Package: valtools (via r-universe)

September 4, 2024

Title Automate Validated Package Creation

Version 0.4.0.9000

Description Automate the steps necessary to create a validation ready package to make the process of validation simple. This includes setting up the specifications, test cases, test code, and validation report. Also provides tools to be able to execute the validation report from a variety of situations to provide documentation for validation.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Suggests covr, XML, pdftools, bookdown, here, xml2, rvest, fs

Imports desc, devtools, R6, roxygen2, lubridate, rlang, yaml, whoami, usethis, rmarkdown, withr, callr, rprojroot, whisker, glue, knitr, kableExtra, testthat, tidyselect, rstudioapi

VignetteBuilder knitr

SystemRequirements lua

URL `https://phuse-org.github.io/valtools/`

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

RemoteUrl https://github.com/phuse-org/valtools

RemoteRef HEAD

RemoteSha 7408758518f73e41f932022081ec847da85936af

Contents

`dynamic_reference_rendering`

Dynamic Reference Rendering

Description

enable dynamic referencing by reading file and converting any dynamic references into their values for rendering in the validation report.

Usage

```
dynamic_reference_rendering(input, reference = NULL)
```

Arguments

`input`	R object or path to the file to convert dynamic referencing to values
`reference`	which dynamic referencer to use. When NULL, uses internal dynamic referencer.

Value

text with dynamic referencing evaluated

`scrape_roxygen` *Scrape Roxygen blocks*

Description

valtools uses roxygen across multiple file types to provide documentation. this function provides the tooling necessary to scrape from the major file types that we use (R, R test code, markdown, Rmarkdown) and provides a consistent output type to capture the information necessary to help high level functions make assumptions.

Usage

```
scrape_roxygen(file, ..., type = tools::file_ext(file))
```

Arguments

`file`	file to scrape roxygen block from
`...`	These dots are for future extensions and must be empty.
`type`	method of parse_roxygen to use if other that file extension

Value

a list of roxygen blocks found in the file.

`vt_add_file_to_config` *Add validation file ordering to validation config file*

Description

Impose ordering of validation child files

Remove validation file orderingfrom the projects validation config file

Add user information to the projects validation config file to make for easier documentation

Remove user information from the projects validation config file

Get recorded user information from the validation config file to make for easier documentation

Use a Validation Config File Validation configuration for working/output directories, validation report naming conventions, and tracking user information(username,name,title, role). Provides a single location for setting behaviors.

Capture the information on a user that is going to be involved with validation

Usage

```
vt_add_file_to_config(filename, before = NULL, after = NULL)

vt_drop_file_from_config(filename)

vt_add_user_to_config(username = whoami::username(), name, title, role)

vt_drop_user_from_config(username)

vt_get_user_info(username, type = c("name", "title", "role"))

vt_use_config(
  pkg = ".",
  package,
  working_dir,
  output_dir,
  report_rmd_name = "validation.Rmd",
  report_naming_format = "Validation_Report_{package}_v{version}_{date}",
  username_list = list(),
  validation_files = list(),
  ...,
  overwrite = FALSE
)

vt_user(username, name, title, role, ...)
```

Arguments

`filename`	character vector containing filenames in order
`before, after`	Optional destination of new filenames, default is end of existing list. Supports <`tidy-select`> functions. Specifying both is error.
`username`	username of the user.
`name`	full name of the user.
`title`	title of the user.
`role`	role of the user. Can be more than one.
`type`	type of information to pull. select at least one: name, title, role
`pkg`	where to write config file
`package`	character name of package or set validation is being performed for.
`working_dir`	character which directory to be have working validation contents that are used interactively
`output_dir`	character which folder should the contents for validation output to.
`report_rmd_name`	character name of rmarkdown document that is to be used for validation.

`report_naming_format`	character a glue friendly string of the naming structure of the output validation report. use `{package}` for package name, `{version}` to record package version, and `{date}` to capture the date the report was run.
`username_list`	list of user objects created by make_user. Each user contains entries for username, name, title, and role to be used for documentation.
`validation_files`	list of validation files: requirements, test cases and test code. Validation report content will be populated using this list in order.
`...`	additional information about the user to be passed into a list.
`overwrite`	`[boolean]`If a validation file exists, should it be overwritten? Defaults to FALSE.

Value

Used for side effect of adding validation file ordering to validation config file. Invisibly returns TRUE on success.

Used for side effect of removing file ordering information from validation config file. Invisibly returns TRUE on success.

Used for side effect of adding user information to validation config file. Invisibly returns TRUE on success.

Used for side effect of removing user information to validation config file. Invisibly returns TRUE on success.

a character vector length of types requested containing the user information from the validation config file.

Used for side effect to create validation config file. Invisibly returns TRUE on success.

a "user" object

Examples

```
## Not run:

vt_use_validation()

vt_add_file_to_config(filename = "myReqFile.Rmd")

## End(Not run)
## Not run:

vt_use_validation()

vt_add_file_to_config(filename = "myReqFile.Rmd")

vt_drop_file_from_config(filename = "myReqFile.Rmd")


## End(Not run)
## Not run:
```

```
vt_use_validation()

vt_add_user_to_config(
    username = "ellis",
    name = "Ellis Hughes",
    title = "Statistical Programmer",
    role = "Programmer")


## End(Not run)
## Not run:

vt_use_validation()

vt_add_user_to_config(
    username = "ellis",
    name = "Ellis Hughes",
    title = "Statistical Programmer",
    role = "Programmer")

vt_drop_user_from_config(username = "ellis")


## End(Not run)
## Not run:

vt_use_validation()

vt_add_user_to_config(
    username = "ellis",
    name = "Ellis Hughes",
    title = "Statistical Programmer",
    role = "Programmer")

vt_get_user_info(username = "ellis", type = c("name","title"))


## End(Not run)

withr::with_tempdir({
vt_use_validation(
              package = "test.package",
              working_dir = ".",
              output_dir  = ".",
              report_naming_format = "Validation_Report_{package}_v{version}_{date}",
              username_list = list(
                 vt_user(
                       name = "test",
                       title = "test",
                       role = "tester",
                       username = "test"
                       )))
 })
```

```
vt_user(
    username = "ellis",
    name = "Ellis Hughes",
    title = "Statistical Programmer",
    role = "Programmer")
```

vt_file *print files for report generation*

Description

valtools assists the user in generating the validation report by allowing the user to define the order in which

Usage

```
vt_file(file, ..., dynamic_referencing = FALSE)
```

Arguments

`file`	file to evaluate
`...`	These dots are for future extensions and must be empty.
`dynamic_referencing`	Whether to employ dynamic referencing or not. defaults to FALSE.

Value

a list of roxygen blocks found in the file.

vt_get_all_users *Get all users from validation config file without knowing usernames*

Description

Get all users from validation config file without knowing usernames

Usage

```
vt_get_all_users()
```

Value

list of all users in config file

`vt_get_child_files` *Identify ordering of validation or user-designated child files*

Description

Identify ordering of validation or user-designated child files

Usage

```
vt_get_child_files(
  loc = c("folder", "yml"),
  validation_order = c("requirements", "test_cases", "test_code")
)
```

Arguments

`loc`	location to explore. Either "folder" for naive inclusion of validation folder contents, or "yml" to use validation_folder field of validation.yml
`validation_order`	optional ordering of validation folders to search

Value

vector of child file names to include in validation report

Examples

```
withr::with_tempdir({
  vt_use_validation()
  vt_use_test_case("testcase1", username = "a user", open = FALSE)
  vt_use_req("req1", username = "a user", open = FALSE)
  vt_use_test_code("testcode1", username = "another user", open = FALSE)

  # as listed in validation.yml validation_files
  vt_get_child_files(loc = "yml")

  # as ordered in validation subfolders
  vt_get_child_files(loc = "folder",
    validation_order = c("requirements", "test_cases", "test_code"))

})
```

`vt_kable_coverage_matrix` *Kable handler for output of* `vt_scrape_coverage_matrix`

Description

Kable handler for output of `vt_scrape_coverage_matrix`

Usage

```
vt_kable_coverage_matrix(x, format = vt_render_to())
```

Arguments

`x`	data.frame as output from `vt_scrape_coverage_matrix`
`format`	passed to `kable`

