

There are 20 states: **s0** to **s19**. There are only three actions that an agent can take in any one state: the agent can move **up**, **down**, or **left** (but cannot move right). Rewards associated with entering all states are **0** (zero) except for state **s0**, the goal, which has a reward of **+10** and **s1**, a punishing end state, which has a negative reward of **-10**. **s3** is the **Start State**. **s0** is the **Goal State** and **s1** is an **End State**. Both states **s0** and **s1** are also “absorbing states” which means that an agent can move into either of those two states, but then the game ends, meaning that the agent cannot exit from, or move out of, either of these two states.

If an agent is in any one of the 20 states (except for states **s0** and **s1)**, that agent can attempt to take any one of **three actions: move up, move down**, or **move left** (but the agent cannot move right). However, the “physics” of this “Race to Goal” ‘world’ are such that it is impossible to pass through the four outer wall boundaries or perimeters of this world. For example, if the agent is in state **s8** and attempts to move left, the agent simply bounces off the perimeter wall and remains in state **s8**. Or if the agent is in state **s18** and attempts to move up, the agent remains in state **s18**.

Furthermore, the heavy red border walls on portions of the interior of this ‘world’ are also impenetrable. So, for example, if the agent is in state **s14** and attempts to move left, the agent bounces off the heavy red border boundary and simply remains in state **s14.**

You have just been hired by **Alphabet Inc.** the parent company of **Google**, as a **Reinforcement Learning Data Scientist** for **$205,000** annual salary. On your first day of work your supervisor hands you this assignment with the following questions to answer:

1) Using the **ReinforcementLearning** package in R, and modifying the attached RR.R file that we demonstrated in video lectures 17 & 18 of the Udemy course **Reinforcement Learning with R**, write R code to create a new Race-to-Goal environment which models the complete “physics” of our modified environment, including **all states, actions, and rewards**. You will also need to write R code to: **define the state and action sets**; **to run the sampleExperience() function** to generate simulated data; and **to run the ReinforcementLearning() function** to output your modified **RACE.model.** Also see the attached Race\_to\_Goal\_template.R file to help you get started.

**Several tips:** Use my attached **Race\_to\_Goal\_template.R** file to get you started. **MAKE SURE to use the same set.seed(1234) command that you see on line 44 on that file each time just before you run the sampleEnvironment() function so that you get the same outputted data!** Use the same reinforcement learning parameter values in the control structure that you see “hard-wired” into lines 56-60 of the .R file.

As a check, note that my solution file used exactly **38 separate lines of code** to define all possible state-action-new-state triples inside the modified **RACE.env()** function. You do not need to have more lines, and you cannot have fewer lines and adequately define all of the possible state-action-new-state triples, in my opinion.

Created a R package with Racetogoal environment for Reinforcement learning. Model contains all states , actions and rewards.

2) Run all of the code **using exactly N=1000 resamples** in the **sampleExperience()** function. Generate: (1) **the optimal policy** (the “best” complete set of state-action pairs given that you are in any particular state on the grid and must take an action to leave that state (exclude absorbing states **s0** and **s1** from the optimal policy); and (2) **the optimal “plan”** (sequence of specific states to move through beginning in state **s3 Start State** to reach state **s0 Goal State**. **MAKE SURE THAT YOU RUN set.seed(1234) each time just before you run the sampleExperience() function** to collect your data.

**What is the optimal policy?** Your answer would include every state-action pair in the policy except for states **s0** and **s1**. Note that in my solution there are no impossible state-action pairs such as moving right when the agent is in state **s11**. There are also no counter-intuitive state-action pairs such as moving up from state **s8**.

**What is the optimal plan?** This would include the entire sequence of states traversed in moving from Start State **s3** until you are in **s0** Goal State.

Optimal Policy

s14 s15 s16 s17 s18 s19 s2 s3 s4 s5 s6

"up" "up" "down" "down" "left" "left" "up" "up" "down" "left" "up"

s7 s8 s9 s10 s11 s12 s13

"up" "down" "down" "up" "up" "down" "down"

