R_Subsetting

Ni

Elements and sequence of elements

```
x=c("a","b","c","d","a")
# Numerical index
# First Element
x[1]
```

```
## [1] "a"
```

```
# Sequence of elements
x[1:4]
```

```
## [1] "a" "b" "c" "d"
```

```
# Logical index
x[x>"a"]
```

```
## [1] "b" "c" "d"
```

```
# Create a logical vector
u=x>"a"
u
```

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

```
# subset the vector x that get all the elements that are greater than a
x[u]
```

```
## [1] "b" "c" "d"
```

Subsecting lists

-single [] returns elements that's the same class as the original

-double [[]] return a sequence of...

-dollar sign $ can only be used with literal names

-pass a vector, extract multiple elements of a list

```r
x=list(itnum=1:4,food=c("Chocolate","Honey","Lemon"))
x
```

```
## $itnum
## [1] 1 2 3 4
##
## $food
## [1] "Chocolate" "Honey"     "Lemon"
```

```r
# The first element
# single [] returns elements that's the same class as the original
x[1] # get back a list
```

```
## $itnum
## [1] 1 2 3 4
```

```r
# double [[]] return a sequence of 1 through 4
x[[1]]
```

```
## [1] 1 2 3 4
```

```r
# returns elements that in food
# $ can only be used with literal names
x$food
```

```
## [1] "Chocolate" "Honey"     "Lemon"
```

```r
x[["food"]]
```

```
## [1] "Chocolate" "Honey"     "Lemon"
```

```r
# return a lis
x["food"]
```

```
## $food
## [1] "Chocolate" "Honey"     "Lemon"
```

```r
# extract multiple elements of a list
x=list(itnum=1:4,food=c("Chocolate","Honey","Lemon"),say="helo")
# extract the 1st and 3rd element
x[c(1,3)] # pass a vector
```

```
## $itnum
## [1] 1 2 3 4
##
## $say
## [1] "helo"
```

```r
x=list(itnum=1:4,food=c("Chocolate","Honey","Lemon"))
name="food"
x[[name]] # computed index for "itnum"
```

```
## [1] "Chocolate" "Honey"     "Lemon"
```

```r
x$name # element "name" does not exist!
```

```
## NULL
```

```r
x$food # element "itnum" does exist
```

```
## [1] "Chocolate" "Honey"     "Lemon"
```

Subsetting Nested Elements of a list

-The [[]] can take an integer sequence

```r
x=list(a=list(10,12,14),b=c(3.14,2.81))
# The third element of the first sublist
x[[c(1,3)]]
```

```
## [1] 14
```

```r
# The first element of the second sublist
x[[c(2,1)]]
```

```
## [1] 3.14
```

#Subsetting Matrices

```r
# (i, j)
x=matrix(1:6,2,3)
x
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```r
x[1,2]
```

```
## [1] 3
```

```r
# indices can also be missing
x[1,]
```

```
## [1] 1 3 5
```

```
x[,2]
```

```
## [1] 3 4
```

by default, when a single ele of a matrix is retrived, it is returned as a vector of length 1 rather than a 1*1 matrix. This behavior can be turned off by setting drop = FALSE

```
x=matrix(1:6,2,3)
x[1,2]
```

```
## [1] 3
```

```
# get back a matrix
x[1,2,drop=FALSE]
```

```
##      [,1]
## [1,]    3
```

```
# similarly, subsetting a single column or a single row will give you a vector, not a matrix, but we ca
# get first row
x[1,,drop=FALSE]
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
```

```
# get second row
x[2,,drop=FALSE]
```

```
##      [,1] [,2] [,3]
## [1,]    2    4    6
```

```
# get first col
x[,1,drop=FALSE]
```

```
##      [,1]
## [1,]    1
## [2,]    2
```

Partial matching

-partial matching of names is allowed with [[and $ can save a lot of typing at the command line

```
x=list(tenacious = 1:6)
# look for a name in this list that mattches the letter t
x$t
```

```
## [1] 1 2 3 4 5 6
```

```r
# The problem for [[]] is that the name must be exact match for one of the # names in the list
x[["t"]]
```

```
## NULL
```

```r
#
x[["t",exact=FALSE]]
```

```
## [1] 1 2 3 4 5 6
```

Removing NA values

```r
x=c(1,2,NA,4,NA,5)
# create a logical vector that tells you where the NA's are and so that you can remove them by sub-sett
bad=is.na(x)
# bad is a logical vector
bad
```

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

```r
# not TRUE
x[!bad]
```

```
## [1] 1 2 4 5
```

```r
# multiple things and wanna take the subset with no missing values
y=c("a","b","NA","d","NA","F")
# use the complete cases function on both vectors which will give me a vector that tells me which of th
good=complete.cases(x,y)
good
```

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

```r
x[good]
```

```
## [1] 1 2 4 5
```

```r
y[good]
```

```
## [1] "a" "b" "d" "F"
```

Vectorized operations

```r
x=1:4;y=2:5
x+y
```

```
## [1] 3 5 7 9
```

```r
x*y
```

```
## [1]  2  6 12 20
```

```r
x/y
```

```
## [1] 0.5000000 0.6666667 0.7500000 0.8000000
```

```r
x>2
```

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

Vectorized matrix operations

```r
x=matrix(1:4,2,2); y=matrix(rep(10,4),2,2)
x
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
```

```r
y
```

```
##      [,1] [,2]
## [1,]   10   10
## [2,]   10   10
```

```r
# element-wise multiplication
x*y
```

```
##      [,1] [,2]
## [1,]   10   30
## [2,]   20   40
```

```r
x/y
```

```
##      [,1] [,2]
## [1,]  0.1  0.3
## [2,]  0.2  0.4
```

```r
# true matrix multiplication
x%*%y
```

```
##      [,1] [,2]
## [1,]   40   40
## [2,]   60   60
```

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

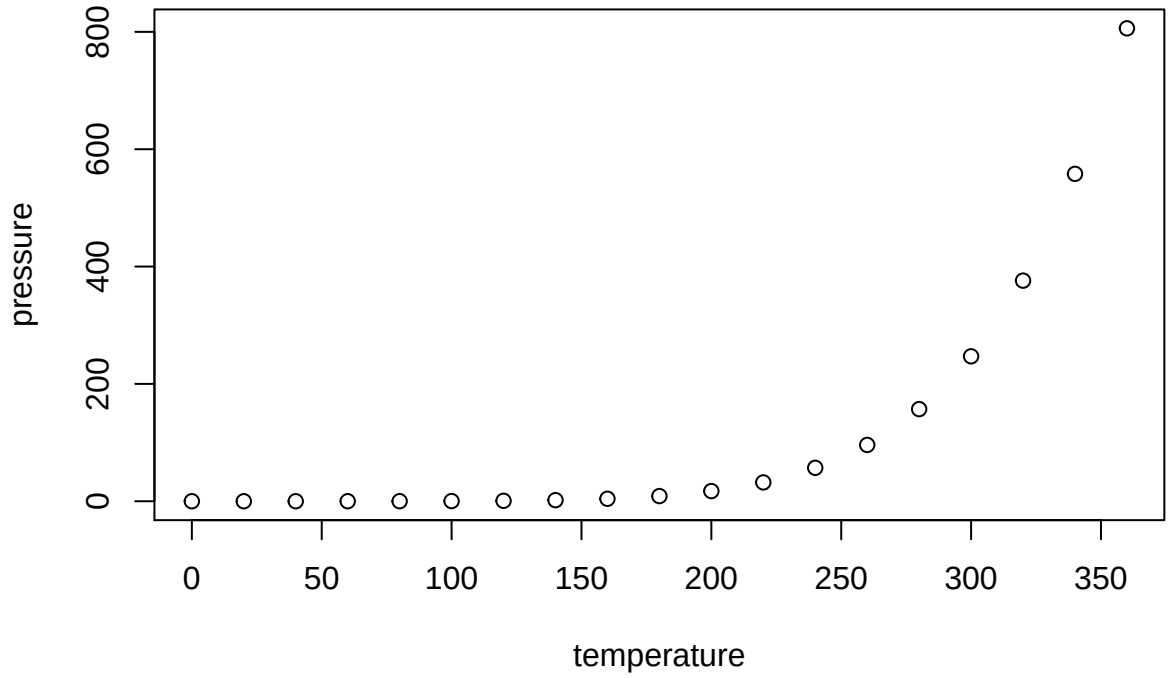
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```r
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.