

Recap From Last Time:

- What is R and why should we use it?
- Familiarity with R's basic syntax.
- Familiarity with major R data structures namely **vectors** and **data.frames**.
- Understand the basics of using **functions** (arguments, vectorization and re-cycling).
- Appreciate how you can use R scripts to aid with reproducibility.

[MPA Link]

Today's Learning Goals

- Appreciate the major elements of **exploratory data analysis** and why it is important to visualize data.
- Be conversant with **data visualization best practices** and understand how good visualizations optimize for the human visual system.
- Be able to generate informative graphical displays including **scatterplots**, **histograms**, **bar graphs**, **boxplots**, **dendograms** and **heatmaps** and thereby gain exposure to the extensive graphical capabilities of R.
- Appreciate that you can build even more complex charts with **ggplot** and additional R packages such as **rgl**.

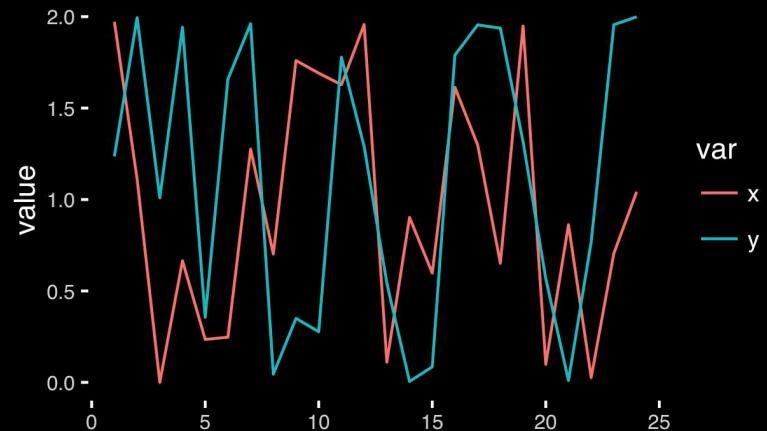
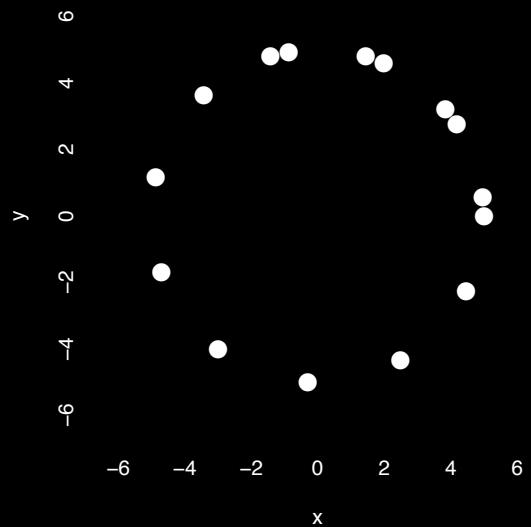
Today's Learning Goals

- Appreciate the major elements of **exploratory data analysis** and why it is important to visualize data.
- Be conversant with **data visualization best practices** and understand how good visualizations optimize for the human visual system.
- Be able to generate informative graphical displays including **scatterplots**, **histograms**, **bar graphs**, **boxplots**, **dendograms** and **heatmaps** and thereby gain exposure to the extensive graphical capabilities of R.
- Appreciate that you can build even more complex charts with **ggplot** and additional R packages such as **rgl**.

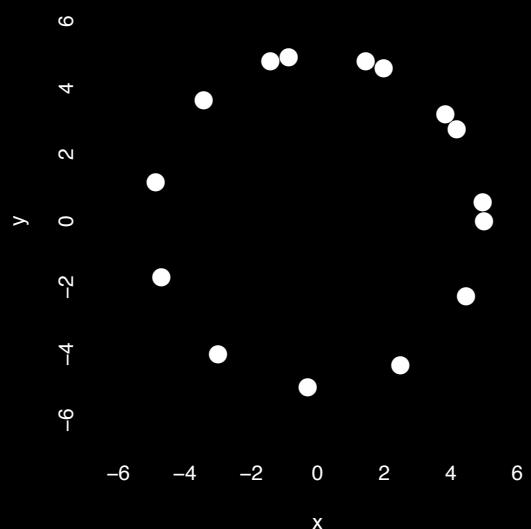
Why visualize at all?

	x	y
1	5.00	0.00
2	4.18	2.75
3	1.98	4.59
4	-0.86	4.92
5	-3.43	3.64
6	-4.86	1.16
7	-4.70	-1.70
8	-2.99	-4.01
9	-0.30	-4.99
10	2.49	-4.34
11	4.46	-2.25
12	4.97	0.57
13	3.84	3.20
14	1.45	4.79
15	-1.42	4.79

	x	y
Min.	-4.86	-4.99
1st Qu.	-2.21	-1.98
Median	1.45	1.16
Mean	0.65	0.87
3rd Qu.	4.01	4.12
Max.	5.00	4.92



https://bioboot.github.io/bggns18/class-material/05_draw_circle_points/

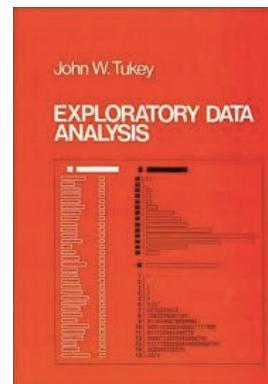


Exploratory Data Analysis

- ALWAYS look at your data!
- If you can't see it, then don't believe it!
- Exploratory Data Analysis (EDA) allows us to:
 1. Visualize distributions and relationships
 2. Detect errors
 3. Assess assumptions for confirmatory analysis
- EDA is the first step of data analysis!

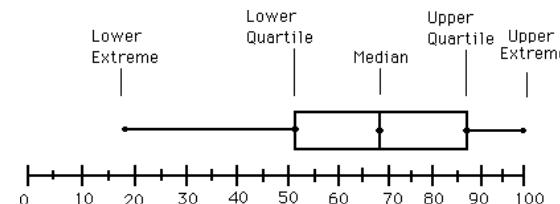
Exploratory Data Analysis 1977

- Based on insights developed at Bell Labs in the 60's
- Techniques for visualizing and summarizing data
- What can the data tell us? (in contrast to "confirmatory" data analysis)
- Introduced many basic techniques:
 - 5-number summary, box plots, stem and leaf diagrams,...
- 5 Number summary:
 - extremes (min and max)
 - median & quartiles
 - More robust to skewed & longtailed distributions



Side-note: boxplots

- **Box-and-whisker plot** : a graphical form of 5-number summary (Tukey)



```
boxplot( rnorm(1000,0) )
```

```
summary(); hist()
```

16

The Trouble with Summary Stats

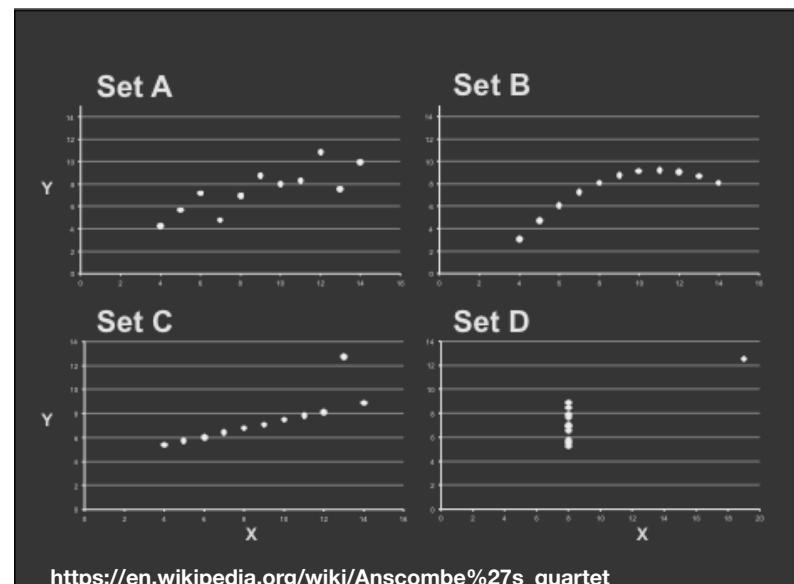
Set A		Set B		Set C		Set D	
X	Y	X	Y	X	Y	X	Y
10	8.04	10	9.14	10	7.46	8	6.58
8	6.95	8	8.14	8	6.77	8	5.76
13	7.58	13	8.74	13	12.74	8	7.71
9	8.81	9	8.77	9	7.11	8	8.84
11	8.33	11	9.26	11	7.81	8	8.47
14	9.96	14	8.1	14	8.84	8	7.04
6	7.24	6	6.13	6	6.08	8	5.25
4	4.26	4	3.1	4	5.39	19	12.5
12	10.84	12	9.11	12	8.15	8	5.56
7	4.82	7	7.26	7	6.42	8	7.91
5	5.68	5	4.74	5	5.73	8	6.89

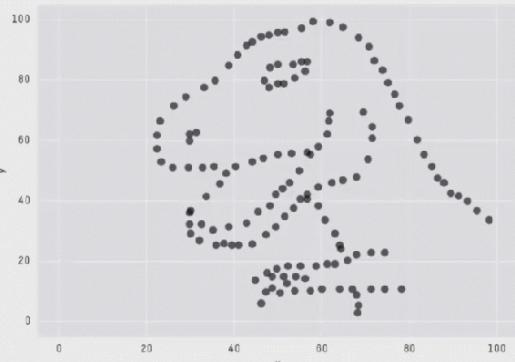
Summary Statistics Linear Regression

$u_x = 9.0 \quad \sigma_x = 3.317 \quad Y = 3 + 0.5 X$
 $u_y = 7.5 \quad \sigma_y = 2.03 \quad R^2 = 0.67$

[Anscombe 73]

Looking at Data





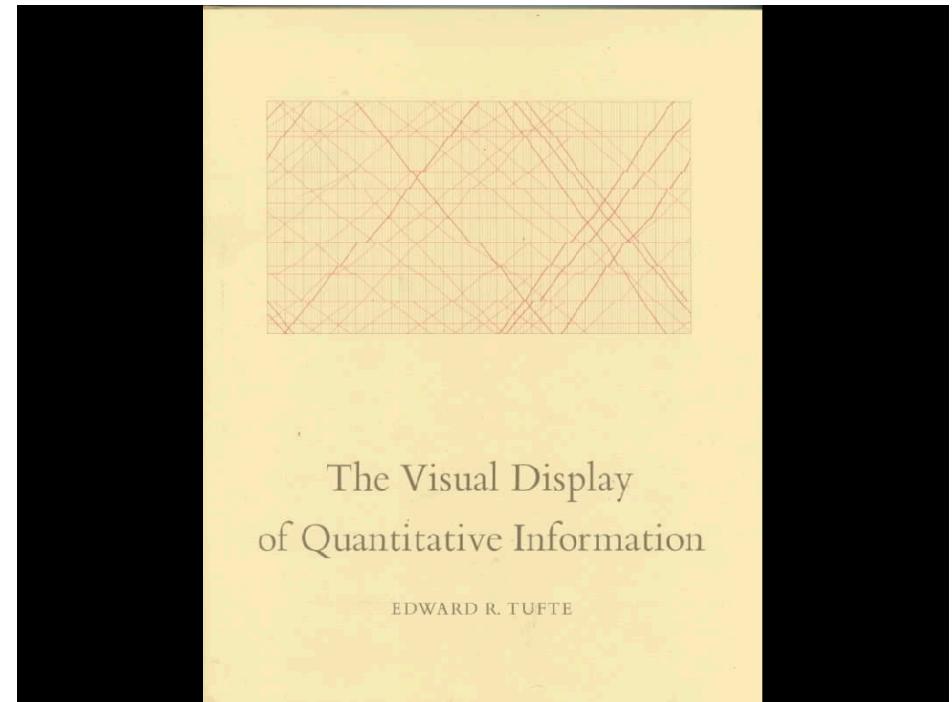
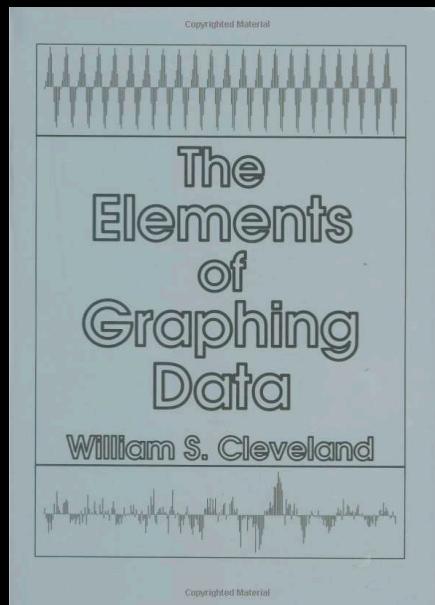
X Mean: 54.2659224
 Y Mean: 47.8313999
 X SD : 16.7649829
 Y SD : 26.9342120
 Corr. : -0.0642526

Key point: You need to visualize your data!

<https://github.com/stephlocke/datasauRus>

Today's Learning Goals

- Appreciate the major elements of **exploratory data analysis** and why it is important to visualize data.
- Be conversant with **data visualization best practices** and understand how good visualizations optimize for the human visual system.
- Be able to generate informative graphical displays including **scatterplots, histograms, bar graphs, boxplots, dendograms** and **heatmaps** and thereby gain exposure to the extensive graphical capabilities of R.
- Appreciate that you can build even more complex charts with **ggplot** and additional R packages such as **rgl**.



Key Point:
Good visualizations optimize
for the human visual system.

Key Point: The most important measurement should exploit the highest ranked encoding possible

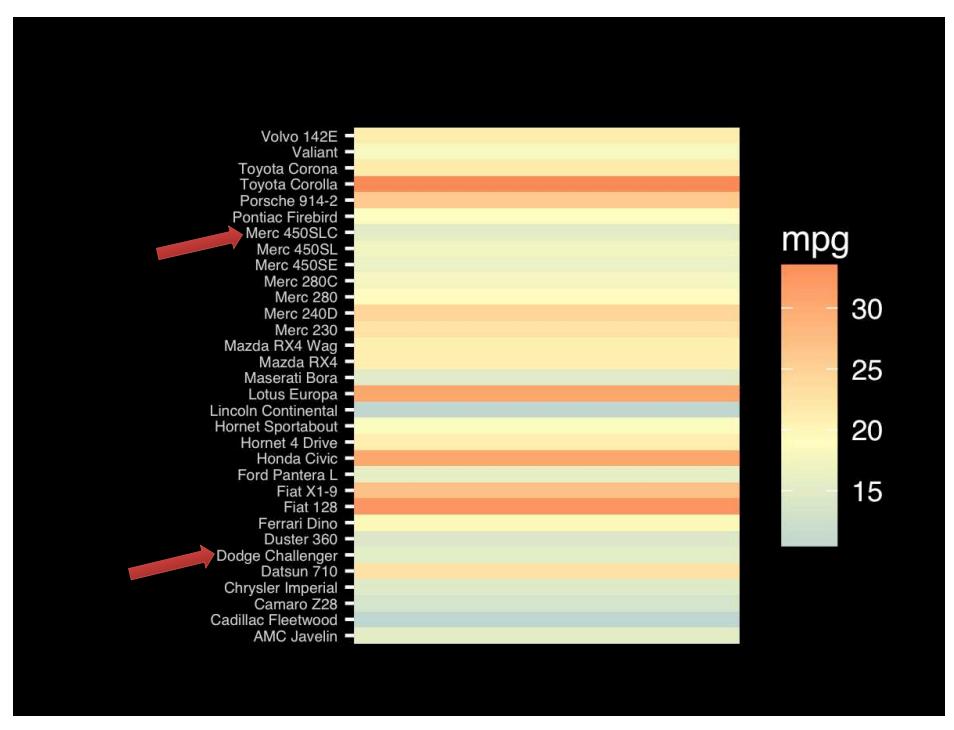
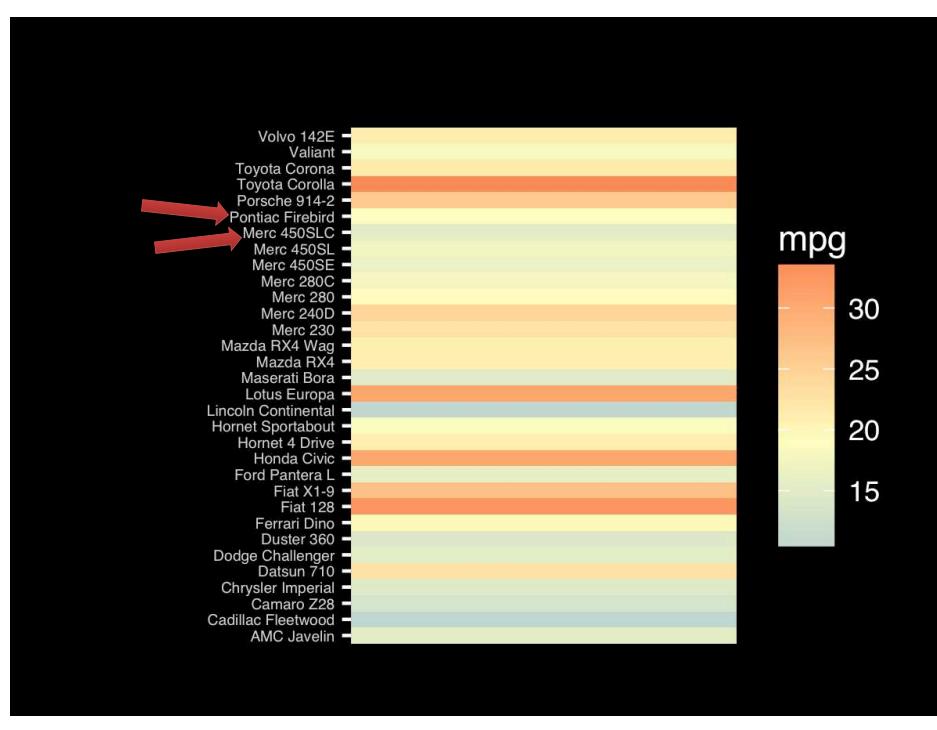
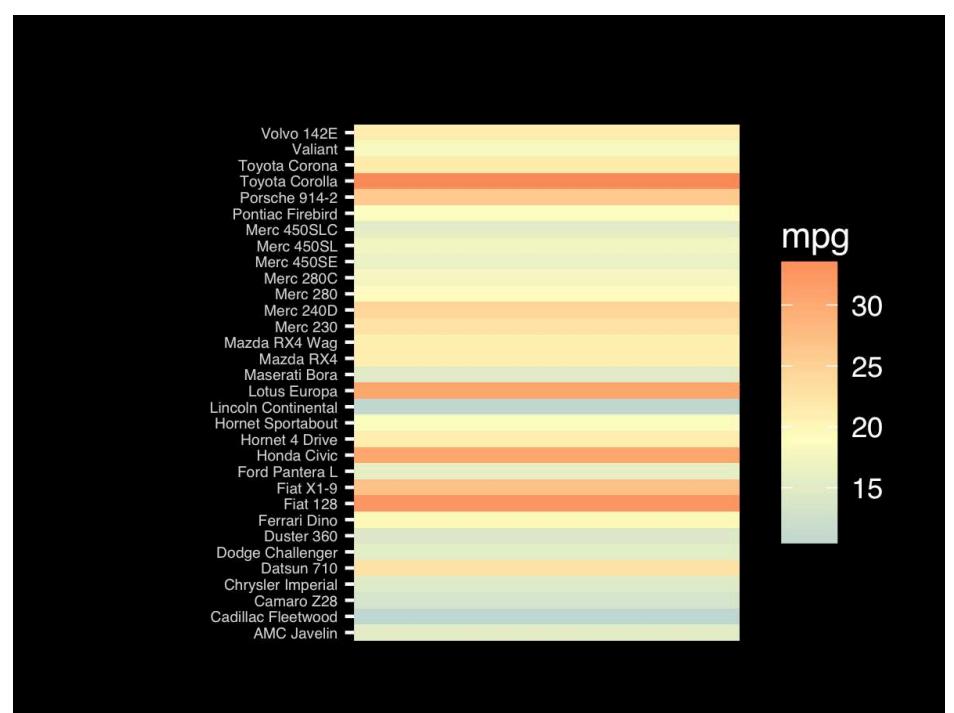
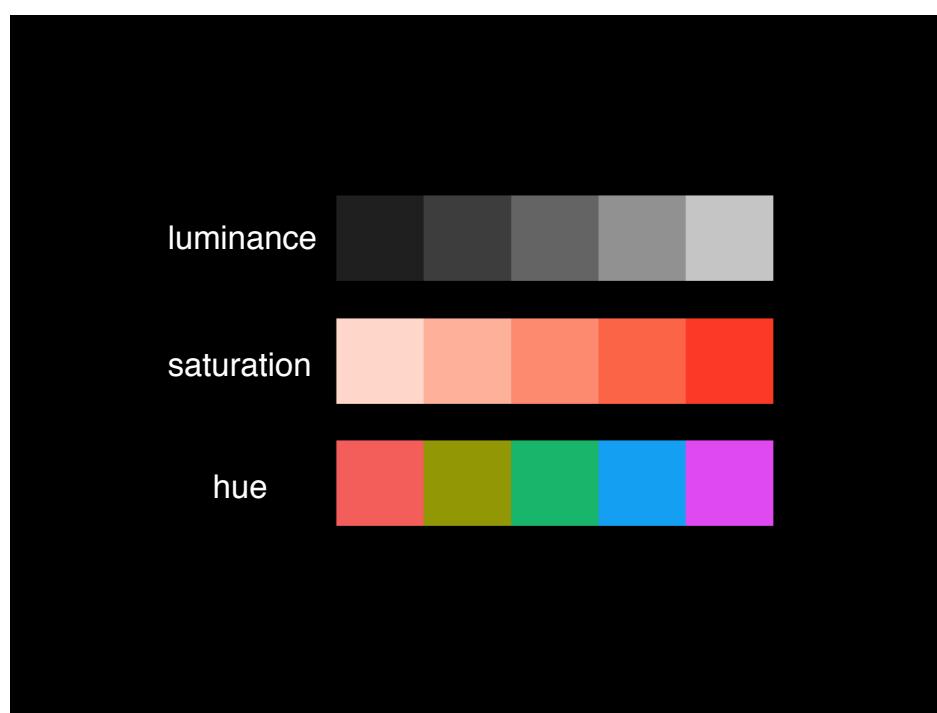
- Position along a common scale
- Position on identical but nonaligned scales
- Length
- Angle or Slope
- Area
- Volume or Density or Color saturation/hue

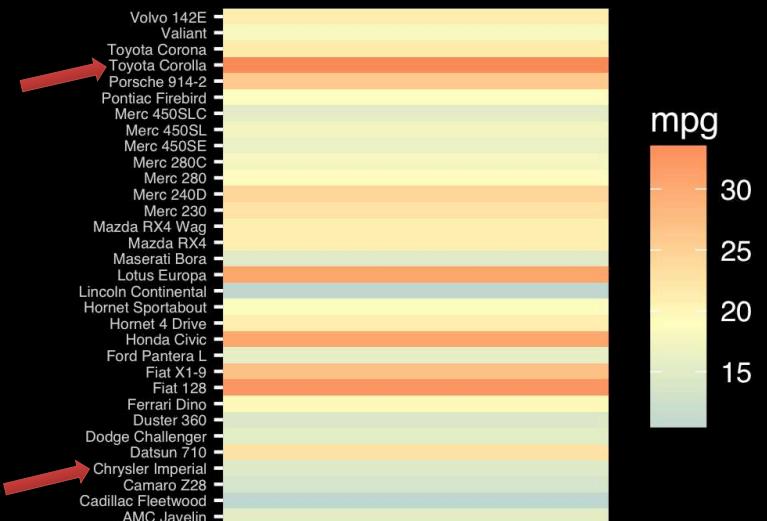
Key Point: The most important measurement should exploit the highest ranked encoding possible

- 
- Position along a common scale
 - Position on identical but nonaligned scales
 - Length
 - Angle or Slope
 - Area
 - Volume or Density or Color saturation/hue

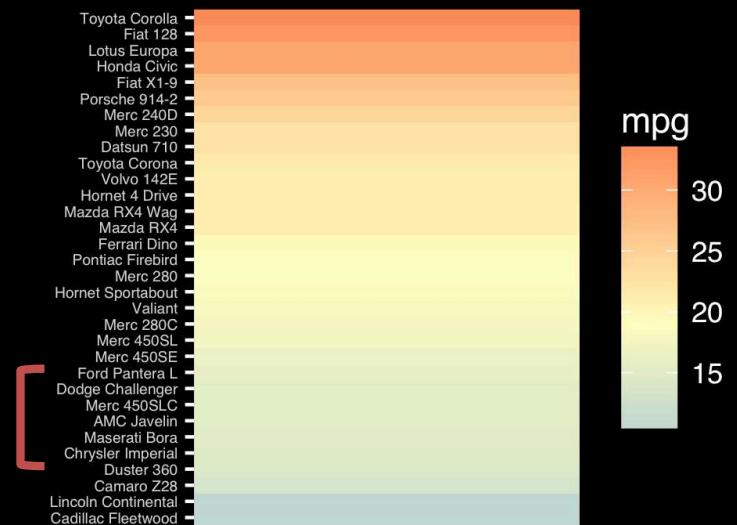
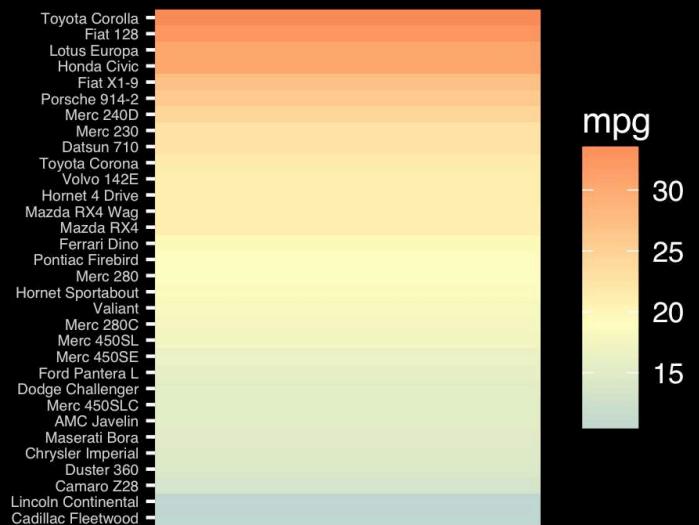
Key Point: The most important measurement should exploit the highest ranked encoding possible

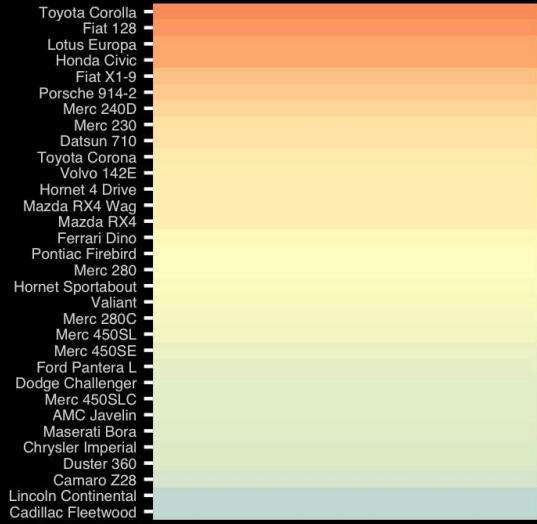
- 
- Position along a common scale
 - Position on identical but nonaligned scales
 - Length
 - Angle or Slope
 - Area
 - Volume or Density or **Color saturation/hue**



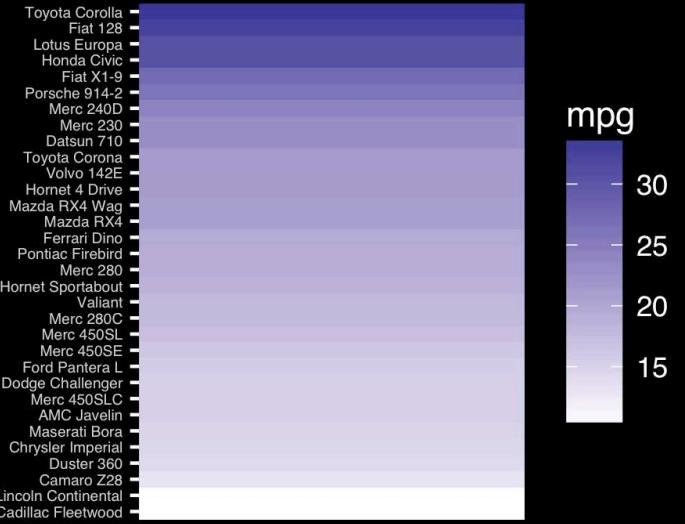


Observation: Alphabetical is almost never the correct ordering of a categorical variable.



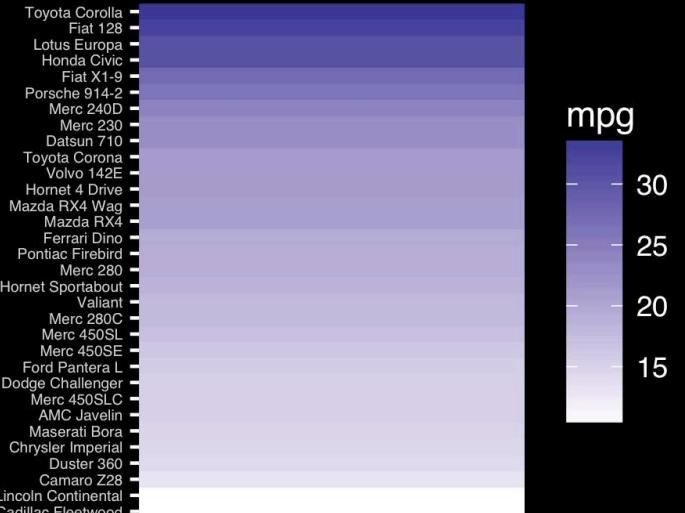


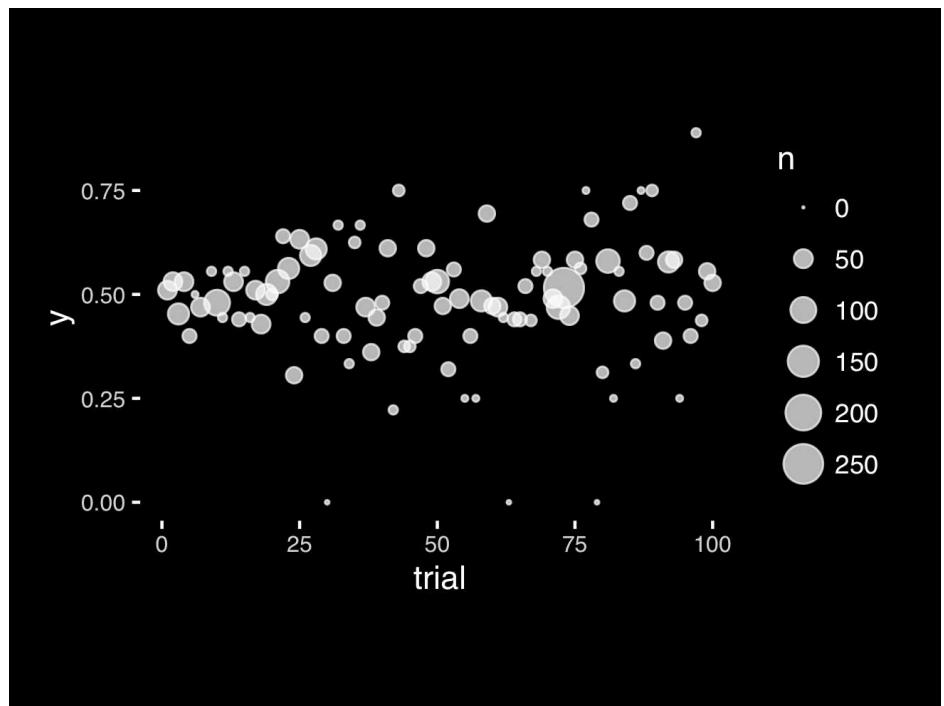
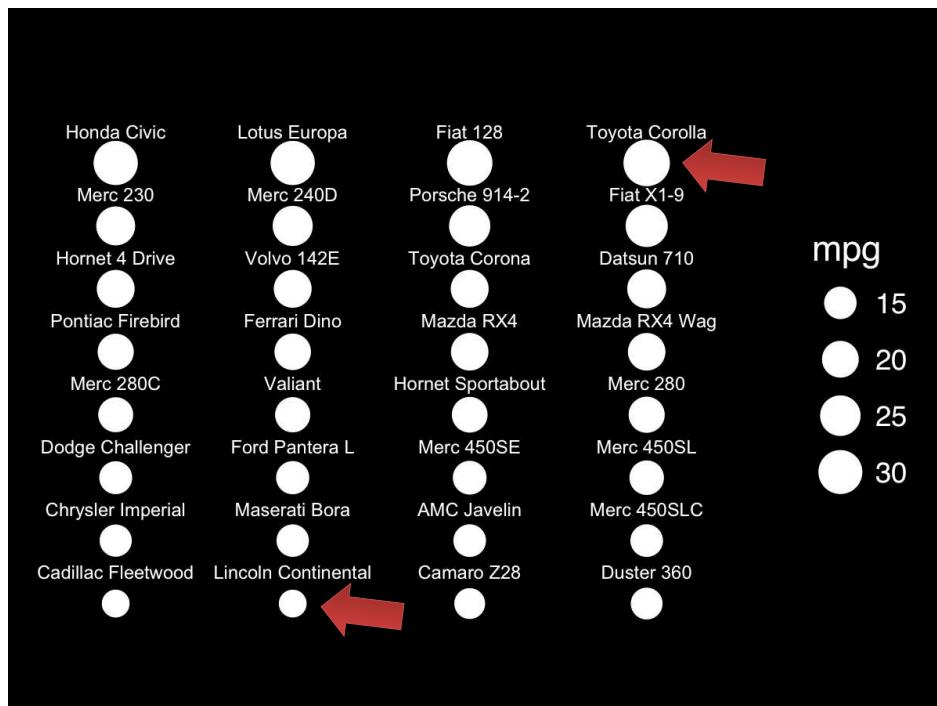
If we did not have the legend would you know which was low or high mpg?



The most important measurement should exploit the highest ranked encoding possible.

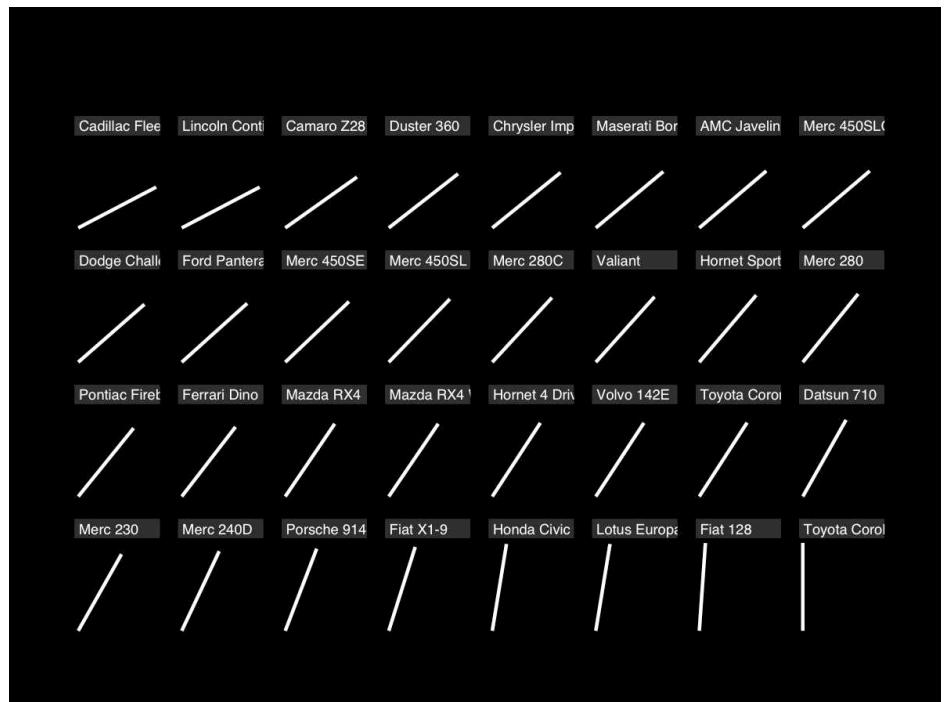
- Position along a common scale
- Position on identical but nonaligned scales
- Length
- Angle or Slope
- **Area**
- Volume or Density or Color saturation/hue

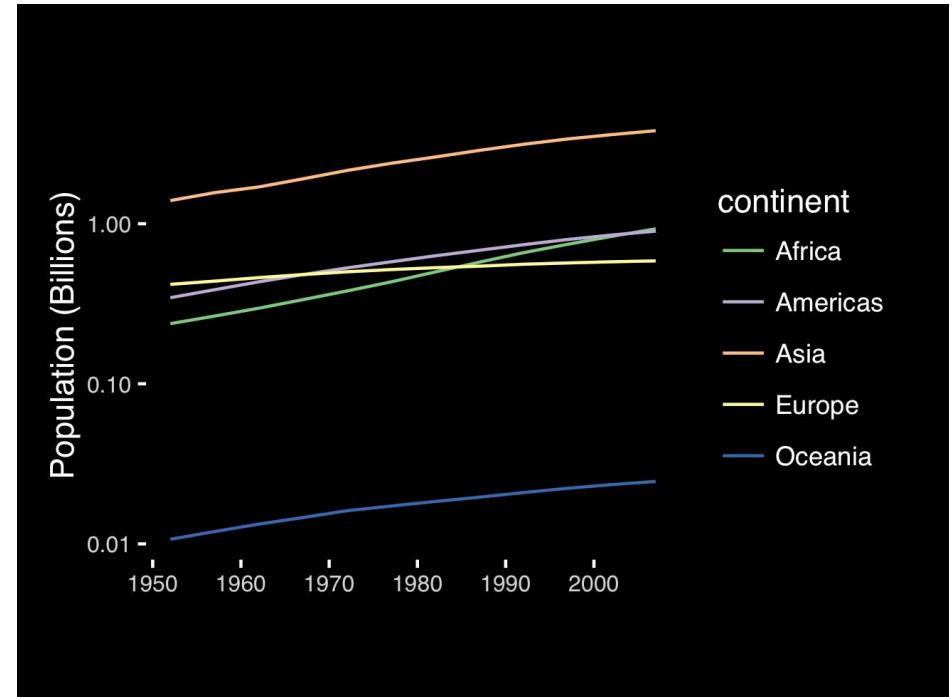
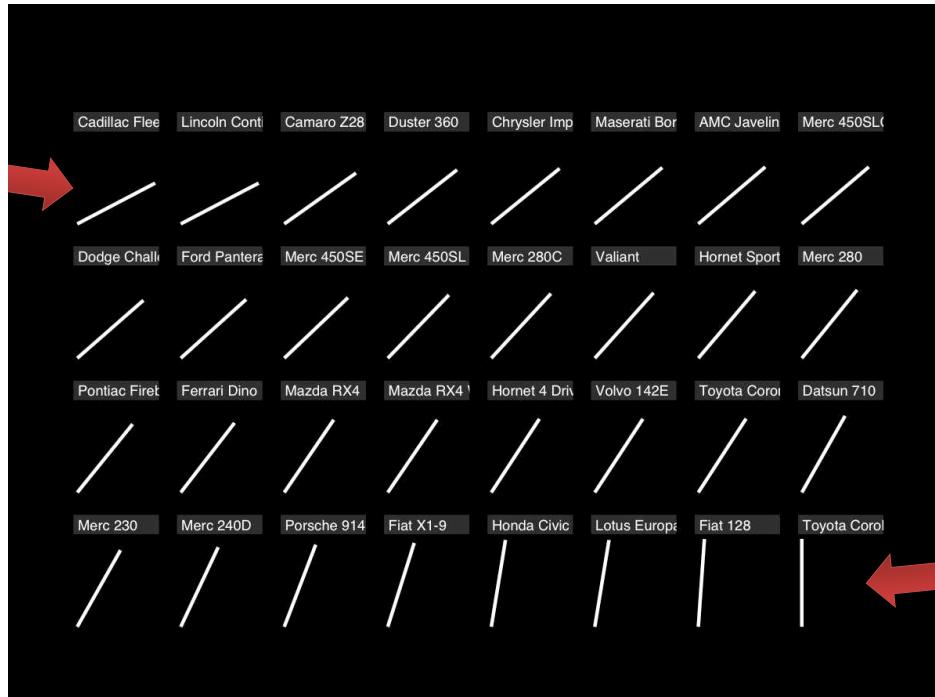




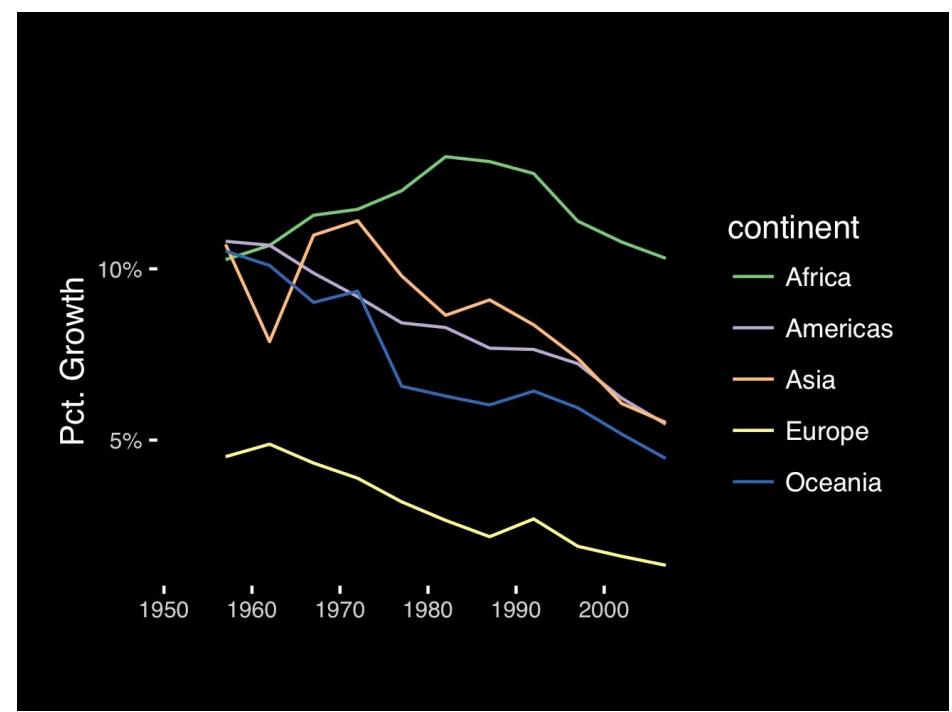
The most important measurement should exploit the highest ranked encoding possible.

