11 En pokersimulation

Den følgende applikation handler om at simulere en pokerhånd. Emnet for applikationen synes måske at være temmelig kuriøst og kun interessant for inkarnerede pokerspillere, men den indeholder faktisk noget interessant VBA kode.

I tilfælde af, at du ikke er pokerspiller kan jeg fortælle dig, at i poker gives der fem kort fra et ordinært kortspil på 52 kort. Der er mulighed for, at der gives forskellige "hånde" til en spiller som beskrevet herunder:

- **Et par**: to kort af samme slags og tre andre slags (eks. to 10'ere, en 5'er, en knægt, og en 8'er)
- **To par**: to kort af én slags og to kort af en anden slags og et andet kort
- **Tre ens**: tre kort af samme slags og to andre kort af forskellig slags
- **En straight**: fem kort på stribe, f.eks. 5, 6, 7, 8, 9 men ikke i samme farve
- **En flush**: fem kort i samme farve, f.eks. fem ruder.
- **Et full house**: tre kort af samme slags og to af en anden slags, f.eks. tre 5'ere og to 9'ere
- **Fire ens**: fire af samme slags, og et andet kort
- **En straight flush**: fem på stribe i samme farve
- **En bust**: ingen af ovenstående.

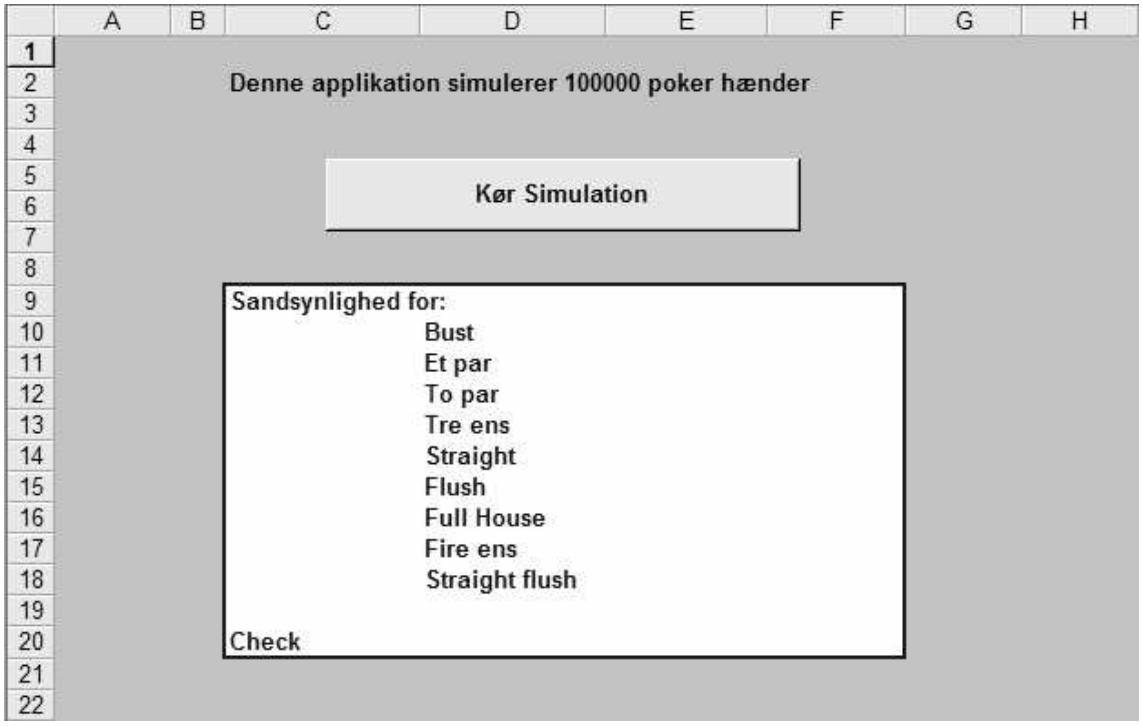
Undtagen bust, er alle ovenstående listet efter deres værdi og i stigende orden, i.e. en flush slår en straight, som slår en …

Denne applikation simulerer 100.000 "hænder", alle med 5 kort og alle givet fra en kortbunke med 52 kort. Applikationen tæller herefter antallet af gange, hver type af hånd fra ovenstående liste optræder. Simulationen kan bla. afsløre, om en hånd bestående af ét par er mere sandsynlig end en hånd bestående af tre ens.

