Data Summarization

Introduction to R for Public Health Researchers

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
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0
```

Statistical summarization

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

```
[1] 146.6875
```

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

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

Statistical summarization

```r
median(mtcars$wt)
```

```
[1] 3.325
```

```r
quantile(mtcars$wt, probs = 0.6)
```

```
  60%
3.44
```

Statistical summarization

`t.test` will be covered more in detail later, gives a 95% CI:

```
t.test(mtcars$wt)
```

```
    One Sample t-test

data:  mtcars$wt
t = 18.6, df = 31, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 2.864478 3.570022
sample estimates:
mean of x
  3.21725
```

Statistical summarization

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

```
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/summerR_2016/data/Charm_City_
Circulator_Ridership.csv

```
circ = read_csv(
  "http://www.aejaffe.com/summerR_2016/data/Charm_City_Circ
```

```
Parsed with column specification:
cols(
  day = col_character(),
  date = col_character(),
  orangeBoardings = col_integer(),
  orangeAlightings = col_integer(),
  orangeAverage = col_double(),
  purpleBoardings = col_integer(),
  purpleAlightings = col_integer(),
  purpleAverage = col_double(),
  greenBoardings = col_integer(),
```

Subsetting to specific columns

Let's just take columns that represent average ridership:

```r
library(dplyr)
circ2 = select(circ, date, day, ends_with("Average"))
head(circ2, 4)
```

```
# A tibble: 4 × 6
       date        day orangeAverage purpleAverage greenAve
      <chr>      <chr>         <dbl>         <dbl>        <
1 01/11/2010    Monday         952.0            NA
2 01/12/2010   Tuesday         796.0            NA
3 01/13/2010 Wednesday        1211.5            NA
4 01/14/2010  Thursday        1213.5            NA
# ... with 1 more variables: bannerAverage <dbl>
```

column and row means

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

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

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

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

Summary

```r
summary(circ2)
```

```
     date                day              orangeAverage    purpl
 Length:1146        Length:1146         Min.   :   0      Min.
 Class :character   Class :character    1st Qu.:2001      1st Q
 Mode  :character   Mode  :character    Median :2968      Media
                                        Mean   :3033      Mean
                                        3rd Qu.:4020      3rd Q
                                        Max.   :6926      Max.
                                        NA's   :10        NA's

  greenAverage    bannerAverage          daily
 Min.   :   0    Min.   :   0.0     Min.   :   0
 1st Qu.:1491    1st Qu.: 632.5     1st Qu.:2097
 Median :2079    Median : 763.0     Median :2846
 Mean   :1958    Mean   : 827.3     Mean   :2878
 3rd Qu.:2340    3rd Qu.: 945.9     3rd Qu.:3646
 Max.   :5094    Max.   :4617.0     Max.   :6123
 NA's   :661     NA's   :876        NA's   :10
```

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

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

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

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

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

```
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()`: 'grouping' 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
  5600.75  5002.25  6123.00  3980.25  4820.50  4855.25
```

Perform Operations By Groups: dplyr

group_by allows you group the data in a more intuitive way than
tapply

We will use group_by to group the data by line, then use
summarize (or summarise) to get the mean Average ridership:

```
summarize(group_by(circ, day), max_avg = max(daily, na.rm =
```

```
# A tibble: 7 × 2
         day max_avg
       <chr>   <dbl>
1     Friday 21951.0
2     Monday 13982.0
3   Saturday 22074.5
4     Sunday 15224.5
5   Thursday 17580.0
6    Tuesday 14775.5
7  Wednesday 15672.5
```

Data Summarization/Visualization

- 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.
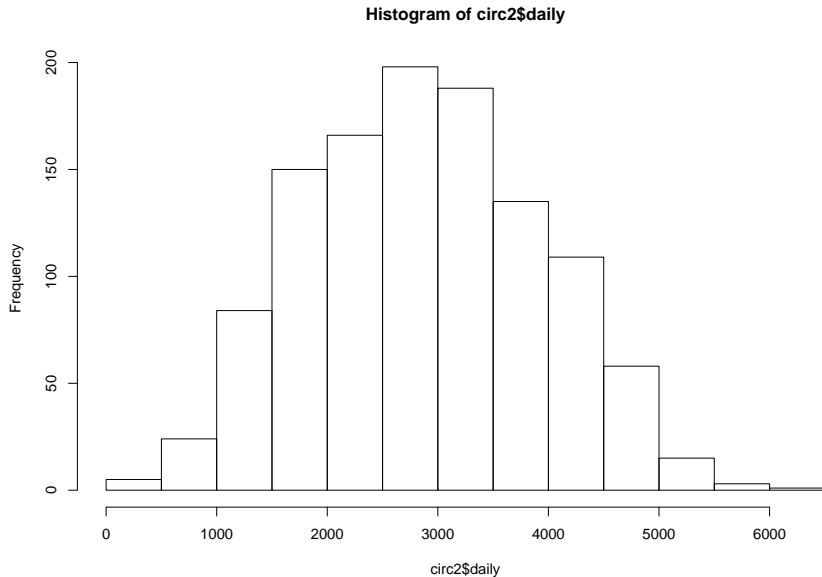We will go over formatting and making plots look nicer in additional
lectures.