- Position along a common scale
- Position on identical but nonaligned scales
- Length
- Angle or Slope
- Area
- Volume or Density or Color saturation/hue





If growth (slope) is important, plot it directly.



The most important measurement should exploit the highest ranked encoding possible.

- Position along a common scale
- Position on identical but nonaligned scales
- Length
- **Angle or Slope**
- Area
- Volume or Density or Color saturation/hue

Observation: Pie charts are ALWAYS a mistake.

Apart from MPAs :-)

Piecharts are the information visualization equivalent of a roofing hammer to the frontal lobe. They have no place in the world of grownups, and occupy the same semiotic space as short pants, a runny nose, and chocolate smeared on one's face. They are as professional as a pair of assless chaps.

<http://blog.codahale.com/2006/04/29/google-analytics-the-goggles-they-do-nothing/>

Piecharts are the information visualization equivalent of a roofing hammer to the frontal lobe. They have no place in the world of grownups, and occupy the same semiotic space as short pants, a runny nose, and chocolate smeared on one's face. **They are as professional as a pair of assless chaps.**

<http://blog.codahale.com/2006/04/29/google-analytics-the-goggles-they-do-nothing/>

Who do you think did a better job in tonight's debate?

Among Republicans

Among Democrats

Donald Trump 47%
Hillary Clinton 53%

Among Republicans

Among Democrats

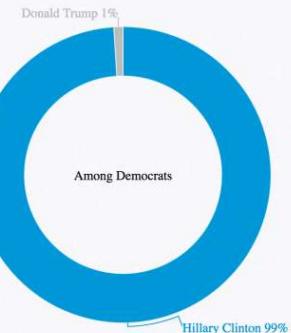
Share

POLITICO

Who do you think did a better job in tonight's debate?

Among Republicans

Among Democrats



Share

POLITICO

Tables are preferable to graphics for many small data sets. A table is nearly always better than a dumb pie chart; the only thing worse than a pie chart is several of them, for then the viewer is asked to compare quantities located in spatial disarray both within and between pies... Given their low data-density and failure to order numbers along a visual dimension, **pie charts should never be used.**

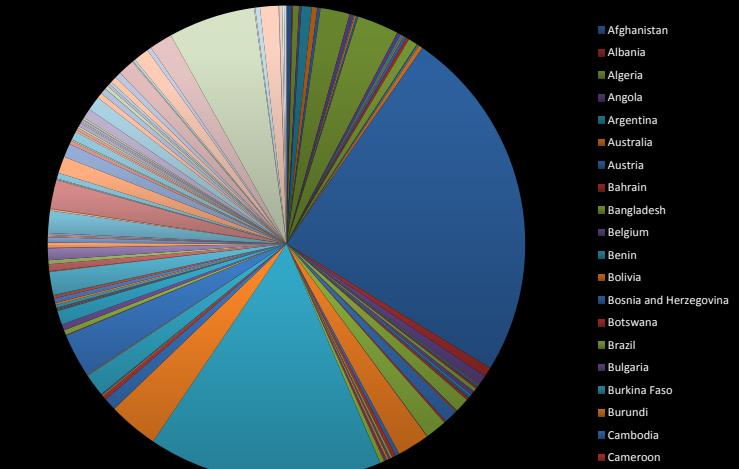
-Edward Tufte, *The Visual Display of Quantitative Information*

Tables are preferable to graphics for many small data sets. A table is nearly always better than a dumb pie chart; the only thing worse than a pie chart is several of them, for then the viewer is asked to compare quantities located in spatial disarray both within and between pies... Given their low data-density and failure to order numbers along a visual dimension, pie charts should never be used.

-Edward Tufte, *The Visual Display of Quantitative Information*

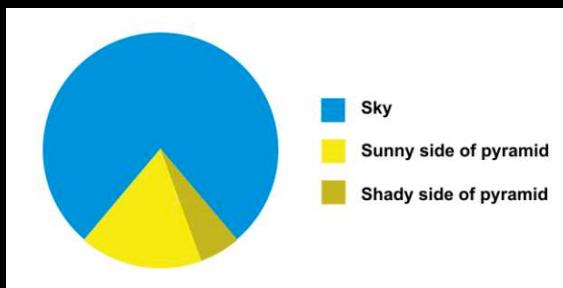
Who do you think did a better job in tonight's debate?

	Clinton	Trump
Among Democrats	99%	1%
Among Republicans	53%	47%



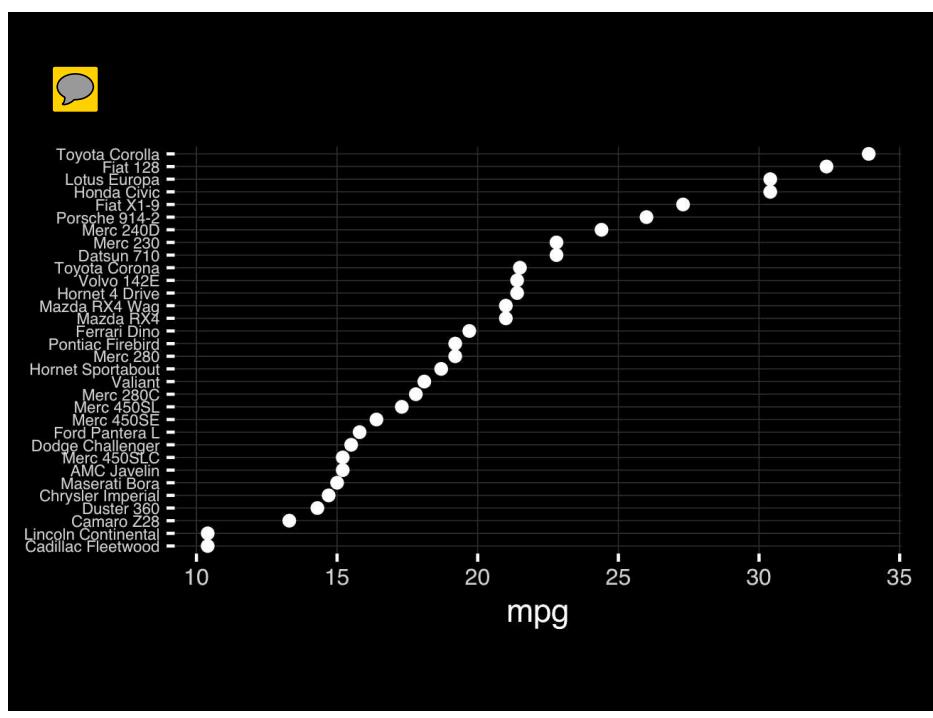
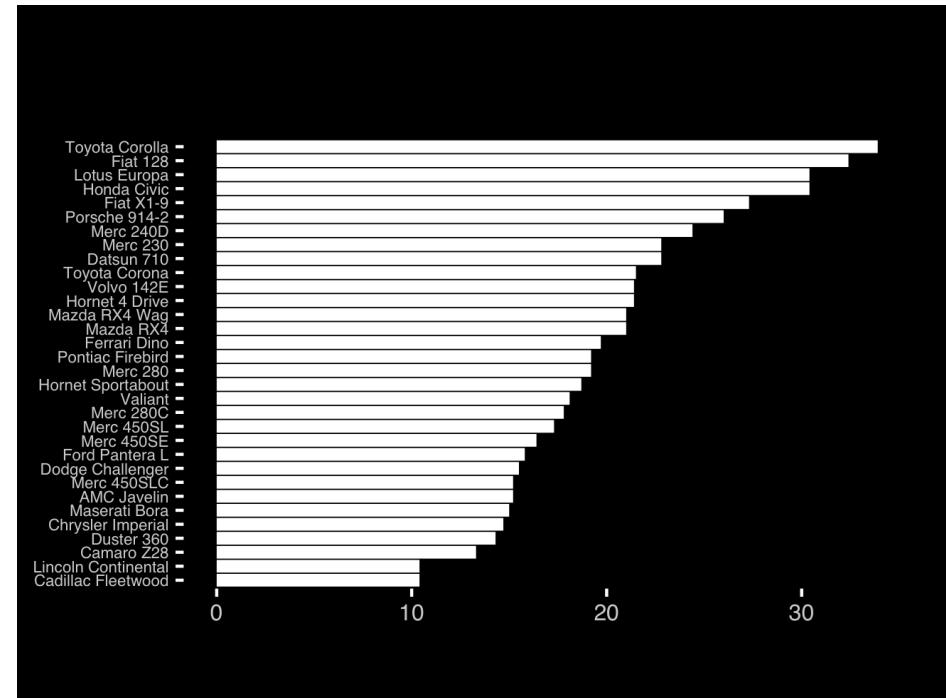
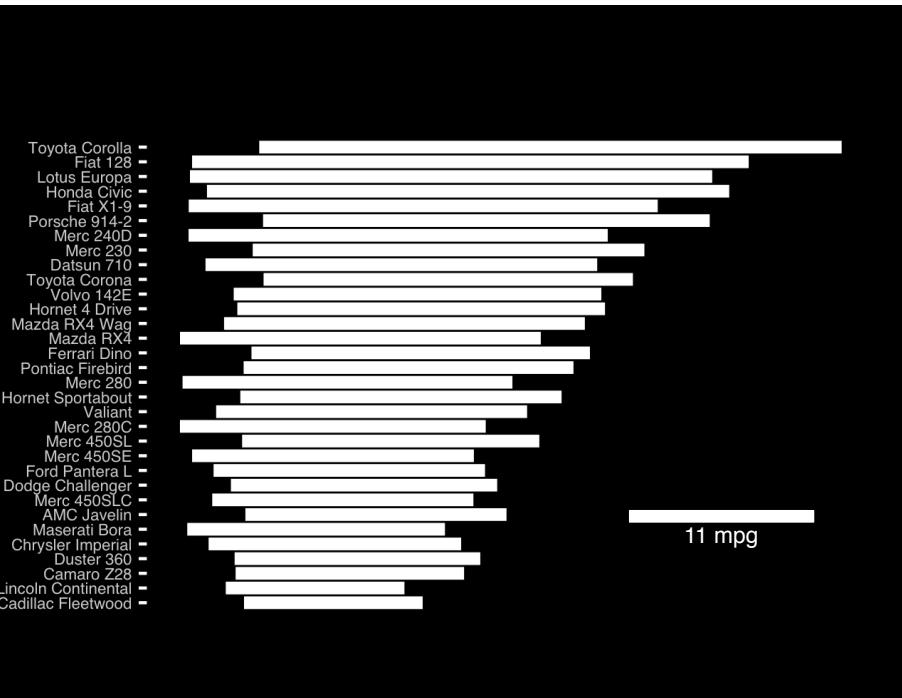
Afghanistan
Albania
Algeria
Angola
Argentina
Australia
Austria
Bahrain
Bangladesh
Belgium
Benin
Bolivia
Bosnia and Herzegovina
Botswana
Brazil
Bulgaria
Burkina Faso
Burundi
Cambodia
Cameroon

All good pie charts are jokes...



