Bridge Assembly

Some years ago, a coworker and weekend Bridge player, told me he had received two 25 point hands in one evening, something he thought rare, and asked if I would calculate the probability of this event. Now, the distribution of cards in four, 13 card bridge hands is not difficult to model, but the distribution of points is somewhat more complicated (face cards are the only ones with point value: Ace=4 points, King=3, Queen=2, and Jack=1, giving multiple possibilities for, say, a 12 point hand – one with three Aces, or four Kings, or two Aces and two Queens, or four Queens and four Jacks, etc.). For each combination of 13 or fewer face cards there exist $\binom{36}{13-N_{face\_cards}}$ distinct combinations of numbered cards that can be combined with the face cards to form a 13 card Bridge hand (there are 36 numbered cards and 13-N_{face_cards} non-face card positions available). The total number of possible hands is $\binom{52}{13}$, giving $\frac{\binom{36}{13-N_{face\_cards}}}{\binom{52}{13}}$ as the probability of an individual face card combination. Using this, I decided to identify all hands with a point value greater than or equal to 25 and sum their individual probabilities. Deciding on FORTRAN for a lean solution, I wrote a program for my IBM XT, launched it, and waited. And waited. And still waited. After an hour or so, I terminated the program, inserted some progress indicators and time stamps, and finally calculated a requirement of about two weeks to execute the program to completion. At that time, the early stages of PC programming, it was necessary to write assembly language procedures to accomplish much of what we take for granted today - clearing the screen, reading a communication port, etc. And so I thought of using the binary digits of an Intel 8088 register (16 bits – the number of face cards in a deck) to model bridge hands. The great thing about assembly language is that a program executes very efficiently, with no extraneous overhead, the only instructions executed are those explicitly supplied by the programmer. In fact, the calculation portion of the program I wrote to solve this problem contained only 17 machine instructions, which is very compact. The drawback is that the set of available operations is very limited and primitive - increment a register, check for carry, shift left (binary multiplication by 2), etc. It is important to recognize, however, that all computer programs, whether developed in C, SQL, SAS, or HTML, ultimately reduce to machine instructions, which are all that the CPU can process. The assembly language programmer simply "cuts to the chase" and instructs his microprocessor directly, without interpretation.

Following is the calculation portion of my program:

```
DATA    SEGMENT'DATA'
PVAL    DB 0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4         ;POINT VALUES
NCARDS  DB 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0         ;NUMBER OF 25 PT HANDS

CODE    SEGMENT'CODE'

        Preparatory code appears here

        MOV CX,0FFFFH               ;2**16 POSSIBLE FACE CARDS
D:      XOR DL,DL                   ;CLEAR DL REGISTER, IT WILL ACCUMULATE POINTS
        XOR SI,SI                   ;CLEAR SI REGISTER, NUMBER CARDS IN HAND
        MOV BX,10H                  ;SCAN BITS IN CX, 16 IN ALL, BIT=1 THEN CARD IN HAND
        PUSH CX                     ;SAVE CURRENT HAND BINARY ENCODING
A:      SHL CX,1                    ;EXAMINE LEFT-MOST BIT
        JNC B                       ;IF NO CARRY THEN BIT IS OFF (CARD NOT IN HAND)
        ADD DL,PVAL[BX]             ;ACCUM POINTS, PVAL => PT VAL FOR CURRENT BIT (CARD)
        INC SI                      ;ADD 1 TO CARDS IN HAND COUNT
B:      DEC BX                      ;PREPARE TO EXAMINE NEXT BIT (CARD)
        JNZ A                       ;CONTINUE UNTIL ALL BITS (CARDS) EXAMINED

        POP CX                      ;RESTORE CURRENT INITIAL BIT PATTERN FOR NEXT ROUND
        CMP DL,25                   ;25 POINTS IN CURRENT HAND?
        JL C                        ;IGNORE IF POINTS<25
        ADD SI,SI                   ;OTHERWISE POINT TO WORD CONTAINING COUNT
        ADD NCARDS[SI],1            ;OF 25+ POINT HANDS WITH CURRENT CARDS IN HAND
C:      LOOP D                      ;DECREMENT CX (UNTIL 0) AND ANALYZE NEXT PATTERN
```

