An Update on Broom

Alex Hayes

2018-07-18

Outline

1. What is broom?
2. Progress so far
    - Bug fixes and pull requests
    - New test suite
    - New documentation
3. broom 0.5.0 release
4. Lessons learned
5. The future of broom

What is broom?

`broom` makes it easy to work with model objects

- `tidy()` summarizes information about model components
- `glance()` reports information about the entire model
- `augment()` adds informations about observations to a dataset
- Output is a tidy tibble.
- Easy to interact with programmatically.

Quick usage examples

```
fit <- lm(hp ~ ., mtcars)

tidy(fit)
```

```
## # A tibble: 11 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)    79.0      185.      0.428   0.673
## 2 mpg            -2.06       2.09   -0.987   0.335
## 3 cyl             8.20      10.1      0.813   0.425
## # ... with 8 more rows
```

```
glance(fit)
```

What I've been working on

Breakdown of time spent so far

- 2 weeks of bug fixes and merging pull requests
- 2 weeks writing tests
- 2 weeks writing documentation

Bug fixes and pull requests: notes

- Triaged several years of issues
- Bugs are generally easy to fix
    - But there are a lot
- Pull requests make the world go round
    - High levels of contributor enthusiasm!!!
    - Contributors have wide range of R experience
    - Tests sometime missing or limited
    - Documentation sometimes sparse or missing

Closed ~70 bugs and ~30 pull requests so far

Bug fixes and pull requests: adding tidiers vignette

Goal: Make it easier to make a good PR

- Missing key piece: documentation about standards
- New vignette addresses this
  - Work in progress
  - Like `CONTRIBUTING.md` on steroids
  - Eventually: full walk-through like recipes custom steps vignette

https://broom.tidyverse.org/articles/adding-tidiers.html

Test suite: coverage

- Pre 0.5.0 line coverage ~40 percent
- Most lines have some coverage
  - ~80 percent line coverage
  - Higher because deprecated tests skipped
- Line coverage less important than model coverage

Aside: model coverage

Aside:

```
# glance.arima coverage was 100 percent.
# tested output of:
glance(arima(lh, order = 1:3))

# but this was broken until recently:
glance(arima(lh, order = 1:3, method = "CSS"))
```

- Same class can correspond to many varied model objects
- Hard to write varied tests for unfamiliar model objects

Aside: subtle bugs

- Easy to extract wrong `df` from model
  - About to change `df` for `lm` objects
- Arguments can disappear into `...`
  - Not sure how to test
  - Current approach: warnings in documentation

Example: arguments disappearing into ...

```
fit <- lm(hp ~ ., mtcars)

# misspelled argument
td <- tidy(fit, conf.int = TRUE, comf.level = 0.9)

# no error, output looks exactly like
# you might expect
```

Test suite: tibble output

Test that

- `tidy()`, `glance()`, and `augment()` return tibbles.
- `glance()` returns a single row.
- `augment()` does some input validation.
    - In progress

```
fit <- lm(hp ~ ., mtcars)
td <- tidy(fit)
check_tidy_output(td)
```

Test suite: argument checking

```
check_arguments(tidy.lm)
```

- Checks arguments against master list
- Checks default arguments
  - Shouldn't be missing
  - `conf.int = FALSE`
  - `conf.level = 0.95`
  - `conf.int` and `conf.level` always come as a pair

Goal: enforce consistency, especially in new PRs

- Checked this semi-manually in 0.5.0
- Tests will automate this in 0.7.0

Test suite: column naming

```
library(lavaan)

cfa.fit <- cfa(
   'F =~ x1 + x2 + x3 + x4 + x5',
   data = HolzingerSwineford1939, group = "school"
)

select(glance(cfa.fit), 1:5)
```

```
## # A tibble: 1 x 5
##    agfi   aic   bic   cfi chisq
##   <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 0.971 4473. 4584. 0.766  99.3
```

Test suite: column naming strategy

- Goal: push consistency burden onto PR authors
- Describe acceptable column names in `tidy.yaml`:

```
- column: AIC
  description: Akaike's Criterion.
  used_by:
    - ivreg
```

- Compile `tidy.yaml` into a `column_glossary` tibble
- Export `column_glossary` (downstream package maintainers have asked for this)
- Test output column names against `column_glossary`
- Populate documentation from `column_glossary`

Documentation: templates

Many repeated arguments:

```
tidy.betareg <- function(x,
  conf.int = FALSE,
  conf.level = .95, ...)

tidy.ivreg <- function(x,
  conf.int = FALSE,
  conf.level = .95,
  exponentiate = FALSE, ...)
```

Should share documentation for `conf.int`.

Documentation: templates

`roxygen2` templates make this easy:

```
@template param_confint
```

Where `man-roxygen/param_confint.R` looks like:

```
#' @param conf.int Logical indicating
#'   whether or not to include a
#'   confidence interval in the tidied
#'   output. Defaults to `FALSE`.
#' @md
```

Documentation: templates

Templates currently used to generate:

- `@title`,
- `@description`,
- `@params`, and
- some `@return`

documentation sections.