Value

knitr_kable object

`vt_kable_sig_table` *Kable defaults for rendering validation report*

Description

Kable defaults for rendering validation report

Usage

```
vt_kable_sig_table(people, format = vt_render_to())
```

Arguments

`people`	A dataframe with the columns role, Name and Title, Signature, and "Date"
`format`	passed to knitr::kable

Value

knitr_kable object

vt_kable_val_env *Generates kable code for validation environment details*

Description

Generates kable code for validation environment details

Usage

```
vt_kable_val_env(val_env, format = vt_render_to())
```

Arguments

`val_env`	data.frame as output from `vt_scrape_val_env`
`format`	passed to `knitr::kable`

Value

knitr_kable object

vt_path *Use dynamic file paths in a validation.*

Description

vt_path() allows access of files relative to the working directory,which is identified by the config file. It is also required to be used in the validation report for cases where validation of installed packages is intended as it will shift access to the correct location for the installed package for access.

Usage

```
vt_path(...)

vt_find_config()
```

Arguments

`...`	`[character]` Path components below the validation folder, can be empty. Each argument should be a string containing one or more path components separated by a forward slash `"/"`.

Details

vt_find_config() locates the config file in the working directory, and returns the full path to it.

Examples

```
withr::with_tempdir({callr::r(function(){

 valtools::vt_use_validation()

 valtools::vt_path()
 valtools::vt_path("some", "reqs", "req01.md")
 valtools::vt_path("some/reqs/req01.md")

 valtools::vt_find_config()

})})
```

`vt_render_to` *output to render kable to*

Description

reads the knitr and rmarkdown options to determine which output type is being rendered

Usage

```
vt_render_to()
```

`vt_render_validation_report` *provide a nice wrapper to set states around render*

Description

This package is not intended for use by the end users. This is to be used within valtools packages.

Usage

```
vt_render_validation_report(
  report_path,
  output_dir = dirname(report_path),
  output_file = NULL,
  ...,
  render_time = c("build", "installed"),
  package = ""
)
```

Arguments

`report_path`	path to the validation report rmarkdown
`output_dir`	path to directory to output rendered report. defaults to same folder
`output_file`	expected output filename sans extension
`...`	arguments passed to render
`render_time`	type of rendering of validation to run, "build" or "installed".
`package`	the report type of rendering of validation to run, "build" or "installed".

`vt_run_test_code_file` *Evaluate the test code file*

Description

Evaluate the test code file

Turn test code results data.frame into kable output

Usage

```
vt_run_test_code_file(file, test_env = new.env(), ..., ref = vt_path())

vt_kable_test_code_results(results, format = vt_render_to())
```

Arguments

`file`	full path to test code file.
`test_env`	environment to perform tests in
`...`	argument passed to `knitr::kable()`
`ref`	reference path to use. Defaults to vt_path()
`results`	results data.frame from `vt_run_test_code_file()`
`format`	passed to `knitr::kable`

Value

a kable with variables: `Test`, `Expected`, `Results`, `Pass/Fail`. Suitable for including in validation report

kableExtra object with formatting

`vt_scrape_change_log` *Scrape change log from a validation project*

Description

Scrape change log from a validation project

Format change log info table for validation report

Initiate a change_log file

Usage

```
vt_scrape_change_log()

vt_kable_change_log(change_log_info, format = vt_render_to())

vt_use_change_log(date = NULL, version = NULL, open = interactive())
```

Arguments

`change_log_info`	data.frame as exported from `vt_scrape_change_log`
`format`	passed to `knitr::kable`
`date`	passed to template
`version`	version to set in news file
`open`	whether to open the file after

Value

data.frame with variables `version`, `effective_date`, `description`

a knitr_kable object

path to change log file, used for side effect of creating change_log

Note

Extracts validation version, date, and description from change log items that start with `[validation]`.

Examples

```
withr::with_tempdir({
 file.create(".here")
 vt_use_validation()

 vt_use_change_log()

 log_data <- vt_scrape_change_log()
 print(log_data)
 vt_kable_change_log(log_data)

})
```

vt_scrape_coverage_matrix

Scrape "coverage" tag in test code to generate mapping

Description

Scrape "coverage" tag in test code to generate mapping

Usage

```
vt_scrape_coverage_matrix(
  type = c("long", "wide"),
  reference = NULL,
  src = ".",
  ref = vt_path()
)
```

Arguments

type	one of "long" or "wide" which determines shape of output table
reference	dynamic reference holder if it already exists
src, ref	passed to vt_scrape_tags_from

Value

a data.frame mapping requirement ids to test case ids.

vt_scrape_requirement_editors

Scrape authorship information

Description

These functions provide utitilies to scrape the editor and editDate roxygen tags and put them into a nice data.frame for use in the validation reports. In addition, opinionated kable formatting functions are provided as well to facilitate nice printing in the reports.

