Session 3. Exploratory data analysis I: Descriptive statistics

2026-02-13

Highlights:

This is my mini-reflection. Paragraphs must be indented.

It can contain multiple paragraphs.

Write the concepts that in your opinion are threshold concepts in this exercise. A threshold concept is a key idea that once you grasp it, it changes your understanding of a topic, phenomenon, subject, method, etc. Write between three and five threshold concepts that apply to your learning experience working on this exercise.

> "We never look beyond our assumptions and what's worse, we have given up trying to meet others; we just meet ourselves."
>
> —– Muriel Barbery

Session outline

- What is EDA?
- Data summaries revisited
- Appropriate summary statistics by scale of measurement
- Properties of data: central tendency and spread
- Univariate description
- Bivariate description
- Multivariate description
- `Pandas`

Reminder

NOTE: This is an Quarto Markdown document. This type of document is a plain text file that can recognize chunks of code. When you execute code within the document, the results are displayed beneath. Quarto Markdown files are *computational notebooks* which implement a coding philosophy called *literate programming*. Literate computing emphasizes the use of natural language to communicate with humans and chunks of code to communicate with the computer. By making the main audience other humans, this style of coding flips around the usual way in which code is written (computer is main audience, humans come second). This helps to make learning how to code more intuitive and accessible.

Preliminaries

Load packages. Remember, packages are units of shareable code that augment the functionality of base `Python`. For this session, the following package/s is/are used:

```python
import pandas as pd
import numpy as np
from scipy.stats import chisquare, chi2_contingency
```

```
# Set display options to show all rows and columns
pd.set_option('display.max_rows', None)    # Show all rows
pd.set_option('display.max_columns', None) # Show all columns
pd.set_option('display.width', None)       # Auto-detect width
pd.set_option('display.max_colwidth', None) # Show full column content
```

We also will utilize some data from the `edashop` R package. To convert these R data files to `Python` files, we will use the `reticulate` package:

```
library(edashop)  # A Package for a Workshop on Exploratory Data Analysis
library(reticulate)
```

From `edashop`, we will also load the following data frames for this session:

```
data("auctions_amf")
data("auctions_pf")
data("auctions_phy")
data("auctions_sef")
```

These data frames contain information about real estate transactions in distressed markets in Italy. You can check the documentation in the usual way:

```
?auctions_amf
```

To be able to convert these datasets in Python, we need to convert them into structures that `Python` recognizes. Following chunk transform them into a Pandas DataFrame:

```
auctions_amf = r.auctions_amf
auctions_pf = r.auctions_pf
auctions_phy = r.auctions_phy
auctions_sef = r.auctions_sef
```

What is EDA?

Exploratory Data Analysis is the process of learning from the data by concentrating on its intrinsic characteristics and attributes. John W. Tukey, the statistician most responsible for clarifying the distinction between exploratory and confirmatory data analysis, likened exploratory data analysis to detective work. Exploratory data analysis is useful to discover essential evidence regarding the phenomenon of interest, similar to checking fingerprints and alibis that can be used in a trial - the equivalent of confirmatory data analysis, where hypotheses are tested and the evidence is evaluated.

To effectively deploy EDA, it is important to approach the data with as few assumptions as possible. By allowing the data to speak for themselves, EDA aims to:

1. *Simplify* descriptions to make them easier to handle with available cognitive power; and
2. Look *below* previously described surfaces to make the description more effective.

The main tools of EDA are descriptive statistics and visualization techniques. The focus in this session is on descriptive statistics, with an emphasis on *appropriate* descriptors for different types of data.

Before proceeding, it is worthwhile to briefly think about the things that we are first interested in when we begin working with a data set. What are the most important characteristics of the data that you care about?

-
-
-

Data descriptions revisited

In the previous session we used the method `describe()` from Pandas to obtain quick statistic descriptions of data. These descriptions already provided some key information about the data, including some summary statistics. For example, in our table with information about auctions in distressed markets:

```
auctions_phy.describe()
```

```
       gross_building_area
count           111.000000
mean            199.747207
std             230.352999
min              11.400000
25%              91.370000
50%             117.830000
75%             184.300000
max            1855.000000
```

We can see that the method `description()`, when set the `include` parameter equals to `all`, understands what type of data it is dealing with, and provides summaries that are different for categorical and quantitative variables.

