

O uso do Software R em Jogos de Azar

Gastão Coelho Gomes

Departamento de Métodos Estatísticos, Universidade Federal do Rio de Janeiro

E-mail: gastao@im.ufrj.br

RESUMO

O objetivo deste trabalho é didático, visando entender probabilidade via o conceito frequentista e clássico com comparação dos dois métodos. Usamos o software R para este fim e um cenário envolvendo jogos de moedas e dados, usamos quatro problemas para ilustração. Os resultados frequentistas, gerados no R foram comparados com a solução clássica de probabilidade, cujos cálculos também são apresentados.

PALAVRAS CHAVE, Software R, Probabilidade, Simulação Monte Carlo.

ABSTRACT

The main goal of this paper is didactical, aiming to understand probability through the frequentist concept. We used the R language to support it and a context involving games of coins and dice. The frequentists results, generated in R were compared with the classic probability solution, whose calculations were also presented. Besides that, we presented a study of the empirical distribution of this events.

KEYWORDS, , Software R, Probability, Monte Carlo Simulation.

1, Introdução

Visamos entender probabilidade via o conceito frequentista. Usamos o software R para este fim. O R (<http://www.r-project.org>) é um pacote, de domínio público, que oferece várias funções, já implementadas, dos mais variados métodos estatísticos e probabilísticos. Além disso, é também, um ambiente de programação onde se pode usar o que de bom ele já contém para se desenvolver novas implementações. No R, foram elaborados códigos para que, através de simulações poderíamos calcular, de forma frequentista, as probabilidades referentes aos quatro problemas apresentados aqui:

1. Simulando lançamento de moedas;
2. Total de pontos em dois lançamentos de um dado;
3. Jogo de cinco dados;
4. Jogo de Poker com dados.

O ponto de partida do estudo sistemático das probabilidades pode ser situado em meados século XVII, quando Méré desejava descobrir uma estratégia de jogo que otimizasse o ganho ou reduzisse o risco. Há diversas situações práticas onde é possível calcular probabilidades

de determinados eventos ocorrerem, fazendo uma analogia entre esses problemas e os jogos de azar. Por isso nos livros de probabilidade é muito comum aparecerem vários exemplos com moedas, dados, baralhos, roletas, etc. Nesses tipos de exemplo os espaços amostrais considerados são finitos e equilibrados, todos os eventos elementares têm a mesma probabilidade de ocorrência. De fato, no lançamento de um dado equilibrado, não há razões para se supor que “1” tem mais chance de ocorrer que “2”, “3”, ..., “6”. O conceito clássico de probabilidade, apresentado a seguir é perfeitamente adequado a este tipo de problemas.

Conceito Clássico de Probabilidade: $P(A) = \#(A) / \#(\Omega)$; onde $\#(\Omega)$ é o número de resultados possíveis do experimento e $\#(A)$ é o número de resultados favoráveis à ocorrência do evento A,

Outra maneira de definir probabilidade, de uma forma aproximada, é o Conceito Frequentista: Considere um experimento onde o evento A pode ou não ocorrer. Como definir o que é a probabilidade de o evento A ocorrer? Uma resposta possível a essa pergunta é a que dá origem ao conceito frequentista de probabilidade: Suponha que esse experimento foi repetido n vezes, sempre sob as mesmas condições, e que o evento A ocorreu m vezes entre essas n realizações do experimento, Então a fração m/n é uma boa aproximação para a probabilidade de A, se o número n de repetições for bastante grande. Simbolicamente, $P(A) \cong m/n$

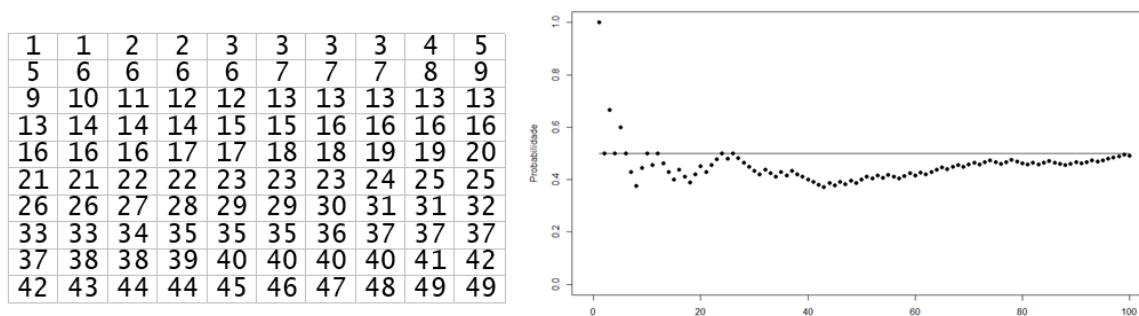
Os resultados frequentistas, gerados no R foram compararmos com a solução clássica de probabilidade, cujos cálculos também são apresentados. Além disso, apresentamos um estudo da distribuição empírica de cada um desses 7 eventos.

