|  |
| --- |
| **W5 – Exploratory Data Analysis – Part 2** |

Save your W5 notebook with the following naming conventions.

ID\_Name\_SecNo\_W5.ipynb,

for example

**6113333\_JohnWick\_541\_W5.ipynb**

**Patterns over Time**

Before we can begin to look at potential patterns over time, we need to help pandas understand that data in a given column is in fact date or time data. When a CSV file is imported into pandas, date and time data are typically interpreted as strings, as we see here.

**A blue and white rectangle with black text

Description automatically generated**

We can fix that by adding the parse\_dates keyword argument to the CSV import and setting it equal to a list of column names that should be interpreted as DateTime data. Now, when we check the data types of the imported CSV, the indicated column is a DateTime object. This data type opens up many possibilities for analysis, such as looking at patterns over years, months, or even days of the week.

**A screenshot of a computer code

Description automatically generated**

Of course, we may wish to update data types to DateTime data after we import the data. This is possible with pd-dot-to\_datetime, which converts the argument passed to it to DateTime data. Here, we pass the marriage\_date column with values stored as strings to pd-dot-to\_datetime. This returns DateTime data which we save as the new marriage\_date column.

**A screenshot of a computer

Description automatically generated**

**Creating DateTime data**

pd.to\_datetime has lots of other useful functionality. For example, if a DataFrame has month, day, and year data stored in three different columns, as this one does, we can combine these columns into a single DateTime value by passing them to pd.to\_datetime. Note that for this trick to work, columns must be named "month", "day", and "year", but can appear in any order in the DataFrame.

A screenshot of a computer code

Description automatically generated

A screenshot of a computer program

Description automatically generated

Conversely, we might want to extract just the month, day, or year from a column containing a full date. If data is already stored in DateTime format, as marriage\_date is, we can append .dt.month to extract the month attribute, for example. We'll save the month data as a new column in the DataFrame so that we can use it in our analysis.

A screenshot of a computer screen

Description automatically generated

Line plots are a great way to examine relationships between variables. In Seaborn, line plots aggregate y values at each value of x and show the estimated mean and a confidence interval for that estimate. Perhaps we'd like to check whether there is any relationship between the month that a now-divorced couple got married and the length of their marriage. We can set x equal to the marriage\_month column and y equal to marriage\_duration. The results show some variation in mean marriage duration between months. The blue line represents the mean marriage duration for our dataset, while the confidence intervals in the lighter blue shading indicate the area that, with 95% probability, the population mean duration could fall between. The wide confidence intervals suggest that further analysis is needed!

A graph with blue lines

Description automatically generated

# Let’s Practice.

You'll now work with the entire divorce dataset! The data describes Mexican marriages dissolved between 2000 and 2015. It contains marriage and divorce dates, education level, birthday, income for each partner, and marriage duration, as well as the number of children the couple had at the time of divorce.

1. **Import divorce.csv, saving as a DataFrame, divorce; indicate in the import function that the divorce\_date, dob\_man, and dob\_woman, columns should be imported as DateTime values. Check the data types.**

|  |
| --- |
| Question Which of the columns in the divorce DataFrame has not been updated to a DateTime data type but should be?   * divorce\_date * marriage\_date * education\_woman * num\_kids |

1. **Convert the marriage\_date column of the divorce DataFrame to DateTime values. Check the data types.**

**Define a column called marriage\_year, which contains just the year portion of the marriage\_date column.**

**Create a line plot showing the average number of kids a couple had during their marriage, arranged by the year that the couple got married.**

**Correlation**

Correlation describes the direction of the relationship between two variables as well as its strength. Understanding this relationship can help us use variables to predict future outcomes. A quick way to see the pairwise correlation of numeric columns in a DataFrame is to use pandas' .corr() method.

* A negative correlation coefficient indicates that as one variable increases, the other decreases.
* A value closer to zero is indicative of a weak relationship,
* While values closer to one or negative one indicate stronger relationships.

Note that .corr() calculates the Pearson correlation coefficient, measuring the linear relationship between two variables.

