# Confounding:

به حالتی گفته می‌شود که دو متغیر با هم همبستگی زیادی دارند و به نظر می‌رسد که یکی از متغیرها باعث تغییر در متغیر دیگر می‌شود ولی چنین نیست و در واقع متغیر سومی وجود دارد که باعث تغیر در هر دو متغیر اول می‌شود.

# R:

## انواع داده‌ها:

* Atomic classes
  + Numeric
  + Logical
  + Character
  + Integer
  + Complex
* Vectors: same class
* Lists: different classes
* Factors: categorical data
* Missing values: NaN and NA
* Data frames
* Names (dimnames for matrices)
* Date  
  date <- as.date(“1995-02-26”)
* Times
  + POSIXct  
    uses one integer to save time since 1970-01-01  
    useful to save time in data frames
  + POSIXlt  
    uses a list to save time

## بارگذاری کد از فایل:

source(fileAddress)

## Subsetting:

### Drop

m <- matrix(1:6, 2, 3)

m[1, ]  
result is a vector

m[1, , drop = FALSE]  
result is a matrix

### Exact

v <- vector(aardvark = 1:4)  
v$a  
result is vector  
v[[“a”]]  
result is NULL  
v[[“a”, exact = FALSE]]  
result is vector

### Merge

a <- (1, 2, NA, NA, 5)  
b <- (6, 7, NA, 9, NA)  
complete.cases(a, b)  
TRUE TRUE FALSE FALSE FALSE

## توابع apply

lapply(x, func, args)

تابع func را روی تک‌تک اعضای x اعمال می‌کند و خروجی به شکل یک لیست در می‌آید.

sapply(x, func, args)

همان کار تابع lapply را انجام می‌دهد ولی در صورتی که بتواند خروجی را به شکل یک ماتریس نمایش دهد، ماتریس نمایش می‌دهد، در غیر این صورت دقیقا مثل lapply عمل می‌کند.

apply(x, margin, func, args)

برای اعمال تابع func روی اعضای x در جهت margin است. اگر margin یک باشد خروجی به طول ردیف‌ها و اگر دو باشد به طول ستون‌ها خواهد بود.

mapply(func, …, MoreArgs = NULL, SIMPLIFY = TRUE, USE.NAMES = TRUE)

تابع func را روی چندین لیست اعمال می‌کند.

tapply(x, index, func, …, simplify = TRUE)

مانند sapply عمل می‌کند ولی می‌توان با index که طول یکسانی با x دارد اعضایی را که باید تابع روی آن‌ها اعمال شود را کنترل کرد.

با تابع gl می‌توان دسته‌بندی کرد.

vapply(x, func, …)

ممکن است خروجی sapply لیست‌هایی به طول یکسان نباشند و این تابع مثل lapply عمل کند. اگر از vapply استفاده کنیم درصورتی که این اتفاق بیافتد error نمایش داده می‌شود.

split(x, f, drop = FALSE, …)

قدم اول tapply را انجام می‌دهد. یعنی x را به دسته‌های جدا تقسیم می‌کند.

split و data frameها: split(df, df$colName)

تابع interaction دو دسته را مخلوط می‌کند. اگر دو دسته را به صورت یک لیست به split بدهیم این کار را انجام می‌دهد. با آرگومان drop = TRUE می‌توان دسته‌هایی که خالی هستند را حذف کنیم.

## توزیع‌های احتمالاتی

همه توزیع‌های استاندارد در R وجود دارند. برای استفاده از آن‌ها از چهار پیشوند استفاده می‌شود. r (random) برای تولید اعداد تصادفی، d (density) برای تابع چگالی، p (probability) برای تابع تجمعی و q (quantile) برای تابع quantile استفاده می‌شود.

## دانلود داده

برای دانلود داده‌ها از download.file(url, destfile, method) استفاده می‌کنیم. بهتر است تاریخ دانلود را نیز نگه داریم.

## Excel

پکیج‌های xlsx و XLConnect مناسب هستند.

## data.table

dt[, sum(x)] # sum of column x

dt[, sum(x+y), by=a] # sum of columns x and y grouped by column a

dt[, y := 2\*x] # adding new column

dt[, .N, by=a] # count of observations gouped by column a

setkey(dt, k) # dt[sth] will be dt[, k=sth]

setkey(d2, k) # merge(dt, d2) will join the tables

## Cross tabs

xtabs(Freq ~ Gender + Admit, data=dataFrame)

## Flat tables

ftable(xtabs(~. , data=dataFame))

Reshape

library(reshape2)  
melted <- melt(dataFrame, id=c(col1, col2), measure.vars=c(col3, col4))  
col2Data <- dcast(melted, col2 ~ variable) # frequency of variables for each col2  
col2Data <- dcast(melted, col2 ~ variable, mean) # mean of variables for each col2

