Introduction to Probability and Statistics Using R

Third Edition

G. Jay Kerns

2017-02-04

Contents

Chapter 1

Placeholder

Chapter 2

An Introduction to Probability and Statistics

Chapter 3

Probability

In this chapter we define the basic terminology associated with probability and derive some of its properties. We discuss three interpretations of probability. We discuss conditional probability and independent events, along with Bayes' Theorem. We finish the chapter with an introduction to random variables, which paves the way for the next two chapters.

In this book we distinguish between two types of experiments: *deterministic* and *random*. A *deterministic* experiment is one whose outcome may be predicted with certainty beforehand, such as combining Hydrogen and Oxygen, or adding two numbers such as $2 + 3$. A *random* experiment is one whose outcome is determined by chance. We posit that the outcome of a random experiment may not be predicted with certainty beforehand, even in principle. Examples of random experiments include tossing a coin, rolling a die, and throwing a dart on a board, how many red lights you encounter on the drive home, how many ants traverse a certain patch of sidewalk over a short period, *etc*.

What do I want them to know?

- that there are multiple interpretations of probability, and the methods used depend somewhat on the philosophy chosen
- nuts and bolts of basic probability jargon: sample spaces, events, probability functions, *etc*.
- how to count
- conditional probability and its relationship with independence
- Bayes' Rule and how it relates to the subjective view of probability
- what we mean by 'random variables', and where they come from

3.1 Sample Spaces

For a random experiment E, the set of all possible outcomes of E is called the *sample space* and is denoted by the letter S. For a coin-toss experiment, S would be the results "Head" and "Tail", which we may represent by $S = \{H, T\}$. Formally, the performance of a random experiment is the unpredictable selection of an outcome in S.

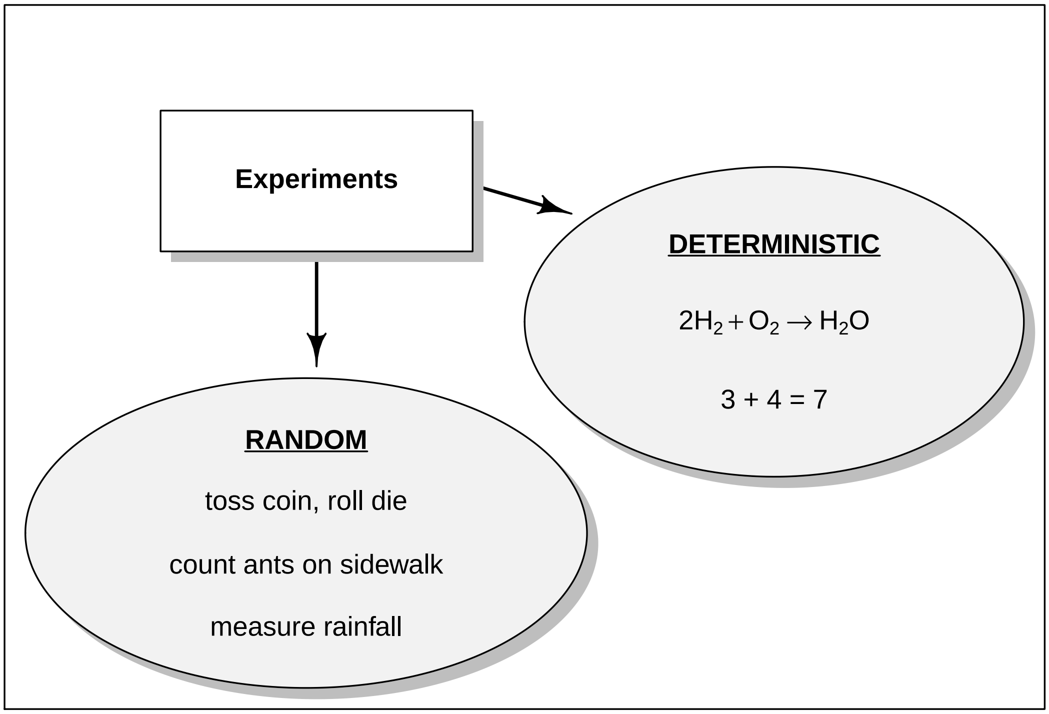

Figure 3.1: (ref:cap-dagram)

3.1.1 How to do it with R

Most of the probability work in this book is done with the `prob` package [77]. A sample space is (usually) represented by a *data frame*, that is, a rectangular collection of variables (see Section **??**). Each row of the data frame corresponds to an outcome of the experiment. The data frame choice is convenient both for its simplicity and its compatibility with the R Commander. Data frames alone are, however, not sufficient to describe some of the more interesting probabilistic applications we will study later; to handle those we will need to consider a more general *list* data structure. See Section 4.4.2 for details.

Example 3.1. Consider the random experiment of dropping a Styrofoam cup onto the floor from a height of four feet. The cup hits the ground and eventually comes to rest. It could land upside down, right side up, or it could land on its side. We represent these possible outcomes of the random experiment by the following.

```
S <- data.frame(lands = c("down","up","side"))
S
```

```
  lands
1  down
2    up
3  side
```

The sample space S contains the column `lands` which stores the outcomes `down`, `up`, and `side`.

Some sample spaces are so common that convenience wrappers were written to set them up with minimal effort. The underlying machinery that does the work includes the `expand.grid` function in the `base` package [106], `combn` in the `combinat` package [22], and `permsn` in the `prob` package [77][1].

Consider the random experiment of tossing a coin. The outcomes are H and T. We can set up the sample space quickly with the `tosscoin` function:

```
tosscoin(1)
```

```
  toss1
1     H
2     T
```

The number 1 tells `tosscoin` that we only want to toss the coin once. We could toss it three times:

```
tosscoin(3)
```

```
  toss1 toss2 toss3
1     H     H     H
2     T     H     H
3     H     T     H
4     T     T     H
```

[1]The seasoned R user can get the job done without the convenience wrappers. I encourage the beginner to use them to get started, but I also recommend that introductory students wean themselves as soon as possible. The wrappers were designed for ease and intuitive use, not for speed or efficiency.

```
5    H    H    T
6    T    H    T
7    H    T    T
8    T    T    T
```

Alternatively we could roll a fair die:

```r
rolldie(1)
```

```
  X1
1  1
2  2
3  3
4  4
5  5
6  6
```

The `rolldie` function defaults to a 6-sided die, but we can specify others with the `nsides` argument. The command `rolldie(3, nsides = 4)` would be used to roll a 4-sided die three times.

Perhaps we would like to draw one card from a standard set of playing cards (it is a long data frame):

```r
head(cards())
```

```
  rank suit
1    2 Club
2    3 Club
3    4 Club
4    5 Club
5    6 Club
6    7 Club
```

The `cards` function that we just used has optional arguments `jokers` (if you would like Jokers to be in the deck) and `makespace` which we will discuss later. There is also a `roulette` function which returns the sample space associated with one spin on a roulette wheel. There are EU and USA versions available. Interested readers may contribute any other game or sample spaces that may be of general interest.

3.1.2 Sampling from Urns

This is perhaps the most fundamental type of random experiment. We have an urn that contains a bunch of distinguishable objects (balls) inside. We shake up the urn, reach inside, grab a ball, and take a look. That's all.

But there are all sorts of variations on this theme. Maybe we would like to grab more than one ball – say, two balls. What are all of the possible outcomes of the experiment now? It depends on how we sample. We could select a ball, take a look, put it back, and sample again. Another way would be to select a ball, take a look – but do not put it back – and sample again (equivalently, just reach in and grab two balls). There are certainly more possible outcomes of the experiment in the former case than in the latter. In the first (second) case we say that sampling is done *with (without) replacement*.

There is more. Suppose we do not actually keep track of which ball came first. All we observe are the two balls, and we have no idea about the order in which they were selected. We call this *unordered sampling* (in contrast to *ordered*) because the order of the selections does not matter with respect to what we observe. We might as well have selected the balls and put them in a bag before looking.

Note that this one general class of random experiments contains as a special case all of the common elementary random experiments. Tossing a coin twice is equivalent to selecting two balls labeled H and T from an urn, with replacement. The die-roll experiment is equivalent to selecting a ball from an urn with six elements, labeled 1 through 6.

How to do it with R

The `prob` package [77] accomplishes sampling from urns with the `urnsamples` function, which has arguments `x`, `size`, `replace`, and `ordered`. The argument `x` represents the urn from which sampling is to be done. The `size` argument tells how large the sample will be. The `ordered` and `replace` arguments are logical and specify how sampling will be performed. We will discuss each in turn.
Example 3.2. Let our urn simply contain three balls, labeled 1, 2, and 3, respectively. We are going to take a sample of size 2 from the urn.

Ordered, With Replacement

If sampling is with replacement, then we can get any outcome 1, 2, or 3 on any draw. Further, by “ordered” we mean that we shall keep track of the order of the draws that we observe. We can accomplish this in R with

```r
urnsamples(1:3, size = 2, replace = TRUE, ordered = TRUE)
```

```
  X1 X2
1  1  1
2  2  1
3  3  1
4  1  2
5  2  2
6  3  2
7  1  3
8  2  3
9  3  3
```

Notice that rows 2 and 4 are identical, save for the order in which the numbers are shown. Further, note that every possible pair of the numbers 1 through 3 are listed. This experiment is equivalent to rolling a 3-sided die twice, which we could have accomplished with `rolldie(2, nsides = 3)`.

Ordered, Without Replacement

Here sampling is without replacement, so we may not observe the same number twice in any row. Order is still important, however, so we expect to see the outcomes `1,2` and `2,1` somewhere in our data frame.

```
urnsamples(1:3, size = 2, replace = FALSE, ordered = TRUE)
```

```
  X1 X2
1  1  2
2  2  1
3  1  3
4  3  1
5  2  3
6  3  2
```

This is just as we expected. Notice that there are less rows in this answer due to the more restrictive sampling procedure. If the numbers 1, 2, and 3 represented “Fred”, “Mary”, and “Sue”, respectively, then this experiment would be equivalent to selecting two people of the three to serve as president and vice-president of a company, respectively, and the sample space shown above lists all possible ways that this could be done.

Unordered, Without Replacement

Again, we may not observe the same outcome twice, but in this case, we will only retain those outcomes which (when jumbled) would not duplicate earlier ones.

```
urnsamples(1:3, size = 2, replace = FALSE, ordered = FALSE)
```

```
  X1 X2
1  1  2
2  1  3
3  2  3
```

This experiment is equivalent to reaching in the urn, picking a pair, and looking to see what they are. This is the default setting of `urnsamples`, so we would have received the same output by simply typing `urnsamples(1:3, 2)`.

Unordered, With Replacement

The last possibility is perhaps the most interesting. We replace the balls after every draw, but we do not remember the order in which the draws came.

```
urnsamples(1:3, size = 2, replace = TRUE, ordered = FALSE)
```

```
  X1 X2
1  1  1
2  1  2
```

```
3  1  3
4  2  2
5  2  3
6  3  3
```

We may interpret this experiment in a number of alternative ways. One way is to consider this as simply putting two 3-sided dice in a cup, shaking the cup, and looking inside – as in a game of *Liar's Dice*, for instance. Each row of the sample space is a potential pair we could observe. Another way is to view each outcome as a separate method to distribute two identical golf balls into three boxes labeled 1, 2, and 3. Regardless of the interpretation, `urnsamples` lists every possible way that the experiment can conclude.

Note that the urn does not need to contain numbers; we could have just as easily taken our urn to be `x = c("Red","Blue","Green")`. But, there is an *important# point to mention before proceeding. Astute readers will notice that in our example, the balls in the urn were* distinguishable* in the sense that each had a unique label to distinguish it from the others in the urn. A natural question would be, "What happens if your urn has indistinguishable elements, for example, what if `x = c("Red","Red","Blue")`?" The answer is that `urnsamples` behaves as if each ball in the urn is distinguishable, regardless of its actual contents. We may thus imagine that while there are two red balls in the urn, the balls are such that we can tell them apart (in principle) by looking closely enough at the imperfections on their surface.

In this way, when the `x` argument of `urnsamples` has repeated elements, the resulting sample space may appear to be `ordered = TRUE` even when, in fact, the call to the function was `urnsamples(..., ordered = FALSE)`. Similar remarks apply for the `replace` argument.

3.2 Events

An *event* A is merely a collection of outcomes, or in other words, a subset of the sample space[2]. After the performance of a random experiment E we say that the event A *occurred* if the experiment's outcome belongs to A. We say that a bunch of events $A_1, A_2, A_3, \ldots$ are *mutually exclusive* or *disjoint* if $A_i \cap A_j = \emptyset$ for any distinct pair $A_i \neq A_j$. For instance, in the coin-toss experiment the events $A = \{\text{Heads}\}$ and $B = \{\text{Tails}\}$ would be mutually exclusive. Now would be a good time to review the algebra of sets in Appendix **??**.

3.2.1 How to do it with R

Given a data frame sample/probability space `S`, we may extract rows using the `[]` operator:

```
S <- tosscoin(2, makespace = TRUE)
S[1:3, ]
```

[2]This naive definition works for finite or countably infinite sample spaces, but is inadequate for sample spaces in general. In this book, we will not address the subtleties that arise, but will refer the interested reader to any text on advanced probability or measure theory.

```
  toss1 toss2 probs
1     H     H  0.25
2     T     H  0.25
3     H     T  0.25
```

```
S[c(2,4), ]
```

```
  toss1 toss2 probs
2     T     H  0.25
4     T     T  0.25
```

and so forth. We may also extract rows that satisfy a logical expression using the `subset` function, for instance

```
S <- cards()
```

```
subset(S, suit == "Heart")
```

```
   rank  suit
27    2 Heart
28    3 Heart
29    4 Heart
30    5 Heart
31    6 Heart
32    7 Heart
33    8 Heart
34    9 Heart
35   10 Heart
36    J Heart
37    Q Heart
38    K Heart
39    A Heart
```

```
subset(S, rank %in% 7:9)
```

```
   rank    suit
6     7    Club
7     8    Club
8     9    Club
19    7 Diamond
20    8 Diamond
21    9 Diamond
32    7   Heart
33    8   Heart
34    9   Heart
45    7   Spade
46    8   Spade
47    9   Spade
```

We could continue indefinitely. Also note that mathematical expressions are allowed:

```
subset(rolldie(3), X1+X2+X3 > 16)
```

```
    X1 X2 X3
180  6  6  5
210  6  5  6
215  5  6  6
216  6  6  6
```

3.2.2 Functions for Finding Subsets

It does not take long before the subsets of interest become complicated to specify. Yet the main idea remains: we have a particular logical condition to apply to each row. If the row satisfies the condition, then it should be in the subset. It should not be in the subset otherwise. The ease with which the condition may be coded depends of course on the question being asked. Here are a few functions to get started.

The `%in%` function

The function `%in%` helps to learn whether each value of one vector lies somewhere inside another vector.

```
x <- 1:10
y <- 8:12
y %in% x
```

```
[1]  TRUE  TRUE  TRUE FALSE FALSE
```

Notice that the returned value is a vector of length 5 which tests whether each element of `y` is in `x`, in turn.

