates` и выведите его.

```
# Удалить повторяющиеся комбинации магазинов/типов
store_types = sales.drop_duplicates(subset=["store", "type"])

print(store_types.head())

# Удалить повторяющиеся комбинации магазинов/отделов
store_depts = sales.drop_duplicates(subset=["store", "department"])
print(store_depts.head())

# Выбрать строки, где is_holiday равно True, и удалить повторяющиеся даты
holiday_dates = sales[sales["is_holiday"]].drop_duplicates(subset=["date"])

# Вывести столбец date из holiday_dates
print(holiday_dates[["date"]])
```

```
      Unnamed: 0  store type  department        date  weekly_sales  \
0              0      1    A           1  2010-02-05      24924.50
901          901      2    A           1  2010-02-05      35034.06
1798        1798      4    A           1  2010-02-05      38724.42
2699        2699      6    A           1  2010-02-05      25619.00
3593        3593     10    B           1  2010-02-05      40212.84

      is_holiday  temperature_c  fuel_price_usd_per_l  unemployment
0          False       5.727778              0.679451         8.106
901        False       4.550000              0.679451         8.324
1798       False       6.533333              0.686319         8.623
2699       False       4.683333              0.679451         7.259
3593       False      12.411111              0.782478         9.765
    Unnamed: 0  store type  department        date  weekly_sales  is_holiday  \
0            0      1    A           1  2010-02-05      24924.50       False
12          12      1    A           2  2010-02-05      50605.27       False
24          24      1    A           3  2010-02-05      13740.12       False
36          36      1    A           4  2010-02-05      39954.04       False
48          48      1    A           5  2010-02-05      32229.38       False

    temperature_c  fuel_price_usd_per_l  unemployment
0        5.727778              0.679451         8.106
12       5.727778              0.679451         8.106
24       5.727778              0.679451         8.106
36       5.727778              0.679451         8.106
48       5.727778              0.679451         8.106
           date
498   2010-09-10
691   2011-11-25
2315  2010-02-12
6735  2012-09-07
6810  2010-12-31
6815  2012-02-10
6820  2011-09-09
```

Подсчет категориальных переменных

Подсчет - отличный способ получить обзор данных и заметить любопытные особенности, которые вы могли бы упустить иначе. В этом упражнении вы посчитаете количество каждого

типа магазина и количество каждого номера отдела, используя DataFrame, созданные в предыдущем упражнении:

DataFrame `store_types` и `store_depts`, созданные в последнем упражнении, доступны, и `pandas` импортирован как `pd`.

Инструкции

- Посчитайте количество магазинов каждого типа в `store_types`.
- Посчитайте долю магазинов каждого типа в `store_types`.
- Посчитайте количество различных отделов в `store_depts`, отсортировав подсчеты по убыванию.
- Посчитайте долю различных отделов в `store_depts`, отсортировав доли по убыванию.

```
# Подсчитать количество магазинов каждого типа
store_counts = store_types["type"].value_counts()
print(store_counts)

# Получить долю магазинов каждого типа
store_props = store_types["type"].value_counts(normalize=True)
print(store_props)

# Подсчитать количество каждого номера отдела и отсортировать
dept_counts_sorted = store_depts["department"].value_counts(sort=True)
print(dept_counts_sorted)

# Получить долю отделов каждого номера и отсортировать
dept_props_sorted = store_depts["department"].value_counts(sort=True, normalize=True)
print(dept_props_sorted)
```

```
type
A    11
B     1
Name: count, dtype: int64
type
A    0.916667
B    0.083333
Name: proportion, dtype: float64
department
1     12
55    12
72    12
71    12
67    12
      ..
37    10
48     8
50     6
39     4
43     2
Name: count, Length: 80, dtype: int64
department
1     0.012917
55    0.012917
72    0.012917
71    0.012917
```

```
67    0.012917
        ...
37    0.010764
48    0.008611
50    0.006459
39    0.004306
43    0.002153
Name: proportion, Length: 80, dtype: float64
```

Какой процент продаж произошел в каждом типе магазина?

Хотя `.groupby()` полезен, вы можете вычислить группированные сводные статистики и без его использования.

Walmart различает три типа магазинов: "суперцентры", "скидочные магазины" и "рынки в районе", закодированные в этом наборе данных как тип "A", "B" и "C". В этом упражнении вы будете вычислять общие продажи в каждом типе магазина, не используя `.groupby()`. Затем вы сможете использовать эти числа, чтобы увидеть, какая часть общих продаж Walmart пришлась на каждый тип.

Доступен DataFrame `sales`, и pandas импортирован как pd.

