

# CHREST Tutorial

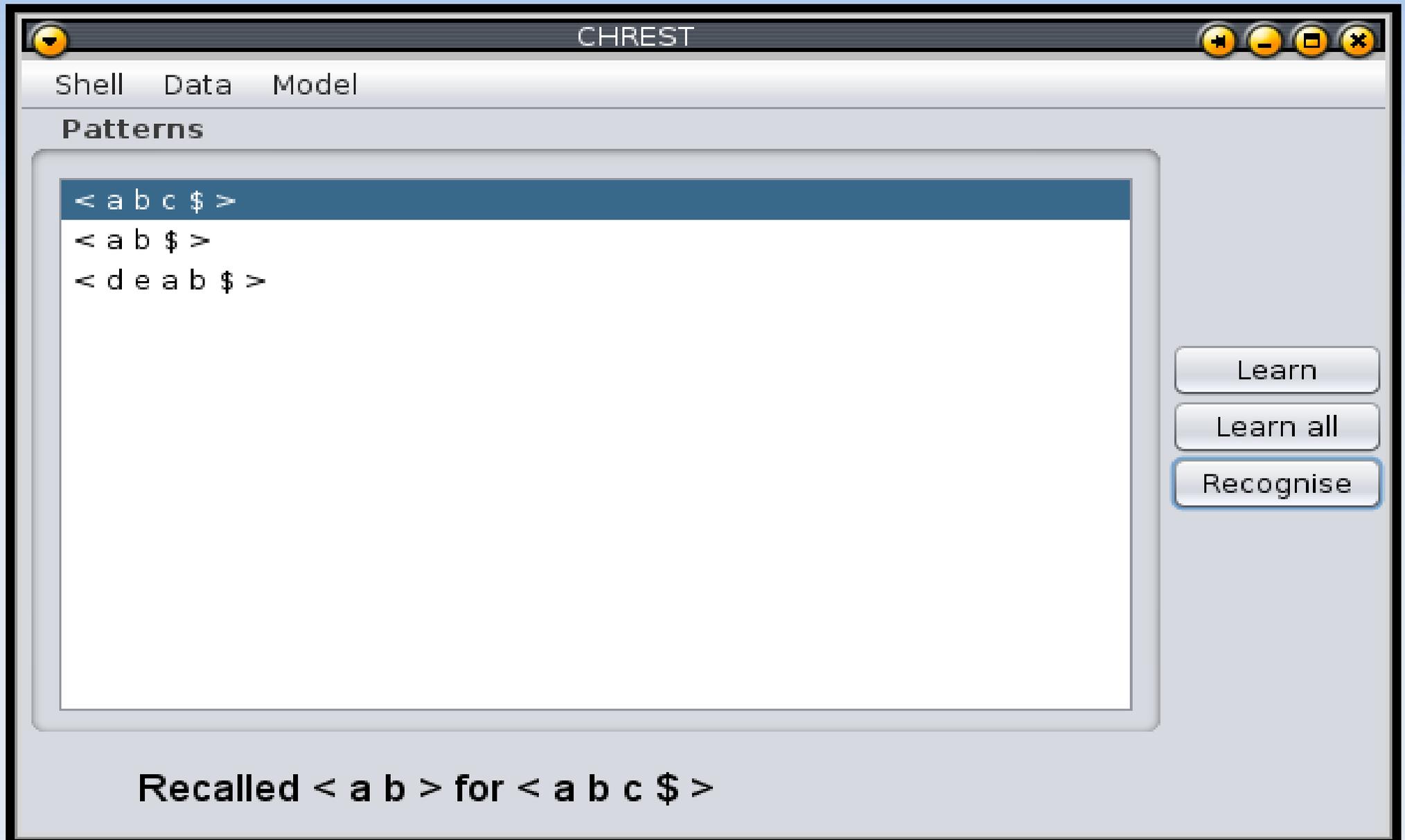
## Working with your own data

- **Data format to load into interface**
- **Developing your own models**

# Data Format

- Supported domains
  - recognition only
  - serial anticipation
  - paired associate
  - categorisation
  - visual attention and recall
- Simple text format for graphical interface
  - first line determines domain
  - second etc lines provide data
  - patterns given as list of atoms, space separated
  - end marker added for you

# Learn and Recognise



The screenshot shows a window titled "CHREST" with a menu bar containing "Shell", "Data", and "Model". Below the menu bar is a section titled "Patterns" containing a list of three patterns:

- < a b c \$ >
- < a b \$ >
- < d e a b \$ >

The first pattern, "< a b c \$ >", is highlighted in blue. To the right of the pattern list are three buttons: "Learn", "Learn all", and "Recognise". The "Recognise" button is currently selected, indicated by a blue border. At the bottom of the window, the text "Recalled < a b > for < a b c \$ >" is displayed.

