broom: An R Package to Convert Statistical Models into Tidy Data Frames

David Robinson

4/11/2015

What is tidy data?

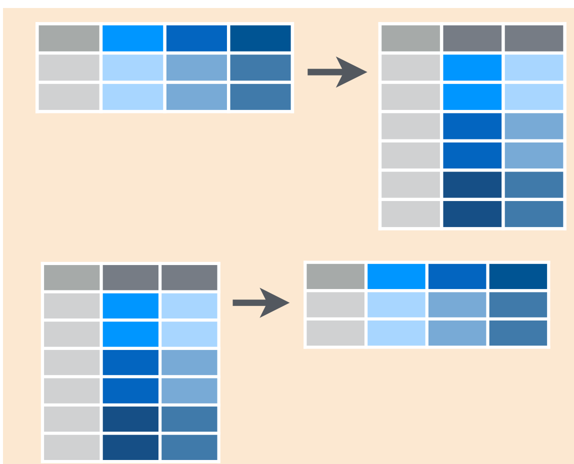
Data frames arranged as:

- One row for each *observation*
- One column for each *variable*
- One table for each *type of observational unit*

For details, see Tidy Data (Wickham 2014)

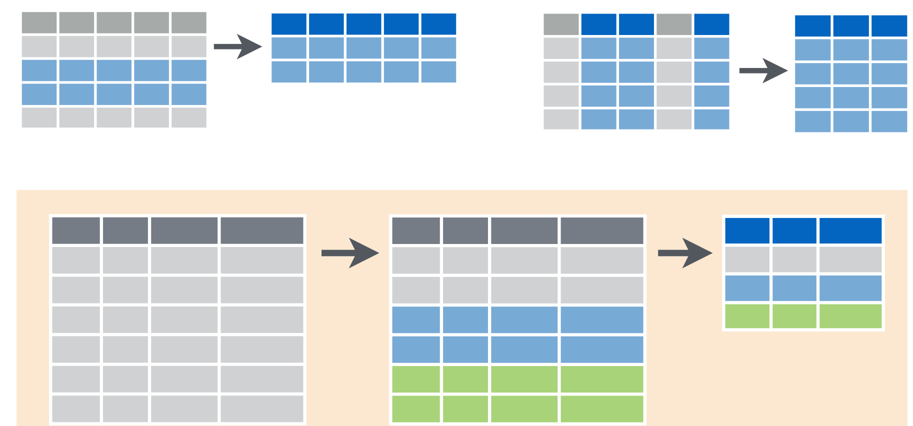
“Tidy tools” work with tidy data frames

tidyr



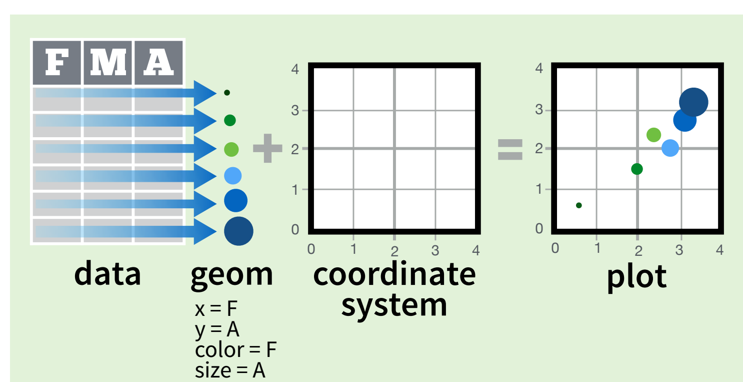
reshape data to be tidy

dplyr



manipulate and summarize tidy data

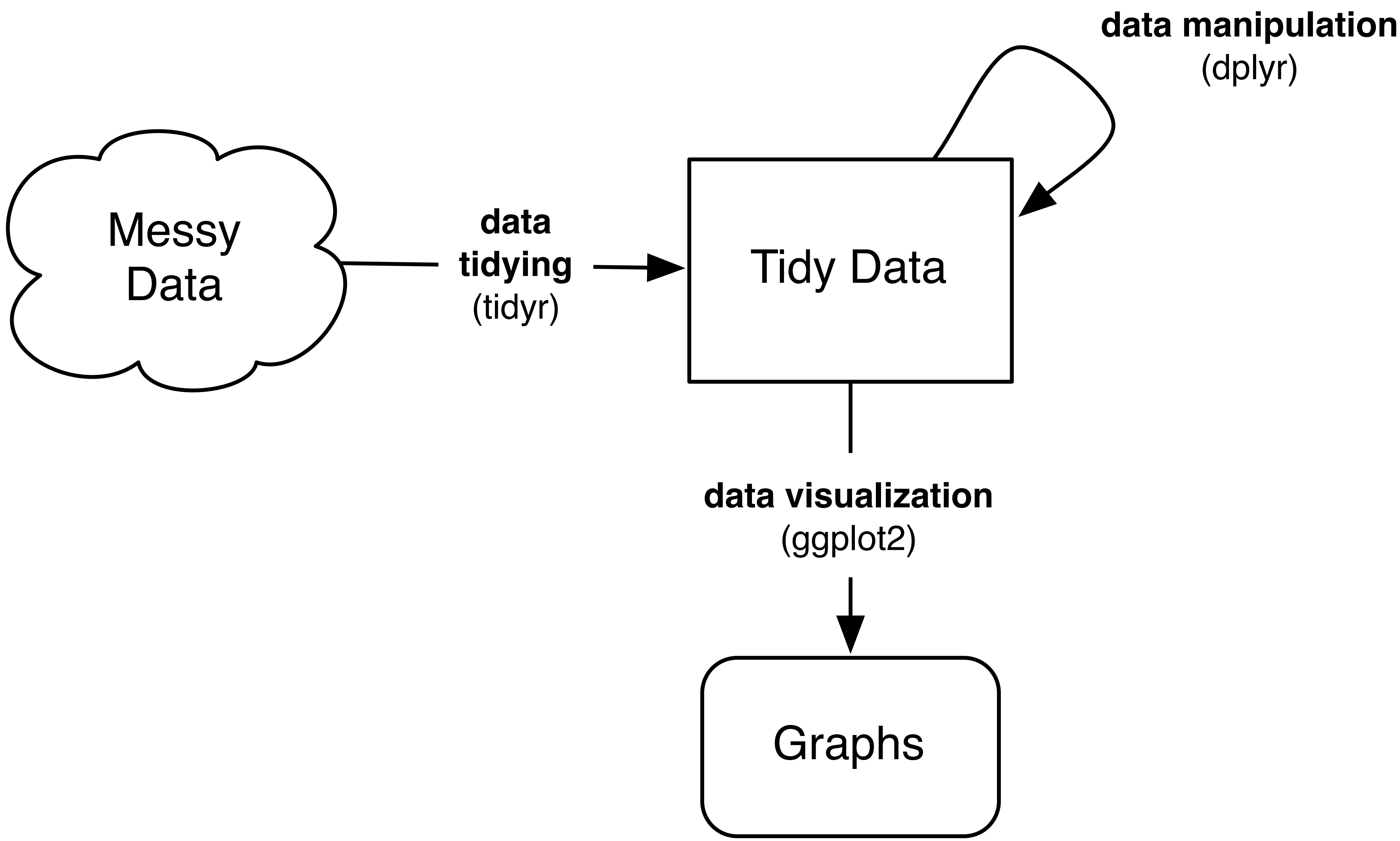
ggplot2



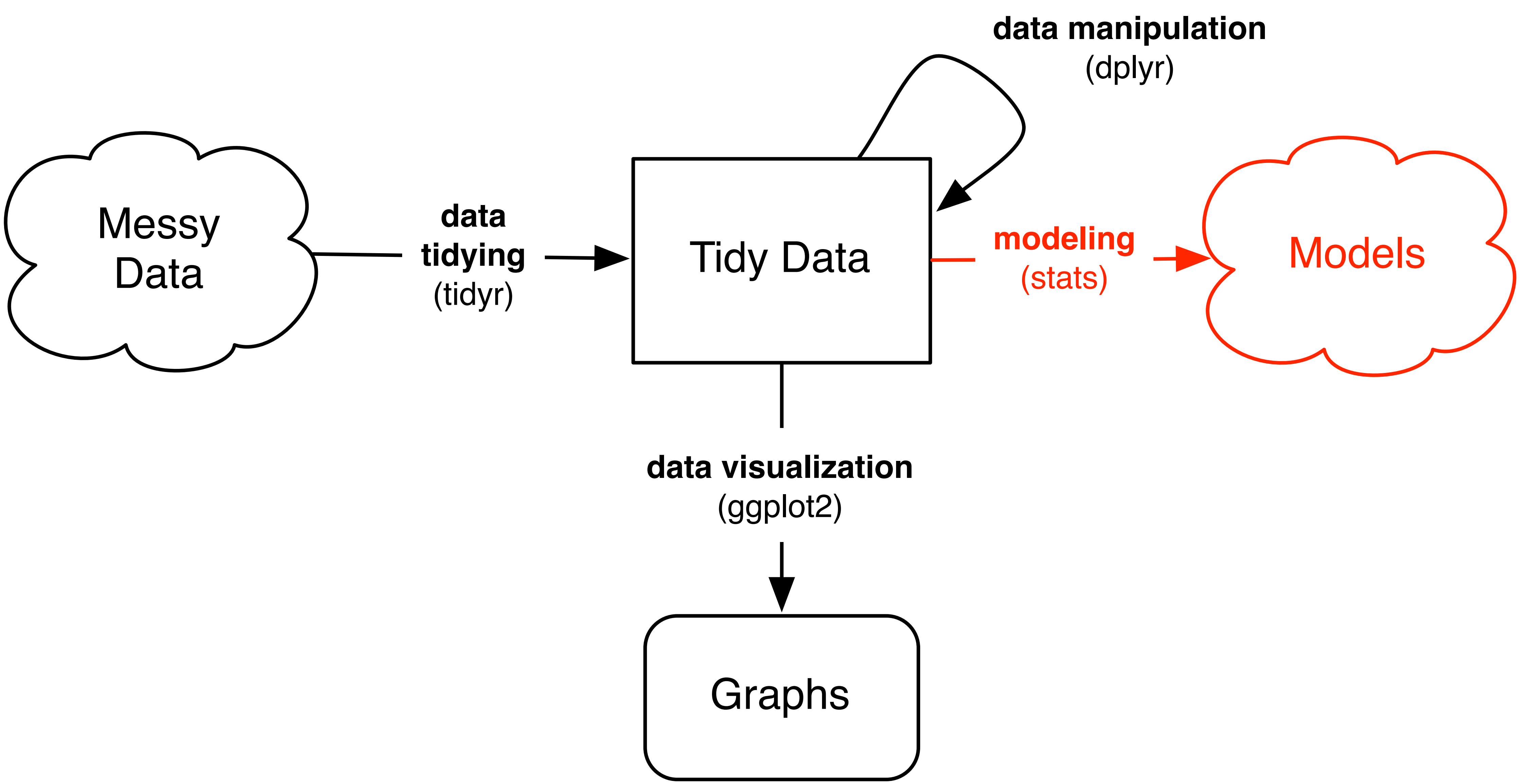
visualize tidy data

Source: RStudio Data Wrangling Cheatsheet
RStudio Data Visualization Cheatsheet

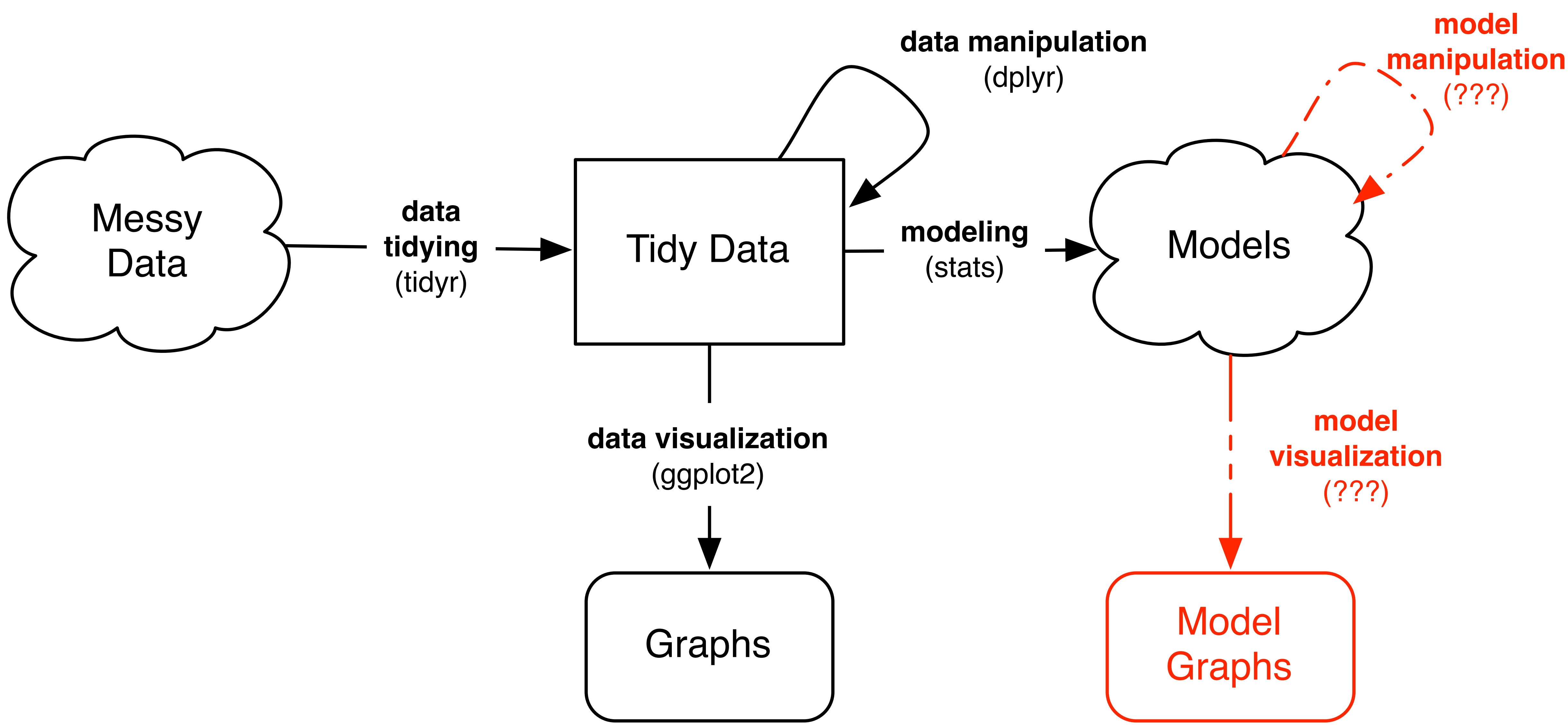
Tidy tools work together in exploratory data analysis



Everything works well until…



Visualizing and manipulating model objects is difficult



Model objects are messy

Example:
linear regression

What's "messy" about a linear regression?

```
> lmfit <- lm(mpg ~ wt + qsec, mtcars)
```

What's "messy" about a linear regression?

```
> summary(lmfit)

Call:
lm(formula = mpg ~ wt + qsec, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.3962 -2.1431 -0.2129  1.4915  5.7486 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  19.7462     5.2521   3.760 0.000765 ***
wt           -5.0480     0.4840 -10.430 2.52e-11 ***
qsec          0.9292     0.2650   3.506 0.001500 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.596 on 29 degrees of freedom
Multiple R-squared:  0.8264,  Adjusted R-squared:  0.8144 
F-statistic: 69.03 on 2 and 29 DF,  p-value: 9.395e-12
```

What's "messy" about a linear regression?

```
> summary(lmfit)

Call:
lm(formula = mpg ~ wt + qsec, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.3962 -2.1431 -0.2129  1.4915  5.7486 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  19.7462     5.2521   3.760 0.000765 ***
wt           -5.0480     0.4840 -10.430 2.52e-11 ***
qsec          0.9292     0.2650   3.506 0.001500 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.596 on 29 degrees of freedom
Multiple R-squared:  0.8264,	Adjusted R-squared:  0.8144 
F-statistic: 69.03 on 2 and 29 DF,  p-value: 9.395e-12
```

1. **Extracting coefficients takes multiple steps:**

```
data.frame(coef(summary(lmfit)))
```

What's "messy" about a linear regression?

```
> summary(lmfit)

Call:
lm(formula = mpg ~ wt + qsec, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-4.3962 -2.1431 -0.2129  1.4915  5.7486

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  19.7462     5.2521   3.760 0.000765 ***
wt           -5.0480     0.4840 -10.430 2.52e-11 ***
qsec          0.9292     0.2650   3.506 0.001500 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.596 on 29 degrees of freedom
Multiple R-squared:  0.8264,	Adjusted R-squared:  0.8144
F-statistic: 69.03 on 2 and 29 DF,  p-value: 9.395e-12
