

R package risk assessment: Balancing programmatic and human inputs

Arianna D. Cascone, PhD, Atorus Research Inc.
Chelsea N. Dickens, Atorus Research Inc.

Outline

- Introduction
- Assessing Risk
- 5 Risk Considerations
 - Purpose/Package Type
 - Package Maintenance
 - Community Use
 - Documentation
 - Testing
- Conclusions & Key Takeaways

Introduction

Proprietary software

Open-source software



Assessing Risk

- Quantifiable Elements

- Objective

- Examples:

- Code coverage
- Vignettes/No vignettes



- Qualitative Elements

- More subjective

- Examples:

- Substance of unit tests
- Quality of references



Assessing Risk - Programmatic

- Retrieve information about a package using code
 - Examples:
 - Test code coverage
 - Website
 - Vignettes
 - NEWS file
 - Fast, but may miss some details



```
Console Terminal x Workbench Jobs x
R 4.4.1 ~/
> assess_has_website(pkg_ref("tern"))
[1] "https://insightsengineering.github.io/tern/"
[2] "https://github.com/insightsengineering/tern/"
> assess_has_vignettes(pkg_ref("tern"))
[1] 5
```

Assessing Risk - Manual

- Slow and tedious
- Inconsistent
- May catch quality concerns better than programmatic assessment

tern: Create Common TLGs Used in Clinical Trials

Table, Listings, and Graphs (TLG) library for common outputs used in clinical trials.

Version: 0.9.7
Depends: R (≥ 3.6), [rtables](#) (≥ 0.6.11)
Imports: [broom](#) (≥ 0.5.4), [car](#) (≥ 3.0-13), [checkmate](#) (≥ 2.1.0), [cowplot](#) (≥ 1.0.0), [dplyr](#) (≥ 1.0.0), [emmeans](#) (≥ 1.10.4), [forcats](#) (≥ 1.0.0), [formatters](#) (≥ 0.5.10), [ggplot2](#) (≥ 3.5.0), [grid](#), [gridExtra](#) (≥ 2.0.0), [gtable](#) (≥ 0.3.0), [labeling](#), [lifecycle](#) (≥ 0.2.0), [magrittr](#) (≥ 1.5), [MASS](#) (≥ 7.3-60), [methods](#), [nestcolor](#) (≥ 0.1.1), [Rdpack](#) (≥ 2.4), [rlang](#) (≥ 1.1.0), [scales](#) (≥ 1.2.0), [stats](#), [survival](#) (≥ 3.6-4), [tibble](#) (≥ 2.0.0), [tidyr](#) (≥ 0.8.3), [utils](#)
Suggests: [knitr](#) (≥ 1.42), [lattice](#) (≥ 0.18-4), [lubridate](#) (≥ 1.7.9), [rmarkdown](#) (≥ 2.23), [stringr](#) (≥ 1.4.1), [svglite](#) (≥ 2.1.2), [testthat](#) (≥ 3.1.9), [withr](#) (≥ 2.0.0)
Published: 2025-01-17
DOI: [10.32614/CRAN.package.tern](#)
Author: Joe Zhu [aut, cre], Daniel Sabanés Bové [aut], Jana Stoilova [aut], Davide Garolini [aut], Emily de la Rua [aut], Abinaya Yogasekaram [aut], Heng Wang [aut], Francois Collin [aut], Adrian Waddell [aut], Pawel Rucki [aut], Chendi Liao [aut], Jennifer Li [aut], F. Hoffmann-La Roche AG [cph, fnd]
Maintainer: Joe Zhu <joe.zhu@roche.com>
BugReports: <https://github.com/insightsengineering/tern/issues>
License: [Apache License 2.0](#)
URL: <https://insightsengineering.github.io/tern/>, <https://github.com/insightsengineering/tern/>
NeedsCompilation: no
Language: en-US
Materials: [README NEWS](#)
CRAN checks: [tern results](#)

Documentation:

Reference manual: [tern.pdf](#)

Vignettes: [Missing Values in Tern](#) (source, R code)
[Tabulation](#) (source, R code)
[Introduction to tern](#) (source, R code)
[Formatting Functions](#) (source, R code)
[Understanding 'tern' functions](#) (source, R code)

5 Risk Considerations

Purpose

- What *type* of package is this?
 - How is the package intended to be used?
 - Does this package perform calculations?

ggstatsplot: 'ggplot2' Based Plots with Statistical Details

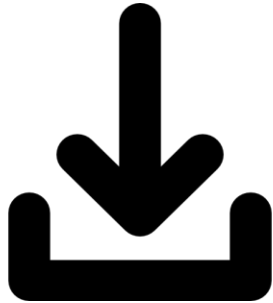
Extension of 'ggplot2', 'ggstatsplot' creates graphics with details from statistical tests included in the plots themselves. It provides an easier syntax to generate information-rich plots for statistical analysis of continuous (violin plots, scatterplots, histograms, dot plots, dot-and-whisker plots) or categorical (pie and bar charts) data. Currently, it supports the most common types of statistical approaches and tests: parametric, nonparametric, robust, and Bayesian versions of t-test/ANOVA, correlation analyses, contingency table analysis, meta-analysis, and regression analyses. References: Patil (2021) <[doi:10.21105/joss.03236](https://doi.org/10.21105/joss.03236)>.

Version: 0.13.0
Depends: R (≥ 4.3.0)
Imports: [correlation](#) (≥ 0.8.6), [datawizard](#) (≥ 0.13.0), [dplyr](#) (≥ 1.1.4), [ggcorrplot](#) (≥ 0.1.4.1), [ggplot2](#) (≥ 3.5.1), [ggrepel](#) (≥ 0.9.6), [ggside](#) (≥ 0.3.1), [ggsignif](#) (≥ 0.6.4), [glue](#) (≥ 1.8.0), [insight](#) (≥ 1.0.0), [paletteer](#) (≥ 1.6.0), [parameters](#) (≥ 0.24.0), [patchwork](#) (≥ 1.3.0), [performance](#) (≥ 0.12.4), [purrr](#) (≥ 1.0.2), [rlang](#) (≥ 1.1.4), [statsExpressions](#) (≥ 1.6.2), [tidyr](#) (≥ 1.3.1), [utils](#)
Suggests: [afex](#), [BayesFactor](#) (≥ 0.9.12-4.7), [gapminder](#), [knitr](#), [lme4](#) (≥ 1.1-35.2), [MASS](#), [metaBMA](#), [metafor](#), [metaplan](#), [psych](#), [rmarkdown](#), [rstantools](#), [stats](#), [survival](#), [testthat](#) (≥ 3.2.1), [tibble](#), [vdiffr](#) (≥ 1.0.8), [withr](#), [WRS2](#)
Published: 2024-12-04
DOI: [10.32614/CRAN.package.ggstatsplot](https://doi.org/10.32614/CRAN.package.ggstatsplot)
Author: Indrajeet Patil  [cre, aut, cph], Chuck Powell  [ctb]
Maintainer: Indrajeet Patil <patilindrajeet.science@gmail.com>

Community Use

- Is the package widely used?

Number of downloads can be used as a proxy for community testing



bug

question

enhancement

documentation



```
assess_downloads_1yr()
```

Documentation

- *Presence* of documentation can be assessed programmatically



```
assess_has_vignettes()
```

```
assess_has_examples()
```

```
assess_has_website()
```

