

Data Sorcery with  
**Clojure & Incanter**

*Introduction to  
Datasets & Charts*

National Capital Area Clojure Meetup  
18 February 2010

David Edgar Liebke  
[liebke@incanter.org](mailto:liebke@incanter.org)



## Overview

- What is Incanter?
- Getting started
- Incanter libraries

## Datasets

- Creating
- Reading data
- Saving data
- Column selection
- Row selection
- Sorting
- Rolling up

## Charts

- Scatter plots
- Chart options
- Saving charts
- Adding data
- Bar & line charts
- XY & function plots
- Histograms & box-plots
- Annotating charts

## Wrap up



# Outline

---

## Overview

- What is Incanter?
- Getting started
- Incanter libraries

## Datasets

- Creating
- Reading data
- Saving data
- Column selection
- Row selection
- Sorting
- Rolling up

## Charts

- Scatter plots
- Chart options
- Saving charts
- Adding data
- Bar & line charts
- XY & function plots
- Histograms & box-plots
- Annotating charts

## Wrap up



## **Incanter is a Clojure-based, R-like platform for statistical computing and graphics.**

It can be used either as a standalone, interactive data analysis environment or embedded within other analytics systems as a modular suite of libraries.

### **Features**

- Charting & visualization functions
- Data manipulation functions
- Mathematical functions
- Statistical functions
- Matrix & linear algebra functions
- Machine learning functions



*As of 6 April 2010*

## Committers

- David Edgar Liebke
- Bradford Cross (FlightCaster)
- Alex Ott (build master)
- Tom Faulhaber (autodoc)
- Sean Devlin (chrono 2.0)
- Jared Strate (FlightCaster)

## Contributors

- Felix Breuer
- Chris Burroughs
- Michael Fogus
- Phil Hagelberg
- Edmund Jackson (proposed time-series)
- Steve Jenson
- Jake McCrary
- Steve Purcell
- Brendan Ribera
- Alexander Stoddard

## Community

- Incanter Google group (139 members)
- Github repository (237 watchers, 22 forks)

## Related project

- Rincanter (Joel Boehland)



**Build a system** that has R's strengths as an interactive data analysis environment but that is also well suited to building stand-alone and distributed applications.

**Build a community** of data-geeks around Clojure by providing a core set of numerical, statistical, data manipulation, and visualization functions that others can extend and specialize.



Incanter: Statistical Computing and Graphics Environment for Clojure

http://incanter.org/

Documentation | Blog | Code | Discussion | Twitter | T-shirts

**INCANTER**  
Data Sorcery

Google Custom Search Search

**Incanter** is a Clojure-based, R-like platform for statistical computing and graphics.

Incanter can be used as a standalone, interactive data analysis environment or embedded within other analytics systems as a modular suite of libraries.

**Features**

- Charting & visualization functions
- Mathematical functions
- Statistical functions
- Matrix & linear algebra functions
- Data manipulation functions

Download INCANTER

Get Started



Data Sorcery with Clojure

http://data-sorcery.org/ RSS Google

## Data Sorcery with Clojure

FRONT PAGE ABOUT TABLE OF CONTENTS BOOK RECOMMENDATIONS

### Dark theme for Incanter charts

February 6, 2010 · 2 Comments

JFreeChart has been a fantastic library, I've been able to include useful charting functionality in [Incanter](#) very quickly because of it, but I'm not a big fan of its default visual theme. Eventually I'd like to create some new themes, or better yet include themes created by others, but in the meantime I have created the `set-theme` function, which accepts a chart and either a keyword indicating a built-in theme or a JFreeChart ChartTheme object, and applies the theme to the chart.

At the moment, the only built-in themes are `:default` and `:dark`, but hopefully that will change in the future.

Here's an example of using `set-theme`. First I'll create a chart with the default theme,

```
1 (use '(incanter core charts datasets))
2
3 (with-data (get-dataset :iris)
4   (view (scatter-plot :Sepal.Length :Sepal.Width :group-by :Species)))
```

### Scatter Plot

4.50 |

#### SEARCH

To search, type and hit enter

#### INCANTER

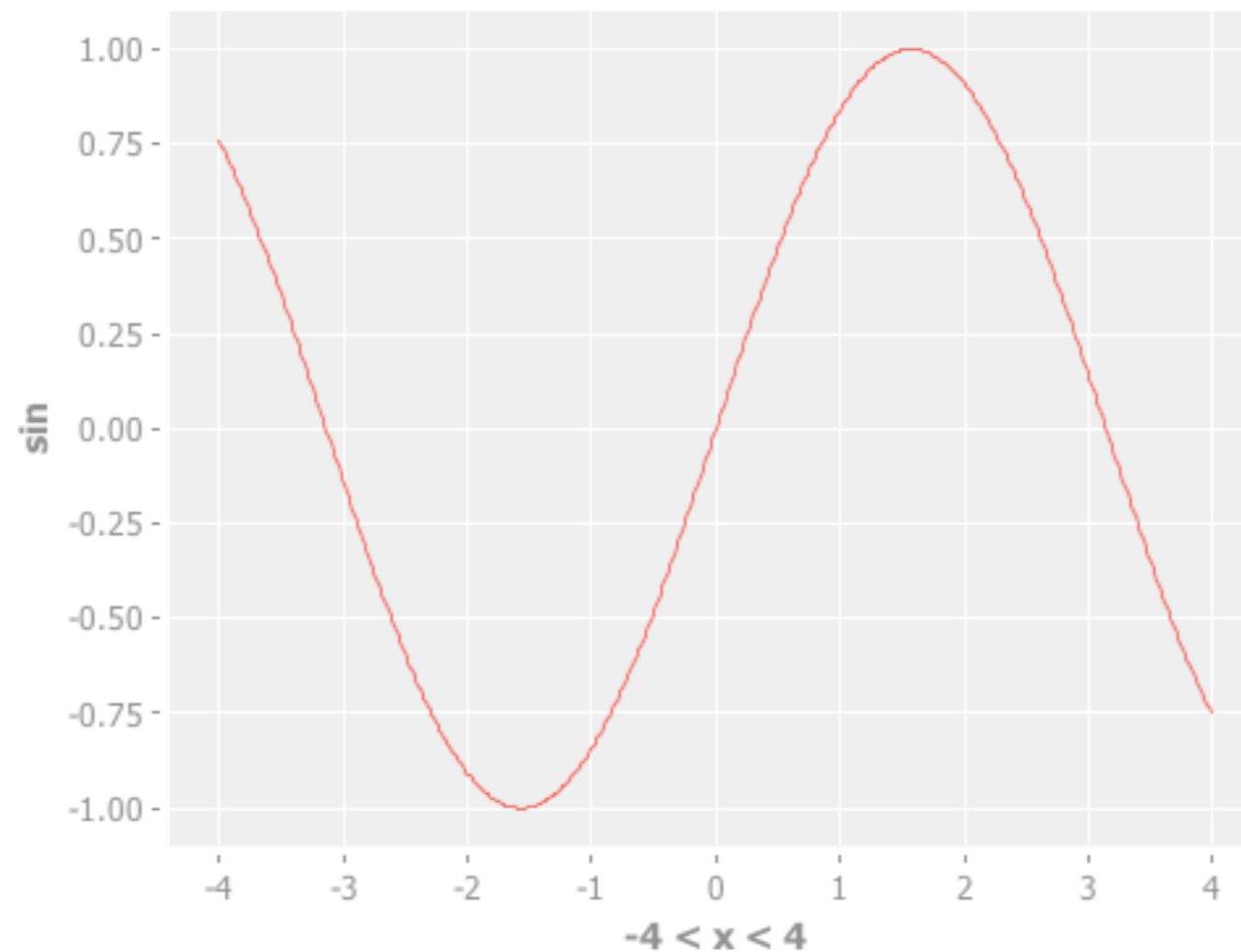
- API documentation
- Github source repository
- Incanter Cheat Sheet
- Incanter Google Group
- Incanter Website
- Incanter Wiki

#### CLOJURE

- Asymmetrical View
- Clojure API documentation
- Clojure - Functional Programming for the JVM
- Clojure Google group
- Clojure on blip.tv
- Clojure Programming wikibook



```
$ wget http://incanter.org/downloads/incanter-latest.zip
$ unzip incanter-latest.zip
$ cd incanter
$ script/repl
user=> (use '(incanter core charts))
user=> (view (function-plot sin -4 4))
```



**Just add the following dependency to your project.clj file:**

```
[incanter "1.2.1-SNAPSHOT"]
```

**To include only a subset of the Incanter libraries, use one of the submodules as a dependency instead:**

```
[incanter/incanter-core "1.2.1-SNAPSHOT"]  
[incanter/incanter-charts "1.2.1-SNAPSHOT"]  
[incanter/incanter-chrono "1.2.1-SNAPSHOT"]  
[incanter/incanter-processing "1.2.1-SNAPSHOT"]  
[incanter/incanter-mongodb "1.2.1-SNAPSHOT"]
```



- **bayes**
- **censored**
- **charts** (based on JFreeChart)
- **classification**
- **core** (based on ParallelColt)
- **datasets**
- **incremental-stats**
- **information-theory**
- **internal**
- **io**
- **mongodb** (based on Congomongo)
- **optimize**
- **pdf** (based on iText)
- **probability**
- **processing** (based on Processing)
- **som**
- **stats** (based on ParallelColt)
- **transformations**



- bayes
- censored
- **charts**
- classification
- **core**
- datasets
- incremental-stats
- information-theory
- internal
- io
- mongodb
- optimize
- pdf
- probability
- processing
- som
- stats
- transformations



- bayes
- censored
- **charts**
- classification
- **core**
- **datasets**
- incremental-stats
- information-theory
- internal
- io
- **mongodb**
- **optimize**
- **pdf**
- probability
- processing
- som
- **stats**
- transformations



# Outline

---

## Overview

- What is Incanter?
- Getting started
- Incanter libraries

## Datasets

- **Creating**
- **Reading data**
- **Saving data**
- **Column selection**
- **Row selection**
- **Sorting**
- **Rolling up**

## Charts

- Scatter plots
- Chart options
- Saving charts
- Adding data
- Bar & line charts
- XY & function plots
- Histograms & box-plots
- Annotating charts

## Wrap up

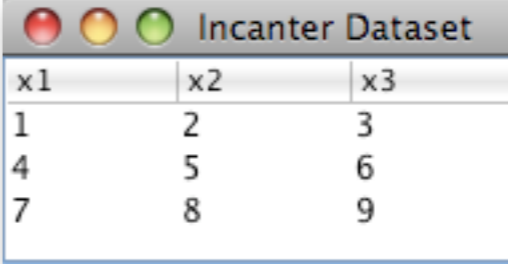


# Creating datasets

*create a small dataset*

```
(use 'incanter.core)
```

```
(dataset ["x1" "x2" "x3"]  
  [[1 2 3]  
   [4 5 6]  
   [7 8 9]])
```



x1	x2	x3
1	2	3
4	5	6
7	8	9

```
(to-dataset [{"x1" 1, "x2" 2, "x3" 3}  
  {"x1" 4, "x2" 5, "x3" 6}  
  {"x1" 7, "x2" 8, "x3" 9}])
```

```
(to-dataset [[1 2 3]  
  [4 5 6]  
  [7 8 9]])
```

```
(conj-cols [1 4 7] [2 5 8] [3 6 9])
```

```
(conj-rows [1 2 3] [4 5 6] [7 8 9])
```



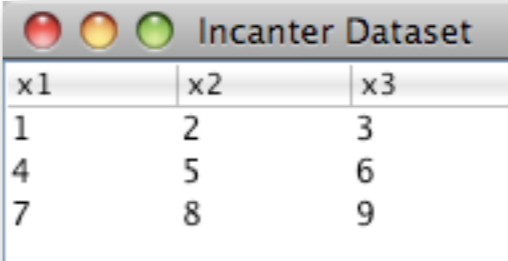
# Creating datasets

*convert a sequence of maps*

```
(use 'incanter.core)
```

```
(dataset ["x1" "x2" "x3"]  
  [[1 2 3]  
   [4 5 6]  
   [7 8 9]])
```

```
(to-dataset [{"x1" 1, "x2" 2, "x3" 3}  
            {"x1" 4, "x2" 5, "x3" 6}  
            {"x1" 7, "x2" 8, "x3" 9}])
```



x1	x2	x3
1	2	3
4	5	6
7	8	9

```
(to-dataset [[1 2 3]  
            [4 5 6]  
            [7 8 9]])
```

```
(conj-cols [1 4 7] [2 5 8] [3 6 9])
```

```
(conj-rows [1 2 3] [4 5 6] [7 8 9])
```



# Creating datasets

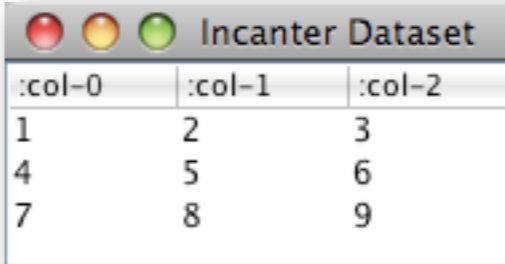
*convert a sequence of sequences*

```
(use 'incanter.core)
```

```
(dataset ["x1" "x2" "x3"]  
        [[1 2 3]  
         [4 5 6]  
         [7 8 9]])
```

```
(to-dataset [{"x1" 1, "x2" 2, "x3" 3}  
            {"x1" 4, "x2" 5, "x3" 6}  
            {"x1" 7, "x2" 8, "x3" 9}])
```

```
(to-dataset [[1 2 3  
            4 5 6  
            7 8 9]])
```



:col-0	:col-1	:col-2
1	2	3
4	5	6
7	8	9

```
(conj-cols [1 4 7] [2 5 8] [3 6 9])
```

```
(conj-rows [1 2 3] [4 5 6] [7 8 9])
```



# Creating datasets

*conj columns together*

```
(use 'incanter.core)
```

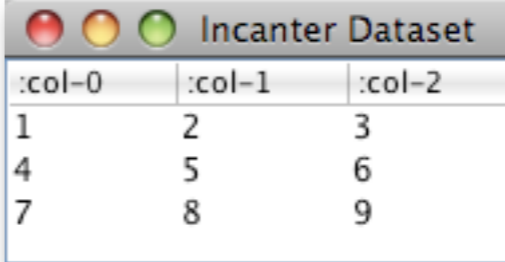
```
(dataset ["x1" "x2" "x3"]  
        [[1 2 3]  
         [4 5 6]  
         [7 8 9]])
```

```
(to-dataset [{"x1" 1, "x2" 2, "x3" 3}  
            {"x1" 4, "x2" 5, "x3" 6}  
            {"x1" 7, "x2" 8, "x3" 9}])
```

```
(to-dataset [[1 2 3]  
            [4 5 6]  
            [7 8 9]])
```

```
(conj-cols [1 4 7] [2 5 8] [3 6 9])
```

```
(conj-rows [1 2 3] [4 5 6] [7 8 9])
```



:col-0	:col-1	:col-2
1	2	3
4	5	6
7	8	9



# Creating datasets

*conj rows together*

```
(use 'incanter.core)
```

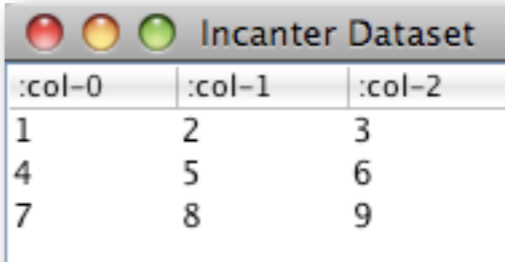
```
(dataset ["x1" "x2" "x3"]  
  [[1 2 3]  
   [4 5 6]  
   [7 8 9]])
```

```
(to-dataset [{"x1" 1, "x2" 2, "x3" 3}  
  {"x1" 4, "x2" 5, "x3" 6}  
  {"x1" 7, "x2" 8, "x3" 9}])
```

```
(to-dataset [[1 2 3]  
  [4 5 6]  
  [7 8 9]])
```

```
(conj-cols [1 4 7] [2 5 8] [3 6 9])
```

```
(conj-rows [1 2 3] [4 5 6] [7 8 9])
```



:col-0	:col-1	:col-2
1	2	3
4	5	6
7	8	9



# Reading data

*read a comma-delimited file*

```
(use ' (incanter core io))
```

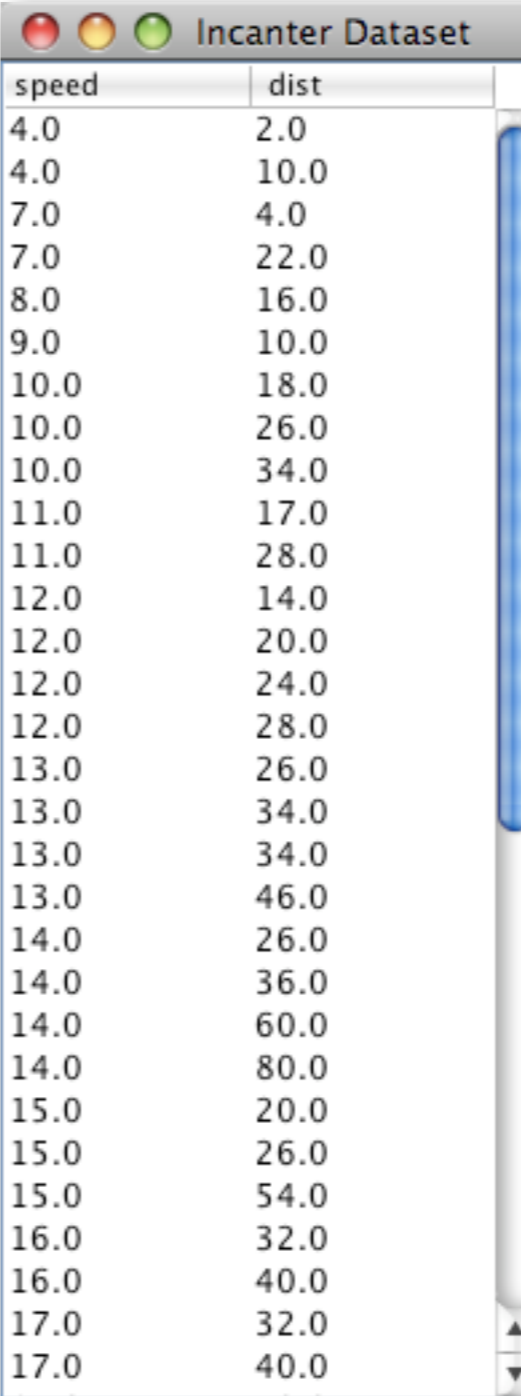
```
(read-dataset "./data/cars.csv"  
             :header true)
```

```
(read-dataset "./data/cars.tdd"  
             :header true  
             :delim \tab)
```

```
(read-dataset "http://bit.ly/aZyjKa"  
             :header true)
```

```
(use 'incanter.datasets)  
(get-dataset :cars)
```

```
(use 'incanter.mongodb)  
(use 'somnium.congomongo)  
(mongo! :db "mydb")  
(view (fetch-dataset :cars))
```



The screenshot shows a window titled "Incanter Dataset" displaying a table with two columns: "speed" and "dist". The table contains 20 rows of data, with the first row being the header. The data points are as follows:

speed	dist
4.0	2.0
4.0	10.0
7.0	4.0
7.0	22.0
8.0	16.0
9.0	10.0
10.0	18.0
10.0	26.0
10.0	34.0
11.0	17.0
11.0	28.0
12.0	14.0
12.0	20.0
12.0	24.0
12.0	28.0
13.0	26.0
13.0	34.0
13.0	34.0
13.0	46.0
14.0	26.0
14.0	36.0
14.0	60.0
14.0	80.0
15.0	20.0
15.0	26.0
15.0	54.0
16.0	32.0
16.0	40.0
17.0	32.0
17.0	40.0



# Reading data

*read a tab-delimited file*

```
(use ' (incanter core io))

(read-dataset "./data/cars.csv"
  :header true)

(read-dataset "./data/cars.tdd"
  :header true
  :delim \tab)

(read-dataset "http://bit.ly/aZyjKa"
  :header true)

(use 'incanter.datasets)
(get-dataset :cars)

(use 'incanter.mongodb)
(use 'somnium.congomongo)
(mongo! :db "mydb")
(view (fetch-dataset :cars))
```



The screenshot shows a window titled "Incanter Dataset" displaying a table with two columns: "speed" and "dist". The table contains 20 rows of data. The "speed" column values range from 4.0 to 17.0, and the "dist" column values range from 2.0 to 80.0. The window has a standard macOS-style title bar with red, yellow, and green buttons.

speed	dist
4.0	2.0
4.0	10.0
7.0	4.0
7.0	22.0
8.0	16.0
9.0	10.0
10.0	18.0
10.0	26.0
10.0	34.0
11.0	17.0
11.0	28.0
12.0	14.0
12.0	20.0
12.0	24.0
12.0	28.0
13.0	26.0
13.0	34.0
13.0	34.0
13.0	46.0
14.0	26.0
14.0	36.0
14.0	60.0
14.0	80.0
15.0	20.0
15.0	26.0
15.0	54.0
16.0	32.0
16.0	40.0
17.0	32.0
17.0	40.0



