Off-Platform Project: Stack Overflow Survey Trends

Understanding developer trends in Stack Overflow survey data

You’ve learned a lot about missing data and how to handle it, so now it’s time to put these strategies into action!

Project Overview

You work for a staffing agency that specializes in finding qualified candidates for development roles. One of your latest clients is growing rapidly and wants to understand what kinds of developers they can hire, and to understand general trends of the technology market. Your organization has access to this [Stack Overflow dataset](), which consists of survey responses by developers all over the world for the last few years.

Your project is to put together several statistical analyses about the community to educate your client about the potential hiring market for their company.

Project Steps

Explore data

You decide to start by performing some Exploratory Data Analysis (EDA). This will provide you with a high-level understanding of the data fields, as well as help you identify which columns have missing data. In this case, you load the dataset into a `pandas` DataFrame and call it `df`. Take a moment to explore which columns you have in the data.

```
import pandas as pd

df = pd.read_csv('developer_dataset.csv')

print(df.columns)

Index(['RespondentID', 'Year', 'Country', 'Employment'
'UndergradMajor', 'DevType', 'LanguageWorkedWith',
'LanguageDesireNextYear','DatabaseWorkedWith', 'DatabaseDesireNextYear', 'PlatformWorkedWith'
'PlatformDesireNextYear', 'MainBranch', 'Hobbyist',
'EdLevel', 'OrgSize', 'YearsCodePro', 'JobSeek',
'JobFactors', 'ConvertedComp','WorkWeekHrs', 'OpSys',
```

```
'WelcomeChange', 'NEWJobHunt','NEWJobHuntResearch', 'NEWLearn'],
    dtype='object')
```

At an initial glance, you notice the following kinds of information: A variety of columns that identify the person (`RespondentID`, `Year`, `Country`) Information about their experiences (`LanguageWorkedWith`, `DatabaseWorkedWith`, `UndergradMajor`, etc.) Information about what they might want to do in the future (`LanguageDesireNextYear`, `DatabaseDesireNextYear`, etc.)

At this point, you want a good understanding of how much data you have. Since this is a survey where each question is optional, you don't expect every column to have a full set of data.

Run `df.count()` to see a row count for each column. It should look something like this:

```
RespondentID           111209
Year                   111209
Country                111209
Employment             109425
UndergradMajor          98453
DevType                100433
LanguageWorkedWith     102018
LanguageDesireNextYear  96044
DatabaseWorkedWith      85859
DatabaseDesireNextYear  74234
PlatformWorkedWith      91609
PlatformDesireNextYear  85376
Hobbyist                68352
OrgSize                 54804
YearsCodePro            94793
JobSeek                 60556
ConvertedComp           91333
WorkWeekHrs             51089
NEWJobHunt              19127
NEWJobHuntResearch      18683
NEWLearn                24226
```

From here, you can perform some basic summary statistics on the dataset. This will allow you to understand things like:

- average values
- max and min values
- the number of missing data points

This will only work for numerical columns, but that will still be helpful. Use the following code:

```
df.describe()
```

You should see the following output:

```
    RespondentID  Year     YearsCodePro  ConvertedComp  WorkWeekHrs
count 111209.00000  111209.00000  94793.00000  9.133300e+04  51089.00000
mean  19262.039709  2018.854832  9.547045   1.251777e+05  41.051670
std   11767.011322  0.777503    7.548931   2.461218e+05  13.833929
min   1.000000     2018.000000  0.000000   0.000000e+00  1.000000
25%   9268.000000   2018.000000  4.000000   4.600000e+04  40.000000
50%   18535.000000  2019.000000  8.000000   7.900000e+04  40.000000
75%   28347.000000  2019.000000  14.000000  1.200000e+05  42.000000
max   42857.000000  2020.000000  50.000000  2.000000e+06  475.000000
```

Based on the above information, what observations can you make about the dataset?

- Are there columns that have more missing data than others?
- Which columns seem interesting? What insights would you want to gain from the data?
- Are there columns that have potentially more sensitive data than others? How would that change our strategies in dealing with them?

Delete highly missing data

You notice this dataset has a number of columns with a significant amount of missing data. With this much missing data, it is unlikely that any statistical analysis using that data would be accurate and representative of the developers who filled out the survey. Luckily, you recall that you can safely remove columns with ~60% or more missing data.

Run the below code to see the percentage missing data for each column.

```
maxRows = df['RespondentID'].count()

print('% Missing Data:')
print((1 - df.count() / maxRows) * 100)
```

Your output should look like this:

```
% Missing Data:
RespondentID           0.000000
Year                   0.000000
Country                0.000000
Employment             1.604187
UndergradMajor        11.470295
DevType                9.689863
LanguageWorkedWith     8.264619
```

```
LanguageDesireNextYear    13.636486
DatabaseWorkedWith        22.794918
DatabaseDesireNextYear    33.248208
PlatformWorkedWith        17.624473
PlatformDesireNextYear    23.229235
Hobbyist                  38.537349
OrgSize                   50.719816
YearsCodePro              14.761395
JobSeek                   45.547573
ConvertedComp             17.872654
WorkWeekHrs               54.060373
NEWJobHunt                82.800852
NEWJobHuntResearch        83.200101
NEWLearn                  78.215792
```

Based on the above numbers, you assume that it is safe to remove the following columns:

- `NEWJobHunt`
- `NEWJobHuntResearch`
- `NEWLearn`

Use pandas to drop those DataFrame columns.

```
df.drop(['NEWJobHunt','NEWJobHuntResearch','NEWLearn'],
  axis=1,
  inplace=True)
```

Analyze developers by country

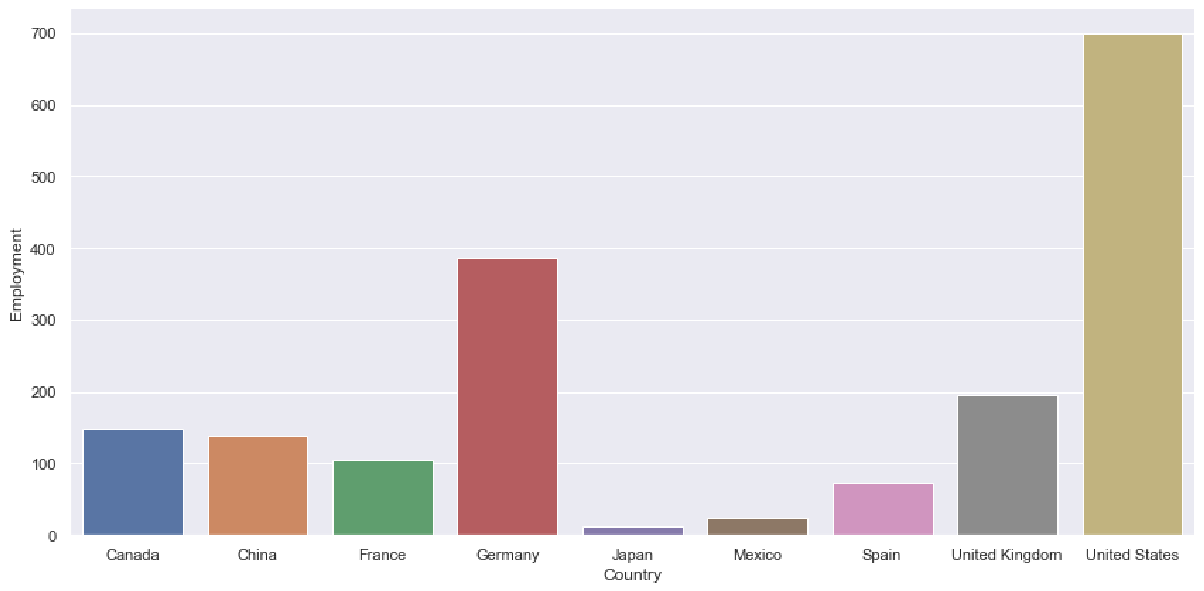
Start thinking about the questions you want to ask of the data. You decide to investigate the distribution of employment and developer type from a geographical (i.e. `Country`) perspective.

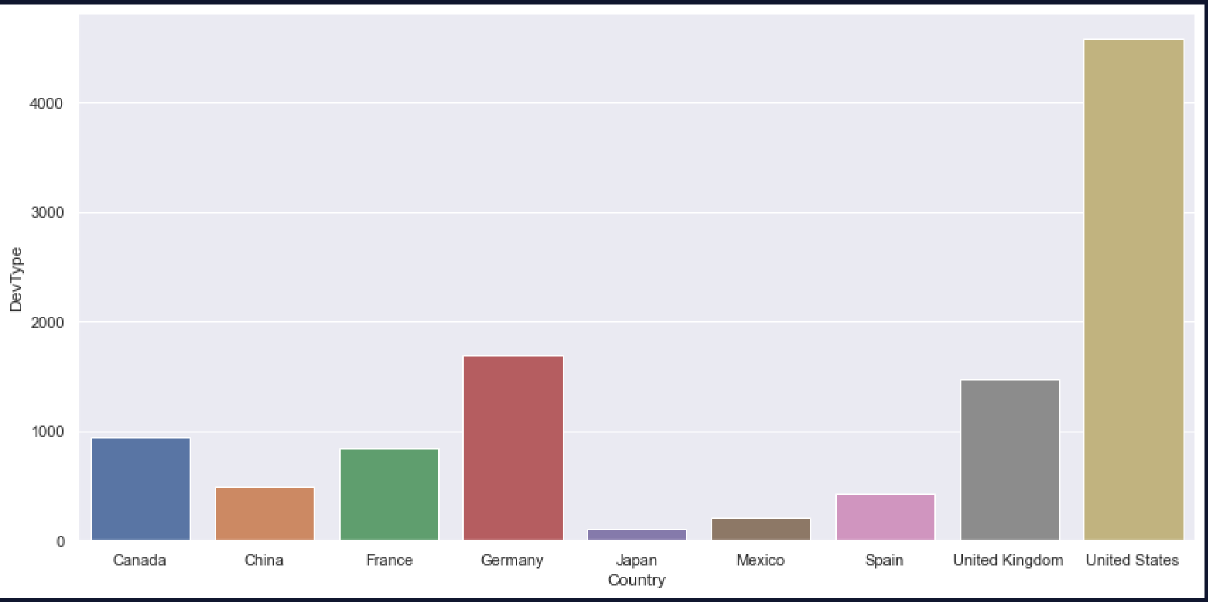
Both the `Employment` and `DevType` fields have missing data, but not a very significant amount, both with less than 10% missing. This is going to be foundational for your analyses moving forward, so you want to ensure that there are no missing data points.

Determine what kind of missing data you have for employment and developer type. One way to do that is check, at a country level, where the data is missing for each field:

```
import seaborn as sns
import matplotlib.pyplot as plt

df[['RespondentID','Country']].groupby('Country').count()
```

```
missingData = df[['Employment','DevType']].isnull().groupby(df['Country']).sum().reset_index()

A=sns.catplot(
    data=missingData, kind="bar",
    x="Country", y="Employment",
    height = 6, aspect = 2)
B=sns.catplot(
    data=missingData, kind="bar",
    x="Country", y="DevType",
    height = 6, aspect = 2)

Country      RespondentID
Canada          8979
China           2072
France          6861
Germany         16215
Japan           1049
Mexico          1918
Spain           4534
United Kingdom   15854
United States    53727
```





Note: *You can click on images to enlarge them.*

As we can see from the above plots, the data doesn't appear to be missing for any country significantly more than any other. Using your domain knowledge, you understand that the missing data appears to scale with the relative size of each country (e.g. there is more missing data in the United States vs. Japan because there will be more respondents there). You also note that the United States and Germany have significantly more developers (on average) than the other countries, explaining why they have more missing data points.

