Statistical Computing with R: Masters in Data Science 503 (S12 & 13) Second Batch, SMS, TU, 2023

Shital Bhandary

Associate Professor

Statistics/Bio-statistics, Demography and Public Health Informatics

Patan Academy of Health Sciences, Lalitpur, Nepal

Faculty, Data Analysis and Decision Modeling, MBA, Pokhara University, Nepal

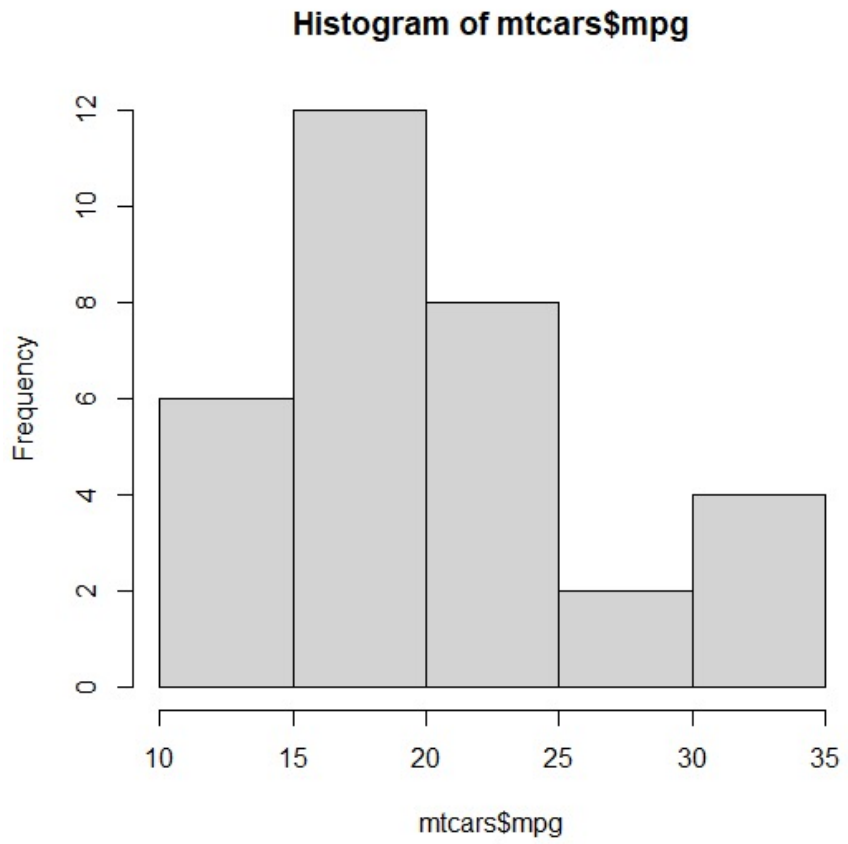
Faculty, FAIMER Fellowship in Health Professions Education, India/USA.

Review Preview

- Basic graphics/plots:
  - Additional features
- Special graph:
  - Social Network Analysis (next class)

Graph from ”data frame” variables

- Check the structure of in-built “mtcars” data
- Histogram of “mpg” variable
- How to change title?
- How to change x-axis label?



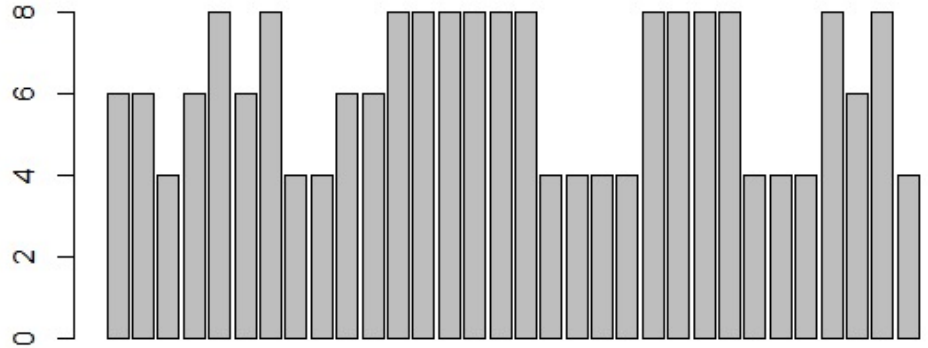
How to get bar diagram of a categorical variable from data frame?

#Define as data frame, if required

- df <- as.data.frame(mtcars)

#Bar plot of cylinder data

- barplot(df$cyl)



- This barplot shows the number of cylinders for 50 cars of the dataset

- **Do we want this?**

How to get bar diagram of a categorical variable from data frame?

#Define cyl as factor variable

- f.cyl <- as.factor(df$cyl)

#Bar plot of cylinder data

- barplot(f.cyl)

- Did you get the barplot?

- **Why?**

- Error in barplot.default(f.cyl) : 'height' must be a vector or a matrix

- What to do now?

How to get bar diagram from data frame?

First we need frequencies of cars with 4, 6 and 8 cylinders

- table(df$cyl)

#Bar plot of freq. of cylinder data

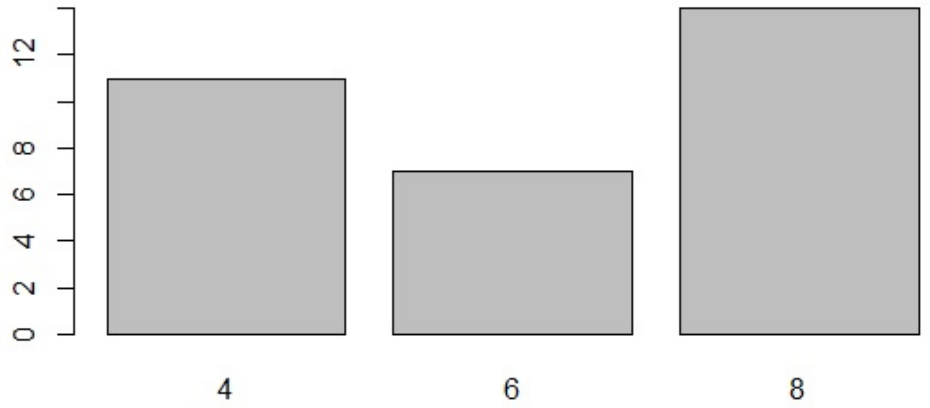
- barplot(table(df$cyl))

#We can assign this as object

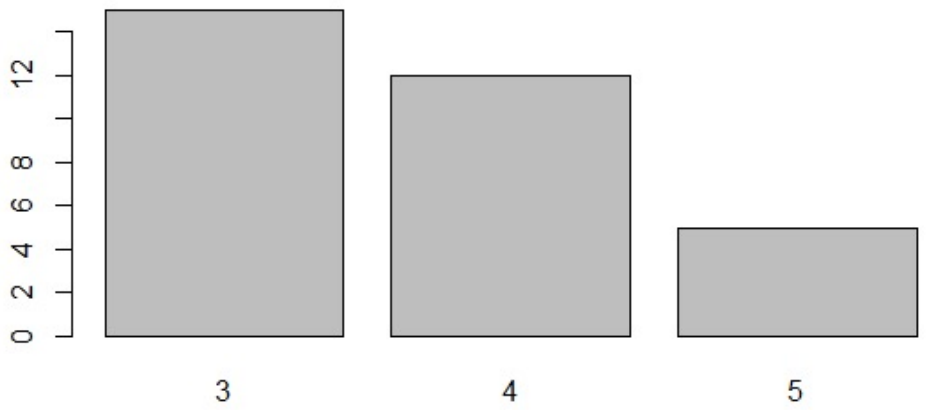
- bpd <- table(df$cyl)

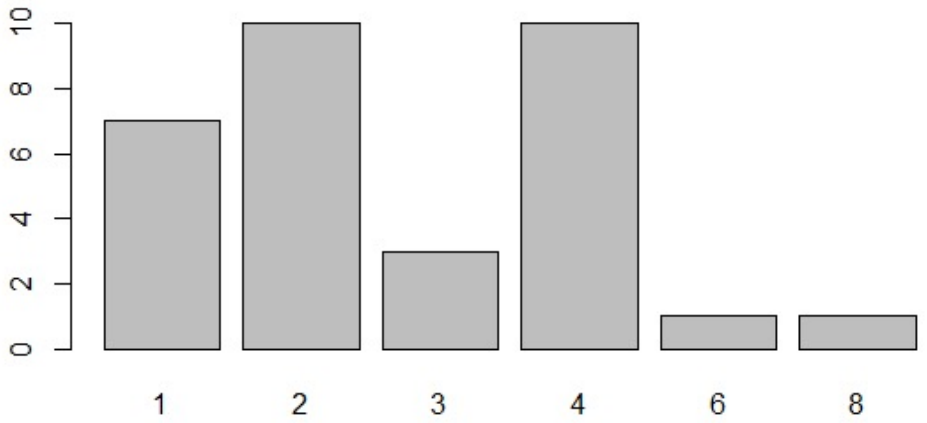
#Get the barplot

- barplot(bpd)



We can get the barplot of “gear” and “carb” too as they are factors (categorical variables)





Class work: How to get barplot of “mpg” variable? mpg: miles per gallon (continuous variable)

#MPG – class interval and its width?

- range(df$mpg)
- R = 33.9 - 10.4 #23.5
- I = round(sqrt(R)) # 5

#We need to construct 5 classes with width of 5 (10, 15, 20, 25, 30)

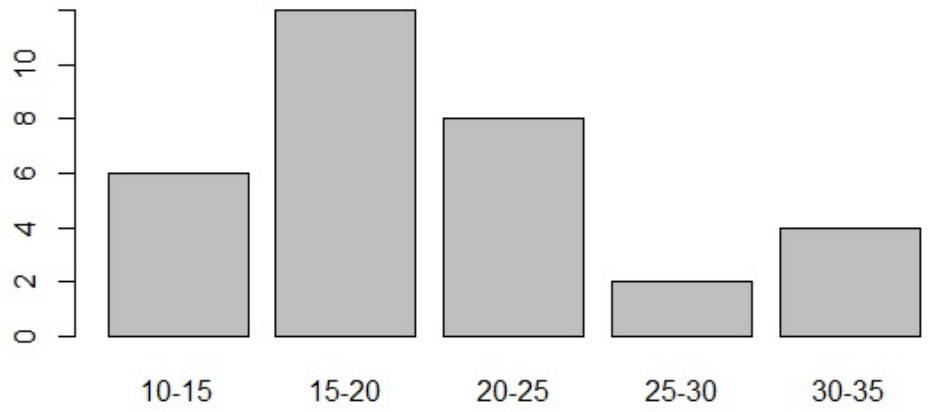
#We need to define the breaks

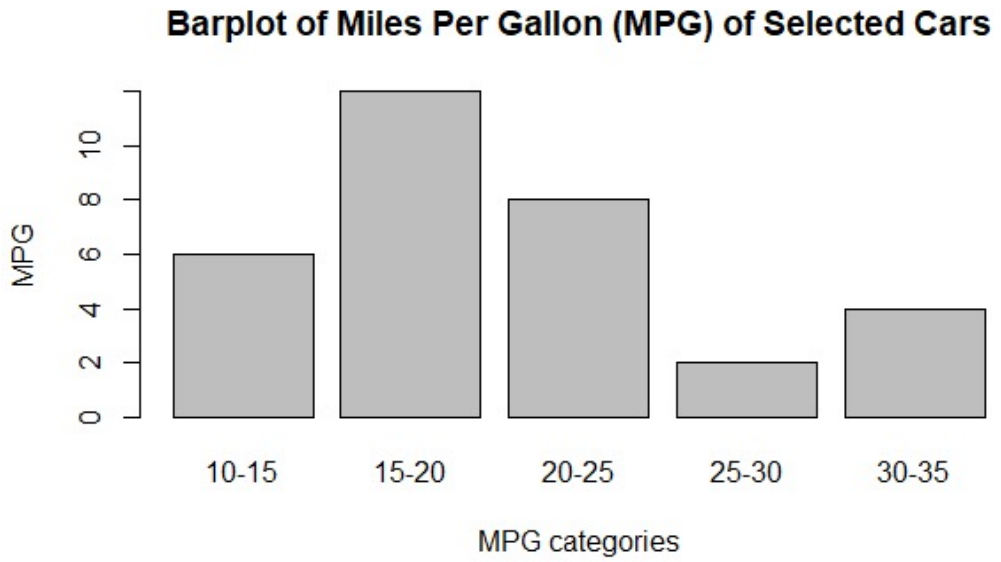
breaks = c(10, 15, 20, 25, 30, 35) or

breaks = seq(10, 35, by=5)

- mpg.bin <- cut(df$mpg, breaks, labels = c("10-15", "15-20", "20-25", "25-30", "30-35"))
- mpg.bin
- table(mpg.bin)
- barplot(table(mpg.bin))
- hist(df$mpg) #Different?

Outputs:

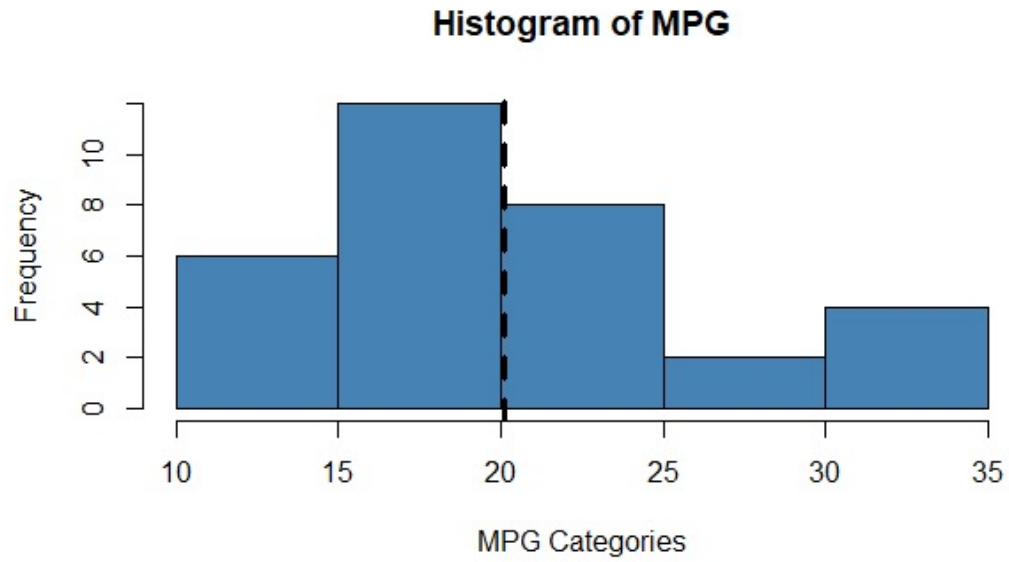

0
2
4
6
8
10
10-15
15-20
20-25
25-30
30-35


Barplot of Miles Per Gallon (MPG) of Selected Cars
MPG
0
2
4
6
8
10
10-15
15-20
20-25
25-30
30-35
MPG categories

Histogram and abline for mean of “mpg”:

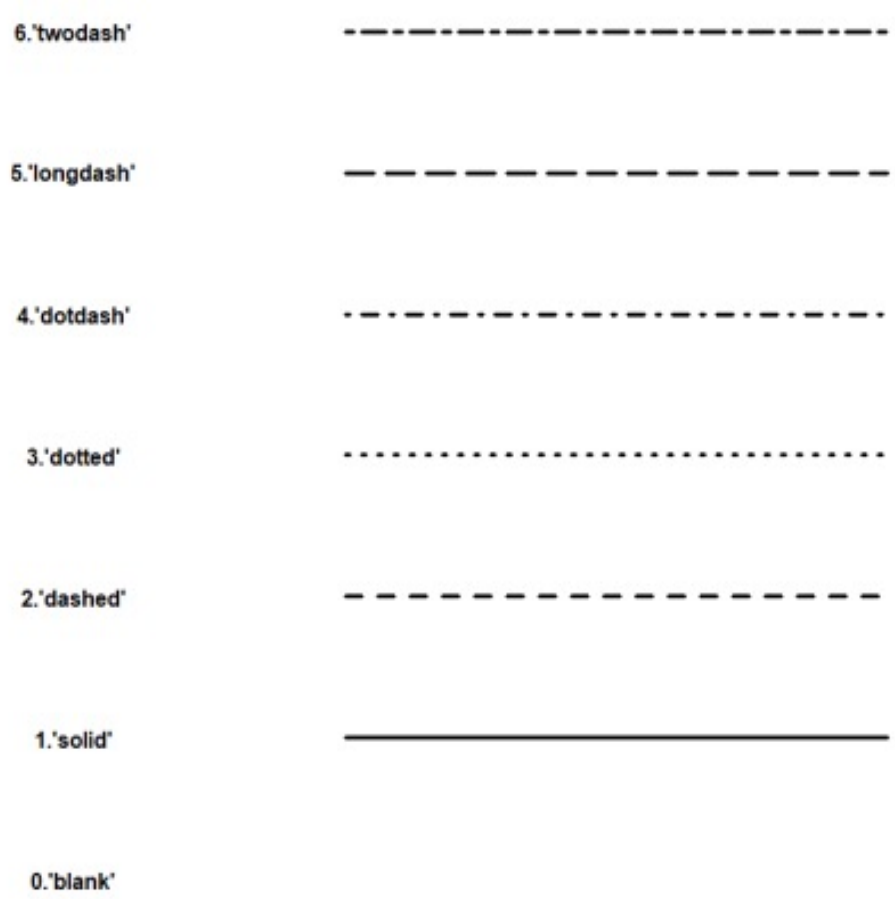
- hist(df$mpg, col = "steelblue", main = "Histogram of MPG", xlab = "MPG Categories")
- abline(**v**=mean(df$mpg), lwd=3, lty=2)

- v = vertical “abline”
- h = horizontal “abline”
- lwd = line width (3 = 3 times wide)
- lty =line types (2 = dashed line)



Line types:

- lty = 1 (solid line)
- lty = 2 (dashed line)
- lty = 3 (dotted line)
- lty = 4 (dot and dashed line)
- lty = 5 (long dash line)
- lty = 6 (two dashed line)

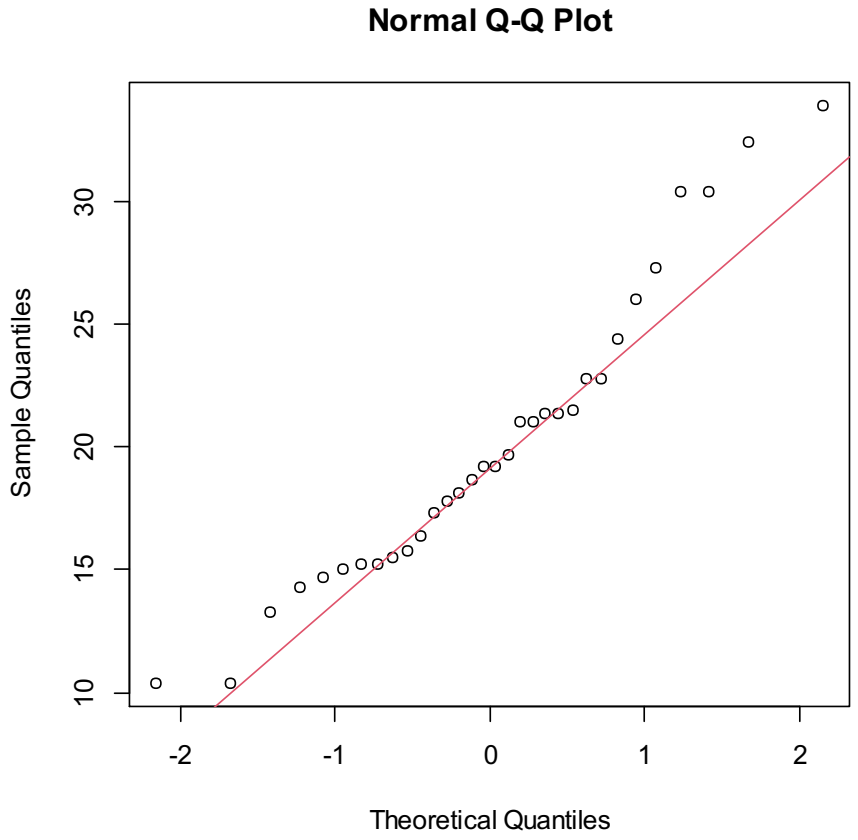


Can you justify the use of mean for “mpg” variable in the histogram?

