Reference semantics

2015-06-24

This vignette discusses *data.table*'s reference semantics which allows to *add/update/delete* columns of a *data.table by reference*, and also combine them with `i` and `by`. It is aimed at those who are already familiar with *data.table* syntax, its general form, how to subset rows in `i`, select and compute on columns, and perform aggregations by group. If you're not familar with these concepts, please read the *"Introduction to data.table"* vignette first.

Data

We will use the same `flights` data as in the *"Introduction to data.table"* vignette.

```
flights <- fread("https://raw.githubusercontent.com/wiki/arunsrinivasan/flights/NYCflights14/flights14.csv")
flights
#         year month day dep_time dep_delay arr_time arr_delay cancelled carrier tailnum flight
#      1: 2014     1   1      914        14     1238        13         0      AA  N338AA      1
#      2: 2014     1   1     1157        -3     1523        13         0      AA  N335AA      3
#      3: 2014     1   1     1902         2     2224         9         0      AA  N327AA     21
#      4: 2014     1   1      722        -8     1014       -26         0      AA  N3EHAA     29
#      5: 2014     1   1     1347         2     1706         1         0      AA  N319AA    117
#     ---
# 253312: 2014    10  31     1459         1     1747       -30         0      UA  N23708   1744
# 253313: 2014    10  31      854        -5     1147       -14         0      UA  N33132   1758
# 253314: 2014    10  31     1102        -8     1311        16         0      MQ  N827MQ   3591
# 253315: 2014    10  31     1106        -4     1325        15         0      MQ  N511MQ   3592
# 253316: 2014    10  31      824        -5     1045         1         0      MQ  N813MQ   3599
#         origin dest air_time distance hour min
#      1:    JFK  LAX      359     2475    9  14
#      2:    JFK  LAX      363     2475   11  57
#      3:    JFK  LAX      351     2475   19   2
#      4:    LGA  PBI      157     1035    7  22
#      5:    JFK  LAX      350     2475   13  47
#     ---
# 253312:    LGA  IAH      201     1416   14  59
# 253313:    EWR  IAH      189     1400    8  54
# 253314:    LGA  RDU       83      431   11   2
# 253315:    LGA  DTW       75      502   11   6
# 253316:    LGA  SDF      110      659    8  24
dim(flights)
# [1] 253316     17
```

Introduction

In this vignette, we will

1. first discuss reference semantics briefly and look at the two different forms in which the `:=` operator can be used
2. then see how we can *add/update/delete* columns *by reference* in `j` using the `:=` operator and how to combine with `i` and `by`.
3. and finally we will look at using `:=` for its *side-effect* and how we can avoid the side effects using `copy()`.

1. Reference semantics

All the operations we have seen so far in the previous vignette resulted in a new data set. We will see how to *add* new column(s), *update* or *delete* existing column(s) on the original data.

a) Background

Before we look at *reference semantics*, consider the *data.frame* shown below:

```
DF = data.frame(ID = c("b","b","b","a","a","c"), a = 1:6, b = 7:12, c=13:18)
DF
#   ID a  b  c
# 1  b 1  7 13
# 2  b 2  8 14
# 3  b 3  9 15
# 4  a 4 10 16
# 5  a 5 11 17
# 6  c 6 12 18
```

When we did:

```
DF$c <- 18:13              # (1) -- replace entire column
# or
DF$c[DF$ID == "b"] <- 15:13 # (2) -- subassign in column 'c'
```

both (1) and (2) resulted in *deep* copy of the entire *data.frame* in versions of `R` versions `< 3.1`. It copied more than once. To improve performance by avoiding these redundant copies, *data.table* utilised the available but unused `:=` operator in R.

Great performance improvements were made in `R v3.1` as a result of which only a *shallow* copy is made for (1) and not *deep* copy. However, for (2) still, the entire column is *deep* copied even in `R v3.1+`. This means the more columns one subassigns to in the *same query*, the more *deep* copies R does.

