### Grouping & Summarizing Data in R

**Greater Boston useR Group**  
March 3, 2011

**by**  
Jeffrey Breen  
jbreen@cambridge.aero

---

<table>
<thead>
<tr>
<th>#</th>
<th>comp</th>
<th>mass</th>
<th>time</th>
<th>rt_index</th>
<th>chg</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>300.130</td>
<td>13,271</td>
<td>218.720</td>
<td>35.200</td>
<td>71.000</td>
<td>38.900</td>
<td>48.500</td>
<td>63.700</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>300.142</td>
<td>16,546</td>
<td>277.850</td>
<td>37.800</td>
<td>22.600</td>
<td>38.900</td>
<td>48.500</td>
<td>63.700</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>300.150</td>
<td>21,879</td>
<td>369.890</td>
<td>26.700</td>
<td>38.900</td>
<td>48.500</td>
<td>63.700</td>
<td>22.600</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>300.156</td>
<td>14,974</td>
<td>250.627</td>
<td>11.400</td>
<td>26.700</td>
<td>26.700</td>
<td>25.700</td>
<td>22.600</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>300.160</td>
<td>23,724</td>
<td>405.368</td>
<td>1390.000</td>
<td>1080.000</td>
<td>463.000</td>
<td>552.000</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>300.170</td>
<td>27,365</td>
<td>459.410</td>
<td>27.700</td>
<td>21.400</td>
<td>21.400</td>
<td>21.400</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>300.174</td>
<td>18,146</td>
<td>303.145</td>
<td>191.000</td>
<td>20.300</td>
<td>16.000</td>
<td>16.000</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>11</td>
<td>300.184</td>
<td>22,930</td>
<td>390.230</td>
<td>21.600</td>
<td>23.800</td>
<td>23.800</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>12</td>
<td>300.189</td>
<td>19,380</td>
<td>322.203</td>
<td>23.500</td>
<td>23.500</td>
<td>23.500</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>13</td>
<td>300.190</td>
<td>20,750</td>
<td>350.840</td>
<td>123.000</td>
<td>114.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>14</td>
<td>300.218</td>
<td>26,257</td>
<td>441.103</td>
<td>472.000</td>
<td>306.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>16</td>
<td>300.480</td>
<td>18,599</td>
<td>309.823</td>
<td>18.500</td>
<td>26.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>18</td>
<td>300.480</td>
<td>14,272</td>
<td>237.073</td>
<td>23.500</td>
<td>39.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>19</td>
<td>300.500</td>
<td>27,109</td>
<td>455.170</td>
<td>40.900</td>
<td>37.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>22</td>
<td>300.640</td>
<td>29,241</td>
<td>490.978</td>
<td>80.200</td>
<td>86.800</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>23</td>
<td>300.649</td>
<td>17,936</td>
<td>300.046</td>
<td>44.500</td>
<td>45.200</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>24</td>
<td>300.665</td>
<td>15,088</td>
<td>253.020</td>
<td>119.000</td>
<td>366.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>26</td>
<td>300.690</td>
<td>14,023</td>
<td>232.220</td>
<td>446.000</td>
<td>399.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>27</td>
<td>300.682</td>
<td>21,130</td>
<td>357.260</td>
<td>43.800</td>
<td>15.400</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>28</td>
<td>300.690</td>
<td>20,134</td>
<td>337.710</td>
<td>27.400</td>
<td>21.200</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>29</td>
<td>300.698</td>
<td>22,291</td>
<td>377.470</td>
<td>1400.000</td>
<td>1290.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>31</td>
<td>300.700</td>
<td>18,390</td>
<td>306.732</td>
<td>182.000</td>
<td>156.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>33</td>
<td>300.850</td>
<td>26,615</td>
<td>447.026</td>
<td>130.000</td>
<td>139.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>36</td>
<td>301.155</td>
<td>19.741</td>
<td>329.060</td>
<td>22.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>37</td>
<td>301.158</td>
<td>36.577</td>
<td>619.438</td>
<td>16.800</td>
<td>38.600</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>38</td>
<td>301.160</td>
<td>18.014</td>
<td>301.190</td>
<td>183.000</td>
<td>202.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>40</td>
<td>301.163</td>
<td>14.308</td>
<td>237.775</td>
<td>75.300</td>
<td>76.600</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>41</td>
<td>301.164</td>
<td>16.116</td>
<td>270.677</td>
<td>24.900</td>
<td>19.700</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>42</td>
<td>301.168</td>
<td>16.362</td>
<td>274.773</td>
<td>24.200</td>
<td>29.500</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>44</td>
<td>301.181</td>
<td>20.429</td>
<td>344.366</td>
<td>54.300</td>
<td>51.600</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>45</td>
<td>301.187</td>
<td>21.619</td>
<td>365.506</td>
<td>104.000</td>
<td>104.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>48</td>
<td>301.194</td>
<td>19.293</td>
<td>320.552</td>
<td>47.700</td>
<td>56.300</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>49</td>
<td>301.200</td>
<td>24.271</td>
<td>412.480</td>
<td>129.000</td>
<td>127.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>50</td>
<td>301.200</td>
<td>22.727</td>
<td>385.908</td>
<td>204.000</td>
<td>283.000</td>
<td>341.000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Outline