```

2. Information stored in row names (can't combine models)

What's "messy" about a linear regression?

```
> summary(lmfit)

Call:
lm(formula = mpg ~ wt + qsec, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.3962 -2.1431 -0.2129  1.4915  5.7486 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  19.7462     5.2521   3.760 0.000765 ***
wt           -5.0480     0.4840 -10.430 2.52e-11 ***
qsec          0.9292     0.2650   3.506 0.001500 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.596 on 29 degrees of freedom
Multiple R-squared:  0.8264,	Adjusted R-squared:  0.8144 
F-statistic: 69.03 on 2 and 29 DF,  p-value: 9.395e-12
```

3. Column names are inconvenient (access with `$"Pr(>|t|)"`, converts to `Pr...t..`)

What’s “messy” about a linear regression?

```
> summary(lmfit)

Call:
lm(formula = mpg ~ wt + qsec, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.3962 -2.1431 -0.2129  1.4915  5.7486 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  19.7462     5.2521   3.760 0.000765 ***
wt           -5.0480     0.4840 -10.430 2.52e-11 ***
qsec          0.9292     0.2650   3.506 0.001500 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.596 on 29 degrees of freedom
Multiple R-squared:  0.8264,	Adjusted R-squared:  0.8144 
F-statistic: 69.03 on 2 and 29 DF,  p-value: 9.395e-12
```

4. Information is computed in `print` method, not stored

What's "messy" about a linear regression?

```
> summary(lmfit)

Call:
lm(formula = mpg ~ wt + qsec, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.3962 -2.1431 -0.2129  1.4915  5.7486 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  19.7462     5.2521   3.760 0.000765 ***
wt           -5.0480     0.4840 -10.430 2.52e-11 ***
qsec          0.9292     0.2650   3.506 0.001500 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.596 on 29 degrees of freedom
Multiple R-squared:  0.8264,	Adjusted R-squared:  0.8144 
F-statistic: 69.03 on 2 and 29 DF,  p-value: 9.395e-12
```

1. Extracting coefficients takes multiple steps:

`data.frame(coef(summary(lmfit)))`

2. Information stored in row names (can't combine models)

3. Column names are inconvenient and inconsistent

4. Information is computed in `print` method, not stored

These inconveniences aren't an exception, they're the rule

broom's `tidy()` method does the work for you

```
> tidy(lmfit)
         term estimate std.error statistic  p.value
1 (Intercept)   19.746     5.252      3.76 7.65e-04
2          wt   -5.048     0.484    -10.43 2.52e-11
3        qsec    0.929     0.265      3.51 1.50e-03
```

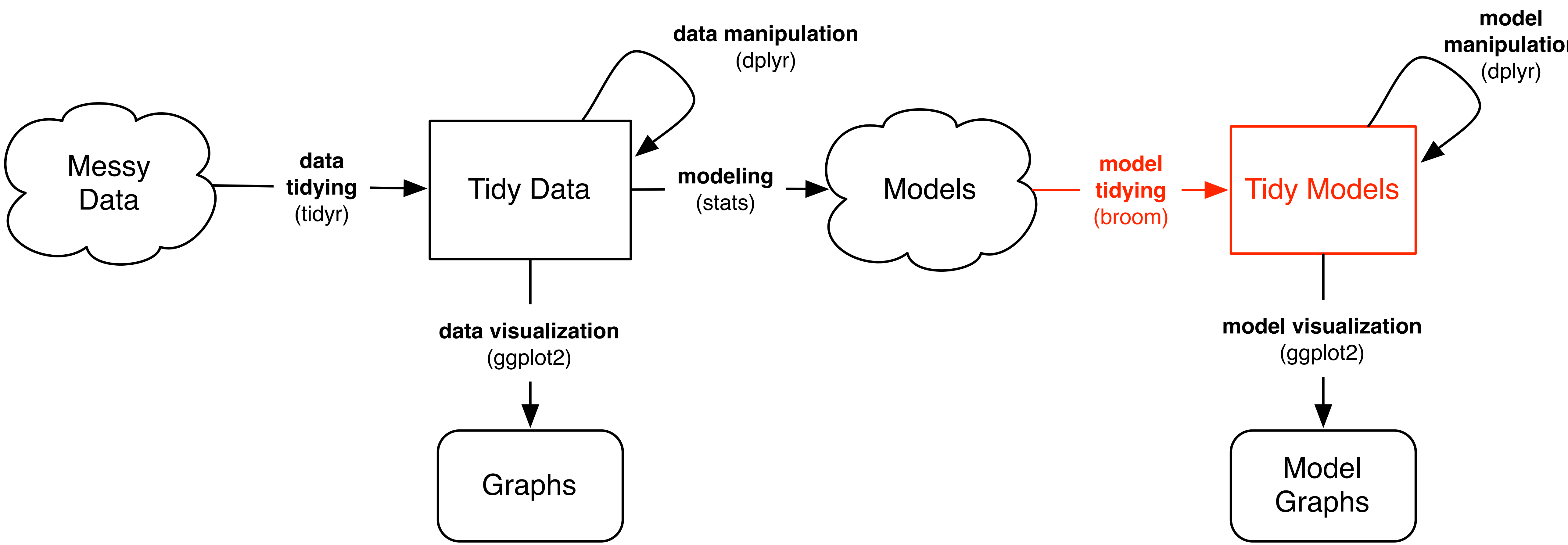
broom's `tidy()` method does the work for you

One function to call

```
> tidy(lmfit)
         term estimate std.error statistic  p.value
1 (Intercept)   19.746     5.252      3.76 7.65e-04
2          wt   -5.048     0.484    -10.43 2.52e-11
3        qsec    0.929     0.265      3.51 1.50e-03
```

Convenient column names

Information stored in columns, never row names broom takes model objects and turns them into tidy data frames that can be used with tidy tools


Messy Data
data tidying (tidyr)
Tidy Data
data manipulation (dplyr)
modeling (stats)
Models
model tidying (broom)
Tidy Models
model manipulation (dplyr)
data visualization (ggplot2)
Graphs
model visualization (ggplot2)
Model Graphs

Introduction to broom

```
> install.packages("broom")
> library(broom)
```

broom’s three methods

- broom defines tidying methods for extracting three kinds of statistics from an object:
  - **`tidy()`**: component-level statistics
  - **`augment()`**: observation-level statistics
  - **`glance()`**: model-level statistics

Example: three levels of a linear regression

```
> summary(lmfit)

Call:
lm(formula = mpg ~ wt + qsec, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.3962 -2.1431 -0.2129  1.4915  5.7486 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  19.7462     5.2521   3.760 0.000765 ***
wt           -5.0480     0.4840 -10.430 2.52e-11 ***
qsec          0.9292     0.2650   3.506 0.001500 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.596 on 29 degrees of freedom
Multiple R-squared:  0.8264,	Adjusted R-squared:  0.8144 
F-statistic: 69.03 on 2 and 29 DF,  p-value: 9.395e-12
```

Observation Level:
fitted values, residuals
`augment()`

Component Level:
coefficients, p-values
`tidy()`

Model Level:
R^2, F-statistic, deviance
`glance()`

The `tidy()` method: component-level statistics

```
> tidy(lmfit)
         term estimate std.error statistic  p.value
1 (Intercept)   19.746     5.252      3.76 7.65e-04
2          wt   -5.048     0.484    -10.43 2.52e-11
3        qsec    0.929     0.265      3.51 1.50e-03
```

each row is a coefficient

The `augment()` method: observation-level statistics

```
> augment(lmfit)
             .rownames  mpg   wt qsec .fitted .se.fit  .resid   .hat .sigma
1            Mazda RX4 21.0 2.62 16.5   21.82   0.683 -0.8151 0.0693   2.64
2        Mazda RX4 Wag 21.0 2.88 17.0   21.05   0.547 -0.0482 0.0444   2.64
3           Datsun 710 22.8 2.32 18.6   25.33   0.640 -2.5273 0.0607   2.60
4       Hornet 4 Drive 21.4 3.21 19.4   21.58   0.623 -0.1806 0.0576   2.64
5    Hornet Sportabout 18.7 3.44 17.0   18.20   0.512  0.5039 0.0389   2.64
6              Valiant 18.1 3.46 20.2   21.07   0.803 -2.9686 0.0957   2.58
7           Duster 360 14.3 3.57 15.8   16.44   0.701 -2.1434 0.0729   2.61
8            Merc 240D 24.4 3.19 20.0   22.23   0.730  2.1729 0.0791   2.61
9             Merc 230 22.8 3.15 22.9   25.12   1.410 -2.3237 0.2950   2.59
10            Merc 280 19.2 3.44 18.3   19.39   0.491 -0.1855 0.0358   2.64
11           Merc 280C 17.8 3.44 18.9   19.94   0.557 -2.1430 0.0460   2.61
12          Merc 450SE 16.4 4.07 17.4   15.37   0.615  1.0310 0.0561   2.63
13          Merc 450SL 17.3 3.73 17.6   17.27   0.520  0.0289 0.0402   2.64
14         Merc 450SLC 15.2 3.78 18.0   17.39   0.539 -2.1904 0.0431   2.61
15  Cadillac Fleetwood 10.4 5.25 18.0    9.95   1.092  0.4487 0.1768   2.64
16 Lincoln Continental 10.4 5.42 17.8    8.92   1.161  1.4757 0.2001   2.62
17   Chrysler Imperial 14.7 5.34 17.4    8.95   1.115  5.7486 0.1844   2.35
18            Fiat 128 32.4 2.20 19.5   26.73   0.751  5.6679 0.0836   2.39
19         Honda Civic 30.4 1.61 18.5   28.80   0.892  1.5975 0.1180   2.62
20      Toyota Corolla 33.9 1.83 19.9   28.97   0.909  4.9258 0.1226   2.45
```

each row is an observation from the original data

The `augment()` method: observation-level statistics

note that new columns start with .

```
> augment(lmfit)
             .rownames  mpg   wt qsec .fitted .se.fit  .resid   .hat .sigma
1            Mazda RX4 21.0 2.62 16.5   21.82   0.683 -0.8151 0.0693   2.64
2        Mazda RX4 Wag 21.0 2.88 17.0   21.05   0.547 -0.0482 0.0444   2.64
3           Datsun 710 22.8 2.32 18.6   25.33   0.640 -2.5273 0.0607   2.60
4       Hornet 4 Drive 21.4 3.21 19.4   21.58   0.623 -0.1806 0.0576   2.64
5    Hornet Sportabout 18.7 3.44 17.0   18.20   0.512  0.5039 0.0389   2.64
6              Valiant 18.1 3.46 20.2   21.07   0.803 -2.9686 0.0957   2.58
7           Duster 360 14.3 3.57 15.8   16.44   0.701 -2.1434 0.0729   2.61
8            Merc 240D 24.4 3.19 20.0   22.23   0.730  2.1729 0.0791   2.61
9             Merc 230 22.8 3.15 22.9   25.12   1.410 -2.3237 0.2950   2.59
10            Merc 280 19.2 3.44 18.3   19.39   0.491 -0.1855 0.0358   2.64
11           Merc 280C 17.8 3.44 18.9   19.94   0.557 -2.1430 0.0460   2.61
12          Merc 450SE 16.4 4.07 17.4   15.37   0.615  1.0310 0.0561   2.63
13          Merc 450SL 17.3 3.73 17.6   17.27   0.520  0.0289 0.0402   2.64
14         Merc 450SLC 15.2 3.78 18.0   17.39   0.539 -2.1904 0.0431   2.61
15  Cadillac Fleetwood 10.4 5.25 18.0    9.95   1.092  0.4487 0.1768   2.64
16 Lincoln Continental 10.4 5.42 17.8    8.92   1.161  1.4757 0.2001   2.62
17   Chrysler Imperial 14.7 5.34 17.4    8.95   1.115  5.7486 0.1844   2.35
18            Fiat 128 32.4 2.20 19.5   26.73   0.751  5.6679 0.0836   2.39
19         Honda Civic 30.4 1.61 18.5   28.80   0.892  1.5975 0.1180   2.62
20      Toyota Corolla 33.9 1.83 19.9   28.97   0.909  4.9258 0.1226   2.45
```

each row is an observation from the original data

The `glance()` method: model-level statistics

```
> glance(lmfit)
  r.squared adj.r.squared sigma statistic  p.value df logLik AIC BIC deviance
1     0.826         0.814   2.6        69 9.39e-12  3  -74.4 157 163      195
```

one row for the model

broom works across many kinds of model objects

Nonlinear least squares: **before**

```
> n <- nls(mpg ~ k * e ^ wt, data = mtcars, start = list(k = 1, e = 2))
> summary(n)

Formula: mpg ~ k * e^wt

Parameters:
  Estimate Std. Error t value Pr(>|t|)
k  49.6597     3.7888    13.1    6e-14 ***
e   0.7456     0.0199    37.5   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.67 on 30 degrees of freedom

Number of iterations to convergence: 10
Achieved convergence tolerance: 2.04e-06
```

Nonlinear least squares: **after**

```
> tidy(n)
  term estimate std.error statistic  p.value
1    k   49.660    3.7888      13.1 5.96e-14
2    e    0.746    0.0199      37.5 8.86e-27
```

each row is one estimated parameter

```
> augment(n)
   mpg   wt .fitted .resid
1 21.0 2.62    23.0 -2.012
2 21.0 2.88    21.4 -0.352
3 22.8 2.32    25.1 -2.331
4 21.4 3.21    19.3  2.076
5 18.7 3.44    18.1  0.611
6 18.1 3.46    18.0  0.117
...
```

each row is an observation from the original data

```
> glance(n)
  sigma isConv   finTol logLik AIC BIC deviance df.residual
1  2.67   TRUE 2.04e-06  -75.8 158 162      214          30
```

one row for the model

K-means clustering: **before**

```
> k
K-means clustering with 3 clusters of sizes 47, 103, 100

Cluster means:
  oracle      x1     x2
1   3.00 -3.3256 -2.398
2   2.03  0.0477  0.901
3   1.00  4.8963 -1.188

Clustering vector:
  [1] 2 3 2 2 2 2 2 2 1 3 2 3 2 2 2 3 1 3 2 3 2 3 3 3 2 3 3 3 2 2 1 2
 [33] 2 3 2 2 3 3 1 2 1 2 1 3 2 3 2 3 2 3 1 3 3 3 2 1 3 3 3 2 1 1 3 2
 [65] 3 3 3 2 3 2 2 1 3 2 2 2 3 3 2 2 1 3 3 3 2 2 3 2 3 1 2 2 2 3 2 3
 [97] 3 3 2 2 3 2 3 3 1 2 1 1 2 2 2 2 3 2 2 1 1 1 3 3 2 3 2 3 3 1 2 1
[129] 1 1 3 3 3 3 2 1 3 3 3 1 2 3 3 2 2 1 2 3 1 3 1 2 2 3 3 3 1 3 3 1
[161] 2 3 2 3 2 2 2 3 1 3 2 2 3 3 3 3 2 2 2 3 3 3 3 2 1 2 3 3 2 2 3 2
[193] 1 3 3 3 1 1 3 3 3 2 2 3 1 2 1 2 1 3 2 1 1 3 3 1 2 2 1 2 2 3 2 1
[225] 1 2 2 1 2 2 3 2 2 2 2 2 1 3 2 3 2 1 3 2 2 3 2 3 3 2

Within cluster sum of squares by cluster:
[1]  81.5 206.4 181.4
 (between_SS / total_SS =  86.5 %)
```