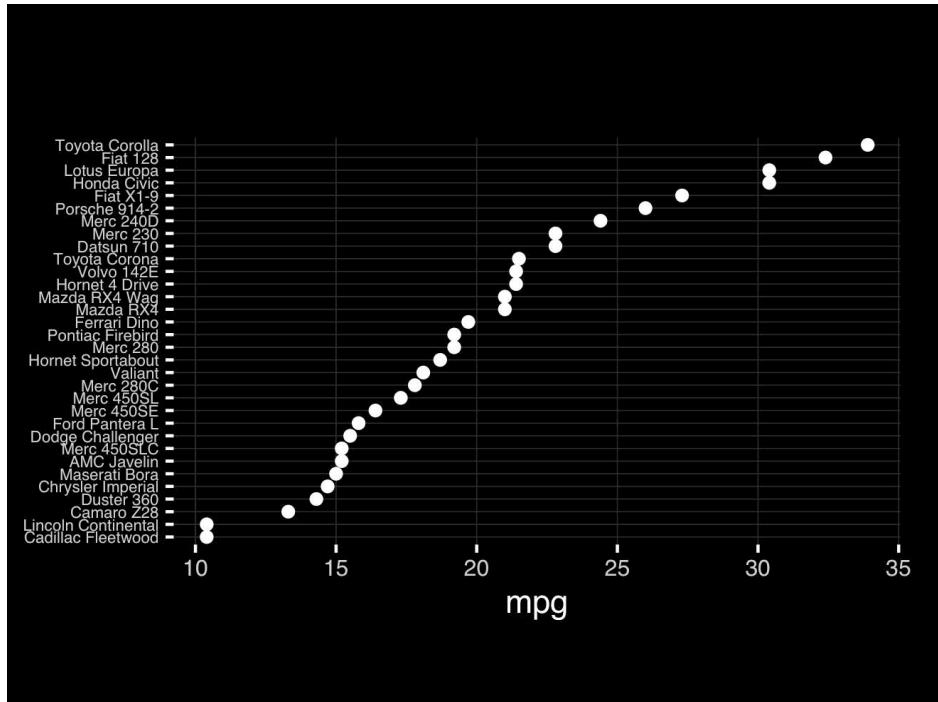
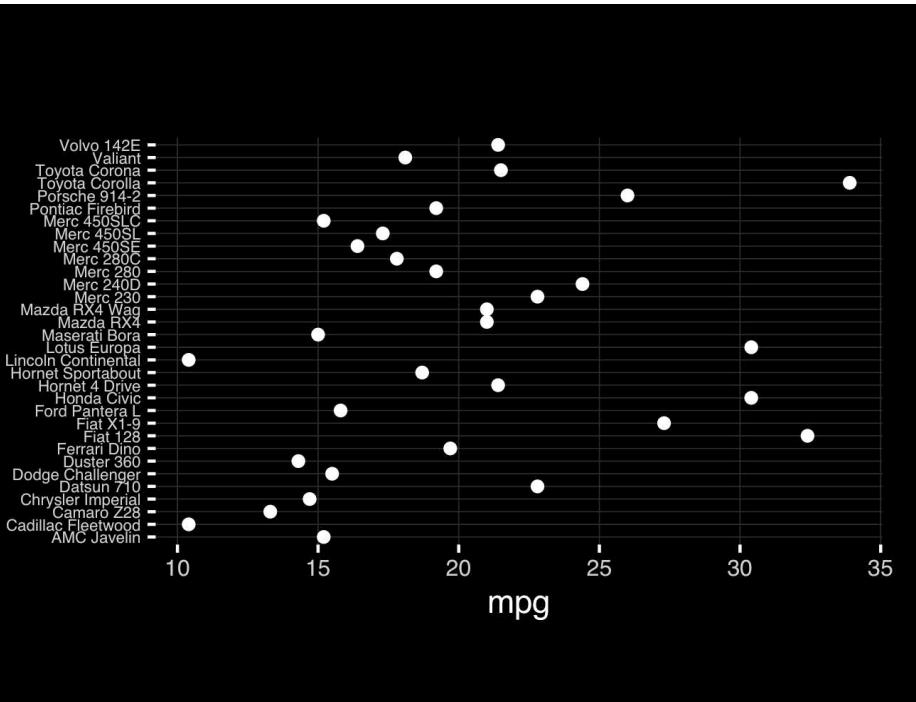
The most important measurement should exploit the highest ranked encoding possible.

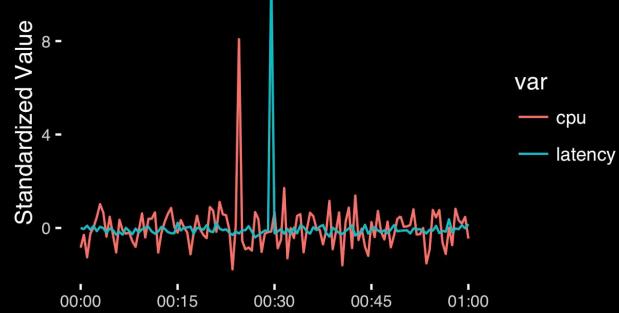
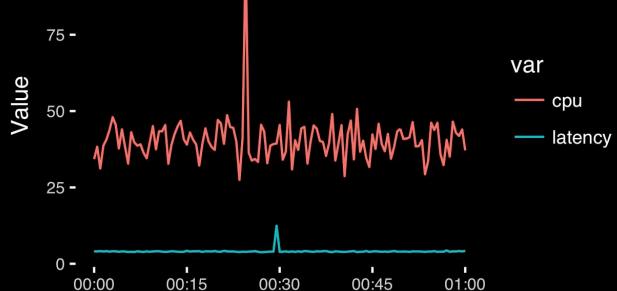
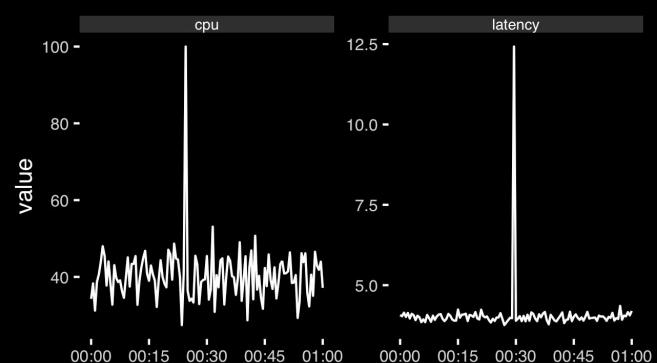
- Position along a common scale
- Position on identical but nonaligned scales
- Length
- Angle or Slope
- Area
- Volume or Density or Color saturation/hue



The most important measurement should exploit the highest ranked encoding possible.

- Position along a common scale
- Position on identical but nonaligned scales
- Length
- Angle or Slope
- Area
- Volume or Density or Color saturation/hue





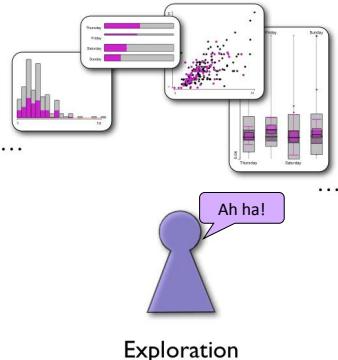
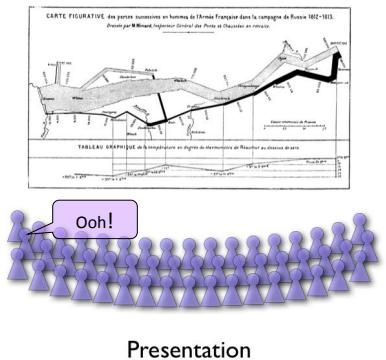
Today's Learning Goals

- Appreciate the major elements of **exploratory data analysis** and why it is important to visualize data.
- Be conversant with **data visualization best practices** and understand how good visualizations optimize for the human visual system.
- Be able to generate informative graphical displays including **scatterplots**, **histograms**, **bar graphs**, **boxplots**, **dendrograms** and **heatmaps** and thereby gain exposure to the extensive graphical capabilities of R.
- Appreciate that you can build even more complex charts with **ggplot** and additional R packages such as **rgl**.

Different graphs for different purposes

Exploratory graphs: many images for a narrow audience (you!)

Presentation graphs: single image for a large audience



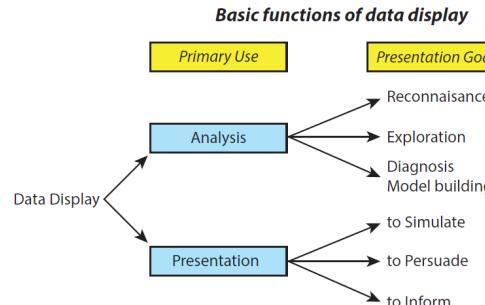
17

Roles of graphics in data analysis

- Graphs (& tables) are forms of communication:
 - What is the audience?
 - What is the message?

Analysis graphs: design to see patterns, trends, aid the process of data description, interpretation

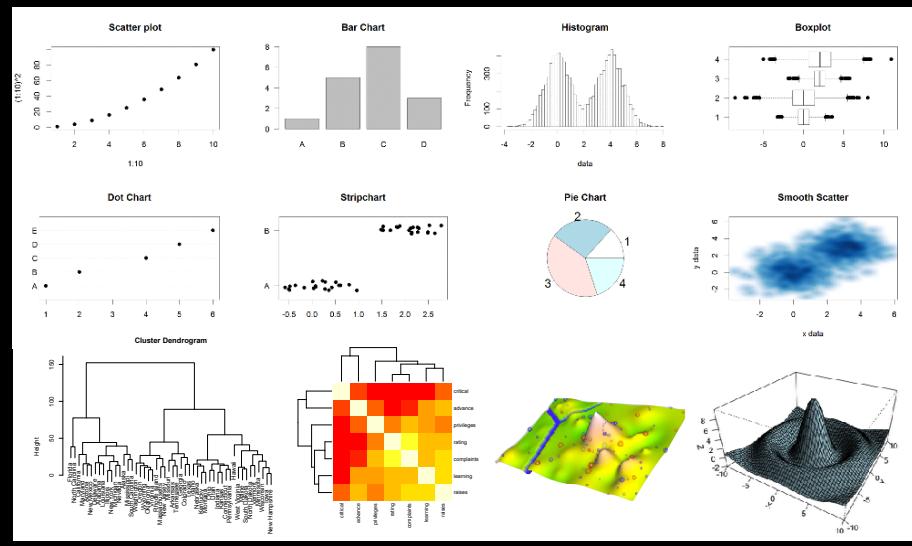
Presentation graphs: design to attract attention, make a point, illustrate a conclusion



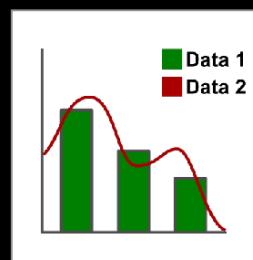
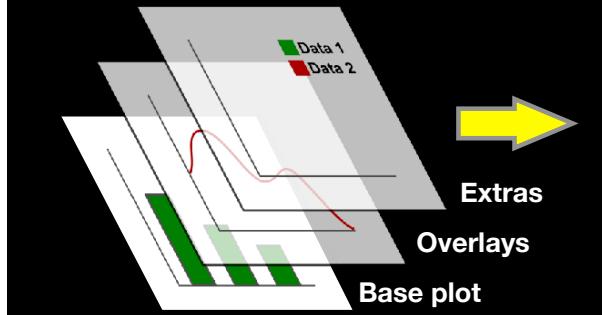
Exploratory (for you!)