Инструкции

- Посчитайте общие еженедельные продажи по всему набору данных.
- Выберите магазины типа "A" и посчитайте их общие еженедельные продажи.
- Сделайте то же самое для магазинов типа "B" и "C".
- Объедините результаты типов A/B/C в список и разделите на `sales_all`, чтобы получить долю продаж по типу.

```
# Вычисление общих еженедельных продаж
sales_all = sales["weekly_sales"].sum()

# Выбор магазинов типа A и вычисление общих еженедельных продаж
sales_A = sales[sales["type"] == "A"]["weekly_sales"].sum()

# Выбор магазинов типа B и вычисление общих еженедельных продаж
sales_B = sales[sales["type"] == "B"]["weekly_sales"].sum()

# Выбор магазинов типа C и вычисление общих еженедельных продаж
sales_C = sales[sales["type"] == "C"]["weekly_sales"].sum()

# Получение доли для каждого типа
sales_propn_by_type = [sales_A, sales_B, sales_C] / sales_all
print(sales_propn_by_type)
```

```
[0.9097747 0.0902253 0.       ]
```

Вычисления с помощью .groupby()

Метод `.groupby()` значительно упрощает жизнь. В этом упражнении вы выполните те же вычисления, что и в прошлый раз, за исключением того, что будете использовать метод `.groupby()`. Вы также выполните вычисления на данных, сгруппированных по двум переменным, чтобы увидеть, отличаются ли продажи в зависимости от типа магазина и является ли неделя праздничной или нет.

Доступен DataFrame `sales`, и `pandas` загружен как `pd`.

Инструкции 1/2

- Сгруппируйте данные `sales` по `"type"`, возьмите сумму `"weekly_sales"` и сохраните как `sales_by_type`.
- Рассчитайте долю продаж для каждого типа магазина, разделив на сумму `sales_by_type`. Присвойте переменной `sales_propn_by_type`.

```
# Группировка по типу; вычисление общих еженедельных продаж
sales_by_type = sales.groupby("type")["weekly_sales"].sum()

# Получение доли для каждого типа
sales_propn_by_type = sales_by_type / sum(sales_by_type)
print(sales_propn_by_type)
```

```
type
A    0.909775
B    0.090225
Name: weekly_sales, dtype: float64
```

Инструкции 2/2

Сгруппируйте данные `sales` по `"type"` и `"is_holiday"`, возьмите сумму `weekly_sales` и сохраните как `sales_by_type_is_holiday`

```
### # Из предыдущего шага

sales_by_type = sales.groupby("type")["weekly_sales"].sum()

# Группировка по типу и is_holiday; вычисление общих еженедельных продаж
sales_by_type_is_holiday = sales.groupby(["type",
"is_holiday"])["weekly_sales"].sum()
print(sales_by_type_is_holiday)
```

```
type  is_holiday
A     False         2.336927e+08
      True          2.360181e+04
B     False         2.317678e+07
      True          1.621410e+03
Name: weekly_sales, dtype: float64
```

Несколько сгруппированных сводных статистик

Ранее в этой главе вы видели, что метод `.agg()` полезен для вычисления нескольких статистик по нескольким переменным. Он также работает с сгруппированными данными. NumPy, который импортирован как `np`, имеет множество различных функций сводных статистик, включая: `np.min, np.max, np.mean и np.median.`

Доступен DataFrame `sales`, и `pandas` импортирован как `pd`.

Инструкции

- Получите `min, max, mean` и `median для weekly_sales` для каждого типа магазина, используя `.groupby()` и `.agg()`. Сохраните это как `sales_stats`. Обязательно используйте функции numpy!

- Получите `min, max, mean` и `median` для `unemployment` и `fuel_price_usd_per_l` для каждого типа магазина. Сохраните это как `unemp_fuel_stats`.

```
# Для каждого типа магазина агрегируйте weekly_sales: получите минимум, максимум, среднее и медиану
sales_stats = sales.groupby("type")["weekly_sales"].agg(['min', 'max', 'mean', 'median'])

# Выведите на экран sales_stats
print(sales_stats)

# Для каждого типа магазина агрегируйте unemployment и fuel_price_usd_per_l: получите минимум, максимум, среднее и медиану
unemp_fuel_stats = sales.groupby("type")[["unemployment", "fuel_price_usd_per_l"]].agg(['min', 'max', 'mean', 'median'])

# Выведите на экран unemp_fuel_stats
print(unemp_fuel_stats)
```

```
         min        max          mean    median
type
A    -1098.0  293966.05  23674.667242  11943.92
B     -798.0  232558.51  25696.678370  13336.08
     unemployment                         fuel_price_usd_per_l           \
              min    max      mean median                  min       max
type
A          3.879  8.992  7.972611  8.067             0.664129  1.107410
B          7.170  9.765  9.279323  9.199             0.760023  1.107674


          mean    median
type
A     0.744619  0.735455
B     0.805858  0.803348
```

Сводная таблица по одной переменной

Таблицы сводных данных - стандартный способ агрегирования информации в электронных таблицах.