> ### *shallow* vs *deep* copy
> A *shallow* copy is just a copy of the vector of column pointers (corresponding to the columns in a *data.frame* or *data.table*). The actual data is not physically copied in memory.
>
> A *deep* copy on the other hand copies the entire data to another location in memory.

With *data.table's* `:=` operator, absolutely no copies are made in *both* (1) and (2), irrespective of R version you are using. This is because `:=` operator updates *data.table* columns *in-place* (by reference).

b) The `:=` operator

It can be used in `j` in two ways:

a. The `LHS := RHS` form

```
DT[, c("colA", "colB", ...) := list(valA, valB, ...)]

# when you have only one column to assign to you
# can drop the quotes and list(), for convenience
DT[, colA := valA]
```

b. The functional form

```
DT[, `:=`(colA = valA, # valA is assigned to colA
          colB = valB, # valB is assigned to colB
          ...
)]
```

> Note that the code above explains how `:=` can be used. They are not working examples. We will start using them on `flights` *data.table* from the next section.

> - Form (a) is usually easy to program with and is particularly useful when you don't know the columns to assign values to in advance.
> - On the other hand, form (b) is handy if you would like to jot some comments down for later.
> - The result is returned *invisibly*.
> - Since `:=` is available in `j`, we can combine it with `i` and `by` operations just like the aggregation operations we saw in the previous vignette.

In the two forms of `:=` shown above, note that we don't assign the result back to a variable. Because we don't need to. The input

data.table is modified by reference. Let's go through examples to understand what we mean by this.

For the rest of the vignette, we will work with `flights` *data.table*.

2. Add/update/delete columns *by reference*

a) Add columns by reference

– How can we add columns *speed* and *total delay* of each flight to `flights` *data.table*?

```
flights[, `:=`(speed = distance / (air_time/60), # speed in km/hr
               delay = arr_delay + dep_delay)]   # delay in minutes
head(flights)
#    year month day dep_time dep_delay arr_time arr_delay cancelled carrier tailnum flight origin
# 1: 2014     1   1      914        14     1238        13         0      AA  N338AA      1    JFK
# 2: 2014     1   1     1157        -3     1523        13         0      AA  N335AA      3    JFK
# 3: 2014     1   1     1902         2     2224         9         0      AA  N327AA     21    JFK
# 4: 2014     1   1      722        -8     1014       -26         0      AA  N3EHAA     29    LGA
# 5: 2014     1   1     1347         2     1706         1         0      AA  N319AA    117    JFK
# 6: 2014     1   1     1824         4     2145         0         0      AA  N3DEAA    119    EWR
#    dest air_time distance hour min    speed delay
# 1:  LAX      359     2475    9  14 413.6490    27
# 2:  LAX      363     2475   11  57 409.0909    10
# 3:  LAX      351     2475   19   2 423.0769    11
# 4:  PBI      157     1035    7  22 395.5414   -34
# 5:  LAX      350     2475   13  47 424.2857     3
# 6:  LAX      339     2454   18  24 434.3363     4

## alternatively, using the 'LHS := RHS' form
# flights[, c("speed", "delay") := list(distance/(air_time/60), arr_delay + dep_delay)]
```

> Note that
>
> - We did not have to assign the result back to `flights`.
> - The `flights` *data.table* now contains the two newly added columns. This is what we mean by *added by reference*.
> - We used the functional form so that we could add comments on the side to explain what the computation does. You can also see the `LHS := RHS` form (commented).

b) Update some rows of columns by reference - *sub-assign* by reference

Let's take a look at all the `hours` available in the `flights` *data.table*:

```
# get all 'hours' in flights
flights[, sort(unique(hour))]
#  [1]  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
```

We see that there are totally `25` unique values in the data. Both *0* and *24* hours seem to be present. Let's go ahead and replace *24* with *0*.

