

# Package ‘OuhscMunge’

February 7, 2018

**Title** Data Manipulation Operations

**Description** Data manipulation operations frequently used in OUHSC BBMC projects.

**Version** 0.1.8.9006

**Date** 2018-01-30

**Author** Will Beasley [aut, cre]

**Maintainer** Will Beasley <wibeasley@hotmail.com>

**URL** <https://github.com/OuhscBbmc/OuhscMunge>, <http://ouhsc.edu/bbmc/>

**BugReports** <https://github.com/OuhscBbmc/OuhscMunge/issues>

**License** GPL-2

**LazyData** TRUE

**Depends** R(>= 3.4.0)

**Imports** checkmate,  
DBI,  
devtools (>= 1.8.0),  
dplyr,  
glue,  
lubridate,  
magrittr,  
odbc,  
purrr,  
readr,  
remotes,  
rlang,  
testit,  
tibble,

**Suggests** RODBC,  
RODBCext,  
testthat

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 6.0.1

R topics documented:

assert . . . . .	2
clump_date . . . . .	3
deterge . . . . .	4
execute_sql_file . . . . .	5
first_nonmissing . . . . .	6
headstart_utilities . . . . .	6
match_statistics . . . . .	7
open_dsn_channel_sqls . . . . .	8
OuhscMunge . . . . .	9
replace_nas_with_explicit . . . . .	10
replace_with_nas . . . . .	11
retrieve_key_value . . . . .	12
snake_case . . . . .	13
trim . . . . .	14
update_packages_addin . . . . .	15
upload_sqls_rodbs . . . . .	16
verify_data_frame . . . . .	17
verify_value_headstart . . . . .	17
<b>Index</b>	<b>19</b>

---

assert	<i>Assert vector characteristics</i>
--------	--------------------------------------

---

Description

Assert a vector meets important data-quality requirements.

Usage

```
assert_non_na(x, class_vector, proportion_minimum)
assert_non_na_and_unique(x, class_vector)
```

Arguments

- x                      Vector to inspect. Required.
- class\_vector        The required `base::class()` of the vector. If the parameter is missing, the object's class is not checked.
- proportion\_minimum        The (inclusive) minimum proportion of the vector's elements that should meet the requirement. If missing, all elements must pass.

Examples

```
requireNamespace("OuhscMunge")
OuhscMunge::assert_non_na(1:30, "integer")
## Not run:
OuhscMunge::assert_non_na(c(1:30, NA_integer_), "integer")

## End(Not run)
```

---

clump\_date*Assign date for a given year & month*

---

## Description

This accepts a date, but changes the day. Set/degrade/clump all the days within a month/week to the same day.

## Usage

```
clump_month_date( date_detailed, day_of_month=15L )  
clump_week_date( date_detailed, day_of_week=2L )
```

## Arguments

date_detailed	The Date value containing the desired year and month. The day will be overwritten. Required
day_of_month	The factor label assigned to the missing value. Defaults to 15 ( <i>i.e.</i> , the middle of the month).
day_of_week	The factor label assigned to the missing value. Defaults to 2 ( <i>i.e.</i> , Monday).

## Details

We use this frequently to set/degrade/clump all the days to the middle of their respective month (ie, the 15th day). The midpoint of a month is usually the most appropriate summary location. It makes graphs more intuitive. Using the midpoint of month can also avoid problems with timezones. A date won't get nudged to a neighboring month accidentally.

## Value

An array of Date values.

## Note

A stop error will be thrown if date\_detailed is not a Date. day\_of\_month must be bounded by [1, 31], and day\_of\_week must be bounded by [1, 7]. Be careful that if you set a November date the 31st day, the result will be December 1st. Consequently, we recommend not setting the day to a value after the 28.

## Author(s)

Will Beasley

## See Also

These functions are gloves around `lubridate::day()` and `lubridate::wday()`. Essentially the add just error-checking and default values.

