Dataframes

Dataframes - "Chapter Check-In"

Activity	Solution
Creating	read.table, data.frame, as.data.frame (to convert matrices)
Editing	Workspace viewer in RStudio
Meta Info:	rownames, names, nrow, ncol, sapply
Indexing:	Use bracket notation, subset command, or split command
Transform:	Use transform command, rbind, cbind, or $ notation to create new columns
Missing Values:	Use complete.cases to find only complete cases
Combining:	Use cbind, rbind, or merge
Summarizing:	Use summary, colmeans, rowmeans, (make sure you are dealing with numeric)
Factors:	Use factor command or leave as character until you need the factor
Sort:	Use the order function or rank function

Dataframes - Creating

A **data frame** is a special type of list that contains data in a format that allows for easier manipulation, reshaping, and open-ended analysis.

Data frames are tightly coupled collections of variables. It is one of the more important constructs you will encounter when using R so learn all you can about it.

A data frame is an analogue to the Excel spreadsheet. In general this is the most popular construct for storing, manipulating, and analyzing data.

Data frames can be constructed from existing vectors, lists, or matrices. Many times they are created by reading in comma delimited files, (CSV files), using the read.table command.

Once you become accustomed to working with data frames, R becomes so much easier to use.

Dataframes

Here we have 4 vectors two of which are character and two of which are numeric. We could work with them in the following fashion if we wanted to do some type of summary on them.

```
names = c("P1","P2","P3","P4","P5")
temp = c(98.2,101.3,97.2,100.2,98.5)
pulse = c(66,72,83,85,90)
gender = c("M","F","M","M","F")

# We could write a for loop to get information for each patient but this isn't so
convenient or scalable.

for (ii in 1:length(gender)) {
   print.string = c(names[ii],temp[ii],pulse[ii],gender[ii])
   print(print.string)
}

[1] "P1"   "98.2" "66"   "M"
[1] "P2"    "101.3" "72"    "F"
[1] "P3"   "97.2" "83"   "M"
[1] "P4"    "100.2" "85"    "M"
[1] "P5"   "98.5" "90"   "F"
```

Dataframes

A data frame can be regarded as a matrix with columns possibly of differing modes and attributes. It may be displayed in matrix form, and its rows and columns extracted using matrix indexing conventions. Let's create a data frame:

```
names=c("P1","P2","P3","P4","P5")
temp=c(98.2,101.3,97.2,100.2,98.5)
pulse=c(66,72,83,85,90)
gender=c("M","F","M","M","F")

my_df = data.frame(names,temp,pulse,gender) # Much more flexible
my_df
```

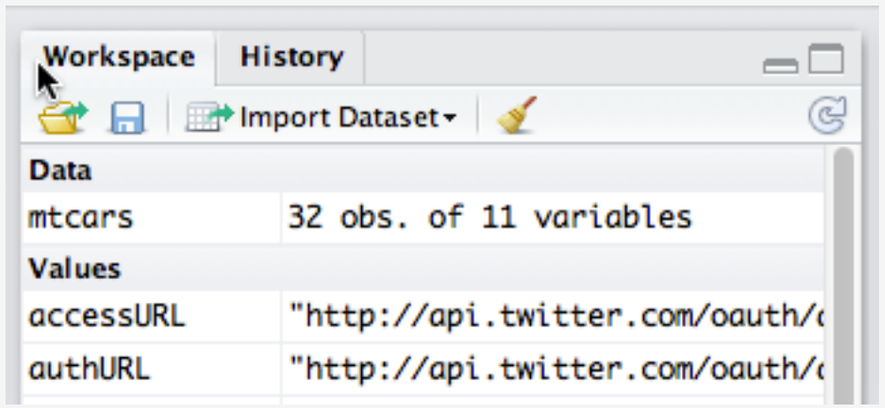
```
  names  temp pulse gender
1    P1  98.2    66      M
2    P2 101.3    72      F
3    P3  97.2    83      M
4    P4 100.2    85      M
5    P5  98.5    90      F
```

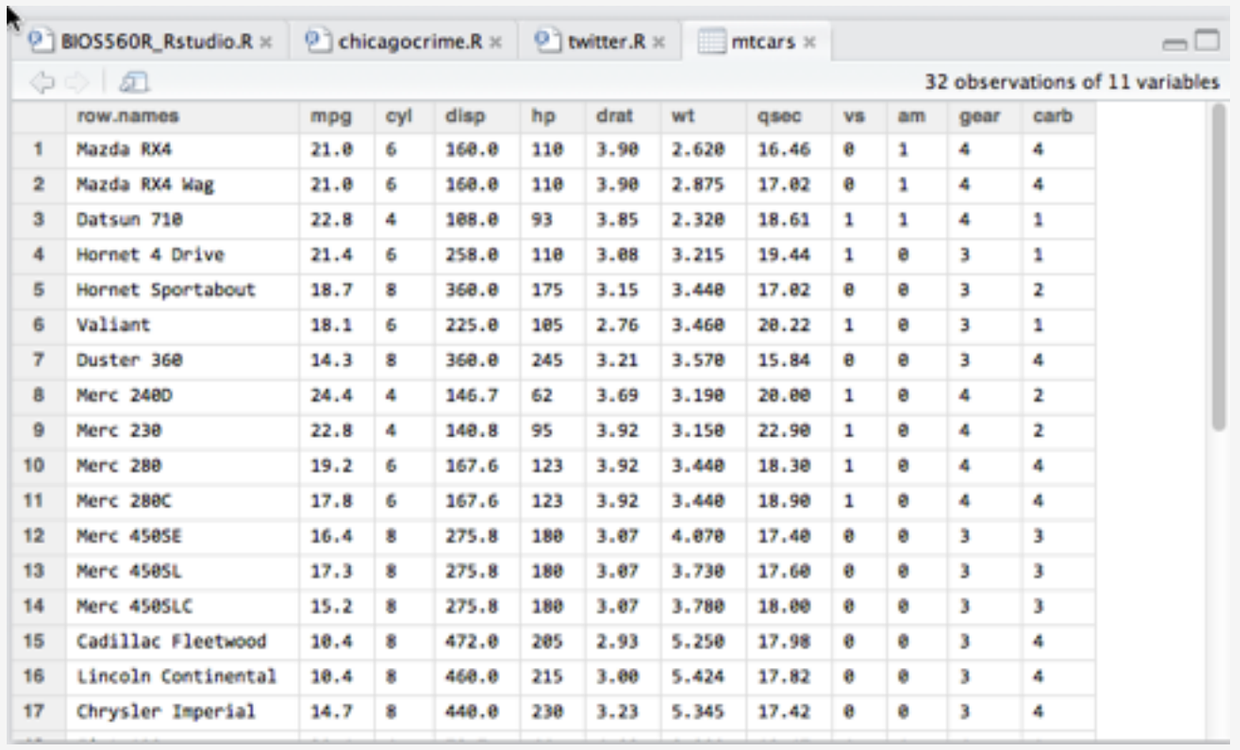
```
plot(my_df$pulse ~ my_df$temp,main="Pulse Rate",xlab="Patient",ylab="BPM")

mean(my_df[,2:3])
 temp pulse
99.08 79.20
```