Applikationens formål er med andre ord at give 5 kort et antal gange og tælle, hvor mange gange en bestemt hånd optræder.

11.1 Eksekvering af applikationen

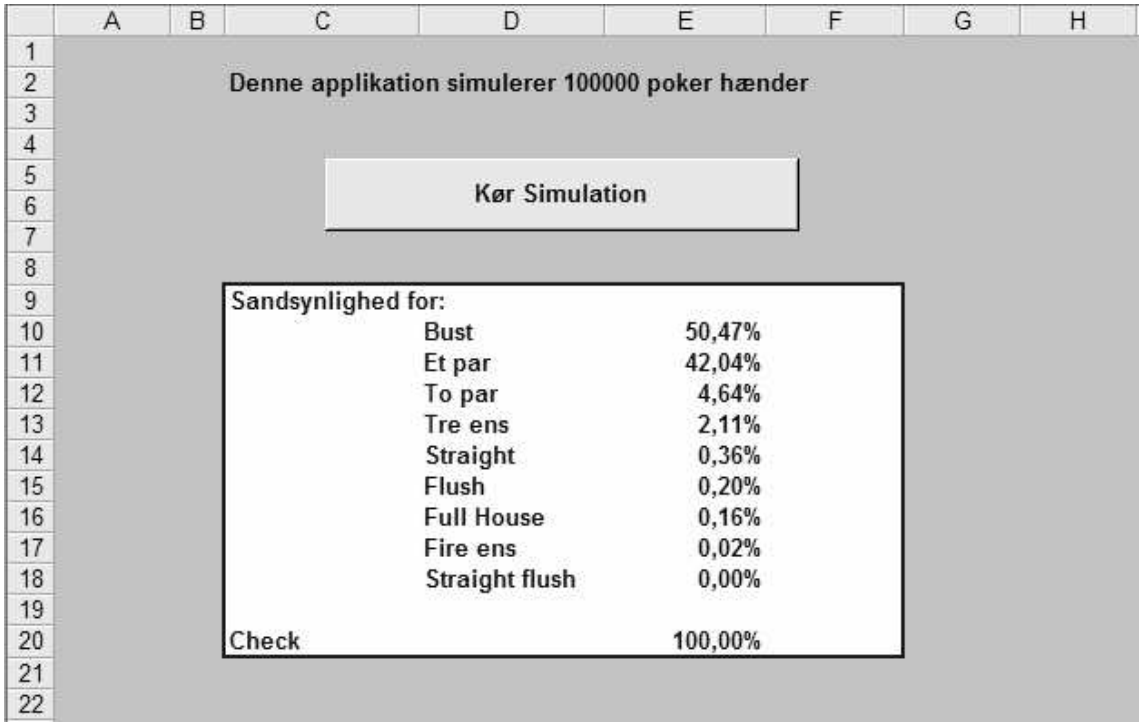
Applikationen kan du finde i filen **Poker.xls**. Filen indeholder et enkelt ark, som hedder Report, som vises, når filen åbnes. Arket er vist i nedenstående figur.



