Data Collection Methodology, Sources, Tools, and Techniques

Melody Goldanloo, Jenna Ramsey-Rutledge, Byron Selvage

Introduction:

We found that the most detailed and comprehensive collection of data was from Education Data Explorer. This API streamlined the data collection process and allowed us to collect a large amount of data without code. We used their API to collect a .csv file with over 1900 rows and 142 features.

```
# Keep only high school data
df = df[df['school_level'] == 'High']
```

```
# Read the data and add some error handling for file not found
try:
    df = pd.read_csv("/content/drive/MyDrive/Colab_Notebooks/Schools_Data.csv")
except pd.errors.ParserError as e:
    print(f"Error reading file: {e}")
```

Preprocessing Steps:

After data collection, we wanted to narrow our focus to just high schools.

Next, we determined the features that make sense for our model. This dropped us down to 39 features. We cleaned the data by removing rows that had more than 3 missing values.

```
# Drop rows with multiple missing values (more than 3)
# (grad_rate_midpt and cohort_num are giving every row at least 2 NA so thats why the 5)
df_cleaned = df.dropna(thresh=len(df.columns) - 5)
```

Next, we imputed all remaining missing values with the median.

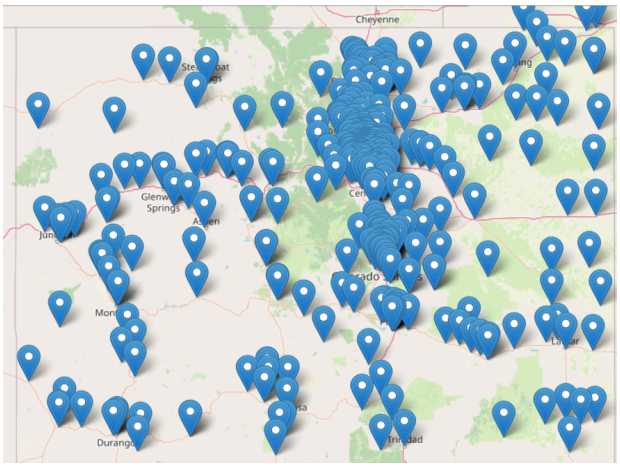
```
# Fill missing values with median - can refine this later, other fill methods may make more sense
num_cols = df_cleaned.columns.difference(['city_location', 'urban_centric_locale'])
df_cleaned[num_cols] = df_cleaned[num_cols].fillna(df_cleaned[num_cols].median())
df_cleaned.info()
```

Finally, we converted all the data to the correct data types.

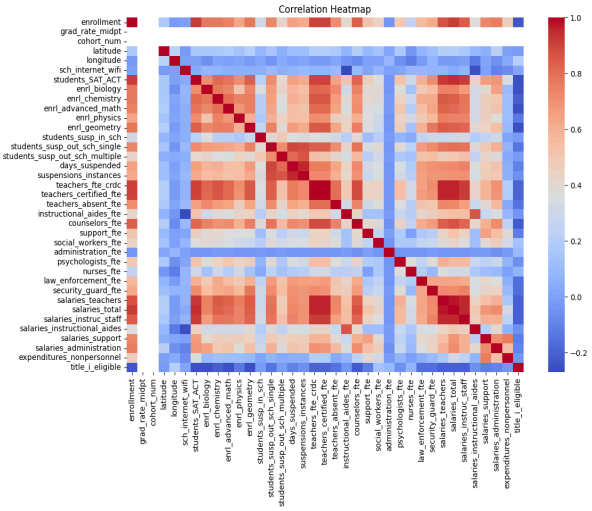
```
# Remap "Yes" and "No" to 1 and 0
df_cleaned['title_i_eligible'] = df_cleaned['title_i_eligible'].map({'Yes': 1, 'No': 0})
df_cleaned['sch_internet_wifi'] = df_cleaned['sch_internet_wifi'].map({'Yes': 1, 'No': 0})

# Convert numeric data stored as strings to floats
df_cleaned[['enrollment', 'salaries_teachers', 'salaries_instruc_staff', 'students_susp_in_sch', 'students_susp_out_sch_singl
```

