

# Package ‘framework’

February 4, 2026

**Type** Package

**Title** Structured Data Science in R

**Version** 1.0.1

**Maintainer** Erik Westlund <erik@table1.org>

**Description** Project scaffolding and workflow tools for reproducible data science.

Manages packages, tracks data integrity, handles database connections,  
generates notebooks, and publishes to S3-compatible storage.

**License** MIT + file LICENSE

**URL** <https://framework.table1.org>, <https://github.com/table1/framework>

**BugReports** <https://github.com/table1/framework/issues>

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE, roclists = c(``rd", ``namespace",  
``collate"))

**RoxygenNote** 7.3.3

**Imports** checkmate, DBI, RSQLite, yaml, fs, readr, dotenv, openssl,  
lubridate, jsonlite, plumber

**Suggests** testthat (>= 3.0.0), arrow, aws.s3, aws.signature,  
BiocManager, cli, cyclocomp, devtools, dplyr, DT, duckdb,  
ggplot2, haven, htmltools, htmlwidgets, httpgd, knitr,  
languageserver, odbc, pool, prismjs, qs, R.utils, readxl,  
remotes, renv, RMariaDB, rmarkdown, RPostgres, tm, usethis,  
withr, writexl

**Config/testthat.edition** 3

**Depends** R (>= 4.1.0)

**NeedsCompilation** no

**Author** Erik Westlund [aut, cre, cph]

## Contents

framework-package . . . . .	8
.apply_auto_fix . . . . .	8
.build_directory_table . . . . .	8
.calculate_file_hash . . . . .	9
.catalog_field_default . . . . .	9

.catalog_find_field . . . . .	9
.catalog_project_type_defaults . . . . .	10
.check_duckdb_exists . . . . .	10
.check_env_gitignored . . . . .	10
.check_gitignore_coverage . . . . .	11
.check_git_available . . . . .	11
.check_git_history . . . . .	11
.check_git_status . . . . .	11
.check_mysql_exists . . . . .	12
.check_path_ignored . . . . .	12
.check_postgres_exists . . . . .	12
.check_private_data_exposure . . . . .	13
.check_sqlite_exists . . . . .	13
.check_sqlserver_exists . . . . .	13
.cleanup_gitkeep_files . . . . .	14
.collect_all_s3_connections . . . . .	14
.commit_after_scaffold . . . . .	14
.configure_git_hooks . . . . .	15
.connect_duckdb . . . . .	15
.connect_mysql . . . . .	15
.connect_postgres . . . . .	16
.connect_sqlite . . . . .	16
.connect_sqlserver . . . . .	17
.copy_config_files . . . . .	17
.create_ai_files . . . . .	17
.create_ai_instructions . . . . .	18
.create_code_workspace . . . . .	18
.create_config_file . . . . .	18
.create_dev_rprofile . . . . .	19
.create_duckdb_db . . . . .	19
.create_gitignore . . . . .	19
.create_ide_configs . . . . .	20
.create_initial_commit . . . . .	20
.create_init_file . . . . .	20
.create_mysql_db . . . . .	21
.create_postgres_db . . . . .	21
.create_project_config . . . . .	22
.create_project_directories . . . . .	22
.create_renvignore . . . . .	23
.create_rproj_file . . . . .	23
.create_scaffold_file . . . . .	23
.create_sqlite_db . . . . .	24
.create_sqlserver_db . . . . .	24
.create_stub_files . . . . .	24
.create_template_db . . . . .	25
.create_vscode_settings . . . . .	25
.create_vscode_workspace . . . . .	25
.customize_project_files . . . . .	26
.data_spec_update . . . . .	26
.default_directory_table . . . . .	27
.default_render_dirs_for_type . . . . .	27
.default_root_render_dir_for_type . . . . .	27

.delete_init_file . . . . .	27
.df_to_list_of_lists . . . . .	28
.display_next_steps . . . . .	28
.ensure_framework_db . . . . .	28
.ensure_framework_template . . . . .	29
.ensure_output_dir . . . . .	29
.find_project_root . . . . .	29
.find_stub_template . . . . .	30
.framework_catalog_default_path . . . . .	30
.framework_catalog_user_path . . . . .	30
.framework_templates_dir . . . . .	31
.generate_data_section . . . . .	31
.generate_environment_section . . . . .	31
.generate_function_reference . . . . .	31
.generate_header_section . . . . .	32
.generate_hook_script . . . . .	32
.generate_notes_section . . . . .	32
.generate_packages_section . . . . .	33
.generate_project_type_section . . . . .	33
.get_cache . . . . .	33
.get_cache_dir . . . . .	34
.get_data_directories . . . . .	34
.get_data_pathSuggestions . . . . .	34
.get_data_record . . . . .	35
.get_db_connection . . . . .	35
.get_driver_info . . . . .	35
.get_example_data_path . . . . .	36
.get_hook_setting . . . . .	36
.get_metadata . . . . .	36
.get_notebook_dir_from_config . . . . .	37
.get_package_list_from_config . . . . .	37
.get_package_requirements . . . . .	38
.get_placeholders . . . . .	38
.get_scaffold_history . . . . .	38
.get_settings_file . . . . .	39
.git_available . . . . .	39
.guess_content_type . . . . .	40
.has_settings_file . . . . .	40
.identify_private_dirs . . . . .	40
.init_db . . . . .	41
.init_git_repo . . . . .	41
.init_renv . . . . .	41
.init_standard . . . . .	42
.install_package . . . . .	42
.install_package_base . . . . .	43
.install_package_renv . . . . .	43
.install_required_packages . . . . .	44
.is_data_file . . . . .	44
.is_git_repo . . . . .	44
.is_initialized . . . . .	45
.list_available_stubs . . . . .	45
.load_ai_template . . . . .	46

.load_configuration . . . . .	46
.load_environment . . . . .	46
.load_functions . . . . .	47
.load_libraries . . . . .	47
.load_template_content . . . . .	47
.make_env . . . . .	48
.mark_scaffolded . . . . .	48
.migrate_config_from_previous . . . . .	49
.normalize_expire_after . . . . .	49
.normalize_notebook_name . . . . .	50
.normalize_package_spec . . . . .	50
.parse_assistant_selection . . . . .	51
.parse_package_spec . . . . .	51
.parse_sections . . . . .	52
.pretty_print_config . . . . .	52
.print_audit_summary . . . . .	52
.prompt_ai_support_init . . . . .	53
.prompt_ai_support_install . . . . .	53
.read_framework_template . . . . .	53
.remove_cache . . . . .	54
.remove_data . . . . .	54
.remove_init . . . . .	54
.remove_metadata . . . . .	55
.replace_moustache_placeholders . . . . .	55
.replace_section . . . . .	55
.require_driver . . . . .	56
.require_git_repo . . . . .	56
.reset_framework_template . . . . .	57
.s3_client . . . . .	57
.s3_public_url . . . . .	57
.s3_upload_dir . . . . .	58
.s3_upload_file . . . . .	58
.save_audit_result . . . . .	59
.save_result . . . . .	59
.scan_for_orphaned_files . . . . .	59
.set_cache . . . . .	60
.set_data . . . . .	60
.set_ggplot_theme . . . . .	61
.set_metadata . . . . .	61
.set_random_seed . . . . .	62
.slugify . . . . .	62
.sync_packages_to_renv . . . . .	63
.to_kebab_case . . . . .	63
.update_data_with_hash . . . . .	63
.update_frameworkrc . . . . .	64
.update_gitignore_for_renv . . . . .	64
.update_hook_config . . . . .	64
.uses_split_file . . . . .	65
.validate_driver . . . . .	65
.validate_refresh . . . . .	66
.validate_settings_catalog . . . . .	66
.write_framework_template . . . . .	66

add_project_to_config . . . . .	67
ai_generate_context . . . . .	67
ai_regenerate_context . . . . .	68
ai_sync_context . . . . .	69
bootstrap_project_init . . . . .	70
cache . . . . .	70
cache_flush . . . . .	71
cache_forget . . . . .	71
cache_get . . . . .	71
cache_list . . . . .	72
cache_remember . . . . .	72
capture_output . . . . .	73
config . . . . .	74
configure_ai_agents . . . . .	74
configure_author . . . . .	75
configure_connection . . . . .	76
configure_data . . . . .	77
configure_directories . . . . .	78
configure_global . . . . .	79
configure_packages . . . . .	80
connections_list . . . . .	82
connections_s3 . . . . .	82
connection_begin . . . . .	82
connection_check . . . . .	83
connection_check_leaks . . . . .	83
connection_close_all . . . . .	84
connection_commit . . . . .	84
connection_delete . . . . .	85
connection_find . . . . .	85
connection_find_by . . . . .	86
connection_insert . . . . .	86
connection_pool . . . . .	87
connection_pool_close . . . . .	88
connection_pool_close_all . . . . .	89
connection_pool_list . . . . .	90
connection_restore . . . . .	90
connection_rollback . . . . .	91
connection_update . . . . .	91
connection_with_pool . . . . .	92
connection_with_transaction . . . . .	92
data_add . . . . .	93
data_info . . . . .	94
data_list . . . . .	95
data_read . . . . .	95
data_read_or_cache . . . . .	96
data_save . . . . .	96
data_spec_get . . . . .	97
db_connect . . . . .	97
db_drivers_install . . . . .	98
db_drivers_status . . . . .	99
db_execute . . . . .	99
db_list . . . . .	100

db_query . . . . .	101
db_transaction . . . . .	101
db_with . . . . .	102
docs_export . . . . .	103
dot-has_column . . . . .	105
dot-list_columns . . . . .	106
dot-list_tables . . . . .	107
env_clear . . . . .	108
env_default_template_lines . . . . .	108
env_lines_from_variables . . . . .	109
env_resolve_lines . . . . .	109
env_summary . . . . .	109
framework_template_path . . . . .	110
framework_view . . . . .	110
fw_config_dir . . . . .	111
get_default_global_config . . . . .	111
get_global_setting . . . . .	111
git_add . . . . .	112
git_commit . . . . .	113
git_diff . . . . .	113
git_hooks_disable . . . . .	114
git_hooks_enable . . . . .	114
git_hooks_install . . . . .	115
git_hooks_list . . . . .	116
git_hooks_uninstall . . . . .	116
git_log . . . . .	117
git_pull . . . . .	117
git_push . . . . .	118
git_security_audit . . . . .	118
git_status . . . . .	120
gui . . . . .	121
init_global_config . . . . .	122
list_framework_templates . . . . .	122
list_metadata . . . . .	122
load_settings_catalog . . . . .	123
make_notebook . . . . .	123
make_presentation . . . . .	125
make_qmd . . . . .	126
make_revealjs . . . . .	127
make_rmd . . . . .	128
make_script . . . . .	129
new . . . . .	130
new_course . . . . .	131
new_presentation . . . . .	132
new_project . . . . .	132
new_project_sensitive . . . . .	134
now . . . . .	135
outputs . . . . .	135
packages_install . . . . .	135
packages_list . . . . .	136
packages_restore . . . . .	136
packages_snapshot . . . . .	137

packages_status . . . . .	137
packages_update . . . . .	138
project_add_directory . . . . .	138
project_create . . . . .	140
project_info . . . . .	141
project_list . . . . .	141
publish . . . . .	142
publish_data . . . . .	143
publish_dir . . . . .	143
publish_list . . . . .	144
publish_notebook . . . . .	145
quarto_generate_all . . . . .	146
quarto_regenerate . . . . .	147
read_frameworkrc . . . . .	147
read_framework_template . . . . .	148
remove_project_from_config . . . . .	148
renv_disable . . . . .	149
renv_enable . . . . .	149
renv_enabled . . . . .	150
renv_restore . . . . .	150
renv_snapshot . . . . .	151
renv_status . . . . .	151
renv_sync . . . . .	151
renv_update . . . . .	152
reset_framework_template . . . . .	152
result_list . . . . .	153
save_figure . . . . .	153
save_model . . . . .	155
save_notebook . . . . .	155
save_report . . . . .	157
save_table . . . . .	157
scaffold . . . . .	158
scratch_capture . . . . .	160
scratch_clean . . . . .	161
settings . . . . .	161
settings_read . . . . .	162
settings_write . . . . .	163
setup . . . . .	163
standardize_wd . . . . .	164
status . . . . .	165
storage_test . . . . .	165
stubs_list . . . . .	166
stubs_path . . . . .	167
stubs_publish . . . . .	167
view . . . . .	169
view_create . . . . .	169
view_detail . . . . .	170
write_frameworkrc . . . . .	171
write_framework_template . . . . .	172

---

```
framework-package  framework: Structured Data Science in R
```

---

## Description

Project scaffolding and workflow tools for reproducible data science. Manages packages, tracks data integrity, handles database connections, generates notebooks, and publishes to S3-compatible storage.

## Author(s)

**Maintainer:** Erik Westlund <erik@pequod.sh> [copyright holder]

## See Also

Useful links:

- <https://framework.table1.org>
- <https://github.com/table1/framework>
- Report bugs at <https://github.com/table1/framework/issues>

---

```
.apply_auto_fix      Apply auto-fix for common issues
```

---

## Description

Apply auto-fix for common issues

## Usage

```
.apply_auto_fix(findings, verbose)
```

---

```
.build_directory_table
Build directory table from config
```

---

## Description

Build directory table from config

## Usage

```
.build_directory_table(dirs, project_type)
```

---

.calculate\_file\_hash  
*Calculate hash of a file*

---

### Description

Calculate hash of a file

### Usage

.calculate\_file\_hash(file\_path)

### Arguments

file\_path Path to the file

### Value

The hash of the file as a character string

---

.catalog\_field\_default  
*Retrieve a default value from the catalog, falling back when missing*

---

### Description

Retrieve a default value from the catalog, falling back when missing

### Usage

.catalog\_field\_default(catalog, field\_id, fallback = NULL)

---

.catalog\_find\_field  
*Convenience accessor for catalog field definitions*

---

### Description

Convenience accessor for catalog field definitions

### Usage

.catalog\_find\_field(catalog, field\_id)

`.catalog_project_type_defaults`

*Convert catalog project type metadata into default configuration values*

### Description

Convert catalog project type metadata into default configuration values

### Usage

`.catalog_project_type_defaults(project_types)`

`.check_duckdb_exists`

*Check if a DuckDB database exists*

### Description

Check if a DuckDB database exists

### Usage

`.check_duckdb_exists(config)`

### Arguments

<code>config</code>	Connection configuration from settings.yml
---------------------	--

### Value

TRUE if database exists, FALSE otherwise

`.check_env_gitignored`

*Check if .env is gitignored*

### Description

Checks if .env is listed in .gitignore and warns if not.

### Usage

`.check_env_gitignored()`

### Value

Invisibly returns logical indicating if .env is gitignored

---

```
.check_gitignore_coverage  
Check gitignore coverage for data files
```

---

### Description

Check gitignore coverage for data files

### Usage

```
.check_gitignore_coverage(data_dirs, extensions, verbose)
```

---

```
.check_git_available  
Check if git is available
```

---

### Description

Check if git is available

### Usage

```
.check_git_available()
```

---

```
.check_git_history Check git history for leaked data files
```

---

### Description

Check git history for leaked data files

### Usage

```
.check_git_history(data_dirs, extensions, history_depth, verbose)
```

---

```
.check_git_status Check git status and provide helpful reminder
```

---

### Description

Check git status and provide helpful reminder

### Usage

```
.check_git_status()
```

---

```
.check_mysql_exists
```

*Check if a MySQL/MariaDB database exists*

---

**Description**

Check if a MySQL/MariaDB database exists

**Usage**

```
.check_mysql_exists(config)
```

**Arguments**

config	Connection configuration from settings.yml
--------	--

**Value**

TRUE if database exists, FALSE otherwise

---

```
.check_path_ignored
```

*Check if path is ignored by .gitignore patterns*

---

**Description**

Check if path is ignored by .gitignore patterns

**Usage**

```
.check_path_ignored(path, gitignore_patterns)
```

---

```
.check_postgres_exists
```

*Check if a PostgreSQL database exists*

---

**Description**

Check if a PostgreSQL database exists

**Usage**

```
.check_postgres_exists(config)
```

**Arguments**

config	Connection configuration from settings.yml
--------	--

**Value**

TRUE if database exists, FALSE otherwise

---

.check\_private\_data\_exposure  
*Check if private data is tracked by git*

---

### Description

Check if private data is tracked by git

### Usage

.check\_private\_data\_exposure(data\_dirs, git\_available, verbose)

---

.check\_sqlite\_exists  
*Check if a SQLite database exists*

---

### Description

Check if a SQLite database exists

### Usage

.check\_sqlite\_exists(config)

### Arguments

config Connection configuration from settings.yml

### Value

TRUE if database exists, FALSE otherwise

---

.check\_sqlserver\_exists  
*Check if a SQL Server database exists*

---

### Description

Check if a SQL Server database exists

### Usage

.check\_sqlserver\_exists(config)

### Arguments

config Connection configuration from settings.yml

### Value

TRUE if database exists, FALSE otherwise

---

`.cleanup_gitkeep_files`

*Remove .gitkeep files from data/ and functions/ directories*

---

### Description

Remove .gitkeep files from data/ and functions/ directories

### Usage

`.cleanup_gitkeep_files(target_dir = ".")`

---

`.collect_all_s3_connections`

*Collect all S3 connections from config*

---

### Description

Gathers all S3/storage bucket connections from configuration, along with the default bucket name if specified.

### Usage

`.collect_all_s3_connections(config)`

### Arguments

`config` Configuration object from `settings_read()`.

### Value

List with connections and `default_bucket` fields.

---

`.commit_after_scaffold`

*Create initial commit after first successful scaffold*

---

### Description

Create initial commit after first successful scaffold

### Usage

`.commit_after_scaffold()`

### Note

This function is now deprecated. Initial commits are created during `project_create()` instead of `scaffold()`. Kept for backward compatibility with older projects.

---

.configure\_git\_hooks

*Configure git hooks based on environment variables*

---

## Description

Configure git hooks based on environment variables

## Usage

.configure\_git\_hooks(target\_dir = ".")

---

.connect\_duckdb

*Connect to a DuckDB database*

---

## Description

Connect to a DuckDB database

## Usage

.connect\_duckdb(config)

## Arguments

config Connection configuration from settings.yml

## Value

A DuckDB database connection

---

.connect\_mysql

*Connect to a MySQL or MariaDB database*

---

## Description

Connect to a MySQL or MariaDB database

## Usage

.connect\_mysql(config)

## Arguments

config Connection configuration from settings.yml

## Value

A MySQL/MariaDB database connection

---

`.connect_postgres` *Connect to a PostgreSQL database*

---

### Description

Connect to a PostgreSQL database

### Usage

`.connect_postgres(config)`

### Arguments

`config` Connection configuration from `settings.yml`

### Value

A PostgreSQL database connection

---

`.connect_sqlite` *Connect to a SQLite database*

---

### Description

Connect to a SQLite database

### Usage

`.connect_sqlite(config)`

### Arguments

`config` Connection configuration from `settings.yml`

### Value

A SQLite database connection

---

.connect\_sqldatabase *Connect to a SQL Server database*

---

### Description

Connect to a SQL Server database

### Usage

.connect\_sqldatabase(config)

### Arguments

config Connection configuration from settings.yml

### Value

A SQL Server database connection via ODBC

---

.copy\_config\_files *Copy config files between directories*

---

### Description

Copy config files between directories

### Usage

.copy\_config\_files(from\_dir, to\_dir)

---

.create\_ai\_files *Create AI context files*

---

### Description

Create AI context files

### Usage

.create\_ai\_files(project\_dir, assistants, canonical\_content, type)

```
.create_ai_instructions
Create AI Assistant Instruction Files
```

### Description

Internal function called during `project_create()` to create AI assistant instruction files based on user preferences.

### Usage

```
.create_ai_instructions(
  assistants,
  target_dir = ".",
  project_name = NULL,
  project_type = "project"
)
```

### Arguments

<code>assistants</code>	Character vector of assistants: "claude", "copilot", "agents"
<code>target_dir</code>	Target directory (default: current directory)
<code>project_name</code>	Project name for template substitution
<code>project_type</code>	Project type for template selection ("project", "project_sensitive", "course", "presentation")

```
.create_code_workspace
Create .code-workspace file for VSCode/Positron
```

### Description

Create `.code-workspace` file for VSCode/Positron

### Usage

```
.create_code_workspace(project_dir, name)
```

```
.create_config_file
Create settings.yml from template
```

### Description

Create `settings.yml` from template

### Usage

```
.create_config_file(type = "analysis", attach_defaults = TRUE, subdir = NULL)
```

---

.create\_dev\_rprofile  
Create development .Rprofile

---

### Description

Create development .Rprofile

### Usage

.create\_dev\_rprofile(subdir = NULL)

---

.create\_duckdb\_db Create a new DuckDB database

---

### Description

Create a new DuckDB database

### Usage

.create\_duckdb\_db(config)

### Arguments

config Connection configuration from settings.yml

### Value

TRUE if successful

---

.create\_gitignore Create .gitignore file

---

### Description

Create .gitignore file

### Usage

.create\_gitignore(project\_dir, content)

---

```
.create_ide_configs  
Create IDE configuration files
```

---

### Description

Create IDE configuration files

### Usage

```
.create_ide_configs(project_name, target_dir = ".", python = FALSE)
```

### Arguments

project_name	Project name
target_dir	Target directory
python	Include Python configuration

---

```
.create_initial_commit  
Create initial git commit after all initialization is complete
```

---

### Description

Create initial git commit after all initialization is complete

### Usage

```
.create_initial_commit(target_dir = ".")
```

---

```
.create_init_file  Create init.R from template
```

---

### Description

Create init.R from template

### Usage

```
.create_init_file(project_name, type, lintr, subdir = NULL)
```

---

.create\_mysql\_db     *Create a new MySQL/MariaDB database*

---

### Description

Create a new MySQL/MariaDB database

### Usage

.create\_mysql\_db(config)

### Arguments

config        Connection configuration from settings.yml

### Value

TRUE if successful

---

.create\_postgres\_db  
    *Create a new PostgreSQL database*

---

### Description

Create a new PostgreSQL database

### Usage

.create\_postgres\_db(config)

### Arguments

config        Connection configuration from settings.yml

### Value

TRUE if successful

---

```
.create_project_config  
Create project config.yml
```

---

### Description

Create project config.yml

### Usage

```
.create_project_config(  
  project_dir,  
  name,  
  type,  
  author,  
  packages,  
  directories,  
  extraDirectories,  
  ai,  
  git,  
  scaffold,  
  settings_dir,  
  connections = NULL,  
  connections_file = NULL,  
  render_dirs = NULL,  
  quarto = NULL  
)
```

---

```
.create_project_directories  
Create project subdirectories
```

---

### Description

Create project subdirectories

### Usage

```
.create_project_directories(  
  project_dir,  
  directories,  
  extraDirectories,  
  render_dirs = NULL  
)
```

---

`.create_renvignore` *Create .renvignore file for Framework projects*

---

### Description

Creates a .renvignore file that excludes Framework data and output directories from renv dependency scanning.