2, Problemas analisados:

2.1, Simulando a probabilidade de cara nos lançamentos de uma moeda

Usando o software R, foram simulados 100 lançamentos de uma moeda equilibrada, isto é, onde as chances de cara e de coroa são iguais. Depois de cada lançamento, foi anotado o número acumulado de caras obtidas até esse momento e foi calculada a proporção de caras correspondente. Na tabela a seguir estão apresentados os valores correspondentes ao número acumulado de caras ao longo do processo. Por exemplo, para a jogada de número 30 (último número da 3ª linha) o número acumulado de caras é 13 e a fração de caras é 13/29, O gráfico abaixo mostra a evolução dessa fração à medida que foram feitos os 100 lançamentos da moeda,

Figura-1 – Cara ou coroa? A visão frequentista



Observe que no começo há uma grande variabilidade do valor da probabilidade estimada m/n , mas ele tende a se estabilizar em 0,5 quando o número n de tentativas vai aumentando. Esta é uma propriedade de todo experimento aleatório, chamada *estabilidade estatística*: à medida que o número n de realizações do experimento aumenta, a probabilidade empírica de um dado evento tende a se estabilizar em uma constante.

Observe que o “ponto de estabilidade” – a saber, 0,5 – corresponde ao valor que seria obtido para a probabilidade de cara, ao usarmos o conceito clássico de probabilidade no espaço amostral finito uniforme {cara, coroa}.

Comandos do R para a execução da simulação:

```
x=1:100;  
y=cumsum(sample(0:1,100,rep=T))  
plot(x,y/1:100, ylim=c(0,1), xlim=c(0,100), pch=16)  
segments(1,0,5,100,0,5)
```

O Comando “**x=1:100**” gera uma sequência de 1 a 100 e armazena, através, do comando “=” no vetor **x**.

O Comando “**sample**” seleciona uma amostra de tamanho, “size” especificado a partir dos elementos de um conjunto **x**, sendo com ou sem reposição,

sample(x, size, replace = FALSE, prob = NULL)

x conjunto a ser amostrado, vetor de um ou mais elementos de onde se seleciona uma amostra, **size** um número positivo, o número de itens a serem escolhidos, o tamanho da amostra, **replace** se a amostragem é com ou sem reposição, (rep=T ou rep=F), **prob** um vetor de pesos de probabilidade para gerar os elementos do vetor **x**.

O Comando “**cumsum**” retorna um vetor cujos elementos são somas cumulativas.

O Comando “**plot(x,y, ...)**” elabora um gráfico de dispersão do vetor **x** com o vetor **y**, de mesma dimensão. O “,” se refere a argumentos a serem passados na elaboração do gráfico, são parâmetros do gráfico (ver par). Usamos aqui **ylim=c(0,1)**, **xlim=c(0,100)**, que limitam o comprimento dos eixos X (abscisas) e Y (ordenadas).

Os principais argumentos são: a) **type** tipo gráfico a ser desenhado, Tipos possíveis são: “p” para pontos; “l” para linhas; “b” para pontos e linhas (both); “o” tanto para “overplotted”; “h” para linhas verticais de “histograma”; “s” para escada; “n” para não traçar, b) **main**, um título geral para o gráfico, 3) **sub**, um subtítulo, 4) **xlab** e **ylab**, um título para cada eixo x e y, 5) **pch**, tipo de ponto (point character), 6) **lwd**, espessura da linha (line width), 7) **lty**, tipo de linha (line type).

2.2. Simulando o total de pontos em dois lançamentos de dados

São dois jogadores. Em cada jogada, cada jogador lança um dado e somam-se os pontos dos dois dados,

- O Jogador **A** marca um ponto se a soma for 5, 6, 7 ou 8;
- O Jogador **B** marca um ponto se a soma for 2, 3, 4, 9, 10, 11 ou 12.

Qual, entre os jogadores A e B, tem maior probabilidade de ganhar?

