Module 5: advanced data manipulation

BIOSTAT 213: Introduction to Computing in the R Software Environment

Yea-Hung Chen, PhD, MS

Wednesday, February 13, 2019

Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using reshape()

Brief Exercise

Brief Exercise
Date objects
rbind() and cbind()
Importing other data files
Merging data
Reshaping data
Data structures
Wide to long "manually"
Wide to long using reshape()

Brief Exercise

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

1. Re-import `nhanes.csv` into R, using `read.csv()`.
2. List the 10th, 20th, 30th, ..., through 100th values of body mass index (`bmxbmi`). Are we indexing a 1-dimensional object or a 2-dimensional object? Are we indexing using a numeric, logical, or character object?
3. List the second measurement of diastolic blood pressures (`bpxdi2`) of individuals with a first measurement (`bpxdi1`) equal to 0. Are we indexing a 1-dimensional object or a 2-dimensional object? Are we indexing using a numeric, logical, or character object?

More on the next slide...

Brief Exercise
Date objects
rbind() and cbind()
Importing other data files
Merging data
Reshaping data
Data structures
Wide to long "manually"
Wide to long using reshape()

Brief Exercise

4. List the 201st through 210th rows of ID (`seqn`), age (`ridageyr`), and age group (`agegroup`). Are we indexing a 1-dimensional object or a 2-dimensional object? For the rows, are we indexing using a numeric, logical, or character object? For the columns, are we indexing using a numeric, logical, or character object?
5. Using indexing, create a subset of the data, including only individuals with 4 hours of sleep (`sleep`) per night on average. Check the number of rows. Using `subset()`, create a subset of the data, including only individuals with 4 hours of sleep (`sleep`) per night on average. Check the number of rows.

Brief Exercise
Date objects
rbind() and cbind()
Importing other data files
Merging data
Reshaping data
Data structures
Wide to long "manually"
Wide to long using reshape()

Date objects

Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using reshape()

Indexing, continued

```
> pp[seq(10,nrow(pp),10),c('start','end')]
        start        end
10  4/04/1841  4/03/1845
20  4/03/1881 19/09/1881
30   2/8/1923  4/03/1929
40 20/01/1981 20/01/1989
```


Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using reshape()

Date objects

```
> pp$end-pp$start
Warning in Ops.factor(pp$end, pp$start): '-' not meaningful
for factors
 [1] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
[19] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
[37] NA NA NA NA NA NA NA NA
```


Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using reshape()

Date objects

- the subtraction failed because R is not currently recognizing `start` and `end` as dates:

```
> class(pp$start)
[1] "factor"
> class(pp$end)
[1] "factor"
```


Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using reshape()

Date objects

- one method for converting to dates is `as.Date()`

Brief Exercise

Date objects

`rbind()` and `cbind()`

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using `reshape()`

Date objects

```
> head(pp$start)
[1] 30/04/1789 4/03/1797  4/03/1801  4/03/1809  4/03/1817
[6] 4/03/1825
44 Levels: 12/4/1945 14/9/1901 15/04/1865 ... 9/8/1974
> pp$start<-as.Date(pp$start,format='%d/%m/%Y')
```

- the `format` should replicate the way the date is stored
- for this variable, that means including `/` in between values
- `%d`: day of the month
- `%m`: month, as a number
- `%Y`: year, stored with 4 digits (if the year is stored with 2 digits, use %y)

Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using reshape()

Date objects

```
> head(pp$start)
[1] "1789-04-30" "1797-03-04" "1801-03-04" "1809-03-04"
[5] "1817-03-04" "1825-03-04"
```


Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using reshape()

Date objects

```
> class(pp$start)
[1] "Date"
> is.vector(pp$start)
[1] FALSE
```

- `start` is now a Date object, a type of date object
- Date objects look like vectors but are not vectors

Brief Exercise

Date objects

`rbind()` and `cbind()`

Importing other data files

Merging data

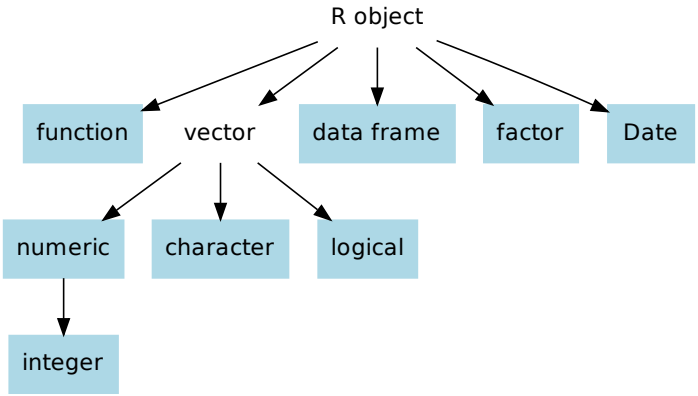
Reshaping data

Data structures

Wide to long "manually"

Wide to long using `reshape()`

Date objects



Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using reshape()

Date objects

```
> head(pp$end)
[1] 4/03/1797 4/03/1801 4/03/1809 4/03/1817 4/03/1825
[6] 4/03/1829
44 Levels: 12/4/1945 14/9/1901 15/04/1865 ... 9/8/1974
> pp$end<-as.Date(pp$end,format='%d/%m/%Y')
> head(pp$end)
[1] "1797-03-04" "1801-03-04" "1809-03-04" "1817-03-04"
[5] "1825-03-04" "1829-03-04"
> class(pp$end)
[1] "Date"
```


Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using reshape()

Date objects

```
> pp$duration<-pp$end-pp$start
> min(pp$duration)
Time difference of 31 days
> max(pp$duration)
Time difference of 4422 days
```


Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

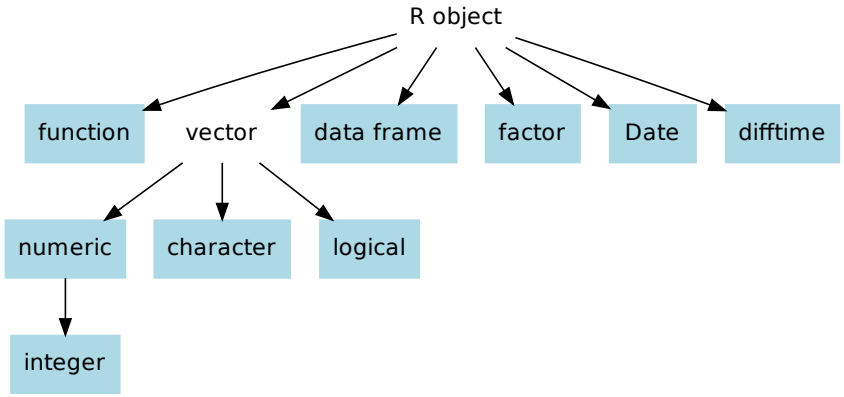
Wide to long using reshape()

difftime objects

```
> head(pp$duration)
Time differences in days
[1] 2865 1460 2922 2922 2922 1461
> class(pp$duration)
[1] "difftime"
> is.vector(pp$duration)
[1] FALSE
```

- `duration` is a difftime object
- difftime objects look like vectors but are not vectors

difftime objects



difftime objects

- to convert a Date object to a numeric object (if for example, you find the unit labels distracting) use `as.numeric()`:

```
> head(pp$duration)
Time differences in days
[1] 2865 1460 2922 2922 2922 1461
> pp$duration<-as.numeric(pp$duration)
> head(pp$duration)
[1] 2865 1460 2922 2922 2922 1461
```

Date objects

```
> head(pp$dob)
[1] February 22, 1732 October 30, 1735  April 13, 1743
[4] March 16, 1751    April 28, 1758    July 11, 1767
43 Levels: April 13, 1743 April 23, 1791 ... September 15, 1857
> pp$dob<-as.Date(pp$dob,format='%B %d, %Y')
```

- %B for month, as non-abbreviated names (if the month names are abbreviated, use %b)
- the spaces and comma replicate the format seen in head()

Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using reshape()

Date objects

- it's a good idea to check your work:

```
> head(pp$dob)
[1] "1732-02-22" "1735-10-30" "1743-04-13" "1751-03-16"
[5] "1758-04-28" "1767-07-11"
> class(pp$dob)
[1] "Date"
```


Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using reshape()

idu5.csv

```
> idu5<-read.csv('idu5.csv')
```

- a stripped-down data set of individuals from San Francisco's 5th implementation of CDC's National HIV Behavioral Surveillance for injection drug users (IDU)

Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using reshape()

Date objects

```
> head(idu5$idate)
[1] 21394 21394 21394 21394 21394 21394
> class(idu5$idate)
[1] "integer"
> idu5$idate<-as.Date(idu5$idate,origin='1960-01-01')
```

- to convert SAS-like dates, use the `origin` argument
- specify the `origin` using a `year-month-day` format

Date objects

- checking work:

```
> head(idu5$idate)
[1] "2018-07-29" "2018-07-29" "2018-07-29" "2018-07-29"
[5] "2018-07-29" "2018-07-29"
> class(idu5$idate)
[1] "Date"
> summary(idu5$idate)
        Min.      1st Qu.       Median         Mean
"2018-07-29" "2018-09-02" "2018-10-07" "2018-10-07"
     3rd Qu.         Max.
"2018-11-08" "2018-12-20"
```


Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using reshape()

Visualization of the data

```
> ggplot(aes(x=idate,y=age),data=idu5)+geom_point()+
+   geom_smooth(method='loess')+
+   xlab('Date')+ylab('Age')
```


Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

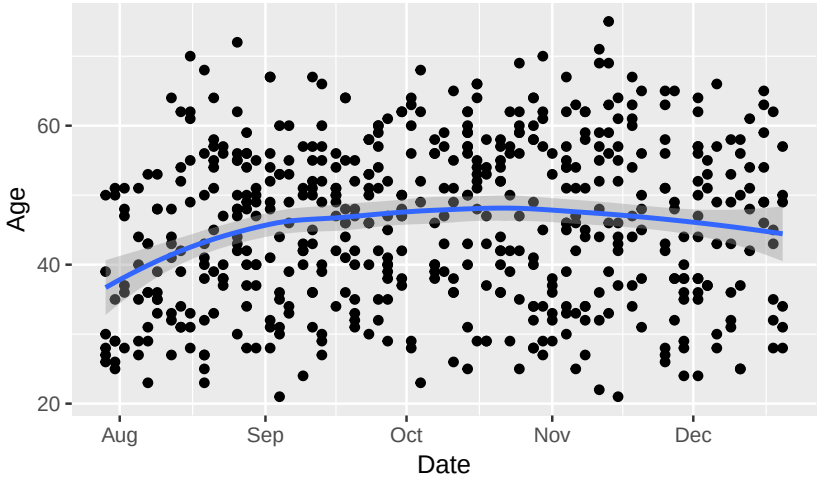
Reshaping data

Data structures

Wide to long "manually"

Wide to long using reshape()

Visualization of the data



Date objects

- Date objects (capital D) do not store time
- other date objects in R, however, do
- to convert to date objects that store time as well, try `strptime()`

Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using reshape()

rbind() and cbind()

Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using reshape()

rbind()

```
> ii.123<-ii[1:3,c('bmxht','bmxwt','bmxbmi')]
> ii.45<-ii[4:5,c('bmxht','bmxwt','bmxbmi')]
> rbind(ii.123,ii.45)
  bmxht bmxwt bmxbmi
1 171.3  78.3   26.7
2 176.8  89.5   28.6
3 175.3  88.9   28.9
4 162.4  52.0   19.7
5 158.7 105.0   41.7
```

- `rbind()` stands for row bind and, as the name implies, combines ("binds") data vertically

rbind()

```
> very.high<-ii[!is.na(ii$ldlcat)&(ii$ldlcat=='Very high'),
+               c('bmxht','bmxwt','bmxbmi')]
> high<-ii[!is.na(ii$ldlcat)&(ii$ldlcat=='High'),
+               c('bmxht','bmxwt','bmxbmi')]
> dim(very.high)
[1] 38  3
> dim(high)
[1] 182   3
> high.or.very.high<-rbind(high,very.high)
> dim(high.or.very.high)
[1] 220   3
```


Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using reshape()

rbind()

- it is possible to `rbind()` more than 2 objects:

```
> ii.1<-ii[1,]
> ii.2<-ii[2,]
> ii.3<-ii[3,]
> ii.123<-rbind(ii.1,ii.2,ii.3)
> nrow(ii.123)
[1] 3
```


Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using reshape()

rbind()

- the number of columns must match
- the column names must also match, if column names exist

rbind()