# Reading data

*read a comma-delimited file from a URL*

```
(use '(incanter core io))

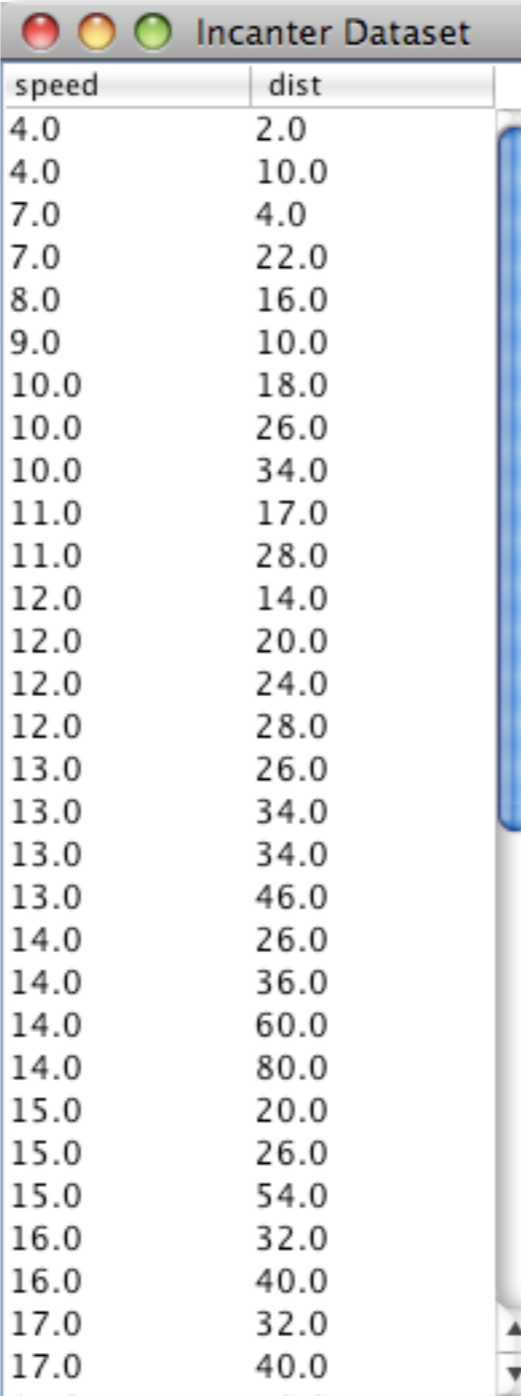
(read-dataset "./data/cars.csv"
              :header true)

(read-dataset "./data/cars.tdd"
              :header true
              :delim \tab)

(read-dataset "http://bit.ly/aZyjKa"
              :header true)

(use 'incanter.datasets)
(get-dataset :cars)

(use 'incanter.mongodb)
(use 'somnium.congomongo)
(mongo! :db "mydb")
(view (fetch-dataset :cars))
```



The screenshot shows a window titled "Incanter Dataset" displaying a table with two columns: "speed" and "dist". The table contains 20 rows of data, with the first row being the header. The data points are as follows:

speed	dist
4.0	2.0
4.0	10.0
7.0	4.0
7.0	22.0
8.0	16.0
9.0	10.0
10.0	18.0
10.0	26.0
10.0	34.0
11.0	17.0
11.0	28.0
12.0	14.0
12.0	20.0
12.0	24.0
12.0	28.0
13.0	26.0
13.0	34.0
13.0	34.0
13.0	46.0
14.0	26.0
14.0	36.0
14.0	60.0
14.0	80.0
15.0	20.0
15.0	26.0
15.0	54.0
16.0	32.0
16.0	40.0
17.0	32.0
17.0	40.0



# Reading data

*read a built-in sample dataset*

```
(use 'incanter core io)
```

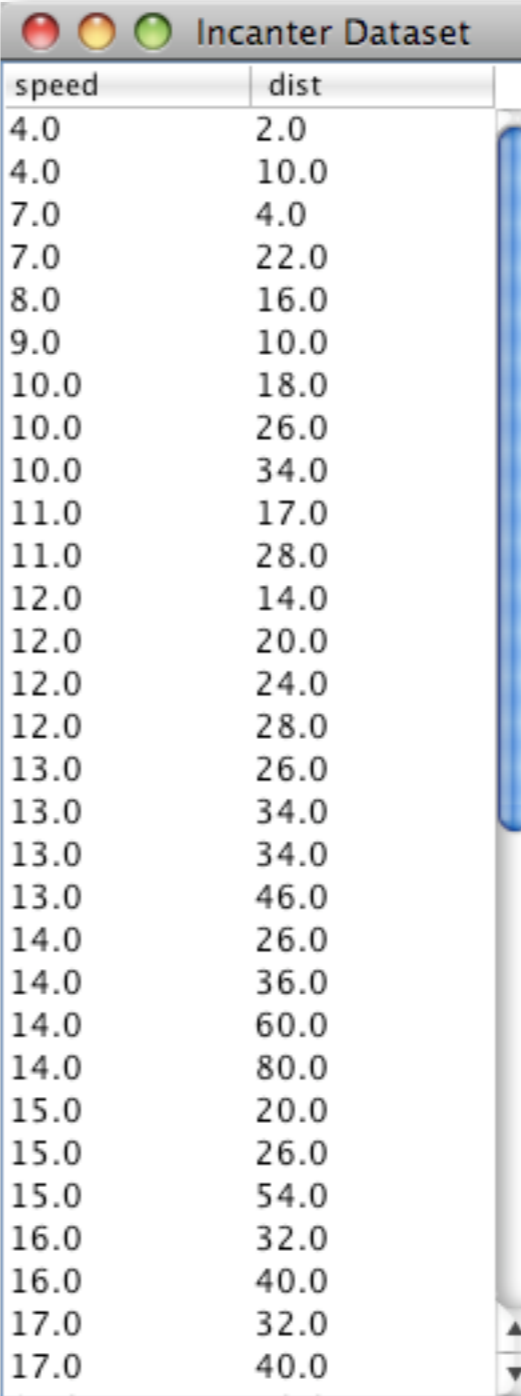
```
(read-dataset "./data/cars.csv"  
             :header true)
```

```
(read-dataset "./data/cars.tdd"  
             :header true  
             :delim \tab)
```

```
(read-dataset "http://bit.ly/aZyjKa"  
             :header true)
```

```
(use 'incanter.datasets)  
(get-dataset :cars)
```

```
(use 'incanter.mongodb)  
(use 'somnium.congomongo)  
(mongo! :db "mydb")  
(view (fetch-dataset :cars))
```



The screenshot shows a window titled "Incanter Dataset" displaying a table with two columns: "speed" and "dist". The table contains 20 rows of data. The "speed" column values range from 4.0 to 17.0, and the "dist" column values range from 2.0 to 80.0. The window has a standard macOS-style title bar with red, yellow, and green buttons.

speed	dist
4.0	2.0
4.0	10.0
7.0	4.0
7.0	22.0
8.0	16.0
9.0	10.0
10.0	18.0
10.0	26.0
10.0	34.0
11.0	17.0
11.0	28.0
12.0	14.0
12.0	20.0
12.0	24.0
12.0	28.0
13.0	26.0
13.0	34.0
13.0	34.0
13.0	46.0
14.0	26.0
14.0	36.0
14.0	60.0
14.0	80.0
15.0	20.0
15.0	26.0
15.0	54.0
16.0	32.0
16.0	40.0
17.0	32.0
17.0	40.0



# Reading data

*retrieve a dataset from MongoDB*

```
(use 'incanter core io)

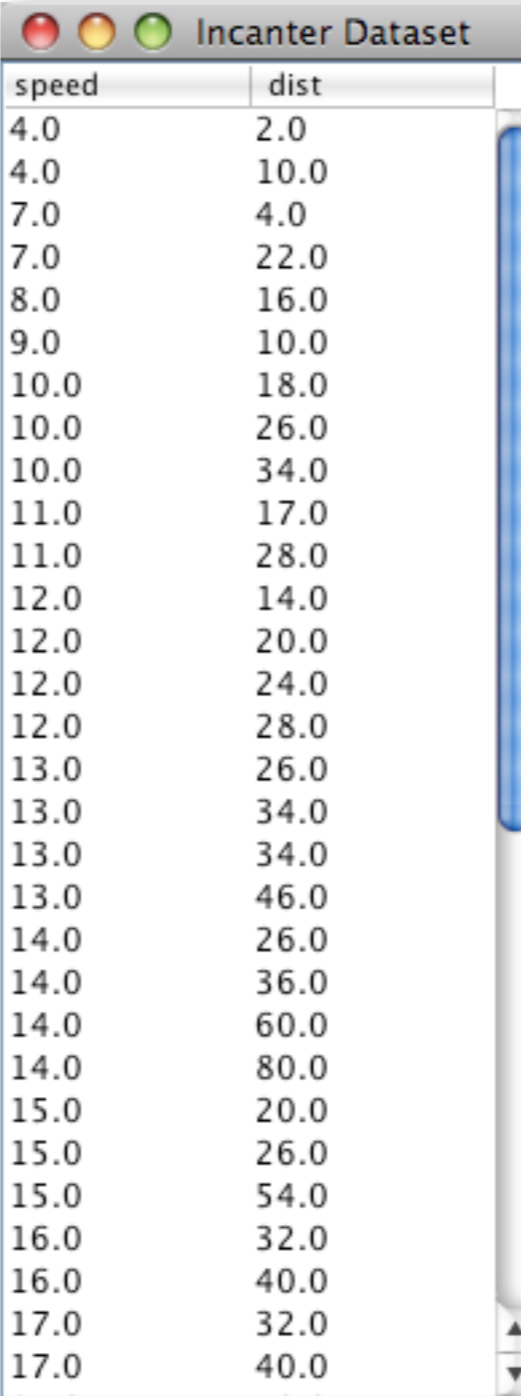
(read-dataset "./data/cars.csv"
  :header true)

(read-dataset "./data/cars.tdd"
  :header true
  :delim \tab)

(read-dataset "http://bit.ly/aZyjKa"
  :header true)

(use 'incanter.datasets)
(get-dataset :cars)

(use 'incanter.mongodb)
(use 'somnium.congomongo)
(mongo! :db "mydb")
(view (fetch-dataset :cars))
```



The screenshot shows a window titled "Incanter Dataset" containing a table with two columns: "speed" and "dist". The table lists 20 rows of data, with values for speed and distance. A vertical scrollbar is visible on the right side of the table.

speed	dist
4.0	2.0
4.0	10.0
7.0	4.0
7.0	22.0
8.0	16.0
9.0	10.0
10.0	18.0
10.0	26.0
10.0	34.0
11.0	17.0
11.0	28.0
12.0	14.0
12.0	20.0
12.0	24.0
12.0	28.0
13.0	26.0
13.0	34.0
13.0	34.0
13.0	46.0
14.0	26.0
14.0	36.0
14.0	60.0
14.0	80.0
15.0	20.0
15.0	26.0
15.0	54.0
16.0	32.0
16.0	40.0
17.0	32.0
17.0	40.0



```
(use '(incanter core io))  
(save data "./data.csv")
```

```
(use '(incanter core mongodb))  
(use 'somnium.congomongo)  
(mongo! :db "mydb")  
(insert-dataset :cars data)
```



# Saving data

---

*save a dataset in a MongoDB database*

```
(use '(incanter core io))  
(save data "./data.csv")
```

```
(use '(incanter core mongodb))  
(use 'somnium.congomongo)  
(mongo! :db "mydb")  
(insert-dataset :cars data)
```



# Column selection

*select a single column*

```
(use ' (incanter core datasets))
```

```
($ :speed (get-dataset :cars))
```

```
(4 4 7 7 8 9 10 10 10 11 11 12 12 12 12 13 13 13 13 14 14 14 14 15 15  
15 16 16 17 17 17 18 18 18 18 19 19 19 20 20 20 20 20 22 23 24 24 24 24  
25)
```

```
(with-data (get-dataset :cars)  
  [(mean ($ :speed))  
   (sd ($ :speed))])
```

```
(with-data (get-dataset :iris)  
  (view $data)  
  (view ($ [:Sepal.Length :Sepal.Width :Species]))  
  (view ($ [:not :Petal.Width :Petal.Length]))  
  (view ($ 0 [:not :Petal.Width :Petal.Length])))
```



# Column selection

*use with-data macro to bind dataset*

```
(use ' (incanter core datasets))
```

```
($ :speed (get-dataset :cars))
```

```
(4 4 7 7 8 9 10 10 10 11 11 12 12 12 12 13 13 13 13 14 14 14 14 15 15  
15 16 16 17 17 17 18 18 18 18 19 19 19 20 20 20 20 20 22 23 24 24 24 24  
25)
```

```
(with-data (get-dataset :cars)  
  [(mean ($ :speed))  
   (sd ($ :speed))])
```

```
[15.4 5.29]
```

```
(with-data (get-dataset :iris)  
  (view $data)  
  (view ($ [:Sepal.Length :Sepal.Width :Species]))  
  (view ($ [:not :Petal.Width :Petal.Length]))  
  (view ($ 0 [:not :Petal.Width :Petal.Length])))
```



# Column selection

*view dataset bound to \$data*

```
(use '(incanter core datasets))
```

```
($ :speed (get-dataset :cars))
```

```
(4 4 7 7 8 9 10 10 10 11 11 12 12 12 12 13 13 13 13 14 14 14 14 15 15  
15 16 16 17 17 17 18 18 18 18 19 19 19 20 20 20 20 20 22 23 24 24 24 24  
25)
```

```
(with-data (get-dataset :cars)  
  [(mean ($ :speed))  
   (sd ($ :speed))])
```

```
[15.4 5.29]
```

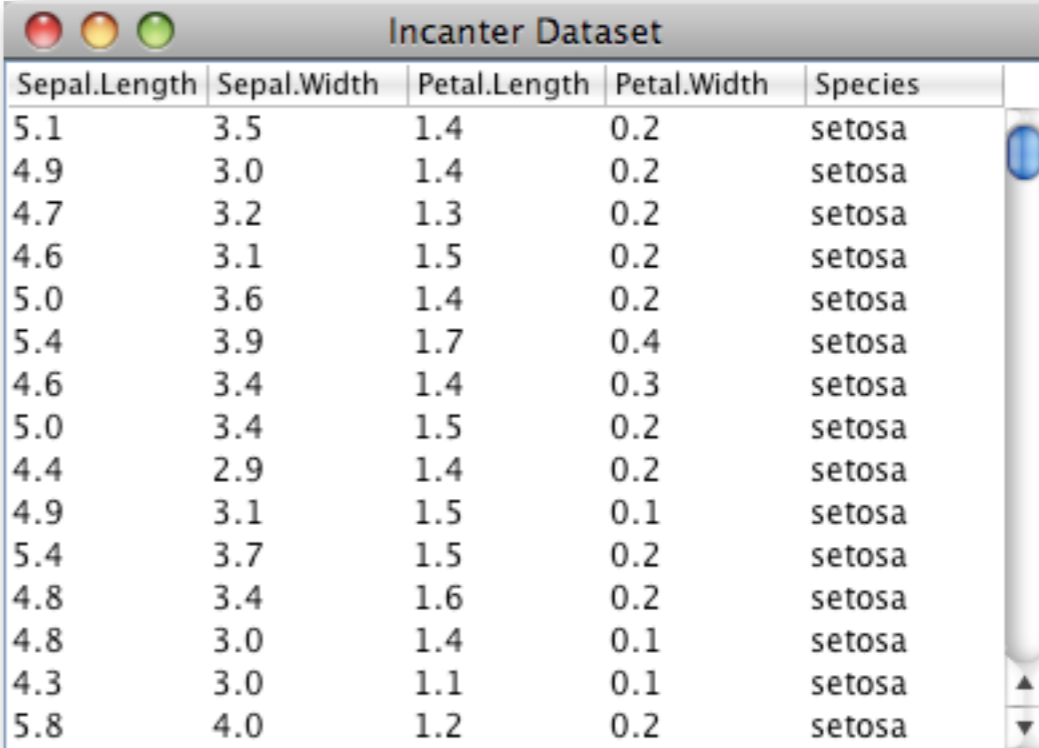
```
(with-data (get-dataset :iris)
```

```
(view $data)
```

```
(view ($ [:Sepal.Length :Sepal.Width :Species]))
```

```
(view ($ [:not :Petal.Width :Petal.Length]))
```

```
(view ($ 0 [:not :Petal.Width :Petal.Length]))
```



The screenshot shows a window titled "Incanter Dataset" with a table of data. The table has five columns: Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, and Species. The data is for the Iris setosa species, with 15 rows of values.

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3.0	1.4	0.1	setosa
4.3	3.0	1.1	0.1	setosa
5.8	4.0	1.2	0.2	setosa



# Column selection

*select multiple columns*

```
(use '(incanter core datasets))
```

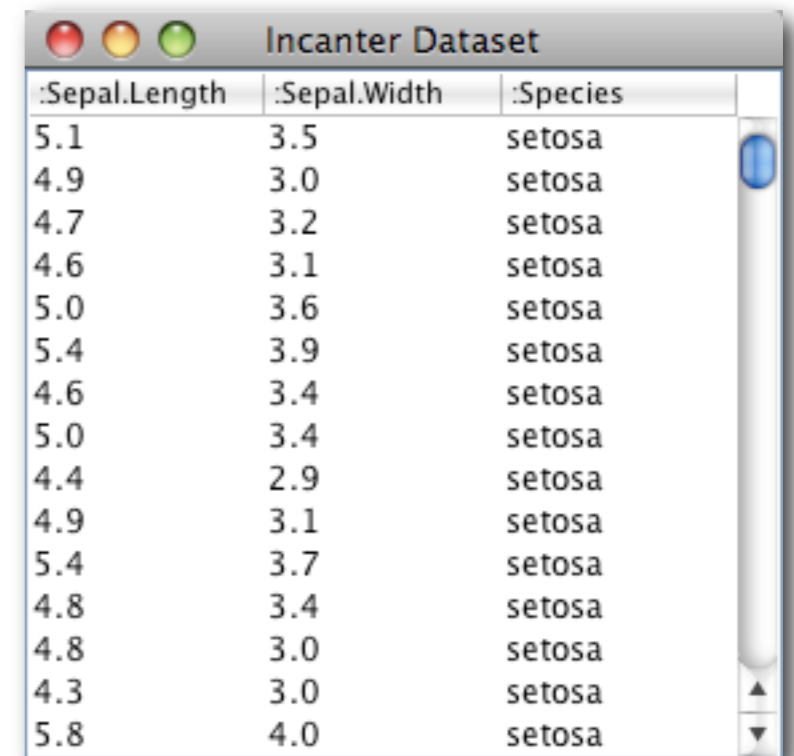
```
($ :speed (get-dataset :cars))
```

```
(4 4 7 7 8 9 10 10 10 11 11 12 12 12 12 13 13 13 13 14 14 14 14 15 15  
15 16 16 17 17 17 18 18 18 18 19 19 19 20 20 20 20 20 22 23 24 24 24 24  
25)
```

```
(with-data (get-dataset :cars)  
  [(mean ($ :speed))  
   (sd ($ :speed))])
```

```
[15.4 5.29]
```

```
(with-data (get-dataset :iris)  
  (view $data)  
  (view ($ [:Sepal.Length :Sepal.Width :Species]))  
  (view ($ [:not :Petal.Width :Petal.Length]))  
  (view ($ 0 [:not :Petal.Width :Petal.Length])))
```



The screenshot shows a window titled "Incanter Dataset" with a table of data. The table has three columns: ":Sepal.Length", ":Sepal.Width", and ":Species". The data rows are as follows:

:Sepal.Length	:Sepal.Width	:Species
5.1	3.5	setosa
4.9	3.0	setosa
4.7	3.2	setosa
4.6	3.1	setosa
5.0	3.6	setosa
5.4	3.9	setosa
4.6	3.4	setosa
5.0	3.4	setosa
4.4	2.9	setosa
4.9	3.1	setosa
5.4	3.7	setosa
4.8	3.4	setosa
4.8	3.0	setosa
4.3	3.0	setosa
5.8	4.0	setosa



# Column selection

*exclude multiple columns*

```
(use '(incanter core datasets))
```

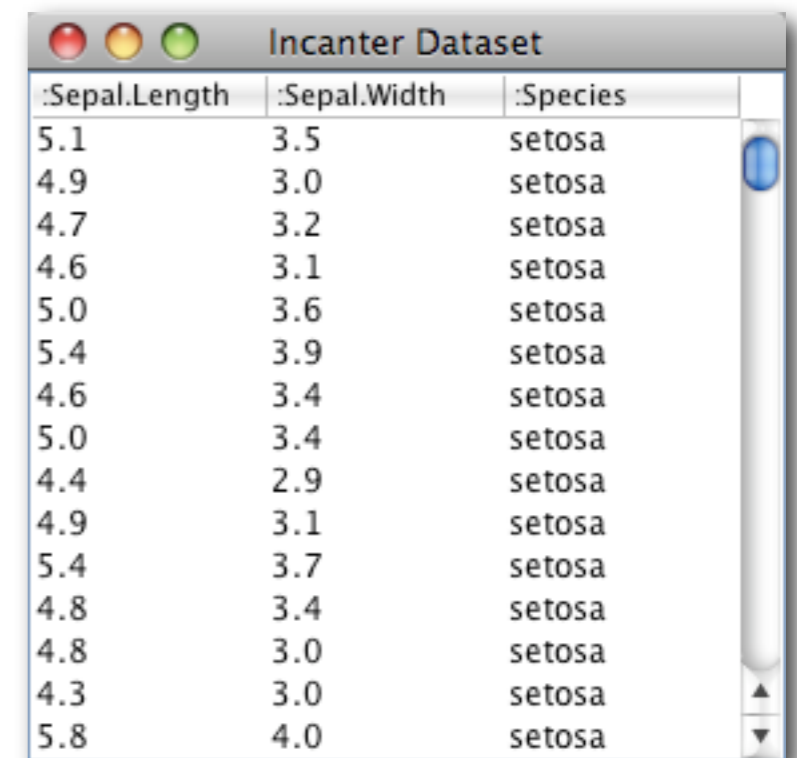
```
($ :speed (get-dataset :cars))
```

```
(4 4 7 7 8 9 10 10 10 11 11 12 12 12 12 13 13 13 13 14 14 14 14 15 15  
15 16 16 17 17 17 18 18 18 18 19 19 19 20 20 20 20 20 22 23 24 24 24 24  
25)
```

```
(with-data (get-dataset :cars)  
  [(mean ($ :speed))  
   (sd ($ :speed))])
```

```
[15.4 5.29]
```

```
(with-data (get-dataset :iris)  
  (view $data)  
  (view ($ [:Sepal.Length :Sepal.Width :Species]))  
  (view ($ [:not :Petal.Width :Petal.Length]))  
  (view ($ 0 [:not :Petal.Width :Petal.Length])))
```



The screenshot shows a window titled "Incanter Dataset" with a table of data. The table has three columns: ":Sepal.Length", ":Sepal.Width", and ":Species". The data rows are as follows:

:Sepal.Length	:Sepal.Width	:Species
5.1	3.5	setosa
4.9	3.0	setosa
4.7	3.2	setosa
4.6	3.1	setosa
5.0	3.6	setosa
5.4	3.9	setosa
4.6	3.4	setosa
5.0	3.4	setosa
4.4	2.9	setosa
4.9	3.1	setosa
5.4	3.7	setosa
4.8	3.4	setosa
4.8	3.0	setosa
4.3	3.0	setosa
5.8	4.0	setosa



# Column selection

*select only the first row*

```
(use '(incanter core datasets))
```

```
($ :speed (get-dataset :cars))
```

```
(4 4 7 7 8 9 10 10 10 11 11 12 12 12 12 13 13 13 13 14 14 14 14 15 15  
15 16 16 17 17 17 18 18 18 18 19 19 19 20 20 20 20 20 22 23 24 24 24 24  
25)
```

```
(with-data (get-dataset :cars)  
  [(mean ($ :speed))  
   (sd ($ :speed))])
```

```
[15.4 5.29]
```

```
(with-data (get-dataset :iris)  
  (view $data)  
  (view ($ [:Sepal.Length :Sepal.Width :Species]))  
  (view ($ [:not :Petal.Width :Petal.Length]))  
  (view ($ 0 [:not :Petal.Width :Petal.Length])))
```

```
(5.1 3.5 setosa)
```



# Row selection

*select rows where species equals 'setosa'*

```
(use ' (incanter core datasets))
```

```
($where {"Species" "setosa"}  
  (get-dataset :iris))
```

```
($where {"Petal.Width" {:lt 1.5}}  
  (get-dataset :iris))
```

```
($where {"Petal.Width" {:gt 1.0, :lt 1.5}}  
  (get-dataset :iris))
```

```
($where {"Petal.Width" {:gt 1.0, :lt 1.5}  
  "Species" {:in #{"virginica" "setosa"}}}  
  (get-dataset :iris))
```

```
($where (fn [row]  
  (or (< (row "Petal.Width") 1.0)  
      (> (row "Petal.Length") 5.0)))  
  (get-dataset :iris))
```



# Row selection

*select rows where petal-width < 1.5*

```
(use ' (incanter core datasets))
```

```
($where {"Species" "setosa"}  
  (get-dataset :iris))
```

```
($where {"Petal.Width" {:lt 1.5}}  
  (get-dataset :iris))
```

```
($where {"Petal.Width" {:gt 1.0, :lt 1.5}}  
  (get-dataset :iris))
```

```
($where {"Petal.Width" {:gt 1.0, :lt 1.5}  
  "Species" {:in #{"virginica" "setosa"}}}  
  (get-dataset :iris))
```

```
($where (fn [row]  
  (or (< (row "Petal.Width") 1.0)  
      (> (row "Petal.Length") 5.0)))  
  (get-dataset :iris))
```



# Row selection

*select rows where  $1.0 < \text{petal-width} < 1.5$*

```
(use ' (incanter core datasets))
```

```
($where {"Species" "setosa"}  
  (get-dataset :iris))
```

```
($where {"Petal.Width" {< 1.5}}  
  (get-dataset :iris))
```

```
($where {"Petal.Width" {> 1.0, < 1.5}}  
  (get-dataset :iris))
```

```
($where {"Petal.Width" {> 1.0, < 1.5}  
  "Species" {in #{"virginica" "setosa"}}}  
  (get-dataset :iris))
```

```
($where (fn [row]  
  (or (< (row "Petal.Width") 1.0)  
      (> (row "Petal.Length") 5.0)))  
  (get-dataset :iris))
```



# Row selection

*select rows where species is 'virginica' or 'setosa'*

```
(use ' (incanter core datasets))
```

```
($where {"Species" "setosa"}  
  (get-dataset :iris))
```

```
($where {"Petal.Width" {:lt 1.5}}  
  (get-dataset :iris))
```

```
($where {"Petal.Width" {:gt 1.0, :lt 1.5}}  
  (get-dataset :iris))
```

```
($where {"Petal.Width" {:gt 1.0, :lt 1.5}  
  "Species" {:in #{"virginica" "setosa"}}}  
  (get-dataset :iris))
```

```
($where (fn [row]  
  (or (< (row "Petal.Width") 1.0)  
      (> (row "Petal.Length") 5.0)))  
  (get-dataset :iris))
```



# Row selection

*select rows using an arbitrary predicate function*

---

```
(use ' (incanter core datasets))
```

```
($where {"Species" "setosa"}  
  (get-dataset :iris))
```

```
($where {"Petal.Width" {:lt 1.5}}  
  (get-dataset :iris))
```

```
($where {"Petal.Width" {:gt 1.0, :lt 1.5}}  
  (get-dataset :iris))
```

```
($where {"Petal.Width" {:gt 1.0, :lt 1.5}  
  "Species" {:in #{"virginica" "setosa"}}}  
  (get-dataset :iris))
```

```
($where (fn [row]  
  (or (< (row "Petal.Width") 1.0)  
      (> (row "Petal.Length") 5.0)))  
  (get-dataset :iris))
```



# Sorting data

*hair & eye color data*

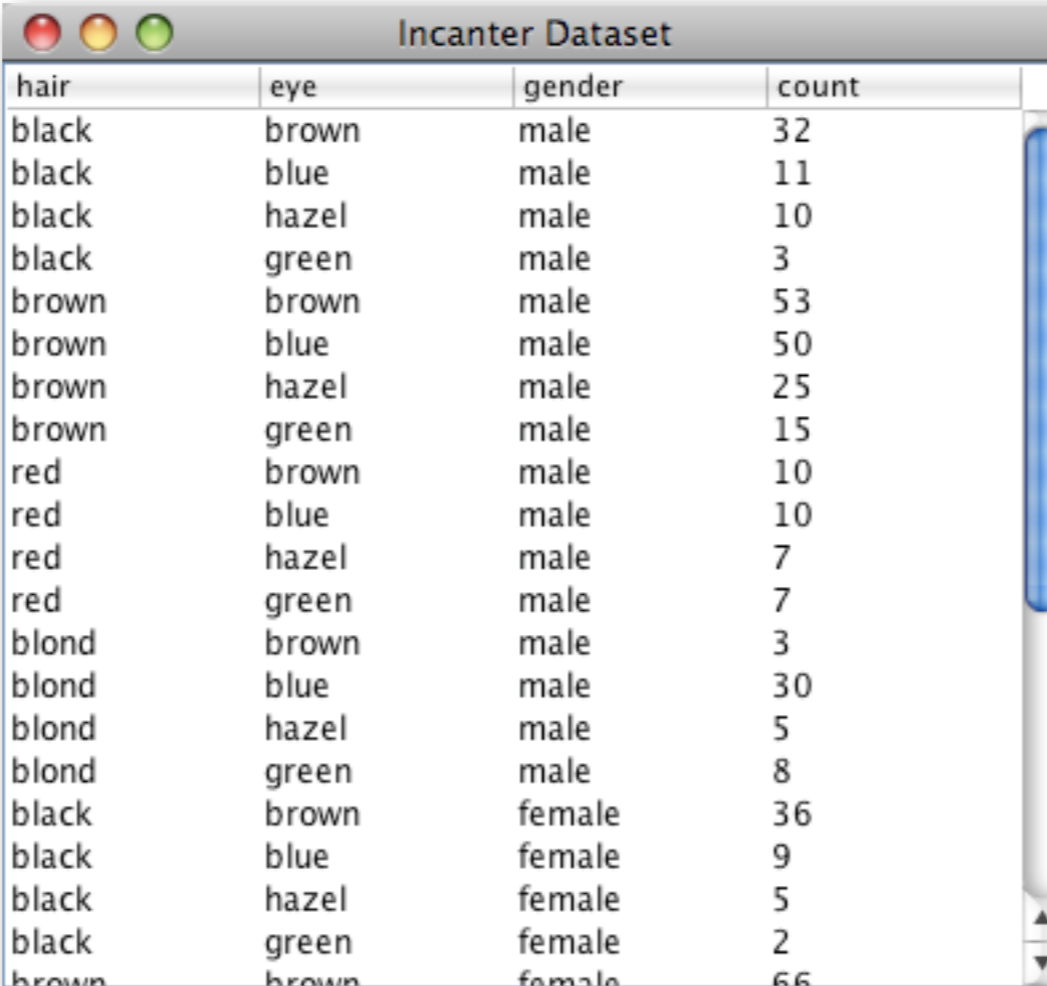
```
(use ' (incanter core datasets))
```

```
(with-data (get-dataset :hair-eye-color)
```

```
  (view $data)
```

```
  (view ($order :count :desc))
```

```
  (view ($order [:hair :eye] :desc)))
```



The screenshot shows a window titled "Incanter Dataset" displaying a table with four columns: hair, eye, gender, and count. The data is sorted by count in descending order. The table contains 20 rows of data.

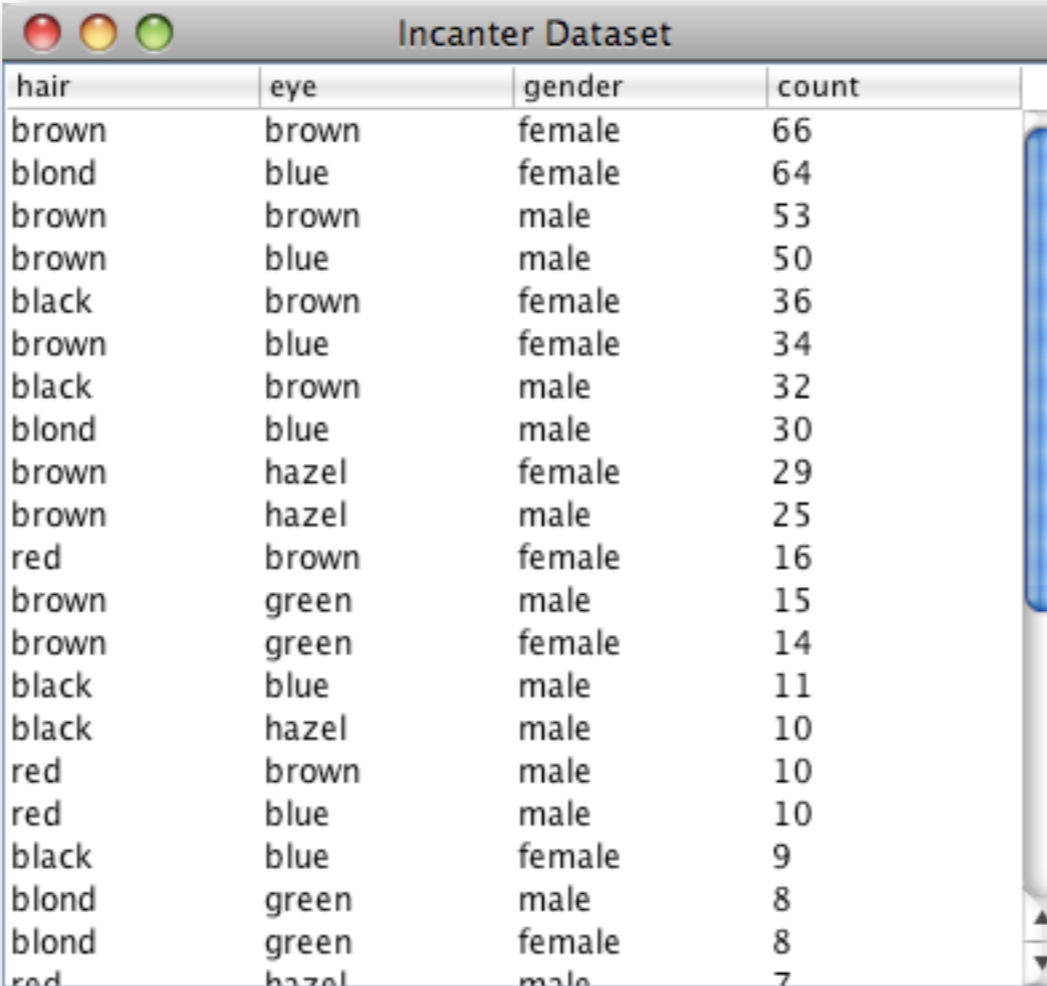
hair	eye	gender	count
black	brown	male	32
black	blue	male	11
black	hazel	male	10
black	green	male	3
brown	brown	male	53
brown	blue	male	50
brown	hazel	male	25
brown	green	male	15
red	brown	male	10
red	blue	male	10
red	hazel	male	7
red	green	male	7
blond	brown	male	3
blond	blue	male	30
blond	hazel	male	5
blond	green	male	8
black	brown	female	36
black	blue	female	9
black	hazel	female	5
black	green	female	2
brown	brown	female	66



# Sorting data

*sort by count in descending order*

```
(use ' (incanter core datasets))  
  
(with-data (get-dataset :hair-eye-color)  
  (view $data)  
  (view ($order :count :desc) )  
  (view ($order [:hair :eye] :desc)))
```



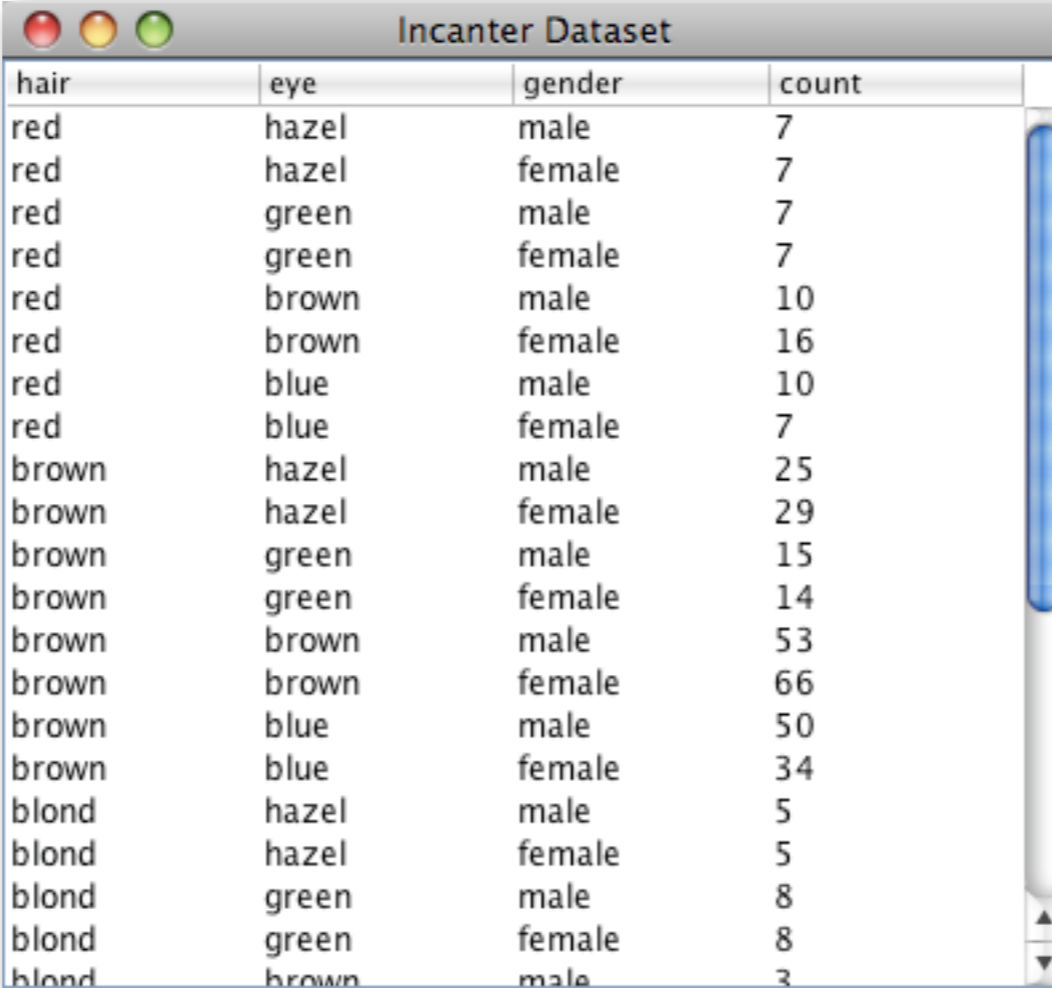
hair	eye	gender	count
brown	brown	female	66
blond	blue	female	64
brown	brown	male	53
brown	blue	male	50
black	brown	female	36
brown	blue	female	34
black	brown	male	32
blond	blue	male	30
brown	hazel	female	29
brown	hazel	male	25
red	brown	female	16
brown	green	male	15
brown	green	female	14
black	blue	male	11
black	hazel	male	10
red	brown	male	10
red	blue	male	10
black	blue	female	9
blond	green	male	8
blond	green	female	8
red	hazel	male	7



# Sorting data

*sort by hair and then eye color*

```
(use '(incanter core datasets))  
  
(with-data (get-dataset :hair-eye-color)  
  (view $data)  
  (view ($order :count :desc))  
  (view ($order [:hair :eye] :desc)))
```



The screenshot shows a window titled "Incanter Dataset" with a table of data. The table has four columns: hair, eye, gender, and count. The data is sorted by hair color and then by eye color. The counts are listed in descending order for each hair color.

hair	eye	gender	count
red	hazel	male	7
red	hazel	female	7
red	green	male	7
red	green	female	7
red	brown	male	10
red	brown	female	16
red	blue	male	10
red	blue	female	7
brown	hazel	male	25
brown	hazel	female	29
brown	green	male	15
brown	green	female	14
brown	brown	male	53
brown	brown	female	66
brown	blue	male	50
brown	blue	female	34
blond	hazel	male	5
blond	hazel	female	5
blond	green	male	8
blond	green	female	8
blond	brown	male	3



# Rolling up data

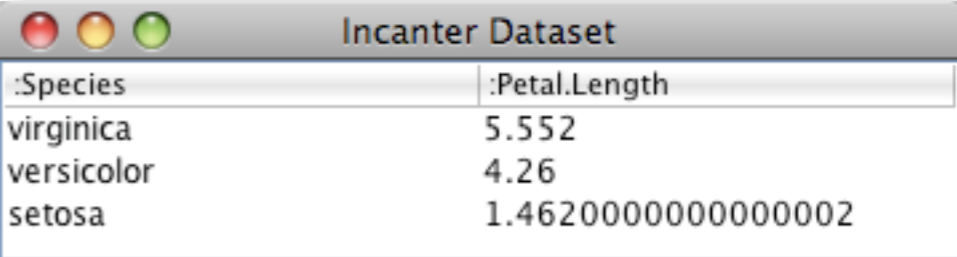
*mean petal-length by species*

```
(use ' (incanter core datasets stats))

(->> (get-dataset :iris)
      ($rollup mean :Petal.Length :Species)
      view)

(->> (get-dataset :iris)
      ($rollup #(/ (sd %) (count %))
               :Petal.Length :Species)
      view)

(->> (get-dataset :hair-eye-color)
      ($rollup sum :count [:hair :eye])
      ($order :count :desc)
      view)
```



:Species	:Petal.Length
virginica	5.552
versicolor	4.26
setosa	1.4620000000000002



# Rolling up data

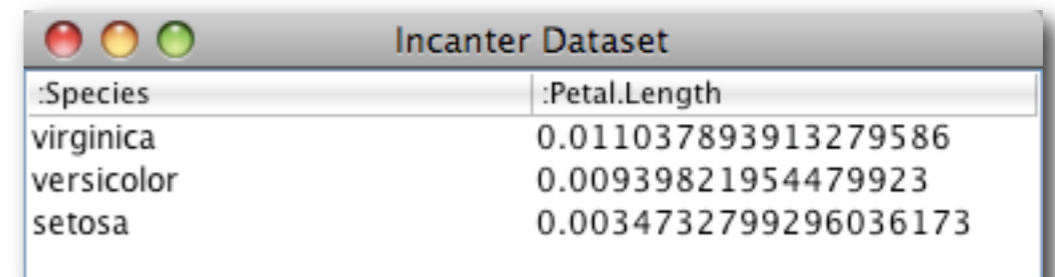
*standard error of petal-length*

```
(use ' (incanter core datasets stats))

(->> (get-dataset :iris)
      ($rollup mean :Petal.Length :Species)
      view)

(->> (get-dataset :iris)
      ($rollup #(/ (sd %) (count %))
               :Petal.Length :Species)
      view)

(->> (get-dataset :hair-eye-color)
      ($rollup sum :count [:hair :eye])
      ($order :count :desc)
      view)
```



:Species	:Petal.Length
virginica	0.011037893913279586
versicolor	0.00939821954479923
setosa	0.0034732799296036173



# Rolling up data

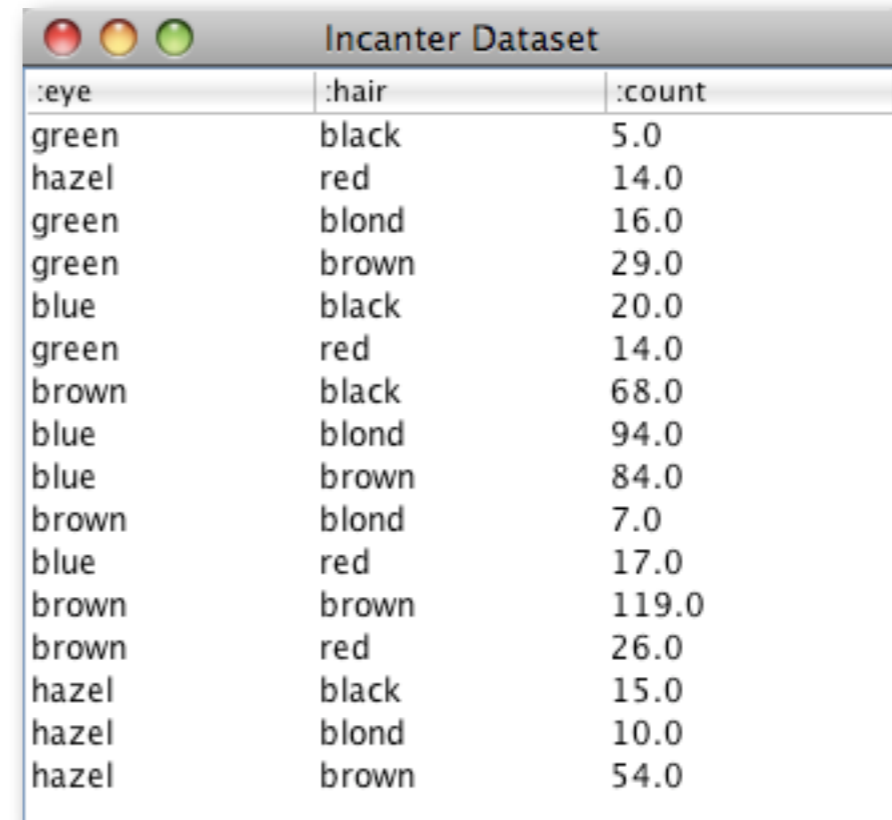
*sum of people with each hair/eye color combination*

```
(use ' (incanter core datasets stats))
```

```
(->> (get-dataset :iris)  
     ($rollup mean :Petal.Length :Species)  
     view)
```

```
(->> (get-dataset :iris)  
     ($rollup #(/ (sd %) (count %))  
              :Petal.Length :Species)  
     view)
```

```
(->> (get-dataset :hair-eye-color)  
     ($rollup sum :count [:hair :eye])  
     ($order :count :desc)  
     view)
```



