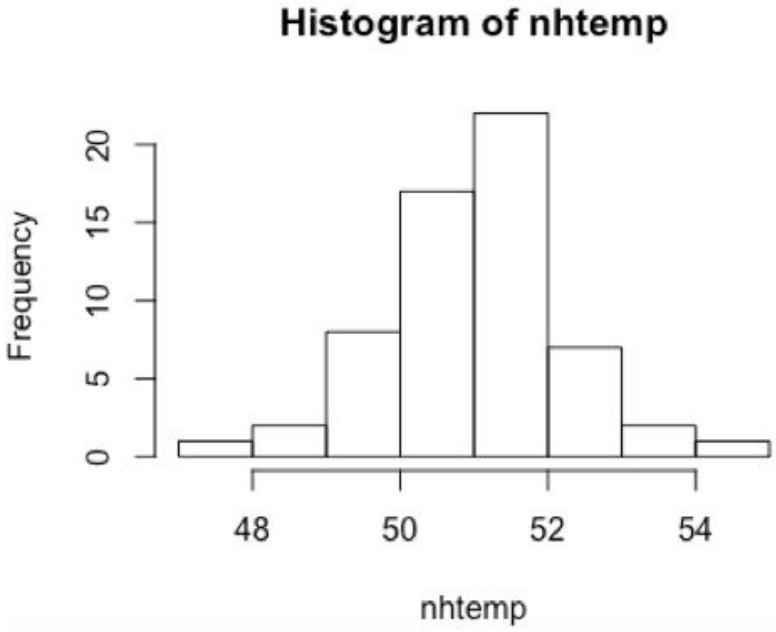
The Basics of Plotting in R

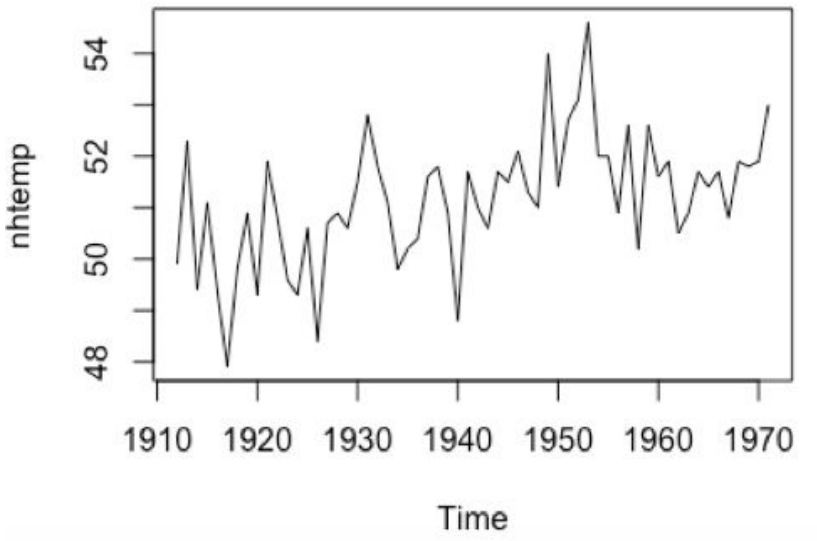
R has a built-in Datasets Package:

- iris
- mtcars
- precip
- faithful
- state.x77
- USArrests
- presidents
- ToothGrowth
- USJudgeRatings

You can call built-in functions like `hist()` or `plot()` on a built-in data set to quickly produce a chart

Basic plotting in R



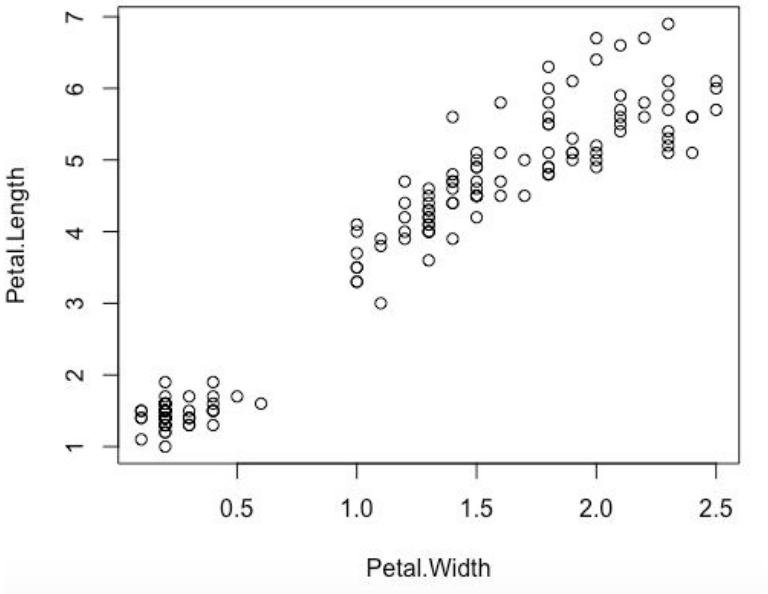


`hist(nhtemp)`

`plot(nhtemp)`

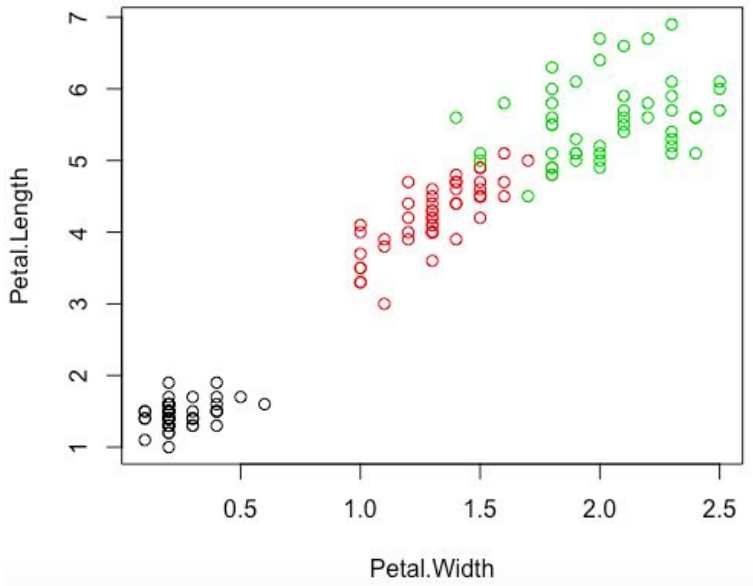
Basic plotting in R

```
plot(Petal.Length ~ Petal.Width, data = iris)
```



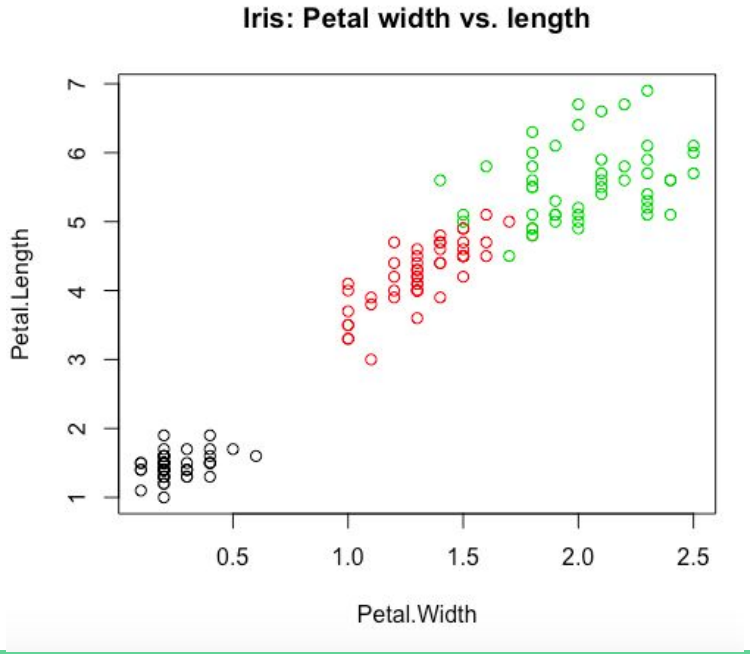
Basic plotting in R

```
plot(Petal.Length ~ Petal.Width, col = Species, data = iris)
```



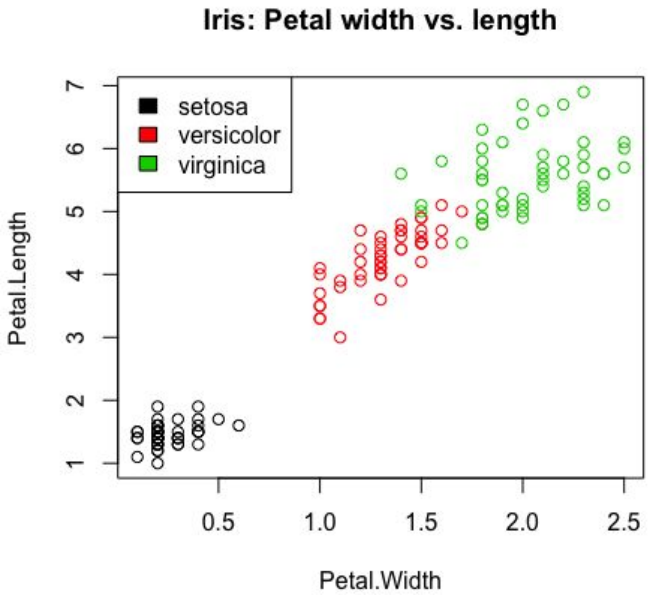
Basic plotting in R

```
plot(Petal.Length ~ Petal.Width, col = Species, main =
“Iris: Petal width vs. length”, data = iris)
```



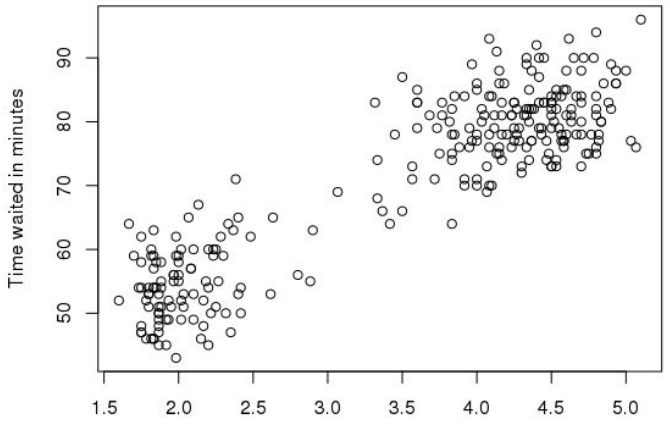
Legends

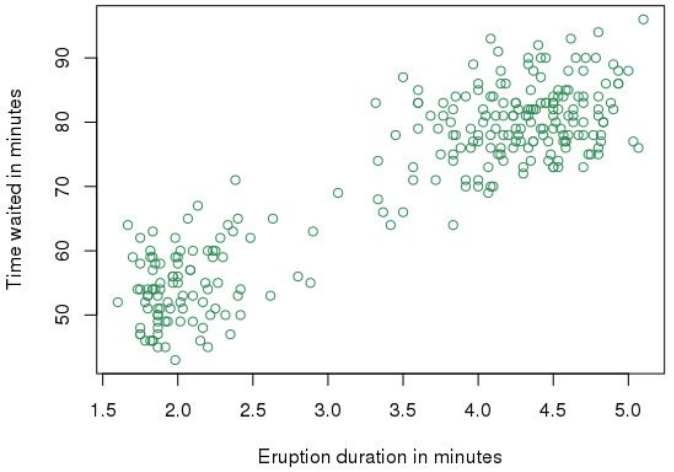
```
legend("topleft", legend = unique(iris$Species), fill =
unique(iris$Species))
```



