messy.cats R Package Assessment & Improvement Recommendations

By Brice Richard

Objective: The purpose of this document is to humbly provide the messy.cats R development team with recommendations designed to improve its *messy.cats* package functionality by broadening the scope of its utility.

Affirmation: An evaluation was conducted on 1-Dec-2022 to assess the general functionality of the *messy.cats* package. During this evaluation, a few areas for improvement were discovered.

Assessment Version & Scope: The version of the *messy.cats* package that was evaluated was version 1.0. The scope of the *messy.cats* package assessment focused on an analysis of three principal functions believed to provide the highest degree of utility within the package. Consistent with this aim, the *messy.cats* functions that were evaluated are listed as follows:

- cat_match
- cat_replace
- select_metric

General Assessment Remarks: In package version 1.0, there is no plot visualization capability *directly* provided within the *messy.cats* package. An inquiry arises as to whether a plot visualization is applicable to a language processing package like *messy.cats*. The results of the assessment, in part, concluded the following:

- One or more plot visualizations are directly applicable to the messy.cats package.
- Both functionality and capability would enhance the *messy.cats* package with the introduction of a plot visualization.

RECOMMENDATION 1: In the **Messy.Cats Introduction** help page, there is a sample report generated from the execution of the *cat_match* function in which a third-party function (called *gt* from the gt package) actually generates a report. It shows a return list from one to three proposed string matches for each compromised, or "bad" vector element. A corresponding data frame in R is provided for reference as **Figure 1**.

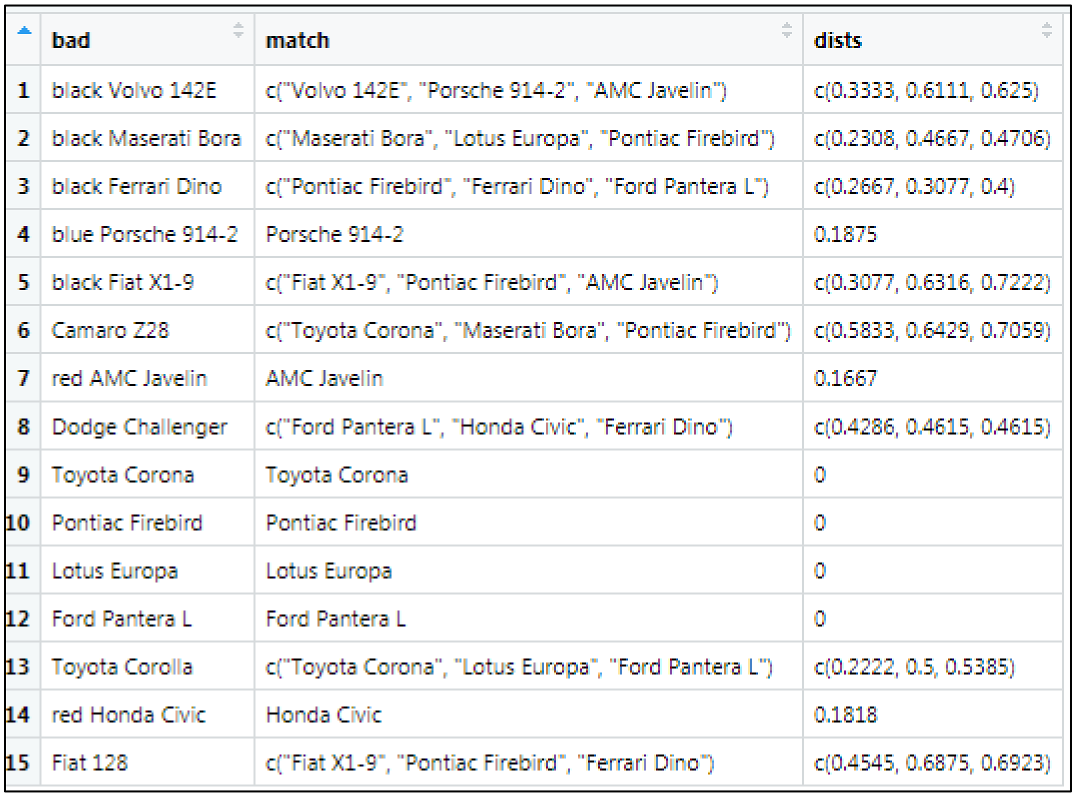
The challenge with the output presented in this form is that it's difficult to read, cannot be easily analyzed, violates data table normalization rules, and requires significant programmatic transformations to correctly normalize as a viable dataset.

Subsequent to executing a set of data transformations, data utility is substantially improved. **Figure 2** shows the output of a correctly defined and structured data frame object. The code used to execute this data transformation is provided in **Figure 3**.

Conclusion: Data output, generated by the *cat_match* function, should be correctly structured and ordered as to form. Data output should remain in the form of a data frame object.

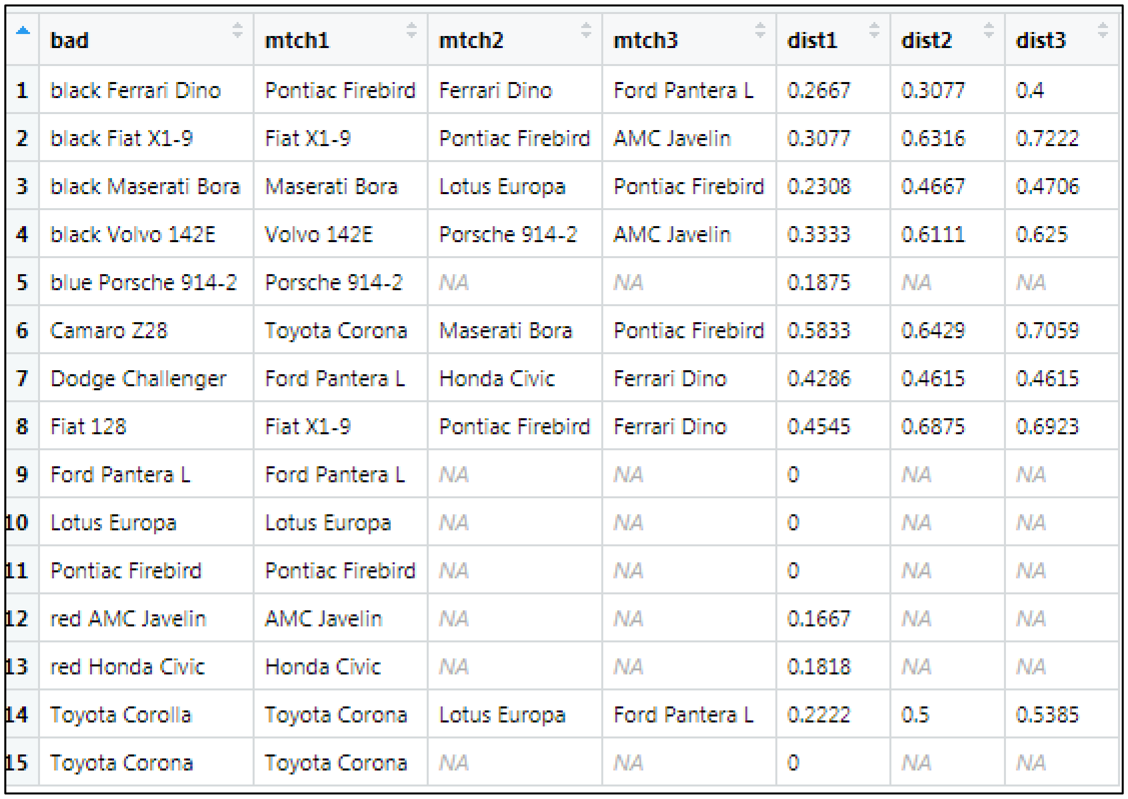
RECOMMENDATION 2: Structured data output as provided in **Figure 2** should be functionally included in any function contained in the *messy.cats* package where string-distance algorithms

Figure 1. A data frame showing bad vector items along with proposed matches and string-distance metrics.



