# Package 'jmvReadWrite'

November 4, 2026

**Title** Read and Write 'jamovi' Files ('.omv')

**Version** 0.4.15

**Author** Sebastian Jentschke [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-2576-5432>)

**Maintainer** Sebastian Jentschke <sebastian.jentschke@uib.no>

**Description** The free and open a statistical spreadsheet 'jamovi' (<https://www.jamovi.org>) aims to make statistical analyses easy and intuitive. 'jamovi' produces syntax that can directly be used in R (in connection with the R-package 'jmv'). Having import / export routines for the data files 'jamovi' produces ('.omv') permits an easy transfer of data and analyses between 'jamovi' and R.

**License** AGPL-3

**LazyData** true

**Language** en-GB

**Encoding** UTF-8

**VignetteBuilder** knitr

**URL** https://sjentsch.github.io/jmvReadWrite/, https://github.com/sjentsch/jmvReadWrite

**BugReports** https://github.com/sjentsch/jmvReadWrite/issues

**Depends** R (>= 3.5.0)

**Imports** jsonlite, methods, zip

**Suggests** jmv, jmvcore (>= 2.4.7), foreign, haven, knitr, rmarkdown, RProtoBuf, testthat (>= 3.0.0)

**Config/testthat/edition** 3

**NeedsCompilation** no

**Config/roxygen2/version** 8.1.1

**Repository** CRAN

**Date/Publication** 2026-11-04 01:00:16 UTC

# Contents

aggregate_omv . . . 2
AlbumSales . . . 6
arrange_cols_omv . . . 7
bfi_sample . . . 9
bfi_sample2 . . . 11
bfi_sample3 . . . 12
combine_cols_omv . . . 14
convert_to_omv . . . 16
describe_omv . . . 18
distances_omv . . . 21
label_vars_omv . . . 25
long2wide_omv . . . 28
merge_cols_omv . . . 31
merge_rows_omv . . . 33
read_omv . . . 36
replace_omv . . . 38
search_omv . . . 40
sort_omv . . . 42
ToothGrowth . . . 44
transform_vars_omv . . . 45
transpose_omv . . . 47
wide2long_omv . . . 49
write_omv . . . 52

**Index** **54**

---

`aggregate_omv` *Aggregates data from a data set / frame (in long format) and returns them as a .omv-file for the statistical spreadsheet 'jamovi' (https://www.jamovi.org)*

---

## Description

Aggregates data from a data set / frame (in long format) and returns them as a .omv-file for the statistical spreadsheet 'jamovi' (https://www.jamovi.org)

## Usage

```
aggregate_omv(
  dtaInp = NULL,
  fleOut = "",
  varAgg = NULL,
  grpAgg = NULL,
  clcN = FALSE,
  clcMss = FALSE,
  clcMn = FALSE,
```

```
  clcMdn = FALSE,
  clcMde = FALSE,
  clcSum = FALSE,
  clcSD = FALSE,
  clcVar = FALSE,
  clcRng = FALSE,
  clcMin = FALSE,
  clcMax = FALSE,
  clcIQR = FALSE,
  drpNA = TRUE,
  varAdj = TRUE,
  usePkg = c("foreign", "haven"),
  selSet = "",
  ...
)
```

## Arguments

| | |
|---|---|
| dtaInp | Either a data frame or the name of a data file to be read (including the path, if required; "FILENAME.ext"; default: NULL); files can be of any supported file type, see Details below. |
| fleOut | Name of the data set / file to be written (including the path, if required; "FILE_OUT.omv"; default: ""); if empty, the resulting data frame is returned instead. |
| varAgg | A character vector (default: NULL) with the names of the variables which shall be aggregated. |
| grpAgg | A character vector (default: NULL) with the variables to group the aggregation variable (varAgg) by. If no grouping variable is given, the aggregation happens over the whole data set. If several grouping variables are given, the aggregation happens for each step / possible combination of these variables. See Details for more information. |
| clcN | If TRUE, counts the number of valid values for each step / combination of values in the grouping variable. The suffix appended '"_N"´ is appended to each variable in the resulting data set. |
| clcMss | If TRUE, counts the number of missing values for each step / combination of values in the grouping variable. The suffix "_Mss" is appended to each variable in the resulting data set. |
| clcMn | If TRUE, calculates the mean for each step / combination of values in the grouping variable. The suffix "_Mn" is appended to each variable in the resulting data set. |
| clcMdn | If TRUE, calculates the median for each step / combination of values in the grouping variable. The suffix "_Mdn" is appended to each variable in the resulting data set. |
| clcMde | If TRUE, calculates the mode for each step / combination of values in the grouping variable. The suffix "_Mde" is appended to each variable in the resulting data set. |

| | |
|---|---|
| clcSum | If TRUE, calculates the sum for each step / combination of values in the grouping variable. The suffix "_Sum" is appended to each variable in the resulting data set. |
| clcSD | If TRUE, calculates the std. deviation for each step / combination of values in the grouping variable. The suffix "_SD" is appended to each variable in the resulting data set. |
| clcVar | If TRUE, calculates the variance for each step / combination of values in the grouping variable. The suffix "_Var" is appended to each variable in the resulting data set. |
| clcRng | If TRUE, calculates the range for each step / combination of values in the grouping variable. The suffix "_Rng" is appended to each variable in the resulting data set. |
| clcMin | If TRUE, determines the minimum at each step / combination of values in the grouping variable. The suffix "_Min" is appended to each variable in the resulting data set. |
| clcMax | If TRUE, determines the maximum at each step / combination of values in the grouping variable. The suffix "_Max" is appended to each variable in the resulting data set. |
| clcIQR | If TRUE, calculates the IQR for each step / combination of values in the grouping variable. The suffix "_IQR" is appended to each variable in the resulting data set. |
| drpNA | If TRUE (default: TRUE), NA values are removed before the aggregation, and the aggregation is calculated using the valid values. If FALSE, the result would be NA for any step / combination of values in the grouping variable that contains a NA value. |
| varAdj | If TRUE (default: TRUE), all calculations for one variable are kept next to each other (e.g., "V1_N", "V1_Mss", etc.), otherwise the same calculations are kept next to each other (e.g., "V1_N", "V2_N", etc.). |
| usePkg | Name of the package: "foreign" or "haven" that shall be used to read SPSS, Stata, and SAS files; "foreign" is the default (it is included in base R), but "haven" is newer and more comprehensive; you may have to install using `install.packages("haven", dep = TRUE)`. |
| selSet | Name of the object / data set that is to be selected from the workspace (only relevant when reading .RData-files which can contain several objects / data sets) |
| ... | Additional arguments passed on to methods; see Details below. |

## Details

- varAgg must be a character vector containing the variables to be aggregated. From these variables, a new data set is created where the values in the original data set are aggregated for each possible combination of steps in the grouping variable(s) given in grpAgg. For example, if one grouping variable were a participant ID, and another grouping variable were measurement time points, the resulting new dataset would contain as many rows as the number of participants multiplied by the number of measurement time points. For each variable in varAgg, and each possible aggregation (clcN, clcMn, etc. if set to TRUE), a new column would be generated in the resulting new data set.

- drpNA determines whether NA should be dropped / removed before calculating an aggregation. If set to FALSE, the result for a combination of steps in the grouping variable(s) where at least one value is NA would be NA. If set to TRUE, only the values for any given combination of steps in the grouping variable(s) that are not NA are considered for the calculation. For clcN and clcMss, drpNA has no effect. NB: If drpNA is set to TRUE, any row where any of the grouping variable(s) has the value NA is excluded from the aggregation, if drpNA is set to FALSE and there is any row where any of the group variables has the value NA, an error is thrown and no aggregation is carried out.
- The ellipsis-parameter (. . .) can be used to submit arguments / parameters to the functions that are used for reading or transforming the data. By clicking on the respective function under "See also", you can get a more detailed overview over which parameters each of those functions take.

## Value

a data frame containing a symmetric matrix (only returned if fleOut is empty) containing the distances between the variables / columns (clmDst == TRUE) or rows (clmDst == FALSE)

## See Also

aggregate_omv internally uses stats::aggregate() as function for the aggregation, and base::is.na(), base::mean(), stats::median(), base::sum(), stats::sd(), stats::var(), base::range(), base::min(), base::max(), and stats::quantile() for calculation. It furthermore uses the following functions for reading and writing data files in different formats: read_omv() and write_omv() for jamovi-files, utils::read.table() for CSV / TSV files, load() for reading .RData-files, readRDS() for .rds-files, haven::read_sav() or foreign::read.spss() for SPSS-files, haven::read_dta() or foreign::read.dta() for Stata-files, haven::read_sas() for SAS-data-files, and haven::read_xpt() or foreign::read.xport() for SAS-transport-files.

## Examples

```
# generate a test dataframe with 100 (imaginary) participants / units of
#  observation (ID), 10 measurement (measure) of two variables (V1, V2)
dtaInp <- data.frame(ID = rep(as.character(seq(1, 100)), each = 10),
                     Measure = rep(seq(1, 10), times = 100),
                     V1 = runif(1000,   0, 100),
                     V2 = round(rnorm(1000, 100,  15)))
cat(str(dtaInp))
# the output should look like this
# 'data.frame': 1000 obs. of  4 variables:
#  $ ID     : chr  "1" "1" "1" "1" ...
#  $ Measure: int  1 2 3 4 5 6 7 8 9 10 ...
#  $ V1     : num  ...
#  $ V2     : num  ...
# this data set is stored as (temporary) RDS-file and later processed by long2wide
nmeInp <- tempfile(fileext = ".rds")
nmeOut <- tempfile(fileext = ".omv")
saveRDS(dtaInp, nmeInp)
jmvReadWrite::aggregate_omv(dtaInp = nmeInp, fleOut = nmeOut, varAgg = c("V1", "V2"),
                            grpAgg = "ID", clcN = TRUE, clcMn = TRUE, clcSD = TRUE)
```

```
# it is required to give at least the arguments dtaInp, varAgg and grpAgg, each of
# the different switches (clc...) requests a aggregation measure (e.g., mean, median,
# SD, IQR, etc.) to be calculated
# check whether the file was created and its size
cat(list.files(dirname(nmeOut), basename(nmeOut)))
# -> "file[...].omv" ([...] contains a random combination of numbers / characters
cat(file.info(nmeOut)$size)
# -> 4898 (approximate size; size may differ in every run [in dependence of
#          how well the generated random data can be compressed])
cat(str(jmvReadWrite::read_omv(nmeOut, sveAtt = FALSE)))
# the data set contains now the ID variable identifying the different steps of
# aggregation and one column for each combination of aggregation variable (V1 / V2)
# and which calculation was requested (N, mean and SD)
# 'data.frame': 100 obs. of  7 variables:
#  $ ID   : chr  "1" "10" "100" "11" ...
#   ..- attr(*, "jmv-id")= logi TRUE
#   ..- attr(*, "missingValues")= list()
#  $ V1_N : int  10 10 10 10 10 10 10 10 10 10 ...
#   ..- attr(*, "jmv-desc")= chr "V1 (N)"
#   ..- attr(*, "missingValues")= list()
#  $ V1_Mn: num  45.4 51.9 49.4 54.8 47.2 ...
#   ..- attr(*, "jmv-desc")= chr "V1 (Mean)"
#   ..- attr(*, "missingValues")= list()
#  $ V1_SD: num  31.7 31.4 26.5 20.2 29.1 ...
#   ..- attr(*, "jmv-desc")= chr "V1 (SD)"
#   ..- attr(*, "missingValues")= list()
#  $ V2_N : int  10 10 10 10 10 10 10 10 10 10 ...
#   ..- attr(*, "jmv-desc")= chr "V2 (N)"
#   ..- attr(*, "missingValues")= list()
#  $ V2_Mn: num  96.4 102.3 101.6 104.6 108.7 ...
#   ..- attr(*, "jmv-desc")= chr "V2 (Mean)"
#   ..- attr(*, "missingValues")= list()
#  $ V2_SD: num  14.8 18.4 11.2 10.1 14.3 ...
#   ..- attr(*, "jmv-desc")= chr "V2 (SD)"
#   ..- attr(*, "missingValues")= list()

unlink(nmeInp)
unlink(nmeOut)
```

---

`AlbumSales` *Imagine that you worked for a record company and that your boss was interested in predicting album sales from advertising.*

---

## Description

The data set is fictional and was constructed by Andy Field who therefore owns the copyright. The data set is also publicly available on the website that accompanies Andy Field's book, https://edge.sagepub.com/field5e. Without Andy Field's explicit consent, this data set may not be distributed for commercial purposes,

this data set may not be edited, and this data set may not be presented without acknowledging its source (i.e., the terms of a CC BY-NC-ND license).

## Usage

```
AlbumSales
```

## Format

A data frame with 60 rows, each one representing a different album, and 5 variables:

`selSbj` Select the data in this row (1) or not (0)

`Adverts` Amount (in thousands of pounds) spent promoting the album before release

`Airplay` How many times songs from the album were played on a prominent national radio station in the week before release

`Image` How attractive people found the band's image (out of 10)

`Sales` Sales (in thousands) of each album in the week after release

## Details

Reference: Field, A. P. (2017). Discovering Statistics Using IBM SPSS Statistics (5th ed.). Sage.

---

`arrange_cols_omv` *Re-arrange columns / variables in .omv-files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org)*

---

## Description

Re-arrange columns / variables in .omv-files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org)

## Usage

```
arrange_cols_omv(
  dtaInp = NULL,
  fleOut = "",
  varOrd = NULL,
  varMve = list(),
  psvAnl = FALSE,
  usePkg = c("foreign", "haven"),
  selSet = "",
  ...
)
```

## Arguments

| | |
|---|---|
| dtaInp | Either a data frame or the name of a data file to be read (including the path, if required; "FILENAME.ext"; default: NULL); files can be of any supported file type, see Details below. |
| fleOut | Name of the data set / file to be written (including the path, if required; "FILE_OUT.omv"; default: ""); if empty, the resulting data frame is returned instead. |
| varOrd | Character vector with the desired order of variable(s) in the data frame (see Details; default: NULL) |
| varMve | Named list defining to how much a particular variable (name of a list entry) should be moved up (neg. value of a list entry) or down (pos. value) in the data frame (see Details; default: NULL) |
| psvAnl | Whether analyses that are contained in the input file shall be transferred to the output file (TRUE / FALSE; default: FALSE) |
| usePkg | Name of the package: "foreign" or "haven" that shall be used to read SPSS, Stata, and SAS files; "foreign" is the default (it is included in base R), but "haven" is newer and more comprehensive; you may have to install using `install.packages("haven", dep = TRUE)`. |
| selSet | Name of the object / data set that is to be selected from the workspace (only relevant when reading .RData-files which can contain several objects / data sets) |
| ... | Additional arguments passed on to methods; see Details below |

## Details

- `varOrd` is a character vector. If not all variables of the original data set are contained in `varOrd`, a warning is issued but otherwise the list of variables defined in `varOrd` is used (removing variables not contained in `varOrd`).
- `varMve` is a named list. For example would `list(VARNAME = -3)` move the variable `VARNAME` three positions up in the list of variables (towards the first column), and `list(VARNAME = 3)` would move it three positions down (towards the last column). If the number of steps the variable is to be moved leads to the position being either lower than the first or higher than the total number of variables in the data set, an error message is issued. Please note that the list entries are processed one after another, that is, for a second list entry, you have to consider how the first list entry may have changed to order of variables.
- Using `varOrd` makes more sense for changing the position of several variables, whereas using `varMve` makes more sense for one variable. If both parameters are given, a warning is issued and `varOrd` takes precedence.
- The ellipsis-parameter (`...`) can be used to submit arguments / parameters to the functions that are used for reading the data. By clicking on the respective function under "See also", you can get a more detailed overview over which parameters each of those functions take.

## Value

a data frame (only returned if `fleOut` is empty) where the order of variables / columns of the input data set is re-arranged

## See Also

arrange_cols_omv internally uses the following functions for reading and writing data files in different formats: read_omv() and write_omv() for jamovi-files, utils::read.table() for CSV / TSV files, load() for reading .RData-files, readRDS() for .rds-files, haven::read_sav() or foreign::read.spss() for SPSS-files, haven::read_dta() or foreign::read.dta() for Stata-files, haven::read_sas() for SAS-data-files, and haven::read_xpt() or foreign::read.xport() for SAS-transport-files.

## Examples

```
nmeInp <- system.file("extdata", "AlbumSales.omv", package = "jmvReadWrite")
nmeOut <- tempfile(fileext = ".omv")
# the original file has the variables in the order: "Adverts", "Airplay", "Image", "Sales"
names(read_omv(nmeInp))
# first, we move the variable "Sales" to the first place using the varOrd-parameter
jmvReadWrite::arrange_cols_omv(dtaInp = nmeInp, fleOut = nmeOut,
  varOrd = c("Sales", "Adverts", "Airplay", "Image"))
names(jmvReadWrite::read_omv(nmeOut))
unlink(nmeOut)
# now, we move the variable "Sales" to the first place using the varMve-parameter
jmvReadWrite::arrange_cols_omv(dtaInp = nmeInp, fleOut = nmeOut, varMve = list(Sales = -3))
names(jmvReadWrite::read_omv(nmeOut))
unlink(nmeOut)
```

---

# bfi_sample — *Twenty-five personality self-report items taken from the International Personality Item Pool*

---

## Description

The data set contains responses from 250 participants filling in twenty-five personality self-report items taken from the International Personality Item Pool (https://ipip.ori.org) as part of the Synthetic Aperture Personality Assessment (SAPA) web-based personality assessment (https://sapa-project.org) project. The 25 items are organized by five putative factors: Agreeableness (A1 to A5), Conscientiousness (C1 to C5), Extraversion (E1 to E5), Neuroticism (N1 to N5), and Openness (N1 to N5). The items were short phrases that the respondent should answer by indicating how accurately the statement describes their typical behaviour or attitude. Responses were collected using a 6-point scale: 1 - Very inaccurate, 2 - Moderately inaccurate, 3 - Slightly inaccurate, 4 - Slightly accurate, 5 - Moderately accurate, 6 - Very accurate.

## Usage

```
bfi_sample
```

## Format

A data frame with 254 rows (250 original respondents, 4 generated for testing) and 33 variables:

`ID` Respondent ID

`A1` Am indifferent to the feelings of others. (reversed)

`A2` Inquire about others' well-being.

`A3` Know how to comfort others.

`A4` Love children.

`A5` Make people feel at ease.

`C1` Am exacting in my work.

`C2` Continue until everything is perfect.

`C3` Do things according to a plan.

`C4` Do things in a half-way manner. (reversed)

`C5` Waste my time. (reversed)

`E1` Don't talk a lot. (reversed)

`E2` Find it difficult to approach others. (reversed)

`E3` Know how to captivate people.

`E4` Make friends easily.

`E5` Take charge.

`N1` Get angry easily.

`N2` Get irritated easily.

`N3` Have frequent mood swings.

`N4` Often feel blue.

`N5` Panic easily.

`O1` Am full of ideas.

`O2` Avoid difficult reading material. (reversed)

`O3` Carry the conversation to a higher level.

`O4` Spend time reflecting on things.

`O5` Will not probe deeply into a subject. (reversed)

`gender` Gender of the respondent (female, male)

`age` Age of the respondent (years)

`AD` Exponent of age (computed: EXP(age))

`AF` Random data (for testing)

`AG` Random data (for testing)

`age_tr` Age of the respondent (transformed, as decades: 1 - 10-19, 2 - 20-29, 3 - 30-39, 4 - 40-49, 5 - 50-59, 6 - 60 and over)

`ID2` Respondent ID (for testing; "A" + random-generated 5-digit-code)

`bfi_sample2` *Twenty-five personality self-report items taken from the International Personality Item Pool (includes jamovi-attributes; should result in a file identical to bfi_sample2.omv under "extdata" when written with write_omv)*

## Description

The data set contains responses from 250 participants filling in twenty-five personality self-report items taken from the International Personality Item Pool (https://ipip.ori.org) as part of the Synthetic Aperture Personality Assessment (SAPA) web-based personality assessment (https://sapa-project.org) project. The 25 items are organized by five putative factors: Agreeableness (A1 to A5), Conscientiousness (C1 to C5), Extraversion (E1 to E5), Neuroticism (N1 to N5), and Openness (N1 to N5). The items were short phrases that the respondent should answer by indicating how accurately the statement describes their typical behaviour or attitude. Responses were collected using a 6-point scale: 1 - Very inaccurate, 2 - Moderately inaccurate, 3 - Slightly inaccurate, 4 - Slightly accurate, 5 - Moderately accurate, 6 - Very accurate. The data set includes the jamovi-attributes. It is supposed to result in an identical file compared to the bfi_sample2.omv-file contained in the extdata-directory of the package when written with write_omv.

## Usage

```
bfi_sample2
```

## Format

A data.frame with 250 rows and 29 variables

`ID` Respondent ID

`A1` Am indifferent to the feelings of others. (reversed)

`A2` Inquire about others' well-being.

`A3` Know how to comfort others.

`A4` Love children.

`A5` Make people feel at ease.

`C1` Am exacting in my work.

`C2` Continue until everything is perfect.

`C3` Do things according to a plan.

`C4` Do things in a half-way manner. (reversed)

`C5` Waste my time. (reversed)

`E1` Don't talk a lot. (reversed)

`E2` Find it difficult to approach others. (reversed)

`E3` Know how to captivate people.

`E4` Make friends easily.

E5 Take charge.

N1 Get angry easily.

N2 Get irritated easily.

N3 Have frequent mood swings.

N4 Often feel blue.

N5 Panic easily.

O1 Am full of ideas.

O2 Avoid difficult reading material. (reversed)

O3 Carry the conversation to a higher level.

O4 Spend time reflecting on things.

O5 Will not probe deeply into a subject. (reversed)

gender Gender of the respondent (female, male)

age Age of the respondent (years)

ID2 Respondent ID (for testing; "A" + random-generated 4-digit-code)

---

bfi_sample3 *Twenty-five personality self-report items taken from the International Personality Item Pool (testing file for ordered factors / "Ordinal"-variables in jamovi)*

---

## Description

The data set contains responses from 250 participants filling in twenty-five personality self-report items taken from the International Personality Item Pool (https://ipip.ori.org) as part of the Synthetic Aperture Personality Assessment (SAPA) web-based personality assessment (https://sapa-project.org) project. The 25 items are organized by five putative factors: Agreeableness (A1 to A5), Conscientiousness (C1 to C5), Extraversion (E1 to E5), Neuroticism (N1 to N5), and Openness (N1 to N5). The items were short phrases that the respondent should answer by indicating how accurately the statement describes their typical behaviour or attitude. Responses were collected using a 6-point scale: 1 - Very inaccurate, 2 - Moderately inaccurate, 3 - Slightly inaccurate, 4 - Slightly accurate, 5 - Moderately accurate, 6 - Very accurate. The data set includes the jamovi-attributes. It is supposed to result in an identical file compared to the bfi_sample2.omv-file contained in the extdata-directory of the package when written with write_omv.

## Usage

```
bfi_sample3
```

**Format**

A data.frame with 250 rows and 28 variables

`ID` Respondent ID

`A1` Am indifferent to the feelings of others. (reversed)

`A2` Inquire about others' well-being.

`A3` Know how to comfort others.

`A4` Love children.

`A5` Make people feel at ease.

`C1` Am exacting in my work.

`C2` Continue until everything is perfect.

`C3` Do things according to a plan.

`C4` Do things in a half-way manner. (reversed)

`C5` Waste my time. (reversed)

`E1` Don't talk a lot. (reversed)

`E2` Find it difficult to approach others. (reversed)

`E3` Know how to captivate people.

`E4` Make friends easily.

`E5` Take charge.

`N1` Get angry easily.

`N2` Get irritated easily.

`N3` Have frequent mood swings.

`N4` Often feel blue.

`N5` Panic easily.

`O1` Am full of ideas.

`O2` Avoid difficult reading material. (reversed)

`O3` Carry the conversation to a higher level.

`O4` Spend time reflecting on things.

`O5` Will not probe deeply into a subject. (reversed)

`gender` Gender of the respondent (Females, Males)

`age` Age of the respondent (years)

combine_cols_omv *Combines pairs of columns from a raw data matrix in .omv-files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org)*

## Description

Combines pairs of columns from a raw data matrix in .omv-files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org)

## Usage

```
combine_cols_omv(
  dtaInp = NULL,
  fleOut = "",
  varPrs = list(),
  mdeCmb = c("none", "first", "second"),
  psvAnl = FALSE,
  usePkg = c("foreign", "haven"),
  selSet = "",
  ...
)
```

## Arguments

| | |
|---|---|
| dtaInp | Either a data frame or the name of a data file to be read (including the path, if required; "FILENAME.ext"; default: NULL); files can be of any supported file type, see Details below. |
| fleOut | Name of the data set / file to be written (including the path, if required; "FILE_OUT.omv"; default: ""); if empty, the resulting data frame is returned instead. |
| varPrs | Definition of variable pairs; a list containing either list(s) or character vector(s) with the names of pairs of variables to be combined (default: list()). |
| mdeCmb | Mode of combining the variables when conflicting values occur, either "none", "first", or "second" (default: "none"), see Details below. |
| psvAnl | Whether analyses that are contained in the input file shall be transferred to the output file (TRUE / FALSE; default: FALSE) |
| usePkg | Name of the package: "foreign" or "haven" that shall be used to read SPSS, Stata, and SAS files; "foreign" is the default (it is included in base R), but "haven" is newer and more comprehensive; you may have to install using `install.packages("haven", dep = TRUE)`. |
| selSet | Name of the object / data set that is to be selected from the workspace (only relevant when reading .RData-files which can contain several objects / data sets) |
| ... | Additional arguments passed on to methods; see Details below. |

## Details

- The need to combine two columns into one is quite common after merging columns or rows (using, e.g., `merge_cols_omv` or `merge_rows_omv`). `varPrs` defines the variable pairs to be combined. It is a list containing the pairs of variables to be combined, either as list or as character vector; e.g., list(c("A", "B"), c("C", "D")) or list(list("A", "B"), list("C", "D")). `mdeCmb` defines what to to if values in the first and the second variable of a variable pair contain conflicting / different values: "none" does not merge the variables (and instead throws an error), "first" makes that the values from the first variable of each pair are taken if the values are conflicting, and "second" use the values from the second variable of each pair in case of conflicts.
- The ellipsis-parameter (`...`) can be used to submit arguments / parameters to the functions that are used for reading the data. By clicking on the respective function under "See also", you can get a more detailed overview over which parameters each of those functions take.

## Value

a data frame containing the column pairs given in `varPrs` combined and the original columns removed

## See Also

`combine_cols_omv` uses the following functions for reading and writing data files in different formats: `read_omv()` and `write_omv()` for jamovi-files, `utils::read.table()` for CSV / TSV files, `load()` for reading .RData-files, `readRDS()` for .rds-files, `haven::read_sav()` or `foreign::read.spss()` for SPSS-files, `haven::read_dta()` or `foreign::read.dta()` for Stata-files, `haven::read_sas()` for SAS-data-files, and `haven::read_xpt()` or `foreign::read.xport()` for SAS-transport-files.

