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?

```r
> 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?

```r
> 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)    19.74623    5.25211   3.760 0.000765 ***
w t            -5.04800    0.48400 -10.430 2.52e-11 ***
qsec           0.92922    0.26500   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:

```r
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
What’s “messy” about a linear regression?

```r
> 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 **
```

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

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 ***
w              -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

```r
> 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

```r
> 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
```

Information stored in columns, never row names

One function to call

Convenient column names
broom takes model objects and turns them into tidy data frames that can be used with tidy tools
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
The `tidy()` method: component-level statistics

```r
> 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

```r
> 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  -0.0482  0.0444  2.64
5 Hornet Sportabout  18.7  3.44  17.0   18.20  0.512  -2.5273  0.0607  2.64
  Valiant  18.1  3.46  20.2   21.07  0.803  -2.9686  0.0957  2.58
  Duster 360  14.3  3.57  15.8   16.44  0.701  -2.1434  0.0729  2.61
  Merc 240D  24.4  3.19  20.0   22.23  0.730   2.1729  0.0791  2.61
  Merc 230  22.8  3.15  22.9   25.12  1.410  -2.3237  0.2950  2.59
  Merc 280  19.2  3.44  18.3   19.39  0.491  -0.1855  0.0358  2.64
  Merc 280C  17.8  3.44  18.9   19.94  0.557  -2.1430  0.0460  2.61
  Merc 450SE  16.4  4.07  17.4   15.37  0.615   1.0310  0.0561  2.63
  Merc 450SL  17.3  3.73  17.6   17.27  0.520   0.0289  0.0402  2.64
  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

```r
> 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  -0.0482  0.0444  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
```

Note that each row is an observation from the original data.
The `glance()` method: model-level statistics

```r
> 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

```r
> 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)`

<table>
<thead>
<tr>
<th>term</th>
<th>estimate</th>
<th>std.error</th>
<th>statistic</th>
<th>p.value</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>49.660</td>
<td>3.7888</td>
<td>13.1</td>
<td>5.96e-14</td>
</tr>
<tr>
<td>e</td>
<td>0.746</td>
<td>0.0199</td>
<td>37.5</td>
<td>8.86e-27</td>
</tr>
</tbody>
</table>

`> augment(n)`

<table>
<thead>
<tr>
<th>mpg</th>
<th>wt</th>
<th>fitted</th>
<th>resid</th>
</tr>
</thead>
<tbody>
<tr>
<td>21.0</td>
<td>2.62</td>
<td>23.0</td>
<td>-2.012</td>
</tr>
<tr>
<td>21.0</td>
<td>2.88</td>
<td>21.4</td>
<td>-0.352</td>
</tr>
<tr>
<td>22.8</td>
<td>2.32</td>
<td>25.1</td>
<td>-2.331</td>
</tr>
<tr>
<td>21.4</td>
<td>3.21</td>
<td>19.3</td>
<td>2.076</td>
</tr>
<tr>
<td>18.7</td>
<td>3.44</td>
<td>18.1</td>
<td>0.611</td>
</tr>
<tr>
<td>18.1</td>
<td>3.46</td>
<td>18.0</td>
<td>0.117</td>
</tr>
</tbody>
</table>

`> glance(n)`

<table>
<thead>
<tr>
<th>sigma</th>
<th>isConv</th>
<th>finTol</th>
<th>logLik</th>
<th>AIC</th>
<th>BIC</th>
<th>deviance</th>
<th>df.residual</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.67</td>
<td>TRUE</td>
<td>2.04e-06</td>
<td>-75.8</td>
<td>158</td>
<td>162</td>
<td>214</td>
<td>30</td>
</tr>
</tbody>
</table>
K-means clustering:

before

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

Cluster means:

<table>
<thead>
<tr>
<th></th>
<th>x1</th>
<th>x2</th>
</tr>
</thead>
<tbody>
<tr>
<td>oracle</td>
<td>3.00</td>
<td>-3.3256</td>
</tr>
<tr>
<td>1</td>
<td>2.03</td>
<td>0.0477</td>
</tr>
<tr>
<td>2</td>
<td>1.00</td>
<td>4.8963</td>
</tr>
<tr>
<td>3</td>
<td>3.00</td>
<td>-1.188</td>
</tr>
</tbody>
</table>

