Package 'thoth'

February 26, 2025

Title Reproducible Analytics Framework with Data Version Control

Version 0.0.0.9000

Author Sebastian Rauschert [aut, cre]

Maintainer Sebastian Rauschert <seb.rauschert@gmail.com>

Description A comprehensive framework for setting up reproducible analytics projects with integrated version control for data using DVC (Data Version Control), containerization using Docker, dependency management using renv, and customizable reporting using Quarto. While DVC is recommended for full functionality, the package can operate without it installed by creating mock .dvc files. The package implements best practices for project organization, workflow management, and reproducible research.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

VignetteBuilder knitr

Depends R (>= 4.1.0)

Imports cli (>= 3.0.0), digest (>= 0.6.25), readr (>= 2.0.0), renv (>= 1.0.0), rlang (>= 1.0.0), rstudioapi (>= 0.13), usethis (>= 2.0.0), yaml (>= 2.3.0), rmarkdown (>= 2.10), tools (>= 4.1.0), magrittr (>= 2.0.0), janitor (>= 2.2.0), yardstick (>= 1.3.0), fs, glue

Suggests covr (>= 3.5.0), devtools (>= 2.4.0), dplyr (>= 1.0.0), knitr (>= 1.30), mockery (>= 0.4.3), pkgdown (>= 2.0.0), purrr (>= 1.0.0), testthat (>= 3.0.0), tibble (>= 3.0.0), withr (>= 2.4.0)

Config/testthat/edition 3

URL https://github.com/sebrauschert/thoth, https://sebrauschert.github.io/thoth/

BugReports https://github.com/sebrauschert/thoth/issues

SystemRequirements DVC (>= 2.0.0) (https://dvc.org) [recommended], Python (>= 3.7), Docker (>= 20.10.0) [optional]

Language en-US

LazyData true

NeedsCompilation no

Contents

`thoth-package` *thoth: Reproducible Analytics Framework with Data Version Control*

Description

A comprehensive framework for setting up reproducible analytics projects with integrated version control for data using 'DVC' (Data Version Control), containerization using 'Docker', dependency management using 'renv', and customizable reporting using 'Quarto'.

Key Features

- Project organization and structure
- Data version control with DVC
- Containerization with Docker
- Dependency management with renv
- Customizable reporting with Quarto

Main Functions

- `create_analytics_project`: Create a new analytics project
- `dvc_track`: Track files with DVC
- `write_csv_dvc`: Write and track CSV files
- `write_rds_dvc`: Write and track RDS files

Author(s)

Maintainer: Sebastian Rauschert `<seb.rauschert@gmail.com>`

See Also

Useful links:

- `https://github.com/sebrauschert/thoth`
- `https://sebrauschert.github.io/thoth/`
- Report bugs at `https://github.com/sebrauschert/thoth/issues`

`apply_template_to_report` *Apply Template to Report*

Description

Applies a custom template to a Quarto report

Usage

```
apply_template_to_report(report_path, template_name)
```

Arguments

`report_path`	Character. Path to the Quarto report
`template_name`	Character. Name of the template to apply

Value

Invisibly returns TRUE on success

Examples

```
## Not run:
apply_template_to_report("reports/analysis.qmd", "company_template")

## End(Not run)
```

`check_dvc` *Check DVC Installation*

Description

Check DVC Installation

Usage

```
check_dvc()
```

`check_git` *Check Git Installation*

Description

Check Git Installation

Usage

```
check_git()
```

`check_system_requirements` *Check System Requirements*

Description

Check System Requirements

Usage

```
check_system_requirements(use_dvc, use_docker)
```

Arguments

`use_dvc`	Logical. Whether DVC is required
`use_docker`	Logical. Whether Docker is required

`conf_mat` *Create a confusion matrix*

Description

Create a confusion matrix

Usage

```
conf_mat(data, truth, estimate, ...)
```

Arguments

`data`	A data frame containing the columns specified in `truth` and `estimate`.
`truth`	The column name containing the true values.
`estimate`	The column name containing the predicted values.
`...`	Additional arguments passed to yardstick::conf_mat.

Value

A confusion matrix.

