The Universal Java Matrix Package (UJMP)

Everything is a Matrix!

ICML/MLOSS, Haifa, 2010-06-25

Holger Arndt
Technical University of Munich
Department of Computer Science
Garching, Germany
mail@holger-arndt.com
http://www.holger-arndt.com

Find out more at:
http://www.ujmp.org

Outline

- Introduction
- Comparison of Matrix Libraries for Java
- Concepts for a Next-Generation Matrix Library
- Integration of Other Matrix Libraries
- Calculation Methods
- Matrix Annotation
- Automatic Entry Type Conversion
- Demo
- Summary and Discussion

Introduction

Why do we need yet another Java matrix library?

- Matrix computations essential in various fields of computer science (machine learning, data mining, etc.)
- Collaborative networks require large sparse adjacency matrices
- Increasing amount of data

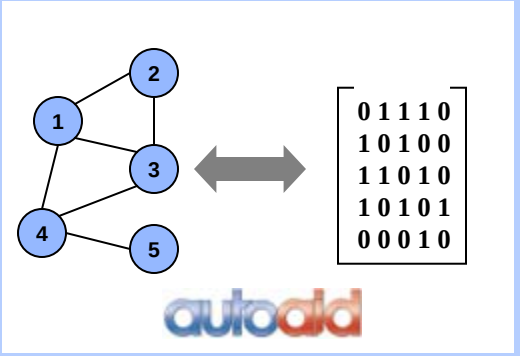
But:

- No direct support for matrix algebra in JDK
- Matlab or Octave cannot always be used
- Other libraries have limitations: JAMA, Colt, MTJ, commons-math

Online Marketing

Bio-Medical Data Analysis

Collaborative Networks



Comparison of Matrix Libraries for Java

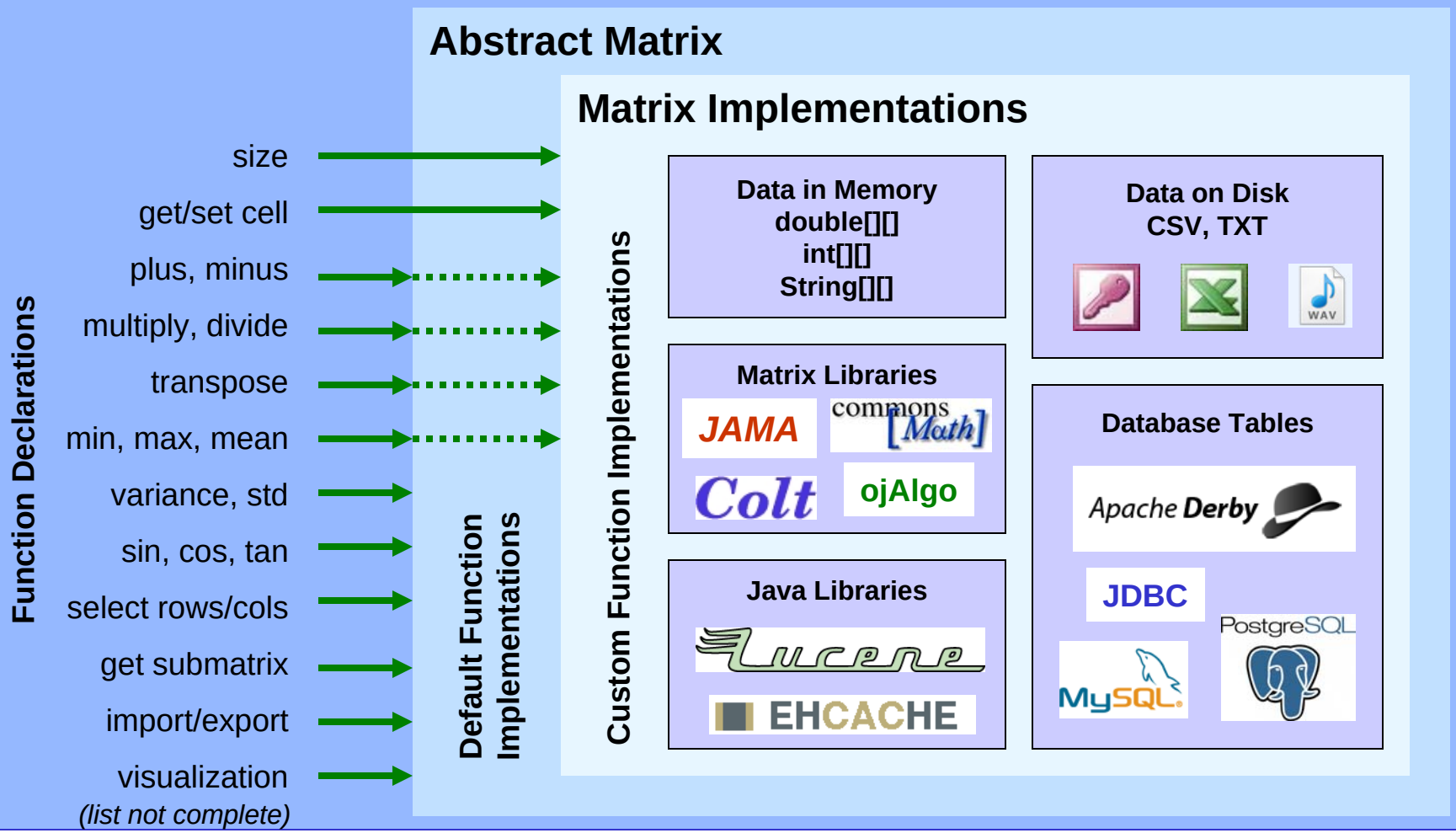
No single Java matrix library can fulfill all needs! We need a „universal“ matrix package...