This may seem a bit cryptic, but it worked, and completed in a matter of seconds. Note how the exhaustive decrementing of the CX register models the complete population of face card combinations and, therefore, point combinations. Fast-forward to the present, and having access to SQL and SAS, I thought I would test the performance of modern tools with this problem (what will we say of *them* in 25 years?). Following is an SQL program that calculates the probability of all possible point values in a Bridge hand:

```
create proc BridgeHand as

-- Calculate Bridge hand point probability distribution

-- Assumptions:  Player holds 13 random cards with point value calculated as
-- Number of Aces * 4 + number of Kings * 3 + number of Queens * 2 + number of Jacks * 1

-- Calculate probability of point value p as the ratio of total number of hands with
-- point value p to the total number of possible hands (combinations of 52 items taken
-- 13 at a time)

-- Note that there are a total of 2**16 possible face card combinations (cards with point
-- value), but at most 13 are held in a valid hand

-- Also in every hand are 13-FaceCardCount numbered cards (each with no point value)

-- Each face card combination can be combined with (36 taken 13-FaceCardCount at a time)
-- combinations of numbered cards

-- Therefore, the probability of a particular face card combination is
-- (36 taken 13-FaceCardCount at a time)/(combinations of 52 items taken 13 at a time)

-- Final probabilities are individual face card combination probabilities accumulated by
-- point value

declare @n tinyint, @c bigint, @comb52in13 bigint
declare @inout table(i tinyint)
declare @NumberedCardComb table(CardsInHand tinyint, Combinations int)

-- Create 0 and 1 elements to indicate whether a card is in or absent from an encoded
-- hand
insert into @inout values(0)
insert into @inout values(1)

-- Create count of numbered card combinations (combinations of non-face cards in a 13
-- card hand)
-- These are used to multiply a face card combination by to count total number of hands
-- (face and numbered
-- cards) with a given combination of face cards
-- Generate for 36 cards in 0 through 12 positions for hands with 1 through 13 face cards
-- (non-zero points)
insert into @NumberedCardComb values(0,1)
insert into @NumberedCardComb values(1,36)
select @c=36, @n=2
while(@n<13)
  begin
    -- Calculate combinations(36,n)
    select @c=@c*(36-@n+1)/@n
    insert into @NumberedCardComb values(@n, @c)
    select @n=@n+1
  end

-- Calculate total number of 13 card hands from 52 cards
select @comb52in13=52, @n=2
while(@n<14)
  begin
    select @comb52in13=@comb52in13*(52-@n+1)/@n, @n=@n+1
  end

-- Generate all possible combinations of face cards
-- Calculate binary encoding of each combination to indicate which cards are included
-- (for reference only)
```

```
-- Calculate score of each combination (hand):
-- 4 points for each Ace + 3 for each King + 2 for each Queen + 1 for each Jack
-- Count hands by score
-- Calculate proportion of hands by score to total possible hands (52 cards taken 13 at a
-- time)
select   FaceComb.Points,
         str(sum(convert(real,NumberedComb.Combinations)/@comb52in13),16,12) as
         ProbabilityOfOccurrence
from     ( select i0.i + 2*i1.i + 4*i2.i + 8*i3.i + 16*i4.i + 32*i5.i + 64*i6.i +
                  128*i7.i + 256*i8.i + 512*i9.i + 1024*i10.i + 2048*i11.i + 4096*i12.i +
                  8192*i13.i + 16384*i14.i + 32768*i15.i as Encoding,
                  i0.i + i1.i + i2.i  + i3.i  + i4.i  + i5.i  + i6.i  + i7.i +
                  i8.i + i9.i + i10.i + i11.i + i12.i + i13.i + i14.i + i15.i as
                  CardsInHand,
                  (i15.i + i14.i + i13.i + i12.i)*4 + (i11.i + i10.i + i9.i + i8.i)*3 +
                  (i7.i  + i6.i  + i5.i  + i4.i)*2  +  i3.i  + i2.i  + i1.i + i0.i  as
                  Points
           from   -- Join 0 and 1 in 16-order combinations
                  -- Each resulting tuple is a combination 16 digits of 0 or 1, each
                  -- indicating whether
                  -- the corresponding face card is in the combination (hand) or not
                  -- Aces are in high order positions (15, 14, 13, 12), Jacks low (3, 2,
                  -- 1, 0)
                  @inout i0 cross join @inout i1 cross join @inout i2
                  cross join @inout i3 cross join @inout i4 cross join @inout i5
                  cross join @inout i6 cross join @inout i7 cross join @inout i8
                  cross join @inout i9 cross join @inout i10 cross join @inout i11
                  cross join @inout i12 cross join @inout i13 cross join @inout i14
                  cross join @inout i15
         ) FaceComb join @NumberedCardComb NumberedComb
           on 13-FaceComb.CardsInHand=NumberedComb.CardsInHand
group by Points
order by Points
```