You determine that the missing data for these two columns can be categorized as `MCAR`. This means you can safely delete the rows that have missing data in these columns! This is a prime example of where you can employ Pairwise Deletion to only delete rows that have missing data for either `Employment` or `DevType`:

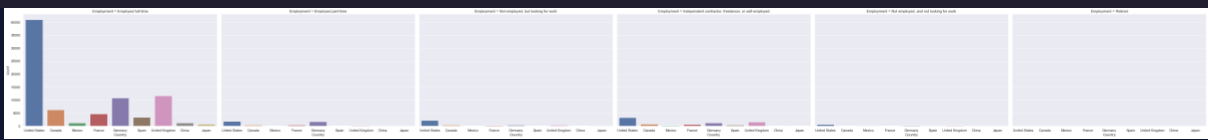
```
df.dropna(subset = ['Employment','DevType'],
   inplace = True,
   how = 'any')
```

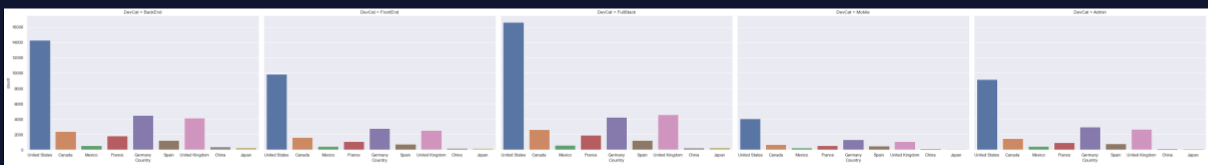
Now you can analyze the distribution of employment and developer types by country. You decide to aggregate the employment data by key developer roles that align with major parts of the development lifecycle:

- Front-end
- Back-end

- Full-stack
- Mobile development
- Administration roles

```
empfig = sns.catplot(x="Country", col="Employment",
            data=df, kind="count",
            height=6, aspect=1.5);

# Focus on a few of the key developer types outlined in the Stack Overflow survey
devdf = df[['Country','DevType']]
devdf.loc[devdf['DevType'].str.contains('back-end'), 'BackEnd'] = True
devdf.loc[devdf['DevType'].str.contains('front-end'), 'FrontEnd'] = True
devdf.loc[devdf['DevType'].str.contains('full-stack'), 'FullStack'] = True
devdf.loc[devdf['DevType'].str.contains('mobile'), 'Mobile'] = True
devdf.loc[devdf['DevType'].str.contains('administrator'), 'Admin'] = True

devdf = devdf.melt(id_vars=['Country'],
   value_vars=['BackEnd','FrontEnd','FullStack','Mobile','Admin'],
   var_name='DevCat',
   value_name='DevFlag')

devdf.dropna(how='any', inplace=True)

devFig = sns.catplot(x="Country", col="DevCat",
            data=devdf, kind="count",
            height=6, aspect=1.5);
```





Note: *You can click on images to enlarge them.*

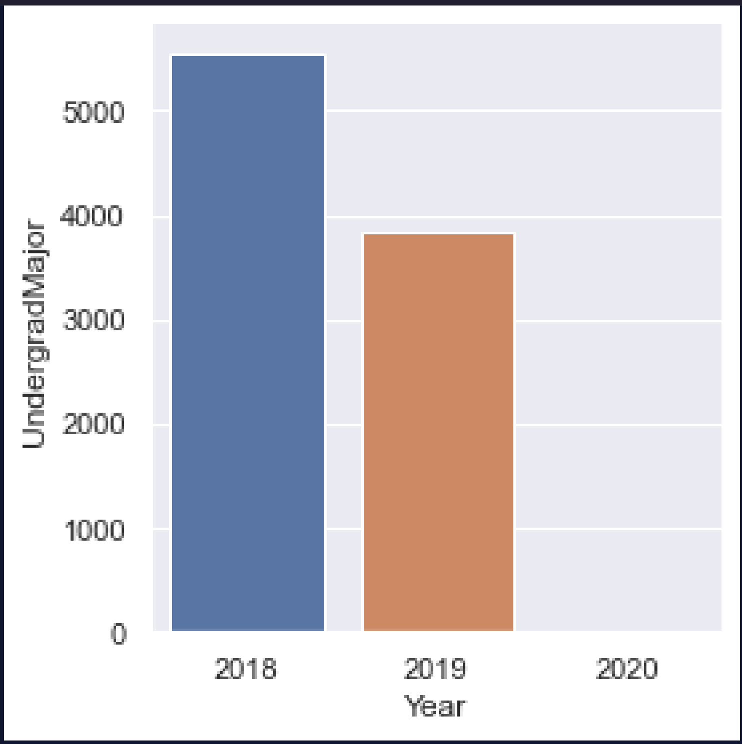
You see that the vast majority of respondents are employed full-time. Since these developers are mainly employed, this data will be relevant for a client who wants to see what developers look for in a potential job. You also see that the majority of developers will have skill sets in front-end, back-end, or full-stack development. This is interesting, and shows that the market values developers who can excel in at least a major part of the development lifecycle, if not the entire stack.