K-means clustering:
after

```
> tidy(k)
    x1      x2     x3 size withinss cluster
1 3.00 -3.3256 -2.398   47     81.5       1
2 2.03  0.0477  0.901  103    206.4       2
3 1.00  4.8963 -1.188  100    181.4       3
> augment(k, kdat)
  oracle     x1    x2 .cluster
1      2  0.345 1.512        2
2      1  5.784 0.246        3
3      2 -0.291 1.378        2
4      2 -0.922 0.503        2
5      2 -0.456 0.860        2
6      2 -0.897 1.247        2
...
> glance(k)
  totss tot.withinss betweenss iter
1  3484          469      3015    2
```

each row is one cluster

each row is one assignment

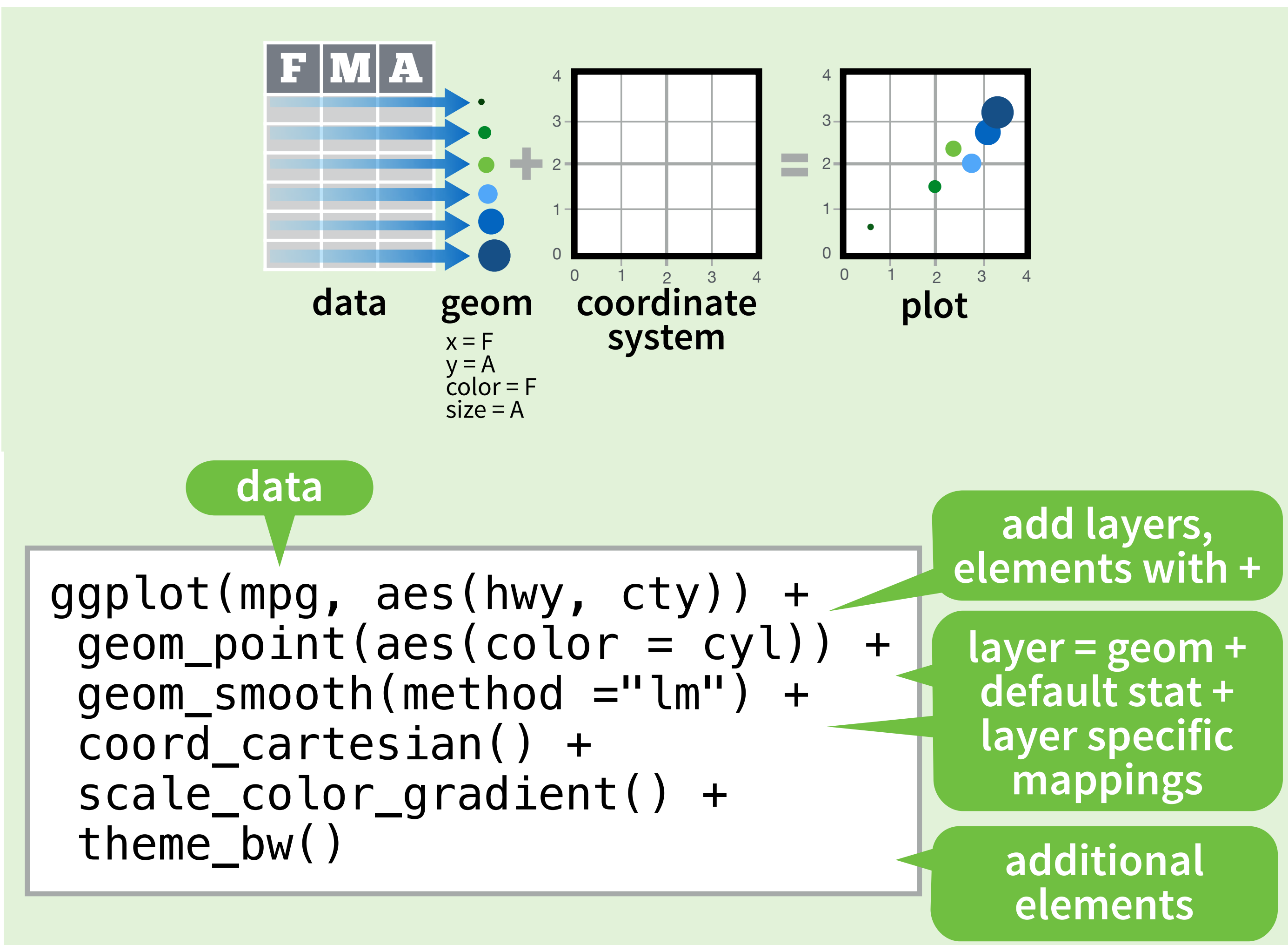
one row describing the entire clustering operation

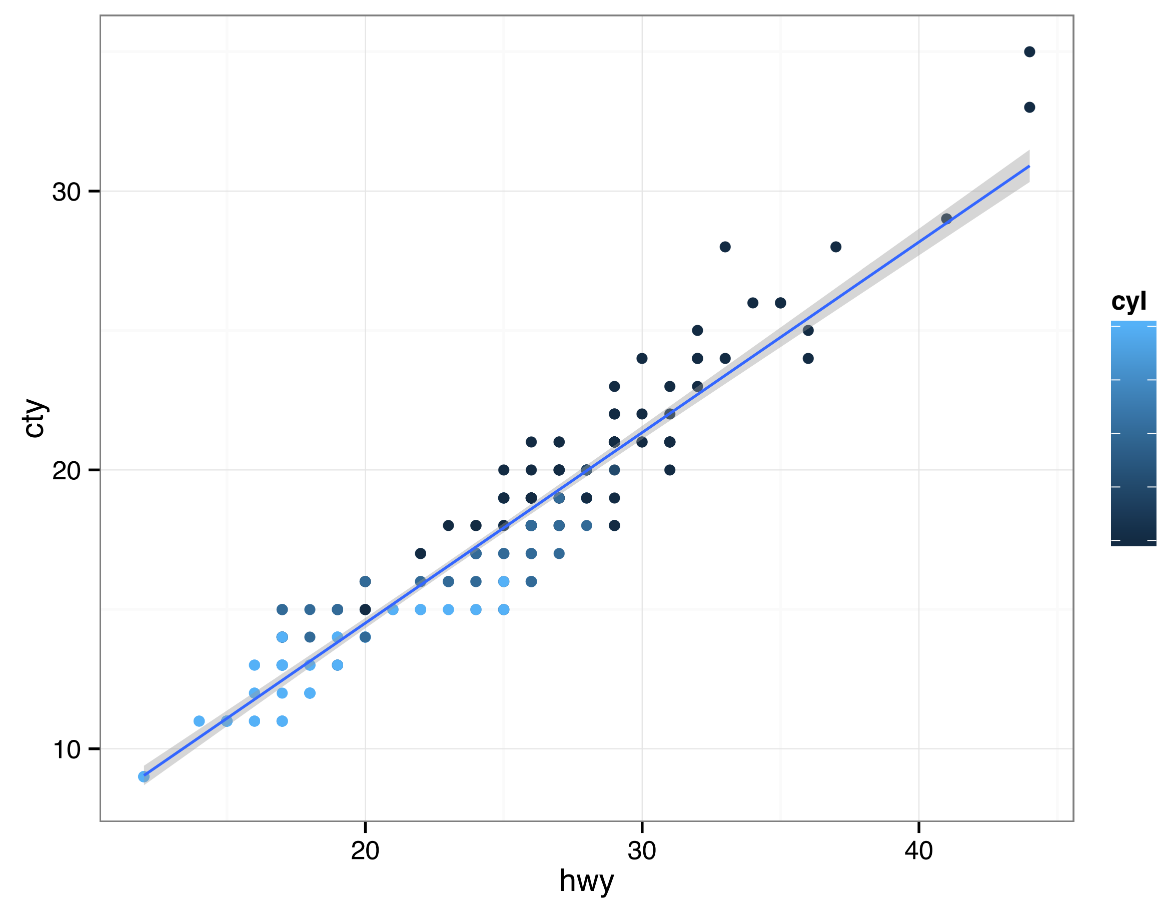
And many others…

package	class	tidy	augment	glance
base	`data.frame`, `matrix`	✓		✓
	table	✓		
stats	anova, aov, aovlist, density, ftable, manova, pairwise.htest, power.htest, spec, ts, TukeyHSD	✓		
	kmeans, lm, glm, nls	✓	✓	✓
	smooth.spline		✓	✓
	Arima, htest, summaryDefault	✓		✓
biglm	biglm, bigglm	✓		✓
glmnet	cv.glmnet, glmnet	✓		✓
lfe	felm	✓	✓	✓
lmtest	coeftest	✓	✓	✓
lme4	mer, merMod	✓	✓	✓
maps	map	✓		
MASS	ridgelm	✓		✓
multcomp	cld, confint.glht, glht, summary.glht	✓		
plm	plm	✓	✓	✓
sp	Line, Lines, Polygon, Polygons, SpatialLinesDataFrame, SpatialPolygons, SpatialPolygonsDataFrame	✓		
survival	aareg, cch, pyears, survexp, survfit	✓		✓
	coxph, survreg	✓	✓	✓
zoo	zoo	✓		

Why are tidy models useful?

