Data Types, Vectors, and Subsetting

Data analyst's perspective

- Think in terms of variables an ordered collection of measurements on a group of subjects
- Care about the kind of measuremet values: it informs the type of analysis we might perform, e.g., it makes sense to compute the mean/median of numeric values, but not categorical values
- Care about missing data we adjust our analyses depending on the amount and kind of missingness

Data types

- R has a number of built-in data types. The three most basic types are numeric, character, and logical
- You can check the type using the class function.

```
class(3.5)

## [1] "numeric"

class("Hello there")

## [1] "character"

class(TRUE)

## [1] "logical"
```

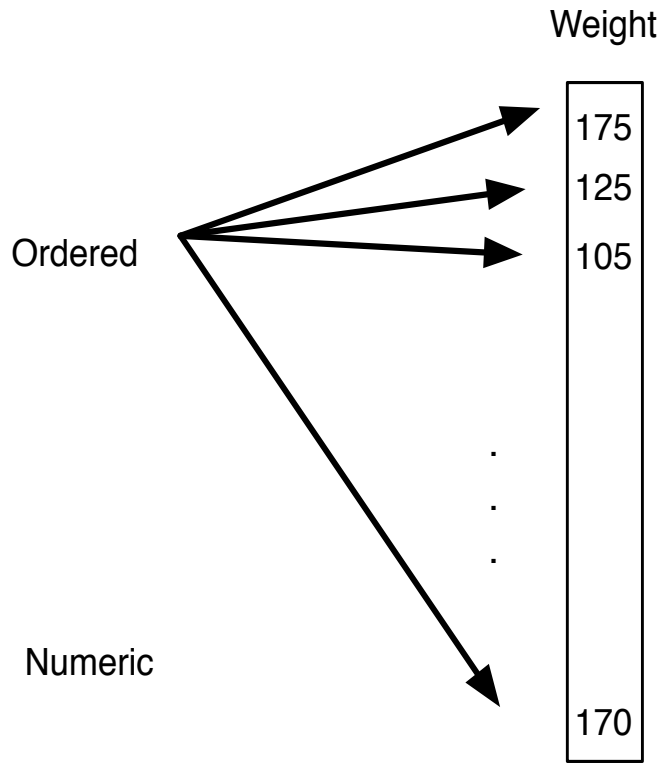
- Another important type is factor

Note about data types

- Actually, the types are numeric, character, and logical vectors. Theres no such thing as a scalar in R, just a vector of length one.

Vectors

- **Ordered** container
- Primitive elements of the **same type**



Vectors: family data example

- We have data on a 14-member family vectors of first names, age, gender, weight, height, whether or not they are over weight (BMI above 25).
- What are the data types?

```
load(url(
  "http://www.stat.berkeley.edu/users/nolan/data/afamil
```

- More readable:
load(url("http://www.stat.berkeley.edu/users/nolan/data/afamily.rda"))

First names and ages

```
fnames
```

```
##  [1] "Tom"    "Maya"   "Joe"    "Robert" "Sue"    "Liz"
##  [8] "Sally"  "Tim"    "Tom"    "Ann"    "Dan"    "Art"
```

```
class(fnames)
```

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

```
fage
```

```
##  [1] 77 33 79 47 27 33 67 52 59 27 55 24 46 48
```

```
class(fage)
```

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

Gender and over weight

```
fsex
```

```
##  [1] m f m m f f m f m m f m m f
## Levels: f m
```

```
class(fsex)
```

```
## [1] "factor"
```

```
foverWt
```

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

```
class(foverWt)
```

```
## [1] "logical"
```

More on data types

- A logical vector contains values that are either TRUE or FALSE.
- A factor vector is a special storage class used for qualitative data. The values are internally stored as integers by each integer corresponds to a level, which is a character string

```
levels(fsex)

## [1] "f" "m"
```

Special values

- The missing value symbol is NA
- It stands for Not Available
- NA can be an element of a vector of any type
- NA is different from the character string NA
- You can check for the presence of NA values using the is.na() function.

Special values

- Other special values are NaN, for not a number, which typically arises when you try to compute an indeterminate form such as 0/0.

```
0/0
```

```
## [1] NaN
```

- The result of dividing a non-zero number by zero is Inf (or -Inf).

```
12/0
```

```
## [1] Inf
```

Special values

- Other special values are NaN, for not a number, which typically arises when you try to compute an indeterminate form such as 0/0.

```
0/0
```

```
## [1] NaN
```

- The result of dividing a non-zero number by zero is Inf (or -Inf).

```
12/0
```

```
## [1] Inf
```

Special values

- NULL is a special value value that denotes an empty vector

```
names(fweight)

## NULL
```

- Here we asked for the names of the elements of the vector fweight. The function names returns a character vector of element names. Since this vector has no element names, the return value is a NULL vector

Finding out more information

- Retrieve the number of elements in the vector
- Examine the first 6 elements in the vector
- Elements can have names height has names
- Are any of the elements in the vector missing?

Finding out information: R code

```
length(fweight)

## [1] 14

head(fweight)

## [1] 175 124 185 156  98 190

names(fheight)

##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m"

is.na(fweight)

##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FAL
## [12] FALSE FALSE FALSE
```

Finding out information (contd)

- Aggregator functions operate on the elements of the vector
- Functions can tell us the about the data type
- Check if a vector is empty
- Convert a vector to a specified data type

Finding out information (contd): R code

```
min(fweight)

## [1] 98

is.logical(fweight)

## [1] FALSE

is.null(fheight)

## [1] FALSE

as.numeric(fsex)

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

Managing variables in the workspace

- Give names of all variables
- Remove one or more variables
- Save objects for future use
- Restore saved variables
- Save an entire workspace, and it will automatically load when you start R again

Managing variables: R code

```
ls()

## [1] "fage"    "family"  "fbmi"    "fheight" "fnames"  "f
## [8] "fweight"

rm(x)

## Warning in rm(x):  object 'x' not found

save(fage, fbmi, fweight, fheight,
     fsex, file="cdc200.rda")
load("cdc200.rda")
```

Subsetting: Extracting information

BMI of the 10th person in the family
Ages of all but the first person in the family

```
fbmi[10]

##         j
## 30.04911

fage[-1]

##  [1] 33 79 47 27 33 67 52 59 27 55 24 46 48
```

Suppose we want:

Height of person "j" (subset by name) Genders of the family members who are overweight (subset by logical value)

```
fheight["j"]

##  j
## 71

fsex[foverWt]

## [1] m f m m m f
## Levels: f m
```

Assign values to elements of a vector

- In general, the same indexing may be used to assign values to elements of a vector.
- Make sure the vector exists first, or you will get an error.

Assign values to elements of a vector

- Can you guess what fheight will look like after each of the following lines?
- fheight fheight[2]=61 fheight[-13]=62 fheight["e"]=67 fheight[overWt]=NA fheight[] = 70 fheight = 70

(Hint: inclusion, exclusion, name, logical, all, problem!)

More examples

a b c d e f g h i j k l m n
70 64 73 67 61 68 68 65 68 71 67 66 66 62

```
fheight[2]=61
fheight

##  a  b  c  d  e  f  g  h  i  j  k  l  m  n
## 70 61 73 67 61 68 68 65 68 71 67 66 66 62

fheight[-13]=62
fheight

##  a  b  c  d  e  f  g  h  i  j  k  l  m  n
## 62 62 62 62 62 62 62 62 62 62 62 62 66 62

fheight["e"]=67
fheight

##  a  b  c  d  e  f  g  h  i  j  k  l  m  n
```

More examples (logical)

T F F F F F T T F T T T F F F

```
fheight[foverWt]=NA
fheight

##  a  b  c  d  e  f  g  h  i  j  k  l  m  n
## NA 62 62 62 67 NA NA 62 NA NA NA 62 66 62

fheight[]=70
fheight

##  a  b  c  d  e  f  g  h  i  j  k  l  m  n
## 70 70 70 70 70 70 70 70 70 70 70 70 70 70

fheight=70
fheight

## [1] 70
```

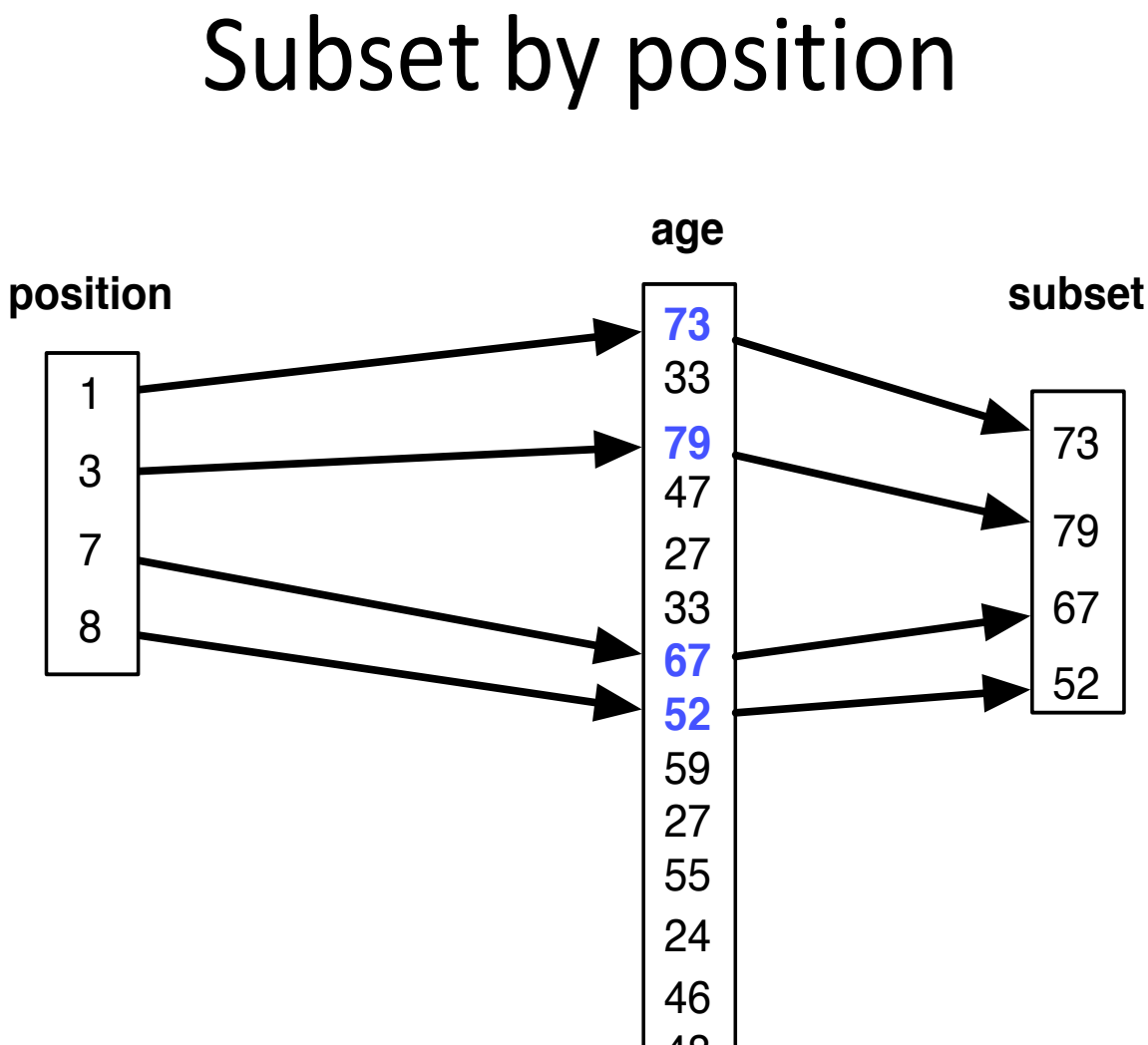
Suppose we are interested in

- Age of those who are not overweight
- Weights of the women in our family
- BMI of Tim and Tom
- Create a new variable for last name, all Smith

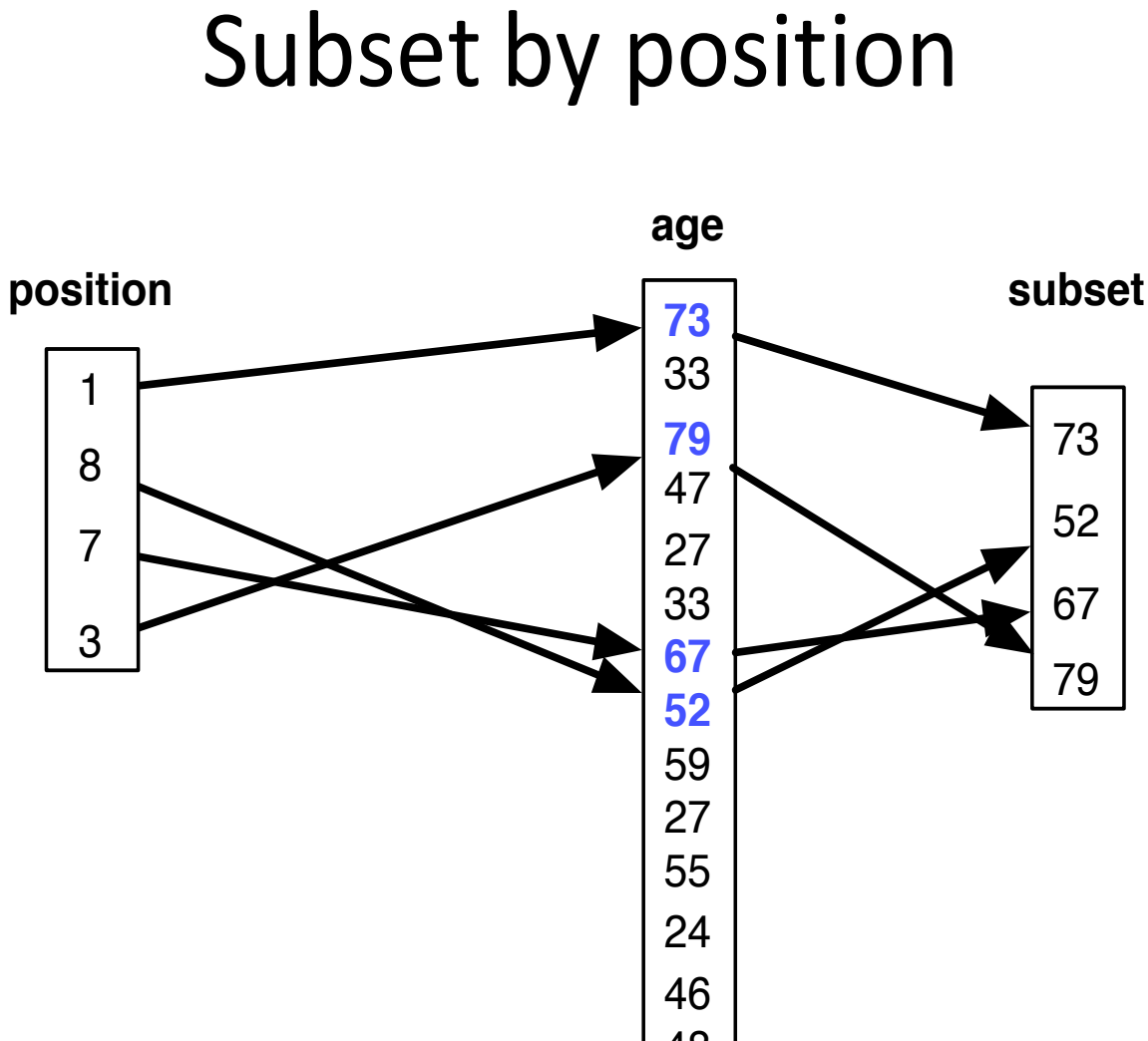
We need to better understand:

- How to use logical operators to create logical vectors
