**Subsetting and Sorting**

* **Quick review**

X <- data.frame

X <- x[sample(1:5), ]

X[, 1]

X[, “var1”]

X[1:2, “var2”]

* **Logical ands and ors**
* **Dealing with missing values**

X[which(X$var2 > 8, ]

* **Sorting**

Sort(x$var1)

Sort(x$var1, decreasing = TRUE)

Sort(x$var1, na.last = TRUE)

* **Ordering**

X[order(x$var1. X$var3), ]

* **Ordering with plyr**

Library(plyr)

Arrange(x, var1)

Arrange(x, desc(var1))

* **Adding rows and columns**

X$var4 <- rnorm(5)

Y <- cbind(x, rnorm(5))

**Summarizing data**

* **Example dataset**
* **Getting the data from the web**

If(!file.exists(“./data”)) {

Dir.create(“./data”)

}

fileUrl <- <https://vudoanthe1910>

download.file(fileUrl, destfile = “./data/restaurant.csv”, method = “curl”)

restData <- read.csv(“./data/restaurant.csv”)

* **1st impression**

Head(restData, n = 3)

Tail(restData, n = 3)

Summary(restData)

* **More in depth info**

Str(restData)

* **Quantiles of quantitative variables**

Quantile(restData$councilDistrict, na.rm = TRUE)

Quantile(restData$councilDistrict, probs = c(0.5, 0.75, 0.9))

* **Make tables**

Table(restData$zipCode, useNA = “ifany”)

Table(restData$zipCode, restData$councilDistrict)

* **Check for missing values**

Sum(is.na(restData$councilDistrict))

Any(is.na(restData$councilDistrict))

All(restData$zipCode > 0)

* **Row and column sums**

colSums(is.na(restData))

all(colSums(is.na(restData)) == 0)

* **Values with specific characteristics**

Table(restData$zipCode %in% c(“21212”))

Table(restData$zipCode %in% c(“21212”, “21213”))

restData[restData$zipCode %in% c(“21212”, “21213”), ]

* **Cross tabs**

Data(UCBAdmissions)

DF = as.data.frame(UCBAdmissions)

Summary(DF)

xt <- xtabs(Freq ~ Gender + Admit, data = DF)

* **Flat tables**

Warpbreaks$replicate <- rep(1:9, len = 54)

Xt = xtabs(breaks ~., data = warpbreaks)

Xt

Ftable(xt)

* **Size of a data set**

Object.size(data)

**Creating new variables**

* **Why?**
  + - Often the raw data do not have a value you looking for
    - Transform data to get what you like
    - Common variables to create
      * Missingness indicators
      * “cutting up” quantitative variables
      * Applying transforms
* **Example dataset**

If(!file.exists(“./data”)) {

Dir.create(“./data”)

}

fileUrl <- <https://vudoanthe1910>

download.file(fileUrl, destfile = “./data/restaurant.csv”, method = “curl”)

restData <- read.csv(“./data/restaurant.csv”)

* **Creating sequences**

s1 <- seq(1, 10, by = 2)

* **Subsetting variables**

restData$nearMe = restData$neighborhood %in% c(“Roland Park”, “Homeland”)

table(restData$nearMe)

* **Creating binary variables**

restData$zipWrong = ifelse(restData$zipCode < 0, TRUE, FALSE)

table(restData$zipWrong, restData$zipCode < 0)

* **Creating categorical variables**

restData$zipGroups = cut(restData$zipCode, breaks = quantile(restData$zipCode)

table(restData$zipGroups)

table(restData$zipGroups, restData$zipCode)

* **Easier cutting**

Library(Hmisc)

restData$zipGroups = cut2(restData$zipCode, g = 4)

table(restData$zipGroups)

* **Creating factor variables**

restData$zcf <- factor(restData$zipCode)

restData$zcf[1:10]

class(restData$zcf)

* **Levels of factor variables**

Yesno <- sample(c(“yes”, “no”), size = 10, replace = TRUE)

Yesnofac = factor(yesno, levels = c(“yes”, “no”)

Relevel(yesnofac, ref = “yes”)

As.numeric(yesnofac)

* **Cutting produces factor variables**

Library(Hmisc)

restData$zipGroups = cut2(restData$zipCode, g = 4)

table(restData$zipGroups)

* **Using the mutate function**

Library(Hmisc), library(plyr)\_

restData2 = mutate(restData, zipGroups = cut2(zipCode, g = 4)

table(restData2$zipGroups)

**Reshaping Data**

* **The goal is tidy data**
  + - Each variable forms a column
* **Start with reshaping**

Library(reshape2)

Head(mtcars)

* **Melting data frames**

Mtcars$carname <- row.names(mtcars)

carMelt <- melt(mtcars, id = c(“carname”, “gear”, “cyl”)

measure.vars = c(“mpg”, “hp”)

head(carMelt, n = 3)

* **Casting data frames**

cylData <- dcast(carMelt, cyl ~ variable)

cylData

cylData <- dcast(carMelt, cyl ~ variable, mean)

cylData

* **Averaging values**

Head(InsectSprays)

tapply(InsectSprays$count, InsectSprays$spray, sum)

* **Another way – split, lapply, combine**

spIns = split(InsectSprays$count, InsectSprays$spray)

spIns

sprCount = lapply(spIns, sum)

sprCount

* **Another way – combine**

unlist(sprCount)

sapply(spIns, sum)

* **Another way – plyr package**

ddply(InsectSprays, .(spray), summarize, sum = ave(count, FUN = sum)

dim(spraySums)

head(spraySums)

* **More?**

acast

arrange

mutate

**Managing data frames with dplyr – Introduction**

* **dplyr** 
  + - One observation per row
    - Each column represents a variable or measure or characteristic
* **Basics**
  + - Select
    - Filter
    - Arrange
    - Rename
    - Mutate
    - Summarize
* **Properties**
  + - The 1st argument is a data frame
    - The coming args describe what to do with it’
    - The new result is a new data frame
    - Data frames must be formatted and annotated for dplyr to be useful

**Managing data frames with dplyr – Basics**

* **Load the package**

Library(dplyr)

* **Select function**

Chicago <- readRDS(“Chicago.rds”)

Dim(Chicago)

Str(Chicago)

Names(Chicago)

Head(select(Chicago, city:dptp))

Head(select(Chicago, -(city:dptp)))

* **Filter function**

Chic.f <- filter(Chicago, pm25tmean2 > 30)

Head(chic.f, 10)

Chic.f <- filter(Chicago, pm25tmean2 > 30 & tpmd > 80)

* **Arrange function**

Chicago <- arrange(Chicago, date)

Head(Chicago)

Chicago <- arrange(Chicago, desc(date))

Head(Chicago)

* **Rename function**

Chicago <- rename(Chicago, pm25 = pm25tmean2, dewpoint = dptp)

Head(Chicago)

* **Mutate function**

Chicago <- mutate(Chicago, pm25detrend = pm25-mean(pm25, na.rm = TRUE)

Head(select(Chicago, pm25, pm25detrend))

* **Summarize function**

Chicago <- mutate(Chicago, tempcat = factor(1 \* (tmpd > 80), labels = c(“cold”, “hot”))

Hotcold <- group\_by(Chicago, tempcat)

Hotcold

Summarize(hotcold, pm25 = mean(pm25), o3 = max(o3tmean2), no2 = median(no2tmean2))

**Merging data**

dfList = list(df1, df2, df3)

join\_all(dfList)