Info for others,
publications &
sharing etc.

Core R Graph Types



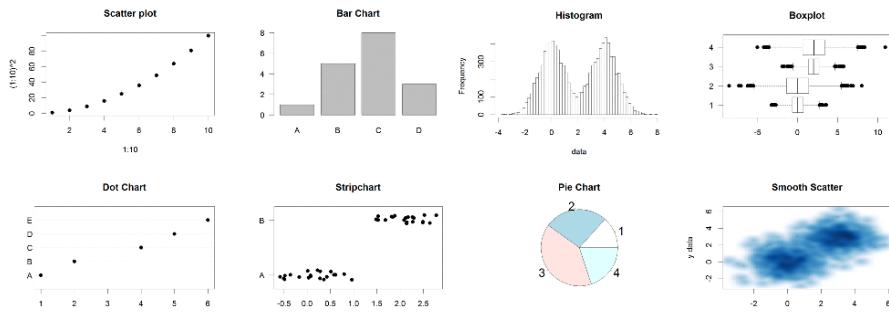
The R Painters Model



Side-Note: "Red and green should never be seen"



Core Graph Types



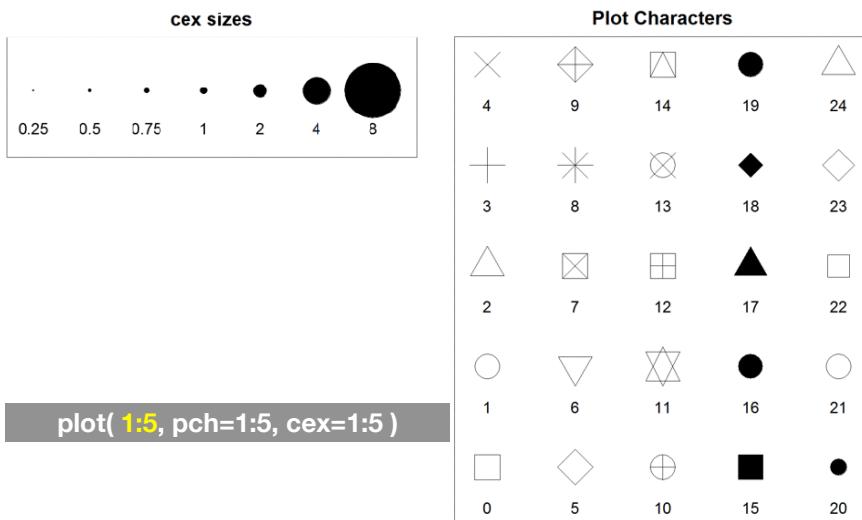
- Local options to change a specific plot
- Global options to affect all graphs

Common Options

- Axis scales
 - `xlim c(min,max)`
 - `ylim c(min,max)`
- Axis labels
 - `xlab(text)`
 - `ylab(text)`
- Plot titles
 - `main(text)`
 - `sub(text)`
- Plot characters
 - `pch(number)`
 - `cex(number)`

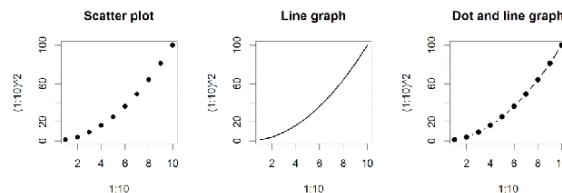
- Local options to change a specific plot
- Global options to affect all graphs

Plot Characters



Plot Type Specific Options

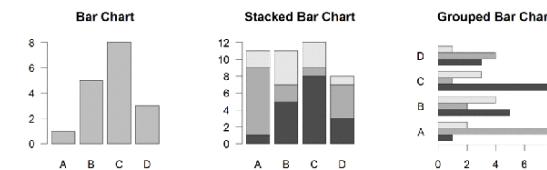
Plot (scatterplots and line graphs)



- Input: Almost anything. 2 x Vectors
- Output: Nothing
- Options:
 - type l=line, p=point, b=line+point
 - lwd line width (thickness)
 - lty line type (1=solid,2=dashed,3=dotted etc.)

```
plot( c(1:10)^2, typ="b", lwd=4, lty=3 )
```

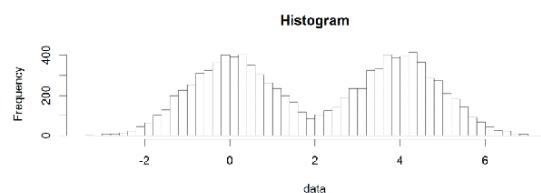
Barplot (bar graphs)



- Input: Vector (single) or Matrix (stack or group)
- Output: Bar centre positions
- Options:
 - names.arg Bar labels (if not from data)
 - horiz=TRUE Plot horizontally
 - beside=TRUE Plot multiple series as a group not stacked

```
barplot(VADeaths, beside = TRUE)
```

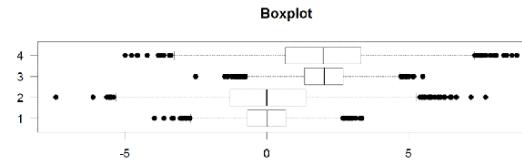
Hist (histograms)



- Input: Vector
- Output: Summary of binned data
- Options:
 - breaks Number or limits of bins
 - probability Y axis is probability, not freq
 - labels Per bin text labels

```
hist( c( rnorm(1000,0), rnorm(1000,4) ), breaks=20 )
```

Boxplot



- Input: Vector, List or formula (data~factor)
- Output: Summary of the boxplot parameters
- Options:
 - range Sensitivity of whiskers
 - varwidth Width represents total observations
 - horizontal Plot horizontally

```
boxplot( cbind( rnorm(1000,0), rnorm(1000,4) ) )
```

Controlling plot area options with `par`

Par

- The `par` function controls global parameters affecting all plots in the current plot area
- Changes affect all subsequent plots
- Many `par` options can also be passed to individual plots

`?par`

Par examples

- Reading current value
 - `par()$cex`
- Setting a value
 - `par(cex=1.5) -> old.par`
- Restoring a value
 - `par(old.par)`
 - `dev.off()`

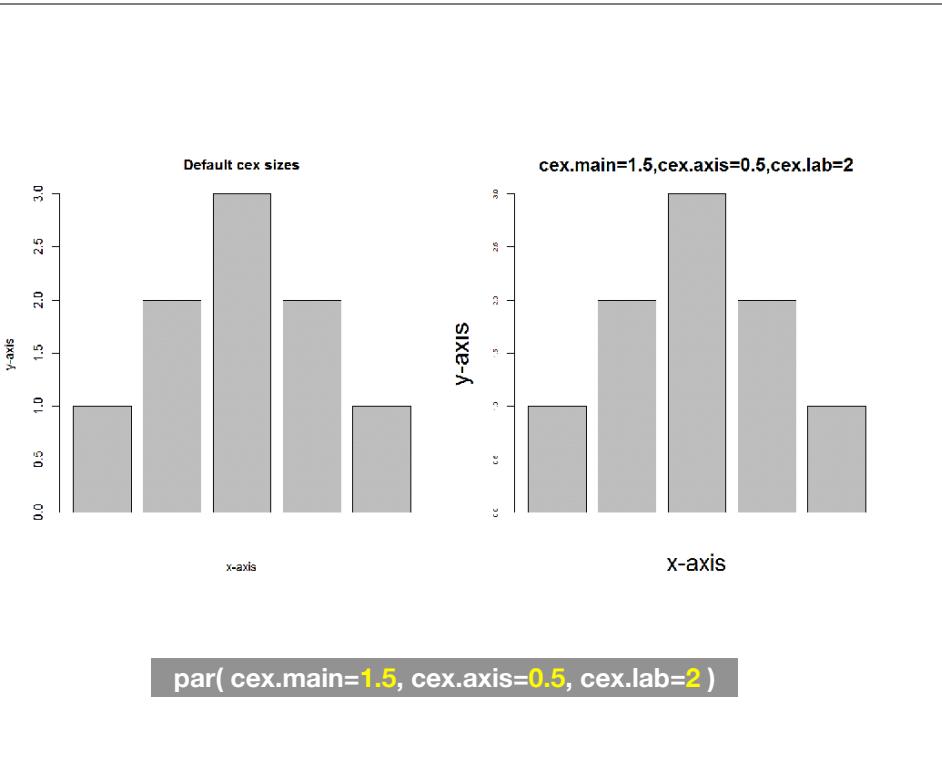
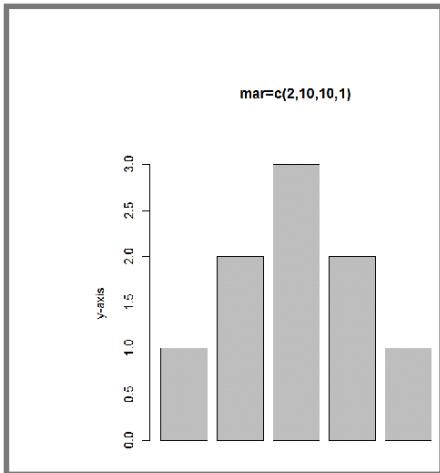
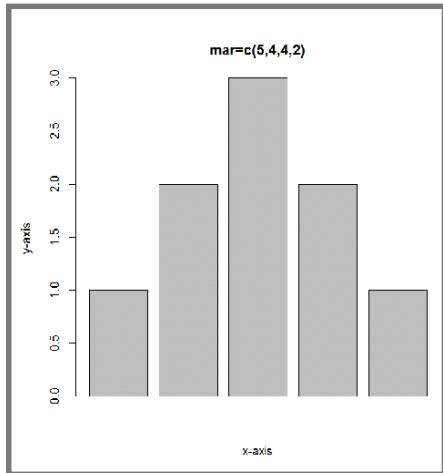
Par options

