PLSC 502: "Statistical Methods for Political Research"

Univariate Graphics
September 6, 2016

Before we get into estimating models using our data, it's always a good idea to just *look* at them. Political scientists typically don't do nearly enough looking at data, preferring instead to estimate dozens (or hundreds) of statistical models instead. We should do more, for a lot of reasons...

Three Big Reasons to Look at Your Data

1. Doing so will give you a sense of *what's going on* in the world you're studying.
2. Doing so is also often the best way to detect *errors* in the data.
3. You might *learn something*.

Example: Africa, 2001

We'll focus to day on looking at data one variable at a time – that is, on graphical methods for *univariate* data. That may sound easy, and boring, but trust me: it's neither. Next class, we'll consider methods for multivariate data (that is, looking at more than one variable at a time). As a running example, we'll be using data on 43 countries in Africa, for the year 2001.[1] Summary statistics – we'll get to those a bit later – are on the first page of the handout; the variables are:

- `ccode` – the country's COW country code (an identifier),
- `cabbr` – a country name abbreviation,
- `country` – the country's name,
- `population` – the country's population,
- `popthou` – `population` divided by 1,000,
- `popden` – population divided by the country's area, in km^2,
- `polity` – the country's aggregate POLITY democracy/autocracy score, ranging from -10 (autocratic) to 10 (democratic)
- `gdppppd` – GDP per capita, in thousands of constant U.S. dollars,

[1] There's no particular significance to these data, other than that they come easily to hand and that they contain both categorical and continuous variables.

- `tradegdp` – total trade, as a fraction of GDP,
- `war` – an indicator variable coded 1 if the country was engaged in an interstate war and 0 otherwise,
- `adrate` – the adult (16+) HIV infection rate,
- `healthexp` – public and private expenditures on health, per capita, a a (constant-dollar) percentage of GDP,
- `subsaharan` – an indicator variable coded 1 if the country is in sub-Saharan Africa and 0 otherwise,
- `muslperc` – the percentage of the population that is Muslim,
- `literacy` – the percentage of the population that is literate (by the WHO definition),
- `internalwar` – an indicator variable coded 1 if the country in question had an internal (civil) war and 0 otherwise, and
- `intensity` – a four-point ordinal scale, indicating the intensity of the internal war (if any), ranging from zero (no war) to three (war with > 1000 battle deaths in 2001).

For purposes of eyeballing data, we can treat interval- and ratio-level data the same. We'll talk about each of these, as well as binary (dichotomous) data, in turn.

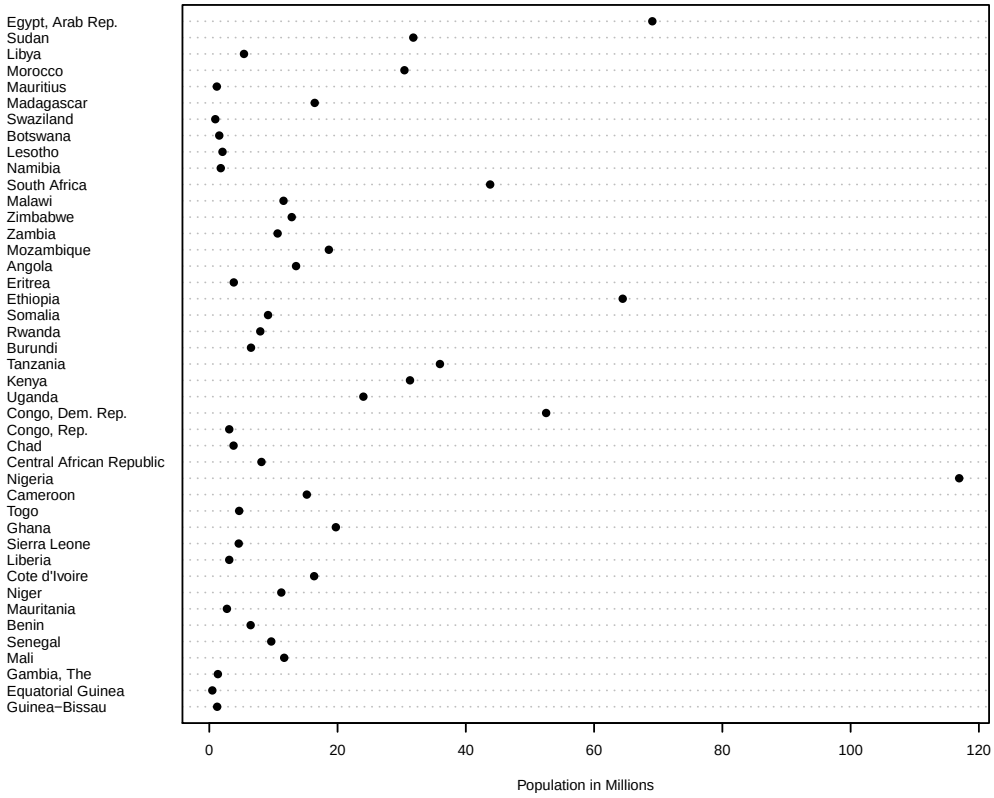
Dotplots and Barplots

The most basic thing a plot can do is to simply *display* the data in question. Despite the fact that doing so doesn't reduce the complexity of the data in any way, this can still be a valuable thing to do. The most common way to display data are *dotplots* and *barplots* (also sometimes called *dotcharts* and *barcharts*). These use a location on a horizontal axis to indicate a variable's value, while the vertical axis simply lists the observations, usually in some order.