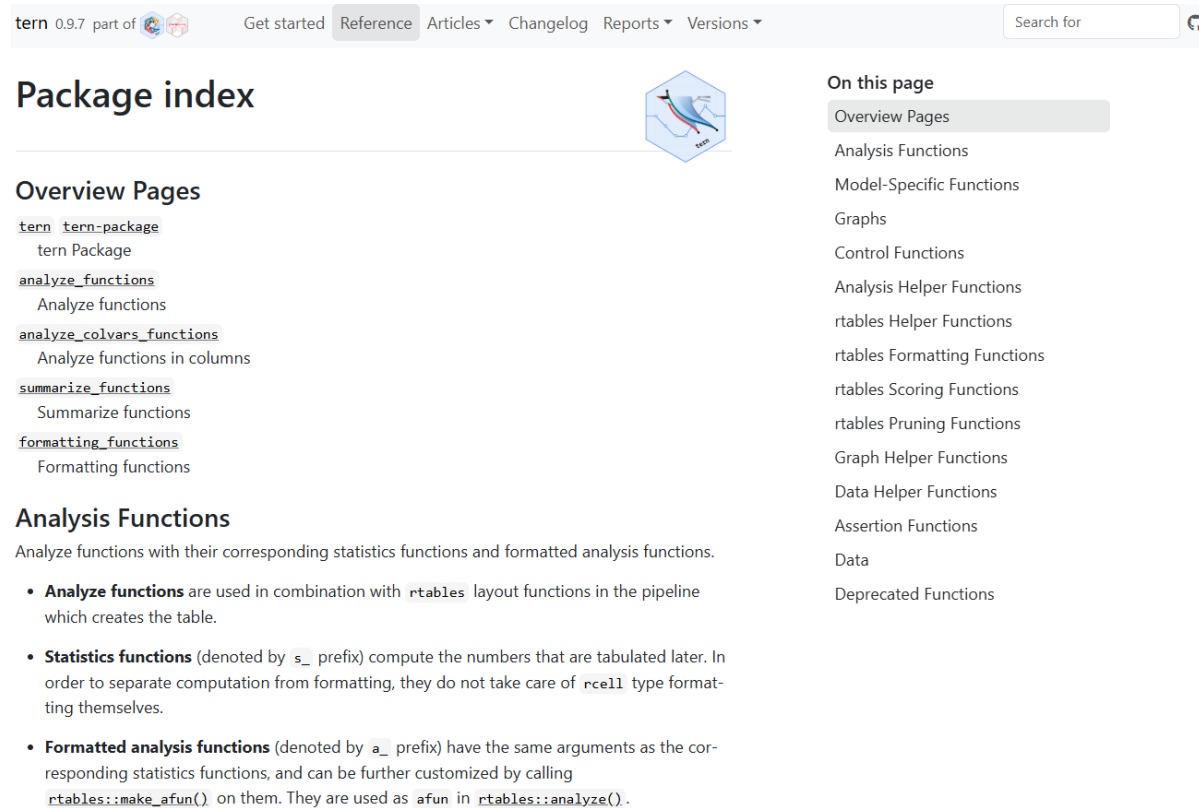
tern: Create Common TLGs Used in Clinical Trials

Table, Listings, and Graphs (TLG) library for common outputs used in clinical trials.

