In [23]:
```
using Printf
using DataFrames
using CSV
using LinearAlgebra
using ProgressMeter
# using DataStructures

include("RLalgo.jl")
```

Out[23]:
```
pack_state (generic function with 1 method)
```

In [ ]:
```

```

In [24]:
```
infile = "data/large.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[24]: 100000×4 DataFrame *99975 rows omitted*

Row	s	a	r	sp
	Int64	Int64	Int64	Int64
1	291311	3	0	291312
2	291312	9	0	291312
3	291312	1	5	291212
4	291212	7	0	291212
5	291212	6	0	291212
6	291212	9	0	291212
7	291212	9	0	291212
8	291212	2	0	291213
9	291213	4	0	291113
10	291113	8	0	291113
11	291113	4	0	291114
12	291114	8	0	291114
13	291114	4	0	291214
⋮	⋮	⋮	⋮	⋮
99989	291203	7	0	291203
99990	291203	2	0	291202
99991	291202	4	0	291302
99992	291302	3	0	291202
99993	291202	1	0	291203
99994	291203	4	0	291303
99995	291303	1	0	291304
99996	291304	3	0	291204
99997	291204	2	0	291304
99998	291304	2	0	291303
99999	291303	3	0	291203
100000	291203	3	0	291103

In [25]:
```
# S = [[pos,vel] for pos in 1:500, vel in 1:100]   # FIXME: need to reshape
S = [i for i in 1:312020 ]
A = collect(1:9)
γ = 0.95
T = NaN
R = NaN
TR = NaN

prob = MDP(γ,S,A,T,R,TR)
```

Out[25]:
```
MDP(0.9, [1, 2, 3, 4, 5, 6, 7, 8, 9, 10  …  312011, 312012, 312013, 31201
4, 312015, 312016, 312017, 312018, 312019, 312020], [1, 2, 3, 4, 5, 6, 7,
8, 9], NaN, NaN, NaN)
```

In [26]:
```
α = 0.01
Q = zeros((length(S), length(A)))

model = QLearning(S,A,γ,Q,α)
```

Out[26]:
```
QLearning([1, 2, 3, 4, 5, 6, 7, 8, 9, 10  …  312011, 312012, 312013, 3120
14, 312015, 312016, 312017, 312018, 312019, 312020], [1, 2, 3, 4, 5, 6,
7, 8, 9], 0.9, [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 [35]:
```
@time train_offline_simple(prob, model, df,10)
```

```
data size: 100000
data size: 100000
data size: 100000
data size: 100000
data size: 100000
data size: 100000
data size: 100000
data size: 100000
data size: 100000
data size: 100000
  0.729075 seconds (24.98 M allocations: 488.033 MiB, 7.03% gc time)
```

In [28]:
```
# a_opt = [findmax([model.Q(model.ϑ, s, a) for a in A])[2] for s in [[pos,v
```

In [29]:
```
# model.ϑ
```

In [ ]:
```

```

In [33]:
```
a_opt = [findmax(model.Q[x, :])[2] for x in 1:312020]
```

Out[33]:
```
312020-element Vector{Int64}:
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 ⋮
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
```

In [34]:
```
file = open("large.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 [ ]:

In [ ]:

In [ ]: