Data Visualization In R

Agenda

- Types Of Visualization
- Graphs In R
- Type Of Graphs
- Line Plots
- Dot Plots
- Bar Plots
- Pie Charts
- Histograms
- Scatterplots
- 3-D Scatterplots
- Interactive Graphs

Types Of Visualization

A Data Visualisation Challenge…

You will see 3 questions.
You have 30 seconds.
Try it!

Your timer
starts now

How Many Numbers Are Above 100?

23	32	71	72	58	87	11	77	70	16
17	21	56	44	68	51	84	20	60	40
37	8	107	14	12	41	69	14	18	71
62	55	59	64	33	55	71	58	103	92
101	56	45	34	43	15	73	78	6	93
39	53	22	26	26	94	60	82	99	74
11	12	36	67	70	71	97	59	73	99
75	74	69	69	51	48	2	66	92	98
15	10	41	58	104	94	92	84	74	82
12	52	10	57	33	77	88	81	81	91
15	56	25	30	21	7	66	66	78	87
29	23	5	34	11	96	74	99	99	88
37	10	43	15	50	71	65	60	101	98
46	34	19	102	57	70	95	84	63	91
3	34	39	37	60	81	65	63	9	71
48	46	25	50	22	64	91	76	71	79

How Many Numbers Are Below 10?

23	32	71	72	58	87	11	77	70	16
17	21	56	44	68	51	84	20	60	40
37	8	107	14	12	41	69	14	18	71
62	55	59	64	33	55	71	58	103	92
101	56	45	34	43	15	73	78	6	93
39	53	22	26	26	94	60	82	99	74
11	12	36	67	70	71	97	59	73	99
75	74	69	69	51	48	2	66	92	98
15	10	41	58	104	94	92	84	74	82
12	52	10	57	33	77	88	81	81	91
15	56	25	30	21	7	66	66	78	87
29	23	5	34	11	96	74	99	99	88
37	10	43	15	50	71	65	60	101	98
46	34	19	102	57	70	95	84	63	91
3	34	39	37	60	81	65	63	9	71
48	46	25	50	22	64	91	76	71	79

Which Quadrant Has The Highest Total?

23	32	71	72	58	87	11	77	70	16
17	21	56	44	68	51	84	20	60	40
37	8	107	14	12	41	69	14	18	71
62	55	59	64	33	55	71	58	103	92
101	56	45	34	43	15	73	78	6	93
39	53	22	26	26	94	60	82	99	74
11	12	36	67	70	71	97	59	73	99
75	74	69	69	51	48	2	66	92	98
15	10	41	58	104	94	92	84	74	82
12	52	10	57	33	77	88	81	81	91
15	56	25	30	21	7	66	66	78	87
29	23	5	34	11	96	74	99	99	88
37	10	43	15	50	71	65	60	101	98
46	34	19	102	57	70	95	84	63	91
3	34	39	37	60	81	65	63	9	71
48	46	25	50	22	64	91	76	71	79

A Data Visualisation Challenge…

We’ll answer the same questions again.
But with simple visual cues.
See how long it takes.

Your timer
starts now

How Many Numbers Are Above 100?

1

23	32	71	72	58	87	11	77	70	16
17	21	56	44	68	51	84	20	60	40
37	8	107	14	12	41	69	14	18	71
62	55	59	64	33	55	71	58	103	92
101	56	45	34	43	15	73	78	6	93
39	53	22	26	26	94	60	82	99	74
11	12	36	67	70	71	97	59	73	99
75	74	69	69	51	48	2	66	92	98
15	10	41	58	104	94	92	84	74	82
12	52	10	57	33	77	88	81	81	91
15	56	25	30	21	7	66	66	78	87
29	23	5	34	11	96	74	99	99	88
37	10	43	15	50	71	65	60	101	98
46	34	19	102	57	70	95	84	63	91
3	34	39	37	60	81	65	63	9	71
48	46	25	50	22	64	91	76	71	79

Which Quadrant Has The Highest total?

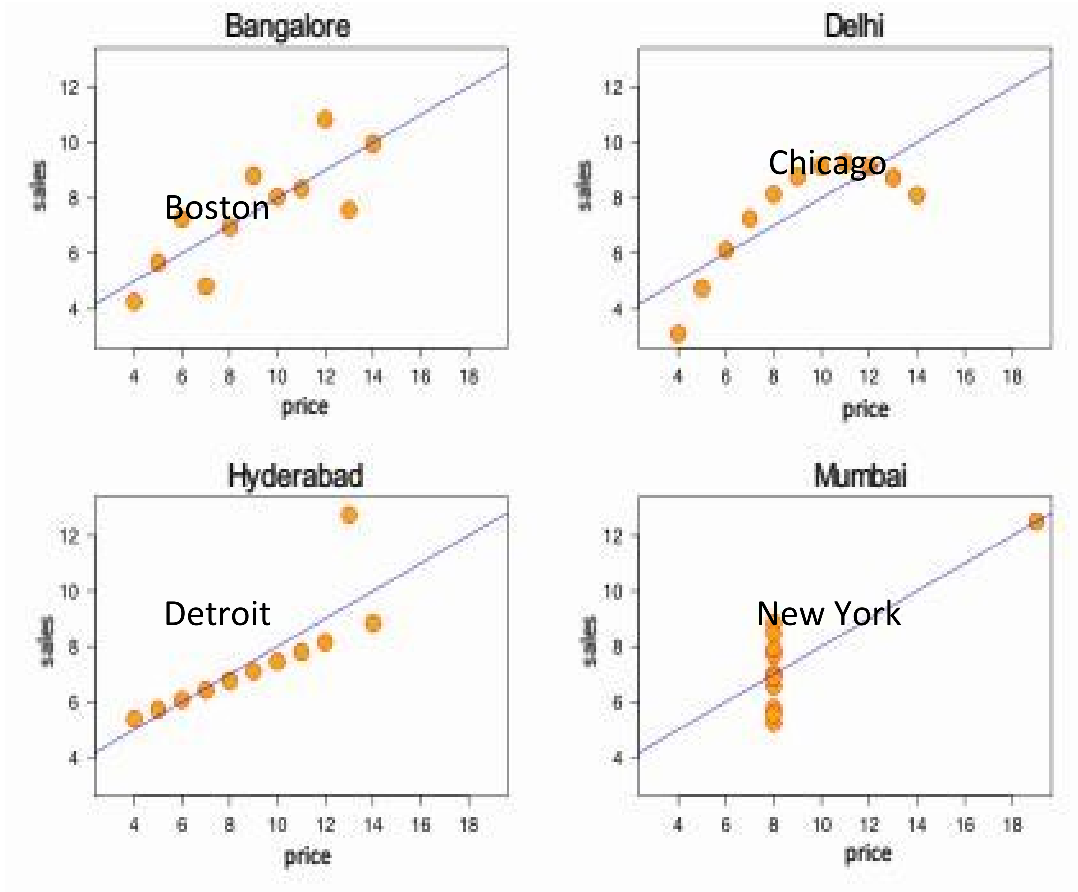
23	32	71	72	58	87	11	77	70	16
17	21	56	44	68	51	84	20	60	40
37	8	107	14	12	41	69	14	18	71
62	55	59	64	33	55	71	58	103	92
101	56	45	34	43	15	73	78	6	93
39	53	22	26	26	94	60	82	99	74
11	12	36	67	70	71	97	59	73	99
75	74	69	69	51	48	2	66	92	98
15	10	41	58	104	94	92	84	74	82
12	52	10	57	33	77	88	81	81	91
15	56	25	30	21	7	66	66	78	87
29	23	5	34	11	96	74	99	99	88
37	10	43	15	50	71	65	60	101	98
46	34	19	102	57	70	95	84	63	91
3	34	39	37	60	81	65	63	9	71
48	46	25	50	22	64	91	76	71	79

WHY VISUALISE?

You will be shown a set of numbers
along with a summary (average, etc.)
Can you make sense of the figures?

Are They Really Identical? Check Again…

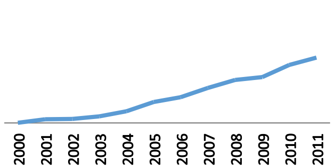
- But in fact, the four cities are totally different in behaviour.
- Boston's sales has generally increased with price.
- Detroit has a nearly perfect increase in sales with price, except for one aberration.
- Chicago shows a decline in sales beyond a price of 10.
- New York's sales fluctuates despite a nearly constant price.



Market

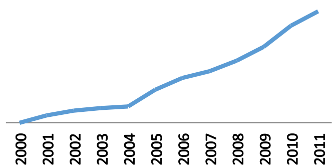
Transaction data

Increasing data being churned out by systems in information highway



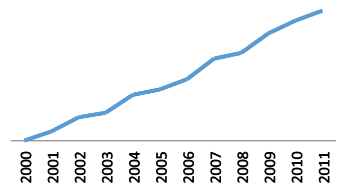
Social network data

Consumers embracing Web 2.0 and the social media lifestyle



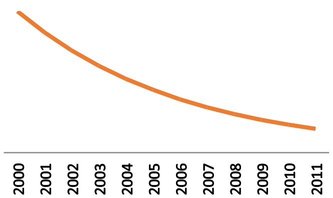
M2M data

Portable devices generating data for consumption by systems



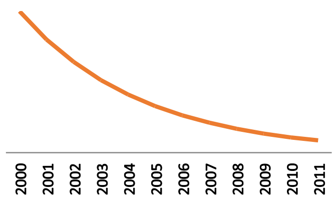
Storage cost

Material science research has led to significant increase in data density



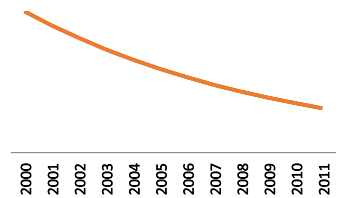
Bandwidth cost

Driven by massive investments in fibre capacity



Processing cost

Moore's law has doubled the processing power per $ every 1.5 yrs



Gartner's BI Magic Quadrant Trends

- Emergence of data discovery/ visualization
- Increased willingness for new low-cost options
- Embedded low-cost purpose-built analytic apps
- Need for intuitive BI tools on mobile platforms

