Package: renviron (via r-universe)

October 22, 2024

Title Environment Variable Management for R Projects

Version 0.5.1

Description The `renviron` package is an essential toolkit for managing environment variables in R projects, offering advanced capabilities for modifying, creating, and deleting variables within `.Renviron` and other custom configuration files. This package supports dynamic handling of environment variables directly from R, making it invaluable for projects that demand precise configuration management. Key features include the ability to specify custom file names for environment settings, selectively load variables, and ensure secure handling of sensitive data like API keys and database credentials. Whether updating single variables, managing global settings across multiple projects, or securely masking variable values for confidentiality, `renviron` provides robust functionality wrapped in an easy-to-use interface. This makes it an ideal solution for developers and data scientists seeking efficient and secure environment variable management.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Imports cli, fs, magrittr, readr, usethis

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Repository https://adatar-do.r-universe.dev

RemoteUrl https://github.com/adatar-do/renviron

RemoteRef HEAD

RemoteSha ec2d5cdd93ebc05de5949f45812089bbbb3b89b1

Contents

`renviron_add` *Add or update an environment variable in the environment file*

Description

This function adds a new environment variable or updates the value of an existing variable within the specified environment file (.Renviron by default), considering both user and project scopes. It offers the flexibility to either save the changes back to the environment file or to manipulate the variables in a more controlled manner by not saving (`in_place = FALSE`). Changes are immediately updated in the current R session's environment, regardless of the `in_place` setting.

Usage

```
renviron_add(key, value, .renviron = NULL, in_place = FALSE, ...)
```

Arguments

`key`	A character string specifying the name of the environment variable to add or update.
`value`	The new value for the environment variable, provided as a character string.
`.renviron`	An optional named list of environment variables to modify instead of loading the environment file using `renviron_load()`. This can be useful for testing or batch processing.
`in_place`	A logical flag indicating whether to save the changes back to the environment file (`TRUE`) or to return the modified list without saving (`FALSE`). Default is `FALSE`. When `FALSE`, the function allows for controlled manipulation of variables without permanently affecting the environment file.
`...`	Additional arguments: - `scope`: Specifies the scope(s) to search for the environment file when loading variables. Valid values are "user" and "project", searched in the provided order. Default is `c("user", "project")`. - `.file`: Specifies the filename to be considered as the environment file within the scope. Default is ".Renviron". - `confirm`: Indicates whether to confirm changes before saving the environment file. Default is `TRUE`.

Value

If `in_place` is `TRUE`, the function invisibly returns the modified list of environment variables after saving it to the environment file. If `in_place` is `FALSE`, it returns the modified list without saving, allowing further manipulation or inspection. The current R session's environment reflects the updated values.

Examples

```
## Not run:
# Add or update the CENSUS_API_KEY variable in the environment file and save it
renviron_add("CENSUS_API_KEY", "new_key_value", in_place = TRUE)

# Add or update the variable in a provided list without saving
env_list <- renviron_load()  # Load current environment variables
modified_env <- renviron_add("NEW_VAR", "some_value", .renviron = env_list, in_place = FALSE)
print(modified_env$NEW_VAR)  # Output the value of NEW_VAR

## End(Not run)
```

`renviron_delete` *Remove an environment variable from the environment file and system*

Description

This function removes a specified environment variable from the .Renviron file and the system environment. By default, the change is saved back to the .Renviron file (`in_place = TRUE`), but the function can also operate without saving (`in_place = FALSE`), which allows for temporary modification of environment variables for testing or other purposes. It is designed to handle both user and project scopes effectively when no custom list is provided.

Usage

```
renviron_delete(key, .renviron = NULL, in_place = FALSE, ...)
```

Arguments

`key`	A character string specifying the name of the environment variable to remove.
`.renviron`	An optional named list of environment variables from which to remove the specified variable. If provided, the function modifies this list instead of automatically loading the .Renviron file. This can be particularly useful for testing or handling temporary environment changes.
`in_place`	A logical flag indicating whether to save the changes back to the .Renviron file (`TRUE`) or just modify the environment variables temporarily (`FALSE`). The default is `TRUE`. When `FALSE`, changes are made only to the runtime environment and the provided list, without affecting the .Renviron file.