The `isin` function

It is more common to want to know whether the *whole* vector `y` is in `x`. We can do this with the `isin` function.

```
isin(x,y)
```

```
[1] FALSE
```

Of course, one may ask why we did not try something like `all(y %in% x)`, which would give a single result, `TRUE`. The reason is that the answers are different in the case that `y` has repeated values. Compare:

```
x <- 1:10
y <- c(3,3,7)
```

```r
all(y %in% x)
```

```
[1] TRUE
```

```r
isin(x,y)
```

```
[1] FALSE
```

The reason for the above is of course that `x` contains the value 3, but `x` does not have *two* 3's. The difference is important when rolling multiple dice, playing cards, *etc*. Note that there is an optional argument `ordered` which tests whether the elements of `y` appear in `x` in the order in which they are appear in `y`. The consequences are

```r
isin(x, c(3,4,5), ordered = TRUE)
```

```
[1] TRUE
```

```r
isin(x, c(3,5,4), ordered = TRUE)
```

```
[1] FALSE
```

The connection to probability is that have a data frame sample space and we would like to find a subset of that space. A `data.frame` method was written for `isin` that simply applies the function to each row of the data frame. We can see the method in action with the following:

```r
S <- rolldie(4)
subset(S, isin(S, c(2,2,6), ordered = TRUE))
```

```
     X1 X2 X3 X4
188   2  2  6  1
404   2  2  6  2
620   2  2  6  3
836   2  2  6  4
1052  2  2  6  5
1088  2  2  1  6
1118  2  1  2  6
1123  1  2  2  6
1124  2  2  2  6
1125  3  2  2  6
1126  4  2  2  6
1127  5  2  2  6
1128  6  2  2  6
1130  2  3  2  6
1136  2  4  2  6
1142  2  5  2  6
1148  2  6  2  6
1160  2  2  3  6
1196  2  2  4  6
1232  2  2  5  6
1268  2  2  6  6
```

There are a few other functions written to find useful subsets, namely, `countrep` and `isrep`. Essentially these were written to test for (or count) a specific number of designated values in outcomes. See the documentation for details.

3.2.3 Set Union, Intersection, and Difference

Given subsets A and B, it is often useful to manipulate them in an algebraic fashion. To this end, we have three set operations at our disposal: union, intersection, and difference. Below is a table that summarizes the pertinent information about these operations.

Table 3.1: Basic set operations. The first column lists the name, the second shows the typical notation, the third describes set membership, and the fourth shows how to accomplish it with R.

Name	Denoted	Defined by elements	Code
Union	$A \cup B$	in A or B or both	`union(A,B)`
Intersection	$A \cap B$	in both A and B	`intersect(A,B)`
Difference	$A \backslash B$	in A but not in B	`setdiff(A,B)`

Some examples follow.

```
S <- cards()
A <- subset(S, suit == "Heart")
B <- subset(S, rank %in% 7:9)
```

We can now do some set algebra:

```
union(A,B)
```

```
   rank    suit
6     7    Club
7     8    Club
8     9    Club
19    7 Diamond
20    8 Diamond
21    9 Diamond
27    2   Heart
28    3   Heart
29    4   Heart
30    5   Heart
31    6   Heart
32    7   Heart
33    8   Heart
34    9   Heart
35   10   Heart
36    J   Heart
37    Q   Heart
```

```
38    K   Heart
39    A   Heart
45    7   Spade
46    8   Spade
47    9   Spade
```

```
intersect(A,B)
```

```
   rank  suit
32    7 Heart
33    8 Heart
34    9 Heart
```

```
setdiff(A,B)
```

```
   rank  suit
27    2 Heart
28    3 Heart
29    4 Heart
30    5 Heart
31    6 Heart
35   10 Heart
36    J Heart
37    Q Heart
38    K Heart
39    A Heart
```

```
setdiff(B,A)
```

```
   rank    suit
6     7    Club
7     8    Club
8     9    Club
19    7 Diamond
20    8 Diamond
21    9 Diamond
45    7   Spade
46    8   Spade
47    9   Spade
```

Notice that `setdiff` is not symmetric. Further, note that we can calculate the *complement* of a set A, denoted A^c and defined to be the elements of S that are not in A simply with `setdiff(S,A)`. There have been methods written for `intersect`, `setdiff`, `subset`, and `union` in the case that the input objects are of class `ps`. See Section 4.4.2.

Note. When the `prob` package [77] loads you will notice a message: `The following object(s) are masked from package:base: intersect, setdiff`. The reason for this message is that there already exist methods for the functions `intersect`, `setdiff`, `subset`, and `union` in the `base` package which ships with R. However, these methods were designed for when the arguments are vectors of the same mode. Since we are manipulating sample spaces which are data frames and lists, it was necessary to write methods to handle those cases as well. When the `prob` package is loaded, R recognizes that there are multiple versions of the same function in the search path and acts to shield the new definitions from the existing ones. But there is no cause for alarm, thankfully, because the `prob` functions have been carefully defined to match the usual `base` package definition in the case that the arguments are vectors.

Chapter 4

+END_note

4.1 Model Assignment

Let us take a look at the coin-toss experiment more closely. What do we mean when we say "the probability of Heads" or write $\mathbb{P}(\text{Heads})$? Given a coin and an itchy thumb, how do we go about finding what $\mathbb{P}(\text{Heads})$ should be?

4.1.1 The Measure Theory Approach

This approach states that the way to handle $\mathbb{P}(\text{Heads})$ is to define a mathematical function, called a *probability measure*, on the sample space. Probability measures satisfy certain axioms (to be introduced later) and have special mathematical properties, so not just any mathematical function will do. But in any given physical circumstance there are typically all sorts of probability measures from which to choose, and it is left to the experimenter to make a reasonable choice – one usually based on considerations of objectivity. For the tossing coin example, a valid probability measure assigns probability p to the event {Heads}, where p is some number $0 \leq p \leq 1$. An experimenter that wishes to incorporate the symmetry of the coin would choose $p = 1/2$ to balance the likelihood of {Heads} and {Tails}.

Once the probability measure is chosen (or determined), there is not much left to do. All assignments of probability are made by the probability function, and the experimenter needs only to plug the event {Heads} into to the probability function to find $\mathbb{P}(\text{Heads})$. In this way, the probability of an event is simply a calculated value, nothing more, nothing less. Of course this is not the whole story; there are many theorems and consequences associated with this approach that will keep us occupied for the remainder of this book. The approach is called *measure theory* because the measure (probability) of a set (event) is associated with how big it is (how likely it is to occur).

The measure theory approach is well suited for situations where there is symmetry to the experiment, such as flipping a balanced coin or spinning an arrow around a circle with well-defined pie slices. It is also handy because of its mathematical simplicity, elegance, and flexibility. There are literally volumes of information that one can prove about probability measures, and the cold rules of mathematics allow us to analyze intricate probabilistic problems with vigor.

The large degree of flexibility is also a disadvantage, however. When symmetry fails it is not always obvious what an "objective" choice of probability measure should be; for instance, what probability should we assign to {Heads} if we spin the coin rather than flip it? (It is not $1/2$.) Furthermore, the mathematical rules are restrictive when we wish to incorporate subjective knowledge into the model, knowledge which changes over time and depends on the experimenter, such as personal knowledge about the properties of the specific coin being flipped, or of the person doing the flipping.

The mathematician who revolutionized this way to do probability theory was Andrey Kolmogorov, who published a landmark monograph in 1933. See http://www-history.mcs.st-andrews.ac.uk/Mathematicians/Kolmogorov.html for more information.

4.1.2 Relative Frequency Approach

This approach states that the way to determine $\mathbb{P}(\text{Heads})$ is to flip the coin repeatedly, in exactly the same way each time. Keep a tally of the number of flips and the number of Heads observed. Then a good approximation to $\mathbb{P}(\text{Heads})$ will be

$$\mathbb{P}(\text{Heads}) \approx \frac{\text{number of observed Heads}}{\text{total number of flips}}. \tag{4.1}$$

The mathematical underpinning of this approach is the celebrated *Law of Large Numbers# which may be loosely described as follows. Let E be a random experiment in which the event A either does or does not occur. Perform the experiment repeatedly, in an identical manner, in such a way that the successive experiments do not influence each other. After each experiment, keep a running tally of whether or not the event A occurred. Let S_n count the number of times that A occurred in the n experiments. Then the law of large numbers says that

$$\frac{S_n}{n} \to \mathbb{P}(A) \text{ as } n \to \infty. \tag{4.2}$$

As the reasoning goes, to learn about the probability of an event A we need only repeat the random experiment to get a reasonable estimate of the probability's value, and if we are not satisfied with our estimate then we may simply repeat the experiment more times all the while confident that with more and more experiments our estimate will stabilize to the true value.

The frequentist approach is good because it is relatively light on assumptions and does not worry about symmetry or claims of objectivity like the measure-theoretic approach does. It is perfect for the spinning coin experiment. One drawback to the method is that one can never know the exact value of a probability, only a long-run approximation. It also does not work well with experiments that can not be repeated indefinitely, say, the probability that it will rain today, the chances that you get will get an A in your Statistics class, or the probability that the world is destroyed by nuclear war.

This approach was espoused by Richard von Mises in the early twentieth century, and some of his main ideas were incorporated into the measure theory approach. See http://www-history.mcs.st-andrews.ac.uk/Biographies/Mises.html for more.

4.1.3 The Subjective Approach

The subjective approach interprets probability as the experimenter's *degree of belief* that the event will occur. The estimate of the probability of an event is based on the totality of the individual's knowledge at the time. As new information becomes available, the estimate is modified accordingly to best reflect his/her current knowledge. The method by which the probabilities are updated is commonly done with Bayes' Rule, discussed in Section 4.6.

So for the coin toss example, a person may have $\mathbb{P}(\text{Heads}) = 1/2$ in the absence of additional information. But perhaps the observer knows additional information about the coin or the thrower that would shift the probability in a certain direction. For instance, parlor magicians may be trained to be quite skilled at tossing coins, and some are so skilled that they may toss a fair coin and get nothing but Heads, indefinitely. I have *seen* this. It was similarly claimed in *Bringing Down the House* [97] that MIT students were accomplished enough with cards to be able to cut a deck to the same location, every single time. In such cases, one clearly should use the additional information to assign $\mathbb{P}(\text{Heads})$ away from the symmetry value of $1/2$.

This approach works well in situations that cannot be repeated indefinitely, for example, to assign your probability that you will get an A in this class, the chances of a devastating nuclear war, or the likelihood that a cure for the common cold will be discovered.

The roots of subjective probability reach back a long time. See http://en.wikipedia.org/wiki/Subjective_probability for a short discussion and links to references about the subjective approach.

4.1.4 Equally Likely Model (ELM)

We have seen several approaches to the assignment of a probability model to a given random experiment and they are very different in their underlying interpretation. But they all cross paths when it comes to the equally likely model which assigns equal probability to all elementary outcomes of the experiment.

The ELM appears in the measure theory approach when the experiment boasts symmetry of some kind. If symmetry guarantees that all outcomes have equal "size", and if outcomes with equal "size" should get the same probability, then the ELM is a logical objective choice for the experimenter. Consider the balanced 6-sided die, the fair coin, or the dart board with equal-sized wedges.

The ELM appears in the subjective approach when the experimenter resorts to indifference or ignorance with respect to his/her knowledge of the outcome of the experiment. If the experimenter has no prior knowledge to suggest that (s)he prefer Heads over Tails, then it is reasonable for the him/her to assign equal subjective probability to both possible outcomes.

The ELM appears in the relative frequency approach as a fascinating fact of Nature: when we flip balanced coins over and over again, we observe that the proportion of times that the coin comes up

Heads tends to $1/2$. Of course if we assume that the measure theory applies then we can prove that the sample proportion must tend to 1/2 as expected, but that is putting the cart before the horse, in a manner of speaking.

The ELM is only available when there are finitely many elements in the sample space.

How to do it with R

In the `prob` package [77], a probability space is an object of outcomes `S` and a vector of probabilities (called `probs`) with entries that correspond to each outcome in `S`. When `S` is a data frame, we may simply add a column called `probs` to `S` and we will be finished; the probability space will simply be a data frame which we may call `S`. In the case that `S` is a list, we may combine the `outcomes` and `probs` into a larger list, `space`; it will have two components: `outcomes` and `probs`. The only requirements we need are for the entries of `probs` to be nonnegative and `sum(probs)` to be one.

To accomplish this in R, we may use the `probspace` function. The general syntax is `probspace(x, probs)`, where `x` is a sample space of outcomes and `probs` is a vector (of the same length as the number of outcomes in `x`). The specific choice of `probs` depends on the context of the problem, and some examples follow to demonstrate some of the more common choices.
Example 4.1. The Equally Likely Model asserts that every outcome of the sample space has the same probability, thus, if a sample space has n outcomes, then `probs` would be a vector of length n with identical entries $1/n$. The quickest way to generate `probs` is with the `rep` function. We will start with the experiment of rolling a die, so that $n = 6$. We will construct the sample space, generate the `probs` vector, and put them together with `probspace`.

```
outcomes <- rolldie(1)
p <- rep(1/6, times = 6)
probspace(outcomes, probs = p)
```

```
  X1     probs
1  1 0.1666667
2  2 0.1666667
3  3 0.1666667
4  4 0.1666667
5  5 0.1666667
6  6 0.1666667
```

The `probspace` function is designed to save us some time in the most common situations. For example, due to the especial simplicity of the sample space in this case, we could have achieved the same result with only (note the name change for the first column)

```
probspace(1:6, probs = p)
```

```
  x     probs
1 1 0.1666667
2 2 0.1666667
3 3 0.1666667
4 4 0.1666667
```

```
5 5 0.1666667
6 6 0.1666667
```

Further, since the equally likely model plays such a fundamental role in the study of probability the `probspace` function will assume that the equally model is desired if no `probs` are specified. Thus, we get the same answer with only

```
probspace(1:6)
```

```
  x     probs
1 1 0.1666667
2 2 0.1666667
3 3 0.1666667
4 4 0.1666667
5 5 0.1666667
6 6 0.1666667
```

And finally, since rolling dice is such a common experiment in probability classes, the `rolldie` function has an additional logical argument `makespace` that will add a column of equally likely `probs` to the generated sample space:

```
rolldie(1, makespace = TRUE)
```

```
  X1     probs
1  1 0.1666667
2  2 0.1666667
3  3 0.1666667
4  4 0.1666667
5  5 0.1666667
6  6 0.1666667
```

or just `rolldie(1, TRUE)`. Many of the other sample space functions (`tosscoin`, `cards`, `roulette`, *etc.*) have similar `makespace` arguments. Check the documentation for details.

One sample space function that does NOT have a `makespace` option is the `urnsamples` function. This was intentional. The reason is that under the varied sampling assumptions the outcomes in the respective sample spaces are NOT, in general, equally likely. It is important for the user to carefully consider the experiment to decide whether or not the outcomes are equally likely and then use `probspace` to assign the model.

Example 4.2 (An unbalanced coin)**.** While the `makespace` argument to `tosscoin` is useful to represent the tossing of a *fair* coin, it is not always appropriate. For example, suppose our coin is not perfectly balanced, for instance, maybe the H side is somewhat heavier such that the chances of a H appearing in a single toss is 0.70 instead of 0.5. We may set up the probability space with

```
probspace(tosscoin(1), probs = c(0.70, 0.30))
```

```
  toss1 probs
1     H   0.7
2     T   0.3
```

The same procedure can be used to represent an unbalanced die, roulette wheel, *etc*.

4.1.5 Words of Warning

It should be mentioned that while the splendour of R is uncontested, it, like everything else, has limits both with respect to the sample/probability spaces it can manage and with respect to the finite accuracy of the representation of most numbers (see the R FAQ 7.31). When playing around with probability, one may be tempted to set up a probability space for tossing 100 coins or rolling 50 dice in an attempt to answer some scintillating question. (Bear in mind: rolling a die just 9 times has a sample space with over *10 million* outcomes.)

Alas, even if there were enough RAM to barely hold the sample space (and there were enough time to wait for it to be generated), the infinitesimal probabilities that are associated with *so many* outcomes make it difficult for the underlying machinery to handle reliably. In some cases, special algorithms need to be called just to give something that holds asymptotically. User beware.