## Examples

```
dtaInp <- jmvReadWrite::bfi_sample2
# create a new column (A1_1) containing a subset of the values in the original variable
# whereas those lines are replaced with NAs
set.seed(1)
selRow <- rnorm(nrow(dtaInp)) < 0
dtaInp[selRow,  "A1_1"] <- dtaInp[selRow, "A1"]
dtaInp[selRow,  "A1"]   <- NA
head(dtaInp[, c("A1", "A1_1")])
dtaOut <- combine_cols_omv(dtaInp, varPrs = list(c("A1", "A1_1")))
# show the differences before and after combining the values in the columns and ensure
# that all values are the same as in the original data set
dtaInp[, "A1"]
dtaOut[, "A1"]
all(dtaOut[, "A1"] == jmvReadWrite::bfi_sample2[, "A1"])

# create a new column, containing values that are different from the original variable
dtaInp <- jmvReadWrite::bfi_sample2
dtaInp[selRow,  "A1_1"] <- dtaInp[selRow,  "A1"] + 1
# [1] if mdeCmb is "none" (or if mdeCmb is not given - "none" is the default) an error would be
# thrown (therefore the next line is commented out)
# dtaOut <- combine_cols_omv(dtaInp, varPrs = list(c("A1", "A1_1")), mdeCmb = "none")
# [2] if mdeCmb is "first", missing values are replaced and values from the first column ("A1")
```

```
# take precedence if the values are unequal
dtaOut <- combine_cols_omv(dtaInp, varPrs = list(c("A1", "A1_1")), mdeCmb = "first")
head(cbind(dtaOut[, "A1"], dtaInp[, c("A1", "A1_1")]))
# [3] if mdeCmb is "second", missing values are replaced and values from the second column
# ("A1_1") take precedence if the values are unequal
dtaOut <- combine_cols_omv(dtaInp, varPrs = list(c("A1", "A1_1")), mdeCmb = "second")
head(cbind(dtaOut[, "A1"], dtaInp[, c("A1", "A1_1")]))
```

---

`convert_to_omv` *Convert data files (CSV, R, other statistics packages) into .omv-files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org)*

---

## Description

Convert data files (CSV, R, other statistics packages) into .omv-files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org)

## Usage

```
convert_to_omv(
  fleInp = "",
  fleOut = "",
  varSrt = NULL,
  usePkg = c("foreign", "haven"),
  selSet = "",
  ...
)
```

## Arguments

| | |
|---|---|
| `fleInp` | Name (including the path, if required) of the data file to be read ("FILENAME.ext"; default: ""); supports CSV and R-files natively, or other file types if "foreign" or "haven" are installed, see Details below |
| `fleOut` | Name (including the path, if required) of the data file to be written ("FILENAME.omv"; default: ""); if empty, the extension of fleInp is replaced with ".omv" |
| `varSrt` | Variable(s) that are used to sort the data frame (see Details; if empty, the row order of the input file is kept; default: NULL) |
| `usePkg` | Name of the package: "foreign" or "haven" that shall be used to read SPSS, Stata and SAS files; "foreign" is the default (it comes with base R), but "haven" is newer and more comprehensive |
| `selSet` | Name of the data set that is to be selected from the workspace (only applies when reading .RData-files) |
| `...` | Additional arguments passed on to methods; see Details below |

## Details

- In difference to the remaining helper functions, `convert_to_omv` doesn't accept a data frame as input and it neither does return a data frame if `fleOut` is left empty: If you want to write a data frame, use `write_omv`. If you want to have a data frame returned use `read_omv` for jamovi-files or any of the functions listed in the bullet point below for any other file type.
- `varSrt` can be either a character or a character vector (with one or more variables respectively). The sorting order for a particular variable can be inverted with preceding the variable name with "-". Please note that this doesn't make sense and hence throws a warning for certain variable types (e.g., factors).
- The ellipsis-parameter (`...`) can be used to submit arguments / parameters to the functions that are used for reading and writing the data. By clicking on the respective function under "See also", you can get a more detailed overview over which parameters each of those functions take. The functions are: `read_omv` and `write_omv` (for jamovi-files), `read.table` (for CSV / TSV files; using similar defaults as `read.csv` for CSV and `read.delim` for TSV which both are based upon `read.table`), `load` (for .RData-files), `readRDS` (for .rds-files), `read_sav` (needs the R-package `haven`) or `read.spss` (needs the R-package `foreign`) for SPSS-files, `read_dta` (`haven`) / `read.dta` (`foreign`) for Stata-files, `read_sas` (`haven`) for SAS-data-files, and `read_xpt` (`haven`) / `read.xport` (`foreign`) for SAS-transport-files. If you would like to use `haven`, you may need to install it using `install.packages("haven", dep = TRUE)`.

## Value

the function doesn't have a return value (it returns NULL)

## See Also

`convert_to_omv` internally uses the following functions for reading and writing data files in different formats: `read_omv()` and `write_omv()` for jamovi-files, `utils::read.table()` for CSV / TSV files, `load()` for reading .RData-files, `readRDS()` for .rds-files, `haven::read_sav()` or `foreign::read.spss()` for SPSS-files, `haven::read_dta()` or `foreign::read.dta()` for Stata-files, `haven::read_sas()` for SAS-data-files, and `haven::read_xpt()` or `foreign::read.xport()` for SAS-transport-files.

## Examples

```
# Example 1: Convert from RDS
# (use ToothGrowth as example, save it as RDS)
nmeInp <- tempfile(fileext = ".rds")
nmeOut <- tempfile(fileext = ".omv")
saveRDS(jmvReadWrite::ToothGrowth, nmeInp)
jmvReadWrite::convert_to_omv(fleInp = nmeInp, fleOut = nmeOut)
cat(list.files(dirname(nmeOut), basename(nmeOut)))
# -> "file[...].omv" ([...] contains a random combination of numbers / characters
cat(file.info(nmeOut)$size)
# -> 2448 (size may differ on different OSes)
cat(str(jmvReadWrite::read_omv(nmeOut, sveAtt = FALSE)))
# gives a overview of the dataframe (all columns and some attributes,
# sveAtt is intentionally set to FALSE to make the output not too overwhelming)
```

```
unlink(nmeInp)
unlink(nmeOut)

# Example 2: Convert from CSV
# (use ToothGrowth again as example, this time save it as CSV)
nmeInp <- tempfile(fileext = ".csv")
nmeOut <- tempfile(fileext = ".omv")
write.csv(jmvReadWrite::ToothGrowth, nmeInp)
jmvReadWrite::convert_to_omv(fleInp = nmeInp, fleOut = nmeOut)
cat(list.files(dirname(nmeOut), basename(nmeOut)))
cat(file.info(nmeOut)$size)
# -> 2104 (size may differ acc. to OS; the size is smaller than for the RDS-file
# because CSV can store fewer attributes, e.g., labels)
cat(str(jmvReadWrite::read_omv(nmeOut, sveAtt = FALSE)))
# gives a overview of the dataframe (all columns and some attributes)
unlink(nmeInp)
unlink(nmeOut)
```

---

`describe_omv` *Adds a title and a description for a data set stored as .omv-file for the statistical spreadsheet 'jamovi' (https://www.jamovi.org)*

---

## Description

Adds a title and a description for a data set stored as .omv-file for the statistical spreadsheet 'jamovi' (https://www.jamovi.org)

## Usage

```
describe_omv(
  dtaInp = NULL,
  fleOut = "",
  dtaTtl = NULL,
  dtaDsc = NULL,
  lngDsc = "EN",
  usePkg = c("foreign", "haven"),
  selSet = "",
  ...
)
```

## Arguments

| | |
|---|---|
| `dtaInp` | Either a data frame or the name of a data file to be read (including the path, if required; "FILENAME.ext"; default: NULL); files can be of any supported file type, see Details below. |
| `fleOut` | Name of the data set / file to be written (including the path, if required; "FILE_OUT.omv"; default: ""); if empty, the resulting data frame is returned instead. |

| | |
|---|---|
| dtaTtl | Character vector with a title to be added to the data set (see Details; default: "") |
| dtaDsc | Description of the data set, either as character vector (HTML-formatted) or as named list with the entries "description", "variables", "references", and "license" (see Details; default: "") |
| lngDsc | Language of the description (localizes the description components: "Description", "Variables", "References", and "License"; default: "EN") |
| usePkg | Name of the package: "foreign" or "haven" that shall be used to read SPSS, Stata, and SAS files; "foreign" is the default (it is included in base R), but "haven" is newer and more comprehensive; you may have to install using `install.packages("haven", dep = TRUE)`. |
| selSet | Name of the object / data set that is to be selected from the workspace (only relevant when reading .RData-files which can contain several objects / data sets) |
| ... | Additional arguments passed on to methods; see Details below |

## Details

- The aim of this function is to add a title and a data set description to jamovi data files. Two typical use cases would be (1) to help creating data sets to be used in teaching (i.e., either creating or using data sets in R, and afterwards adding a description to those), and (2) to provide "properly described" data when publishing in a repository, e.g., the OSF).
- NB: The data set should not contain any existing analyses. These will be overwritten (a warning is issued informing you about that).
- `dtaTtl` is a title for the dataset (at the top of the results output, i.e., that title which initially is "Results" when you create a new data set in jamovi).
- `dtaDsc` can either be a character vector (with length = 1) containing HTML-formatted text that describes the data set (see `chrDsc` in the examples for HTML tags that are currently implemented; putting "unformatted" text is not a problem, but then the result is just plain text without formatting). Alternatively, `dtaDcs` can be a named list with the entries `description`, `variables`, `references`, `license`. All entries except from `variables` contain character vectors (typically with the length = 1); `variables` shall be a named list with the variable name as name and a description what the variable contains as entry. `description` and `variables` must be given, `references` and `license` can be left blank (""; but the names must be present in the list). `license` can also a character vector containing one of the following keywords, producing a license text fitting the respective license: "CC0" (Creative Commons, CC0), "DT_CC4-BY-NC-ND" and "FC_CC4-BY-NC-ND" (Creative Commons, CC BY-NC-ND 4.0, for data from scientific studies "DT_" or generated / simulated data sets "FC_"; both need the license holder to be given as the second element of the character vector for the list entry `license`, e.g., `list(..., list = c("DT_CC4-BY-NC-ND", "John Doe"))`, "RP_GPL2", "RP_GPL3", "RP_AGPL3", and "RP_LGPL3" are for data sets that were originally part of R-packages under different versions of the GNU General Public License (respectively, GNU General Public License 2.x, GNU General Public License 3.0, GNU Affero General Public License 3.0, and GNU Lesser General Public License 3.0; these need the name of the R-package to be given as the second element of the character vector for the list entry `license`, e.g., `list(..., list = c("RP_GPL2", "jmv")`. Examples for both named lists with a description (`lstDsc`), as well as a character vector with all HTML tags that are implemented (`chrDsc`) can be found in the examples below.

- The ellipsis-parameter (. . .) can be used to submit arguments / parameters to the functions that are used for reading the data. By clicking on the respective function under "See also", you can get a more detailed overview over which parameters each of those functions take.

## Value

a data frame (only returned if `fleOut` is empty) where a description and a title are added to the input data

## See Also

`describe_omv` internally uses the following functions for reading and writing data files in different formats: `read_omv()` and `write_omv()` for jamovi-files, `utils::read.table()` for CSV / TSV files, `load()` for reading .RData-files, `readRDS()` for .rds-files, `haven::read_sav()` or `foreign::read.spss()` for SPSS-files, `haven::read_dta()` or `foreign::read.dta()` for Stata-files, `haven::read_sas()` for SAS-data-files, and `haven::read_xpt()` or `foreign::read.xport()` for SAS-transport-files.

## Examples

```
nmeOut <- tempfile(fileext = ".omv")

# the paste's underneath are only for readability (without them, the vignettes are misformatted)
# example for a data set without a license
dtaFrm <- jmvReadWrite::ToothGrowth[, c("len", "supp", "dose")]
lstDsc <- list(description = paste("The response is the length of odontoblasts (cells",
                                   "responsible for tooth growth) in 60 guinea pigs. Each",
                                   "animal received one of three dose levels of vitamin C",
                                  "(0.5, 1, and 2 mg / day) by one of two delivery methods,",
                                   "orange juice or ascorbic acid (a form of vitamin C and",
                                      "coded as VC)."),
               variables = list(len  = "Tooth length",
                                supp = "Supplement type (VC or OJ)",
                                dose = "Dose (in milligrams / day)"),
           references = paste("Crampton, E. W. (1947). The growth of the odontoblast of",
                              "the incisor teeth as a criterion of vitamin C intake of",
                             "the guinea pig. <em>The Journal of Nutrition, 33</em>(5),",
                                 "491-504. https://doi.org/10.1093/jn/33.5.491"),
               license = "")
jmvReadWrite::describe_omv(dtaInp = dtaFrm, fleOut = nmeOut, dtaTtl = "ToothGrowth",
                           dtaDsc = lstDsc)
# don't include the unlink, if you copy the code and want to look at the resulting output file
unlink(nmeOut)

# example for a data set which is part of the R-package `datasets` under the GPL2
dtaFrm <- datasets::penguins
lstDsc <- list(description = paste("Data on adult penguins covering three species found on",
                                   "three islands in the Palmer Archipelago, Antarctica,",
                                  "including their size (flipper length, body mass, bill",
                                    "dimensions), and sex."),
               variables = list(species     = "Species (Adelie, Chinstrap, and Gentoo)",
                                island      = "Island (Biscoe, Dream, and Torgersen)",
```

```
                                bill_len    = "Bill length (in mm)",
                                bill_dep    = "Bill depth (in mm)",
                                flipper_len = "Flipper length (in mm)",
                                body_mass   = "Body mass (in g)",
                                sex         = "Sex (female and male)",
                                year        = "Study year: 2007, 2008, or 2009"),
              references = paste("Gorman, K. B., Williams, T. D., & Fraser, W. R. (2014).",
                                 "Ecological sexual dimorphism and environmental variability",
                                "within a community of antarctic penguins (Genus pygoscelis).",
                                       "<em>PLoS ONE, 9</em>(3), e90081.",
                                       "https://doi.org/10.1371/journal.pone.0090081"),
                license = c("RP_GPL2", "datasets"))
jmvReadWrite::describe_omv(dtaInp = dtaFrm, fleOut = nmeOut, dtaTtl = "Penguins",
                           dtaDsc = lstDsc)
# don't include the unlink, if you copy the code and want to look at the resulting output file
unlink(nmeOut)

# the code underneath should cover all formatting options jamovi is able to use (paste0 is only
# for readability)
dtaFrm <- jmvReadWrite::ToothGrowth[, c("len", "supp", "dose")]
chrDsc <- paste0("<p><strong>Trial - all formattings:</strong><br/>",
                 "<strong>bold</strong><br/><strong><em>bold, italics</em></strong><br/>",
                   "<em>italics</em><br/><u>underlined</u><br/><s>strikethrough</s><br/>",
                   "C<sub>2</sub>H<sub>5</sub>OH<br/>R<sup>2</sup><br/>",
               "<span style=\"background-color:#e60000\">background colour: red</span><br/>",
                   "<span style=\"color:#e60000\">foreground color: red</span></p>",
                   "<p class=\"ql-align-center\">centered</p>",
                   "<p class=\"ql-align-right\">right</p>",
                  "<p class=\"ql-align-justify\">justify justify justify justify justify ",
                 "justify justify justify justify justify justify justify justify justify ",
                 "justify justify justify justify justify justify justify justify justify ",
                   "justify justify</p><p><br/></p>",
                   "<ol><li>numbered list</li><li>numbered list</li></ol><p><br/></p>",
                   "<ul><li>bullet point</li><li>bullet point</li></ul>",
                   "<p class=\"ql-indent-1\">indented once</p>",
                   "<p class=\"ql-indent-2\">indented twice</p>",
                   "<p class=\"ql-indent-1\">indented once</p>",
                   "<p>Formula: <span class=\"ql-formula\">e=mc^2</span></p>",
                   "<pre>Preformatted</pre><p>normal again</p><h2>Heading</h2>")
jmvReadWrite::describe_omv(dtaInp = dtaFrm, fleOut = nmeOut, dtaTtl = "ToothGrowth",
                           dtaDsc = chrDsc)
# don't include the unlink, if you copy the code and want to look at the resulting output file
unlink(nmeOut)
```

