StatR 101: Fall 2012
Homework 6 - Solutions
Eli Gurarie

1. **Card playing:** A standard playing deck of 52 cards consists of 13 unique valued cards (2 to 10, and four *face cards*, the Ace, Jack, Queen and King) each in 4 different *suits* (hearts - ♡, diamonds - ♢, clubs - ♣, and spades - ♠).

(a) *How many possible hands of five unique cards can be drawn from a deck of cards?*
Because the order does not matter, the answer is $\binom{52}{5}$ = 2,598,960

(b) *Write a function called* `IsFlush` *that draws five cards from a deck and determines (*`TRUE/FALSE`*) whether or not it is a flush.*
Almost every solution to this that I saw was unique, and almost everybody's worked correctly, though some were more cumbersome than others. Here is mine:

```
SuitDeck <- rep(1:4, each=13)
IsFlush <- function(Deck)
{
  Hand <- sample(SuitDeck, 5, replace=FALSE)
  min(Hand) == max(Hand)
}
```

Note that I used a deck that only has information on the suits.

(c) *Simulate this process 10,000 times and count how many times you draw a flush. What is the approximate probability of a flush based on this experiment? Note there are two ways that you can do this: with a loop, or by creating a matrix of decks and using* `apply()`*.*
Using a loop:

```
Flushes <- rep(NA, 1e5)
for(i in 1:1e5)
  Flushes[i] <- IsFlush(SuitDeck)
```

I get 192 flushes, or P(Flush) ≈ 0.00192. Running this code was pretty fast, but still noticeable (1.45 seconds on my machine). Interestingly, it was about twice as fast as my original code, which was to test `length(unique(Hand))==1`. My guess is the `unique` function is somewhat costly.
You can use the `apply()` function on a matrix of 1e5 complete decks as follows:

```
ManyDecks <- matrix(SuitDeck, ncol=1e5, nrow=52)
Flushes <- apply(ManyDecks, 2, IsFlush)
```

But that was surprisingly a little bit slower (1.6 seconds). I was always taught that "apply" was faster than looping, but apparently that is not always the case.[1].

(d) *Repeat exercises (b) and (c) for the straight, i.e. creating a function called* `IsStraight()` *and repeating it 10,000 times.*

Again, lots of possibilities for the straight function. My first attempt was this:

```
NumberDeck <- rep(1:13, 4)
IsStraight <- function(Deck)
{
  Hand <- sample(Deck, 5, replace=FALSE)
  sum(diff(sort(Hand))==1) == 4
}

Straights <- rep(NA, 1e5)
for(i in 1:1e5)
  Straights[i] <- IsStraight(NumberDeck)
```

But that took over 11 seconds! The following version:

```
IsStraight <- function(Deck)
{
  Hand <- sample(Deck, 5, replace=FALSE)
  (length(unique(Hand))==5) & ((max(Hand) - min(Hand) == 4))
}
```

took just about 4 seconds. Note that it first make sure that all the cards are unique, and then checks the range, which are sufficient conditions for a straight. I'm not sure if there's an even faster way to do this ... probably. The lesson here is that the more basic the functions you use, the faster the process is. Thus: `diff`, `min`, `max`, etc. I got, in a typical run, about 363 straights, i.e. $\Pr(\text{Straight}) \approx 0.00363$.

(e) *Based on these results, which is the more likely hand to be dealt in a game of poker?*

The straight is almost twice as likely. See derivations below.

Bonus*: Compute the exact probabilities of a flush and a straight. This is most easily done by counting the number of possible flushes and straights and dividing by the answer to problem (a). While you're at it, count the number of straight flushes (this, you can do with a friend on your fingers and toes).*

The number of possible flushes of a given suit is $\binom{13}{5}$, because you are drawing 5 of the 13 cards in that suit with no regard to order. There are four possible suits, so: $\text{N(flushes)} = 4 \times \binom{13}{5} = 5,148$.

[1]See a related on-line discussion here: http://stackoverflow.com/questions/5533246/why-is-apply-method-slower-than-a-for-loop-in-r

The probability of a flush is:

$$\text{P(flush)} = \frac{N(\text{flushes})}{N(\text{unique hands})} = \frac{4 \times \binom{13}{5}}{\binom{52}{5}} = 0.001980792. \tag{1}$$

The reasoning behind calculating the number of straights is a bit trickier. Within a single suit of cards there are 9 possible straights (i.e. everything from {2,3,4,5,6} to {10, J, Q, K, A} (10 if Ace is high and low). Within a single straight, each of those numbers can be of any suit, such that there $4 \times 4 \times 4 \times 4 \times 4 = 4^5$ configurations of suits for any given straight, and a total of $9 \times 4^5 = 9218$ different straight in a deck. Thus:

$$\text{P(straight)} = \frac{N(\text{straights})}{N(\text{unique hands})} = \frac{9 \times 4^5}{\binom{52}{5}} = 0.003546034. \tag{2}$$

There are just 36 straight flushes (9 in each suit).

