R Testing Package Survey

Seth Russell

6/19/2018

This notebook performs an analysis of use of testing methodologies for all R packages avaialble on CRAN.

Two methods are used and results are compared.

Method 1: Grep for non-empty testing directories. The commonly used R testing packages all recommend placing tests in a directory by themselves, which we look for.

Method 2: Check for stated dependencies on testing packages. It is considered best practice to list dependencies on a testing framework even though standard usage of a package may not require it.

I've Picked 2008 (10 years) as the last updated cutoff year based on manual inspection. While there are still packages in CRAN that were last updated in 2007 and prior, there are not very many, so ignore them in the visualizations.

```
## Loading required package: xml2
```

```
## Loading required package: future
```

```
##
## Attaching package: 'future.apply'
```

```
## The following object is masked from 'package:future':
##
##     future_lapply
```

Method 1 of testing analysis

This method downloads packages from CRAN and evaluates the files and folders for the presence of a non-empty test directory.

Output hidden as when running against entire CRAN it will be very large.

```
# if value 100%, then read all packages, otherwise, randomly select number of packages provided
sample_size <- '100%'
# dont need to untar, but is useful for manual analysis
untar_files <- FALSE

# Although most unit testing packages use test/ or tests/ as the common base directory, some
# such as svUnit recommend the directory /inst/unitTests
#
# code at https://github.com/rorynolan/exampletestr/blob/master/analysis/CRAN_test_analysis.Rmd misses
# test directories since it checks for "^tests/.+". As an example, R packages using svUnit (such as gsu
# often call the test directory unitTests
#

# call code from shared_fn.R to download files and extract them if desired
package_df <- initialize(search_pattern="[Tt]est[^/]*/.+", sample_size=sample_size, untar_files=untar_f

if (nrow(package_df[package_df$download_error == TRUE, ]) > 0) {
  print('Unable to download or untar these packages - they will not be considered in analysis')
  print(package_df[package_df$download_error == TRUE, ])
```

```
} else {
  print('It appears that all files downloaded and untared successfully.')
}

print(paste('As of', date(), 'there are', nrow(package_df), 'packages available on CRAN'))
```

