`trimmer` 0.7.5 is now available on CRAN.

`trimmer` is a lightweight toolkit to trim a (potentially big) R object without breaking the results of a given function call, where the (trimmed) R object is given as argument.

The `trim` function is the bread and butter of `trimmer`. It seeks to reduce the size of an R object by recursively removing elements from the object one-by-one. It does so in a 'greedy' fashion – it constantly tries to remove the element that uses the most memory.

The trimming process is constrained by a reference function call. The trimming procedure will not allow elements to be removed from the object, that will cause results from the function call to diverge from the original results of the function call.

Motivation

There can be many data reasons as to why, you might want to 'trim' an R object.

A typical example could be a R model object. It will typically contain all kinds of (more or less useful) stuff and meta data with information about the model. You might want to try to reduce the size of the object for (memory) efficiency purposes, such that the model only contains only what is in fact needed to predict new observations – and *nothing* else!

Installation

Install the development version of `trimmer` with:

```
remotes::install_github("smaakage85/trimmer")
```

Or install the version released on CRAN:

```
install.packages("trimmer")
```

Trimming Process

The trimming procedure – conducted with `trim()` – consists of the following steps:

1. Call the specified function with the object before trimming and save results for reference.
2. Compute size of elements in the most shallow layer of object. These elements are the candidates for elimination.
3. Identify the candidate element that uses most memory.
4. Call function again, but this time with the element from step 3 removed from object.
5. If results from function call are the same as in step 1, remove element from object. If results diverge, keep element – and expand the list with candidates for elimination with elements of most shallow layer of this object (if there are any).
6. Repeat steps 3 to 5, until target size is reached or until no further elements can be removed without results from function call diverge.

Workflow Example

Get ready by loading the package.

```
library(trimmer)
```

Train a model on the famous `mtcars` data set.

```
# load training data.
trn <- datasets::mtcars

# estimate model.
mdl <- lm(mpg ~ ., data = trn)
```

I want to trim the model object `mdl` as possible without affecting the predictions, computed with function `predict()`, for the resulting model.

The trimming is then simply conducted by invoking:

```
mdl_trim <- trim(obj = mdl,
                 obj_arg_name = "object",
                 fun = predict,
                 newdata = trn)
#> * Initial object size: 22.22 kB
#> Begin trimming object.
#> ~ Trying to remove element [[c('model')]], element size = 14.05 kB
#> v Element removed.
#> * Object size after removal: 18.19 kB [v4.03 kB]
#> ~ Trying to remove element [[c('qr')]], element size = 7.79 kB
#> x Element could not be removed.
#> ~ Trying to remove element [[c('terms')]], element size = 7.63 kB
#> x Element could not be removed.
#> ~ Trying to remove element [[c('qr','qr')]], element size = 6.66 kB
#> v Element removed.
#> * Object size after removal: 14.95 kB [v7.27 kB]
#> ~ Trying to remove element [[c('residuals')]], element size = 2.86 kB
#> v Element removed.
#> * Object size after removal: 14.53 kB [v7.7 kB]
#> ~ Trying to remove element [[c('fitted.values')]], element size = 2.86 kB
#> v Element removed.
#> * Object size after removal: 11.66 kB [v10.56 kB]
#> ~ Trying to remove element [[c('effects')]], element size = 1.4 kB
#> v Element removed.
#> * Object size after removal: 10.76 kB [v11.46 kB]
#> ~ Trying to remove element [[c('coefficients')]], element size = 1.09 kB
#> x Element could not be removed.
#> ~ Trying to remove element [[c('call')]], element size = 728 B
#> v Element removed.
#> * Object size after removal: 10.09 kB [v12.14 kB]
#> ~ Trying to remove element [[c('xlevels')]], element size = 208 B
#> v Element removed.
#> * Object size after removal: 9.85 kB [v12.38 kB]
#> ~ Trying to remove element [[c('qr','qraux')]], element size = 176 B
#> v Element removed.
#> * Object size after removal: 9.62 kB [v12.61 kB]
#> ~ Trying to remove element [[c('assign')]], element size = 96 B
#> v Element removed.
#> * Object size after removal: 9.46 kB [v12.76 kB]
#> ~ Trying to remove element [[c('qr','pivot')]], element size = 96 B
#> x Element could not be removed.
#> ~ Trying to remove element [[c('rank')]], element size = 56 B
#> x Element could not be removed.
#> ~ Trying to remove element [[c('df.residual')]], element size = 56 B
#> v Element removed.
```

```
#> * Object size after removal: 9.31 kB [v12.91 kB]
#> ~ Trying to remove element [[c('qr','tol')]], element size = 56 B
#> v Element removed.
#> * Object size after removal: 9.17 kB [v13.06 kB]
#> Trimming completed.
```

And that's it!

