Statistical Computing with R: MDS 503 (S14) Fourth Batch, SMS, TU, 2025

Shital Bhandary
Associate Professor
Statistics/Bio-statistics, Demography and Public Health Informatics
Patan Academy of Health Sciences, Lalitpur, Nepal
Faculty, Masters in Medical Research, NHRC/Kathmandu University
Faculty, FAIMER Fellowship in Health Professions Education, India/USA

Review Preview

- Basic graphics/plots:
  - Goodness-of-fit test
  - Multiple graphs in a single window

- Special graph:
  - Social Network Analysis

Test of normality: Goodness-of-fit test after graphical exploration of a continuous variable

- Null hypothesis (H_0): Observed data (of a variable under consideration) follows normal distribution ($p>0.05$)

- Null hypothesis (H_1): Observed data (of a variable under consideration) does not follow normal distribution ($p<=0.05$)

- Test of normality is confirmatory test

- Before using this:
  - We use histogram
  - We use density plot
  - We use normal Q-Q plot

Test of normality: types

- Skewness and kurtosis based
  - Jarque-Bera test of normality
- Large sample based
  - Kolmogorov-Smirnov test of normality
  - **What is the code for it in R?**
- Small sample based
  - Shapiro-Wilk test of normality
- How large?
- How small?
- **Did we use the correct test of normality in the aq data?**

Let us use "airquality" data: aq <- airquality

- Ozone variable of aq dataframe

- hist(aq$Ozone)
- plot(density(aq$Ozone, na.rm = T))
- qqnorm(aq$Ozone)
- qqline(aq$Ozone, col = 2)
- shapiro.test(aq$Ozone)

- Solar Radiation variable

- hist(aq$Solar.R)
- plot(density(aq$Solar.R, na.rm = T))
- qqnorm(aq$Solar.R)
- qqline(aq$Solar.R, col = 2)
- shapiro.test(aq$Solar.R)

Testing normality of continuous variable

- Wind variable of aq dataframe

- hist(aq$Wind)
- plot(density(aq$Wind, na.rm = T))
- qqnorm(aq$Wind)
- qqline(aq$Wind, col = 2)
- shapiro.test(aq$Wind)

- Speed variable of cars dataframe

- hist(cars$speed)
- plot(density(cars$speed, na.rm = T))
- qqnorm(cars$speed)
- qqline(cars$speed, col = 2)
- shapiro.test(cars$speed)

Parametric vs Non-parametric tests (We will practice it in Unit 4)

- If a dependent variable (speed, wind etc.) is normally distributed then we use parametric tests:
- T-test = compare mean of the dependent variable in two groups e.g. speed by 4 and 6 cylinder cars
- 1-way ANOVA = compare mean of the dependent variable in more than two groups e.g. speed by 4, 6 and 8 cylinder cars
- Post-hoc test is done if 1-way ANOVA p-value <=0.05

- If a dependent variable (speed, wind etc.) is not normally distributed then we use non-parametric tests:
- Mann-Whitney test = compare median of the dependent variable in two groups e.g. speed by 4 and 6 cylinder cars
- Kruskal-Wallis test = compare median of the dependent variable in more than two groups e.g. speed by 4, 6 and 8 cylinder cars
- Post-hoc test is done if K-W test p-value <=0.05

Multiple graphs in a single window?

# Random Data

- x <- rnorm(500)
- y <- x + rnorm(500)

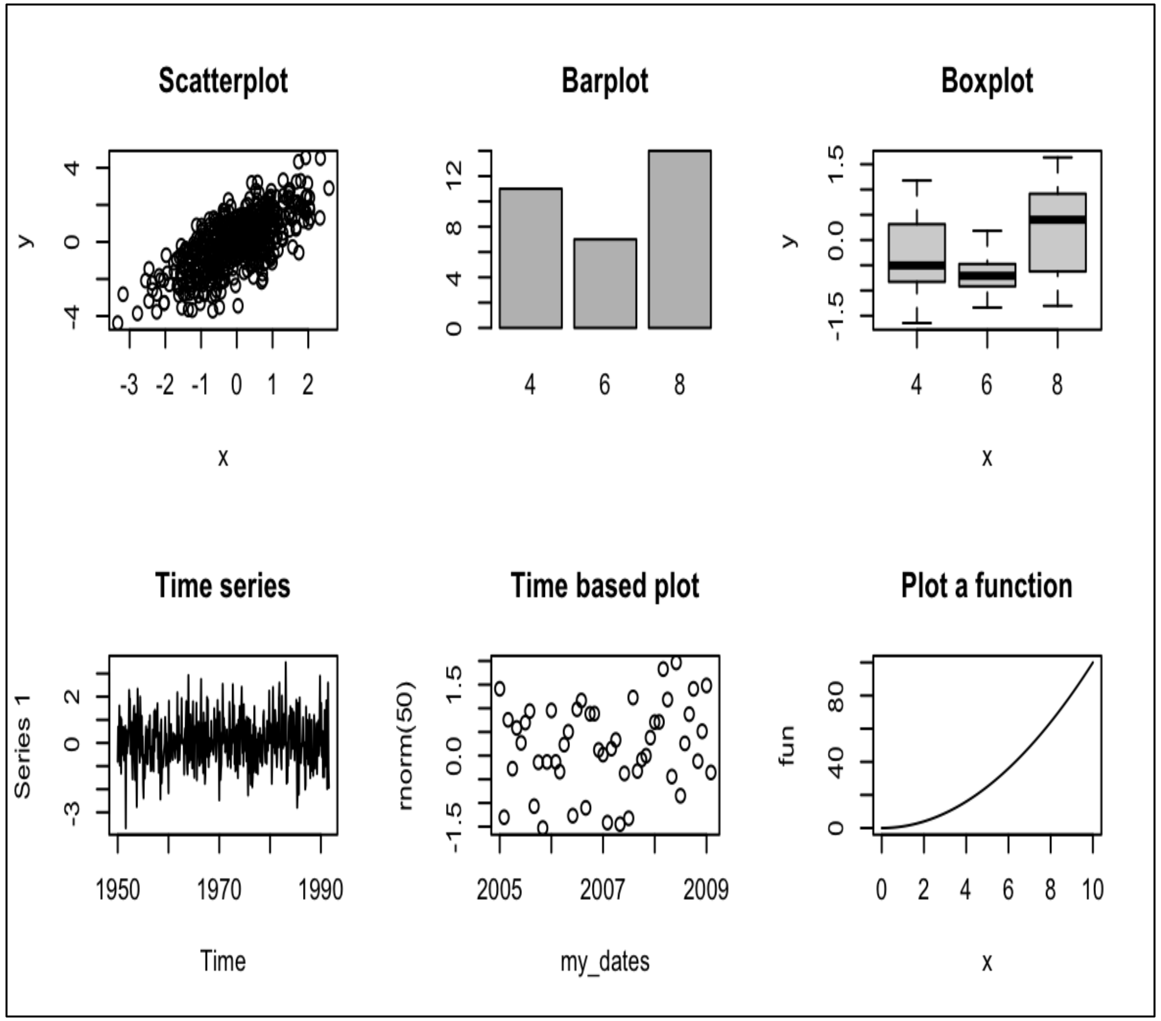
# Data

- 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

Multiple graphs in a single window?

#Creat a window for graphs in 2 rows and 3 columns

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

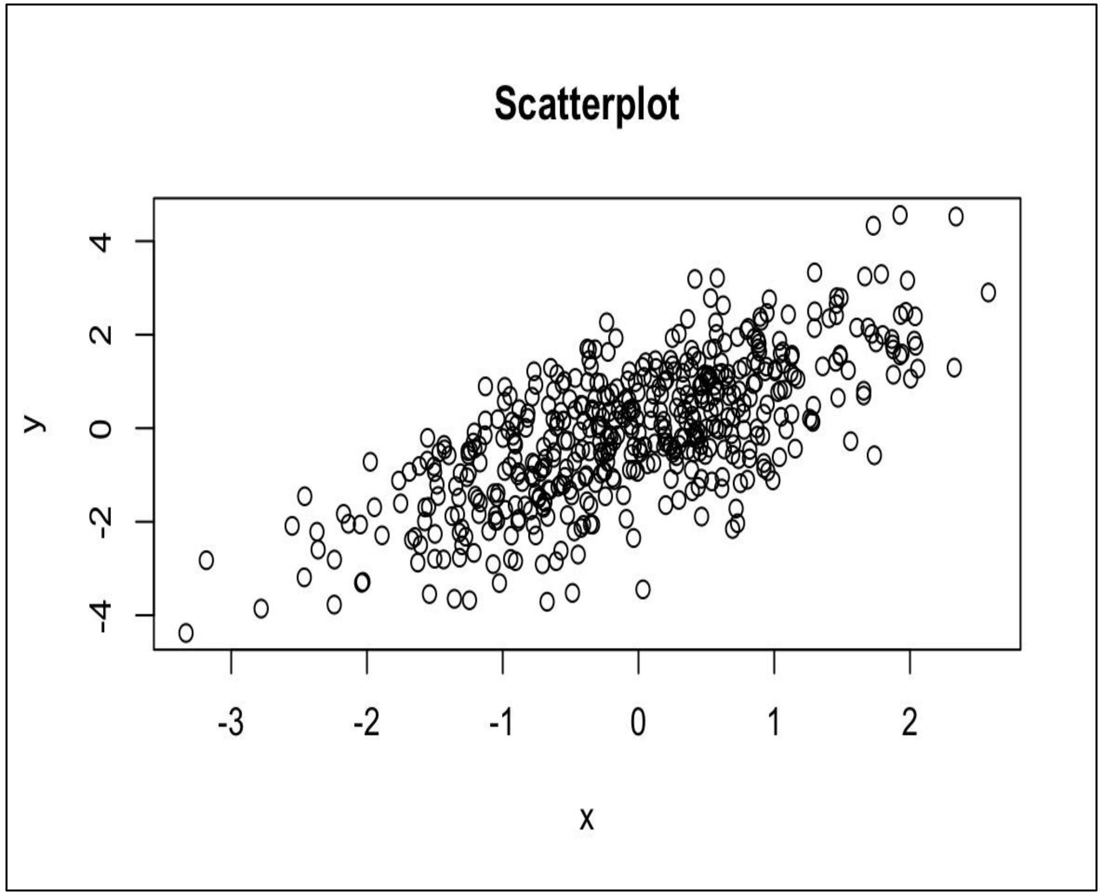


Changing to default mode

#Graph is default mode

- **par(mfrow = c(1, 1))**
- plot(x, y, main = "Scatterplot")