- Overview
- Sample data: airfares
- A few options
  - Be naïve: subset()
  - Use a loop
  - tapply()
  - aggregate()
  - doBy's summaryBy()
  - plyr's ddply()
  - more more more...
Overview: group & summarize data

• Very common to have detail-level data from which you need summary-level statistics based on some grouping variable or variables
  • Sales by region, market share by company, discount and margin by product line and rep, etc.

• Hadley Wickham coined term “split-apply-combine” to describe this analysis pattern
  • c.f. SQL's “GROUP BY”, SAS has “by”, MapReduce

• There's more than one way to do it in R

• Often discussed on StackOverflow.com &c.
Sample data: BOS-NYC airfares

Individual airfares paid from Boston Logan (BOS) to New York City airports (EWR, JFK, LGA) last year:

```r
> nrow(df)
[1] 1852

> head(df)
  Origin Dest Carrier   Fare
  1    BOS  EWR      CO  56.32
  2    BOS  EWR      9L   0.00
  3    BOS  EWR      CO 102.00
  4    BOS  EWR      CO 109.00
  5    BOS  EWR      CO 130.00
  6    BOS  EWR      CO 147.50

> tail(df)
  Origin Dest Carrier   Fare
 1847    BOS  LGA      DL 208.87
 1848    BOS  LGA      DL 223.79
 1849    BOS  LGA      US 100.46
 1850    BOS  LGA      UA 125.89
 1851    BOS  LGA      US 167.63
 1852    BOS  LGA      US 186.68

> unique(df$Dest)
[1] "EWR" "JFK" "LGA"
```
Naïve approach – split by hand

```r
> ewr = subset(df, Dest=='EWR')
> jfk = subset(df, Dest=='JFK')
> lga = subset(df, Dest=='LGA')

> # counts:
> nrow(ewr)
[1] 392
> nrow(jfk)
[1] 572
> nrow(lga)
[1] 888

> # averages:
> mean(ewr$Fare)
[1] 267.6365
> median(ewr$Fare)
[1] 210.85
> mean(jfk$Fare)
[1] 147.3658
> median(jfk$Fare)
[1] 113.305
> mean(lga$Fare)
[1] 190.2382
> median(lga$Fare)
[1] 171
```
Automating naïveté with a loop

\[
\text{results} = \text{data.frame()}
\]

\[
\text{for ( dest in unique(df$Dest) ) }
\{
  \text{tmp} = \text{subset(df, Dest==dest)}
  \text{count} = \text{nrow(tmp)}
  \text{mean} = \text{mean(tmp$Fare)}
  \text{median} = \text{median(tmp$Fare)}
  \text{results} = \text{rbind(results, data.frame(dest, count, mean, median) ) }
\}
\]

\[
> \text{results}
\]

\[
\begin{array}{cccc}
\text{dest} & \text{count} & \text{mean} & \text{median} \\
\text{EWR} & 392 & 267.6365 & 210.850 \\
\text{JFK} & 572 & 147.3658 & 113.305 \\
\text{LGA} & 888 & 190.2382 & 171.000 \\
\end{array}
\]

Rule of Thumb: if you're using a loop in R, you're probably doing something wrong
Base R's `tapply()`

Applying functions repeatedly sounds like a job for Base R's *apply() functions:

```r
> tapply(df$Fare, df$Dest, FUN=length)
  EWR  JFK  LGA
  392  572  888

> tapply(df$Fare, df$Dest, FUN=mean)
  EWR    JFK    LGA
  267.64 147.37 190.24

> tapply(df$Fare, df$Dest, FUN=median)
  EWR    JFK    LGA
  210.85 113.31 171.00
```

I'm honestly not thrilled with the output format, but I'm sure we could wrestle into a `data.frame` which includes the grouping variable thanks to the `names()` function.
> aggregate(Fare~Dest, data=df, FUN="mean")
  Dest   Fare
1  EWR  267.6365
2  JFK  147.3658
3  LGA  190.2382

> aggregate(Fare~Dest, data=df, FUN="median")
  Dest   Fare
1  EWR  210.850
2  JFK  113.305
3  LGA  171.000

> aggregate(Fare~Dest, data=df, FUN="length")
  Dest Fare
1  EWR  392
2  JFK  572
3  LGA  888

