Examples of some of the widgets

Jianhua Zhang

July 22, 2024

1 Introduction

This package contains several widgets built using the R *tcltk* package or the *widgetTools* package to provide interfaces to perform different tasks. Available widgets include:

- `argsWidget` - a widget that allows users to input values for an argument list passed to the widget as an argument
- `DPExplorer` - a widget that allows users to view a `Bioconductor` data package and make selections
- `fileBrowser` - a widget that allows users to select a group of files from a local directory
- `importWizard` - a widget that allows users to import a data file into R, following an Excel style
- `objectBrowser` - a widget that allows users to browse R objects and make a selection
- `vExplorer` - a widget that allows users to explore vignettes of available R packages
- `eExplorer` - a widget that allows users to explor the executable examples contained by a given R package
- `pExplorer` - a widget that allows users to explor the structure and contents of a given R package

The following code invokes a widget that contains clickable buttons that brings up the widget when a button is clicked.

```
> library(tkWidgets)
> PWEnv <- new.env(hash = TRUE, parent = parent.frame(1))
> argsFun <- function(){
+     argsWidget(list(Entry1 = "aaaaa", Ehtry2 = "bbbb"))
+ }
> button1 <- button(wName = "button1", wValue = "argsWidget",
+                       wFuns = list(command = argsFun), wEnv = PWEnv)
> button2 <- button(wName = "button2", wValue = "DPExplorer",
+                       wFuns = list(command = DPExplorer), wEnv = PWEnv)
> button3 <- button(wName = "button3", wValue = "fileBrowser",
+                       wFuns = list(command = fileBrowser), wEnv = PWEnv)
> button4 <- button(wName = "button4", wValue = "importWizard",
+                       wFuns = list(command = importWizard), wEnv = PWEnv)
> button5 <- button(wName = "button5", wValue = "objectBrowser",
+                       wFuns = list(command = objectBrowser), wEnv = PWEnv)
> button6 <- button(wName = "button6", wValue = "vExplorer",
+                       wFuns = list(command = vExplorer), wEnv = PWEnv)
> button7 <- button(wName = "button7", wValue = "pExplorer",
+                       wFuns = list(command = pExplorer), wEnv = PWEnv)
> fun <- function(){eExplorer("tkWidgets")}
> button8 <- button(wName = "button8", wValue = "eExplorer",
+                       wFuns = list(command = fun), wEnv = PWEnv)
> pWidgets <- list(A = list(button1 = button1, button2 = button2), C =
+ list(button3 = button3, button4 = button4), D = list(button5 = button5,
+ button6 = button6), E = list(button7 = button7, button8 = button8))
> if(interactive()){
+ viewWidget <- widget(wTitle = "tkWidgets Examples", pWidgets, funs = list(),
+                     preFun = function() print("Hello"),
+                     postFun = function() print("Bye"), env = PWEnv)
+ }
```

This vignette provides more detailed describtions of the following three widgets.

2 pExplorer

`pExplorer` allows users to explore the contents of an R package. The widget can be invoked by providing a package name, a path to the directory containing the package, and the names of files and or subdirectory to be excluded from showing by the widget. The default behavior is to open the first package in the library of locally installed R (`.libPaths()`) with subdirectories/files named "Meta" and "latex" excluded if nothing is provided by a user.

Figure 2 shows the widget invoked by typing `pExplorer("base")`.

