Updates and Errata for *An R Companion to Applied Regression, Third Edition*

John Fox
Department of Sociology
McMaster University

Sanford Weisberg
School of Statistics
University of Minnesota

2020-05-17

1 Updates

1.1 Default Handling of Character Data in `read.table()` and Other Functions (2020-05-17)

Prior to version 4.0.0 of R, the `read.table()` and `data.frame()` functions, along with other related functions such as `read.csv()`, by default changed all character data to factors. This behavior was controlled by the `"stringsAsFactors"` option, which defaulted to `TRUE`.

With the release of R 4.0.0, the default value of the `"stringsAsFactors"` option has changed to `FALSE`, and at some as yet unspecified time in the future, this option will disappear entirely. Consequently `read.table()`, `data.frame()`, and related functions now by default leave character data as class `"character"` rather than changing them to factors.

This change to the default handling of character-string data in R is largely benign and transparent, because, for example, a character variable used in a model formula is treated as if it were a factor, with the levels given by the unique values of the character variable, in alphabetical order. Should you wish to use a different order for the levels, you can easily convert the character variable to a factor.

For example, imagine that the variable `type` in the `Prestige` data set in the **carData** package is a character variable rather than a factor, and that you wish to rearrange its levels in non-alphabetic order. You can use a command like

```
> Prestige$type <- factor(Prestige$type,
+     levels=c("bc", "wc", "prof"))
```

as on page 198 of the *R Companion* (where, however, the original variable `type` in the data set is already a factor), or, equivalently,

```
> Prestige <- within(Prestige,
+     factor(type, levels=c("bc", "wc", "prof")))
```

The function `strings2factors()` in **car** version 3.0-8 or later converts the character variables in a data frame or a specified subset of the character variables to factors, optionally specifying the order of the levels of the factors.

We are aware of a couple of infelicities introduced by this change (and we may discover more):

1. The standard R `summary()` method for data frames doesn't report the number of cases for each unique values of a character variable as it does for each level of a factor. For example:

   ```
   > summary(Duncan)

      type        income         education         prestige
    bc  :21   Min.   : 7.00   Min.   :  7.00   Min.   : 3.00
    prof:18   1st Qu.:21.00   1st Qu.: 26.00   1st Qu.:16.00
    wc  : 6   Median :42.00   Median : 45.00   Median :41.00
              Mean   :41.87   Mean   : 52.56   Mean   :47.69
              3rd Qu.:64.00   3rd Qu.: 84.00   3rd Qu.:81.00
              Max.   :81.00   Max.   :100.00   Max.   :97.00

   > Duncan.1 <- Duncan
   > Duncan.1$type <- as.character(Duncan.1$type)
   > summary(Duncan.1)
   ```

```
    type                  income          education          prestige
 Length:45            Min.   : 7.00   Min.   :  7.00   Min.   : 3.00
 Class :character     1st Qu.:21.00   1st Qu.: 26.00   1st Qu.:16.00
 Mode  :character     Median :42.00   Median : 45.00   Median :41.00
                      Mean   :41.87   Mean   : 52.56   Mean   :47.69
                      3rd Qu.:64.00   3rd Qu.: 84.00   3rd Qu.:81.00
                      Max.   :81.00   Max.   :100.00   Max.   :97.00
```

For this reason, we added a `"data.frame"` method for the `S()` function in the **car** package (as of version 3.0-8):

```
> S(Duncan.1)
```

```
   type        income          education          prestige
 bc  :21   Min.   : 7.00   Min.   :  7.00   Min.   : 3.00
 prof:18   1st Qu.:21.00   1st Qu.: 26.00   1st Qu.:16.00
 wc  : 6   Median :42.00   Median : 45.00   Median :41.00
           Mean   :41.87   Mean   : 52.56   Mean   :47.69
           3rd Qu.:64.00   3rd Qu.: 84.00   3rd Qu.:81.00
           Max.   :81.00   Max.   :100.00   Max.   :97.00
```

2. As we explain in Section 6.3 of the *R Companion*, a factor can appear on the left-hand side of the model formula for a binary GLM, but `glm()` does not allow character variables as the response. So, for example, suppose that `lfp` in the `Mroz` data set is alternatively a factor with levels `"no"` and `"yes"` or a character variable with similar unique values. We can formula a command such as the following (see page 280)

```
> glm(lfp ~ k5 + k618 + age + wc + hc + lwg + inc,
+     family=binomial, data=Mroz)
```

if `lfp` is a factor, but not if it is a character variable. A logical response still works, however, as in

```
> glm(lfp == "yes" ~ k5 + k618 + age + wc + hc + lwg + inc,
+     family=binomial, data=Mroz)
```

The change in R to the handling of character data also modifies some of the examples in several sections of Chapters 2 and 4 of the *R Companion*:

1. Page 58. The paragraph beginning "In either case..." in the middle of the page should now read:

   > In either case, the variables `condition` and `sex` are saved as character vectors. In the `brief()` output, `cooperation` is flagged as a numeric variable (`[n]`), and `condition` and `sex` are flagged as character variables (`[c]`).

2. Page 58. The output from the `str()` function near the bottom of the page now reads

   ```
   > str(Guyer)

   'data.frame':        20 obs. of  3 variables:
    $ cooperation: num  49 64 37 52 68 54 61 79 64 29 ...
    $ condition  : chr  "public" "public" "public" "public" ...
    $ sex        : chr  "male" "male" "male" "male" ...
   ```

   That is, the variables `condition` and `sex`, previously of class `Factor` are now of class `chr`, short for `"character"`.

3. Page 60. In the `brief()` output at the top of the page, the variable `type` is now of class `[c]` for `"character"`, not `[f]` for class `"factor"`:

   ```
   > brief(Duncan)

   45 x 4 data.frame (40 rows omitted)
              type income education prestige
               [c]    [i]       [i]      [i]
   accountant prof     62        86       82
   pilot      prof     72        76       83
   architect  prof     75        92       90
   . . .
   policeman  bc       34        47       41
   waiter     bc        8        32       10
   ```

4. Page 60. The bullet beginning “The `read.table()` function automatically and silently converts character data to factors …” is no longer correct and can be disregarded.

5. Page 62. In line 6 of the bullet at the top of the page, change *factor* to *character variable*: “Consequently, if a numeric variable unexpectedly become a character variable…”

6. Page 66. In Section 2.1.6, replace item #2 with the following, reflecting a modification to the `Import()` function in version 3.0-8 of the **car** package:

   > Both the `import()` and `Import()` functions by default treat character-string data as character variables, but `Import()` allows character-string data to be converted to factors by setting the argument `stringsAsFactors=TRUE`.

7. Page 67. In line 3, you can ignore the sentence, “To suppress converting character variables to factors, add the argument `stringsAsFactors=FALSE`.”

8. Page 68. In numbered item #6, change “does not automatically …” to “does not have an option to automatically ….”

9. Page 198. The first sentence of the paragraph starting on line 6 is no longer correct: As explained above, as of R 4.0.0, `read.table()` and `read.csv()` do not by default automatically convert character data to factors.

   Categorical variables can be represented in R either as character vectors (i.e., vectors of class `"character"`) or as factors (i.e., objects of class `"factor"`). R functions such as statistical modelling functions almost always treat character vectors as if they were factors when it is necessary to do so.

   Data sets supplied by packages may include either character variables, factors, or both. In particular, the data sets in the **carData** package still include factors.

2 Errata

None yet.