edm

Extended Data Management

It is an R package providing a set of tools to manage your data.

Documentation for each functions is written in edm1/man.

To see the documentation in your R shell:

1. library("devtools")
2. devtools::load_all(path_to_edm_pkg)
3. ?function_from_edm

PKG Requirements

1. stringr
2. readxl
3. openxlsx
4. writexl
5. stringi

Documentation

diff_xlsx

Allow to see the difference between two datasets and output it into an xlsx file

@param file_ is the file where the data is @param sht is the sheet where the data is @param v_old_begin is the corrdinate (row, column) where the data to be compared starts @param v_old_end is the same but for its end @param v_new_begin is the coordinates where the comparator data starts @param v_new_end is the same but for its end @param df2 is optional, if the comparator dataset is directly a dataframe @param overwrite allow to overwrite differences is (set to T by default) @param color_ is the color the differences will be outputed @param pattern is the pattern that will be added to the differences if overwritten is set to TRUE @param output is the name of the outputed xlsx (can be set to NA if no output) @param new_val if overwrite is TRUE, then the differences will be overwritten by the comparator data @param pattern_only will cover differences by pattern if overwritten is set to TRUE

insert_df

Allow to insert dataframe into another dataframe according to coordinates (row, column) from the dataframe that will be inserted @param df_in is the dataframe that will be inserted @param df_ins is the dataset to be inserted ins_loc is a vector containg two parameters (row, column) of the begining for the insertion

pattern_generator

Allow to create patterns which have a part that is varying aech time randomly @param base_ is the pattern that will be kept @param from_ is the vector from which the element of the varying part will be generated @param hmn is how many of varying pattern from the same base will be created @param after is set to 1 by default, it means that the varying part will be after the fixed part, set to 0 if you want the varying part to be before

pattern_tuning

Allow to tune a pattern very precisely @param pattrn is the character that will be tuned @param spe_nb is the number of new character that will be replaced @param spe_l is the source vector from which the new characters will be replace old ones @param exclude_type is character that won't be replaced @param hmn is how many output the function will return @param rg is a vector with two parameters (index of the first letter that will be replaced, index of the last letter that will be replaced) default is set to all the letters from the source pattern

unique_pos

Return the indexes of the first unique values from a vector

@param vec is the input vector

can_be_num

Return TRUE if a variable can be converted to a number and FALSE if not @param x is the input value

data_gen

Allo to generate in a csv all kind of data you can imagine according to what you provide @param type_ is a vector for wich argument is a column, a column can be made of numbers ("number"), string ("string") or both ("mixed") @param strt_l is a vector containing for each column the row from which the data will begin to be generated @param nb_r is a vector containing for each column, the number of row full from generated data
@param output is the name of the output csv file @param type_distri is a vector which, for each column, associate a type of distribution ("random", "poisson", "gaussian"), it meas that non only the number but also the length of the string will be randomly generated according to these distribution laws @param properties is linked to type_distri because it is the parameters ("min_val-max_val") for "random type", ("u-x") for the poisson distribution, ("u-d") for gaussian distribution @param str_source is the source (vector) from which the character creating random string are (defult set to the occidental alphabet) @param round_l is a vector which, for each column containing number, associate a round value @param sep_ is the separator used to write data in the csv @return new generated data in addition to saving it in the output

data_meshup

Allow to automatically arrange 1 dimensional data according to vector and parameters @param data is the data provided (vector) each column is separated by a unic separator and each dataset from the same column is separated by another unic separator (ex: c(“ *”, c(”d”, ”-”, ”e”, ”-”, ”f”), ”*”, c(“a”, “a1”, “-”, “b”, “-”, “c”, “c1”)“ *”) @param cols is the colnames of the data generated in a csv @param file* is the file to which the data will be outputed @param sep_ is the separator of the csv outputed @param organisation is the way variables include themselves, for instance ,resuming precedent example, if organisation=c(1, 0) so the data output will be: d, a d, a1 e, c f, c f, c1 @param unic_sep1 is the unic separator between variables (default is” *“) @param unic_sep2 is the unic separator between datasets (default is”-“) jsq <- sum(sep_dd[!is.na(sep_dd)]) - 1 #numb of var hmn <- (sum(sep_dd[!is.na(sep_dd)]) / jsq) #numb of datasets val* <- v_rel[var_] # val to be added

letter_to_nb

Allow to get the number of a spreadsheet based column by the letter ex: AAA = 703

@param letter is the letter (name of the column)

nb_to_letter

Allow to get the letter of a spreadsheet based column by the number ex: 703 = AAA

@param x is the number of the column

cost and taxes

Allow to calculate basic variables related to cost and taxes from a bunch of products (elements) So put every variable you know in the following order:

