In [24]:
```
using Printf
using DataFrames
using CSV
using LinearAlgebra
using DataStructures
using ProgressMeter

include("RLalgo.jl")
```

Out[24]:
```
pack_state (generic function with 1 method)
```

initialization of database

In [25]:
```
A = reshape(collect(1:100),(10,10))
```

Out[25]:
```
10×10 Matrix{Int64}:
  1  11  21  31  41  51  61  71  81   91
  2  12  22  32  42  52  62  72  82   92
  3  13  23  33  43  53  63  73  83   93
  4  14  24  34  44  54  64  74  84   94
  5  15  25  35  45  55  65  75  85   95
  6  16  26  36  46  56  66  76  86   96
  7  17  27  37  47  57  67  77  87   97
  8  18  28  38  48  58  68  78  88   98
  9  19  29  39  49  59  69  79  89   99
 10  20  30  40  50  60  70  80  90  100
```

In [26]:
```
infile = "data/small.csv"
df = CSV.File(infile) |> DataFrame
# data_mat = Matrix(df);

# x = [mod(df.s[j], 10)!=0 ? mod(df.s[j], 10) : 10 for j in 1:size(df,1)]
# y = [mod(df.s[j], 10)!=0 ? df.s[j] ÷ 10 + 1 : df.s[j] ÷ 10  for j in 1:size(df,1)

# df = insertcols!(df, 2, :s_i => [x[i] for i in 1:size(df,1)])
# df = insertcols!(df, 3, :s_j => [y[j] for j in 1:size(df,1)])

# xp = [mod(df.sp[j], 10)!=0 ? mod(df.sp[j], 10) : 10 for j in 1:size(df,1)]
# yp = [mod(df.sp[j], 10)!=0 ? df.sp[j] ÷ 10 + 1 : df.sp[j] ÷ 10  for j in 1:size(d

# df = insertcols!(df, 7, :sp_i => [xp[i] for i in 1:size(df,1)])
# df = insertcols!(df, 8, :sp_j => [yp[j] for j in 1:size(df,1)])
```

Out[26]: 50000×4 DataFrame *49975 rows omitted*

Row	s	a	r	sp
	Int64	Int64	Int64	Int64
1	85	3	0	86
2	86	2	0	87
3	87	3	0	97
4	97	2	0	87
5	87	1	0	86
6	86	3	0	76
7	76	4	0	66
8	66	1	0	65
9	65	2	0	66
10	66	3	0	76
11	76	2	0	66
12	66	4	0	56
13	56	4	0	46
⋮	⋮	⋮	⋮	⋮
49989	98	4	0	88
49990	88	1	0	89
49991	89	1	0	79
49992	79	2	0	80
49993	80	3	0	79
49994	79	1	0	78
49995	78	3	0	77
49996	77	2	0	78
49997	78	1	0	77
49998	77	4	0	67
49999	67	1	0	66
50000	66	3	0	76

In [27]:

```
idx = findall(df.r .> 0)
k   = rand([2,19,5,6,10])
```

Out[27]:

```
19
```

Gradient Q-learning (not deep, w/o experience replay)

In [28]:

```
# S = [[x,y] for x in 1:10, y in 1:10]   # FIXME: need to reshape this
S = [i for i in 1:100]
```

```
A = [1,2,3,4]
γ = 0.95
T = NaN
R = NaN
TR = NaN

α = 0.05
Q = zeros((length(S), length(A)))
N = zeros((length(S), length(A)))
prob = MDP(γ,S,A,T,R,TR)
```

Out[28]:

```
MDP(0.95, [1, 2, 3, 4, 5, 6, 7, 8, 9, 10  …  91, 92, 93, 94, 95, 96, 97, 98, 99, 1
00], [1, 2, 3, 4], NaN, NaN, NaN)
```

In [29]:

```
model = QLearning(S,A,γ,Q,α)
```

Out[29]:

```
QLearning([1, 2, 3, 4, 5, 6, 7, 8, 9, 10  …  91, 92, 93, 94, 95, 96, 97, 98, 99, 1
00], [1, 2, 3, 4], 0.95, [0.0 0.0 0.0 0.0; 0.0 0.0 0.0 0.0; … ; 0.0 0.0 0.0 0.0;
0.0 0.0 0.0 0.0], 0.05)
```

In [ ]:

```

```

In [35]:

```
@time train_offline_simple(prob, model, df, 1)
```

```
data size: 50000
  0.030614 seconds (847.99 k allocations: 16.754 MiB)
```

In [31]:

```
display("text/plain", model.Q)
```

```
100×4 Matrix{Float64}:
 0.154771  0.278654  0.239608  0.164459
 0.273725  0.399008  0.418518  0.251273
 0.600643  0.994452  1.00351   0.560868
 1.1107    1.85738   2.00993   1.18526
 1.95401   2.47358   3.18484   2.17818
 3.71498   4.12302   4.56231   3.15673
 5.34712   5.41774   7.09655   5.41958
 6.09717   3.23372   5.70245   3.41182
 3.78218   2.27063   3.4092    2.16855
 2.25264   1.47906   2.34337   1.69174
 0.289651  0.543547  0.56603   0.326835
 0.427921  0.99021   0.889068  0.401677
 1.2057    2.04963   1.4798    1.26983
 ⋮
 1.07157   0.54452   0.475539  1.1299
 0.534949  0.341449  0.329837  0.642675
 3.61518   4.397     3.77607   4.3425
 4.63553   4.15585   4.73292   4.94717
 3.99812   2.62384   3.39032   3.74438
 2.4913    1.89872   1.87462   2.47631
 1.44576   1.11254   1.11112   1.41319
 1.0769    0.746249  0.615632  1.03643
 0.678196  0.597754  0.459029  0.832629
 0.577799  0.372764  0.409338  0.797802
 0.444206  0.281472  0.196917  0.436348
 0.285971  0.173983  0.129712  0.255363
```

In [ ]:

```

```

In [32]:

```
A = reshape(collect(1:100),(10,10))
```

Out[32]:

```
10×10 Matrix{Int64}:
  1  11  21  31  41  51  61  71  81   91
  2  12  22  32  42  52  62  72  82   92
  3  13  23  33  43  53  63  73  83   93
  4  14  24  34  44  54  64  74  84   94
  5  15  25  35  45  55  65  75  85   95
  6  16  26  36  46  56  66  76  86   96
  7  17  27  37  47  57  67  77  87   97
  8  18  28  38  48  58  68  78  88   98
  9  19  29  39  49  59  69  79  89   99
 10  20  30  40  50  60  70  80  90  100
```

In [33]:

```
a_opt = [findmax(model.Q[x, :])[2] for x in 1:100]
# a_opt2 = reshape(a_opt, (10,10))
```

Out[33]:

```
100-element Vector{Int64}:
 2
 3
 3
 3
 3
 3
 3
 1
 1
 3
 3
 2
 2
 ⋮
 4
 4
 2
 4
 1
 1
 1
 1
 4
 4
 1
 1
```

In [34]:

```
file = open("small.policy", "w")

# Write each element of the vector to the file on a new line
for element in a_opt
    println(file, element)
end

# Close the file
close(file)
```

In [ ]:

In [ ]:

In [ ]: