Independent Component Analysis

– in Clojure

AIM Lab group meeting, May 19, 2010
Presented by: David Lebech

Outline

- Independent Component Analysis
  - A quick intro
- Clojure
  - Another quick intro
- Results
- Future work and conclusion

Motivation

- I wanted:
  - To learn a new programming language
    - Clojure
  - Apply the language to a specific problem
- Solution:
  - Build upon last year's ICA research
  - Implement an ICA algorithm in Clojure

What is ICA?

- Separates data into maximally independent components
- We assume that the original signals are *independent*
  - $P(A \cap B) = P(A)*P(B)$
- We assume that the original signals are *linearly mixed*
  - $x1 = a*s1 + b*s2$

ICA – an intuitive example

- The *cocktail party problem*
  - *n* persons in a room
    - All speaking at the same time
  - *n* mixed signals
    - E.g. *n* microphones in different locations
  - Our brain can easily distinguish the *n* signals
  - A computer needs help, e.g. ICA

Definition of ICA

- Given an *m* by *n* data matrix *x*
  - *m* is the (assumed) number of original signals
  - *n* is the number of data points
- Assume that *x* is a linear mix of original signals *s*, according to:

$$x = As$$

- *A* is thus an *m* by *m* mixing matrix

Definition of ICA – cont.

- Goal is to find an unmixing matrix *W* such that:

$$s' = Wx$$

- *s'* is an approximation to the original signals *s*
- *W* is the inverse of *A*
  - $W = A^{-1}$

ICA caveats

- We cannot determine the original *sign* (+/-) of the found components
- We cannot determine the *order* of the found components.
- The distributions of the original signals most be *nongaussian*
  - gaussian distribution = normal distribution

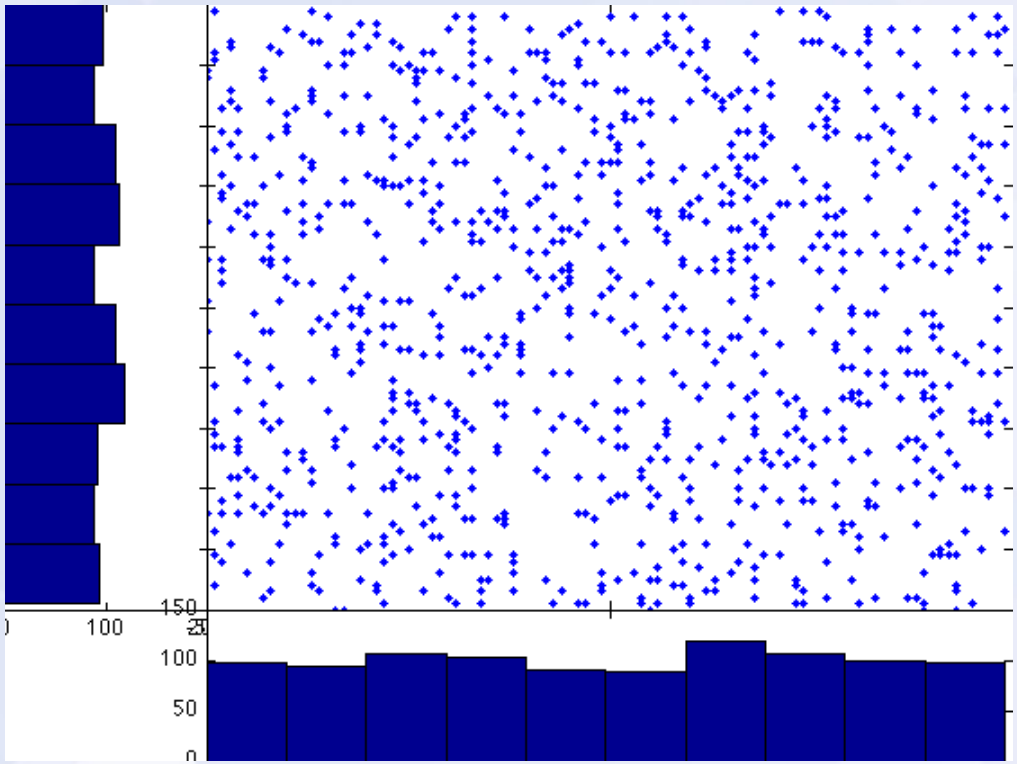
How does ICA work?

- Central Limit Theorem
  - The distribution of the sum of independent variables is more gaussian than the distribution of the variables themselves
- For an unmixing vector *w*, ICA maximizes the nongaussianity of

$$w^T x$$

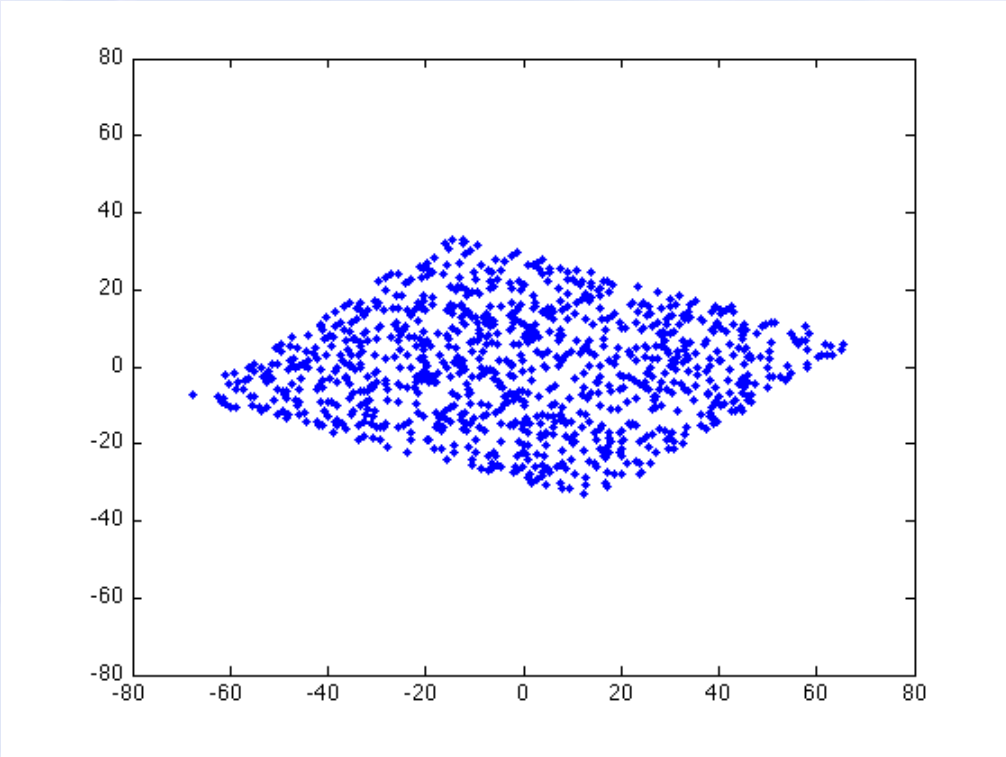
Let's look at an example

ICA example – random signals



Scatter-plot of two random signals, along with their distributions which are uniform