**A screenshot of a computer

Description automatically generated**

Let's wrap our divorce.corr() results in a Seaborn heatmap for quick visual interpretation. A heatmap has the benefit of color coding so that strong positive and negative correlationsare easier to spot. Setting the annot argument to True labels the correlation coefficient inside each cell. Here, we can see that marriage year and marriage duration are strongly negatively correlated; in this dataset, marriages in later years are typically shorter.

A white rectangular object with black text

Description automatically generated

A screenshot of a graph

Description automatically generated

A screenshot of a computer

Description automatically generated

For example, the monthly income of the female partner and the male partner at the time of divorce showed a correlation coefficient of 0.32 in our heatmap. Let's check that this correctly indicates a small positive relationship between the two variables by passing them as x and y arguments to Seaborn's scatterplot function. It looks like the relationship exists but is not particularly strong, just as our heatmap suggested.

A screen shot of a graph

Description automatically generated

We can take our scatterplots to the next level with Seaborn's pairplot. When passed a DataFrame, pairplot plots all pairwise relationships between numerical variables in one visualization. On the diagonal from upper left to lower right, we see the distribution of each variable's observations. This is useful for a quick overview of relationships within the dataset. However, having this much information in one visual can be difficult to interpret, especially with big datasets which lead to very small plot labels like the ones we see here.

A screenshot of a graph

Description automatically generated

We can limit the number of plotted relationships by setting the vars argument equal to the variables of interest. This visual reassures us that what our correlation coefficients told us was true: variables representing the income of each partner as well as the marriage duration variable all have fairly weak relationships with each other. We also notice in the lower right plot that the distribution of marriage durations includes many shorter marriages and fewer longer marriages.

A screenshot of a computer screen

Description automatically generated

# Let’s Practice

According to the following relationships between variables in the divorce DataFrame, which is the possible correct answer?

A screenshot of a computer

Description automatically generated

* marriage\_duration is strongly positively correlated with marriage\_month
* The correlation between num\_kids and income\_man is stronger than the correlation between num\_kids and marriage\_duration.
* A later marriage\_year causes a lower number of children, represented by num\_kids.
* A latter marriage\_year is correlated with having fewer children.

In the last exercise, you may have noticed that a longer marriage\_duration is correlated with having more children, represented by the num\_kids column. The correlation coefficient between the marriage\_duration and num\_kids variables is approximately 0.45.

1. **Create a scatterplot showing marriage\_duration on the x-axis and num\_kids on the y-axis.**
2. **Create a pairplot to visualize the relationships between income\_woman and marriage\_duration in the divorce DataFrame.**

**Categorial Relationships**

We haven't explored the categorical variables related to education level yet. Let's do it! Checking the value\_counts for education\_man, we see that most men have an education level between primary and professional, with a few men in the "None" or "Other" categories.

A screenshot of a computer code

Description automatically generated

Categorical variables are harder to summarize numerically, so we often rely on visualizations to explore their relationships. Perhaps we are interested in the relationship between marriage duration and the education level of the man in the dissolved marriage. We could begin by making a histogram of the distribution of marriage duration.

A graph of a person

Description automatically generated

Then layer in the information we have on male education level by setting education\_man as the hue argument.

A graph of different colored bars

Description automatically generated

However, because the education levels are stacked on top of each other, the relationship between marriage duration and male education level isn't super clear.

**Seaborn KDE Plots**

Seaborn's Kernel Density Estimate or KDE plots address this issue. Similar to histograms, KDEs allow us to visualize distributions. KDEs are considered more interpretable, though, especially when multiple distributions are shown as they are here. Notice that the location of the peak marriage duration for each level of the male partner's education is more identifiable in this KDE plot than it was in the histogram. However, due to the smoothing algorithm used in KDE plots, the curve can include values that don't make sense, so it's important to set good smoothing parameters.

A graph of a person with colored lines

Description automatically generated

Here's an example: zooming in on the KDE plot showing the distribution of male education levels, we can see that the distribution seems to suggest that some couples had marriage durations of less than zero. That's impossible!