Usage

```
vt_scrape_requirement_editors(
  tags = c("editor", "editDate"),
  src = ".",
  ref = vt_path(),
  dynamic_ref = NULL
)

vt_scrape_test_case_editors(
  tags = c("editor", "editDate"),
  src = ".",
  ref = vt_path(),
  dynamic_ref = NULL
)

vt_scrape_test_code_editors(
  tags = c("editor", "editDate", "deprecate"),
  src = ".",
  ref = vt_path(),
  dynamic_ref = NULL
)

vt_scrape_function_editors(
  tags = c("editor", "editDate", "export"),
  src = ".",
  ref = vt_path()
)

vt_kable_requirement_editors(x, format = vt_render_to())

vt_kable_function_editors(x, format = vt_render_to())

vt_kable_test_case_editors(x, format = vt_render_to())

vt_kable_test_code_editors(x, format = vt_render_to())
```

Arguments

`tags`	which tags to keep. defaults to editor and editDate
`src`	path to package sources. defaults to current directory and passed to `vt_scrape_tags_from`
`ref`	reference path to whre validation documentation lives. defaults to `vt_path` annd passed to `vt_scrape_tags_from`.
`dynamic_ref`	dynamic referencer object
`x`	data.frame as exported from vt_scrape_*
`format`	passed to `knitr::kable`, NULL by default

Value

data.frame containing the results of the scraped roxygen tags for each section

knitr_kable object

Note

vt_scrape_functions Requires access to raw R/ or function documentation parsed via valtools into validation/ folder. Cannot pull information from installed R/ location.

Examples

```
withr::with_tempdir({

captured_output <- capture.output({vt_create_package(open = FALSE)})
vt_use_req(
     name = "req1",
     username = "B user",
     title = "Requirement 1",
     open = FALSE)
writeLines(c(
    "#' @title Say Hello",
    "#' @editor B User",
    "#' @editDate 2021-04-27",
    "#' @export",
    "hello <- function(){print(\"Hello\")}"
    ), con = "R/hello.R")
vt_use_test_case(
    name = "testcase1",
    username = "B user",
    title = "TesT Case 1",
    open = FALSE)
vt_use_test_code(
    name = "testcode1",
    username = "C user",
    open = FALSE)

req_editors <- vt_scrape_requirement_editors()
vt_kable_requirement_editors(req_editors)

fun_editors <- vt_scrape_function_editors()
vt_kable_function_editors(fun_editors)

t_case_editors <- vt_scrape_test_case_editors()
vt_kable_test_case_editors(t_case_editors)

t_code_editors <- vt_scrape_test_code_editors()
vt_kable_test_code_editors(t_code_editors)


})
```

`vt_scrape_risk_assessment`
Scrape "riskAssessment" tag in requirements to generate table

---

Description

Scrape "riskAssessment" tag in requirements to generate table

Kable handler for output of `vt_scrape_risk_assessment`

Usage

```
vt_scrape_risk_assessment(reference = NULL, src = ".", ref = vt_path())

vt_kable_risk_assessment(x, format = vt_render_to())
```

Arguments

`reference`	dynamic reference holder if it already exists
`src, ref`	passed to `vt_scrape_tags_from`
`x`	data.frame as output from `vt_scrape_risk_assessment`
`format`	passed to `kable`

Value

a data.frame documenting requirements to risk assessments

knitr_kable object

`vt_scrape_section` *Retrieve the value block of a custom section tagged via roxygen2*

Description

Looks for the value in a custom roxygen sections. Custom sections are named using `@section <NAME>:`, where colon is use to indicate end of the name, and value starts on next line.

Usage

```
vt_scrape_section(tag, block)
```

Arguments

`tag`	name of the section, case insensitive.
`block`	character vector that holds the documentation block. `#'` may be present or omitted.

