

# Solving the model representation problem with broom

---

Alex Hayes

2018-12-31

Who am I?

Statistical models

The model representation problem

The broom package

Use cases

`tidymodels`



These slides are available online!

<https://tinyurl.com/broom-talk>

**Who am I?**

---

# About Me



- Summer intern at RStudio last summer

# About Me



- Summer intern at RStudio last summer
- Primary maintainer of the [broom](#) package

# About Me



- Summer intern at RStudio last summer
- Primary maintainer of the [broom](#) package
- Just started a PhD in Statistics at UW-Madison



# About Me



- Summer intern at RStudio last summer
- Primary maintainer of the [broom](#) package
- Just started a PhD in Statistics at UW-Madison
- Active on [#rstats](#) Twitter and Github

# Statistical models

---

# What is a model?

Let  $x$  be values that live in some space  $\mathcal{X}$ , and let  $y$  be observations of interest that live in some space  $\mathcal{Y}$ . A **statistical model** is a *set* of probability distributions  $\mathcal{P}(y|x)$  indexed by parameters  $\theta \in \Theta$ <sup>1</sup>.

---

<sup>1</sup>In some cases we treat  $\theta$  as itself random, which means that our model is a class of probability distributions  $\mathcal{P}(y, \theta|x)$ .

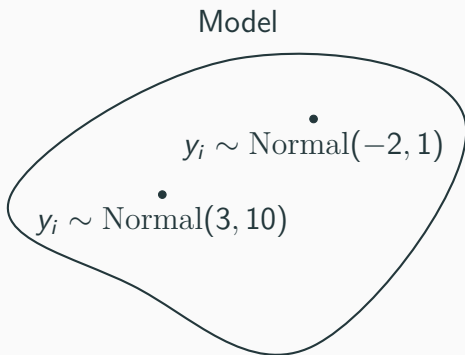
## Example: the normal model

$$y_i \stackrel{\text{iid}}{\sim} \text{Normal}(\mu, \sigma^2)$$

Here  $\theta = (\mu, \sigma^2)$  and the parameter space is  $\mathbb{R} \times \mathbb{R}^+$ .

# Visualizing the normal model

Again: a model is a *set*.



We call a single element of a model a **fit**. The distribution with  $\mu = -2, \sigma^2 = 1$  is a fit, for example.

## Another (parametric) example: the linear model

Given response  $y$  and predictor variables  $x_1$  and  $x_2$ , the linear model looks like:

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \varepsilon_i \quad \varepsilon_i \stackrel{\text{iid}}{\sim} \text{Normal}(0, \sigma^2)$$

This model says that  $y$  is i.i.d with a mean that depends on  $x$  and  $\vec{\beta}$ , and with fixed variance  $\sigma^2$ .

An **estimator** is a way to calculate the parameters of a model from data<sup>2</sup>. There are many estimators for any given model, and which one we think is best depends on how we define “agrees best with the data.”

---

<sup>2</sup>This is equivalent to selecting the best fit!

## Some estimators for the normal model

- $\hat{\mu} = 1$        $\hat{\sigma}^2 = 10$



## Some estimators for the normal model

- $\hat{\mu} = 1 \qquad \hat{\sigma}^2 = 10$

- $\hat{\mu} = \frac{1}{n} \sum_{i=1}^n x_i \qquad \hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$

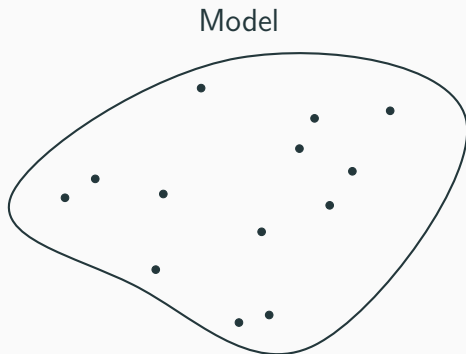
## Some estimators for the normal model

- $\hat{\mu} = 1 \quad \hat{\sigma}^2 = 10$

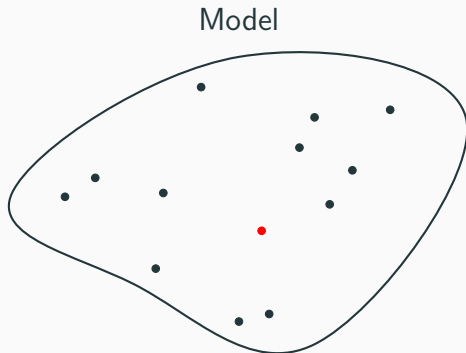
- $\hat{\mu} = \frac{1}{n} \sum_{i=1}^n x_i \quad \hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$

