Pandas - 2

Summary Statistics

- df[‘col’].mean()
- .meadian(), .mode(), .min(), .max(), .var(), .std(), .sum(), .quantile()
- .agg() method allows us to perform an operation inside a function on the specified dataframe
- def pcr30(column):
  - return column.quantile(0.3) # returns 30 quantile
- dogs[“weight_kg”].agg(pct30)
- We can also pass a list of functions to agg and this would return the results sequentially.
- dogs[[“weight_kg”, ‘height’]].agg([pct30, pct40])
- The .agg() method allows you to apply your own custom functions to a DataFrame, as well as apply functions to more than one column of a DataFrame at once, making your aggregations super-efficient. For example,

df['column'].agg(function)

- We can pass in sort, normalize, dropna in value_counts method

Cumulative Statistics - cumsum(), cummax(), cummin(), cumprod()

- We can make use of the cumSum() method to get the cumulative sum of the whole dataset
- dogs[‘weight’].cumsum()
  - 24, 48, 72 - Here 24 is the weight of first dog, 48 us the weight of the first 2 dogs and 72 is the weight of the first 3 dogs

Counting

- **Dropping Duplicates**
  - **vet_visits.drop_duplicates(subset = “name”)**
    - **All the names that occur more than once in the name column have their row dropped**
- **Dropping Duplicate Pairs**
- **unique_dogs = vet_visits_duplicates(subset = [“name”, “breed”])**
  - **All the rows having the same name and breed columns are dropped till we have only 1 of each pair left**
- **value_counts(normalize = True)**
  - **Returns the number of values**
  - **When normalize is True, we return a normalized count of the total, we can use this to calculate the percentage as the norm sums to 1.**

code Snippets

```
# Drop duplicate store/type combinations
store_types = sales.drop_duplicates(["store", "type"])
print(store_types.head())

# Drop duplicate store/department combinations
store_depts = sales.drop_duplicates(["store", "department"])
print(store_depts.head())

# Subset the rows where is_holiday is True and drop duplicate dates
holiday_dates = sales[sales["is_holiday"] == True].drop_duplicates("date")

# Print date col of holiday_dates
print(holiday_dates["date"])
```

```
# Count the number of stores of each type
store_counts = store_types["type"].value_counts()
print(store_counts)

# Get the proportion of stores of each type
store_props = store_counts / len(store_types["type"])
print(store_props)

# Count the number of each department number and sort
dept_counts_sorted = store_depts['department'].value_counts(sort = True,
ascending = False)
print(dept_counts_sorted)

# Get the proportion of departments of each number and sort
dept_props_sorted = store_depts["department"].value_counts(sort = True,
normalize = True, ascending = False)
print(dept_props_sorted)
```

Summaries by group

```
dogs[dogs[“color”] == “Black”][“weight_kg”].mean()
dogs[dogs[“color”] == “Brown”][“weight_kg”].mean()
dogs[dogs[“color”] == “White”][“weight_kg”].mean()
dogs[dogs[“color”] == “Gray”][“weight_kg”].mean()
```

dogs[dogs[“color”] == “Tan”][“weight_kg”].mean()

- We can get the mean of dogs using the above code snippet. As there is a lot of redundant code, we will make use of Group by method to perform the same calculation

dogs.groupby(“color”)

- Group by color variable

dogs.groupby(“color”)[“weight_kg”]

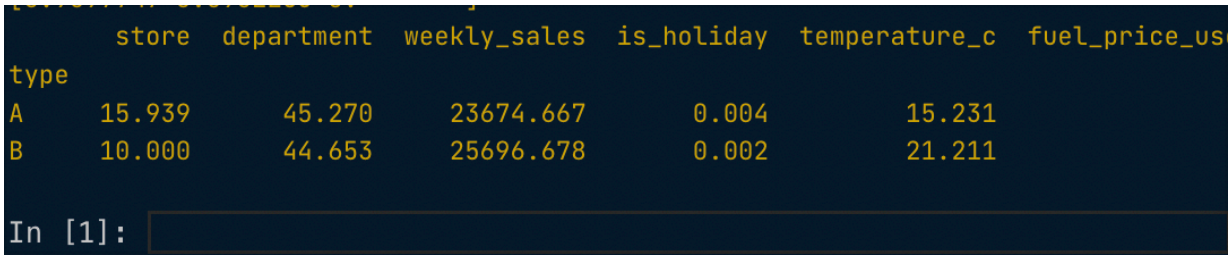
- Select the weight column

dogs.groupby(“color”)[“weight_kg”].mean()

- Then perform mean operation on the group color weights

Groupie returns a groupby object, it is only useful once we perform an aggregation operation on the grouped data

print(sales.groupby("type").mean())

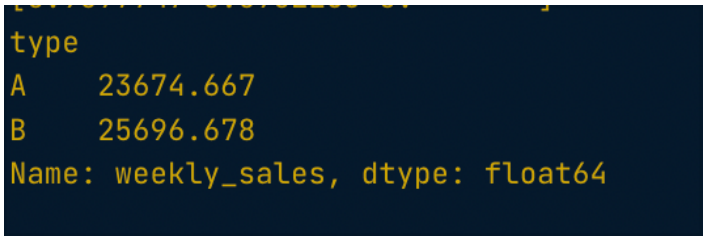


```
       store  department  weekly_sales  is_holiday  temperature_c  fuel_price_us
type
A     15.939      45.270     23674.667       0.004         15.231
B     10.000      44.653     25696.678       0.002         21.211

In [1]:
```

print(sales.groupby("type")["weekly_sales"].mean())

