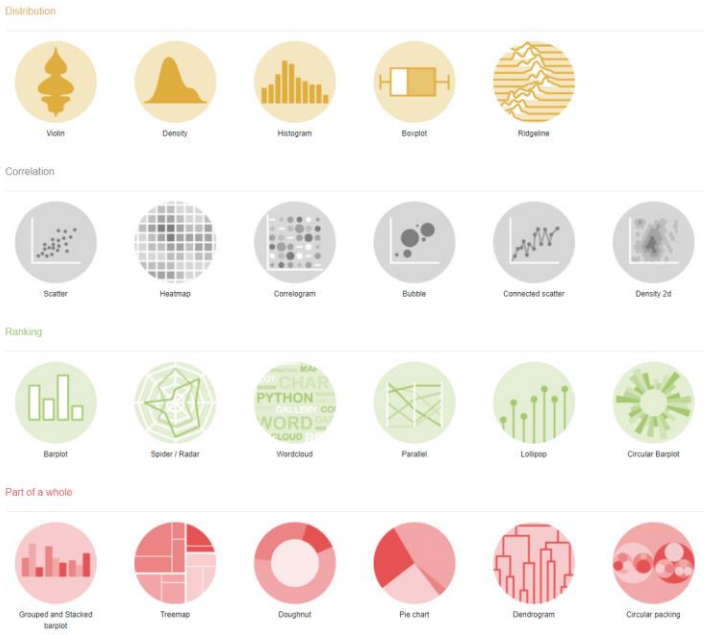


R Graph Basics Part 1

Pilsung Kang

School of Industrial Management Engineering

Korea University

R Graphics

- Basic functions that are provided by "graphics" package

Graphics package function	Description
barplot	Bar and column charts
dotchart	Cleveland dot plots
hist	Histograms
density	Kernel density plots
stripchart	Strip charts
qqnorm (in stats package)	Quantile-quantile plots
xplot	Scatter plots
smoothScatter	Smooth scatter plots
qqplot (in stats package)	Quantile-quantile plots
pairs	Scatter plot matrices
image	Image plots
contour	Contour plots
persp	Perspective charts of three-dimensional data
interaction.plot	Summary of the response for two-way combinations of factors
sunflowerplot	Sunflower plots

R Graphics

- Fisher's Iris dataset (default dataset provided by R)
  - ✓ Five variables
    - sepal length in cm, sepal width in cm, petal length in cm, petal width in cm, and
    - Species : Iris Setosa, Iris Versicolour, and Iris Virginica.

Setosa

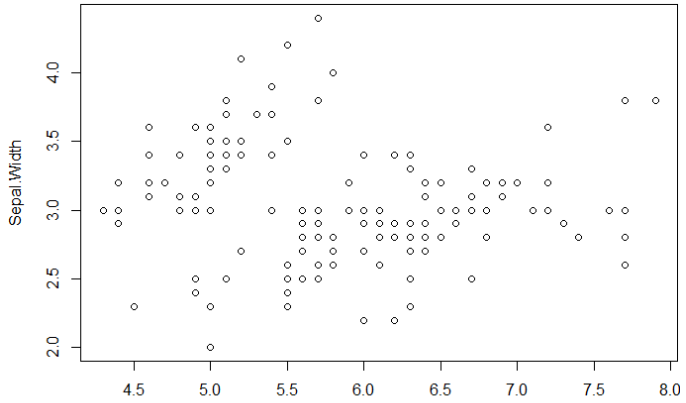
Versicolor

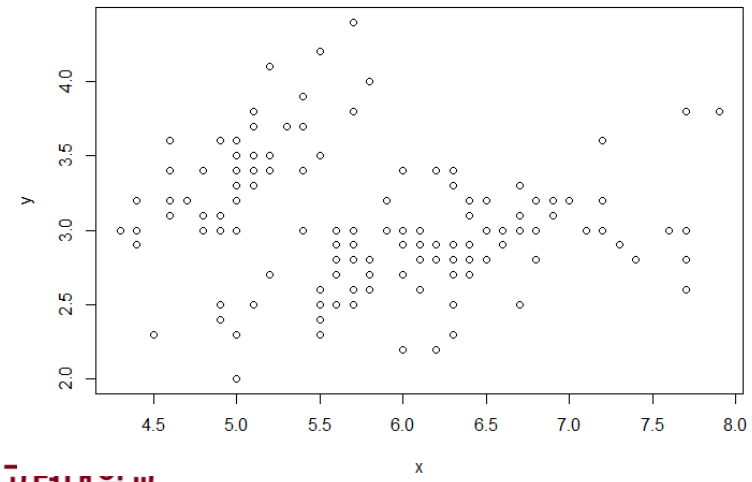
Virginica

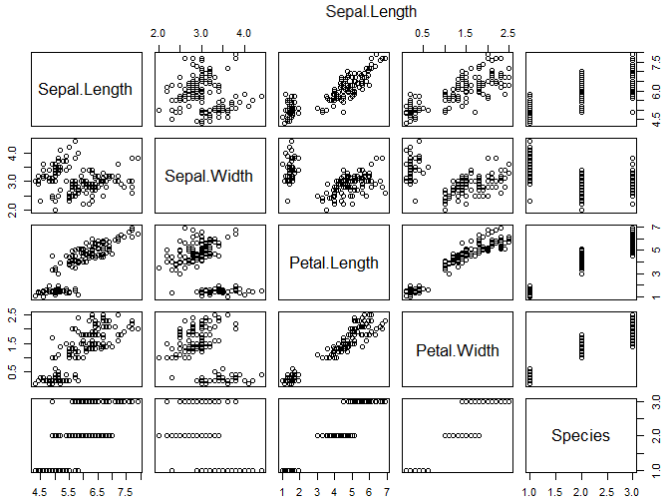
R Graphics

- Polymorphism of R graph functions
  - ✓ polymorphic function: has different operations for different arguments
  - ✓ ex: plot( )

```
Console ~/
> data(iris)
> x <- iris[,1]
> y <- iris[,2]
> subiris <- iris[,1:2]
> plot(x,y)
> plot(subiris)
> plot(iris)
```



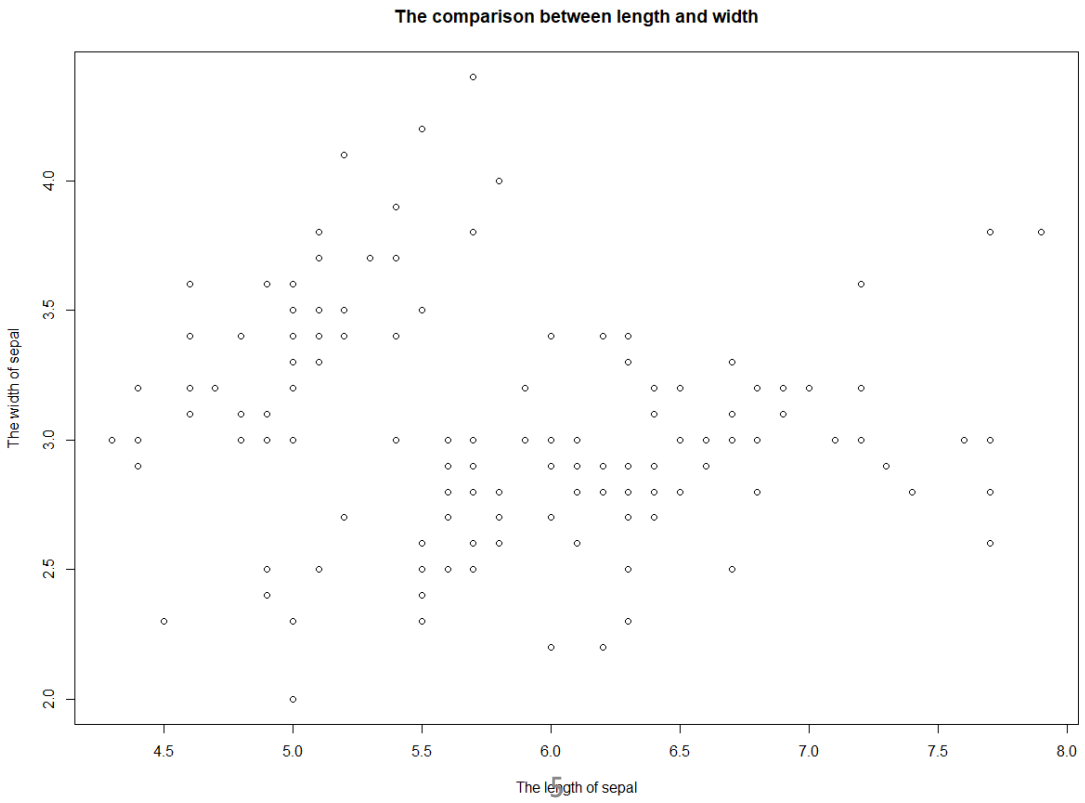




R Graphics

- Titles and labels in a graph
  - ✓ title: main, x-axis label: xlab, y-axis label: ylab

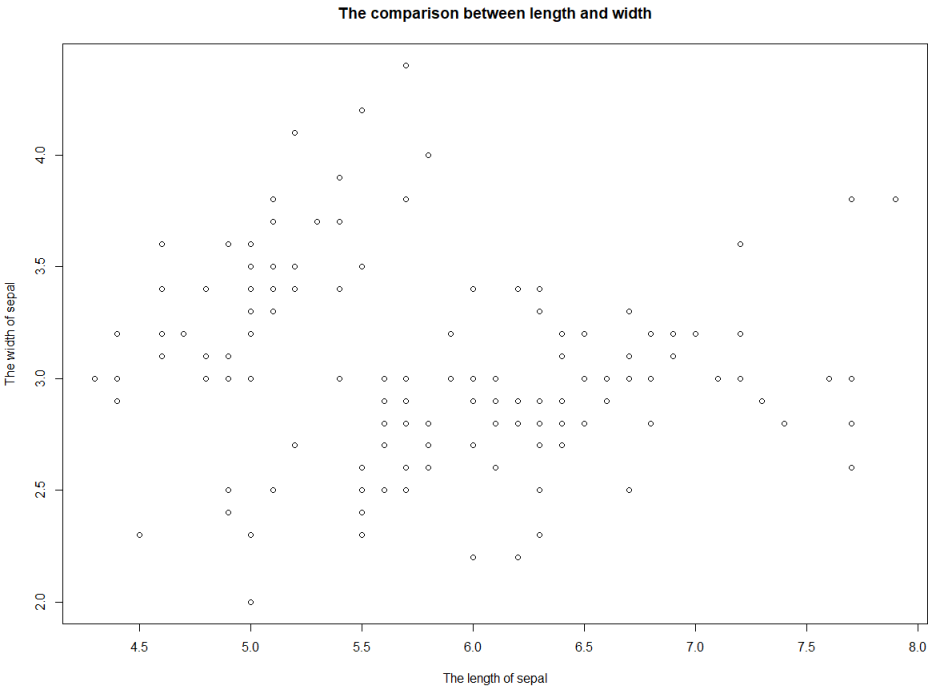
```
# Add title and x,y labels
plot(subiris, main="The comparison between length and width",
        xlab = "The length of sepal", ylab = "The width of sepal")
```

