MATH 323: PROBABILITY
COMBINATORIAL PROBABILITY: POKER HANDS

We have the table of probabilities for poker hands as follows: recall that the 'scoring' cards are denoted x and y, the other cards are denoted a, b etc. The denominator in the probability calculation is

$$n_S = \binom{52}{5}.$$

The probabilities can be computed in `R` using the `choose` function:

```
nS<-choose(52,5)
n1<-choose(13,1)*choose(4,2)*choose(12,3)*choose(4,1)^3
n2<-choose(13,2)*choose(4,2)^2*choose(11,1)*choose(4,1)
n3<-choose(13,1)*choose(4,3)*choose(12,2)*choose(4,1)^2
n4<-choose(10,1)*choose(4,1)^5-choose(10,1)*choose(4,1)
n5<-choose(13,5)*choose(4,1)-choose(10,1)*choose(4,1)
n6<-choose(13,1)*choose(4,3)*choose(12,1)*choose(4,2)
n7<-choose(13,1)*choose(4,4)*choose(12,1)*choose(4,1)
n8<-10*choose(4,1)-choose(4,1)
n9<-choose(4,1)
n0<-nS-(n1+n2+n3+n4+n5+n6+n7+n8+n9)
round(c(n0,n1,n2,n3,n4,n5,n6,n7,n8,n9)/nS,7)

+  [1] 0.5011774 0.4225690 0.0475390 0.0211285 0.0039246 0.0019654 0.0014406
+  [8] 0.0002401 0.0000139 0.0000015

Probs<-c(n0,n1,n2,n3,n4,n5,n6,n7,n8,n9)/nS
```

Hand	Configuration	n_A	Prob.
ONE PAIR	$xxabc$	$\binom{13}{1}\binom{4}{2} \times \binom{12}{3}\binom{4}{1}\binom{4}{1}\binom{4}{1}$	0.422569
TWO PAIRS	$xxyya$	$\binom{13}{2}\binom{4}{2}\binom{4}{2} \times \binom{11}{1}\binom{4}{1}$	0.047539
THREE OF A KIND	$xxxab$	$\binom{13}{1}\binom{4}{3} \times \binom{12}{2}\binom{4}{1}\binom{4}{1}$	0.0211285
STRAIGHT	Run of five cards	$\binom{10}{1}\binom{4}{1}^5 - \binom{10}{1}\binom{4}{1}$	0.0039246
FLUSH	Five cards in same suit	$\binom{13}{5}\binom{4}{1} - \binom{10}{1}\binom{4}{1}$	0.0019654
FULL HOUSE	$xxxyy$	$\binom{13}{1}\binom{4}{3} \times \binom{12}{1}\binom{4}{2}$	0.0014406
FOUR OF A KIND	$xxxxa$	$\binom{13}{1}\binom{4}{4} \times \binom{12}{1}\binom{4}{1}$	0.0002401
STRAIGHT FLUSH	Run of five cards in same suit	$\binom{10}{1}\binom{4}{1} - \binom{4}{1}$	0.0000139
ROYAL FLUSH	AKQJ10 in any suit	$1\binom{4}{1}$	0.0000015

Table 1: Probabilities for different poker hands. For STRAIGHT and FLUSH, we must remember to subtract the STRAIGHT FLUSHes (which are counted separately); similarly, for STRAIGHT FLUSH, must remember to subtract ROYAL FLUSHes.

We can attempt to check these numbers using a simulation in the computer package `R`. For example, if we identify the numbers $\{1, 2, 3, \ldots, 52\}$ with the cards in a pack listed suit-wise (Hearts, Clubs, Diamonds, Spades) and within a suit, denomination-wise, so that

$$1 = \text{'Two of Hearts'} \qquad 2 = \text{'Three of Hearts'} \qquad \cdots \qquad 12 = \text{'King of Hearts'} \qquad 13 = \text{'Ace of Hearts'}$$

then

$$14 = \text{'Two of Clubs'} \qquad 15 = \text{'Three of Clubs'} \qquad \cdots \qquad 25 = \text{'King of Clubs'} \qquad 26 = \text{'Ace of Clubs'}$$

etc., we can simulate a hand of give cards using the `sample` function. Thus the suit is decided by assessing which quartile of the range the selected number lies, and the denomination is decided by writing the selected number modulo 13 (and following the convention that 13 mod 13 = 13, unlike the usual rule in modulo arithmetic).