Comandos do R para a execução da simulação:

```
soma,dado=sample(1:6,1000,replace=T)+sample(1:6,1000,replace=T)  
table(sample(1:6,1000,replace=T)+sample(1:6,1000,replace=T))/1000
```

O Comando **soma,dado=sample(1:6,1000,replace=T)+sample(1:6,1000,replace=T)** gera a cada sample 1000 jogadas de dois dados, independentes, que são somadas para a construção do vetor, de 1000 posições, armazenado em soma.dado.

O Comando “table”, neste caso, usa os elementos do vetor ou fator de classificação para construir uma tabela das contagens em cada nível do fator. No caso de dupla entrada, o comando “table” constrói uma tabela de contingência das contagens em cada combinação de níveis dos dois fatores.

Apresentamos então quatro rodados destes comando:

x	2	3	4	5	6	7	8	9	10	11	12	P(A)
Fre- quen- tista	0,024	0,055	0,077	0,111	0,131	0,178	0,153	0,106	0,084	0,059	0,022	0,573
	0,040	0,059	0,082	0,115	0,129	0,155	0,143	0,102	0,090	0,058	0,027	0,542
	0,030	0,054	0,078	0,100	0,137	0,155	0,140	0,123	0,086	0,074	0,023	0,532
P(X=x)	0,023	0,047	0,083	0,123	0,137	0,164	0,140	0,122	0,075	0,065	0,021	0,564
Clássico P(X=x)	1/36	2/36	3/36	4/36	5/36	6/36	5/36	4/36	3/36	2/36	1/36	20/36

Solução analítica:

Consideremos o experimento que consiste no lançamento de 2 dados, A variável aleatória X a ser considerada aqui é a soma dos pontos nos 2 dados, O espaço amostral Ω é composto de 36 elementos que são os pares (resultado do 1º lançamento, resultado do 2º lançamento).

É claro que, considerando D1 e D2 lançamentos independentes, todos esses pares têm a mesma probabilidade 1/36 de ocorrer.

Elaboração da Função de Probabilidade do número de pontos no lançamento de 2 dados

D1 D2	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

A variável aleatória X é tal que:

$$X(1,1) = 2,$$

$$X(1,2) = X(2,1) = 3$$

$$X(1,3) = X(2,2) = X(3,1) = 4$$

⋮

$$X(5,6) = X(6,5) = 11$$

$$X(6,6) = 12$$

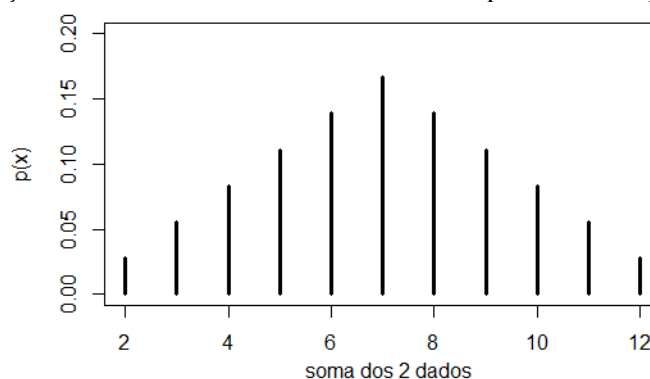
Neste exemplo os valores possíveis de X são 2,3,4,5,6,7,8,9,10,11,12

Então não é difícil concluirmos que a função de probabilidade de X será representada pela tabela:

x	2	3	4	5	6	7	8	9	10	11	12
P(X=x)	1/36	2/36	3/36	4/36	5/36	6/36	5/36	4/36	3/36	2/36	1/36

Onde, $P(A) = P(X=5 \text{ ou } X=6 \text{ ou } X=7 \text{ ou } X=8) = 4/36 + 5/36 + 6/36 + 5/36 = 20/36 = 0.556 > P(B)$ e a função de probabilidade de X terá seu gráfico como abaixo:

Figura -2: A Função de Probabilidade do número total de pontos no lançamento de 2 dados



```
plot(2:12,c(1:6,5:1)/36, type="h", ylim=c(0,20), pch=16, lwd=3,
     xlab="soma dados",ylab="p(x)")
```

2.3. Simulando o Jogo dos Cinco Dados

Lança-se cinco dados. Para ganharmos tem que sair o número 5, mas não pode sair o 6. Qual a probabilidade de ganhar?

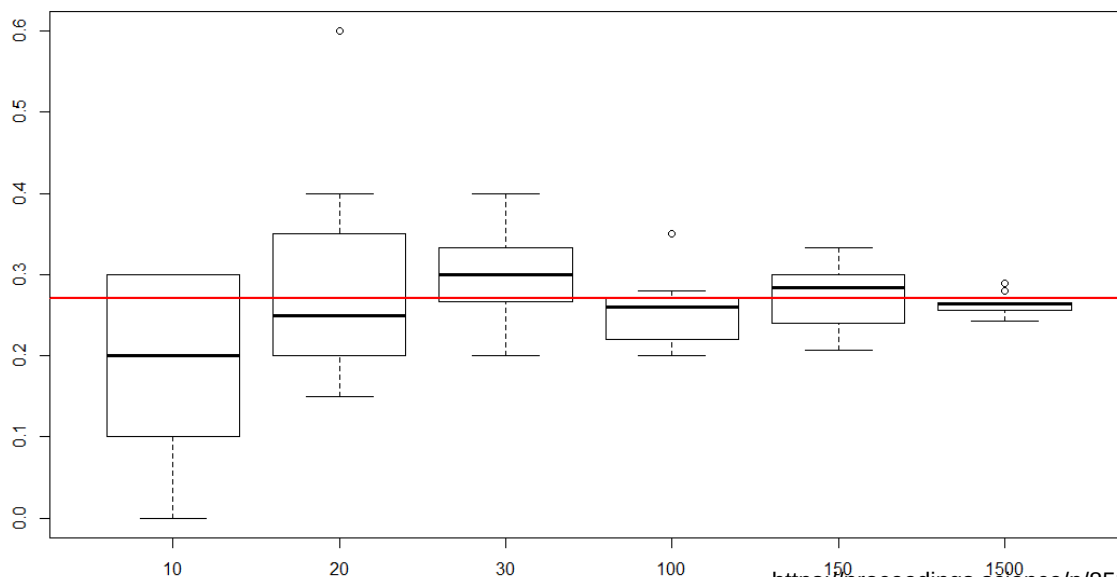
Para executarmos a simulação, várias vezes, para valores diferentes de N (o número de jogadas) fizemos uso de uma função elaborada no R. Foi montada uma amostra (*sample*) que escolhia 5 vezes um número de 1 a 6, com reposição desse número, simulando assim os 5 dados independentes. Foram contados então as ocorrências de cada número e analisado se satisfaziam as condições necessárias. Caso satisfeito marcaria a variável aleatória como **Sucesso (1)** ou **Fracasso (0)**. Esse experimento foi executado N vezes. Ao final, para obtermos o valor da probabilidade associada àquele experimento ocorrer com sucesso precisamos apenas dividir a quantidade de sucessos pela quantidade de experimentos.

```
Jogo.5dados=function(N)
{
  ind=rep(0,N)
  for (j in 1:N) {      # criar uma jogada de 5 dados
    D=sample(1:6, 5, replace=T)
    # verificar de existe 5 e se existe 6 em cada jogada
    i5=0
    i6=0
    for (i in 1:5) {
      if (D[i] == 5) i5=1
      if (D[i] == 6) i6=1
    }
    # verificar se tem 5 e entao se nao tem 6
    if ( (i5==1) & (i6==0) ) ind[j]=1
  }
  sum(ind)/N
}
```

Vamos executar esta função 10 vezes para cada N=10, 20, 30, 100, 150 e 1500. Desta forma vamos observar a variabilidade e a aproximação com o resultado no conceito clássico, a medida que a amostra cresce. Com este objetivo fizemos a sequência de comandos:

```
matriz=matrix(numeric(60),10,6) #inicia com 0's uma matriz 10 linhas por 6 colunas
j=1      # iniciando o índice de coluna em 1
for(N in c(10,20,30,100,150,1500)) {for(i in 1:10) matriz[i,j]=Jogo.5dados(N); j=j+1}
boxplot.matriz(matriz, names=c(10,20,30,100,150,1500)) # boxplot das colunas da matriz
abline(h=0.2702, lwd=2, col="red") # reta horizontal (y=0,2702, prob. clássica)
```

Figura-3: Boxplot do resultado das 10 rodadas da função “jogo.5dados” para vários N’s.



Observamos na figura-3 que, quando N é pequeno, é grande a variabilidade do resultado. A linha vermelha é o resultado calculado pelo método Clássico, no caso de N pequeno a mediana das 10 rodadas fica distante dessa linha, mas quando N cresce, pela regularidade estatística, o valor se estabiliza em 0,2702, probabilidade pelo método clássico.