Investigate developer undergraduate majors

You decide to dive into the background for each type of developer to see trends in their educational experience that ultimately led to a career with technology. In particular, you look at the overall trend of majors year over year for respondents. As you saw before, you are missing about 11% of the data for `UndergradMajor`. Why do you think this data is missing? Could something have happened over the course of these three years? Is the fact that data is missing accurate?

To test your theory, take a look at the distribution of majors over each year:

```
missingUndergrad = df['UndergradMajor'].isnull().groupby(df['Year']).sum().reset_index()

sns.catplot(x="Year", y="UndergradMajor",
            data=missingUndergrad, kind="bar",
            height=4, aspect=1);
```



Note: *You can click on images to enlarge them.*

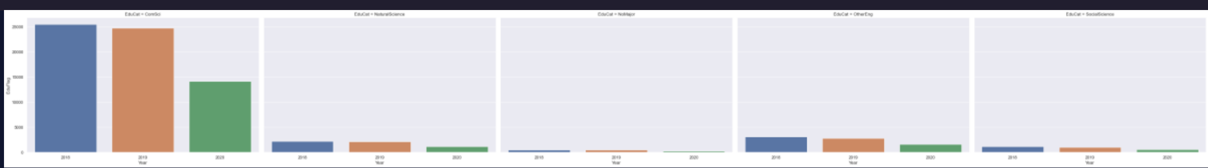
You see that all of the data for 2020 undergrad majors is filled in, indicating that each participant in these surveys had some level of decision for their undergrad major. For the purposes of your analysis, you are most interested in what major a person ultimately landed on, as this would be the educational background they would carry into a job search. You want to carry that value backwards for each participant to fill in any missing data.

This is a great use for one of our Single Imputation techniques: NOCB! Fill in the gaps using NOCB:

```
# Sort by ID and Year so that each person's data is carried backwards correctly
df = df.sort_values(['RespondentID','Year'])

df['UndergradMajor'].bfill(axis=0, inplace=True)
```

From here, you analyze the major distribution for each year, using a vertical bar chart visualization:

```
# Key major groups outlined in the Stack Overflow survey
majors = ['social science','natural science','computer science','development','another engineering','never declared']

edudf = df[['Year','UndergradMajor']]
edudf.dropna(how='any', inplace=True)
edudf.loc[edudf['UndergradMajor'].str.contains('(?i)social science'), 'SocialScience'] = True
edudf.loc[edudf['UndergradMajor'].str.contains('(?i)natural science'), 'NaturalScience'] = True
edudf.loc[edudf['UndergradMajor'].str.contains('(?i)computer science'), 'ComSci'] = True
edudf.loc[edudf['UndergradMajor'].str.contains('(?i)development'), 'ComSci'] = True
edudf.loc[edudf['UndergradMajor'].str.contains('(?i)another engineering'), 'OtherEng'] = True
edudf.loc[edudf['UndergradMajor'].str.contains('(?i)never declared'), 'NoMajor'] = True

edudf = edudf.melt(id_vars=['Year'],
  value_vars=['SocialScience','NaturalScience','ComSci','OtherEng','NoMajor'],
  var_name='EduCat',
  value_name='EduFlag')

edudf.dropna(how='any', inplace=True)
edudf = edudf.groupby(['Year','EduCat']).count().reset_index()

eduFig = sns.catplot(x="Year", y='EduFlag', col="EduCat",
          data=edudf, kind="bar",
          height=6, aspect=1.5);
```