---

distances_omv *Calculates distances (returning a symmetric matrix) from a raw data matrix in .omv-files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org)*

---

## Description

Calculates distances (returning a symmetric matrix) from a raw data matrix in .omv-files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org)

## Usage

```
distances_omv(
  dtaInp = NULL,
  fleOut = "",
  varDst = NULL,
  clmDst = TRUE,
  stdDst = "none",
  nmeDst = "euclid",
  mtxSps = FALSE,
  mtxTrL = FALSE,
  mtxDgn = TRUE,
  usePkg = c("foreign", "haven"),
  selSet = "",
  ...
)
```

## Arguments

| | |
|---|---|
| dtaInp | Either a data frame or the name of a data file to be read (including the path, if required; "FILENAME.ext"; default: NULL); files can be of any supported file type, see Details below. |
| fleOut | Name of the data set / file to be written (including the path, if required; "FILE_OUT.omv"; default: ""); if empty, the resulting data frame is returned instead. |
| varDst | Variable (default: NULL) containing a character vector with the names of the variables for which distances are to be calculated. See Details for more information. |
| clmDst | Whether the distances shall be calculated between columns (TRUE) or rows (FALSE; default: TRUE). See Details for more information. |
| stdDst | Character string indicating whether the variables in varDst are to be standardized and how (default: "none"). See Details for more information. |
| nmeDst | Character string indicating which distance measure is to be calculated calculated (default: "euclidean"). See Details for more information. |
| mtxSps | Whether the symmetric matrix to be returned should be sparse (default: FALSE) |
| mtxTrL | Whether the symmetric matrix to be returned should only contain the lower triangular (default: FALSE) |
| mtxDgn | Whether the symmetric matrix to be returned should retain the values in the main diagonal (default: TRUE) |
| usePkg | Name of the package: "foreign" or "haven" that shall be used to read SPSS, Stata, and SAS files; "foreign" is the default (it is included in base R), but "haven" is newer and more comprehensive; you may have to install using `install.packages("haven", dep = TRUE)`. |

| | |
|---|---|
| selSet | Name of the object / data set that is to be selected from the workspace (only relevant when reading .RData-files which can contain several objects / data sets) |
| ... | Additional arguments passed on to methods; see Details below. |

## Details

- varDst must a character vector containing the variables to calculated distances over. If clmDst is set to TRUE, distances are calculated between all possible variable pairs and over subjects / rows in the original data frame. If clmDst is set to FALSE, distances are calculated between participants and over all variables given in varDst. If clmDst is set to TRUE, the symmetric matrix that is returned has the size V x V (V being the number of variables in varDst; if mtxSps is set to TRUE, the size is V - 1 x V - 1, see below); if clmDst is set to FALSE, the symmetric matrix that is returned has the size R x R (R being the number of rows in the original dataset; it is if mtxSps is set to TRUE, the size is R - 1 x R - 1, see below).
- stdDst can be one of the following calculations to standardize the selected variables before calculating the distances: none (do not standardize; default), z (z scores), sd (divide by the std. dev.), range (divide by the range), max (divide by the absolute maximum), mean (divide by the mean), rescale (subtract the mean and divide by the range).
- nmeDst can be one of the following distance measures. (1) For interval data: euclid (Euclidean), seuclid (squared Euclidean), block (city block / Manhattan), canberra (Canberra). chebychev (maximum distance / supremum norm / Chebychev), minkowski_p (Minkowski with power p; NB: needs p), power_p_r (Minkowski with power p, and the r-th root; NB: needs p and r), cosine (cosine between the two vectors), correlation (correlation between the two vectors). (2) For frequency count data: chisq (chi-square dissimilarity between two sets of frequencies), ph2 (chi-square dissimilarity normalized by the square root of the number of values used in the calculation). (3) For binary data, all measure have to optional parts p and np which indicate presence (p; defaults to 1 if not given) or absence (np; defaults to zero if not given). (a) matching coefficients: rr_p_np (Russell and Rao), sm_p_np (simple matching), jaccard_p_np / jaccards_p_np (Jaccard similarity; as in SPSS), jaccardd_p_np (Jaccard dissimiliarity; as in dist(..., "binary") in R), dice_p_np (Dice or Czekanowski or Sorenson similarity), ss1_p_np (Sokal and Sneath measure 1), rt_p_np (Rogers and Tanimoto), ss2_p_np (Sokal and Sneath measure 2), k1_p_np (Kulczynski measure 1), ss3_p_np (Sokal and Sneath measure 3). (b) conditional probabilities: k2_p_np (Kulczynski measure 2), ss4_p_np (Sokal and Sneath measure 4), hamann_p_np (Hamann). (c) predictability measures: lambda_p_np (Goodman and Kruskal Lambda), d_p_np (Anderberg's D), y_p_np (Yule's Y coefficient of colligation), q_p_np (Yule's Q). (d) other measures: ochiai_p_np (Ochiai), ss5_p_np (Sokal and Sneath measure 5), phi_p_np (fourfold point correlation), beuclid_p_np (binary Euclidean distance), bseuclid_p_np (binary squared Euclidean distance), size_p_np (size difference), pattern_p_np (pattern difference), bshape_p_np (binary Shape difference), disper_p_np (dispersion similarity), variance_p_np (variance dissimilarity), blwmn_p_np (binary Lance and Williams non-metric dissimilarity). (4) none (only carry out standardization, if stdDst is different from none).
- If mtxSps is set, a sparse matrix is returned. Those matrices are similar to the format one often finds for correlation matrices. The values are only retained in the lower triangular, the columns range from the first to the variable that is second to the last in varDst (or respectively, the columns contain the first to the second to the last row of the original dataset when clmDst is set to FALSE), and the rows contain the second to the last variable in varDst (or respectively, the rows contain the second to the last row of the original dataset when clmDst is set to FALSE).

- By default, a full symmetric matrix is returned (i.e., a matrix that has no NAs in any cell). This behaviour can be changed with setting `mtxTrL` and `mtxDgn`: If `mtxTrL` is set to `TRUE`, the values from the upper triangular matrix are removed / replaced with NAs; if `mtxDgn` is set to `FALSE`, the values from the main diagonal are removed / replaced with NAs.
- The ellipsis-parameter (`...`) can be used to submit arguments / parameters to the functions that are used for reading or transforming the data. By clicking on the respective function under "See also", you can get a more detailed overview over which parameters each of those functions take.

## Value

a data frame containing a symmetric matrix (only returned if `fleOut` is empty) containing the distances between the variables / columns (clmDst == TRUE) or rows (clmDst == FALSE)

## See Also

`distances_omv` internally uses the following function for calculating the distances for interval data `stats::dist()`. It furthermore uses the following functions for reading and writing data files in different formats: `read_omv()` and `write_omv()` for jamovi-files, `utils::read.table()` for CSV / TSV files, `load()` for reading .RData-files, `readRDS()` for .rds-files, `haven::read_sav()` or `foreign::read.spss()` for SPSS-files, `haven::read_dta()` or `foreign::read.dta()` for Stata-files, `haven::read_sas()` for SAS-data-files, and `haven::read_xpt()` or `foreign::read.xport()` for SAS-transport-files.

## Examples

```
# create matrices for the different types of distance measures: continuous
# (cntFrm), frequency counts (frqFrm) or binary (binFrm); all 20 R x 5 C
set.seed(1)
cntFrm <- stats::setNames(as.data.frame(matrix(rnorm(100, sd = 10),
            ncol = 5)), sprintf("C_%02d", seq(5)))
frqFrm <- stats::setNames(as.data.frame(matrix(sample(seq(10), 100,
            replace = TRUE), ncol = 5)), sprintf("F_%02d", seq(5)))
binFrm <- stats::setNames(as.data.frame(matrix(sample(c(TRUE, FALSE), 100,
            replace = TRUE), ncol = 5)), sprintf("B_%02d", seq(5)))
nmeOut <- tempfile(fileext = ".omv")

# calculates the distances between columns, nmeDst is not required: "euclid"
# is the default
jmvReadWrite::distances_omv(dtaInp = cntFrm, fleOut = nmeOut, varDst =
  names(cntFrm), nmeDst = "euclid")
dtaFrm <- jmvReadWrite::read_omv(nmeOut)
unlink(nmeOut)
# the resulting matrix (10 x 10) with the Euclidian distances
print(dtaFrm)

# calculates the (Euclidean) distances between rows (clmDst = FALSE)
jmvReadWrite::distances_omv(dtaInp = cntFrm, fleOut = nmeOut, varDst =
  names(cntFrm), clmDst = FALSE, nmeDst = "euclid")
dtaFrm <- jmvReadWrite::read_omv(nmeOut)
unlink(nmeOut)
```

```
# the resulting matrix (20 x 20) with the Euclidian distances
print(dtaFrm)

# calculates the (Euclidean) distances between columns; the original data
# are z-standardized before calculating the distances (stdDst = "z")
jmvReadWrite::distances_omv(dtaInp = cntFrm, fleOut = nmeOut, varDst =
  names(cntFrm), stdDst = "z", nmeDst = "euclid")
dtaFrm <- jmvReadWrite::read_omv(nmeOut)
unlink(nmeOut)
# the resulting matrix (10 x 10) with the Euclidian distances using the
# z-standardized data
print(dtaFrm)

# calculates the correlations between columns
jmvReadWrite::distances_omv(dtaInp = cntFrm, fleOut = nmeOut, varDst =
  names(cntFrm), nmeDst = "correlation")
dtaFrm <- jmvReadWrite::read_omv(nmeOut)
unlink(nmeOut)
# the resulting matrix (10 x 10) with the correlations
print(dtaFrm)

# calculates the chi-square dissimilarity (nmeDst = "chisq") between columns
jmvReadWrite::distances_omv(dtaInp = frqFrm, fleOut = nmeOut, varDst =
  names(frqFrm), nmeDst = "chisq")
dtaFrm <- jmvReadWrite::read_omv(nmeOut)
unlink(nmeOut)
# the resulting matrix (10 x 10) with the chi-square dissimilarities
print(dtaFrm)

# calculates the Jaccard similarity (nmeDst = "jaccard") between columns
jmvReadWrite::distances_omv(dtaInp = binFrm, fleOut = nmeOut, varDst =
  names(binFrm), nmeDst = "jaccard")
dtaFrm <- jmvReadWrite::read_omv(nmeOut)
unlink(nmeOut)
# the resulting matrix (10 x 10) with the Jaccard similarities
print(dtaFrm)
```

---

`label_vars_omv` *Label columns / variables in .omv-files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org)*

---

## Description

Label columns / variables in .omv-files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org)

## Usage

```
label_vars_omv(
  dtaInp = NULL,
  fleOut = "",
  varLbl = NULL,
  psvAnl = FALSE,
  usePkg = c("foreign", "haven"),
  selSet = "",
  ...
)
```

## Arguments

| | |
|---|---|
| dtaInp | Either a data frame or the name of a data file to be read (including the path, if required; "FILENAME.ext"; default: NULL); files can be of any supported file type, see Details below. |
| fleOut | Name of the data set / file to be written (including the path, if required; "FILE_OUT.omv"; default: ""); if empty, the resulting data frame is returned instead. |
| varLbl | Variable (default: NULL) containing either a character (a file name; the file must contain two columns one with variable names, the other with the labels), a data frame (one column the variable names, the other the labels), or a character vector (with the same length as the data set, containing the variable labels). See Details for more information. |
| psvAnl | Whether analyses that are contained in the input file shall be transferred to the output file (TRUE / FALSE; default: FALSE) |
| usePkg | Name of the package: "foreign" or "haven" that shall be used to read SPSS, Stata, and SAS files; "foreign" is the default (it is included in base R), but "haven" is newer and more comprehensive; you may have to install using `install.packages("haven", dep = TRUE)`. |
| selSet | Name of the object / data set that is to be selected from the workspace (only relevant when reading .RData-files which can contain several objects / data sets) |
| ... | Additional arguments passed on to methods; see Details below |

## Details

- `varLbl` can be either (1) a character with a file name to read (the file must contain to columns, one with the variable names, the other with the variable labels); (2) a data frame with two columns (one with the variable names, the other with the variable labels), or (3) a character vector containing the variable labels (with a length equal to the number of variables in the input data set).
- The ellipsis-parameter (`...`) can be used to submit arguments / parameters to the functions that are used for reading the data. By clicking on the respective function under "See also", you can get a more detailed overview over which parameters each of those functions take.

## Value

a data frame (only returned if `fleOut` is empty) where the order of variables / columns of the input data set is re-arranged

## See Also

`label_vars_omv` internally uses the following functions for reading and writing data files in different formats: `read_omv()` and `write_omv()` for jamovi-files, `utils::read.table()` for CSV / TSV files, `load()` for reading .RData-files, `readRDS()` for .rds-files, `haven::read_sav()` or `foreign::read.spss()` for SPSS-files, `haven::read_dta()` or `foreign::read.dta()` for Stata-files, `haven::read_sas()` for SAS-data-files, and `haven::read_xpt()` or `foreign::read.xport()` for SAS-transport-files.