Solução analítica:

Número de casos possíveis quando lançam-se 5 dados:

$$\underline{6} \underline{6} \underline{6} \underline{6} \underline{6} = 6^5 = 7776$$

Número de casos favoráveis (sair 5, mas **não** sair 6):

Será feito em duas etapas:

1. Não sair 6, só os números de 1 a 5 $\Rightarrow 5^5 = 3125$
2. Não sair 6, mas tem que sair 5,

Dos 3125 subtrair os casos em que também não sai 5,

Números de 1 a 4 $\Rightarrow 4^5 = 1024$

Casos favoráveis $\Rightarrow 3125 - 1024 = 2101$

$$P(\text{ganhar}) = \frac{2101}{7766} = 0,27019$$

2.4 Simulando o jogo de Poker com dados

No jogo de Poker com dados lança-se os cinco dados simultaneamente, os possíveis resultados do jogo estão expressos nos eventos:

- A= um par e os outros diferentes;
- B= dois pares;
- C= trinca;
- D= todos dados diferentes
- E= quatro dados iguais;
- F= cinco dados iguais; e
- G= uma trinca e um par (fullhouse),

Desejamos calcular as probabilidades desses 7 eventos.

Elaborando os códigos de experimentação

Os algoritmos elaborados para verificar as probabilidades apresentadas seguiram a mesma idéia do “jogo.5dados”: Aqui foi montada uma amostra (*sample*) no R que escolhia 5 vezes um número de 1 a 6, com reposição desse número, simulando assim os 5 dados independentes. Foram contados então as ocorrências de cada número e analisado se satisfaziam as condições necessárias daquele item, na letra *A* por exemplo se um determinado número aparecia duas vezes. Caso satisfeito marcaria a variável aleatória correspondente ao status daquele experimento como *Sucesso (1)* ou *Fracasso (0)*. Esse experimento foi executado N vezes. Ao final, para obtermos o valor da probabilidade associada àquele experimento ocorrer com sucesso precisamos apenas dividir a quantidade de sucessos pela quantidade de experimentos. De maneira mais didática, vamos exemplificar o caso onde queremos saber a probabilidade associada à ocorrência de um par.

```

Poker.1par=function(N) {
  ind=numeric(N)
  for (j in 1:N) {
    # criar uma jogada de 5 dados
    D=sample(1:6, 5, replace=T)
    # declarar variável para a quantidade de ocorrência de cada numero e para o # de pares
    i1=0; i2=0; i3=0; i4=0; i5=0; i6=0;
    par=0
    # iteração para contar a quantidade de cada número resultante
    for (i in 1:5) {
      if (D[i] == 1) i1 = i1+1;    if (D[i] == 2) i2 = i2+1;    if (D[i] == 3) i3 = i3+1
      if (D[i] == 4) i4 = i4+1;    if (D[i] == 5) i5 = i5+1;    if (D[i] == 6) i6 = i6+1
    }
    # iteração para contar a quantidade de pares
    if (i1 == 2) par = par + 1;    if (i2 == 2) par = par + 1;    if (i3 == 2) par = par + 1
    if (i4 == 2) par = par + 1;    if (i5 == 2) par = par + 1;    if (i6 == 2) par = par + 1
    # verificando se não há 3, 4, ou 5 números iguais
    if ( (i1>2) | (i2>2) | (i3>2) | (i4>2) | (i5>2) | (i6>2) ) par = 10
    # verificar se é apenas 1 par
    if ( par==1 ) ind[j]=1
  }
  sum(ind)/N
}

```

Código em R para verificar a probabilidade associada à ocorrência de 1 par

O código acima descreve um método para o experimento do jogo de 5 dados de poker, que verifica apenas a ocorrência associada a um par.

O comando `numeric(N)` atribui valor 0 ao vetor `ind` em `N` posições. A variável `ind` será responsável por armazenar os valores 1 para sucesso e 0 para fracasso. O primeiro “for” será responsável por realizar os `N` experimentos. O comando `sample()` irá simular as jogadas dos 5 dados. Como explicado anteriormente seus parâmetros 1:6 representam os valores que o dado pode assumir, o segundo parâmetro de valor 5 é o número de dados a serem lançados, e `replace = T` significa que há reposição ao sortear um número, garantido assim a independência entre os lançamentos dos dados.

A seguir foram declaradas variáveis para cada valor possível do dado, a fim de contar o número de vezes que cada número apareceu no dado nas cinco jogadas. A variável *par* é responsável por contar o número de ocorrências de que dois números saíam iguais.

Estabelecido o ambiente para o experimento, devem ser avaliados os números sorteados na amostra. Assim, o segundo “for”, de 1 a 5 representa a verificação das 5 amostras, ou 5 lançamentos de dado, que incrementa a variável correspondente ao número sorteado em cada lançamento. Em seguida, devem ser analisados os números sorteados e contados quantos pares foram obtidos, que é o que a próxima sequência de “if s” após o fim do “for” anterior representa.

Por fim para validar se não há mais de um par, o que deve ser feito é checar se nenhum número passou de 2 vezes e caso isso ocorra, na variável “par” será armazenado um valor absurdo, no caso escolhemos o valor 10. Então caso haja apenas um par `ind[j]` será marcado como sucesso.

Para avaliar a probabilidade de haver apenas um par, devemos então somar todos os valores de “ind” e dividir pelo número de experimentos, como fazemos na última linha.

Um código geral para a elaboração dos outros experimentos, de B a G, pode ser encontrado no apêndice ou no link do Dropbox e não será discutido aqui, pois, os outros experimentos seguem a mesma linha de raciocínio. O link é:

<https://www.dropbox.com/sh/uphzjzlx7tcnw6q/AABUM7khFIX1xZ-9sAqALhCna?dl=0>

O resultado de uma simulação pode ser visto abaixo. Para cada experimento (lançamento dos 5 dados) rodamos 100 vezes esse experimento para obtermos uma probabilidade baseada nesses 50 lançamentos. Então, executamos 100 vezes esse conjunto de 50 lançamentos a fim de realizarmos uma análise exploratória dessas probabilidades. A lista de comando usada no R foi.

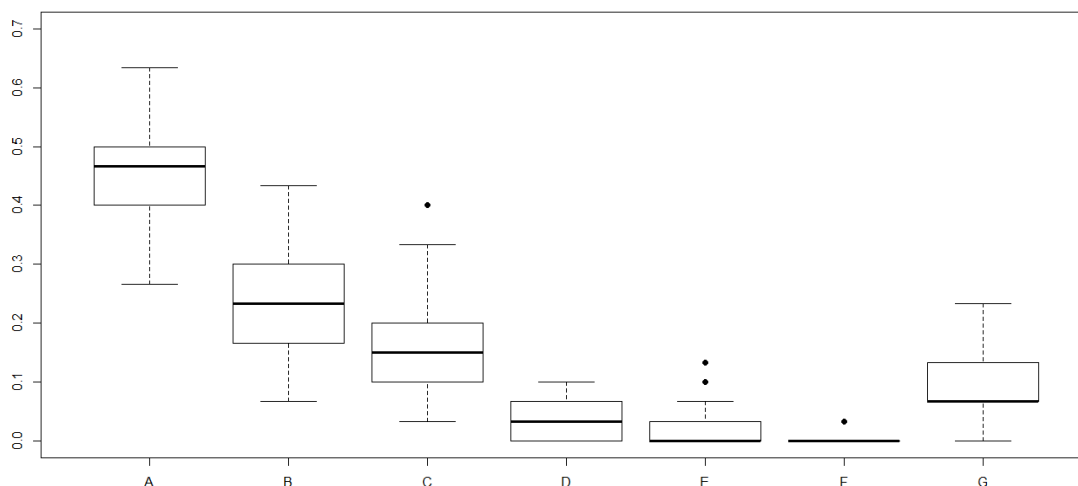
```
N=50; M=100
res= matrix(numeric(M*7),M,7)
for (i in 1:M) res[i,]= poker.5dados(N)
#
mat.r= matrix(numeric(7*8),7,8)
for (i in 1:7) {
  mat.r[i,1:6]=summary(res[,i])
  mat.r[i,7:8]= c(sd(res[,i]), IQR(res[,i]))
}
P.Cl=c(.4630,.2315,.1543,.0926,.0193,.0008,.0386)
mat.fim=cbind(P.Cl,round(mat.r, 4))
rownames(mat.fim)=LETTERS[1:7]
colnames(mat.fim)= c("Prob.", "Min", "Q1", "Q2", "Média", "Q3", "Max", "DP", "DIQ")
round(mat.fim,4)
```

Os resultados desse experimento pode ser observado abaixo:

	Prob.	Min	Q1	Q2	Media	Q3	Max	DP	DIQ
A	0.4630	0.2667	0.4000	0.4667	0.4533	0.5000	0.6333	0.0825	0.1000
B	0.2315	0.0667	0.1667	0.2333	0.2437	0.3000	0.4333	0.0780	0.1333
C	0.1543	0.0333	0.1000	0.1500	0.1597	0.2000	0.4000	0.0671	0.1000
D	0.0926	0.0000	0.0000	0.0333	0.0323	0.0667	0.1000	0.0326	0.0667
E	0.0193	0.0000	0.0000	0.0000	0.0213	0.0333	0.1333	0.0313	0.0333
F	0.0008	0.0000	0.0000	0.0000	0.0007	0.0000	0.0333	0.0047	0.000

