Graphing One Dimensional Numeric Data

Guy Lebanon

January 31, 2011

In this note we give an overview of basic techniques to graph one dimensional numeric data. It includes a description of strip plots, histograms, box-plots, and QQ-plots and explain how to graph them with R. We assume we are given a sample of $n$ points which we denote as

$$X = (X^{(1)},\ldots,X^{(n)}), \quad X^{(i)} \in \mathbb{R}.$$ 

The points are assumed to be drawn from a distribution $P$ that is in general unknown. An effective visualization of $X$ may, however, provide useful information concerning the unknown distribution $P$.

Strip Plots

A strip plot graphs one dimensional data by displaying each sample point as a point on a one dimensional axis. The main difficulties with this technique is (i) when several samples have identical values they will be printed on top of each other, and (ii) in the case of a large sample the graph will become overly crowded and less informative. The two ways to alleviate these difficulties are: (a) adding random noise to each sample point (this is called jittering), and (b) plotting each point in a transparent way so that many overlapping points are discernible from a single point or a few points.

```r
> # Strip plot of diamond prices for several cuts. The plot is not very informative due to overlapping values and overcrowding
> theme_set(theme_bw(base_size=8)); # for nice bw printing, decrease fonts
> png('fig1.png',height=200);
> print(qplot(cut, price, data=diamonds));
> invisible(dev.off());
```

In this and the following two figures the number of displayed points is very high making a pdf figure inconveniently large. In order to create a more reasonably sized pdf file we used png files which are based pixelized graphics rather than vector graphics.

```r
> # Strip plot of diamond prices for several cuts. A jitter (mostly horizontal) is applied in order to avoid overlapping values and overcrowding
> png('fig2.png',height=200);
> print(qplot(cut, price, data=diamonds, position='jitter'));
> invisible(dev.off());
```
The above plot shows that the full range of prices is realized for all diamond cuts but the probability of a high price increases substantially with the cut quality. Presumably the wide variability in the price for a single cut is due to the size and weight of the diamond.

Note how the semi-transparency reveals the large number of samples at the very bottom prices thus even for ideal cuts most diamonds value less than $2500. We can also see that there is little difference in the price distribution for the top three cut types.

Histograms

A histogram is constructed by binning the range of possible values into a number of distinct bins (usually equally sized) and counting how many observations fall in each bin. The histogram is graphed by displaying the bins with their height corresponding to the number of observations in the bins. A variation of that displays the height of the bins as the number of observations in each bin divided by the total number of observations (this makes the y axis interpretable as probability). The bin width is an important quantity – if set too large the histogram will be overly smooth and fail to capture fine details. If set too small the histogram will capture noise in the sampling process. This reflects the bias-variance tradeoff in statistics: assuming the histogram is a non-parametric approximation of the underlying density the estimation error is decomposed as a sum of bias and variance which go up and down respectively as the bin width increases. A general rule of thumb is that a small bin width is useful for small n and a large bin width is useful for large n.

We show below the histogram corresponding to the diamond price data above as well as three histograms of samples drawn from three normal distributions for several bin widths. We show how to display histograms in R first using the simpler `qplot()` function and then with the more complex `ggplot()` function and layering mechanism.
# histogram of ideal cut diamond prices using default bin width.

```
IC = diamonds[diamonds$cut == 'Ideal',];
print(qplot(price, geom='histogram',
            data=IC, main='price of ideal cut diamonds'))
```

![Histogram of Ideal Cut Diamond Prices](image)

Comparing this figure to the strip plot above we see that the histogram is more effective than strip plot for large datasets. Below are histograms of 200 samples from three different normal distributions.

```
# sample from three different normal distributions and create a dataframe
D = data.frame(samples = c(rnorm(200, 1, 1), rnorm(200, 0, 1), rnorm(200, 0, 2)));
# add a column describing the parameter value
D$parameter[1:200] = 'N(1,1)'; D$parameter[201:400] = 'N(0,1)'; D$parameter[401:600] = 'N(0,2)';
print(qplot(samples, facets=parameter~., geom='histogram', binwidth=1, data=D,
            main='Bin width=1'))  # plot histogram with binwidth=1
```

![Histograms of 200 Samples from Three Normal Distributions](image)

```
print(qplot(samples, facets=parameter~., geom='histogram', binwidth=0.5, data=D,
            main='Bin width=0.5'))  # decrease the bin width to 0.5
```