If you set remove the `include` parameter, only the numerical variables will considered for the summary statistics:

```
auctions_phy.describe(include='all')
```

```
         id type_class  gross_building_area   quality state_maintenance
count   125        122           111.000000       110               107
unique  125          7                  NaN         5                 5
top       1  Residence                  NaN  Adequate              Good
freq      1         90                  NaN        31                33
mean    NaN        NaN           199.747207       NaN               NaN
std     NaN        NaN           230.352999       NaN               NaN
min     NaN        NaN            11.400000       NaN               NaN
25%     NaN        NaN            91.370000       NaN               NaN
50%     NaN        NaN           117.830000       NaN               NaN
75%     NaN        NaN           184.300000       NaN               NaN
max     NaN        NaN          1855.000000       NaN               NaN
```

Properties of data

Description statistics are information reduction techniques. Recall that the objective of EDA is to see the data from different perspectives. Two important properties of data that we often wish to summarize are their central tendency and dispersion. These are discussed next.

Central tendency

A measure of central tendency is a summary of a distribution of values that gives a “typical” value, or the one most frequently observed. Conceptually, this is similar to organizing all data values and finding the location of the *center of mass* of the distribution. To illustrate the concept of center of mass consider the following sequence of quantitative values:

```
x = [20, 30, 32, 34, 41, 41, 45, 46, 48, 51, 53, 54, 54, 56, 57, 58, 58, 59, 60, 61, 64, 65, 65, 69, 71
```

The same sequence of values is shown below in the style of a stem-and-leaf table:

stem	leaf
2	0
3	024
4	11568
5	134467889
6	014559
7	1479
8	8
9	4

Where is the distribution "heavier"? Thereabouts will be its center of mass. There are various measures of central tendency, three of which are discussed next.

In the case of nominal variables, the categories do not have a meaningful order, and yet the center of mass is always the same. Consider for instance:

Mode

The mode of a distribution is the most frequent value found in a distribution. Since it only involves counting the instances of each values, it is appropriate for nominal and ordinal variables. We can find the mode by tabulating the values. Let us do so for the variable `type_class` (factor) in data frame `auctions_phy`.

```
auctions_phy[["type_class"]].describe()
```

```
       type_class
count         122
unique          7
top     Residence
freq           90
```

We see that the mode (top) of the distribution is "Residence", the most frequent value of the variable in this distribution. We can see that there are 113 values for the "type_class" variable, seven possible unique values, with 90 of the rows referring to the "Residence" category.

However, how can we summarize *all* possible values in a categorical variable from a Pandas DataFrame? To do this, we can use the `value_counts()` method that first check for all categories within a variable, and then count the amount of instances in each of them:

```
auctions_phy["type_class"].value_counts()
```

```
type_class
Residence                 90
Factory                    9
Build-on Land              7
Agricultural Building      6
Mixed                      5
Retail                     4
Office                     1
Name: count, dtype: int64
```

If you want to check this information as proportions, you can set the `normalize` parameter as `True` within the `value_counts()` method:

```
auctions_phy["type_class"].value_counts(normalize=True)
```

```
type_class
Residence                0.737705
Factory                  0.073770
Build-on Land            0.057377
Agricultural Building    0.049180
Mixed                    0.040984
Retail                   0.032787
Office                   0.008197
Name: proportion, dtype: float64
```

Now, we see that the "Residence" refers to around 73% of the cases, while the other classes comprise the remaining 27%.