- however, the columns do not need to be in the same order:

```
> ii.1<-ii[1,c('bmxwt','bmxht')]
> ii.2<-ii[2,c('bmxht','bmxwt')]
> rbind(ii.1,ii.2)
  bmxwt bmxht
1  78.3 171.3
2  89.5 176.8
```


Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using reshape()

cbind()

```
> ii.bmx<-ii[1:3,c('bmxht','bmxwt')]
> ii.dmd<-ii[1:3,c('dmdborn4','dmdcitzn')]
> cbind(ii.bmx,ii.dmd)
  bmxht bmxwt dmdborn4 dmdcitzn
1 171.3  78.3        1        1
2 176.8  89.5        1        1
3 175.3  88.9        1        1
```

- `cbind()` stands for column bind and, as the name implies, combines ("binds") data horizontally

cbind()

- the number of rows must match
- the row names do not need to match, if row names exist

Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using reshape()

cbind()

- it is possible to misalign rows:

```
> ii.bmx<-ii[1:3,c('bmxht','bmxwt')]
> ii.dmd<-ii[101:103,c('dmdborn4','dmdcitzn')]
> cbind(ii.bmx,ii.dmd)
  bmxht bmxwt dmdborn4 dmdcitzn
1 171.3  78.3        1        1
2 176.8  89.5        1        1
3 175.3  88.9        1        1
```


Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using reshape()

Importing other data files

Brief Exercise

Date objects

`rbind()` and `cbind()`

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using `reshape()`

kevin_love.txt

- kevin_love.txt contains data on Kevin Love's NBA career
- it is a tab-delimited text file

Module 5: advanced data manipulation
Yea-Hung Chen, PhD, MS
Brief Exercise
Date objects
rbind() and cbind()
Importing other data files
Merging data
Reshaping data
Data structures
Wide to long "manually"
Wide to long using reshape()

Importing tab-delimited text files

- to import tab-delimited text files, use `read.delim()`:

```
> ll<-read.delim('kevin_love.txt')
> ll[1:3,1:10]
   season age  tm  lg pos  g gs   mp  fg  fga
1 2008-09  20 MIN NBA   C 81 37 25.3 3.9  8.5
2 2009-10  21 MIN NBA  PF 60 22 28.6 4.9 10.8
3 2010-11  22 MIN NBA  PF 73 73 35.8 6.6 14.1
```


Brief Exercise
Date objects
rbind() and cbind()
Importing other data files
Merging data
Reshaping data
Data structures
Wide to long "manually"
Wide to long using reshape()

Importing text files

- `read.delim()` and `read.csv()` are special cases of `read.table()`
- some useful arguments:
  - `header`: `TRUE` if the first row of the data contains the variable names
  - `sep`: the value used to separate cells (eg, `'\t'` for tab-delimited data)
  - `na.strings`: the character used to indicate missing values (eg, `''` for empty cells)

Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using reshape()

Importing other data files

- useful packages for reading other data files:
  - foreign for reading Stata, SPSS, or SAS XPORT files
  - readstata13 for reading Stata 13+ files
  - sas7bdat for reading SAS sas7bdat files
  - readxl for reading Excel files

DR1TOT_H.XPT

- DR1TOT_H.XPT contains data from the dietary interview component of NHANES, also known as What We Eat in America (WWEIA)
- documentation is available here

Importing SAS XPORT files

- just once ever:

```
> install.packages('foreign')
```

- just once per session:

```
> library(foreign)
```


Brief Exercise
Date objects
rbind() and cbind()
Importing other data files
Merging data
Reshaping data
Data structures
Wide to long "manually"
Wide to long using reshape()

Importing SAS XPORT files

```
> dietary<-read.xport('DR1TOT_H.XPT')
```


Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using reshape()

Importing SAS XPORT files

- checking work:

```
> dietary[1:5,1:7]
   SEQN    WTDRD1    WTDR2D DR1DRSTZ DR1EXMER DRABF DRDINT
1 73557  16888.33  12930.89        1       49     2      2
2 73558  17932.14  12684.15        1       59     2      2
3 73559  59641.81  39394.24        1       49     2      2
4 73560 142203.07 125966.37        1       54     2      2
5 73561  59052.36  39004.89        1       63     2      2
```


Brief Exercise
Date objects
rbind() and cbind()
Importing other data files
Merging data
Reshaping data
Data structures
Wide to long "manually"
Wide to long using reshape()

Importing SAS XPORT files

- I like making variable names lower case:

```
> names(dietary)<-tolower(names(dietary))
```


Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using reshape()

Merging data

Brief Exercise

Date objects

`rbind()` and `cbind()`

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using `reshape()`

Merging data

- prior to merging, it's useful to check for duplicated IDs:

```
> table(duplicated(ii$seqn))

FALSE
 5769
> table(duplicated(dietary$seqn))

FALSE
 9813
```


Brief Exercise
Date objects
rbind() and cbind()
Importing other data files
Merging data
Reshaping data
Data structures
Wide to long "manually"
Wide to long using reshape()

Merging data

- to merge data frames, use `merge()`:

```
> ii.new<-merge(ii,dietary,by='seqn',all.x=TRUE,all.y=FALSE)
```

- `by` specifies the identifying variable (usually some ID number)
- if the variable names differ, use `by.x` and `by.y` for the first the first and second data, respectively
- `all.x` and `all.y` indicate whether to keep all rows in the first and second data, respectively

Brief Exercise
Date objects
rbind() and cbind()
Importing other data files
Merging data
Reshaping data
Data structures
Wide to long "manually"
Wide to long using reshape()

Merging data

- thinking about sets or a Venn diagram:
  - `all=TRUE` means *X OR Y*
  - `all=FALSE` means *X AND Y*
  - `all.x=TRUE, all.y=FALSE` means *X*
  - `all.x=FALSE, all.y=TRUE` means *Y*

Brief Exercise

Date objects

`rbind()` and `cbind()`

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using `reshape()`

Merging data

- checking work:

```
> nrow(ii)
[1] 5769
> nrow(dietary)
[1] 9813
> nrow(ii.new)
[1] 5769
```


Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using reshape()

Merging data

- checking work:

```
> tail(names(ii))
[1] "vra"     "aspirin" "sleep"   "is"      "hs"
[6] "ldlcat"
> tail(names(ii.new))
[1] "drd370sq" "drd370t"  "drd370tq" "drd370u"  "drd370uq"
[6] "drd370v"
```


Brief Exercise
Date objects
rbind() and cbind()
Importing other data files
Merging data
Reshaping data
Data structures
Wide to long "manually"
Wide to long using reshape()

Merging data

- replacing the original data:

```
> ii<-ii.new
```

- I could have done this at the very beginning, avoiding `ii.new` altogether, but it can be helpful to wait until after checking the new data

Brief Exercise
Date objects
rbind() and cbind()
Importing other data files
Merging data
Reshaping data
Data structures
Wide to long "manually"
Wide to long using reshape()

Reshaping data

Brief Exercise

Date objects

`rbind()` and `cbind()`

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using `reshape()`

Data structures

Brief Exercise
Date objects
rbind() and cbind()
Importing other data files
Merging data
Reshaping data
Data structures
Wide to long "manually"
Wide to long using reshape()

Wide data

- wide data refer to data structures in which each row is an individual (or unit of analysis)
- if there are repeated measures, the measurements appear as multiple columns
- for example, in the NHANES data, there are 4 variables for diastolic blood pressure: `bpxdi1`, `bpxdi2`, `bpxdi3`, and `bpxdi4`

Wide data

- diastolic blood pressure in NHANES:

```
> ii[1:4,c('seqn','bpxdi1','bpxdi2','bpxdi3','bpxdi4')]
   seqn bpxdi1 bpxdi2 bpxdi3 bpxdi4
1 73557     72     76     74     NA
2 73558     62     80     42     NA
3 73559     90     76     80     NA
4 73561     86     88     86     NA
```


Brief Exercise
Date objects
rbind() and cbind()
Importing other data files
Merging data
Reshaping data
Data structures
Wide to long "manually"
Wide to long using reshape()

Long data

- long data refer to data structures in which each individual (or unit of analysis) may appear in more than one row
- I prefer thinking of "long data" as "tall data" since long doesn't, in my view, unambiguously refer to height

Brief Exercise
Date objects
`rbind()` and `cbind()`
Importing other data files
Merging data
Reshaping data
Data structures
Wide to long "manually"
Wide to long using `reshape()`

