

Quantifying and lightening the package dependency burden

Robert Castelo
Universitat Pompeu Fabra
Barcelona

robert.castelo@upf.edu - @robertclab

BioC Developer's Forum
February 20th, 2020

Motivation for this discussion

<https://stat.ethz.ch/pipermail/bioc-devel/2020-February/016146.html>

[Bioc-devel] how to trace 'Matrix' as package dependency for 'GenomicScores'

Robert Castelo robert@c@@te|o @end|ng |rom up|@edu

Thu Feb 6 12:36:08 CET 2020

- Previous message (by thread): [\[Bioc-devel\] Account Activation Issue](#)
- Next message (by thread): [\[Bioc-devel\] how to trace 'Matrix' as package dependency for 'GenomicScores'](#)
- **Messages sorted by:** [\[date \]](#) [\[thread \]](#) [\[subject \]](#) [\[author \]](#)

hi,

when i load the package 'GenomicScores' in a clean session i see thorough the 'sessionInfo()' that the package 'Matrix' is listed under "loaded via a namespace (and not attached)".

i'd like to know what is the dependency that 'GenomicsScores' has that ends up requiring the package 'Matrix'.

GenomicScores

platforms	all	rank	255 / 1828	posts	0	in Bioc	3 years
build	warnings	updated	< 1 month	dependencies	116		

R Under development (unstable) (2020-01-29 r77745)

Platform: x86_64-pc-linux-gnu (64-bit)

Running under: CentOS Linux 7 (Core)

Matrix products: default

BLAS: /opt/R/R-devel/lib64/R/lib/libRblas.so

LAPACK: /opt/R/R-devel/lib64/R/lib/libRlapack.so

locale:

```
[1] LC_CTYPE=en_US.UTF8      LC_NUMERIC=C
[3] LC_TIME=en_US.UTF8       LC_COLLATE=en_US.UTF8
[5] LC_MONETARY=en_US.UTF8   LC_MESSAGES=en_US.UTF8
[7] LC_PAPER=en_US.UTF8      LC_NAME=C
[9] LC_ADDRESS=C             LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF8 LC_IDENTIFICATION=C
```

attached base packages:

```
[1] parallel stats4      stats      graphics grDevices utils      datasets
[8] methods  base
```

other attached packages:

```
[1] GenomicScores_1.11.4 GenomicRanges_1.39.2 GenomeInfoDb_1.23.10
[4] IRanges_2.21.3       S4Vectors_0.25.12    BiocGenerics_0.33.0
[7] colorout_1.2-2
```

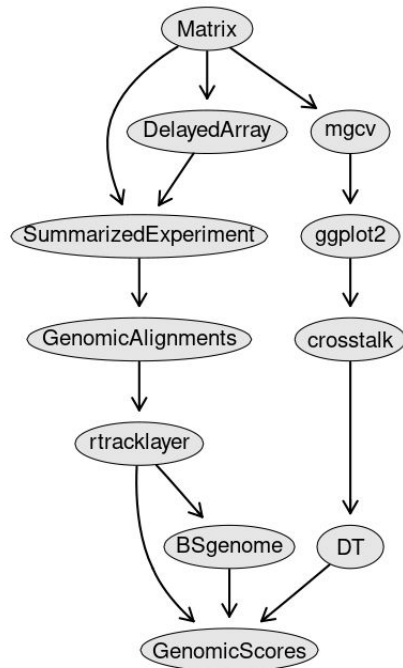
loaded via a namespace (and not attached):

```
[1] Rcpp_1.0.3          lattice_0.20-38
[3] shinycustomloader_0.9.0 Rsamtools_2.3.3
[5] Biostrings_2.55.4    assertthat_0.2.1
[7] digest_0.6.23        mime_0.9
[9] BiocFileCache_1.11.4 R6_2.4.1
[11] RSQLite_2.2.0        httr_1.4.1
[13] pillar_1.4.3         zlibbioc_1.33.1
[15] rlang_0.4.4          curl_4.3
[17] data.table_1.12.8    blob_1.2.1
[19] DT_0.12              Matrix_1.2-18
[21] shinythemes_1.1.2    shinyjs_1.1
[23] BiocParallel_1.21.2  AnnotationHub_2.19.7
[25] htmlwidgets_1.5.1    RCurl_1.98-1.1
[27] bit_1.1-15.1         shiny_1.4.0
[29] DelayedArray_0.13.3  compiler_4.0.0
[31] httpuv_1.5.2         rtracklayer_1.47.0
[33] pkgconfig_2.0.3      htmltools_0.4.0
[35] tidyselect_1.0.0     SummarizedExperiment_1.17.1
[37] tibble_2.1.3         GenomeInfoDbData_1.2.2
[39] interactiveDisplayBase_1.25.0 matrixStats_0.55.0
[41] XML_3.99-0.3         crayon_1.3.4
[43] dplyr_0.8.4          dbplyr_1.4.2
[45] later_1.0.0          GenomicAlignments_1.23.1
[47] bitops_1.0-6         rappdirs_0.3.1
[49] grid_4.0.0           xtable_1.8-4
[51] DBI_1.1.0            magrittr_1.5
[53] XVector_0.27.0       promises_1.1.0
[55] vctrs_0.2.2          tools_4.0.0
[57] bit64_0.9-7         BSgenome_1.55.3
[59] Biobase_2.47.2       glue_1.3.1
[61] purrr_0.3.3         BiocVersion_3.11.1
[63] fastmap_1.0.1       yaml_2.2.1
[65] AnnotationDbi_1.49.1 BiocManager_1.30.10
[67] memoise_1.1.0
```

