Package: TidyPanel (via r-universe)

November 2, 2026

Title Universal Messy Panel Data Cleaner

Version 0.2.133

Author Tony Lu [aut, cre]

Maintainer Tony Lu <xulunt123@gmail.com>

Description A robust toolkit designed to standardize and clean complex tabular data from commercial enterprise systems, healthcare records, logistics software, and HR databases. Now natively supports Excel files, flat-file databases (CSV/TSV/RDB), gzip-compressed CSV/TSV/TXT/RDB files, and raw data.frame inputs. Features include intelligent regex parsing for domain-specific noise (currencies, percentages), horizontal side-by-side panel splitting, automated metadata key-value extraction from decoy rows, gap-based block clustering, and messy table resolution. Methods draw on tidy data principles described in Wickham (2014) <doi:10.18637/jss.v059.i10> and the 'readxl' parsing infrastructure described in Wickham & Bryan (2023) <https://readxl.tidyverse.org>. A local JSON command-line interface supports reproducible table-cleaning jobs and bounded agent handoffs.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports dplyr, stringr, stringi, readxl, jsonlite (>= 1.8.9)

Suggests knitr, rmarkdown

VignetteBuilder knitr

Config/TidyPanel/Edition public

Config/pak/sysreqs libicu-dev

Repository https://tonyisfool.r-universe.dev

Date/Publication 2026-11-02 03:57:23 UTC

RemoteUrl https://github.com/tonyisfool/tidypanel

RemoteRef HEAD

RemoteSha 6adce22b7301903656855f4be80e45c67b1ff2e0

Contents

clean_variable_names 2
detect_panel_structure 3
infer_data_types 4
normalize_units 5
read_messy_panel 6
tidypanel_cli 8
validate_panel 9

Index **11**

`clean_variable_names` *Standardize and Clean Variable Names*

Description

`clean_variable_names()` standardizes column names in a messy data frame. It converts all names to snake_case, normalizes Unicode numeric symbols, keeps percent, common parenthetical time units, categorical suffixes, and fiscal-year headers explicit, and strips other special characters (except _), translates Excel serial dates (e.g., `44197`) into ISO date strings (`2021-01-01`), and maps common financial/academic synonyms (e.g., `gvkey`, `permno`, `cusip`) to standard names (`id`, `ticker`). Explicit identifier suffixes are protected from approximate financial mappings. Qualified population income measures retain their meaning rather than becoming profit.

Usage

```
clean_variable_names(data)
```

Arguments

`data`	A `data.frame`. The data frame with messy column names.

Value

A `data.frame` with the same data but standardized column names.

Examples

```
# Toy example: standardize column names in a data frame
df <- data.frame(
  `Total Revenue ($)` = 1,
  `PERMNO` = 3,
  `My Custom Column!` = 4,
  check.names = FALSE
)
clean_df <- clean_variable_names(df)
colnames(clean_df)
# Returns: c("revenue", "id", "my_custom_column")

# Excel serial dates are also handled
df2 <- data.frame(`44197` = 2, check.names = FALSE)
colnames(clean_variable_names(df2))
# Returns: "2021-01-01"
```

---

detect_panel_structure

Diagnose the Structure of a Messy Excel Panel File

Description

`detect_panel_structure()` performs a static, non-destructive analysis of a raw Excel file and returns a structured report describing its "messiness". It detects decoy metadata rows, multi-level headers, temporal wide columns, aggregation rows, and phantom columns. It also generates a recommended `read_messy_panel()` call based on the findings.

Usage

```
detect_panel_structure(path, sheet = 1, verbose = TRUE)
```

Arguments

`path`	A character string. Path to the `.xlsx` or `.xls` file.
`sheet`	A character string or integer. The sheet to analyse. Defaults to `1`.
`verbose`	Logical. If `TRUE`, prints a human-readable summary to the console. Default is `TRUE`.

Value

A named list with the following elements:

`n_rows` Total rows in the raw sheet.

`n_cols` Total columns in the raw sheet.

`estimated_decoy_rows` Number of estimated metadata/noise rows at the top.