Fizemos também uma análise gráfica destas probabilidades através do Boxplot. Os comandos que usamos no R foram:

```
boxplot(res[,1],res[,2],res[,3],res[,4],res[,5],res[,6], res[,7],
        names=LETTERS[1:7],pch=16)
```



Solução analítica:

1

- Um par e os outros diferentes: De uma maneira geral, podemos realizar os cálculos que envolvem pares e trincar da seguinte forma: De um lado, colocaremos todas as combinações necessárias para representarmos a **escolha dos números**, No caso de apenas um par, temos $\binom{6}{1}$ para o número que será o par, e $\binom{5}{1}, \binom{4}{1}$ e $\binom{3}{1}$ para a escolha dos números restantes, Do outro lado, colocaremos as **escolhas de posicionamento** referente aos números já obtidos, Assim, temos $\binom{5}{2}$ para os números dos pares e $\binom{3}{1}, \binom{2}{1}$ e $\binom{1}{1}$, para os demais números, Seguindo desta maneira, obtemos :

$$\#(A) = \binom{6}{1} \times \binom{5}{1} \times \binom{4}{1} \times \binom{3}{1} \times \binom{5}{2} \times \binom{3}{1} \times \binom{2}{1} \times \binom{1}{1} = 3600.$$

Podemos seguir o mesmo raciocínio para os demais problemas com relação ao cálculo dos números favoráveis:

- Dois pares : $\#(B) = \binom{6}{2} \times \binom{4}{1} \times \binom{5}{2} \times \binom{3}{2} = 1800$
- Uma trinca:: $\#(C) = \binom{6}{1} \times \binom{5}{1} \times \binom{4}{1} \times \binom{5}{3} = 1200$
- Uma trinca e um par : $\#(D) = \binom{6}{1} \times \binom{5}{1} \times \binom{5}{3} = 300$

5. Quatro iguais : $\#(E) = \binom{6}{1} \times \binom{5}{1} \times \binom{5}{4} = 150$
6. Cinco iguais : $\#(F) = \binom{6}{1} = 6$
7. Todos os dados diferentes : Queremos 5 números distintos entre 6, Como há $\binom{6}{5}$ possibilidades de escolher 5 números diferentes de 1 a 6, e 5! maneiras diferentes de ordenação possíveis para cada um dos 5 dados, temos, $\#(G) = \binom{6}{5} \times 5! = 720$

A tabela resume os resultados da contagem e apresenta as probabilidades, método Clássico:

Evento	A	B	C	D	E	F	G
$\#(.)$	3600	1800	1200	300	150	6	720
$P(.) = \#(.) / 6^5$	0,4630	0,2315	0,1543	0,0386	0,0193	0,0008	0,0926

3. Conclusão:

O software R se mostrou bastante eficiente nos cálculos das probabilidades, via método frequentista, nas simulações dos quatro problemas apresentados. Usamos também a elaboração de gráficos, que nos ajudou a visualizar a estabilização das possibilidades a medida que o tamanho de realizações do experimento cresce.

O entendimento do conceito Frequentista de Probabilidade, Método de Monte Carlo, contribui nos modelos usados na Pesquisa Operacional e o R é um significativo elemento neste contexto.

4. Referências:

- ROSS, S; **A first course in Probability**, Pearson; 2010.
- DE GROOT, M. H; SCHERVISH M. J. **Probability and Statistics**, Addison-Wesley; 2011.
- PINHEIRO, J. I. D.; CARVAJAL, S. S. R.; CUNHA, S. B.; GOMES. G. C.;. **Estatística básica – A arte de trabalhar com dados**, Elsevier; 2015.
- GOMES. G. C.;, PINHEIRO, J. I. D.; CARVAJAL, S. S. R.; CUNHA, S. B., **O software R como instrumento de ensino em Estatística Básica**, Minicurso do SBPO 2010,
- PINHEIRO, J. I. D.; CARVAJAL, S. S. R.; CUNHA, S. B.; GOMES. G. C.; **Probabilidade e Estatística: quantificando a incerteza**; Editora Campus, 2012.
- GABRICH. R. e GOMES, C.G. Poster: **Uso do software R em simulação do jogo de Poker de dados: Visão clássica versus frequentista**, - SBPO-2016.
- KABACOFF, R.; **R in Action, Data Analysis and Graphics with R**; Manning Publications Co., 2011.