Long data

- the UNPS data:

```
> uu[uu$pid=='10830027090305',c('pid','wave')]
                  pid wave
1753 10830027090305 2009
3355 10830027090305 2010
4778 10830027090305 2011
```


Brief Exercise
Date objects
rbind() and cbind()
Importing other data files
Merging data
Reshaping data
Data structures
Wide to long "manually"
Wide to long using reshape()

Long data

- example of long data, from a hypothetical longitudinal study of 2 individuals, with 3 visits per individual:

	individual	visit	systolic	daistolic
1	1	1	92	64
2	1	2	102	62
3	1	3	108	66
4	2	1	144	62
5	2	2	132	72
6	2	3	116	74

- each row represents a person-visit

Long data

- example of long data, from a hypothetical cross-sectional study of 2 individuals, with information collected on up to 5 sexual partners per individual:

	individual	partner	partner.age	condomless
1	1	1	25	1
2	1	2	32	0
3	1	3	27	0
4	1	4	29	1
5	2	1	42	0
6	2	2	46	0

- each row represents a sexual partnership

Wide or long?

- what we consider wide or long can depend on what we consider the unit of analysis to be (and this can change within project!)
- for example, with the Kevin Love data:
  - if we say that the unit of analysis is an individual's NBA season, the data are in wide format, because each row represents an individual's NBA season
  - if we say that the unit of analysis is the individual, the data are in long format, because each individual can appear in multiple rows

Wide to long “manually”

Brief Exercise

Date objects

rbind() and cbind()

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long “manually”

Wide to long using reshape()

Reshaping wide to long "manually"

- suppose we want to create long data, for the first 2 measurements of diastolic blood pressure
- we could do this "manually" as follows:

```
> dia1<-data.frame(seqn=ii$seqn,order=1,diastolic=ii$bpxdi1)
> dia2<-data.frame(seqn=ii$seqn,order=2,diastolic=ii$bpxdi2)
> mm<-rbind(dia1,dia2)
```


Brief Exercise
Date objects
rbind() and cbind()
Importing other data files
Merging data
Reshaping data
Data structures
Wide to long "manually"
Wide to long using reshape()

Reshaping wide to long "manually"

- checking work:

```
> head(mm[order(mm$seqn),])
      seqn order diastolic
1    73557     1        72
5770 73557     2        76
2    73558     1        62
5771 73558     2        80
3    73559     1        90
5772 73559     2        76
```

Reshaping wide to long "manually"

- checking work:

```
> nrow(ii)*2
[1] 11538
> nrow(mm)
[1] 11538
> length(unique(mm$seqn))
[1] 5769
> nrow(ii)
[1] 5769
```


Brief Exercise
Date objects
rbind() and cbind()
Importing other data files
Merging data
Reshaping data
Data structures
Wide to long "manually"
Wide to long using reshape()

Reshaping wide to long "manually"

- we have 4 measurements of diastolic blood pressure though
- we could create a long data frame of all 4 measurements manually, but this would be slightly tedious
- with other data sets, and more complex requirements, this would be even more tedious
- even with the NHANES data, for example, what if we also wanted to include systolic blood pressure?

Wide to long using `reshape()`

Brief Exercise

Date objects

`rbind()` and `cbind()`

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using `reshape()`

Reshaping wide to long using reshape()

```
> kk<-c('seqn','ridageyr','bpxdi1','bpxdi2','bpxdi3','bpxdi4')
> mm<-reshape(data=ii[,kk],
+             varying=kk[3:6],
+             sep='',direction='long',
+             idvar='seqn',timevar='order')
```

- data: the data to reshape (could use ii, but using a subset here for simplicity of presentation)
- varying: the variable names of the repeated measurements
- sep: if varying use a naming convention of some stub name plus some number, sep is the character separating the stub and the number
- direction: the direction of reshaping

Reshaping wide to long using reshape()

```
> kk<-c('seqn','ridageyr','bpxdi1','bpxdi2','bpxdi3','bpxdi4')
> mm<-reshape(data=ii[,kk],
+             varying=kk[3:6],
+             sep='',direction='long',
+             idvar='seqn',timevar='order')
```

- `idvar` (*optional*): an ID variable, possibly using a variable name in the original data (if you don't include this, R will create a new variable called `id`)
- `timevar` (*optional*): the name for the new variable that will distinguish rows (the default name is `time`)

Brief Exercise
Date objects
rbind() and cbind()
Importing other data files
Merging data
Reshaping data
Data structures
Wide to long "manually"
Wide to long using reshape()

Reshaping wide to long using `reshape()`

- checking work:

```
> head(mm[order(mm$seqn),])
         seqn ridageyr order bpxdi
73557.1 73557       69     1    72
73557.2 73557       69     2    76
73557.3 73557       69     3    74
73557.4 73557       69     4    NA
73558.1 73558       54     1    62
73558.2 73558       54     2    80
```

- notice that `reshape()` automatically adds row names

Reshaping wide to long using `reshape()`

- checking work:

```
> nrow(ii)*4
[1] 23076
> nrow(mm)
[1] 23076
> length(unique(mm$seqn))
[1] 5769
> length(unique(ii$seqn))
[1] 5769
```


Brief Exercise
Date objects
rbind() and cbind()
Importing other data files
Merging data
Reshaping data
Data structures
Wide to long "manually"
Wide to long using reshape()

Reshaping wide to long using `reshape()`

- what if we had omitted `idvar` and `timevar`?

```
> kk<-c('seqn','ridageyr','bpxdi1','bpxdi2','bpxdi3','bpxdi4')
> mm<-reshape(data=ii[,kk],
+             varying=kk[3:6],
+             sep='',direction='long',
+             idvar='seqn',timevar='order')
> mm.simpler<-reshape(data=ii[,kk],
+                     varying=kk[3:6],
+                     sep='',direction='long')
```

Reshaping wide to long using `reshape()`

```
> mm[mm$seqn==73557,]
         seqn ridageyr order bpxdi
73557.1 73557       69     1    72
73557.2 73557       69     2    76
73557.3 73557       69     3    74
73557.4 73557       69     4    NA
> mm.simpler[mm.simpler$seqn==73557,]
     seqn ridageyr time bpxdi id
1.1 73557       69    1    72  1
1.2 73557       69    2    76  1
1.3 73557       69    3    74  1
1.4 73557       69    4    NA  1
```


Brief Exercise
Date objects
rbind() and cbind()
Importing other data files
Merging data
Reshaping data
Data structures
Wide to long "manually"
Wide to long using reshape()

Reshaping wide to long using reshape()

- including systolic blood pressure as well:

```
> kk<-c(kk,'bpxsy1','bpxsy2','bpxsy3','bpxsy4')
> mm<-reshape(data=ii[,kk],
+             varying=kk[3:10],
+             sep='',direction='long',
+             idvar='seqn',timevar='order')
```

Reshaping wide to long using `reshape()`

- checking work:

```
> head(mm[order(mm$seqn),])
         seqn ridageyr order bpxdi bpxsy
73557.1 73557       69     1    72   122
73557.2 73557       69     2    76   114
73557.3 73557       69     3    74   102
73557.4 73557       69     4    NA    NA
73558.1 73558       54     1    62   156
73558.2 73558       54     2    80   160
```


Brief Exercise
Date objects
rbind() and cbind()
Importing other data files
Merging data
Reshaping data
Data structures
Wide to long "manually"
Wide to long using reshape()

Reshaping wide to long using `reshape()`

- checking work:

```
> nrow(ii)*4
[1] 23076
> nrow(mm)
[1] 23076
> length(unique(mm$seqn))
[1] 5769
> length(unique(ii$seqn))
[1] 5769
```


Brief Exercise
Date objects
rbind() and cbind()
Importing other data files
Merging data
Reshaping data
Data structures
Wide to long "manually"
Wide to long using reshape()

Reshaping wide to long using `reshape()`

- reshaping the Kevin Love data:

```
> mm<-reshape(ll[c('season','age','fgp','ftp')],
+             varying=c('fgp','ftp'),
+             v.names='value',timevar='stat',
+             direction='long',
+             idvar='season',
+             times=c('Field goals','Free throws'))
```

- we don't have repeating variable names, so instead of `sep` we use:
  - `v.names` to specify the name for the new "value" variable
  - `timevar` to specify the name for the new "label" variable

Reshaping wide to long using `reshape()`

```
> mm<-reshape(ll[c('season','age','fgp','ftp')],
+             varying=c('fgp','ftp'),
+             v.names='value',timevar='stat',
+             direction='long',
+             idvar='season',
+             times=c('Field goals','Free throws'))
```

- `times` (*optional*): specifies the values for the "label" variable (the default would be sequential numbering)

Brief Exercise
Date objects
rbind() and cbind()
Importing other data files
Merging data
Reshaping data
Data structures
Wide to long "manually"
Wide to long using reshape()

Reshaping wide to long using `reshape()`

- checking work:

```
> head(mm[order(mm$season),])
                     season age        stat value
2008-09.Field goals 2008-09  20 Field goals 0.459
2008-09.Free throws 2008-09  20 Free throws 0.789
2009-10.Field goals 2009-10  21 Field goals 0.450
2009-10.Free throws 2009-10  21 Free throws 0.815
2010-11.Field goals 2010-11  22 Field goals 0.470
2010-11.Free throws 2010-11  22 Free throws 0.850
```

Reshaping wide to long using `reshape()`

- checking work:

```
> nrow(ll)*2
[1] 22
> nrow(mm)
[1] 22
> length(unique(mm$season))
[1] 11
> length(unique(ll$season))
[1] 11
```


Brief Exercise
Date objects
rbind() and cbind()
Importing other data files
Merging data
Reshaping data
Data structures
Wide to long "manually"
Wide to long using reshape()

Reshaping wide to long using `reshape()`

- what if we had omitted `times`?

```
> mm.simpler<-reshape(ll[c('season','age','fgp','ftp')],
+             varying=c('fgp','ftp'),
+             v.names='value',timevar='stat',
+             direction='long',
+             idvar='season')
```


Brief Exercise

Date objects

`rbind()` and `cbind()`

Importing other data files

Merging data

Reshaping data

Data structures

Wide to long "manually"

Wide to long using `reshape()`

Reshaping wide to long using `reshape()`

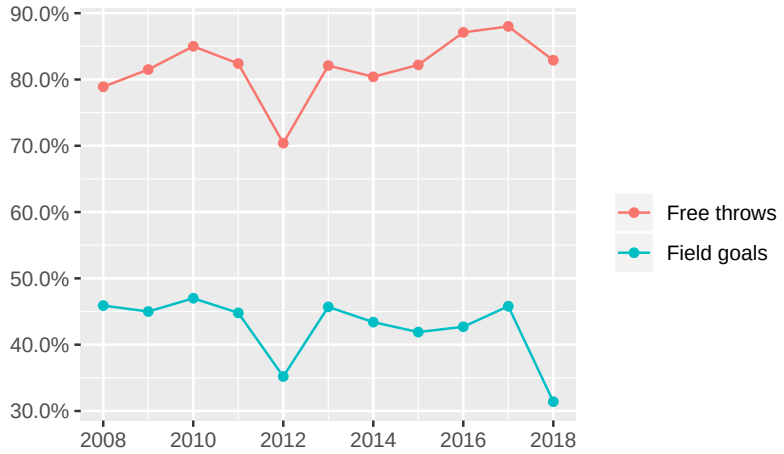
```
> head(mm.simpler[order(mm.simpler$season),])
           season age stat value
2008-09.1 2008-09  20    1 0.459
2008-09.2 2008-09  20    2 0.789
2009-10.1 2009-10  21    1 0.450
2009-10.2 2009-10  21    2 0.815
2010-11.1 2010-11  22    1 0.470
2010-11.2 2010-11  22    2 0.850
```

Visualization of the long data

```
> mm$season<-as.numeric(substr(mm$season,0,4))
> mm$stat<-factor(mm$stat,c('Free throws','Field goals'))
> ggplot(aes(x=season,y=value,col=stat),data=mm)+
+   geom_point()+geom_line()+
+   xlab('')+ylab('')+labs(col='Statistic')+
+   scale_y_continuous(labels=scales::percent)+
+   scale_x_continuous(breaks=seq(2008,2018,2))
```


Brief Exercise
Date objects
rbind() and cbind()
Importing other data files
Merging data
Reshaping data
Data structures
Wide to long "manually"
Wide to long using reshape()

Visualization of the long data



Brief Exercise
Date objects
rbind() and cbind()
Importing other data files
Merging data
Reshaping data
Data structures
Wide to long "manually"
Wide to long using reshape()