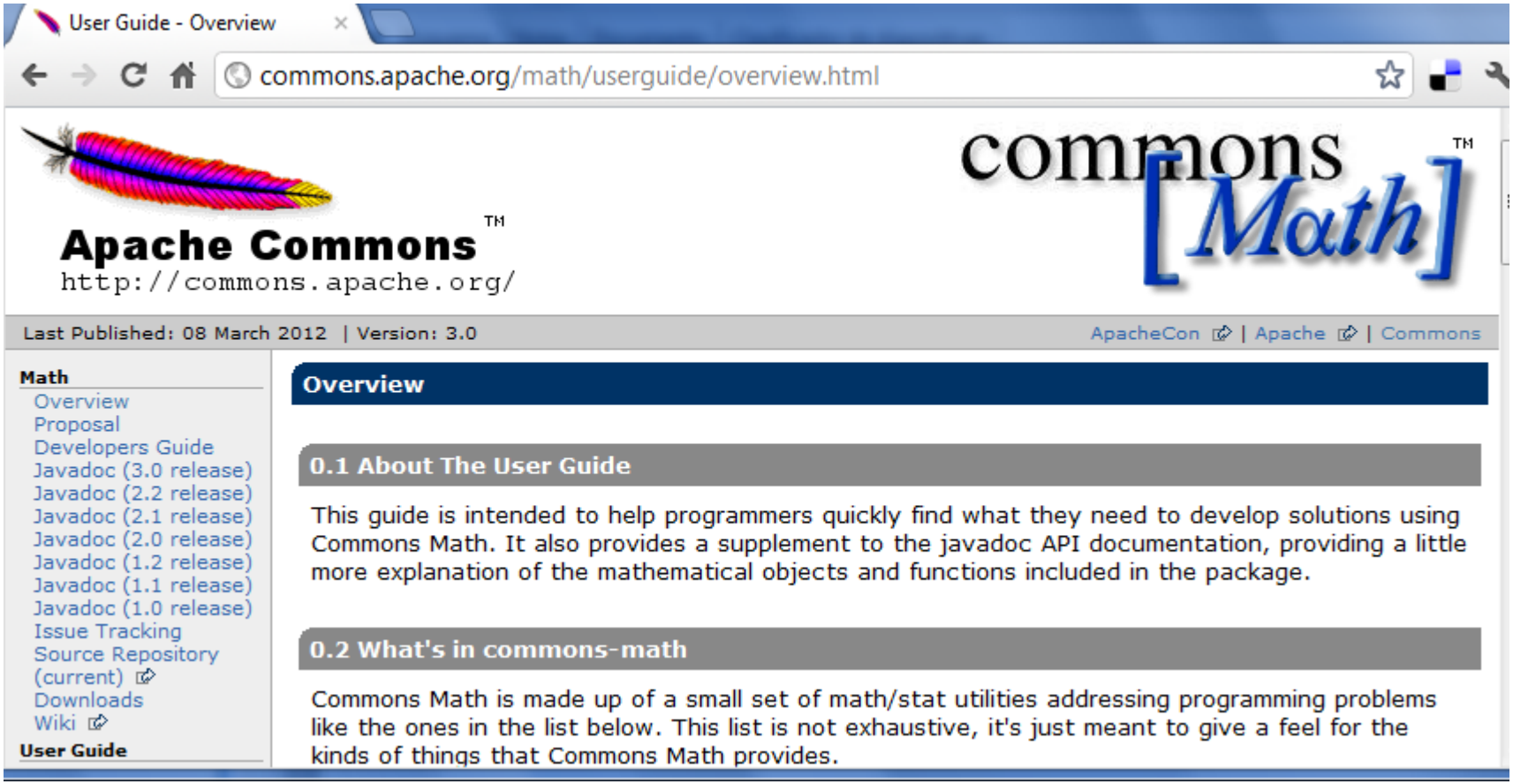
UNIVERSIDAD DE ALMERÍA

Grado en Ingeniería Informática
Metodología de la programación
2013

Utilización de bibliotecas (librerías)