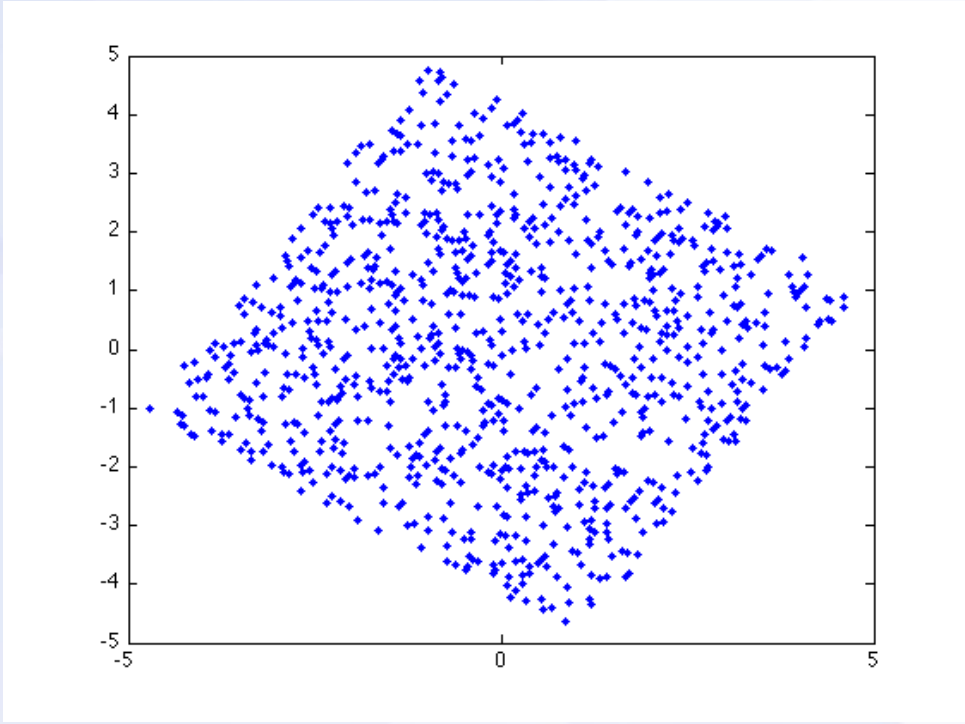
ICA example – mixed signals



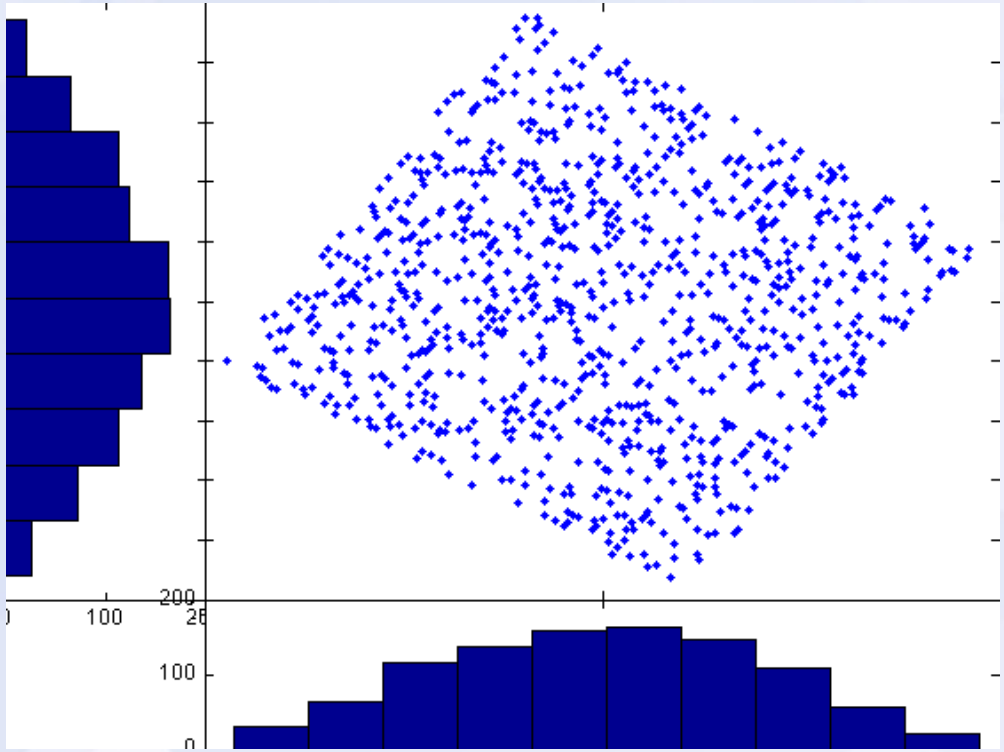
Scatter-plot of the mixed signals

ICA example – preprocessing

- *Whitening* of data components:
- Uncorrelation
- Variances equal unity



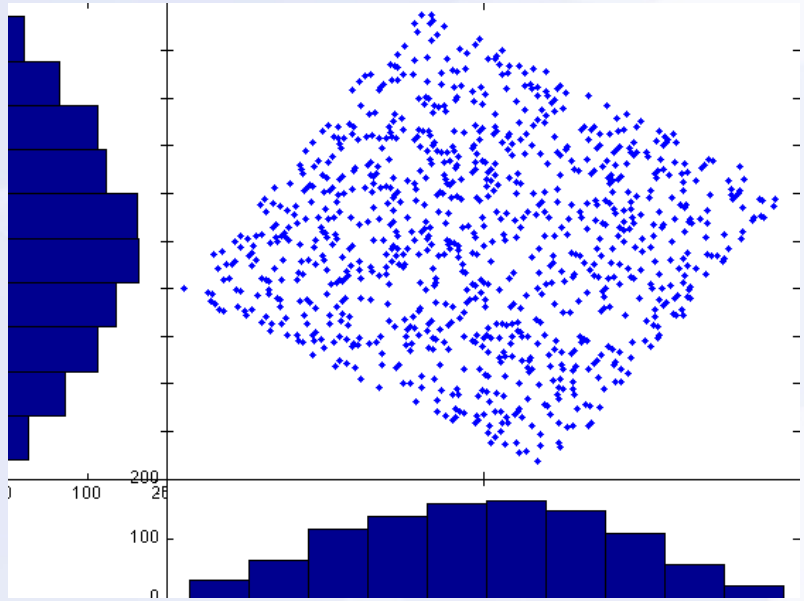
ICA example – whitened data



Scatter-plot of whitened mixed signals, along with their distributions which are more gaussian than the original signals

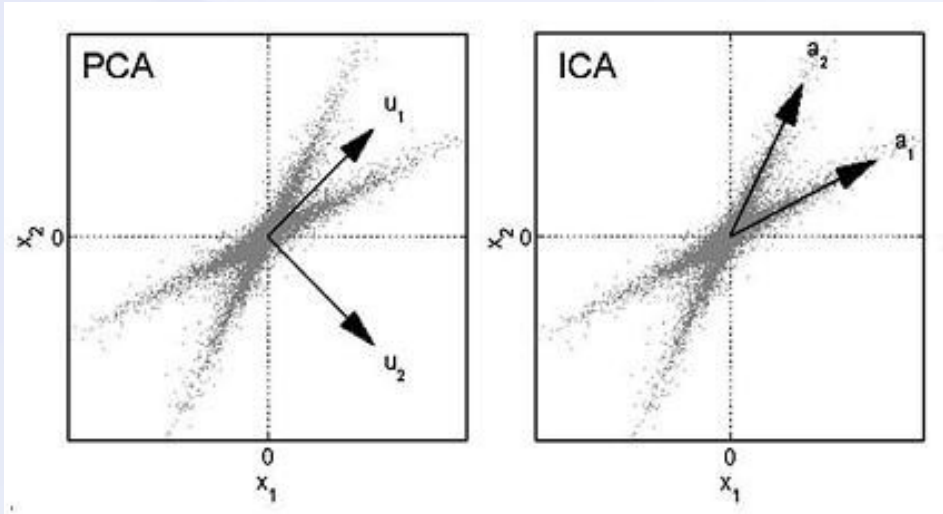
ICA example – finding components intuitively