Version: 0.9.7
Depends: R (≥ 3.6), [rtables](#) (≥ 0.6.11)
Imports: [broom](#) (≥ 0.5.4), [car](#) (≥ 3.0-13), [checkmate](#) (≥ 2.1.0), [cowplot](#) (≥ 1.0.0), [dplyr](#) (≥ 1.0.0), [emmeans](#) (≥ 1.10.4), [forcats](#) (≥ 1.0.0), [formatters](#) (≥ 0.5.10), [ggplot2](#) (≥ 3.5.0), [grid](#), [gridExtra](#) (≥ 2.0.0), [gtable](#) (≥ 0.3.0), [labeling](#), [lifecycle](#) (≥ 0.2.0), [magrittr](#) (≥ 1.5), [MASS](#) (≥ 7.3-60), [methods](#), [nestcolor](#) (≥ 0.1.1), [Rdpack](#) (≥ 2.4), [rlang](#) (≥ 1.1.0), [scales](#) (≥ 1.2.0), [stats](#), [survival](#) (≥ 3.6-4), [tibble](#) (≥ 2.0.0), [tidyr](#) (≥ 0.8.3), [utils](#)
Suggests: [knitr](#) (≥ 1.42), [lattice](#) (≥ 0.18-4), [lubridate](#) (≥ 1.7.9), [rmarkdown](#) (≥ 2.23), [stringr](#) (≥ 1.4.1), [svglite](#) (≥ 2.1.2), [testthat](#) (≥ 3.1.9), [withr](#) (≥ 2.0.0)
Published: 2025-01-17
DOI: [10.32614/CRAN.package.tern](#)
Author: Joe Zhu [aut, cre], Daniel Sabanés Bové [aut], Jana Stoilova [aut], Davide Garolini [aut], Emily de la Rua [aut], Abinaya Yogasekaram [aut], Heng Wang [aut], Francois Collin [aut], Adrian Waddell [aut], Pawel Rucki [aut], Chendi Liao [aut], Jennifer Li [aut], F. Hoffmann-La Roche AG [cph, fnd]
Maintainer: Joe Zhu <joe.zhu@roche.com>
BugReports: <https://github.com/insightsengineering/tern/issues>
License: [Apache License 2.0](#)
URL: <https://insightsengineering.github.io/tern/>, <https://github.com/insightsengineering/tern/>
NeedsCompilation: no
Language: en-US
Materials: [README NEWS](#)
CRAN checks: [tern results](#)
Documentation:
Reference manual: [tern.pdf](#)
Vignettes: [Missing Values in Tern](#) (source, R code), [Tabulation](#) (source, R code), [Introduction to tern](#) (source, R code), [Formatting Functions](#) (source, R code), [Understanding 'tern' functions](#) (source, R code)

Documentation

- *Quality* of documentation can be assessed manually



tern 0.9.7 part of   Get started Reference Articles ▾ Changelog Reports ▾ Versions ▾ Search for

Package index



Overview Pages

- [tern](#) [tern-package](#)
tern Package
- [analyze_functions](#)
Analyze functions
- [analyze_colvars_functions](#)
Analyze functions in columns
- [summarize_functions](#)
Summarize functions
- [formatting_functions](#)
Formatting functions

Analysis Functions

Analyze functions with their corresponding statistics functions and formatted analysis functions.

- **Analyze functions** are used in combination with `rtables` layout functions in the pipeline which creates the table.
- **Statistics functions** (denoted by `s_` prefix) compute the numbers that are tabulated later. In order to separate computation from formatting, they do not take care of `rcell` type formatting themselves.
- **Formatted analysis functions** (denoted by `a_` prefix) have the same arguments as the corresponding statistics functions, and can be further customized by calling `rtables::make_afun()` on them. They are used as `afun` in `rtables::analyze()`.

On this page

- Overview Pages
- Analysis Functions
- Model-Specific Functions
- Graphs
- Control Functions
- Analysis Helper Functions
- rtables Helper Functions
- rtables Formatting Functions
- rtables Scoring Functions
- rtables Pruning Functions
- Graph Helper Functions
- Data Helper Functions
- Assertion Functions
- Data
- Deprecated Functions

Documentation

- *Quality* of documentation can be assessed manually

tern 0.9.7 part of 

[Get started](#) [Reference](#) [Articles](#) [Changelog](#) [Reports](#) [Versions](#)



Analysis Functions

Analyze functions with their corresponding statistics functions and formatted analysis functions.

- **Analyze functions** are used in combination with `rtables` layout functions in the pipeline which creates the table.
- **Statistics functions** (denoted by `s_` prefix) compute the numbers that are tabulated later. In order to separate computation from formatting, they do not take care of `rcell` type formatting themselves.
- **Formatted analysis functions** (denoted by `a_` prefix) have the same arguments as the corresponding statistics functions, and can be further customized by calling `rtables::make_afun()` on them. They are used as `afun` in `rtables::analyze()`.

