Data Summarization

Andrew Jaffe

January 5, 2016
Data Summarization

- Basic statistical summarization
  - `mean(x)`: takes the mean of \( x \)
  - `sd(x)`: takes the standard deviation of \( x \)
  - `median(x)`: takes the median of \( x \)
  - `quantile(x)`: displays sample quantities of \( x \). Default is min, IQR, max
  - `range(x)`: displays the range. Same as \( c(\min(x), \max(x)) \)
Some examples

We can use the mtcars and Charm City Circulator datasets to explore different ways of summarizing data.

head(mtcars)

## mpg cyl disp hp drat wt qsec vs am gear carb
## Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4
## Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4
## Datsun 710 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1
## Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1
## Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2
## Valiant 18.1 6 225 105 2.76 3.460 20.22 1 0 3 1
Statistical summarization

```r
mean(mtcars$hp)
```

```
## [1] 146.6875
```

```r
quantile(mtcars$hp)
```

```
## 0% 25% 50% 75% 100%
## 52.0 96.5 123.0 180.0 335.0
```
Statistical summarization

\texttt{median(mtcars$wt)}

## [1] 3.325

\texttt{quantile(mtcars$wt, probs = 0.6)}

## 60%
## 3.44
Statistical summarization

Note that many of these functions have additional inputs regarding missing data, typically requiring the `na.rm` argument.

```r
x = c(1,5,7,NA,4,2, 8,10,45,42)
mean(x)
## [1] NA

mean(x, na.rm=TRUE)
## [1] 13.77778

quantile(x, na.rm=TRUE)
##    0%   25%   50%   75%  100%
##    1    4    7   10   45
```
Data Summarization on matrices/data frames

- **Basic statistical summarization**
  - `rowMeans(x)`: takes the means of each row of `x`
  - `colMeans(x)`: takes the means of each column of `x`
  - `rowSums(x)`: takes the sum of each row of `x`
  - `colSums(x)`: takes the sum of each column of `x`
  - `summary(x)`: for data frames, displays the quantile information
Charm City Circulator data

Please download the Charm City Circulator data:

http://www.aejaffe.com/winterR_2016/data/Charm_City_Circulator_Ridership.csv

circ = read.csv("http://www.aejaffe.com/winterR_2016/data/Charm_City_Circulator_Ridership.csv", header=TRUE, as.is=TRUE)
Subsetting to specific columns

Let's just take columns that represent average ridership:

```r
library(dplyr, quietly = TRUE)
circ2 = select(circ, date, day, ends_with("Average"))
```
column and row means

```r
avgs = select(circ2, ends_with("Average"))
colMeans(avgs, na.rm=TRUE)
```

```r
## orangeAverage purpleAverage greenAverage bannerAverage
## 3033.1611 4016.9345 1957.7814 827.2685
```

```r
circ2$daily = rowMeans(avgs, na.rm=TRUE)
head(circ2$daily)
```

```r
## [1] 952.0  796.0 1211.5 1213.5 1644.0 1490.5
```
### Summary

```r
circle2
```

<table>
<thead>
<tr>
<th></th>
<th>date</th>
<th>day</th>
<th>orangeAverage</th>
<th>purpleAverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length: 1146</td>
<td>Length: 1146</td>
<td>Min. : 0</td>
<td>Min. : 0</td>
<td></td>
</tr>
<tr>
<td>Class: character</td>
<td>Class: character</td>
<td>1st Qu.: 2001</td>
<td>1st Qu.: 2795</td>
<td></td>
</tr>
<tr>
<td>Mode: character</td>
<td>Mode: character</td>
<td>Median : 2968</td>
<td>Median : 4222</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mean : 3033</td>
<td>Mean : 4017</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3rd Qu.: 4020</td>
<td>3rd Qu.: 5147</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Max. : 6926</td>
<td>Max. : 8090</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NA's : 10</td>
<td>NA's : 15</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>greenAverage</th>
<th>bannerAverage</th>
<th>daily</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min. : 0</td>
<td>Min. : 0.0</td>
<td>Min. : 0</td>
<td>10</td>
</tr>
<tr>
<td>1st Qu.: 1491</td>
<td>1st Qu.: 632.5</td>
<td>1st Qu.: 2097</td>
<td></td>
</tr>
<tr>
<td>Median : 2079</td>
<td>Median : 763.0</td>
<td>Median : 2846</td>
<td></td>
</tr>
<tr>
<td>Mean : 1958</td>
<td>Mean : 827.3</td>
<td>Mean : 2878</td>
<td></td>
</tr>
<tr>
<td>3rd Qu.: 2340</td>
<td>3rd Qu.: 945.9</td>
<td>3rd Qu.: 3646</td>
<td></td>
</tr>
<tr>
<td>Max. : 5094</td>
<td>Max. : 4617.0</td>
<td>Max. : 6123</td>
<td></td>
</tr>
<tr>
<td>NA's : 661</td>
<td>NA's : 876</td>
<td>NA's : 10</td>
<td></td>
</tr>
</tbody>
</table>
Apply statements