- Find weight vectors *w1*, *w2* that "rotate" the mixed signals
- *w1* and *w2* maximize non-gaussianity of the components of *x*



Principal Components Analysis versus ICA

- PCA finds components with decreasing "importance"
- ICA finds maximally independent components
- PCA is sometimes used as a pre-processing step for ICA



Clojure overview

- Dynamic
  - Direct interaction through the REPL
    - REPL = Read Eval Print Loop
    - Compiles on-the-fly
- LISP dialect
- Functional programming language
  - Functions are first-class objects

Clojure overview

- Designed for concurrency
  - Software transactional memory (STM)
  - Immutable and persistent data structures
- Runs on the Java Virtual Machine (JVM)
  - Compiles into bytecode
  - Access to the entire Java library
    - But discourages use of Java data structures since they are not immutable

Clojure syntax

()

Clojure syntax

- But seriously, it is very sparse
- Clojure uses the code-as-data principle as other LISPs
  - S-expressions “()” are parsed into data structures and then compiled
    - In other words, everything is basically a data structure
  - Prefix notation

Is Clojure cool?

- Potentially
  - It is gaining a lot of attention
  - A lot of people are overly excited about it
  - Makes concurrency easier
    - No locking
- It requires a *totally* different mindset than normal OO/imperative programming
  - I haven't gotten it yet

ICA and Clojure – challenges

- Going from mathematical definition to code
- Understanding Clojure
  - Working with immutable data structures
    - Data cannot “change”
  - Recursion versus looping

FastICA

- Assumes data to be centered and whitened
- Uses *negentropy* to measure nongaussianity
  - Negentropy is maximal for a distribution that is nongaussian

FastICA

- General update rule (from literature):

  $\mathbf{w}' = E\{\mathbf{x}g(\mathbf{w}^T\mathbf{x})\} - E\{g'(\mathbf{w}^T\mathbf{x})\}\mathbf{w}$

  $\mathbf{w} = \mathbf{w}' / ||\mathbf{w}'||$

  repeat until converged (i.e. old and new **w** point in same direction)

- In practice (from Matlab implementation)

  $\mathbf{w}' = E\{\mathbf{x}(\mathbf{x}^T\mathbf{w})^3\} - 3\mathbf{w}$

  $\mathbf{w} = \mathbf{w}' / ||\mathbf{w}'||$

Current progress

- FastICA implemented in Clojure
  - Based largely on the Matlab version
  - Uses the Clojure library *Incanter*
    - Goal of Incanter is to supply R-like functionality in Clojure
  - Simple “deflation” approach for decorrelating found basis vectors

Current progress

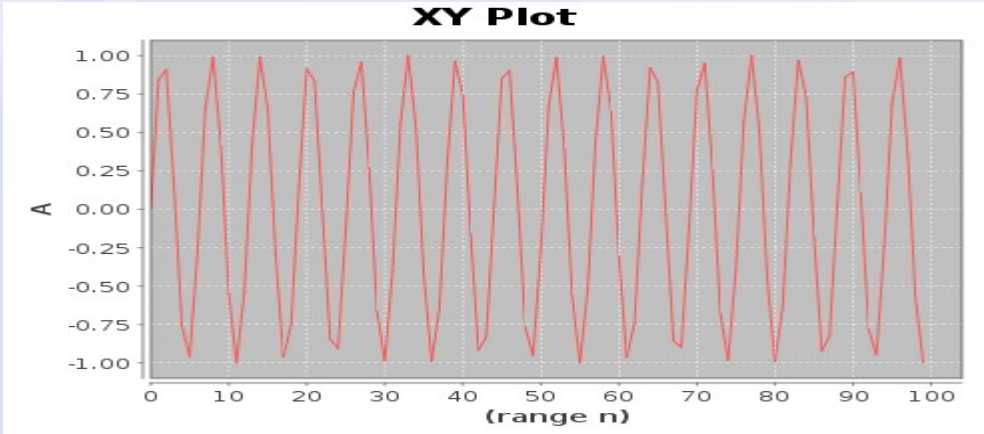
- Comparison with Matlab implementation
  - Looking at found independent components (very subjective)
  - Measuring running time
    - Requires very large data sets
      - Current experiments run in ms
    - I haven't done this yet.

Test 1

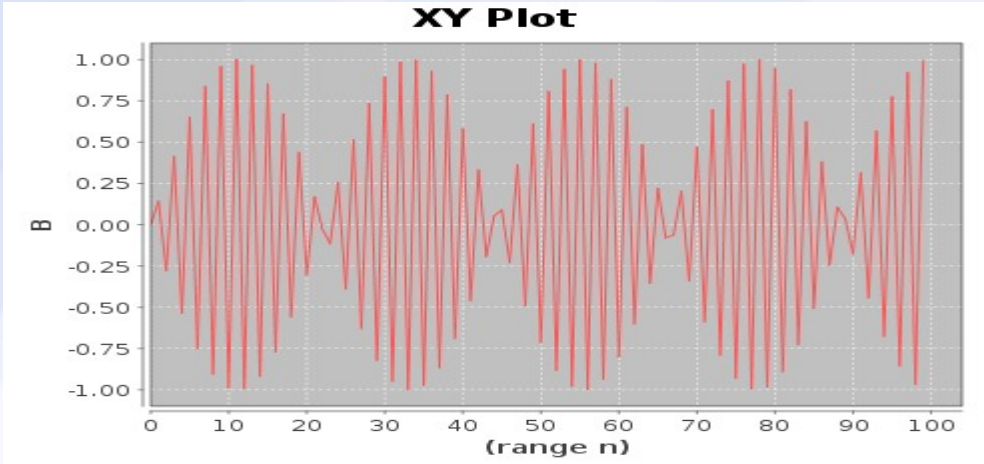
- Two sources
  - s1 = sin(x), x = 0,...,99
  - s2 = sin(3x), x = 0,...,99
- Two mixed signals (linear combinations)
  - x1 = s1 – 2*s2
  - x2 = 1.73*s1 + 3.41*s2

