There’s been alot of talk about “dependencies” in the R universe of late. This is not really a post about that but more of a “really, don’t do this” if you decide you want to poke the dependency bear by trying to build a deeply flawed model off of CRAN package metadata.

CRAN packages undergo checks. Here’s [one for akima](https://cran.rstudio.com/web/checks/check_results_akima.html) (I :heart: me some gridded interpolation functions, plus this package is not in any hot-button R tribe right now):

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Flavor** | **Version** | **Tinstall** | **Tcheck** | **Ttotal** | **Status** | **Flags** |
| r-devel-linux-x86\_64-debian-clang | 0.6-2 | 9.83 | 32.67 | 42.50 | [OK](https://www.r-project.org/nosvn/R.check/r-devel-linux-x86_64-debian-clang/akima-00check.html) |  |
| r-devel-linux-x86\_64-debian-gcc | 0.6-2 | 8.53 | 26.56 | 35.09 | [OK](https://www.r-project.org/nosvn/R.check/r-devel-linux-x86_64-debian-gcc/akima-00check.html) |  |
| r-devel-linux-x86\_64-fedora-clang | 0.6-2 |  |  | 53.33 | [NOTE](https://www.r-project.org/nosvn/R.check/r-devel-linux-x86_64-fedora-clang/akima-00check.html) |  |
| r-devel-linux-x86\_64-fedora-gcc | 0.6-2 |  |  | 51.03 | [NOTE](https://www.r-project.org/nosvn/R.check/r-devel-linux-x86_64-fedora-gcc/akima-00check.html) |  |
| r-devel-windows-ix86+x86\_64 | 0.6-2 | 53.00 | 76.00 | 129.00 | [OK](https://www.r-project.org/nosvn/R.check/r-devel-windows-ix86+x86_64/akima-00check.html) |  |
| r-patched-linux-x86\_64 | 0.6-2 | 9.26 | 28.32 | 37.58 | [OK](https://www.r-project.org/nosvn/R.check/r-patched-linux-x86_64/akima-00check.html) |  |
| r-patched-solaris-x86 | 0.6-2 |  |  | 66.20 | [OK](https://www.r-project.org/nosvn/R.check/r-patched-solaris-x86/akima-00check.html) |  |
| r-release-linux-x86\_64 | 0.6-2 | 8.59 | 28.25 | 36.84 | [OK](https://www.r-project.org/nosvn/R.check/r-release-linux-x86_64/akima-00check.html) |  |
| r-release-windows-ix86+x86\_64 | 0.6-2 | 39.00 | 69.00 | 108.00 | [OK](https://www.r-project.org/nosvn/R.check/r-release-windows-ix86+x86_64/akima-00check.html) |  |
| r-release-osx-x86\_64 | 0.6-2 |  |  |  | [OK](https://www.r-project.org/nosvn/R.check/r-release-osx-x86_64/akima-00check.html) |  |
| r-oldrel-windows-ix86+x86\_64 | 0.6-2 | 28.00 | 71.00 | 99.00 | [OK](https://www.r-project.org/nosvn/R.check/r-oldrel-windows-ix86+x86_64/akima-00check.html) |  |
| r-oldrel-osx-x86\_64 | 0.6-2 |  |  |  | [OK](https://www.r-project.org/nosvn/R.check/r-oldrel-osx-x86_64/akima-00check.html) |  |

Check Details

Version: 0.6-2

Check: compiled code

Result: NOTE

File ‘akima/libs/akima.so’:

Found no call to: ‘R\_useDynamicSymbols’

It is good practice to register native routines and to disable symbol search.

See ‘Writing portable packages’ in the ‘Writing R Extensions’ manual.

The Status field can be “OK”, “NOTE”, “WARN[ING]”, “ERROR”, or “FAIL”.

You’ll also note that there are checks for a future, even cooler R (“devel”), spiffy R (“release”/”patched”) and :yawn: R (“oldrel”). Remember those, they are important.

Now, let’s say you wanted to perform an honest appraisal of whether packages with more dependencies are more likely to have one or more “bad” CRAN check conditions. You’d likely lump “NOTE” with “OK” and not mark that particular check against the package. That leaves “WARN[ING]” (the reason for the [ING] is that different check RDS files include/forego the [ING]…yay consistency?) “ERROR” and “FAIL”. Obviously we can just use those without further concern, right?

**WARN[ING] Will Robinson!**

You can get a copy of the check details at <https://cran.r-project.org/web/checks/check_details.rds>. I happen to have a local copy (used “Mar 18 02:47” version) now that my CRAN mirror is re-humming along nicely again. Let’s make sure it’s being read in OK:

library(tidyverse)

det <- as\_tibble(readRDS("check\_details.rds")) # it's got tons of classes & I like readable data frame prints

nrow(distinct(det, Package))

## [1] 13094

OK, that number tracks with the count as of the last rsync. So, what do these WARN[ING]s look like?

filter(det, Status == "WARNING") %>%

select(Output)

## # A tibble: 2,299 x 1

## Output

##

## 1 "Found the following significant warnings:\n Warning: unable to r…

## 2 "Found the following significant warnings:\n Warning: unable to r…

## 3 "Found the following significant warnings:\n Warning: unable to r…

## 4 "Warning in parse(file = files, n = -1L) :\n invalid input found …

## 5 "Warning in parse(file = files, n = -1L) :\n invalid input found …

## 6 "Found the following significant warnings:\n Warning: unable to r…

## 7 "Found the following significant warnings:\n Warning: unable to r…

## 8 "Found the following significant warnings:\n Warning: unable to r…

## 9 "Found the following significant warnings:\n track\_methods.cpp:22…

## 10 "Found the following significant warnings:\n Warning: unable to r…

## # … with 2,289 more rows

EEK! Well, actually not really. The checks are automated and we can use some substring machinations to try to get better groups:

filter(det, Status == "WARNING") %>%

mutate(bits = substring(trimws(Output), 1, 30)) %>%

count(bits, sort=TRUE) %>%

mutate(pct = n/sum(n))

## # A tibble: 29 x 3

## bits n pct

##

## 1 Found the following significan 1316 0.572

## 2 Error in re-building vignettes 702 0.305

## 3 Error(s) in re-building vignet 90 0.0391

## 4 Output from running autoreconf 56 0.0244

## 5 Warning in parse(file = files, 37 0.0161

## 6 Missing link or links in docum 15 0.00652

## 7 "network.dyadcount:\n function(" 10 0.00435

## 8 Found the following executable 9 0.00391

## 9 dyld: Library not loaded: /Bui 8 0.00348

## 10 Errors in running code in vign 8 0.00348

## # … with 19 more rows

OK, so *some* “eeking” is warranted for those “significant” ones but thirty percent of these findings are about vignettes. Sure, vignettes are important and ideally they build fine but there are *tons* of reasons they don’t on CRAN’s ever-changing infrastructure. I say they need to be excluded. Drop a note in the comments with a different opinion since *this is an analyst’s opinion*. But I happen to know CRAN *really* well and would seriously suggest that in the context of the *question* regarding high dependency package efficacy that this should be ignored unless further investigated individually.

So, where are these “significant” WARN[ING]s?

filter(det, Status == "WARNING") %>%

filter(grepl("significant", Output, ignore.case = TRUE)) %>%

mutate(flavor\_flav = case\_when(

grepl("devel", Flavor) ~ "devel",

grepl("oldrel", Flavor) ~ "oldrel",

TRUE ~ "current"

)) %>%

count(flavor\_flav, sort=TRUE) %>%

mutate(pct = n/sum(n))

## # A tibble: 3 x 3

## flavor\_flav n pct

##

## 1 devel 904 0.686

## 2 current 280 0.213

## 3 oldrel 133 0.101

I posit that if the goal really is to create a model to help decide whether you should take on the risk of packages with multiple++ dependencies you cannot include “devel”. Nobody sane runs “devel” in production and that’s the real goal: a safe production environment. So you literally have to throw out 68% of these, too (some folks are stuck on :yawn: “oldrel” R in orgs with draconian IT practices or fragile workflow systems). We’re not at “0” yet so what are some of these issues?

filter(det, Status == "WARNING") %>%

filter(grepl("significant", Output, ignore.case = TRUE)) %>%

mutate(flavor\_flav = case\_when(

grepl("devel", Flavor) ~ "devel",

grepl("oldrel", Flavor) ~ "oldrel",

TRUE ~ "current"

