PS6 Lecture 6

Daniel Lim

University of California, Los Angeles

4/16/2014

Agenda

- Intro to Bivariate Analysis
- Contingency Tables
- Scatterplots

Bivariate analysis

So far, we've only looked at single variables/columns of data.

However, most things we do in social science are about explaining relationships

- relationships require multiple variables
- let's start with most basic type – bivariate relationships (i.e. between 2 variables)

Visualizing bivariate data

Upon loading multivariate data, you should...

- visualize univariate dists. using techniques covered so far.
- visualize bivariate relationships

Visualizing bivariate data

In fact, we already saw one way to view bivariate relationships, using side-by-side boxplots (lecture 2). Here we'll cover 2 more:

- contingency tables (AKA cross tabs)
- scatterplots

Contingency tables

Contingency tables let us look at the breakdown of data by bivariate relationship.

Let's load a small dataset to demonstrate.

Composition of one PS6 section by sex, field and year in school

```
> f.roster = 'c:/users/daniel/dropbox/ps6/data/roster.csv'
> roster = read.csv(f.roster, stringsAsFactors = F)
```

Contingency tables

Breakdowns by sex and field

```
> table(roster$female, roster$field)

    hsci hum oth ssci
  0    2   4   3   10
  1    1  12   3   15
```

Contingency tables

. . . by sex and class year.

```
> table(roster$female, roster$year)

     0  1  2  3  4
  0  3  0  7  7  2
  1  3  1  8  8 11
```

Contingency tables

...by field and class year

```
> table(roster$field, roster$year)

        0  1  2  3  4
  hsci  0  1  0  2  0
  hum   0  0  4  5  7
  oth   6  0  0  0  0
  ssci  0  0 11  8  6
```

- Such tables allow us to see how one var. varies with another
- Very useful for simple data – quick and succinct summary
- While not included here, contingency tables often have an addt'l row and column summing the results by row and column. These are called the **margins**.

Margins

You can add margins to a table as follows:

```
> tab1 = table(roster$field, roster$year)
> mar.row = margin.table(tab1, 1)
> tab1 = cbind(tab1, mar.row)
> mar.col = margin.table(tab1, 2)
> tab1 = rbind(tab1, mar.col)
> tab1
```

```
        0 1  2  3  4 mar.row
hsci    0 1  0  2  0       3
hum     0 0  4  5  7      16
oth     6 0  0  0  0       6
ssci    0 0 11  8  6      25
mar.col 6 1 15 15 13      50
```

Notes regarding code on previous slide

- ‘margin.table’ computes margins for the table specified in *arg1*, for the dimension specified in *arg2*
  - 1 is by row, 2 is by column
- ‘cbind’ adds a column to the end of a table
- ‘rbind’ adds a row to the end of a table
  - can add as many rows/cols as you want, but order matters
  - ‘cbind’ing N vectors of length M creates an $M \times N$ table
  - ‘rbind’ing N vectors of length M creates an $N \times M$ table

Limitations of contingency tables

One limitation of contingency tables: they don't work well with continuous data, or discrete data with too many unique values.

Try the following to see what I mean.

```
> table(gdp$Year, gdp$GDPcapImp)
```

To use contingency tables with such data, we need to segment the data, limiting the number of unique values that we are making a table out of.

We can do this in R using the ‘cut’ function

arg 1: the data that you want to segment

arg 2: can be one of 2 things:

- single number specifying how many bins
- vector of points at which to cut the data.

Here's a simple example to see how 'cut' works

```
> fake = 1:10
> fake1 = cut(fake, 2)
> fake2 = cut(fake, c(-100, 2, 7, 100))
> table(fake1)

fake1
(0.991,5.5]    (5.5,10]
          5           5

> table(fake2)

fake2
(-100,2]    (2,7]  (7,100]
       2        5        3
```

Here we use 'cut' to preprocess the GDP data before tabling.

```
> year2 = cut(gdp$Year, seq(1960, 2000, 10))
> gdp2 = cut(gdp$GDPcapImp, c(0, 1000, 5000, 10000, 20000, 1e6))
> tab2 = table(year2, gdp2)
```

```
> tab2

                          gdp2
year2                       (0,1e+03] (1e+03,5e+03] (5e+03,1e+04]
  (1.96e+03,1.97e+03]       352           171             0
  (1.97e+03,1.98e+03]       515           505           151
  (1.98e+03,1.99e+03]       322           553           205
  (1.99e+03,2e+03]          134           472           156
                          gdp2
year2                       (1e+04,2e+04] (2e+04,1e+06]
  (1.96e+03,1.97e+03]                   0             0
  (1.97e+03,1.98e+03]                  17             5
  (1.98e+03,1.99e+03]                 207            13
  (1.99e+03,2e+03]                    146            59
```

We see that...

