Data Analysis: Homework One report

Taylor Woodard

August 1, 2023

1 Preface

Hello Dr Gutierrez and class. please feel free to scrutinize this preliminary report as see fit. This is my first report-style homework assignment and it has been over five years since I have written a paper of this style. All critisism is welcome.

Thank you!

–Taylor Woodard–

2 Introduciton

The goal of the intial report of the data was to sort a small subset of natality data presented in a flat flie. The flat file contained 1,800,103 records were presented using cp1252, a single-byte character encoding, where every integer in our string of numbers coresponded to a given data dictionary. The dictionary broke down the interpretation of the string of integers, such that the individual, or pair, of integers in a line of data represented a specific category of information. This dictionary could be used to sort our data into desired categories based on the need of information or specificity.

3 Methods

3.1 Sorting and Interpreting the Data

The Flat file is loaded into Python as a *.dat* file and sorted using a small sorting algorithm to catagorize the the data. Please note that python uses a zero-indexing where the first term in a string of numbers starts with zero, rather than one First, the data is narrowed down in the first iteration of the loop using *if x[11] == "1" and x[25:27] == "74" and x[52:54] == "55"* where *x[11]* pulls the data that satisfies our first condition condition, *1*, which represents residencey status.
Next we sort the data by terms that satisfies our second condition, *x[25:27]*, where all entries in our data equal *74*.
Next we sort the terms by *x[51:53] == "55" or x[53:55] =="55"* equals to *55*. This condition of our sorting represents the natility data of a instances only when the child was born alive. Please note, Python uses a slicing operation, which can be read *sequence[start_index:stop_index]* where *start_index* is the first term in our sequence which is to be included and *stop_index* is the first term that will be excluded in our index. Therefore when *x[25:27]* equals *74* our Boolean values are satisfied. In this case *x[25:27]* represents elements of our data that represents Residents of Texas.
Finally, the last chunk our loop sorts the data who's values satisfy *x[99:101]*, who's desired values are defined as *educounter_years*, which represents the number of years the individual spent in education, then sorted with respect to *educoutner_years* by terms satisfying *x[62]* which represents the birth order of the child. The overall break down can be interpreted as, for example, "Resident of the United States, Resident of Texas, x years the mother spent in school, birth order of x"

3.2 Graphical Representation and interpretation of our data

Because the data is sorted in such a way that should imply that there is a corelation between A mother's education level verus the Birth Order of a Child. The purpose of selecting these two conditions was to challenge the hypothesis that the less a mother was educated, the more likely that the Birth order of the child would be higher. In otherwords, a mother with less education was assumed to have a higher tendency to have more children when compared to a woman with more education.
Two types of graphs were used to display the data, given our sorting conditions: a Bar Chart and a Violin Plot.

The Bar Chart was intended to show the spread of the data across our specified catagories. Due to the fact that it can be reasonably assumed that women would be more inclined to have children after graduating highschool, 12 years of education,and therefore a catagory that shows an extreme spread of data compared to the other catagories, a second bar chart was created where 12 years of education was omitted from the graph to show a better spread of the data across the catagories. The Bar Chart is set up to show the most amount of variance between the classes of data in such way that a comparison of the data between different values of a mother's education can be easily visulized. Because we are sorting through multiple variables that constrain the data under multiple parameters the Bar Chat A Violin Plot was used to display categorical density of birth order per specified years of education. The wider portions of the graph represent birth order with a higher number of our elements falling into the category of Years of Education v Birth Order. The Black Bar represents the Interquartile range covering the middle 50% of our data. The goal of the Violin Plot was to try to represent the spread of our data in a clearer visual with the hopes that it would make the answer of our hypothesis more obvious. Please reference the code provided at the end of the document.

4 Question Responses

4.1 Question 1: Why is the date and time of birth no longer recorded

4.1.1 Initial thoughts

When looking at the flat file of data, one might assume that the reason why time and date are no longer recorded in birth records would be very clear due to the number of categories that are already being recorded, as well as the vast number of children being born every years the reason for the change could be assumed to be that the time and date were just superfluous bits of information about the birth of the child. With hundreds, if not thousands of children being born in a singlar hospital, let alone the entire country, recording the information specifically of the time of the birth would just add too many unneeded complexities when recording every birth in America.