- $\hat{\mu} = \frac{1}{n} \sum_{i=1}^n x_i \quad \hat{\sigma}^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$

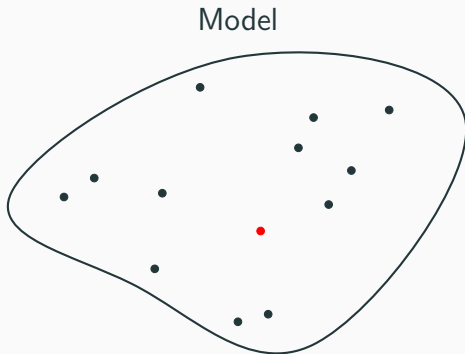
# Visualizing estimation



# Visualizing estimation

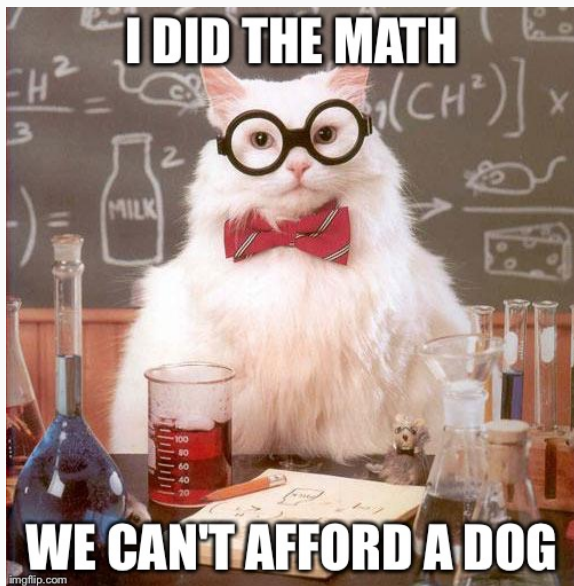


## Estimates found!



Best fit:  $y_i \sim \text{Normal}(-2, 1)$ . Finally we can do inference!

Do all the inference



# Some intuition about models, estimators and fits

A **model** corresponds to a set of possible truths about the world. A **fit** is a single truth about the world. An **estimator** is a way to choose the truth most suggested by the data from a set of many possible truths.

Aside: the actual truth is almost certainly *not* in the model!

# The model representation problem

---



# Representing a fit in mathematical terms

Hopefully it feels natural to describe models mathematically

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

# Representing a fit in mathematical terms

Hopefully it feels natural to describe models mathematically

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

# Representing a fit in mathematical terms

Hopefully it feels natural to describe models mathematically

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

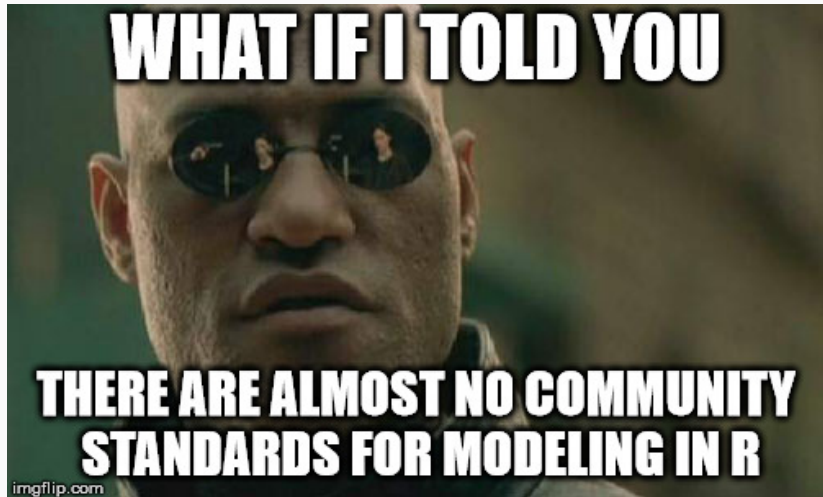
- Normal model:  $y_i \sim \text{Normal}(\mu, \sigma^2)$
- Fit from normal model:  $y_i \sim \text{Normal}(-2, 1)$ 
  - Can also identify fits by parameter vectors:  $\theta = (-2, 1)$
- Estimators for normal model:
  - $\hat{\mu} = \frac{1}{n} \sum_{i=1}^n x_i$        $\hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^n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

# Representing a fit in mathematical terms

Hopefully it feels natural to describe models mathematically

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

How does R represent models, estimators and fits?



