CS 2410 - 01
Group #5 Presentation

May 8, 2023

Presented by:

Andrew Benavides
Marc Cruz
Scott Chang

Overview:

- **Problem Statement**
  - Introduction and Background
- **Data Sets**
  - About Our Data sets
  - Resources
- **Data Wrangling**
  - Data Cleaning and Examples
- **Visualizations**
  - What's In Our Data?
  - Coding and Visualization
  - Explanation of Our Graph
- **Results/Findings**
  - Findings and Interpretations
- **Future Work**
- **Code Demonstration**

Problem Statement

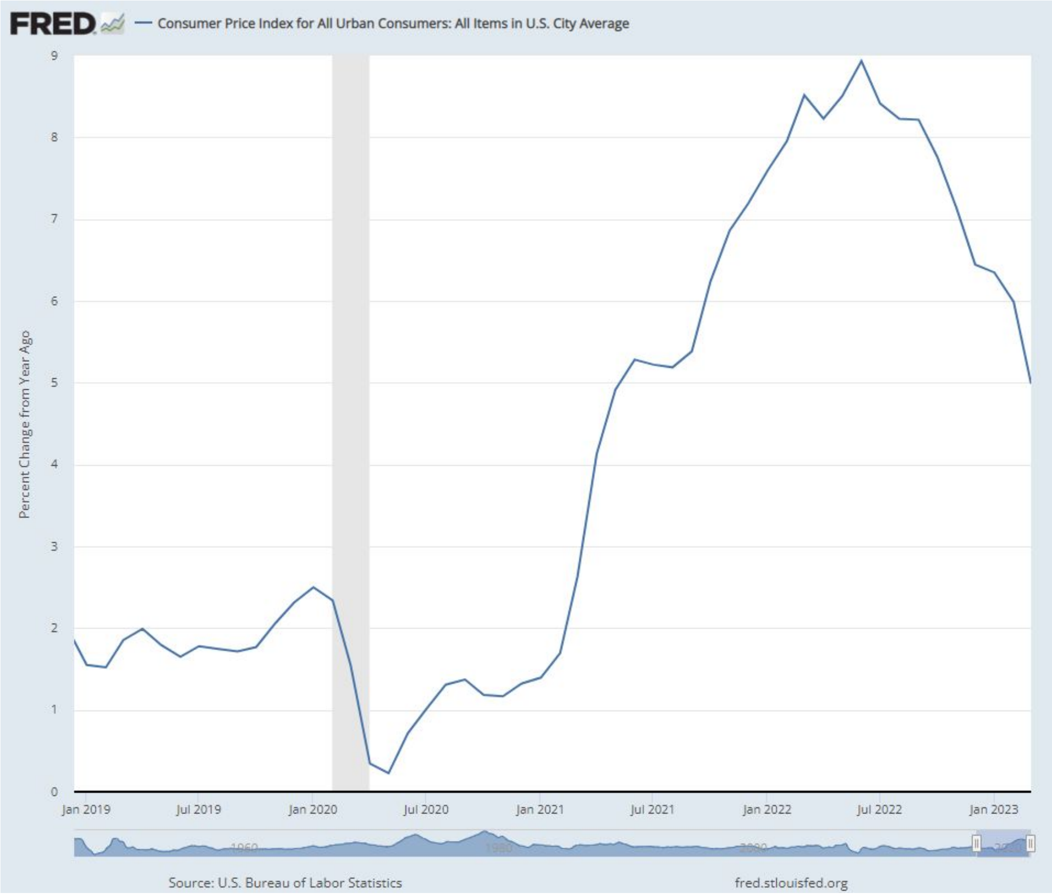
Introduction

Problem Statement

For a given job role,
in which city does your income provide the most purchasing power?

Background:

- Different parts of the country differ in salary pay and wage scales for the same job role.
- Each part of the country also differs costs of living.
- Since 2021, the U.S. economy has experienced rising inflation.
- U.S. Federal Reserve (the "Fed") aims for 2% inflation as the sweet spot for price stability. However, since 2021, inflation has been staggering and well above the 2% mark.
- Thus, we seek to determine in which city/area a person would have the greatest purchasing power for a given job role.



Data Sets

Ask A Manager Salary Survey 2021

(Source: www.askamanager.org)

Features:

- Large set, over 24,000 entries
- Many attributes, such as:
  - Age
  - Job Title
  - Work City
  - State
  - Gender
  - Race
  - Industry
  - Education

AdvisorSmith Cost of Living Index

(Source: https://advisorsmith.com/data/coli)

Features:

- Cost-of-Living index for 510 cities reported.
- Matches same year of *Ask A Manager Survey (2021)*
- Constructed based on reliable yet public sources such as:
  - U.S. Bureau of Labor Statistics
  - U.S. Department of Housing
  - Zillow's Home Value Index
  - U.S. Department of Energy's Natural Gas
  - Bureau of Economic Analysis

U.S. BUREAU OF LABOR STATISTICS

(Source: https://www.bls.gov/soc/2018/major_groups.htm)

2018 Standard Occupational Classification System

NOTE: The information on this page relates to the 2018 SOC, please see the 2010 SOC System for information on the previous version of the SOC.

Each occupation in the 2018 SOC is placed within one of these 23 major groups:

- 11-0000 Management Occupations
- 13-0000 Business and Financial Operations Occupations
- 15-0000 Computer and Mathematical Occupations
- 17-0000 Architecture and Engineering Occupations
- 19-0000 Life, Physical, and Social Science Occupations
- 21-0000 Community and Social Service Occupations
- 23-0000 Legal Occupations
- 25-0000 Educational Instruction and Library Occupations
- 27-0000 Arts, Design, Entertainment, Sports, and Media Occupations
- 29-0000 Healthcare Practitioners and Technical Occupations
- 31-0000 Healthcare Support Occupations
- 33-0000 Protective Service Occupations
- 35-0000 Food Preparation and Serving Related Occupations
- 37-0000 Building and Grounds Cleaning and Maintenance Occupations
- 39-0000 Personal Care and Service Occupations
- 41-0000 Sales and Related Occupations
- 43-0000 Office and Administrative Support Occupations
- 45-0000 Farming, Fishing, and Forestry Occupations
- 47-0000 Construction and Extraction Occupations
- 49-0000 Installation, Maintenance, and Repair Occupations
- 51-0000 Production Occupations
- 53-0000 Transportation and Material Moving Occupations
- 55-0000 Military Specific Occupations

O*NET OnLine

*O*NET OnLine is sponsored by the U.S. Department of Labor*

(Source: https://www.onetonline.org)

Occupation Keyword Search

Occupations matching "dental"

Search again: dental Go

20 occupations shown Show matches: Closest All How do they match?

Code	Occupation
51-9081.00	Dental Laboratory Technicians
31-9091.00	Dental Assistants Bright Outlook
29-1292.00	Dental Hygienists
29-1021.00	Dentists, General
29-1023.00	Orthodontists
29-1024.00	Prosthodontists
29-1022.00	Oral and Maxillofacial Surgeons
25-1071.00	Health Specialties Teachers, Postsecondary
49-9062.00	Medical Equipment Repairers
11-3051.00	Industrial Production Managers
11-9111.00	Medical and Health Services Managers
11-9199.00	Managers, All Other
25-1194.00	Career/Technical Education Teachers, Postsecondary

Data Wrangling

Data Cleaning and Processing

- ***Ask a Manager*** **dataset - remove rows not compatible with our criteria**

- **Identify all unique job titles and the industry they work in**
  - Some titles are ambiguous
  - Used “Job Title Context” column for additional clues.
  - Same job titles could mean different roles in different industries.

- **Standardizing job roles by assigning BLS Taxonomic code to identify job family/type**
  - O-Net keyword search to help identify roles
  - Where unclear, we make an informed guess to the type of role

- **14,086 distinct, self-reported job titles/industry combinations identified**

- **Joined back to original dataset**

- **Final table of 17,025 usable records**

Data Cleaning and Processing

- ***Advisor Smith*** **- Cost of Living (CoL) dataset**

- **Index based on score of 100, and represents comparison to their defined "average American city".**

- **However, the "median" of values in their index is 91.9.**
  - Implies that much more than 50% of locations have below "average" income!
  - More meaningful to compare median values as payscale is relative.
  - Rescaled CoL indices to be relative to median value
  - ex: if a city was previously CoL=100, rescaled against the median value, the city is now Relative CoL = 108.8.

- **Where CoL was available for a city listed in *Ask a Manager survey*, CoL index was matched.**

- **Cities in *Ask a Manager,* suburbs of a larger Metro area are considered to be part of the larger listed city.**
  - Best guess depending on distance to larger city.

Data Cleaning and Processing

Example - Data not compatible (Currency, Country, City not relevant)

at industry ou work	Job title	If your job title needs additional	What is your annual salary? (You'll	How much additional monetary	Please indicate the currency	If "Other," please indicate the	If your income needs additional	What country do you work in?	If you're in the U.S., what state do you	What city do you work in?	How many years of professional	How many years of professiona
nputing or T	Front end developer		48000		EUR			France		Paris	5-7 years	2 - 4 years
cation (Prin	teacher		58000		EUR			germany		mannheim	21 - 30 years	5-7 years
keting, Adv	Marketing Manager		91000		CAD			Canada		Swift current	5-7 years	5-7 years
nputing or T	Principal consultant		110000		GBP			United Kingdom		Glasgow	21 - 30 years	21 - 30 year
ineering or	Senior process engineer		424823	71845	Other	THB		Thailand		Bangkok	5-7 years	5-7 years
nputing or T	contract manager		80000	6000	CAD			canada		toronto	1 year or less	1 year or le
& Design	Product Development Manager		60200		EUR			Austria		Salzburg	8 - 10 years	5-7 years
nputing or T	Senior Software Engineer		3000000		Other	INR		INDIA		BANGALORE	5-7 years	5-7 years
ineering or	Foreman		20000	2000	EUR			Slovenia		Ljubljana	2 - 4 years	2 - 4 years
nputing or T	Software Engineer		88000	25000	EUR			Netherlands		Remote	8 - 10 years	8 - 10 years
iness or Cor	Associate Cons	I lead and supp	680000	0	Other	INR		India		Bangalore	2 - 4 years	2 - 4 years
ail	Product Manager		30000	0	USD			Thailand		Bangkok	5-7 years	2 - 4 years
ecommunica	Intermediate Data analyst		79000	5000	CAD			Canada		Toronto	5-7 years	2 - 4 years
cation (High	Director of Stuc	English as a sec	25000	0	EUR		Unpaid for less	Spain		Madrid	8 - 10 years	5-7 years
ehousing	Forklift driver		19500		GBP			England		Peterborough	5-7 years	2 - 4 years
pitality & Ev	Senior Analyst	Email developn	31538		GBP			Scotland		Glasgow	8 - 10 years	2 - 4 years
& Design	UX/UI Designer	Sole designer a	100000	0	AUD/NZD			New Zealand		Auckland	2 - 4 years	2 - 4 years

Data Cleaning and Processing

Example - Job Role Ambiguity

- **Administrative, Financial, Data?**

What industry do you work in?	Job title	If your job title needs additional context, please clarify here:
Organized Labor	Analyst	I work with data.

