# PROBABILITY

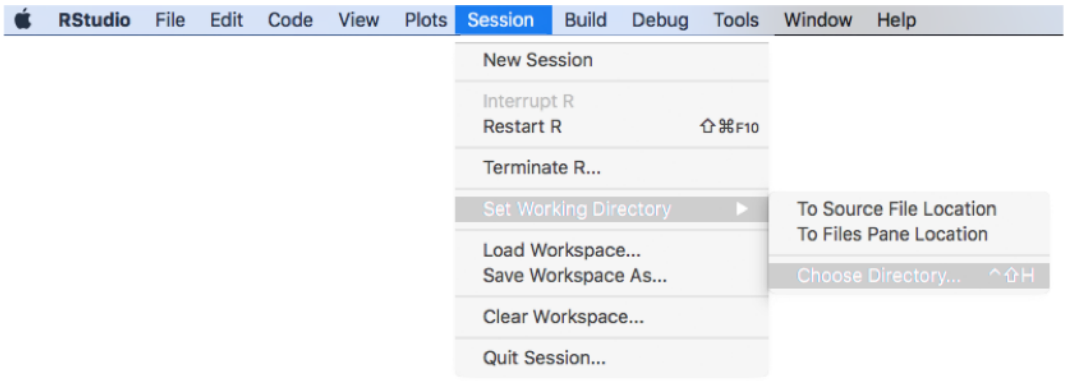
This handout is one of a series that accompanies *An Adventure in Statistics: The Reality Enigma* by me, Andy Field. These handouts are offered for free (although I hope you will buy the book).[1]

## Overview

In this handout we will look at how to do the procedures explained in Chapter 7 using R an open-source free statistics software. If you are not familiar with **R** there are many good websites and books that will get you started; for example, if you like *An Adventure In Statistics* you might consider looking at my book *Discovering Statistics Using R*.

### Some basic things to remember

- RStudio: I assume that you're working with RStudio because most sane people use this software instead of the native **R** interface. You should download and install both R and RStudio. A few minutes on Google will find you introductions to R Studio in the event that I don't write one, but these handouts don't particularly rely on R Studio except in setting the working directory (see below).
- Dataframe: A dataframe is a collection of columns and rows of data, a bit like a spreadsheet (but less pretty)
- Variables: variables in dataframes are referenced using the $ symbol, so *catData$fishesEaten* would refer to the variable called *fishesEaten* in the dataframe called *catData*
- Case sensitivity: **R** is case sensitive so it will think that the variable *fishesEaten* is completely different to the variable *fisheseaten*. If you get errors make sure you check for capitals or lower case letters where they shouldn't be.
- Working directory: you should place the data files that you need to use for this handout in a folder, then set it as the working directory by navigating to that folder when you execute the command accessed through the **Session>Set Working Directory>Choose Directory ...** menu




[1] This document is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, basically you can use it for teaching and non-profit activities but not meddle with it.

## Packages used in this chapter

We install packages using the *install.package("package name")* function, so if you haven't already got them installed execute the following commands:

```
install.packages("dplyr")
install.packages("prob")
```

This installs the package **prob**, which has various useful functions for doing probability stuff, and **dplyr** which we're only going to use for convenience to show a random sample of one data set. Once installed we need to initialise these packages in the current **R** session by executing the *library(packagename)* function as follows.

```
library(dplyr)
library(prob)
```

### The data

In the book, Zach goes to the JIG:SAW complex with Milton. However, to get there he has to cross the bridge of death. Along the way Milton explains some probability using the example of sampling names from the contacts in Zach's Proteus, and then Zach has to use what he has learned (and what he already knows about *z*-scores) to solve various puzzles along the bridge.