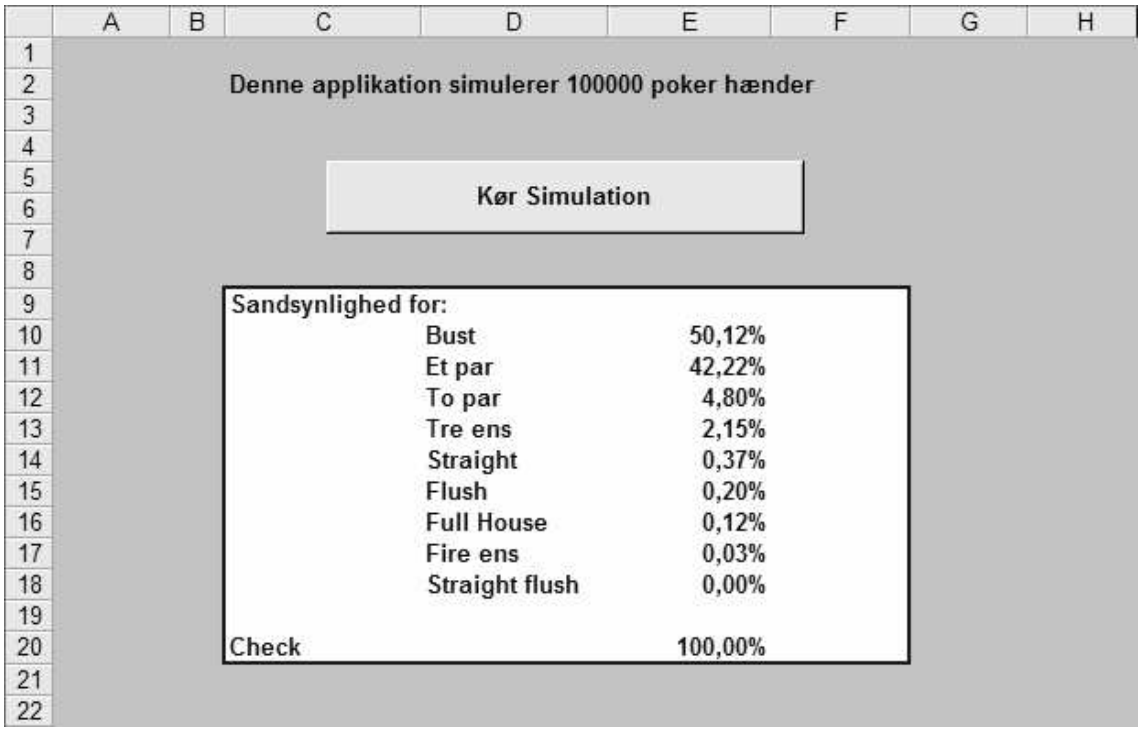
Figur 11-1 Reportarket før simulationen køres.

Hver gang en bruger trykker på knappen, gives 100.000 nye fem-korts hænder, og resultatet af, hvor mange gange en bestemt hånd gives ud af det totale antal givet hænder, vises på arket. Dette er vist i Figur 11-2 nedenfor.

Jeg skrev at 100.000 *nye* hænder simuleres, fordi hvert gennemløb anvender et nyt sæt af tilfældige tal i simulationen. Resultatet er derfor forskelligt hver gang, der trykkes på knappen. I Figur 11-3 er et andet resultat for et gennemløb vist. Resultaterne i Figur 11-3 ligner selvfølgelig i værdi meget de i Figur 11-2 viste – det er jo netop ideen bag simulation.



Figur 11-2 Resultat af en simulation



Figur 11-3 Resultat af en anden simulation

Hver af disse simulationskørsler illustrerer, hvad der formelt kan vises fra et formelt statistisk argument – nemlig at sandsynligheden for en bestemt hånd er omvendt proportional med værdien af hånden. En bust er mest sandsynlig, et par er dernæst mest sandsynlig, osv. Det mest

nedslående er egentlig, at der åbenbart skal mere til end held for at få f.eks. fire ens eller straight flush.

11.2 Opsætning af Excelarket og modulet

Der er faktisk ikke noget at sætte op i Report arket på designtidspunktet – ud over de labels og farver som er vist i ovenstående figurer. Simulationen foregår udelukkende i en VBA kode – der er ingen ark, hvor der foretages udregninger.

Der skal kun oprettes ét modul til denne applikation. Følgende kodestump skal dog skrives i ThisWorkbook kode vinduet:

```
Private Sub Workbook_Open()
       Range("E10:E20").ClearContents
       Range("D6").Select
End Sub
```

11.3 VBA koden i modulet

Indtil videre virker denne applikation temmelig simpel, men du vil ved selvsyn finde ud af, at koden er langt fra simpel. Koden indeholder bla. ikke trivielle logiske strukturer og gør i udstrakt grad brug af arrays. Som bekendt er et menneske meget hurtig i stand til at genkende mønstre, og vil meget hurtigt finde ud af, om den uddelte hånd indeholder et par, en straight, eller noget andet brugbart – uden at kortene skal arrangeres på forhånd. Som du vil se, skal der betragtelig meget programmeringskode til at gøre en computer i stand til bare at genkende et par.

