Module 6: More data manipulation

BIOSTAT 213: Introduction to Computing in the R Software Environment

Monday, October 28, 2019

Indexing 2-dimensional objects

Using numeric vectors

Indexing using numeric vectors

- Assignment 2, revisited:

```
> mm<-lm(dr1tkcal~gender+dr1tprot+I(dr1tprot^2),data=ii)
```

- the `I()` is necessary for the squared term

Indexing using numeric vectors

```
> nn<-data.frame(gender=c('Female','Male'),
+                dr1tprot=seq(25,150,25))
> nn$predictions<-predict(mm,nn)
> nn
  gender dr1tprot predictions
1 Female       25    988.6088
2   Male       50   1658.9172
3 Female       75   1953.6557
4   Male      100   2570.6203
5 Female      125   2812.0150
6   Male      150   3375.6358
```

Indexing using numeric vectors

```
> nn[1:3,2:3]
  dr1tprot predictions
1       25    988.6088
2       50   1658.9172
3       75   1953.6557
```

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

Indexing using numeric vectors

```
> nn[,2:3]
  dr1tprot predictions
1       25    988.6088
2       50   1658.9172
3       75   1953.6557
4      100   2570.6203
5      125   2812.0150
6      150   3375.6358
```

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

Indexing using numeric vectors

```
> nn[1:3,]
  gender dr1tprot predictions
1 Female       25    988.6088
2   Male       50   1658.9172
3 Female       75   1953.6557
```

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

Indexing using numeric vectors

```
> nn[nrow(nn),ncol(nn)]
[1] 3375.636
```

Indexing using numeric vectors

```
> nn$predictions
[1]  988.6088 1658.9172 1953.6557 2570.6203 2812.0150 3375.6358
> order(nn$predictions,decreasing=TRUE)
[1] 6 5 4 3 2 1
> nn[order(nn$predictions,decreasing=TRUE),]
  gender dr1tprot predictions
6   Male      150   3375.6358
5 Female      125   2812.0150
4   Male      100   2570.6203
3 Female       75   1953.6557
2   Male       50   1658.9172
1 Female       25    988.6088
```

Using logical vectors

Indexing using logical vectors

```
> nn[c(TRUE,TRUE,TRUE,FALSE,FALSE,FALSE),c(FALSE,TRUE,TRUE)]
  dr1tprot predictions
1       25    988.6088
2       50   1658.9172
3       75   1953.6557
```

Indexing using logical vectors

```
> nn[nn$dr1tprot<=100,]
  gender dr1tprot predictions
1 Female       25    988.6088
2   Male       50   1658.9172
3 Female       75   1953.6557
4   Male      100   2570.6203
```

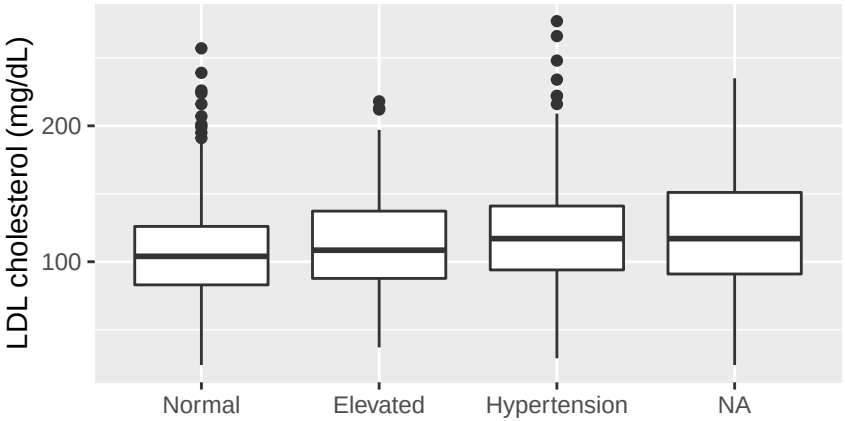
Indexing using logical vectors

```
> nn[nn$gender=='Male',]
  gender dr1tprot predictions
2   Male       50    1658.917
4   Male      100    2570.620
6   Male      150    3375.636
```

Indexing using logical vectors

```
> nn[nn$predictions<2000,]
  gender dr1tprot predictions
1 Female       25    988.6088
2   Male       50   1658.9172
3 Female       75   1953.6557
```

Indexing using logical vectors



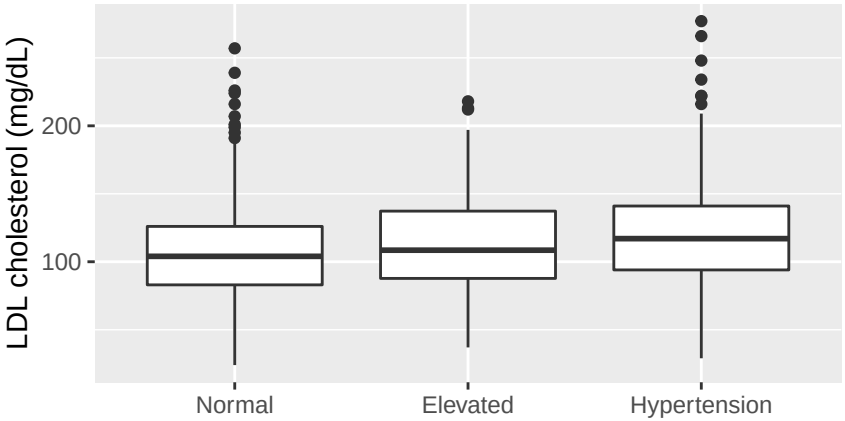
Indexing using logical vectors

- one method of omitting the NA group from the plot is to change the data argument to `ggplot()`:

```
gg<-ggplot(mapping=aes(x=bpcat,y=lbdldl),
           data=ii[!is.na(ii$bpcat),])
gg+
  geom_boxplot()+
  xlab('')+
  ylab('LDL cholesterol (mg/dL)')
```

- here we are using `ii[!is.na(ii$bpcat),]` whereas last week we used `subset(ii,!is.na(bpcat))` (more on this later)

Indexing using logical vectors



Using character vectors

Indexing using character vectors

```
> nn[,c('dr1tprot','predictions')]
  dr1tprot predictions
1       25    988.6088
2       50   1658.9172
3       75   1953.6557
4      100   2570.6203
5      125   2812.0150
6      150   3375.6358
```

Indexing using character vectors

```
> nn[,'dr1tprot']
[1]  25  50  75 100 125 150
> nn$dr1tprot
[1]  25  50  75 100 125 150
```

Indexing using character vectors

```
> mm<-lm(bpxdi1~bmxbmi+ridageyr+gender,data=ii)
> summary(mm)$coefficients
              Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 57.7316789 0.93198721 61.944711 0.000000e+00
bmxbmi       0.1860423 0.02611511  7.123932 1.237196e-12
ridageyr     0.1325029 0.01250141 10.599033 6.597068e-26
genderMale   3.1953131 0.37716468  8.471931 3.342051e-17
> ee<-summary(mm)$coefficients
```

Indexing using character vectors

```
> ee['bmxbmi',]
    Estimate   Std. Error      t value     Pr(>|t|)
1.860423e-01 2.611511e-02 7.123932e+00 1.237196e-12
```

Indexing using character vectors

```
> ee[c('bmxbmi','genderMale'),]
             Estimate Std. Error  t value     Pr(>|t|)
bmxbmi      0.1860423 0.02611511 7.123932 1.237196e-12
genderMale 3.1953131 0.37716468 8.471931 3.342051e-17
```

Using combined approaches

Indexing using combined approaches

```
> ii[1:4,c('ridageyr','bpxsy1','bmxwt')]
  ridageyr bpxsy1 bmxwt
1       62    128  94.8
2       53    146  90.4
3       56    132 109.8
4       42    100  55.2
```

- the index for the first dimension is a numeric vector
- the index for the second dimension is a character vector

Indexing using combined approaches

```
> ii[ii$bmxwt==55,c('ridageyr','gender','bmxwt')]
     ridageyr gender bmxwt
1523       59 Female    55
2178       56 Female    55
3711       49 Female    55
```

- the index for the first dimension is a logical vector
- the index for the second dimension is a character vector

