Data Summarization

Module 7

Andrew Jaffe

June 17, 2015

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))
- 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

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
- 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

column and row means

```
circ2 = read.csv("../data/charmcitycirc_reduced.csv",
            header=TRUE,as.is=TRUE)
colMeans(circ2[,3:6],na.rm=TRUE)
```

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

```r
head(rowMeans(circ2[,3:6],na.rm=TRUE))
```

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

Summary

```r
summary(circ2)
```

```
##      day                 date           orangeAverage  purpleAverage
##  Length:1146        Length:1146        Min.   :   0   Min.   :   0
##  Class :character   Class :character   1st Qu.:2001   1st Qu.:2795
##  Mode  :character   Mode  :character   Median :2968   Median :4222
##                                        Mean   :3033   Mean   :4017
##                                        3rd Qu.:4020   3rd Qu.:5147
##                                        Max.   :6926   Max.   :8090
##                                        NA's   :10     NA's   :153
##   greenAverage  bannerAverage        daily
##  Min.   :   0   Min.   :   0.0   Min.   :    0
##  1st Qu.:1491   1st Qu.: 632.5   1st Qu.: 4293
##  Median :2079   Median : 763.0   Median : 6702
##  Mean   :1958   Mean   : 827.3   Mean   : 7233
##  3rd Qu.:2340   3rd Qu.: 945.9   3rd Qu.:10501
##  Max.   :5094   Max.   :4617.0   Max.   :22075
##  NA's   :661    NA's   :876      NA's   :124
```

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
tmp = circ2[,3:6]
apply(tmp,2,mean,na.rm=TRUE) # column means
```

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

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

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

```r
apply(tmp,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.

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

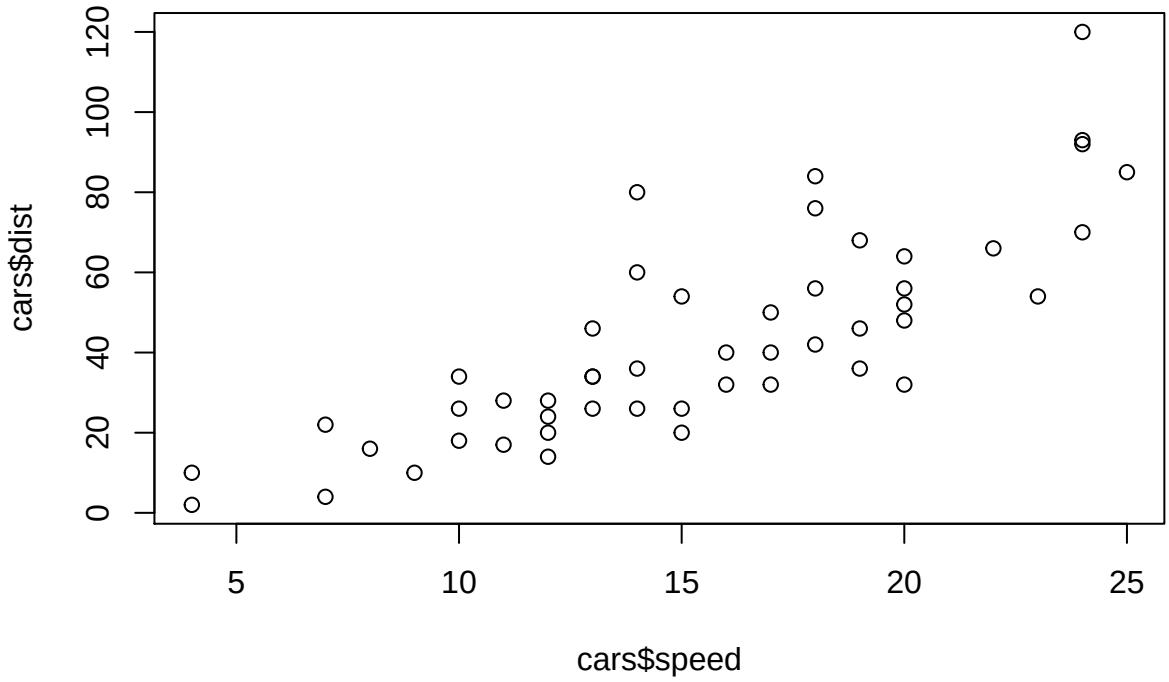
```
##    Friday    Monday  Saturday    Sunday  Thursday   Tuesday Wednesday
##   21951.0   13982.0   22074.5   15224.5   17580.0   14775.5   15672.5
```

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.