Figure 1: A snapshot of the widget when the example code chunk is executed

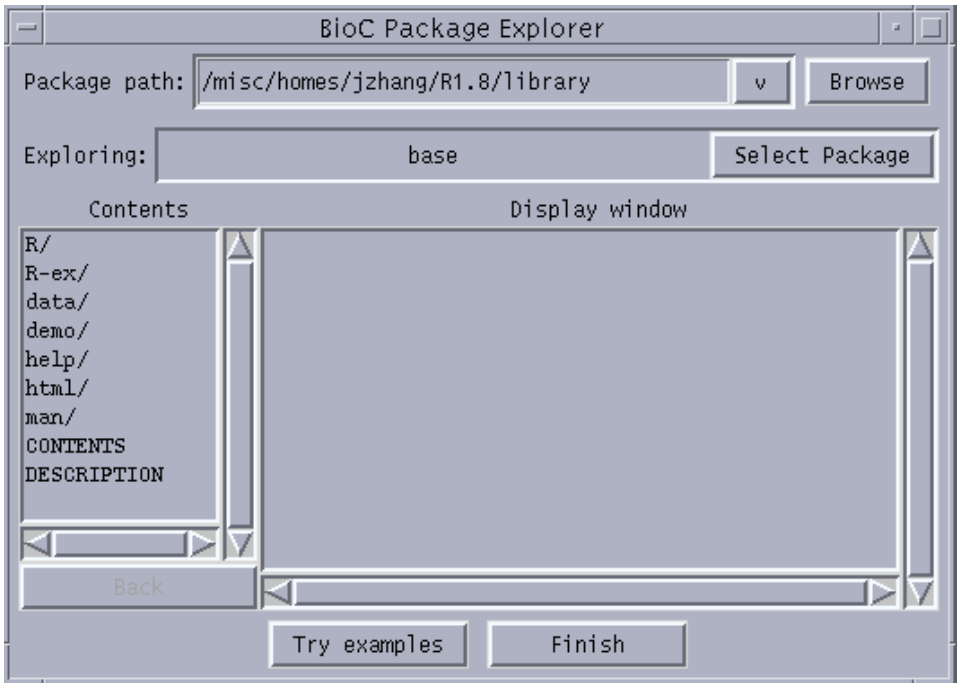

Figure 2: A snapshot of the widget when `pExplorer` is called

The following tasks can be performed through the interface:

- Typing in a valid path to a local directory containing R packages and then press the Enter key will update the name of the package being explored and show the contents of the first R package in the path in the `Contents` list box.
- Clicking the dropdown button for package path and selecting a path from the resulting dropdown list by double clicking will achieve the same results as typing in a new path.
- Clicking the `Browse` button allows users to selecting a directory containing R packages to achieve the same results as typing in a new path.
- Clicking the `Select Package` button allows users to pick a new package to explore and the name and contents of the package being explored are updated.
- Double clicking an item in the `Contents` list box results in different responses depending on the nature of the item being clicked. If the item is a subdirectory, the `Contents` will be updated with the contents of the subdirectory corresponding to the item just being clicked and the `Back` right below the list box will be activated. If the item is a file, the contents of the file will be displayed in the `Display window` text box.

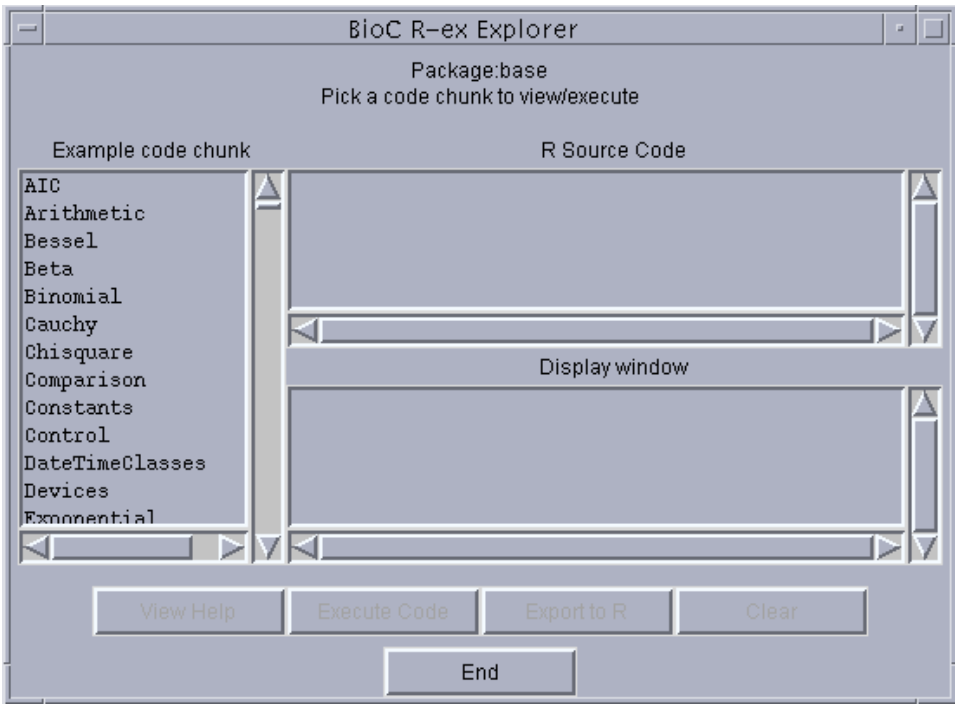

Figure 3: A snapshot of the widget when `pExplorer` is called by typing `eExplorer("base")`

- Clicking the `Back` button allows users to go back to the parent of the subdirectory whose contents are being displayed in `Contents` list box.
- Clicking the `Try examples` button invokes `eExplorer` going to be discussed next.

eExplorer

`eExplorer` allows users to explore the example code from all the help files for the specified package. When invoked by passing a valid package name, `eExplorer` will display the names of the files for example code stored in the `R-ex` subdirectory in the `Example code chunk` list box as shown by Figure 3.