![Histograms with Bin Widths 1 and 0.5](image)
The same graph can be generated with the `ggplot()` function. Note how the first argument to `ggplot()` is the data frame, the second is the data passed as a `aes` object (aesthetics). Before plotting we need to add additional layers: histogram geometry, and facets (additional layers are added with a plus signs as needed).

```r
> ggplot(D, aes(x=samples)) + geom_histogram(aes(y=..density..), binwidth=0.5, fill='grey') + facet_grid(parameter~.)
```

This requires somewhat more work than the using the simpler `qplot()` function but provides a lot more flexibility in particular when overlaying multiple graphs (see example below) or generating novel graphics procedures. It is important to be familiar with both `qplot` for producing simple graphs quickly and the more flexible `ggplot`.

```r
> qplot(samples, facets=parameter~., geom='histogram', binwidth=0.01, data=D, + main='Bin width=0.01') # decrease the bin width to 0.01
```

```r
> ggplot(D, aes(x=samples)) + geom_histogram(binwidth=0.5) + facet_grid(parameter~.);
```

> # The notation ..density.. below scales the y axis to show density rather than counts.
> # This is needed for overlaying the density function on top of the histogram.
> p=ggplot(D, aes(x=samples)) + geom_histogram(aes(y=..density..), binwidth=0.5, fill=I('grey')) + facet_grid(parameter~.);

> # add the corresponding pdf as a new layer on top of the histogram
> x=seq(-6,6,length=200);
> R=data.frame(density=c(dnorm(x,1,1),dnorm(x,0,1),dnorm(x,0,2)));R$x=x;
> R$parameter[1:200] = 'N(1,1)';R$parameter[201:400] = 'N(0,1)';R$parameter[401:600] = 'N(0,2)';
> print(p + geom_line(aes(x=x,y=density), data=R, color='red'));
```
Box Plots
Recall the definition of the quantile function $Q_X$ as the inverse of the CDF $F_X$. When $F_X$ is one to one and onto, $Q_X(\alpha) = \beta$ if $F_X(\beta) = \alpha$. In the general case, however, $F_X$ is neither one-to-one nor onto. The general definitions (which reduces to the above if $F_X$ is one to one and onto) is

$$Q_X(r) = \inf\{x \in \mathbb{R} : r \leq F_X(x)\}.$$

Thus $Q_X(r)$ is the smallest $x$ value at which the cdf is still larger than $r$.

The most common quantile value is $Q_X(1/2)$ also known as the median - the value for which there is equal probability (1/2) of being smaller or larger than (for continuous $F_X$). The median is similar to the mean in that it captures the general location of the distribution. It differs from the mean that it is less sensitive to outliers. For example, the mean of the average income is significantly higher than the median average income due to the presence of a small number of outliers (billionaires) which pull the mean to the right without affecting the median. Similarly, the first and third quartiles $Q_X(1/4), Q_X(3/4)$ provide additional information through the interquartile range (IQR) defined as the interval $[Q_X(1/4), Q_X(3/4)]$ which holds 75% of the probability mass. When $r$ in $Q_X(r)$ is a fraction with two decimal points the quantile values are called percentiles. For example, the 75% percentile is the value $x$ at which 75% of the distribution is smaller than (it is also the third quartile $Q_X(3/4)$).

The above definitions are expressed for distributions. There are several definitions of the quantile function of a sample $X = (X^{(1)}, \ldots, X^{(n)})$ but in general they mirror the definition of $Q_X$ above (type help('quantile') in R for more information). For example, one way to define the quantile function of a sample is as the inverse of the empirical cumulative distribution function (ecdf) associated with the sample $X$ as

$$F_n(r) = \frac{1}{n} \sum_{i=1}^{n} I(X^{(i)} \leq r)$$

($I$ above is the indicator function which equals 1 if its arguments hold and 0 otherwise). $F_n(r)$ thus equals the number of samples less than or equal to $r$ divided by $n$. The quantiles of the sample $X$ are then defined as the inverse of the ecdf $F_n$ of $X$: $Q_X(r) = \inf\{x \in \mathbb{R} : r \leq F_n(x)\}$. The median, the first and third quartiles $Q_X(1/4), Q_X(3/4)$ and the minimum and maximum values together constitute the five number summary of the sample.

For example consider the sample $X = (1, 2, 10)$ The median is 2 is much smaller than the mean 13/3 due the skewness of the distribution to the right. We plot below the ecdf of that sample and overlay it with two vertical lines representing the median and the mean.

