IIT ROORKEE

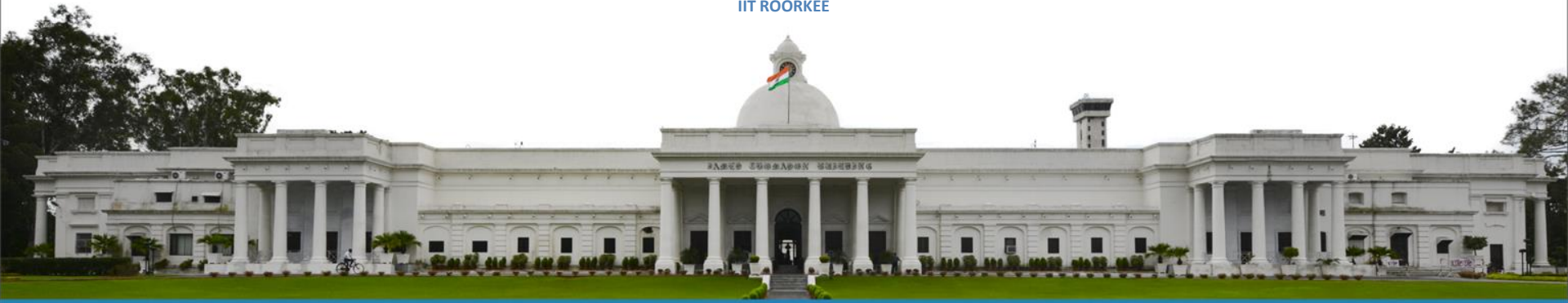
FREE ONLINE EDUCATION swayam शिक्षित भारत, उन्नत भारत

NPTEL ONLINE CERTIFICATION COURSE

Data Analytics with Python

Lecture 3: Python – Fundamentals - II

Dr. A. Ramesh
DEPARTMENT OF MANAGEMENT
IIT ROORKEE



Looking at Columns, Rows, and Cells

- **Subset Rows by Index Label: loc**

```
In [36]: print(df.head())
```

```
       country  year         pop continent  lifeExp   gdpPercap
0  Afghanistan  1952   8425333.0      Asia   28.801  779.445314
1  Afghanistan  1957   9240934.0      Asia   30.332  820.853030
2  Afghanistan  1962  10267083.0      Asia   31.997  853.100710
3  Afghanistan  1967  11537966.0      Asia   34.020  836.197138
4  Afghanistan  1972  13079460.0      Asia   36.088  739.981106
```

get the first row

- *Python counts from 0*

```
In [37]: print(df.loc[0])
```

```
country      Afghanistan
year                1952
pop          8.42533e+06
continent           Asia
lifeExp           28.801
gdpPercap         779.445
Name: 0, dtype: object
```

- *# get the 100th row*
  # Python counts from 0

```
In [38]: print(df.loc[99])
```

```
country        Bangladesh
year                 1967
pop            6.28219e+07
continent            Asia
lifeExp            43.453
gdpPercap          721.186
Name: 99, dtype: object
```

- *get the last row*

```
In [39]: print(df.tail(n=1))
```

```
      country  year         pop continent  lifeExp   gdpPercap
1703  Zimbabwe  2007  12311143.0    Africa   43.487  469.709298
```

Subsetting Multiple Rows

- *# select the first, 100th, and 1000th rows*

```
In [40]: print(df.loc[[0, 99, 999]])
```

	country	year	pop	continent	lifeExp	gdpPercap
0	Afghanistan	1952	8425333.0	Asia	28.801	779.445314
99	Bangladesh	1967	62821884.0	Asia	43.453	721.186086
999	Mongolia	1967	1149500.0	Asia	51.253	1226.041130

Subset Rows by Row Number: iloc

- *# get the 2nd row*

```
In [41]: print(df.iloc[1])

country      Afghanistan
year                1957
pop          9.24093e+06
continent           Asia
lifeExp           30.332
gdpPercap         820.853
Name: 1, dtype: object
```

- *get the 100th row*

```
In [42]: print(df.iloc[99])
```

```
country        Bangladesh
year                 1967
pop            6.28219e+07
continent             Asia
lifeExp             43.453
gdpPercap          721.186
Name: 99, dtype: object
```

- *# using -1 to get the last row*

```
In [43]: print(df.iloc[-1])
```

```
country          Zimbabwe
year                 2007
pop           1.23111e+07
continent         Africa
lifeExp            43.487
gdpPercap         469.709
Name: 1703, dtype: object
```

With `iloc`, we can pass in the `-1` to get the last row—something we couldn't do with `loc`.

- *# get the first, 100th, and 1000th rows*

```
In [44]: print(df.iloc[[0, 99, 999]])
```

	country	year	pop	continent	lifeExp	gdpPercap
0	Afghanistan	1952	8425333.0	Asia	28.801	779.445314
99	Bangladesh	1967	62821884.0	Asia	43.453	721.186086
999	Mongolia	1967	1149500.0	Asia	51.253	1226.041130

Subsetting Columns

- The Python slicing syntax uses a colon, :
- If we have just a colon, the attribute refers to everything.
- So, if we just want to get the first column using the loc or iloc syntax, we can write something like df.loc[:, [columns]] to subset the column(s).

