R Cheat Sheet: Data Frames (tabular data in rows and columns)

Create Set column names # same for rownames()
- The R way of doing spreadsheets colnames(df) <- c('date', 'alpha', 'beta')
- Internally, a data.frame is a list of colnames(df)[1] <- 'new.name.for.col.1'
equal length vectors or factors. colnames(df)[colnames(df) %in% c('a', 'b')]
- Observations in rows; Variables in cols <- c('x', 'y') # order of sub from cols
empty <- data.frame()# empty data frame
c1 <- 1:10 # vector of integers Selecting multiple rows
c2 <- letters[1:10] # vector of strings firstTenRows <- df[1:10,] # head(df, 10)
df <- data.frame(col1=c1,col2=c2) everythingButRowTwo <- df[-2,]
sub <- df[(df$x > 5 & df$y < 5),]
Import from and export to file sub <- subset(df, x > 5 & y < 5)
d2 <- read.csv('fileName.csv', header=TRUE) # Note: vector Boolean (&, |) in above
library(gdata); d3 <- read.xls('file.xls') notLastRow <- head(df, -1) # df[-nrow(df),]
write.csv(df, file='fileName.csv') # export
print(xtable(df), type = "html") # to HTML Selecting multiple columns
df <- df[, c(1, 2, 3)] # keep cols 1 2 3
Basic information about the data frame df <- df[, c('col1', 'col3')] # by name
Function Returns # drop columns ...
is.data.frame(df) TRUE df <- df[, -1] # keep all but first column
class(df) "data.frame" df <- df[, -c(1, 3)] # drop cols 1 and 3
nrow(df); ncol(df) Row and Col counts df <- df[, !(colnames(df) %in%
colnames(df); NULL or char vector c('notThis', 'norThis'))]# drop by name
rownames(df) NULL or char vector
Also: head(df); tail(df); summary(df) Replace column elements by row selection
df[df$col3 == 'A', 'col2'] <-

Referencing cells [row, col] [[r, c]] c('j', 'a', 'a', 'a', 'j')
# [[for single cell selection; [for multi
vec <- df[[5, 2]] # get cell by row/col num Manipulation
newDF <- df[1:5, 1:2] # get multi in new df sorted <- df[order(df$col2),]
df[[2, 'col1']] <- 12 # set single cell backwards <- df[rev(order(df$col2)),]
df[3:5, c('col1', 'col2')] <- 9 # set multi transposed <- as.data.frame(t(df))
merged <- merge(df1,df2,by='col',all=TRUE)

Referencing rows [r,] molten <- melt(df, id=c('year', 'month'),
# returns a data frame (and not a vector!) measure=c('col5', 'col10', col'15))
row.1 <- df[1,]; row.n <- df[nrow(df),] # Note: melt comes from the reshape package
# to get a row as a vector, use following rownames(df)<- seq_len(nrow(df))#renum rows
vrow <- as.numeric(as.vector(df[row,])) summary <- ddply(df, ~col1, summarise,
vrow <- as.character(as.vector(df[row,])) N=length(col3). mean=mean(col3))
summary <- ddply(df, .(col1, col2),

Referencing columns [,c] [c] [[c]] $col summarise, sd=sd(col3) mean=mean(col3))
# most column references return a vector # Note: ddply comes from the plyr package
col.vec <- df$cats # returns a vector
col.vec <- df[, 'horses'] # returns vector Missing data (NA)
col.vec <- df[, a] # a is int or string any(is.na(df)) # detect anywhere in df
col.vec <- df[['frogs']] # returns a vector any(is.na(df$col)) # anywhere in col
frogs.df <- df['frogs'] # returns 1 col df # delete selected missing data rows
first.df <- df[1] # returns 1 col df df <- df[!is.na(df$col),]
first.col <- df[, 1] # returns a vector # replace NAs with something else
last.col <- df[, ncol(df)] # returns vector df[is.na(df)] <- 0 # works on whole df
df$col[is.na(df$col)] <- newValue

Adding rows df$col <- ifelse(is.na(df$col), 0, df$col)
# The right way ... (both args are DFs) df <- orig[!is.na(orig$series),
df <- rbind(df, data.frame(col1='d', c('Date', series)] # selecting on r & c
col2=3, col3='A'))
Traps

Adding columns 1 for loops on possibly empty df's, use:
df$newCol <- rep(NA, nrow(df)) # NA column for(i in seq_len(nrow(df))
df[, 'copyOfCol'] <- df$col # copy a col 2 columns coerced to factors, avoid with
df$y.percent.of.x <- df$y / sum(df$x) * 100 the argument stringsAsFactors=FALSE
df <- cbind(col, df); df <- cbind(df, col) 3 confusing row numbers and rows with
df$c3 <- with(df, c1 + c2) # no quotes numbered names (hint: avoid row names)
transform(df, col3 = col1 * col2) 4 although rbind() accepts vectors and
df <- within(df, colC <- colA + colB) lists; this can fail with factor cols