
daru - Data Analysis in RUBY

chat on gitter chat on gitter chat on gitter code helpers 9

Introduction

daru (Data Analysis in RUBY) is a library for storage, analysis, manipulation and visualization of data in Ruby.

daru makes it easy and intuitive to process data predominantly through 2 data structures: `Daru::DataFrame` and `Daru::Vector`. Written in pure Ruby works with all ruby implementations. Tested with MRI 2.5.1 and 2.7.1.

daru plugin gems

- **daru-view**

daru-view is for easy and interactive plotting in web application & IRuby notebook. It can work in any Ruby web application frameworks like Rails, Sinatra, Nanoc and hopefully in others too.

Articles/Blogs, that summarize powerful features of daru-view:

- GSoC 2017 daru-view
- GSoC 2018 Progress Report
- HighCharts Official blog post regarding daru-view

- **daru-io**

This gem extends support for many Import and Export methods of `Daru::DataFrame`. This gem is intended to help Rubyists who are into Data Analysis or Web Development, by serving as a general purpose conversion library that takes input in one format (say, JSON) and converts it another format (say, Avro) while also making it incredibly easy to getting started on analyzing data with daru. One can read more in SciRuby/blog/daru-io.

Features

- Data structures:
 - Vector - A basic 1-D vector.

-
- DataFrame - A 2-D spreadsheet-like structure for manipulating and storing data sets. This is daru's primary data structure.

- Compatible with IRuby notebook, statsample, statsample-glm and statsample-timeseries.
- Support for time series.
- Singly and hierarchically indexed data structures.
- Flexible and intuitive API for manipulation and analysis of data.
- Easy plotting, statistics and arithmetic.
- Plentiful iterators.
- Optional speed and space optimization on MRI with NMatrix and GSL.
- Easy splitting, aggregation and grouping of data.
- Quickly reducing data with pivot tables for quick data summary.
- Import and export data from and to Excel, CSV, SQL Databases, ActiveRecord and plain text files.

Installation

```
1 $ gem install daru
```

Notebooks

Notebooks on most use cases

- Overview of most daru functions
- Basic Creation of Vectors and DataFrame
- Detailed Usage of Daru::Vector
- Detailed Usage of Daru::DataFrame
- Searching and combining data in daru
- Grouping, Splitting and Pivoting Data
- Usage of Categorical Data

Visualization

- Visualizing Data With Daru::DataFrame
- Plotting using Nyaplot
- Plotting using GnuplotRB
- Vector plotting with Gruff
- DataFrame plotting with Gruff

Notebooks on Time series

- Basic Time Series
- Time Series Analysis and Plotting

Notebooks on Indexing

- Indexing in Vector
- Indexing in DataFrame

Case Studies

- Logistic Regression Analysis with daru and statsample-glm
- Finding and Plotting most heard artists from a Last.fm dataset
- Analyzing baby names with daru
- Example usage of Categorical Data
- Example usage of Categorical Index

Blog Posts

- Data Analysis in RUBY: Basic data manipulation and plotting
- Data Analysis in RUBY: Splitting, sorting, aggregating data and data types
- Finding and Combining data in daru
- Introduction to analyzing datasets with daru library

Time series

- Analysis of Time Series in daru
- Date Offsets in Daru

Categorical Data

- Categorical Index
- Categorical Data
- Visualization with Categorical Data

Basic Usage

daru exposes two major data structures: `DataFrame` and `Vector`. The `Vector` is a basic 1-D structure corresponding to a labelled Array, while the `DataFrame` - daru's primary data structure - is 2-D spreadsheet-like structure for manipulating and storing data sets.

Basic `DataFrame` initialization.

```
1 data_frame = Daru::DataFrame.new(  
2   {  
3     'Beer' => ['Kingfisher', 'Snow', 'Bud Light', 'Tiger Beer', '  
4       Budweiser'],  
5     'Gallons sold' => [500, 400, 450, 200, 250]  
6   },  
7   index: ['India', 'China', 'USA', 'Malaysia', 'Canada']  
8 )  
9 data_frame
```

Out[3]:

Daru::DataFrame:97667760 rows: 5 cols: 2		
	Beer	Gallons sold
India	Kingfisher	500
China	Snow	400
USA	Bud Light	450
Malaysia	Tiger Beer	200
Canada	Budweiser	250

