Rprogramming notes

me

6/28/2021

Objects

create any vector by `vector()` list can include explicity wants a integer suffix L 1(num) 1L(int) `attributes()` can modify names, dimensions, class, length etc. `c()` is a func to be used to create vectors of objects finally , by using `vector()` "datatype", length=.. you can set the vector as you wanted then uses `class(x)` to know the datatype currently

```
x<- c(0.5, 0.6)     ## numeric
x<- c(TRUE, FALSE)  ## logical
x<- c(T, F)         ## logical
x<- c("a","b","c")  ## charac
x<- 9:29            ## integer
x<- c(1+0i, 2+3i)   ## complex

x <- vector("numeric", length =10)
x
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0
```

```
class(x)
```

```
## [1] "numeric"
```

mixed objects

```
y <- c(2,"y") ## character
y
```

```
## [1] "2" "y"
```

```
y <- c(TRUE, 3) ## numeric
y
```

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

```
y <- c("a", TRUE) ## character
y
```

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

using as.datatype(x)

```
x <- 0:8
x
```

```
## [1] 0 1 2 3 4 5 6 7 8
```

```
class(x)
```

```
## [1] "integer"
```

```
as.logical(x)
```

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

```
as.numeric(x)
```

```
## [1] 0 1 2 3 4 5 6 7 8
```

```
as.character(x)
```

```
## [1] "0" "1" "2" "3" "4" "5" "6" "7" "8"
```

```
as.complex(x)
```

```
## [1] 0+0i 1+0i 2+0i 3+0i 4+0i 5+0i 6+0i 7+0i 8+0i
```

```
b <- c("1","2","3")
class(b)
```

```
## [1] "character"
```

```
as.logical(b)
```

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

```
as.numeric(b)
```

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

lists

every elements in the lists can be different classes and length (dataframe, same length and within cols and rows)

```
x <- list( TRUE, 2+3i, "g", 3)
x
```

```
## [[1]]
## [1] TRUE
##
## [[2]]
## [1] 2+3i
##
## [[3]]
## [1] "g"
##
## [[4]]
## [1] 3
```

matrices <- vectors that have dimensions

matrix same class with cols and rows create a matrix by 3 ways 1. `matrix(3:10, ncol=3, nrow = 4)` 2. made a sequence first, then attribute dimension

```
m <- 1:15
dim(m)<-c(5, 3)
m
```

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

3. combine vectors into matrix

```
x <- 1:4
y<- 10:13
cbind(x,y)
```

```
##      x  y
## [1,] 1 10
## [2,] 2 11
## [3,] 3 12
## [4,] 4 13
```

```
rbind(x,y)
```

```
##   [,1] [,2] [,3] [,4]
## x    1    2    3    4
## y   10   11   12   13
```

you can use `dim()` for knowing the col and row for the matrix and you've learned `matrix(3:10, ncol=3, nrow = 4)``cbind(a,b)``rbind(x,y)`.

factors

factors are used for represent categorical data

```
x<- factor(c("yes", "yes", "no", "yes", "no"))
x
```

```
## [1] yes yes no  yes no
## Levels: no yes
```

```
table(x)
```

```
## x
##  no yes
##   2   3
```

```
unclass(x)
```

```
## [1] 2 2 1 2 1
## attr(,"levels")
## [1] "no"  "yes"
```

`table()`can show you how many levels are there and `unclass()` can tell you how it's calculate as a integer in R, it attribute 1 as no and 2 as yes, this follows the order of the alphabet, n go forward than y, so if you want to redefine the baseline of the factor, you may use `levels = c("yes", "no")` in the factor func.

```
x <- factor(c("yes", "no", "yes", "yes", "no"), levels =c("yes","no"))
x
```

```
## [1] yes no  yes yes no
## Levels: yes no
```

subset of is not set up for use with latex!!

missing values

nan c na nan only finds nan na finds both nan and na

```r
x <- c(1,2,NaN, NA,4)
is.nan(x)
```

