Programming with R
Day 5
R-GUIs
tcltk(2) package
University of Potsdam

Detlef Groth

Last Lecture

- install.packages
- library
- require
- writing packages
- devtools/roxygen2
- minimal approach

Outline

Day 1 (basics)

- setup, install
  editor, simple programs
- variables, operators
- data structures, control flow

Day 2 (basics)

- functions
- file input/output
- terminal interaction
- command line arguments

Day 3 (advanced)

- object oriented progr.
- code documentation
- R base graphics system

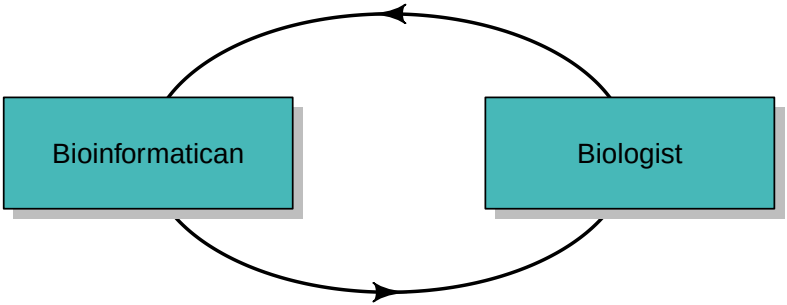
Day 4 (advanced)

- using packagesackages
- package documentation

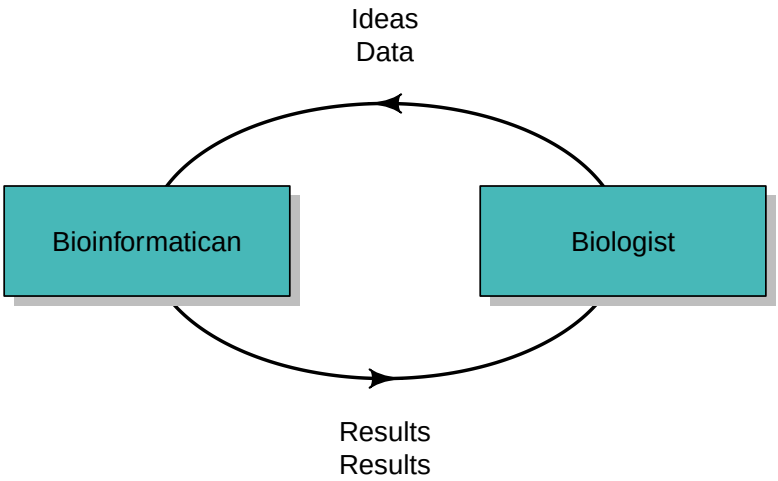
Day 5 (advanced)

- **graphical user interfaces**
- **tcltk** (shiny)

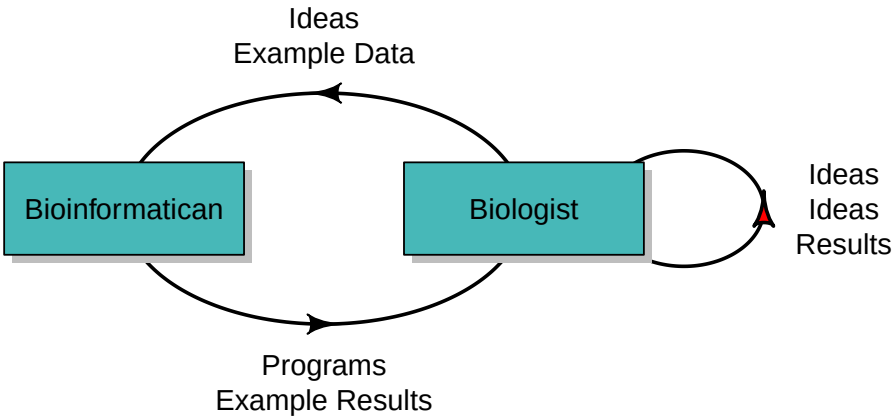
1 R-GUIs with tcltk



Standard (?)



The Better Way



Principle

- almost no exchange of data
- almost only exchange of programs
- Pros:
  - you save time
  - you stick in the project
  - you can learn something new
- Cons:
  - you need to learn something new(?)

Bioconductors Target Search Package

BMC Bioinformatics

BioMed Central

Software

Open Access

TargetSearch - a Bioconductor package for the efficient preprocessing of GC-MS metabolite profiling data

Álvaro Cuadros-Inostroza*[1,2], Camila Caldana[1], Henning Redestig[1,3], Miyako Kusano[3], Jan Lisec[1], Hugo Peña-Cortés[2], Lothar Willmitzer[1] and Matthew A Hannah[1,4]

Address: [1]Max Planck Institute of Molecular Plant Physiology, Am Mühlenberg 1, D-14476 Potsdam-Golm, Germany, [2]Centro de Biotecnología, Universidad Técnica Federico Santa María, General Bari 699 Valparaíso, Chile, [3]RIKEN Plant Science Center, Tsurumi-ku, Suehiro-cho, 1-7-22 Yokohama, Kanagawa, 230-0045, Japan and [4]Bayer BioScience N.V., Technologiepark 38, B-9052, Gent, Belgium

Email: Álvaro Cuadros-Inostroza* - inostroza@mpimp-golm.mpg.de; Camila Caldana - caldana@mpimp-golm.mpg.de; Henning Redestig - henning.red@psc.riken.jp; Miyako Kusano - mkusano005@psc.riken.jp; Jan Lisec - lisec@mpimp-golm.mpg.de; Hugo Peña-Cortés - hugo.pena@usm.cl; Lothar Willmitzer - willmitzer@mpimp-golm.mpg.de; Matthew A Hannah - matthew.hannah@bayercropscience.com

* Corresponding author

Published: 16 December 2009

BMC Bioinformatics 2009, **10**:428 doi:10.1186/1471-2105-10-428

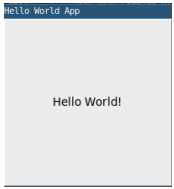
Received: 20 August 2009
Accepted: 16 December 2009

This article is available from: http://www.biomedcentral.com/1471-2105/10/428

GUI Aims

- providing an easy to use interface for the user to interact with an application
- CLI: commandline interfaces reduced interaction, good for automating, worser for intuitive usage
- GUI: graphical user interface good for interactive usage, bad for automated usage

1.1 Hello World




```
> library(tcltk) # library
> tt=tktoplevel() # main window
> tkwm.title(tt,"Hello World App") # app name
<Tcl>
> tkpack( # geometry manager
+     ttkbutton(  # widget
+     tt,text="Hello World!", # parent and options
+     command=function () {  # callback
+         tkdestroy(tt)
+     }),
+     fill='both',expand=TRUE) # geometry options
<Tcl>
> tkdestroy(tt) # just to close it in my script
```

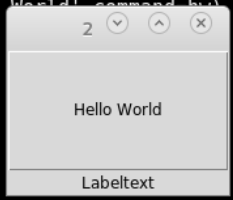
Hello World in steps

```
> library(tcltk2) # more widgets herein
> hw = function () { print('Hello World'); }
> tt=tktoplevel()
> tkwm.title(tt,'DGApp')
<Tcl>
> tkbtn=ttkbutton(tt,text='Hello World',
+     command=hw)
> tkpack(tkbtn,side='top',expand=TRUE,
+     fill='both')
<Tcl>
> tklbl=ttklabel(tt,text='Labeltext')
> tkpack(tklbl)
<Tcl>
> tkdestroy(tt)
```

```
 tkbutton(tt,text='Hello World',command=hw)
ID
1] ".2.1"

env
environment: 0x16f48c8>

ttr(,"class")
1] "tkwin"
 tkpack(tkbtn,side='top',expand=TRUE,fill='both')
rror in .Tcl.args.objv(...) : object 'tkbtn' not fo
 tkbtn=tkbutton(tt,text='Hello World',command=hw)
 tkpack(tkbtn,side='top',expand=TRUE,fill='both')
Tcl>
 tklbl=tklabel(tt,text='Labeltext')
 tkpack(tklbl)
Tcl>
 [1] "Hello World"
```



Hello World observations

- tktoplevel - main window
- tkwm - properties of main window
- tkpack - geometry mamanger
- ttkbutton, ttklabel - widget
- command option - callback with function

Tk: R vs Python3

- Python:

```
root = tk.toplevel()
button = tk.Button(root)
button.pack()
```

- object.method()

- R:

```
root = tktoplevel()
button = tkbutton(root)
tkpack(button)
```

- tkmethod(object)

⇒ **One API for Python, Perl, R, Ruby, C++ and Tcl/Tk!!**

Advantages of Tcl/Tk API

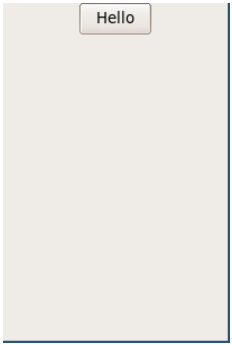
- installed with overy R (Mac OSX?)
- crossplatform (all major OS)
- lightweight
- cross language (Perl, Python, Ruby, R, ...)
- can be easily extended
- highly introspective as R

1.2 Layout Widgets

Toplevel

- toplevel is a window of the application
- you can have several toplevel in the same application
- toplevels contain other widgets, like frames, labels, buttons, checkboxes, menus etc.
- every GUI application needs a toplevel
- you can set the title of a toplevel and other properties like width, height and position on the screen

```
> tt=tktoplevel()
> tkwm.title(tt,'DGApp')
<Tcl>
> tkwm.geometry(tt,'140x200')
<Tcl>
> tkpack(ttkbutton(tt,text='Hello'))
<Tcl>
> source('../scripts/screenshot.r')
> screenshot(tt,'toplevel.png')
> tkdestroy(tt)
```



screenshot.r

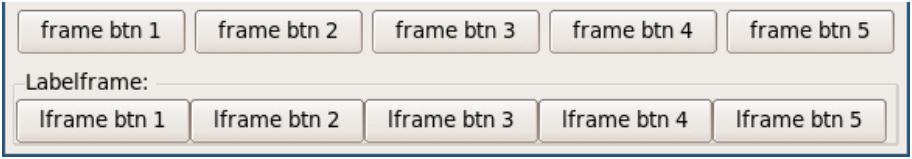
What other tkwm commands are available?

```
> options(width=55)
> grep("tkwm",ls("package:tcltk"),value=TRUE)
 [1] "tkwm.aspect"            "tkwm.client"
 [3] "tkwm.colormapwindows"   "tkwm.command"
 [5] "tkwm.deiconify"         "tkwm.focusmodel"
 [7] "tkwm.frame"             "tkwm.geometry"
 [9] "tkwm.grid"              "tkwm.group"
[11] "tkwm.iconbitmap"        "tkwm.iconify"
[13] "tkwm.iconmask"          "tkwm.iconname"
[15] "tkwm.iconposition"      "tkwm.iconwindow"
[17] "tkwm.maxsize"           "tkwm.minsize"
[19] "tkwm.overrideredirect" "tkwm.positionfrom"
[21] "tkwm.protocol"          "tkwm.resizable"
[23] "tkwm.sizefrom"          "tkwm.state"
[25] "tkwm.title"             "tkwm.transient"
[27] "tkwm.withdraw"
```

Frames / Labelframes

```
> tt=tktoplevel()
> tkframe=ttkframe(tt)
> for (i in 1:5) {
+   tkpack(ttkbutton(tkframe,
+   text=paste('frame btn',i)),side='left',
+   padx=3)
+ }
> tklframe=ttklabelframe(tt,text='Labelframe: ')
> for (i in 1:5) {
+    tkpack(ttkbutton(tklframe,
+     text=paste('lframe btn',i)),side='left')
+ }
> tkpack(tkframe,padx=5,pady=5)
<Tcl>
> tkpack(tklframe,padx=5,pady=5,fill='both',
+     expand=TRUE)
<Tcl>
```

```
> screenshot(tt,'toplevel-02.png')
> tkdestroy(tt)
```



Layout manager commands

- **tkpack**, just pack into the current container
- options `http://www.tcl.tk/man/tcl/TkCmd/pack.htm`
  - padx, pady amount of space around the widget
  - ipadx, ipady internal padding
  - expand: should container propagate if parent widget size is changed
  - fill: x,y or both should container fill space in x and/or y direction
- **tkgrid**, matrix like layout:
- options `http://www.tcl.tk/man/tcl/TkCmd/grid.htm`
  - column, columnspan, row, rowspan
  - padx, pady, ipadx, ipady

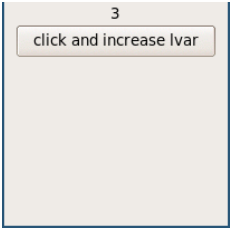
1.3 Basic Widgets

- Frames, layout container
- Labels (tklabel, ttklabel)
- Buttons (tkbutton, ttkbutton)
- Checkbutons (tkcheckbutton, ttkcheckbutton)
- Radiobutons (tkradiobutton,ttkradiobutton)
- Entries (tkentry, ttkentry)
- Comboboxes (ttkcombobx)

Label and Button

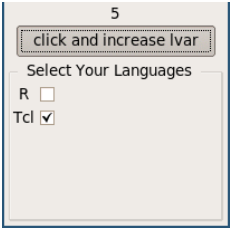
```
> tt=tktoplevel()
> lvar=tclVar('1')
> increaseLvar = function () {
+     tclvalue(lvar) =
+     as.integer(tclvalue(lvar)) +1
+ }
> tkl=ttklabel(tt,textvariable=lvar)
> tkbtn=ttkbutton(tt,command=increaseLvar,
+     text='click and increase lvar')
> tkpack(tkl)
<Tcl>
> tkpack(tkbtn)
<Tcl>
> tkinvoke(tkbtn)
<Tcl> 2
> tkinvoke(tkbtn)
```

```
<Tcl> 3
> screenshot(tt,'textvar.png')
```



