St. Louis
Clojure

Clojure Incanter

Christopher Mark Gore

cgore.com

Tuesday, April 28, AD 2015

Why Incanter?

- charts
- statistics
- data
- graphics
- don’t have to use R or MATLAB!

Getting Started: Your `project.clj`

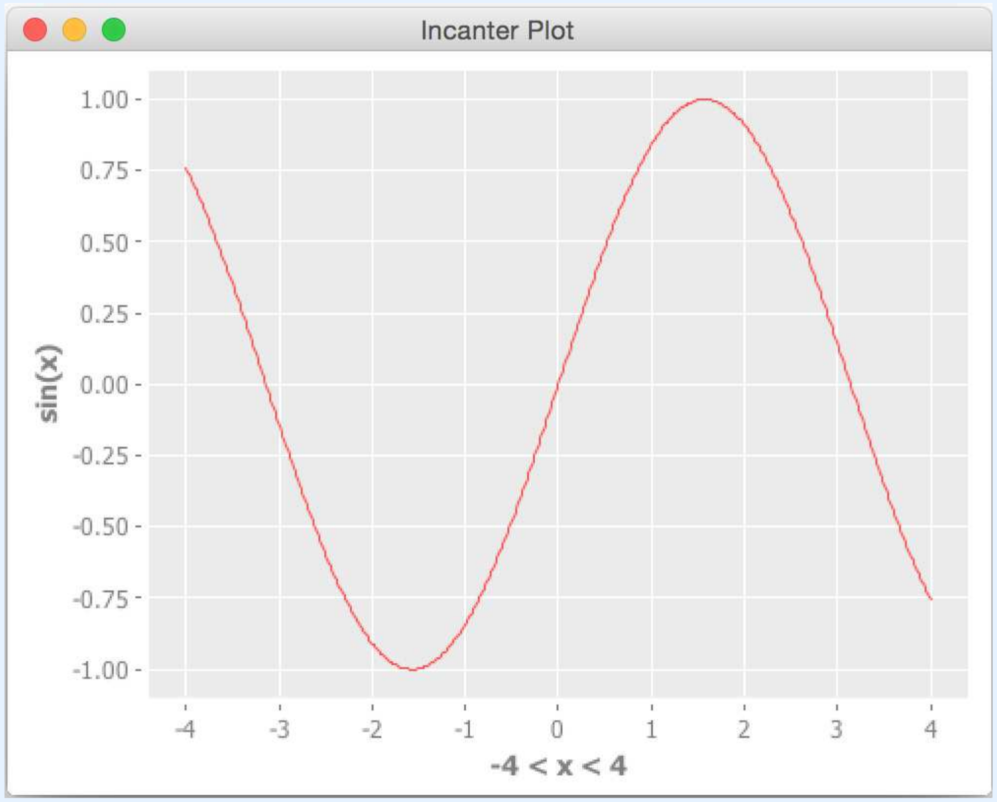
```
:dependencies [... [incanter "1.5.6"] ...]
```

Getting Started: Your Namespace Declaration

```
(ns code.core
  "Howdy␣Incanter!"
  (:require [incanter.core :as i
              :refer [$ conj-cols conj-rows dataset
                      dim save to-dataset view]]
            [incanter.datasets :as ids]
            [incanter.stats :as is]
            [incanter.charts :as ic
                              :refer [histogram]]
            [incanter.io :as iio
                          :refer [read-dataset]]))
```

Sine Waves

```
(view (ic/function-plot #(Math/sin %) -4 4
                        :y-label "sin(x)"))
```



Data Sets

You probably want to look at data if you are interested in Incanter. For a really small data set, you might just define it inline.

```
(def small-data (dataset ["x" "y" "theta"]
                          [[1   2   3]
                           [4   5   6]
                           [7   8   9]]))
```

Data Sets from CSVs

If you are working with a real data set, then it's probably living in a CSV file or a database.

```
(def pass-data (read-dataset "../Pass.csv"
                             :header true))
(def fail-data (read-dataset "../Pass.csv"
                             :header true))
```

Data Sets from Hash Maps

Clojure *loves* hash maps. How do you make a data set out of them?

```
(def data-from-hashmaps (to-dataset [{:x 1 :y 2}
                                     {:x 3 :y 4}
                                     {:x 5 :y 6}]))
```

Data Sets from Vectors

```
(def data-from-vecs (to-dataset [[1 2 3]
                                 [4 5 6]
                                 [7 8 9]]))
(def data-cols (conj-cols [1 4 7]
                          [2 5 8]
                          [3 6 9]))
(def data-rows (conj-rows [1 2 3]
                          [4 5 6]
                          [7 8 9]))
```

Data Sets from the Internet

There's no need to download the CSV, if you know the path to it.

```
(def air-passengers
  (read-dataset
   (str "http://vincentarelbundock.github.io"
        "/Rdatasets/csv/datasets/AirPassengers.csv")
   :header true))
```

Included Sample Data Sets

Incanter has a lot of sample data sets included, mostly borrowed from R. Standard data sets are commonly used if you need to test out an algorithm, or compare it to existing algorithms.

```
(def sample-data (ids/get-dataset :hair-eye-color))
```

$$\begin{pmatrix} :hair & :eye & :gender & :count \\ black & brown & male & 32 \\ black & blue & male & 11 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Saving Data Sets

Incanter provides an easy way to save your data sets to CSV files for use in other tools.

```
(save some-data "some.csv")
```

The $ Operator

The $ operator is a shortcut to get that column of data out of a dataset.

```
(defn x [dataset]
  ($ :x dataset))
(defn y [dataset]
  ($ :y dataset))
(defn theta [dataset]
  ($ :theta dataset))
(defn mpi [dataset]
  ($ (keyword "Monthly␣Personal␣Income") dataset))
```

