

# Package ‘gDRstyle’

May 7, 2024

**Type** Package

**Title** A package with style requirements for the gDR suite

**Version** 1.2.0

**Date** 2024-03-25

**Description** Package fills a helper package role for whole gDR suite. It helps to support good development practices by keeping style requirements and style tests for other packages. It also contains build helpers to make all package requirements met.

**License** Artistic-2.0

**LazyLoad** yes

**Depends** R (>= 4.2)

**Imports** BiocCheck, BiocManager, checkmate, desc, git2r, lintr (>= 3.0.0), rcmdcheck, remotes, yaml, rjson, pkgbuild, withr

**Suggests** BiocStyle, knitr, testthat (>= 3.0.0)

**URL** <https://github.com/gdrplatform/gDRstyle>,  
<https://gdrplatform.github.io/gDRstyle/>

**BugReports** <https://github.com/gdrplatform/gDRstyle/issues>

**biocViews** Software, Infrastructure

**VignetteBuilder** knitr

**ByteCompile** TRUE

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.1

**Encoding** UTF-8

**git\_url** <https://git.bioconductor.org/packages/gDRstyle>

**git\_branch** RELEASE\_3\_19

**git\_last\_commit** 417070b

**git\_last\_commit\_date** 2024-04-30

**Repository** Bioconductor 3.19

**Date/Publication** 2024-05-07

**Author** Allison Vuong [aut],  
 Dariusz Scigocki [aut],  
 Marcin Kamianowski [aut],  
 Aleksander Chlebowski [ctb],  
 Janina Smola [aut],  
 Arkadiusz Gladki [cre, aut] (<<https://orcid.org/0000-0002-7059-6378>>),  
 Bartosz Czech [aut] (<<https://orcid.org/0000-0002-9908-3007>>)

**Maintainer** Arkadiusz Gladki <gladki.arkadiusz@gmail.com>

## Contents

gDRstyle-package . . . . .	2
avoid_new_lines . . . . .	3
checkDependencies . . . . .	4
checkPackage . . . . .	5
compare_versions . . . . .	6
installAllDeps . . . . .	6
installLocalPackage . . . . .	7
lintPkgDirs . . . . .	8
roxygen_tag_linter . . . . .	9
test_notes_check . . . . .	9

**Index** **12**

---

gDRstyle-package	<i>gDRstyle: A package with style requirements for the gDR suite</i>
------------------	--

---

## Description

Package fills a helper package role for whole gDR suite. It helps to support good development practices by keeping style requirements and style tests for other packages. It also contains build helpers to make all package requirements met.

## Value

package help page

## Note

To learn more about functions start with `help(package = "gDRstyle")`

## Author(s)

**Maintainer:** Arkadiusz Gladki <gladki.arkadiusz@gmail.com> ([ORCID](#))

Authors:

- Allison Vuong

- Dariusz Scigocki
- Marcin Kamianowski
- Janina Smola
- Bartosz Czech ([ORCID](#))

Other contributors:

- Aleksander Chlebowski [contributor]

## See Also

Useful links:

- <https://github.com/gdrplatform/gDRstyle>
- <https://gdrplatform.github.io/gDRstyle/>
- Report bugs at <https://github.com/gdrplatform/gDRstyle/issues>

---

avoid_new_lines	<i>Avoid new lines in sprintf output. Function helps to avoid line length limits without affecting sprintf output</i>
-----------------	---

---

## Description

Avoid new lines in sprintf output. Function helps to avoid line length limits without affecting sprintf output

## Usage

```
avoid_new_lines(fmt)
```

## Arguments

fmt	string, formatted as sprintf input
-----	------------------------------------

## Value

string

## Examples

```
sprintf(avoid_new_lines(  
    "Lorem ipsum dolor sit amet, %s adipiscing elit, sed do eiusmod  
    tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim  
    veniam."  
), "consectetur")
```

---

checkDependencies	<i>Check for alignment of package dependencies across Rplatform and package specifications.</i>
-------------------	---

---

### Description

Check the package dependency version specifications in the rplatform/dependencies.yaml and DESCRIPTION.

