SoftwareX 27 (2024) 101865

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

# clickR: Semi-automatic pre-processing of messy data with change tracking for integral dataset cleaning

David Hervas [a,*], David Fuente [b]

[a] Department of Applied Statistics, Operations Research, and Quality, Universitat Politècnica de València, Plaça Ferrándiz i Carbonell, 1, Alcoi, Spain
[b] Instituto de Aplicaciones de las Tecnologías de la Información y de las Comunicaciones Avanzadas, Universitat Politècnica de València, Camino de Vera s/n, Valencia, Spain

ARTICLE INFO

*Keywords:*
Data pre-processing
Data cleaning
R package

ABSTRACT

In this contribution, we present *clickR*, an **R** package intended for data cleaning following a semi-automatic and supervised procedure. Few packages and commercial software with cleaning capacities are available. In all cases, their functionalities just cover part of the overall data pre-processing and do not follow an integral approach to cleaning up the data. In contrast, *clickR* brings together all functions needed for correcting the main structural, variable-assignment and typographical errors found in databases and allows researchers to have a strict control on the suggested changes. This is possible because the package creates a data frame that keeps track of all the implemented data modifications. To prove its capacity for detecting and fixing errors, we clean a messy database that exhibits multiple types of errors within date, numeric and factor variables.

## Code metadata

| | |
|---|---|
| Current code version | 0.9.41 |
| Permanent link to code/repository used for this code version | https://github.com/ElsevierSoftwareX/SOFTX-D-23-00501 |
| Permanent link to Reproducible Capsule | |
| Legal Code License | GPL-2 \| GPL-3 |
| Code versioning system used | git |
| Software code languages, tools, and services used | R |
| Compilation requirements, operating environments & dependencies | R packages *beeswarm*, *future*, *future.apply*, *methods* and *stringdist* |
| If available Link to developer documentation/manual | https://cran.r-project.org/web/packages/clickR/clickR.pdf |
| Support email for questions | daherma@eio.upv.es |

## 1. Motivation and significance

In the advent of the big data era, vast amounts of generated data often contain typographical mistakes or display inconsistent formats [1]. In addition, many public repositories store information from different sources that once merged might present dissimilar structures, such as multiple date formats or display both decimal separators [2]. Moreover, massive databases are daily created by means of scrapping methods. Such data can enhance research, but that information is often not reliable and messy. Additionally, human mistakes are always prone to appear, especially when data are inputted manually. Indeed, there are many types of errors that can affect databases [3]: Incompleteness of data, presence of missing observations, inconsistencies in the codification of categorical variables, typing errors, impossible or out-of-range values in the case of numerical variables or multiple date formats, among others. As the size of datasets increase, so does the number of present errors and the corresponding data cleaning time. Even after a careful revision, some mistakes can be overlooked, leading to biased or inaccurate results. Thus, tools accelerating data scrubbing can save significant amounts of working time and capital resources, avoiding researchers to reach spurious conclusions. Therefore, data cleaning becomes an often-underestimated task that is fundamental for a rigorous and correct data analysis.

There are different tools aimed at screening and checking data quality such as Alteryx Designer [4], Trifacta [5] and OpenRefine,

* Corresponding author.
*E-mail addresses:* daherma@eio.upv.es (David Hervas), dafueher@upv.es (David Fuente).

https://doi.org/10.1016/j.softx.2024.101865
Received 31 July 2023; Received in revised form 15 June 2024; Accepted 22 August 2024
Available online 26 August 2024