## Examples

```
library(OuhscMunge)
detailed <- seq.Date(from=as.Date("2011-04-21"), to=as.Date("2011-07-14"), by="day")

clumped_month <- clump_month_date(detailed)
table(clumped_month)
# 2011-04-15 2011-05-15 2011-06-15 2011-07-15
#           10          31          30          14

clumped_week <- clump_week_date(detailed)
table(clumped_week)
# 2011-04-18 2011-04-25 2011-05-02 2011-05-09 2011-05-16 2011-05-23 2011-05-30
#           3           7           7           7           7           7           7
# 2011-06-06 2011-06-13 2011-06-20 2011-06-27 2011-07-04 2011-07-11
#           7           7           7           7           7           5
```

---

deterge

---

*Convert (and possibly clean) a vector*


---

## Description

Cast values to desired data type.

## Usage

```
deterge_to_double(x, bound_lower, bound_upper)
deterge_to_integer(x, bound_lower, bound_upper)
deterge_to_ascii(x, substitution_character)
```

## Arguments

x	The input vector that needs to be cast/converted. Required.
bound_lower	Elements below this inclusive threshold will be set to NA.
bound_upper	Elements above this inclusive threshold will be set to NA.
substitution_character	If the character does not have an equivalent in ASCII, replace it with this character. Defaults to a question mark ( <i>i.e.</i> , '?').

## Details

The functions `deterge_to_double()` and `deterge_to_integer()` accept character representations of a number, and return a numeric or integer vector. Elements outside `bound_lower` and `bound_upper` are converted to `NA_real_`/`NA_integer_`.

The function `deterge_to_ascii()` accepts a character vector and returns a character vector. The encoding is changed to ASCII. Individual elements are allowed to be `NA_character_`.

## Value

An array of values.

**Author(s)**

Will Beasley

**See Also**

The real work in `deterge_to_ascii()` is performed by `base::iconv()`. `base::iconv(x=x, from="latin1", to="AS`

**Examples**

```
library(OuhscMunge)
deterge_to_double(c(NA, 1:10), 4, 8)
deterge_to_integer(c(NA, 1:10), 4L, 8L)

x <- c("Ekstr\xfb8m", "J\xfb6reskog", "bi\xdfchen Z\xfbcrcher")
deterge_to_ascii(x)
```

---

execute_sql_file	<i>Execute a SQL file</i>
------------------	---------------------------

---

**Description**

Read a SQL file, and execute its text using the `odbc` and `DBI` packages.

**Usage**

```
execute_sql_file(path_sql, dsn)
```

**Arguments**

<code>path_sql</code>	A vector to of names to convert. Required character.
<code>dsn</code>	The name of a <b>DSN</b> defined on your local machine Required character.

**Value**

A vector of converted names.

**Author(s)**

Will Beasley

**Examples**

```
## Not run:
execute_sql_file("inst/hdid-select.sql", "cdw_cache_staging")
execute_sql_file("inst/condense-date.sql", "cdw_cache_staging")

## End(Not run)
```

---

first_nonmissing	<i>First nonmissing element in a vector</i>
------------------	---

---

**Description**

Take the first value that isn't missing. Adapted from <http://stackoverflow.com/a/40515261/1082435>.

**Usage**

```
first_nonmissing(x)
```

**Arguments**

x	A vector to of names to convert.
---	----------------------------------

**Value**

A vector of converted names.

**Author(s)**

Will Beasley

**Examples**

```
first_nonmissing(c(NA, "b", "c"))
first_nonmissing(c(NA_character_, NA_character_))
first_nonmissing(character(0))
```

---

headstart_utilities	<i>Utilities for outputting characteristics of a dataset used it code.</i>
---------------------	--

---

**Description**

These functions are used during the execution of a program. Rather they produce snippets that can be pasted into code, and help the developer avoid some typing.

**Usage**

```
column_rename_headstart( d, try_snake_case=TRUE )
column_class_headstart( d )
column_value_headstart( x )
```

**Arguments**

d	A data.frame to describe.
try_snake_case	If TRUE column names are attempted to be converted to snake_case.
x	A vector to describe.

**Value**

Prints formatted code to the console.