## Examples

```
# use one of the data files included in the package, but only the first 28 columns
# (the latter columns contain data for testing calculations, etc.)
nmeInp <- system.file("extdata", "bfi_sample.omv", package = "jmvReadWrite")
dtaInp <- jmvReadWrite::read_omv(nmeInp)[1:28]
nmeOut <- tempfile(fileext = ".omv")
# in the original file, the variable labels - attr(*, "jmv-desc") - are empty
lapply(dtaInp, attr, "jmv-desc")
# the definition of the variable labels can be read from a file with two columns,
# the first containing the variable name, the second the variable labels
# you can easily create such a file in Excel and save it as CSV
# if your CSV contains column names (e.g., varNme and varLbl) in the first row are they ignored
lblFle <- system.file("extdata", "label_example.csv", package = "jmvReadWrite")
lblDtF <- utils::read.csv(lblFle, header = FALSE)
str(lblDtF)

# there are three options to give the varLbl parameter:
# (1) as file name, ...
jmvReadWrite::label_vars_omv(dtaInp = dtaInp, fleOut = nmeOut, varLbl = lblFle)
lapply(jmvReadWrite::read_omv(nmeOut), attr, "jmv-desc")
unlink(nmeOut)

# (2) as data frame (using lblDtF from above), or ...
jmvReadWrite::label_vars_omv(dtaInp = dtaInp, fleOut = nmeOut, varLbl = lblDtF)
lapply(jmvReadWrite::read_omv(nmeOut), attr, "jmv-desc")
unlink(nmeOut)

# (3) as character vector (with the same length as there are columns in the input data set)
lblChr <- lblDtF[[2]]
head(lblChr)
jmvReadWrite::label_vars_omv(dtaInp = dtaInp, fleOut = nmeOut, varLbl = lblChr)
lapply(jmvReadWrite::read_omv(nmeOut), attr, "jmv-desc")
unlink(nmeOut)
```

long2wide_omv *Converts .omv-files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org) from long to wide format*

## Description

Converts .omv-files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org) from long to wide format

## Usage

```
long2wide_omv(
  dtaInp = NULL,
  fleOut = "",
  varTgt = NULL,
  varExc = NULL,
  varID = "ID",
  varTme = NULL,
  varSep = "_",
  varOrd = c("times", "vars"),
  varAgg = c("mean", "first"),
  varSrt = NULL,
  usePkg = c("foreign", "haven"),
  selSet = "",
  ...
)
```

## Arguments

| | |
|---|---|
| dtaInp | Either a data frame or the name of a data file to be read (including the path, if required; "FILENAME.ext"; default: NULL); files can be of any supported file type, see Details below. |
| fleOut | Name of the data set / file to be written (including the path, if required; "FILE_OUT.omv"; default: ""); if empty, the resulting data frame is returned instead. |
| varTgt | Names of one or more variables to be transformed / reshaped (other variables are excluded, if empty (NULL) all variables except varTme, varID and varExc are included; default: NULL) |
| varExc | Name of the variable(s) should be excluded from the transformation, typically this will be between-subject-variable(s) (default: NULL) |
| varID | Names of one or more variables that identify the same group / individual (default: NULL) |
| varTme | Name of the variable(s) that differentiates multiple records from the same group / individual (default: NULL) |
| varSep | Separator character when concatenating the fixed and time-varying part of the variable name ("VAR1_1", "VAR1_2"; default: "_") |

| | |
|---|---|
| varOrd | How variables / columns are organized: for "times" (default) the steps of the time varying variable are adjacent, for "vars" the steps of the original columns in the long dataset. |
| varAgg | How multiple occurrences of particular combinations of time varying variables are aggregated: either "mean" (calculate the mean over occurrences), or "first" (take the first occurrence). |
| varSrt | Variable(s) that are used to sort the data frame (see Details; if empty, the order returned from reshape is kept; default: NULL) |
| usePkg | Name of the package: "foreign" or "haven" that shall be used to read SPSS, Stata, and SAS files; "foreign" is the default (it is included in base R), but "haven" is newer and more comprehensive; you may have to install using `install.packages("haven", dep = TRUE)`. |
| selSet | Name of the object / data set that is to be selected from the workspace (only relevant when reading .RData-files which can contain several objects / data sets) |
| ... | Additional arguments passed on to methods; see Details below |

## Details

- If `varTgt` is empty, it is tried to generate it using all variables in the data frame except those defined by `varID`, `varTme` and `varExc`. The variable(s) in `varID` need to be unique identifiers (in the original dataset), those in `varExc` don't have this requirement. It is generally recommended that the variable names in `varExc` and `varID` should not contain the variable separator (defined in `varSep`; default: "_").
- `varSrt` can be either a character or a character vector (with one or more variables respectively). The sorting order for a particular variable can be inverted with preceding the variable name with "-". Please note that this doesn't make sense and hence throws a warning for certain variable types (e.g., factors).
- The transformation from long to wide uses `reshape`. `varTgt` matches (~) `v.names` in `reshape`, `varID ~ idvar`, `varTme ~ timevar`, and `varSep ~ sep`. The help for `reshape` is very explanatory, click on the link under "See also" to access it, particularly what is explained under "Details".
- The ellipsis-parameter (`...`) can be used to submit arguments / parameters to the functions that are used for reading or transforming the data. By clicking on the respective function under "See also", you can get a more detailed overview over which parameters each of those functions take.

## Value

a data frame (only returned if `fleOut` is empty) where the input data set is converted from long to wide format

## See Also

`long2wide_omv` internally uses the following functions: The transformation from long to wide uses `stats::reshape()`. For reading and writing data files in different formats: `read_omv()` and `write_omv()` for jamovi-files, `utils::read.table()` for CSV / TSV files, `load()` for reading .RData-files, `readRDS()` for .rds-files, `haven::read_sav()` or `foreign::read.spss()` for

SPSS-files, haven::read_dta() or foreign::read.dta() for Stata-files, haven::read_sas() for SAS-data-files, and haven::read_xpt() or foreign::read.xport() for SAS-transport-files.

## Examples

```
# generate a test dataframe with 100 (imaginary) participants / units of
#  observation (ID), 8 measurement (measure) of one variable (X)
dtaInp <- data.frame(ID = rep(as.character(seq(1, 100)), each = 8),
                     measure = rep(seq(1, 8), times = 100),
                     X = runif(800, -10, 10))
cat(str(dtaInp))
# the output should look like this
# 'data.frame': 800 obs. of  3 variables:
#  $ ID     : chr  "1" "1" "1" "1" ...
#  $ measure: int  1 2 3 4 5 6 7 8 1 2 ...
#  $ X      : num  ...
# this data set is stored as (temporary) RDS-file and later processed by long2wide
nmeInp <- tempfile(fileext = ".rds")
nmeOut <- tempfile(fileext = ".omv")
saveRDS(dtaInp, nmeInp)
jmvReadWrite::long2wide_omv(dtaInp = nmeInp, fleOut = nmeOut, varTgt = "X", varID = "ID",
  varTme = "measure")
# it is required to give at least the arguments dtaInp, varID and varTme
# check whether the file was created and its size
cat(list.files(dirname(nmeOut), basename(nmeOut)))
# -> "file[...].omv" ([...] contains a random combination of numbers / characters
cat(file.info(nmeOut)$size)
# -> 6851 (approximate size; size may differ in every run [in dependence of
#          how well the generated random data can be compressed])
cat(str(jmvReadWrite::read_omv(nmeOut, sveAtt = FALSE)))
# the data set is now transformed into wide (and each the measurements is now
# indicated as a suffix to X; X_1, X_2, ...)
# 'data.frame': 100 obs. of  9 variables:
#  $ ID : chr  "1" "2" "3" "4" "5" "6" "7" "8" "9" "10" ...
#   ..- attr(*, "jmv-id")= logi TRUE
#   ..- attr(*, "missingValues")= list()
#  $ X_1: num  ...
#   ..- attr(*, "missingValues")= list()
#  $ X_2: num  ...
#   ..- attr(*, "missingValues")= list()
#  $ X_3: num  ...
#   ..- attr(*, "missingValues")= list()
#  $ X_4: num  ...
#   ..- attr(*, "missingValues")= list()
#  $ X_5: num  ...
#   ..- attr(*, "missingValues")= list()
#  $ X_6: num  ...
#   ..- attr(*, "missingValues")= list()
#  $ X_7: num  ...
#   ..- attr(*, "missingValues")= list()
#  $ X_8: num  ...
#   ..- attr(*, "missingValues")= list()
```

```
unlink(nmeInp)
unlink(nmeOut)
```

---

`merge_cols_omv` *Merges two or more data files by adding the content of other input files as columns to the first input file and outputs them as files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org)*

---

## Description

Merges two or more data files by adding the content of other input files as columns to the first input file and outputs them as files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org)

## Usage

```
merge_cols_omv(
  dtaInp = NULL,
  fleOut = "",
  typMrg = c("outer", "inner", "left", "right"),
  varBy = list(),
  varSrt = NULL,
  psvAnl = FALSE,
  usePkg = c("foreign", "haven"),
  selSet = "",
  ...
)
```

## Arguments

| | |
|---|---|
| `dtaInp` | Either a data frame (with the attribute "fleInp" containing the files to merge) or vector with the names of the input files (including the path, if required; "FILE-NAME.ext"; default: NULL); files can be of any supported file type, see Details below. |
| `fleOut` | Name of the data set / file to be written (including the path, if required; "FILE_OUT.omv"; default: ""); if empty, the resulting data frame is returned instead. |
| `typMrg` | Type of merging operation: "outer" (default), "inner", "left" or "right"; see Details below. |
| `varBy` | Name of the variable by which the data sets are matched, can either be a string, a character or a list (see Details below; default: list()) |
| `varSrt` | Variable(s) that are used to sort the data frame (see Details; if empty, the order after merging is kept; default: NULL) |
| `psvAnl` | Whether analyses that are contained in the input file shall be transferred to the output file (TRUE / FALSE; default: FALSE) |

| | |
|---|---|
| usePkg | Name of the package: "foreign" or "haven" that shall be used to read SPSS, Stata, and SAS files; "foreign" is the default (it is included in base R), but "haven" is newer and more comprehensive; you may have to install using `install.packages("haven", dep = TRUE)`. |
| selSet | Name of the object / data set that is to be selected from the workspace (only relevant when reading .RData-files which can contain several objects / data sets) |
| ... | Additional arguments passed on to methods; see Details below |

## Details

- Using data frames with the input parameter `dtaInp` is primarily thought to be used when calling `merge_cols_omv` from the jamovi-modules `jTransform` and `Rj`. For the use in R, it is strongly recommended to use a character vector with the file names instead.
- There are four different types of merging operations (defined via `typMrg`): "outer" keeps all cases (but columns in the resulting data set may contain empty cells / missing values if same input data sets did not have a row containing the matching variable (defined in `varBy`). "inner" keeps only those cases where all datasets contain the same value in the matching variable, for "left" all cases from the first data set in `dtaInp` are kept (whereas cases that are only contained in the second or any later input data set are dropped), for "right" all cases from the second (or any higher) data set in `dtaInp` are kept. The behaviour of "left" and "right" may be somewhat difficult to predict in case of merging several data sets, therefore "outer" might be a safer choice if several data sets are merged.
- The variable that is used for matching (`varBy`) can either be a string (if all datasets contain a matching variable with the same name), a character vector (containing more than one matching variables that are contained in / the same for all data sets) or a list with the same length as dtaInp. In such list, each cell can again contain either a string (one matching variable for each data set in dtaInp) or a character vector (several matching variables for each data set in dtaInp; NB: all character vectors in the cells of the list must have the same length as it is necessary to always use the same number of matching variables when merging).
- `varSrt` can be either a character or a character vector (with one or more variables respectively). The sorting order for a particular variable can be inverted with preceding the variable name with "-". Please note that this doesn't make sense and hence throws a warning for certain variable types (e.g., factors).
- Adding columns uses `merge`. `typMrg` is implemented by setting `TRUE` or `FALSE` to `all.x` and `all.y` in `merge`, `varBy` matches `by.x` and `by.y`. The help for `merge` can be accessed by clicking on the link under "See also".
- The ellipsis-parameter (`...`) can be used to submit arguments / parameters to the functions that are used for transforming or reading the data. By clicking on the respective function under "See also", you can get a more detailed overview over which parameters each of those functions take.

## Value

a data frame (only returned if `fleOut` is empty) where the columns of all input data sets (given in the `dtaInp`-argument) are concatenated

## See Also

merge_cols_omv internally uses the following functions: Adding columns uses merge(). For reading and writing data files in different formats: read_omv() and write_omv() for jamovi-files, utils::read.table() for CSV / TSV files, load() for reading .RData-files, readRDS() for .rds-files, haven::read_sav() or foreign::read.spss() for SPSS-files, haven::read_dta() or foreign::read.dta() for Stata-files, haven::read_sas() for SAS-data-files, and haven::read_xpt() or foreign::read.xport() for SAS-transport-files.

## Examples

```
dtaInp <- jmvReadWrite::bfi_sample2
nmeInp <- paste0(tempfile(), "_", 1:3, ".rds")
nmeOut <- tempfile(fileext = ".omv")
for (i in seq_along(nmeInp)) {
   saveRDS(stats::setNames(dtaInp, c("ID", paste0(names(dtaInp)[-1], "_", i))), nmeInp[i])
}
# save dtaInp three times (i.e., the length of nmeInp), adding "_" + 1 ... 3 as index
# to the data variables (A1 ... O5, gender, age → A1_1, ...)
jmvReadWrite::merge_cols_omv(dtaInp = nmeInp, fleOut = nmeOut, varBy = "ID")
cat(file.info(nmeOut)$size)
# -> 17731 (size may differ on different OSes)
dtaOut <- jmvReadWrite::read_omv(nmeOut, sveAtt = FALSE)
# read the data set where the three original datasets were added as columns and show
# the variable names
cat(names(dtaOut))
cat(names(dtaInp))
# compared to the input data set, we have the same names (expect for "ID" which was
# used for matching and that each variable had added an indicator from which data
# set they came)
cat(dim(dtaInp), dim(dtaOut))
# the first dimension of the data sets (rows) stayed the same (250), whereas the
# second dimension is now approx. three times as large (28 -> 82):
# 28 - 1 (for "ID") = 27 * 3 + 1 (for "ID") = 82
cat(colMeans(dtaInp[2:11]))
cat(colMeans(dtaOut[2:11]))
# it's therefore not much surprise that the values of the column means for the first
# 10 variables of dtaInp and dtaOut are the same too

unlink(nmeInp)
unlink(nmeOut)
```

---

merge_rows_omv — *Merges two .omv-files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org) by adding the content of the second, etc. file(s) as rows to the first file*

---

## Description

Merges two .omv-files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org) by adding the content of the second, etc. file(s) as rows to the first file

## Usage

```
merge_rows_omv(
  dtaInp = NULL,
  fleOut = "",
  typMrg = c("all", "common"),
  colInd = FALSE,
  rstRwN = TRUE,
  rmvDpl = FALSE,
  varSrt = NULL,
  usePkg = c("foreign", "haven"),
  selSet = "",
  ...
)
```

## Arguments

| | |
|---|---|
| dtaInp | Either a data frame (with the attribute "fleInp" containing the files to merge) or vector with the names of the input files (including the path, if required; "FILE-NAME.ext"; default: NULL); files can be of any supported file type, see Details below. |
| fleOut | Name of the data set / file to be written (including the path, if required; "FILE_OUT.omv"; default: ""); if empty, the resulting data frame is returned instead. |
| typMrg | Type of merging operation: "all" (default) or "common"; see also Details. |
| colInd | Add a column with an indicator (the basename of the file minus the extension) marking from which input data set the respective rows are coming (default: FALSE) |
| rstRwN | Reset row names (i.e., do not keep the row names of the original input data sets but number them consecutively - one to the row number of all input data sets added up; default: TRUE) |
| rmvDpl | Remove duplicated rows (i.e., rows with the same content as a previous row in all columns; default: FALSE) |
| varSrt | Variable(s) that are used to sort the data frame (see Details; if empty, the order after merging is kept; default: NULL) |
| usePkg | Name of the package: "foreign" or "haven" that shall be used to read SPSS, Stata, and SAS files; "foreign" is the default (it is included in base R), but "haven" is newer and more comprehensive; you may have to install using `install.packages("haven", dep = TRUE)`. |
| selSet | Name of the object / data set that is to be selected from the workspace (only relevant when reading .RData-files which can contain several objects / data sets) |
| ... | Additional arguments passed on to methods; see Details below |

## Details

- Using data frames with the input parameter `dtaInp` is primarily thought to be used when calling `merge_cols_omv` from the jamovi-modules `jTransform` and `Rj`. For the use in R, it is strongly recommended to use a character vector with the file names instead.
- There are four different types of merging operations (defined via `typMrg`): "all" keeps all existing variables / columns that are contained in any of the input data sets and fills them up with NA where the variable / column doesn't exist in an input data set. "common" only keeps the variables / columns that are common to all input data sets (i.e., that are contained in all data sets).
- `varSrt` can be either a character or a character vector (with one or more variables respectively). The sorting order for a particular variable can be inverted with preceding the variable name with "-". Please note that this doesn't make sense and hence throws a warning for certain variable types (e.g., factors).
- Adding columns uses `rbind` (with some further operation, adding missing columns (filled with NAs), if `typMrg` is "all").
- The ellipsis-parameter (`...`) can be used to submit arguments / parameters to the functions that are used for reading or transforming the data. By clicking on the respective function under "See also", you can get a more detailed overview over which parameters each of those functions take.

