**Rstud: a package for studbooks in R**

*Package Goal*

Capacity to manipulate, extract, edit, analyze, present, and save studbook data in a variety of formats

*General Background*

Rstud is a package for reading, using (graphing and analyzing), editing, and saving data held in studbook-style databases in R. R is a free language and environment for statistical computing and graphics based off of S, Bell Labs’ in house statistical and graphics software. The main benefits of R are that it is free (*gratis*, costs nothing so you don’t have to pay to use it)and that it is free (*libre*, open so that the world can collaborate). As a result of collaboration, and in particular the package system, a single user can easily implement a vast array of code that has been written, optimized, and vetted by other individuals.

For the purposes of studbook analyses, this means things like kinship and inbreeding coefficients can be calculated from a single line, and so too can complicated contemporary statistical approaches, such as the animal model (pedigree-based analyses of trait values). By building an interface for studbooks in R, we can then take advantage of all of the amazing code that folks have already written to do whatever kinds of analyses we want.

General documentation on writing R packages can be found [here](http://cran.r-project.org/doc/manuals/R-exts.pdf) and the current repository of the over 6500 (and counting) officially approved packages can be found on the CRAN (Comprehensive R Archive Network) [here](http://cran.r-project.org/web/packages/). There are also packages available that have not been officially approved by the CRAN, such as some of the [Bioconductor](http://www.bioconductor.org/) packages like GeneticsPed.

*History, Current, and Future*

Initially, Rstud began as a project to analyze inbreeding and its effects on zoo populations using existing studbooks. This obviously required importing data from studbooks into R, calculating basic genetic metrics, conducting statistical analyses, and producing output. Once we had this initial set of code up and running, I realized that having studbook data in R expanded the potential uses of the data beyond what was easy/doable in existing software (SPARKS, PopLink, PMx) by either taking advantage of the existing code/packages that others have written or by writing additional routines.

For example, collaborations with the PMC and specific programs have fostered the development of code for interfacing studbook data with existing routines for running the animal model (to evaluate quantitative trait genetics), for batch editing and re-formatting of studbook data (e.g. splitting mnemonics), and robustly analyzing interactions between management and biology (e.g. examining the impact of transfer events on longevity/mortality). With a small amount of coding, one can conduct a whole suite of analyses on studbook data. However, many current users of studbooks might not be comfortable with excessive (or much at all) command-line interaction. Therefore, a main goal of this project has been and continues to be developing a framework for interaction with existing functions, which will greatly minimize the coding required of any user.

Rstud currently exists as a simple script of function code that is being used by the PVA team for inbreeding analyses conducted associated with their reports. Because this began as a one-off project with a specific goal (inbreeding analyses), I kept the initial coding relatively simple (see below) and did not employee S4 object orientation. In beginning to plan the expansion of Rstud into a more legitimate, broad package, I am evaluating the need for a S4 approach (which is more rigid but has great built-in validation capacity for structures/classes of objects).

I have set up a repository on GitHub for version control, and would be happy to share the link with anyone who is interested in collaborating on this project. I would also greatly appreciate ideas, suggestions, etc. on particular analyses, summaries, graphics, or types of edits that folks are finding time-consuming or difficult (or impossible) to conduct in existing software. I’m developing a wish list that I will begin tackling once the basic structure is in place.

*Current Status*

Currently, the Rstud code is not in full package format such that it could be submitted to CRAN, but it is in a precursor, simpler format. As of now, there are three main files: Rstud\_Functions.R, Rstud\_Example.R, and Rstud\_package\_documentation.docx. I also have code living in a handful of other scripts that I will be importing into these files, as well (it needs to be formatted in).

Functionality:

Read in, interact with, edit, save, utilize SQL studbooks (i.e. PopLink)

Apply overlays

Measure basic genetic metrics (kinship, inbreeding) and visualize pedigree

Simple analyses of inbreeding impacts on demographic traits

(infant) mortality, brood size, birth sex ratio

Editing (simple and batch) and saving of studbooks

*Future Plans*

Obviously, studbooks come in a variety of formats, so I need to generalize the input/interface and architecture beyond SQL. I’ve been working on a very differently formatted “studbook” in the USFWS’ red wolf wild database and have been working to figure out how to generalize the input routines very, very broadly. The shift to S4 hopefully should help with this.

Specific goals:

Create a “studbook” class with the flexibility to handle a variety of input types/completeness

Increase capacity for different studbook formats for I/O

Develop a “report” structure and set of functions with relatively simple interfaces

Data checking/validation routines

Make a better set of examples with how-to and screen shots, etc.

Future possibilities:

Build interface for PVA capacity, for example a wrapper for VORTEX code

Incorporate PMCTrack data

Things I can use help with/need to figure out

Other formats of studbook data that folks interact with (example files in particular would be great)