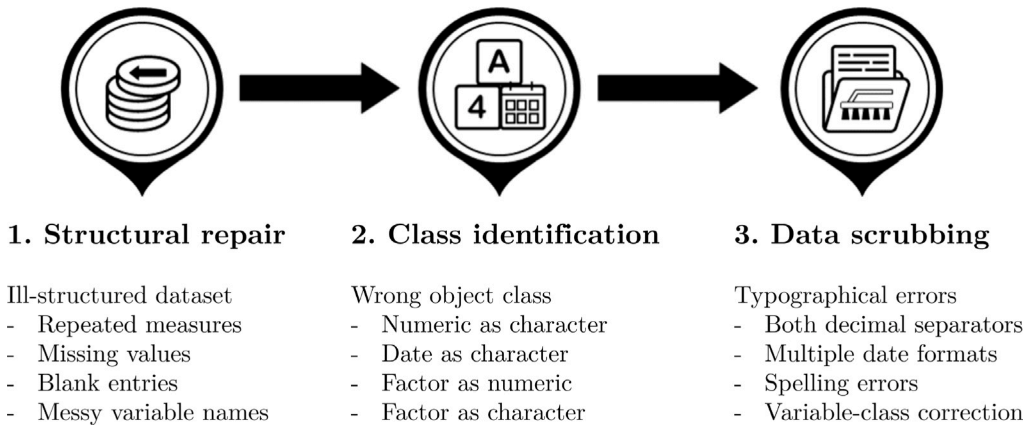


**Fig. 1.** Scheme of the data pre-processing routine followed in *clickR*.

previously called Google Refine [6], among others. Most of these tools offer user-friendly interfaces where programming is not necessary and data-processing workflows can be created. However, many of the tools are in reality oriented towards data visualization and exploration and particular methods for data cleaning have to be programmed. Regarding the R language [7], there are some packages oriented towards data wrangling and screening, but few can be used for data cleaning. The *validate* [8] and *editrules* [9] packages check for mistakes by user-defined rules. The *janitor* package incorporates some functions for curation of variable names and includes other features for managing Excel data [10]. *varhandle* offers a function to check which character data might be safely converted to the numerical class [11], while *lubridate* contains functions that can identify and parse date-time information [12]. Another useful package is *dataMaid*, which generates an overview document showing the errors and warnings encountered during the data-quality assessment [13]. However, it does not have any data-fixing tools and requires manual fixing of the found errors.

*clickR* functions fix common mistakes within numeric, date and factor variables in a semi-automatic way with minimal external input. Remarkably, all performed changes are thoroughly registered in a data frame that can be consulted or exported as an external document. Code for dataset cleaning is compact, easy to use and can be directly reapplied to other databases: It allows users to clean very messy datasets within minutes.

## 2. Software description

*clickR* is designed to perform a complete data cleaning routine covering all error types present in a database. This routine is split into three different stages: Correction of the dataset structure, class identification of each variable and finally, fixing of variable-specific errors. All modifications made on the dataset are stored in a table that can be used to track the changes and, in case it is needed, to easily revert them. We show a simplified scheme of the data pre-processing performed by *clickR* in Fig. 1. The package can be installed by executing `install.packages('clickR')` in a running R session.

### 2.1. Software architecture

The software is designed as an R package, and its structure follows the guidelines enforced by the Comprehensive R Archive Network repository (CRAN): in consists on the files DESCRIPTION, NAMESPACE and NEWS and the directories data, man and R. This last directory contains three .R files with the code defining the 35 exported functions (intended for users) and 15 internal functions of the package. *clickR* also imports some accessory functions from the R packages *beeswarm* [14] (`beeswarm()` function to create bee swarm plots), *future* and *future.apply* [15] (`future_lapply()` function to parallelize tasks), and *stringdist* [16] (`stringdistmatrix()`) function to estimate string distances for the `fix_levels()` function. Use of the package follows the standard procedure of most R packages, where exported functions are made accessible to the user when the package is loaded by the command *library(clickR)*.

### 2.2. Software functionalities

*clickR* incorporates three groups of functions related to data cleaning: a set of functionalities for performing exploratory data analysis and assessing data quality, another for detecting and correcting errors, and a third one with functions to review and restore the changes applied to the data by the different corrective functions. Among the functions for detecting and correcting errors, there are three subtypes: those that perform structural repair and work on all the dataset, those that work on all dataset variables from a specific class and those that work on a specific variable at a time.