	JAMA	Colt	MTJ	commons-math	UJMP
extendable	✘	✓	✓	•	✓
dense matrices	✓	✓	✓	✓	✓
sparse matrices	✘	✓	✓	✘	✓
2D matrices	✓	✓	✓	✓	✓
3D matrices	✘	✓	✘	✘	✓
4D matrices	✘	✘	✘	✘	✓
matrices > 4D	✘	✘	✘	✘	✓
> 2^31 rows/columns	✘	✘	✘	✘	✓
object entries	✘	✓	✘	✘	✓
generic entries	✘	✘	✘	✘	✓
matrices > RAM	✘	✘	✘	✘	✓
advanced operators	✘	✓	✘	✘	✓
import/export filters	✘	✘	✓	✘	✓
Matlab/Octave/R interface	✘	✘	✘	✘	✓
visualization methods	✘	✘	✘	✘	✓

Concepts for a Next-Generation Matrix Library

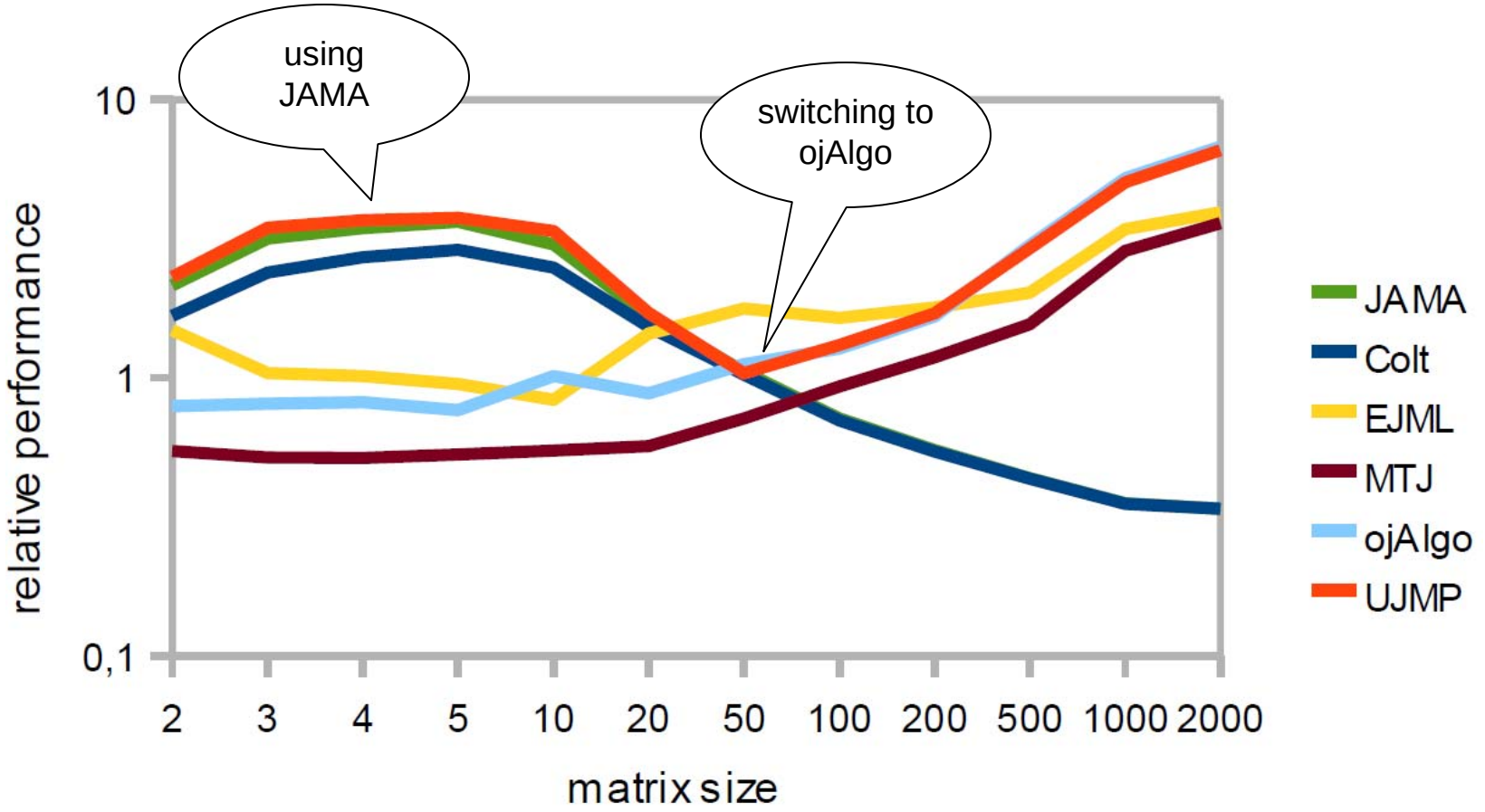
The actual implementation of a matrix becomes secondary!