A graph of a number of plots

Description automatically generated

To fix this, we can use the cut keyword argument. cut tells Seaborn how far past the minimum and maximum data values the curve should go when smoothing is applied. When we set cut equal to zero, the curve will be limited to values between the minimum and maximum x values,

A graph of a marriage

Description automatically generated

If we're interested in the cumulative distribution function, we can set the cumulative keyword argument to True. This graph describes the probability that marriage duration is less than or equal to the value on the x-axis for each level of male partner education.

A screen shot of a graph

Description automatically generated

Perhaps we are interested in whether divorced couples who got married when they were older typically have higher levels of education. We can create columns representing the approximate age at marriage for men and women by subtracting each partner's birth year from the marriage year.

A close-up of a marriage

Description automatically generated

Then, we create a scatterplot using these variables on the x and y-axis. It looks like there is a positive correlation between them!

A screen shot of a graph

Description automatically generated

A screen shot of a graph

Description automatically generated

The results suggest that men with a professional education level, represented with orange dots, may tend to get married later.

**Let’s Practice.**

1. **Create a scatter plot that shows woman\_age\_marriage on the x-axis and income\_woman on the y-axis; each data point should be colored based on the woman's level of education, represented by education\_woman.**

|  |
| --- |
| Note: From the graph, it looks like there is a positive correlation between professional education and higher salaries, as you might expect. The relationship between women's age at marriage and education level is a little less clear. |

1. **Create a KDE plot that shows marriage\_duration on the x-axis and a different colored line for each possible number of children that a couple might have, represented by num\_kids.**

**Notice that the plot currently shows marriage durations less than zero; update the KDE plot so that marriage duration cannot be smoothed past the extreme data points.**

**Update the code for the KDE plot from the previous step to show a cumulative distribution function for each number of children a couple has.**

|  |
| --- |
| Note: It looks as though there is a positive correlation between longer marriages and more children, but of course, this doesn't indicate causation. You can also see that there is much less data on couples with more than two children; this helps us understand how reliable our findings are. |

**Considerations for Categorial Data**

Let's see how we convert exploratory data analysis into action! We'll start by looking at class frequencies.

Recall that EDA is performed for a variety of reasons, like

* detecting patterns and relationships in data,
* generating questions or hypotheses, or
* preparing data for machine learning models.

A close-up of a flag

Description automatically generated

With categorical data, one of the most important considerations is about the representation of classes, which is another term for labels. For example, say we collect data on people's attitudes to marriage. As part of our data collection we find out their marital status, with the classes including single, married, and divorced.

A close-up of a white background

Description automatically generated

When we perform EDA we realize only 50 people were married, while 700 were divorced and the remaining 250 were single. Do we think that this sample accurately represents the general public's opinion about marriage? Are divorced people more likely to have a negative view towards marriage? This is an example of class imbalance, where one class occurs more frequently than others. This can bias results, particularly if this class does not occur more frequently in the population.

A graph with different colored squares

Description automatically generated

**Class Frequency**

Say that we know 40 percent of internal Indian flights go to Delhi. We can use value\_counts method again, but this time set the normalize keyword argument equal to True. This returns the relative frequencies for each class, showing that Delhi only represents 11.82 percent of destinations in our dataset. Again, this could suggest that our data is not representative of the population - in this case, internal flights in India.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Cross-tabulation**

Another method for looking at class frequency is cross-tabulation, which enables us to examine the frequency of combinations of classes. Let's look at flight route frequencies. We'll start by calling pandas-dot-crosstab function.

A diagram of a function

Description automatically generated

A screenshot of a table

Description automatically generated

We see the most popular route is from Delhi to Cochin, making up 4318 flights.

Say we know the median price for all internal flight routes in India. Here they are for the routes in our dataset, measured in Indian Rupees. We can calculate the median price for these routes in our DataFrame, and compare the difference to these expected values.

A table with numbers and a few black text

Description automatically generated

We do this by adding two keyword arguments to pd.crosstab(). We pass the Price column to the values argument, and use aggfunc to select what aggregated calculation we want to perform. We can pass a summary statistic as a string, in this case setting it equal to median. The results show median values for all possible routes in the dataset.