Mostly we will input the data manually. However, the final puzzle is a game called Deathscotch, and the data for this example is in a csv file on the companion website for the book. Execute the command below, which uses the *file.choose()* function to open a dialogue box so that you can navigate to the file that you want to open, which in this case will be *AiS Ch 07 Deathscotch.csv*. The rest of the command tells **R** to import this file into a dataframe called **deathscotch**.

```
deathscotch<-read.csv(file.choose())
```

Let's look at 20 random rows selected from the data to give us an idea of what it looks like. In **R** you can access an element in a dataframe by executing *dataName[row, column]* in which *row* and *column* are numbers (or names) referring to the row/column that you want. For example, *dataName[1:10, 2]* will return rows 1 to 10 of column 2. If *row* or *column* is left blank then all rows or columns are returned. For example, *dataName[1:10, ]* returns rows 1 to 10 of every column. The command below uses the *sample_n(dataName, x)* function from the **dplyr** package, which returns *x* randomly sampled rows from the specified dataframe.[2] In this case we've asked for 20 rows from the deathscotch dataframe.[3]

```
sample_n(deathscotch, 20)
```

```
##    Stone   Vanish
## 51 Heart  Remains
## 26 Skull  Remains
## 23 Skull  Remains
## 52 Heart  Remains
## 39 Heart  Remains
## 67 Heart Vanishes
## 34 Skull Vanishes
## 9  Skull  Remains
```

[2] If you don't want to use the **dplyr** package you can achieve the same goal with: *deathscotch[sample(nrow(deathscotch), 20), ]*

[3] In case this isn't obvious you will most likely get different output to this handout because the 20 rows are randomly selected.

```
## 4  Skull  Remains
## 57 Heart  Remains
## 12 Skull  Remains
## 40 Heart  Remains
## 60 Heart  Remains
## 75 Heart Vanishes
## 21 Skull  Remains
## 70 Heart Vanishes
## 32 Skull Vanishes
## 72 Heart Vanishes
## 2  Skull  Remains
## 8  Skull  Remains
```

Looking at the 20 random cases you will see that the dataframe contains 2 variables, which will make a lot more sense if you have read the book:

- **Stone**: identifies whether the stone was shaped like a heart or skull.
- **Vanish**: identifies whether the stone remained or vanished when stepped upon.

# Probability

## Classical probability

In the book, Milton explains classical probability to Zach suing the example of sampling contacts from his Proteus. It turns out that Zach has 5 contacts: his band mates Joel (guitar), Nick (drums) and Jessika (bass), Celia (who he met a JIG:SAW recruitment event) and Lemmy (because at the time of writing I assumed he would live forever and, therefore, be part of this futuristic world[4]). Let's create a variable that contains these contacts, and another one that records their sex.

```
name<-c("Nick Franklin", "Celia Genial-Thing", "Joel", "Jessika Kenny", "Lemmy")
sex<-c("Male", "Female", "Male", "Female", "Male")

data.frame(name, sex)
```

```
##                 name    sex
## 1      Nick Franklin   Male
## 2 Celia Genial-Thing Female
## 3               Joel   Male
## 4      Jessika Kenny Female
## 5             Lemmy   Male
```

The first line creates a variable called **name** containing the five names of the contacts in Zach's Proteas and the second creates a variable called **sex** that records the sex of each contact (in the same order). The third line uses the *data.frame()* function to display the two variables in columns so that we can see the correspondence between the names and the sex of the person.

The first challenge Milton sets Zach is to calculate the probability that if they randomly sampled 1 contact, they would sample Joel. There are several ways to do this in **R**, and the way I'm going to explain is a bit long-winded, but it has the advantage that we can extent the process to more complex situations. We're going to use some functions from the **prob** package.

First, we are going to to use the **urnsamples()** function to generate the sample space for an experiment in which we sample 1 name:

[4] I'm sad to say that in true rock 'n' roll style he proved me wrong just 2 months after I submitted the manuscript.

```
nameSamples1<-urnsamples(name, 1)
nameSamples1
```

```
##                 out
## 1      Nick Franklin
## 2 Celia Genial-Thing
## 3               Joel
## 4      Jessika Kenny
## 5              Lemmy
```

The first line creates the sample space in an object I've called *nameSamples1* by telling the function *urnsamples()* to sample from the variable called **name** (which we created above) and to take 1 item in each experiment and not replace it. The second line shows the contents of *nameSamples1* and you'll notice it's the same as the original variable **name**.[5] The next step is to calculate the probability of each outcome using the *probspace()* function:

```
nameProbSpace1<-probspace(nameSamples1)
nameProbSpace1
```

```
##                 out probs
## 1      Nick Franklin   0.2
## 2 Celia Genial-Thing   0.2
## 3               Joel   0.2
## 4      Jessika Kenny   0.2
## 5              Lemmy   0.2
```

The first line creates the probability space in an object I've called *nameProbSpace1* by telling the function *probspace()* to use the sample space that we previously created called *nameSamples1*. The second line shows the contents of *nameProbSpace1* and you'll notice it's a dataframe with two variables: **out** lists the possible outcomes and **probs** contains the associated probability of each outcome. We can see from this already that the probability of selecting Joel is 0.2 as shown in the book. In fact, the probability of sampling any one person (not just Joel) is 0.2.

