



Ficha2

Escreva uma função que receba o número de dados e o número de faces de cada dado e imprima o histograma relativo a 10,000 lançamentos. Represente o histograma para os seguintes valores: 5d2; 2d5; 4d6; 2d100; 100d2;

```
var f= 4 //faces
var d= 6 //n dados

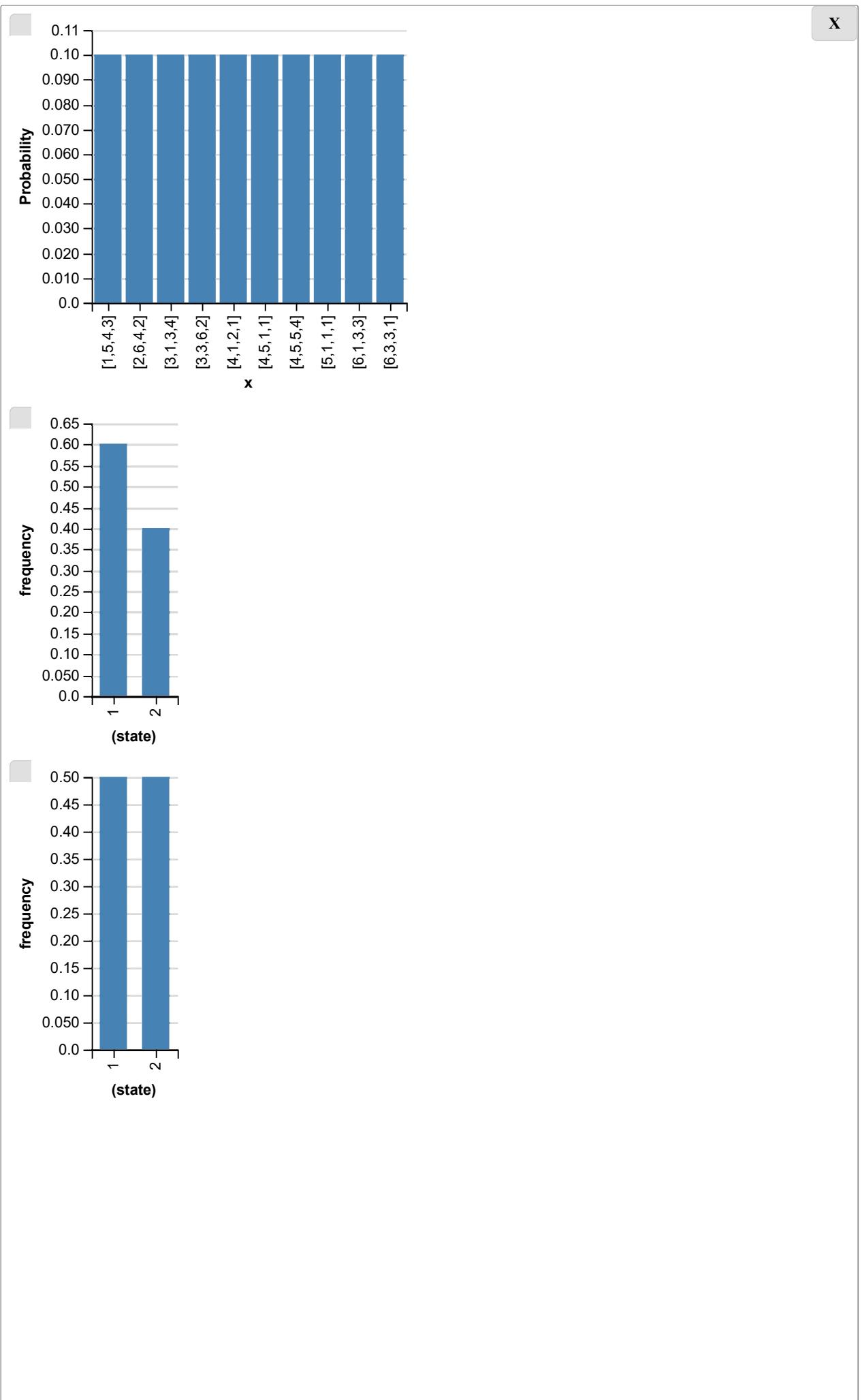
var dado= function(f){
  var aux= function(x){ return x+1}
  var listafaces= mapN(aux,f) //cria se uma lista com f elementos =nºs das faces
  var c= function(){ return categorical({vs:listafaces})} //sai uma face
  return c
}
var dados= function(d,faces){ //repete-se d vezes o lancamento de uma face
  var r= repeat(d,dado(faces))
  return r
}
dados(2,5)
//var roll = function(){return repeat(10,dados(f,d))}

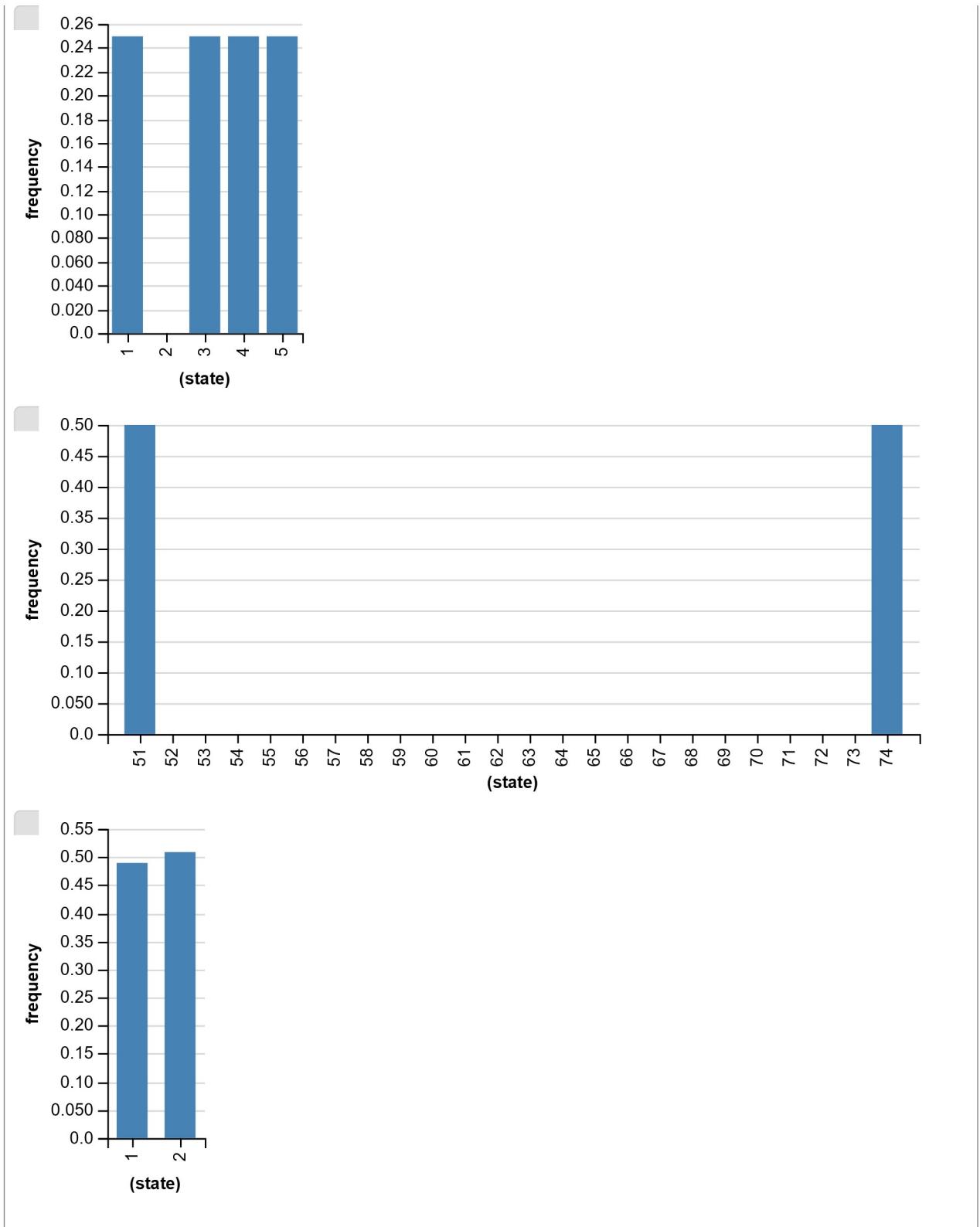
var roll= function(){return dados(f,d)} //para poder ter uma funcao sem argumentos
viz.hist(repeat(10,roll))

viz(dados(5,2))
viz(dados(2,5))
viz(dados(4,6))
viz(dados(2,100))
viz(dados(100,2))
```