Examples

```
## Not run:
library(dplyr)
data(mtcars)
# Create a binary outcome
mtcars <- mtcars %>%
  mutate(vs_factor = factor(vs))
# Fit a model
model <- glm(vs ~ mpg + cyl, data = mtcars, family = "binomial")
```

```
# Make predictions
preds <- predict(model, type = "response")
# Create prediction data frame
pred_data <- mtcars %>%
  mutate(pred = factor(ifelse(preds > 0.5, 1, 0)))
# Calculate confusion matrix
conf_mat(pred_data, truth = vs_factor, estimate = pred)

## End(Not run)
```

`create_analytics_project`

Create a New Analytics Project

Description

Sets up a new analytics project with standardized structure and configuration for reproducible analysis using DVC, Docker, renv, and Quarto.

Usage

```
create_analytics_project(
  path,
  use_dvc = TRUE,
  use_docker = TRUE,
  use_renv = TRUE,
  git_init = TRUE,
  open = rlang::is_interactive()
)
```

Arguments

`path`	Character. The path where the project should be created.
`use_dvc`	Logical. Whether to initialize DVC. Default is TRUE.
`use_docker`	Logical. Whether to set up Docker configuration. Default is TRUE.
`use_renv`	Logical. Whether to initialize renv. Default is TRUE.
`git_init`	Logical. Whether to initialize git repository. Default is TRUE.
`open`	Logical. Whether to open the new project in RStudio. Default is TRUE.

Value

Invisibly returns the path to the created project.

Examples

```
## Not run:
create_analytics_project("my_analysis")

## End(Not run)
```

`create_custom_css` *Create Custom CSS for Quarto Template*

Description

Create Custom CSS for Quarto Template

Usage

```
create_custom_css(
  primary_color = NULL,
  secondary_color = NULL,
  font_family = NULL
)
```

`create_quarto_template` *Create Custom Quarto Template*

Description

Creates a custom Quarto template with specified branding options

Usage

```
create_quarto_template(
  template_name,
  logo_path = NULL,
  primary_color = NULL,
  secondary_color = NULL,
  font_family = NULL,
  output_dir = "reports/templates"
)
```

Arguments

`template_name`	Character. Name of the template
`logo_path`	Character. Path to logo file (optional)
`primary_color`	Character. Primary brand color in hex format (optional)
`secondary_color`	Character. Secondary brand color in hex format (optional)
`font_family`	Character. Main font family to use (optional)
`output_dir`	Character. Directory to save the template (optional)

Value

Invisibly returns the path to the created template

Examples

```
## Not run:
create_quarto_template(
  template_name = "company_template",
  logo_path = "path/to/logo.png",
  primary_color = "#FF0000"
)

## End(Not run)
```

`create_template_yaml` *Create Template YAML Configuration*

Description

Create Template YAML Configuration

Usage

```
create_template_yaml(template_name, logo_path = NULL)
```

`decision_tracking` *Decision Tracking Functions*

Description

Functions for tracking and documenting human decisions in analyses

`dvc_add` *Track Files with DVC*

Description

Track Files with DVC

Usage

```
dvc_add(path, message = NULL, recursive = FALSE, git_add = TRUE)
```

Arguments

`path`	Character vector of file paths to track
`message`	Optional commit message for DVC
`recursive`	Logical. Whether to recursively add directories. Default is FALSE.
`git_add`	Logical. Whether to automatically add the .dvc files to git. Default is TRUE.

Value

Invisibly returns the tracked paths

`dvc_commit` *Commit Changes to DVC*

Description

Commit Changes to DVC

Usage

```
dvc_commit(path, message)
```

Arguments

`path`	Character vector of file paths to commit
`message`	Commit message

Value

Invisibly returns TRUE if successful

`dvc_pull` *Pull Data from DVC Remote*

Description

Pull Data from DVC Remote

Usage

```
dvc_pull(path = NULL, remote = NULL)
```

Arguments

`path`	Optional character vector of specific paths to pull
`remote`	Optional name of the remote to pull from

Value

Invisibly returns TRUE if successful

dvc_push *Push Data to DVC Remote*

Description

Push Data to DVC Remote

Usage

```
dvc_push(path = NULL, remote = NULL)
```

Arguments

`path`	Optional character vector of specific paths to push
`remote`	Optional name of the remote to push to