`multi_level_header` Logical. Whether multi-level (merged) headers are detected.

`has_temporal_cols` Logical. Whether wide-format temporal columns (years, quarters) are detected.

`n_aggregation_rows` Number of suspected Total/Sum aggregation rows found.

`n_phantom_cols` Number of fully-empty ghost columns.

`n_data_blocks` Total number of disjoint data panels found in the sheet.

`recommended_call` A character string with a suggested `read_messy_panel()` call.

Examples

```
example_path <- system.file(
  "extdata", "demo_03_subtotals.csv", package = "TidyPanel", mustWork = TRUE
)
report <- detect_panel_structure(example_path, verbose = FALSE)
str(report)
```

`infer_data_types` *Smart Type Coercion & NA Recognition*

Description

`infer_data_types()` scans character columns in a data frame, identifies common financial placeholders for missing data (e.g., "-", "N/A", "n.m."), safely replaces them with `NA`, retains identifier-like columns such as phone numbers as text, and then infers `numeric`, `Date` or `logical` values from the remaining columns. This is heuristic coercion, not schema validation or a guarantee of literal text preservation. Categorical labels matching boolean tokens can become logical values; nonmatching values can become missing when a threshold is met. A stricter threshold does not preserve labels that match an inferred type. Use recognized identifier/code field names for literal codes, retain the source, and review output types and missing positions before analysis. Compact mixed digit/letter fields with leading-zero evidence retain text. Decimal and exponent numeric forms keep their existing conversion behavior. Existing compact-month calendar handling keeps its prior precedence. Exact `ident` fields and `_ident` suffixes retain text. The exact normalized `iso_country` alias needs two distinct other uppercase two-letter observations to protect literal `NA`; labels and contaminated profiles keep baseline behavior.

Usage

```
infer_data_types(
  data,
  na_strings = c("-", "N/A", "n/a", "n.m.", "n.m", "NA", "null", "NULL", "."),
  num_threshold = 0.95
)
```

Arguments

data	A data.frame.
na_strings	A character vector of strings to be interpreted as NA. Literal NA is retained in explicit two-letter ISO code fields when observed values match the code schema.
num_threshold	Numeric between 0 and 1. The minimum matching fraction for the existing numeric, date and logical inference paths. Default is 0.95. Nonmatching observations can become missing; this is not a literal-text lock.

Value

A data.frame with inferred data types.

Examples

```
# Clean financial placeholders and coerce to numeric
df <- data.frame(val = c("1.5", "-", "2.0", "N/A"), stringsAsFactors = FALSE)
df_clean <- infer_data_types(df)
df_clean$val  # numeric: c(1.5, NA, 2.0, NA)
is.numeric(df_clean$val)  # TRUE
```

---

normalize_units *Normalize Numeric Columns Based on Header Unit Declarations*

Description

normalize_units() scans the column names of a data frame for financial/scientific unit declarations (e.g., "Revenue (in millions)", "Assets ($k)", "Employees ('000)"). It automatically multiplies the numeric values in the corresponding columns by the detected multiplier (1,000, 1,000,000, etc.) and optionally strips the unit declaration from the column name.

Usage

```
normalize_units(data, strip_units = TRUE)
```

Arguments

data	A data.frame. The data frame to be normalized.
strip_units	Logical. If TRUE, removes the unit declarations from the column names. Default is TRUE.

Value

A data.frame with the normalized data and updated column names.

Examples

```
# Scale columns declared in millions and thousands
df <- data.frame(
  `Revenue ($M)` = c(1.5, 2.0),
  `Cost (in thousands)` = c(500, 600),
  check.names = FALSE
)
result <- normalize_units(df)
result$Revenue  # c(1500000, 2000000)
result$Cost     # c(500000, 600000)
```

