

Normal distribution and samples

February 5, 2008

Lada Adamic
SI 544

1 First, how to do a few things more efficiently in R

Plots need not take up that much space if they are placed in a table. We can use the `par()` function for this. And, just a reminder, since it is a bit tiresome to be always typing `libraries$POPU`, `libraries$TOTINCM`, etc., we have the option to `attach()` the data after reading it in (or loading a library). e.g.

```
states =  
read.table("http://www-personal.umich.edu/~ladamic/courses/si544w08/data/states.dat",  
sep="\t",quote="",head=T)  
attach(states)  
plot(obesity_rate ~ medianinc)
```

Now R will understand what **POPU** is without me having to type `libraries$POPU` every time. When we don't need `libraries` any more, and we don't want to get confused with similarly named columns for other data frames, we simply detach as so:

```
> detach(states)
```

Bye, bye states. Well, no, if you actually detached the states dataset, attach it back. OK, so let's plot some stuff. Let's do a 2x2 grid for plotting. What we're after is finding a correlation between obesity rates (percentage of adults that is overweight), and other statistics, such as per capita annual beer and wine consumption (in gallons), median income, and percentage of the state's population that voted for President Bush in the last election.

```
par(mfrow=c(2,2))  
plot(obesity_rate ~ beer,cex.lab=1.5)  
plot(obesity_rate ~ wine,cex.lab=1.5)  
plot(obesity_rate ~ medianinc,cex.lab=1.5)  
plot(obesity_rate ~ percentvotedbush,cex.lab=1.5)  
par(mfrow=c(1,1))
```

I've reset the plot to go back to a single axis just so I don't forget to do it later. I've also made the axis labels 50% bigger using the `cex.lab` option. This will leave them visible even when they are crunched together in a 2x2 grid.

We can make a few observations, namely that all that attribution of guts to beer seems unfounded. States with higher beer consumption don't tend to have higher obesity rates. Interestingly, higher wine consumption actually correlates with lower rates of obesity, as does higher average income. And voting for President Bush at first appears to correlate with higher obesity rates, but then appears to fall off for the highest rates of Bush-voting. In just a few lectures, we'll learn to calculate correlation coefficients, which will tell us whether these apparent trends are statistically significant.

We can also look at pairwise relationships between all columns in our data set using the `plot()` or `pairs()` function:

```
plot(states)
# or equivalently...
pairs(states)
```

Do you see any other correlations among the variables? What could be a hidden variable that may be behind many of these correlations?

As an aside, if I hadn't attached the states data set, I could have still told plot what I meant by typing:

```
> plot(beer ~ wine,data=states)
```

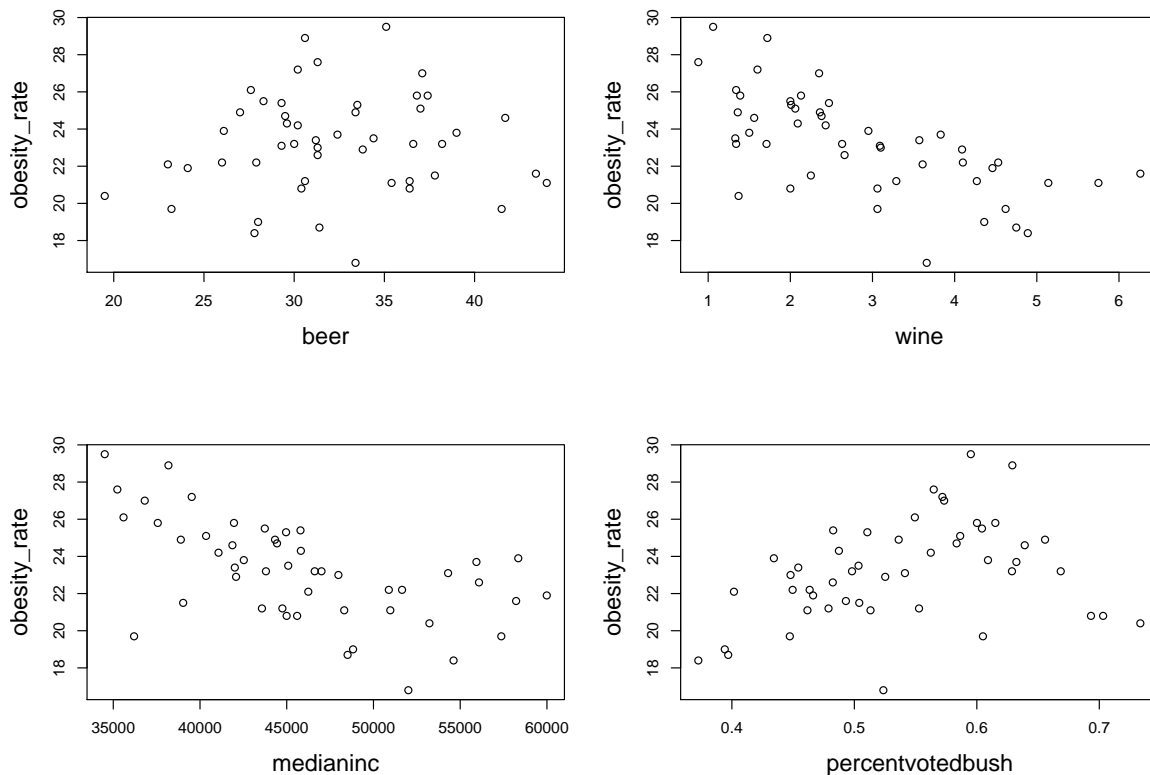


Figure 1: Attendance vs. population served for each library.

Hopefully, at this point, you've started to be curious about what a few of those outliers are. Some states have fairly low median income, but somehow stay fit. Which ones are they? For this this, we can use the `identify` function.

```
> plot(obesity_rate ~ medianinc)
> outlierpoint = identify(medianinc,obesity_rate,n=1)
> states[outlierpoint,]
      state medianinc obesity_rate beer wine percentvotedbush
25 Montana   36200         19.7  41.5  3.06         0.605
```

What happened (and you'll get to try this in class), is that I called the `identify()` function with the same x and y vectors as the plot. I told it I wanted to quit after identifying `n=1` points. If I don't specify `n`, then I would just keep clicking until I was done. Then, on windows I would click on the right mouse button and tell it I wanted to "stop". On a Mac, you would hit the escape 'ESC' key.

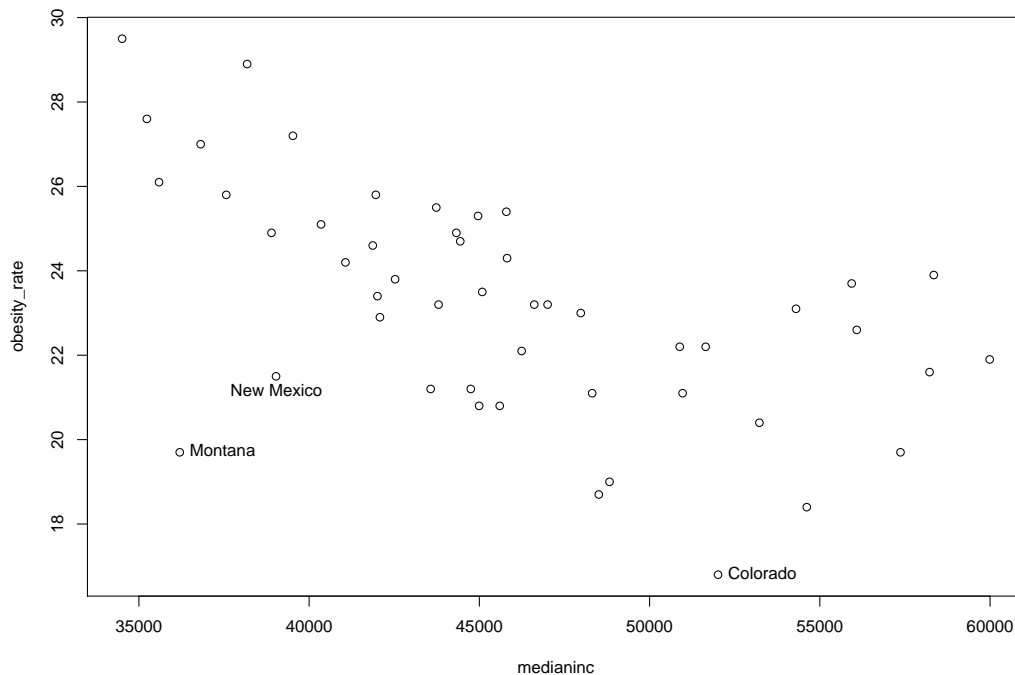


Figure 2: Attendance vs. population served for each library with Ann Arbor identified and labeled.

This was an OK way to go about it, but maybe a bit cumbersome. I could actually just have R label anything I click on. Let's try that:

```
> plot(obesity_rate ~ medianinc)
> identify(medianinc,obesity_rate,label=state)
```

Aha! So the mountain states manage to stay relatively thin, no matter their income.

On the other hand, maybe we want to locate our favorite state on the plot. First let's get the row index of Michigan:

```
> states[state=="Michigan",]
      state medianinc obesity_rate beer wine percentvotedbush
21 Michigan    45793         25.4  29.3  2.47             0.4827
```

So I pick the row that corresponds to Michigan. I add a point on the plot using a big fat red circle to mark the Michigan, and then label it in a big red font for good measure. But in order to know where to place the text, I can either use the coordinates (i.e. income and obesity rate), or use the **locator** function.

```
michigan = states[state=="Michigan",]
points(michigan$medianinc,michigan$obesity_rate,col="red",cex=2,lwd=5)
text(locator(1),"MI",col="red",pos=3,cex=2)
```

What exactly did I tell the **text()** function? I told it where to put the text, what the text is, to make it red, to make it big (cex=2) and to position it above the text (pos=3). Try it, it's fun. Et Voila:

Yikes, I guess we're on the heftier for our median income range.

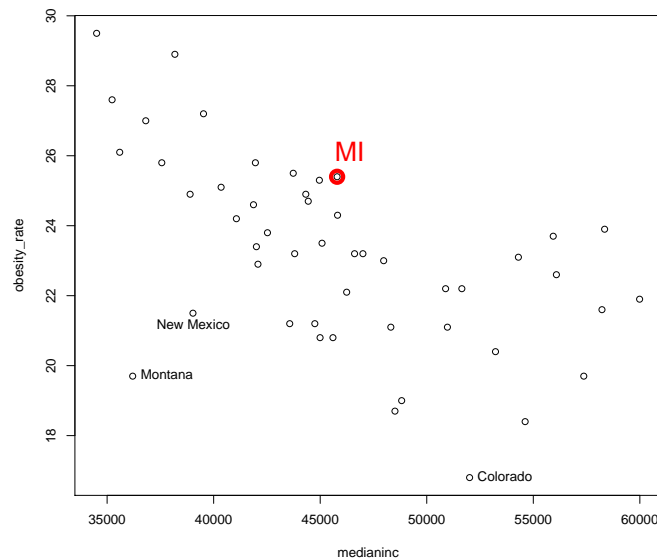


Figure 3: Obesity vs. median income with Michigan labeled

2 Samples

As we learned previously, even if the random variable X is not normally distributed, the sample average \bar{X} tends to be. For small samples, however, we should actually use the student t distribution. Why use the student t? We should use it in the case where we are estimating the sample error mean (SEM) from the sample itself. The t distribution allows more extreme values for small samples than the normal distribution (and is therefore more likely to accept the null hypothesis). A parameter of the t-statistic is the number of degrees of freedom. After we compute the mean of a sample of size n , we have $(n - 1)$ degrees of freedom left.