4.1.2 Found answer

Although I was unable to find any information in regards to the revision of the data processing of 1988, information was found on the CDC's website: *(https://www.cdc.gov/nchs/nvss/revisions-of-the-us-standard-certificates-and-reports.htm)* which had detailed documents about the the reason for the changes in the year 2003. To quote the document, "the Working Group to Improve Data Quality had found a decline in vital statistics birth data quality associated with electronic registration of vital events." Which, given the state of technology at the time, as well as the rise of the era of electronic records, the revision can be assumed to have come the need for a lower level of complexity when due to the fact of the hefty $_{\text{hefty}_{\text{hefty}}}$ amout of variables that would be required for the category of recording time alone.

4.2 Question 2: Why was a flat file used?

A Flat file was used for the simpliticy and straightfoward use when the data is paired with a clear dictionary. Also, because flat files are present in strings of integers, sometimes containing delimiters such as commas, the required memory to store and send is dramatically reduced when comepared to something as trivial as an excel spreadsheet. Furthermore, flatfiles are a nearly universal means of containing data. Because the file is so straightfoward, different coding languages as well as different methods can be used to sort the data which would make sharing results of data sorting very simple. Fianlly, because of its fairly simple means of translation as well as typically requireing less memory, flat files can easily be used as a backup storage option.

4.3 Question 3: What problem occured when uploading the data into Microsoft Excel?

Due primarily to memory constraints, Microsoft Excel has a limit on the number of rows that can be entered into a spreadsheet, which is $1,048,576$ rows. And due to the fact that our given data set contained $1,800,103$, a good size portion of data was unavalible to view.

5 Data Questions

5.1 How many Live Births occured in Texas in 1969 from mothers residing in Texas?

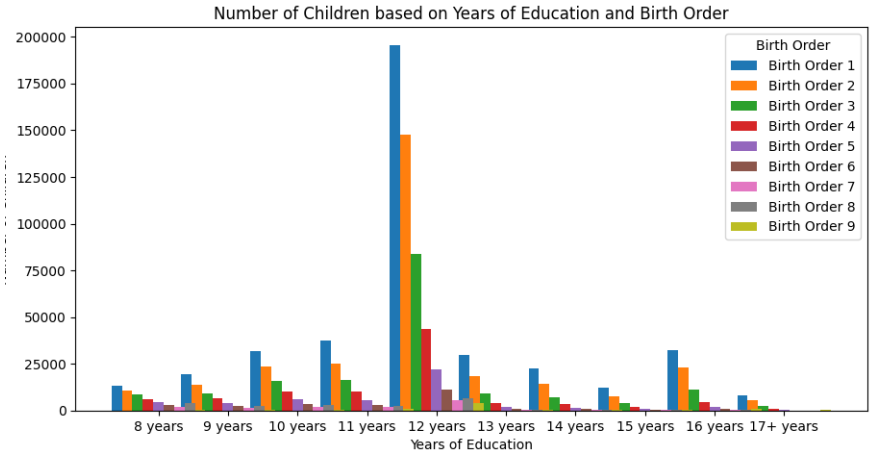
Accorting the my sorting algorithm, 53,854 live births occured in Texas in 1969.

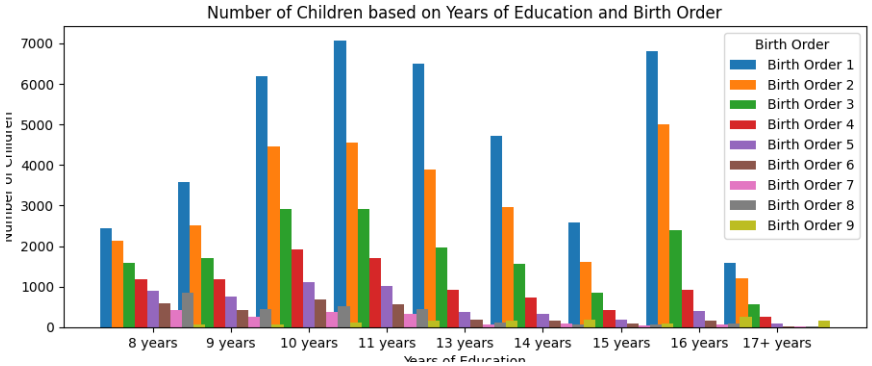
5.1.1 Bonus question: How would you visualize births from each state with respect to every other state?