- How to create vectors with specific numbers and/or letters

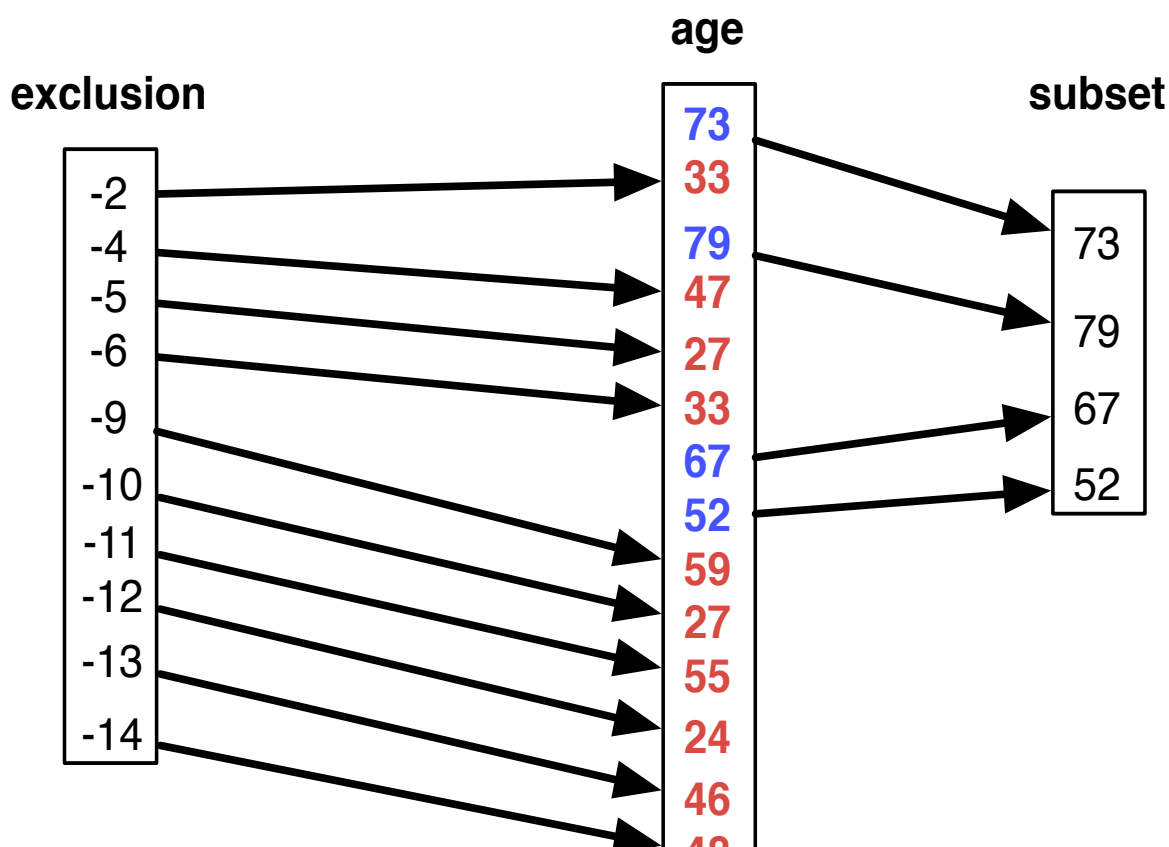
SUBSETTING: Subset by position

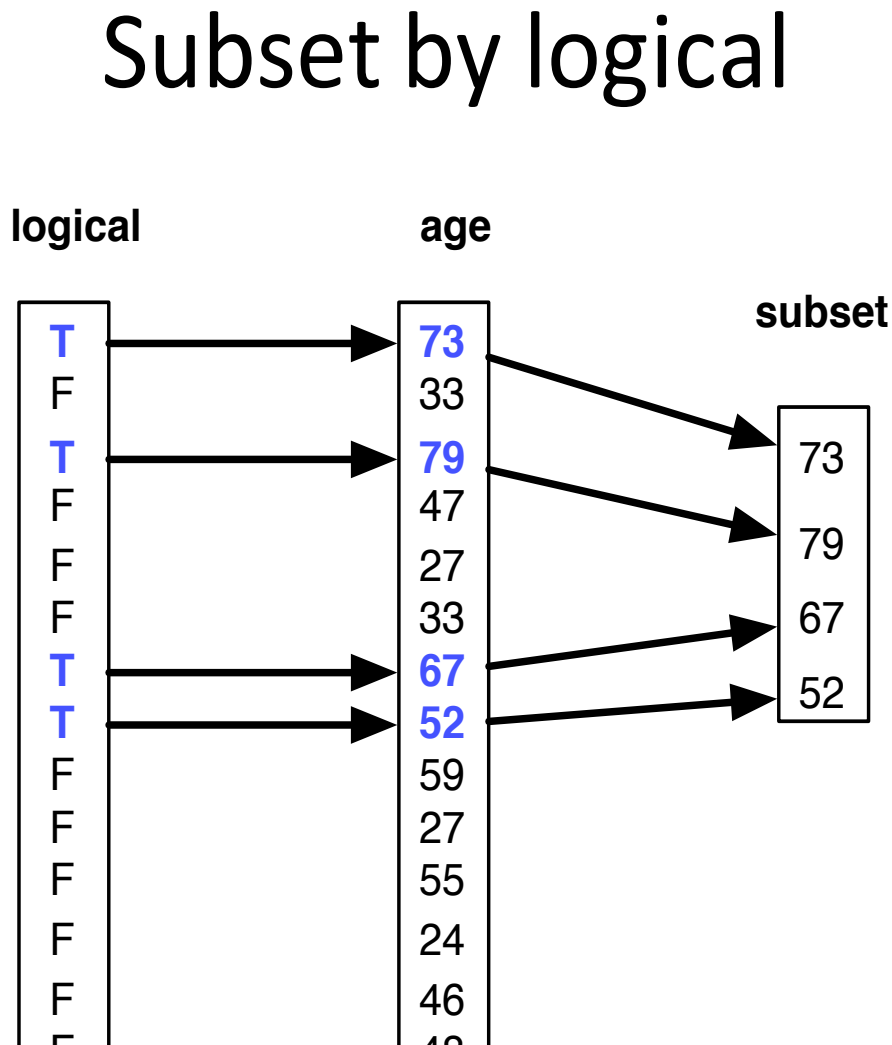


Subset by position (2)

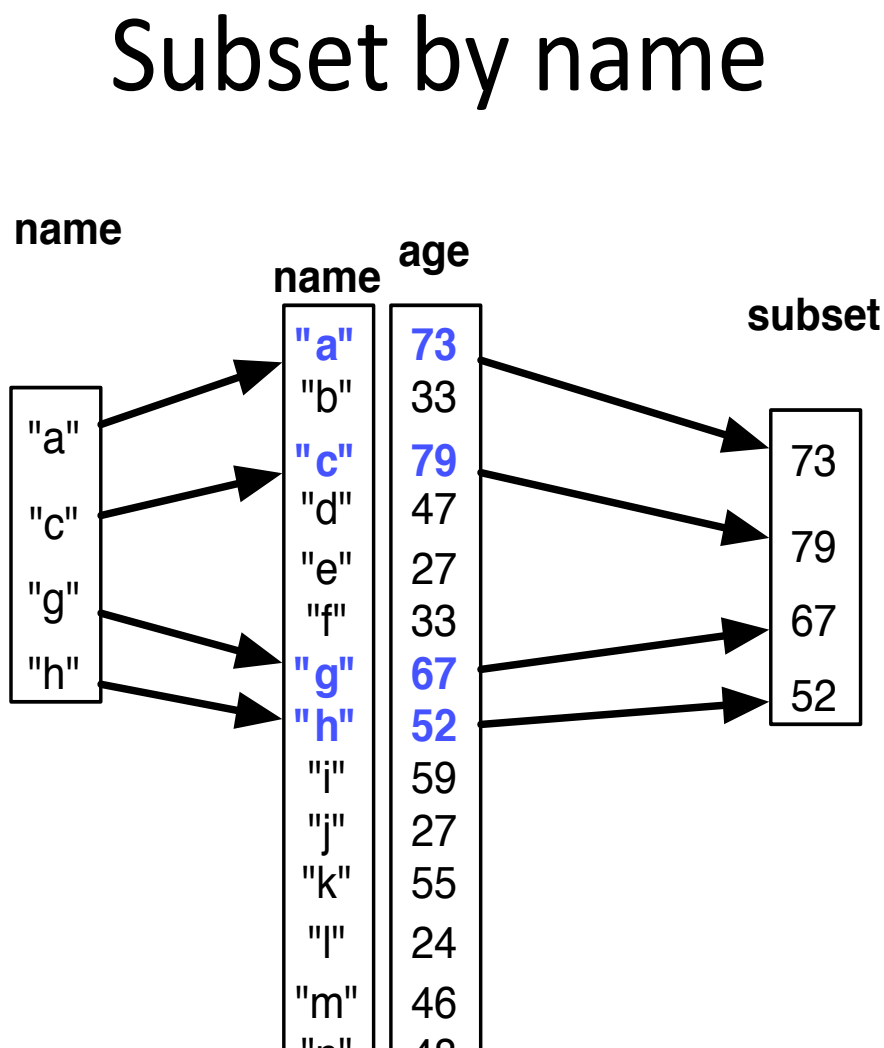


Subset by exclusion




Subset by logical
logical
age
subset
T
F
T
F
F
F
T
T
F
F
F
F
F
73
33
79
47
27
33
67
52
59
27
55
24
46
73
79
67
52

Subset by name



Five ways to subset a vector

- Position - indices of the element you want
- Exclusion - indices of elements to exclude
- Logical - logical vector the same length as the vector being subset. Keep the elements corresponding to TRUE.
- Name - character vector of names of elements to keep. Vector being subsetted must have names associated with elements
- All - all the elements

Logical/relational operators

- In addition to operators such as $+$, -, *, and / R also has logical operators
- They are relational operators $>, <, >=, <=, !=$, and $==$
- These return a value of TRUE or FALSE
- They are also vectorized operations

Examples

```
4 < 3
```

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

```
"a"=="A"
```

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

```
"A"=="A"
```

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

```
4!=3
```

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

```
fweight > 150
```

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

```
fsex!="m"
```

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

```
fbmi
```

```
##        a        b        c        d        e        f
## 25.16239 21.32906 24.45884 24.48414 18.55566 28.94981 28
##        i        j        k        l        m        n
## 26.66430 30.04911 26.05364 22.64384 24.26126 22.91060
```

```
fbmi==25.16239
```

```
##     a     b     c     d     e     f     g     h
```

Weights of the women in our family

- Create a logical expression that identifies the women in the family

```
fsex=="f"
```

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

- Use this logical expression to subset the vector of fweight

```
fweight[fsex=="f"]
```

```
## [1] 124  98 190 124 166 125
```

Boolean algebra

- Boolean algebra is a mathematical formalization of the truth or falsity of statements.
- It has three operations, not, or, and and.
- Boolean algebra tells us how to evaluate the truth or falsity of compound statements that are built using these operations. For example, if A and B are statements, some compound statements are
- A and B
- (not A) or B

- The "not" operation just causes the statement following it to switch its truth value.
  So not TRUE is FALSE and not FALSE is TRUE.
- The compound statement A and B is TRUE only if both A and B are TRUE.
- The compound statement A or B is TRUE if either or both A or B is TRUE.
- In R, we write ! for "not," & for "and," and | for "or." Note: all of these are vectorized!

```r
!(fweight > 150)
```

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

```r
(fweight > 150) & (fnames == "Tom")
```

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

```r
(fweight > 150) | (fage > 65)
```

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

Two other functions: all and any

Guess what these functions are doing:

```
all(fage > 18)
```

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

```
any(fage < 18)
```

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

```
any(fweight < 150)
```

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

```
all(fweight < 150)
```

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

Examples: of all and any

```
fage < 50
```

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

```
fsex == "f"
```

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

```
!foverWt
```

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

```
(fsex == "m") &(fheight < 70)
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FAI
```

Examples (contd)

Previous: Under 50, Women, Not over weight, Males who are under 70 inches tall

```
fbmi[fnames =="Tim" | fnames == "Tom" ]

##         a         i         j
## 25.16239 26.66430 30.04911

fbmi[fheight >72] = NA
fage[fsex == "f"] =
 fage[fsex == "f"] + 5
```

Above: BMI of Tim and Tom, Assigns BMI an NA for those over 72 inches tall, Add 5 years to all female ages

Use logical expressions to obtain the following subsets

```
fage[ !foverWt ]

## [1] 38 79 47 32 57 24 46 53

fsex[ fage > 50 ]

## [1] m m m f m f f
## Levels: f m

fbmi[ fheight == max(fheight) ]

##        a        b        c        d        e        f
## 25.16239 21.32906 24.45884 24.48414 18.55566 28.94981 28
##        i        j        k        l        m        n
## 26.66430 30.04911 26.05364 22.64384 24.26126 22.91060
```

Ages of all non-overweight members of the family, Genders of those over 50, BMI of the tallest member of the family

Creating vectors

Concatenate:

```
c(3, 2, 1)

## [1] 3 2 1

c(bob =3, alice = 2, john = 1)

##   bob alice  john
##     3     2     1
```

- A vector of three numbers, 3, 2, 1, in that order
- Elements in a vector this time with names

Subset vector based on names

```
fheight[c("a", "c", "f")]

## [1] NA NA NA

fheight[c("a", "f", "f", "c")]

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

- Order of names determines order in subset
- If we repeat a name we get the element multiple times

Construct vectors of sequences

```
1:3
```

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

```
10:6
```

```
## [1] 10  9  8  7  6
```

```
1.1:5.7
```

```
## [1] 1.1 2.1 3.1 4.1 5.1
```

```
5.7:-1.1
```

```
## [1]  5.7  4.7  3.7  2.7  1.7  0.7 -0.3
```

- Convenient way to create vectors containing a sequence of numbers

seq function to construct vectors of sequences

```
seq(1, 6, by = 2)
```

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

```
seq(1, 6, length = 3)
```

```
## [1] 1.0 3.5 6.0
```

```
seq(to = 6, length = 3, by = 2)
```

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

```
seq(from = 1, length = 3, by = 2)
```

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

- Arguments: from, to, by, length

Use seq to subset vector

```
fbmi[seq(from = 1, to = length(fbmi),
         by = 2)]

##        a        c        e        g        i        k
## 25.16239 24.45884 18.55566 28.18797 26.66430 26.05364 24
```

rep command

```
rep(3,2)

## [1] 3 3

x = c(7,1,3)
rep(x, 2)

## [1] 7 1 3 7 1 3

rep(x, c(3, 2, 1))

## [1] 7 7 7 1 1 3

rep(x, each = 2)

## [1] 7 7 1 1 3 3
```

Repeat characters

```
flastnames = rep("Smith",times = length(fbmi))
flastnames = character(length = length(fbmi))
flastnames[ ] = "Smith"
```

- vector where repeat "Smith" multiple (length of fbmi) times
- vector of characters multiple (length of fbmi) times
- each element of this vector gets "Smith"

Producing vectors without typing all values out

```
rep(seq(0, 8, by = 2), each = 5)
```

```
##  [1] 0 0 0 0 0 2 2 2 2 2 4 4 4 4 4 6 6 6 6 6 8 8 8 8 8
```

```
rep(1:5, 5)
```

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

```
rep(1:5, 5) + rep(0:4, each = 5)
```

```
##  [1] 1 2 3 4 5 2 3 4 5 6 3 4 5 6 7 4 5 6 7 8 5 6 7 8 9
```

- Code to produce 0 0 0 0 0 2 2 2 2 2 4 4 4 4 4 6 6 6 6 6 8 8 8 8 8?

sort function

```
fage

##  [1] 77 38 79 47 32 38 67 57 59 27 60 24 46 53

sort(fage)

##  [1] 24 27 32 38 38 46 47 53 57 59 60 67 77 79

sort(fage, decreasing = TRUE)

##  [1] 79 77 67 60 59 57 53 47 46 38 38 32 27 24
```

order function

```
fage

##  [1] 77 38 79 47 32 38 67 57 59 27 60 24 46 53

order(fage)

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

order tells us 12th element of fage is smallest, the 5th is the second smallest,... This function has a decreasing argument too.

Assign values to elements of a vector

```
fheight

## [1] 70

fheight[2] = 61
fheight[-13] = 62
fheight["e"] = 67
fheight[overWt] = NA

## Error in fheight[overWt] = NA: object 'overWt' not
found

fheight[] = 70
fheight = 70
```

By inclusion, exclusion, name, logical, all

```
fheight[foverWt]

## [1] 70 NA NA NA NA NA

fheight

## [1] 70

fheight[] = 70
fheight

## [1] 70

fheight = 70
```

By inclusion, exclusion, name, logical, all

Summary of functions

- c()
- :
- seq()
- rep()
- sort()
- order()