	bad	match	dists
1	black Volvo 142E	c("Volvo 142E", "Porsche 914-2", "AMC Javelin")	c(0.3333, 0.6111, 0.625)
2	black Maserati Bora	c("Maserati Bora", "Lotus Europa", "Pontiac Firebird")	c(0.2308, 0.4667, 0.4706)
3	black Ferrari Dino	c("Pontiac Firebird", "Ferrari Dino", "Ford Pantera L")	c(0.2667, 0.3077, 0.4)
4	blue Porsche 914-2	Porsche 914-2	0.1875
5	black Fiat X1-9	c("Fiat X1-9", "Pontiac Firebird", "AMC Javelin")	c(0.3077, 0.6316, 0.7222)
6	Camaro Z28	c("Toyota Corona", "Maserati Bora", "Pontiac Firebird")	c(0.5833, 0.6429, 0.7059)
7	red AMC Javelin	AMC Javelin	0.1667
8	Dodge Challenger	c("Ford Pantera L", "Honda Civic", "Ferrari Dino")	c(0.4286, 0.4615, 0.4615)
9	Toyota Corona	Toyota Corona	0
10	Pontiac Firebird	Pontiac Firebird	0
11	Lotus Europa	Lotus Europa	0
12	Ford Pantera L	Ford Pantera L	0
13	Toyota Corolla	c("Toyota Corona", "Lotus Europa", "Ford Pantera L")	c(0.2222, 0.5, 0.5385)
14	red Honda Civic	Honda Civic	0.1818
15	Fiat 128	c("Fiat X1-9", "Pontiac Firebird", "Ferrari Dino")	c(0.4545, 0.6875, 0.6923)

Figure 2. A correctly structured and ordered data frame of string-distance information that is easy to analyze and model. It replicates the data provided in **Figure 1**.



	bad	mtch1	mtch2	mtch3	dist1	dist2	dist3
1	black Ferrari Dino	Pontiac Firebird	Ferrari Dino	Ford Pantera L	0.2667	0.3077	0.4
2	black Fiat X1-9	Fiat X1-9	Pontiac Firebird	AMC Javelin	0.3077	0.6316	0.7222
3	black Maserati Bora	Maserati Bora	Lotus Europa	Pontiac Firebird	0.2308	0.4667	0.4706
4	black Volvo 142E	Volvo 142E	Porsche 914-2	AMC Javelin	0.3333	0.6111	0.625
5	blue Porsche 914-2	Porsche 914-2	NA	NA	0.1875	NA	NA
6	Camaro Z28	Toyota Corona	Maserati Bora	Pontiac Firebird	0.5833	0.6429	0.7059
7	Dodge Challenger	Ford Pantera L	Honda Civic	Ferrari Dino	0.4286	0.4615	0.4615
8	Fiat 128	Fiat X1-9	Pontiac Firebird	Ferrari Dino	0.4545	0.6875	0.6923
9	Ford Pantera L	Ford Pantera L	NA	NA	0	NA	NA
10	Lotus Europa	Lotus Europa	NA	NA	0	NA	NA
11	Pontiac Firebird	Pontiac Firebird	NA	NA	0	NA	NA
12	red AMC Javelin	AMC Javelin	NA	NA	0.1667	NA	NA
13	red Honda Civic	Honda Civic	NA	NA	0.1818	NA	NA
14	Toyota Corolla	Toyota Corona	Lotus Europa	Ford Pantera L	0.2222	0.5	0.5385
15	Toyota Corona	Toyota Corona	NA	NA	0	NA	NA