run







Crie uma nova função onde os dados repetidos são removidos;

```
var f= 6 //faces  
var d= 3 //n dados  
  
var dado= function(f){  
    var aux= function(x){ return x+1}  
    var listafaces= mapN(aux,f) //cria se uma lista com f elementos =nºs das faces
```

```

var c= function(){ return categorical({vs:listafaces}) } //sai uma face
return c
}

var dados= function(d,faces){ //repete-se d vezes o lancamento de uma face
  var r= repeat(d,dado(faces))
  return r
}

}

var nrep=function(lista){
  console.log("dados originais: ",lista)
  return map(listMean,(groupBy(function(x,y){return x==y},lista)))
}
//a funcao separa em sublistas com elementos iguais- groupby
//a map aplica isso a cada sublista
// a media, que é o proprio numero porque sao todos iguais -listMean

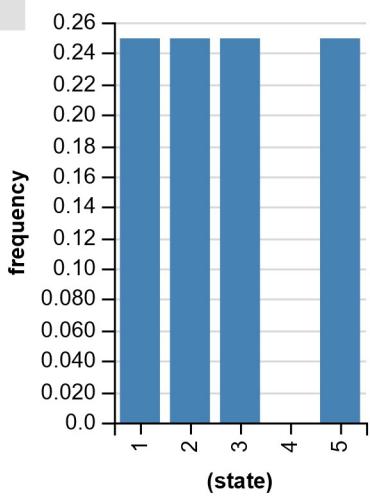
```

run

▼

dados originais: 5,2,1,3,5

X



Crie uma função para o sistema Roll & Keep. Represente o histograma para 1k1; 3k1; 5k1; 7k2; 9k4

```

//exercicio 3
var dado= function(){
  return randomInteger(10)+1
}

var RK=function(X,Y){
  var dados= function(X){ //repete-se d vezes o lancamento de uma face
    return repeat(X,exp(dado))
  }

  var exp= function(dado){ //testa se saiu um 10 e se saiu volta a lancar outro e soma
    if (dado%10==0) {
      var dfinal=dado+randomInteger(10) +1
      return exp(dfinal)
    }
    else {return dado}
  }
}

