# R Markdownクローン「rubydown」の作成と応用

Rubyアソシエーション 開発助成金2018 最終報告書

## プロジェクトの概要(応募内容の再掲)

近年「ノートブック」と呼ばれるプログラムコードとドキュメントを一ファイル中に混在させたファイル形式が主にデータサイエンスのワークフローの共有で重宝されている。このノートブックの実装の主なもの2つに「Jupyter Notebook」と「R Markdown」がある。Jupyter NotebookにはRuby言語サポートを追加する機能拡張が存在するがそのプログラム構成はR Markdownと比較して複雑でありメンテナンスが容易ではなかった。このプロジェクトではRuby言語によるR Markdownクローン「rubydown」のgemを作成すると共にその応用例を示す。

## 応募者名(応募内容の再掲)

- 西田孝三
- 西山和広

## プロジェクトのゴール

Ruby言語をコードチャンクに記述したMarkdownを実行し、その結果を含むHTMLを生成するgem (rubydownと命名) の”作成” と”応用”が本プロジェクトで達成すべきこととなる。
“作成”の段では、「生成するHTML中で、コード実行結果に応じたテーブル・静的グラフ画像の埋め込み・JavaScriptを用いた可視化を実現する」ことがrubydownに必ず求められる機能となる。
“応用”の段には、初歩的な応用(rubydownの実用例)と発展的な応用(エコシステムとしての改善)がある。初歩的な応用の具体例としては「再現可能なデータ可視化・機械学習ワークフロー例の提供」がある。また発展的な応用の具体例としては「rubydownと静的サイトジェネレーターJekyllの連携を行うgemの提供」「rubydownとJupyter Notebookの相互運用性の確保」「プログラム向けエディターとrubydownの連携」がある。

# プロジェクトの成果

## ソースコードや成果物の公開URL

https://github.com/sciruby-jp/rubydown
https://rubygems.org/gems/rubydown

## プロジェクトのロードマップ(応募内容の再掲)

1. jekyll-rubydown-converter の実装
2. rubydown gem の実装
3. rubydown gem の改良 (HTML以外の出力形式のサポートやJupyter Notebookとの互換性の確保など)
4. rubydown をサポートするVisual Studio Code拡張機能の実装 (オプショナル)

## ロードマップ達成状況

1は順序変更により後回しにした。また未達成である。応募時にはjekyll-rubydown-converterはrubydownと全く別のgemとして実装する予定だった。またjekyll-rubydown-converterの方がrubydownの実装より容易と踏み、順を先に設定していた。しかし実際にはrubydownの実装が容易であったため、jekyll-rubydown-converterはrubydownを完成させた後の応用として順を後に回すこととなった。
2は達成。「プロジェクトのゴール」節で挙げた必須機能を満たすgemを https://rubygems.org/gems/rubydown で公開。rubydownの実用例(入力のMarkdownと出力のHTML) は https://github.com/sciruby-jp/rubydown/tree/master/examples で公開。
3はまだ取り組んでいる最中であり未達成。特にJupyter Notebookとの互換性において新たなアイデア(後述)が生まれたため時間を要している。
4も未達成。

## 達成事項と未達成事項のまとめ

### 達成

- 「ノートブック」に必須な機能のrubydownへの実装
- rubydownのrubygems.orgでの公開
- rubydownの実用例の充実化

### 未達成

- HTML以外のファイルフォーマットへの出力
- エコシステムの一部としてのrubydownの応用

- rubydownとJupyter Notebookの互換性の確保
- rubydownとJekyllの連携
- プログラム向けエディターとの連携

# プロジェクト遂行中に気づいたこと

プロジェクトを遂行してみると新たに優先すべきタスクが生まれた。その背景として重要なものを挙げる。

## Ruby言語の特徴を活かした独自プログラムデザイン

応募当初は「R Markdownのベースとなるknitrを模倣したAPIを持つようにrubydownを実装する」と宣言していた。しかしプロジェクト遂行中にRよりRubyに言語的に近いPythonにおいてR Markdownクローン(knitpy, stitch)が存在することを知り、knitrを模倣することは効率的でないと判断した。さらにknitpy, stitchと比較しても「Rubyのオープンクラス機能の活用」や「kramdown(pandocに依存しないMarkdownコンバーター) gemの活用」といった独自デザインが本プロジェクトに最適と判断したため、プログラムデザインは他のどの同目的プロジェクトとも異なるものとなった。

## 単体テストよりも結合テストや実用例(ワークフロー)を増やすことの必要性

プロジェクト遂行中にscikit-learnに似たインターフェースを持つ機械学習gem https://rubygems.org/gems/rumale が公開された。rumaleの出現によりRubyにおけるデータサイエンスのワークフローで機能的に不足する要素が減り、現状でも様々な実例を共有することが可能となった。そのためrubydownの単体テストに注力するよりも「rumale などを用いたワークフローの実例を増やし、rubydownで実行した際に生じた問題から解決していく」という開発方針が効率的であると考えた。
(rubydownのソースコードにテストが無くexamples下に実例を多く追加しているのはこのため。)

## エコシステムの一部であることを強く意識することの重要性

プロジェクト開始前はrubydownをJupyter NotebookのRuby対応版であるIRuby Notebookの代替物として提案することを考えていた。またエコシステムの一部としてのrubydownの役割を意識しておらず、rubydownとIRuby Notebookの互換性に関する考えが浅かった。プロジェクトを進めるにあたり「ipynb(Jupyter Notebookのデータファイル)をrubydownの入力としても同じ結果を得ることを可能とする」rubydownの機能実装案が生まれており実現に時間を要している。
(単なるファイルフォーマット変換だけではなく、ipynbに対する細かな処理をrubydown内でJupyter Notebook用のツールを用い行う必要があるため。)
この互換性が確保できれば charty のような IRuby Notebookでの利用を前提としたgemの実用例 ( https://github.com/red-data-tools/charty/tree/master/examples )は変更を加えることなく

rubydownで実行可能となる。またこれにより目的に応じたRubyのノートブック実行系の提供が可能となる。(REPL的インタラクティブ性が必要な用途ではIRuby Notebookを、プログラム向けエディターでの本格的編集が必要な用途ではrubydownを、といったように。)

# 今後の計画

- Jupyter Notebookをrubydownの入力として受け入れるように機能を追加
- さらなる実用例(ワークフロー)の追加
- プログラム向けエディターとrubydownの連携の実現
- Jekyllのconverterとしてのrubydownの応用

# 謝辞

rubydownの元となる https://github.com/genya0407/rbmark をデザイン・作成してくださった https://github.com/genya0407/ さんに感謝する。またrubydownで活用している下記gemとその作者の皆様に感謝する。

- https://rubygems.org/gems/kramdown
- https://rubygems.org/gems/numo-gnuplot
- https://rubygems.org/gems/rbplotly

加えてエコシステムの一部としてのrubydownの開発の方針についてアドバイスしてくださったメンター村田さんに感謝する。