

# Demography 213

## Calculating rates using `tapply()` and IPUMS

Carl Mason  
carlm@demog.berkeley.edu

October 5, 2011

### Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Data</b>	<b>2</b>
<b>3</b>	<b>Assumptions and considerations</b>	<b>2</b>
<b>4</b>	<b>Example: calculating ASFR and TFR</b>	<b>4</b>
<b>5</b>	<b>Assignment</b>	<b>9</b>
5.1	Getting data from IPUMS to R . . . . .	9
5.1.1	Download your data to /data/commons/userid . . . . .	9
5.1.2	Uncompress your data file onto /72hours . . . . .	10
5.1.3	Run STATA do produce a .sda file that R can read . . . . .	10
5.1.4	Run stata to produce the .dta file that R can read . . . . .	11
5.1.5	Reading the data into R . . . . .	11

#### Abstract

This week we will use the American Community Survey (ACS) to calculate age specific rates. The example herein is to calculate ASFR and TFR for US resident women broken out by some interesting characteristic. Your job will be to find some other sort of rate to calculate – and to make a cool graph of the result. While this problem *could* be solved using loops—it is advantageous to use `tapply()`.

## 1 Introduction

The *Total Fertility Rate* is a synthetic measure that sort of more or less estimates the number of children a woman would have if she lived through her entire reproductive life and behaved—fertility wise in the same way as women of each age *at a particular point in time* behave. This number is often reported in the

media as the “number of children per woman”. It is a nothing more than the sum of the ASFRs over the reproductive years. In other words, the work is in calculating the ASFR.

Equation 1 shows the well known formulae for ASFR and TFR.

This week, we will calculate ASFRs and (trivially) TFRs for US resident, non-institutionalized women disaggregated by some characteristic or another. Although fertility is kind of interesting, what this class is really about is calculating rates – or more generally measures that require numerators and denominators. Proportions and probabilities are like this too. This comes up a lot in Demography. Demographers tend to see the world in terms of events (numerator) and risk sets (denominator). As Equation 1 makes “clear” an ASFR is the number of events (births to women of a certain age) divided by the risk set (the number of person years experienced by women of that age in that calendar year).

The `tapply()` function is particularly well suited to doing these sorts of calculations – and these sorts of calculations are what you will be doing this week.

$$\text{ASFR}_a = \frac{\sum \text{births to women of age } a}{\sum \text{woman years of exposure at age } a} \quad (1)$$

$$\text{TFR} = \sum_{a=15}^{45} \text{ASFR}_a = \sum_{a=15}^{45} E(\text{births}_a | \text{Survival through age } a) \quad (2)$$

$$a = \text{Single year of age} \quad (3)$$

## 2 Data

For this project we will use data from the Integrated Public Use Microseries (IPUMS). IPUMS is a fabulous resource for demographers so if you have not encountered it before, then you are in for treat.

If you have not already done so, visit [www.ipums.org/usa](http://www.ipums.org/usa) and sign up for access. Do this by clicking on **Login** and then taking advantage of the obvious opportunities that follow (Request and Account).

I have already created an extract of the 2009 American Community Survey (ACS) which is adequate for doing the example. You **may** want to create your own extract for the second part of the assignment.

## 3 Assumptions and considerations

The ACS provides age and sex as well as a large number of other characteristics of each person in the sample (which is 1 percent of the US). Crucially, they also ask whether or not each person has had a child in the last year (actually, the Census is clever enough to only ask this question of women). It would have been nicer if they had asked for the *date of the last birth* but that’s life. We can

estimate a set of ASFRs with what we have. (If we were ambitious we *might* be able to improve our estimate by using the year of birth variable – but let’s not)

Some details that we shall need to consider are the following:

- The data are not a simple random sample. **We must use weights.**
- It is feasible and therefore desirable to use single year age cohorts for women at risk of birth.
- We would like calculate ASFRs/TFRs for women disaggregated by some interesting characteristic. There are many to choose from.
- Mortality could have an effect—some women might have died before the Census Bureau could interview them. Their births and their person-years-at-risk are lost to us.
- Twin and multiple births pose problems (for parents as well as demographers). The possibility of multiple single births during the (one year) period of observation also creates problems for all concerned.
- We are calculating a *period* TFR rather than a *cohort* TFR, consequently, we would like to observe people over a particular calendar year. In a *perfect* world, that year of observation would also correspond exactly to the ages of those under observation. The ACS does not work that way:
  - The ACS surveys are collected throughout the year. The ACS sample is divided into 12 bits and surveys are administered to one bit every month over the year.
  - Relatively few women filled out the ACS survey on their birthday.

We deserve a simpler world than the one we forced to study. Fortunately, as social scientists, we can fight back against the forces of unreasonable complexity with at least one powerful weapon: The Assumption.

While we will explicitly deal with sample weights— we can assume our way out of many difficulties by:

- Defining “birth” as one *or more babies*.
- Ignoring the possibility of two birth events in a single year.
- Assume that women’s birthdays are randomly distributed throughout the year.
- Defining our ASFRs as applying in some way to 2008 and 2009.

Pause briefly to consider the implications. OK that’s long enough.

## 4 Example: calculating ASFR and TFR

For this example we'll read in a data extract that I created. In Section 5.1, it is revealed, how you will create your own IPUMS data extract and prepare it to be read into R. NOTE that this procedure is slightly different from the one you will pursue on your own. Refer to Section 5.1 when you are ready to import your own data extract.

```
> library(foreign)
> ## READ the data
>
> ## create a symlink to where the data are -- the data are in
> ## /data/commons, which is a good place to put data like this (easily
> ## reproducible IPUMS holds on to your extract definitions for at
> ## least a decade)
> system("ln -s /data/commons/carlm/ACS2009/ ACS2009")
> ## I compressed (gzipped) the data because that is what good
> ## responsible computer users do. To use the data we need to
> ## uncompress it. A GREAT place to uncompress the data is
> ## /72hours. If you use zcat as shown, the compressed copy will stay in
> ## /data/commons. The uncompressd copy will wind up in /72hours. It
> ## will be erased for you in a few days -- but you can get it back
> ## anytime you want.
> ##uncompress the data to /72hours
> system("zcat ACS2009/usa_00084.dta.gz > /72hours/acs09.dta")
> ## read the data into R.
> ## the "foreign" library has the code to read and process stata .dta files
> library(foreign)
> acs09<-read.dta(file="/72hours/acs09.dta")
> names(acs09)

 [1] "year"      "datanum"   "serial"    "hhwt"      "region"    "statefip"
 [7] "gq"        "pernum"    "perwt"     "famsize"   "nchild"    "nchlt5"
[13] "age"       "sex"       "marst"     "marrno"    "marrinyr"  "widinyr"
[19] "divinyr"   "fertyr"    "race"      "raced"     "bpl"       "bpld"
[25] "citizen"   "yrnatur"   "yrimmig"   "hispan"    "hispanid"  "hcovany"
[31] "hcovpriv"  "hinsemp"   "hinspur"   "occ"       "migrate1"  "migrate1d"
[37] "diffmob"

> ## sum the person weight to see how many people the observations represent
> sum(acs09$perwt) ## 75.9 million women between age 15 and 50

[1] 75974672

> ## IPUMS GOTCHA # 1 --
> is.numeric(acs09$age)

[1] FALSE
```

```

> is.factor(acs09$age)

[1] TRUE

> ##levels(acs09$age) ## look at all these stupid levels -- which in
> ##this case have no observations.
>
> ## create a numeric age variable
> acs09$ageN<-as.numeric(acs09$age) -1
> ## make sure it looks right
> table(acs09$ageN)

  15   16   17   18   19   20   21   22   23   24   25   26   27
20231 20318 20815 20648 19442 17924 17160 16530 17209 17110 17088 17239 17532
  28   29   30   31   32   33   34   35   36   37   38   39   40
17824 18005 18402 17777 17792 17265 17429 17828 17983 18853 20534 20579 20932
  41   42   43   44   45   46   47   48   49   50
19360 20286 20679 22050 23425 22969 23360 23886 23918 24931

> table(as.character(acs09$age))

  15   16   17   18   19   20   21   22   23   24   25   26   27
20231 20318 20815 20648 19442 17924 17160 16530 17209 17110 17088 17239 17532
  28   29   30   31   32   33   34   35   36   37   38   39   40
17824 18005 18402 17777 17792 17265 17429 17828 17983 18853 20534 20579 20932
  41   42   43   44   45   46   47   48   49   50
19360 20286 20679 22050 23425 22969 23360 23886 23918 24931

> ## IPUMS GOTCHA #2 --
>
> ## fertyr is the answer to the question: did respondent have birth in
> ## the last year? One would think that this was a yes/no question... but
>
> is.logical(acs09$fertyr)

[1] FALSE

> is.factor(acs09$fertyr)

[1] TRUE

> table(acs09$fertyr)

  N/A   No   Yes
    0 665705 39608

> ## The "N/A" here is NOT the same as NA..
>
> acs09$fertyrL<-ifelse(acs09$fertyr=="Yes",TRUE,FALSE)
> table(acs09$fertyrL)