Test 1 – original sources

s1



s2

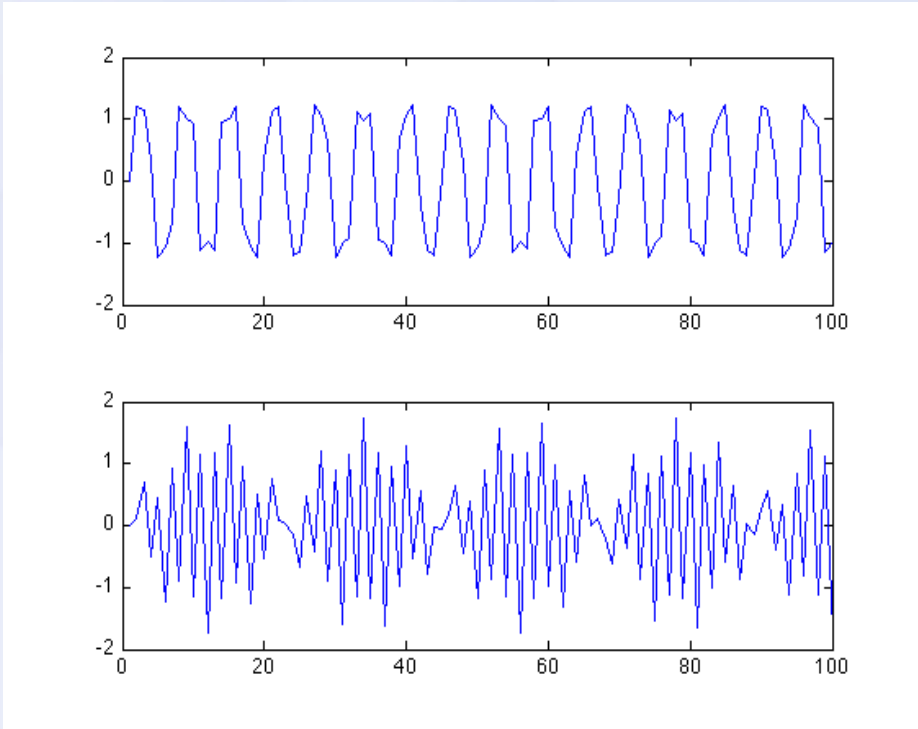


Test 1 – mixed signals

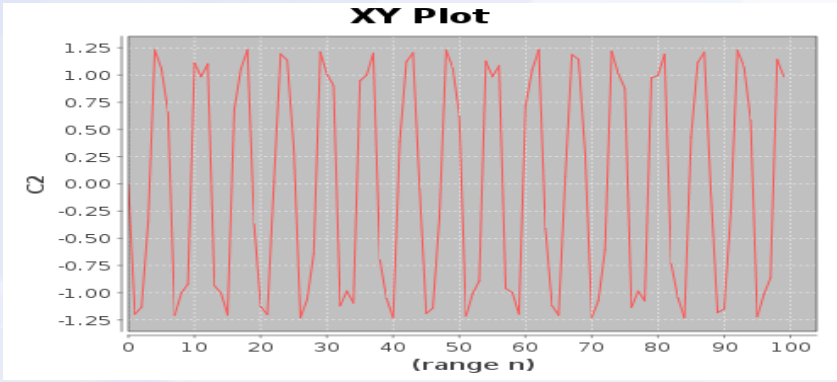
x1

x2

Test 1 – found components



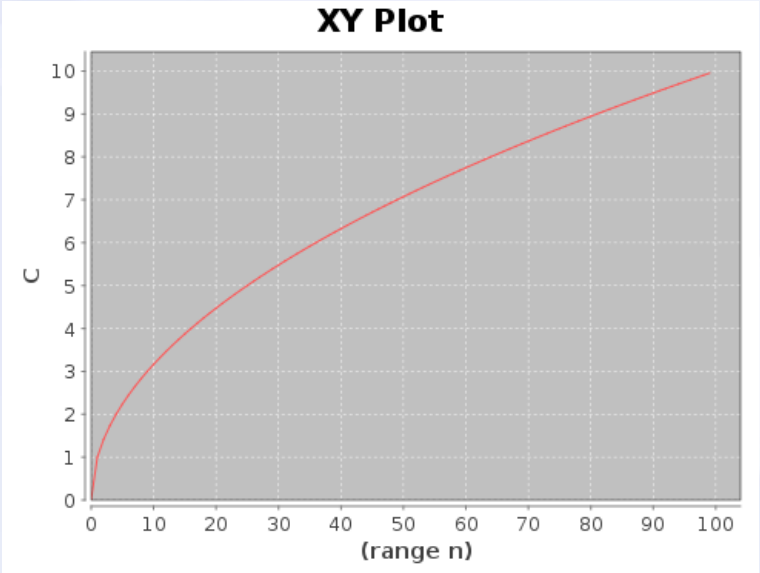
Matlab implementation



Clojure implementation

Test 2

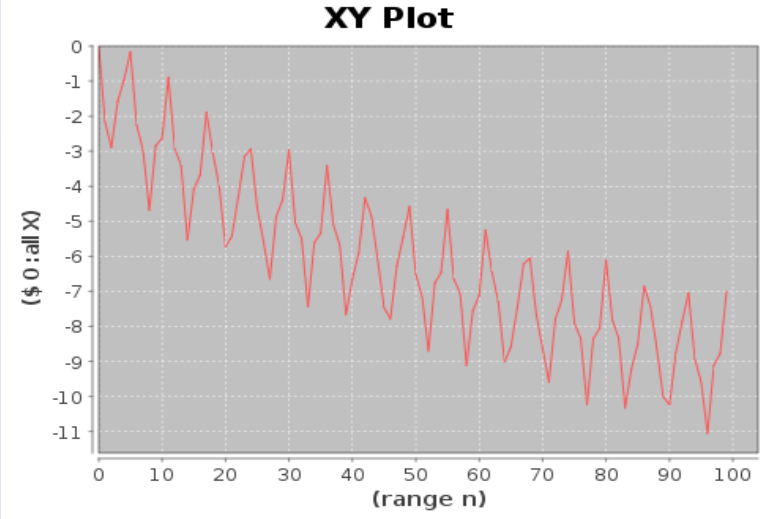
- Add a third component
  - s3 = sqrt(x), x = 0,...,99
- x1, x2 and x3 are mixed by a *random* mixing matrix A with values between -3 and 3