4.2 Properties of Probability

4.2.1 Probability Functions

A *probability function* is a rule that associates with each event A of the sample space a single number $\mathbb{P}(A) = p$, called the *probability of* A. Any probability function $\mathbb{P}$ satisfies the following three Kolmogorov Axioms:

Axiom 1. <> $\mathbb{P}(A) \geq 0$ for any event $A \subset S$.

Axiom 2. <> $\mathbb{P}(S) = 1$.

Axiom 3. <> If the events $A_1, A_2, A_3 \ldots$ are disjoint then

$$\mathbb{P}\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{i=1}^{n} \mathbb{P}(A_i) \text{ for every } n, \tag{4.3}$$

and furthermore,

$$\mathbb{P}\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} \mathbb{P}(A_i). \tag{4.4}$$

The intuition behind the axioms goes like this: first, the probability of an event should never be negative. Second, since the sample space contains all possible outcomes, its probability should be one, or 100%. The last axiom may look intimidating but it simply means that in a sequence of disjoint events (in other words, sets that do not overlap), the total probability (measure) should equal the sum of its parts. For example, the chance of rolling a 1 or a 2 on a die should be the chance of rolling a 1 plus the chance of rolling a 2.

4.2.2 Properties

For any events A and B,

1. <> $\mathbb{P}(A^c) = 1 - \mathbb{P}(A)$. **Proof:** Since $A \cup A^c = S$ and $A \cap A^c = \emptyset$, we have
$$1 = \mathbb{P}(S) = \mathbb{P}(A \cup A^c) = \mathbb{P}(A) + \mathbb{P}(A^c).$$
2. $\mathbb{P}(\emptyset) = 0$. **Proof:** Note that $\emptyset = S^c$, and use Property 1.
3. If $A \subset B$, then $\mathbb{P}(A) \leq \mathbb{P}(B)$. **Proof:** Write $B = A \cup (B \cap A^c)$, and notice that $A \cap (B \cap A^c) = \emptyset$; thus
$$\mathbb{P}(B) = \mathbb{P}(A \cup (B \cap A^c)) = \mathbb{P}(A) + \mathbb{P}(B \cap A^c) \geq \mathbb{P}(A),$$
since $\mathbb{P}(B \cap A^c) \geq 0$.
4. $0 \leq \mathbb{P}(A) \leq 1$. **Proof:** The left inequality is immediate from Axiom **??**, and the second inequality follows from Property 3 since $A \subset S$.
5. **The General Addition Rule.**
$$\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \cap B). \tag{4.5}$$
More generally, for events $A_1, A_2, A_3, \ldots, A_n$,
$$\mathbb{P}\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{i=1}^{n} \mathbb{P}(A_i) - \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathbb{P}(A_i \cap A_j) + \cdots + (-1)^{n-1} \mathbb{P}\left(\bigcap_{i=1}^{n} A_i\right) \tag{4.6}$$
6. **The Theorem of Total Probability.** Let $B_1, B_2, \ldots, B_n$ be mutually exclusive and exhaustive. Then
$$\mathbb{P}(A) = \mathbb{P}(A \cap B_1) + \mathbb{P}(A \cap B_2) + \cdots + \mathbb{P}(A \cap B_n). \tag{4.7}$$

4.2.3 Assigning Probabilities

A model of particular interest is the *equally likely model*. The idea is to divide the sample space S into a finite collection of elementary events $\{a_1, a_2, \ldots, a_N\}$ that are equally likely in the sense that each a_i has equal chances of occurring. The probability function associated with this model must satisfy $\mathbb{P}(S) = 1$, by Axiom 2. On the other hand, it must also satisfy
$$\mathbb{P}(S) = \mathbb{P}(\{a_1, a_2, \ldots, a_N\}) = \mathbb{P}(a_1 \cup a_2 \cup \cdots \cup a_N) = \sum_{i=1}^{N} \mathbb{P}(a_i),$$
by Axiom 3. Since $\mathbb{P}(a_i)$ is the same for all i, each one necessarily equals $1/N$.

For an event $A \subset S$, we write A as a collection of elementary outcomes: if $A = \{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\}$ then A has k elements and

$$
\begin{aligned}
\mathbb{P}(A) &= \mathbb{P}(a_{i_1}) + \mathbb{P}(a_{i_2}) + \cdots + \mathbb{P}(a_{i_k}),\\
&= \frac{1}{N} + \frac{1}{N} + \cdots + \frac{1}{N},\\
&= \frac{k}{N} = \frac{\#(A)}{\#(S)}.
\end{aligned}
$$

In other words, under the equally likely model, the probability of an event A is determined by the number of elementary events that A contains.

Example 4.3. Consider the random experiment E of tossing a coin. Then the sample space is $S = \{H, T\}$, and under the equally likely model, these two outcomes have $\mathbb{P}(H) = \mathbb{P}(T) = 1/2$. This model is taken when it is reasonable to assume that the coin is fair.

Example 4.4. Suppose the experiment E consists of tossing a fair coin twice. The sample space may be represented by $S = \{HH, HT, TH, TT\}$. Given that the coin is fair and that the coin is tossed in an independent and identical manner, it is reasonable to apply the equally likely model.

What is $\mathbb{P}$(at least 1 Head)? Looking at the sample space we see the elements HH, HT, and TH have at least one Head; thus, $\mathbb{P}$(at least 1 Head) = $3/4$.

What is $\mathbb{P}$(no Heads)? Notice that the event {no Heads} = {at least one Head}^{c}, which by Property **??** means $\mathbb{P}$(no Heads) = $1 - \mathbb{P}$(at least one head) $= 1 - 3/4 = 1/4$. It is obvious in this simple example that the only outcome with no Heads is TT, however, this complementation trick can be handy in more complicated problems.

Example 4.5. Imagine a three child family, each child being either Boy (B) or Girl (G). An example sequence of siblings would be BGB. The sample space may be written

$$S = \{BBB,\ BGB,\ GBB,\ GGB,\ BBG,\ BGG,\ GBG,\ GGG,\ \}.$$

Note that for many reasons (for instance, it turns out that girls are slightly more likely to be born than boys), this sample space is *not* equally likely. For the sake of argument, however, we will assume that the elementary outcomes each have probability $1/8$.

What is $\mathbb{P}$(exactly 2 Boys)? Inspecting the sample space reveals three outcomes with exactly two boys: $\{BBG, BGB, GBB\}$. Therefore $\mathbb{P}$(exactly 2 Boys) = $3/8$.

What is $\mathbb{P}$(at most 2 Boys)? One way to solve the problem would be to count the outcomes that have 2 or less Boys, but a quicker way would be to recognize that the only way that the event {at most 2 Boys} does *not* occur is the event {all Boys}.

Thus

$$\mathbb{P}(\text{at most 2 Boys}) = 1 - \mathbb{P}(BBB) = 1 - 1/8 = 7/8.$$

Example 4.6. Consider the experiment of rolling a six-sided die, and let the outcome be the face showing up when the die comes to rest. Then $S = \{1, 2, 3, 4, 5, 6\}$. It is usually reasonable to suppose that the die is fair, so that the six outcomes are equally likely.

Example 4.7. Consider a standard deck of 52 cards. These are usually labeled with the four *suits*: Clubs, Diamonds, Hearts, and Spades, and the 13 *ranks*: 2, 3, 4, . . . , 10, Jack (J), Queen (Q), King (K), and Ace (A). Depending on the game played, the Ace may be ranked below 2 or above King.

Let the random experiment E consist of drawing exactly one card from a well-shuffled deck, and let the outcome be the face of the card. Define the events $A = \{\text{draw an Ace}\}$ and $B = \{\text{draw a Club}\}$. Bear in mind: we are only drawing one card.

Immediately we have $\mathbb{P}(A) = 4/52$ since there are four Aces in the deck; similarly, there are 13 Clubs which implies $\mathbb{P}(B) = 13/52$.

What is $\mathbb{P}(A \cap B)$? We realize that there is only one card of the 52 which is an Ace and a Club at the same time, namely, the Ace of Clubs. Therefore $\mathbb{P}(A \cap B) = 1/52$.

To find $\mathbb{P}(A \cup B)$ we may use the above with the General Addition Rule to get

$$\begin{aligned}
\mathbb{P}(A \cup B) &= \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \cap B), \\
&= 4/52 + 13/52 - 1/52, \\
&= 16/52.
\end{aligned}$$

Example 4.8. Staying with the deck of cards, let another random experiment be the selection of a five card stud poker hand, where "five card stud" means that we draw exactly five cards from the deck without replacement, no more, and no less. It turns out that the sample space S is so large and complicated that we will be obliged to settle for the trivial description $S = \{\text{all possible 5 card hands}\}$ for the time being. We will have a more precise description later.

What is $\mathbb{P}(\text{Royal Flush})$, or in other words, $\mathbb{P}(\text{A, K, Q, J, 10 all in the same suit})$?

It should be clear that there are only four possible royal flushes. Thus, if we could only count the number of outcomes in S then we could simply divide four by that number and we would have our answer under the equally likely model. This is the subject of Section 4.3.

How to do it with R

Probabilities are calculated in the `prob` package [77] with the `prob` function.

Consider the experiment of drawing a card from a standard deck of playing cards. Let's denote the probability space associated with the experiment as `S`, and let the subsets `A` and `B` be defined by the following:

```r
S <- cards(makespace = TRUE)
A <- subset(S, suit == "Heart")
B <- subset(S, rank %in% 7:9)
```

Now it is easy to calculate

```r
Prob(A)
```

```
[1] 0.25
```

Note that we can get the same answer with

```r
Prob(S, suit == "Heart")
```

```
[1] 0.25
```

We also find `Prob(B) = 0.23` (listed here approximately, but 12/52 actually). Internally, the `prob` function operates by summing the `probs` column of its argument. It will find subsets on-the-fly if desired.

We have as yet glossed over the details. More specifically, `prob` has three arguments: `x`, which is a probability space (or a subset of one), `event`, which is a logical expression used to define a subset, and `given`, which is described in Section 4.4.

WARNING. The `event` argument is used to define a subset of `x`, that is, the only outcomes used in the probability calculation will be those that are elements of `x` and satisfy `event` simultaneously. In other words, `Prob(x, event)` calculates `Prob(intersect(x, subset(x, event)))`.

Consequently, `x` should be the entire probability space in the case that `event` is non-null.

4.3 Counting Methods

The equally-likely model is a convenient and popular way to analyze random experiments. And when the equally likely model applies, finding the probability of an event A amounts to nothing more than counting the number of outcomes that A contains (together with the number of events in S). Hence, to be a master of probability one must be skilled at counting outcomes in events of all kinds.
Proposition 4.1 (The Multiplication Principle)**.** Suppose that an experiment is composed of two successive steps. Further suppose that the first step may be performed in n_1 distinct ways while the second step may be performed in n_2 distinct ways. Then the experiment may be performed in $n_1 n_2$ distinct ways. More generally, if the experiment is composed of k successive steps which may be performed in $n_1, n_2, \ldots, n_k$ distinct ways, respectively, then the experiment may be performed in $n_1 n_2 \cdots n_k$ distinct ways.
Example 4.9. We would like to order a pizza. It will be sure to have cheese (and marinara sauce), but we may elect to add one or more of the following five (5) available toppings:

pepperoni, sausage, anchovies, olives, and green peppers.

How many distinct pizzas are possible?

There are many ways to approach the problem, but the quickest avenue employs the Multiplication Principle directly. We will separate the action of ordering the pizza into a series of stages. At the first stage, we will decide whether or not to include pepperoni on the pizza (two possibilities). At the next stage, we will decide whether or not to include sausage on the pizza (again, two possibilities). We will continue in this fashion until at last we will decide whether or not to include green peppers on the pizza.

At each stage we will have had two options, or ways, to select a pizza to be made. The Multiplication Principle says that we should multiply the 2's to find the total number of possible pizzas: $2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 2^5 = 32$.

Example 4.10. We would like to buy a desktop computer to study statistics. We go to a website to build our computer our way. Given a line of products we have many options to customize our computer. In particular, there are 2 choices for a processor, 3 different operating systems, 4 levels of memory, 4 hard drives of differing sizes, and 10 choices for a monitor. How many possible types of computer must the company be prepared to build? **Answer:** $2 \cdot 3 \cdot 4 \cdot 4 \cdot 10 = 960$

4.3.1 Ordered Samples

Imagine a bag with n distinguishable balls inside. Now shake up the bag and select k balls at random. How many possible sequences might we observe?
Proposition 4.2. The number of ways in which one may select an ordered sample of k subjects from a population that has n distinguishable members is

- n^k if sampling is done with replacement,
- $n(n-1)(n-2)\cdots(n-k+1)$ if sampling is done without replacement.

Recall from calculus the notation for *factorials*:

$$\begin{aligned} 1! &= 1, \\ 2! &= 2 \cdot 1 = 2, \\ 3! &= 3 \cdot 2 \cdot 1 = 6, \\ &\vdots \\ n! &= n(n-1)(n-2)\cdots 3 \cdot 2 \cdot 1. \end{aligned}$$

Fact: The number of permutations of n elements is $n!$.
Example 4.11. Take a coin and flip it 7 times. How many sequences of Heads and Tails are possible? **Answer:** $2^7 = 128$.
Example 4.12. In a class of 20 students, we randomly select a class president, a class vice-president, and a treasurer. How many ways can this be done? **Answer:** $20 \cdot 19 \cdot 18 = 6840$.
Example 4.13. We rent five movies to watch over the span of two nights. We wish to watch 3 movies on the first night. How many distinct sequences of 3 movies could we possibly watch? *Answer:# $5 \cdot 4 \cdot 3 = 60$.

4.3.2 Unordered Samples

Proposition 4.3. The number of ways in which one may select an unordered sample of k subjects from a population that has n distinguishable members is

- $(n-1+k)!/[(n-1)!k!]$ if sampling is done with replacement,
- $n!/[k!(n-k)!]$ if sampling is done without replacement.

The quantity $n!/[k!(n-k)!]$ is called a *binomial coefficient* and plays a special role in mathematics; it is denoted

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{4.8}$$

and is read “n choose k”.
Example 4.14. You rent five movies to watch over the span of two nights, but only wish to watch 3 movies the first night. Your friend, Fred, wishes to borrow some movies to watch at his house on the first night. You owe Fred a favor, and allow him to select 2 movies from the set of 5. How many choices does Fred have? **Answer:** $\binom{5}{2} = 10$.
Example 4.15. Place 3 six-sided dice into a cup. Next, shake the cup well and pour out the dice. How many distinct rolls are possible? *Answer:* $(6-1+3)!/[(6-1)!3!] = \binom{8}{5} = 56$.

How to do it with R

The factorial $n!$ is computed with the command `factorial(n)` and the binomial coefficient $\binom{n}{k}$ with the command `choose(n,k)`.

The sample spaces we have computed so far have been relatively small, and we can visually study them without much trouble. However, it is *very* easy to generate sample spaces that are prohibitively large. And while R is wonderful and powerful and does almost everything except wash windows, even R has limits of which we should be mindful.

But we often do not need to actually generate the sample space; it suffices to count the number of outcomes. The `nsamp` function will calculate the number of rows in a sample space made by `urnsamples` without actually devoting the memory resources necessary to generate the space. The arguments are `n`, the number of (distinguishable) objects in the urn, `k`, the sample size, and `replace`, `ordered`, as above.

(ref:tab-sampling-k-from-n)

Table 4.1: Sampling k from n objects with `urnsamples`.

