Java doc(distill)

Project: 程式設計一期末專案
Group：7

AUTHOR
董宸賓(主筆)

PUBLISHED
Jan. 13, 2024

Contents

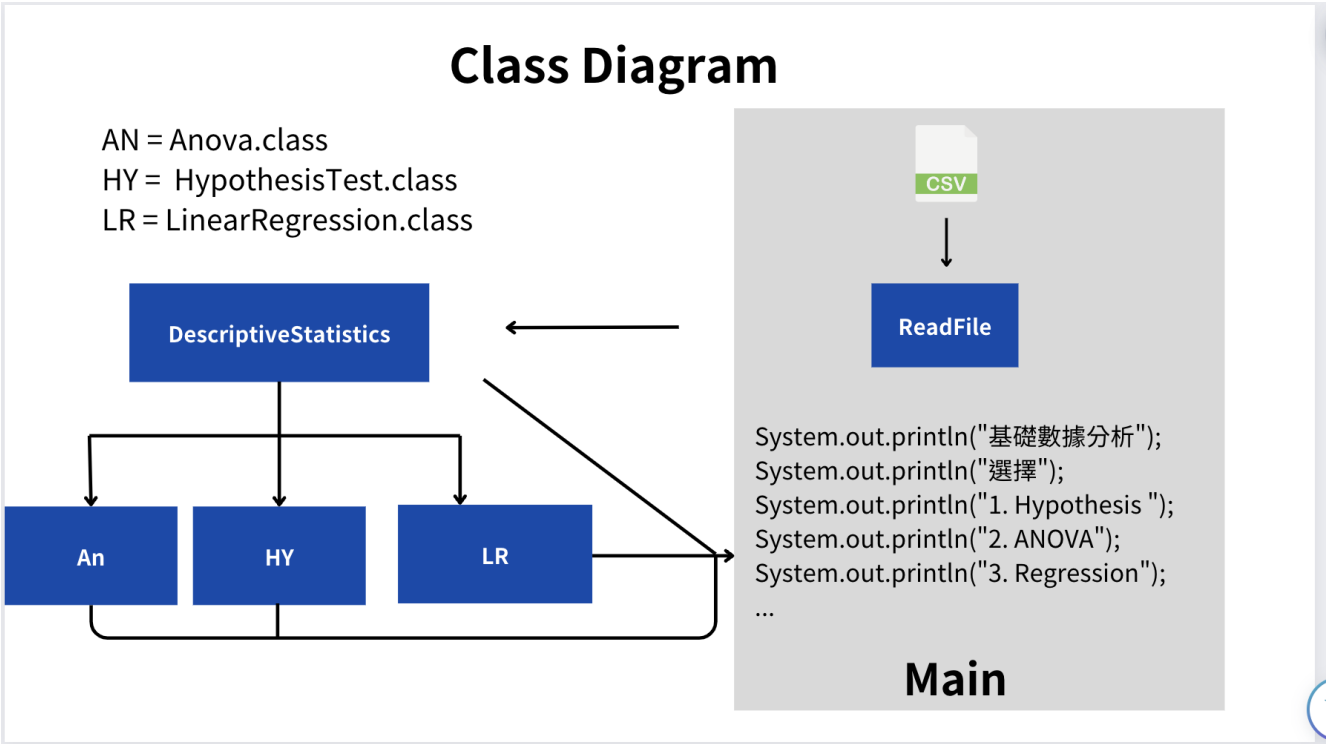
說明與動機

這是來自Java Project的說明文件，主要是為了方便使用者了解這個專案的功能，以及如何使用這個專案。

About class

這個專案有5個class，分別是ReadFile、DescriptiveStatistics、LinearRegression、Main、Anova。

相信以下圖表勝於千言萬語，以下是這個專案的類別圖。



類別圖

DescriptiveStatistics 類別說明

類別名稱： `DescriptiveStatistics`

類別說明： 這個類別提供了計算數據集描述性統計的方法，包括平均值、中位數、標準偏差等。

Constructors

建構函數	
`DescriptiveStatistics(double[] data, String name)`	用指定的數據陣列和數據集名

Methods

方法名	返回類型	描述
`getData()`	`double[]`	返回當前數據集。

方法名	返回類型	描述
getName()	String	返回數據集的名稱。
setData(double[] data)	void	設置新的數據集。
setName(String name)	void	設置數據集的新名稱。
mean()	double	計算數據集的平均值。
median()	double	計算數據集的中位數。
standardDeviation()	double	計算數據集的標準偏差。
sampleSize()	int	返回數據集的樣本大小。
populationVariance()	double	計算數據集的總體方差。
populationStandardDeviation()	double	計算數據集的總體標準偏差。
summary()	String	提供數據集的描述性統計摘要。

code

```
import java.util.Arrays;

/**
 * DescriptiveStatistics 類別提供了計算數據集描述性統計的方法。
 * 這些統計包括平均值、中位數、標準偏差、樣本大小、母體方差和母體標準偏差。
 */
public class DescriptiveStatistics {

    private double[] data;
    private String name;

    /**
     * 使用指定的數據構造一個新的 DescriptiveStatistics 實例。
     *
     * @param data 要分析的雙精度值數組。
     * @param name 數據的名稱。
     */
    public DescriptiveStatistics(double[] data, String name) {
        this.data = data;
        this.name = name;
    }

    /**
     * 獲取當前數據集。
     *
     * @return 當前存儲的雙精度數組數據。
     */
    public double[] getData() {
```

```
        return data;
    }

    /**
     * 獲取數據集的名稱。
     *
     * @return 數據集的名稱字符串。
     */
    public String getName() {
        return name;
    }

    /**
     * 將數據集設置為指定的雙精度數組。
     *
     * @param data 新的數據集雙精度數組。
     */
    public void setData(double[] data) {
        this.data = data;
    }

    /**
     * 設置數據集的名稱。
     *
     * @param name 數據集的新名稱字符串。
     */
    public void setName(String name) {
        this.name = name;
    }

    /**
     * 計算數據集的平均值。
     *
     * @return 平均值的雙精度數值。
     */
    public double mean() {
        double sum = 0.0;
        for (double num : this.data) {
            sum += num;
        }
        return sum / this.data.length;
    }

    /**
     * 計算數據集的中位數。
     *
```

```
     * @return 中位數的雙精度數值。
     */
    public double median() {
        int size = this.data.length;
        double[] sortedData = Arrays.copyOf(this.data, size);
        Arrays.sort(sortedData);
        if (size % 2 == 0) {
            return (sortedData[size / 2 - 1] + sortedData[size / 2]) / 2.0;
        } else {
            return sortedData[size / 2];
        }
    }

    /**
     * 計算數據集的標準偏差。
     *
     * @return 標準偏差的雙精度數值。
     */
    public double standardDeviation() {
        double mean = mean();
        double sumOfSquares = 0.0;
        for (double num : this.data) {
            sumOfSquares += Math.pow(num - mean, 2);
        }
        return Math.sqrt(sumOfSquares / this.data.length);
    }

    /**
     * 計算數據集的樣本大小。
     *
     * @return 整數值，表示數據集中的樣本數量。
     */
    public int sampleSize() {
        return this.data.length;
    }

    /**
     * 計算數據集的母體方差。
     *
     * @return 母體方差的雙精度數值。
     */
    public double populationVariance() {
        double mean = mean();
        double sumOfSquares = 0.0;
        for (double num : this.data) {
            sumOfSquares += Math.pow(num - mean, 2);
```

```
        }
        return sumOfSquares / this.data.length;
    }

    /**
     * 計算數據集的母體標準偏差。
     *
     * @return 母體標準偏差的雙精度數值。
     */
    public double populationStandardDeviation() {
        return Math.sqrt(populationVariance());
    }

    /**
     * 提供數據集的描述性統計摘要。
     *
     * @return 描述數據集統計信息的字符串。
     */
    public String summary() {
        return "數據名稱: " + this.name + "\n平均值: " + mean() + "\n中位數: " +
        median() + "\n標準偏差: " + standardDeviation() + "\n樣本大小: " +
        sampleSize() + "\n母體方差: " + populationVariance() + "\n母體標準偏差:
        " + populationStandardDeviation();
    }

    /**
     * 提供數據集描述性統計分析的簡介。
     *
     * @return 描述數據集分析的字符串。
     */
    public String description() {
        return "以下將把每個數據集的數據進行描述性統計分析，包括平均值、中位數、標準(偏)
        差、樣本大小、母體方差和母體標準偏差。";
    }

    /**
     * 講解數據集分析中使用的公式。
     *
     * @return 關於分析公式的解釋字符串。
     */
    public String explain(){
        return "以下為您講解公式\n"+ "平均值 = Σx / n\n" + "中位數 = (x[n/2] +
        x[n/2+1]) / 2\n" + "標準偏差 = √(Σ(x - x平均值)^2 / n)\n"  + "母體方差 =
        Σ(x - x平均值)^2 / n\n" + "母體標準偏差 = √(Σ(x - x平均值)^2 / n)\n";
    }
}
```