- Apache Commons Math


User Guide - Overview
commons.apache.org/math/userguide/overview.html
Apache Commons™
http://commons.apache.org/
commons [Math]™
Last Published: 08 March 2012 | Version: 3.0
ApacheCon | Apache | Commons
Math
Overview
Proposal
Developers Guide
Javadoc (3.0 release)
Javadoc (2.2 release)
Javadoc (2.1 release)
Javadoc (2.0 release)
Javadoc (1.2 release)
Javadoc (1.1 release)
Javadoc (1.0 release)
Issue Tracking
Source Repository (current)
Downloads
Wiki
User Guide
Overview
0.1 About The User Guide
This guide is intended to help programmers quickly find what they need to develop solutions using Commons Math. It also provides a supplement to the javadoc API documentation, providing a little more explanation of the mathematical objects and functions included in the package.
0.2 What's in commons-math
Commons Math is made up of a small set of math/stat utilities addressing programming problems like the ones in the list below. This list is not exhaustive, it's just meant to give a feel for the kinds of things that Commons Math provides.

0.2 What's in commons-math

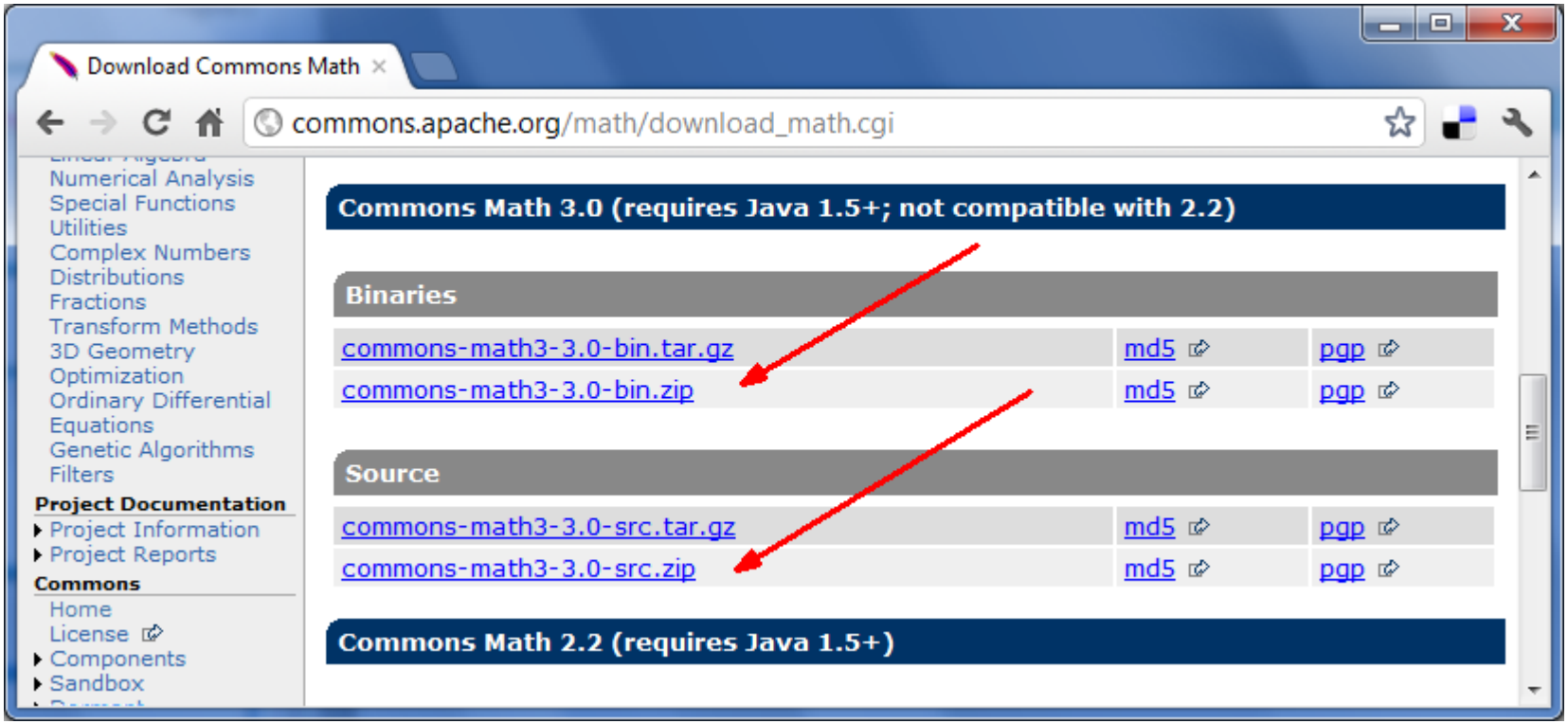
Commons Math is made up of a small set of math/stat utilities addressing programming problems like the ones in the list below. This list is not exhaustive, it's just meant to give a feel for the kinds of things that Commons Math provides.