`analyze_vars()` `s_summary()` `a_summary()` stable

Analyze variables

`analyze_vars_in_cols()` experimental

Analyze numeric variables in columns

`analyze_num_patients()` `summarize_num_patients()` `s_num_patients()`

`s_num_patients_content()` stable

Count number of patients

`analyze_patients_exposure_in_cols()` `summarize_patients_exposure_in_cols()`

`s_count_patients_sum_exposure()` `a_count_patients_sum_exposure()` stable

Count number of patients and sum exposure across all patients in columns

`compare_vars()` `s_compare()` stable

Compare variables between groups

`count_abnormal()` `s_count_abnormal()` `a_count_abnormal()` stable

Count patients with abnormal range values

`count_abnormal_by_baseline()` `s_count_abnormal_by_baseline()`

`a_count_abnormal_by_baseline()` stable

On this page

- Overview Pages
- Analysis Functions
- Model-Specific Functions
- Graphs
- Control Functions
- Analysis Helper Functions
- rtables Helper Functions
- rtables Formatting Functions
- rtables Scoring Functions
- rtables Pruning Functions
- Graph Helper Functions
- Data Helper Functions
- Assertion Functions
- Data
- Deprecated Functions

Documentation

- *Quality* of documentation can be assessed manually

tern 0.9.7 part of  

[Get started](#) [Reference](#) [Articles](#) [Changelog](#) [Reports](#) [Versions](#)



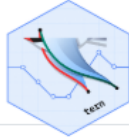
Analyze variables

Source: [R/analyze_variables.R](#)

lifecycle

stable

The `analyze_vars()` function creates a layout element to summarize one or more variables, using the S3 generic function `s_summary()` to calculate a list of summary statistics. A list of all available statistics for numeric variables can be viewed by running `get_stats("analyze_vars_numeric")` and for non-numeric variables by running `get_stats("analyze_vars_counts")`. Use the `.stats` parameter to specify the statistics to include in your output summary table. Use `compare_with_ref_group = TRUE` to compare the variable with reference groups.



On this page

- Usage
- Arguments
- Value
- Details
- Functions
- Note
- Examples



Documentation

- *Quality* of documentation can be assessed manually

```
library(dplyr)
library(tern)

## Fabricate dataset
dta_test <- data.frame(
  USUBJID = rep(1:6, each = 3),
  AVISIT = rep(paste0("V", 1:3), 6),
  ARM = rep(LETTERS[1:3], rep(6, 3)),
  AVAL = c(9:1, rep(NA, 9))
) %>%
mutate(ABLFLL = AVISIT == "V1") %>%
group_by(USUBJID) %>%
mutate(
  BLVAL = AVAL[ABLFLL],
  CHG = AVAL - BLVAL
) %>%
ungroup()
```

Classic use of `summarize_change()`.