)) %>%

filter(flavor\_flav != "devel") %>%

mutate(Output = gsub("Found the following significant warnings:\n ", "", trimws(Output))) %>%

mutate(bits = substring(trimws(Output), 1, 50)) %>%

count(bits, sort=TRUE) %>%

mutate(pct = n/sum(n))

## # A tibble: 110 x 3

## bits n pct

##

## 1 "Warning: S3 methods '[.fun\_list', '[.grouped\_df', " 158 0.383

## 2 Warning: 'rgl\_init' failed, running with rgl.useNU 60 0.145

## 3 "Found the following significant warnings:\n\n Warn… 9 0.0218

## 4 Warning: package ‘dplyr’ was built under R version 9 0.0218

## 5 "bgc\_hmm.c:241:31: warning: ‘, ’ directive writing " 4 0.00969

## 6 driver.c:381:26: warning: cast from pointer to int 4 0.00969

## 7 hash.c:144:5: warning: ‘strncpy’ specified bound 2 4 0.00969

## 8 RngStream.c:347:4: warning: ‘strncpy’ specified bo 4 0.00969

## 9 Warning: ‘\_\_var\_1\_mmb.offset’ is used uninitialize 4 0.00969

## 10 /home/hornik/tmp/R.check/r-patched-gcc/Work/build/ 3 0.00726

## # … with 100 more rows

Fun fact: a re-run of this with a 2019-03-19 RDS pulled from CRAN shows 79 vs 158 (and those 79 packages were’t magically re-submitted). This is usually a CRAN check “hiccup” on Windows:

filter(det, Status == "WARNING") %>%

filter(grepl("significant", Output, ignore.case = TRUE)) %>%

mutate(flavor\_flav = case\_when(

grepl("devel", Flavor) ~ "devel",

grepl("oldrel", Flavor) ~ "oldrel",

TRUE ~ "current"

)) %>%

filter(flavor\_flav != "devel") %>%

mutate(Output = gsub("Found the following significant warnings:\n ", "", trimws(Output))) %>%

mutate(bits = substring(trimws(Output), 1, 50)) %>%

filter(grepl("S3 methods", bits)) %>%

count(bits, Flavor, sort=TRUE) %>%

mutate(pct = n/sum(n))

## # A tibble: 6 x 4

## bits Flavor n pct

##

## 1 "Warning: S3 methods '[.fun\_list'… r-oldrel-windows-i… 79 0.485

## 2 "Warning: S3 methods '[.fun\_list'… r-release-windows-… 79 0.485

## 3 Warning: S3 methods 'as\_mapper.ch… r-oldrel-windows-i… 2 0.0123

## 4 Warning: S3 methods '.DollarNames… r-oldrel-windows-i… 1 0.00613

## 5 Warning: S3 methods 'as.promise.F… r-release-windows-… 1 0.00613

## 6 Warning: S3 methods 'format.stati… r-oldrel-windows-i… 1 0.00613

Yep! So, we really have to ignore some portion of these but not many (remember, these are test counts, not package counts).

Perhaps we’ll have better luck ~~ginning up the analysis~~ focusing on “ERROR”!

**To ERROR Is Definitely Human When Assumptions Are Flawed**

Let’s see about these ERRORs:

filter(det, Status == "ERROR") %>%

mutate(flavor\_flav = case\_when(

grepl("devel", Flavor) ~ "devel",

grepl("oldrel", Flavor) ~ "oldrel",

TRUE ~ "current"

)) %>%

filter(flavor\_flav != "devel") %>%

mutate(bits = substring(trimws(Output), 1, 20)) %>%

count(bits, sort=TRUE) %>%

print(n=66)

