

CHREST Tutorial

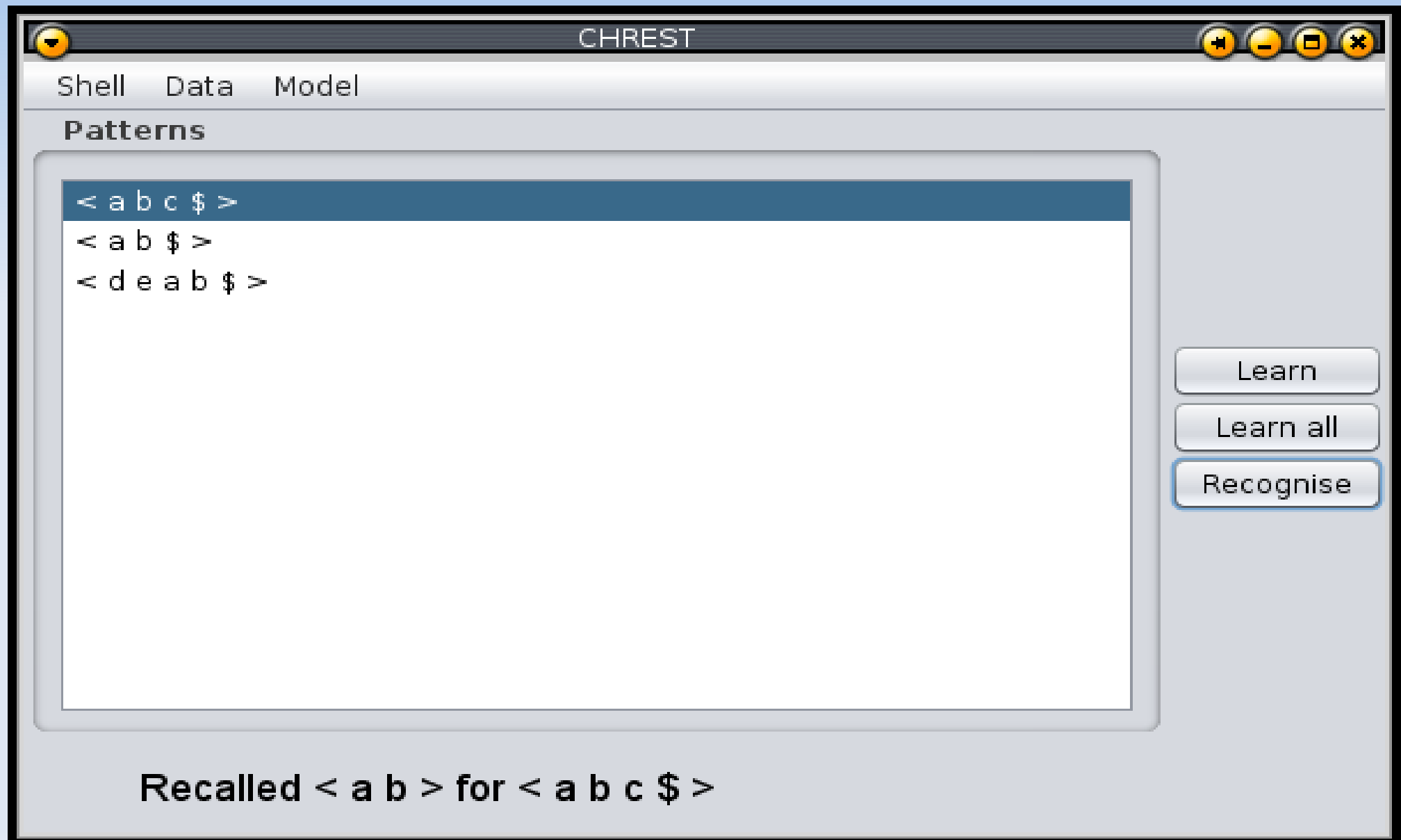
Working with your own data

- **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



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) 2,000

Inter item time (ms) 2,000

☒ Random order

Restart

Run Trial

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 \$ >	< D \$ >	< D \$ >	< D \$ >	< D \$ >
< B I F \$ >	< B I F \$ >	< B I F \$ >	< B I F \$ >	< 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 \$ >	< B I F \$ >	< B I F \$ >	< B I F \$ >	< B I F \$ >
< W \$ >	< W \$ >	< W \$ >	< W \$ >	< 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 has a title bar with the name 'CHREST' and standard window controls. Below the title bar is a menu bar with 'Shell', 'Data', and 'Model'. The main area is divided into two panels: 'Categorisation data' on the left and 'Protocol' on the right.