*Exploratory functions*

Exploratory functions are key for detecting problems in the data, allowing for easy identification of wrongly assigned classes, typographical errors and bad data structure. They are also useful for checking the data after all corrections have been performed. *clickR* provides two summarizing functions for performing exploratory data analysis:

- `peek`
  The object of this function is to display the elementary information of each variable of the data frame by showing the first 10 rows (by default). The comparable R *base* function would be `head`. However, `peek` reports the range and the class of each variable.
- `descriptive`
  It creates a detailed summary of the data frame, including more statistics and a better output format (i.e., a data.frame) than the `summary` function from the *base* package.

*Structural-repair functions*

These functions aim at correcting structural deficiencies in the datasets:

- `nice_names`
  It contributes to tidying up the data by cleaning up the variable names. These are converted to lower case (optionally), whitespace characters transformed into underscore ones and symbols or non-ASCII characters removed.

- `fix_NA`
  Its main capacity is to identify miscoded missing values. `fix_NA` leverages regular expressions to search for different missing value strings such as "?", "-" or "" and substitutes them with a valid `NA` value.
- `remove_empty`
  This function allows to remove empty rows and columns from the data frame.

*Class-identification and data-scrubbing functions*

In *clickR*, the functions performing class identification do not just identify and modify variable classes but also correct most present errors for that variable type at once: The steps 2 and 3 depicted in Fig. 1 are tackled together. Thus, after the recognition of the adequate category for the selected column, data scrubbing takes over in the same function.

- `fix_dates`
  The function is applied to a data frame for detecting and standardizing date variables imported as characters or factors.
- `fix_numerics`
  This function assesses and fixes numeric data misrecognized as factors or characters in a data frame by correcting erroneous decimal separators and removing thousands separators as well as other non-numeric characters.
- `fix_factors`
  It aims at tidying up factors and transforming variables imported as numeric or character into factors.

*Exclusively data-scrubbing functions*

In contrast to the variables described in the previous sections, which are applied to a whole data frame, these functions are applied specifically on a single variable at a time. Therefore, they rely on a correct class identification at an earlier step of the data cleaning procedure.

- `fix_levels`
  This function can correct misspelled factor categories. The algorithm performs hierarchical clustering based on the variable levels using the string distance matrix defined in the `method` argument. Distances are calculated by the `stringdistmatrix` function from package *stringdist* [16].
- `fix_concat`
  It is common for some categorical variables that they take multiple values at the same time. If single observations have different concatenated values, the analysis of the data cannot be appropriate. Therefore, `fix_concat` aims at fixing the problem by splitting the variable in different logical variables, one for each of the possible categories.
- `manual_fix`
  In some cases, the specific functions described above will not be able to fix all the mistakes in a dataset, and there might be a need for some fine tuning on specific observations. This fine tuning could be performed with *base* R functions, but the change-tracking functionality of the package would be interrupted. `manual_fix` is used for such cases, to assign specific values to given observations while keeping the tracking of the implemented changes.

*Tracking-related functions*

- `track_changes`
  All corrections applied on the data by the different functions are recorded in a change-tracking data frame that is stored as an attribute of the data frame containing the dataset. `track_changes` prints the data frame with tracked changes, potentially subsetting it with logical expressions using the `subset` argument of the function.
- `restore_changes`
  All changes recorded in the change-tracking data frame can be reverted by using the `restore_changes` function. The function needs only one argument, which must be a data frame generated by the `track_changes` function, including all or any subset of the implemented modifications. All changes within the provided change-tracking data frame will be reverted in the dataset. Changes can be restored in arbitrary order.

### 2.3. Performance

In order to provide an estimate of the time it takes to clean a dataset, Table 1 shows the computational time taken to clean different datasets with varying sizes. From the results, it can be deduced that the computational time scales linearly with the number of cells (rows × columns) to process. The use of parallelization, which is implemented in most *clickR* functions, significantly reduces computational times.