Indexing using combined approaches

```
> ii[ii$bmxwt==92,1:3]
      seqn sddsrvyr ridstatr
2556 89722        9        2
2914 90585        9        2
2996 90764        9        2
3002 90786        9        2
3188 91226        9        2
```

- the index for the first dimension is a ??? vector
- the index for the second dimension is a ??? vector

Indexing using logical vectors

```
> ii[!is.na(ii$bpxsy1)&ii$bpxsy1<=85,
+    c('ridageyr','gender','bpxsy1')]
     ridageyr gender bpxsy1
280        28 Female     84
913        62   Male     84
2068       21 Female     82
2298       26   Male     84
3176       59 Female     84
3196       29 Female     84
```

- when indexing using a logical vector and there are missing values in the variable, it is necessary to include `!is.na()`
- there is a note on this on the CLE, and we'll also discuss further later today

Indexing using combined approaches

```
> ee[1,'Estimate']
[1] 57.73168
```

- the index for the first dimension is a ??? vector
- the index for the second dimension is a ??? vector

Indexing using combined approaches

```
> ee[c('bmxbmi','ridageyr'),c(1,4)]
          Estimate     Pr(>|t|)
bmxbmi   0.1860423 1.237196e-12
ridageyr 0.1325029 6.597068e-26
```

- the index for the first dimension is a ??? vector
- the index for the second dimension is a ??? vector

Indexing using combined approaches

```
> ee[ee[,4]<=0.05,]
              Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 57.7316789 0.93198721 61.944711 0.000000e+00
bmxbmi       0.1860423 0.02611511  7.123932 1.237196e-12
ridageyr     0.1325029 0.01250141 10.599033 6.597068e-26
genderMale   3.1953131 0.37716468  8.471931 3.342051e-17
```

- the index for the first dimension is a logical vector
- that logical vector involves indexing by a numeric vector

Indexing using combined approaches

```
> ee[ee[,'Pr(>|t|)']<=0.05,]
             Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 57.7316789 0.93198721 61.944711 0.000000e+00
bmxbmi       0.1860423 0.02611511  7.123932 1.237196e-12
ridageyr     0.1325029 0.01250141 10.599033 6.597068e-26
genderMale   3.1953131 0.37716468  8.471931 3.342051e-17
```

- the index for the first dimension is a logical vector
- that logical vector involves indexing by a character vector

Indexing versus `subset()`

Indexing versus `subset()`

- `subset()` is conceptually equivalent to indexing by a logical vector
- indexing overall is more powerful

Indexing versus subset()

- for a 1-dimensional object:

```
> subset(ii$bmxwt,ii$bmxwt<=40)
[1] 39.7 37.0 39.0 39.9 39.6 39.8 39.2
> ii$bmxwt[ii$bmxwt<=40]
[1] 39.7 37.0 39.0 39.9 39.6 39.8 39.2
```

Indexing versus subset()

```
> subset(ii$ridageyr,ii$bmxwt<=40)
[1] 21 18 20 23 58 64 25
> ii$ridageyr[ii$bmxwt<=40]
[1] 21 18 20 23 58 64 25
```

Indexing versus `subset()`

- as mentioned earlier (and also in the added note on the CLE), when indexing using a logical vector and there are missing values in the variable, it is necessary to include `!is.na()`

Indexing versus subset()

```
> ii$ridageyr[ii$bpxsy1<=85]
  [1] NA NA NA NA NA NA NA NA NA 28 NA NA NA NA NA NA NA NA NA NA
 [21] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
 [41] NA NA 62 NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
 [61] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
 [81] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
[101] NA NA 21 NA NA NA NA NA 26 NA NA NA NA NA NA NA NA NA NA NA
[121] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
[141] NA NA NA NA NA NA 59 NA 29 NA NA NA NA NA NA NA NA NA NA NA
[161] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
[181] NA NA NA NA NA NA NA NA NA NA NA NA NA
```

Indexing versus subset()

```
> ii$ridageyr[!is.na(ii$bpxsy1)&ii$bpxsy1<=85]
[1] 28 62 21 26 59 29
> subset(ii$ridageyr,ii$bpxsy1<=85)
[1] 28 62 21 26 59 29
```

- `!is.na()` statement is not needed in `subset()`, however

Indexing versus subset()

- for a 2-dimensional object:

```
> subset(nn,predictions<=1800)
  gender dr1tprot predictions
1 Female       25    988.6088
2   Male       50   1658.9172
> nn[nn$predictions<=1800,]
  gender dr1tprot predictions
1 Female       25    988.6088
2   Male       50   1658.9172
```

- another difference between `subset()` and indexing is that when using `subset()` on 2-dimensional objects with column names, it is not necessary to use dollar-sign notation

Indexing versus subset()

```
> ii[ii$bmxwt==37,c('ridageyr','gender','bmxwt')]
    ridageyr gender bmxwt
805       18 Female    37
```

- an equivalent approach in subset() is to add the select argument:

```
> subset(ii,bmxwt==37,select=c('ridageyr','gender','bmxwt'))
    ridageyr gender bmxwt
805       18 Female    37
```

Indexing versus subset()

```
> subset(ii,bmxwt==37,select=c('ridageyr','gender','bmxwt'))
    ridageyr gender bmxwt
805       18 Female    37
> subset(ii,bmxwt==37,select=c(ridageyr,gender,bmxwt))
    ridageyr gender bmxwt
805       18 Female    37
```

- note that it is not necessary to include quotation marks when using the `select` argument
- in comparison, quotation marks are necessary when indexing by a character vector

Indexing versus subset()

```
> ii[ii$bmxwt==37,1:3]
     seqn sddsrvyr ridstatr
805 85601        9        2
> subset(ii,bmxwt==37,select=1:3)
     seqn sddsrvyr ridstatr
805 85601        9        2
```

Indexing versus subset()

```
> ii[!is.na(ii$bpxsy1)&ii$bpxsy1==82,1:3]
      seqn sddsrvyr ridstatr
2068 88607        9        2
> subset(ii,bpxsy1==82,select=1:3)
      seqn sddsrvyr ridstatr
2068 88607        9        2
```

- as mentioned earlier, when indexing using a logical vector and there are missing values in the variable, it is necessary to include `!is.na()`
- however, `!is.na()` is not needed in `subset()`

Indexing and assignment

Indexing and assignment

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

Indexing and assignment

```
> animals
[1] "giraffe"  "zebra"    "kangaroo" "lion"
> is.na(animals[2])<-TRUE
> animals
[1] "giraffe"  NA         "kangaroo" "lion"
```

Indexing and assignment

```
> animals
[1] "giraffe"  NA         "kangaroo" "lion"
> animals[is.na(animals)]<-'elephant'
> animals
[1] "giraffe"  "elephant" "kangaroo" "lion"
```

Indexing and assignment

```
> animals
[1] "giraffe"  "elephant" "kangaroo" "lion"
> animals[nchar(animals)<=5]<-'hippopotamus'
> animals
[1] "giraffe"      "elephant"     "kangaroo"     "hippopotamus"
```

Indexing and assignment

- categorization, using an approach from last week:

```
> ii$age2<-(ii$ridageyr>=18)+(ii$ridageyr>=30)
> ii$age2<-factor(ii$age2,1:2)
> levels(ii$age2)<-c('18-29','30+')
> table(ii$age2,useNA='always')

18-29   30+  <NA>
 1042  3167     0
```

Indexing and assignment

- categorization, using indexing and assignment:

```
> ii$age2<-''
> ii$age2[ii$ridageyr<30]<-'18-29'
> ii$age2[ii$ridageyr>=30]<-'30+'
> ii$age2<-as.factor(ii$age2)
> table(ii$age2,useNA='always')

18-29   30+  <NA>
 1042  3167     0
```

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

Indexing and assignment

- this highlights another difference between indexing and `subset()`: it is not possible to combine `subset()` with assignment
- the following results in an error:

```
> ii$age2<-''
> subset(ii$age2,ii$ridageyr<30)<-'18-29'
```

- ... even though the following are equivalent:

```
> ii$age2[ii$ridageyr<30]
> subset(ii$age2,ii$ridageyr<30)
```

Indexing and assignment

```
> y<-1:5
> x<-c(1,0,1,0,1)
> is.na(x[4])<-TRUE
> y
[1] 1 2 3 4 5
> x
[1]  1  0  1 NA  1
> y[x==1]
[1]  1  3 NA  5
> y[!is.na(x)&x==1]
[1] 1 3 5
```

Indexing and assignment

```
> y[!is.na(x)&x==1]
[1] 1 3 5
> y[x==1]
[1]  1  3 NA  5
> y[x==1]<-99
> y
[1] 99  2 99  4 99
```

- interestingly, when combining indexing and object assignment, it is no longer necessary to worry about missing values!

Indexing and assignment

- born in the USA (`dmdborn4`) versus citizenship (`dmdcitzn`):

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

          1    2    7    9 <NA>
  1    2856    0    0    0    0
  2     610  735    2    5    0
  99      0    0    0    0    1
  <NA>    0    0    0    0    0
```

Indexing and assignment

```
> ii$cs<-''
> ii$cs[ii$dmdborn4==1]<-'Birthright'
> ii$cs[(ii$dmdborn4==2)&(ii$dmdcitzn==1)]<-'Naturalized'
> ii$cs[(ii$dmdborn4==2)&(ii$dmdcitzn==2)]<-'No'
> is.na(ii$cs[ii$cs==''])<-TRUE
> ii$cs<-as.factor(ii$cs)
> table(ii$cs,useNA='always')

 Birthright Naturalized          No        <NA>
       2856         610         735           8
```

ifelse()

ifelse()

```
> animals<-c('giraffe','zebra','elephant','lion')
> ifelse(nchar(animals)<=5,toupper(animals),animals)
[1] "giraffe"  "ZEBRA"    "elephant" "LION"
```

- `ifelse()` should have 3 arguments: a logical vector, a vector of values for the `TRUE` output, and a vector of values for the `FALSE` output

ifelse()

- the "replacement vectors" can be single values:

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

- each replacement value is recycled as many times as necessarys

ifelse()

- categorization revisited:

```
> ii$age2<-ifelse(ii$ridageyr<30,'18-29','30+')
> ii$age2<-as.factor(ii$age2)
> table(ii$age2,useNA='always')

18-29   30+  <NA>
 1042  3167     0
```

- note the `ifelse()` approach would only work for creating binary variables, unless you combine multiple `ifelse()`

Exercise

Exercise

You may work in groups of 2–3. Use `nhanes.csv` for all problems.

1. List the systolic blood pressure variables (`bpxsy1`, `bpxsy2`, `bpxsy3`, and `bpxsy4`) for individuals 24 years of age (`ridageyr`) or younger with a history of military service (`military`).
2. List the 10th, 30th, 50th, and 70th columns for individuals 24 years of age (`ridageyr`) or younger with a history of military service (`military`).
3. List the last row of the 10th, 30th, 50th, and 70th columns.

More on the next slide...

Exercise

4. List the 100th, 200th, and 300th rows for all columns that have a column name containing only 3 characters.
5. List the age (`ridageyr`), gender (`gender`), and daily sodium intake (`dr1tsodi`) variables for all individuals with a daily sodium intake exceeding 10,000 mg.
6. List the age (`ridageyr`), gender (`gender`), and daily sodium intake (`dr1tsodi`) variables for all males with a daily sodium intake exceeding 10,000 mg.

More on the next slide...

Exercise

7. As mentioned last week, it is possible to re-order the levels of `bmicat` as follows:

```
> ii$bmicat<-factor(ii$bmicat,levels(ii$bmicat)[c(5,4,1:3)])
```

   Apply this re-ordering if you have not already.

More on the next slide...

Exercise

8. Use the categorization method discussed last week to create a new variable for BMI category. The cutoffs should be `bmxbmi>=0`, `bmxbmi>=25`, `bmxbmi>=30`, `bmxbmi>=35`, and `bmxbmi>=40`. These correspond to Underweight or normal weight, Overweight, Class I obesity, Class II obesity, and Class III obesity, respectively. Check that your variable matches the existing `bmicat` variable.

9. Use indexing and assignment to create a new variable for BMI category, using the cutoffs mentioned above. Check that your variable matches the existing `bmicat` variable.

10. Use multiple `ifelse()` statements to create a new variable for BMI category, using the cutoffs mentioned above. Check that your variable matches the existing `bmicat` variable.