DISCLAIMER:

ReadMe website is intended for academic and demonstration purposes only. We’re only showing a preview of the book to respect the author’s copyright. Thank you for your understanding!

- Group 4: The Classified

Algorithms to Live By

The Computer Science of Human Decisions

Brian Christian and
Tom Griffiths

Here we'll analyze one of the simplest cases: where we know for certain the price range in which offers will come, and where all offers within that range are equally likely. If we don't have to worry about the offers (or our savings) running out, then we can think purely in terms of what we can expect to gain or lose by waiting for a better deal. If we decline the current offer, will the chance of a better one, multiplied by how *much* better we expect it to be, more than compensate for the cost of the wait? As it turns out, the math here is quite clean, giving us an explicit function for stopping price as a function of the cost of waiting for an offer.

This particular mathematical result doesn't care whether you're selling a mansion worth millions or a ramshackle shed. The only thing it cares about is the difference between the highest and lowest offers you're likely to receive. By plugging in some concrete figures, we can see how this algorithm offers us a considerable amount of explicit guidance. For instance, let's say the range of offers we're expecting runs from $400,000 to $500,000. First, if the cost of waiting is trivial, we're able to be almost infinitely choosy. If the cost of getting another offer is only a dollar, we'll maximize our earnings by waiting for someone willing to offer us $499,552.79 and not a dime less. If waiting costs $2,000 an offer, we should hold out for an even $480,000. In a slow market where waiting costs $10,000 an offer, we should take anything over $455,279. Finally, if waiting costs half or more of our expected range of offers—in this case, $50,000—then there's no advantage whatsoever to holding out; we'll do best by taking the very first offer that comes along and calling it done. Beggars can't be choosers.

know pays out seven times out of ten! As you go down the diagonal, notice that a record of 1–1 yields an index of 0.6346, a record of 2–2 yields 0.6010, and so on. If such 50%-successful performance persists, the index does ultimately converge on 0.5000, as experience confirms that the machine is indeed nothing special and takes away the "bonus" that spurs further exploration. But the convergence happens fairly slowly; the exploration bonus is a powerful force. Indeed, note that even a failure on the very first pull, producing a record of 0–1, makes for a Gittins index that's still above 50%.

We can also see how the explore/exploit tradeoff changes as we change the way we're discounting the future. The following table presents exactly the same information as the preceding one, but assumes that a payoff next time is worth 99% of one now, rather than 90%. With the future weighted nearly as heavily as the present, the value of making a chance discovery, relative to taking a sure thing, goes up even more. Here, a totally untested machine with a 0–0 record is worth a guaranteed 86.99% chance of a payout!

Losses \ Wins	0	1	2	3	4	5	6	7	8	9
0	.8699	.9102	.9285	.9395	.9470	.9525	.9568	.9603	.9631	.9655
1	.7005	.7844	.8268	.8533	.8719	.8857	.8964	.9051	.9122	.9183
2	.5671	.6726	.7308	.7696	.7973	.8184	.8350	.8485	.8598	.8693
3	.4701	.5806	.6490	.6952	.7295	.7561	.7773	.7949	.8097	.8222
4	.3969	.5093	.5798	.6311	.6697	.6998	.7249	.7456	.7631	.7781
5	.3415	.4509	.5225	.5756	.6172	.6504	.6776	.7004	.7203	.7373
6	.2979	.4029	.4747	.5277	.5710	.6061	.6352	.6599	.6811	.6997
7	.2632	.3633	.4337	.4876	.5300	.5665	.5970	.6230	.6456	.6653
8	.2350	.3303	.3986	.4520	.4952	.5308	.5625	.5895	.6130	.6337
9	.2117	.3020	.3679	.4208	.4640	.5002	.5310	.5589	.5831	.6045

Gittins index values as a function of wins and losses, assuming that a payoff next time is worth 99% of a payoff now.

The Gittins index, then, provides a formal, rigorous justification for preferring the unknown, provided we have some opportunity to exploit the

score him on his average speed across all attempts. (You can see why they don't let computer scientists run these things.)

Moreover, a computer scientist would want to know the *worst* sort time. Worst-case analysis lets us make hard guarantees: that a critical process will finish in time, that deadlines won't be blown. So for the rest of this chapter—and the rest of this book, actually—we will be discussing only algorithms' worst-case performance, unless noted otherwise.