We can use a dotplot to examine (e.g.) the population of our 43 African countries:

```
> with(Africa, dotchart(popthou/1000,pch=19,labels=country,
              cex=0.5,xlab="Population in Millions"))
```

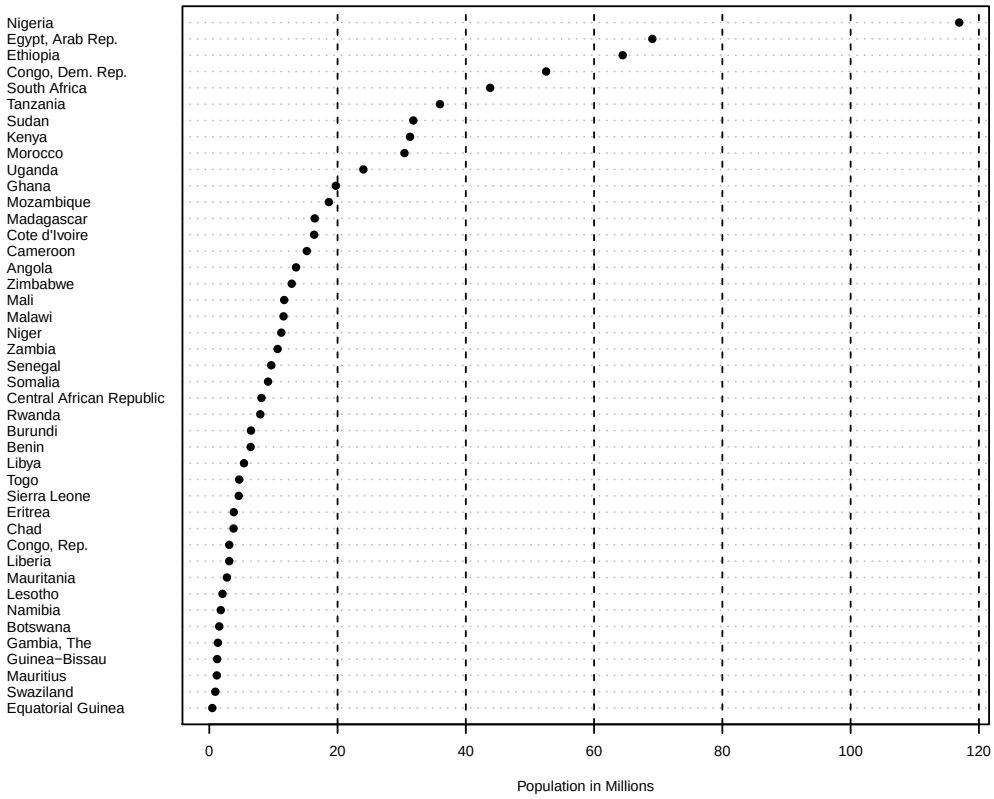
Figure 1: Dotchart of African Country Populations in Millions, 2001



This plot is fine but could be better. For example, we can *sort* the data on the variable of interest; we can also add *reference lines* to make the plot easier to read:

```
> Africa<-Africa[order(Africa$popthou),]
> with(Africa, dotchart(popthou/1000,pch=19,labels=country,
                        cex=0.5,xlab="Population in Millions"))
> abline(v=c(20,40,60,80,100,120),lty=2,lwd=1)
```

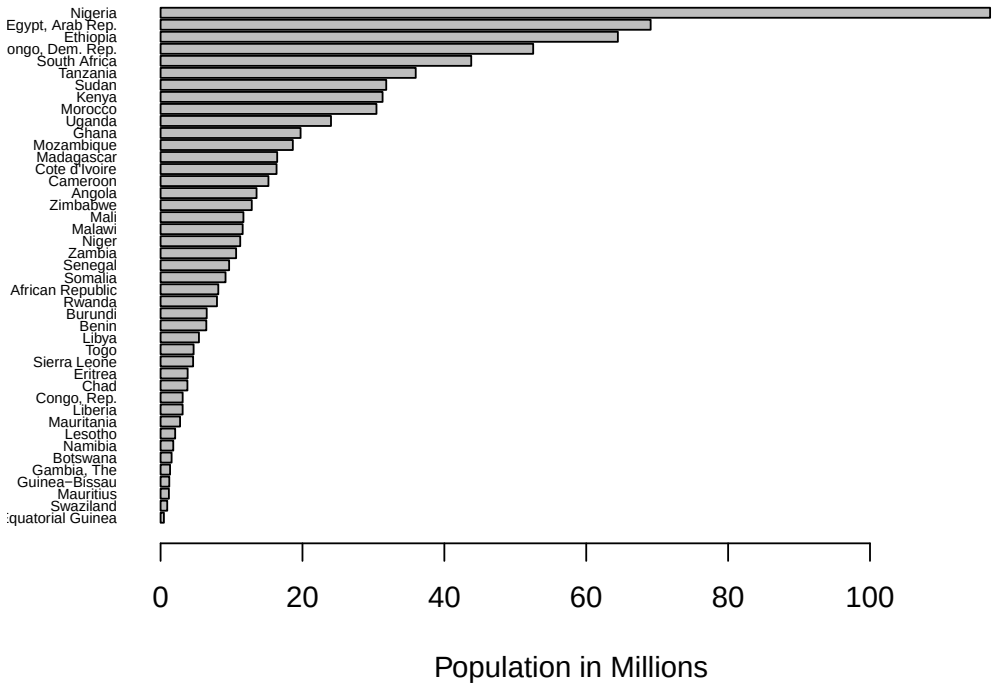
Figure 2: Dotchart of African Country Populations in Millions, 2001



That's actually more useful. A barplot is nothing more than a dotplot that uses bars instead of dots:

```
> Africa<-Africa[order(Africa$popthou),]
> with(Africa, barplot(popthou/1000,horiz=TRUE,names.arg=country,
            las=1,cex.names=0.5,xlab="Population in Millions"))
```