- *# subset columns with loc*
  # note the position of the colon
  # it is used to select all rows

```
In [45]: subset = df.loc[:, ['year', 'pop']]
         print(subset.head())
```

```
   year         pop
0  1952   8425333.0
1  1957   9240934.0
2  1962  10267083.0
3  1967  11537966.0
4  1972  13079460.0
```

- # subset columns with iloc
- # iloc will alow us to use integers
- # -1 will select the last column

```
In [51]: subset = df.iloc[:, [2, 4, -1]]
         print(subset.head())
```

```
          pop  lifeExp   gdpPercap
0   8425333.0   28.801  779.445314
1   9240934.0   30.332  820.853030
2  10267083.0   31.997  853.100710
3  11537966.0   34.020  836.197138
4  13079460.0   36.088  739.981106
```

Subsetting Columns by Range

- *# create a range of integers from 0 to 4 inclusive*

```
In [52]: small_range = list(range(5))
         print(small_range)
```

```
[0, 1, 2, 3, 4]
```

- *# subset the dataframe with the range*

```
In [53]: subset = df.iloc[:, small_range]
         print(subset.head())
```

	country	year	pop	continent	lifeExp
0	Afghanistan	1952	8425333.0	Asia	28.801
1	Afghanistan	1957	9240934.0	Asia	30.332
2	Afghanistan	1962	10267083.0	Asia	31.997
3	Afghanistan	1967	11537966.0	Asia	34.020
4	Afghanistan	1972	13079460.0	Asia	36.088

Subsetting Rows and Columns

- *# using loc*

```
In [54]: # using loc
         print(df.loc[42, 'country'])

         Angola
```

- *# using iloc*

```
In [55]: print(df.iloc[42, 0])

         Angola
```

Subsetting Multiple Rows and Columns

- *#get the 1st, 100th, and 1000th rows*
  # from the 1st, 4th, and 6th columns

```
In [56]: print(df.iloc[[0, 99, 999], [0, 3, 5]])
```

```
          country continent    gdpPercap
0     Afghanistan      Asia   779.445314
99     Bangladesh      Asia   721.186086
999       Mongolia      Asia  1226.041130
```

- *if we use the column names directly,*
  # it makes the code a bit easier to read
  # note now we have to use loc, instead of iloc

```
In [57]: print(df.loc[[0, 99, 999], ['country', 'lifeExp', 'gdpPercap']])
```

```
         country  lifeExp    gdpPercap
0    Afghanistan   28.801   779.445314
99    Bangladesh   43.453   721.186086
999     Mongolia   51.253  1226.041130
```

```
In [58]: print(df.loc[10:13, ['country', 'lifeExp', 'gdpPercap']])
```

```
       country  lifeExp    gdpPercap
10  Afghanistan   42.129   726.734055
11  Afghanistan   43.828   974.580338
12      Albania   55.230  1601.056136
13      Albania   59.280  1942.284244
```

```
In [59]: print(df.head(n=10))
```

```
       country  year      pop continent  lifeExp   gdpPercap
0  Afghanistan  1952  8425333.0      Asia   28.801  779.445314
1  Afghanistan  1957  9240934.0      Asia   30.332  820.853030
2  Afghanistan  1962 10267083.0      Asia   31.997  853.100710
3  Afghanistan  1967 11537966.0      Asia   34.020  836.197138
4  Afghanistan  1972 13079460.0      Asia   36.088  739.981106
5  Afghanistan  1977 14880372.0      Asia   38.438  786.113360
6  Afghanistan  1982 12881816.0      Asia   39.854  978.011439
7  Afghanistan  1987 13867957.0      Asia   40.822  852.395945
8  Afghanistan  1992 16317921.0      Asia   41.674  649.341395
9  Afghanistan  1997 22227415.0      Asia   41.763  635.341351
```

Grouped Means

- *# For each year in our data, what was the average life expectancy?*
  # To answer this question,
  # we need to split our data into parts by year;
  # then we get the 'lifeExp' column and calculate the mean

```
In [60]: print(df.groupby('year')['lifeExp'].mean())
```

```
year
1952    49.057620
1957    51.507401
1962    53.609249
1967    55.678290
1972    57.647386
1977    59.570157
1982    61.533197
1987    63.212613
1992    64.160338
1997    65.014676
2002    65.694923
2007    67.007423
Name: lifeExp, dtype: float64
```

In [61]:
```
multi_group_var = df.\
    groupby(['year', 'continent'])\
    [['lifeExp', 'gdpPercap']].\
    mean()
print(multi_group_var)
```

```
                     lifeExp     gdpPercap
year continent
1952 Africa        39.135500   1252.572466
     Americas      53.279840   4079.062552
     Asia          46.314394   5195.484004
     Europe        64.408500   5661.057435
     Oceania       69.255000  10298.085650
1957 Africa        41.266346   1385.236062
     Americas      55.960280   4616.043733
     Asia          49.318544   5787.732940
     Europe        66.703067   6963.012816
     Oceania       70.295000  11598.522455
1962 Africa        43.319442   1598.078825
     Americas      58.398760   4901.541870
     Asia          51.563223   5729.369625
```

- If you need to “flatten” the dataframe, you can use the `reset_index` method.

```
In [62]: flat = multi_group_var.reset_index()
         print(flat.head(15))
```

	year	continent	lifeExp	gdpPercap
0	1952	Africa	39.135500	1252.572466
1	1952	Americas	53.279840	4079.062552
2	1952	Asia	46.314394	5195.484004
3	1952	Europe	64.408500	5661.057435
4	1952	Oceania	69.255000	10298.085650
5	1957	Africa	41.266346	1385.236062
6	1957	Americas	55.960280	4616.043733
7	1957	Asia	49.318544	5787.732940
8	1957	Europe	66.703067	6963.012816
9	1957	Oceania	70.295000	11598.522455
10	1962	Africa	43.319442	1598.078825
11	1962	Americas	58.398760	4901.541870
12	1962	Asia	51.563223	5729.369625
13	1962	Europe	68.539233	8365.486814
14	1962	Oceania	71.085000	12696.452430