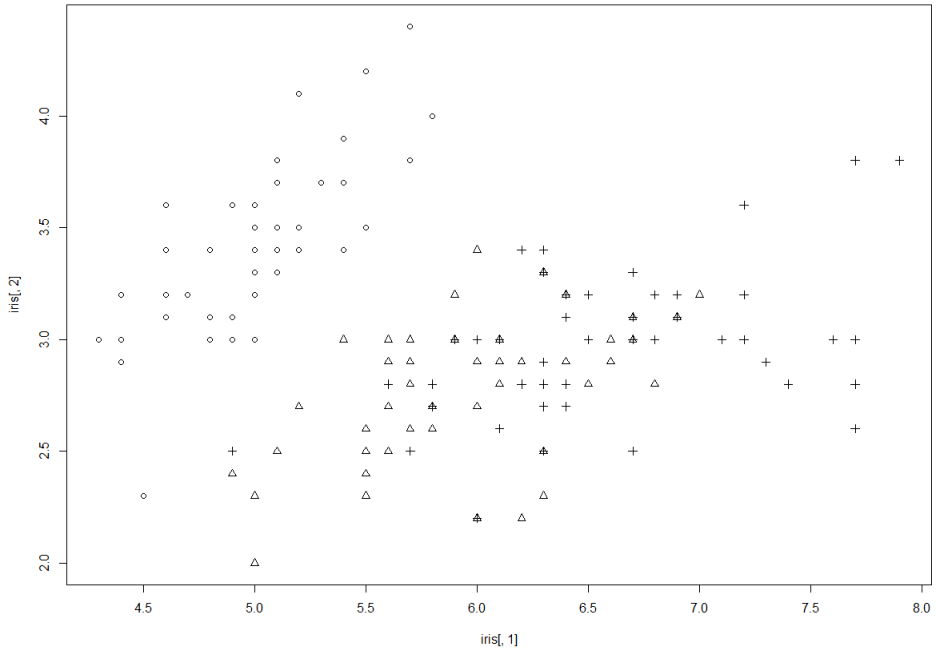


R Graphics

- Some options for basic R graphs
  - ✓ pch: shape, cex: size, col: color

```
# Scatter plot with different shapes for different classes
plot(iris[,1],iris[,2],pch=as.integer(iris[,5]))
```

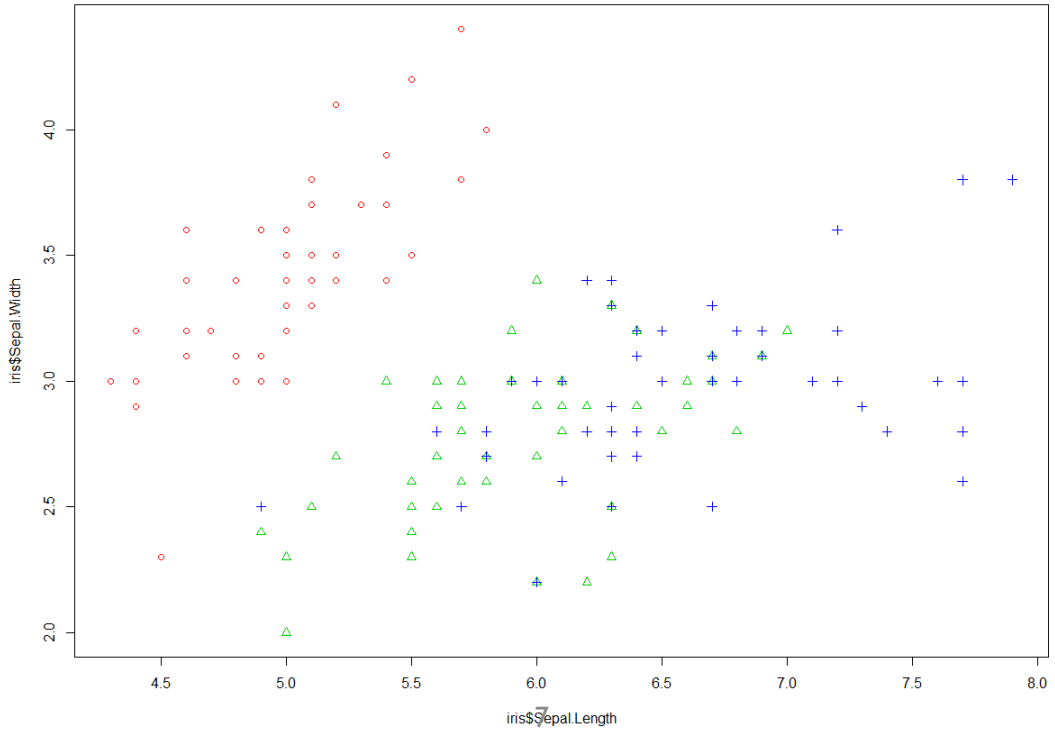




R Graphics

- Symbols in graphs
  - ✓ pch: shape, cex: size, col: color

```
# Scatter plot with different shapes & colors for different classes
plot(iris$Sepal.Length,iris$Sepal.Width,
     pch=as.integer(iris$Species),col=as.integer(iris$Species)+1)
```



R Graphics

- Options for better readability

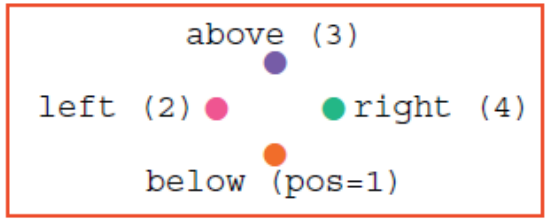
A: Plot symbols and text; specify colors and/or character expansion; draw rectangle

```
par(fig=c(0, 1, 0.415, 1))
plot(0, 0, xlim=c(0, 13), ylim=c(0, 19), type="n")
xpos <- rep((0:12)+0.5, 2);  ypos <- rep(c(14.5,12.75), c(13,13))
points(xpos, ypos, cex=2.5, col=1:26, pch=0:25)
text(xpos, ypos, labels=paste(0:25), cex=0.75)
```


```
## Plot characters, vary cex (expansion)
text((0:4)+0.5, rep(9*ht, 5), letters[1:5], cex=c(2.5,2,1,1.5,2))
```

a b c d e

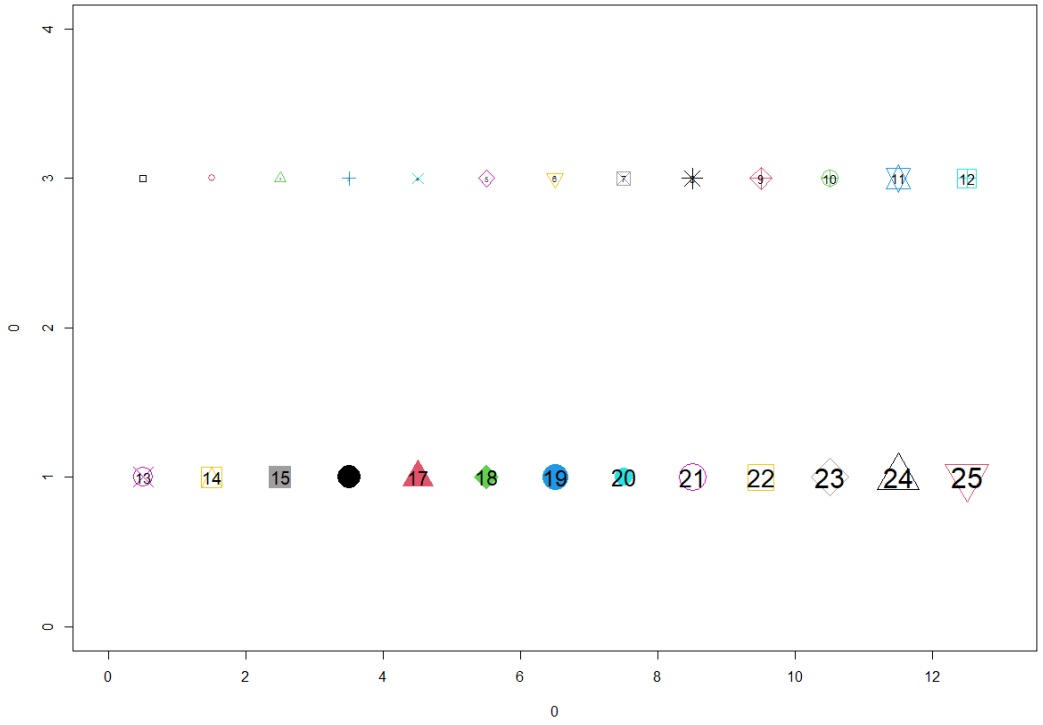
```
## Position label with respect to point
xmid <- 10.5; xoff <- c(0, -0.5, 0, 0.5)
ymid <- 5.8; yoff <- c(-1,0,1,0)
col4 <- colors()[c(52, 116, 547, 610)]
points(xmid+xoff, ymid+yoff, pch=16, cex=1.5, col=col4)
posText <- c("below (pos=1)", "left (2)", "above (3)", "right (4)")
text(xmid+xoff, ymid+yoff, posText, pos=1:4)
rect(xmid-2.3, ymid-2.3, xmid+2.3, ymid+2.3, border="red")
```



R Graphics

- Options for better readability

```
# Predefined shapes and colors
plot(0,0, xlim=c(0,13), ylim=c(0,4), type="n")
xpos <- rep((0:12)+0.5,2)
ypos <- rep(c(3,1), c(13,13))
points(xpos, ypos, cex=seq(from=1,to=3,length=26), col=1:26, pch=0:25)
text(xpos, ypos, labels = paste(0:25), cex=seq(from=0.1,to=1,length=26))
```



R Graph Basic 1: Bar Plot

- Bar Plot: Usage

```
barplot(height, …)
# S3 method for default
barplot(height, width = 1, space = NULL,
        names.arg = NULL, legend.text = NULL, beside = FALSE,
        horiz = FALSE, density = NULL, angle = 45,
        col = NULL, border = par("fg"),
        main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
        xlim = NULL, ylim = NULL, xpd = TRUE, log = "",
        axes = TRUE, axisnames = TRUE,
        cex.axis = par("cex.axis"), cex.names = par("cex.axis"),
        inside = TRUE, plot = TRUE, axis.lty = 0, offset = 0,
        add = FALSE, ann = !add && par("ann"), args.legend = NULL, …)
```

R Graph Basic 1: Bar Plot

- Bar Plot: Arguments

Arguments

height	either a vector or matrix of values describing the bars which make up the plot. If `height` is a vector, the plot consists of a sequence of rectangular bars with heights given by the values in the vector. If `height` is a matrix and `beside` is `FALSE` then each bar of the plot corresponds to a column of `height`, with the values in the column giving the heights of stacked sub-bars making up the bar. If `height` is a matrix and `beside` is `TRUE`, then the values in each column are juxtaposed rather than stacked.
width	optional vector of bar widths. Re-cycled to length the number of bars drawn. Specifying a single value will have no visible effect unless `xlim` is specified.
space	the amount of space (as a fraction of the average bar width) left before each bar. May be given as a single number or one number per bar. If `height` is a matrix and `beside` is `TRUE`, `space` may be specified by two numbers, where the first is the space between bars in the same group, and the second the space between the groups. If not given explicitly, it defaults to `c(0,1)` if `height` is a matrix and `beside` is `TRUE`, and to 0.2 otherwise.
names.arg	a vector of names to be plotted below each bar or group of bars. If this argument is omitted, then the names are taken from the `names` attribute of `height` if this is a vector, or the column names if it is a matrix.
legend.text	a vector of text used to construct a legend for the plot, or a logical indicating whether a legend should be included. This is only useful when `height` is a matrix. In that case given legend labels should correspond to the rows of `height`; if `legend.text` is true, the row names of `height` will be used as labels if they are non-null.
beside	a logical value. If `FALSE`, the columns of `height` are portrayed as stacked bars, and if `TRUE` the columns are portrayed as juxtaposed bars.
horiz	a logical value. If `FALSE`, the bars are drawn vertically with the first bar to the left. If `TRUE`, the bars are drawn horizontally with the first at the bottom.
density	a vector giving the density of shading lines, in lines per inch, for the bars or bar components. The default value of `NULL` means that no shading lines are drawn. Non-positive values of `density` also inhibit the drawing of shading lines.
angle	the slope of shading lines, given as an angle in degrees (counter-clockwise), for the bars or bar components.
col	a vector of colors for the bars or bar components. By default, grey is used if `height` is a vector, and a gamma-corrected grey palette if `height` is a matrix.
border	the color to be used for the border of the bars. Use `border = NA` to omit borders. If there are shading lines, `border = TRUE` means use the same colour for the border as for the shading lines.
main,sub	overall and sub title for the plot.