```

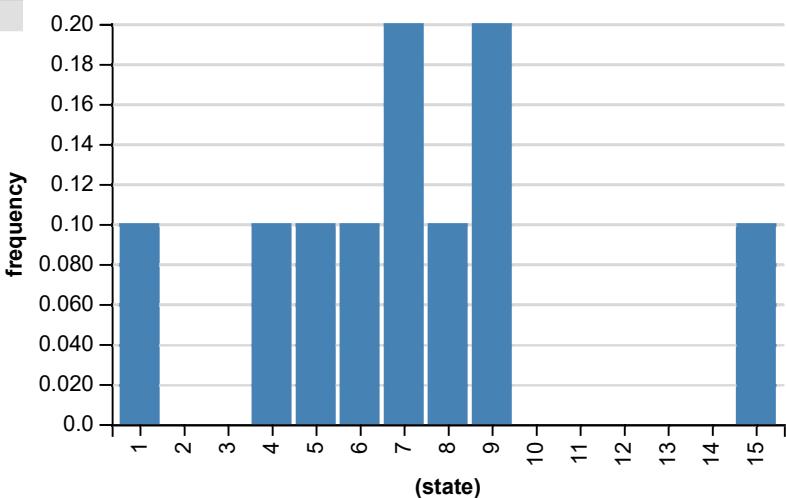
```
}

var f= map(function (x) {return exp(x)},dados(X))
//confirma se cada dado sorteado é um 10, se for aplica exp
return sum(sort(f, gt).slice(0, Y))

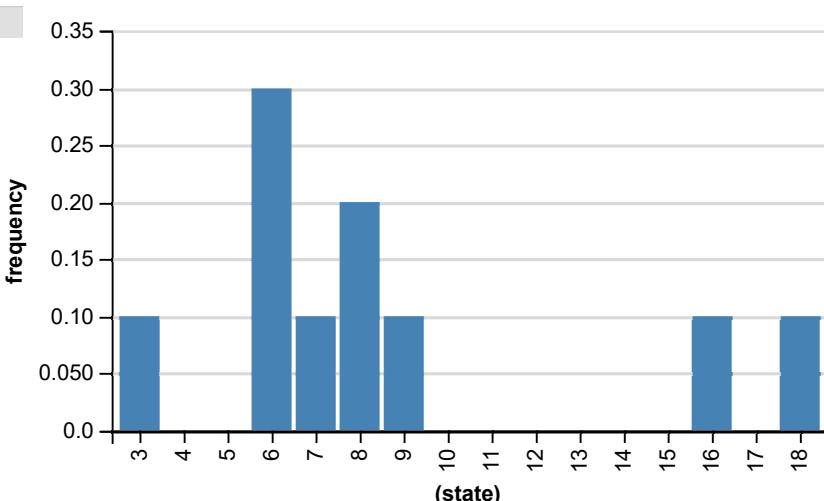
}

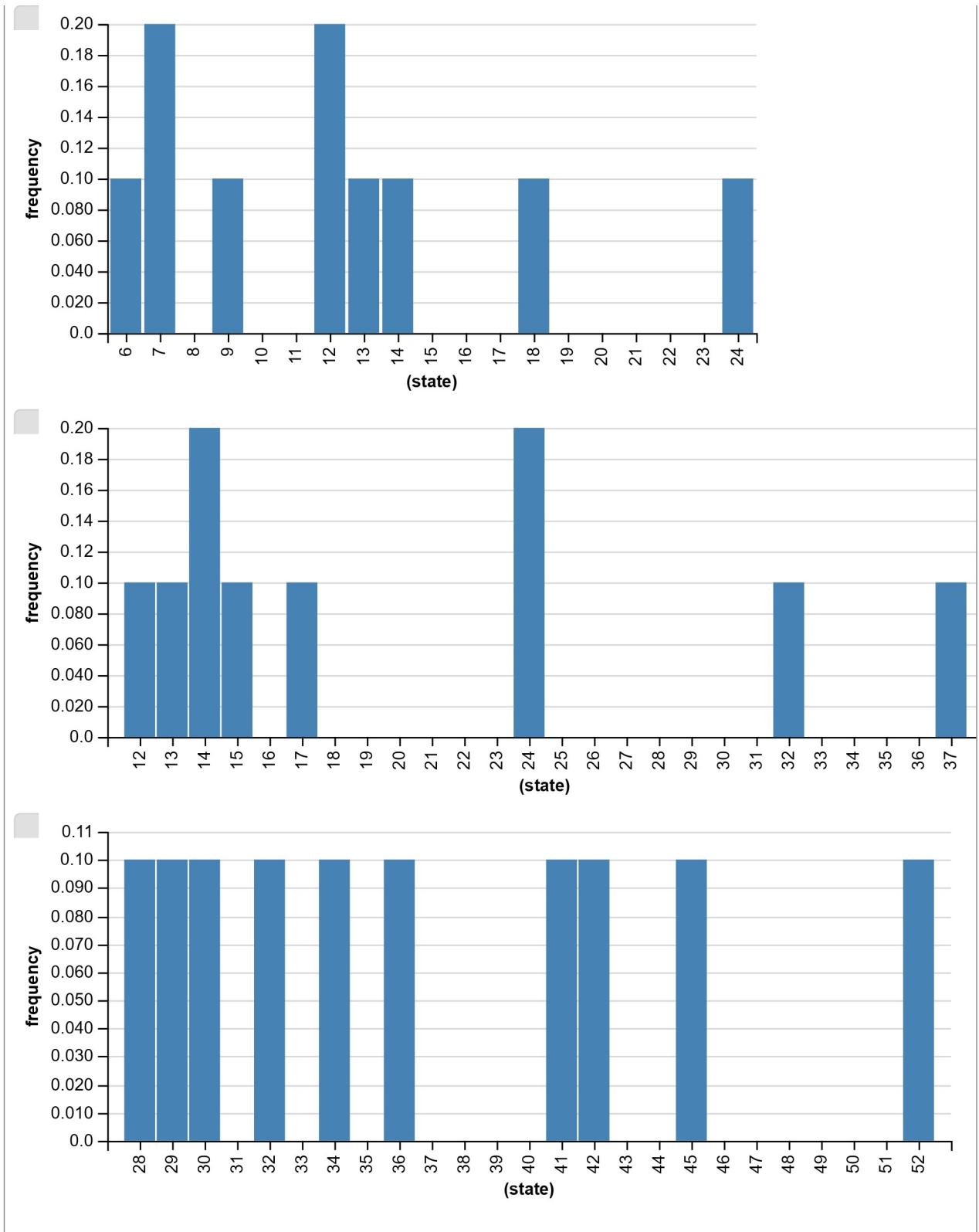
var rk1= function(){return RK(1,1)}
viz(repeat(10,rk1))
var rk2= function(){return RK(3,1)}
viz(repeat(10,rk2))
var rk3= function(){return RK(5,1)}
viz(repeat(10,rk3))
var rk4= function(){return RK(7,2)}
viz(repeat(10,rk4))
var rk5= function(){return RK(9,4)}
viz(repeat(10,rk5))
```

run ▾



X





Crie uma função para o sistema WoD. Represente o histograma do nº de sucessos para vários nºs de dados e TN:

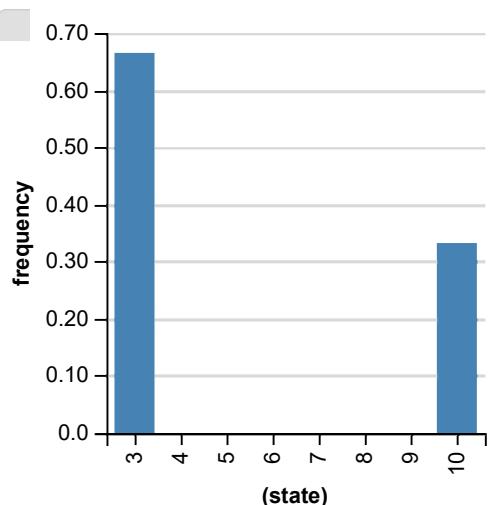
```
// histograma e wod separados para poder reaproveitar codigo se necessario
var TN=5
var d10= function(){ //retorna um nº entre 1 e 10
    return randomInteger(10) +1
}
```

```
var dado= function(f){  
    var aux= function(x){ return x+1}  
    var c= function(){ return categorical({vs:mapN(aux,f)}) } //sai uma face  
    return c  
}  
  
var dados= function(d){ //repete-se d vezes o lancamento de uma face  
    var r= repeat(d,d10)  
    return r  
}  
  
var wod= function(TN,dados){  
    console.log("dados: ",dados)  
    var succ= reduce(function(x,acc)  
        {return acc+1} ,0 ,filter(function(x)  
            { return x > TN; },dados))  
    console.log("succ: ", succ)  
    var fail= dados.length-(remove(1,dados)).length //quantidade de 1's na string  
    console.log("fail: ", fail)  
    var result=succ-fail  
    if (result<0){console.log("Botch")}  
    return result  
  
}  
  
//algoritmo do histograma  
var histg= function(TN,dados){ //cria uma lista  
    console.log("dados: ",dados)  
    var s= dados  
    var s=filter(function(x){ return x > TN; },dados)  
    return s  
}  
  
viz(histg(2,(dados(3))))  
viz(histg(2,(dados(6))))  
viz(histg(0,(dados(2))))  
viz(histg(8,(dados(10))))  
//wod(2,dados(2))  
  
//nota: este codgo nao preve a situaçao em que todos os lançamentos sao rejeitados.  
//isto é, no caso em que devíamos ter um grafico vazio vai dar erro  
//nem nesta alínea nem em toda a ficha
```

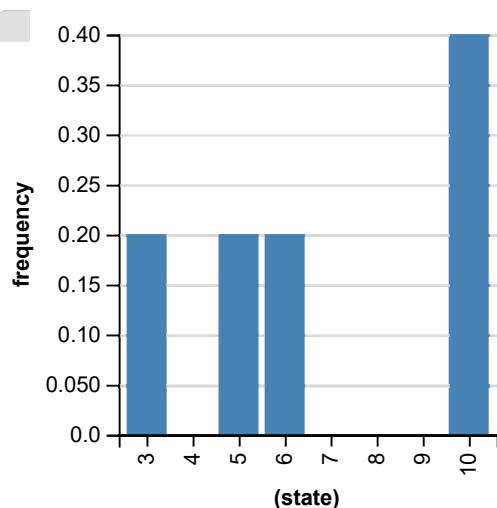
run

dados: 10,3,3

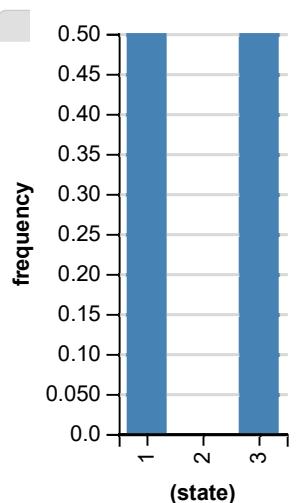
X



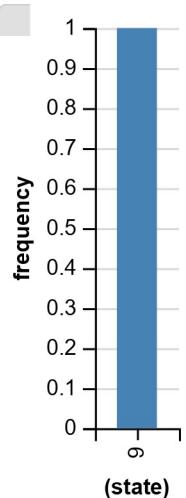
dados: 2,10,10,5,6,3



dados: 1,3



dados: 3,6,5,7,2,6,1,7,9,1



Represente gráficamente o nº de sucessos para os vários TN usando um heatmap quando se lançam 10 dados