Grouped Frequency Counts

- use the `nunique` to get counts of unique values on a Pandas `Series`.

```
In [63]: print(df.groupby('continent')['country'].nunique())
```

```
continent
Africa      52
Americas    25
Asia        33
Europe      30
Oceania      2
Name: country, dtype: int64
```

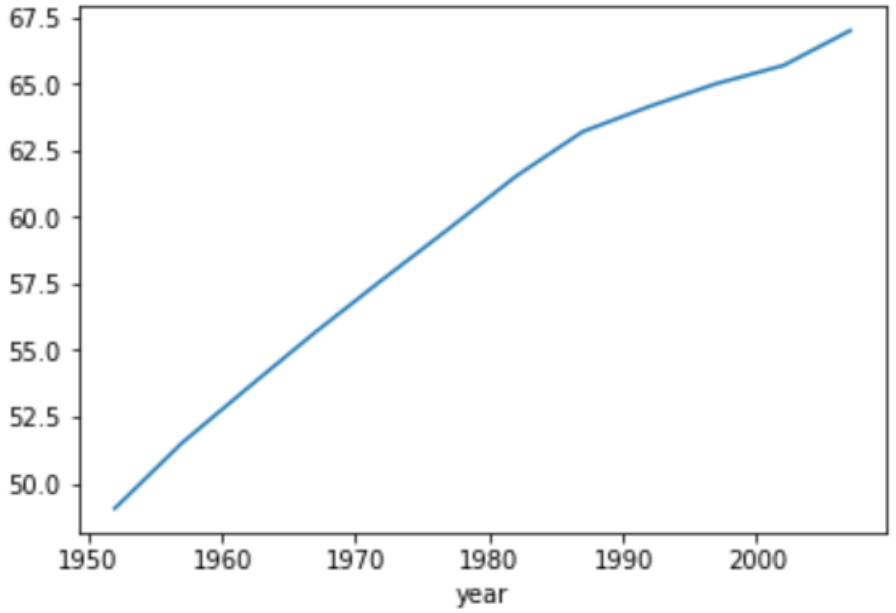
Basic Plot

In [65]:
```
global_yearly_life_expectancy = df.groupby('year')['lifeExp'].mean()
print(global_yearly_life_expectancy)
```

```
year
1952    49.057620
1957    51.507401
1962    53.609249
1967    55.678290
1972    57.647386
1977    59.570157
1982    61.533197
1987    63.212613
1992    64.160338
1997    65.014676
2002    65.694923
2007    67.007423
Name: lifeExp, dtype: float64
```

```
In [66]: global_yearly_life_expectancy.plot()
```

```
Out[66]: <matplotlib.axes._subplots.AxesSubplot at 0x229d38dd320>
```



Visual Representation of the Data

- Histogram -- vertical bar chart of frequencies
- Frequency Polygon -- line graph of frequencies
- Ogive -- line graph of cumulative frequencies
- Pie Chart -- proportional representation for categories of a whole
- Stem and Leaf Plot
- Pareto Chart
- Scatter Plot

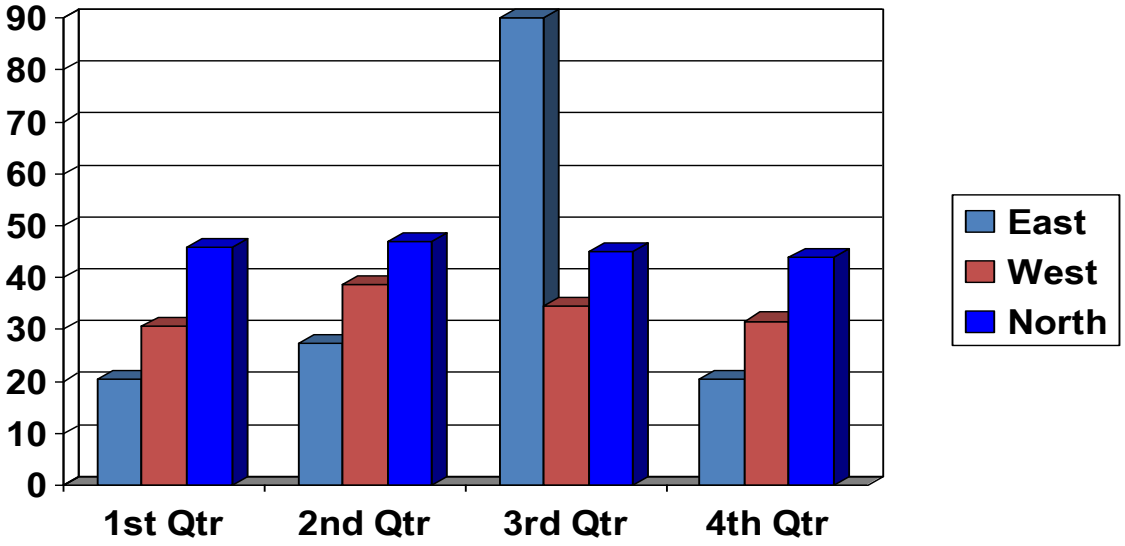
Methods of visual presentation of data

- Table

	1st Qtr	2nd Qtr	3rd Qtr	4th Qtr
East	20.4	27.4	90	20.4
West	30.6	38.6	34.6	31.6
North	45.9	46.9	45	43.9