@param qte is the quantity of elements @param pu is the price of a single elements without taxes @param prix_ht is the duty-free price of the whole set of elements @param tva is the percentage of all taxes @param prix_ttc is the price of all the elements with taxes @param prix_tva is the cost of all the taxes @param pu_ttc is the price of a single element taxes included @param adjust is the discount percentage @param prix_d_ht is the free-duty price of an element after discount @param prix_d_ttc is the price with taxes of an element after discount @param pu_d is the price of a single element after discount and without taxes @param pu_d_ttc is the free-duty price of a single element after discount

the function return a vector with the previous variables in the same order those that could not be calculated will be represented with NA value xx-month-xxxx -> xx-xx-xxxx

formate_date

Allow to convert xx-month-xxxx date type to xx-xx-xxxx

@param f_dialect are the months from the language of which the month come @param sentc is the date to convert @param sep_in is the separator of the dat input (default is “-”) @param sep_out is the separator of the converted date (default is “-”)

until_stnl

Maxes a vector to a chosen length

ex: if i want my vector c(1, 2) to be 5 of length this function will return me: c(1, 2, 1, 2, 1) @param vec1 is the input vector @param goal is the length to reach

vlookup_df

Alow to perform a vlookup on a dataframe

@param df is the input dataframe @param v_id is a vector containing the ids @param col_id is the column that contains the ids (default is equal to 1) @param included_col_id is if the result should return the col_id (default set to yes)

multitud

From a list containing vectors allow to generate a vector following this rule:

list(c("a", "b"), c("1", "2"), c("A", "Z", "E")) –> c("a1A", "a2A", "b1A", "b2A", "a1Z", . . .) @param l is the list @param sep_ is the separator between elements (default is set to "" as you see in the example)

df_tuned

Allow to return a list from a dataframe following these rules: First situation, I want the vectors from the returned list be composed of values that are separated by special values contained in a vector ex: data.frame(c(1, 1, 2, 1), c(1, 1, 2, 1), c(1, 1, 1, 2)) will return list(c(1, 1), c(1, 1, 1), c(1, 1, 1, 1)) or list(c(1, 1, 2), c(1, 1, 1, 2), c(1, 1, 1, 1, 2)) if i have chosen to take in count the 2. As you noticed here the value to stop is 2 but it can be several contained in a vector Second situation: I want to return a list for every jump of 3. If i take this dataframe data.frame(c(1, 1, 2, 1, 4, 4), c(1, 1, 2, 1, 3, 3), c(1, 1, 1, 2, 3, 3)) it will return list(c(1, 1, 2), c(1, 4, 4), c(1, 1, 2), c(1, 3, 3), c(1, 1, 1), c(2, 3, 3))

@param df is the input data.frame @param val_to_stop is the vector containing the values to stop @param index_rc is the value for the jump (default set to NA so default will be first case) @param included is if the values to stop has to be also returned in the vectors (defaultn set to "yes")

see_df

Allow to return a datafame with TRUE cells where the condition entered are respected and FALSE where these are not

@param df is the input dataframe @param condition_l is the vector of the possible conditions ("==", ">", "<", "!=", "%") @param val_l is the vector with the values related to condition_l (so the values has to be placed in the same order)

days_from_month

Allow to find the number of days month from a month date, take in count leap year @param date_ is the input date @param sep_ is the separator of the input date

vec_in_df

Allow to see if vectors are present in a dataframe

ex: 1, 2, 1 3, 4, 1 1, 5, 8

the vector c(4, 1) with the coefficient 1 and the start position at the second column is containded in the dataframe

@param df_ is the input dataframe @param vec_l is a list the vectors @param coeff_ is the related coefficient of the vector @param strt_l is a vector containing the start position for each vector eturns the number of row that contains vectors (multiple in a list), coeff and strt_l may vary istinct is the value you are sure is not contained in the matrix

closest_date

return the closest dates from a vector compared to the input date

@param date_ is the input date @param vec is a vector containing the dates to be compared to the input date @param frmt is the format of the input date, (deault set to "snhdmy" (second, minute, hour, day, month, year), so all variable are taken in count), if you only want to work with standard date for example change this variable to "dmy" @param sep_ is the separator for the input date @param sep_vec is the separator for the dates contained in vec @param only is can be changed to "+" or "-" to repectively only return the higher dates and the lower dates (default set to "both") @param head is the number of dates that will be returned (default set to NA so all dates in vec will be returned)

change_date

Allow to add to a date second-minute-hour-day-month-year

@param date_ is the input date @param sep_ is the date separator @param day_ is the day to add (can be negative) @param month_ is the month to add (can be negative) @param year_ is the year to add (can be negative) @param hour_ is the hour to add (can be negative) @param min_ is the minute to add (can be negative) @param second_ is the second to add (can be negative) @param frmt is the format of the input date, (deault set to "snhdmy" (second, minute, hour, day, month, year), so all variable are taken in count), if you only want to work with standard date for example change this variable to "dmy"

pattern_gettr

Search for pattern(s) contained in a vector in another vector and return matched one and their position according to these rules:

First case: Search for patterns strictly, it means that the searched pattern(s) will be matched only if the patterns containded in the vector that is beeing explored by the function are present like this c("pattern_searched", "other", . . . , "pattern_searched") and not as c("other_thing pattern_searched other_thing", "other", . . . , "pattern_searched other_thing") Second case: It is the opposite to the first case, it means that if the pattern is partially present like in the first position and the last, it will be considered like a matched pattern @param word_ is the vector containing the patterns @param vct is the vector being searched for patterns @param occ a vector containing the occurence of the pattern in word_ to be matched in the vector being searched, if the occurence is 2 for the nth pattern in word_ and only one occurence is found in vct so no pattern will be matched @param strict a vector containing the "strict" condition for each nth vector in word_ ("strict" is the string to activate this option) @param btwn is a vector containing the condition ("yes" to activate this option) meaning that if "yes", all elements between two matched patern in vct will be returned @param all_in_word is a value (default set to "no", "yes" to activate this option) that, if activated, won't authorized a previous matched pattern to be matched again