You can apply more general functions to the rows or columns of a matrix or data frame, beyond the mean and sum.

apply(X, MARGIN, FUN, ...)

X : an array, including a matrix.
MARGIN : a vector giving the subscripts which the function will be applied over. E.g., for a matrix 1 indicates rows, 2 indicates columns, c(1, 2) indicates rows and columns. Where X has named dimnames, it can be a character vector selecting dimension names.
FUN : the function to be applied: see ‘Details’.
... : optional arguments to FUN.
Apply statements

```r
apply(avgs, 2, mean, na.rm = TRUE)  # column means

## orangeAverage  purpleAverage  greenAverage  bannerAverage
## 3033.1611   4016.9345   1957.7814    827.2685
```

```r
apply(avgs, 2, sd, na.rm = TRUE)  # columns sds

## orangeAverage  purpleAverage  greenAverage  bannerAverage
## 1227.5779   1406.6544    592.8969    436.0487
```

```r
apply(avgs, 2, max, na.rm = TRUE)  # column maxs

## orangeAverage  purpleAverage  greenAverage  bannerAverage
## 6926.5    8089.5       5094.0     4617.0
```
Other Apply Statements

- `tapply()`: ‘table’ apply
- `lapply()`: ‘list’ apply [tomorrow]
- `sapply()`: ‘simple’ apply [tomorrow]
- Other less used ones...

See more details here: http://nsaunders.wordpress.com/2010/08/20/a-brief-introduction-to-apply-in-r/
tapply()

From the help file: “Apply a function to each cell of a ragged array, that is to each (non-empty) group of values given by a unique combination of the levels of certain factors.”

tapply(X, INDEX, FUN = NULL, ..., simplify = TRUE)

Simply put, you can apply function FUN to X within each categorical level of INDEX. It is very useful for assessing properties of continuous data by levels of categorical data.
tapply()

For example, we can estimate the highest average daily ridership for each day of the week in 1 line in the Circulator dataset.

```
tapply(circ2$daily, circ2$day, max, na.rm=TRUE)
```

```
## Friday Monday Saturday Sunday Thursday Tuesday Wednesday
## 5600.75 5002.25 6123.00 3980.25 4820.50 4855.25 4985.00
```
Data Summarization

- Basic summarization plots
  - `plot(x,y)`: scatterplot of x and y
  - `boxplot(y~x)`: boxplot of y against levels of x
  - `hist(x)`: histogram of x
  - `density(X)`: kernel density plot of x
Basic Plots

Plotting is an important component of exploratory data analysis. We will review some of the more useful and informative plots here. We will go over formatting and making plots look nicer in additional lectures.
Scatterplot

\texttt{plot(mtcars$mpg, mtcars$disp)}
Histograms

```r
hist(circ2$daily)
```
Density

```r
## plot(density(circ2$daily))
plot(density(circ2$daily, na.rm=TRUE))
```

density.default(x = circ2$daily, na.rm = TRUE)

N = 1136   Bandwidth = 232.1

![Density Plot](image-url)
Boxplots

\texttt{boxplot(circ2\$daily \sim circ2\$day)}
Boxplots

```r
boxplot(daily ~ day, data=circ2)
```
Data Summarization for data.frames

- Basic summarization plots
  - `matplot(x,y)`: scatterplot of two matrices, x and y
  - `pairs(x,y)`: plots pairwise scatter plots of matrices x and y, column by column
Matrix plot

`matplot(avgs)`