- **Action Value Methods:** We begin by looking more closely at methods for estimating the values of actions and for using the estimates to make action selection decisions, which we collectively call action-value methods. Recall that the true value of an action is the mean reward when that action is selected. One natural way to estimate this is by averaging the rewards actually received:

  $$Q_t(a) = \frac{\text{sum of rewards when a taken prior to t}}{\text{number of times a taken prior to t}} = \frac{\sum i = 1^{t-1} R_i * 1_{A_i=a}}{\sum i = 1^{t-1} 1_{A_i=a}}$$

  Where $1_{\text{predicate}}$ denotes a random variable that is 1 if $predicate$ is true and 0 otherwise. If the denominator is zero then, instead we define $Q_t(a)$ as some default value such as 0. As the denominator goes to infinity, by the law of large numbers, $Q_t(a)$ converges to $q_*(a)$. We call this the $sample - average$ method for estimating action values because each estimate is an average of the sample of relevant rewards.

- The simplest action selection rule is to select one of the actions with the highest estimated value, that is, one of the greedy actions as defined in the previous section. If there is more than one greedy action, then a selection is made among them in some arbitrary way, perhaps randomly. We write this greedy action selection method as $A_t \doteq \text{argmax}_a Q_t(a)$

  Where argmax_a denotes the action a for which the expression that follows is maximized (again, with ties broken arbitrarily). Greedy action selection always exploits current knowledge to maximize immediate reward; it spends no time at all sampling apparently inferior actions to see if they might really be better. A simple alternative is to behave greedily most of the time, but every once in a while, say with small probability ϵ, instead select randomly from among all the actions with equal probability, independently of the action-value estimates. We call methods using this near-greedy action selection rule $\epsilon\text{-greedy}$ methods. An advantage of these methods is that, in the limit as the number of steps increases, every action will be sampled an infinite number of times, thus ensuring that all the $Q_t(a)$ converge to $q_*(a)$. This of course implies that the probability of selecting the optimal action converges to greater than $1 - \epsilon$, that is, to near certainty. These are just asymptotic guarantees, however, and say little about the practical effectiveness of
  the methods.

- **Incremental Implementation:** The action-value methods we have discussed so far all estimate action values as sample averages of observed rewards. We now turn to the question of how these averages can be computed in a computationally efficient manner, in particular, with constant memory and constant per-time-step computation.
- To simplify notation we concentrate on a single action. Let R_i now denote the reward received after the ith selection of this $action$, and let Q_n denote the estimate of its action value after it has been selected $n-1$ times, which we can now write simply as

  $$Q_n = \frac{R_1 + R_2 + \ + R_{n-1}}{n-1}$$

  The obvious implementation would be to maintain a record of all the rewards and then perform this computation whenever the estimated value was needed. However, if this is done, then the memory and computational requirements would grow over time as more rewards are seen. Each additional reward would require additional memory to store it and additional computation to compute the sum in the numerator.
- As you might suspect, this is not really necessary. It is easy to devise incremental formulas for updating averages with small, constant computation required to process each new reward. Given Q_n and the nth reward, R_n, the new average of all n rewards can be computed by

  $$\begin{aligned}
  &Q_{n+1} \\
  &= \frac{1}{n}\sum_{i=1}^{n} R_i \\
  &= \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) \\
  &= \frac{1}{n}\left(R_n + (n-1)\,\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right) \\
  &= \frac{1}{n}\left(R_n + (n-1)\,Q_n\right) \\
  &= \frac{1}{n}\left(R_n + nQ_n - Q_n\right) \\
  &= Q_n + \frac{1}{n}\left[R_n - Q_n\right],
  \end{aligned}$$

  which holds even for $n = 1$, obtaining $Q_2 = R_1$ for arbitrary Q_1. This implementation requires memory only for Qn and n, and only the small computation (2.3) for each new Reward. This update rule (2.3) is of a form that occurs frequently throughout this book. The general form is

  $$\text{StepSize}\,[\text{Target - OldEstimate}] + \text{OldEstimate} \rightarrow \text{NewEstimate}$$

  The expression $[Target - OldEstimate]$ is an $error$ in the estimate. It is reduced by taking a step toward the “Target.” The target is presumed to indicate a desirable direction

in which to move, though it may be noisy. In the case above, for example, the target is the nth reward.

Note that the step-size parameter $(StepSize)$ used in the incremental method (2.3) changes from time step to time step. In processing the nth reward for action a, the method uses the step-size parameter $\frac{1}{n}$. In this book we denote the step-size parameter by α or, more generally, by $\alpha_t(a)$