Documentation

Contents

Installation	 	 		•	 •	 •	•	 •		•	•	 •	•		•		•		•	 •	•	•	1
Usage	 	 													•		•						1

Installation

Before anything, make sure the *DependencyReviewer* package is installed.

$\mathbf{remotes}$

The latest version is usually available on GitHub, and is installable with the *remotes* package.

```
# If you do not have remotes installed:
install.packages("remotes")
```

```
# Install DependencyReviewer with remotes:
remotes::install_github("darwin-eu/DependencyReviewer")
```

install.packages

DependencyReviewer 1.0.0 is also available on CRAN, and can be installed using install.packages as well.

install.packages("DependencyReviewer")

Usage

library(DependencyReviewer)

```
# Other packages that are used in the examples
library(DT)
library(ggplot2)
library(dplyr)
library(igraph)
library(GGally)
```

getDefaultPermittedPackages

What does it do? The getDefaultPermittedPackages function retrieves a list of packages from several on,- and offline data sources. These data sources include:

- 1. Base packages with a high priority installed.packages(lib.loc = .Library, priority =
 "high")
- 2. Tidyverse packages
- 3. OHDSI/HADES packages
- 4. Packages hosted on the DependencyReviewerWhitelists repository
- 5. Finally the function will also retrieve the defined packages' dependencies recursively, and add them to the list.

These packages are deemed OK to use. This list will, and should change overtime as packages become outdated, get replaced, or added to the list.

What does it need? getDefaultPermittedPackages does not require any arguments.

What does it return? getDefaultPermittedPackages returns a class of data.frame with columns: *package* and *version*

datatable(getDefaultPermittedPackages())
#> PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, pleas

Show 10 🗸	entries	Search:
	package	version
1	checkmate	2.1.0
2	desc	1.4.2
3	DT	0.26
4	ggraph	2.1.0
5	glue	1.6.2
6	here	1.0.1
7	igraph	1.3.5
8	shiny	1.7.3
9	shinyAce	0.4.2
10	shinyjs	2.1.0
Showing 1 to 10	of 14 entries	Previous 1 2 Next

checkDependencies

What does it do? Now that we have defined our 'whitelisted' packages, checkDependencies allows us to check our currently used dependencies against it. checkDependencies will run getDefaultPermittedPackages internally so there is no need to run the two separately to check your dependencies against the white list.

What does it need? checkDependencies has two optional arguments:

- 1. packageName default (NULL): Expects a character string of a package name. Example: "ggplot2".
- 2. dependencyType default (c("Imports", "Depends")): Expects a character vector of at least length 1 of dependency types. The supported types are: "Imports", "Depends", and "Suggests".

Because both arguments are optional it can also be run without specifying anything. The function will then assume that it is run **inside** a package-project environment. This is specifically useful when working on, or reviewing a package.

What does it return? checkDependencies prints out a message in the console that informs the user if all their used package dependencies are whitelisted or not. If not it instructs the user where to go to request the packages to be whitelisted.

```
# Assumes the current environment is a package-project
# Defaults are: packageName = NULL, packageTypes = c("Imports", "Depends")
checkDependencies()
# Check dependencies for installed package "dplyr"
checkDependencies(
    packageName = "dplyr"
)
```

1. If packages are not approved yet:

```
# Check Imports and Suggests
checkDependencies(
 packageName = "dplyr",
 dependencyType = c("Imports", "Suggests")
)
#>
#> -- Checking if packages in Imports and Suggests have been approved --
#>
#> ! Found 33 packages in Imports and Suggests that are not
#> approved
#> > 1) cli
#> > 2) generics
#> >
      3) lifecycle
#> >
      4) magrittr
#> >
     5) methods
#> > 6) pillar
#> >
     7) R6
#> > 8) rlang
#> > 9) tibble
#> > 10) tidyselect
```

