DSC 5101 – Computer Programming in DSAI

Part II - Working with Data
HO 06 - Exploratory Data Analysis

Opim Salim Sitompul

Department of Data Science and Artificial Intelligence
Universitas Sumatera Utara

Co-funded by the Erasmus+ Programme of the European Union

Outline I

Introduction

“My two friends that have given birth recently to their first babies, BOTH went almost 2 weeks overdue before going into labour or being induced.”
“My first one came 2 weeks late and now I think the second one is going to come out two weeks early!!”
“I don’t think that can be true because my sister was my mother’s first and she was early, as with many of my cousins.”
(source: [4], p. 10).

Introduction

- The given reports are called **anecdotal evidence** because they are based on data that is unpublished and usually personal.
- What is needed is more persuasive evidence and more reliable answers.

Introduction

- **Anecdotal evidence** fails, because of:
  - Small number of observations: we might have to compare a large number of observation to be sure that a difference exists ([4], p. 10-11).
  - Selection bias: process of selecting data would bias the results.
  - Confirmation bias: either contribute examples to confirm or provides counter examples
  - Inaccuracy: misremembered, misrepresented, repeated inaccuracies, etc.

Statistical Approach

- Use statistical tools:
  - Data collection: data from large national survey designed explicitly to generate statistically valid inferences.
  - Descriptive statistics: generate statistics that summarize the data concisely, and evaluate different ways to visualize data.
  - Exploratory data analysis: look for interesting patterns, differences, and other features, check for inconsistencies and identify limitations.
  - Estimation: use data from a sample to estimate characteristics of the general population.
  - Hypothesis testing: evaluate whether apparent effects happen by chance.

Statistical Approach

- Two types of study:
  - Cross-sectional: captures a snapshot of a group at a point in time.
  - Longitudinal: observes a group repeatedly over a period of time.

Data Collection

- Predicting the age of abalone from physical measurements.
- Importing *abalone* (csv data) using Pandas:

```
import pandas as pd

df = pd.read_csv("https://archive.ics.uci.edu/
    ml/machine-learning-databases/abalone/
    abalone.data", header=None, names=['Sex', '
    Length', 'Diameter', 'Height', 'Whole
    weight', 'Shucked weight', 'Viscera weight'
    , 'Shell weight', 'Rings'])
print(df.head())
```

Data Collection

- See how many rows and columns:

```
rows = df.shape[0]
cols = df.shape[1]
print("rows = ", rows)
print("columns = ", cols)
```

- Output:
  rows = 4177
  columns = 9

Data Collection

- See columns:

```
print(df.columns) →
Index(['Sex', 'Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight', 'Viscera weight', 'Shell weight', 'Rings'], dtype='object')
sex_col = df['Sex']
print(type(sex_col)) →
<class 'pandas.core.series.Series'>
print(sex_col) →
0    M
1    M
..
4176    M
Name: Sex, Length: 4177, dtype: object
```

Variables

- There are 9 (nine) variables in dataset, shown as columns.
- 8 variables are called independent variables:
  'Sex': nominal / – / M, F, and I (infant)
  'Length': continuous / mm / Longest shell measurement
  'Diameter': continuous / mm / perpendicular to lengt
  'Height': continuous / mm / with meat in shell
  'Whole weight': continuous / grams / whole abalone
  'Shucked weight': continuous / grams / weight of meat
  'Viscera weight': continuous / grams / gut weight (after bleeding)
  'Shell weight': continuous / grams / after being dried
- Another variable, called 'Rings' is a prediction value, either as a continuous value or as a classification problem. integer / – / +1.5 gives the age in years

Transformation

- Data imported should be checked for errors, special values, convert to different format, and performed calculation, called **data cleaning** [4].
- The abalone dataset doesn't have missing attribute values.
- Example: Converting 'Sex' attribute into one-hot encoding, where 'M', 'F', and 'I' is encoded into '[1,0,0]', '[0,1,0]', and '[0,0,1]', respectively.