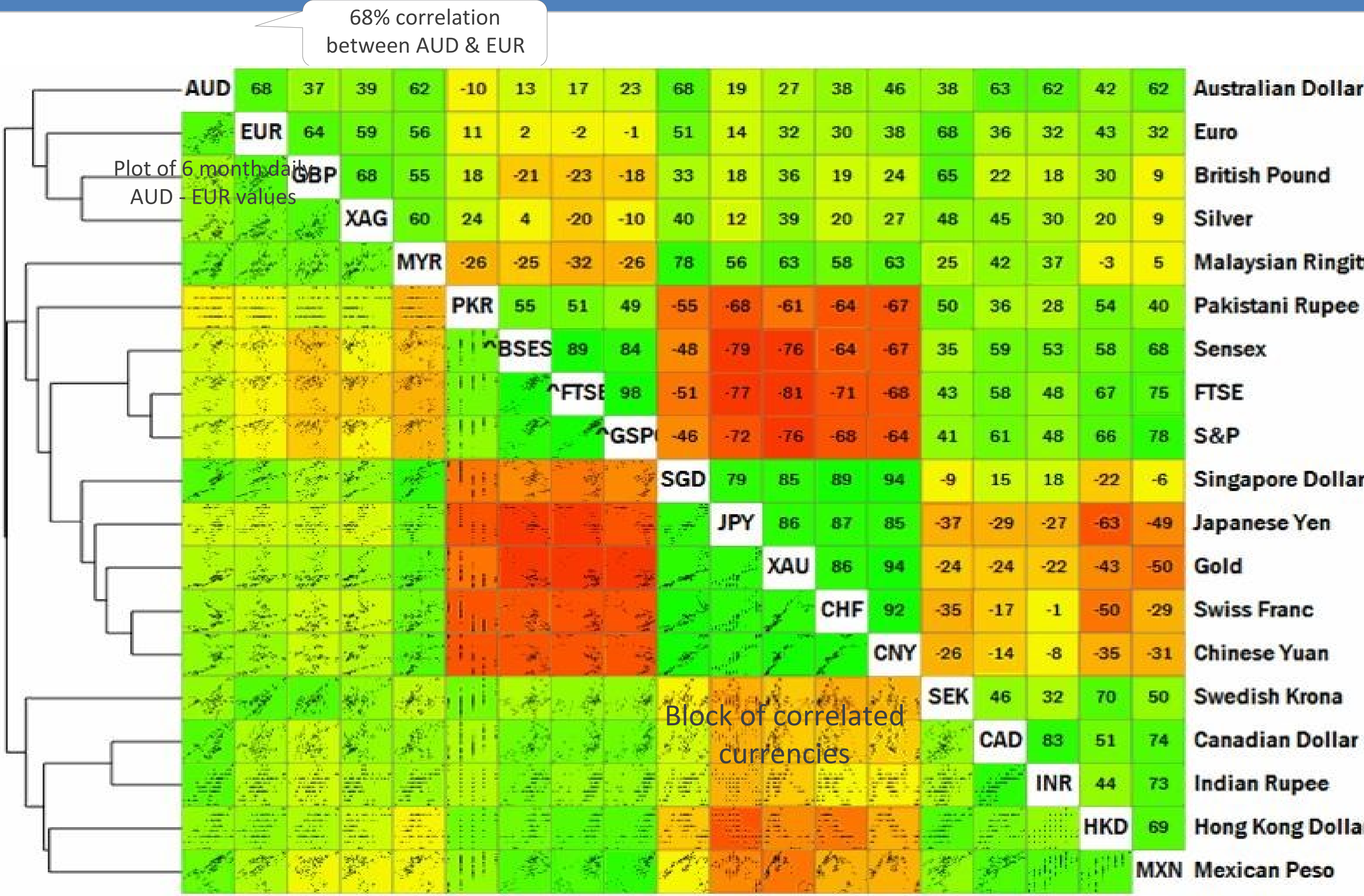
Growth in available data, and the potential for exploiting these, have grown exponentially in the last 10 years.

This changing data landscape heralds a radical shift in business decision-making approach, even for mere survival in this new age.

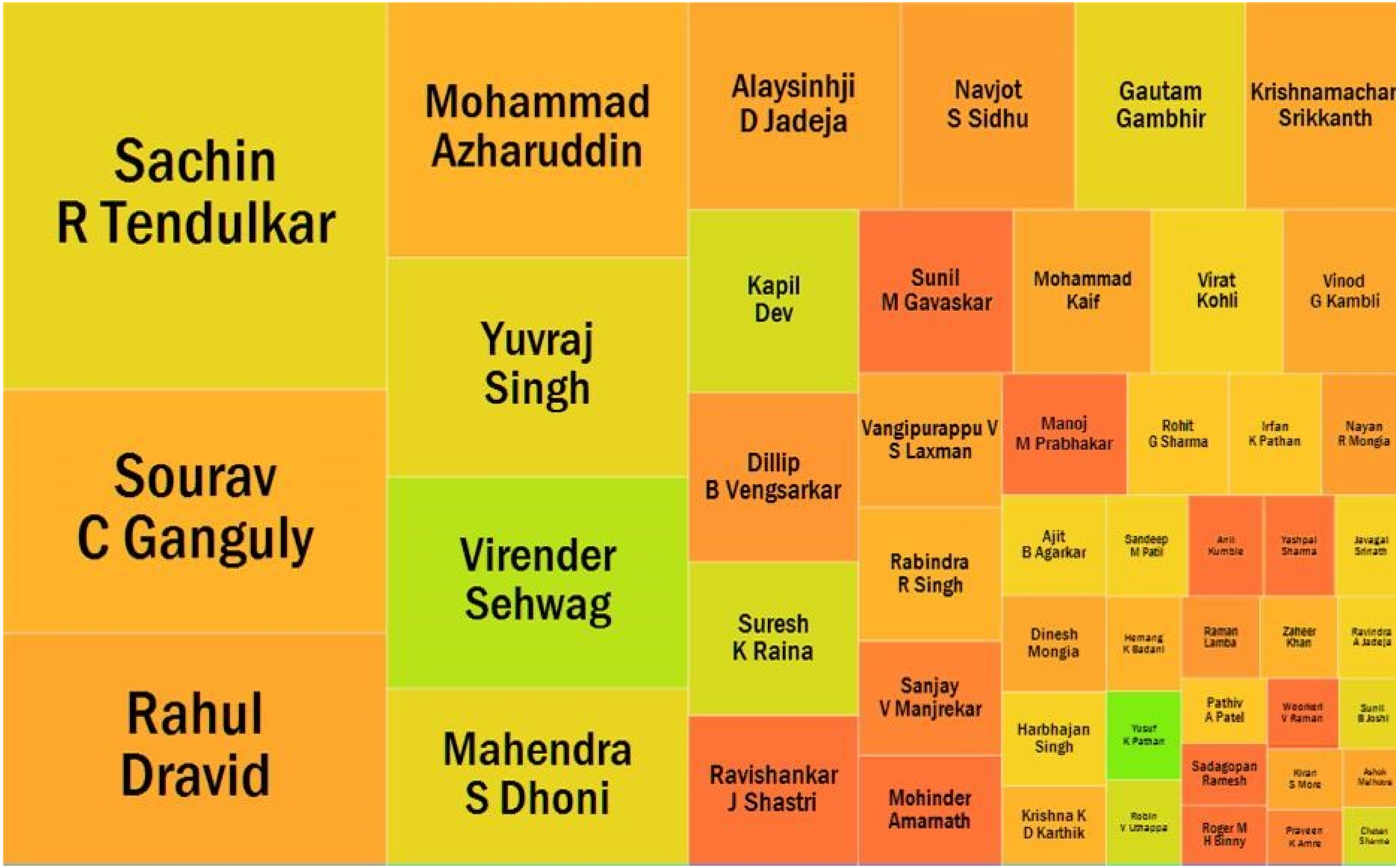
Data growth to 7.9 ZB by 2015 posing a real 'Data Tsunami'

Information is the oil of the 21st century, and analytics is the combustion engine

Correlation Between AUD & EUR



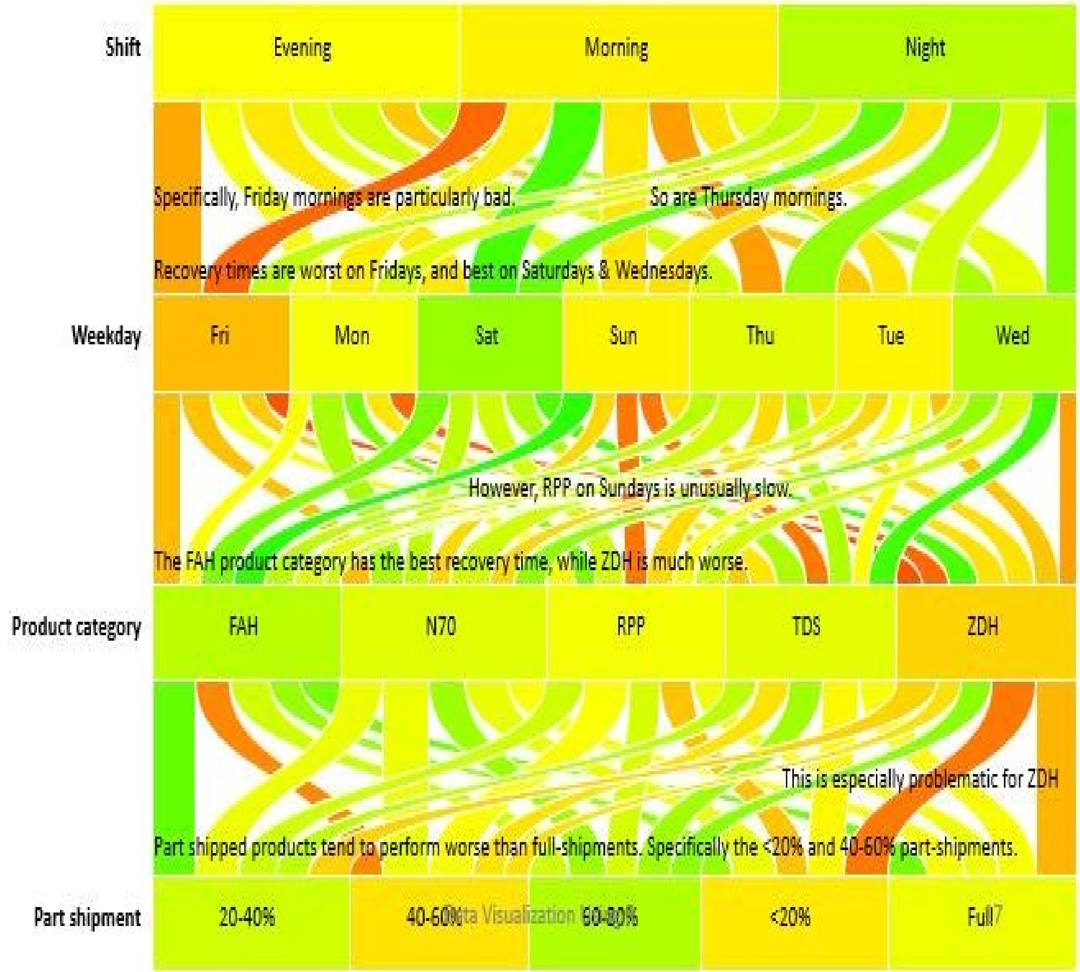
India ODI Batting



Cargo Delay

- This visualisation measures the recovery time (time from arrival of the flight until delivery), and identifies which factors most influence the recovery time.
- This visualisation is part of a suite of analytical techniques we call “grouped means” that allows us to measure the impact of every parameter (shifts, weekdays, etc.) on any measure of interest – recovery time in this case, but this could be extended to revenue, operational efficiency, or ability to cross-sell.
- It allows automatically detection of statistically significant flows and highlights only relevant ones to users.
- The system therefore analyses all possible patterns, but users only see the insights that matter.