### Usage

```
.create_renvignore()
```

### Value

Invisibly returns NULL

---

`.create_rproj_file` *Create .Rproj file*

---

### Description

Create .Rproj file

### Usage

```
.create_rproj_file(project_dir, name)
```

---

`.create_scaffold_file`  
*Create scaffold.R file*

---

### Description

Create scaffold.R file

### Usage

```
.create_scaffold_file(project_dir, scaffold)
```

---

`.create_sqlite_db` *Create a new SQLite database*

---

**Description**

Create a new SQLite database

**Usage**

`.create_sqlite_db(config)`

**Arguments**

`config` Connection configuration from `settings.yml`

**Value**

TRUE if successful

---

`.create_sqlserver_db` *Create a new SQL Server database*

---

**Description**

Create a new SQL Server database

**Usage**

`.create_sqlserver_db(config)`

**Arguments**

`config` Connection configuration from `settings.yml`

**Value**

TRUE if successful

---

`.create_stub_files` *Create stub files for specific project types*

---

**Description**

Create stub files for specific project types

**Usage**

`.create_stub_files(project_dir, type, name, author)`

---

.create\_template\_db

*Create the template SQLite database*

---

## Description

Create the template SQLite database

## Usage

.create\_template\_db(delete\_existing = FALSE)

---

.create\_vscode\_settings

*Create VS Code settings.json*

---

## Description

Create VS Code settings.json

## Usage

.create\_vscode\_settings(target\_dir = ".", python = FALSE)

## Arguments

target\_dir    Target directory (defaults to current)  
python        Include Python configuration

---

.create\_vscode\_workspace

*Create VS Code workspace file*

---

## Description

Create VS Code workspace file

## Usage

.create\_vscode\_workspace(project\_name, target\_dir = ".", python = FALSE)

## Arguments

project\_name   Project name  
target\_dir     Target directory (defaults to current)  
python         Include Python configuration

---

`.customize_project_files`

*Customize project files with user-specific substitutions*

---

## Description

Customize project files with user-specific substitutions

## Usage

```
.customize_project_files(
    target_dir,
    author_name = NULL,
    author_email = NULL,
    author_affiliation = NULL
)
```

---

`.data_spec_update`    *Update data spec in the correct YAML file*

---

## Description

Traverses a dot-notated key like "final.public.test" and updates or inserts the given spec in the corresponding YAML file (either embedded in settings.yml or in an external settings/data.yml file). Automatically handles nested paths and creates intermediate structures as needed.

## Usage

```
.data_spec_update(path, spec)
```

## Arguments

path	Dot notation key (e.g., "final.public.test") indicating where to place the spec in the configuration hierarchy
spec	A named list containing the data spec fields (path, type, delimiter, locked, encrypted, etc.)

## Value

Invisibly returns NULL. Function is called for its side effect of updating the YAML configuration file.

---

```
.default_directory_table  
Get default directory table for project type
```

---

**Description**

Get default directory table for project type

**Usage**

```
.default_directory_table(project_type)
```

---

```
.default_render_dirs_for_type  
Resolve default render directory mappings for a project type from the catalog
```

---

**Description**

Resolve default render directory mappings for a project type from the catalog

**Usage**

```
.default_render_dirs_for_type(type)
```

---

```
.default_root_render_dir_for_type  
Resolve default root render_dir (if defined) for a project type
```

---

**Description**

Resolve default root render\_dir (if defined) for a project type

**Usage**

```
.default_root_render_dir_for_type(type)
```

---

```
.delete_init_file Delete init.R after successful initialization
```

---

**Description**

Delete init.R after successful initialization

**Usage**

```
.delete_init_file(subdir = NULL)
```

`.df_to_list_of_lists`

*Convert data frame to list of lists for YAML serialization JSON arrays of objects become data frames in R, but YAML needs list of lists*

**Description**

Convert data frame to list of lists for YAML serialization JSON arrays of objects become data frames in R, but YAML needs list of lists

**Usage**

```
.df_to_list_of_lists(df)
```

`.display_next_steps`

*Display next steps after initialization*

**Description**

Display next steps after initialization

**Usage**

```
.display_next_steps(project_name = NULL, type = "project", use_renv = FALSE)
```

`.ensure_framework_db`

*Ensure framework database exists with all required tables*

**Description**

Ensure framework database exists with all required tables

**Usage**

```
.ensure_framework_db(project_root = NULL)
```

**Arguments**

`project_root` Optional project root used to resolve the database path.

---

.ensure\_framework\_template

*Ensure the requested template exists in the user config directory*

---

## Description

Ensure the requested template exists in the user config directory

## Usage

.ensure\_framework\_template(name)

---

.ensure\_output\_dir *Ensure a directory exists, creating it lazily with feedback*

---

## Description

Ensure a directory exists, creating it lazily with feedback

## Usage

.ensure\_output\_dir(dir\_path, dir\_type = "output")

## Arguments

dir\_path      The directory path to ensure exists

dir\_type      Human-readable type for messaging (e.g., "tables", "figures")

## Value

The directory path (invisibly)

---

.find\_project\_root *Find project root by walking up directory tree*

---

## Description

Find project root by walking up directory tree

## Usage

.find\_project\_root(start\_dir)

## Arguments

start\_dir      Starting directory for search

## Value

Path to project root, or NULL if not found

---

```
.find_stub_template  
Find Stub Template
```

---

### Description

Searches for stub templates in user stubs/ directory first, then framework inst/stubs/ directory.

### Usage

```
.find_stub_template(stub, ext)
```

### Arguments

stub	Character. Stub name (e.g., "default", "analysis")
ext	Character. File extension ("qmd", "Rmd", or "R")

### Value

Path to stub template file

---

```
.framework_catalog_default_path  
Get the path to the packaged settings catalog YAML
```

---

### Description

Get the path to the packaged settings catalog YAML

### Usage

```
.framework_catalog_default_path()
```

---

```
.framework_catalog_user_path  
Get the path to the user-editable settings catalog override
```

---

### Description

Get the path to the user-editable settings catalog override

### Usage

```
.framework_catalog_user_path()
```

---

```
.framework_templates_dir
```

*Get the Framework templates directory*

---

### Description

Get the Framework templates directory

### Usage

```
.framework_templates_dir(...)
```

---

```
.generate_data_section
```

*Generate Data Management section*

---

### Description

Generate Data Management section

### Usage

```
.generate_data_section(config, project_type)
```

---

```
.generate_environment_section
```

*Generate Framework Environment section*

---

### Description

Generate Framework Environment section

### Usage

```
.generate_environment_section(config, project_type)
```

---

```
.generate_function_reference
```

*Generate Function Reference section*

---

### Description

Generate Function Reference section

### Usage

```
.generate_function_reference()
```

---

```
.generate_header_section  
Generate header section
```

---

**Description**

Generate header section

**Usage**

```
.generate_header_section(project_name)
```

---

```
.generate_hook_script  
Generate pre-commit hook script
```

---

**Description**

Generate pre-commit hook script

**Usage**

```
.generate_hook_script(  
    ai_sync_enabled,  
    data_security_enabled,  
    check_sensitive_dirs_enabled  
)
```

---

```
.generate_notes_section  
Generate notes section (user-editable)
```

---

**Description**

Generate notes section (user-editable)

**Usage**

```
.generate_notes_section()
```

---

```
.generate_packages_section
```

*Generate Installed Packages section*

---

### Description

Generate Installed Packages section

### Usage

```
.generate_packages_section(config)
```

---

```
.generate_project_type_section
```

*Generate project-type specific section*

---

### Description

Generate project-type specific section

### Usage

```
.generate_project_type_section(project_type)
```

---

```
.get_cache
```

*Get a cache value*

---

### Description

Get a cache value

### Usage

```
.get_cache(name, file = NULL, expire_after = NULL)
```

### Arguments

name

The cache name

file

Optional file path to store the cache (default: cache/{name}.rds)

expire\_after

Optional expiration time in hours (default: from config)

### Value

The cached result, or NULL if not found, expired, or hash mismatch

`.get_cache_dir`      *Get the cache directory, respecting FW\_CACHE\_DIR environment variable*

### Description

Get the cache directory, respecting FW\_CACHE\_DIR environment variable

### Usage

```
.get_cache_dir()
```

### Value

The cache directory path

`.get_data_directories`      *Get data directories from config*

### Description

Get data directories from config

### Usage

```
.get_data_directories(config, verbose)
```

`.get_data_pathSuggestions`      *Get suggestions for available data paths*

### Description

Helper function that extracts all available data paths from config. Used to provide helpful suggestions when data\_read() fails.

### Usage

```
.get_data_pathSuggestions(attempted_path = NULL)
```

### Arguments

`attempted_path`

The path that the user tried (optional, for future fuzzy matching)

### Value

Character vector of available data paths

---

.get\_data\_record    *Get a data value*

---

### Description

Get a data value

### Usage

.get\_data\_record(name)

### Arguments

name                  The data name

### Value

The data metadata (hash), or NULL if not found

---

.get\_db\_connection    *Get a connection to the framework database*

---

### Description

Get a connection to the framework database

### Usage

.get\_db\_connection(project\_root = NULL)

### Arguments

project\_root    Optional project root used to resolve the database path.

---

.get\_driver\_info    *Get driver information for a given database type*

---

### Description

Internal helper to map database driver names to their R packages and human-readable names.

### Usage

.get\_driver\_info(driver)

### Arguments

driver                  Character. Database driver name (e.g., "postgres", "mysql", "sqlite")

**Value**

Named list with package, name, and optionally `install_command`

**Examples**

```
## Not run:
.get_driver_info("postgres")
.get_driver_info("mysql")

## End(Not run)
```

`.get_example_data_path`

*Get example data path based on config*

**Description**

Get example data path based on config

**Usage**

```
.get_example_data_path(dirs, project_type, path_type)
```

`.get_hook_setting`

*Get git hook setting with optional alias fallback*

**Description**

Get git hook setting with optional alias fallback

**Usage**

```
.get_hook_setting(key, alias = NULL, config_file = NULL, default = FALSE)
```

`.get_metadata`

*Get a metadata value*

**Description**

Get a metadata value

**Usage**

```
.get_metadata(key, project_root = NULL)
```

**Arguments**

`key` The metadata key

`project_root` Optional project root for database resolution

**Value**

The metadata value, or NULL if not found

---

```
.get_notebook_dir_from_config
```

*Get Notebook Directory from Config*

---

**Description**

Reads config to determine where notebooks should be created. Falls back to "notebooks", "work", or current directory if config unavailable.

**Usage**

```
.get_notebook_dir_from_config()
```

**Value**

Character path to notebook directory

---

```
.get_package_list_from_config
```

*Extract package list from config*

---

**Description**

Handles two config structures:

1. New: packages = list(use\_renv = ..., default\_packages = ...)
2. Old: packages as a flat list of package specs

**Usage**

```
.get_package_list_from_config(config)
```

**Arguments**

<code>config</code>	Configuration object from <code>settings_read()</code>
---------------------	--

**Value**

List of package specifications, or empty list if none

`.get_package_requirements`  
*Get package requirements from config*

### Description

Get package requirements from config

### Usage

`.get_package_requirements(config)`

### Arguments

<code>config</code>	Configuration object from <code>settings_read()</code>
---------------------	--

`.get_placeholders` *Generate database-appropriate parameter placeholders*

### Description

Generate database-appropriate parameter placeholders

### Usage

`.get_placeholders(conn, n)`

### Arguments

<code>conn</code>	Database connection
<code>n</code>	Number of placeholders needed

### Value

Character vector of placeholders

`.get_scaffold_history`  
*Retrieve scaffold metadata from the database*

### Description

Retrieve scaffold metadata from the database

### Usage

`.get_scaffold_history(project_root = NULL)`

---

.get\_settings\_file *Get settings file path*

---

## Description

Returns path to settings.yml (preferred) or config.yml (backward compatibility). Walks up directory tree to find project root if not found in current directory.

## Usage

```
.get_settings_file(path = ".")
```

## Arguments

path	Optional path to check in (default: current directory)
------	--

## Value

Path to settings file, or NULL if neither exists

---

.git\_available      *Check if git is available on the system*

---

## Description

Check if git is available on the system

## Usage

```
.git_available()
```

## Value

TRUE if git is available, FALSE otherwise

---

```
.guess_content_type  
    Guess content type from file extension
```

---

### Description

Guess content type from file extension

### Usage

```
.guess_content_type(file)
```

### Arguments

file            Character. File path.

### Value

Character. MIME type.

---

```
.has_settings_file Check if settings file exists
```

---

### Description

Checks for settings.yml (preferred) or settings.yaml (backward compatibility).

### Usage

```
.has_settings_file(path = ".")
```

### Arguments

path            Optional path to check in (default: current directory)

### Value

TRUE if either file exists, FALSE otherwise

---

```
.identify_private_dirs  
    Identify directories that should be treated as private/sensitive
```

---

### Description

Identify directories that should be treated as private/sensitive

### Usage

```
.identify_private_dirs(data_dirs)
```

---

.init_db	<i>Initialize the framework database</i>
----------	--

---

### Description

Initialize the framework database

### Usage

.init\_db()

---

.init_git_repo	<i>Initialize git repository</i>
----------------	----------------------------------

---

### Description

Initialize git repository

Initialize git repository

### Usage

.init\_git\_repo(project\_dir, hooks)

.init\_git\_repo(project\_dir, hooks)

---

.init_renv	<i>Initialize renv</i>
------------	------------------------

---

### Description

Initialize renv

### Usage

.init\_renv(project\_dir)

---

`.init_standard`      *Standard initialization process (shared by both paths)*

---

## Description

Standard initialization process (shared by both paths)

## Usage

```
.init_standard(  
    project_name,  
    type,  
    lintr,  
    author_name = NULL,  
    author_email = NULL,  
    author_affiliation = NULL,  
    default_notebook_format = NULL,  
    subdir,  
    force,  
    use_git = TRUE  
)
```

---

`.install_package`      *Install a package if not already installed*

---

## Description

Install a package if not already installed

## Usage

```
.install_package(pkg_spec)
```

## Arguments

`pkg_spec`      Package specification (may include version pin)

---

.install\_package\_base  
*Install package without renv*

---

## Description

Installs a package using base R functions, handling version pinning and GitHub sources.

## Usage

```
.install_package_base(spec)
```

## Arguments

spec Parsed package specification from .parse\_package\_spec()

## Value

Invisibly returns TRUE on success

---

.install\_package\_renv  
*Install package via renv*

---

## Description

Installs a package using renv, handling version pinning and GitHub sources.

## Usage

```
.install_package_renv(spec)
```

## Arguments

spec Parsed package specification from .parse\_package\_spec()

## Value

Invisibly returns TRUE on success

---

```
.install_required_packages  
Install required packages from config
```

---

**Description**

Install required packages from config

**Usage**

```
.install_required_packages(config)
```

**Arguments**

config	Configuration object from settings_read()
--------	---

---

```
.is_data_file      Check if file is a data file based on extension
```

---

**Description**

Check if file is a data file based on extension

**Usage**

```
.is_data_file(file, extensions)
```

---

```
.is_git_repo      Check if current directory is a git repository
```

---

**Description**

Check if current directory is a git repository

**Usage**

```
.is_git_repo()
```

---

.is\_initialized      *Check if project is initialized*

---

### Description

Checks for existence of settings.yml/settings.yaml to determine initialization status.

### Usage

```
.is_initialized(subdir = NULL)
```

### Arguments

subdir      Optional subdirectory to check.

### Value

Logical indicating if project is initialized.

---

.list\_available\_stubs  
List Available Stub Templates

---

### Description

List Available Stub Templates

### Usage

```
.list_available_stubs(ext = NULL)
```

### Arguments

ext      Character. File extension to filter by

### Value

Character vector of stub names

---

```
.load_ai_template  Load AI context template for a project type
```

---

**Description**

Load AI context template for a project type

**Usage**

```
.load_ai_template(project_type, project_name = "My Project")
```

**Arguments**

project\_type Project type

project\_name Project name for placeholder substitution

**Value**

Character string with template content

---

```
.load_configuration
Load configuration from settings file
```

---

**Description**

Load configuration from settings file

**Usage**

```
.load_configuration(config_file = NULL)
```

---

```
.load_environment  Load environment variables from .env file
```

---

**Description**

Load environment variables from .env file

**Usage**

```
.load_environment(config_file = NULL, project_root = NULL)
```

---

.load\_functions     *Load all R files from functions directories*

---

### Description

Load all R files from functions directories

### Usage

```
.load_functions(config_file = NULL, project_root = NULL)
```

---

.load\_libraries     *Load all libraries specified in config*

---

### Description

Load all libraries specified in config

### Usage

```
.load_libraries(config)
```

### Arguments

config     Configuration object from settings\_read()

---

.load\_template\_content  
    *Load template content from inst/templates*

---

### Description

Load template content from inst/templates

### Usage

```
.load_template_content(template_name)
```

`.make_env` *Create or append to .env file*

## Description

Creates a .env file (if it doesn't exist) and appends environment variables. Warns if .env is not in .gitignore to prevent accidental secret exposure.

## Usage

```
.make_env(..., comment = NULL, check_gitignore = TRUE)
```

## Arguments

...	Named arguments for environment variables (e.g., DB_PASSWORD = "secret")
comment	Optional comment to add before the variables
check_gitignore	Logical; if TRUE (default), warns if .env not gitignored

## Value

Invisibly returns TRUE on success

## Examples

```
## Not run:
# Create .env with database credentials
.make_env(
  DB_HOST = "localhost",
  DB_PORT = "5432",
  DB_PASSWORD = "secret",
  comment = "Database connection"
)

# Add API keys
.make_env(
  OPENAI_API_KEY = "sk-...",
  comment = "API credentials"
)

## End(Not run)
```

`.mark_scaffolded` *Mark project as scaffolded*

## Description

Mark project as scaffolded

### Usage

```
.mark_scaffolded(project_root = NULL)
```

### Arguments

project\_root Optional project root where the scaffold marker should be written. Falls back to the current working directory when NULL.

---

.migrate\_config\_from\_previous

*Migrate config from previous R versions or legacy location*

---

### Description

Checks for configs in legacy `~/.config/framework` location and copies them to the current `R_user_dir` location if found.

### Usage

```
.migrate_config_from_previous(target_dir)
```

### Arguments

target\_dir Target config directory

### Value

TRUE if migration occurred, FALSE otherwise

---

.normalize\_expire\_after

*Normalize expire\_after input to numeric hours*

---

### Description

Normalize `expire_after` input to numeric hours

### Usage

```
.normalize_expire_after(expire_after, default = NULL)
```

### Arguments

expire\_after Numeric or character (e.g., "1 day", "12 hours")

default Default value if `expire_after` is NULL/empty

---

```
.normalize_notebook_name
```

*Normalize Notebook Name and Detect Type*

---

## Description

Normalize Notebook Name and Detect Type

## Usage

```
.normalize_notebook_name(name, type = c("quarto", "rmarkdown", "script"))
```

## Arguments

name	Character. File name with or without extension
type	Character. Type preference

## Value

List with name, type, and ext

---

```
.normalize_package_spec
```

*Normalize package specification from config*

---

## Description

Converts the various package representations supported in settings.yml into a consistent structure that downstream helpers can rely on.

## Usage

```
.normalize_package_spec(spec)
```

## Arguments

spec	Character string or list describing a package dependency
------	--

## Value

List with fields: name, source, version, repo, ref, auto\_attach

---

```
.parse_assistant_selection  
Parse Assistant Selection from User Input
```

---

## Description

Helper to convert user input like "1,3" or "4" into assistant names.

## Usage

```
.parse_assistant_selection(selection)
```

## Arguments

selection      User input string

## Value

Character vector of assistant names

---

```
.parse_package_spec  
Parse package specification with source detection
```

---

## Description

Parses package specifications that may include explicit sources, version pins, or GitHub/Bioconductor references. Supports both scalar strings and list-style entries from `settings.yml`.

## Usage

```
.parse_package_spec(spec)
```

## Arguments

spec      Character or list describing the package

## Details

Examples:

- "dplyr" -> list(name = "dplyr", source = "cran")
- "dplyr@1.1.0" -> list(name = "dplyr", version = "1.1.0", source = "cran")
- "tidyverse/dplyr@main" -> list(name = "dplyr", repo = "tidyverse/dplyr", ref = "main", source = "github")
- list(name = "DESeq2", source = "bioc") -> list(name = "DESeq2", source = "bioc")

## Value

List with normalized components (name, source, version, repo, ref, auto\_attach)

## Examples

```
## Not run:
.parse_package_spec("dplyr")
.parse_package_spec("dplyr@1.1.0")
.parse_package_spec("tidyverse/dplyr@main")
.parse_package_spec(list(name = "DESeq2", source = "bioc", auto_attach = FALSE))

## End(Not run)
```

---

`.parse_sections`     *Parse markdown into sections based on ## headings*

---

## Description

Parse markdown into sections based on ## headings

## Usage

```
.parse_sections(content)
```

---

`.pretty_print_config`     *Pretty-print configuration values (internal)*

---

## Description

Intelligently formats configuration output based on session type. In interactive sessions, pretty-prints nested structures. In scripts, returns raw values.

## Usage

```
.pretty_print_config(value)
```

## Arguments

<code>value</code>	Configuration value to format
--------------------	-------------------------------

## Value

In interactive mode with complex values: invisible return after printing. Otherwise: raw value.

---

`.print_audit_summary`     *Print audit summary*

---

## Description

Print audit summary

## Usage

```
.print_audit_summary(result)
```

---

.prompt\_ai\_support\_init

*Get AI Support Preferences (Non-interactive)*

---

### Description

Called during project\_create() to check if AI instructions should be created. NO prompting - just returns saved preferences.

### Usage

```
.prompt_ai_support_init()
```

---

.prompt\_ai\_support\_install

*Set AI Support Preferences (Non-interactive)*

---

### Description

Called from bash CLI to set AI preferences. NO prompting - bash handles all user interaction.

### Usage

```
.prompt_ai_support_install(support = "never", assistants = character(0))
```

### Arguments

support        "yes" or "never"

assistants     Character vector like c("claude", "copilot")

---

.read\_framework\_template

*Read a framework template*

---

### Description

Read a framework template

### Usage

```
.read_framework_template(name)
```

---

`.remove_cache`      *Remove a cache value*

---

**Description**

Remove a cache value

**Usage**

`.remove_cache(name, file = NULL)`

**Arguments**

<code>name</code>	The cache name to remove
<code>file</code>	Optional file path of the cache (default: <code>cache/{name}.rds</code> )

---

`.remove_data`      *Remove a data value*

---

**Description**

Remove a data value

**Usage**

`.remove_data(name)`

**Arguments**

<code>name</code>	The data name to remove
-------------------	-------------------------

---

`.remove_init`      *Remove initialization*

---

**Description**

Removes settings.yml/settings.yaml to mark project as uninitialized. WARNING: This will delete your project configuration!

**Usage**

`.remove_init(subdir = NULL)`

**Arguments**

<code>subdir</code>	Optional subdirectory to check.
---------------------	---------------------------------

**Value**

Logical indicating if removal was successful.