Next, let us try variable `quality` (ordered factor). First as count:

```
auctions_phy["quality"].value_counts()
```

```
quality
Adequate     31
Fair         31
Good         24
Poor         18
Excellent     6
Name: count, dtype: int64
```

And as proportions:

```
auctions_phy["quality"].value_counts(normalize=True)
```

```
quality
Adequate     0.281818
Fair         0.281818
Good         0.218182
Poor         0.163636
Excellent    0.054545
Name: proportion, dtype: float64
```

We see that the mode of this distribution is "Adequate" and "Fair". Since ordinal variables have by definition a natural order, the shape of their distribution can be conveniently presented in the style of a stem-and-leaf table, with each "I" representing one instance of the value:

stem	leaf
Poor	IIIII IIIII IIIII III
Adequate	IIIII IIIII IIIII IIIII IIIII IIIII I
Fair	IIIII IIIII IIIII IIIII IIIII IIIII I
Good	IIIII IIIII IIIII IIIII IIII
Excellent	IIIII I

Median

The median is the quantile that splits a quantitative variables in two parts of equal size, the bottom 50% and the top 50% of values.

Check again the stem-and-leaf table of our sample quantitative variable.

stem	leaf
2	0
3	024
4	11568
5	134467889
6	014559
7	1479
8	8
9	4

There are $n = 30$ observations in this vector. Which value splits the distribution in half?

For a `list` data with quantative values, as the `x` dataset, the median can be reported by using the `median()` method from the `numpy` library:

```
np.median(x)
```

```
57.5
```

Mean

The mean is probably the best known measure of central tendency, and it is defined as the sum of the values divided by the number of observations. Since it involves arithmetic operations it is not appropriate for categorical variables. The mean of quantitative variables can be reported by using the `mean()` method from the `numpy` library:

```
np.mean(x)
```

```
56.8
```

Spread

Another important property of a distribution of values is how wide or compact it is. Compare the two steam-and-leaf tables below.

stem	leaf
2	0
3	024
4	11568
5	134467889
6	014559
7	1479
8	8
9	4

stem	leaf
1	48
2	08
3	024
4	1156
5	13789
6	01459
7	149
8	468
9	45
10	7

The first stem-and-leaf table is more “compact”: the tails of the distribution are closer together and the center of mass is “heavier”, compared to the second table, that has a wider spread.

Minimum and maximum

The minimum and maximum values give an idea of how spread the distribution is. In the first of the preceding tables the minimum is 20 and the maximum is 94. In the second table, the minimum is 14 and the maximum is 107. The *range* is the difference between the maximum and the minimum:

```
94 - 20
```

```
[1] 74
```

```
107 - 14
```

```
[1] 93
```

The second distribution is more spread.

Inter-quartile range

The inter-quartile range is similar to the range, but instead of being calculated using the minimum and maximum values of the distribution, it uses the third and first quartiles. Quartiles are a form of quantile that divides a sequence of values in four equal parts, so the second quantile represents the value that separates the lowest 25% of the sample from the remaining 75%, and the third quantile is the value that splits the highest 25% of the sample from the lowest 75%.

If we describe the days_on_market variable, we see that the quartiles are reported (25% is the first quartile, 50% is the median, and 75% is the third quartiles). The inter-quartile range can be calculated using those values.

```
auctions_amf["days_on_market"].describe()
```

```
count     120.000000
mean      831.291667
std       561.336856
min       190.000000
25%       496.750000
50%       684.000000
75%       931.250000
max      4104.000000
Name: days_on_market, dtype: float64
```

The difference between the third and first quartile is:

```
931 - 497
```

```
[1] 434
```

The inter-quartile range involves an arithmetic operation, which is why it is not an appropriate statistic for categorical variables.