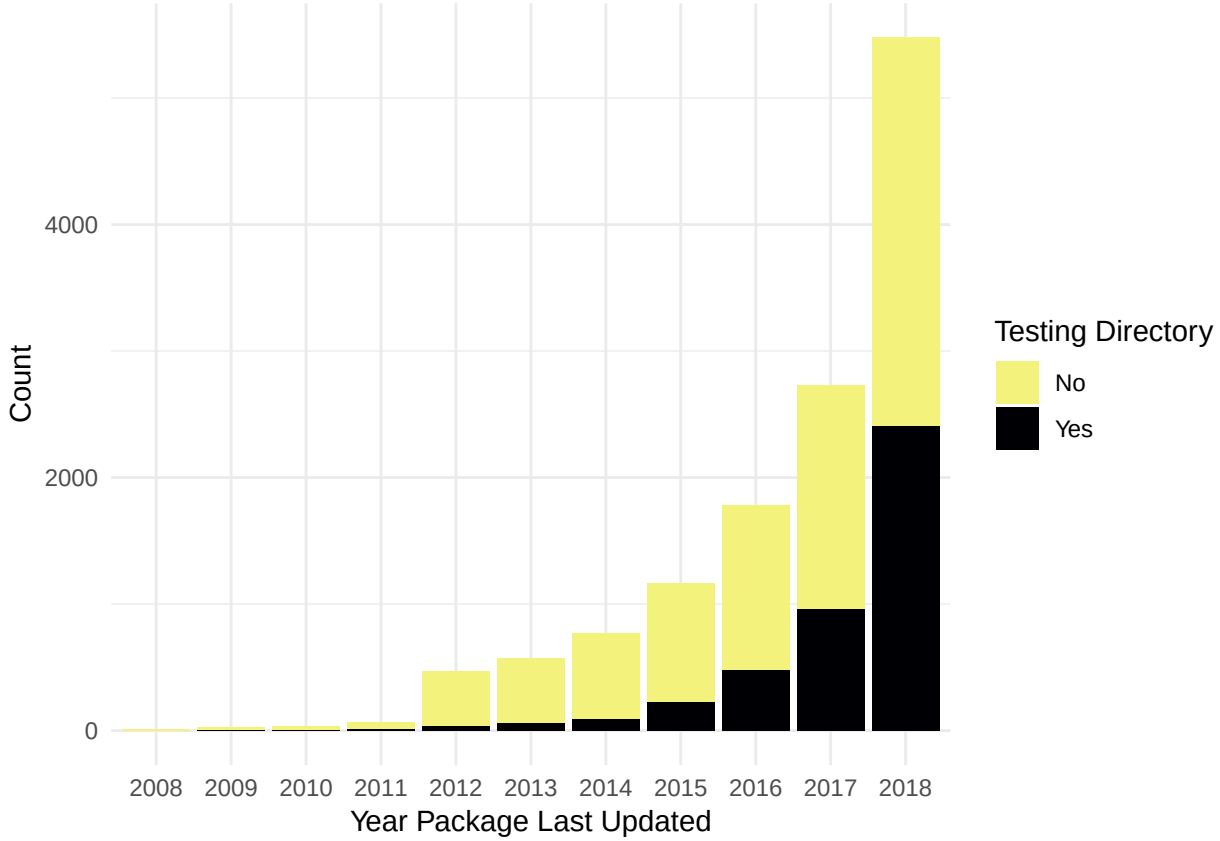
Visualizations for method 1

Histogram by year with testing packages shown

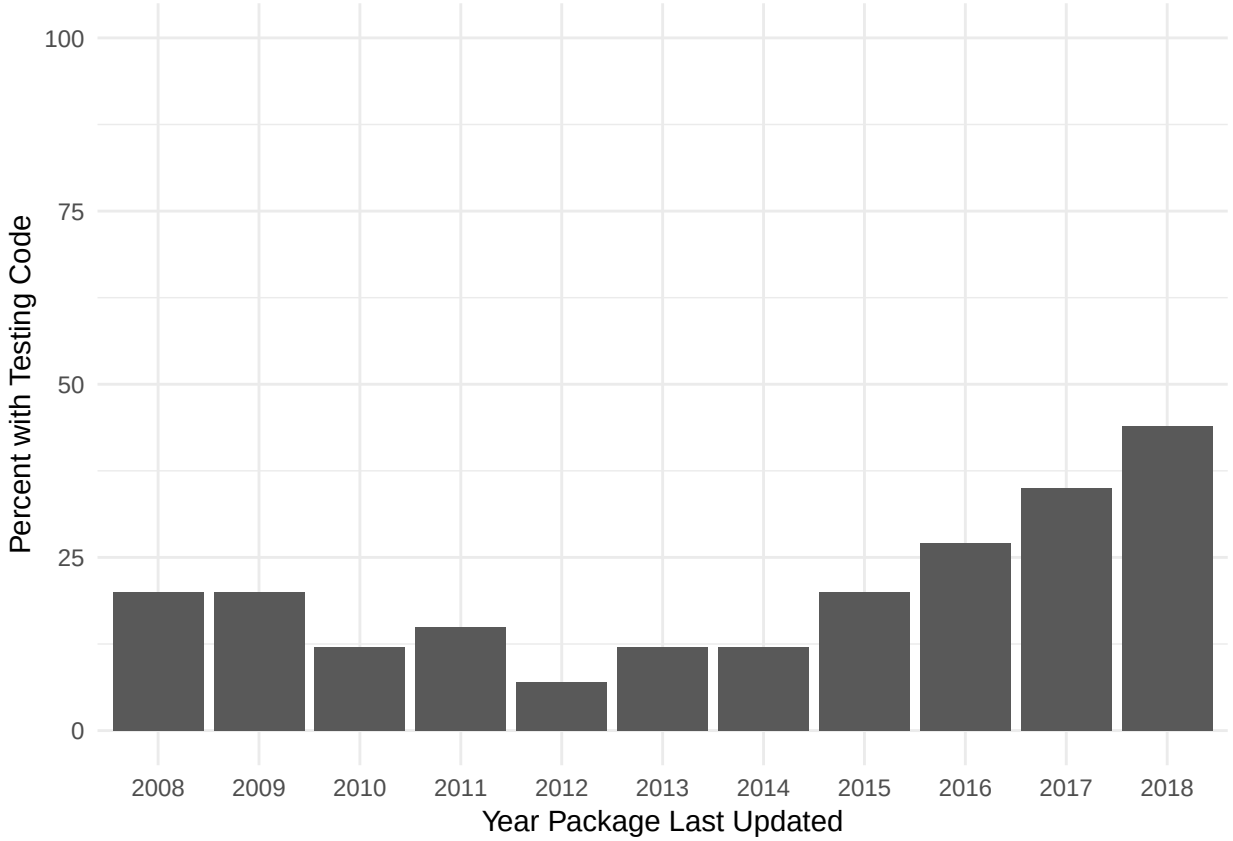
```
# build data needed for plots
gtf <- grep_table_freqs(package_df)
gtt <- grep_table_totals(gtf)
```

```
## [1] "Summary Table for Grep Results"
##       [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]   [,8]   [,9]
## Year  "2005" "2006" "2007" "2008" "2009" "2010" "2011" "2012" "2013"
## Total "   1" "   4" "   1" "  10" "  25" "  32" "  65" " 467" " 573"
## Grep  NA     "   1" NA     "   2" "   5" "   4" "  10" "  33" "  66"
## Pct   NA     "25"   NA     "20"   "20"   "12"   "15"   " 7"   "12"
##       [,10]  [,11]  [,12]  [,13]  [,14]
## Year  "2014" "2015" "2016" "2017" "2018"
## Total " 768" "1165" "1783" "2725" "5475"
## Grep  "  96" " 228" " 483" " 960" "2407"
## Pct   "12"   "20"   "27"   "35"   "44"
```

```
# generate plots
freq_plot <- grep_viz_freq(gtf, image_prefix = 'testing', label = 'Testing')
freq_plot
```




```
pct_plot <- grep_viz_pct(gtt, image_prefix = 'testing', label = 'Testing')
pct_plot
```



Method 2 of testing analysis

```
# function from shared_fn.R - sets package_df$package_deps
package_df <- calc_dependencies(package_df)
```

```
## testthat    RUnit   testit   svUnit unitizer unittest
##     3277      129       25       10        3        1
```

```
# are there any that have more than one dependency listed?
# these are all counted multiple times in histogram
show_multiple_dependencies(package_df, look_for_packages)
```

```
## Packages with 0 dependencies: 9665
## Packages with 1 dependencies: 3413
## Packages with 2 dependencies: 16
## Multiple Dependency Table (dep=2):
##  RUnit,testthat testit,testthat
##               9               7
## Packages with 3 dependencies: 0
## Packages with 4 dependencies: 0
## Packages with 5 dependencies: 0
## Packages with 6 dependencies: 0
## Packages with 7 dependencies: 0
```