**Author(s)**

Will Beasley

**Examples**

```
column_rename_headstart(datasets::OrchardSprays)
column_rename_headstart(datasets::iris)
column_class_headstart(datasets::OrchardSprays)
column_value_headstart(datasets::OrchardSprays$treatment)
```

---

match_statistics	<i>Create explicit factor level for missing values.</i>
------------------	---

---

**Description**

Missing values are converted to a factor level. This explicit assignment can reduce the chances that missing values are inadvertently ignored. It also allows the presence of a missing to become a predictor in models.

**Usage**

```
match_statistics(d_parent, d_child, join_columns)
```

**Arguments**

d_parent	A data.frame of the parent table.
d_child	A data.frame of the child table.
join_columns	The character vector of the column names used to join to parent and child tables.

**Details**

If a nonexistent column is passed to join\_columns, an error will be thrown naming the violating column name.

More information about the 'parent' and 'child' terminology and concepts can be found in the [Hierarchical Database Model](#) Wikipedia entry, among many other sources.

**Value**

A numeric array of the following elements:

- parent\_in\_child The count of parent records found in the child table.
- parent\_not\_in\_child The count of parent records *not* found in the child table.
- parent\_na\_any The count of parent records with a NA in at least one of the join columns.
- deadbeat\_proportion The proportion of parent records *not* found in the child table.
- child\_in\_parent The count of child records found in the parent table.

- `child_not_in_parent` The count of child records *not* found in the parent table.
- `child_na_any` The proportion of child records *not* found in the parent table.
- `orphan_proportion` The count of child records with a NA in at least one of the join columns.

### Note

The `join_columns` parameter is passed directly to `dplyr::semi_join()` and `dplyr::anti_join()`.

### Author(s)

Will Beasley

### Examples

```
ds_parent <- data.frame(
  parent_id      = 1L:10L,
  letter         = rep(letters[1:5], each=2),
  index          = rep(1:2, times=5),
  dv             = runif(10),
  stringsAsFactors = FALSE
)
ds_child <- data.frame(
  child_id       = 101:140,
  parent_id      = c(4, 5, rep(6L:14L, each=4), 15, 16),
  letter         = rep(letters[3:12], each=4),
  index          = rep(1:2, each=2, length.out=40),
  dv             = runif(40),
  stringsAsFactors = FALSE
)

#Match on one column:
match_statistics(ds_parent, ds_child, join_columns="parent_id")

#Match on two columns:
match_statistics(ds_parent, ds_child, join_columns=c("letter", "index"))
```

---

`open_dsn_channel_sqls` *Open an ODBC channel to a SQL Server database*

---

### Description

Creates & opens a channel and checks its important characteristics.

### Usage

```
open_dsn_channel_sqls(dsn_name,
  driver_version_minimum = numeric_version("13.0"),
  driver_version_maximum = numeric_version("99.0"))
```



## Arguments

<code>dsn_name</code>	Name of the locally-defined DSN passed to <code>RODBC::odbcConnect()</code> .
<code>driver_version_minimum</code>	The driver must be at least this version number. Represented as a <code>base::numeric_version()</code>
<code>driver_version_maximum</code>	The driver must not exceed this version number. Represented as a <code>base::numeric_version()</code>

## Details

A DSN channel requires more code than usual to diagnose problems, because the DSN is defined on the local computer, and is not under the control of the repository. This function wraps the basic `RODBC::odbcConnect()` function with some checks. If unsuccessful, it returns some hints how to correct the problem, such as downloading the newest version from the [Microsoft website](#).

## Note

Assuring a minimum version is important, because driver versions can interpret values differently. For example, earlier version (before 11.0) returned dates as characters, which would propagate undetected through our code until it broke something with an unhelpful error message.