Transformation

```
"""
One-hot Encoding adapted from
'How to One Hot Encode Sequence Data in
    Python'
by Jason Brownlee
Filename: Abalone_data_encoding.ipynb]
"""
from numpy import argmax
import pandas as pd
```

Transformation

```
def int_encode(sex_col, alphabet):
    # define a mapping of chars to integers
    char_to_int = dict((c, i) for i, c in
        enumerate(alphabet))
    # integer encode input data
    i_encoded = [char_to_int[char] for char in
        sex_col]
    return i_encoded
```

Transformation

```python
def encode(i_encoded, alphabet):
  """one hot encoding"""
  onehot_encoded = list()
  for value in i_encoded:
    letter = [0 for _ in range(len(alphabet)
        )]
    letter[value] = 1
    onehot_encoded.append(letter)
  return(onehot_encoded)
```

Transformation

```python
def decode(onehot_encoded, alphabet):
  """invert encoding"""
  alphabet = 'MFI'
  int_to_char = dict((i, c) for i, c in
      enumerate(alphabet))
  inverted = list()
  for s in onehot_encoded:
    sex = int_to_char[argmax(s)]
    inverted.append(sex)
  return(inverted)
```

Transformation

```
def main():
    df = pd.read_csv("https://archive.ics.uci.
        edu/ml/machine-learning-databases/
        abalone/abalone.data", header=None,
        names=['Sex', 'Length', 'Diameter', '
        Height', 'Whole weight', 'Shucked
        weight', 'Viscera weight', 'Shell
        weight', 'Rings'])
    # define input to encode
    sex_col = df['Sex']
    # define universe of possible input values
    alphabet = 'MFI'
```

Transformation

```
  i_encoded = int_encode(sex_col, alphabet)
  encoding = encode(i_encoded, alphabet)
  enc = pd.Series(encoding).head()
  # replace 'Sex' column with one-hot encoding
  df['Sex'] = enc
  print(df.head())

if __name__ == "__main__":
  main()
```

Replacing Missing Values

- Missing values in a dataset which is displayed as '?' could be replaced by NumPy NaN (Not a Number) value.
- Example: hepatitis data from UCI Machine Learning Repository

Transformation

```
"""Filename: hepatitis_data.ipynb"""
import pandas as pd
import numpy as np

data = pd.read_csv("https://archive.ics.uci.
    edu/ml/machine-learning-databases/
    hepatitis/hepatitis.data", header=None,
names=['Class', 'AGE', 'SEX', 'STEROID', '
    ANTIVIRALS', 'FATIGUE', 'MALAISE', '
    ANOREXIA', 'LIVER BIG',   'LIVER FIRM', '
    SPLEEN PALPABLE', 'SPIDERS', 'ASCITES', '
    VARICES', 'BILIRUBIN', 'ALK PHOSPHATE',
      'SGOT', 'ALBUMIN', 'PROTIME', '
    HISTOLOGY'])
df = pd.DataFrame(data)
```

Transformation

```
"""Relace ? with NaN"""
def replace_missing(df):
  for i, j in df.iterrows():
    ct = 0
    for col in j:
      if(col == '?'):
        df.iloc[i,ct] = np.nan
      ct += 1
  return df

df_new = replace_missing(df)
df_new
```

Validation

- Validation is necessary if data is exported from another source and imported into another where errors might be introduced.
- Data could be validated using basic statistics and compared to published results if available.
- Example: summarizing variable in dataset:

```
df_new.columns[0] → 'Class'
df_new['Class'].value_counts().sort_index() →
1    32
2    123
Name: Class, dtype: int64
```

Validation

- Select axis=0 to count the values in each Column:

  df.count(0) $\rightarrow$

```
Class           155
AGE             155
SEX             155
                ...
VARICES         150
BILIRUBIN       149
ALK PHOSPHATE   126
SGOT            151
ALBUMIN         139
PROTIME         88
HISTOLOGY       155
dtype: int64
```

