Data Exploration with MongoDB and SQL

Prepared For: CS527

Prepared By: Josh Levine, Pranav Shivkumar, Pratik Mistry, Shounak Rangwala, Swapnil Kamate, Vikhyat Dhamija

The Data

- Our data set included a small csv file called challenger with 3 columns.
  - O-Ring Failure: Y/N
  - Launch Temperature: Integer
  - Leak-Check pressure: High/Medium/Low

- Our tasks our to load the csv file into a MongoDB/SQL server and write java scripts/SQL stored procedures for various aggregate functions respectively to perform univariate and bivariate analysis on the data

MongoDB + Java DEMO

Task 1: Count and Count%: Categorical Univariate Analysis

- Count: returns a count of each category within one categorical attribute.
- Count%: returns the percentage of each category within one categorical attribute.

Leak-check pressure
High
High
Medium
Low

Example

- Count
  - High: 2
  - Medium: 1
  - Low: 1
- Count%
  - High: 50%
  - Medium: 25%
  - Low: 25%

Task 1: Count and Count% Demo (Java+MongoDB)

```java
private static void computeCountAndCountPercent(String columnName, MongoCollection<Document> collection)
{
    ArrayList<CountHelper> counter = new ArrayList<>();
    FindIterable<Document> docs = collection.find();
    int totalDocs = 0;

    //Iterate over each document
    for(Document doc : docs)
    {
        totalDocs++;
        boolean flag = true;
        String docCategory = doc.getString(columnName);
        //System.out.println(doc.getString(columnName));
        for(CountHelper count: counter)
        {
            if(count.category.equals(docCategory))
            {
                count.count = (count.count+1);
                flag = false;
            }
        }
        if(flag) counter.add(new CountHelper(docCategory, 1));
    }

    System.out.println("-----------------------------------------------");
    System.out.println("Column: " + columnName);
    for(CountHelper count: counter)
    {
        double percent = (double)((count.count * 100.0f) / totalDocs);
        System.out.println("Category: " + count.category + " has a count of " + count.count + " and a count% of " + percent);
    }
    System.out.println("-----------------------------------------------");
}
```

Task 1: (Java+MongoDB) Output

```
Column: Leak-check pressure
Category: Low has a count of 6 and a count% of 26.086956024169922%
Category: Medium has a count of 2 and a count% of 8.695652008056644%
Category: High has a count of 15 and a count% of 65.21739196777344%
```

Task 2: Min, Max, Range, Mean, Mode, Median, Variance, Standard Deviation and Coefficient of Variation

- Numeric Univariate Analysis.
- Min: The smallest value of a data set.
- Max: The largest value of a data set.
- Range: The difference between the upper and lower bound. (Max - Min)
- Mean: The average of the data set. $\bar{X} = \frac{\sum X}{N}$
- Mode: The value in the data set that appears the most.
- Median: Sort the data in order, the median is the data point in the middle.

Task 2: ..Continued

- Variance: How much a data point varies from its expected value.

$$S^2 = \frac{\sum(X - \bar{X})^2}{N - 1}$$ ($\bar{X}$ is the average)

- Standard Deviation: A measure of how spread out the data points are

$$S = \sqrt{S^2}$$

- Coefficient of Variation: Ratio of the standard deviation to the mean.

$$CV = \frac{S}{\bar{X}} \times 100\%$$

Task 2: Demo (Java + MongoDB)

```java
public static void computeMinMaxRangeMeanModeMedianVarianceStandardDeviationCoefficentOfVariation(String columnName, MongoCollection<Document> collection)
{
    FindIterable<Document> docs = collection.find();
    int[] values = new int[(int) collection.countDocuments()];
    int i = 0;
    HashMap<Integer, Integer> modes = new HashMap<>();
    ArrayList<Integer> medians = new ArrayList<>();

    int min = docs.first().getInteger(columnName); int max = docs.first().getInteger(columnName); double mean = 0.0; int total = 0; int mode = 0; int modeValue = 0; int median; double variance = 0

    for(Document doc : docs)
    {
        int value = doc.getInteger(columnName);
        values[i] = value;
        i++;
        medians.add(value);
        if(value < min) min = value;
        if(value > max) max = value;
        mean += value;
        total++;
        if(modes.containsKey(value))
        {
            int count = modes.get(value);
            modes.put(value, count+1);
        }
        else
        {
            modes.put(value, 1);
        }
    }

    for (Map.Entry<Integer, Integer> entry : modes.entrySet())
    {
        if(entry.getValue() > modeValue)
        {
            modeValue = entry.getValue();
            mode = entry.getKey();
        }
    }

    variance = variance(values, total);
    double standardDeviation = Math.sqrt(variance);

    Collections.sort(medians);
    median = medians.get(medians.size()/2);
```