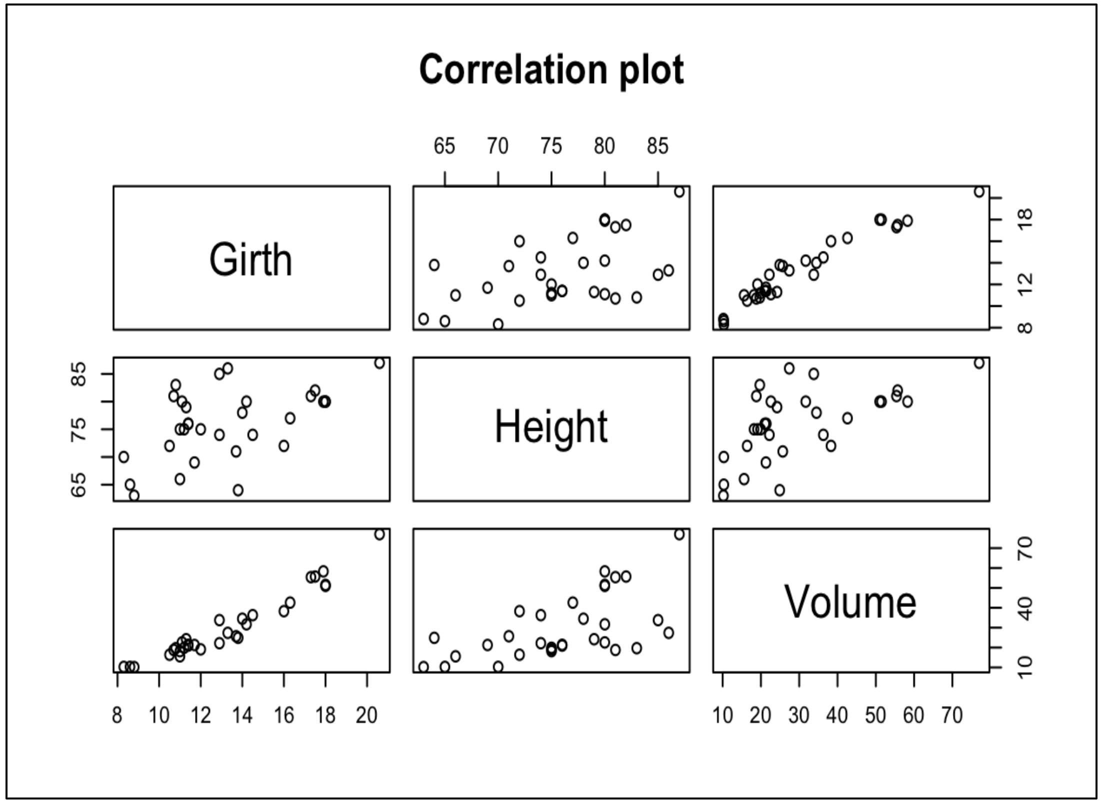
- The scatterplot of two random variables x and y looks “tentative” linear
- So we can use Pearson’s linear correlation coefficient here



Correlation matrix plot of 3 continuous variable (multiple plots in a single window)

#Correlation matrix plot

- plot(trees[, 1:3], main = "Correlation plot")
- Girth and Height is not linear so we must use Spearman's correlation coefficient
- Girth and Volume is linear so we must use Pearson's correlation coefficient
- **Height and Volume?**



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))

- 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'")

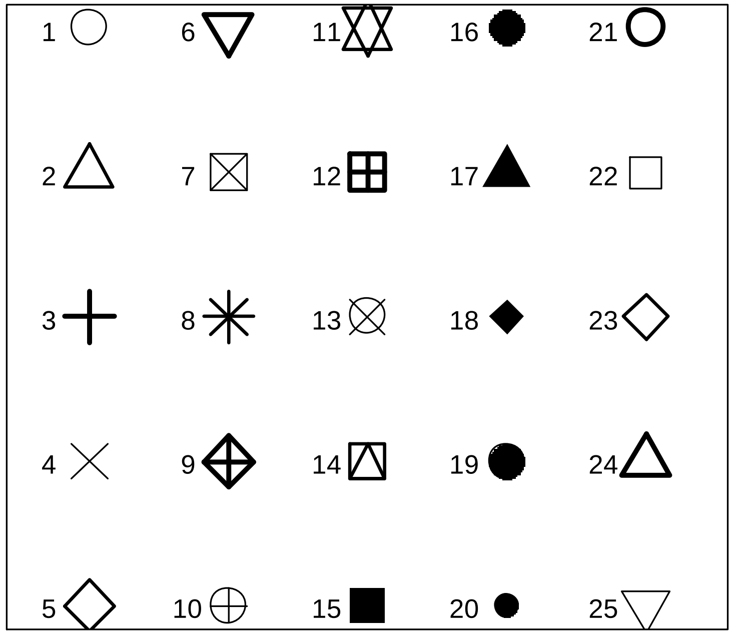
- par(mfrow = c(1, 1))

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)

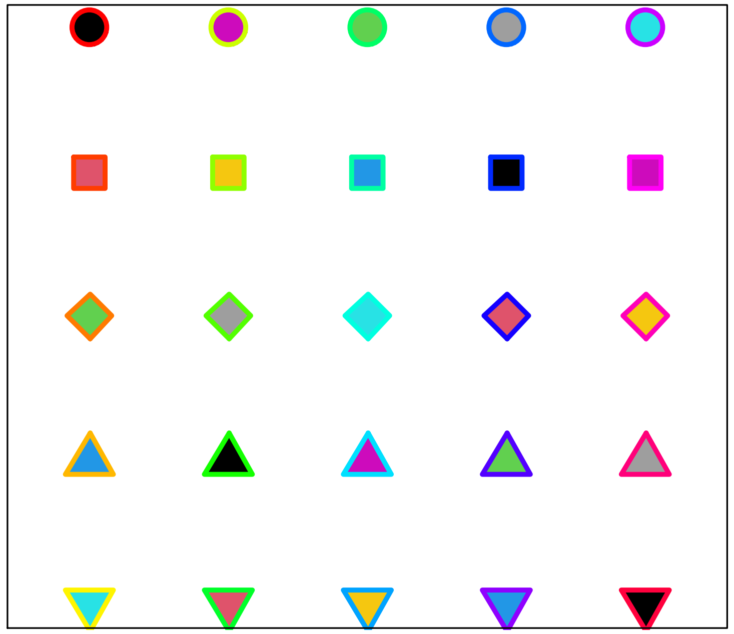


Note:

- 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!**

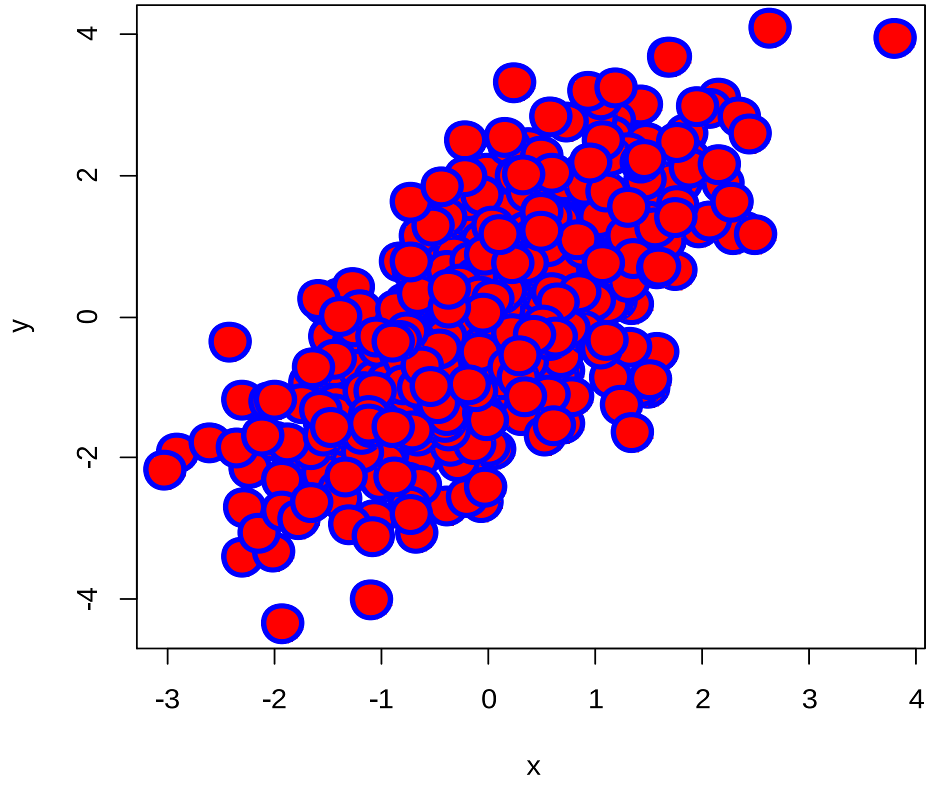
What will happen now?