:eye	:hair	:count
green	black	5.0
hazel	red	14.0
green	blond	16.0
green	brown	29.0
blue	black	20.0
green	red	14.0
brown	black	68.0
blue	blond	94.0
blue	brown	84.0
brown	blond	7.0
blue	red	17.0
brown	brown	119.0
brown	red	26.0
hazel	black	15.0
hazel	blond	10.0
hazel	brown	54.0



# Rolling up data

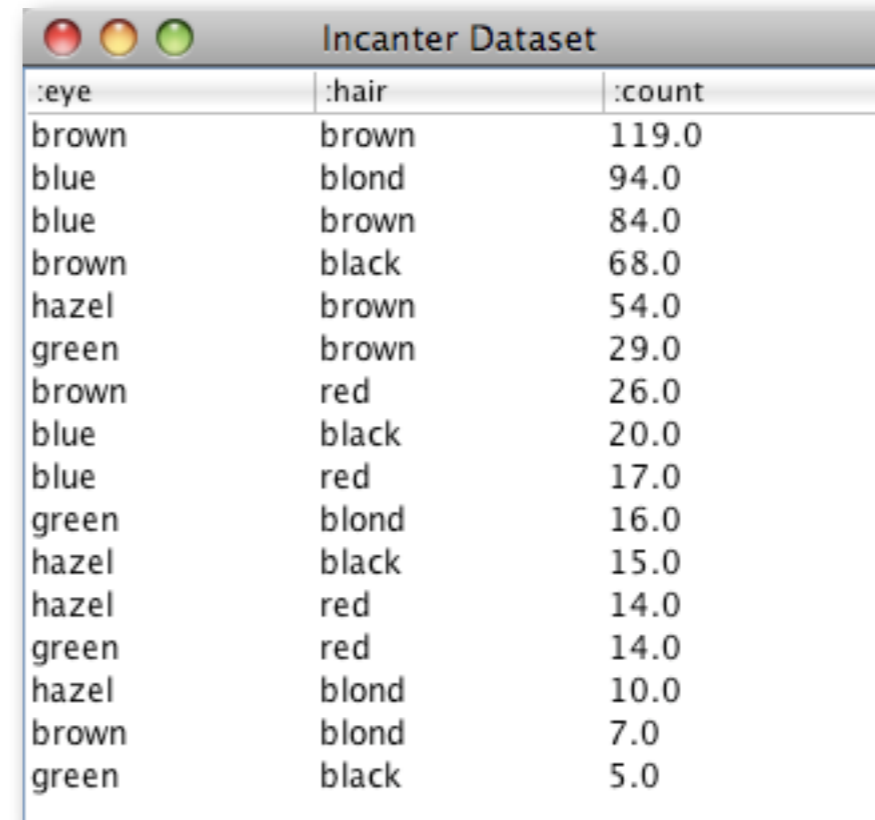
*sorted from most to least common*

```
(use ' (incanter core datasets stats))

(->> (get-dataset :iris)
      ($rollup mean :Petal.Length :Species)
      view)

(->> (get-dataset :iris)
      ($rollup #(/ (sd %) (count %))
               :Petal.Length :Species)
      view)

(->> (get-dataset :hair-eye-color)
      ($rollup sum :count [:hair :eye])
      ($order :count :desc)
      view)
```



:eye	:hair	:count
brown	brown	119.0
blue	blond	94.0
blue	brown	84.0
brown	black	68.0
hazel	brown	54.0
green	brown	29.0
brown	red	26.0
blue	black	20.0
blue	red	17.0
green	blond	16.0
hazel	black	15.0
hazel	red	14.0
green	red	14.0
hazel	blond	10.0
brown	blond	7.0
green	black	5.0



# Outline

---

## Overview

- What is Incanter?
- Getting started
- Incanter libraries

## Datasets

- Creating
- Reading data
- Saving data
- Column selection
- Row selection
- Sorting
- Rolling up

## Charts

- Scatter plots
- Chart options
- Saving charts
- Adding data
- Bar & line charts
- XY & function plots
- Histograms & box-plots
- Annotating charts

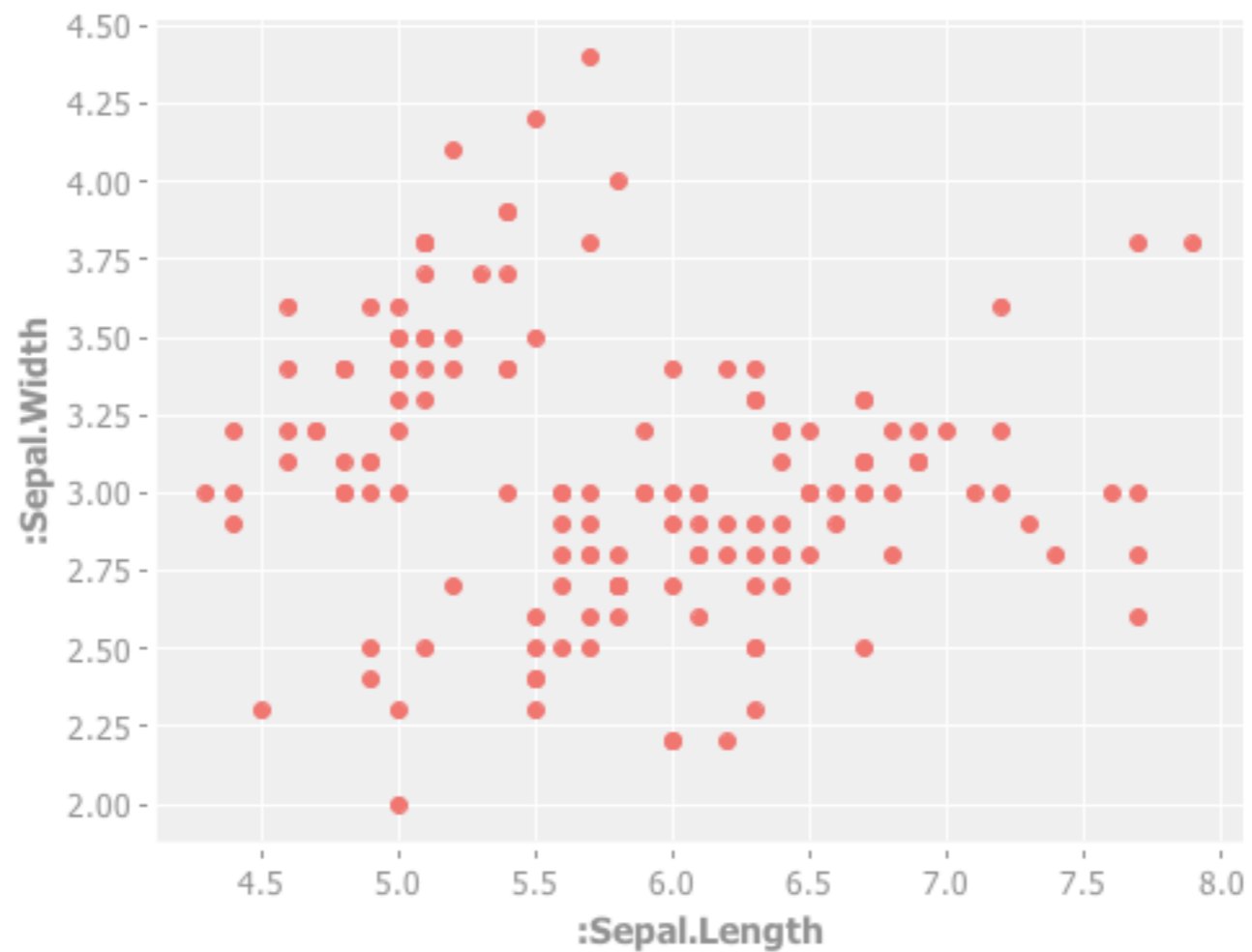
## Wrap up



# Scatter plots

*plot 'sepal length' vs 'sepal width'*

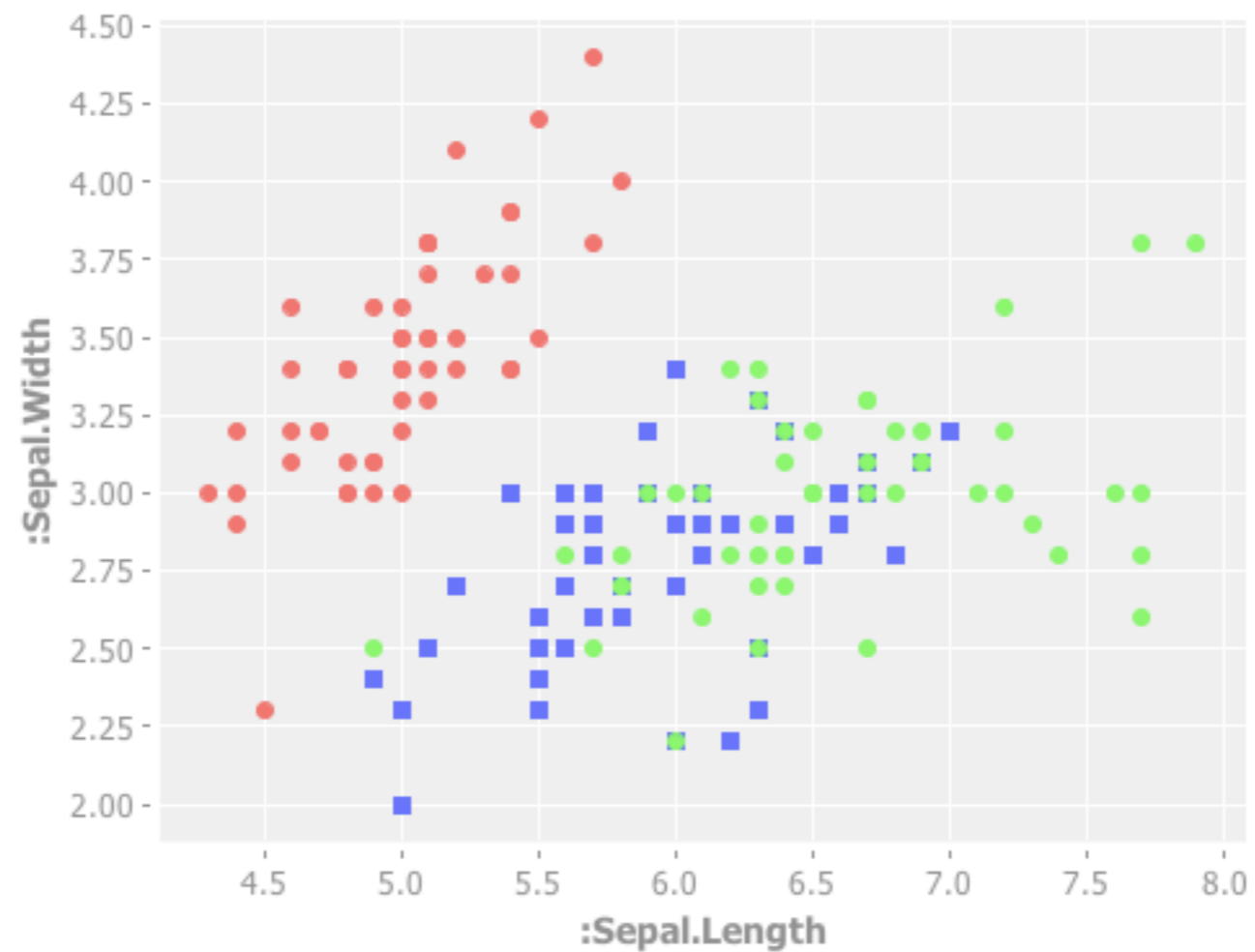
```
(view (scatter-plot :Sepal.Length :Sepal.Width  
                  :data (get-dataset :iris)))
```



# Chart options

*group by species values*

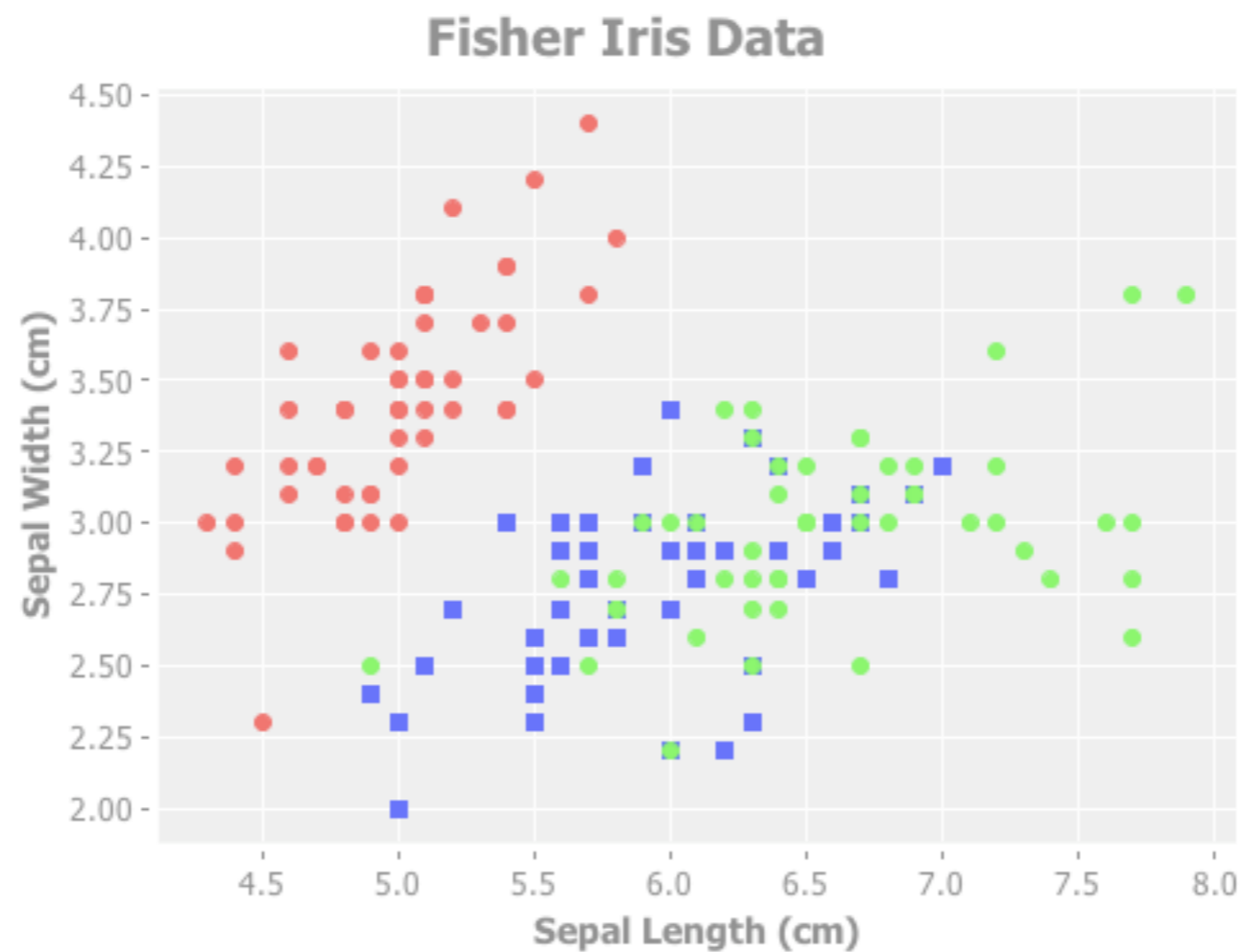
```
(view (scatter-plot :Sepal.Length :Sepal.Width  
:data (get-dataset :iris)  
:group-by :Species))
```



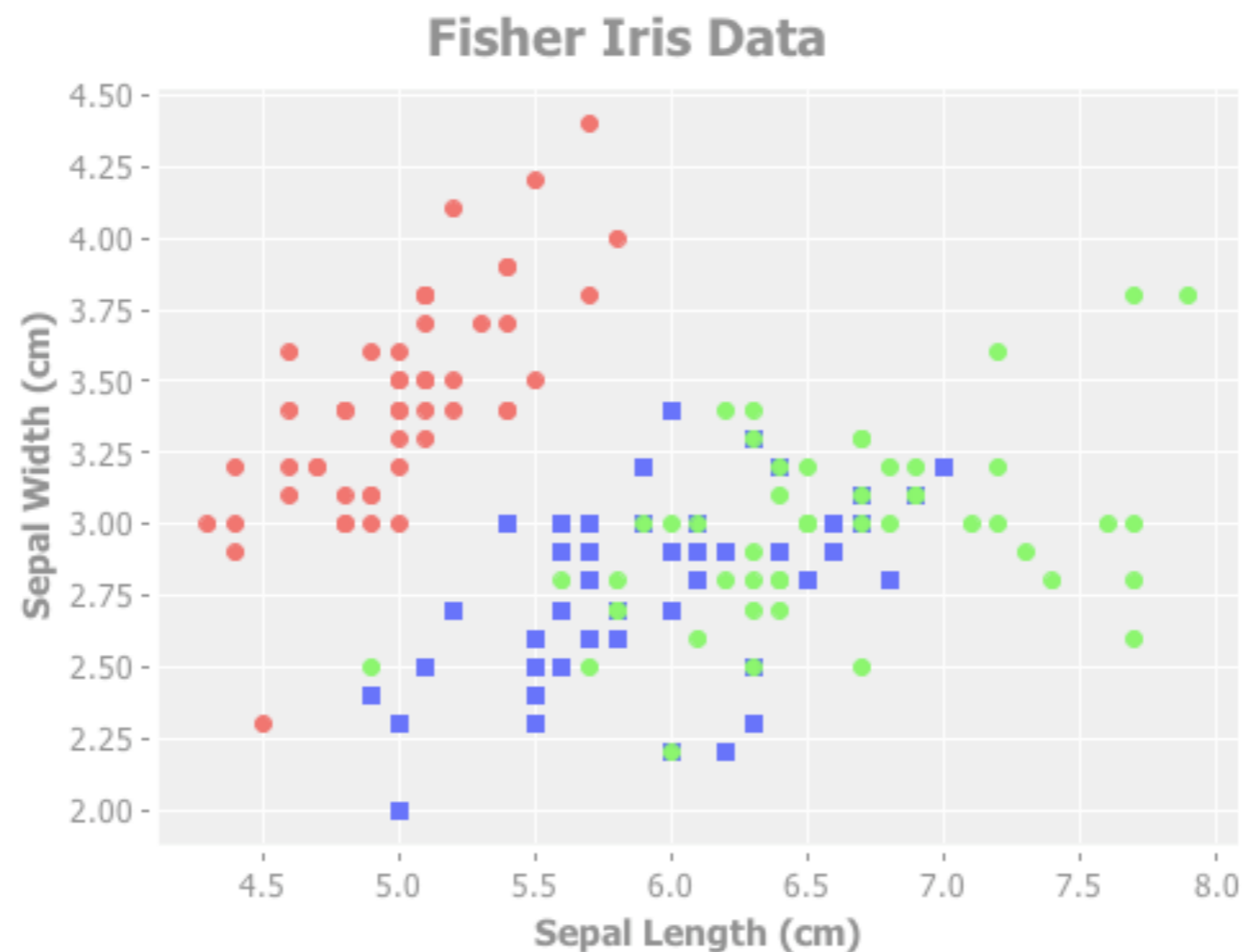
# Chart options

*change chart title & axes labels*

```
(view (scatter-plot :Sepal.Length :Sepal.Width  
:data (get-dataset :iris)  
:group-by :Species  
:title "Fisher Iris Data"  
:x-label "Sepal Length (cm)"  
:y-label "Sepal Width (cm)"))
```