- When a particular column was selected after grouping was done



```
type
A    23674.667
B    25696.678
Name: weekly_sales, dtype: float64
```

Grouped summaries

```
dogs.groupby("color")["weight_kg"].mean()
```

```
color
Black    26.5
Brown    24.0
Gray     17.0
Tan       2.0
White    74.0
Name: weight_kg, dtype: float64
```

Multiple Grouped Summaries

- **To get multiple statistics of a group and to avoid code redundancy, we make use of agg method**
- **dogs.groupby(“color”)[“weight_kg”].agg([min, max, sum])**

Multiple grouped summaries

```
dogs.groupby("color")["weight_kg"].agg([min, max, sum])
```

```
       min  max  sum
color
Black   24   29   53
Brown   24   24   48
Gray    17   17   17
Tan      2    2    2
White   74   74   74
```

We can also group by multiple columns and perform summary statistics

dogs.groupby([“color”, “breed”])[“weight_kg”].mean()

Grouping by multiple variables

```
dogs.groupby(["color", "breed"])["weight_kg"].mean()
```

```
color  breed
Black  Chow Chow      25
       Labrador       29
       Poodle         24
Brown  Chow Chow      24
       Labrador       24
Gray   Schnauzer      17
Tan    Chihuahua       2
White  St. Bernard    74
Name: weight_kg, dtype: int64
```

We can also group by multiple columns and aggregate multiple columns

Many groups, many summaries

```
dogs.groupby(["color", "breed"])[["weight_kg", "height_cm"]].mean()
```

```
                  weight_kg  height_cm
color breed
Black Labrador            29         59
      Poodle              24         43
Brown Chow Chow           24         46
      Labrador            24         56
Gray  Schnauzer           17         49
Tan   Chihuahua            2         18
White St. Bernard         74         77
```

Pivot Tables

– **Pivot Tables are another way of calculating summary statistics**

Group by to pivot table

```
dogs.groupby("color")["weight_kg"].mean()
```

```
color
Black    26
Brown    24
Gray     17
Tan       2
White    74
Name: weight_kg, dtype: int64
```

```
dogs.pivot_table(values="weight_kg",
                 index="color")
```

```
       weight_kg
color
Black       26.5
Brown       24.0
Gray        17.0
Tan          2.0
White       74.0
```

By default pivot table calculates the mean, the groupby column is passed as the index and the value on which the mean needs to be calculated is passed in the values parameter.

To get different summary statistics, we can make use of the aggfunc parameter and pass in it the function name.
dogs.pivot_table(values = “weight_kg”, index = “color”, aggfunc = np.median)

Different statistics

```
import numpy as np
dogs.pivot_table(values="weight_kg", index="color", aggfunc=np.median)
```

```
       weight_kg
color
Black       26.5
Brown       24.0
Gray        17.0
Tan          2.0
White       74.0
```

Multiple statistics

```
dogs.pivot_table(values="weight_kg", index="color", aggfunc=[np.mean, np.median])
```

```
           mean    median
      weight_kg weight_kg
color
Black      26.5      26.5
Brown      24.0      24.0
Gray       17.0      17.0
Tan         2.0       2.0
White      74.0      74.0
```

To group on two columns, we will pass in the value of the second column to group in the columns parameter, here NaN values indicate that there are no values for the combination in the table. To fill in those NaN values, we will make use of the fill_value parameter, here we will pass in the value 0 to indicate that we have no value for this combination.

Pivot on two variables

```
dogs.groupby(["color", "breed"])["weight_kg"].mean()
```

```
dogs.pivot_table(values="weight_kg", index="color", columns="breed")
```

```
breed  Chihuahua  Chow Chow  Labrador  Poodle  Schnauzer  St. Bernard
color
Black        NaN        NaN      29.0    24.0        NaN          NaN
Brown        NaN       24.0      24.0     NaN        NaN          NaN
Gray         NaN        NaN       NaN     NaN       17.0          NaN
Tan          2.0        NaN       NaN     NaN        NaN          NaN
White        NaN        NaN       NaN     NaN        NaN         74.0
```

Filling missing values in pivot tables

```
dogs.pivot_table(values="weight_kg", index="color", columns="breed", fill_value=0)
```

breed	Chihuahua	Chow Chow	Labrador	Poodle	Schnauzer	St. Bernard
color						
Black	0	0	29	24	0	0
Brown	0	24	24	0	0	0
Gray	0	0	0	0	17	0
Tan	2	0	0	0	0	0
White	0	0	0	0	0	74

If we set the margins parameter to be True, then the last row and column will contain the means of their respective rows and columns.
The last right column contains the mean of all the dogs in the dataset.

Summing with pivot tables

```
dogs.pivot_table(values="weight_kg", index="color", columns="breed",
                 fill_value=0, margins=True)
```

breed	Chihuahua	Chow Chow	Labrador	Poodle	Schnauzer	St. Bernard	All
color							
Black	0	0	29	24	0	0	26.500000
Brown	0	24	24	0	0	0	24.000000
Gray	0	0	0	0	17	0	17.000000
Tan	2	0	0	0	0	0	2.000000
White	0	0	0	0	0	74	74.000000
All	2	24	26	24	17	74	27.714286