Figure 3. R code used to generate the data transformations provided in **Figure 2**. The original data output comes from the execution of the *cat_match* function.

```
library(easyr)
library(gtools)
library(messy.cats)
library(misty)
library(tidyr)

messy_short = c("Fiat 128", "red Honda Civic", "Toyota Corolla",
"Toyota Corona", "Dodge Challenger", "red AMC Javelin", "Camaro Z28",
"Pontiac Firebird", "black Fiat X1-9", "blue Porsche 914-2", "Lotus
Europa", "Ford Pantera L", "black Ferrari Dino", "black Maserati Bora",
"black Volvo 142E")

clean_short = c("Honda Civic", "Toyota Corona", "AMC Javelin", "Pontiac
Firebird", "Fiat X1-9", "Porsche 914-2", "Lotus Europa", "Ford Pantera
L", "Ferrari Dino", "Maserati Bora", "Volvo 142E")

mtch = cat_match(messy_short, clean_short, method = "jaccard",
return_lists = 3, threshold = 0.2)

mtch$rem = ifelse(left(mtch$match, 2) == "c(", mid(mtch$match, 3,
nchar(mtch$match)), mtch$match)
mtch$rem = ifelse(right(mtch$rem, 1) == ")", left(mtch$rem,
nchar(mtch$rem)-1), mtch$rem)
mtch$rem = gsub(pattern = chr(34), replacement = "", x = mtch$rem)

mtch$distprs = ifelse(left(mtch$dists, 2) == "c(", mid(mtch$dists, 3,
nchar(mtch$dists)), mtch$dists)
mtch$distprs = ifelse(right(mtch$distprs , 1) == ")", left(mtch$distprs
, nchar(mtch$distprs )-1), mtch$distprs)

mtch1 = mtch %>% separate(col = rem, into = c("mtch1", "mtch2",
"mtch3"), sep = ",")
mtch1$mtch1 = trimws(mtch1$mtch1, which = "both")
mtch1$mtch2 = trimws(mtch1$mtch2, which = "both")
mtch1$mtch3 = trimws(mtch1$mtch3, which = "both")

dist1 = mtch %>% separate(col = distprs, into = c("dist1", "dist2",
"dist3"), sep = ",")
dist1$dist1 = trimws(dist1$dist1, which = "both")
dist1$dist2 = trimws(dist1$dist2, which = "both")
dist1$dist3 = trimws(dist1$dist3, which = "both")

struct_mtch = cbind(bad = mtch$bad, mtch1[,4:6], dist1[,5:7])
struct_mtch = df.sort(struct_mtch, bad)
```

are used to match similarly constructed string vectors. Structured data output should not be negatively impacted by a user-defined value provided in the *cat_match* function's *return_lists* argument. For example, if the *return_lists* value is five, a structured data frame should be returned with a maximum of five match fields and five distance fields.

If research to determine the maximum reasonable return list value has not been conducted, it should be. Removing the *return_lists* argument from the *cat_match* function by replacing it with a static value would reduce the number of conditions required to be encoded. If a static value is to be applied, a value of three seems both plausible and reasonable. However, datasets containing more than 100,000 observations may benefit from a return list value greater than three.

Conclusion: Conclusions drawn from Recommendation 2 are provided as follows:

- Add code functionality necessary to convert data frame output generated by the *cat_match* function to a correctly structured dataset.
- Conduct research to determine an optimum return list value.
- If an optimal return list value is identified, remove the *return_lists* argument from the *cat_match* function, replacing it in code with a static value.

RECOMMENDATION 3: If the *cat_match* function is modified to generate a correctly structured dataset, a boolean argument should be added to determine whether the original row number, subsequent to sorting, is retained or not. In the last line of code provided in **Figure 3**, the *df.sort* function, found in the *misty* package, reorders the row number after the struct_mtch data frame has been sorted by a variable called *bad*. Conversely, a sorting action initiated by the *Sort* function found in the *lessR* package sorts a data frame by a target variable, but retains the original row number of the unsorted dataset.

RECOMMENDATION 4: While it is believed that adding correctly structured output to the *cat_match* function would substantially improve the utility of the *messy.cats* R package, the most compelling reason for this recommendation is to facilitate the development of a graphical plotting capability.

Consequently, in version 1.0 of the *messy.cats* package, there is no direct, graphical plotting functionality.

In general terms, it is much easier to plot data in R from a structured dataset than from one that is disordered or unstructured. For example, the structure of the dataset provided in **Figure 2** is ideal for programmatically generating a diagrammatic representation of string-distance information.

To that end, the *consort* package offers a compelling diagrammatic capability within which to plot *cat-match* functional output. While the *consort* package is primarily used as a diagrammatic tool for capturing sequence elements of a Randomized Controlled Clinical Trial (RCT), diagrams can be repurposed through independent customization. As a result, graphical plotting functionality from the *consort* package can be customized for use in the *messy.cats* package in one of two ways:

- Create new graphical plotting functions by repurposing the code provided in the *consort* package, or
- Apply consort-based functions as a set of wrappers used to support graphical plotting functionality

The figure below applies R code that programmatically extracts data from the eighth record in the dataset represented in **Figure 2**. The code generates a diagram from a set of functions found in the *consort* package.

Figure 4. Repurposed code taken from the *consort* package is used to support proposed graphical plotting functionality in the *messy.cats* package.

