Formatting of column data

Kirill Müller

2016-08-17

The presentation of a column in a tibble is powered by the S3 generics `type_sum()` and, to a limited extent, `obj_sum()`. The former determines what goes into the column header, the latter is mostly responsible for rendering list columns. Package authors who provide S3 or S4 classes that can be used as a column in a data frame can override these generics to make sure their data is represented ideally in a tibble. This vignette assumes a package that implements an S3 class `"foo"` and uses `roxygen2` to create documentation and the `NAMESPACE` file.

A prerequisite for overriding the generics is to import the `tibble` package. Add `tibble` to the `Imports:` section of your `DESCRIPTION`, or simply call

```
devtools::use_package("tibble")
```

in your package directory.

`type_sum()`

Implementations of this method is expected to return an atomic character value that can be used in a column header. Examples:

```
type_sum(1)
```

```
## [1] "dbl"
```

```
type_sum(1:10)
```

```
## [1] "int"
```

```
type_sum(Sys.time())
```

```
## [1] "dttm"
```

```
type_sum(structure(1:10, class = "unknown"))
```

```
## [1] "S3: unknown"
```

The default implementation works reasonably well for any kind of object, but the generated output may be too wide and waste precious space when displaying the tibble. To avoid this for `foo`, you'd write

```
#' @export
```

```
type_sum.foo <- function(x, ...) {
  "foo"
}
```

obj_sum()

This method is primarily used for displaying list columns. A list column is a powerful way to attach hierarchical or unstructured data to an observation in a data frame. Implementations of `obj_sum()` are expected to return a character vector as long as the input, with brief description of the contents of each input element. Examples:

```
obj_sum(1)
```

```
## [1] "dbl [1]"
```

```
obj_sum(1:10)
```

```
## [1] "int [10]"
```

```
obj_sum(Sys.time())
```

```
## [1] "dttm [1]"
```

```
obj_sum(list(1:5))
```

```
## [1] "int [5]"
```

```
obj_sum(as.list(1:5))
```

```
## [1] "int [1]" "int [1]" "int [1]" "int [1]" "int [1]"
```

The default implementation calls `type_sum()` and appends the size of the object in brackets. If `foo` encapsulates a simple vector, or a list, the default should be fine, and you don't need to override `obj_sum()`. An example for a necessary override is the included implementation for the `POSIXlt` class. Recall that objects of this class are stored as named lists where each component has the same length. Because of this, using objects of this class is not recommended in a tibble or data frame, and the summary method just returns the same text without revealing the data:

```
tibble:::obj_sum.POSIXlt
```

```
## function(x) {
##   rep("POSIXlt", length(x))
## }
## <environment: namespace:tibble>
```