A small simulated example is as follows:

```
set.seed(2101)                          #Set the random number generator seed value
suit.names<-c('H','C','D','S')
denomination.names<-c(as.character(2:10),'J','Q','K','A')
suit.list<-rep(suit.names,each=13)
denomination.list<-rep(denomination.names,4)
selected.hand<-sample(1:52,size=5)
denomination<-denomination.list[selected.hand]
suit<-suit.list[selected.hand]
denomination;suit

+ [1] "8" "2" "5" "7" "8"
+ [1] "H" "H" "D" "S" "D"
```

Thus the hand drawn is

(`9 of Spades, Queen of Spades, 9 of Clubs, 4 of Diamonds, King of Diamonds`)

which is a PAIR (two nines). To decide upon which hand has been dealt, we need to carry out some basic manipulations and calculations.

```
denomination.count<-rep(0,52)
denomination.count[selected.hand]<-1
denomination.count<-apply(matrix(denomination.count,nrow=13,ncol=4),1,sum)

if(max(denomination.count) == 4){
    print('4 of a kind !')
}else if(max(denomination.count) == 3){
    if(max(denomination.count[denomination.count != 3]) == 2){
        print('Full house !')
    }else{
        print('3 of a kind !')
    }
}else if(max(denomination.count) == 2){
    if(sum(denomination.count == 2) == 2){
        print('Two pairs !')
    }else{
        print('Pair !')
    }
}else if(max(table(suit)) == 5){
    denom.vals<-(selected.hand %% 13)
    denom.vals[denom.vals==0]<-13
    print(denom.vals)
    if(max(denom.vals)-min(denom.vals) == 4){
        if(max(denom.vals) == 13){
            print('Royal Flush !')
        }else{
            print('Straight Flush !')
```

```
        }
    }else{
        print('Flush !')
    }
}else{
    denom.vals<-(selected.hand %% 13)
    denom.vals[denom.vals==0]<-13
    if(max(denom.vals)-min(denom.vals) == 4){
        print('Straight !')
    }
}

+ [1] "Pair !"
```

This series of calculations return the result that the simulated hand is a `Pair`.

We can check the computed probabilities from the table by carrying out a large simulation study, drawing 10 million hands, say, and counting up the number of occurrences of each hand type. First, we can write a small function in `R` to take a simulated hand and determine what the hand type is.

```
hand.type<-function(selected.hand){

    denomination<-denomination.list[selected.hand]
    suit<-suit.list[selected.hand]
    denomination.count<-rep(0,52)
    denomination.count[selected.hand]<-1
    denomination.count<-apply(matrix(denomination.count,nrow=13,ncol=4),1,sum)

    hand.type<-0
    if(max(denomination.count) == 4){
        #print('4 of a kind !')
        hand.type<-7
    }else if(max(denomination.count) == 3){
        if(max(denomination.count[denomination.count != 3]) == 2){
            #print('Full house !')
            hand.type<-6
        }else{
            #print('3 of a kind !')
            hand.type<-3
        }
    }else if(max(denomination.count) == 2){
        if(sum(denomination.count == 2) == 2){
            #print('Two pairs !')
            hand.type<-2
        }else{
            #print('Pair !')
            hand.type<-1
        }
    }else if(max(table(suit)) == 5){
        denom.vals<-(selected.hand %% 13)
        denom.vals[denom.vals==0]<-13
        if(max(denom.vals)-min(denom.vals) == 4){
            if(max(denom.vals) == 13){
                #print('Royal Flush !')
                hand.type<-9
            }else{
                #print('Straight Flush !')
                hand.type<-8
            }
        }else{
            #print('Flush !')
            hand.type<-5
```

```
        }
    }else{
        denom.vals<-(selected.hand %% 13)
        denom.vals[denom.vals==0]<-13
        if(max(denom.vals)-min(denom.vals) == 4){
            #print('Straight !')
            hand.type<-4
        }
    }
    return(hand.type)
}
Nsamp<-1000000
sample.hands<-t(replicate(Nsamp,sample(1:52,size=5)))
hand.totals<-apply(sample.hands,1,hand.type)
hand.totals<-tabulate(hand.totals+1,nbins=10)
simulated.probs<-hand.totals/Nsamp
```

Here is a summary of the simulation results:

```
Hands<-c('Null','Pair','Two Pairs','Three of a Kind','Straight','Flush',
         'Full House','Four of a Kind','Straight Flush','Royal Flush')
#hand.totals<-c(n0,n1,n2,n3,n4,n5,n6,n7,n8,n9)
data.frame(Hands,Expected=round(Nsamp*Probs),Simulated=hand.totals,
            TrueProb=Probs,EstimatedProb=simulated.probs)

+               Hands Expected Simulated       TrueProb EstimatedProb
+ 1              Null   501177    502341 0.501177394035      0.502341
+ 2              Pair   422569    422059 0.422569027611      0.422059
+ 3         Two Pairs    47539     47315 0.047539015606      0.047315
+ 4   Three of a Kind    21128     21134 0.021128451381      0.021134
+ 5          Straight     3925      3446 0.003924646782      0.003446
+ 6             Flush     1965      2032 0.001965401545      0.002032
+ 7        Full House     1441      1438 0.001440576230      0.001438
+ 8    Four of a Kind      240       216 0.000240096038      0.000216
+ 9    Straight Flush       14        18 0.000013851695      0.000018
+ 10      Royal Flush        2         1 0.000001539077      0.000001
```