#>	>	11)	utils
#>	>	12)	vctrs
#>	>	13)	bench
#>	>	14)	broom
#>	>	15)	callr
#>	>	16)	cour
#>	>	17)	DBI
#>	>	18)	dbplyr
#>	>	19)	ggplot2
#>	>	20)	knitr
#>	>	21)	Lahman
#>	>	22)	lobstr
#>	>	23)	microbenchmark
#>	>	24)	nycflights13
#>	>	25)	purrr
#>	>	26)	rmarkdown
#>	>	27)	RMySQL
#>	>	28)	RPostgreSQL
#>	>	29)	RSQLite
#>	>	30)	stringi
#>	>	31)	testthat
#>	>	32)	tidyr
#>	>	33)	withr
#>	!	Pleas	e create a new issue at https://github.com/mvankessel-EMC/DependencyReviewerWhitelists/ to re
#>	>	/pack	ige version date downloads_last_month license url
#>	1:		/:/:/:/:/:/:/:/

As you can see, it returns a list of all the packages that are not white listed. Below the list it will display some information in a *markdown table* format. This will come in handy later on. The table has six columns: 1) package, 2) version, 3) date, 4) downloads_last_month, 5) license, and 6) url.

Note that only packages available on CRAN are reported in the table. Non-CRAN packages will still show up in the list.

2. If all packages are approved:

```
# Only check directly imported dependencies of installed package "dplyr"
checkDependencies(
  packageName = "dplyr",
  dependencyType = c("Imports")
)
#>
#> -- Checking if package in Imports have been approved --
#>
#> ! Found 12 packages in Imports that are not
#> approved
#> >
     1) cli
#> >
      2) generics
#> >
      3) lifecycle
#> >
      4) magrittr
     5) methods
#> >
#> >
      6) pillar
#> > 7) R6
```

```
#> > 8) rlang
#> > 9) tibble
#> > 10) tidyselect
#> > 11) utils
#> > 12) vctrs
#> ! Please create a new issue at https://github.com/mvankessel-EMC/DependencyReviewerWhitelists/ to re
#> > /package /version /date / downloads_last_month/license /url /
#> /:-----/:---/:---/:---/
```

Notice how "Imports" and "Depends" packages of dplyr are whitelisted, but "Suggests" packages are not.

Requesting packages to be whitelisted

If you find that some packages are not yet whitelisted, you can request them to be. The DependencyReviewerWhitelists repository on GitHub houses the white list for *DependencyReviewer*.

To request new packages a new issue can be created on this repository.

Assuming we have the following output from checkDependencies:

```
Get from temp file
```

Checking if packages in Imports and Suggests have been approved

! Found 3 packages in Imports and Suggests that are not approved 1) GGally **→** 2) lintr 3) pak ! Please create a new issue at https://github.com/mvankessel-EMC/DependencyReviewerWhitelists/ to reque downloads_last_month|license |package |version |date lurl 86657|GPL (>= 2.0) |GGally |2.1.2 2021-06-21 03:40:10 https://ggobi.github. llintr 13.0.2 2022-10-19 08:52:37 61729|MIT + file LICENSE |https://github.com/r-39420|GPL-3 https://pak.r-lib.org 0.3.1 |2022-09-08 20:30:02 | pak

When creating a new issue, a request template is available.



Figure 1: Request template button

This template asks for some basic information about the requested packages, and a reason as to why the requested packages should be whitelisted.

Initially it displays some dummy information as to what a request might look like.

Firstly it asks us is to supply a table in markdown format with some basic information about the packages. We can copy this from the output message from the checkDependencies function.

Then it asks us to give a description as to why we would like these packages to be whitelisted.

Finally, we can add some additional information if required.

lssue: Request				
Suggest packages for the whitelist. If this doesn't lool	k right, choose a different type.			
My Request				
Write Preview		н в <i>І</i>	\equiv \diamond \mathscr{O} \equiv $\frac{1}{2}$	≣ 55 @ 57 ←
Table of packages to request in markdo	own format (package, version, d	late, downloads_last_n	onth, license, url).	
package version date down : : :	loads_last_month license : :	url :		
checkmate 2.1.0 2022-04-21 06:30:05 https://github.com/mllg/checkmate desc 1.4.2 2022-09-08 09:52:55	277387 BSD_3_clause 581711 MIT + file LICEN	+ file LICENSE https:/	//mllg.github.io/check p.com/r-lib/desc#read	mate/ Ime https://r-
lib.github.io/desc/ DT 0.26 2022-10-18 23:17:54 	249779 GPL-3 file	e LICENSE https://git	hub.com/rstudio/DT	
A short description as to why you would A clear and concise description.	d like these packages to be whi	itelisted.		
Additional information Add any other information you deem use	efull.			
Attach files by dragging & dropping, selecti	ng or pasting them.			Mł
Styling with Markdown is supported				Submit new issue