- qqnorm(mtcars$mpg)
- > qqline(mtcars$mpg, col=2)

- Which measure of central tendency is most useful for the mpg variable?

- Which measure of central tendency can be located by histogram “graphically”?

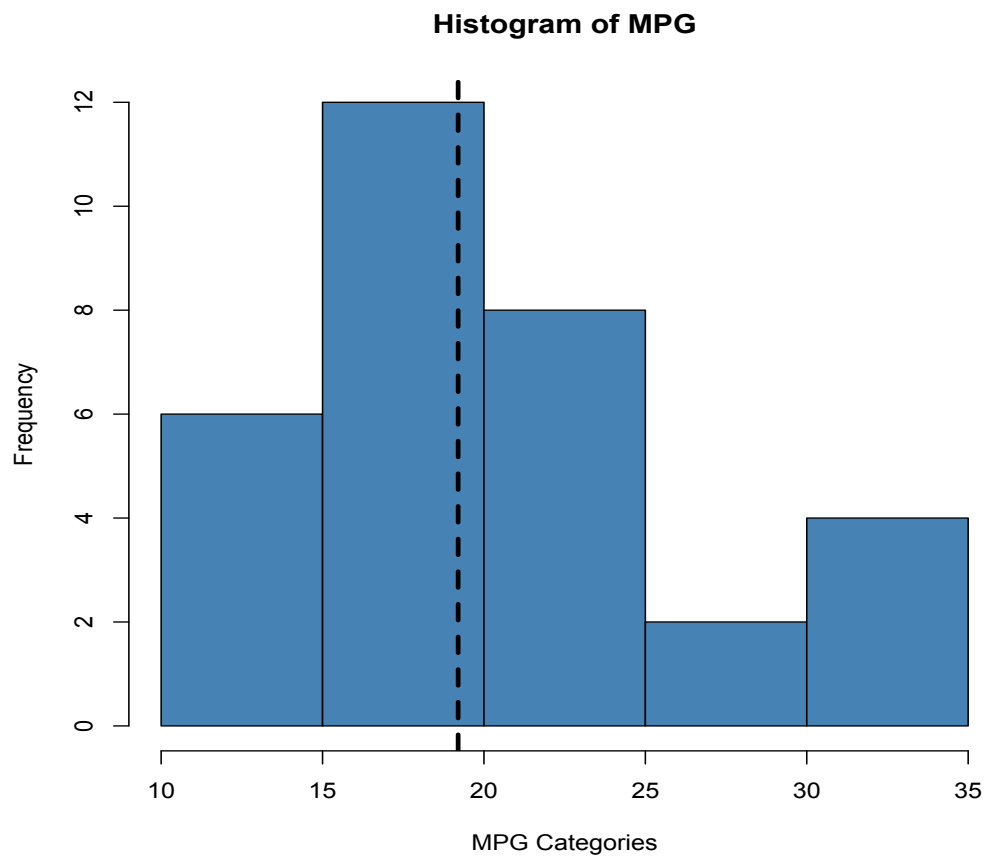


Histogram and abline of median of “mpg”:

- hist(df$mpg, col = "steelblue", main = "Histogram of MPG", xlab = "MPG Categories")
- abline(**v**=median(df$mpg), lwd=3, lty=2)

summary(df$mpg)

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
10.40	15.43	19.20	20.09	22.80	33.90



Class work: Assignment 3.1
Locate median graphically for “mpg” variable

- Create a more than cumulative frequency curve of “mpg” using cumsum function
- Create a less than cumulative frequency curve of “mpg” using cumsum function
- Draw both of them in a single plot
- The point of intersection of more than and less than cumulative frequency curve will locate the “median” value
- Draw a perpendicular to the x-axis with mpg value from this point of intersection to find the median
- Compare it with the result of the “median” function of R!

ChatGPT: how to locate median graphically with ogives in r? (Working!)

- data <- c(10, 15, 20, 25, 30, 35, 40, 45, 50)
- cumulative_freq <- cumsum(table(data))
- plot(cumulative_freq, type = "s", main = "Ogive of Data")
- total_freq <- length(data)
- median_freq <- total_freq / 2
- **Correct:** median_freq2 <- (median_freq + 1)/2 **(Why?)**
- median <- approx(cumulative_freq, 1:length(cumulative_freq), xout = median_freq)$y
- median
- abline(h = median, lty = 2, col = "red")

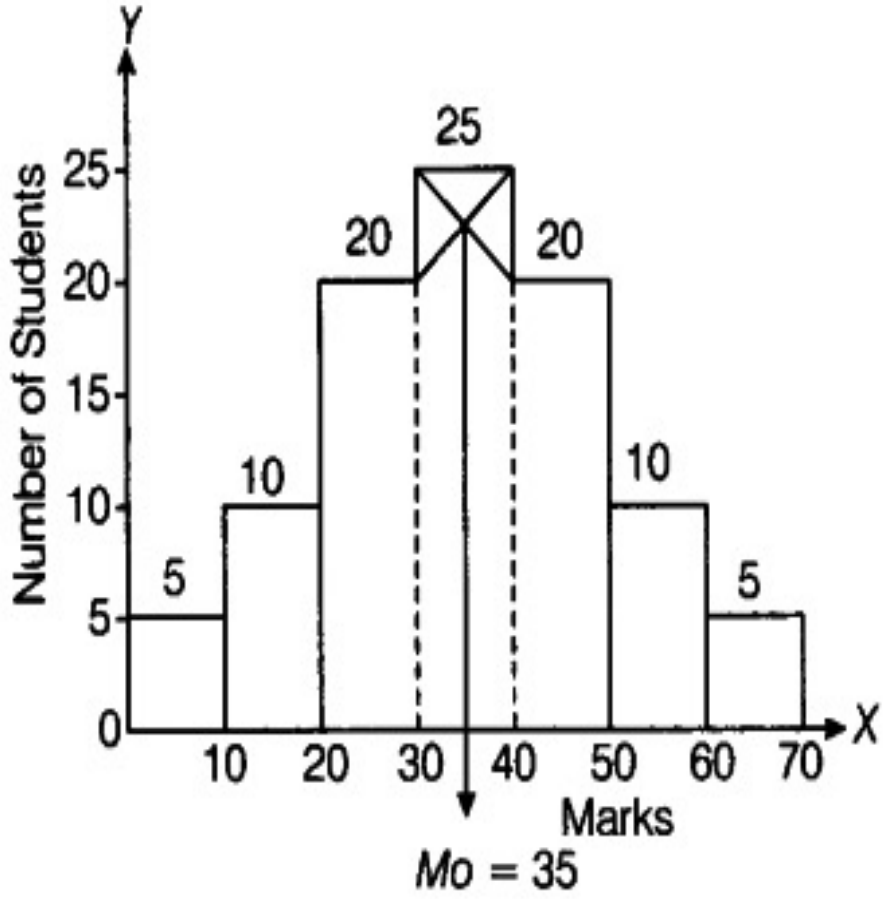
- **But: Are you happy?**
- **Check with median(data) in R!**

ChatGPT: how to locate median graphically with ogives in r? (Good but Not working!)

- # Load the plotrix package
- library(plotrix)
- # Create a vector of data for the first ogive
- data1 <- c(2, 3, 4, 5, 6, 7, 8, 9, 10, 11)
- # Create a vector of data for the second ogive
- data2 <- c(4, 5, 6, 7, 8, 9, 10, 11, 12, 13)
- # Calculate the cumulative frequencies for each data vector
- cumfreq1 <- cumsum(table(data1))
- cumfreq2 <- cumsum(table(data2))
- # Plot the first ogive
- **ogive**(cumfreq1, col="blue", xlab="Data Values", ylab="Cumulative Frequency", main="Two Ogives")
- # Add the second ogive to the plot
- **ogive**(cumfreq2, col="red", add=TRUE)
- # Add a legend to the plot
- legend("topright", legend=c("Ogive 1", "Ogive 2"), col=c("blue", "red"), lty=1)

Class work: Assignment 3.2

Locate the mode in the histogram graphically!



- Now locate mode of “mpg” variable with histogram in R Studio with R Script!

- Check your value with the built-in function of mode in R!
- mode(df$mpg) **??**
- table(df$mgp) **??** (highest freq**?**)
- which.max(df$mpg) **??**

ChatGPT: how to locate mode graphically in r with histogram? (Not exact but Working!)

Generate some example data

- data <- rnorm(1000, mean = 10, sd = 2)

Create a histogram of the data

- hist(data, **prob=T**)

Add a density line to the histogram

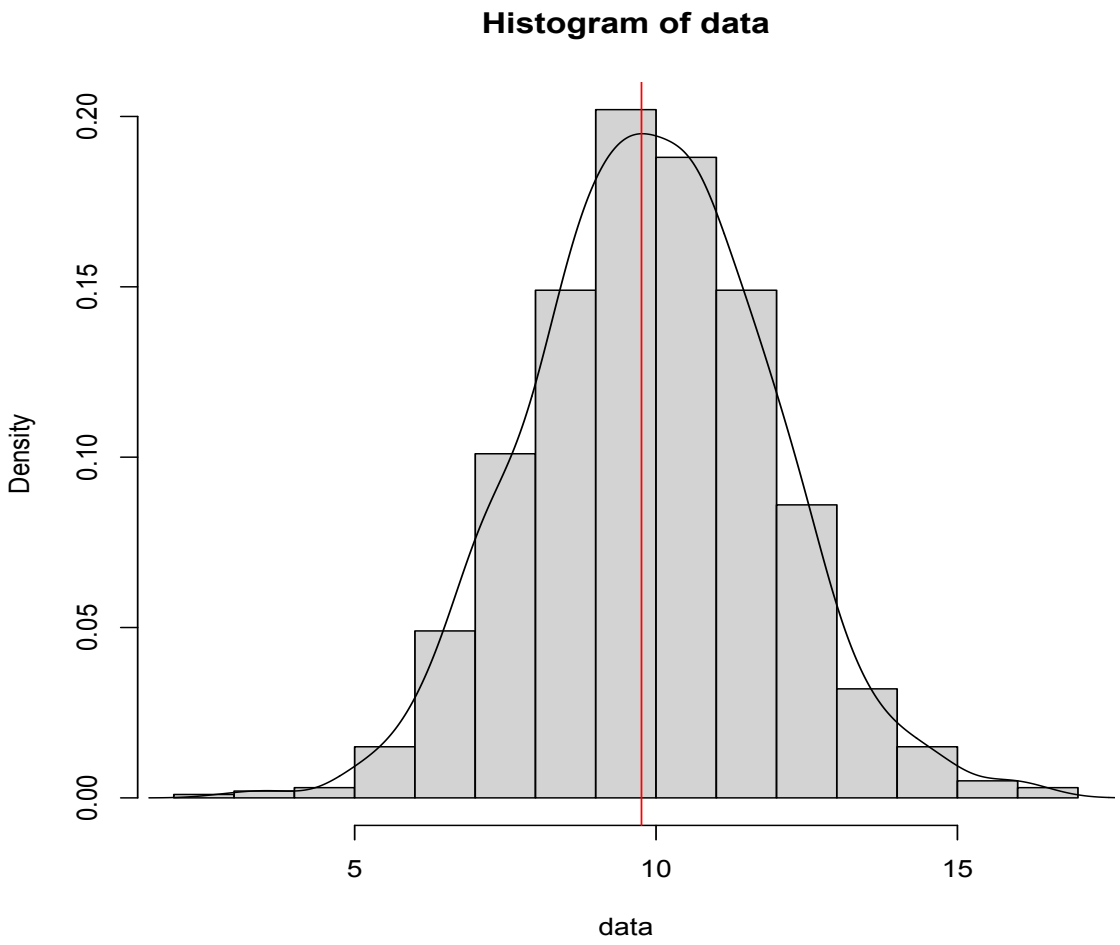
- lines(density(data))

Locate the mode

- density_values <- density(data)$y
- max_density <- max(density_values)
- mode <- density(data)$x[which.max(density_values)]

Add a vertical line at the mode location

- abline(v = mode, col = "red")



How to get mode of a variable with bi-model or multi-model distribution like “mpg”?

x	freq
10.4	**2**
13.3	1
14.3	1
14.7	1
15.0	1
15.2	**2**
15.5	1
15.8	1
16.4	1
17.3	1
17.8	1
18.1	1
18.7	1

x	freq
19.2	**2**
19.7	1
21.0	**2**
21.4	**2**
21.5	1
22.8	**2**
24.4	1
26.0	1
27.3	1
30.4	**2**
32.4	1
33.9	1

Mode = 3 Median – 2 Mean

mode <- 3*median(df$mpg) - 2 *mean(df$mpg)

mode
[1] 17.41875 **(How to interpret?)**

Now show this value as “mode” in the histogram!

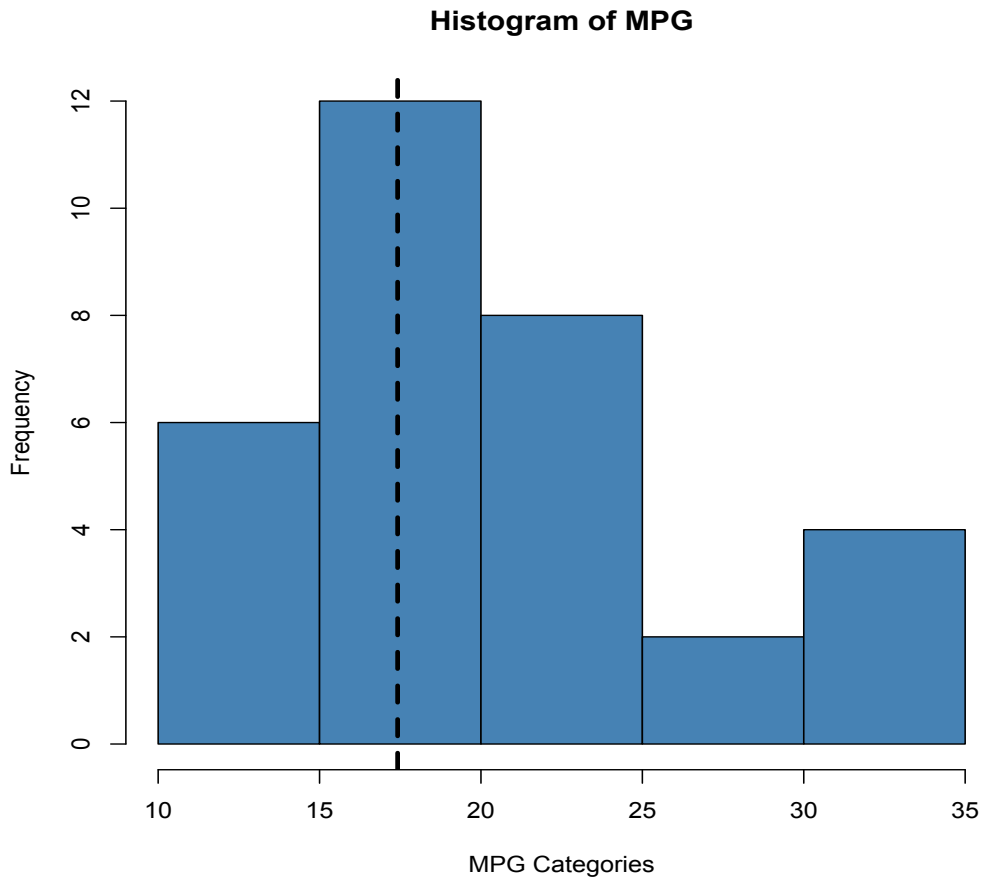
Histogram and abline of mode of “mpg”:

- hist(df$mpg, col = "steelblue", main = "Histogram of MPG", xlab = "MPG Categories")
- abline(**v**=3*median(df$mpg)-2*mean(df$mpg), lwd=3, lty=2)

summary(df$mpg)

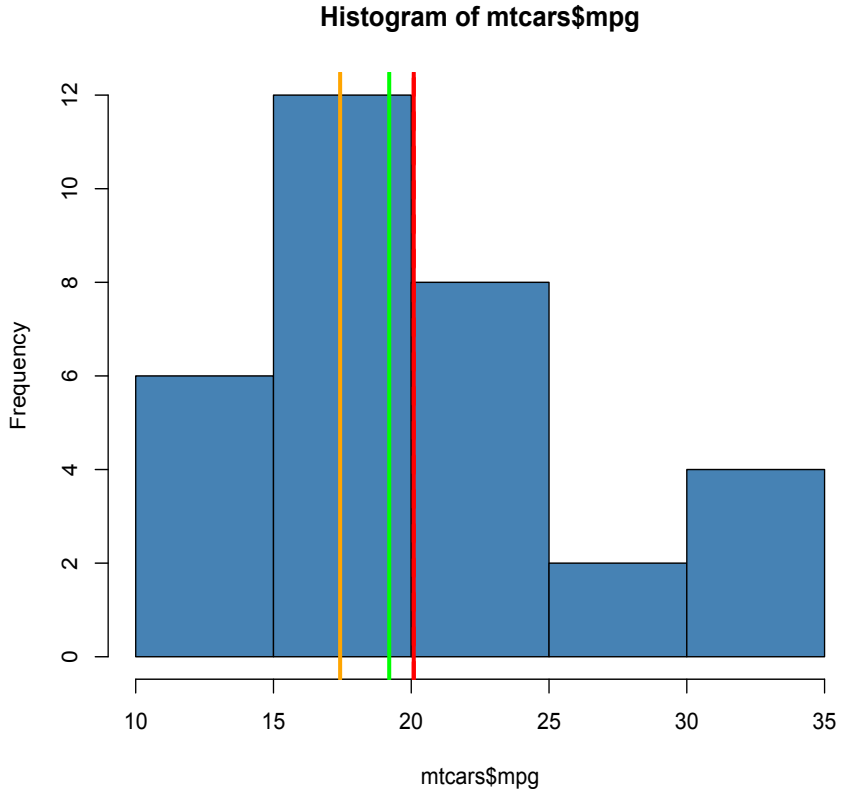
Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
10.40	15.43	19.20	20.09	22.80	33.90

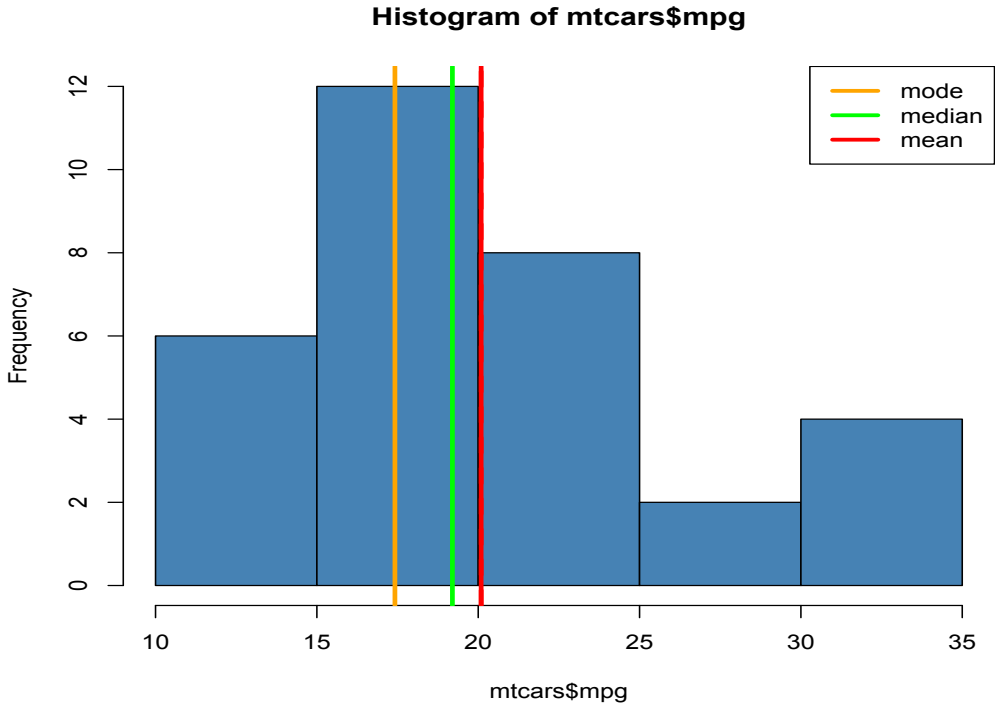
Mode = 3 * 19.20 - 2 * 20.09 = 17.42



Class work: Show the mode, median and mean of “mpg” variable in a single histogram

- With ablines of different colors
- With a legend for each color





What other measures of central tendency can we use? When to use them??