Value

Invisibly returns TRUE if successful

dvc_stage *Create a DVC Pipeline Stage*

Description

Create a DVC Pipeline Stage

Create a DVC Stage

Usage

```
dvc_stage(
  name,
  cmd,
  deps = NULL,
  outs = NULL,
  metrics = FALSE,
  plots = FALSE,
  params = NULL,
  always_changed = FALSE
)

dvc_stage(
  name,
  cmd,
  deps = NULL,
  outs = NULL,
  metrics = FALSE,
  plots = FALSE,
  params = NULL,
  always_changed = FALSE
)
```

Arguments

`name`	Stage name
`cmd`	Command to execute
`deps`	Dependencies
`outs`	Outputs
`metrics`	Logical or character vector indicating whether to track metrics
`plots`	Logical or character vector indicating whether to track plots
`params`	Named list of parameters
`always_changed`	Logical indicating whether the stage should always be re-run

Value

Invisibly returns TRUE if successful

Invisibly returns TRUE if successful

`dvc_track` *Track files with DVC after writing*

Description

This function adds DVC tracking to files that have been written using tidyverse write functions. It is designed to be used in a pipe chain after write operations.

Usage

```
dvc_track(path, message = NULL, push = FALSE)
```

Arguments

`path`	The path to the file that was written
`message`	An optional commit message for DVC
`push`	Logical. Whether to push changes to Git remote (default: FALSE)

Value

The input path (invisibly) to allow for further piping

Examples

```
## Not run:
data |>
  readr::write_csv("data/processed/mydata.csv") |>
  dvc_track("Updated processed data", push = TRUE)

## End(Not run)
```

`example_projects` *Example Analytics Project Data*

Description

A dataset containing example analytics project metrics for demonstration purposes. This dataset includes project characteristics and performance metrics.

Usage

```
example_projects
```

Format

A data frame with 100 rows and 6 variables:

project_id Unique identifier for each project

start_date Project start date

team_size Number of team members

uses_dvc Whether the project uses DVC (logical)

uses_docker Whether the project uses Docker (logical)

completion_rate Project completion rate (0-100)

Source

Generated for demonstration purposes

`export_decision_tree` *Export Decision Tree to Various Formats*

Description

Export Decision Tree to Various Formats

Usage

```
export_decision_tree(file_path, format = "md", output_path = NULL)
```

Arguments

`file_path`	Path to the decision tree YAML file
`format`	Output format ("html", "pdf", or "md")
`output_path`	Path where to save the output file

Value

Invisibly returns the path to the exported file

`generate_methods_section` *Generate Methods Section from Decision Tree*

Description

Generate Methods Section from Decision Tree

Usage

```
generate_methods_section(file_path, format = "markdown")
```

Arguments

`file_path`	Path to the decision tree YAML file
`format`	Output format ("markdown" or "text")

Value

Character string containing the methods section

`git_add` *Add Files to Git*

Description

Adds file contents to the index (staging area).

Usage

```
git_add(path, force = FALSE)
```

Arguments

`path`	Character vector of file paths to add
`force`	Logical. Whether to force add ignored files. Default is FALSE.

Value

Invisibly returns the added paths

Examples

```
## Not run:
git_add("analysis.R")
git_add(c("data/results.csv", "plots/figure1.png"))
git_add(".", force = FALSE)  # add all changes

## End(Not run)
```

`git_branch` *Create a New Git Branch*

Description

Creates a new branch and optionally switches to it.

Usage

```
git_branch(branch_name, checkout = TRUE)
```

Arguments

`branch_name`	Name of the new branch
`checkout`	Logical. Whether to checkout the new branch. Default is TRUE.

Value

Invisibly returns TRUE if successful

Examples

```
## Not run:
git_branch("feature/new-analysis")
git_branch("hotfix/bug-123", checkout = FALSE)

## End(Not run)
```

`git_branch_list` *List Git Branches*

Description

Shows a list of all branches in the repository.

Usage

```
git_branch_list(all = FALSE)
```

Arguments

`all`	Logical. Whether to show all branches (including remotes). Default is FALSE.