Figure 2: Request template

Issue:	Reque	est											
Suggest pa	ackages for	the whitelist. If thi	s doesn't look	right, choose a differe	nt type.								
•	My Req	uest											
	Write	Preview				ΗE	3 I) ∂ ∷	1=	₽≣ @	¢	€
	**Table of	f packages to reque	est in markdow I downloa	n format (package, ve ds last monthllicense	rsion, date, dowi lurl	nloads_	last_m	onth, lice	ense, url).	**			
	:- GGally 2	2.1.2 2021-06-21	 03:40:10 gally	86657 GPL (>= 2	: 0) https://go	gobi.git	hub.io/	'ggally/,					I
	lintr 3.0	0.2 2022-10-19 08 3.1 2022-09-08 2	3:52:37 20:30:02	61729 MIT + file Ll 39420 GPL-3	CENSE https://g https://pak.r	ithub.co -lib.org,	om/r-lil /	o/lintr, h	ttps://lint	tr.r-lib. 	org	I	
	**A short I'm the de	description as to w ev, I can do whatev	vhy you would er I want.	like these packages to	be whitelisted.*	*							I
	Additio Maybe I s	nal information hould just add Dep	pendencyRevie	wer to the whitelist 🔮)								•
	Attach file	es by dragging & dro	opping, selecting	g or pasting them.									M#
	Styling w	ith Markdown is supp	orted							2	Submit n	ew issu	ue

Figure 3: Request filled out

(and the						
kages for the	e whitelist. II	this doesn't look	right, choose a different typ	e.		
My Reques	st					
Write	Preview					
Table of pack	kages to rec	quest in markdow	n format (package, version,	date, downloa	ds_last_month, license, url).	
package	version	date	downloads_last_month	license	url	
GGally	2.1.2	2021-06-21 03:40:10	86657	GPL (>= 2.0)	https://ggobi.github.io/ggally/, https://github.com/ggobi/ggally	
lintr	3.0.2	2022-10-19 08:52:37	61729	MIT + file LICENSE	https://github.com/r-lib/lintr, https:// lib.org	/lintr.r-
pak	0.3.1	2022-09-08 20:30:02	39420	GPL-3	https://pak.r-lib.org/	
A short desc 'm the dev, l Additional ir	ription as to can do wha nformation) why you would atever I want.	like these packages to be w	hitelisted.		
	A short desc write write able of pack package GGally lintr pak A short desc writhe dev, Additional i	kages for the whitelist. If My Request Write Preview Table of packages to require Table of package version GGally 2.1.2 lintr 3.0.2 pak 0.3.1 A short description as two Additional information	A ages for the whitelist. If this doesn't look My Request My Request Write Preview Gable of packages to request in markdow package version date GGally 2.1.2 2021-06-21 03:40:10 03:40:10 lintr 3.0.2 2022-10-19 08:52:37 03:31 2022-09-08 20:30:02 pak 0.3.1 2022-09-08 20:30:02 03:02 03:02 A short description as to why you would 'm the dev, I can do whatever I want. Weight of the state of	A cages for the whitelist. If this doesn't look right, choose a different typ My Request Write Preview Table of packages to request in markdown format (package, version, package version date downloads_last_month GGally 2.1.2 2021-06-21 03:40:10 86657 lintr 3.0.2 2022-10-19 08:52:37 61729 pak 0.3.1 2022-09-08 20:30:02 39420 A short description as to why you would like these packages to be w 'm the dev, I can do whatever I want. Additional information	ages for the whitelist. If this doesn't look right, choose a different type. My Request My Request Write Preview Fable of packages to request in markdown format (package, version, date, downloads package) Version gackage version date downloads_last_month license GGally 2.1.2 2021-06-21 86657 GPL (>= 2.0) lintr 3.0.2 2022-10-19 61729 MIT + file LICENSE pak 0.3.1 2022-09-08 39420 GPL-3 A short description as to why you would like these packages to be whitelisted. Image: Market and the dev, I can do whatever I want. Market and the dev i want.	Additional information

Figure 4: Request preview

We can then preview our request issue:

If everything looks good, we can submit the issue.

summariseFunctionUse

What does it do? summariseFunctionUse goes through all specified R-files and attempts to list all the functions used in those files. It will also report in what file the function was found, at what line number the function call was found, and from which package the function comes from.