---

`.remove_metadata` *Remove a metadata value*

---

### Description

Remove a metadata value

### Usage

```
.remove_metadata(key, project_root = NULL)
```

### Arguments

`key` The metadata key to remove

`project_root` Optional project root for database resolution

---

`.replace_moustache_placeholders`

*Replace moustache-style placeholders in template content*

---

### Description

Supports both `{}{name}` and `{} { name }` styles by trimming whitespace around the variable identifier before replacement.

### Usage

```
.replace_moustache_placeholders(content, replacements)
```

### Arguments

`content` Character vector containing template lines.

`replacements` Named list or vector of replacements.

### Value

Character vector with placeholders replaced.

---

`.replace_section` *Replace content of a specific section*

---

### Description

Replace content of a specific section

### Usage

```
.replace_section(content, heading, new_content)
```

---

<code>.require_driver</code>	<i>Check if a database driver package is available</i>
------------------------------	--

---

## Description

Internal helper to check if a required database driver package is installed. Throws an informative error if the package is missing.

## Usage

```
.require_driver(driver_name, package_name, install_command = NULL)
```

## Arguments

<code>driver_name</code>	Character. Human-readable name of the database (e.g., "PostgreSQL", "MySQL")
<code>package_name</code>	Character. Name of the R package required (e.g., "RPostgres", "RMariaDB")
<code>install_command</code>	Character. Optional custom install command. Defaults to <code>install.packages()</code>

## Value

`NULL` (invisible). Throws error if package not available.

## Examples

```
## Not run:
.require_driver("PostgreSQL", "RPostgres")
.require_driver("MySQL", "RMariaDB")

## End(Not run)
```

---

<code>.require_git_repo</code>	<i>Require git repository or stop</i>
--------------------------------	---------------------------------------

---

## Description

Require git repository or stop

## Usage

```
.require_git_repo()
```

---

.reset\_framework\_template  
*Reset a template back to its packaged default*

---

### Description

Reset a template back to its packaged default

### Usage

.reset\_framework\_template(name)

---

.s3\_client                   *Create an S3 client from connection configuration*

---

### Description

Creates an S3 client object using credentials from the connection configuration. Credentials are resolved from connection config, falling back to environment variables. Loads .env file if present to ensure env vars are available.

### Usage

.s3\_client(conn\_config)

### Arguments

conn\_config   List. Connection configuration from config.yml

### Value

An S3 client object (from aws.s3 package)

---

.s3\_public\_url           *Generate public URL for S3 object*

---

### Description

Generate public URL for S3 object

### Usage

.s3\_public\_url(key, s3\_config)

### Arguments

key                   Character. Object key in S3.  
s3\_config           List. S3 configuration.

**Value**

Character. Public URL.

---

`.s3_upload_dir`      *Upload a directory to S3*

---

**Description**

Recursively uploads all files in a directory to S3.

**Usage**

```
.s3_upload_dir(dir, dest, s3_config, pattern = NULL)
```

**Arguments**

<code>dir</code>	Character. Local directory path to upload.
<code>dest</code>	Character. Destination prefix in S3 bucket.
<code>s3_config</code>	List. S3 configuration from <code>.resolve_s3_connection()</code> .
<code>pattern</code>	Character or NULL. Optional file pattern filter.

**Value**

Character vector. S3 URIs of uploaded files.

---

`.s3_upload_file`      *Upload a file to S3*

---

**Description**

Uploads a single file to an S3 bucket.

**Usage**

```
.s3_upload_file(file, dest, s3_config, content_type = NULL)
```

**Arguments**

<code>file</code>	Character. Local file path to upload.
<code>dest</code>	Character. Destination key (path) in S3 bucket.
<code>s3_config</code>	List. S3 configuration from <code>.resolve_s3_connection()</code> .
<code>content_type</code>	Character or NULL. MIME type (auto-detected if NULL).

**Value**

Character. The S3 URI of the uploaded file.

---

`.save_audit_result` *Save audit result to framework database*

---

### Description

Save audit result to framework database

### Usage

```
.save_audit_result(result)
```

---

`.save_result` *Log a saved result to the framework database*

---

### Description

Internal function called by `save_table()`, `save_figure()`, etc. to track saved outputs in the results table.

### Usage

```
.save_result(name, path, type, public = FALSE, comment = NULL)
```

### Arguments

<code>name</code>	Result name/identifier (typically the filename)
<code>path</code>	Full file path to the saved result
<code>type</code>	Result type: "table", "figure", "model", "report", "notebook"
<code>public</code>	Whether saved to public outputs directory
<code>comment</code>	Optional description

### Value

`NULL` invisibly

---

`.scan_for_orphaned_files`  
*Scan for orphaned data files outside configured directories*

---

### Description

Scan for orphaned data files outside configured directories

### Usage

```
.scan_for_orphaned_files(data_dirs, extensions, verbose)
```

`.set_cache`*Set a cache value***Description**

Set a cache value

**Usage**

```
.set_cache(name, value, file = NULL, expire_after = NULL)
```

**Arguments**

<code>name</code>	The cache name
<code>value</code>	The value to cache
<code>file</code>	Optional file path to store the cache (default: <code>cache/{name}.rds</code> )
<code>expire_after</code>	Optional expiration time in hours (default: from config)

`.set_data`*Set a data value***Description**

Set a data value

**Usage**

```
.set_data(
  name,
  path = NULL,
  type = NULL,
  delimiter = NULL,
  locked = FALSE,
  hash = NULL
)
```

**Arguments**

<code>name</code>	The data name
<code>path</code>	The file path
<code>type</code>	The data type (csv, rds, etc.)
<code>delimiter</code>	The delimiter for CSV files
<code>locked</code>	Whether the data is locked
<code>hash</code>	The hash of the data

---

`.set_ggplot_theme` *Set ggplot2 theme for consistent styling*

---

## Description

Sets ggplot2 theme if configured. Checks for theme settings in this order:

1. Project settings.yml (ggplot\_theme and set\_theme\_on\_scaffold)
2. Skip if set\_theme\_on\_scaffold is FALSE or theme is empty

## Usage

```
.set_ggplot_theme(config)
```

## Arguments

<code>config</code>	Configuration object from <code>settings_read()</code>
---------------------	--

---

`.set_metadata` *Set a metadata value*

---

## Description

Set a metadata value

## Usage

```
.set_metadata(key, value, project_root = NULL)
```

## Arguments

<code>key</code>	The metadata key
<code>value</code>	The metadata value
<code>project_root</code>	Optional project root for database resolution

`.set_random_seed`     *Set random seed for reproducibility*

## Description

Sets the random seed for reproducibility. Checks for seed in this order:

1. Project settings.yml (seed: value)
2. Global ~/.frameworkrc (FW\_SEED)
3. Skip seeding if both are NULL or empty

## Usage

`.set_random_seed(config)`

## Arguments

<code>config</code>	Configuration object from <code>settings_read()</code>
---------------------	--

`.slugify`     *Slugify a String*

## Description

Converts a string to a filesystem-safe slug:

- Converts to lowercase
- Replaces spaces and special characters with hyphens
- Removes consecutive hyphens
- Trims leading/trailing hyphens

## Usage

`.slugify(text)`

## Arguments

<code>text</code>	Character. String to slugify
-------------------	------------------------------

## Value

Character. Slugified string

---

.sync\_packages\_to\_renv  
*Sync packages from settings.yml to renv*

---

### Description

Reads the packages list from settings.yml and installs them via renv, then snapshots the result to renv.lock.

### Usage

.sync\_packages\_to\_renv()

### Value

Invisibly returns TRUE on success

---

.to\_kebab\_case      *Convert string to kebab-case*

---

### Description

Convert string to kebab-case

### Usage

.to\_kebab\_case(str)

---

.update\_data\_with\_hash  
*Update data with hash in the data table*

---

### Description

Update data with hash in the data table

Update data with hash in the data table

### Usage

.update\_data\_with\_hash(name, hash)  
.update\_data\_with\_hash(name, hash)

### Arguments

name	The data name
hash	The hash to store

---

`.update_frameworkrc`

*Update ~/.frameworkrc with AI Preferences*

---

## Description

Helper to update the frameworkrc file with AI support settings.

## Usage

```
.update_frameworkrc(frameworkrc_path, support, assistants)
```

## Arguments

frameworkrc_path	Path to .frameworkrc file
support	"yes", "never", or ""
assistants	Character vector of assistant names

---

`.update_gitignore_for_renv`

*Update .gitignore for renv*

---

## Description

Adds renv-related entries to .gitignore if they don't already exist.

## Usage

```
.update_gitignore_for_renv()
```

## Value

Invisibly returns NULL

---

`.update_hook_config`

*Update hook configuration in settings.yml/settings.yaml*

---

## Description

Update hook configuration in settings.yml/settings.yaml

## Usage

```
.update_hook_config(hook_name, enabled, config_file)
```

---

`.uses_split_file`    *Determine if a project field uses split file or inline settings*

---

### Description

Determine if a project field uses split file or inline settings

### Usage

```
.uses_split_file(project_path, field_name)
```

### Arguments

`project_path` Path to project directory

`field_name` Name of the field to check (e.g., "packages", "connections")

### Value

List with `use_split`, `main_file`, `split_file`, and `has_default`

---

`.validate_driver`    *Validate driver availability before connection*

---

### Description

Internal helper that combines driver info lookup and availability check. Used by connection functions to ensure required packages are installed.

### Usage

```
.validate_driver(driver)
```

### Arguments

`driver`    Character. Database driver name

### Value

NULL (invisible). Throws error if driver package not available.

### Examples

```
## Not run:  
.validate_driver("postgres")  
.validate_driver("mysql")  
  
## End(Not run)
```

---

```
.validate_refresh  Validate refresh parameter
```

---

### Description

Validate refresh parameter

### Usage

```
.validate_refresh(refresh)
```

### Arguments

refresh	Boolean or function that returns boolean
---------	--

### Value

Boolean indicating if refresh is needed

---

```
.validate_settings_catalog
Basic validation for the settings catalog structure
```

---

### Description

Basic validation for the settings catalog structure

### Usage

```
.validate_settings_catalog(catalog)
```

---

```
.write_framework_template
Write (overwrite) a framework template
```

---

### Description

Write (overwrite) a framework template

### Usage

```
.write_framework_template(name, contents)
```

---

```
add_project_to_config
```

*Add project to global configuration*

---

### Description

Add project to global configuration

### Usage

```
add_project_to_config(project_dir, project_name = NULL, project_type = NULL)
```

### Arguments

project\_dir Path to project directory

project\_name Optional project name

project\_type Optional project type

### Value

Invisibly returns the project ID

---

---

```
ai_generate_context
```

*Generate AI Context File*

---

### Description

Generates a complete AI context file (CLAUDE.md, AGENTS.md, etc.) from scratch for a new project. The content is tailored to the project type and configuration.

### Usage

```
ai_generate_context(  
    project_path = ".",  
    project_name = NULL,  
    project_type = NULL,  
    config = NULL  
)
```

### Arguments

project\_path Path to the project directory (default: current directory)

project\_name Name of the project (for header)

project\_type Project type: "project", "project\_sensitive", "course", "presentation"

config Project configuration (if NULL, reads from settings.yml)

**Value**

Character string with the complete AI context content

**Examples**

```
## Not run:
# Generate AI context for current project
content <- ai_generate_context()

# Generate for a specific project type
content <- ai_generate_context(project_type = "project_sensitive")

## End(Not run)
```

**ai\_regenerate\_context**

*Regenerate Dynamic Sections in AI Context File*

**Description**

Updates only the sections marked with `<!-- @framework:regenerate -->` in an existing AI context file, preserving user customizations in unmarked sections.

**Usage**

```
ai_regenerate_context(project_path = ".", sections = NULL, ai_file = NULL)
```

**Arguments**

project_path	Path to the project directory
sections	Which sections to regenerate. <code>NULL</code> = all regeneratable sections. Options: "environment", "packages", "data", "functions"
ai_file	Name of the AI context file (default: from settings or "CLAUDE.md")

**Value**

Invisible TRUE on success

**Examples**

```
## Not run:
# Regenerate all dynamic sections
ai_regenerate_context()

# Regenerate only packages section
ai_regenerate_context(sections = "packages")

## End(Not run)
```

---

ai\_sync\_context      *Sync AI Assistant Context Files*

---

## Description

Copies content from the canonical AI assistant file to all other AI files, adding a warning header to non-canonical files.

## Usage

```
ai_sync_context(config_file = NULL, force = FALSE, verbose = TRUE)
```

## Arguments

config_file	Path to configuration file (default: auto-detect settings.yml/settings.yaml)
force	Logical; if TRUE, overwrite even if target is newer (default: FALSE)
verbose	Logical; if TRUE (default), show sync messages

## Details

This function reads the `ai.canonical_file` setting from `settings.yml` and copies its content to all other AI assistant instruction files that exist in the project.

The canonical file is copied as-is. Non-canonical files receive a warning header indicating they are auto-synced and should not be edited directly.

Supported files:

- **AGENTS.md** - Cross-platform AI assistants
- **CLAUDE.md** - Claude Code
- **.github/copilot-instructions.md** - GitHub Copilot

## Value

Invisible list with sync results

## Examples

```
## Not run:  
# Sync AI context files  
ai_sync_context()  
  
# Force sync even if targets are newer  
ai_sync_context(force = TRUE)  
  
# Silent sync (for git hooks)  
ai_sync_context(verbose = FALSE)  
  
## End(Not run)
```

`bootstrap_project_init`  
*Bootstrap project initialization file*

## Description

Generates an init.R file showing the initialization logic. Useful for documentation and understanding how the project was set up.

## Usage

```
bootstrap_project_init(output_file = "init.R")
```

## Arguments

`output_file` Path where init.R should be written. Default: "init.R"

## Value

Invisibly returns TRUE on success

## Examples

```
## Not run:  

# Create init.R to see initialization logic  

bootstrap_project_init()  

  

# Write to a different location  

bootstrap_project_init("docs/init_reference.R")  

  

## End(Not run)
```

`cache` *Cache a value*

## Description

Cache a value

## Usage

```
cache(name, value, file = NULL, expire_after = NULL)
```

## Arguments

<code>name</code>	The cache name
<code>value</code>	The value to cache
<code>file</code>	Optional file path to store the cache (default: <code>cache/{name}.rds</code> )
<code>expire_after</code>	Optional expiration time in hours (default: from config)

**Value**

The cached value

---

cache_flush	<i>Clear all cached values</i>
-------------	--------------------------------

---

**Description**

Clear all cached values

**Usage**

```
cache_flush()
```

---

cache_forget	<i>Remove a cached value</i>
--------------	------------------------------

---

**Description**

Remove a cached value

**Usage**

```
cache_forget(name, file = NULL)
```

**Arguments**

name	The cache name to remove
file	Optional file path of the cache (default: cache/{name}.rds)

---

cache_get	<i>Get a cached value</i>
-----------	---------------------------

---

**Description**

Get a cached value

**Usage**

```
cache_get(name, file = NULL, expire_after = NULL)
```

**Arguments**

name	The cache name
file	Optional file path to store the cache (default: cache/{name}.rds)
expire_after	Optional expiration time in hours (default: from config)

**Value**

The cached value, or NULL if not found, expired, or hash mismatch

<code>cache_list</code>	<i>List all cached values</i>
-------------------------	-------------------------------

## Description

Returns a data frame of all cache entries with their names, expiration times, and status (expired or active).

## Usage

```
cache_list()
```

## Value

A data frame with columns:

- name** Cache key name
- expire\_at** Expiration timestamp (NA if no expiration)
- created\_at** When the cache was created
- updated\_at** When the cache was last updated
- last\_read\_at** When the cache was last read
- status** Either "active" or "expired"

Returns an empty data frame if no cache entries exist.

## Examples

```
## Not run:
# List all cache entries
cache_list()

# Filter to see only expired caches
cache_list() |> dplyr::filter(status == "expired")

## End(Not run)
```

<code>cache_remember</code>	<i>Remember a value (get from cache or compute and store)</i>
-----------------------------	---

## Description

Attempts to retrieve a cached value by name. If the cache doesn't exist, is expired, or a refresh is requested, evaluates the expression and caches the result. This is the primary caching interface for expensive computations.

## Usage

```
cache_remember(
  name,
  expr,
  file = NULL,
  expire_after = NULL,
  refresh = FALSE,
  expire = NULL
)
```

## Arguments

<code>name</code>	The cache name (non-empty string identifier)
<code>expr</code>	The expression to evaluate and cache if cache miss occurs. Expression is evaluated in the parent frame.
<code>file</code>	Optional file path to store the cache (default: <code>cache/{name}.rds</code> )
<code>expire_after</code>	Optional expiration time in hours (default: from config). Character durations like "1 day" or "2 hours" are accepted.
<code>refresh</code>	Optional boolean or function that returns boolean to force refresh. If TRUE or if function returns TRUE, cache is invalidated and expression is re-evaluated.
<code>expire</code>	Optional alias for <code>expire_after</code> (accepts the same formats)

## Value

The cached value (if cache hit) or the result of evaluating `expr` (if cache miss or refresh requested)

## Examples

```
## Not run:
# Cache expensive computation
result <- cache_remember("my_analysis", {
  expensive_computation()
})

# Force refresh when data changes
result <- cache_remember("analysis", {
  run_analysis()
}, refresh = file.mtime("data.csv") > cache_time)

## End(Not run)
```

## Description

Capture console output and errors from an expression

**Usage**

```
capture_output(expr)
```

**Arguments**

<code>expr</code>	Expression to evaluate
-------------------	------------------------

**Value**

List containing status (boolean), console\_output (character vector), and result (return value or error)

<code>config</code>	<i>Get configuration value (alias for settings)</i>
---------------------	---

**Description**

Internal alias for `settings`.

**Usage**

```
config(key = NULL, default = NULL, config_file = NULL)
```

**Arguments**

<code>key</code>	Character. Dot-notation key path
<code>default</code>	Optional default value if key is not found
<code>config_file</code>	Configuration file path (default: auto-discover)

**Value**

The configuration value

<code>configure_ai_agents</code>	<i>Configure AI Assistant Support</i>
----------------------------------	---------------------------------------

**Description**

Non-interactive function to configure AI assistant support. Should be called from bash CLI with parameters, not directly from R.

**Usage**

```
configure_ai_agents(support = NULL, assistants = NULL)
```

**Arguments**

<code>support</code>	"yes", "never", or NULL (show current status)
<code>assistants</code>	Character vector of assistants: "claude", "copilot", "agents"

## Details

Supported AI assistants:

- **Claude Code:** Creates CLAUDE.md in project root
- **GitHub Copilot:** Creates .github/copilot-instructions.md
- **AGENTS.md:** Creates AGENTS.md (cross-platform, industry standard)

## Examples

```
## Not run:  
# Enable AI support with Claude and Copilot  
configure_ai_agents(support = "yes", assistants = c("claude", "copilot"))  
  
# Disable AI support  
configure_ai_agents(support = "never")  
  
# Show current status  
configure_ai_agents()  
  
## End(Not run)
```

---

configure\_author    *Configure Author Information*

---

## Description

Interactively set author information in settings.yml (or settings.yml for legacy projects). This information is used in notebooks, reports, and other documents.

## Usage

```
configure_author(  
  name = NULL,  
  email = NULL,  
  affiliation = NULL,  
  interactive = TRUE  
)
```

## Arguments

name	Character. Author name (optional, prompts if not provided)
email	Character. Author email (optional, prompts if not provided)
affiliation	Character. Author affiliation/institution (optional, prompts if not provided)
interactive	Logical. If TRUE, prompts for missing values. Default TRUE.

## Value

Invisibly returns updated config

## Examples

```
## Not run:
# Interactive mode (prompts for all fields)
configure_author()

# Provide values directly
configure_author(
  name = "Jane Doe",
  email = "jane@example.com",
  affiliation = "University of Example"
)

## End(Not run)
```

### `configure_connection`

*Configure Database Connection*

## Description

Interactively add a database connection to settings.yml (or settings.yml for legacy projects). Connections can be defined inline or in a split file (settings/connections.yml).

## Usage

```
configure_connection(
  name = NULL,
  driver = NULL,
  host = NULL,
  port = NULL,
  database = NULL,
  user = NULL,
  password = NULL,
  interactive = TRUE
)
```

## Arguments

<code>name</code>	Character. Connection name (e.g., "db", "warehouse")
<code>driver</code>	Character. Database driver: "sqlite", "postgresql", "mysql", etc.
<code>host</code>	Character. Database host (for network databases)
<code>port</code>	Integer. Database port (for network databases)
<code>database</code>	Character. Database name
<code>user</code>	Character. Database user (for network databases)
<code>password</code>	Character. Database password (stored in .env)
<code>interactive</code>	Logical. If TRUE, prompts for missing values. Default TRUE.

## Value

Invisibly returns updated config

## Examples

```
## Not run:
# Interactive mode
configure_connection()

# SQLite connection
configure_connection(
  name = "mydb",
  driver = "sqlite",
  database = "data/mydb.db"
)

# PostgreSQL connection
configure_connection(
  name = "warehouse",
  driver = "postgresql",
  host = "localhost",
  port = 5432,
  database = "analytics",
  user = "analyst"
)

## End(Not run)
```

<code>configure_data</code>	<i>Configure Data Source</i>
-----------------------------	------------------------------

## Description

Interactively add a data source to settings.yml (or settings.yml for legacy projects). Data sources are defined with dot-notation paths (e.g., "source.private.survey") and include metadata like file path, type, and whether the data is locked.

## Usage

```
configure_data(
  path = NULL,
  file = NULL,
  type = NULL,
  locked = FALSE,
  interactive = TRUE
)
```

## Arguments

<code>path</code>	Character. Dot-notation path for the data source (e.g., "source.private.survey")
<code>file</code>	Character. File path to the data file
<code>type</code>	Character. Data type: "csv", "tsv", "rds", "excel", "stata", "spss", "sas", or "auto"
<code>locked</code>	Logical. If TRUE, file is read-only and errors on changes
<code>interactive</code>	Logical. If TRUE, prompts for missing values. Default TRUE.

**Value**

Invisibly returns updated config

**Examples**

```
## Not run:
# Interactive mode
configure_data()

# Provide values directly
configure_data(
  path = "source.private.survey",
  file = "inputs/raw/survey.csv",
  type = "csv",
  locked = TRUE
)

## End(Not run)
```

**configure\_directories**

*Configure Project Directories*

**Description**

Interactively configure project directory structure in settings.yml (or settings.yaml for legacy projects). Directories control where Framework creates and looks for files.

**Usage**

```
configure_directories(directory = NULL, path = NULL, interactive = TRUE)
```

**Arguments**

directory	Character. Directory name to configure (e.g., "notebooks", "scripts")
path	Character. Path for the directory
interactive	Logical. If TRUE, prompts for missing values. Default TRUE.

**Details****Standard Directories:**

- notebooks - Where make\_notebook() creates files
- scripts - Where make\_script() creates files
- functions - Where scaffold() looks for custom functions
- inputs\_raw - Source data (gitignored)
- inputs\_intermediate - Cleaned-but-input datasets
- inputs\_final - Curated analytic datasets
- inputs\_reference - External documentation/codebooks
- outputs\_private - Working artifacts (tables/figures/models)

- outputs\_public - Share-ready artifacts
- outputs\_docs - Narrative/report outputs (private)
- outputs\_docs\_public - Narrative/report outputs (public)
- cache - Cached computation results
- scratch - Temporary workspace

**Value**

Invisibly returns updated config

**Examples**

```
## Not run:
# Interactive mode
configure_directories()

# Set specific directory
configure_directories(
  directory = "notebooks",
  path = "analysis"
)

## End(Not run)
```

**configure\_global     *Configure Global Framework Settings*****Description**

Unified function for reading and writing global Framework settings to `~/.frameworkrc.json`. This function provides a single source of truth for global configuration, used by both the CLI and GUI interfaces.