Task 2: Output (Java + MongoDB)

```
Column: Launch temperature
Min: 53
Max: 81
Range: 28
Mean: 69.56521739130434
Mode: 70
Median: 70
Variance: 47.637051039697546
Standard Deviation: 6.901959941907628
Coefficient of Variation: 0.004313724963692267
```

Task 3: Compute Chi2 - Categorical Bivariate Analysis

- Compute the Chi2 value of two categorical attributes.
- Determines if there is a statistical difference between the expected data and the observed data

$$\chi^2 = \sum \frac{(O-E)^2}{E}$$

Where O = Observed value, E = expected value

Task 3: ...Continued

- Because the challenger data set does not have expected values, we need to come up with expected values by binning the data.

- We will bin the data by Leak-check pressure category High. We will take the average of the temperature and use that as our expected value and we will use each value in the Launch Temperature as the observed value.

- Averaging the temperature in the High category gives us ~70

Task 3: Demo (Java + MongoDB)

```java
private static void computeChiSquared(MongoCollection<Document> collection)
{
    FindIterable<Document> docs = collection.find();
    ArrayList<Integer> temps = new ArrayList<>();
    double sum = 0.0; double average = 0.0;

    //Iterates over the collection and finds all launch temps with a high leak pressure
    for(Document doc : docs)
    {
        if(doc.getString("Leak-check pressure").equals("High"))
        {
            int temp = doc.getInteger("Launch temperature");
            sum += temp;
            temps.add(temp);
        }
    }

    average = sum/temps.size();

    //Calculating the chi squared
    double chiSquared = 0.0;
    for(Integer temp: temps)
    {
        chiSquared += (Math.pow(temp-average, 2))/average;//chi squared formula
    }

    System.out.println("-------------------------------------------------");
    System.out.println("The chi squared value of high Leak-check pressure Temperature vs the expected average is: " + chiSquared);
    System.out.println("-------------------------------------------------");

}
```

Task 3: Output (Java + MongoDB)

```
The chi squared value of high Leak-check pressure Temperature vs the expected average is: 15.148325358851674
```

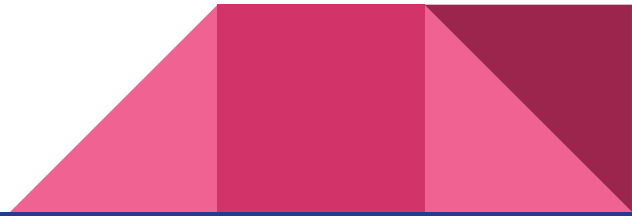
- Our Results do not mean much because they serve as input for the p-value.
  - P-Value: the probability of obtaining results as extreme as the observed results of our statistical hypothesis.

Task 4: Numeric Bivariate Analysis: Compute Linear Correlation

- Compute the linear Correlation of two numerical attributes.

- Linear correlation is a measure of dependence between two random variables.

- Types of Correlations:
  - Positive: both variables change in the same direction
  - Neutral: No relationship in the variables.
  - Negative: both variables change in the opposite direction



Task 4: (Java + MongoDB) Demo

```java
private static void computeLinearCorrelation(String columnNameOne, String columnNameTwo, MongoCollection<Document> collection)
{
    FindIterable<Document> docs = collection.find();

    int[] valuesInColumnOne = new int[(int) collection.countDocuments()];
    int[] valuesInColumnTwo = new int[(int) collection.countDocuments()];

    double varianceOfColumnOne = 0.0;
    double varianceOfColumnTwo = 0.0;

    int total = 0;
    int i = 0;

    for(Document doc : docs)
    {
        total++;

        int columnValue1 = doc.getInteger(columnNameOne);
        int columnValue2 = doc.getInteger(columnNameTwo);

        valuesInColumnOne[i] = columnValue1;
        valuesInColumnTwo[i] = columnValue2;

        i++;
    }

    varianceOfColumnOne = variance(valuesInColumnOne, total);
    varianceOfColumnTwo = variance(valuesInColumnTwo, total);

    double linearRegression = (calculateCovariance(valuesInColumnOne, valuesInColumnTwo, total)/Math.sqrt(varianceOfColumnOne*varianceOfColumnTwo));

    System.out.println("-------------------------------------------------");
    System.out.println("Column One: " + columnNameOne + "| Column Two: " + columnNameTwo);
    System.out.println("Linear Correlation between: " + columnNameOne + " and " + columnNameTwo + " is: " + linearRegression);
    System.out.println("-------------------------------------------------");
}
```

Task 4: (Java + MongoDB) Output

```
Column One: O-Ring failure| Column Two: Launch temperature
Linear Correlation between: O-Ring failure and Launch temperature is: -0.695775642615103
```

Note: A negative correlation means that there is an inverse relationship between O-Ring Failure and Launch Temperature columns.