...	Additional arguments that are passed to the `renviron_save()` function if `in_place = TRUE`. This can include: - `scope`: A character vector specifying the scope(s) to search for the .Renviron file when loading environment variables. Valid values are "user" and "project", with "project" given priority if not specified. - `.file`: Specifies the filename to be considered as the environment file within the scope. Default is ".Renviron". - `confirm`: A logical flag indicating whether to confirm changes before saving the .Renviron file. Useful for preventing accidental modifications.

Value

If `in_place` is `TRUE`, the function invisibly returns the modified list of environment variables after saving it to the .Renviron file and removes the variable from the system environment. If `in_place` is `FALSE`, it returns the modified list without saving, allowing further manipulation or inspection, but still removes the variable from the system environment.

Examples

```
## Not run:
# Permanently remove the CENSUS_API_KEY variable from both the .Renviron file and system environment
renviron_delete("CENSUS_API_KEY")

# Temporarily remove the OBSOLETE_VAR from the current environment settings
without affecting the .Renviron file
env_list <- renviron_load() # Load current environment variables
modified_env <- renviron_delete("OBSOLETE_VAR", .renviron = env_list, in_place = FALSE)
print(modified_env) # OBSOLETE_VAR will not be part of this list

## End(Not run)
```

`renviron_exists` *Check if an environment variable exists*

Description

This function checks for the existence of a specified environment variable either in the .Renviron file or within a provided named list of environment variables. It is useful for validating configuration before attempting to use environment variables in your application.

Usage

```
renviron_exists(key, .renviron = NULL, ...)
```

Arguments

`key`	A character string specifying the name of the environment variable to check.
`.renviron`	An optional named list of environment variables to check against. If not provided, the function will load the variables from the .Renviron file using `renviron_load()`, considering the specified scope.
`...`	Additional arguments to `renviron_load()` when `.renviron` is not provided: - `scope`: A character vector specifying the scope(s) to search for the .Renviron file. Valid options are "user" and "project". The function searches in the order provided, defaulting to `c("user", "project")` if not specified. This allows for flexibility in determining where the environment variables are loaded from based on operational needs. - `.file`: Specifies the filename to consider as the environment file within the scope. This can be used to specify a different environment file if needed.

Value

TRUE if the environment variable exists in the specified scope or list, FALSE otherwise. This boolean value can be used to make decisions in scripts or applications based on the availability of required configuration.

Examples

```
## Not run:
# Check if the CENSUS_API_KEY variable exists in the .Renviron file within the default scope
exists <- renviron_exists("CENSUS_API_KEY")
print(exists)  # Prints TRUE if the variable exists, FALSE otherwise

# Check for the variable in a provided list
env_list <- list(CENSUS_API_KEY='12345', GITHUB_PAT='abcde')
exists_in_list <- renviron_exists("CENSUS_API_KEY", .renviron = env_list)
print(exists_in_list)  # Expected output: TRUE

## End(Not run)
```

`renviron_get` *Retrieve a specific environment variable from the environment file*

Description

This function fetches the value of a specified environment variable from the .Renviron file, considering both user and project scopes as defined by the `scope` argument when loading the file. If the `.renviron` argument is provided, it uses this list directly, bypassing the need to load from the file. This is particularly useful for accessing specific environment variables in a secure and controlled manner, especially during testing or when working with a dynamically modified environment.

Usage

```
renviron_get(key, .renviron = NULL, ...)
```

Arguments

`key`	A character string specifying the name of the environment variable to retrieve.
`.renviron`	An optional named list of environment variables to use instead of loading from the .Renviron file. This can be useful for testing or when working with a modified set of environment variables that has not been saved back to the file.
`...`	Additional arguments that are passed to the `renviron_load()` function: - `scope`: A character vector specifying the scope(s) to search for the .Renviron file. Valid values are "user" and "project", with "project" typically having precedence unless otherwise specified. This is used only when `.renviron` is `NULL`. The default is `c("user", "project")`. - `.file`: Specifies the filename to be considered as the environment file within the specified scope. Default is ".Renviron".

Value

The value of the environment variable corresponding to `key`, if it exists. Returns `NULL` if the key does not exist in the list or .Renviron file within the specified scope.