## dplyr

select(df, col1:col3)  
select(df, -(col1:col3))

filter(df, predicate)

arrange(df, col1)  
arrange(df, desc(col2))

rename(df, ncol1 = col1)

mutate(df, ncol = col1 + col2)

group\_by(df, col1)

df %>% mutate(ncol1 = col1+col2) %>% group\_by(col3) %>% summarize(sumcol1 = ncol1 / col2)

merge(df1, df2, by.x, by.y, all)

join(df1, df2, col1)

join\_all(list(df1, df2, df3))

## Base Plot

hist(df$col1) # plot a histogram

rug(df$clo1) # plot grayscale 1D to show density

with(df, plot(col1, col2)) # plot a scatter plot

boxplot(col1 ~ col2, df) # plot a boxplot

تعدادی از آرگومان‌های مهم

pch: the plotting symbol

lty: line type

lwd: line width

col: color

xlab: x-axis label – ylab: y-axis label

از par() برای مشخص کردن متغیرهای global استفاده می‌شود. تعدادی از آرگومان‌های مهم

las: orientation of the axis labels on the labels

bg: background color

mar: margin size

oma: outer margin size

mfrow: number of plots per row – mfcol: number of plots per column

تعدادی از توابع مهم

plot: make scatter plot or other type of plot depending on the class of data

line: add lines – points: add points

text: add text using x, y coordinates

title: add annotations to x, y labels, title, subtitle, outer margin

mtext: add text to margins

axis: add axis ticks and labels

توضیحات بیشتر

plot(type = “n”) # plot a scatter plot without it’s points   
with(subset(), points(col=”red”)) # add class 1 points in red   
with(subset(), points(col=”blue”)) # add class 2 points in blue

legend() # add legend

model <- lm(col1 ~ col2, df) # fit regression line  
abline(model) # add regression line

par(mfrow = (1, 2))  
with(df, {plot(), plot()})

اگر تعداد نمونه‌ها زیاد باشد می‌توان دو کار کرد. یا تعدادی از نمونه‌ها را انتخاب کرد و نمایش داد. یا از smoothScatter استفاده کرد.

## Lattice

xyplot: scatterplot

bwplot: box-and-whiskers plot

histogram

stripplot: like boxplot with actual points

dotplot: plot dots on violin strings

splom: scatterplot matrix

levelplotc, contourplot: image plot

## ggplot2

qplot(df.col1, cf.col2) # scatterplot  
qplot(…, geom = c(“point”, “smooth”)) # add geom  
qplot(…, color = col3) # color with respect to col3 value

qplot(df.col1) # histogram  
qplot(, fill = col3) # color with respect to col3 value

geom = “density”

### Facets

qplot(…, facets = col1 ~ col2) # create separate plots with respect to values of col1 in different rows and col2 in different columns… use . to create one plot

در ggplot2 هر plot از data frame، aesthetic mapping، geoms، facets، stats، scales و coordinate system تشکیل شده است. می‌توان به جای استفاده از qplot هرکدام از این لایه‌ها را جداگانه ساخت و به هم اضافه کرد.

qplot() + geom\_point() + facet\_grid(. ~ col1) + geom\_smooth(method = “lm”)

## رنگ

colors() # list of color names in R

pal <- colorRamp(c(“red”, “blue”))  
pal(a) # 0 <= a <= 1

pal <- colorRampPalette(c(“red”, “blue”))  
pal(a) # a should be integer. It will return list of a colors.

### RColorBrewer

این بسته سه دسته‌بندی از رنگ‌ها را دارد. Sequential، diverging و qualitative که به ترتیب برای نمایش داده‌هایی دارای ترتیب، داده‌هایی که مرتب و دارای مقادیر مثبت و منفی (یا کمتر از آستانه و بیشتر از آستانه) هستند و داده‌های بدون ترتیب (مانند جنسیت) استفاده می‌شوند.

cols <- brewer.pal(3, colorset)  
pal <- colorRampPalette(cols)  
# then use pal

colorspace یک بسته مناسب برای کار با انواع نمایش رنگ‌ها است.

## Hierarchical Clustering

distxy <- dist(df)  
hclustering <- hclust(disxy)  
plot(hclustering) # this will plot a dendrogram

dm <- as.matrix(df)  
heatmap(dm)

## K-Means

kmeans(df, centers = k)

## Principal Components Analysis and Singular Value Decomposition

تقریب زدن داده اصلی با استفاده از یک مؤلفه اصلی

approx1 <- svd$u[, 1] %\*% svd$v[, 1] \* svd$d[1]

تقریب زدن داده اصلی با استفاده از k مؤلفه اصلی

Approxk <- svd$u[, k] %\*% diag(svd$d[1:k]) %\*% t(svd$v[1:k])