### Usage

```
checkDependencies(  
  dep_path,  
  desc_path,  
  skip_pkgs = "R",  
  combo_path = "/mnt/vol/dependencies_combo.yaml"  
)
```

### Arguments

dep_path	String of path to the rplatform dependencies.yaml file.
desc_path	String of the path to the package DESCRIPTION file.
skip_pkgs	vector of packages from DESCRIPTION to skip; defaults to R
combo_path	String of path to the combo image dependencies.yaml file. Defaults to the current directory.

### Details

This function is used for its side effects in the event that there are dependency clashes.

### Value

NULL invisibly.

### Examples

```
checkDependencies(  
  dep_path =  
    system.file(package = "gDRstyle", "testdata", "dependencies.yaml"),  
  desc_path = system.file(package = "gDRstyle", "DESCRIPTION"),  
  skip_pkgs = c("testthat", "lintr")  
)
```

---

checkPackage	<i>Check R package</i>
--------------	------------------------

---

## Description

Used in gDR platform packages' CI/CD pipelines to check that the package abides by gDRstyle stylistic requirements, passes rcmdcheck, and ensures that the dependencies.yml file used to build gDR platform's docker image is kept up-to-date with the dependencies listed in the package's DESCRIPTION file.

## Usage

```
checkPackage(  
  pkgName,  
  repoDir,  
  subdir = NULL,  
  fail_on = "warning",  
  bioc_check = FALSE,  
  run_examples = TRUE,  
  skip_lint = FALSE,  
  skip_tests = FALSE,  
  build_vignettes = TRUE,  
  check_vignettes = TRUE,  
  as_cran = FALSE  
)
```

## Arguments

pkgName	String of package name.
repoDir	String of path to repository directory.
subdir	String of relative path to the R package root directory from the repoDir.
fail_on	String specifying the level at which check fail. Supported values: "note", "warning" (default) and "error".
bioc_check	Logical whether bioc check should be performed
run_examples	Logical whether examples check should be performed
skip_lint	skip lint checks
skip_tests	skip tests
build_vignettes	build vignettes
check_vignettes	check vignettes
as_cran	run with as_cran flag

**Value**

NULL invisibly.

**Examples**

```
checkPackage(
  pkgName = "fakePkg",
  repoDir = system.file(package = "gDRstyle", "tst_pkgs", "dummy_pkg"),
  fail_on = "error"
)
```

---

compare_versions	<i>Compare listed package versions dependencies.</i>
------------------	--

---

**Description**

Compare listed package versions in the dependencies.yaml file as compared to the package DESCRIPTION file.

**Usage**

```
compare_versions(rp, desc)
```

**Arguments**

rp	Named list of package version requirements specified by rplatform dependencies.yaml.
desc	Named list of package version requirements specified by package DESCRIPTION file.

**Value**

Character vector of any misaligned package versions between rplatform dependencies.yaml and package DESCRIPTION.

---

installAllDeps	<i>Install all package dependencies from yaml file for building image purposes</i>
----------------	--

---

**Description**

Install all package dependencies from yaml file for building image purposes

**Usage**

```
installAllDeps(
  additionalRepos = NULL,
  base_dir = "/mnt/vol",
  use_ssh = FALSE,
  test_mode = FALSE
)
```

**Arguments**

additionalRepos	List of additional Repos
base_dir	String of base working directory.
use_ssh	logical, if use ssh keys
test_mode	logical, whether to run the function in the test mode (if TRUE the dependencies are not installed but only listed)

**Value**

NULL

**Examples**

```
installAllDeps(
  base_dir = system.file(package = "gDRstyle", "testdata"),
  test_mode = TRUE
)
```

---

installLocalPackage     *Install locally cloned repo for building image purposes*

---

**Description**

Install locally cloned repo for building image purposes

**Usage**

```
installLocalPackage(repo_path, additionalRepos = NULL, base_dir = "/mnt/vol")
```

**Arguments**

repo_path	String of repository directory.
additionalRepos	List of additional Repos
base_dir	String of base working directory.

