Broom: Converting Statistical Models to Tidy Data Frames

David Robinson
6/28/2016
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

tidyyr

dplyr

ggplot2

data.table

pandas

y_t = β'x_{it} + μ_i + ε_{it}

Source: RStudio: Data Wrangling Cheatsheet
RStudio: Data Visualization Cheatsheet
DataCamp: Data Analysis The data.table Way (DataCamp)
http://pandas.pydata.org/
Tidy tools work together in exploratory data analysis.

Messy Data → data tidying (tidyr) → Tidy Data → data manipulation (dplyr) → data visualization (ggplot2) → Graphs
Everything works well until…
Visualizing and manipulating model objects is difficult

Messy Data → Tidy Data
- data tidying (tidyr)
- data visualization (ggplot2)

Tidy Data → Models
- modeling (stats)
- data manipulation (dplyr)

Models → Model Graphs
- model visualization (???)
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?

```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 **
---
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?

```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 **

---

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 t            -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 ***
w t                      -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 ***
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

1. Extracting coefficients takes multiple steps:
   data.frame(coef(summary(lmfit)))

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

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

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

One function to call

```r
> tidy(lmfit)

<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>(Intercept)</td>
<td>19.746</td>
<td>5.252</td>
<td>3.76</td>
<td>7.65e-04</td>
</tr>
<tr>
<td>wt</td>
<td>-5.048</td>
<td>0.484</td>
<td>-10.43</td>
<td>2.52e-11</td>
</tr>
<tr>
<td>qsec</td>
<td>0.929</td>
<td>0.265</td>
<td>3.51</td>
<td>1.50e-03</td>
</tr>
</tbody>
</table>
```

Information stored in columns, never row names

Convenient column names
broom takes model objects and turns them into tidy data frames that can be used with tidy tools
Messy Data → Tidy Data (tidyr) → Models (stats) → Tidy Models (broom)

- Data tidying: `tidyr`
- Modeling: `stats`
- Data manipulation: `dplyr`
- Model tidying: `broom`
- Data visualization: `ggplot2`
- Model visualization: `ggplot2`

Graphs
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

```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)            19.74628   5.25212   3.760  0.000765 ***
w t                   -5.04801   0.48401 -10.430  2.52e-11 ***
q sec              0.929237   0.26502   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:
- R², F-statistic, deviance
  - glance()
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

> augment(lmfit)

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

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

```r
> 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.

```r
> 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.

```r
> 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

> m <- as.matrix(iris[, -5])
> k <- kmeans(m, 3)
> k

K-means clustering with 3 clusters of sizes 62, 50, 38

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.901613    2.748387     4.393548    1.433871
2     5.006000    3.428000     1.462000    0.246000
3     6.850000    3.073684     5.742105    2.071053

Clustering vector:
[1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[65] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[129] 3 3 3 3 3 1 3 3 3 3 1 3 3 3 1 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

Within cluster sum of squares by cluster:
(between_SS / total_SS =  88.4 %)

Available components:
[1] "cluster" "centers" "totss" "withinss" "tot.withinss" "betweenss" "size" "iter"
[9] "ifault"
K-means clustering:

after

```r
> tidy(k)
   x1       x2       x3       x4 size withinss cluster
1 5.901613 2.748387 4.393548 1.433871   62 39.82097       1
2 5.006000 3.428000 1.462000 0.246000   50 15.15100       2
3 6.850000 3.073684 5.742105 2.071053   38 23.87947       3
> head(augment(k, m))
1          5.1         3.5          1.4         0.2        2
2          4.9         3.0          1.4         0.2        2
3          4.7         3.2          1.3         0.2        2
4          4.6         3.1          1.5         0.2        2
5          5.0         3.6          1.4         0.2        2
6          5.4         3.9          1.7         0.4        2
> glance(k)
  totss tot.withinss betweenss iter
1 681.3706     78.85144  602.5192    2
```

- each row is one cluster
- each row is one assignment
- one row describing the entire clustering operation
And many others...
Why are tidy models useful?
ggplot2 can visualize tidy data

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
teachlibrary(survival)
surv_fit <- survfit(coxph(Serv(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.lse, lty = 2)
Multiple Models
Tidy models can be combined and compared

<table>
<thead>
<tr>
<th>Model</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>
</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>
</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>
</tr>
</tbody>
</table>

- 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

<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>

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

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

### Models can be annotated with additional columns

<table>
<thead>
<tr>
<th>Parameter 1</th>
<th>Parameter 2</th>
<th>Parameter 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Model</td>
<td>Model</td>
</tr>
<tr>
<td>Model</td>
<td>Model</td>
<td>Model</td>
</tr>
<tr>
<td>Model</td>
<td>Model</td>
<td>Model</td>
</tr>
<tr>
<td>Model</td>
<td>Model</td>
<td>Model</td>
</tr>
<tr>
<td>Model</td>
<td>Model</td>
<td>Model</td>
</tr>
<tr>
<td>Model</td>
<td>Model</td>
<td>Model</td>
</tr>
<tr>
<td>Model</td>
<td>Model</td>
<td>Model</td>
</tr>
</tbody>
</table>

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></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>
</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>
</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>
</tr>
</tbody>
</table>
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

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: many models

http://r4ds.had.co.nz/
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.
Contribute: GitHub

Convert statistical analysis objects from R into tidy format — Edit

- 332 commits
- 2 branches
- 9 releases
- 25 contributors

- R: Moved act tidiers to stats_tidiers.
- inst/extdata: Various edits to MCMC tidiers; mostly style changes. Added 8schools.a...
- man-roxygen: Overhaul of how augmenting works across many objects. In particular t...
- man: Moved act tidiers to stats_tidiers.
- tests: Fixed to be compatible with dplyr 0.5
- vignettes: update bootstrap vignette
- .Rbuildignore: Added codecov.io
- .gitignore: Update cran comments.
- .travis.yml: Added codecov.io
- CONDUCT.md: Added Code of Conduct

Latest commit 35b7b463 3 days ago
Thank you!

- broom package/paper
  - Matthieu Gomez
  - Boris Demeshev
  - Dieter Meine
  - Benjamin Nutter
  - Luke Johnston
  - Ben Bolker
  - Francois Briatte
  - Bob Muenchen
  - Hadley Wickham

CC BY David Robinson 2016