Computer science has developed a shorthand specifically for measuring algorithmic worst-case scenarios: it's called "Big-O" notation. Big-O notation has a particular quirk, which is that it's inexact by design. That is, rather than expressing an algorithm's performance in minutes and seconds, Big-O notation provides a way to talk about the kind of *relationship* that holds between the size of the problem and the program's running time. Because Big-O notation deliberately sheds fine details, what emerges is a schema for dividing problems into different broad classes.

Imagine you're hosting a dinner party with n guests. The time required to clean the house for their arrival doesn't depend on the number of guests at all. This is the rosiest class of problems there is: called "Big-O of one," written $O(1)$, it is also known as "constant time." Notably, Big-O notation doesn't care a whit how long the cleaning actually takes—just that it's always the same, totally invariant of the guest list. You've got the same work to do if you have one guest as if you have ten, a hundred, or any other n.

Now, the time required to pass the roast around the table will be "Big-O of n," written $O(n)$, also known as "linear time"—with twice the guests, you'll wait twice as long for the dish to come around. And again, Big-O notation couldn't care less about the number of courses that get served, or whether they go around for second helpings. In each case, the time still depends linearly on the guest list size—if you drew a graph of the number of guests vs. the time taken, it would be a straight line. What's more, the existence of *any* linear-time factors will, in Big-O notation, swamp *all* constant-time factors. That is to say, passing the roast once around the table, or remodeling your dining room for three months and *then* passing the roast once around the table, are both, to a computer scientist, effectively equivalent. If that seems crazy, remember that computers deal with values

*It’s interesting to note that NCAA’s March Madness tournament is consciously designed to mitigate this flaw in its algorithm. The biggest problem in Single Elimination, as we’ve said, would seem to be a scenario where the first team that gets eliminated by the winning team is actually the second-best team overall, yet lands in the (unsorted) bottom half. The NCAA works around this by seeding the teams, so that top-ranked teams cannot meet each other in the early rounds. The seeding process appears to be reliable at least in the most extreme case, as a sixteenth-seeded team has never defeated a first seed in the history of March Madness.

5 Scheduling

First Things First

> *How we spend our days is, of course, how we spend our lives.*
>
> —ANNIE DILLARD

> *“Why don’t we write a book on scheduling theory?” I asked.… “It shouldn’t take much time!” Book-writing, like war-making, often entails grave miscalculations. Fifteen years later,* Scheduling *is still unfinished.*
>
> —EUGENE LAWLER

It’s Monday morning, and you have an as-yet blank schedule and a long list of tasks to complete. Some can be started only after others are finished (you can’t load the dishwasher unless it’s unloaded first), and some can be started only after a certain time (the neighbors will complain if you put the trash out on the curb before Tuesday night). Some have sharp deadlines, others can be done whenever, and many are fuzzily in between. Some are urgent, but not important. Some are important, but not urgent. “We are what we repeatedly do,” you seem to recall Aristotle saying—whether it’s mop the floor, spend more time with family, file taxes on time, learn French.

So what to do, and when, and in what order? Your life is waiting.

Though we always manage to find *some* way to order the things we do in our days, as a rule we don’t consider ourselves particularly good at it—hence the perennial bestseller status of time-management guides. Unfortunately, the guidance we find in them is frequently divergent and inconsistent. *Getting Things Done* advocates a policy of immediately doing any task of two minutes or less as soon as it comes to mind. Rival bestseller *Eat That Frog!* advises beginning with the most difficult task and moving

*Ironically enough, Pathfinder software team leader Glenn Reeves would blame the bug on “deadline pressures,” and on the fact that fixing this particular issue during development had been deemed a “lower priority.” So the root cause, in a sense, mirrored the problem itself.

Failing the marshmallow test—and being less successful in later life—may not be about lacking willpower. It could be a result of believing that adults are not dependable: that they can't be trusted to keep their word, that they disappear for intervals of arbitrary length. Learning self-control is important, but it's equally important to grow up in an environment where adults are consistently present and trustworthy.

Priors in the Age of Mechanical Reproduction

> *As if someone were to buy several copies of the morning paper to assure himself that what it said was true.*
>
> —LUDWIG WITTGENSTEIN

> *He is careful of what he reads, for that is what he will write. He is careful of what he learns, for that is what he will know.*
>
> —ANNIE DILLARD

The best way to make good predictions, as Bayes's Rule shows us, is to be accurately informed about the things you're predicting. That's why we can do a good job of projecting human life spans, but perform poorly when asked to estimate the reigns of pharaohs.