Value

section value

Last Updated By

Marie Vendettuoli

Last Updated Date

2021-02-18

Examples

```
roxy_block1 <- c("@title Title1", "@param param1 definition",
    "@section Last updated date:", "2021-01-01", "@importFrom utils sessionInfo",
    "@export" )
vt_scrape_section("Last updated date:", roxy_block1)

roxy_block2 <- paste0("#' ", roxy_block1)
vt_scrape_section("Last updated date:", roxy_block2)
```

---

vt_scrape_sig_table *Generate a signature table for a validation report*

Description

Generate a signature table for a validation report

Usage

```
vt_scrape_sig_table(usernames = NULL)
```

Arguments

usernames	list of vt_names to use when validation.yml does not exist

Value

A dataframe created from the validation config containing a row for each user with the columns: role, name_and_title, signature, and date.

`vt_scrape_tags_from` *Retrieve roxygen tags as a data.frame from requirements, test cases, test code and functions*

Description

Looks for roxygen2 function documentation in /R for author details. Assumes that author and date are tagged via custom sections `@section Last updated by:` and `@section Last updated date:`, respectively. To exclude a roxygen block from this scraping, omit these section names.

If using a dummy documentation file, looks for `@name` to capture function name, otherwise uses the actual function call.

Exported or internal status does not affect scraping.

Usage

```
vt_scrape_tags_from(
  type,
  tags = c("editor", "editDate"),
  src = ".",
  ref = vt_path()
)
```

Arguments

`type`	type of scraping to be done. one of "requirements","test_cases","test_code","functions". to call functions. working directory must be an R package, or path identified in src must be an R package.
`tags`	which tags to keep. defaults to editor and editDate
`src`	path to package source. defaults to the current directory.
`ref`	reference path to where validation documentation lives. defaults to vt_path()

Last Updated by

Ellis Hughes

Last updated date

2021-03-05

Note

At this time, this function does not retrieve documentation captured for functions dispatched within an R6 class. Tags at the class level documentation are retrieved.

`vt_scrape_val_env` *Retrieve validation environment*

Description

Retrieves dependencies used in validation report. Includes: OS, R version, packages listed in the validation package DESCRIPTION (Depends/Imports/Suggests), packages present in current session.

Usage

```
vt_scrape_val_env(pkg = ".")
```

Arguments

`pkg`	path to package

Value

data.frame with columns:

- `resource` "OS", "R" or package name
- `type` identifier of requirement type:
  - system - OS or R resource
  - package_req - from DESCRIPTION Depends/Imports of the package being validated
  - extended_req - from DESCRIPTION Suggests of the package being validated
  - session - packages in current workspace not captured via package_req/extended_req
- `detail` OS or version details

Last updated by

Marie Vendettuoli

Last updated date

2021-02-03

vt_username *Get current username*

Description

Wrapper for whoami::username

Usage

```
vt_username()
```

Details

@returns `[character]` Username of the person that called the function

Examples

```
withr::with_tempdir({
vt_use_validation(
    username_list = list(vt_user(
      username = whoami::username(),
      name = "test",
      title = "title",
      role = "role")))
vt_username()
})
```

vt_use_report *Create validation report from template*

Description

Create validation report from template

Usage

```
vt_use_report(
  pkg_name = NULL,
  template = "validation",
  dynamic_referencing = FALSE,
  open = is_interactive()
)
```

Arguments

`pkg_name`	name of package
`template`	what validation report template from valtools to use, one of "validation" (default) or "requirements"
`dynamic_referencing`	Should dynamic referencing be enabled by default. Boolean defaults to FALSE.
`open`	boolean to open the validation report for further editing

---

`vt_use_test_case` *Create a validation requirement, test case, or test code file*

Description

Create a validation requirement, test case, or test code file

Usage

```
vt_use_test_case(
  name,
  username = vt_username(),
  title = NULL,
  open = interactive(),
  add_before = NULL,
  add_after = NULL
)

vt_use_test_code(
  name,
  username = vt_username(),
  open = interactive(),
  add_before = NULL,
  add_after = NULL
)

vt_use_req(
  name,
  username = vt_username(),
  title = NULL,
  open = interactive(),
  add_before = NULL,
  add_after = NULL
)
```

Arguments

`name`	The name/path of the validation item. These can be named with your file system separator and will be organized as a directory structure. Items are located at `./inst/validation/<ItemType>/{name}`.
`username`	The username to insert into the validation item as the author.
`title`	Title for the requirement defaults to be the base name passed sans file paths or extensions.
`open`	Should the newly made file be opened for editing.
`add_before`, `add_after`	If either parameters is supplied, the location to add the validation item to the validation configuration. If no parameter is passed the item is added at the end.

