Statistical Computing (36-350)
Lecture 12: Split/Apply/Combine with Base R

Cosma Shalizi
Massive thanks to Vince Vu

7 October 2013
Agenda

- Splitting and aggregated for data analysis
- Examples of the pattern
- Unemployment and strikes across countries
- Tools in base R: `subset`, `split`, `*apply`, `*bind`, `do.call`

Reading: *The R Cookbook*, chapter 6; Matloff, chapter 6
Lots of problems in programming and data analysis can be solved by similar types and sequences of actions. **Design patterns** and **Analysis patterns**

We will look at the pattern called “split, apply, combine” (Hadley Wickham)
The Point of Learning Patterns

Distinguish between what you want to do and how you want to do it
Focusing on what brings clarity to intentions
How also matters, but can obscure the high-level problem
Learn the pattern, recognize the pattern, love the pattern
Re-use good solutions
Large data sets are usually highly structured
Structure lets us group data in many different ways
Sometimes we focus on individual pieces of data
Often we aggregate information within groups, and compare across them
A Trivial Example

Row (column) means of a matrix
- Divide the matrix into rows (columns)
- Compute the mean of each row (column)
- Combine the results into a vector
Row Means

Matrix
(an array of dimension 2)
Row Means

(vector, dimension 1)

(vector, dimension 1)

(vector, dimension 1)
Row Means

mean()

mean()

mean()
Row Means
Row Means

vector (of dimension 1)
Data organized into 48 continental states
Fit a different model for each of 4 different geographic regions
Splitting by Region

data.frame
Splitting by Region

data.frames
Splitting by Region

\[ \text{lm( )} \]

\[ \text{lm( )} \]

\[ \text{lm( )} \]

\[ \text{lm( )} \]

\[ \text{lm( )} \]
Splitting by Region

1m objects
list of lm objects
The Basic Pattern

split

apply

combine

\[ f(\quad) \]

\[ f(\quad) \]

\[ f(\quad) \]
The Basic Pattern (cont’d.)

Split  divide the problem into smaller pieces
Apply  Work on each piece independently
Combine  Recombine the pieces

A common pattern for both programming and data analysis, many implementations
Python: map(), filter(), reduce()
Google mapReduce
R: split, *apply, aggregate,…
R: plyr package
Could always do the same thing with for loops, but those are
- verbose — lots of “how”, obscures “what”
- painful/error-prone book-keeping (indices, placeholders, …)
- clumsy — hard to parallelize
x <- array(STUFF, dim=c(10,10,100))

Data: 10 × 10 grid of locations, 100 measurements / location
Desired: sample SD at each location
Iteration:

```r
sds <- array(dim= dim(x)[1:2])
for (i in 1:dim(x)[1]) {
  for (j in 1:dim(x)[2]) {
    sds[i,j] <- sd(x[i,j,])
  }
}
```

apply:

```r
sds <- apply(x, 1:2, sd)
```
y <- apply(X, MARGIN, FUNCTION, ...)

- **X** an array
- **MARGIN** vector of subscripts which the function is applied over
- **FUNCTION** the function to be applied
- **...** additional arguments to function (held constant)

Returns an array if it can, a list if all else fails
apply()

```r
y <- apply(x, c(1,3), f)
Compute \( f(x[i, , j, ]) \) for all \( i,j \)
```

```r
y <- apply(x, 2:4, f)
Compute \( f(x[,i,j,k,]) \) for all \( i,j,k \)
```
*apply()*

Variants for different data structures:
- `apply()` for arrays
- `lapply()` and `sapply()` for lists and vectors
- `mapply()` for multivariate functions

Consult textbooks and R help for details
But... 

What about ragged data — different numbers of observations at each location? 
More complex situations?
Politics and Labor Action

Does having a friendlier government make labor action more or less likely?