```r
> samples=c(1,2,10);
> median(samples); mean(samples);

[1] 2

[1] 4.333333
```
A box plot graphs the sample $X$ by drawing the inter-quartile range ($Q_X(1/4), Q_X(3/4)$) as a box with the median as a line intersecting it. Two whiskers extend on either side of the box capturing the smallest and largest sample points that are still within 1.5 IQR of the lower and upper edges of the box. Finally, any outliers that remain outside the box and whiskers area are plotted explicitly.

```r
# box plot of diamond prices for several cut types.
p <- ggplot(diamonds, aes(x=factor(cut), y=price)) + geom_boxplot()
print(p);
```

```r
# ECDF = data.frame(samples=c(0, samples), ECDF = ecdf_samples(c(0, samples)));
p <- ggplot(ECDF, aes(x=samples, y=ECDF)) + geom_step();
print(p+geom_vline(xintercept=median(samples), color=I('red'), size=1.5, lty=2) +
  geom_vline(xintercept=mean(samples), color=I('blue'), lty=3, size=1.5))
```

```r
# x=seq(0,1,length=200);y=quantile(samples,x,type=1);print(qplot(x,y,geom='line'),main='Quantile Function of the sample (1,2,10)');
```
Interestingly we see a slight downward trend between cut quality and price median and the lower quartile! However, there is a clear upward trend between the cut quality and the top whisker (except for ideal cut). Can you speculate on why this is the case?

**QQ-Plots**

The quantile function of a sample \( \mathcal{X} = (X^{(1)}, \ldots, X^{(n)}) \) is defined similarly: \( Q_X(\alpha) \) is the minimal \( x \) value for which \( \alpha \cdot 100 \) percent of the sample points \( \mathcal{X} \) are smaller than. The QQ-plot of a sample \( \mathcal{X} \) and a distribution \( X \) is a scatter plot showing the relationship between \( Q_X(\alpha) \) and \( Q_X(\alpha) \) for several \( \alpha \) values. Note that \( X \) may or may not be related to the sample \( \mathcal{X} \). In particular the sample may be drawn from a distribution other than \( X \).

1. If the distribution of the sample is similar to the distribution generating the sample the scatter plot will be approximately linear with slope 1 passing through the origin.
2. If the distribution of the sample is \( X + c \) (a shifted version of \( X \)) the scatter plot will resemble a line with slope 1. through the origin.
3. If the distribution of the sample is \( aX \) (a scaled copy of \( X \)) the scatter plot will resemble a line passing through the origin.
4. If the distribution of the sample is \( aX + b \) (scaled and translated copy of \( X \)) the scatter plot will resemble an arbitrary line.
5. If the distribution of the sample and \( X \) are unrelated the scatter plot may exhibit non-linear trend. That trend reveals qualitative differences between the distribution of the sample and \( X \) including which one is more heavy tailed or light tailed.

Below is a QQ-plot of a sample drawn from a normal distribution \( N(0,1) \) (with mean 0 and variance 1) and three different theoretical distributions (corresponding to three different \( X \)s).

```R
\textbf{print} (ggplot(D, aes(sample=samples))+stat_qq()+facet_grid(.~parameter));
```
Note how all three plots are linear (since the distribution of the sample is a scaled and translated copy of the three normal RVs. In particular: the left patterns is a slope 1 line that passed through the origin (the sample RV has the same distribution as $X$). The middle panel is a straight line through the origin with a slope different than one (since the distribution of the sample is $X/\sqrt{2} - X$ in this case is $N(0,2)$). The right panel shows a line with slope 1 that does not pass through the origin (the sample distribution is $X - 1 - X$ in this case is $N(0,1)$).

As a final example we show the QQ-plot of the $N(0,1)$ samples with $X$ being a t-distribution with 2 degrees of freedom.

The precise formula defining the pdf or cdf of the t-distribution is not important here. Note, however that the t-distribution like the normal is bell shaped symmetric distribution, but it has heavier tails. In fact the tails of the t-distribution are so heavy that the variance is not defined (at least in this case of 2 degrees of freedom). Note the non-linear “S” shape scatter plot confirms the fact that the t-distribution has heavier tails than the distribution of the sample.

```R
> x = seq(-6, 6, length=200);
> tdensity = dt(x, 2); # sample from the t-distribution with 2 dof
> # add to a filled area plot of the N(0, 1) density a layer showing
> # the t-distribution density as a line plot
> print(ggplot(R[201:400,], aes(x=x, y=density))+geom_area(fill='grey')+geom_line(aes(x=x, y=tdensity)))

> pm = list(df=2) # generate ggplot of sample from t-distribution and X=N(0, 1)
> print(ggplot(D[201:400,], aes(sample=samples))+stat_qq(distribution=qt, dparams=pm));
```