- **Same title, different jobs:**
    - Business process: code 13
    - Librarian: code 25

What industry do you work in?	Job title	If your job title needs additional context, please clarify here:
Health Insurance	Knowledge Analyst	Analyze business and project requirements and the develop procedural documentation.
Government and Public Administra	Knowledge Analyst	Iâ€™m basically a librarian

- **General Ambiguity:**
    - Data management: 15 or 43!
    - Business: code 13
    - Admin Assistant: 43
    - Health care: 29 or 43
    - Law: 23
    - Think Tank Specialist?
    - Nonprofits?

Environmental sciences	Specialist	General data management and reporting
Marketing, Advertising & PR	specialist	part-time
Nonprofits	Specialist	
Health care	Specialist	Pre certification & patient placement for outpatient care.
Business or Consulting	Specialist	Borderline between individual contributor and management for consultants focused on on
Think tank	Specialist	
Nonprofits	Specialist	
Nonprofits	Specialist	
Education (Higher Education)	Specialist	Iâ€™m an admin assistant with additional responsibilities
Law	Specialist	Professional Development

Data Cleaning and Processing

Example: Cost of Living - Rescaling

- For each city, CoL was rescaled against the median value of the AdvisorSmith CoL dataset.

SUM = 1 + (D2-H2) / H2

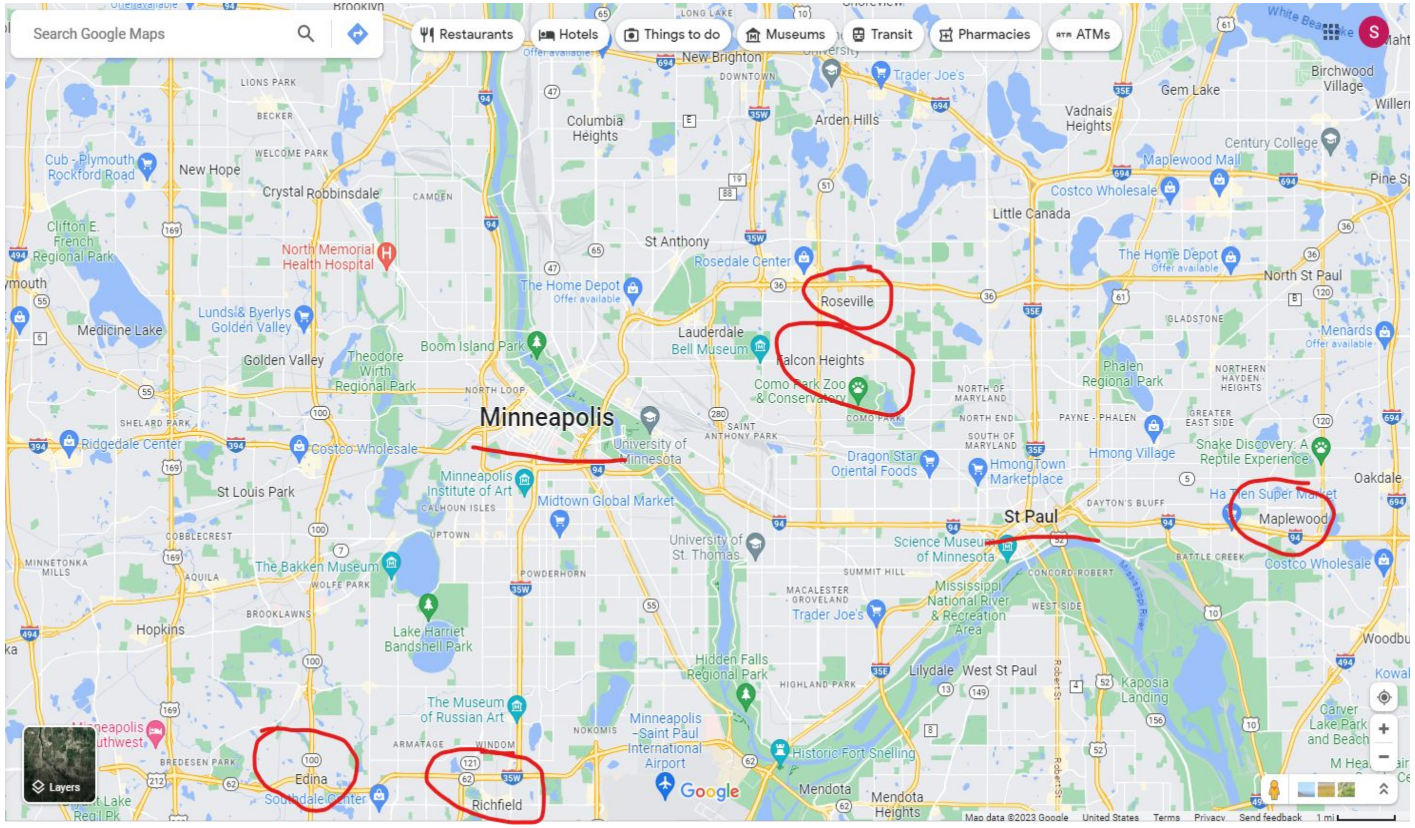
Rank	City	State	Cost of Living Index		Rescale		Median
1	Aberdeen	WA	97.6		H2		91.9
2	Abilene	TX	89.1		0.969532		
3	Adrian	MI	90.5		0.984766		
4	Akron	OH	89.4		0.972797		
5	Alamogor	NM	85.8		0.933624		
6	Albany	GA	87.3		0.949946		
7	Albany	NY	100.1		1.089227		
8	Albany	OR	105.4		1.146899		

Data Cleaning and Processing

Example: Grouping Suburbs with their Cities, Clarifying Area Nicknames

City	State	Cost of Living Ind
Mankato	MN	95.3
Minneapolis	MN	105.4
Rochester	MN	97.3

MN	Twin Cities Metro
MN	Twin Cities
MN	Twin Cities
MN	Twin Cities Suburbs
MN	Falcon Heights
MN	Twin Cities
MN	Twin Cities
MN	Maplewood
MN	Twin Cities
MN	Twin Cities
MN	Edina - Minneapolis
MN	Maplewood
MN	Twin Cities
MN	Hopkins
MN	Maplewood
MN	Richfield
MN	Twin Cities (State Wide Org)
MN	Twin Cities
MN	Twin Cities
MN	Roseville
MN	Twin Cities Metro
MN	Twin Cities
MN	Twin Cities Area
MN	Twin Cities
MN	Edina
MN	Roseville
MN	Edina

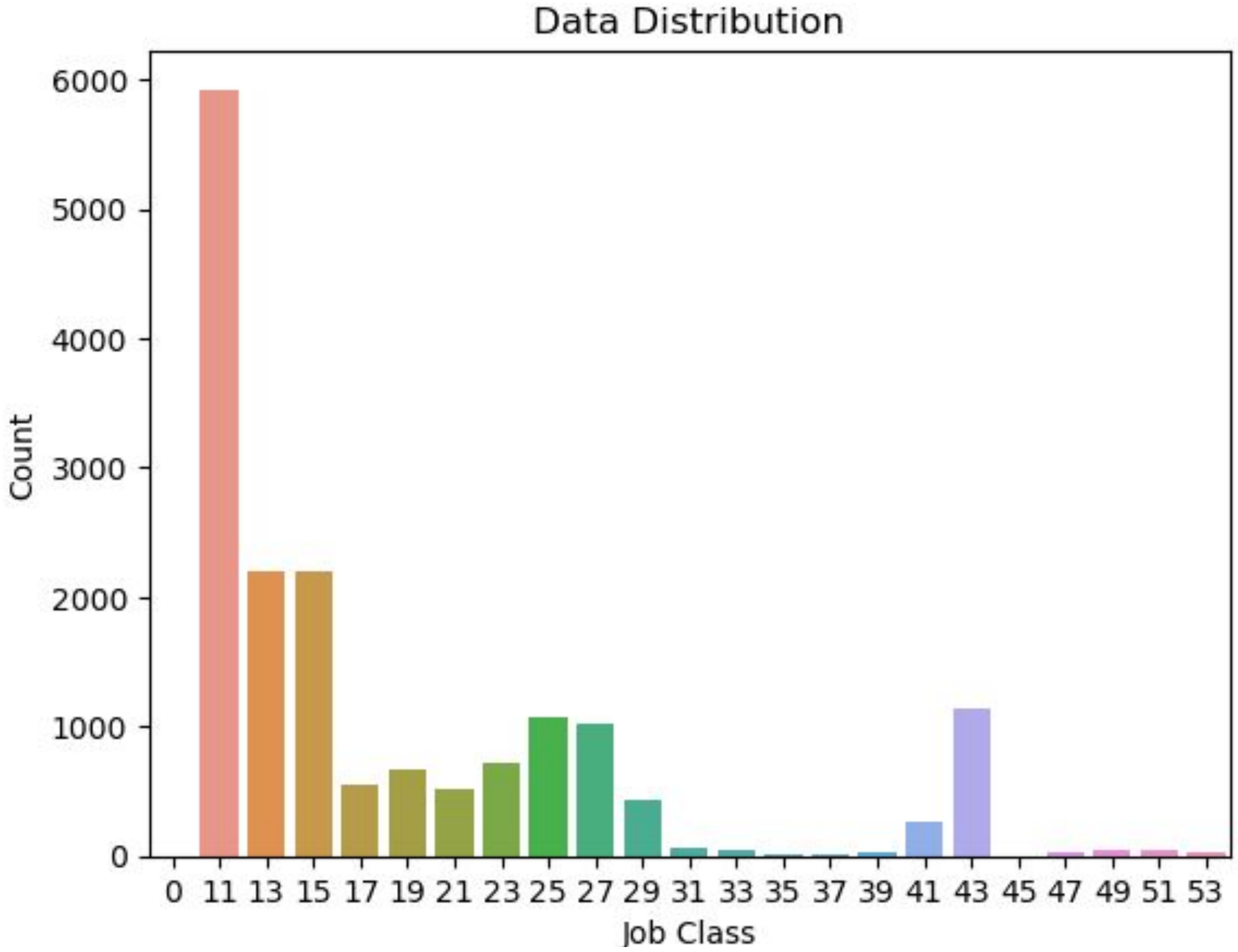


Visualizations

Job Class	Count
11	5923
13	2204
15	2197
43	1139
25	1064
27	1029
23	712
19	664
17	558
21	511
29	437
41	270
31	60
33	47
51	43
49	38
39	34
53	33
47	24
35	18
37	12
55	4
45	2
0	2

