Solving the model representation problem with `broom`

Alex Hayes (RStudio Intern 2018, `broom` maintainer)

2019-01-18

The model representation problem

Representing a fit in mathematical terms

Hopefully it feels natural to describe models mathematically

- Normal model: $y_i \sim \mathrm{Normal}(\mu, \sigma^2)$

Representing a fit in mathematical terms

Hopefully it feels natural to describe models mathematically

- Normal model: $y_i \sim \mathrm{Normal}(\mu, \sigma^2)$
  - Fit from normal model: $y_i \sim \mathrm{Normal}(-2, 1)$

Representing a fit in mathematical terms

Hopefully it feels natural to describe models mathematically

- Normal model: $y_i \sim \mathrm{Normal}(\mu, \sigma^2)$
  - Fit from normal model: $y_i \sim \mathrm{Normal}(-2, 1)$
  - Can also identify fits by parameter vectors: $\theta = (-2, 1)$

Representing a fit in mathematical terms

Hopefully it feels natural to describe models mathematically

- Normal model: $y_i \sim \mathrm{Normal}(\mu, \sigma^2)$
  - Fit from normal model: $y_i \sim \mathrm{Normal}(-2, 1)$
  - Can also identify fits by parameter vectors: $\theta = (-2, 1)$
  - Estimators for normal model:

Representing a fit in mathematical terms

Hopefully it feels natural to describe models mathematically

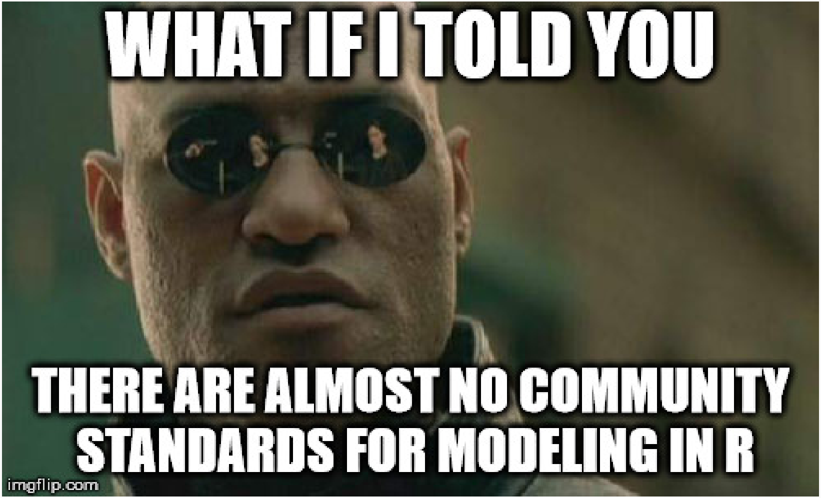
- Normal model: $y_i \sim \mathrm{Normal}(\mu, \sigma^2)$
  - Fit from normal model: $y_i \sim \mathrm{Normal}(-2, 1)$
  - Can also identify fits by parameter vectors: $\theta = (-2, 1)$
  - Estimators for normal model:
  - $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$

Representing a fit in mathematical terms

Hopefully it feels natural to describe models mathematically

- Normal model: $y_i \sim \mathrm{Normal}(\mu, \sigma^2)$
  - Fit from normal model: $y_i \sim \mathrm{Normal}(-2, 1)$
  - Can also identify fits by parameter vectors: $\theta = (-2, 1)$
  - Estimators for normal model:
  - $\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i \qquad \hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2$
- Key: shared notation and community standards

How does R represent models, estimators and fits?



A taste of the pain

Suppose you want class probabilities from a fit called `obj`:

Object	Code
`lda`	`predict(obj)`
`glm`	`predict(obj, type = "response")`
`gbm`	`predict(obj, type = "response", n.trees)`
`mda`	`predict(obj, type = "posterior")`
`rpart`	`predict(obj, type = "prob")`
`Weka`	`predict(obj, type = "probability")`
`logitboost`	`predict(obj, type = "raw", nIter)`
`pamr.train`	`pamr.predict(obj, type = "posterior")`

The model representation problem

We have no shared framework or understanding of how to represent statistical models, estimation methods and fits with R objects.

