EXISTING DATA

1. **`data()`** shows the existing datasets
2. **`data(mtcars)`** to load packages

HELP

3. Searching how to read csv: `?csv` or `??csv`. Leads to `read.table()`. Look up `?read.table`. Use `read.csv()`. Then `example(read.csv)`.
4. `apropos("csv")` for all functions with csv in function name.

5. Also use ?"%%" to show use of quotes to look up help on operators (it's modulus)

DATATYPES

6. Vectors: lists of same datatype. `v <- c(1,2,3,4)` c is for combine.
7. Dataframe: a mixed-type matrix. Example, mtcars:

```
data.frame(
      session=c("lockpicking", "flying", "plotting"),
      person=c("grant", "arjun", "neha"),
      day = c('2015-05-12', '2015-05-01', '2015-05-22')
)
```

   a. field variables: `mtcars$cyl`

EXPLORING DATA

8. Indexing begins at 1
9. `NA` means unknown, `NULL` means non-existent. `?mean`
10. First thing to do with data, look at
    a. `summary(mtcars),`
    b. `nrow(mtcars), ncol(mtcars),`
    c. `head(mtcars), tail(mtcars)`
    d. `hist(mtcars$cyl)`
    e. `plot(mtcars$cyl, mtcars$hp)`
    f. `mean(), median(), etc.`
11. Modifying frame: deleting column by setting it null. `mtcars$mpg = NULL` poof.
12. rownames can be changed `rownames(df) <- c(...)`
13. `df[c(1)]` selects column, and `df[c(1),]` selects mentioned rows
14. `sort(df$col)` returns sorted column.
15. order df using: `a[order(a$day),]`
16. `merge using` merge(df1,df2, by=<fieldname)

```
# Merging
df1 = data.frame(
      session=c('lockpicking', 'flying', 'plotting'),
      person = c('grant', 'arjun', 'neha'),
```

```
      day = c('2015-05-12', '2015-05-01', '2015-05-22')
      )
df2 = data.frame(
      person=c('neha','arjun','grant'),
      adviser=c('geoff', 'alex', 'stefan')
)
merge(a,b, by='person')
```

PLOTTING:

1. Install/update:
   ```
   install.packages('ggplot2')
   update.packages('ggplot2')
   ```
2. Include:
   ```
   library(ggplot2)
   ```
3. `ggplot()` is the basic function.
   a. Takes the df and an aesthetic variables corresponding to geom_
   b. geom_ elements on graph. `??geom_`
4. basic histogram:
   ```
   ggplot(mtcars, aes(x=mpg)) + geom_histogram(binwidth=5)
   ```
5. 2 variable dot plot:
   ```
   ggplot(mtcars, aes(x=hp, y=mpg)) +
         geom_point()
   ```
6. Color encoding transmission type:
   ```
   ggplot(mtcars, aes(x=hp, y=mpg, color=factor(am))) +
         geom_point()
   ```
7. Color by wt and shape by transmission:
   ```
   ggplot(mtcars, aes(x=hp, y=mpg)) +
         geom_point(aes(color=wt, shape=factor(am)))  +
         geom_smooth()
   ```
8. multiple elements on plot (linear regression fitted)
   ```
   mtcars$pred.mpg <-  predict(lm(mpg~hp, data=mtcars))

   ggplot(mtcars, aes(x=hp, y=mpg)) +
         geom_point() +
         geom_line(aes(y=pred.mpg))
   ```

9. can add color of points according to weight:
   ```
   ggplot(mtcars, aes(x=hp, y=mpg)) +
         geom_point(aes(color=wt)) +
         geom_line(aes(y=pred.mpg))
   ```

10. can also get a smoothed model:

```
ggplot(mtcars, aes(x=hp, y=mpg)) +
      geom_point(aes(color=wt)) +
      geom_smooth()
```

11. geoms have "stats". there's stat="bin" in geom_bar. these correspond to `stat_bin` function, etc. there are many of them and we can usually change them. the stats corresponding to stat_bin has a binwidth parameter. so we can do `geom_bar(stat="bin", binwidth=10)`, etc.
12. scales: a lot of things (colors, labels, etc.) are actually scales. Look for all of them using `??scale_`

```
ggplot(mtcars, aes(x=factor(gear), y=mpg)) +
      geom_point(aes(color=wt)) +
      scale_x_discrete("number of gears",
            breaks = c(3, 4, 5),
            labels = c("three", "four", "five")) +
      scale_color_continuous("weight",
            breaks = c(min(mtcars$wt), median(mtcars$wt),
max(mtcars$wt)),
            labels = c("light", "medium", "heavy"),
            low="blue",
            high="red")
```

with point size changing according to wt:

```
ggplot(mtcars, aes(x=factor(gear), y=mpg)) +
     geom_point(aes(size=wt)) +
      scale_x_discrete("number of gears",
            breaks = c(3, 4, 5),
            labels = c("three", "four", "five")) +
      scale_size_continuous("weight",
            breaks = c(min(mtcars$wt), median(mtcars$wt),
max(mtcars$wt)),
            labels = c("light", "medium", "heavy"))
```

DATA MANIPULATION

`plyr`: split-apply-combine.

`summarize` returns a new df with specified columns:

```
summarize(mtcars, mpg.cyl.avg=mpg/cyl, hp.cyl.avg=hp/cyl, am=am,
gear=gear)
```

`ddply` does split-apply-combine on dfs

```
td<-ddply(mtcars, "cyl", summarize, hp.cyl.avg = hp/cyl, am=am,
gear=gear)
```

FACETING

```
ggplot(td, aes(x=cyl, y=hp.cyl.avg)) +
       geom_point() +
       facet_wrap(~ am)

ggplot(td, aes(x=cyl, y=hp.cyl.avg)) +
       geom_point() +
       facet_grid(gear ~ am)
```

THEMES

for all the background stuff -- plot background, axes, legends, facet labels, etc.

`theme_gray()` is default. `theme_bw()` is nicer

```
ggplot(mtcars, aes(x=factor(gear), y=mpg)) +
     geom_point(aes(color=wt)) +
       scale_x_discrete("number of gears",
           breaks = c(3, 4, 5),
           labels = c("three", "four", "five")) +
       scale_color_continuous("weight",
           breaks = c(min(mtcars$wt), median(mtcars$wt),
max(mtcars$wt)),
           labels = c("light", "medium", "heavy"),
           low="red",
           high="blue") +
theme_gray()
```

then start tweaking. for plot background:

```
theme(plot.background = element_rect(fill="yellow"))
```

text pieces modified with element_text, shapes modified with element_rect.

```
theme(axis.text.x = element_text(color="red"),
     panel.background = element_rect(fill="lightblue"))
```

suppose we want to plot 2 variables (columns) in df against a third column and want to change the color depending on these 2 variables. we double the data by melting it: length(rownames(mtcars)): 32

```
library(reshape2)
mtc.m <- melt(mtcars, measure.vars=c("mpg", "hp"))
# adds 2 columns: “value” the value to be plotted, “variable” the
variable we are plotting.
ggplot(mtc.m , aes(x=wt, y=value, color=variable)) + geom_point()
```