Figure 3: Barchart of African Country Populations in Millions, 2001



The Noble Histogram

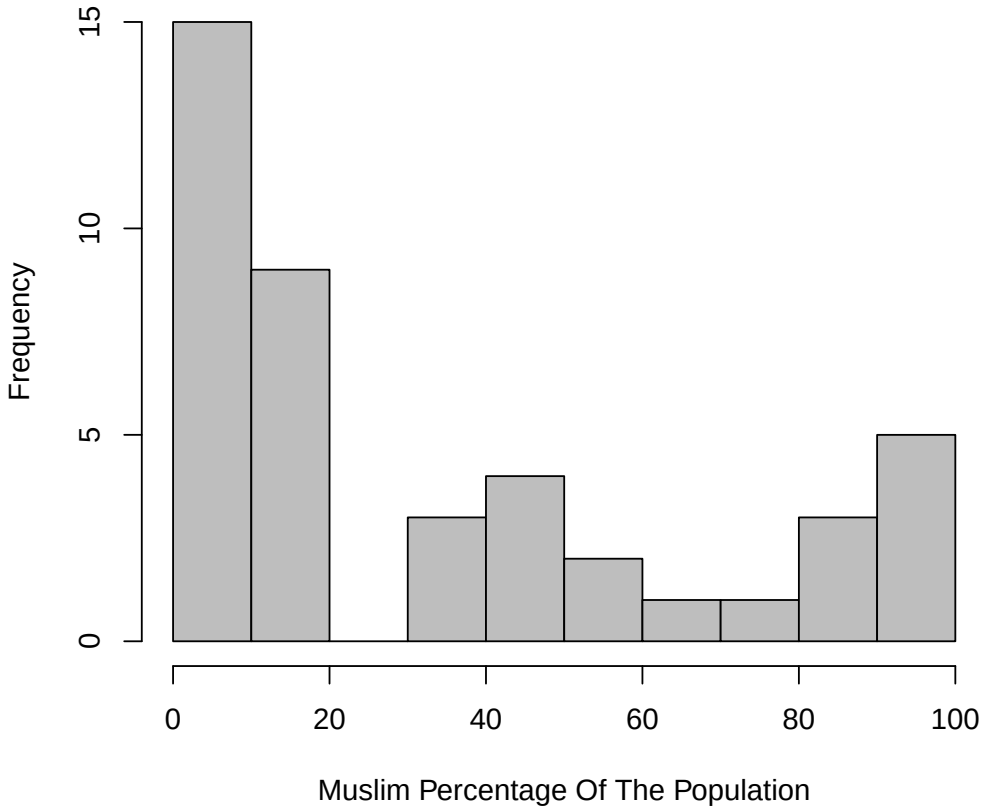
Probably the most useful univariate summary figure is the *histogram*; also commonly (but incorrectly) known as the bar chart. A histogram plots the values of the data on the X-axis, and the *density* (that is, the frequency) of those data on the Y-axis. A simple histogram (along with its associated R command) looks like this:

```
> with(Africa, hist(muslperc,breaks=10,col="grey",main=" ",
             xlab="Muslim Percentage Of The Population"))
```

This tells us that:

- Fifteen countries had Muslim populations that were between zero and ten percent of their total population,
- Nine countries had Muslim populations between ten and twenty percent,
- etc.

Figure 4: Histogram of Muslim Population Percentages in Africa, 2001



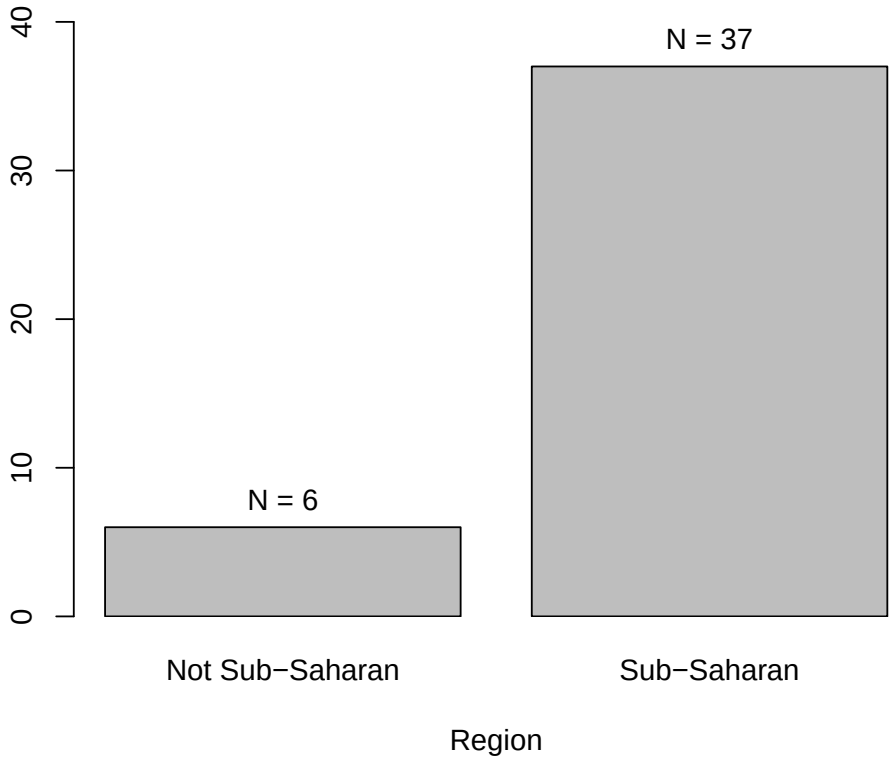
For nominal and ordinal data, a histogram typically just places observations into their respective categories. For continuous data, histograms sort data into "bins" of a particular width; it often takes a bit of experimentation to get the bin width exactly right, though a good rule of thumb is to use values that are either intuitive or substantively meaningful.