R Graph Basic 1: Bar Plot

- Bar Plot: Arguments

xlab	a label for the x axis.
ylab	a label for the y axis.
xlim	limits for the x axis.
ylim	limits for the y axis.
xpd	logical. Should bars be allowed to go outside region?
log	string specifying if axis scales should be logarithmic; see `plot.default`.
axes	logical. If `TRUE`, a vertical (or horizontal, if `horiz` is true) axis is drawn.
axisnames	logical. If `TRUE`, and if there are `names.arg` (see above), the other axis is drawn (with `lty = 0`) and labeled.
cex.axis	expansion factor for numeric axis labels.
cex.names	expansion factor for axis names (bar labels).
inside	logical. If `TRUE`, the lines which divide adjacent (non-stacked!) bars will be drawn. Only applies when `space = 0` (which it partly is when `beside = TRUE`).
plot	logical. If `FALSE`, nothing is plotted.
axis.lty	the graphics parameter `lty` applied to the axis and tick marks of the categorical (default horizontal) axis. Note that by default the axis is suppressed.
offset	a vector indicating how much the bars should be shifted relative to the x axis.
add	logical specifying if bars should be added to an already existing plot; defaults to `FALSE`.
ann	logical specifying if the default annotation (`main`, `sub`, `xlab`, `ylab`) should appear on the plot, see `title`.
args.legend	list of additional arguments to pass to `legend()`; names of the list are used as argument names. Only used if `legend.text` is supplied.

formula	a formula where the `y` variables are numeric data to plot against the categorical `x` variables. The formula can have one of three forms:
	`y ~ x`
	`y ~ x1 + x2`
	`cbind(y1, y2) ~ x`
	, see the examples.

data	a data frame (or list) from which the variables in formula should be taken.
subset	an optional vector specifying a subset of observations to be used.
na.action	a function which indicates what should happen when the data contain `NA` values. The default is to ignore missing values in the given variables.
...	arguments to be passed to/from other methods. For the default method these can include further arguments (such as `axes`, `asp` and `main`) and graphical parameters (see `par`) which are passed to `plot.window()`, `title()` and `axis`.

R Graph Basic 1: Bar Plot

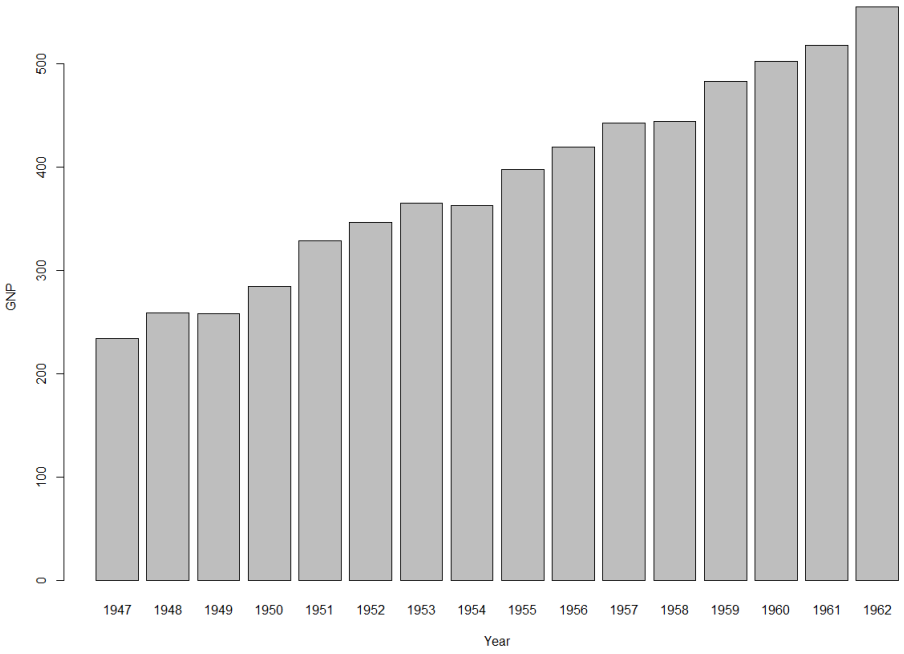
- Bar Plot: Example
  - ✓ "longley" dataset: A macroeconomic data set which provides a well-known example for a highly collinear regression.

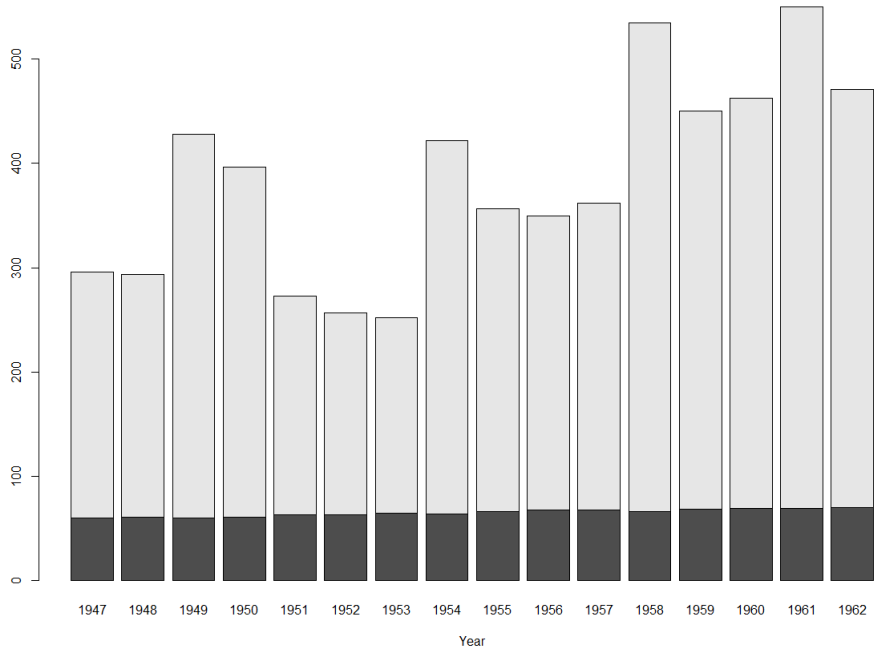
	GNP.deflator	GNP	Unemployed	Armed.Forces	Population	Year	Employed
1947	83.0	234.289	235.6	159.0	107.608	1947	60.323
1948	88.5	259.426	232.5	145.6	108.632	1948	61.122
1949	88.2	258.054	368.2	161.6	109.773	1949	60.171
1950	89.5	284.599	335.1	165.0	110.929	1950	61.187
1951	96.2	328.975	209.9	309.9	112.075	1951	63.221
1952	98.1	346.999	193.2	359.4	113.270	1952	63.639
1953	99.0	365.385	187.0	354.7	115.094	1953	64.989
1954	100.0	363.112	357.8	335.0	116.219	1954	63.761
1955	101.2	397.469	290.4	304.8	117.388	1955	66.019
1956	104.6	419.180	282.2	285.7	118.734	1956	67.857
1957	108.4	442.769	293.6	279.8	120.445	1957	68.169
1958	110.8	444.546	468.1	263.7	121.950	1958	66.513
1959	112.6	482.704	381.3	255.2	123.366	1959	68.655
1960	114.2	502.601	393.1	251.4	125.368	1960	69.564
1961	115.7	518.173	480.6	257.2	127.852	1961	69.331
1962	116.9	554.894	400.7	282.7	130.081	1962	70.551

R Graph Basic 1: Bar Plot

- Bar Plot: Example 1

```
# Basic plot 1: bar plot
View(longley)
barplot(GNP ~ Year, data = longley)
barplot(cbind(Employed, Unemployed) ~ Year, data = longley)
```





R Graph Basic 1: Bar Plot

- Bar Plot: Example 2

```
data(Titanic)
View(Titanic)
summary(d.Titanic <- as.data.frame(Titanic))
```

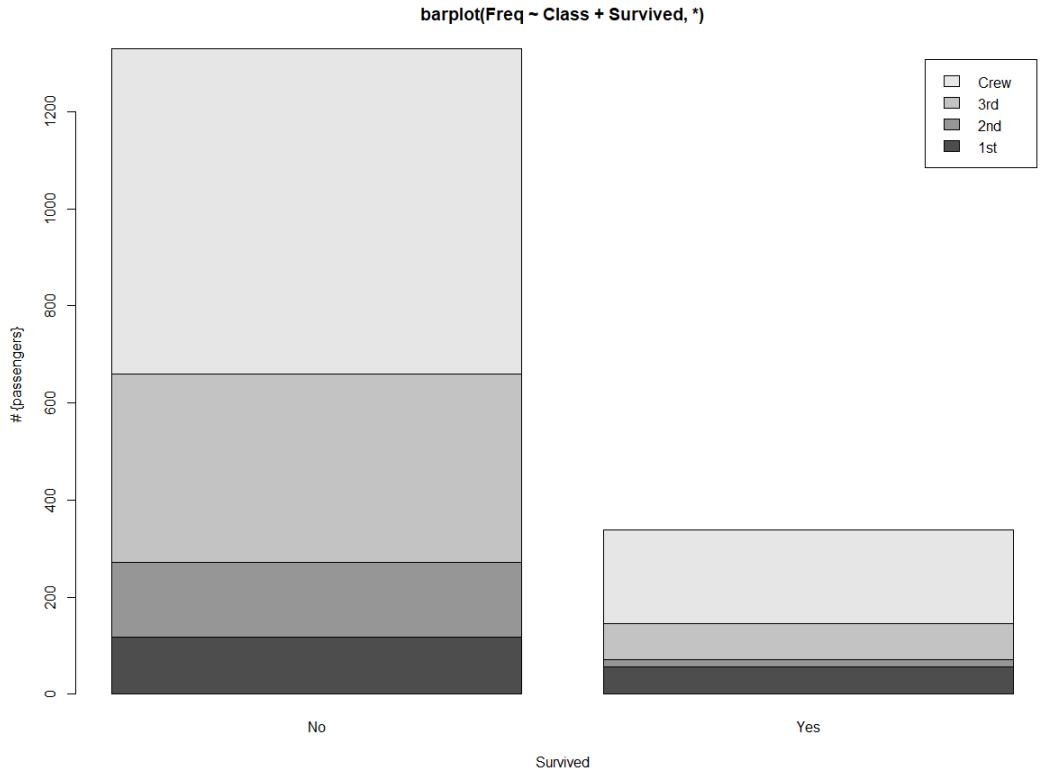
	Class	Sex	Age	Survived	Freq
1	1st	Male	Child	No	0
2	2nd	Male	Child	No	0
3	3rd	Male	Child	No	35
4	Crew	Male	Child	No	0
5	1st	Female	Child	No	0
6	2nd	Female	Child	No	0
7	3rd	Female	Child	No	17
8	Crew	Female	Child	No	0
9	1st	Male	Adult	No	118
10	2nd	Male	Adult	No	154
11	3rd	Male	Adult	No	387
12	Crew	Male	Adult	No	670
13	1st	Female	Adult	No	4
14	2nd	Female	Adult	No	13
15	3rd	Female	Adult	No	89
16	Crew	Female	Adult	No	3
17	1st	Male	Child	Yes	5
18	2nd	Male	Child	Yes	11
19	3rd	Male	Child	Yes	13
20	Crew	Male	Child	Yes	0
21	1st	Female	Child	Yes	1
22	2nd	Female	Child	Yes	13
23	3rd	Female	Child	Yes	14
24	Crew	Female	Child	Yes	0
25	1st	Male	Adult	Yes	57
26	2nd	Male	Adult	Yes	14
27	3rd	Male	Adult	Yes	75
28	Crew	Male	Adult	Yes	192
29	1st	Female	Adult	Yes	140
30	2nd	Female	Adult	Yes	80
31	3rd	Female	Adult	Yes	76
32	Crew	Female	Adult	Yes	20

```
> summary(d.Titanic <- as.data.frame(Titanic))
  Class       Sex         Age     Survived      Freq
 1st :8   Male   :16   Child:16   No :16   Min.   :  0.00
 2nd :8   Female:16   Adult:16   Yes:16   1st Qu.:  0.75
 3rd :8                                   Median : 13.50
 Crew:8                                   Mean   : 68.78
                                          3rd Qu.: 77.00
                                          Max.   :670.00
```

R Graph Basic 1: Bar Plot

- Bar Plot: Example 2

```
barplot(Freq ~ Class + Survived, data = d.Titanic,
        subset = Age == "Adult" & Sex == "Male",
        main = "barplot(Freq ~ Class + Survived, *)",
        ylab = "# {passengers}", legend = TRUE)
```

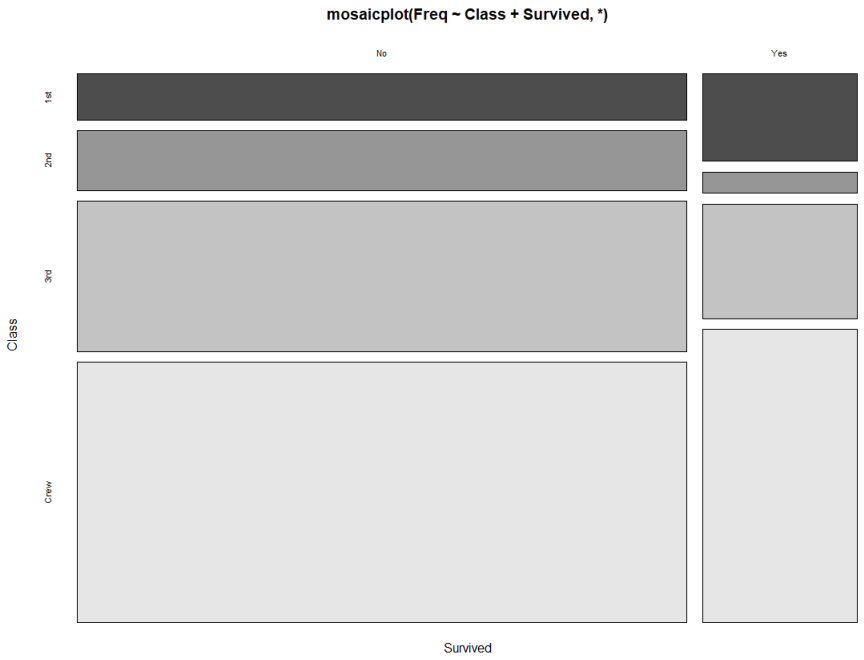


R Graph Basic 1: Bar Plot

- Bar Plot: Example 2

```
# Corresponding table :
(xt <- xtabs(Freq ~ Survived + Class + Sex, d.Titanic,
         subset = Age=="Adult"))
# Alternatively, a mosaic plot :
mosaicplot(xt[,,"Male"], main = "mosaicplot(Freq ~ Class + Survived, *)",
         color=TRUE)
```

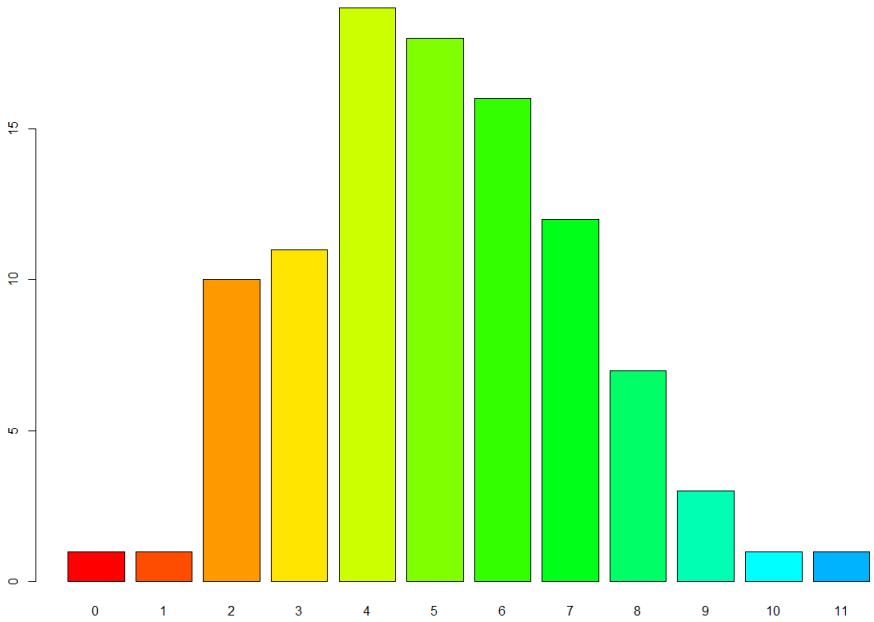
```
, , Sex = Male

        Class
Survived 1st 2nd 3rd Crew
     No  118 154 387  670
     Yes  57  14  75  192

, , Sex = Female

        Class
Survived 1st 2nd 3rd Crew
     No    4  13  89    3
     Yes 140  80  76   20
```

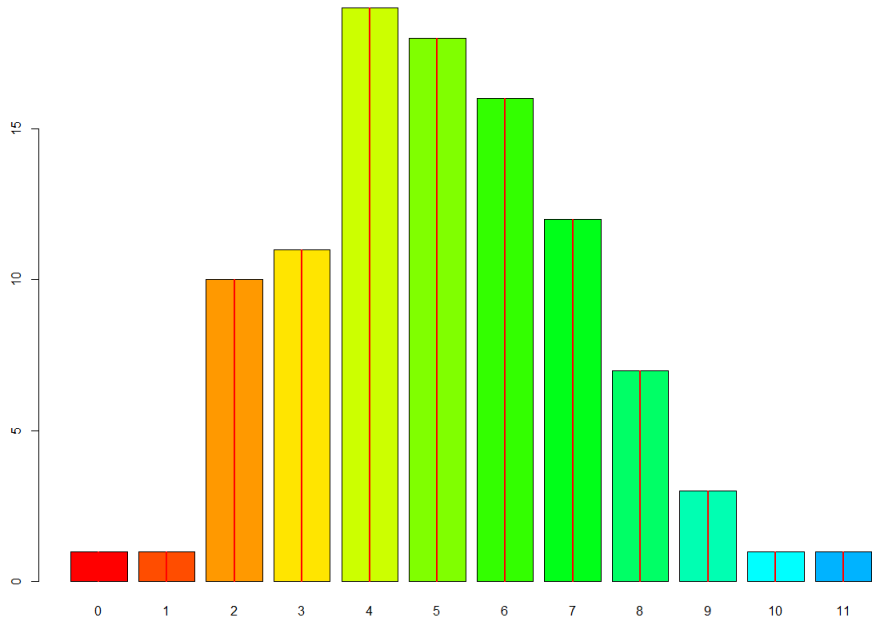


R Graph Basic 1: Bar Plot

- Bar Plot: Example 3 (Coloring)

```
# Coloring bar charts
require(grDevices) # for colours
tN <- table(Ni <- stats::rpois(100, lambda = 5))
r <- barplot(tN, col = rainbow(20))
#- type = "h" plotting *is* 'bar'plot
lines(r, tN, type = "h", col = "red", lwd = 2)
```

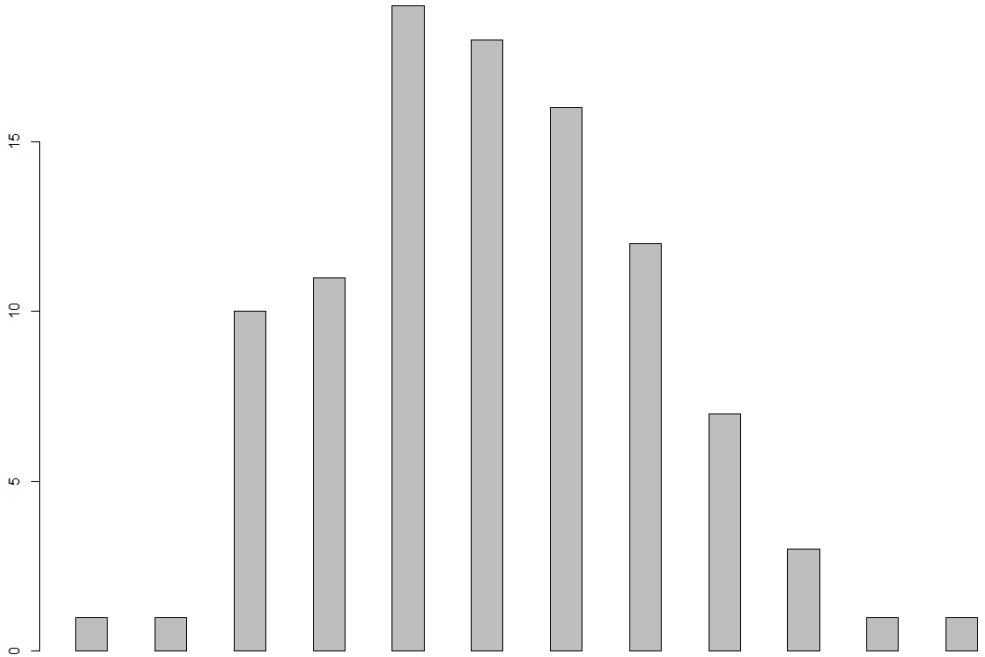




R Graph Basic 1: Bar Plot

- Bar Plot: Example 5 (Space between bars)

```
# Control the space between bars
barplot(tN, space = 1.5, axisnames = FALSE,
        sub = "barplot(..., space= 1.5, axisnames = FALSE)")
```

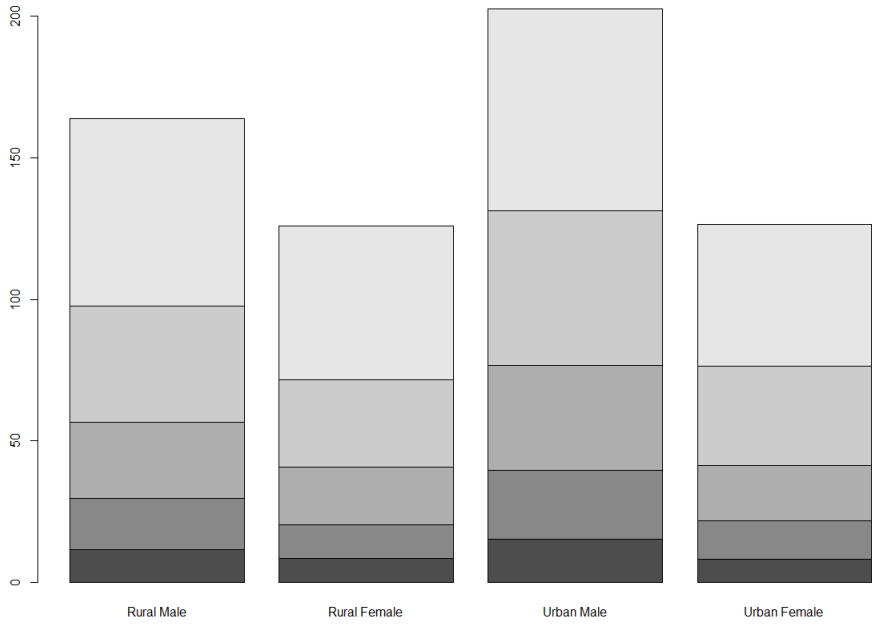


R Graph Basic 1: Bar Plot

- Bar Plot: Example 6
  - ✓ "VADeath" dataset: Death rates per 1000 in Virginia in 1940.

```
# VADeaths dataset
View(VADeaths)
barplot(VADeaths)
```

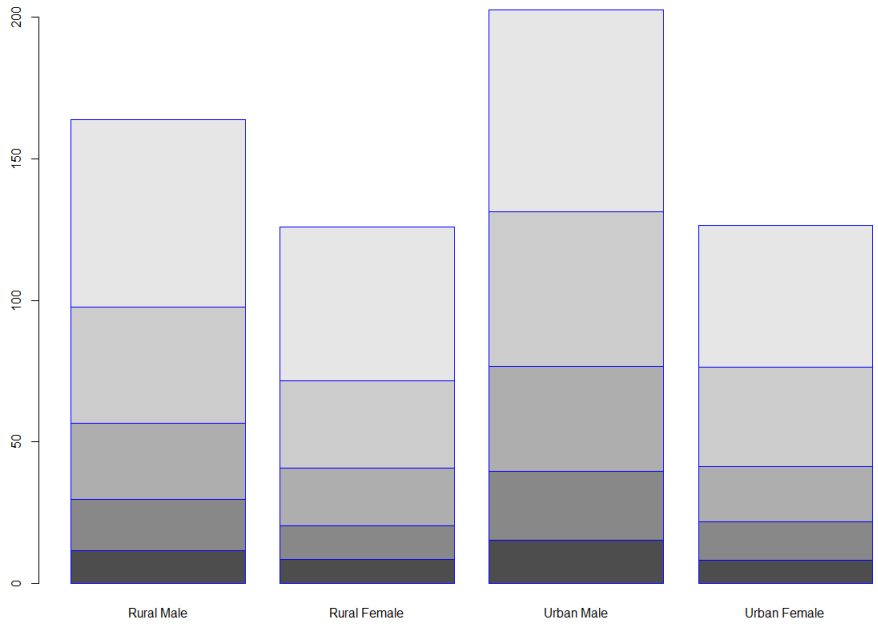
	Rural Male	Rural Female	Urban Male	Urban Female
50-54	11.7	8.7	15.4	8.4
55-59	18.1	11.7	24.3	13.6
60-64	26.9	20.3	37.0	19.3
65-69	41.0	30.9	54.6	35.1
70-74	66.0	54.3	71.1	50.0

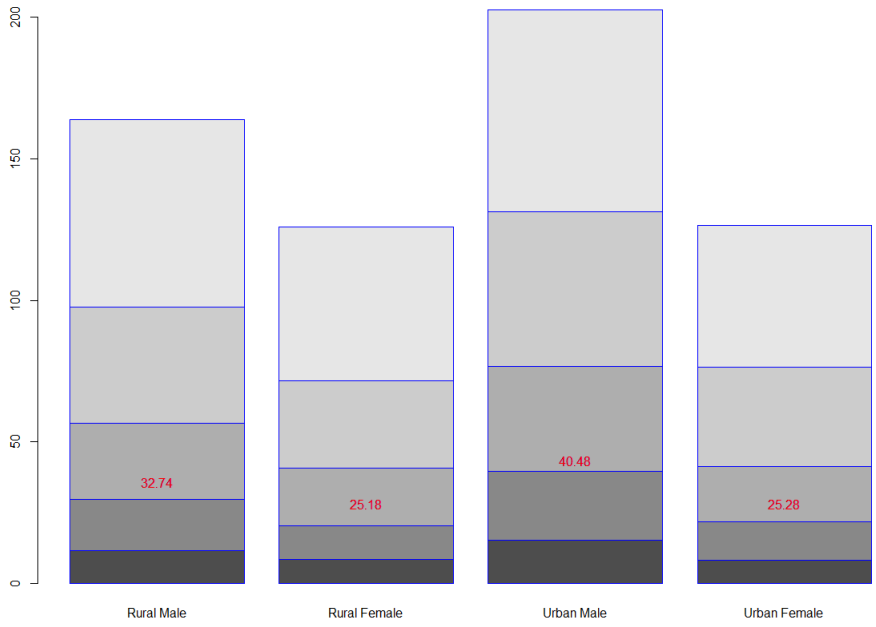


R Graph Basic 1: Bar Plot

- Bar Plot: Example 6

```
# Border color
barplot(VADeaths, border = "blue")
bar_VA <- barplot(VADeaths, border = "blue")
tot <- colMeans(VADeaths)
text(bar_VA, tot + 3, format(tot), xpd = TRUE, col = "red")
```

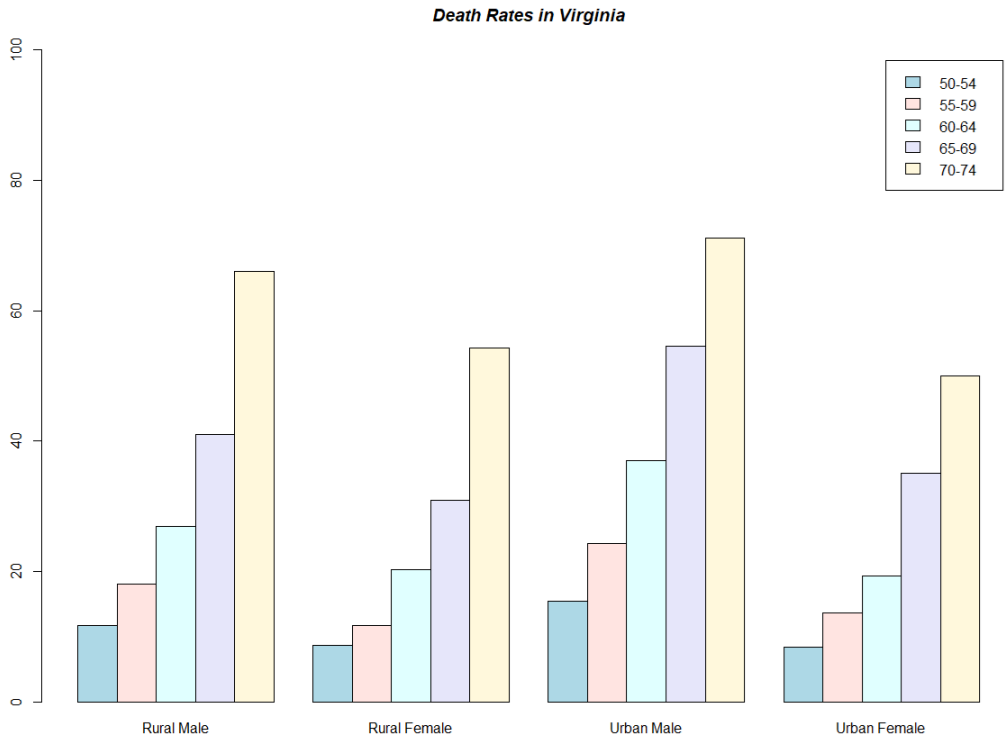




R Graph Basic 1: Bar Plot

- Bar Plot: Example 6

```
barplot(VADeaths, beside = TRUE,
        col = c("lightblue", "mistyrose", "lightcyan",
                "lavender", "cornsilk"),
        legend = rownames(VADeaths), ylim = c(0, 100))
title(main = "Death Rates in Virginia", font.main = 4)
```



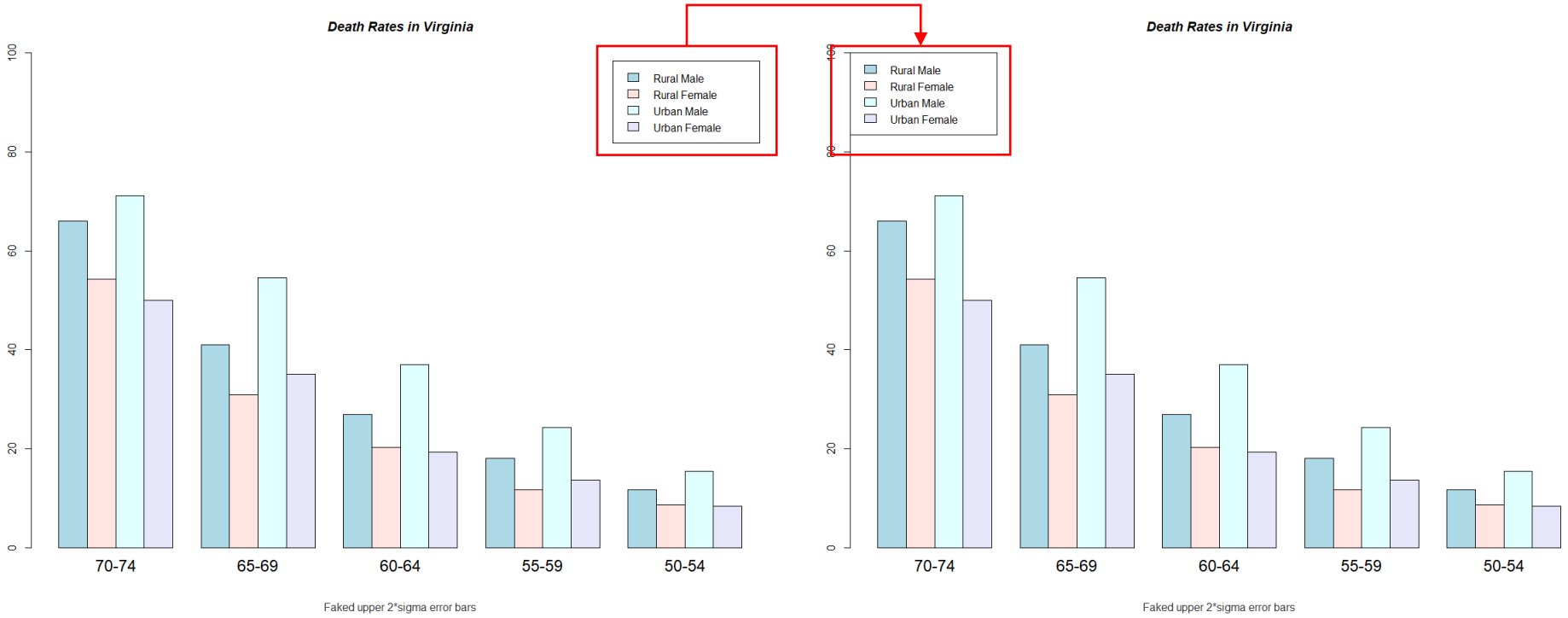
R Graph Basic 1: Bar Plot

- Bar Plot: Example 6

```
reverse_VA <- t(VADeaths)[, 5:1]
mybarcol <- "gray20"
barplot(reverse_VA, beside = TRUE,
        col = c("lightblue", "mistyrose", "lightcyan", "lavender"),
        legend = colnames(VADeaths), ylim = c(0,100),
        main = "Death Rates in Virginia", font.main = 4,
        sub = "Faked upper 2*sigma error bars",
        col.sub = mybarcol,
        args.legend = list(x = "topleft"), # option
        cex.names = 1.5)
```

R Graph Basic 1: Bar Plot

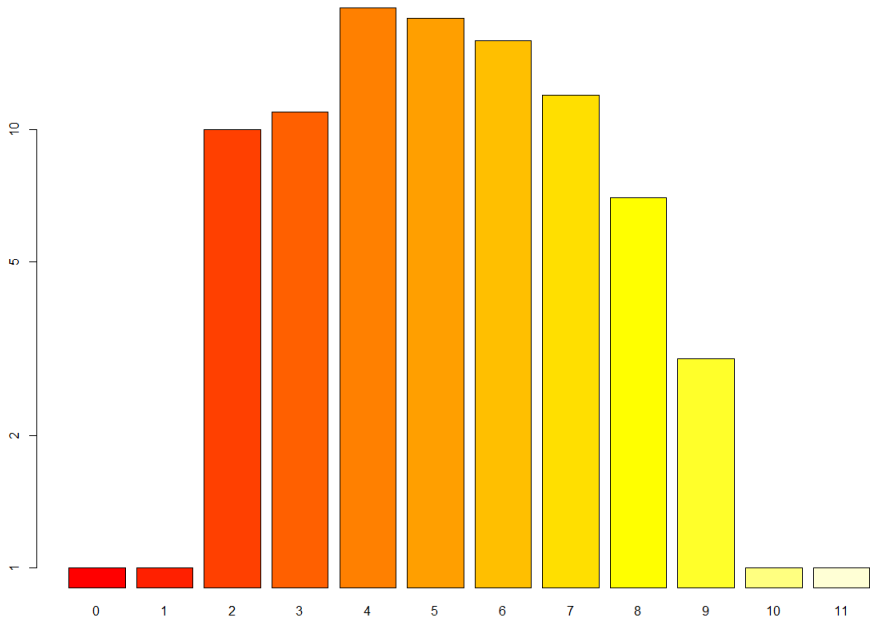
- Bar Plot: Example 6

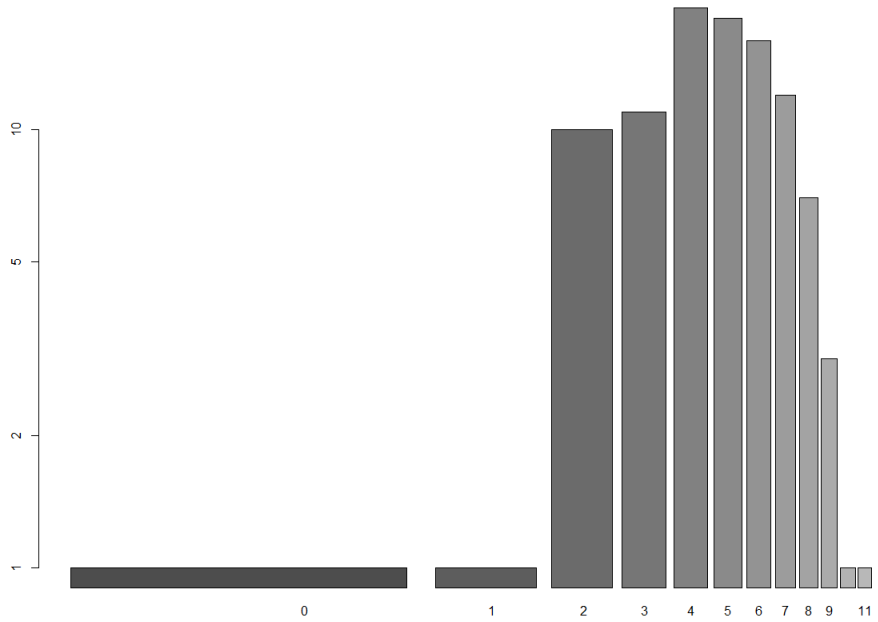


R Graph Basic 1: Bar Plot

- Bar Plot: Example 7

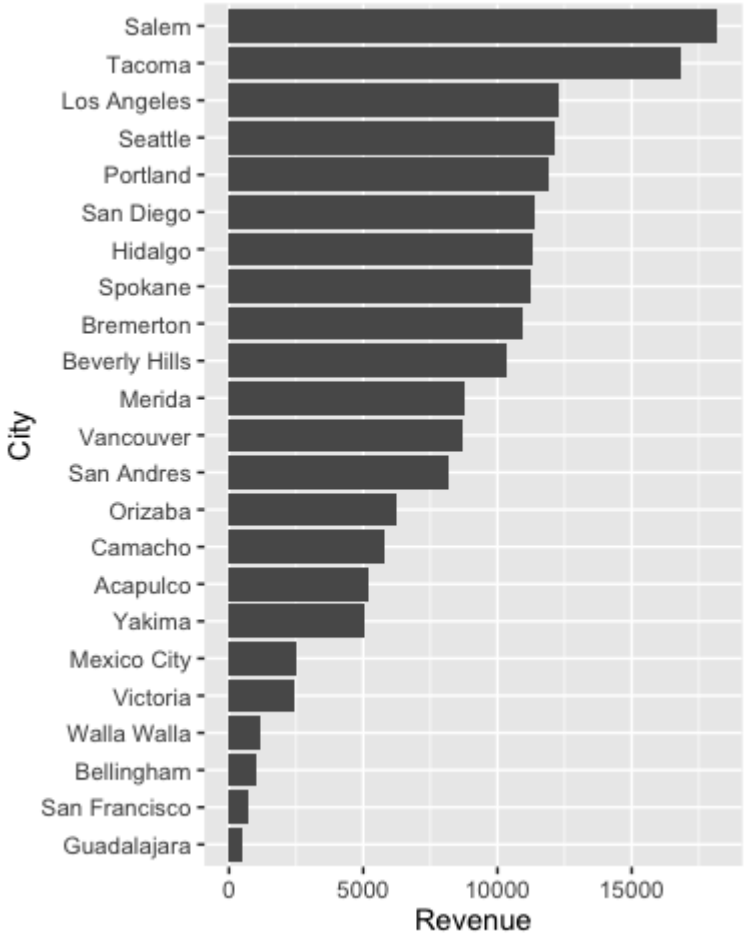
```
# Log scales
barplot(tN, col = heat.colors(12), log = "y")
barplot(tN, col = gray.colors(20), log = "xy")
```

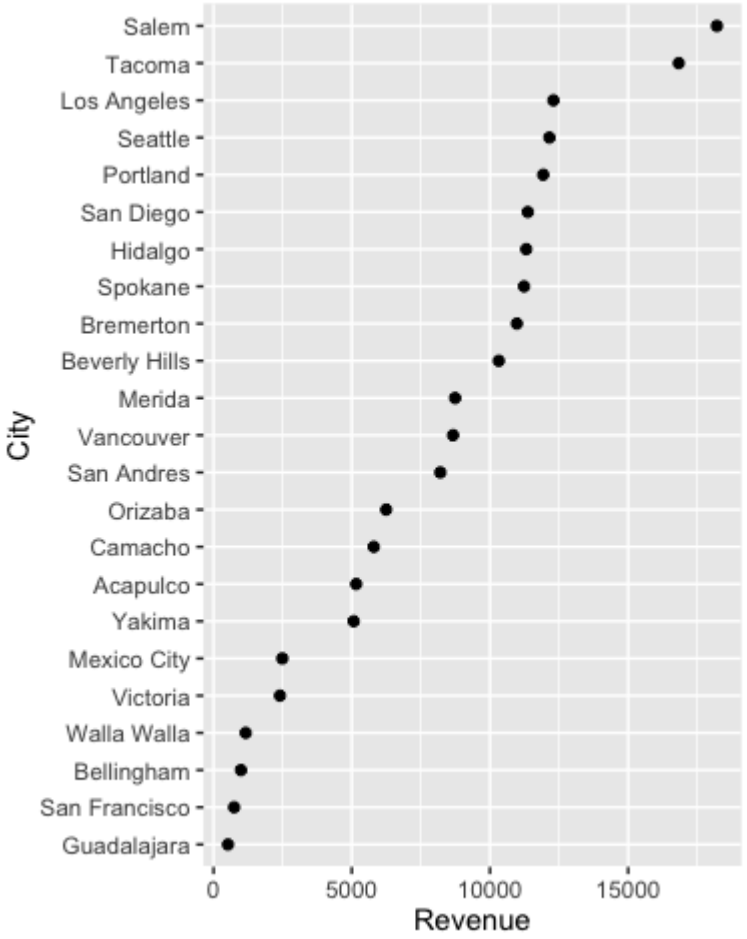




R Graph Basic 2: dot chart

- Dot chart as an alternative to a bar chart

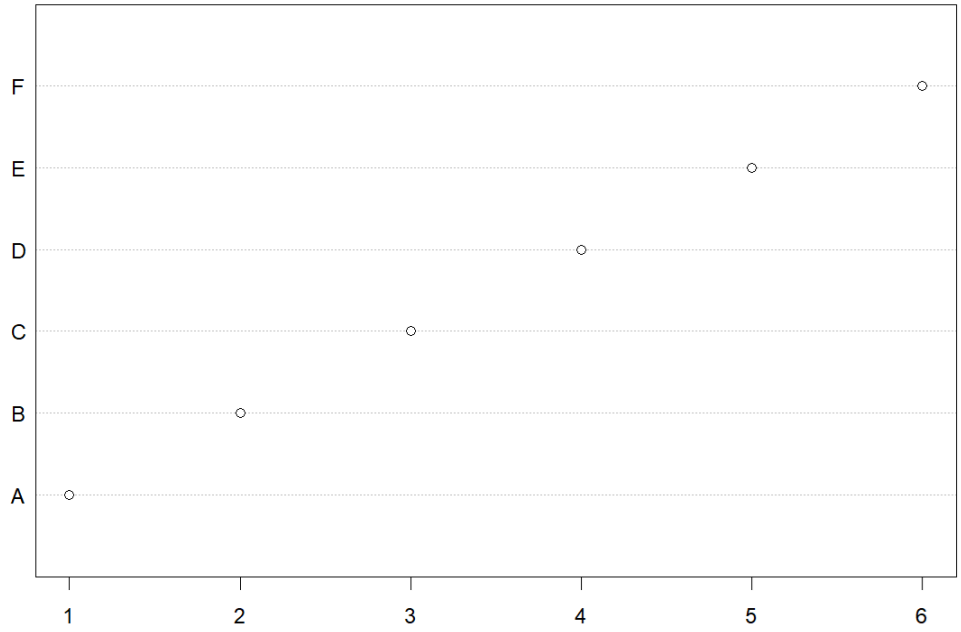




R Graph Basic 2: dot chart

- Dot chart: Example 1

```
# Basic plot 3: dotchart
vectorToPlot <- c(1:6)
names(vectorToPlot) <- c(LETTERS[1:6])
dotchart(vectorToPlot, cex = 1.5)
```



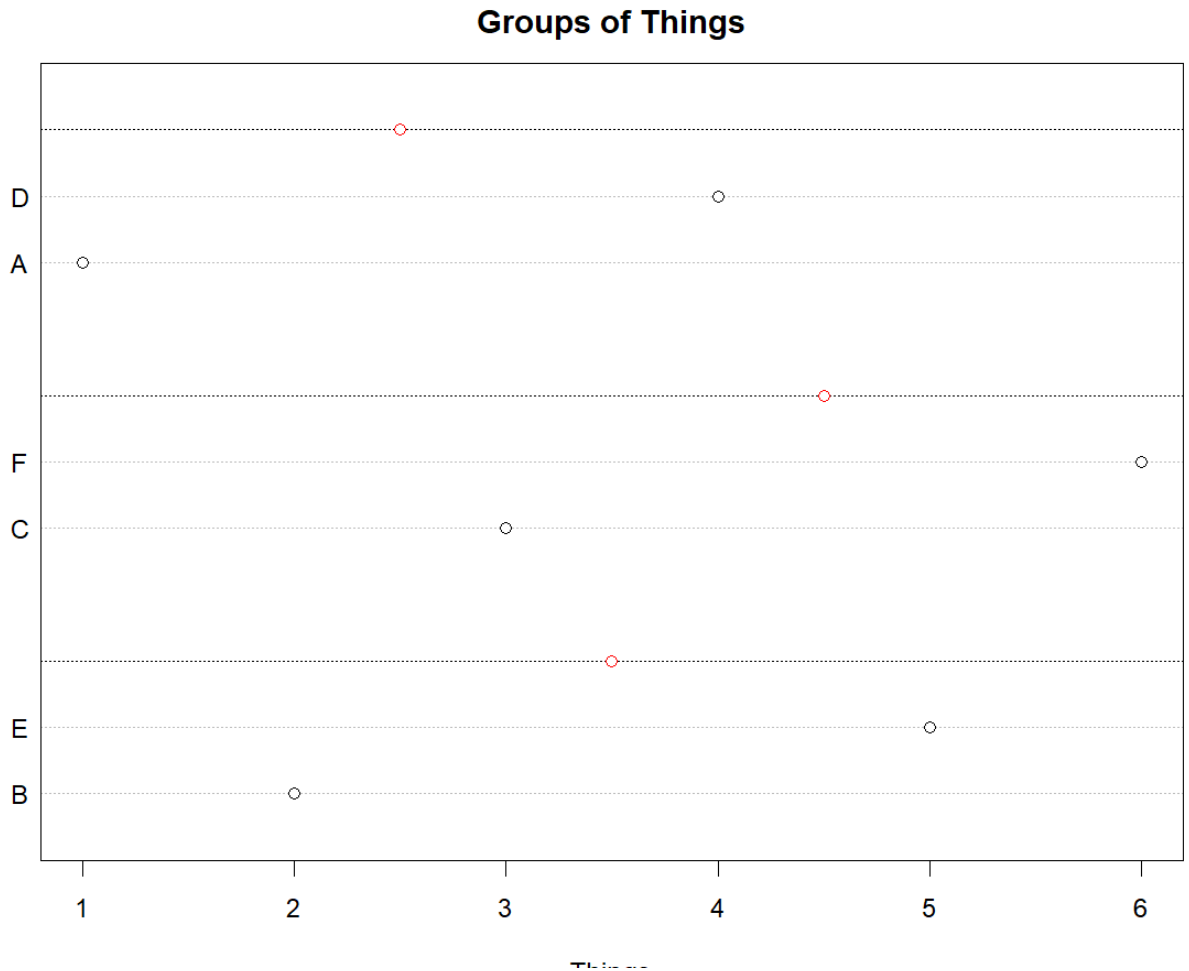
R Graph Basic 2: dot chart

- Dot chart: Example 1

```
myGroup <- factor(c("group1","group3","group2","group1","group3","group2"))
dotchart(vectorToPlot, groups = myGroup)
dotchart(vectorToPlot,
         gcolor = "red", groups = myGroup,
         gdata = c(median(vectorToPlot[myGroup == "group1"]),
                   median(vectorToPlot[myGroup == "group2"]),
                   median(vectorToPlot[myGroup == "group3"])),
         cex = 1.5,
         main = "Groups of Things", xlab = "Things")
```

R Graph Basic 2: dot chart

- Dot chart: Example 1



R Graph Basic 2: dot chart

- Dot chart: Example 2
  - ✓ "WorldPhones" dataset: The number of telephones in various regions of the world (in thousands).

	N.Amer	Europe	Asia	S.Amer	Oceania	Africa	Mid.Amer
1951	45939	21574	2876	1815	1646	89	555
1956	60423	29990	4708	2568	2366	1411	733
1957	64721	32510	5230	2695	2526	1546	773
1958	68484	35218	6662	2845	2691	1663	836
1959	71799	37598	6856	3000	2868	1769	911
1960	76036	40341	8220	3145	3054	1905	1008
1961	79831	43173	9053	3338	3224	2005	1076

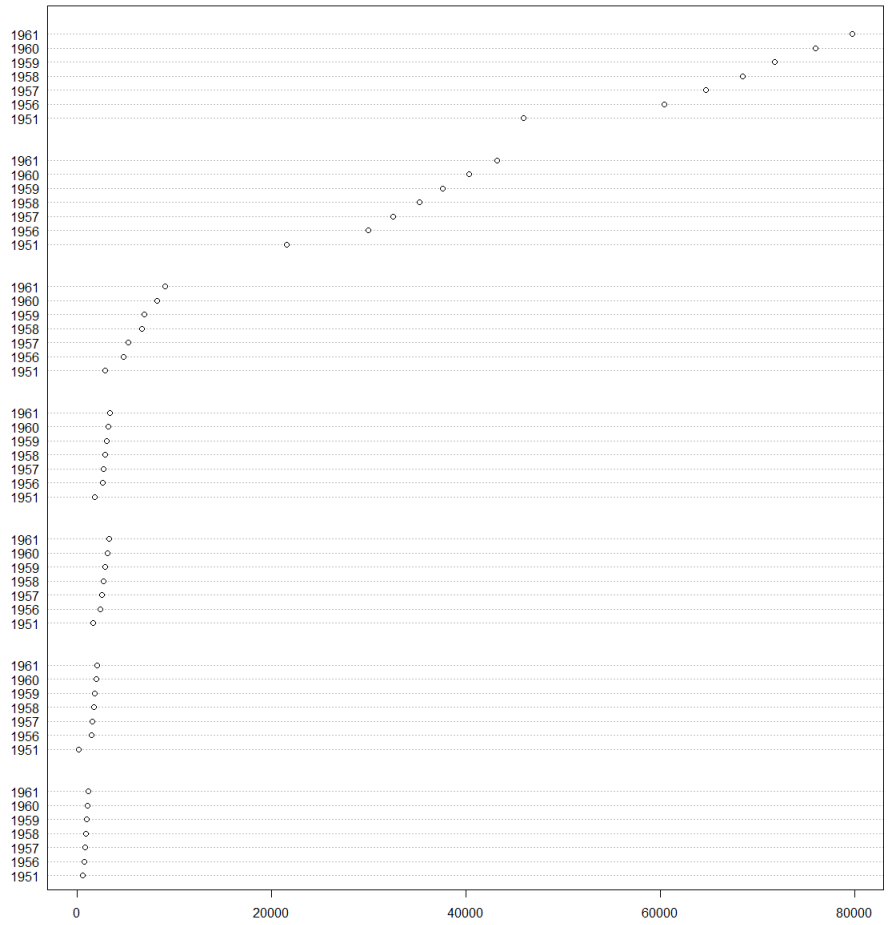
R Graph Basic 2: dot chart

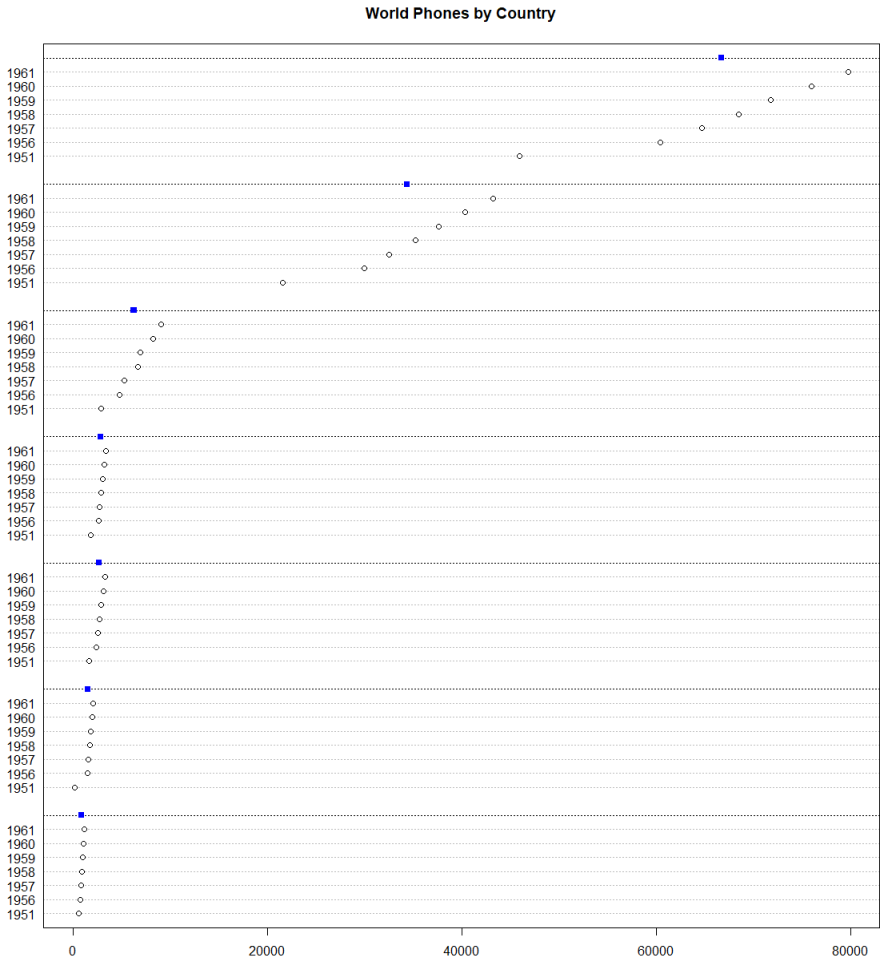
- Dot chart: Example 2

```
# dotplot for matrix ----
View(WorldPhones)
str(WorldPhones)
# worldphones is a matrix - not a dataframe
dotchart(WorldPhones) # works, but it's messy
dotchart(WorldPhones, gcolor = "Blue", cex = 1,
         gdata = colMeans(WorldPhones), gpch = 15,
         main = "World Phones by Country")
```

R Graph Basic 2: dot chart

- Dot chart: Example 2





R Graph Basic 3: Histogram

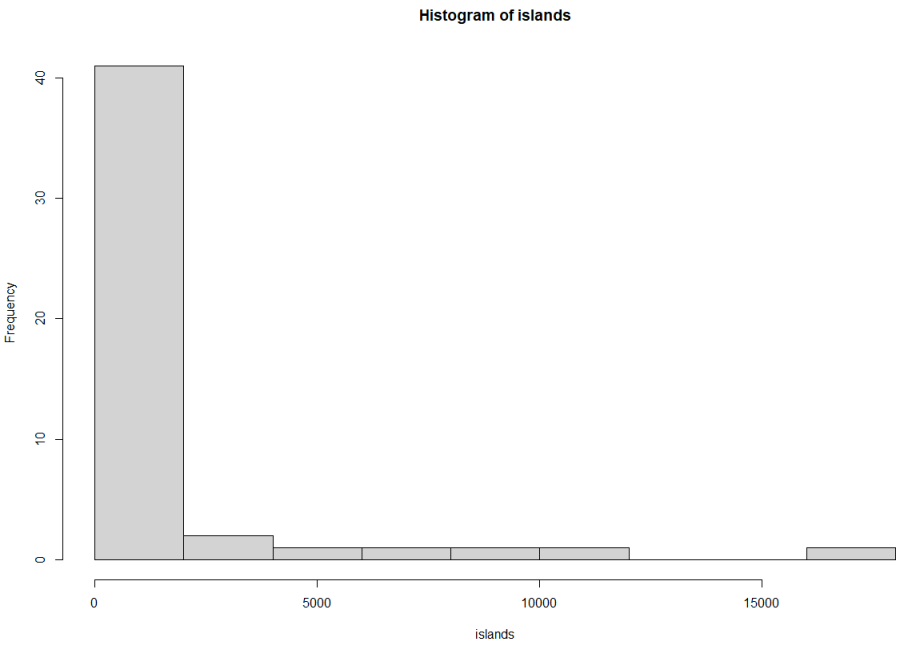
- Histogram: Example 1
  - ✓ "island" dataset: The areas in thousands of square miles of the landmasses which exceed 10,000 square miles.