Basic plotting in R

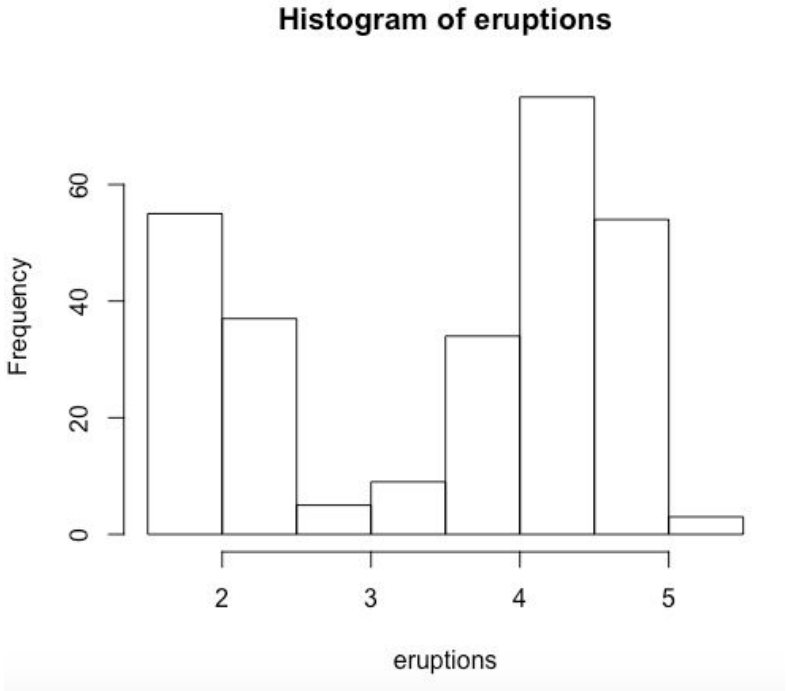
```
faithful$duration =
faithful$eruptions

plot(waiting ~ duration, xlab =
"Eruption duration in minutes",
ylab = "Time waited in minutes",
data = faithful)

plot(waiting ~ duration, xlab =
"Eruption duration in minutes",
ylab = "Time waited in minutes",
col = "seagreen", data = faithful)
```





Histograms

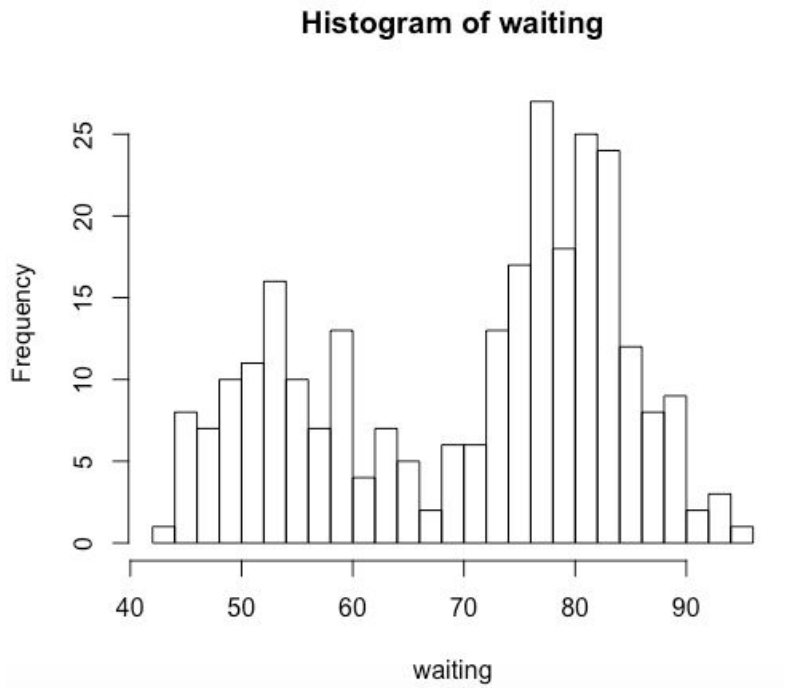
`hist(faithful$waiting)`



`hist(faithful$eruptions)`

Histograms


```
hist(faithful$waiting, breaks = 5)
```



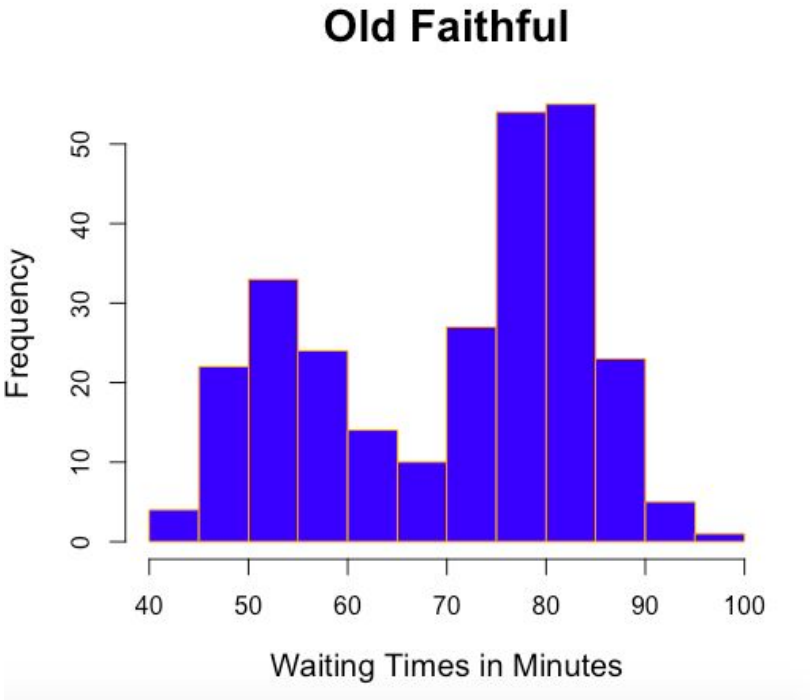

```
hist(faithful$waiting, breaks = 20)
```

Histograms

```
hist(faithful$waiting,
     main = "Old Faithful",
     cex.main = 1.75,
     xlab = "Waiting Time in Minutes",
     ylab = "Frequency",
     cex.lab = 1.25)
```

Histograms

```
hist(faithful$waiting,
     main = "Old Faithful",
     cex.main = 1.75,
     xlab = "Waiting Time in Minutes",
     ylab = "Frequency",
     cex.lab = 1.25,
     col = "blue",
     border = "orange")
```



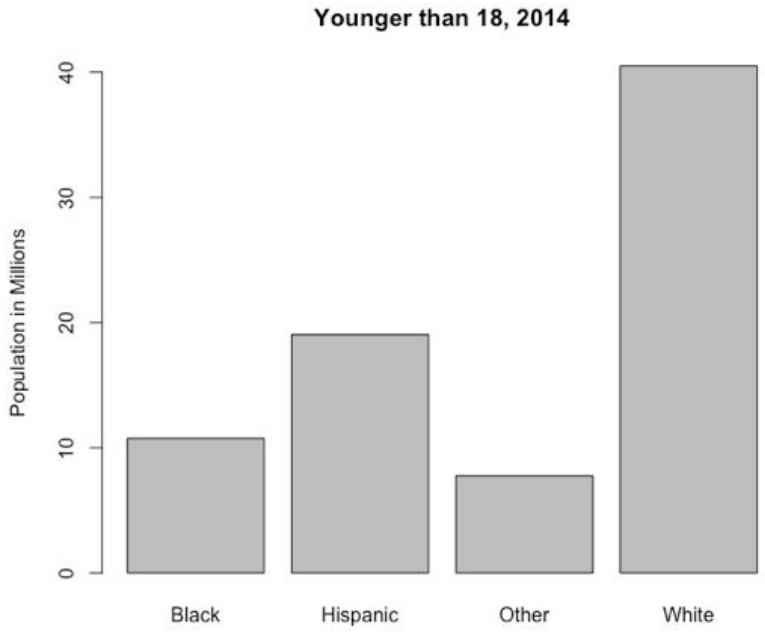
Stem and leaf plot

```
> stem(faithful$waiting)

  The decimal point is 1 digit(s) to the right of the |

  4 | 3
  4 | 55566666777788899999
  5 | 000001111112222233333333444444444
  5 | 555555666677788889999999
  6 | 00000022223334444
  6 | 555667899
  7 | 00001111123333333444444
  7 | 55555555666666666777777777777888888888888888999999999
  8 | 000000001111111111112222222222233333333333333344444444
  8 | 55555566666677888888999
  9 | 00000012334
  9 | 6
```

```
> stem(faithful$eruptions)

  The decimal point is 1 digit(s) to the left of the |

  16 | 070355555588
  18 | 000022233333335577777777888822335777888
  20 | 00002223378800035778
  22 | 0002335578023578
  24 | 00228
  26 | 23
  28 | 080
  30 | 7
  32 | 2337
  34 | 250077
  36 | 0000823577
  38 | 2333335582225577
  40 | 000000335778888800223355557777 8
  42 | 03335555778800233333555777 8
  44 | 022223355577800000002333335777 8888
  46 | 00002333577000000 23578
  48 | 00000022335800333
  50 | 0370
```

