Package: pointblankops (via r-universe)

June 1, 2026

Title Extending Pointblank with Precision Failure Analysis

Version 0.1.0

Description Extension of Pointblank providing lightweight alternatives to pointblank agents for efficient failure detection and reporting.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports pointblank, dplyr, tibble, rlang

Suggests testthat (>= 3.0.0), knitr, rmarkdown, quarto, DBI, arrow, duckdb, RSQLite

VignetteBuilder quarto

URL https://petrbouchal.github.io/pointblankops/, https://github.com/petrbouchal/pointblankops

BugReports https://github.com/petrbouchal/pointblankops/issues

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libxml2-dev libssl-dev libnode-dev

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

Date/Publication 2025-08-27 19:21:54 UTC

RemoteUrl https://github.com/petrbouchal/pointblankops

RemoteRef HEAD

RemoteSha 7b8db8c5b3d2878aeb1e78c33af13a25095413bc

Contents

`create_operative` *Create operative for focused failure detection*

Description

Creates a lightweight operative object for specialized data validation intelligence gathering. Operatives are streamlined versions of pointblank agents designed for efficient failure detection without the overhead of full reporting capabilities.

Usage

```
create_operative(tbl, tbl_name = NULL, label = NULL)
```

Arguments

`tbl`	A data.frame, tibble, or database table (tbl_sql) to validate
`tbl_name`	Optional name for the table (if not provided, inferred from object)
`label`	Optional label for the operative

Value

A pointblank agent object configured as an operative with class `ptblank_operative`

Examples

```
## Not run:
# Create operative from data frame
data <- data.frame(id = 1:5, value = c(1, 2, NA, 4, 5))
operative <- create_operative(data) |>
  col_vals_not_null(columns = vars(value))

# Create operative from database table
con <- DBI::dbConnect(duckdb::duckdb(), ":memory:")
operative <- create_operative(tbl(con, "my_table")) |>
  col_vals_between(columns = vars(amount), left = 0, right = 1000)

## End(Not run)
```

debrief *Debrief operative and extract failure intelligence*

Description

Extracts failure information from an operative (or agent) that has been configured with validation steps. This is a memory-efficient alternative to full interrogation that focuses only on identifying and reporting validation failures.

Usage

```
debrief(
  operative,
  row_id_col,
  parquet_path = NULL,
  con = NULL,
  output_tbl = NULL,
  chunk_size = 1000
)
```

Arguments

operative	A pointblank agent or operative object with validation steps
row_id_col	Character vector of column names to use as row identifiers. These columns will be included in the output to identify failing rows.
parquet_path	Optional path to save failures as a parquet file. If provided, failures will be written to this file instead of returned as a tibble. Requires the 'arrow' package to be installed.
con	Optional database connection to save failures. If provided, failures will be inserted into a database table. Requires the 'DBI' package to be installed.
output_tbl	Optional table name for database output. If not provided and con is specified, the table name will be inferred as {source_table}_failures.
chunk_size	Number of rows to process at once for memory efficiency. Default is 1000.

Value

If no output path is specified, returns a tibble containing failure records with ID columns, test metadata, and failure details. If parquet_path or con is provided, returns NULL invisibly after writing the failures.

Examples

```
## Not run:
# Basic usage - return failures as tibble
operative <- create_operative(mtcars) |>
  col_vals_not_null(columns = vars(mpg)) |>
```

```
  col_vals_between(columns = vars(cyl), left = 4, right = 8)

failures <- debrief(operative, row_id_col = c("gear", "carb"))

# Save to parquet file (requires arrow package)
# install.packages("arrow")
debrief(operative, row_id_col = "gear", parquet_path = "failures.parquet")

# Save to database (requires DBI package)
# install.packages("DBI")
con <- DBI::dbConnect(duckdb::duckdb(), ":memory:")
debrief(operative, row_id_col = "gear", con = con, output_tbl = "car_failures")

## End(Not run)
```

Index