The nice thing about a histogram is that it's very intuitive, and can tell you a lot about the data in question. Moreover, you can do some things to make it even more useful, like adding labels, adjusting "bin" sizes, etc. Best of all, histograms will "work" with any level of measurement: binary, nominal, ordinal, and interval/ratio. For example, we can display a summary of the nominal/binary variable indicating whether a country is in northern or sub-Saharan Africa using a histogram (note that the R syntax is a bit different, in that it uses `barplot`):

```
> xx<-with(Africa, barplot(table(subsaharan),col="grey",main=" ",
         xlab="Region",ylim=c(0,45),
         beside=TRUE,xpd=FALSE))
> # Add Ns to top of bars:
```

```
> with(Africa, text(xx, table(subsaharan),pos=3,
         labels=paste("N = ",c(table(subsaharan)),sep="")))
```

Figure 5: Histogram of Region in Africa



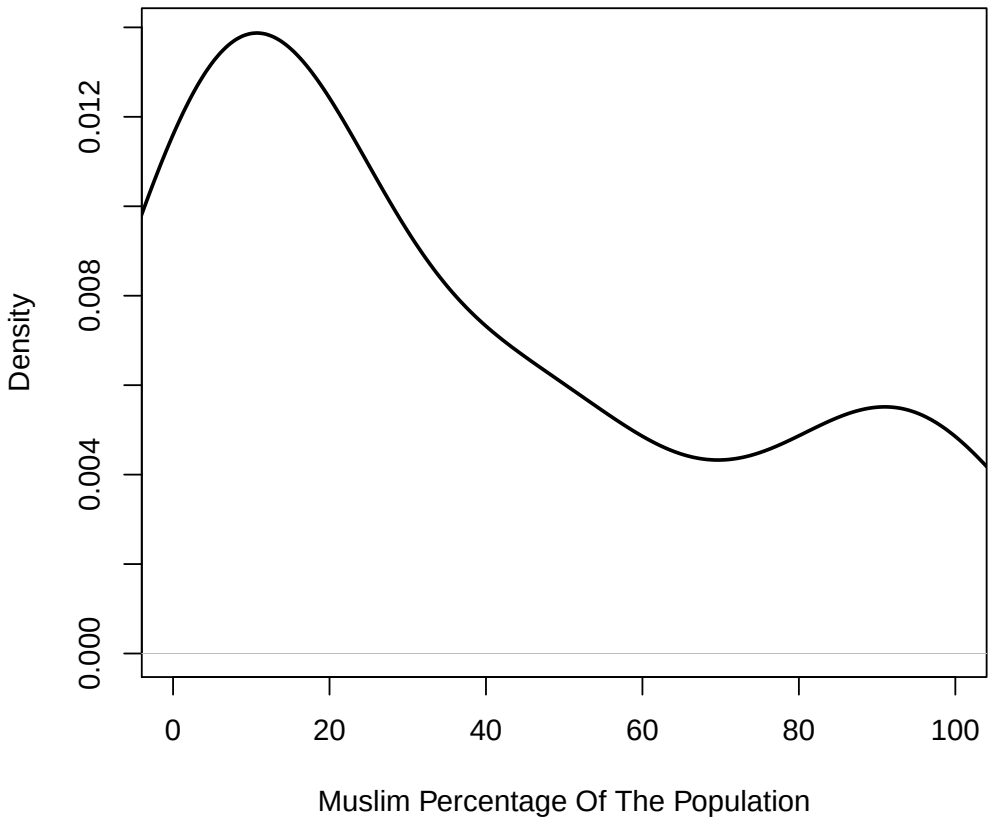
Super-Histograms: Kernel Density Plots

Histograms are fine, but particularly when the variable of interest is continuous (i.e., interval- or ratio-level), they can lose information. A better option there is the *kernel density plot*. This is, in a sense, a variation on a histogram, but with the data "smoothed" across bins to get a more accurate sense of its distribution.

The details of that smoothing are (potentially) complicated, since there are a lot of different smoothing algorithms that can be used. But the general effect looks like this:

```
with(Africa, plot(density(muslperc),t="l",lwd=2,main="",
             xlab="Muslim Percentage Of The Population",
             xlim=c(0,100)))
```