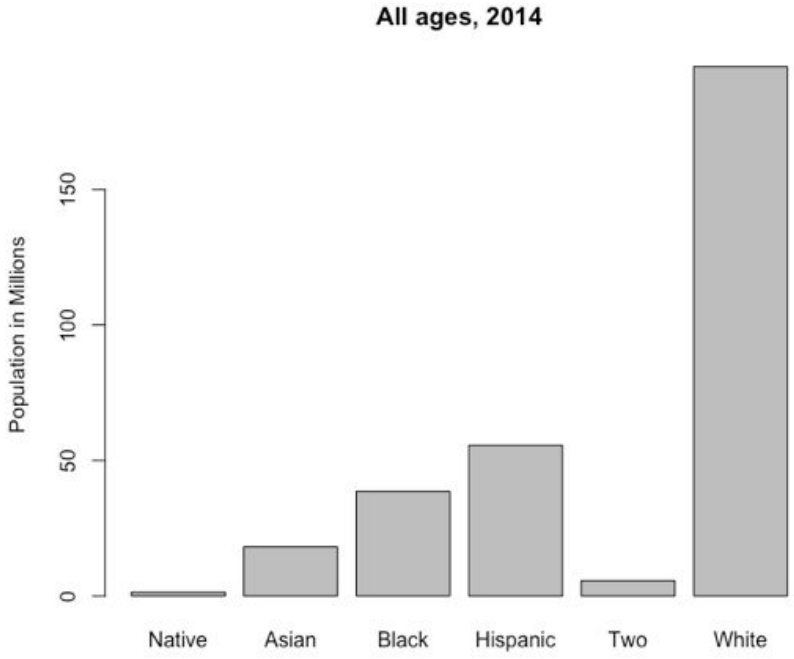
Bar charts

```
> races_younger
[1] "Black" "Hispanic" "Other" "White"

> population_in_millions_younger
[1] 10.76 19.03 7.76 40.50

>
barplot(population_in_millions_younger,
    names.arg = races_younger,
    main = "Younger than 18, 2014",
    ylab = "Population in Millions")
```

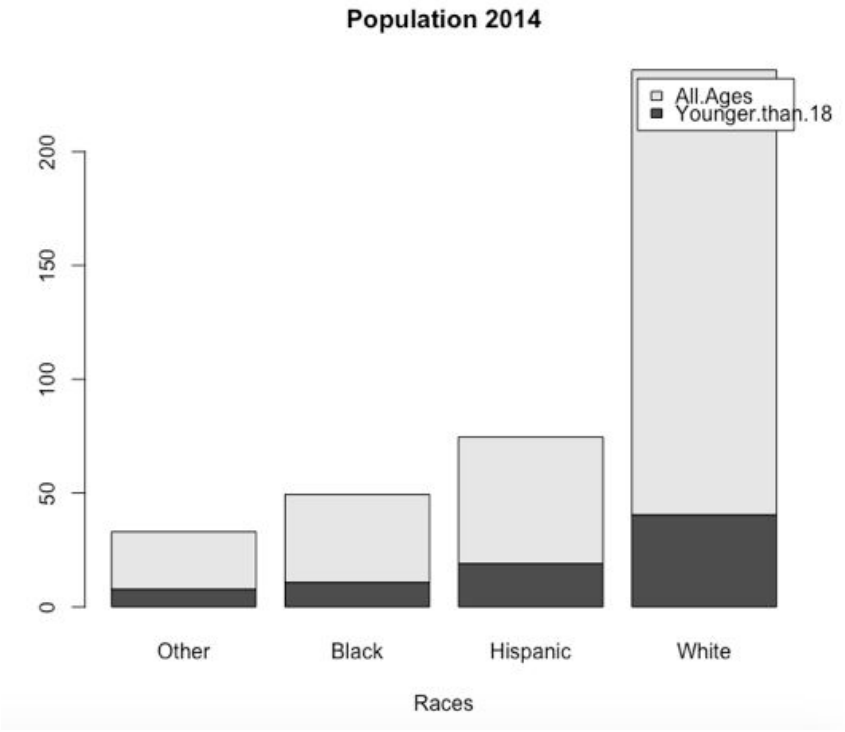


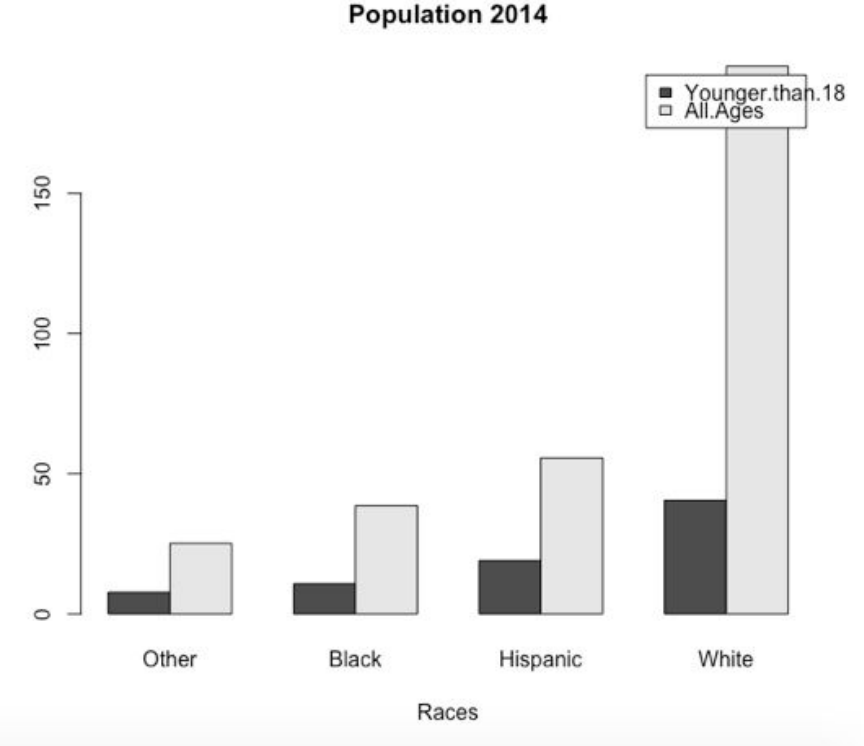
Bar charts

```
> races_all
[1] "Native" "Asian" "Black"
"Hispanic" "Two" "White"

> population_in_millions_all
[1] 1.39 18.12 38.60 55.61 5.67 195.35

barplot(population_in_millions_all,
    names.arg = races_all,
    main = "All ages, 2014",
    ylab = "Population in Millions")
```



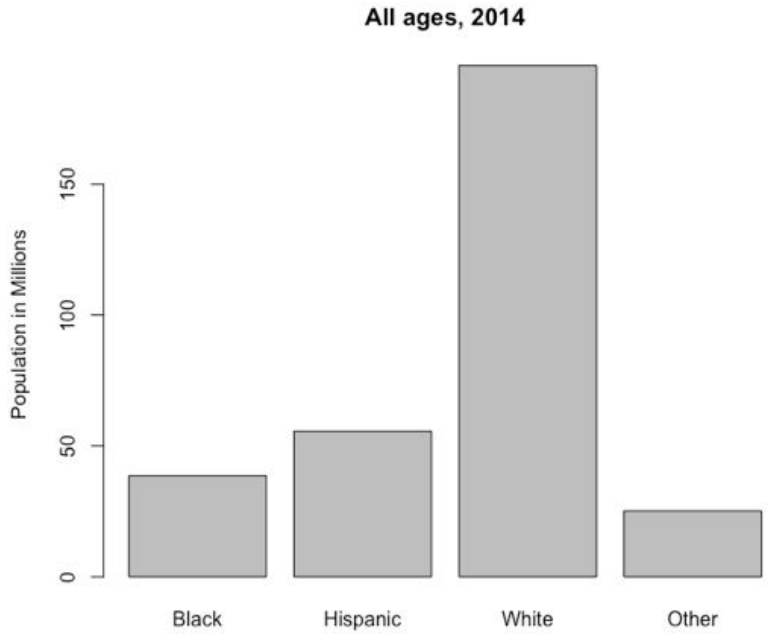
Stacked and grouped bar charts





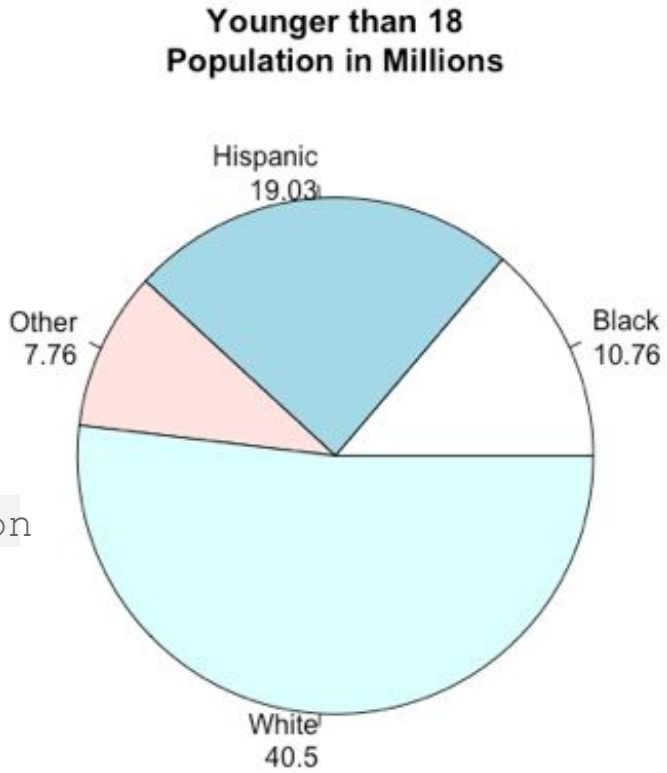
Bar charts

```
> races_all_other
[1] "Black" "Hispanic" "White" "Other"

> population_in_millions_all_other
[1] 38.60 55.61 195.35 25.19

barplot(population_in_millions_all_other,
    names.arg = races_all_other,
    main = "All ages, 2014",
    ylab = "Population in Millions")
```



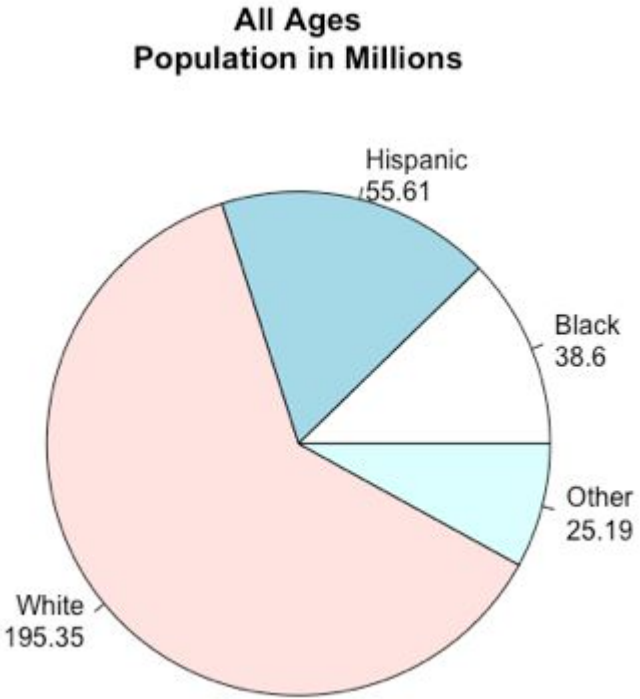
Pie charts

```
pie_labels_younger <-
    paste(races_younger,
    population_in_millions_younger,
    sep = "\n")

pie(population_in_millions_younger,
    main = "Younger than 18\nPopulation
in Millions",
    labels = pie_labels_younger)
```