- 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))



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

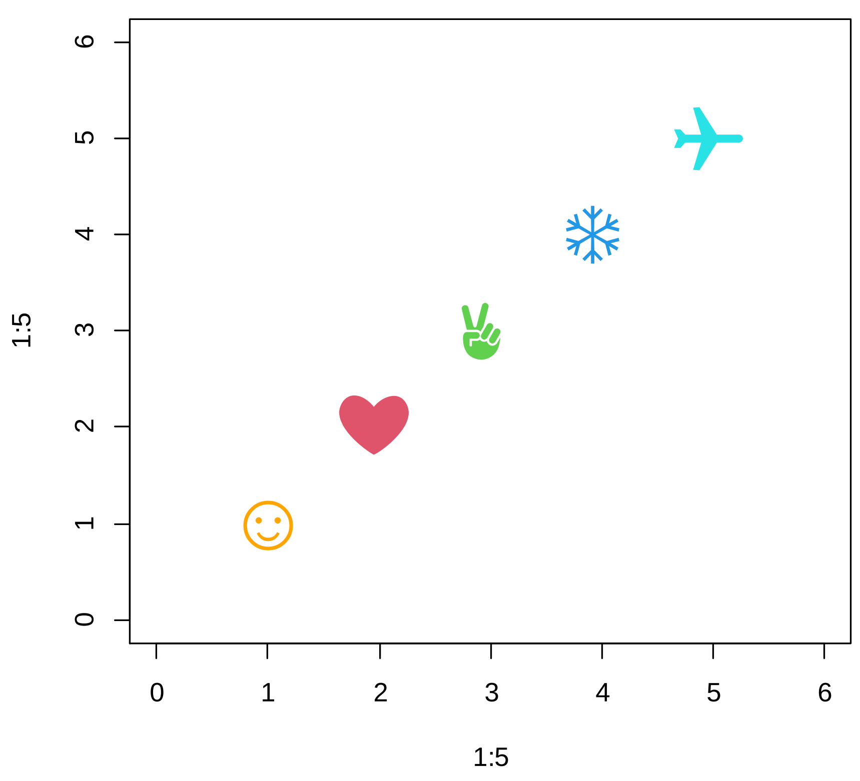


What will happen?

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

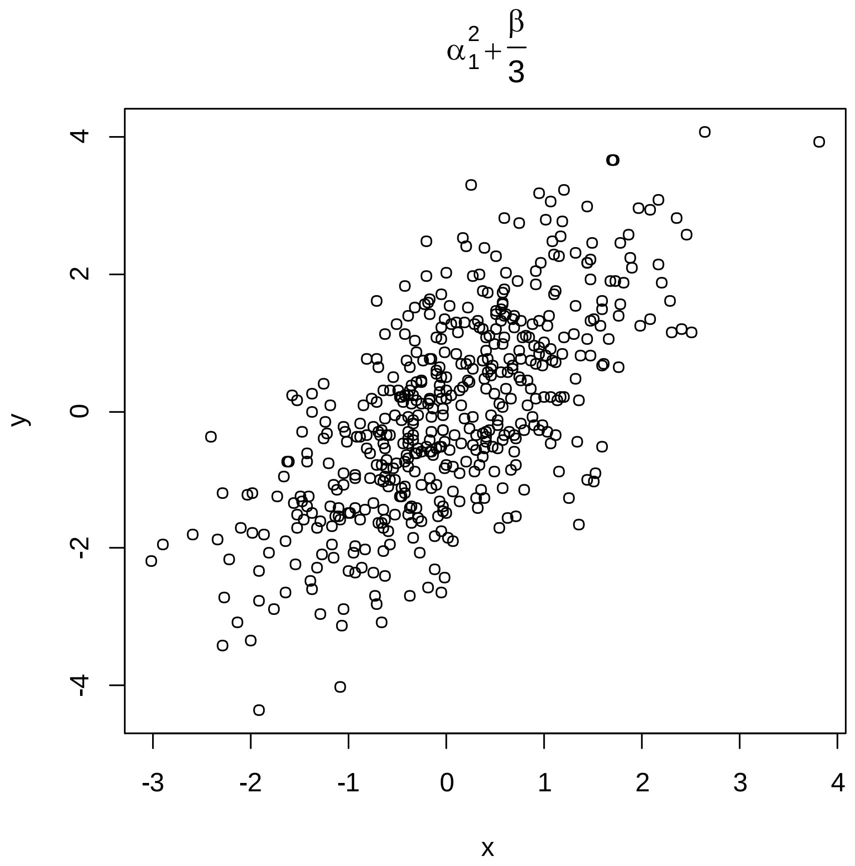
    col = c("orange", 2:5), cex = 3,

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



What will happen?

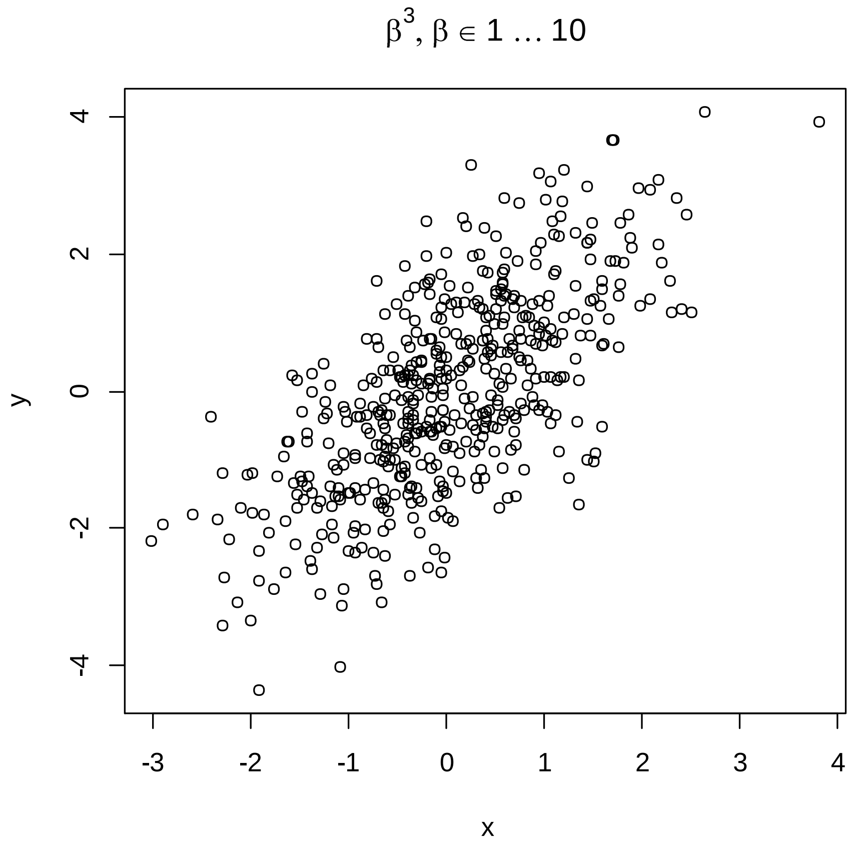
- plot(x, y, main = expression(alpha[1] ^ 2 + frac(beta, 3)))
- Read more by typing:
- ?plotmath in R console for LaTex type symbols of R!



How to add LaTex syntax?

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

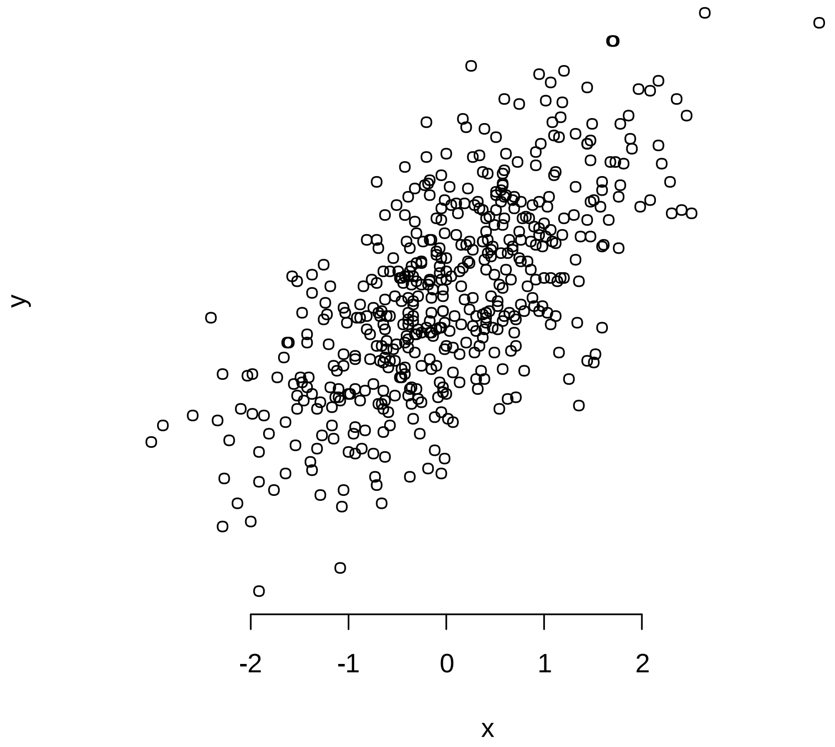
- plot(x, y, main = TeX('$\\beta^3, \\beta \\in 1 \\ldots 10$'))



What will happen?

- plot(x, y, axes = FALSE)

- axis(1, at = -2:2)

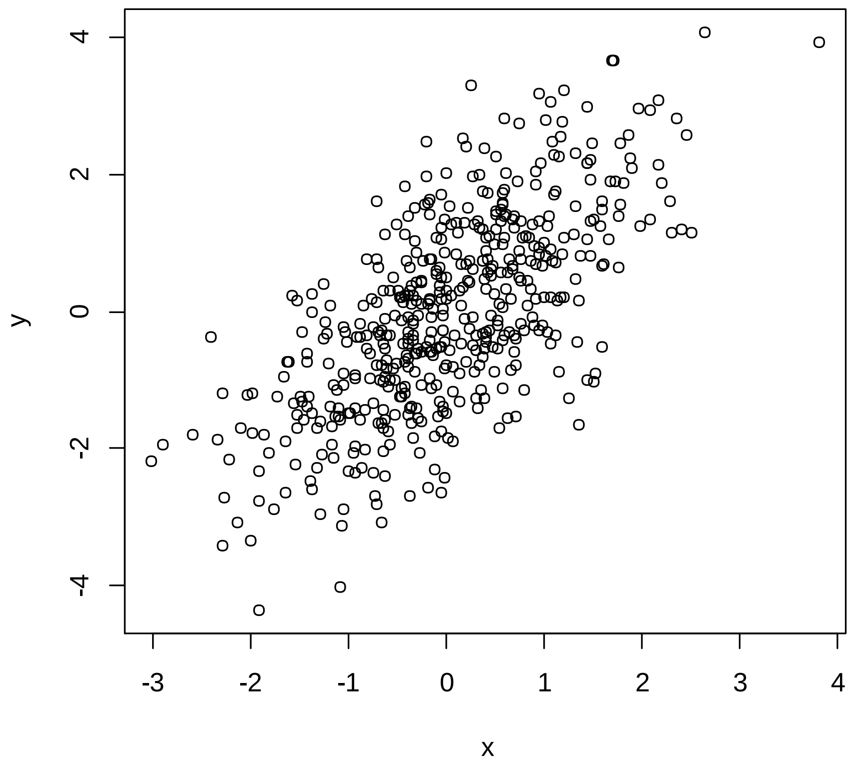


What will happen?