Dataframes

Once you have the data frame you could edit it with a GUI editor. Or you can use the Workspace Viewer/Editor in RStudio

```
data(mtcars)   # This will load a copy of mtcars into your workspace.
```





	row.names	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
2	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
3	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
4	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
5	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
6	Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
7	Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
8	Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
9	Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
10	Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
11	Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
12	Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
13	Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
14	Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
15	Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
16	Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
17	Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4

If you are using the basic R console then you can type:

```
fix(mtcars)   # or whatever dataframe you are editing
```

Dataframes

R comes with a variety of built-in data sets that are very useful for getting used to data sets and how to manipulate them.

```
library(help="datasets")

# Gives detailed descriptions on available data sets

AirPassengers           Monthly Airline Passenger Numbers 1949-1960
BJsales                 Sales Data with Leading Indicator
BOD                     Biochemical Oxygen Demand
CO2                     Carbon Dioxide Uptake in Grass Plants
ChickWeight             Weight versus age of chicks on different diets
DNase                   Elisa assay of DNase
EuStockMarkets          Daily Closing Prices of Major European Stock
                        Indices, 1991-1998
Formaldehyde            Determination of Formaldehyde
HairEyeColor            Hair and Eye Color of Statistics Students

help(mtcars)    # Get details on a given data set
```

Dataframes

```
data(mtcars)

str(mtcars)
'data.frame':  32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...

nrow(mtcars)   # How many rows does it have ?
[1] 32

ncol(mtcars)   # How many columns are there ?
[1] 11

sapply(mtcars, class)  # Equivalent to above
```

Dataframes

There are a number of functions that provide metadata about a data frame.

```
rownames(mtcars)
 [1] "Mazda RX4"           "Mazda RX4 Wag"       "Datsun 710"
 [4] "Hornet 4 Drive"      "Hornet Sportabout"   "Valiant"
 ..
[19] "Honda Civic"         "Toyota Corolla"      "Toyota Corona"
[22] "Dodge Challenger"    "AMC Javelin"         "Camaro Z28"
[25] "Pontiac Firebird"    "Fiat X1-9"           "Porsche 914-2"
[28] "Lotus Europa"        "Ford Pantera L"      "Ferrari Dino"
[31] "Maserati Bora"       "Volvo 142E"

rownames(mtcars) = 1:32

head(mtcars)
   mpg cyl disp  hp drat   wt qsec vs transmission gear carb
1 21.0   6  160 110 3.90 2.62 16.5  0            1    4    4
2 21.0   6  160 110 3.90 2.88 17.0  0            1    4    4

rownames(mtcars) = paste("car",1:32,sep="_")
head(mtcars)
       mpg cyl disp  hp drat   wt qsec vs transmission gear carb
car_1 21.0   6  160 110 3.90 2.62 16.5  0            1    4    4
car_2 21.0   6  160 110 3.90 2.88 17.0  0            1    4    4
car_3 22.8   4  108  93 3.85 2.32 18.6  1            1    4    1
```

Dataframes

There are a number of functions that provide metadata about a data frame.

```
names(mtcars)     # See the column names/attributes/variables

 [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
[11] "carb"
```

Note that you could change the names of a given column if you wanted to. Let's say that you wanted to change the "am" column to "transmission". This corresponds to the ninth element the names vector.

```
my.names = names(mtcars)

my.names[9] = "transmission"

names(mtcars) = my.names

names(mtcars)
"mpg"          "cyl"          "disp"         "hp"           "drat"         "wt"
"qsec"         "vs"           "transmission" "gear"         "carb"
```

Dataframes

There are various ways to **select, remove, or exclude** rows and columns from a data frame.

```
mtcars[,-11]
                   mpg cyl disp  hp drat    wt  qsec vs am gear
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4

mtcars     # Notice that carb is included
                    mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4          21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag      21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710         22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1

mtcars[,-3:-5] # Print all columns except for columns 3 through 5
                    mpg cyl    wt  qsec vs am gear      carb
Mazda RX4          21.0   6 2.620 16.46  0  1    4 0.6020600
Mazda RX4 Wag      21.0   6 2.875 17.02  0  1    4 0.6020600
Datsun 710         22.8   4 2.320 18.61  1  1    4 0.0000000

mtcars[,c(-3,-5)] # Print all columns except for colums 3 AND 5
                    mpg cyl  hp    wt  qsec vs am gear      carb
Mazda RX4          21.0   6 110 2.620 16.46  0  1    4 0.6020600
Mazda RX4 Wag      21.0   6 110 2.875 17.02  0  1    4 0.6020600
Datsun 710         22.8   4  93 2.320 18.61  1  1    4 0.0000000
```

Dataframes

There are various ways to **select, remove, or exclude** rows and columns from a data frame.

```
mtcars[mtcars$mpg >= 30.0,]
                mpg cyl disp  hp drat    wt  qsec vs am gear carb
Fiat 128       32.4   4 78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic    30.4   4 75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla 33.9   4 71.1  65 4.22 1.835 19.90  1  1    4    1
Lotus Europa   30.4   4 95.1 113 3.77 1.513 16.90  1  1    5    2

mtcars[mtcars$mpg >= 30.0,2:6]
                mpg cyl disp  hp drat
Fiat 128       32.4   4 78.7  66 4.08
Honda Civic    30.4   4 75.7  52 4.93
Toyota Corolla 33.9   4 71.1  65 4.22
Lotus Europa   30.4   4 95.1 113 3.77

mtcars[mtcars$mpg >= 30.0 & mtcars$cyl < 6,]
                mpg cyl disp  hp drat    wt  qsec vs am gear carb
Fiat 128       32.4   4 78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic    30.4   4 75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla 33.9   4 71.1  65 4.22 1.835 19.90  1  1    4    1
Lotus Europa   30.4   4 95.1 113 3.77 1.513 16.90  1  1    5    2
```

Dataframes

Find all rows that correspond to Automatic and Count them

```
mtcars[mtcars$am==0,]
                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
..

nrow(mtcars[mtcars$am == 0,])
[1] 19

nrow(mtcars[mtcars$am == 1,])
[1] 13
```

Dataframes

Extract all rows whose MPG value exceeds the mean MPG for the entire data frame

```
> mtcars[mtcars$mpg > mean(mtcars$mpg),]
                mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4      21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag  21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710     22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive 21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Merc 240D      24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230       22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Fiat 128       32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic    30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Toyota Corona  21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
Fiat X1-9      27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
>
```

Dataframes

```
# Find the quartiles for the MPG vector

quantile(mtcars$mpg)
    0%    25%    50%    75%   100%
10.400 15.425 19.200 22.800 33.900

# Now find the cars for which the MPG exceeds the 75% value:

mtcars[mtcars$mpg > quantile(mtcars$mpg)[4],]
                mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Merc 240D      24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Fiat 128       32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic    30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Fiat X1-9      27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
```

Dataframes

There is an alternative to the bracket notation. It is called the subset function.

```
subset(mtcars, mpg >= 30.0)   # Get all records with MPG > 30.0

                mpg cyl disp  hp drat    wt  qsec vs am gear carb
Fiat 128       32.4   4 78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic    30.4   4 75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla 33.9   4 71.1  65 4.22 1.835 19.90  1  1    4    1
Lotus Europa   30.4   4 95.1 113 3.77 1.513 16.90  1  1    5    2

subset(mtcars, mpg >= 30.0, select=c(mpg:drat) )  # Get just columns mpg-drat

                mpg cyl disp  hp drat
Fiat 128       32.4   4 78.7  66 4.08
Honda Civic    30.4   4 75.7  52 4.93
Toyota Corolla 33.9   4 71.1  65 4.22
Lotus Europa   30.4   4 95.1 113 3.77

subset(mtcars, mpg >= 30.0 & cyl < 6 ) # Get all records with MPG >=30 and cyl <6

                mpg cyl disp  hp drat    wt  qsec vs am gear carb
Fiat 128       32.4   4 78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic    30.4   4 75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla 33.9   4 71.1  65 4.22 1.835 19.90  1  1    4    1
Lotus Europa   30.4   4 95.1 113 3.77 1.513 16.90  1  1    5    2
```

Dataframes

Many times data will be read in from a comma delimited ,("CSV"), file exported from Excel. The file can be read from local storage or from the Web.

```
url = "http://www.bimcore.emory.edu/BIOS560R/DATA.DIR/hsb2.csv"

data1 = read.table(url,header=T,sep=",")

head(data1)
  gender  id race ses schtyp  prgtype read write math science socst
1      0  70    4   1      1  general   57    52   41      47    57
2      1 121    4   2      1   vocati   68    59   53      63    61
3      0  86    4   3      1  general   44    33   54      58    31
4      0 141    4   3      1   vocati   63    44   47      53    56
5      0 172    4   2      1 academic   47    52   57      53    61
6      0 113    4   2      1 academic   44    52   51      63    61
```

Dataframes

Back to the mtcars data frame. What columns appear to be candidates for a factor ?
It would be variables who have only "a few" number of different values. If we do something like this we can get an idea. Looks like the last 4 columns might be what they want.

```
str(mtcars)
'data.frame':  32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...

unique(mtcars$am)   # Tells us what the unique values are
[1] 1 0
```

Dataframes

See how many unique values each columns takes on. Potential factors are in red.

```
sapply(mtcars, function(x) length(unique(x)))
 mpg  cyl disp   hp drat   wt qsec   vs   am gear carb
  25    3   27   22   22   29   30    2    2    3    6

mtcars$am = factor(mtcars$am, levels = c(0,1), labels = c("Auto","Man") )

str(mtcars$am)
Factor w/ 2 levels "Auto","Man": 2 2 2 1 1 1 1 1 1 1 ...

head(mtcars,5)
                   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  Man    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  Man    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  Man    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1 Auto    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0 Auto    3    2
```

Dataframes

And we can do some aggregation and summary directly on the data frame.

```
tapply(mtcars$mpg, mtcars$am,mean)
    Auto      Man
17.14737 24.39231

tapply(mtcars$mpg,mtcars$am,quantile)
$Auto
   0%   25%   50%   75%  100%
10.40 14.95 17.30 19.20 24.40

$Man
  0%  25%  50%  75% 100%
15.0 21.0 22.8 30.4 33.9
```

We will investigate some more powerful aggregation functions in a later session

Dataframes

We can also easily add columns to a data frame. Let's say we have a 31 element vector called "myrate" that we want to put into our data frame. "G","B","O" stands for "Good","Bad","Okay". There are a couple of ways to do this:

```
myrate
 [1] "B" "G" "G" "G" "B" "G" "G" "G" "B" "O" "B" "O" "B" "B" "O" "G" "B" "G" "G"
[20] "G" "B" "G" "B" "B" "G" "B" "O" "B" "B" "O" "B" "O"

mtcars = cbind(mtcars,myrate)
                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb myrate
Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4    B
Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4    G
Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1    G
Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1    G

-OR more simply-

mtcars$myrate = myrate    # The column just shows up
```

Dataframes

You can also use the **transform()** command to change the types/classes of the columns

```
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
```

```
transform(mtcars,wt = (wt*1000), qsec = round(qsec), am = factor(am,labels=c("A","M")))
```

```
                    mpg cyl  disp  hp drat   wt qsec vs am gear carb
Mazda RX4          21.0   6 160.0 110 3.90 2620   16  0  M    4    4
Mazda RX4 Wag      21.0   6 160.0 110 3.90 2875   17  0  M    4    4
Datsun 710         22.8   4 108.0  93 3.85 2320   19  1  M    4    1
Hornet 4 Drive     21.4   6 258.0 110 3.08 3215   19  1  A    3    1
Hornet Sportabout  18.7   8 360.0 175 3.15 3440   17  0  A    3    2
```

Dataframes - missing values

The **NA (datum Not Available)** is R's way of dealing with missing data. NAs can give you trouble unless you explicitly tell functions to ignore them, or pass the data through na.omit() (drop all NAs in the data), na.exclude() or complete.cases(). In some cases you may wish to give the NAs a specific value. Use **na.omit()** to eliminate missing data from a data set.

```
data <- data.frame(x=c(1,2,3,4), y=c(5, NA, 8,3),z=c("F","M","F","M"))
```

```
data
  x  y z
1 1  5 F
2 2 NA M          # Note missing value
3 3  8 F
4 4  3 M
```

Dataframes - missing values

```
complete.cases(data)
[1]  TRUE FALSE  TRUE  TRUE


sum(complete.cases(data))  # total number of complete cases
[1] 3


sum(!complete.cases(data)) # total number of incomplete cases
[1] 1


data[complete.cases(data),]
  x y z
1 1 5 F
3 3 8 F
4 4 3 M
```

Dataframes - missing values

```
url = "http://homepages.wmich.edu/~hgv7680/data/SAS/hs0.csv"
data1 = read.table(url,header=F,sep=",")
names(data1) = c("gender","id","race","ses","schtyp","prgtype",
                 "read", "write","math","science","socst")

head(data1, n=3)
  gender  id race ses schtyp  prgtype read write math science socst
1      0  70    4   1      1  general   57    52   41      47    57
2      1 121    4   2      1   vocati   68    59   53      63    61
3      0  86    4   3      1  general   44    33   54      58    31

nrow(data1)
[1] 200

sum(complete.cases(data1))
[1] 195

sum(!complete.cases(data1))
[1] 5

data1[!complete.cases(data1),]
   gender  id race ses schtyp  prgtype read write math science socst
9       0  84    4   2      1  general   63    57   54      NA    51
18      0 195    4   2      2  general   57    57   60      NA    56
37      0 200    4   2      2 academic   68    54   75      NA    66
55      0 132    4   2      1 academic   73    62   73      NA    66
76      0   5    1   1      1 academic   47    40   43      NA    31
```