LinearRegression 類別說明

類別名稱： `LinearRegression`

繼承： `DescriptiveStatistics`

類別說明： 這個類別繼承自 `DescriptiveStatistics`，提供線性回歸分析的功能。它可以計算線性回歸模型的斜率和截距，並使用模型進行預測。

Constructors

建構函數
`LinearRegression(double[] xData, double[] yData, String name)` 使用自變量和

Methods

方法名	返回類型	描述
`calculateSlope()`	`double`	計算回歸線的斜率（beta）。
`calculateIntercept()`	`double`	計算回歸線的截距（alpha）。
`predict(double x)`	`double`	使用線性回歸模型預測給定 x 值的 y 值。
`summary()`	`String`	提供線性回歸模型的摘要。
`explain()`	`String`	提供線性回歸的基本概念和模型解釋。
`description()`	`String`	提供簡單線性回歸的詳細說明和公式。

code

```
import java.util.Arrays;

/**
 * LinearRegression 類繼承自 DescriptiveStatistics 類，提供線性回歸分析的功能。
 * 它能計算線性回歸模型的斜率和截距，並使用模型進行預測。
 */
public class LinearRegression extends DescriptiveStatistics {

    private double[] xData; // 自變量
    private double[] yData; // 因變量

    /**
```

```
     * 使用自變量和因變量的數據，以及數據集的名稱來構造 LinearRegression 物件。
     *
     * @param xData 自變量的數據陣列。
     * @param yData 因變量的數據陣列。
     * @param name 數據集的名稱。
     */
    public LinearRegression(double[] xData, double[] yData, String name) {
        super(yData, name); // 使用因變量初始化 DescriptiveStatistics
        this.xData = xData;
        this.yData = yData;
    }

    /**
     * 計算回歸線的斜率（beta）。
     *
     * @return 斜率值。計算方法是將 xData 和 yData 的協方差除以 xData 的變異數。
     */
    public double calculateSlope() {
        double meanX = new DescriptiveStatistics(xData, "x").mean();
        double meanY = mean();
        double numerator = 0.0;
        double denominator = 0.0;

        for (int i = 0; i < xData.length; i++) {
            numerator += (xData[i] - meanX) * (yData[i] - meanY);
            denominator += Math.pow(xData[i] - meanX, 2);
        }

        return numerator / denominator;
    }

    /**
     * 計算回歸線的截距（alpha）。
     *
     * @return 截距值。計算方法是 y 數據的平均值減去斜率乘以 x 數據的平均值。
     */
    public double calculateIntercept() {
        double meanX = new DescriptiveStatistics(xData, "x").mean();
        double meanY = mean();
        return meanY - calculateSlope() * meanX;
    }

    /**
     * 使用線性回歸模型預測給定 x 值的 y 值。
     *
     * @param x 自變量 x 的值。
```

```
     * @return 預測的 y 值。計算方法是截距加上斜率乘以 x。
     */
    public double predict(double x) {
        return calculateIntercept() + calculateSlope() * x;
    }

    /**
     * 提供線性回歸模型的摘要。
     *
     * @return 描述線性回歸模型的字符串，包括斜率、截距和回歸方程。
     */
    public String summary() {
        return "線性回歸模型 - " + getName() + "\n斜率 (beta0): " +
        calculateSlope() + "\n截距 (beta1): " + calculateIntercept() + "\n本數
        據的迴歸模型是: ŷ = " + calculateSlope() + "x + " +
        calculateIntercept();
    }

    /**
     * 提供線性回歸的基本概念和模型解釋。
     *
     * @return 線性回歸的基本概念和模型解釋的字符串。
     */
    public String explain() {
        return "Statitical (True) Model is: y = f(x) + ε \n" +
               "其中： y = 應變數； x = 自變數 \n" +
               "Statitical (True) Model ------>  Fitted model , eg: ŷ = b1x +
        b0\n" +
               "利用 data 來尋找Y和X的關係（不一定是因果關係）\n";
    }

    /**
     * 提供簡單線性回歸的詳細說明和公式。
     *
     * @return 簡單線性回歸的詳細說明和公式的字符串。
     */
    public String description() {
        return "以下是簡單線性回歸的概念\n" +
               "=> involves one independent variable and one dependent
        variable.\n" +
               "Suppose: y = B0 + B1x + ε , then E(y) = B0 + B1x , ε~NID(0 ,
        σ^2)\n" +
               "Sampling and Fitted : ŷ = b0 + b1x\n" +
               "Estimated: ŷ = b0 + b1x ------> E(y) = B0 + B1x\n" +
               "b0 ------> B0  ;  b1 ------> B1\n" +
               "(residual) e = y - ŷ  ------> ε = y - E(y)\n" +
               "by min Σ(y - ŷ)^2 = min Σ(y - b0 + b1x)^2\n" +
               "由 ∂SSE / ∂b0 = 0  and  ∂SSE / ∂b1 = 0\n" +
```

```
                "得 b1 = Σ(x − x̄)^2 (y − ȳ)^2 / Σ(x − x̄)^2\n" +
                "b0 = ȳ − b1x̄";
    }
}
```

Anova 類別說明

類別名稱： Anova

繼承： DescriptiveStatistics

類別說明： 這個類別提供了執行單因素方差分析（ANOVA）的方法。用於分析多組數據集之間的均值是否存在顯著差異。

Constructors

建構函數	描述
Anova(double[][] groups, String name)	使用多組數據構造 Anova 對象，每個子數組代

Methods

方法名	返回類型	描述
flatten(double[][] arrays)	double[]	將二維數組展平為一維數組。
overallMean()	double	計算總體均值。
totalSumOfSquares()	double	計算總體平方和（SST）。
betweenGroupSumOfSquares()	double	計算組間平方和（SSB）。
withinGroupSumOfSquares()	double	計算組內平方和（SSW）。
calculateFValue()	double	計算 ANOVA 的 F 值。
summary()	String	提供 ANOVA 分析的摘要。
explain()	String	提供 ANOVA 分析的基本概念和使用方法的解釋
description()	String	提供 ANOVA 的詳細描述和公式。

code

```
import java.util.Arrays;

/**
 * Anova 類別提供了執行單因素方差分析（ANOVA）的方法。
 * 這個類別繼承自 DescriptiveStatistics，用於分析多組數據集之間的均值是否存在顯著差異。
 */
public class Anova extends DescriptiveStatistics {
    private double[][] groups;
```

```
/**
 * 使用多組數據構造 Anova 對象。
 *
 * @param groups 二維數組，每個子數組代表一組數據。
 * @param name 數據集的名稱。
 */
public Anova(double[][] groups, String name) {
    super(flatten(groups), name);
    this.groups = groups;
}

/**
 * 將二維數組展平為一維數組。
 *
 * @param arrays 要展平的二維數組。
 * @return 展平後的一維數組。
 */
public static double[] flatten(double[][] arrays) {
    return
    Arrays.stream(arrays).flatMapToDouble(Arrays::stream).toArray();
}

/**
 * 計算總體均值。
 *
 * @return 總體均值。
 */
public double overallMean() {
    return mean();
}

/**
 * 計算總體平方和（SST）。
 *
 * @return 總體平方和。
 */
public double totalSumOfSquares() {
    double overallMean = overallMean();
    return Arrays.stream(this.getData()).map(x -> Math.pow(x -
    overallMean, 2)).sum();
}

/**
 * 計算組間平方和（SSB）。
 *
```

```
     * @return 組間平方和。
     */
    public double betweenGroupSumOfSquares() {
        double overallMean = overallMean();
        return Arrays.stream(groups).mapToDouble(group ->
            group.length * Math.pow(new DescriptiveStatistics(group,
        "").mean() - overallMean, 2)
        ).sum();
    }

    /**
     * 計算組內平方和（SSW）。
     *
     * @return 組內平方和。
     */
    public double withinGroupSumOfSquares() {
        return totalSumOfSquares() - betweenGroupSumOfSquares();
    }

    /**
     * 提供 ANOVA 分析的摘要。
     *
     * @return 描述 ANOVA 分析結果的字符串。
     */
    public String summary() {
        return "ANOVA 分析 - " + getName() + "\n總體平方和 (SST): " +
        totalSumOfSquares() +
                "\n組間平方和 (SSB): " + betweenGroupSumOfSquares() +
                "\n組內平方和 (SSW): " + withinGroupSumOfSquares();
    }

    /**
     * 計算 ANOVA 的 F 值。
     * F 值是用於測量組間變異與組內變異的比率。
     *
     * @return F 統計量的值。
     */
    public double calculateFValue() {
        double ssb = betweenGroupSumOfSquares(); // 已計算的組間平方和
        double ssw = withinGroupSumOfSquares(); // 已計算的組內平方和
        int dfBetween = groups.length - 1; // 組間自由度
        int dfWithin = Arrays.stream(groups).mapToInt(arr ->
        arr.length).sum() - groups.length; // 組內自由度

        double msb = ssb / dfBetween; // 組間均方
        double msw = ssw / dfWithin; // 組內均方
```

```
        return msb / msw; // 計算 F 值
    }

    /**
     * 提供 ANOVA 分析的基本概念和使用方法的解釋。
     *
     * @return ANOVA 分析的基本概念和方法的字符串描述。
     */
    public String explain() {
        return "ANOVA（分析變異）用於比較三個或更多組的平均數是否有顯著差異。" +
               "它將總變異分解為組間變異和組內變異，並通過 F 統計量來評估組間變異是否顯
        著大於組內變異。";
    }

    /**
     * 提供 ANOVA 的詳細描述和公式。
     *
     * @return ANOVA 的詳細描述和公式的字符串描述。
     */
    public String description() {
        return "ANOVA 通過計算 F 值來測試組間差異的顯著性。" +
               "F 值是組間均方（MSB）和組內均方（MSW）的比率，" +
               "其中 MSB = 組間平方和(SSB) / 組間自由度(dfBetween)，" +
               "MSW = 組內平方和(SSW) / 組內自由度(dfWithin)。" +
               "高 F 值通常表明組間變異顯著大於組內變異，從而指示組間存在顯著差異。";
    }
}
```

