The R-Tcl/Tk interface: Potential usage for graphical models

Peter Dalgaard
Department of Biostatistics
Faculty of Health Sciences, University of Copenhagen

gR-2003
Overview

• What is Tcl/Tk?
  – Tool control language + (GUI) Tool kit
  – Jeff Ousterhout, Sun Microsystems
  – BSD license

• The interface to R

• Usage for graphical model specification
Tcl/Tk

```tcl
button .a
pack .a
.a configure -text hello
```

- Brief, command-shell like language
- Object creation commands
- Geometry manager `pack` positions `.a` within parent window
- `Widget command` `.a` with subcommands
Tk widgets

- Labels, buttons, radiobuttons, checkbuttons, menubuttons
- Text, entry fields, listboxes
- Scrollbars, sliders
- Callbacks, event handling
- Graphics canvas
Interface with R

- Easy to set up interpreter and call Tcl from R and vice versa
- Convert Tcl commands to R function

Similarities
  - Similar imperative programming style
  - Similar option structure (–text hello vs. text="hello")

Differences
  - Tcl uses textual variable substitution extensively
  - “Widget commands” with subcommands
Tk widgets as R objects

- In R, widgets are created as object based on parent
- Widget commands are implemented as commands acting on widgets

```r
tt <- tktoplevel()
but <- tkbutton(tt)
tkpack(but)
tkconfigure(but, text="hello")
```
Callbacks

- You can attach an R function to (e.g.) a button press

  ```r
  tkconfigure(but, command=function()
              cat("hello\n"))
  ```

- Or when there are no parameters

  ```r
  tkconfigure(but, command=quote(cat("hello\n")))
  ```
Callbacks with arguments

• Tcl uses %-substitutions
  
  ```tcl
  bind .t <Button-1> {puts %x %y}
  ```

• In R, we use arguments to function:
  
  ```r
  tkbind(t,"<Button-1>", function(x,y)
          cat(x,y,"\n")
  ```

• Notice the use of the event specifier "<Button-1>" to bind a function to a left-click of the mouse.
Tcl objects

- The original Tcl credo was “Everything is a string”
- This leads to “quoting hell” when you need to pass special characters into Tcl
- The R interface used to create Tcl commands as text strings, which was sometimes very slow.
- The modified Tcl credo: “Everything can be converted to a string”
- Dual ported Tcl objects can be created at the C level and used to construct commands as a vector of objects
- This is what R does as of 1.8.0
The Tk canvas

- Basic objects (lines, rectangles, polygons, ovals, text, …)
- Display list, notion of which objects are above others
- Possibility of binding events to specific objects
- Nice optimized redraw algorithm when objects are changed
- *Tags* allow grouping of items
Example: graphdiddle

- Very simple manual graph manipulation
- Undirected graph, moving nodes around for nicer display
- Data structures, input: vectors X, Y, from, to, Labels
- Auxiliary structure: nodeEdges, list of edges connected to a given point. Redundant, but efficient.
Setup edges

for ( i in seq(along=from) )
{
  f <- from[i]
  t <- to[i]
  e <- tkcreate(canvas, "line",
                X[f], Y[f], X[t], Y[t], width=2)
  nodeEdges[[f]] <- c(nodeEdges[[f]],
                     list(list(to=t, edgeItem=e)))
  nodeEdges[[t]] <- c(nodeEdges[[t]],
                     list(list(to=f, edgeItem=e)))
}
Setup nodes

for ( i in seq(along=x) ) {
  p <- tkcreate(canvas, "oval",
                X[i]-6, Y[i]-6, X[i]+6, Y[i]+6, fill="red")
  l <- tkcreate(canvas, "text",
                X[i]+6, Y[i], text=Labels[i], anchor="nw")
  tag <- paste("node", i, sep=" ")
  tkaddtag(canvas, tag, "withtag", p)
  tkaddtag(canvas, tag, "withtag", l)
  nodeItem[i] <- tag
  tkitembind(canvas, p, "<B1-Motion>", moveNode(i))
}
Callback to move nodes about

NB: Function returns function with i in lexical scope, so that node “knows” its own number

moveNode <- function(i){
    force(i)
    function(x,y){
        x <- as.numeric(x) ; y <- as.numeric(y)
        for ( e in nodeEdges[[i]] )
            tkcoords(canvas,e$edgeItem,x,y,X[e$to],Y[e$to])
        tkmove(canvas, nodeItem[i], x-X[i],y-Y[i])
        X[i] <<- x ; Y[i] <<- y
    }
}
Discussion

• Everyone hates Tcl and/or Tk for some reason…
Discussion

• Everyone hates Tcl and/or Tk for some reason…

• The Tcl language is clunky (but you don’t have to see much of it)

• Tk looks old (Motif style) esp. on Windows (but changes are being planned)

• Lacks functionality (but extension packages exist)

• It’s slow (but not so bad)
Discussion

• Everyone hates Tcl and/or Tk for some reason…

• The Tcl language is clunky (but you don’t have to see much of it)

• Tk looks old (Motif style) esp. on Windows (but changes are being planned)

• Lacks functionality (but extension packages exist)

• It’s slow (but not so bad)

• Just about the only thing that works cross-platform (now!).

• Very easy to get started with (getting it done vs. getting it right)