Lesson 05 - Data Modeling

Jul 11, 2016

In this lesson, I will present a basic overview of: (1) probability distributions in R, and (2) modeling data in R using built-in statistical functions. It is helpful to think about what type of statistical distribution best captures the pattern of your data, in order to do hypothesis tests, estimation, prediction, and/or any other statistical inference.

We will use the code that is provided in this script.

Probability Distributions in R

Statistics in R

There are a lot of statistical functions available in base R. To see a list of those which are in the package `stats`, enter the following in your console. This is a very long list, so we will just focus on a select few.

```
library(help = "stats")
```

Families

Each probability distribution in R comes in a family of *four* related functions, with the following prefixes:

- `d*` for *density*
- `p*` for *probability*
- `q*` for *quantile*, and
- `r*` for *random*

such that the `norm` or normal distribution in R is accessed using `dnorm()`, `pnorm()`, `qnorm()`, or `rnorm()`, depending on what you are trying to do. [1] Each of these four types take different inputs and give different outputs.

To simulate random data, will mostly use the `r*` prefix. This should be familiar to you. In Lesson 03 and Lesson 04, we encountered the random normal, `rnorm()` distribution, and in Lesson 04, the random (continuous) uniform `runif()` distribution. A few other probability distributions you may be familiar with from an introductory probability/statistics course are: (1) the Poisson, `rpoiss()` distribution, (2) the binomial, `rbinom()`, distribution, and the exponential, `rexp()`, distribution. In contrast with the normal, uniform, and exponential distributions, which are *continuous*, the Poisson and Binomial distributions are *discrete*, i.e. they deal with count data.

Poisson

In the poisson distribution, we can describe the expected number of times that an event will occur within a given time period (or volume, or genome length, etc.) Or more precisely, we describe the expected **rate of occurrence**.

One assumption we make when modeling data with a Poisson distribution is that the number of occurrences within each given interval is independently drawn from an identical underlying distribution. That means that if we know the numer of events for the first interval, this does not affect the expected number of events for the next interval, and so on.

In other words, the **event rate** is considered a **random variable** that is distributed independently and identically from the same true underlying Poisson process, with a mean of λ, indicating the average or expected number of events per interval. [2]

Example: Prediction

The average number of times the word "near" is encountered in a book, "The Farthings", is 1.5 times per page ($\lambda = 1.5$). We are told that the rate of occurrence of "near", which we are calling ***X***, follows a Poisson distribution. What is the probability that we will find exactly 2 instances of “near” in the next 1 page?

For those who like formal notation, we have the following:

$$X \sim \text{Pois}(\lambda)$$

$$Pr(X == 2 \mid \lambda = 1.5) = ?$$

Solution:

We can use the `d*` function for poisson, `dpois`, which gives the expected probability for any ***X***, given λ:

```
dpois(x=2, lambda=1.5)
```

This is not a very sophisticated form of prediction. However, this enables us to retrieve a probability of an event, after making some assumptions about the underlying distribution, `poisson`, and parameter(s), `lambda`.

Data Set: Bad Words in Select Movies

Read in the data.

```
download.file(
        "http://mauriziopaul.github.io/intro-to-R/data/mov.csv",
        destfile="mov.csv")
mov <- read.csv("mov.csv")
```

In this data set, there are 4 columns: `movie`, `type`, `word`, and `minutes_in`.

The column `movie` has seven levels corresponding to 7 movies.

The column `type` has two levels corresponding to the type of event being recorded (`word` or `death`).

The column `word` has 61 levels corresponding to 60 unique *censored* words (not appropriate for class, thus bleeped out), with `word_1` representing a placeholder for death (I could have reassigned these to have `NA` values). [3]

The final column, `minutes_in`, is a numeric/float column, which indicates how many minutes into the movie the event occurs, with values extending between 0.4 to 160.4.

Example: Prediction, revisited

In this case, we want to know the average per-minute rate for a specific event, `word`. That will be the λ. We then can use this to ask: If the film continued, what is the likelihood that we would encounter 5 more `word`'s in the next minute of film? For the moment, we assume that the counts per minute follow a Poisson distribution.

This will take a little bit of pre-processing to get through.

Let's start with `movie=="Reservoir Dogs"`, and focus on the `type=="word"` events. We will convert the fractional minutes to whole minute values using the `floor` command.

```
m.sub.word <- subset(mov, movie=="Reservoir Dogs" & type=="word")
m.sub.word$minute <- floor(m.sub.word$minutes_in)
```

This data set does not specify the total number of minutes in the movie, but let's suppose that we know from elsewhere that the movie has 99 minutes in it (from minute 0 through minute 98).

How many `word` events are there?

```
nrow(m.sub.word)
```

We can make a table of counts per minute across all 99 minutes, including all the minutes with zero counts. [4]

```
datmat <- table(rep(0:98))

for(i in 0:98){
        thiscount <- nrow(subset(m.sub.word, minute==i))
        datmat[names(datmat)==i] <- thiscount
}
```

What is the average count per minute, and the range of counts per minute? Let's also plot these in a bar plot.

```
mean(datmat)
range(datmat)
barplot(datmat)
```

How does the following plot differ from the last?

```
barplot(table(m.sub.word$minute))
```

What does the distribution of rates look like? How does it compare with a random sample from a Poisson distribution that has the same `n` and `lambda` as our data?

```
hist(datmat, xlim=c(0,16), breaks=15, xlab="rate per min")
hist(rpois(n=421, lambda=4.252525), xlim=c(0,16), breaks=10, xlab="rate
```

If we take that to be our λ, and we want to know what is the probability that `x=5`, for a Poisson with $\lambda = 4.252525$, we have:

```
dpois(x=5, lambda=4.252525)
```

or a `16.49%` chance of happening.

The Poisson distribution may or may not be a good fit for this specific data set. However, the Poisson distribution, and related distributions (such as the negative binomial), are useful for modeling next-gen sequencing data. Consider an RNA-seq experiment, and imagine that the minutes in the movie are instead nucleotides across a chromosome / transcriptome, and that the number of `word` occurrences are the read counts (depth) at each nucleotide site. This is the foundation for our understanding of the distribution of short read counts in an RNA-seq experiment.

Additional details about other probability distributions are available at this link: http://www.r-tutor.com/elementary-statistics/probability-distributions

Other Distributions

Binomial

```
bin <- rbinom(n=100,size=1,prob=0.5)
plot(bin, pch=16, ylim=c(-1,2))
hist(bin, breaks=2)

bin <- rbinom(n=100,size=1,prob=0.2)
plot(bin, pch=16, ylim=c(-1,2))
hist(bin, breaks=2)
```

Try changing the number of samples (`n`), or the probability (i.e., coin flip "bias") and plotting the result. How close does your sample, `bin`, come to the probability that you set?

```
mean(bin)
```

Exponential

```
exp <- rexp(n=100, rate=5)
plot(exp, pch=16)
hist(exp)
```

And so on. See some additional details here
http://www.statmethods.net/advgraphs/probability.html, and here
http://www.cyclismo.org/tutorial/R/probability.html

Modeling Data in R

The point of this section is to show you the relevant functions for simple data modeling, not to teach you the theory behind the models.

The Student's *t*-test

You will likely be familiar with the Student's *t*-test. We will focus on the **two-sample** *t*-tests, using both the **paired** and **unpaired** versions.

Unpaired

First, let's subset the data on two of the species, and plot.

```
iris.sub <- droplevels(subset(iris,
        Species %in% c("setosa", "versicolor")))
plot(Sepal.Length ~ Species, data = iris.sub)
beeswarm(Sepal.Length ~ Species, data=iris.sub, add=TRUE, pch=16)
```

The two methods below are essentially equivalent.

```
with(iris.sub, t.test(Sepal.Length[Species == "setosa"],
        Sepal.Length[Species == "versicolor"]))
t.test(Sepal.Length ~ Species, data = iris.sub)
```

Do you understand what each line of output means?

Conveniently, the second method, using a formula, is both simpler and consistent with the plot formula we used previously.

Paired

Suppose we had before/after data, and we wanted to see if there was a difference in the `before` and `after` values, accounting for the pairing of the data.

```
before = c(12.9, 13.5, 12.8, 15.6, 17.2, 19.2, 12.6, 15.3, 14.4, 11.3)
after = c(12.0, 12.2, 11.2, 13.0, 15.0, 15.8, 12.2, 13.4, 12.9, 11.0)
t.test(before, after, paired=TRUE)
```

Linear Regression

Suppose we wanted to know if there was a significant effect of `Species` on `Sepal.Length`, but there are more than two groups of `Species` in our data set. We would start by building a linear model of our data.