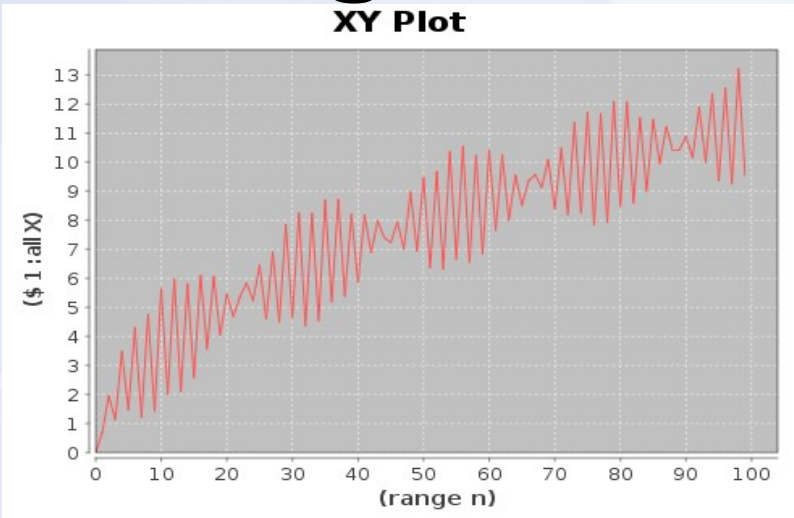


s3

Test 2 – mixed signals

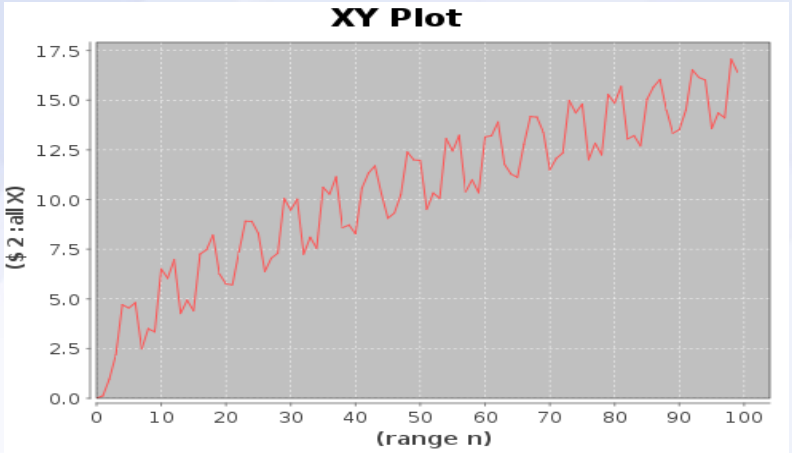
x1



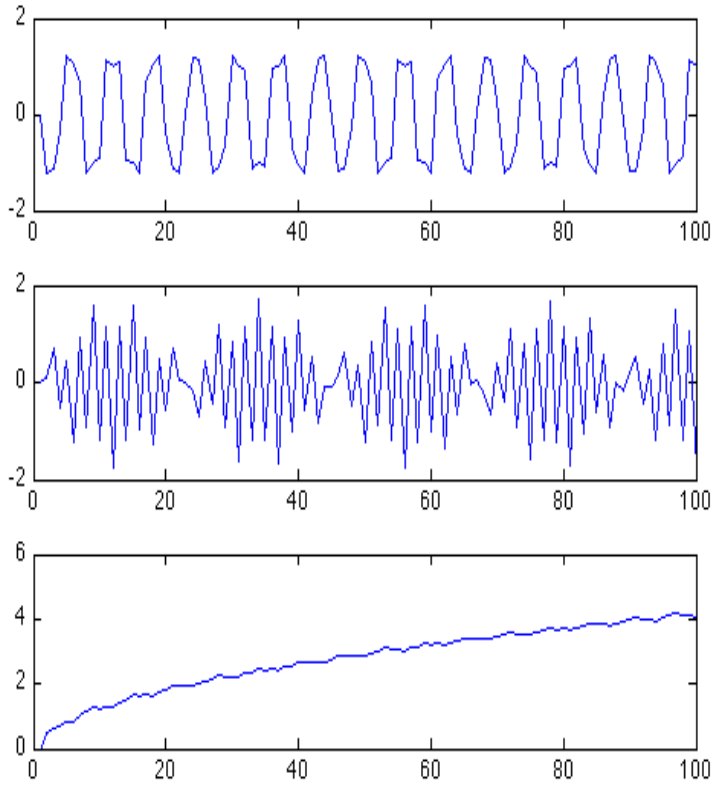


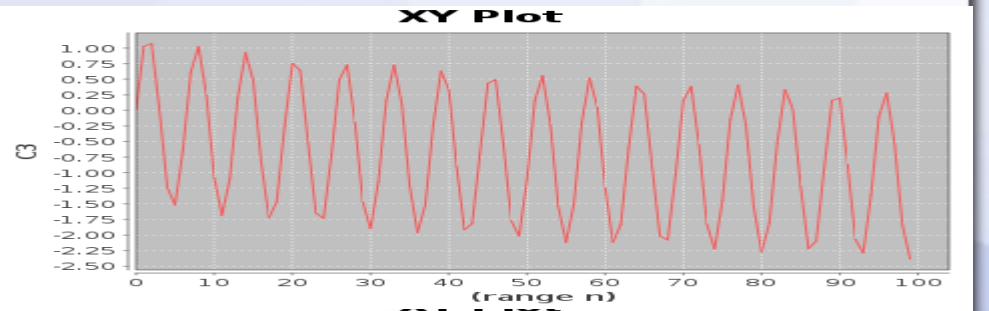
x2

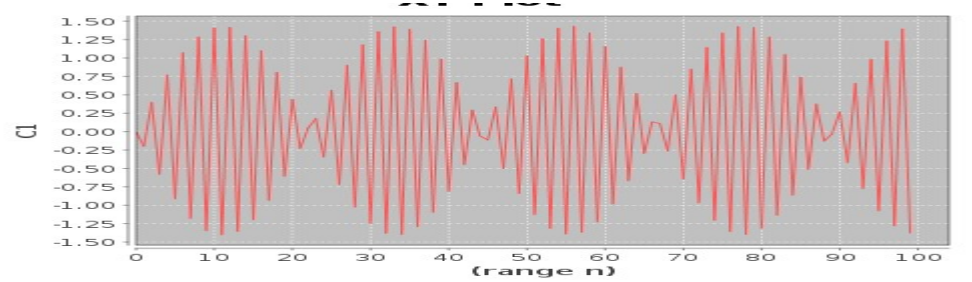
x3

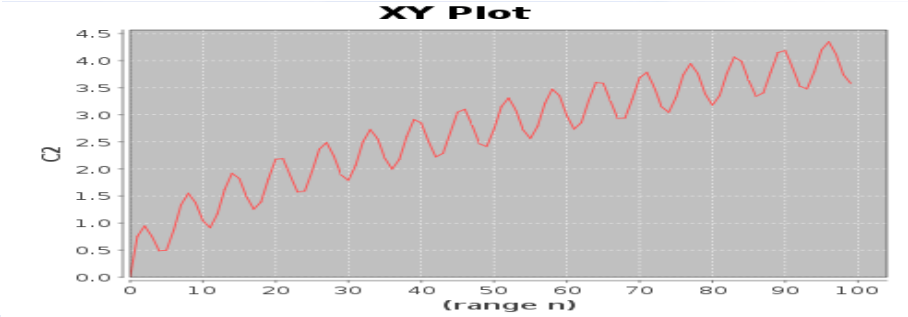


Test 2 – found components



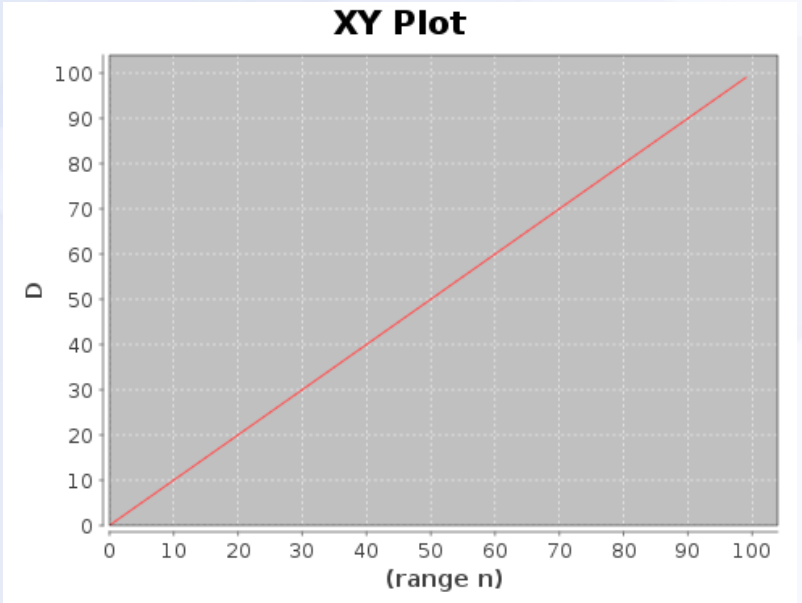






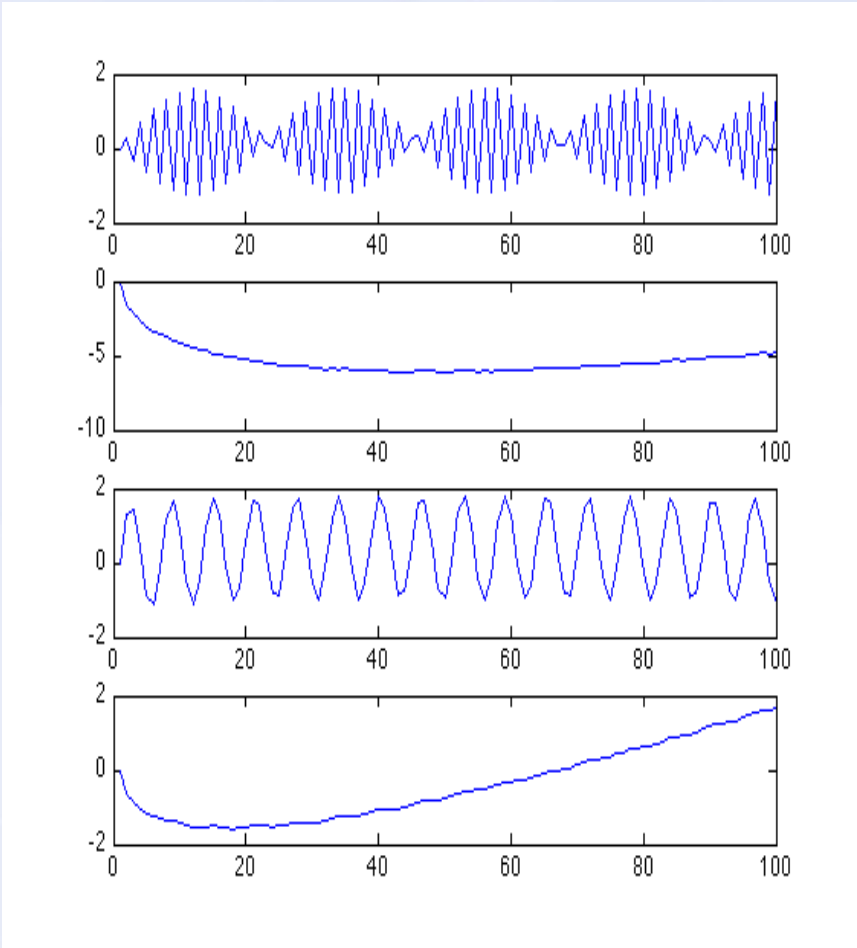
Test 3

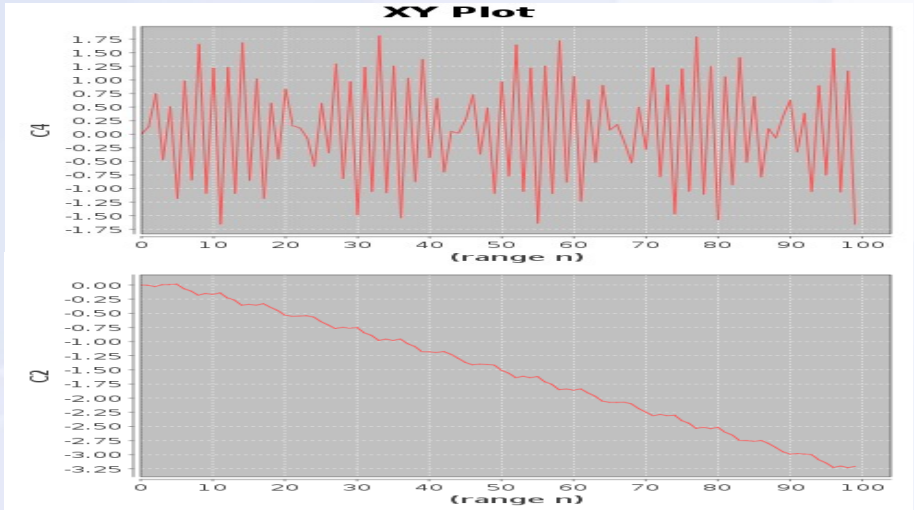
- Add a fourth component
  - s4 = x, x = 0,...,99