What does it need? summariseFunctionUse has several optional arguments:

- 1. r_files default (list.files(here::here("R"))): If in_package = TRUE expects a character vector of at least length 1 of file names in the /R/ folder. If in_package == FALSE expects full paths to the R-files.
- 2. verbose default (FALSE): If verbose = TRUE will print messages in the console on which file the function is currently working. Useful when reviewing large R-files. If verbose = FALSE will not print said messages.
- 3. in_package default (TRUE): If in_package = TRUE will expect that the function is run inside a package-project. If in_package = FALSE will expect that the function is run outside a package-project and will expect full file paths to the files reviewed.

By default summariseFunctionUse will expect that it is ran inside a package-project and will look at the /R/ folder inside the project.

What does it return? summariseFunctionUse returns a class of data.frame with the following columns: *r_file*, *line*, *pkg*, *fun*.

Assumes the function is run inside a package-project.
datatable(
 summariseFunctionUse(list.files(here::here("R"), full.names = TRUE)
))

Show	10 • entries	S	earch:						
	r_file	line 🗍	pkg	*	fun	×			
1	checkDependencies.R	27	base		function				
2	checkDependencies.R	29	dplyr		filter				
3	checkDependencies.R	29	base		is.na				
4	checkDependencies.R	30	dplyr		rename				
5	checkDependencies.R	31	dplyr		left_join				
6	checkDependencies.R	33	base		с				
7	checkDependencies.R	35	dplyr		filter				
8	checkDependencies.R	48	base		function				
9	checkDependencies.R	51	dplyr		filter				
10	checkDependencies.R	52	dplyr		anti_joir	1			
Showing	g 1 to 10 of 274 entries Previous 1 2	3 4	5		28	Next			
<pre>if (into # Any foundl r_f: ver) datat; }</pre>	<pre>eractive()) { other R-file, with verbose messages Funs <- summariseFunctionUse(iles = "/inst/testScript.R", bose = TRUE able(foundFuns)</pre>								

The found functions can then be plotted out for each package. For the sake of this demonstration, only a few packages will be plotted.

```
if (interactive()) {
  funCounts <- foundFuns %>%
    group_by(fun, pkg, name = "n") %>%
    tally() %>%
    dplyr::filter(pkg %in% c("checkmate", "DBI", "dplyr"))

ggplot(
    data = funCounts,
    mapping = aes(x = fun, y = n, fill = pkg)
) +
```

```
geom_col() +
facet_wrap(
    vars(pkg),
    scales = "free_x",
    ncol = 1
    ) +
    theme_bw() +
    theme(
    legend.position = "none",
    axis.text.x = (element_text(angle = 45, hjust = 1, vjust = 1))
    )
}
```

getGraphData

What does it do? getGraphData creates an igraph graph object of all the dependencies that the root package depends on. This includes direct and transitive dependencies.

What does it need? getGraphData has three optional parameters:

- 1. **path** default (here::here()): Path to the package to get the graph data of. By default assumes that the function is ran inside a package-project.
- 2. excluded_packages default (c("")): A character vector of packages to be excluded. By default is empty.
- 3. package_types default (c("imports", "depends")): Package dependency types to be included. By default imports and depends are included. Available types are: 1) "imports", 2) "depends", 3) "suggests",
 4) "enhances", 5) "linkingto"

Without any of these specified, the getGraphData function assumes that it is ran inside an package-project.

What does it return? getGraphData returns a class of igraph.

graphData <- getGraphData()</pre>

Because the amount of dependencies in the graph quickly get out of hand, it is suggested that you would either filter the igraph object after the fact, or only look at one kind of package type. In the following example we'll look at *"Imports"* only to keeps things simple.