11.3.1 Options statements og modul-niveau variablene

Modul niveau variablene listes først, og som sædvanligt deklareres de med Dim kodeordet. Som du så i de første kapitler, behøver vi ikke at deklarere variablene med kodeordet Public, fordi vi i denne applikation kun har et modul. Vi vil dog gøre brug af modul-nivaeu variable – globale variable – som defineres øverst i modulet, fordi de enkelte subrutiner skal kunne tilgå disse variable.

```
Option Explicit
Option Base 1

' Definitioner af modul-niveau variable:
'   NBust – antallet af de 100000 hænder som resulterer i en bust (med lignende
'       definitioner for NPair, N2Pair, osv.
'   Denom - array som indikerer hvilket nummer (1 to 13) hvert kort
'       i deck'et er
'   Card - array som indikerer kortene I hånden – feks. hvis Card(3) = 37,
'       betyder det, at det tredje kort givet er det 37'te kort i bunken
'   NReps – antallet af simulerede hænder, her 100000
```

```
Dim NBust As Long, NPair As Long, N2Pair As Long, N3ofKind As Long, _
    NFullHouse As Integer, N4ofKind As Integer, NStraight As Integer, _
    NFlush As Integer, NStraightFlush As Integer, Denom(52) As Integer, _
    Card(5) As Integer, NReps As Long
```

Som du ser, gemmes antallet af de enkelte muligheder i hver deres variabel. Det er selvfølgelig klart, at der i stedet kunne have været defineret et array af passende dimension. Prøv at gøre det selv.

11.3.2 Main koden

Main koden kaldes, når der trykkes på knappen i arket Report. I koden kaldes for det første subrutinen InitStats, som nulstiller alle tællere. Main koden kalder herefter subrutinen SetupDeck, som definerer/lister en kortbunke. Koden anvender herefter en For løkke til at styre de 100.000 replikationer, som simulationen udfører. I hvert step i For løkken kaldes to subrutiner. Den ene subrutine, Deal, "giver" kortene til brugeren, og den anden subrutine Evaluate evaluerer, hvilken hånd der er givet i dette step. Endelig kaldes Report subrutinen, som har til opgave at skrive resultaterne til Report arket. Bemærk kaldet til Randomize funktionen, som er placeret i begyndelsen af Main rutinen. Randomize funktionen sikrer, at der altid anvendes nye tilfældige tal – hver gang simulationen udføres. Syntaxen for Randomize har jeg listet herunder:

Randomize Statement

Initializes the random-number generator.

Syntax

Randomize [*number*]

The optional *number* argument is a Variant or any valid numeric expression.

Remarks

Randomize uses *number* to initialize the **Rnd** function's random-number generator, giving it a new seed value. If you omit *number*, the value returned by the system timer is used as the new seed value.

If **Randomize** is not used, the **Rnd** function (with no arguments) uses the same number as a seed the first time it is called, and thereafter uses the last generated number as a seed value.

Note To repeat sequences of random numbers, call **Rnd** with a negative argument immediately before using **Randomize** with a numeric argument. Using **Randomize** with the same value for *number* does not repeat the previous sequence.