Examples

```
## Not run:
# Assuming the .Renviron file contains:
# CENSUS_API_KEY='12345'

# Retrieve the CENSUS_API_KEY value from the .Renviron file within the default scope
api_key <- renviron_get("CENSUS_API_KEY")
print(api_key)

# Retrieve a variable value from a provided list of environment variables
env_vars <- list(CENSUS_API_KEY='67890', GITHUB_PAT='fghij')
api_key_from_list <- renviron_get("CENSUS_API_KEY", .renviron = env_vars)
print(api_key_from_list)

## End(Not run)
```

`renviron_list` *List and mask environment variables*

Description

This function provides a secure way to view all environment variables from the .Renviron file or a provided list, with their values masked for security purposes. It can operate over both user and project scopes as defined by the `scope` argument when loading from the .Renviron file. This feature is useful for debugging or auditing without compromising sensitive information by exposing actual values of the variables.

Usage

```
renviron_list(.renviron = NULL, ...)
```

Arguments

`.renviron`	An optional named list of environment variables that should be listed. If not provided, the function will load the variables from the .Renviron file using `renviron_load()` and mask their values according to the specified scope.
`...`	Additional arguments that are passed to the `renviron_load()` function: - `scope`: A character vector specifying the scope(s) to search for the .Renviron file. Valid options are "user" and "project", with "project" typically having precedence unless otherwise specified. This determines which .Renviron file the variables are loaded from if `.renviron` is not provided. The default is `c("user", "project")`. - `.file`: Specifies the filename to be considered as the environment file within the specified scope. Default is ".Renviron".

Value

A named list of all environment variables with their values masked as "*****". This list includes all variables found in the specified scope if `.renviron` is not provided, or from the provided list otherwise. This method ensures that no sensitive data is displayed, while still allowing users to understand which variables are set.

Examples

```
## Not run:
# List and mask all environment variables from the .Renviron file within the default scope
renviron_list()

# Directly list and mask variables from a custom provided named list
custom_env <- list(API_KEY = "12345", SECRET = "s3cr3t")
masked_env <- renviron_list(.renviron = custom_env)
print(masked_env)

## End(Not run)
```

`renviron_load` *Load Environment Variables from the .Renviron File*

Description

This function reads the .Renviron file, considering both user and project scopes as defined by the `scope` argument. It loads the variables defined within the file into a named list and sets them in the system environment. Only lines that follow the pattern key='value' or key="value" (where quotes are optional and can be single or double) are processed. Variables should be in the `KEY=VALUE` format, with optional outer quotes around the value. Lines that do not match this format are ignored.

Usage

```
renviron_load(
  scope = c("user", "project"),
  .file = ".Renviron",
  .vars = NULL,
  verbosity = 1,
  ...
)
```

Arguments

`scope`	A character vector specifying the scope(s) to search for the .Renviron file. Valid values are "user" and "project". The function searches in the order provided. The default order is `c("user", "project")`. The "user" scope refers to the user's home directory, while the "project" scope refers to the current project directory.
`.file`	Optional filename to search within the specified scope.
`.vars`	Optional character vector specifying which variables to load; if NULL, all variables are loaded.
`verbosity`	An integer specifying the level of verbosity in messages. Default is 1.
`...`	Additional arguments passed to other internal functions.

Value

A named list of environment variables set in the system environment, invisibly returned.

Examples

```
## Not run:
# Load and set environment variables from the .Renviron file
env_vars <- renviron_load()

# Load and set specific environment variables from the .Renviron file
env_vars <- renviron_load(.vars = c("CENSUS_API_KEY", "GITHUB_PAT"))

# Load and set environment variables from a custom file
env_vars <- renviron_load(.file = ".env")

# Load and set environment variables from the user scope only
env_vars <- renviron_load(scope = "user")

## End(Not run)
```

renviron_path *Retrieve the path to the .Renviron file*

Description

This function determines the path to a specified environment file (by default, .Renviron) by considering both user and project scopes, utilizing an internal approach inspired by `usethis:::scoped_path_r`. It prioritizes the project-specific file if present; otherwise, it falls back to the user-level file. The function allows specifying the desired scope to directly target either the user-level or project-level file. Additionally, a specific filename can be specified, allowing for flexible file management.

Usage

```
renviron_path(
  scope = c("user", "project"),
  .file = ".Renviron",
  verbosity = 1,
  ...
)
```

Arguments

`scope`	A character string or vector specifying the scope to search for the environment file. Valid values are "user" and "project". If both are provided, the function will search in the order provided. The default order is c("user", "project"). The "user" scope refers to the user's home directory, while the "project" scope refers to the current project directory.
`.file`	The name of the environment file to search for within the specified scope(s). The default is '.Renviron'. This allows the function to be used to find other environment files as needed.
`verbosity`	An integer specifying the level of verbosity. The default is 1, which prints a message when the file is found. A value of 0 suppresses all messages.
`...`	Additional parameters passed to the internal path finding function.