- x1, x2, x3 and x4 are mixed by a *random* mixing matrix A with values between -3 and 3

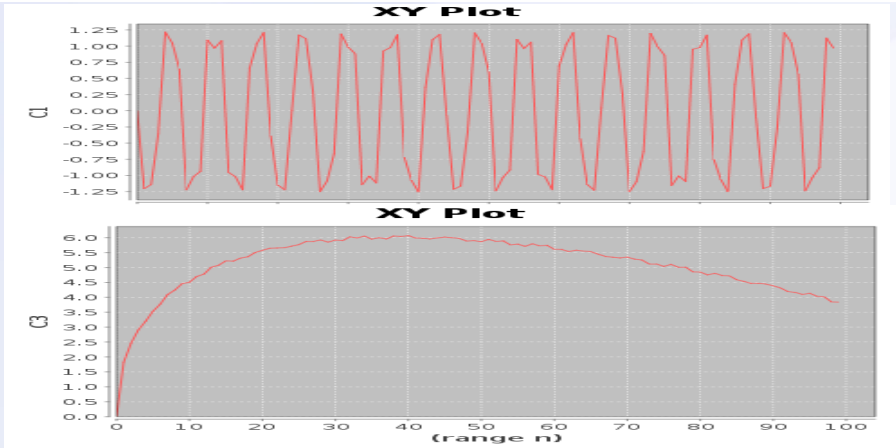


s3

Test 3 – found components







Test results

- Same algorithm, different outcome, why?
  - Different mixing matrices
    - But for test 1, they were identical
  - Different start vectors
    - Converges differently?

ICA on real world data

- Demixing sound signals
- Face recognition
  - Facial “components”
- Stock market prices
  - Underlying hidden factors
    - I tried this

