Practical statistics in R for SPR4104

Diana Santos

Practical training per week

1 Get to know probability distributions

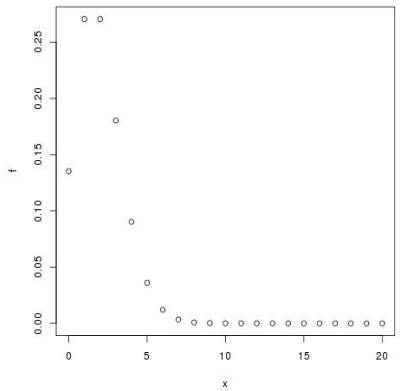
Probability distributions are special functions that R knows about. They come in two flavours: continuous, or discrete.

Let us start with the Poisson distribution, R nickname `pois`. Each distribution has a parameter (lambda) that corresponds to a different mathematical function.

Poisson is a discrete distribution function, so it only has values for 0, 1, 2, etc. Let us choose lambda=2, and draw the Poisson distribution function.

```
x<-0:20
f<-dpois(x,2)
plot(x,f)
```

First we created a vector called `x` with 21 values (0 until 20). Then we computed the values of a Poisson distribution function with lambda=2 and put them in a vector called `f`, and then we plotted the result.



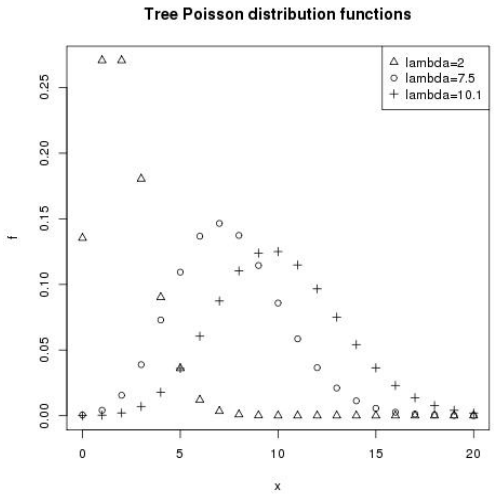
Here is what you should have got.

Try out with other lambda values, for example 7.5.

```
x<-0:20
f<-dpois(x,7.5)
plot(x,f)
```

Let us produce a drawing with several Poisson distribution functions in the same canvas. For that you cannot use two or more plots, you have to "open" one plot and then add lines or points.

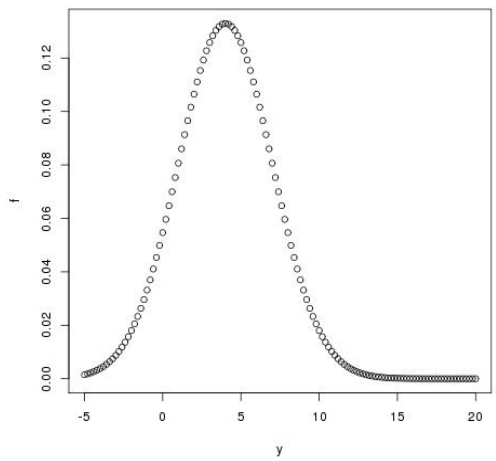
```
f<-dpois(x,2)
g<-dpois(x,7.5)
h<-dpois(x,10.1)
plot(x,f,type="n",main="Tree Poisson distribution functions",ylab="")
points(x,f,pch=2)
points(x,g)
points(x,h,pch=3)
legend("topright",c("lambda=2", "lambda=7.5", "lambda=10.1"),pch=c(2,1,3))
```



Let us now try the Gaussian (or normal), which is continuous. The Gaussian requires two numbers: μ (the mean) and σ (the standard deviation).

First we create a set of numbers with the R command `seq`, where you indicate the first, the last, and the difference for the next.

```
y<-seq(-5,20,by=.2)
f<-dnorm(y,mean=4,sd=3)
plot(y,f)
```



This is what you should get.

