Amsterdam University of Applied Sciences | DIGITAL SOCIETY SCHOOL

DATA-DRIVEN TRANSFORMATION

R

FIRST VIZ

TRANS DATA-DRIVEN FORMATION

TABLE OF CONTENT

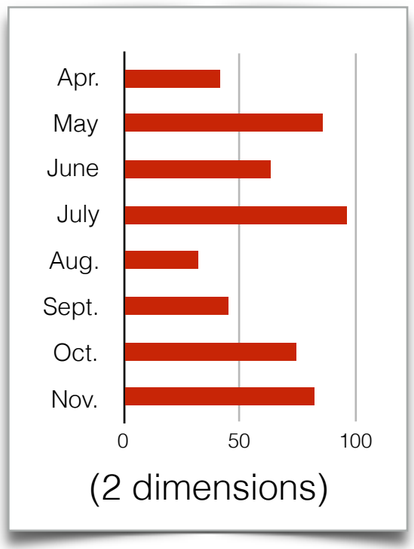
BASICS OF DATA VISUALIZATION

VISUAL DIMENSIONS

Visual features can be used to represent different variables

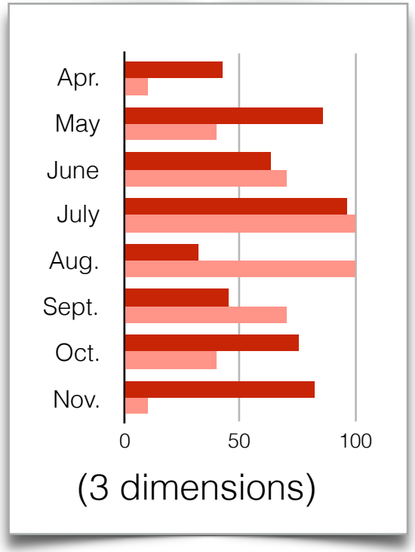
Variables are the data dimensions encoded into visual dimensions

Length



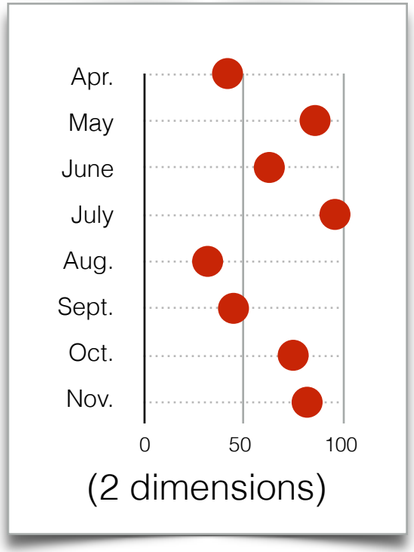
(2 dimensions)

Color



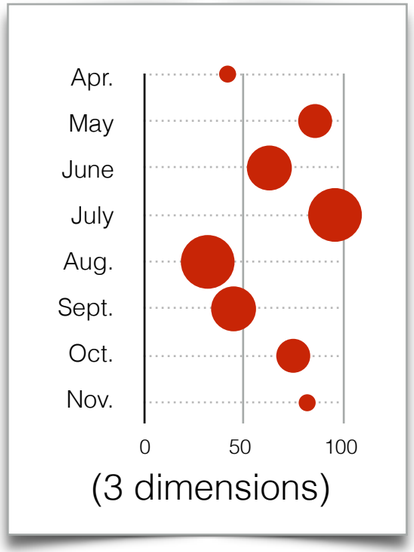
(3 dimensions)

Position



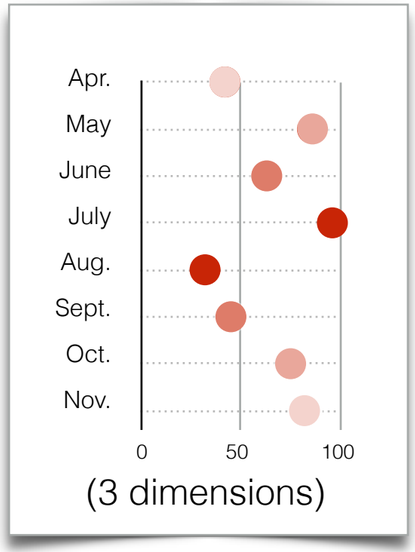
(2 dimensions)

Size



(3 dimensions)

Transparency



(3 dimensions)

VISUAL DIMENSIONS

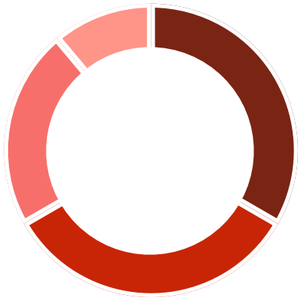
Visual features offer many options to represent the same data…

…but some visual features are harder to compare with human eyes

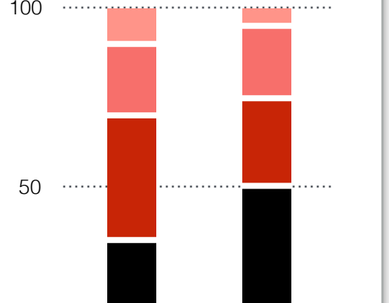
Area

Pie Chart

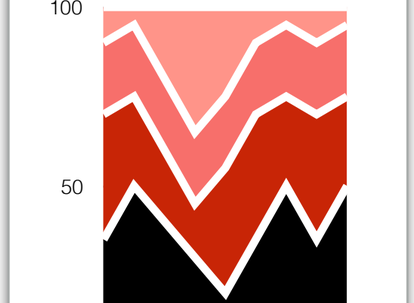
Length



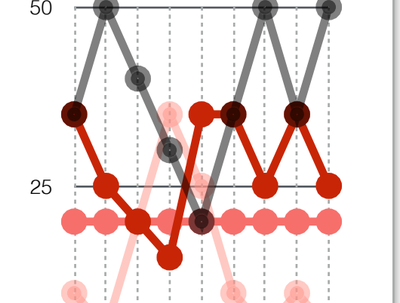
Donut Chart



Area



Position

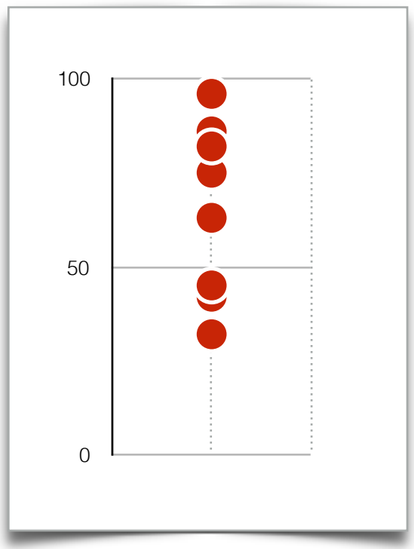


MULTIPLE DIMENSIONS

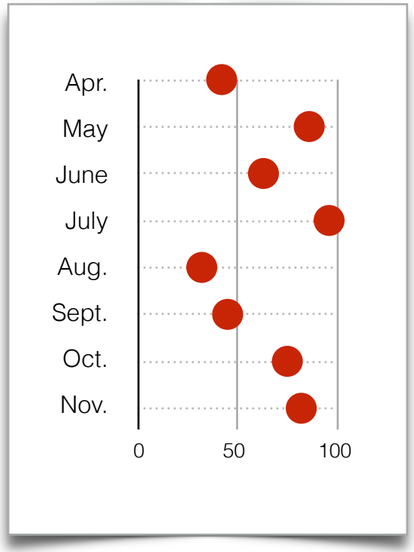
To add a data dimension,
add a visual dimension