We could stop here because we have our answer, but in the interests of illustrating a process that applies more generally, we can now use the *Prob()* function to compute the particular probability of a specific event. We input the object containing the probability space (*nameProbSpace1*) into the function, and then specify a condition related to the question we want to answer. We have set the condition to be *out == "Joel"*, which tells the function that I want the probability that the variable *out* within the object *nameProbSpace1* is equal to (not the double equals sign) the word "Joel" (note the speech-marks). The result will again be 0.2.

```
Prob(nameProbSpace1, out == "Joel")
```

```
## [1] 0.2
```

The next task Milton sets is to discover the probability of selecting a woman. We can follow the same process.

```
sexSamples1<-urnsamples(sex, 1)
sexProbSpace1<-probspace(sexSamples1)
Prob(sexProbSpace1, out == "Female")
```

```
## [1] 0.4
```

The process is the same as before, line 1 creates the sample space based on sampling 1 item from the variable called **sex** that we created earlier on. Line 2 creates the probability space for each outcome in

[5] Like I said, this method is long winded but, as we will see, it generalizes to more complex examples.

the sample space.[6] The final line then returns the probability of the outcome being equal to female (*out == "Female"*) using the probability space created for the variable **sex**. The result, as in the book, is 0.4.

Milton then ramps up the difficulty by asking Zach to calculate the probability of selecting Joel if they samples pairs of names rather than a single name. This is where my long-winded approach comes in handy.

```
nameSamples<-urnsamples(name, 2)
nameSamples
```

```
##                   X1                X2
## 1       Nick Franklin Celia Genial-Thing
## 2       Nick Franklin              Joel
## 3       Nick Franklin     Jessika Kenny
## 4       Nick Franklin             Lemmy
## 5  Celia Genial-Thing              Joel
## 6  Celia Genial-Thing     Jessika Kenny
## 7  Celia Genial-Thing             Lemmy
## 8               Joel     Jessika Kenny
## 9               Joel             Lemmy
## 10      Jessika Kenny             Lemmy
```

The first line we set up the sample space but note that now we are sampling 2 items from the variable **name** rather than 1. The resulting sample space shows 10 possible pairings (as explained in the book) and contains two variables labelled **X1** (the name of the first person in the pair) and **X2** (the name of the second person in the pair). Next we compute the probability space in the same way as before.

```
nameProbSpace<-probspace(nameSamples)
nameProbSpace
```

```
##                   X1                X2 probs
## 1       Nick Franklin Celia Genial-Thing   0.1
## 2       Nick Franklin              Joel   0.1
## 3       Nick Franklin     Jessika Kenny   0.1
## 4       Nick Franklin             Lemmy   0.1
## 5  Celia Genial-Thing              Joel   0.1
## 6  Celia Genial-Thing     Jessika Kenny   0.1
## 7  Celia Genial-Thing             Lemmy   0.1
## 8               Joel     Jessika Kenny   0.1
## 9               Joel             Lemmy   0.1
## 10      Jessika Kenny             Lemmy   0.1
```

The first line creates the probability space in an object I've called *nameProbSpace* by telling the function *probspace()* to use the sample space that we previously created called *nameSamples*. The second line shows the contents of *nameProbSpace* and you'll notice it's a dataframe with three variables: the **X1** and **X2** from the sample space, and **probs** which contains the associated probability of each outcome. To work out the probability that a pair contains Joel, we need to set a condition that looks at whether Joel is either the first person in the pair (**X1**) or the second (**X2**). In **R** we use the | symbol to denote 'OR'.

```
Prob(nameProbSpace, X1 == "Joel" | X2 == "Joel")
```

```
## [1] 0.4
```