Value

Character vector of branch names

Examples

```
## Not run:
git_branch_list()
git_branch_list(all = TRUE)  # include remote branches

## End(Not run)
```

`git_checkout` *Checkout a Git Branch*

Description

Switches to a specified branch, optionally creating it if it doesn't exist.

Usage

```
git_checkout(branch_name, create = FALSE)
```

Arguments

`branch_name`	Name of the branch to checkout
`create`	Logical. Whether to create the branch if it doesn't exist. Default is FALSE.

Value

Invisibly returns TRUE if successful

Examples

```
## Not run:
git_checkout("main")
git_checkout("feature/new-analysis", create = TRUE)

## End(Not run)
```

`git_commit` *Commit Changes to Git*

Description

Records changes to the repository.

Usage

```
git_commit(message, all = FALSE)
```

Arguments

`message`	Commit message
`all`	Logical. Whether to automatically stage modified and deleted files. Default is FALSE.

Value

Invisibly returns TRUE if successful

Examples

```
## Not run:
git_commit("Add analysis script")
git_commit("Update results", all = TRUE)

## End(Not run)
```

---

git_log *Get Git Log*

Description

Shows the commit logs.

Usage

```
git_log(n = 10, oneline = TRUE)
```

Arguments

`n`	Number of commits to show. Default is 10.
`oneline`	Logical. Whether to show each commit on one line. Default is TRUE.

Value

Character vector containing log output

Examples

```
## Not run:
git_log()
git_log(n = 20, oneline = FALSE)  # detailed log

## End(Not run)
```

git_pull *Pull Changes from Git Remote*

Description

Fetches changes from a remote repository and integrates them into the current branch.

Usage

```
git_pull(remote = NULL, branch = NULL)
```

Arguments

`remote`	Name of the remote. Default is NULL (uses default remote).
`branch`	Name of the branch. Default is NULL (uses current branch).

Value

Invisibly returns TRUE if successful

Examples

```
## Not run:
git_pull()
git_pull("origin", "main")

## End(Not run)
```

git_push *Push Changes to Git Remote*

Description

Uploads local branch commits to a remote repository.

Usage

```
git_push(remote = NULL, branch = NULL)
```

Arguments

`remote`	Name of the remote. Default is NULL (uses default remote).
`branch`	Name of the branch. Default is NULL (uses current branch).

Value

Invisibly returns TRUE if successful

Examples

```
## Not run:
git_push()
git_push("origin", "feature/new-analysis")

## End(Not run)
```

---

git_status *Get Git Status*

Description

Shows the working tree status, indicating which files have been modified, added, deleted, or untracked.

Usage

```
git_status(short = TRUE)
```

Arguments

short	Logical. Whether to show status in short format. Default is TRUE.

Value

Character vector containing status output

Examples

```
## Not run:
git_status()
git_status(short = FALSE)  # detailed output

## End(Not run)
```

---

initialize_decision_tree *Initialize a Decision Tree*

Description

Initialize a Decision Tree

Usage

```
initialize_decision_tree(analysis_id, analyst, description, path = "decisions")
```

Arguments

`analysis_id`	Character string identifying the analysis
`analyst`	Character string with analyst name
`description`	Character string describing the analysis
`path`	Character string specifying where to save the decision tree

Value

Invisibly returns the path to the created decision tree file

`metrics` *Calculate model performance metrics*

Description

Calculate model performance metrics

Usage

```
metrics(data, truth, estimate, event_level = NULL, ...)
```

Arguments

`data`	A data frame containing the columns specified in `truth` and `estimate`.
`truth`	The column name containing the true values.
`estimate`	The column name containing the predicted values.
`event_level`	A character string indicating which level of the outcome is considered the "event".
`...`	Additional arguments passed to yardstick::metrics.

Value

A tibble with model performance metrics.