Variance and standard deviation

The variance is another widely used measure of the spread of a distribution. It is defined as:

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

In this formula, $\bar{x}$ is the mean of x and n is the number of observations in the sample. Accordingly, $x_i - \bar{x}$ is the deviation of x_i from the mean of x. If we rewrite this as follows:

$$z_i = (x_i - \bar{x})^2$$

It is easy to see that the variance is actually the mean of the square of the deviations from the mean:

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} z_i$$

The standard deviation is simply the square root of the variance and returns the variance to the same units as the original variable. The standard deviation is reported by using the `description()` method for pandas DataFrames:

```
auctions_amf["days_on_market"].describe()
```

```
count     120.000000
mean      831.291667
std       561.336856
min       190.000000
25%       496.750000
50%       684.000000
75%       931.250000
max      4104.000000
Name: days_on_market, dtype: float64
```

The standard deviation can also be calculated with the `std()` method of the `numpy` library:

```
np.std(auctions_amf["days_on_market"])
```

```
558.9930589287809
```

We see that the typical deviation from the mean of days_on_market was about 561.28 days. To check for its variance, we use the method var(), also from numpy:

```
np.var(auctions_amf["days_on_market"])
```

```
312473.2399305556
```

Univariate description

Summary statistics of central tendency and spread refer to a single variable and are appropriately called univariate descriptors. These descriptors are very important, and we neglect exploring them at our own peril. They often tell us important aspects of the data, including how complete a data set is, how much variation is there, whether there are atypical or unusual values.

As an example, let us calculate the mean, standard deviation, and maximum of days_on_market:

```
mean_dom = np.mean(auctions_amf["days_on_market"])
sd_dom = np.std(auctions_amf["days_on_market"])
max_dom = np.nanmax(auctions_amf["days_on_market"])
```

The property that stayed the longest in auction in this data set did so for $4,104$ days. Just how common or unusual is this value? That depends on how close (or far away) from the mean of the distribution this is, as well as on the spread of the distribution. The deviation from the mean is:

```
max_dom - mean_dom
```

```
3272.7083333333335
```

That is, approximately $3,273$. But the typical deviation from the mean in the sample was only about 561 days! Now, calculating:

```
(max_dom - mean_dom)/sd_dom
```

```
5.854649321773236
```

This tells us that the census tract with the highest days on market receives over five times more than the average days on market of the census tracts. This observation is indeed quite unusual. How unusual was the property with the lowest days on market? Let us retrieve the minimum duration:

```
min_dom = np.min(auctions_amf["days_on_market"])
min_dom
```

```
190.0
```

```
(min_dom - mean_dom)/sd_dom
```

```
-1.1472265288867765
```

The property that stayed the least in auction is much closer to the mean, and barely above one standard deviation below the mean.

Univariate description is a powerful way to get to know our data before doing any more sophisticated explorations or analysis.

Bivariate description

Moving on from univariate description, understanding how two variables relate to one another is another key aspect of EDA.

Categorical variables: cross-tabulations

Univariate description of a categorical variable involves tabulating the number of instances of each response. This can be expanded to simultaneously tabulating two categorical variables. Again, we can use the `value_counts()` method, but now passing two (or more) variables to perfom the counts:

```
auctions_phy[["quality", "type_class"]].value_counts()
```

```
quality    type_class
Adequate   Residence               26
Fair       Residence               25
Good       Residence               14
Poor       Residence               14
Excellent  Residence                6
Good       Factory                  4
Fair       Mixed                    2
Good       Mixed                    2
           Build-on Land            2
Poor       Agricultural Building    2
Fair       Factory                  2
Adequate   Retail                   2
           Factory                  2
Poor       Factory                  1
Fair       Retail                   1
Good       Agricultural Building    1
Adequate   Build-on Land            1
Poor       Mixed                    1
Good       Retail                   1
Fair       Office                   1
Name: count, dtype: int64
```

And the values can be displayed as proportions:

```
auctions_phy[["type_class", "quality"]].value_counts(normalize = "True")
```

```
type_class             quality
Residence              Adequate     0.236364
                       Fair         0.227273
                       Good         0.127273
                       Poor         0.127273
                       Excellent    0.054545
Factory                Good         0.036364
Agricultural Building  Poor         0.018182
Mixed                  Fair         0.018182
Retail                 Adequate     0.018182
Mixed                  Good         0.018182
Factory                Fair         0.018182
                       Adequate     0.018182
Build-on Land          Good         0.018182
Agricultural Building  Good         0.009091
Office                 Fair         0.009091
Mixed                  Poor         0.009091
Factory                Poor         0.009091
Build-on Land          Adequate     0.009091
Retail                 Fair         0.009091
                       Good         0.009091
Name: proportion, dtype: float64
```

What do we learn from this table?