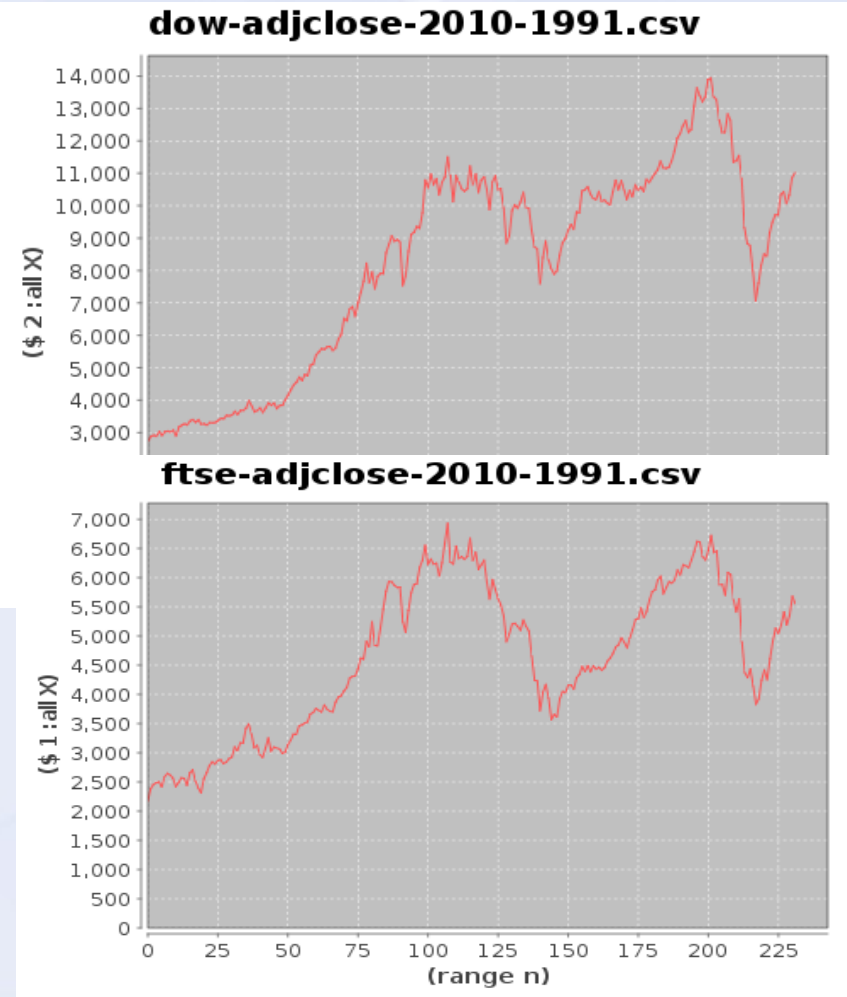
ICA on stock market

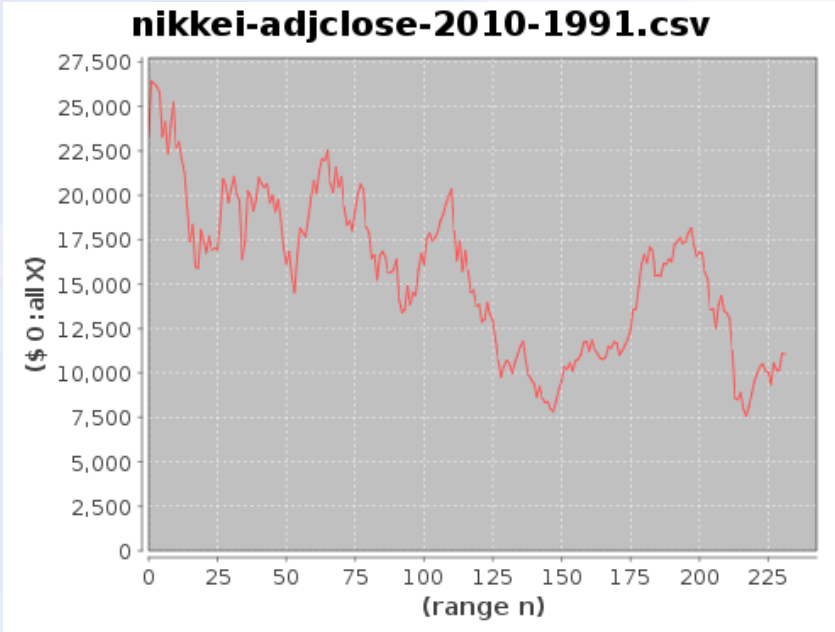
- Three large stock market indexes
  - Dow Jones (USA)
  - Nikkei (Japan)
  - FTSE (Europe)
- Monthly data from 1991-2010
- Each stock index is an observed variable
- Each time point is an observation

ICA on stock market

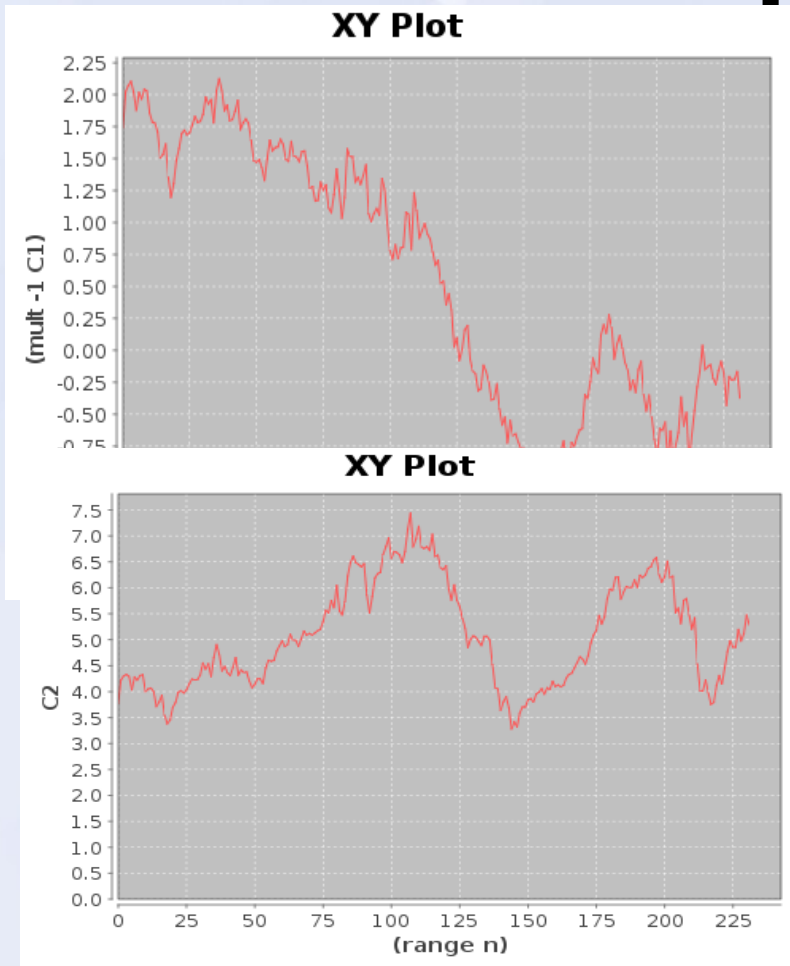
- Hypothesis:
  - The three indexes have some underlying hidden factors that affect all three stock markets
    - Can this be true?