I boksen kan du bla. se, at hvis der ikke angives et tal efter Randomize kaldet, initialiseres seed værdien med systemtiden. Du vil derfor altid få en ny række af tilfældige tal – uanset hvornår du eksekverer koden.

```
Sub Main()
    Dim i As Long
    Randomize
    NReps = 100000

' Set counters to 0.
    Call InitStats

' "Name" the cards in the deck.
    Call SetupDeck

' Deal out NReps poker hands and evaluate each one.
    For i = 1 To NReps
        Call Deal
        Call Evaluate
    Next

' Report the summary stats from the NReps hands.
    Call Report

    Range("D6").Select
End Sub
```

11.3.3 InitStats koden

InitStats koden sætter alle tællere (antallet af Busts, antallet af Et par, osv.) til 0

```
Sub InitStats()
    NBust = 0
    NPair = 0
    N2Pair = 0
    N3ofKind = 0
    NStraight = 0
    NFlush = 0
    NFullHouse = 0
    N4ofKind = 0
    NStraightFlush = 0
End Sub
```

11.3.4 SetupDeck koden

SetupDeck koden "definerer" kortbunken ved at skrive passende værdier til Denom arrayet. Dette udføres ved brug af to nestede For løkker. Hvis du følger logikken, ser du at Denom(1) til Denom(4) alle initieres til 1 – det er jo esserne. Denom(5) til Denom(8) sættes alle til værdien 2 – det er jo to'erne, osv. Husk her på, at 11 svarer til knægtene, 12 til damerne, og 13 til kongerne. Der ikke angivet eksplicit, hvad der er hjerte, ruder, klør eller spar, men du kan tænke på 1, 5, 9 osv. som værende hjerter; 2, 6, 10 osv. som ruder; 3, 7, 11 osv. som klør, og 4, 8, 12 osv. som spar.

```
Sub SetupDeck()
    Dim i As Integer, j As Integer
' Giv de første 4 kort værdien 1 (essere),
' de næste værdien 2 (2'ere), osv.
```

```
    For i = 1 To 13
        For j = 1 To 4
            Denom(4 * (i - 1) + j) = i
        Next
    Next
End Sub
```

11.3.5 Deal koden

Deal subrutinen vælger tilfældigt fem kort fra en 52 kortbunke. Dette er den eneste subrutine, hvor der foregår noget simulation. Det er med andre ord den eneste rutine, som arbejder med tilfældige tal. Rutinen anvender VBA funktionen, Rnd, til at trække et uniformt fordelt tilfældigt tal imellem 0 og 1 – se evt. Rnd fuktionens syntax i online hjælpen. Når et tilfældig tal imellem 0 og 1 af typen double skal transformeres til et heltal imellem 1 og 52, specificeres følgende kodelinie:

```
CardIndex = Int(Rnd * 52) + 1
```

Hvor Int(Rnd * 52) giver det største heltal, som er <= værdien af Rnd*52. Tallet ligger dermed imellem 0 og 51 – så vi adderer udtrykket med værdien 1 for at få et tal imellem 1 og 52.

Det boolske array Used holder styr på hvilket kort, der allerede er udtrukket, så det ikke bliver trukket igen! Essentielt trækkes der derfor tilfældige tal, indtil en brugbar hånd opnås, og med udgangen af denne rutine skrives de fem kort til Card arrayet. F.eks. hvis Card(4) = 47 betyder det, at det fjerde kort i denne hånd er det 47. kort i den ordnede kortbunke = klør dame.

```
Sub Deal()
    Dim i As Integer, j As Integer
    Dim CardIndex As Integer, Used(52) As Boolean, NewCard As Boolean

' Used arrayet holder styr på hvilke kort, der allerede er blevet givet
' Hvis Used(i)=True så er kort i allerede uddelt
    For i = 1 To 52
        Used(i) = False
    Next

' NewCard er et flag som vedbliver at være falsk indtil et nyt kort gives
    For i = 1 To 5
        NewCard = False
        Do
            CardIndex = Int(Rnd * 52) + 1
            If Not Used(CardIndex) Then
                NewCard = True
                Used(CardIndex) = True
            End If
        Loop Until NewCard = True

' Card(i) holder kortnummeret givet på plads i
        Card(i) = CardIndex
    Next
End Sub
```

11.3.6 Evaluate koden

Subrutinen Evaluate er den mest besværlige subrutine. Når rutinen kaldes, er Card arrayet allerede initialiseret og genereret. Card indeholder måske kort 2, 7, 19, 28 og 47, men problemet er bare, hvilken hånd det repræsenterer? Er det Et par, Tre ens eller ? Evaluate rutinen finder svaret.