# Learn and Recognise: File Format

recognise-and-learn

a b c

a b

d e a b

# Serial Anticipation or Paired Associate Learning

CHREST

Shell Data Model

### Stimulus-response pairs

< G X J \$ >	< W A P \$ >
< Z X K \$ >	< S O K \$ >
< G X K \$ >	< Q I L \$ >
< L Q F \$ >	< D A G \$ >
< G X F \$ >	< B I F \$ >
< L X J \$ >	< R O V \$ >
< Z H J \$ >	< S A J \$ >
< M B W \$ >	< B I P \$ >
< G Q K \$ >	< W E K \$ >

Experiment time (ms) 1400000

End trial time (ms)

Inter item time (ms)

Random order

### Protocol

Trial 31	Trial 32	Trial 33	Trial 34	Trial 35
< W >	< W >	< W A >	< W A >	< W A >
< NONE >	< S O K \$ >	< S O K \$ >	< S O K \$ >	< S O K \$ >
< NONE >	< NONE >	< NONE >	< Q I L \$ >	< Q I L \$ >
< D \$ >				
< B I F \$ >				
< R O >	< R O >	< R O >	< R O >	< R O >
< S A >	< S A >	< S A >	< S A >	< S A >
< B I F \$ >				
< W \$ >				
8	7	7	6	6

# Serial Anticipation: File Format

serial-anticipation

D A G

B I F

G I H

J A L

M I Q

P E L

S U J

# Paired Associate : File Format

paired-associate

G X J : W A P

Z X K : S O K

G X K : Q I L

L Q F : D A G

G X F : B I F

L X J : R O V

Z H J : S A J

M B W : B I P

G Q K : W E K

# Categorisation

The screenshot shows the CHREST software interface. The window title is "CHREST". It has three menu items: "Shell", "Data", and "Model".

The interface is divided into two main panels:

- Categorisation data:** A list of 16 items, each consisting of a source string, a target string, and a "Training" checkbox. The first 10 items are checked, and the last 6 are unchecked.
- Protocol:** A table with 5 columns: "Source", "Target", "Trial 1", "Trial 2", and "Trial 3". It contains 16 rows of data, with the 10th row highlighted in blue. Below the table, there is a summary row for "Errors" with values 16, 14, and 13.

At the bottom left, there is a checked checkbox for "Random order". At the bottom center, there are two buttons: "Restart" and "Run Trial".

Source	Target	Trial 1	Trial 2	Trial 3
< 1 1 1 0 \$ >	< A \$ >	< NONE >	< A \$ >	< A \$ >
< 1 0 1 0 \$ >	< A \$ >	< NONE >	< NONE >	< NONE >
< 1 0 1 1 \$ >	< A \$ >	< NONE >	< NONE >	< NONE >
< 1 1 0 1 \$ >	< A \$ >	< NONE >	< A \$ >	< A \$ >
< 0 1 1 1 \$ >	< A \$ >	< NONE >	< NONE >	< A \$ >
< 1 1 0 0 \$ >	< B \$ >	< NONE >	< A \$ >	< A \$ >
< 0 1 1 0 \$ >	< B \$ >	< NONE >	< NONE >	< A \$ >
< 0 0 0 1 \$ >	< B \$ >	< NONE >	< NONE >	< NONE >
< 0 0 0 0 \$ >	< B \$ >	< NONE >	< NONE >	< NONE >
< 1 1 1 1 \$ >	< X \$ >	< NONE >	< A \$ >	< A \$ >
< 1 0 0 1 \$ >	< X \$ >	< NONE >	< NONE >	< NONE >
< 1 0 0 0 \$ >	< X \$ >	< NONE >	< NONE >	< NONE >
< 0 1 0 1 \$ >	< X \$ >	< NONE >	< NONE >	< A \$ >
< 0 1 0 0 \$ >	< X \$ >	< NONE >	< NONE >	< A \$ >
< 0 0 1 ... >	< X \$ >	< NONE >	< NONE >	< NONE >
< 0 0 1 ... >	< X \$ >	< NONE >	< NONE >	< NONE >
Errors:		16	14	13