- AM
- GM
- HM
- Trimmed mean

- Which graph is used to locate the arithmetic mean “graphically”?

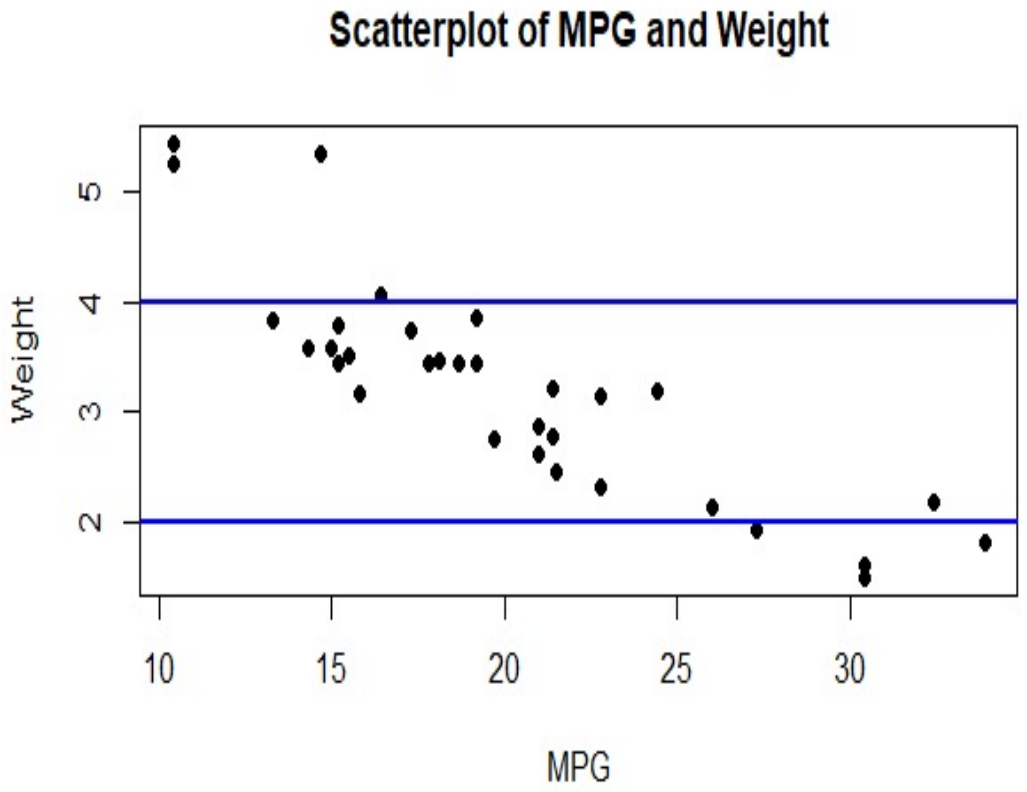
Scatterplot with horizontal “abline”:

```
#Scatterplot with abline
plot(df$mpg, df$wt, pch=16, main =
"Scatterplot of MPG and Weight",
xlab = "MPG", ylab = "Weight")
abline(h=2, col = "blue", lwd=2)
abline(h=4, col = "blue", lwd=2)
```

h = horizontal line in y-axis

lwd = line width parameter

pch = plot character



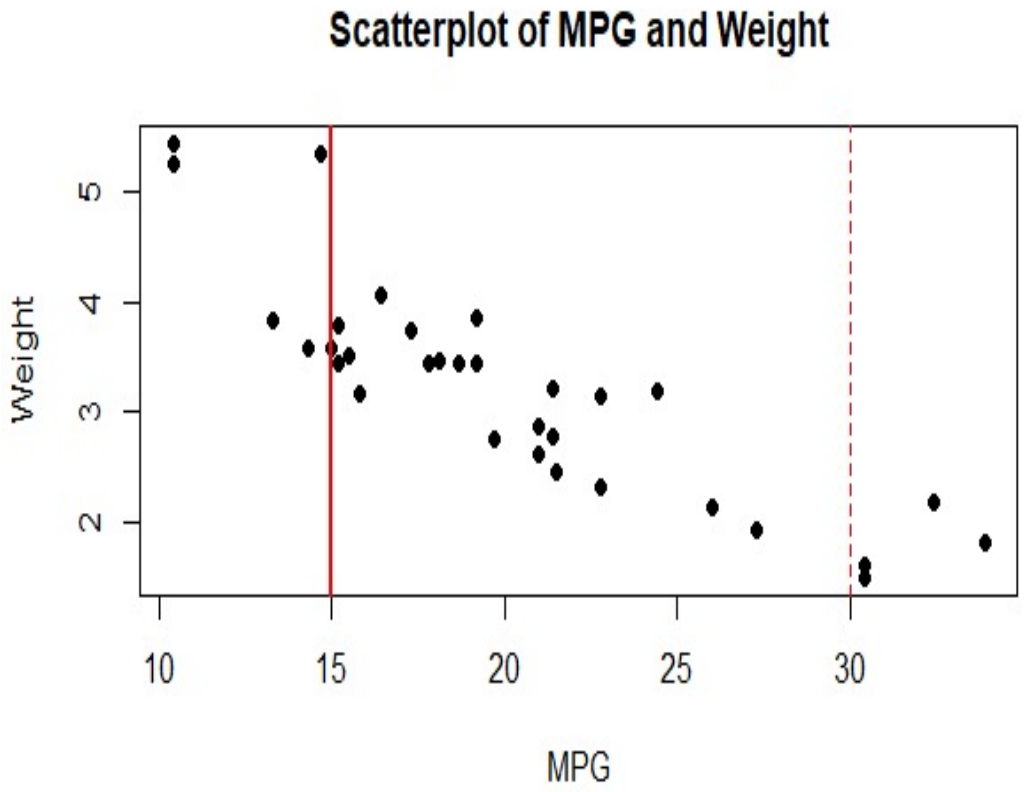
“pch” attributes:

- pch = 0,square
- pch = 1,circle
- pch = 2,triangle point up
- pch = 3,plus
- pch = 4,cross
- pch = 5,diamond
- pch = 6,triangle point down
- pch = 7,square cross
- pch = 8,star
- pch = 9,diamond plus
- pch = 10,circle plus
- pch = 11,triangles up and down
- pch = 12,square plus
- pch = 13,circle cross
- pch = 14,square and triangle down
- pch = 15, filled square
- **pch = 16, filled circle**
- pch = 17, filled triangle point-up
- pch = 18, filled diamond
- pch = 19, solid circle
- pch = 20,bullet (smaller circle)
- **pch = 21, filled circle blue**
- pch = 22, filled square blue
- pch = 23, filled diamond blue
- pch = 24, filled triangle point-up blue
- pch = 25, filled triangle point down blue

Scatterplot with vertical “abline”:

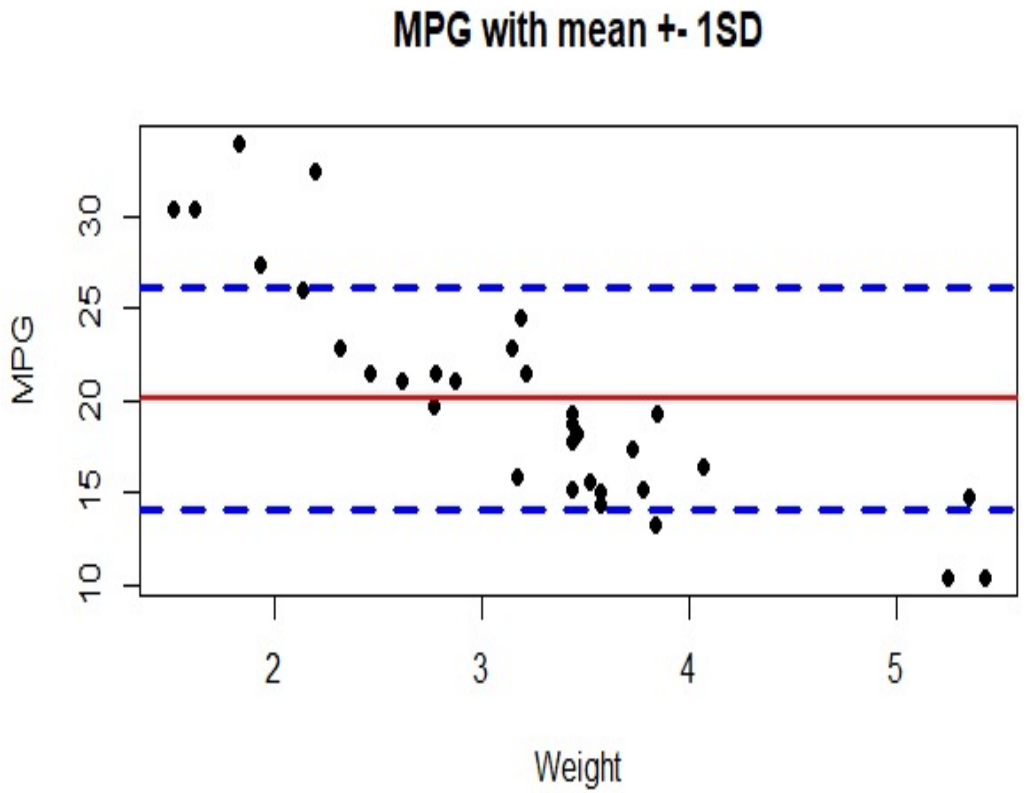
- plot(dfmpg, dfwt, pch=16, main = "Scatterplot of MPG and Weight", xlab = "MPG", ylab = "Weight")
- abline(v=15, col = "red", lwd=2)
- abline(v=30, col = "red", lty=2)

- Here, v=Vertical line at x-axis and lty = line type parameter



Scatterplot with mean ± 1*sd of y-variable: Add mean ± 2*sd to this graph and ylim if required

- plot(dfwt, dfmpg, pch=16)
- abline(h=mean(df$mpg), lwd = 2, col = "red")
- abline(h=mean(df$mpg) + 1*sd(df$mpg), col = "blue", lwd=3, lty = 2)
- abline(h=mean(df$mpg) - 1*sd(df$mpg), col = "blue", lwd=3, lty = 2)

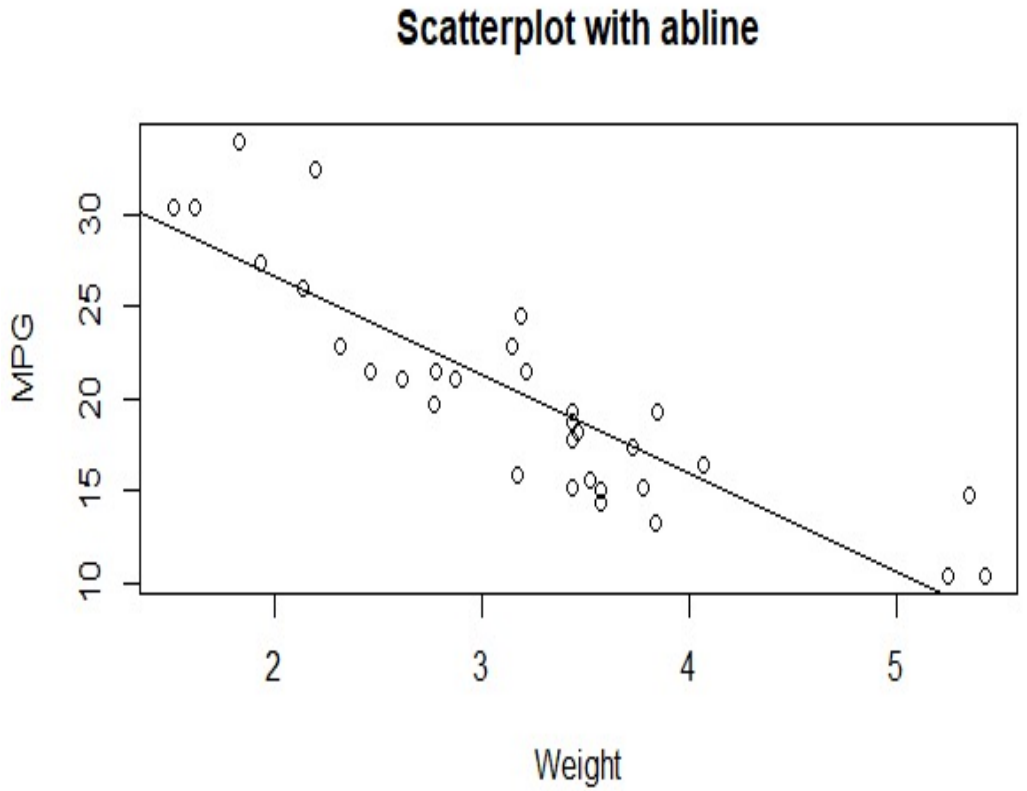


Which cars are outliers based on mean ±2*sd values?

Scatterplot with “abline” from a model:

- plot(dfwt, dfmpg, main = "Scatterplot with abline", xlab = "Weight", ylab = "MPG")
- reg_mod <- lm(df$mpg ~ df$wt)
- abline(reg_mod)

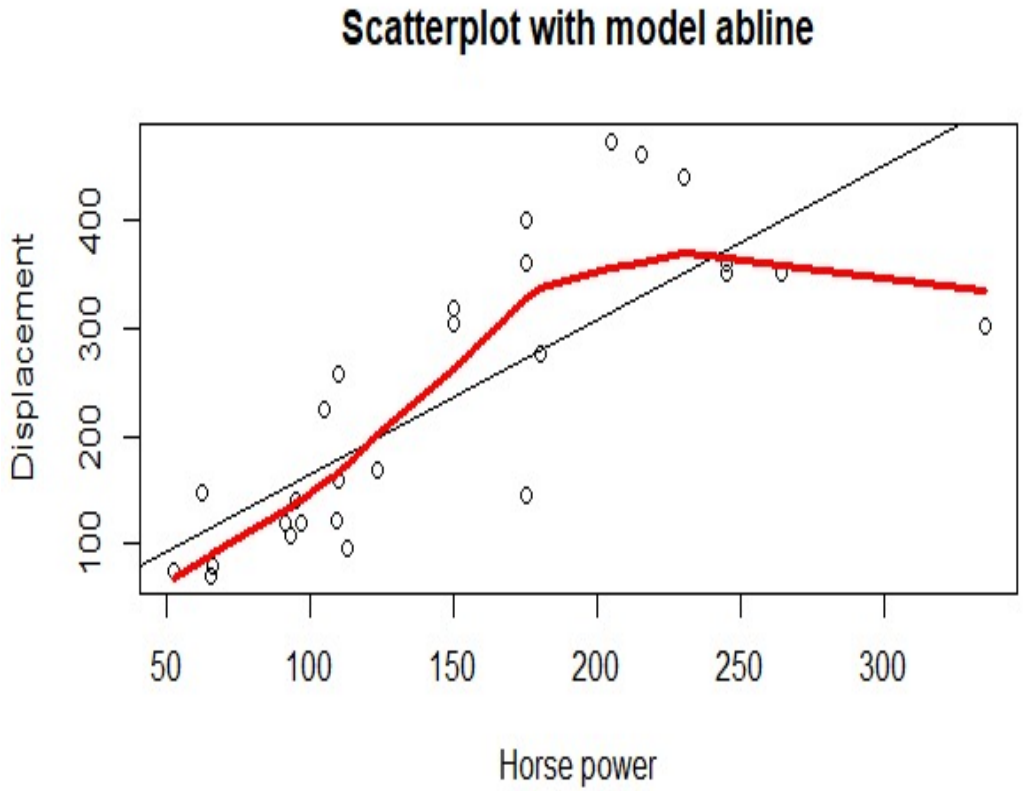
- plot(dfwt, dfmpg, main = "Scatterplot with abline", xlab = "Weight", ylab = "MPG")
- abline(lm(df$mpg ~ df$wt))



Scatterplot with “abline” and “lines” for a non-linear data:

- plot(dfhp, dfdisp, main = "Scatterplot with model abline", xlab = "Horse power", ylab = "Displacement")
- abline(lm(df$disp ~ df$hp))
- lines(lowess(dfhp, dfdisp), col = "red", lwd = 3)

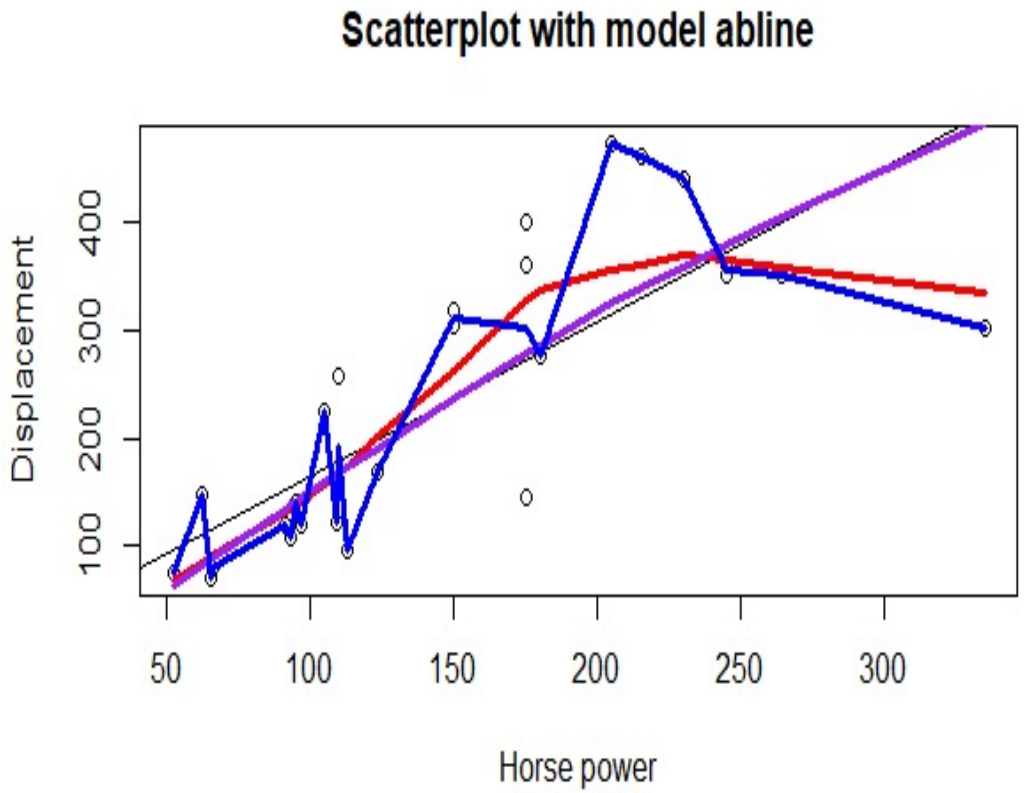
- Lowess = Locally weighted Scatterplot Smoothing



This LOWESS plot suggest that non-linear model is better than linear model for this data.

Scatterplot with “abline” and “lines” for a non-linear data:

- plot(dfhp, dfdisp, main = "Scatterplot with model abline", xlab = "Horse power", ylab = "Displacement")
- abline(lm(df$disp ~ df$hp))
- lines(lowess(dfhp, dfdisp), col = "red", lwd = 3)
- lines(lowess(dfhp, dfdisp, **f=1**), col = "purple", lwd = 3)
- lines(lowess(dfhp, dfdisp, **f=0.1**), col = "blue", lwd = 3)



The argument f gives the proportion of data points which influence the smoothening at each value. Larger values give more smoothness. The default value is f = 2/3.

Question/Queries so far?

More on plots: https://r-coder.com/plot-r/

- set.seed(1)

Generate sample data
- x <- rnorm(500)
- y <- x + rnorm(500)

Plot the data
- plot(x, y)

Equivalent
- M <- cbind(x, y)
- plot(M)

#Try
- plot(cars)

Function and arguments	Output plot
• plot(x,y)	• Scatterplot of x and y numeric vectors
• **plot(factor)**	• **Barplot of the factor**
• plot(factor, y)	• Boxplot of the numeric vector and the levels of the factor
• plot(time_series)	• Time series plot

Function and arguments	Output plot
• plot(date, y)	• Plots a date-based vector
• plot(function, lower, upper)	• Plot of the function between the lower and maximum value specified

What will happen?

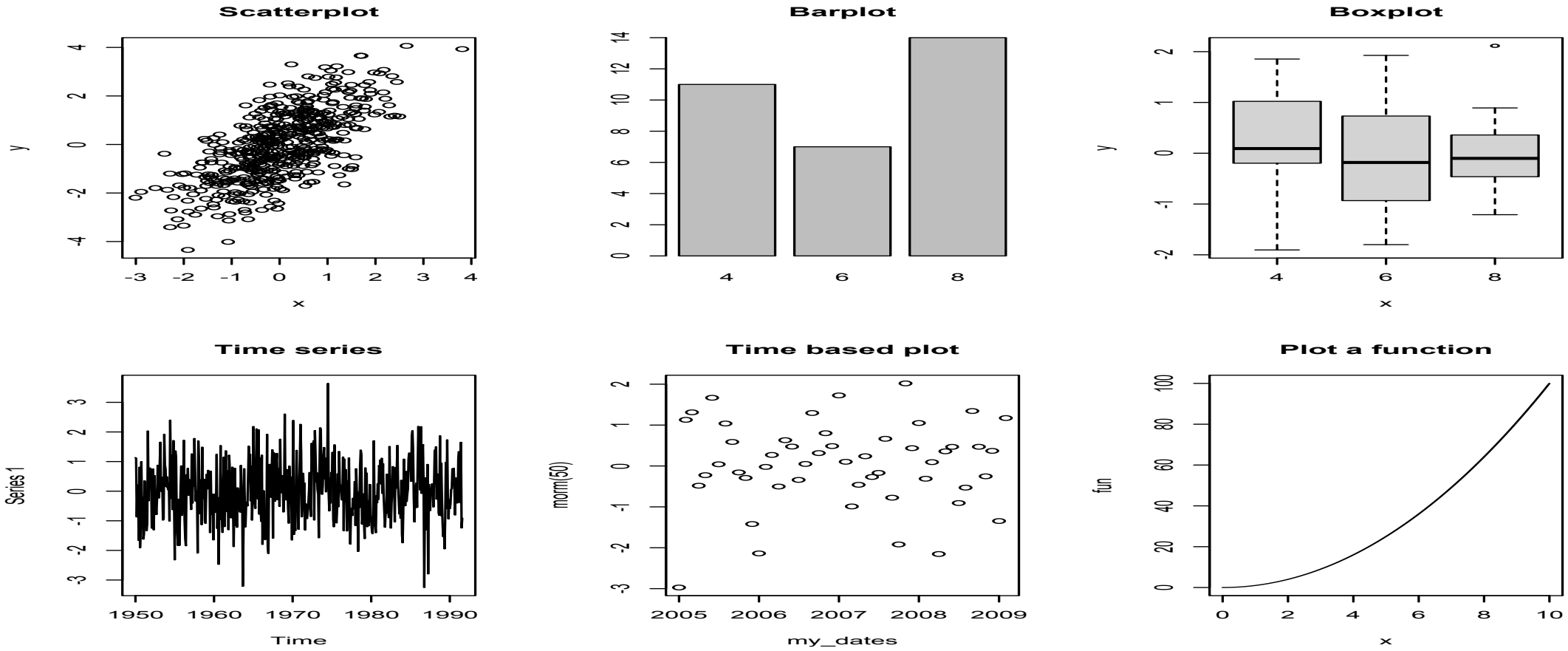
- # Data
- my_ts <- ts(matrix(rnorm(500), nrow = 500, ncol = 1), start = c(1950, 1), frequency = 12)
- my_dates <- seq(as.Date("2005/1/1"), by = "month", length = 50)
- my_factor <- factor(mtcars$cyl)
- fun <- function(x) x^2

- **par(mfrow = c(2, 3)) # 2x3 plots**
- plot(x, y, main = "Scatterplot")
- plot(my_factor, main = "Barplot")
- plot(my_factor, rnorm(32), main = "Boxplot")
- plot(my_ts, main = "Time series")
- plot(my_dates, rnorm(50), main = "Time based plot")
- plot(fun, 0, 10, main = "Plot a function")

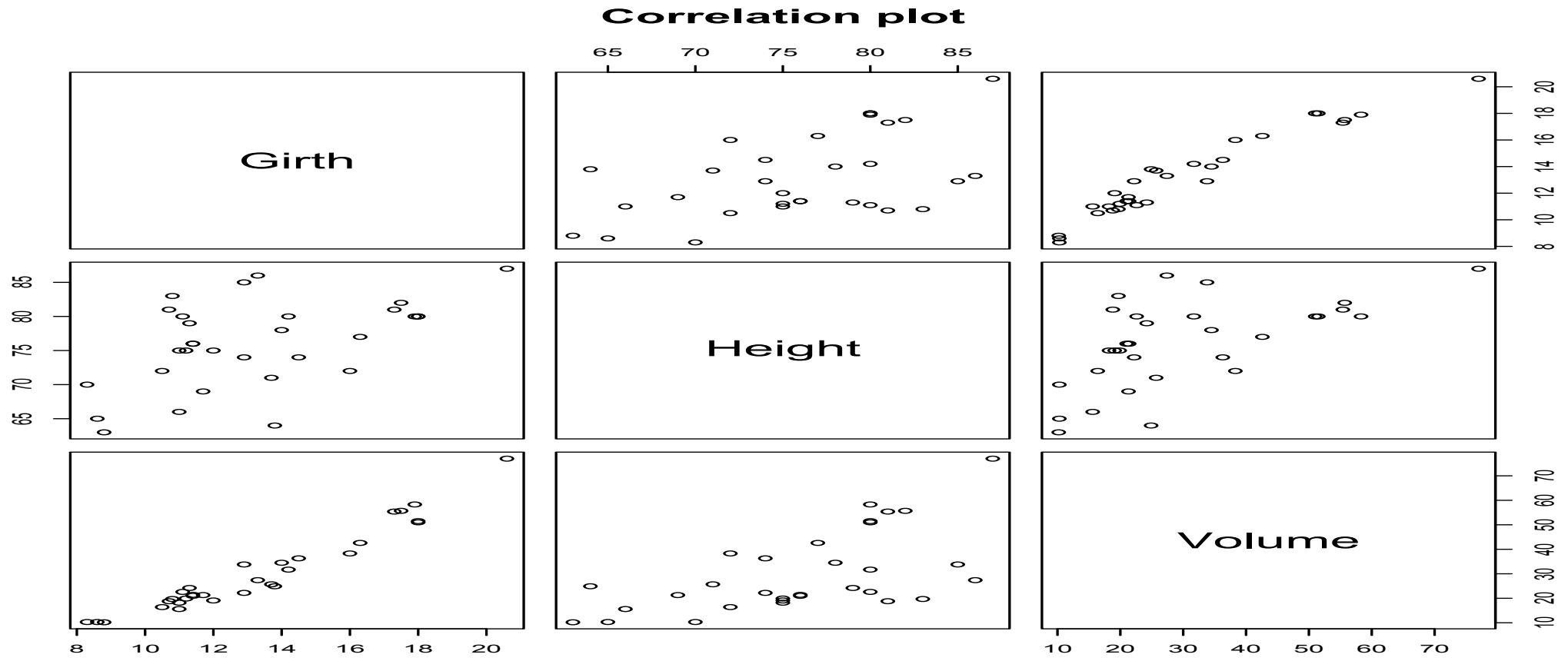
#Always run this afterwards:

- **par(mfrow = c(1, 1)) #1x1 plot**

Plots with **par(mfrow = c(2, 3))**
Six plots in a single window!



Plot after **par(mfrow = c(1, 1)) gives single plot:**
plot(trees[, 1:3], main = "Correlation matrix plot")



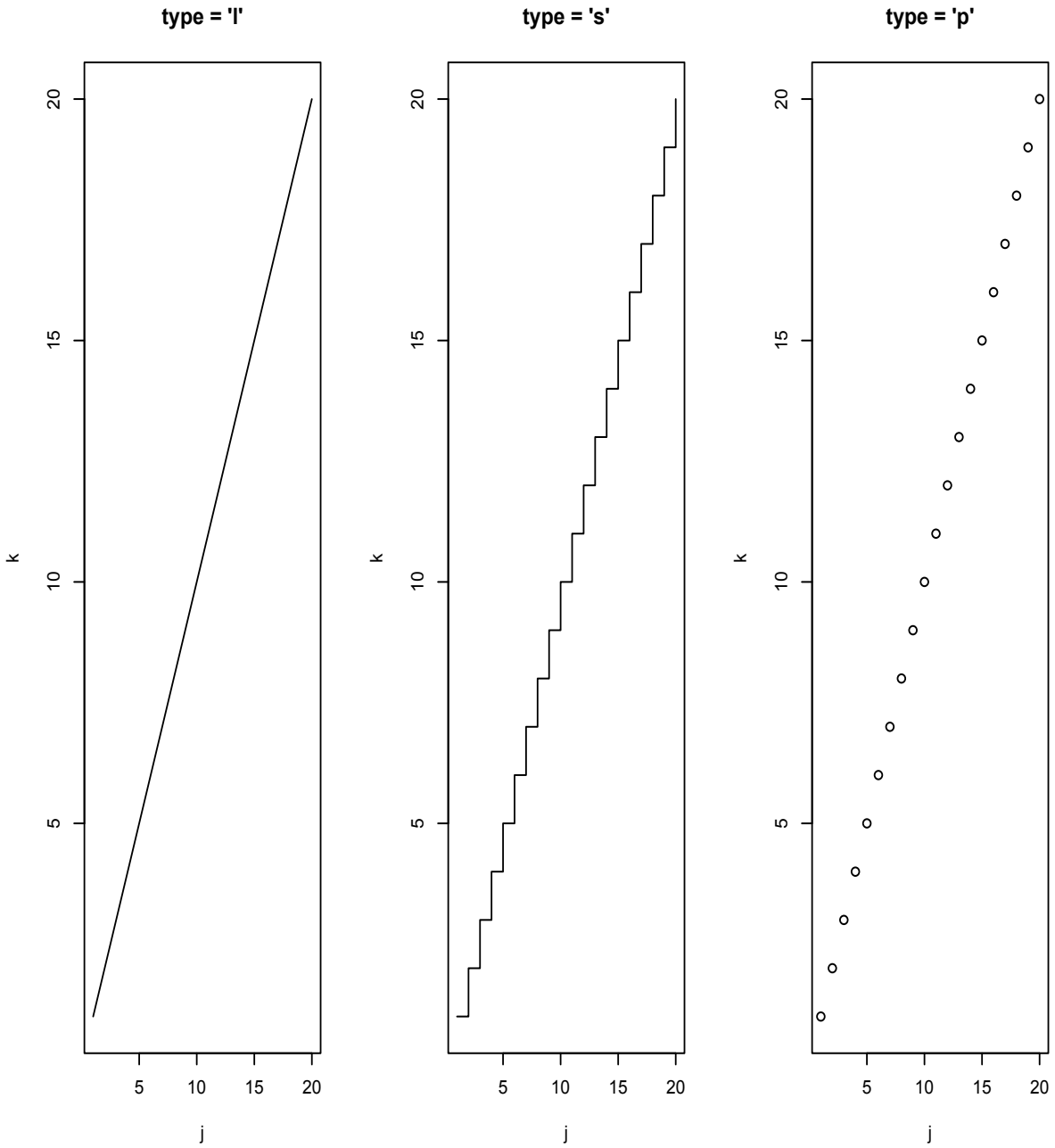
What will happen?

- j <- 1:20
- k <- j

- par(mfrow = c(1, 3))

- plot(j, k, type = "l", main = "type = 'l'")
- plot(j, k, type = "s", main = "type = 's'")
- plot(j, k, type = "p", main = "type = 'p'")

- par(mfrow = c(1, 1))



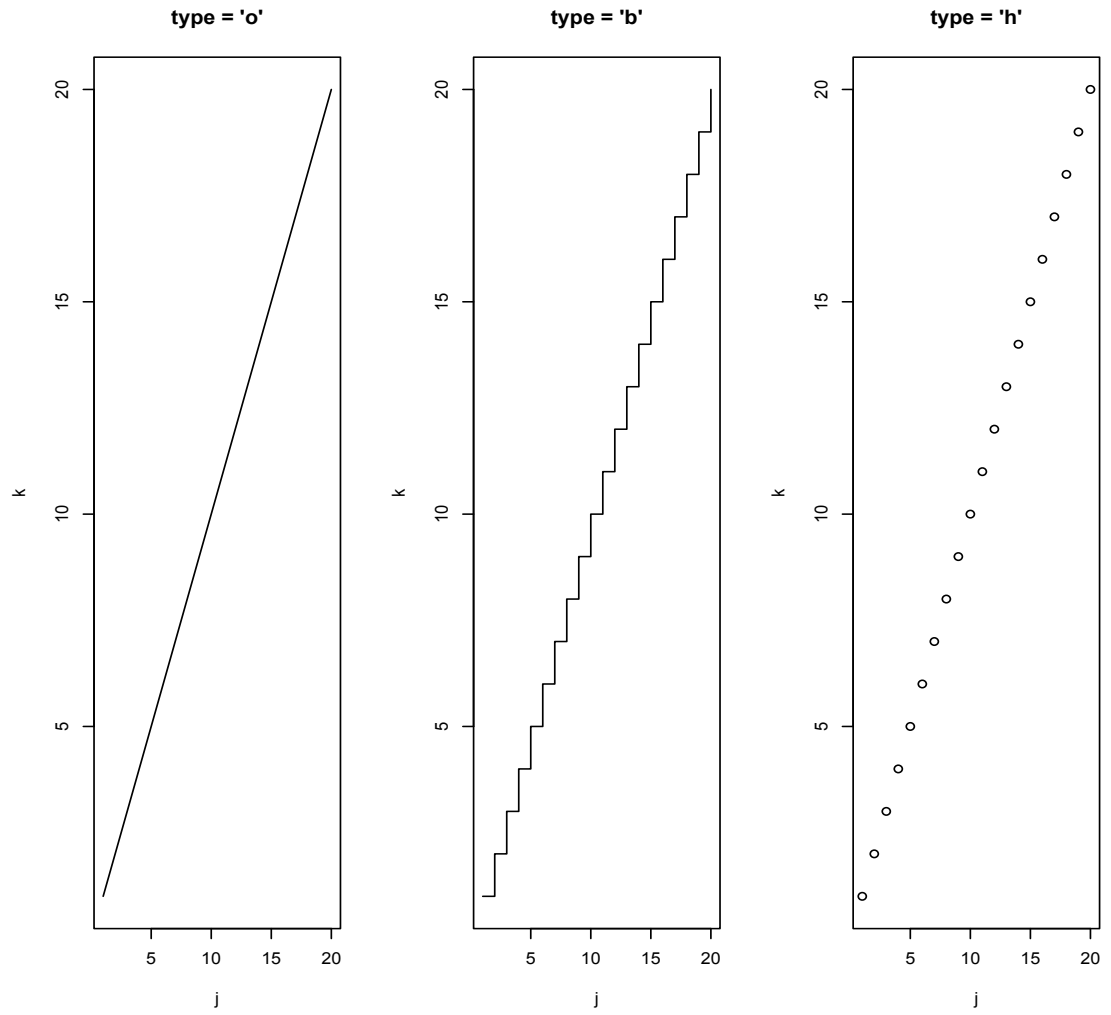
What will happen?

- j <- 1:20
- k <- j

- par(mfrow = c(1, 3))

- plot(j, k, type = "l", main = "type = 'o'")
- plot(j, k, type = "s", main = "type = 'b'")
- plot(j, k, type = "p", main = "type = 'h'")

- par(mfrow = c(1, 1))



Plot type and its description

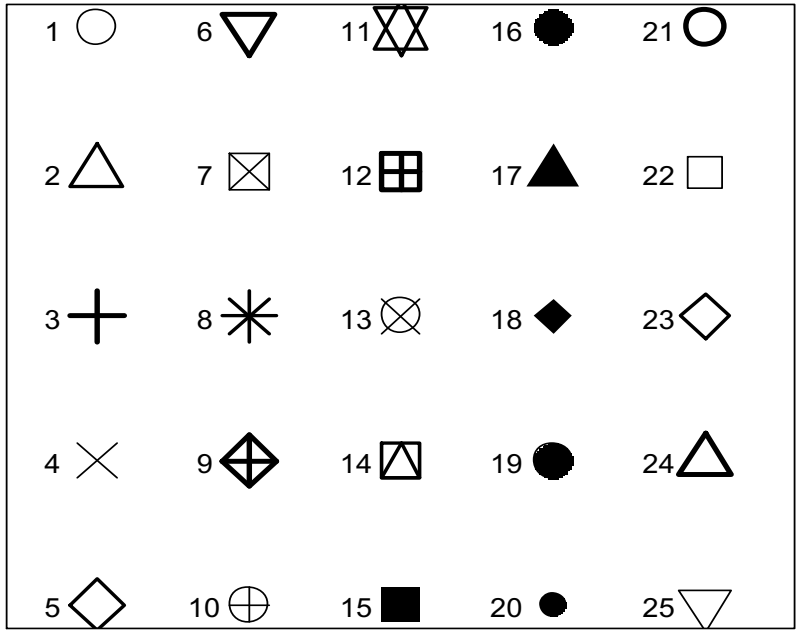
- p Points plot (default)
- l Line plot
- b Both (points and line)
- o Both (overplotted)
- s Stairs plot
- h Histogram-like plot
- n No plotting

What will happen?

- r <- c(sapply(seq(5, 25, 5), function(i) rep(i, 5)))
- t <- rep(seq(25, 5, -5), 5)

- plot(r, t, **pch = 1:25**, cex = 3, yaxt = "n", xaxt = "n", **ann** = FALSE, xlim = c(3, 27), lwd = 1:3)

- text(r - 1.5, t, 1:25)



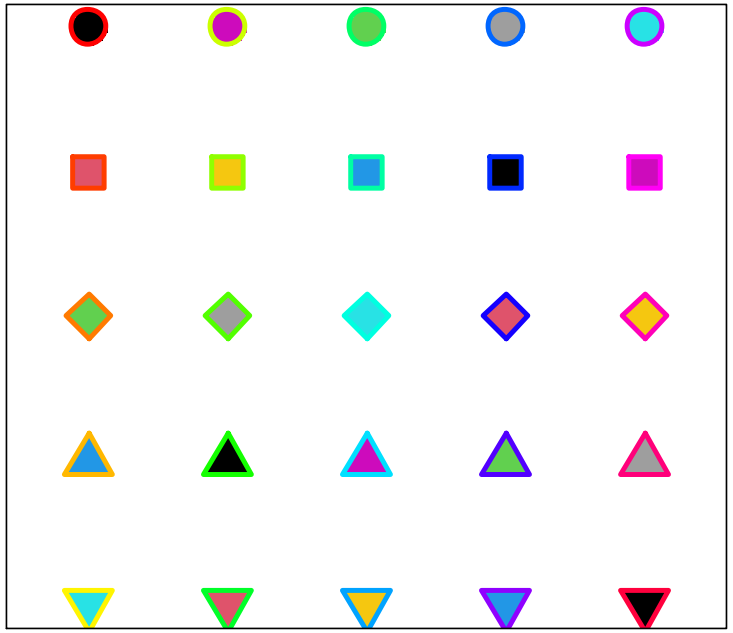
Note:

- The **pch** argument allows to modify the symbol of the points in the plot.
- The main symbols can be selected passing numbers 1 to 25 as parameters.
- You can also change the symbols size with the cex argument and the line width of the symbols (except 15 to 18) with the lwd argument.
- **ann = Annotations!**

What will happen now?