HypothesisTest 類別說明

類別名稱： `HypothesisTest`

類別說明： 這個類別繼承自 `DescriptiveStatistics`，專為進行假設檢定設計，提供了計算和分析假設檢定所需的各種工具和方法。

Constructors

建構函數	
`HypothesisTest(double[] data, String name)`	使用指定的數據和名稱初始化 Hypothes

屬性

屬性名	類型	描述
a	double	顯著性水平 alpha。
hypo	double	假設的平均值。
direct	String	檢定的方向（左尾、右尾或雙尾）。

Methods

方法名	返回類型
`getAlpha()`	double
`setDirect(String direct)`	void
`getDirect()`	String
`setAlpha(double alpha)`	void
`setNullHypo(double hypo, String direct)`	void
`p_Population(double populationStandardDeviation)`	double
`t_Sample()`	double
`analysis(double p)`	void
`tAnalysis(double t)`	void
`instruction()`	void
`instruction1()`	void
`instruction2()`	void
`instruction3()`	void

方法名	返回類型
findzTable(double z)	double
findtTable(double alpha, int degreesOfFreedom, boolean twoTail)	double

```
import java.util.HashMap;
import java.util.Map;

/**
 * 使用指定的數據和名稱構造 HypothesisTest 對象。初始化顯著性水平 alpha 為 0.05。
 *
 * @param data 用於統計分析的數據數組。
 * @param name 數據集的名稱。
 */

public class HypothesisTest extends DescriptiveStatistics{


    private double a; // 顯著性水平 alpha
    private double hypo; // 假設的平均值
    private String direct;


        public HypothesisTest(double[] data, String name) {
            super(data,name);
            this.a = 0.05;
        }

  /**
 * 設置檢定的方向。
 *
 * @param direct 檢定的方向（左尾、右尾或雙尾）。
 */
    public double getAlpha() {
        return a;
    }

    /**
 * 獲取檢定的方向。
 *
 * @return 檢定的方向。
 */
    public void setDirect(String direct) {
        this.direct = direct;
    }
```

```
    /**
* 設置顯著性水平（alpha）。
*
* @param alpha 要設置的顯著性水平。
*/
    public String getDirect() {
        return direct;
    }



    public void setAlpha(double alpha) {
        this.a = alpha;
    }



    /**
* 設置零假設和對立假設的值。
*
* @param hypo   零假設的值。
* @param direct 檢定的方向（左尾、右尾或雙尾）。
*/
    public void setNullHypo(double hypo, String direct) {

        this.hypo = hypo;
        this.direct = direct;

        boolean flag = false;
        while (!flag) {
            if (direct.equals("左尾")) {
                System.out.println("H0: μ>=" + hypo + " H1: μ<" + hypo);
                flag = true;
            } else if (direct.equals("右尾")) {
                System.out.println("H0: μ<=" + hypo + " H1: μ>" + hypo);
                flag = true;
            } else if (direct.equals("雙尾")) {
                System.out.println("H0: μ=" + hypo + " H1: μ≠" + hypo);
                flag = true;
            } else {
                System.out.println("輸入錯誤，請重新輸入");
            }
        }
    }

    /**
* 計算假設檢定的 p 值（適用於總體標準差已知的情況）。
```

```
*
* @param populationStandardDeviation 總體標準差。
* @return 假設檢定的 p 值。
*/
    public double p_Population(double populationStandardDeviation) {

        double diff = super.mean() - this.hypo;
        double error = populationStandardDeviation /
        Math.sqrt(super.sampleSize());
        double z = diff / error;
        double p = findzTable(z); // 假設 findzTable 方法返回 z 值對應的 p 值

        if (z > 0) {
            return (z < 3.59) ? p : 0.4999;
        } else {
            return 0.5 - p;
        }
    }



      /**
     * 使用樣本變異數計算假設檢定的 t 值。
     *
     * @return 假設檢定的 t 值。
     */
    public double t_Sample() {
        double diff = super.mean() - this.hypo;
        double error = super.standardDeviation() /
        Math.sqrt(super.sampleSize());
        double t = diff / error;

        return t;
    }



    /**
   * 根據計算得到的 p 值分析結果。
   *
   * @param p 計算得到的 p 值。
   */
    public void analysis(double p) {

        if (direct.equals("雙尾")) {

            if (p < a / 2) {
```

```
                System.out.println("在此情境中，由於p-value(" + p + ")小於
        alpha/2 (" + a / 2 + ")，可知此資料拒絕H0之假設");
            } else {
                System.out.println("在此情境中，由於p-value(" + p + ")大於或等於
        alpha/2 (" + a / 2 + ")，可知此資料無法拒絕H0之假設");
            }


        } else {


            if (p > a) {
                System.out.println("在此情境中，由於p-value("+p+")大於
        alpha("+a+")，可知此資料無法拒絕H0之假設");
            } else {
                System.out.println("在此情境中，由於p-value("+p+")小於
        alpha("+a+")，可知此資料無法拒絕H0之假設");
            }
        }
    }



/**
 * 根據計算得到的 t 值分析結果。
 *
 * @param t 計算得到的 t 值。
 */
    public void tAnalysis(double t) {

        this.a = Math.round(this.a * 100.0) / 100.0;

        int n = super.sampleSize();

        if (direct.equals("右尾")) {

            Double tValue = findtTable(this.a, n, false);

            if (tValue > t) {
                System.out.println("在此情境中，由於t值("+t+")小於臨界值
        ("+tValue+")，可知此資料無法拒絕H0之假設");
            } else {
                System.out.println("在此情境中，由於t值("+t+")大於臨界值
        ("+tValue+")，可知此資料拒絕H0之假設");
            }

        } else if (direct.equals("左尾")) {

            double tValue = findtTable(this.a, n, false);
            if (tValue < t) {
```

```
            System.out.println("在此情境中，由於t值("+t+")大於臨界值
        ("+tValue+")，可知此資料無法拒絕H0之假設");
        } else {
            System.out.println("在此情境中，由於t值("+t+")小於臨界值
        ("+tValue+")，可知此資料拒絕H0之假設");
        }

    } else if (direct.equals("雙尾")) {

        Double tValue = findtTable(this.a, n, true);

        if (-tValue < t && tValue > t) {
            System.out.println("在此情境中，由於t值("+t+")落在正負臨界值
        ("+tValue+")之間，可知此資料無法拒絕H0之假設");
        } else {
            System.out.println("在此情境中，由於t值無落在正負臨界值之間，可知此資
        料拒絕H0之假設");
        }

    }

}


public void instruction() {

    System.out.println("假說檢定（英語：hypothesis testing）是推論統計中用於檢
    定現有數據是否足以支持特定假設的方法。\n"
            + "一旦能估計未知母數，就會希望根據結果對未知的真正母數值做出適當的推
    論。\n" + "欲檢定統計上假設的正確性的為虛無假說，虛無假說通常由研究者決定，反映研
    究者對未知母數的看法。\n"
            + "相對於虛無假說的其他有關母數之論述是對立假說，它通常反應了執行檢定的
    研究者對母數可能數值的另一種（對立的）看法\n");
}

public void instruction1() {

    System.out.println("以下說明此步驟之意義：\n" + "在檢定之前，我們需要針對情境
    進行假設。其中包含「虛無假設」以及「對立假設」兩種情況：\n"
            + "「虛無假設」對母體參數提出一個主張，假設此主張為「真實」（除非能證明
    此主張非真！）。\n"
            + "對立假設是相對於虛無假設所提出的另一個不同(相反）的假設或主張，必須有
    足夠的證據，才能說明此主張為真。\n" + "檢定方式又因虛無假設之不同，分為三種檢定：
    \n"
            + "雙尾檢定： H0 : θ=θ0； H1 : θ≠θ0\n" + "__左尾檢定： H0 : θ≧θ0
    ； H1 : θ<θ0\n" + "__右尾檢定： H0 : θ≦θ0 ； H1 : θ>θ0");

}
```

```
  public void instruction2() {

      System.out.println("檢定統計量是由樣本所算出來的一個值，用來決定是否接受或拒絕H0。\n" + "__不同參數的假設檢定，使用不同的檢定統量。常用的檢定統計量有：Z, t。\n"
              + "在母體平均數的假設檢定裡，不同的情形下使用不同的檢定統計量。\n"
              + "若母體變異數已知，則使用z = (xbar-μ)/(σ-√n)\n，其中xbar代表檢定量、μ代表母體平均數、σ代表母體標準差、n代表母體樣本數量"
              + "若母體變異數未知，但樣本數超過30，可將樣本視為一母體，同樣使用z = (xbar-μ)/(σ-√n)\n"
              + "若母體變異數未知，但樣本數小於30，則同樣使用t = (xbar-μ)/(s-√n)，其中s代表樣本標準差\n");

  }

  public void instruction3() {

      System.out
              .println("研究人員必須決定一個決策法則，以瞭解何時『不拒絕』 H0 ；何時拒絕 H0 。\n" + "一般我們說『不拒絕』H0 ，而不說接受 H0 ，因為我們只是沒有足夠證據拒絕，而不是接受。\n"
                      + "決策法則通常是決定一個不拒絕域 (NonrejectionRegion, 或稱接受域 ) 與拒絕域 (Rejection Region)。\n"
                      + "當檢定統計量落入不拒絕域： 『不拒絕』 H0 ！\n" + "當檢定統計量落入拒絕域：拒絕 H0 ；接受 H1\n"
                      + "接受域與拒絕域的接點，稱為臨界點(Critical Point)。\n" + "臨界值的決定，是根據顯著水準α並利用機率分配計算而得，分成三種形式：\n"
                      + "雙尾檢定 (落在兩邊拒絕)、右尾檢定 (落在右邊拒絕)、左尾檢定(落在左邊拒絕)\n"
                      + "在z檢定時，我們也會經由計算其p-value並與alpha值比較，並得到結果\n" + "一般來說，當p-value小於alpha，則拒絕H0");

  }


/**
 * 查找 z 表以獲得對應的 p 值。
 *
 * @param z z 值。
 * @return 對應的 p 值。
 */
  public double findzTable(double z) {

      Map<Double, Double> zTable = new HashMap<>();

      zTable.put(0.00, 0.0000);
      zTable.put(0.01, 0.0040);
      zTable.put(0.02, 0.0080);
```

```
        zTable.put(0.03, 0.0120);
        zTable.put(0.04, 0.0160);
        zTable.put(0.05, 0.0199);
        zTable.put(0.06, 0.0239);
        zTable.put(0.07, 0.0279);
        zTable.put(0.08, 0.0319);
        zTable.put(0.09, 0.0359);
        zTable.put(0.10, 0.0398);
        zTable.put(0.11, 0.0438);
        zTable.put(0.12, 0.0478);
        zTable.put(0.13, 0.0517);
        zTable.put(0.14, 0.0557);
        zTable.put(0.15, 0.0596);
        zTable.put(0.16, 0.0636);
        zTable.put(0.17, 0.0675);
        zTable.put(0.18, 0.0714);
        zTable.put(0.19, 0.0753);
        zTable.put(0.20, 0.0793);
        zTable.put(0.21, 0.0832);
        zTable.put(0.22, 0.0871);
        zTable.put(0.23, 0.0910);
        zTable.put(0.24, 0.0948);
        zTable.put(0.25, 0.0987);
        zTable.put(0.26, 0.1026);
        zTable.put(0.27, 0.1064);
        zTable.put(0.28, 0.1103);
        zTable.put(0.29, 0.1141);
        zTable.put(0.30, 0.1179);
        zTable.put(0.31, 0.1217);
        zTable.put(0.32, 0.1255);
        zTable.put(0.33, 0.1293);
        zTable.put(0.34, 0.1331);
        zTable.put(0.35, 0.1368);
        zTable.put(0.36, 0.1406);
        zTable.put(0.37, 0.1443);
        zTable.put(0.38, 0.1480);
        zTable.put(0.39, 0.1517);
        zTable.put(0.40, 0.1554);
        zTable.put(0.41, 0.1591);
        zTable.put(0.42, 0.1628);
