ALTREP: Alternate Representations of Basic R Objects

Luke Tierney

Department of Statistics & Actuarial Science
University of Iowa

December 14, 2017

Introduction

- My first visit to New Zealand was for NZSA 1997 to present work on MCMC methodology and interactive graphics software.
- Sometime after this visit Ross and Robert added me to their mailing list and gave me write access to the R CVS archive
- It took a year or two before I started actively contributing, but has been a major focus for me ever since.
- I have worked mostly on computational infrastructure, such as
  - memory management
  - name space management
  - error handling framework
  - compilation
  - parallel computing support
- Much of this is enabling technology not used directly by typical users or only by package authors.

Introduction

- Today's talk is about joint work with Gabe Becker, Tomas Kalibera on another such technology: alternate representations for basic R objects.
- The C level R implementation works with a fixed set of data types, e.g. INTSXP, REALSXP, ENVSXP.
- These have a particular memory layout, but are accessed only through a function/macro abstraction.
- For vector data the accessors are
  - LENGTH for the number of elements;
  - DATAPTR for a pointer to a contiguous region in memory.
- The memory is typically allocated by malloc

Introduction

- ALTREP allows for alternate representations of these data types.
- Some examples of things we want to enable:
  - allow vector data to be in a memory-mapped file or distributed;
  - allow compact representation of arithmetic sequences;
  - allow adding meta-data to objects;
  - allow computations/allocations to be deferred;
  - support alternative representations of environments.
- To existing C code ALTREP objects look like ordinary R objects.
- Updated C code may be able to take advantage of special features.
- Current state is available in the ALTREP SVN branch.
- More details are available in ALTREP.html at the branch root.

Example: Compact Integer Sequences

- Vectors created by n1:n2, seq_along or seq_len can be represented compactly.
- In 3.4.x with JIT disabled:

```
system.time(for (i in 1:1e9) break)
##   user  system elapsed
##  0.258   1.141   1.400
```

- In the ALTREP branch:

```
system.time(for (i in 1:1e9) break)
##   user  system elapsed
##  0   0.004   0.000   0.003
```

Example: Compact Integer Sequences

- In 3.4.x creating a larger sequence may fail:

```
x <- 1:1e10
## Error: cannot allocate vector of size 74.5 Gb
```

- In the ALTREP branch this succeeds:

```
x <- 1:1e10
length(x)
## [1] 1e+10
```

- Some operations may fail fail:

```
y <- x + 1L
## Error: cannot allocate vector of size 74.5 Gb
```

Example: Deferred String Conversions

- Converting integers or reals to strings is expensive.
- In lm and glm default row labels on design matrices are created but rarely used.
- The ALTREP branch
  - modifies the internal coerce function to return a *deferred string conversion* object;
  - this class has a subset method that returns another deferred conversion object.

Example: Deferred String Conversions

- For lm with $n = 10^7$ and $p = 2$:

```
x <- rnorm(1e7)
y <- x + rnorm(1e7)
system.time(lm(y ~ x))
##    user  system elapsed
## 19.804   0.860  20.703   R 3.4.2 patched
##  1.960   1.184   3.147   ALTREP
```

- For glm:

```
system.time(glm(y ~ x))
##    user  system elapsed
## 20.880   1.624  22.517    R 3.4.2 patched
##  6.144   5.508  11.657    ALTREP
```

- Deferred evaluation could be useful in many other settings as well.

Example: Wrapper Objects and Meta-Data

- Currently changing an attribute on a shared vector requires a copy of the vector data.
- Wrapper objects can hold the new attribute value and a reference to the original object to access its data.
- Wrapper objects can also be used to attach meta-data, such as
  - is the vector sorted;
  - are there no NA values.
- The sort function returns a wrapper that records that the vector is sorted and whether there are no NA values.

