STATS 290: Paradigms for Computing with Data, winter 2013-14

PROJECT: hschnegg.quantified.self

Herve SCHNEGG, ID# 05849557

March 19th, 2014

1. Introduction

Our initial ambition was to build the software required to centralise the data generated by various wearable devices so that it can easily be analysed in R.

Our plan was for Herve Schnegg to focus on the data management side of the project and for Naxin Wang to develop some visualisation and analysis tools.

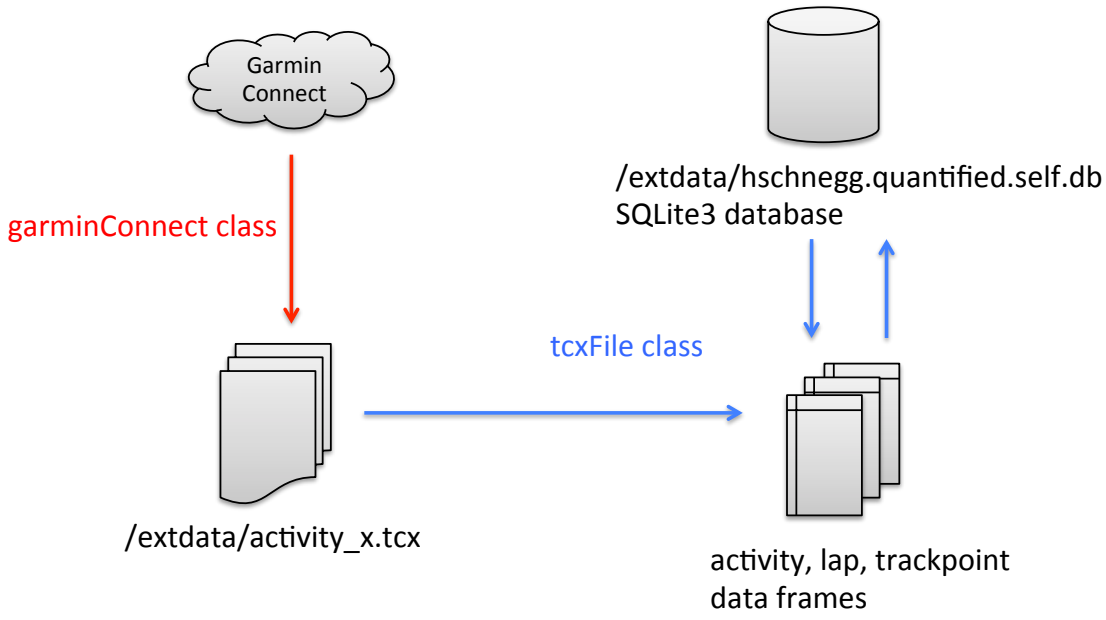
This plan was altered after we realised that most of the complexity of the project would consist of writing the interfaces with the devices' APIs. Some of the devices' vendors do not even document their APIs and want users to only use proprietary software to analyse the data.

Starting from there, we decided that Nancy Wang would focus on Jawbone Up and that Herve Schnegg would focus on Garmin.

hschnegg.quantified.self R package deals with accessing the data produced by Garmin sport and fitness devices (http://www.garmin.com/en-US/explore/intosports/).

Even with that change of scope, we believe that the software produced here is quite useful for people interested in fitness data and having some knowledge of R. It builds a bridge between the closed world of Garmin and the open world of R.

2. High Level Architecture



All the Garmin sport and fitness devices synchronize their data with the Garmin Connect website (http://connect.garmin.com). A test account has been setup for the project:

Username: stats290_test Password: stats290_test

The basic unit of information stored in Garmin Connect is an activity. An activity is composed of activity, lap, track and trackpoint data. Traditional GPS data and specific fitness data are stored at each of those levels.