        zTable.put(0.43, 0.1664);
        zTable.put(0.44, 0.1700);
        zTable.put(0.45, 0.1736);
        zTable.put(0.46, 0.1772);
        zTable.put(0.47, 0.1808);
        zTable.put(0.48, 0.1844);
```

```
        zTable.put(0.49, 0.1879);
        zTable.put(0.50, 0.1915);
        zTable.put(0.51, 0.1950);
        zTable.put(0.52, 0.1985);
        zTable.put(0.53, 0.2019);
        zTable.put(0.54, 0.2054);
        zTable.put(0.55, 0.2088);
        zTable.put(0.56, 0.2123);
        zTable.put(0.57, 0.2157);
        zTable.put(0.58, 0.2190);
        zTable.put(0.59, 0.2224);
        zTable.put(0.60, 0.2257);
        zTable.put(0.61, 0.2291);
        zTable.put(0.62, 0.2324);
        zTable.put(0.63, 0.2357);
        zTable.put(0.64, 0.2389);
        zTable.put(0.65, 0.2422);
        zTable.put(0.66, 0.2454);
        zTable.put(0.67, 0.2486);
        zTable.put(0.68, 0.2517);
        zTable.put(0.69, 0.2549);
        zTable.put(0.70, 0.2580);
        zTable.put(0.71, 0.2611);
        zTable.put(0.72, 0.2642);
        zTable.put(0.73, 0.2673);
        zTable.put(0.74, 0.2704);
        zTable.put(0.75, 0.2734);
        zTable.put(0.76, 0.2764);
        zTable.put(0.77, 0.2794);
        zTable.put(0.78, 0.2823);
        zTable.put(0.79, 0.2852);
        zTable.put(0.80, 0.2881);
        zTable.put(0.81, 0.2910);
        zTable.put(0.82, 0.2939);
        zTable.put(0.83, 0.2967);
        zTable.put(0.84, 0.2995);
        zTable.put(0.85, 0.3023);
        zTable.put(0.86, 0.3051);
        zTable.put(0.87, 0.3078);
        zTable.put(0.88, 0.3106);
        zTable.put(0.89, 0.3133);
        zTable.put(0.90, 0.3159);
        zTable.put(0.91, 0.3186);
        zTable.put(0.92, 0.3212);
        zTable.put(0.93, 0.3238);
        zTable.put(0.94, 0.3264);
```

```
        zTable.put(0.95, 0.3289);
        zTable.put(0.96, 0.3315);
        zTable.put(0.97, 0.3340);
        zTable.put(0.98, 0.3365);
        zTable.put(0.99, 0.3389);
        zTable.put(1.00, 0.3413);

        zTable.put(1.01, 0.3438);
        zTable.put(1.02, 0.3461);
        zTable.put(1.03, 0.3485);
        zTable.put(1.04, 0.3508);
        zTable.put(1.05, 0.3531);
        zTable.put(1.06, 0.3554);
        zTable.put(1.07, 0.3577);
        zTable.put(1.08, 0.3599);
        zTable.put(1.09, 0.3621);
        zTable.put(1.10, 0.3643);
        zTable.put(1.11, 0.3665);
        zTable.put(1.12, 0.3686);
        zTable.put(1.13, 0.3708);
        zTable.put(1.14, 0.3729);
        zTable.put(1.15, 0.3749);
        zTable.put(1.16, 0.3770);
        zTable.put(1.17, 0.3790);
        zTable.put(1.18, 0.3810);
        zTable.put(1.19, 0.3830);
        zTable.put(1.20, 0.3849);
        zTable.put(1.21, 0.3869);
        zTable.put(1.22, 0.3888);
        zTable.put(1.23, 0.3907);
        zTable.put(1.24, 0.3925);
        zTable.put(1.25, 0.3944);
        zTable.put(1.26, 0.3962);
        zTable.put(1.27, 0.3980);
        zTable.put(1.28, 0.3997);
        zTable.put(1.29, 0.4015);
        zTable.put(1.30, 0.4032);
        zTable.put(1.31, 0.4049);
        zTable.put(1.32, 0.4066);
        zTable.put(1.33, 0.4082);
        zTable.put(1.34, 0.4099);
        zTable.put(1.35, 0.4115);
        zTable.put(1.36, 0.4131);
        zTable.put(1.37, 0.4147);
        zTable.put(1.38, 0.4162);
        zTable.put(1.39, 0.4177);
```

```
        zTable.put(1.40, 0.4192);
        zTable.put(1.41, 0.4207);
        zTable.put(1.42, 0.4222);
        zTable.put(1.43, 0.4236);
        zTable.put(1.44, 0.4251);
        zTable.put(1.45, 0.4265);
        zTable.put(1.46, 0.4279);
        zTable.put(1.47, 0.4292);
        zTable.put(1.48, 0.4306);
        zTable.put(1.49, 0.4319);
        zTable.put(1.50, 0.4332);
        zTable.put(1.51, 0.4345);
        zTable.put(1.52, 0.4357);
        zTable.put(1.53, 0.4370);
        zTable.put(1.54, 0.4382);
        zTable.put(1.55, 0.4394);
        zTable.put(1.56, 0.4406);
        zTable.put(1.57, 0.4418);
        zTable.put(1.58, 0.4429);
        zTable.put(1.59, 0.4441);
        zTable.put(1.60, 0.4452);
        zTable.put(1.61, 0.4463);
        zTable.put(1.62, 0.4474);
        zTable.put(1.63, 0.4484);
        zTable.put(1.64, 0.4495);
        zTable.put(1.65, 0.4505);
        zTable.put(1.66, 0.4515);
        zTable.put(1.67, 0.4525);
        zTable.put(1.68, 0.4535);
        zTable.put(1.69, 0.4545);
        zTable.put(1.70, 0.4554);
        zTable.put(1.71, 0.4564);
        zTable.put(1.72, 0.4573);
        zTable.put(1.73, 0.4582);
        zTable.put(1.74, 0.4591);
        zTable.put(1.75, 0.4599);
        zTable.put(1.76, 0.4608);
        zTable.put(1.77, 0.4616);
        zTable.put(1.78, 0.4625);
        zTable.put(1.79, 0.4633);
        zTable.put(1.80, 0.4641);
        zTable.put(1.81, 0.4649);
        zTable.put(1.82, 0.4656);
        zTable.put(1.83, 0.4664);
        zTable.put(1.84, 0.4671);
        zTable.put(1.85, 0.4678);
```

```
        zTable.put(1.86, 0.4686);
        zTable.put(1.87, 0.4693);
        zTable.put(1.88, 0.4699);
        zTable.put(1.89, 0.4706);
        zTable.put(1.90, 0.4713);
        zTable.put(1.91, 0.4719);
        zTable.put(1.92, 0.4726);
        zTable.put(1.93, 0.4732);
        zTable.put(1.94, 0.4738);
        zTable.put(1.95, 0.4744);
        zTable.put(1.96, 0.4750);
        zTable.put(1.97, 0.4756);
        zTable.put(1.98, 0.4761);
        zTable.put(1.99, 0.4767);

        zTable.put(2.00, 0.4772);
        zTable.put(2.01, 0.4778);
        zTable.put(2.02, 0.4783);
        zTable.put(2.03, 0.4788);
        zTable.put(2.04, 0.4793);
        zTable.put(2.05, 0.4798);
        zTable.put(2.06, 0.4803);
        zTable.put(2.07, 0.4808);
        zTable.put(2.08, 0.4812);
        zTable.put(2.09, 0.4817);
        zTable.put(2.10, 0.4821);
        zTable.put(2.11, 0.4826);
        zTable.put(2.12, 0.4830);
        zTable.put(2.13, 0.4834);
        zTable.put(2.14, 0.4838);
        zTable.put(2.15, 0.4842);
        zTable.put(2.16, 0.4846);
        zTable.put(2.17, 0.4850);
        zTable.put(2.18, 0.4854);
        zTable.put(2.19, 0.4857);
        zTable.put(2.20, 0.4861);
        zTable.put(2.21, 0.4864);
        zTable.put(2.22, 0.4868);
        zTable.put(2.23, 0.4871);
        zTable.put(2.24, 0.4875);
        zTable.put(2.25, 0.4878);
        zTable.put(2.26, 0.4881);
        zTable.put(2.27, 0.4884);
        zTable.put(2.28, 0.4887);
        zTable.put(2.29, 0.4890);
        zTable.put(2.30, 0.4893);
```

```
        zTable.put(2.31, 0.4896);
        zTable.put(2.32, 0.4898);
        zTable.put(2.33, 0.4901);
        zTable.put(2.34, 0.4904);
        zTable.put(2.35, 0.4906);
        zTable.put(2.36, 0.4909);
        zTable.put(2.37, 0.4911);
        zTable.put(2.38, 0.4913);
        zTable.put(2.39, 0.4916);
        zTable.put(2.40, 0.4918);
        zTable.put(2.41, 0.4920);
        zTable.put(2.42, 0.4922);
        zTable.put(2.43, 0.4925);
        zTable.put(2.44, 0.4927);
        zTable.put(2.45, 0.4929);
        zTable.put(2.46, 0.4931);
        zTable.put(2.47, 0.4932);
        zTable.put(2.48, 0.4934);
        zTable.put(2.49, 0.4936);
        zTable.put(2.50, 0.4938);
        zTable.put(2.51, 0.4940);
        zTable.put(2.52, 0.4941);
        zTable.put(2.53, 0.4943);
        zTable.put(2.54, 0.4945);
        zTable.put(2.55, 0.4946);
        zTable.put(2.56, 0.4948);
        zTable.put(2.57, 0.4949);
        zTable.put(2.58, 0.4951);
        zTable.put(2.59, 0.4952);
        zTable.put(2.60, 0.4953);
        zTable.put(2.61, 0.4955);
        zTable.put(2.62, 0.4956);
        zTable.put(2.63, 0.4957);
        zTable.put(2.64, 0.4959);
        zTable.put(2.65, 0.4960);
        zTable.put(2.66, 0.4961);
        zTable.put(2.67, 0.4962);
        zTable.put(2.68, 0.4963);
        zTable.put(2.69, 0.4964);
        zTable.put(2.70, 0.4965);
        zTable.put(2.71, 0.4966);
        zTable.put(2.72, 0.4967);
        zTable.put(2.73, 0.4968);
        zTable.put(2.74, 0.4969);
        zTable.put(2.75, 0.4970);
        zTable.put(2.76, 0.4971);
```

```
        zTable.put(2.77, 0.4972);
        zTable.put(2.78, 0.4973);
        zTable.put(2.79, 0.4974);
        zTable.put(2.80, 0.4974);
        zTable.put(2.81, 0.4975);
        zTable.put(2.82, 0.4976);
        zTable.put(2.83, 0.4977);
        zTable.put(2.84, 0.4977);
        zTable.put(2.85, 0.4978);
        zTable.put(2.86, 0.4979);
        zTable.put(2.87, 0.4979);
        zTable.put(2.88, 0.4980);
        zTable.put(2.89, 0.4981);
        zTable.put(2.90, 0.4981);
        zTable.put(2.91, 0.4982);
        zTable.put(2.92, 0.4982);
        zTable.put(2.93, 0.4983);
        zTable.put(2.94, 0.4984);
        zTable.put(2.95, 0.4984);
        zTable.put(2.96, 0.4985);
        zTable.put(2.97, 0.4985);
        zTable.put(2.98, 0.4986);
        zTable.put(2.99, 0.4986);