## Value

a data frame (only returned if `fleOut` is empty) where the rows of all input data sets (given in the `dtaInp`-argument) are concatenated

## See Also

`merge_rows_omv` internally uses the following functions: Adding columns uses `rbind()`. For reading and writing data files in different formats: `read_omv()` and `write_omv()` for jamovi-files, `utils::read.table()` for CSV / TSV files, `load()` for reading .RData-files, `readRDS()` for .rds-files, `haven::read_sav()` or `foreign::read.spss()` for SPSS-files, `haven::read_dta()` or `foreign::read.dta()` for Stata-files, `haven::read_sas()` for SAS-data-files, and `haven::read_xpt()` or `foreign::read.xport()` for SAS-transport-files.

## Examples

```
dtaInp <- jmvReadWrite::bfi_sample2
nmeInp <- paste0(tempfile(), "_", 1:3, ".rds")
nmeOut <- tempfile(fileext = ".omv")
for (i in seq_along(nmeInp)) saveRDS(dtaInp[-i - 1], nmeInp[i])
# save dtaInp three times (i.e., the length of nmeInp), removing one data columns in
# each data set (for demonstration purposes, A1 in the first, A2 in the second, ...)
jmvReadWrite::merge_rows_omv(dtaInp = nmeInp, fleOut = nmeOut, colInd = TRUE)
cat(file.info(nmeOut)$size)
# -> 10767 (size may differ on different OSes)
dtaOut <- jmvReadWrite::read_omv(nmeOut, sveAtt = FALSE)
unlink(nmeOut)
# read the data set where the three original datasets were added as rows and show
```

```
# the variable names
cat(names(dtaInp))
cat(names(dtaOut))
# compared to the input data set, we have the same variable names; fleInd (switched
# on by colInd = TRUE and showing from which data set the rows are coming from) is
# new and A1 is moved to the end of the list (the "original" order of variables may
# not always be preserved and columns missing from at least one of the input data
# sets may be added at the end)
cat(dim(dtaInp), dim(dtaOut))
# the first dimension of the data sets (rows) is now three times of that of the input
# data set (250 -> 750), the second dimension (columns / variables) is increased by 1
# (for "fleInd")

jmvReadWrite::merge_rows_omv(dtaInp = nmeInp, fleOut = nmeOut, typMrg = "common")
# the argument typMrg = "common" removes the columns that are not present in all of
# the input data sets (i.e., A1, A2, A3)
dtaOut <- jmvReadWrite::read_omv(nmeOut, sveAtt = FALSE)
unlink(nmeOut)
# read the data set where the three original datasets were added as rows and show
# the variable names
cat(names(dtaInp))
cat(names(dtaOut))
# compared to the input data set, the variables that were missing in at least one
# data set (i.e., "A1", "A2" and "A3") are removed
cat(dim(dtaInp), dim(dtaOut))
# the first dimension of the data sets (rows) is now three times of that of the
# input data set (250 -> 750), the second dimension (columns / variables) is
# reduced by 3 (i.e., "A1", "A2", "A3")

unlink(nmeInp)
```

---

`read_omv` *Read files created of the statistical spreadsheet 'jamovi' (https://www.jamovi.org)*

---

## Description

Read files created of the statistical spreadsheet 'jamovi' (https://www.jamovi.org)

## Usage

```
read_omv(
  fleInp = "",
  useFlt = FALSE,
  rmMsVl = FALSE,
  sveAtt = TRUE,
  getSyn = FALSE,
  getHTM = FALSE
)
```

## Arguments

| | |
|---|---|
| fleInp | Name (including the path, if required) of the 'jamovi'-file to be read ("FILE-NAME.omv"; default: "") |
| useFlt | Apply filters (remove the lines where the filter is set to 0; default: FALSE)? |
| rmMsVl | Remove values defined as missing values (replace them with NA; default: FALSE)? |
| sveAtt | Store attributes that are not required in the data set (if you want to write the same data set using write_omv; default: FALSE)? |
| getSyn | Extract syntax from the analyses in the 'jamovi'-file and store it in the attribute "syntax" (default: FALSE)? |
| getHTM | Store index.html in the attribute "HTML" (default: FALSE)? |

## Value

data frame (can be directly used with functions included in the R-package jmv and syntax from 'jamovi'; also compatible with the format of the R-package foreign)

## Examples

```
nmeInp <- system.file("extdata", "ToothGrowth.omv", package = "jmvReadWrite")
data <- jmvReadWrite::read_omv(fleInp = nmeInp)
str(data)
# shows the data frame including all jamovi attributes:
# 'data.frame': 60 obs. of  16 variables:
#  $ Filter 1          : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
#   ..- attr(*, "name")= chr "Filter 1"
#   ..- attr(*, "id")= int 18
#   ..- attr(*, "columnType")= chr "Filter"
#   ..- attr(*, "dataType")= chr "Integer"
#   ..- attr(*, "measureType")= chr "Nominal"
#   ..- attr(*, "formula")= chr "logLen < 1.5"
#   ..- attr(*, "formulaMessage")= chr ""
#   ..- attr(*, "parentId")= int 0
#   ..- attr(*, "width")= int 78
#   ..- attr(*, "type")= chr "integer"
#   ..- attr(*, "importName")= chr ""
#   ..- attr(*, "description")= chr ""
#   ..- attr(*, "transform")= int 0
#   ..- attr(*, "edits")= list()
#   ..- attr(*, "missingValues")= list()
#   ..- attr(*, "filterNo")= int 0
#   ..- attr(*, "active")= logi FALSE
# ... (continued)

# getSyn requires jmvcore and RProtoBuf, and is thus not run here
## Not run:
data <- jmvReadWrite::read_omv(fleInp = nmeInp, getSyn = TRUE)

## End(Not run)
# if the syntax couldn't be extracted, an empty list - length = 0 - is returned,
# otherwise, the commands are shown and the first analysis is run, with the output
```

```
# from the second analysis being assigned to the variable result
if (length(attr(data, "syntax")) >= 1) {
    print(attr(data, "syntax"))
    if (nzchar(system.file(package = "jmv"))) {
       # the print-function is only used to force devtools::run_examples() to show output
        eval(parse(text = paste0("result = ", attr(data, "syntax")[1])))
        # without assigning the output to a variable, the command would be:
        # eval(parse(text = attr(data, "syntax")[1]))
        print(names(result))
        print(result$main)
        # -> "main"      "assump"    "contrasts" "postHoc"   "emm"       "residsOV"
        # (the names of the six output tables)
    }
}
```

---

replace_omv *Search values in .omv-files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org)*

---

## Description

Search values in .omv-files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org)

## Usage

```
replace_omv(
  dtaInp = NULL,
  fleOut = "",
  rplLst = list(),
  whlTrm = TRUE,
  varInc = NULL,
  varExc = NULL,
  incNum = TRUE,
  incOrd = TRUE,
  incNom = TRUE,
  incID = TRUE,
  incCmp = TRUE,
  incRcd = TRUE,
  psvAnl = FALSE,
  usePkg = c("foreign", "haven"),
  selSet = "",
  ...
)
```

## Arguments

| | |
|---|---|
| dtaInp | Either a data frame or the name of a data file to be read (including the path, if required; "FILENAME.ext"; default: NULL); files can be of any supported file type, see Details below. |
| fleOut | Name of the data set / file to be written (including the path, if required; "FILE_OUT.omv"; default: ""); if empty, the resulting data frame is returned instead. |
| rplLst | A list where each entry is a vector (with length 2) containing the original value and the to-replace-value (default: list()) |
| whlTrm | Whether the search term (first entry in the vectors) must be found exactly (TRUE) or whether a partial match is sufficient (FALSE; default: TRUE) |
| varInc | Names of variables (character vector) to be included in the replacement (default: NULL) |
| varExc | Names of variables (character vector) to be excluded from the replacement (default: NULL) |
| incNum | Whether to include continuous variables in the replacement (default: TRUE) |
| incOrd | Whether to include ordinal variables in the replacement (default: TRUE) |
| incNom | Whether to include nominal variables in the replacement (default: TRUE) |
| incID | Whether to include ID variables in the replacement (default: TRUE) |
| incCmp | Whether to include Computed variables in the replacement (default: TRUE) |
| incRcd | Whether to include Recoded variables in the replacement (default: TRUE) |
| psvAnl | Whether analyses that are contained in the input file shall be transferred to the output file (TRUE / FALSE; default: FALSE) |
| usePkg | Name of the package: "foreign" or "haven" that shall be used to read SPSS, Stata, and SAS files; "foreign" is the default (it is included in base R), but "haven" is newer and more comprehensive; you may have to install using `install.packages("haven", dep = TRUE)`. |
| selSet | Name of the object / data set that is to be selected from the workspace (only relevant when reading .RData-files which can contain several objects / data sets) |
| ... | Additional arguments passed on to methods; see Details below |

## Details

- `rplLst` is a list. Each list entry contains a vector (with length 2), where the first entry is the original value, and the second entry is the value the original value is to be replaced with.
- `whlTrm` indicates whether partial matches of the original value(s) shall replaced (e.g., for original: 24 and replacement: 34, 241 will be changed into 341).
- `varInc` and `varExc` determine which variables are included or excluded from the replacement. If both are given, a warning is issued and `varInc` takes precedence. `varInc` makes that only in these variables, the replacement requested by `rplLst` is carried out, if `varExc` is given, for all variables of the input data set, except those defined in `varExc`, the replacement is carried out.
- The ellipsis-parameter (`...`) can be used to submit arguments / parameters to the functions that are used for reading the data. By clicking on the respective function under "See also", you can get a more detailed overview over which parameters each of those functions take.

## Value

a data frame (only returned if `fleOut` is empty) with the replaced values

## See Also

`replace_omv` internally uses the following functions for reading and writing data files in different formats: `read_omv()` and `write_omv()` for jamovi-files, `utils::read.table()` for CSV / TSV files, `load()` for reading .RData-files, `readRDS()` for .rds-files, `haven::read_sav()` or `foreign::read.spss()` for SPSS-files, `haven::read_dta()` or `foreign::read.dta()` for Stata-files, `haven::read_sas()` for SAS-data-files, and `haven::read_xpt()` or `foreign::read.xport()` for SAS-transport-files.

## Examples

```
## Not run:
bfi_sample <- jmvReadWrite::bfi_sample
# the gender in the original data file is plural...
table(bfi_sample$gender)
# and shall be converted to singular
rplDF <- jmvReadWrite::replace_omv(dtaInp = bfi_sample,
           rplLst = list(c("Females", "Female"), c("Males", "Male")))
table(rplDF$gender)
# with giving an output file name, the data set is written
nmeOut <- tempfile(fileext = ".omv")
jmvReadWrite::replace_omv(bfi_sample, fleOut = nmeOut,
  rplLst = list(c("Females", "Female"), c("Males", "Male")))
file.exists(nmeOut)
rplDF <- jmvReadWrite::read_omv(nmeOut)
table(rplDF$gender)
unlink(nmeOut)
# it is sensible to check / search for the original values before running replace_omv
jmvReadWrite::search_omv(bfi_sample, 24, whlTrm = TRUE)
rplDF <- jmvReadWrite::replace_omv(bfi_sample, rplLst = list(c(24, NA)))
table(rplDF$age)

## End(Not run)
```

---

`search_omv` *Search values in .omv-files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org)*

---

## Description

Search values in .omv-files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org)

## Usage

```
search_omv(
  dtaInp = NULL,
  srcTrm = NULL,
  whlTrm = FALSE,
  ignCse = FALSE,
  incNum = TRUE,
  incOrd = TRUE,
  incNom = TRUE,
  incID = TRUE,
  incCmp = TRUE,
  incRcd = TRUE,
  usePkg = c("foreign", "haven"),
  selSet = "",
  ...
)
```

## Arguments

| | |
|---|---|
| dtaInp | Either a data frame or the name of a data file to be read (including the path, if required; "FILENAME.ext"; default: NULL); files can be of any supported file type, see Details below. |
| srcTrm | (Character or numeric) Vector (with length = 1) with a search term to be found in the data frame (default: NULL) |
| whlTrm | Whether the exact search term shall be found (TRUE) or whether a partial match is sufficient (FALSE; default: FALSE) |
| ignCse | Whether to ignore the case of the search term (default: FALSE) |
| incNum | Whether to include continuous variables in the search (default: TRUE) |
| incOrd | Whether to include ordinal variables in the search (default: TRUE) |
| incNom | Whether to include nominal variables in the search (default: TRUE) |
| incID | Whether to include ID variables in the search (default: TRUE) |
| incCmp | Whether to include Computed variables in the search (default: TRUE) |
| incRcd | Whether to include Recoded variables in the search (default: TRUE) |
| usePkg | Name of the package: "foreign" or "haven" that shall be used to read SPSS, Stata, and SAS files; "foreign" is the default (it is included in base R), but "haven" is newer and more comprehensive; you may have to install using `install.packages("haven", dep = TRUE)`. |
| selSet | Name of the object / data set that is to be selected from the workspace (only relevant when reading .RData-files which can contain several objects / data sets) |
| ... | Additional arguments passed on to methods; see Details below |

## Details

- The ellipsis-parameter (`...`) can be used to submit arguments / parameters to the functions that are used for reading the data. By clicking on the respective function under "See also", you can get a more detailed overview over which parameters each of those functions take.