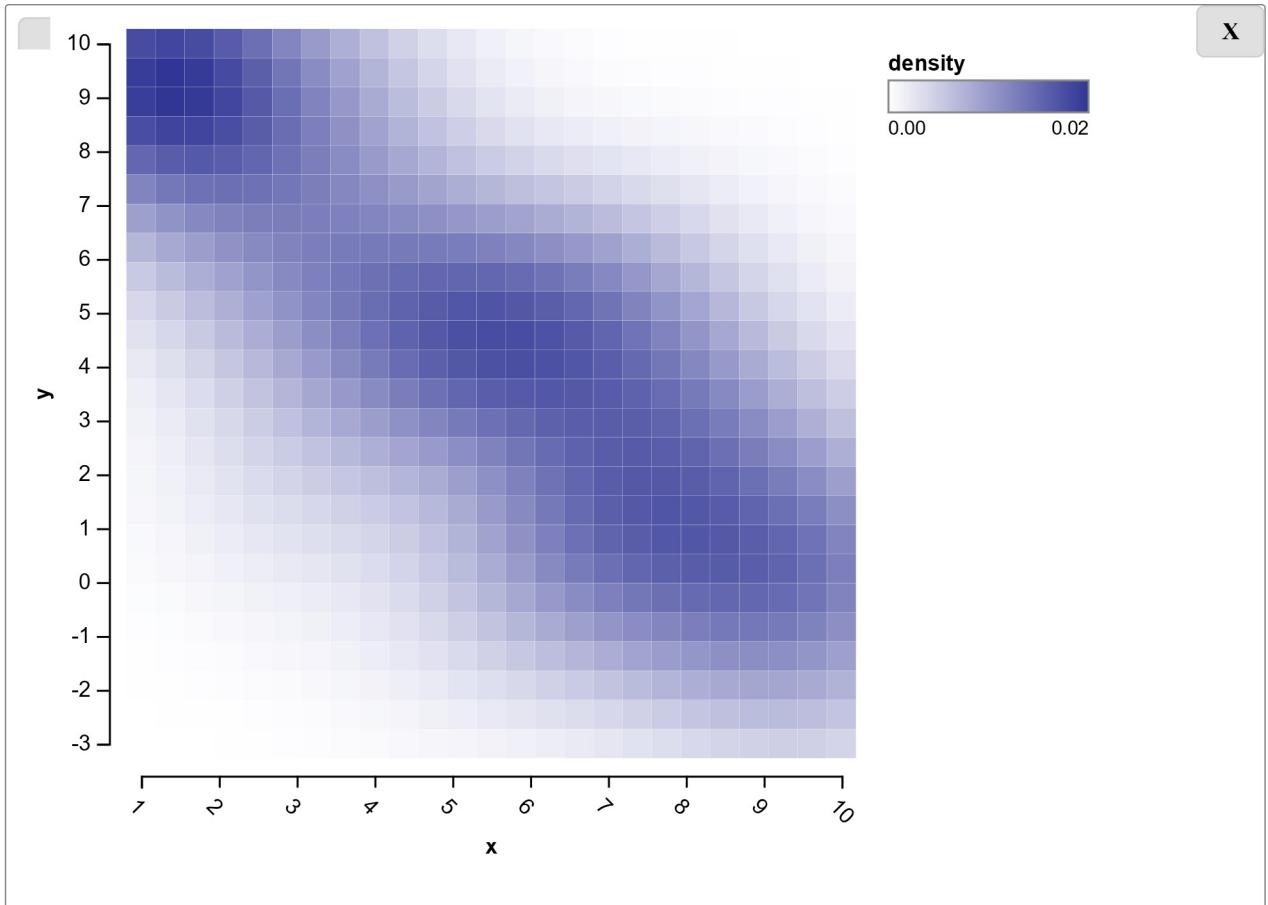
```
var d10=function(){
    return randomInteger(10)+1
}

var sucessos= function(d,TN){ //repete-se d vezes o lancamento de uma face
    var ndados= repeat(d,d10)
    var succ=filter(function(x){return x>=TN},ndados)
    var fails=filter(function(x){return x==1},ndados)
    return succ.length-fails.length
}
var WoD=function(){
    var TN=randomInteger(10)+1
    var ndados=10
    return[TN,sucessos(ndados,TN) ]
}
viz.heatMap(repeat(100,WoD))

//analise grafica
//Espera-se um gráfico com um maior numero de sucessos para TNs pequenos e
```

run

▼



Represente um heatmap com o nº de dados vs o nº de sucessos para o TN de 9;

```

var d10=function(){
    return randomInteger(10)+1
}
var sucessos= function(d,TN){ //repete-se d vezes o lancamento de uma face
    var ndados= repeat(d,d10)
    var succ=filter(function(x){return x>=TN},ndados)
    var fails=filter(function(x){return x==1},ndados)
    return succ.length-fails.length
}

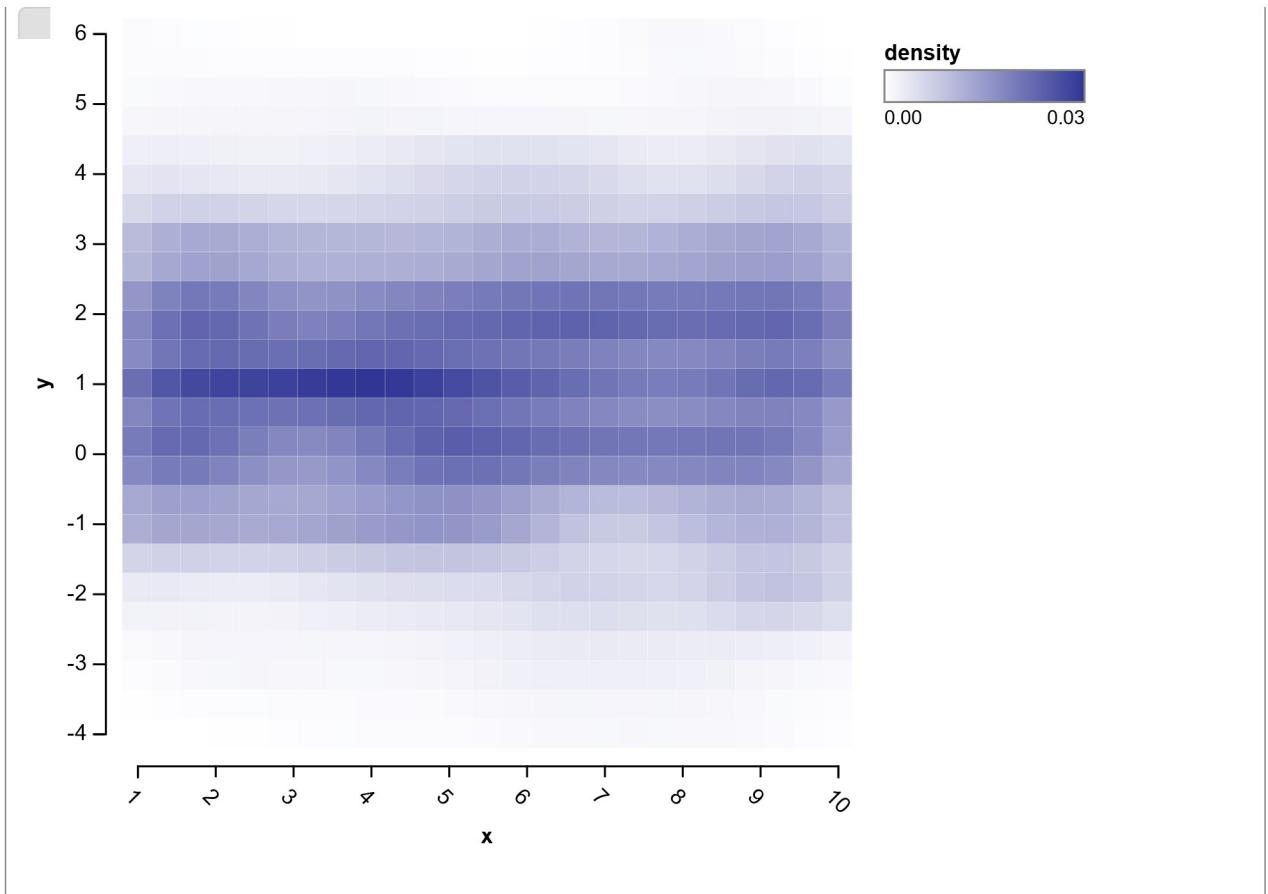
var WoD=function(){
    var ndados=randomInteger(10)+1
    var TN=9
    return[ndados,sucessos(10,TN) ]
}
viz.heatMap(Infer(WoD))

//analise matematica
//estamos a espera de uma função cuja imagem vai estar centrada em numeros pequenos
//ou mesmo no 0 dado a TN ser tao grande.

```

run

Enumerate timeout: max time was set to 5000..quit enumerate
Using "rejection"



Represente graficamente contested rolls de Roll & Keep;
veja o impacto dos Traits e Skills;

```
//exercicio 7_1jogador
var dado= function(){
    return randomInteger(10)+1
}

var RK=function(X,Y){
var dados= function(X){ //repete-se d vezes o lancamento de uma face
    return repeat(X,exp(dado))
}

var exp= function(dado) { //testa se saiu um 10 e se saiu volta a lancar outro e soma
if (dado%10==0) {
    var dfinal=dado+randomInteger(10) +1
    return exp(dfinal)
}
else {return dado}
}
var f= map(function (x) {return exp(x)},dados(X))
//confirma se cada dado sorteado é um 10, se for aplica exp
return sum(sort(f, gt).slice(0, Y))
}

var kfixo=function(){ //gera resultados para um n° fixo de dados guardados
    var ndados=randomInteger(10)+1
    return[ndados,RK(ndados,5)]
}
//viz.heatMap(Infer(kfixo))
```

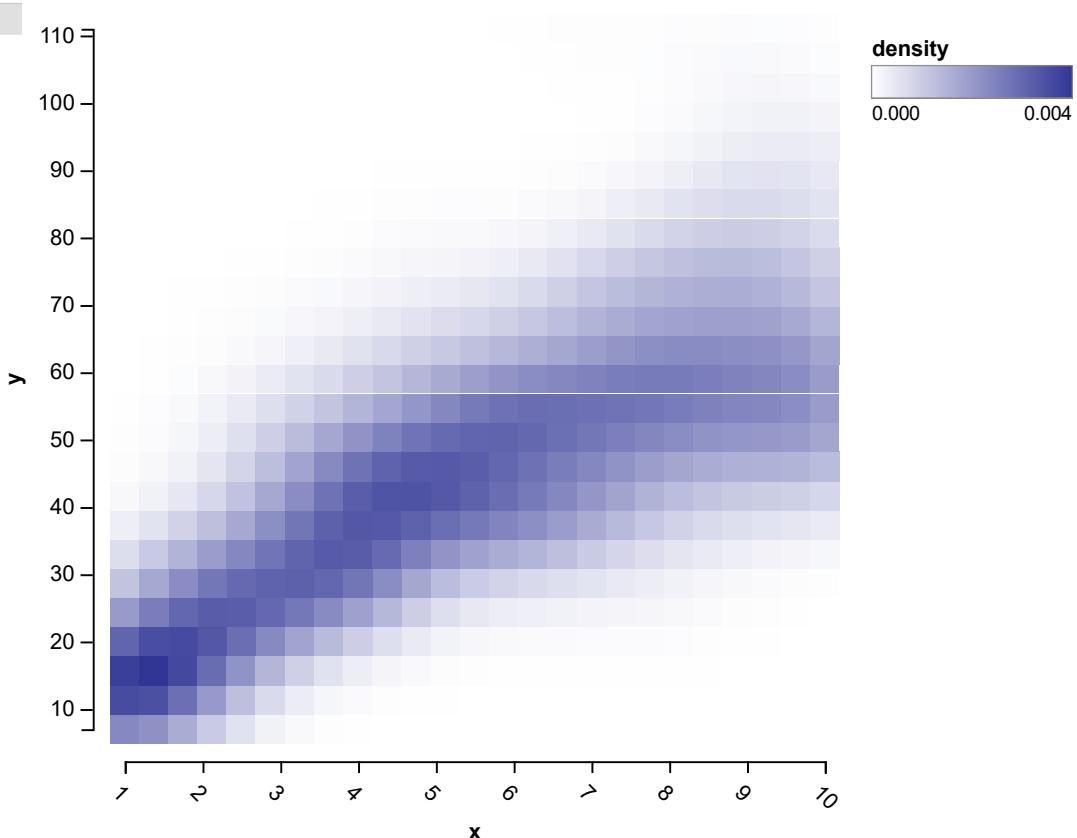
```
var rfixo=function(){ //gera resultados para x lançados e y guardados, ambos aleatorios
  var x=10
  var keep=function(f){
    var aux= function(x){ return x+1}
    var listafaces= mapN(aux,f)
    var c= categorical({vs:listafaces}) //sai uma face
    return c
  }
  var y=keep(x)
  return [y,RK(x,y)]
}
viz.heatMap(Infer(rfixo))
```

run ▾

Enumerate timeout: max time was set to 5000..quit enumerate

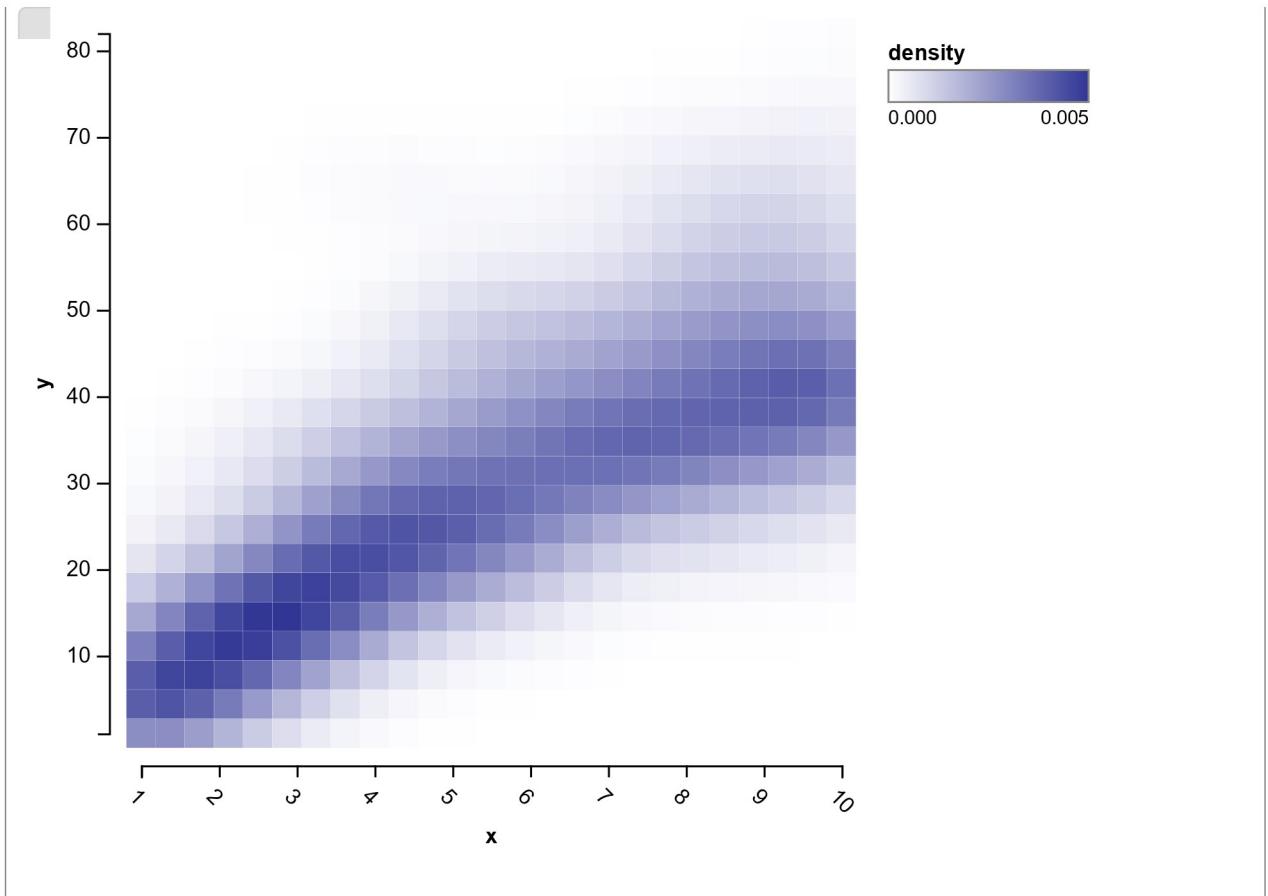
X

Using "rejection"



Enumerate timeout: max time was set to 5000..quit enumerate

Using "rejection"



conclusões para 1 jogador:

Quando fixamos o numero de dados que guardamos e variamos o numero de dados que lançamos observamos o seguinte. Quanto maior o numero de dados é mais provável obter valores mais elevados

Quando fixamos o numero de dados lançados quando variamos o numero de dados que lançamos observamos o seguinte. Quanto maior o numero de dados guardados mais facilmente atingimos um valor alto

```
//exercicio 7_contested
var dado= function(){
    return randomInteger(10)+1
}

var RK=function(X,Y){
var dados= function(X){ //repete-se d vezes o lançamento de uma face
    return repeat(X,exp(dado))
}

var exp= function(dado) { //testa se saiu um 10 e se saiu volta a lançar outro e soma
if (dado%10==0) {
    var dfinal=dado+randomInteger(10) +1
    return exp(dfinal)
}
else {return dado}
}

var f= map(function (x) {return exp(x)},dados(X))
//confirma se cada dado sorteado é um 10, se for aplica exp
return sum(sort(f, gt).slice(0, Y))
}
```

```

var contested_caso1=function(){
  var d1=function(){return RK(4,3)}
  var d2=function(){return RK(5,1)}
  var r=d1()<d2()
  if (r){return 1} else {return 0}
  viz(repeat(100,contested_caso1))

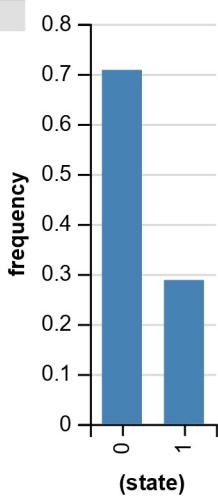
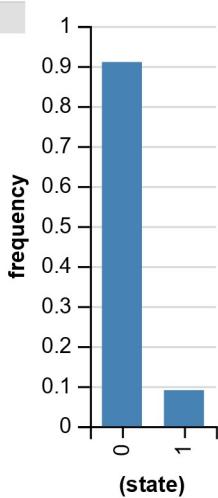
var contested_caso2=function(){
  var d1=function(){return RK(4,3)}
  var d2=function(){return RK(5,2)}
  var r=d1()<d2()
  if (r){return 1} else {return 0}
  viz(repeat(100,contested_caso2))

contested_caso1()

```

run

▼



0

X

Conclusões par 2 jogadores

Podemos ver que em ambos os casos compensa mais lançar mais dados e guardar menos($5k_1/5k_2$) é geralmente maior. Por outro lado compensa mais ter $5k_2$ do que $5k_1$, tal como estavamo a esperar