Examples

```
## Not run:
library(dplyr)
data(mtcars)
# Create a binary outcome
mtcars <- mtcars %>%
  mutate(vs_factor = factor(vs))
# Fit a model
model <- glm(vs ~ mpg + cyl, data = mtcars, family = "binomial")
# Make predictions
preds <- predict(model, type = "response")
# Create prediction data frame
pred_data <- mtcars %>%
  mutate(pred = factor(ifelse(preds > 0.5, 1, 0)))
# Calculate metrics
metrics(pred_data, truth = vs_factor, estimate = pred)

## End(Not run)
```

`record_decision` *Record a Decision*

Description

Record a Decision

Usage

```
record_decision(
  file_path,
  check,
  observation,
  decision,
  reasoning,
  evidence = NULL
)
```

Arguments

`file_path`	Path to the decision tree YAML file
`check`	Character string describing what was checked
`observation`	Character string describing what was observed
`decision`	Character string describing the decision made
`reasoning`	Character string explaining the reasoning
`evidence`	Character string pointing to supporting evidence (e.g., plot path)

Value

Invisibly returns the updated decision tree

`setup_docker` *Set up Docker Configuration*

Description

Set up Docker Configuration

Usage

```
setup_docker()
```

`setup_dvc_tracking` *Set up DVC Tracking*

Description

Set up DVC Tracking

Usage

```
setup_dvc_tracking()
```

`setup_quarto_template` *Set up Quarto Template*

Description

Set up Quarto Template

Usage

```
setup_quarto_template()
```

`version_control` *Version Control Functions*

Description

Functions for interacting with DVC and Git from R

`write_csv_dvc` *Write a CSV file and track it with DVC*

Description

Write a CSV file and track it with DVC

Usage

```
write_csv_dvc(
  x,
  path,
  message,
  stage_name = NULL,
  deps = NULL,
  params = NULL,
  metrics = FALSE,
  push = FALSE
)
```

Arguments

x	A data frame to write to CSV
path	Path to save the CSV file
message	Git commit message
stage_name	Optional DVC stage name
deps	Optional vector of dependency files
params	Optional list of parameters
metrics	Logical, whether to track as DVC metrics (default: FALSE)
push	Logical, whether to push changes to Git remote (default: FALSE)

Value

The input data frame (invisibly) to allow for further piping

Examples

```
## Not run:
# Simple tracking
data |> write_csv_dvc(
  "data/processed/results.csv",
  message = "Add processed results",
  push = TRUE
)

# As part of a pipeline
data |> write_csv_dvc(
  "data/processed/features.csv",
  message = "Add feature matrix",
  stage_name = "feature_engineering",
  deps = "data/raw/input.csv",
  params = list(n_components = 10),
  push = TRUE
)

## End(Not run)
```

---

write_gitignore *Write Default .gitignore File*

Description

Write Default .gitignore File

Usage

```
write_gitignore()
```

`write_rds_dvc` *Write RDS with DVC tracking*

Description

A wrapper around saveRDS that automatically tracks the output file with DVC and optionally creates a DVC pipeline stage.

Usage

```
write_rds_dvc(
  object,
  file,
  message = NULL,
  stage_name = NULL,
  deps = NULL,
  metrics = FALSE,
  plots = FALSE,
  params = NULL,
  push = FALSE,
  ...
)
```

Arguments

`object`	Object to save
`file`	Path to write to
`message`	Optional DVC commit message
`stage_name`	Optional name for the DVC stage. If provided, creates a pipeline stage.
`deps`	Character vector of dependency files (optional, for pipeline stages)
`metrics`	Logical. Whether to mark the output as a DVC metric
`plots`	Logical. Whether to mark the output as a DVC plot
`params`	Named list of parameters for the stage (optional)
`push`	Logical. Whether to push changes to Git remote (default: FALSE)
`...`	Additional arguments passed to saveRDS

Value

The input object (invisibly) to allow for further piping

Examples

```
## Not run:
# Simple tracking
model |> write_rds_dvc(
  "models/model.rds",
  message = "Updated model",
  push = TRUE
)
```

```
# As part of a pipeline
model |> write_rds_dvc(
  "models/rf_model.rds",
  message = "Save trained random forest model",
  stage_name = "train_model",
  deps = c("data/processed/training.csv", "R/train_model.R"),
  params = list(ntree = 500),
  push = TRUE
)

## End(Not run)
```

`write_readme` *Write Project README*

Description

Write Project README

Usage

```
write_readme(project_name)
```

`%>%` *Pipe operator*

Description

See `magrittr::%>%` for details.

Usage

```
lhs %>% rhs
```

Value

The result of applying rhs to lhs

Index