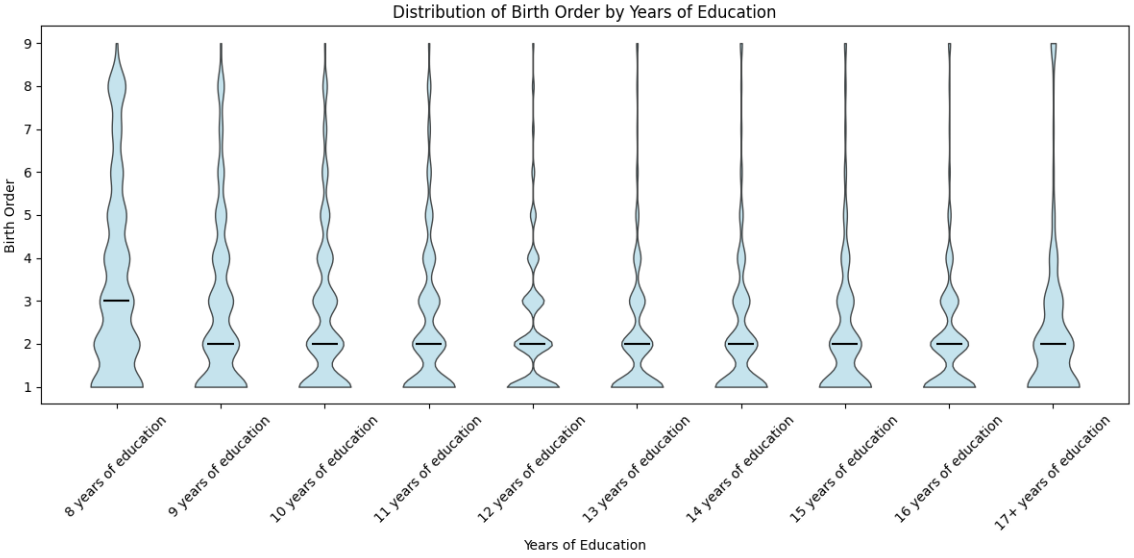
Hypothetically, if the data for all of the births in all of the states was sorted using the same method presented above, the data would have to be represented in 3 dimensions where our new axis represents all 50 states. If we wanted to add more conditions to our comparision, we would have to represent the data in n-dimensions where n is the number of dimensions that are chosen to evaluate the data.

5.2 Show graphically how the level of education of the mother is related to the birth order (1st born, second child, third, etc.)

Please note that do to a formatting a error, the y-axis for the barchart titles should read "Number of Children", the x-axis should read "Years of Education".







5.2.1 Bonus question: How would you visualize each variable with respect to every other variable?

Because there are so many veriables and so few ways to represent encapsulate chuncks of the data with a hefty $_{\text{hefty}_{\text{hefty}}}$ number of variables, the highest demension of visualization that we could achieve would be a 3D bubble chart which is equivalent to a fourth dimensional representataion, where our 3 axis (plural) represent the relation between three parameters or variables and the fourth dimensional represents categorical density.