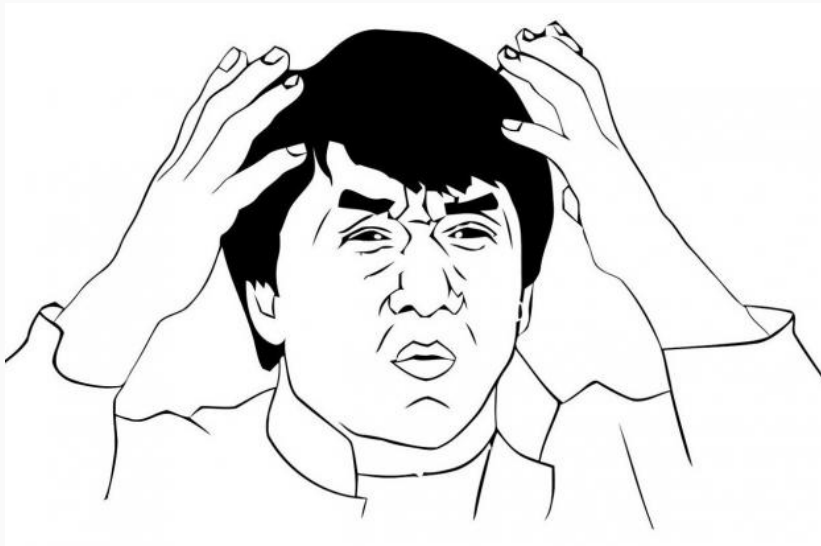
# A taste of the pain

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

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



How am I supposed to keep track of all this!?



# The model representation problem

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

- People who write and share code to enable data analysis are awesome

- People who write and share code to enable data analysis are awesome
- This isn't anyone's fault

- People who write and share code to enable data analysis are awesome
- This isn't anyone's fault
- Trying to get things done can still be frustrating

# The broom package

---

# The scene

Say we:

1. pick a model,

# The scene

Say we:

1. pick a model,
2. pick an estimator for that model, and then



# The scene

Say we:

1. pick a model,
2. pick an estimator for that model, and then
3. get an R object `fit` using that estimator.

# The scene

Say we:

1. pick a model,
2. pick an estimator for that model, and then
3. get an R object `fit` using that estimator.
  - The `fit` object is different for every model.

# The scene

Say we:

1. pick a model,
2. pick an estimator for that model, and then
3. get an R object `fit` using that estimator.
  - The `fit` object is different for every model.
  - The `fit` object could be great to work with, or awful.

# broom provides a standard way to represent fits

broom treats fits as having three parts:

1. A table of information about fit parameters

# `broom` provides a standard way to represent fits

`broom` treats fits as having three parts:

1. A table of information about fit parameters
2. A table of information about each observation used to estimate the fit

# broom provides a standard way to represent fits

broom treats fits as having three parts:

1. A table of information about fit parameters
2. A table of information about each observation used to estimate the fit
3. A table of overall goodness-of-fit measures

# broom provides a standard way to represent fits

broom treats fits as having three parts:

1. A table of information about fit parameters
2. A table of information about each observation used to estimate the fit
3. A table of overall goodness-of-fit measures
  - Each of table reported as a tidy tibble.

# broom provides a standard way to represent fits

broom treats fits as having three parts:

1. A table of information about fit parameters
2. A table of information about each observation used to estimate the fit
3. A table of overall goodness-of-fit measures
  - Each of table reported as a tidy tibble.
  - Together these constitute a *tidy representation* of a fit.



# What is a tibble

A **tibble** is a more consistent data.frame. tibbles are used widely throughout the **tidyverse**.

```
library(tibble)
```

```
tibble(x = 1:2, y = 3:4)
```

```
## # A tibble: 2 x 2
```

```
##       x       y
```

```
##   <int> <int>
```

```
## 1     1     3
```

```
## 2     2     4
```

# What does it mean to be tidy?

I highly recommend reading Hadley Wickham's [Tidy Data](#) paper! The short version: a **tidy dataset** is one where:

1. Each variable forms a column.

# What does it mean to be tidy?

I highly recommend reading Hadley Wickham's [Tidy Data](#) paper! The short version: a **tidy dataset** is one where:

1. Each variable forms a column.
2. Each observation forms a row.

# What does it mean to be tidy?

I highly recommend reading Hadley Wickham's [Tidy Data](#) paper! The short version: a **tidy dataset** is one where:

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

# The broom generics

So how do we turn fits into tidy tibbles?