- Margins
 - `mai` (set margins in inches)
 - `mar` (set margins in number of lines)
 - `mex` (set lines per inch)
 - 4 element vector (bottom, left, top, right)
- Warning
 - Error in `plot.new()` : figure margins too large

`par(mar=c(2, 10, 1, 1))`

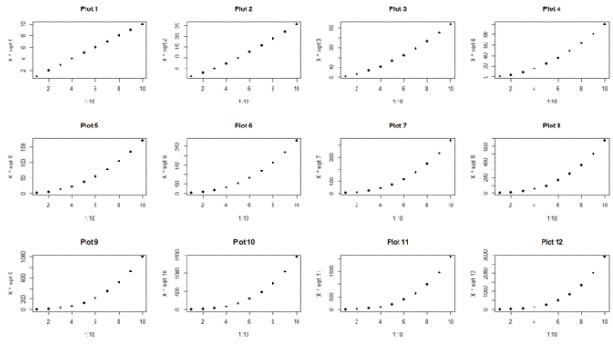
Par options

- Fonts and labels
 - cex - global char expansion
 - cex.axis
 - cex.lab
 - cex.main
 - cex.sub



Par options

- Multi-panel
 - par(mfrow(rows,cols))



Exercise 1

Using Color

Specifying colors

- Hexadecimal strings
 - #FF0000 (red)
 - #0000FF (blue)
 - #CC00CC (purple)
- Controlled names
 - “red” “green” etc.
 - colors()

Built in color schemes

- Functions to generate colors
- Pass in number of colors to make
- Functions:
 - rainbow()
 - heat.colors()
 - cm.colors()
 - terrain.colors()
 - topo.colors()

rainbow(7)



Color Packages

- **Color Brewer**
 - Set of pre-defined, optimized palettes
 - `library(RColorBrewer)`
 - `brewer.pal(n_colours, palette)`
- **ColorRamps**
 - Create smooth palettes for ramped color
 - Generates a function to make actual color vectors
 - `colorRampPalette(c("red","white","blue"))`
 - `colorRampPalette(c("red","white","blue"))(5)`

Applying Color to Plots

- Vector of numbers or specified colors passed to the `col` parameter of a plot function
- Vector of factors used to divide the data
 - Colors will be taken from the set color palette
 - Can read or set using `pallete` function
 - `palette()`
 - `palette(brewer.pal(9, "Set1"))`

```
plot( 1:5, col=1:5, pch=15, cex=2)
```

Dynamic use of color

- **Coloring by density**
 - Pass data and palette to `densCols()`
 - Vector of colors returned
- **Coloring by value**
 - Need function to map values to colors

<https://www.rdocumentation.org/packages/grDevices/versions/3.4.3/topics/densCols>

Exercise 2

Q: 2B. `stringsAsFactors = TRUE` vs `stringsAsFactors = FALSE`

(POOR!) Color Mapping Function

```
map.colors <- function(value,high.low,palette) {  
  proportion <- ((value-high.low[1])/(high.low[2]-high.low[1]))  
  index <- round ((length(palette)-1)*proportion)+1  
  return (palette[index])  
}
```

Talking point:

- Can you figure out what this function it is supposed to do?
- What format should the inputs be in order to work?
- How could we improve this function?

Exercise 2C Revisited

- Open your previous Lecture5 RStudio **project** (and your saved **R script**)
- Locate and open in RStudio the downloaded file `color_to_value_map.r`
- This is an example of a poorly written function typical of something you might get from a lab mate that knows some R...

1. What are the function inputs?

```
map.colors2 <- function(x, high.low, palette) {  
  proportion <- ((x - high.low[1])/(high.low[2] - high.low[1]))  
  index <- round( (length(palette)-1) * proportion )+1  
  return(palette[index])  
}
```

Let's first space things out so it is easier for us to read and then change to use `x` as our numeric input vector.

1. What are the function inputs?

```
map.colors2 <- function(x, high.low, palette) {  
  proportion <- ((x - high.low[1])/(high.low[2] - high.low[1]))  
  index <- round( (length(palette)-1) * proportion )+1  
  return(palette[index])  
}
```

Let's first space things out so it is easier for us to read and then change to use `x` as our numeric input vector.

We can guess that `high.low` is a two element numeric vector and `palette` is probably a vector of colors

2. What is the function doing?

```
map.colors2 <- function(x, high.low, palette) {  
  # Determine percent values of the 'high.low' range  
  proportion <- ((x - high.low[1])/(high.low[2] - high.low[1]))  
  index <- round( (length(palette)-1) * proportion )+1  
  return(palette[index])  
}
```

Let's add a **comment** to explain the logic of the first line

2. What is the function doing?

```
map.colors2 <- function(x, high.low, palette) {  
  # Determine percent values of the 'high.low' range  
  precent <- ((x - high.low[1])/(high.low[2] - high.low[1]))  
  index <- round( (length(palette)-1) * precent )+1  
  return(palette[index])  
}
```

Let's change the object name from `proportion` to `precent` so it is more meaningful for us. Remember to change it everywhere ;-)

2. What is the function doing?

```
map.colors2 <- function(x, high.low, palette) {  
  # Determine percent values of the 'high.low' range  
  precent <- ((x - high.low[1])/(high.low[2] - high.low[1]))  
  #index <- round( (length(palette)-1) * precent )+1  
  index <- round( length(palette) * precent )  
  return(palette[index])  
}
```

Perhaps we can simplify the next line, which determines the corresponding index position in the color 'palette' vector?

2. What is the function doing?

```
map.colors2 <- function(x, high.low, palette) {  
  
  # Determine precent values of the 'high.low' range  
  precent <- ((x - high.low[1])/(high.low[2] - high.low[1]))  
  
  #index <- round( (length(palette)-1) * precent )+1  
  index <- round( length(palette) * precent )  
  
  return(palette[index])  
}
```

Doh! What happens if our precent value is zero or very small?

We will get an `index` value of zero, will cause a problem when accessing `palette[index]` in the last line

2. What is the function doing?

```
map.colors2 <- function(x, high.low, palette) {  
  
  # Determine precent values of the 'high.low' range  
  precent <- ((x - high.low[1])/(high.low[2] - high.low[1]))  
  
  # Find correspoding index position in the color 'palette'  
  # note catch for 0 precent values to 1  
  index <- round( (length(palette)-1) * precent )+1  
  
  return(palette[index])  
}
```

Add a comment again to describe the logic of what our code is doing

3. How could we improve this function?

```
map.colors2 <- function(x, high.low, palette) {  
  
  ## Description: Map the values of the input vector 'x'  
  ## to the input colors vector 'palette'  
  
  # Determine precent values of the 'high.low' range  
  precent <- ((x - high.low[1])/(high.low[2] - high.low[1]))  
  
  # Find correspoding index position in the color 'palette'  
  # note catch for 0 precent values to 1  
  index <- round( (length(palette)-1) * precent )+1  
  
  return(palette[index])  
}
```

Make more user friendly in lots of ways including adding **more description**, input argument defaults, error checking of inputs etc.

3. How could we improve this function?

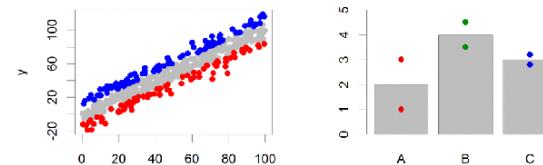
```
map.colors3 <- function(x,  
                        low.high = range(x),  
                        palette = cm.colors(100)) {  
  
  ## Description: Map the values of the input vector 'x'  
  ## to the input colors vector 'palette'  
  
  # Determine precent values of the 'high.low' range  
  precent <- ((x - low.high[2])/(low.high[1] - low.high[2]))  
  
  # Find correspoding index position in the color 'palette'  
  # note catch for 0 precent values to 1  
  index <- round( (length(palette)-1) * precent )+1  
  
  return(palette[index])  
}
```

Make more user friendly in lots of ways including adding **more description**, **input argument defaults**, error checking of inputs etc.

Plot Overlays

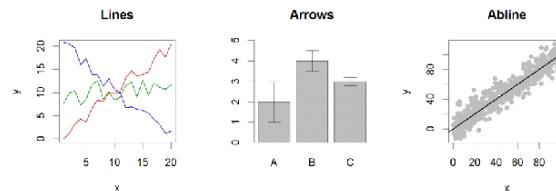
Exercise 3

Points



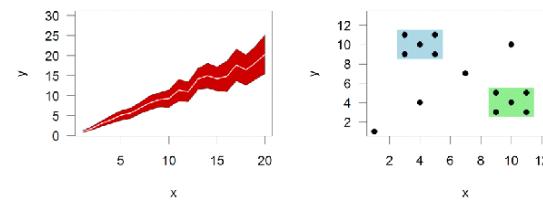
- Input: 2 Vectors (x and y positions)
- Options:
 - pch
 - cex

Lines / Arrows / Abline



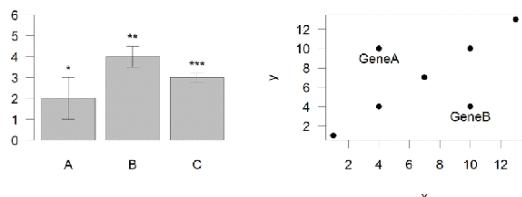
- Input:
 - Lines 2 vectors (x and y)
 - Arrows 4 vectors (x_0, x_1, y_0, y_1)
 - Abline Intercept and slope (or correlation object)
- Options:
 - lwd
 - angle (arrows)

Polygon (shaded areas)



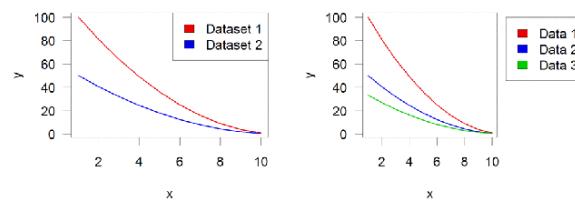
- Input:
 - 2 vectors (x and y) for bounding region
- Options:
 - col

Text (in plot text)



- Input:
 - Text, x, y
- Options:
 - adj (x and y offsets)
 - pos (auto offset 1=below,2=left,3=above, 4=right)

Legend



- Input:
 - Position (x,y or “topright”, “bottomleft” etc)
 - Text labels
- Options:
 - fill (colours for shaded boxes)
 - xpd=NA (draw outside plot area)

Exercise 3

[Muddy Point Assessment Form Link](#)

Homework!

New [DataCamp](#) Assignments

- Introduction to R Markdown
- Functions
- Loops

[Muddy Point Assessment Form Link](#)