The `broom` package

`broom` provides a standard way to represent fits

1. `tidy()`: summarize information about fit components

`broom` provides a standard way to represent fits

1. `tidy()`: summarize information about fit components
2. `glance()`: report goodness of fit measures

`broom` provides a standard way to represent fits

1. `tidy()`: summarize information about fit components
2. `glance()`: report goodness of fit measures
3. `augment()`: add information about observations to a dataset

The normal model: an example

```r
# simulate Normal(-2, 1) data
x <- rnorm(5000, -2, 1)

# create a fit object using
# MLE estimator and normal model

normal_fit <- MASS::fitdistr(
  x,
  dnorm,
  start = list(mean = 0, sd = 1)
)
```

What is `normal_fit`?

```
## $estimate
##      mean        sd
## -2.034081  1.014121
##
## $sd
##       mean         sd
## 0.01434184 0.01014118
##
## $vcov
##               mean            sd
## mean  2.056884e-04 -4.617406e-13
## sd   -4.617406e-13  1.028435e-04
##
## $loglik
## [1] -7164.801
```

What is the tidy representation of `normal_fit`?

```
library(tidyverse)
library(broom)

tidy(normal_fit)
## # A tibble: 2 x 3
##   term  estimate std.error
##   <chr>    <dbl>     <dbl>
## 1 mean     -2.03    0.0143
## 2 sd        1.01    0.0101
```

What is the tidy representation of `normal_fit`?

```
glance(normal_fit)
## # A tibble: 1 x 4
##   logLik        AIC    BIC  nobs
##   <logLik>   <dbl>  <dbl> <int>
## 1 -7164.801 14334. 14347.  5000
```

There's no `augment()` method defined for univariate distributions at the moment.

Another example: the linear model

```r
# create a fit object using the
# OLS estimator for the linear model
ols_fit <- lm(hp ~ mpg + cyl, mtcars)

# try the following for yourself:
```

```r
str(ols_fit)
```

The tidy representation of `lm` objects

```
tidy(ols_fit)
## # A tibble: 3 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)    54.1      86.1      0.628 0.535
## 2 mpg            -2.77      2.18    -1.27  0.213
## 3 cyl            24.0       7.35     3.26  0.00281
```

The tidy representation of lm objects

```
glance(ols_fit)
## # A tibble: 1 x 12
##   r.squared adj~1 sigma sta~2 p.value    df log~3   AIC
##       <dbl> <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl>
## 1     0.709 0.689  38.2  35.4 1.66e-8     2 -160.  329.
## # ... with abbreviated variable names
## #   1: adj.r.squared, 2: statistic, 3: logLik, and 4
## #   more variables: BIC <dbl>, deviance <dbl>,
## #   df.residual <int>, nobs <int>
```

The tidy representation of lm objects

```
augment(ols_fit)
## # A tibble: 32 x 10
##   .rownames    hp   mpg   cyl .fi~1 .resid   .hat .si~2
##   <chr>     <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl> <dbl>
## 1 Mazda RX4   110  21       6 140.  -29.7  0.0320  38.5
## 2 Mazda RX~   110  21       6 140.  -29.7  0.0320  38.5
## 3 Datsun 7~    93  22.8     4  86.7   6.28 0.121   38.9
## # ... with 29 more rows, abbreviated variable names
## #   1: .fitted, 2: .sigma, and 2 more variables:
## #   .cooksd <dbl>, .std.resid <dbl>
```

Use cases

Use cases

Report model coefficients with `tidy()`

```r
kable2 <- function(data)
  knitr::kable(mutate_if(data, is.numeric, round, 2))

tidy(ols_fit) %>%
  kable2()
```

term	estimate	std.error	statistic	p.value
(Intercept)	54.07	86.09	0.63	0.53
mpg	-2.77	2.18	-1.27	0.21
cyl	23.98	7.35	3.26	0.00

Comparing models by goodness of fit measures

```r
fits <- list(
  fit1 = lm(hp ~ cyl, mtcars),
  fit2 = lm(hp ~ cyl + mpg, mtcars),
  fit3 = lm(hp ~ ., mtcars)
)

gof <- map_df(fits, glance, .id = "model") %>%
  arrange(AIC)
```

