Regular expressions and reshaping using data tables and the nc package

by Toby Dylan Hocking

Abstract Regular expressions are powerful tools for extracting tables from non-tabular text data. Capturing regular expressions that describe information to extract from column names can be especially useful when reshaping a data table from wide (one row with many columns) to tall (one column with many rows). We present the R package **nc**, which provides functions for data reshaping, regular expressions, and a uniform interface to three C libraries (PCRE, RE2, ICU). We describe the main features of **nc**, then provide detailed comparisons with related R packages (**stats**, **utils**, **data.table**, **tidyr**, **reshape2**, **cdata**).

Introduction

Regular expressions are powerful tools for text processing that are available in many programming languages, including R. A regular expression *pattern* defines a set of *matches* in a *subject* string. For example, the pattern `.*[.].*` matches zero or more non-newline characters, followed by a period, followed by zero or more non-newline characters. It would match the subjects `Sepal.Length` and `Petal.Width`, but it would not match in the subject `Species`.

The focus of this article is patterns with capture groups, which are typically defined using parentheses. For example, the pattern `(.*)[.](.*)` results in the same matches as the pattern in the previous paragraph, and it additionally allows the user to capture and extract the substrings by group index (e.g. group 1 matches `Sepal`, group 2 matches `Length`).

Named capture groups allow extracting the a substring by name rather than by index. Using names rather than indices is useful in order to create more readable regular expressions (names document the purpose of each sub-pattern), and to create more readable R code (it is easier to understand the intent of named references than numbered references). For example, the pattern `(?<part>.*)[.](?<dimension>.*)` documents that the flower part appears before the measurement dimension; the `part` group matches `Sepal` and the `dimension` group matches `Length`.

Recently, Hocking (2019) proposed a new syntax for defining named capture groups in R code. Using this new syntax, the pattern in the previous paragraph can be written as `part=".*","[.]",dimension=".*"`.

In this article our original contribution is the R package **nc** which provides a new implementation of the previously proposed syntax for named capture regular expressions, in addition to several new features for data reshaping. The main new ideas are (1) using un-named groups to provide a uniform interface to three regex C libraries, (2) integration of regex and **data.table** functionality, and (3) specifying wide-to-tall reshape operations with a new syntax which results in less repetitive user code than other packages.

The organization of this article is as follows. The rest of this introduction provides an overview of current R packages for regular expressions and data reshaping. The second section describes the proposed functions of the **nc** package. The third section provides detailed comparisons with other R packages, in terms of syntax and computation times. The article concludes with a summary and discussion.

Related work

```
> nc::capture_melt_single

function (subject.df, ..., id.vars = NULL, variable.name = "variable",
    value.name = "value", na.rm = FALSE, verbose = getOption("datatable.verbose"))
{
    if (!is.data.frame(subject.df)) {
        stop("subject must be a data.frame")
    }
    variable <- names(subject.df)
    match.dt <- capture_first_vec(variable, ..., nomatch.error = FALSE)
    no.match <- apply(is.na(match.dt), 1, all)
    if (all(no.match)) {
        stop("no column names match regex below\n", var_args_list(...)$pattern)
```

pkg::function	single	multiple	regex	na.rm	types	list
nc::capture_melt_multiple	no	unsorted	capture	yes	any	yes
nc::capture_melt_single	yes	no	capture	yes	any	yes
tidyr::pivot_longer	yes	unsorted	capture	yes	some	yes
stats::reshape	yes	sorted	match	no	no	no
data.table::melt,patterns	yes	sorted	match	yes	no	yes
tidyr::gather	yes	no	no	yes	some	yes
reshape2::melt	yes	no	no	yes	no	no
cdata::rowrecs_to_blocks	yes	unsorted	no	no	no	yes
utils::stack	yes	no	no	no	no	no

Table 1: Reshaping functions in R support various features: "single" for converting input columns into a single output column; "multiple" for converting input columns (either "sorted" in a regular order, or "unsorted" for any order) into multiple output columns of different types; "regex" for regular expressions to only "match" input column names or to "capture" and create new output column names; "na.rm" for removal of missing values; "types" for converting input column names to non-character output columns; "list" for output of list columns.

```
    }
    names.dt <- data.table(variable, match.dt)[!no.match]
    if (is.null(id.vars)) {
        id.vars <- which(no.match)
    }
    tall.dt <- melt(data.table(subject.df), id.vars = id.vars,
        measure.vars = which(!no.match), variable.name = variable.name,
        value.name = value.name, na.rm = na.rm, variable.factor = FALSE,
        value.factor = FALSE, verbose = verbose)
    on.vec <- structure("variable", names = variable.name)
    tall.dt[names.dt, on = on.vec]
}
<bytecode: 0x22545f8>
<environment: namespace:nc>
attr(,"ex")
function ()
{
    library(data.table)
    iris.dt <- data.table(observation = 1:nrow(iris), iris)
    (iris.tall <- nc::capture_melt_single(iris.dt, part = ".*",
        "[.]", dim = ".*"))
    (iris.part.cols <- dcast(iris.tall, observation + Species +
        dim ~ part))
    iris.part.cols[Sepal < Petal]
    (iris.dim.cols <- dcast(iris.tall, observation + Species +
        part ~ dim))
    iris.dim.cols[Length < Width]
}
<environment: namespace:nc>
```

Reproducible research statement. The source code for this article can be freely downloaded from https://github.com/tdhock/nc-article

Bibliography

T. D. Hocking. Comparing namedcapture with other r packages for regular expressions. *R Journal*, 2019. [p1]

Toby Dylan Hocking
School of Informatics, Computing, and Cyber Systems
Northern Arizona University

Flagstaff, Arizona
USA
toby.hocking@nau.edu