- the distribution of GDP/cap expands upward over time
  - beyond this basic point, hard to see other patterns using table
- given the continuous nature of gdp/cap...
  - we've lost a lot of information by segmenting
  - visualizing using a side-by-side boxplot or scatterplot would be a better choice

It's helpful to see how these data look using a scatterplot.

Scatterplots

Plot with a single mark (typically a dot) for each observation

- x coordinate is each observation's value on one variable
- y coordinate is value on the other variable

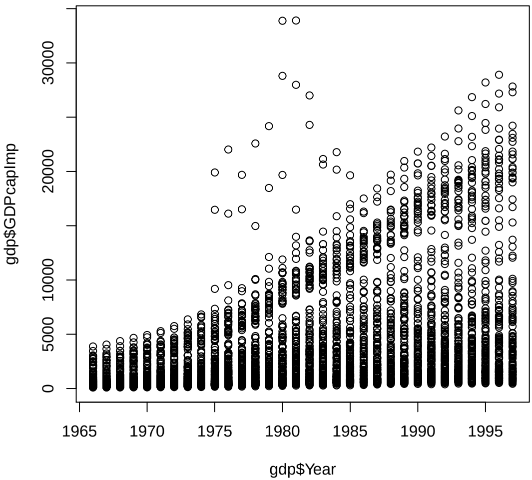
Scatterplots

In R, use the 'plot' command to create scatterplots.

- We previously used 'plot' with the "type='l' " argument to create densities.
- For scatterplots, just omit 'type'.

For example:

```
> plot(gdp$Year, gdp$GDPcapImp)
```

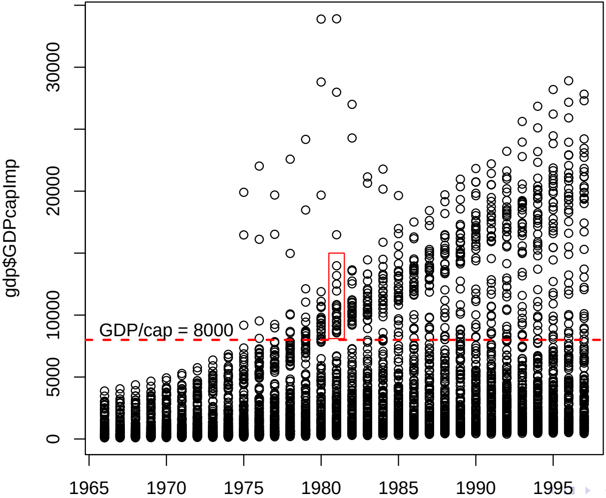

gdp$GDPcapImp
0
5000
10000
20000
30000
1965
1970
1975
1980
1985
1990
1995
gdp$Year

What are some things we notice from this plot?

- growing gap between GDP/cap in rich and poor countries – the rich get richer, the poor stay relatively poor
- hard to say definitively but it looks like there's a lot more poor countries than rich
- there are a few extremely high outliers between 1975 and 1985

The gap between rich and poor

- While it seems fairly obvious, let's first make sure it's the same countries that continue to get richer from year to year. We'll do this by coloring the countries in the upper band a separate color.
- I'm going to use 1981 as a reference year. It looks like the upper band of countries are those with GDP/cap $>$ \$8000 in that year.
- I just eyeballed this as in the next slide.


gdp$GDPcapImp
0
5000
10000
20000
30000
GDP/cap = 8000
1965
1970
1975
1980
1985
1990
1995

To color these countries separately:

1. subset countries that had GDP/cap > \$8000 in 1981
2. add an indicator ('high') to the full dataset that equals 1 if in the list from step 1, and 0 otherwise
3. plot only those countries with 'high==0', then add those with 'high==1' separately in a different color

Here's the code for steps 1 and 2.

```
> h81 = subset(gdp, gdp$Year == 1981 & GDPcapImp > 8000)
> highCountries = h81$Country
> gdp$high = ifelse(gdp$Country %in% highCountries, 1, 0)
```

Notes regarding code on previous slide

line 1: since we subset on ‘gdp’, which is the entire dataset, we will get all variables for every observation that satisfies the 2 conditions in *arg2* of the subset command.

- you can verify this by running ‘str’ on gdp, then on h81

line 2: we are extracting the list of countries in the subset to the variable ‘highCountries’

- enter ‘highCountries’ to see what’s in that vector line 3: we create a new variable (i.e. column) in ‘gdp’ called high using the return from calling ‘ifelse’