…but the graph becomes complex

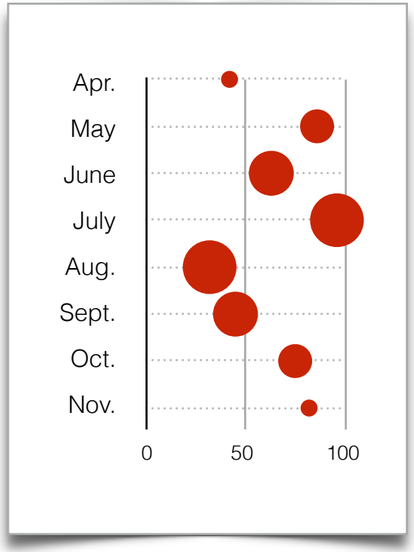
1 Dimension



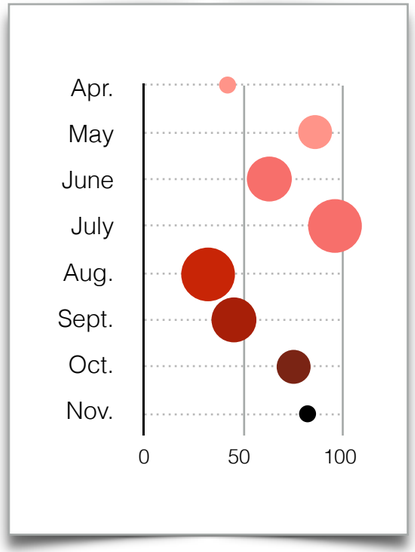
2 Dimensions



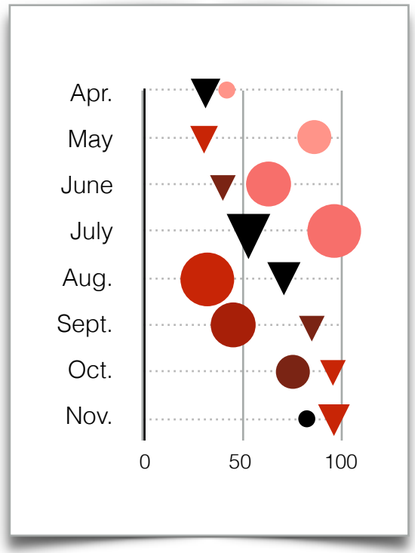
3 Dimensions



4 Dimensions

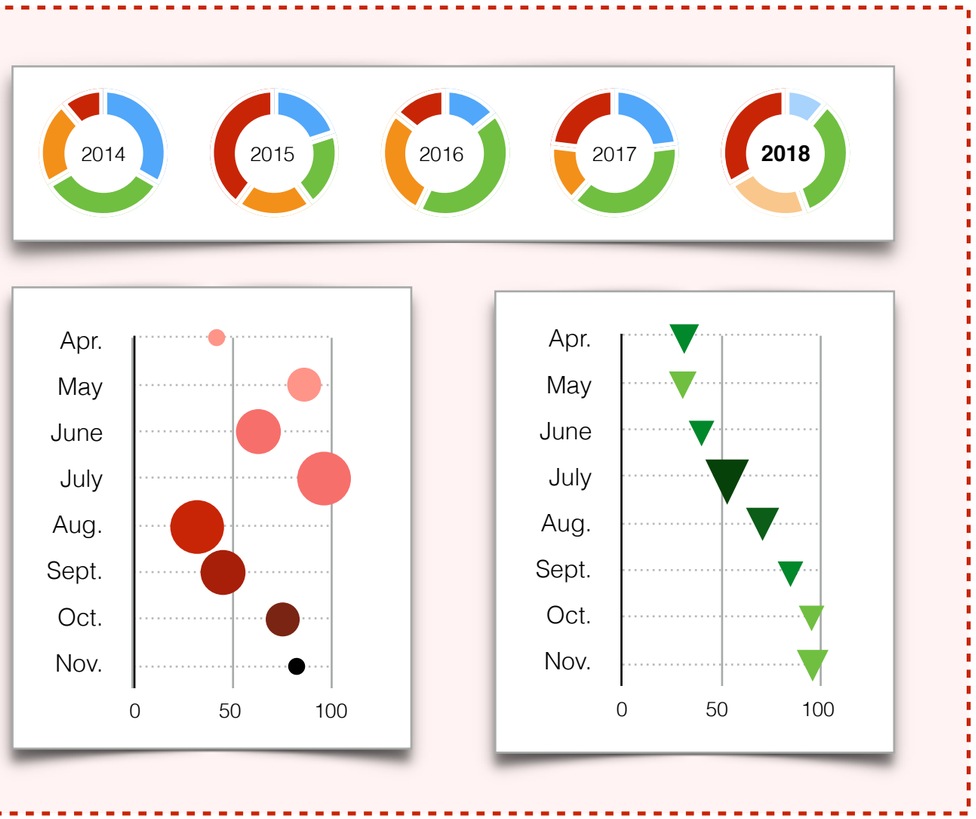


5 Dimensions

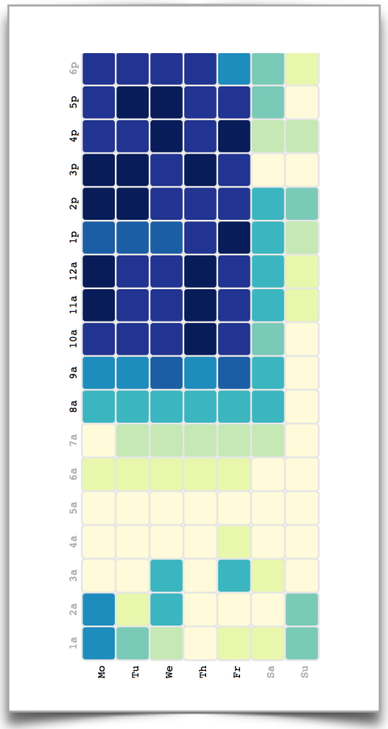


MULTIPLE DIMENSIONS

