In [1]:
```
using Printf
using DataFrames
using CSV
using LinearAlgebra
# using DataStructures

include("RLalgo.jl")
```

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

In [ ]:
```

```

In [2]:
```
infile = "data/medium.csv"
df = CSV.File(infile) |> DataFrame
# data_mat = Matrix(df);

# df = insertcols!(df, 2, :pos => [extract_state(df.s[i])[1] for i in 1:siz
# df = insertcols!(df, 3, :vel => [extract_state(df.s[i])[2] for i in 1:siz

# df = insertcols!(df, 7, :pos_ => [extract_state(df.sp[i])[1] for i in 1:s
# df = insertcols!(df, 8, :vel_ => [extract_state(df.sp[i])[2] for i in 1:s
```

Out[2]: 100000×4 DataFrame *99975 rows omitted*

Row	s	a	r	sp
	Int64	Int64	Int64	Int64
1	25203	1	-225	24703
2	24703	3	-25	24703
3	24703	1	-225	24202
4	24202	1	-225	24202
5	24202	2	-100	23701
6	23701	3	-25	23700
7	23700	1	-225	23699
8	23699	3	-25	23198
9	23198	4	0	23697
10	23697	3	-25	23196
11	23196	6	-100	23695
12	23695	1	-225	23694
13	23694	3	-25	23693
⋮	⋮	⋮	⋮	⋮
99989	43129	6	-100	44143
99990	44143	6	-100	45157
99991	45157	4	0	45171
99992	45171	6	-100	46186
99993	46186	6	-100	46701
99994	46701	5	-25	46717
99995	46717	5	-25	46732
99996	46732	4	0	46747
99997	46747	5	-25	46263
99998	46263	5	-25	45777
99999	45777	3	-25	45291
100000	25343	1	-225	24720

In [3]:
```
# S = [[pos,vel] for pos in 1:500, vel in 1:100]   # FIXME: need to reshape
S = [i for i in 1:50000]
A = collect(1:7)
γ = 1
T = NaN
R = NaN
TR = NaN

prob = MDP(γ,S,A,T,R,TR)
```

Out[3]:
```
MDP(1, [1, 2, 3, 4, 5, 6, 7, 8, 9, 10  …  49991, 49992, 49993, 49994, 499
95, 49996, 49997, 49998, 49999, 50000], [1, 2, 3, 4, 5, 6, 7], NaN, NaN,
NaN)
```

In [4]:
```
α = 0.01
Q = zeros((length(S), length(A)))

model = QLearning(S,A,γ,Q,α)
```

Out[4]:
```
QLearning([1, 2, 3, 4, 5, 6, 7, 8, 9, 10  …  49991, 49992, 49993, 49994,
49995, 49996, 49997, 49998, 49999, 50000], [1, 2, 3, 4, 5, 6, 7], 1, [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], 0.01)
```

In [ ]:
```

```

In [5]:
```
@time train_offline_simple(prob, model, df,3)
```

```
data size: 100000
```

```
Progress: 100%|████████████████████████████████████████| Time: 0:00:00
```

```
data size: 100000
data size: 100000
  1.267081 seconds (9.17 M allocations: 252.743 MiB, 3.16% gc time, 80.8
7% compilation time)
```

In [6]:
```
# a_opt = [findmax([model.Q(model.ϑ, s, a) for a in A])[2] for s in [[pos,v
```

In [7]:
```
# model.ϑ
```

In [ ]:
```

```

In [10]:
```
a_opt = [findmax(model.Q[x, :])[2] for x in 1:50000];
```

In [9]:
```
file = open("medium.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 [ ]:
```

```