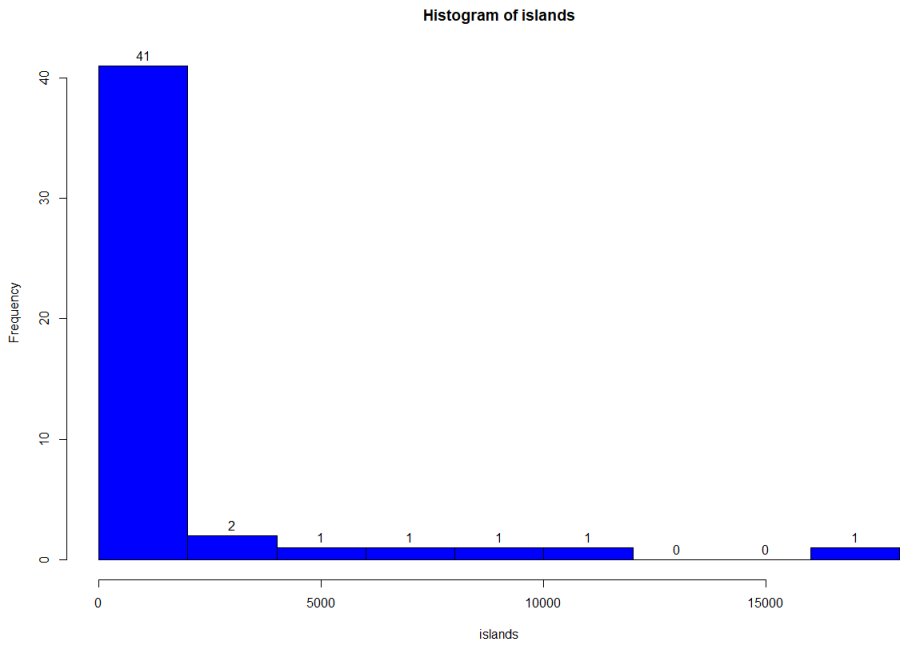
	V1
Africa	11506
Antarctica	5500
Asia	16988
Australia	2968
Axel Heiberg	16
Baffin	184
Banks	23
Borneo	280
Britain	84
Celebes	73
Celon	25
Cuba	43
Devon	21
Ellesmere	82
Europe	3745
Greenland	840
Hainan	13
Hispaniola	30
Hokkaido	30
Honshu	89
Iceland	40
Ireland	33
Java	49
Kyushu	14
Luzon	42
Madagascar	227
Melville	16
Mindanao	36
Moluccas	29
New Britain	15
New Guinea	306
New Zealand (N)	44
New Zealand (S)	58
Newfoundland	43
North America	9390
Novaya Zemlya	32
Prince of Wales	13
Sakhalin	29
South America	6795
Southampton	16
Spitsbergen	15
Sumatra	183
Taiwan	14
Tasmania	26
Tierra del Fuego	19
Timor	13
Vancouver	12
Victoria	82

R Graph Basic 3: Histogram

- Histogram: Example 1

```
# Basic plot 4: histogram
View(islands)
hist(islands)
hist(islands, col = "gray", labels = TRUE)
hist(sqrt(islands), breaks = 12, col = "lightblue", border = "pink")
```

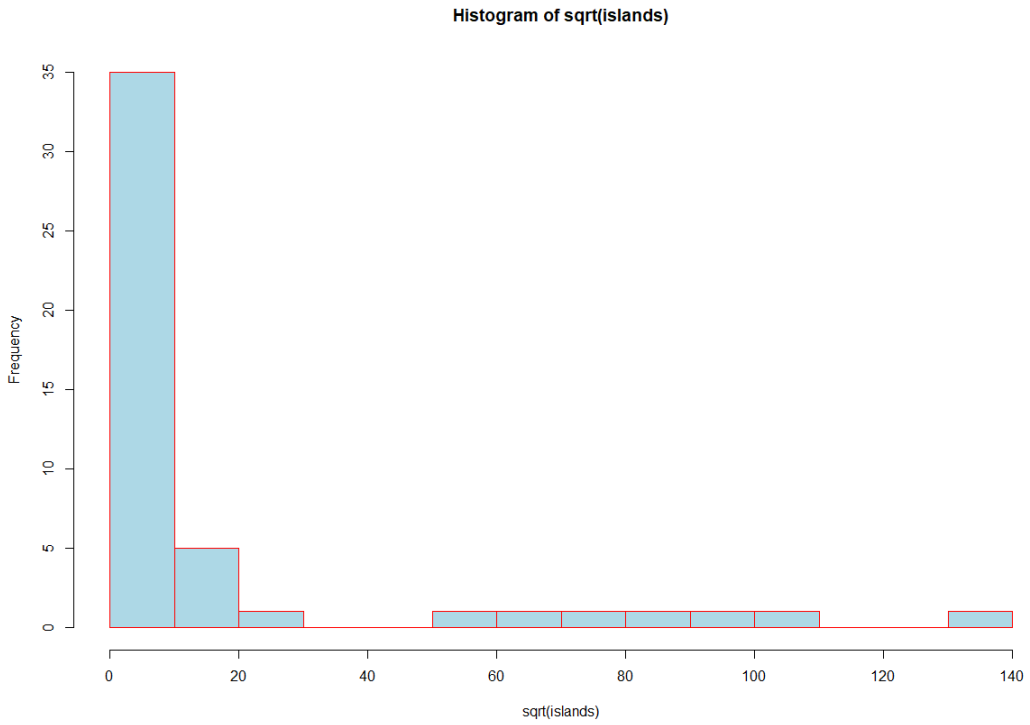




R Graph Basic 3: Histogram

- Histogram: Example 1

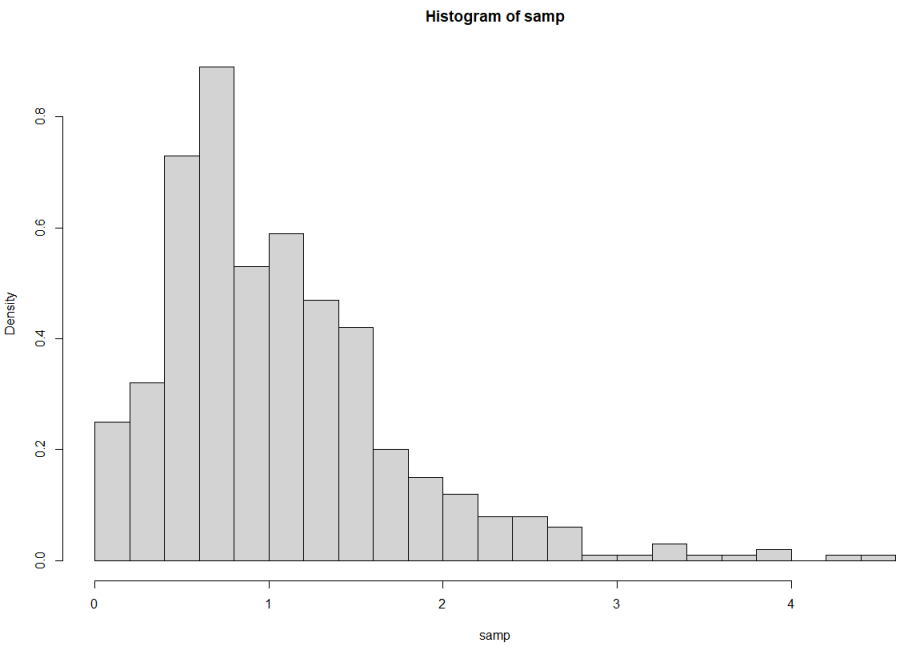
```
# Basic plot 4: histogram
View(islands)
hist(islands)
hist(islands, col = "gray", labels = TRUE)
hist(sqrt(islands), breaks = 12, col = "lightblue", border = "red")
```

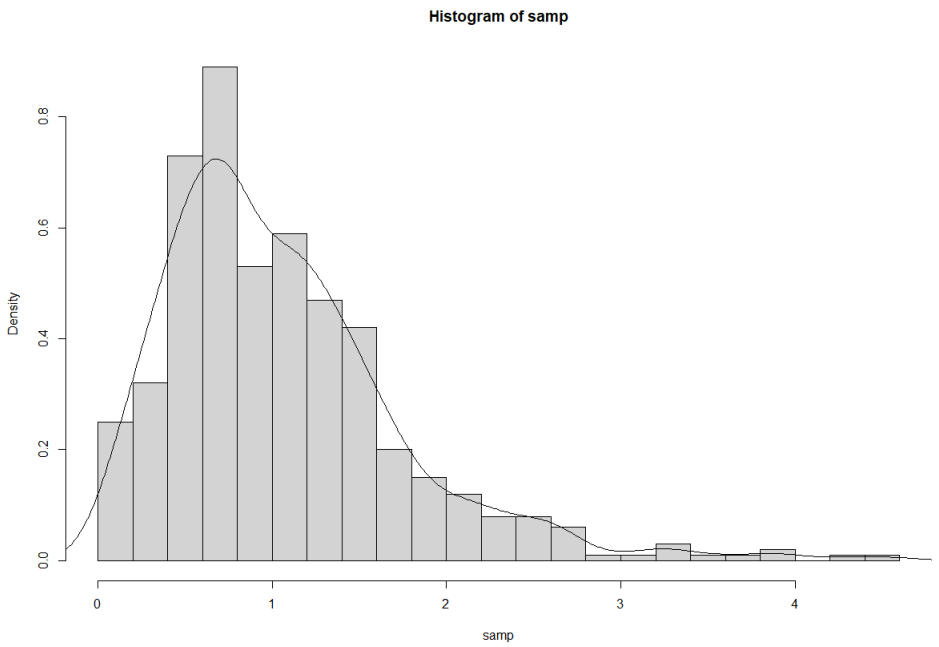


R Graph Basic 3: Histogram

- Histogram: Example 2

```
# Histogram with the estimated distribution
samp <- rgamma(500,2,2)
hist(samp, 20, prob=T)
lines(density(samp))
```



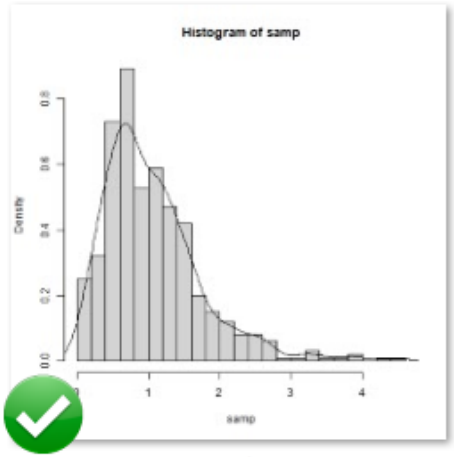


R Graph Basic 3: Histogram

- Histogram: Example 3

```
# Save the plot as a png format
png("Hist_dist.png")
hist(samp, 20, prob=T)
lines(density(samp))
dev.off()

# Save the plot as a pdf format
pdf("Hist_dist.pdf")
hist(samp, 20, prob=T)
lines(density(samp))
dev.off()
```

Hist_dist



Hist_dist


ANY
questions?