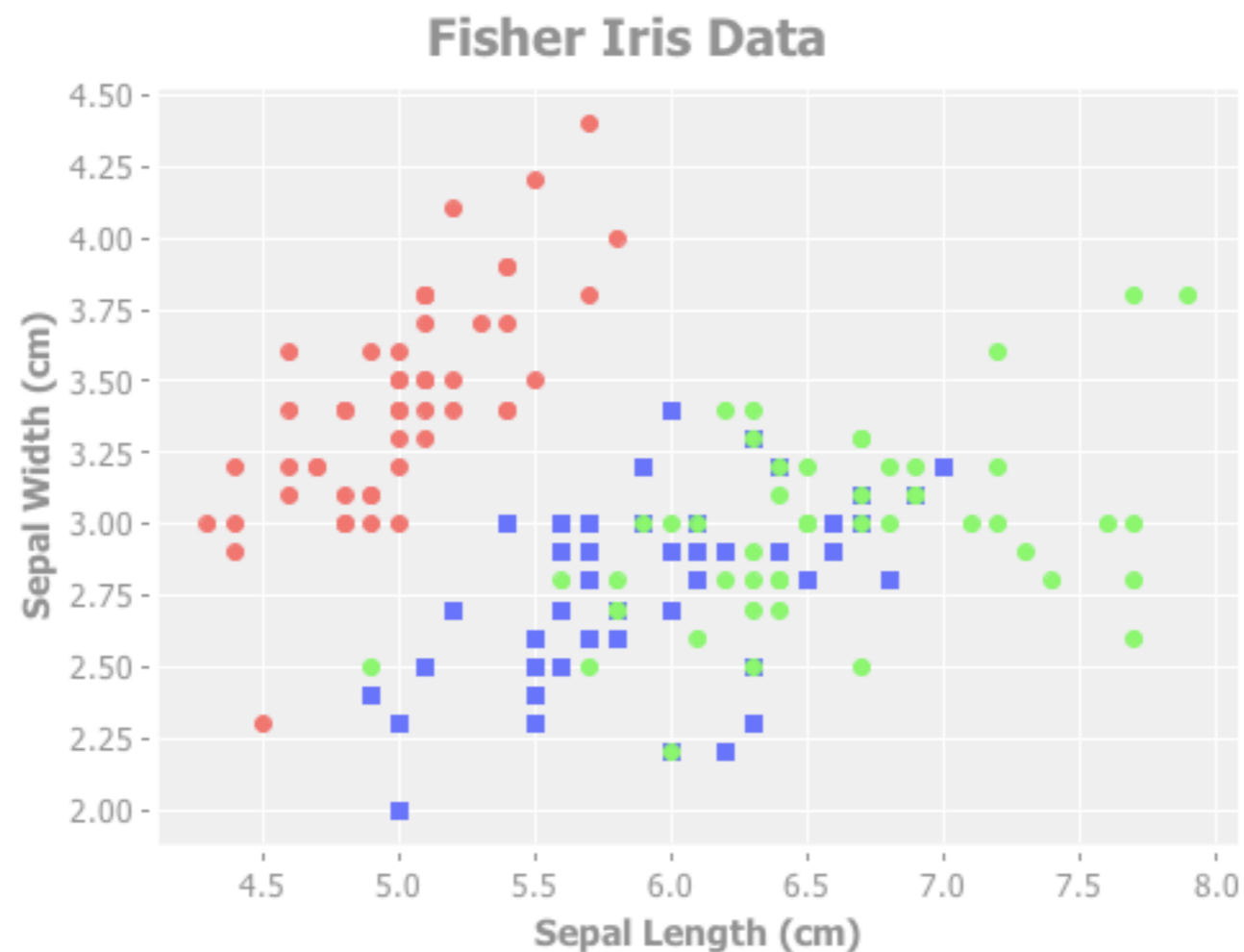
```
(save (scatter-plot :Sepal.Length :Sepal.Width
  :data (get-dataset :iris)
  :group-by :Species
  :title "Fisher Iris Data"
  :x-label "Sepal Length (cm)"
  :y-label "Sepal Width (cm)")
  "./iris-plot.png")
```



# Saving charts

*save chart to an OutputStream*

```
(def output-stream (java.io.ByteArrayOutputStream.))  
(save (scatter-plot :Sepal.Length :Sepal.Width  
  :data (get-dataset :iris)  
  :group-by :Species  
  :title "Fisher Iris Data"  
  :x-label "Sepal Length (cm)"  
  :y-label "Sepal Width (cm)")  
  output-stream)
```

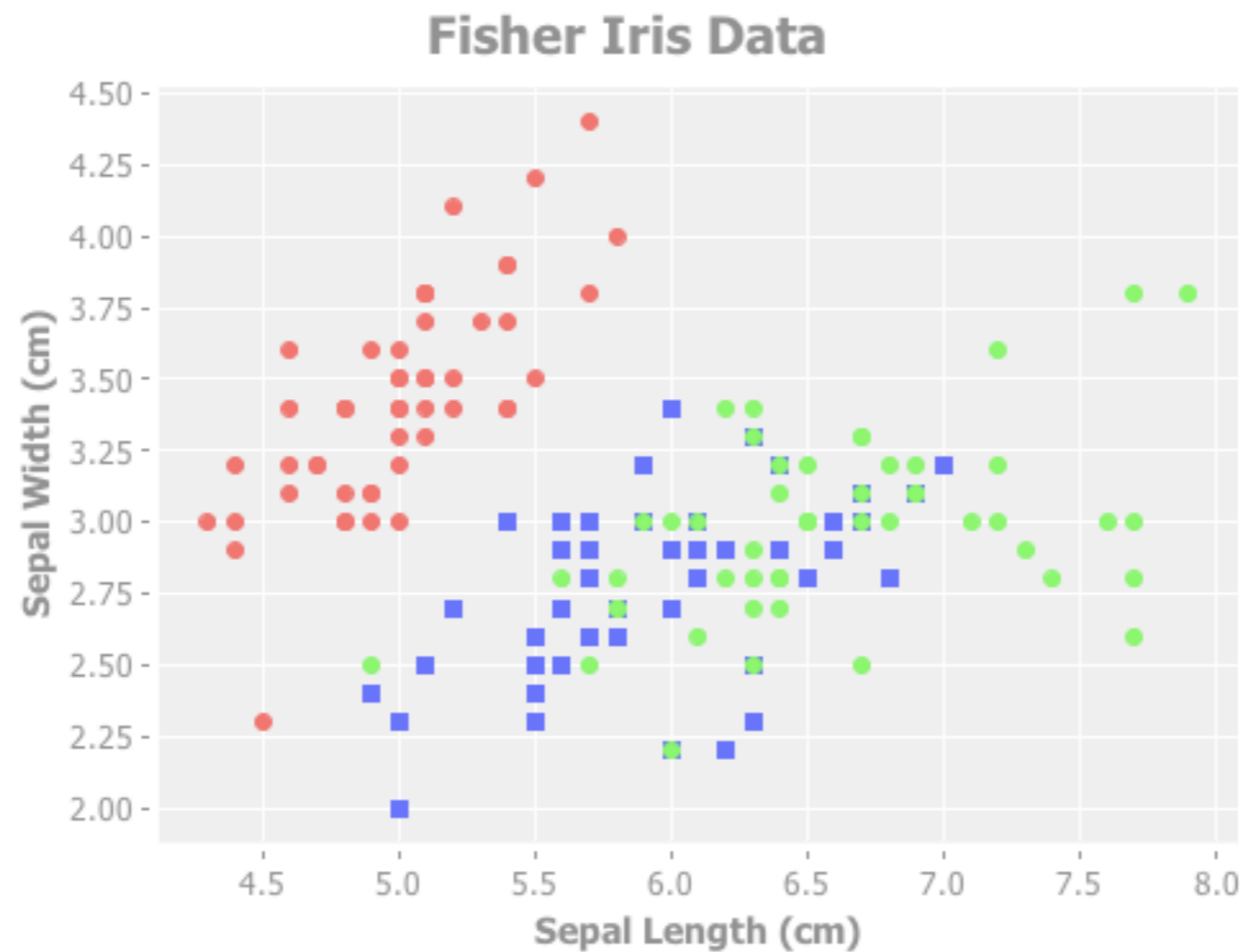


# Saving charts

*save chart to a PDF file*

(use 'incanter.pdf')

```
(save-pdf (scatter-plot :Sepal.Length :Sepal.Width
  :data (get-dataset :iris)
  :group-by :Species
  :title "Fisher Iris Data"
  :x-label "Sepal Length (cm)"
  :y-label "Sepal Width (cm)")
  "./iris-plot.pdf")
```



# Adding data

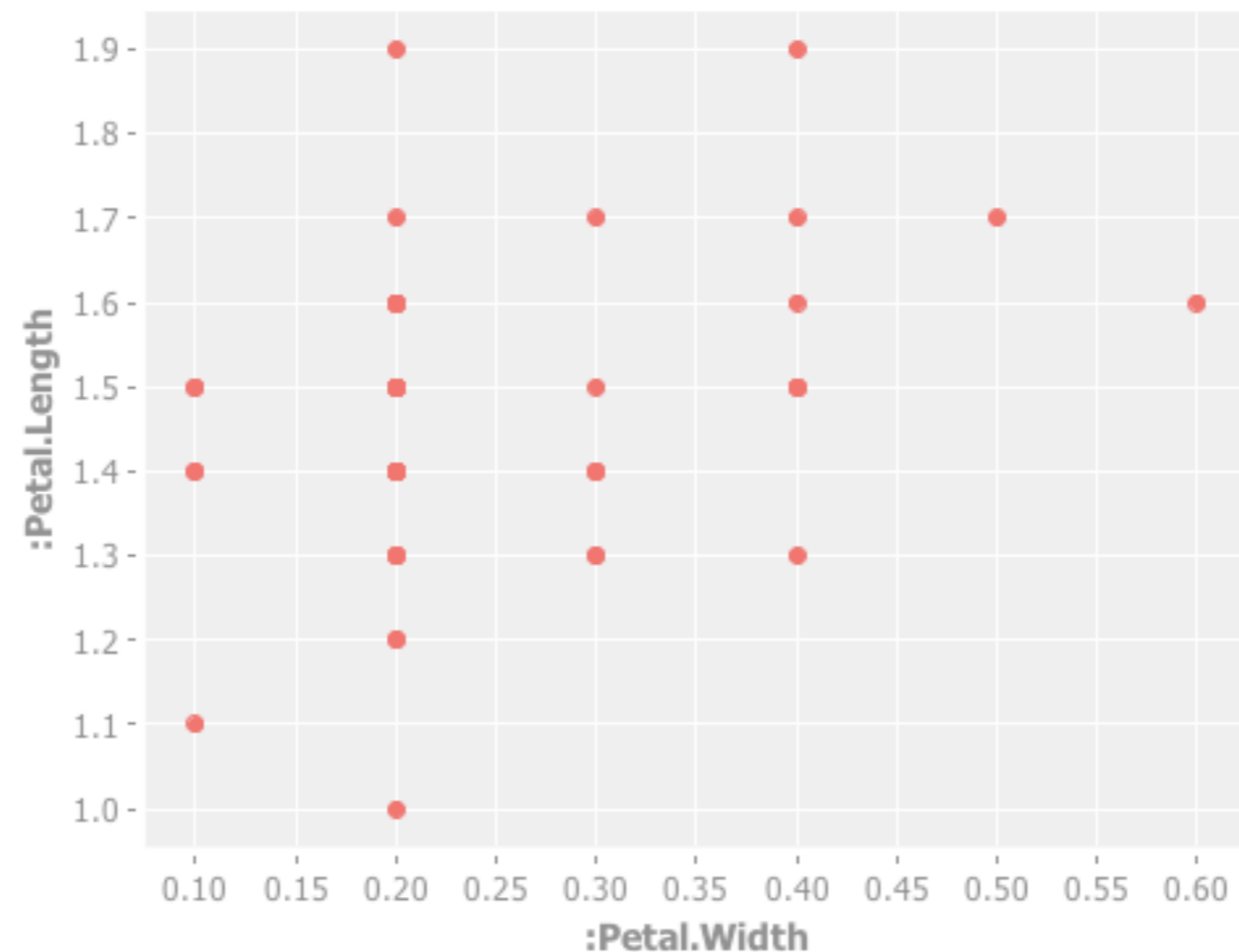
*plot points where petal-length  $\leq 2$  & petal-width  $< .75$*

```
(use '(incanter core charts datasets))
```

```
(with-data (get-dataset :iris)
```

```
(doto (scatter-plot :Petal.Width :Petal.Length  
                  :data ($where {"Petal.Length" {:lte 2.0}  
                                "Petal.Width"  {:lt 0.75}}))
```

```
view)))
```



# Adding data

*add points where petal-length > 2 & petal-width >= .75*

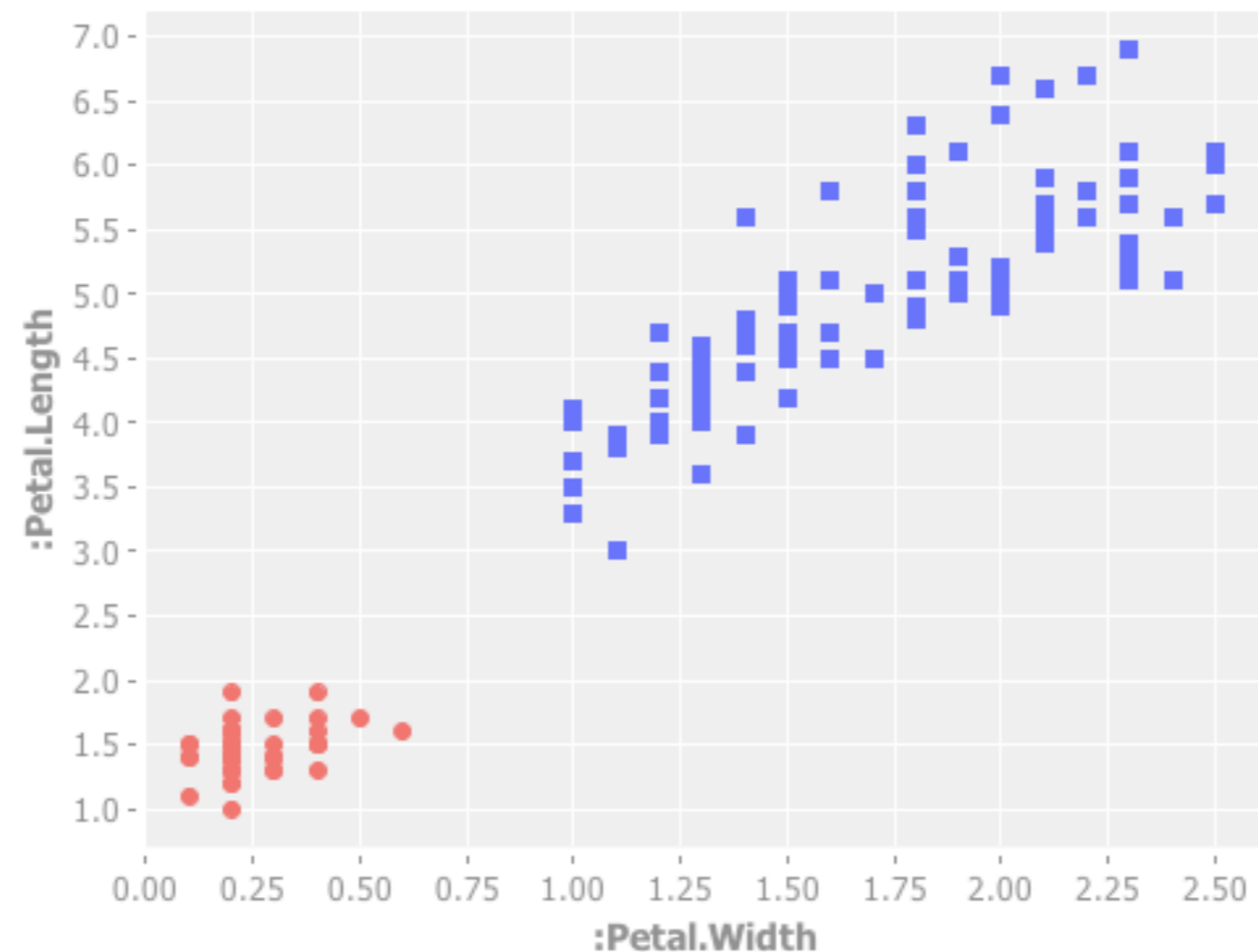
```
(use '(incanter core charts datasets))
```

```
(with-data (get-dataset :iris)
```

```
(doto (scatter-plot :Petal.Width :Petal.Length  
                  :data ($where {"Petal.Length" {:lte 2.0}  
                                "Petal.Width"  {:lt 0.75}})))
```

```
(add-points :Petal.Width :Petal.Length  
           :data ($where {"Petal.Length" {:gt 2.0}  
                           "Petal.Width" {:gte 0.75}}))
```

```
view)))
```



# Adding data

*add a regression line*

```
(use '(incanter core charts datasets stats))
```

```
(with-data (get-dataset :iris)
```

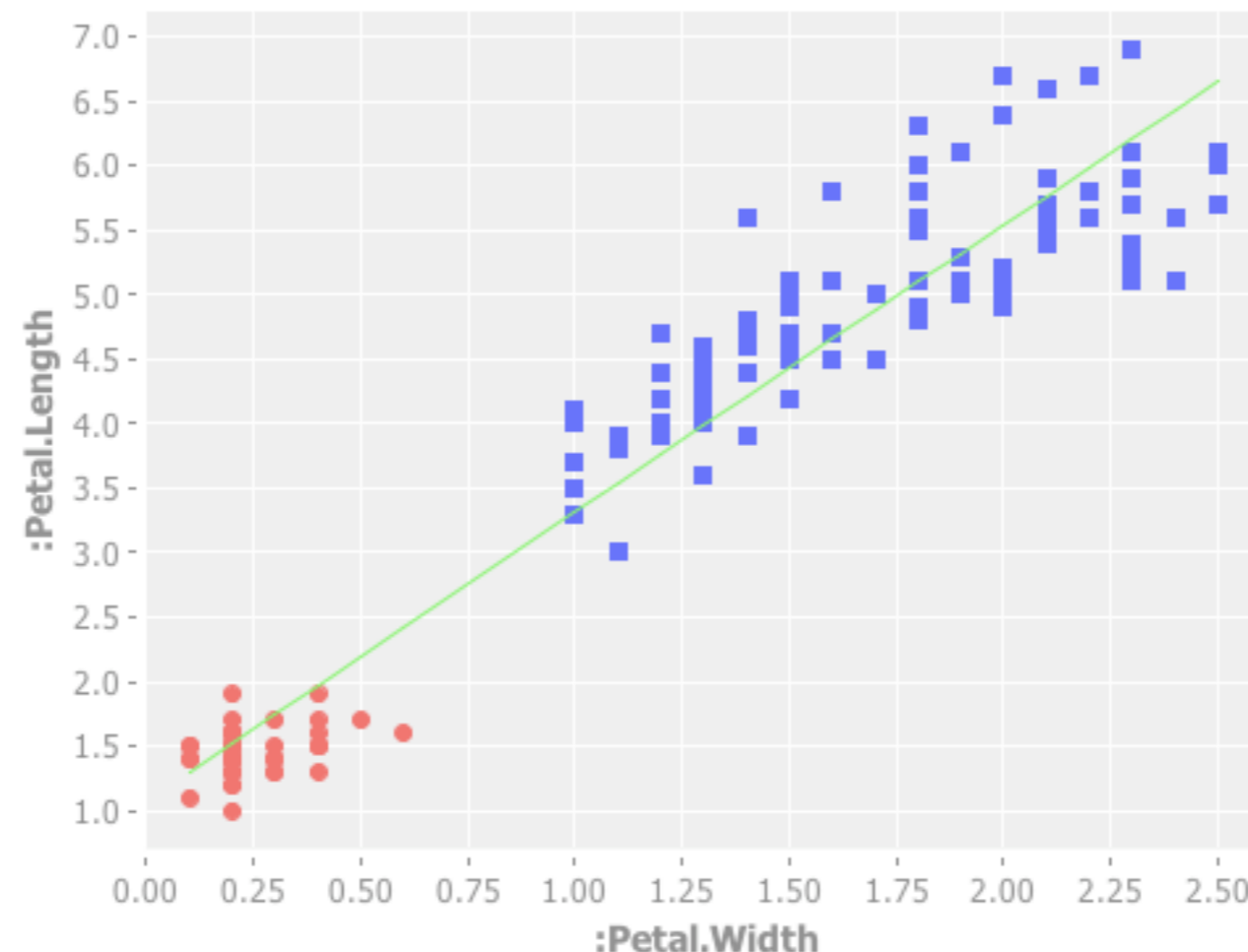
```
  (let [lm (linear-model ($ :Petal.Length) ($ :Petal.Width))]
```

```
    (doto (scatter-plot :Petal.Width :Petal.Length
                       :data ($where {"Petal.Length" {:lte 2.0}
                                       "Petal.Width"  {:lt 0.75}})))
```

```
      (add-points :Petal.Width :Petal.Length
                  :data ($where {"Petal.Length" {:gt 2.0}
                                  "Petal.Width"  {:gte 0.75}})))
```

```
      (add-lines :Petal.Width (:fitted lm))
```

```
    view)))
```



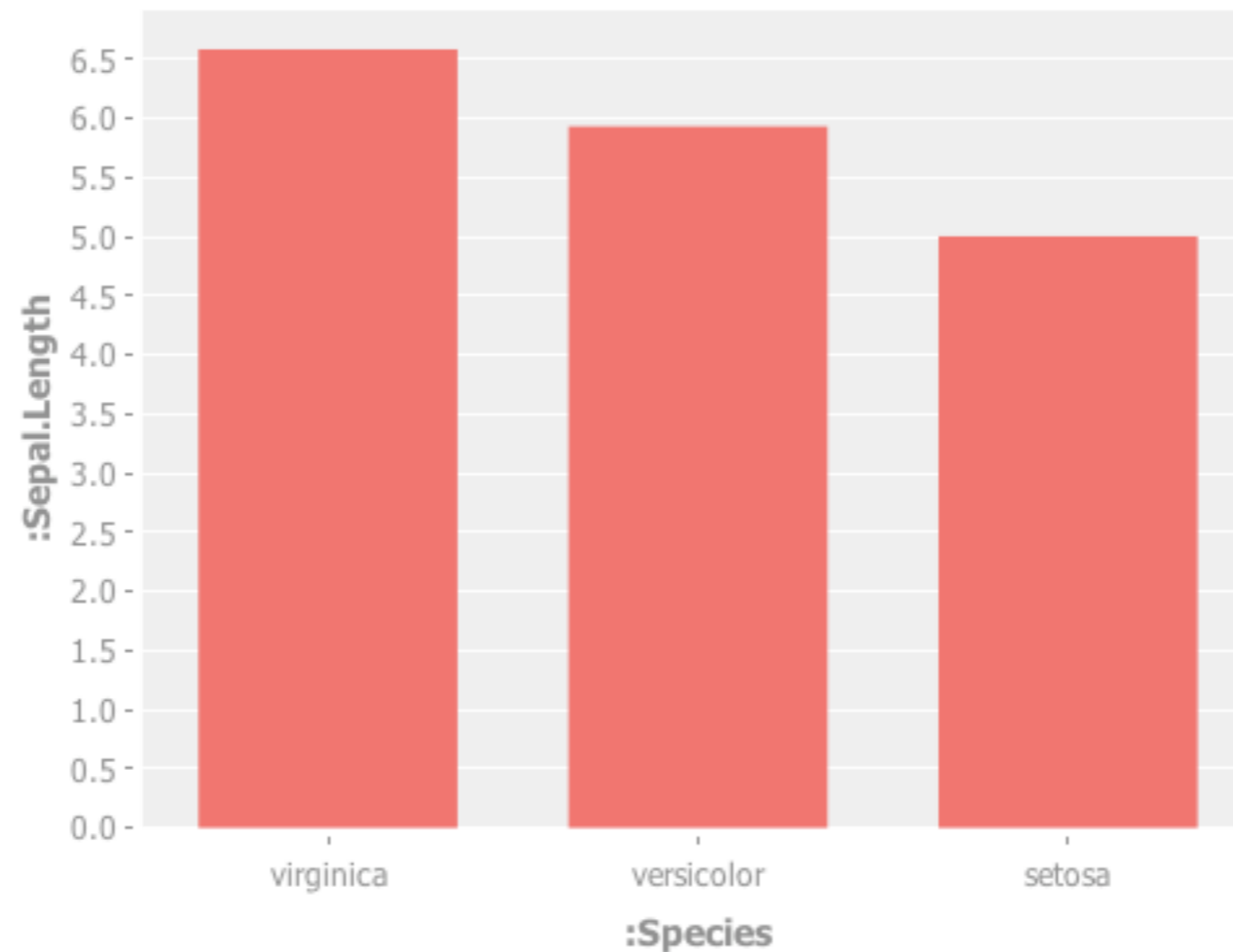
# Bar & line charts

*bar-chart of mean sepal-length for each species*

```
(use '(incanter core charts datasets))
```

```
(with-data ($rollup mean :Sepal.Length :Species  
                  (get-dataset :iris))
```

```
(view (bar-chart :Species :Sepal.Length)))
```