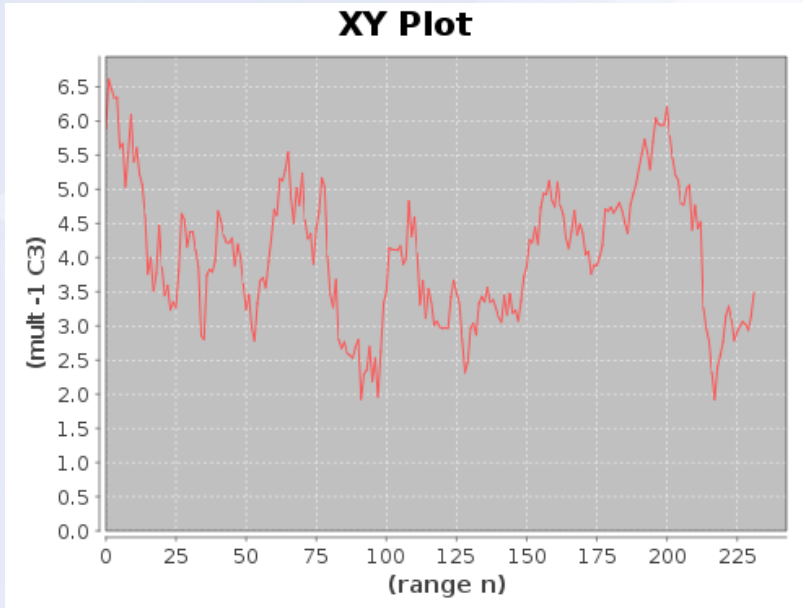
Stock market indexes



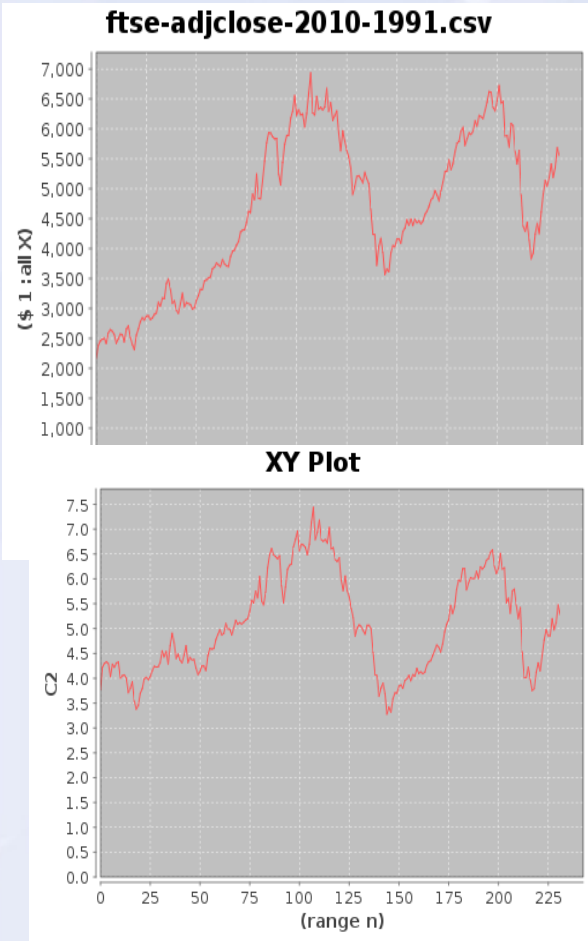


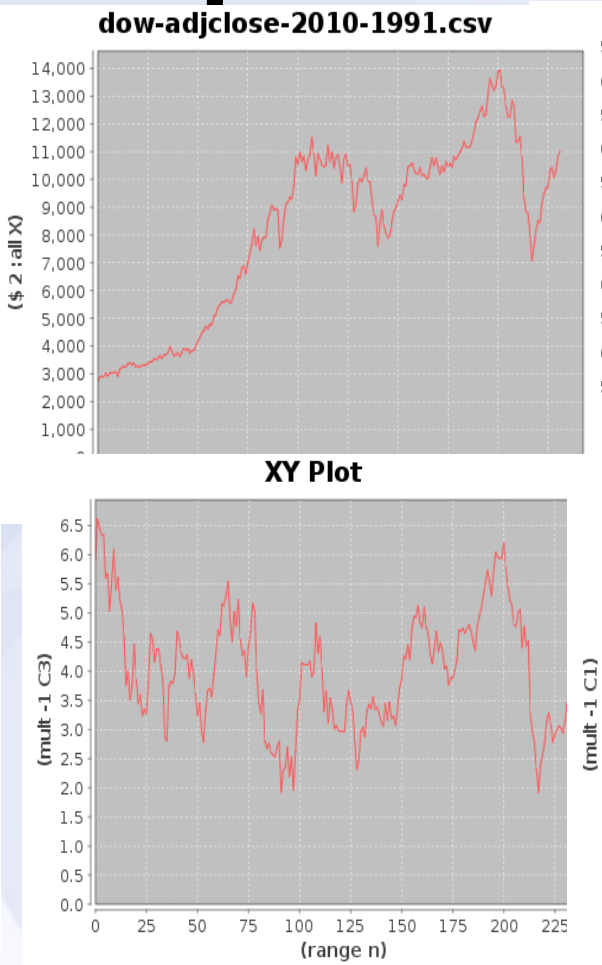
Components

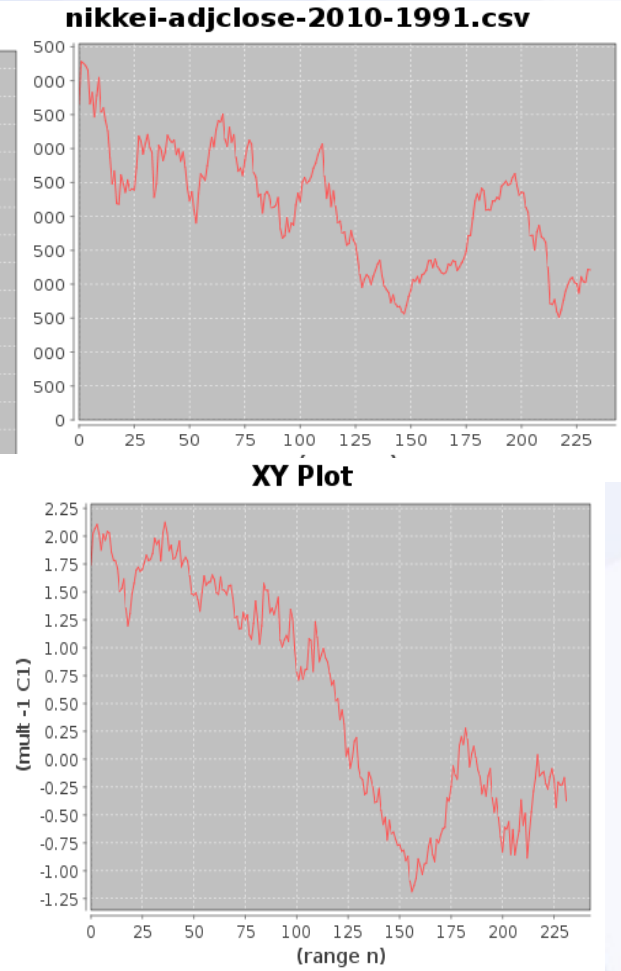

XY Plot
(mult -1 C1)
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
-0.25
-0.50
XY Plot
C2
7.5
7.0
6.5
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
25
50
75
100
125
150
175
200
225
(range n)


XY Plot
(mult -1 C3)
6.5
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
25
50
75
100
125
150
175
200
225
(range n)

Comparison


ftse-adjclose-2010-1991.csv
($ 1 : all X)
XY Plot
C2
(range n)


dow-adjclose-2010-1991.csv
($ 2 : all X)
XY Plot
(mult -1 C3)
(range n)


nikkei-adjclose-2010-1991.csv
XY Plot
(mult -1 C1)
(range n)

Stock market indexes

- Not much can be concluded currently
  - Requires more domain/history knowledge
  - More tests need to be carried out
  - Maybe it shows the unpredictability of the stock market

Future work and conclusion

- ICA successfully implemented in Clojure in the simplest version
  - Satisfying result for finding few components
  - Need more experiments on different and larger data sets
  - Implement more contrast functions
  - Implement symmetric decorrelation

Questions?

Thank you