Categorisation data panel: This panel contains a list of 16 items, each consisting of a binary string followed by a target label and a 'Training' checkbox. The items are:

- <1110\$> <A\$> ☒ Training
- <1010\$> <A\$> ☒ Training
- <1011\$> <A\$> ☒ Training
- <1101\$> <A\$> ☒ Training
- <0111\$> <A\$> ☒ Training
- <1100\$> <B\$> ☒ Training
- <0110\$> <B\$> ☒ Training
- <0001\$> <B\$> ☒ Training
- <0000\$> <B\$> ☒ Training
- <1111\$> <X\$> ☐ Training
- <1001\$> <X\$> ☐ Training
- <1000\$> <X\$> ☐ Training
- <0101\$> <X\$> ☐ Training
- <0100\$> <X\$> ☐ Training

At the bottom of this panel, there is a checkbox for 'Random order' (checked) and two buttons: 'Restart' and 'Run Trial'.

Protocol panel: This panel contains a table with 5 columns: 'Source', 'Target', 'Trial 1', 'Trial 2', and 'Trial 3'. The table has 16 rows of data, with the 9th row highlighted in blue. Below the table, there is a row labeled 'Errors:' with values 16, 14, and 13 under the last three columns.

Source	Target	Trial 1	Trial 2	Trial 3
<111...	<A\$>	<NONE>	<A\$>	<A\$>
<101...	<A\$>	<NONE>	<NONE>	<NONE>
<101...	<A\$>	<NONE>	<NONE>	<NONE>
<110...	<A\$>	<NONE>	<A\$>	<A\$>
<011...	<A\$>	<NONE>	<NONE>	<A\$>
<110...	<B\$>	<NONE>	<A\$>	<A\$>
<011...	<B\$>	<NONE>	<NONE>	<A\$>
<000...	<B\$>	<NONE>	<NONE>	<NONE>
<000...	<B\$>	<NONE>	<NONE>	<NONE>
<111...	<X\$>	<NONE>	<A\$>	<A\$>
<100...	<X\$>	<NONE>	<NONE>	<NONE>
<100...	<X\$>	<NONE>	<NONE>	<NONE>
<010...	<X\$>	<NONE>	<NONE>	<A\$>
<010...	<X\$>	<NONE>	<NONE>	<A\$>
<001...	<X\$>	<NONE>	<NONE>	<NONE>
<001...	<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