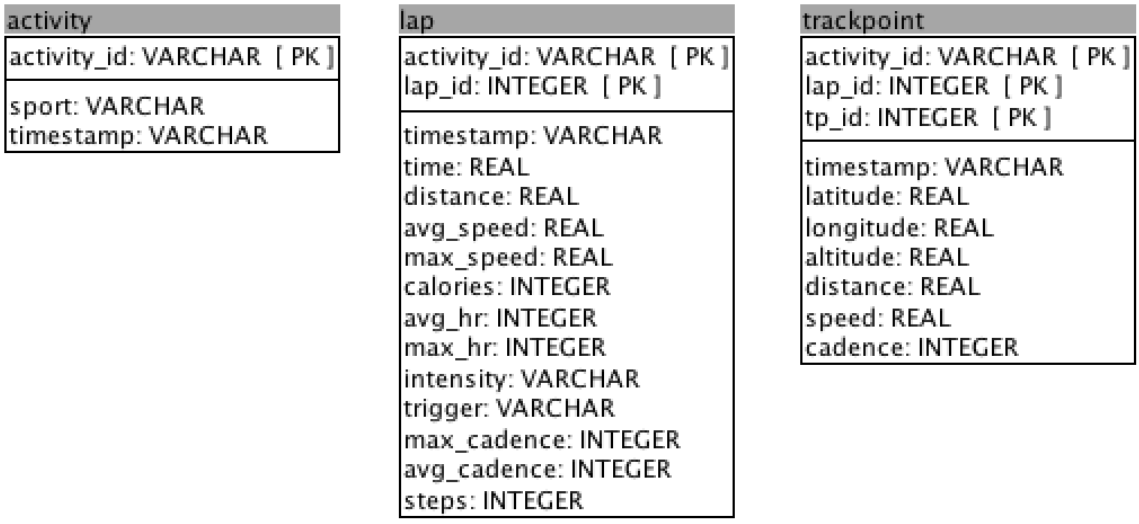
The package contains a reference class `garminConnect` offering the tools required to login to Garmin Connect, check the available activities, and download an activity.

The files exported from Garmin Connect are XML files with a .tcx extension (http://en.wikipedia.org/wiki/Training_Center_XML). A .tcx file contains more data than a generic GPS data file as it also stores fitness data like heart rate, running cadence and calories.

The reference class `tcxFile` enables the management of those files. It allows to read and parse a .tcx file, to load the content of a file into R data frames, and to manage the storage and retrieval of the data inside an internal SQLite database.

Activities are stored into three data frames: activity, lap and trackpoint. Each of those stores the activity at different level of aggregation.

The SQLite structure also contains three tables that allow the persistence of the data. The database is stored in /extdata/hschnegg.quantified.self.db



The code of the package is available in GitHub:
https://github.com/hschnegg/hschnegg.quantified.self

3. Package Components

Note that the package documentation has further details on some of the components.

3.1. garminConnect class

Reference class containing the tools required to access data stored in the Garmin Connect website.

username field: Used to store the Garmin Connect username

password field: Used to store the Garmin Connect password

login method: This method uses RCurl to pass the username and password to the website and performs all the authentication stages required by the website. The method returns a curl handle.

retrieveActivityList method: This method calls the login method and then queries the website for a list of the most recent activities available. The result is returned as a data frame.

downloadTcx method: This methods expects an activity id as a parameter. It calls the login method and then calls the website API to download a tcxFile. The file is stored in /extdata.

3.2. tcxFile class

Reference class used to process, store and retrieve .tcx files.

activity field: A data frame used to store activity details

lap field: A data frame used to store lap details

trackpoint field: A data frame used to store trackpoint details

read method: Expects an activity id or full .tcx filename as a parameter. The method parses the file and stores the data retrieved in the three data frames (fields) activity, lap and trackpoint.

saveToDb method: Insert the content of the three data frames (fields) activity, lap and trackpoint in the package database.

readFromDb method: Used to read the data stored in the package database. It has two parameters. One controls which activity to retrieve (everything if missing), the other controls which of the three data frames (fields) to load (the three of them if missing).

3.3. userInterface.R

Contains a few helper functions for the user.

mapActivity function: Retrieves an activity from the database and plots its trackpoints on a Google map. Expects activity id as a parameter.

listLoadedActivity function: Lists all the activity already loaded in the package database.

listGarminConnectActivity function: Lists activities stored in Garmin Connect.

listNewActivity function: Lists activities stored in Garmin Connect that are not yet stored in the local database.

3.4. constants.R

Defines functions returning constants used throughout the package (Garmin Connect URLs, package name, database file name).

3.5. extdata folder

Downloaded .tcx files and internal package database (hschnegg.quantified.self.db) are stored there.

3.6. sql folder

createTables.sql script: Script used to recreate the internal database tables.

4. Usage and Examples

4.1. Test data

A few activities are available in a test Garmin Connect account created for the project: http://connect.garmin.com/en-US/signin ; Username: stats290_test ; Password: stats290_test .

Two activities are loaded in the package database: 433485172, 439915402.

A .tcx file is available in the extdata folder: activity_439915418.tcx

4.2. Example script

doc/examples/examples.R runs the user through the main features of the package.

5. Not Covered

Further development of the package should address the following points:

- Add units conversion functions
- Add support for more devices
- Add forms for manual data entry
- Add analysis and reporting tools
- Add a user interface