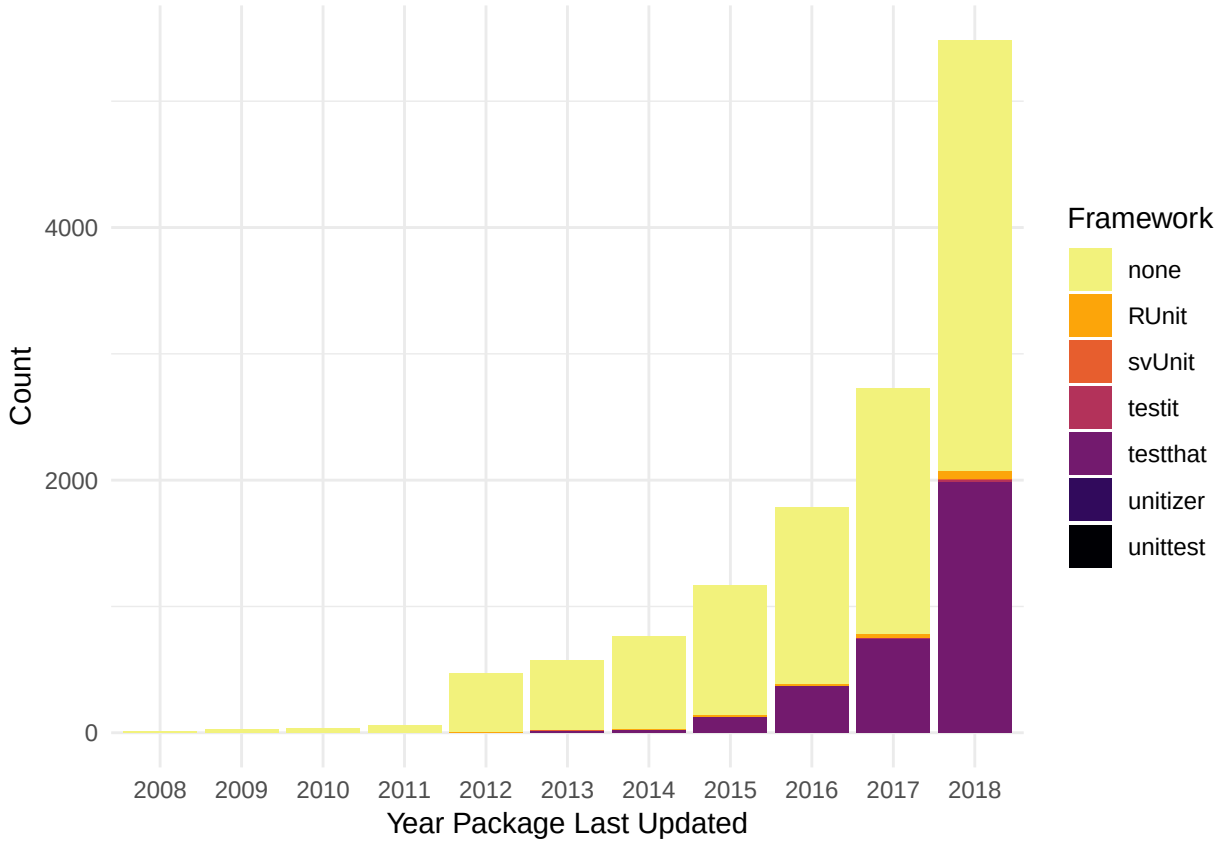
Visualizations for method 2

Histogram by year with testing packages shown

```
# need to merge transformed_df with package_df to keep those with no dependency and create better histo
transformed_df <- transform_df(package_df)
transformed_df <- merge(transformed_df, package_df[package_df$package_deps == 'none', ][colnames(transf

dep_freqs <- aggregate(strftime(transformed_df$date, "%Y"),
                       by=list(strftime(transformed_df$date, "%Y"), transformed_df$package_deps),
                       FUN=length)

names(dep_freqs) <- c('Year', 'Dependency', 'Count')
fplot <- ggplot(data=dep_freqs[dep_freqs$Year > 2007, ], aes(x=Year, y=Count, fill=Dependency)) +
  geom_bar(stat="identity") +
  scale_fill_viridis_d(direction = -1, option="inferno", end = 0.96) +
  xlab("Year Package Last Updated") +
  labs(fill = "Framework")
fplot
```




```
ggsave(filename = paste0(image_base, "testing_dependency_stacked_bar.png"), fplot,
       width = 7.2, height = 5.5, dpi = 600, units = "in", device='png')

dep_totals <- dependency_table(dep_freqs)
print('Dependency Table:')
```

```
## [1] "Dependency Table:"
```

```
dep_totals
```

```
##              [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]   [,8]
## Year         "2005" "2006" "2007" "2008" "2009" "2010" "2011" "2012"
## Count        "   1" "   4" "   1" "  10" "  25" "  32" "  65" " 467"
## svUnit       "0"    "0"    "0"    "0"    "0"    "0"    "0"    "1"
## RUnit        " 0"   " 0"   " 0"   " 0"   " 0"   " 0"   " 1"   " 3"
## RUnit_Pct    "0"    "0"    "0"    "0"    "0"    "0"    "2"    "1"
## testthat     "   0" "   0" "   0" "   0" "   0" "   0" "   1" "   1"
## testthat_Pct " 0"   " 0"   " 0"   " 0"   " 0"   " 0"   " 2"   " 0"
## testit       " 0"   " 0"   " 0"   " 0"   " 0"   " 0"   " 0"   " 0"
## unitizer     "0"    "0"    "0"    "0"    "0"    "0"    "0"    "0"
## unittest     "0"    "0"    "0"    "0"    "0"    "0"    "0"    "0"
##              [,9]   [,10]  [,11]  [,12]  [,13]  [,14]
## Year         "2013" "2014" "2015" "2016" "2017" "2018"
## Count        " 573" " 768" "1168" "1786" "2726" "5484"
## svUnit       "1"    "1"    "0"    "2"    "1"    "4"
## RUnit        " 1"   " 6"   "13"   " 9"   "33"   "63"
## RUnit_Pct    "0"    "1"    "1"    "1"    "1"    "1"
## testthat     "  21" "  25" " 127" " 372" " 745" "1985"
## testthat_Pct " 4"   " 3"   "11"   "21"   "27"   "36"
## testit       " 0"   " 0"   " 2"   " 2"   " 3"   "18"
## unitizer     "0"    "0"    "0"    "0"    "1"    "2"
## unittest     "0"    "0"    "0"    "0"    "1"    "0"
```