## Examples

```
## Not run:
requireNamespace("OuhscMunge")

OuhscMunge::open_dsn_channel_sqls(
  dsn_name      = "miechv_eval"
)

## End(Not run)
```

---

OuhscMunge	<i>Data manipulation operations frequently used in OUHSC BBMC projects.</i> <a href="http://www.ouhsc.edu/bbmc/">http://www.ouhsc.edu/bbmc/</a>
------------	---

---

## Description

Thanks to Funders, including [HRSA/ACF D89MC23154](#)  
*OUHSC CCAN Independent Evaluation of the State of Oklahoma Competitive Maternal, Infant, and Early Childhood Home Visiting (MIECHV) Project.*, which evaluates MIECHV expansion and enhancement of Evidence-based Home Visitation programs in four Oklahoma counties.

## Details

OuhscMunge.

## Note

The release version will eventually be available through CRAN by running `install.packages('OuhscMunge')`. The most recent development version is available through [GitHub](#) by running `devtools::install_github(repo = 'OuhscBbmc/OuhscMunge')` (make sure `devtools` is already installed). If you're having trouble with the package, please install the development version. If this doesn't solve your problem, please create a [new issue](#), or email Will.

**Author(s)**

**William Howard Beasley**, University of Oklahoma Health Sciences Center, College of Medicine, Dept of Pediatrics, **BBMC**.

Maintainer: Will Beasley [wibeasley@hotmail.com](mailto:wibeasley@hotmail.com)

**Examples**

```
## Not run:
# Install/update REDCapR with the release version from CRAN.
install.packages('OuhscMunge') #But it's not on CRAN yet.

# Install/update REDCapR with the development version from GitHub
#install.packages('devtools') #Uncomment if `devtools` isn't installed already.
devtools::install_github('OuhscBmc/OuhscMunge')

## End(Not run)
```

---

```
replace_nas_with_explicit
```

*Create explicit factor level for missing values.*

---

**Description**

Missing values are converted to a factor level. This explicit assignment can reduce the chances that missing values are inadvertently ignored. It also allows the presence of a missing to become a predictor in models.

The function is retained so existing code doesn't break. For new code, consider using `dplyr::coalesce()`. if you don't need to convert the missing code to a factor level.

**Usage**

```
replace_nas_with_explicit(scores, new_na_label = "Unknown",
  create_factor = FALSE, add_unknown_level = FALSE)
```

**Arguments**

<code>scores</code>	An array of values, ideally either factor or character. Required
<code>new_na_label</code>	The factor label assigned to the missing value. Defaults to Unknown.
<code>create_factor</code>	Converts scores into a factor, if it isn't one already. Defaults to FALSE.
<code>add_unknown_level</code>	Should a new factor level be created? (Specify TRUE if it already exists.) Defaults to FALSE.

**Value**

An array of values, where the NA values are now a factor level, with the label specified by the `new_na_label` value.

**Note**

The `create_factor` parameter is respected only if `scores` isn't already a factor. Otherwise, levels without any values would be lost.

A stop error will be thrown if the operation fails to convert all the NA values.

**Author(s)**

Will Beasley

**Examples**

```
library(OuhscMunge) #Load the package into the current R session.
missing_indices <- c(3, 6, 8, 25)
# With a character variable:
a <- letters
a[missing_indices] <- NA_character_
a <- OuhscMunge::replace_nas_with_explicit(a)

# With a factor variable:
b <- factor(letters, levels=letters)
b[missing_indices] <- NA_character_
b <- OuhscMunge::replace_nas_with_explicit(b, add_unknown_level=TRUE)
```

---

replace_with_nas	<i>Convert blank, zero-length values to NAs for a variety of data types.</i>
------------------	--

---

**Description**

Elements of zero-length are converted to NAs. Can force coercion to an optionally-specified data type.

The function is retained so existing code doesn't break. For new code, consider using `dplyr::na_if()`.

**Usage**

```
replace_with_nas(x, return_type = NULL)
```

**Arguments**

<code>x</code>	An array of values. Required
<code>return_type</code>	Data type of returned vector. Optional

**Details**

If `return_type` is missing, returned data type will match input. Supports coercion to integer, numeric, character, logical, and Date vectors.

If `return_type=logical`, a logical vector will be returned if `x` contains only blanks and the characters "0" and "1".

**Value**

An array of values with NAs.

**Note**

Contact the package author if you'd like the function generalized so that additional values (other than "") are converted to NAs.