Being a good Bayesian means representing the world in the correct proportions—having good priors, appropriately calibrated. By and large, for humans and other animals this happens naturally; as a rule, when something surprises us, it *ought* to surprise us, and when it doesn't, it ought not to. Even when we accumulate biases that aren't objectively correct, they still usually do a reasonable job of reflecting the specific part of the world we live in. For instance, someone living in a desert climate might overestimate the amount of sand in the world, and someone living at the poles might overestimate the amount of snow. Both are well tuned to their own ecological niche.

Everything starts to break down, however, when a species gains language. What we talk about isn't what we experience—we speak chiefly of interesting things, and those tend to be things that are uncommon. More or less by definition, events are always *experienced* at their proper frequencies, but this isn't at all true of language. Anyone who has experienced a snake bite or a lightning strike will tend to retell those

that ultimately swayed him in favor of marriage. His book budget was a distraction.

Before we get too critical of Darwin, however, painting him as an inveterate overthinker, it's worth taking a second look at this page from his diary. Seeing it in facsimile shows something fascinating. Darwin was no Franklin, adding assorted considerations for days. Despite the seriousness with which he approached this life-changing choice, Darwin made up his mind exactly when his notes reached the bottom of the diary sheet. *He was regularizing to the page.* This is reminiscent of both Early Stopping and the Lasso: anything that doesn't make the page doesn't make the decision.

His mind made up to marry, Darwin immediately went on to overthink the timing. "When? Soon or Late," he wrote above another list of pros and cons, considering everything from happiness to expenses to "awkwardness" to his long-standing desire to travel in a hot air balloon and/or to Wales. But by the end of the page he resolved to "Never mind, trust to chance"—and the result, within several months' time, was a proposal to Emma Wedgwood, the start of a fulfilling partnership and a happy family life.

expressions, such as $2x^3 + 13x^2 + 22x + 8$ and $(2x + 1) \times (x + 2) \times (x + 4)$, working out whether those expressions are in fact the same function—by doing all the multiplication, then comparing the results—can be incredibly time-consuming, especially as the number of variables increases.

Here again randomness offers a way forward: just generate some random *x*s and plug them in. If the two expressions are *not* the same, it would be a big coincidence if they gave the same answer for some randomly generated input. And an even bigger coincidence if they also gave identical answers for a second random input. And a bigger coincidence still if they did it for three random inputs in a row. Since there is no known deterministic algorithm for efficiently testing polynomial identity, this randomized method—with multiple observations quickly giving rise to near-certainty—is the only practical one we have.

In Praise of Sampling

The polynomial identity test shows that sometimes our effort is better spent checking random values—sampling from the two expressions we want to know about—than trying to untangle their inner workings. To some extent this seems reasonably intuitive. Given a pair of nondescript gadgets and asked whether they are two different devices or two copies of the same one, most of us would start pushing random buttons rather than crack open the cases to examine the wiring. And we aren't particularly surprised when, say, a television drug lord knifes open a few bundles at random to be reasonably certain about the quality of the entire shipment.

There are cases, though, where we don't turn to randomness—and maybe we should.

Arguably the most important political philosopher of the twentieth century was Harvard's John Rawls, who set for himself the ambitious task of reconciling two seemingly opposite key ideas in his field: *liberty* and *equality*. Is a society more "just" when it's more free, or more equal? And do the two really have to be mutually exclusive? Rawls offered a way of approaching this set of questions that he called the "veil of ignorance." Imagine, he said, that you were about to be born, but didn't know as whom: male or female, rich or poor, urban or rural, sick or healthy. And before learning your status, you had to choose what kind of society you'd live in.

delivered by hand across the very terrain that contains the enemy, meaning there's a chance that any given message will never arrive.

The first general, say, suggests a time for the attack, but won't dare go for it unless he knows for sure that his comrade is moving, too. The second general receives the orders and sends back a confirmation—but won't dare attack unless he knows that the first general received that confirmation (since otherwise the first general won't be going). The first general receives the confirmation—but won't attack until he's certain that the second general *knows* he did. Following this chain of logic requires an infinite series of messages, and obviously that won't do. Communication is one of those delightful things that work only in practice; in theory it's impossible.