To explore the data, we used the folium library to create an interactive map of all the locations of the schools and what city they are in.

```python
# Create a base map
map_center = [39.5501, -105.7821]  # Center of Colorado
m = folium.Map(location=map_center, zoom_start=7)

# Add markers for each school
for _, row in df.iterrows():
    folium.Marker(
        location=[row['latitude'], row['longitude']],
        popup=row['city_location'],
    ).add_to(m)

# Save the map to an HTML file
m.save('colorado_cities_map.html')
```

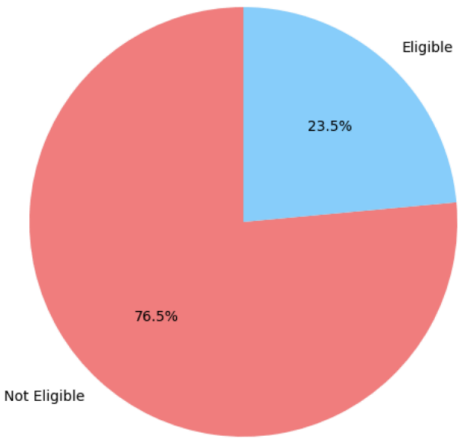


This visualization was very useful in showing what spread we had across the entirety of Colorado. Additional visualizations included creating a correlation map, a pie chart, and a bar graph.

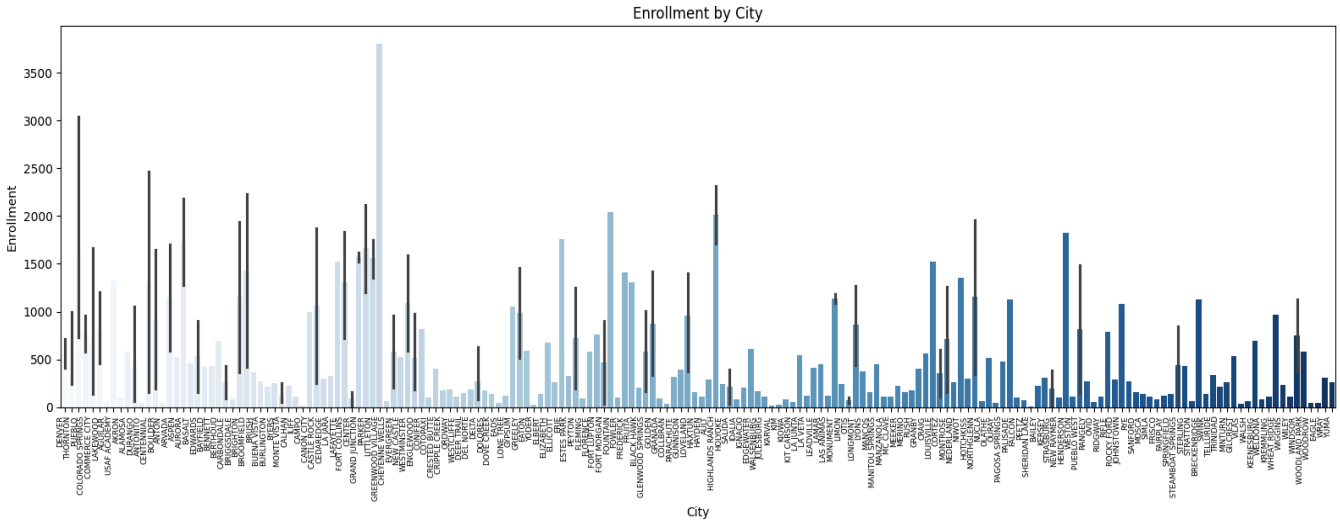



```python
# Plot correlation heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(df_cleaned[num_cols].corr(), annot=False, fmt='.2f', cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