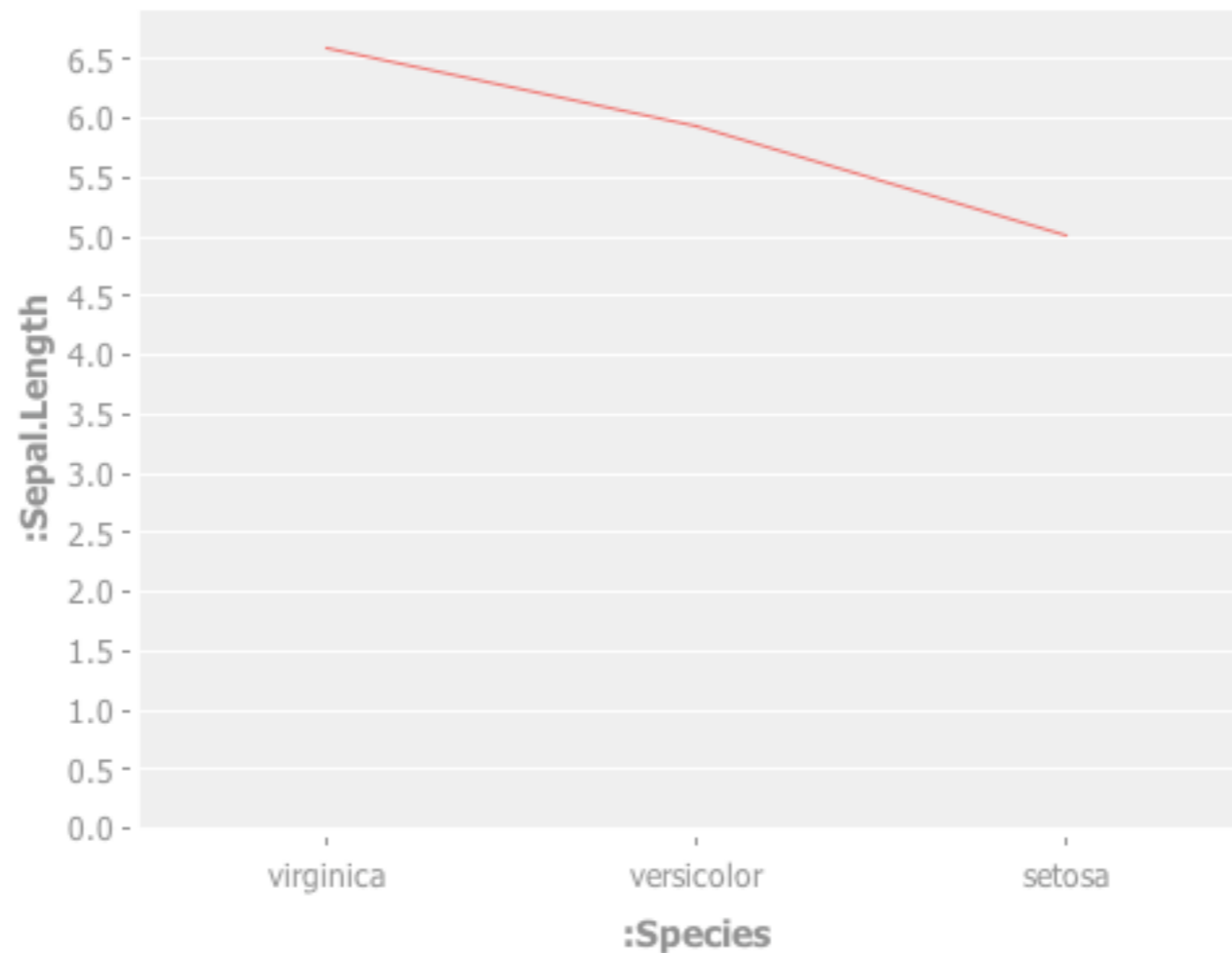
# Bar & line charts

## *line-chart of mean sepal-length for each species*

```
(use '(incanter core charts datasets))
```

```
(with-data ($rollup mean :Sepal.Length :Species  
                  (get-dataset :iris))
```

```
  (view (line-chart :Species :Sepal.Length)))
```



# Bar & line charts

*rollup the :count column using mean*

```
(with-data ($rollup :mean :count [:hair :eye]
                  (get-dataset :hair-eye-color))
  (view $data)
  (view (bar-chart :hair :count
                  :group-by :eye
                  :legend true
                  :theme :dark)))
```

:eye	:hair	:count
green	black	5/2
hazel	red	7
green	blond	8
green	brown	29/2
blue	black	10
green	red	7
brown	black	34
blue	blond	47
blue	brown	42
brown	blond	7/2
blue	red	17/2
brown	brown	119/2
brown	red	13
hazel	black	15/2
hazel	blond	5
hazel	brown	27

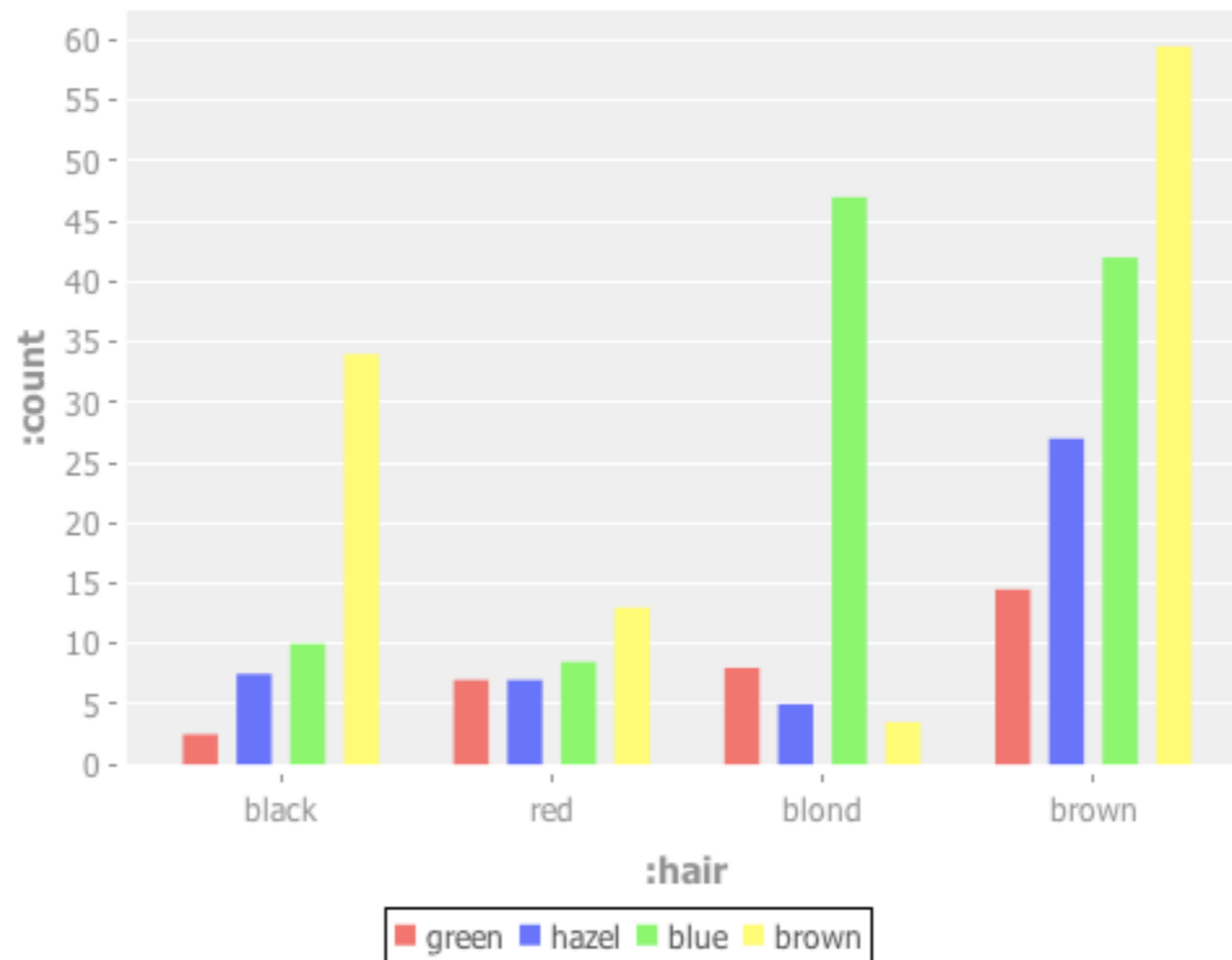


# Bar & line charts

*bar-chart grouped by eye color*

```
(with-data ($rollup :mean :count [:hair :eye]
                 (get-dataset :hair-eye-color))
  (view $data)
  (view (bar-chart :hair :count
                  :group-by :eye
                  :legend true)))
```

:eye	:hair	:count
green	black	5/2
hazel	red	7
green	blond	8
green	brown	29/2
blue	black	10
green	red	7
brown	black	34
blue	blond	47
blue	brown	42
brown	blond	7/2
blue	red	17/2
brown	brown	119/2
brown	red	13
hazel	black	15/2
hazel	blond	5
hazel	brown	27

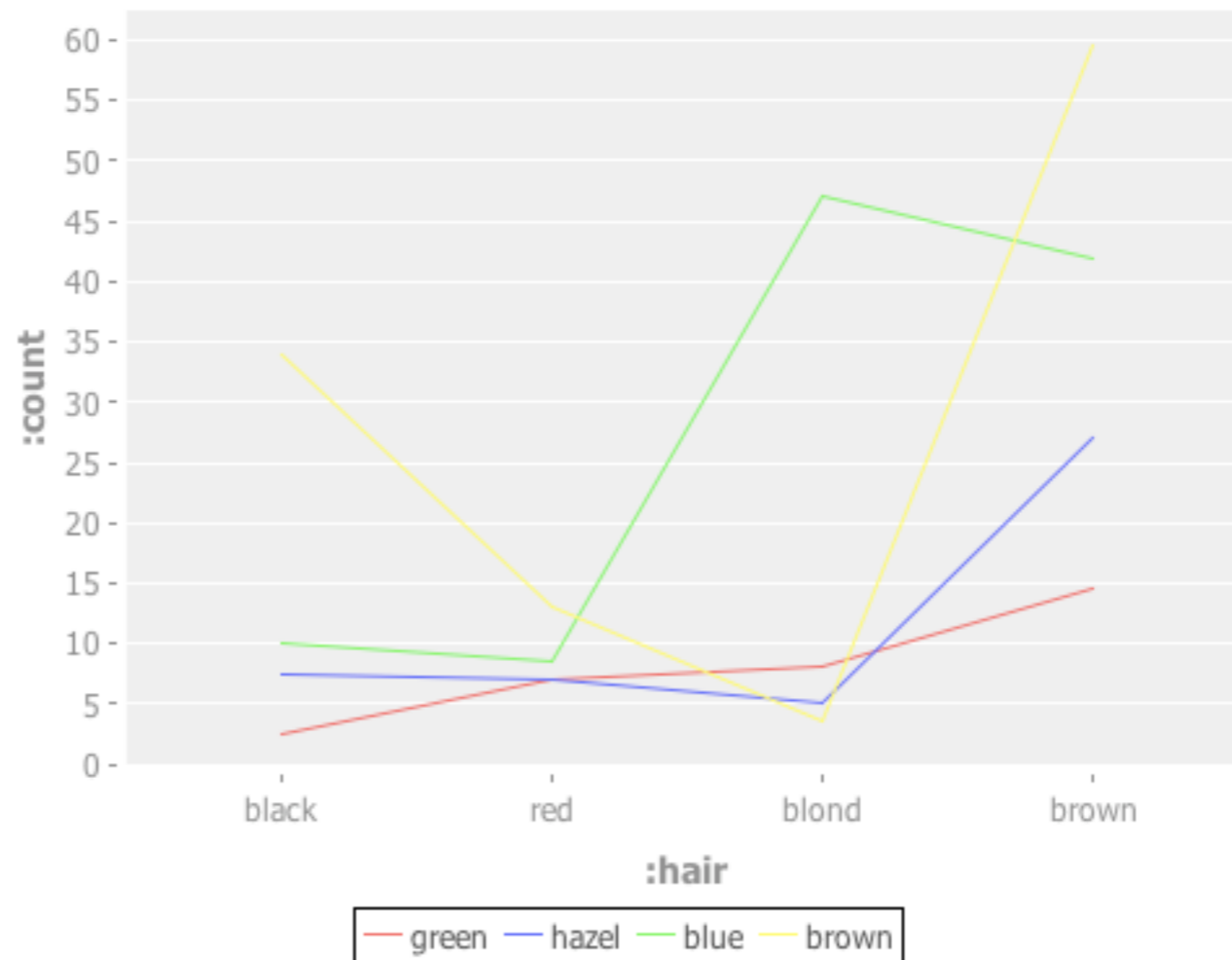


# Bar & line charts

## *line-chart grouped by eye color*

```
(with-data ($rollup :mean :count [:hair :eye]
                (get-dataset :hair-eye-color))
  (view $data)
  (view (line-chart :hair :count
                    :group-by :eye
                    :legend true)))
```

:eye	:hair	:count
green	black	5/2
hazel	red	7
green	blond	8
green	brown	29/2
blue	black	10
green	red	7
brown	black	34
blue	blond	47
blue	brown	42
brown	blond	7/2
blue	red	17/2
brown	brown	119/2
brown	red	13
hazel	black	15/2
hazel	blond	5
hazel	brown	27



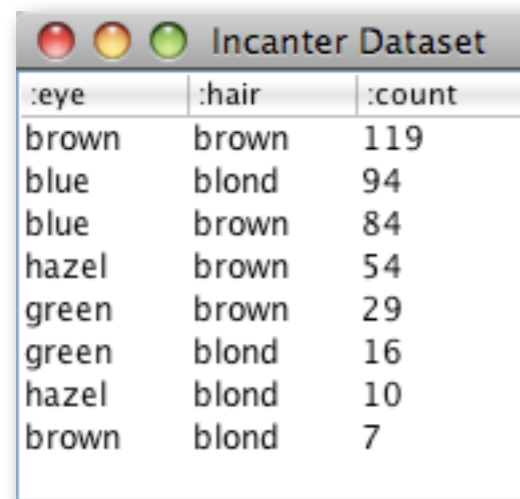
# Bar & line charts

*sort by sum of :count column*

```
(with-data (->> (get-dataset :hair-eye-color)
                ($where {:hair {:in #{"brown" "blond"}}})
                ($rollup :sum :count [:hair :eye])
                ($order :count :desc))
```

**(view \$data)**

```
(view (bar-chart :hair :count
                 :group-by :eye
                 :legend true))
```



:eye	:hair	:count
brown	brown	119
blue	blond	94
blue	brown	84
hazel	brown	54
green	brown	29
green	blond	16
hazel	blond	10
brown	blond	7



# Bar & line charts

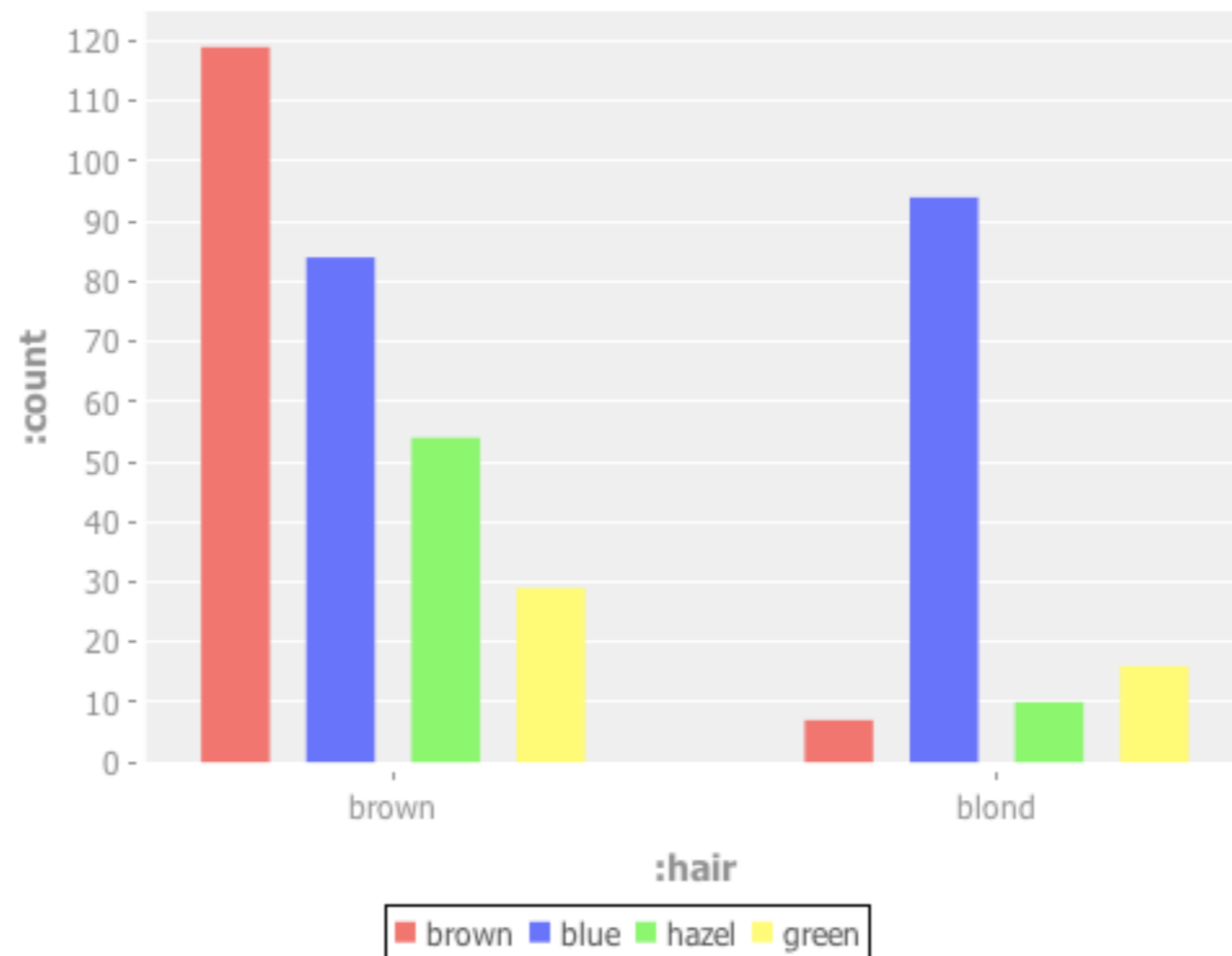
## *bar-chart grouped by eye color*

```
(with-data (->> (get-dataset :hair-eye-color)
  ($where {:hair {:in #{"brown" "blond"}}})
  ($rollup :sum :count [:hair :eye])
  ($order :count :desc))
```

```
(view $data)
```

```
(view (bar-chart :hair :count
  :group-by :eye
  :legend true))
```

:eye	:hair	:count
brown	brown	119
blue	blond	94
blue	brown	84
hazel	brown	54
green	brown	29
green	blond	16
hazel	blond	10
brown	blond	7