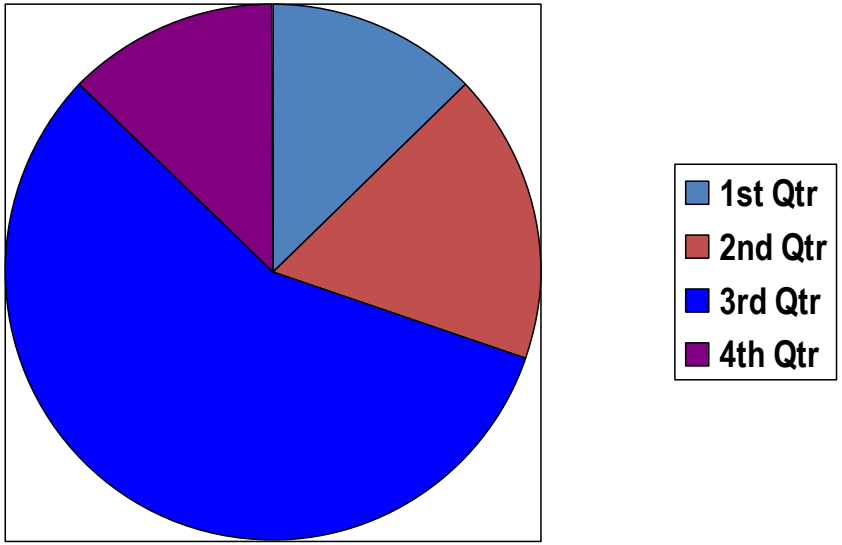
Methods of visual presentation of data

- Graphs



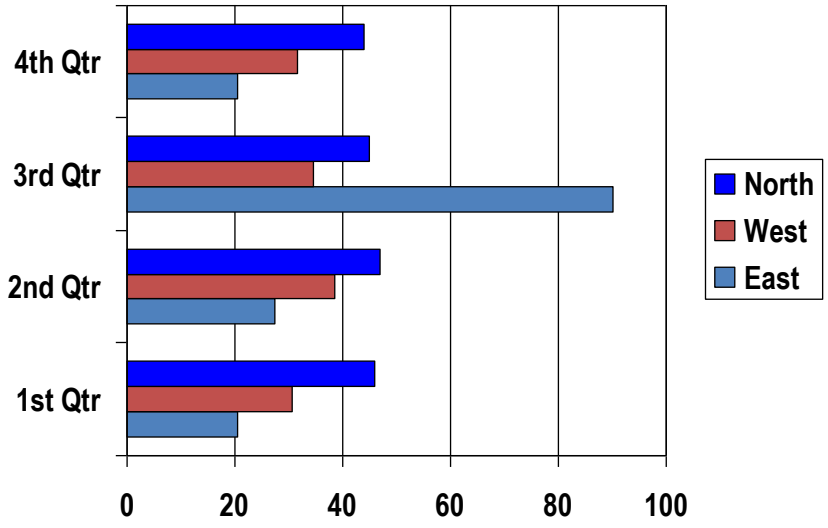
Methods of visual presentation of data

- Pie chart



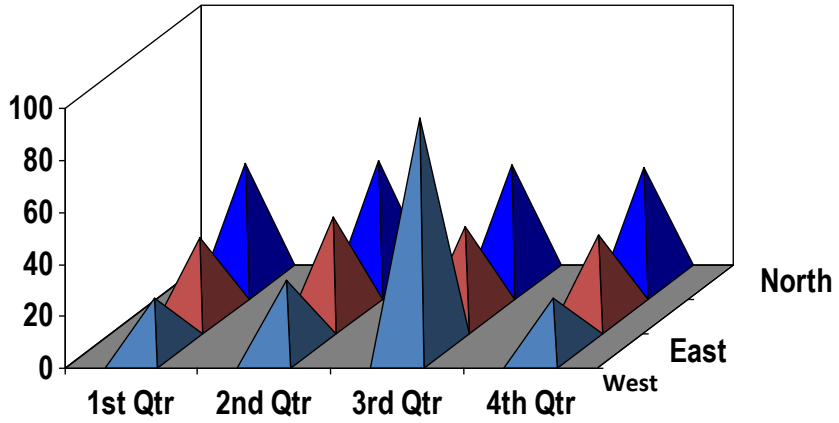
Methods of visual presentation of data

- Multiple bar chart



Methods of visual presentation of data

- Simple pictogram

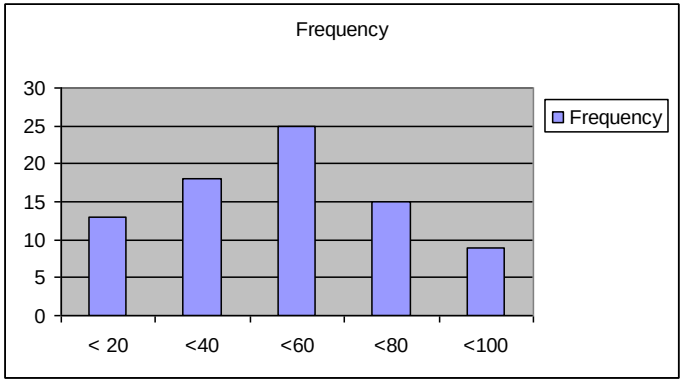


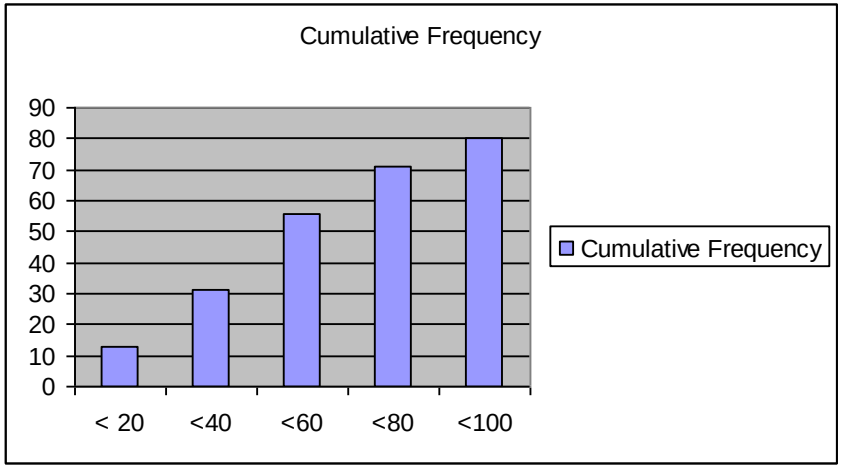