Scatterplot

```
plot(mtcars$mpg, mtcars$disp)
```
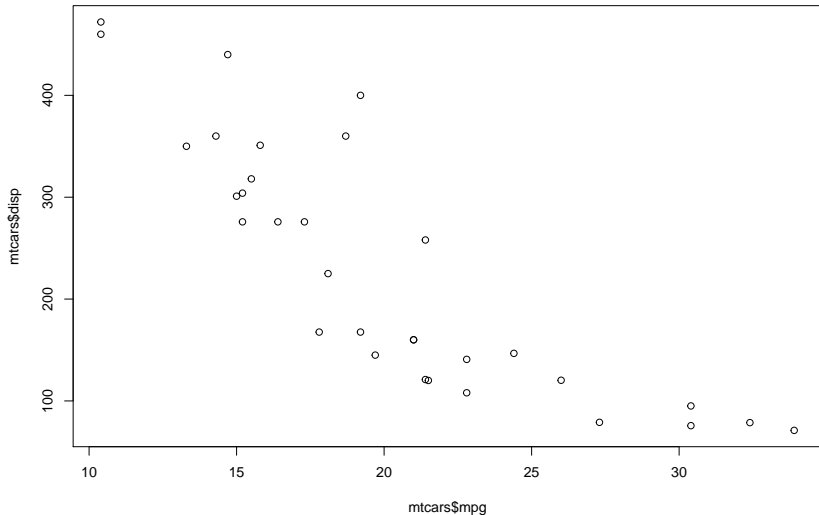
Histograms

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


Histogram of circ2$daily

Plot with a line

type = "l" means a line

```
library(lubridate)
```

```
Attaching package: 'lubridate'

The following object is masked from 'package:base':

    date
```
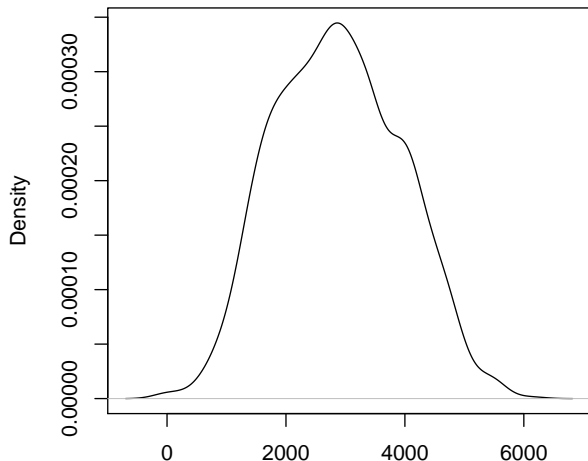
```
circ2$date = mdy(circ2$date)
plot(circ2$date, circ2$daily, type = "l")
```
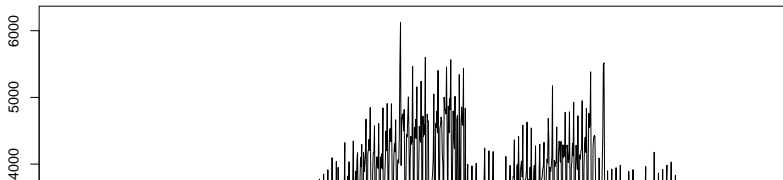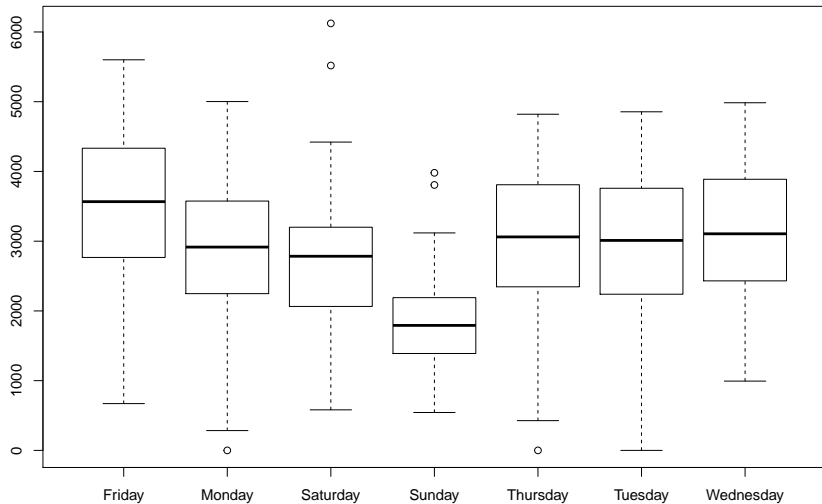
Density

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

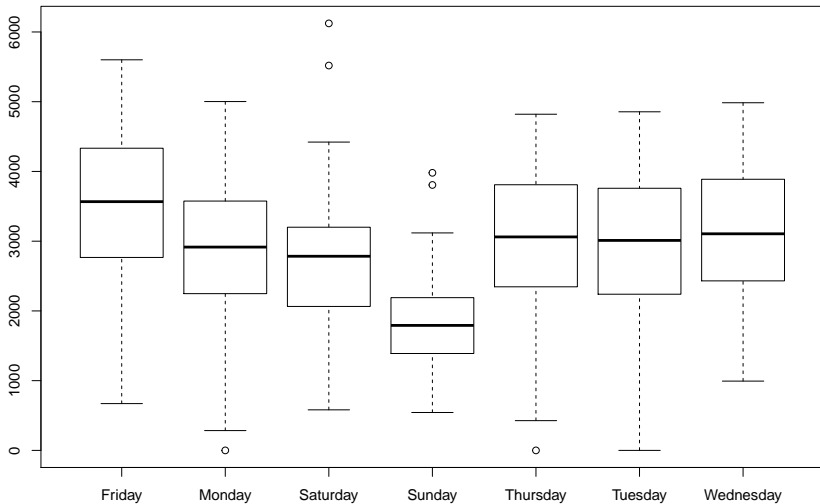**density.default(x = circ2$daily, na.rm = TRUE)**

Boxplots

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

Boxplots

```
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

```
pairs(avgs)
```