Oncoscape

Paul Shannon

April 30, 2014

1 Introduction and Basic Operations

Oncoscape (though the name will soon change) is a hybrid R and Javascript web browser application, offering three features:

- Interactive graphics in a web browser, for "lay" users to explore and analyze clinical and molecualr cancer data
- Analyses and data provided by R
- A simple well-separated software architecture (an R server, HTML/CSS/Javascript running in the browser) encouraging fast and agile development.

Taking these in turn:

- Interactive graphics are built upon Javascript, jQuery, d3, and cytoscape.js. (links needed).
- The data-handling and analysis capabilities are provided by R code and libraries.
- The software architecture separates these two rich and capable environments, connecting them only by passing simple JSON messages back and forth. HTML/Javascript/CSS sprawl is tamed by identifying separable"components" and by assembling html at build-time via the unix tools "make" and "m4".

The JSON messages are at present passed over websockets; alternative mechanisms could be used. The messags five fields:

- cmd: "fetchClinicalData", "calculatePCA"
- status: "request", "response", "error"
- source: identifies the message sender
- target: indentifies the target
- payload: arbitrarily complex, message-specific data

2 Other Packages Needed

- websockets: see `https://github.com/rstudio/R-Websockets`
- RJSONIO
- base64enc
- pls
- RUnit
- survival
- coin
- ggplot2

For all packages except websockets (which must be obtained from githbub, and built from source), this command will download and install the necessary packages:

```
> source("http://bioconductor.org/biocLite.R");
> biocLite(c("RJSONIO", "base64enc", "pls", "RUnit", "survival", "coin", "ggplot2",))
```

3 Softare Organization and Package Structure

The Oncoscape web app is made up of "components" each of which has a user interface (seen in the browser) and operations (sometimes performed in javascript, often carried out by sending messages to R, with results returned). A full-featured Oncoscape assembles a number of these components into a single web page, arranged one component per user interface tab – though other arrangements are possible.

The package has inst/scripts/ subdirectories, one per component, and a few more subdirectories which combine those components into a more-or-less integrated application ("app directories"). This encourages independent development and testing of the components. Each component directory has standard files: widget.html, code.js, index.pre, makefile, run.R. Issuing the 'make' command builds and runs each directory's version of Oncoscape, either a single component (for testing and development) or an app made of multiple components.

The traditional unix development tools "make" and "m4" assemble browser-ready HTML files, one per component or app. All of the HTML, CSS and Javascaript are are built into the R package, installed along with the package, and at run time, served up to the user's browser by a small HTTP server in the package (and provided by the websockets R package).

4 Running Oncoscape

In order to run the current Oncoscape webapp:

- library(Oncoscape)
- startWebApp("tabsApp/index.hmtl", 7781L)

Socket handling needs to be improved. Sockets are usually, but not always, released upon program termination. An on.exit hook should help. An adaptive strategy to find an open socket would be a boon to the user. A utility which surveys current socket claims would be handy.

To build Oncoscape in development mode. For example:

- cd Oncoscape/inst/scripts/gbmPathways
- make

The make command assembles HTML, CSS and Javascript into a single file – 'index.html' – the user interface. It builds and installs the R package, containing clinical and molecular data, and all of the operations requested by the user interface. Finally, make will start R, load the new package, and direct your browser to display 'index.html'. The Javascript and the R websockets server establish a communication channel over the specified port, data is loaded, the web page is rendered, and the application is ready.