Frequency distributions

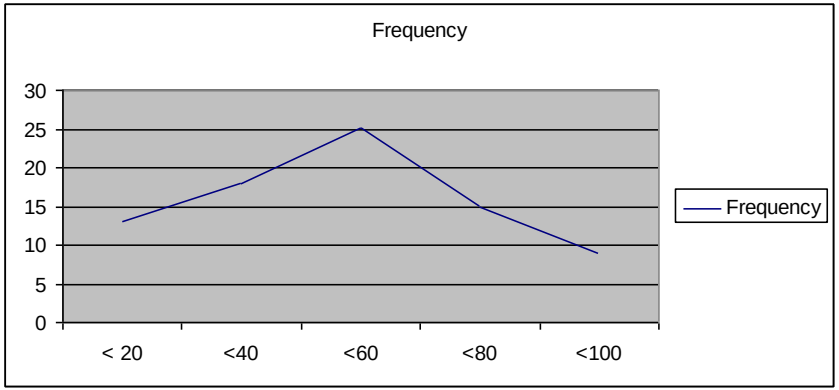
- Frequency tables

Observation Table		
Class Interval	Frequency	Cumulative Frequency
< 20	13	13
<40	18	31
<60	25	56
<80	15	71
<100	9	80

Frequency diagrams

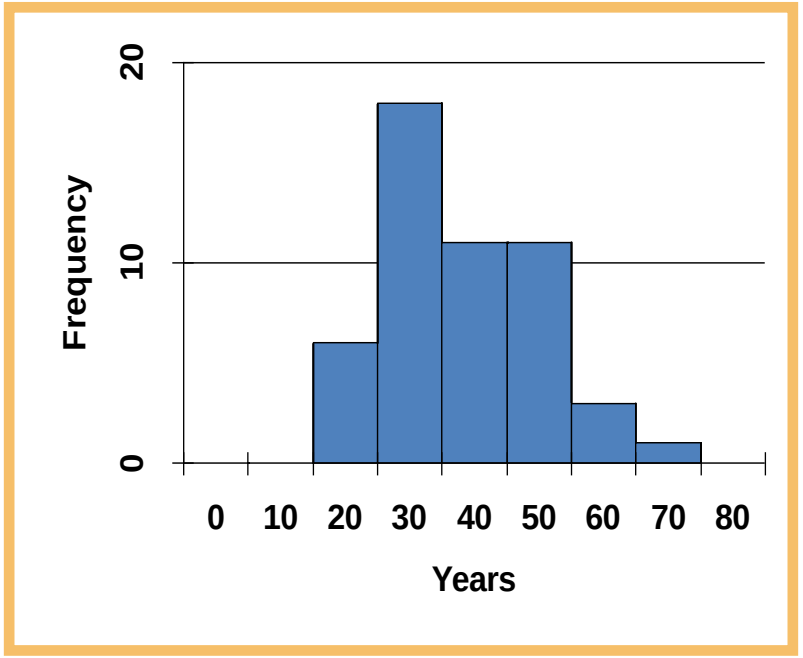






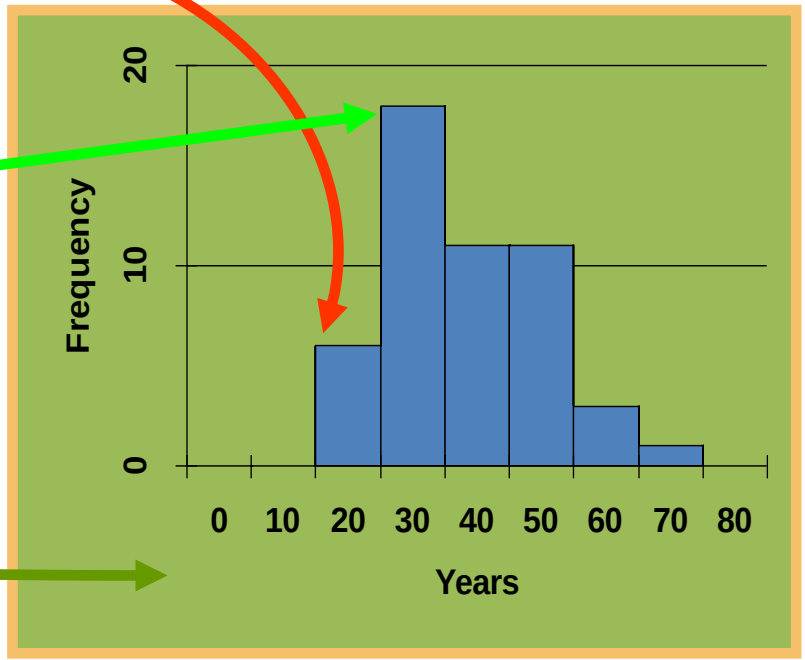
Histogram

Class Interval	Frequency
20-under 30	6
30-under 40	18
40-under 50	11
50-under 60	11
60-under 70	3
70-under 80	1