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

```r
is.na(x)
```

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

Data frame

dataframe doesn't need same class or same length. and everyrow have a name

```r
x <- data.frame(foo = 1:4, bar =c(T,T,F,F))
x
```

```
##   foo   bar
## 1   1  TRUE
## 2   2  TRUE
## 3   3 FALSE
## 4   4 FALSE
```

```r
nrow(x)
```

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

```r
ncol(x)
```

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

————————error————————

```r
a<- c("   ", "list","matrix","dataframe","same class","X","O","X", "cols and rows","X","O","O","same le
dim(a)<- c(4,4)
a
```

```
##      [,1]        [,2]         [,3]            [,4]
## [1,] "   "       "same class" "cols and rows" "same length"
## [2,] "list"      "X"          "X"             "X"
## [3,] "matrix"    "O"          "O"             "O"
## [4,] "dataframe" "X"          "O"             "X"
```

```r
b <- data.frame(sameclass = ("X","O","X"),cols&rows <- c("X","O","O"), same_length <-c("X","O","X"))
b
```

————————fixed———————— prac on my own by constructing vectors forwardly, then combine them using data.frame(col = vec)

```r
sameclass <- c("X","O","X")
sameclass
```

```
## [1] "X" "O" "X"
```

```r
colrow <- c("X","O","O")
colrow
```

```
## [1] "X" "O" "O"
```

```r
same_length <-c("X","O","X")
same_length
```

```
## [1] "X" "O" "X"
```

```r
data1<- c("list","matrix", "data frame")
data1
```

```
## [1] "list"       "matrix"     "data frame"
```

```r
c <- data.frame(datatype = data1, sameclass = sameclass, colrow = colrow, same_length = same_length)
c
```

```
##     datatype sameclass colrow same_length
## 1       list         X      X           X
## 2     matrix         O      O           O
## 3 data frame         X      O           X
```

name attribute

```r
x <- 1:3
names(x)
```

```
## NULL
```

```r
names(x) <- c("1st","2nd","3rd")
names(x)
```

```
## [1] "1st" "2nd" "3rd"
```

```r
x
```

```
## 1st 2nd 3rd
##   1   2   3
```

```r
x <- list(x=1,y=2,c=3)
x
```

```
## $x
## [1] 1
##
## $y
## [1] 2
##
## $c
## [1] 3
```

```r
m <- matrix(1:6, nrow = 3, ncol = 2)
m
```

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

```r
dimnames(m) <- list(c("a","b","c"),c("one","two"))
m
```

```
##   one two
## a   1   4
## b   2   5
## c   3   6
```

reading data

`read.table()` `read.table()` set the `comment.char = ""` if there are no commented line in my file. `colClasses= "numeric"`will be wriiten in the read function, and it's really important, it can run FASTER , it'll dont't need to figure out the colClasses on its own.

dput/dget and dump()

made up a dataframe,then dput it and d get it. unlike wrinting out a table or csv file, dput and dump preserve the metadata(like the pic with json, formateed), so that another user doesn't have to specify it all over again. adv : it's textual + metadata, it won't need reconstruct

```r
y <- data.frame(a=1, b="a")
y
```

```
##   a b
## 1 1 a
```

```
dput(y)
```

```
## structure(list(a = 1, b = "a"), class = "data.frame", row.names = c(NA,
## -1L))
```

```
dput(y,file = "dputtest.R")
new.calldput <- dget("dputtest.R")
new.calldput
```

```
##   a b
## 1 1 a
```

dput can only be used in 1 r object, dump() can be used on multiple r obj.

```
x <- "w"
y <- data.frame(a=1 , b = "a")
dump(c("x","y"),file = "dumptry.R")
rm(x,y)
source("dumptry.R")
y
```

```
##   a b
## 1 1 a
```

```
x
```

```
## [1] "w"
```