First quick and dirty solution

```
repos <- BiocManager::repositories()[c("BioCsoft", "CRAN")]
db <- available.packages(repos=repos)
deps <- tools::package_dependencies("GenomicScores", db, recursive=TRUE)[[1]]
length(deps)
[1] 116
deps <- tools::package_dependencies(c("GenomicScores", deps), db)
g <- graph::graphNEL(nodes=names(deps), edgeL=deps, edgemode="directed")
igraph::all_simple_paths(igraph::igraph.from.graphNEL(g),
                        from="GenomicScores", to="Matrix", mode="out")

[[1]]
[1] GenomicScores      BSgenome           rtracklayer       GenomicAlignments
[5] SummarizedExperiment DelayedArray       Matrix
[[2]]
[1] GenomicScores      BSgenome           rtracklayer       GenomicAlignments
[5] SummarizedExperiment Matrix
[[3]]
[1] GenomicScores DT          crosstalk         ggplot2           mgcv
[6] Matrix
[[4]]
[1] GenomicScores      rtracklayer       GenomicAlignments
[4] SummarizedExperiment DelayedArray       Matrix
[[5]]
[1] GenomicScores      rtracklayer       GenomicAlignments
[4] SummarizedExperiment Matrix
```



Discussion followed ..

[Bioc-devel] how to trace 'Matrix' as package dependency for 'GenomicScores'

Martin Morgan mtmorg@n@b|oc @end|ng |rom gm@||@com

Sat Feb 8 18:01:53 CET 2020

- Previous message (by thread): [\[Bioc-devel\] how to trace 'Matrix' as package dependency for 'GenomicScores'](#)
- Next message (by thread): [\[Bioc-devel\] how to trace 'Matrix' as package dependency for 'GenomicScores'](#)
- **Messages sorted by:** [\[date \]](#) [\[thread \]](#) [\[subject \]](#) [\[author \]](#)

I find it quite interesting to identify formal strategies for removing dependencies, but also a little outside my domain of expertise. This code

[...]

shows me, via `n.remove`, that I can remove the dependency on AnnotationHub by removing the dependency on just one package (AnnotationHub!), but to remove BiocFileCache I'd also have to remove another package (AnnotationHub, I'd guess). So this provides some measure of the ease with which a package can be removed.

I'd like a 'benefit' column, too -- if I were to remove AnnotationHub, how many additional packages would I also be able to remove, because they are present only to satisfy the dependency on AnnotationHub? More generally, perhaps there is a dependency of AnnotationHub that is only used by AnnotationHub and BSgenome. So removing AnnotationHub as a dependency would make it easier to remove BSgenome, etc. I guess this is a graph optimization problem.