This command uses the *Prob()* function to return the probability of the variable **X1** OR **X2** being equal to the word Joel (*X1 == "Joel" | X2 == "Joel"*) within the object called *nameProbSpace*. The result is 0.4, as in the book. As you get more confident with **R** you might want to skip some of these steps and incorporate them into a single command. In this example we could get to the same result by executing a command that skips the process of creating objects along the way:

[6] If you want to you can look at the sample space and probability space by executing their names.

```
Prob(probspace(urnsamples(name, 2)), X1 == "Joel" | X2 == "Joel")
## [1] 0.4
```

Finally, Milton asks Zach what the probability of selecting exactly 1 man would be if they sampled pairs of contacts. We can approach this puzzle in the same way.

```
sexSamples<-urnsamples(sex, 2)
sexProbSpace<-probspace(sexSamples)
Prob(sexProbSpace, X1 != X2)
## [1] 0.6
```

The first line sets up the sample space by sampling 2 items from the variable **sex**. As before, the resulting sample space shows 10 possible pairings and contains two variables labelled **X1** (the sex of the first person in the pair) and **X2** (the sex of the second person in the pair). The second line computes the probability space based on the sample space and stored it in *sexProbSpace*, which is a dataframe with three variables: the **X1** and **X2** from the sample space, and **probs** which contains the associated probability of each outcome.

To work out the probability that a pair contains exactly 1 male, we need to set a condition. There are four types of pair we could have: Male-Male, Female-Female, Male-Female, Female-Male. In the first type there are 2 males, in the second no males, and the third and fourth 1 male. So we need to set a condition that selects these final two possibilities. There are lots of ways you could do this but the easiest way is to set the condition that **X1** is not equal to **X2**. In other words, the sex of the first person is not the same as the sex of the second person. This will leave us will the pairs in which the sex is different (in other words there will be exactly one male). In **R** we use *!=* to mean 'not equal to', so our condition will be *X1 != X2* (i.e., the value of **X1** is not equal to **X2**).[7] The result is 0.6 as in the book.

## Empirical probability

Milton and Zach then turn their attention to surviving their journey across the bridge. Milton explains how to calculate the empirical probability of survival based on 10 observations in which 3 people survived and 7 people perished (see Figure 7.2 in the book). In **R** we can do this using the *empirical()* function in the **probs** package.

```
bridgeOutcome<-c("Dead", "Dead", "Alive", "Dead", "Alive", "Dead", "Dead", "Dead", "Alive", "Dea
d")
bridgeOutcome<-data.frame(bridgeOutcome)
bridgeOutcome
##    bridgeOutcome
## 1           Dead
## 2           Dead
## 3          Alive
## 4           Dead
## 5          Alive
## 6           Dead
## 7           Dead
## 8           Dead
## 9          Alive
## 10          Dead
empirical(bridgeOutcome)
```

[7] Again you can wrap this all up into a single command of *Prob(probspace(urnsamples(sex, 2)), X1 != X2)* if you prefer.

```
##   bridgeOutcome probs
## 1         Alive   0.3
## 2          Dead   0.7
```

Line 1 inputs the data in Figure 7.2 into an object called *bridgeOutcome*, the second line converts it to a dataframe (which is required by the *empirical()* function), the third line shows us the data (compare with the outcomes in Figure 7.2 in the book), and the final line uses the *empirical()* function to display the empirical probabilities for each outcome. As in the book, the empirical probability for survival (i.e., the outcome of 'Alive') is 0.3.

## Probability and frequency distributions

### The probability fo a score greater than x: the discs of death

The first challenge Zach faces on the bridge are the discs of death. He has to decide which of two discs (the red or blue) to choose based on knowing the mean speed of rotation (RPM) and standard deviation of the population of discs from which each disc was sampled. The blue disc was sampled from a population with mean RPM $\mu$ = 45 and $\sigma$ = 2, and the red from a population with mean RPM $\mu$ = 33 and $\sigma$ = 8. Zach believed that he could withstand an RPM up to 50 RPM, so the question is which disc is more likely to rotate at a speed of 50RPM or more? Zach could use what he knew about *z*-scores to answer this question. First he needed to convert 50 RPM into a *z*-score for each population using the standard equation:

$$z = \frac{X - \mu}{\sigma}$$

We could convert a score of 50 to a *z*-score separately using the means and standard deviations for the red and blue populations by executing the equation in **R**:

```
(50-33)/8
## [1] 2.125
(50-45)/2
## [1] 2.5
```