В библиотеке pandas сводные таблицы по сути являются еще одним способом выполнения групповых вычислений. То есть метод `.pivot_table()` представляет собой альтернативу `.groupby()`.

В этом упражнении вы будете использовать `.pivot_table()` для выполнения расчетов и воспроизведения вычислений, проведенных в предыдущем уроке с использованием `.groupby()`.

Доступен DataFrame `sales`, и pandas импортирован как pd.

1/3

Получите среднее значение `weekly_sales` по типу с помощью `.pivot_table()` и сохраните как `mean_sales_by_type`

```
# Сводная таблица для средних еженедельных продаж для каждого типа магазина
mean_sales_by_type = sales.pivot_table(values="weekly_sales", index="type")
```

```
# Вывод mean_sales_by_type на экран
print(mean_sales_by_type)
```

```
      weekly_sales
type
A     23674.667242
B     25696.678370
```

2/3

Получите среднее и медиану (используя функции NumPy) для `weekly_sales` по типу с помощью `.pivot_table()` и сохраните как `mean_med_sales_by_type`.

```
# Сводная таблица для средних еженедельных продаж по типу магазина и празднику
mean_sales_by_type_holiday = sales.pivot_table(values="weekly_sales", index="type",
columns="is_holiday")

# Вывод mean_sales_by_type_holiday на экран
print(mean_sales_by_type_holiday)
```

```
is_holiday         False      True
type
A           23768.583523  590.04525
B           25751.980533  810.70500
```

Заполнение отсутствующих значений и подсчет суммы значений с помощью сводных таблиц

Метод `.pivot_table()` имеет несколько полезных аргументов, включая `fill_value` и `margins`.

- `fill_value` заменяет отсутствующие значения на реальное значение (известное как импутация). Чем заменять отсутствующие значения - это достаточно большая тема, чтобы иметь свой собственный курс (работа с отсутствующими данными в Python), но самый простой способ - заменить их фиктивным значением.
- `margins` - это сокращение для случая, когда вы создали сводную таблицу по двум переменным, но также хотели бы создать сводные таблицы для каждой из этих переменных по отдельности: он дает итоги строк и столбцов содержимого сводной таблицы. В этом упражнении вы попрактикуетесь в использовании этих аргументов, чтобы улучшить свои навыки работы со сводными таблицами, что поможет вам более эффективно обрабатывать числовые данные!

Доступен DataFrame `sales`, и `pandas` импортирован как pd.

Инструкции 1/2

- Выведите средние еженедельные продажи по отделу и типу, заполнив все отсутствующие значения нулями.

```
# Вывести средние еженедельные продажи по отделу и типу; заполнить отсутствующие
значения нулями

print(sales.pivot_table(values="weekly_sales", index="department", columns="type",
fill_value=0))
```

```
type                    A             B
department
1            30961.725379   44050.626667
```

```
2            67600.158788  112958.526667
3            17160.002955   30580.655000
4            44285.399091   51219.654167
5            34821.011364   63236.875000
...                   ...            ...
95          123933.787121   77082.102500
96           21367.042857    9528.538333
97           28471.266970    5828.873333
98           12875.423182     217.428333
99             379.123659       0.000000

[80 rows x 2 columns]
```

Инструкции 2/2

Выведите средние еженедельные продажи по отделу и типу, заполнив все отсутствующие значения нулями и суммируйте все строки и столбцы.

```
# Вывести средние еженедельные продажи по отделу и типу; заполнить отсутствующие значения нулями; подсчитать сумму всех строк и столбцов
print(sales.pivot_table(values="weekly_sales", index="department", columns="type", fill_value=0, margins=True))
```

```
type                  A              B           All
department
1           30961.725379   44050.626667  32052.467153
2           67600.158788  112958.526667  71380.022778
3           17160.002955   30580.655000  18278.390625
4           44285.399091   51219.654167  44863.253681
5           34821.011364   63236.875000  37189.000000
...                  ...            ...           ...
96          21367.042857    9528.538333  20337.607681
97          28471.266970    5828.873333  26584.400833
98          12875.423182     217.428333  11820.590278
99            379.123659       0.000000    379.123659
All         23674.667242   25696.678370  23843.950149

[81 rows x 3 columns]
```