Probably also worth mentioning the itdepends package (<https://github.com/r-lib/itdepends>), which I think tries primarily to determine the relationship between package dependencies and lines of code, which seems like complementary information.

Martin

Discussion followed ..

[Bioc-devel] how to trace 'Matrix' as package dependency for 'GenomicScores'

Vincent Carey [@tyjc@end|ng|rom|ch|nn|ng|h@rv|rd|edu](mailto:tyjc@end|ng|rom|ch|nn|ng|h@rv|rd|edu)
Sun Feb 9 13:31:24 CET 2020

- Previous message (by thread): [\[Bioc-devel\] how to trace 'Matrix' as package dependency for 'GenomicScores'](#)
- Next message (by thread): [\[Bioc-devel\] how to trace 'Matrix' as package dependency for 'GenomicScores'](#)
- Messages sorted by: [\[date \]](#) [\[thread \]](#) [\[subject \]](#) [\[author \]](#)

On Sat, Feb 8, 2020 at 12:02 PM Martin Morgan <tmorgan.bioc.using@gmail.com> wrote:

> I find it quite interesting to identify formal strategies for removing
> dependencies, but also a little outside my domain of expertise. This code
>

It would be nice to collect the ideas in this thread into some recommendations. The themes I am thinking of are "how developers can make their packages robust to loss of external packages" and "how can the Bioc ecosystem best deal with departures of packages from itself and from CRAN?" A good and well-adopted solution to the first one makes the second one moot.

Two CRAN-related events I know of that required some effort are (temporary) loss of ashR and (recently) archiving of Seurat.

- Developer: How to quantify and lighten the package dependency burden.
- BioC core team: How to identify packages that compromise and are compromised by the package dependency burden.

[Bioc-devel] how to trace 'Matrix' as package dependency for 'GenomicScores'

Sean Davis [@e@nd@v|@end|ng|rom|gm@||@com](mailto:e@nd@v|@end|ng|rom|gm@||@com)
Sun Feb 9 17:01:28 CET 2020

- Previous message (by thread): [\[Bioc-devel\] how to trace 'Matrix' as package dependency for 'GenomicScores'](#)
- Next message (by thread): [\[Bioc-devel\] how to trace 'Matrix' as package dependency for 'GenomicScores'](#)
- Messages sorted by: [\[date \]](#) [\[thread \]](#) [\[subject \]](#) [\[author \]](#)

There are some good ideas here that would provide enhancement to [BiocPkgTools](#). I don't have the bandwidth to incorporate right now, but filing issues or a pull request with a skeleton would be helpful to keep track.

Sean

SOFTWARE TOOL ARTICLE



BiocPkgTools: Toolkit for mining the *Bioconductor* package ecosystem [version 1; peer review: 2 approved, 1 approved with reservations]

Shian Su¹, Vincent J. Carey ², Lori Shepherd³, Matthew Ritchie ¹, Martin T. Morgan³, Sean Davis⁴

BiocPkgTools

platforms rank 623 / 1823 posts 0 in Bioc 1.5 years
build updated before release dependencies 92

DOI: [10.18129/B9.bioc.BiocPkgTools](https://doi.org/10.18129/B9.bioc.BiocPkgTools)

Collection of simple tools for learning about Bioc Packages

Bioconductor version: Release (3.10)

Bioconductor has a rich ecosystem of metadata around packages, usage, and build status. This package is a simple collection of functions to access that metadata from R. The goal is to expose metadata for data mining and value-added functionality such as package searching, text mining, and analytics on packages.