Which can be observed when you sort the data by Launch Temperature.

O-Ring failure	Launch temperature	Leak-check pressur
0	81	High
0	79	High
0	78	High
0	76	High
0	76	High
0	75	High
0	75	High
0	73	Medium
0	72	Low
1	70	Low
0	70	Medium
1	70	High
0	70	High
0	69	Low
0	68	Low
0	67	Low
0	67	High
0	67	High
0	66	Low
1	63	High
1	58	High
1	57	High
1	53	High

Task 5: Categorical and Numeric Bivariate Analysis: Compute Z Value

- Compute the Z Value of one numerical and one categorical attribute.

- Z-score describes the values difference from the mean.
  - Positive means it lies above the mean.
  - Negative means it lies below the mean.

- Z-Score is important because it helps standardize values
  - Allows for researchers to calculate the probability of a score occuring within a standard normal distribution.
  - Allows us to to compare two scores that are from different samples.

Task 5: Compute Z Value

O-Ring failure	Launch temperature	Leak-check pressure
0	81	High
0	79	High
0	78	High
0	76	High
0	76	High
0	75	High
0	75	High
0	70	High
0	67	High
0	67	High
0	72	Low
0	69	Low
0	68	Low
0	67	Low
0	66	Low
0	73	Medium
0	70	Medium
1	70	High
1	63	High
1	58	High
1	57	High
1	53	High
1	70	Low

- Categorical: O-Ring Failure
- Numerical: Launch Temperature
- Our two lists of variables to compare will be all Launch temperatures with an O-Ring failure of 0 vs all Launch temperatures with an O-Ring failure of

$$Z = \frac{\overline{X}_1 - \overline{X}_2}{\sqrt{\frac{S_1^2}{N_1} + \frac{S_2^2}{N_2}}}$$

Where S = variance,
N = counts

Task 5: (Java + MongoDB) Demo

```java
private static void computeZValue(String categoryColumn, String numericColumn, MongoCollection<Document> collection)
{
    FindIterable<Document> docs = collection.find();

    ZValueHelper valuesInCatOne = new ZValueHelper("1");
    ZValueHelper valuesInCatTwo = new ZValueHelper("0");

    for(Document doc : docs)
    {
        if(doc.getInteger(categoryColumn) == 1)//sort into the yes cat
        {
            valuesInCatOne.addValue(doc.getInteger(numericColumn));
        }
        else//sort into the no cat
        {
            valuesInCatTwo.addValue(doc.getInteger(numericColumn));
        }
    }

    valuesInCatOne.average = mean(valuesInCatOne.values, valuesInCatOne.values.size());
    valuesInCatTwo.average = mean(valuesInCatTwo.values, valuesInCatTwo.values.size());

    valuesInCatOne.initializeArray();
    valuesInCatTwo.initializeArray();

    double varianceOne = variance(valuesInCatOne.valuesArray, valuesInCatOne.valuesArray.length);
    double varianceTwo = variance(valuesInCatTwo.valuesArray, valuesInCatTwo.valuesArray.length);
    double averages = valuesInCatTwo.average - valuesInCatOne.average;
    double varianceDivTotalOne = (varianceOne)/valuesInCatOne.valuesArray.length;
    double varianceDivTotalTwo = (varianceTwo)/valuesInCatTwo.valuesArray.length;

    double zValue = averages/(Math.sqrt(varianceDivTotalOne + varianceDivTotalTwo));

    System.out.println("-------------------------------------------------");
    System.out.println("Average 1: " + valuesInCatOne.average);
    System.out.println("Average 2: " + valuesInCatTwo.average);

    System.out.println("Variance 1: " + varianceOne);
    System.out.println("Variance 2: " + varianceTwo);

    System.out.println("Size 1: " + varianceOne);
    System.out.println("Size 2: " + varianceTwo);

    System.out.println("Category Column: " + categoryColumn + "| Numeric Column: " + numericColumn);
    System.out.println("Z-Value: " + zValue);
    System.out.println("-------------------------------------------------");
}
```

Task 5: (Java + MongoDB) Output

```
Category Column: O-Ring failure| Numeric Column: Launch temperature
Average 1: 61.833333333333336
Average 2: 72.29411764705883
Variance 1: 41.80555555555555
Variance 2: 21.14878892733564
Size 1: 41.80555555555555
Size 2: 21.14878892733564
Z-Value: 3.6504745357794635
```

- We are deviating about 3.65 from the mean.