**Usage**

```
configure_global(settings = NULL, validate = TRUE)
```

**Arguments**

settings	List. Settings to update (partial updates supported)
validate	Logical. Validate settings before saving (default: TRUE)

**Details****Global Settings Structure:**

- author - Author information (name, email, affiliation)
- defaults - Project defaults
  - project\_type - Default project type ("project", "presentation", "course")
  - notebook\_format - Default notebook format ("quarto", "rmarkdown")
  - ide - IDE preference ("vscode", "rstudio", "both", "none")

- use\_git - Initialize git repositories by default
- use\_renv - Enable renv by default
- seed - Default random seed
- seed\_on\_scaffold - Set seed during scaffold()
- ai\_support - Enable AI assistant support
- ai\_assistants - List of AI assistants ("claude", "agents", etc.)
- ai\_canonical\_file - Canonical AI instruction file
- packages - Default package list
- directories - Default directory structure
- git\_hooks - Git hook preferences
- projects - Registered projects list
- active\_project - Currently active project path

### Value

Invisibly returns updated global configuration

### Examples

```
## Not run:
# Update author information
configure_global(settings = list(
  author = list(
    name = "Jane Doe",
    email = "jane@example.com"
  )
))

# Update default project type
configure_global(settings = list(
  defaults = list(
    project_type = "presentation"
  )
))

# Get current settings (read-only)
current <- configure_global()

## End(Not run)
```

`configure_packages` *Configure Package Dependencies*

### Description

Interactively add package dependencies to `settings.yml` (or `settings.yaml` for legacy projects). Packages can be installed from CRAN, GitHub, or Bioconductor, with version pinning support.

**Usage**

```
configure_packages(  
  package = NULL,  
  auto_attach = TRUE,  
  version = NULL,  
  interactive = TRUE  
)
```

**Arguments**

package	Character. Package name (e.g., "dplyr", "tidyverse/dplyr")
auto_attach	Logical. If TRUE, package is loaded automatically during scaffold()
version	Character. Version constraint (e.g., "@1.1.0", "@main" for GitHub)
interactive	Logical. If TRUE, prompts for missing values. Default TRUE.

**Details****Package Specifications:**

- CRAN: "dplyr", "ggplot2"
- CRAN with version: "dplyr@1.1.0"
- GitHub: "tidyverse/dplyr", "user/repo@branch"
- GitHub with tag: "user/repo@v1.2.3"

**Value**

Invisibly returns updated config

**Examples**

```
## Not run:  
# Interactive mode  
configure_packages()  
  
# Add CRAN package with auto-attach  
configure_packages(  
  package = "dplyr",  
  auto_attach = TRUE  
)  
  
# Add GitHub package  
configure_packages(  
  package = "tidyverse/dplyr@main",  
  auto_attach = FALSE  
)  
  
## End(Not run)
```

---

`connections_list`    *List all connections (databases and object storage)*

---

**Description**

Prints both database connections defined under `connections:` and object storage profiles (S3-compatible buckets). Use this to see everything Framework can talk to from your config.

**Usage**

```
connections_list()
```

**Value**

Invisibly returns `NULL` after printing summaries.

---

`connections_s3`    *S3 Connection Functions*

---

**Description**

Functions for connecting to and interacting with S3-compatible storage.

---

`connection_begin`    *Begin a database transaction*

---

**Description**

Manually begin a database transaction. You must call `connection_commit()` to save changes or `connection_rollback()` to discard them.

**Usage**

```
connection_begin(conn)
```

**Arguments**

`conn`              Database connection

**Details**

**Note:** Using `db_transaction()` is preferred as it automatically handles commit/rollback.

**Value**

`NULL` (invisible)

---

connection\_check     *Check if a connection is ready to use*

---

### Description

Diagnoses whether a configured database connection can be established. Checks driver availability and configuration validity without actually connecting to the database.

### Usage

```
connection_check(connection_name)
```

### Arguments

connection\_name  
Character. Name of the connection in config.yml

### Value

A list with diagnostic information:

- ready: Logical. TRUE if connection appears ready
  - driver: Driver name
  - package: Required package
  - package\_installed: Whether package is available
  - config\_valid: Whether configuration appears valid
  - messages: Character vector of diagnostic messages
- 

---

connection\_check\_leaks  
Check for leaked database connections

---

### Description

Scans the global environment and parent frames for open database connections. Useful for debugging connection leaks in interactive sessions or long-running scripts.

### Usage

```
connection_check_leaks(warn = TRUE)
```

### Arguments

warn              Logical. If TRUE (default), emits a warning if leaked connections found

### Value

A data frame with information about open connections:

- object\_name: Name of the variable holding the connection
- class: Connection class (e.g., "PqConnection", "SQLiteConnection")
- valid: Whether connection is still valid

`connection_close_all`

*Close all open database connections*

### Description

Safely closes all open database connections in the global environment. Useful for cleaning up after interactive sessions or when resetting state.

### Usage

```
connection_close_all(force = FALSE, quiet = FALSE)
```

### Arguments

<code>force</code>	Logical. If TRUE, closes even invalid connections. Default: FALSE
<code>quiet</code>	Logical. If TRUE, suppresses messages. Default: FALSE

### Value

Invisibly returns the number of connections closed

`connection_commit` *Commit a database transaction*

### Description

Commits the current transaction, making all changes permanent.

### Usage

```
connection_commit(conn)
```

### Arguments

<code>conn</code>	Database connection
-------------------	---------------------

### Value

NULL (invisible)

---

```
connection_delete  Delete a record from a table
```

---

### Description

Deletes a record from a table. Supports soft-delete pattern where records have a deleted\_at column. Hard-delete can be forced with soft = FALSE.

### Usage

```
connection_delete(conn, table_name, id, soft = TRUE)
```

### Arguments

conn	Database connection
table_name	Name of the table
id	The ID of the record to delete
soft	Whether to use soft-delete if available (default: TRUE)

### Value

Number of rows affected

---

```
connection_find  Find a record by ID
```

---

### Description

Finds a single record in a table by its ID. Supports soft-delete patterns where records have a deleted\_at column.

### Usage

```
connection_find(conn, table_name, id, with_trashed = FALSE)
```

### Arguments

conn	Database connection
table_name	Name of the table to query
id	The ID to look up (integer or string)
with_trashed	Whether to include soft-deleted records (default: FALSE). Only applies if deleted_at column exists in the table.

### Value

A data frame with the record, or empty data frame if not found

## Examples

```
## Not run:
conn <- db_connect("postgres")
user <- connection_find(conn, "users", 42)
DBI::dbDisconnect(conn)

## End(Not run)
```

*connection\_find\_by* *Find records by column values*

## Description

Finds records in a table matching specified column values. Supports soft-delete patterns where records have a deleted\_at column.

## Usage

```
connection_find_by(conn, table_name, ..., with_trashed = FALSE)
```

## Arguments

conn	Database connection
table_name	Name of the table to query
...	Named arguments for column = value pairs (e.g., email = "test@example.com")
with_trashed	Whether to include soft-deleted records (default: FALSE). Only applies if deleted_at column exists in the table.

## Value

A data frame with matching records, or empty data frame if none found

*connection\_insert* *Insert a record into a table*

## Description

Inserts a new record into a table with automatic timestamp handling. If the table has created\_at/updated\_at columns, they will be set automatically.

## Usage

```
connection_insert(conn, table_name, values, auto_timestamps = TRUE)
```

### Arguments

conn	Database connection
table_name	Name of the table
values	Named list of column-value pairs
auto_timestamps	Whether to automatically set created_at/updated_at (default: TRUE)

### Value

The ID of the inserted record (if auto-increment ID exists), or number of rows affected

connection\_pool     *Get or create a connection pool*

### Description

Returns a connection pool for the specified database connection. Connection pools automatically manage connection lifecycle, reuse connections across operations, and handle cleanup. This is the recommended way to work with databases in Framework.

### Usage

```
connection_pool (
    name,
    min_size = 1,
    max_size = Inf,
    idle_timeout = 60,
    validation_interval = 60,
    recreate = FALSE
)
```

### Arguments

name	Character. Name of the connection in settings.yml
min_size	Integer. Minimum number of connections to maintain (default: 1)
max_size	Integer. Maximum number of connections allowed (default: Inf)
idle_timeout	Integer. Seconds before idle connections are closed (default: 60)
validation_interval	Integer. Seconds between connection health checks (default: 60)
recreate	Logical. If TRUE, closes existing pool and creates new one (default: FALSE)

### Details

**Connection pools are stored in a package environment and reused across calls.** You don't need to manage pool lifecycle - Framework handles it automatically.

#### Advantages of connection pools:

- Automatic connection reuse (faster than creating new connections)
- Handles connection failures gracefully (auto-reconnects)

- Thread-safe for Shiny apps
- No need to manually disconnect
- Health checking prevents using stale connections

**When to use:**

- Long-running R sessions (notebooks, Shiny apps)
- Multiple database operations
- Any production code

**When NOT to use:**

- One-off queries (use `query_get()` instead)
- Short scripts (overhead not worth it)

**Value**

A pool object that can be used like a regular DBI connection

**Examples**

```
## Not run:
# Get a pool (reuses existing pool if already created)
pool <- connection_pool("my_db")

# Use like a regular connection
users <- DBI::dbGetQuery(pool, "SELECT * FROM users")

# No need to disconnect - pool manages connections automatically

# Multiple operations reuse connections
result <- connection_with_pool("my_db", {
  users <- DBI::dbGetQuery(pool, "SELECT * FROM users")
  posts <- DBI::dbGetQuery(pool, "SELECT * FROM posts")
  list(users = users, posts = posts)
})

# Clean up all pools when done (optional)
connection_pool_close_all()

## End(Not run)
```

***connection\_pool\_close***

*Close a specific connection pool*

**Description**

Closes and removes a connection pool. All connections in the pool are gracefully closed.

**Usage**

```
connection_pool_close(name, quiet = FALSE)
```

**Arguments**

name	Character. Name of the connection pool to close
quiet	Logical. If TRUE, suppresses messages (default: FALSE)

**Value**

Invisibly returns TRUE if pool was closed, FALSE if it didn't exist

**Examples**

```
## Not run:  
connection_pool_close("my_db")  
  
## End(Not run)
```

---

```
connection_pool_close_all  
Close all connection pools
```

---

**Description**

Closes all active connection pools. Useful for cleanup when shutting down R sessions or resetting state.

**Usage**

```
connection_pool_close_all(quiet = FALSE)
```

**Arguments**

quiet	Logical. If TRUE, suppresses messages (default: FALSE)
-------	--

**Value**

Invisibly returns the number of pools closed

**Examples**

```
## Not run:  
# Close all pools  
connection_pool_close_all()  
  
# Quiet mode  
connection_pool_close_all(quiet = TRUE)  
  
## End(Not run)
```

---

`connection_pool_list`

*List active connection pools*

---

## Description

Shows all currently active connection pools with their status.

## Usage

```
connection_pool_list()
```

## Value

A data frame with pool information:

- name: Pool name
- valid: Whether pool is valid
- connections: Number of active connections (if available)

## Examples

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

---

`connection_restore` *Restore a soft-deleted record*

---

## Description

Restores a soft-deleted record by setting deleted\_at to NULL. Only works on tables with a deleted\_at column.

## Usage

```
connection_restore(conn, table_name, id)
```

## Arguments

<code>conn</code>	Database connection
<code>table_name</code>	Name of the table
<code>id</code>	The ID of the record to restore

## Value

Number of rows affected

---

```
connection_rollback
```

*Rollback a database transaction*

---

### Description

Rolls back the current transaction, discarding all changes.

### Usage

```
connection_rollback(conn)
```

### Arguments

conn	Database connection
------	---------------------

### Value

NULL (invisible)
------------------

---

```
connection_update  Update a record in a table
```

---

### Description

Updates an existing record in a table with automatic timestamp handling. If the table has an updated\_at column, it will be set automatically.

### Usage

```
connection_update(conn, table_name, id, values, auto_timestamps = TRUE)
```

### Arguments

conn	Database connection
------	---------------------

table_name	Name of the table
------------	-------------------

id	The ID of the record to update
----	--------------------------------

values	Named list of column-value pairs to update
--------	--

auto_timestamps	Whether to automatically set updated_at (default: TRUE)
-----------------	---

### Value

Number of rows affected
-------------------------

`connection_with_pool`

*Execute code with a connection pool*

## Description

Convenience wrapper for working with connection pools. Gets or creates a pool and makes it available as `pool` within the code block.

## Usage

```
connection_with_pool(connection_name, code, ...)
```

## Arguments

<code>connection_name</code>	Character. Name of the connection in <code>settings.yml</code>
<code>code</code>	Expression to evaluate with the pool
<code>...</code>	Additional arguments passed to <code>connection_pool()</code>

## Value

The result of evaluating `code`

## Examples

```
## Not run:
# Simple usage
users <- connection_with_pool("my_db", {
  DBI::dbGetQuery(pool, "SELECT * FROM users WHERE active = TRUE")
})

# Multiple operations
result <- connection_with_pool("my_db", {
  users <- DBI::dbGetQuery(pool, "SELECT * FROM users")
  posts <- DBI::dbGetQuery(pool, "SELECT * FROM posts")
  list(users = users, posts = posts)
})

## End(Not run)
```

`connection_with_transaction`

*Execute code with transaction if not already in one*

## Description

Similar to `db_transaction()`, but only starts a new transaction if not already in one. Useful for functions that can be called both standalone and within an existing transaction.

**Usage**

```
connection_with_transaction(conn, code)
```

**Arguments**

conn	Database connection
code	Expression or code block to execute

**Value**

The result of the code expression

---

data_add	<i>Add an existing file to the data catalog</i>
----------	---

---

**Description**

Registers an existing data file with the Framework data catalog. This allows you to track files that were created outside of Framework (e.g., downloaded from external sources, copied from other projects) and use them with `data_read()` using dot notation.

**Usage**

```
data_add(  
    file_path,  
    name = NULL,  
    type = NULL,  
    delimiter = "comma",  
    locked = TRUE,  
    update_config = TRUE  
)
```

**Arguments**

file_path	Path to the existing file (must exist)
name	Optional dot notation name for the data catalog (e.g., <code>inputs.raw.survey_data</code> ). If NULL, derives name from file path relative to project root.
type	Optional type override. Auto-detected from file extension if NULL.
delimiter	Delimiter for CSV files ("comma", "tab", "semicolon", "space")
locked	Whether the file should be locked (hash-verified on read)
update_config	If TRUE (default), also updates the YAML config with the data spec

**Value**

Invisibly returns the data spec that was created

## Examples

```
## Not run:
# Add a downloaded CSV file to the catalog
data_add("inputs/raw/survey_results.csv", name = "inputs.raw.survey_results")

# Now you can read it with dot notation
data_read("inputs.raw.survey_results")

# Add with auto-generated name
data_add("inputs/intermediate/cleaned_data.rds")
# Name will be derived as "inputs.intermediate.cleaned_data"

## End(Not run)
```

*data\_info*

*Get data specification from config*

## Description

Gets the data specification for a given dot notation path from settings.yml. Supports dot notation (e.g., "source.private.example"), relative paths, and absolute paths. Auto-detects file type from extension and applies intelligent defaults for common formats.

## Usage

`data_info(path)`

## Arguments

<code>path</code>	Dot notation path (e.g. "source.private.example"), relative path, or absolute path to a data file
-------------------	---

## Value

A list with data specification including:

- `path` - Full file path
- `type` - File type (csv, rds, stata, spss, sas, etc.)
- `delimiter` - Delimiter for CSV files (comma, tab, etc.)
- `locked` - Whether file is locked for integrity checking
- `private` - Whether file is in private data directory
- `description` - Optional description of the dataset (displayed when loading)

**Examples**

```
## Not run:  
# Get info from dot notation  
info <- data_info("source.private.my_data")  
  
# Get info from file path  
info <- data_info("data/public/example.csv")  
  
## End(Not run)
```

---

data\_list

*List all data entries from config*

---

**Description**

Lists all data specifications defined in the configuration, showing the data key, path, type, and description (if available).

**Usage**

```
data_list()
```

**Value**

A data frame with columns: name, path, type, locked, description

**Examples**

```
## Not run:  
# List all data entries  
data_list()  
  
# Use the alias  
list_data()  
  
## End(Not run)
```

---

data\_read

*Read data using dot notation path or direct file path*

---

**Description**

Supports CSV, TSV, RDS, Excel (.xlsx, .xls), Stata (.dta), SPSS (.sav, .zsav, .por), and SAS (.sas7bdat, .xpt) file formats.

**Usage**

```
data_read(path, delim = NULL, keep_attributes = FALSE, ...)
```

**Arguments**

<code>path</code>	Dot notation path (e.g. "source.private.example") or direct file path
<code>delim</code>	Optional delimiter for CSV files ("comma", "tab", "semicolon", "space")
<code>keep_attributes</code>	Logical flag to preserve special attributes (e.g., haven labels). Default: FALSE (strips attributes)
<code>...</code>	Additional arguments passed to read functions (readr::read_delim, readxl::read_excel, haven::read_*, etc.)

`data_read_or_cache` *Read data with caching (DEPRECATED)*

**Description****[Deprecated]**

This function is deprecated. Use `cache_remember()` with `data_read()` instead:

```
df <- cache_remember("my_data", data_read("source.private.example"))
```

**Usage**

```
data_read_or_cache(path, expire_after = NULL, refresh = FALSE)
```

**Arguments**

<code>path</code>	Dot notation path to load data (e.g. "source.private.example")
<code>expire_after</code>	Optional expiration time in hours (default: from config)
<code>refresh</code>	Optional boolean or function that returns boolean to force refresh

**Value**

The loaded data, either from cache or file

`data_save` *Save data using dot notation or file path*

**Description**

Save data using dot notation or file path

**Usage**

```
data_save(
  data,
  path,
  type = NULL,
  delimiter = "comma",
  locked = TRUE,
  force = FALSE
)
```

**Arguments**

data	Data frame to save
path	Either: <ul style="list-style-type: none"><li>• Dot notation: <code>inputs.raw.filename</code> resolves to <code>inputs/raw/filename.rds</code></li><li>• Direct path: "<code>inputs/raw/filename.csv</code>" uses path as-is</li></ul> Dot notation uses your configured directories (e.g., <code>inputs.raw</code> , <code>inputs.intermediate</code> , <code>outputs.private</code> ).
type	Type of data file ("csv" or "rds"). Auto-detected from extension if path includes one.
delimiter	Delimiter for CSV files ("comma", "tab", "semicolon", "space")
locked	Whether the file should be locked after saving
force	If TRUE, creates missing directories. If FALSE (default), errors if directory doesn't exist.

---

`data_spec_get`*Get data specification (DEPRECATED)***Description****[Deprecated]**

This function has been renamed to `data_info()`. Please use that instead.

**Usage**

```
data_spec_get(path)
```

**Arguments**

path	Dot notation path or file path
------	--------------------------------

**Value**

A list with data specification

---

`db_connect`*Get a database connection***Description**

Gets a database connection based on the connection name in config.yml. For most use cases, prefer `db_query()` or `db_execute()` which handle connection lifecycle automatically.

**Usage**

```
db_connect(name)
```

**Arguments**

name	Character. Name of the connection in config.yml (e.g., "postgres")
------	--

**Value**

A database connection object (DBIConnection)

**Examples**

```
## Not run:
# Preferred: use db_query() which auto-disconnects
users <- db_query("SELECT * FROM users", "postgres")

# Manual connection management (remember to disconnect!)
conn <- db_connect("postgres")
DBI::dbListTables(conn)
DBI::dbDisconnect(conn)

## End(Not run)
```

*db\_drivers\_install Install database drivers***Description**

Interactive helper to install one or more database drivers. Provides helpful instructions and handles special cases (like ODBC).

**Usage**

```
db_drivers_install(drivers = NULL, repos =getOption("repos"))
```

**Arguments**

drivers	Character vector. Database driver names to install (e.g., "postgres", "mysql", "duckdb"). If NULL, shows interactive menu.
repos	Character. CRAN repository URL. Default:getOption("repos")

**Value**

NULL (invisible). Installs packages as side effect.

**Examples**

```
## Not run:
# Install specific drivers
db_drivers_install(c("postgres", "mysql"))

# Interactive mode
db_drivers_install()

## End(Not run)
```

---

```
db_drivers_status  Check if database drivers are installed
```

---

## Description

Checks which database drivers are currently available on the system. Returns a data frame showing the status of all supported database drivers.

## Usage

```
db_drivers_status(quiet = FALSE)
```

## Arguments

quiet            Logical. If TRUE, suppresses messages. Default: FALSE

## Value

A data frame with columns:

- driver: Database driver name
- package: Required R package
- installed: Whether the package is installed
- version: Package version (if installed)

## Examples

```
## Not run:  
# Check all drivers  
db_drivers_status()  
  
# Quiet mode (no messages)  
db_drivers_status(quiet = TRUE)  
  
## End(Not run)
```

---

```
db_execute  Execute a database statement
```

---

## Description

Executes a SQL statement on a database without returning results. The connection is created, used, and automatically closed.

## Usage

```
db_execute(query, connection_name, ...)
```

**Arguments**

query	SQL statement to execute
connection_name	Name of the connection in config.yml
...	Additional arguments passed to DBI::dbExecute

**Value**

Number of rows affected

**Examples**

```
## Not run:
rows <- db_execute("DELETE FROM cache WHERE expired = TRUE", "my_db")

## End(Not run)
```

**db\_list**

*List all database connections from configuration*

**Description**

Lists all database connections defined in the configuration, showing the connection name, driver, host, and database name (if applicable).

**Usage**

```
db_list()
```

**Value**

Invisibly returns NULL after printing connection list

**Examples**

```
## Not run:
# List all connections
db_list()

## End(Not run)
```

---

db_query	<i>Get data from a database query</i>
----------	---------------------------------------

---

## Description

Gets data from a database using a query and connection name. The connection is created, used, and automatically closed.

## Usage

```
db_query(query, connection_name, ...)
```

## Arguments

query	SQL query to execute
connection_name	Name of the connection in config.yml
...	Additional arguments passed to DBI::dbGetQuery

## Value

A data frame with the query results

## Examples

```
## Not run:  
users <- db_query("SELECT * FROM users", "my_db")  
## End(Not run)
```

---

db_transaction	<i>Execute code within a database transaction</i>
----------------	---

---

## Description

Wraps code execution in a database transaction with automatic commit on success and rollback on error. This ensures atomicity of multiple database operations.