Note: *You can click on images to enlarge them.*

You notice that the vast majority of people who enter the workforce for development have some background in a Computer Science major. Interestingly, however, the number of Computer Science majors significantly declined over the years surveyed, indicating that there could be other majors that have successfully entered the workforce for their desired job. This would require further analysis and could allow an individual to pursue a separate education path and still end up in some kind of developer role.

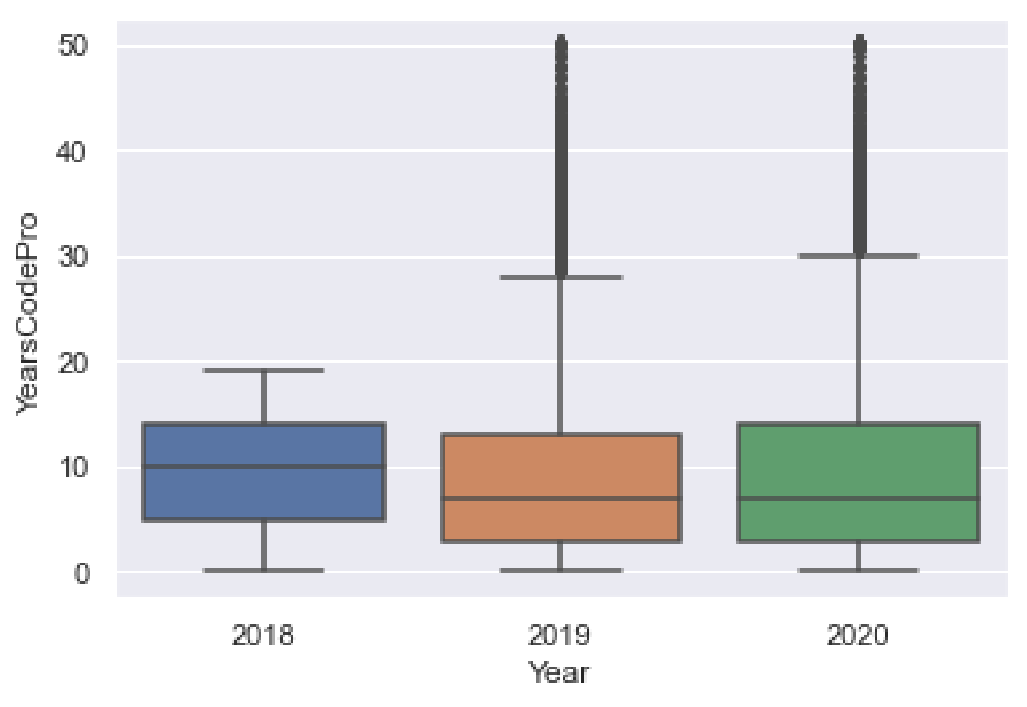
Examine the relationship between years of experience and compensation