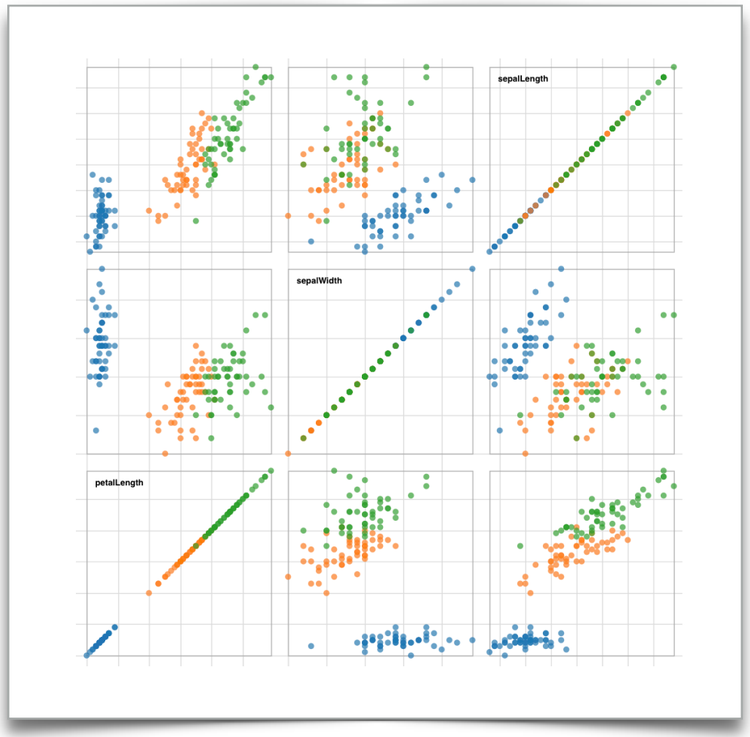
Multiple Views



Heatmap

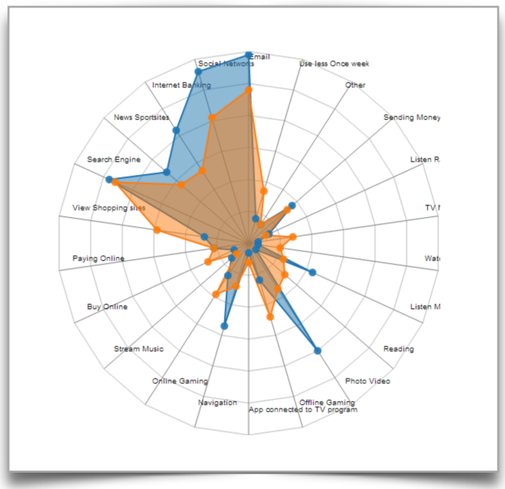


Scatterplot Matrix

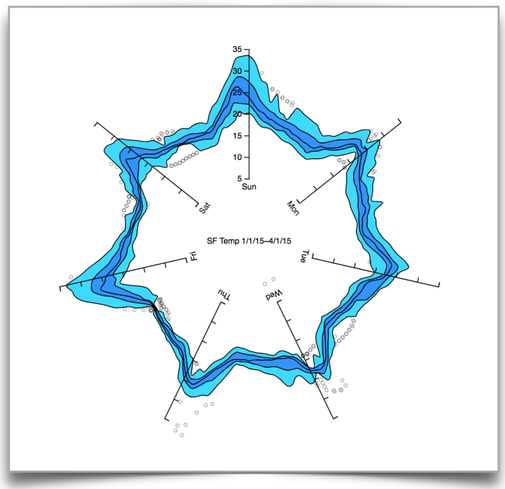


MULTIPLE DIMENSIONS

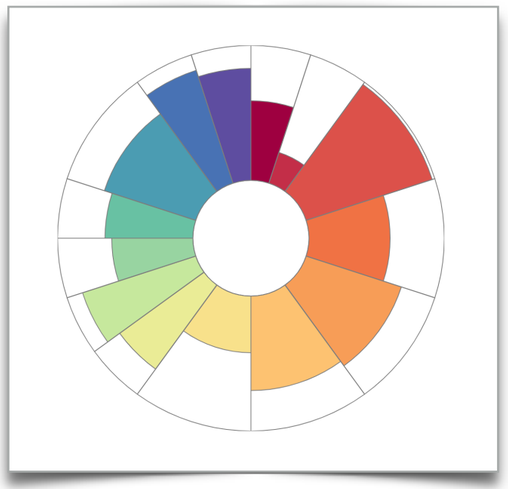
Radar Chart



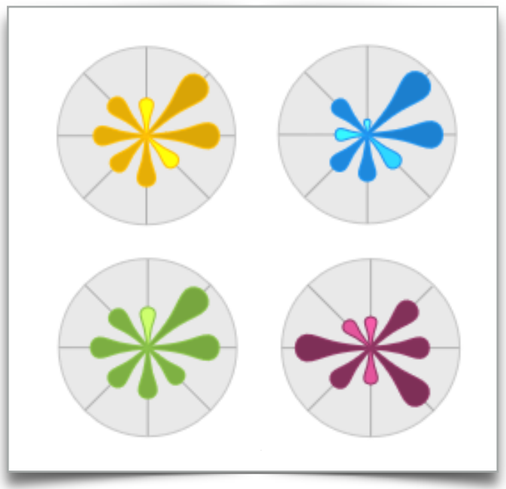
Cyclic Graph



Radial Graph