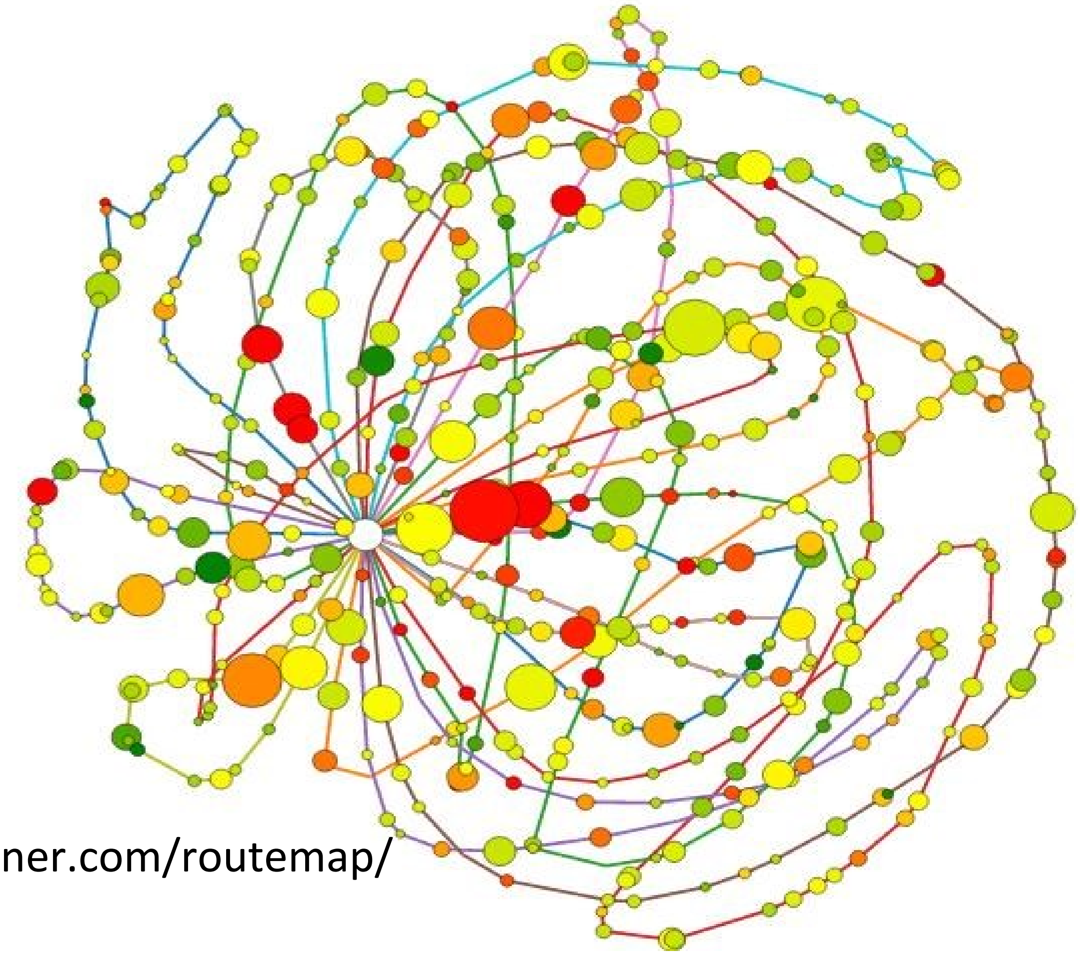
Recovery times are neutral during the evening and morning shifts (mornings are slightly worse), night times are the best.



Transport Route Optimization

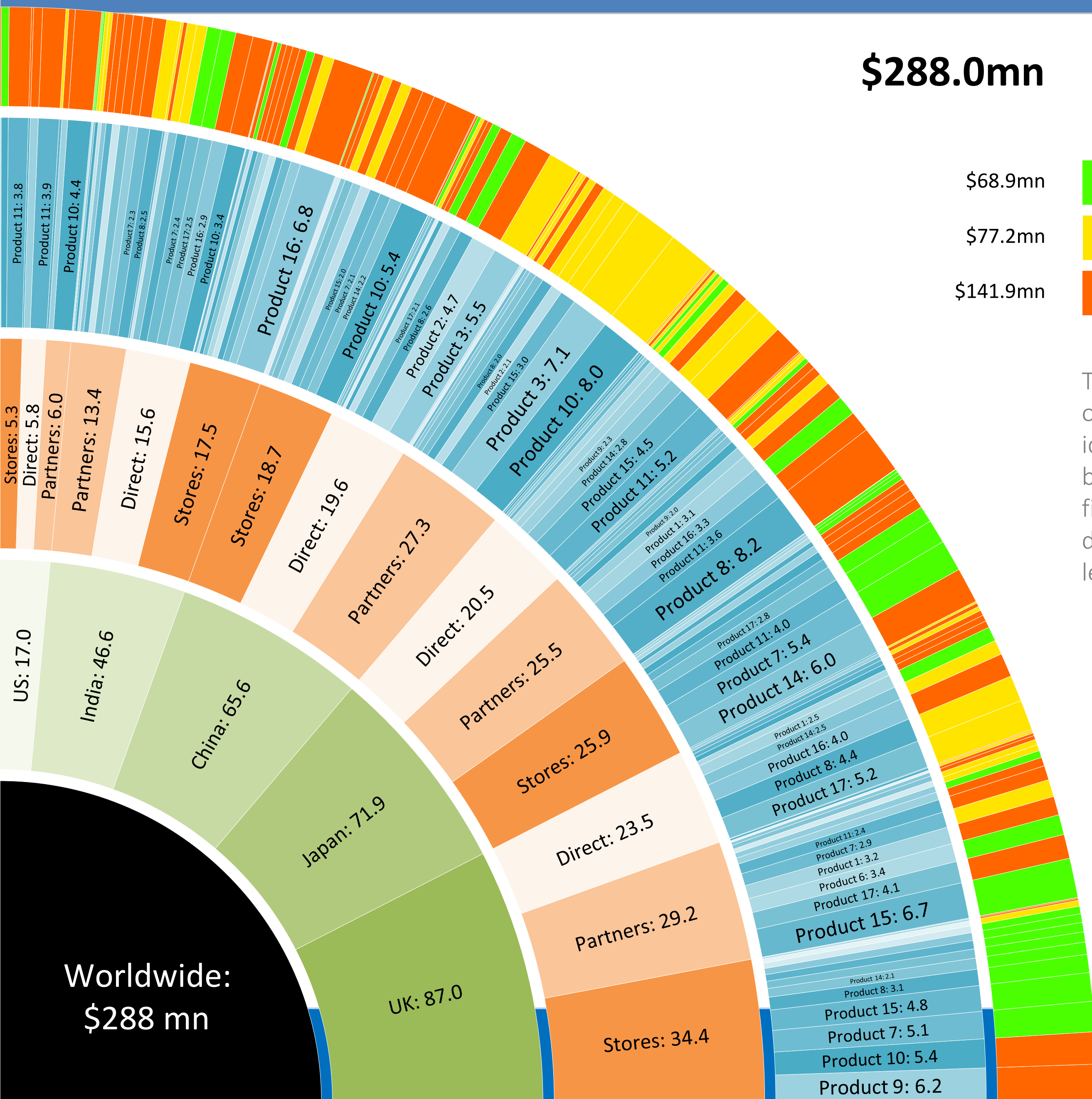
Below is the (anonymised) supplier network for a plant. Each circle represents a supplier. The size of the circle represents the volume of supplies purchased from that supplier. The larger the circle, the larger the purchases. The colour of the circle represents the price at which it was purchased. Red indicates a high price, yellow is neutral, and green is low.

Each sequence of lines represents a supply route along which trucks are sent to puck up supplies. The length of the lines is approximately the distance between two suppliers.



https://gramener.com/routemap/

Portfolio Performance Visual



$288.0mn **Worldwide**

$68.9mn	A: Accelerate
$77.2mn	B: Build
$141.9mn	C: Cut down

The visualization shows the market opportunities across various countries to identify areas of focus. This chart has been built as an *interactive-app* to present the key findings, while letting user click-through and drill-down to a custom view across 4 different levels.

Graphs In R

Graphs In R

Building graphs in R is easy

```
x <- c (1, 2, 3, 4, 5)
y <- c (1, 5, 3, 2, 0)
plot (x, y)
```

plot () function considers the vector x and y, interprets them on a x – y plane and draws the points.