Documentation: individualize documentation

Previously `tidy.object`, `glance.object` and `augment.object` would all be documented together.

- Gave each function it's own roxygen documentation and Rd file
  - Less magical
  - Heavily cross-linked and aliased
- Replaced lots of confusing documentation like:

```
#' @rdname augment.lm
#' @export
augment.glm <- augment.lm
```

Documentation: return columns

Goal for 0.7.0: populate `@return` from `column_glossary`.

- Writing the glossary will take lots of time
- Currently have `@template` prototype
  - Hadley recommended using `@evalRd` instead

broom 0.5.0

broom 0.5.0: features

- Tibble output
- New test suite
- New documentation
  - Vignettes
  - Function documentation
- ~10 new tidiers (all contributed)
- Tons of bug fixes (mostly contributed)

broom 0.5.0: tibble output

Tibbles break some things, mostly when:

- subsetting with `[` and expecting a vector.
- setting rownames on a tibble.
- using augment on models making use of matrix covariates / outcomes.
  - i.e. `survival::Surv()`

broom 0.5.0: matrix column and augment example

```
y <- rnorm(5)
x <- matrix(rnorm(10), nrow = 5)

df <- data.frame(x, y)   # ok
tibble::tibble(x, y)     # errors

fit <- lm(y ~ x, df)     # problem: this works
augment(fit)             # this goes kaboom
```

Passing data argument can help:

```
augment(fit, data = df)  # happy again
```

broom 0.5.0: deprecations

- Broom tidies some non-statistical objects
- Moving away from this. Deprecating
  - `tidy.data.frame()`
  - `tidy.matrix()`
  - `tidy.numeric()`
  - `tidy.character()`
- Should use `tibble::as_tibble()` instead
- Couple more of these coming in 0.7.0

broom 0.5.0: deprecations: mixed models

Moving tidiers for

- lme, lme4 and nmle models,
- brms models,
- rstanarm models, and
- mcmc objects

to Ben Bolker's `broom.mixed` package

Lessons learned

Making systematic changes is time consuming

- 100+ tidiers
- Model objects are unfamiliar, oftentimes idiosyncratic
- Changing all tidiers (i.e. new tests/doc) take 1-1.5 weeks

Broom depends on high quality PRs

- If you do not use a model, writing a good tidier is incredibly difficult
  - What information is important?
  - What do people use it for?
  - Documentation for models varies in quality
  - Can be hard to understand workflow
  - Model objects, input and output format can all be weird
- A good PR means you don't have to deal with this
- A bad PR means you still have to do most of this work

Key: empower contributors to make high quality PRs

`augment()` is hard

Original thought: `tidy()` is most ambiguous method, will be hardest to work with

Incorrect: `augment()` is hard

- Need different behavior for `data` and `newdata` args
- People often don't implement it
- Have to deal with model both model input and output

There are many useful way to represent a model

Representations of a fit model:

- Mathematical: $y \sim \mathcal{N}(X\hat{\beta}, \sigma^2)$
- Code object: `fit <- lm(hp ~ . , mtcars); fit`
- Relational: `tidy(fit), glance(fit), augment(fit)`
- ???

Opinion: need a *tidy modelling* paper to clarify the key objects in play like *tidy data* did

The Future of Broom

The big split

What: split broom into domain specific tidying packages

Why: high maintanence and design burden

Delays:

- Want to clean up internals, which were messier than anticipated
- Tidier behavior not fully specified
    - `augment()` NA behavior
- Lots of tidiers don't meet existing specifications

The big split: vision

- import `tidy()`, `glance()` and `augment()` from `modelgenerics`
- `broom` tidies models in `base` and `stats`
- domain-specific packages import broom
    - tests guarantee tidiers meet specification
- some system for tracking where tidiers live

```
library(tidymodels)  # load everything
```

Possible domain specific packages

- `sweep`
- `tidytext`
- `broomstick`
- `broom.mixed`
- `biobroom`
- `broom.base`
- `schoenberg`
- `tidybayes`

Should `tidy.betareg` live in the `betareg` package?

No. At least, not yet. Tidiers are not consistent enough at the moment. The definitions of `tidy()`, `glance()` and `augment()` are not yet strict enough to guarantee consistency across packages.

Timeline

Priorities somewhat indeterminate at the moment

- July: finalize tidier specifications
- July: start collaborating on domain specific packages
- Early August: implement as much spec as possible for 0.7.0 release
- August 20: internship ends
- Late August: 0.7.0 release
    - I will likely take over as package maintainer
    - Unclear if broom will be split by this point
- Early September: grad school starts (dev slows down)
- September+: rewrite the broom paper with Dave

Questions?

Read more about broom 0.5.0 release on the tidyverse blog.

You can follow broom development on our Github page.

@alexpghayes on Twitter
alexpghayes@gmail.com