

valtools :: CHEAT SHEET



Validation Framework

{valtools} implements a convention for organizing information referenced during validation of R packages to generate reports. This framework is consistent for all five (5) validation modes.

| Validation Modes | |
|-----------------------|------------------|
| separate from package | - validation/ |
| | - requirements/ |
| from source code | - test_cases/ |
| | - test_code/ |
| at installation | - validation.yml |
| | - change_log.md |
| after installation | |
| for distribution | - report.Rmd |

Getting started

vt_create_package() Start an R package with validation enabled.
analogue to `usethis::create_package()`

vt_validation() Initiate just the validation framework. Use when adding to an existing package or validating separate from package.

USERNAMES

{valtools} tracks `Sys.getenv("USERNAME")` to quickly tag validation elements with extended details including names for authorship credit, title and role.

vt_add_users_to_config() Add additional username details to the configuration file.

vt_get_all_users() See a list of usernames already present

Validation Markdown Components

{valtools} populates each component file with some basic info in the header

```
#' @title Test_Case_002
#' @editor An Author
#' @editDate 2021-05-20
```

Filename
username
Current date
don't edit!!
{roxygen2} tags

REQUIREMENTS

vt_use_req() initiates and opens a markdown file in the correct folder. Requirements may have additional header info regarding risk assessment.

```
#' @riskAssessment
#' 1.1:1, High Risk, High Impact
#' 1.2:2, Medium Risk, Medium Impact
```

{roxygen2} tag
Requirement identifier
Risk description

TEST CASES

vt_use_test_case() initiates and opens a markdown file in the correct folder. Test cases may have additional header info regarding coverage, which maps test cases to requirements. A single test case may span multiple requirements

```
#' @coverage
#' 2.1: 2.1
#' 2.2: 2.1, 2.2
```

{roxygen2} tag
Test case identifier
Requirement identifier(s)

Validation Test Code

vt_use_test_code() initiates and opens an R script in the correct folder. Each test is a call to `testthat::test_that()`.

```
#' @editor A Different Author
#' @editDate 2021-05-20
test_that("T2.1", {
  test_data <- data.frame(number = 1:3,
    color = c("red", "black", "blue"))

  testthat::expect_type(test_data$color, "character")
  testthat::expect_equal(test_data$number[1], 1)
})
```

Test identifier
{roxygen2} header tags provided for first test. Add manually for any subsequent tests
Each expectation result will be displayed individually in report

Report Code

vt_use_report(template) Initiate a report from one of the {valtools} templates. The report references content captured in markdown components and test code.

```
template = "validation": Full validation report.
template = "requirement_adoption": Only details needed for acceptance of requirements.
```

REPORT ELEMENTS

{valtools} include pointer functions to capture the environment, scraping {roxygen2} header information and maintaining flexibility across validation modes. Use these to customize a report.

vt_path() Path to the validation/ folder. Used in place of `here::here()`.

vt_get_child_files() list of files in markdown component and test code

vt_file() – for a vector of markdown component and/or test code files, render as child document in the report. Used in place of setting {knitr} chunk option “*child*”.

vt_scrape_* family of functions to scrape info into data.frames:

vt_scrape_sig_table() – usernames and roles

vt_scrape_val_env() – environment details

vt_scrape_change_log() – changelog details

vt_scrape_risk_assessment() – summarize all risk assessment tags in requirements files

vt_scrape_coverage_matrix() – summarize all coverage tags in test case files

vt_scrape_function_editors() – function editor info

vt_scrape_requirement_editors() – requirement editor info

vt_scrape_test_case_editors() – test case editor info

vt_scrape_test_code_editors() – test code editor info

vt_kable_* – converts the output of a **vt_scrape_*** call in a kable object for embedding in report

Validation

Running validation compiles the report code to an eye-readable report suitable for sharing, including evaluation of test code and pointers in the current environment.

separate from package

vt_validate_report()

Uses only code that has already been installed to current workspace to render report.

from source code

vt_validate_source()

Installs package code to temp location to render report. Does not alter current workspace.

at installation

vt_validate_install()

Installs package to run report. Persistent update to workspace.

after installation

vt_validate_installed_package()

Re-run validation report for package that was built/installed under {valtools} paradigm

for distribution

vt_validate_build()

Build a R package tarball bundle suitable for distribution via CRAN. Users will be able to re-validate after installation.