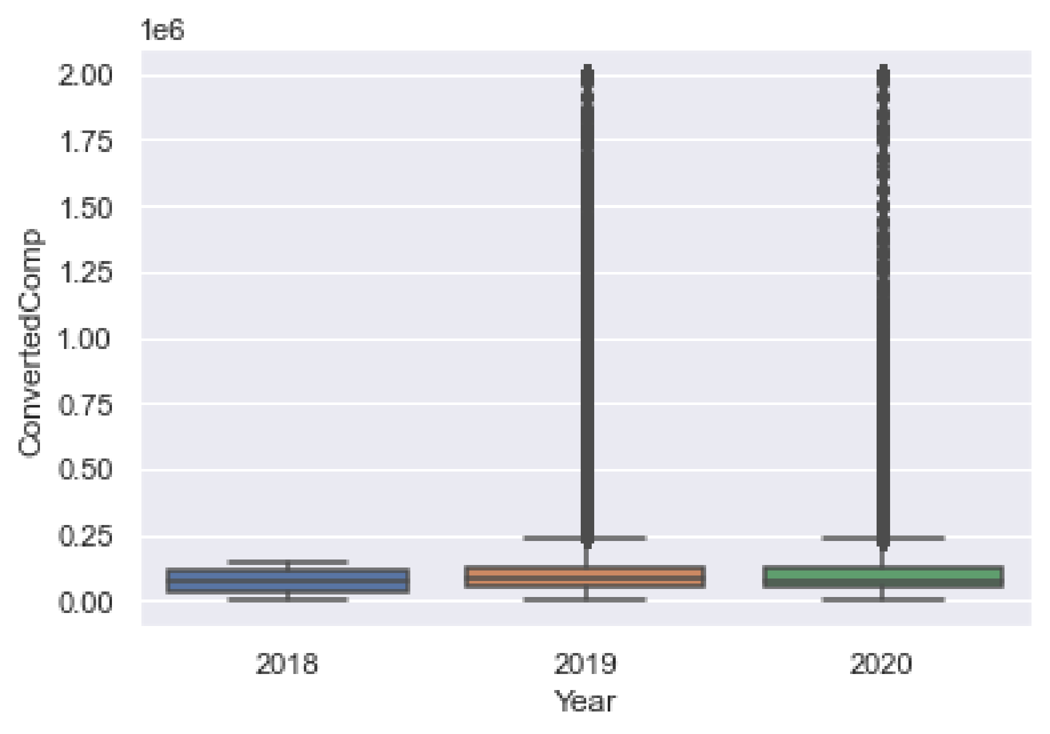
At this point, you have studied the demographics of developers around the world, from where they live to the education paths they have taken. Now, you turn your focus to the various aspects that would influence the job-hunting process.

Years of experience are an important metric when looking to understand the general skill and technical capabilities of a potential candidate. Compensation is also important for our client to understand what the "going rate" for a particular developer is in today's market. You might assume that there is a strong correlation between experience and job compensation, making it an excellent hypothesis to explore.

In order to understand a bit about the data for each of these two fields, perform some more exploratory analysis:

```
compFields = df[['Year','YearsCodePro','ConvertedComp']]

D = sns.boxplot(x="Year", y="YearsCodePro",
        data=compFields)

E = sns.boxplot(x="Year", y="ConvertedComp",
        data=compFields)
```