Validation

- Select axis=1 to count the values in each Rows, not counting NaN values:

  df.count(1) $\rightarrow$

```
0       19
1       19
2       19
3       19
4       18
..
150     19
151     19
152     19
153     20
154     20
Length: 155, dtype: int64
```

Interpretation

- To work with data effectively, there are two levels that should be thought at the same time: the level of statistics and the level of context. [4]
- The following code construct a default dictionary for age attribute:

```python
from collections import defaultdict
def MakeHepatitisMap(df):
  d = defaultdict(list)
  for index, caseid in df.AGE.iteritems():
    d[caseid].append(index)
  return d
```

Interpretation

- Rows containing age = 30 could be accessed:

```
d = MakeHepatitisMap(df)
x = d.get(30)
print(x) → [0, 9, 13, 54, 57, 82, 87, 96]
```

- Using this rows sex: 1 (M) and 2 (F) could be seen whether they are 1 (DIE) or 2 (LIVE):

```
s = {}
for i in x:
    dic = {df.SEX[i]: (df.Class[i])}
```

Interpretation

- Result obtained is:

```
print(dic) →
    {2: 2}
    {1: 2}
    {1: 2}
    {1: 2}
    {1: 2}
    {1: 2}
    {1: 1}
    {1: 2}
```

- Meaning that most of males at age of 30 are lived and only one is died.
- As for female there is only one female at age 30 and she is lived.

Distribution

- To describe a variable is performed by reporting the values that appear in the dataset and how many times each value appears.
- This description is called the distribution of the variable [4].
- Data that had been collected and verified needs to be compiled and displayed to highlights essential features.
- Only after properly presented, the statistical analysis can be performed properly [5].

Distribution

- There are three modes of presentation of data i.e. textual presentation, tabular presentation, and diagrammatic presentation.
- Bar Graph and Histogram are the two ways to display data in the form of a diagram [5].
- Histogram refers to a graphical representation, that displays data by way of bars to show the *frequency* of numerical data.
- Bar graph is a pictorial representation of data that uses bars to compare different *categories* of data.
- Histogram was first introduced by Karl Pearson (1857-1936).

Distribution

- Histogram is an accurate graphical representation of non-discrete variables (quantitative data) distribution.
- Bar graph indicates quantitative (categorical data).
- We can plot Age of hepatitis data in form of bar graph by referring to following data:

```
print("Age minimum: ", df['AGE'].min()) → 7
print("Age maximum: ", df['AGE'].max()) → 78
print("Age counts: ", df['AGE'].count()) → 155
```

Distribution

- Create dictionary of Age data:

```
"""Filename: hepatitis_distribution.ipynb"""
age_col = df['AGE']
age_data = {}
for x in age_col:
  age_data[x] = age_data.get(x, 0) + 1
```

- Get Age and each frequency into arrays:

```
ages = []
freq = []
for i in age_data:
  ages.append(i)
  freq.append(age_data.get(i))
```

Distribution

- Plot bar graph:

```
x = ages
y = freq
plt.bar(x,y,align='center') # A bar chart
plt.xlabel('Age (Years)')
plt.ylabel('Number of Patients')
for i in range(len(y)):
  plt.hlines(y[i],0,x[i], color="red",
      linestyle=':') # Here you are drawing
      the horizontal lines
plt.show()
```

