Chapter 4: Subsetting

Emeka Mbazor

3/23/2020

Data Types Exercises

1. Fix each of the following common data frame subsetting errors:

```
mtcars[mtcars$cyl = 4, ]
mtcars[-1:4, ]
mtcars[mtcars$cyl <= 5]
mtcars[mtcars$cyl == 4 | 6, ]
```

```
mtcars[mtcars$cyl == 4, ]
```

```
##                mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Datsun 710     22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Merc 240D      24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230       22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Fiat 128       32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic    30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona  21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Fiat X1-9      27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

```
mtcars[2:4, ]
```

```
##                mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4 Wag  21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710     22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive 21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
```

```
mtcars[mtcars$cyl <= 5, ]
```

```
##                mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Datsun 710     22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Merc 240D      24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230       22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Fiat 128       32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
```

```
## Honda Civic    30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona  21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Fiat X1-9      27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

```
mtcars[(mtcars$cyl == 4 | mtcars$cyl == 6), ]
```

```
##                  mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4       21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag   21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710      22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive  21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Valiant         18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Merc 240D       24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230        22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280        19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C       17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Fiat 128        32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic     30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla  33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona   21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Fiat X1-9       27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2   26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa    30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ferrari Dino    19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Volvo 142E      21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

2. Why does `x <- 1:5; x[NA]` yield five missing values? (Hint: why is it different from `x[NA_real_]`?)

```
x <- 1:5
x[NA]
```

```
## [1] NA NA NA NA NA
```

```
x[NA_real_]
```

```
## [1] NA
```

`x <- 1:5; x[NA]` yields five missing values because since `NA` is a logical vector it gets recycled four times.

3. What does `upper.tri()` return? How does subsetting a matrix with it work?

```
x <- outer(1:5, 1:5, FUN = "*")
x[upper.tri(x)]
```

```
x <- outer(1:5, 1:5, FUN = "*")
x
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    2    3    4    5
## [2,]    2    4    6    8   10
## [3,]    3    6    9   12   15
## [4,]    4    8   12   16   20
## [5,]    5   10   15   20   25
```

```
x[upper.tri(x)]
```

```
##  [1]  2  3  6  4  8 12  5 10 15 20
```

`upper.tri()` returns a logical matrix with the positions in the upper triangluar half of the matrix being marked `TRUE` and the lower triangular half of the matrix marked `FALSE`. Subsetting a matrix with it returns only the values in upper triagular half.

4. Why does `mtcars[1:20]` return an error? How does it differ from the similar `mtcars[1:20, ]`?

```
str(mtcars)
```

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

```
#mtcars[1:20]
mtcars[1:11]
```

```
##                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4          21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag      21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710         22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive     21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout  18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant            18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360         14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D          24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230           22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280           19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C          17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SE         16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL         17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC        15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood 10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
```

```
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

```
mtcars[1:20, ]
```

```
##                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4          21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag      21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710         22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive     21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout  18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant            18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360         14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D          24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230           22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280           19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C          17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SE         16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL         17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC        15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood 10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4  8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial  14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Fiat 128           32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic        30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla     33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
```

`mtcars[1:20]` returns an error because when you use a single dimension to subset a 2D object, you're subsetting that object's columns and `mtcars` only have 11 columns.

`mtcars[1:20, ]` is different because it's subsetting row-wise instead of column-wise.

5. Implement your own function that extracts the diagonal entries form a matrix. (it shoudld behave like `diag(x)` where x is matrix).

```
x <- matrix(1:16, ncol = 4, nrow = 4)
x
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    5    9   13
## [2,]    2    6   10   14
## [3,]    3    7   11   15
## [4,]    4    8   12   16
```

```
diag(x)
```

```
## [1]  1  6 11 16
```

```
y <- matrix(1:15, ncol = 3, nrow = 5)
y
```

```
##      [,1] [,2] [,3]
## [1,]    1    6   11
## [2,]    2    7   12
## [3,]    3    8   13
## [4,]    4    9   14
## [5,]    5   10   15
```

```
diag(y)
```

```
## [1]  1  7 13
```

```
diag2 <- function(x) {

  if(nrow(x) < ncol(x)) {
    limit <- nrow(x)
  } else {
    limit <- ncol(x)
  }

  y <- rep(0, limit)

  for(i in 1:limit) {
    y[i] <- x[i, i]
  }

  return(y)
}

diag2(x)
```

```
## [1]  1  6 11 16
```

```
diag2(y)
```

```
## [1]  1  7 13
```

6. What does `df[is.na(df)] <- 0` do? How does it work?

It replaces all `NA` values with 0. This is because `is.na()` returns a data frame filled with logical values based on the presence of a missing value or not.