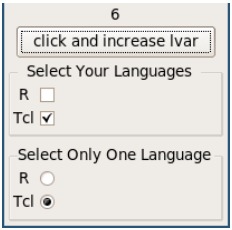
Checkbutton, Radiobuttons

```
> tklf=ttklabelframe(tt,
+     text='  Select Your Languages  ')
> checkedR=tclVar('0')
> checkedTcl=tclVar('1')
> tkgrid(tklabel(tklf,text='R'),
+     ttkcheckbutton(tklf,variable=checkedR))
<Tcl>
> tkgrid(tklabel(tklf,text='Tcl'),
+     ttkcheckbutton(tklf,variable=checkedTcl))
<Tcl>
> tkpack(tklf,padx=4,pady=4,fill='both',
+     expand=TRUE)
<Tcl>
> screenshot(tt,'checkbutton.png')
```




```
> tklf=ttklabelframe(tt,
+     text='Select Only One Language ')
> checkedLang=tclVar('Tcl')
> tkgrid(ttklabel(tklf,text='R'),
+     ttkradiobutton(tklf,variable=checkedLang,
+         value='R'))
<Tcl>
> tkgrid(ttklabel(tklf,text='Tcl'),
+     ttkradiobutton(tklf,variable=checkedLang,
+        value='Tcl'))
<Tcl>
```

```
> tkpack(tklf,padx=4,pady=4,fill='both',
+      expand=TRUE)
<Tcl>
> screenshot(tt,'radiobutton.png')
> tkdestroy(tt)
```

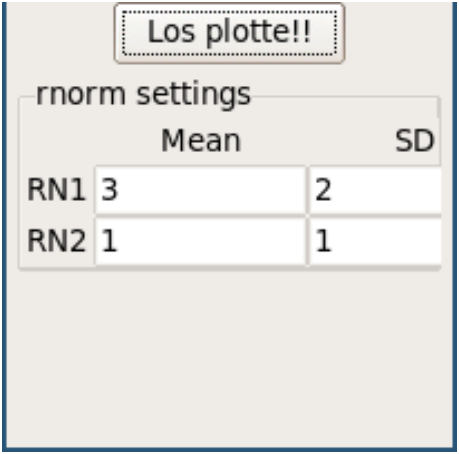


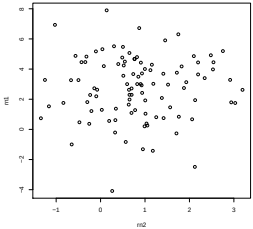
Entry

```
> tt=tktoplevel()
> tkwm.geometry(tt,'300x150')
<Tcl>
> tklf=ttklabelframe(tt,text='rnorm settings')
> rn1Mean=tclVar('1')
> rn1Sd=tclVar('2')
> rn2Mean=tclVar('1')
> rn2Sd=tclVar('1')
> asInt = function (vname) {
+     return(as.integer(tclvalue(vname)))
+ }
> plotXY = function () {
+     pdf('test-plot.pdf')
+     rn1=rnorm(100,mean=asInt(rn1Mean),
+      sd=asInt(rn1Sd))
+     rn2=rnorm(100,mean=asInt(rn2Mean),
```

```
+       sd=asInt(rn2Sd))
+      print(head(rn1))
+      plot(rn1~rn2)
+      dev.off()
+ }
> tkgrid(tklabel(tklf,text=''),ttklabel(tklf,
+   text='Mean'),ttklabel(tklf,text='SD'))
<Tcl>
> tkgrid(ttklabel(tklf,text='RN1'),
+   ttkentry(tklf,textvariable=rn1Mean,width=10),
+   ttkentry(tklf,textvariable=rn1Sd,width=10))
<Tcl>
> tkgrid(ttklabel(tklf,text='RN2'),
+   ttkentry(tklf,textvariable=rn2Mean,width=10),
+   ttkentry(tklf,textvariable=rn2Sd,width=10))
<Tcl>
> cmdBtn=ttkbutton(tt,command=plotXY,
+   text='Los plotte!!')
> tkpack(cmdBtn)
```

```
<Tcl>
> tkpack(tklf,padx=5,pady=5)
<Tcl>
> tclvalue(rn1Mean)='3'
> tkinvoke(cmdBtn)
[1]  4.8284627  6.7186792  4.4521076  4.3341658
[5]  4.6583481 -0.2660852
<Tcl>
> screenshot(tt,'plotxy.png')
```


Los plotte!!
rnorm settings
Mean
SD
RN1
3
2
RN2
1
1


rn1
rn2

1.4 Documentation

```
> options(useFancyQuotes = FALSE)
> getAnywhere('tkwm.geometry')
A single object matching 'tkwm.geometry' was found
It was found in the following places
  package:tcltk
  namespace:tcltk
with value

function (...)
tcl("wm", "geometry", ...)
<bytecode: 0x55b28ff34378>
<environment: namespace:tcltk>
```

Try: `$ man n wm` and search for `/wm geometry`

```
> getAnywhere('tkdestroy')
```

```
A single object matching 'tkdestroy' was found
It was found in the following places
  package:tcltk
  namespace:tcltk
with value

function (win)
{
    tcl("destroy", win)
    ID <- .Tk.ID(win)
    env <- get("parent", envir = win$env)$env
    if (exists(ID, envir = env, inherits = FALSE))
        rm(list = ID, envir = env)
}
<bytecode: 0x55b2901049d8>
<environment: namespace:tcltk>
```

```
> getAnywhere('ttknotebook')
```

```
A single object matching 'ttknotebook' was found
It was found in the following places
  package:tcltk
  namespace:tcltk
with value

function (parent, ...)
tkwidget(parent, "ttk::notebook", ...)
<bytecode: 0x55b28f387df8>
<environment: namespace:tcltk>
```

Try: `$ man n ttk::notebook`

```
ttk::notebook(n)                Tk Themed Widget                ttk::notebook(n)


NAME
       ttk::notebook - Multi-paned container widget

SYNOPSIS
       ttk::notebook pathname ?options...?
       pathname add window ?options...?
       pathname insert index window ?options...?

DESCRIPTION
       A  ttk::notebook widget manages a collection of windows and displays a sin-
       gle one at a time.  Each slave window is associated with a tab,  which  the
       user may select to change the currently-displayed window.
```