Clicking an item in the list displays the code example in the `R Source Code` text box for users to view. Users may view the help file containing the example code by clicking the `View Help` button, execute the example code and have the output of the execution displayed in the `Diaplay window` by clicking the `Execute Code` button, or export the result of execution (if R objects have been created as the result of execution) to the global environment of the current R session by clicking the `Export to R`.

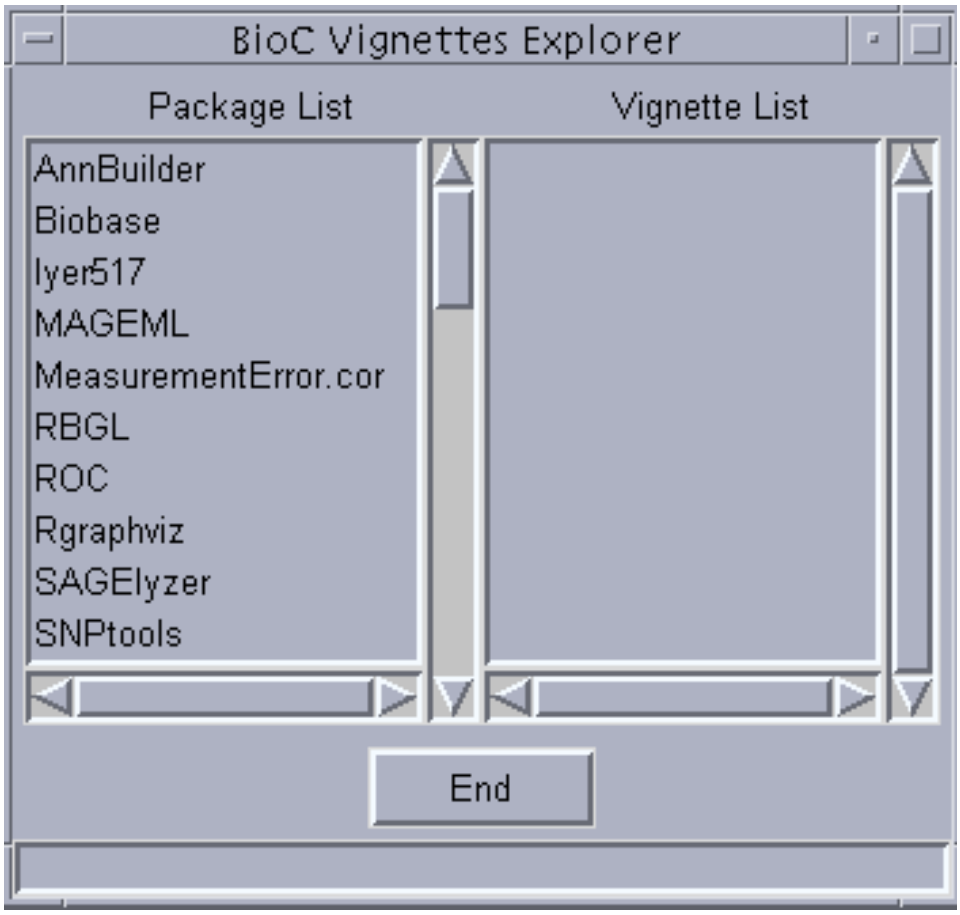

Figure 4: A snapshot of the widget when `vExplorer` is called by typing `vExplorer()`

3 vExplorer

`vExplorer` allows users to explore the vignettes for a given R package by taking the advantages of the functionalities provided by `Sweave`. When `vExplorer` is invoked without a package name as shown in Figure 3, the names of all the installed R packages containing vignettes will be shown in the `Package List` list box. When `vExplorer` is invoked with a package name, the `Vignette List` list box will be populated with the names of vignettes of the package.

Clicking a package name in the list box shows the names of all the vignettes of the package in the `Vignette List` list box. When a vignette name is clicked, a new widget as shown by Figure 4 appears that allows users to perform the following tasks if the selected vignettes contains any execute-able code chunks:

- Clicking an active button in the `Code Chunk` text box displays the code chunk in the `R Source Code` text box for users to view.