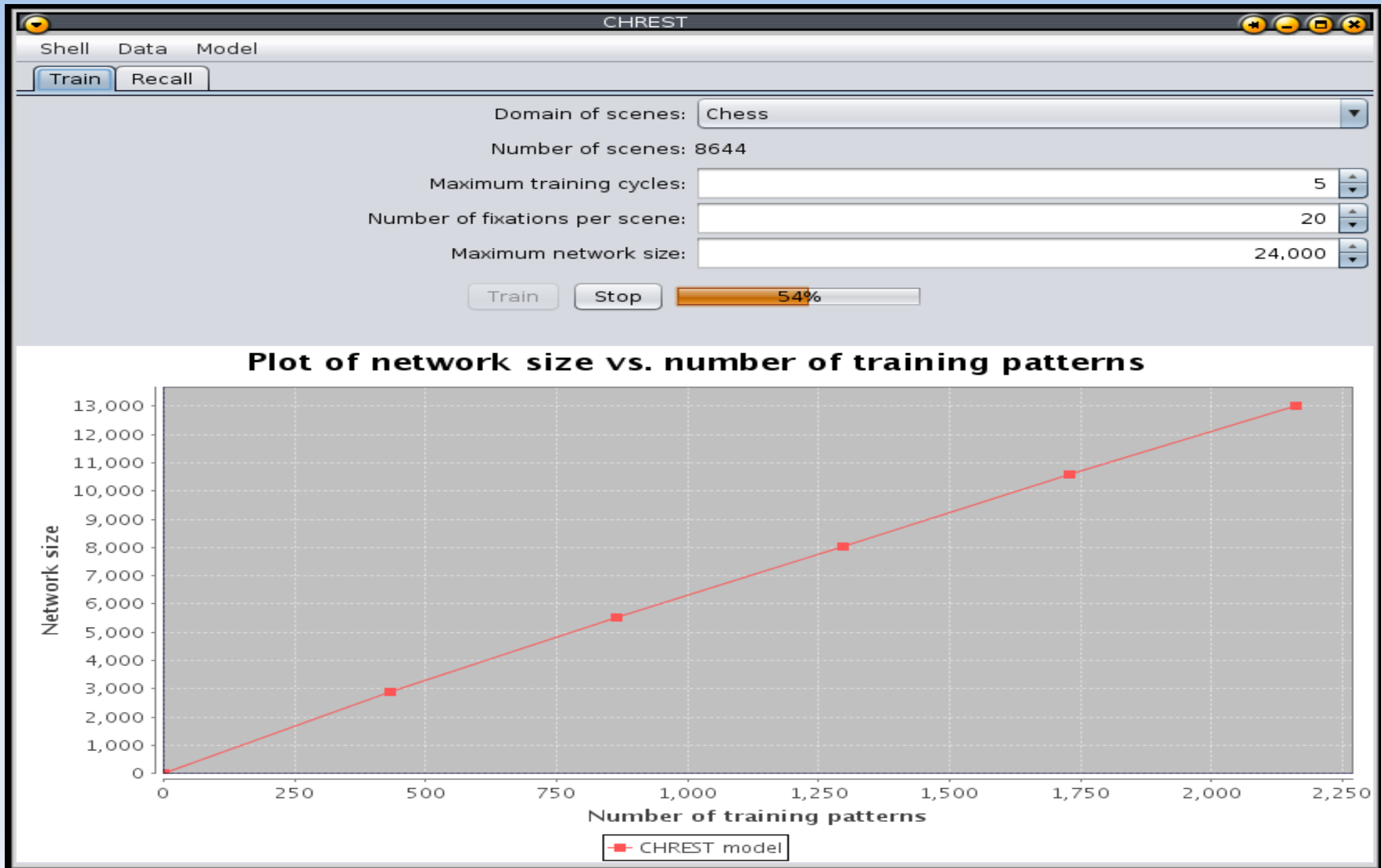
0 1 1 1 : A

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 6.7.1 interface. The main editor window displays a Ruby script named `main.rb` with the following code:

```
# 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)
```

The left sidebar shows a project tree with the following files:

- Clock
- CollectionsListExamp
- CollectionsMapExamp
- CollectionsSetExamp
- Converter
- Demo
- FoodAssignment2
- FrequencyBag
- Game
- GraphicalInterfaces2
- GraphicInterfaces1
- HelloWorld
- InterfaceExample
- JavaThreads1
- JavaThreads2
- ObserverExample
- Refactoring1
- Refactoring2
- SayHello
- TestJRuby**
- TestJRuby2
- UMLProject1
- Unit1Exercises
- Unit1Floats
- Unit1References
- Unit1SimpleFactorial
- Unit8Test

The bottom pane shows the Test Results output for the `TestJRuby` project:

```
Test Results
Output - TestJRuby
Tasks

/home/peter/Code/Java/Downloads/jruby-1.4.0RC1/lib/ruby/site_ruby/shared/builtin/javasupport/core_ext/
/home/peter/Code/Java/Downloads/jruby-1.4.0RC1/lib/ruby/site_ruby/shared/builtin/javasupport/core_ext/
/home/peter/Code/Java/Downloads/jruby-1.4.0RC1/lib/ruby/site_ruby/shared/builtin/javasupport/core_ext/
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 a menu bar (File, Edit, View, Search, Go, Mode, Lisp, Help) and a toolbar. The main editor displays the contents of `classification.lisp`, which defines a function `construct-patterns` to generate weather patterns for classification. The left sidebar shows the file buffers and the current buffer's contents. The bottom pane shows the execution results of the code.

Location: /home/peter/Projects/JChrest/scripts/Lisp/chess-attention.lisp

```
;;; 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))
```

Location: jlisp

```
[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
```

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](mailto:peter.lane@bcs.org.uk)