R vs Tcl/Tk

```
# Session in R
tt <- tktoplevel()
b <- ttkbutton(tt,
    text="Click Me")
tkpack(b, side = "top",
  anchor="w",
  padx=3,pady=3)
foo <- function() {
  print("Hello World")
}
tkconfigure(b,
 command=foo)
tkcget(b,text=NULL)
```

```
# Same Session in Tcl
set tt [toplevel .1]
set b [ttk::button $tt.b \
  -text "Click Me"]
pack $b -side top -anchor w \
  -padx 3 -pady 3
proc foo {} {
    puts "Hello World"
}
$b configure -command foo
$b cget -text
```

1.5 Dialogs

- tkmessageBox
- tkgetOpenFile
- tkgetSaveFile
- tkchooseDirectory
- (tkchooseColor)
- (tkchooseFont)

tkmessageBox

```
> #screenshot('Greetings','messagebox-01.png')
> res <- tkmessageBox(title = "Greetings",
+   message = "Hello, world!", icon = "info",
+   type = "ok")
> res            # This is a Tcl variable
<Tcl> ok
> ## <Tcl> ok
> tclvalue(res) # Get the value from a Tcl variable
[1] "ok"
> ## [1] "ok"
> as.character(res) # It works also that way
[1] "ok"
> ## [1] "ok"
```

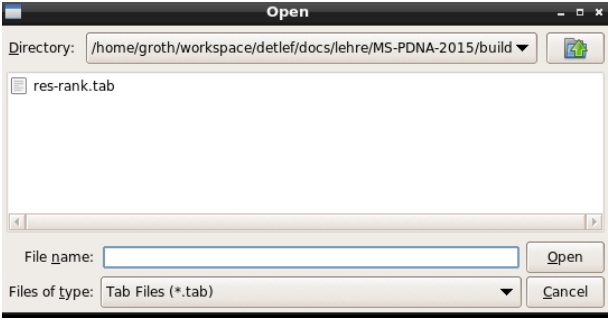
Other types: `http://www.tcl.tk/man/tcl/TkCmd/messageBox.htm`

```
tkmessageBox(
    message = "An error has occurred!",
    icon = "error", type = "ok")
tkmessageBox(
    message = "Do you want to save before quitting?"
    icon = "question", type = "yesnocancel",
    default = "yes")
```

tkgetOpenFile

```
> getTabFileData <- function() {
+  name <- tclvalue(tkgetOpenFile(title = 'Open',
+   filetypes = "{{Tab Files} {.tab}}
+   {{CSV Files} {.csv}}
+   {{All Files} *}"))
+   if (name == "") { #Return an empty data frame
+      return(data.frame())#if no file was selected
+  }
+  data <- read.table(name,sep="\t",header=TRUE)
+  assign("tab_data", data, envir = .GlobalEnv)
+ }
> tt=tktoplevel()
> btn=ttkbutton(tt,text="Load Tab Data",
+     command=getTabFileData)
> tkpack(btn)
<Tcl>
```

```
> tkinvoke(btn)
<Tcl>
> tkdestroy(tt)
```



tkgetSaveFile

- Example: save the current plot to a png

```
> png_filename <- tclvalue(tkgetSaveFile(
+     title='Save',initialfile = "test.png",
+   filetypes = "{{PNG Files} {.png}}
+   {{JPG Files} {.jpg .jpeg}} {{All Files} * }"))
> print(png_filename)
[1] "/home/groth/workspace/delfgroth/docs/lehre/PwR-2020/
```

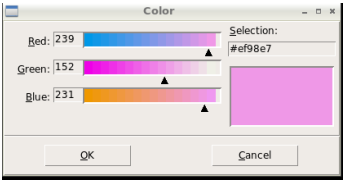
tkchooseDirectory

```
> opendir<-function(...) {
+   return(tclvalue(tkchooseDirectory(...)))
+ }
> openfile<-function(...) {
+   return(tclvalue(tkgetOpenFile(...)))
+ }
> openfile(title='Open any file ...')
[1] "/home/groth/workspace/delfgroth/docs/lehre/PwR-2020/
```

tkchooseColor

- Not implemented in R-tcltk but in Tcl/Tk itself.
- Let's implement it:

```
> tkchooseColor=function(...)
+     tcl("tk_chooseColor", ...)
> col=tkchooseColor(title='Color')
> col
<Tcl> #efebe7
> tclvalue(col)
[1] "#efebe7"
```

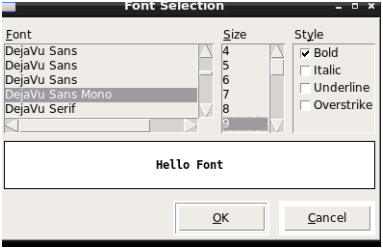


tkchooseFont

- Not in R tcltk, tcltk2
- But in standard TclTk

⇒ our own implementation

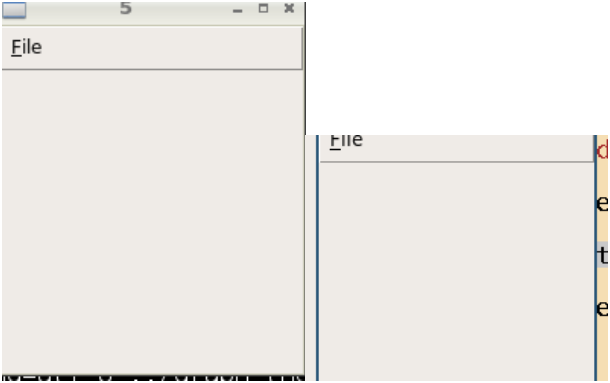
```
> tkchooseFont = function (...) {
+     tcl('tk','fontchooser','configure',...)
+     tcl('tk','fontchooser','show')
+ }
> tt=tktoplevel()
> font=tkchooseFont(title='Font Selection',
+     parent=tt)
> print(tclvalue(font))
[1] ""
```


Font Selection
Font
DejaVu Sans
DejaVu Sans
DejaVu Sans
DejaVu Sans Mono
DejaVu Serif
Size
4
5
6
7
8
9
Style
Bold
Italic
Underline
Overstrike
Hello Font
OK
Cancel

1.6 Menues

```
> openfile=function() {}
> opendir=function() {}
> tt<-tktoplevel()
> topMenu<-tkmenu(tt,tearoff=FALSE)
> tkconfigure(tt,menu=topMenu)
<Tcl>
> fileMenu<-tkmenu(topMenu, tearoff=FALSE)
> tkadd(fileMenu,"command",label="Open file",
+     command=openfile,underline=0)
<Tcl>
> tkadd(fileMenu, "command",
+     label="Open directory",
+     command=opendir,underline=5)
<Tcl>
> tkadd(fileMenu,'separator')
<Tcl>
```

```
> tkadd(fileMenu,"command",label="Quit",
+     command=function() tkdestroy(tt),
+     underline=0)
<Tcl>
> tkadd(topMenu, "cascade", label="File",
+     menu=fileMenu,
+     underline=0)
<Tcl>
> tkfocus(tt)
<Tcl>
> # just for screenshot
> geo=as.character(tkwinfo('geometry',tt))
> geo=strsplit(geo,"\\+",perl=TRUE)[[1]]
> tkpost(fileMenu,as.integer(geo[2])+2,
+     as.integer(geo[3])+8)
<Tcl>
> screenshot(tt,'tcltk-menu-post.png',rootcrop=TRUE)
> tkdestroy(tt)
```


5
File
File

1.7 Advanced Widgets Combobox

A Entry widget with a dropdown list. Content can be tied as well to a tcl variable

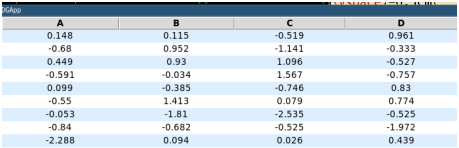
```
> tt=tktoplevel()
> fruit=tclVar('Orange')
> tkcombo=ttkcombobox(tt,textvariable=fruit)
> tkconfigure(tkcombo,
+     values=c('Apple','Orange','Banana'))
<Tcl>
> tkpack(tkcombo)
<Tcl>
> #screenshot(tt,'ttkcombo.png')
> tkdestroy(tt)
>
```


Orange
Apple
Orange
Banana

Table (Treeview)-Widget

```
> tt=tktoplevel()
> tkwm.title(tt,'DGApp')
<Tcl>
> tview=ttktreeview(tt,columns=LETTERS[1:4],
+     show='headings')
> # our own implementation
> tkheading = function (widget,...) {
+     tcl(widget,'heading',...)
+ }
> for (l in LETTERS[1:4]) {
+     tkheading(tview,l,text=l)
+     tcl(tview,"column",l,anchor='center')
+ }
> tktag.configure(tview,"band0",background='#FFFFFF')
<Tcl>
> tktag.configure(tview,"band1",background='#DDEEFF')
```

```
<Tcl>
> for (i in 1:30) {
+     band=paste('band',i%%2,sep='')
+     item=tkinsert(tview,'','end',
+         values=round(rnorm(4),3))
+     tktag.add(tview,band,item)
+ }
> tkpack(tview,side='top',fill='both',expand=TRUE)
<Tcl>
> #screenshot(tt,'tcltk-04.png')
```



A	B	C	D
0.148	0.115	-0.519	0.961
-0.68	0.952	-1.141	-0.333
0.449	0.93	1.096	-0.527
-0.591	-0.034	1.567	-0.757
0.099	-0.385	-0.746	0.83
-0.55	1.413	0.079	0.774
-0.053	-1.81	-2.535	-0.525
-0.84	-0.682	-0.525	-1.972
-2.288	0.094	0.026	0.439

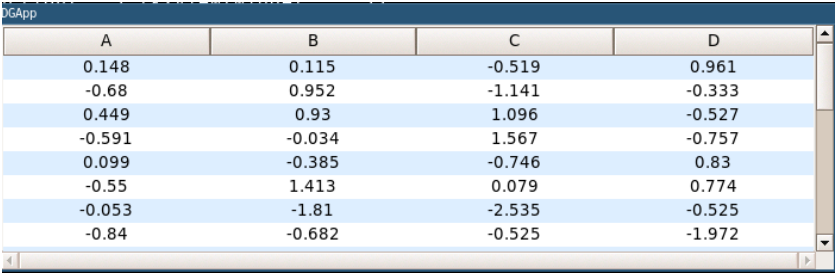
Manualpage:

https://www.tcl.tk/man/tcl8.6/TkCmd/ttk_treeview.htm

Scrollbars

```
> tkscrolled = function (parent,widget,...) {
+  scrx <- tk2scrollbar(parent,
+     orient = 'horizontal',
+     command = function(...) tkxview(widget, ...))
+  scry <- tk2scrollbar(parent, orient = 'vertical',
+     command = function(...) tkyview(widget, ...))
+  tkconfigure(widget, xscrollcommand =
+     function(...) tkset(scrx, ...),
+      yscrollcommand = function(...) tkset(scry, ...))
+   tkgrid(widget, scry, sticky = 'nsew')
+   tkgrid.rowconfigure(parent, widget, weight = 1)
+   tkgrid.columnconfigure(parent, widget, weight = 1)
+   tkgrid(scrx, sticky = 'ew')
+ }
> tkpack.forget(tview)
<Tcl>
> tkscrolled(tt,tview)
```

```
<Tcl>
> #screenshot(tt,'tcltk-06.png')
> tkdestroy(tt)
```



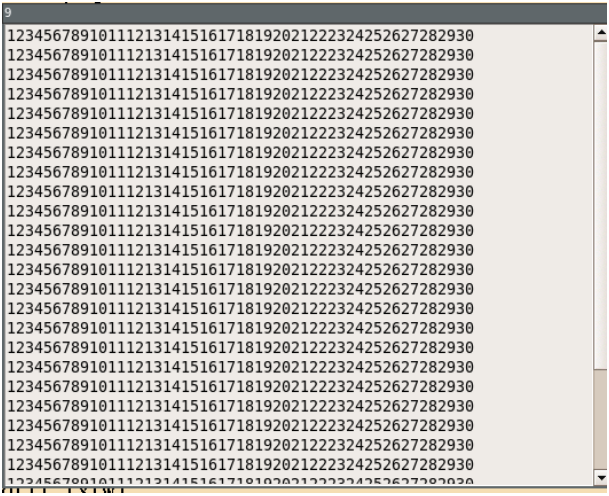
A	B	C	D
0.148	0.115	-0.519	0.961
-0.68	0.952	-1.141	-0.333
0.449	0.93	1.096	-0.527
-0.591	-0.034	1.567	-0.757
0.099	-0.385	-0.746	0.83
-0.55	1.413	0.079	0.774
-0.053	-1.81	-2.535	-0.525
-0.84	-0.682	-0.525	-1.972

Autoscroll - Implementation

```
> tk2autoscrolled = function (parent,widget,...) {
+  tclRequire('autoscroll')
+  scrx <- tk2scrollbar(parent, orient = 'horizontal',
+     command = function(...) tkxview(widget, ...))
+  scry <- tk2scrollbar(parent,orient = 'vertical',
+     command = function(...) tkyview(widget, ...))
+  tkconfigure(widget, xscrollcommand =
+     function(...) tkset(scrx, ...),
+     yscrollcommand = function(...) tkset(scry, ...))
+     tkpack(scrx,side='bottom',fill='x')
+     tkpack(scry,side='right',fill='y')
+     tkpack(widget,side='top',fill='both',expand=TRUE)
+     tcl('::autoscroll::autoscroll',scry)
+     tcl('::autoscroll::autoscroll',scrx)
+ }
```