We can also do the computations for the red and blue discs at the same time by using vectors:

```
popMeans<-c(33, 45)
popSD<-c(8, 2)
z<-(50-popMeans)/popSD
z
## [1] 2.125 2.500
```

Line 1 creates an object *popMeans* containing the population means for the red and blue discs respectively, the second line creates a similar object *popSD* containing the population standard deviations and line 3 computes *z* by using the equation for *z* (above) and replacing *X* with the value 50. The fourth line is optional: it displays the two values of *z* (which match the values in the book). To get the associated *p*-values from the normal distribution we use the *pnorm()* function and input the values from *z*.*pnorm()* gives us the proportion *below* the value of *z* so to get the proportion above we subtract the result from 1.[8]

```
p<-1-pnorm(z)
percent<-100*p
```

[8] Remember that the total area under the curve is 1.

```
colour<-c("Red", "Blue")
data.frame(colour, popMeans, popSD, z, p, percent)
```

```
##   colour popMeans popSD     z           p   percent
## 1    Red       33     8 2.125 0.016793306 1.6793306
## 2   Blue       45     2 2.500 0.006209665 0.6209665
```

Line 1 creates an object *p* containing the proportion of the normal curve that is greater or equal to the values of *z*, and the second line converts these values to a percentage by multiplying *p* by 100. The last two lines create labels to identify whether values are for the red or blue disc, and then uses the *data.frame()* function to display all of the values we have used in a convenient format. These last two lines aren't necessary, you can just look at the objects *z* and *p* but it hopefully gives you a sense of what we've done.

## The probability fo a score less than x: the tunnels of death

Next up Zach faces the tunnels of death. These tunnels are full of quicksand, and Zach believes he could survive if the tunnel were shorter than 5m long. He has to decide whether the left or right tunnel is more likely to be 5m or less based on the means and standard deviations of the populations from which they came. For the left tunnel the population values are $\mu = 4.5$, $\sigma = 1.5$, and for the right tunnel they are $\mu = 4$, $\sigma = 1$. We can solve this puzzle in R in much the same way as the previous one.

```
popMeans<-c(4.5, 4)
popSD<-c(1.5, 1)
z<-(5-popMeans)/popSD
```

These commands do the same as for the discs, all that has changed are the values of the means and the standard deviations.

```
p<-pnorm(z)
percent<-100*p
colour<-c("Left", "Right")
data.frame(colour, popMeans, popSD, z, p, percent)
```

```
##   colour popMeans popSD         z         p  percent
## 1   Left      4.5   1.5 0.3333333 0.6305587 63.05587
## 2  Right      4.0   1.0 1.0000000 0.8413447 84.13447
```

Line 1 creates an object *p* containing the proportion of the normal curve that is less than or equal to the values of *z*. Remember that *pnorm()* returns the proportion up to the value of *z* so unlike with the discs of death we do not subtract the result from 1. As with the previous example, second line converts the probabilities to percentages, the third line create labels to identify whether values are for the left or right tunnel, and the final line displays all of the values in a convenient format.

## The probability fo a score between two values: the catapults of death

Next up Zach faces the catapults of death. These catapults will fire Zach and Milton across a gap in the bridge and (hopefully) into a net. The only problem is that if it is too weak they will fall to their deaths in the sea below, but if it's too string they'll overshoot and get splatted on the stone floor. Zach believes he needs a catapult that will fire him between 100m and 105m. He has to decide whether the left or right catapult is more likely to fire them the appropriate distance based on the means and standard deviations of the populations from which they came. For the left catapult the population values are $\mu = 103$, $\sigma = 2$, and for the right catapult they are $\mu = 97$, $\sigma = 5$. We can solve this puzzle in R in much the same way as

the previous one. First we need to compute the *z*-scores for when *X* is 100 and then when it is 105. We do this in the same way as before (but twice!).

```
popMeans<-c(103, 97)
popSD<-c(2, 5)

z100<-(100-popMeans)/popSD
z105<-(105-popMeans)/popSD
```