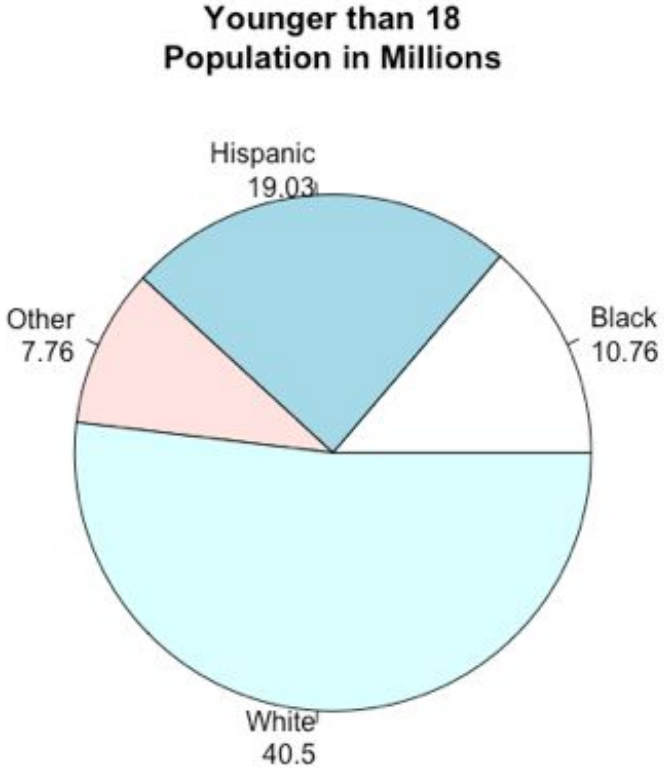


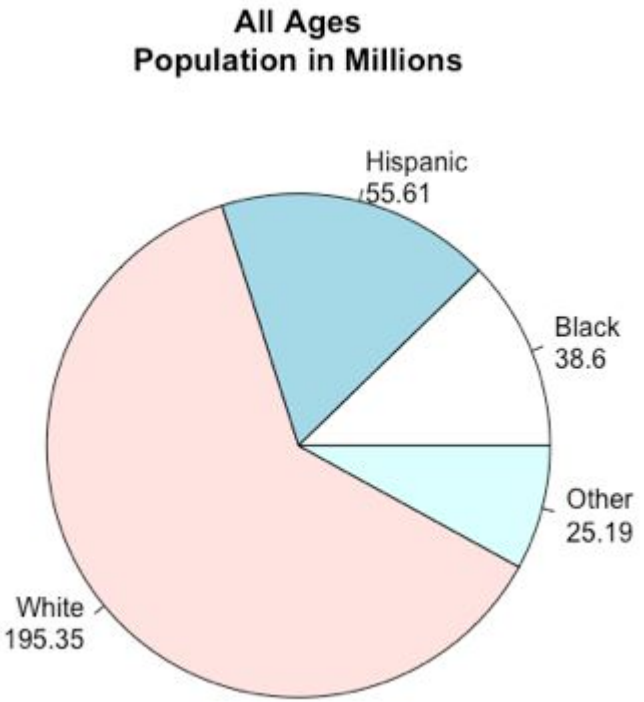
Pie charts

```
pie_labels_all_other <-
    paste(races_all_other,
    population_in_millions_all_other,
    sep = "\n")

pie(population_in_millions_all_other,
    main = "All Ages\nPopulation in
Millions",
    labels = pie_labels_all_other)
```



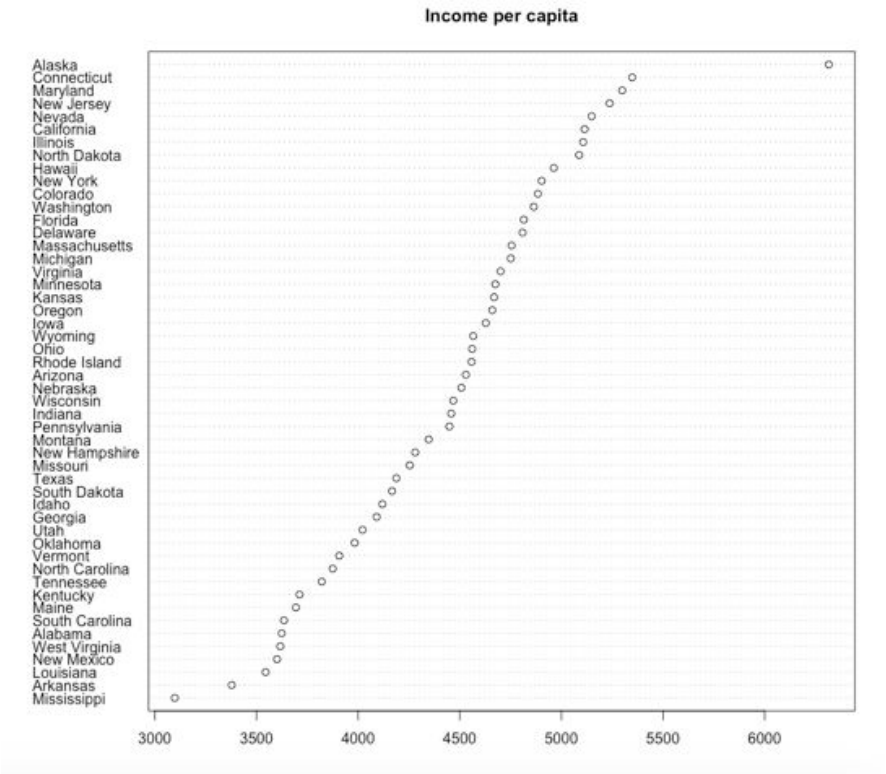
Pie charts


Younger than 18
Population in Millions
Hispanic
19.03
Black
10.76
Other
7.76
White
40.5


All Ages
Population in Millions
Hispanic
55.61
Black
38.6
Other
25.19
White
195.35

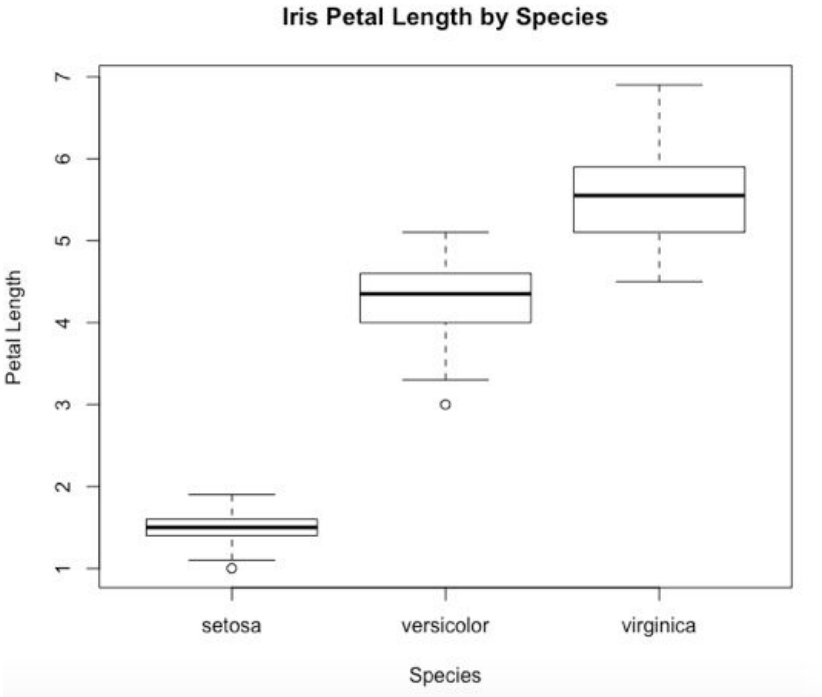
Dot plots

```
states <- data.frame(state.x77)

sorted_states <-
states[order(states$Income), ]

dotchart(sorted_states$Income,
    rownames(sorted_states),
    cex.lab = .25,
    main = "Income per capita")
```



Box plots

```
boxplot(Petal.Length ~ Species,
    data = iris,
    main = "Iris Petal Length by Species",
    xlab = "Species",
    ylab = "Petal Length")
```

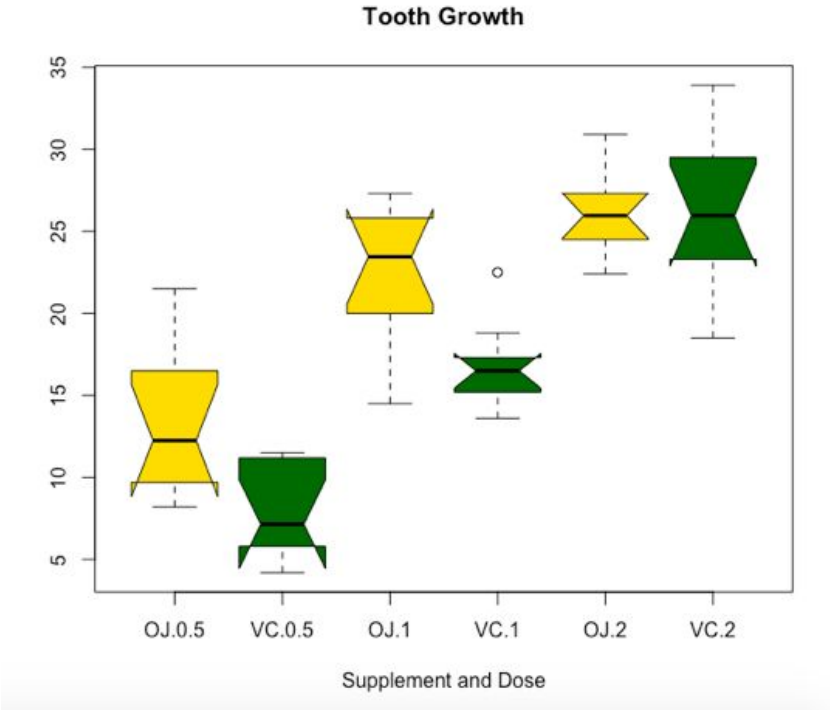


Notched Box plots

```
boxplot(len ~ supp * dose,
    data = ToothGrowth,
    notch = TRUE,
    col = (c("gold", "darkgreen")),
    main = "Tooth Growth",
    xlab = "Supplement and Dose")
```

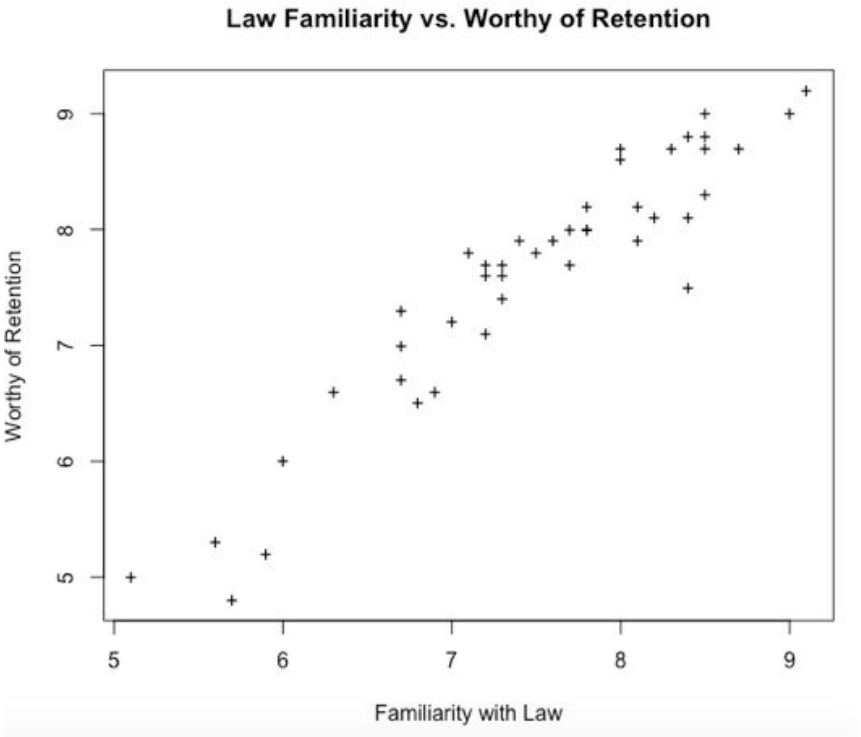
If the notches do not overlap, then the medians of the groups are different

(because their confidence intervals do not overlap).