It could then be plotted like so:

#> Updated metadata database: 4.93 MB in 12 files. Updated metadata database: 4.93 MB in 12 files.
#>

#> Updating metadata database Updating metadata database Updating metadata database ... done Updatin

Get graphData with only imports
graphData <- getGraphData()</pre>

```
if (!is.null(graphData)) {
  # Calculate colour of nodes based on distances from root package
cols <- factor(as.character(apply(</pre>
 X = distances(graphData, V(graphData)[1]),
 MARGIN = 2,
 FUN = max
)))
# Plot graph
ggnet2(
 net = graphData,
 arrow.size = 1,
 \operatorname{arrow.gap} = 0.025,
 label = TRUE,
 palette = "Set2",
 color.legend = "distance",
  color = cols,
  legend.position = "bottom",
  edge.alpha = 0.25,
  node.size = 2.5,
  label.size = 1,
  legend.size = 2
)
}
```



mon 🕒 0 🛑 1 🛑 2 🎒 3

runShiny

What does it do? runShiny runs a local shiny app that houses all the before mentioned functionality in one environment. runSHiny assumes that it is being ran inside a package-project.

What does it need? runShiny Takes no arguments

What does it return? runShiny returns a class of shiny.appobj.

runShiny()

The shiny application has three main tabs: 1) Package review, 2) Dependency Graph, and 3) Path to dependency.

Package review

On the package review tab there are three main panels.

- 1. **Settings**: The settings have two parts on this panel: A file picker, and tick boxes to packages. Currently all the files are in the summariseFunctionUse table.
- 2. summariseFunctionUse table and plot: The summariseFunctionUse table for the specified files, or all files if *ALL* is picked in the file picker in the settings.
- 3. Script preview: A preview of the contents of the selected file. If *ALL* is chosen, a dummy script will appear, or the last viewed contents will stay.

ALL	Exclude Packages	nu s					
Function	odov Plot				0.0000		
Show 10	• entries				Search:	2 3 1f(x == 3) {	
	r hie		line 0	pkg	2 tun	$ \mathbf{x} \in \{\mathbf{x}_i\}$	
1	checkDependencies R		27	base	Tunction		
2	checkDependencies R		29	dplyr	filter		
8	checkDependencies R		29	base.	ts na		
4	checkDependencies.R		30	dplyr	rename		
5	checkDependencies R		31	dolar	lett iom		
6	cherkDecendencies R		33	base	¢		
7	checkDenandencies D		ः अ	dolar	flice		
	check Dependencies R		20	bara.	But the		
	CONTRACTOR CONTRACTOR OF CONTRACTOR OFICACTOR OFICON OFICO		47	LNOSE	MACTOR		
9	checkDependencies.R		30	dply	liter	-	
	and the second se		-1	a standard and	COMPANY AND A DESCRIPTION OF A DESCRIPTI		

Figure 5: Function review

Notice how the **Settings**, **summariseFunctionUse table and plot**, and **Script preview** dynamically change when the darwinLint.R file is selected.

When swapping from the **Function review** to the **Plot** tab a bar graph is shown for each package used in the file. The bars represent the amount of function calls in that file per package.

Lets say base functions are not interesting for your use case, you can then tick the *base* tick box in the Exclude Packages in the settings.

darwinLint.R •	Exclusion of the second	e 🔘 Intr 🗍 here I 🗍 unknown 📋 dpt	r.	
Function review Plot				
how 10 + entries			Search:	1 = br. darwini.txtPackage 2 = #
	line :	pkg	fun	 a Darwin Lintz object, using default Lintz object with cumellage a sectore List of list adjects.
t	11	base	function	6 # SURDOT Lintr 7 #: Becont 8 #
2	12	base	tryCation	9 # generations 19 # devide_intPockage[] 19 devide_intPockage[]
3	13	Britz	Int_package	13 tryCatch() 13 listrillat_cackage()
4	14	here	here	14 path = here: (here:) 15 listers = listr: listers_with_defaults(16 listr: isofect_mame_Lister(styles = "camel(ase")))
5	15	lintz	Inters_with_defaults	17), troor = function(e) (18 cil.icil_stert_comper(e) 19 troor
6	16	000	object_name_inter	28 "Error was coupt during the listing of your package. The package all wight be to large to list all together. use: narvdnlistFile(filewame)")
7	17	base	function	
8	18	CR.	cli_alert_danger	28 N # involutionEntropy 7 #
9	19	base	stop	38 2 thet a gives file.
10	21	unknown	darwini_intFile	