`read_messy_panel` *Robust Parsing and Extraction of Messy Excel Panel Data*

Description

`read_messy_panel()` is an industrial-grade parser designed to extract clean, standardized data frames from heavily malformed, human-readable Excel reports (e.g., financial statements, ERP exports). It automatically bypasses decoy rows, stitches N-dimensional hierarchical headers, extracts structural indentation hierarchies (parent-child relationships), reshapes series-oriented workbooks with metadata rows and a date axis, reads RDB plus gzip-compressed CSV, TSV, TXT, and RDB files, decodes UTF-16 BOM text plus CP932/Shift-JIS, CP949, Big5, Windows-1250, and Windows-1251 monthly text tables, rejects HTML responses disguised as CSV files, retains stable headerless daily records, detects official whitespace-delimited text tables, amputates embedded subtotals, and standardizes financial/scientific numbers. Stable categorical labels in headerless numeric tables are retained. Supply numeric report observation sections as plain tables rather than report metadata. Commented delimited headers retain labels and quoted field boundaries. Explicit tab-delimited records take precedence over whitespace-table heuristics, retaining labels that contain spaces. Explicit named `field,value` tables retain all unique records and literal text, including hash characters and blank trailing values. Fields named exactly `site` retain literal identifiers; related measurement names such as `site_count` still receive ordinary numeric cleaning. Exact `ident` fields and `_ident` suffixes retain text and missing positions. Qualified `iso_country` fields retain literal `NA` with at least two distinct other uppercase two-letter codes. Coded reference catalogs require a prepared table. Sampled averages with time and integer observation-count fields are retained. Supply delimited sources with metadata preambles as plain observation tables. Headerless whitespace records with quoted labels require CSV or a named data frame. Scientific unit descriptor rows require a separately prepared observation table. Slash-dated series exports require a separately prepared named observation table. Sectioned control-record reports require a separately prepared named observation table. Sectioned observation exports require a separately prepared named table too. Prefaced measurement exports require a separately prepared named table. Prefaced monthly series exports also require a prepared named table. Qualified total-uncertainty measurements with finite original numeric evidence are retained; ordinary subtotal columns remain removable.

Usage

```
read_messy_panel(
```

```
  file_path,
  sheet = NULL,
 na_strings = c("", ":", "NA", "#N/A", "NULL", "null", "S", "D", "ND", "N/A", "?", "*", "**",
  "***", ".", "x", "c", "s", "z", "#VALUE!", "#REF!", "#DIV/0!", "#NUM!", "#NAME?",
  "none", "NR", "--", "---", "-999", "n.a.", "N.A.", "n/a", "Not Applicable", "\\N", "\u2026", "NaN", "
  clean_vars = TRUE,
  auto_pivot = FALSE,
  return_audit = FALSE,
  extract_all_blocks = FALSE,
  archive_member = NULL
)
```

Arguments

file_path	Character string. A path to an Excel, CSV, TSV, TXT, RDB, ZIP, or gzip-compressed CSV, TSV, TXT, or RDB file.
sheet	Optional sheet name or index. If NULL (the default), it selects the largest usable data panel across all sheets. If "ALL", parses and merges all sheets.
na_strings	Character vector. Strings to interpret as missing values. Supports complex missing-value lexicons. NaN is recognized in numeric contexts, not pure textual categories. Literal NA is retained in explicit two-letter ISO code fields when observed values match the code schema.
clean_vars	Logical. If TRUE (default), standardizes variable names to snake_case using clean_variable_names().
auto_pivot	Logical. If TRUE, attempts to reshape wide temporal columns (e.g., FY2021, Q1_2022) into a long format (time_period, value).
return_audit	Logical. If TRUE, returns a list containing $data (the cleaned data frame) and $audit (a detailed log of all algorithmic modifications made).
extract_all_blocks	Logical. If TRUE, extracts all disjoint data tables on a sheet as a list of data frames. Default is FALSE (extracts only the largest block).
archive_member	Optional member name for a ZIP archive containing multiple tabular files. ZIP archives with exactly one supported tabular member are read automatically.

Details

English day-month-year dates with hyphenated three-letter months are converted only when all observed values form valid calendar dates, independently of the time locale. Explicit identifier fields retain their text values. Missing explicit calendar values are not filled from neighboring records. Sparse country-year records and qualified physical total-energy measurements are retained rather than discarded as incomplete records or subtotal columns. Lowercase na becomes missing in numeric contexts, not pure text categories or recognized identifier fields. A custom na_strings can exclude this marker.