These commands do the same as for the tunnels, all that has changed are the values of the means and the standard deviations. Note we have computed two lots of *z*-scores and called them *z100* and *z105* to differentiate whether they are expressing the value of 100 m or 105m as a *z*-score.

```
p<-pnorm(z105)-pnorm(z100)
percent<-100*p
colour<-c("Left", "Right")
data.frame(colour, popMeans, popSD, z100, z105, p, percent)
```

```
##   colour popMeans popSD z100 z105         p  percent
## 1   Left      103     2 -1.5  1.0 0.7745375 77.45375
## 2  Right       97     5  0.6  1.6 0.2194538 21.94538
```

Line 1 creates an object *p* containing the proportion of the normal curve that lies between the *z*-scores for 105m and 100m. Remember that *pnorm()* returns the proportion up to the value of *z* so whet we're doing here is taking the proportion below the larger *z* (105m) and subtracting form it the proportion below the smaller *z* (100m), in doing so we get the proportion in between the two *z*s. As with the previous example, second line converts the probabilities to percentages, the third line create labels to identify whether values are for the left or right catapult, and the final line displays all of the values in a convenient format. Note that the *z* match those in equation 7.7 in the book, and the *p*s match the text.

## Conditional probability: Deathscotch

Zach's final challenge is deathscotch: he has to cross a chasm by hopping on stones, but some of them disappear when someone stands on them. The stones are shaped as hearts and skulls and Zach has a table showing how how many of each type of stone vanishes of remains upon contact. These data are already loaded in the dataframe that we called *deathscotch* (see earlier). First, let's recreate the contingency table using the *xtabs()* function. We tell this function to use the dataframe *deathscotch* and to create a table based on the variables **Stone** (which will be placed in rows) and **Vanish** (which will be placed in columns). It is important to remember the ~ symbol.

```
xtabs( ~Stone + Vanish, data = deathscotch)
```

```
##        Vanish
## Stone   Remains Vanishes
##   Heart      25       16
##   Skull      28        7
```

The resulting table matches the values in Table 7.1 in the book (although, unlike the book, **R** orders the levels within each variable alphabetically so you'll need to have your wits about you when you compare the tables). To complete this puzzle Zach needed to compute two conditional probabilities: the probability of a stone vanishing given it was a skull (*p(vanishlskull)*) and the probability of a stone vanishing given it was a heart (*p(vanishlheart)*). The first step is to calculate the probabilities for all combinations of types of stones and outcomes. We can do this by inputting the data into the *empirical()* function.

```
probDeath<-empirical(deathscotch)
probDeath
```

```
##   Stone   Vanish      probs
## 1 Heart  Remains 0.32894737
## 2 Skull  Remains 0.36842105
## 3 Heart Vanishes 0.21052632
## 4 Skull Vanishes 0.09210526
```

The first line creates an object called *probDeath* and the second shows it to us. The object *probDeath* has three variables: the first two define the combinations of types of stones and whether or not they vanished, and the third variable **probs** has the empirical probability of each combination of events. For example, the third row tells us that the probability of being a heart stone and vanishing is 0.21 (this is called the intersection). We can use these values to compute conditional probabilities:

```
Prob(probDeath, Vanish =="Vanishes", given = Stone == "Skull")
## [1] 0.2
Prob(probDeath, Vanish =="Vanishes", given = Stone == "Heart")
## [1] 0.3902439
```

The first line uses the *Prob()* function and inputs the *probDeath* object that we just created that contains the probabilities associated with all combinations of stones and outcomes. It uses these probabilities to calculate the probability that the variable **Vanish** is equal to (note the ==) the value *Vanishes* (note the speech-marks), given that the variable **Stone** is equal to the value *Skull*. The result is 0.2, which matches the value in equation 7.11 in the book. The second line does the same but conditional on the variable **Stone** is equal to the value *Heart* and returns a value of 0.39, which matches the value in equation 7.12 in the book.

---

This handout is written to be used in conjunction with: Field, A. P. (2016). An adventure in statistics; the reality enigma. London: Sage.