Represente graficamente contested rolls de WoD; veja o impacto no nº de dados.

```
var d10=function(){
    return randomInteger(10)+1
}

var sucessos= function(d,TN){ //repete-se d vezes o lancamento de uma face
    var ndados= repeat(d,d10)
    var succ=filter(function(x){return x>=TN},ndados)
    var fails=filter(function(x){return x==1},ndados)
    return succ.length-fails.length
}

var WoD=function(){ //se no grafico tiver falso: ganhou j1, verdadeiro: ganhou j2
    var ndados=randomInteger(10)+1
    var TN=9
    return[ndados,sucessos(10,TN) ]
}

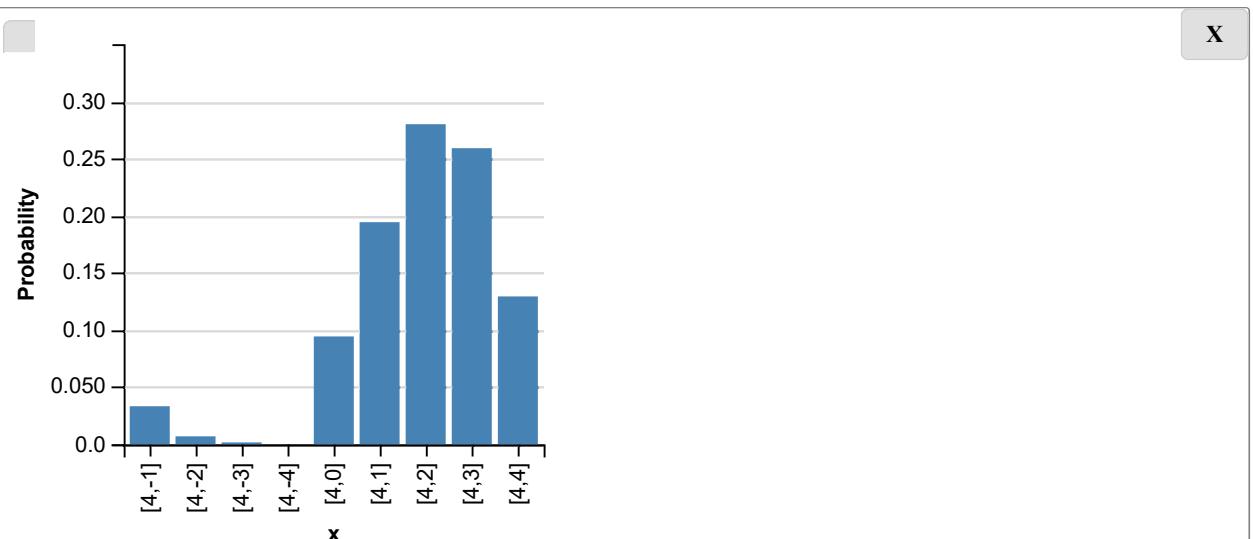
var contested=function(){//cenario completamente aleatorio do qual nao controlamos nada
    var j1=WoD()
    var j2=WoD()
    return j1<j2
}

var contested_cvalors=function(){ //cenario onde forçamos um dos jogadores a ter -1 dado
    var j1=sucessos(4,5)
    var j2=sucessos(2,5)
    return j1<j2
} //mesmo só co um dado de diferença observa-se que ha muitos menos casos verdadeiros

//viz.hist(repeat(10,contested_cvalors))
var f1=function(){return [4,sucessos(4,5)]}
viz.hist(Infer(f1))
var f2=function(){return [10,sucessos(10,5)]}
viz.hist(Infer(f2))
var f3=function(){return [10,sucessos(10,7)]}
viz.hist(Infer(f3))
var f4=function(){return [2,sucessos(2,3)]}
viz.hist(Infer(f4))
```

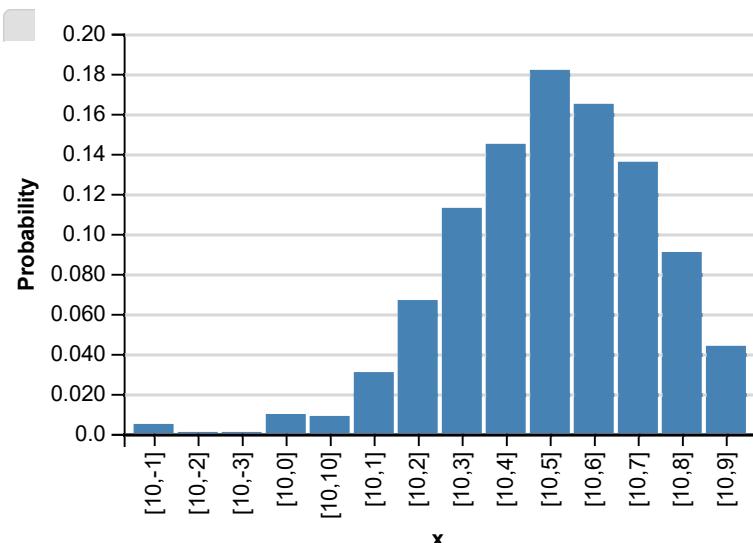
run





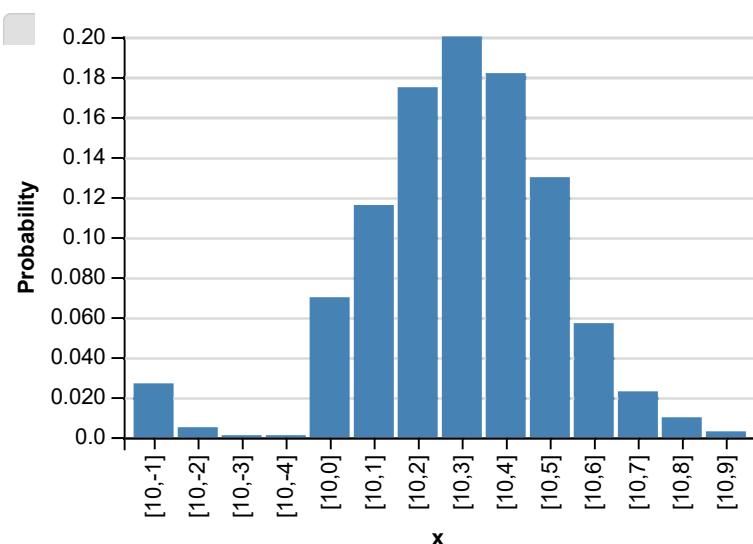
```
Enumerate timeout: max time was set to 5000..quit enumerate
```

Using "rejection"



```
Enumerate timeout: max time was set to 5000..quit enumerate
```

Using "rejection"



Conclusoes:

Obersva-se do primeiro para o segundo graficp que com o aumento dos dados as probabilidades melhoraram ligeiramente. Contudo, nao mudam linearmente uma vez que mais dados quer dizer tambem maior chance de sair 1s.

Por outro lado ao aumentar o TN mesmo que se se aumente aos dados as probabilidades nao sao significativamente melhores (comparando o ultimo grafico ao primeiro)