Note, that I provide the `trim` function with the extra argument `newdata`, that is passed to the function call with `fun`. This means, that the trimming is constrained by, that the results of 'fun' (=`predict`) *MUST* be exactly the same on these data before and after the trimming.

The trimmed model object now measures 9.17 kB. The original object measured 22.22 kB.

Set Target Size

If you just want the object size to be below some threshold, you can set that as a criterion. The 'trimming' process will continue no further, when this threshold is reached. This approach can be time-saving compared to minimizing the object as much as possible (=default setting).

```
mdl_trim <- trim(obj = mdl,
                 obj_arg_name = "object",
                 fun = predict,
                 newdata = trn,
                 size_target = 0.015)
#> * Initial object size: 22.22 kB
#> * Target object size: <= 15 kB
#> Begin trimming object.
#> ~ Trying to remove element [[c('model')]], element size = 14.05 kB
#> v Element removed.
#> * Object size after removal: 18.19 kB [v4.03 kB]
#> ~ Trying to remove element [[c('qr')]], element size = 7.79 kB
#> x Element could not be removed.
#> ~ Trying to remove element [[c('terms')]], element size = 7.63 kB
#> x Element could not be removed.
#> ~ Trying to remove element [[c('qr','qr')]], element size = 6.66 kB
#> v Element removed.
#> * Object size after removal: 14.95 kB [v7.27 kB]
#> Trimming completed.
```

With these settings, the trimmed model object measures 14.95 kB. The original object measured 22.22 kB.

Other Applications

`trimmer` is compatible with all R objects, that inherit from the `list` class – not just R model objects – and all kinds of functions – not just the `predict` function. Hence `trimmer` is quite a flexible tool.

To illustrate I will trim the same object but under the constraint, that the results from the `summary()` function must be preserved.

```
mdl_trim <- trim(obj = mdl,
                 obj_arg_name = "object",
                 fun = summary)
#> * Initial object size: 22.22 kB
#> Begin trimming object.
```

```
#> ~ Trying to remove element [[c('model')]], element size = 14.05 kB
#> v Element removed.
#> * Object size after removal: 18.19 kB [v4.03 kB]
#> ~ Trying to remove element [[c('qr')]], element size = 7.79 kB
#> x Element could not be removed.
#> ~ Trying to remove element [[c('terms')]], element size = 7.63 kB
#> x Element could not be removed.
#> ~ Trying to remove element [[c('qr','qr')]], element size = 6.66 kB
#> x Element could not be removed.
#> ~ Trying to remove element [[c('residuals')]], element size = 2.86 kB
#> x Element could not be removed.
#> ~ Trying to remove element [[c('fitted.values')]], element size = 2.86 kB
#> x Element could not be removed.
#> ~ Trying to remove element [[c('effects')]], element size = 1.4 kB
#> v Element removed.
#> * Object size after removal: 17.42 kB [v4.81 kB]
#> ~ Trying to remove element [[c('coefficients')]], element size = 1.09 kB
#> x Element could not be removed.
#> ~ Trying to remove element [[c('call')]], element size = 728 B
#> x Element could not be removed.
#> ~ Trying to remove element [[c('xlevels')]], element size = 208 B
#> v Element removed.
#> * Object size after removal: 17.21 kB [v5.02 kB]
#> ~ Trying to remove element [[c('qr','qraux')]], element size = 176 B
#> v Element removed.
#> * Object size after removal: 16.94 kB [v5.28 kB]
#> ~ Trying to remove element [[c('assign')]], element size = 96 B
#> v Element removed.
#> * Object size after removal: 16.76 kB [v5.46 kB]
#> ~ Trying to remove element [[c('qr','pivot')]], element size = 96 B
#> x Element could not be removed.
#> ~ Trying to remove element [[c('rank')]], element size = 56 B
#> x Element could not be removed.
#> ~ Trying to remove element [[c('df.residual')]], element size = 56 B
#> x Element could not be removed.
#> ~ Trying to remove element [[c('qr','tol')]], element size = 56 B
#> v Element removed.
#> * Object size after removal: 16.65 kB [v5.58 kB]
#> Trimming completed.
```

Other Notes

You can choose, that certain elements *MUST NOT* be removed during the trimming process. Do this with the `dont_touch` argument.

You can choose whether or not to tolerate warnings from reference function calls with the argument `tolerate_warnings`.