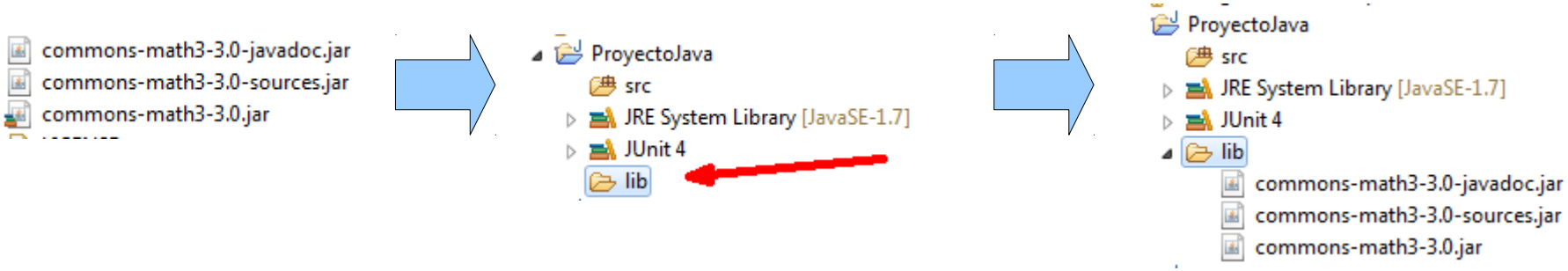
- Computing means, variances and other summary statistics for a list of numbers
- Fitting a line to a set of data points using linear regression
- Finding a smooth curve that passes through a collection of points (interpolation)
- Fitting a parametric model to a set of measurements using least-squares methods
- Solving equations involving real-valued functions (i.e. root-finding)
- Solving systems of linear equations
- Solving Ordinary Differential Equations
- Minimizing multi-dimensional functions
- Generating random numbers with more restrictions (e.g distribution, range) than what is possible using the JDK
- Generating random samples and/or datasets that are "like" the data in an input file
- Performing statistical significance tests
- Miscellaneous mathematical functions such as factorials, binomial coefficients and "special functions" (e.g. gamma, beta functions)

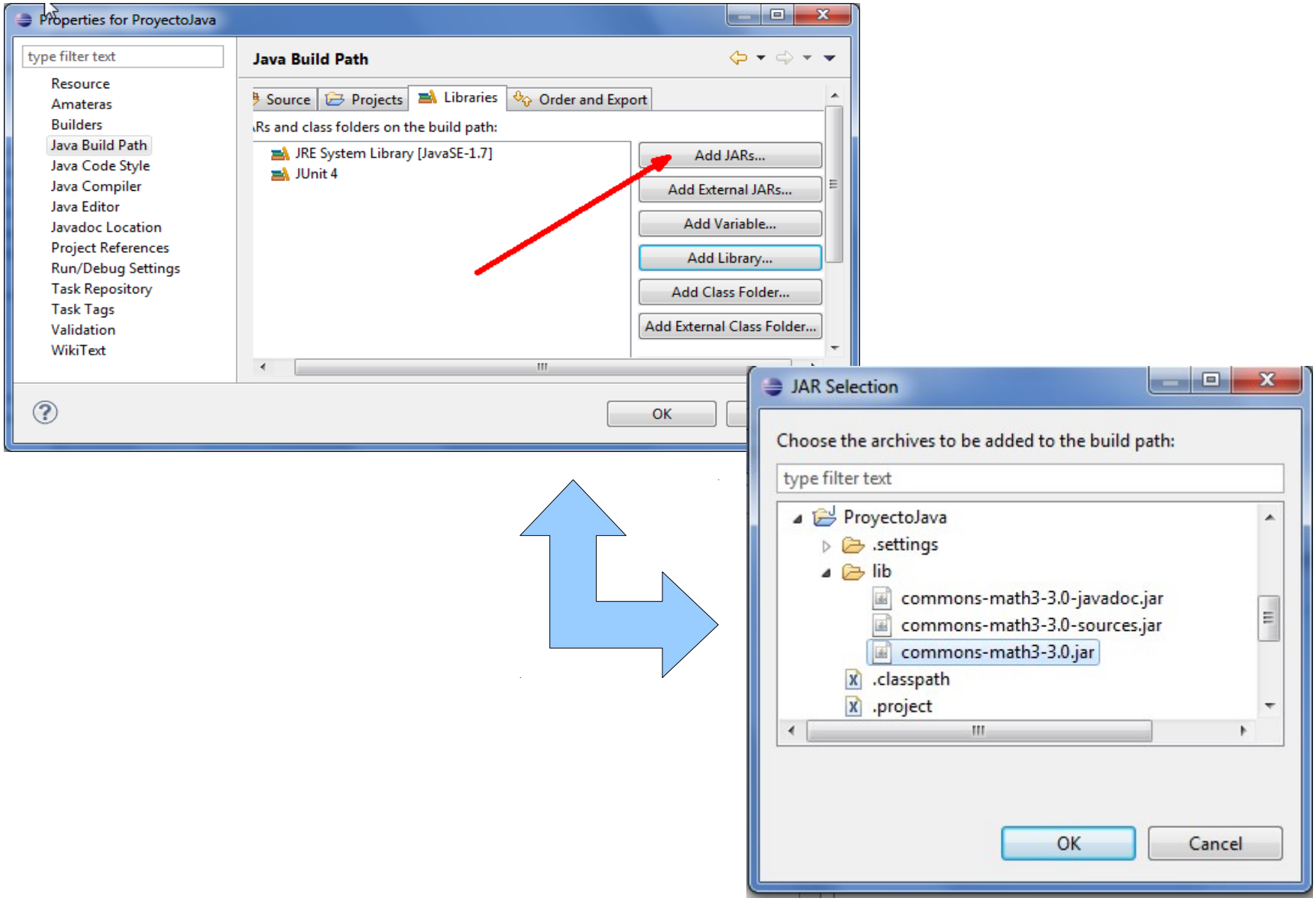
Commons Math is divided into fourteen subpackages, based on functionality provided.