- `tidy()` summarizes information about fit components

# The broom generics

So how do we turn fits into tidy tibbles?

- `tidy()` summarizes information about fit components
- `glance()` reports information about the entire fit

# The broom generics

So how do we turn fits into tidy tibbles?

- `tidy()` summarizes information about fit components
- `glance()` reports information about the entire fit
- `augment()` adds information about observations to a dataset

## Continuing with the normal model example

```
# 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, # use the normal model!
  start = list(mean = 0, sd = 1)
)
```



# What is normal\_fit?

```
str(normal_fit)
## List of 5
## $ estimate: Named num [1:2] -2.03 1.01
## ..- attr(*, "names")= chr [1:2] "mean" "sd"
## $ sd       : Named num [1:2] 0.0143 0.0101
## ..- attr(*, "names")= chr [1:2] "mean" "sd"
## $ vcov     : num [1:2, 1:2] 2.06e-04 -4.62e-13 -4.62e-13 1.03
## ..- attr(*, "dimnames")=List of 2
## .. ..$ : chr [1:2] "mean" "sd"
## .. ..$ : chr [1:2] "mean" "sd"
## $ loglik   : num -7165
## $ n        : int 5000
## - attr(*, "class")= chr "fitdistr"
```

# 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
##       n logLik    AIC    BIC
##   <int> <dbl> <dbl> <dbl>
## 1  5000 -7165. 14334. 14347.
```

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

## Another example: the linear model

```
# create a fit object using the  
# OLS estimator for the linear model  
ols_fit <- lm(hp ~ mpg + cyl, mtcars)  
  
# try the following for yourself:  
  
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)[, 1:5]
```

## # A tibble: 1 x 5

##	<i>r.squared</i>	<i>adj.r.squared</i>	<i>sigma</i>	<i>statistic</i>	<i>p.value</i>
##	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
## 1	0.709	0.689	38.2	35.4	0.0000000166

# The tidy representation of `lm` objects

```
augment(ols_fit)[, 1:7]
```

```
## # A tibble: 32 x 7
```

##	.rownames	hp	mpg	cyl	.fitted	.se.fit	.resid
##	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
## 1	Mazda RX4	110	21	6	140.	6.84	-29.7
## 2	Mazda RX4 Wag	110	21	6	140.	6.84	-29.7
## 3	Datsun 710	93	22.8	4	86.7	13.3	6.28
## #	... with 29 more rows						

# Notes about the tidy representation

- Always get a tibble back



# Notes about the tidy representation

- Always get a tibble back
- Column names in returned tibbles are consistent

# Notes about the tidy representation

- Always get a tibble back
- Column names in returned tibbles are consistent
- Some information in the original R object is lost

# Use cases

---

## Report model coefficients with tidy()

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

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

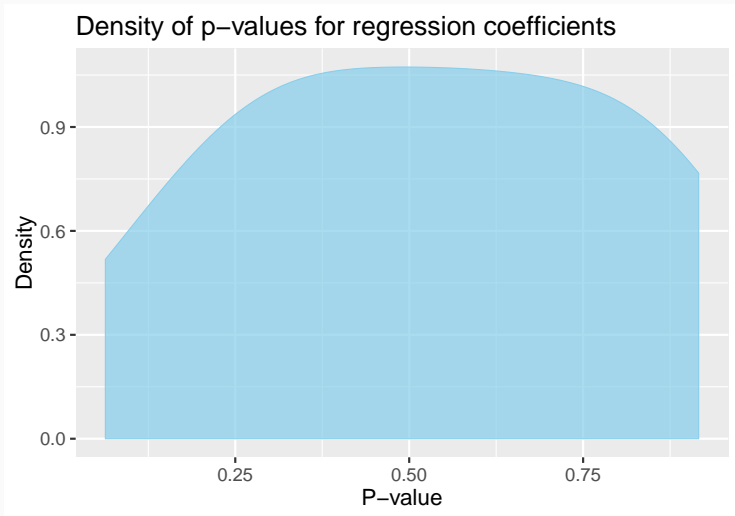
term	estimate	std.error	statistic	p.value
(Intercept)	54.07	86.09	0.63	0.53
mpg	-2.77	2.18	-1.27	0.21
cyl	23.98	7.35	3.26	0.00

# Plot histograms/densities of p-values

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

p <- ggplot(td, aes(p.value)) +
  geom_density(
    fill = "skyblue", color = "skyblue", alpha = 0.7
  ) +
  labs(
    title = "Density of p-values for regression coefficients",
    x = "P-value",
    y = "Density"
  )
```

