mult: a Multivariate R Package with a Dynamic Java Frontend

E. James Harner[1*] and Dajie Luo[1]

1. Dept. of Statistics, West Virginia University, USA

* Contact author: jharner@stat.wvu.edu

Keywords: Multivariate R package, Java frontend, Dynamic graphics

`mult` is yet another R package for doing multivariate analyses. However, `mult` has two unique features which greatly enhance its usefulness. First, `mult` is complete in the sense that it includes canonical and partial analyses. Second, these analyses can be visualized using high-dimensional dynamic graphics.

As an R package, `mult` uses S3 classes, and a function returns the output as an object whose class corresponding to the function call, e.g., the `pca` function returns an object of class `pca`. The analyses are quite general. For example, principal component analysis not only allows for robust versions, but also for partial, canonical, and partial canonical variants. Factor analysis, multiple correlation analysis, canonical correlation analysis, and discriminant analysis also have flexible implementations. Standard R graphics, e.g., biplots, are supported.

Multivariate space is most naturally explored by interactive and dynamic graphics. JavaStat (Harner and Luo, 2009) was designed to support highly interactive data analyses and dynamic graphics in a platform-independent way. The architecture of JavaStat is described elsewhere, e.g, Harner, Luo, and Tan (2008a, 2008b). JavaStat also acts as a front-end to R, but the use of R as a compute engine is transparent to the user and is optional. JavaStat has a client/server architecture to allow for high-performance computing, but the server can reside on the same machine as the client.

JavaStat has an interface to the `mult` package. Basically, certain menu selections in JavaStat trigger a series of actions. First, a Java object corresponding to the selected variables in a data table is sent from the client to the server using RMI. On the server side, the object is taken apart into a bunch of arrays, e.g., a double array represents a numeric column in the original table. Then the JRI API is used to convert these arrays into vectors in R. These vectors are created in the current workspace of R and are assigned names. Optionally a data frame can be created. After preparing data in the workspace, a modeling command, e.g., `pca`, can be issued to R through a JRI evaluation method. Since the returned object exists in the current workspace, subsequent commands are sent to extract useful results from this object. Those results are then converted into Java objects and assembled into a more complex Java object, which is sent back to the client as a whole.

At this point, the user can generate dynamic plots, e.g., a constrained grand tour or a dynamic high-dimensional biplot, or do subsequent analyses. The analyses and plots based on the returned values from R can be supplemented by the built-in functionalities of JavaStat, e.g., a dynamic parallel coordinate plot.

References

Harner, E. James and Luo, Dajie (2009). JavaStat Home, `http://javastat.stat.wvu.edu`

Harner, E. James, Luo, Dajie, and Tan, Jun (2008a). JavaStat: a Java/R-based statistical computing environment. *Computational Statistics* , Online SpringerLink version available in 2008.

Harner, E. James, Luo, Dajie, and Tan, Jun (2008b). JavaStat: a Java-based R Front-end, User!2008 (Dortmund, Germany), July 2008.