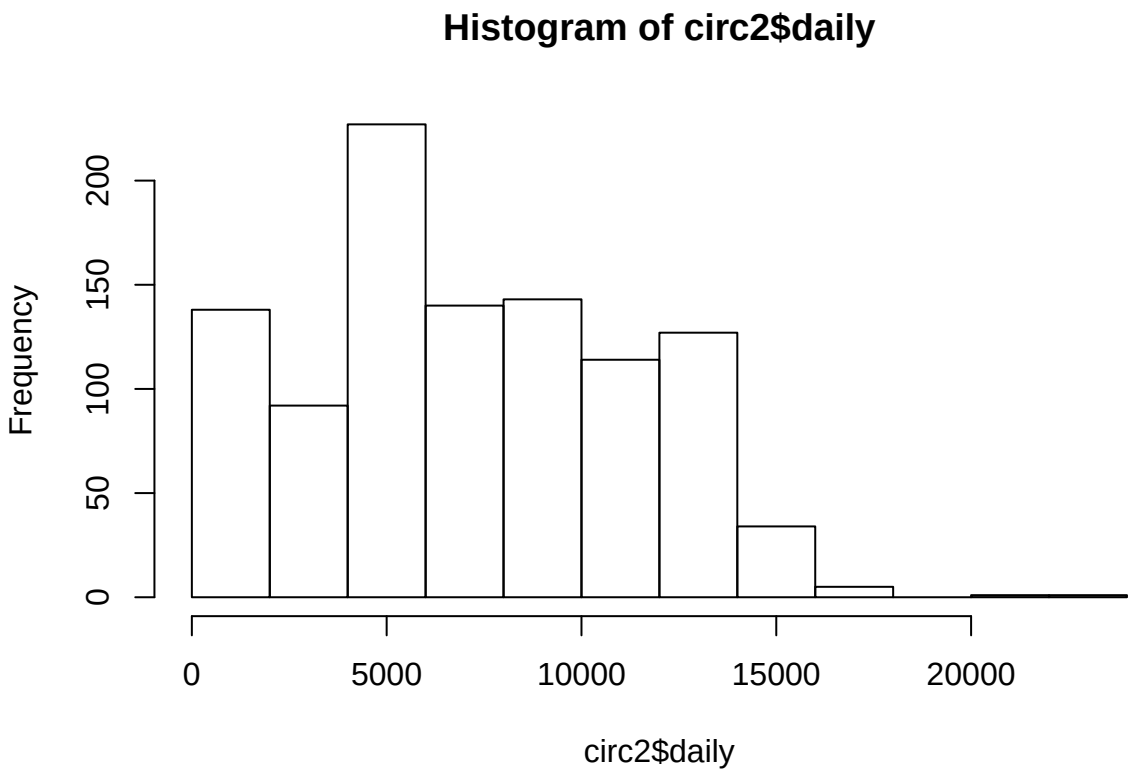
Scatterplot

```r
data(cars)
plot(cars$speed, cars$dist)
```