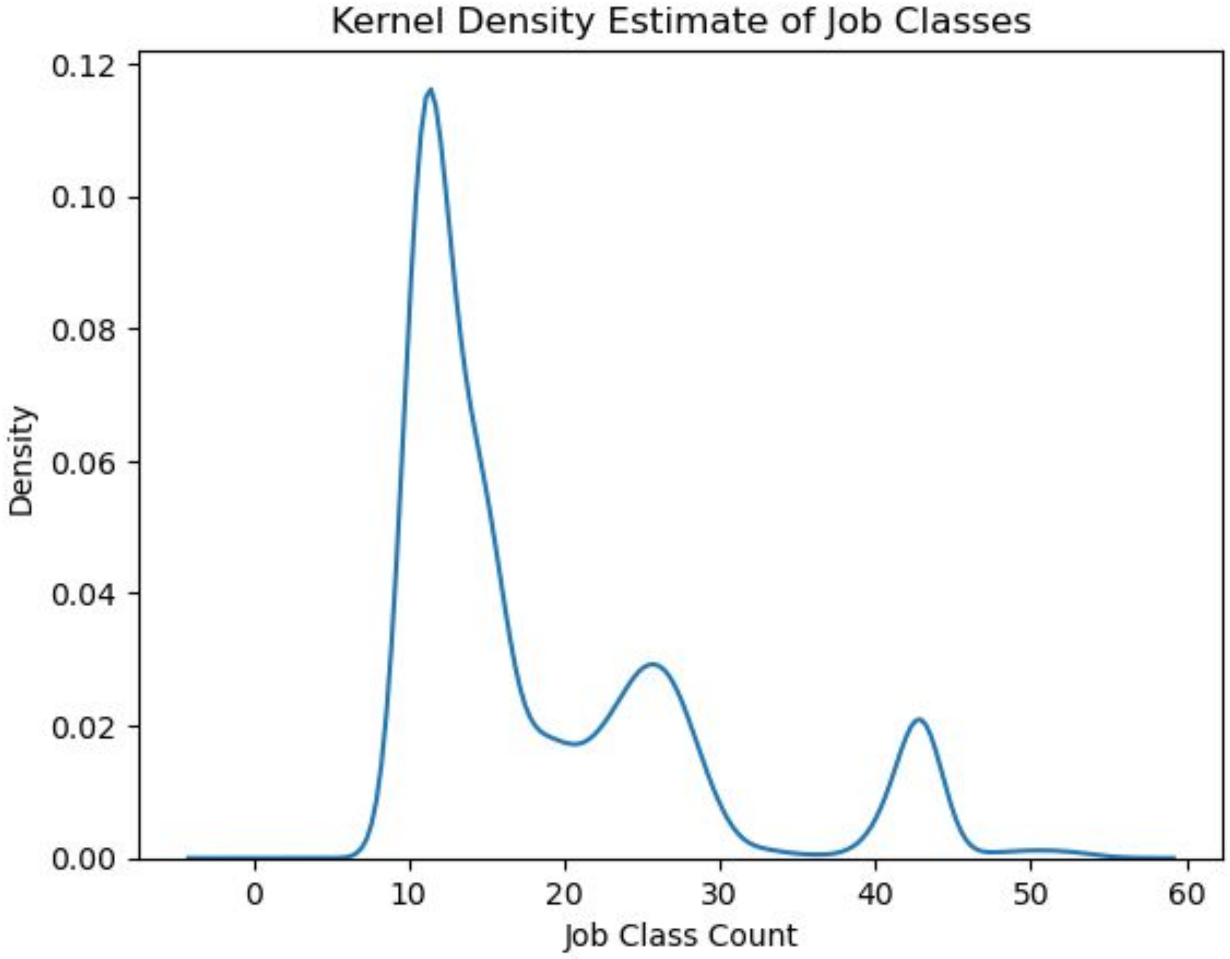
Data Initial Look:

What's represented in this dataset?



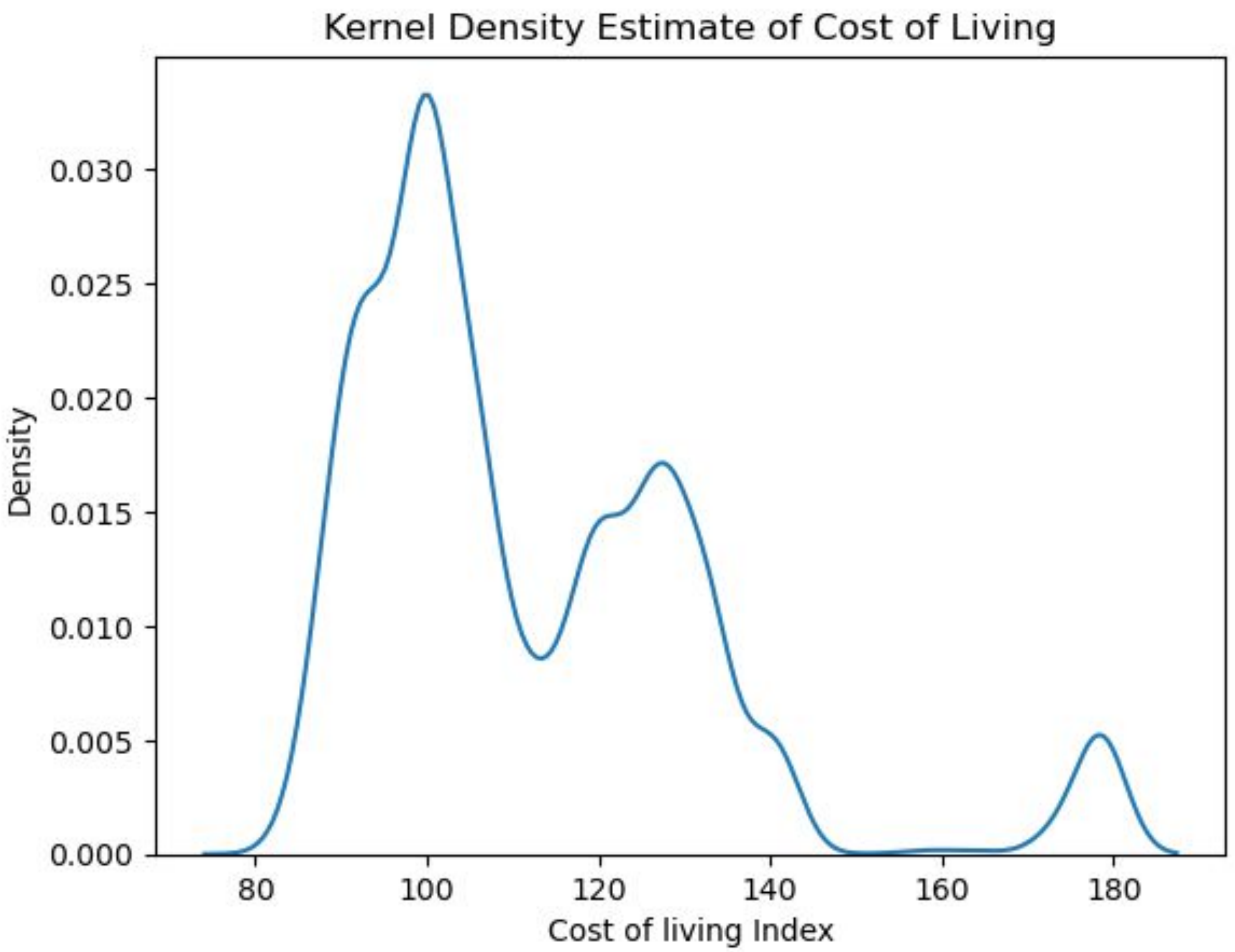
Data Initial Look:

What’s represented in this dataset?



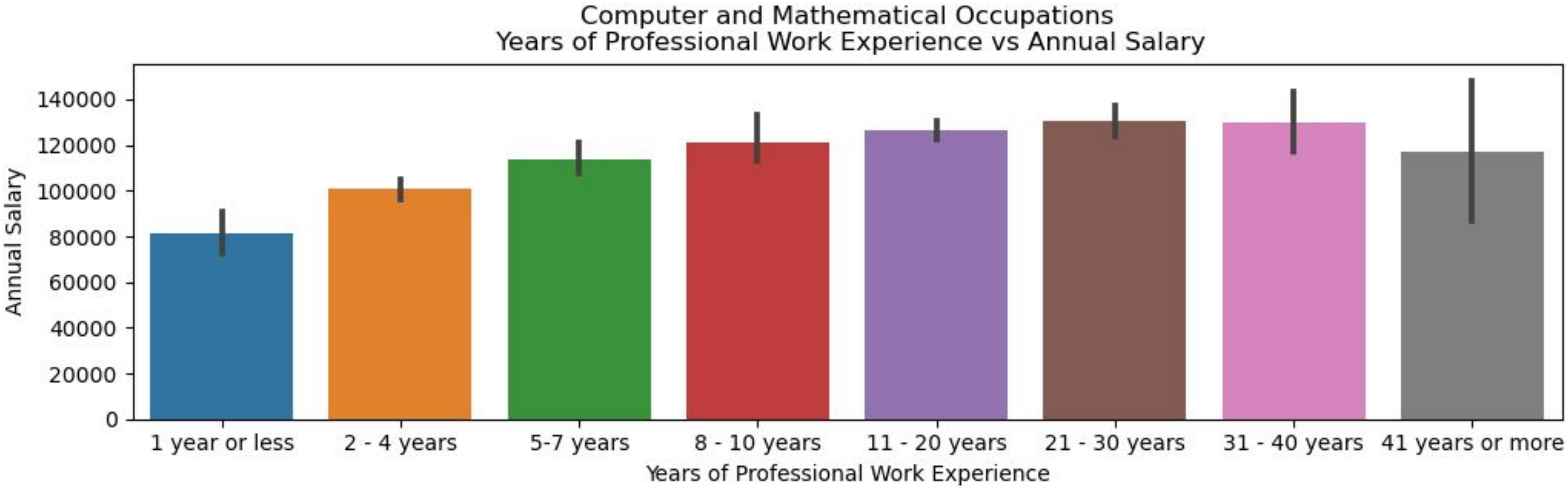
Data Initial Look:

What's represented in this dataset?