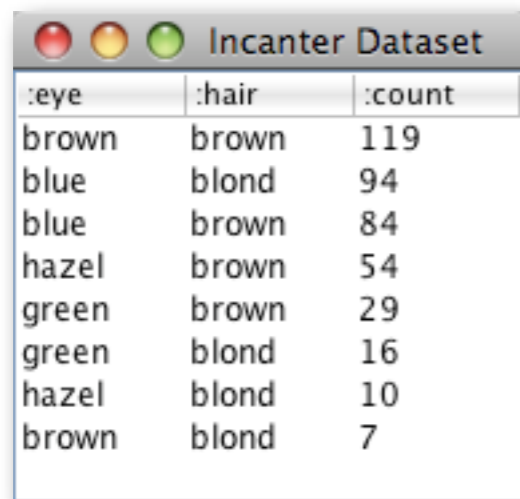


# Bar & line charts

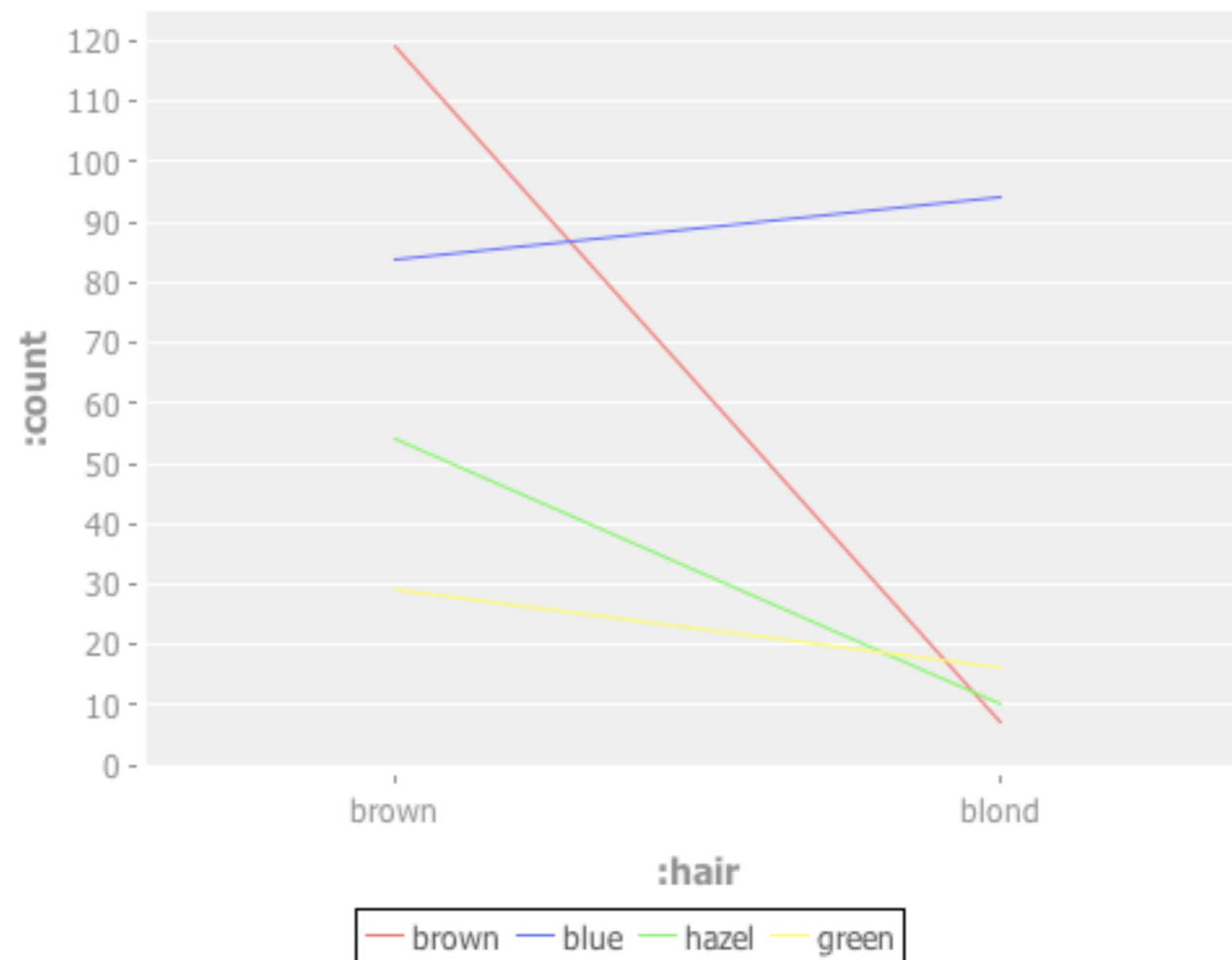
## *line-chart grouped by eye color*

```
(with-data (->> (get-dataset :hair-eye-color)
  ($where {:hair {:in #{"brown" "blond"}}})
  ($rollup :sum :count [:hair :eye])
  ($order :count :desc))

(view $data)
(view (line-chart :hair :count
  :group-by :eye
  :legend true))
```



:eye	:hair	:count
brown	brown	119
blue	blond	94
blue	brown	84
hazel	brown	54
green	brown	29
green	blond	16
hazel	blond	10
brown	blond	7

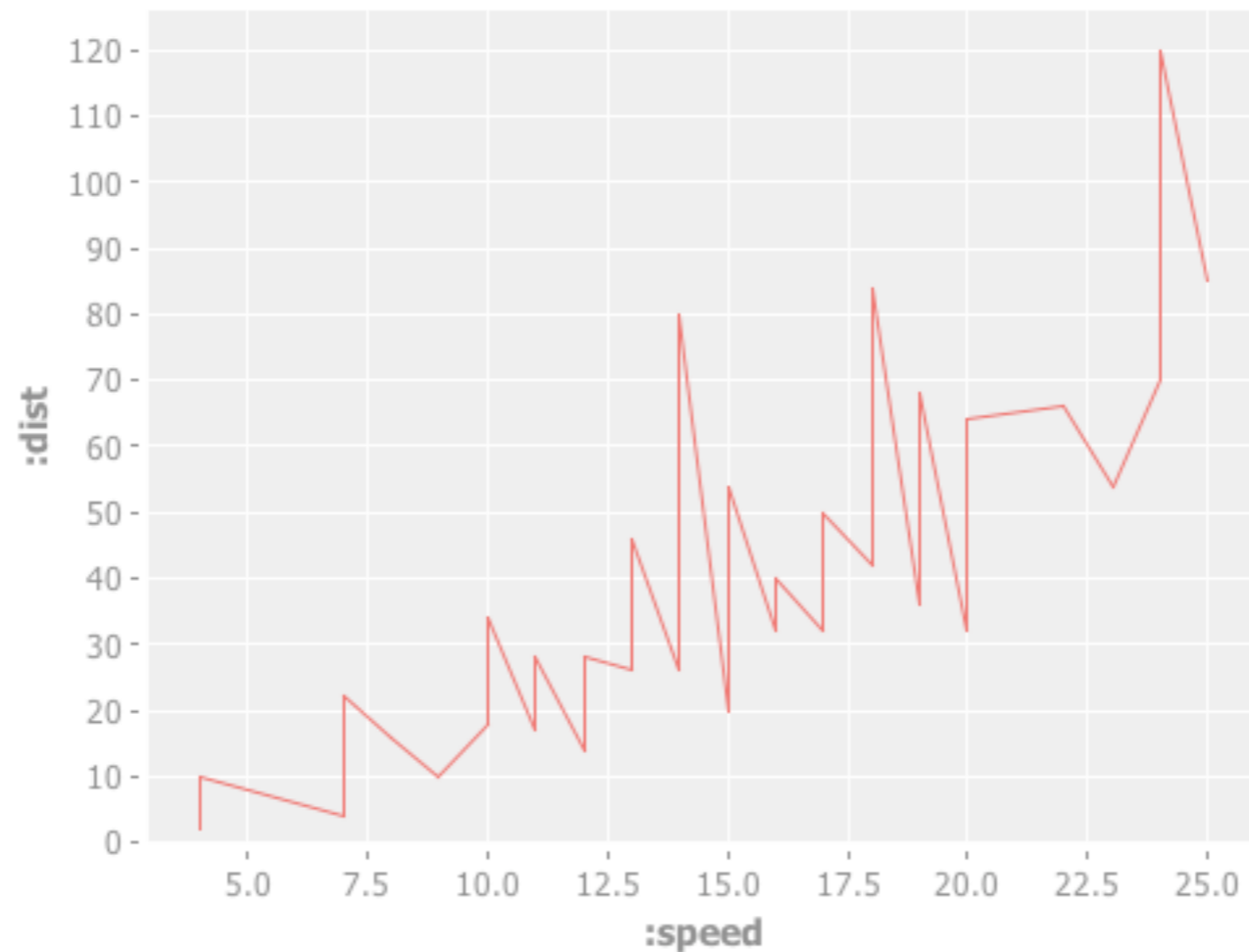


# XY & function plots

*xy-plot of two continuous variables*

```
(use '(incanter core charts))
```

```
(with-data (get-dataset :cars)  
  (view (xy-plot :speed :dist)))
```

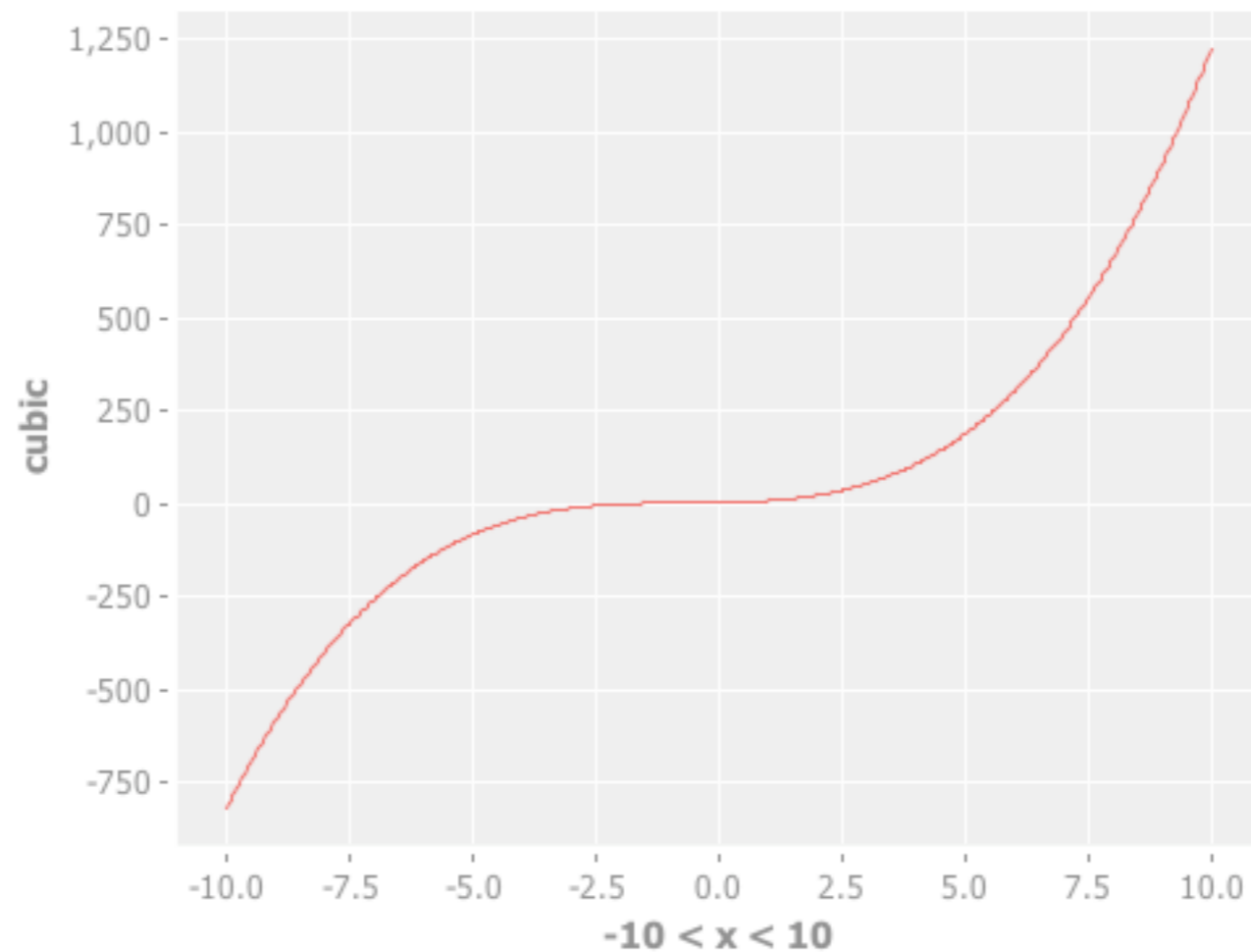


```
(use '(incanter core charts optimize))
```

```
(defn cubic [x] (+ (* x x x) (* 2 x x) (* 2 x) 3))
```

```
(doto (function-plot cubic -10 10)
```

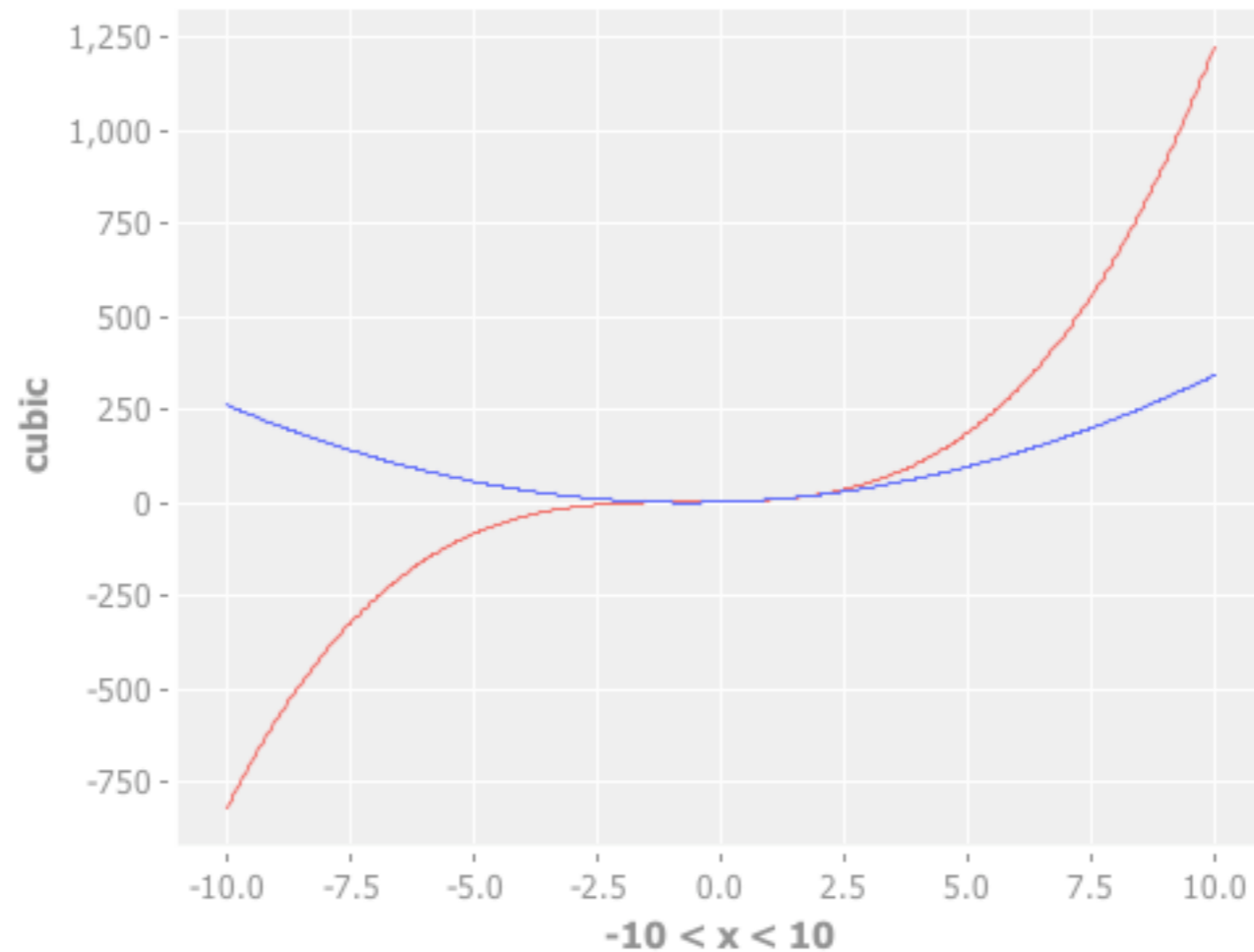
```
view)
```



# XY & function plots

*add the derivative of the function*

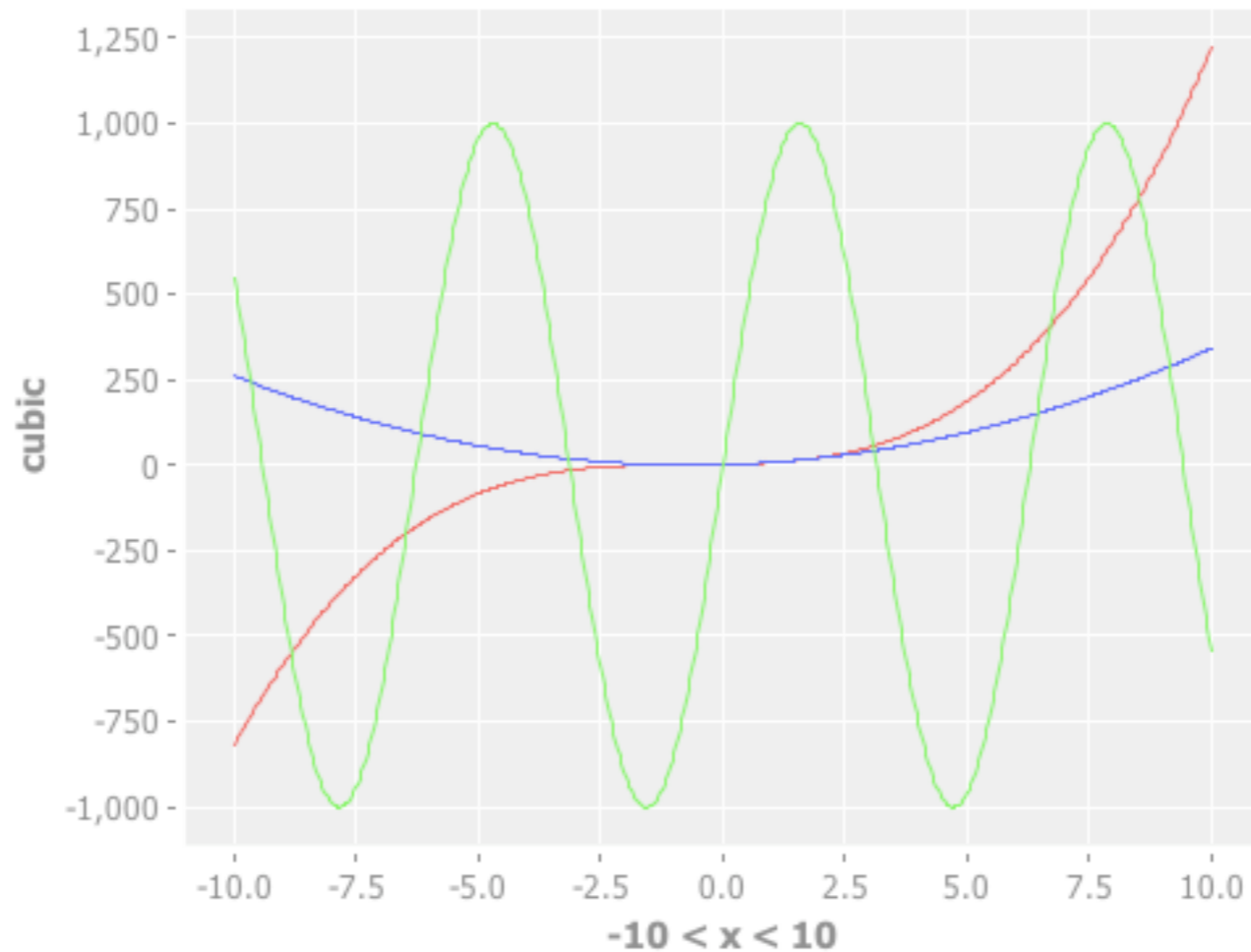
```
(use ' (incanter core charts optimize))  
  
(defn cubic [x] (+ (* x x x) (* 2 x x) (* 2 x) 3))  
  
(doto (function-plot cubic -10 10)  
  (add-function (derivative cubic) -10 10)  
  
  view)
```



# XY & function plots

*add a sine wave*

```
(use '(incanter core charts optimize))  
  
(defn cubic [x] (+ (* x x x) (* 2 x x) (* 2 x) 3))  
  
(doto (function-plot cubic -10 10)  
  (add-function (derivative cubic) -10 10)  
  (add-function #(* 1000 (sin %)) -10 10)  
  view)
```

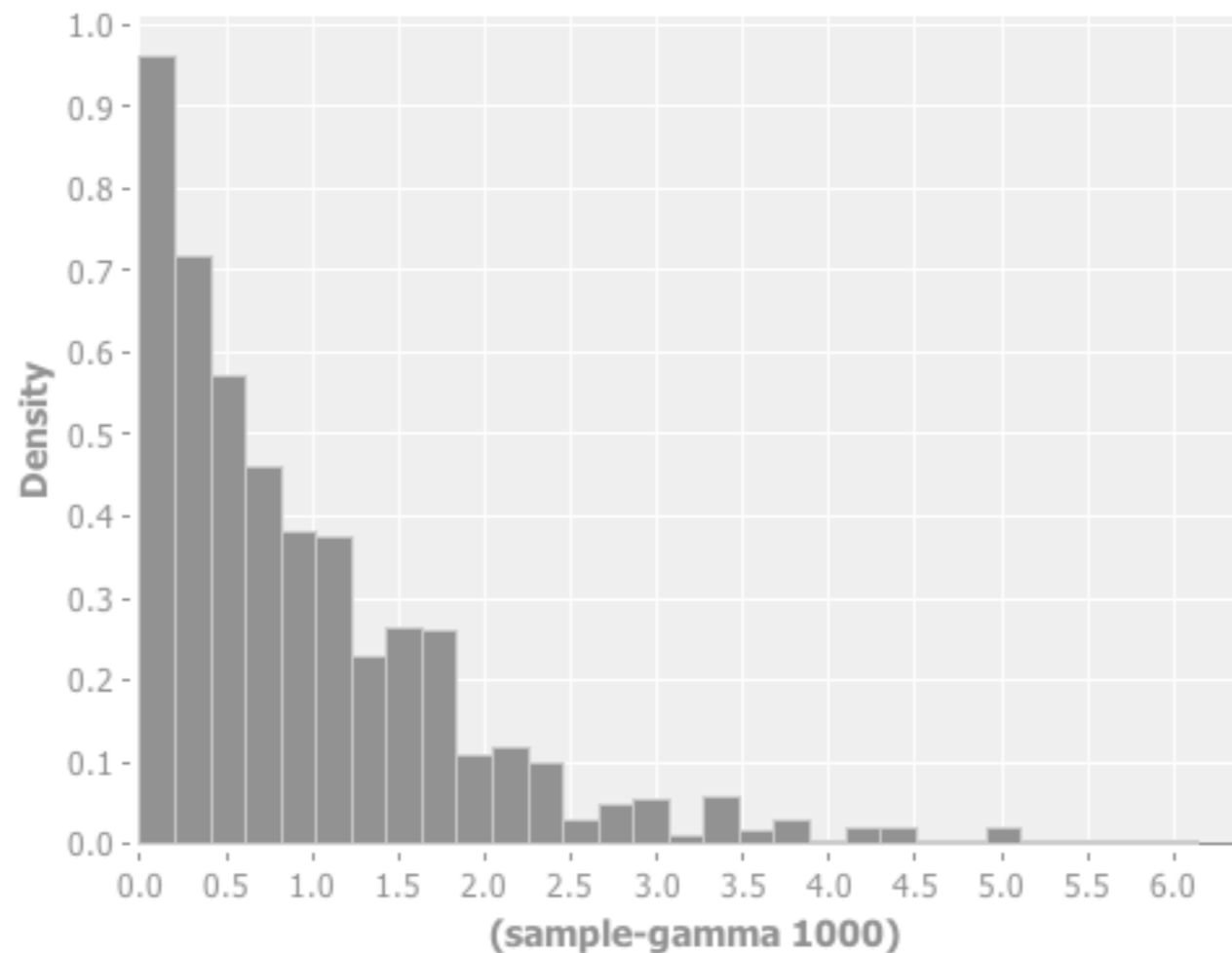


# Histograms & box-plots

*plot a sample from a gamma distribution*

```
(use '(incanter core charts stats))  
(doto (histogram (sample-gamma 1000)  
              :density true  
              :nbins 30)
```