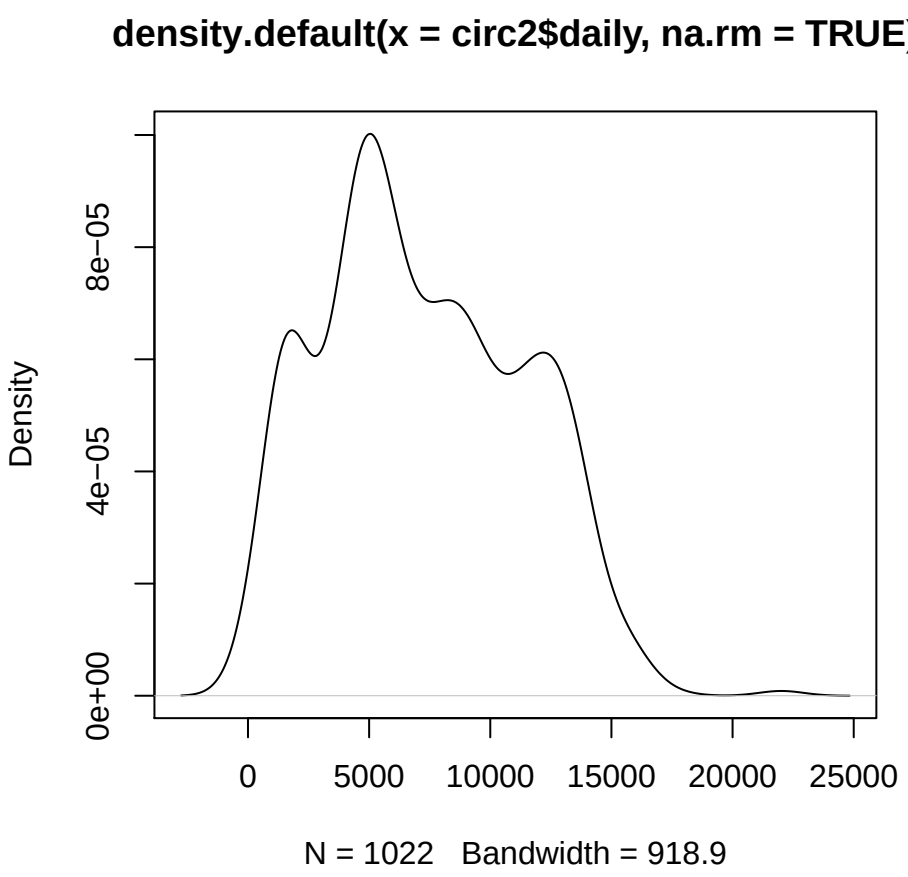
Histograms

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



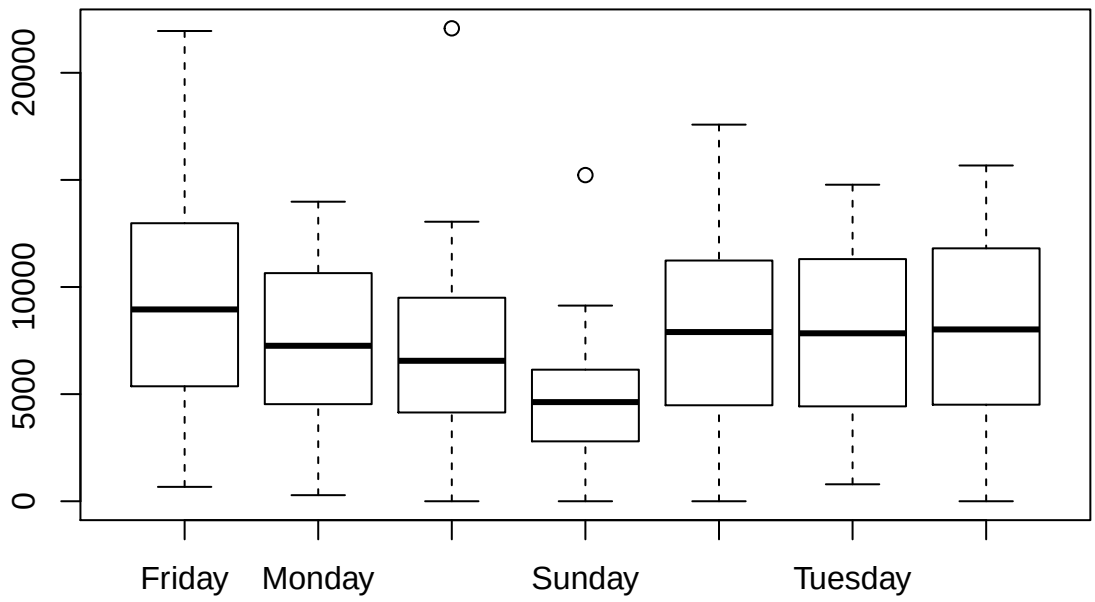
Density

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



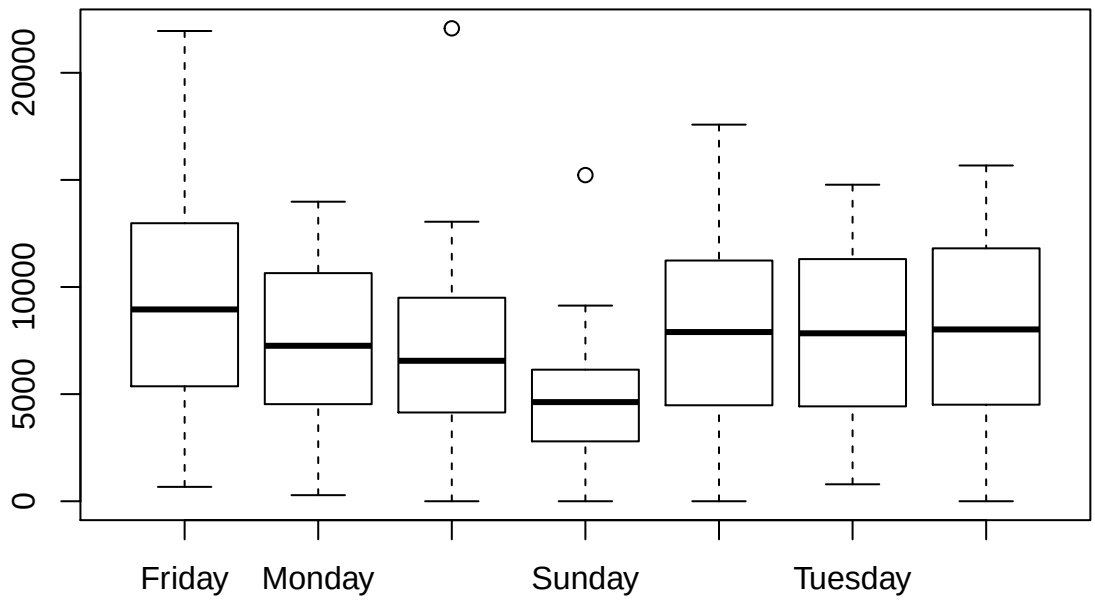