A screenshot of a computer program

Description automatically generated

A screenshot of a graph

Description automatically generated

Comparing our prices with the expected values, most are similar. However, routes from Banglore to Delhi and New Delhi are significantly different (more expensive and less expensive) in the dataset, suggesting they aren't representative of the population.

**Let’s Practice.**

# Checking for class imbalance

The [2022 Kaggle Survey](https://www.kaggle.com/kaggle-survey-2022) captures information about data scientists' backgrounds, preferred technologies, and techniques. It is seen as an accurate view of what is happening in data science based on the volume and profile of responders.

Having looked at the job titles and categorized to align with our salaries DataFrame, you can see the following proportion of job categories in the Kaggle survey:

A screenshot of a graph

Description automatically generated

Load Salary\_Rupee\_USD.csv to salaries with index\_col = 0

1. **Print the relative frequency of the "Job\_Category" column from salaries DataFrame.**

|  |
| --- |
| Note: It looks like Data Science is the most popular class and has a similar representation. Still, the other categories have quite different relative frequencies, which might not be surprising given the target audience is data scientists! Given the difference in relative frequencies, can you trust the salaries DataFrame to accurately represent Managerial roles? |

# Cross-tabulation

Cross-tabulation can help identify how observations occur in combination.

Using the salaries dataset, you'll perform cross-tabulation on multiple variables, including the use of aggregation, to see the relationship between "Company\_Size" and other variables.

1. **Perform cross-tabulation, setting "Company\_Size" as the index, and the columns to classes in "Experience".**

**Perform cross-tabulate "Job\_Category" and classes of "Company\_Size" as column names.**

**Update pd.crosstab() to return the mean "Salary\_USD" values.**

Note: Looks like the largest mean salary is for Managerial data roles in large companies!

**Generating New Features.**

Sometimes the format of our data can limit our ability to detect relationships or inhibit the potential performance of machine learning models. One method to overcome these issues is to generate new features from our data!

Checking correlation with a heatmap, we see a moderate positive correlation between Price and Duration, but it looks like those are the only numeric variables in our dataset.

A screenshot of a computer

Description automatically generated

|  |
| --- |
| planes = pd.read\_csv('Airlines\_unclean.csv', index\_col = 0,  parse\_dates=['Date\_of\_Journey','Dep\_Time','Arrival\_Time'])  # Remove the string character  planes["Duration"] = planes["Duration"].str.replace("h", ".")  planes["Duration"] = planes["Duration"].str.replace("m", "")  planes["Duration"] = planes["Duration"].str.replace(" ", "")  # Convert to float data type  planes["Duration"] = planes["Duration"].astype(float)  print(planes.info())  ax = sns.heatmap(planes.corr(), annot=True)  ax.set\_ylim([0,2])  plt.show() |

Viewing the data types confirms this is the case. However, Total\_Stops should also be numeric. Viewing the value\_counts, we see need to remove string characters.

A screenshot of a computer screen

Description automatically generated

We use the string-dot-replace method to first remove " stops", including the space, so that flights with two, three, or four stops are ready to convert. Next, we clean flights with one stop. Lastly, we change "non-stop" to "0", then set the data type to integer.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

Unsurprisingly, Total\_Stops is strongly correlated with Duration. What is interesting is that Total\_Stops and Price are more strongly correlated than Duration is with Price! Let's see what else we can find out!

Rechecking our data types, notice that there are three datetime variables - Date\_of\_Journey, Dep\_Time, and Arrival\_Time.

A screenshot of a computer program

Description automatically generated

**Extracting Months and Weekdays**

We know how to extract attributes from datetime values, so we can see if these offer any insights into pricing. To start, let's look at Date\_of\_Journey. If we think prices vary per month, it's worth using this attribute - we create it as a column in our DataFrame. Perhaps prices might also differ depending on the day of the week? Let's grab that using the dt.weekday attribute. It returns values of zero, representing Monday, through to seven, for Sunday. Previewing these columns we see the first flight, departing on the 6th September, was a Friday, indicated by a four.