**Author(s)**

Will Beasley

**Examples**

```
library(OuhscMunge) #Load the package into the current R session.
replace_with_nas(c("a", "b", "", "d", ""))
replace_with_nas(c("a", "b", "", "d", ""), return_type="character")

replace_with_nas(c(1, 2, "", "", 5), return_type="character")
replace_with_nas(c(1, 2, "", "", 5)) #Equivalent to previous line.
replace_with_nas(c(1, 2, "", "", 5), return_type="integer")
replace_with_nas(c(1, 2, "", "", 5), return_type="numeric")

replace_with_nas(c("2011-02-03", "", "", "2011-02-24"), return_type="Date")
replace_with_nas(c("T", "", "", "F", "FALSE", "", "TRUE"), return_type="logical")
replace_with_nas(c("1", "", "", "0", "0", "", "1"), return_type="logical")
```

---

retrieve_key_value	<i>Read a value stored in a database.</i>
--------------------	---

---

**Description**

Facilitates retrieval of key-value pairs that are stored in a SQL Server database.

**Usage**

```
retrieve_key_value(key, project_name, dsn_name, channel = NULL)
```

**Arguments**

key	The key associated with the desired value. Required character vecotr with one element
project_name	The project name associated with the desired value. Required character vecotr with one element
dsn_name	Name of the locally-defined DSN passed to <code>RODBC::odbcConnect()</code> .
channel	An <i>optional</i> connection handle as returned by <code>RODBC::odbcConnect()</code> . See Details below. Optional.

**Details**

The database table and stored procedure must defined as:

```

CREATE TABLE [security_private].[tbl_key_value_static](
    [id] [smallint] IDENTITY(1,1) NOT NULL,
    [project] [varchar](50) NOT NULL,
    [attribute] [varchar](90) NOT NULL,
    [value] [varchar](200) NOT NULL,
    [file_last_updated_date] [date] NOT NULL,
    [retired] [bit] NOT NULL,
    CONSTRAINT [PK_tblUr1] PRIMARY KEY CLUSTERED
    (
        [id] ASC
    )
) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, A
) ON [PRIMARY]

CREATE PROCEDURE [Security].[prc_key_value_static]
    @project varchar(50),
    @attribute varchar(90)
AS
BEGIN
    SET NOCOUNT ON;
    SELECT value from security_private.tbl_key_value_static
    WHERE project=@project AND attribute=@attribute
END

```

**Value**

A character vector with one element.

**Note**

Currently only the 'static' key-value pairs are retrieved through this function. Talk to Will if you need to retrieve the 'rolling' or the 'personal' key-value pairs.

**Author(s)**

Will Beasley

**Examples**

```

## Not run:
value <- retrieve_key_value("file-server", "bbmc", "BbmcSecurity")

## End(Not run)

```

---

snake\_case

---

*Convert variable names to snake\_case*


---

**Description**

This function attempts to convert variables to snake\_case, even if it's already in snake\_case. The important regex lines were posted by Stack Overflow user [epost](#) in "[Elegant Python function to convert CamelCase to snake\\_case?](#)".

**Usage**

```
snake_case(x)
```

**Arguments**

**x** A vector to of names to convert.

**Value**

A vector of converted names.

**Note**

This series of regexes has an advantages over the current implementations of `lettercase::str_snake_case()` and `snakecase::to_snake_case()`. The former converts "PatientDOB" to "patientdob" and the latter converts "patient.dob" to "patient\_.dob". I'll keep an eye on these packages (*i.e.*, `lettercase #1` for 'camelCase' and `snakecase #101`). I'd prefer to use one of them, instead of maintaining the functions.

**Author(s)**

Will Beasley

**Examples**

```
snake_case(colnames(datasets::OrchardSprays))
snake_case(colnames(datasets::iris))
snake_case(c("PatientID", "PatientDOB", "DOB", "name.last", "name.first"))
```

---

trim

*Trim extreme values*


---

**Description**

Trim extreme values from an atomic vector, and replace with a specific value (typically NA\_\*).