i still cannot figure out what does source means

reading files from external

there are `file()` `url()` `bzfile()` `gzfile()` don't forget close the connection after using it by `close()`.

```
con <- file("midterm.csv")
x <- readLines(con, 10)
x
```

```
##  [1] "id,sex,age,group,irb" "1 ,1 ,45 ,1 ,1 "      "2 ,2 ,72 ,2 ,0 "
##  [4] "3 ,1 ,55 ,1 ,0 "      "4 ,1 ,79 ,2 ,1 "      "5 ,1 ,18 ,1 ,1 "
##  [7] "6 ,1 ,33 ,2 ,0 "      "7 ,1 ,66 ,1 ,1 "      "8 ,2 ,68 ,2 ,1 "
## [10] "9 ,1 ,68 ,1 ,1 "
```

```
hey <- url("https://www.jhsph.edu","r")
x <- readLines(hey,10)
head(x)
```

```
## [1] "<!DOCTYPE html>"
## [2] "<html lang=\"en\">"
## [3] ""
## [4] "<head>"
## [5] "<meta charset=\"utf-8\" />"
## [6] "<title>Johns Hopkins Bloomberg School of Public Health</title>"
```

subsetting

[] returns [[]] $

```
x <- c("a","b","c","c","d","a")
x
```

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

```
x[1]
```

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

```
x[2]
```

```
## [1] "b"
```

```
x[2:5]
```

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

```
x[3:7]
```

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

```
x[x> "b"]
```

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

```
u <- x > "b"
u
```

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

```
x[u]
```

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

subsetting lists

```
x <- list(foo = 1:4, bar =0.6)
x ## it'll print 2 list with name and sequence
```

```
## $foo
## [1] 1 2 3 4
##
## $bar
## [1] 0.6
```

```r
x[1] ## the first list
```

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

```r
## same returns : the sequence of the first list
x[[1]]
```

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

```r
x[["foo"]]
```

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

```r
x$f
```

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

```r
x[["f", exact = FALSE]]
```

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

```r
x["bar"] ## the 2nd list
```

```
## $bar
## [1] 0.6
```

```r
x$bar ## $ is similar to [[]]
```

```
## [1] 0.6
```

```r
x[[2]]
```

```
## [1] 0.6
```

```r
x$b
```

```
## [1] 0.6
```

```r
x<- list(foo = 1:4, bar = 0.6, baz = "hello")
x ## can see 3 lists
```

```
## $foo
## [1] 1 2 3 4
##
## $bar
## [1] 0.6
##
## $baz
## [1] "hello"
```

```r
x[c(2,3)] ## choosing multiple  ele
```

```
## $bar
## [1] 0.6
##
## $baz
## [1] "hello"
```

```r
name <- "foo"
x[name] ## whole list
```

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

```r
x[[name]] ## sequence only
```

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

```r
x$name ## NULL  $ can only be used with literal name
```

```
## NULL
```

```r
x$foo ## sequence
```

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

```r
x <- list(a= list(11,13,15),b = c(3.14,2.81))
x ## a as a list will have index. bdoes not
```

```
## $a
## $a[[1]]
## [1] 11
##
## $a[[2]]
## [1] 13
##
## $a[[3]]
## [1] 15
##
##
## $b
## [1] 3.14 2.81
```

```r
x[[c(1,2)]]
```

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

```
x[[1]][[3]]
```

```
## [1] 15
```

```
x[[c(2,1)]]
```

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

```
x[[2]][[2]]
```

```
## [1] 2.81
```

subsetting matrix

```
x <- matrix(1:6,2,3) ## data row col
x
```

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

```
x[1,2] ## row=1 col=2 but only number returns
```

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

```
x[1,2,drop = FALSE] ## one by one matrix
```

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

```
x[2, ] ## row =2 col=
```

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

```
x[2, ,drop = FALSE]
```

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

partial matching

a tool that saves you from typing

```
x <- list(allthatbase = 1:5)
x$a
```

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

```r
x[["a"]] ## [[]] needs the exact name, no partial mathching
```

```
## NULL
```

```r
x[["a", exact = FALSE]] ## Turn off the default
```