Distribution

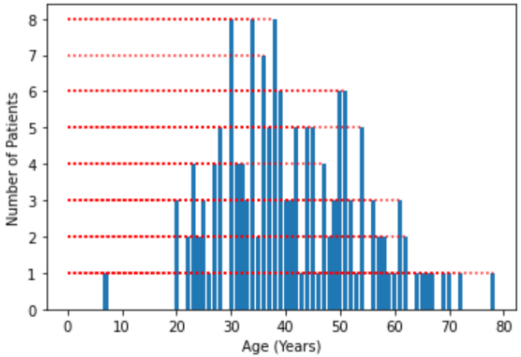

Figure 1: Bar graph of Age in hepatitis data

Distribution

- In order to construct an histogram:
  - *Bin* the range of values — that is, divide the entire range of values into a series of intervals.
  - The bins are usually specified as consecutive, non-overlapping intervals of a variable.
  - The bins (intervals) must be adjacent and are often (but are not required to be) of equal size.
  - Count how many values fall into each interval.
  - The bins are usually specified as consecutive, non-overlapping intervals of a variable.
  - The bins (intervals) must be adjacent and are often (but are not required to be) of equal size.

Distribution

- Plot Histogram using *pyplot.bar*:

```
np_hist = age_col
hist,bin_edges = np.histogram(np_hist)
bin_edges = np.round(bin_edges,0)
plt.figure(figsize=[10,8])
plt.bar(bin_edges[:-1], hist, width = 5,
    color='#0504aa',alpha=0.7)
plt.xlim(min(bin_edges), max(bin_edges))
plt.grid(axis='y', alpha=0.75)
plt.xlabel('Age',fontsize=15)
plt.ylabel('Frequency',fontsize=15)
```

Distribution

```
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.ylabel('Frequency',fontsize=15)
plt.title('Age Distribution Histogram',
    fontsize=15)
plt.show()
```

- Frequency of the histogram bar could be seen using:
  hist $\rightarrow$
  array([1, 3, 21, 29, 37, 21, 24, 11, 6, 2], dtype=int64)
- and the bin edges:
  bin_edges $\rightarrow$
  array([7., 14., 21., 28., 35., 42., 50., 57., 64., 71., 78.])

Distribution

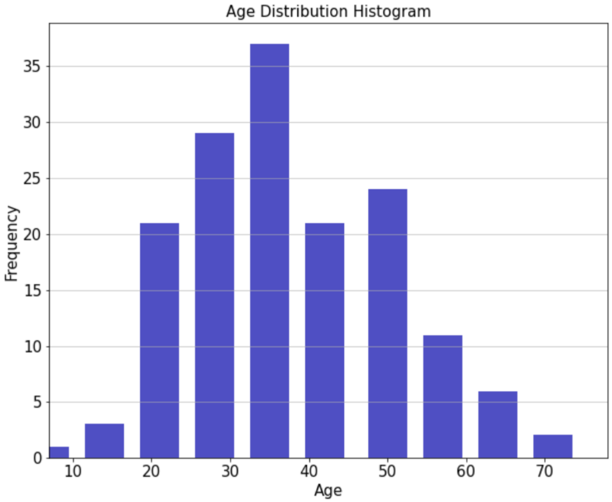

Figure 2: Histogram bar of Age in hepatitis data

Distribution

- Plotting Histogram using *pyplot.hist*:

```
plt.figure(figsize=[10,8])
n, bins, patches = plt.hist(x=np_hist, bins
    =8, color='#0504aa',alpha=0.7, rwidth
    =0.85)
plt.grid(axis='y', alpha=0.75)
plt.xlabel('Age',fontsize=15)
plt.ylabel('Frequency',fontsize=15)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.ylabel('Frequency',fontsize=15)
plt.title('Age Distribution Histogram',
    fontsize=15)
plt.show()
```