	`ordered = TRUE`	`ordered = FALSE`
`replace = TRUE`	n^k	$(n-1+k)!/[(n-1)!k!]$
`replace = FALSE`	$n!/(n-k)!$	$\binom{n}{k}$

Example 4.16. We will compute the number of outcomes for each of the four `urnsamples` examples that we saw in Example 3.2. Recall that we took a sample of size two from an urn with three distinguishable elements.

```
nsamp(n=3, k=2, replace = TRUE, ordered = TRUE)
```

```
[1] 9
```

```
nsamp(n=3, k=2, replace = FALSE, ordered = TRUE)
```

```
[1] 6
```

```r
nsamp(n=3, k=2, replace = FALSE, ordered = FALSE)
```

```
[1] 3
```

```r
nsamp(n=3, k=2, replace = TRUE, ordered = FALSE)
```

```
[1] 6
```

Compare these answers with the length of the data frames generated above.

The Multiplication Principle

A benefit of `nsamp` is that it is *vectorized* so that entering vectors instead of numbers for `n`, `k`, `replace`, and `ordered` results in a vector of corresponding answers. This becomes particularly convenient for combinatorics problems.

Example 4.17. There are 11 artists who each submit a portfolio containing 7 paintings for competition in an art exhibition. Unfortunately, the gallery director only has space in the winners' section to accommodate 12 paintings in a row equally spread over three consecutive walls. The director decides to give the first, second, and third place winners each a wall to display the work of their choice. The walls boast 31 separate lighting options apiece. How many displays are possible?

Answer: The judges will pick 3 (ranked) winners out of 11 (with `rep = FALSE`, `ord = TRUE`). Each artist will select 4 of his/her paintings from 7 for display in a row (`rep = FALSE`, `ord = TRUE`), and lastly, each of the 3 walls has 31 lighting possibilities (`rep = TRUE`, `ord = TRUE`). These three numbers can be calculated quickly with

```r
n <- c(11,7,31)
k <- c(3,4,3)
r <- c(FALSE,FALSE,TRUE)
```

```r
x <- nsamp(n, k, rep = r, ord = TRUE)
```

(Notice that `ordered` is always `TRUE`; `nsamp` will recycle `ordered` and `replace` to the appropriate length.) By the Multiplication Principle, the number of ways to complete the experiment is the product of the entries of `x`:

```r
prod(x)
```

```
[1] 24774195600
```

Compare this with the some other ways to compute the same thing:

```r
(11*10*9)*(7*6*5*4)*31^3
```

```
[1] 24774195600
```

or alternatively

```r
prod(9:11)*prod(4:7)*31^3
```

```
[1] 24774195600
```

or even

```
prod(factorial(c(11,7))/factorial(c(8,3)))*31^3
```

```
[1] 24774195600
```

As one can guess, in many of the standard counting problems there aren't substantial savings in the amount of typing; it is about the same using `nsamp` versus `factorial` and `choose`. But the virtue of `nsamp` lies in its collecting the relevant counting formulas in a one-stop shop. Ultimately, it is up to the user to choose the method that works best for him/herself.
Example 4.18 (The Birthday Problem)**.** Suppose that there are n people together in a room. Each person announces the date of his/her birthday in turn. The question is: what is the probability of at least one match? If we let the event A represent

$$A = \{\text{there is at least one match}\},$$

then we are looking for $\mathbb{P}(A)$, but as we soon will see, it will be more convenient for us to calculate $\mathbb{P}(A^c)$.

For starters we will ignore leap years and assume that there are only 365 days in a year. Second, we will assume that births are equally distributed over the course of a year (which is not true due to all sorts of complications such as hospital delivery schedules). See http://en.wikipedia.org/wiki/Birthday_problem for more.

Let us next think about the sample space. There are 365 possibilities for the first person's birthday, 365 possibilities for the second, and so forth. The total number of possible birthday sequences is therefore $\#(S) = 365^n$.

Now we will use the complementation trick we saw in Example 4.5. We realize that the only situation in which A does *not* occur is if there are *no* matches among all people in the room, that is, only when everybody's birthday is different, so

$$\mathbb{P}(A) = 1 - \mathbb{P}(A^c) = 1 - \frac{\#(A^c)}{\#(S)},$$

since the outcomes are equally likely. Let us then suppose that there are no matches. The first person has one of 365 possible birthdays. The second person must not match the first, thus, the second person has only 364 available birthdays from which to choose. Similarly, the third person has only 363 possible birthdays, and so forth, until we reach the n^{th} person, who has only $365 - n + 1$ remaining possible days for a birthday. By the Multiplication Principle, we have $\#(A^c) = 365 \cdot 364 \cdots (365 - n + 1)$, and

$$\mathbb{P}(A) = 1 - \frac{365 \cdot 364 \cdots (365 - n + 1)}{365^n} = 1 - \frac{364}{365} \cdot \frac{363}{365} \cdots \frac{(365 - n + 1)}{365}. \tag{4.9}$$

As a surprising consequence, consider this: how many people does it take to be in the room so that the probability of at least one match is at least 0.50? Clearly, if there is only $n = 1$ person in the room then the probability of a match is zero, and when there are $n = 366$ people in the room there is a 100% chance of a match (recall that we are ignoring leap years). So how many people does it take so that there is an equal chance of a match and no match?

When I have asked this question to students, the usual response is "somewhere around $n = 180$ people" in the room. The reasoning seems to be that in order to get a 50% chance of a match, there should be 50% of the available days to be occupied. The number of students in a typical classroom is 25, so as a companion question I ask students to estimate the probability of a match when there are $n = 25$ students in the room. Common estimates are a 1%, or 0.5%, or even 0.1% chance of a match. After they have given their estimates, we go around the room and each student announces their birthday. More often than not, we observe a match in the class, to the students' disbelief.

Students are usually surprised to hear that, using the formula above, one needs only $n = 23$ students to have a greater than 50% chance of at least one match. Figure 4.1 shows a graph of the birthday probabilities:

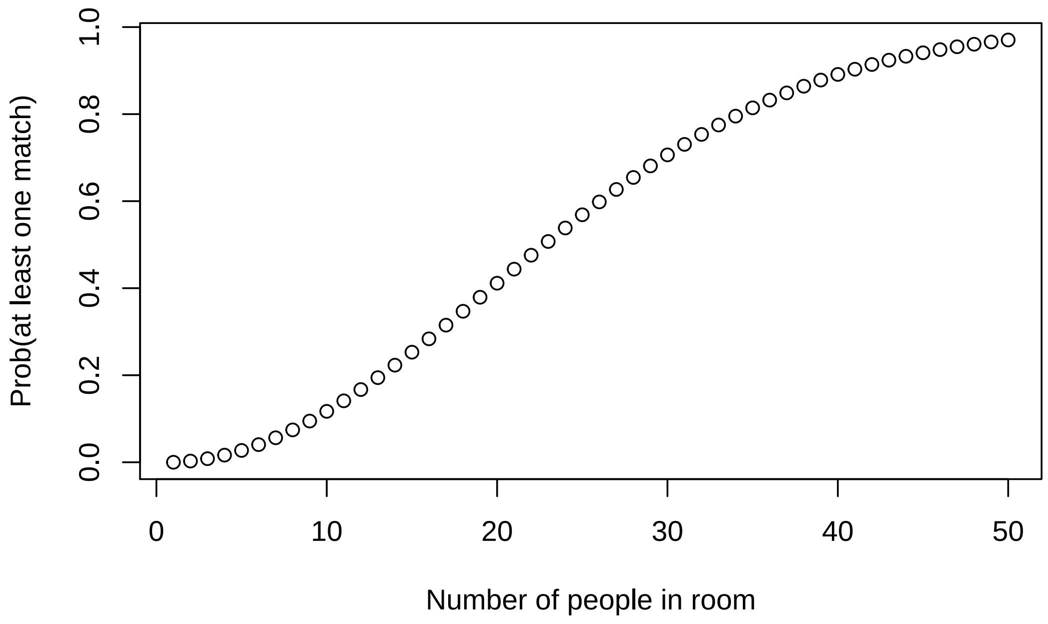

Figure 4.1: The birthday problem. The horizontal line is at $p = 0.50$ and the vertical line is at $n = 23$.

How to do it with R

We can make the plot in Figure 4.1 with the following sequence of commands.

```r
library(RcmdrPlugin.IPSUR)
g <- Vectorize(pbirthday.ipsur)
plot(1:50, g(1:50), xlab = "Number of people in room",
  ylab = "Prob(at least one match)" )
abline(h = 0.5)
abline(v = 23, lty = 2)
remove(g)
```

There is a `Birthday problem` item in the `Probability` menu of `RcmdrPlugin.IPSUR`. In the base R version, one can compute approximate probabilities for the more general case of probabilities other than 1/2, for differing total number of days in the year, and even for more than two matches.

4.4 Conditional Probability

Consider a full deck of 52 standard playing cards. Now select two cards from the deck, in succession. Let $A = \{\text{first card drawn is an Ace}\}$ and $B = \{\text{second card drawn is an Ace}\}$. Since there are four Aces in the deck, it is natural to assign $\mathbb{P}(A) = 4/52$. Suppose we look at the first card. What now is the probability of B? Of course, the answer depends on the value of the first card. If the first card is an Ace, then the probability that the second also is an Ace should be $3/51$, but if the first card is not an Ace, then the probability that the second is an Ace should be $4/51$. As notation for these two situations we write

$$\mathbb{P}(B|A) = 3/51, \quad \mathbb{P}(B|A^c) = 4/51.$$

Definition 4.1. The conditional probability of B given A, denoted $\mathbb{P}(B|A)$, is defined by

$$\mathbb{P}(B|A) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(A)}, \quad \text{if } \mathbb{P}(A) > 0. \tag{4.10}$$

We will not be discussing a conditional probability of B given A when $\mathbb{P}(A) = 0$, even though this theory exists, is well developed, and forms the foundation for the study of stochastic processes[1].

Toss a coin twice. The sample space is given by $S = \{HH,\ HT,\ TH,\ TT\}$. Let $A = \{\text{a head occurs}\}$ and $B = \{\text{a head and tail occur}\}$. It should be clear that $\mathbb{P}(A) = 3/4$, $\mathbb{P}(B) = 2/4$, and $\mathbb{P}(A \cap B) = 2/4$. What now are the probabilities $\mathbb{P}(A|B)$ and $\mathbb{P}(B|A)$?

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)} = \frac{2/4}{2/4} = 1,$$

in other words, once we know that a Head and Tail occur, we may be certain that a Head occurs. Next

$$\mathbb{P}(B|A) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(A)} = \frac{2/4}{3/4} = \frac{2}{3},$$

which means that given the information that a Head has occurred, we no longer need to account for the outcome TT, and the remaining three outcomes are equally likely with exactly two outcomes lying in the set B.

Example 4.19. Toss a six-sided die twice. The sample space consists of all ordered pairs (i, j) of the numbers $1, 2, \ldots, 6$, that is, $S = \{(1, 1),\ (1, 2), \ldots, (6, 6)\}$. We know from Section 4.3 that $\#(S) = 6^2 = 36$. Let $A = \{\text{outcomes match}\}$ and $B = \{\text{sum of outcomes at least 8}\}$. The sample space may be represented by a matrix:

[1] Conditional probability in this case is defined by means of *conditional expectation*, a topic that is well beyond the scope of this text. The interested reader should consult an advanced text on probability theory, such as Billingsley, Resnick, or Ash Dooleans-Dade.

```
A <- rolldie(2)
B <- subset(A, X1==X2)
C <- subset(A, X1+X2 > 7)
B$lab <- rep("X", dim(B)[1])
C$lab <- rep("O", dim(C)[1])
p <- ggplot(rbind(B, C), aes(x=X1, y=X2, label=lab))
p + geom_text(size = 15) + xlab("First roll") + ylab("Second roll")
```

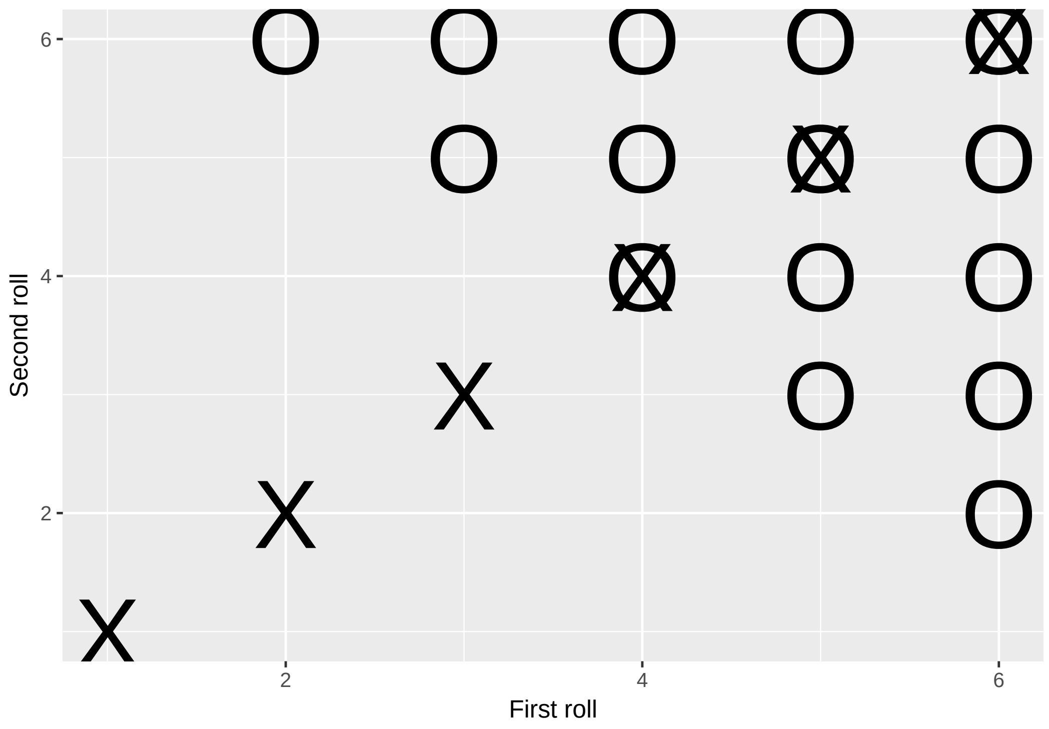

Figure 4.2: Rolling two dice. The outcomes in A are marked with X, the outcomes in B are marked with O.

The outcomes lying in the event A are marked with the symbol "X", the outcomes falling in B are marked with "O", and the outcomes in $A \cap B$ are those where the letters overlap. Now it is clear that $\mathbb{P}(A) = 6/36$, $\mathbb{P}(B) = 15/36$, and $\mathbb{P}(A \cap B) = 3/36$. Finally,

$$\mathbb{P}(A|B) = \frac{3/36}{15/36} = \frac{1}{5}, \quad \mathbb{P}(B|A) = \frac{3/36}{6/36} = \frac{1}{2}.$$

Again, we see that given the knowledge that B occurred (the 15 outcomes in the upper right triangle), there are 3 of the 15 that fall into the set A, thus the probability is 3/15. Similarly, given that A occurred (we are on the diagonal), there are 3 out of 6 outcomes that also fall in B, thus, the probability of B given A is 1/2.