Example: Wrapper Objects and Meta-Data

- Sorting a large vector takes some time:

```
x <- rnorm(1e8)
system.time(y <- sort(x, method = "shell"))
##    user  system elapsed
## 23.652   0.108  23.762
```

- The result y is known to be sorted:

```
system.time(sort(y, method = "shell"))
##    user  system elapsed
##   0.220   0.060   0.281
```

- The sorting process reveals that there are no NA values, so this is recorded in the result y and used by anyNA:

```
system.time(anyNA(x))
##    user  system elapsed
##   0.136   0.000   0.136
system.time(anyNA(y))
##    user  system elapsed
##       0       0       0
```

Example: Wrapper Objects and Meta-Data

- Compact integer sequences also carry meta-data:

```
indx <- seq_along(x)
system.time(anyNA(indx))
##   user  system elapsed
##      0       0       0
system.time(sort(indx))
##   user  system elapsed
##  1.288   0.644   1.932
system.time(sort(indx, method = "shell"))
##   user  system elapsed
##  0.224   0.036   0.260
```

- ALTREP objects can also provide methods for some basic summaries:

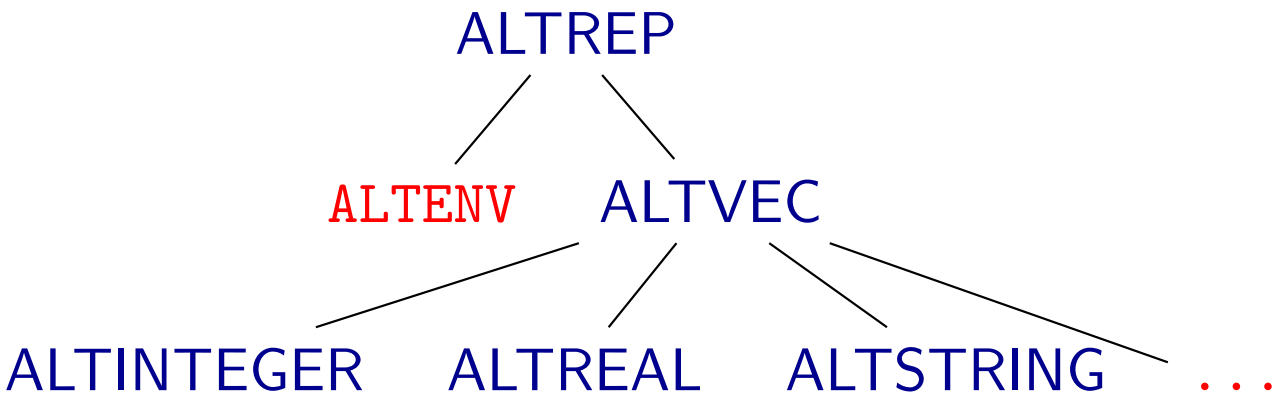
```
system.time(sum(x))
##   user  system elapsed
##  0.176   0.000   0.176
system.time(as.double(indx))
##   user  system elapsed
##      0       0       0
```

Example: Memory Mapped Vectors

- The ALTREP branch includes sample classes for memory mapped integer and real vectors.
- The file can be opened for reading and writing or in read-only mode.
- When used by ALTREP-aware code these will not result in allocating memory for holding all the data.
- Using non-aware functions may result in attempts to allocate large objects.
- The class provides an option for signaling an error when the raw data pointer is requested.
- A variant is also available as a small experimental package `simplemmap`.

Abstract Classes

- The framework is designed around a set of *abstract classes*.
- A set of abstract classes for R data types:



- The most specific classes correspond to R data types.
- Concrete classes specialize one of these.
- Each abstract class level defines a set of methods.
- Each concrete class has a table of method implementations.

- ALTREP object methods:
  - Duplicate
  - Coerce
  - Length
  - Inspect
- The standard macros defer to these methods for ALTREP objects.
- Duplicate and Coerce methods can return NULL to fall back to the default behavior.

- ALTVEC methods:
  - Dataptr
  - Dataptr_or_null
  - Extract_subset
  - `Extract_subarray`
- Dataptr may need to allocate memory; for now GC is suspended when calling the method.
- Dataptr_or_null will not allocate.
- Dataptr_or_null and Extract_subset can be used to avoid fully allocating an object.
- Adding `Extract_subarray` will help for interfacing to structured storage systems.