In most scenarios the consequences of communication lapses are rarely so dire, and the need for certainty rarely so absolute. In TCP, a failure generally leads to retransmission rather than death, so it's considered enough for a session to begin with what's called a "triple handshake." The visitor says hello, the server acknowledges the hello and says hello back, the visitor acknowledges that, and if the server receives this third message, then no further confirmation is needed and they're off to the races. Even after this initial connection is made, however, there's still a risk that some later packets may be damaged or lost in transit, or arrive out of order. In the postal mail, package delivery can be confirmed via return receipts; online, packet delivery is confirmed by what are called acknowledgment packets, or ACKs. These are critical to the functioning of the network.

The way that ACKs work is both simple and clever. Behind the scenes of the triple handshake, each machine provides the other with a kind of serial number—and it's understood that every packet sent after that will increment those serial numbers by one each time, like checks in a checkbook. For instance, if your computer initiates contact with a web server, it might send that server, say, the number 100. The ACK sent by the server will in turn specify the serial number at which the server's own packets will begin (call it 5,000), and also will say "Ready for 101." Your machine's ACK will carry the number 101 and will convey in turn "Ready for 5,001." (Note that these two numbering schemes are totally independent, and the number that begins each sequence is typically chosen at random.)

going directly to the best strategy, assuming rational play. In rock-paper-scissors, scrutinizing your opponent's face for signs of what they might throw next may not be worthwhile, if you know that simply throwing at random is an unbeatable strategy in the long run.

More generally, the Nash equilibrium offers a prediction of the stable long-term outcome of any set of rules or incentives. As such, it provides an invaluable tool for both predicting and shaping economic policy, as well as social policy in general. As Nobel laureate economist Roger Myerson puts it, the Nash equilibrium "has had a fundamental and pervasive impact in economics and the social sciences which is comparable to that of the discovery of the DNA double helix in the biological sciences."

Computer science, however, has complicated this story. Put broadly, the object of study in mathematics is *truth*; the object of study in computer science is *complexity*. As we've seen, it's not enough for a problem to have a solution if that problem is intractable.

In a game-theory context, knowing that an equilibrium exists doesn't actually tell us what it is—or how to get there. As UC Berkeley computer scientist Christos Papadimitriou writes, game theory "predicts the agents' equilibrium behavior typically with no regard to the ways in which such a state will be reached—a consideration that would be a computer scientist's foremost concern." Stanford's Tim Roughgarden echoes the sentiment of being unsatisfied with Nash's proof that equilibria always exist. "Okay," he says, "but we're computer scientists, right? Give us something we can use. Don't just tell me that it's there; tell me how to find it." And so, the original field of game theory begat algorithmic game theory—that is, the study of theoretically ideal strategies for games became the study of how machines (and people) *come up* with strategies for games.

As it turns out, asking too many questions about Nash equilibria gets you into computational trouble in a hurry. By the end of the twentieth century, determining whether a game has more than one equilibrium, or an equilibrium that gives a player a certain payoff, or an equilibrium that involves taking a particular action, had all been proved to be intractable problems. Then, from 2005 to 2008, Papadimitriou and his colleagues proved that simply *finding* Nash equilibria is intractable as well.

Conclusion

Computational Kindness

> *I firmly believe that the important things about humans are social in character and that relief by machines from many of our present demanding intellectual functions will finally give the human race time and incentive to learn how to live well together.*
>
> —MERRILL FLOOD

Any dynamic system subject to the constraints of space and time is up against a core set of fundamental and unavoidable problems. These problems are computational in nature, which makes computers not only our tools but also our comrades. From this come three simple pieces of wisdom.

First, there are cases where computer scientists and mathematicians have identified good algorithmic approaches that can simply be transferred over to human problems. The 37% Rule, the Least Recently Used criterion for handling overflowing caches, and the Upper Confidence Bound as a guide to exploration are all examples of this.