Simple Scatterplots

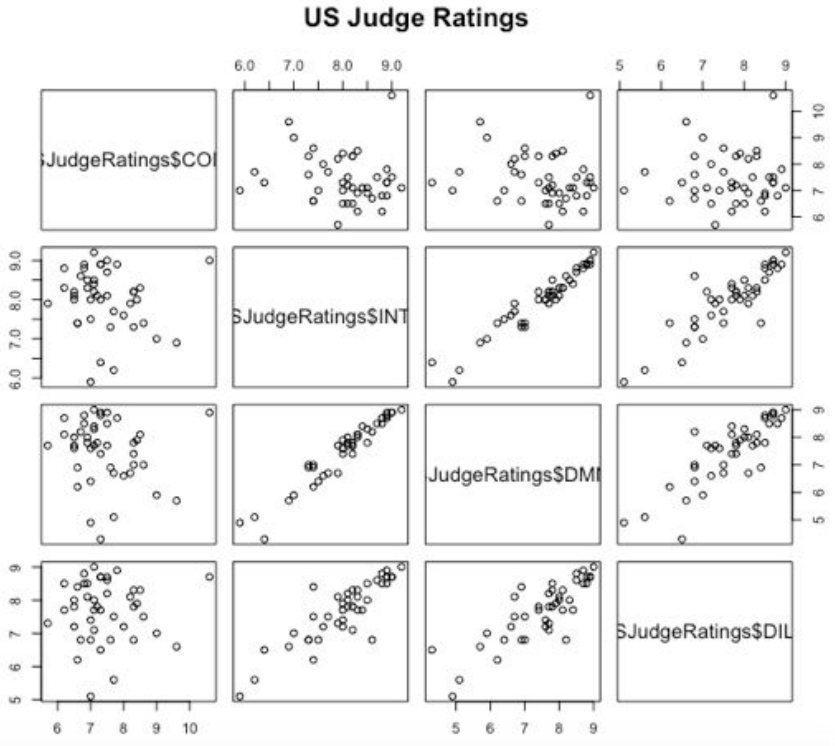
```
plot(USJudgeRatings$RTEN ~
    USJudgeRatings$FAMI,
    xlab = "Familiarity with Law",
    ylab = "Worthy of Retention",
    main = "Law Familiarity vs.
    Worthy of Retention",
    pch = "+")
```



Scatterplot Matrices

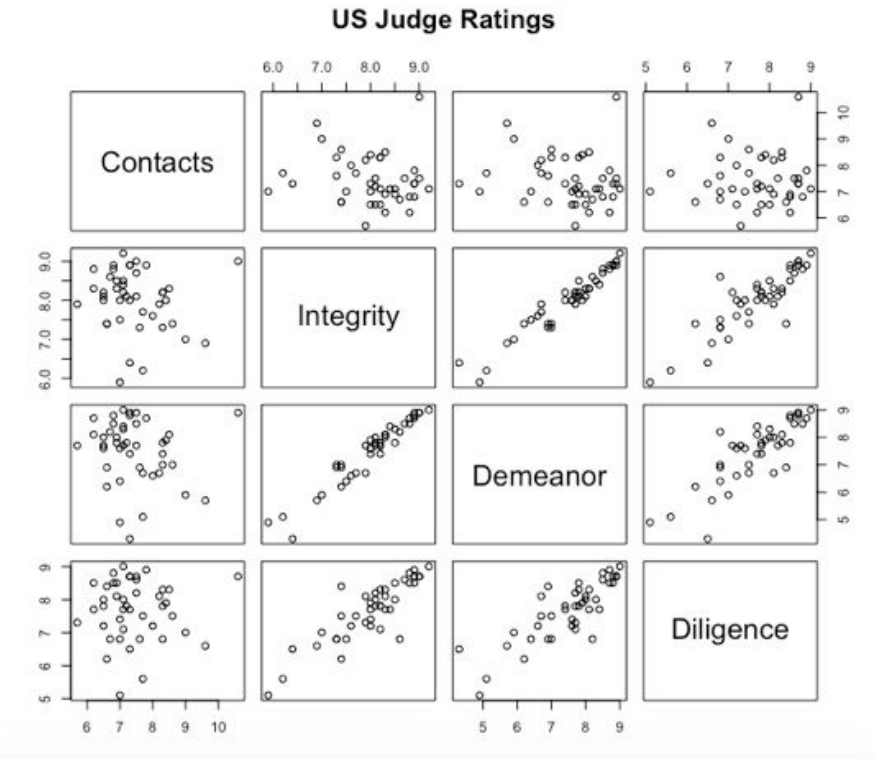
CONT Number of contacts of lawyer with judge.
INTG Judicial integrity.
DMNR Demeanor.
DILG Diligence.

```
pairs(~USJudgeRatings$CONT +
    USJudgeRatings$INTG +
    USJudgeRatings$DMNR +
    USJudgeRatings$DILG,
    main = "US Judge Ratings")
```



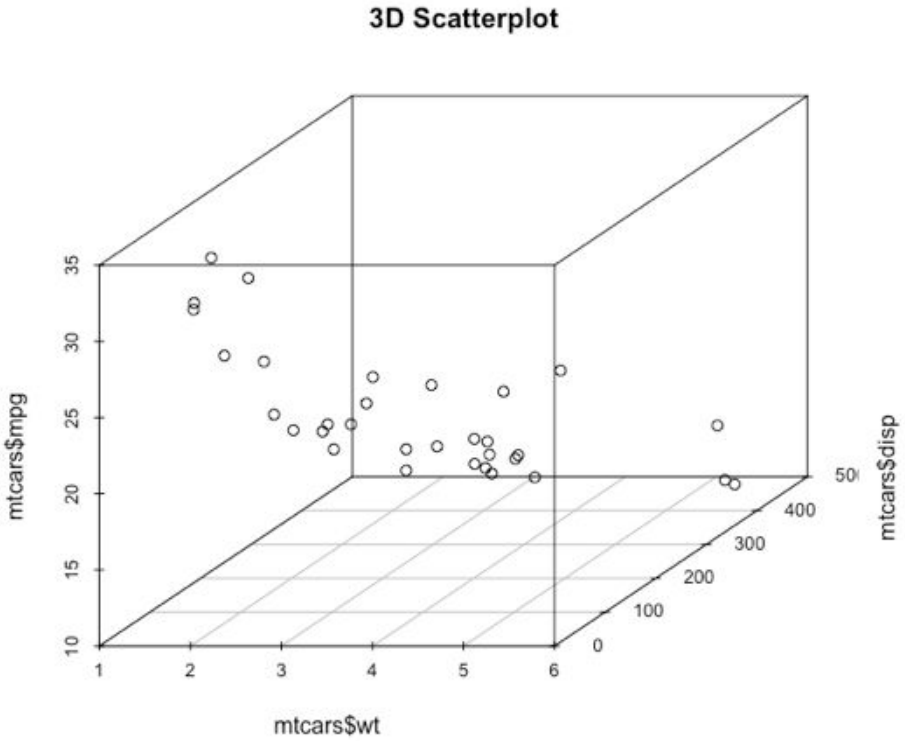
Scatterplot Matrices

```
features <- c("Contacts", "Integrity",
"Demeanor", "Diligence")

pairs(~USJudgeRatings$CONT +
    USJudgeRatings$INTG +
    USJudgeRatings$DMNR +
    USJudgeRatings$DILG,
    labels = features,
    main = "US Judge Ratings")
```



3D Scatterplots

```
library(scatterplot3d)

scatterplot3d(mtcars$wt,
    mtcars$disp, mtcars$mpg,
    main = "3D Scatterplot")
```



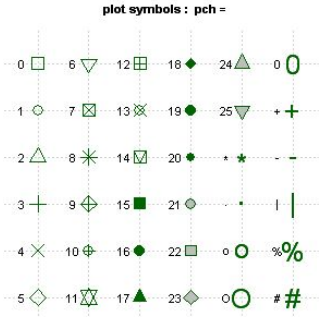
Graphical parameters

- Text and symbol size (`cex`: `.axis`, `.lab`, `.main`)
- Fonts (`font`: `.axis`, `.lab`, `.main`):
  - 1=plain, 2=bold, 3=italic, 4=bold italic, 5=symbol
- Colors (`col`): colors, and more colors

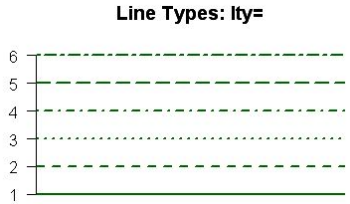
http://www.statmethods.net/advgraphs/parameters.html

Graphical parameters (cont’d)

Plotting symbols (`pch`)



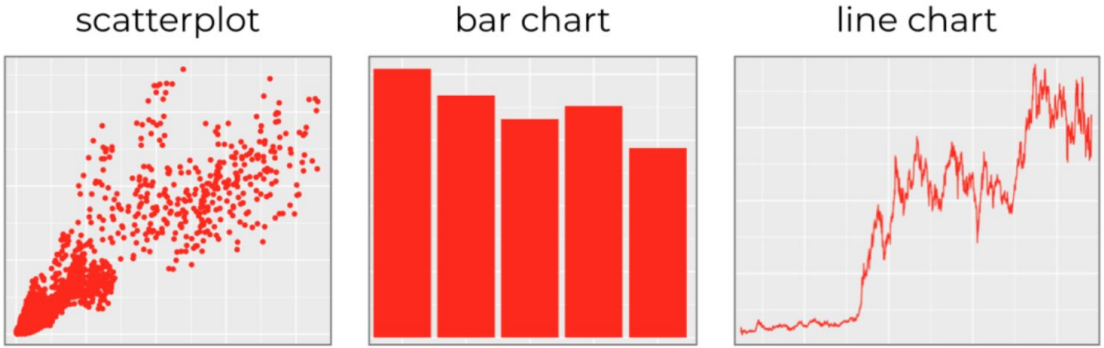
Line type (`lty`)



ggplot: A visualization package

What is ggplot?