– Replace those rows where `hour == 24` with the value `0`

```
# subassign by reference
flights[hour == 24L, hour := 0L]
```

> - We can use `i` along with `:=` in `j` the very same way as we have already seen in the *"Introduction to data.table"* vignette.
> - Column `hour` is replaced with `0` only on those *row indices* where the condition `hour == 24L` specified in `i` evaluates to `TRUE`.
> - `:=` returns the result invisibly. Sometimes it might be necessary to see the result after the assignment. We can accomplish that by adding an empty `[]` at the end of the query as shown below:

```
flights[hour == 24L, hour := 0L][]
#         year month day dep_time dep_delay arr_time arr_delay cancelled carrier tailnum flight
#      1: 2014     1   1      914        14     1238        13         0      AA  N338AA      1
#      2: 2014     1   1     1157        -3     1523        13         0      AA  N335AA      3
#      3: 2014     1   1     1902         2     2224         9         0      AA  N327AA     21
#      4: 2014     1   1      722        -8     1014       -26         0      AA  N3EHAA     29
#      5: 2014     1   1     1347         2     1706         1         0      AA  N319AA    117
#     ---
# 253312: 2014    10  31     1459         1     1747       -30         0      UA  N23708   1744
# 253313: 2014    10  31      854        -5     1147       -14         0      UA  N33132   1758
# 253314: 2014    10  31     1102        -8     1311        16         0      MQ  N827MQ   3591
# 253315: 2014    10  31     1106        -4     1325        15         0      MQ  N511MQ   3592
# 253316: 2014    10  31      824        -5     1045         1         0      MQ  N813MQ   3599
#         origin dest air_time distance hour min    speed delay
#      1:    JFK  LAX      359     2475    9  14 413.6490    27
#      2:    JFK  LAX      363     2475   11  57 409.0909    10
#      3:    JFK  LAX      351     2475   19   2 423.0769    11
#      4:    LGA  PBI      157     1035    7  22 395.5414   -34
#      5:    JFK  LAX      350     2475   13  47 424.2857     3
#     ---
# 253312:    LGA  IAH      201     1416   14  59 422.6866   -29
# 253313:    EWR  IAH      189     1400    8  54 444.4444   -19
# 253314:    LGA  RDU       83      431   11   2 311.5663     8
# 253315:    LGA  DTW       75      502   11   6 401.6000    11
# 253316:    LGA  SDF      110      659    8  24 359.4545    -4
```

Let's look at all the `hours` to verify.

```
# check again for '24'
flights[, sort(unique(hour))]
#  [1]  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
```

c) Delete column by reference

– Remove `delay` column

```
flights[, c("delay") := NULL]
head(flights)
#    year month day dep_time dep_delay arr_time arr_delay cancelled carrier tailnum flight origin
# 1: 2014     1   1      914        14     1238        13         0      AA  N338AA      1    JFK
# 2: 2014     1   1     1157        -3     1523        13         0      AA  N335AA      3    JFK
# 3: 2014     1   1     1902         2     2224         9         0      AA  N327AA     21    JFK
# 4: 2014     1   1      722        -8     1014       -26         0      AA  N3EHAA     29    LGA
# 5: 2014     1   1     1347         2     1706         1         0      AA  N319AA    117    JFK
# 6: 2014     1   1     1824         4     2145         0         0      AA  N3DEAA    119    EWR
#    dest air_time distance hour min    speed
# 1:  LAX      359     2475    9  14 413.6490
# 2:  LAX      363     2475   11  57 409.0909
# 3:  LAX      351     2475   19   2 423.0769
# 4:  PBI      157     1035    7  22 395.5414
# 5:  LAX      350     2475   13  47 424.2857
# 6:  LAX      339     2454   18  24 434.3363

## or using the functional form
# flights[, `:=`(delay = NULL)]
```

- Assigning `NULL` to a column *deletes* that column. And it happens *instantly*.
- We can also pass column numbers instead of names in the `LHS`, although it is good programming practice to use column names.
- When there is just one column to delete, we can drop the `c()` and double quotes and just use the column name *unquoted*, for convenience. That is:

```
flights[, delay := NULL]
```

is equivalent to the code above.

d) `:=` along with grouping using `by`

We have already seen the use of `i` along with `:=` in Section 2b. Let's now see how we can use `:=` along with `by`.