Second, knowing that you are using an optimal algorithm should be a relief even if you don't get the results you were looking for. The 37% Rule fails 63% of the time. Maintaining your cache with LRU doesn't guarantee that you will always find what you're looking for; in fact, neither would clairvoyance. Using the Upper Confidence Bound approach to the explore/exploit tradeoff doesn't mean that you will have *no* regrets, just that those regrets will accumulate ever more slowly as you go through life. Even the best strategy sometimes yields bad results—which is why computer scientists take care to distinguish between "process" and "outcome." If you followed the best possible process, then you've done all you can, and you shouldn't blame yourself if things didn't go your way.

online commerce comprises hundreds of billions: Online sales estimated by Forrester Research. See, for instance, "US Online Retail Sales to Reach $370B By 2017; €191B in Europe," *Forbes*, 3/14/2013, http://www.forbes.com/sites/forrester/2013/03/14/us-online-retail-sales-to-reach-370b-by-2017-e191b-in-europe/.

best algorithms to use remain hotly contested: Chris Stucchio, for instance, penned a cutting article titled "Why Multi-armed Bandit Algorithms Are Superior to A/B Testing," which was then countered by an equally cutting article called "Don't Use Bandit Algorithms—They Probably Won't Work for You"—also written by Chris Stucchio. See https://www.chrisstucchio.com/blog/2012/bandit_algorithms_vs_ab.html and https://www.chrisstucchio.com/blog/2015/dont_use_bandits.html. Stucchio's 2012 post was written partly in reference to an article by Paras Chopra titled "Why Multi-armed Bandit Algorithm Is Not 'Better' than A/B Testing" (https://vwo.com/blog/multi-armed-bandit-algorithm/), which was itself written partly in reference to an article by Steve Hanov titled "20 lines of code that will beat A/B testing every time" (http://stevehanov.ca/blog/index.php?id=132).

it appeared in the *Washington Star*: Jean Heller, "Syphilis Patients Died Untreated," *Washington Star*, July 25, 1972.

document known as the Belmont Report: *The Belmont Report: Ethical principles and guidelines for the protection of human subjects of research*, April 18, 1979. Available at http://www.hhs.gov/ohrp/humansubjects/guidance/belmont.html.

proposed conducting "adaptive" trials: See Zelen, "Play the Winner Rule and the Controlled Clinical Trial." While this was a radical idea, Zelen wasn't the first to propose it. That honor goes to William R. Thompson, an instructor in the School of Pathology at Yale, who formulated the problem of identifying whether one treatment is more effective than another, and proposed his own solution, in 1933 (Thompson, "On the Likelihood That One Unknown Probability Exceeds Another").

The solution that Thompson proposed—randomly sampling options, where the probability of choosing an option corresponds to the probability that it is the best based on the evidence observed so far—is the basis for much recent work on this problem in machine learning (we return to the algorithmic uses of randomness and sampling in chapter 9).

Neither Frederick Mosteller nor Herbert Robbins seemed to be aware of Thompson's work when they started to work on the two-armed bandit problem. Richard Bellman found the "little-known papers" a few years later, noting that "We confess that we found these papers in the standard fashion, namely while thumbing through a journal containing another paper of interest" (Bellman, "A Problem in the Sequential Design of Experiments").

predictor model finds the best coefficients for all values of x up to x^9, estimating a polynomial of degree 9.

through each and every point on the chart: In fact, it's a mathematical truth that you can always draw a polynomial of degree $n - 1$ through any n points.

people's baseline level of satisfaction: Lucas et al., "Reexamining Adaptation and the Set Point Model of Happiness."

not always better to use a more complex model: Statisticians refer to the various factors in the model as "predictors." A model that's too simple, such as a straight line attempting to fit a curve, is said to exhibit "bias." The opposite kind of systemic error, where a model is made too complicated and therefore gyrates wildly because of small changes in the data, is known as "variance."

The surprise is that these two kinds of errors—bias and variance—can be *complementary*. Reducing bias (making the model more flexible and complicated) can increase variance. And increasing bias (simplifying the model and fitting the data less tightly) can sometimes reduce variance.

Like the famous Heisenberg uncertainty principle of particle physics, which says that the more you know about a particle's momentum the less you know about its position, the so-called bias-variance tradeoff expresses a deep and fundamental bound on how good a model can be—on what it's possible to know and to predict. This notion is found in various places in the machine-learning literature. See, for instance, Geman, Bienenstock, and Doursat, "Neural Networks and the Bias/Variance Dilemma," and Grenander, "On Empirical Spectral Analysis of Stochastic Processes."

in the Book of Kings: The bronze snake, known as Nehushtan, gets destroyed in 2 Kings 18:4.