RDBMS: SQL DEMO

Task 1: Count and Count% Demo (SQL SP's)

```
CREATE procedure [dbo].[task1]
as
declare @total_count float

set @total_count=(select count(Leak_check_pressure) from Challenger)

select Leak_check_pressure , count() as Count ,(count()*100/@total_count) as Count_percent  from
challenger group by Leak_check_pressure
Go
```

Task 1: Count and Count% Output (SQL SP's)

	Leak_check_pressure	Count	Count_percent
1	High	15	65.2173913043478
2	Low	6	26.0869565217391
3	Medium	2	8.69565217391304

Task 2: Demo (SQL SP)

```
Create procedure [dbo].[task2] as
declare @mean_temp float,@max_temp float,@min_temp float,@mode_temp float, @median_temp float, @range_temp
float, @variance_temp float, @std_dev float, @count_temp float, @cov float

set @count_temp=(select count(*) from Challenger)
set @mean_temp=(select avg(Launch_temperature) from Challenger)
set @max_temp=(select max(Launch_temperature) from Challenger)
set @min_temp=(select min(Launch_temperature) from Challenger)
set @mode_temp=(select top 1 Launch_temperature  from challenger group by Launch_temperature order by count(*) desc)
set @median_temp=(select Launch_temperature as Median from (SELECT Launch_temperature , ROW_NUMBER () OVER
(ORDER BY Launch_temperature) As rows FROM challenger) as temp WHERE temp.rows = 12)
set @range_temp=@max_temp-@min_temp
set @variance_temp=(select sum(square(Launch_temperature- @mean_temp))/@count_temp from challenger)
set @std_dev=SQRT(@variance_temp)
set @cov=(@std_dev*100/@mean_temp)

select @min_temp as Minimum , @max_temp as Maximum , @range_temp as Range , @mean_temp as Mean ,
@median_temp as Median , @mode_temp as Mode , @variance_temp as Variance , @std_dev as Std_dev , @cov as COV
go
```

Task 2: Output (SQL SP)

	Minimum	Maximum	Range	Mean	Median	Mode	Variance	Std_dev	COV
1	53	81	28	69	70	70	47.9565217391304	6.92506474620494	10.0363257191376

Task 3: Pivot Table

- Below is the pivot table for two categorical columns on which we will perform Chi 2 Test:

	O_Ring_failure	Low	Medium	High
1	0	5	2	10
2	1	1	0	5

Task 3: Demo (SQL SP)

```
CREATE procedure [dbo].[task3]
as
declare @total_low float,@total_high float,@total_medium float, @total_one float, @total_zero float, @total float,
@zero_low float, @zero_high float, @zero_medium float, @one_low float, @one_high float, @one_medium float,
@exp_zero_low float, @exp_zero_high float, @exp_zero_medium float, @exp_one_low float, @exp_one_high float,
@exp_one_medium float, @chi_sq float, @cont_coeff float

set @total_low=(select sum([Low]) from chi_pivot)
set @total_high=(select sum([High]) from chi_pivot)
set @total_medium=(select sum([Medium]) from chi_pivot)
set @total_zero=(select sum([Low])+sum([High])+sum([Medium]) from chi_pivot where O_ring_failure=0)
set @total_one=(select sum([Low])+sum([High])+sum([Medium]) from chi_pivot where O_ring_failure=1)
set @total=@total_zero+@total_one
set @zero_low=(select [Low] from chi_pivot where O_ring_failure=0)
set @zero_high=(select [High] from chi_pivot where O_ring_failure=0)
set @zero_medium=(select [Medium] from chi_pivot where O_ring_failure=0)\
set @one_low=(select [Low] from chi_pivot where O_ring_failure=1)
set @one_high=(select [High] from chi_pivot where O_ring_failure=1)
```