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

reremoving NA values

using`is.na()` to return the logical response and`x[!bad]` to exclude the ele that reach `is.na()`

```r
x <-  c(1, 2, NA , 4, NA, 5)
x
```

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

```r
x[3]
```

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

```r
bad <- is.na(x) ## is.na() is a function of NA logical return
bad
```

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

```r
x[!bad]
```

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

`complete.cases()` is a func to record the index of complete cases in the list, watch out for using wrong index!!

```r
x <- c(1, 2, NA , 4, NA, 5, NA)
y <- c("a", "b", NA, "d", NA, "f", "g")

xgood <- complete.cases(x)
xgood
```

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

```r
ygood <- complete.cases(y)
ygood
```

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

```r
allgood <- complete.cases(x,y)
allgood
```

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

```r
x[ygood]
```

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

```r
x[xgood]
```

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

```r
y[ygood]
```

```
## [1] "a" "b" "d" "f" "g"
```

```r
cc <- data.frame(x = x, y = y)
cc
```

```
##    x    y
## 1  1    a
## 2  2    b
## 3 NA <NA>
## 4  4    d
## 5 NA <NA>
## 6  5    f
## 7 NA    g
```

```r
cc <- complete.cases(cc)
cc
```

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

```r
airquality[1,2]
```

```
## [1] 190
```

```r
airquality[1:10, ]
```

```
##    Ozone Solar.R Wind Temp Month Day
## 1     41     190  7.4   67     5   1
## 2     36     118  8.0   72     5   2
## 3     12     149 12.6   74     5   3
## 4     18     313 11.5   62     5   4
## 5     NA      NA 14.3   56     5   5
## 6     28      NA 14.9   66     5   6
## 7     23     299  8.6   65     5   7
## 8     19      99 13.8   59     5   8
## 9      8      19 20.1   61     5   9
## 10    NA     194  8.6   69     5  10
```

```
good <- complete.cases(airquality)
airquality[good, ][1:10, ]
```

```
##    Ozone Solar.R Wind Temp Month Day
## 1     41     190  7.4   67     5   1
## 2     36     118  8.0   72     5   2
## 3     12     149 12.6   74     5   3
## 4     18     313 11.5   62     5   4
## 7     23     299  8.6   65     5   7
## 8     19      99 13.8   59     5   8
## 9      8      19 20.1   61     5   9
## 12    16     256  9.7   69     5  12
## 13    11     290  9.2   66     5  13
## 14    14     274 10.9   68     5  14
```

```
data.frame(airquality = airquality[good, ])
```

```
##     airquality.Ozone airquality.Solar.R airquality.Wind airquality.Temp
## 1                 41                190             7.4              67
## 2                 36                118             8.0              72
## 3                 12                149            12.6              74
## 4                 18                313            11.5              62
## 7                 23                299             8.6              65
## 8                 19                 99            13.8              59
## 9                  8                 19            20.1              61
## 12                16                256             9.7              69
## 13                11                290             9.2              66
## 14                14                274            10.9              68
## 15                18                 65            13.2              58
## 16                14                334            11.5              64
## 17                34                307            12.0              66
## 18                 6                 78            18.4              57
## 19                30                322            11.5              68
## 20                11                 44             9.7              62
## 21                 1                  8             9.7              59
## 22                11                320            16.6              73
## 23                 4                 25             9.7              61
## 24                32                 92            12.0              61
## 28                23                 13            12.0              67
## 29                45                252            14.9              81
## 30               115                223             5.7              79
## 31                37                279             7.4              76
## 38                29                127             9.7              82
## 40                71                291            13.8              90
## 41                39                323            11.5              87
## 44                23                148             8.0              82
## 47                21                191            14.9              77
## 48                37                284            20.7              72
## 49                20                 37             9.2              65
## 50                12                120            11.5              73
## 51                13                137            10.3              76
## 62               135                269             4.1              84
```

```
## 63   49  248  9.2  85
## 64   32  236  9.2  81
## 66   64  175  4.6  83
## 67   40  314 10.9  83
## 68   77  276  5.1  88
## 69   97  267  6.3  92
## 70   97  272  5.7  92
## 71   85  175  7.4  89
## 73   10  264 14.3  73
## 74   27  175 14.9  81
## 76    7   48 14.3  80
## 77   48  260  6.9  81
## 78   35  274 10.3  82
## 79   61  285  6.3  84
## 80   79  187  5.1  87
## 81   63  220 11.5  85
## 82   16    7  6.9  74
## 85   80  294  8.6  86
## 86  108  223  8.0  85
## 87   20   81  8.6  82
## 88   52   82 12.0  86
## 89   82  213  7.4  88
## 90   50  275  7.4  86
## 91   64  253  7.4  83
## 92   59  254  9.2  81
## 93   39   83  6.9  81
## 94    9   24 13.8  81
## 95   16   77  7.4  82
## 99  122  255  4.0  89
## 100  89  229 10.3  90
## 101 110  207  8.0  90
## 104  44  192 11.5  86
## 105  28  273 11.5  82
## 106  65  157  9.7  80
## 108  22   71 10.3  77
## 109  59   51  6.3  79
## 110  23  115  7.4  76
## 111  31  244 10.9  78
## 112  44  190 10.3  78
## 113  21  259 15.5  77
## 114   9   36 14.3  72
## 116  45  212  9.7  79
## 117 168  238  3.4  81
## 118  73  215  8.0  86
## 120  76  203  9.7  97
## 121 118  225  2.3  94
## 122  84  237  6.3  96
## 123  85  188  6.3  94
## 124  96  167  6.9  91
## 125  78  197  5.1  92
## 126  73  183  2.8  93
## 127  91  189  4.6  93
## 128  47   95  7.4  87
## 129  32   92 15.5  84
```