## Usage

```
db_transaction(conn, code)
```

## Arguments

conn	Database connection
code	Expression or code block to execute within the transaction

## Details

The function automatically:

- Begins a transaction with `DBI::dbBegin()`
- Executes the provided code
- Commits the transaction on success with `DBI::dbCommit()`
- Rolls back the transaction on error with `DBI::dbRollback()`

Transactions are essential for maintaining data integrity when performing multiple related operations. If any operation fails, all changes are rolled back.

## Value

The result of the code expression

## Examples

```
## Not run:
conn <- db_connect("postgres")

# Basic transaction
db_transaction(conn, {
  DBI::dbExecute(conn, "INSERT INTO users (name, age) VALUES ('Alice', 30)")
  DBI::dbExecute(conn, "INSERT INTO users (name, age) VALUES ('Bob', 25)")
})

# Transaction with error handling - auto-rollback on error
tryCatch({
  db_transaction(conn, {
    DBI::dbExecute(conn, "INSERT INTO users (name) VALUES ('Alice')")
    stop("Something went wrong") # This will trigger rollback
  })
}, error = function(e) {
  message("Transaction failed: ", e$message)
})

DBI::dbDisconnect(conn)

## End(Not run)
```

## *db\_with*

*Execute code with a managed database connection*

## Description

Provides automatic connection lifecycle management. The connection is automatically closed when the code block finishes, even if an error occurs. This prevents connection leaks and ensures proper resource cleanup.

## Usage

```
db_with(connection_name, code)
```

**Arguments**

connection_name	Character. Name of the connection in config.yml
code	Expression to evaluate with the connection (use conn to access the connection)

**Value**

The result of evaluating code

**Examples**

```
## Not run:
# Safe - connection auto-closes
users <- db_with("my_db", {
  DBI::dbGetQuery(conn, "SELECT * FROM users WHERE active = TRUE")
})

# Multiple operations with same connection
result <- db_with("my_db", {
  DBI::dbExecute(conn, "INSERT INTO users (name) VALUES ('Alice')")
  DBI::dbGetQuery(conn, "SELECT * FROM users")
})

# Connection closes even on error
tryCatch(
  db_with("my_db", {
    stop("Something went wrong") # Connection still closes
  }),
  error = function(e) message(e$message)
)

## End(Not run)
```

**Description**

Parses roxygen2-generated .Rd files and exports structured documentation to SQLite (for GUI) or other formats. This enables searchable documentation in the Framework GUI and powers the public documentation website.

**Usage**

```
docs_export(
  output_path = "docs.db",
  man_dir = "man",
  package_name = "framework",
  package_version = NULL,
  include_internal = FALSE,
  verbose = TRUE
)
```

## Arguments

```

output_path  Path to SQLite database file. Default: "docs.db"
man_dir      Directory containing .Rd files. Default: "man"
package_name Package name for metadata. Default: "framework"
package_version
              Package version for metadata. Default: NULL (auto-detect)
include_internal
              Include internal/non-exported functions. Default: FALSE
verbose       Print progress messages. Default: TRUE

```

## Details

The exporter reads all .Rd files from the man/ directory and extracts:

- Function name, title, description, details
- Arguments/parameters with descriptions
- Usage signatures
- Examples (with dontrun detection)
- See Also references
- Custom sections and subsections
- Keywords

The SQLite output includes FTS5 full-text search for fast querying.

## Value

Invisibly returns the database connection path

## Examples

```

## Not run:
# Export to default location (exported functions only)
docs_export()

# Export to custom location
docs_export("inst/gui/docs.db")

# Include internal/private functions too
docs_export("all_docs.db", include_internal = TRUE)

# Query the exported docs
con <- DBI::dbConnect(RSQLite::SQLite(), "docs.db")
DBI::dbGetQuery(con, "SELECT name, title FROM functions WHERE name LIKE 'data_%'")
DBI::dbDisconnect(con)

## End(Not run)

```

---

<code>dot-has_column</code>	<i>Check if a column exists in a table (S3 generic)</i>
-----------------------------	---

---

## Description

Cross-database method to check if a column exists in a table. Uses database-specific introspection methods via S3 dispatch.

## Usage

```
.has_column(conn, table_name, column_name)

## S3 method for class 'SQLiteConnection'
.has_column(conn, table_name, column_name)

## S3 method for class 'PqConnection'
.has_column(conn, table_name, column_name)

## S3 method for class 'MariaDBConnection'
.has_column(conn, table_name, column_name)

## S3 method for class ``Microsoft SQL Server``
.has_column(conn, table_name, column_name)

## S3 method for class 'duckdb_connection'
.has_column(conn, table_name, column_name)

## Default S3 method:
.has_column(conn, table_name, column_name)
```

## Arguments

<code>conn</code>	Database connection (DBIConnection)
<code>table_name</code>	Character. Name of the table
<code>column_name</code>	Character. Name of the column to check

## Value

Logical. TRUE if column exists, FALSE otherwise

## Functions

- `.has_column(SQLiteConnection)`: SQLite implementation using PRAGMA
- `.has_column(PqConnection)`: PostgreSQL implementation using information\_schema
- `.has_column(MariaDBConnection)`: MySQL/MariaDB implementation using information\_schema
- `.has_column(`Microsoft SQL Server`)`: SQL Server implementation using information\_schema
- `.has_column(duckdb_connection)`: DuckDB implementation using information\_schema
- `.has_column(default)`: Default implementation for unknown database types

## Examples

```
## Not run:
conn <- connection_get("my_db")
has_deleted_at <- .has_column(conn, "users", "deleted_at")
DBI::dbDisconnect(conn)

## End(Not run)
```

**dot-list\_columns**    *List all columns in a table (S3 generic)*

## Description

Cross-database method to list all columns in a table. Uses database-specific introspection methods via S3 dispatch.

## Usage

```
.list_columns(conn, table_name)

## S3 method for class 'SQLiteConnection'
.list_columns(conn, table_name)

## S3 method for class 'PqConnection'
.list_columns(conn, table_name)

## S3 method for class 'MariaDBConnection'
.list_columns(conn, table_name)

## S3 method for class ``Microsoft SQL Server``
.list_columns(conn, table_name)

## S3 method for class 'duckdb_connection'
.list_columns(conn, table_name)

## Default S3 method:
.list_columns(conn, table_name)
```

## Arguments

conn	Database connection (DBIConnection)
table_name	Character. Name of the table

## Value

Character vector of column names

## Functions

- `.list_columns(SQLiteConnection)`: SQLite implementation using PRAGMA
- `.list_columns(PqConnection)`: PostgreSQL implementation using information\_schema
- `.list_columns(MariaDBConnection)`: MySQL/MariaDB implementation using information\_schema
- `.list_columns(`Microsoft SQL Server`)`: SQL Server implementation using information\_schema
- `.list_columns(duckdb_connection)`: DuckDB implementation using information\_schema
- `.list_columns(default)`: Default implementation using information\_schema

## Examples

```
## Not run:
conn <- connection_get("my_db")
columns <- .list_columns(conn, "users")
DBI::dbDisconnect(conn)

## End(Not run)
```

`dot-list_tables`     *List all tables in a database (S3 generic)*

## Description

Cross-database method to list all tables. Uses database-specific methods via S3 dispatch.

## Usage

```
.list_tables(conn)

## Default S3 method:
.list_tables(conn)
```

## Arguments

`conn`     Database connection (DBIConnection)

## Value

Character vector of table names

## Functions

- `.list_tables(default)`: Default implementation using DBI::dbListTables

## Examples

```
## Not run:
conn <- connection_get("my_db")
tables <- .list_tables(conn)
DBI::dbDisconnect(conn)

## End(Not run)
```

`env_clear` *Clear R environment*

## Description

Cleans up the R environment by removing objects, closing plots, detaching packages, and running garbage collection. Does not clear the console.

## Usage

```
env_clear(keep = character())
```

## Arguments

<code>keep</code>	Character vector of object names to keep (default: empty)
-------------------	---

## Value

Invisibly returns NULL

## Examples

```
## Not run:
# Clean everything
env_clear()

# Keep specific objects
env_clear(keep = c("config", "data"))

## End(Not run)
```

`env_default_template_lines`  
*Default Framework .env template lines*

## Description

Provides the baseline .env content that ships with Framework. Other helper functions (`project_create()`, GUI scaffolders) reuse these lines when users haven't customized their own template.

## Usage

```
env_default_template_lines()
```

---

env\_lines\_from\_variables  
*Convert env() configuration into file lines*

---

**Description**

Convert env() configuration into file lines

**Usage**

```
env_lines_from_variables(vars)
```

---

env\_resolve\_lines *Resolve env template lines from configuration*

---

**Description**

Resolve env template lines from configuration

**Usage**

```
env_resolve_lines(env_config = NULL)
```

**Arguments**

env\_config Either a character string (raw .env content) or a list with raw and/or variables.

**Value**

Character vector of lines ready to be written to .env

---

env\_summary *Summarize R environment*

---

**Description**

Displays a summary of the current R environment including loaded packages, objects in the global environment, and memory usage.

**Usage**

```
env_summary()
```

**Value**

Invisibly returns a list with environment information

## Examples

```
## Not run:
env_summary()

## End(Not run)
```

`framework_template_path`

*Get the user-editable path for a Framework template*

## Description

Get the user-editable path for a Framework template

## Usage

```
framework_template_path(name)
```

## Arguments

<code>name</code>	Template identifier (e.g., "notebook", "gitignore", "ai_claude")
-------------------	--

## Value

Absolute path to the template file, ensuring it exists.

`framework_view`

*(Deprecated) Use view\_create() or view\_detail() instead*

## Description

### [Deprecated]

`framework_view()` was renamed to `view_create()` to follow the package's noun\_verb naming convention for better discoverability and consistency.

**Recommended:** Use `view_detail()` for the clearest, most user-friendly name.

## Usage

```
framework_view(x, title = NULL, max_rows = 5000)
```

## Arguments

<code>x</code>	The data to view (data.frame, plot, list, function, or other R object)
<code>title</code>	Optional title for the view. If <code>NULL</code> , uses the object name.
<code>max_rows</code>	Maximum number of rows to display for data frames (default: 5000). Large data frames are automatically truncated with a warning.

## Value

Opens a browser window (called for side effects)

---

fw\_config\_dir      *Get Framework config directory path*

---

**Description**

Returns the path to Framework's global configuration directory. Uses `tools::R_user_dir("framework", "config")` by default (CRAN compliant). Can be overridden with the FW\_CONFIG\_HOME environment variable.

**Usage**

```
fw_config_dir()
```

**Value**

Character string with the config directory path

---

get\_default\_global\_config  
  *Get default global configuration structure*

---

**Description**

Get default global configuration structure

**Usage**

```
get_default_global_config()
```

**Value**

List with default global configuration

---

get\_global\_setting *Get Global Configuration Setting*

---

**Description**

Retrieve a specific setting from the global configuration file (`~/.frameworkrc.json`). This is a helper function primarily for use by the CLI script.

**Usage**

```
get_global_setting(key, default = "", print = TRUE)
```

**Arguments**

key	Character. The setting key to retrieve (e.g., "defaults.ide", "author.name")
default	Character. Default value if setting is not found (default: "")
print	Logical. If TRUE, prints the value (for bash consumption). Default TRUE.

**Value**

The setting value as a character string

**Examples**

```
## Not run:
# Get IDE setting
get_global_setting("defaults.ide")

# Get with default value
get_global_setting("defaults.notebook_format", default = "quarto")

## End(Not run)
```

**git\_add***Stage Files for Commit***Description**

Add file contents to the staging area.

**Usage**

```
git_add(files = ".")
```

**Arguments**

files	Character vector of file paths to stage, or "." for all (default)
-------	---

**Value**

Invisibly returns TRUE on success

**Examples**

```
## Not run:
git_add()                      # Stage all changes
git_add("README.md")            # Stage specific file
git_add(c("R/foo.R", "R/bar.R"))

## End(Not run)
```

---

git_commit	<i>Commit Staged Changes</i>
------------	------------------------------

---

## Description

Record changes to the repository with a commit message.

## Usage

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

## Arguments

message	Commit message (required)
all	Logical; if TRUE, automatically stage modified/deleted files (default: FALSE)

## Value

Invisibly returns TRUE on success

## Examples

```
## Not run:  
git_commit("Fix bug in data loading")  
git_commit("Update README", all = TRUE) # Stage and commit  
  
## End(Not run)
```

---

git_diff	<i>Show Changes (Diff)</i>
----------	----------------------------

---

## Description

Show changes between commits, working tree, etc.

## Usage

```
git_diff(staged = FALSE, file = NULL)
```

## Arguments

staged	Logical; if TRUE, show staged changes (default: FALSE shows unstaged)
file	Optional file path to show diff for specific file

## Value

Invisibly returns the diff output as a character vector

## Examples

```
## Not run:
git_diff()           # Show unstaged changes
git_diff(staged = TRUE) # Show staged changes
git_diff(file = "R/foo.R")

## End(Not run)
```

**git\_hooks\_disable** *Disable Specific Git Hook*

## Description

Disables a specific hook in settings and reinstalls the pre-commit hook.

## Usage

```
git_hooks_disable(hook_name, config_file = NULL, verbose = TRUE)
```

## Arguments

hook_name	Name of hook: "ai_sync", "data_security", or "check_sensitive_dirs"
config_file	Path to configuration file (default: auto-discover settings.yml or settings.yaml)
verbose	Logical; if TRUE (default), show messages

## Value

Invisible TRUE on success

**git\_hooks\_enable** *Enable Specific Git Hook*

## Description

Enables a specific hook in settings and reinstalls the pre-commit hook.

## Usage

```
git_hooks_enable(hook_name, config_file = NULL, verbose = TRUE)
```

## Arguments

hook_name	Name of hook: "ai_sync", "data_security", or "check_sensitive_dirs"
config_file	Path to configuration file (default: auto-discover settings.yml or settings.yaml)
verbose	Logical; if TRUE (default), show messages

## Value

Invisible TRUE on success

## Examples

```
## Not run:  
git_hooks_enable("ai_sync")  
git_hooks_enable("data_security")  
  
## End(Not run)
```

---

git\_hooks\_install *Install Git Pre-commit Hook*

---

## Description

Creates a pre-commit hook that runs Framework checks based on settings.yml settings.

## Usage

```
git_hooks_install(config_file = NULL, force = FALSE, verbose = TRUE)
```

## Arguments

config_file	Path to configuration file (default: "settings.yml")
force	Logical; if TRUE, overwrite existing hook (default: FALSE)
verbose	Logical; if TRUE (default), show installation messages

## Details

Creates or updates .git/hooks/pre-commit to run enabled Framework hooks:

- **ai\_sync**: Sync AI assistant context files before commit
- **data\_security**: Run security audit to catch data leaks
- **check\_sensitive\_dirs**: Warn about unignored sensitive directories

Hook behavior is controlled by git.hooks.\* settings in settings.yml.

## Value

Invisible TRUE on success, FALSE on failure

## Examples

```
## Not run:  
# Install hooks based on settings.yml  
git_hooks_install()  
  
# Force reinstall (overwrites existing hook)  
git_hooks_install(force = TRUE)  
  
## End(Not run)
```

---

git\_hooks\_list      *List Git Hook Status*

---

**Description**

Shows which hooks are enabled and their current status.

**Usage**

```
git_hooks_list(config_file = NULL)
```

**Arguments**

config\_file Path to configuration file (default: auto-discover settings.yml or settings.yaml)

**Value**

Data frame with hook information

---

git\_hooks\_uninstall  
Uninstall Git Pre-commit Hook

---

**Description**

Removes the Framework-managed pre-commit hook.

**Usage**

```
git_hooks_uninstall(verbose = TRUE)
```

**Arguments**

verbose      Logical; if TRUE (default), show messages

**Value**

Invisible TRUE if hook was removed, FALSE otherwise

---

`git_log`*Show Commit Log*

---

**Description**

Show recent commit history.

**Usage**

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

**Arguments**

n	Number of commits to show (default: 10)
oneline	Logical; if TRUE, show condensed one-line format (default: TRUE)

**Value**

Invisibly returns the log output as a character vector

**Examples**

```
## Not run:  
git_log()  
git_log(n = 5)  
git_log(oneline = FALSE) # Full format  
  
## End(Not run)
```

---

`git_pull`*Pull from Remote*

---

**Description**

Fetch and integrate changes from the remote repository.

**Usage**

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

**Arguments**

remote	Remote name (default: "origin")
branch	Branch name (default: current branch)

**Value**

Invisibly returns TRUE on success

## Examples

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

## End(Not run)
```

`git_push`

*Push to Remote*

## Description

Push commits to the remote repository.

## Usage

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

## Arguments

<code>remote</code>	Remote name (default: "origin")
<code>branch</code>	Branch name (default: current branch)

## Value

Invisibly returns TRUE on success

## Examples

```
## Not run:
git_push()
git_push(remote = "origin", branch = "main")

## End(Not run)
```

`git_security_audit` *Security audit for Framework projects*

## Description

Performs a comprehensive security audit of data files in Framework projects, checking for unignored data files, git history leaks, and orphaned data files outside configured directories.

## Usage

```
git_security_audit(
  config_file = NULL,
  check_git_history = TRUE,
  history_depth = "all",
  auto_fix = FALSE,
  verbose = TRUE,
  extensions = c("csv", "rds", "tsv", "txt", "dat", "xlsx", "xls", "sqlite", "db",
    "sav", "zsav", "por", "sas7bdat", "sas7bcat", "xpt", "parquet", "feather",
    "json", "xml", "h5", "hdf5")
)
```

## Arguments

<code>config_file</code>	Path to configuration file (default: auto-detect settings.yml/settings.yaml)
<code>check_git_history</code>	Logical; if TRUE (default), check git history for leaked data files
<code>history_depth</code>	Character or numeric. "all" for full history, "shallow" for recent 100 commits, or numeric for specific commit count (default: "all")
<code>auto_fix</code>	Logical; if TRUE, automatically update .gitignore (default: FALSE)
<code>verbose</code>	Logical; if TRUE (default), show progress messages
<code>extensions</code>	Character vector of data file extensions to detect (default: common data formats)

## Details

The security audit performs the following checks:

- **gitignore\_coverage**: Verifies all private data files are in .gitignore
- **git\_history**: Scans git history for accidentally committed data files
- **orphaned\_files**: Finds data files outside configured directories
- **private\_data\_exposure**: Checks if private data is tracked by git

Status levels:

- **pass**: No issues found
- **warning**: Potential issues that should be reviewed
- **fail**: Critical security issues requiring immediate action

## Value

A structured list containing:

<code>summary</code>	Data frame with check names, status (pass/warning/fail), and counts
<code>findings</code>	List of data frames with detailed findings for each check
<code>recommendations</code>	Character vector of actionable recommendations
<code>audit_metadata</code>	List with audit timestamp, Framework version, and config info

## Examples

```
## Not run:
# Basic audit (report only)
audit <- git_security_audit()
print(audit$summary)
View(audit$findings$orphaned_files)

# Quick scan without git history
audit <- git_security_audit(check_git_history = FALSE)

# Verbose with limited git history
audit <- git_security_audit(history_depth = 100, verbose = TRUE)

# Auto-fix mode (updates .gitignore)
audit <- git_security_audit(auto_fix = TRUE)

## End(Not run)
```

*git\_status*

*Show Git Status*

## Description

Display the working tree status from the R console.

## Usage

```
git_status(short = FALSE)
```

## Arguments

short	Logical; if TRUE, show short format (default: FALSE)
-------	--

## Value

Invisibly returns the status output as a character vector

## Examples

```
## Not run:
git_status()
git_status(short = TRUE)

## End(Not run)
```

---

gui

*Launch Framework GUI*

---

## Description

Opens a beautiful web-based interface for Framework with documentation, project management, and settings configuration.

## Usage

```
gui(port = 8080, host = "127.0.0.1", browse = TRUE, route = NULL)
```

## Arguments

port	Port number to use (default: 8080)
host	Host address to bind to. Default "127.0.0.1" for local access only. Use "0.0.0.0" to allow connections from other machines (requires appropriate network security).
browse	Automatically open browser (default: TRUE)
route	Initial route to open (default: NULL for home page)

## Value

Invisibly returns the plumber server object

## See Also

[setup\(\)](#) for first-time configuration

## Examples

```
## Not run:  
# Launch the GUI  
framework::gui()  
  
# Launch on specific port  
framework::gui(port = 8888)  
  
# Open directly to settings  
framework::gui(route = "#/settings/basics")  
  
# Run as standalone server (no browser, accessible from network)  
framework::gui(port = 8080, host = "0.0.0.0", browse = FALSE)  
  
## End(Not run)
```

---

`init_global_config` *Initialize global Framework settings*

---

**Description**

Creates the Framework config directory (via `fw_config_dir()`) and copies default settings files if they don't already exist. Also handles migration from previous R versions or legacy `~/.config/framework` location.

**Usage**

```
init_global_config(force = FALSE)
```

**Arguments**

`force` If TRUE, overwrites existing settings (default: FALSE)

**Value**

Invisibly returns NULL

---

`list_framework_templates`  
*List available framework templates*

---

**Description**

List available framework templates

**Usage**

```
list_framework_templates()
```

---

`list_metadata` *List all metadata*

---

**Description**

List all metadata

**Usage**

```
list_metadata()
```

**Value**

A data frame of metadata with keys, values, and timestamps

---

```
load_settings_catalog
```

*Read the Framework settings catalog*

---

## Description

The catalog defines metadata (labels, hints) and default values for settings sections. Users can override the packaged defaults by placing a `settings-catalog.yml` file in their Framework config directory (`tools:::R_user_dir("framework", "config")`). When an override exists it is merged on top of the packaged catalog.

## Usage

```
load_settings_catalog(include_user = TRUE, validate = TRUE)
```

## Arguments

<code>include_user</code>	Logical indicating whether to merge user overrides. Defaults to TRUE.
<code>validate</code>	Logical indicating whether to perform basic validation on the catalog structure. Defaults to TRUE.

## Value

A nested list representing the settings catalog.

---

```
make_notebook
```

*Create a Notebook or Script from Stub Template*

---

## Description

Creates a new Quarto (.qmd), RMarkdown (.Rmd) notebook, or R script (.R) from stub templates. Searches for user-provided stubs first (in `stubs/` directory), then falls back to framework defaults.

## Usage

```
make_notebook(  
  name,  
  type = NULL,  
  dir = NULL,  
  stub = "default",  
  overwrite = FALSE,  
  subdir = NULL  
)
```

## Arguments

name	Character. The file name. Extension determines type:
	<ul style="list-style-type: none"> <li>• .qmd: Quarto notebook (default if no extension)</li> <li>• .Rmd: RMarkdown notebook</li> <li>• .R: R script Examples: 1-init, 1-init.qmd, analysis.Rmd, script.R</li> </ul>
type	Character. File type: "quarto", "rmarkdown", or "script". Auto-detected from extension if provided. If NULL (default):
	<ol style="list-style-type: none"> <li>1. Checks config default_notebook_format (or legacy options\$default_notebook)</li> <li>2. Falls back to "quarto" (Framework is Quarto-first)</li> </ol>
dir	Character. Directory to create the file in. Uses your project's configured directories\$notebook setting. Default: "notebooks/".
stub	Character. Name of the stub template to use. Defaults to "default". User can create custom stubs in stubs/notebook-{stub}.qmd, stubs/notebook-{stub}.Rmd, or stubs/script-{stub}.R.
overwrite	Logical. Whether to overwrite existing file. Default FALSE.
subdir	Optional subdirectory under dir (e.g., "analyses/exploratory").

## Details

**Convenient aliases:** Use `make_qmd()` or `make_rmd()` for explicit Quarto or RMarkdown notebook creation. Use `make_revealjs()` or `make_presentation()` for reveal.js presentations.

### Default Output:

Notebooks are created in the `notebooks/` directory by default:

```
notebooks/
  1-data-cleaning.qmd
  2-analysis.qmd
  3-visualization.qmd
```

### Extension Normalization:

- If name includes `.qmd` or `.Rmd`, type is auto-detected
- If no extension provided, `.qmd` is used (Quarto-first)
- Use `type = "rmarkdown"` to default to `.Rmd`

### Stub Template Resolution:

The function searches for stub templates in this order:

1. User stubs: `stubs/notebook-{stub}.qmd` or `stubs/notebook-{stub}.Rmd`
2. Framework stubs: `inst/stubs/notebook-{stub}.qmd` or `inst/stubs/notebook-{stub}.Rmd`

Custom stub templates can use placeholders:

- `{filename}` - The notebook filename without extension
- `{date}` - Current date (YYYY-MM-DD)

## Value

Invisible path to created notebook

**See Also**

[make\\_qmd\(\)](#), [make\\_rmd\(\)](#), [make\\_revealjs\(\)](#), [make\\_presentation\(\)](#)

**Examples**

```
## Not run:
# Create notebooks/1-init.qmd (defaults to Quarto)
make_notebook("1-init")

# Create notebooks/analysis.Rmd (RMarkdown, extension-based)
make_notebook("analysis.Rmd")

# Explicit type parameter
make_notebook("report", type = "rmarkdown")

# Use custom stub template
make_notebook("report", stub = "minimal")

# Create in specific directory
make_notebook("explore", dir = "work")

# Convenient aliases (recommended for explicit types)
make_qmd("analysis")      # Always creates .qmd
make_rmd("report")         # Always creates .Rmd
make_revealjs("slides")    # Creates reveal.js presentation
make_presentation("deck")  # Alias for make_revealjs()

## End(Not run)
```

make\_presentation *Create a Presentation*

**Description**

Alias for [make\\_revealjs\(\)](#). Creates a Quarto reveal.js presentation.

**Usage**

```
make_presentation(name, dir = NULL, overwrite = FALSE, subdir = NULL)
```

**Arguments**

name	Character. The presentation name (with or without .qmd extension)
dir	Character. Directory to create the file in. Uses your project's configured <code>directories\$notebooks</code> setting. Default: "notebooks/".
overwrite	Logical. Whether to overwrite existing file. Default FALSE.
subdir	Optional subdirectory under <code>dir</code> (e.g., "slides/week-01").

**Value**

Invisible path to created presentation

**See Also**

[make\\_notebook\(\)](#), [make\\_revealjs\(\)](#)

**Examples**

```
## Not run:
# Create notebooks/deck.qmd with reveal.js format
make_presentation("deck")

## End(Not run)
```

**make\_qmd**

*Create a Quarto Notebook*

**Description**

Convenient alias for `make_notebook(type = "quarto")`. Creates a .qmd file from stub templates.

**Usage**

```
make_qmd(name, dir = NULL, stub = "default", overwrite = FALSE, subdir = NULL)
```

**Arguments**

<code>name</code>	Character. The file name (with or without .qmd extension)
<code>dir</code>	Character. Directory to create the file in. Uses your project's configured <code>directories\$notebooks</code> setting. Default: "notebooks/".
<code>stub</code>	Character. Name of the stub template to use. Default "default".
<code>overwrite</code>	Logical. Whether to overwrite existing file. Default FALSE.
<code>subdir</code>	Optional subdirectory under <code>dir</code> (e.g., "slides/week-01").

**Value**

Invisible path to created notebook

**See Also**

[make\\_notebook\(\)](#), [make\\_rmd\(\)](#)

**Examples**

```
## Not run:
# Create notebooks/analysis.qmd
make_qmd("analysis")

# Use custom stub
make_qmd("report", stub = "minimal")

# Create in specific directory
make_qmd("explore", dir = "work")
```

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

---

make\_revealjs      *Create a Reveal.js Presentation*

---

## Description

Convenient alias for creating reveal.js presentations. Always creates a Quarto notebook with the revealjs stub template.

## Usage

```
make_revealjs(name, dir = NULL, overwrite = FALSE, subdir = NULL)
```

## Arguments

name	Character. The presentation name (with or without .qmd extension)
dir	Character. Directory to create the file in. Uses your project's configured <code>directories\$notebook</code> setting. Default: "notebooks/".
overwrite	Logical. Whether to overwrite existing file. Default FALSE.
subdir	Optional subdirectory under <code>dir</code> (e.g., "slides/week-01").

## Value

Invisible path to created presentation

## See Also

[make\\_notebook\(\)](#), [make\\_qmd\(\)](#), [make\\_presentation\(\)](#)

## Examples

```
## Not run:  
# Create notebooks/slides.qmd with reveal.js format  
make_revealjs("slides")  
  
# Create in specific directory  
make_revealjs("presentation", dir = "presentations")  
  
## End(Not run)
```

---

make\_rmd

*Create an RMarkdown Notebook*

---

## Description

Convenient alias for `make_notebook(type = "rmarkdown")`. Creates a .Rmd file from stub templates.

## Usage

```
make_rmd(name, dir = NULL, stub = "default", overwrite = FALSE, subdir = NULL)
```

## Arguments

<code>name</code>	Character. The file name (with or without .Rmd extension)
<code>dir</code>	Character. Directory to create the file in. Uses your project's configured <code>directories\$notebooks</code> setting. Default: "notebooks/".
<code>stub</code>	Character. Name of the stub template to use. Default "default".
<code>overwrite</code>	Logical. Whether to overwrite existing file. Default FALSE.
<code>subdir</code>	Optional subdirectory under <code>dir</code> (e.g., "analyses/exploratory").

## Value

Invisible path to created notebook

## See Also

[make\\_notebook\(\)](#), [make\\_qmd\(\)](#)

## Examples

```
## Not run:  
# Create notebooks/analysis.Rmd  
make_rmd("analysis")  
  
# Use custom stub  
make_rmd("report", stub = "minimal")  
  
# Create in specific directory  
make_rmd("explore", dir = "work")  
  
## End(Not run)
```

---

make\_script      *Create an R Script from Stub Template*

---

## Description

Convenience wrapper for `make_notebook()` that creates R scripts (.R files). This is identical to calling `make_notebook("name.R")`.

## Usage

```
make_script(name, dir = NULL, stub = "default", overwrite = FALSE)
```

## Arguments

<code>name</code>	Character. The script name (with or without .R extension). Examples: "process-data", "process-data.R"
<code>dir</code>	Character. Directory to create the script in. Uses your project's configured <code>directories\$scripts</code> setting. Default: "scripts/".
<code>stub</code>	Character. Name of the stub template to use. Defaults to "default". User can create custom stubs in <code>stubs/script-{stub}.R</code> .
<code>overwrite</code>	Logical. Whether to overwrite existing file. Default FALSE.

## Details

This function is a convenience wrapper that:

1. Ensures the name ends with .R extension
2. Uses `script_dir` config option instead of `notebook_dir`
3. Calls `make_notebook()` with `type = "script"`

### Default Output:

Scripts are created in the `scripts/` directory by default:

```
scripts/
  process-data.R
  build-features.R
  run-model.R
```

## Value

Invisible path to created script

## See Also

[make\\_notebook\(\)](#) for creating Quarto/RMarkdown notebooks

## Examples

```
## Not run:
# Create script (extension optional)
make_script("process-data")
make_script("process-data.R")

# Use custom stub
make_script("etl-pipeline", stub = "etl")

# Create in specific directory
make_script("analysis", dir = "analysis/")

## End(Not run)
```

new

*Create a New Project (Master Wrapper)*

## Description

Flexible project creation interface. Alias for `new_project()` that accepts type as a parameter.

## Usage

```
new(
  name = NULL,
  location = NULL,
  type = "project",
  browse = interactive(),
  ...
)
```

## Arguments

<code>name</code>	Project name. If NULL (default), prompts interactively.
<code>location</code>	Directory path where project will be created. If NULL (default), prompts interactively.
<code>type</code>	Project type. One of "project" (default), "project_sensitive", "course", or "presentation".
<code>browse</code>	Whether to open the project folder after creation (default: TRUE in interactive sessions)
<code>...</code>	Additional arguments passed to 'project_

## Value

Invisibly returns the result from `project_create()`

## See Also

[new\\_project\(\)](#), [new\\_project\\_sensitive\(\)](#), [new\\_presentation\(\)](#), [new\\_course\(\)](#)

## Examples

```
## Not run:  
# Create different project types  
new("analysis", "~/projects/analysis")  
new("study", "~/projects/study", type = "project_sensitive")  
new("slides", "~/projects/slides", type = "presentation")  
new("course-materials", "~/projects/course", type = "course")  
  
## End(Not run)
```

---

new\_course

*Create a Course Project*

---

## Description

Shorthand for `new_project(..., type = "course")`. Creates a project structured for teaching materials with slides, assignments, and modules.

## Usage

```
new_course(name = NULL, location = NULL, browse = interactive(), ...)
```

## Arguments

name	Project name. If NULL (default), prompts interactively.
location	Directory path where project will be created. If NULL (default), prompts interactively.
browse	Whether to open the project folder after creation (default: TRUE in interactive sessions)
...	Additional arguments passed to ‘project_’

## Value

Invisibly returns the result from `project_create()`

## See Also

[new\\_project\(\)](#)

## Examples

```
## Not run:  
new_course("stats-101", "~/projects/stats-101")  
  
## End(Not run)
```

`new_presentation`    *Create a Presentation Project*

## Description

Shorthand for `new_project(..., type = "presentation")`. Creates a project optimized for RevealJS presentations.

## Usage

```
new_presentation(name = NULL, location = NULL, browse = interactive(), ...)
```

## Arguments

<code>name</code>	Project name. If NULL (default), prompts interactively.
<code>location</code>	Directory path where project will be created. If NULL (default), prompts interactively.
<code>browse</code>	Whether to open the project folder after creation (default: TRUE in interactive sessions)
<code>...</code>	Additional arguments passed to ‘project_’

## Value

Invisibly returns the result from `project_create()`

## See Also

[new\\_project\(\)](#)

## Examples

```
## Not run:
new_presentation("quarterly-review", "~/projects/q4-review")

## End(Not run)
```

`new_project`    *Create a New Framework Project*

## Description

Convenience wrapper for creating Framework projects from the command line. Uses global settings configured via `setup()` as defaults, prompts for missing required values (name and location).

## Usage

```
new_project(  
  name = NULL,  
  location = NULL,  
  type = "project",  
  browse = interactive(),  
  ...  
)
```

## Arguments

name	Project name. If NULL (default), prompts interactively.
location	Directory path where project will be created. If NULL (default), prompts interactively.
type	Project type. One of "project" (default), "project_sensitive", "course", or "presentation".
browse	Whether to open the project folder after creation (default: TRUE in interactive sessions)
...	Additional arguments passed to 'project_

## Details

This function is designed for the streamlined workflow:

```
remotes::install_github("table1/framework")  
framework::setup()          # One-time global configuration  
framework::new_project()    # Create projects using saved defaults
```

Global settings from `tools::R_user_dir("framework", "config")` are used for:

- Author information (name, email, affiliation)
- Default packages
- Directory structure
- Git settings
- AI assistant configuration
- Quarto format preferences

## Value

Invisibly returns the result from `project_create()` (list with success, path, and `project_id`)

## See Also

[setup\(\)](#) for initial configuration, [project\\_create\(\)](#) for full control

## Examples

```
## Not run:
# Interactive - prompts for name and location
new_project()

# With name and location specified
new_project("my-analysis", "~/projects/my-analysis")

# Create a sensitive data project
new_project("medical-study", "~/projects/medical", type = "project_sensitive")

## End(Not run)
```

### **new\_project\_sensitive**

*Create a Sensitive Data Project*

## Description

Shorthand for `new_project(..., type = "project_sensitive")`. Creates a project with additional privacy protections for handling sensitive data.

## Usage

```
new_project_sensitive(
  name = NULL,
  location = NULL,
  browse = interactive(),
  ...
)
```

## Arguments

<code>name</code>	Project name. If <code>NULL</code> (default), prompts interactively.
<code>location</code>	Directory path where project will be created. If <code>NULL</code> (default), prompts interactively.
<code>browse</code>	Whether to open the project folder after creation (default: <code>TRUE</code> in interactive sessions)
<code>...</code>	Additional arguments passed to ‘ <code>project_</code>

## Value

Invisibly returns the result from `project_create()`

## See Also

[new\\_project\(\)](#)

**Examples**

```
## Not run:  
new_project_sensitive("medical-study", "~/projects/medical")  
  
## End(Not run)
```

---

now

*Get current datetime*

---

**Description**

Get current datetime

**Usage**

```
now()
```

**Value**

Current datetime as an ISO 8601 formatted character string

---

outputs

*Output Save Functions*

---

**Description**

First-class functions for saving tables, figures, models, and reports. These functions implement lazy directory creation with console feedback.

---

packages\_install

*Install packages from configuration*

---

**Description**

Installs all packages defined in the configuration that are not already installed. This is the same logic used in scaffold(), but exposed as a standalone function.

**Usage**

```
packages_install()
```

**Value**

Invisibly returns TRUE on success

## Examples

```
## Not run:
# Install all configured packages
packages_install()

## End(Not run)
```

**packages\_list**      *List all packages from configuration*

## Description

Lists all packages defined in the configuration, showing the package name, version pin (if specified), and source (CRAN or GitHub).

## Usage

```
packages_list()
```

## Value

Invisibly returns NULL after printing package list

## Examples

```
## Not run:
# List all packages
packages_list()

## End(Not run)
```

**packages\_restore**      *Restore packages from renv.lock*

## Description

Wrapper around `renv::restore()` that requires Framework's `renv` integration to be enabled first.

## Usage

```
packages_restore(prompt = FALSE)
```

## Arguments

<code>prompt</code>	Logical. If TRUE, <code>renv</code> prompts before restoring.
---------------------	---

## Value

Invisibly returns TRUE on success.

---

packages\_snapshot    *Snapshot current package library (renv)*

---

## Description

Wrapper around `renv::snapshot()` that requires Framework's renv integration to be enabled first.

## Usage

```
packages_snapshot(prompt = FALSE)
```

## Arguments

`prompt`        Logical. If TRUE, renv prompts before writing the snapshot.

## Value

Invisibly returns TRUE on success.

---

packages\_status    *Show renv package status*

---

## Description

Wrapper around `renv::status()` that requires Framework's renv integration.

## Usage

```
packages_status()
```

## Value

The status object returned by `renv::status()`.

---

packages_update	<i>Update packages from configuration</i>
-----------------	---

---

## Description

Updates packages defined in the configuration. If renv is enabled, uses `renv::update()`. Otherwise, reinstalls packages using standard installation methods.

## Usage

```
packages_update(packages = NULL)
```

## Arguments

packages	Character vector of specific packages to update, or <code>NULL</code> to update all
----------	---

## Value

Invisibly returns `TRUE` on success

## Examples

```
## Not run:
# Update all packages
packages_update()

# Update specific packages
packages_update(c("dplyr", "ggplot2"))

## End(Not run)
```

---

project_add_directory	<i>Add custom directories to an existing project</i>
-----------------------	--

---

## Description

Adds new directories to a project's configuration and creates them on the filesystem. This function is used by the GUI to allow users to add custom directories to their project structure without modifying existing directories.

## Usage

```
project_add_directory(project_path, key, label, path)
```

## Arguments

project_path	Character string. Absolute path to the project root directory.
key	Character string. Internal key for the directory (e.g., "analysis_archive"). Must be unique within the project's directory configuration.
label	Character string. Human-readable label for the directory (e.g., "Analysis Archive").
path	Character string. Relative path where the directory should be created (e.g., "analysis/archive"). Must be relative, not absolute. Parent directories will be created as needed.

## Details

This function performs the following steps:

1. Validates all input arguments
2. Reads the project's config.yml file
3. Checks for duplicate keys in existing directories
4. Adds the new directory to the directories section
5. Writes the updated config.yml back to disk
6. Creates the directory on the filesystem (with recursive creation)

The function follows a non-destructive, additive-only approach. It will not:

- Rename existing directories
- Delete existing directories
- Modify existing directory paths
- Change the project type

## Value

List with success status and directory information:

- success: Logical indicating whether the operation succeeded
- directory: List containing key, label, path, absolute\_path, and created flag
- error: Character string with error message (only present if success is FALSE)

## Safety

The function includes several safety checks:

- Rejects absolute paths (must be relative)
- Rejects paths containing ".." (no directory traversal)
- Checks for duplicate keys before adding
- Wraps filesystem operations in error handling

---

project_create	<i>Create a new Framework project (internal)</i>
----------------	--

---

## Description

Low-level function that creates a complete Framework project. This is called by the GUI and by user-facing functions like `new()` and `new_project()`. Users should typically use `new()` instead.

## Usage

```
project_create(
  name,
  location,
  type = "project",
  author = list(name = "", email = "", affiliation = ""),
  packages = list(use_renv = FALSE, default_packages = list()),
  directories = list(),
  extra_directories = list(),
  ai = list(enabled = FALSE, assistants = c(), canonical_content = ""),
  git = list(use_git = TRUE, hooks = list(), gitignore_content = ""),
  scaffold = list(seed_on_scaffold = FALSE, seed = "", set_theme_on_scaffold = TRUE,
    ggplot_theme = "theme_minimal"),
  connections = NULL,
  env = NULL,
  quarto = NULL,
  render_dirs = NULL
)
```

## Arguments

<code>name</code>	Project name (used for project title)
<code>location</code>	Full path to the project directory (will be created)
<code>type</code>	Project type: "project", "project_sensitive", "course", "presentation"
<code>author</code>	List with name, email, affiliation
<code>packages</code>	List with use_renv (logical) and default_packages (list of package configs)
<code>directories</code>	Named list of directory paths (notebooks, scripts, functions, etc.)
<code>extra_directories</code>	List of additional custom directories
<code>ai</code>	List with enabled, assistants, canonical_content
<code>git</code>	List with use_git, hooks, gitignore_content
<code>scaffold</code>	List with seed_on_scaffold, seed, set_theme_on_scaffold, ggplot_theme, ide, positron
<code>quarto</code>	List with html and revealjs format configurations for Quarto
<code>render_dirs</code>	Named list of render directory paths for Quarto outputs

## Value

List with success status, project path, and project ID

---

project_info	<i>Display project structure information</i>
--------------	--

---

**Description**

Shows configured directories and their status (created or pending lazy creation). Useful for understanding the project structure and discovering available paths.

**Usage**

```
project_info(verbose = FALSE)
```

**Arguments**

verbose	If TRUE, shows additional details about each directory
---------	--

**Value**

A data frame with directory information (invisibly)

**Examples**

```
## Not run:  
# Show project structure  
project_info()  
  
# Get detailed info  
project_info(verbose = TRUE)  
  
## End(Not run)
```

---

project_list	<i>List all projects in global configuration</i>
--------------	--

---

**Description**

List all projects in global configuration

**Usage**

```
project_list()
```

**Value**

Data frame with project information

## publish

*Publishing Functions***Description**

Functions for publishing notebooks, data, and files to S3 storage.

Upload files or directories to an S3 bucket. This is the generic publishing function - use `publish_notebook()` for Quarto documents or `publish_data()` for data files.

**Usage**

```
publish(source, dest = NULL, connection = NULL, overwrite = TRUE)
```

**Arguments**

<code>source</code>	Character. Local file or directory path to upload.
<code>dest</code>	Character or NULL. Destination path in S3 bucket. If NULL, derives from source filename.
<code>connection</code>	Character or NULL. S3 connection name from config.yml. If NULL, uses the connection marked with <code>default: true</code> .
<code>overwrite</code>	Logical. Whether to overwrite existing files. Default TRUE.

**Value**

Character. The public URL(s) of uploaded file(s).

**Examples**

```
## Not run:
# Upload a single file
publish("outputs/report.html")
# -> https://bucket.s3.region.amazonaws.com/prefix/report.html

# Upload with custom destination
publish("outputs/report.html", dest = "reports/q4-2024.html")

# Upload a directory
publish("outputs/charts/", dest = "reports/charts/")

# Use specific connection
publish("data.csv", connection = "s3_backup")

## End(Not run)
```

---

publish_data	<i>Publish data to S3</i>
--------------	---------------------------

---

## Description

Uploads a data frame or existing data file to S3.

## Usage

```
publish_data(data, dest, format = "csv", connection = NULL, compress = FALSE)
```

## Arguments

data	Data frame or character path to existing file.
dest	Character. Destination path in S3 (required for data frames).
format	Character. Output format when data is a data frame: "csv", "rds", "parquet", or "json". Default "csv".
connection	Character or NULL. S3 connection name, or NULL for default.
compress	Logical. Whether to gzip compress. Default FALSE.

## Value

Character. Public URL of the published data.

## Examples

```
## Not run:  
# Publish a data frame  
publish_data(my_df, "datasets/customers.csv")  
  
# Publish as RDS  
publish_data(my_df, "datasets/customers.rds", format = "rds")  
  
# Publish existing file  
publish_data("outputs/model.rds", "models/v2/model.rds")  
  
## End(Not run)
```

---

publish_dir	<i>Publish a directory to S3</i>
-------------	----------------------------------

---

## Description

Recursively uploads all files in a directory to S3.

**Usage**

```
publish_dir(
  dir,
  dest = NULL,
  connection = NULL,
  pattern = NULL,
  recursive = TRUE
)
```

**Arguments**

dir	Character. Local directory path.
dest	Character or NULL. Destination prefix in S3. If NULL, uses the directory name.
connection	Character or NULL. S3 connection name, or NULL for default.
pattern	Character or NULL. Optional regex pattern to filter files.
recursive	Logical. Whether to include subdirectories. Default TRUE.

**Value**

Character vector. Public URLs of uploaded files.

**Examples**

```
## Not run:
# Upload entire directory
publish_dir("outputs/dashboard/")

# Upload to specific location
publish_dir("outputs/dashboard/", dest = "dashboards/v2/")

# Upload only HTML files
publish_dir("outputs/", pattern = "\\.html$")

## End(Not run)
```

**publish\_list**      *List published files in S3*

**Description**

Lists files in an S3 bucket/prefix.

**Usage**

```
publish_list(prefix = NULL, connection = NULL, max = 1000L)
```

**Arguments**

prefix	Character or NULL. Prefix to filter by. If NULL, lists all files under the connection's configured prefix.
connection	Character or NULL. S3 connection name, or NULL for default.
max	Integer. Maximum number of files to list. Default 1000.

**Value**

Data frame with columns: key, size, last\_modified.

**Examples**

```
## Not run:  
# List all published files  
publish_list()  
  