Now let us add different normal curves with the same mean but a larger standard deviation:

```
lines(y,dnorm(mean=4, sd=7)
```

For a continuous distribution, it makes mor sense to print a (continuous) line than just points. Try the following commands:

```
y<-seq(-40,80,by=.2)
f<-dnorm(y,mean=4,sd=3)
g<-dnorm(y,mean=4,sd=5)
h<-dnorm(y,mean=4,sd=7.5)
plot(y,f,type="n",main="Three Gaussians with different standard deviations",
+ ylab="")
lines(y,f)
lines(y,g, lty=2)
lines(y,h, lty=3)
legend("topright",c("sd=3", "sd=5", "sd=7.5"),lty=c(1,2,3))
```

The way to compute the probability of $x <= X$ is to use the corresponding cumulative distribution. For the normal, the `pnorm` command. To compute the probability that $X <= 5$, for example

```
pnorm(5,mean=4,sd=2)
[1] 0.6914625
```

If we do the same for the other distributions, we get respectively 0.5792597 and 0.5530351: lower values because there is less area that is covered by the bell shape.

I have added the line X=5 to the graph in order for one to see the area to the left.

```
lines(x=c(5,5),y=c(0,0.135))
```

The command lines gives two points – which define a line. It gives the point (5, 0) and the point (5,0.135).

If one wanted to compute the probability of something being between 1 and 6, the solution would be the probability of something less that six, minus the probability of soemthing less than 1. For a normal with mean 4 and standard deviation 7.5, this would be

```
pnorm(6,mean=4,sd=7.5) - pnorm(1,mean=4,sd=7.5)
[1] 0.2605588
```

2 Visualizing your sample

When one has just numbers, it is important to be able to characterize them by what is called "statistics": numbers computed from your sample.

Before that, learn to know your samples:

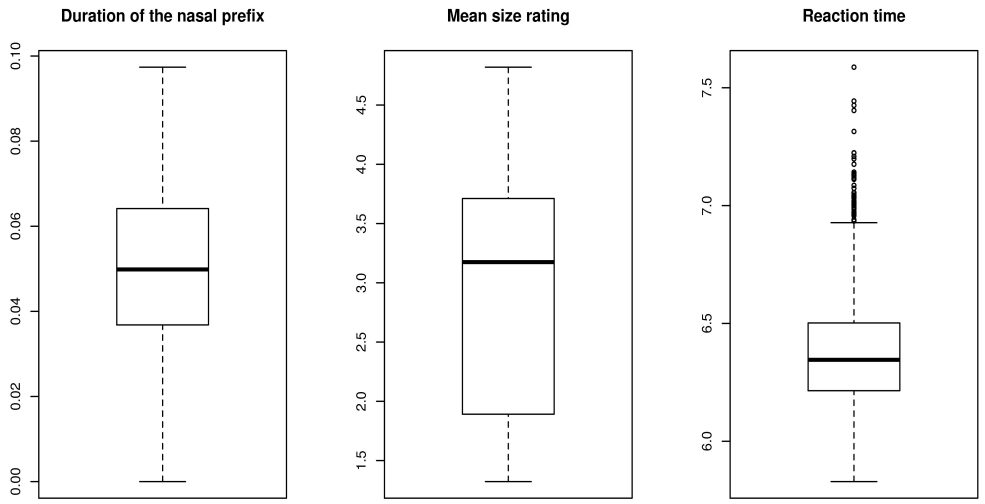
```
library(languageR)
durationsOnt$DurationPrefixNasal
ratings$meanSizeRating
lexdec$RT
```

You should also see all values, if continuous, as a boxplot, see Figure 1 or as a histogram, in Figure 2.

If your numbers correspond to counts (are discrete, not continuous), you should display them as a barplot.

First, if you have categorical data, and want to count it from a dataframe, you can simply tabulate it with the command `table()`.

```
table(auxiliaries$Aux)
table(verbs$AnimacyOfRec)
table(ratings$Class)
table(oldFrenchMeta$Genre)
table(spanishMeta$Author)
```