Another way of displaying this information is by doing a cross tabulation by applying the method `.crosstab()`:

```
pd.crosstab(auctions_phy["type_class"], auctions_phy["quality"])
```

```
quality                Poor  Adequate  Fair  Good  Excellent
type_class
Agricultural Building     2         0     0     1          0
Build-on Land             0         1     0     2          0
Factory                   1         2     2     4          0
Mixed                     1         0     2     2          0
Office                    0         0     1     0          0
Residence                14        26    25    14          6
Retail                    0         2     1     1          0
```

To improve the readability of the previous output, we can add the total sum of the columns as a row at the bottom of the table:

```
crosstab_agsn_ipvs = pd.crosstab(auctions_phy["type_class"], auctions_phy["quality"])
crosstab_agsn_ipvs.loc["Total"] = crosstab_agsn_ipvs.sum()
crosstab_agsn_ipvs
```

```
quality                Poor  Adequate  Fair  Good  Excellent
type_class
Agricultural Building     2         0     0     1          0
Build-on Land             0         1     0     2          0
Factory                   1         2     2     4          0
Mixed                     1         0     2     2          0
Office                    0         0     1     0          0
Residence                14        26    25    14          6
Retail                    0         2     1     1          0
Total                    18        31    31    24          6
```

Or the total sums of the rows as a column at the right of the table:

```
crosstab_agsn_ipvs["Total"] = crosstab_agsn_ipvs.sum(axis=1)
crosstab_agsn_ipvs
```

```
quality                Poor  Adequate  Fair  Good  Excellent  Total
type_class
Agricultural Building     2         0     0     1          0      3
Build-on Land             0         1     0     2          0      3
Factory                   1         2     2     4          0      9
Mixed                     1         0     2     2          0      5
Office                    0         0     1     0          0      1
Residence                14        26    25    14          6     85
Retail                    0         2     1     1          0      4
Total                    18        31    31    24          6    110
```

There are in total $n = 110$ valid observations when we consider variables `type_class` and `quality` simultaneously. The proportions (by column) are as follows:

What else do we learn from this table?

If the quality of the buildings did not vary by the type of building, we would expect the percentages in the bottom row to be roughly the same, since every type of building would have a uniform chance of being in any state. We can calculate the values in the table *If* this was true.

This is done by multiplying the row total by the column total for each `quality` and `type_class` combination, and dividing by the size of the sample:

$$E_{ij} = \frac{1}{n} \sum_i x_i \sum_j x_j$$

For example, the expected value for "Agricultural Building" and "Poor" is:

```
(18 * 3) / 110
```

```
[1] 0.4909091
```

The observed value in contrast is 2. So we see that agricultural buildings are of poor quality more often than if the quality of buildings was random. It is possible to summarize the differences between the observed and expected values in a cross-tabulation:

$$\chi^2 = \sum_i \sum_j \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

If the observed and expected values are identical in every case, the value of χ^2 would be zero. Contrariwise, χ^2 will tend to grown as the differences between observed and expected counts grow. This would suggest that the observed counts are unlikely to follow a random pattern.

We can use the `chi2_contingency()` function from the `scipy` library to computes χ^2 and produces also a p-value to aid in the decision whether the distribution follows a random pattern or not. A small p-value would indicate a small probability of the distribution being random:

```
chi2_table = pd.crosstab(auctions_phy["quality"], auctions_phy["type_class"])
chi2_table
```

```
type_class  Agricultural Building  Build-on Land  Factory  Mixed  Office  \
quality
Poor                            2              0        1      1       0
Adequate                        0              1        2      0       0
Fair                            0              0        2      2       1
Good                            1              2        4      2       0
Excellent                       0              0        0      0       0

type_class  Residence  Retail
quality
Poor               14       0
Adequate           26       2
Fair               25       1
Good               14       1
Excellent           6       0
```

```
chi2_contingency(chi2_table)
```

```
Chi2ContingencyResult(statistic=22.54445000819922, pvalue=0.5468050784387253, dof=24, expected_freq=arra
        13.90909091,  0.65454545],
       [ 0.84545455,  0.84545455,  2.53636364,  1.40909091,  0.28181818,
        23.95454545,  1.12727273],
       [ 0.84545455,  0.84545455,  2.53636364,  1.40909091,  0.28181818,
        23.95454545,  1.12727273],
       [ 0.65454545,  0.65454545,  1.96363636,  1.09090909,  0.21818182,
        18.54545455,  0.87272727],
       [ 0.16363636,  0.16363636,  0.49090909,  0.27272727,  0.05454545,
         4.63636364,  0.21818182]]))
```