4.4.1 How to do it with R

Continuing with Example 4.19, the first thing to do is set up the probability space with the `rolldie` function.

```
S <- rolldie(2, makespace = TRUE)  # assumes ELM
head(S)                            #  first few rows
```

```
  X1 X2      probs
1  1  1 0.02777778
2  2  1 0.02777778
3  3  1 0.02777778
4  4  1 0.02777778
5  5  1 0.02777778
6  6  1 0.02777778
```

Next we define the events

```
A <- subset(S, X1 == X2)
B <- subset(S, X1 + X2 >= 8)
```

And now we are ready to calculate probabilities. To do conditional probability, we use the `given` argument of the `prob` function:

```
Prob(A, given = B)
```

```
[1] 0.2
```

```
Prob(B, given = A)
```

```
[1] 0.5
```

Note that we do not actually need to define the events A and B separately as long as we reference the original probability space S as the first argument of the `prob` calculation:

```
Prob(S, X1==X2, given = (X1 + X2 >= 8) )
```

```
[1] 0.2
```

```
Prob(S, X1+X2 >= 8, given = (X1==X2) )
```

```
[1] 0.5
```

4.4.2 Properties and Rules

The following theorem establishes that conditional probabilities behave just like regular probabilities when the conditioned event is fixed.
Theorem 4.1. For any fixed event A with $\mathbb{P}(A) > 0$,

1. $\mathbb{P}(B|A) \geq 0$, for all events $B \subset S$,
2. $\mathbb{P}(S|A) = 1$, and

3. If $B_1, B_2, B_3, \ldots$ are disjoint events, then

$$\mathbb{P}\left(\bigcup_{k=1}^{\infty} B_k \middle| A\right) = \sum_{k=1}^{\infty} \mathbb{P}(B_k|A). \tag{4.11}$$

In other words, $\mathbb{P}(\cdot|A)$ is a legitimate probability function. With this fact in mind, the following properties are immediate:
Proposition 4.4. For any events A, B, and C with $\mathbb{P}(A) > 0$, 1. $\mathbb{P}(B^c|A) = 1 - \mathbb{P}(B|A)$. 2. If $B \subset C$ then $\mathbb{P}(B|A) \leq \mathbb{P}(C|A)$. 3. $\mathbb{P}[(B \cup C)|A] = \mathbb{P}(B|A) + \mathbb{P}(C|A) - \mathbb{P}[(B \cap C|A)]$. 4. **The Multiplication Rule.** For any two events A and B,

$$\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B|A). \tag{4.12}$$

And more generally, for events $A_1, A_2, A_3, \ldots, A_n$,

$$\mathbb{P}(A_1 \cap A_2 \cap \cdots \cap A_n) = \mathbb{P}(A_1)\mathbb{P}(A_2|A_1) \cdots \mathbb{P}(A_n|A_1 \cap A_2 \cap \cdots \cap A_{n-1}). \tag{4.13}$$

The Multiplication Rule is very important because it allows us to find probabilities in random experiments that have a sequential structure, as the next example shows.
Example 4.20. At the beginning of the section we drew two cards from a standard playing deck. Now we may answer our original question, what is $\mathbb{P}(\text{both Aces})$?

$$\mathbb{P}(\text{both Aces}) = \mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B|A) = \frac{4}{52} \cdot \frac{3}{51} \approx 0.00452.$$

How to do it with R

Continuing Example 4.20, we set up the probability space by way of a three step process. First we employ the `cards` function to get a data frame `L` with two columns: `rank` and `suit`. Both columns are stored internally as factors with 13 and 4 levels, respectively.

Next we sample two cards randomly from the `L` data frame by way of the `urnsamples` function. It returns a list `M` which contains all possible pairs of rows from `L` (there are `choose(52,2)` of them). The sample space for this experiment is exactly the list `M`.

At long last we associate a probability model with the sample space. This is right down the `probspace` function's alley. It assumes the equally likely model by default. We call this result `N` which is an object of class `ps` – short for "probability space".

But do not be intimidated. The object `N` is nothing more than a list with two elements: `outcomes` and `probs`. The `outcomes` element is itself just another list, with `choose(52,2)` entries, each one a data frame with two rows which correspond to the pair of cards chosen. The `probs` element is just a vector with `choose(52,2)` entries all the same: `1/choose(52,2)`.

Putting all of this together we do

```
L <- cards()
M <- urnsamples(L, size = 2)
N <- probspace(M)
N[[1]][[1]];  N$probs[1]
```

```
  rank suit
1    2 Club
2    3 Club
```

```
[1] 0.0007541478
```

Now that we have the probability space `N` we are ready to do some probability. We use the `prob` function, just like before. The only trick is to specify the event of interest correctly, and recall that we were interested in $\mathbb{P}(\text{both Aces})$. But if the cards are both Aces then the `rank` of both cards should be `A`, which sounds like a job for the `all` function:

```
Prob(N, all(rank == "A"))
```

```
[1] 0.004524887
```

Note that this value matches what we found in Example 4.20, above. We could calculate all sorts of probabilities at this point; we are limited only by the complexity of the event's computer representation.
Example 4.21. Consider an urn with 10 balls inside, 7 of which are red and 3 of which are green. Select 3 balls successively from the urn. Let $A = \{1^{\text{st}} \text{ ball is red}\}$, $B = \{2^{\text{nd}} \text{ ball is red}\}$, and $C = \{3^{\text{rd}} \text{ ball is red}\}$. Then

$$\mathbb{P}(\text{all 3 balls are red}) = \mathbb{P}(A \cap B \cap C) = \frac{7}{10} \cdot \frac{6}{9} \cdot \frac{5}{8} \approx 0.2917.$$

How to do it with R

Example 4.21 is similar to Example 4.20, but it is even easier. We need to set up an urn (vector `L`) to hold the balls, we sample from `L` to get the sample space (data frame `M`), and we associate a probability vector (column `probs`) with the outcomes (rows of `M`) of the sample space. The final result is a probability space (an ordinary data frame `N`).

It is easier for us this time because our urn is a vector instead of a `cards()` data frame. Before there were two dimensions of information associated with the outcomes (rank and suit) but presently we have only one dimension (color).

```
L <- rep(c("red","green"), times = c(7,3))
M <- urnsamples(L, size = 3, replace = FALSE, ordered = TRUE)
N <- probspace(M)
```

Now let us think about how to set up the event {all 3 balls are red}. Rows of `N` that satisfy this condition have `X1` == ”`red`” & `X2` == ”`red`” & `X3` == ”`red`”, but there must be an easier way. Indeed, there is. The `isrep` function (short for “is repeated”) in the `prob` package was written for this purpose. The command `isrep(N,"red",3)` will test each row of `N` to see whether the value `red` appears 3 times. The result is exactly what we need to define an event with the `prob` function. Observe

```
Prob(N, isrep(N, "red", 3))
```

```
[1] 0.2916667
```

Note that this answer matches what we found in Example 4.21. Now let us try some other probability questions. What is the probability of getting two `red`'s?

```
Prob(N, isrep(N, "red", 2))
```

```
[1] 0.525
```

Note that the exact value is $21/40$; we will learn a quick way to compute this in Section **??**. What is the probability of observing `red`, then `green`, then `red`?

```
Prob(N, isin(N, c("red","green","red"), ordered = TRUE))
```

```
[1] 0.175
```

Note that the exact value is $7/40$ (do it with the Multiplication Rule). What is the probability of observing `red`, `green`, and `red`, in no particular order?

```
Prob(N, isin(N, c("red","green","red")))
```

```
[1] 0.525
```

We already knew this. It is the probability of observing two `red`'s, above.
Example 4.22. Consider two urns, the first with 5 red balls and 3 green balls, and the second with 2 red balls and 6 green balls. Your friend randomly selects one ball from the first urn and transfers it to the second urn, without disclosing the color of the ball. You select one ball from the second urn. What is the probability that the selected ball is red?

Let $A = \{\text{transferred ball is red}\}$ and $B = \{\text{selected ball is red}\}$. Write

$$\begin{aligned} B &= S \cap B \\ &= (A \cup A^c) \cap B \\ &= (A \cap B) \cup (A^c \cap B) \end{aligned}$$

and notice that $A \cap B$ and $A^c \cap B$ are disjoint. Therefore

$$\begin{aligned} \mathbb{P}(B) &= \mathbb{P}(A \cap B) + \mathbb{P}(A^c \cap B) \\ &= \mathbb{P}(A)\mathbb{P}(B|A) + \mathbb{P}(A^c)\mathbb{P}(B|A^c) \\ &= \frac{5}{8}\cdot\frac{3}{9} + \frac{3}{8}\cdot\frac{2}{9} \\ &= \frac{21}{72} \end{aligned}$$

(which is 7/24 in lowest terms).

Example 4.23. We saw the `RcmdrTestDrive` data set in Chapter **??** in which a two-way table of the smoking status versus the gender was

```
library(RcmdrPlugin.IPSUR)
data(RcmdrTestDrive)
.Table <- xtabs( ~ smoking + gender, data = RcmdrTestDrive)
addmargins(.Table) # Table with marginal distributions
```

```
           gender
smoking      Female Male Sum
  Nonsmoker      61   75 136
  Smoker          9   23  32
  Sum            70   98 168
```

If one person were selected at random from the data set, then we see from the two-way table that $\mathbb{P}(\text{Female}) = 70/168$ and $\mathbb{P}(\text{Smoker}) = 32/168$. Now suppose that one of the subjects quits smoking, but we do not know the person's gender. If we now select one nonsmoker at random, what would be $\mathbb{P}(\text{Female})$? This example is just like the last example, but with different labels. Let $A = \{\text{the quitter is a female}\}$ and $B = \{\text{selected nonsmoker is a female}\}$. Write

$$\begin{aligned} B &= S \cap B \\ &= (A \cup A^c) \cap B \\ &= (A \cap B) \cup (A^c \cap B) \end{aligned}$$

and notice that $A \cap B$ and $A^c \cap B$ are disjoint. Therefore

$$\begin{aligned} \mathbb{P}(B) &= \mathbb{P}(A \cap B) + \mathbb{P}(A^c \cap B), \\ &= \mathbb{P}(A)\mathbb{P}(B|A) + \mathbb{P}(A^c)\mathbb{P}(B|A^c), \\ &= \frac{9}{32} \cdot \frac{62}{137} + \frac{23}{32} \cdot \frac{76}{137}, \\ &= \frac{2306}{4384}, \end{aligned}$$

(which is 1153/2192 in lowest terms).

Using the same reasoning, we can return to the example from the beginning of the section and show that

$$\mathbb{P}(\{\text{second card is an Ace}\}) = 4/52.$$

4.5 Independent Events

Toss a coin twice. The sample space is $S = \{HH, HT, TH, TT\}$. We know that $\mathbb{P}(1^{\text{st}} \text{ toss is } H) = 2/4$, $\mathbb{P}(2^{\text{nd}} \text{ toss is } H) = 2/4$, and $\mathbb{P}(\text{both } H) = 1/4$. Then

$$\begin{aligned}\mathbb{P}(2^{\text{nd}} \text{ toss is } H \mid 1^{\text{st}} \text{ toss is } H) &= \frac{\mathbb{P}(\text{both } H)}{\mathbb{P}(1^{\text{st}} \text{ toss is } H)}, \\ &= \frac{1/4}{2/4}, \\ &= \mathbb{P}(2^{\text{nd}} \text{ toss is } H).\end{aligned}$$

Intuitively, this means that the information that the first toss is H has no bearing on the probability that the second toss is H. The coin does not remember the result of the first toss.
Definition 4.2. Events A and B are said to be *independent* if

$$\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B). \tag{4.14}$$

Otherwise, the events are said to be *dependent*.

The connection with the above example stems from the following. We know from Section 4.4 that when $\mathbb{P}(B) > 0$ we may write

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}. \tag{4.15}$$

In the case that A and B are independent, the numerator of the fraction factors so that $\mathbb{P}(B)$ cancels with the result:

$$\mathbb{P}(A|B) = \mathbb{P}(A) \text{ when } A, B \text{ are independent.} \tag{4.16}$$

The interpretation in the case of independence is that the information that the event B occurred does not influence the probability of the event A occurring. Similarly, $\mathbb{P}(B|A) = \mathbb{P}(B)$, and so the occurrence of the event A likewise does not affect the probability of event B. It may seem more natural to define A and B to be independent when $\mathbb{P}(A|B) = \mathbb{P}(A)$; however, the conditional probability $\mathbb{P}(A|B)$ is only defined when $\mathbb{P}(B) > 0$. Our definition is not limited by this restriction. It can be shown that when $\mathbb{P}(A),\ \mathbb{P}(B) > 0$ the two notions of independence are equivalent.
Proposition 4.5. If the events A and B are independent then

- A and B^c are independent,
- A^c and B are independent,
- A^c and B^c are independent.

Proof. Suppose that A and B are independent. We will show the second one; the others are similar. We need to show that

$$\mathbb{P}(A^c \cap B) = \mathbb{P}(A^c)\mathbb{P}(B).$$

To this end, note that the Multiplication Rule, Equation (4.12) implies

$$\begin{aligned}\mathbb{P}(A^c \cap B) &= \mathbb{P}(B)\mathbb{P}(A^c|B),\\ &= \mathbb{P}(B)[1 - \mathbb{P}(A|B)],\\ &= \mathbb{P}(B)\mathbb{P}(A^c).\end{aligned}$$

□

Definition 4.3. The events A, B, and C are *mutually independent* if the following four conditions are met:

$$\begin{aligned}\mathbb{P}(A \cap B) &= \mathbb{P}(A)\mathbb{P}(B),\\ \mathbb{P}(A \cap C) &= \mathbb{P}(A)\mathbb{P}(C),\\ \mathbb{P}(B \cap C) &= \mathbb{P}(B)\mathbb{P}(C),\end{aligned}$$

and

$$\mathbb{P}(A \cap B \cap C) = \mathbb{P}(A)\mathbb{P}(B)\mathbb{P}(C).$$

If only the first three conditions hold then A, B, and C are said to be independent *pairwise*. Note that pairwise independence is not the same as mutual independence when the number of events is larger than two.

We can now deduce the pattern for n events, $n > 3$. The events will be mutually independent only if they satisfy the product equality pairwise, then in groups of three, in groups of four, and so forth, up to all n events at once. For n events, there will be $2^n - n - 1$ equations that must be satisfied (see Exercise **??**). Although these requirements for a set of events to be mutually independent may seem stringent, the good news is that for most of the situations considered in this book the conditions will all be met (or at least we will suppose that they are).

Example 4.24. Toss ten coins. What is the probability of observing at least one Head?

Answer: Let $A_i = \{\text{the } i^{\text{th}} \text{ coin shows } H\},\ i = 1, 2, \ldots, 10$. Supposing that we toss the coins in such a way that they do not interfere with each other, this is one of the situations where all of the A_i may be considered mutually independent due to the nature of the tossing. Of course, the only way that there will not be at least one Head showing is if all tosses are Tails. Therefore,

$$\begin{aligned}\mathbb{P}(\text{at least one } H) &= 1 - \mathbb{P}(\text{all } T),\\ &= 1 - \mathbb{P}(A_1^c \cap A_2^c \cap \cdots \cap A_{10}^c),\\ &= 1 - \mathbb{P}(A_1^c)\mathbb{P}(A_2^c)\cdots\mathbb{P}(A_{10}^c),\\ &= 1 - \left(\frac{1}{2}\right)^{10},\end{aligned}$$

which is approximately 0.9990234.

4.5.1 How to do it with R

Example 4.25. Toss ten coins. What is the probability of observing at least one Head?

```
S <- tosscoin(10, makespace = TRUE)
A <- subset(S, isrep(S, vals = "T", nrep = 10))
1 - Prob(A)
```

```
[1] 0.9990234
```

Compare this answer to what we got in Example 4.24.

4.5.2 Independent, Repeated Experiments

Generalizing from above it is common to repeat a certain experiment multiple times under identical conditions and in an independent manner. We have seen many examples of this already: tossing a coin repeatedly, rolling a die or dice, *etc*.

The `iidspace` function was designed specifically for this situation. It has three arguments: `x`, which is a vector of outcomes, `ntrials`, which is an integer telling how many times to repeat the experiment, and `probs` to specify the probabilities of the outcomes of `x` in a single trial.
Example 4.26 (An unbalanced coin)**.** (Continued, see Example 4.2). It was easy enough to set up the probability space for one unbalanced toss, however, the situation becomes more complicated when there are many tosses involved. Clearly, the outcome HHH should not have the same probability as TTT, which should again not have the same probability as HTH. At the same time, there is symmetry in the experiment in that the coin does not remember the face it shows from toss to toss, and it is easy enough to toss the coin in a similar way repeatedly.