        zTable.put(3.00, 0.4987);
        zTable.put(3.01, 0.4987);
        zTable.put(3.02, 0.4987);
        zTable.put(3.03, 0.4988);
        zTable.put(3.04, 0.4988);
        zTable.put(3.05, 0.4988);
        zTable.put(3.06, 0.4989);
        zTable.put(3.07, 0.4989);
        zTable.put(3.08, 0.4989);
        zTable.put(3.09, 0.4990);
        zTable.put(3.10, 0.4990);
        zTable.put(3.11, 0.4990);
        zTable.put(3.12, 0.4991);
        zTable.put(3.13, 0.4991);
        zTable.put(3.14, 0.4991);
        zTable.put(3.15, 0.4992);
        zTable.put(3.16, 0.4992);
        zTable.put(3.17, 0.4992);
        zTable.put(3.18, 0.4992);
        zTable.put(3.19, 0.4993);
        zTable.put(3.20, 0.4993);
        zTable.put(3.21, 0.4993);
        zTable.put(3.22, 0.4993);
```

```
        zTable.put(3.23, 0.4994);
        zTable.put(3.24, 0.4994);
        zTable.put(3.25, 0.4994);
        zTable.put(3.26, 0.4994);
        zTable.put(3.27, 0.4994);
        zTable.put(3.28, 0.4995);
        zTable.put(3.29, 0.4995);

        zTable.put(3.30, 0.4995);
        zTable.put(3.31, 0.4995);
        zTable.put(3.32, 0.4995);
        zTable.put(3.33, 0.4995);
        zTable.put(3.34, 0.4995);
        zTable.put(3.35, 0.4996);
        zTable.put(3.36, 0.4996);
        zTable.put(3.37, 0.4996);
        zTable.put(3.38, 0.4996);
        zTable.put(3.39, 0.4996);
        zTable.put(3.40, 0.4996);
        zTable.put(3.41, 0.4997);
        zTable.put(3.42, 0.4997);
        zTable.put(3.43, 0.4997);
        zTable.put(3.44, 0.4997);
        zTable.put(3.45, 0.4997);
        zTable.put(3.46, 0.4997);
        zTable.put(3.47, 0.4997);
        zTable.put(3.48, 0.4998);
        zTable.put(3.49, 0.4998);
        zTable.put(3.50, 0.4998);
        zTable.put(3.51, 0.4998);
        zTable.put(3.52, 0.4998);
        zTable.put(3.53, 0.4998);
        zTable.put(3.54, 0.4998);
        zTable.put(3.55, 0.4998);
        zTable.put(3.56, 0.4998);
        zTable.put(3.57, 0.4998);
        zTable.put(3.58, 0.4998);
        zTable.put(3.59, 0.4998);

        Double result = zTable.get(z);

        if (result != null) {

            if (z < -3.59) {
                return 0.0001;
            } else if (z > 3.59) {
```

```
                return 0.9999;
            } else if (z > 0 && z < 3.59) {
                return 0.5 + result.doubleValue();
            } else if (z < 0 && z > -3.59) {
                return 0.5 - result.doubleValue();

            }
        }
        return 0.0;

    }


    /**
    * 創建並返回一個映射，該映射將整數索引映射到對應的雙精度值。
    * 通常用於創建嵌套映射結構，例如統計表。
    *
    * @param values 要映射的雙精度值序列。
    * @return 基於提供的值序列的映射。
    */
    private Map<Integer, Double> createInnerMap(double... values) {
        Map<Integer, Double> innerMap = new HashMap<>();
        for (int i = 1; i <= values.length; i++) {
            innerMap.put(i, values[i - 1]);
        }
        return innerMap;
    }


    /**
    * 查找並返回 t 表中對應於特定 alpha 值和自由度的 t 值。
    * 此方法用於雙尾和單尾 t 檢定。
    *
    * @param alpha            顯著性水平 alpha。
    * @param degreesOfFreedom 自由度。
    * @param twoTail          是否為雙尾檢定。
    * @return 對應於指定 alpha 值和自由度的 t 值。
    * @throws IllegalArgumentException 如果找不到對應的 alpha 值或自由度。
    */
    public double findtTable(double alpha, int degreesOfFreedom, boolean
        twoTail) {

        Map<Double, Map<Integer, Double>> oneTailMap = new HashMap<>();
        oneTailMap.put(0.05,
                createInnerMap(6.3138, 2.92, 2.3534, 2.1319, 2.015, 1.9432,
        1.8946, 1.8595, 1.8331, 1.8124, 1.7959,
```

```
                1.7823, 1.7709, 1.7613, 1.753, 1.7459, 1.7396,
1.7341, 1.7291, 1.7247, 1.7207, 1.7172, 1.7139,
                1.7109, 1.7081, 1.7056, 1.7033, 1.7011, 1.6991));
oneTailMap.put(0.025,
        createInnerMap(12.7065, 4.3026, 3.1824, 2.7764, 2.5706,
2.4469, 2.3646, 2.306, 2.2621, 2.2282, 2.201,
                2.1788, 2.1604, 2.1448, 2.1314, 2.1199, 2.1098,
2.1009, 2.093, 2.086, 2.0796, 2.0739, 2.0686,
                2.0639, 2.0596, 2.0555, 2.0518, 2.0484, 2.0452));
oneTailMap.put(0.01,
        createInnerMap(31.8193, 6.9646, 4.5407, 3.747, 3.365, 3.1426,
2.998, 2.8965, 2.8214, 2.7638, 2.7181,
                2.681, 2.6503, 2.6245, 2.6025, 2.5835, 2.5669,
2.5524, 2.5395, 2.528, 2.5176, 2.5083, 2.4998,
                2.4922, 2.4851, 2.4786, 2.4727, 2.4671, 2.462));
oneTailMap.put(0.005,
        createInnerMap(63.6551, 9.9247, 5.8408, 4.6041, 4.0322,
3.7074, 3.4995, 3.3554, 3.2498, 3.1693, 3.1058,
                3.0545, 3.0123, 2.9768, 2.9467, 2.9208, 2.8983,
2.8784, 2.8609, 2.8454, 2.8314, 2.8188, 2.8073,
                2.797, 2.7874, 2.7787, 2.7707, 2.7633, 2.7564));
oneTailMap.put(0.0025,
        createInnerMap(127.3447, 14.0887, 7.4534, 5.5976, 4.7734,
4.3168, 4.0294, 3.8325, 3.6896, 3.5814,
                3.4966, 3.4284, 3.3725, 3.3257, 3.286, 3.252, 3.2224,
3.1966, 3.1737, 3.1534, 3.1352, 3.1188,
                3.104, 3.0905, 3.0782, 3.0669, 3.0565, 3.0469,
3.038));
oneTailMap.put(0.001,
        createInnerMap(318.493, 22.3276, 10.2145, 7.1732, 5.8934,
5.2076, 4.7852, 4.5008, 4.2969, 4.1437,
                4.0247, 3.9296, 3.852, 3.7874, 3.7328, 3.6861,
3.6458, 3.6105, 3.5794, 3.5518, 3.5272, 3.505,
                3.485, 3.4668, 3.4502, 3.435, 3.4211, 3.4082,
3.3962));
oneTailMap.put(0.0005,
        createInnerMap(636.045, 31.5989, 12.9242, 8.6103, 6.8688,
5.9589, 5.4079, 5.0414, 4.7809, 4.5869,
                4.4369, 4.3178, 4.2208, 4.1404, 4.0728, 4.015,
3.9651, 3.9216, 3.8834, 3.8495, 3.8193, 3.7921,
                3.7676, 3.7454, 3.7251, 3.7067, 3.6896, 3.6739,
3.6594));

Map<Double, Map<Integer, Double>> twoTailMap = new HashMap<>();
twoTailMap.put(0.1,
        createInnerMap(6.3138, 2.92, 2.3534, 2.1319, 2.015, 1.9432,
1.8946, 1.8595, 1.8331, 1.8124, 1.7959,
                1.7823, 1.7709, 1.7613, 1.753, 1.7459, 1.7396,
1.7341, 1.7291, 1.7247, 1.7207, 1.7172, 1.7139,
                1.7109, 1.7081, 1.7056, 1.7033, 1.7011, 1.6991));
twoTailMap.put(0.05,
```

```
        createInnerMap(12.7065, 4.3026, 3.1824, 2.7764, 2.5706,
2.4469, 2.3646, 2.306, 2.2621, 2.2282, 2.201,
                2.1788, 2.1604, 2.1448, 2.1314, 2.1199, 2.1098,
2.1009, 2.093, 2.086, 2.0796, 2.0739, 2.0686,
                2.0639, 2.0596, 2.0555, 2.0518, 2.0484, 2.0452));
twoTailMap.put(0.02,
        createInnerMap(31.8193, 6.9646, 4.5407, 3.747, 3.365, 3.1426,
2.998, 2.8965, 2.8214, 2.7638, 2.7181,
                2.681, 2.6503, 2.6245, 2.6025, 2.5835, 2.5669,
2.5524, 2.5395, 2.528, 2.5176, 2.5083, 2.4998,
                2.4922, 2.4851, 2.4786, 2.4727, 2.4671, 2.462));
twoTailMap.put(0.01,
        createInnerMap(63.6551, 9.9247, 5.8408, 4.6041, 4.0322,
3.7074, 3.4995, 3.3554, 3.2498, 3.1693, 3.1058,
                3.0545, 3.0123, 2.9768, 2.9467, 2.9208, 2.8983,
2.8784, 2.8609, 2.8454, 2.8314, 2.8188, 2.8073,
                2.797, 2.7874, 2.7787, 2.7707, 2.7633, 2.7564));
twoTailMap.put(0.005,
        createInnerMap(127.3447, 14.0887, 7.4534, 5.5976, 4.7734,
4.3168, 4.0294, 3.8325, 3.6896, 3.5814,
                3.4966, 3.4284, 3.3725, 3.3257, 3.286, 3.252, 3.2224,
3.1966, 3.1737, 3.1534, 3.1352, 3.1188,
                3.104, 3.0905, 3.0782, 3.0669, 3.0565, 3.0469,
3.038));
twoTailMap.put(0.002,
        createInnerMap(318.493, 22.3276, 10.2145, 7.1732, 5.8934,
5.2076, 4.7852, 4.5008, 4.2969, 4.1437,
                4.0247, 3.9296, 3.852, 3.7874, 3.7328, 3.6861,
3.6458, 3.6105, 3.5794, 3.5518, 3.5272, 3.505,
                3.485, 3.4668, 3.4502, 3.435, 3.4211, 3.4082,
3.3962));
twoTailMap.put(0.001,
        createInnerMap(636.045, 31.5989, 12.9242, 8.6103, 6.8688,
5.9589, 5.4079, 5.0414, 4.7809, 4.5869,
                4.4369, 4.3178, 4.2208, 4.1404, 4.0728, 4.015,
3.9651, 3.9216, 3.8834, 3.8495, 3.8193, 3.7921,
                3.7676, 3.7454, 3.7251, 3.7067, 3.6896, 3.6739,
3.6594));

Map<Double, Map<Integer, Double>> selectedMap = twoTail ? twoTailMap
: oneTailMap;
if (selectedMap.containsKey(alpha)) {
    // Get the inner map for the specific alpha value
    Map<Integer, Double> innerMap = selectedMap.get(alpha);

    // Check if the 'degreesOfFreedom' value exists in the inner map
    if (innerMap.containsKey(degreesOfFreedom)) {
        // Get the value corresponding to the alpha and degrees of
freedom
        double result = innerMap.get(degreesOfFreedom);
```

```
                // Now 'result' contains the desired value
                return result;
            } else {
                // Handle case where degrees of freedom value is not found
                throw new IllegalArgumentException("Degrees of freedom not
        found for alpha: " + alpha);
            }
        } else {
            // Handle case where alpha value is not found
            throw new IllegalArgumentException("Alpha value not found: " +
        alpha);
        }
    }

}
```