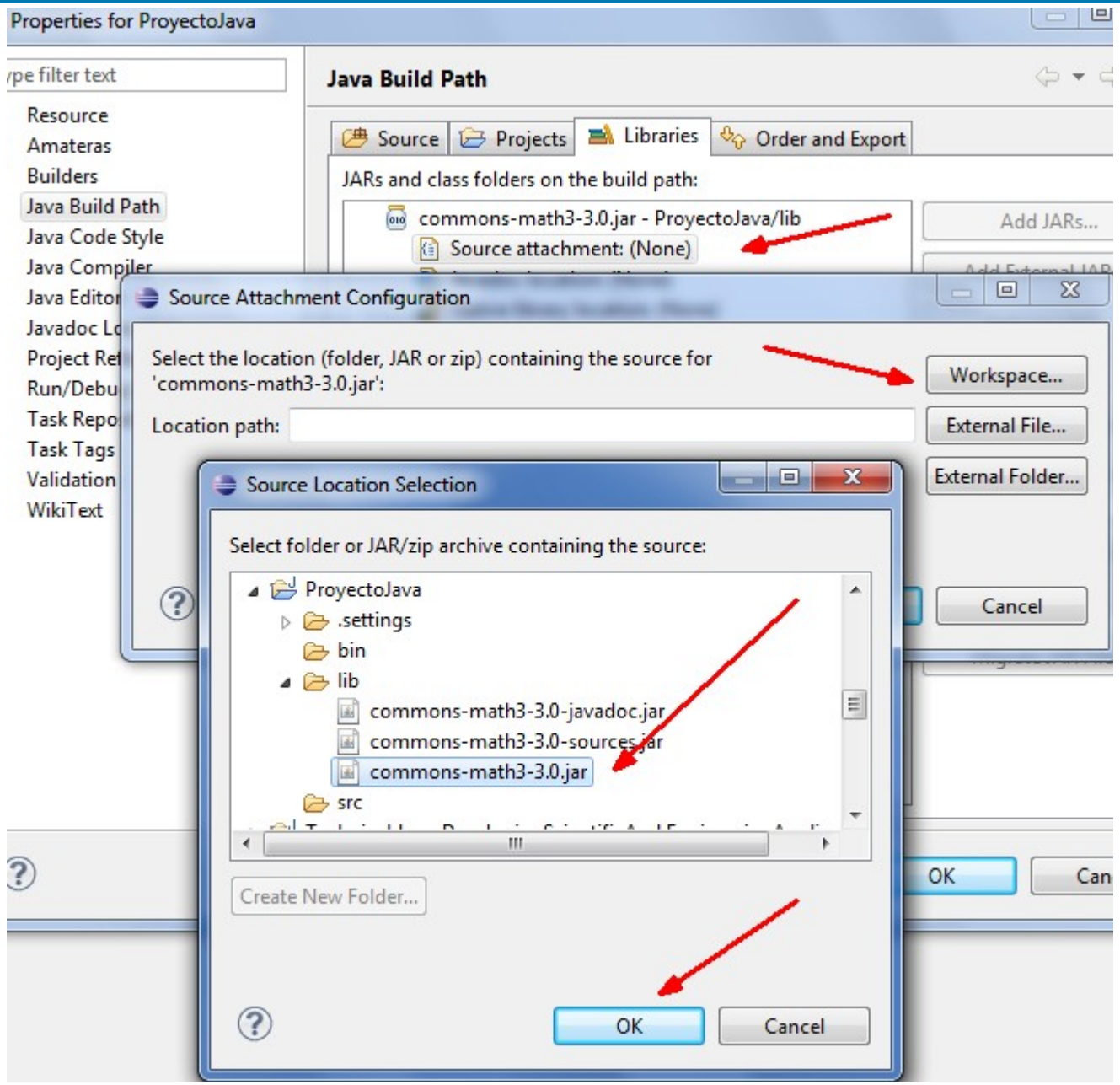
1. org.apache.commons.math3.stat - statistics, statistical tests
2. org.apache.commons.math3.analysis - rootfinding, integration, interpolation, polynomials
3. org.apache.commons.math3.random - random numbers, strings and data generation
4. org.apache.commons.math3.special - special functions (Gamma, Beta)
5. org.apache.commons.math3.linear - matrices, solving linear systems
6. org.apache.commons.math3.util - common math/stat functions extending java.lang.Math
7. org.apache.commons.math3.complex - complex numbers
8. org.apache.commons.math3.distribution - probability distributions
9. org.apache.commons.math3.fraction - rational numbers
10. org.apache.commons.math3.transform - transform methods (Fast Fourier)
11. org.apache.commons.math3.geometry - geometry (Euclidean spaces and Binary Space Partitioning)
12. org.apache.commons.math3.optimization - function maximization or minimization
13. org.apache.commons.math3.ode - Ordinary Differential Equations integration
14. org.apache.commons.math3.genetics - Genetic Algorithms