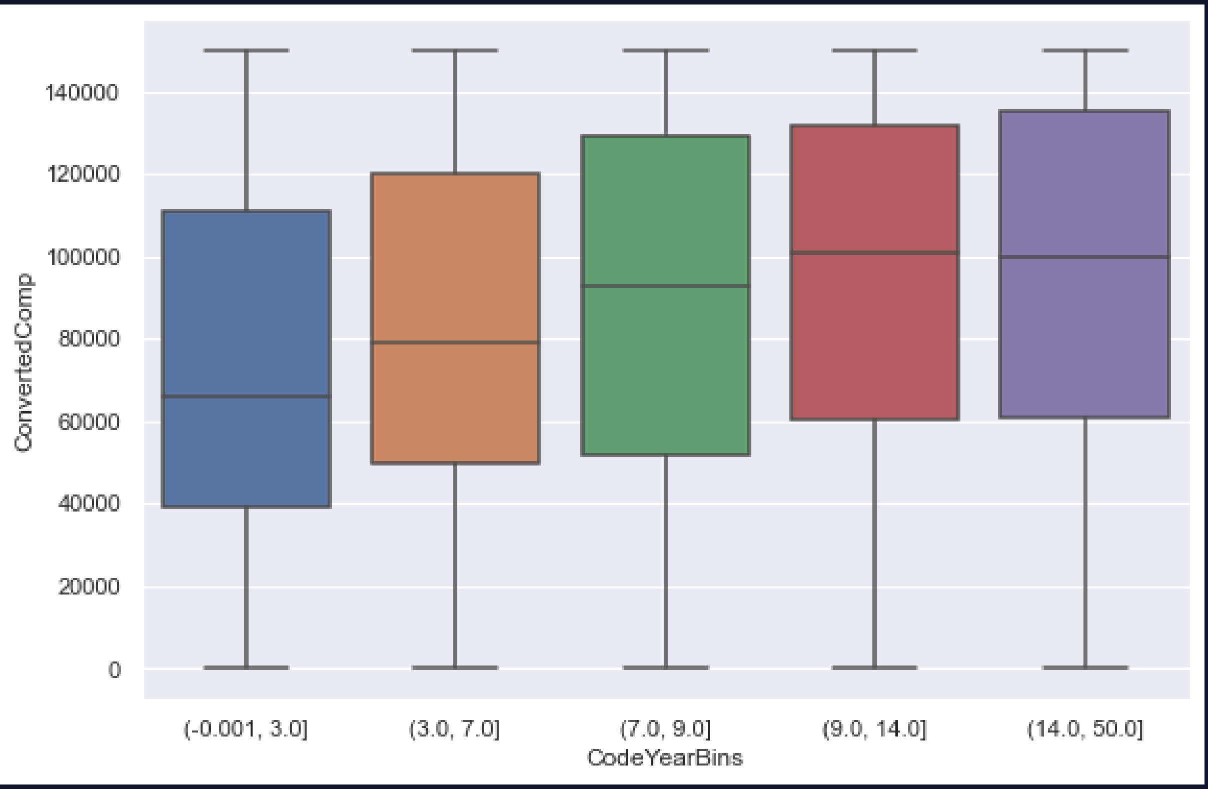
Note: *You can click on images to enlarge them.*

You see that although there are some outlier data points for each column, the overall distribution is fairly consistent year-over-year. This indicates that there is a strong correlation between the data points, which should tell a good story about how experience can translate into compensation. Since there is a clear trend with the data points, you decide the best method for filling in the missing data for these two columns is through Multiple Imputation:

```
import numpy as np
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.model_selection import train_test_split

imputedf = df[['YearsCodePro','ConvertedComp']]

traindf, testdf = train_test_split(imputedf, train_size=0.1)

# Create the IterativeImputer model to predict missing values
imp = IterativeImputer(max_iter=20, random_state=0)

# Fit the model to the the test dataset
imp.fit(imputedf)

# Transform the model on the entire dataset
compdf = pd.DataFrame(np.round(imp.transform(imputedf),0), columns=['YearsCodePro','ConvertedComp'])
```

The above code will loop through (up to 20 times), and fill in the missing data based on the context provided by the other column. This should create data points that are indicative of the overall trend of the data. Now, you can analyze the relationship
between `YearsCodePro` and `CinvertedComp` through the use of a boxplot like so:

```
compPlotdf = compdf.loc[compdf['ConvertedComp'] <= 150000]
compPlotdf['CodeYearBins'] = pd.qcut(compPlotdf['YearsCodePro'], q=5)

sns.boxplot(x="CodeYearBins", y="ConvertedComp",
        data=compPlotdf)
```



Note: *You can click on images to enlarge them.*

The plot above validates your hypothesis from before. While there are high (and low) earning developers at every experience level, experience appears to correlate with compensation. The more experienced a developer was, the more (on average) they were compensated.

Summary and Results

At this point, we have analyzed information about the developer community from a variety of points of view. Our client understands the global presence of the developer community, their varied backgrounds, and how their experience translates into compensation. Overall, these statistical analyses can guide actions in moving forward with a staffing plan that aligns with your client's growth plan and technical requirements.

By using a variety of techniques for handling missing data, you were able to reliably curate a cleaner dataset to fuel this set of analyses. These strategies allow you to salvage otherwise messy data, and should help you in the future with other datasets.

You can download an [example solution and the resulting datasets here].

Dataset Acknowledgements

The dataset provided is the result of a series of surveys hosted by Stack Overflow to understand their developer community. For the purposes of this project, the dataset has been slightly modified for the years 2018-2020.