# List files under a prefix  
publish_list("reports/")  
  
# List from specific connection  
publish_list(connection = "s3_backup")  
  
## End(Not run)
```

---

publish\_notebook    *Publish a Quarto notebook to S3*

---

**Description**

Renders a Quarto document and uploads it to S3. The notebook is rendered to a temporary directory, uploaded, then cleaned up.

**Usage**

```
publish_notebook (  
  file,  
  dest = NULL,  
  connection = NULL,  
  self_contained = TRUE,  
  format = "html",  
  ...  
)
```

**Arguments**

file	Character. Path to .qmd file.
dest	Character or NULL. Destination path in S3 (without extension). If NULL, derives from filename (e.g., "analysis.qmd" -> "analysis").
connection	Character or NULL. S3 connection name, or NULL for default.
self_contained	Logical. Whether to embed all resources. Default TRUE. Ignored if static_hosting: false (always renders self-contained).
format	Character. Output format. Default "html".
...	Additional arguments passed to quarto render.

## Details

The URL format depends on the S3 connection's `static_hosting` setting:

- `static_hosting: true` -> uploads to `dest/index.html`, returns `dest/`
- `static_hosting: false` (default) -> uploads as `dest.html`, returns `dest.html`

## Value

Character. Public URL of the published notebook.

## Examples

```
## Not run:
# With static_hosting: true -> returns /analysis/
# With static_hosting: false -> returns /analysis.html
publish_notebook("notebooks/analysis.qmd")

# Publish to specific location
publish_notebook("notebooks/analysis.qmd", dest = "reports/2024/q4")

# Publish non-self-contained (only with static_hosting: true)
publish_notebook("notebooks/analysis.qmd", self_contained = FALSE)

## End(Not run)
```

## *quarto\_generate\_all*

*Generate Quarto Configurations for Project*

## Description

Main entry point for generating all `_quarto.yml` files in a project. Generates root config and directory-specific configs based on project type.

## Usage

```
quarto_generate_all(
  project_path,
  project_type,
  render_dirs = NULL,
  quarto_settings = NULL,
  directories = NULL,
  root_output_dir = NULL
)
```

## Arguments

<code>project_path</code>	Character. Path to project root
<code>project_type</code>	Character. One of "project", "project_sensitive", "course", "presentation"
<code>render_dirs</code>	Named list. Render directories with their paths
<code>quarto_settings</code>	List. Quarto settings (html and revealjs configs)

```
directories  Named list. Source directories keyed the same as render_dirs  
root_output_dir  
          Optional output directory to set on the root _quarto.yml
```

### Value

List with success status and paths of generated files

---

quarto\_regenerate *Regenerate Quarto Configurations*

---

### Description

Regenerates all \_quarto.yml files in a project. **WARNING: This will overwrite any manual edits.** Should only be called when user explicitly requests regeneration.

### Usage

```
quarto_regenerate(project_path, backup = TRUE)
```

### Arguments

project_path	Character. Path to project root
backup	Logical. If TRUE, backs up existing files before overwriting. Default TRUE.

### Value

List with success status, backed up files, and regenerated files

---

read\_frameworkrc *Read global Framework configuration*

---

### Description

Read global Framework configuration

### Usage

```
read_frameworkrc(use_defaults = TRUE)
```

### Arguments

use_defaults	Whether to merge with default structure (default: TRUE)
--------------	---

### Value

List containing global configuration

---

```
read_framework_template  
Read the contents of a Framework template
```

---

**Description**

Read the contents of a Framework template

**Usage**

```
read_framework_template(name)
```

**Arguments**

name	Template identifier (e.g., "notebook", "gitignore", "ai_claude")
------	--

**Value**

Character scalar containing template contents

---

```
remove_project_from_config  
Remove project from global configuration
```

---

**Description**

Remove project from global configuration

**Usage**

```
remove_project_from_config(project_id)
```

**Arguments**

project_id	Project ID to remove
------------	----------------------

**Value**

Invisibly returns NULL

---

renv_disable	<i>Disable renv for this project</i>
--------------	--------------------------------------

---

## Description

Deactivates renv integration while preserving renv.lock for future use. Removes the .framework\_renv\_enabled marker file.

## Usage

```
renv_disable(keep_renv = TRUE)
```

## Arguments

keep\_renv Logical; if TRUE (default), keep renv.lock and renv/ directory

## Value

Invisibly returns TRUE on success

## Examples

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

---

renv_enable	<i>Enable renv for this project</i>
-------------	-------------------------------------

---

## Description

Initializes renv integration for the current Framework project. This:

- Creates .framework\_renv\_enabled marker file
- Initializes renv if not already initialized
- Syncs packages from settings.yml to renv.lock
- Updates .gitignore to exclude renv cache

## Usage

```
renv_enable(sync = TRUE)
```

## Arguments

sync Logical; if TRUE (default), sync packages from settings.yml

## Value

Invisibly returns TRUE on success

## Examples

```
## Not run:
renv_enable()

## End(Not run)
```

renv_enabled	<i>Check if renv is enabled for this project</i>
--------------	--

## Description

Determines whether renv integration is active by checking for the `.framework_renv_enabled` marker file in the project root.

## Usage

```
renv_enabled()
```

## Value

Logical indicating whether renv is enabled

## Examples

```
if (renv_enabled()) {
  message("Using renv for package management")
}
```

renv_restore	<i>Restore packages from renv.lock</i>
--------------	--

## Description

Internal wrapper around `renv::restore()`. Use `packages_restore()` instead.

## Usage

```
renv_restore(prompt = FALSE)
```

## Arguments

prompt	Logical; if TRUE, prompt before restoring
--------	---

## Value

Invisibly returns TRUE on success

---

renv_snapshot	<i>Snapshot current package versions to renv.lock</i>
---------------	---

---

**Description**

Internal wrapper around `renv::snapshot()`. Use `packages_snapshot()` instead.

**Usage**

```
renv_snapshot(prompt = FALSE)
```

**Arguments**

prompt	Logical; if TRUE, prompt before creating snapshot
--------	---

**Value**

Invisibly returns TRUE on success

---

renv_status	<i>Show package status</i>
-------------	----------------------------

---

**Description**

Internal wrapper around `renv::status()`. Use `packages_status()` instead.

**Usage**

```
renv_status()
```

**Value**

Invisibly returns the status object from `renv::status()`

---

renv_sync	<i>Sync packages with renv.lock</i>
-----------	-------------------------------------

---

**Description**

Internal function that resolves inconsistencies between installed packages and `renv.lock` by restoring then snapshotting.

**Usage**

```
renv_sync(prompt = FALSE)
```

**Arguments**

prompt	Logical; if TRUE, prompt before making changes
--------	--

**Value**

Invisibly returns TRUE on success

---

renv_update	<i>Update packages</i>
-------------	------------------------

---

**Description**

Internal wrapper around `renv::update()`. Use `packages_update()` instead.

**Usage**

```
renv_update(packages = NULL)
```

**Arguments**

packages	Character vector of package names to update, or NULL for all
----------	--

**Value**

Invisibly returns TRUE on success

---

reset_framework_template	<i>Reset a Framework template back to the packaged default</i>
--------------------------	--

---

**Description**

Reset a Framework template back to the packaged default

**Usage**

```
reset_framework_template(name)
```

**Arguments**

name	Template identifier (e.g., "notebook", "gitignore", "ai_claude")
------	--

---

result_list	<i>List saved results from the framework database</i>
-------------	---

---

## Description

Retrieves a list of all saved results (tables, figures, models, reports, notebooks) that have been tracked via the save\_\* functions.

## Usage

```
result_list(type = NULL, public = NULL)
```

## Arguments

type	Optional filter by type: "table", "figure", "model", "report", "notebook"
public	Optional filter: TRUE for public results only, FALSE for private only

## Value

A data frame with columns: name, type, public, comment, hash, created\_at, updated\_at. Returns an empty data frame if no results found or database unavailable.

## Examples

```
## Not run:  
# List all results  
result_list()  
  
# List only tables  
result_list(type = "table")  
  
# List only public figures  
result_list(type = "figure", public = TRUE)  
  
## End(Not run)
```

---

save_figure	<i>Save a figure to the outputs directory</i>
-------------	---

---

## Description

Saves a ggplot2 plot or base R graphics to the configured figures directory. The directory is created lazily on first use.

**Usage**

```
save_figure(
  plot = NULL,
  name,
  format = "png",
  width = 8,
  height = 6,
  dpi = 300,
  public = FALSE,
  overwrite = TRUE,
  ...
)
```

**Arguments**

<code>plot</code>	A <code>ggplot2</code> object, or <code>NULL</code> to save the current plot
<code>name</code>	The name for the output file (without extension)
<code>format</code>	Output format: "png" (default), "pdf", "svg", or "jpg"
<code>width</code>	Width in inches (default: 8)
<code>height</code>	Height in inches (default: 6)
<code>dpi</code>	Resolution in dots per inch (default: 300)
<code>public</code>	If <code>TRUE</code> , saves to public outputs directory (for project-sensitive type)
<code>overwrite</code>	If <code>TRUE</code> , overwrites existing files (default: <code>TRUE</code> )
<code>...</code>	Additional arguments passed to <code>ggsave</code> or the graphics device

**Value**

The path to the saved file (invisibly)

**Examples**

```
## Not run:
# Save a ggplot
p <- ggplot(mtcars, aes(mpg, hp)) + geom_point()
save_figure(p, "mpg_vs_hp")

# Save as PDF for publication
save_figure(p, "mpg_vs_hp", format = "pdf", width = 10, height = 8)

# Save to public directory
save_figure(p, "summary_plot", public = TRUE)

## End(Not run)
```

---

save_model	<i>Save a model to the outputs directory</i>
------------	--

---

## Description

Saves a fitted model object to the configured models directory. The directory is created lazily on first use.

## Usage

```
save_model(model, name, format = "rds", public = FALSE, overwrite = TRUE, ...)
```

## Arguments

model	A fitted model object (lm, glm, tidyverse workflow, etc.)
name	The name for the output file (without extension)
format	Output format: "rds" (default) or "qs" (faster, requires qs package)
public	If TRUE, saves to public outputs directory (for project_sensitive type)
overwrite	If TRUE, overwrites existing files (default: TRUE)
...	Additional arguments passed to the underlying save function

## Value

The path to the saved file (invisibly)

## Examples

```
## Not run:  
# Fit and save a model  
model <- lm(mpg ~ hp + wt, data = mtcars)  
save_model(model, "mpg_regression")  
  
# Save with qs for faster serialization  
save_model(model, "mpg_regression", format = "qs")  
  
## End(Not run)
```

---

save_notebook	<i>Save a rendered notebook to the outputs directory</i>
---------------	--

---

## Description

Renders a Quarto or R Markdown notebook and saves the output to the configured notebooks output directory. The directory is created lazily on first use.

**Usage**

```
save_notebook(
  file,
  name = NULL,
  format = "html",
  public = FALSE,
  overwrite = TRUE,
  embed_resources = TRUE,
  ...
)
```

**Arguments**

<code>file</code>	Path to the .qmd or .Rmd file to render
<code>name</code>	Optional new name for the output file (without extension). If NULL, uses the original notebook name.
<code>format</code>	Output format: "html" (default), "pdf", or "docx"
<code>public</code>	If TRUE, saves to public outputs directory (for project_sensitive type)
<code>overwrite</code>	If TRUE, overwrites existing files (default: TRUE)
<code>embed_resources</code>	If TRUE, creates a self-contained file with embedded resources (default: TRUE for html format)
<code>...</code>	Additional arguments passed to quarto render

**Value**

The path to the saved file (invisibly)

**Examples**

```
## Not run:
# Render and save a notebook
save_notebook("notebooks/analysis.qmd")

# Save with a custom name
save_notebook("notebooks/analysis.qmd", name = "final_analysis")

# Render to PDF
save_notebook("notebooks/analysis.qmd", format = "pdf")

# Save to public directory (for sensitive projects)
save_notebook("notebooks/analysis.qmd", public = TRUE)

## End(Not run)
```

---

save_report	<i>Save a report to the outputs directory</i>
-------------	---

---

## Description

Copies or moves a rendered report (HTML, PDF, etc.) to the configured reports directory. The directory is created lazily on first use.

## Usage

```
save_report(file, name = NULL, public = FALSE, overwrite = TRUE, move = FALSE)
```

## Arguments

file	Path to the report file to save
name	Optional new name for the file (without extension). If NULL, uses original name.
public	If TRUE, saves to public outputs directory (for project_sensitive type)
overwrite	If TRUE, overwrites existing files (default: TRUE)
move	If TRUE, moves the file instead of copying (default: FALSE)

## Value

The path to the saved file (invisibly)

## Examples

```
## Not run:  
# Save a rendered HTML report  
save_report("notebooks/analysis.html", "final_analysis")  
  
# Save to public directory  
save_report("notebooks/summary.pdf", "public_summary", public = TRUE)  
  
## End(Not run)
```

---

save_table	<i>Save a table to the outputs directory</i>
------------	--

---

## Description

Saves a data frame or tibble to the configured tables directory. The directory is created lazily on first use.

## Usage

```
save_table(data, name, format = "csv", public = FALSE, overwrite = TRUE, ...)
```

**Arguments**

data	A data frame, tibble, or other tabular data
name	The name for the output file (without extension)
format	Output format: "csv" (default), "rds", "xlsx", or "parquet"
public	If TRUE, saves to public outputs directory (for project_sensitive type)
overwrite	If TRUE, overwrites existing files (default: TRUE)
...	Additional arguments passed to the underlying write function

**Value**

The path to the saved file (invisibly)

**Examples**

```
## Not run:
# Save a simple table
save_table(my_results, "regression_results")

# Save as Excel
save_table(my_results, "regression_results", format = "xlsx")

# Save to public directory (for sensitive projects)
save_table(summary_stats, "summary", public = TRUE)

## End(Not run)
```

**Description**

The primary entry point for working with Framework projects. Call this at the start of every notebook or script to set up your environment with all configured packages, functions, and settings.

**Usage**

```
scaffold(config_file = NULL)
```

**Arguments**

config_file	Path to configuration file. If NULL (default), automatically discovers settings.yml or config.yml in the project.
-------------	---

## Details

`scaffold()` performs the following steps in order:

1. **Standardizes working directory** - Finds and sets the project root, even when called from notebooks in subdirectories
2. **Loads environment variables** - Reads secrets from `.env` file
3. **Loads configuration** - Parses `settings.yml` for project settings
4. **Sets random seed** - For reproducibility (if `seed` is configured)
5. **Installs packages** - Any missing packages from the `packages` list
6. **Loads packages** - Attaches all configured packages
7. **Sources functions** - Loads all `.R` files from `functions/` directory
8. **Creates config object** - Makes `config` available in global environment

After `scaffold()` completes, you have access to:

- All packages listed in `settings.yml`
- All functions from your `functions/` directory
- The `config` object for accessing settings via `config("key")`
- Database connections configured in your project

## Value

Invisibly returns `NULL`. The main effects are side effects: loading packages, sourcing functions, and creating the `config` object.

## Project Discovery

When called without arguments, `scaffold()` searches for a Framework project by:

- Looking for `settings.yml` or `config.yml` in current and parent directories
- Checking for `.Rproj` or `.code-workspace` files with nearby settings
- Recognizing common Framework subdirectories (`notebooks/`, `scripts/`, etc.)

This means you can call `scaffold()` from any subdirectory within your project.

## Configuration

The `settings.yml` file controls what `scaffold()` loads. Key settings include:

- `packages`: List of R packages to install and load
- `seed`: Random seed for reproducibility
- `directories`: Custom directory paths
- `connections`: Database connection configurations

## See Also

- [`project\_create\(\)`](#) to create a new Framework project
- [`standardize\_wd\(\)`](#) for just the working directory standardization
- [`config\(\)`](#) to access configuration values after scaffolding

## Examples

```
## Not run:
# At the top of every notebook or script:
library(framework)
scaffold()

# Now you can use your configured packages and functions
# Access settings via the config object:
config("directories.notebooks")
config("seed")

## End(Not run)
```

**scratch\_capture**      *Capture and Save Data to File*

## Description

Saves data to a file in various formats based on the object type and specified format. If no name is provided, uses the name of the object passed in. If no location is provided, uses the scratch directory from the configuration.

## Usage

```
scratch_capture(x, name = NULL, to = NULL, location = NULL, n = Inf)
```

## Arguments

x	The object to save
name	Optional character string specifying the name of the file (without extension). If not provided, will use the name of the object passed in.
to	Optional character string indicating the output format. One of: "text", "rds", "csv", "tsv". If not provided, will choose based on object type.
location	Optional character string specifying the directory where the file should be saved. If NULL, uses the scratch directory from the configuration.
n	Optional number of rows to capture for data frames (default: all rows)

## Value

The input object x invisibly.

## Examples

```
## Not run:
# Save a character vector as text
scratch_capture(c("hello", "world"))

# Save a data frame as TSV
scratch_capture(mtcars)
```

```
# Save an R object as RDS
scratch_capture(list(a = 1, b = 2), to = "rds")

## End(Not run)
```

---

scratch_clean	<i>Clean up the scratch directory by deleting all files</i>
---------------	---

---

## Description

Clean up the scratch directory by deleting all files

## Usage

```
scratch_clean()
```

---

settings	<i>Get settings value by dot-notation key</i>
----------	---

---

## Description

Framework's primary configuration helper that supports both flat and hierarchical key access using dot notation. Automatically checks common locations for directory settings. Pretty-prints nested structures in interactive sessions.

## Usage

```
settings(key = NULL, default = NULL, settings_file = NULL)
```

## Arguments

key	Character. Dot-notation key path (e.g., "notebooks" or "directories.notebooks" or "connections.db.host"). If NULL, returns entire settings.
default	Optional default value if key is not found (default: NULL)
settings_file	Settings file path (default: checks "settings.yml" then "settings.yml")

## Details

For directory settings, the function checks multiple locations:

- Direct: `settings("notebooks")` checks `directories$notebooks`, then `options$notebook_dir`
- Explicit: `settings("directories.notebooks")` checks only `directories$notebooks`

### File Discovery:

- Checks `settings.yml` first (Framework's preferred convention)
- Falls back to `settings.yaml` if `settings.yml` not found
- You can override with explicit `settings_file` parameter

**Output Behavior:**

- Interactive sessions: Pretty-prints nested lists/structures and returns invisibly
- Non-interactive (scripts): Returns raw value without printing
- Simple values: Always returned directly without modification

**Value**

The settings value, or default if not found. In interactive sessions, nested structures are pretty-printed and returned invisibly.

**Examples**

```
## Not run:
# Get notebook directory (checks both locations)
settings("notebooks")

# Get explicit nested setting
settings("directories.notebooks")
settings("connections.db.host")

# Get entire section
settings("directories") # Returns all directory settings
settings("connections") # Returns all connection settings

# View entire settings
settings() # Returns full configuration

# With default value
settings("missing_key", default = "fallback")

## End(Not run)
```

`settings_read`      *Read project settings*

**Description**

Reads the project settings from settings.yml or config.yml with environment-aware merging and split file resolution. Auto-discovers the settings file if not specified.

**Usage**

```
settings_read(settings_file = NULL, environment = NULL)
```

**Arguments**

<code>settings_file</code>	Path to settings file (default: auto-discover settings.yml or config.yml)
<code>environment</code>	Active environment name (default: R_CONFIG_ACTIVE or "default")

**Value**

The settings as a list

---

settings_write	<i>Write project settings</i>
----------------	-------------------------------

---

### Description

Writes the project settings to settings.yml or config files

### Usage

```
settings_write(settings, settings_file = NULL, section = NULL)
```

### Arguments

settings	The settings list to write
settings_file	The settings file path (default: auto-detect settings.yml/config.yml)
section	Optional section to update (e.g. "data")

---

---

setup	<i>Setup Framework (First-Time Configuration)</i>
-------	---

---

### Description

Initializes Framework's global configuration and launches the GUI for first-time setup. This is the recommended entry point for new users.

### Usage

```
setup(port = 8080, browse = TRUE)
```

### Arguments

port	Port number to use (default: 8080)
browse	Automatically open browser (default: TRUE)

### Details

Use this function after installing Framework to:

- Set your author name and email
- Configure default packages for new projects
- Set IDE preferences (VS Code, RStudio)
- Configure other global defaults

### Value

Invisibly returns the plumber server object

**See Also**

[gui \(\)](#) for launching the GUI without initialization check

**Examples**

```
## Not run:
# First-time setup
framework::setup()

## End(Not run)
```

**standardize\_wd**

*Standardize Working Directory for Framework Projects*

**Description**

This function helps standardize the working directory when working with framework projects, especially useful in Quarto/RMarkdown documents that may be rendered from subdirectories.

**Usage**

```
standardize_wd(project_root = NULL)
```

**Arguments**

`project_root` Character string specifying the project root directory. If `NULL` (default), the function will attempt to find it automatically.

**Details**

The function looks for common framework project indicators:

- `settings.yml` or `settings.yaml` file
- `.Rprofile` file
- Being in common subdirectories (`scratch`, `work`)

It sets both the regular working directory and knitr's `root.dir` option if knitr is available.

**Value**

Invisibly returns the standardized project root path.

**Examples**

```
## Not run:
library(framework)
standardize_wd()
scaffold()

## End(Not run)
```

---

`status`

*Show Framework project status*

---

## Description

Displays comprehensive information about the current Framework project including:

- Framework package version
- Project configuration
- Git status
- AI assistant configuration
- Git hooks status
- Package dependencies
- Directory structure

## Usage

```
status()
```

## Examples

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

---

`storage_test`

*Test storage connection*

---

## Description

Validates that S3/storage credentials and bucket access are working.

## Usage

```
storage_test(connection = NULL)
```

## Arguments

`connection` Character or NULL. Connection name, or NULL for default.

## Value

Logical. TRUE if connection is valid.

## Examples

```
## Not run:
# Test default storage connection
storage_test()

# Test specific connection
storage_test("my_s3_backup")

## End(Not run)
```

`stubs_list`

*List Available Stubs*

## Description

Shows all available stub templates that can be used with `make_notebook()`.

## Usage

```
stubs_list(type = NULL)
```

## Arguments

type	Character. Filter by type: "quarto", "rmarkdown", "script", or NULL (all).
------	--

## Value

Data frame with columns: name, type, source (user/framework)

## Examples

```
## Not run:
# List all stubs
stubs_list()

# List only Quarto stubs
stubs_list("quarto")

# List only script stubs
stubs_list("script")

## End(Not run)
```

---

stubs\_path

*Get Path to Stub Templates Directory*

---

## Description

Returns the path to the user's stubs directory, or the framework stubs directory if no user stubs exist.

## Usage

```
stubs_path(which = "auto")
```

## Arguments

which	Character. Which directory to return:
	• "user" - User's project stubs directory (stubs/)
	• "framework" - Framework's built-in stubs directory
	• "auto" (default) - User directory if it exists, otherwise framework

## Value

Character path to stubs directory

## Examples

```
## Not run:  
# Get active stubs directory  
stubs_path()  
  
# Get framework stubs directory  
stubs_path("framework")  
  
# Get user stubs directory  
stubs_path("user")  
  
## End(Not run)
```

---

stubs\_publish

*Publish Stub Templates for Customization*

---

## Description

Copies framework stub templates to your project's stubs/ directory, allowing you to customize them. Similar to Laravel's artisan vendor:publish command.

## Usage