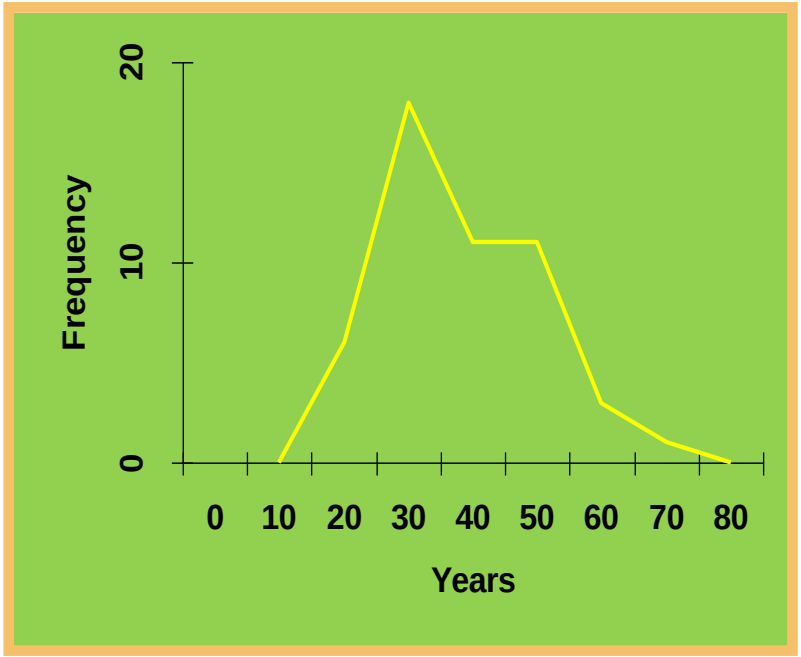
Histogram Construction

Class Interval	Frequency
20-under 30	6
30-under 40	18
40-under 50	11
50-under 60	11
60-under 70	3
70-under 80	1



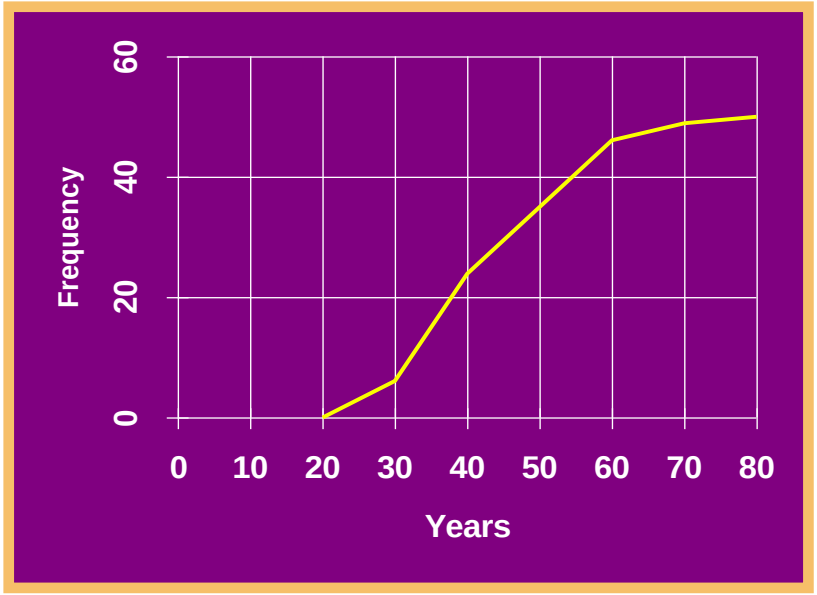
Frequency Polygon

Class Interval	Frequency
20-under 30	6
30-under 40	18
40-under 50	11
50-under 60	11
60-under 70	3
70-under 80	1



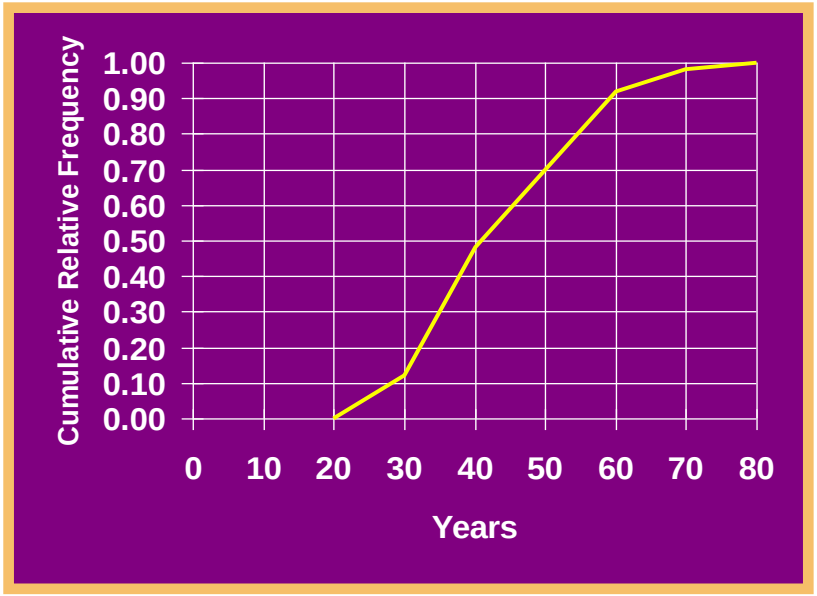
Ogive

Class Interval	Cumulative Frequency
20-under 30	6
30-under 40	24
40-under 50	35
50-under 60	46
60-under 70	49
70-under 80	50