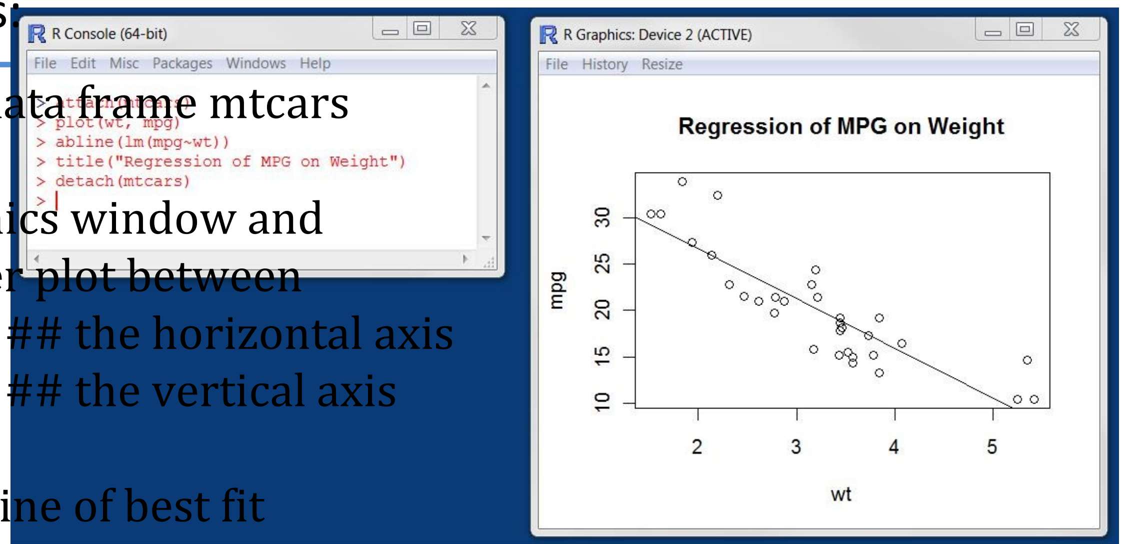
Exercise: Explore the relationship between the following, where x contains numbers from 1 to 100:

1. x and x^2
2. x and x^3
3. x + y = 101
4. xy = 500

Working With Graphs

#Build a graph one statement at a time by adding features,

Consider the following five lines:




```
attach(mtcars) ## attaches the data frame mtcars

plot(wt, mpg)   ## opens a graphics window and
generates a                ## scatter plot between
automobile weight on              ## the horizontal axis
and miles per gallon on            ## the vertical axis

abline(lm(mpg~wt)) ## adds a line of best fit

title("Regression of MPG on Weight") ## adds a title

detach(mtcars) ## detaches the data frame
```

Note: In R, graphs are typically created in this interactive fashion

Working With Graphs (Contd.)

To save a graph via code, sandwich the statements that produce the graph between a statement that sets a destination and a statement that closes that destination.

```
pdf("mygraph.pdf") #saves graph in current working directory:
attach(mtcars)
plot(wt, mpg)
abline(lm(mpg~wt))
title("Regression of MPG on Weight")
detach(mtcars)
dev.off() #shuts down the specified graphics device
```

In addition to pdf(), you can use the functions png(), jpeg(), bmp(), tiff(), xfig(), and postscript() to save graphs in other formats.

Working With Graphs (Contd.)

R can create attractive graphs with a minimum of input from user. But we can also use graphical parameters to specify fonts, colors, line styles, axes, reference lines, and annotations.

A simple example

Let's start with the simple fictitious dataset given in table shown below.
It describes patient response to two drugs at five dosage levels.

Dosage	Response to Drug A	Response to Drug B
20	16	15
30	20	18
40	27	25
45	40	31
60	60	40

You can input this data using this code:

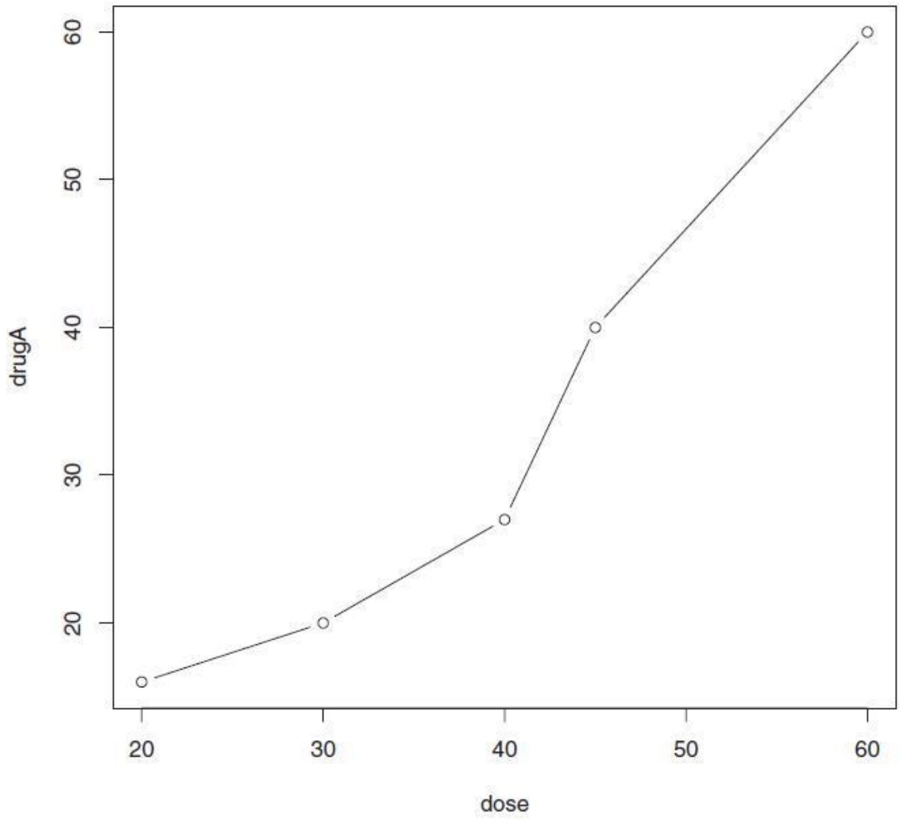
```
dose <- c(20, 30, 40, 45, 60)
drugA <- c(16, 20, 27, 40, 60)
drugB <- c(15, 18, 25, 31, 40)
```

Working With Graphs (Contd.)

Create a line graph relating dose to response for drug A :

```
plot(dose, drugA, type="b")
```

- plot() function that plots objects in R
- The option type="b" indicates that both points and lines should be plotted.
- Use help(plot) to view other options.



Graphical Parameters

- Customize graph's fonts, colors, axes, titles through options called graphical parameters.
- To specify these options through the par() function. Values will remain in effect for the rest of the session or until they're changed.
- The format is par(optionname=value, optionname=value, ...).

#In existing example , use a **star** rather than an open circle as plotting symbol, and connect points using a **dashed line** rather than a solid line.

```
opar <- par(no.readonly=TRUE) #list of current graphical settings that can be modified.
par(lty=2, pch=8) # lty for dashed line type, pch value 17 for solid triangle as symbol
plot(dose, drugA, type="b")
```

Symbols & Lines

Parameter	Description
pch	plot character specifies the symbol to use when plotting points
cex	Number indicating the amount by which plotted text and symbols should be scaled relative to the default =1
lty	specifies the line type
lwd	specifies the line width.

```
plot(dose, drugA, type="b", lty=3, lwd=3, pch=15, cex=2)
```

Colors

There are several color-related parameters in R. Table below shows some of the common ones.

Parameters	Description
col	Default plotting color. For e.g, if col=c(“red”, “blue”)and three lines are plotted, the first line will be red, the second blue, and the third red.
col.axis	Color for axis text
col.lab	Color for axis labels
col.main	Color for titles
col.sub	Color for subtitles
fg	The plot’s foreground color
bg	The plot’s background color

- Colors can be specified in R by index, name, hexadecimal, RGB, or HSV. For example, col=1, col="white", col="#FFFFFF", col=rgb(1,1,1), and col=hsv(0,0,1) are equivalent ways of specifying the color white.

Text Characteristics

Parameters controlling text size are explained in table shown below.

Parameter	Description
cex	Number indicating the amount by which plotted text and symbols should be scaled relative to the default. 1=default, 1.5 is 50% larger, 0.5 is 50% smaller, etc.
cex.axis	Magnification of axis text relative to cex.
cex.lab	Magnification of axis labels relative to cex.
cex.main	Magnification of titles relative to cex.
cex.sub	Magnification of subtitles relative to cex.

For example, all graphs created after the statement :

```
par(font.lab=3, cex.lab=1.5, font.main=4, cex.main=2)
```

will have italic axis labels that are 1.5 times the default text size, and bold italic titles that are twice the default text size.

Text Characteristics

Parameter	Description
font	Integer specifying font to use for plotted text.. 1=plain, 2=bold, 3=italic, 4=bold italic, 5=symbol
font.axis	Font for axis text.
font.lab	Font for axis labels
font.main	Font for titles.
font.sub	Font for subtitles.
ps	Font point size (roughly 1/72 inch).
family	Font family for drawing text. Standard values are serif, sans, and mono

Graph & Margin Dimensions

Parameter	Description
pin	Plot dimensions (width, height) in inches.
mai	A numerical vector of the form c(bottom, left, top, right) which gives the margin size specified in inches

The code produces graphs that are 4 inches wide by 3 inches tall, with a 1-inch margin on the bottom and top, a 0.5-inch margin on the left, and a 0.2-inch margin on the right.

```
par(pin=c(4,3), mai=c(1,.5, 1, .2))
```

Using Parameters To Control Graph Appearance

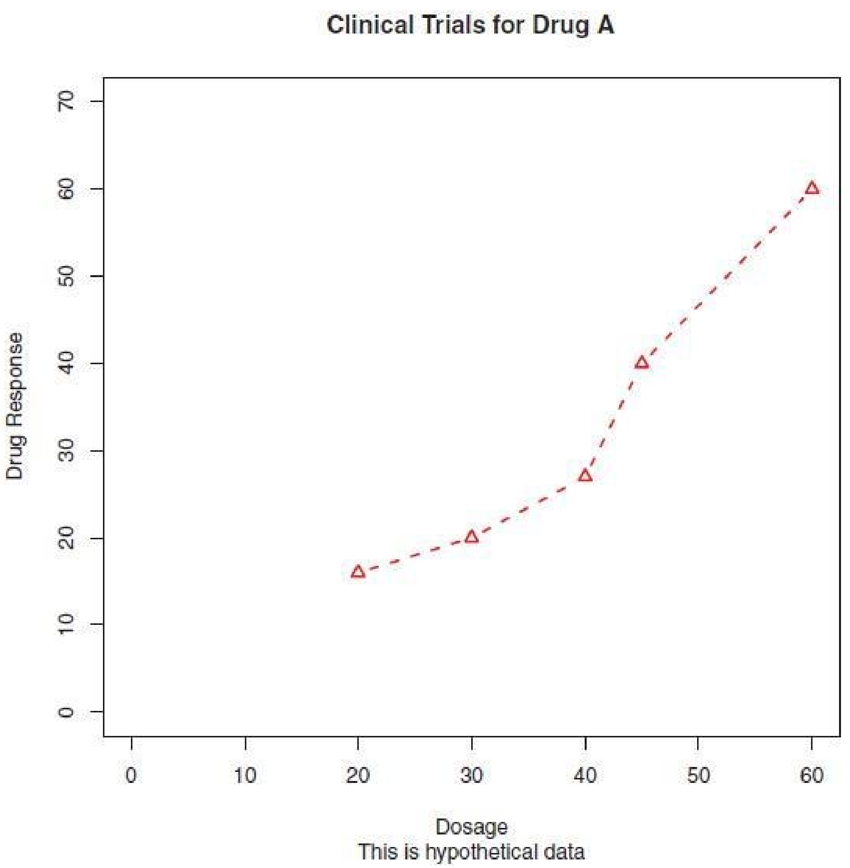
```
dose <- c(20, 30, 40, 45, 60)
drugA <- c(16, 20, 27, 40, 60)
drugB <- c(15, 18, 25, 31, 40)
# save the current graphical parameter setting to restore later
opar <- par(no.readonly=TRUE)
# graphs will be 2 inches wide by 3 inches tall
par(pin=c(2, 3))
#lines will be twice the default width and symbols will be 1.5 times the default size
par(lwd=2, cex=1.5)
# Axis text is italic and scaled to 75 percent of the default
par(cex.axis=.75, font.axis=3)
# first plot has filled red circles and dashed lines
plot(dose, drugA, type="b", pch=19, lty=2, col="red")
# Next plot has green filled diamonds and a blue border and blue dashed lines
plot(dose, drugB, type="b", pch=23, lty=6, col="blue", bg="green")
# Restore the original graphical parameter settings
par(opar)
```

.