Package javadocs are here


Download Commons Math
commons.apache.org/math/download_math.cgi
Numerical Analysis
Special Functions
Utilities
Complex Numbers
Distributions
Fractions
Transform Methods
3D Geometry
Optimization
Ordinary Differential Equations
Genetic Algorithms
Filters
Project Documentation
Project Information
Project Reports
Commons
Home
License
Components
Sandbox
Commons Math 3.0 (requires Java 1.5+; not compatible with 2.2)
Binaries
commons-math3-3.0-bin.tar.gz
md5
pgp
commons-math3-3.0-bin.zip
md5
pgp
Source
commons-math3-3.0-src.tar.gz
md5
pgp
commons-math3-3.0-src.zip
md5
pgp
Commons Math 2.2 (requires Java 1.5+)


commons-math3-3.0-javadoc.jar
commons-math3-3.0-sources.jar
commons-math3-3.0.jar
ProyectoJava
src
JRE System Library [JavaSE-1.7]
JUnit 4
lib
ProyectoJava
src
JRE System Library [JavaSE-1.7]
JUnit 4
lib
commons-math3-3.0-javadoc.jar
commons-math3-3.0-sources.jar
commons-math3-3.0.jar


Properties for ProyectoJava
type filter text
Resource
Amateras
Builders
Java Build Path
Java Code Style
Java Compiler
Java Editor
Javadoc Location
Project References
Run/Debug Settings
Task Repository
Task Tags
Validation
WikiText
Java Build Path
Source
Projects
Libraries
Order and Export
Rs and class folders on the build path:
JRE System Library [JavaSE-1.7]
JUnit 4
Add JARs...
Add External JARs...
Add Variable...
Add Library...
Add Class Folder...
Add External Class Folder...
OK
JAR Selection
Choose the archives to be added to the build path:
type filter text
ProyectoJava
.settings
lib
commons-math3-3.0-javadoc.jar
commons-math3-3.0-sources.jar
commons-math3-3.0.jar
.classpath
.project
OK
Cancel


