Contents

1 DE Introduction

To be able to evaluate lisp in babel blocks you must invoke slime.

```
(format t "~a~%" (list 1 2 3))
```

1.1 Can I make a simple plot?

1.1.1 First we need a bunch of stuff

```
     ;; will need some packages/systems I am sure.
     ;; will try gnuplot -- need to install that package
     ;; and eazy-gnuplot
(ql:quickload '(:eazy-gnuplot :clml.statistics :clml.utility))
(use-package :eazy-gnuplot)
```

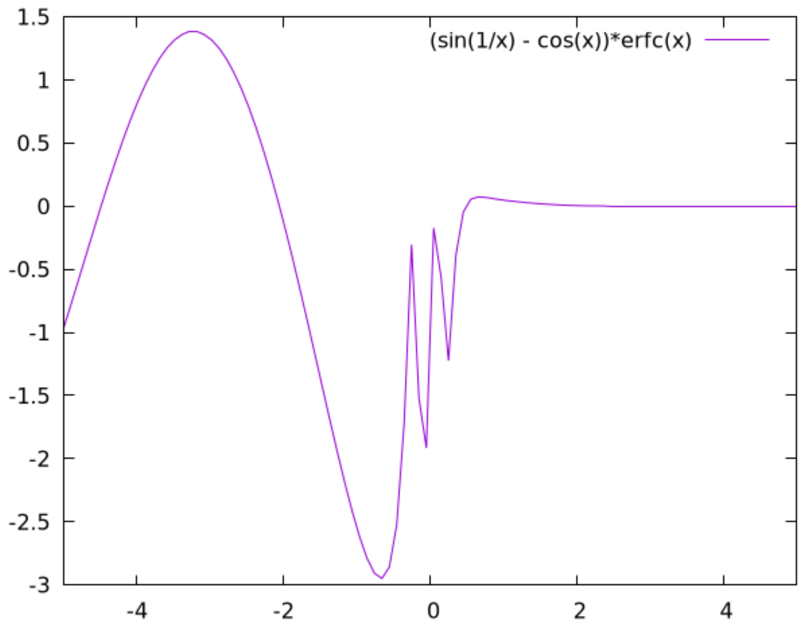
1.1.2 Now a basic example?

Here is a "cookbook".

The examples make use of a local subdirectory called "images". Create it.

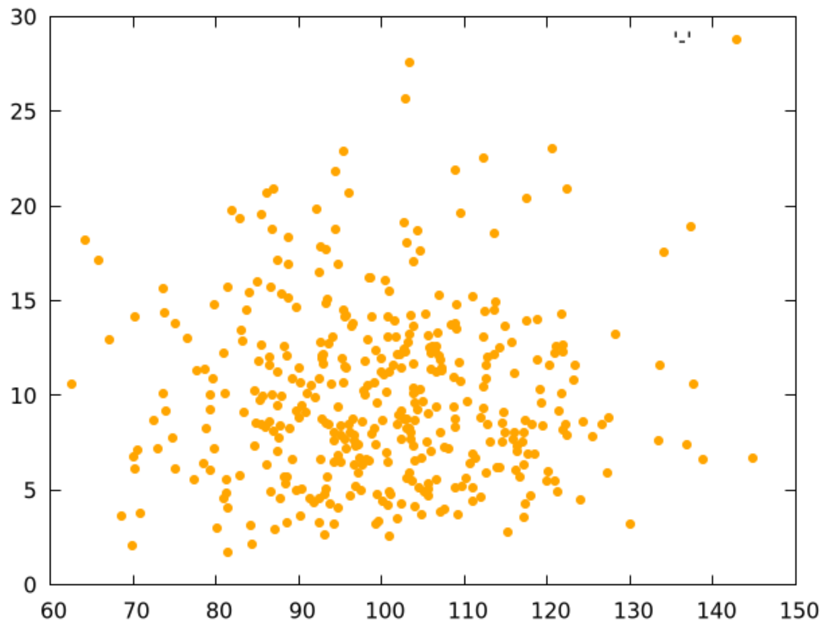
The first cookbook example seems to use a function we don't need: `png-from-file`. This is probably related to the cookbook being a jupyter notebook.

```
(defun function-plot (output)
  (with-plots (*standard-output* :debug nil)
    (gp-setup :terminal '(pngcairo) :output output)
    (func-plot "[-5:5] (sin(1/x) - cos(x))*erfc(x)"))
  output)
(function-plot "images/function-plot.png")
```



1.1.3 Scatter Plot Example

```
(defun scatter-plot (output)
  (let ((point-n 400)
(point-type 7)
(point-color "orange"))
    (with-plots (*standard-output* :debug nil)
      (gp-setup :terminal '(pngcairo) :output output)
      (plot
       (lambda ()
 (loop for p in (map 'list (lambda (x y) (list x y))
   (clml.statistics:rand-n
    (clml.statistics:chi-square-distribution 100) point-n)
   (clml.statistics:rand-n
    (clml.statistics:chi-square-distribution 10) point-n))
     do (format t "~&~{~a~^ ~}" p)))
       :with `(:points :pt ,point-type :lc :rgb ,point-color)))
  output))
 (scatter-plot "./images/scatter-plot.png")
```



''
30
25
20
15
10
5
0
60
70
80
90
100
110
120
130
140
150