To visualize the test results in a better way, we can create different variables and assign their values by:

```
stat, p, dof, expected = chi2_contingency(chi2_table)
```

```
stat
```

```
22.54445000819922
```

```
p
```

```
0.5468050784387253
```

```
dof
```

```
24
```

```
expected
```

```
array([[ 0.49090909,  0.49090909,  1.47272727,  0.81818182,  0.16363636,
        13.90909091,  0.65454545],
       [ 0.84545455,  0.84545455,  2.53636364,  1.40909091,  0.28181818,
        23.95454545,  1.12727273],
       [ 0.84545455,  0.84545455,  2.53636364,  1.40909091,  0.28181818,
        23.95454545,  1.12727273],
       [ 0.65454545,  0.65454545,  1.96363636,  1.09090909,  0.21818182,
        18.54545455,  0.87272727],
       [ 0.16363636,  0.16363636,  0.49090909,  0.27272727,  0.05454545,
         4.63636364,  0.21818182]])
```

We also get a DataFrame with expected values:

```
expected_df = pd.DataFrame(expected, index = chi2_table.index, columns = chi2_table.columns)
expected_df
```

```
type_class  Agricultural Building  Build-on Land   Factory     Mixed  \
quality
Poor                     0.490909       0.490909  1.472727  0.818182
Adequate                 0.845455       0.845455  2.536364  1.409091
Fair                     0.845455       0.845455  2.536364  1.409091
Good                     0.654545       0.654545  1.963636  1.090909
Excellent                0.163636       0.163636  0.490909  0.272727

type_class    Office  Residence    Retail
quality
Poor        0.163636  13.909091  0.654545
Adequate    0.281818  23.954545  1.127273
Fair        0.281818  23.954545  1.127273
Good        0.218182  18.545455  0.872727
Excellent   0.054545   4.636364  0.218182
```

Which can be compared to the observed counts:

```python
chi2_table
```

```
type_class  Agricultural Building  Build-on Land  Factory  Mixed  Office  \
quality
Poor                            2              0        1      1       0
Adequate                        0              1        2      0       0
Fair                            0              0        2      2       1
Good                            1              2        4      2       0
Excellent                       0              0        0      0       0

type_class  Residence  Retail
quality
Poor               14       0
Adequate           26       2
Fair               25       1
Good               14       1
Excellent           6       0
```

We see that the distribution is dominated by buildings of class "Residence", which tend to be close to the expected values.

Quantitative variables: correlation

The mean and standard deviation are key univariate descriptors of quantitative variables. When we are interested in the relationship between two quantitative variables we use a related concept, the *covariance*. The covariance is the mean of the product of the deviations from the mean of two variables:

$$C(x,y) = \frac{1}{n}\sum_i (x_i - \bar{x})(y_i - \bar{y})$$

In the formula above, $\bar{x}$ and $\bar{y}$ are the means of the two variables. The differences from the mean can be positive, negative, or zero. When both deviations are positive, the product is positive, thus increasing the covariance. Likewise when the two deviations are negative. But when one deviation is positive and the other negative, the product is negative, which subtracts from the covariance. When differences are like-like (positive-positive or negative-negative) the covariance will tend to be a large positive number. The contrary happens when the differences are opposed.