ggplot2 can visualize tidy data

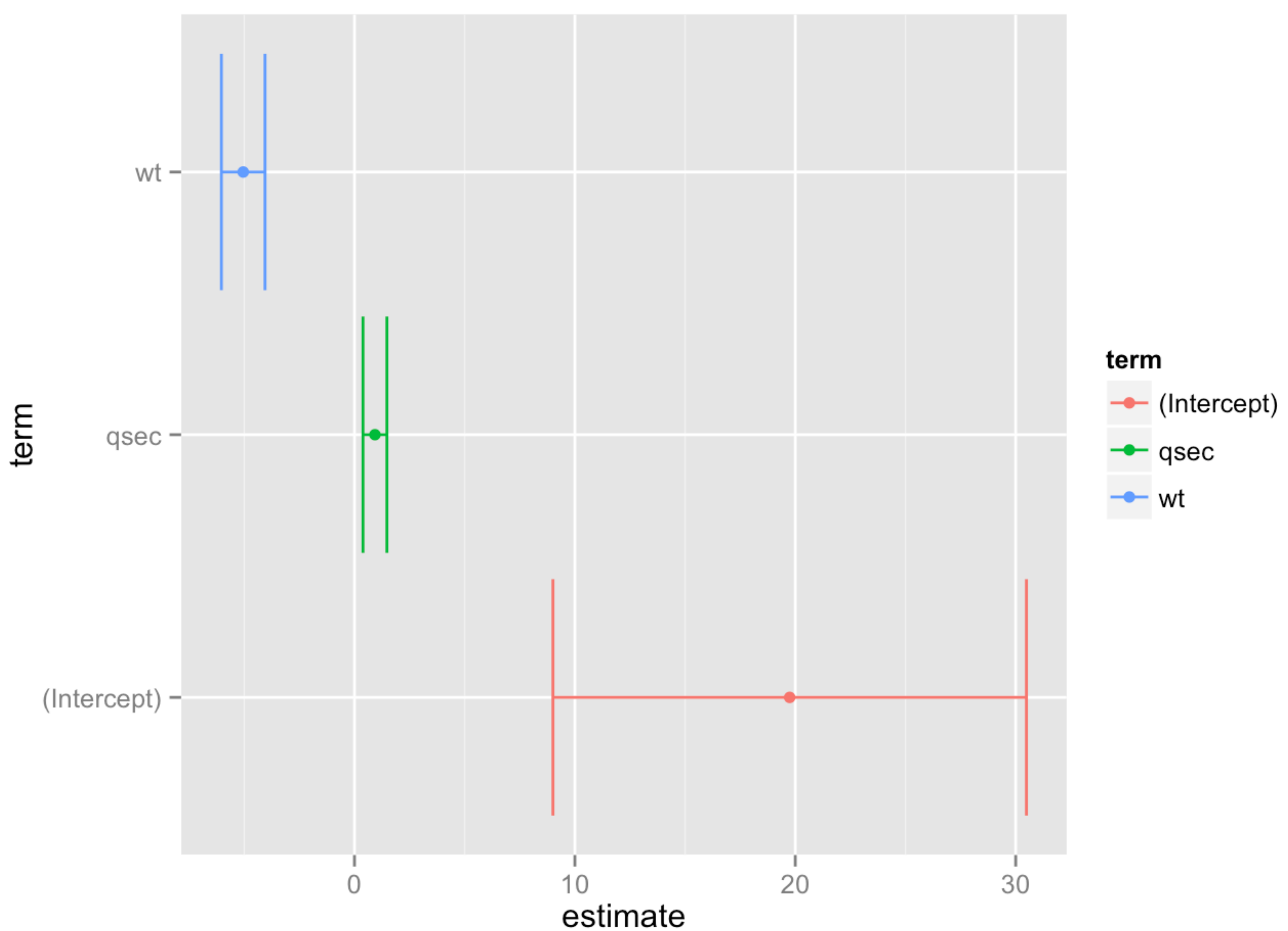




<u>Source</u>: RStudio Data Visualization Cheatsheet

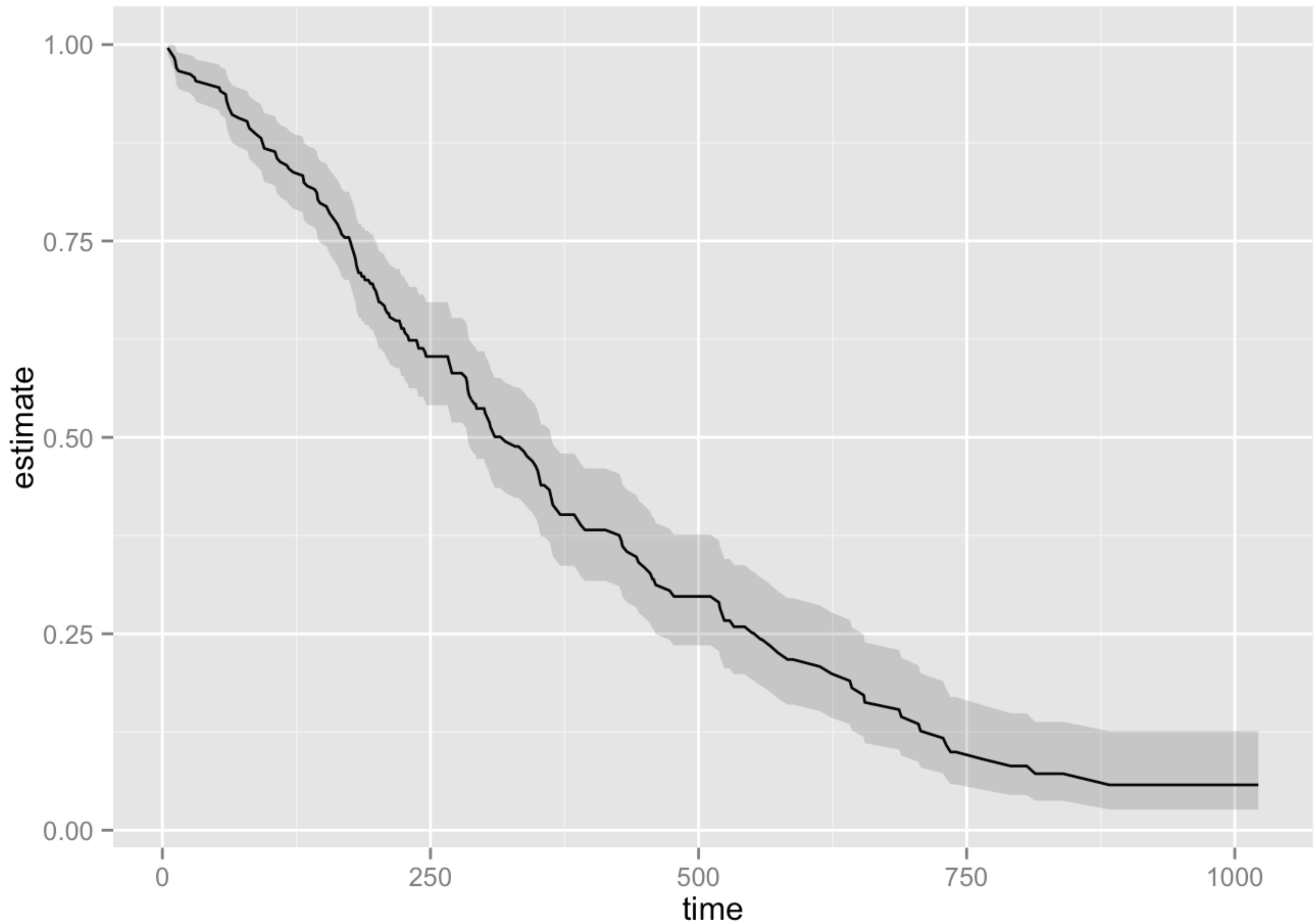
Example: coefficient plot

```
td <- tidy(lmfit, conf.int = TRUE)
ggplot(td, aes(estimate, term, color = term)) +
    geom_point() +
    geom_errorbarh(aes(xmin = conf.low, xmax = conf.high))
```



Example: survival curves

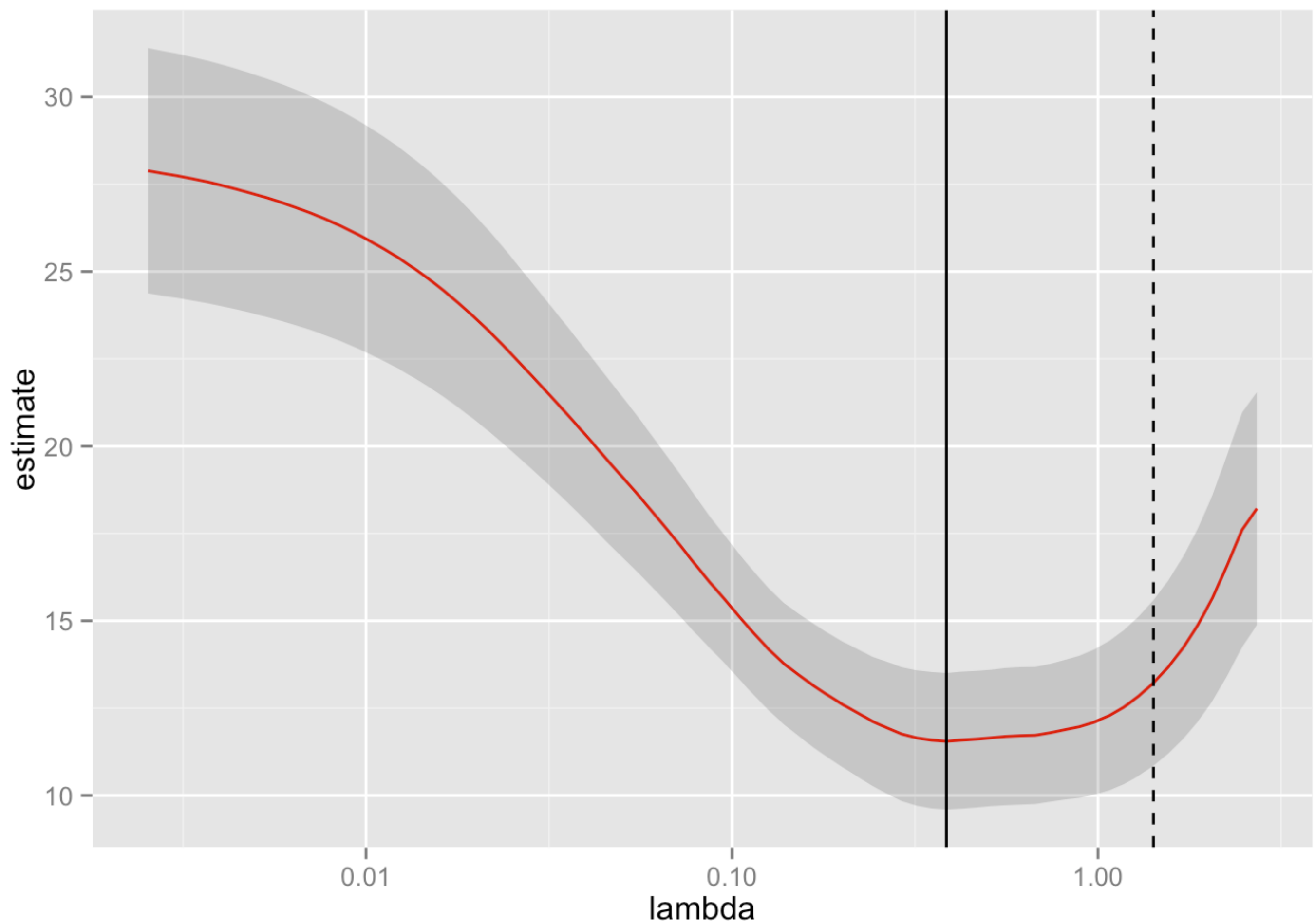
```
library(survival)
surv_fit <- survfit(coxph(Surv(time, status) ~ age + sex, lung))
td <- tidy(surv_fit)
ggplot(td, aes(time, estimate)) + geom_line() +
    geom_ribbon(aes(ymin = conf.low, ymax = conf.high), alpha = .2)
```



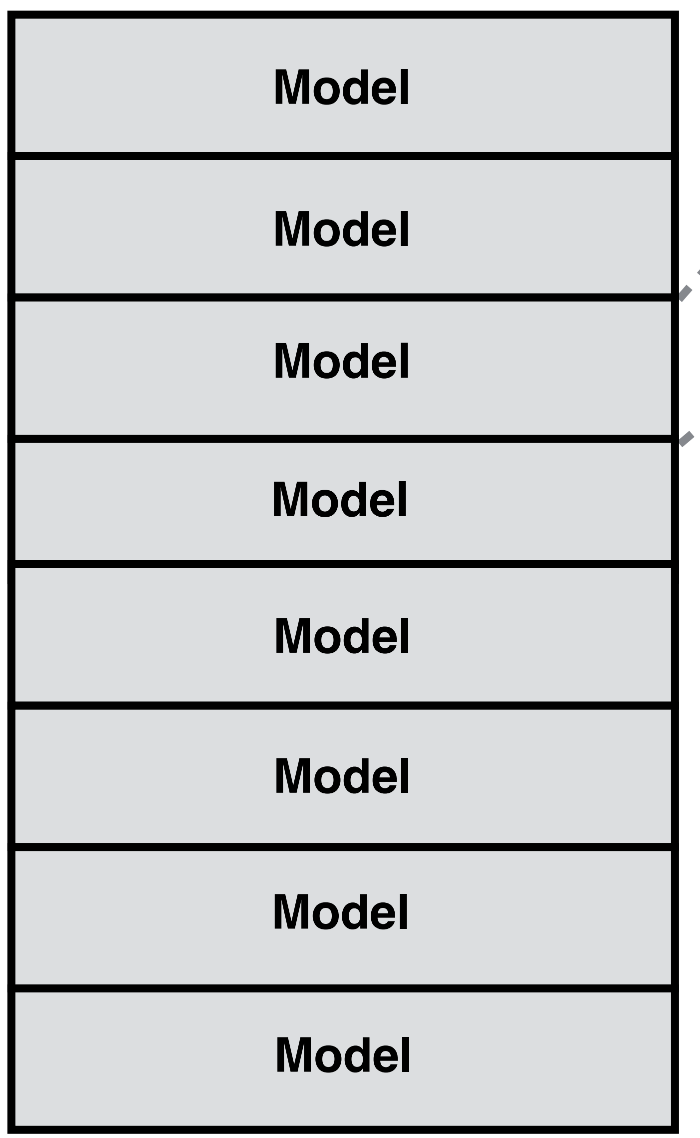
Example: LASSO regression

```
tidied_cv <- tidy(glmnet_fit)
glance_cv <- glance(glmnet_fit)

ggplot(tidied_cv, aes(lambda, estimate)) + geom_line(color = "red") +
    geom_ribbon(aes(ymin = conf.low, ymax = conf.high), alpha = .2) +
    scale_x_log10() +
    geom_vline(xintercept = glance_cv$lambda.min) +
    geom_vline(xintercept = glance_cv$lambda.1se, lty = 2)
```