Subsetting Operators Exercises

1. Given a linear model, e.g., `mod <- lm(mpg ~ wt, data = mtcars)`, extract the residual degrees of freedom. Extract the R squared from the model summary (summary(mod))

```
mod <- lm(mpg ~ wt, data = mtcars)

# a model is a list of 12 elements
str(mod)
```

```
## List of 12
##  $ coefficients : Named num [1:2] 37.29 -5.34
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "wt"
##  $ residuals    : Named num [1:32] -2.28 -0.92 -2.09 1.3 -0.2 ...
##   ..- attr(*, "names")= chr [1:32] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
##  $ effects      : Named num [1:32] -113.65 -29.116 -1.661 1.631 0.111 ...
##   ..- attr(*, "names")= chr [1:32] "(Intercept)" "wt" "" "" ...
##  $ rank         : int 2
##  $ fitted.values: Named num [1:32] 23.3 21.9 24.9 20.1 18.9 ...
##   ..- attr(*, "names")= chr [1:32] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
##  $ assign       : int [1:2] 0 1
##  $ qr           :List of 5
##   ..$ qr   : num [1:32, 1:2] -5.657 0.177 0.177 0.177 0.177 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:32] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
##   .. .. ..$ : chr [1:2] "(Intercept)" "wt"
##   .. ..- attr(*, "assign")= int [1:2] 0 1
##   ..$ qraux: num [1:2] 1.18 1.05
##   ..$ pivot: int [1:2] 1 2
##   ..$ tol  : num 1e-07
##   ..$ rank : int 2
##   ..- attr(*, "class")= chr "qr"
##  $ df.residual  : int 30
##  $ xlevels      : Named list()
##  $ call         : language lm(formula = mpg ~ wt, data = mtcars)
##  $ terms        :Classes 'terms', 'formula'  language mpg ~ wt
##   .. ..- attr(*, "variables")= language list(mpg, wt)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "mpg" "wt"
##   .. .. .. ..$ : chr "wt"
##   .. ..- attr(*, "term.labels")= chr "wt"
##   .. ..- attr(*, "order")= int 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. ..- attr(*, "predvars")= language list(mpg, wt)
##   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:2] "mpg" "wt"
##  $ model        :'data.frame':   32 obs. of  2 variables:
##   ..$ mpg: num [1:32] 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##   ..$ wt : num [1:32] 2.62 2.88 2.32 3.21 3.44 ...
##   ..- attr(*, "terms")=Classes 'terms', 'formula'  language mpg ~ wt
```

```
##   .. .. ..- attr(*, "variables")= language list(mpg, wt)
##   .. .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. ..$ : chr [1:2] "mpg" "wt"
##   .. .. .. .. ..$ : chr "wt"
##   .. .. ..- attr(*, "term.labels")= chr "wt"
##   .. .. ..- attr(*, "order")= int 1
##   .. .. ..- attr(*, "intercept")= int 1
##   .. .. ..- attr(*, "response")= int 1
##   .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. .. ..- attr(*, "predvars")= language list(mpg, wt)
##   .. .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. .. ..- attr(*, "names")= chr [1:2] "mpg" "wt"
##  - attr(*, "class")= chr "lm"
```

```
# residual degrees of freedom
mod$df.residual
```

```
## [1] 30
```

```
# a model's summary is a list of 11 elements
str(summary(mod))
```

```
## List of 11
##  $ call         : language lm(formula = mpg ~ wt, data = mtcars)
##  $ terms        :Classes 'terms', 'formula'  language mpg ~ wt
##   .. ..- attr(*, "variables")= language list(mpg, wt)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "mpg" "wt"
##   .. .. .. ..$ : chr "wt"
##   .. ..- attr(*, "term.labels")= chr "wt"
##   .. ..- attr(*, "order")= int 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. ..- attr(*, "predvars")= language list(mpg, wt)
##   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:2] "mpg" "wt"
##  $ residuals    : Named num [1:32] -2.28 -0.92 -2.09 1.3 -0.2 ...
##   ..- attr(*, "names")= chr [1:32] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
##  $ coefficients : num [1:2, 1:4] 37.285 -5.344 1.878 0.559 19.858 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "wt"
##   .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
##  $ aliased      : Named logi [1:2] FALSE FALSE
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "wt"
##  $ sigma        : num 3.05
##  $ df           : int [1:3] 2 30 2
##  $ r.squared    : num 0.753
##  $ adj.r.squared: num 0.745
##  $ fstatistic   : Named num [1:3] 91.4 1 30
##   ..- attr(*, "names")= chr [1:3] "value" "numdf" "dendf"
```

```
##  $ cov.unscaled : num [1:2, 1:2] 0.38 -0.1084 -0.1084 0.0337
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "wt"
##   .. ..$ : chr [1:2] "(Intercept)" "wt"
##  - attr(*, "class")= chr "summary.lm"
```

```
# R-Squared
summary(mod)$r.squared
```

```
## [1] 0.7528328
```

Application Exercises

1. How would you randomly permute the columns of a data frame? (This is an important technique in random forests.) Can you simultaneously permute the rows and columns in one step?

```
df <- data.frame(x = rep(1:3, each = 2), y = 6:1, z = letters[1:6])

df
```

```
##   x y z
## 1 1 6 a
## 2 1 5 b
## 3 2 4 c
## 4 2 3 d
## 5 3 2 e
## 6 3 1 f
```

```
#randomly permute the columns of a data frame
df[, sample(ncol(df))]
```

```
##   y z x
## 1 6 a 1
## 2 5 b 1
## 3 4 c 2
## 4 3 d 2
## 5 2 e 3
## 6 1 f 3
```

```
# randomly permute the rows and columns in one step
df[sample(nrow(df)), sample(ncol(df))]
```

```
##   z x y
## 2 b 1 5
## 5 e 3 2
## 3 c 2 4
## 4 d 2 3
## 1 a 1 6
## 6 f 3 1
```

2. How would you select a random sample of m rows from a data frame? s

```
m <- sample(nrow(df))[sample(nrow(df))[1]]

df[sample(nrow(df), m),]
```

```
##   x y z
## 1 1 6 a
```

3. How could you put the columns in a data frame in alphabetical order?

```
df[order(names(df))]
```

```
##   x y z
## 1 1 6 a
## 2 1 5 b
## 3 2 4 c
## 4 2 3 d
## 5 3 2 e
## 6 3 1 f
```