Toddler Steps: Selecting Portions of Data Sets with R

Having mastered much of the basics of working with R, we'll now learn a few more commands that will help you deal with data, especially the issue of selecting portions or subsets of a data set. The basic idea is when you read in a csv file, or use a data statement, someone has provided you with that data. However, you may only wish to analyze or plot a portion of that data, not all of it. Hence, you need to know how to select portions of data. We'll also see a few other commands for good measure.

Selecting a Portion of a Vector

Selecting a portion of a vector is pretty straightforward, and there are several options. The examples shown here work for vectors of any data type/mode. Square brackets are used to hold the selection criteria:

```
vec <- c(20, 22, 1:5, 85, 43, 51, 14, 96:100)  # make up some data
vec  # see the data, note it is not sorted
```

```
##  [1]  20  22   1   2   3   4   5  85  43  51  14  96  97  98  99 100
```

```
length(vec)  # gives the number of entries
```

```
## [1] 16
```

```
sort(vec)  # sorts it; not assigned so only displayed
```

```
##  [1]   1   2   3   4   5  14  20  22  43  51  85  96  97  98  99 100
```

```
vec[15]  # display the 15th entry
```

```
## [1] 99
```

```
vec1 <- vec[1:5]  # select the 1st 5 values & assign to new object
vec1  # see the values
```

```
## [1] 20 22  1  2  3
```

```
vec2 <- vec[vec > 50]  # select only the values > 50
vec2
```

```
## [1]  85  51  96  97  98  99 100
```

```
vec > 50  # note the form of the answer here, T/F
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE
## [12]  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
which(vec > 50)  # but here it is the indices
```

```
## [1]  8 10 12 13 14 15 16
```

```
vec3 <- vec[!vec > 50]  # select those values not > 50; ! means negation
vec3
```

```
## [1] 20 22  1  2  3  4  5 43 14
```

```
vec4 <- vec[-c(10:20)]  # remove values 10 thru 20
vec4
```

```
## [1] 20 22  1  2  3  4  5 85 43
```

```
str(vec4)  # note mode numeric (num)
```

```
##  num [1:9] 20 22 1 2 3 4 5 85 43
```

```
name.vec <- c("John", "Sally", "Joe", "Susie",
    "Frank")
str(name.vec)  # note character (chr) mode, 5 entries
```

```
##  chr [1:5] "John" "Sally" "Joe" "Susie" "Frank"
```

```
name.vec[3]  # entries are in quotes, which is a reminder they are character strings
```

```
## [1] "Joe"
```

Selecting a Portion of a Data Frame

Remember a data frame is a collection of data whose types/mode may differ. Rows represent the same sample, measured variables are in columns and must be all of the same type/mode in a particular column. Let's load the `Puromycin` data frame we saw before:

```
data(Puromycin)  # loads the data set
str(Puromycin)  # gives the structure of Puroymcin
```

```
## 'data.frame': 23 obs. of  3 variables:
##  $ conc : num  0.02 0.02 0.06 0.06 0.11 0.11 0.22 0.22 0.56 0.56 ...
##  $ rate : num  76 47 97 107 123 139 159 152 191 201 ...
##  $ state: Factor w/ 2 levels "treated","untreated": 1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "reference")= chr "A1.3, p. 269"
```

You can see (all) the values in a data frame several ways:

```
Puromycin  # gives the whole thing, but if it's a large data frame, seeing all of it is
pointless
```

```
##    conc rate     state
## 1  0.02   76   treated
## 2  0.02   47   treated
## 3  0.06   97   treated
## 4  0.06  107   treated
## 5  0.11  123   treated
## 6  0.11  139   treated
## 7  0.22  159   treated
## 8  0.22  152   treated
## 9  0.56  191   treated
## 10 0.56  201   treated
## 11 1.10  207   treated
## 12 1.10  200   treated
## 13 0.02   67 untreated
## 14 0.02   51 untreated
## 15 0.06   84 untreated
## 16 0.06   86 untreated
## 17 0.11   98 untreated
## 18 0.11  115 untreated
## 19 0.22  131 untreated
## 20 0.22  124 untreated
## 21 0.56  144 untreated
## 22 0.56  158 untreated
## 23 1.10  160 untreated
```

```
head(Puromycin)  # just the beginning, or tail(Puromycin) for just the end
```

```
##   conc rate   state
## 1 0.02   76 treated
## 2 0.02   47 treated
## 3 0.06   97 treated
## 4 0.06  107 treated
## 5 0.11  123 treated
## 6 0.11  139 treated
```

You can select just certain columns:

```
Puromycin$conc  # displays all the concentration values as a vector
```

```
##  [1] 0.02 0.02 0.06 0.06 0.11 0.11 0.22 0.22 0.56 0.56 1.10 1.10 0.02 0.02
## [15] 0.06 0.06 0.11 0.11 0.22 0.22 0.56 0.56 1.10
```

```
# Puromycin[, 1] # selects the 1st column, which is the
#   conc values
# Puromycin[, 'conc'] # yet another way to get only
#   conc
```

But be careful – above, we selected just one column, in which all the values had (by definition) the same mode/type, so the result was a vector. If we select just one row, the result is a data frame, not a vector, because the one row contains several columns of different data modes:

```
Puromycin[1:3, ]  # 1st 3 rows, all columns, still a data frame
```

```
##   conc rate   state
## 1 0.02   76 treated
## 2 0.02   47 treated
## 3 0.06   97 treated
```

```
Puromycin[1:3, 1]  # concentration values of 1st 3 rows (a vector, because one column was
selected)
```

```
## [1] 0.02 0.02 0.06
```

```
Puromycin[1, ]  # still a data frame because each column has different mode
```

```
##   conc rate   state
## 1 0.02   76 treated
```

Any of the above expressions could be assigned to a new value, rather than being displayed for inspection. Use `conc <- Puromycin$conc` for example.

Some More Complicated Selections

You probably noticed that `Puromycin` contains readings from "treated" and "untreated" samples. If you just wanted the treated values, you would use:

```
treated <- subset(Puromycin, state == "treated")
# this means get only the values where 'state exactly
#   equals treated'
str(treated)  # note that 'state' still has two levels, but one has no members
```

```
## 'data.frame': 12 obs. of  3 variables:
##  $ conc : num  0.02 0.02 0.06 0.06 0.11 0.11 0.22 0.22 0.56 0.56 ...
##  $ rate : num  76 47 97 107 123 139 159 152 191 201 ...
##  $ state: Factor w/ 2 levels "treated","untreated": 1 1 1 1 1 1 1 1 1 1 ...
```

```
summary(treated)  # observe that R knows how to summarize data of different modes
```

```
##       conc            rate           state
##  Min.   :0.020   Min.   : 47   treated  :12
##  1st Qu.:0.060   1st Qu.:104   untreated: 0
##  Median :0.165   Median :146
##  Mean   :0.345   Mean   :142
##  3rd Qu.:0.560   3rd Qu.:193
##  Max.   :1.100   Max.   :207
```

```
# Try making these plots yourself; I've left them
#   unevaluated here to save space.
plot(treated$conc, treated$rate)  # note that each rate was measured twice
plot(rate ~ conc, treated)  # also works, this is the formula method of base graphics
```