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 2 1 2
[33] 2 3 2 2 3 3 1 2 1 2 1 3 2 3 2 3 2 3 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 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 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 3 3 2 1 2 3 3 3 2 2 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 3 2 3 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

```r
> tidy(k)

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>x2</td>
<td>x3</td>
<td>size</td>
<td>withinss</td>
<td>cluster</td>
</tr>
<tr>
<td>1</td>
<td>3.00</td>
<td>-3.3256</td>
<td>-2.398</td>
<td>47</td>
<td>81.5</td>
</tr>
<tr>
<td>2</td>
<td>2.03</td>
<td>0.0477</td>
<td>0.901</td>
<td>103</td>
<td>206.4</td>
</tr>
<tr>
<td>3</td>
<td>1.00</td>
<td>4.8963</td>
<td>-1.188</td>
<td>100</td>
<td>181.4</td>
</tr>
</tbody>
</table>

> augment(k, kdat)

<table>
<thead>
<tr>
<th></th>
<th>oracle</th>
<th>x1</th>
<th>x2</th>
<th>.cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.345</td>
<td>1.512</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>5.784</td>
<td>0.246</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>-0.291</td>
<td>1.378</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>-0.922</td>
<td>0.503</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>-0.456</td>
<td>0.860</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>-0.897</td>
<td>1.247</td>
<td>2</td>
</tr>
</tbody>
</table>

> glance(k)

<table>
<thead>
<tr>
<th></th>
<th>totss</th>
<th>tot.withinss</th>
<th>betweenss</th>
<th>iter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3484</td>
<td>469</td>
<td>3015</td>
<td>2</td>
</tr>
</tbody>
</table>
```
And many others…

<table>
<thead>
<tr>
<th>package</th>
<th>class</th>
<th>tidy</th>
<th>augment</th>
<th>glance</th>
</tr>
</thead>
<tbody>
<tr>
<td>base</td>
<td>data.frame, matrix</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>table</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>stats</td>
<td>anova, aov, aovlist, density, ftable, manova, pairwise.htest, power.htest, spec, ts, TukeyHSD</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>kmeans, lm, glm, nls</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>smooth.spline</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Arima, htest, summaryDefault</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>biglm</td>
<td>biglm, bigglm</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>glmnet</td>
<td>cv.glmnet, glmnet</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>lfe</td>
<td>felm</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>lmtest</td>
<td>coeftest</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>lme4</td>
<td>mer, merMod</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>maps</td>
<td>map</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>MASS</td>
<td>ridgelm</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>multcomp</td>
<td>cld, confint.glht, glht, summary.glht</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>plm</td>
<td>plm</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>sp</td>
<td>Line, Lines, Polygon, Polygons, SpatialLinesDataFrame, SpatialPolygons, SpatialPolygonsDataFrame</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>survival</td>
<td>aareg, cch, pyears, survexp, survfit</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>coxph, survreg</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>zoo</td>
<td>zoo</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
Why are tidy models useful?
ggplot2 can visualize tidy data

```
ggplot(mpg, aes(hwy, cty)) +
  geom_point(aes(color = cyl)) +
  geom_smooth(method = "lm") +
  coord_cartesian() +
  scale_color_gradient() +
  theme_bw()
```

**Source:** 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

```r
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

```r
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

- different parameters
- different methods
- bootstrap replicates
- subgroup models (within each country, gene…)
- ensemble voting
Tidy models can be combined and compared

Models can be annotated with additional columns

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

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameter</th>
<th>term</th>
<th>estimate</th>
<th>std.error</th>
<th>statistic</th>
<th>p.value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(Intercept)</td>
<td>19.746</td>
<td>5.252</td>
<td>3.76</td>
<td>7.65e-04</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>wt</td>
<td>-5.048</td>
<td>0.484</td>
<td>-10.43</td>
<td>2.52e-11</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>qsec</td>
<td>0.929</td>
<td>0.265</td>
<td>3.51</td>
<td>1.50e-03</td>
<td></td>
</tr>
</tbody>
</table>
Tidy models can be combined and compared

- different parameters
- different methods
- bootstrap replicates
- subgroup models (within each country, gene...)
- ensemble voting
If you can plot one nonlinear least squares fit...

```r
augmented <- augment(nlsfit)
ggplot(augmented, aes(wt, .fitted)) + geom_line()
```
...you can plot 50 bootstrap replicates of it

```r
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

```r
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
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/dguttwo/broom
Contribute!

https://github.com/dgprtwo/broom/issues
Thank you!

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