# Categorisation: File Format

categorisation

1 1 1 0 : A

1 0 1 0 : A

1 0 1 1 : A

1 1 0 1 : A

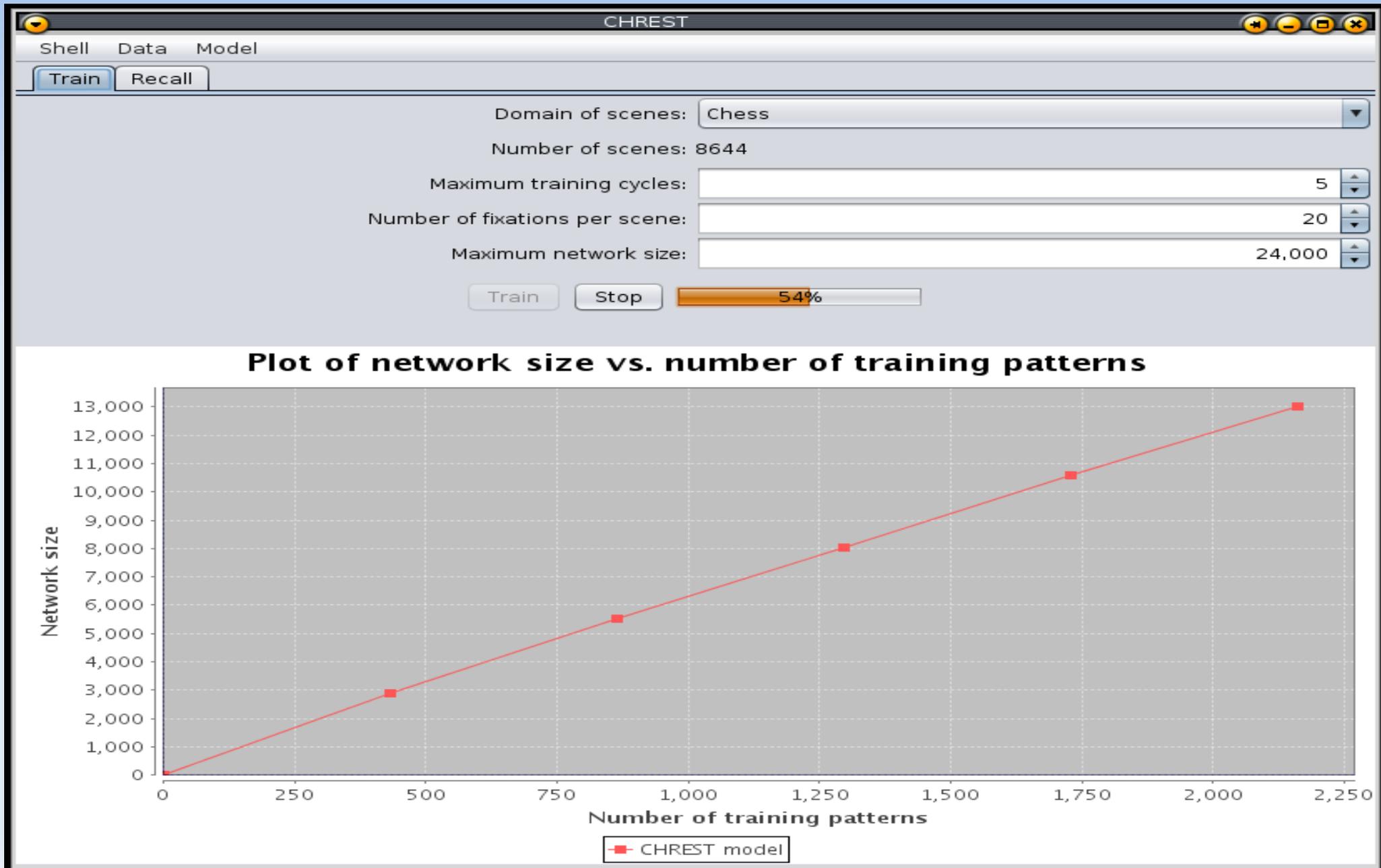
0 1 1 1 : A

1 1 0 0 : B

0 1 1 0 : B

0 0 0 1 : B

# Visual Attention and Recall



# Visual Attention: File Format

visual-search

8 8

. . r . . . . .

pb . . . pkp

. p . pq . . .

n . . p . . p .