Load data from CSV files.

```
1 df = Daru::DataFrame.from_csv('TradeoffData.csv')
```

Out[5]:

Daru::DataFrame:98110580 rows: 64 cols: 4				
	Group	RelativeFitness	Replicate	Treatment
0	BKB	0.869962555792838	1	Tube
1	BKB	1.00036299125423	2	Tube
2	BKB	0.982935090384188	3	Tube
3	BAC	0.810391635206191	1	Tube
4	BAC	0.795106571577928	2	Tube
5	JDK	0.849203581734814	1	Tube
6	JDK	0.917636977577209	2	Tube

Basic Data Manipulation

Selecting rows.

```
1 data_frame.row['USA']
```

Out[6]:

Daru::Vector:73120330 size: 2	
	USA
Beer	Bud Light
Gallons sold	450

Selecting columns.

```
1 data_frame['Beer']
```

Out[8]:

Daru::Vector:85747500 size: 5	
	Beer
India	Kingfisher
China	Snow
USA	Bud Light
Malaysia	Tiger Beer
Canada	Budweiser

A range of rows.

```
1 data_frame.row['India'..'USA']
```

Out[11]:

Daru::DataFrame:85582940 rows: 3 cols: 2		
	Beer	Gallons sold
China	Snow	400
USA	Bud Light	450
Malaysia	Tiger Beer	200

The first 2 rows.

```
1 data_frame.first(2)
```

Out[4]:

Daru::DataFrame:86627850 rows: 2 cols: 2		
	Beer	Gallons sold
India	Kingfisher	500
China	Snow	400

The last 2 rows.

```
1 data_frame.last(2)
```

Out[5]:

Daru::DataFrame:86722320 rows: 2 cols: 2		
	Beer	Gallons sold
Malaysia	Tiger Beer	200
Canada	Budweiser	250

Adding a new column.

```
1 data_frame['Gallons produced'] = [550, 500, 600, 210, 240]
```

Out[8]:

Daru::DataFrame:86411760 rows: 5 cols: 3			
	Beer	Gallons sold	Gallons produced
India	Kingfisher	500	550
China	Snow	400	500
USA	Bud Light	450	600
Malaysia	Tiger Beer	200	210
Canada	Budweiser	250	240

Creating a new column based on data in other columns.

```
1 data_frame['Demand supply gap'] = data_frame['Gallons produced'] -  
  data_frame['Gallons sold']
```

Out[11]:

Daru::DataFrame:86411760 rows: 5 cols: 4				
	Beer	Gallons sold	Gallons produced	Demand supply gap
India	Kingfisher	500	550	50
China	Snow	400	500	100
USA	Bud Light	450	600	150
Malaysia	Tiger Beer	200	210	10
Canada	Budweiser	250	240	-10

Condition based selection

Selecting countries based on the number of gallons sold in each. We use a syntax similar to that defined by Arel, i.e. by using the `where` clause.

```
1 data_frame.where(data_frame['Gallons sold'].lt(300))
```

Out[6]:

Daru::DataFrame:88142620 rows: 2 cols: 4				
	Beer	Gallons sold	Gallons produced	Demand supply gap
Malaysia	Tiger Beer	200	210	10
Canada	Budweiser	250	240	-10

You can pass a combination of boolean operations into the `#where` method and it should work fine:

```
1 data_frame.where(
2   data_frame['Beer']
3   .in(['Snow', 'Kingfisher', 'Tiger Beer'])
4   .and(
5     data_frame['Gallons produced'].gt(520).or(data_frame['Gallons
6       produced'].lt(250))
7   )
8 )
```