5. Apêndice: Código completo para o jogo de Poker.

```
poker=function(N) {
  ind_1par=numeric(N); ind_2par=numeric(N); ind_3iguais=numeric(N)
  ind_4iguais=numeric(N); ind_5iguais=numeric(N); ind_fullhouse=numeric(N)
  ind_todosDiferentes=numeric(N)
  for (j in 1:N) {
    # criar uma jogada de 5 dados
    D=sample(1:6, 5, replace=T)

    # declarar variável para a quantidade de ocorrencia de cada numero
    i1=0; i2=0; i3=0;
    i4=0; i5=0; i6=0
    # declarar variavel para o número de realizações de cada evento
    par=0
    trinca=0
    quadra=0
    quina=0
    fullhouse=0

    # iteração para contar a quantidade de cada número resultante
    for (i in 1:5) {
      if (D[i] == 1) i1 = i1+1
      if (D[i] == 2) i2 = i2+1
      if (D[i] == 3) i3 = i3+1
      if (D[i] == 4) i4 = i4+1
      if (D[i] == 5) i5 = i5+1
      if (D[i] == 6) i6 = i6+1
    }

    # iteração para contar a quantidade de pares
    if (i1 == 2) par = par + 1
    if (i2 == 2) par = par + 1
    if (i3 == 2) par = par + 1
    if (i4 == 2) par = par + 1
    if (i5 == 2) par = par + 1
    if (i6 == 2) par = par + 1
    # verificando se não há 3, 4, ou 5 números iguais
    if ( (i1>2) | (i2>2) | (i3>2) | (i4>2) | (i5>2) | (i6>2) ) par = 10

    # verificar se há apenas 1 par
    if ( par==1 ) ind_1par[j]=1

    # verificar se são 2 pares
    if ( par==2 ) ind_2par[j]=1

    # iteração para contar a quantidade de trincas
    if (i1 == 3) trinca = trinca + 1
    if (i2 == 3) trinca = trinca + 1
    if (i3 == 3) trinca = trinca + 1
    if (i4 == 3) trinca = trinca + 1
    if (i5 == 3) trinca = trinca + 1
    if (i6 == 3) trinca = trinca + 1
```

```

# verificando se não há um par além da trinca (o que seria um fullhouse)
if ( (i1==2) | (i2==2) | (i3==2) | (i4==2) | (i5==2) | (i6==2) ) trinca = 10
# verificar se são 3 iguais
if ( trinca==1 ) ind_3iguais[j]=1

# iteração para contar a quantidade de quadras
if (i1 == 4) quadra = quadra + 1
if (i2 == 4) quadra = quadra + 1
if (i3 == 4) quadra = quadra + 1
if (i4 == 4) quadra = quadra + 1
if (i5 == 4) quadra = quadra + 1
if (i6 == 4) quadra = quadra + 1
# iteração para verificar se são 4 iguais e não 5 iguais
if ( (i1==5) | (i2==5) | (i3==5) | (i4==5) | (i5==5) | (i6==5) ) quadra = 2
if ( quadra==1 ) ind_4iguais[j]=1

# iteração para verificar se são 5 iguais
if ( (i1==5) | (i2==5) | (i3==5) | (i4==5) | (i5==5) | (i6==5) ) quina = 1
if ( quina==1 ) ind_5iguais[j]=1

# iteração para verificar o fullhouse
if ( (i1==2) | (i2==2) | (i3==2) | (i4==2) | (i5==2) | (i6==2) ) fullhouse = 2
if ( (i1==3) | (i2==3) | (i3==3) | (i4==3) | (i5==3) | (i6==3) ) fullhouse = fullhouse + 3
if ( fullhouse==5 ) ind_fullhouse[j]=1

# verificar se todos são diferentes, ou seja, já saiu 1 ou 0 de cada número
if ( (i1<2) & (i2<2) & (i3<2) & (i4<2) & (i5<2) & (i6<2) ) ind_todosDiferentes[j]=1
}
sum(ind_1par)/N
sum(ind_2par)/N
sum(ind_3iguais)/N
sum(ind_4iguais)/N
sum(ind_5iguais)/N
sum(ind_fullhouse)/N
sum(ind_todosDiferentes)/N
values <- c(UmPar=sum(ind_1par)/N, DoisPares=sum(ind_2par)/N,
  TresIguais=sum(ind_3iguais)/N, QuatroIguais=sum(ind_4iguais)/N,
  CincoIguais=sum(ind_5iguais)/N, Fullhouse=sum(ind_fullhouse)/N,
  TodosDiferentes=sum(ind_todosDiferentes)/N)
values
}

```