Adding Text, Customized Axes & Legends

The following adds a title (main), subtitle (sub), axis labels (xlab, ylab), and axis ranges (xlim, ylim).

```
plot(dose, drugA, type="b",
col="red", lty=2, pch=2, lwd=2,
main="Clinical Trials for Drug A",
sub="This is hypothetical data",
xlab="Dosage", ylab="Drug
Response",
xlim=c(0, 60), ylim=c(0, 70))
```



Titles

Use the title() function to add title and axis labels to a plot.

The format is :

```
title(main="main title", sub="sub-title",
xlab="x-axis label", ylab="y-axis label")
```

Graphical parameters (such as text size, font, rotation, and color) can also be specified in the title() function.

For example, the following produces a red title and a blue subtitle, and creates green x and y labels that are 25 percent smaller than the default text size:

```
title(main="My Title", col.main="red",
sub="My Sub-title", col.sub="blue",
xlab="My X label", ylab="My Y label",
col.lab="green", cex.lab=0.75)
```

Type Of Graphs

Types Of Graphs

Following are the basic types of graphs, which can be chosen based on the situation and the data available.

- Line Plot
- Dot Plot
- Bar Plot
- Pie Chart
- Box Plot
- Scatter Plot

R also has special charts which are used for specific purposes.

Some special graphs:

- 3 D charts
- Interactive graphs

Line Plots

Line Plots

Line Charts

Line charts are created with the function lines(x, y, type=) where x and y are numeric vectors of (x,y) points to connect. type= can take the following values:

- The lines( ) function adds information to a graph.
- It can not produce a graph on its own.
- Usually it follows a plot(x, y) command that produces a graph.
- By default, plot( ) plots the (x,y) points.
- Use the type="n" option in the plot( ) command, to create the graph with axes, titles, etc., but without plotting the points.
- In the following code each of the type= options is applied to the same dataset.
- The plot( ) command sets up the graph, but does not plot the points.

Type	Description
p	points
l	lines
o	overplotted points and lines
b, c	points (empty if “c”) joined by lines
s, S	stair steps
h	histogram-like vertical lines
n	does not produce any points or lines

Line Plots

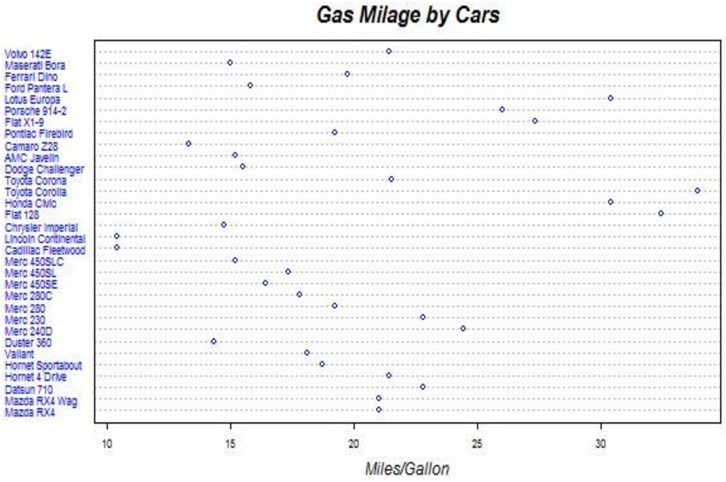
```
x <- c(1:5); y <- x # create some data
par(pch=22, col="red") # plotting symbol and color
par(mfrow=c(2,4)) # all plots on one page
opts = c("p","l","o","b","c","s","S","h")
for(i in 1:length(opts)){
  heading = paste("type=",opts[i])
  plot(x, y, type="n", main=heading)
  lines(x, y, type=opts[i])
}
```

Dot Plots

Dot Plots

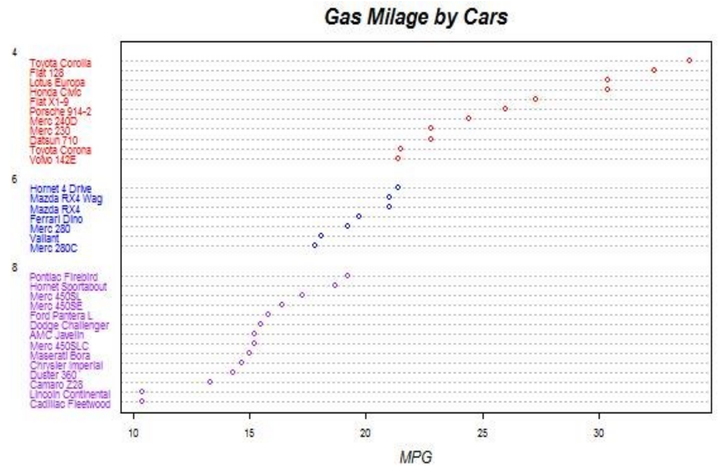
Basic Dot Plot

```
dotchart(mtcars$mpg,labels=row.names(mtcars),
        cex=.5, color = "blue",
        main="Gas Milage by Cars", xlab="Miles/Gallon")
```



Its also easy to use Dot plots to grouping in a Dot plot

```
myCars <- mtcars[order(mtcars$mpg),] # ordering the cars
myCars$cyl <- factor(myCars$cyl) # making cyl into factor
myCars$color[myCars$cyl==4] <- "red" # assigning colors
myCars$color[myCars$cyl==6] <- "blue" # assigning colors
myCars$color[myCars$cyl==8] <- "purple"  # assigning colors
dotchart(myCars$mpg,labels=row.names(myCars),
cex=.5, groups= myCars$cyl,
         main="Gas Milage by Cars",
         xlab="MPG", color=myCars$color)
```



Bar Plots

Bar Plots

Types:

- Simple Bar Plot
- Simple Horizontal Bar Plot with Added Labels
- Stacked Bar Plot with Colors and Legend

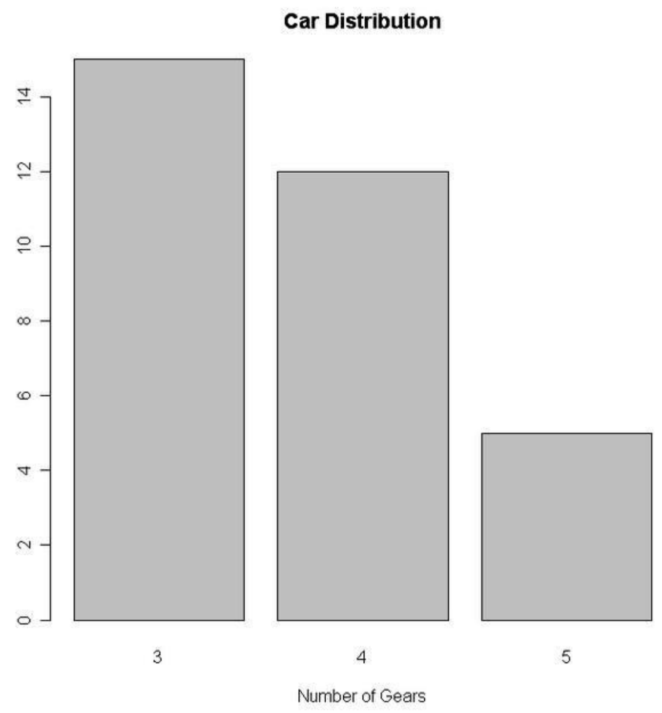
Simple Bar Plot

Simple Bar Plot

```
counts <- table(mtcars$gear)
barplot(counts, main="Car Distribution",
          xlab="Number of Gears")
```

Simple Horizontal Bar Plot with Added Labels

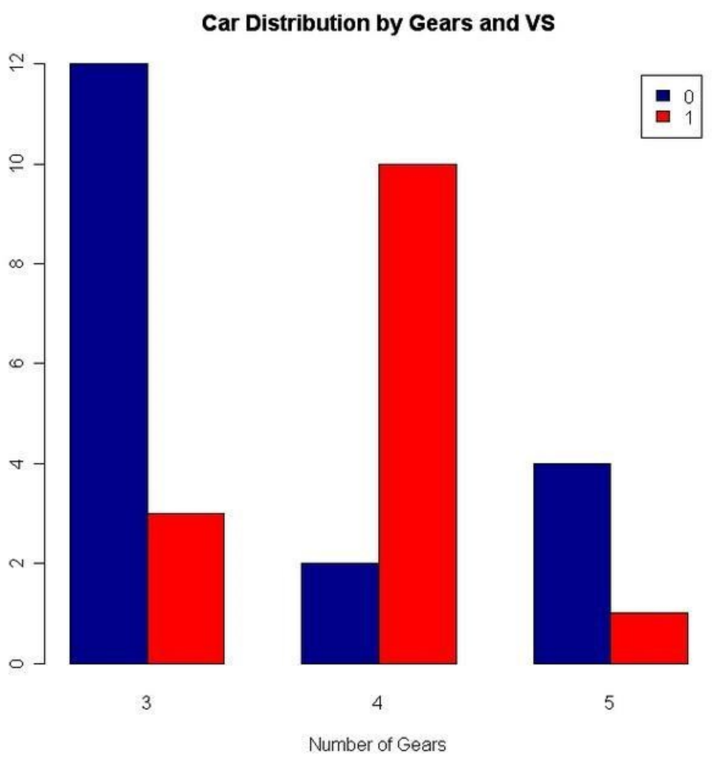
```
counts <- table(mtcars$gear)
barplot(counts, main="Car Distribution", horiz=TRUE,
 names.arg=c("3 Gears", "4 Gears", "5 Gears"))
```