Value

A character string representing the path to the specified environment file within the chosen scope. If the file exists in the 'project' scope and "project" is included in the scope parameter, that path will be returned; otherwise, it will return the path to the user-level file. If no file is found, the function will return NULL.

Examples

```
## Not run:
# To get the path to the user-level .Renviron file:
user_env_path <- renviron_path("user")
print(user_env_path)
```

```
# To get the path to the project-level .Renviron file, if it exists:
project_env_path <- renviron_path("project")
print(project_env_path)

# To search for a different file first in the user scope, then in the project scope:
custom_file_path <- renviron_path(c("user", "project"), .file = ".myenv")
print(custom_file_path)

## End(Not run)
```

---

renviron_save *Save environment variables to the environment file*

Description

This function takes a named list of environment variables and saves them to the appropriate .Renviron file based on the specified scope. The function prompts for user confirmation before overwriting the file if `confirm` is set to TRUE, helping to prevent accidental data loss. This feature is particularly useful for programmatically updating or setting environment variables that need to persist across R sessions.

Usage

```
renviron_save(.Renviron, confirm = TRUE, ...)
```

Arguments

`.Renviron`	A named list where each name is an environment variable key and each value is the corresponding value for that key. The function expects the list values to be character strings. Example: `list(CENSUS_API_KEY = "12345", GITHUB_PAT = "abcde")`.
`confirm`	Logical; if TRUE, the function prompts for user confirmation before overwriting the .Renviron file. This helps to prevent unintended overwrites. Default is TRUE. Set this to FALSE for automated scripts or non-interactive session where confirmation is not desired.
`...`	Additional arguments to configure the operation: - `scope`: A character vector specifying the scope(s) where the .Renviron file is located. Valid options are "user" for the user-level file or "project" for the project-level file. Default is "project". The function saves to the .Renviron file in the specified scope. - `.file`: Optionally specify a different filename to save the environment variables to. Default is ".Renviron".

Value

This function does not return a value and operates invisibly, with the primary side effect being the modification of the .Renviron file.

Examples

```
## Not run:
# Assume you want to set or update environment variables
new_env <- list(CENSUS_API_KEY = "12345", GITHUB_PAT = "abcde")

# Save these variables to the .Renviron file, with confirmation
renviron_save(new_env, confirm = TRUE)

# Automatically save variables without confirmation for automated scripts
renviron_save(new_env, confirm = FALSE)

## End(Not run)
```

`renviron_unset_all` *Unset all environment variables from the environment file*

Description

This function unsets all environment variables that are loaded from the specified .Renviron file within the given scope. It is particularly useful for cleaning up the environment in R sessions during testing or after loading configurations that are no longer needed.

Usage

```
renviron_unset_all(...)
```

Arguments

`...`	Additional arguments to `renviron_load()`: - `scope`: A character vector specifying the scope(s) to search for the .Renviron file. Valid values are "user" and "project". The function searches in the order provided. The default order is `c("user", "project")`. - `.file`: Optionally specify a different filename to use within the specified scope. Defaults to ".Renviron".

Value

Invisibly returns NULL after unsetting the variables.

Examples

```
## Not run:
renviron_unset_all(scope = "project")
renviron_unset_all(scope = "user", .file = "custom.env")

## End(Not run)
```

scoped_path_r *Determine Scoped Path for R Environments*

Description

This function is a utility to find paths scoped to the user or the current project. It is based on the internal function `scoped_path_r` from the `usethis` package. It has been adapted for use here with proper attribution.

Usage

```
scoped_path_r(scope = c("user", "project"), ..., envvar = NULL)
```

Arguments

`scope`	A character string specifying the scope of the path. Options are "user" for user-level configuration, or "project" for project-level.
`...`	Additional arguments passed to `path`.
`envvar`	An optional environment variable that can specify a path.

Details

This function is particularly useful for managing paths in user or project specific configurations, such as .Renviron files.

Value

A string representing the path scoped as specified.

References

Function adapted from `usethis:::scoped_path_r` for demonstration purposes. usethis package: https://usethis.r-lib.org/

Examples

```
## Not run:
# Get the path to the user-level R configuration
scoped_path_r("user")

# Get the path to the current project's root directory
scoped_path_r("project")

## End(Not run)
```

Index