quantile-improved

Sebastian Fischer

11/02/2020

The line references are referring to this version:

https://github.com/wch/r-source/blob/5a156a0865362bb8381dcd69ac335f5174a4f60c/src/library/stats/R/quantile.R#L24

criticism:

- not enough subfunctions. Therefore the code becomes quite hard to read. Using more subfunctions also modularizes the code and makes it easier to modify

Solution: Modularize the code, i.e. split it up into the various subtasks.

- Keep it simple:
  - the main issue is the high cyclomatic complexity: especially stacked if/switch statements are hard to read (e.g. lines 50 - 102 and 29 - 35)
  - sometimes too much is done in one line: e.g. line 41: why not assign p.ok <- !is.na(probs) in the line before,
  - Line 57: x[hi] != qs implies index > lo

Solution: don't overload single lines, avoid high cyclomatic complexity also by creating more subfunctions and restructuring the code

- defensive programming:
  - no explicit input checks are done
  - no tailored warning messages
  - no warning messages if inputs are implicitely transformed by the functions that are called within quantile.default

Solution: Check inputs explicitely and give tailored warning messages

To illustrate the consequences of no input checking:

- if the type is negative, the function behaves quite unexpected: Because then it holds that type <= 3 in (line 63) but afterwards h is assigned NULL (line 68ff) and at line 91 the quantiles are updated but nothing is done in the following lines because h is NULL...

In addition to that there are simply some inconsitencies and errors/ bugs:

- probabilities that are in (0 - eps, 0) or (1, 1 + eps) are not corrected if probs does not contain any NAs (line 44 - 48).
- rounding errors are only adressed for the continuous quantiles: why?
- bug when names == FALSE and probs contains an NA (arises due to 'names(o.pr)[p.ok] <- names(qs)')
- whether or not character/ Date inputs for x work for types that are not 1 or 3 depends on the exact probs and x. Solution: only allow types 1 and 3 as is done for ordered factors

- some more comments: e.g. line 91 would be much easier to understand with a precise comment, line 98: explain why one needs the if(any(other))

Another major issue, that makes the code hard to read are the terrible names:

- notation is inconsistent with the help page
- abbreviations (e.g. nppm, i, sml ...) make it hard to know what the variable contains
- points are used in names which is reserved for S3/S4 (e.g. o.pr, na.p, p.ok)
- name 'other' (line 91) other is not really informative about what this variable means
- index (line 52): indices are usually whole numbers but not in this case

Summary of improvements:

- write more subfunctions/ modularize the code into the various subtasks.
- simplify the structure, specifically remove the nested if-conditionals

- Adhere to the styleguide and improve variable names
- Write explicit checks and exclude inputs that don't make sense
- remove bugs and some major inconsistencies

Pseudocode

The quantile_tidy function is decomposed into three main components

1. The input checking: check_inputs

- This function checks and modifies the inputs, as well as creates variables that are needed later

2. The actual quantile function: quantile_main

- This function calculates the actual quantiles according to the type
- It presupposes correct input that was already transformed by check_inputs

3. The formatting/ modifcation of outputs: modify_quantiles

- formats the output depending on the value of names and whether the input was a factor

Only the middle part is descriped mathematically, the other parts will only be sketched informally.

1. check_inputs

Algorithm 1: check and modify inputs for the quantile function and create relevant variables

Result: check inputs
Input : x, probs, na.rm, names, type
Output: x, probs, na.rm, names, type, probs_original, probs_NA, x_levels

1 Check that **type** is an almost an integer in 1...9 and if yes round to that integer
2 Check that **names** can be reasonably be interpreted as logical and extract the first element in case length > 1
3 Check that **na.rm** can be reasonably be interpreted as logical and extract the first element in case length > 1
4 Check that **probs** contains only values within $[0 - \epsilon, 1 + \epsilon]$ and round values in $[-\epsilon, 0)$ to 0 and values in $[1, 1 + \epsilon]$ to 1
5 Remove NA values from **probs** and store the original **probs** vector in **probs_original** as well as a vector **probs_NA** that indicates which values in the original **probs** vector were NA
6 Check that **x** is of type numeric, complex, logical, character, date, ordered factor or NULL; break otherwise
7 Remove NAs from **x** if **na.rm** is TRUE, otherwise stop if NAs are contained
8 Safe the levels of **x** in **x_levels**
9 **return** $list($***x, probs, na.rm, names, type, probs_original, probs_NA, x_levels***$)$

2. quantile_main

quantile_main only calls quantile_type_7 in case type is 7 and quantile_general otherwise. Both algorithms are described:

quantile_type_7

Algorithm 2: algorithm for quantile type 7

Result: quantiles according to type 7
Input : $p = (p_1, ..., p_m), \quad x = (x_1, ..., x_n)$
Output: $q = (q_1, ..., q_m)$

1 $k \leftarrow 1 + (n - 1) \times p$
2 $k^{(f)} \leftarrow \lfloor k \rfloor$
3 $k^{(c)} \leftarrow \lceil k \rceil$
4 $l \leftarrow \text{unique}(k_1^{(f)}, ...k_n^{(f)}, k_1^{(c)}, ..., k_n^{(c)})$
5 $x \leftarrow \text{partial\_sort}(x, \text{partial} = l)$
6 $w \leftarrow x_k - x_{k^{(f)}}$
7 $q \leftarrow (1 - w) \times x_{k^{(f)}} + w \times x_{k^{(c)}}$
8 **return** q

quantile_general

Algorithm 3: general quantile algorithm

Result: quantile for types 1 to 9
Input : $p = (p_1, ..., p_m), \quad x = (x_1, ..., x_n)$
Output: $q = (q_1, ..., q_m)$

$a \leftarrow$ get_a(type)
$b \leftarrow$ get_b(type)
$k \leftarrow a + p \times (n + 1 - a - b)$
$k^{(f)} = \lfloor k \rfloor$
$k^{(c)} \leftarrow k^{(f)} + 1$
$l \leftarrow \text{unique}((1, x_{k^{(f)}}, x_{k^{(c)}}, n))$
$x \leftarrow \text{partial\_sort}(x, \text{partial} = l)$
$w \leftarrow$ get_weights(type, $k, k^{(f)}$)
for $i = 1...m$ **do**
    if $k_i < 0$ **then**
        $q_i \leftarrow x_1$
    end
    else if $k_i > n - 1$ **then**
        $q_i \leftarrow x_n$
    end
    else
        $q_i \leftarrow (1 - w_i) \times k_i^{(f)} + w_i \times k_i^{(c)}$
    end
end
return q

3. modify_quantiles

Algorithm 4: modify quantiles

Result: modified quantiles
Input : probs_NA, q, names, x_levels, p
Output: $q = (q_1, ..., q_m)$

Extend **q** with the NAs in the original **probs** vector
Assign **x_levels** to **q**
if ***names*** *is TRUE* **then**
    Assign the corresponding probability values **p** as the names of **q**
end
return q