# Plot histograms/densities of p-values



# Comparing models by goodness of fit measures

```
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

```
select(gof, -c(2:7))
```

## # A tibble: 3 x 6

##	model	logLik	AIC	BIC	deviance	df.residual
##	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<int>
## 1	fit3	-143.	310.	327.	14165.	21
## 2	fit1	-161.	329.	333.	44743.	30
## 3	fit2	-160.	329.	335.	42369.	29

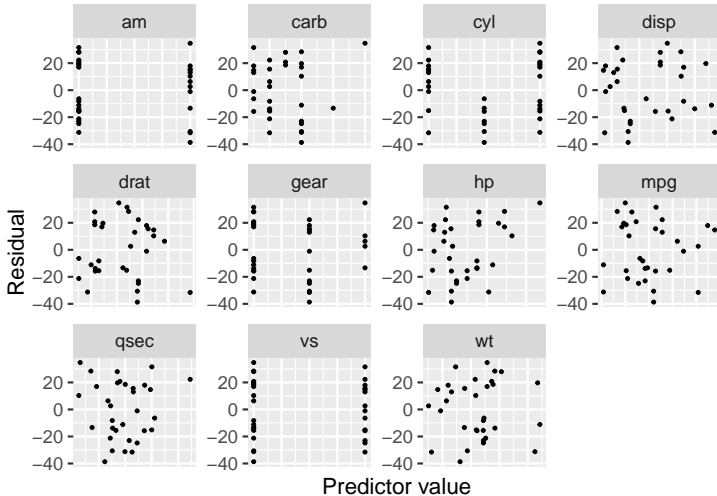


# Inspecting residuals from multiple linear regression

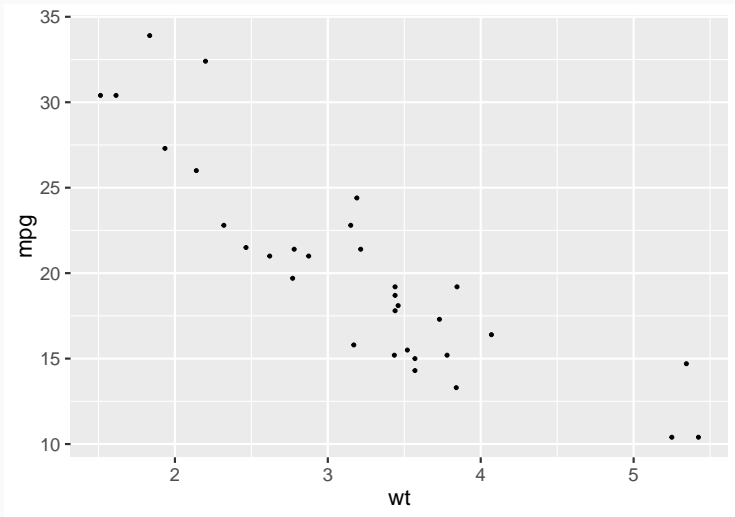
```
fit <- lm(hp ~ ., mtcars)
au <- broom::augment(fit)

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:

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

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 [32/10]> Bootstrap001
## 2 <split [32/7]>  Bootstrap002
## 3 <split [32/13]> Bootstrap003
## # ... with 97 more rows
```

# Bootstrapping

```
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 [32/10]> Bootstrap001 <S3: nls> <tibble [2 x 5]>
```

```
## 2 <split [32/7]>  Bootstrap002 <S3: nls> <tibble [2 x 5]>
```

```
## 3 <split [32/13]> Bootstrap003 <S3: nls> <tibble [2 x 5]>
```

```
## # ... with 97 more rows
```

# Bootstrapping

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