Task 3: Demo (SQL SP) Continued

```
set @one_medium=(select [Medium] from chi_pivot where O_ring_failure=1)
set @exp_zero_low=(@total_zero*@total_low)/@total
set @exp_zero_medium=(@total_zero*@total_medium)/@total
set @exp_zero_high=(@total_zero*@total_high)/@total
set @exp_one_low=(@total_one*@total_low)/@total
set @exp_one_medium=(@total_one*@total_medium)/@total
set @exp_one_high=(@total_one*@total_high)/@total

set@chi_sq=(square(@exp_zero_low-@zero_low)/@exp_zero_low)+(square(@exp_zero_medium-@zero_medium)/@exp
_zero_medium)+(square(@exp_zero_high-@zero_high)/@exp_zero_high)+(square(@exp_one_low-@one_low)/@exp_on
e_low)+(square(@exp_one_medium-@one_medium)/@exp_one_medium)+(square(@exp_one_high-@one_high)/@exp_o
ne_high) set @cont_coeff=sqrt(@chi_sq/(sqrt(2)*@total))

select @chi_sq as CHI2 , @cont_coeff as Contigency_coefficient

Go
```

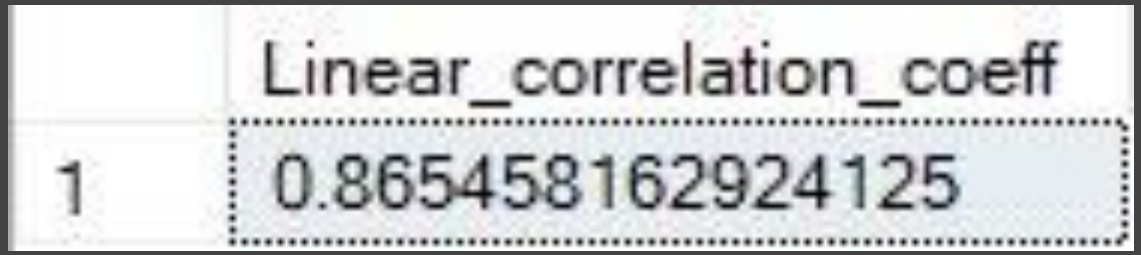
Task 3: Output (SQL SP)

	CHI2	Contigency_coefficient
1	1.390522875816 99	0.206760537283153

Task 4: (SQL SP) Demo

```
Create procedure [dbo].[task4]
as
select (Avg(ConfirmedCases * Fatalities) - (Avg(ConfirmedCases) * Avg(Fatalities))) /
(StDevP(ConfirmedCases) * StDevP(Fatalities)) as Linear_correlation_coeff from COVID
Go
```

Task 4: (SQL SP) Output



	Linear_correlation_coeff
1	0.865458162924125

Task 5: (SQL SP) Demo

```
CREATE procedure [dbo].[task5]
as
declare @mean_zero float, @mean_one float, @std_dev_one float, @std_dev_zero float
set @mean_zero=(select avg(Launch_temperature) from challenger where O_Ring_failure=0)
set @mean_one=(select avg(Launch_temperature) from challenger where O_Ring_failure=1)
set @std_dev_one=(select stdev(Launch_temperature) from challenger where O_Ring_failure=1)
set @std_dev_zero=(select stdev(Launch_temperature) from challenger where O_Ring_failure=0)
select O_Ring_failure,Launch_temperature,((Launch_temperature-@mean_zero)/@std_dev_zero) as z_score from
challenger where O_Ring_failure=0
Union ALL
select O_Ring_failure,Launch_temperature,((Launch_temperature-@mean_one)/@std_dev_one) as z_score from
challenger where O_Ring_failure=1
Go
```

Task 5: (SQL SP) Output

	O_Ring_failure	Launch_temperature	z_score
1	0	66	-1.26573861473755
2	0	69	-0.632869307368774
3	0	68	-0.843825743158365
4	0	67	-1.05478217894796
5	0	72	0
6	0	73	0.210956435789591
7	0	70	-0.421912871579183
8	0	78	1.26573861473755
9	0	67	-1.05478217894796
10	0	67	-1.05478217894796
11	0	75	0.632869307368774
12	0	70	-0.421912871579183
13	0	81	1.89860792210632
14	0	76	0.843825743158365
15	0	79	1.47669505052714
16	0	75	0.632869307368774
17	0	76	0.843825743158365
18	1	70	1.27067617440453
19	1	57	-0.564744966402015
20	1	63	0.282372483201008
21	1	70	1.27067617440453
22	1	53	-1.12948993280403
23	1	58	-0.423558724801512

Thank You