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

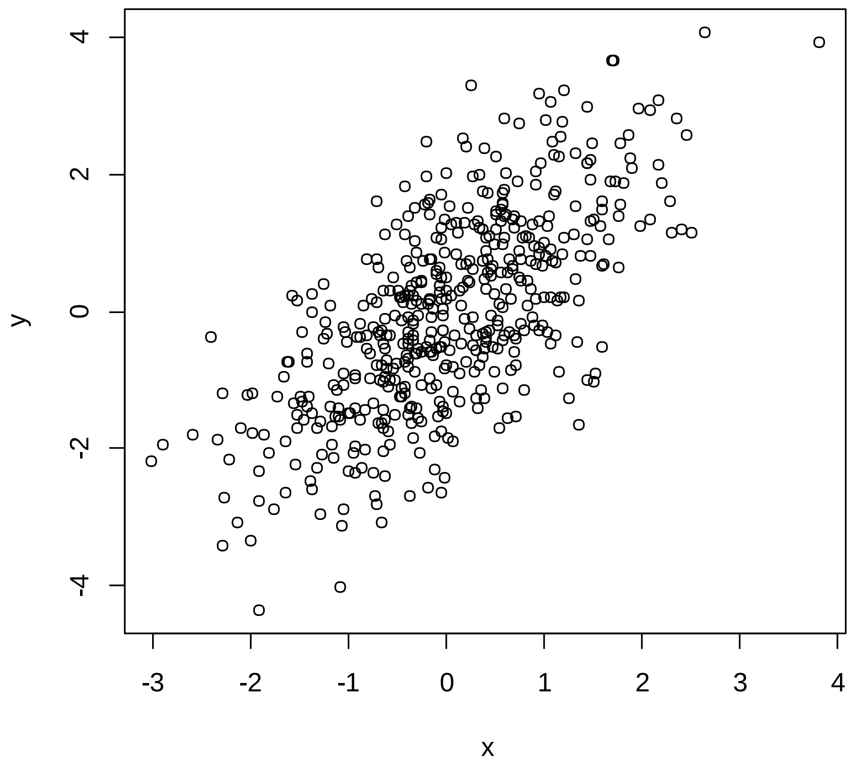
- plot(x, y)
- minor.tick(nx = 3, ny = 3, tick.ratio = 0.5)

- **Does it work?**



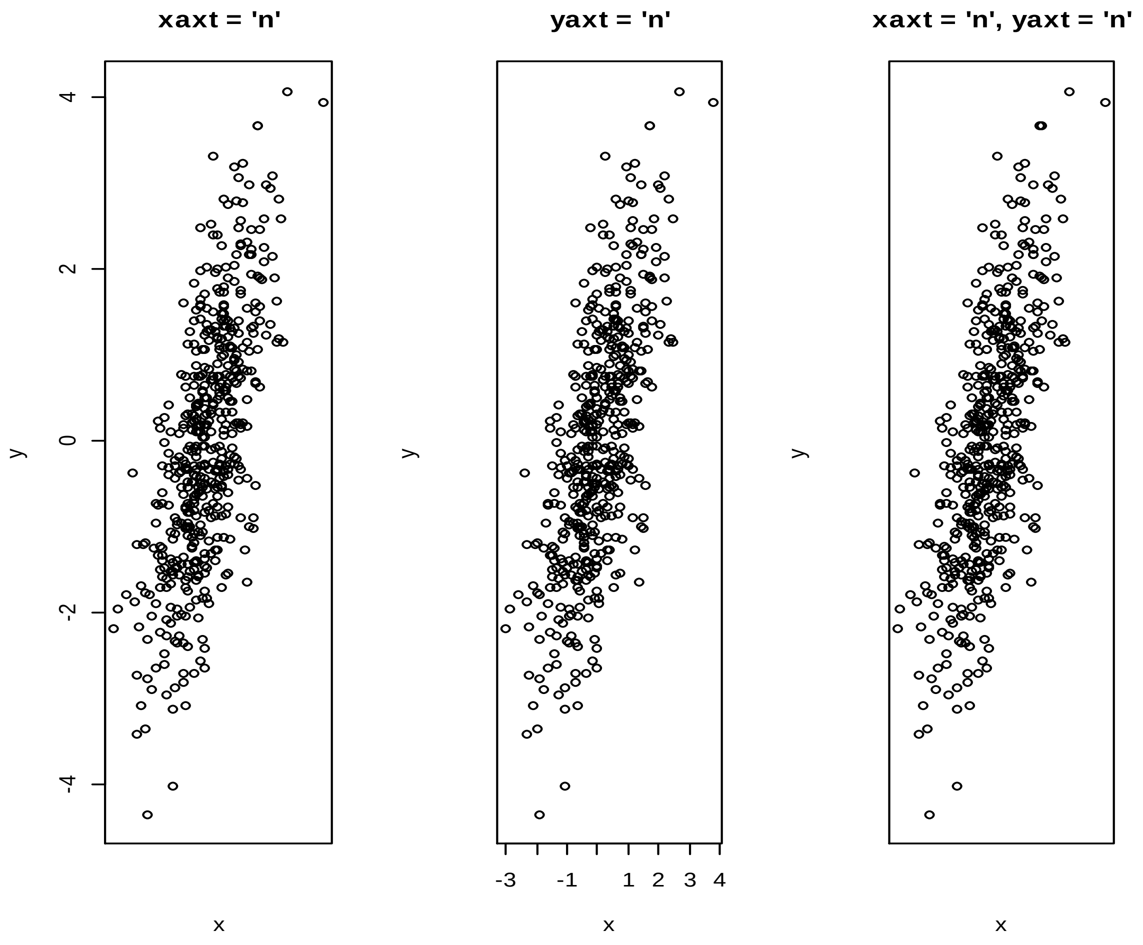
What will happen?

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



What will happen?

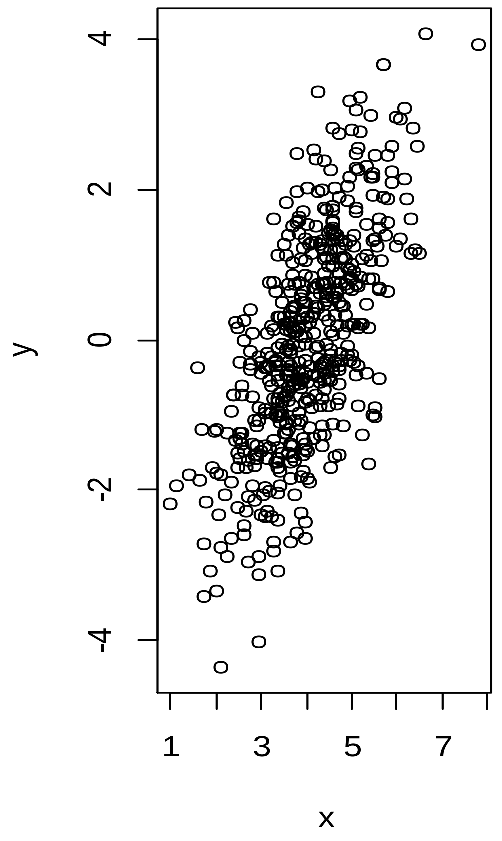
- par(mfrow = c(1, 3))
- # Remove X axis tick labels
- plot(x, y, xaxt = "n", main = "xaxt = 'n'")
- # Remove Y axis tick labels
- plot(x, y, yaxt = "n", main = "yaxt = 'n'")
- # Remove both axis tick labels
- plot(x, y, yaxt = "n", xaxt = "n", main = "xaxt = 'n', yaxt = 'n'")
- par(mfrow = c(1, 1))

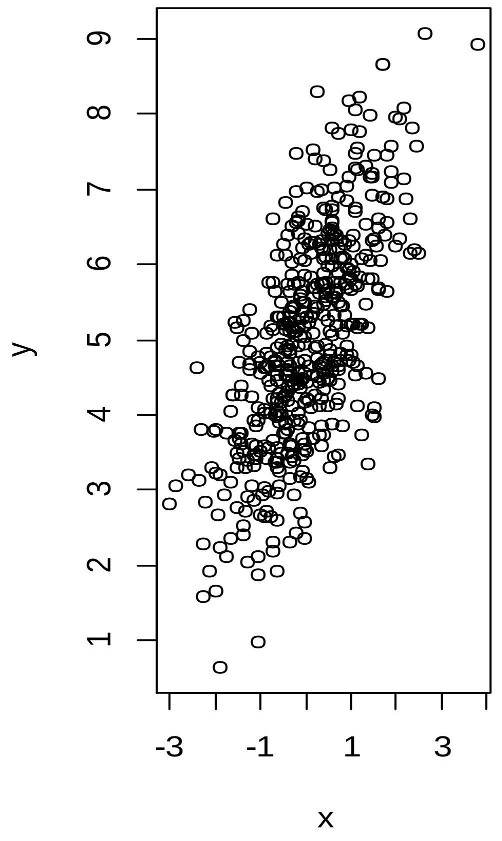


What will happen?