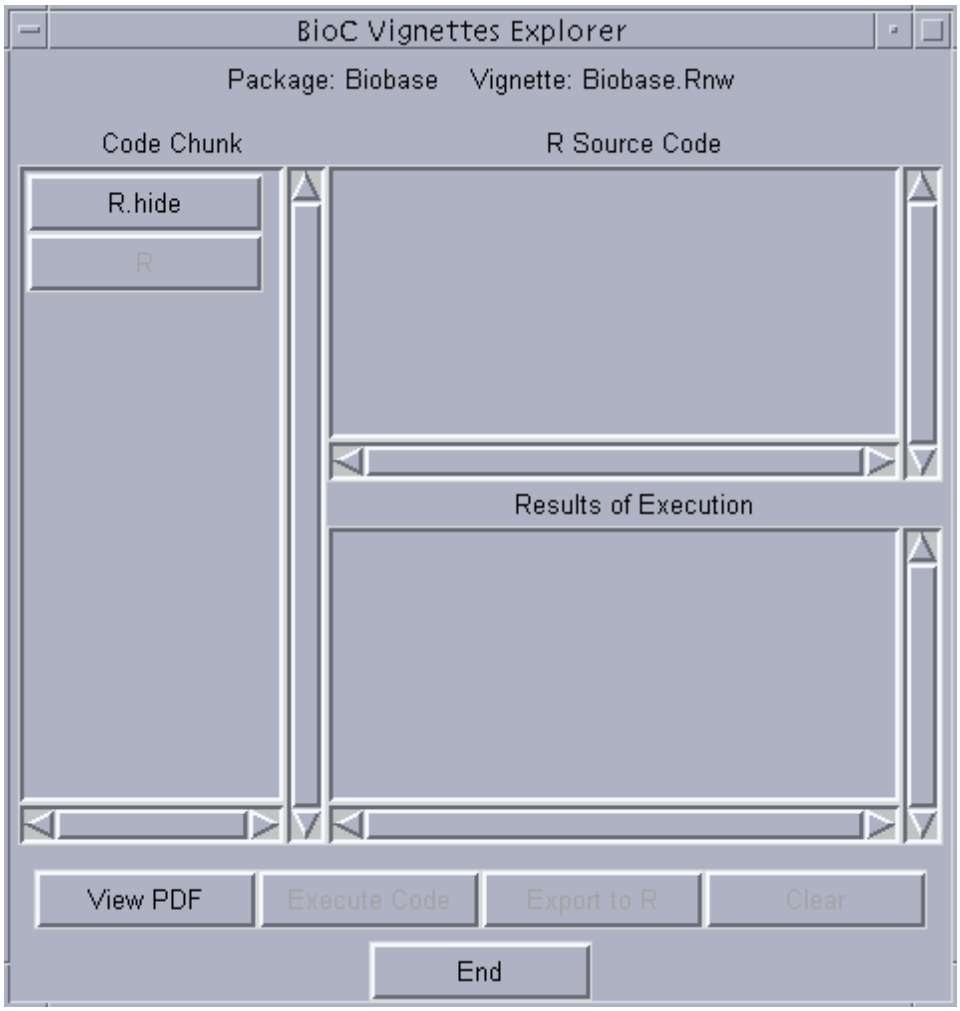

Figure 5: A snapshot of the widget when `Biobase.Rnw` of the *Biobase* package was clicked

- Clicking the `View PDF` buttons shows the pdf version of the vignette currently being explored.
- Clicking the `Execute Code` button executes the code shown in the text box and displays the results of the execution in `Results of Execution`.
- Clicking the `Export to R` button export the result of execution to the global environment of the current R session.

The evaluation model for code chunks in a vignette is that they are evaluated sequentially. In order to ensure sequential evaluation and to help guide the user in terms of which code chunk is active we inactivate all buttons except the one associated with the first unevaluated code chunk.

There are other widgets that are used by specific packages that may not be of general interest. These widgets are thus not included in the examples.

4 Session Information

The version number of R and packages loaded for generating the vignette were:

- R version 4.4.1 (2024-06-14), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Time zone: `Etc/UTC`
- TZcode source: `system (glibc)`
- Running under: `Ubuntu 24.04 LTS`
- Matrix products: default
- BLAS: `/usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3`
- LAPACK: `/usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.26.so` ; LAPACK version3.12.0
- Base packages: base, datasets, grDevices, graphics, methods, stats, tcltk, tools, utils
- Other packages: DynDoc 1.83.0, tkWidgets 1.83.0, widgetTools 1.83.0
- Loaded via a namespace (and not attached): buildtools 1.0.0, compiler 4.4.1, knitr 1.48, maketools 1.3.0, sys 3.4.2, xfun 0.46