Dataframes - missing values

Many R functions have a way to exclude missing values.

```
 data1[!complete.cases(data1),]
   gender  id race ses schtyp  prgtype read write math science socst
9       0  84    4   2      1  general   63    57   54      NA    51
18      0 195    4   2      2  general   57    57   60      NA    56
37      0 200    4   2      2 academic   68    54   75      NA    66
55      0 132    4   2      1 academic   73    62   73      NA    66
76      0   5    1   1      1 academic   47    40   43      NA    31


> mean(data1$science)
[1] NA

> mean(data1$science,na.rm=T)
[1] 51.66154
```

Supplemental Dataframes - more involved

Missing values can be set by using correlations between variables or by using the most frequent value for that column, mean or median, similarity or correlations with other variables. There are other possibilities of course.

```
# Using the median

> data1$science = ifelse(is.na(data1$science),
                  median(data$science,na.rm=T),data1$science)
```

Sometimes we can look to see if the variable that has missing values is strongly correlated with another variable, which, in turn, could be used to predict a value.

```
> cor(data1[,c(7:11)],use="complete.obs")

             read     write      math   science     socst
read    1.0000000 0.5967765 0.6622801 0.3665406 0.6214843
write   0.5967765 1.0000000 0.6174493 0.4160699 0.6047932
math    0.6622801 0.6174493 1.0000000 0.3635822 0.5444803
science 0.3665406 0.4160699 0.3635822 1.0000000 0.3239351
socst   0.6214843 0.6047932 0.5444803 0.3239351 1.0000000
```

Supplemental Dataframes - more involved

The strongest correlation for science and another variable is 0.41, which corresponds to writing. This isn't so strong actually but it's the best we have here.

```
             read     write      math   science     socst
read    1.0000000 0.5967765 0.6622801 0.3665406 0.6214843
write   0.5967765 1.0000000 0.6174493 0.4160699 0.6047932
math    0.6622801 0.6174493 1.0000000 0.3635822 0.5444803
science 0.3665406 0.4160699 0.3635822 1.0000000 0.3239351
socst   0.6214843 0.6047932 0.5444803 0.3239351 1.0000000

> ( my.lm = lm(science ~ write,data1) )

Call:
lm(formula = science ~ write, data = data1)

Coefficients:
(Intercept)        write
    20.7840       0.5601
```

Assuming this mode is any good (and that is a very big "if") then we could use the equation
missing_science_val = 0.56*write + 20.7840

Supplemental Dataframes - more involved

Assuming this mode is any good (and that is a very big “if”) then we could use the equation:

missing_science_val = 0.56*write + 20.7840

```
> summary(my.lm)
Call:
lm(formula = science ~ write, data = data1)

Residuals:
    Min      1Q  Median      3Q     Max
-56.512  -4.060   0.350   6.698  28.251

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  20.7841     4.6645   4.456 1.40e-05 ***
write         0.5601     0.0870   6.438 8.94e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 11.63 on 198 degrees of freedom
Multiple R-squared: 0.1731,    Adjusted R-squared: 0.1689
F-statistic: 41.45 on 1 and 198 DF,  p-value: 8.937e-10
```

Supplemental Dataframes - more involved

Assuming this mode is any good (and that is a very big “if”) then we could use the equation:

missing_science_val = 0.56*write + 20.7840

```
> my.write.vals = data1[!complete.cases(data1),"write"]

> my.write.vals
[1] 57 57 54 62 40

> predict(my.lm,data.frame(write=my.vals),interval="predict")
       fit      lwr      upr
1 52.71156 29.70278 75.72033
2 52.71156 29.70278 75.72033
3 51.03116 28.03285 74.02948
4 55.51222 32.46047 78.56396
5 43.18932 20.08776 66.29087

> my.pred.science = predict(my.lm,data.frame(write=my.vals),interval="predict")

> my.pred.science[,1]
       1        2        3        4        5
52.71156 52.71156 51.03116 55.51222 43.18932
```

Supplemental Dataframes - more involved

Assuming this mode is any good (and that is a very big "if") then we could use the equation:
missing_science_val = 0.56*write + 20.7840

```
> ( for.replace = which(!complete.cases(data1)) )
[1]  9 18 37 55 76

> data1[ for.replace ,]
   gender  id race ses schtyp  prgtype read write math science socst
9       0  84    4   2      1  general   63    57   54      NA    51
18      0 195    4   2      2  general   57    57   60      NA    56
37      0 200    4   2      2 academic   68    54   75      NA    66
55      0 132    4   2      1 academic   73    62   73      NA    66
76      0   5    1   1      1 academic   47    40   43      NA    31


> data1[ for.replace, ]$science = my.fit[,1]


> data1[ for.replace, ]
   gender  id race ses schtyp  prgtype read write math  science socst
9       0  84    4   2      1  general   63    57   54 52.71156    51
18      0 195    4   2      2  general   57    57   60 52.71156    56
37      0 200    4   2      2 academic   68    54   75 51.03116    66
55      0 132    4   2      1 academic   73    62   73 55.51222    66
76      0   5    1   1      1 academic   47    40   43 43.18932    31
```

Dataframes: merge

```
tb1 = data.frame(indiv_id = 1:4, snp1 = c(1,1,0,1), snp2 = c(1,1,0,0))

tb2 = data.frame(indiv_id = c(1,3,4,6), cov1 = c(1.14,4.50,0.80,1.39),
                                          cov2 = c(74.6,79.4,48.2,68.1))

tb1
  indiv_id snp1 snp2
         1    1    1
         2    1    1
         3    0    0
         4    1    0

tb2
  indiv_id cov1 cov2
         1 1.14 74.6
         3 4.50 79.4
         4 0.80 48.2
         6 1.39 68.1

merge(tb1, tb2, by="indiv_id", all=TRUE)

  indiv_id SNP1 SNP2 cov1 cov2
         1    1    1 1.14 74.6
         2    1    1   NA   NA
         3    0    0 4.50 79.4
         4    1    0 0.80 48.2
         6   NA   NA 1.39 68.1
```

Dataframes: split

The split function lets us break up a data frame based on a grouping variable. We'll look at this in greater detail in the aggregate section but for now let's focus on how to do this.

Let's say we want to split up mtcars based on the number of cylinders which take on the values 4,6,8. We use the split command to do this and what it gives back to us is a list with each element containing a part of the data frame corresponding to each cylinder group.