Stacked Bar Plot

```
counts <- table(mtcars$vs, mtcars$gear)
barplot(counts, main="Car Distribution by
Gears and V/S",
  xlab="Number of Gears",
col=c("darkblue","red"),
              legend = rownames(counts))
```

```
counts <- table(mtcars$vs, mtcars$gear)
barplot(counts, main="Car Distribution by Gears
and V/S",
  xlab="Number of Gears",
col=c("darkblue","red"),
              legend = rownames(counts),
beside=TRUE)
```



Pie Charts

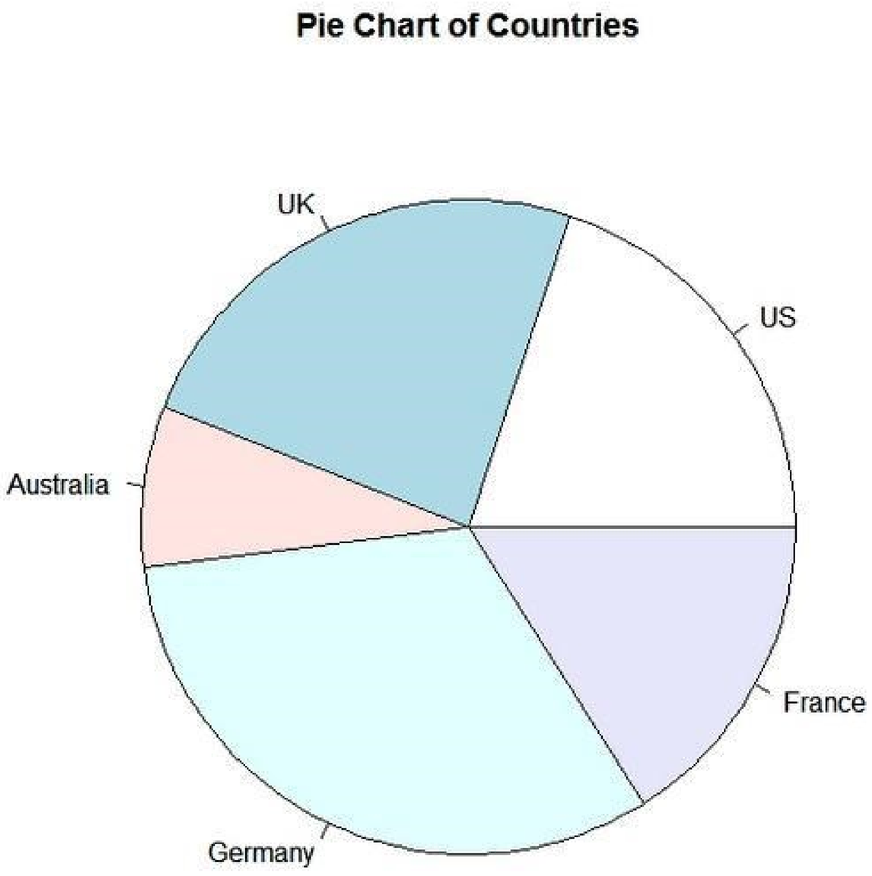
Pie Charts

Types of Pie Charts

- Simple Pie Charts
- Pie Chart with Annotated Percentages

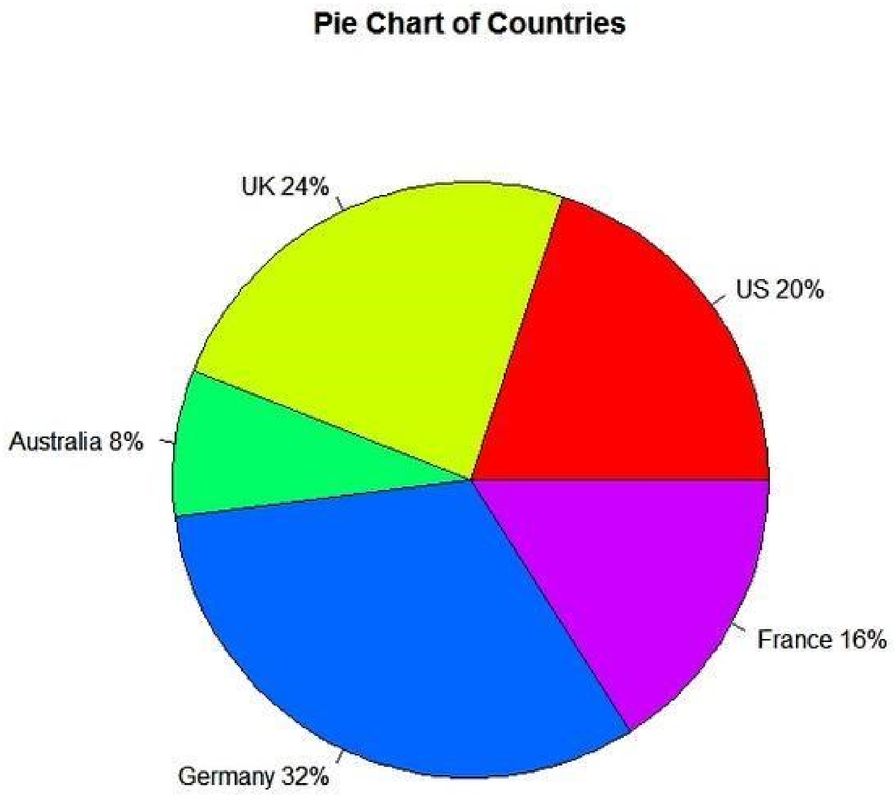
Simple Pie Charts

```
slices <- c(10, 12,4, 16, 8)
lbls <- c("US", "UK", "Australia", "Germany",
"France")
pie(slices, labels = lbls, main="Pie Chart of
Countries")
```



Pie Chart With Annotated Percentages

```
slices <- c(10, 12, 4, 16, 8)
lbls <- c("US", "UK", "Australia", "Germany", "France")
pct <- round(slices/sum(slices)*100)
lbls <- paste(lbls, pct) # add percents to labels
lbls <- paste(lbls,"%",sep="") # add % to labels
pie(slices,labels = lbls, col=rainbow(length(lbls)),
        main="Pie Chart of Countries")
```

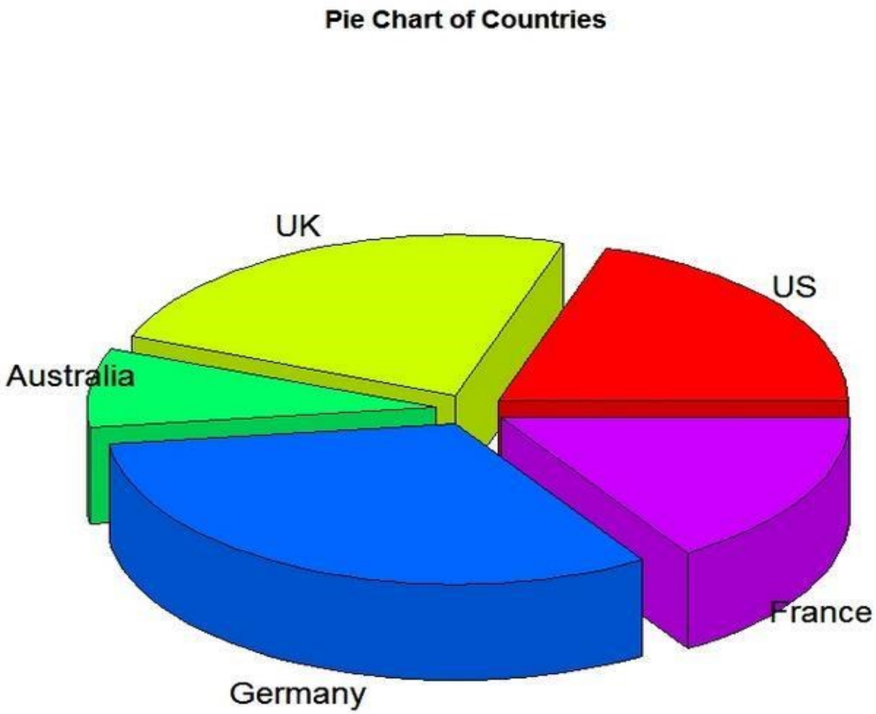


3D Pie Chart

- The pie3D( ) function in the plotrix package provides 3D exploded pie charts.

3D Exploded Pie Chart

```
library(plotrix)
slices <- c(10, 12, 4, 16, 8)
lbls <- c("US", "UK", "Australia", "Germany",
"France")
pie3D(slices,labels=lbls,explode=0.1,
  main="Pie Chart of Countries ")
```

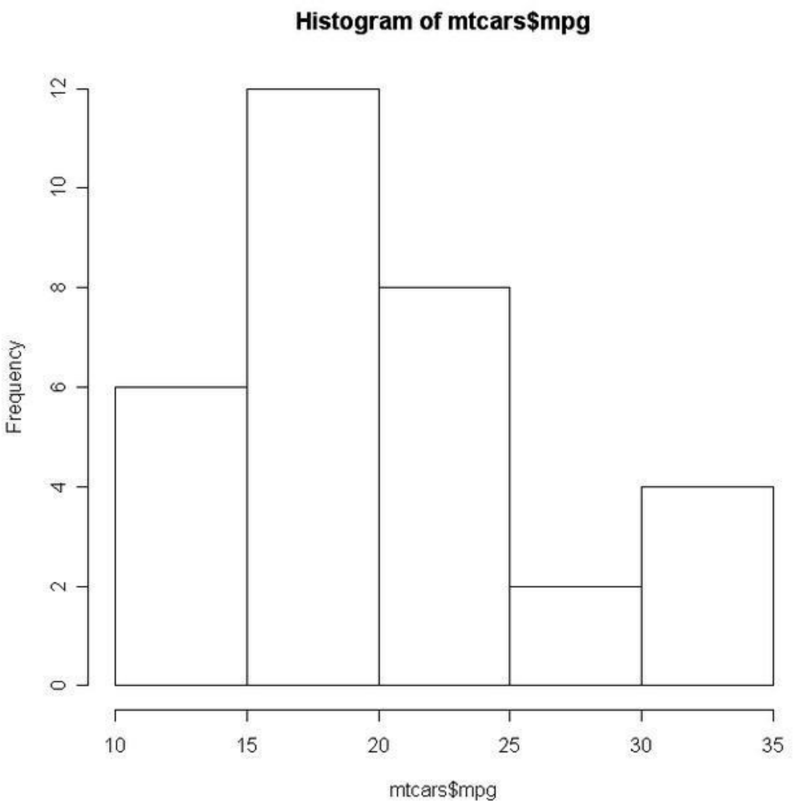


Histograms

Histograms

Histograms

Create histograms with the function hist(x) where x is a numeric vector of values to be plotted.
The option freq=FALSE plots probability densities instead of frequencies.
The option breaks= controls the number of bins.