Autoscroll with Text widget

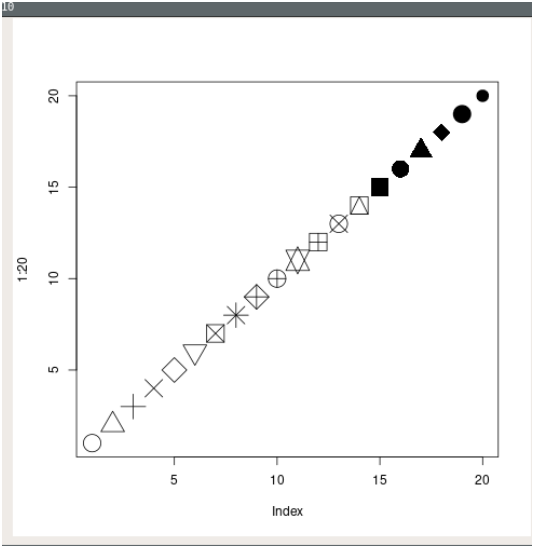
```
> tt=tktoplevel()
> txt=tktext(tt,wrap='none')
> tk2autoscrolled(tt,txt)
<Tcl> .13.2
> for (i in 1:30) {
+     for (j in 1:30) {
+         tkinsert(txt,'end',j)
+     }
+     tkinsert(txt,'end',"\n")
+ }
> tkwm.geometry(tt,'300x400+30+30')
<Tcl>
> screenshot(tt,'tcltk-autoscroll-01.png',update=TRUE)
> tkwm.geometry(tt,'500x600+30+30')
<Tcl>
> #screenshot(tt,'tcltk-autoscroll-02.png',update=TRUE)
> try(tkdestroy(tt))
```



```
123456789101112131415161718192021222324252627282930
123456789101112131415161718192021222324252627282930
123456789101112131415161718192021222324252627282930
123456789101112131415161718192021222324252627282930
123456789101112131415161718192021222324252627282930
123456789101112131415161718192021222324252627282930
123456789101112131415161718192021222324252627282930
123456789101112131415161718192021222324252627282930
123456789101112131415161718192021222324252627282930
123456789101112131415161718192021222324252627282930
123456789101112131415161718192021222324252627282930
123456789101112131415161718192021222324252627282930
123456789101112131415161718192021222324252627282930
123456789101112131415161718192021222324252627282930
123456789101112131415161718192021222324252627282930
123456789101112131415161718192021222324252627282930
123456789101112131415161718192021222324252627282930
123456789101112131415161718192021222324252627282930
123456789101112131415161718192021222324252627282930
123456789101112131415161718192021222324252627282930
123456789101112131415161718192021222324252627282930
123456789101112131415161718192021222324252627282930
123456789101112131415161718192021222324252627282930
```

Images

```
> tt=tktoplevel()
> png('test-plot-01.png')
> plot(1:20,pch=1:20,cex=3)
> dev.off()
null device
          1
> imgfile = 'test-plot-01.png'
> image1 = tclVar()
> tkimage.create('photo', image1, file = imgfile)
<Tcl> ::RTcl10
> tkpack(tklabel(tt,image=image1))
<Tcl>
> tkdestroy(tt)
```


1:20
5
10
15
20
5
10
15
20
Index

1.8 Advanced Layout Widgets

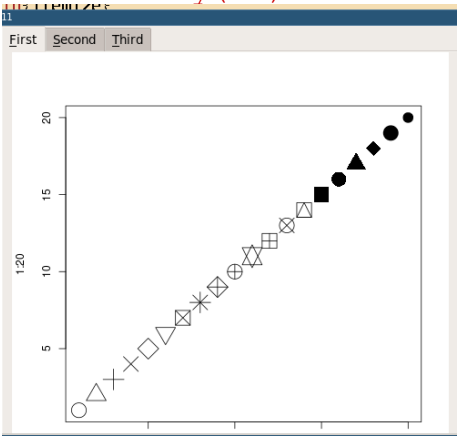
Layout - Notebooks

- container widget
- `ttk::notebook`
- `http://www.tcl.tk/man/tcl/TkCmd/ttk_notebook.htm`
- no need to pack added pages
- tabs are added using `tkadd`

```
> tt=tktoplevel()
> tknb=ttknotebook(tt)
> tf1=ttkframe(tknb)
> tf2=ttkframe(tknb)
> tf3=ttkframe(tknb)
> tkadd(tknb,tf1,text='First',underline=0)
```

```
<Tcl>
> tkadd(tknb,tf2,text='Second',underline=0)
<Tcl>
> tkadd(tknb,tf3,text='Third',underline=0)
<Tcl>
> tcl('::ttk::notebook::enableTraversal', tknb)
<Tcl> .15.1
> args(tk2notetraverse)
function (nb)
NULL
> body(tk2notetraverse)
{
    res <- tcl("ttk::notebook::enableTraversal", nb)
    return(invisible(res))
}
> tkpack(tknb,side='top',expand=TRUE,fill='both')
<Tcl>
```

```
> tkpack(tklabel(tf1,image=image1))
<Tcl>
> tkpack(tkbutton(tf1,text='Hello'))
<Tcl>
> tkdestroy(tt)
```



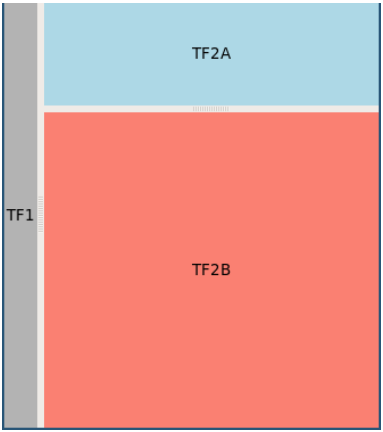
Layout - Panedwindow

- container widget
- added widgets can be resized by dragging at the sash of the panedwindow

```
> tt=tktoplevel()
> tfpw=ttkpanedwindow(tt,orient='horizontal',
+     width=600,height=400)
> tf1=ttkframe(tfpw)
> tf2=ttkframe(tfpw)
> tf2pw=ttkpanedwindow(tf2,orient='vertical',
+     height=400,width=400)
> tf2a=ttkframe(tf2pw)
> tf2b=ttkframe(tf2pw)
> tkadd(tfpw,tf1)
<Tcl>
> tkadd(tfpw,tf2)
```

```
<Tcl>
> tkadd(tf2pw,tf2a)
<Tcl>
> tkadd(tf2pw,tf2b)
<Tcl>
> tkpack(tf2pw,side='top',expand=TRUE,
+     fill='both')
<Tcl>
> # ttklabel has no bg option
> tkpack(tklabel(tf1,text='TF1',bg='grey70'),
+     expand=TRUE,fill='both')
<Tcl>
> tkpack(tklabel(tf2a,text='TF2A',bg='light blue'),
+     expand=TRUE,fill='both')
<Tcl>
> tkpack(tklabel(tf2b,text='TF2B',bg='salmon'),
+     expand=TRUE,fill='both')
<Tcl>
```

```
> # last the main widget at best to avoid flicker
> tkpack(tfpw,side='top',expand=TRUE,fill='both')
<Tcl>
> tkdestroy(tt)
```



1.9 Events

Often specific user actions needs to be captured.