Following is the SQL output:

Points In Hand	Probability of Occurrence
1	0.007884415798
2	0.013561195228
3	0.024623637088
4	0.038454384528
5	0.051861934131
6	0.065540961019
7	0.080280870578
8	0.088921892153
9	0.093562274589
10	0.094051148112
11	0.089446804007
12	0.080268651668
13	0.069143318745
14	0.056933232091
15	0.044236791933
16	0.033109185342
17	0.023616948710
18	0.016050844427
19	0.010361728846
20	0.006435356017

Points In Hand	Probability of Occurrence
21	0.003778671765
22	0.002100427624
23	0.001119036931
24	0.000559033519
25	0.000264277654
26	0.000116682941
27	0.000049066576
28	0.000018567730
29	0.000006671650
30	0.000002198485
31	0.000000611319
32	0.000000171896
33	0.000000035212
34	0.000000007061
35	0.000000000983
36	0.000000000094
37	0.000000000006

My friend's question was on the probability of a 25 point (and I assume greater) hand, which is found by summing the probabilities for all hands with points from 25 through 37 (37 is as high as a Bridge hand goes, with Aces worth 4 points, Kings 3, Queens 2, and Jacks 1, any of the four hands with all of the Aces, all of the Kings, all the Queens, and one of the four Jacks has maximum point value of 37). His answer, for a single hand, was approximately 0.00045, or 0.045%, which is rather small. The probability of

multiple hands is a function of how many hands are played – assuming independence of hands, the probability of 2 or more 25+ hands out of *k* is $1 - 0.99955^k - k \times 0.00045 \times 0.99955^{k-1}$, using binomial probabilities. So, for instance, receiving two or more 25 point or greater hands out of 10 has a probability of $1 - 0.99955^{10} - 10 \times 0.00045 \times 0.99955^9 = 0.00000909$, which is very small and confirms my friend's intuition. This SQL solution completes in less than two seconds, well worth the wait.