March on Washington, 1963

Madison protests, 2011
Compiled by Prof. Bruce Western at Harvard
Data frame of 8 columns

country, year, days on strike per 1000 workers, unemployment, inflation, left-wing share of
gov’t, centralization of unions, union density

“centralization” not useful to us so we’ll drop it

625 observations from 18 countries, 1951–1985

18 × 35 = 630 > 625, ∴ some years missing from some countries

strikes <- read.csv("http://www.stat.cmu.edu/~cshalizi/uADA/12/hw/06/strikes.csv")
### A Little Bit of the Data

<table>
<thead>
<tr>
<th>country</th>
<th>year</th>
<th>strike.volume</th>
<th>unemployment</th>
<th>inflation</th>
<th>left.parliament</th>
<th>density</th>
</tr>
</thead>
<tbody>
<tr>
<td>Australia</td>
<td>1983</td>
<td>313</td>
<td>9.8</td>
<td>10.1</td>
<td>60</td>
<td>48.5</td>
</tr>
<tr>
<td>Australia</td>
<td>1984</td>
<td>241</td>
<td>8.9</td>
<td>4</td>
<td>55.4</td>
<td>47.6</td>
</tr>
<tr>
<td>Australia</td>
<td>1985</td>
<td>226</td>
<td>8.2</td>
<td>6.7</td>
<td>55.4</td>
<td>45.9</td>
</tr>
<tr>
<td>Austria</td>
<td>1951</td>
<td>43</td>
<td>3.5</td>
<td>27.5</td>
<td>43.6</td>
<td>NA</td>
</tr>
<tr>
<td>Austria</td>
<td>1952</td>
<td>39</td>
<td>4.7</td>
<td>13.6</td>
<td>43.6</td>
<td>NA</td>
</tr>
<tr>
<td>Austria</td>
<td>1953</td>
<td>20</td>
<td>5.8</td>
<td>-1.6</td>
<td>46.7</td>
<td>NA</td>
</tr>
</tbody>
</table>
Plan

- Look at the relation between strikes and left-wing parties for a country
- Encapsulate the analysis into a function
- **Split** the data by country
- **Apply** the function to each country
- **Combine** the results
df <- subset(strikes, country == "Italy")
italy <- lm(strike.volume ~ left.parliament, data=df)
plot(strike.volume ~ left.parliament, data=df)
abline(italy)
Italy, For Example
What About Country X?

strikes_vs_left <- function(df, coefficients.only=FALSE) {
  fit <- lm(strike.volume ~ left.parliament, data=df)
  if (coefficients.only) {
    return(coefficients(fit))
  } else {
    return(fit)
  }
}

How about Belgium?

belgium <- strikes_vs_left(subset(strikes, country=="Belgium"))

EXERCISE: Make a plot like the one for Italy
x <- split(strikes, strikes$country)

$country is a factor vector: countries are levels of the factor
split the data frame according to the levels of $country
x is a list of data frames
y <- lapply(x, strikes_vs_left, coefficients.only=TRUE)

Apply strikes_vs_left() to each element of x
Result is a list of coefficient vectors
Turning off coefficients.only would give a list of lm model objects
Combine the vectors into an array

```r
coops <- do.call(rbind, y)
```

Equivalent to

```r
rbind(y[[1]], y[[2]], ... y[[18]])
```

but don’t have to know how long y is

Vectors bound together have to be of the same length
split, apply, combine, using only base R

x <- split(strikes, strikes$country)
y <- lapply(x, strikes_vs_left, coefficients.only=TRUE)
coefs <- do.call(rbind, y)

Iteration

coops <- matrix(nrow=nlevels(strikes$country),ncol=2)
for (i in 1:nlevels(strikes$country)) {
  x <- subset(strikes, country==levels(strikes$country)[i])
  coefs[i,] <- strikes_vs_left(x,coefficients.only=TRUE)
}
rownames(coefs) <- levels(strikes$country)

EXERCISE: replace subset() with more iteration
plot(coefs[,2], xaxt="n", xlab="", ylab="Regression coefficient")
axis(side=1, at=seq(along=rownames(coefs)), labels=rownames(coefs),
     las=2, cex.axis=0.5)
abline(h=0, col="grey")
Lots of (apparent) heterogeneity across countries
Actual differences across countries might be conflated with different economic circumstances: try adding covariates to the regression
Arranging countries alphabetically is uninformative — maybe by geography or cultural groupings?

**EXERCISE:** Re-arrange so all English-speaking countries are on the far right

Really should have error bars if we’re going to compare

**EXERCISE:** Modify code to return standard errors for coefficients, use `segments` to add ±2se error bars to each point estimate
The split, apply, combine pattern is very common
Recognize it!
Iteration is usually not a good solution
*apply is usually a better solution
Next time: abstracting the pattern with the plyr package