. . . P . . . . .

Q . . NP . P .

PP . . . PBP

. . . . . RK .

File type

size: number of rows and columns

blank line

items on row, 'dot' for space

... (repeat all rows)

blank line

next scene

... (repeat scenes)

# Writing New Models

- Why write models?
  - CHREST is a flexible learning algorithm, capable of coping with a variety of data.
  - Many experiments have complex manipulations, and require multiple runs, which are tedious to do by hand.
  - Complex models of attention require coding of domain-specific heuristics.
- How to write models?
  - As CHREST is implemented in Java, you can use CHREST as a library to your favourite language
  - Depending on the language you choose, you might be able to take advantage of a development environment

# Ruby and Netbeans

The screenshot shows the NetBeans IDE interface with the following components:

- Project Explorer:** Lists various projects, with **TestJRuby** selected.
- Main Editor:** Displays the code in `main.rb`.

```
# Train the model a few times on the patterns
4.times do
  for pat in Patterns
    model.recogniseAndLearn pat
  end
end

# Display the results
puts "Current model time: #{model.getClock}"
for pat in Patterns
  print "For pattern: #{pat.toString} model retrieves "
  puts "#{model.recallPattern(pat).toString}"
end

# And display the Model in a graphical view
ChrestView.new(nil, model)
```
- Output Window:** Shows the execution results for `TestJRuby`.

```
Current model time: 72000
For pattern: < 1 2 3 > model retrieves < 1 2 >
For pattern: < 1 3 2 > model retrieves < 1 3 2 >
For pattern: < 2 1 3 > model retrieves < 2 >
```

# Lisp Environment

The screenshot shows a Lisp environment window with the following components:

- Title Bar:** J 0.23.0
- Menu Bar:** File Edit View Search Go Mode Lisp Help
- Toolbar:** Contains icons for file operations (star, folder, download, close, undo, redo, cut, copy, paste, zoom, home, refresh).
- Buffers (4):**
  - classification.lisp
  - chess-attention.lisp
  - jlist
  - chrest-system.lisp
- Code Editor:** Location: /home/peter/Projects/JChrest/scripts/Lisp/chess-attention.lisp. The code is as follows:

```
;;; classification.lisp
;;; Written by Peter Lane, 2010.
;;; Simple illustration of using Chrest to perform classification

(load "chrest-system")
(use-package :chrest)

(defstruct example features class)

(defun construct-patterns (days)
  "Construct a list pattern given a list of day definitions"
  (mapcar #'(lambda (day)
            (make-example
             :features (make-list-pattern
                       (list (concatenate 'string "outlook-" (nth 0 day))
                             (concatenate 'string "temperature-" (nth 1 day))
                             (concatenate 'string "humidity-" (nth 2 day))
                             (concatenate 'string "windy-" (nth 3 day))))
             :class (make-name-pattern (nth 4 day))))
          days))
```
- Output Window:** Location: jlist. The output is as follows:

```
[1] CL-USER(8): *WEATHER*
[1] CL-USER(9): Performance on cycle 1 is: 0.00
Performance on cycle 2 is: 0.00
Performance on cycle 3 is: 0.00
Performance on cycle 4 is: 0.00
Performance on cycle 5 is: 0.00
Performance on cycle 6 is: 0.00
Performance on cycle 7 is: 0.36
```
- Status Bar:** \*weather\* Lisp Line 56 Col 1

# Examples on CDROM

- See the manual for more suggestions
- See the 'scripts' folder:
  - Detailed examples are given in Ruby and Lisp
  - Some examples for Groovy, Clojure and Scheme.
- The software folder includes a 'doc' folder which is the javadoc documentation for CHREST.
- Do ask if needed – peter.lane 'at' bcs.org.uk