```
stubs_publish(type = "all", overwrite = FALSE, stubs = NULL)
```

## Arguments

<code>type</code>	Character vector. Which stub types to publish: <ul style="list-style-type: none"> <li>• "notebooks" - Quarto/RMarkdown notebook stubs</li> <li>• "scripts" - R script stubs</li> <li>• "all" - All stubs (default)</li> </ul>
<code>overwrite</code>	Logical. Whether to overwrite existing stubs. Default FALSE.
<code>stubs</code>	Character vector. Specific stub names to publish (e.g., "default", "minimal"). If NULL (default), publishes all stubs of the specified type.

## Details

### Stub Customization Workflow:

1. Publish stubs to your project: `stubs_publish()`
2. Edit stubs in `stubs/` directory to match your preferences
3. Use `make_notebook()` or `make_script()` - your custom stubs are used automatically

### Stub Naming Convention:

Stubs follow this naming pattern:

- Notebooks: `stubs/notebook-{name}.qmd` or `stubs/notebook-{name}.Rmd`
- Scripts: `stubs/script-{name}.R`

Framework searches user stubs first, then falls back to built-in stubs.

### Available Placeholders:

Stubs can use these placeholders:

- `{filename}` - File name without extension
- `{date}` - Current date (YYYY-MM-DD)

## Value

Invisible list of published file paths

## See Also

[make\\_notebook\(\)](#), [make\\_script\(\)](#), [stubs\\_list\(\)](#), [stubs\\_path\(\)](#)

## Examples

```
## Not run:
# Publish all stubs
stubs_publish()

# Publish only notebook stubs
stubs_publish("notebooks")

# Publish specific stub
stubs_publish(stubs = "default")

# Overwrite existing stubs
stubs_publish(overwrite = TRUE)

## End(Not run)
```

---

`view`*View data in an interactive browser viewer*

---

## Description

Opens an interactive, browser-based viewer for R objects. This is the primary function for viewing data frames, plots, lists, and other R objects with enhanced formatting.

## Usage

```
view(x, title = NULL, max_rows = 5000)
```

## Arguments

<code>x</code>	The data to view (data.frame, plot, list, function, or other R object)
<code>title</code>	Optional title for the view. If NULL, uses the object name.
<code>max_rows</code>	Maximum number of rows to display for data frames (default: 5000).

## Value

Invisibly returns NULL. Opens a browser window.

## Examples

```
## Not run:  
# View a data frame  
view(mtcars)  
  
# View with a title  
view(iris, title = "Iris Dataset")  
  
# View a ggplot  
library(ggplot2)  
p <- ggplot(mtcars, aes(mpg, hp)) + geom_point()  
view(p)  
  
## End(Not run)
```

---

`view_create`*Create an enhanced view of R objects in the browser*

---

## Description

Opens an interactive, browser-based viewer for R objects with syntax highlighting, tabbed interfaces, and enhanced data table support. Handles data frames, plots, lists, functions, and more with appropriate rendering for each type.

## Usage

```
view_create(x, title = NULL, max_rows = 5000)
```

**Arguments**

- `x` The data to view (data.frame, plot, list, function, or other R object)  
`title` Optional title for the view. If NULL, uses the object name.  
`max_rows` Maximum number of rows to display for data frames (default: 5000). Large data frames are automatically truncated with a warning.

**Value**

Invisibly returns NULL. Function is called for its side effect of opening a browser window with the rendered view.

**Examples**

```
## Not run:
# View a data frame with interactive table
view_create(mtcars)

# View a plot
library(ggplot2)
p <- ggplot(mtcars, aes(mpg, hp)) + geom_point()
view_create(p, title = "MPG vs HP")

# View a list with YAML and R structure tabs
view_create(list(a = 1, b = 2, c = list(d = 3)))

## End(Not run)
```

*view\_detail*

*View data with enhanced browser-based interface*

**Description**

Opens an interactive, browser-based viewer for R objects with search, filtering, sorting, pagination, and export capabilities (CSV/Excel). Provides a rich DataTables interface for data frames and enhanced views for plots, lists, and other R objects. This is the recommended function for exploring data in detail.

**Usage**

```
view_detail(x, title = NULL, max_rows = 5000)
```

**Arguments**

- `x` The data to view (data.frame, plot, list, function, or other R object)  
`title` Optional title for the view. If NULL, uses the object name.  
`max_rows` Maximum number of rows to display for data frames (default: 5000). Large data frames are automatically truncated with a warning.

## Details

Unlike R's built-in `View()`, this function:

- Works consistently across all IDEs (VS Code, RStudio, Positron, terminal)
- Provides search and column filtering
- Allows export to CSV and Excel
- Offers sorting and pagination
- Respects IDE-native viewers (doesn't override them)

## Value

Invisibly returns NULL. Function is called for its side effect of opening a browser window with the rendered view.

## See Also

[view](#)

## Examples

```
## Not run:  
# View a data frame with interactive table  
view_detail(mtcars)  
  
# View with custom title  
view_detail(iris, title = "Iris Dataset")  
  
# View a plot  
library(ggplot2)  
p <- ggplot(mtcars, aes(mpg, hp)) + geom_point()  
view_detail(p)  
  
# View a list  
view_detail(list(a = 1, b = 2, c = list(d = 3)))  
  
## End(Not run)
```

---

write\_frameworkrc    *Write global Framework configuration*

---

## Description

Write global Framework configuration

## Usage

`write_frameworkrc(config)`

## Arguments

<code>config</code>	List containing configuration to write
---------------------	--

**Value**

Invisibly returns NULL

---

`write_framework_template`

*Overwrite a Framework template with new contents*

---

**Description**

Overwrite a Framework template with new contents

**Usage**

```
write_framework_template(name, contents)
```

**Arguments**

<code>name</code>	Template identifier (e.g., "notebook", "gitignore", "ai_claude")
<code>contents</code>	Character string to write to the template file.

# Index

\* **internal**

- .apply\_auto\_fix, 8
- .build\_directory\_table, 8
- .calculate\_file\_hash, 9
- .catalog\_field\_default, 9
- .catalog\_find\_field, 9
- .catalog\_project\_type\_defaults, 10
- .check\_duckdb\_exists, 10
- .check\_env\_gitignored, 10
- .check\_git\_available, 11
- .check\_git\_history, 11
- .check\_git\_status, 11
- .check\_gitignore\_coverage, 11
- .check\_mysql\_exists, 12
- .check\_path\_ignored, 12
- .check\_postgres\_exists, 12
- .check\_private\_data\_exposure, 13
- .check\_sqlite\_exists, 13
- .check\_sqlserver\_exists, 13
- .cleanup\_gitkeep\_files, 14
- .collect\_all\_s3\_connections, 14
- .commit\_after\_scaffold, 14
- .configure\_git\_hooks, 15
- .connect\_duckdb, 15
- .connect\_mysql, 15
- .connect\_postgres, 16
- .connect\_sqlite, 16
- .connect\_sqlserver, 17
- .copy\_config\_files, 17
- .create\_ai\_files, 17
- .create\_ai\_instructions, 18
- .create\_code\_workspace, 18
- .create\_config\_file, 18
- .create\_dev\_rprofile, 19
- .create\_duckdb\_db, 19
- .create\_gitignore, 19
- .create\_ide\_configs, 20
- .create\_init\_file, 20
- .create\_initial\_commit, 20
- .create\_mysql\_db, 21
- .create\_postgres\_db, 21
- .create\_project\_config, 22
- .create\_project\_directories, 22
- .create\_renignore, 23
- .create\_rproj\_file, 23
- .create\_scaffold\_file, 23
- .create\_sqlite\_db, 24
- .create\_sqlserver\_db, 24
- .create\_stub\_files, 24
- .create\_template\_db, 25
- .create\_vscode\_settings, 25
- .create\_vscode\_workspace, 25
- .customize\_project\_files, 26
- .data\_spec\_update, 26
- .default\_directory\_table, 27
- .default\_render\_dirs\_for\_type, 27
- .default\_root\_render\_dir\_for\_type, 27
- .delete\_init\_file, 27
- .df\_to\_list\_of\_lists, 28
- .display\_next\_steps, 28
- .ensure\_framework\_db, 28
- .ensure\_framework\_template, 29
- .ensure\_output\_dir, 29
- .find\_project\_root, 29
- .find\_stub\_template, 30
- .framework\_catalog\_default\_path, 30
- .framework\_catalog\_user\_path, 30
- .framework\_templates\_dir, 31
- .generate\_data\_section, 31
- .generate\_environment\_section, 31
- .generate\_function\_reference, 31
- .generate\_header\_section, 32
- .generate\_hook\_script, 32
- .generate\_notes\_section, 32
- .generate\_packages\_section,

```

    33
  .generate_project_type_section,
    33
  .get_cache, 33
  .get_cache_dir, 34
  .get_data_directories, 34
  .get_data_pathSuggestions,
    34
  .get_data_record, 35
  .get_db_connection, 35
  .get_driver_info, 35
  .get_example_data_path, 36
  .get_hook_setting, 36
  .get_metadata, 36
  .get_notebook_dir_from_config,
    37
  .get_package_list_from_config,
    37
  .get_package_requirements, 38
  .get_placeholders, 38
  .get_scaffold_history, 38
  .get_settings_file, 39
  .git_available, 39
  .guess_content_type, 40
  .has_settings_file, 40
  .identify_private_dirs, 40
  .init_db, 41
  .init_git_repo, 41
  .init_renv, 41
  .init_standard, 42
  .install_package, 42
  .install_package_base, 43
  .install_package_renv, 43
  .install_required_packages,
    44
  .is_data_file, 44
  .is_git_repo, 44
  .is_initialized, 45
  .list_available_stubs, 45
  .load_ai_template, 46
  .load_configuration, 46
  .load_environment, 46
  .load_functions, 47
  .load_libraries, 47
  .load_template_content, 47
  .make_env, 48
  .mark_scaffolded, 48
  .migrate_config_from_previous,
    49
  .normalize_expire_after, 49
  .normalize_notebook_name, 50
  .normalize_package_spec, 50
  .parse_assistant_selection,
    51
  .parse_package_spec, 51
  .parse_sections, 52
  .pretty_print_config, 52
  .print_audit_summary, 52
  .prompt_ai_support_init, 53
  .prompt_ai_support_install,
    53
  .read_framework_template, 53
  .remove_cache, 54
  .remove_data, 54
  .remove_init, 54
  .remove_metadata, 55
  .replace_moustache_placeholders,
    55
  .replace_section, 55
  .require_driver, 56
  .require_git_repo, 56
  .reset_framework_template, 57
  .s3_client, 57
  .s3_public_url, 57
  .s3_upload_dir, 58
  .s3_upload_file, 58
  .save_audit_result, 59
  .save_result, 59
  .scan_for_orphaned_files, 59
  .set_cache, 60
  .set_data, 60
  .set_ggplot_theme, 61
  .set_metadata, 61
  .set_random_seed, 62
  .slugify, 62
  .sync_packages_to_renv, 63
  .to_kebab_case, 63
  .update_data_with_hash, 63
  .update_frameworkrc, 64
  .update_gitignore_for_renv,
    64
  .update_hook_config, 64
  .uses_split_file, 65
  .validate_driver, 65
  .validate_refresh, 66
  .validate_settings_catalog,
    66
  .write_framework_template, 66
bootstrap_project_init, 70
config, 74
configure_ai_agents, 74
configure_author, 75
configure_connection, 76
configure_data, 77

```

configure\_directories, 78  
configure\_packages, 80  
connection\_begin, 82  
connection\_check, 83  
connection\_check\_leaks, 83  
connection\_close\_all, 84  
connection\_commit, 84  
connection\_delete, 85  
connection\_find, 85  
connection\_find\_by, 86  
connection\_insert, 86  
connection\_pool, 87  
connection\_pool\_close, 88  
connection\_pool\_close\_all, 89  
connection\_pool\_list, 90  
connection\_restore, 90  
connection\_rollback, 91  
connection\_update, 91  
connection\_with\_pool, 92  
connection\_with\_transaction,  
    92  
data\_read\_or\_cache, 96  
data\_spec\_get, 97  
dot-has\_column, 105  
dot-list\_columns, 106  
dot-list\_tables, 107  
env\_default\_template\_lines,  
    108  
env\_lines\_from\_variables, 109  
env\_resolve\_lines, 109  
framework-package, 8  
framework\_template\_path, 110  
framework\_view, 110  
get\_default\_global\_config,  
    111  
get\_global\_setting, 111  
init\_global\_config, 122  
list\_framework\_templates, 122  
list\_metadata, 122  
project\_add\_directory, 138  
project\_create, 140  
renv\_restore, 150  
renv\_snapshot, 151  
renv\_status, 151  
renv\_sync, 151  
renv\_update, 152  
view\_create, 169  
view\_detail, 170  
.apply\_auto\_fix, 8  
.build\_directory\_table, 8  
.calculate\_file\_hash, 9  
.catalog\_field\_default, 9  
.catalog\_find\_field, 9  
.catalog\_project\_type\_defaults,  
    10  
.check\_duckdb\_exists, 10  
.check\_env\_gitignored, 10  
.check\_git\_available, 11  
.check\_git\_history, 11  
.check\_git\_status, 11  
.check\_gitignore\_coverage, 11  
.check\_mysql\_exists, 12  
.check\_path\_ignored, 12  
.check\_postgres\_exists, 12  
.check\_private\_data\_exposure, 13  
.check\_sqlite\_exists, 13  
.check\_sqlserver\_exists, 13  
.cleanup\_gitkeep\_files, 14  
.collect\_all\_s3\_connections, 14  
.commit\_after\_scaffold, 14  
.configure\_git\_hooks, 15  
.connect\_duckdb, 15  
.connect\_mysql, 15  
.connect\_postgres, 16  
.connect\_sqlite, 16  
.connect\_sqlserver, 17  
.copy\_config\_files, 17  
.create\_ai\_files, 17  
.create\_ai\_instructions, 18  
.create\_code\_workspace, 18  
.create\_config\_file, 18  
.create\_dev\_rprofile, 19  
.create\_duckdb\_db, 19  
.create\_gitignore, 19  
.create\_ide\_configs, 20  
.create\_init\_file, 20  
.create\_initial\_commit, 20  
.create\_mysql\_db, 21  
.create\_postgres\_db, 21  
.create\_project\_config, 22  
.create\_project\_directories, 22  
.create\_renvignore, 23  
.create\_rproj\_file, 23  
.create\_scaffold\_file, 23  
.create\_sqlite\_db, 24  
.create\_sqlserver\_db, 24  
.create\_stub\_files, 24  
.create\_template\_db, 25  
.create\_vscode\_settings, 25  
.create\_vscode\_workspace, 25  
.customize\_project\_files, 26  
.data\_spec\_update, 26  
.default\_directory\_table, 27  
.default\_render\_dirs\_for\_type, 27

```

.default_root_render_dir_for_type,
    27
.delete_init_file, 27
.df_to_list_of_lists, 28
.display_next_steps, 28
.ensure_framework_db, 28
.ensure_framework_template, 29
.ensure_output_dir, 29
.find_project_root, 29
.find_stub_template, 30
.framework_catalog_default_path,
    30
.framework_catalog_user_path, 30
.framework_templates_dir, 31
.generate_data_section, 31
.generate_environment_section, 31
.generate_function_reference, 31
.generate_header_section, 32
.generate_hook_script, 32
.generate_notes_section, 32
.generate_packages_section, 33
.generate_project_type_section,
    33
.get_cache, 33
.get_cache_dir, 34
.get_data_directories, 34
.get_data_pathSuggestions, 34
.get_data_record, 35
.get_db_connection, 35
.get_driver_info, 35
.get_example_data_path, 36
.get_hook_setting, 36
.get_metadata, 36
.get_notebook_dir_from_config, 37
.get_package_list_from_config, 37
.get_package_requirements, 38
.get_placeholders, 38
.get_scaffold_history, 38
.get_settings_file, 39
.git_available, 39
.guess_content_type, 40
.has_column(dot-has_column), 105
.has_settings_file, 40
.identify_private_dirs, 40
.init_db, 41
.init_git_repo, 41
.init_renv, 41
.init_standard, 42
.install_package, 42
.install_package_base, 43
.install_package_renv, 43
.install_required_packages, 44
.is_data_file, 44
.is_git_repo, 44
.is_initialized, 45
.list_available_stubs, 45
.list_columns(dot-list_columns),
    106
.list_tables(dot-list_tables),
    107
.load_ai_template, 46
.load_configuration, 46
.load_environment, 46
.load_functions, 47
.load_libraries, 47
.load_template_content, 47
.make_env, 48
.mark_scaffolded, 48
.migrate_config_from_previous, 49
.normalize_expire_after, 49
.normalize_notebook_name, 50
.normalize_package_spec, 50
.parse_assistant_selection, 51
.parse_package_spec, 51
.parse_sections, 52
.pretty_print_config, 52
.print_audit_summary, 52
.prompt_ai_support_init, 53
.prompt_ai_support_install, 53
.read_framework_template, 53
.remove_cache, 54
.remove_data, 54
.remove_init, 54
.remove_metadata, 55
.replace_moustache_placeholders,
    55
.replace_section, 55
.require_driver, 56
.require_git_repo, 56
.reset_framework_template, 57
.s3_client, 57
.s3_public_url, 57
.s3_upload_dir, 58
.s3_upload_file, 58
.save_audit_result, 59
.save_result, 59
.scan_for_orphaned_files, 59
.set_cache, 60
.set_data, 60
.set_ggplot_theme, 61
.set_metadata, 61
.set_random_seed, 62
.slugify, 62
.sync_packages_to_renv, 63

```

.to\_kebab\_case, 63  
.update\_data\_with\_hash, 63  
.update\_frameworkrc, 64  
.update\_gitignore\_for\_renv, 64  
.update\_hook\_config, 64  
.uses\_split\_file, 65  
.validate\_driver, 65  
.validate\_refresh, 66  
.validate\_settings\_catalog, 66  
.write\_framework\_template, 66  
  
add\_project\_to\_config, 67  
ai\_generate\_context, 67  
ai\_regenerate\_context, 68  
ai\_sync\_context, 69  
  
bootstrap\_project\_init, 70  
  
cache, 70  
cache\_flush, 71  
cache\_forget, 71  
cache\_get, 71  
cache\_list, 72  
cache\_remember, 72  
capture\_output, 73  
config, 74  
config(), 159  
configure\_ai\_agents, 74  
configure\_author, 75  
configure\_connection, 76  
configure\_data, 77  
configure\_directories, 78  
configure\_global, 79  
configure\_packages, 80  
connection\_begin, 82  
connection\_check, 83  
connection\_check\_leaks, 83  
connection\_close\_all, 84  
connection\_commit, 84  
connection\_delete, 85  
connection\_find, 85  
connection\_find\_by, 86  
connection\_insert, 86  
connection\_pool, 87  
connection\_pool\_close, 88  
connection\_pool\_close\_all, 89  
connection\_pool\_list, 90  
connection\_restore, 90  
connection\_rollback, 91  
connection\_update, 91  
connection\_with\_pool, 92  
connection\_with\_transaction, 92  
connections\_list, 82  
connections\_s3, 82  
data\_add, 93  
data\_info, 94  
data\_info(), 97  
data\_list, 95  
data\_read, 95  
data\_read\_or\_cache, 96  
data\_save, 96  
data\_spec\_get, 97  
db\_connect, 97  
db\_drivers\_install, 98  
db\_drivers\_status, 99  
db\_execute, 99  
db\_list, 100  
db\_query, 101  
db\_transaction, 101  
db\_with, 102  
docs\_export, 103  
dot\_has\_column, 105  
dot\_list\_columns, 106  
dot\_list\_tables, 107  
  
env\_clear, 108  
env\_default\_template\_lines, 108  
env\_lines\_from\_variables, 109  
env\_resolve\_lines, 109  
env\_summary, 109  
  
framework(*framework-package*), 8  
framework-package, 8  
framework\_template\_path, 110  
framework\_view, 110  
fw\_config\_dir, 111  
  
get\_default\_global\_config, 111  
get\_global\_setting, 111  
git\_add, 112  
git\_commit, 113  
git\_diff, 113  
git\_hooks\_disable, 114  
git\_hooks\_enable, 114  
git\_hooks\_install, 115  
git\_hooks\_list, 116  
git\_hooks\_uninstall, 116  
git\_log, 117  
git\_pull, 117  
git\_push, 118  
git\_security\_audit, 118  
git\_status, 120  
gui, 121  
gui(), 164  
  
init\_global\_config, 122

list\_framework\_templates, 122  
 list\_metadata, 122  
 load\_settings\_catalog, 123  
  
 make\_notebook, 123  
 make\_notebook(), 126–129, 168  
 make\_presentation, 125  
 make\_presentation(), 124, 125, 127  
 make\_qmd, 126  
 make\_qmd(), 124, 125, 127, 128  
 make\_revealjs, 127  
 make\_revealjs(), 124–126  
 make\_rmd, 128  
 make\_rmd(), 124–126  
 make\_script, 129  
 make\_script(), 168  
  
 new, 130  
 new(), 140  
 new\_course, 131  
 new\_course(), 130  
 new\_presentation, 132  
 new\_presentation(), 130  
 new\_project, 132  
 new\_project(), 130–132, 134, 140  
 new\_project\_sensitive, 134  
 new\_project\_sensitive(), 130  
 now, 135  
  
 outputs, 135  
  
 packages\_install, 135  
 packages\_list, 136  
 packages\_restore, 136  
 packages\_snapshot, 137  
 packages\_status, 137  
 packages\_update, 138  
 project\_add\_directory, 138  
 project\_create, 140  
 project\_create(), 133, 159  
 project\_info, 141  
 project\_list, 141  
 publish, 142  
 publish\_data, 143  
 publish\_dir, 143  
 publish\_list, 144  
 publish\_notebook, 145  
  
 quarto\_generate\_all, 146  
 quarto regenerate, 147  
  
 read\_framework\_template, 148  
 read\_frameworkrc, 147  
 remove\_project\_from\_config, 148  
  
 renv\_disable, 149  
 renv\_enable, 149  
 renv\_enabled, 150  
 renv\_restore, 150  
 renv\_snapshot, 151  
 renv\_status, 151  
 renv\_sync, 151  
 renv\_update, 152  
 reset\_framework\_template, 152  
 result\_list, 153  
  
 save\_figure, 153  
 save\_model, 155  
 save\_notebook, 155  
 save\_report, 157  
 save\_table, 157  
 scaffold, 158  
 scratch\_capture, 160  
 scratch\_clean, 161  
 settings, 74, 161  
 settings\_read, 162  
 settings\_write, 163  
 setup, 163  
 setup(), 121, 133  
 standardize\_wd, 164  
 standardize\_wd(), 159  
 status, 165  
 storage\_test, 165  
 stubs\_list, 166  
 stubs\_list(), 168  
 stubs\_path, 167  
 stubs\_path(), 168  
 stubs\_publish, 167  
  
 view, 169, 171  
 view\_create, 169  
 view\_detail, 170  
  
 write\_framework\_template, 172  
 write\_frameworkrc, 171