Det første der undersøges er, om der er tale om en straight. Rutinen finder værdien på kortene og gemmer disse i arrayet CardDenom. F.eks. er det værdien af det første kort Denom(Card(1)), som gemmes i CardDenom(1) – med andre ord, hvis Card(1) = 11, vil Denom(Card(1)) finde, at der er tale om en 3'er. Værdierne er måske ikke ordnet som f.eks. 5, 3, 7, 6, 4, så rutinen anvender to For løkker til at sortere værdierne i stigende orden. Rutinen undersøger herefter, om disse sorterede værdier former en rækkefølge, som 3, 4, 5, 6, 7. (prøv at komme op med en anden metode!)

Det andet, der undersøges, er om der er en Flush. For eksempel er en hånd bestående af kortene 3, 15, 23, 39, 51 en flush. Det er fordi kortene 3, 7, 11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 51 alle tilhører den samme farve – f.eks. klør. En hurtig måde at finde ud af, om fem kort har den samme farve, er ved at dividere hver af dem med tallet 4 og se, om resten af divisionen alle er ens. (ved kortene 3, 7, 11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 51 er resten ens for alle og = 3) I VBA er en sådan operation meget let – vi anvender VBAs Mod operator. F.eks. repræsneterer `51 Mod 4` resten af divisionen 51/4.

Det tredje, der undersøges for, er for en Straight flush, som jo kun kan opstå, hvis både den boolske variabel HasStraight og den boolske variabel HasFlush begge har værdien True.

Hvis hånden er enten en straight eller en flush – eller begge, skal der ikke tjekkes yderligere. Hvis det på den anden side ikke er tilfældet, skal der tjekkes for et par, to ens, og de resterende muligheder. Alle disse går ud på at finde ud af, om værdien af kortene er den samme. F.eks. er en hånd bestående af to par kendetegnet ved, at der er to kort med samme værdi, og to andre med en anden, men ens værdi. Groups arrayet anvendes til at holde denne information. Full House har Groups(3) = 1 og Group(2) = 1 som betyder, at hånden har en gruppe af størrelsen 3 og en gruppe af størrelsen 2. På samme vis har bust Groups(1) = 5, tre ens har Groups(3) = 1 og Groups(2) = 1[24] osv. Så ved at oprette Groups arrayet og tjekke dets indhold, kan programmet finde ud af, hvilken hånd der pt. er blevet givet.

---

[24] Tre ens indeholder jo også to ens!

```
Sub Evaluate()
    Dim i As Integer, Count(13) As Integer, Groups(4) As Integer, _
        j As Integer, HasStraight As Boolean, _
        HasFlush As Boolean, CardDenom(5) As Integer, Temp As Integer

' Først tjek for en straight.
    HasStraight = False
    For i = 1 To 5
        CardDenom(i) = Denom(Card(i))
    Next

' Sorter i stigende orden.
    For i = 1 To 4
        For j = i + 1 To 5
            If CardDenom(j) < CardDenom(i) Then
                Temp = CardDenom(j)
                CardDenom(j) = CardDenom(i)
                CardDenom(i) = Temp
            End If
        Next
    Next

' Tjek om der er en straight - i.e. 4, 5, 6, 7, 8.
    If CardDenom(2) = CardDenom(1) + 1 And _
        CardDenom(3) = CardDenom(2) + 1 And _
        CardDenom(4) = CardDenom(3) + 1 And _
        CardDenom(5) = CardDenom(4) + 1 Then
        HasStraight = True
        NStraight = NStraight + 1
    End If

' dernæst, tjek for en flush.
    HasFlush = False
    If Card(1) Mod 4 = Card(2) Mod 4 And Card(2) Mod 4 = Card(3) Mod 4 _
                And Card(3) Mod 4 = Card(4) Mod 4 _
                And Card(4) Mod 4 = Card(5) Mod 4 Then
        HasFlush = True
        NFlush = NFlush + 1
    End If

' Er der straight flush?
    If HasStraight And HasFlush Then
        NStraightFlush = NStraightFlush + 1
        NStraight = NStraight - 1
        NFlush = NFlush - 1
    End If

' Ingen grund til at tjekke videre hvis straight, flush eller begge
    If HasStraight Or HasFlush Then Exit Sub

' Ellers, tjek for alle andre muligheder - Count(i) er antallet af kort med
' kortnummeret i på hånden
    For i = 1 To 13
        Count(i) = 0
    Next
    For i = 1 To 5
        Count(Denom(Card(i))) = Count(Denom(Card(i))) + 1
    Next

' Groups(i) er antallet af "groups" af størrelse i.
' For eksempel, hvis Groups(2) = 1, så er
' der en gruppe af størrelsen 2, => et par .
    For i = 1 To 4
        Groups(i) = 0
    Next
```

```
    For i = 1 To 13
        If Count(i) > 0 Then Groups(Count(i)) = Groups(Count(i)) + 1
    Next

' tjek alle muligheder
    If Groups(1) = 5 Then
        NBust = NBust + 1
    ElseIf Groups(1) = 3 And Groups(2) = 1 Then
        NPair = NPair + 1
    ElseIf Groups(1) = 1 And Groups(2) = 2 Then
        N2Pair = N2Pair + 1
    ElseIf Groups(1) = 2 And Groups(3) = 1 Then
        N3ofKind = N3ofKind + 1
    ElseIf Groups(2) = 1 And Groups(3) = 1 Then
        NFullHouse = NFullHouse + 1
    Else
        N4ofKind = N4ofKind + 1
    End If
End Sub
```