- `data.frame` in, `data.frame` out (works for time series `ts, mts` too)
- Uses formula notation & `data.frame` environment aware (no `$`s)
doBy package's `summaryBy()`

More capable and simpler (at least for me)

- Accepts formula to access multiple columns for values and groupings
- Accepts anonymous functions which can use `c()` to perform multiple operations
- `doBy` package also provides formula-based `lapplyBy()` and my favorite sorting function, `orderBy()`

```r
> summaryBy(Fare~Dest, data=df, FUN=function(x)
c(count=length(x), mean=mean(x), median=median(x)))
```

<table>
<thead>
<tr>
<th>Dest</th>
<th>Fare.count</th>
<th>Fare.mean</th>
<th>Fare.median</th>
</tr>
</thead>
<tbody>
<tr>
<td>EWR</td>
<td>392</td>
<td>267.6365</td>
<td>210.850</td>
</tr>
<tr>
<td>JFK</td>
<td>572</td>
<td>147.3658</td>
<td>113.305</td>
</tr>
<tr>
<td>LGA</td>
<td>888</td>
<td>190.2382</td>
<td>171.000</td>
</tr>
</tbody>
</table>
Hadley Wickham's plyr package

• Provides a standard naming convention:
  • X + Y + “ply”
    – X = input data type
    – Y = output data type
    – Types:
      • “a” = array
      • “d” = data.frame
      • “l” = list
      • “m” = matrix
      • “_” = no output returned
  • Example: ddply() expects and returns a data.frame

• Most plyr functions wrap other plyr, Base functions
**ddply() in action**

Like summaryBy(), can use multiple-part grouping variables and functions

```r
> ddply(df, 'Dest', function(x) c(count=nrow(x), mean=mean(x$Fare), median=median(x$Fare)))
   Dest count    mean    median
1  EWR   392 267.6365 210.850
2  JFK   572 147.3658 113.305
3  LGA   888 190.2382 171.000

> ddply(df, c('Dest', 'Carrier'), function(x) c(count=nrow(x), mean=mean(x$Fare), median=median(x$Fare)))
   Dest Carrier count    mean    median
1  EWR      9L    33 181.9697 131.500
2  EWR      CO   326 279.7623 264.250
3  EWR      XE    33 233.5152 152.500
4  JFK      AA     6 129.6600 140.120
5  JFK      B6  112 132.2796 108.245
[...]```
“Are we there yet?”

Good news: plyr provides “.parallel” & “.progress” bar options for long-running jobs

```r
> ddply(df, 'Dest', function(x) c(count=nrow(x), mean=mean(x$Fare), median=median(x$Fare)), .progress='text')
```

|==============================================================================| 100%
<table>
<thead>
<tr>
<th>Dest</th>
<th>count</th>
<th>mean</th>
<th>median</th>
</tr>
</thead>
<tbody>
<tr>
<td>EWR</td>
<td>392</td>
<td>267.6365</td>
<td>210.850</td>
</tr>
<tr>
<td>JFK</td>
<td>572</td>
<td>147.3658</td>
<td>113.305</td>
</tr>
<tr>
<td>LGA</td>
<td>888</td>
<td>190.2382</td>
<td>171.000</td>
</tr>
</tbody>
</table>

Bad news: you may need them

- Has been getting faster, major work planned for summer 2011 (Hadley's goal: “as fast as data.table” !)
- “immutable” `idata.frame` in plyr 1.0 can help now
- Great speed & alternatives discussion:
  [http://stackoverflow.com/questions/3685492/r-speeding-up-group-by-operations/3685919](http://stackoverflow.com/questions/3685492/r-speeding-up-group-by-operations/3685919)
Other options & approaches

- Loops that don't (necessarily) suck: foreach
  - Works with parallel backends (SMP, MPI, SNOW, etc.)
- Have data in a database?
  - DBI & friends to access and group (RMySQL, RPostgresSQL, ROracle, RJDBC, RODBC, etc.)
  - For MySQL, Postgres, look at dbApply() to aggregate
- sqldf will create a temporary database for you
- Data > Memory? http://www.bigmemory.org/
  - or just use Hadoop for everything: RHIPE
References and further reading

- StackOverflow.com discussions (has an active “[r]” tag for searching)
  - “for each group summarise means for all variables in dataframe (ddply? Split?)”
    - http://stackoverflow.com/questions/1407449/for-each-group-summarise-means-for-all-variables-in-dataframe-ddply-split
  - “How to split a data frame by rows, and then process the blocks?”
  - “R Grouping functions: sapply vs. lapply vs. apply vs. tapply vs. by vs. aggregate vs. …”
  - “how to aggregate this data in R”

- JD Long: A Fast intro to Plyr

- Kane & Emerson: “Scalable Strategies for Computing with Massive Data: The Bigmemory Project”
  - http://www.slideshare.net/joshpaulson/big-memory