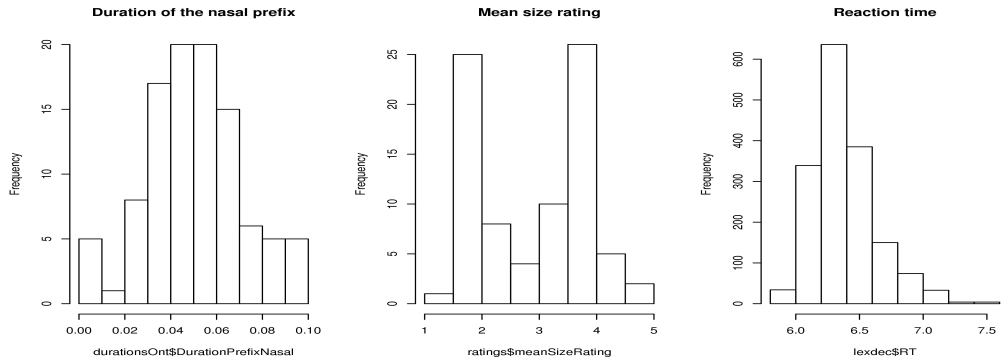
Figur 1: Boxplots

Or crosstabulate it with the command `xtabs`, that produces contingency tables:

```
xtabs(~ AnimacyOfRec + RealizationOfRec, data=verbs)
AnimReal<-xtabs(~ AnimacyOfRec + RealizationOfRec, data=verbs)
barplot(xtabs(~ AnimacyOfRec + RealizationOfRec, data=verbs))
```

You can add more information, as well as change the form of display. Try out

```
barplot(xtabs(~ AnimacyOfRec + RealizationOfRec, data=verbs,add=T))
```



Figur 2: Histograms

```
barplot(AnimReal,legend=rownames(AnimReal))
```

Finally, if you have proportions, you can also use a pie chart, or a mosaicplot.

```
pie(table(verbs$AnimacyOfRec))
```

But note that pie charts are considered very unreliable as visualization tools. See for example the information on `help(pie)`.

3 Computing statistics about your sample

After you visualized and have an idea of the data, you should know how to describe it.

For the samples above compute the mean and the median – measures of central tendency, and the variance and the range – measures of spread. For example,

```
mean(durationsOnt$DurationPrefixNasal)
sd(durationsOnt$DurationPrefixNasal)
var(durationsOnt$DurationPrefixNasal)
median(durationsOnt$DurationPrefixNasal)
max(durationsOnt$DurationPrefixNasal)
range(durationsOnt$DurationPrefixNasal)
```

See also what the handy R command `summary()` provides.

4 Correlation

Correlation is a measure (independent of the units and the values) of whether two measurements go "in the same direction", "in the opposite direction", or have no relationship whatsoever.

If you have two different measurements of something, you can display them as scatterplot, with one value in the X dimension, and the other in the Y dimension, as the two arguments of `plot`. The first is on the X-axis, the second on the Y-axis.

Try out some scatterplots to get a feeling of the way of representing. By the way, there are other formats (in addition to the tab-separated one) that R also accepts, such as `read.delim()`. The `f1data` dataframe has the average F1 frequency for males and for females for several vowels in different languages.

```
f1data <- read.delim("http://folk.uio.no/dssantos/cursoR/F1_data_KJ.txt")
attach(f1data)
plot(female,male)
```

You will have a plot with the F1 for females in the X-axis, and the one for males in the Y-axis.

You can compute the correlation between the two properties (female average F1 and male average F1), but together with this correlation you should also test how probable is the null hypothesis that the correlation is zero.

```
cor.test(female,male)
```

This is what you get

```
Pearson's product-moment correlation

data:  female and male
t = 17.676, df = 17, p-value = 2.23e-12
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.9318264 0.9901071
sample estimates:
      cor
0.9738566
```

A very high correlation, and highly significant.

Try out the opposite way:

```
cor.test(male,female)
```

Now let us repeat this for other datasets: Easter indicates the number of mentions to Easter by week number in the Brazilian and Portuguese newspapers.

```
easter <- read.table("http://folk.uio.no/dssantos/cursoR/PascoaCHAVEvar.txt")
colnames(easter)<-c("week","PT","BR")
attach(easter)
plot(PT,BR)
cor.test(PT,BR)
```

Although the correlation is also very high and significant, you should note that it is based in very different data.

