Stat 2994: Statistical Computing

Introduction to R: 4

One very useful way of extracting information from a large matrix is to use the row or column name of the matrix in logical expressions as subscripts. We will use the built-in R dataset *state.x77* which is a matrix of 50 rows and 8 columns.

```
> ?state
```

```
> data(state)
> colnames(state.x77)
[1] "Population" "Income"     "Illiteracy" "Life Exp"   "Murder"
[6] "HS Grad"    "Frost"      "Area"
```

The technique here is to use the column and row names instead of the location number and apply a logical statement to get only those observations we are interested in.

```
> state.x77[,"Area"]
```

```
       This will show us the Area of each state
```

```
> state.x77[,"Area"]>80000
```

```
       This will give a vector of TRUE/FALSE values for each state
```

Now we can use the logical vector from above to subscript the original dataset and get all of the information for states with an Area greater than 80,000 square miles. Notice that in the Row argument, we have the logical statement state.x77[,"Area"] > 80000, this tells R to only include the rows where the Area is greater than 80000, and the column argument is left blank indicating we want to include all of the columns available. This will allow us to work with all of the variables in a subset of the original data.

```
> state.x77[state.x77[,"Area"]>80000,]
           Population Income Illiteracy Life Exp Murder HS Grad Frost   Area
Alaska            365   6315        1.5    69.31   11.3    66.7   152 566432
Arizona          2212   4530        1.8    70.55    7.8    58.1    15 113417
California      21198   5114        1.1    71.71   10.3    62.6    20 156361
Colorado         2541   4884        0.7    72.06    6.8    63.9   166 103766
Idaho             813   4119        0.6    71.87    5.3    59.5   126  82677
Kansas           2280   4669        0.6    72.58    4.5    59.9   114  81787
Montana           746   4347        0.6    70.56    5.0    59.2   155 145587
Nevada            590   5149        0.5    69.03   11.5    65.2   188 109889
New Mexico       1144   3601        2.2    70.32    9.7    55.2   120 121412
Oregon           2284   4660        0.6    72.13    4.2    60.0    44  96184
Texas           12237   4188        2.2    70.90   12.2    47.4    35 262134
Utah             1203   4022        0.6    72.90    4.5    67.3   137  82096
Wyoming           376   4566        0.6    70.29    6.9    62.9   173  97203
```

```
> state.x77[state.x77[,"Area"]>80000,"HS Grad"]
    Alaska    Arizona California   Colorado      Idaho     Kansas    Montana
Nevada
      66.7       58.1       62.6       63.9       59.5       59.9       59.2
65.2
New Mexico     Oregon      Texas       Utah    Wyoming
      55.2       60.0       47.4       67.3       62.9
```

Exercise 1: Use the data set from above to answer the following questions:

Which states have a population over 5000 (thousand)?

What is the average Life Expectancy for all fifty states?

What is the Life Expectancy of states with a Murder rate over 10?

Find the median Income level for states with an Illiteracy rate above and below 1.

Data Frames

A quick word on data frames in R and their difference from matrices. A matrix in R can only have elements that are numerical values. A data frame in R can have numeric and character values as its elements. It is this reason that many pre-loaded data sets are of the class data frame and not matrix, and also why many functions will only work with data frames. Fortunately you can easily change a matrix into a data frame with the as.data.frame() function.

As an example, we will consider the attach() function, which allows us to use the variable name for each column as a shortcut to indexing those values. This function however does not work with matrices. The state.x77 data is currently a matrix in R.

```
> class(state.x77)
[1] "matrix"
> Income
Error: object 'Income' not found
> attach(state.x77)
Error in attach(state.x77) :
  'attach' only works for lists, data frames and environments

> state_data <- as.data.frame(state.x77)
> class(state_data)
[1] "data.frame"
> Income
 [1] 3624 6315 4530 3378 5114 4884 5348 4809 4815 4091 4963 4119 5107 4458
[15] 4628 4669 3712 3545 3694 5299 4755 4751 4675 3098 4254 4347 4508 5149
[29] 4281 5237 3601 4903 3875 5087 4561 3983 4660 4449 4558 3635 4167 3821
[43] 4188 4022 3907 4701 4864 3617 4468 4566
```

In a similar way the as.matrix() function will convert a data frame into a matrix, as long as there are no character values included.

Conditional Computations

The basic structure for a conditional statement in R is of the form

if (*condition*) *expression-1* else *expression-2*

where *condition* is an expression that evaluates to a logical value (TRUE/FALSE), *expression-1* is an R expression that will be executed if the value of *condition* is TRUE and *expression-2* is an R expression that will be executed if the value of *condition* is FALSE.

```
> a<-25;b<-50
> if(a > b) c<-a else c<-b
> c
[1] 50
```

```
> gpa<-3.5;sat<-560
> if (gpa<3 || sat<600){
+   category <- "B"
+   score <- gpa + 0.007*sat
+ } else {
+   category <- "A"
+   score <- gpa + 0.006*sat
+ }
> category
[1] "B"
> score
[1] 7.42
```

If the logical expression condition results in a vector of logical values, the ifelse() function is better suited for conditional evaluation. It is of the form

ifelse(*condition, expression-1, expression-2*)

and returns the a value of the same shape as *expression-1*, containing elements from *expression-1* or *expression-2* depending on whether the corresponding elements of *condition* are TRUE or FALSE, respectively.

```
> x <- 6:-4
> x
 [1]  6  5  4  3  2  1  0 -1 -2 -3 -4
> ifelse(x > 0, x, -x)
 [1] 6 5 4 3 2 1 0 1 2 3 4
> ifelse(x > 0, x, NA)
 [1]  6  5  4  3  2  1 NA NA NA NA NA
```

Exercise 2: First enter the objects below into R and then write conditional statements for each of the following scenarios.

Home = 27 TeamA = 20 Home_run = 98 TeamA_run = 123

Home_pass = 274 TeamA_pass = 223

1. Compare the home score to the other teams' scores separately, if home is larger assign a new object as "Home WIN!!" if it is not larger, the new object should be "We Lost :("

2. If the home team has more run yards and more pass yards, a new object *outcome* should say "Sure Win" and a second object *pred_score* will use the following equation to predict the score difference:

(Home_run – TeamA_run) / 4

If the Team A has more yards in either category, *outcome* should say "Not Sure" and *pred_score* should use the following equation to predict the score difference:

(Home_pass – TeamA_pass) / 7