11.3.7 Report koden

Report subrutinen lister resultaterne på Report arket. Bemærk at koden afrapporterer de relative frekvenser. Formlen, der skrives i celle E20, er egentlig ikke nødvendig, men specificeres alligevel for at se, om de relative frekvenser summer til 1. Hvis andet observeres, er der fejl i programmet.

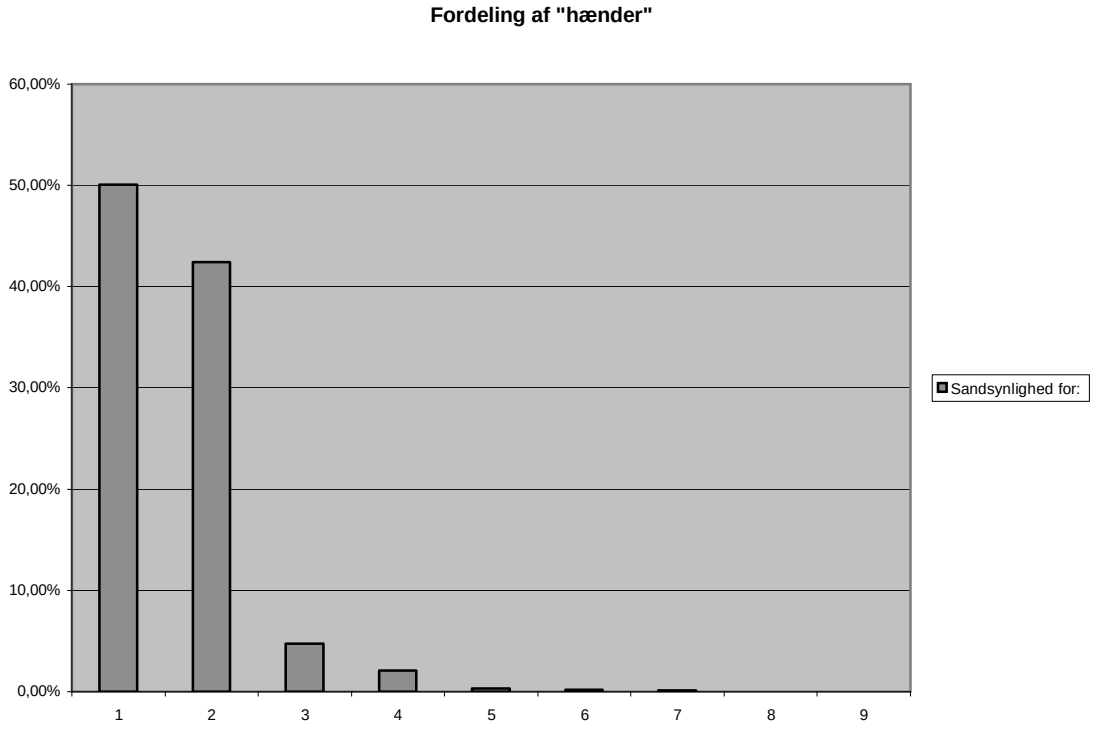
```
Sub Report()
    Range("E10") = NBust / NReps
    Range("E11") = NPair / NReps
    Range("E12") = N2Pair / NReps
    Range("E13") = N3ofKind / NReps
    Range("E14") = NStraight / NReps
    Range("E15") = NFlush / NReps
    Range("E16") = NFullHouse / NReps
    Range("E17") = N4ofKind / NReps
    Range("E18") = NStraightFlush / NReps
    Range("E20").Formula = "=Sum(E10:E18)"
End Sub
```