Properties for ProyectoJava
Java Build Path
Resource
Amateras
Builders
Java Build Path
Java Code Style
Java Compiler
Source
Projects
Libraries
Order and Export
JARs and class folders on the build path:
commons-math3-3.0.jar - ProyectoJava/lib
Source attachment: (None)
Add JARs...
Source Attachment Configuration
Select the location (folder, JAR or zip) containing the source for 'commons-math3-3.0.jar':
Location path:
Workspace...
External File...
External Folder...
Cancel
Source Location Selection
Select folder or JAR/zip archive containing the source:
ProyectoJava
.settings
bin
lib
commons-math3-3.0-javadoc.jar
commons-math3-3.0-sources.jar
commons-math3-3.0.jar
src
Create New Folder...
OK
Cancel
OK

```java
package org.pc.tema07;

import java.text.NumberFormat;

public class Fracciones {
    // ver http://commons.apache.org/math/userguide/fraction.html

    public static void main(String[] args) {
        // Constructores
        Fraction f = new Fraction(1, 3); // 1 / 3
        Fraction g = new Fraction(0.25); // 1 / 4

        System.out.println("Valor");// Valor
        System.out.println("toString f: " + f);
        System.out.println("double de f: " + f.doubleValue());
        System.out.println("porcentaje de f: " + f.percentageValue());

        Fraction lhs = new Fraction(1, 3);
        Fraction rhs = new Fraction(2, 5);

        System.out.println("Operaciones");
        Fraction answer = lhs.add(rhs); // suma dos fracciones
        System.out.println(lhs + " + " + rhs + " = " + answer);
        System.out.println(lhs + " - " + rhs + " = " + lhs.subtract(rhs));
        System.out.println("abs de lhs: " + lhs.abs());
        System.out.println("reciproca de lhs: " + lhs.reciprocal());
```

```java
        // formateado con FractionFormat
        NumberFormat nf = NumberFormat.getInstance(Locale.getDefault());
        FractionFormat format = new FractionFormat(nf);
        Fraction f1 = new Fraction(2000, 3333);
        String s = format.format(f1); // s contiene "2.000 / 3.333"
        System.out.println("Formateado igual numerador denominador: " + s);

        NumberFormat nf2 = NumberFormat.getInstance(Locale.US);
        format = new FractionFormat(nf, nf2);
        s = format.format(f1); // s contiene "2.000 / 3,333"
        System.out.println("Formateado distinto numerador denominador: " + s);

        // parseado con FractionFormat
        FractionFormat ff = new FractionFormat();
        Fraction f2 = ff.parse("-10 / 21");
        System.out.println("Parseado: " + f2);
    }
}
```

```
Valor
toString f: 1 / 3
double de f: 0.3333333333333333
porcentaje de f: 33.333333333333336
Operaciones
1 / 3 + 2 / 5 = 11 / 15
1 / 3 - 2 / 5 = -1 / 15
abs de lhs: 1 / 3
reciproca de lhs: 3
Equals
1/2 equals 2/4: true
Formateado igual numerador denominador: 2.000 / 3.333
Formateado distinto numerador denominador: 2.000 / 3,333
Parseado: -10 / 21
```

```java
package org.pc.tema07;

import org.apache.commons.math3.complex.Complex;

public class Complejos {
    // ver http://commons.apache.org/math/userguide/complex.html
    public static void main(String[] args) {
        //Constructor
        Complex c1 = new Complex(1.0, 3.0); // 1 + 3i
        System.out.println("toString f: " + c1);

        Complex lhs = new Complex(1.0, 3.0);
        Complex rhs = new Complex(2.0, 5.0);

        System.out.println("Operaciones");
        System.out.println(lhs + " + "+ rhs + " = " + lhs.add(rhs));
        System.out.println(lhs + " - "+ rhs + " = " + lhs.subtract(rhs));
        System.out.println("abs de lhs: " + lhs.abs());
        System.out.println("conjugado de lhs: " + lhs.conjugate());

        System.out.println("Otras Operaciones");
        System.out.println("Coseno de "+ lhs + " = " + lhs.cos());
        System.out.println("Exponencial de "+ lhs + " = " + lhs.exp());
        System.out.println("Logaritmo de "+ lhs + " = " + lhs.log());