```
## 130              20               252            10.9              80
## 131              23               220            10.3              78
## 132              21               230            10.9              75
## 133              24               259             9.7              73
## 134              44               236            14.9              81
## 135              21               259            15.5              76
## 136              28               238             6.3              77
## 137               9                24            10.9              71
## 138              13               112            11.5              71
## 139              46               237             6.9              78
## 140              18               224            13.8              67
## 141              13                27            10.3              76
## 142              24               238            10.3              68
## 143              16               201             8.0              82
## 144              13               238            12.6              64
## 145              23                14             9.2              71
## 146              36               139            10.3              81
## 147               7                49            10.3              69
## 148              14                20            16.6              63
## 149              30               193             6.9              70
## 151              14               191            14.3              75
## 152              18               131             8.0              76
## 153              20               223            11.5              68
##     airquality.Month airquality.Day
## 1                  5               1
## 2                  5               2
## 3                  5               3
## 4                  5               4
## 7                  5               7
## 8                  5               8
## 9                  5               9
## 12                 5              12
## 13                 5              13
## 14                 5              14
## 15                 5              15
## 16                 5              16
## 17                 5              17
## 18                 5              18
## 19                 5              19
## 20                 5              20
## 21                 5              21
## 22                 5              22
## 23                 5              23
## 24                 5              24
## 28                 5              28
## 29                 5              29
## 30                 5              30
## 31                 5              31
## 38                 6               7
## 40                 6               9
## 41                 6              10
## 44                 6              13
## 47                 6              16
## 48                 6              17
```

```
## 49                6               18
## 50                6               19
## 51                6               20
## 62                7                1
## 63                7                2
## 64                7                3
## 66                7                5
## 67                7                6
## 68                7                7
## 69                7                8
## 70                7                9
## 71                7               10
## 73                7               12
## 74                7               13
## 76                7               15
## 77                7               16
## 78                7               17
## 79                7               18
## 80                7               19
## 81                7               20
## 82                7               21
## 85                7               24
## 86                7               25
## 87                7               26
## 88                7               27
## 89                7               28
## 90                7               29
## 91                7               30
## 92                7               31
## 93                8                1
## 94                8                2
## 95                8                3
## 99                8                7
## 100               8                8
## 101               8                9
## 104               8               12
## 105               8               13
## 106               8               14
## 108               8               16
## 109               8               17
## 110               8               18
## 111               8               19
## 112               8               20
## 113               8               21
## 114               8               22
## 116               8               24
## 117               8               25
## 118               8               26
## 120               8               28
## 121               8               29
## 122               8               30
## 123               8               31
## 124               9                1
## 125               9                2
```

```