Value

If return_audit = FALSE, a cleaned and standardized data.frame. If return_audit = TRUE, a named list containing:

data	The cleaned data.frame.
audit	A data.frame detailing exactly what transformations, truncations, or imputations were applied.

Examples

```
example_path <- system.file(
  "extdata", "demo_01_numbers.csv", package = "TidyPanel", mustWork = TRUE
)
result <- read_messy_panel(example_path)
head(result)
```

---

tidypanel_cli *Run the Local JSON Cleaning Interface*

Description

A local adapter for the public table cleaner. It reads one bounded UTF-8 JSON request and writes one JSON response. It does not evaluate request text, start a service or upload data.

Usage

```
tidypanel_cli(args = commandArgs(trailingOnly = TRUE), protected_roots = character())
```

Arguments

args	Character arguments: describe, or clean --request PATH. Use - for standard input. clean alone also reads standard input.
protected_roots	Additional existing directories where output is forbidden. The installed package directory is always protected. The launcher adds its own package directory.

Details

Run describe for the JSON request schema. Select one local file and a new output directory with an existing parent. Explicitly supply na_strings from the source declaration. Only public reader options are exposed. Preview values are opt-in and may contain sensitive data.

The description also includes four fictional example request templates with installed input paths and explicit options. It does not read caller tables or create output directories. Choose a new output directory before reuse. Installed paths may identify a caller's local environment; keep them local unless sharing is authorized.

The adapter writes full cleaned R objects, a convenience CSV, a tabular audit and column metadata. CSV does not preserve every R type or missing-value distinction. The RDS file is for compatible trusted R use. A failed write can leave a partial new output directory; it is not deleted or reused. The audit is not a complete cell-by-cell history. This interface inherits the caller's filesystem permissions and is not a sandbox.

Use the bundled `exec/tidypanel.R` launcher with `Rscript --vanilla`. It requires the matching installed public source edition and exits with the returned status. Cleaner output other than the JSON response is directed to standard error. Warnings are recorded in the response, with bounded messages.

Value

Invisibly, an integer exit status: 0 for success, 2 for invalid requests, 3 for an edition mismatch, 4 for cleaning failures, or 5 for output failures. The function does not terminate R.

Examples

```
## Not run:
tidypanel_cli("describe")

## End(Not run)
```

---

validate_panel *Validate Data Quality of a Cleaned Panel Data Frame*

Description

`validate_panel()` performs a post-cleaning quality audit on a data frame. It checks for NA rates, outliers (via IQR), duplicate rows, sparse columns, columns that should be numeric but remain character, and optional time-series gap detection. Returns a structured report and optionally prints a summary.

Usage

```
validate_panel(
  data,
  time_col = NULL,
  na_warn_threshold = 0.5,
  sparse_warn_threshold = 0.1,
  verbose = TRUE
)
```

Arguments

`data`	A `data.frame`. The cleaned panel data to validate.
`time_col`	A character string. Optional. Name of the time/date column to check for temporal continuity. Default is `NULL` (skip time check).
`na_warn_threshold`	Numeric between 0-1. Columns with NA rate above this are flagged as high-risk. Default is `0.5`.
`sparse_warn_threshold`	Numeric between 0-1. Columns with valid value rate below this are flagged as sparse. Default is `0.1`.
`verbose`	Logical. If `TRUE`, prints a human-readable report. Default is `TRUE`.

Value

A named list with quality metrics per column and summary flags.

Examples

```
# Toy example: validate a simple data frame
df <- data.frame(
  id   = c(1, 2, 2, 4),
  val  = c(100, 200, 200, NA),
  flag = c("1.5", "2.0", "2.0", "N/A")
)
report <- validate_panel(df, verbose = FALSE)
report$n_duplicates  # 1 duplicate row
report$high_na_cols  # columns with > 50% NA
```

Index

clean_variable_names, 2

detect_panel_structure, 3

infer_data_types, 4

normalize_units, 5

read_messy_panel, 6

tidypanel_cli, 8

validate_panel, 9