**Usage**

```
trim_numeric(x, bounds=c(-Inf, Inf), replacement=NA_real_)
trim_integer(x, bounds=c(-2147483647L, 2147483647L), replacement=NA_integer_)
trim_date(
  x,
  bounds      = as.Date(c("1940-01-01", "2030-01-01")),
  replacement = as.Date(NA_character_)
)
```

**Arguments**

**x** The input vector to be trimmed. Required

**bounds** A two-element vector that establishes the lower and upper *inclusive* bounds of x.

**replacement** A scalar that will replace all instances of x that fall outside of bounds.

**Value**

An atomic vector with the same number of elements as `x`.

**Note**

The data type of `x`, `bounds`, and `replacement` must match the atomic data type of the function. In other words, `trim_numeric()` accepts only parameters of type 'numeric' (otherwise known as 'double-precision floating point'). Likewise, `trim_date()` accepts only parameters of type `Date`.

The lower bound must be less than or equal the upper bound.

The default bounds for numerics and integers are at the extremes of the data type. The default bounds for **dates** are arbitrary, because the origin is slippery.

**Author(s)**

Will Beasley

**Examples**

```
library(OuhscMunge)
trim_numeric(runif(10, -1, 10), bounds=c(4, 8))
trim_integer(c(NA, 1:10), bounds=c(4L, 8L))
trim_date(
  x = as.Date(c("1902-02-02", "1999-09-09", "2020-02-22", "1930-01-01", "1930-01-02")),
  bounds = as.Date(c("1990-01-01", "2030-01-01"))
)
```

---

update\_packages\_addin *Download and install dependencies*

---

**Description**

When called in the repo of an R package, its package dependencies are inspected and the obsolete ones are updated. This function is a thin wrapper around `remotes::update_packages(remotes::dev_package_deps(` Unlike the 'Update' button in RStudio's 'Packages' panel, this function will (a) update from CRAN and remote sources like GitHub and (b) not attempt to install local packages that are unrelated to the current package.

**Usage**

```
update_packages_addin()
```

**Note**

This function only works if run inside a valid package. It reads the dependencies enumerated in the package's **DESCRIPTION** file.

upload\_sqls\_rodbs

*Upload to a SQL Server database using RODBC***Description**

The function performs some extra configuration to improve robustness.

**Usage**

```
upload_sqls_rodbs(d, table_name, dsn_name, clear_table = FALSE,
  create_table = FALSE, convert_logical_to_integer = FALSE,
  transaction = FALSE, verbose = TRUE)
```

**Arguments**

d	Dataset to be uploaded to the dataset. The object must inherit from data.frame.
table_name	Name of the database destination table.
dsn_name	Name of the locally-defined DSN passed to <code>RODBC::odbcConnect()</code> .
clear_table	If TRUE, calls <code>RODBC::sqlClear()</code> before writing to the table.
create_table	If the table structure has not yet been defined in the database, it will be created if create_table is TRUE.
convert_logical_to_integer	Convert all logical columns to integer. This helps the database store the values as bits.
transaction	Should the clear and upload steps be wrapped in a rollback transaction?
verbose	Write a message about the status of a successful upload.

**Details**

If transaction is TRUE and the upload fails, the table is rolled back to the state before function was called. This includes rolling back the (optional) clearing of records, and uploading the new records. Decide if it's more robust to rollback to the previous state, or if it's better to leave the table in the incomplete state. The latter is helpful diagnosing which record caused the write to fail; look at the last successful record contained in the database

**Examples**

```
## Not run:
requireNamespace("OuhscMunge")

OuhscMunge::upload_sqls_rodbs(
  d                = ds_client,          # Some data.frame that exists in RAM
  table_name       = "tbl_client",
  dsn_name         = "miechv_eval",
  create_table     = FALSE,
  clear_table      = TRUE,
  transaction      = TRUE,
  verbose          = TRUE,
  convert_logical_to_integer = TRUE
)

## End(Not run)
```



---

verify_data_frame	<i>Object inherits from <code>Rhrefbase::data.frame()</code><code>data.frame</code>.</i>
-------------------	--

---

### Description

Check that the object inherits from `data.frame`. If not, throw an error.