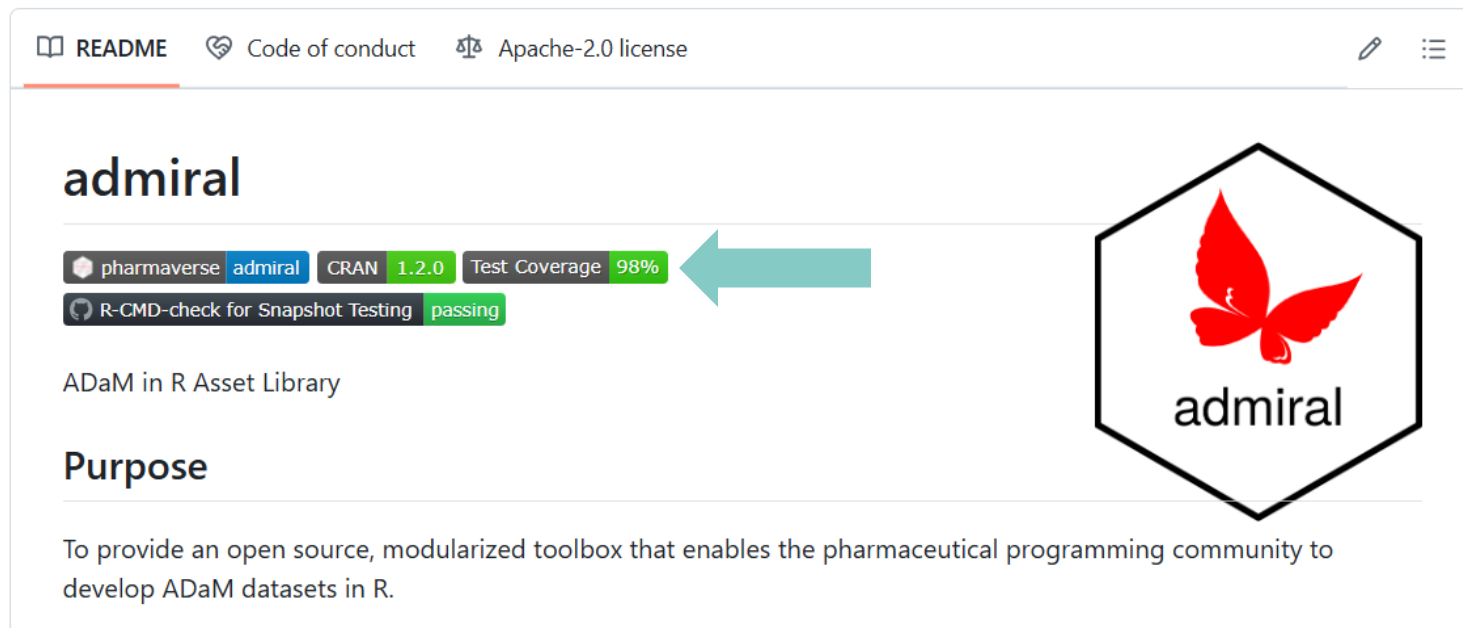
```
fix_layout <- basic_table() %>%
  split_cols_by("ARM") %>%
  split_rows_by("AVISIT")

# Dealing with NAs: na_rm = TRUE
fix_layout %>%
  summarize_change("CHG", variables = list(value = "AVAL", baseline_flag = "ABLFLL")) %>%
  build_table(dta_test) %>%
  print()
```

	A	B	C
#>			
#>			
#> V1			
#> n	2	1	0
#> Mean (SD)	7.50 (2.12)	3.00 (NA)	NA
#> Median	7.50	3.00	NA
#> Min - Max	6.00 - 9.00	3.00 - 3.00	NA
#> V2			
#> n	2	1	0
#> Mean (SD)	-1.00 (0.00)	-1.00 (NA)	NA
#> Median	-1.00	-1.00	NA
#> Min - Max	-1.00 - -1.00	-1.00 - -1.00	NA
#> V3			
#> n	2	1	0
#> Mean (SD)	-2.00 (0.00)	-2.00 (NA)	NA
#> Median	-2.00	-2.00	NA
#> Min - Max	-2.00 - -2.00	-2.00 - -2.00	NA

Testing

- Unit testing is crucial for ensuring functions produce expected, reliable outputs