```python
# Create the pie chart
eligibility_counts = df['title_i_eligible'].value_counts()
plt.figure(figsize=(8, 6))
plt.pie(eligibility_counts, labels=['Not Eligible', 'Eligible'], autopct='%1.1f%%', startangle=90, colors=['lightcoral', 'lightskyblue'])
plt.title('Title I Eligibility Distribution')
plt.axis('equal')
plt.show()
```

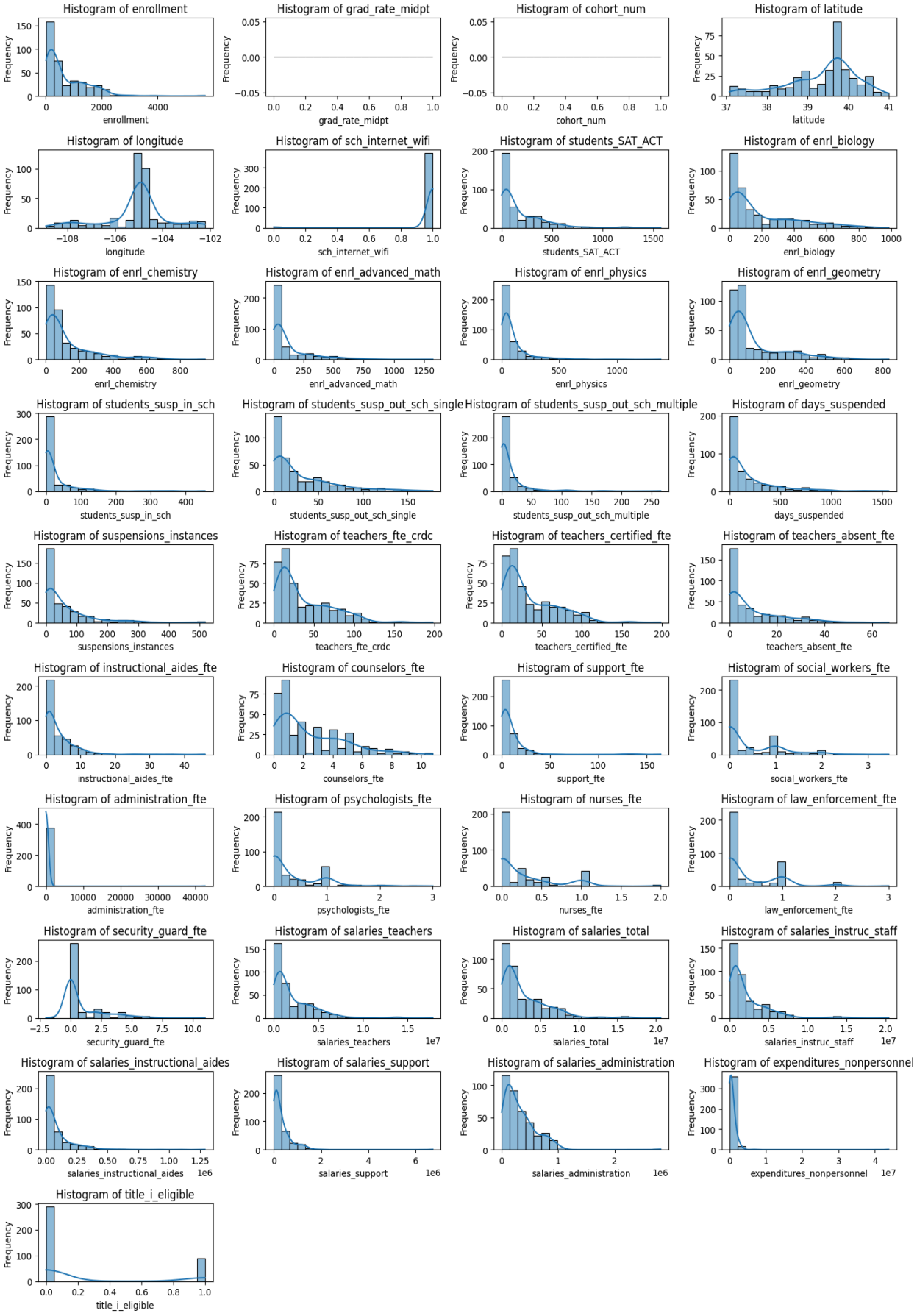


```
# Create the bar chart
plt.figure(figsize=(16, 6))
sns.barplot(x='city_location', y='enrollment', data=df_cleaned, palette='Blues')
plt.title('Enrollment by City')
plt.xlabel('City')
plt.ylabel('Enrollment')
plt.xticks(rotation=90, ha='right', fontsize=6)
plt.tight_layout()
plt.show()
```



The last visualization we completed was plotting histograms of all the columns.