Following is a SAS variation, which also completes in less than two seconds:

```
/* Bridge Hand Point Problem

   or

   What is the probability distribution of points in Bridge hands?

   In Bridge, 52 cards are randomly dealt to four players, 13 cards to each,
   the game is played, then each player's hand is scored, face cards carry value
   (Ace=4 points, King=3 points,
   Queen=2 points, and Jack=1 point) and number cards receive no point value.

   Outline:

   1.  Generate all combinations of face cards containing 13 or fewer cards
   2.  Calculate the number of possible number card combinations (36 cards taken
       13-FaceCardCount at a time) for each face card combination
   3.  Calculate probability of an individual face card combination as
       the ratio of possible number card combinations to the total number of possible
       hands (52 cards taken 13 at a time)
   4.  Accumulate probabilities by point value

   Note that there exist 2**16 distinct combinations of face cards (there are a total of
   16 face cards, each one either in or out of a given hand), but at most 13 appear in
   any hand.
   Also in every hand are 13-FaceCardCount numbered cards and each face card combination
   can be combined with (36 taken 13-FaceCardCount at a time) combinations of numbered
   cards
   Therefore, the probability of a particular face card combination is
   (36 taken 13-FaceCardCount at a time)/(combinations of 52 items taken 13 at a time)    */

options nosource;

/* Generate all combinations (hands) of face cards */
data FaceHand(keep=c1-c16 CardsInHand Points);
  /* Setup array of cards, one position for each, Aces in high order pos, Jacks low */
  array c c1-c16 (0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0);
  /* Binary add and carry 1 2**16 times to exhaust all combinations
     Discontinue when no carry since at that point leftmost bits unchanged */
  do i=1 to 65536;
    j=1;
    do until(^carry or j=17);
      c{j}=1-c{j};
      if(c{j}=0)then carry=1; else carry=0;
      j+1;
    end;
    /* Omit combinations with more than 13 cards
       Calculate points */
    CardsInHand=0;
    Points=0;
    do j=1 to 16;
      CardsInHand=CardsInHand+c{j};
      Points=Points+c{j}*(int((j-1)/4)+1);
    end;
    if(CardsInHand<14)then output;
  end;
run;

/* Create count of numbered card combinations (combinations of non-face cards in a
   13 card hand)
   These are used to multiply a face card combination by to count total number of hands
   (face and numbered cards) with a given combination of face cards
```

```
   Generate for 36 cards in 0 through 12 positions for hands with 1 through 13 face cards
   (non-zero points) */
data NumberedCardComb(keep=CardsInHand Combinations);
  CardsInHand=0; Combinations=1; output;
  CardsInHand=1; Combinations=36; output;
  do CardsInHand=2 to 12;
    /* Calculate combinations(36, CardsInHand) */
    Combinations=Combinations*(36-CardsInHand+1)/CardsInHand;
    output;
  end;
run;

/* Calculate and save total number of 13 card hands from 52 cards */
%global Comb52in13;
data _null_;
  comb52in13=52; n=2;
  do while(n<14);
    comb52in13=comb52in13*(52-n+1)/n;
    n=n+1;
  end;
  call symput('Comb52in13',comb52in13);
run;

title 'Bridge Hand Probabilities by Points in Hand';

/* Combine face cards with count of numbered card combinations
   Calculate probability of each face card hand (numbered card comb / total number of
   possible 52 card hands)
   Accumulate probabilities by points in hand */
proc sql;
  select   FaceHand.Points, sum(NumberedCardComb.Combinations/&Comb52in13) as P
  from     FaceHand
           join NumberedCardComb on 13-FaceHand.CardsInHand=NumberedCardComb.CardsInHand
  group by FaceHand.Points;
quit;
```

Following is the SAS output, which is identical to that of the SQL program:

Points	P	Points	P	Points	P	Points	P
1	0.007884	11	0.089447	21	0.003779	31	6.113E-7
2	0.013561	12	0.080269	22	0.002100	32	1.719E-7
3	0.024624	13	0.069143	23	0.001119	33	3.521E-8
4	0.038454	14	0.056933	24	0.000559	34	7.061E-9
5	0.051862	15	0.044237	25	0.000264	35	9.83E-10
6	0.065541	16	0.033109	26	0.000117	36	9.45E-11
7	0.080281	17	0.023617	27	0.000049	37	6.30E-12
8	0.088922	18	0.016051	28	0.000019		
9	0.093562	19	0.010362	29	6.672E-6		
10	0.094051	20	0.006435	30	2.198E-6		