## Value

a named list with the places where the search term was found: names in the list are the variables / columns, the entries the respective row names within that variable / column (row names are used for being tolerant to filtered-out cases in jamovi, if a filter is used, row numbers would be incorrect)

## See Also

search_omv internally uses the following functions for reading and writing data files in different formats: read_omv() and write_omv() for jamovi-files, utils::read.table() for CSV / TSV files, load() for reading .RData-files, readRDS() for .rds-files, haven::read_sav() or foreign::read.spss() for SPSS-files, haven::read_dta() or foreign::read.dta() for Stata-files, haven::read_sas() for SAS-data-files, and haven::read_xpt() or foreign::read.xport() for SAS-transport-files.

## Examples

```
# the exact value 24 appears 13 times in age
bfi_sample <- jmvReadWrite::bfi_sample
jmvReadWrite::search_omv(bfi_sample, 24, whlTrm = TRUE)
# taking the fifth entry from the search results
bfi_sample["61", "age"]
# with the following search, both Males and Females are found
# (the M of Males, wouldn't be matched if ignCse were FALSE and males is
#  only a partial match within Females, thus whlTrm must be set to FALSE)
jmvReadWrite::search_omv(bfi_sample, "males", whlTrm = FALSE, ignCse = TRUE)
# the first entry is a female, the first entry is a male
bfi_sample["1", "gender"] # Females
bfi_sample["6", "gender"] # Males
# using the search results assigned to a variable
srcRes <- jmvReadWrite::search_omv(bfi_sample, "males", whlTrm = FALSE, ignCse = TRUE)
bfi_sample[srcRes[[1]][1], names(srcRes[1])] # Females
bfi_sample[srcRes[[1]][6], names(srcRes[1])] # Males
```

---

sort_omv *Sort data (using one or more variables) in .omv-files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org)*

---

## Description

Sort data (using one or more variables) in .omv-files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org)

## Usage

```
sort_omv(
  dtaInp = NULL,
  fleOut = "",
  varSrt = NULL,
  psvAnl = FALSE,
  usePkg = c("foreign", "haven"),
  selSet = "",
  ...
)
```

## Arguments

| | |
|---|---|
| dtaInp | Either a data frame or the name of a data file to be read (including the path, if required; "FILENAME.ext"; default: NULL); files can be of any supported file type, see Details below. |
| fleOut | Name of the data set / file to be written (including the path, if required; "FILE_OUT.omv"; default: ""); if empty, the resulting data frame is returned instead. |
| varSrt | Variable(s) that are used to sort the data frame (see Details; default: NULL) |
| psvAnl | Whether analyses that are contained in the input file shall be transferred to the output file (TRUE / FALSE; default: FALSE) |
| usePkg | Name of the package: "foreign" or "haven" that shall be used to read SPSS, Stata, and SAS files; "foreign" is the default (it is included in base R), but "haven" is newer and more comprehensive; you may have to install using `install.packages("haven", dep = TRUE)`. |
| selSet | Name of the object / data set that is to be selected from the workspace (only relevant when reading .RData-files which can contain several objects / data sets) |
| ... | Additional arguments passed on to methods; see Details below |

## Details

- `varSrt` can be either a character or a character vector (with one or more variables respectively). The sorting order for a particular variable can be inverted with preceding the variable name with "-". Please note that this doesn't make sense and hence throws a warning for certain variable types (e.g., factors).
- The ellipsis-parameter (`...`) can be used to submit arguments / parameters to the functions that are used for reading the data. By clicking on the respective function under "See also", you can get a more detailed overview over which parameters each of those functions take.

## Value

a data frame (only returned if `fleOut` is empty) where the input data set is sorted (according to the variables in `varSrt`)

## See Also

sort_omv internally uses the following functions for reading and writing data files in different formats: read_omv() and write_omv() for jamovi-files, utils::read.table() for CSV / TSV files, load() for reading .RData-files, readRDS() for .rds-files, haven::read_sav() or foreign::read.spss() for SPSS-files, haven::read_dta() or foreign::read.dta() for Stata-files, haven::read_sas() for SAS-data-files, and haven::read_xpt() or foreign::read.xport() for SAS-transport-files.

## Examples

```
nmeInp <- system.file("extdata", "AlbumSales.omv", package = "jmvReadWrite")
nmeOut <- tempfile(fileext = ".omv")
jmvReadWrite::sort_omv(dtaInp = nmeInp, fleOut = nmeOut, varSrt = "Image")
dtaFrm <- jmvReadWrite::read_omv(nmeOut)
unlink(nmeOut)
cat(dtaFrm$Image)
# shows that the variable "Image" is sorted in ascending order
cat(is.unsorted(dtaFrm$Image))
# is.unsorted (which checks for whether the variable is NOT sorted) returns FALSE
jmvReadWrite::sort_omv(dtaInp = nmeInp, fleOut = nmeOut, varSrt = "-Image")
# variables can also be sorted in descending order by preceding them with "-"
dtaFrm <- jmvReadWrite::read_omv(nmeOut)
unlink(nmeOut)
cat(dtaFrm$Image)
# shows that the variable "Image" is now sorted in descending order
cat(is.unsorted(dtaFrm$Image))
# this first returns TRUE (the variable is not in ascending order, i.e., unsorted)
cat(is.unsorted(-dtaFrm$Image))
# if the sign of the variable is changed, it returns FALSE (i.e., the variable is
# NOT unsorted)
```

---

# ToothGrowth *The Effect of Vitamin C on Tooth Growth in Guinea Pigs*

---

## Description

The Effect of Vitamin C on Tooth Growth in Guinea Pigs

## Usage

```
ToothGrowth
```

## Format

A data frame with 60 rows and 6 variables:

ID ID of the guinea pig

supp Supplement type (VC: Vitamin C or OJ: Orange juice)

supp2 Transformation of the supplement type (factor to numerical: VC = 1; OJ = 2)

dose Dose in grams / day

dose2 Dose in grams / day

len Tooth length

logLen Natural logarithm of the tooth length (len)

---

`transform_vars_omv` *Transform skewed variables (aiming at they conform to a normal distribution) in .omv-files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org)*

---

## Description

Transform skewed variables (aiming at they conform to a normal distribution) in .omv-files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org)

## Usage

```
transform_vars_omv(
  dtaInp = NULL,
  fleOut = "",
  varXfm = NULL,
  psvAnl = FALSE,
  usePkg = c("foreign", "haven"),
  selSet = "",
  ...
)
```

## Arguments

| | |
|---|---|
| dtaInp | Either a data frame or the name of a data file to be read (including the path, if required; "FILENAME.ext"; default: NULL); files can be of any supported file type, see Details below. |
| fleOut | Name of the data set / file to be written (including the path, if required; "FILE_OUT.omv"; default: ""); if empty, the resulting data frame is returned instead. |
| varXfm | Named list variable where the name indicates which transformation is to be carried out and where each list entry points to one or more variables to be transformed using this transformation. See Details for more information. |
| psvAnl | Whether analyses that are contained in the input file shall be transferred to the output file (TRUE / FALSE; default: FALSE) |
| usePkg | Name of the package: "foreign" or "haven" that shall be used to read SPSS, Stata, and SAS files; "foreign" is the default (it is included in base R), but "haven" is newer and more comprehensive; you may have to install using `install.packages("haven", dep = TRUE)`. |
| selSet | Name of the object / data set that is to be selected from the workspace (only relevant when reading .RData-files which can contain several objects / data sets) |
| ... | Additional arguments passed on to methods; see Details below |

## Details

- varXfm has to be a named list variable where the names can either indicate the type of transformation or the kind and degree of skewness that shall be corrected. For the type of transformation, the following names are valid: posSqr, negSqr, posLog, negLog, posInv, negInv; where the second part of the name indicates the transformation to be carried out: ...Sqr - square root, ...Log - logarithm to the basis 10, ...Inv - inversion, i.e., 1 / original value), and where the first part of the name indicates whether the original value is used (pos...) or whether the original value is subtracted from the maximum value of that variable (neg...; a constant of 1 is added to the maximum value for ...Log and ...Inv transformations). For the degree and kind of skewness, the following names are valid: mdrPos, strPos, svrPos, mdrNeg, strNeg, svrNeg (degree: moderate, strong, severe; kind: positive or negative). Since variables either are positively or negatively skewed they can only appear once within those entries requesting transformations for positively skewed variables (i.e., posSqr, posLog, or posInv) OR those entries requesting transformations for negatively skewed variables (i.e., negSqr, negLog, or negInv). If a variable appears, e.g., under posSqr AND negSqr an error message is shown.
- The ellipsis-parameter (...) can be used to submit arguments / parameters to the functions that are used for reading the data. By clicking on the respective function under "See also", you can get a more detailed overview over which parameters each of those functions take.

## Value

a data frame (only returned if fleOut is empty) where the order of variables / columns of the input data set is re-arranged

## See Also

transform_vars_omv internally uses the following functions for reading and writing data files in different formats: read_omv() and write_omv() for jamovi-files, utils::read.table() for CSV / TSV files, load() for reading .RData-files, readRDS() for .rds-files, haven::read_sav() or foreign::read.spss() for SPSS-files, haven::read_dta() or foreign::read.dta() for Stata-files, haven::read_sas() for SAS-data-files, and haven::read_xpt() or foreign::read.xport() for SAS-transport-files.

## Examples

```
# generate skewed variables
set.seed(335)
dtaInp <- data.frame(MP = rnorm(1000) * 1e-1 + rexp(1000, 2) * (1 - 1e-1),
                     MN = rnorm(1000) * 1e-1 - rexp(1000, 2) * (1 - 1e-1),
                     SP = rnorm(1000) * 1e-2 + rexp(1000, 2) * (1 - 1e-2),
                     SN = rnorm(1000) * 1e-2 - rexp(1000, 2) * (1 - 1e-2),
                     EP = rnorm(1000) * 1e-4 + rexp(1000, 2) * (1 - 1e-4),
                     EN = rnorm(1000) * 1e-4 - rexp(1000, 2) * (1 - 1e-4))
jmv::descriptives(data = dtaInp, skew = TRUE, sw = TRUE)

crrXfm <- list(posSqr = c("MP"), negSqr = c("MN"), posLog = c("MP", "SP"), negLog = c("SN"),
               posInv = c("MP", "SP", "EP"), negInv = c("EN"))
dtaOut <- jmvReadWrite::transform_vars_omv(dtaInp = dtaInp, varXfm = crrXfm)
```

```
jmv::descriptives(data = dtaOut, skew = TRUE, sw = TRUE)

crrXfm <- list(mdrPos = c("MP"), mdrNeg = c("MN"), strPos = c("SP"), strNeg = c("SN"),
               svrPos = c("EP"), svrNeg = c("EN"))
dtaOut <- jmvReadWrite::transform_vars_omv(dtaInp = dtaInp, varXfm = crrXfm)
jmv::descriptives(data = dtaOut, skew = TRUE, sw = TRUE)
```

---

`transpose_omv` *Transpose .omv-files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org)*

---

## Description

Transpose .omv-files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org)

## Usage

```
transpose_omv(
  dtaInp = NULL,
  fleOut = "",
  varNme = "",
  usePkg = c("foreign", "haven"),
  selSet = "",
  ...
)
```

## Arguments

| | |
|---|---|
| `dtaInp` | Either a data frame or the name of a data file to be read (including the path, if required; "FILENAME.ext"; default: NULL); files can be of any supported file type, see Details below. |
| `fleOut` | Name of the data set / file to be written (including the path, if required; "FILE_OUT.omv"; default: ""); if empty, the resulting data frame is returned instead. |
| `varNme` | Name of the variables in the output data frame; see Details below |
| `usePkg` | Name of the package: "foreign" or "haven" that shall be used to read SPSS, Stata, and SAS files; "foreign" is the default (it is included in base R), but "haven" is newer and more comprehensive; you may have to install using `install.packages("haven", dep = TRUE)`. |
| `selSet` | Name of the object / data set that is to be selected from the workspace (only relevant when reading .RData-files which can contain several objects / data sets) |
| `...` | Additional arguments passed on to methods; see Details below |

## Details

- If varNme empty, the row names of the input data set are used (preceded by "V_" if all row names are numbers); if varNme has length 1, then it is supposed to point to a variable in the input data frame; if varNme has the same length as the number of rows in the input data frame, then the values in varNme are assigned as column names to the output data frame.
- The ellipsis-parameter (...) can be used to submit arguments / parameters to the functions that are used for reading the data. By clicking on the respective function under "See also", you can get a more detailed overview over which parameters each of those functions take.

## Value

a data frame (only returned if fleOut is empty) where the input data set is transposed

## See Also

transpose_omv internally uses the following functions for reading and writing data files in different formats: read_omv() and write_omv() for jamovi-files, utils::read.table() for CSV / TSV files, load() for reading .RData-files, readRDS() for .rds-files, haven::read_sav() or foreign::read.spss() for SPSS-files, haven::read_dta() or foreign::read.dta() for Stata-files, haven::read_sas() for SAS-data-files, and haven::read_xpt() or foreign::read.xport() for SAS-transport-files.