```
# Plotting histograms of numerical values
plt.figure(figsize=(15, 20))
for i, col in enumerate(df_cleaned.select_dtypes(include='number')):
    plt.subplot(10, 4, i + 1)
    sns.histplot(df_cleaned[col], bins=20, kde=True)
    plt.title(f'Histogram of {col}')
    plt.xlabel(col)
    plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```


Histogram of enrollment
Histogram of grad_rate_midpt
Histogram of cohort_num
Histogram of latitude
Histogram of longitude
Histogram of sch_internet_wifi
Histogram of students_SAT_ACT
Histogram of enrl_biology
Histogram of enrl_chemistry
Histogram of enrl_advanced_math
Histogram of enrl_physics
Histogram of enrl_geometry
Histogram of students_susp_in_sch
Histogram of students_susp_out_sch_single
Histogram of students_susp_out_sch_multiple
Histogram of days_suspended
Histogram of suspensions_instances
Histogram of teachers_fte_crdc
Histogram of teachers_certified_fte
Histogram of teachers_absent_fte
Histogram of instructional_aides_fte
Histogram of counselors_fte
Histogram of support_fte
Histogram of social_workers_fte
Histogram of administration_fte
Histogram of psychologists_fte
Histogram of nurses_fte
Histogram of law_enforcement_fte
Histogram of security_guard_fte
Histogram of salaries_teachers
Histogram of salaries_total
Histogram of salaries_instruc_staff
Histogram of salaries_instructional_aides
Histogram of salaries_support
Histogram of salaries_administration
Histogram of expenditures_nonpersonnel
Histogram of title_i_eligible
Frequency
enrollment
grad_rate_midpt
cohort_num
latitude
longitude
sch_internet_wifi
students_SAT_ACT
enrl_biology
enrl_chemistry
enrl_advanced_math
enrl_physics
enrl_geometry
students_susp_in_sch
students_susp_out_sch_single
students_susp_out_sch_multiple
days_suspended
suspensions_instances
teachers_fte_crdc
teachers_certified_fte
teachers_absent_fte
instructional_aides_fte
counselors_fte
support_fte
social_workers_fte
administration_fte
psychologists_fte
nurses_fte
law_enforcement_fte
security_guard_fte
salaries_teachers
salaries_total
salaries_instruc_staff
salaries_instructional_aides
salaries_support
salaries_administration
expenditures_nonpersonnel
title_i_eligible
1e7
1e6

Our last preprocessing steps were to scale the data appropriately and perform PCA to reduce the dimensions of our data.

```
y = df_cleaned['grad_rate_midpoint']
X = df_cleaned.drop('grad_rate_midpt')

#Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
```

Discussion of Challenges and Strategies to Overcome Challenges:

One of the first major challenges we faced was determining what datasets had the most complete data. The Kaggle dataset we thought we would use was riddled with missing values in very important features defining college readiness, graduation rates, and teacher salaries. The API offered minimal missing values and more features.

Another challenge was figuring out the best ways to collaborate on the code. We attempted to use Google Collab in order to code simultaneously. This presented some issues like requiring refreshes and additional saving. The learning curve for the Mines GitLab was a bit too steep to attempt just yet, so Google Collab made do. In the future, we may change what we use to collaborate on.

Furthermore, picking out a strategy for imputation was difficult and we will likely adjust this to be more case dependent or use a matrix in the future for better predictions of those missing values. In the meantime, we used the median to replace the missing values.

One of the challenges of dealing with geographical data is the visualizations. We were able to use the folium library to expand on the ability to plot. However, we are still fairly new to using this library, so we will need to look deeper into the documentation to see what is available to us for future visualizations.