Let's see what we have for the normal and the t distribution when the sample size (and correspondingly, the degrees of freedom) are small:

```
> ## normal distribution
> # P[Z <= 1], where Z is the distance from the mean, measured in SEMs.
> pnorm(1)
[1] 0.8413447
> # P[Z <= -1]
> pnorm(-1)
[1] 0.1586553
> # Between 1 SEM below & 1 SEM above
> pnorm(1) - pnorm(-1)
[1] 0.6826895
> # if we have a sample of just 9, the student distribution will differ
> # from the normal
> pt(1,8)
[1] 0.8267032
> pt(-1,8)
[1] 0.1732968
> pt(1,8)-pt(-1,8)
[1] 0.6534065
```

So for the normal distribution we have 68% of the data points lying within 1 standard deviation of the mean, but for the student t distribution it is a bit less: 65%. If we increase our sample size to 100, the normal and t distributions are quite close:

```
> pt(1,99)-pt(-1,99)
[1] 0.6802515
```

We can actually also compare graphically the normal and t distributions like so:

```
> plot(x,dt(x,8),type="l",xlab="No. of SEMs from the mean",ylab="probability density")
> lines(x,dnorm(x),type="l",lty=2)
> legend(-3,0.35, c("t(8 d.f.)","normal"),lty=c(1,2))
```

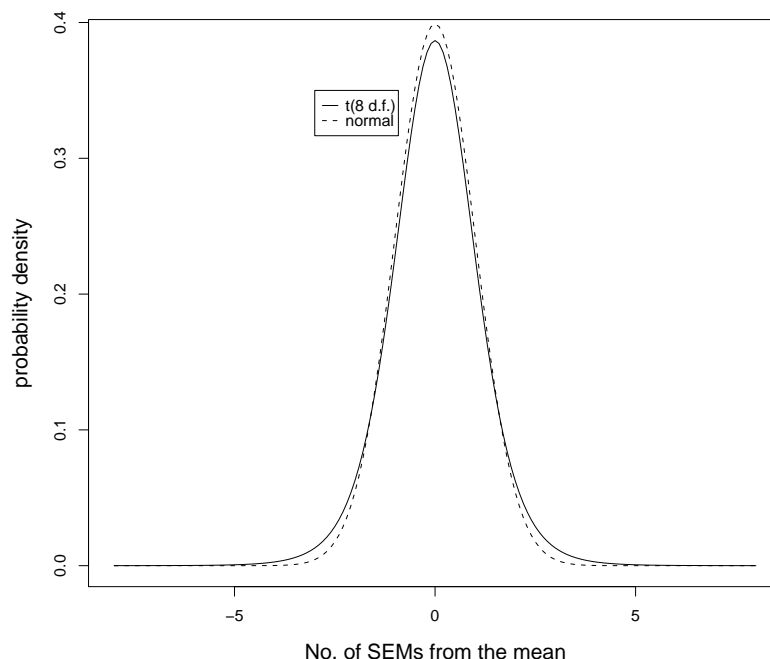


Figure 4: t vs. the normal distribution.

What about constructing a confidence interval. 95% of the distribution will lie within 1.96 standard deviations.

Typically we don't know the standard deviation of mean (SEM), so we estimate it from the standard deviation of the sample: $SEM = \frac{s}{\sqrt{n}}$.