## 3. Illustrative example

To demonstrate the functionality of the package, we will analyze a modified version of the *R* dataset `mtcars` called `mtcars_messy`, which is included in the *clickR* package.

Our first step within the cleaning procedure will be to check the variable names. We always can make use of the `track_changes` function to review the fixes that have been applied to the data. The columns in the track_changes data.frame are the following: `variable` refers to the name of the variable that has been modified, `observation` refers to the rowname of the observation that has been modified, `original` contains the original value in the data prior to its modification, `new` contains the new value and `fun` is the name of the function that performed the modification. In this case, the names of the variables `mpg`, `hp` and `gear` have been corrected.

```
> mtcars_messy <- nice_names(mtcars_messy)
> track_changes(mtcars_messy)

variable observation original      new        fun
     mpg     varname      Mpg      mpg nice_names
      hp     varname       hp       hp nice_names
 n_gears     varname  n Gears  n_gears nice_names
```

Then, we continue with the detection of potential missing values. We call `track_changes` againg to review the changes, but this time subsetting the results to display only those modifications performed by the `fix_NA` function. Four missing values have beeen detected and correctly assigned.

```
> mtcars_messy <- fix_NA(mtcars_messy)
> track_changes(mtcars_messy, fun == "fix_NA")

variable        observation original  new    fun
    drat         Duster 360           <NA> fix_NA
    drat        Honda Civic        -  <NA> fix_NA
      vs  Cadillac Fleetwood       ?  <NA> fix_NA
      vs        Porsche 914-2   NULL  <NA> fix_NA
```

We will continue by fixing the dates present in the data with the `fix_dates` function. As can be seen, all the different formats are correctly recognized. The variable `date`, which previously was of class 'character', now its class 'Date' and 7 dates in non-standard format have been converted to the standard format in R.

```
> mtcars_messy <- fix_dates(mtcars_messy)
> track_changes(mtcars_messy, fun == "fix_dates")

 variable    observation       original        new       fun
     date            all      character       Date fix_dates
     date     Duster 360     26/08/1974 1974-08-26 fix_dates
     date      Merc 280C 3rd July 1974 1974-07-03 fix_dates
     date     Merc 450SE     06/25/1974 1974-06-25 fix_dates
     date Toyota Corolla       19/06/74 1974-06-19 fix_dates
     date   Lotus Europa   14 /08-1974 1974-08-14 fix_dates
     date   Maserati Bora   1973, 03, 03 1973-03-03 fix_dates
     date     Volvo 142E       12/22/73 1973-12-22 fix_dates
```

**Table 1**
Dataset cleaning times at varying dataset sizes for the different main fix-functions.

| Dataset size | fix_numerics() | fix_dates() | fix_factors() | fix_NA() |
|---|---|---|---|---|
| 100 variables × 5k rows | 1.20 s | 0.12 s | 0.06 s | 2.64 s |
| 100 variables × 50k rows | 10.49 s | 1.13 s | 0.48 s | 26.13 s |
| 100 variables × 100k rows | 22.27 s | 2.33 s | 1.18 s | 53.01 s |
| 100 variables × 1M rows | 236.07 s | 21.79 s | 9.56 s | 523.22 s |
| 500 variables × 5k rows | 6.6 s | 0.67 s | 0.27 s | 13.18 s |
| 500 variables × 100k rows | 119.99 s | 11.50 s | 5.62 s | 267.91 s |
| 500 variables × 100k rows (parallelized in 4 cores) | 56.03 s | – | – | 169.96 s |

Next, we apply the `fix_numerics` function to fix all inconsistencies in numerical variables. Again, all errors have been fixed.

```
> mtcars_messy <- fix_numerics(mtcars_messy)
> track_changes(mtcars_messy, fun == "fix_numerics")

 variable          observation  original     new          fun
      mpg                  all character numeric fix_numerics
     drat                  all character numeric fix_numerics
       wt                  all character numeric fix_numerics
      mpg           Datsun 710      22,8    22.8 fix_numerics
      mpg       Hornet 4 Drive     21.,4    21.4 fix_numerics
      mpg           Duster 360  14.3 mpg    14.3 fix_numerics
      mpg             Merc 280      19.2    19.2 fix_numerics
      mpg            Merc 280C   1.78e01    17.8 fix_numerics
       wt  Lincoln Continental     5,424   5.424 fix_numerics
       wt    Chrysler Imperial     5,345   5.345 fix_numerics
```