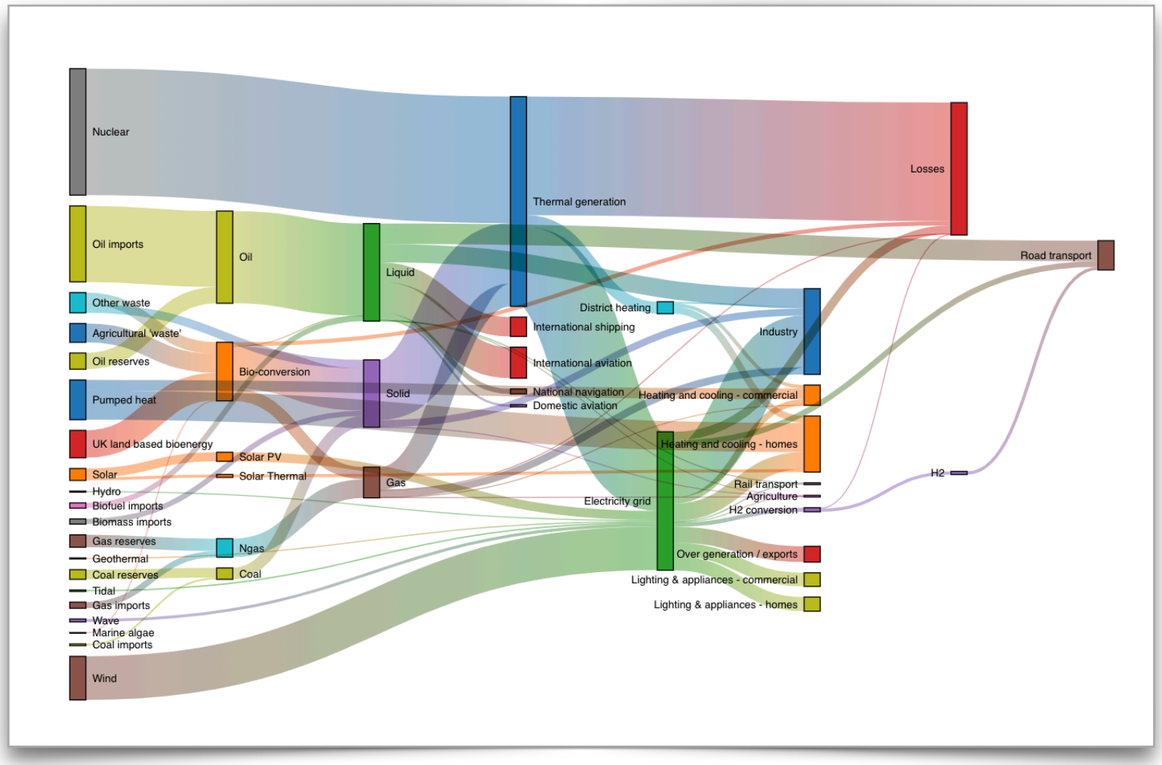


Glyph

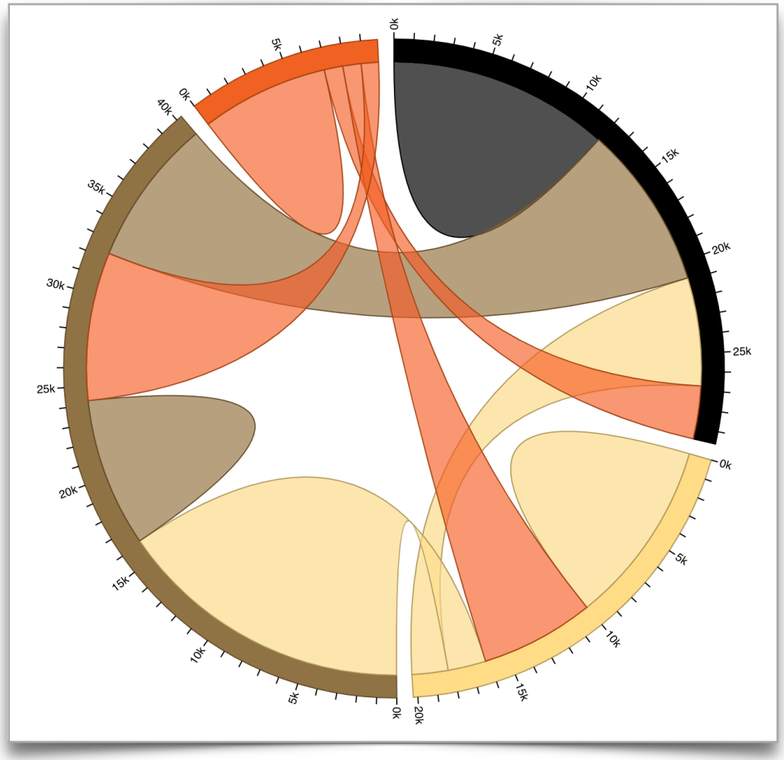


NETWORK & FLOW

Sankey Diagram

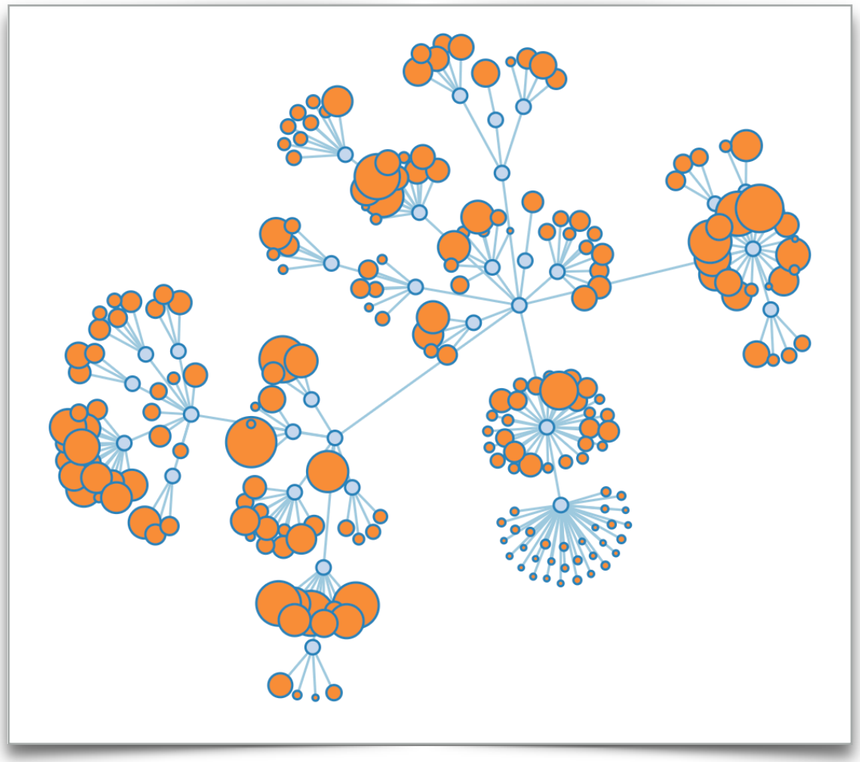


Chord Diagram

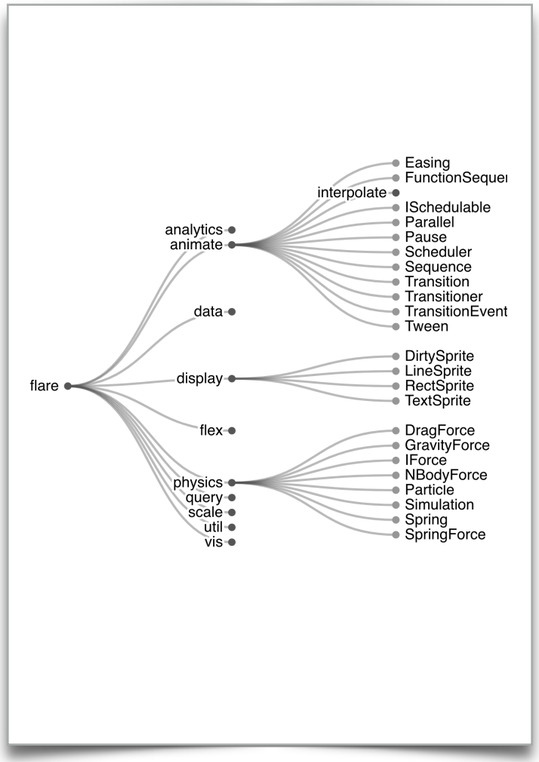


NETWORK & FLOW

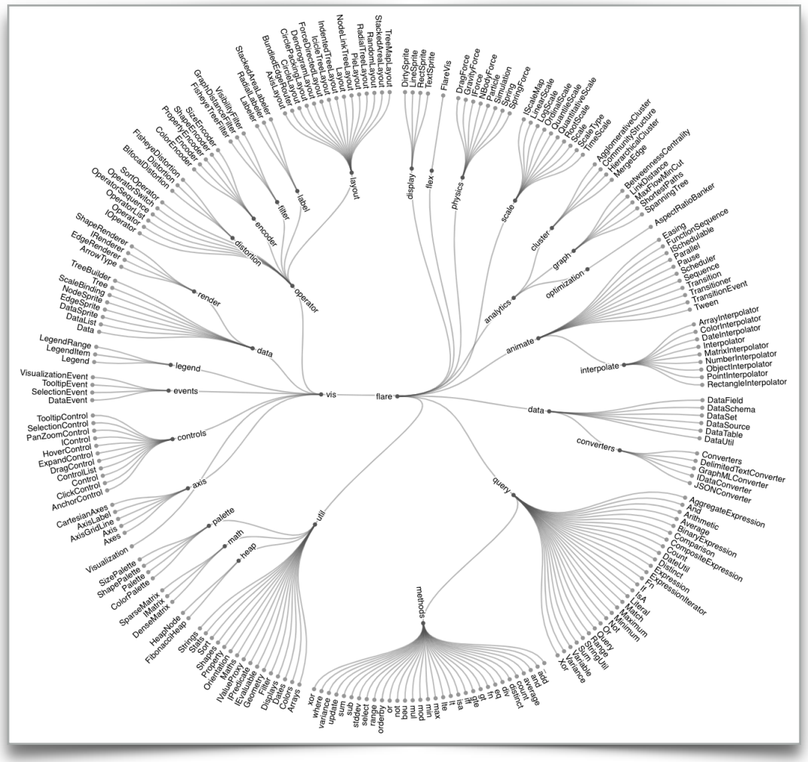
Force-Directed Graph



Tree