We could use the subset command or bracket notation to pull out the information

```
eight.cyl = mtcars[mtcars$cyl == 8,]

six.cyl = mtcars[mtcars$cyl == 6, ]

four.cyl = mtcars[mtcars$cyl == 4, ]
```

Dataframes: split

```
(hold = split(mtcars, mtcars$cyl) )
hold
$`4`
                mpg cyl  disp hp drat    wt  qsec vs am gear carb
Datsun 710     22.8   4 108.0 93 3.85 2.320 18.61  1  1    4    1
Merc 240D      24.4   4 146.7 62 3.69 3.190 20.00  1  0    4    2
Merc 230       22.8   4 140.8 95 3.92 3.150 22.90  1  0    4    2
Fiat 128       32.4   4  78.7 66 4.08 2.200 19.47  1  1    4    1
Honda Civic    30.4   4  75.7 52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla 33.9   4  71.1 65 4.22 1.835 19.90  1  1    4    1

$`6`
                mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4      21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag  21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Hornet 4 Drive 21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Valiant        18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Merc 280       19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C      17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4

$`8`
                   mpg cyl  disp  hp drat   wt  qsec vs am gear carb
Hornet Sportabout  18.7   8 360.0 175 3.15 3.44 17.02  0  0    3    2
Duster 360         14.3   8 360.0 245 3.21 3.57 15.84  0  0    3    4
Merc 450SE         16.4   8 275.8 180 3.07 4.07 17.40  0  0    3    3
Merc 450SL         17.3   8 275.8 180 3.07 3.73 17.60  0  0    3    3
Merc 450SLC        15.2   8 275.8 180 3.07 3.78 18.00  0  0    3    3
Cadillac Fleetwood 10.4   8 472.0 205 2.93 5.25 17.98  0  0    3    4
```

Dataframes: split

```
(hold = split(mtcars, mtcars$cyl) )

lapply(hold,nrow)
$`4`
[1] 11

$`6`
[1] 7

$`8`
[1] 14
```

Why is this useful ? Well we might want to focus in on only the cars occupying a certain cylinder group while ignoring the rest. So if we wanted only the 8 cylinder cars:

```
eight.cyl = hold$`8`

-OR-

eight.cyl = hold[[3]]
```

Dataframes: split

We could also use this approach to do some summary reporting. This might seem advanced at this point but its good for you too see this kind of approach as it is common in R. This example gives is the mean MPG for each cylinder group:

```
(hold = split(mtcars,cyl) )
lapply(hold, function(x) mean(x$mpg))
```

We can use a more complicated function of our own design:

```
my.func <- function(x) {
    hold = x[x$am == 0,]
    retvec = c(mean=mean(hold$mpg),sd=sd(hold$mpg))
    return(retvec)
}

> lapply(hold, my.func)
$`4`
     mean        sd
22.900000  1.452584

$`6`
     mean        sd
19.125000  1.631717

$`8`
     mean        sd
15.050000  2.774396
```

Dataframes: split

You can "unsplit" the split list at any time using the "unsplit" function.

```
unsplit(hold,mtcars$cyl)
```

Note that we must use the original category that we first did the split with.

Dataframes - order/sort

Let's take a look at what the order command does. It returns the record/row numbers of the data frame from lowest MPG to highest. So record #15 must be the lowest MPG automobile in the set. And record #20 must have the highest MPG

```
order(mtcars$mpg)
 [1] 15 16 24  7 17 31 14 23 22 29 12 13 11  6  5 10 25 30  1  2  4 32 21  3  9
[26]  8 27 26 19 28 18 20


mtcars[15,]
                     mpg cyl disp  hp drat   wt  qsec vs am gear carb
Cadillac Fleetwood 10.4   8  472 205 2.93 5.25 17.98  0  0    3    4

mtcars[20,]
                mpg cyl disp hp drat    wt qsec vs am gear carb
Toyota Corolla 33.9   4 71.1 65 4.22 1.835 19.9  1  1    4    1
```

Dataframes - order/sort

Ordering and sorting data frames is an important technique

```
# sort by mpg (ascending)
newdata <- mtcars[order(mtcars$mpg),]
newdata
                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4

newdata <- mtcars[rev(order(mtcars$mpg)),]
newdata
                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2

# sort by mpg and cyl
newdata = mtcars[order(mtcars$mpg, mtcars$cyl),]
newdata
                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
```

Dataframes - sample

The sample() function is quite useful when you want to take, well, a sample of your data. You can sample with or without replacement. The basic function works as follows:

```
# Take a random sample of something - in this case a vector of numbers from 1 to 20
my_vec = 1:20

sample(my_vec,10,replace=TRUE)    # Repetition is possible
 [1]  3 20 16 14 16 10 18  7  7  6

sample(my_vec, 10, replace=TRUE)  # Different results each time
 [1]  5  1  2  2 19  8 20 11  3 19

sample(my_vec, 10, replace=FALSE) # Don't replace to insure unique numbers
 [1]  2  8  9  6 17 18  3  5 14 15

sample(1:20, 10, replace=FALSE)   # Short cut
 [1] 13  6  4 14  3 19 16 17 20 12
```

To sample from a data frame you will need to know the total number of records/observations. Use the **nrow()** function to get this. Then decide how many records you want to sample. You probably want only unique records in your sample so then set **replace=FALSE**

Dataframes - sample

Use the **sample( )** function to take a random sample of size n from a dataset.

```
# Take a random sample of size 10 from dataset mtcars
# Sample without replacement

my_records = sample(1:nrow(mtcars), 10, replace = FALSE)

my_records
 [1] 21  6  9 30 29 28  3 11 12  1

sample_of_ten = mtcars[my_records,]
sample_of_ten
                 mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Toyota Corona   21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
Valiant         18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Merc 230        22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Ferrari Dino    19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
Ford Pantera L 15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
Lotus Europa    30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
Datsun 710      22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Merc 280C       17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Merc 450SE      16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
Mazda RX4       21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
```

Dataframes - "Chapter Checkout"

Activity	Solution
Creating	read.table, data.frame, as.data.frame (to convert matrices)
Editing	Workspace viewer in RStudio
Meta Info:	rownames, names, nrow, ncol, sapply
Indexing:	Use bracket notation, subset command, or split command
Transform:	USe transform command, rbind, cbind, or $ notation to create new columns
Missing Values:	Use complete.cases to find only complete cases
Combining:	Use cbind, rbind, or merge
Summarizing:	Use summary, colmeans, rowmeans, (make sure you are dealing with numeric)
Factors:	Use factor command or leave as character until you need the factor
Sort:	Use the order function or rank function

Supplemental - Chicago Crime

The City of Chicago let's you download lots of different data for analysis.

https://data.cityofchicago.org/

Supplemental - Chicago Crime

I've put this on the class server if you want to download it and give it a whirl. This a about 82MB so don't try reading it over a home-based connection. Also, my laptop has 4GB of RAM. I suspect if you have 2GB of RAM on your laptop you will be okay but I cannot be sure. On campus it took about 1 minute for R to process it.

```
url = "http://www.bimcore.emory.edu/BIOS560R/DATA.DIR/chi_crimes.csv"

chi = read.table(url,header=T,sep=",")
```

I tried reading this file into Excel. While it ultimately loaded the file it took a long time and response was very slow on my laptop. Part of the problem is that Excel loads the whole thing for purposes of display when in reality it might not be necessary to see everything. In fact with 300K records it is impractical to want to see every record.