Main 類別說明

Main 類別說明

1.首先，先建立一個ArrayList，用來存放DescriptiveStatistics物件，因為有可能會使用到兩種以上的資料集，所以用ArrayList來存放。

2.讓使用者選擇數據來源，有三種選擇，分別是CSV、手動輸入、使用範例。

- CSV：使用者輸入CSV檔的路徑，程式會自動讀取CSV檔的數據，並將數據存放到ArrayList中。
- 手動輸入：使用者輸入數據欄位名、數據個數、數據，程式會自動將數據存放到ArrayList中。
- 使用範例：程式會自動將範例數據存放到ArrayList中。

3.接著，將ArrayList中的數據進行描述性統計分析，包括平均值、中位數、標準偏差等。(最基礎的DescriptiveStatistics class)

4.接著，讓使用者選擇要進行的分析，有兩種進階選擇，分別是ANOVA、Regression

- ANOVA：讓使用者選擇要進行ANOVA的資料，並計算ANOVA的F值。
    1. 先用description()說明ANOVA的概念和公式
    2. 接著，列出所有的資料集，讓使用者選擇要進行ANOVA的資料集，並計算ANOVA的F值。

3. 再來，用summary()提供ANOVA分析的摘要。
4. 最後，用explain()提供ANOVA分析的基本概念和使用方法的解釋。

- Regression：讓使用者選擇要進行Regression的資料，並計算Regression的斜率和截距。
  1. 先用description()說明Regression的概念和公式
  2. 接著，列出所有的資料集，讓使用者選擇要進行Regression的資料集，並計算Regression的斜率和截距。
  3. 再來，用summary()提供Regression分析的摘要。
  4. 用explain()提供Regression分析的基本概念和使用方法的解釋。
  5. 最後，使用predict()來預測給定x值的y值。
- HypothesisTest：讓使用者選擇要進行HypothesisTest的資料，並計算HypothesisTest的t值和p值。
  1. 讓使用者選擇要進行的分析的數據

  2.讓使用者填入假設檢定與方向
  3. 如果資料為小於30的陣列，則進行t檢定，並且用tAnalysis()來分析結果
  4. 如果資料為大於30的陣列，則進行z檢定，並且用analysis()來分析結果

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Scanner;
import java.io.IOException;



public class Main {
        public static void main(String[] args) {

        //有可能會使用到兩種以上的資料集
        //所以用ArrayList來存放

        Scanner sc = new Scanner(System.in);
        ArrayList<DescriptiveStatistics> statsList = new ArrayList<>();

        System.out.println("請選擇數據來源 ? 1:CSV 2:手動輸入 3.使用範例");
        int choice1 = sc.nextInt();

        switch(choice1){
```

```
        case 1:
          System.out.println("輸入csv檔: ");
          String filePath = sc.next();
          ReadFile.readDataToStatsList(filePath, statsList);
          break;
        case 2:
          while (true) {

                System.out.println("數據欄位名: ");
                String name = sc.next();