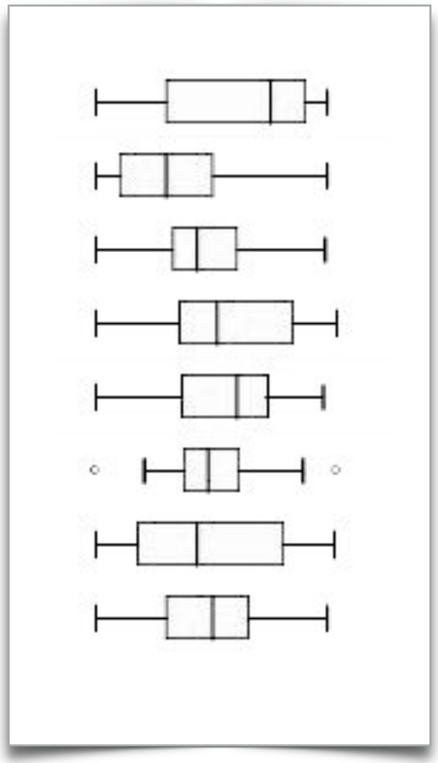


Radial Layout

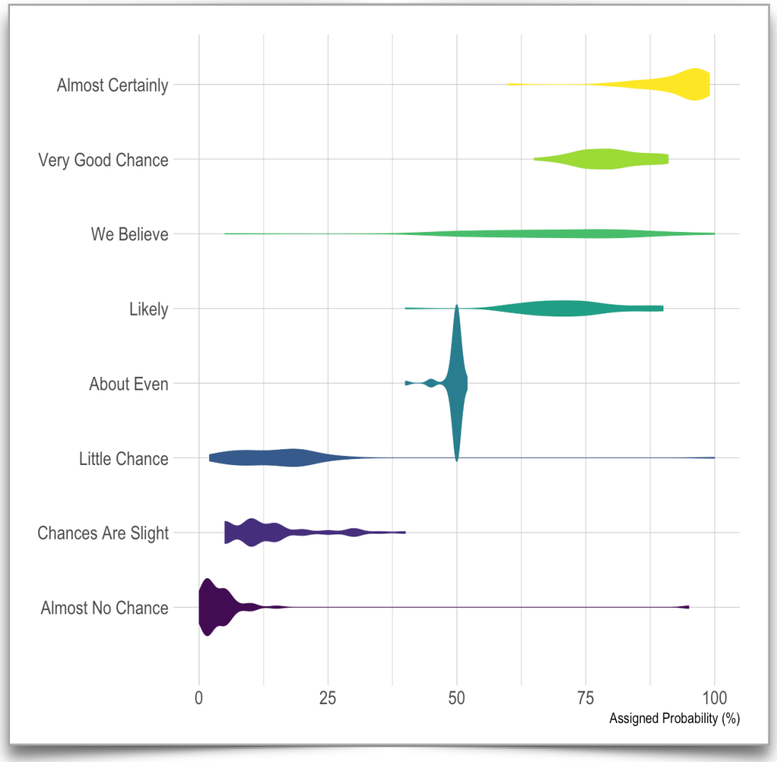


VARIANCE

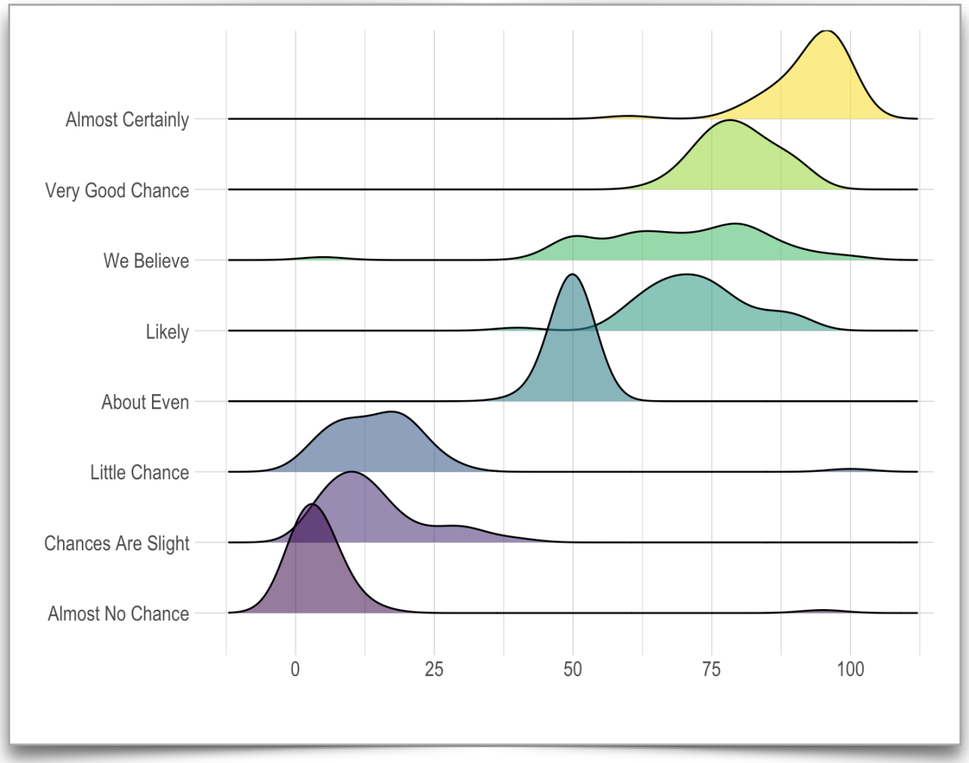
Boxplot



Violin Plot



Ridge Lines



VISUALIZATION WITH R

https://ggplot2.tidyverse.org/
https://github.com/rstudio/cheatsheets/blob/master/data-visualization-2.1.pdf

```
library(ggplot2)
```