Multiple Columns with the $ Operator

To select a few columns:

```
($ ["x" "y"] small-data)
```

To remove one of the columns:

```
($ [:not "theta"] small-data)
```

Both produce:

$$\begin{pmatrix} x & y \\ 1 & 2 \\ 4 & 5 \\ 7 & 8 \end{pmatrix}$$

Single Rows with the $ Operator

We can select a few columns:

```
($ ["x" "y"] small-data)
```

$$\begin{pmatrix} x & y \\ 1 & 2 \\ 4 & 5 \\ 7 & 8 \end{pmatrix}$$

And then select a single row, zero-indexed:

```
($ 1 ["x" "y"] small-data) ;; Returns '(4 5)
```

Statistics

There is a lot of statistics available. Some of the basics:

```
(def mpi-stats {:mean (is/mean mpi-filtered)
                :variance (is/variance mpi-filtered)
                :std-dev (is/sd mpi-filtered)
                :median (is/median mpi-filtered)
                :kurtosis (is/kurtosis mpi-filtered)})
```

The `$where` Operator

The `$order` Operator

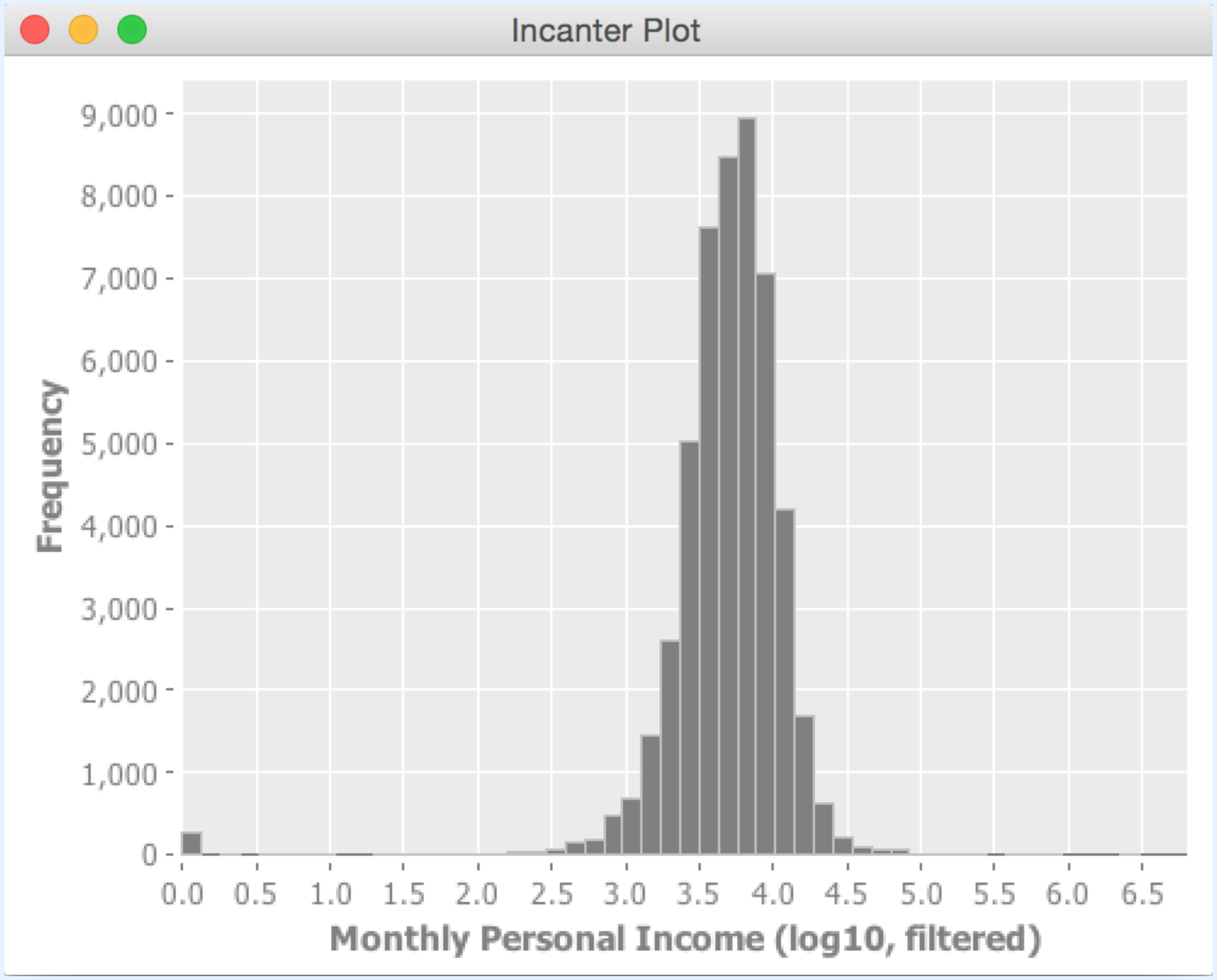
The `$rollup` Operator

Histograms

What's my data look like?

```
(let [mpi-filtered (filter #(< 0 %) (mpi pass-data))
      mpi-log10 (map #(Math/log10 %) mpi-filtered)]
  (view (histogram mpi-log10
                   :x-label "Monthly␣Personal␣Income"
                   :nbins 50)))
```

Histograms


Incanter Plot
Frequency
9,000
8,000
7,000
6,000
5,000
4,000
3,000
2,000
1,000
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
Monthly Personal Income (log10, filtered)

Scatter Plots

Bar Charts

Line Charts

Box Plots

Questions?