Supplemental - Chicago Crime

So I downloaded a .CSV file containing data for all reported crimes in the 2012 year.

```
system("ls -lh chi*")
-rw-r--r--@ 1 fender  staff    82M Sep 13 06:20 chi_crimes.csv

system("wc -l chi*")          # 334,142 lines !!
  334142 chi_crimes.csv

# It takes about 25 seconds to read this in on my laptop

system.time(mychi <- read.table("chi_crimes.csv",header=T,sep=","))
   user  system elapsed
 25.026   0.323  25.417

nrow(mychi)
[1] 334141

ncol(mychi)
[1] 22
```

Supplemental - Chicago Crime

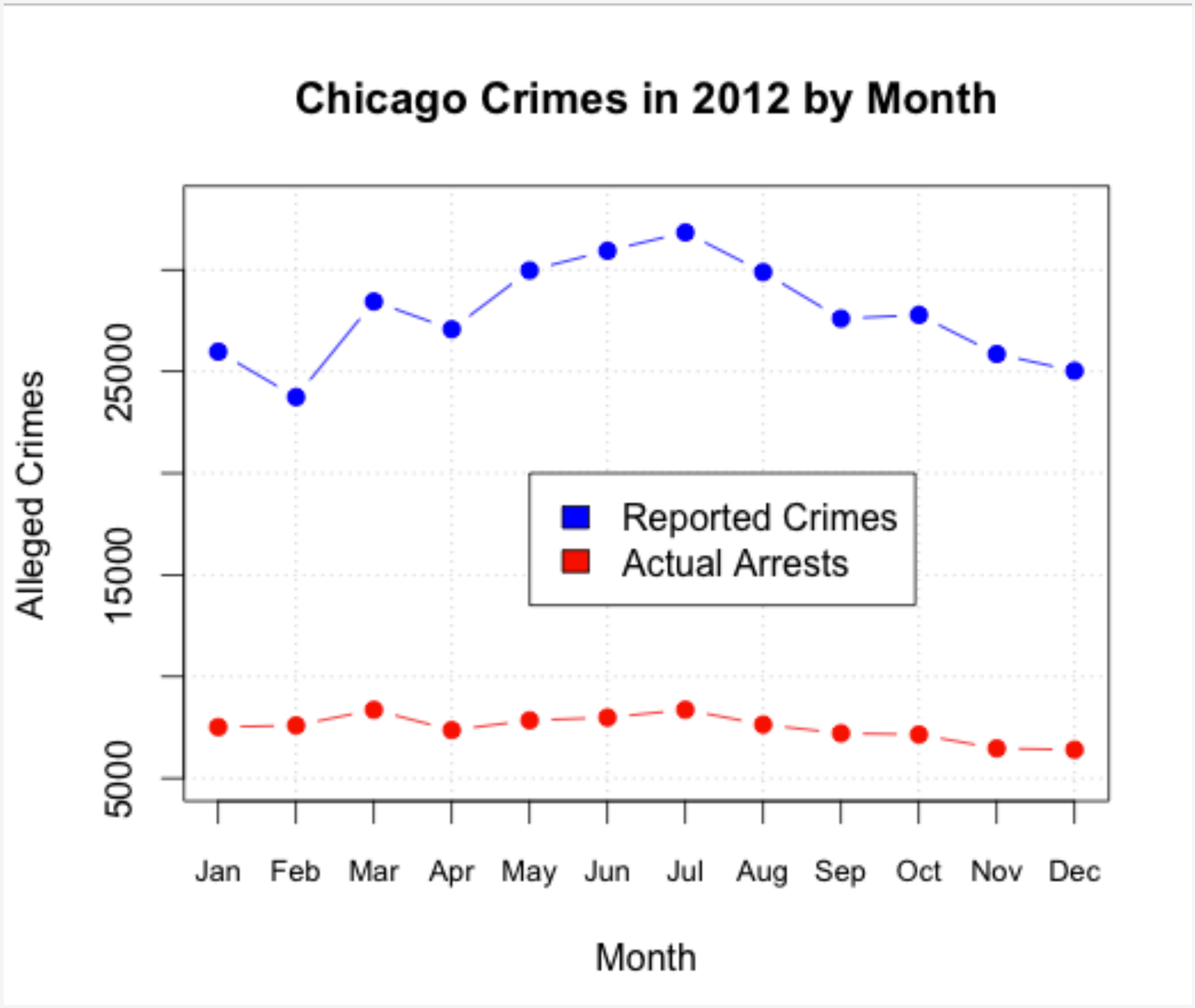
```
names(chi)
 [1] "Case.Number"          "ID"
 [3] "Date"                 "Block"
 [5] "IUCR"                 "Primary.Type"
 [7] "Description"          "Location.Description"
 [9] "Arrest"               "Domestic"
[11] "Beat"                 "District"
[13] "Ward"                 "FBI.Code"
[15] "X.Coordinate"         "Community.Area"
[17] "Y.Coordinate"         "Year"
[19] "Latitude"             "Updated.On"
[21] "Longitude"            "Location"
[23] "month"
```

```
sapply(chi, function(x) length(unique(x)))
         Case.Number                   ID                 Date
              334114               334139               121480
               Block                IUCR         Primary.Type
               28383                  358                   30
         Description Location.Description               Arrest
                 296                  120                    2
            Domestic                 Beat             District
                   2                  302                   25
                Ward             FBI.Code          X.Coordinate
                  51                   30                60704
      Community.Area         Y.Coordinate                 Year
                  79                89895                    1
            Latitude           Updated.On            Longitude
              180396                 1311               180393
            Location                month
              178534                   12
```