Figure 6: Package review



Figure 7: Function review plot



Figure 8: Package review

file	Exclus	le Packages e 🔲 Intr. 🖾 here			
Garwiichiere.	0.0	🗉 илклоwл 💷 арђ	r		
Function review Flot					
ow 10 • entries			Search:	2 N: dersicalmethachage 2 N	
	line -	pkg	: fun	 4 Dervice lists signst, using default lists signst with samellane 4 dervice list of list signsts. 	
	13	hete:	Int_package	7 · Gesport	
	14	here.	here	 8 * Second and Second (1) 10 definition of the second (1) 11 definition of the second (1) 	
	15	tiote :	Inters_with_defaults	11 tryCatch() 13 lintr:Lint_package	
	16	linta:	object_name_Inter	14 path + hare: hare (), 13 linter = linter:linter.uith_sefaults) 10 linter:sefuet mam linter: https://pumlias/))	
	18	CR.	cli_alert_danger	<pre>17), error = function(e) (18 clinical_stert_sterger(e) </pre>	
	21	unknown	darwinLimFile	20 Throw was caught during the listing of your package. The package might be to large to list all together. Use: danishistfile(fileNews)")	
	42	lintr	lint.		
	44	lintr	Inters_with_defaults	23 26 s ⁻ saruitistriis	
	45	lintr	object_name_linter	22 F list a given file.	
0	72	unknown	IntFunction		

Figure 9: Package review

base packages are now excluded from both the summariseFunctionUse table and plot.

Dependency Graph

The Dependency Graph tab displays a graph, like plotted earlier, using the graphData function. On the right-hand-side different kinds of dependencies are able to be chosen to be included in the graph.



Figure 10: Package review

Path to dependency

The path to dependency tab displays how the root package depends on any recursive dependency.

On the right-hand-side a dependency found somewhere included in the root package can be chosen. A cutoff can be defined to limit the distance from the root package to the chosen dependency.

${\bf darwinLintFile}$

What does it do? darwinLintFile is an extension of the default Lintr object, but instead of *snake_case*, it uses *camelCase*. As the name suggest it will run the lintr on a specified file.

What does it need? darwinLintFile takes one parameter: 1. fileName: Path to an R-file.

What does it return? It returns a class of lints.

However the output of a lintr function can be cast to a data.frame.



Figure 11: Package review

```
if (interactive()) {
    lintOut <- data.frame(
        darwinLintFile(
            fileName = "../inst/testScript.R"
        )
    )
}</pre>
```

Which can then be manipulated to get a summary of lint messages.

```
if (interactive()) {
    lintOut %>%
    group_by(type, message) %>%
    tally(sort = TRUE) %>%
    datatable()
}
```

darwinLintPackage

What does it do? darwinLintPackage is an extension of the default Lintr object, but instead of *snake_case*, it uses *camelCase*. But unlike darwinLintFile, will run the lintr on the entire package. Therefore it will assume that the function is ran inside a package-project.

What does it need? darwinLintPackage Does not take any arguments.

What does it return? It returns a class of lints.

darwinLintScore

What does it do? darwinLintScore calculates a percentage per type of lint-message from the lintr.

The percentage is calculated as:

$$darwinLintScore_{type} = \frac{n_{messages}}{n_{lines}} \times 100$$

What does it need? darwinLintScore takes one predefined argument: 1. lintFunction: A lint function extended from lintr::lint_package or lintr::lint 2. ...: Any other arguments that the lint function might need

What does it return? Returns a class of data.frame with two columns: 1) type, and 2) pct.

It will also print out colour coded messages with the percentages per message type.

```
if (interactive()) {
   darwinLintScore(darwinLintPackage)
}
```

i style: 5.9% of lines of code have linting messages i warning: 0.95% of lines of code have linting messages

type	pct
style	5.9
warning	0.95