Summary: `http://www.sciviews.org/recipes/tcltk/TclTk-event-binding/`

- keyboard events <Keypress-Ctrl-a>
- mouse events <Mouse-1> left mouse button click
- virtual events like «Tabchanged», <Enter> (mouse pointer enters a widget)

Syntax:

```
tkbind(widget,'event',function)
```

Examples

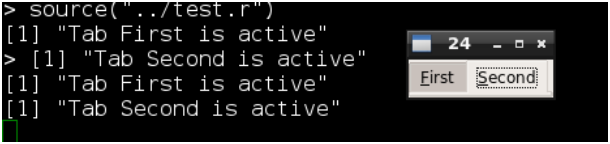
```
> tt=tktoplevel()
> entering = function () {
+     print('Entering the Label ...')
+ }
> leaving = function () {
+     print('leaving the label')
+ }
> tkl=ttklabel(tt,text='enter me')
> tkbind(tkl,'<Enter>',entering)
<Tcl>
> tkbind(tkl,'<Leave>',leaving)
<Tcl>
> tkpack(tkl)
<Tcl>
> tkdestroy(tt)
```

```
> source("../test.r")
> [1] "Entering the Label ..."
[1] "leaving the label"
[1] "Entering the Label ..."
[1] "leaving the label"
[1] "Entering the Label ..."
[1] "leaving the label"
[1] "Entering the Label ..."
[1] "leaving the label"
```

Virtual Events

```
> tt=tktoplevel()
> tn=ttknotebook(tt)
> # wrapper function
> # ttk::notebook has a tab method which
> # is not wrapped by tcltk2
> tktab = function (widget,...) {
+     tcl(widget,'tab', ...)
+ }
> # event function
> changeTab = function () {
+   current=tclvalue(tkindex(tn,'current'))
+   # can't use text ... Tcl-like option -text
+   print(paste("Tab",
+     tktab(tn,current,'-text'),"is active"))
+     # R like option text=NULL
+   print(paste("Tab",
```

```
+     tktab(tn,current,text=NULL),"is active"))
+
+ }
> tkbind(tn,"<<NotebookTabChanged>>",changeTab)
<Tcl>
> ttkf1=ttkframe(tn)
> ttkf2=ttkframe(tn)
> tkadd(tn,ttkf1,text="First",underline=0)
<Tcl>
> tkadd(tn,ttkf2,text="Second",underline=0)
[1] "Tab First is active"
[1] "Tab First is active"
<Tcl>
> tkpack(tn)
<Tcl>
> tkdestroy(tt)
```


> source("../test.r")
[1] "Tab First is active"
> [1] "Tab Second is active"
[1] "Tab First is active"
[1] "Tab Second is active"
24
First
Second

Binding Parameters

Sometimes we would like to know which widgets has gotten an event.

`http://www.tcl.tk/man/tcl/TkCmd/bind.htm` (Substitutions)

- important substitions:
  - %x => xclick coordinates
  - %y => yclick coordinates
  - %W => which widget got the event
  - many others ...

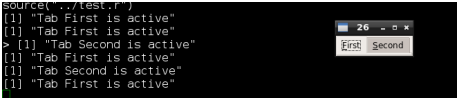
The R syntax is slightly different:

```
> .Tcl.callback(function(W,x,y)cat(W,x,y,'\n'))
[1] "R_call 0x55b28f29d918 %W %x %y"
```

Binding Parameters - Notebook Example

```
> tt=tktoplevel()
> tn=ttknotebook(tt)
> # wrapper function
> # ttk::notebook has a tab method which
> # is not wrapped by tcltk2
> tktab = function (widget,...) {
+     tcl(widget,'tab', ...)
+ }
> # event function
> changeTab = function (widget) {
+   # no global variable anymore
+   current=tclvalue(tkindex(widget,'current'))
+   # can't use text ...
+   print(paste("Tab",
+     tktab(widget,current,'-text'),"is active"))
+ }
```

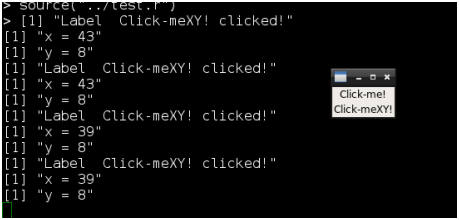
```
> tkbind(tn,"<<NotebookTabChanged>>",
+  function(W) changeTab(W) )
<Tcl>
> ttkf1=ttkframe(tn)
> ttkf2=ttkframe(tn)
> tkadd(tn,ttkf1,text="First",underline=0)
<Tcl>
> tkadd(tn,ttkf2,text="Second",underline=0)
<Tcl>
> tkpack(tn)
<Tcl>
> tkdestroy(tt)
```



Binding Parameters - XY - Example

```
> tt=tktoplevel()
> clickLabel = function (W,x=0,y=0) {
+     # Tcl like with minus
+     print(paste("Label ",tkcget(W,'-text'),
+         "clicked!"))
+     # R like with NULL
+     print(paste("Label ",tkcget(W,text=NULL),
+         "clicked!"))
+     if(x>0) {
+         print(paste("x =",x))
+         print(paste("y =",y))
+     }
+ }
> tkl2=tklabel(tt,text='Click-me!')
> tkbind(tkl2,'<Button-1>',
+     function(W){ clickLabel(W) })
```

```
<Tcl>
> tkl3=tklabel(tt,text='Click-meXY!')
> tkbind(tkl3,'<Button-1>',
+  function(W,x,y){ clickLabel(W,x,y) })
<Tcl>
> tkpack(tkl2)
<Tcl>
> tkpack(tkl3)
<Tcl>
> tkdestroy(tt)
```