Comparing models by goodness of fit measures

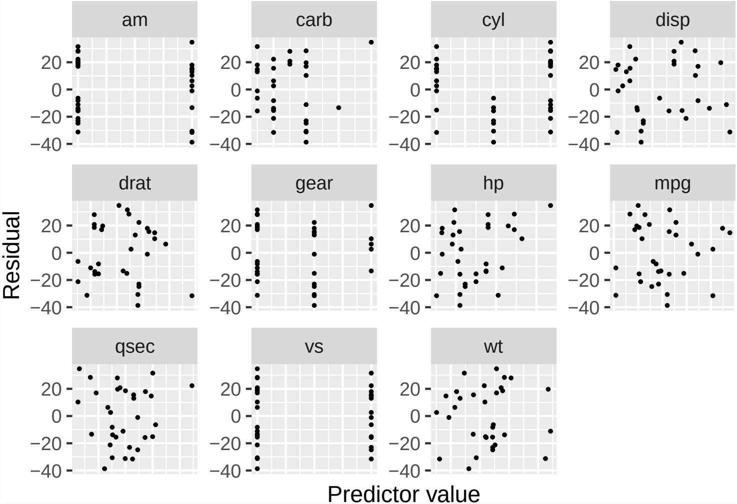
```
gof
## # A tibble: 3 x 13
##   model r.squared adj~1 sigma sta~2 p.value    df log~3
##   <chr>     <dbl> <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl>
## 1 fit3      0.903 0.857  26.0  19.5 1.90e-8    10 -143.
## 2 fit1      0.693 0.683  38.6  67.7 3.48e-9     1 -161.
## 3 fit2      0.709 0.689  38.2  35.4 1.66e-8     2 -160.
## # ... with abbreviated variable names
## #   1: adj.r.squared, 2: statistic, 3: logLik, and 5
## #   more variables: AIC <dbl>, BIC <dbl>,
## #   deviance <dbl>, df.residual <int>, nobs <int>
```

Inspecting residuals from multiple linear regression

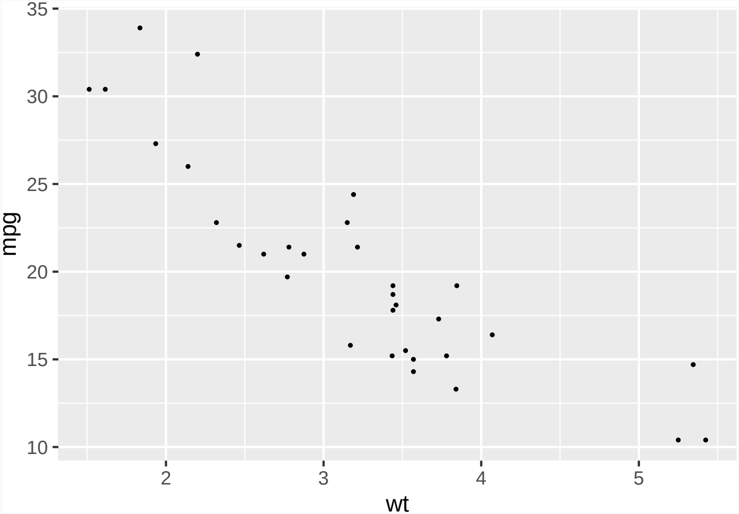
```r
fit <- lm(hp ~ ., mtcars)
au <- broom::augment(fit)

p <- au %>%
  gather(x, val, -contains(".")) %>%
  ggplot(aes(val, .resid)) +
  geom_point() +
  facet_wrap(~x, scales = "free") +
  labs(x = "Predictor value", y = "Residual") +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank())
```

Inspecting residuals from multiple linear regression



Bootstrapping



Bootstrapping

Consider a model:

$$\texttt{mpg} = \frac{k}{\texttt{wt}} + b + \varepsilon, \quad \varepsilon \sim \mathrm{Normal}(0, \sigma^2)$$

Suppose we want to know the sampling distributions of k and b via bootstrapping

Bootstrapping

```
library(rsample)

boots <- bootstraps(mtcars, times = 100)
boots
## # Bootstrap sampling
## # A tibble: 100 x 2
##   splits          id
##   <list>          <chr>
## 1 <split [32/7]>  Bootstrap001
## 2 <split [32/11]> Bootstrap002
## 3 <split [32/10]> Bootstrap003
## # ... with 97 more rows
```

Bootstrapping

```r
fit_nls_on_bootstrap <- function(split) {
  nls(
    mpg ~ k / wt + b,
    analysis(split),
    start = list(k = 1, b = 0)
  )
}
```

Bootstrapping

```
boot_fits <- boots %>%
  mutate(fit = map(splits, fit_nls_on_bootstrap),
         coef_info = map(fit, tidy))

boot_fits
## # Bootstrap sampling
## # A tibble: 100 x 4
##   splits          id           fit    coef_info
##   <list>          <chr>        <list> <list>
## 1 <split [32/7]>  Bootstrap001 <nls>  <tibble [2 x 5]>
## 2 <split [32/11]> Bootstrap002 <nls>  <tibble [2 x 5]>
## 3 <split [32/10]> Bootstrap003 <nls>  <tibble [2 x 5]>
## # ... with 97 more rows
```