Value

Path to the newly created validation item file, invisibly.

Examples

```
withr::with_tempdir({
vt_create_package("example.package")
setwd("example.package")
vt_add_user_to_config(
  username = whoami::username(),
  name = "Sample Name",
  title = "Sample",
  role = "example"
 )
# Create req at the cases top level `inst/validation/cases/case1`
vt_use_test_case("case1", open = FALSE)

# Create req at `inst/validation/cases/regTests/Update2/case2`
vt_use_test_case("regTests/Update2/case2", open = FALSE, add_before = "case1.md")

# Create a test case using tidy select
vt_use_test_case("case1a", open = FALSE, add_after = tidyselect::starts_with("case1"))

})
```

`vt_use_validation` *Create a validation structure*

Description

Creates a structure for validation artifacts. Validation items are stored in `inst/validation`

Create the validation packet infrastructure. Intended to create validation infrastructure external to an R package.

Usage

```
vt_use_validation(pkg = ".", working_dir, ...)

vt_create_package(
  pkg = ".",
  ...,
  fields = list(),
  rstudio = rstudioapi::isAvailable(),
  roxygen = TRUE,
  check_name = TRUE,
  open = rlang::is_interactive()
)

vt_create_packet(
  path = ".",
  target,
  ...,
  rstudio = rstudioapi::isAvailable(),
  open = rlang::is_interactive()
)
```

Arguments

`pkg`	Top level directory of a package
`working_dir`	validation working directory of the project. Defaults to
`...`	Additional argument passed to `vt_use_config()`
`fields`	A named list of fields to add to `DESCRIPTION`, potentially overriding default values. See `use_description()` for how you can set personalized defaults using package options.
`rstudio`	If `TRUE`, calls `use_rstudio()` to make the new package or project into an RStudio Project. If `FALSE` and a non-package project, a sentinel `.here` file is placed so that the directory can be recognized as a project by the here or rprojroot packages.
`roxygen`	Do you plan to use roxygen2 to document your package?
`check_name`	Whether to check if the name is valid for CRAN and throw an error if not.
`open`	If `TRUE`, activates the new project: • If using RStudio desktop, the package is opened in a new session. • If on RStudio server, the current RStudio project is activated. • Otherwise, the working directory and active project is changed.
`path`	A path. If it exists, it is used. If it does not exist, it is created, provided that the parent path exists.
`target`	target of validation. Character name of package or scope validation packet is being performed for.

`vt_validate_source` *Validate a package*

Description

`vt_validate_source` runs the validation on the current source, temporarily installing the source to properly evaluate the report. `vt_validate_build()` runs the same step, then compiles a bundle that includes th the validation report and any other contents that are required for validation. Finally, `vt_validate_installed_package` run rerun the validation report for packages that were built and then installed using the`vt_validate_build()`.

Usage

```
vt_validate_source(src = ".", open = interactive())

vt_validate_build(src = ".", ...)

vt_validate_install(
  src = ".",
  ...,
  install_verbose = TRUE,
  install_tests = TRUE,
  reload = TRUE
)

vt_validate_installed_package(
  package,
  output_directory = ".",
  open = interactive()
)

vt_validate_report(version, open = interactive())
```

Arguments

`src`	location of the source code. Assumed to be the same location as "pkg"
`open`	should the validation report be opened after it is built?
`...`	Additional argument passed to `devtools::build()`
`install_verbose`	should the installation be verbose?
`install_tests`	should the installation include installation of package-specific tests (if any)?
`reload`	Should package be reloaded after install? defaults to TRUE
`package`	installed package name
`output_directory`	Location of directory to output validation report

`version`	version of validation report to output. If missing, it tries to use the change_log.md, if that is missing then looks at the package version if the validation package is of an R package.

Value

path to either the validation report or the bundled package

Index