FireData Connecting R to Google Firebase

Project Title: FireData: Connecting R to Google Firebase

short title: FireData: A Firebase API

Background

Persistent Data Storage and transfer with R can be complicated and time consuming. While there are great packages to connect R to DBs like mySQL/sqlLite/postgreSQL those databases itself need to be hosted and maintained. To implement a DB based login system in Shiny now probably takes 2 weeks of work to get it done in a secure way, and a IoT project for data collection requires a 24/7 running server.

The Google team took that common problem (not only of R) and created Firebase. Firebase is a online database that is not only intuitive to integrate into Apps & Web applications, but can also be used to build user registration/login systems with persistent data storage (thinking about shiny) or to transfer data between IoT devices via Firebase directly to a R environment. Writing a intuitive R to Firebase connector enables R to be used in new "cross platform" realtime applications. Thats why we strongly believe the "FireData" package will improve the R open source environment.

Related work

The R universe offers with the `httr` package to apply PUT/POST commands to servers (including Firebase). In combination with a JSON library like `jsonlite` it is possible to send & retrieve data from Firebase.

Google Firebase on the contrary is more sophisticated than simple PUT/POST requests and allows JWT tokenizations, registrations & logins and even file uploads. Firebase brings all functionality and ease to be the to-go Database for R IoT or Shiny related projects.

Looking at other API implementations, R seems to be the only major language (C++ & JS are supported natively) that does not yet has a Firebase API implementation (here the official Google Firebase API list):

Choose a Helper Library

You can read and write data through the REST API directly from the command line. In this guide we'll use cURL for all of our examples. We also have several third-party helper libraries for interacting with the REST API from your favorite language. You can use one of the following helper libraries or create your own:

Language	Libraries
Clojure	talka by Cloudfuji
Dart	IO Client in the official firebase-dart library
Go	Firego by Steven Berlanga and Tim Gossett
Go	Go Firebase by Cosmin Nicolaescu and Justin Tulloss
Java	firebase4j by Brandon Gresham
Perl	Firebase-Perl by Kiran Kumar and JT Smith
PHP	firebase-php by Tamas Kalman
Python	Pyrebase by James Childs-Maidment python-firebase by Özgür Vatansever python-firebase by Michael Huynh
Ruby	firebase-ruby by Oscar Del Ben BigBertha by Fernand Galiana rest-firebase by Codementor

Details of your coding project

With the experience of GSOC 2016 we will start with a larger team (other open source developers signaled it would help to improve their own DB experience working with R and Shiny) to get the FireData package CRAN

ready before the September deadline.

- Now - April : Adding prove of concepts of the Firebase integration into Shiny and IoT applications.
- April - June : Writing and implementing (including unit test) the various firebase connecting functions
- June - July : Adding the s4 classes/functions to the FireData package & writing Vignettes/Documentation
- July - August : Finishing up the FireData package with focus on OS compatibility testing and identifiying and possible fixing weakpoints of the package.

Expected impact

R is one of the strongest players in data science. The aim is to connect its strength in data analysis with the actual data. By making it easier to directly connect R to the inputs of IoT devices or other applications such as websites/shiny it will open new possibilities for open source developers around the world.

The final product, the FireData package, will connect two great environments with each other. When we succeed to make it stable & intuitive alot of projects will be build ontop of it.

Management of Coding Project

Daily submits to the github repo. The project will be started in a test-driven fashion by writing "testhat" tests for every method implemented. For the Firebase testing we will accordingly create a test database (firebase allows to upload xml database mirrors which correspond to real db environments).

Risks/Challenges & opportunities:

Implementing the 3rd party Firebase API can always bear the challenge of sudden new features or changes to its functionality. Due to the large userbase of firebase and the versioning we are pretty sure to complete the task staying at the same firebase version (firebase 2/firebase3 had breaking changes). One opportunity is that a recent change added firebase function calls which are especially interesting for our project as it allows firebase to trigger on certain data events to send data to R applications.

Required Skills

Knowledge of noSQL databases (Google Firebase is a noSQL database), experience with package development (roxygen2) including unit testing (testhat) and S4 Classes. In addition a general understanding of APIs & server architecture is required.

Proof of concept / Test for the student:

- A tutorial on Shiny & Google Firebase Logins written by the student https://r.codes/ItemPreview/-KaLAwn-zqMgARFUnxpo
- proof that the student can create R packages with roxygen2 and devtools: https://github.com/Kohze/Discovr
- proof that the student can create unit tests (testhat package) https://github.com/Kohze/misc/tree/master/unitTesting

Mentors

Samuel Schmidt, Ph.D . (has GSOC Mentor experience). | samuelschmidt218@gmail.com

Lindolfo Pedraza, MA. (Database and API specialist) | lindolfo.pedraza@gmail.com

Potential Student

Robin Kohze | pi2robin@gmail.com

https://github.com/Kohze | Project leader of https://R.Codes | GSOC student of 2016 with the discovr package