We may represent tossing our unbalanced coin three times with the following:

```
iidspace(c("H","T"), ntrials = 3, probs = c(0.7, 0.3))
```

```
  X1 X2 X3 probs
1  H  H  H 0.343
2  T  H  H 0.147
3  H  T  H 0.147
4  T  T  H 0.063
5  H  H  T 0.147
6  T  H  T 0.063
7  H  T  T 0.063
8  T  T  T 0.027
```

As expected, the outcome HHH has the largest probability, while TTT has the smallest. (Since the trials are independent, $\mathbb{P}(HHH) = 0.7^3$ and $\mathbb{P}(TTT) = 0.3^3$, *etc*.) Note that the result of the function call is a probability space, not a sample space (which we could construct already with the `tosscoin` or `urnsamples` functions). The same procedure could be used to model an unbalanced die or any other experiment that may be represented with a vector of possible outcomes.

Note that `iidspace` will assume `x` has equally likely outcomes if no `probs` argument is specified. Also note that the argument `x` is a *vector*, not a data frame. Something like : iidspace(tosscoin(1),...) would give an error.

4.6 Bayes' Rule

We mentioned the subjective view of probability in Section 4.1. In this section we introduce a rule that allows us to update our probabilities when new information becomes available.
Theorem 4.2 (Bayes' Rule). Let B_1, B_2, ..., B_n be mutually exclusive and exhaustive and let A be an event with $\mathbb{P}(A) > 0$. Then

$$\mathbb{P}(B_k|A) = \frac{\mathbb{P}(B_k)\mathbb{P}(A|B_k)}{\sum_{i=1}^{n}\mathbb{P}(B_i)\mathbb{P}(A|B_i)}, \quad k = 1, 2, \ldots, n. \tag{4.17}$$

Proof. The proof follows from looking at $\mathbb{P}(B_k \cap A)$ in two different ways. For simplicity, suppose that $P(B_k) > 0$ for all k. Then

$$\mathbb{P}(A)\mathbb{P}(B_k|A) = \mathbb{P}(B_k \cap A) = \mathbb{P}(B_k)\mathbb{P}(A|B_k).$$

Since $\mathbb{P}(A) > 0$ we may divide through to obtain

$$\mathbb{P}(B_k|A) = \frac{\mathbb{P}(B_k)\mathbb{P}(A|B_k)}{\mathbb{P}(A)}.$$

Now remembering that $\{B_k\}$ is a partition, the Theorem of Total Probability (Equation (4.7)) gives the denominator of the last expression to be

$$\mathbb{P}(A) = \sum_{k=1}^{n}\mathbb{P}(B_k \cap A) = \sum_{k=1}^{n}\mathbb{P}(B_k)\mathbb{P}(A|B_k).$$

$\square$

What does it mean? Usually in applications we are given (or know) *a priori* probabilities $\mathbb{P}(B_k)$. We go out and collect some data, which we represent by the event A. We want to know: how do we *update# $\mathbb{P}(B_k)$ to $\mathbb{P}(B_k|A)$? The answer: Bayes' Rule.
Example 4.27 (Misfiling Assistants)**.** In this problem, there are three assistants working at a company: Moe, Larry, and Curly. Their primary job duty is to file paperwork in the filing cabinet when papers become available. The three assistants have different work schedules:

Table 4.2: Misfiling assistants: workload.

	Moe	Larry	Curly
Workload	60%	30%	10%

That is, Moe works 60% of the time, Larry works 30% of the time, and Curly does the remaining 10%, and they file documents at approximately the same speed. Suppose a person were to select one of the documents from the cabinet at random. Let M be the event

$$M = \{\text{Moe filed the document}\}$$

and let L and C be the events that Larry and Curly, respectively, filed the document. What are these events' respective probabilities? In the absence of additional information, reasonable prior probabilities would just be

Table 4.3: Misfiling assistants: prior.

	Moe	Larry	Curly
Prior Probability	0.6	0.3	0.1

Now, the boss comes in one day, opens up the file cabinet, and selects a file at random. The boss discovers that the file has been misplaced. The boss is so angry at the mistake that (s)he threatens to fire the one who erred. The question is: who misplaced the file?

The boss decides to use probability to decide, and walks straight to the workload schedule. (S)he reasons that, since the three employees work at the same speed, the probability that a randomly selected file would have been filed by each one would be proportional to his workload. The boss notifies *Moe# that he has until the end of the day to empty his desk.

But Moe argues in his defense that the boss has ignored additional information. Moe's likelihood of having misfiled a document is smaller than Larry's and Curly's, since he is a diligent worker who pays close attention to his work. Moe admits that he works longer than the others, but he doesn't make as many mistakes as they do. Thus, Moe recommends that – before making a decision – the boss should update the probability (initially based on workload alone) to incorporate the likelihood of having observed a misfiled document.

And, as it turns out, the boss has information about Moe, Larry, and Curly's filing accuracy in the past (due to historical performance evaluations). The performance information may be represented by the following table:

Table 4.4: Misfiling assistants: misfile rate.

	Moe	Larry	Curly
Misfile Rate	0.003	0.007	0.010

In other words, on the average, Moe misfiles 0.3% of the documents he is supposed to file. Notice that Moe was correct: he is the most accurate filer, followed by Larry, and lastly Curly. If the boss were to make a decision based only on the worker's overall accuracy, then *Curly# should get the axe. But Curly hears this and interjects that he only works a short period during the day, and consequently makes mistakes only very rarely; there is only the tiniest chance that he misfiled this particular document.

The boss would like to use this updated information to update the probabilities for the three assistants, that is, (s)he wants to use the additional likelihood that the document was misfiled to update his/her beliefs about the likely culprit. Let A be the event that a document is misfiled. What the boss would like to know are the three probabilities

$$\mathbb{P}(M|A),\ \mathbb{P}(L|A),\ \text{and}\ \mathbb{P}(C|A).$$

We will show the calculation for $\mathbb{P}(M|A)$, the other two cases being similar. We use Bayes' Rule in the form

$$\mathbb{P}(M|A) = \frac{\mathbb{P}(M \cap A)}{\mathbb{P}(A)}.$$

Let's try to find $\mathbb{P}(M \cap A)$, which is just $\mathbb{P}(M) \cdot \mathbb{P}(A|M)$ by the Multiplication Rule. We already know $\mathbb{P}(M) = 0.6$ and $\mathbb{P}(A|M)$ is nothing more than Moe's misfile rate, given above to be $\mathbb{P}(A|M) = 0.003$. Thus, we compute

$$\mathbb{P}(M \cap A) = (0.6)(0.003) = 0.0018.$$

Using the same procedure we may calculate

$$\mathbb{P}(L \cap A) = 0.0021 \text{ and } \mathbb{P}(C \cap A) = 0.0010.$$

Now let's find the denominator, $\mathbb{P}(A)$. The key here is the notion that if a file is misplaced, then either Moe or Larry or Curly must have filed it; there is no one else around to do the misfiling. Further, these possibilities are mutually exclusive. We may use the Theorem of Total Probability (4.7) to write

$$\mathbb{P}(A) = \mathbb{P}(A \cap M) + \mathbb{P}(A \cap L) + \mathbb{P}(A \cap C).$$

Luckily, we have computed these above. Thus

$$\mathbb{P}(A) = 0.0018 + 0.0021 + 0.0010 = 0.0049.$$

Therefore, Bayes' Rule yields

$$\mathbb{P}(M|A) = \frac{0.0018}{0.0049} \approx 0.37.$$

This last quantity is called the posterior probability that Moe misfiled the document, since it incorporates the observed data that a randomly selected file was misplaced (which is governed by the misfile rate). We can use the same argument to calculate

Table 4.5: Misfiling assistants: posterior.

	Moe	Larry	Curly
Posterior Probability	≈ 0.37	≈ 0.43	≈ 0.20

The conclusion: **Larry** gets the axe. What is happening is an intricate interplay between the time on the job and the misfile rate. It is not obvious who the winner (or in this case, loser) will be, and the statistician needs to consult Bayes' Rule to determine the best course of action.

Example 4.28. Suppose the boss gets a change of heart and does not fire anybody. But the next day (s)he randomly selects another file and again finds it to be misplaced. To decide whom to fire now, the boss would use the same procedure, with one small change. (S)he would not use the prior probabilities 60%, 30%, and 10%; those are old news. Instead, she would replace the prior probabilities with the posterior probabilities just calculated. After the math she will have new posterior probabilities, updated even more from the day before.

In this way, probabilities found by Bayes' rule are always on the cutting edge, always updated with respect to the best information available at the time.

4.6.1 How to do it with R

There are not any special functions for Bayes' Rule in the `prob` package [77], but problems like the ones above are easy enough to do by hand.

Example 4.29 (Misfiling Assistants)**.** (Continued from Example 4.27). We store the prior probabilities and the likelihoods in vectors and go to town.

```
prior <- c(0.6, 0.3, 0.1)
like <- c(0.003, 0.007, 0.010)
post <- prior # like
post / sum(post)
```

```
[1] 0.6 0.3 0.1
```

Compare these answers with what we got in Example 4.27. We would replace `prior` with `post` in a future calculation. We could raise `like` to a power to see how the posterior is affected by future document mistakes. (Do you see why? Think back to Section 4.5.)

Example 4.30. Let us incorporate the posterior probability (`post`) information from the last example and suppose that the assistants misfile seven more documents. Using Bayes' Rule, what would the new posterior probabilities be?

```
newprior <- post
post <- newprior # like^7
post / sum(post)
```

```
[1] 0.6 0.3 0.1
```

We see that the individual with the highest probability of having misfiled all eight documents given the observed data is no longer Larry, but Curly.

There are two important points. First, we did not divide `post` by the sum of its entries until the very last step; we do not need to calculate it, and it will save us computing time to postpone normalization until absolutely necessary, namely, until we finally want to interpret them as probabilities.

Second, the reader might be wondering what the boss would get if (s)he skipped the intermediate step of calculating the posterior after only one misfiled document. What if she started from the *original* prior, then observed eight misfiled documents, and calculated the posterior? What would she get? It must be the same answer, of course.

```
fastpost <- prior # like^8
fastpost / sum(fastpost)
```

```
[1] 0.6 0.3 0.1
```

Compare this to what we got in Example 4.28.

4.7 Random Variables

We already know about experiments, sample spaces, and events. In this section, we are interested in a *number* that is associated with the experiment. We conduct a random experiment E and after learning the outcome ω in S we calculate a number X. That is, to each outcome ω in the sample space we associate a number $X(\omega) = x$.

Definition 4.4. A *random variable* X is a function $X : S \to \mathbb{R}$ that associates to each outcome $\omega \in S$ exactly one number $X(\omega) = x$.

We usually denote random variables by uppercase letters such as X, Y, and Z, and we denote their observed values by lowercase letters x, y, and z. Just as S is the set of all possible outcomes of E, we call the set of all possible values of X the *support* of X and denote it by S_X.

Example 4.31. Let E be the experiment of flipping a coin twice. We have seen that the sample space is $S = \{HH, HT, TH, TT\}$. Now define the random variable $X =$ the number of heads. That is, for example, $X(HH) = 2$, while $X(HT) = 1$. We may make a table of the possibilities:

(ref:tab-flip-coin-twice)

Table 4.6: Flipping a coin twice.

$\omega \in S$	HH	HT	TH	TT
$X(\omega) = x$	2	1	1	0

Taking a look at the second row of the table, we see that the support of X – the set of all numbers that X assumes – would be $S_X = \{0, 1, 2\}$.

Example 4.32. Let E be the experiment of flipping a coin repeatedly until observing a Head. The sample space would be $S = \{H, TH, TTH, TTTH, \ldots\}$. Now define the random variable $Y =$ the number of Tails before the first head. Then the support of Y would be $S_Y = \{0, 1, 2, \ldots\}$.

Example 4.33. Let E be the experiment of tossing a coin in the air, and define the random variable $Z =$ the time (in seconds) until the coin hits the ground. In this case, the sample space is inconvenient to describe. Yet the support of Z would be $(0, \infty)$. Of course, it is reasonable to suppose that the coin will return to Earth in a short amount of time; in practice, the set $(0, \infty)$ is admittedly too large. However, we will find that in many circumstances it is mathematically convenient to study the extended set rather than a restricted one.

There are important differences between the supports of X, Y, and Z. The support of X is a finite collection of elements that can be inspected all at once. And while the support of Y cannot be exhaustively written down, its elements can nevertheless be listed in a naturally ordered sequence.

Random variables with supports similar to those of X and Y are called *discrete random variables*. We study these in Chapter **??**.

In contrast, the support of Z is a continuous interval, containing all rational and irrational positive real numbers. For this reason[2], random variables with supports like Z are called *continuous random variables*, to be studied in Chapter **??**.

4.7.1 How to do it with R

The primary vessel for this task is the `addrv` function. There are two ways to use it, and we will describe both.

Supply a Defining Formula

The first method is based on the `transform` function. See `?transform`. The idea is to write a formula defining the random variable inside the function, and it will be added as a column to the data frame. As an example, let us roll a 4-sided die three times, and let us define the random variable $U = X1 - X2 + X3$.

```
S <- rolldie(3, nsides = 4, makespace = TRUE)
S <- addrv(S, U = X1-X2+X3)
```

Now let's take a look at the values of U. In the interest of space, we will only reproduce the first few rows of S (there are $4^3 = 64$ rows in total).

```
head(S)
```

```
  X1 X2 X3 U    probs
1  1  1  1 1 0.015625
2  2  1  1 2 0.015625
3  3  1  1 3 0.015625
4  4  1  1 4 0.015625
5  1  2  1 0 0.015625
6  2  2  1 1 0.015625
```

We see from the U column it is operating just like it should. We can now answer questions like

```
Prob(S, U > 6)
```

```
[1] 0.015625
```

Supply a Function

Sometimes we have a function laying around that we would like to apply to some of the outcome variables, but it is unfortunately tedious to write out the formula defining what the new variable

[2]This isn't really the reason, but it serves as an effective litmus test at the introductory level. See Billingsley or Resnick.

would be. The addrv function has an argument FUN specifically for this case. Its value should be a legitimate function from R, such as sum, mean, median, and so forth. Or, you can define your own function. Continuing the previous example, let's define $V = \max(X1, X2, X3)$ and $W = X1+X2+X3$.

```
S <- addrv(S, FUN = max, invars = c("X1","X2","X3"), name = "V")
S <- addrv(S, FUN = sum, invars = c("X1","X2","X3"), name = "W")
head(S)
```

```
  X1 X2 X3 U V W    probs
1  1  1  1 1 1 3 0.015625
2  2  1  1 2 2 4 0.015625
3  3  1  1 3 3 5 0.015625
4  4  1  1 4 4 6 0.015625
5  1  2  1 0 2 4 0.015625
6  2  2  1 1 2 5 0.015625
```

Notice that addrv has an invars argument to specify exactly to which columns one would like to apply the function FUN. If no input variables are specified, then addrv will apply FUN to all non-probs columns. Further, addrv has an optional argument name to give the new variable; this can be useful when adding several random variables to a probability space (as above). If not specified, the default name is X.