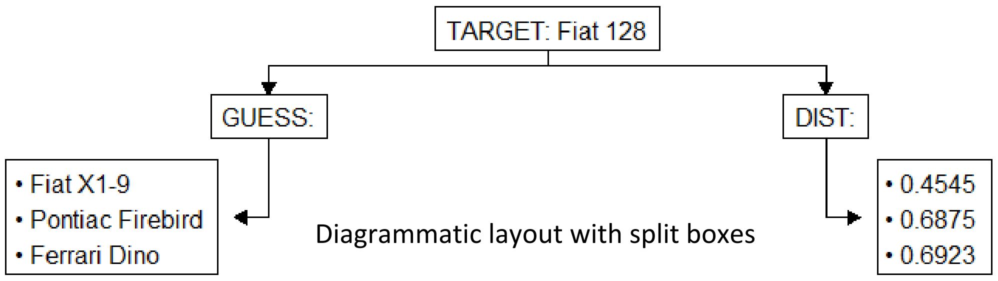
```
library(consort)
library(gtools)
```

```
txt1 = paste0("TARGET: ", struct_mtch[8,1])
txt2 = paste(c("GUESS:", "DIST:"))
u = "\u2022"
brk = "\n"
sp = chr(32)

txt3  =  paste0(u,  sp,  struct_mtch[8,2],  brk,  u,  sp,  struct_mtch[8,3],
brk, u, sp, struct_mtch[8,4])

txt4  =  paste0(u,  sp,  struct_mtch[8,5],  brk,  u,  sp,  struct_mtch[8,6],
brk, u, sp, struct_mtch[8,7])

dgram = add_box(txt = txt1) |>
        add_split(txt = txt2) |>
        add_side_box(txt = c(txt3, txt4))
plot(dgram)
```



Diagrammatic layout with split boxes

With a slight code modification, constructing various diagrammatic layouts is possible. **Figure 5** and **Figure 6** provide examples.

Figure 5. Diagrammatic layout with a direct box.

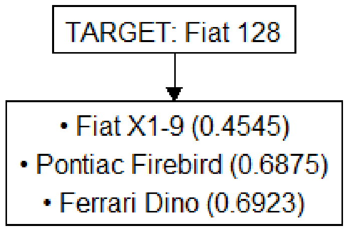
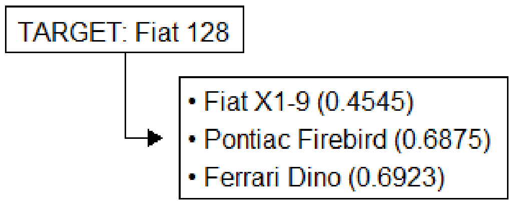


Figure 6. Diagrammatic layout with a side box.



For more information on diagrammatic plotting, see the *consort* package found on CRAN.

<u>**RECOMMENDATION 5:**</u> In conjunction with Recommendation 4, how would a graphical plotting function in *messy.cats* be represented? The proposed structural framework for a graphical plotting function supporting the *messy.cats* package is provided below:

PLOT FUNCTION NAME:	plot_str_dist
FUNCTION NAME MEANING:	Plot String Distance
NBR OF ARGUMENTS:	4
ARGUMENT NAMES:	df, rec, layout, and title

ARGUMENT DEFINITIONS:	**df** = Represents the structured data frame object passed to the function.
	rec = References the row name of the observation used to populate the diagram.
	layout = The layout type used to construct the diagram; passes a numeric value from 1 to 3; this argument could also be represented by the diagram type such as split, direct, or side.
	title = The name of the diagram; for example: "Variable Category Match Recommendations."
FUNCTIONAL SYNTAX:	plot_str_dist(df, rec, layout = 1, title = "Variable Category Match Recommendations")

Conclusion: This evaluation offers a few dimensional insights designed to improve the functional utility of the *messy.cats* package. It certainly does not answer all the questions related to a successful implementation of a plotting capability in an actively published R package. Consequently, this assessment is principally designed as a baseline starting point for considering the possibilities.

Please direct any questions you or your team may have regarding this document or the ideas therein to Brice Richard at **wem3mew@msn.com**. I will help the *messy.cats* team in any way I can in support of improving the functional viability of the package.

Thank you for the opportunity to provide this proposal of ideas. I look forward to the next version of the *messy.cats* package!