- 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)

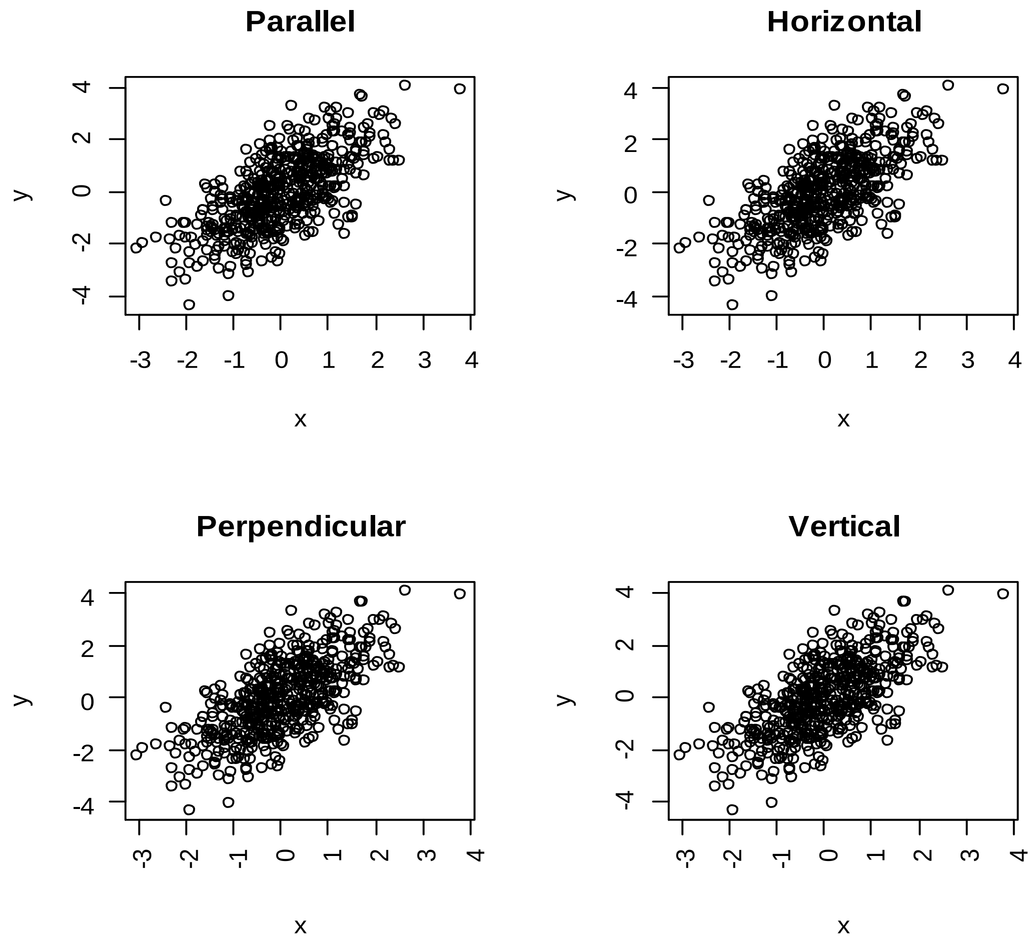
- # Change Y axis tick labels
- plot(x, y, yaxt = "n")
- axis(2, at = seq(round(min(y)), round(max(y)), by = 1), labels = 1:9)
- par(mfrow = c(1, 1))





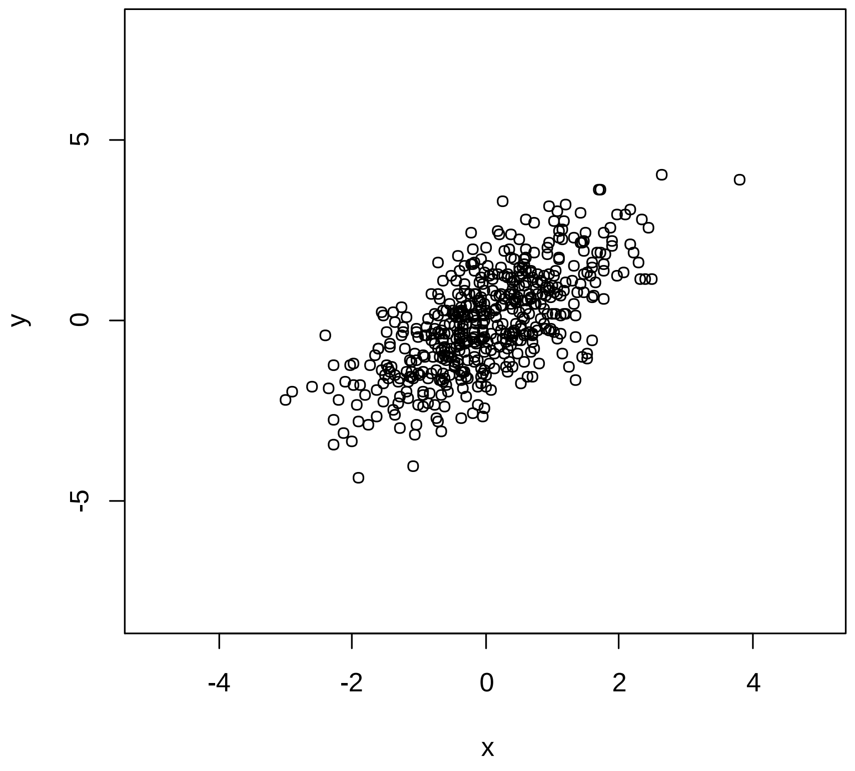
What will happen?

- 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?

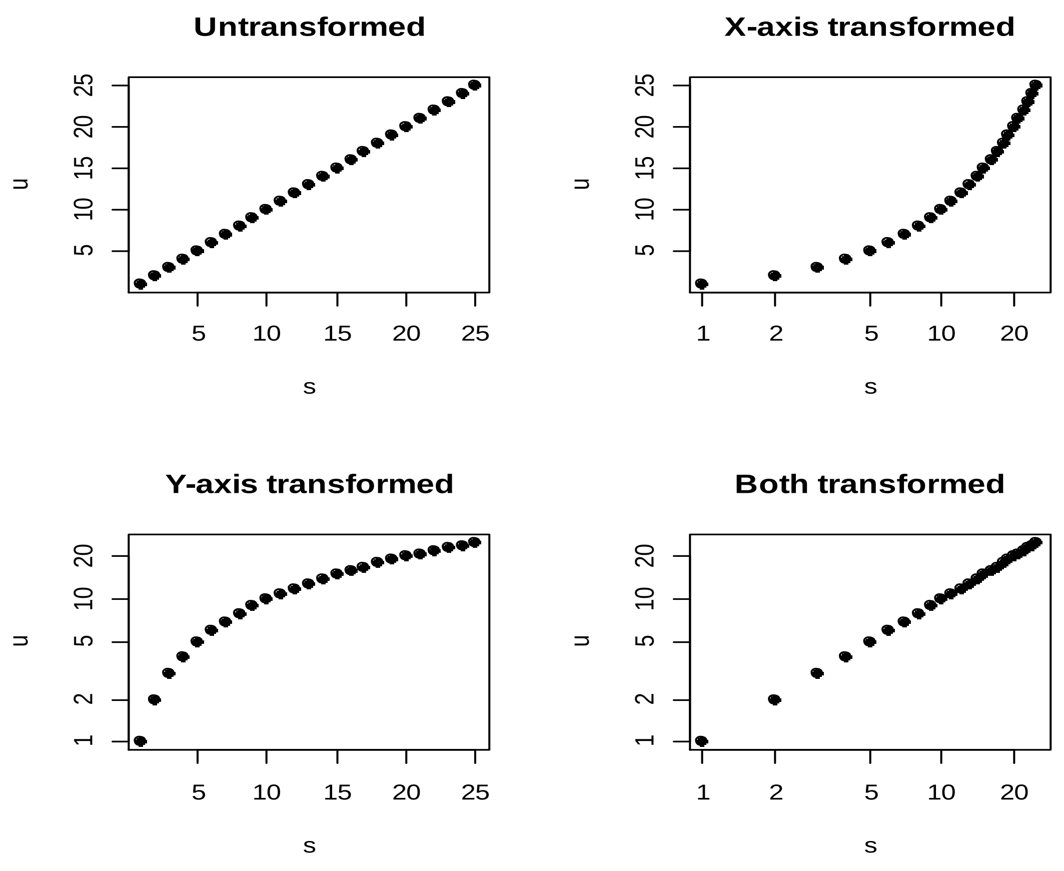
- plot(x, y,
- ylim = c(-8, 8), # Y-axis limits from -8 to 8
- xlim = c(-5, 5)) # X-axis limits from -5 to 5



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))

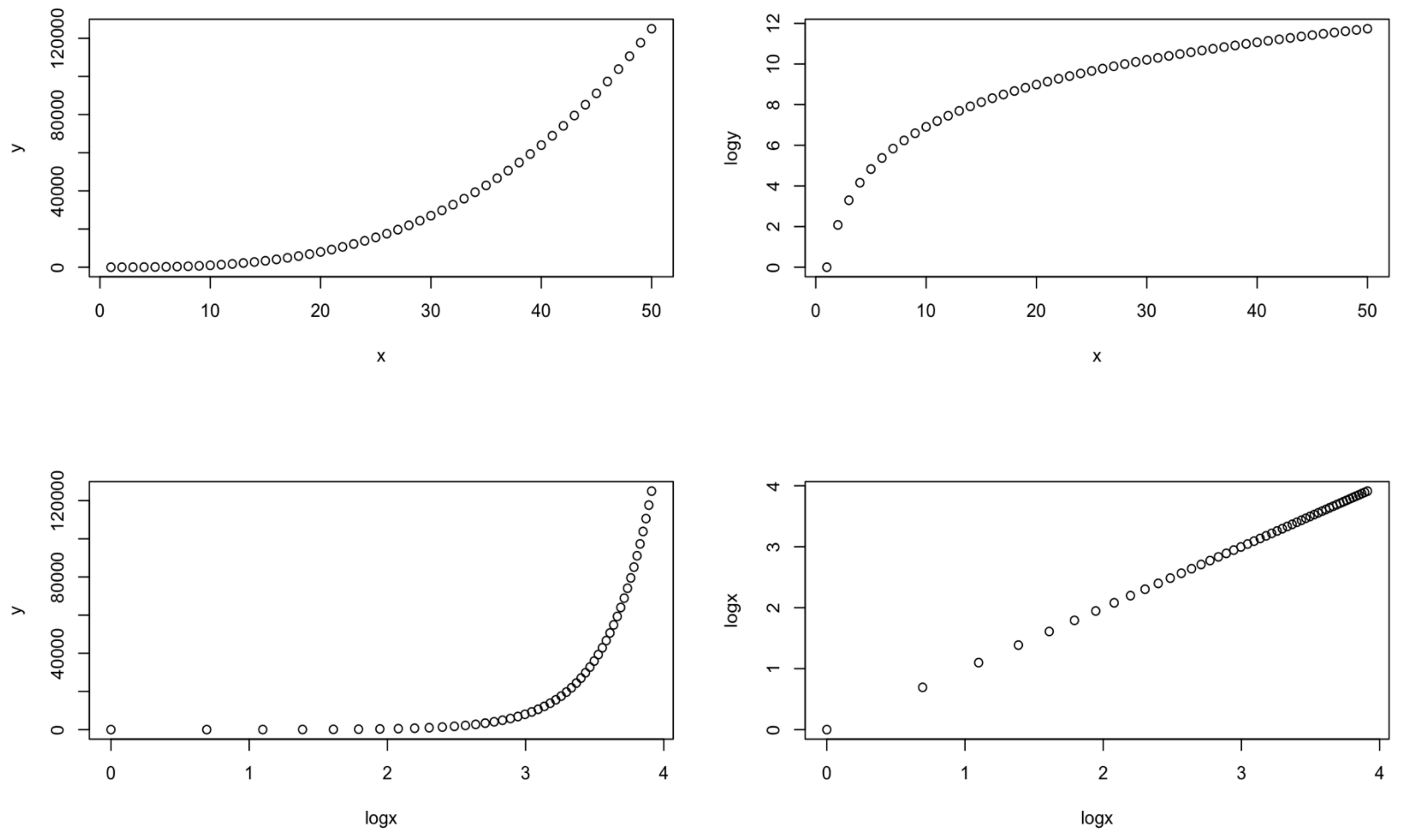


Log transformation of x and y (We will practice it in Unit 4)