Try also a pedagogical dataset with the length in letters of the stimuli words and the time in miliseconds that a language learner takes to decide (from Gries's book).

```
react<-read.table("http://folk.uio.no/dssantos/cursoR/Gries03-reactiontimes.txt"
attach(react)
plot(LENGTH,MS_LEARNER)
cor.test(LENGTH,MS_LEARNER)
```

A more realistic case comes from Baayen and his `ratings` dataset, where the two columns can be correlated:

```
library(languageR)
plot(ratings$meanWeightRating, ratings$meanSizeRating)
attach(ratings)
cor.test(meanSizeRating, meanWeightRating)
```

It is however possible to have strong negative correlations, or no correlation at all...

5 Linear models

One model is something that can "replace" our data and make things simpler for us. If the values are highly correlated, I may decide to obtain one from the other.

Let us model, in the previous cases, one variable by the other, using `lm()` in R. The first and easiest case is to obtain the regression line, that is, a line that minimizes the "residuals", namely the difference between the modeled value and the real value. That line, a straight line, can be fully characterized by the intercept and the slope. But since we are doing statistics, we'd better learn at once how reliable are these two numbers.

```
summary(lm(meanSizeRating ~ meanWeightRating))

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      0.526998   0.010632   49.57   <2e-16 ***
meanWeightRating 0.926474   0.003837  241.45   <2e-16 ***
```

What we get is a model that describes meanSizeRating as being approximable by a straight line = .53 + .93 * meanWeightRating.

And that we can be pretty sure that those numbers are not zero.

Nowhere are we saying that one is caused by the other, or that it explains the other. Just that we can more or less reliably infer one from the other. The other way around would be just as fine:

```
summary(lm(meanWeightRating ~ meanSizeRating ))
```

In fact, the first thing you should check from your model is the (Adjusted) R-square (which in this case is just the correlation cofficient squared), but which in general represents the percentage of the variance that the model explains.

```
cor(meanWeightRating,meanSizeRating)^2
```

It is practical to know how to draw lines in R. Not only the models (linear models created with lm), but any line that one may want to draw. For this one has the command `abline()` (check it with `help()`).

Let us thus add the regression line to the data with this abline command (y=a+bx):

```
plot(meanSizeRating, meanWeightRating)
abline(lm(meanWeightRating ~ meanSizeRating))
```

One can also add other lines at will, like

```
abline(a=2,b=0)
abline(v=4)
abline(a=1, b=1.077900 )
```

You should now be able to draw all lines that model the previous datasets.

6 Linear models with factors

The simplest generalization of linear models is one-way ANOVA, when you may have different groups according to levesl of your factors, and you want to model whether the groups are diferent or the null hypothesis – that there are no differences between groups – holds.

If you have only two groups, you can use the t-test. If you have more, you have to use `lm()`.

Note that a factor is a categorical variable, with several levels. R usually understands when something is a factor or a number, but one can supersede R's understanding by explicitly indicating that a particular column represents a factor. One can also change the order of the values (by default it is alphabetical).

```
cd<-read.table("http://folk.uio.no/dssantos/cursoR/condiv.txt",header=TRUE)
summary(cd)
cd$decade<-factor(cd$decade,levels=c("50","70","2000"))
summary(cd)
cd$variety<-factor(cd$variety,levels=c("PT","BR"))
summary(cd)
```

Let us then use data from Gries about the creation of new lexical items, and visualize it:

```
dice<-read.table("http://folk.uio.no/dssantos/cursoR/wordformationGries.txt",
header=TRUE)
boxplot(dice$DICE~dice$PROCESS)
```

It appears that each group is quite distinct, but we have to do a statistical test. The simplest is to use the `aov()` command from R

```
attach(dice)
summary(aov(DICE~PROCESS))
```

It says that it is easy to reject the null hypothesis that PROCESS is not relevant for describing the data.

What we usually want, is to know a little more than that, and then the `lm()` comes in handy.

```
summary(lm(DICE~PROCESS))
```

Now you'll see one line per each value of the factor (minus one). And in this case all of them are significantly different from zero.

```
Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)       0.138130   0.002915   47.38   <2e-16 ***
PROCESSComplClip -0.068192   0.004123  -16.54   <2e-16 ***
PROCESSCompound  -0.104660   0.004123  -25.39   <2e-16 ***
```

(to be continued...)