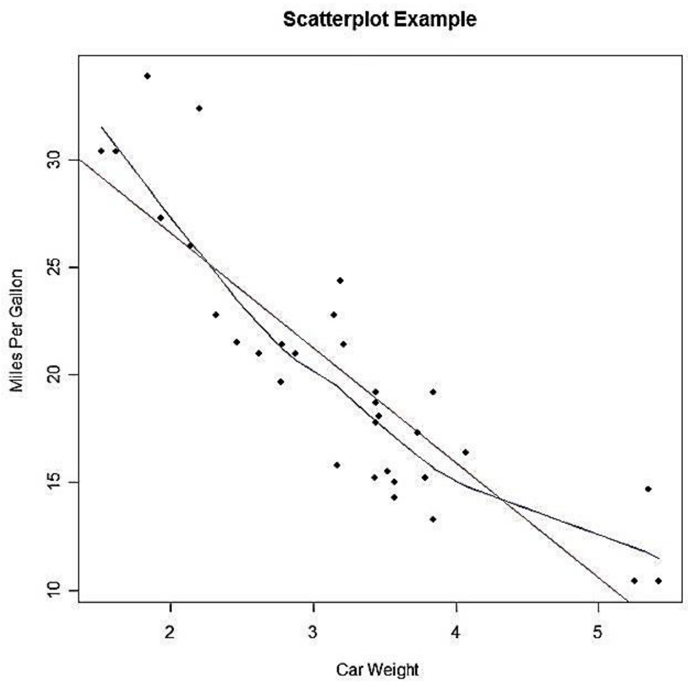
Simple Histogram
hist(mtcars$mpg)

Scatterplots

Scatterplots

Simple Scatterplot

```
attach(mtcars)
plot(wt, mpg, main="Scatterplot Example",
          xlab="Car Weight ", ylab="Miles Per
Gallon ", pch=19)
```



Add fit lines

```
abline(lm(mpg~wt), col="red") # regression line
lines(lowess(wt,mpg), col="blue") #local
polynomial regression fitting
```

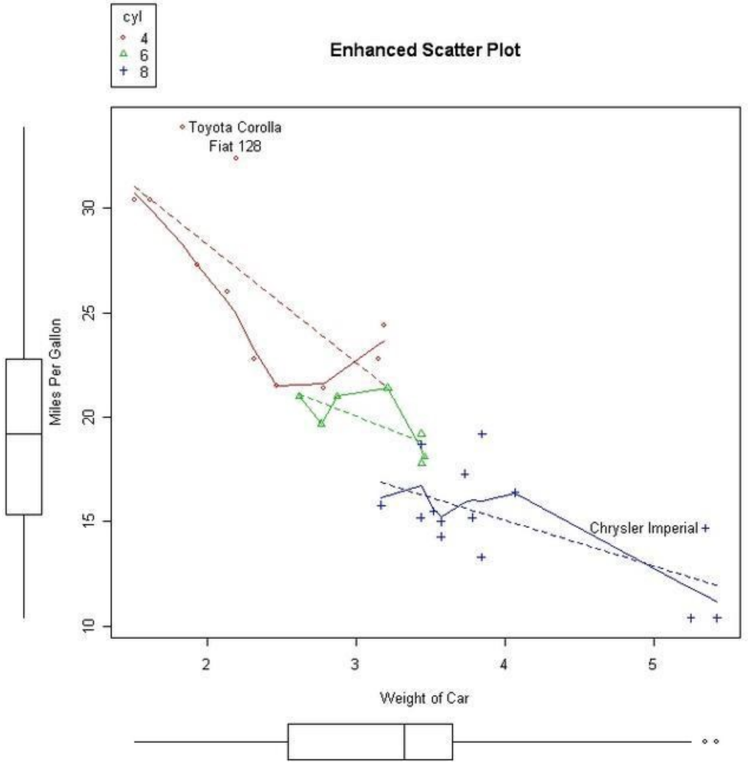
Scatterplots

- The scatterplot( ) function in the car package offers many enhanced features, including fit lines, marginal box plots, conditioning on a factor, and interactive point identification. Each of these features is optional.

Enhanced Scatterplot of MPG vs. Weight

by Number of Car Cylinders

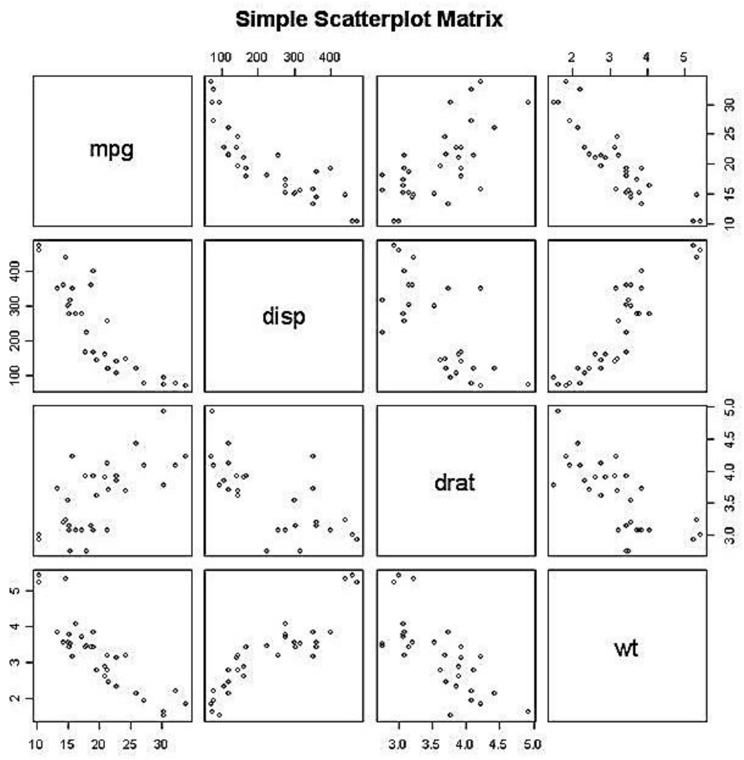
```
library(car)
scatterplot(mpg ~ wt | cyl, data=mtcars,
        xlab="Weight of Car", ylab="Miles
Per Gallon",
  main="Enhanced Scatter Plot",
  labels=row.names(mtcars))
```



Scatterplot Matrices

Basic Scatterplot Matrix
Under graphics package

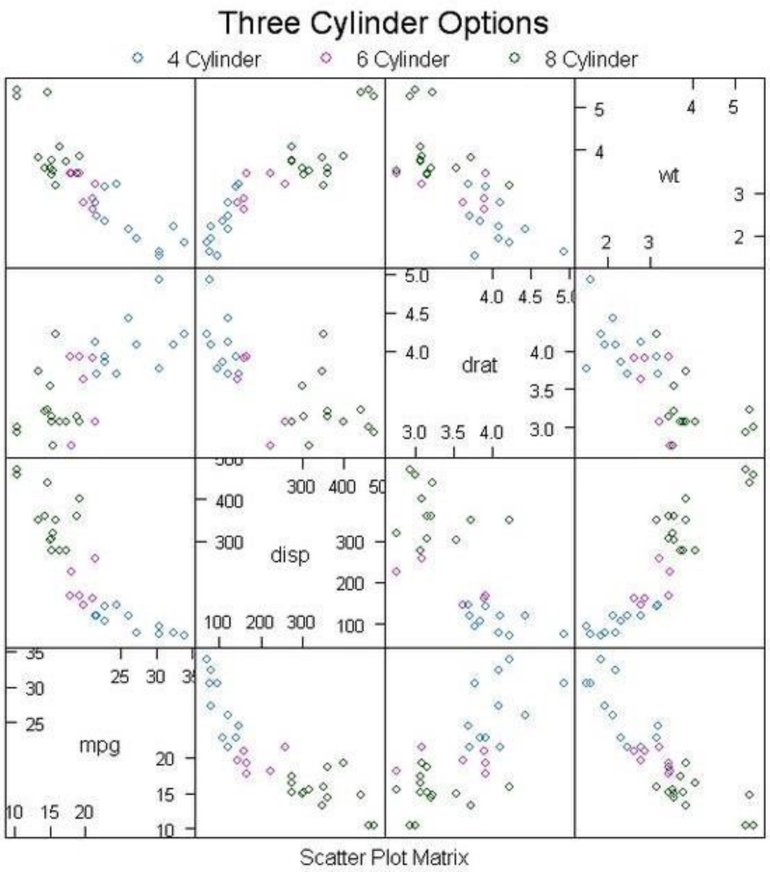
```
pairs(~mpg+disp+drat+wt,data=mtcars,
  main="Simple Scatterplot Matrix")
```