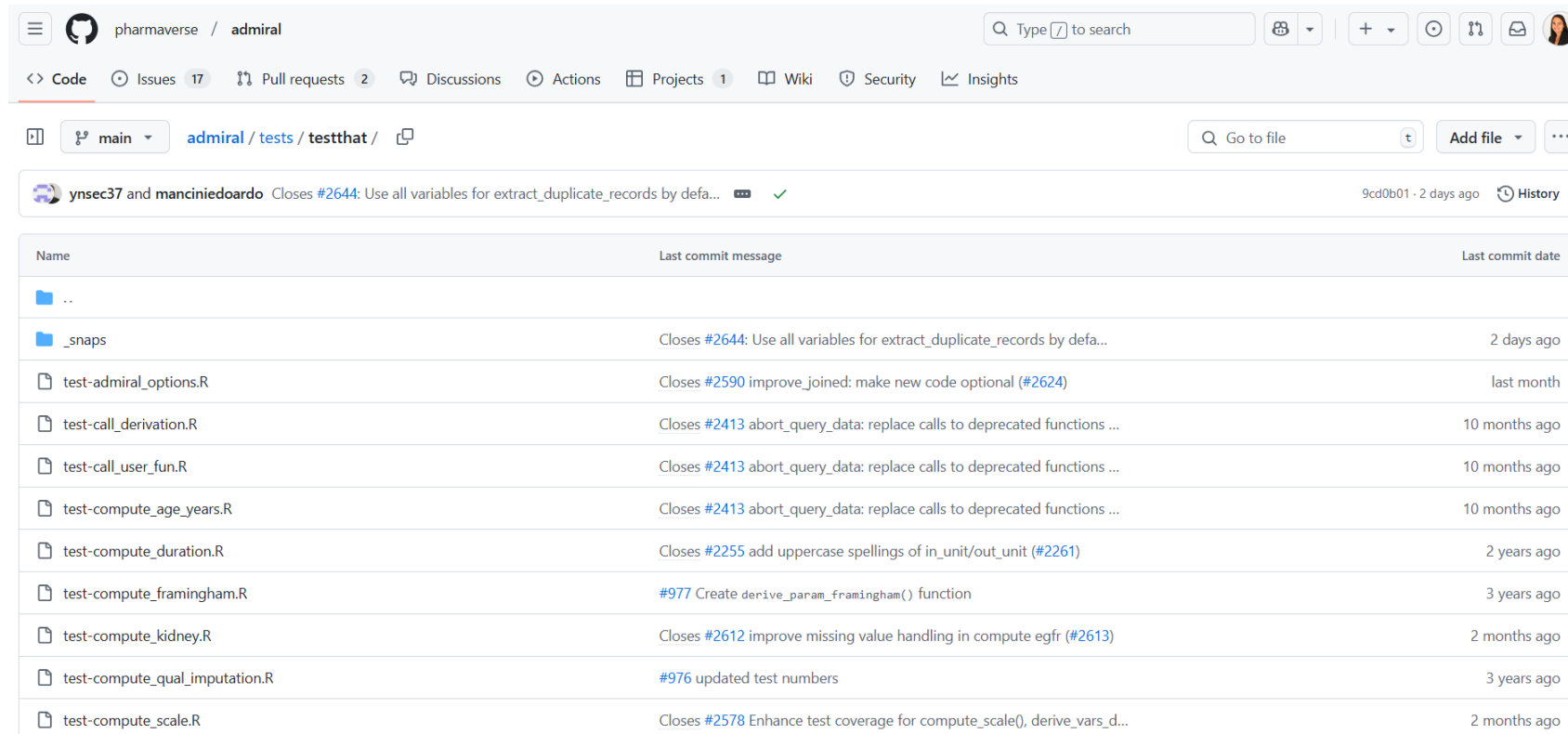


The screenshot shows the GitHub repository for the 'admiral' R package. At the top, there are links for 'README', 'Code of conduct', and 'Apache-2.0 license'. The repository name 'admiral' is prominently displayed. Below it, a row of badges shows the package is on 'pharmaverse', 'CRAN', version '1.2.0', and has 'Test Coverage 98%'. A teal arrow points to the 'Test Coverage 98%' badge. Below the badges, it says 'R-CMD-check for Snapshot Testing passing'. The description reads 'ADaM in R Asset Library'. The 'Purpose' section states: 'To provide an open source, modularized toolbox that enables the pharmaceutical programming community to develop ADaM datasets in R.' To the right of the repository information is a hexagonal logo with a red butterfly and the word 'admiral' below it.