## 126                9              3
## 127                9              4
## 128                9              5
## 129                9              6
## 130                9              7
## 131                9              8
## 132                9              9
## 133                9             10
## 134                9             11
## 135                9             12
## 136                9             13
## 137                9             14
## 138                9             15
## 139                9             16
## 140                9             17
## 141                9             18
## 142                9             19
## 143                9             20
## 144                9             21
## 145                9             22
## 146                9             23
## 147                9             24
## 148                9             25
## 149                9             26
## 151                9             28
## 152                9             29
## 153                9             30
```

```
data.frame(airquality[good, ])
```

```
##     Ozone Solar.R Wind Temp Month Day
## 1      41     190  7.4   67     5   1
## 2      36     118  8.0   72     5   2
## 3      12     149 12.6   74     5   3
## 4      18     313 11.5   62     5   4
## 7      23     299  8.6   65     5   7
## 8      19      99 13.8   59     5   8
## 9       8      19 20.1   61     5   9
## 12     16     256  9.7   69     5  12
## 13     11     290  9.2   66     5  13
## 14     14     274 10.9   68     5  14
## 15     18      65 13.2   58     5  15
## 16     14     334 11.5   64     5  16
## 17     34     307 12.0   66     5  17
## 18      6      78 18.4   57     5  18
## 19     30     322 11.5   68     5  19
## 20     11      44  9.7   62     5  20
## 21      1       8  9.7   59     5  21
## 22     11     320 16.6   73     5  22
## 23      4      25  9.7   61     5  23
## 24     32      92 12.0   61     5  24
## 28     23      13 12.0   67     5  28
## 29     45     252 14.9   81     5  29
## 30    115     223  5.7   79     5  30
```

```
## 31     37     279  7.4   76     5  31
## 38     29     127  9.7   82     6   7
## 40     71     291 13.8   90     6   9
## 41     39     323 11.5   87     6  10
## 44     23     148  8.0   82     6  13
## 47     21     191 14.9   77     6  16
## 48     37     284 20.7   72     6  17
## 49     20      37  9.2   65     6  18
## 50     12     120 11.5   73     6  19
## 51     13     137 10.3   76     6  20
## 62    135     269  4.1   84     7   1
## 63     49     248  9.2   85     7   2
## 64     32     236  9.2   81     7   3
## 66     64     175  4.6   83     7   5
## 67     40     314 10.9   83     7   6
## 68     77     276  5.1   88     7   7
## 69     97     267  6.3   92     7   8
## 70     97     272  5.7   92     7   9
## 71     85     175  7.4   89     7  10
## 73     10     264 14.3   73     7  12
## 74     27     175 14.9   81     7  13
## 76      7      48 14.3   80     7  15
## 77     48     260  6.9   81     7  16
## 78     35     274 10.3   82     7  17
## 79     61     285  6.3   84     7  18
## 80     79     187  5.1   87     7  19
## 81     63     220 11.5   85     7  20
## 82     16       7  6.9   74     7  21
## 85     80     294  8.6   86     7  24
## 86    108     223  8.0   85     7  25
## 87     20      81  8.6   82     7  26
## 88     52      82 12.0   86     7  27
## 89     82     213  7.4   88     7  28
## 90     50     275  7.4   86     7  29
## 91     64     253  7.4   83     7  30
## 92     59     254  9.2   81     7  31
## 93     39      83  6.9   81     8   1
## 94      9      24 13.8   81     8   2
## 95     16      77  7.4   82     8   3
## 99    122     255  4.0   89     8   7
## 100    89     229 10.3   90     8   8
## 101   110     207  8.0   90     8   9
## 104    44     192 11.5   86     8  12
## 105    28     273 11.5   82     8  13
## 106    65     157  9.7   80     8  14
## 108    22      71 10.3   77     8  16
## 109    59      51  6.3   79     8  17
## 110    23     115  7.4   76     8  18
## 111    31     244 10.9   78     8  19
## 112    44     190 10.3   78     8  20
## 113    21     259 15.5   77     8  21
## 114     9      36 14.3   72     8  22
## 116    45     212  9.7   79     8  24
## 117   168     238  3.4   81     8  25
```

```
## 118    73     215  8.0   86     8  26
## 120    76     203  9.7   97     8  28
## 121   118     225  2.3   94     8  29
## 122    84     237  6.3   96     8  30
## 123    85     188  6.3   94     8  31
## 124    96     167  6.9   91     9   1
## 125    78     197  5.1   92     9   2
## 126    73     183  2.8   93     9   3
## 127    91     189  4.6   93     9   4
## 128    47      95  7.4   87     9   5
## 129    32      92 15.5   84     9   6
## 130    20     252 10.9   80     9   7
## 131    23     220 10.3   78     9   8
## 132    21     230 10.9   75     9   9
## 133    24     259  9.7   73     9  10
## 134    44     236 14.9   81     9  11
## 135    21     259 15.5   76     9  12
## 136    28     238  6.3   77     9  13
## 137     9      24 10.9   71     9  14
## 138    13     112 11.5   71     9  15
## 139    46     237  6.9   78     9  16
## 140    18     224 13.8   67     9  17
## 141    13      27 10.3   76     9  18
## 142    24     238 10.3   68     9  19
## 143    16     201  8.0   82     9  20
## 144    13     238 12.6   64     9  21
## 145    23      14  9.2   71     9  22
## 146    36     139 10.3   81     9  23
## 147     7      49 10.3   69     9  24
## 148    14      20 16.6   63     9  25
## 149    30     193  6.9   70     9  26
## 151    14     191 14.3   75     9  28
## 152    18     131  8.0   76     9  29
## 153    20     223 11.5   68     9  30
```

```
bb <- data.frame(airquality[good, ])
bb[1:10,] ## without this method, how to add the length into the dataframe??
```

```
##    Ozone Solar.R Wind Temp Month Day
## 1     41     190  7.4   67     5   1
## 2     36     118  8.0   72     5   2
## 3     12     149 12.6   74     5   3
## 4     18     313 11.5   62     5   4
## 7     23     299  8.6   65     5   7
## 8     19      99 13.8   59     5   8
## 9      8      19 20.1   61     5   9
## 12    16     256  9.7   69     5  12
## 13    11     290  9.2   66     5  13
## 14    14     274 10.9   68     5  14
```