– How can we add a new column which contains for each `orig,dest` pair the maximum speed?

```
flights[, max_speed := max(speed), by=.(origin, dest)]
head(flights)
#    year month day dep_time dep_delay arr_time arr_delay cancelled carrier tailnum flight origin
# 1: 2014     1   1      914        14     1238        13         0      AA  N338AA      1    JFK
# 2: 2014     1   1     1157        -3     1523        13         0      AA  N335AA      3    JFK
# 3: 2014     1   1     1902         2     2224         9         0      AA  N327AA     21    JFK
# 4: 2014     1   1      722        -8     1014       -26         0      AA  N3EHAA     29    LGA
# 5: 2014     1   1     1347         2     1706         1         0      AA  N319AA    117    JFK
# 6: 2014     1   1     1824         4     2145         0         0      AA  N3DEAA    119    EWR
#    dest air_time distance hour min    speed max_speed
# 1:  LAX      359     2475    9  14 413.6490  526.5957
# 2:  LAX      363     2475   11  57 409.0909  526.5957
# 3:  LAX      351     2475   19   2 423.0769  526.5957
# 4:  PBI      157     1035    7  22 395.5414  517.5000
# 5:  LAX      350     2475   13  47 424.2857  526.5957
# 6:  LAX      339     2454   18  24 434.3363  518.4507
```

- We add a new column `max_speed` using the `:=` operator by reference.
- We provide the columns to group by the same way as shown in the *Introduction to data.table* vignette. For each group, `max(speed)` is computed, which returns a single value. That value is recycled to fit the length of the group. Once again, no copies are being made at all. `flights` *data.table* is modified *in-place*.
- We could have also provided `by` with a *character vector* as we saw in the *Introduction to data.table* vignette, e.g., `by = c("origin", "dest")`.

e) Multiple columns and `:=`

– How can we add two more columns computing `max()` of `dep_delay` and `arr_delay` for each month, using `.SD`?

```
in_cols  = c("dep_delay", "arr_delay")
out_cols = c("max_dep_delay", "max_arr_delay")
flights[, c(out_cols) := lapply(.SD, max), by = month, .SDcols = in_cols]
head(flights)
#    year month day dep_time dep_delay arr_time arr_delay cancelled carrier tailnum flight origin
# 1: 2014     1   1      914        14     1238        13         0      AA  N338AA      1    JFK
# 2: 2014     1   1     1157        -3     1523        13         0      AA  N335AA      3    JFK
# 3: 2014     1   1     1902         2     2224         9         0      AA  N327AA     21    JFK
# 4: 2014     1   1      722        -8     1014       -26         0      AA  N3EHAA     29    LGA
# 5: 2014     1   1     1347         2     1706         1         0      AA  N319AA    117    JFK
# 6: 2014     1   1     1824         4     2145         0         0      AA  N3DEAA    119    EWR
#    dest air_time distance hour min    speed max_speed max_dep_delay max_arr_delay
# 1:  LAX      359     2475    9  14 413.6490  526.5957           973           996
# 2:  LAX      363     2475   11  57 409.0909  526.5957           973           996
# 3:  LAX      351     2475   19   2 423.0769  526.5957           973           996
# 4:  PBI      157     1035    7  22 395.5414  517.5000           973           996
# 5:  LAX      350     2475   13  47 424.2857  526.5957           973           996
# 6:  LAX      339     2454   18  24 434.3363  518.4507           973           996
```

- We use the `LHS := RHS` form. We store the input column names and the new columns to add in separate variables and provide them to `.SDcols` and for `LHS` (for better readability).
- Note that since we allow assignment by reference without quoting column names when there is only one column as explained in Section 2c, we can not do `out_cols := lapply(.SD, max)`. That would result in adding one new column named `out_col`. Instead we should do either `c(out_cols)` or simply `(out_cols)`. Wrapping the variable name with `(` is enough to differentiate between the two cases.