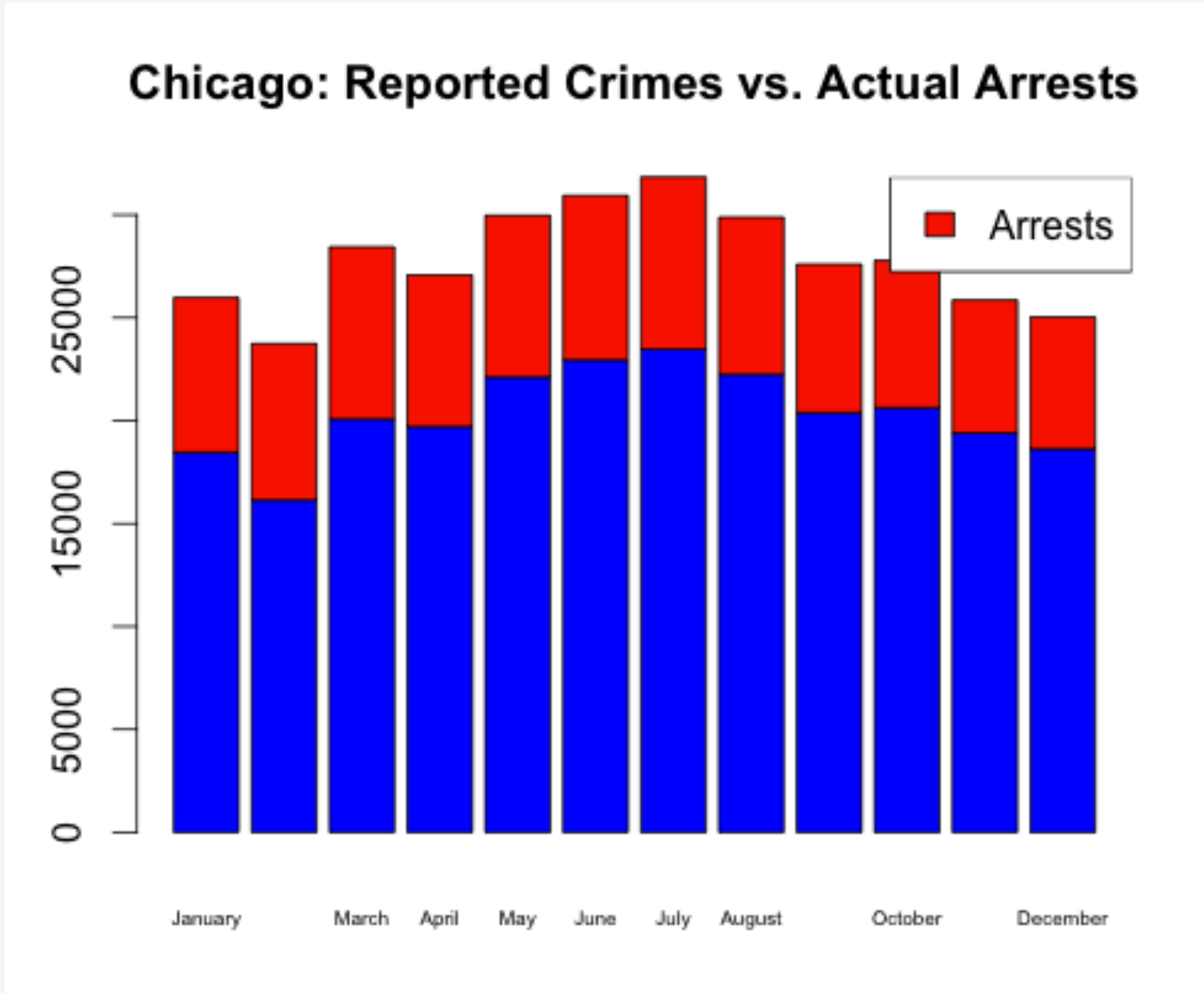
Supplemental - Chicago Crime

```
chi$Date = strptime(chi$Date,"%m/%d/%Y %r") # Change Dates from factor to a "real" Date
chi$month = months(chi$Date)
chi$month = factor(chi
$month,levels=c("January","February","March","April","May","June","July","August",
                "September","October","November","December"),ordered=TRUE)

# Okay how many crimes were committed in each Month of the year ?

plot(1:12,as.vector(table(chi$month)),type="n",xaxt="n",ylab="Alleged
Crimes",xlab="Month",main="Chicago Crimes in 2012 by Month",ylim=c(5000,33000))
grid()
axis(1,at=1:12,labels=as.character(sapply(levels(chi$month),
     function(x) substr(x,1,3))),cex.axis=0.8)
points(1:12,as.vector(table(chi$month)),type="b",pch=19,col="blue")
points(1:12,as.vector(table(chi$month,chi$Arrest)[,2]),col="red",pch=19,type="b")
legend(5,20000,c("Reported Crimes","Actual Arrests"),fill=c("blue","red"))

# Might look better in a barplot

barplot(table(chi$Arrest,chi$month),col=c("blue","red"),cex.names=0.5,main="Chicago: Reported
Crimes vs. Actual Arrests")
legend("topright",c("Arrests"),fill="red")


# Even easier to do

rev(sort(table(chi$month)))
barplot(rev(sort(hold)),horiz=F,las=1,cex.names=0.5,col=heat.colors(12),main="Chicago: Reported
Crimes in 2012 by Month")

# Looks like the Summer is when more crimes are committed
```