- plot(r, t, pch = 21:25, cex = 3, yaxt = "n", xaxt = "n", lwd = 3, ann = FALSE, xlim = c(3, 27), bg = 1:25, col = rainbow(25))

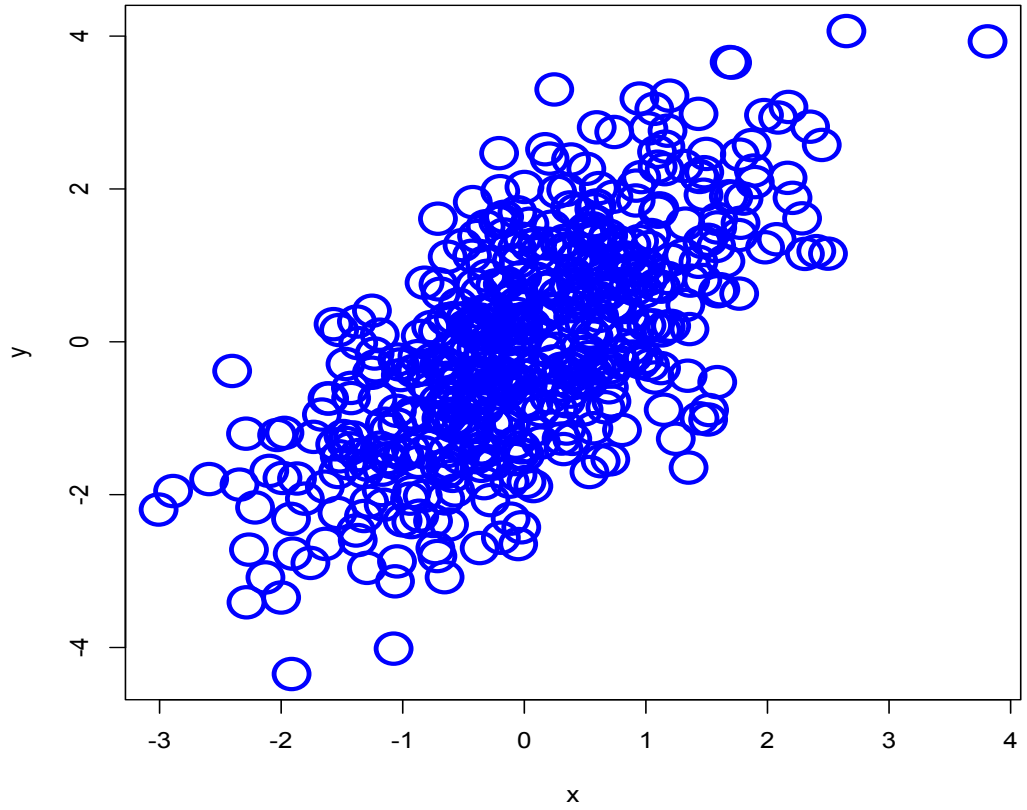
- bg = background



What will happen?

- # Example

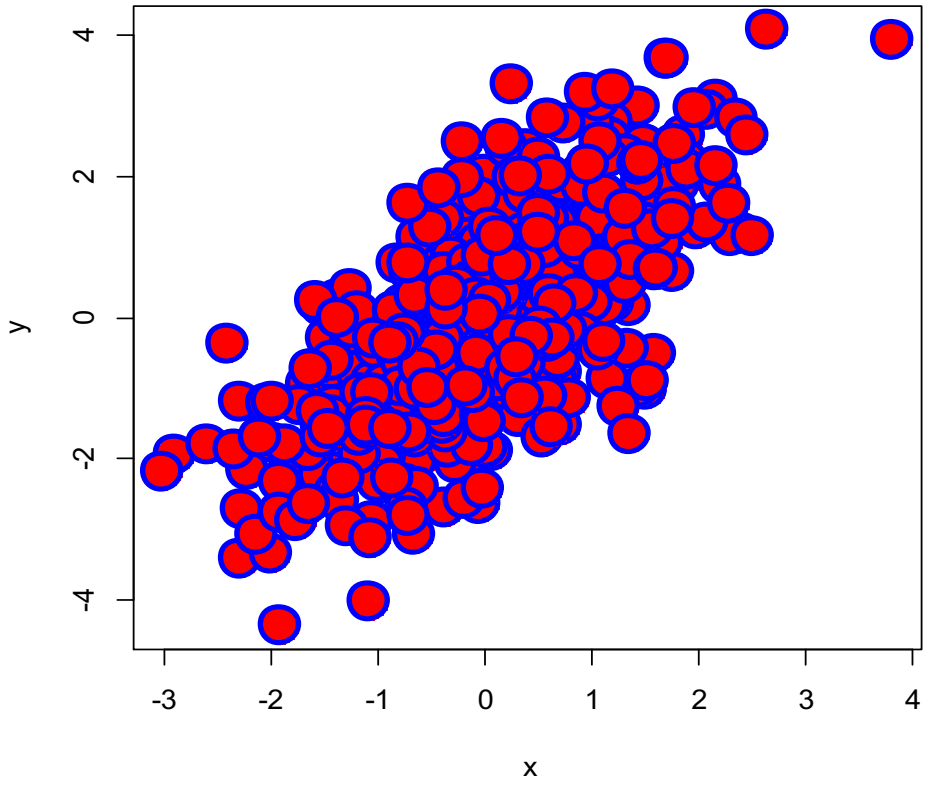
- plot(x, y,
- bg = "red", # Fill color
- col = "blue", # Border color
- cex = 3, # Symbol size
- lwd = 3) # Border width



What will happen?

- # Example

- plot(x, y, pch = 21,
- bg = "red", # Fill color
- col = "blue", # Border color
- cex = 3, # Symbol size
- lwd = 3) # Border width

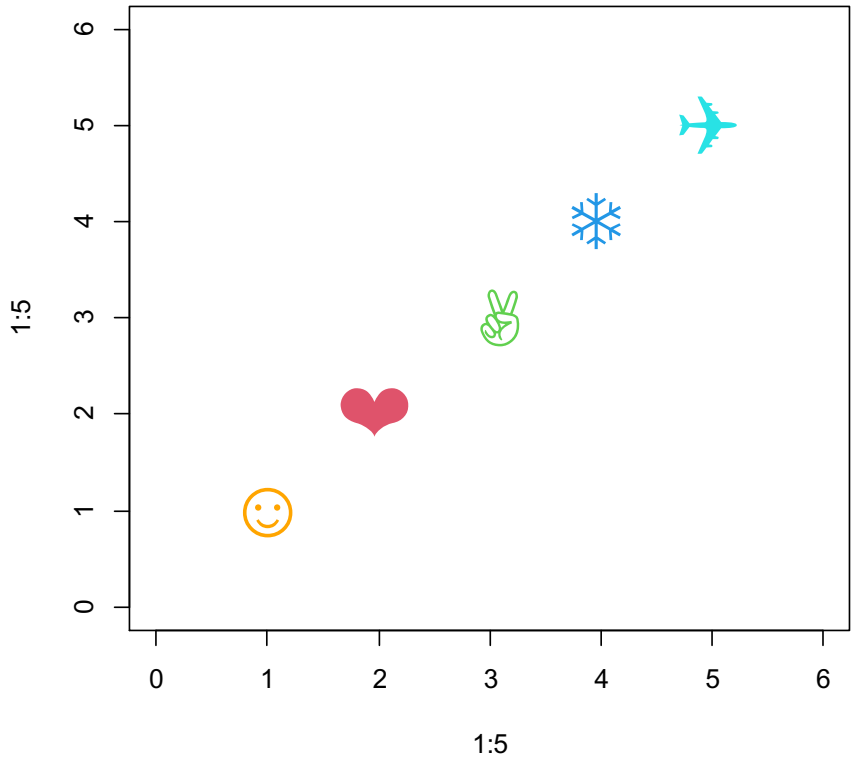


What will happen?

- # Custom symbols
- plot(1:5, 1:5, pch = c("☺️", "❤️", "✌️", "❄️", "✈️"),

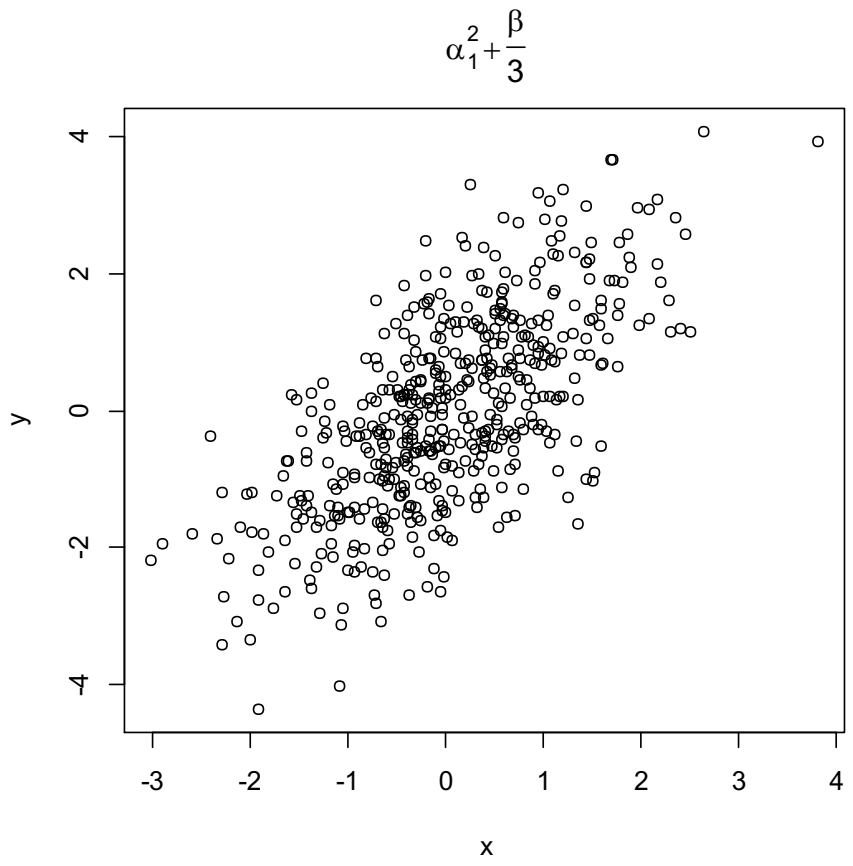
  col = c("orange", 2:5), cex = 3,

  xlim = c(0, 6), ylim = c(0, 6))



What will happen?

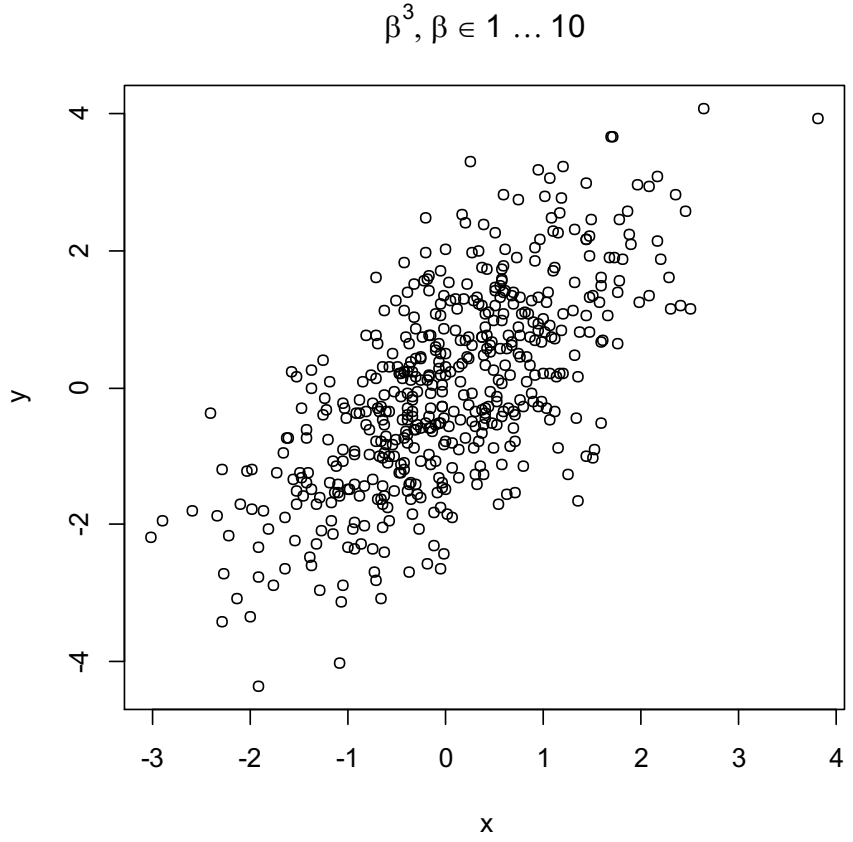
- plot(x, y, main = expression(alpha[1] ^ 2 + frac(beta, 3)))
- Read more by typing:
- ?plotmath in R console for LaTex type symbols of R!



How to add LaTex syntax?

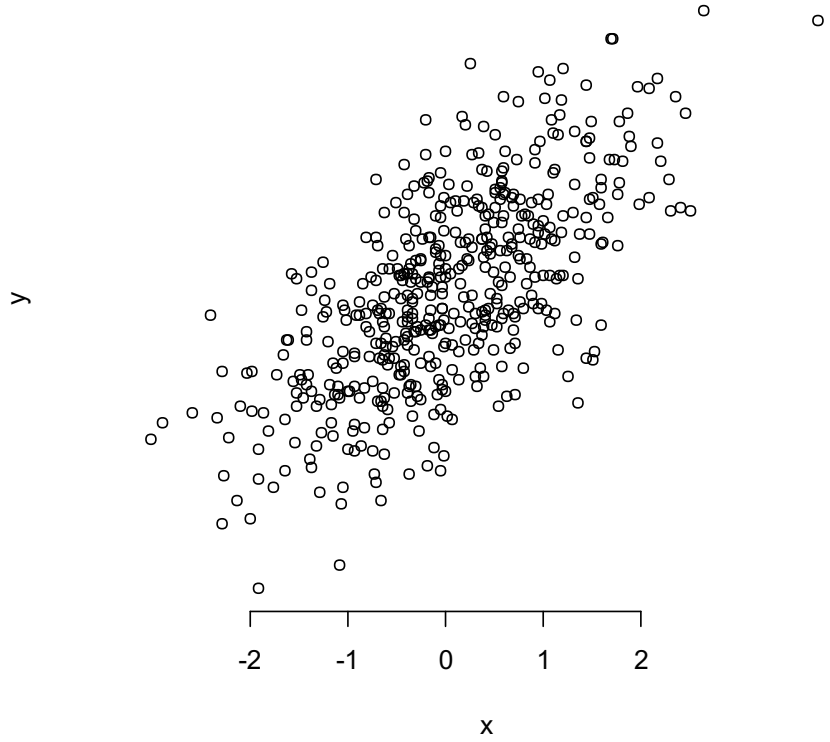
- install.packages("latex2exp")
- library(latex2exp)

- plot(x, y, main = TeX('$\\beta^3, \\beta \\in 1 \\ldots 10$'))



What will happen?

- plot(x, y, axes = FALSE)
- axis(1, at = -2:2)

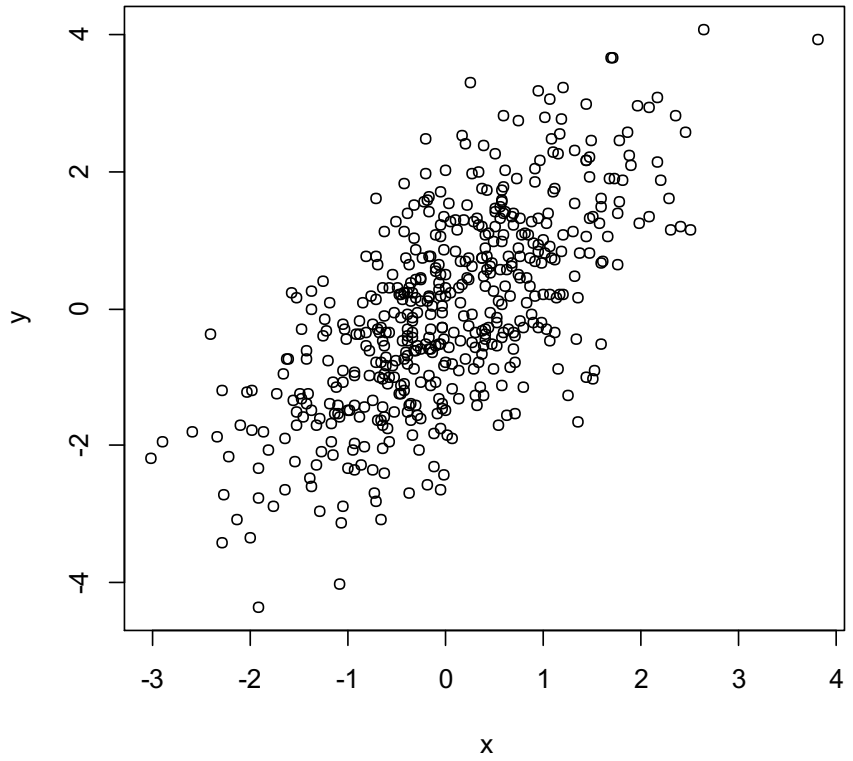


What will happen?

- install.packages("Hmisc")
- library(Hmisc)

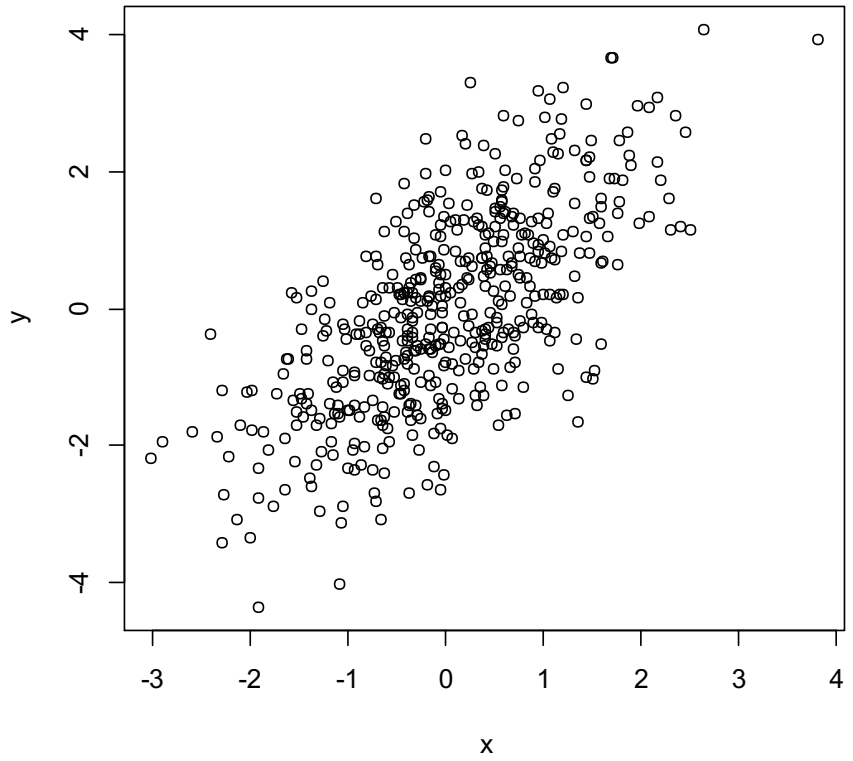
- plot(x, y)
- minor.tick(nx = 3, ny = 3, tick.ratio = 0.5)

- **Does it work?**



What will happen?

- # Interior ticks
- plot(x, y, tck = 0.02)



What will happen?

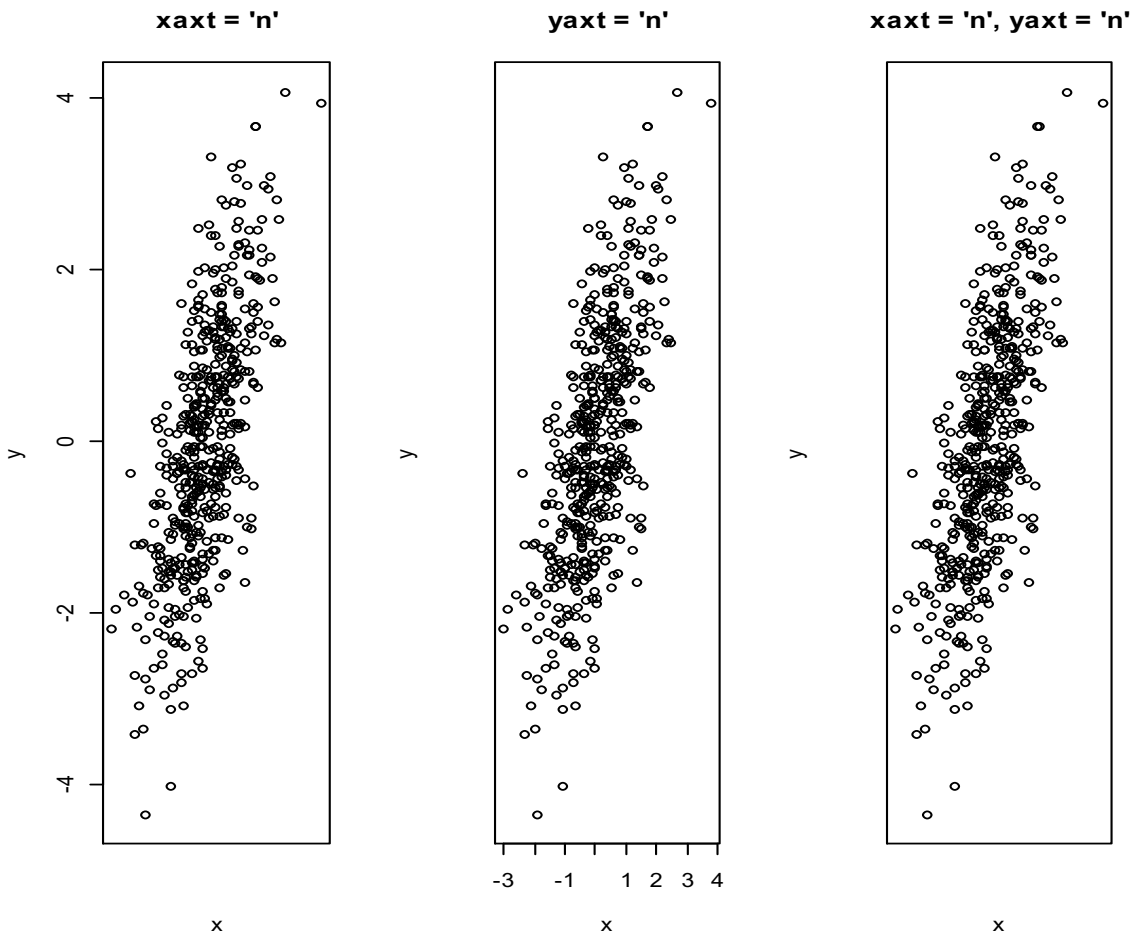
- par(mfrow = c(1, 3))

- # Remove X axis tick labels
- plot(x, y, xaxt = "n", main = "xaxt = 'n'")

- # Remove Y axis tick labels
- plot(x, y, yaxt = "n", main = "yaxt = 'n'")

- # Remove both axis tick labels
- plot(x, y, yaxt = "n", xaxt = "n", main = "xaxt = 'n', yaxt = 'n'")

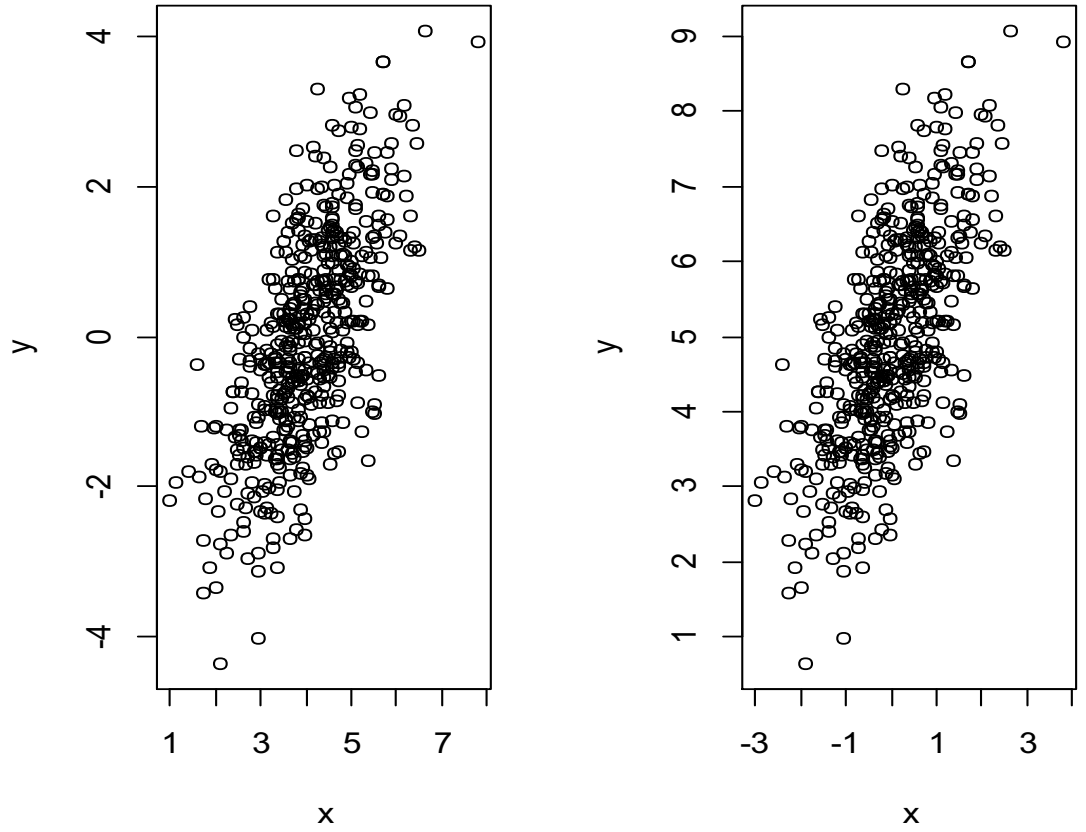
- par(mfrow = c(1, 1))



What will happen?

- par(mfrow = c(1, 2))
- # Change X axis tick labels
- plot(x, y, xaxt = "n")
- axis(1, at = seq(round(min(x)), round(max(x)), by = 1), labels = 1:8)

- # Change Y axis tick labels
- plot(x, y, yaxt = "n")
- axis(2, at = seq(round(min(y)), round(max(y)), by = 1), labels = 1:9)
- par(mfrow = c(1, 1))



What will happen?

las = label axis style

- par(mfrow = c(2, 2))

Parallel to axis (default)

- plot(x, y, **las = 0**, main = "Parallel")

Horizontal

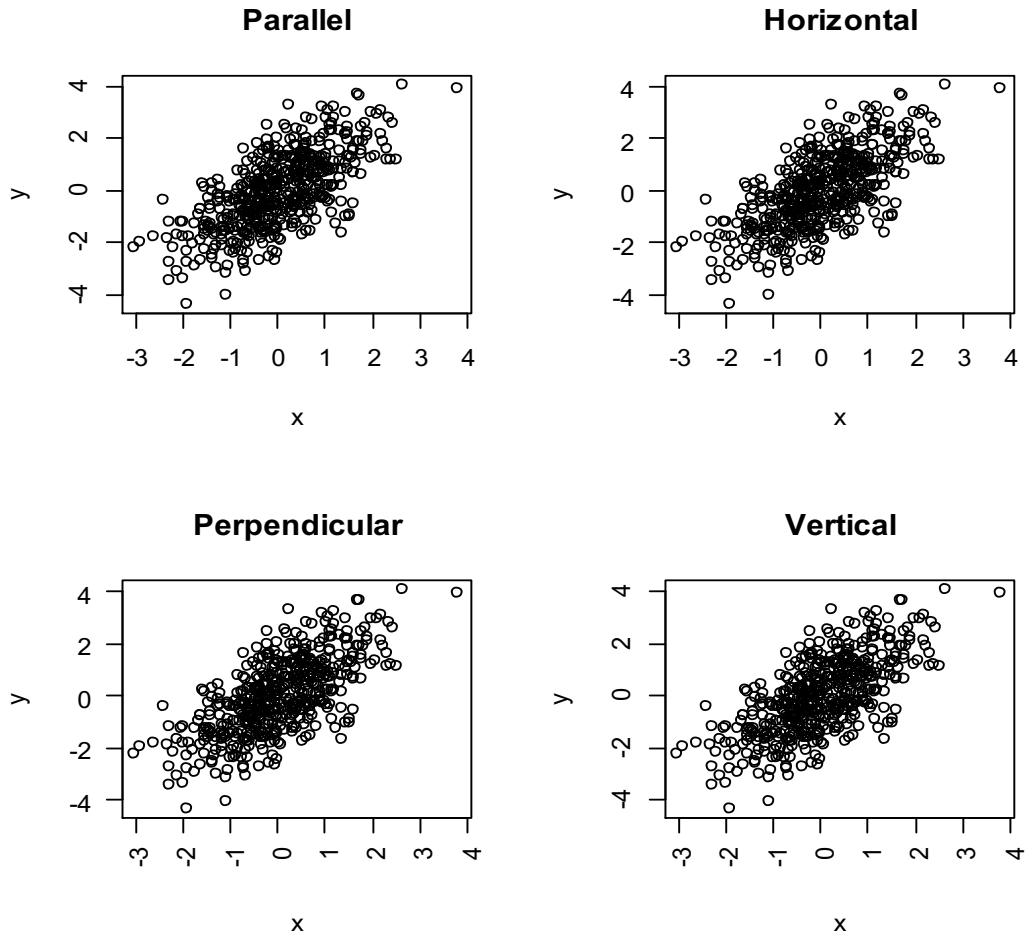
plot(x, y, **las = 1**, main = "Horizontal")

Perpendicular to axis

- plot(x, y, **las = 2**, main = "Perpendicular")

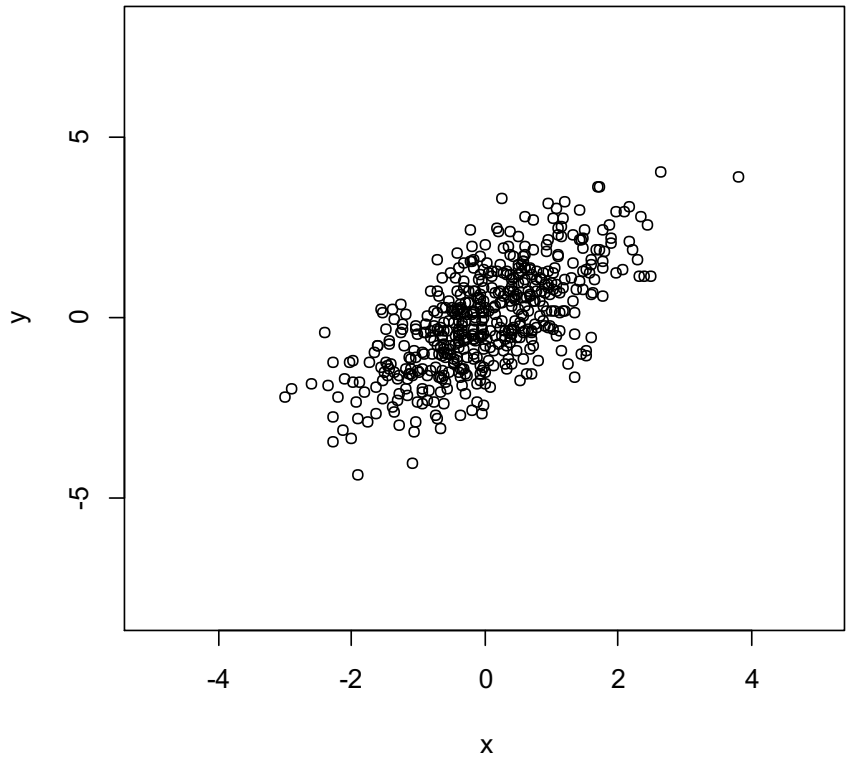
Vertical

- plot(x, y, **las = 3**, main = "Vertical")
- par(mfrow = c(1, 1))



What will happen?

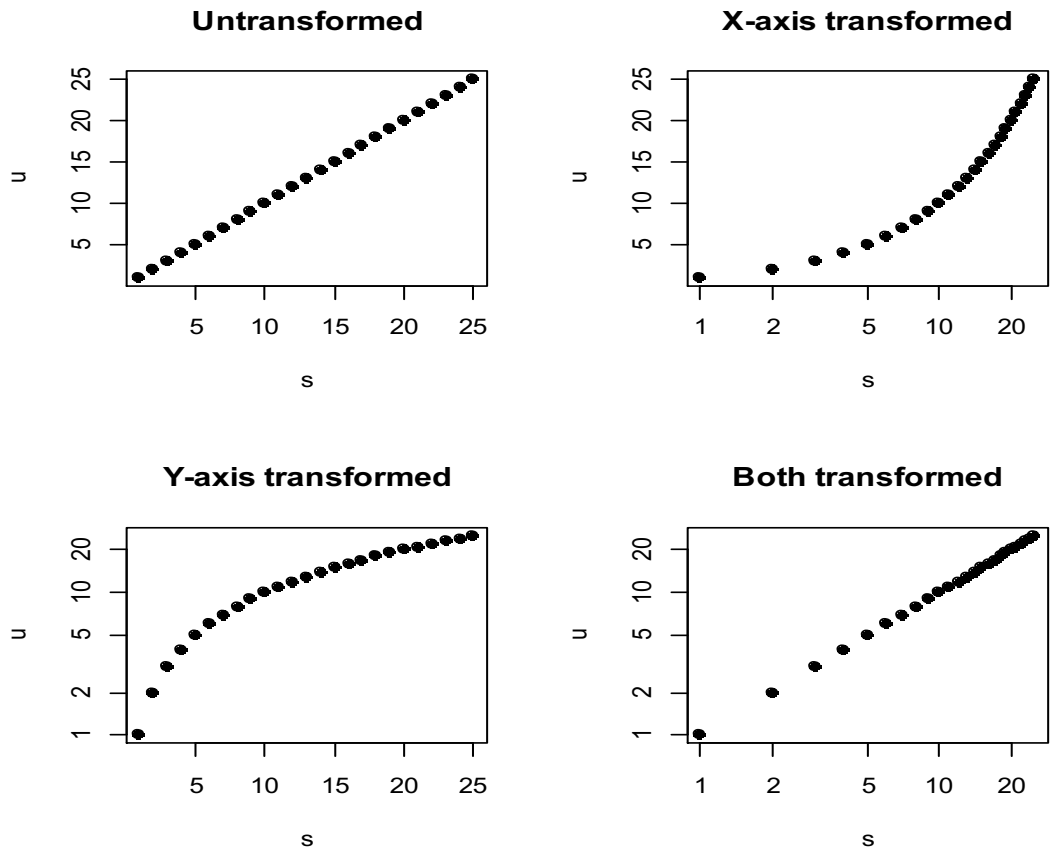
- plot(x, y,
- ylim = c(-8, 8), # Y-axis limits from -8 to 8
- xlim = c(-5, 5)) # X-axis limits from -5 to 5



What will happen?

 # New data to avoid negative numbers

- s <- 1:25
- u <- 1:25
- par(mfrow = c(2, 2))
- plot(s, u, pch = 19, main = "Untransformed")
- plot(s, u, pch = 19, log = "x", main = "X-axis transformed")
- plot(s, u, pch = 19, log = "y", main = "Y-axis transformed")
- plot(s, u, pch = 19, log = "xy", main = "Both transformed")
- par(mfrow = c(1, 1))



What will happen?

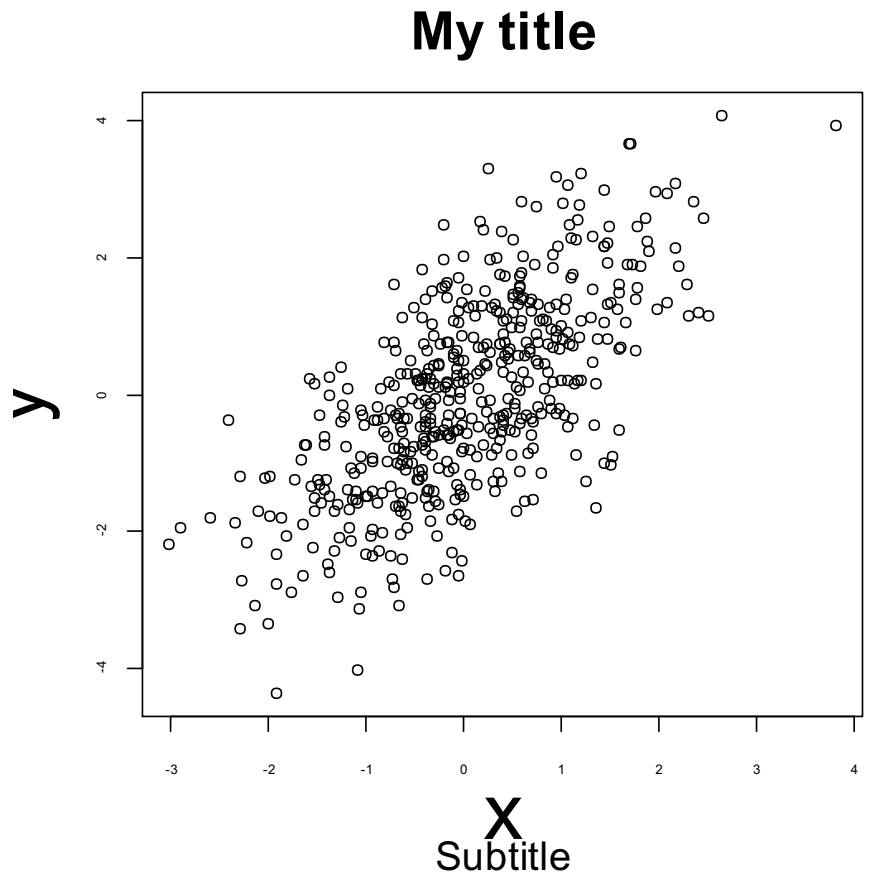
- plot(x, y, main = "My title", sub = "Subtitle",

  cex.main = 2, # Title size

  cex.sub = 1.5, # Subtitle size

  cex.lab = 3, # X-axis and Y-axis labels size

  cex.axis = 0.5) # Axis labels size



Note:

• cex.main	• Sets the size of the title
• cex.sub	• Sets the size of the subtitle
• cex.lab	• Sets the X and Y axis labels size
• cex.axis	• Sets the tick axis labels size

What will happen?

- par(mfrow = c(1, 3))

Bold

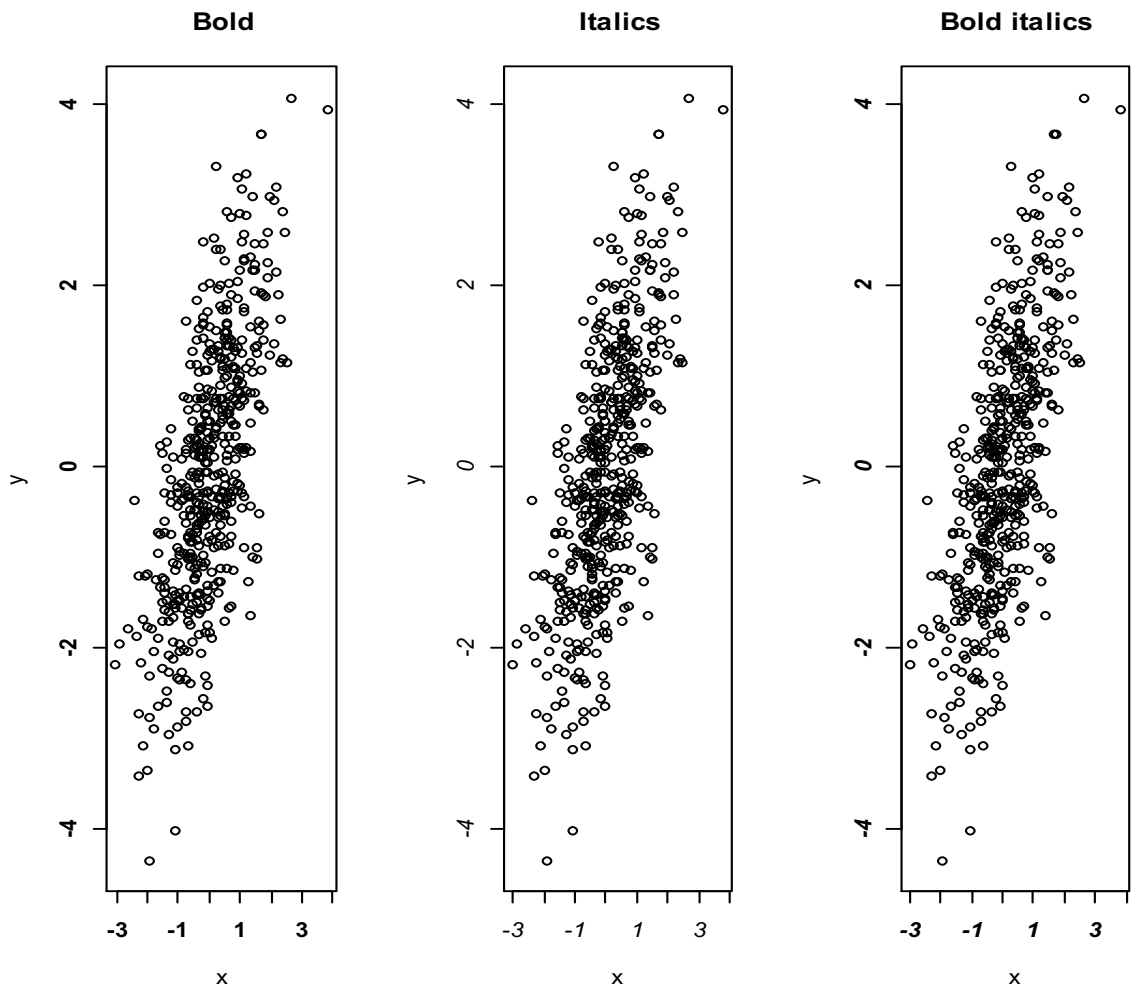
- plot(x, y, font = 2, main = "Bold")

Italics

- plot(x, y, font = 3, main = "Italics")

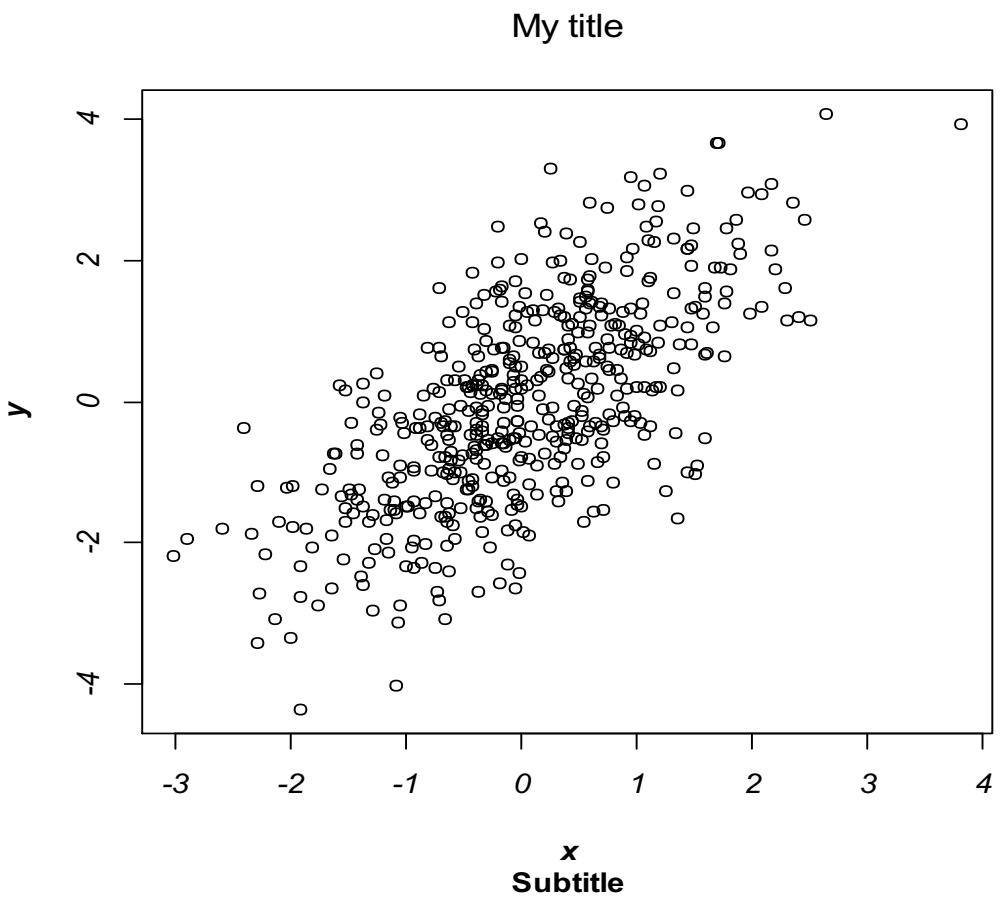
Bold italics

- plot(x, y, font = 4, main = "Bold italics")
- par(mfrow = c(1, 1))



What will happen?

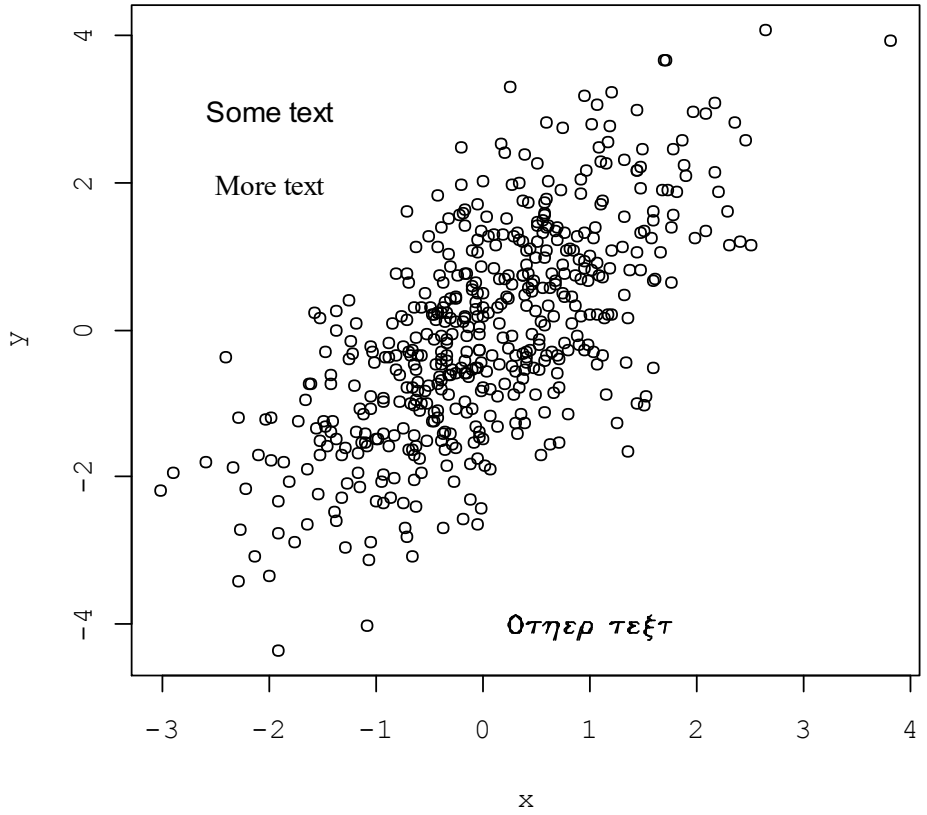
- plot(x, y,
- main = "My title",
- sub = "Subtitle",
- font.main = 1, # Title font style = plain
- font.sub = 2, # Subtitle font style = bold
- font.axis = 3, # Axis tick labels font style = italics
- font.lab = 4) # Font style of X and Y axis labels = Bold italics



What will happen?

- # All available fonts
- names(pdfFonts())

- plot(x, y, family = "mono")
- text(-2, 3, "Some text", family = "sans")
- text(-2, 2, "More text", family = "serif")
- text(1, -4, "Other text", family = "HersheySymbol")



Alternatively:

- An alternative is to use the extrafont package.

- # install.packages("extrafont")
- library(extrafont)

- # Auto detect the available fonts in your computer
- # This can take several minutes to run
- font_import()

- # Font family names
- fonts()

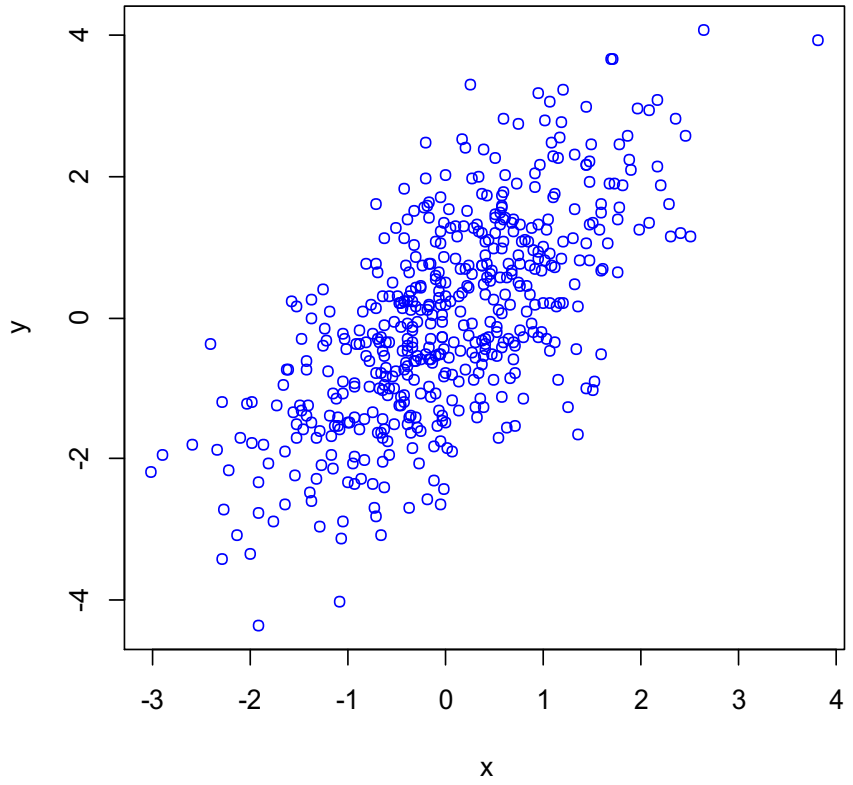
- # Data frame containing the font family names
- fonttable()

What will happen?

- # Return all colors
- colors()

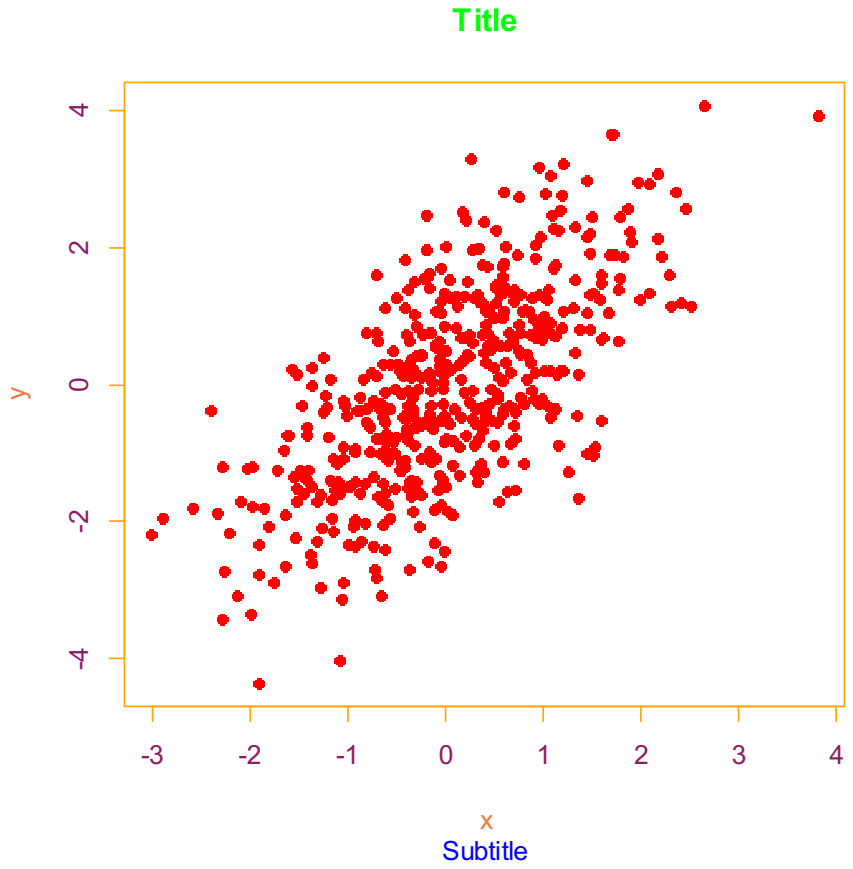
- # Return all colors that contain the word 'green'
- cl <- colors()
- cl[grep("green", cl)]

- # Plot with blue dots
- plot(x, y, col = "blue")
- plot(x, y, col = 4) # Equivalent
- plot(x, y, col = "#0000FF") # Equivalent



What will happen?

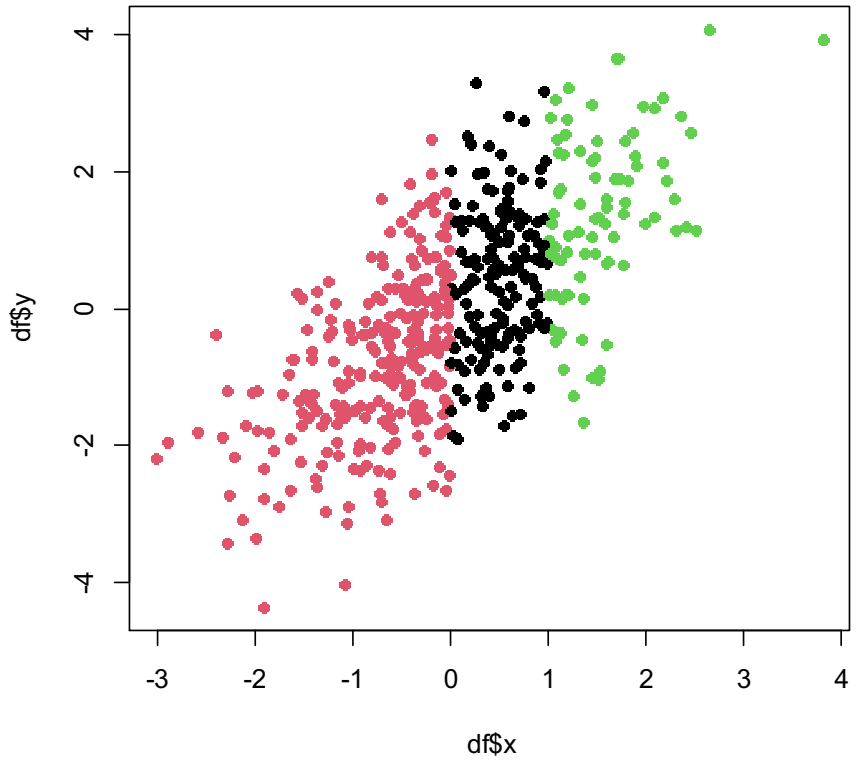
- plot(x, y, main = "Title", sub = "Subtitle",
- pch = 16,
- col = "red", # Symbol color
- col.main = "green", # Title color
- col.sub = "blue", # Subtitle color
- col.lab = "sienna2", # X and Y-axis labels color
- col.axis = "maroon4", # Tick labels color
- fg = "orange") # Box color



What will happen?

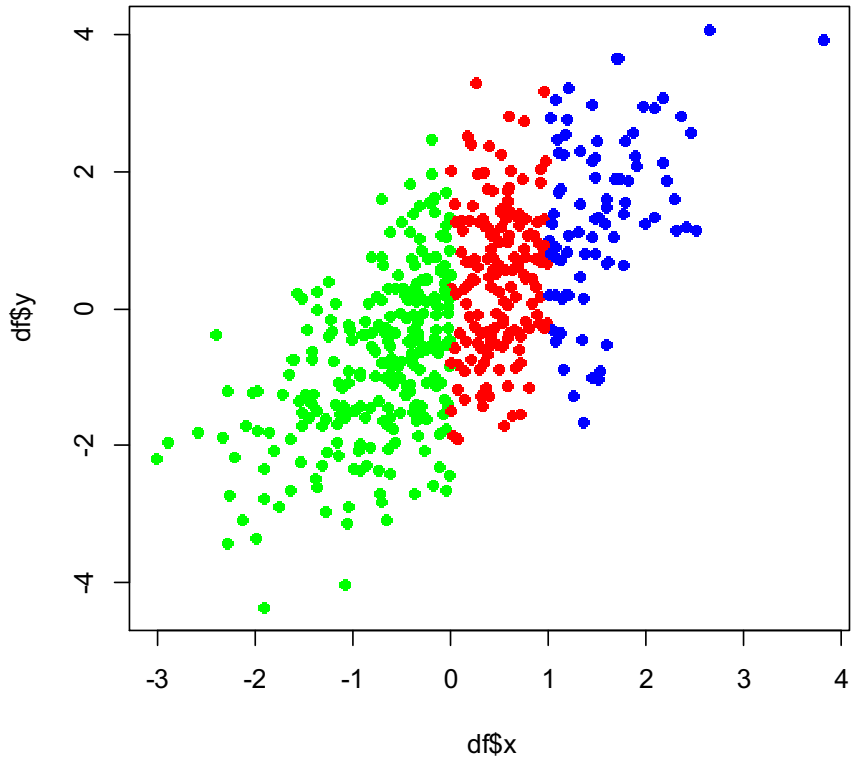
- # Create dataframe with groups
- group <- ifelse(x < 0 , "car", ifelse(x > 1, "plane", "boat"))
- df <- data.frame(x = x, y = y, group = factor(group))

- # Color by group
- plot(dfx, dfy, col = df$group, pch = 16)



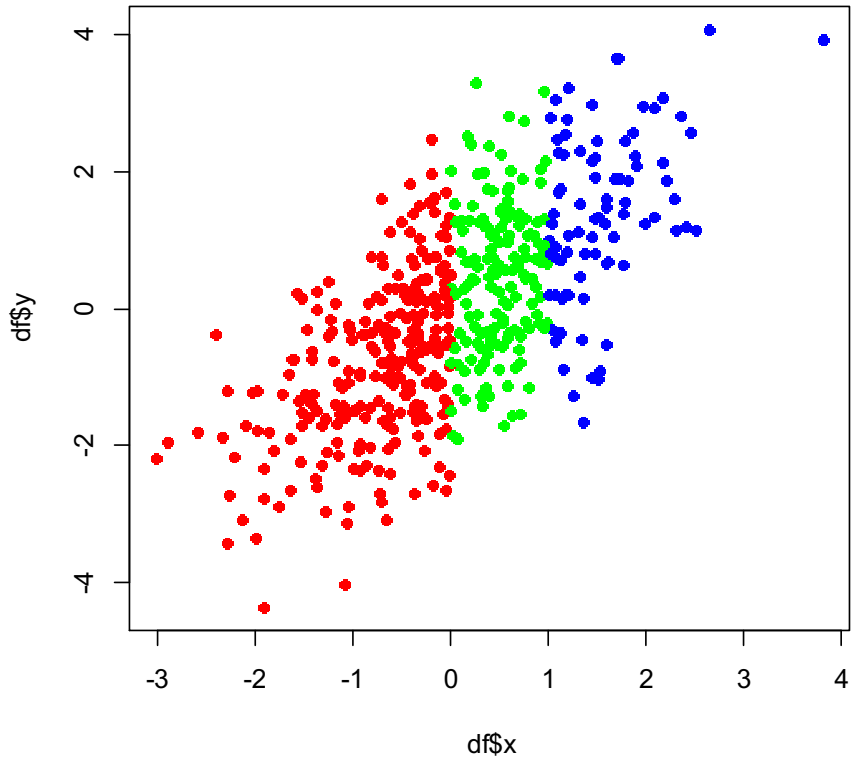
What will happen?

