Project title: Turning R objects into Pandoc's markdown

Project short title (30 characters): Pander

URL of project idea page: http://rwiki.sciviews.org/doku.php?id=developers:projects:gsoc2014:pander

Bio of Student

I'm a Fulbright Graduate student from Ukraine, now pursuing master's degree in Purdue university, USA. I'm interested in programming languages implementation, now working on fastR project (https://github.com/allr) with Purdue university and Oracle, which is focused on implementing R on top of Graal compiler and Truffle framework. More specifically, now I'm working on creating a concise test suite for build in C functions in GNU R (https://github.com/allr/testr), which will in future help testing new R implementations. I've used R for my Data Mining and Machine learning course and find it very useful for statistical computations. During my undergrad we used markdown for reporting for some assignments so I'm familiar with syntax and usage of it.

I think my background qualifies for this project because I have good background working with R (both internals and scripts), I have good understanding of it's semantics, limitations and best usages. Also I've used markdown for reporting before, and feel that better reporting directly to markdown from R is needs improvement.

CONTACT INFORMATION

Student name: Roman Tsegelskyi

Link_id: romantsegelskyi

Student affiliation

Institution: Purdue University

Program: Master of Science (MS)

Stage of completion: graduation in May 2015

Schedule Conflicts:

My friend has a wedding on 22nd of May, so I will not be available to work May 21st - 25th. Probably will attend useR conference in LA, June 30 - July 3rd.

MENTORS

Mentor names: Gergely Daróczi, László Szakács

Mentor link_ids: daroczig, cocinerox

Have you been in touch with the mentors? When and how?

I've contacted Gergely after I've implemented test task to ask for review, some questions about Pander and procedural questions about GSoC. Also we discussed feasibility of the ideas from my proposal, different implementation details and received feedback on my ideas.

CODING PLAN & METHODS

During this GSoC session I have want to achieve such objectives:

1. Objective: create new pander methods for not yet supported R classes

List of classes to implement markdown rendering for:

- Time Series (timeSeries package)
- xtabs (stats package)
- coxph (survival package) - supported in xtable
- zoo series (zoo package) - supported in xtable
- linear mixed models (lme, glmer) (nlme package)
- elrm (exact logistic regression) (elrm package)
- formula class (stats package)
- CrossTable (descr package)

Currently those classes are not supported by Pander. For most of them default methods does not produce any reasonable output, so I think it would be a good addition to the list of supported classes. Some of things listed are supported in other rendering packages like stargazer and xtable.

The implementation for each of the class is kind of similar. Pander utilizes S3 method for rendering markdown for classes, so extending Pander to support will require implemented methods print.*class** for each of the classes. This will require some time to get complete understanding the structure of classes, to extract information properly.

Expected obstacle is that some classes really require deeper understanding of it's internals and extra treatment, so they might take more time that expected.

This is a list of classes that I want to implement for sure, also if time permits more classes will be added (packages like xtable, stargazer and sjPlot can be taken as a basis for rendering examples).

For this objective successful implementation can be easily evaluated in different way. First of all, just by rendering those classes. Also for the classes that are supported by other rendering packages, end result can be compared (for example Time Series is supported by xtable to translate to LaTeX, so result achieved by Pander translation to Markdown should be somewhat similar. Also using pandoc, we can compare result exactly). And another method for evaluating success is to use those new classes inside Pandoc.brew function to render text mixed with R data.

2. Objective: extend Pandoc.table to support configurable width.

Currently pander does support configurable column width, but with huge limitations. Right now only max width for all columns can be set as an option and it does not reflect data without white spaces (for example cell with number like 123456789, will not be broken in multiple lines even if split option is smaller then number length). Also, it is a global option for every column, which is not always convenient.

So I propose extending configurable with option in 3 ways:

1. Allow to specify max width of the column separately with a vector.
2. Break lines in cells, even in case when there are no white spaces. For example with '-' - break word configuration into confi - guration.
3. Allow to specify relative width of columns (like 50%, 25%, 25%) and integrate that with exactly configurable max width.