                System.out.println("請問有幾個數據: ");
                int n = sc.nextInt();
                double[] data = new double[n];

                System.out.println("輸入數據: ");
                for (int i = 0; i < n; i++) {
                    data[i] = sc.nextDouble();
                }

                DescriptiveStatistics stats = new
    DescriptiveStatistics(data, name);
                statsList.add(stats);

                System.out.println("Do you want to enter another set of
    data? (Y/N)");
                String answer = sc.next();

                if (answer.equalsIgnoreCase("N")) {
                    break;
                }
            }
          break;
        case 3:
          System.out.println("為您提供以下示範");
          double[] data1 = {1, 2, 3, 4, 5};
          double[] data2 = {2, 5 , 6, 7, 8};
          double[] data3 = {3, 4, 5, 6, 7};
          statsList.add(new DescriptiveStatistics(data1, "data1"));
          statsList.add(new DescriptiveStatistics(data2, "data2"));
          statsList.add(new DescriptiveStatistics(data3, "data3"));
          System.out.println("data1: " + Arrays.toString(data1));
          System.out.println("data2: " + Arrays.toString(data2));
          System.out.println("data3: " + Arrays.toString(data3));
          System.out.println("");
          System.out.println("-------------------------");
          System.out.println("");
```

```
            break;
        }

        System.out.println("基礎數據分析");
        System.out.println(statsList.get(0).description());
        for (DescriptiveStatistics stats : statsList) {
            System.out.println(stats.summary());
            System.out.println();
        }

        System.out.println(statsList.get(0).explain());
        System.out.println("-------------------------");
        System.out.println("");

        System.out.println("選擇要進行的分析：");
        System.out.println("1. HypothesisTest");
        System.out.println("2. ANOVA");
        System.out.println("3. Regression");
        int choice = sc.nextInt();
        System.out.println("");

        switch(choice){
            case 1:
              HypothesisTest hy_instruction = new HypothesisTest(null,
        "name");
                    System.out.println("歡迎使用「假設檢定」功能，請問有需要為您進行
        假設檢定之概念講解嗎？(輸入y/n)");

                    if (sc.next().equals("y")) {
                        hy_instruction.instruction();
                    }

                    for (int i = 0; i < statsList.size(); i++) {
                        System.out.println((i + 1) + ". " +
        statsList.get(i).getName());
                    }
                    System.out.println("請選擇要做假設檢定的資料(輸入對應數字)：");
                    int index_hy = sc.nextInt() - 1;
                    System.out.println("您選擇的資料為：" +
        statsList.get(index_hy).getName());
                    System.out.println("-----------------------------------
        ---");

                    HypothesisTest hy = new
        HypothesisTest(statsList.get(index_hy).getData(),
        statsList.get(index_hy).getName());
                    System.out.println("第一個步驟為「確立虛無假設」，請問有需要為您
        進行該步驟之概念講解嗎？(輸入y/n)");
```

```
                if (sc.next().equals("y")) {
                    hy_instruction.instruction1();
                }

                System.out.println("輸入假設之均值：");
                double mean = sc.nextDouble();
                System.out.println("輸入假設之方向「左尾、右尾、雙尾」：");
                String direction = sc.next();
                hy.setNullHypo(mean, direction);
                System.out.println("-----------------------------");

                System.out.println("第二個步驟為「確立檢定統計量」，請問有需要為您進行該步驟之概念講解嗎？(輸入y/n)");
                if (sc.next().equals("y")) {
                        hy_instruction.instruction2();
                      }
                //如果數據>30，則使用z檢定
                if(hy.getData().length > 30) {
                  System.out.println("數據大於30筆，適用的是Z分析");
                    System.out.println("是否得知或遇輸入母體體標準差？(輸入y/n)");
                    String answer_hy = sc.next();

                    double std;
            if(answer_hy.equals("y")){
                System.out.println("請輸入母體標準差：");
                std = sc.nextDouble();
            }else{
                System.out.println("將以樣本標準差替代");
                std = hy.standardDeviation();
            }

                    System.out.println("-----------------------------");
                    double pValue = hy.p_Population(std);
                    System.out.println("P 值: " + pValue);
                    hy.analysis(pValue);
                }else{
                  System.out.println("數據小於30筆，適用的是T分析");
                  System.out.println("-----------------------------");
                  double tValue = hy.t_Sample();
              System.out.println("T 值: " + tValue);
              hy.tAnalysis(tValue);
                }
                hy_instruction.instruction3();
                System.out.println("");
          break;
```

```
    case 2:
      //ANOVA-1 解釋
      System.out.println("要聽一下Anova的概念嗎？(Y/N)");
      String answera1 = sc.next();
      while(answera1.equalsIgnoreCase("Y")){
        System.out.println("概念講解");
        double[][] dataForAnova = {{1, 2, 3, 4, 5}, {2, 5 , 6, 7,
8}};
        Anova anova = new Anova(dataForAnova, "ANOVA Test");
        System.out.println(anova.description());
        System.out.println("要在聽一次嗎？(Y/N)");
        String answera2 = sc.next();
        if (answera2.equalsIgnoreCase("N")) {
            System.out.println("好的，那我們進入實戰環節！");
            System.out.println("");
            break;
        }
      }

      //ANOVA-2 實戰
      System.out.println("請選擇要做ANOVA的資料(輸入對應數字，輸入-1結
束)：");
      for (int i = 0; i < statsList.size(); i++) {
          System.out.println((i + 1) + ". " +
statsList.get(i).getName());
      }

      ArrayList<Integer> indexList = new ArrayList<>();
      while(true) {
          int index = sc.nextInt() - 1;
          if (index == -2) {
              break;
          } else if (index >= 0 && index < statsList.size()) {
              indexList.add(index);
              System.out.println("已選擇 " +
statsList.get(index).getName());
          } else {
              System.out.println("無效的索引，請重新輸入");
          }
      }

      double[][] dataForAnova = new double[indexList.size()][];
      for (int i = 0; i < indexList.size(); i++) {
          DescriptiveStatistics stats =
statsList.get(indexList.get(i));
          dataForAnova[i] = stats.getData();
      }
```

```
            if (dataForAnova.length > 1) {
                Anova anova = new Anova(dataForAnova, "ANOVA Test");
                System.out.println(anova.summary());
                System.out.println("F 值: " + anova.calculateFValue());
                System.out.println(anova.explain());
            } else {
                System.out.println("至少需要選擇兩組數據進行ANOVA分析");
            }
            break;
        case 3:
            //提供線性回歸的解釋
            System.out.println("要聽一下線性回歸的概念嗎？(Y/N)");
            String answer = sc.next();
            while(answer.equalsIgnoreCase("Y")){
              System.out.println("概念講解");
              double[] xData = {1, 2, 3, 4, 5};
              double[] yData = {2, 5 , 6, 7, 8};
              LinearRegression reg = new LinearRegression(xData, yData, "線
性回歸示範");
              System.out.println(reg.description());
              System.out.println("要在聽一次嗎？(Y/N)");
              String answerl1 = sc.next();
              if (answerl1.equalsIgnoreCase("N")) {
                  System.out.println("好的，那我們進入實戰環節！");
                  System.out.println("");
                  break;
              }
            }
            for (int i = 0; i < statsList.size(); i++) {
                System.out.println((i + 1) + ". " +
statsList.get(i).getName());
            }
            System.out.println("請選擇要做迴歸分析的因變數：");
            int index1 = sc.nextInt() - 1;
            System.out.println("請選擇要做迴歸分析的自變數：");
            int index2 = sc.nextInt() - 1;
            String regName = statsList.get(index1).getName() + " vs. " +
statsList.get(index2).getName();
            LinearRegression reg = new
LinearRegression(statsList.get(index1).getData(),
statsList.get(index2).getData(), regName);

            System.out.println("");
            System.out.println("迴歸分析結果：");
            System.out.println(reg.summary());
            System.out.println(reg.explain());
            System.out.println("-------------------------");
```

```
            System.out.println("");

            System.out.println("請問要預測 y(應變數) 值嗎？(Y/N)");
            String answerl2 = sc.next();
            while(answerl2.equalsIgnoreCase("Y")){
              System.out.println("請輸入 y 值：");
              double x = sc.nextDouble();
              System.out.println("預測的 x 值為：" + reg.predict(x));
              System.out.println("要在預測一次嗎？(Y/N)");
              String answerl3 = sc.next();
              if (answerl3.equalsIgnoreCase("N")) {
                  System.out.println("好的，再見！");
                  break;
              }
            }
          }
        sc.close();
    }
}
```

Example

1:使用範例+回歸分析

```
請選擇數據來源 ? 1:CSV 2:手動輸入 3.使用範例
3
為您提供以下示範
data1: [1.0, 2.0, 3.0, 4.0, 5.0]
data2: [2.0, 5.0, 6.0, 7.0, 8.0]
data3: [3.0, 4.0, 5.0, 6.0, 7.0]

--------------------------


基礎數據分析
以下將把每個數據集的數據進行描述性統計分析，包括平均值、中位數、標準(偏)差、樣本大小、母體方
        差和母體標準偏差。
數據名稱: data1
平均值: 3.0
中位數: 3.0
標準偏差: 1.4142135623730951
樣本大小: 5
母體方差: 2.0
母體標準偏差: 1.4142135623730951

數據名稱: data2
平均值: 5.6
中位數: 6.0
標準偏差: 2.0591260281974
樣本大小: 5
母體方差: 4.24
母體標準偏差: 2.0591260281974

數據名稱: data3
平均值: 5.0
中位數: 5.0
標準偏差: 1.4142135623730951
樣本大小: 5
母體方差: 2.0
母體標準偏差: 1.4142135623730951