We will end this short demonstration with the correction of the variable `maker`, which contains some typographical errors in some of its values. For this, we can use the `fix_levels` function. The function has corrected the values 'ornet', 'merck' and 'toyotta' with their corresponding expected values 'hornet', 'merc' and 'toyota'.

```
> mtcars_messy <- fix_levels(mtcars_messy, "maker",
  plot=TRUE, k=22, levels="auto")
> track_changes(mtcars_messy, fun == "fix_levels"
  & tolower(original) != tolower(new))

variable     observation original    new        fun
   maker  Hornet 4 Drive    ornet hornet fix_levels
   maker       Merc 450SE    merck   merc fix_levels
   maker   Toyota Corona  toyotta toyota fix_levels
```

After reviewing of the changes, it can be concluded that all performed modifications on the dataset are correct. Nevertheless, we will show how to restore an unwanted modification. In this case we will restore the changes performed by the `fix_numerics` function on the `wt` variable. The modified values are showed first and then, after restoration, the original values are recovered.

```
> mtcars_messy$wt[rownames(mtcars_messy)
"Chrysler Imperial")]
[1] 5.424 5.345
> mtcars_messy <- restore_changes(track_changes(mtcars_messy,
                                      fun == "fix_numerics" &
                                      variable == "wt"))
> mtcars_messy$wt[rownames(mtcars_messy)
"Chrysler Imperial")]
[1] "5,424" "5,345"
```

## 4. Impact

Eficient cleaning of messy and faulty datasets is a fundamental procedure in scientific research. To that end, *clickR* provides functionality that was not available before for cleaning messy datasets in a user-friendly, semi-automatic and intuitive way. The package functionalities save plenty of time to researchers as most errors are amended automatically. In this regard, it is often claimed that data cleaning decisions have to be carefully taken by humans. However, such curation can also be automatized when the correction process is supervised by the users: They can accept the performed changes or restore original values if necessary. In summary, we think that *clickR* can be an effective tool for scientists regardless of their programming skills, since it is easy to use and incorporates all routines needed for a fast and integral cleaning of messy databases.

The utility of the *clickR* package is proven by the number of downloads from CRAN (63000 since its publication) and currently around 750 per month. Additionally, numerous studies that have been published as scientific papers have used the package for data-cleaning tasks. For instance, Labusch et al. [17], Blandino et al. [18], Ribelles et al. [19], and Lago et al. [20].

## 5. Conclusions

In research, databases often store vast amounts of information from various sources, making them susceptible to human input errors, system glitches, and data integration issues. These errors can significantly undermine the reliability and validity of research findings, leading to erroneous conclusions and non-reproducible results. Fixing all errors in a database can be an extremely time-consuming task, and it is also prone to human errors. The package *clickR* provides easy-to-use tools for performing all the data cleaning tasks in a semi-automatic and supervised way, potentially saving large amounts of researchers' time and avoiding the use of faulty datasets. In this contribution, we have described the structure of the package, outlined all its functionality, and provided a short illustrative example covering many of its main functions. Showing how to easily fix a messy dataset with a few lines of code.

Future versions of the package will focus on improving computation times for the data-cleaning tasks and adding additional functionalities suggested by the users.

## CRediT authorship contribution statement

**David Hervas:** Conceptualization, Formal analysis, Investigation, Methodology, Software, Supervision, Writing – original draft, Writing – review & editing. **David Fuente:** Validation, Writing – original draft, Writing – review & editing.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

The data is available as part of the software.