State-Action function Q

up down left

s14 0.0682494503 1.057079e-06 8.665862e-04

s15 0.0275544565 1.563913e-03 3.443078e-04

s16 0.0404247606 1.160892e+00 1.966809e-01

s0 0.0000000000 0.000000e+00 0.000000e+00

s17 0.0068343087 5.252089e-01 2.418180e-02

s1 0.0000000000 0.000000e+00 0.000000e+00

s18 0.0060014597 4.119263e-03 2.376408e-01

s19 0.0089826881 1.529615e-04 9.617140e-02

s2 0.0021447340 5.590067e-05 -9.282102e+00

s3 0.0007152097 1.097300e-05 3.696321e-05

s4 1.8465364044 9.999993e+00 4.632084e+00

s5 0.5081611380 -8.146980e+00 4.992259e+00

s6 0.0059122259 5.535809e-05 8.341709e-05

s7 0.0022426518 6.912323e-07 5.337973e-08

s8 0.2545117947 4.975574e+00 8.578049e-01

s9 0.1270015339 2.480143e+00 2.745857e-01

s10 0.0214415408 2.690600e-07 7.678326e-04

s11 0.0096932127 3.548047e-07 3.554393e-04

s12 0.0413341924 2.460239e+00 4.000805e-01

s13 0.0063718333 1.205398e+00 2.543097e-02

3) Run the code again this time **using exactly N=5000 resamples** in the **sampleExperience()** function. Make sure that you run **set.seed(1234)** just before you run the **sampleExperience()** function.

Do any of the **state-action pairs** from the original optimal policy change compared to **N=1000**? Look at each one carefully. If so, which one(s)?

Look at the values in the outputted **state-action-function-Q-matrix**. Are the numbers’ magnitudes changed from when **N=1000**? If so, why do you think this is so?

Optimal Policy

s14 s15 s16 s17 s18 s19 s2 s3 s4 s5 s6

"up" "up" "down" "down" "left" "left" "up" "up" "down" "left" "up"

s7 s8 s9 s10 s11 s12 s13

"up" "down" "down" "up" "up" "down" "down"

Optimum policy state-action pairs have not changed when compared to 1000 Since the environment is same

State-Action function Q

up down left

s14 0.156249994 0.019145968 0.048797608

s15 0.078124980 0.010631709 0.034349694

s16 0.398379812 1.250000000 0.343960744

s0 0.000000000 0.000000000 0.000000000

s17 0.111720641 0.624999999 0.239102824

s1 0.000000000 0.000000000 0.000000000

s18 0.119455073 0.032462258 0.312499998

s19 0.053399988 0.015696771 0.156249997

s2 0.019531149 0.004086730 -9.528987130

s3 0.009765537 0.002685916 0.004475085

s4 2.350186328 10.000000000 4.924249316

s5 0.991174681 -9.476652367 5.000000000

s6 0.039062410 0.006760002 0.010757749

s7 0.019531173 0.001670740 0.009749146

s8 0.902893184 5.000000000 1.530300042

s9 0.451131556 2.500000000 0.783437181

s10 0.078124985 0.012351002 0.027429839

s11 0.039062423 0.003899996 0.021475483

s12 0.396827050 2.500000000 0.920567021

s13 0.131467651 1.250000000 0.392452327

The values in state-action-function-q matrix have changed , since the probabilities change with respect to change in quantity of data

4) Run the code again this **using exactly N=10000 resamples** in the **sampleExperience()** function. Make sure that you run **set.seed(1234)** just before you run the **sampleExperience()** function.

Do any of the **state-action pairs** from the original optimal policy change compared to **N=1000** and also compared to the **optimal policy when N=5000**? Look at each one carefully. If so, which one(s)?

Look at the values in the outputted **state-action-function-Q-matrix**. Are the numbers’ magnitudes changed from when **N=1000** and to **N=5000**? If so, why do you think this is so?

Optimum Policy

s14 s15 s16 s17 s18 s19 s2 s3 s4 s5 s6

"up" "up" "down" "down" "left" "left" "up" "up" "down" "left" "up"

s7 s8 s9 s10 s11 s12 s13

"up" "down" "down" "up" "up" "down" "down"

Optimum policy state-action pairs have not changed when compared to 1000 Since the environment is same

State-Action function Q

up down left

s14 0.156250000 0.035371926 0.069842078

s15 0.078125000 0.017269104 0.069112053

s16 0.573156570 1.250000000 0.560707233

s0 0.000000000 0.000000000 0.000000000

s17 0.255793979 0.625000000 0.302585130

s1 0.000000000 0.000000000 0.000000000

s18 0.148674251 0.060434318 0.312500000

s19 0.074001565 0.036221773 0.156250000

s2 0.019531250 0.009075204 -9.921448328

s3 0.009765625 0.004193403 0.008785277

s4 2.491288374 10.000000000 4.996035606

s5 1.202039146 -9.817519964 5.000000000

s6 0.039062500 0.009078035 0.018068077

s7 0.019531250 0.004448909 0.018460901

s8 1.161770105 5.000000000 2.404506770

s9 0.580804974 2.500000000 1.103587644

s10 0.078125000 0.017706126 0.035777093

s11 0.039062500 0.008973265 0.035039155

s12 0.586947182 2.500000000 1.209948640

s13 0.242364359 1.250000000 0.586217602