以下為您講解公式
平均值 = Σx / n
```

```
中位數 = (x[n/2] + x[n/2+1]) / 2
標準偏差 = √(Σ(x - x平均值)^2 / n)
母體方差 = Σ(x - x平均值)^2 / n
母體標準偏差 = √(Σ(x - x平均值)^2 / n)

--------------------------

選擇要進行的分析：
1. Hypothesis Testing
2. ANOVA
3. Regression
3

要聽一下線性回歸的概念嗎？(Y/N)
Y
概念講解
以下是簡單線性回歸的概念
=> involves one independent variable and one dependent variable.
Suppose: y = B0 + B1x + ε , then E(y) = B0 + B1x , ε~NID(0 , σ^2)
Sampling and Fitted : ŷ = b0 + b1x
Estimated: ŷ = b0 + b1x -----> E(y) = B0 + B1x
                          b0 -----> B0  ;  b1 -----> B1
         (redisual) e = y - ŷ  -----> ε = y - E(y)
                                by min Σ(y - ŷ)^2 = min Σ(y - b0 + b1x)^2
                                由 ∂SSE / ∂b0 = 0  and  ∂SSE / ∂b1 = 0
                                得 b1 = Σ(x - x̄)^2 (y - ȳ)^2 / Σ(x - x̄)^2
                                   b0 = ȳ - b1x̄
要在聽一次嗎？(Y/N)
N
好的，那我們進入實戰環節！

1. data1
2. data2
3. data3
請選擇要做迴歸分析的因變數：
1
請選擇要做迴歸分析的自變數：
2

迴歸分析結果：
線性回歸模型 - data1 vs. data2
斜率 (beta0): 1.4
截距 (beta1): 1.4000000000000004
本數據的迴歸模型是: ŷ = 1.4x + 1.4000000000000004
Statitical (True) Model is: y = f(x) + ε
其中： y = 應變數  ; x = 自變數
```

```
Statitical (True) Model ----->  Fitted model , eg: ŷ = b1x + b0
利用 data 來 尋找Y和X的關係（不一定是因果關係）


--------------------------


請問要預測 y(應變數) 值嗎？(Y/N)
Y
請輸入 y 值：
100
預測的 x 值為：141.4
要在預測一次嗎？(Y/N)
N
好的，再見！
```

2.使用csv檔+ANOVA

```
請選擇數據來源 ？ 1:CSV 2:手動輸入 3.使用範例
1
輸入csv檔:
datak.csv
基礎數據分析
以下將把每個數據集的數據進行描述性統計分析，包括平均值、中位數、標準(偏)差、樣本大小、母體方
        差和母體標準偏差。
數據名稱: score
平均值: 79.21241736022337
中位數: 79.37917081784681
標準偏差: 11.795080357828915
樣本大小: 100
母體方差: 139.1239206476415
母體標準偏差: 11.795080357828915

數據名稱: course_length
平均值: 17.5
中位數: 17.5
標準偏差: 1.118033988749895
樣本大小: 100
母體方差: 1.25
母體標準偏差: 1.118033988749895

以下為您講解公式
平均值 = Σx / n
中位數 = (x[n/2] + x[n/2+1]) / 2
標準偏差 = √(Σ(x - x平均值)^2 / n)
母體方差 = Σ(x - x平均值)^2 / n
母體標準偏差 = √(Σ(x - x平均值)^2 / n)
```

```
--------------------------

選擇要進行的分析：
1. 離開
2. ANOVA
3. Regression
2

要聽一下Anova的概念嗎？(Y/N)
Y
概念講解
ANOVA 通過計算 F 值來測試組間差異的顯著性。F 值是組間均方 (MSB) 和組內均方 (MSW) 的比
        率，其中 MSB = 組間平方和(SSB) / 組間自由度(dfBetween)，MSW = 組內平方和
        (SSW) / 組內自由度(dfWithin)。高 F 值通常表明組間變異顯著大於組內變異，從而指
        示組間存在顯著差異。
要在聽一次嗎？(Y/N)
N
好的，那我們進入實戰環節！

請選擇要做ANOVA的資料(輸入對應數字，輸入-1結束)：
1. score
2. course_length
1
已選擇 score
2
已選擇 course_length
-1
ANOVA 分析 - ANOVA Test
總體平方和 (SST): 204458.51488688402
組間平方和 (SSB): 190421.12282211994
組內平方和 (SSW): 14037.392064764077
F 值: 2685.9250026520804
ANOVA (分析變異) 用於比較三個或更多組的平均數是否有顯著差異。它將總變異分解為組間變異和組內
        變異，並通過 F 統計量來評估組間變異是否顯著大於組內變異。
```

3.csv+HypothesisTest

```
請選擇數據來源 ? 1:CSV 2:手動輸入 3.使用範例
1
輸入csv檔:
z_data.csv
基礎數據分析
以下將把每個數據集的數據進行描述性統計分析，包括平均值、中位數、標準(偏)差、樣本大小、母體方
        差和母體標準偏差。
數據名稱: Value
```

平均值: 5.140559272313099
中位數: 5.149495498428896
標準偏差: 1.125522920554846
樣本大小: 50
母體方差: 1.2668018446943103
母體標準偏差: 1.125522920554846

以下為您講解公式
平均值 = Σx / n
中位數 = (x[n/2] + x[n/2+1]) / 2
標準偏差 = √(Σ(x - x平均值)^2 / n)
母體方差 = Σ(x - x平均值)^2 / n
母體標準偏差 = √(Σ(x - x平均值)^2 / n)

選擇要進行的分析：
1. HypothesisTest
2. ANOVA
3. Regression
1

歡迎使用「假設檢定」功能，請問有需要為您進行假設檢定之概念講解嗎？(輸入y/n)
y
假說檢定（英語：hypothesis testing）是推論統計中用於檢定現有數據是否足以支持特定假設的方法。
一旦能估計未知母數，就會希望根據結果對未知的真正母數值做出適當的推論。
欲檢定統計上假設的正確性的為虛無假說，虛無假說通常由研究者決定，反映研究者對未知母數的看法。
相對於虛無假說的其他有關母數之論述是對立假說，它通常反應了執行檢定的研究者對母數可能數值的另一種（對立的）看法

1. Value
請選擇要做假設檢定的資料(輸入對應數字)：
1
您選擇的資料為：Value

第一個步驟為「確立虛無假設」，請問有需要為您進行該步驟之概念講解嗎？(輸入y/n)
y
以下說明此步驟之意義：
在檢定之前，我們需要針對情境進行假設。其中包含「虛無假設」以及「對立假設」兩種情況：
「虛無假設」對母體參數提出一個主張，假設此主張為「真實」（除非能證明此主張非真！）。
對立假設是相對於虛無假設所提出的另一個不同(相反）的假設或主張，必須有足夠的證據，才能說明此主張為真。
檢定方式又因虛無假設之不同，分為三種檢定：
雙尾檢定： H0 : θ=θ0； H1 : θ≠θ0

左尾檢定： H0 : θ≧θ0 ； H1 : θ<θ0

右尾檢定： H0 : θ≦θ0 ； H1 : θ>θ0
輸入假設之均值：
5
輸入假設之方向「左尾、右尾、雙尾」：
左尾
H0: μ>=5.0 H1: μ<5.0

第二個步驟為「確立檢定統計量」，請問有需要為您進行該步驟之概念講解嗎？(輸入y/n)
y
檢定統計量是由樣本所算出來的一個值，用來決定是否接受或拒絕 H0。

不同參數的假設檢定，使用不同的檢定統量。常用的檢定統計量有：Z, t。
在母體平均數的假設檢定裡，不同的情形下使用不同的檢定統計量。
若母體變異數已知，則使用z = (xbar-μ)/(σ-√n)
，其中xbar代表檢定量、μ代表母體平均數、σ代表母體標準差、n代表母體樣本數量若母體變異數未知，
但樣本數超過30，可將樣本視為一母體，同樣使用z = (xbar-μ)/(σ-√n)
若母體變異數未知，但樣本數小於30，則同樣使用t = (xbar-μ)/(s-√n)，其中s代表樣本標準差

數據大於30筆，適用的是Z分析
是否得知或遇輸入母體體標準差？(輸入y/n)
y
請輸入母體標準差：
1.125522920554846

P 值: 0.0
在此情境中，由於p-value(0.0)小於alpha(0.05)，可知此資料無法拒絕H0之假設
研究人員必須決定一個決策法則，以瞭解何時『不拒絕』 H0 ；何時拒絕 H0 。
一般我們說『不拒絕』H0 ，而不說接受 H0 ，因為我們只是沒有足夠證據拒絕，而不是接受。
決策法則通常是決定一個不拒絕域 (NonrejectionRegion, 或稱接受域) 與拒絕域 (Rejection
Region)。
當檢定統計量落入不拒絕域： 『不拒絕』 H0 ！
當檢定統計量落入拒絕域：拒絕 H0 ；接受 H1
接受域與拒絕域的接點，稱為臨界點(Critical Point)。
臨界值的決定，是根據顯著水準α並利用機率分配計算而得，分成三種形式：
雙尾檢定 (落在兩邊拒絕)、右尾檢定 (落在右邊拒絕)、左尾檢定 (落在左邊拒絕)
在z檢定時，我們也會經由計算其p-value並與alpha值比較，並得到結果
一般來說，當p-value小於alpha，則拒絕H0

其他資料

Github: https://github.com/blingblingdong/Java_statistic

Web App: https://javashinyapp.fly.dev

Presentation: https://java.lsyverycute.com