Methods
Specific Vector Methods

- Specific vector methods (patterned after JNI):
  - Elt
  - Set_elt
  - Get_region
  - No_NA
  - Is_sorted
  - and several others.
- Some numeric vector methods:
  - Min
  - Max
  - Sum
  - Prod
- A single method for extracting properties specified by a bitmask might be useful.

Changes to Existing Functions

- Existing functions will work without modification.
- But by using the DATAPTR they may cause allocation or reading of full data that can be avoided.
- Some functions modified to avoid using DATAPTR:
  - mean
  - min
  - max
  - sum
  - prod.
- These use Get_region to process data in chunks.
- Many more functions could be modified along these lines.

Changes to Existing Functions

- Subsetting has also been modified to avoid using DATAPTR.
- This means head, sample, for example, avoid allocation:

```
x <- 1:1e12
length(x)
## [1] 1e+12
head(x)
## [1] 1 2 3 4 5 6
> sample(x, 10)
## [1] 736617330192 392069636550 568241239321 224393184527
## [5] 851984238988 174365872796 366347672451  84457266227
## [9]  72327203393 761965661188
```

- Other operations attempt to allocate and fail:

```
x + 1
## Error: cannot allocate vector of size 7450.6 Gb
log(x)
## Error: cannot allocate vector of size 7450.6 Gb
```

Serialization and Package Support

- Classes can provide custom serialization by defining methods for
  - Serialized_state
  - Unserialize
- Packages can register ALTREP classes.
- Serialization records the package and class name.
- Unserializing loads the package namespace and looks up the registered class.
- A sample package implementing a memory mapped vector object is available on GitHub.
- Custom serialization requires a bump in the serialization version:
  - Older R versions cannot handle custom serializations; bumping the format version gives a clearer error message.
  - Some packages that make assumptions about the serialization format may need updates (e.g. `digest`).
  - This provides an opportunity for some other changes (e.g. recording native encoding information).

Some Implementation Details

- ALTREP objects are allocated as CONS cells with an altrep header bit set.
- Standard macros, like LENGTH look at this bit to decide whether to dispatch.
- To allow efficient scalar identification there is also a scalar bit,
- With the ALTREP changes, operations like DATAPTR, STRING_ELT, and SET_STRING_ELT now might cause allocation.
- Eventually code should be rewritten to allow for this.
- For now, GC is suspended in these allocations.

Some Issues and Notes

- Performance can suffer due to:
  - overhead of checking altrep bit for standard objects;
  - dispatching overhead for ALTREP objects.
- Accessing the DATAPTR and possibly allocating may sometimes be much faster.
- Switching to an ALTREP may only pay off if objects are large.
- Deferred evaluations/allocations are very useful, but:
  - allocation failures can be delayed and come at unexpected times;
  - operations may produce unexpected large allocations, e.g. log(1:1e10);
  - some situations can lead to repeated evaluations.
- Memory mapping issues:
  - unserialization failure when the file is not available;
  - some settings might need a conversion layer (e.g. a file of 8-bit integers).
- Deferred edits might be useful for improving complex assignment performance.

Current Status

- ALTREP will hopefully be fully incorporated into R 3.5.0 in 2018.
- The basic framework is now in R-devel, but
  - it is still subject to change;
  - no ALTREP objects are generated yet.
- A necessary change to the internal object structure requiring packages using compiled code to be rebuilt occurred in September 2017.
- This change also reserves 64 bits for vector sizes on 64 bit platforms, which simplifies large vector support.
- The next step will be to incorporate creation of ALTREP object into base code:
  - compact integer sequences;
  - deferred string conversions;
  - meta-data wrappers.
- This will be done after further testing on CRAN and BIOC packages.
- Further performance testing and tuning is also needed.

Conclusions

- The ALTREP changes are evolutionary:
  - Existing code should continue to work.
  - Performance overhead should be minimal.
- The framework should help to
  - allow experimentation with some new ideas;
  - regularize some things currently being done.
- R internals have evolved considerably in the last 20 years.
- The ability to do this is a tribute to the original design Ross and Robert put together.