11.4 Opsamling

Denne applikation illustrerer nogle interessante og bestemt ikke trivielle logiske strukturer, For løkke konstruktioner og arrays. Applikationen viser ligeledes, at jo mere værdifuld en pokerhånd bliver, jo mindre sandsynligt bliver den også at opnå.

11.5 Opgaver

1 Lav din applikation om så den også indeholder en graf, som den vist i Figur 11-4, over de sandsynligheder, som applikationen finder. Lav en knap på Report arket, som fører brugeren til denne graf.



Figur 11-4 Graf til opgave 1

2 Der er mange versioner af poker. Lav nu din applikation om sådan, at brugeren får givet 6 kortog herefter får lov til at smide det dårligste. Antag at brugeren altid vil smide det dårligste kort, sådan at de resterende 5 kort danner den bedst mulige hånd. (Hint: Kør Evaluate subrutinen for hver 5 kortskombination, der findes og behold den bedste kombination.)

3 En mere realistisk version af det forrige problem er, at spilleren gives 5 kort. Spilleren får herefter lov til at bytte op til fire af disse fem kort. Problemet med at simulere dette setup er, at vi mangler spillerens strategi – dvs. afhængig af hvad spilleren er blevet givet, hvad vil han så bytte? Lad os antage følgende strategi: (hint: Kør Evaluate runtinen på den originale hånd for at se, hvad han skal bytte. Kør den så igen på den nye hånd)

- Hvis der gives en bust, bytter spilleren fire kort – og beholder det største.
- Hvis der gives et par, bytter spilleren de andre 3 kort
- Hvis der gives to par, bytter spilleren det sidste kort
- Hvis der gives tre ens, bytter spilleren de sidste to kort

- Hvis der gives alle andre muligheder, bytter spilleren ingen kort

4 (Lidt svær) I de forgående øvelser, prøvede spilleren ikke at fylde huller ud i f.eks. straights eller flushes. F.eks. hvis spilleren sidder med hånden 4, 5, 6, 7, og 10 har han ikke prøvet at bytte 10'eren til enten en 3'er eller en 8'er. Ligeledes har ikke prøvet, hvis han sad med fire hjerter og en spar at bytte denne spar i forsøget på at få en hjerter. Simuler en sådan strategi. Antag, at spilleren først tjekker for en bust. Hvis han modtager en bust, undersøger han, om han har en partial straight, som kan færdiggøres i en af enderne. (f.eks. 4, 5, 6, 7 men ikke 1, 2, 3, 4. Forsøget på at færdiggøre den sidste er for risikabelt, fordi kun en 5'er kan gøre det) Hvis han har en sådan partiel straight, smider han det sidste kort. Ellers tjekker han – stadig med en bust – om der er fire kort i den samme kulør. Hvis dette er tilfældet, bytter han det sidste kort. Resten af strategien er den samme som de, der er listet i forrige opgave. Med andre ord, han forsøger kun at færdiggøre en straight eller en flush, hvis han som udgangspunkt modtager en bust. Er denne strategi bedre, end den strategi du simulerede i forrige opgave?