## # A tibble: 66 x 2

## bits n

##

## 1 Installation failed. 437

## 2 "Running examples in " 424

## 3 Package required but 213

## 4 Running ‘testthat.R’ 150

## 5 Packages required bu 102

## 6 Running 'testthat.R' 58

## 7 Package required and 31

## 8 Running ‘test-all.R’ 17

## 9 Packages required an 12

## 10 Errors in running co 7

## 11 Running ‘spelling.R’ 6

## 12 Running ‘test-that.R 5

## 13 Running 'activate\_te 4

## 14 Running 'test-all.R' 4

## 15 Running ‘activate\_te 3

## 16 Running 'Bernstein-E 2

## 17 Running 'Class+Meth. 2

## 18 Running 'dist\_matrix 2

## 19 Running 'Frechet-tes 2

## 20 Running 'spelling.R' 2

## 21 Running 'test-as-dgC 2

## 22 Running 'valued\_fit. 2

## 23 Running ‘allier.R’ [ 2

## 24 Running ‘aunitizer.R 2

## 25 Running ‘autoprint.R 2

## 26 "Running ‘bdstest.R’ " 2

## 27 Running ‘build-tools 2

## 28 Running ‘exporting-m 2

## 29 Running ‘restfulr\_un 2

## 30 Running ‘run\_test.R’ 2

## 31 Running ‘test\_change 2

## 32 Running ‘test-as-dgC 2

## 33 Running ‘testGBHProc 2

## 34 Running ‘tests-nlgam 2

## 35 Running ‘testthat-pr 2

## 36 Running ‘TimeIn\_Data 2

## 37 Running '000.session 1

## 38 Running '001.setupEx 1

## 39 Running 'bug1.R' [1s 1

## 40 "Running 'failure.R' " 1

## 41 Running 'fold.R' [5s 1

## 42 Running 'Rgui.R' [3s 1

## 43 Running 'Rgui.R' [4s 1

## 44 Running 'SP500-ex.R' 1

## 45 Running ‘aggregate.R 1

## 46 Running ‘as.edgelist 1

## 47 "Running ‘bdstest.R’\n" 1

## 48 Running ‘Class+Meth. 1

## 49 "Running ‘config.R’\n " 1

## 50 Running ‘cX-ui-funct 1

## 51 Running ‘DevEvalFile 1

## 52 "Running ‘emTests.R’\n" 1

## 53 "Running ‘group01.R’ " 1

## 54 Running ‘loop\_genera 1

## 55 Running ‘LTS-special 1

## 56 Running ‘rsolr\_unit\_ 1

## 57 "Running ‘run-all.R’\n" 1

## 58 Running ‘runTests.R’ 1

## 59 "Running ‘sleep.R’\n " 1

## 60 Running ‘SP500-ex.R’ 1

## 61 Running ‘test\_bccaq. 1

## 62 Running ‘test\_scs.R’ 1

## 63 Running ‘test-cluste 1

## 64 "Running ‘test.R’\nRun" 1

## 65 Running ‘tests.R’ [4 1

## 66 Running ‘testthat.r’ 1

We need to investigate more so let’s make some groups:

filter(det, Status == "ERROR") %>%

mutate(flavor\_flav = case\_when(

grepl("devel", Flavor) ~ "devel",

grepl("oldrel", Flavor) ~ "oldrel",

TRUE ~ "current"

)) %>%

filter(flavor\_flav != "devel") %>%

mutate(Output = trimws(Output)) %>%

mutate(output\_grp = case\_when(

grepl("^Running ", Output) ~ "Test/example run",

grepl("^Installation fail", Output) ~ "Install failed",

grepl("Package[s]\* requir", Output) ~ "Missing pacakge(s)",

grepl("Errors in running code in vig", Output) ~ "Vignette issue",

TRUE ~ Output

)) %>%

count(output\_grp, sort=TRUE)

## # A tibble: 4 x 2

## output\_grp n

##

## 1 Test/example run 743

## 2 Install failed 437

## 3 Missing pacakge(s) 358

## 4 Vignette issue 7

Much better. Now, let see *where* those are:

filter(det, Status == "ERROR") %>%

mutate(flavor\_flav = case\_when(

grepl("devel", Flavor) ~ "devel",

grepl("oldrel", Flavor) ~ "oldrel",

TRUE ~ "current"

)) %>%

filter(flavor\_flav != "devel") %>%

mutate(Output = trimws(Output)) %>%

mutate(output\_grp = case\_when(

grepl("^Running ", Output) ~ "Test/example run",

grepl("^Installation fail", Output) ~ "Install failed",

grepl("Package[s]\* requir", Output) ~ "Missing package(s)",

grepl("Errors in running code in vig", Output) ~ "Vignette issue",

TRUE ~ Output

)) %>%

filter(!grepl("solaris", Flavor)) %>% # I love ya Solaris, but you're not relevant anymore

count(output\_grp, Flavor, sort=TRUE) %>%

mutate(pct = n/sum(n))