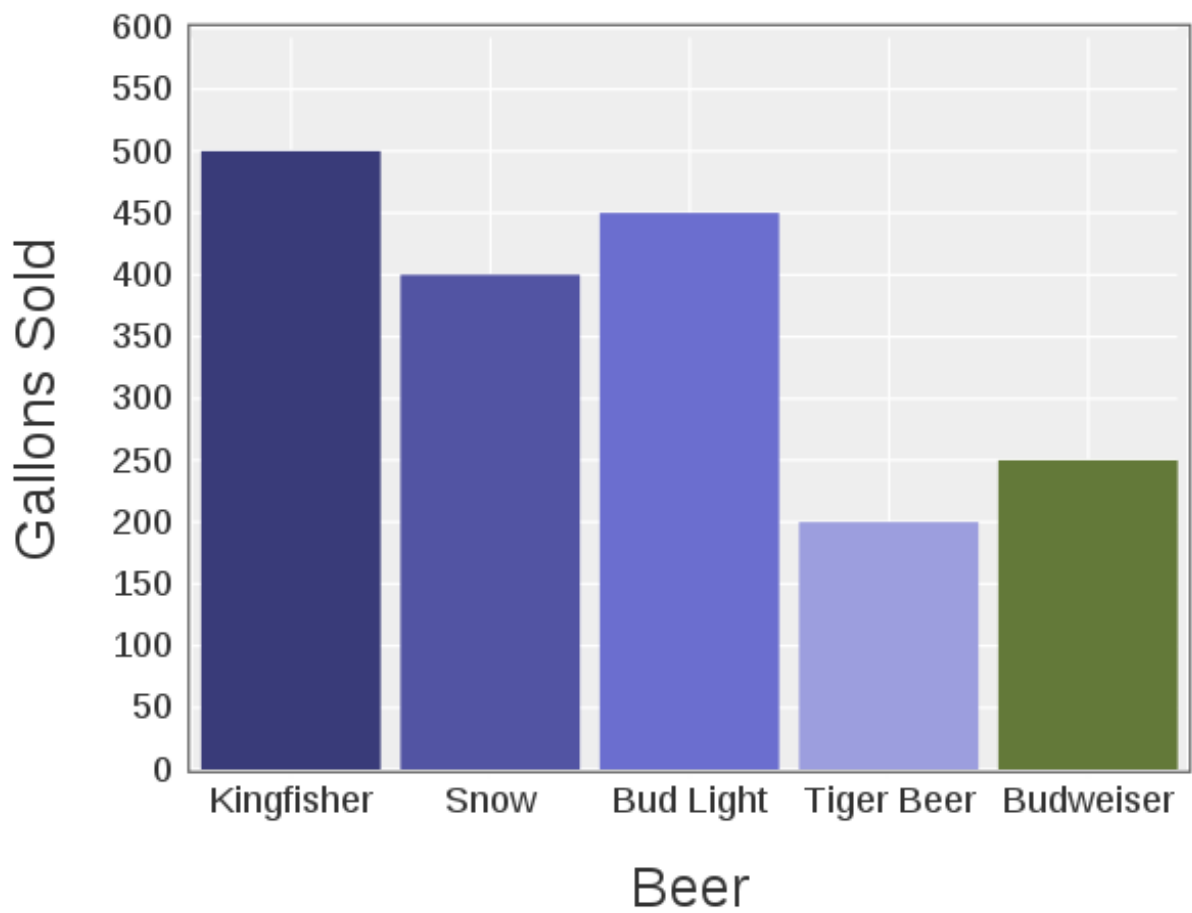
Out[9]:

Daru::DataFrame:88598430 rows: 2 cols: 4				
	Beer	Gallons sold	Gallons produced	Demand supply gap
India	Kingfisher	500	550	50
Malaysia	Tiger Beer	200	210	10

Plotting

Daru supports plotting of interactive graphs with nyaplot. You can easily create a plot with the `#plot` method. Here we plot the gallons sold on the Y axis and name of the brand on the X axis in a bar graph.

```
1 data_frame.plot type: :bar, x: 'Beer', y: 'Gallons sold' do |plot,
2   diagram|
3   plot.x_label "Beer"
4   plot.y_label "Gallons Sold"
5   plot.yrange [0,600]
6   plot.width 500
7   plot.height 400
8 end
```



In addition to `nyaplot`, `daru` also supports plotting out of the box with `gnuplotrb`.

Documentation

Docs can be found [here](#).

Contributing

Pick a feature from the Roadmap or the issue tracker or think of your own and send me a Pull Request!

For details see CONTRIBUTING.

Acknowledgements

- Google and the Ruby Science Foundation for the Google Summer of Code 2016 grant for speed enhancements and implementation of support for categorical data. Special thanks to @lokeshh, @zverok and @agisga for their efforts.
- Google and the Ruby Science Foundation for the Google Summer of Code 2015 grant for further developing daru and integrating it with other ruby gems.
- Thank you last.fm for making user data accessible to the public.

Copyright (c) 2015, Sameer Deshmukh All rights reserved