- The `LHS := RHS` form allows us to operate on multiple columns. In the RHS, to compute the `max` on columns specified in `.SDcols`, we make use of the base function `lapply()` along with `.SD` in the same way as we have seen before in the *"Introduction to data.table"* vignette. It returns a list of two elements, containing the maximum value corresponding to `dep_delay` and `arr_delay` for each group.

Before moving on to the next section, let's clean up the newly created columns `speed`, `max_speed`, `max_dep_delay` and `max_arr_delay`.

```
# RHS gets automatically recycled to length of LHS
flights[, c("speed", "max_speed", "max_dep_delay", "max_arr_delay") := NULL]
head(flights)
#    year month day dep_time dep_delay arr_time arr_delay cancelled carrier tailnum flight origin
# 1: 2014     1   1      914        14     1238        13         0      AA  N338AA      1    JFK
# 2: 2014     1   1     1157        -3     1523        13         0      AA  N335AA      3    JFK
# 3: 2014     1   1     1902         2     2224         9         0      AA  N327AA     21    JFK
# 4: 2014     1   1      722        -8     1014       -26         0      AA  N3EHAA     29    LGA
# 5: 2014     1   1     1347         2     1706         1         0      AA  N319AA    117    JFK
# 6: 2014     1   1     1824         4     2145         0         0      AA  N3DEAA    119    EWR
#    dest air_time distance hour min
# 1:  LAX      359     2475    9  14
# 2:  LAX      363     2475   11  57
# 3:  LAX      351     2475   19   2
# 4:  PBI      157     1035    7  22
# 5:  LAX      350     2475   13  47
# 6:  LAX      339     2454   18  24
```

3) `:=` and `copy()`

`:=` modifies the input object by reference. Apart from the features we have discussed already, sometimes we might want to use the update by reference feature for its side effect. And at other times it may not be desirable to modify the original object, in which case we can use `copy()` function, as we will see in a moment.

a) `:=` for its side effect

Let's say we would like to create a function that would return the *maximum speed* for each month. But at the same time, we would also like to add the column `speed` to *flights*. We could write a simple function as follows:

```
foo <- function(DT) {
  DT[, speed := distance / (air_time/60)]
  DT[, .(max_speed = max(speed)), by=month]
}
ans = foo(flights)
head(flights)
#    year month day dep_time dep_delay arr_time arr_delay cancelled carrier tailnum flight origin
# 1: 2014     1   1      914        14     1238        13         0      AA  N338AA      1    JFK
# 2: 2014     1   1     1157        -3     1523        13         0      AA  N335AA      3    JFK
# 3: 2014     1   1     1902         2     2224         9         0      AA  N327AA     21    JFK
# 4: 2014     1   1      722        -8     1014       -26         0      AA  N3EHAA     29    LGA
# 5: 2014     1   1     1347         2     1706         1         0      AA  N319AA    117    JFK
# 6: 2014     1   1     1824         4     2145         0         0      AA  N3DEAA    119    EWR
#    dest air_time distance hour min    speed
# 1:  LAX      359     2475    9  14 413.6490
# 2:  LAX      363     2475   11  57 409.0909
# 3:  LAX      351     2475   19   2 423.0769
# 4:  PBI      157     1035    7  22 395.5414
# 5:  LAX      350     2475   13  47 424.2857
# 6:  LAX      339     2454   18  24 434.3363
head(ans)
#    month max_speed
# 1:     1  535.6425
# 2:     2  535.6425
# 3:     3  549.0756
# 4:     4  585.6000
# 5:     5  544.2857
# 6:     6  608.5714
```

- Note that the new column `speed` has been added to `flights` *data.table*. This is because `:=` performs operations by reference. Since `DT` (the function argument) and `flights` refer to the same object in memory, modifying `DT` also reflects on `flights`.
- And `ans` contains the maximum speed for each month.

b) The `copy()` function

In the previous section, we used `:=` for its side effect. But of course this may not be always desirable. Sometimes, we would like to pass a *data.table* object to a function, and might want to use the `:=` operator, but *wouldn't* want to update the original object. We can accomplish this using the function `copy()`.