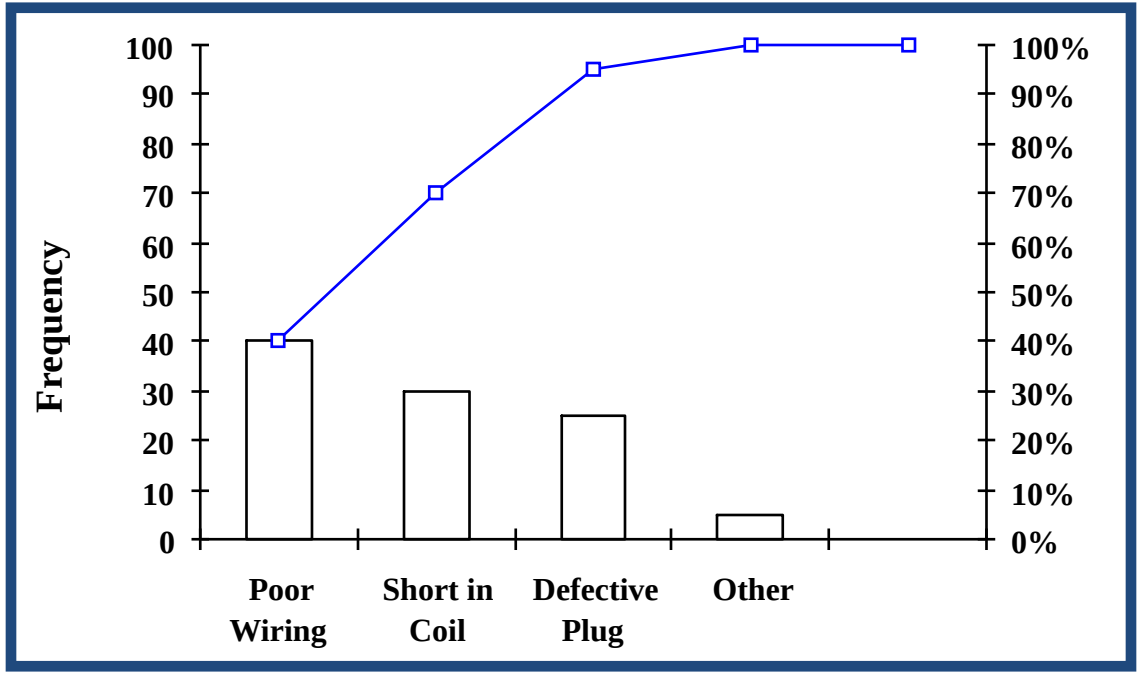


Relative Frequency Ogive

Class Interval	Cumulative Relative Frequency
20-under 30	.12
30-under 40	.48
40-under 50	.70
50-under 60	.92
60-under 70	.98
70-under 80	1.00

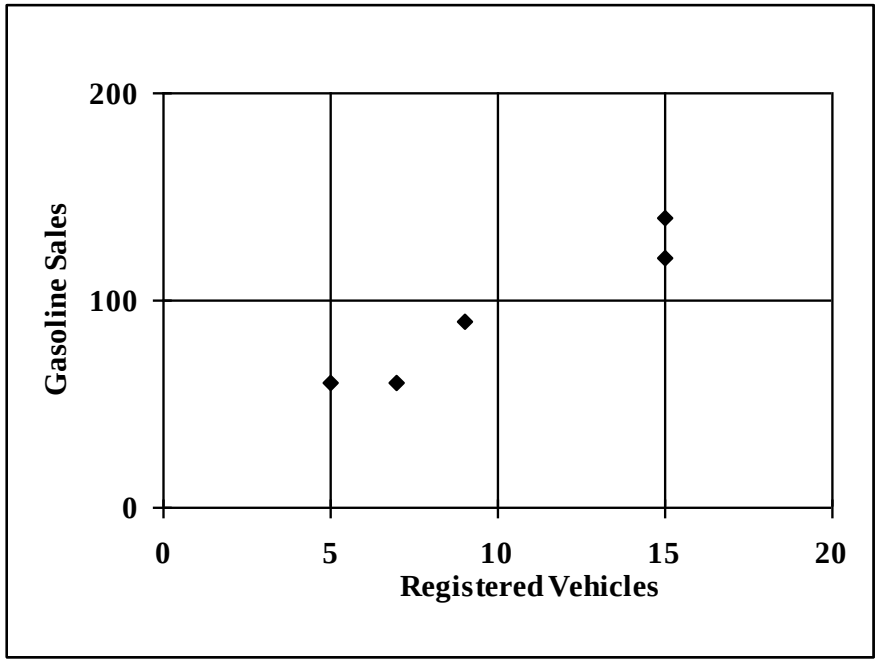


Pareto Chart



Scatter Plot

Registered Vehicles (1000's)	Gasoline Sales (1000's of Gallons)
5	60
15	120
9	90
15	140
7	60