        //formateado con ComplexFormat
        ComplexFormat format = new ComplexFormat(); //
        Complex c = new Complex(1.1111, 2.2222);
        String s = format.format(c); // s contains "1.1
        System.out.println("Formateado: " + s);
        //parseado con ComplexFormat
        ComplexFormat cf = new ComplexFormat();
        Complex c2 = cf.parse("1.110 + 2.222i");
        System.out.println("Parseado: " + c2);
    }
}
```

```
toString f: (1.0, 3.0)
Operaciones
(1.0, 3.0) + (2.0, 5.0) = (3.0, 8.0)
(1.0, 3.0) - (2.0, 5.0) = (-1.0, -2.0)
abs de lhs: 3.1622776601683795
conjugado de lhs: (1.0, -3.0)
Otras Operaciones
Coseno de (1.0, 3.0) = (5.439580991019764, -8.429751080849945)
Exponencial de (1.0, 3.0) = (-2.6910786138197937, 0.383603953541131)
Logaritmo de (1.0, 3.0) = (1.151292546497023, 1.2490457723982544)
Formateado: 1,11 + 2,22i
Parseado: (1110.0, 2222.0)
```

```java
package org.pc.tema07;

import org.apache.commons.math3.random.RandomData;

public class EstadisticaDescriptiva {
    // http://commons.apache.org/math/userguide/stat.html#a1.2_Descriptive_st
    public static void descriptiveStatisticsInstance(double[] datos) {

    public static void summaryStatisticsInstance(double[] datos) {

    public static void statUtils(double[] datos) {

    public static void main(String[] args) {
        /*NumerosAleatorios na = new NumerosAleatorios();
        double[] datos = new double[100];
        for (int i = 0; i < datos.length; i++) {
            datos[i] = na.randomReal();
        }*/

        RandomData randomData = new RandomDataImpl();
        double[] datos = new double[100];
        for (int i = 0; i < 100; i++) {
            datos[i] = randomData.nextInt(1, 100);
        }

        EstadisticaDescriptiva.descriptiveStatisticsInstance(datos);
        EstadisticaDescriptiva.summaryStatisticsInstance(datos);
        EstadisticaDescriptiva.statUtils(datos);
    }
}
```

```java
public static void descriptiveStatisticsInstance(double[] datos) {
    // Crea una instacia DescriptiveStatistics
    DescriptiveStatistics stats = new DescriptiveStatistics();
    // Añade los datos al array
    for (int i = 0; i < datos.length; i++) {
        stats.addValue(datos[i]);
    }
    // Algunos valores estadisticos
    double mean = stats.getMean();
    double std = stats.getStandardDeviation();
    double median = stats.getPercentile(50);
    System.out.println("Media = " + mean +" Desviación estandar =  "+ std + " Mediana =  "+ median);
}
public static void summaryStatisticsInstance(double[] datos) {
    SummaryStatistics stats = new SummaryStatistics();
    for (int i = 0; i < datos.length; i++) {
        stats.addValue(datos[i]);
    }
    double mean = stats.getMean();
    double std = stats.getStandardDeviation();
    // double median = stats.getMedian(); <-- NOT AVAILABLE
    System.out.println("Media = " + mean +" Desviación estandar =  "+ std);
}
public static void statUtils(double[] datos) {
    // Calcula estadisticas desde una double[] array
    double mean = StatUtils.mean(datos);
    double std = StatUtils.variance(datos);
    double median = StatUtils.percentile(datos, 50);
    System.out.println("Media = " + mean +" Desviación estandar =  "+ std + " Mediana =  "+ median);
    // Media de los tres primeros
    //mean = StatUtils.mean(datos, 0, 3);
    System.out.println("media de los tres primeros = " + StatUtils.mean(datos, 0, 3));
    System.out.println( "min: " + StatUtils.min( datos ) );
    System.out.println( "max: " + StatUtils.max( datos ) );
    System.out.println( "media: " + StatUtils.mean( datos ) );
    System.out.println( "producto: " + StatUtils.product( datos ) );
    System.out.println( "suma: " + StatUtils.sum( datos ) );
    System.out.println( "varianza: " + StatUtils.variance( datos ) );
}
```

```
Media = 46.730000000000004 Desviación estandar =  28.91012020910758 Mediana =  43.0
Media = 46.730000000000004 Desviación estandar =  28.910120209107586