- A package in R for creating data visualizations
- Very powerful and very flexible
- Similar capabilities to base R
- But ggplot's visualizations are usually preferred



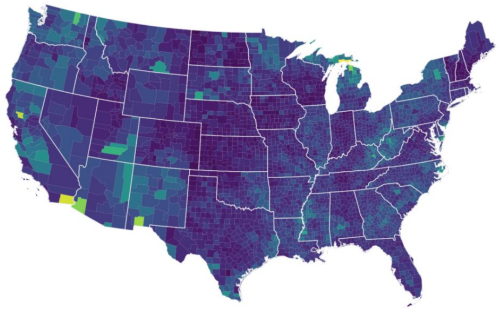


Image Source

ggplot Examples

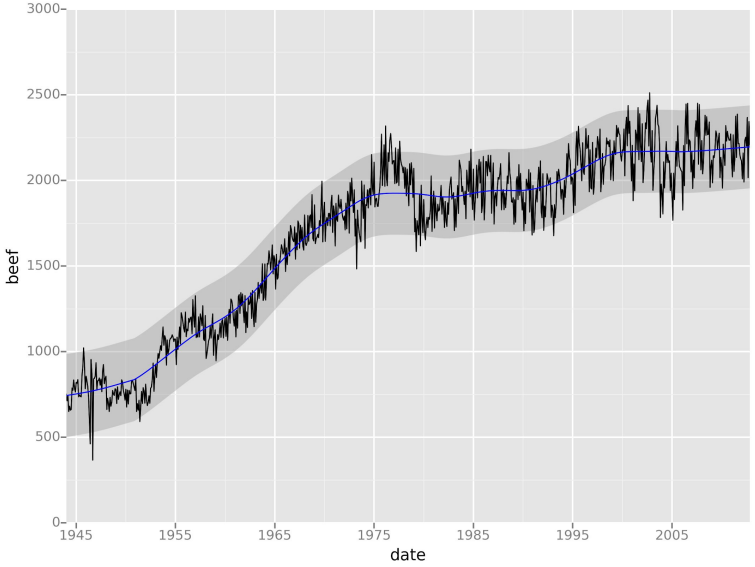


Image Source

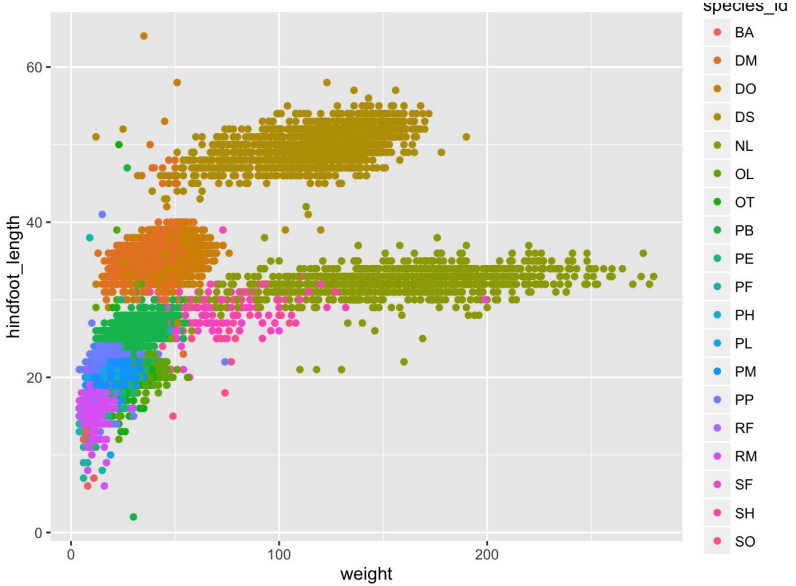
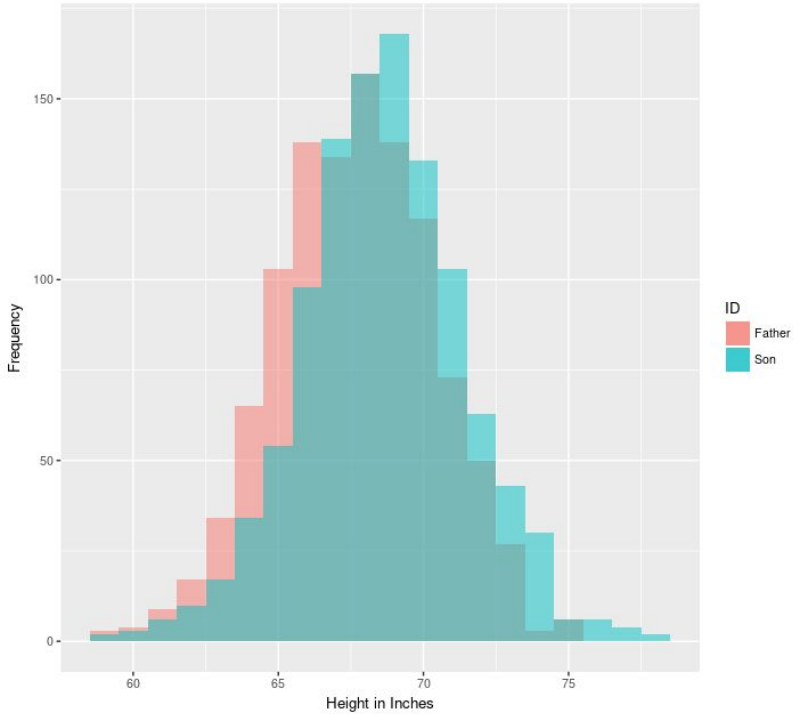
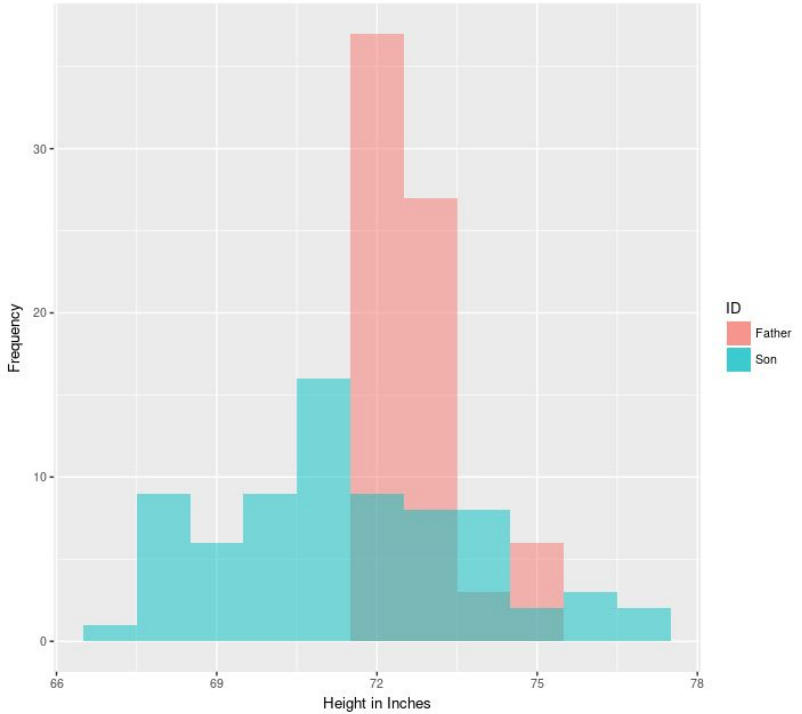


Image Source

ggplot Examples





ggplot = “grammar of graphics”

- Set of (grammatical) rules for creating graphics
- Each graphic made up of several independent components
- Each component can be manipulated individually
- And components can be combined in various ways

ggplot’s “Parts of Speech”

- data: noun/subject
- aes: adjectives
- geom: verb
- stat: adverb
- position: preposition

We will focus on learning how to use data, aes, and geom.

ggplot’s “Sentences”, etc.

- Individual components combine to make a layer (sentence)
- Can layer layers on top of one another (paragraph)
- String it all together with + (punctuation)

Key concepts

- Layers: We build ggplot objects by piling one or more layers on top of one another. Each layer contains data, aesthetics, geometry, etc.
- Aesthetics: `aes` refer to the information in the plot.
  - For example, the variables associated with the `x` and `y` axes.
- Geometric objects: `geom` describes how the information is displayed.
  - For example, a scatter plot or a bar plot.

We build up a plot by combining different layers using the + operator!

The process

1. define data and aesthetic

```
ggplot_object <- ggplot(data = dataframe,
                        aes(x = var1, y = var2))
```

2. add a layer with, for example, the bar geom, or the point geom

```
ggplot_object <- ggplot_object + geom_bar()
ggplot_object <- ggplot_object + geom_point()
```

3. show the plot

```
ggplot_object
```

Example: Using ggplot to plot iris data

STEP 1

We use the `data` parameter to specify the data frame.

We use the `aes` parameter to specify the variables to display.

```
my_plot <- ggplot(data = iris,
                  aes(x = Species, y = Sepal.Length))
```

This statement creates a ggplot object using the prescribed data and aesthetic.

Bar Plot

STEP 2

We use the `geom_col` function to specify that we want to plot bars.

```
bar_plot <- my_plot + geom_col()
```

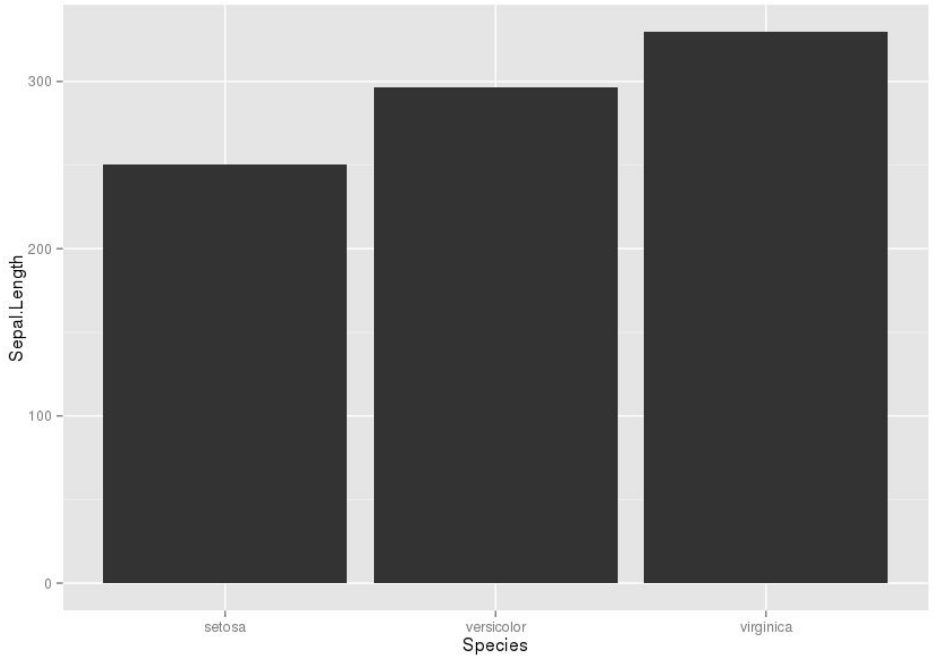
This statement adds the geometry layer to the existing ggplot object.