Package itdepends - Jim Hester

<https://resources.rstudio.com/rstudio-conf-2019/it-depends-a-dialog-about-dependencies>

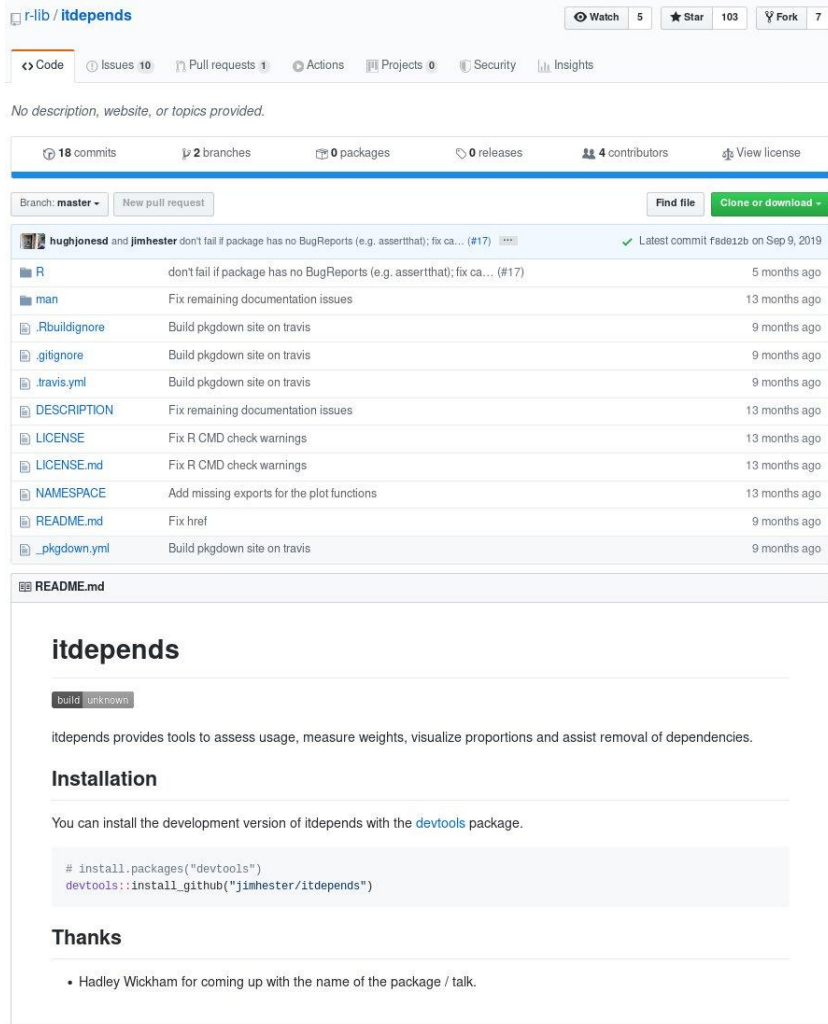
- Dependencies break.
- Not all dependencies are equal (upstream dependencies, system-wide requirements, etc.).
- Users of your package: developers (lightweight and stable), end users (feature-rich).
- Widely used packages are widely tested.
- You may want to remove dependencies from your package: quantification of the “dependency weight” is critical.
- The package itdepends is meant to be a toolbox to make this type of decisions.
- The contrast between the number of functions exported from a dependency and the number of its functions that you are actually importing in your package provide a sense of how much of that package dependency you are using.

Package itdepends

```
devtools::install_github("jimhester/itdepends")
```

```
# Determine usage of dependencies for a package  
itdepends::dep_usage_pkg()
```

```
# Calculate the package "weight"  
itdepends::dep_weight()
```



The screenshot shows the GitHub repository page for 'itdepends' by jimhester. The repository has 18 commits, 2 branches, 0 packages, 0 releases, 4 contributors, and a license. The commit history shows a recent commit by fede12b on Sep 9, 2019. The README.md file is displayed, showing the package name 'itdepends' and a build status of 'unknown'. The README text states: 'itdepends provides tools to assess usage, measure weights, visualize proportions and assist removal of dependencies.' The 'Installation' section says: 'You can install the development version of itdepends with the devtools package.' and provides the following code snippet:

```
# install.packages("devtools")  
devtools::install_github("jimhester/itdepends")
```

 The 'Thanks' section lists: '• Hadley Wickham for coming up with the name of the package / talk.'