Distribution

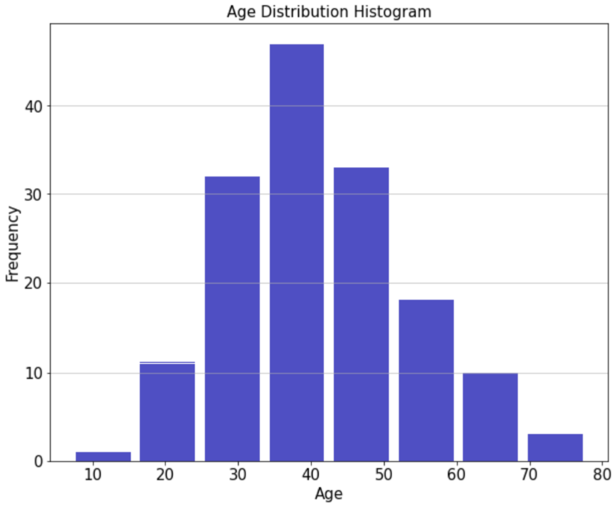

Figure 3: Histogram of Age in hepatitis data

Distribution

- Plotting Two Histogram.

```
plt.figure(figsize=[10,8])
n, bins, patches = plt.hist([m, f])
plt.grid(axis='y', alpha=0.75)
plt.xlabel('Age',fontsize=15)
plt.ylabel('Frequency',fontsize=15)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.ylabel('Frequency',fontsize=15)
plt.title('Age Distribution Histogram',
    fontsize=15)
labels= ["male","female"]
plt.legend(labels)
plt.show()
```

Distribution

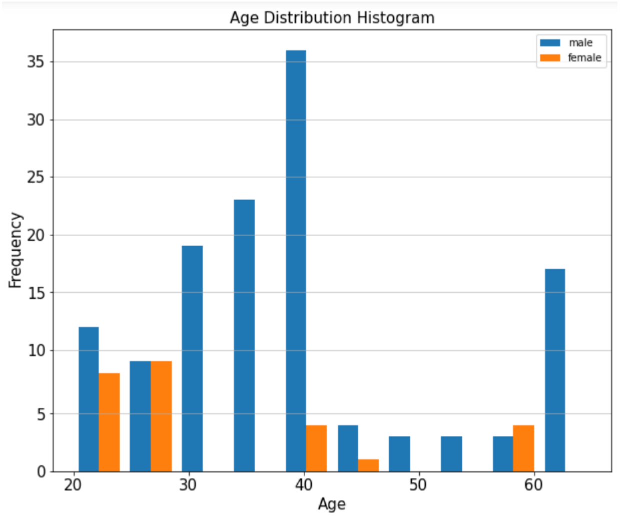

Figure 4: Histogram of Age in hepatitis data

Distribution

- Values of n, bins, and patches are:

```
    n →
array([[12., 9., 19., 23., 36., 4., 3., 3., 3., 17.],
        [ 8., 9., 0., 0., 4., 1., 0., 0., 4., 0.]])
    bins →
array([20. , 24.5, 29. , 33.5, 38. , 42.5, 47. , 51.5,
        56. , 60.5, 65. ])
    patches →
<a list of 2 BarContainer objects>
```

Outliers

- it is easy to identify the most common values and the shape of the distribution, but rare values are not always visible.
- This rare values are called outliers, which are extreme values that might be errors in measurement and recording, or might be accurate reports of rare events.
- First be sure to check for smallest and largest values, for the Age attributes:
  df['AGE'].min() $\rightarrow$ 7
  df['AGE'].max() $\rightarrow$ 78

Outliers

- A complete tally of age distribution could be listed as:

```
"""Filename: hepatitis_outlier.ipynb"""
age_col = df['AGE']
age_data = {}
for x in age_col:
  age_data[x] = age_data.get(x, 0) + 1
dict(list(age_data.items()))
```

```
{30: 8,   47: 4,   61: 3,   28: 5,   48: 2,
 50: 6,   38: 8,   62: 2,   36: 7,   54: 5,
 78: 1,   66: 1,   26: 1,   44: 5,   7: 1,
...
 39: 6,   25: 3,   52: 3,   67: 1,   53: 1,
 32: 4,   49: 3,   33: 3,   59: 1,   43: 1}
 41: 3,   58: 2,   56: 3,   60: 1,
```

Outliers

- From the list of ages, there are two values that are separated distantly from the most age values, namely the age of 7 and 78 at the two extremes.
- As a researcher, we have to make sure whether these two extremes ages are normal in terms of hepatitis case or not.
- The best way to handle outliers depends on domain knowledge; that is information about where the data come from and what they mean [5].

