Module 4: data manipulation

BIOSTAT 213: Introduction to Computing in the R Software Environment

Yea-Hung Chen, PhD, MS

Wednesday, February 6, 2019

Logical objects

is.element(), revisited

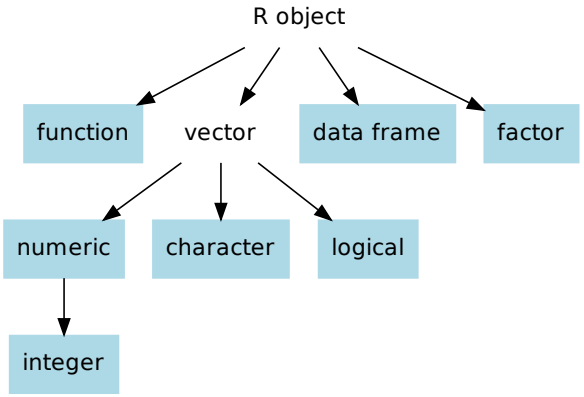
- is each of 1:8 in c(3,7,9)?

```
> tt<-is.element(1:8,c(3,7,9))
> tt
[1] FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE
> class(tt)
[1] "logical"
> is.vector(tt)
[1] TRUE
```

Logical objects

- logical objects store true/false values
- values appear as `TRUE` or `FALSE`
- logical objects are vectors

Logical objects



Equality

- for equality, use 2 equal signs (==):

```
> 3.14==pi
[1] FALSE
```

Equality

```
> years<-2017:2020
> is.2017<-years==2017
> years
[1] 2017 2018 2019 2020
> is.2017
[1]  TRUE FALSE FALSE FALSE
```

Equality

- I often add parentheses for readability, but they are not actually necessary:

```
> is.2017<-years==2017
> is.2017
[1]  TRUE FALSE FALSE FALSE
> is.2017<-(years==2017)
> is.2017
[1]  TRUE FALSE FALSE FALSE
```

Equality

- the equality operator also works with character objects:

```
> animals<-c('giraffe','zebra','elephant','lion')
> class(animals)
[1] "character"
> is.elephant<-(animals=='elephant')
> animals
[1] "giraffe"  "zebra"    "elephant" "lion"
> is.elephant
[1] FALSE FALSE  TRUE FALSE
```

Inequality

- for inequality, use `!=`:

```
> not.2017<-(years!=2017)
> years
[1] 2017 2018 2019 2020
> not.2017
[1] FALSE  TRUE  TRUE  TRUE
```

OR

- for OR, use the vertical bar (|):

```
> ('elephant'=='elephant')|(3==3)
[1] TRUE
> ('elephant'=='elephant')|(3==-3)
[1] TRUE
> ('elephant'=='giraffe')|(3==3)
[1] TRUE
> ('elephant'=='giraffe')|(3==-3)
[1] FALSE
```

OR

```
> is.2017<-(years==2017)
> is.lion<-(animals=='lion')
> years
[1] 2017 2018 2019 2020
> is.2017
[1]  TRUE FALSE FALSE FALSE
> animals
[1] "giraffe"  "zebra"    "elephant" "lion"
> is.lion
[1] FALSE FALSE FALSE  TRUE
> is.2017|is.lion
[1]  TRUE FALSE FALSE  TRUE
```

OR

- as with mathematical operations, the vectors need not be of the same length: R will recycle the shorter vector as many times as necessary
- as mentioned in Module 1's lab, however, I find this confusing and avoid it

```
> is.2<-(1:4==2)
> is.3<-(1:2==3)
> is.2
[1] FALSE  TRUE FALSE FALSE
> is.3
[1] FALSE FALSE
> is.2|is.3
[1] FALSE  TRUE FALSE FALSE
```

AND

- for AND, use the ampersand (&):

```
> ('elephant'=='elephant')&(3==3)
[1] TRUE
> ('elephant'=='elephant')&(3==-3)
[1] FALSE
> ('elephant'=='giraffe')&(3==3)
[1] FALSE
> ('elephant'=='giraffe')&(3==-3)
[1] FALSE
```

AND

```
> is.2020<-(years==2020)
> is.lion<-(animals=='lion')
> years
[1] 2017 2018 2019 2020
> is.2020
[1] FALSE FALSE FALSE  TRUE
> animals
[1] "giraffe"  "zebra"    "elephant" "lion"
> is.lion
[1] FALSE FALSE FALSE  TRUE
> is.2020&is.lion
[1] FALSE FALSE FALSE  TRUE
```

Numerical comparisons

- < for less than
- <= for less than or equal to
- > for greater than
- >= for greater than or equal to

Numerical comparisons

```
> 3.14<pi
[1] TRUE
> 1:6<=3
[1]  TRUE  TRUE  TRUE FALSE FALSE FALSE
> 1:5>4
[1] FALSE FALSE FALSE FALSE  TRUE
```

Numerical comparisons

```
> x<-1:6
> y<-c(3,6,2,1,2,4)
> x
[1] 1 2 3 4 5 6
> y
[1] 3 6 2 1 2 4
> x>y
[1] FALSE FALSE  TRUE  TRUE  TRUE  TRUE
```

Negation

- for negation, use the exclamation point (!):

```
> years
[1] 2017 2018 2019 2020
> is.element(years,c(2016,2018))
[1] FALSE  TRUE FALSE FALSE
> !is.element(years,c(2016,2018))
[1]  TRUE FALSE  TRUE  TRUE
```

Negation

```
> years==2017
[1]  TRUE FALSE FALSE FALSE
> !years==2017
[1] FALSE  TRUE  TRUE  TRUE
> !(years==2017)
[1] FALSE  TRUE  TRUE  TRUE
> years!=2017
[1] FALSE  TRUE  TRUE  TRUE
```

Negation

```
> years>=2018
[1] FALSE  TRUE  TRUE  TRUE
> !years>=2018
[1]  TRUE FALSE FALSE FALSE
> !(years>=2018)
[1]  TRUE FALSE FALSE FALSE
> years<2018
[1]  TRUE FALSE FALSE FALSE
```

Logical objects and mathematical operations

```
> xx<-is.element(years,c(2016,2018))
> xx
[1] FALSE  TRUE FALSE FALSE
> xx*1
[1] 0 1 0 0
> xx+1
[1] 1 2 1 1
> sum(xx)
[1] 1
```

- in mathematical operations, `TRUE` is treated as 1 and `FALSE` is treated as 0

Missing values

Missing values

- there are missing values in the NHANES data:

```
> head(ii$bmxwt,n=9)
[1]  78.3  89.5  88.9  52.0 105.0  93.4    NA  61.8  65.3
```

- missing values appear in R as `NA`

Missing values and `read.csv()`

- `read.csv()` automatically recognized the empty "cells" in `nhanes.csv` as missing values
- use the `na.strings` argument if your CSV file uses other character codings for missing values, such as `.`

is.na()

```
> animals
[1] "giraffe"  "zebra"    "elephant" "lion"
> is.na(animals)
[1] FALSE FALSE FALSE FALSE
> head(ii$bmxwt,n=9)
[1]  78.3  89.5  88.9  52.0 105.0  93.4    NA  61.8  65.3
> is.na(head(ii$bmxwt,n=9))
[1] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
```

Checking for missing values

```
> table(is.na(ii$bmxwt))

FALSE  TRUE
 5533   236
```

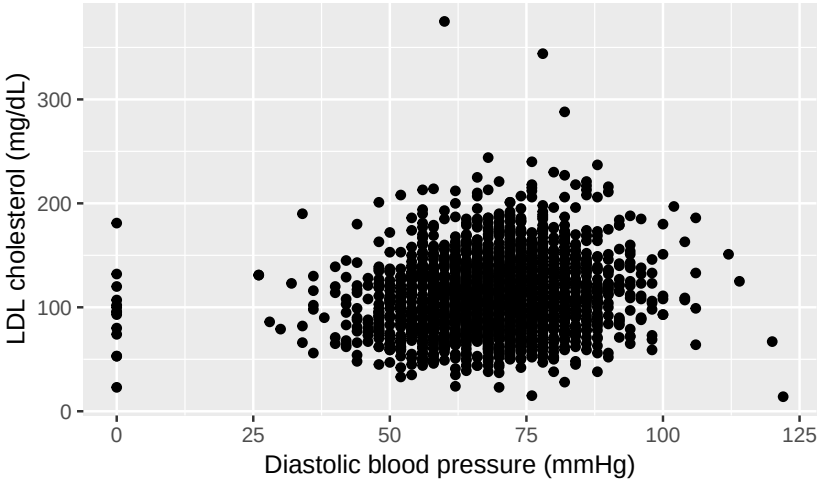
Assigning missing values

```
> x<-15
> x
[1] 15
> is.na(x)<-TRUE
> x
[1] NA
```

Assigning missing values

```
> x<-1:7
> x
[1] 1 2 3 4 5 6 7
> is.na(x)
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> is.na(x)<-(x>5)
> x
[1]  1  2  3  4  5 NA NA
> is.na(x)
[1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE
```

Assigning missing values



Assigning missing values

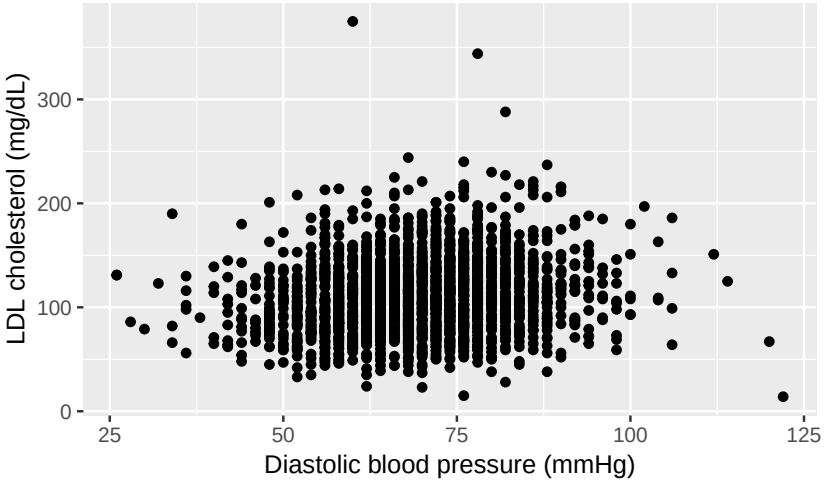
```
> table(is.na(ii$bpxdi1))

FALSE  TRUE
 5111   658
> is.na(ii$bpxdi1)<-(ii$bpxdi1<=10)
> table(is.na(ii$bpxdi1))

FALSE  TRUE
 5080   689
```

- you should, of course, consult with scientific experts and those involved in data collection (where possible) prior to such recoding

Assigning missing values



Preservation of missing values

```
> x
[1]  1  2  3  4  5 NA NA
> x+3
[1]  4  5  6  7  8 NA NA
> x*2
[1]  2  4  6  8 10 NA NA
> x>=3
[1] FALSE FALSE  TRUE  TRUE  TRUE    NA    NA
```

- missing values are generally automatically preserved

Categorization

Categorization

```
> ii$agegroup3<-(ii$ridageyr>=20)+(ii$ridageyr>=40)+
+   (ii$ridageyr>=60)
> ii$agegroup3<-factor(ii$agegroup3,1:3)
> levels(ii$agegroup3)<-c('[20,40)','[40,60)','60+')
> table(ii$agegroup3,useNA='always')

[20,40) [40,60)     60+    <NA>
   1954    1974    1841       0
```

- the first line involves mathematical operations on logical vectors

Categorization

- equivalently:

```
> ii$agegroup3<-with(ii,(ridageyr>=20)+(ridageyr>=40)+
+                        (ridageyr>=60))
> ii$agegroup3<-factor(ii$agegroup3,1:3)
> levels(ii$agegroup3)<-c('[20,40)','[40,60)','60+')
> table(ii$agegroup3,useNA='always')

[20,40) [40,60)     60+    <NA>
   1954    1974    1841       0
```

- `with()` allows for omission of dollar-sign notation
- the first argument is the data frame

Categorization

```
> ii$hdlcat<-with(ii,(lbdhdd>=0)+(lbdhdd>=40)+(lbdhdd>=60))
> ii$hdlcat<-factor(ii$hdlcat,1:3)
> levels(ii$hdlcat)<-c('Low','Medium','High')
> table(ii$hdlcat,useNA='always')

   Low Medium   High   <NA>
  1043   2768   1531    427
```

- the first line involves mathematical operations on logical vectors
- it also involves preservation of missing values

Categorization

- alternatively, use `cut()`:

```
> ii$hdlcat<-cut(ii$lbdhdd,breaks=c(0,40,60,200),right=FALSE)
> table(ii$hdlcat,useNA='always')

  [0,40)  [40,60) [60,200)     <NA>
    1043     2768     1531      427
> class(ii$hdlcat)
[1] "factor"
```

- `right=FALSE` specifies left-inclusive intervals

Indexing

Using logical vectors

Indexing using logical vectors

```
> animals
[1] "giraffe"  "zebra"    "elephant" "lion"
> animals[c(TRUE,FALSE,TRUE,FALSE)]
[1] "giraffe"  "elephant"
```

- the brackets indicate indexing
- indexing refers to taking a subset of the data
- the logical vector inside the brackets should have the same length as the vector you are indexing

Indexing using logical vectors

```
> animals
[1] "giraffe"  "zebra"    "elephant" "lion"
> animals[is.element(animals,c('lion','kangaroo','giraffe'))]
[1] "giraffe" "lion"
```

Indexing using logical vectors

```
> animals
[1] "giraffe"  "zebra"    "elephant" "lion"
> animals[substr(animals,0,1)=='e']
[1] "elephant"
```

Indexing using logical vectors

```
> years
[1] 2017 2018 2019 2020
> years[years<=2018]
[1] 2017 2018
```

Indexing using logical vectors

```
> ii$bpxdi1[!is.na(ii$bpxdi1)&(ii$bpxdi1<=30)]
[1] 26 28 30 30 28 26
```

Indexing using logical vectors

```
> ii$seqn[!is.na(ii$bmxwt)&(ii$bmxwt<=35)]
[1] 75924 80201
```

Indexing using logical vectors

```
> ii$bmxwt[(ii$ridageyr==20)&(ii$gender=='Male')]
 [1]  86.7  77.0  60.1  63.6  74.6  56.3  66.8  64.9  70.6
[10]  83.1  91.6  83.0  56.9  40.4  85.3  64.0  73.6  81.6
[19]  69.9  82.5  99.5 109.1 105.3  89.1  78.5 123.1  59.4
[28] 142.7  71.4 110.8  93.1  69.4  77.2 152.2  74.7  92.7
[37]  68.8  77.2 101.3 122.0  70.3  93.0  69.9  69.0  66.2
[46]  49.5  82.6  95.0    NA  52.8  95.6  79.3  86.7  72.1
[55] 116.5
```

Indexing using logical vectors

```
> mean(ii$bmxwt[ii$gender=='Female'],na.rm=TRUE)
[1] 76.46794
> mean(ii$bmxwt[ii$gender=='Male'],na.rm=TRUE)
[1] 86.91558
```

- indexing is very useful for stratified analysis

Indexing using logical vectors

```
> names(ii)[nchar(names(ii))==3]
[1] "fs1" "fs2" "fs3" "vwa" "vra"
```

Indexing using logical vectors

- in all the examples thus far, we've been indexing 1-dimensional objects
- it is also possible to index 2-dimensional objects

Indexing using logical vectors

```
> mm<-lm(bpxdi1~bmxbmi+ridageyr+gender,data=ii)
> summary(mm)$coefficients
               Estimate  Std. Error   t value     Pr(>|t|)
(Intercept) 62.39236133 0.842544284 74.052323 0.000000e+00
bmxbmi       0.21184677 0.023446381  9.035372 2.287260e-19
ridageyr     0.01052384 0.009310638  1.130302 2.584028e-01
genderMale   2.71320301 0.324881574  8.351360 8.635721e-17
```

Indexing using logical vectors

```
> ee<-summary(mm)$coefficients
> ee[c(FALSE,TRUE,TRUE,TRUE),c(TRUE,FALSE,FALSE,TRUE)]
              Estimate     Pr(>|t|)
bmxbmi      0.21184677 2.287260e-19
ridageyr    0.01052384 2.584028e-01
genderMale 2.71320301 8.635721e-17
```

- the first logical vector is for the rows
- the second logical vector is for the columns
- the comma in between the 2 logical vectors distinguishes the 2 dimensions

Indexing using logical vectors

```
> ee<-summary(mm)$coefficients
> ee[,c(TRUE,FALSE,FALSE,TRUE)]
               Estimate     Pr(>|t|)
(Intercept) 62.39236133 0.000000e+00
bmxbmi       0.21184677 2.287260e-19
ridageyr     0.01052384 2.584028e-01
genderMale   2.71320301 8.635721e-17
```

- the first dimension is empty here, so all of the rows are returned

Indexing using logical vectors

```
> ee[c(FALSE,TRUE,TRUE,TRUE),]
             Estimate  Std. Error  t value     Pr(>|t|)
bmxbmi     0.21184677 0.023446381 9.035372 2.287260e-19
ridageyr   0.01052384 0.009310638 1.130302 2.584028e-01
genderMale 2.71320301 0.324881574 8.351360 8.635721e-17
```

- the second dimension is empty here, so all of the columns are returned

Using numeric vectors

Indexing using numeric vectors

```
> animals
[1] "giraffe"  "zebra"    "elephant" "lion"
> animals[3]
[1] "elephant"
> animals[c(1,3)]
[1] "giraffe"  "elephant"
> animals[2:3]
[1] "zebra"    "elephant"
```

- the numeric vector inside the brackets should include the positions of the desired elements

Indexing using numeric vectors

```
> animals
[1] "giraffe"  "zebra"    "elephant" "lion"
> animals[length(animals)]
[1] "lion"
> animals[2:length(animals)]
[1] "zebra"    "elephant" "lion"
```

Indexing using numeric vectors

```
> ii$bmxwt[1:10]
 [1]  78.3  89.5  88.9  52.0 105.0  93.4    NA  61.8  65.3
[10]  47.1
> ii$bmxwt[seq(1,100,10)]
 [1]  78.3 102.4  96.0  60.9  74.0  95.5  73.9  83.9  56.3
[10]  63.7
```

Indexing using numeric vectors

```
> animals
[1] "giraffe"  "zebra"    "elephant" "lion"
> animals[-1]
[1] "zebra"    "elephant" "lion"
> animals[c(-1,-3)]
[1] "zebra" "lion"
> animals[c(-1,-length(animals))]
[1] "zebra"    "elephant"
```

Indexing using numeric vectors

```
> animals
[1] "giraffe"  "zebra"    "elephant" "lion"
> sort(animals)
[1] "elephant" "giraffe"  "lion"     "zebra"
> order(animals)
[1] 3 1 4 2
> animals[order(animals)]
[1] "elephant" "giraffe"  "lion"     "zebra"
```

Indexing using numeric vectors

```
> levels(ii$bpcat)
[1] "Elevated"            "Hypertensive crisis"
[3] "Normal"              "Stage 1"
[5] "Stage 2"
> ii$bpcat<-factor(ii$bpcat,levels(ii$bpcat)[c(3,1,4,5,2)])
> levels(ii$bpcat)
[1] "Normal"              "Elevated"
[3] "Stage 1"             "Stage 2"
[5] "Hypertensive crisis"
```

Indexing using numeric vectors

```
> ee
                Estimate  Std. Error   t value     Pr(>|t|)
(Intercept) 62.39236133 0.842544284 74.052323 0.000000e+00
bmxbmi       0.21184677 0.023446381  9.035372 2.287260e-19
ridageyr     0.01052384 0.009310638  1.130302 2.584028e-01
genderMale   2.71320301 0.324881574  8.351360 8.635721e-17
```

Indexing using numeric vectors

```
> ee[3,4]
[1] 0.2584028
```

- this indicates the 3rd row and 4th column
- this is analogous to the indexing seen in spreadsheet programs such as Excel
- in Excel, the 3rd row and 4th column is D3

Indexing using numeric vectors

```
> ee[2:4,c(1,4)]
              Estimate     Pr(>|t|)
bmxbmi      0.21184677 2.287260e-19
ridageyr    0.01052384 2.584028e-01
genderMale 2.71320301 8.635721e-17
```

Indexing using numeric vectors

```
> ee[-1,c(1,4)]
              Estimate     Pr(>|t|)
bmxbmi     0.21184677 2.287260e-19
ridageyr   0.01052384 2.584028e-01
genderMale 2.71320301 8.635721e-17
```

Indexing using numeric vectors

```
> ee[,c(1,4)]
                Estimate     Pr(>|t|)
(Intercept) 62.39236133 0.000000e+00
bmxbmi       0.21184677 2.287260e-19
ridageyr     0.01052384 2.584028e-01
genderMale   2.71320301 8.635721e-17
```

Indexing using numeric vectors

```
> ee[order(rownames(ee)),]
               Estimate  Std. Error   t value     Pr(>|t|)
(Intercept) 62.39236133 0.842544284 74.052323 0.000000e+00
bmxbmi       0.21184677 0.023446381  9.035372 2.287260e-19
genderMale   2.71320301 0.324881574  8.351360 8.635721e-17
ridageyr     0.01052384 0.009310638  1.130302 2.584028e-01
```

Indexing using numeric vectors

```
> ee[order(ee[,4]),]
               Estimate  Std. Error   t value     Pr(>|t|)
(Intercept) 62.39236133 0.842544284 74.052323 0.000000e+00
bmxbmi       0.21184677 0.023446381  9.035372 2.287260e-19
genderMale   2.71320301 0.324881574  8.351360 8.635721e-17
ridageyr     0.01052384 0.009310638  1.130302 2.584028e-01
```

Using character vectors

Indexing using character vectors

```
> tt<-t.test(ii$bmxbmi~ii$gender)
> tt$estimate
mean in group Female   mean in group Male
            29.65621             28.49810
> tt$estimate['mean in group Female']
mean in group Female
            29.65621
> tt$estimate['mean in group Male']
mean in group Male
           28.4981
```

Indexing using character vectors

```
> head(ii[,c('bmxwt','bmxht')])
  bmxwt bmxht
1  78.3 171.3
2  89.5 176.8
3  88.9 175.3
4  52.0 162.4
5 105.0 158.7
6  93.4 161.8
```

Indexing using character vectors

```
> mean(ii$bmxwt,na.rm=TRUE)
[1] 81.46234
> mean(ii[,'bmxwt'],na.rm=TRUE)
[1] 81.46234
```

Indexing using character vectors

```
> rownames(ee)
[1] "(Intercept)" "bmxbmi"      "ridageyr"    "genderMale"
> colnames(ee)
[1] "Estimate"   "Std. Error" "t value"    "Pr(>|t|)"
> ee[c('bmxbmi','ridageyr'),c('Estimate','Std. Error')]
           Estimate  Std. Error
bmxbmi    0.21184677 0.023446381
ridageyr  0.01052384 0.009310638
```

Indexing using character vectors

```
> confint(mm)['ridageyr',]
       2.5 %        97.5 %
-0.007729075  0.028776747
```

Using combined approaches

Indexing using combined approaches

```
> ii$bmxwt[(ii$ridageyr==20)&(ii$gender=='Male')]
 [1]  86.7  77.0  60.1  63.6  74.6  56.3  66.8  64.9  70.6
[10]  83.1  91.6  83.0  56.9  40.4  85.3  64.0  73.6  81.6
[19]  69.9  82.5  99.5 109.1 105.3  89.1  78.5 123.1  59.4
[28] 142.7  71.4 110.8  93.1  69.4  77.2 152.2  74.7  92.7
[37]  68.8  77.2 101.3 122.0  70.3  93.0  69.9  69.0  66.2
[46]  49.5  82.6  95.0    NA  52.8  95.6  79.3  86.7  72.1
[55] 116.5
> ii$bmxwt[(ii$ridageyr==20)&(ii$gender=='Male')][5:10]
[1] 74.6 56.3 66.8 64.9 70.6 83.1
```

Indexing using combined approaches

```
> ii[1:5,c('bmxwt','bmxht')]
  bmxwt bmxht
1  78.3 171.3
2  89.5 176.8
3  88.9 175.3
4  52.0 162.4
5 105.0 158.7
```

Indexing using combined approaches

```
> ee[1,'Estimate']
[1] 62.39236
> ee[c('bmxbmi','ridageyr'),c(1,4)]
           Estimate     Pr(>|t|)
bmxbmi    0.21184677 2.287260e-19
ridageyr  0.01052384 2.584028e-01
```

Indexing using combined approaches

```
> ee[,4]
 (Intercept)       bmxbmi     ridageyr   genderMale
0.000000e+00 2.287260e-19 2.584028e-01 8.635721e-17
> ee[ee[,4]<=0.05,]
             Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 62.3923613 0.84254428 74.052323 0.000000e+00
bmxbmi       0.2118468 0.02344638  9.035372 2.287260e-19
genderMale   2.7132030 0.32488157  8.351360 8.635721e-17
```

Indexing using combined approaches

```
> ee[,'Pr(>|t|)']
 (Intercept)       bmxbmi     ridageyr   genderMale
0.000000e+00 2.287260e-19 2.584028e-01 8.635721e-17
> ee[ee[,'Pr(>|t|)']<=0.05,]
             Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 62.3923613 0.84254428 74.052323 0.000000e+00
bmxbmi       0.2118468 0.02344638  9.035372 2.287260e-19
genderMale   2.7132030 0.32488157  8.351360 8.635721e-17
```

Brief Exercise

Brief Exercise

You may work in groups of 2–3. Save your code in a script file.

1. Import `presidencies.csv` into R, using `read.csv()`. Print the variable names using `names()`. Explore the data using `head()`.
2. Find all presidents from the state of Virginia.
3. The `order` variable indicates the order of each presidency. Use that variable to find the president who served the 33rd term.
4. List the name (`name`) and start date (`start`) of all presidents from Ohio.

subset()

subset()

```
> ss<-subset(ii,ridageyr<=30)
> nrow(ss)
[1] 1073
```

- the first argument should be the object you want to subset
- the second argument should be some logical vector
- note that you do not need to use dollar-sign notation

subset()

```
> ss<-subset(ii,ridageyr<=30)
> nrow(ss)
[1] 1073
> ss<-ii[ii$ridageyr<=30,]
> nrow(ss)
[1] 1073
```

subset()

```
> ss<-ii[ii$bmxwt<=40,]
> nrow(ss)
[1] 246
> ss<-subset(ii,bmxwt<=40)
> nrow(ss)
[1] 10
> ss<-ii[!is.na(ii$bmxwt)&(ii$bmxwt<=40),]
> nrow(ss)
[1] 10
```

- subset() automatically excludes missing values

Indexing and assignment

Indexing and assignment

```
> animals
[1] "giraffe"  "zebra"    "elephant" "lion"
> animals[3]<-'kangaroo'
> animals
[1] "giraffe"  "zebra"    "kangaroo" "lion"
```

Indexing and assignment

```
> ii$agegroup2<-''
> ii$agegroup2[ii$ridageyr<30]<-'20-29'
> ii$agegroup2[ii$ridageyr>=30]<-'30+'
> ii$agegroup2<-factor(ii$agegroup2,c('20-29','30+'))
> table(ii$agegroup2,useNA='always')

20-29   30+  <NA>
  956  4813     0
```

- the first line is not strictly necessary but may help prevent errors, particularly if the variable already exists

Indexing and assignment

```
> table(ii$dmdborn4,ii$dmdcitzn,useNA='always')

         1    2    7    9 <NA>
1     4119    0    0    0    0
2      890  750    4    2    0
77       0    0    0    0    4
<NA>     0    0    0    0    0
```

Indexing and assignment

```
> ii$cs<-''
> ii$cs[ii$dmdborn4==1]<-'Birthright citizen'
> ii$cs[(ii$dmdborn4==2)&(ii$dmdcitzn==1)]<-'Naturalized citizen'
> ii$cs[(ii$dmdborn4==2)&(ii$dmdcitzn==2)]<-'Not a citizen'
> ii$cs<-factor(ii$cs,c('Birthright citizen',
+                       'Naturalized citizen','Not a citizen'))
> table(ii$cs,useNA='always')

 Birthright citizen Naturalized citizen       Not a citizen
               4119                  890                 750
               <NA>
                 10
```

ifelse()

ifelse()

```
> score<-c(92,95,87,85,82,98)
> ifelse(score>=90,'A','B')
[1] "A" "A" "B" "B" "B" "A"
```

ifelse()

```
> ii$agegroup2<-ifelse(ii$ridageyr<30,'20-29','30+')
> ii$agegroup2<-factor(ii$agegroup2,c('20-29','30+'))
> table(ii$agegroup2,useNA='always')

20-29   30+  <NA>
  956  4813     0
```

ifelse()

```
> plot(ii$bmxwt~ii$bmxht,
+      col=ifelse(ii$gender=='Female','red','blue'),
+      xlab='Height (cm)',ylab='Weight (kg)')
```

ifelse()