We can simulate drawing many samples as follows (remember to load the libraries data as you did before):

```
# number of samples - usually you do one or two in real life
# here we are simulating what all can happen
samples = 1000

# the size of each sample samplesize = 35

# the vector we'll be looking at - the av. # librarians per 1,000 population
libperpop = libraries$LIBRARIAN/libraries$POPU*1000
```

```

# we calculate the mean and variance of the entire underlying population
mean(libperpop)
sd(libperpop)

# initialize the vector of sample means
samplemeans = c()

# loop through all our samples
for (i in 1:samples) {
  #calculate the sample mean
  samplemeans[i] = mean(sample(libperpop,samplesize,replace=FALSE))
}

#the mean of the means: xbar
xbar.mean = mean(samplemeans)

# the standard deviation of xbar
# normally we would only estimate this from our one sample
# but here we have the luxury of calculating it
xbar.sd = sd(samplemeans)

> # calculating the 95% confidence interval
> xbar.mean + xbar.sd*1.96
[1] 0.4251261
> xbar.mean - xbar.sd*1.96
[1] 0.1955790
>
> # compare with estimate from original population parameters
> mean(libperpop)+1.96*sd(libperpop)/sqrt(samplesize)
[1] 0.4156066
> mean(libperpop)-1.96*sd(libperpop)/sqrt(samplesize)
[1] 0.1941255

hist(samplemeans,50)

```

Note that the two ways of obtaining a confidence interval are nearly identical.

But all the above two methods of obtaining a confidence interval are dependent on either knowing the standard deviation of the whole population, or being able to take lots and lots (1,000) samples. In real life, this is rarely the case. Instead, we can use the t distribution to estimate the error of our estimate directly from the single sample that we took. We can use the function qt to find the quantile (in standard errors) at which we are at a given confidence level. For example, with a sample size of 35, we have 45 degrees of freedom.

```

> samplesize=35
> # constructing a confidence interval from a single sample
> singlesample = sample(libperpop,samplesize,replace=FALSE)
> xbar = mean(singlesample)
> sem = sd(singlesample)/sqrt(samplesize)
> degreesoffreedom = samplesize-1
>
> multiplier = qt(0.975,degreesoffreedom)
>
> xbar - multiplier*sem

```

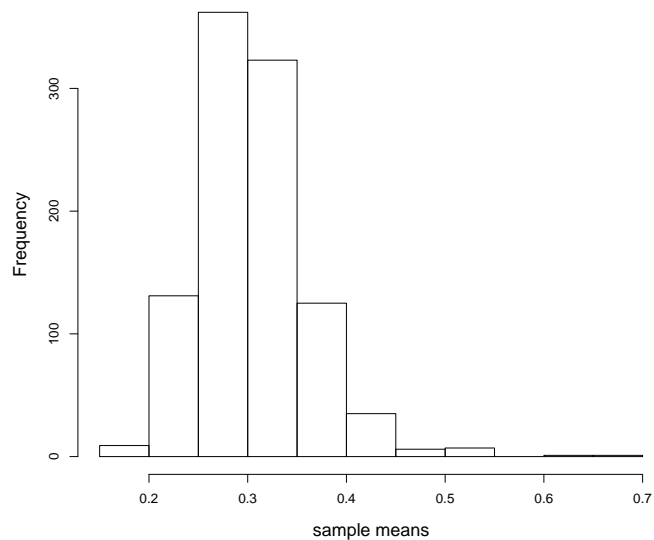


Figure 5: Distribution of sample means for av # of librarians per 1,000 population

```
[1] 0.2272420
> xbar + multiplier*sem
[1] 0.4685247
```

3 Confidence intervals for population proportion

```
#####
## Confidence interval for proportion (from R book)
#####

# example 7.2 presidential job performance
# out of a random sample of 1,013 likely voters
# 466 voters rated the president's job performance as
# "good" or "excellent"
# find the confidence interval

# n = 1013 is large enough such that we can use the
# normal approximation

n = 1013
phat = 466/n # estimate of the average proportion
SE = sqrt(phat*(1-phat)/n)
# that was the formula for the SE of the proportion
# (something new)
#let's select a 95% confidence interval

alpha = 0.05
```

```
# find the corresponding z-value
# this is the number of standard deviations out from
# the mean that we go for a 95% confidence interval

zstar = -qnorm(alpha/2)

# now we construct the confidence interval
> c(phat - zstar * SE, phat + zstar * SE)
[1] 0.4293281 0.4907114
```