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +                                  ] required
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
 stat = <STAT> , position = <POSITION> ) +                ] Not required, sensible defaults supplied
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

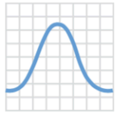
b + geom_abline(aes(intercept=0, slope=1)**)**
b + geom_hline(aes(yintercept = lat)**)**
b + geom_vline(aes(xintercept = long)**)**

b + geom_segment(aes(yend=lat+1, xend=long+1)**)**
b + geom_spoke(aes(angle = 1:1155, radius = 1)**)**

ONE VARIABLE continuous

c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)

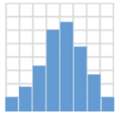
c + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size



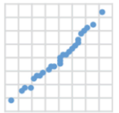
c + geom_density(kernel = "gaussian"**)**
x, y, alpha, color, fill, group, linetype, size, weight

c + geom_dotplot()
x, y, alpha, color, fill

c + geom_freqpoly() x, y, alpha, color, group, linetype, size



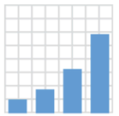
c + geom_histogram(binwidth = 5**)** x, y, alpha, color, fill, linetype, size, weight



c2 + geom_qq(aes(sample = hwy)**)** x, y, alpha, color, fill, linetype, size, weight

discrete

d <- ggplot(mpg, aes(fl))



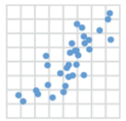
d + geom_bar()
x, alpha, color, fill, linetype, size, weight

TWO VARIABLES

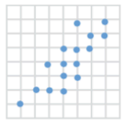
continuous x , continuous y

e <- ggplot(mpg, aes(cty, hwy))

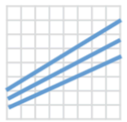
e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE**)** x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust



e + geom_jitter(height = 2, width = 2**)** x, y, alpha, color, fill, shape, size

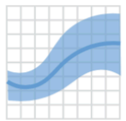


e + geom_point(), x, y, alpha, color, fill, shape, size, stroke

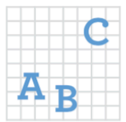


e + geom_quantile(), x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl"**)**, x, y, alpha, color, linetype, size



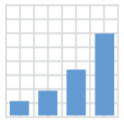
e + geom_smooth(method = lm**)**, x, y, alpha, color, fill, group, linetype, size, weight



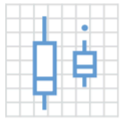
e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE**)**, x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

discrete x , continuous y

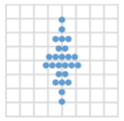
f <- ggplot(mpg, aes(class, hwy))



f + geom_col(), x, y, alpha, color, fill, group, linetype, size



f + geom_boxplot(), x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

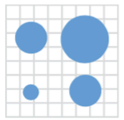


f + geom_dotplot(binaxis = "y", stackdir = "center"**)**, x, y, alpha, color, fill, group

f + geom_violin(scale = "area"**)**, x, y, alpha, color, fill, group, linetype, size, weight

discrete x , discrete y

g <- ggplot(diamonds, aes(cut, color))



g + geom_count(), x, y, alpha, color, fill, shape, size, stroke

COLOR AND FILL SCALES (DISCRETE)

n <- d + geom_bar(aes(fill = fl)**)**

n + scale_fill_brewer(palette = "Blues"**)**
For palette choices:
RColorBrewer::display.brewer.all()

n + scale_fill_grey(start = 0.2, end = 0.8,
na.value = "red"**)**

COLOR AND FILL SCALES (CONTINUOUS)

o <- c + geom_dotplot(aes(fill = ..x..))

o + scale_fill_distiller(palette = "Blues"**)**

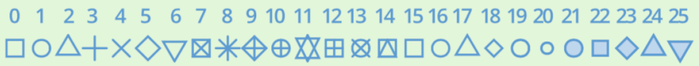
o + scale_fill_gradient(low="red", high="yellow"**)**

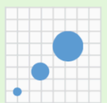
o + scale_fill_gradient2(low="red", high="blue",
mid = "white", midpoint = 25)

o + scale_fill_gradientn(colours=topo.colors(6)**)**
Also: rainbow(), heat.colors(), terrain.colors(),
cm.colors(), RColorBrewer::brewer.pal()

SHAPE AND SIZE SCALES

p <- e + geom_point(aes(shape = fl, size = cyl))
p + scale_shape() + scale_size()
p + scale_shape_manual(values = c(3:7)**)**





p + scale_radius(range = c(1,6)**)**
p + scale_size_area(max_size = 6**)**

continuous bivariate distribution

h <- ggplot(diamonds, aes(carat, price))

h + geom_bin2d(binwidth = c(0.25, 500)**)**
x, y, alpha, color, fill, linetype, size, weight

h + geom_density2d()
x, y, alpha, colour, group, linetype, size

h + geom_hex()
x, y, alpha, colour, fill, size

continuous function

i <- ggplot(economics, aes(date, unemploy))

i + geom_area()
x, y, alpha, color, fill, linetype, size

i + geom_line()
x, y, alpha, color, group, linetype, size

i + geom_step(direction = "hv")
x, y, alpha, color, group, linetype, size

Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables.

t <- ggplot(mpg, aes(cty, hwy)) + geom_point()