4.7.2 Marginal Distributions

As we can see above, often after adding a random variable V to a probability space one will find that V has values that are repeated, so that it becomes difficult to understand what the ultimate behavior of V actually is. We can use the marginal function to aggregate the rows of the sample space by values of V, all the while accumulating the probability associated with V's distinct values. Continuing our example from above, suppose we would like to focus entirely on the values and probabilities of $V = \max(X1, X2, X3)$.

```
marginal(S, vars = "V")
```

```
  V    probs
1 1 0.015625
2 2 0.109375
3 3 0.296875
4 4 0.578125
```

We could save the probability space of V in a data frame and study it further, if we wish. As a final remark, we can calculate the marginal distributions of multiple variables desired using the vars argument. For example, suppose we would like to examine the joint distribution of V and W.

```
marginal(S, vars = c("V", "W"))
```

```
   V  W    probs
1  1  3 0.015625
2  2  4 0.046875
```

```
3  2  5 0.046875
4  3  5 0.046875
5  2  6 0.015625
6  3  6 0.093750
7  4  6 0.046875
8  3  7 0.093750
9  4  7 0.093750
10 3  8 0.046875
11 4  8 0.140625
12 3  9 0.015625
13 4  9 0.140625
14 4 10 0.093750
15 4 11 0.046875
16 4 12 0.015625
```

Note that the default value of `vars` is the names of all columns except `probs`. This can be useful if there are duplicated rows in the probability space.

4.8 Exercises

Exercise. <> Prove the assertion given in the text: the number of conditions that the events A_1, A_2, $\ldots$, A_n must satisfy in order to be mutually independent is $2^n - n - 1$. (*Hint*: think about Pascal's triangle.)

Bibliography

[1] Daniel Adler and Duncan Murdoch. *rgl: 3D visualization device system (OpenGL)*. R package version 0.93.952. 2013. URL: http://CRAN.R-project.org/package=rgl.

[2] A. Agresti and B. A. Coull. “Approximate is better than "exact" for interval estimation of binomial proportions”. In: *The American Statistician* 52 (1998), pp. 119–126.

[3] Alan Agresti. *Categorical Data Analysis*. Wiley, 2002.

[4] Martin Maechler et al. *sfsmisc: Utilities from Seminar fuer Statistik ETH Zurich*. R package version 1.0-24. 2013. URL: http://CRAN.R-project.org/package=sfsmisc.

[5] Fortran code by Alan Genz and R code by Adelchi Azzalini. *mnormt: The multivariate normal and t distributions*. R package version 1.4-5. 2012. URL: http://CRAN.R-project.org/package=mnormt.

[6] Jim Albert. *LearnBayes: Functions for Learning Bayesian Inference*. R package version 2.12. 2012. URL: http://CRAN.R-project.org/package=LearnBayes.

[7] Dirk Eddelbuettel with contributions by Antoine Lucas et al. *digest: Create cryptographic hash digests of R objects*. R package version 0.6.3. 2013. URL: http://CRAN.R-project.org/package=digest.

[8] Tom M. Apostol. *Calculus*. Second. Vol. I. Wiley, 1967.

[9] Tom M. Apostol. *Calculus*. Second. Vol. II. Wiley, 1967.

[10] Robert B. Ash and Catherine Doleans-Dade. *Probability & Measure Theory*. Harcourt Academic Press, 2000.

[11] Douglas Bates and Martin Maechler. *Matrix: Sparse and Dense Matrix Classes and Methods*. R package version 1.0-12. 2013. URL: http://CRAN.R-project.org/package=Matrix.

[12] S original by Berwin A. Turlach R port by Andreas Weingessel <Andreas.Weingessel@ci.tuwien.ac.at>. *quadprog: Functions to solve Quadratic Programming Problems*. R package version 1.5-5. 2013. URL: http://CRAN.R-project.org/package=quadprog.

[13] Peter J. Bickel and Kjell A. Doksum. *Mathematical Statistics*. Vol. I. Prentice Hall, 2001.

[14] Patrick Billingsley. *Probability and Measure*. Wiley Interscience, 1995.

[15] Ben Bolker. *emdbook: Ecological Models and Data in R*. R package version 1.3.4. 2013.

[16] Bruce L. Bowerman, Richard O’Connell, and Anne Koehler. *Forecasting, Time Series, and Regression: An Applied Approach*. South-Western College Pub, 2004.

[17] P. J. Brockwell and R. A. Davis. *Time Series and Forecasting Methods*. Second. Springer, 1991, p. 555.

[18] Angelo Canty and B. D. Ripley. *boot: Bootstrap R (S-Plus) Functions*. R package version 1.3-9. 2013.

[19] Neal L. Carothers. *Real Analysis*. Cambridge University Press, 2000.

[20] George Casella and Roger L. Berger. *Statistical Inference*. Duxbury Press, 2002.

[21] Kung-Sik Chan and Brian Ripley. *TSA: Time Series Analysis*. R package version 1.01. 2012. URL: http://CRAN.R-project.org/package=TSA.

[22] Scott Chasalow. *combinat: combinatorics utilities*. R package version 0.0-8. 2012. URL: http://CRAN.R-project.org/package=combinat.

[23] S original by Chris Fraley et al. *fracdiff: Fractionally differenced ARIMA aka ARFIMA(p,d,q) models*. R package version 1.4-2. 2012. URL: http://CRAN.R-project.org/package=fracdiff.

[24] Erhan Cinlar. *Introduction to Stochastic Processes*. Prentice Hall, 1975.

[25] William S. Cleveland. *The Elements of Graphing Data*. Hobart Press, 1994.

[26] Yves Croissant and Giovanni Millo. "Panel Data Econometrics in R: The plm Package". In: *Journal of Statistical Software* 27.2 (2008). URL: http://www.jstatsoft.org/v27/i02/.

[27] David B. Dahl. *xtable: Export tables to LaTeX or HTML*. R package version 1.7-1. 2013. URL: http://CRAN.R-project.org/package=xtable.

[28] Peter Dalgaard. *Introductory Statistics with R*. Springer, 2008. URL: http://staff.pubhealth.ku.dk/~pd/ISwR.html.

[29] A. C. Davison and D. V. Hinkley. *Bootstrap Methods and Their Applications*. ISBN 0-521-57391-2. Cambridge: Cambridge University Press, 1997. URL: http://statwww.epfl.ch/davison/BMA/.

[30] A. C. Davison and D. V. Hinkley. *Bootstrap Methods and Their Applications*. Cambridge University Press, 1997.

[31] Thomas J. DiCiccio and Bradley Efron. "Bootstrap Confidence Intervals". In: *Statistical Science* 11 (1996), pp. 189–228.

[32] M Dowle, T Short, and S Lianoglou. *data.table: Extension of data.frame for fast indexing, fast ordered joins, fast assignment, fast grouping and list columns.* R package version 1.8.8. 2013. URL: http://CRAN.R-project.org/package=data.table.

[33] Richard Durrett. *Probability: Theory and Examples*. Duxbury Press, 1996.

[34] Rick Durrett. *Essentials of Stochastic Processes*. Springer, 1999.

[35] Christophe Dutang, Vincent Goulet, and Mathieu Pigeon. "actuar: An R Package for Actuarial Science". In: *Journal of Statistical Software* 25.7 (2008), p. 38. URL: http://www.jstatsoft.org/v25/i07.

[36] Dirk Eddelbuettel. *Seamless R and C++ Integration with Rcpp*. ISBN 978-1-4614-6867-7. New York: Springer, 2013.

[37] Dirk Eddelbuettel and Romain François. "Rcpp: Seamless R and C++ Integration". In: *Journal of Statistical Software* 40.8 (2011), pp. 1–18. URL: http://www.jstatsoft.org/v40/i08/.

[38] Dirk Eddelbuettel and Conrad Sanderson. “RcppArmadillo: Accelerating R with high-performance C++ linear algebra”. In: *Computational Statistics and Data Analysis* in press (2013). URL: http://dx.doi.org/10.1016/j.csda.2013.02.005.

[39] Brian Everitt. *An R and S-Plus Companion to Multivariate Analysis*. Springer, 2007.

[40] Gerald B. Folland. *Real Analysis: Modern Techniques and Their Applications*. Wiley, 1999.

[41] Thomas Lumley using Fortran code by Alan Miller. *leaps: regression subset selection*. R package version 2.9. 2009. URL: http://CRAN.R-project.org/package=leaps.

[42] John Fox. *An R and S Plus Companion to Applied Regression*. Sage, 2002.

[43] John Fox. *Applied Regression Analysis, Linear Models, and Related Methods*. Sage, 1997.

[44] John Fox. “Effect Displays in R for Generalised Linear Models”. In: *Journal of Statistical Software* 8.15 (2003), pp. 1–27. URL: http://www.jstatsoft.org/v08/i15/.

[45] John Fox. “The R Commander: A Basic Statistics Graphical User Interface to R”. In: *Journal of Statistical Software* 14.9 (2005), pp. 1–42. URL: http://www.jstatsoft.org/v14/i09.

[46] John Fox and Jangman Hong. “Effect Displays in R for Multinomial and Proportional-Odds Logit Models: Extensions to the effects Package”. In: *Journal of Statistical Software* 32.1 (2009), pp. 1–24. URL: http://www.jstatsoft.org/v32/i01/.

[47] John Fox, Zhenghua Nie, and Jarrett Byrnes. *sem: Structural Equation Models*. R package version 3.1-3. 2013. URL: http://CRAN.R-project.org/package=sem.

[48] John Fox and Sanford Weisberg. *An R Companion to Applied Regression*. Second. Thousand Oaks CA: Sage, 2011. URL: http://socserv.socsci.mcmaster.ca/jfox/Books/Companion.

[49] Michael Friendly. *Visualizing Categorical Data*. SAS Publishing, 2000.

[50] Andrew Gelman et al. *Bayesian Data Analysis*. CRC Press, 2004.

[51] Alan Genz and Frank Bretz. *Computation of Multivariate Normal and t Probabilities*. Lecture Notes in Statistics. Heidelberg: Springer-Verlag, 2009. ISBN: 978-3-642-01688-2.

[52] Alan Genz et al. *mvtnorm: Multivariate Normal and t Distributions*. R package version 0.9-9995. 2013. URL: http://CRAN.R-project.org/package=mvtnorm.

[53] Rob J Hyndman with contributions from George Athanasopoulos et al. *forecast: Forecasting functions for time series and linear models*. R package version 4.06. 2013. URL: http://CRAN.R-project.org/package=forecast.

[54] Gregor Gorjanc. “Using Sweave with LyX”. In: *R News* 1 (2008), pp. 2–9.

[55] Charles M. Grinstead and J. Laurie Snell. *Introduction to Probability*. American Mathematical Society, 1997. URL: http://www.dartmouth.edu/~chance/.

[56] Bettina Grün and Achim Zeileis. “Automatic Generation of Exams in R”. In: *Journal of Statistical Software* 29.10 (2009), pp. 1–14. URL: http://www.jstatsoft.org/v29/i10/.

[57] David Firth with contributions from Heather Turner. *relimp: Relative Contribution of Effects in a Regression Model*. R package version 1.0-3. 2011. URL: http://CRAN.R-project.org/package=relimp.

[58] Richard M. Heiberger. *HH: Statistical Analysis and Data Display: Heiberger and Holland*. R package version 2.3-37. 2013. URL: http://CRAN.R-project.org/package=HH.

[59] Richard M. Heiberger and Burt Holland. *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer, 2004. URL: http://astro.temple.edu/~rmh/HH/.

[60] Richard M. Heiberger and Erich Neuwirth. *R Through Excel: A Spreadsheet Interface for Statistics, Data Analysis, and Graphics*. Springer, 2009. URL: http://www.springer.com/statistics/computanional+statistics/book/978-1-4419-0051-7.

[61] Robert V. Hogg, Joseph W. McKean, and Allen T. Craig. *Introduction to Mathematical Statistics*. Pearson Prentice Hall, 2005.

[62] Robert V. Hogg and Elliot A. Tanis. *Probability and Statistical Inference*. Pearson Prentice Hall, 2006.

[63] Torsten Hothorn, Frank Bretz, and Peter Westfall. “Simultaneous Inference in General Parametric Models”. In: *Biometrical Journal* 50.3 (2008), pp. 346–363.

[64] Torsten Hothorn and Kurt Hornik. *exactRankTests: Exact Distributions for Rank and Permutation Tests*. R package version 0.8-25. 2013. URL: http://CRAN.R-project.org/package=exactRankTests.

[65] Torsten Hothorn, Friedrich Leisch, and Achim Zeileis. *modeltools: Tools and Classes for Statistical Models*. R package version 0.2-19. 2012. URL: http://CRAN.R-project.org/package=modeltools.

[66] Torsten Hothorn et al. “A Lego System for Conditional Inference”. In: *The American Statistician* 60.3 (2006), pp. 257–263.

[67] Torsten Hothorn et al. “Implementing a Class of Permutation Tests: The coin Package”. In: *Journal of Statistical Software* 28.8 (2008), pp. 1–23. URL: http://www.jstatsoft.org/v28/i08/.

[68] Ross Ihaka et al. *colorspace: Color Space Manipulation*. R package version 1.2-2. 2013. URL: http://CRAN.R-project.org/package=colorspace.

[69] Norman L. Johnson, Samuel Kotz, and N. Balakrishnan. *Continuous Univariate Distributions*. Second. Vol. 1. Wiley, 1994.

[70] Norman L. Johnson, Samuel Kotz, and N. Balakrishnan. *Continuous Univariate Distributions*. Second. Vol. 2. Wiley, 1995.

[71] Norman L. Johnson, Samuel Kotz, and N. Balakrishnan. *Discrete Multivariate Distributions*. Wiley, 1997.

[72] Norman L. Johnson, Samuel Kotz, and Adrienne W. Kemp. *Univariate Discrete Distributions*. Second. Wiley, 1993.

[73] Roger W. Johnson. *How Many Fish are in the Pond?* URL: http://www.rsscse.org.uk/ts/gtb/johnson3.pdf.

[74] Frank E Harrell Jr, with contributions from Charles Dupont, and many others. *Hmisc: Harrell Miscellaneous*. R package version 3.12-2. 2013. URL: http://CRAN.R-project.org/package=Hmisc.

[75] Louis Kates and Thomas Petzoldt. *proto: Prototype object-based programming*. R package version 0.3-10. 2012. URL: http://CRAN.R-project.org/package=proto.

[76] G. Jay Kerns. *IPSUR: Introduction to Probability and Statistics Using R*. R package version 1.4. 2012. URL: http://CRAN.R-project.org/package=IPSUR.

[77] G. Jay Kerns. *prob: Elementary Probability on Finite Sample Spaces*. R package version 0.9-2. 2009. URL: http://CRAN.R-project.org/package=prob.

[78] Samuel Kotz, N. Balakrishnan, and Norman L. Johnson. *Continuous Multivariate Distributions*. Second. Vol. 1: Models and Applications. Wiley, 2000.

[79] Duncan Temple Lang. *XML: Tools for parsing and generating XML within R and S-Plus.* R package version 3.98-1.1. 2013. URL: http://CRAN.R-project.org/package=XML.

[80] Michael Lavine. *Introduction to Statistical Thought*. Lavine, Michael, 2009. URL: http://www.math.umass.edu/~lavine/Book/book.html.

[81] Peter M. Lee. *Bayesian Statistics: An Introduction*. Wiley, 1997.

[82] E. L. Lehmann. *Testing Statistical Hypotheses*. Springer-Verlag, 1986.

[83] E. L. Lehmann and George Casella. *Theory of Point Estimation*. Springer, 1998.

[84] Jim Lemon and Philippe Grosjean. *prettyR: Pretty descriptive stats.* R package version 2.0-7. 2013. URL: http://CRAN.R-project.org/package=prettyR.

[85] Andy Liaw and Matthew Wiener. “Classification and Regression by randomForest”. In: *R News* 2.3 (2002), pp. 18–22. URL: http://CRAN.R-project.org/doc/Rnews/.

[86] Uwe Ligges. “Accessing the Sources”. In: *R News* 6 (2006), pp. 43–45.

[87] Uwe Ligges and Martin Mächler. “Scatterplot3d - an R Package for Visualizing Multivariate Data”. In: *Journal of Statistical Software* 8.11 (2003), pp. 1–20. URL: http://www.jstatsoft.org.