A screenshot of a computer program

Description automatically generated

We might wonder if people tend to pay more to depart or arrive at more convenient times. We extract the hour of departure and arrival from those respective columns too.

A close-up of a computer code

Description automatically generated

Because they are numeric, we can calculate correlation between these new datetime features and other variables. Re-plotting our heatmap, unfortunately there aren't any new strong relationships. **But we wouldn't have known this if we hadn't generated these features.**

A screenshot of a graph

Description automatically generated

**Another technique to generate new features.**

There's one more technique we can use to generate new features. We can group numeric data and label them as classes. For example, we don't have a column for ticket type. We could use descriptive statistics to label flights as economy, premium economy, business class, or first class, based on prices within specific ranges, or bins.

A screenshot of a computer code

Description automatically generated

We'll split equally across the price range using quartiles. We first store the 25th percentile using the quantile method. We get the 50th percentile by calling the median. Next we get the 75th percentile, and lastly, we store the maximum value.

A white background with black text

Description automatically generated

Then create labels. Next, we create the bins, a list starting from zero and including our descriptive statistic variables.

A close-up of a computer code

Description automatically generated

A diagram of a process

Description automatically generated

Then preview the price categories.

A screenshot of a computer

Description automatically generated

We can plot the count of flights in different categories per airline by passing our new column to the hue argument when calling sns.countplot.

A white background with black text

Description automatically generated

A graph of different colored bars

Description automatically generated

Looks like Jet Airways has the largest number of "First Class" tickets, while most of IndiGo and SpiceJet's flights are "Economy".

**Let’s practice with salaries DataFrame.** Load the Salaries\_with\_date\_of\_response.csv to salaries, with index\_col = 0 and parse\_date = [‘date\_of\_response’]

Your task is to extract datetime attributes from this column and then create a heat map to visualize the correlation coefficients between variables.

1. **Extract the month from "date\_of\_response", storing it as a column called "month".**

**Create the "weekday" column, containing the weekday that the participants completed the survey.**

**Plot a heat map, including the Pearson correlation coefficient scores.**

Your next task is to convert the "Salary\_USD" column into categories based on its percentiles. First, you need to find the percentiles and store them as variables.

1. **Find the 25th percentile of "Salary\_USD".**

**Store the median of "Salary\_USD" as salaries\_median.**

**Get the 75th percentile of salaries.**

**Create salary\_labels, a list containing "entry", "mid", "senior", and "exec".**

**Finish salary\_ranges, adding the 25th percentile, median, 75th percentile, and largest value from "Salary\_USD"**

**Split "Salary\_USD" based on the labels and ranges you've created.**

**Use sns.countplot() to visualize the count of "Company\_Size", factoring salary level labels.**

**A graph of different colored bars

Description automatically generated**

1. **Regarding comparing salaries, exploratory data analysis is a crucial step in generating hypotheses! You've had an idea you'd like to explore—do data professionals get paid more in the USA than they do in Great Britain? You'll need to subset the data by "Employee\_Location" and produce a plot displaying the average salary between the two groups.**

**Filter salaries where "Employee\_Location" is "US" or "GB", saving as usa\_and\_gb.**

**Use usa\_and\_gb to create a barplot visualizing  "Salary\_USD" against "Employee\_Location".**

**Choosing a hypothesis**

You've seen how visualizations can be used to generate hypotheses, making them a crucial part of exploratory data analysis!

In this exercise, you'll generate a bar plot to inspect how salaries differ based on company size and employment status. For reference, there are four values:

| **Value** | **Meaning** |
| --- | --- |
| CT | Contractor |
| FL | Freelance |
| PT | Part-time |
| FT | Full-time |

## From the plot on 11, What is a reasonable hypothesis to generate based on this plot?

1. On average, small companies pay part-time employees less than large companies.
2. Freelancers earn more at medium-sized companies compared to small or large companies.
3. On average , large companies pay contractors more than medium-sized companies.
4. No hypotheses can be generated from this plot.