This will require changing pandoc.table method and addition of new global options (relative/exact, vector/value) and possibly some optimization/refactoring of pandoc.table methods on the way. Crucial thing is no to change any existing behavior, because most of methods for R objects are based on pandoc.table

3. Objective: refactor existing code base in particular brew function, improve performance of pandoc.table, extend existing test suite.

Current implemetation of Pandoc.brew function is a fork form a brew package. It needs optimization and refactoring to better reflect Pander package in general. So one of the main goals for this GSoC session would be to refactor brew function and other code in Pander (if time permits). Besides that, I want to work on boosting performance of pandoc.table function with C/C++ implementation for helper functions. Now rendering takes some time for big tables, so rewriting some parts of pandoc.table in C/C++ might be helpful to improve performance. Also I would like to extend existing suite to reflect supported classes for rendering and more brew function functionality.

For refactoring, it is easy to evaluate success if it does not break anything and if code becomes easier to understand/maintain, while not loosing much of it's performance characteristics. Despite that results achieved can always be compared to "canonical" brew package renderings, because end result should be somewhat similar and can be compared by diff tool for example.

TIMELINE

May 19 - June 11 (including absence May 21 - May 25) - Objective 1. Implement S3 methods for proposed list of R objects.

June 12 - June 15 - Objective 1 documentation and testing. Extend existing test suite with test cases for new classes.

June 16 - July 9 (including possible absence June 30 - July 3) - Objective 2. Implement options for pandoc.table to support configurable width.

June 27 - Midterm evaluation. By midterm evaluation Objective 1 should be completely finished and concrete implementation ideas should be ready. Possible implementation of the one of 3 things listed in Objective 2.

July 9 - July 14 - Objective 2 documenting and testing. Extend existing test suite with test cases for configurable width options for pandoc.table.

July 14 - August 9 - Objective 3. Refactor existing code base with focus of pander.brew and other functions as time permits, rewrite some parts of of pandoc.table in C/C++.

August 9 - August 12 - Objective 3 documentation and testing. Extend existing test suite with more test case for pander.brew in particular.

August 12 - August 18 - general testing and documentation. Preparation for evaluation.

What is your contingency plan for things not going to schedule?

Contingency plan - each of 3 objectives has minimum goals that I want to achieve, but those goals consist of sub-goals that can very depending on how works got. For example for objective 1, I have a draft of objects that I want to add support for in Pander, so depending on complexity of each particular class, list can be extended with more objects or shrunk down to most important one. Same for objective 2 - if time something goes wrong, list of extending configurable width can be decreased to one of the options. As for objective 3, there is really no limit in refactoring current code base and providing test cases for it.

MANAGEMENT OF CODING PROJECT

How do you propose to ensure code is submitted / tested?

There are couple of ways to ensure that code is tested. First of all, Pander has test suite that tests some basic features of the package and can be used for regressing tests, because it is concise and quick to execute. Secondly, using pandoc and other rendering packages like xtable, stargazer it is possible to generate text/PDF/rtf output and compare files either using diff tool of manually.

How often do you plan to commit? What changes in commit behavior would indicate a problem?

While implementing methods for objects listed in objective 1, I plan to commit every time a method for particular object is implemented and tested (preferably daily or once in 2 days). During work on second objective I plan to commit roughly every time one of the listed options is implemented, hopefully couple of times a week. And when working on third objective I plan to commit everyday refactored parts of Pander package.

Because I would be working on rendering R objects in markdown any problems are easy to spot - if some rendering goes wrong, it is easy to see. Also existing test suite will be helpful in catching bugs in my code.

TEST

Test task was to complete to extend pander to support CrossTable class from descr package.

CrossTable class has specific structure, so basically task was to extract this information in a table (adding summary) and then supplying this to a pandoc.table function which produces markdown rendering for tables.
My implementation lucks an idea which multiline cells (it creates redundant rows), and I plan to improve it as part of the my coding during GSoC session. I also discussed implementation with project mentor, and we agreed that it is acceptable for now.

Exact implementation can be found in my pull request - https://github.com/Rapporter/pander/pull/62 (posting whole code here seemed a little bit redundant).