## # A tibble: 16 x 4

## output\_grp Flavor n pct

##

## 1 Install failed r-oldrel-osx-x86\_64 256 0.197

## 2 Test/example run r-oldrel-windows-ix86+x86\_64 218 0.168

## 3 Missing package(s) r-oldrel-osx-x86\_64 156 0.120

## 4 Missing package(s) r-release-osx-x86\_64 133 0.102

## 5 Test/example run r-oldrel-osx-x86\_64 110 0.0847

## 6 Test/example run r-release-osx-x86\_64 80 0.0616

## 7 Install failed r-release-windows-ix86+x86\_64 73 0.0562

## 8 Test/example run r-patched-linux-x86\_64 57 0.0439

## 9 Test/example run r-release-linux-x86\_64 57 0.0439

## 10 Install failed r-oldrel-windows-ix86+x86\_64 46 0.0354

## 11 Test/example run r-release-windows-ix86+x86\_64 46 0.0354

## 12 Install failed r-release-osx-x86\_64 36 0.0277

## 13 Missing package(s) r-oldrel-windows-ix86+x86\_64 19 0.0146

## 14 Missing package(s) r-release-windows-ix86+x86\_64 5 0.00385

## 15 Vignette issue r-release-osx-x86\_64 4 0.00308

## 16 Vignette issue r-oldrel-osx-x86\_64 3 0.00231

Let’s poke a bit more, but let’s also be aware of the fact that some (many) ERRORs on “oldrel” are due to conditions [like this](https://www.r-project.org/nosvn/R.check/r-oldrel-osx-x86_64/tidystats-00install.html) where the package specifies that it can only be used in release++ versions of R. So we *kinda* have to go all Columbo on every ERROR or exclude “oldrel” (we’ll do the latter since this post is already long) and we should also ignore the missing packages ones since that’s more than likely a CRAN issue.

filter(det, Status == "ERROR") %>%

mutate(flavor\_flav = case\_when(

grepl("devel", Flavor) ~ "devel",

grepl("oldrel", Flavor) ~ "oldrel",

TRUE ~ "current"

)) %>%

filter(!(flavor\_flav %in% c("oldrel", "devel"))) %>%

filter(!grepl("solaris", Flavor)) %>%

mutate(Output = trimws(Output)) %>%

mutate(output\_grp = case\_when(

grepl("^Running ", Output) ~ "Test/example run",

grepl("^Installation fail", Output) ~ "Install failed",

grepl("Package[s]\* requir", Output) ~ "Missing package(s)",

grepl("Errors in running code in vig", Output) ~ "Vignette issue",

TRUE ~ Output

)) %>%

filter(output\_grp != "Missing package(s)") %>%

distinct(Package)

## # A tibble: 254 x 1

## Package

##

## 1 AER

## 2 archdata

## 3 atlantistools

## 4 BAMBI

## 5 biglmm

## 6 biglm

## 7 BIOMASS

## 8 blockingChallenge

## 9 broom

## 10 clusternomics

## # … with 244 more rows

Now we have a target package list (+ ~26 in “FAIL”) that can very likely legitimately have issues. We’ll let more practical data scientists than I am figure out the dependency tree member count for them and then determine proper features and model selection to come up with a far more legitimate “risk metric”.

**Just One More Thing**

Did you read the bit about the 03-18 RDS having some serious differences from 03-19 one? Yeah, so perhaps any model needs to be run a few times or the data collected over the course of time to ensure we’re working with as clean a dataset as possible. Y’know, ask practical data science questions like:

* What data is available to me?
* Will it help me solve the problem?
* Is it enough?
* Is the data quality good enough?

**FIN**

Never contrive an analysis just to fit your preferred message.

Assumptions *matter*. Analysis setup *matters*. Domain expertise *matters*. Dataset knowledge *matters*.

I can sum that up as *mindfulness matters*, and if you approach a project that way then go forth and LIBRARY ALL THE THINGS you need to accomplish your goals.