Quickstart Guide: zzrenvcheck

Ronald (Ryy) G. Thomas

2025-12-26

Overview

zzrenvcheck validates that all R packages used in your code are properly declared in DESCRIPTION and locked in renv.lock, maintaining reproducible environments for collaborative research.

Installation

```
# Install from GitHub
remotes::install_github("rgt47/zzrenvcheck")
```

```
library(zzrenvcheck)
```

Basic Usage

Check Your Project

```
# Validate current project
check_packages()
```

This scans your code for package usage and compares against DESCRIPTION and renv.lock.

Auto-Fix Missing Packages

```
# Add missing packages automatically
fix_packages()
```

This queries CRAN/Bioconductor to find and add missing packages.

Generate Status Report

```
# Get detailed status
report <- report_packages()
print(report)

# Output:
#   package         in_code in_description in_renv_lock status
#   dplyr           TRUE    FALSE          FALSE        missing_description
#   ggplot2         TRUE    TRUE           TRUE         ok
```

Key Features

Package Detection Patterns

The package detects packages from:

- `library(dplyr)` - Direct library calls
- `require(ggplot2)` - Conditional loading
- `tidyr::pivot_longer()` - Namespace calls
- `#' @importFrom dplyr filter` - Roxygen imports

Smart Filtering

19 filters avoid false positives:

- Base R packages (base, utils, stats)
- Placeholder names (myproject, package, foo)
- Generic words (my, your, file, path)
- Invalid package names

Validation Modes

```
# Standard mode (R/, scripts/, analysis/)
check_packages(strict = FALSE)

# Strict mode (includes tests/, vignettes/, inst/)
check_packages(strict = TRUE)
```

Sync to Code

Make code the single source of truth:

```
# Sync DESCRIPTION and renv.lock to match code
sync_packages()

# Preview changes first
sync_packages(dry_run = TRUE)
```

This adds missing packages and removes unused ones.

Package Source Validation

Check if packages exist on CRAN, Bioconductor, or GitHub:

```
# Single package
is_installable("dplyr")
# Returns: list(installable = TRUE, source = "CRAN", package = "dplyr")

# Batch validation
check_installable(c("dplyr", "DESeq2", "NonExistent"))
```

Workflow Integration

Pre-Commit

```
# In pre-commit hook
Rscript -e 'zzrenvcheck::check_packages(auto_fix = FALSE)'
```

CI/CD Pipeline

```yaml
# .github/workflows/check.yaml
- name: Check dependencies
  run: Rscript -e 'zzrenvcheck::check_packages()'
```

Makefile Integration

```
check-renv:
    Rscript -e 'zzrenvcheck::check_packages()'
```

Standalone Shell Script

For environments without R:

```
# Install shell script version
cd zzrenvcheck
./install.sh --prefix ~/bin

# Use anywhere
zzrenvcheck --fix --strict --verbose
```

Features:

- No R installation required
- Uses grep, sed, awk, jq, curl
- Perfect for CI/CD pipelines

Function Reference

Function	Purpose
`check_packages()`	Main validation
`fix_packages()`	Auto-fix wrapper
`sync_packages()`	Sync to code
`report_packages()`	Status report
`clean_description()`	Remove unused
`is_installable()`	Check CRAN/Bioconductor
`extract_code_packages()`	Scan code

Common Workflows

Fix Missing Packages

```
# Check and fix
result <- check_packages(auto_fix = TRUE)

# Then commit
system("git add DESCRIPTION renv.lock")
system('git commit -m "Add missing packages"')
```

Cleanup Unused Packages

```r
# Remove packages no longer in code
clean_description()
sync_packages()
```

Programmatic Use

```r
result <- check_packages(auto_fix = FALSE)

if (result$status == "fail") {
  cat("Missing packages:\n")
  print(result$missing_in_description)
}
```

Next Steps

- `?check_packages` - Main function documentation
- `?sync_packages` - Sync workflow details
- GitHub: https://github.com/rgt47/zzrenvcheck