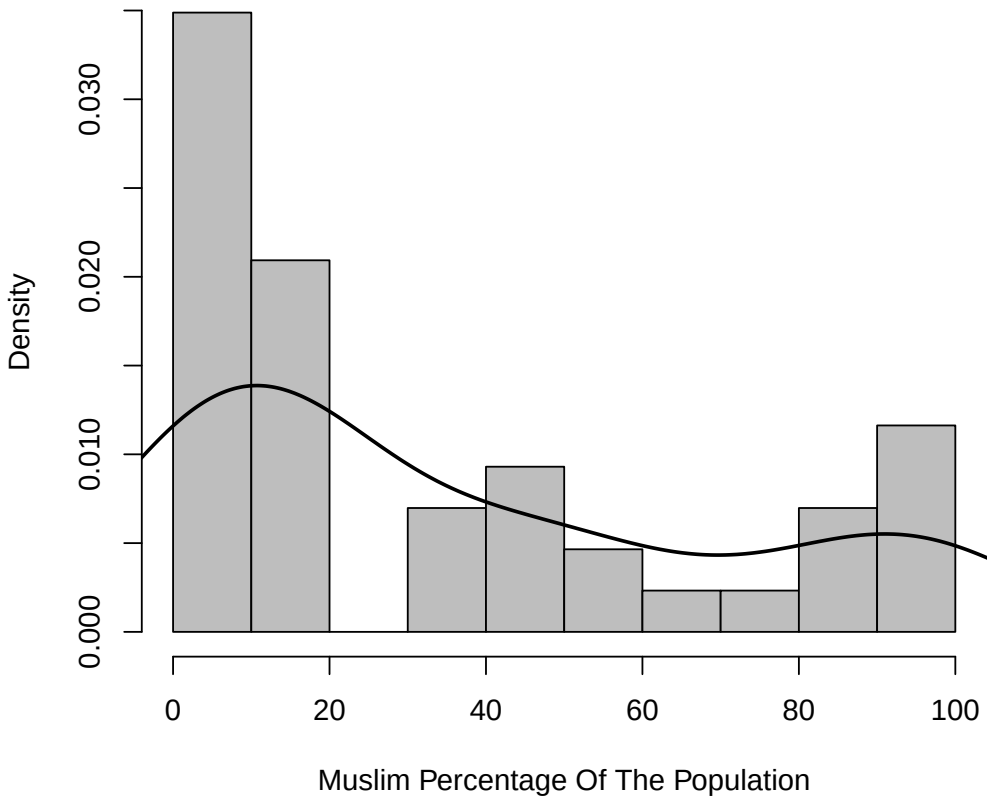
Figure 6: Kernel Density Plot of Muslim Population Percentages in Africa, 2001



Note that this gives a little different picture of the distribution than the histogram. Its also possible (and often valuable) to overlay one over the other:

```
with(Africa, hist(muslperc,breaks=10,col="grey",main=" ",
                  xlab="Muslim Percentage Of The Population",
                  freq=FALSE))
with(Africa, lines(density(muslperc),t="l",lwd=2))
```

Figure 7: Histogram and Kernel Density Plot of Muslim Population Percentages in Africa, 2001

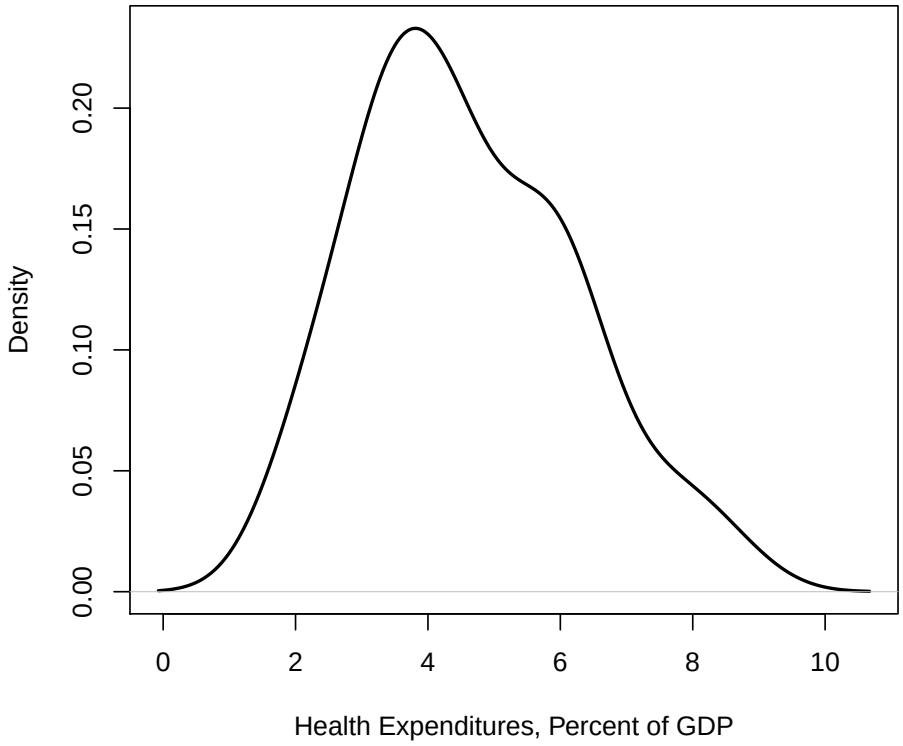


Q-Q Plots

Sometimes it can be useful to see if some data "fit" a particular distribution well or not. For example, we might expect (e.g.) the distribution of test scores to be roughly "bell-shaped," perhaps fitting a Normal distribution.

In our data, for example, consider a variable that measures government health expenditures per capita:

Figure 8: Kernel Density Plot of Health Expenditures in Africa, 2001



That looks like it might be roughly normally-distributed, and we might expect it to be theoretically as well.[2] If it *is* normally-distributed, then we can know exactly how many cases we would expect to have at or below the 10th percentile, the 20th percentile, and so forth. Note that:

- At the 0th percentile, there will be no data below it,
- At the 100th percentile, there will be no data above it (i.e., higher than those values),
- In between, if the data are (say) normally distributed, then we should observe 10 percent of the data at or below the 10th percentile of the theoretical normal distribution, 20 percent below the 20th percentile, and so forth. So, for a normal distribution with a mean of 4.6 and a standard deviation of 1.6 (which are the mean and standard deviation of the observed data), we would expect:
    - about 5 percent of the cases to have values lower than 2.0,

[2]If the potential significance of the variable being normally distributed is lost on you at the moment, don't worry; we'll get to that soon enough.

- about 16 percent of the cases to have values below 3.0,
- 50 percent of the cases to have values below 4.6,[3]
- 93 percent of the cases to have values below 7.0, etc.