6 Appendix

```
    import numpy as np
    import matplotlib.pyplot as plt

    data = []
    var1 = []
    counters = {
        "counter_main": 0,
        "count_r_tx": 0,
        "count_r_ntx": 0,
        "counter2": 0,
        "educounter14": 0,
        "educounter04": 0,
        "educounter05": 0,
        "educounter06": 0,
        "educounter07": 0,
        "educounter08": 0,
        "educounter09": 0,
        "educounter10": 0,
        "educounter11": 0,
        "educounter12": 0,
        "educounter13": 0,
        "birthorder1": 0,
        "birthorder2": 0,
        "birthorder3": 0,
        "birthorder4": 0,
        "birthorder5": 0,
        "birthorder6": 0,
        "birthorder7": 0,
        "birthorder8": 0,
        "birthorder9": 0,
        "alive": 0
    }


    file_path = r"E:\DataAnalysisSum23\DataProject\US1969.dat"
    data1969 = file_path
    with open(file_path, "r", encoding="cp1252") as f:
        for x in f:
            data.append(x.strip())


    educounter_years = ["04", "05", "06", "07", "08", "09", "10", "11", "12", "13"]
    birthorder_terms = [str(i) for i in range(1, 10)]

    educounter_counter = {category: 0 for category in educounter_years}
    birthorder_counters = {year: {str(i): 0 for i in range(1, 10)} for year in educounte

    for x in data:
        if x[11] == "1" and x[25:27] == "74" :
            continue
        if x[51:53] == "55" or x[53:55] == "55":
            counters["alive"] +=1
```

```
        education_level = x[99:101]
        birth_order = x[62]
        if education_level in educounter_years:
            educounter_counter[education_level] += 1
            if x[51:53] == "55":
              counters["alive"] +=1
            if birth_order in birthorder_terms:
                birthorder_counters[education_level][birth_order] += 1

    print("Born Alive", counters["alive"])


    #begin graphical set up
    education_level_labels = {
        "04": "8 years",
        "05": "9 years",
        "06": "10 years",
        "07": "11 years",
        "08": "12 years",
        "09": "13 years",
        "10": "14 years",
        "11": "15 years",
        "12": "16 years",
        "13": "17+ years",
    }
    x_axis_labels = [education_level_labels[year] for year in educounter_years]

    #BARCHART W YEAR 12
    # Create grouped bar chart
    plt.figure(figsize=(12, 6))

    # Data preparation and plotting for each birth order within each education year
    num_educounter_years = len(educounter_years)
    bar_width = 0.15
    bar_positions = np.arange(num_educounter_years)

    for i, birth_order in enumerate(birthorder_terms):
        counts = [birthorder_counters[year][birth_order] for year in educounter_years]
        plt.bar(bar_positions + i * bar_width, counts, width=bar_width, label=f"Birth Or

    plt.xlabel("Years of Education")
    plt.ylabel("Number of Children")
    plt.title("Number of Children based on Years of Education and Birth Order")
    plt.tight_layout()
    plt.xticks(bar_positions + (bar_width * (len(birthorder_terms) - 1)) / 2, x_axis_labe
    plt.legend(title="Birth Order")
    plt.show()
    #BarchartWith12 End

    #BARCHART W/OUT 12 YEARS
    #because the two charts were so similar, it was necesarry to repeat this initilizati
    educounter_years = ["04", "05", "06", "07", "09", "10", "11", "12", "13"]
    birthorder_terms = [str(i) for i in range(1, 10)]

    educounter_counter = {category: 0 for category in educounter_years}
    birthorder_counters = {year: {str(i): 0 for i in range(1, 10)} for year in educounte

    for x in data:
        if x[11] == "1" or x[25:27] == "74":
            continue
        education_level = x[99:101]
        birth_order = x[62]
```

```
        if education_level in educounter_years:
            educounter_counter[education_level] += 1
            if birth_order in birthorder_terms:
                birthorder_counters[education_level][birth_order] += 1

    education_level_labels = {
        "04": "8 years",
        "05": "9 years",
        "06": "10 years",
        "07": "11 years",
        "08": "12 years",
        "09": "13 years",
        "10": "14 years",
        "11": "15 years",
        "12": "16 years",
        "13": "17+ years",
    }
    x_axis_labels = [education_level_labels[year] for year in educounter_years]
    # Create grouped bar chart
    plt.figure(figsize=(12, 6))

