R Cheat Sheet: Data Frames

Create a matrix of data in rows and cols

```
# A list of equal length vectors/factors
# Variables in columns Observations in rows
empty <- data.frame() # an empty data frame
c1 <- 1:10          # vector of integers
c2 <- letters[1:10] # vector of strings
c3 <- factor(rep(LETTERS[1:2], 5))
df <- data.frame(col1=c1, col2=c2, col3=c3,
    stringsAsFactors=FALSE) # example data
```

Import and export to file

```
d2 <- read.csv('fileName.csv', header=TRUE)
library(gdata); d3 <- read.xls('file.xls')
write.csv(df, file='fileName.csv') # export
print(xtable(df), type = "html") # to HTML
```

Basic information about the data frame

```
is.data.frame(df) # -> TRUE or FALSE
mode(df) # -> "list" # same for typeof(df)
class(df) # -> "data.frame"
nrow(df); ncol(df) # -> row & col counts
colnames(df); rownames(df) #vector of names
dim(df); dimnames(df) # alts. to above
# Also: head(df); tail(df); summary(df)
```

Referencing cells [row, col] [[r, c]]

```
# [[ for single cell selection; [ for multi
vec <- df[[5, 2]] # get cell by row/col num
newDF <- df[1:5, 1:2] # get multi in new df
df[[2, 'col1']] <- 12 # set single cell
df[3:5, c('col1', 'col2')] <- 9 # set multi
```

Referencing rows [i,]

```
# returns a data frame (and not a vector!)
row.1 <- df[1, ]; row.n <- df[nrow(df),]
# to get a row as a vector, use following
vrow <- as.numeric(as.vector(df[row,]))
vrow <- as.character(as.vector(df[row,]))
```

Referencing columns [,j] [j] [[j]] $name

```
# most column references return a vector
col.vec <- df$cats # returns a vector
col.vec <- df[, 'horses'] # returns vector
col.vec <- df[, a] # a is int or string
col.vec <- df[['frogs']] # returns a vector
frogs.df <- df['frogs'] # returns 1 col df
first.df <- df[1] # returns 1 col df
first.col <- df[, 1] # returns a vector
last.col <- df[, ncol(df)] # returns vector
```

Adding rows

```
# The right way ... (both args are DFs)
df <- rbind(df, data.frame(col1='d',
    col2=3, col3='A'))
```

Adding columns

```
df$newCol <- rep(NA, nrow(df)) # NA column
df[, newCol] <- df$oldCol # copy a col
df$y.percent.of.x <- df$y / sum(df$x) * 100
df <- cbind(col, df); df <- cbind(df, col)
transform(df, col3 = col1 * col2)
df[, col3] <- with(df, col1 + col2)
df <- within(df, newCol <- colA + colB)
```

Column names # analogous for rownames()

```
colnames(df) <- c('col.1', 'col.2, 'c3')
colnames(df)[1] <- 'new.name.for.col.1'
colnames(df)[colnames(df) %in% c('a', 'b')]
    <- c('x', 'y') # order of sub from cols
```

Selecting multiple rows

```
firstTenRows <- df[1:10, ] # head(df, 10)
everythingButRowTwo <- df[-2, ]
sub <- df[ (df$x > 5 & df$y < 5), ]
sub <- subset(df, x > 5 & y < 5)
# Note: vector Boolean (&, |) in above
allButLastRow <- head(df, -1) # tail(df,-1)
```

Selecting multiple columns

```
df <- df[, c(1, 2, 3)] # keep cols 1, 2 & 3
df <- df[, c('col1', 'col3')] # by name
# drop columns ...
df <- df[, -1] # keep all but first column
df <- df[, -c(1, 3)] # drop cols 1 and 3
df <- df[, !(colnames(df) %in%
    c('notThis', 'norThis'))]# drop by name
```

Replace column elements by row selection

```
df[df$col3 == 'A', 'col2'] <-
    c('j', 'a', 'a', 'a', 'j')
```

Manipulation

```
sorted <- df[order(df$col2), ]
backwards <- df[rev(order(df$col2)), ]
transposed <- as.data.frame(t(df))
merged <- merge(df1,df2,by='col',all=TRUE)
molten <- melt( df, id=c('year', 'month'),
    measure=c('col5', 'col10', col'15) )
# Note: melt comes from the reshape package
rownames(df)<- seq_len(nrow(df))#renum rows
summary <- ddply( df, ~col1, summarise,
    N=length(col3). mean=mean(col3) )
summary <- ddply(df, .(col1, col2),
    summarise, sd=sd(col3) mean=mean(col3))
# Note: ddply comes from the plyr package
```

Missing data (NA)

```
any(is.na(df)) # detect anywhere in df
any(is.na(df$col)) # anywhere in col
# delete selected missing data rows
df <- df[!is.na(df$col), ]
# replace NAs with something else
df[is.na(df)] <- 0 # works on whole df
df$col[is.na(df$col)] <- newValue
df$col <- ifelse(is.na(df$col), 0, df$col)
df <- orig[!is.na(orig$series),
    c('Date', series)] # selecting on r & c
```

Traps

```
# 1) for loops on possibly empty df's, use:
     for(i in seq_len(nrow(df))
# 2) columns coerced to factors, avoid with
#    the argument stringsAsFactors=FALSE
# 3) confusing row numbers and rows with
#    numbered names (hint: avoid row names)
# 4) although rbind() accepts vectors and
#    lists; this can fail with factor cols
```