Matrix Interface multi-dimensional, dense/sparse, 2^63 rows/columns, various cell types



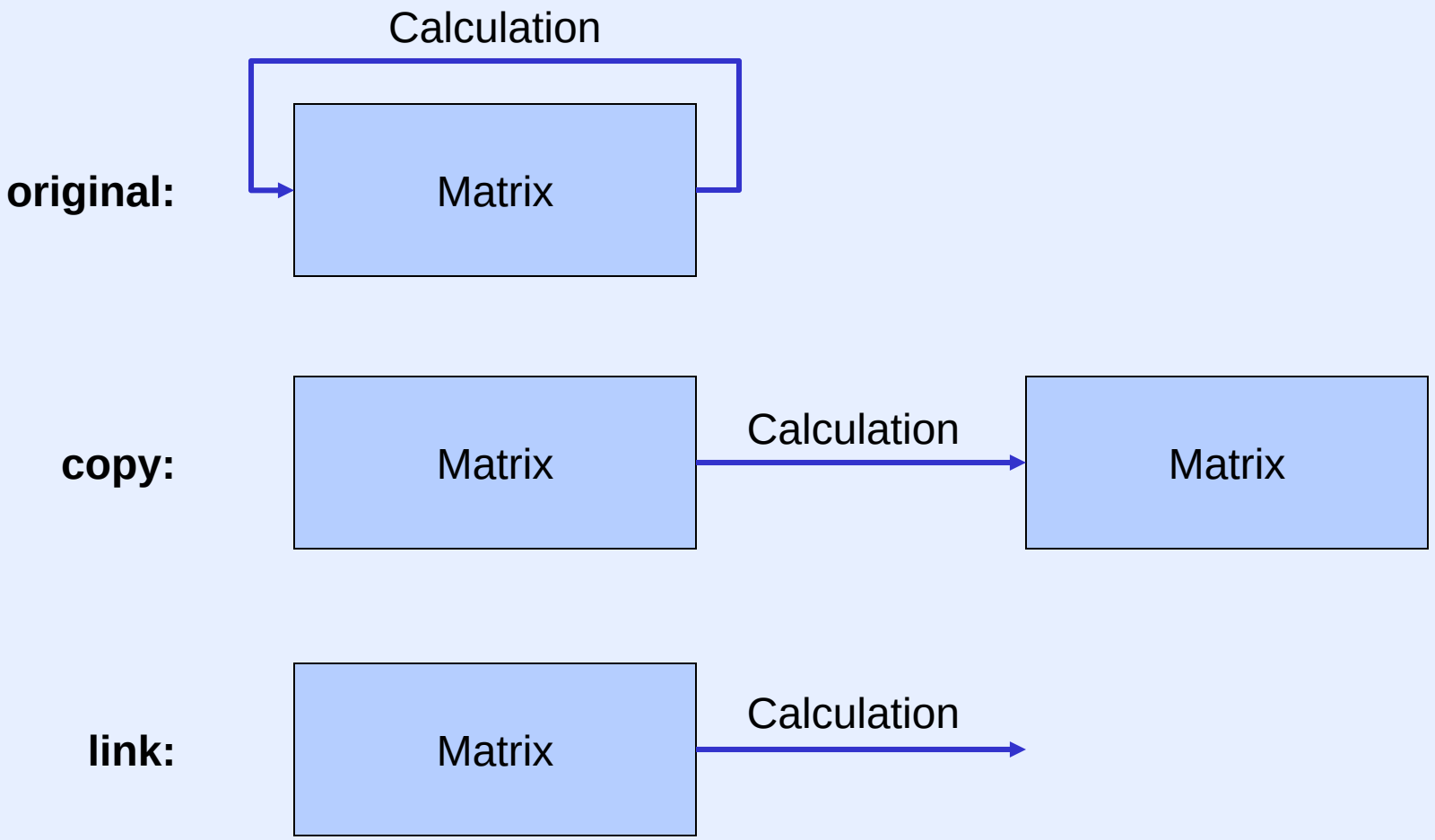
Integration of Other Matrix Libraries

Switching to faster libraries for better performance. Example: SVD



Calculation Methods

There are three different „modes“ to perform a calculation.



Matrix Annotation

Data requires annotation to be valuable.

matrix label

column labels

axis label

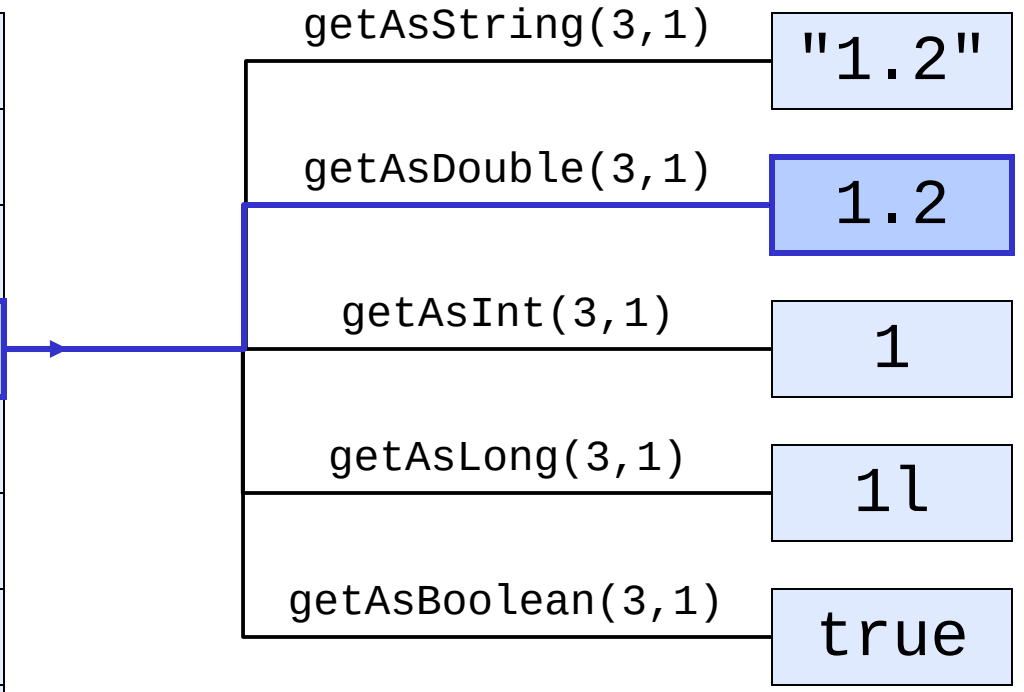
Report June 2009		product data				
		product id	**# of sales**	**in stock**	**margin**	**price [US$]**
label for row axis	**row1**	6757	5	yes	15.4%	230.87
	row2	6876	1	yes	20.3%	330.53
	row3	9976	4	yes	12.3%	321.45
	row4	9975	2	no	7.4%	732.42
	row5	980	1	yes	2.4%	643.32
	row6	8657	1	yes	33.2%	313.53
	row7	7677	5	no	23.4%	832.95
	row8	7657	13	yes	11.5%	542.32
	row9	6678	9	yes	6.5%	232.54
	row10	8865	2	yes	45.6%	335.21

Automatic Entry Type Conversion

Not all matrices contain numerical data.