Scatterplot Matrices

Scatterplot Matrices from the lattice Package

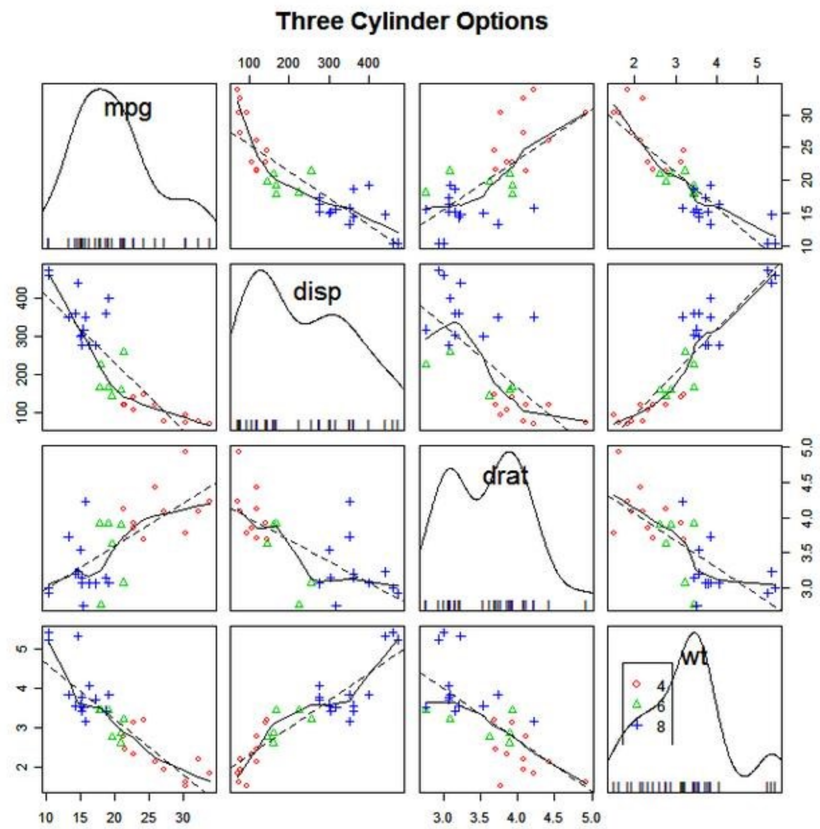
```
library(lattice)
splom(mtcars[c(1,3,5,6)], groups=cyl, data=mtcars,
        panel=panel.superpose,
  key=list(title="Three Cylinder Options",
  columns=3,
  points=list(pch=super.sym$pch[1:3],
  col=super.sym$col[1:3]),
  text=list(c("4 Cylinder","6 Cylinder","8 Cylinder"))))
```



Scatterplot Matrices

Scatterplot Matrices from the car Package

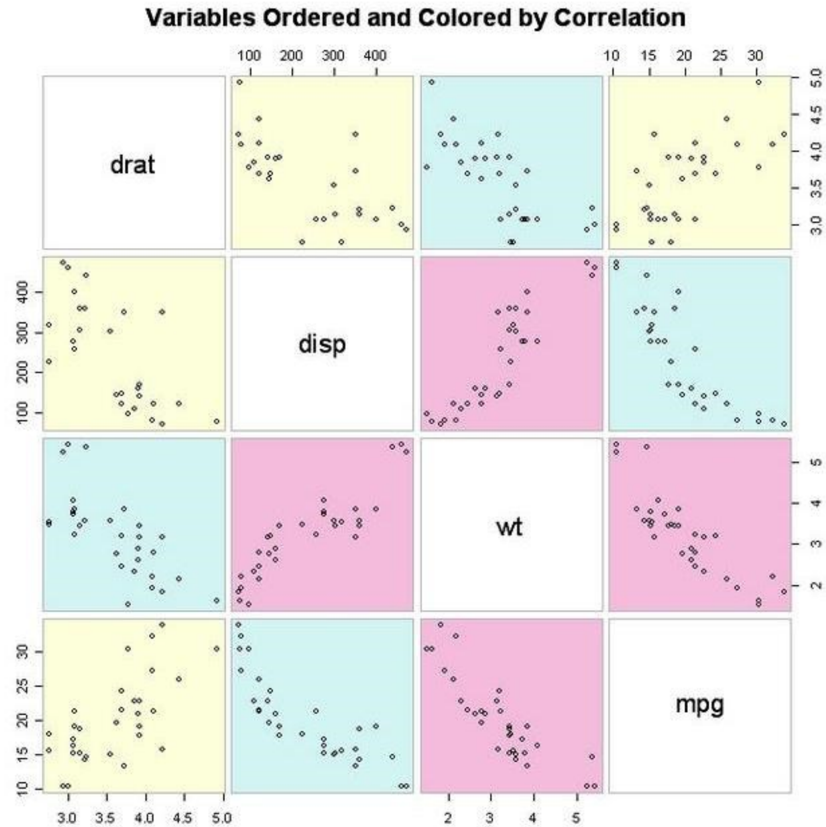
```
library(car)
scatterplotMatrix(~mpg+disp+drat+wt|cyl,
data=mtcars,
             main="Three Cylinder Options")
```



Scatterplot Matrices

Scatterplot Matrices from the gclus Package

```
library(gclus)
dta <- mtcars[c(1,3,5,6)] # get data
dta.r <- abs(cor(dta)) # get correlations
dta.col <- dmat.color(dta.r) # get colors
# reorder variables so those with highest
correlation
# are closest to the diagonal
dta.o <- order.single(dta.r)
cpairs(dta, dta.o, panel.colors=dta.col, gap=.5,
main="Variables Ordered and Colored by
Correlation" )
```



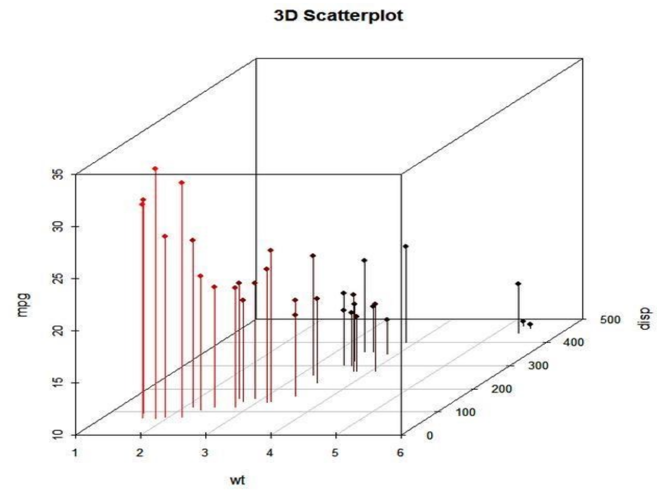
3-D Scatterplots

3D Scatterplots

Create a 3D scatterplot with the scatterplot3d package.
Use the function scatterplot3d(x, y, z).

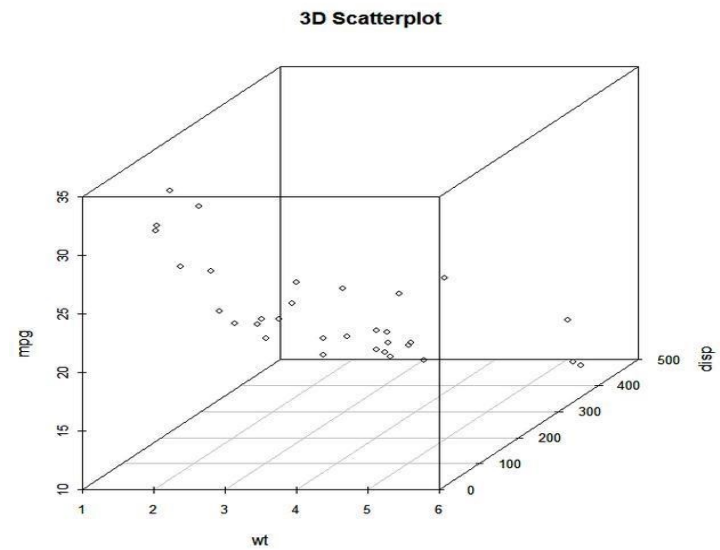
3D Scatterplot

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



3D Scatterplot with Coloring and Vertical Drop Lines

```
library(scatterplot3d)
attach(mtcars)
scatterplot3d(wt,disp,mpg, pch=16,
highlight.3d=TRUE,
  type="h", main="3D Scatterplot")
```

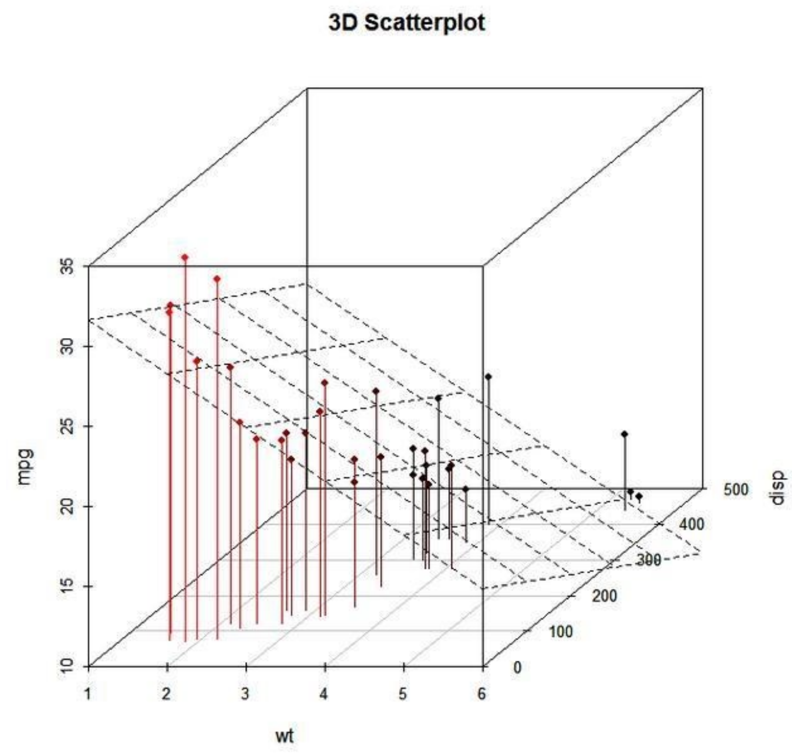


3D Scatterplots

3D Scatterplot with Coloring and Vertical Lines

and Regression Plane

```
library(scatterplot3d)
attach(mtcars)
s3d <-scatterplot3d(wt,disp,mpg, pch=16, highlight.3d=TRUE,
  type="h", main="3D Scatterplot")
fit <- lm(mpg ~ wt+disp)
s3d$plane3d(fit)
```



Interactive Graphs

Interactive Graphs

- Package iplots can be used to make a wide variety of plots including boxplots, bar charts, parallel coordinates, etc

```
install.packages ("iplots")
library(iplots)
ipcp (mtcars)
```

 # Interactive Parallel Coordinates

```
install.packages ("iplots")
library(iplots)
myCars <- data.frame(cbind(mtcars$cyl, mtcars$carb,
mtcars$gear))
```

 # Interactive Mosaic Plots

```
install.packages ("iplots")
library(iplots)
ibar (mtcars$mpg)
ibox (mtcars$disp, mtcars$carb, mtcars$cyl)
```

 # Multiple Interactive Charts

Any Questions?