- # Change group colors
- colors <- c("red", "green", "blue")
- plot(dfx, dfy, col = colors[df$group], pch = 16)



What will happen?

- # Change color order, changing levels order
- plot(dfx, dfy, col = colors[factor(group, levels = c("car", "boat", "plane"))],
- pch = 16)



What will happen?

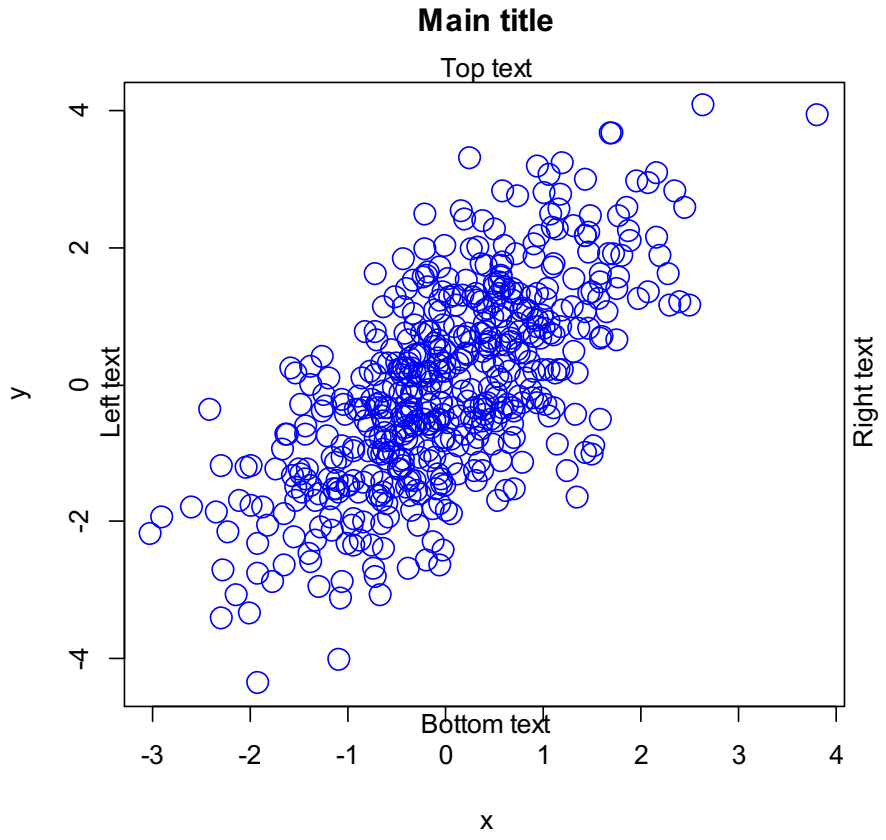
- plot(x, y, main = "Main title", cex = 2, col = "blue")

- # Bottom-center
- mtext("Bottom text", side = 1)

- # Left-center
- mtext("Left text", side = 2)

- # Top-center
- mtext("Top text", side = 3)

- # Right-center
- mtext("Right text", side = 4)



What will happen?

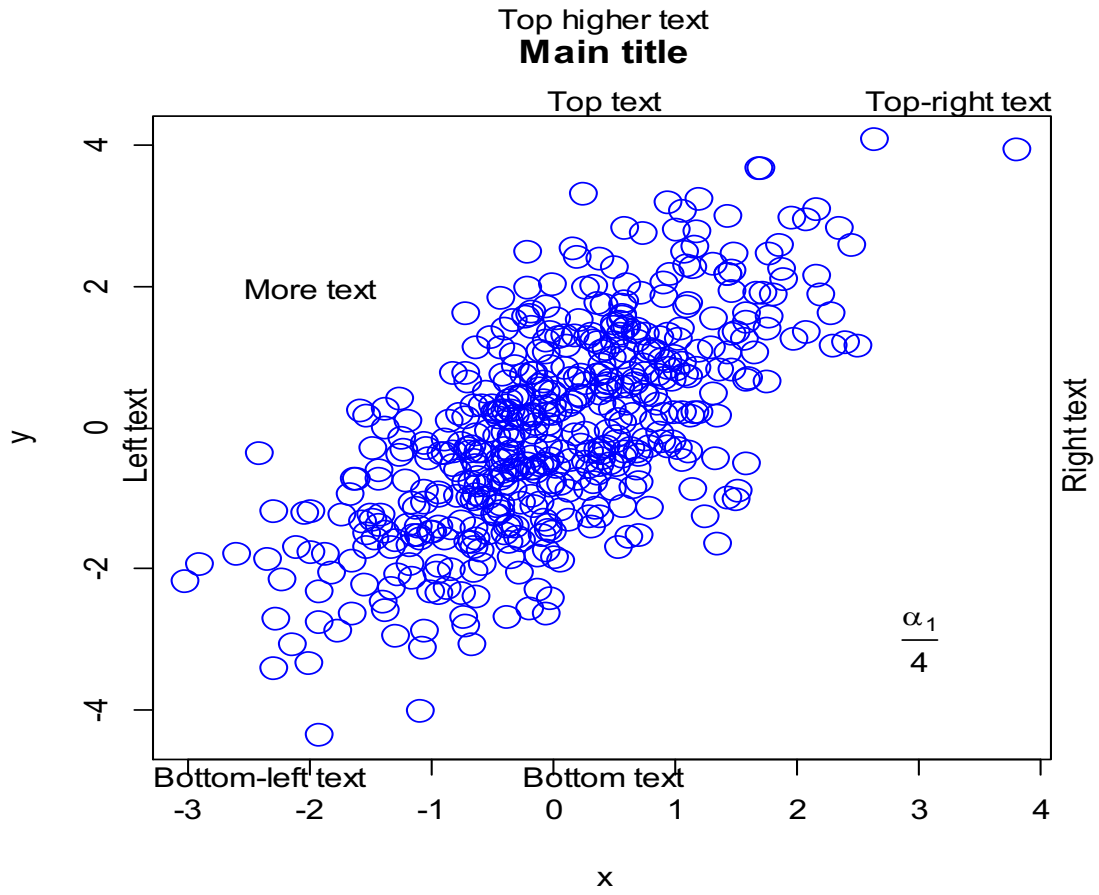
- # Bottom-left
- mtext("Bottom-left text", side = 1, adj = 0)

- # Top-right
- mtext("Top-right text", side = 3, adj = 1)

- # Top with separation
- mtext("Top higher text", side = 3, line = 2.5)

- # Add text at coordinates (-2, 2)
- text(-2, 2, "More text")

- # Add formula at coordinates (3, -3)
- text(3, -3, expression(frac(alpha[1], 4)))



What will happen?

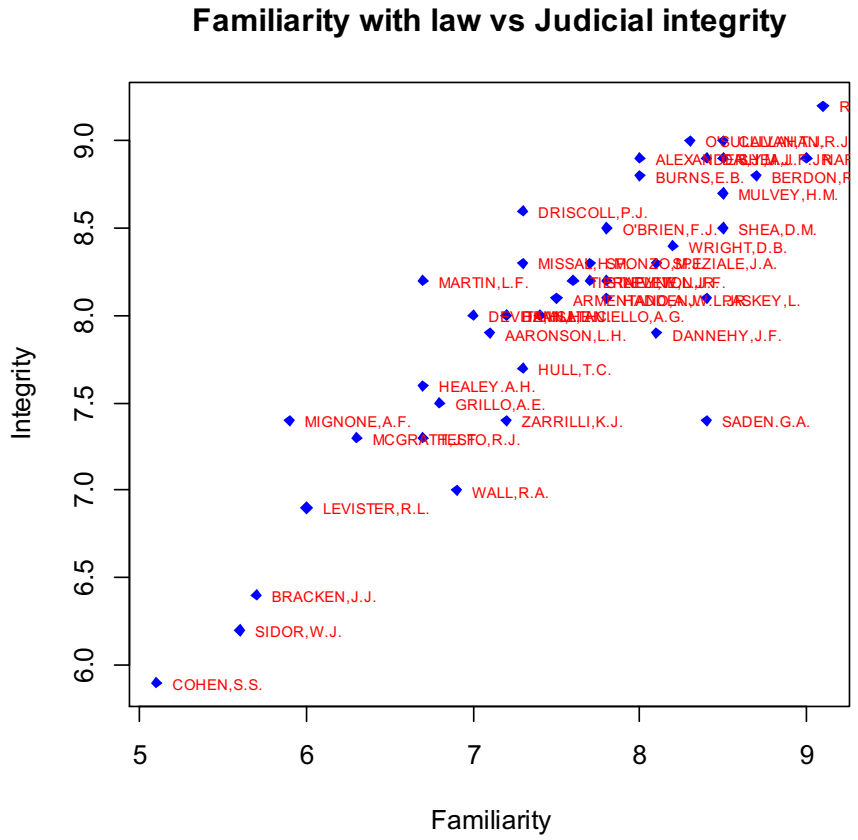
Adding row.names to the plot!

- attach(USJudgeRatings)

- # Create the plot
- plot(FAMI, INTG,
- main = "Familiarity with law vs Judicial integrity",
- xlab = "Familiarity", ylab = "Integrity",
- pch = 18, col = "blue")

- # Plot the labels
- text(FAMI, INTG,
- labels = row.names(USJudgeRatings),
- cex = 0.6, pos = 4, col = "red")

- detach(USJudgeRatings)



What will happen?

Adding selecting row.names to the plot!

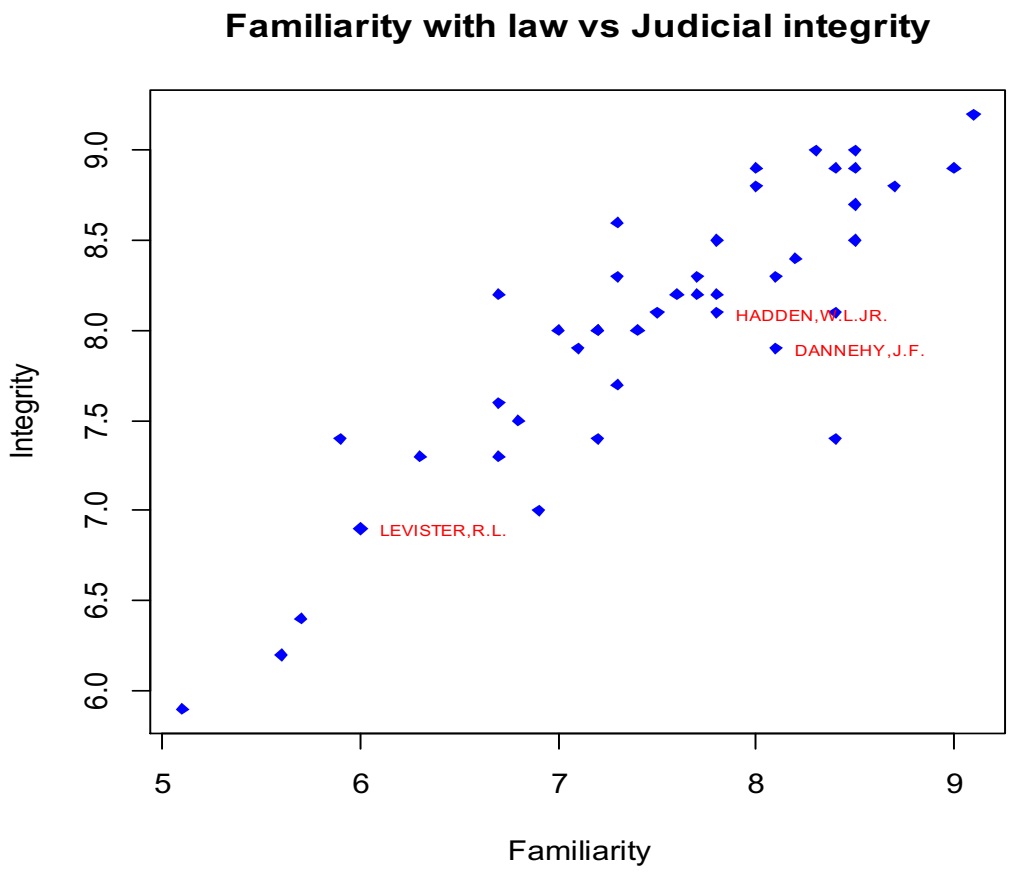
- attach(USJudgeRatings)

- plot(FAMI, INTG,
- main = "Familiarity with law vs Judicial integrity",
- xlab = "Familiarity", ylab = "Integrity",
- pch = 18, col = "blue")

- # Select the index of the elements to be labelled
- selected <- c(10, 15, 20)

- # Index the elements with the vector
- text(FAMI[selected], INTG[selected],
- labels = row.names(USJudgeRatings)[selected],
- cex = 0.6, pos = 4, col = "red")

- detach(USJudgeRatings)

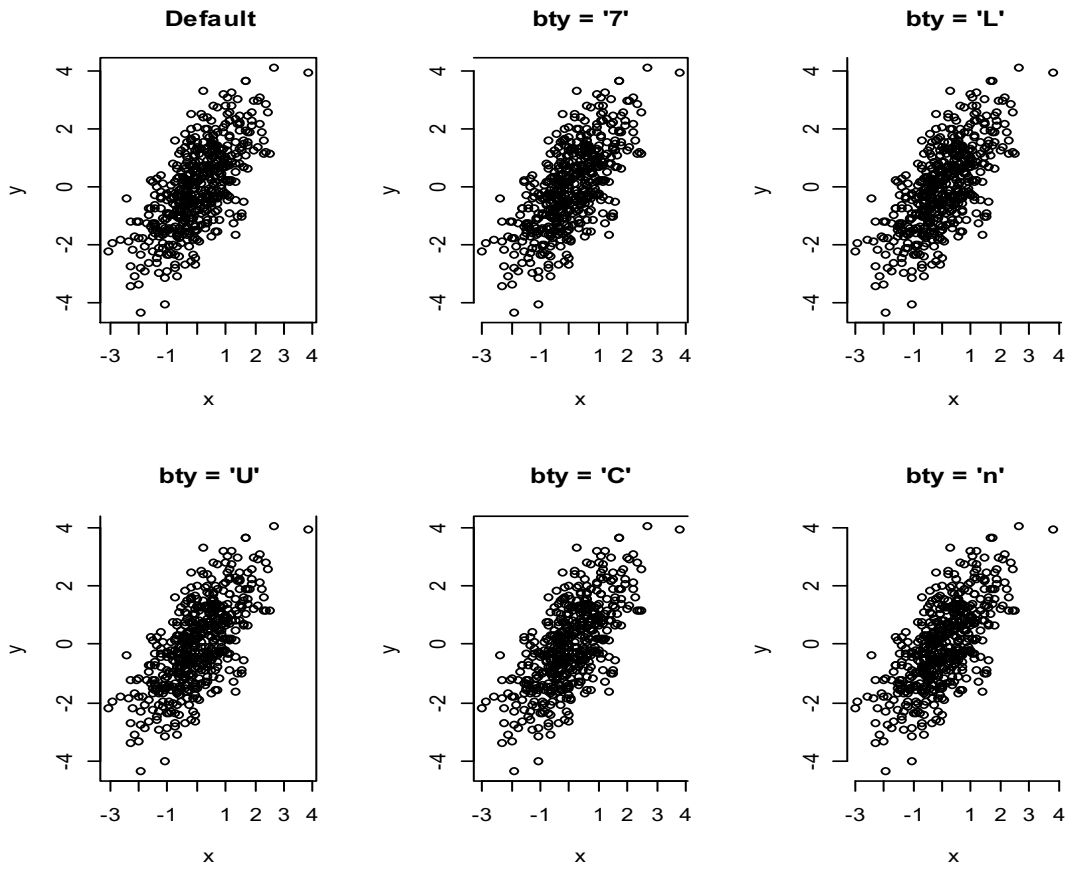


What will happen?

- par(mfrow = c(2, 3))

- plot(x, y, bty = "o", main = "Default")
- plot(x, y, bty = "7", main = "bty = '7'")
- plot(x, y, bty = "L", main = "bty = 'L'")
- plot(x, y, bty = "U", main = "bty = 'U'")
- plot(x, y, bty = "C", main = "bty = 'C'")
- plot(x, y, bty = "n", main = "bty = 'n'")

- par(mfrow = c(1, 1))



Note: “bty” (inside the par function)

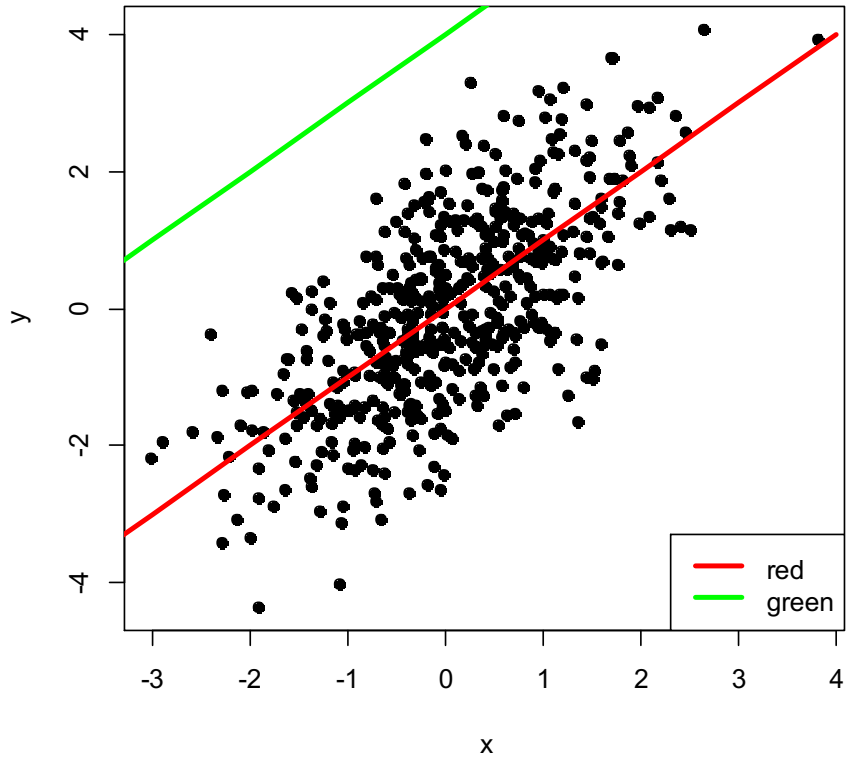
- “o”
- “7”
- “L”
- “U”
- “C”
- “n”

- Entire box (default)
- Top and right
- Left and bottom
- Left, bottom and right
- Top, left and bottom
- No box

What will happen?

- plot(x, y, pch = 19)
- lines(-4:4, -4:4, lwd = 3, col = "red")
- lines(-4:1, 0:5, lwd = 3, col = "green")

- # Adding a legend
- legend("bottomright", legend = c("red", "green"),
- lwd = 3, col = c("red", "green"))



Question/Queries?

Thank you!

@shitalbhandary