This helps check database-reading functions (e.g., `RODBC::sqlQuery()`) that return a `data.frame` if successful, and a character vector is unsuccessful.

A minimum row count can be used to check that a trivially small number of rows was not returned. If `minimum_row_count` is set to 0, the function is similar to `checkmate::assert_class(d, "data.frame")`, but with a more specific error message.

### Usage

```
verify_data_frame(d, minimum_row_count = 10L)
```

### Arguments

<code>d</code>	The object to verify. Required.
<code>minimum_row_count</code>	The <code>data.frame</code> should have at least this many rows. Defaults to 10. The datatype does not have to be an integer, but should be safely convertible to an integer.

### Author(s)

Will Beasley

### See Also

`checkmate::assert_class()`

### Examples

```
verify_data_frame(datasets::OrchardSprays, 20)
verify_data_frame(datasets::iris, 4)
```

---

verify_value_headstart	<i>Generates code to verify a <code>Rhrefbase::data.frame()</code><code>data.frame</code>.</i>
------------------------	--

---

### Description

Inspects properties of a `data.frame` and prints code to the console that a developer can use to start to check the properties of a dataset, such as the names and types of each column.

### Usage

```
verify_value_headstart(d)
```

**Arguments**

`d` The data.frame to verify. Required.

**Author(s)**

Will Beasley

**See Also**

[checkmate](#)

**Examples**

```
library(magrittr)
verify_value_headstart(datasets::OrchardSprays)
verify_value_headstart(datasets::iris)
verify_value_headstart(datasets::BOD)
verify_value_headstart(dplyr::band_members)
storms_2 <- dplyr::storms %>%
  dplyr::mutate(
    storm_date = as.Date(ISOdate(year, month, day))
  )
verify_value_headstart(storms_2)
```

# Index

`assert`, [2](#)  
`assert_non_na(assert)`, [2](#)  
`assert_non_na_and_unique(assert)`, [2](#)  
  
`base::class()`, [2](#)  
`base::iconv()`, [5](#)  
`base::numeric_version()`, [9](#)  
  
`checkmate`, [18](#)  
`clump_date`, [3](#)  
`clump_month_date(clump_date)`, [3](#)  
`clump_week_date(clump_date)`, [3](#)  
`column_class_headstart`  
    (`headstart_utilities`), [6](#)  
`column_rename_headstart`  
    (`headstart_utilities`), [6](#)  
`column_value_headstart`  
    (`headstart_utilities`), [6](#)  
  
`deterge`, [4](#)  
`deterge_to_ascii(deterge)`, [4](#)  
`deterge_to_double(deterge)`, [4](#)  
`deterge_to_integer(deterge)`, [4](#)  
  
`execute_sql_file`, [5](#)  
  
`first_nonmissing`, [6](#)  
  
`headstart_utilities`, [6](#)  
  
`lettercase::str_snake_case()`, [14](#)  
`lubridate::day()`, [3](#)  
`lubridate::wday()`, [3](#)  
  
`match_statistics`, [7](#)  
  
`open_dsn_channel_sqls`, [8](#)  
`OuhscMunge`, [9](#)  
`OuhscMunge-package (OuhscMunge)`, [9](#)  
  
`replace_nas_with_explicit`, [10](#)  
`replace_with_nas`, [11](#)  
`retrieve_key_value`, [12](#)  
`RODBC::odbcConnect()`, [12](#)  
`RODBC::sqlQuery()`, [17](#)  
  
`snake_case`, [13](#)  
`snakecase::to_snake_case()`, [14](#)  
  
`trim`, [14](#)  
`trim_date(trim)`, [14](#)  
`trim_integer(trim)`, [14](#)  
`trim_numeric(trim)`, [14](#)  
  
`update_packages_addin`, [15](#)  
`upload_sqls_rodbs`, [16](#)  
  
`verify_data_frame`, [17](#)  
`verify_value_headstart`, [17](#)