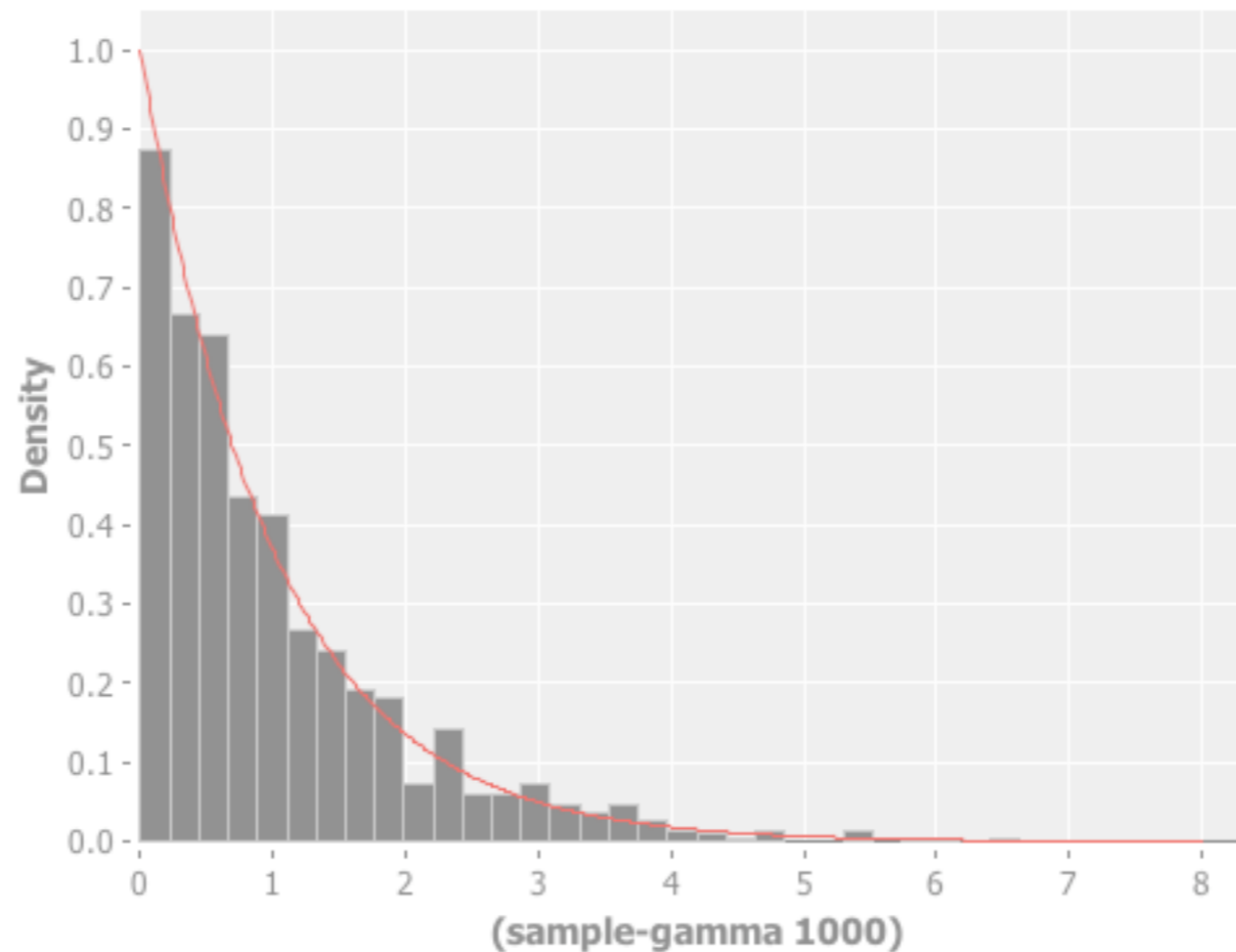
```
view)
```



# Histograms & box-plots

*add a gamma pdf line*

```
(use '(incanter core charts stats))  
(doto (histogram (sample-gamma 1000)  
              :density true  
              :nbins 30)  
      (add-function pdf-gamma 0 8)  
      view)
```

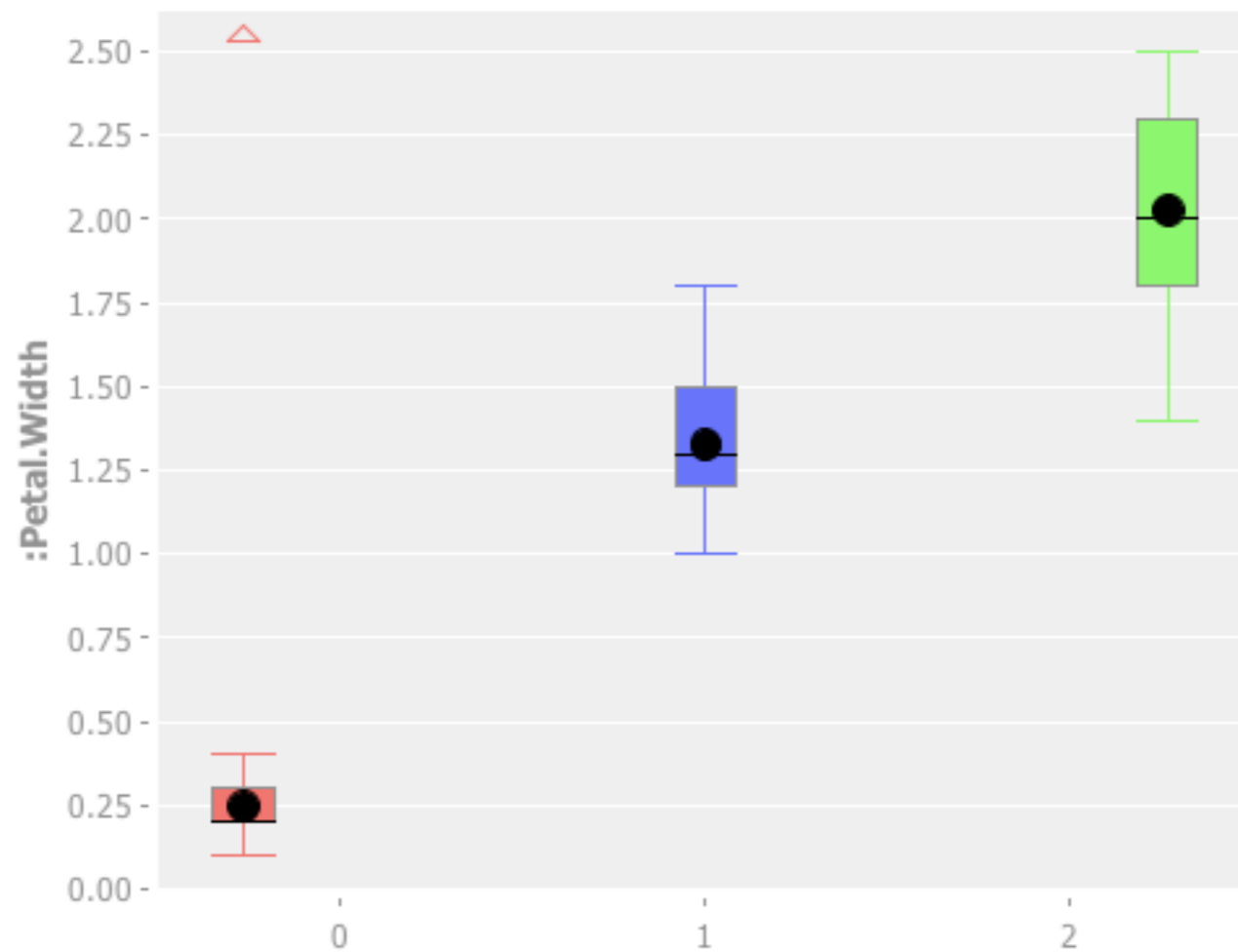


# Histograms & box-plots

*box-plots of petal-width grouped by species*

```
(use '(incanter core datasets))
```

```
(with-data (get-dataset :iris)  
  (view (box-plot :Petal.Width  
                :group-by :Species)))
```

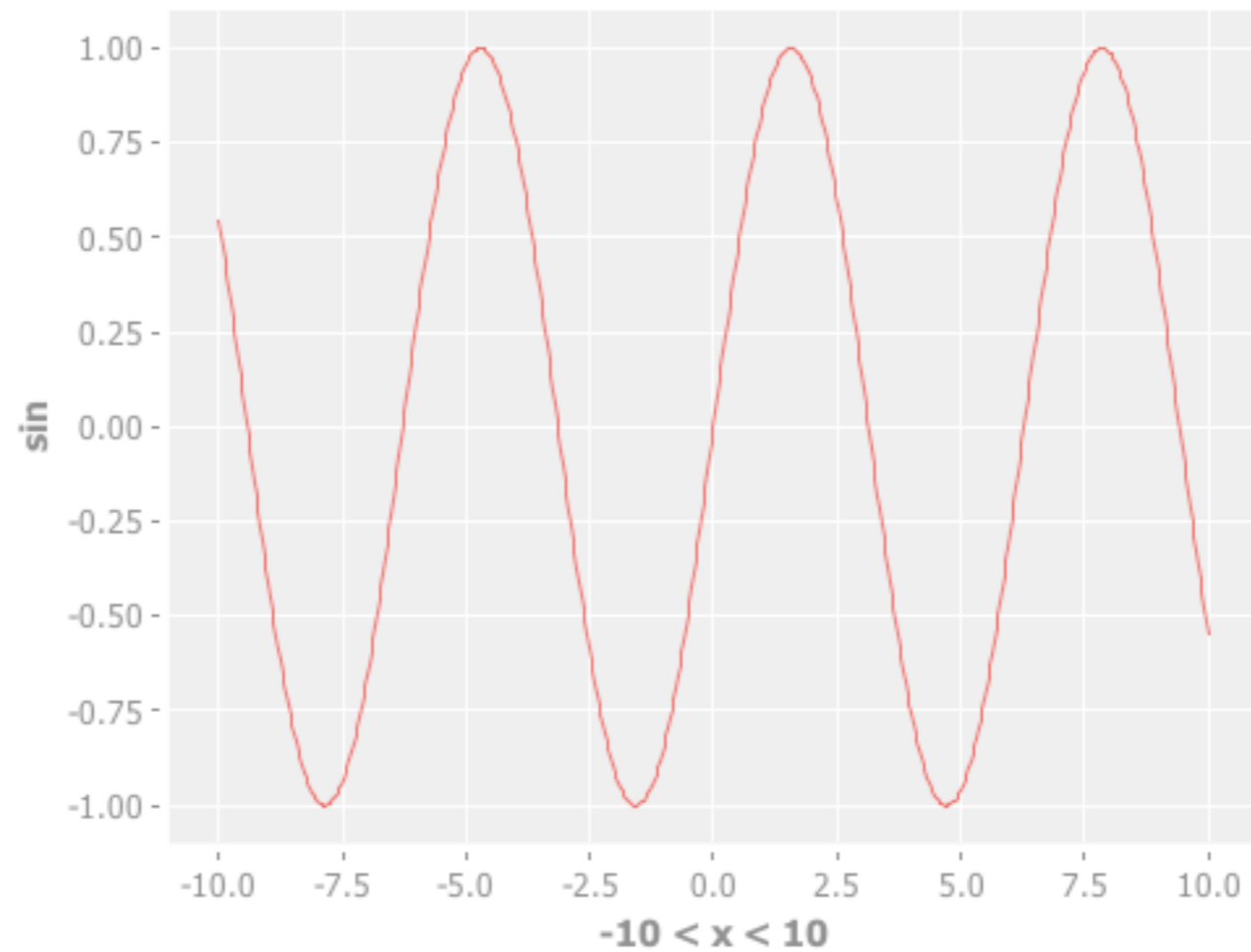


# Annotating charts

*plot sin wave*

```
(use ' (incanter core charts))  
(doto (function-plot sin -10 10)
```

`view)`

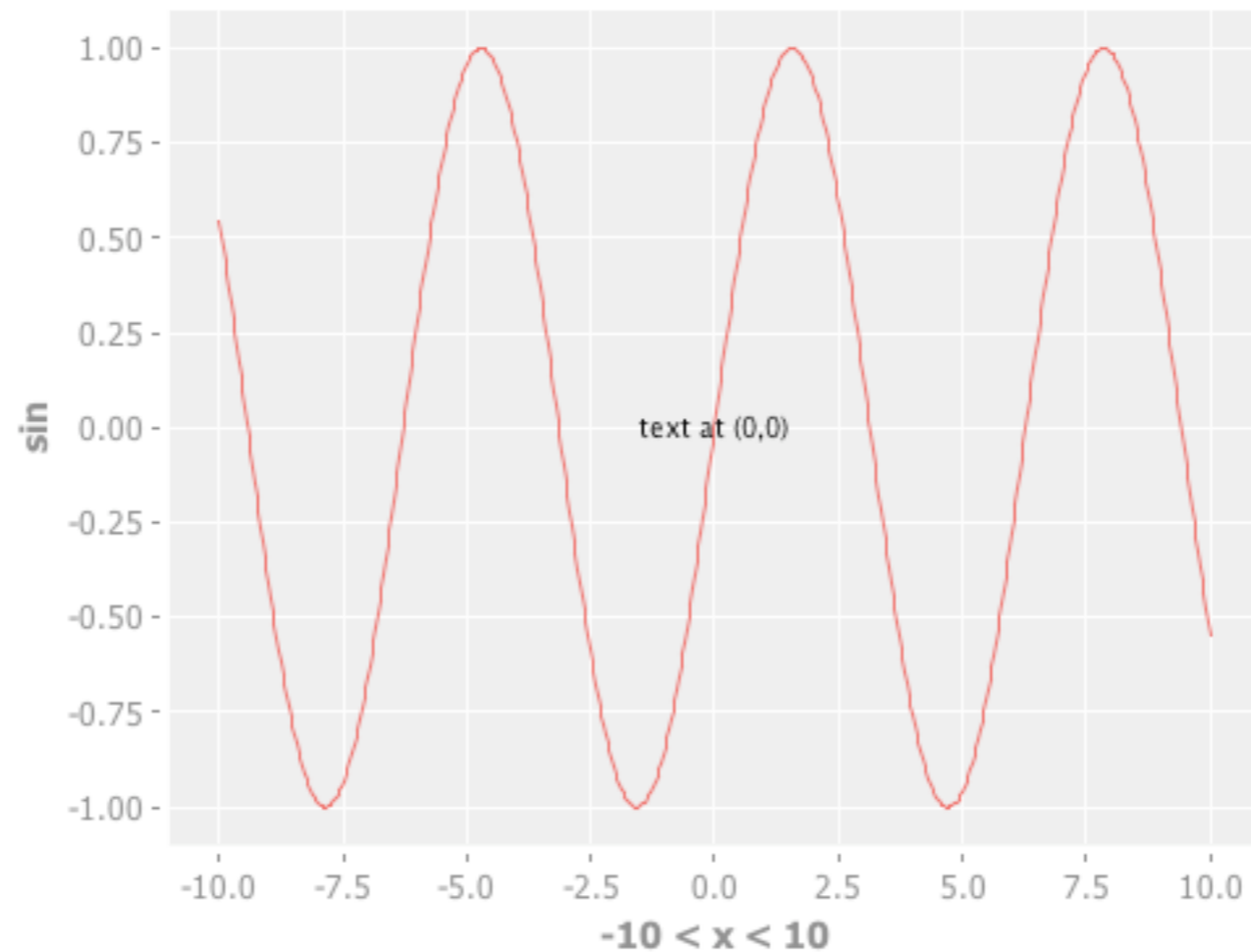


# Annotating charts

*add text annotation (black text only)*

```
(use '(incanter core charts))  
(doto (function-plot sin -10 10)  
      (add-text 0 0 "text at (0,0)"))
```

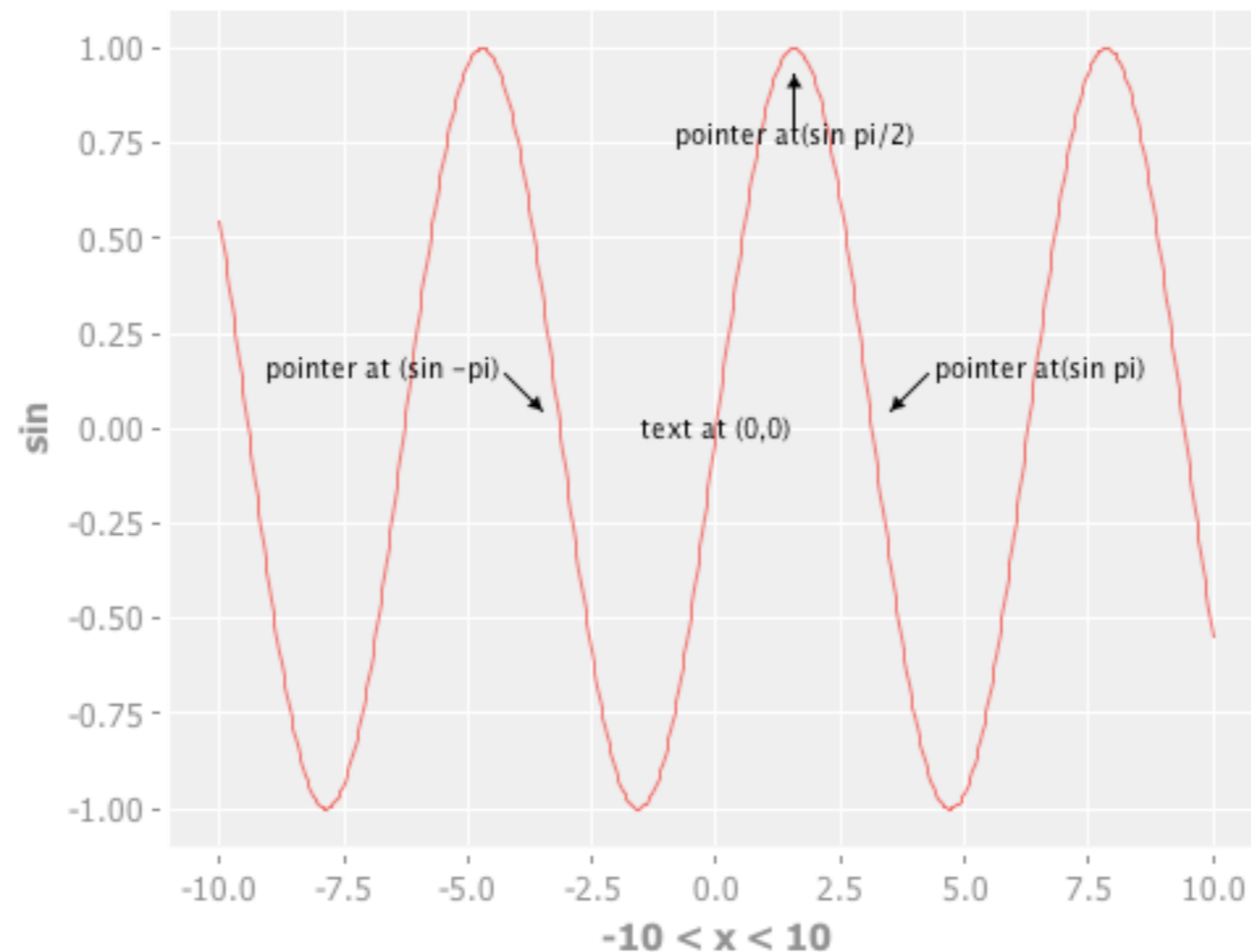
view)



# Annotating charts

*add pointer annotations*

```
(use ' (incanter core charts))
(doto (function-plot sin -10 10)
  (add-text 0 0 "text at (0,0)")
  (add-pointer (- Math/PI) (sin (- Math/PI))
    :text "pointer at (sin -pi)")
  (add-pointer Math/PI (sin Math/PI)
    :text "pointer at(sin pi)"
    :angle :ne)
  (add-pointer (* 1/2 Math/PI) (sin (* 1/2 Math/PI))
    :text "pointer at(sin pi/2)"
    :angle :south)
  view)
```



# Outline

---

## Overview

- What is Incanter?
- Getting started
- Incanter libraries

## Datasets

- Creating
- Reading data
- Saving data
- Column selection
- Row selection
- Sorting
- Rolling up

## Charts

- Scatter plots
- Chart options
- Saving charts
- Adding data
- Bar & line charts
- XY & function plots
- Histograms & box-plots
- Annotating charts

## Wrap up



## Learn more

- Visit <http://incanter.org>
- Visit <http://data-sorcery.org>

## Join the community and contribute

- Join the Google group: <http://groups.google.com/group/incanter>
- Follow Incanter on Github: <http://github.com/liebke/incanter>
- Follow @liebke on Twitter

## Get the slides and code from this presentation

<http://data-sorcery.org/2010/02/11/data-sorcery-pt1>



Thank you