    # Data preparation and plotting for each birth order within each education year
    num_educounter_years = len(educounter_years)
    bar_width = 0.15
    bar_positions = np.arange(num_educounter_years)

    for i, birth_order in enumerate(birthorder_terms):
        counts = [birthorder_counters[year][birth_order] for year in educounter_years]
        plt.bar(bar_positions + i * bar_width, counts, width=bar_width, label=f"Birth Or

    plt.xlabel("Years of Education")
    plt.ylabel("Number of Children")
    plt.title("Education v Birth Order with 12 years")
    plt.tight_layout()
    plt.xticks(bar_positions + (bar_width * (len(birthorder_terms) - 1)) / 2, x_axis_labe
    plt.legend(title="Birth Order")
    plt.show()
    #BarChartWithout12 End

    #VIOLIN DATA PREP Start
    #Needed to reinclude the initial data loop in order for this to work, not quite sure
    educounter_years = ["04", "05", "06", "07", "08", "09", "10", "11", "12", "13"]
    birthorder_terms = [str(i) for i in range(1, 10)]

    educounter_counter = {category: 0 for category in educounter_years}
    birthorder_counters = {year: {str(i): 0 for i in range(1, 10)} for year in educounte

    for x in data:
        if x[11] == "1" or x[25:27] == "74":
            continue
        education_level = x[99:101]
        birth_order = x[62]
        if education_level in educounter_years:
            educounter_counter[education_level] += 1
            if birth_order in birthorder_terms:
                birthorder_counters[education_level][birth_order] += 1

    # Prepare the data for the violin plot
    data_for_violin = []
    for year in educounter_years:
        for birth_order in birthorder_terms:
            for _ in range(birthorder_counters[year][birth_order]):
```

```
                data_for_violin.append((year, int(birth_order)))

    # Convert the data to separate arrays for each education level
    education_levels = educounter_years
    birth_orders = [int(order) for order in birthorder_terms]
    data_arrays = {year: np.array([order for (y, order) in data_for_violin if y == year])

    #ALL OF THE VIOLINS
    # Create the violin plot
    plt.figure(figsize=(12, 6))

    violin_parts = plt.violinplot(
        [data_arrays[year] for year in education_levels],
        positions=np.arange(len(education_levels)),
        widths=0.5,
        showmedians=True,
        showextrema=False,
    )

    plt.xlabel("Years of Education")
    plt.ylabel("Birth Order")
    plt.title("Distribution of Birth Order by Years of Education")
    #ViolinplotALL END

    # Set x-axis labels
    education_level_labels = {
        "04": "8 years of education",
        "05": "9 years of education",
        "06": "10 years of education",
        "07": "11 years of education",
        "08": "12 years of education",
        "09": "13 years of education",
        "10": "14 years of education",
        "11": "15 years of education",
        "12": "16 years of education",
        "13": "17+ years of education",
    }
    plt.xticks(np.arange(len(education_levels)), [education_level_labels[year] for year i


    ## INDIVIDUAL VIOLINS START
    # Customize the violins
    for pc in violin_parts['bodies']:
        pc.set_facecolor('lightblue')
        pc.set_edgecolor('black')
        pc.set_alpha(0.7)

    for partname in ('cmedians',):
        part = violin_parts[partname]
        part.set_edgecolor('black')

    plt.tight_layout()
    plt.show()

    for i, year in enumerate(education_levels):
        fig, ax = plt.subplots(figsize=(6, 6))
        violin_parts = ax.violinplot(
            data_arrays[year],
            positions=[i],
            widths=0.5,
            showmedians=True,
            showextrema=False,
```

```
        )

        ax.set_ylabel("Birth Order")
        # ax.set_title(f"Distribution of Birth Order by dictionary code: {year}")

        # Customize the violins
        for pc in violin_parts['bodies']:
            pc.set_facecolor('lightblue')
            pc.set_edgecolor('black')
            pc.set_alpha(0.7)

        for partname in ('cmedians',):
            part = violin_parts[partname]
            part.set_edgecolor('black')

        # Set x-axis label
        ax.set_xticks([i])
        ax.set_xticklabels([education_level_labels[year]], rotation=45)

        plt.xlabel("Years of Education")
        plt.tight_layout()
        plt.show()
```