**Value**

NULL

**Examples**

```
installLocalPackage(system.file(
  package = "gDRstyle", "tst_pkgs", "dummy_pkg"
))
```

---

lintPkgDirs

*Lint select subdirectories in a package directory.*

---

**Description**

Lint select subdirectories in a package directory.

**Usage**

```
lintPkgDirs(pkg_dir = ".", shiny = FALSE)
```

**Arguments**

pkg_dir	String of path to package directory.
shiny	Boolean of whether or not a shiny directory should also be lint. Defaults to the current directory.

**Details**

Will look for files in the following directories: "R", "tests", and conditionally "inst/shiny" if shiny is TRUE.

**Value**

NULL invisibly.

**Examples**

```
lintPkgDirs(
  pkg_dir= system.file(package = "gDRstyle", "tst_pkgs", "dummy_pkg"))
```



---

roxygen\_tag\_linter      *roxygen\_tag\_linter*

---

### Description

Check that function has documented specific tag in Roxygen skeleton (default @author).

### Usage

```
roxygen_tag_linter(tag = "@author")
```

### Arguments

tag                      character (default @author)

### Value

linter class function

### Author(s)

Kamil Foltynski [kamil.foltynski@contractors.roche.com](mailto:kamil.foltynski@contractors.roche.com)

### Examples

```
linters_config <- lintr::linters_with_defaults(  
  line_length_linter = lintr::line_length_linter(120),  
  roxygen_tag_linter = roxygen_tag_linter()  
)
```

test\_notes\_check

*Assume there is a valid note:*

```
> checking R code for possible problems ... NOTE mini_app:
no visible binding for '<<-' assignment to 'CONFIG'
```

*and we want every other note in this section (and others) to fail check. Accepted NOTE has 2 lines, therefore the length = 2. Then we want to check whether the content of this NOTE is correct, so we take one of the lines (eg. index\_to\_check = 2) and grep for content of this line (eg. text\_to\_check = "assignment to" ) This will result in any other NOTE failing check Take:*

```
list( list(length = 2, index_to_check = 2, text_to_check =
"assignment to") ) “ then following NOTE will be treated
as invalid > checking R code for possible problems ...
NOTE mini_app: no visible binding for '<<-' assignment to
'CONFIG' sandbox_app : sandboxUI: no visible binding for
global variable 'pcg_path' Undefined global functions or
variables: pcg_path
```

**Description**

Assume there is a valid note:

```
> checking R code for possible problems ... NOTE
mini_app: no visible binding for '<<-' assignment to 'CONFIG'
```

and we want every other note in this section (and others) to fail check. Accepted NOTE has 2 lines, therefore the length = 2. Then we want to check whether the content of this NOTE is correct, so we take one of the lines (eg. index\_to\_check = 2) and grep for content of this line (eg. text\_to\_check = "assignment to" ) This will result in any other NOTE failing check Take:

```
list(
  list(length = 2, index_to_check = 2, text_to_check = "assignment to")
)
```

```
then following NOTE will be treated as invalid
> checking R code for possible problems ... NOTE
mini_app: no visible binding for '<<-' assignment to 'CONFIG'
sandbox_app : sandboxUI: no visible binding for global variable
'pcg_path'
Undefined global functions or variables:
pcg_path
```

**Usage**

```
test_notes_check(check_results, bioccheck_results, valid_notes_list)
```

*test\_notes\_check*

11

**Value**

NULL

# Index

## \* **check**

checkDependencies, 4  
checkPackage, 5

## \* **install**

installAllDeps, 6  
installLocalPackage, 7

## \* **internal**

compare\_versions, 6  
gDRstyle-package, 2  
test\_notes\_check, 10

## \* **linter**

avoid\_new\_lines, 3  
lintPkgDirs, 8  
roxygen\_tag\_linter, 9

avoid\_new\_lines, 3

checkDependencies, 4  
checkPackage, 5  
compare\_versions, 6

gDRstyle (gDRstyle-package), 2  
gDRstyle-package, 2

installAllDeps, 6  
installLocalPackage, 7

lintPkgDirs, 8

roxygen\_tag\_linter, 9

test\_notes\_check, 9