The `copy()` function *deep* copies the input object and therefore any subsequent update by reference operations performed on the copied object will not affect the original object.

There are two particular places where `copy()` function is essential:

1. Contrary to the situation we have seen in the previous point, we may not want the input data.table to a function to be modified *by reference*. As an example, let's consider the task in the previous section, except we don't want to modify `flghts` by reference.

   Let's first delete the `speed` column we generated in the previous section.

   ```
   flights[, speed := NULL]
   ```

   Now, we could accomplish the task as follows:

   ```
   foo <- function(DT) {
     DT <- copy(DT)                          ## deep copy
     DT[, speed := distance / (air_time/60)]    ## doesn't affect 'flights'
     DT[, .(max_speed = max(speed)), by=month]
   }
   ans <- foo(flights)
   head(flights)
   #    year month day dep_time dep_delay arr_time arr_delay cancelled carrier tailnum flight origin
   # 1: 2014     1   1      914        14     1238        13         0      AA  N338AA      1    JFK
   # 2: 2014     1   1     1157        -3     1523        13         0      AA  N335AA      3    JFK
   # 3: 2014     1   1     1902         2     2224         9         0      AA  N327AA     21    JFK
   # 4: 2014     1   1      722        -8     1014       -26         0      AA  N3EHAA     29    LGA
   # 5: 2014     1   1     1347         2     1706         1         0      AA  N319AA    117    JFK
   # 6: 2014     1   1     1824         4     2145         0         0      AA  N3DEAA    119    EWR
   #    dest air_time distance hour min
   # 1:  LAX      359     2475    9  14
   # 2:  LAX      363     2475   11  57
   # 3:  LAX      351     2475   19   2
   # 4:  PBI      157     1035    7  22
   # 5:  LAX      350     2475   13  47
   # 6:  LAX      339     2454   18  24
   head(ans)
   #    month max_speed
   # 1:     1  535.6425
   # 2:     2  535.6425
   # 3:     3  549.0756
   # 4:     4  585.6000
   # 5:     5  544.2857
   # 6:     6  608.5714
   ```

   - Using `copy()` function did not update `flights` *data.table* by reference. It doesn't contain the column `speed`.
   - And `ans` contains the maximum speed corresponding to each month.

   However we could improve this functionality further by *shallow* copying instead of *deep* copying. In fact, we would very much like to provide this functionality for `v1.9.8`. We will touch up on this again in the *data.table design* vignette.

2. When we store the column names on to a variable, e.g., `DT_n = names(DT)`, and then *add/update/delete* column(s) *by reference*. It

would also modify `DT_n`, unless we do `copy(names(DT))`.

```
DT = data.table(x=1, y=2)
DT_n = names(DT)
DT_n
# [1] "x" "y"

## add a new column by reference
DT[, z := 3]

## DT_n also gets updated
DT_n
# [1] "x" "y" "z"

## use `copy()`
DT_n = copy(names(DT))
DT[, w := 4]

## DT_n doesn't get updated
DT_n
# [1] "x" "y" "z"
```

Summary

The `:=` operator

- It is used to *add/update/delete* columns by reference.
- We have also seen how to use `:=` along with `i` and `by` the same way as we have seen in the *Introduction to data.table* vignette. We can in the same way use `keyby`, chain operations together, and pass expressions to `by` as well all in the same way. The syntax is *consistent*.
- We can use `:=` for its side effect or use `copy()` to not modify the original object while updating by reference.

So far we have seen a whole lot in `j`, and how to combine it with `by` and little of `i`. Let's turn our attention back to `i` in the next vignette *"Keys and fast binary search based subset"* to peform *blazing fast subsets* by *keying data.tables*.