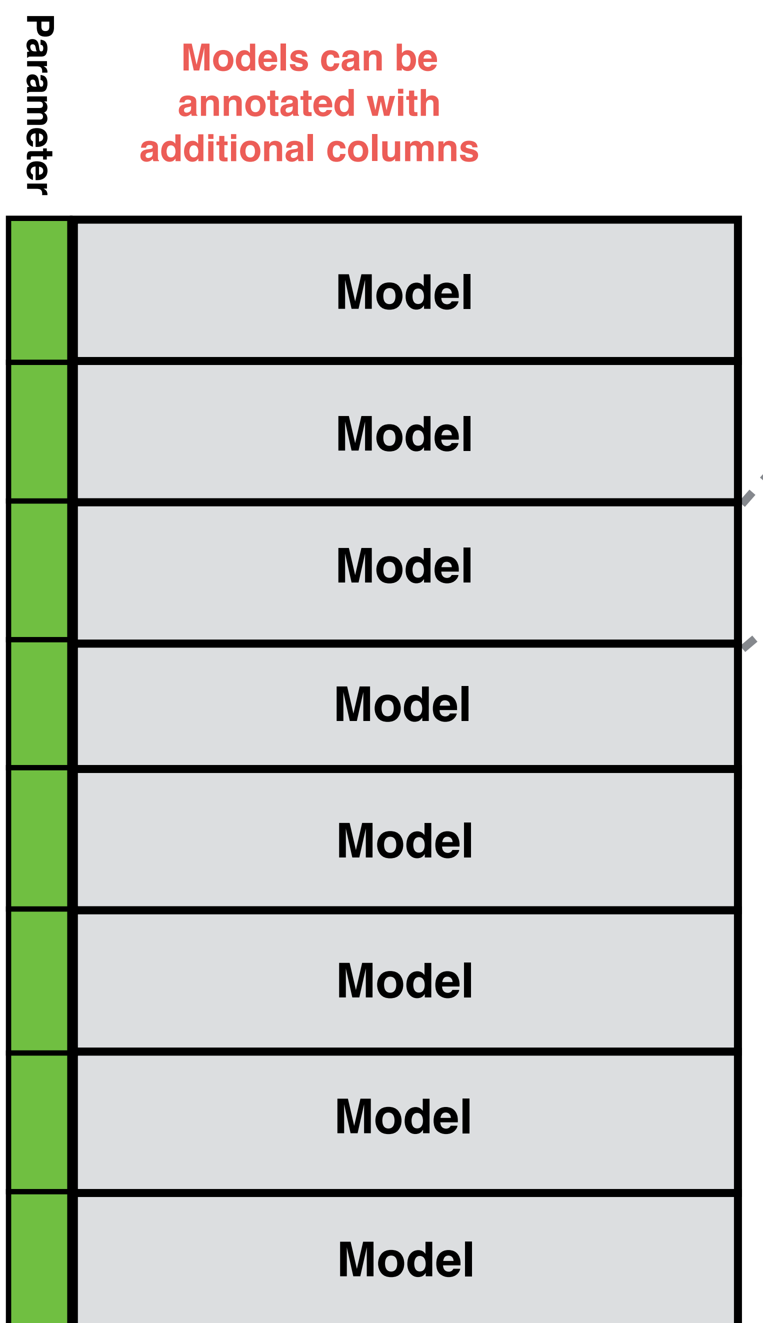
Tidy models can be combined and compared




```
         term estimate std.error statistic  p.value
1 (Intercept)   19.746     5.252      3.76 7.65e-04
2          wt   -5.048     0.484    -10.43 2.52e-11
3        qsec    0.929     0.265      3.51 1.50e-03
```

- different parameters
- different methods
- bootstrap replicates
- subgroup models (within each country, gene…)
- ensemble voting

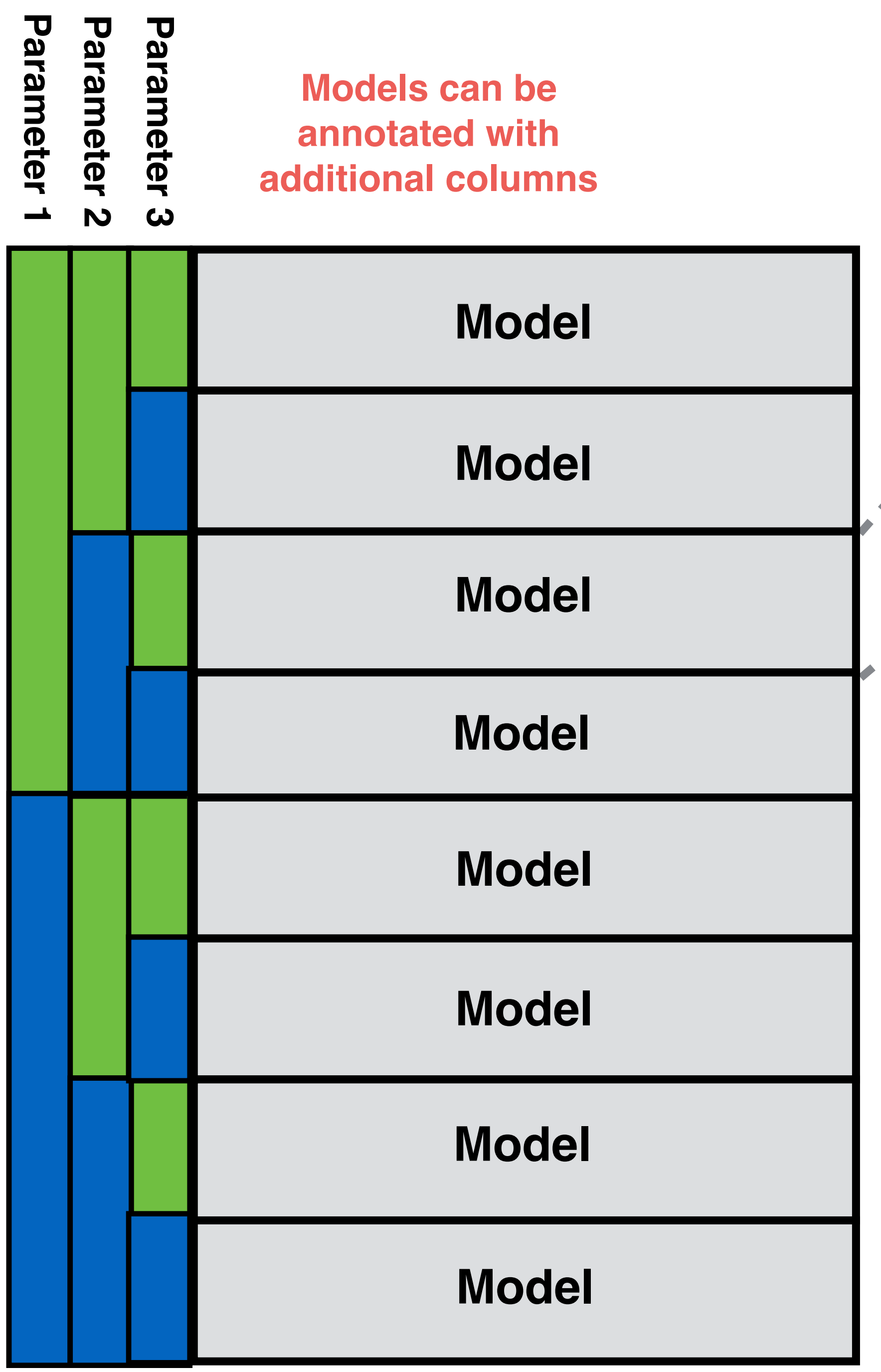
Tidy models can be combined and compared



	term	estimate	std.error	statistic	p.value
1	(Intercept)	19.746	5.252	3.76	7.65e-04
2	wt	-5.048	0.484	-10.43	2.52e-11
3	qsec	0.929	0.265	3.51	1.50e-03

- different parameters
- different methods
- bootstrap replicates
- subgroup models (within each country, gene…)
- ensemble voting

Tidy models can be combined and compared

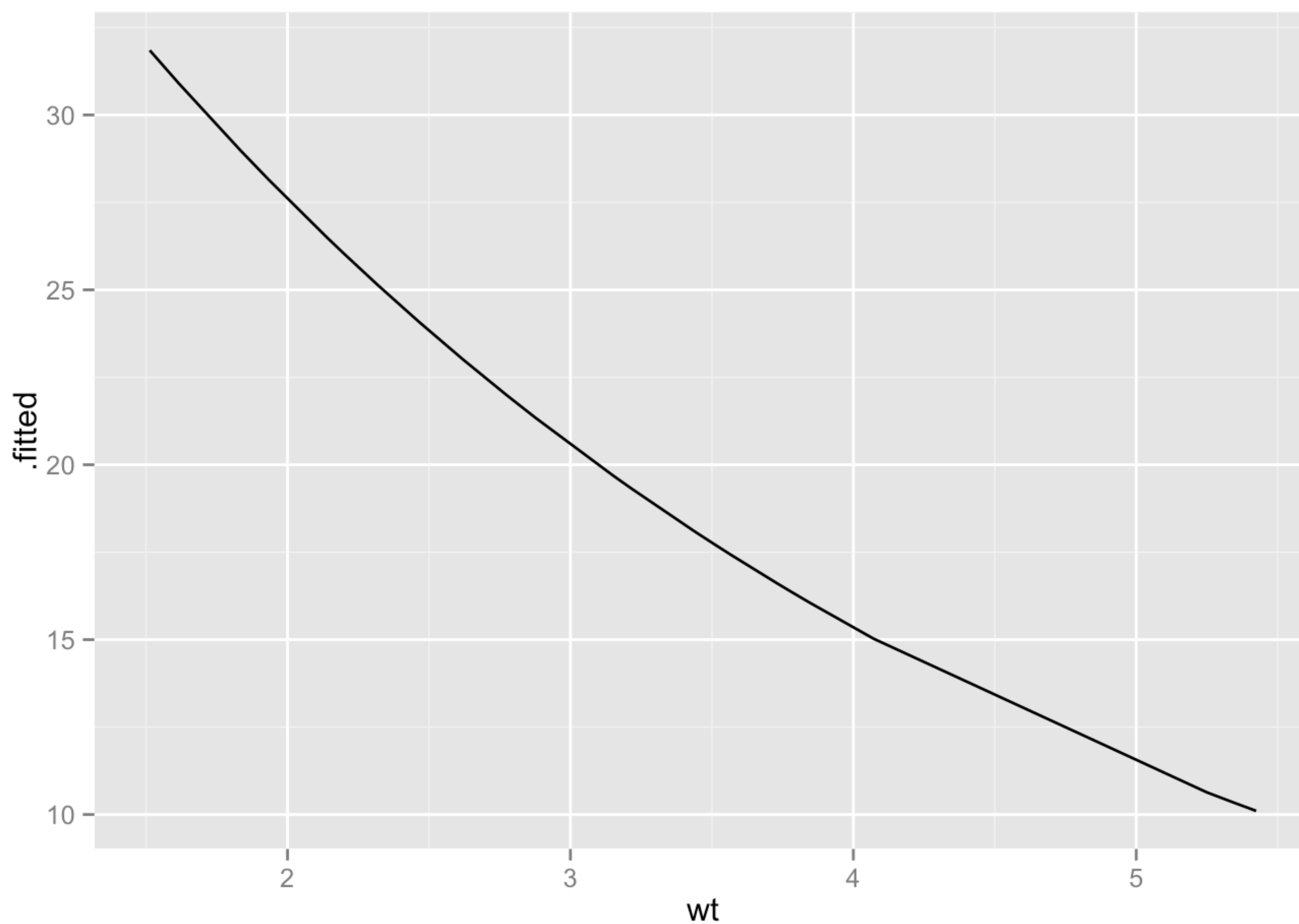



```
         term estimate std.error statistic  p.value
1 (Intercept)   19.746     5.252      3.76 7.65e-04
2          wt   -5.048     0.484    -10.43 2.52e-11
3        qsec    0.929     0.265      3.51 1.50e-03
```

- different parameters
- different methods
- bootstrap replicates
- subgroup models (within each country, gene…)
- ensemble voting

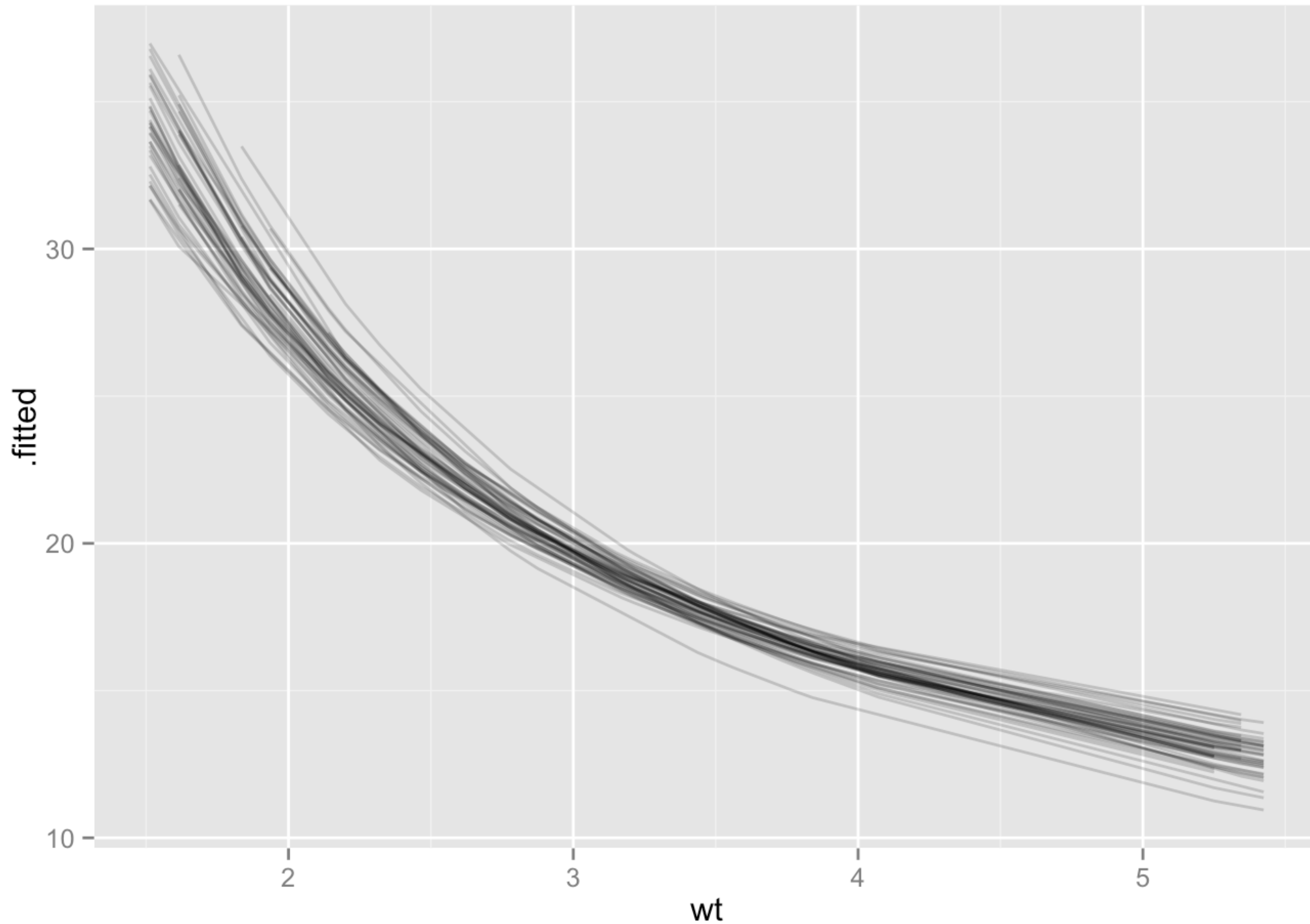
If you can plot one nonlinear least squares fit…

```
augmented <- augment(nlsfit)
ggplot(augmented, aes(wt, .fitted)) + geom_line()
```



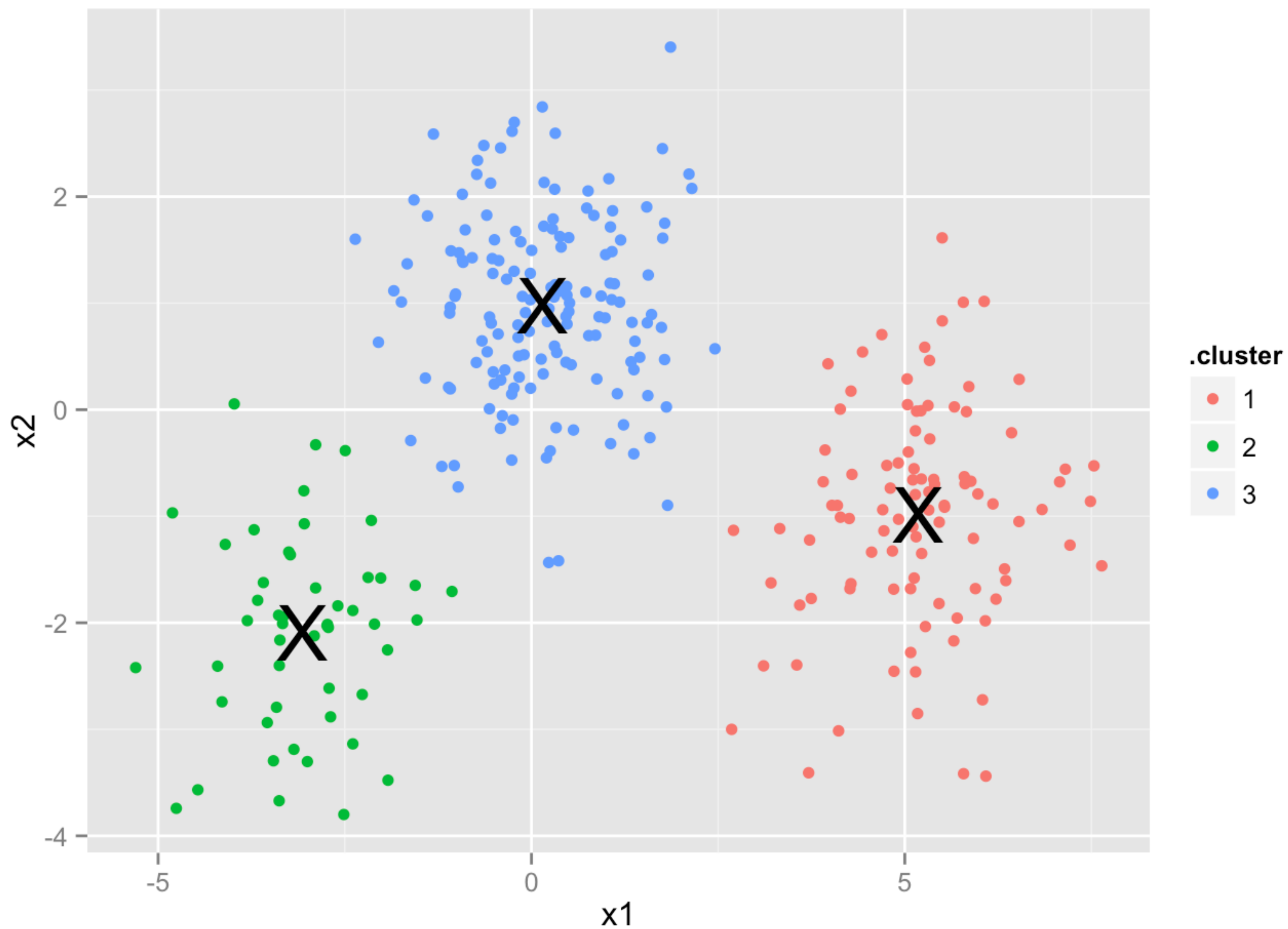
…you can plot 50 bootstrap replicates of it

```
ggplot(combined_augmented, aes(wt, .fitted, group = replicate)) +
    geom_line(alpha = .2)
```



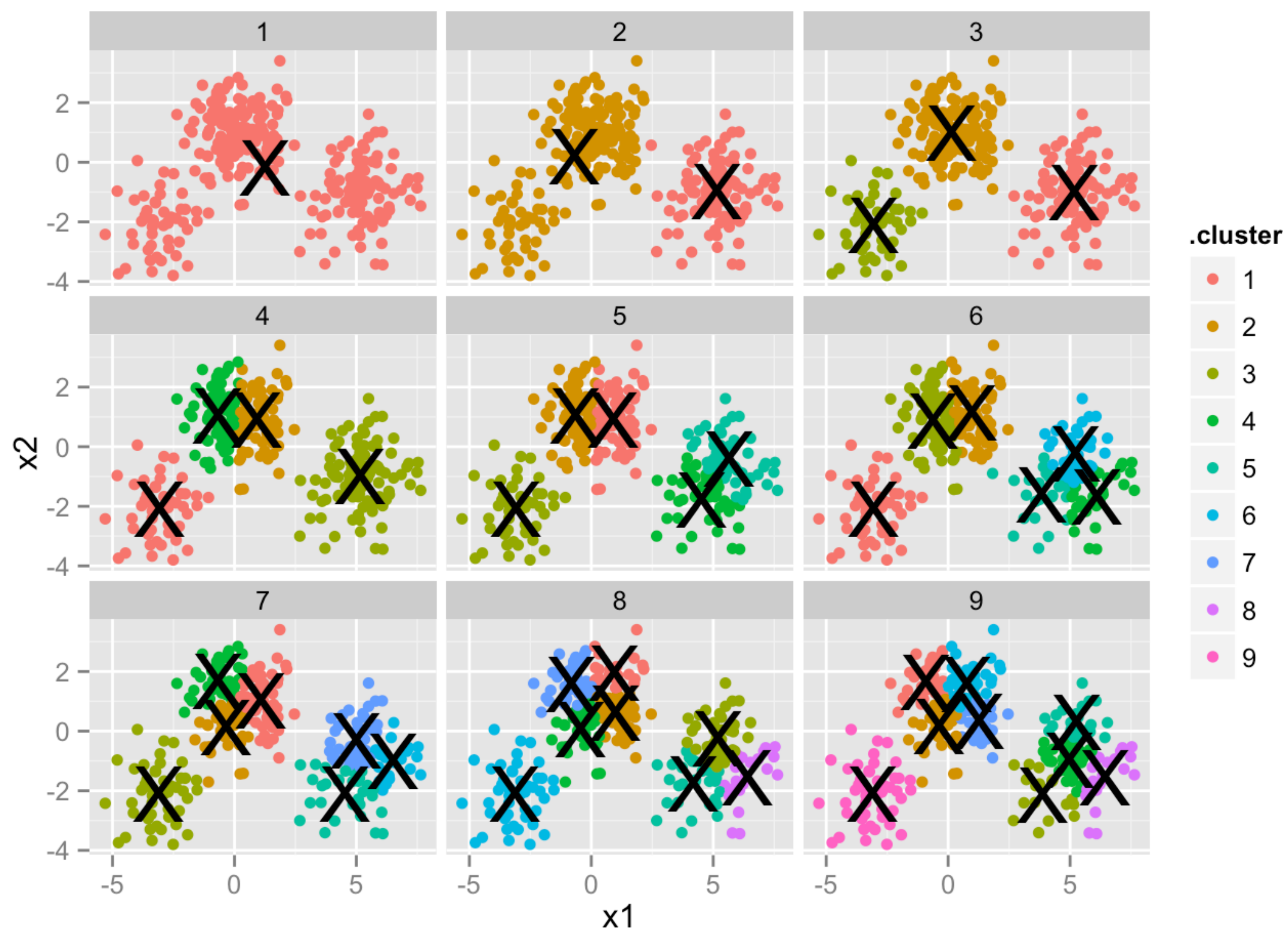
If you can plot one instance of k-means clustering…

```r
ggplot(assignments, aes(x1, x2)) +
    geom_point(aes(color = .cluster)) +
    geom_point(data = clusters, size = 10, shape = "x")
```