- When we get a non-linear relationship then we use Spearman’s correlation coefficient
- However, we will need to use non-linear models to do supervised learning, which is a bit difficult than fitting linear model
- So, we use log-transformation of x and y to make the relationship linear
- We have three options to do so:
- Log-linear: logy ~ x
- Linear-log: y ~ logx
- Log-log: logy ~ logx

Example: logy and logx transformation makes a non-linear to linear!

- x <- 1:50
- y <- x^3
- logx <- log(x)
- logy <- log(y)
- par(mfrow=c(2,2))
- plot(x,y)
- plot(x,logy)
- plot(logx, y)
- plot(logx, logx)
- par(mfrow=c(1,1))



What will happen?

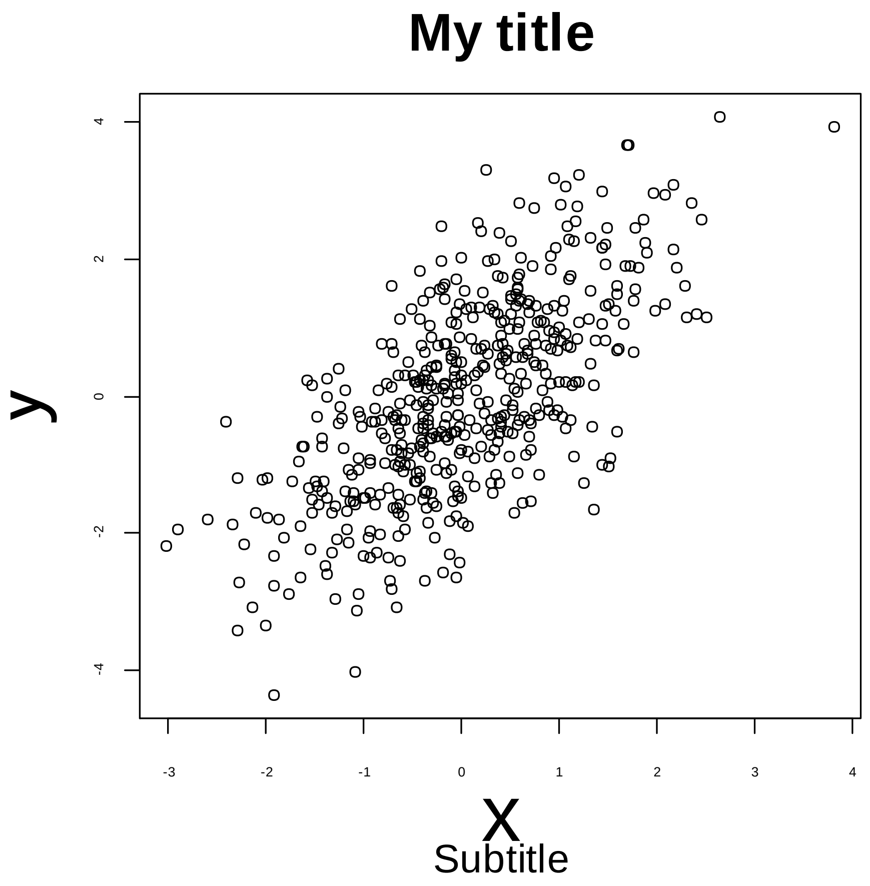
- plot(x, y, main = "My title", sub = "Subtitle",

  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

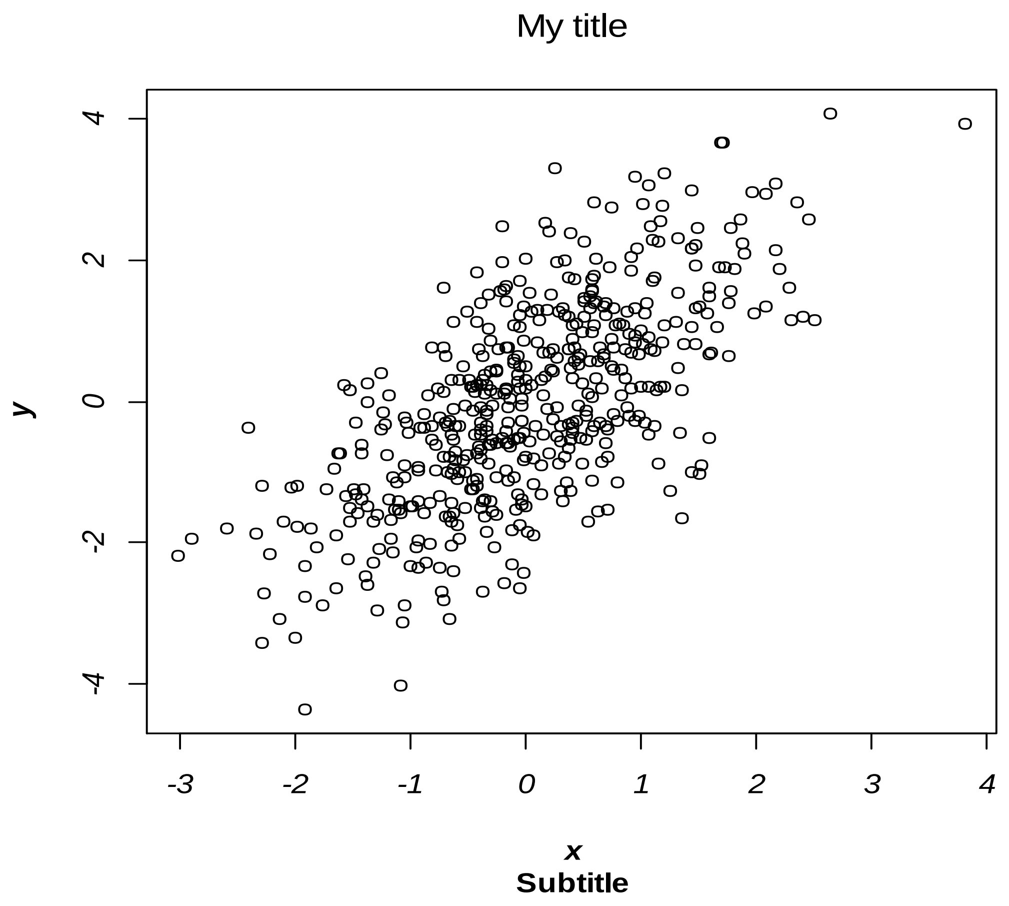


Note:

• 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

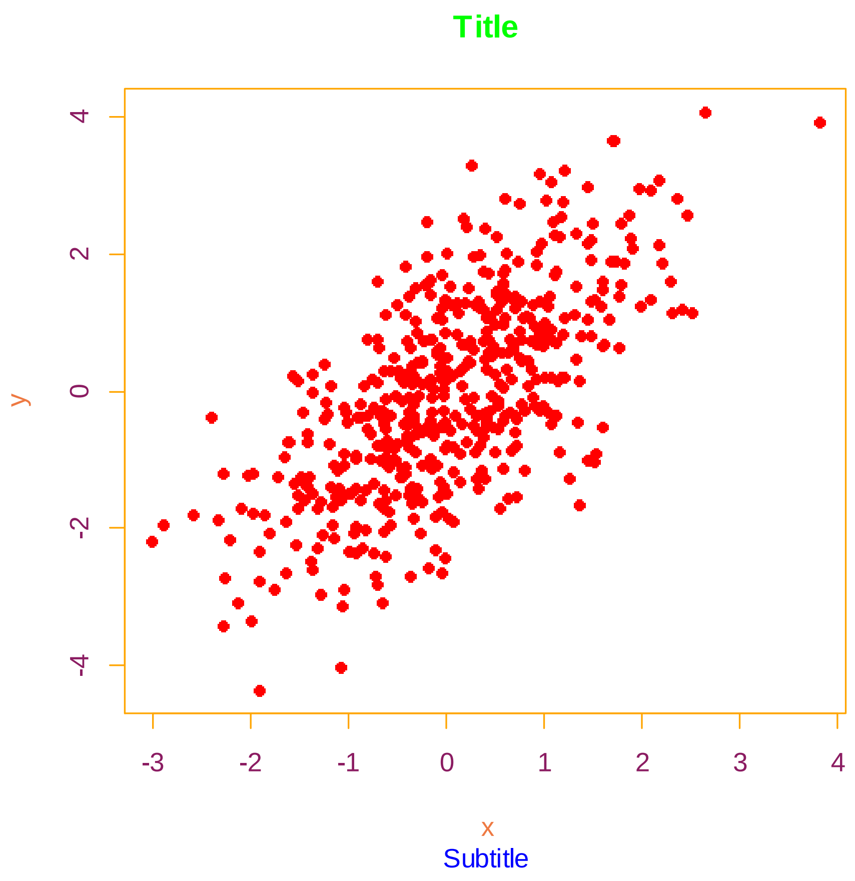
What will happen?

- 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?