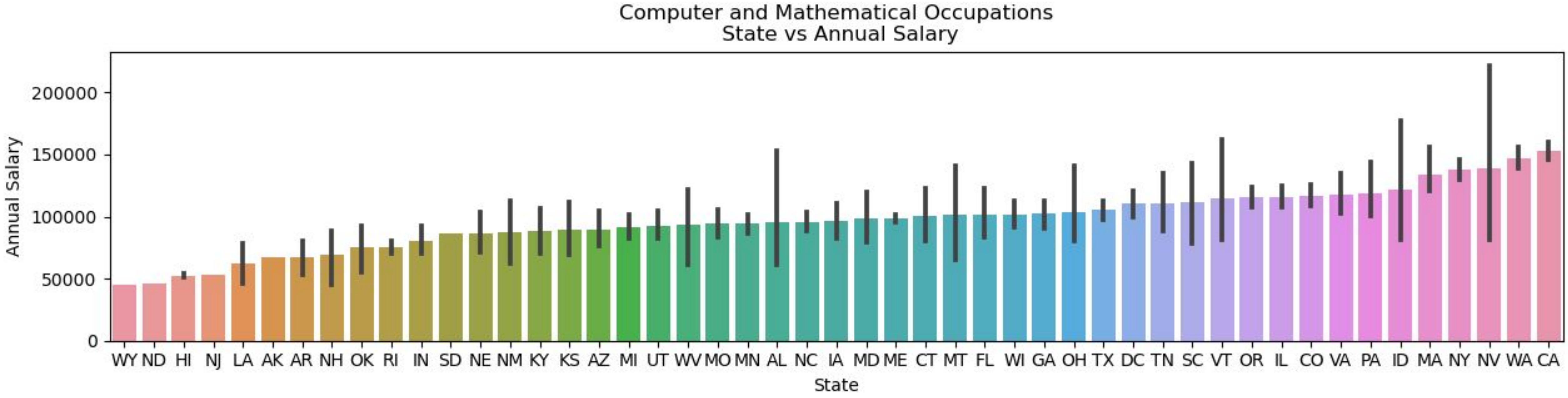
Data Initial Look:

What’s represented in this dataset?



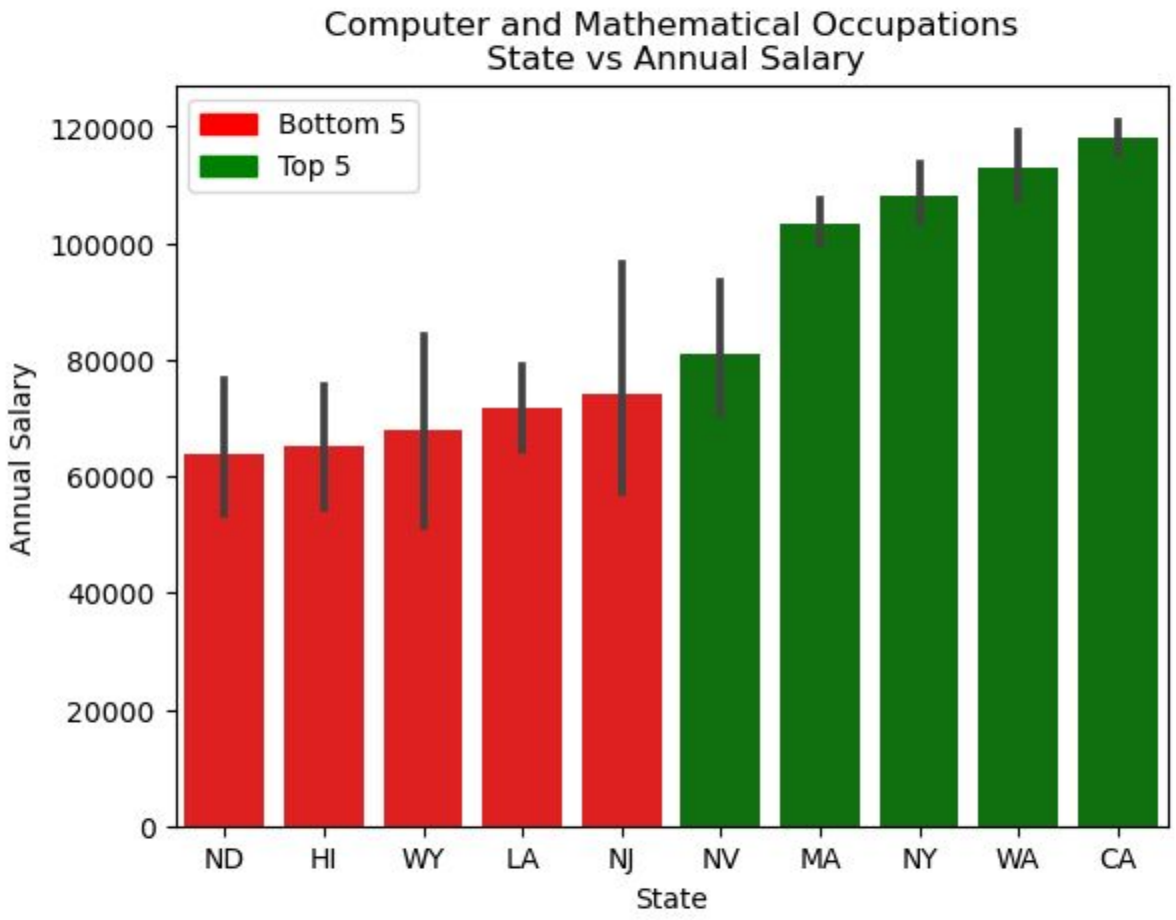
Data Initial Look:

What's represented in this dataset?



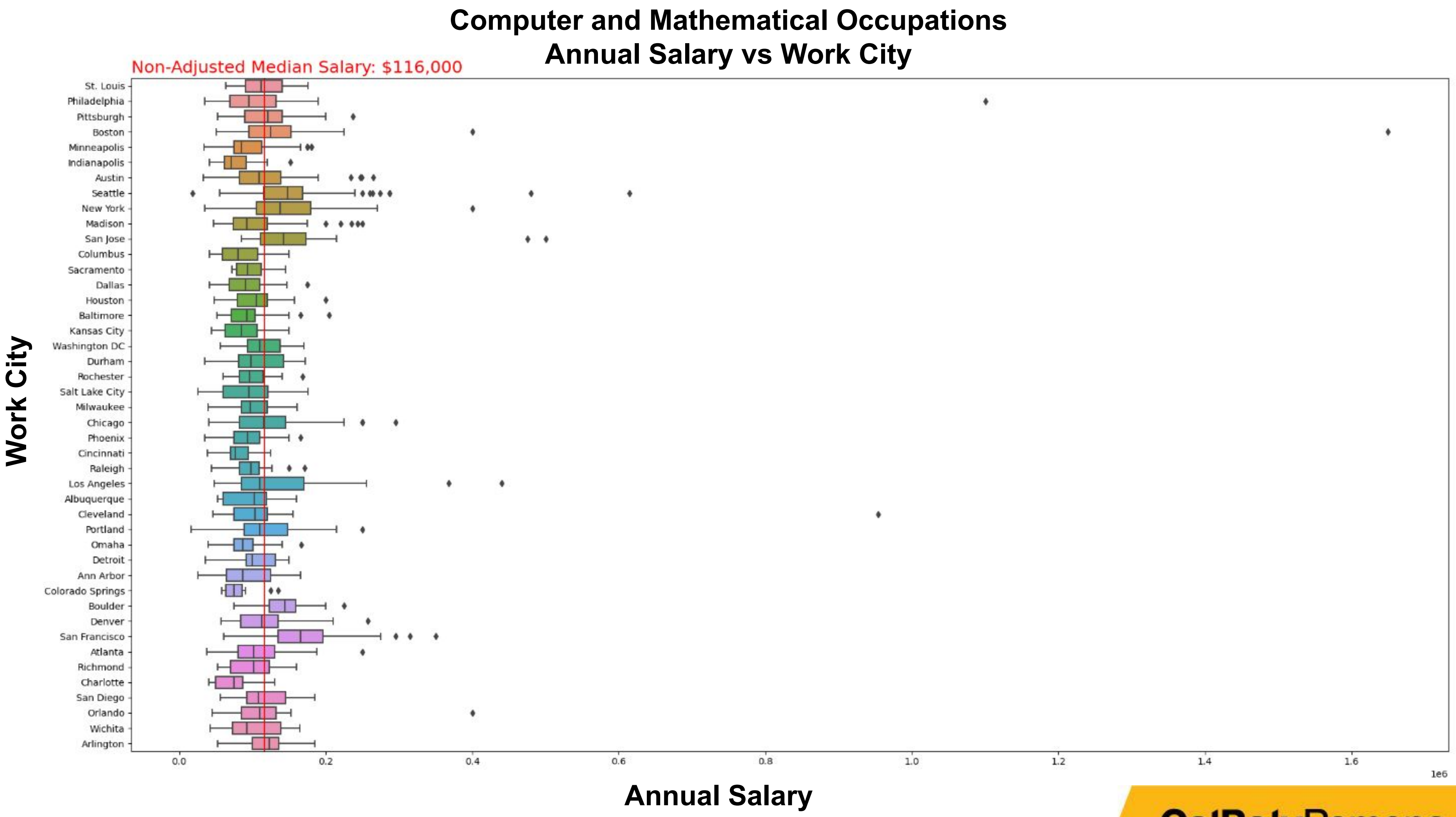
Data Initial Look:

What's represented in this dataset?