…you can plot it for many values of **k**

```
ggplot(combined_assignments, aes(x1, x2)) +
    geom_point(aes(color = .cluster)) +
    geom_point(data = combined_clusters, size = 10, shape = "X") +
    facet_wrap(~ k)
```



Learn more: vignettes

Introduction to broom
broom and dplyr
kmeans with dplyr+broom
Tidy bootstrapping with dplyr+broom

Learn more: manuscript

http://arxiv.org/pdf/1412.3565v2.pdf

broom: An R Package for Converting Statistical Analysis Objects Into Tidy Data Frames

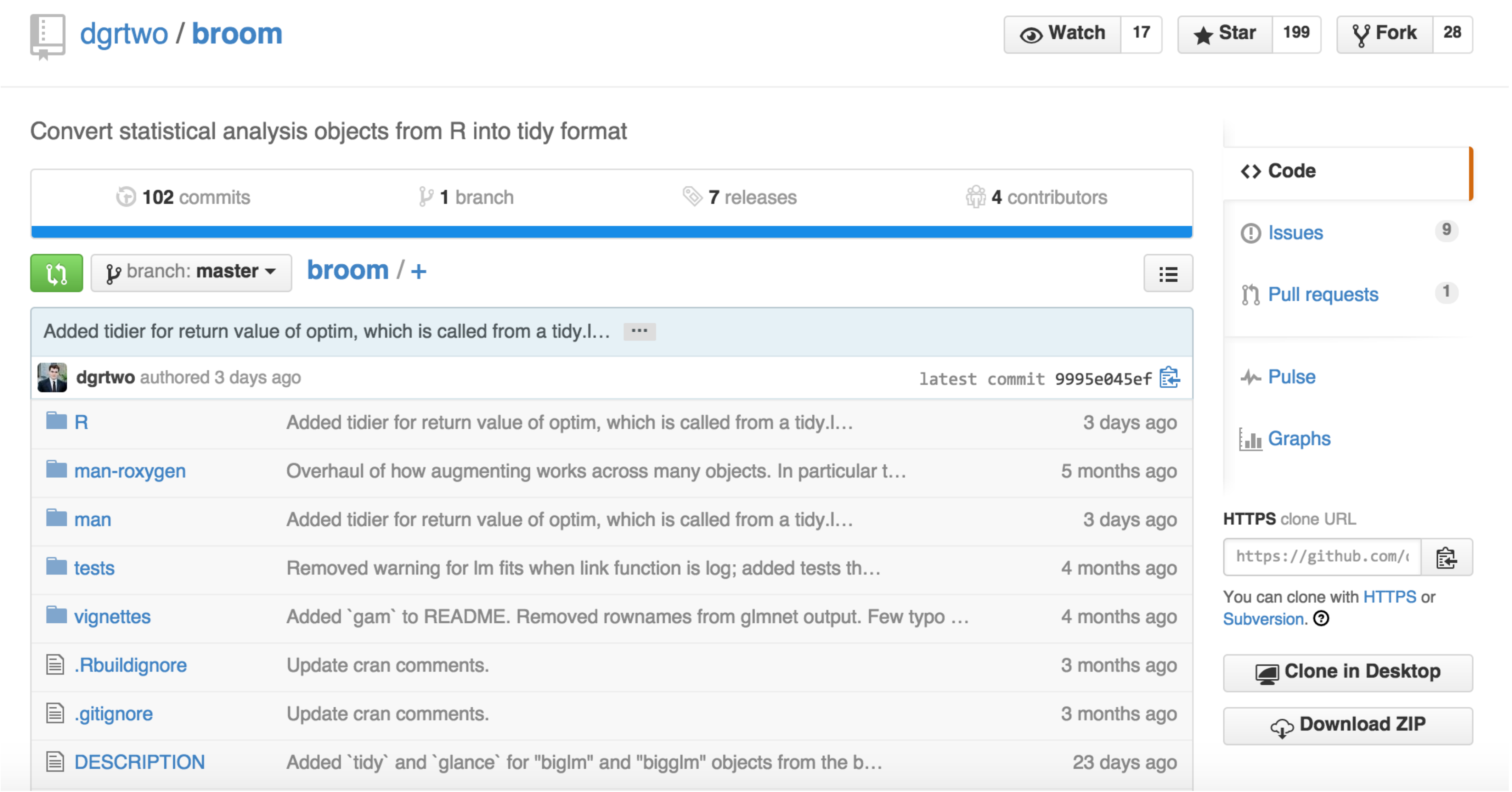
David Robinson

Abstract

The concept of "tidy data" offers a powerful framework for structuring data to ease manipulation, modeling and visualization. However, most R functions, both those built-in and those found in third-party packages, produce output that is not tidy, and that is therefore difficult to reshape, recombine, and otherwise manipulate. Here I introduce the **broom** package, which turns the output of model objects into tidy data frames that are suited to further analysis, manipulation, and visualization with input-tidy tools. **broom** defines the `tidy`, `augment`, and `glance` generics, which arrange a model into three levels of tidy output respectively: the component level, the observation level, and the model level. I provide examples to demonstrate how these generics work with tidy tools to allow analysis and modeling of data that is divided into subsets, to recombine results from bootstrap replicates, and to perform simulations that investigate the effect of varying input parameters.

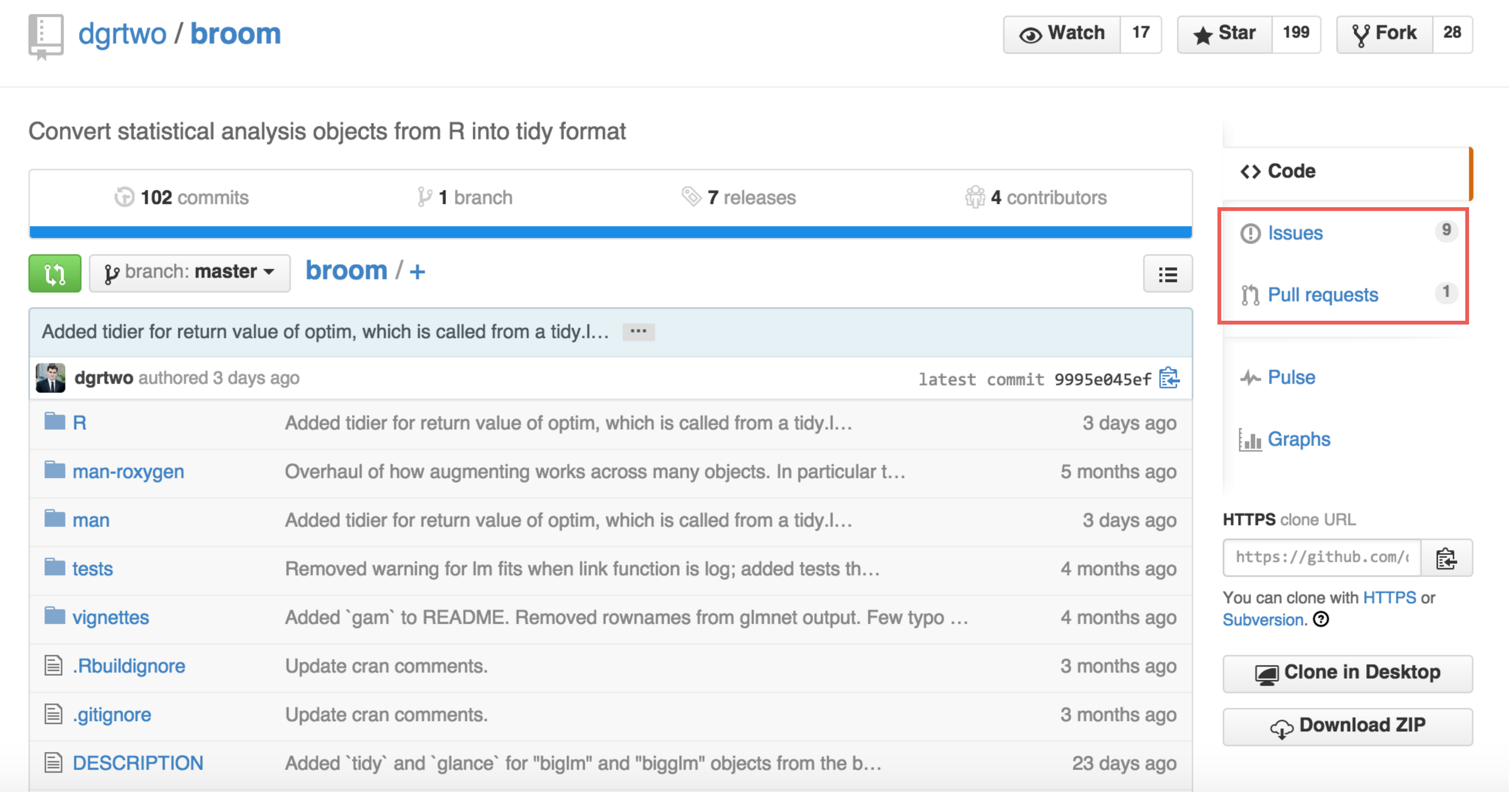
Learn more: GitHub

https://github.com/dgrtwo/broom



Contribute!

https://github.com/dgrtwo/broom/issues



Thank you!

- broom package
  - Matthieu Gomez
  - Boris Demeshev
  - Hadley Wickham
- Presentation
  - Dima Gorenshteyn
  - Storey Lab at Princeton University
  - UP-STAT 2015