

Module 7

Data Summarization

Andrew Jaffe
Instructor

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

```
> dat = read.csv("data/charmcitycirc_reduced.csv", header = T, as.is = T)
> dat2 = dat[, c("day", "date", "orangeAverage", "purpleAverage", "greenAverage",
+ "bannerAverage", "daily")]
> tmp = dat2[, 3:6]
> colMeans(tmp, na.rm = TRUE)
```

```
orangeAverage purpleAverage greenAverage bannerAverage
          2994           4013           1951           964
```

```
> head(rowMeans(tmp, na.rm = TRUE))
```

```
[1] 952 796 1212 1214 1644 1490
```

Summary

```
> summary(dat2)
```

```
      day          date      orangeAverage  purpleAverage
Length:1025      Length:1025      Min.   :    0      Min.   :    0
Class :character  Class :character  1st Qu.:1933      1st Qu.:2761
Mode  :character  Mode  :character  Median :2831      Median :4155
                                           Mean  :2994      Mean  :4013
                                           3rd Qu.:4080      3rd Qu.:5218
                                           Max.  :6926      Max.  :8090
                                           NA's  :9          NA's  :152

      greenAverage  bannerAverage      daily
Min.   :    0      Min.   :    0      Min.   :    0
1st Qu.:1482      1st Qu.: 760      1st Qu.: 4293
Median :2040      Median : 900      Median : 6663
Mean   :1951      Mean   : 964      Mean   : 7230
3rd Qu.:2344      3rd Qu.:1028      3rd Qu.:10501
Max.   :5094      Max.   :4617      Max.   :22074
NA's   :660       NA's   :875       NA's   :3
```

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'. In the case of functions like +, %*%, etc., the function name must be backquoted or quoted.

... : optional arguments to FUN.

Apply statements

```
> tmp = dat2[, 3:6]  
> apply(tmp, 2, mean, na.rm = TRUE) # column means
```

```
orangeAverage purpleAverage greenAverage bannerAverage  
2994          4013          1951          964
```

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

```
orangeAverage purpleAverage greenAverage bannerAverage  
1258.7        1442.4        613.7        527.1
```

```
> apply(tmp, 2, max, na.rm = TRUE) # column maxs
```

```
orangeAverage purpleAverage greenAverage bannerAverage  
6926          8090          5094          4617
```

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

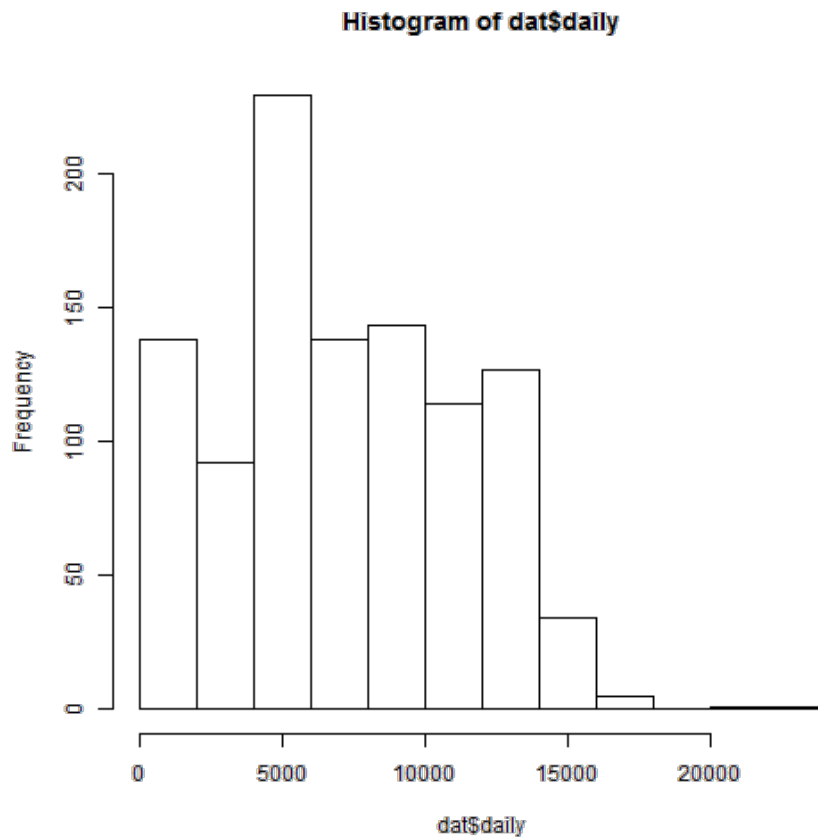
Friday	Monday	Saturday	Sunday	Thursday	Tuesday	Wednesday
21951	13982	22075	15224	17580	14776	15672