"pay good money to remove the tattoos": Gilbert, *Stumbling on Happiness*.

duels less than fifty years ago: If you're not too fainthearted, you can watch video of a duel fought in 1967 at http://passerelle-production.u-bourgogne.fr/web/atip_insulte/Video/archive_duel_france.swf.

as athletes overfit their tactics: For an interesting example of very deliberately overfitting fencing, see Harmenberg, *Epee 2.0*.

"Incentive structures work": Brent Schlender, "The Lost Steve Jobs Tapes," *Fast Company*, May 2012, http://www.fastcompany.com/1826869/lost-steve-jobs-tapes.

"whatever the CEO decides to measure": Sam Altman, "Welcome, and Ideas, Products, Teams and Execution Part I," Stanford CS183B, Fall 2014, "How to Start a Startup," http://startupclass.samaltman.com/courses/lec01/.

complexity of making change, see also Kozen and Zaks, “Optimal Bounds for the Change-Making Problem.”

Consider a large parking lot: Cassady and Kobza, “A Probabilistic Approach to Evaluate Strategies for Selecting a Parking Space,” compares the “Pick a Row, Closest Space (PRCS)” and “Cycling (CYC)” parking space–hunting algorithms. The more complicated CYC includes an optimal stopping rule, while PRCS starts at the destination, pointing away, and simply takes the first space. The more aggressive CYC found better spaces on average, but the simpler PRCS actually won in terms of total time spent. Drivers following the CYC algorithm spent more time finding better spaces than those better spaces saved them in walk time. The authors note that research of this nature might be useful in the design of parking lots. Computational models of parking are also explored in, e.g., Benenson, Martens, and Birfir, “PARKAGENT: An Agent-Based Model of Parking in the City.”

“spinning” and “blocking”: For a deeper look at when to spin and when to block, see, for instance, Boguslavsky et al., “Optimal Strategies for Spinning and Blocking.” (Note that this is the same Leonid Boguslavsky we encountered briefly in chapter 1 on a water-skiing trip.)

Simon, Herbert A. “A Behavioral Model of Rational Choice.” *Quarterly Journal of Economics* 69, no. 1 (1955): 99–118.

———. *Models of Man*. New York: Wiley, 1957.

———. “On a Class of Skew Distribution Functions.” *Biometrika*, 1955, 425–440.

Siroker, Dan. “How Obama Raised $60 Million by Running a Simple Experiment.” *The Optimizely Blog: A/B Testing You’ll Actually Use* (blog), November 29, 2010, https://blog.optimizely.com/2010/11/29/how-obama-raised-60-million-by-running-a-simple-experiment/.

Siroker, Dan, and Pete Koomen. *A/B Testing: The Most Powerful Way to Turn Clicks into Customers*. New York: Wiley, 2013.

Sleator, Daniel D., and Robert E. Tarjan. “Amortized Efficiency of List Update and Paging Rules.” *Communications of the ACM* 28 (1985): 202–208.

Smith, Adam. *The Theory of Moral Sentiments*. Printed for A. Millar, in the Strand; and A. Kincaid and J. Bell, in Edinburgh, 1759.

Smith, M. H. “A Secretary Problem with Uncertain Employment.” *Journal of Applied Probability* 12, no. 3 (1975): 620–624.

Smith, Wayne E. “Various Optimizers for Single-Stage Production.” *Naval Research Logistics Quarterly* 3, nos. 1–2 (1956): 59–66.

Solovay, Robert, and Volker Strassen. “A Fast Monte-Carlo Test for Primality.” *SIAM Journal on Computing* 6 (1977): 84–85.

Starr, Norman. “How to Win a War if You Must: Optimal Stopping Based on Success Runs.” *Annals of Mathematical Statistics* 43, no. 6 (1972): 1884–1893.

Stephens, David W., and John R. Krebs. *Foraging Theory*. Princeton, NJ: Princeton University Press, 1986.

Stewart, Martha. *Martha Stewart’s Homekeeping Handbook: The Essential Guide to Caring for Everything in Your Home*. New York: Clarkson Potter, 2006.

Steyvers, Mark, Michael D. Lee, and Eric-Jan Wagenmakers. “A Bayesian Analysis of Human Decision-Making on Bandit Problems.” *Journal of Mathematical Psychology* 53 (2009): 168–179.

Stigler, George J. “The Economics of Information.” *Journal of Political Economy* 69 (1961): 213–225.

———. “Information in the Labor Market.” *Journal of Political Economy* 70 (1962): 94–105.