[88] Catherine Loader. *locfit: Local Regression, Likelihood and Density Estimation.* R package version 1.5-9.1. 2013. URL: http://CRAN.R-project.org/package=locfit.

[89] Thomas Lumley. *dichromat: Color Schemes for Dichromats*. R package version 2.0-0. 2013. URL: http://CRAN.R-project.org/package=dichromat.

[90] Martin Maechler et al. *cluster: Cluster Analysis Basics and Extensions*. R package version 1.14.4 — For new features, see the ’Changelog’ file (in the package source). 2013.

[91] Jan R. Magnus and Heinz Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. Wiley, 1999.

[92] John Maindonald and John Braun. *Data Analysis and Graphics Using R*. Cambridge University Press, 2003.

[93] John Maindonald and W. John Braun. *DAAG: Data Analysis And Graphics data and functions*. R package version 1.16. 2013. URL: http://CRAN.R-project.org/package=DAAG.

[94] David Meyer, Achim Zeileis, and Kurt Hornik. “The Strucplot Framework: Visualizing Multi-Way Contingency Tables with vcd”. In: *Journal of Statistical Software* 17.3 (2006), pp. 1–48. URL: http://www.jstatsoft.org/v17/i03/.

[95] David Meyer, Achim Zeileis, and Kurt Hornik. *vcd: Visualizing Categorical Data*. R package version 1.2-13. 2012.

[96] David Meyer et al. *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*. R package version 1.6-1. 2012. URL: http://CRAN.R-project.org/package=e1071.

[97] Ben Mezrich. *Bringing Down the House: The Inside Story of Six M.I.T. Students Who Took Vegas for Millions*. Free Press, 2003.

[98] Jeff Miller. *Earliest Known Uses of Some of the Words of Mathematics*. URL: http://jeff560.tripod.com/mathword.html.

[99] John Neter et al. *Applied Linear Regression Models*. Third. McGraw Hill, 1996.

[100] Erich Neuwirth. *RColorBrewer: ColorBrewer palettes*. R package version 1.0-5. 2011. URL: http://CRAN.R-project.org/package=RColorBrewer.

[101] Frederick Novomestky. *matrixcalc: Collection of functions for matrix calculations*. R package version 1.0-3. 2012. URL: http://CRAN.R-project.org/package=matrixcalc.

[102] Jari Oksanen et al. *vegan: Community Ecology Package*. R package version 2.0-8. 2013. URL: http://CRAN.R-project.org/package=vegan.

[103] Jose Pinheiro et al. *nlme: Linear and Nonlinear Mixed Effects Models*. R package version 3.1-110. 2013.

[104] Tony Plate and Richard Heiberger. *abind: Combine multi-dimensional arrays*. R package version 1.4-0. 2011. URL: http://CRAN.R-project.org/package=abind.

[105] R Core Team. *foreign: Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, dBase, ...* R package version 0.8-54. 2013. URL: http://CRAN.R-project.org/package=foreign.

[106] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[107] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[108] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[109] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[110] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[111] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[112] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[113] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[114] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[115] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[116] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[117] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[118] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[119] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[120] R Development Core Team. *R: A Language and Environment for Statistical Computing*. ISBN 3-900051-07-0. R Foundation for Statistical Computing. Vienna, Austria, 2009. URL: http://www.R-project.org.

[121] C. Radhakrishna Rao and Helge Toutenburg. *Linear Models: Least Squares and Alternatives*. Springer, 1999.

[122] Sidney I. Resnick. *A Probability Path*. Birkhauser, 1999.

[123] Brian Ripley and from 1999 to Oct 2002 Michael Lapsley. *RODBC: ODBC Database Access*. R package version 1.3-7. 2013. URL: http://CRAN.R-project.org/package=RODBC.

[124] Maria L. Rizzo. *Statistical Computing with R*. Chapman & Hall/CRC, 2008.

[125] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer, 2004.

[126] Kenneth A. Ross. *Elementary Calculus: The Theory of Calculus*. Springer, 1980.

[127] RStudio. *manipulate: Interactive Plots for RStudio*. R package version 0.97.551. 2011.

[128] RStudio Team. *RStudio: Integrated Development Environment for R*. RStudio, Inc. Boston, MA, 2012. URL: http://www.rstudio.com/.

[129] Peter Ruckdeschel. *startupmsg: Utilities for start-up messages*. R package version 0.8. 2013. URL: http://CRAN.R-project.org/package=startupmsg.

[130] Peter Ruckdeschel. *SweaveListingUtils: Utilities for Sweave together with TeX listings package*. R package version 0.6.1. 2013. URL: http://CRAN.R-project.org/package=SweaveListingUtils.

[131] P. Ruckdeschel et al. "S4 Classes for Distributions". English. In: *R News* 6.2 (May 2006), pp. 2–6. URL: http://www.uni-bayreuth.de/departments/math/org/mathe7/DISTR/distr.pdf.

[132] P. Ruckdeschel et al. "S4 Classes for Distributions". English. In: *R News* 6.2 (May 2006), pp. 2–6. URL: http://www.uni-bayreuth.de/departments/math/org/mathe7/DISTR/distr.pdf.

[133] P. Ruckdeschel et al. "S4 Classes for Distributions". English. In: *R News* 6.2 (May 2006), pp. 2–6. URL: http://www.uni-bayreuth.de/departments/math/org/mathe7/DISTR/distr.pdf.

[134] P. Ruckdeschel et al. *S4 Classes for Distributions—a manual for packages distr, distrSim, distrTEst, distrEx, distrMod, and distrTeach*. English. Tech. rep. Kaiserslautern, Germany: Fraunhofer ITWM, July 2008. URL: http://r-forge.r-project.org/plugins/scmsvn/viewcvs.php/*checkout*/pkg/distrDoc/inst/doc/distr.pdf?root=distr.

[135] Deepayan Sarkar. *Lattice: Multivariate Data Visualization with R*. ISBN 978-0-387-75968-5. New York: Springer, 2008. URL: http://lmdvr.r-forge.r-project.org.

[136] Deepayan Sarkar and Felix Andrews. *latticeExtra: Extra Graphical Utilities Based on Lattice*. R package version 0.6-24. 2012. URL: http://CRAN.R-project.org/package=latticeExtra.

[137] F. E. Satterthwaite. "An Approximate Distribution of Estimates of Variance Components". In: *Biometrics Bulletin* 2 (1946), pp. 110–114.

[138] Luca Scrucca. “qcc: an R package for quality control charting and statistical process control”. In: *R News* 4/1 (2004), pp. 11–17. URL: http://CRAN.R-project.org/doc/Rnews/.

[139] Robert J. Serfling. *Approximation Theorems of Mathematical Statistics*. Wiley, 1980.

[140] S. S. Shapiro and M. B. Wilk. “An analysis of variance test for normality (complete samples)”. In: *Biometrika* 52 (1965), pp. 591–611.

[141] Gavin L. Simpson. *permute: Functions for generating restricted permutations of data*. R package version 0.7-0. 2012. URL: http://CRAN.R-project.org/package=permute.

[142] Greg Snow. *TeachingDemos: Demonstrations for teaching and learning*. R package version 2.9. 2013. URL: http://CRAN.R-project.org/package=TeachingDemos.

[143] Karline Soetaert. *diagram: Functions for visualising simple graphs (networks), plotting flow diagrams*. R package version 1.6.1. 2013. URL: http://CRAN.R-project.org/package=diagram.

[144] Karline Soetaert. *shape: Functions for plotting graphical shapes, colors*. R package version 1.4.0. 2012. URL: http://CRAN.R-project.org/package=shape.

[145] Max Kuhn. Contributions from Steve Weston et al. *odfWeave: Sweave processing of Open Document Format (ODF) files*. R package version 0.8.2. 2012. URL: http://CRAN.R-project.org/package=odfWeave.

[146] James Stewart. *Calculus*. Thomson Brooks/Cole, 2008.

[147] Stephen M. Stigler. *The History of Statistics: The Measurement of Uncertainty before 1900*. Harvard University Press, 1986.

[148] Gilbert Strang. *Linear Algebra and Its Applications*. Harcourt, 1988.

[149] Barbara G. Tabachnick and Linda S. Fidell. *Using Multivariate Statistics*. Allyn and Bacon, 2006.

[150] Justin Talbot. *labeling: Axis Labeling*. R package version 0.2. 2013. URL: http://CRAN.R-project.org/package=labeling.

[151] Terry M. Therneau and Patricia M. Grambsch. *Modeling Survival Data: Extending the Cox Model*. New York: Springer, 2000. ISBN: 0-387-98784-3.

[152] G. Jay Kerns with contributions by Theophilius Boye, Tyler Drombosky, and adapted from the work of John Fox et al. *RcmdrPlugin.IPSUR: An IPSUR Plugin for the R Commander*. R package version 0.1-8. 2012. URL: http://CRAN.R-project.org/package=RcmdrPlugin.IPSUR.

[153] Terry Therneau. *bdsmatrix: Routines for Block Diagonal Symmetric matrices*. R package version 1.3-1. 2013. URL: http://CRAN.R-project.org/package=bdsmatrix.

[154] Terry M Therneau. *A Package for Survival Analysis in S*. R package version 2.37-4. 2013. URL: http://CRAN.R-project.org/package=survival.

[155] Terry Therneau, Beth Atkinson, and Brian Ripley. *rpart: Recursive Partitioning*. R package version 4.1-1. 2013.

[156] Luke Tierney. *codetools: Code Analysis Tools for R*. R package version 0.2-8. 2011. URL: http://CRAN.R-project.org/package=codetools.

[157] Luke Tierney. *tkrplot: TK Rplot*. R package version 0.0-23. 2011. URL: http://CRAN.R-project.org/package=tkrplot.

[158] Adrian Trapletti and Kurt Hornik. *tseries: Time Series Analysis and Computational Finance*. R package version 0.10-32. 2013. URL: http://CRAN.R-project.org/package=tseries.

[159] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Fourth. ISBN 0-387-95457-0. New York: Springer, 2002. URL: http://www.stats.ox.ac.uk/pub/MASS4.

[160] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Fourth. ISBN 0-387-95457-0. New York: Springer, 2002. URL: http://www.stats.ox.ac.uk/pub/MASS4.

[161] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Fourth. ISBN 0-387-95457-0. New York: Springer, 2002. URL: http://www.stats.ox.ac.uk/pub/MASS4.

[162] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Fourth. ISBN 0-387-95457-0. New York: Springer, 2002. URL: http://www.stats.ox.ac.uk/pub/MASS4.

[163] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Fourth. ISBN 0-387-95457-0. New York: Springer, 2002. URL: http://www.stats.ox.ac.uk/pub/MASS4.

[164] William N. Venables and David M. Smith. *An Introduction to R*. 2010. URL: http://www.r-project.org/Manuals.

[165] John Verzani. *Using R for Introductory Statistics*. CRC Press, 2005. URL: http://www.math.csi.cuny.edu/UsingR/.

[166] John Verzani. *UsingR: Data sets for the text "Using R for Introductory Statistics"*. R package version 0.1-18. 2012. URL: http://CRAN.R-project.org/package=UsingR.

[167] Matt Wand. *KernSmooth: Functions for kernel smoothing for Wand and Jones (1995)*. R package version 2.23-10. 2013. URL: http://CRAN.R-project.org/package=KernSmooth.

[168] B. L. Welch. "The generalization of "Student's" problem when several different population variances are involved". In: *Biometrika* 34 (1947), pp. 28–35.

[169] Charlotte Wickham. *munsell: Munsell colour system*. R package version 0.4.2. 2013. URL: http://CRAN.R-project.org/package=munsell.

[170] Hadley Wickham. *ggplot2: elegant graphics for data analysis*. Springer New York, 2009. ISBN: 978-0-387-98140-6. URL: http://had.co.nz/ggplot2/book.

[171] Hadley Wickham. *ggplot2: elegant graphics for data analysis*. Springer New York, 2009. ISBN: 978-0-387-98140-6. URL: http://had.co.nz/ggplot2/book.

[172] Hadley Wickham. *gtable: Arrange grobs in tables.* R package version 0.1.2. 2012. URL: http://CRAN.R-project.org/package=gtable.

[173] Hadley Wickham. "Reshaping Data with the reshape Package". In: *Journal of Statistical Software* 21.12 (2007), pp. 1–20. URL: http://www.jstatsoft.org/v21/i12/.

[174] Hadley Wickham. *scales: Scale functions for graphics.* R package version 0.2.3. 2012. URL: http://CRAN.R-project.org/package=scales.

[175] Hadley Wickham. *stringr: Make it easier to work with strings.* R package version 0.6.2. 2012. URL: http://CRAN.R-project.org/package=stringr.

[176] Hadley Wickham. "The Split-Apply-Combine Strategy for Data Analysis". In: *Journal of Statistical Software* 40.1 (2011), pp. 1–29. URL: http://www.jstatsoft.org/v40/i01/.

[177] Wickham and Hadley. “Reshaping data with the reshape package”. In: *Journal of Statistical Software* 21.12 (2007). URL: http://www.jstatsoft.org/v21/i12/paper.

[178] Peter Wolf and Uni Bielefeld. *aplpack: Another Plot PACKage: stem.leaf, bagplot, faces, spin3R, and some slider functions*. R package version 1.2.7. 2012. URL: http://CRAN.R-project.org/package=aplpack.

[179] S. N. Wood. “Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models”. In: *Journal of the Royal Statistical Society (B)* 73.1 (2011), pp. 3–36.

[180] S. N. Wood. “Modelling and smoothing parameter estimation with multiple quadratic penalties”. In: *Journal of the Royal Statistical Society (B)* 62.2 (2000), pp. 413–428.

[181] S. N. Wood. “Stable and efficient multiple smoothing parameter estimation for generalized additive models”. In: *Journal of the American Statistical Association* 99.467 (2004), pp. 673–686.

[182] S. N. Wood. “Thin-plate regression splines”. In: *Journal of the Royal Statistical Society (B)* 65.1 (2003), pp. 95–114.

[183] S.N Wood. *Generalized Additive Models: An Introduction with R*. Chapman and Hall/CRC, 2006.

[184] Achim Zeileis. “Econometric Computing with HC and HAC Covariance Matrix Estimators”. In: *Journal of Statistical Software* 11.10 (2004), pp. 1–17. URL: http://www.jstatsoft.org/v11/i10/.

[185] Achim Zeileis. “Object-oriented Computation of Sandwich Estimators”. In: *Journal of Statistical Software* 16.9 (2006), pp. 1–16. URL: http://www.jstatsoft.org/v16/i09/..

[186] Achim Zeileis and Yves Croissant. “Extended Model Formulas in R: Multiple Parts and Multiple Responses”. In: *Journal of Statistical Software* 34.1 (2010), pp. 1–13. URL: http://www.jstatsoft.org/v34/i01/.

[187] Achim Zeileis and Gabor Grothendieck. “zoo: S3 Infrastructure for Regular and Irregular Time Series”. In: *Journal of Statistical Software* 14.6 (2005), pp. 1–27. URL: http://www.jstatsoft.org/v14/i06/.

[188] Achim Zeileis, Kurt Hornik, and Paul Murrell. “Escaping RGBland: Selecting Colors for Statistical Graphics”. In: *Computational Statistics & Data Analysis* 53 (2009), pp. 3259–3270. DOI: 10.1016/j.csda.2008.11.033.

[189] Achim Zeileis and Torsten Hothorn. “Diagnostic Checking in Regression Relationships”. In: *R News* 2.3 (2002), pp. 7–10. URL: http://CRAN.R-project.org/doc/Rnews/.

[190] Achim Zeileis, David Meyer, and Kurt Hornik. “Residual-based Shadings for Visualizing (Conditional) Independence”. In: *Journal of Computational and Graphical Statistics* 16.3 (2007), pp. 507–525.

Index