The idea behind *quantile-quantile plots* (often called *Q-Q plots*) is exactly this: To plot the empirical distribution of the data against some theoretical distribution, to see how well the data "fit" that distribution. Details are in Fox; the important thing is that, the better the data fit the theoretical distribution, the closer the plot will be to a 45-degree line with a zero-intercept.

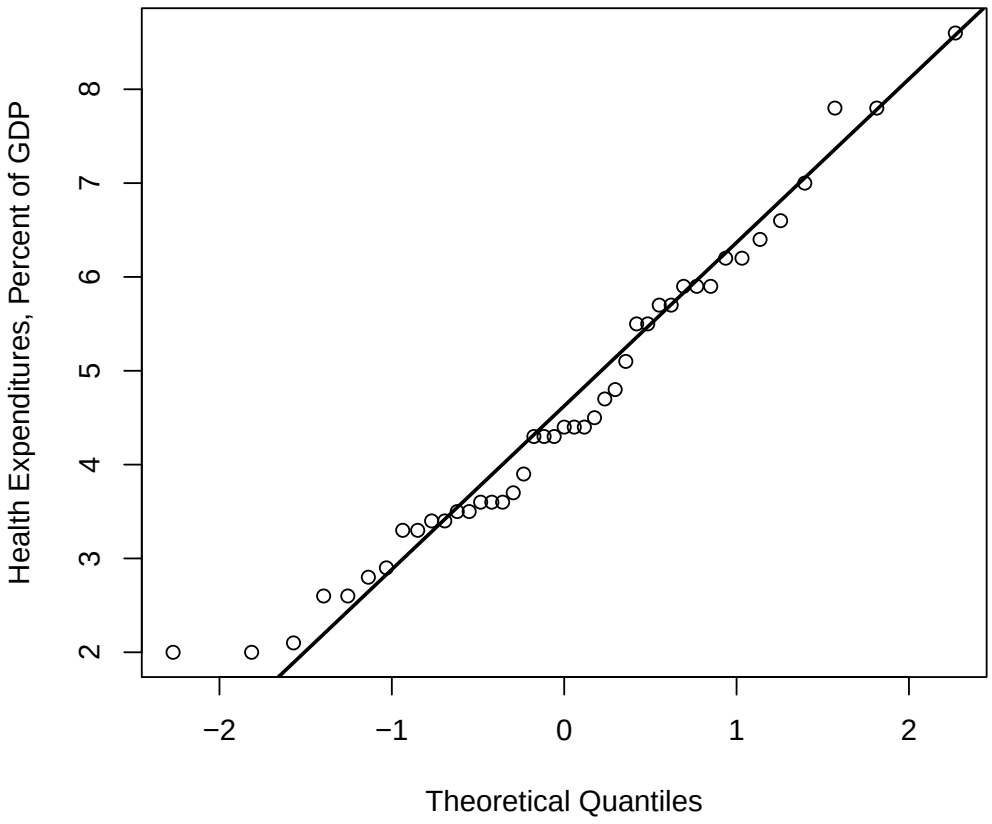
For our health expenditure data, the Q-Q plot looks like this:

```
with(Africa, qqnorm(healthexp,main="",
                    ylab="Health Expenditures, Percent of GDP"))
with(Africa, qqline(healthexp,lwd=2))
```

The fact that the data all lie close to the 45-degree line here means that the data closely approximate a normal distribution. In contrast, if we were to do this with our (highly bimodal) Muslim population data, we'd get:

[3]Since the normal distribution is symmetrical, its mean and its median are identical.

Figure 9: Normal Q-Q Plot of Health Expenditures in Africa, 2001

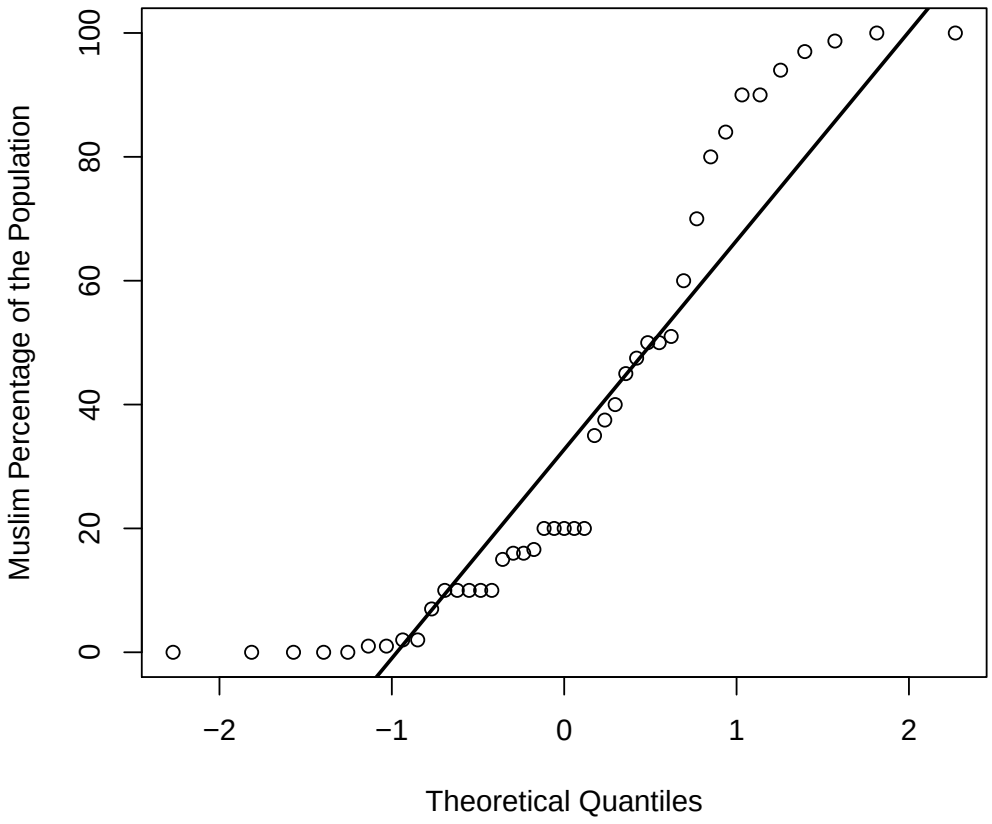



```
with(Africa, qqnorm(muslperc,main="",
                    ylab="Muslim Percentage of the Population"))
with(Africa, qqline(muslperc,lwd=2))
```