Testing

- Assessing the quality of existing unit tests is important



The screenshot shows the GitHub interface for the 'pharmaverse / admiral' repository. The commit history table lists the following entries:

Name	Last commit message	Last commit date
..		
.._snaps	Closes #2644: Use all variables for extract_duplicate_records by defa...	2 days ago
test-admiral_options.R	Closes #2590 improve_joined: make new code optional (#2624)	last month
test-call_derivation.R	Closes #2413 abort_query_data: replace calls to deprecated functions ...	10 months ago
test-call_user_fun.R	Closes #2413 abort_query_data: replace calls to deprecated functions ...	10 months ago
test-compute_age_years.R	Closes #2413 abort_query_data: replace calls to deprecated functions ...	10 months ago
test-compute_duration.R	Closes #2255 add uppercase spellings of in_unit/out_unit (#2261)	2 years ago
test-compute_framingham.R	#977 Create derive_param_framingham() function	3 years ago
test-compute_kidney.R	Closes #2612 improve missing value handling in compute_egfr (#2613)	2 months ago
test-compute_qual_imputation.R	#976 updated test numbers	3 years ago
test-compute_scale.R	Closes #2578 Enhance test coverage for compute_scale(), derive_vars_d...	2 months ago

Testing

`derive_var_extreme_flag` unit testing in {admiral}

admiral 1.2.0

Get Started ▾

Reference

User Guides ▾

Find my function

Changelog

Search for

↺ ⚙️ ↻

Add a Variable Flagging the First or Last Observation Within Each By Group

Source: [R/derive_var_extreme_flag.R](#)

Add a variable flagging the first or last observation within each by group

Usage

```
derive_var_extreme_flag(  
  dataset,  
  by_vars,  
  order,  
  new_var,  
  mode,  
  true_value = "Y",  
  false_value = NA_character_,  
  flag_all = FALSE,  
  check_type = "warning"  
)
```



On this page

- Usage
- Arguments
- Value
- Details
- See also
- Examples

Testing

derive_var_extreme_flag unit testing in {admiral}

```
# derive_var_extreme_flag ----
## Test 1: first observation for each group is flagged ----
test_that("derive_var_extreme_flag Test 1: first observation for each group is flagged", {
  input <- tibble::tribble(
    ~USUBJID, ~AVISITN, ~AVAL,
    1, 1, 12,
    1, 3, 9,
    2, 2, 42,
    3, 3, 14,
    3, 3, 10
  )

  expected_output <- input %>% mutate(firstfl = c("Y", NA, "Y", "Y", NA))

  actual_output <- derive_var_extreme_flag(
    input,
    by_vars = exprs(USUBJID),
    order = exprs(AVISITN, desc(AVAL)),
    new_var = firstfl,
    mode = "first"
  )

  expect_dfs_equal(
    base = expected_output,
    compare = actual_output,
    keys = c("USUBJID", "AVISITN", "AVAL")
  )
})
```

```
## Test 2: last observation for each group is flagged ----
test_that("derive_var_extreme_flag Test 2: last observation for each group is flagged", {
  input <- tibble::tribble(
    ~USUBJID, ~AVISITN, ~AVAL,
    1, 1, 12,
    1, 3, 9,
    2, 2, 42,
    3, 3, 14,
    3, 3, 10
  )

  expected_output <- input %>% mutate(lastfl = c(NA, "Y", "Y", NA, "Y"))

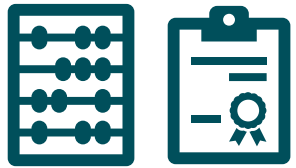
  actual_output <- derive_var_extreme_flag(
    input,
    by_vars = exprs(USUBJID),
    order = exprs(AVISITN, desc(AVAL)),
    new_var = lastfl,
    mode = "last"
  )

  expect_dfs_equal(
    base = expected_output,
    compare = actual_output,
    keys = c("USUBJID", "AVISITN", "AVAL")
  )
})
```

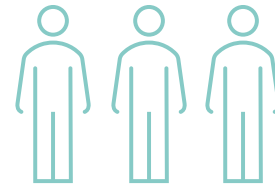
Conclusions & Key Takeaways



Assessing package risk systematically is crucial as the industry moves toward open-source software



Both quantifiable and qualitative aspects of package should be examined and considered when assessing risk



Questions?

Arianna D. Cascone, PhD

arianna.cascone@atorusresearch.com

Chelsea N. Dickens

chelsea.dickens@atorusresearch.com