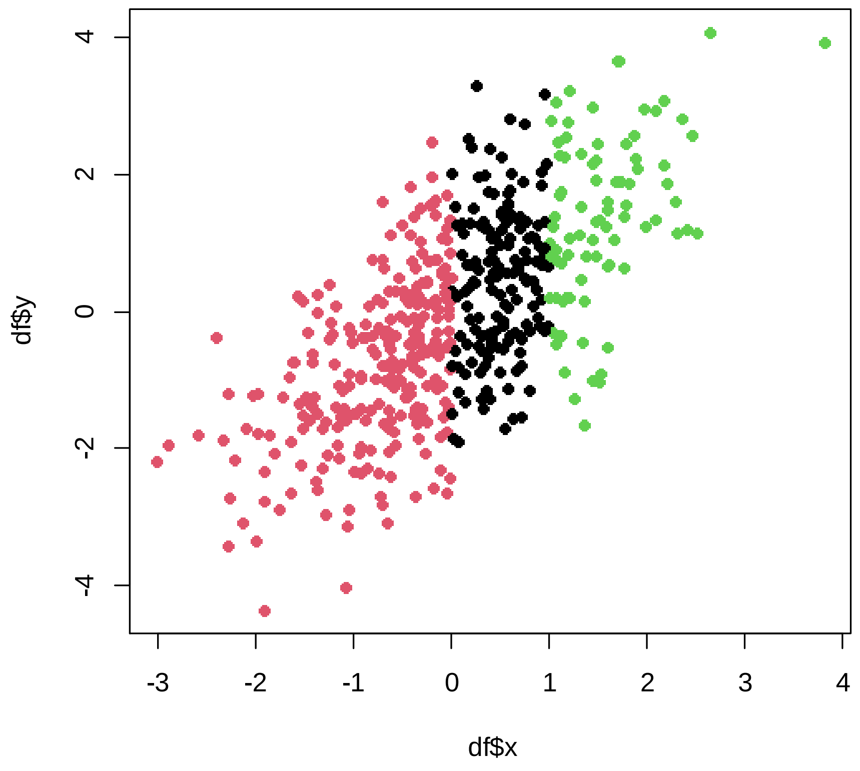
- plot(x, y, main = "Title", sub = "Subtitle",
- pch = 16,
- col = "red",   # Symbol color
- col.main = "green",   # Title color
- col.sub = "blue",   # Subtitle color
- col.lab = "sienna2",   # X and Y-axis labels color
- col.axis = "maroon4",   # Tick labels color
- fg = "orange")   # Box color



What will happen?

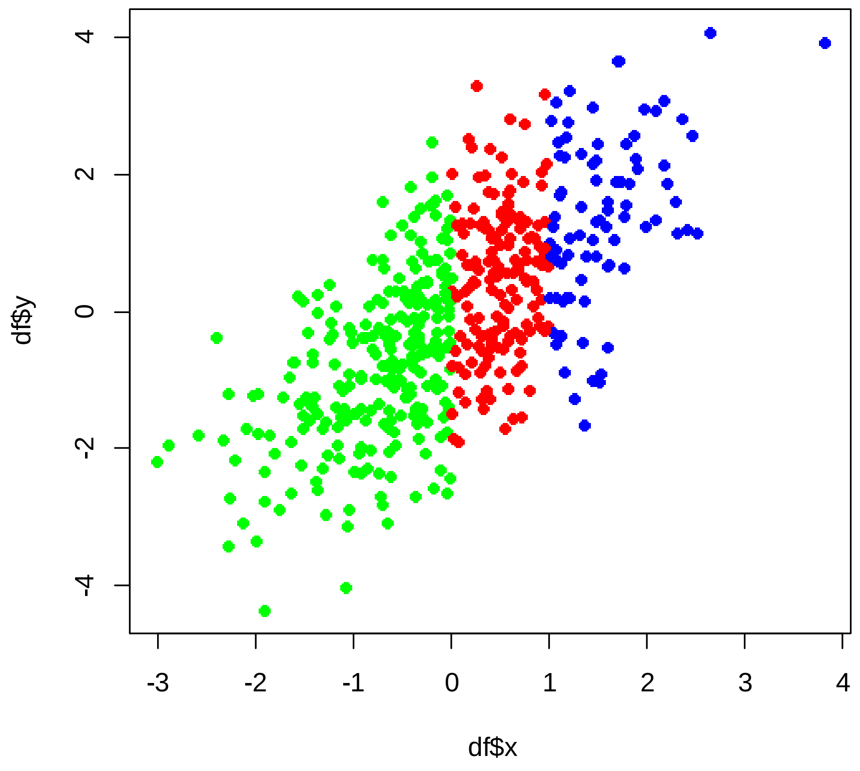
- # Create dataframe with groups
- group <- ifelse(x < 0 , "car", ifelse(x > 1, "plane", "boat"))
- df <- data.frame(x = x, y = y, group = factor(group))

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



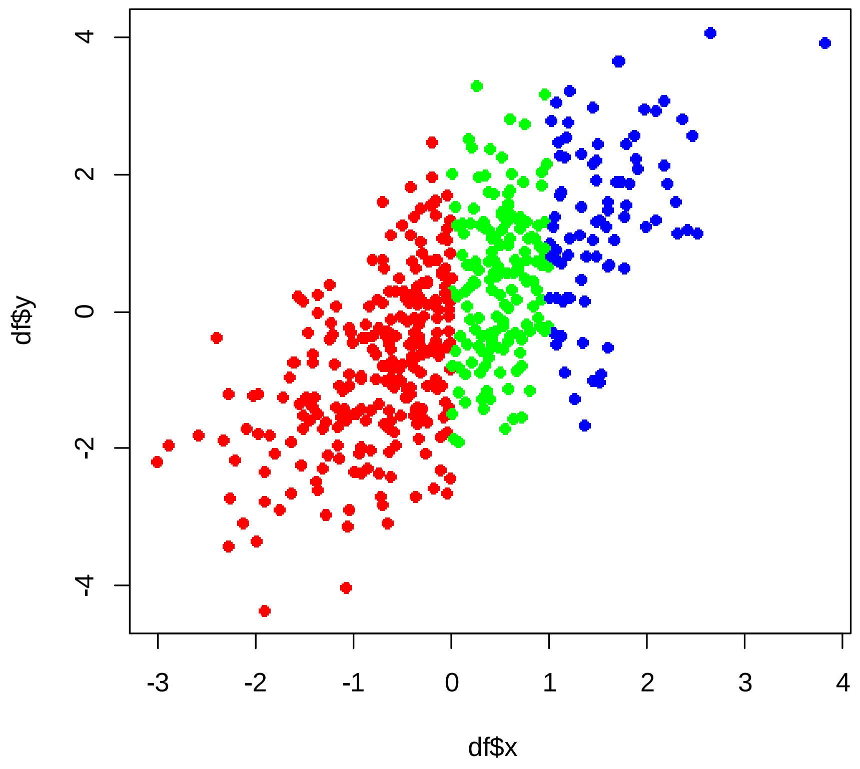
What will happen?

- # 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?

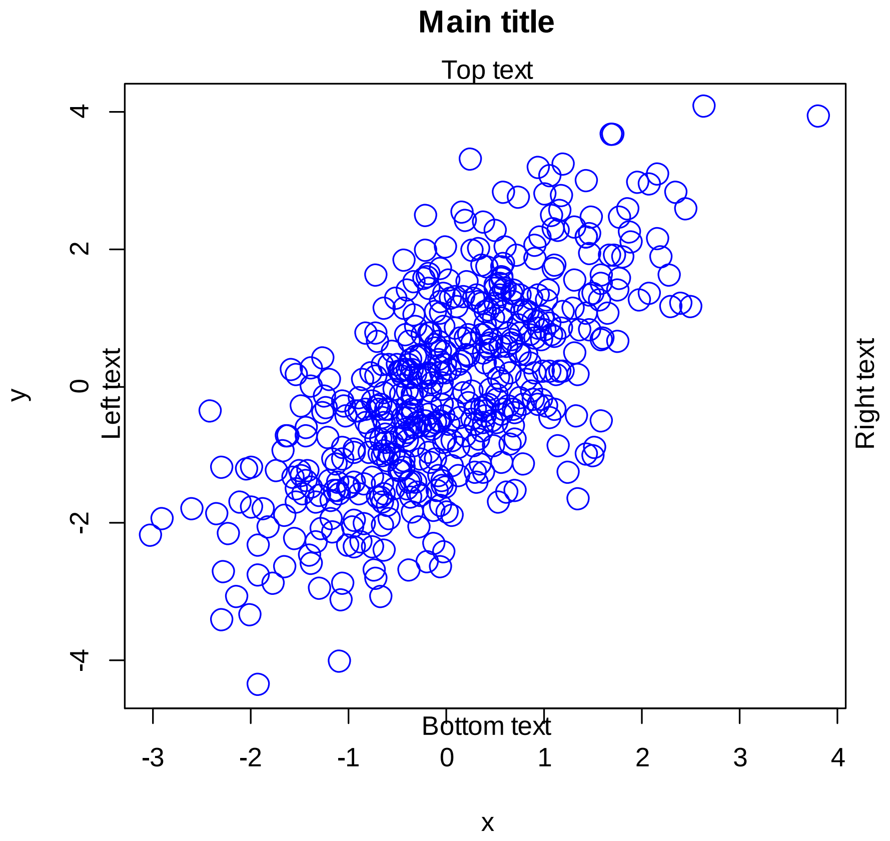
- plot(x, y, main = "Main title", cex = 2, col = "blue")

- # 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)



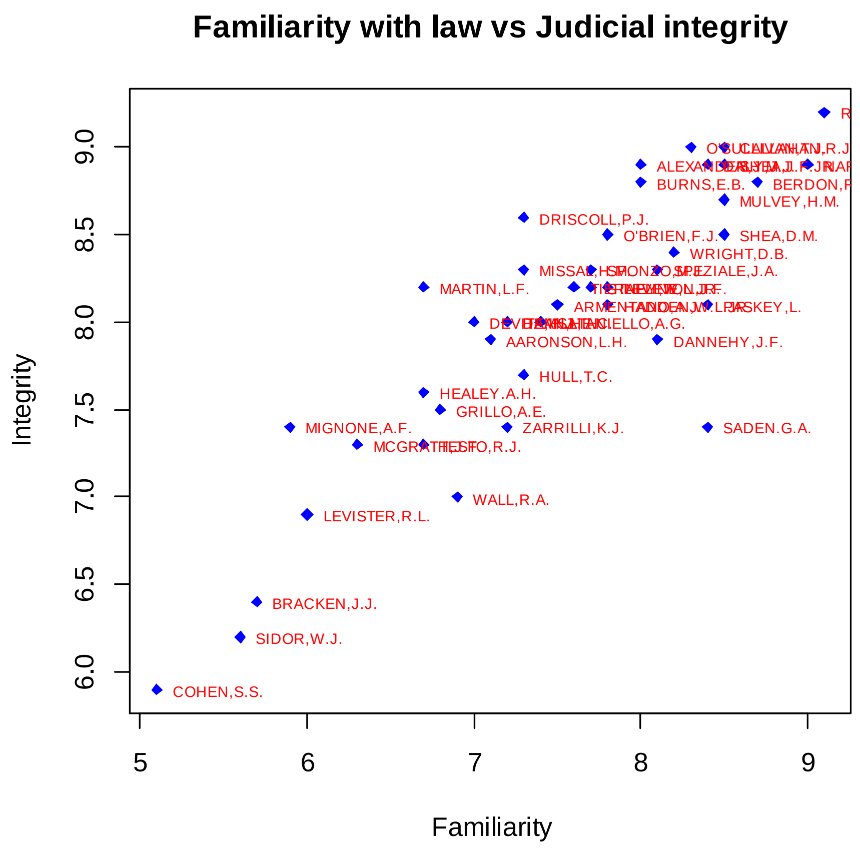
Adding texts from row text variable!