```

```
FALSE TRUE
665705 39608
```

So now we have the data in usable shape, how shall we calculate the ASFRs?

It is useful to separate this problem into two parts: first that of calculating the total number of events (births) to each woman of each age and category; and second that of calculating the risk set total person years at risk (of child birth) for women of each relevant age and category. These two quantities (matrices) can be thought of as a *numerator* and a *denominator*.

If we are clever enough to generate a numerator and denominator which happen to have the same structure – matrices or 13 dimensional arrays or whatever, then one could simply write:

```
■ > ASFR <- numerator / denominator
```

It turns out that `tapply()` makes this is pretty easy to do – once you know how:

```
> ## The numerator should be a matrix of counts of births that occurred
> ## to women by age .. and how about citizenship status
>
> table(acs09$citizen)
```

```

N/A
596433
Born abroad of American parents
7146
Naturalized citizen
41749
Not a citizen
59985
Not a citizen, but has received first papers
0
Foreign born, citizenship status not reported
0
```

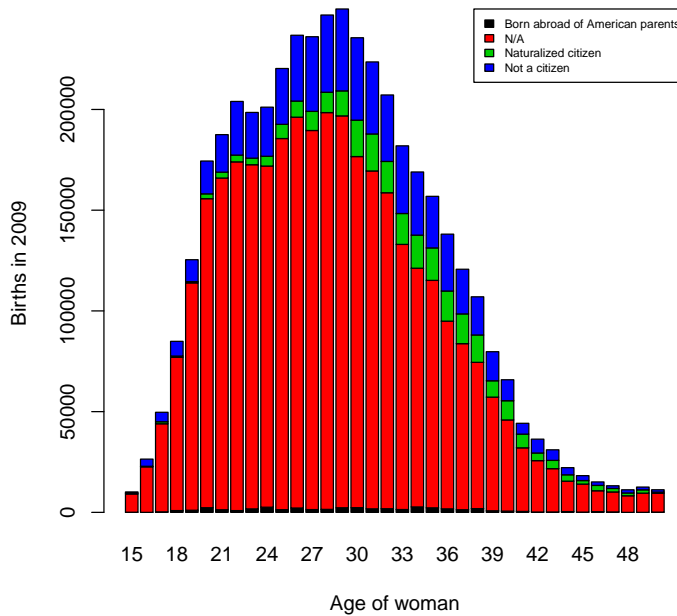
```
> ## the N/A in this case means born in the US.
> ## let's get rid of the obsolete responses which were relevant back in
> ## the 19th Century
>
> ## A clever trick for getting rid of levels in a factor which do not
> ## have any observations:
> acs09$citizenF<-factor(as.character(acs09$citizen))
> numer<-tapply(acs09$perwt*acs09$fertyrL,
+               list(acs09$ageN,acs09$citizenF),
+               sum)
> dim(numer)
```

```
[1] 36 4
```

```
> head( numer )
```

	Born abroad of American parents	N/A	Naturalized citizen	Not a citizen
15	68	9056	426	587
16	0	22597	283	3552
17	257	43721	1000	4698
18	889	76056	652	7304
19	1060	112845	621	10884
20	2296	153410	2403	16299

```
> ## A barplot would be nice right now
> barplot( t( numer ), xlab="Age of woman", ylab="Births in 2009",
+         col=( numercol <- 1:length( levels( acs09$citizenF ) ) )
> legend( x="topright", fill=numercol, legend=levels( acs09$citizenF ),
+        cex=.6 )
```