## Examples

```
set.seed(1)
tmpDF <- stats::setNames(as.data.frame(matrix(sample(6, 1200, replace = TRUE), nrow = 16)),
                         sprintf("sbj_%03d", seq(75)))
str(tmpDF)
# Data sets that were extracted, e.g., from PsychoPy, may look like this (trials as rows
# and participants as columns, one for each participant, manually assembled / copy-and-pasted).
# However, for analyses, one wants the data set transposed (units / participants as columns)...
nmeOut <- tempfile(fileext = ".omv")
jmvReadWrite::transpose_omv(dtaInp = tmpDF, fleOut = nmeOut)
dtaFrm <- jmvReadWrite::read_omv(nmeOut)
unlink(nmeOut)
str(dtaFrm)
# if no varNme-parameter is given, generic variable names are created (V_...)
jmvReadWrite::transpose_omv(dtaInp = tmpDF, fleOut = nmeOut, varNme = sprintf("Trl_%02d", seq(16)))
dtaFrm <- jmvReadWrite::read_omv(nmeOut)
unlink(nmeOut)
str(dtaFrm)
# alternatively, the character vector with the desired variable names (of the same length as
# the number of rows in tmpDF) may be given, "Trl" can easily be exchanged by the name of your
# questionnaire, experimental conditions, etc.
```

wide2long_omv *Converts .omv-files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org) from wide to long format*

## Description

Converts .omv-files for the statistical spreadsheet 'jamovi' (https://www.jamovi.org) from wide to long format

## Usage

```
wide2long_omv(
  dtaInp = NULL,
  fleOut = "",
  varLst = NULL,
  varExc = NULL,
  varID = NULL,
  varTme = "cond",
  varSep = "_",
  varOrd = TRUE,
  varSrt = NULL,
  excLvl = NULL,
  usePkg = c("foreign", "haven"),
  selSet = "",
  ...
)
```

## Arguments

| | |
|---|---|
| dtaInp | Either a data frame or the name of a data file to be read (including the path, if required; "FILENAME.ext"; default: NULL); files can be of any supported file type, see Details below. |
| fleOut | Name of the data set / file to be written (including the path, if required; "FILE_OUT.omv"; default: ""); if empty, the resulting data frame is returned instead. |
| varLst | List / set of variables that are to be transformed into single (time-varying) variables in long format (default: NULL) |
| varExc | Name of the variable(s) should be excluded from the transformation, typically this will be between-subject-variable(s) (default: NULL) |
| varID | Name(s) of one or more variables that (is created to) identify the same group / individual (if empty, "ID" is added with row numbers identifying cases; default: NULL) |
| varTme | Name of the variable (or vector with variable names) that (is created to) differentiate multiple records from the same group / individual (default: "cond"; if required, a counter is added for each time-varying part) |

| | |
|---|---|
| varSep | Character that separates the variables in varLst into a time-varying part and a part that forms the variable name in long format (*"" in "VAR_1", "VAR_2", default: ""*) |
| varOrd | Whether to arrange the variables before the transformation, so that they are in accordance with the different split levels (default: TRUE) |
| varSrt | Variable(s) that are used to sort the data frame (see Details; if empty, the order returned from reshape is kept; default: NULL) |
| excLvl | Integer (or vector of integers) determining which parts of the variable names in varLst shall not be transformed (default: NULL), see Details below |
| usePkg | Name of the package: "foreign" or "haven" that shall be used to read SPSS, Stata, and SAS files; "foreign" is the default (it is included in base R), but "haven" is newer and more comprehensive; you may have to install using `install.packages("haven", dep = TRUE)`. |
| selSet | Name of the object / data set that is to be selected from the workspace (only relevant when reading .RData-files which can contain several objects / data sets) |
| ... | Additional arguments passed on to methods; see Details below |

## Details

- If `varLst` is empty, it is tried to generate it using all variables in the data frame except those defined by `varExc` and `varID`. The variable(s) in `varID` need to be unique identifiers (in the original dataset), those in `varExc` don't have this requirement. It is recommended that the variable names in `varExc` and `varID` should not contain the variable separator (defined in `varSep`; default: "_").
- `varOrd` determines whether the variables are rearranged to match the order of split levels. Consider the `varLst` X_1, Y_1, X_2, Y_2. If `varOrd` were set to FALSE, the original order would be preserved and the second part of the variable name (1, 2, ...) would become condition 1, and the first part condition 2. In most cases, leaving `varOrd` set to TRUE is recommended.
- `varSrt` can be either a character or a character vector (with one or more variables respectively). The sorting order for a particular variable can be inverted with preceding the variable name with "-". Please note that this doesn't make sense and hence throws a warning for certain variable types (e.g., factors).
- `exclLvl` points to a part of the variable names in `varLst` to be excluded. For example, if the variable name is `PART1_PART2_PART3` (split at _), then `excLvl = 1` would exclude PART1 from the transformation. Quite often, one has more that one variable of a particular type (e.g., responses, reaction times, etc.). Those would typically be the first part of each variable name in `varLst` (the conditions then being PART2, PART3, and so on). `excLvl = 1` would exclude those variable types / categories from being transformed into long (i.e., they would be kept as separate columns).
- The transformation from long to wide uses `reshape`: `varID` matches (~) `idvar` in `reshape`, `varTme ~ timevar`, `varLst ~ varying`, and `varSep ~ sep`. The help for `reshape` is very explanatory, click on the link under "See also" to access it, particularly what is explained under "Details".
- The ellipsis-parameter (`...`) can be used to submit arguments / parameters to the functions that are used for reading or transforming the data. By clicking on the respective function under "See also", you can get a more detailed overview over which parameters each of those functions take.

## Value

a data frame (only returned if `fleOut` is empty) where the input data set is converted from wide to long format

## See Also

`long2wide_omv` internally uses the following functions: The transformation from long to wide uses `stats::reshape()`. For reading and writing data files in different formats: `read_omv()` and `write_omv()` for jamovi-files, `utils::read.table()` for CSV / TSV files, `load()` for reading .RData-files, `readRDS()` for .rds-files, `haven::read_sav()` or `foreign::read.spss()` for SPSS-files, `haven::read_dta()` or `foreign::read.dta()` for Stata-files, `haven::read_sas()` for SAS-data-files, and `haven::read_xpt()` or `foreign::read.xport()` for SAS-transport-files.

## Examples

```
## Not run:
# generate a test dataframe with 100 (imaginary) participants / units of
# observation (ID), and 8 repeated measurements of variable (X_1, X_2, ...)
dtaInp <- cbind(data.frame(ID = as.character(seq(1:100))),
                stats::setNames(
                    as.data.frame(matrix(runif(800, -10, 10), nrow = 100)),
                    paste0("X_", 1:8)))
cat(str(dtaInp))
# 'data.frame': 100 obs. of  9 variables:
#  $ ID : chr  "1" "2" "3" "4" ...
#  $ X_1: num  ...
#  $ X_2: num  ...
#  $ X_3: num  ...
#  $ X_4: num  ...
#  $ X_5: num  ...
#  $ X_6: num  ...
#  $ X_7: num  ...
#  $ X_8: num  ...
# this data set is stored as (temporary) RDS-file and later processed by wide2long
nmeInp <- tempfile(fileext = ".rds")
nmeOut <- tempfile(fileext = ".omv")
saveRDS(dtaInp, nmeInp)
jmvReadWrite::wide2long_omv(dtaInp = nmeInp, fleOut = nmeOut, varID = "ID",
    varTme = "measure", varLst = setdiff(names(dtaInp), "ID"),
    varSrt = c("ID", "measure"))
# it is required to give at least the arguments dtaInp (if dtaInp is a data frame,
# fleOut needs to be provided too) and varID
# "reshape" then assigns all variables expect the variable defined by varID to
# varLst (but throws a warning)
# varSrt enforces sorting the data set after the transformation (sorted, the
# measurements within one person come after another; unsorted all measurements
# for one repetition would come after another)

# check whether the file was created and its size
cat(list.files(dirname(nmeOut), basename(nmeOut)))
# -> "file[...].omv" ([...] contains a random combination of numbers / characters
```

```
cat(file.info(nmeOut)$size)
# -> 6939 (approximate size; size may differ in every run [in dependence of how
#          well the generated random data can be compressed])
cat(str(jmvReadWrite::read_omv(nmeOut, sveAtt = FALSE)))
# the data set is now transformed into long (and each the measurements is now
# indicated by the "measure")
# 'data.frame': 800 obs. of  3 variables:
#  $ ID     : Factor w/ 100 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 2 2 ...
#   ..- attr(*, "missingValues")= list()
#  $ measure: Factor w/ 8 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 1 2 ...
#   ..- attr(*, "missingValues")= list()
#  $ X      : num  ...
#   ..- attr(*, "missingValues")= list()

unlink(nmeInp)
unlink(nmeOut)

## End(Not run)
```

---

`write_omv` *Write files to be used with the statistical spreadsheet 'jamovi' (https://www.jamovi.org)*

---

## Description

Write files to be used with the statistical spreadsheet 'jamovi' (https://www.jamovi.org)

## Usage

```
write_omv(
  dtaFrm = NULL,
  fleOut = "",
  wrtPtB = FALSE,
  frcWrt = FALSE,
  retDbg = FALSE,
  vldExt = TRUE
)
```

## Arguments

| | |
|---|---|
| `dtaFrm` | Data frame to be exported (default: NULL) |
| `fleOut` | Name / position of the output file to be generated ("FILENAME.omv"; default: "") |
| `wrtPtB` | Whether to write protocol buffers (see Details; default: FALSE) |
| `frcWrt` | Whether to overwrite existing files with the same name (see Details; default: FALSE) |
| `retDbg` | Whether to return a list with debugging information (see Value; default: FALSE) |
| `vldExt` | Whether to validate that the file name has a correct extension (default: TRUE) |

## Details

- jamovi has a specific measurement level / type "ID" (in addition to the "standard" ones "Nominal", "Ordinal", and "Continuous"). "ID" is used for columns that contain some form of ID (e.g., a participant code). In order to set a variable of your data frame to "ID", you have to set the attribute `jmv-id` (e.g., `attr(dtaFrm$column, "jmv-id") = TRUE`).
- CAUTION: Setting wrtPtB to TRUE currently overwrites analyses that already exist in a data file. It is meant to be used for `describe_omv` only. If you set wrtPtB to TRUE, ensure to use an output file name that isn't would not overwrite any existing file. Protocol buffers are used to exchange data between the different parts of jamovi (the server and the client) and also the format in which analyses are stored in the jamovi data files.
- `write_omv` checks whether the output file already exists and throws an error if this is the case. frcWrt permits you to overwrite the existing file.

## Value

a list (if retDbg == TRUE), containing the meta data (mtaDta, metadata.json in the OMV-file), the extended data (xtdDta, xdata.json in the OMV-file) and the original data frame (dtaFrm)

## Examples

```
# use the data set "ToothGrowth" and, if it exists, write it as
# jamovi-file using write_omv()
jmvReadWrite::ToothGrowth
nmeOut <- tempfile(fileext = ".omv")
# typically, one would use a "real" file name instead of tempfile(),
# e.g., "Data1.omv"
dtaDbg = jmvReadWrite::write_omv(dtaFrm = ToothGrowth, fleOut = nmeOut, retDbg = TRUE)
print(names(dtaDbg))
# the print-function is only used to force devtools::run_examples()
# to show output
# -> "mtaDta" "xtdDta" "dtaFrm"
# returns a list with the metadata (mtaDta, e.g., column and data type),
# the extended data (xtdDta, e.g., variable lables), and the data frame
# (dtaFrm) the purpose of these variables is merely for checking (under-
# standing the file format) and debugging

# check whether the file was written to the disk, get the file informa-
# tion (size, etc.) and delete the file afterwards
print(list.files(dirname(nmeOut), basename(nmeOut)))
# -> "file[...].omv" ([...] is a combination of random numbers / characters
print(file.info(nmeOut)$size)
# -> approx. 2600 (size may differ on different OSes)
unlink(nmeOut)
```

# Index


∗ **datasets**
AlbumSales, 6
bfi_sample, 9
bfi_sample2, 11
bfi_sample3, 12
ToothGrowth, 44

aggregate_omv, 2
AlbumSales, 6
arrange_cols_omv, 7

base::is.na(), *5*
base::max(), *5*
base::mean(), *5*
base::min(), *5*
base::range(), *5*
base::sum(), *5*
bfi_sample, 9
bfi_sample2, 11
bfi_sample3, 12

combine_cols_omv, 14
convert_to_omv, 16

describe_omv, 18
distances_omv, 21

foreign::read.dta(), *5*, *9*, *15*, *17*, *20*, *24*, *27*, *30*, *33*, *35*, *40*, *42*, *44*, *46*, *48*, *51*
foreign::read.spss(), *5*, *9*, *15*, *17*, *20*, *24*, *27*, *29*, *33*, *35*, *40*, *42*, *44*, *46*, *48*, *51*
foreign::read.xport(), *5*, *9*, *15*, *17*, *20*, *24*, *27*, *30*, *33*, *35*, *40*, *42*, *44*, *46*, *48*, *51*

haven::read_dta(), *5*, *9*, *15*, *17*, *20*, *24*, *27*, *30*, *33*, *35*, *40*, *42*, *44*, *46*, *48*, *51*
haven::read_sas(), *5*, *9*, *15*, *17*, *20*, *24*, *27*, *30*, *33*, *35*, *40*, *42*, *44*, *46*, *48*, *51*
haven::read_sav(), *5*, *9*, *15*, *17*, *20*, *24*, *27*, *29*, *33*, *35*, *40*, *42*, *44*, *46*, *48*, *51*
haven::read_xpt(), *5*, *9*, *15*, *17*, *20*, *24*, *27*, *30*, *33*, *35*, *40*, *42*, *44*, *46*, *48*, *51*

label_vars_omv, 25
load(), *5*, *9*, *15*, *17*, *20*, *24*, *27*, *29*, *33*, *35*, *40*, *42*, *44*, *46*, *48*, *51*
long2wide_omv, 28

merge(), *33*
merge_cols_omv, 31
merge_rows_omv, 33

rbind(), *35*
read_omv, 36
read_omv(), *5*, *9*, *15*, *17*, *20*, *24*, *27*, *29*, *33*, *35*, *40*, *42*, *44*, *46*, *48*, *51*
readRDS(), *5*, *9*, *15*, *17*, *20*, *24*, *27*, *29*, *33*, *35*, *40*, *42*, *44*, *46*, *48*, *51*
replace_omv, 38

search_omv, 40
sort_omv, 42
stats::aggregate(), *5*
stats::dist(), *24*
stats::median(), *5*
stats::quantile(), *5*
stats::reshape(), *29*, *51*
stats::sd(), *5*
stats::var(), *5*

ToothGrowth, 44
transform_vars_omv, 45
transpose_omv, 47

utils::read.table(), *5*, *9*, *15*, *17*, *20*, *24*, *27*, *29*, *33*, *35*, *40*, *42*, *44*, *46*, *48*, *51*

wide2long_omv, 49
write_omv, 52
write_omv(), *5*, *9*, *15*, *17*, *20*, *24*, *27*, *29*, *33*, *35*, *40*, *42*, *44*, *46*, *48*, *51*