Bar Plot

STEP 3

Display our bar plot:

```
bar_plot
```



Scatter Plot

STEP 1

We use the `data` parameter to specify the data frame.

We use the `aes` parameter to specify the variables to display.

```
my_plot <- ggplot(data = iris,
                  aes(x = Petal.Width, y = Petal.Length))
```

This statement creates a ggplot object using the prescribed data and aesthetic.

Scatter Plot

STEP 2

We use the `geom_point` function to specify that we want to plot points.

```
scatter_plot <- my_plot + geom_point()
```

This statement adds the geometry layer to the existing ggplot object.

Scatter Plot

STEP 2b

We can also specify color, size, and degree of transparency:

```
scatter_plot <- my_plot + geom_point(color = "blue",
                                     size = 3, alpha = 0.5)
```

We can also customize the labels:

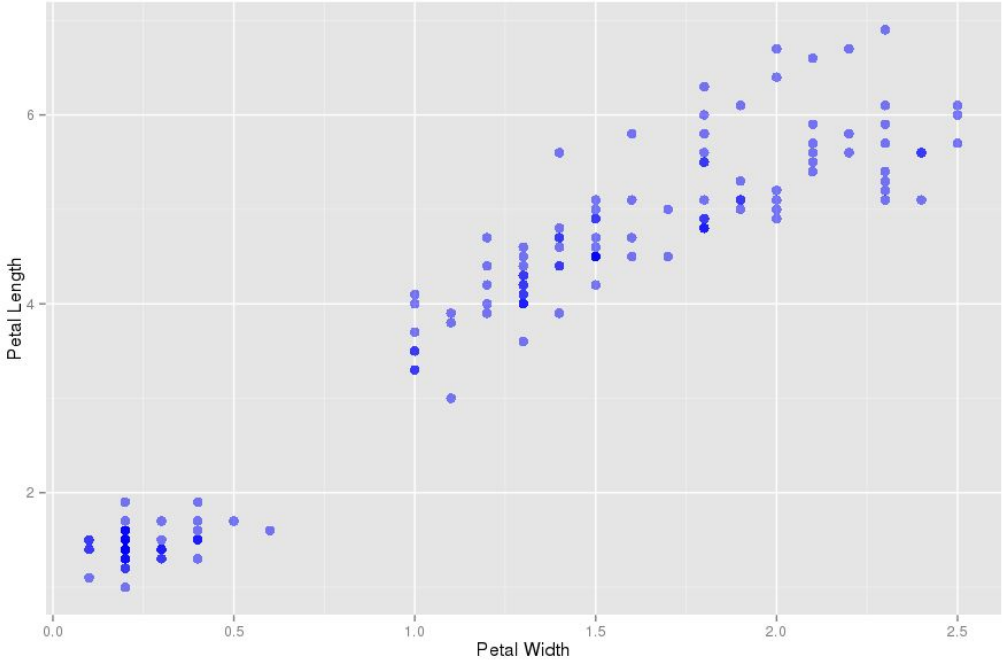
```
scatter_plot <- scatter_plot + labs(x = "Petal Width",
                                    y = "Petal Length")
```

Scatter Plot

STEP 3

Display our scatter plot:

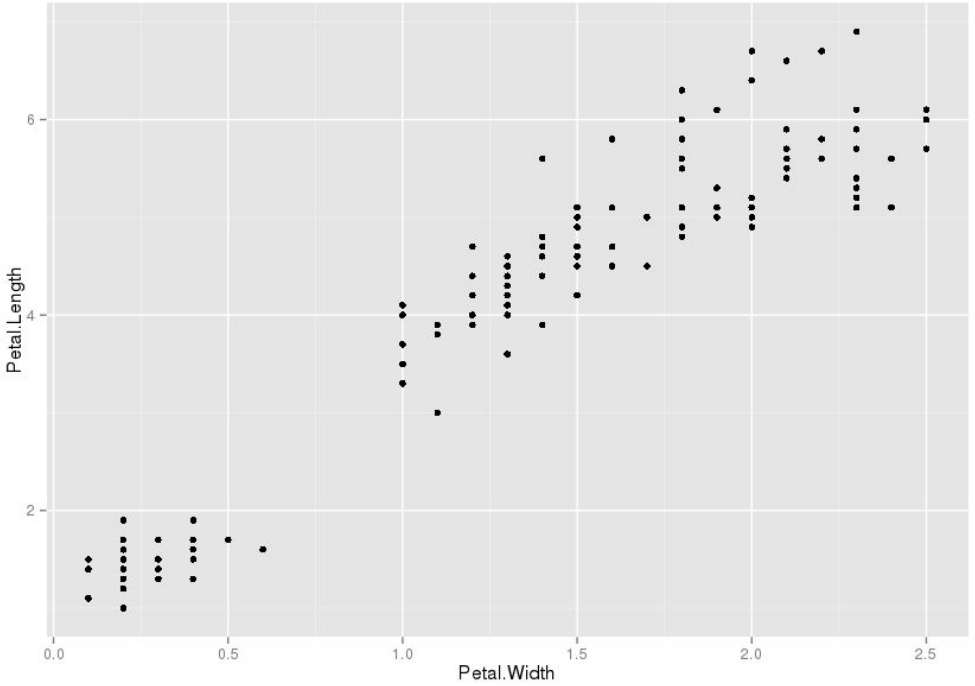
```
scatter_plot
```



Consolidating Code

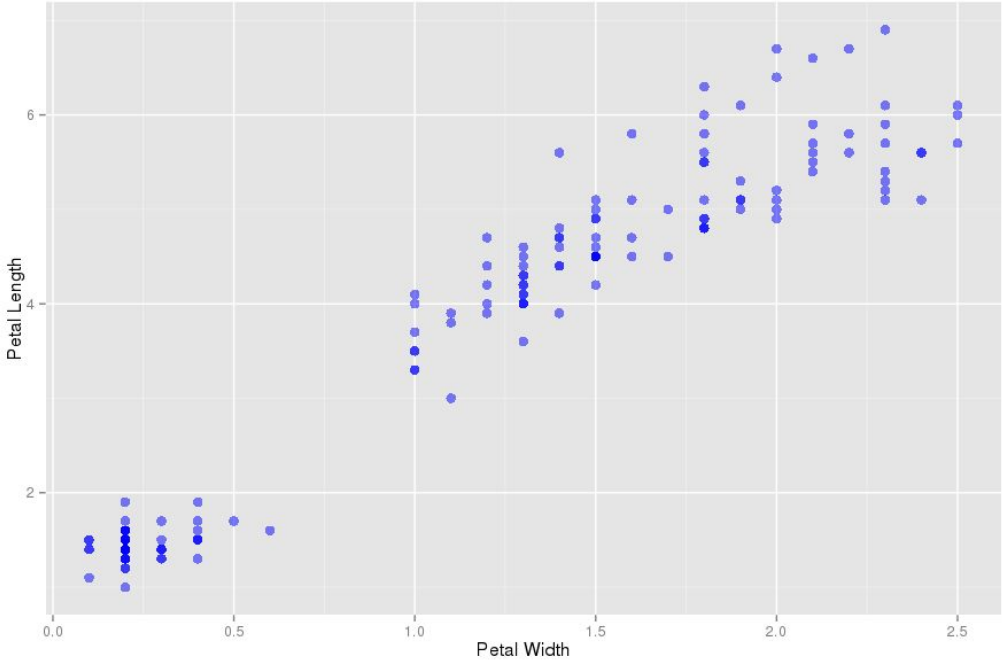
You don't have to build a ggplot object in separate steps. If you want, you can create the object and display it all in one go:

```
ggplot(iris,
aes(x = Petal.Width, y
= Petal.Length)) +
geom_point()
```



Consolidating Code

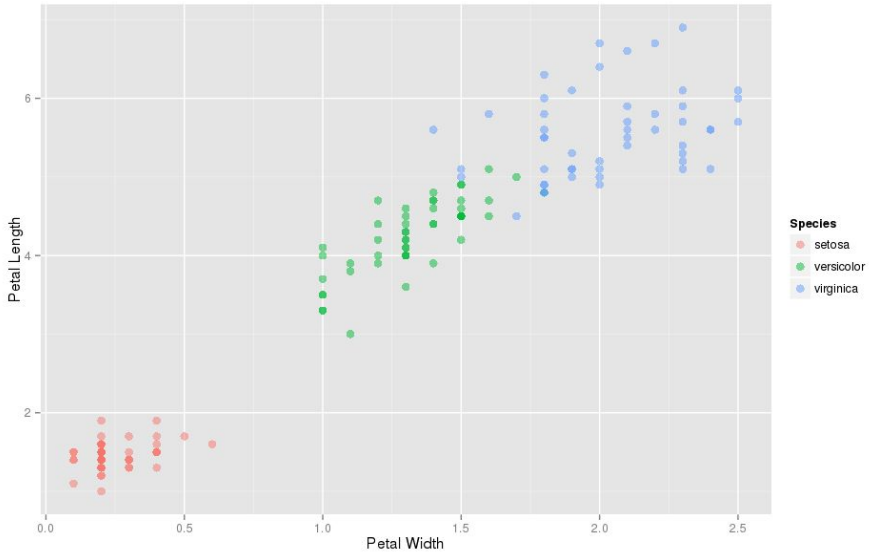
```
ggplot(iris,
aes(x = Petal.Width,
y = Petal.Length)) +
geom_point(color =
"blue", size = 3, alpha =
0.5) +
labs(x = "Petal Width",
y = "Petal Length")
```



Scatter Plot, cont’d

Finally, let’s color-code by species!

```
ggplot(iris, aes(x =
Petal.Width, y = Petal.Length,
color = Species)) +
geom_point(size = 3,
alpha = 0.5) + labs(x = "Petal
Width", y = "Petal Length")
```