Method comparison

As the results from method 1 do not match the results from method 2, explore some of the differences.

Some points discovered: * It is possible for an R package to use a non-standard testing directory, even if they also use a standardized testing framework * It is possible to list a testing framework as a dependency and not actually use the framework. * Speculation - Testing can be performed in-line with other code through use of stop() or stopifnot(). * Speculation - Developers may have their own test cases they run while developing their software, but do not commit to a repository or share with others.

As new packages are being released and updated all the time, these numbers will change.

```
# About 859 more appear to have testing code...
print("Difference between packages that have testing code vs dependency on testing framework:")
```

```
## [1] "Difference between packages that have testing code vs dependency on testing framework:"
```

```
nrow(package_df[package_df$found == TRUE, ])  - nrow(package_df[package_df$package_deps != 'none', ])
```

```
## [1] 866
```

```
# About 1070 packages that don't list a dependency to a testing framework, but have a testing directory
# Randomly looked at several: some don't use a framework (e.g. xlsx)
# others use a framework but don't list it as a dependency (e.g. redcapAPI)
only_grep <- package_df[package_df$found == TRUE & package_df$package_deps == 'none', ]
print(paste("Number of packages that don't list a dependency to a testing framework, but have a testing
```

```
## [1] "Number of packages that don't list a dependency to a testing framework, but have a testing dire
```

```
# About 211 packages that list a dependency to a testing framework, but don't use one
only_dep <- package_df[package_df$found == FALSE & package_df$package_deps != 'none', ]
print(paste("Number of packages that list a dependency to a testing framework, but don't use one:", nrow
```

```
## [1] "Number of packages that list a dependency to a testing framework, but don't use one: 214"
```