Now the denominator:

```
> ## the denominator is easier -- All we want is a weighted count of the
> ## number of women of each age who were "at risk" of giving
> ## birth. We are neglecting mortality -- because dead women do not
> ## fill out census forms so all we need to do is count
```

```

>
> denom<-tapply(acs09$perwt,
+               list(acs09$ageN,acs09$citizenF),
+               sum)
> dim(denom)

[1] 36  4

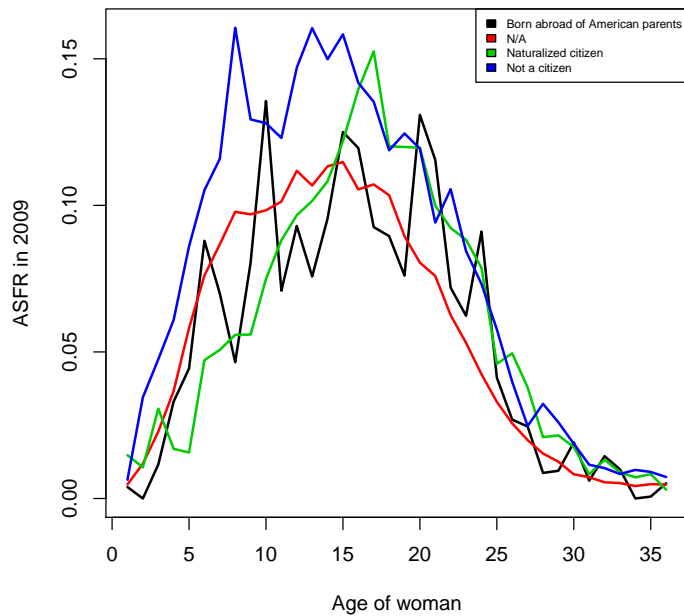
> dim(numer)

[1] 36  4

> ASFR<-numer/denom
> matplot(ASFR,type='l',lty=1,lwd=2,
+ xlab="Age of woman",ylab="ASFR in 2009",
+         col=(numercol<-1:length(levels(acs09$citizenF))))
> legend(x="topright",fill=numercol,legend=levels(acs09$citizenF),
+        cex=.6)
> TFR<-apply(ASFR,2,sum); TFR

```

Born abroad of American parents	N/A
1.989181	2.004676
Naturalized citizen	Not a citizen
2.142767	2.865514



## 5 Assignment

So now you know how to calculate ASFRs, your job is to find another even more interesting rate (or proportion or probability) to calculate. And to plot the result in an informative and attractive manner.

The easiest way to go would be to look for variables that ask “did respondent do X in the last year”. There are several of those. Ambitious students will want to consider other options as well. Since the IPUMS provides ACS data for each year between 2001 and 2009, you can – use calendar time in place of age – or in addition to age. (the rate school attendance by 17 year olds each year for example). If you choose to pursue something longitudinal, consider ways to keep your data set reasonably small:

- Take only the variables that you need.
- Exclude (during the IPUMS extraction process) observations that you do not need (men for example if you were looking at fertility)
- Limit your analysis to one state or region – California is interesting enough.

If you wind up with a data set that needs to be run on coale or galton you will want to do the entire job on the server as the /72hours directories are not shared on the network.

### 5.1 Getting data from IPUMS to R

So now your ready to do science on your own IPUMS data extract. This section is about how to get that extract into R.

The IPUMS website is pretty straightforward – if I wrote instructions on how to use it, you wouldn’t read them. So we’ll skip that and assume that you have managed on your own to:

- Visit [ipums.org/usa](http://ipums.org/usa) (and acquire a userid/password)
- Visit [ipums.org/usa](http://ipums.org/usa) again to select variables and samples.
- Receive email from IPUMS telling you that your extract is ready to download.

Once you have done all of this then read on.

#### 5.1.1 Download your data to /data/commons/userid

The data file that you are about to download is probably on the order 300MB. That’s a moderately large file that **does not belong in your home directory**. The smart place to put it is /data/commons/USERID where USERID is of course, your userid. So create the directory using... one of the 12 most important Unix commands...

```
@:> mkdir /data/commons/USERID
```

THEN browse to where the email from IPUMS tells you to go by opening the link that you were sent by IPUMS with the announcement that your data are ready and download.