Matrix imported from CSV

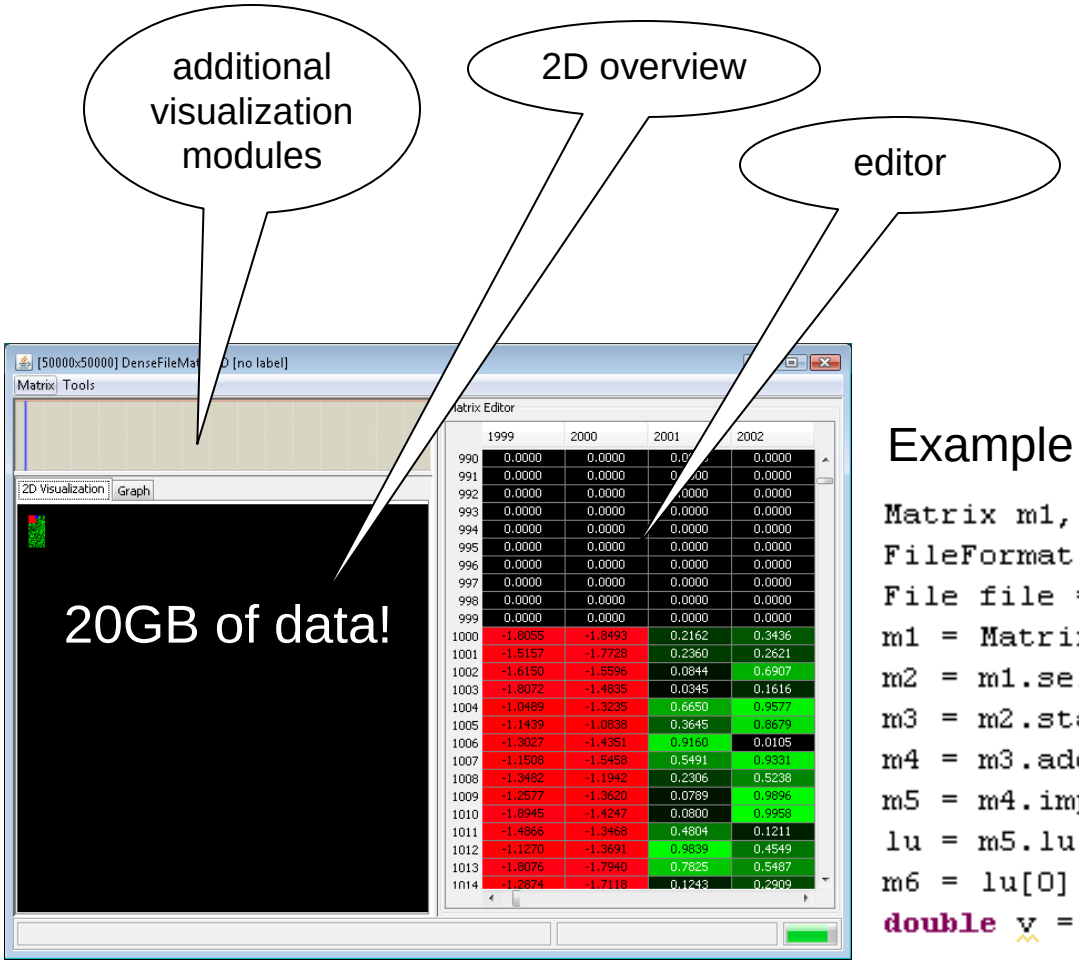
"This"	"5.7"
"matrix"	"1.9"
"contains"	"4.0"
"Strings,"	"1.2"
"data"	"9.1"
"must"	"0.5"
"be"	"7.7"
"converted"	"3.8"



Supported value types: float, double, byte, char, short, int, long, boolean, Date, BigDecimal, BigInteger, String, Object, <Generic>

Demo

It is important to visualize data.



Example Code:

```
Matrix m1, m2, m3, m4, m5, lu[], m6;
FileFormat csv = FileFormat.CSV;
File file = new File("example.csv");
m1 = MatrixFactory.importFromFile(csv, file);
m2 = m1.select(Ret.NEW, "4-100;6,7,8-10,100-200");
m3 = m2.standardize(Ret.NEW, Matrix.ROW);
m4 = m3.addMissing(Ret.NEW, Matrix.ALL, 0.05);
m5 = m4.impute(Ret.NEW, ImputationMethod.KNN, 2);
lu = m5.lu();
m6 = lu[0].bootstrap(Ret.LINK, 1000000);
double v = m6.getAsDouble(0, 0);
```

Summary and Discussion

The Universal Java Matrix Package: A novel and innovative matrix library for Java.

Summary

- Extendable architecture
- Ready for large amounts of data
- Integration of other libraries
- Flexible calculation methods
- Open Source LGPL
- Online forum for Q&A

Future Work:

- Documentation
- Developers wanted!