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.

Histograms

```
> hist(dat$daily)
```



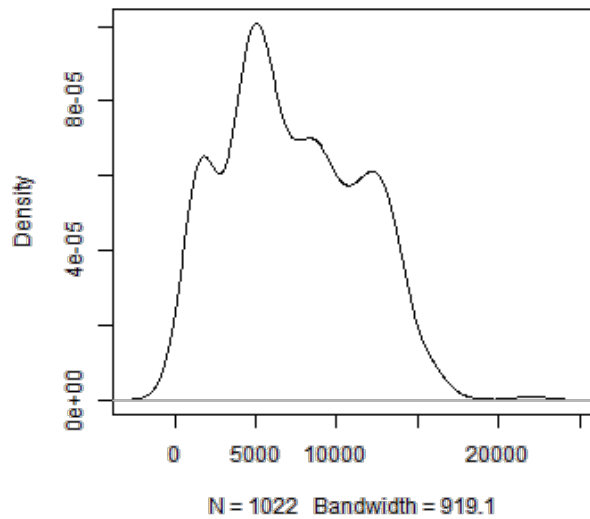
Density

```
> plot(density(dat$daily))
```

```
Error: 'x' contains missing values
```

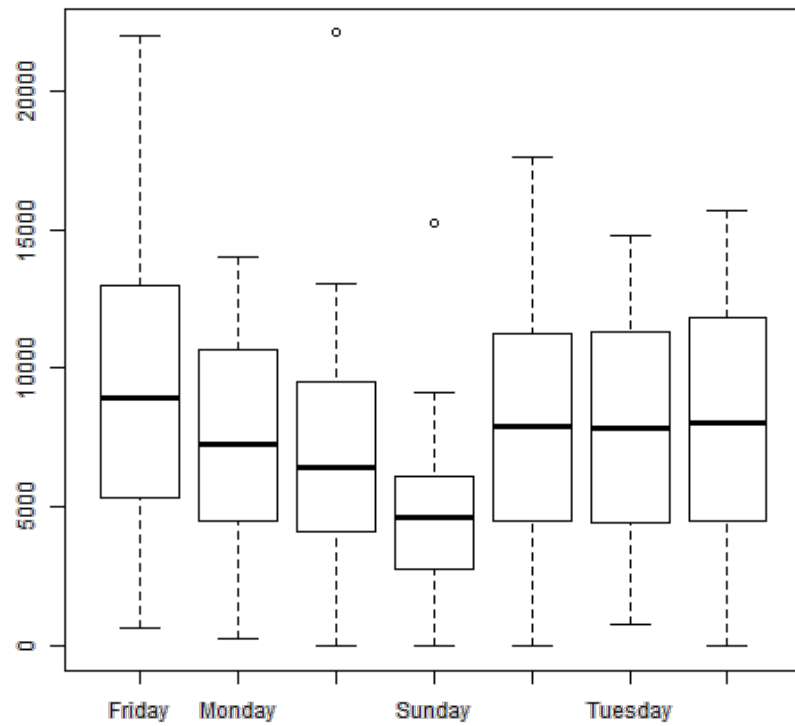
```
> plot(density(dat$daily, na.rm = TRUE))
```

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



Boxplots

```
> boxplot(dat$daily ~ dat$day)
```



Matrix plot

```
> matplot(dat2[, 3:6])
```

