Chapter 3 Exercises

Derek Chiu

July 20, 2016

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

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

   `NA` is a logical that always replaces a vector with `NA`. Coercion doesn't occur with `NA_real_`.

3. What does `upper.tri()` return? How does subsetting a matrix with it work? Do we need any additional subsetting rules to describe its behaviour?

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

   A matrix of logicals where the upper triangular is `TRUE`. It returns elements that belong to the upper triangular. The elements are returned as a vector in column order.

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

   There aren't 20 columns in `mtcars`. `mtcars[1:20, ]` subsets the first 20 rows and all columns.

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

```
diagonal <- function(x) {
  return(unname(unlist(x)[which(row(x) - col(x) == 0)]))
}
x <- matrix(1:25, nrow = 5)
diagonal(x)
```

```
## [1]  1  7 13 19 25
```

```
diagonal(mtcars)
```

```
##  [1]  21.00   6.00 108.00 110.00   3.15   3.46  15.84   1.00   0.00   4.00
## [11]   4.00
```

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

   Replaces all entries of `df` which are `NA` with 0. Matrix subsetting is used since `is.na(df)` is a matrix of logicals.

3.2.4 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)
mod$resid
```

```
##           Mazda RX4       Mazda RX4 Wag          Datsun 710
##          -2.2826106          -0.9197704          -2.0859521
##      Hornet 4 Drive   Hornet Sportabout             Valiant
##           1.2973499          -0.2001440          -0.6932545
##          Duster 360           Merc 240D            Merc 230
##          -3.9053627           4.1637381           2.3499593
##            Merc 280           Merc 280C          Merc 450SE
##           0.2998560          -1.1001440           0.8668731
##          Merc 450SL         Merc 450SLC  Cadillac Fleetwood
##          -0.0502472          -1.8830236           1.1733496
## Lincoln Continental   Chrysler Imperial            Fiat 128
##           2.1032876           5.9810744           6.8727113
##         Honda Civic      Toyota Corolla       Toyota Corona
##           1.7461954           6.4219792          -2.6110037
##    Dodge Challenger         AMC Javelin          Camaro Z28
##          -2.9725862          -3.7268663          -3.4623553
##    Pontiac Firebird           Fiat X1-9       Porsche 914-2
##           2.4643670           0.3564263           0.1520430
##        Lotus Europa      Ford Pantera L        Ferrari Dino
##           1.2010593          -4.5431513          -2.7809399
##       Maserati Bora          Volvo 142E
##          -3.2053627          -1.0274952
```

```
summary(mod)$r.squared
```

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

3.4.9 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?

```
set.seed(1)
x <- matrix(1:25, nrow = 5)
x[, sample(ncol(x))]
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    6   21   16   11    1
## [2,]    7   22   17   12    2
## [3,]    8   23   18   13    3
## [4,]    9   24   19   14    4
## [5,]   10   25   20   15    5
```

```
x[sample(nrow(x)), sample(ncol(x))]
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   10    5   15   20   25
## [2,]    9    4   14   19   24
## [3,]    7    2   12   17   22
## [4,]    8    3   13   18   23
## [5,]    6    1   11   16   21
```

2. How would you select a random sample of m rows from a data frame? What if the sample had to be contiguous (i.e., with an initial row, a final row, and every row in between)?

```
x <- mtcars
m <- 5
x[sample(nrow(x), m), ]
```

```
##                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
```

```
x[do.call(seq, as.list(sort(sample(nrow(x), 2, replace = TRUE)))), ]
```

```
##                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## 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
```

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

```r
x <- data.frame(b = 1:5, c = 6:10, a = 11:15)
x[, sort(names(x))]
```

```
##    a b  c
## 1 11 1  6
## 2 12 2  7
## 3 13 3  8
## 4 14 4  9
## 5 15 5 10
```