Summary Statistics

- If the details of the distribution are important, histogram gives a complete description of the distribution of a sample, where the values of the sample could be constructed.
- But often we want to summarize the distribution with a few descriptive statistics. [5]
  - Central tendency: Do the values tend to cluster around a particular point?
  - Modes: Is there more than one cluster?
  - Spread: How much variability is there in the values?
  - Tails: How quickly do the probabilities drop off as we move away from the modes?
  - Outliers: Are there extreme values far from the modes?
- Statistics designed to answer these questions are called **summary statistics.**

Mean

- The most common summary statistic is the **mean**, which is meant to describe the central tendency of the distribution.
- With a sample of n values, x_i, the mean, $\bar{x}$, is the sum of the values divided by the number of values:

$$\bar{x} = \frac{1}{n} \sum_i x_i \tag{1}$$

- Mean of age distribution:
  np.array(ages).mean() $\rightarrow$ 45.22448979591836 6

Variance

- Variance is a summary statistic intended to describe the variability or spread of a distribution.
- The variance of a set of values is:

$$S^2 = \frac{1}{n} \sum_i (x_i - \bar{x})^2 \tag{2}$$

- The term $(x_i - \bar{x})$ is called *deviation from the mean*.
- Variance is then the *mean squared deviation*, and the square root of variance, S, is called **standard deviation.**
- Variance and standard deviation of age distribution:
  np.array(ages).var() $\rightarrow$ 255.60266555601834
  np.array(ages).std() $\rightarrow$ 15.98757847693071 8

Effect Size

- An effect size is a summary statistic intended to describe the size of an effect.
- For example, to describe the difference between two groups, one obvious choice is the difference in the means.
- Another way to convey the size of the effect is to compare the difference between groups to the variability within groups.
- Cohen's **d** statistic intended to do is defined:

$$d = \frac{\bar{x}_1 - \bar{x}_2}{S} \tag{3}$$

Effect Size

- Let’s see the effect size between male and female ages.
- Function to extract age element from dictionary:

```python
def get_dict_element(a_dict):
  arr = []
  for x in range(len(a_dict)):
    arr.append(a_dict.get(x, 1))
  return arr
```

m = np.array(get_dict_element(m_age))
f = np.array(get_dict_element(f_age))
print(m[:5]) $\rightarrow$ [39, 39, 34, 62, 35]
print(f[:5]) $\rightarrow$ [41, 28, 58, 58, 20]

Effect Size

- Use the following function to calculate effect size [5]:

```python
import math
def CohenEffectSize(group1, group2):
  diff = group1.mean() - group2.mean()
  var1 = group1.var()
  var2 = group2.var()
  n1, n2 = len(group1), len(group2)

  pooled_var = (n1*var1 + n2*var2)/(n1+n2)
  d = diff / math.sqrt(pooled_var)
  return d
```

Effect Size

- Calculate Cohen's effect size:

  d = CohenEffectSize(m, f)

  d $\rightarrow$ 0.5757033718956814
- The difference in means is 0.58 standard deviations, which is large (*see* Wikipedia `https://en.wikipedia.org/wiki/Effect_size`).
- This number shows that the difference in age between men and women is about 7.6 standard deviations.

References I

[1] VanderPlas J., Python Data Science Handbook: Essential Tools for Working with Data, O'Reilly, USA, 2016.

[2] Hunt, J., A Beginners Guide to Python 3 Programming, Springer Nature Switzerland AG, Switzerland, 2019.

[3] Hunt, J., Advance Guide to Python 3 Programming, Springer Nature Switzerland AG, Switzerland, 2019.

[4] Downey, A. B., Think Stats: Exploratory Data Analysis, O'Reilly Media, Kindle Edition, 2014.

[5] Surbhi, S. Difference Between Histogram and Bar Graph, (Available online: https://keydifferences.com/difference-between-histogram-and-bar-graph), 2017.