quiz

16. how nany missing values are in the Ozone col of this data frame?

```
x <- read.csv("hw1_data.csv")
y = subset(x, is.na(Ozone))
y
```

```
##     Ozone Solar.R Wind Temp Month Day
## 5      NA      NA 14.3   56     5   5
## 10     NA     194  8.6   69     5  10
## 25     NA      66 16.6   57     5  25
## 26     NA     266 14.9   58     5  26
## 27     NA      NA  8.0   57     5  27
## 32     NA     286  8.6   78     6   1
## 33     NA     287  9.7   74     6   2
## 34     NA     242 16.1   67     6   3
## 35     NA     186  9.2   84     6   4
## 36     NA     220  8.6   85     6   5
## 37     NA     264 14.3   79     6   6
## 39     NA     273  6.9   87     6   8
## 42     NA     259 10.9   93     6  11
## 43     NA     250  9.2   92     6  12
## 45     NA     332 13.8   80     6  14
## 46     NA     322 11.5   79     6  15
## 52     NA     150  6.3   77     6  21
## 53     NA      59  1.7   76     6  22
## 54     NA      91  4.6   76     6  23
## 55     NA     250  6.3   76     6  24
## 56     NA     135  8.0   75     6  25
## 57     NA     127  8.0   78     6  26
## 58     NA      47 10.3   73     6  27
## 59     NA      98 11.5   80     6  28
## 60     NA      31 14.9   77     6  29
## 61     NA     138  8.0   83     6  30
## 65     NA     101 10.9   84     7   4
## 72     NA     139  8.6   82     7  11
## 75     NA     291 14.9   91     7  14
## 83     NA     258  9.7   81     7  22
## 84     NA     295 11.5   82     7  23
## 102    NA     222  8.6   92     8  10
## 103    NA     137 11.5   86     8  11
## 107    NA      64 11.5   79     8  15
## 115    NA     255 12.6   75     8  23
## 119    NA     153  5.7   88     8  27
## 150    NA     145 13.2   77     9  27
```

```
nrow(y)
```

```
## [1] 37
```