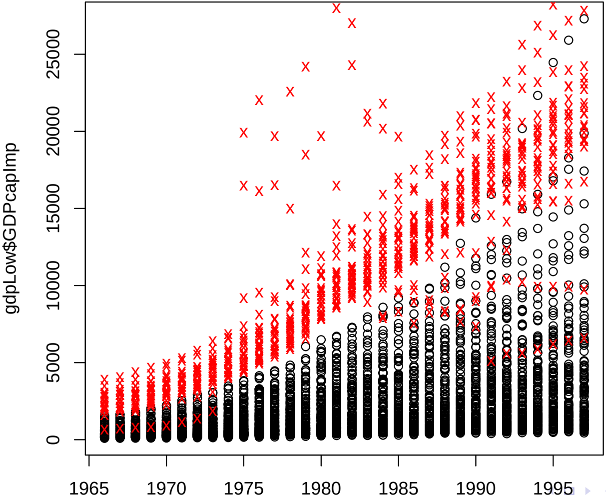
- ‘high’ does not exist in ‘gdp’. For dataframes, if you refer to a column that doesn’t exist, it is created
- ‘ifelse’ takes 3 args: (1) a condition, (2) the value returned when the condition is true, (3) value returned when false

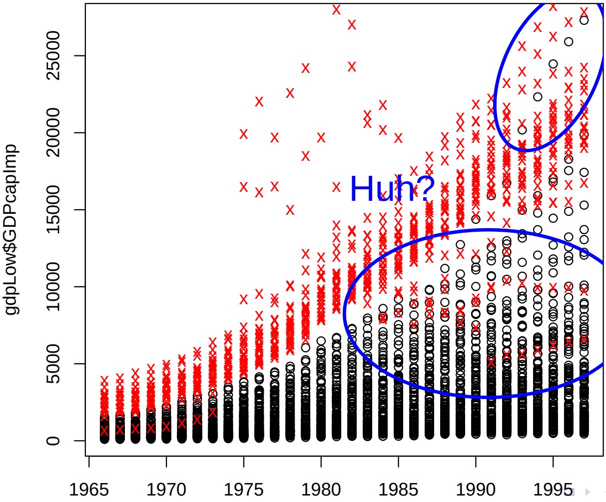
- ‘gdp$Country’ is the vector of country names for the entire ‘gdp’ dataset. This condition checks each entry to see if it is in the vector ‘highCountries’, returning true when it is and false otherwise.

- resultantly, for each observation that is a country in ‘highCountries’, we get 1, and 0 otherwise.

Let's now plot the points. Observations with 'high==1' will be colored red, and marked with an 'x' rather than the default circle.

```
> gdpLow = subset(gdp, gdp$high == 0)
> gdpHigh = subset(gdp, gdp$high == 1)
> plot(gdpLow$Year, gdpLow$GDPcapImp)
> points(gdpHigh$Year, gdpHigh$GDPcapImp, pch='x', col='red')
```


gdpLow$GDPcapImp
0
5000
10000
15000
20000
25000
1965
1970
1975
1980
1985
1990
1995


gdpLow$GDPcapImp
0
5000
10000
15000
20000
25000
1965
1970
1975
1980
1985
1990
1995
Huh?

Some unexpected results.

- there are some countries that were in the high cluster as of 1981, but then started stagnating. (red Xs separate from main pack post-1981).
- there's also countries that were in the low clusters prior to 1981, but then ended up in the high clusters in later years. (black circles in the high clusters).

Set this aside for the moment – we'll resume investigating with line graphs next lecture.

Limitations of scatterplots

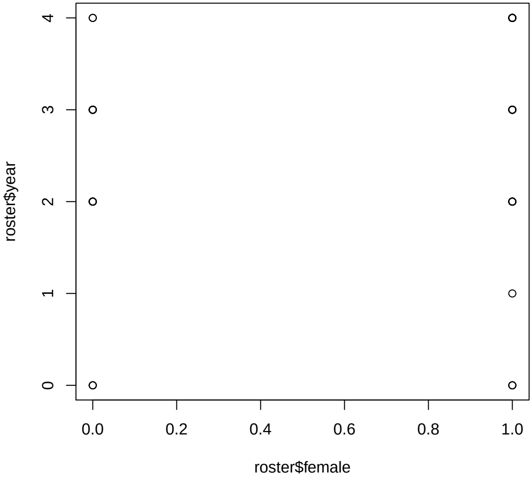
Clearly, scatterplots are nice, but they also have limitations.

They don't deal well with

- discrete/categorical data.
- non-cardinal data (i.e. ordinal data, text)

First, try

```
> plot(roster$female, roster$year)
```



All the values are overlapping – not very helpful is it?

You could change the size of the circle at each nexus based on number of circles there, but then it's not longer strictly a scatterplot.

Next, try

```
> plot(roster$female, roster$field)
```

- This gives an error...
- ...because ‘roster$field’ is full of text values...how would you arrange a *meaningful* axis for 'other', 'humanities', 'social sciences', and 'hard sciences'?

One could substitute values for each category, but this creates other problems:

- *post grad. avg. salary*: nice, except this is no longer the same data
- *arbitrary nums*: the axis no longer holds meaning. Might as well side-by-side barplot

Bottom line: no single plot type is perfect for every situation. You need to know many different types because each has strengths and weaknesses.