2. **Airplane functioning:** *You are planning on taking a flight on Epsilon Airlines across the Pacific Ocean. To have a successful flight, 100 different components on the plane must ALL function correctly. Each of these components has a probability 0.001 of failing.*

(a) *Use the `rbinom()` command to simulate a flight on Epsilon airline by creating a vector of 100 elements which succeed or fail with the appropriate probability.*

`rbinom(100,1,.001)`

(b) *Use the `rbinom()` command in a slightly different to way to produce a vector of length 1,000,000, in which each element is the number of components that fail in each flight. Use this vector to produce an estimate of the probability of failure for the plane.*

```
>  SimFlight <- rbinom(1e6,100,.001)
>  N.failures <- sum(SimFlight > 0)
>  sum(N.failures)/1e6
[1] 0.095545
```

(c) *Calculate the exact probability that the flight will not be successful. What important assumption are you making in this calculation (and in the simulations above)?*

The probability of one or more components failing P(Failure) is 1 - P(all components succeeding). This probability is $0.999^{100} = 0.9047$. So the probability of failure is 0.0952. The fundamental assumption is that the success of each component is independent of the success of neighboring component, which, in mechanical systems, is generally a poor assumption.

3. **Elections:** *Let's imagine a slightly different (but no less strange) variation in which each elector for each state votes for a candidate with probability p equal to the proportion of votes each candidate receives. Consider the following (approximate) results from the 2008 election for three states:*

State	Electors	Obama (%)	McCain (%)
Alaska	3	39	61
Indiana	11	50.5	49.5
California	55	62	38

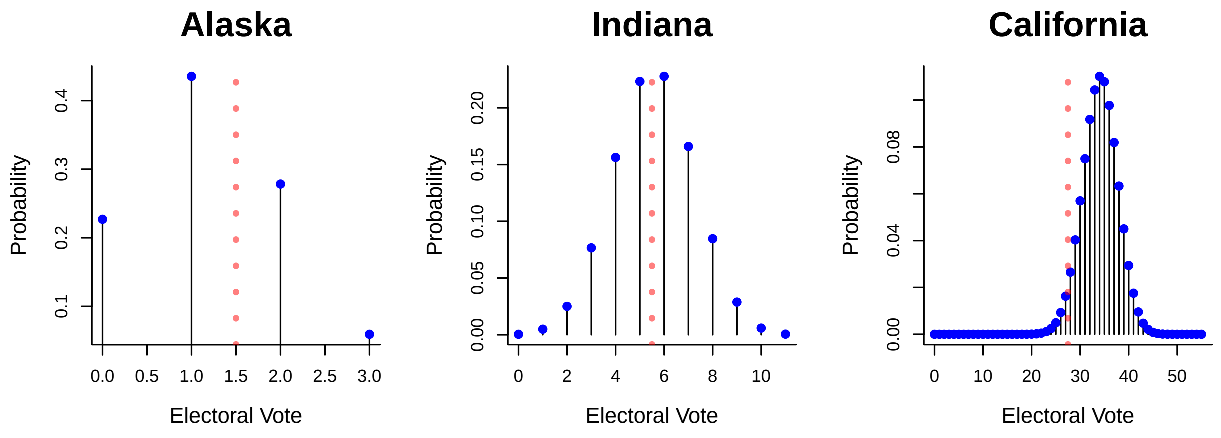
(a) *Illustrate, under this system, the discrete probability distribution of the number of electoral votes Obama would receive from each of these three states.*

Each of these processes is a binomial, i.e. each elector represents a single Bernoulli trial with probability p_i, and the number of electors is the size of the experiment. Thus, the probability mass functions of these distributions are:

```
pmf.Alaska <- dbinom(0:3,size=3,p=0.39)
pmf.Indiana <- dbinom(0:11,size=11,p=0.505)
pmf.California <- dbinom(0:55,size=55,p=0.62)
```

I wrote a generic piece of code to produce the plots below. Note the vertical bar represents the win-lose split (at 50%).

```
PlotCollegeVote <- function(size, p)
{
  plot(0:size, dbinom(0:size, size, p),
        xlab="Electoral Vote",
        ylab="Probability", type="h")
  points(0:size, dbinom(0:size, size, p), pch=19, col="blue")
  abline(v=size/2, lwd=4, lty=3, col=rgb(1,0,0,.5))
}
par(bty="l", mfrow=c(1,3), cex.lab=1.25, cex.main=2)
PlotCollegeVote(3, 0.39); title("Alaska")
PlotCollegeVote(11, 0.505); title("Indiana")
PlotCollegeVote(55, 0.62); title("California")
```



(b) *Calculate for each state the probability that McCain would have won each of these states according to this voting model.*

Here, you just need to sum the cumulative probability that the vote is less than 50 percent of the electoral delegation. Thus:

```
> sum(pmf.Alaska[0:3 < 3/2])
[1] 0.662338
> sum(pmf.Indiana[0:11 < 11/2])
[1] 0.4864671
> sum(pmf.California[0:55 < 55/2])
[1] 0.03479121
```

Note that the probability of a given elector voting a certain way does NOT equate to the probability of a candidate winning! In California, McCain has a very small probability of winning because at the large sample size, the relative width of the binomial distribution is narrower than, e.g., in Alaska. There was considerable confusion among the punditry in this last election cycle as to why, if a poll gives an apparently small advantage to a candidate (say, a 53%/47% split), why that equates to a very high (often over 90%) probability of the candidate with an edge taking an entire state (or nation).

Qualtitatively, at probabilities further from 0.5 and smaller sample sizes (e.g. Alaska), the binomial distribution is more skewed. The larger the sample size, the more symmetric it is, even at high values of p (e.g. California). This is again related to the great tug of the normal distribution, which we are gearing up to explore.