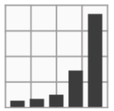
t + facet_grid(. ~ fl)
facet into columns based on fl

t + facet_grid(year ~ .)
facet into rows based on year

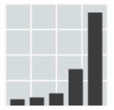
t + facet_grid(year ~ fl)
facet into both rows and columns

t + facet_wrap(~ fl)
wrap facets into a rectangular layout

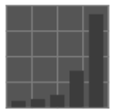
Themes



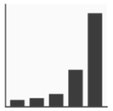
r + theme_bw()
White background with grid lines



r + theme_gray()
Grey background (default theme)

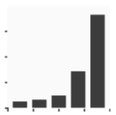


r + theme_dark()
dark for contrast



r + theme_classic()

r + theme_light()



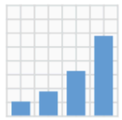
r + theme_linedraw()

r + theme_minimal()
Minimal themes

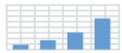
r + theme_void()
Empty theme

Coordinate Systems

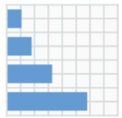
r <- d + geom_bar()



r + coord_cartesian(xlim = c(0, 5)**)**
xlim, ylim
The default cartesian coordinate system

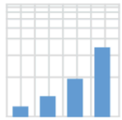


r + coord_fixed(ratio = 1/2**)**
ratio, xlim, ylim
Cartesian coordinates with fixed aspect ratio between x and y units



r + coord_flip()
xlim, ylim
Flipped Cartesian coordinates

r + coord_polar(theta = "x", direction=1 **)**
theta, start, direction
Polar coordinates



r + coord_trans(ytrans = “sqrt"**)**
xtrans, ytrans, limx, limy
Transformed cartesian coordinates. Set xtrans and ytrans to the name of a window function.

π + coord_quickmap()

π + coord_map(projection = "ortho", orientation=c(41, -74, 0)**)**projection, orienztation
xlim, ylim
Map projections from the mapproj package (mercator (default), azequalarea, lagrange, etc.)

SAVING GRAPHS

```
data %>%
   ggplot( aes( x = my_column, y = other_column ) ) +
     geom_point( aes( text = paste("Tip Box:", another_column) ) ) +
     geom_smooth( aes( colour = last_column, fill = last_column ) ) +
     facet_wrap( ~ last_column )

ggsave( "my_visualization.pdf", width=18, height=18)

ggsave( "my_image.png", width=10, height=10, unit="cm")
```

INTERACTIVE VISUALIZATION

PLOT.LY WITH R

```
library(plotly)
```

https://plot.ly/r/
https://moderndata.plot.ly/interactive-r-visualizations-with-d3-ggplot2-rstudio/
http://www.rebeccabarter.com/blog/2017-04-20-interactive/

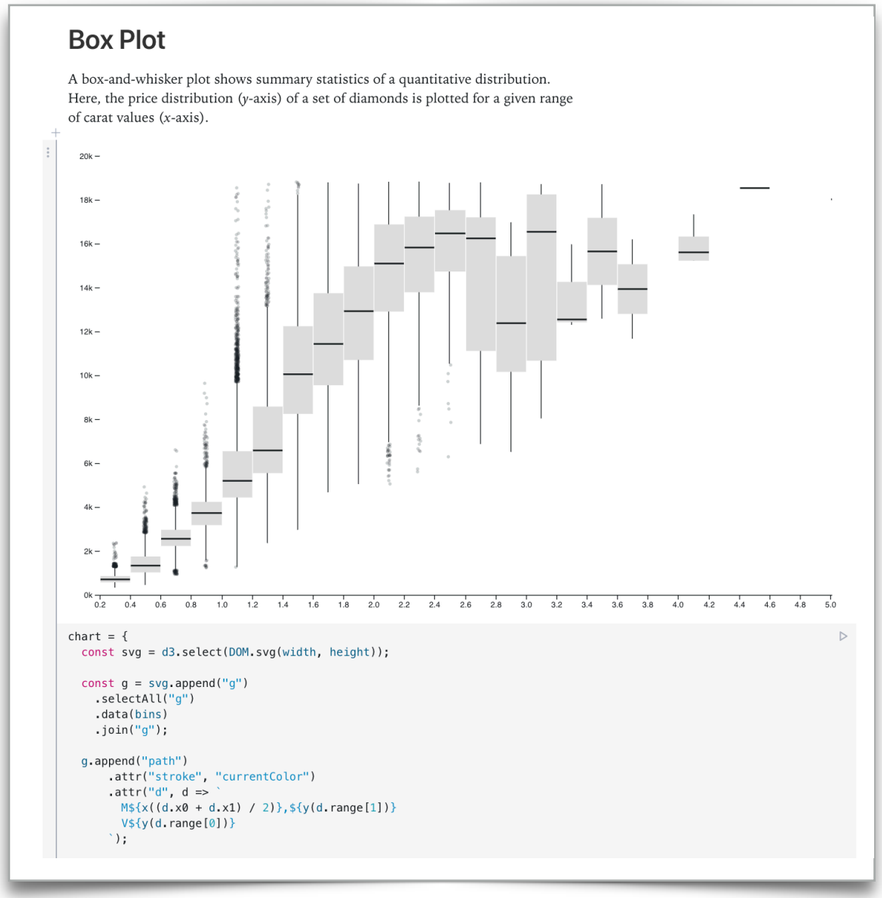
```
data %>%
   ggplot( aes( x = my_column, y = other_column ) ) +
     geom_point( aes( text = paste("Tip Box:", another_column) ) ) +
     geom_smooth( aes( colour = last_column, fill = last_column ) ) +
     facet_wrap( ~ last_column ) %>%

   ggplotly()
```

JAVASCRIPT WITH D3

D3 is a library for data visualization
to display in a web browser

D3 is **written in Javascript**
The W3C provides tutorials [1]

Many open source templates
can be reused [2, 3]



[1] https://www.w3schools.com/js/
[2] https://github.com/d3/d3/wiki/Gallery
[3] https://c3js.org/examples.html