When we generate a linear model, we assume that our values are distributed, in this case, from a distribution where the residual errors are distributed normally.

$$y = \beta_0 + \beta_1 x + \varepsilon$$

$$\varepsilon \sim N(0, \sigma^2)$$

Often, the fit for a simple linear model uses a least squares approach, which draws a line through the data (in 2 or more dimensions), minimizing the residual difference between the points and the line by some metric. The estimate for the `beta`'s is then the value of the slope of the line of best fit. The residuals, `varepsilon`, are assumed to be drawn from a normal distribution, with a standard deviation of σ^2.

Let's plot the data:

```
plot(Sepal.Length ~ Species, data = iris)
beeswarm(Sepal.Length ~ Species, data=iris, add=TRUE, pch=16)
```

Let's build a very simple linear model of our data.

```
fit <- lm(Sepal.Length ~ Species, data=iris)
fit
```

The function `lm()`, by default, includes an intercept, β_0. Perhaps we don't want an intercept in our linear model. There are circumstances when this might be the case. We can then formulate our expression like this:

```
fit0 <- lm(Sepal.Length ~ Species - 1, data=iris)
fit0
```

ANOVA

```
anova(fit)
summary(aov(fit))

anova(fit0)
```

Pairwise Differences

The ANOVA only tells us if levels of a particular factor contribute significantly to the phenotype variation. It does not tell us which levels of that factor are significantly different from which other levels. One way to compute all pairwise differences between levels of a factor is to use `TukeyHSD()`.

```
TukeyHSD(aov(fit))
```

Example: Warpbreaks

Back to our exciting warp breaks per loom data set. Let's look at modeling the effects of two factors.

```
data(warpbreaks)
beeswarm(breaks ~ wool * tension, data=warpbreaks, pch=16)
boxplot(breaks ~ wool * tension, data=warpbreaks, add=TRUE)
```

Let's fit the data with a linear model.

```
wbfit <- lm(breaks ~ wool * tension, data=warpbreaks)
```

What if we build up our model, piece-by-piece?

```
wbfit0 <- lm(breaks ~ wool, data=warpbreaks)
wbfit1 <- lm(breaks ~ wool + tension, data=warpbreaks)
wbfit2 <- lm(breaks ~ wool * tension, data=warpbreaks)
```

What is the difference between `wbfit1` and `wbfit2`?

What was another option for the formula in wbfit0?

We can use `anova()` to compare any two **nested** models.

```
anova(wbfit1, wbfit0)
anova(wbfit2, wbfit1)
```

This seems to indicate that tension is a significant factor (p=0.001378), and the interaction between tension and wool is also significant (p=0.02104).

The order of comparisons does matter, and it can change the question you are asking of the data.

```
wbfit0a <- lm(breaks ~ tension, data=warpbreaks)
wbfit1 <- lm(breaks ~ wool + tension, data=warpbreaks)
anova(wbfit1, wbfit0a)
wbfit <- lm(breaks ~ 1, data=warpbreaks)
```

What is the following expression equivalent to (what does the `.` represent)?

```
wbfit <- lm(breaks ~ ., data=warpbreaks)
```

Quantile-Quantile plots

What do these tell us?

```
qqnorm(iris$Sepal.Length)
qqline(iris$Sepal.Length)

qqnorm(iris$Sepal.Width)
qqline(iris$Sepal.Width)
```

```
qqnorm(iris$Petal.Length)
qqline(iris$Petal.Length)

qqnorm(iris$Petal.Width)
qqline(iris$Petal.Width)
```

How else might you compare the distribution of the iris data with a *true* random normal distribution?

Homework

1. Using `barplot`, plot the number of `words` that occur in Django Unchained, which is 165 minutes long.
2. Run an `anova` test on the `Petal.Length ~ Species` in the `iris` data set.
3. Let's assume that the average number of goals scored in a World Cup football (soccer) match is 2.8, and the score per match follows a Poisson distribution. What is the probability that there will be 4 goals scored in the next World Cup football match?

References

Some of the information in this lesson was obtained from the following resources:

The R Guide, by W. J. Owen, 2010

Software for Data Analysis: Programming with R, by John Chambers, 2008, "Section 6.9: Fitting Statistical Models"

1. The function `norm()` is not related to the four normal distribution functions. It is used for an unrelated matrix algebra calculation. ↩
2. This kind of **random variable** is something different from the **variable** that we talk about in R, which just means an object that we assign a value to. ↩
3. This data set was cleaned up from one that was used in a FiveThirtyEight.com article, which you can try to find yourself after class. ↩
4. There is probably a better way to do this, but this code is fairly easy to follow. ↩