- 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)



Adding selected texts!

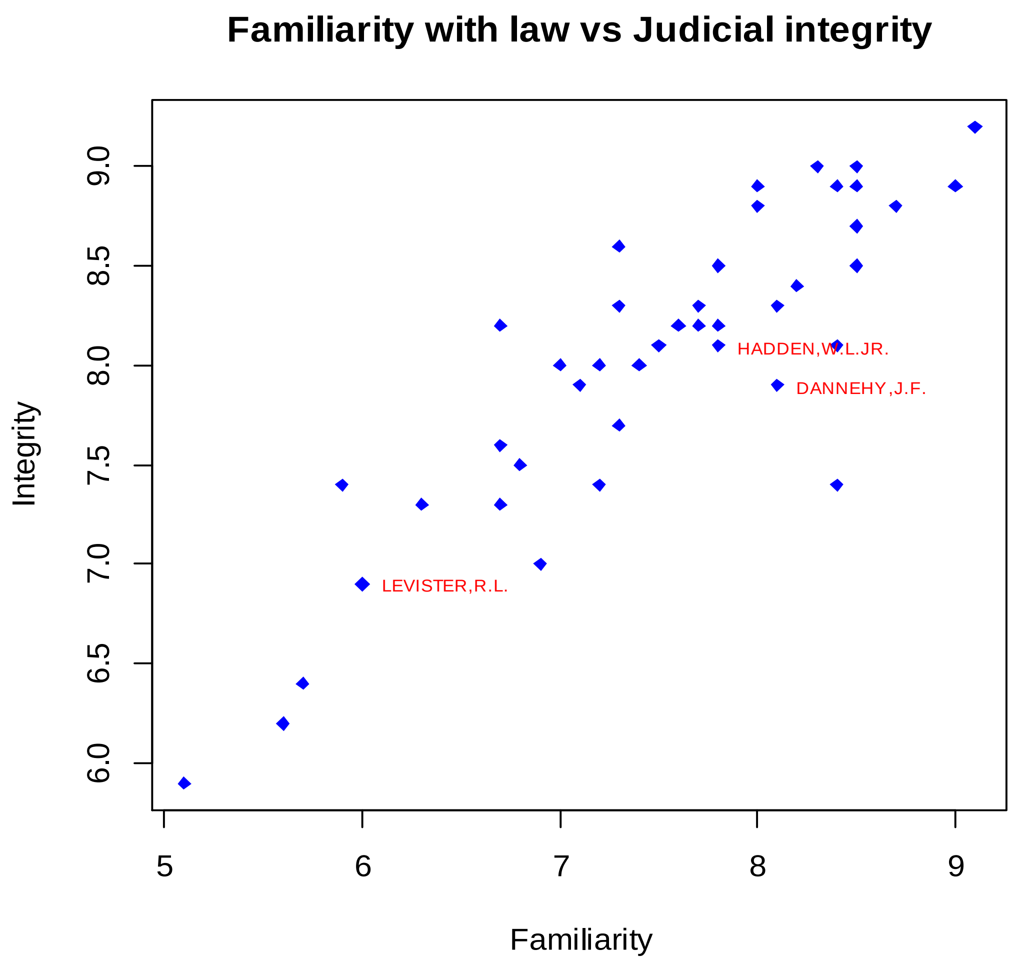
- attach(USJudgeRatings)

- 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)

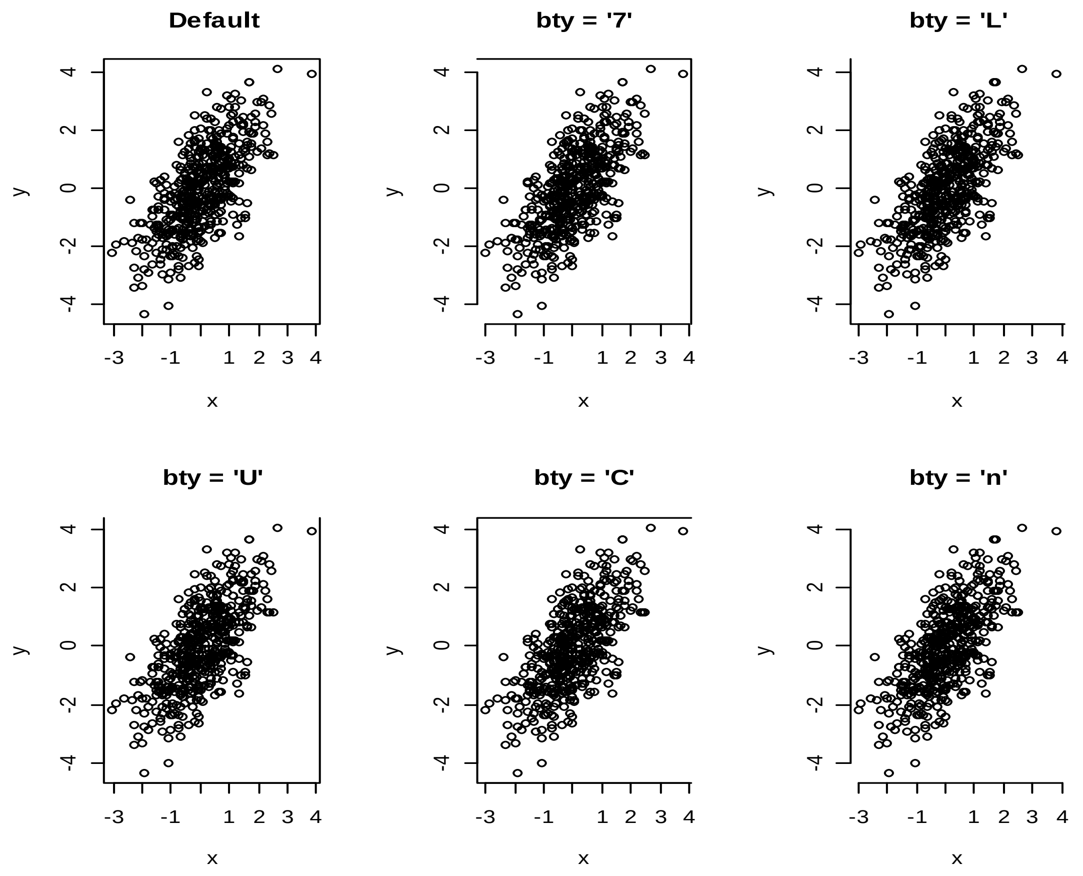


What will happen?

- 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))



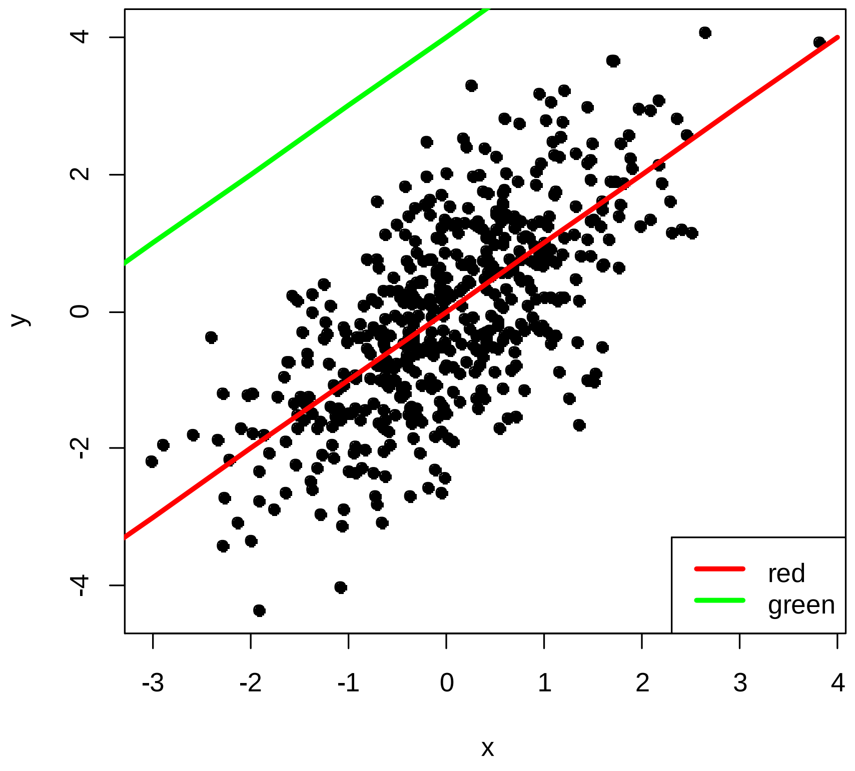
Note: “bty” (inside the par function)

• “o”	• Entire box (default)
• “7”	• Top and right
• “L”	• Left and bottom
• “U”	• Left, bottom and right
• “C”	• Top, left and bottom
• “n”	• 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