```
boot_coefs
```

```
## # A tibble: 200 x 6
```

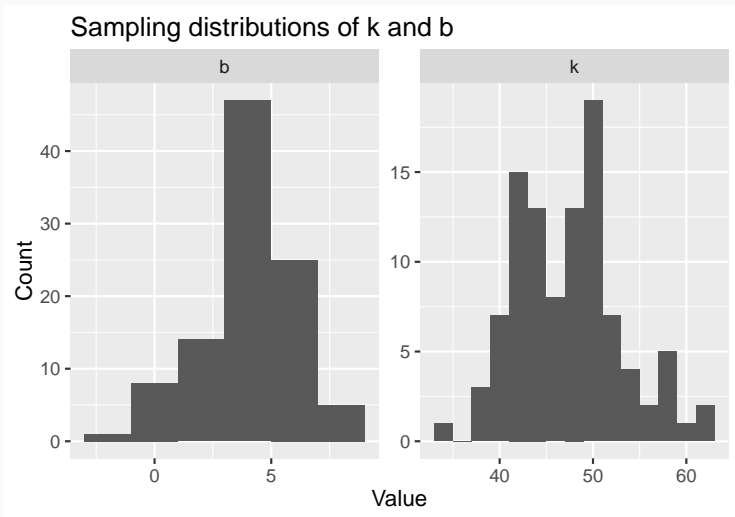
```
##   id            term estimate std.error statistic p.value  
##   <chr>         <chr>    <dbl>    <dbl>    <dbl>    <dbl>  
## 1 Bootstrap001 k         41.8      4.05     10.3  2.18e-11  
## 2 Bootstrap001 b          5.96     1.64      3.64  1.01e- 3  
## 3 Bootstrap002 k         50.6      3.96     12.8  1.16e-13  
## # ... with 197 more rows
```



# Bootstrapping

```
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"  
  )
```

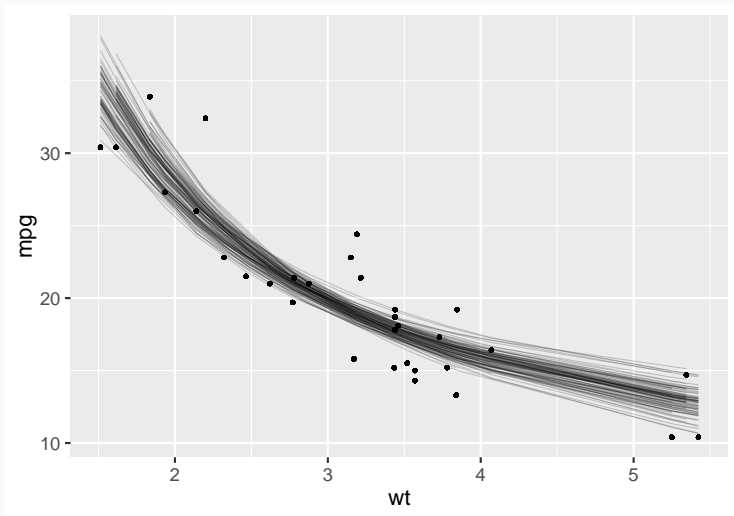
# Bootstrapping



# 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



# General strategy

`broom` provides tidying methods for 100+ classes!

1. Put fits in a list, or a list-column of a tibble

# General strategy

`broom` provides tidying methods for 100+ classes!

1. Put fits in a list, or a list-column of a tibble
2. Use `purrr::map()` to apply `tidy()`, `glance()` or `augment()` to each fit

# General strategy

`broom` provides tidying methods for 100+ classes!

1. Put fits in a list, or a list-column of a tibble
2. Use `purrr::map()` to apply `tidy()`, `glance()` or `augment()` to each fit
3. Use tidyverse tools to manipulate and visualize resulting data!

tidymodels

---





<https://github.com/tidymodels/tidymodels>

A general attempt to make modeling in R more consistent and tidy. Lots of [big projects](#). Highlights:

- [best practices for developing modeling packages](#)
- [parnsip](#): standardized modeling interfaces (in progress)
- [recipes](#): data pre-processing
- [rsample](#): infrastructure for resampling

# Questions?

Read about the recent broom release on the [tidyverse blog](https://broom.tidyverse.org).



<https://broom.tidyverse.org>



<https://github.com/tidymodels/broom/>



@alexpghayes



[alexpghayes@gmail.com](mailto:alexpghayes@gmail.com)