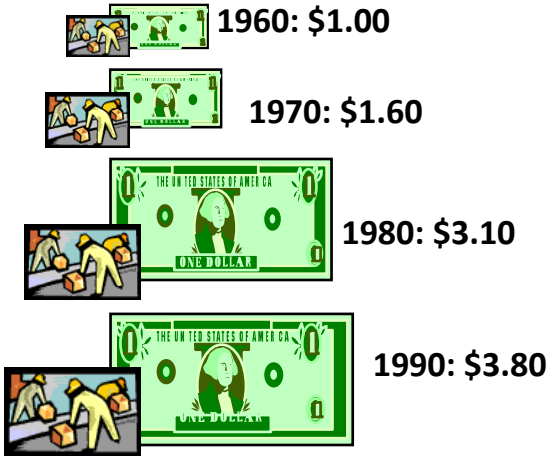
Principles of Excellent Graphs

- The graph should not distort the data
- The graph should not contain unnecessary adornments (sometimes referred to as chart junk)
- The scale on the vertical axis should begin at zero
- All axes should be properly labeled
- The graph should contain a title
- The simplest possible graph should be used for a given set of data

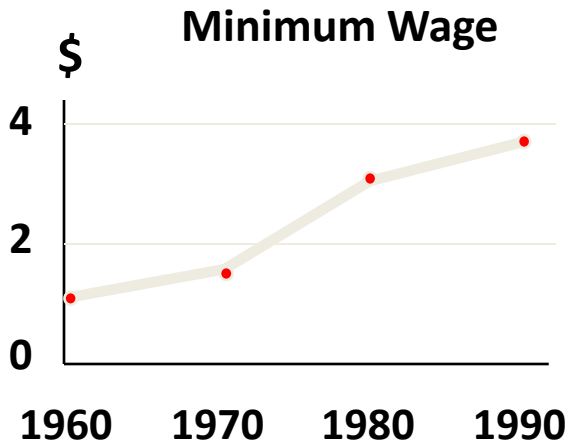
Graphical Errors: Chart Junk

Bad Presentation

Minimum Wage

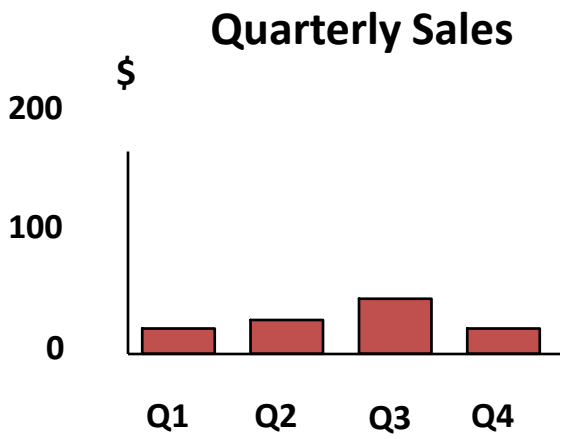


Good Presentation

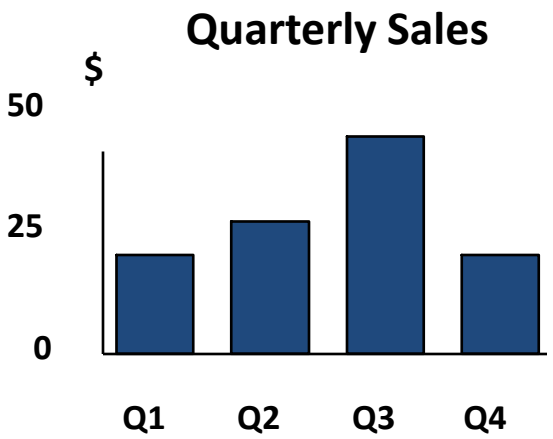


Graphical Errors: Compressing the Vertical Axis

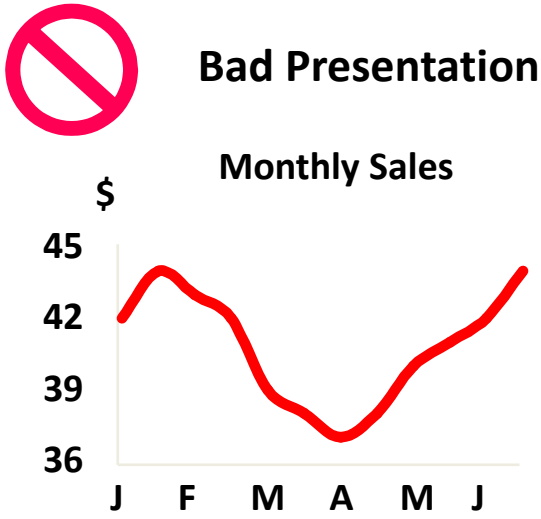
Bad Presentation

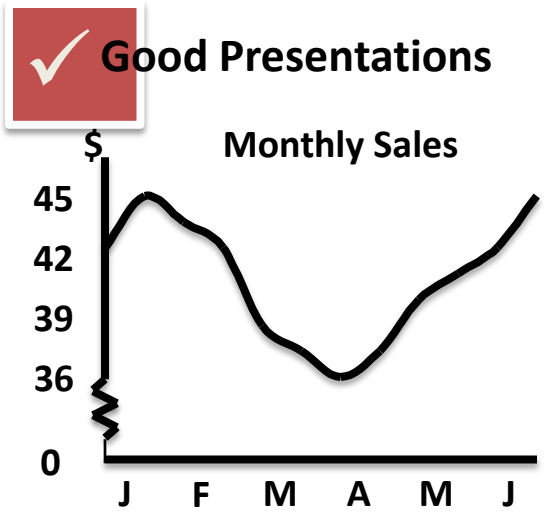


Good Presentation



Graphical Errors: No Zero Point on the Vertical Axis





Graphing the first six months of sales