Notice several things about this plot:

- It doesn't fit the diagonal line very closely, suggesting that the data are not particularly close to a normal distribution,
- In particular, there are more data points above the lowest values of the (theoretical) distribution than there should be, fewer in the "middle," and more at the highest values – again, consistent with the fact that the data have "bumps" at the lowest and highest values.
- In addition, these data are *bounded* – there can't be percentages less than zero or more than 100. That is also reflected in the plot, which has the classic S-shaped Q-Q plot common to proportion/percentage data (and again suggesting that the normal distribution isn't the best fit).

Figure 10: Normal Q-Q Plot of Muslim Population Percentages in Africa, 2001



A few final words about Q-Q plots:

- While the normal distribution is the most commonly used, one can in theory fit any distribution to data in this way. Besides the normal, the most common ones (for reasons we'll see later on) are the uniform distribution and the chi-square distribution, although others (for example, normal probabilities, t distributions, and others) can be fit as well.
- Fox notes that it is also a good idea to include confidence bounds for the plot, and I agree. Unfortunately, this is not easily done in Stata, though it is more straightforward in R.
- As we'll see next time, one can also plot one variable against another in this way.

Boxplots

Boxplots (Tukey 1977) are essentially graphical representations of summary statistics: the mean, median, and quantiles of empirical data. By convention:

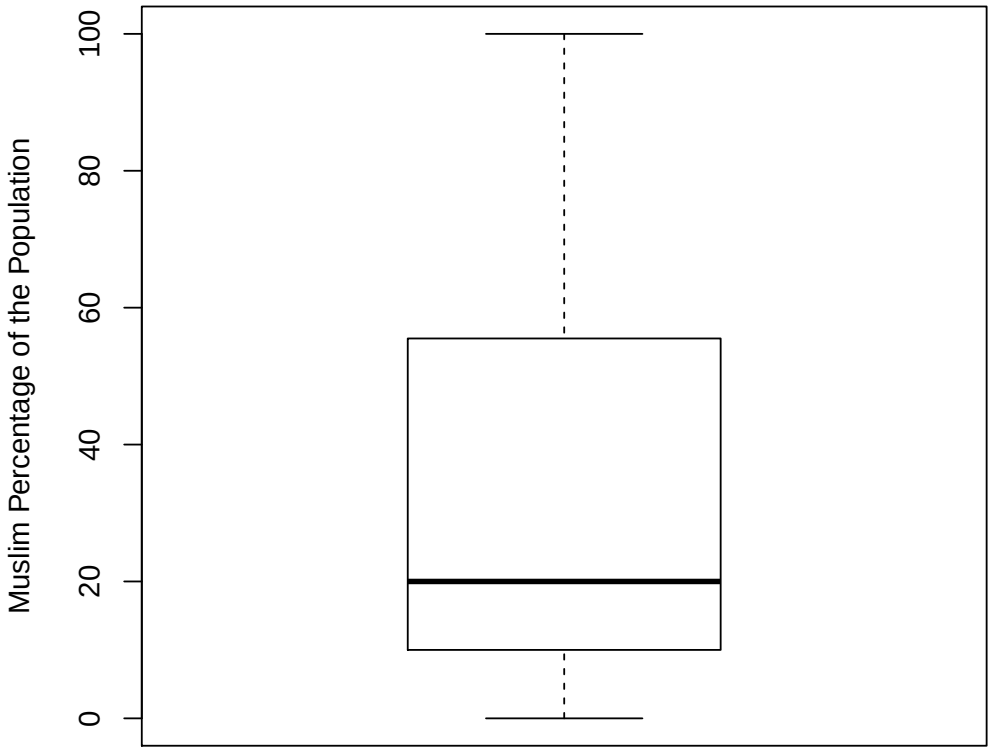
- The scale of the graph is set to the extremes of the data,
- The "box" is drawn between the 25th and 75th percentiles of the empirical distribution of the data,
- a line (or sometimes a circle) within the box denotes the median value of the empirical distribution, and
- the "whiskers" (the lines sprouting from each end of the box) extend to either (a) the extreme observed value of the data in that direction, or (b) to the most extreme "non-outlying" observation,[4] whichever is smaller,
- outliers – observations outside the "whiskers" – are then represented by circles or asterisks.

[4]Defined as outside the "fences;" see Fox (or Tukey) for detail on this.