The covariance can be normalized by the standard deviations of the variables to give the correlation coefficient:

$$r(x,y) = \frac{C(x,y)}{\sigma_x \cdot \sigma_y}$$

This quantity has the census tract of being bound between [-1, 1], and a value of zero indicates that the two variables do not *covary*. Correlations can be calculated from DataFrames using the `corr()` method:

```python
auctions_amf[["days_on_market", "number_auctions"]].corr()
```

```
                 days_on_market  number_auctions
days_on_market         1.000000         0.693218
number_auctions        0.693218         1.000000
```

What does the correlation between these two variables indicate?

Categorical variables: multiple cross-tabulations

It is possible to cross-tabulate more than two variables, however in practice this is seldom done for more than three because the results quickly become difficult to read and interpret. The method `value_counts()` can be applied to three or more columns:

```
auctions_phy[["quality", "state_maintenance", "type_class"]].value_counts()
```

```
quality    state_maintenance  type_class
Adequate   Fair               Residence               13
Fair       Good               Residence               12
           Fair               Residence               10
Adequate   Adequate           Residence                9
Poor       Poor               Residence                8
Good       Good               Residence                7
Excellent  Excellent          Residence                6
Poor       Adequate           Residence                5
Adequate   Poor               Residence                4
Good       Excellent          Residence                3
           Good               Factory                  3
           Fair               Residence                3
Fair       Excellent          Residence                2
           Good               Factory                  2
                              Mixed                    2
Adequate   Good               Factory                  2
Fair       Fair               Office                   1
Adequate   Adequate           Build-on Land            1
Poor       Adequate           Mixed                    1
Good       Good               Retail                   1
Poor       Fair               Factory                  1
                              Residence                1
Good       Good               Build-on Land            1
                              Agricultural Building    1
           Fair               Mixed                    1
Fair       Adequate           Residence                1
Good       Fair               Factory                  1
           Adequate           Residence                1
           Poor               Mixed                    1
Fair       Good               Retail                   1
Adequate   Good               Retail                   1
Poor       Poor               Agricultural Building    1
Name: count, dtype: int64
```

The χ^2 summary of association for a cross-tabulation seen above no longer works for tables in higher dimensions (i.e., when the number of variables cross-tabulated is greater than two). For an example of alternative approaches to describe tables with more than two variables, see the application of Cochran-Mantel-Haenszel χ^2 statistic in Mella-Lira and Paez (2021).

Quantitative variables: correlation matrices

Correlations can be explored among multiple variables. The example below shows how to select the quantitative variables from a data frame and obtain a correlation matrix:

```
auctions_amf.select_dtypes(include=['number']).corr()
```

```
                 days_on_market  number_auctions  discount   premium
days_on_market         1.000000         0.693218 -0.281579  0.082910
number_auctions        0.693218         1.000000 -0.545071  0.192099
discount              -0.281579        -0.545071  1.000000  0.437755
premium                0.082910         0.192099  0.437755  1.000000
```

Note that the default `method` is `pearson`. It is possible to select two other correlation methods (setting the paramether `method` as `kendall` or `spearman`) that work on ranked data instead of the quantities. Ranking the values is a more robust way of calculating the association between two variables for reasons that will become clear next session when we discuss visualization approaches for EDA.

Practice

1. Join tables `auctions_amf`, `auctions_pf`, `auctions_phy`, and `auctions_sef`.
2. Imagine that you are interested in the discount in distressed property sales. The discount (check `?auction_amf`) is the percent variation between the initial listed price (market value as apprised) and selling price. Describe and discuss this variable.
3. How does the discount relate to other variables in the data set? Discuss.
4. Propose some hypotheses about discount based on your exploration of the data set. How would you propose to investigate these hypotheses?
5. Imagine now that you are interested in the state of maintenance of properties that are sold in distressed conditions (check `?auction_phy`). Repeat questions 3 and 4 but for this variable.
6. What can you say so far about the relationship between discount and the categorical variables in the data set? Or between state of maintenance and the quantitative variables in the data set? Propose an approach to explore a combination of categorical and quantitative variables.