Data Initial Look:

What's represented in this dataset?



CalPolyPomona

Visual Code:

code for previous shown graphs

```
def box_colors(x, color1='red', color2='green'):
    '''Function used to make the specific palette used in function "graph_states_barplot()" '''
    num_boxes = len(x)
    half_num_boxes = num_boxes // 2
    colors = [color1] * half_num_boxes + [color2] * (num_boxes - half_num_boxes)
    return colors

def graph_states_barplot(df, SOC_group, label_x, label_y):
    '''Function specific for states vs Annual Salary takes in df, specified job class,
    desired feature comparisons
    '''

    #sets SOC_df to take in all occupations if set to 0
    if SOC_group == 0:
        SOC_df = df
    else:
        SOC_df = df.loc[df["Job Class"] == SOC_group]
    #Labeling
    title = job_type[SOC_group] + "\n " + label_x + " vs " + label_y
    states = ["WY","ND","HI","NJ","LA","MA","NY","NV","WA","CA"]
    new_df = df.loc[df["State"].isin(states)]
    #Organizing states to be arranged least to greatest annual salary
    group_means=new_df.groupby([label_x])[label_y].mean().sort_values(ascending=True)

    sns.barplot(data = new_df,
                x = label_x,
                y = label_y,
                order=group_means.index,
                palette = box_colors(new_df[label_x].unique())
                ).set(title=title)
    colors = ['red','green']
    legend_labels = ["Bottom 5", "Top 5"]
    legend_handles = [plt.Rectangle((0,0),1,1, color=color) for color in colors]
    plt.legend(legend_handles, legend_labels)

    print(new_df[label_x].value_counts())
graph_states_barplot(dataset,15,"State","Annual Salary")
```

```
def graph_barplot(df, SOC_group, label_x, label_y):
    '''Function used to quickly make barplots for different combinations of features'''
    if SOC_group == 0:
        SOC_df = df
    else:
        SOC_df = df.loc[df["Job Class"] == SOC_group]
    title = job_type[SOC_group] + "\n " + label_x + " vs " + label_y
    fig, ax = plt.subplots(figsize=(15, 3))
    print(SOC_df[label_y].describe())
    group_means=SOC_df.groupby([label_x])[label_y].mean().sort_values(ascending=True)

    sns.barplot(data = SOC_df,
                x = label_x,
                y = label_y,
                ax = ax,
                order=group_means.index
                ).set(title=title)
```

```
def graph_kernal_density_plot(df,SOC_group, label_x,label_y):
    '''Function used to quickly make kernal density plots for different features'''
    SOC_df = df.loc[df["Job Class"] == SOC_group]
    title = job_type[SOC_group] + "\n " + label_x + " vs " + label_y

    p = sns.kdeplot(data = SOC_df, x = label_y)
    p.set_xlabel(label_x)
    p.set_ylabel(label_y)
    p.legend_.set_title(title)
```

```
def graph_boxplot(df, income_cat, SOC_group, cat_num, the_median):
    income_raw = income_cat  # income_cat is the category of income (ie normalized or raw inc
    job_class_num = SOC_group
    cat_name = cat[cat_num]
    statecount = df
    median_raw = the_median
    fig, ax = plt.subplots(figsize=(25, statecount[cat_name].nunique()*.3))
    subdata_raw = sns.boxplot(data = statecount,
                              y = cat_name,
                              x = income_raw,
                              whis = 1.5,
                              showfliers = True,
                              width = .8,
                              ax = ax)
    subdata_raw.axvline(median_raw, color='red')
    plt.title(f'Non-Adjusted Median Salary: ${round(median_raw):,}', loc='left', fontsize = 1
```

Coding

- **To analyze data, we developed code to read the processed dataset and visualize.**

- **Developed 5 core functions to help analyze:**

  main function:

  - **true_pay()**

  support functions:

  - **normalized_pay()**
  - **raw_pay()**
  - **color_palette()**
  - **label_formatter()**

The Setup

- **Create dictionaries, to be used for function arguments and visualizations.**
  - job code and its corresponding description.
  - category/features we are interested in visualizing

```
#Dictionary that holds all the possible type of occupations and each are assigned a number
job_type = {
            0  : 'All Occupations' ,
            11 : 'Management Occupations' ,
            13 : 'Business and Financial Operations Occupations' ,
            15 : 'Computer and Mathematical Occupations' ,
            17 : 'Architecture and Engineering Occupations' ,
            19 : 'Life, Physical, and Social Science Occupations' ,
            21 : 'Community and Social Service Occupations' ,
            23 : 'Legal Occupations' ,
            25 : 'Educational Instruction and Library Occupations' ,
            27 : 'Arts, Design, Entertainment, Sports, and Media Occupations' ,
            29 : 'Healthcare Practitioners and Technical Occupations' ,
            31 : 'Healthcare Support Occupations' ,
            33 : 'Protective Service Occupations' ,
            35 : 'Food Preparation and Serving Related Occupations' ,
            37 : 'Building and Grounds Cleaning and Maintenance Occupations' ,
            39 : 'Personal Care and Service Occupations' ,
            41 : 'Sales and Related Occupations' ,
            43 : 'Office and Administrative Support Occupations' ,
            45 : 'Farming, Fishing, and Forestry Occupations' ,
            47 : 'Construction and Extraction Occupations' ,
            49 : 'Installation, Maintenance, and Repair Occupations' ,
            51 : 'Production Occupations' ,
            53 : 'Transportation and Material Moving Occupations'
            }
```

Job Code and its Corresponding Description

```
# categories/features represented in the data
cat        = {
              0 : 'Age',
              1 : 'Industry' ,
              2 : 'State' ,
              3 : 'Work City' ,
              4 : 'Gender',
              5 : 'Race' ,
              6 : 'Years of Professional Work Experience' ,
              7 : 'Years Professional Work Experience in Field' ,
              8 : 'Highest Education Level' ,
              9 : 'Job Title'
              }
```

Categories that can be visualized

Main Function:

true_pay(job classification, grouping category, min # of observations req., optional: search terms)

- **Takes in dataset and create a df from only rows/columns of interest and features of interest:**

```
queryslug = '==' if SOC_group != 0 else '!='

# setting up the dataframe for only those categories that we are interested from the raw dataset.
statecount = dataset[['Job Class',
                      income_norm,
                      income_raw,
                      cat[0],
                      cat[1],
                      cat[2],
                      cat[3],
                      cat[4],
                      cat[5],
                      cat[6],
                      cat[7],
                      cat[8],
                      cat[9]
                     ]].query(f'`Job Class` {queryslug} {job_class_num}')

# Filter out those records that do not have a minimum number of entries of the category selected.
# It is likely that too few entries would produce results that are not representative of actual trends.

query_list = f" `Job Title`.str.contains('')"      # Passing an emtpy query term if there are no keywords to search
                                                    # Otherwise, we will build a query search string
if terms !='':
    word_list = terms.split()
    query_list = ''
    for each in range(len(word_list)):
        query_list = query_list + f" `Job Title`.str.contains('{word_list[each]}' , case = False) "
        if (len(word_list) > 1) and (each < len(word_list)-1) : query_list = query_list + ' and '

# filter out those records that have Annual Salary less that Fed poverty rate ($12880 in 2021).
# Allow query of records if there are keywords to search.
# Finally, filter out records that do not have a minimum number of records (must be greater than number passed as argument)
statecount = statecount.loc[statecount['Annual Salary'] > 12880].query(query_list, engine='python').groupby(cat_name).filter(lambda x: x[cat_name].count()>min_count)
```

Main Function:

true_pay(job classification, grouping category, min # of observations req., optional: search terms)

- **Main body calculates all the variables needed for plotting, such as:**
  - Median of nominal income and normalized income
  - Creating lists of cities to be used for annotating and color coding:
  - Calculation, in %, of the relative increase or decrease of purchasing power
  - Dynamically setting the plot size limits based on query results
  - Dynamically creating plot order and label order

Main Function:

true_pay(job classification, grouping category, min # of observations req., optional: search terms)

```
    # Initial data processing.
    median_norm = statecount[income_norm].median()
    median_raw = statecount[income_raw].median()

    statecount['Job Class'] = statecount['Job Class'].astype('int')    # change the data

    # Find the order.  We are interested in determining which city falls below median in
    # but is above the median in the normalized list.
    # We start by creating a df that contains the list of city names and whether they are
    # medians (True/False)
    order_list_norm = statecount.groupby(by=[cat_name])[income_norm].median().sort_values
    order_list_raw = statecount.groupby(by=[cat_name])[income_raw].median().sort_values(a

    # We find the 25% and 75% quantiles to pass on as arguments to use for setting the pl
    # We are searching through both norm and raw values to find ideal horizontal scale fo
    ptile25 = min( min(statecount.groupby(by=[cat_name])[income_norm].quantile(.25)) ,  m
    ptile75 = max( max(statecount.groupby(by=[cat_name])[income_norm].quantile(.75)) , ma

    med_norm_city_above = order_list_norm >= median_norm     # True/False whether normali
    med_raw_city_above = order_list_raw >= median_raw     # True/False whether raw city i

# calculating distance to median before adjustment and distance to median after adjustmen
# calculating the relative percentage change of distance to median to gauge relative chan

    # We create a df from which to performce distance calculations from median of before
    # We set them to new dataframes.
    med_dist_norm = order_list_norm[:]
    med_dist_raw = order_list_raw[:]

    # We are caonverting the distance to a percentage.
    med_dist_ptage_norm = (med_dist_norm - median_norm)/median_norm
    med_dist_ptage_raw = (med_dist_raw - median_raw)/median_raw

    # To make sure they are in the same order, we reindex the entries from the unadjusted
    med_dist_ptage_raw = med_dist_ptage_raw.reindex(med_dist_ptage_norm.index)

    # We now combine the calculated percentage movement with the to the same order of the
    # df to annotate the labels on the graph.
    # med_dist_delta = pd.DataFrame()
    # med_dist_delta = []
    med_dist_delta = list( f'{round( (med_dist_ptage_norm[row] - med_dist_ptage_raw[row])
```

```
    # We now combine the calculated percentage movement with the to the same order of the CoL adjusted me
    # df to annotate the labels on the graph.
    # med_dist_delta = pd.DataFrame()
    # med_dist_delta = []
    med_dist_delta = list( f'{round( (med_dist_ptage_norm[row] - med_dist_ptage_raw[row]) *100 , 2 ): 06,

    #This section of code is the Check1, Check2, Check3, Check4 content
    # Checking whether a city started above/below the raw median and moved above/below the normalized med
    # These df lists will be passed to the functions to help format the axis and labels of the graphs
    city_hi2hi = (med_raw_city_above * med_norm_city_above) == True

    city_hi2hi = city_hi2hi.loc[city_hi2hi==True].index

    city_low2hi = (med_raw_city_above == False) * (med_norm_city_above == True) == True
    city_low2hi = city_low2hi.loc[city_low2hi==True].index

    city_hi2low = (med_raw_city_above == True) * (med_norm_city_above == False) == True
    city_hi2low = city_hi2low.loc[city_hi2low==True].index

    city_low2low = (med_raw_city_above == False) * (med_norm_city_above == False) == True
    city_low2low = city_low2low.loc[city_low2low==True].index

    # highlight_check = highlight_check.loc[highlight_check==True].index            # and moved to above

    # Find the order.  We are interested in graphing results in descending order.
    # We need only the list of names, and can retrieve it from order_list_norm, which contains both city
    # For the_order, we only need the name of the city.
    the_order = order_list_norm.index
    # print(highlight_check)

    print( 'Processing...' )

    normalized_pay(statecount, the_order, income_norm, SOC_group, cat_num, median_norm,
                   city_hi2hi, city_low2hi, city_hi2low, city_low2low, med_dist_delta, ptile25, ptile75)

    raw_pay(statecount, the_order, income_raw, SOC_group, cat_num, median_raw,
            city_hi2hi, city_low2hi, city_hi2low, city_low2low, med_dist_delta, ptile25, ptile75)
```

Helper Functions:

normalized_pay(df, the_order, income_cat, SOC_group, cat_num, the_median, c1, c2, c3, c4, label_annot, ptile25, ptile75)

raw_pay(df, the_order, income_cat, SOC_group, cat_num, the_median, c1, c2, c3, c4, label_annot, ptile25, ptile75)

- **Uses modified seaborn boxplot to visualize data:**
- **Individual graphs to see normalized (real) pay vs raw (nominal) pay**
- **At a glance, conveys:**
  - Concentration of middle 50% of results (IQR) of each grouping
  - Sets a common plot size for each graph
  - Formats graph titles and subtitles
  - General settings for the labels

Helper Functions:

normalized_pay(df, the_order, income_cat, SOC_group, cat_num, the_median, c1, c2, c3, c4, label_annot, ptile25, ptile75)

raw_pay(df, the_order, income_cat, SOC_group, cat_num, the_median, c1, c2, c3, c4, label_annot, ptile25, ptile75)

```python
    income_norm = income_cat
    job_class_num = SOC_group
    cat_name = cat[cat_num]

    statecount = df

    median_norm = the_median

    # Find the order
    my_order = the_order

    fig, ax = plt.subplots(figsize=(25, statecount[cat_name].nunique()*.3))

    # checking which color scheme to use for the boxplot bars.  We use the default for any category
    # except for cat_num = 2 (for States) or cat_num=3 (for Cities)
    if (cat_num == 2) or (cat_num == 3):
        the_colors = color_palette(my_order, c1, c2, c3, c4)
    else:
        the_colors = 'Pastel2'

    subdata_norm = sns.boxplot(data = statecount,
                               y = cat_name,
                               x = income_norm,
                               whis = 0,
                               showfliers = False,
                               width = 0.8,
                               ax = ax,
                               order = my_order,
                               palette = the_colors)


    # Graph Formatting Settings

    plt.xlim( ptile25 * .9 , ptile75 * 1.1 )
    subdata_norm.axvline(median_norm, color = 'red')

    # additional formatting of the labels and axis
    subdata_norm = label_formatter(subdata_norm, cat_num, c1, c2, c3, c4, label_annot)

    # Formatting title and labels at head of graph.
    plt.title(f'Cost-of-Living Adjusted (Real) Annual Salary\n{job_type[job_class_num]}\n', loc = 'center', fonts
    plt.title(f'Adjusted Median Salary: ${round(median_norm):,}', loc = 'left', fontsize = 18, color = 'red')
    plt.title(f'Each bar shows: [ <-- (   25-75 %tile   ) --> ]', loc = 'right', fontsize = 18, color = 'red')
```

```python
    income_raw = income_cat  # income_cat is the category of income (ie normalized or raw income) from which w
    job_class_num = SOC_group
    cat_name = cat[cat_num]


    statecount = df

    median_raw = the_median

    # Find the order
    my_order = the_order #statecount.groupby(by=[cat_name])[income_raw].median().sort_values(ascending=False).

    fig, ax = plt.subplots(figsize=(25, statecount[cat_name].nunique()*.3))

    # checking which color scheme to use for the boxplot bars.  We use the default for any category
    # except for cat_num = 2 (for States) or cat_num=3 (for Cities)
    if (cat_num == 2) or (cat_num == 3):
        the_colors = color_palette(my_order, c1, c2, c3, c4)
    else:
        the_colors = 'Pastel2'

    subdata_raw = sns.boxplot(data = statecount,
                              y = cat_name,
                              x = income_raw,
                              whis = 0,
                              showfliers = False,
                              width = .8,
                              ax = ax,
                              order = my_order,
                              palette = the_colors)

    # Graph Formatting Settings
    # Here we use income_norm so that both the norm and raw graphs are of the same scale

    plt.xlim( ptile25 * .9 , ptile75 * 1.1)
    subdata_raw.axvline(median_raw, color='red')

    # additional formatting of the labels and axi
    subdata_raw = label_formatter(subdata_raw, cat_num, c1, c2, c3, c4, label_annot)


    # Formatting title and labels at head of graph.
    plt.title(f'Non-Adjusted (Nominal) Annual Salary\n{job_type[job_class_num]}\n', loc='center', fontsize = 2
    plt.title(f'Non-Adjusted Median Salary: ${round(median_raw):,}', loc='left', fontsize = 18, color = 'red')
    plt.title(f'Each bar shows: [ <-- (   25-75 %tile   ) --> ]', loc='right', fontsize = 18, color = 'red')
```

Caption: normalized_pay() on left, and raw_pay() on right. Code is similar, but not same.

More Helper Functions:

color_palette(label_order, check_1, check_2, check_3, check_4)

label_formatter(plotdata, cat_num, check_1, check_2, check_3, check_4, label_annot)

- **Setup of bar colors to enhance visualization**
- **Customizing and color coding the labels to display information density and to enhance ease of understanding**
- **Various chart enhancements to facilitate understanding**
- **At a glance, conveys:**
  - Position of relative direction of movement of purchasing power
  - Comparison to median of the grouping results
  - Ranking in order of purchasing power
  - Facilitates visual reference on how CA cities fare against other city results in a given grouping

More Helper Functions:

color_palette(label_order, check_1, check_2, check_3, check_4)

label_formatter(plotdata, cat_num, check_1, check_2, check_3, check_4, label_annot)

```
    props = dict(boxstyle='round', facecolor='white', alpha=0.5)
    ax.text(.79, .9, textstr, ha = 'left', transform=ax.transAxes, fontsize=14, vertical

    # adding additional annotations
    labels = [item.get_text() for item in ax.get_yticklabels()]

    # add relative percentage movement of median from before adjustment to after CoL ad
    for i in range(len(ax.get_yticklabels())):
        labels[i] = labels[i] + " | " + label_annot[i]

    # Highlight label text if CA or CA city is an entry. If true, se
    if cat_num == 2:  # category number 2 is looking names of States

        state_names = []
        state_names = dataset[cat[2]].unique()
        # labels = [item.get_text() for item in ax.get_yticklabels()

        # Checking for CA in the labels.  If found, mark with *** to
        for i in range(len(ax.get_yticklabels())):
            if 'CA' in str(ax.get_yticklabels()[i]):  labels[i] = '*

    if cat_num == 3:    # category number of 3 is looking for names
        # Compile list of CA cities
        CA_cities = []
        CA_cities = dataset.loc[dataset[ str(cat[2]) ].isin(['CA'])]

        # Storing the labels in case we need to format or change pro
        # labels = [item.get_text() for item in ax.get_yticklabels()

        for i in range(len(ax.get_yticklabels())):     # Check to see if CA city name a
            for j in range(len(CA_cities)):
                if CA_cities[j] in str(ax.get_yticklabels()[i]):
                    labels[i] = '*** ' + labels[i]

    #re-inserting the labels with the changes
    ax.set_yticklabels(labels)

return ax
```

```
    length = len(label_order)

    # initialize list variable to hold the colors.  This will be used to
    bar_colors = [None] * length

    # Check to see if a state or city_name name appears, if so highlight.
    # We are highlighting cities if they are above/below median in raw in
    # We then set the bar color the same as the label color.
    for i in range(length):
        for a in range(len(check_1)):
            if check_1[a] == label_order[i] : bar_colors[i] = '#00bfff'
        for b in range(len(check_2)):
            if check_2[b] == label_order[i]: bar_colors[i] = 'lightgreen'
        for c in range(len(check_3)):
            if check_3[c] == label_order[i]: bar_colors[i] = 'lightcoral'
        for d in range(len(check_4)):
            if check_4[d] == label_order[i]: bar_colors[i] = 'silver'

    return bar_colors
```