Supplemental - Chicago Crime

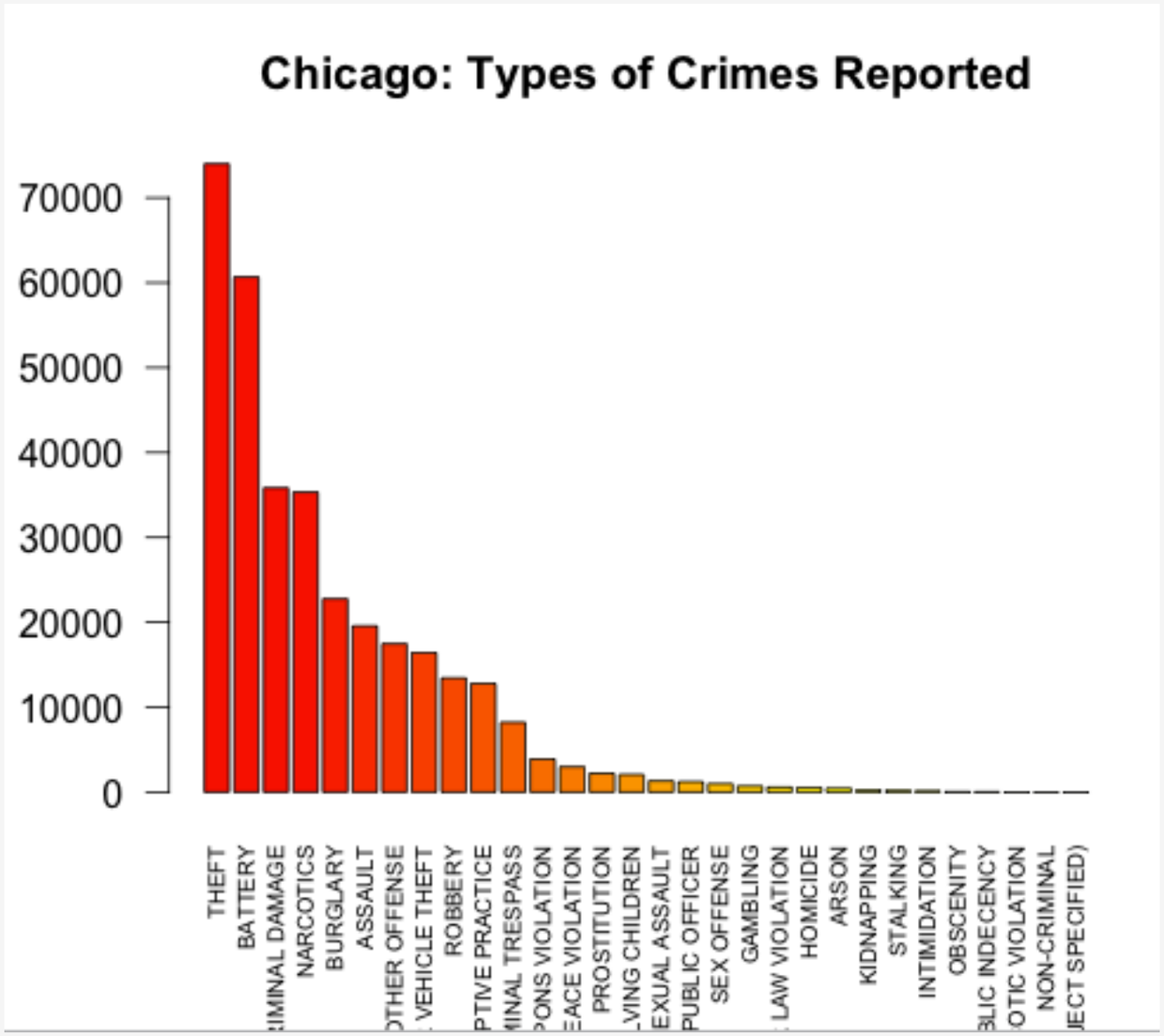


Supplemental - Chicago Crime



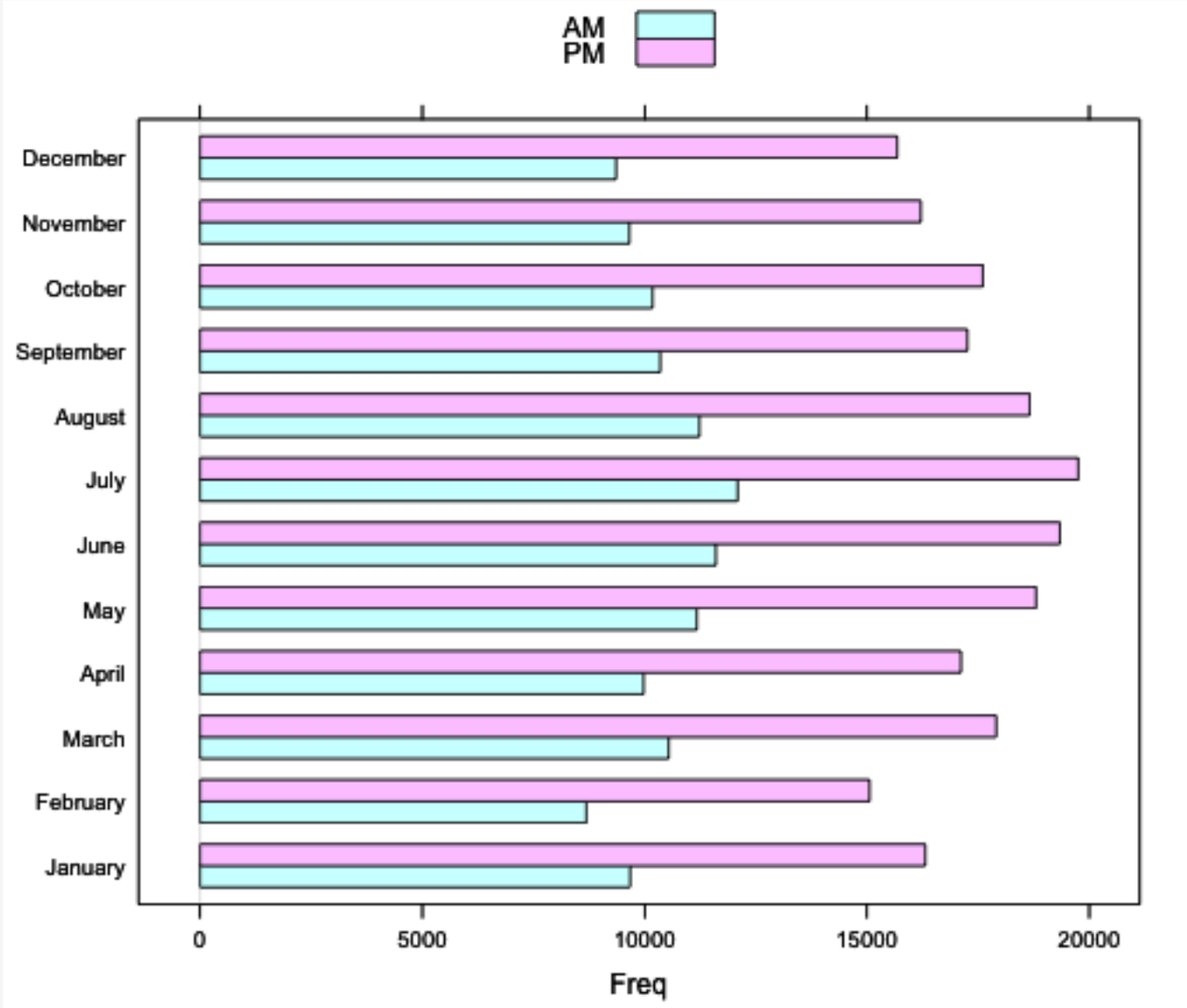
Supplemental - Chicago Crime

```
categories = rev(sort(sapply(unique(as.character(chi$Primary.Type)),
             function(x) { nrow(chi[chi$Primary.Type==x,]) })))

categories = rev(sort(table(chi$Primary.Type)))
barplot(categories,horiz=F,las=1,cex.names=0.6,col=heat.colors(30),las=2,
        main="Chicago: Types of Crimes Reported")
```



Supplemental - Chicago Crime

```
library(lattice)
barchart(table(chi$month,chi$ampm),stack=FALSE,auto.key=T,freq=F)
```



Supplemental - Chicago Crime

```
Let's map some of these reported crimes

# Let's zone in on the reported gambling offenses
# Most of these are for Dice games. Let's see the ones that are Gambling but not dice
related

hold = chi[chi$Primary.Type == "GAMBLING",]
hold = chi[chi$Primary.Type == "GAMBLING" & chi$Description != "GAME/DICE",]

nrow(hold) # How many non-Dice related gambling offenses were there ?
# About 26 I think
# Let's plot them on a map

library(googleVis)  # This is an addon package you must install

hold$LatLon = paste(hold$Latitude,hold$Longitude,sep=":")
hold$Tip = paste(hold$Description,hold$Locate.Description,hold$Block,"<BR>",sep=" ")
```

Supplemental - Chicago Crime