Boxplots

```
boxplot(circ2$daily ~ circ2$day)
```



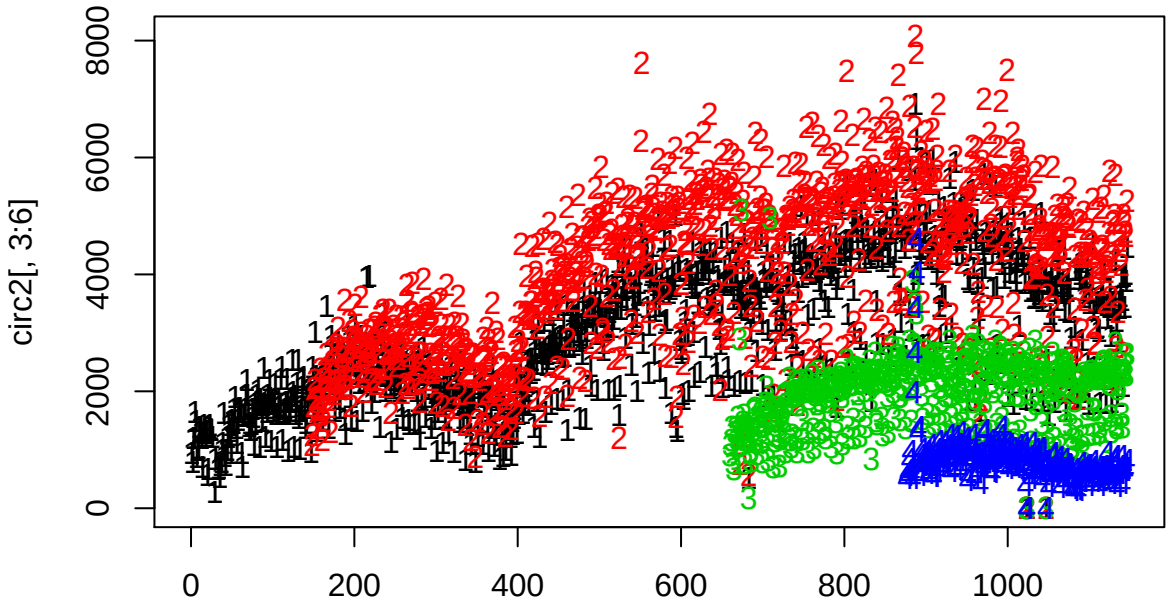
Boxplots

```
boxplot(daily ~ day, data=circ2)
```



Matrix plot

```
matplot(circ2[,3:6])
```


circ2[, 3:6]
0
2000
4000
6000
8000
0
200
400
600
800
1000