17.What is the mean of the Ozone column in this dataset? Exclude missing values (coded as NA) from this calculation.

```
z = subset(x, !is.na(Ozone), select = Ozone)
z
```

```
##     Ozone
## 1      41
## 2      36
## 3      12
## 4      18
## 6      28
## 7      23
## 8      19
## 9       8
## 11      7
## 12     16
## 13     11
## 14     14
## 15     18
## 16     14
## 17     34
## 18      6
## 19     30
## 20     11
## 21      1
## 22     11
## 23      4
## 24     32
## 28     23
## 29     45
## 30    115
## 31     37
## 38     29
## 40     71
## 41     39
## 44     23
## 47     21
## 48     37
## 49     20
## 50     12
## 51     13
## 62    135
## 63     49
## 64     32
## 66     64
## 67     40
## 68     77
## 69     97
## 70     97
## 71     85
## 73     10
## 74     27
## 76      7
## 77     48
## 78     35
```

```
## 79     61
## 80     79
## 81     63
## 82     16
## 85     80
## 86    108
## 87     20
## 88     52
## 89     82
## 90     50
## 91     64
## 92     59
## 93     39
## 94      9
## 95     16
## 96     78
## 97     35
## 98     66
## 99    122
## 100    89
## 101   110
## 104    44
## 105    28
## 106    65
## 108    22
## 109    59
## 110    23
## 111    31
## 112    44
## 113    21
## 114     9
## 116    45
## 117   168
## 118    73
## 120    76
## 121   118
## 122    84
## 123    85
## 124    96
## 125    78
## 126    73
## 127    91
## 128    47
## 129    32
## 130    20
## 131    23
## 132    21
## 133    24
## 134    44
## 135    21
## 136    28
## 137     9
## 138    13
## 139    46
```

```
## 140    18
## 141    13
## 142    24
## 143    16
## 144    13
## 145    23
## 146    36
## 147     7
## 148    14
## 149    30
## 151    14
## 152    18
## 153    20
```

```
colMeans(z)
```

```
##    Ozone
## 42.12931
```

```
apply(z,2,mean) ## what does fun = 2 means?
```

```
##    Ozone
## 42.12931
```

```
                ##  for a matrix 1 indicates rows, 2 indicates columns
```

18.Extract the subset of rows of the data frame where Ozone values are above 31 and Temp values are above 90. What is the mean of Solar.R in this subset?

```
b <- subset(x, Ozone > 30 & Temp > 90, select = Solar.R)
colMeans(b)
```

```
## Solar.R
##   212.8
```

19.What is the mean of "Temp" when "Month" is equal to 6?

```
c <- subset(x, Month == 6, select = Temp:Month)
colMeans(c)
```

```
##  Temp Month
##  79.1   6.0
```

What was the maximum ozone value in the month of May (i.e. Month = 5)?

```
d <- subset(x, Month == 5 & !is.na(Ozone), select = Ozone)
## must except the NAs , max will uses the NAs
apply(d, 2, max)
```

```
## Ozone
##   115
```