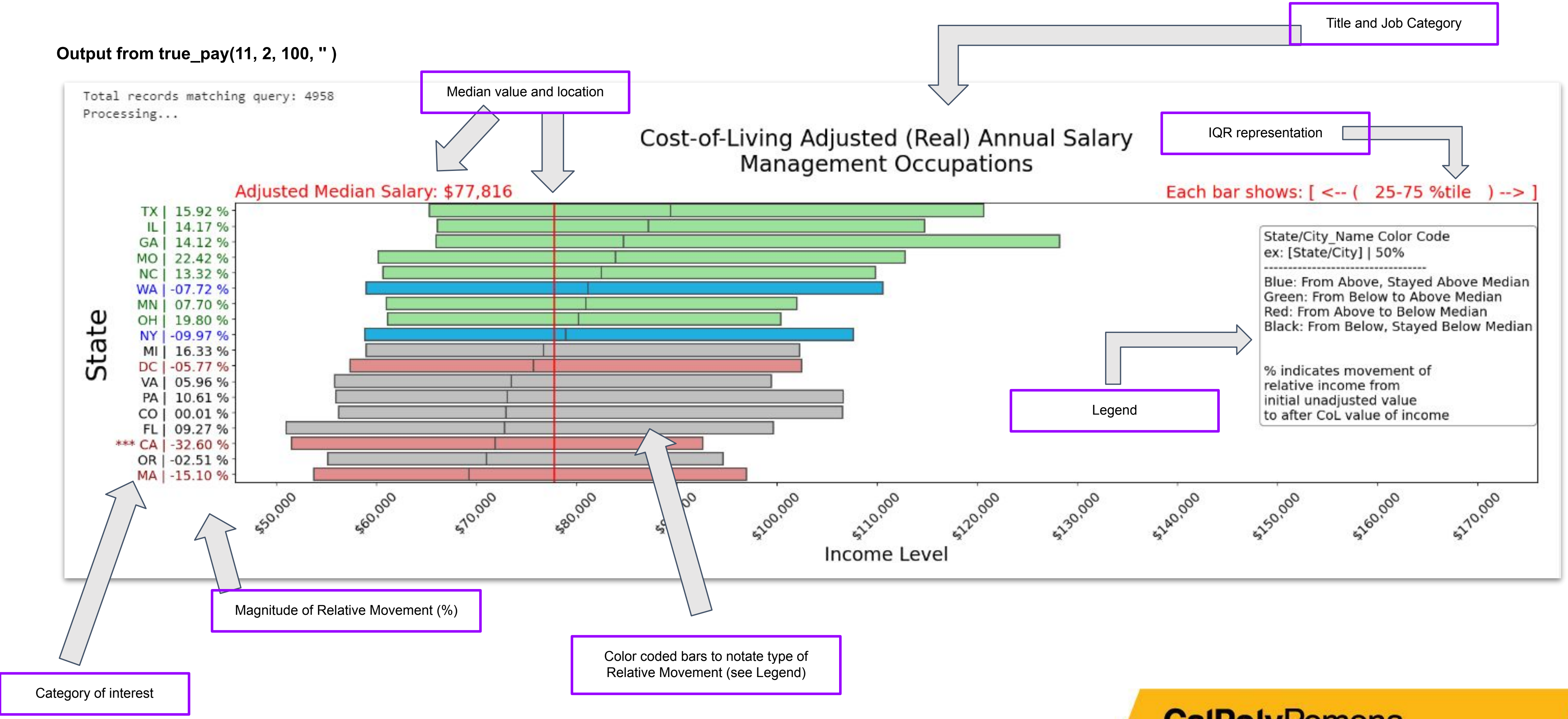
```
    ax = plotdata

    ax.set_xlabel('Income Level', fontsize=20)
    ax.set_ylabel(cat[cat_num], fontsize=30)

    ax.tick_params(axis='x', labelsize=14, rotation=45)
                  xis='y', labelsize=14)
                  or_formatter(ticker.FuncFormatter(lambda x, pos: '$' + '{:,.0f}'.format(x)))
                  or_locator(ticker.MultipleLocator(10000))

                  ) or (cat_num == 3):

                   list variable to hold the colors.  This will be used to assign a matching color to the bar to
                   []

                  ee if a state or city_name name appears, if so highlight.
                  hlighting cities if they are above/below median in raw income and where they move above/below m
                  t the bar color the same as the label color.
                  ge(len(ax.get_yticklabels())):
                   range(len(check_1)):
                  "'{check_1[a]}'" in str(ax.get_yticklabels()[i]): ax.get_yticklabels()[i].set_color('blue')
                   range(len(check_2)):
                  "'{check_2[b]}'" in str(ax.get_yticklabels()[i]): ax.get_yticklabels()[i].set_color('darkgreen'
                   range(len(check_3)):
                  "'{check_3[c]}'" in str(ax.get_yticklabels()[i]): ax.get_yticklabels()[i].set_color('darkred')
                   range(len(check_4)):
                  "'{check_4[d]}'" in str(ax.get_yticklabels()[i]): ax.get_yticklabels()[i].set_color('black')

        legend_label = ['Blue: Stayed Above Median', 'Green: From Below to Above Median', 'Red: From Above to Below
        textstr = '\n'.join((
                        # '   Bars Represents    ',
                        # '| <- 25-50 %tile -> |',
```

Center, Left, and Right: samples of code from helper functions.

Visuals:

How to read the graph - Adjusted values (real)

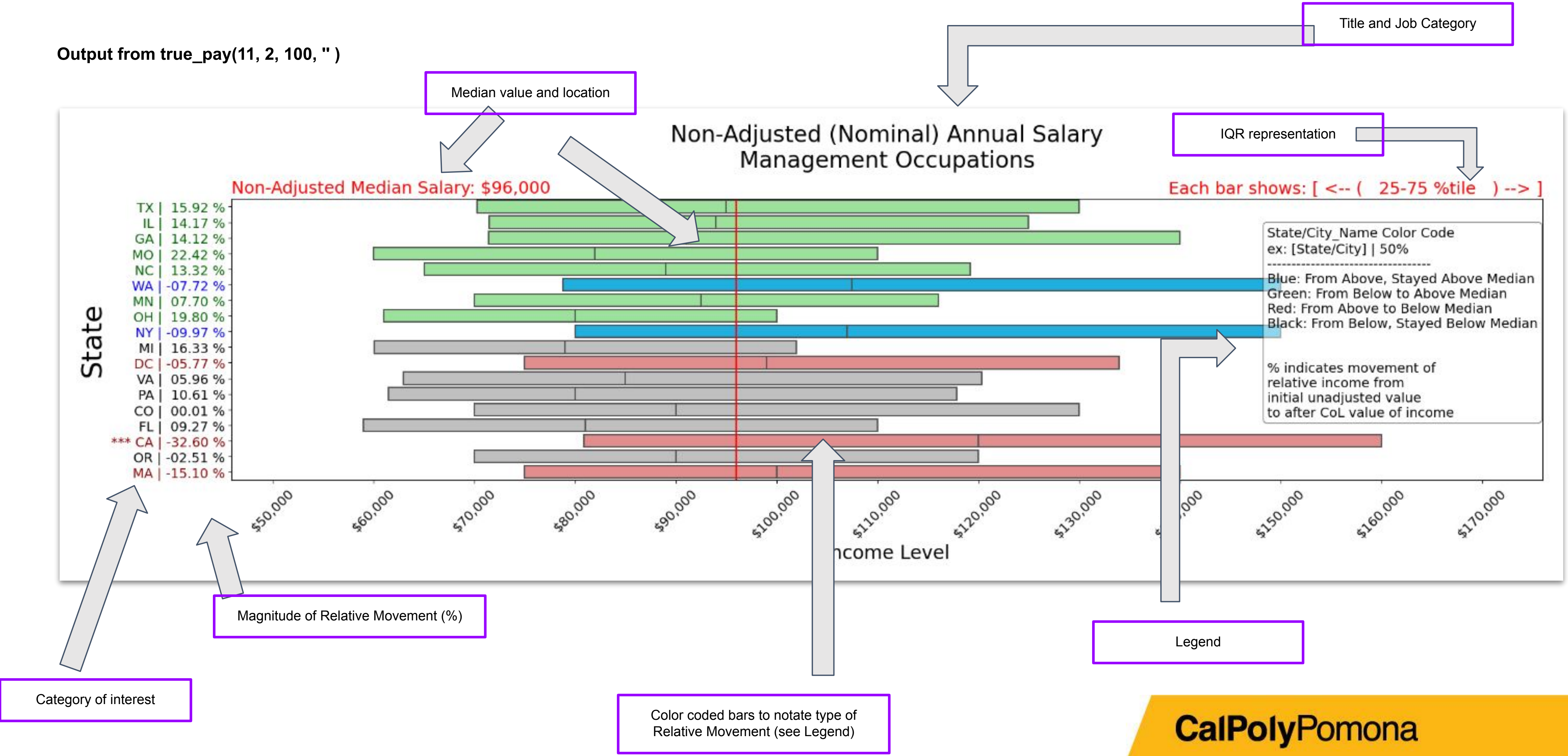
Output from true_pay(11, 2, 100, ")



Visuals:

How to read the graph - Raw values (nominal)

Output from true_pay(11, 2, 100, '')