Bootstrapping

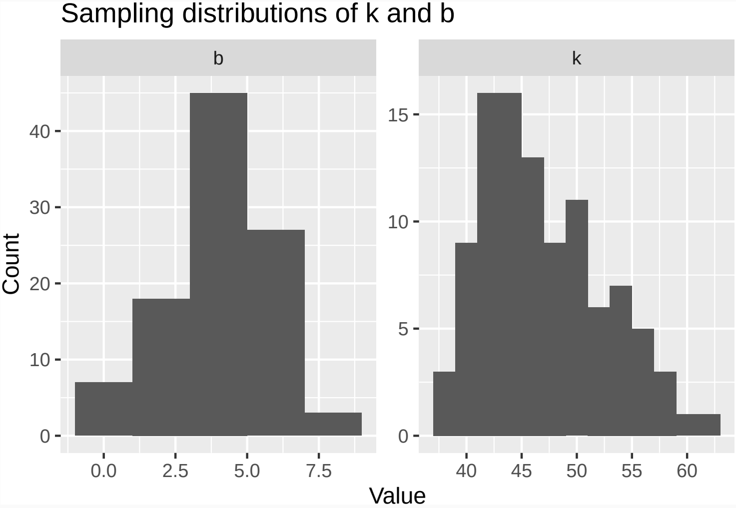
```
boot_coefs <- boot_fits %>%
  unnest(coef_info)

boot_coefs
## # A tibble: 200 x 8
##   splits           id      fit   term  est~1 std~2 sta~3
##   <list>           <chr>   <lis> <chr> <dbl> <dbl> <dbl>
## 1 <split [32/7]>  Bootst~ <nls> k     46.4   4.19 11.1
## 2 <split [32/7]>  Bootst~ <nls> b      4.49  1.60  2.80
## 3 <split [32/11]> Bootst~ <nls> k     43.4   3.85 11.3
## # ... with 197 more rows, abbreviated variable names
## #   1: estimate, 2: std.error, 3: statistic, and 1
## #   more variable: p.value <dbl>
```

Bootstrapping

```r
p <- ggplot(boot_coefs, aes(estimate)) +
  geom_histogram(binwidth = 2) +
  facet_wrap(~ term, scales = "free") +
  labs(
    title = "Sampling distributions of k and b",
    y = "Count",
    x = "Value"
  )
```

Bootstrapping

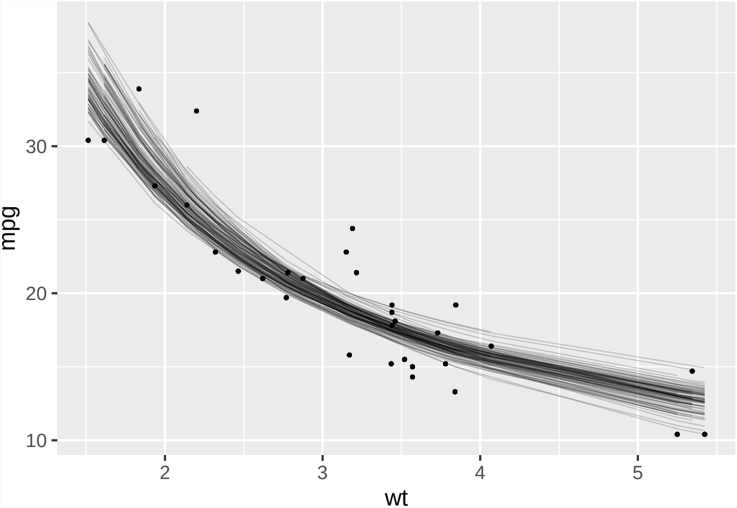


Bootstrapping

```
boot_aug <- boot_fits %>%
  mutate(augmented = map(fit, augment)) %>%
  unnest(augmented)

p <- ggplot(boot_aug, aes(wt, mpg)) +
  geom_point() +
  geom_line(aes(y = .fitted, group = id), alpha = 0.2)
```

Bootstrapping



Thank you! Questions?

Read about the `broom 0.5.0` release or how to implement new tidiers!

https://broom.tidyverse.org

@alexpghayes alexpghayes@gmail.com

alexpghayes.com