The IPUMS download page presents you with several links. The ones that you want to pursue are the **Data** and the **Stata** files. By **RIGHT** clicking on the links you can select **Save Link As** to make sure that the Data file winds up in `/data/commons/USERID` and the Stata file winds up in `~/213/WeekN`.

Keep the page available as those code books are kind of handy too.

**GOTCHA: Is your browser not finding the /data/commons/ directory?** First look for `/data` in the **Filesystem** but if it isn't there then the problem is that your computer has not "mounted" `/data/commons`. The workstations only mount `/data/commons` and certain other directories as they are needed. To trick your machine into mounting `data` all you need to do is:

```
@:> ls /data/commons
```

Any command that references `/data/commons` will cause the machine to mount the directory and thereby make it useable.

### 5.1.2 Uncompress your data file onto /72hours

The "Data" file that you just downloaded is zipped (or compressed). Stata does not read compressed data files, so it behooves you to uncompress it before you try to run stata in the next step. `/72hours` is a great place to uncompress to and read from. In 72 hours the file will be erased, but by then you will be done with it **and** you will still have the **compressed** version in `/data/commons` in case you need to read it again.

```
@:> zcat /data/commons/USERID/usa_0001.dat.gz > /72hours/usa_001.dat
```

Note the use of the special symbol `>`. This redirects the output of whatever is on it's left into the file specified on its right.

`zcat` unzips a compressed file to the standard out, leaving the compressed version in tact.

### 5.1.3 Run STATA do produce a .sda file that R can read

The **STATA** file that you just downloaded should have a name like `usa_0001.do` ("do" NOT ".dat.gz"). It contains a set of stata commands read the data in the `usa_0001.dat (gz)` file, which you uncompressed in the previous step. To make the `.do` file do what we want, we'll need edit it lightly:

Edit the `usa_0001.do` file making the following two changes:<sup>1</sup>

1. Find the line:  
`using 'usa_0001.dat'` and change it to:  
`using /72hours/usa_0001.dat`
2. Go to the bottom of the file and add this line:  
`save /72hours/usa_0001, replace`

Save the file and exit.

#### 5.1.4 Run stata to produce the `.dta` file that R can read

Running `stata` in batch is the simplest way to run the program without understanding how it works:

```
@:> stata -m500 -b "do usa_0001.do"
```

After a few seconds look for a `usa_0001.log` file. At a minimum, run `tail` on it to verify that the data file was written:

```
@:> tail usa_0001.log
```

Look for something like this :

```
.  
. save /72hours/usa_0001, replace  
(note: file /72hours/usa_0001.dta not found)  
file /72hours/usa_0001.dta saved
```

If you don't find the above sort of happy message, the most likely problem is with the names of the files—so double check the two lines of code that you edited.

If you are working with a data file that is larger than 400 or so MB then you should move your work to `galton` or `coale` and increase the `-m500` to a number that is about 10 percent larger than your data file in megabytes.

#### 5.1.5 Reading the data into R

At this point you should be able to read the data into R.

```
> ## add the library that knows how to read .dta files  
> library(foreign)  
> ## without any arguments other than the file this can work. You might
```

---

<sup>1</sup>edit means use Emacs on. `cd` to `/data/commons/userid` and then type: `emacs usa_0001.do`. Obviously the filename could be different.

```

> ## wish to consult the help screen to see what options exist.
>
> acs<- read.dta(file="/72hours/usa_0001.dta")
> names(acs) ## do the column names look familiar ?

 [1] "year"      "datanum"   "serial"    "hhwt"      "region"    "statefip"
 [7] "gq"        "pernum"    "perwt"     "famsize"   "nchild"    "nchlt5"
[13] "age"       "sex"       "marst"     "marrno"    "marrinyr"  "widinyr"
[19] "divinyr"   "fertyr"    "race"      "raced"     "bpl"       "bpld"
[25] "citizen"   "yrnatur"   "yrimmig"   "hispan"    "hispanid"  "hcovany"
[31] "hcovpriv"  "hinsemp"   "hinspur"   "occ"       "migrate1"  "migrate1d"
[37] "diffmob"

> sum(acs$perwt) ##how many people do the observations represent? sum

 [1] 75974672

```