Quantifying package dependency burden

```
du <- itdepends::dep_usage_pkg("GenomicScores")
```

```
du
```

```
# A tibble: 1,333 x 2
```

pkg	fun
<chr>	<chr>

1 base	standardGeneric
--------	-----------------

2 base	{
--------	---

3 base	<-
--------	----

4 utils	as.person
---------	-----------

5 base	if
--------	----

6 base	!
--------	---

7 base	missing
--------	---------

8 base	{
--------	---

9 base	<-
--------	----

10 GenomicScores	.mergeMaintainer
------------------	------------------

```
# ... with 1,323 more rows
```

Quantifying package dependency burden

```
table(du$pkg)
```

AnnotationHub	base	Biobase	BiocGenerics	BSgenome
6	1168	1	15	4
GenomeInfoDb	GenomicRanges	GenomicScores	IRanges	methods
18	19	29	6	1
S4Vectors	utils	XML		
57	7	2		

```
## FROM GenomicScores/NAMESPACE
## importFrom(Biostrings, DNA_BASES)
## importMethodsFrom(Biostrings, match)

expfun <- getNamespaceExports("Biostrings")
nonstd <- expfun[grepl("^\\.", expfun)]
length(setdiff(expfun, nonstd))
[1] 240
```

Quantifying package dependency burden

```
gistURL <- "https://gist.github.com/rcastelo/7429d05178ddb57a38bd42093c2ddfe2")  
devtools::source_gist(gistURL)
```

```
g <- pkgDepGraph("GenomicScores", db)  
gvtx <- graph::nodes(g)  
gdep <- pkgDepGraph("Biostrings", db)  
gdepvtx <- graph::nodes(gdep)  
depov <- length(intersect(gvtx, gdepvtx)) / length(union(gvtx, gdepvtx))  
depov  
[1] 0.075
```

Quantifying package dependency burden

Using functionality from `itdepends` we can quantify the dependency burden as follows.

```
gistURL <- "https://gist.github.com/rcastelo/7429d05178ddb57a38bd42093c2ddfe2")
devtools::source_gist(gistURL)

## build an acyclic digraph of package dependencies
pkgDepGraph()

## calculate package dependency metrics()
pkgDepMetrics()

## find what function calls in a package lead to a target dependency
funCalls2Dep()
```

Quantifying package dependency burden

```
g <- pkgDepGraph("GenomicScores", db)
gvtx <- graph::nodes(g)
gdep <- pkgDepGraph("Biostrings", db)
gdepvtx <- graph::nodes(gdep)
depov <- length(intersect(gvtx, gdepvtx)) / length(union(gvtx, gdepvtx))
depov
[1] 0.075
```

The dependency overlap, calculated as the ratio of the intersection over the union of dependencies, tells us that the dependencies from Biostrings represent a mere 0.075 fraction of the dependencies of GenomicScores.

Quantifying package dependency burden

```
pkgDepMetrics("GenomicScores", db)
```

	ImportedBy	Exported	Usage	DepOverlap
Biobase	1	128	0.781250	0.0250
BSgenome	1	93	1.075269	0.3625
XML	2	176	1.136364	0.0125
IRanges	4	254	1.574803	0.0375
BiocGenerics	5	139	3.597122	0.0125
GenomicRanges	4	104	3.846154	0.1125
S4Vectors	11	262	4.198473	0.0250
GenomeInfoDb	5	53	9.433962	0.0750
AnnotationHub	4	33	12.121212	0.6875
Biostrings	NA	240	NA	0.0750



GenomicScores uses about 1% of the functionality exposed by BSgenome, while BSgenome overlaps a fraction of 0.36 of the dependencies of GenomicScores. Therefore, we would say that BSgenome is a candidate package to lighten the dependency burden of GenomicScores

Quantifying package dependency burden

What are the function calls in GenomicScores to BSgenome?

```
funCalls2Dep("GenomicScores", "BSgenome", db)
# A tibble: 1 x 3
# Groups:   pkg [1]
  pkg      fun              n
  <chr>    <chr>          <int>
1 BSgenome referenceGenome      4
```

Further arguments to funCalls2Dep() allow to see, using the package itdepends, what are the actual lines of code doing those function calls.