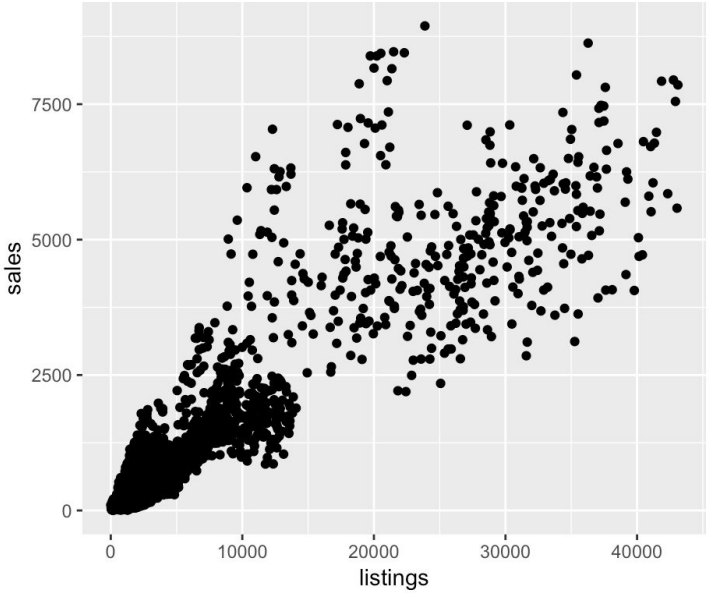


Extras

Scatterplot

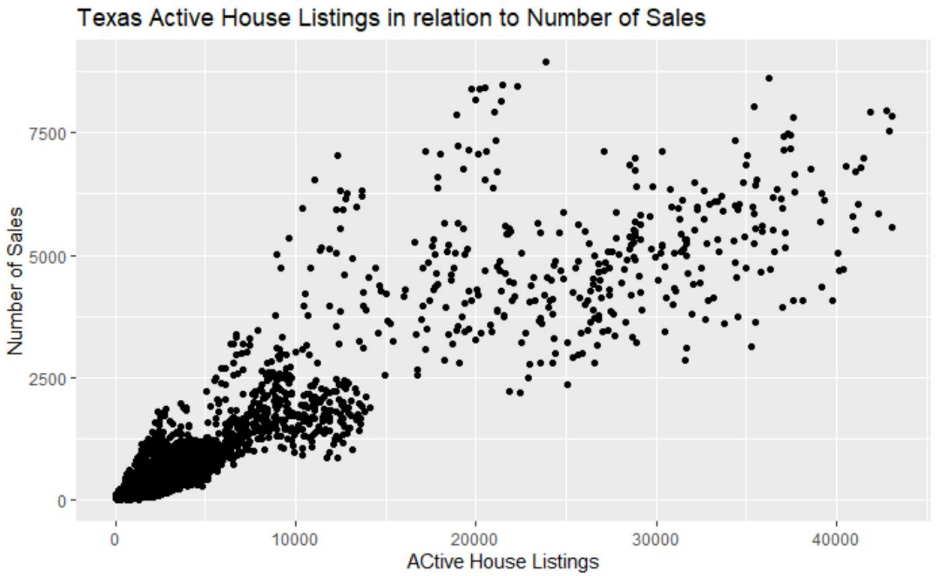
```
ggplot(data = txhousing,
  aes(x = listings, y = sales)) +
  geom_point()
```

- `ggplot`: function call
- `data`: a data frame
- `aes`: x is the explanatory variable, and y is the response variable
- `geom_point`: a scatterplot



Add labels

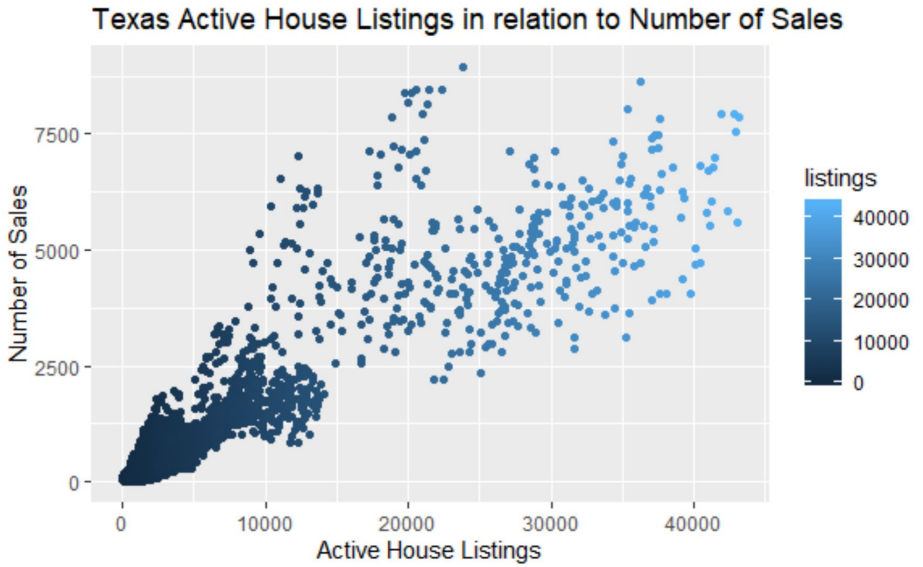
- + `labs()` property
- arguments:
  - `title =`
  - `subtitle =`
  - `x =`
  - `y =`
  - `caption =`




```
+  labs(title = "Texas Active House Listings in relation to Number of
   Sales", x = "Active House Listings", y = "Number of Sales")
```

Add color

- We can also change color by adjusting the aes() inside the geoms
- Within the geom_point() function, add the argument ***color*** and have it equal the independent variable

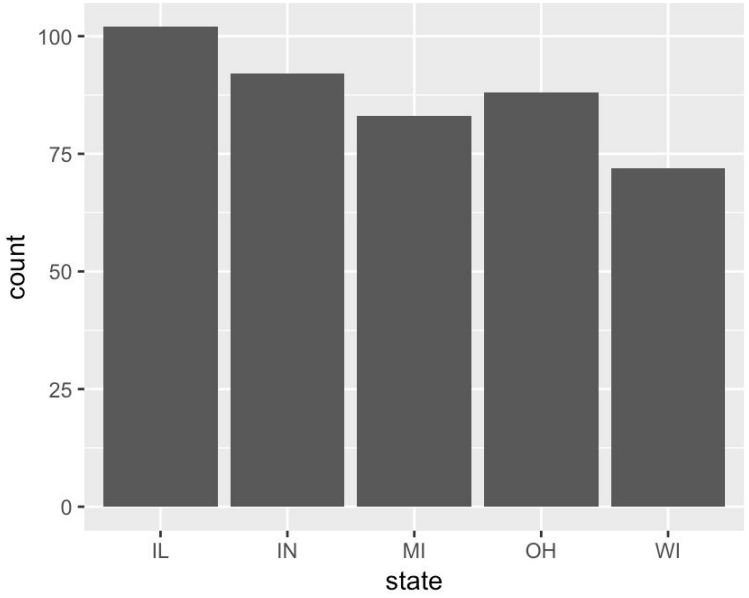



```
geom_point(aes(color = listings))
```

Bar graph

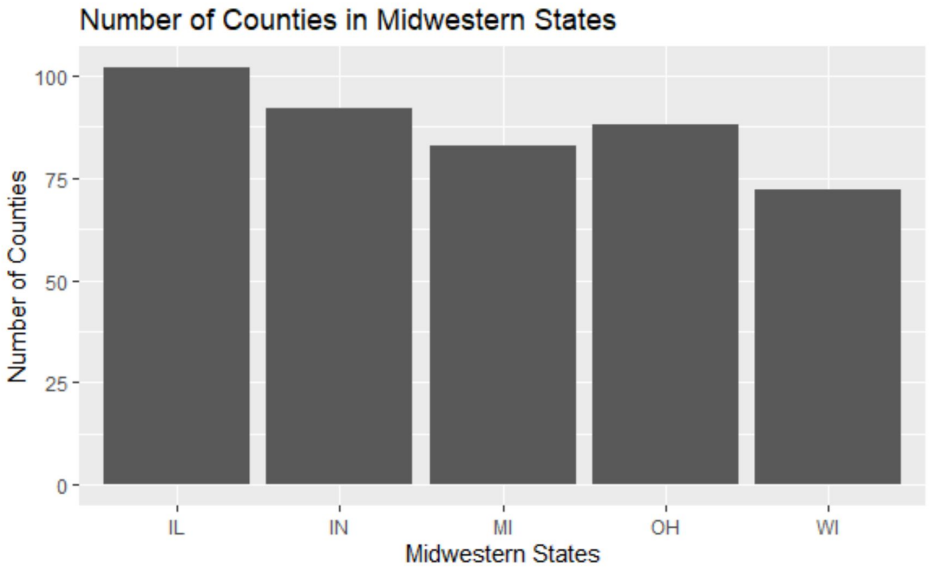
```
ggplot(data = midwest, aes(x = state)) +
  geom_bar()
```

- `ggplot`: function call
- `data`: a data frame
- `aes`: x is the variable
- `geom_bar`: a bar graph



Add labels

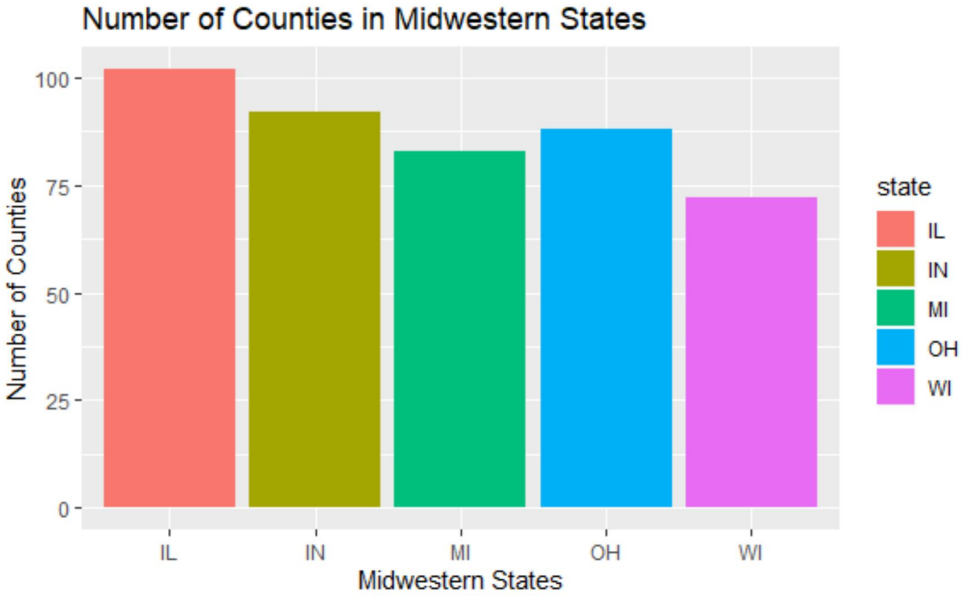
- + `labs()` property
- arguments:
  - `title =`
  - `subtitle =`
  - `x =`
  - `y =`
  - `caption =`




```
+  labs(title = "Number of Counties in Midwestern States", x =
   "Midwestern States", y = "Number of Counties")
```

Add color

- We can also change color by adjusting the aes() inside the geoms
- Within the geom_bar() function, add the argument ***fill*** and have it equal the independent variable




```
geom_bar(aes(fill = state))
```