Media = 46.730000000000004 Desviación estandar =  835.7950505050503 Mediana =  43.0
media de los tres primeros = 41.333333333333336
min: 2.0
max: 99.0
media: 46.730000000000004
producto: 1.749159896358876E154
suma: 4673.0
varianza: 835.7950505050503
```

```java
package org.pc.tema07;
import java.text.NumberFormat;

public class RegresionSimple {
        // ver http://commons.apache.org/math/userguide/stat.html#a1.4_Simple_regression
    public static void main(String[] args) {
        SimpleRegression sr = new SimpleRegression();

        sr.addData(0, 0);
        sr.addData(1, 1.2);
        sr.addData(2, 2.6);
        sr.addData(3, 3.2);
        sr.addData(4, 4);
        sr.addData(5, 5);
        NumberFormat format = NumberFormat.getInstance();
        System.out.println( "Corte: " + format.format( sr.getIntercept() ) );
        System.out.println( "N: " + sr.getN() );
        System.out.println( "Pendiente: " + format.format( sr.getSlope() ) );
        System.out.println( "Confidencia pendiente: " + format.format( sr.getSlopeConfidenceInterval() ) );
        System.out.println( "R cuadrado: " + format.format( sr.getRSquare() ) );

        sr.addData( 400, 100 );
        sr.addData( 300, 105 );
        sr.addData( 350, 70 );
        sr.addData( 200, 50 );
        sr.addData( 150, 300 );
        sr.addData( 50, 500 );
        format = NumberFormat.getInstance(Locale.getDefault());
        System.out.println( "Corte: " + format.format( sr.getIntercept() ) );
        System.out.println( "N: " + sr.getN() );
        System.out.println( "Pendiente: " + format.format( sr.getSlope() ) );
        System.out.println( "Confidencia pendiente: " + format.format( sr.getSlopeConfidenceInterval() ) );
        System.out.println( "R cuadrado: " + format.format( sr.getRSquare() ) );
    }
}
```

```
Corte: 0,238
N: 6
Pendiente: 0,971
Confidencia pendiente: 0,169
R cuadrado: 0,985
Corte: 77,736
N: 12
Pendiente: 0,142
Confidencia pendiente: 0,699
R cuadrado: 0,02
```

```java
package org.pc.tema07;
import org.apache.commons.math3.linear.Array2DRowRealMatrix;
public class SistemasEcuacionesLineales {

    public static void main(String[] args) {
        // Create a real matrix with two rows and three columns
        double[][] matrixData = { {1d,2d,3d}, {2d,5d,3d}};
        RealMatrix m = new Array2DRowRealMatrix(matrixData);
        // One more with three rows, two columns
        double[][] matrixData2 = { {1d,2d}, {2d,5d}, {1d, 7d}};
        RealMatrix n = new Array2DRowRealMatrix(matrixData2);
        // Now multiply m by n
        RealMatrix p = m.multiply(n);
        System.out.println(p.getRowDimension());    // 2
        System.out.println(p.getColumnDimension()); // 2
        System.out.println(p);
        // Invert p, using LU decomposition
        LUDecomposition solver0 = new LUDecomposition(p);
        RealMatrix pInverse = solver0.getSolver().getInverse();
        System.out.println(pInverse);
        /*2x + 3y - 2z = 1
         -x + 7y + 6x = -2
          4x - 3y - 5z = 1*/
        RealMatrix coefficients =
                new Array2DRowRealMatrix(new double[][] { { 2, 3, -2 }, { -1, 7, 6 }, { 4, -3, -5 } },
                                         false);
        LUDecomposition solver = new LUDecomposition(coefficients);
        RealVector constants = new ArrayRealVector(new double[] { 1, -2, 1 }, false);
        RealVector solucion = solver.getSolver().solve(constants);
        System.out.println(solucion);
        QRDecomposition solver3 = new QRDecomposition(coefficients);
        RealVector solucion3 = solver3.getSolver().solve(constants);
        System.out.println(solucion3);
    }
}
```

```
2
2
Array2DRowRealMatrix{{8.0,33.0},{15.0,50.0}}
Array2DRowRealMatrix{{-0.5263157894736843,0.3473684210526316},{0.15789473684210528,-0.08421052631578949}}
{-0,37; 0,18; -0,6}
{-0,37; 0,18; -0,6}
```

¡Muchas Gracias y Fin!