A boxplot of the Muslim population percentage variable looks like this:

```
with(Africa, boxplot(muslperc,main="",
                     ylab="Muslim Percentage of the Population"))
```

Figure 11: Boxplot of Muslim Population Percentages in Africa, 2001



Note that the boxplot tells us quite a bit about these data:

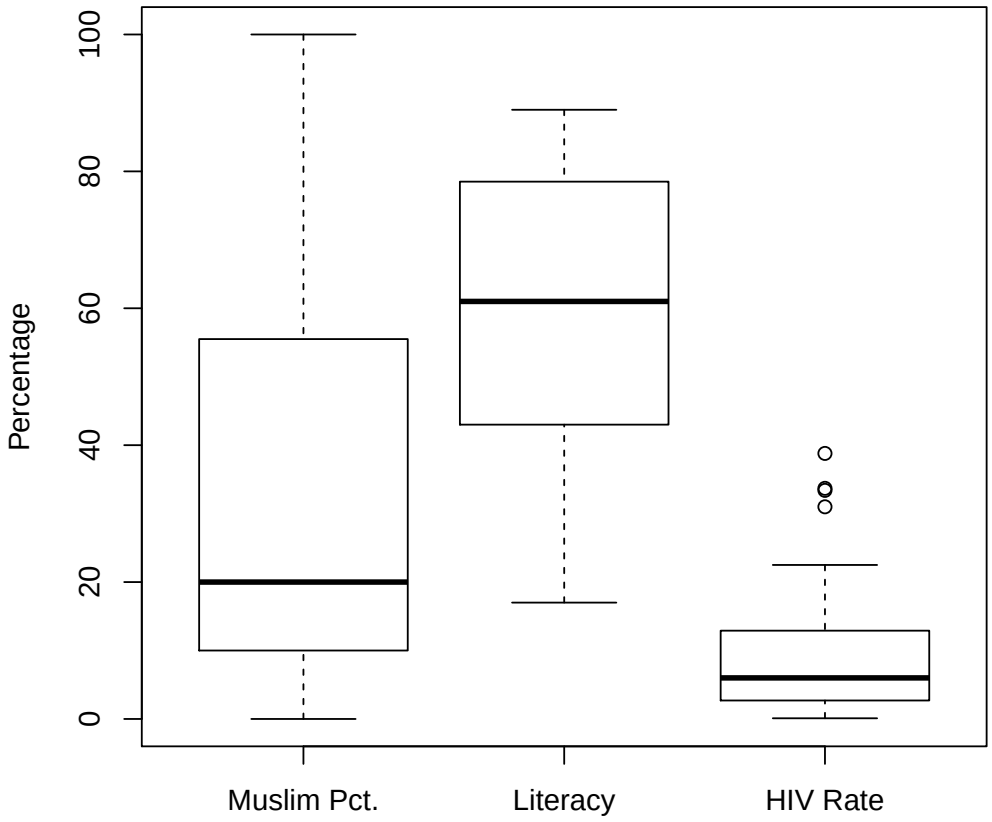
- That the median value is roughly 20 percent,
- That they are right-skewed; the median is significantly lower than the "middle" of the data,
- That most countries have Muslim populations between 10 and 60 percent, and
- That there are no real extreme outliers.

Notice, however, what it *can't* tell you: whether the data are unimodal, bimodal, or what. In fact, a boxplot is nothing more than a graphical representation of something akin to the `summarize..., detail` command in Stata; it gives you summary statistics and information, but can't tell you everything you might need to know about the *shape* of the distribution.

At the same time, one of the valuable things about boxplots is that you can place several on the same plot, to get a sense of how variables compare to one another – so long as the variables are on roughly the same "scale." For example, we can add boxplots of the distributions of the adult literacy rate and the adult HIV/AIDS infection rate for the 43 countries to our plot of the Muslim percentage:

```
boxplot(Africa[,c("muslperc","literacy","adrate")],main="",
        ylab="Percentage",names=c("Muslim Pct.","Literacy","HIV Rate"))
```

Figure 12: Boxplot of Muslim Population Percentage, Adult Literacy, and HIV Prevalence Rates in Africa, 2001



From this, we can learn quite a bit about these three variables...

Special Cases: Binary and Categorical Data

The data we have and care about are often not nice and continuous like the ones we've discussed so far. There are some challenges that come with graphing such data. To be specific:

- Histograms will work just fine for binary/categorical data, including data at the nominal level.
- Density plots, on the other hand, are generally not recommended for use with binary data (they tend to plot two "humps" at zero and one, which is not all that informative) or nominal-level data, and are of only limited use for ordinal data. In general, *the "lumpier" the data (that is, the fewer discrete categories into which the cases are coded), the less useful density plots are.*
- The same is true for Q-Q plots, which are completely useless for nominal-level data, almost completely useless for binary data, and relatively useless for ordinal data.
- Boxplots are similar to Q-Q plots: really not valuable for discrete data of any sort, and particularly useless (in fact, potentially dangerous) for nominal-level data.

In other words: If you have an urge to plot univariate discrete data, histograms are really the only game in town...

A Few Other Univariate Plots

Other univariate plots include:

- *Stripplots* (or *stripcharts*), often used in conjunction with scatterplots or other multivariate graphics;
- *Pie charts* (Don't. Just don't.)
- "Donut" plots (ditto)
- "Stem and leaf" plots (don't bother...)

Graphics Using Stata

Stata will, of course, also do everything here, and more. Its command syntax is pretty intuitive, taking the form:

```
. graph command variable
```

where `command` is the kind of plot you want to make, and `variable` is the variable you want to plot. E.g.,

```
. graph histogram muslperc
```

Many commands also allow you to forego the `graph` at the beginning, shortening the command to (e.g.):

```
. histogram muslperc
```

In Stata , graphs can be modified using options that appear after a comma at the end of the command. For example:

```
. histogram muslperc, kdensity
```

In general. Stata is very user-friendly, and has excellent `help` facilities. One other tip: If you know the `thing` you want to do, but you don't know the Stata command to do it, type:

```
. findit thing
```

This is a generic / text search command for Stata help files, and can be a lifesaver.

Next time: Plots for more than one variable.