## References


[1] Puts M, Daas P, de Waal T. Finding errors in Big Data. Significance 2015;12(3):26–9. http://dx.doi.org/10.1111/j.1740-9713.2015.00826.x.

[2] Hernández MA, Stolfo SJ. Real-world data is dirty: Data cleansing and the merge/purge problem. Data Min Knowl Discov 1998;2(1):9–37. http://dx.doi.org/10.1023/A:1009761603038.

[3] Ge M, Helfert M. A review of information quality research-develop a research agenda. In: Proceedings of the 2007 international conference on information quality. 2007, p. 76–91.

[4] Alteryx, Inc. Alteryx designer software, version 2021.1. 2021, Irvine, CA. URL https://www.alteryx.com/.
[5] Trifacta. Trifacta wrangler software, release 8.1. 2021, San Francisco, CA. URL https://www.trifacta.com/.
[6] Verborgh R, De Wilde M. Using OpenRefine. Packt Publishing Ltd; 2013.
[7] RCore Team. R: A language and environment for statistical computing. Vienna, Austria: R Foundation for Statistical Computing; 2020, URL https://www.R-project.org/.
[8] van der Loo M, de Jonge E. *validate*: Data validation infrastructure for R. 2018, R package version 0.2.6. URL https://CRAN.R-project.org/package=validate.
[9] de Jonge E, van der Loo M. *editrules*: Parsing, applying, and manipulating data cleaning rules. 2018, R package version 2.9.3. URL https://CRAN.R-project.org/package=editrules.
[10] Firke S. *janitor*: Simple tools for examining and cleaning dirty data. 2020, R package version 2.0.1. URL https://CRAN.R-project.org/package=janitor.
[11] Mahmoudian M. *varhandle*: Functions for robust variable handling. 2020, R package version 2.0.5. URL https://CRAN.R-project.org/package=varhandle.
[12] Grolemund G, Wickham H, Spinu V, Constigan I, Lee CH, Cotton R, et al. *lubridate*: Make dealing with dates a little easier. 2013, CRAN Repository. URL http://cran.r-project.org/web/packages/lubridate/index.html.
[13] Petersen AH, Ekstrøm CT. *dataMaid*: Your assistant for documenting supervised data quality screening in R. J Stat Softw 2019;90(6). http://dx.doi.org/10.18637/jss.v090.i06.
[14] Eklund A, Trimble J. Beeswarm: The bee swarm plot, an alternative to stripchart. 2021, R package version 0.4.0. URL https://CRAN.R-project.org/package=beeswarm.
[15] Bengtsson H. A unifying framework for parallel and distributed processing in R using futures. R J 2021;13(2):208–27. http://dx.doi.org/10.32614/RJ-2021-048.
[16] van der Loo M. The stringdist package for approximate string matching. R J 2014;6:111–22, URL https://CRAN.R-project.org/package=stringdist.
[17] Labusch M, Perea M, Sahuquillo-Leal R, Bofill-Moscardó I, Carrasco-Tornero Á, Cañada-Pérez A, et al. Development of moral judgments in impersonal and personal dilemmas in autistic spectrum disorders from childhood to late adolescence. J Autism Dev Disord 2022;1–13.
[18] Blandino A, Scherer D, Rounge TB, Umu SU, Boekstegers F, Barahona Ponce C, et al. Identification of circulating lncRNAs associated with gallbladder cancer risk by tissue-based preselection, cis-eQTL validation, and analysis of association with genotype-based expression. Cancers 2022;14(3):634.
[19] Juan Ribelles A, Barberá S, Yáñez Y, Gargallo P, Segura V, Juan B, et al. Clinical features of neuroblastoma with 11q deletion: an increase in relapse probabilities in localized and 4S stages. Sci Rep 2019;9(1):13806.
[20] Lago V, Fotopoulou C, Chiantera V, Minig L, Gil-Moreno A, Cascales-Campos P, et al. Risk factors for anastomotic leakage after colorectal resection in ovarian cancer surgery: A multi-centre study. Gynecol Oncol 2019;153(3):549–54.