Ruby Association Grant 2018 Final Report

Charty - Visualizing your data in Ruby

Kazuma Furuhashi (@284km)

Project

Charty - Visualizing your data in Ruby

プロジェクト概要(再掲)

Charty is open-source Ruby library for visualizing your data in a simple way. In this project, Implement Data Visualization Tool which has Visualization Layer and Data Abstraction Layer and can support multiple data types and multiple backends. Make the GR Framework available as a backend, and implement Collection Interface and Daru Inteface as Data Abstraction Layer to make it possible to support multiple data types.

プロジェクト計画(再掲)

The scope in Grant is as follows

- (1) Create a thin Ruby library that can use the GR framework as an independent backend.
- (2) Enable GR framework as backend from Charty (use SciRuby/rubyplot as the visualization layer)
- (3) Implements Collection Interface corresponding to Charty's Data Abstraction Layer.
- (4) Implement Daru Interface corresponding to Charty's Data Abstraction Layer.

上記に加え、中間報告の時点で以下 2 つの実装を余力があれば実施するとした。

- (a) Support rubydown (https://github.com/sciruby-jp/rubydown)
- (b) Support ActiveRecord Interface

プロジェクト成果

ソースコード

https://github.com/red-data-tools/charty

プロジェクト達成状況

Grant のスコープである上記 (1)-(4) を達成し、Plotting Abstraction Layer、Data Abstraction Layer の枠組みを作ることが出来た。

(1) (2) の Plotting Abstraction Layer に関するスコープでは、計画していた rubyplot に加え、Matplotlib, Gruff を Charty のサポートする backend に加えることが出来た。

(3) (4) の Data Abstraction Layer に関するスコープでは、計画していた Daru Interface に加え、Numo::NArray、 追加スコープ (b) とした ActiveRecord のサポートを加えることが出来た。

追加スコープ (a) とした rubydown との連携については未達成で、今後の計画に加えている。

Abstraction Layer でのライブラリ対応状況

Charty は 2 つの Abstraction layer を持ち、Charty 自体は薄いライブラリとして存在する。以下の plotting library、データ構造に対応している。

Plotting Abstraction Layer

- Matplotlib
- Gruff
- rubyplot

Data Abstraction Layer

- Daru::DataFrame
- Numo::NArray
- ActiveRecord

Charty によるグラフ出力のイメージをこちらにまとめている。

https://github.com/red-data-tools/charty#examples

プロジェクト進行中に得た知見と課題

Plotting Abstraction Layer に関して

出力可能なグラフの種類が一番多い Matplotlib をリファレンス実装として、その後他のライブラリのサポートを増やしつつ Interface を考えるという手順で作業を進めた。これは、サポートするグラフの種類により描画に必要なデータ構造が異なるため、より多くのグラフ出力に対応した Interface を考えるにあたってはこの手順が有効であった。 当初、GR Framework を backend に持つ rubyplot が魅力的であり真っ先に実装を始めたが、それに続いて Matplotlib のサポートを考えると両ライブラリに対応すること、より多くのグラフ(データ構造)に対応することを同時に考えることとなり難易度が高かったが、村田メンターの助言により順序を変えることで設計の難易度を下げることが出来た。

Data Abstraction Layer に関して

Charty がサポートするデータ構造は複数あるため、各データ構造から一度中間データ構造に変換し、その中間データ構造と Plotting Abstraction Layer とのマッピング処理を用意することにより、新たなデータ構造をサポートしたい場合には中間データ構造にさえ対応するコードを書けばよいという状態を作った。その中間データ構造が Charty::Table である。この点では Charty と思想の近い holoviews(https://github.com/pyviz/holoviews) の Interface を参考にした。 機能の拡充と共に Charty::Table の役割は増えるため、ある程度複雑になる時点で対応するデータ構造に応じた Adapter を用意することを予定している。

今後の計画

- Data Abstraction Layer
  - Support NMatrix
  - Support Red::Arrow
  - Support benchmark_driver (ベンチマーク結果の可視化)
- Plotting Abstraction Layer
  - 出力可能なグラフの追加
  - Support rubydown (https://github.com/sciruby-jp/rubydown)
- 発表など
  - 3/24 の Rails Developers Meetup 2019 で ActiveRecord Interface を中心に紹介する予定
  - その他機会があれば発表し、フィードバックを得て開発を継続する