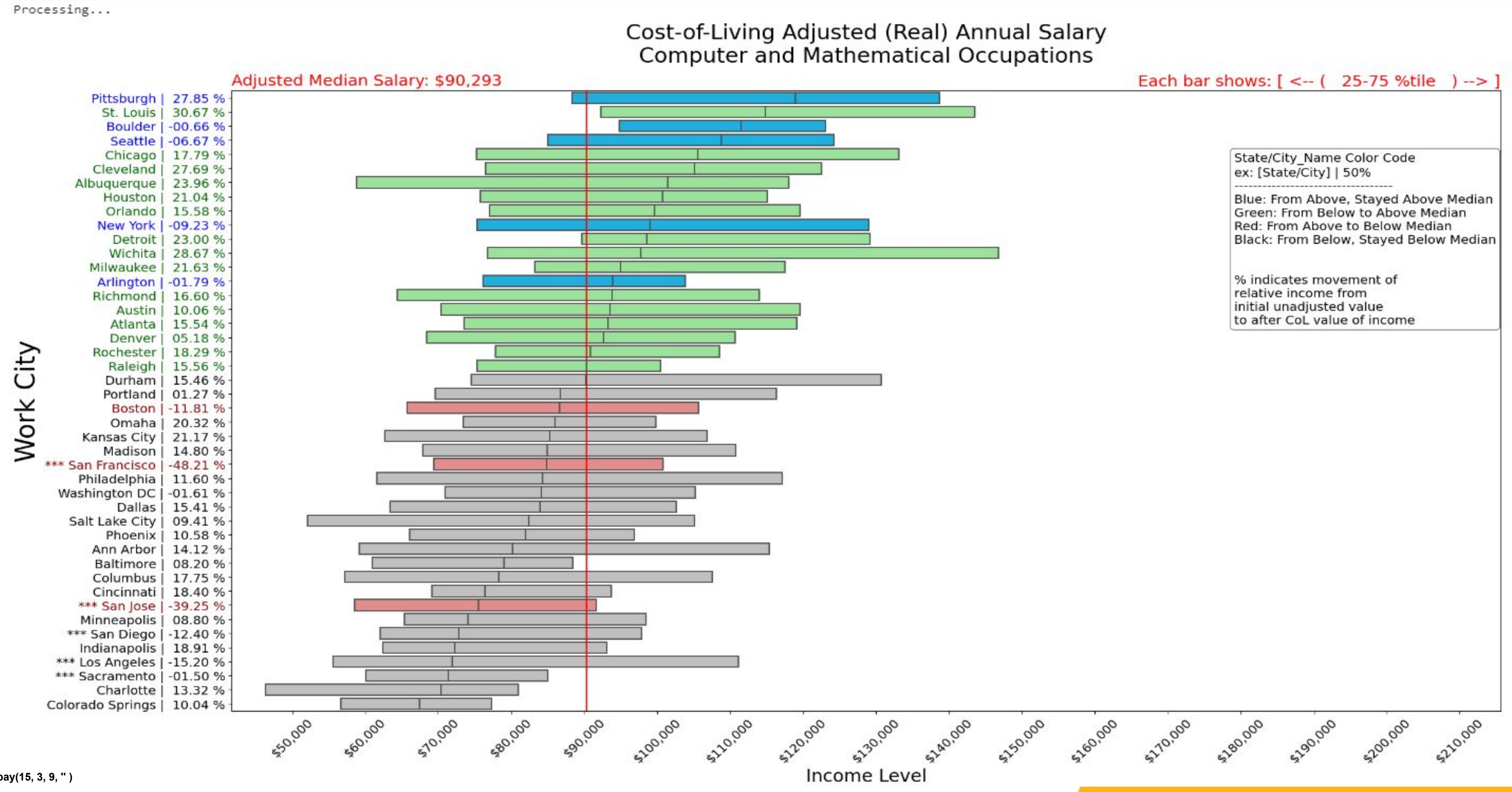


Results/Findings

Problem Statement

For a given job role,

in which city does your income provide the most purchasing power?

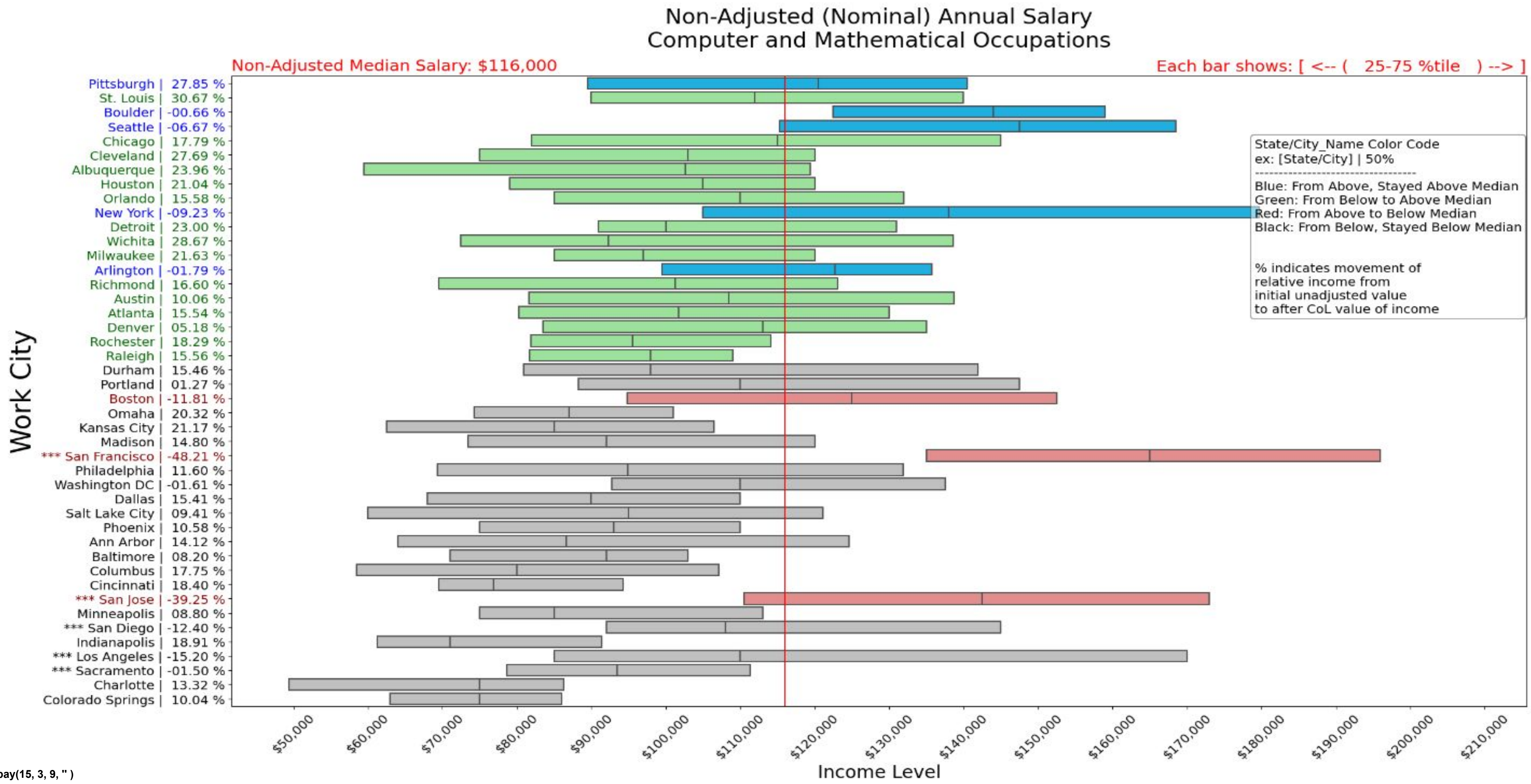


Output from true_pay(15, 3, 9, '')

Problem Statement

For a given job role,

in which city does your income provide the most purchasing power?



Output from true_pay(15, 3, 9, '')

Interpretation:

<u>After adjustment for Cost of Living (CoL):</u>

- **Green and Blue cities are desirable**
- **Individual cities with median values with greater distance to the right of the median of all cities in the search criteria are preferable.**
- **Cities with a large positive % magnitude movement is a plus**
- **For this class of job role, cities with the above positive factors could afford you better purchasing power, despite some cities where the nominal income is not as attractive.**
- **Red cities have nominal incomes that may have been attractive, but has purchasing power that is reduced by CoL**

<u>Interesting finds:</u>

- **Boulder, CO has the most stable Real income. The magnitude of movement is almost 0, suggesting pay levels that balance extremely nicely with Cost of Living (- 0.66%)**
- **Seattle, WA is the only large city with a tech reputation with a very low magnitude of movement, also suggesting pay levels that balance nicely with Cost of Living. (- 6.67%)**
- **St. Louis, MO and Pittsburgh, PA could be the sleepers in disguise, seemingly average pay level cities, but have very large positive movements (27.85% and 30.67% respectively). St. Louis could possibly represent better bang-for-the-buck value as it is a green city and ranked #2 overall.**

Interpretation:

<u>Limitations and Caveats</u>

- **Our conclusions are meant to be a guide.**
- **The *Ask a Manager Survey* was distributed online. The type of respondent could naturally be those who tend to be online, whether in their personal lives, or for their line of work.**
- **Could explain why Management Occupations, Business and Finance, and Computer & Math are over-represented, with Educational Instruction & Library, Arts & Entertainment, and Office & Admin Support are in a distant second.**

- **Pay scale band ranges widely in each city, and the output displays only the middle 50% of all responses. You could be at the high end of a lower pay scale city, and still be fulfilled.**
- **Could also be at the low end of a higher income pay city, and come away feeling unfulfilled.**
- **Due diligence and research is still needed before committing to a potential job offer in a target city.**

- **Pay scale is not the only context or factor in deciding which city is desirable.**
- **Value of intangible things like glamour, culture, environment, climate, lifestyle, political views are just some of the other factors when deciding what is best.**

Future Work

Future Work:

- **Our data source, *Ask a Manager Survey,* used for this project was publicly available. However, the limitation could be the type of reach and the type of respondents that would engage in this type of data collection.**

- **A larger, more reliable dataset that could better paint the picture to the state of employment pay scale could be large private companies or large entities that have access to payroll information. Companies such as ADP, Paychex, Paylocity, and other large market share payroll companies could be in the best position to have this data.**

- ***AdvisorSmith Cost of Living Index* is a publicly available data set. AdvisorSmith is a private company and conducts market research to be used in developing their own business, consulting, and insurance products.**

- **While AdvisorSmith uses reliable sources in building their index, ACCRA is perhaps the gold standard and most comprehensive in Cost of Living studies, done by Council for Community and Economic Research (C2ER). However, it is very expensive to access this data. Our work could be more accurate if funding was provided for additional research.**

Bonus Round

(Live Demo)

Search for Specific Titles

Software Developer: true_pay(15, 3, 1, ‘sof dev’)

Software Engineer: true_pay(15, 3, 1, ‘sof eng’), true_pay(15, 1, 1, ‘sof eng’)

Data Analyst: true_pay(15, 3, 1, ‘dat ana’), true_pay(15, 1, 1, ‘dat ana’)

Data Scientist: true_pay(15, 9, 0, ‘dat sci’), true_pay(15, 1, 1, 'dat sci')

Professorial Roles: true_pay(25, 3, 1, ‘prof’)

Requests?

Thank You