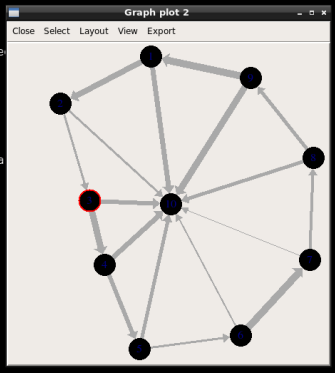


igraph::tkplot

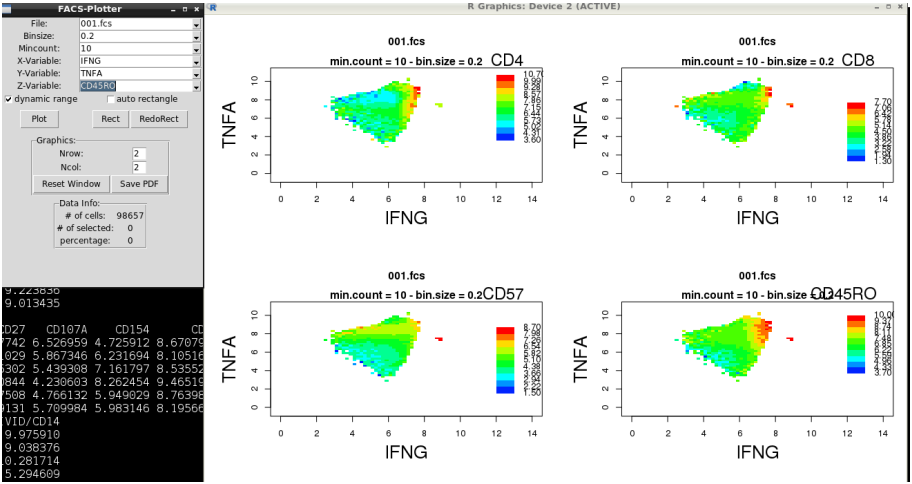
```
The following object is masked from 'package:base':

    union

> g <- make_star(10, center=10) %u% make_ring(9, directe
> E(g)$width <- sample(1:10, ecount(g), replace=TRUE)
> lay <- layout_nicely(g)
>
> id <- tkplot(g, layout=lay)
> id
[1] 1
Warning message:
X11 protocol error: RenderBadPicture (invalid Picture pa
> tkplot.getcoords(id)
Error in eval(expr, envir, enclos) : object 'tkp.1' not
> id <- tkplot(g, layout=lay)
> tkplot.getcoords(id)
          [,1]      [,2]
 [1,] 201.99142 410.00000
 [2,]  75.48916 343.69582
 [3,] 116.00000 208.00000
 [4,] 137.00000 118.00000
 [5,] 185.73608   0.00000
 [6,] 328.10755  19.14816
 [7,] 424.71093 125.11378
 [8,] 430.00000 267.76821
 [9,] 342.08816 380.31503
[10,] 230.12706 203.30838
```



Cytometrie-Plotter



tk vs ttk

- tk is the old widget sets
- on Unix somtimes old looking tkscrollbar
- ttk is the newer widget set
- can be themed
- looking more modern ttkscrollbar or tk2scrollbar
- with ttk some options are missing fg, bg etc
- need to use themes, not every button should have a different color

```
> options(width=55)
> grep('ttk',ls('package:tcltk'),value=TRUE)
 [1] "ttkbutton"       "ttkcheckbutton" "ttkcombobox"
 [4] "ttkentry"        "ttkframe"       "ttklabel"
 [7] "ttklabelframe"   "ttkmenubutton"  "ttknotebook"
[10] "ttkpanedwindow" "ttkprogressbar" "ttkradiobutton"
```

```
[13] "ttkscale"       "ttkscrollbar"   "ttkseparator"
[16] "ttksizegrip"    "ttkspinbox"     "ttktreeview"
```

Summary commands

- tktoplevel (main window)
- ttkframe, ttklabelframe, ttknotebook, ttkpanedwindow (layout)
- tkpack, tkgrid (layout manager)
- tkmenu, ttkbutton, ttkentry, ttklabel and images (basic widgets)
- ttktreeview, tktext, ttkscrollbar (advanced widgets)
- tkbind and events (interaction)

Practical Project RFenView

- a viewer for chess fen positions
- menu, file-exit, file-open (dialog)
- menu, help-about, messagebox
- frame with paned window divider
- treeview left with fen positions
- after click on entry right image label with chessboard
- package for easy access to piece images and sample fen file
- two public functions `fen2png` and `fenview` for starting the GUI

Layout application

```
+-----------------------------------------------+
| File              (menubar)            Help   |
+-----------------------------------------------+
| pos1     |                                    |
| pos2     |                                    |
| pos3     |                                    |
| ...      |      chess position image          |
|        <-|->                                  |
|        (pwi)                                  |
|          |         (label)                    |
| (tview)  |                                    |
|          |                                    |
+----------+------------------------------------+
```

Programming Steps

- layout
- function outlines
- functionality
- packaging
- building, testing, releasing

Summary

- Tcl/Tk-API usable in R as well with Python, Perl, Ruby, D, Go, ...
- How to read Tcl/Tk docs
- Widgets:
  - tktoplevel
  - t(t)kframe, ttklabelframe
  - ttknotebook, ttkpanedwindow
  - ttkbutton, ttklabel, ttkentry
  - ttkradiobutton, ttkcheckbutton, ttkcombobox
  - tkmenu, t(t)kmenubutton
  - ttktreeview, text
  - ttkscrollbar, wrapper tkscrolled
- Layout management: pack, grid

```
> save.image('05-rtcltk.RData')
> Stangle('../05-rtcltk.snw')
Writing to file 05-rtcltk.R
```

05-rtcltk.RData 05-rtcltk.R

Links

- Sciviews examples
  `https://web.archive.org/web/20180826195240/http://www.sciviews.org/images2/recipes-tcltk/Rtcltk.pdf`
- old url: `http://www.sciviews.org/recipes/tcltk/toc/`
- Adrain Wadell: `http://adrian.waddell.ch/EssentialSoftware/Rtcltk_geometry.pdf`
- Special issue on R-GUIs:
  `https://www.jstatsoft.org/issue/view/v049`
- Rcmdr: `https://socialsciences.mcmaster.ca/jfox/Misc/Rcmdr/`
- Tk Manual Pages
  `http://www.tcl.tk/man/TkCmd/contents.htm`

1.10 Exercise 4 - R-tcltk
Practical Project FastaView

- a viewer for FASTA files
- menu, file-exit, file-open (dialog)
- menu, help-about, messagebox
- frame with paned window divider
- treeview left with FASTA ids
- after click on entry text widget with sequences will be filled
- package for easy access to sample FASTA file
- public functions `read.fasta`, `print.FastaUtils`, `summary.FastaUtils` and `fastaview` for starting the GUI

Layout application

```
+-----------------------------------------------+
| File              (menubar)             Help  |
+-----------------------------------------------+
| id1       |                                   |
| id2       |                                   |
| id3       |                                   |
| ...       |      sequence (FASTA)             |
|         <-|->                                 |
|         (pwi)                                 |
|           |          (tktext)                 |
| (tview)   |                                   |
|           |                                   |
+-----------+-----------------------------------+
```

Steps

Step 1 (Layout A):

- toplevel with frame inside
- toplevel title: FenView - authorname
- tkmenu with entries File-Exit and Help-About
- simple placeholder functions:

```
OnExit = function () { print('Exit') }
```

Step 2 (Layout (B):

- ttkpanedwindow
- left add ttktreeview (1 column)
- right add ttktext
- use tkscrolled function for making them scrolled

Step 3 (Functionality):

- file->open should return filename
- file->exit should after questioning close toplevel not exit from R
- help->about should give message about application
- use read.fasta to retrieve all Ids from FASTA file and tkinsert into treeview
- tkbind click event
- on click tkdelete old text and tkinsert wrapped sequence into text widget with FASTA header

References