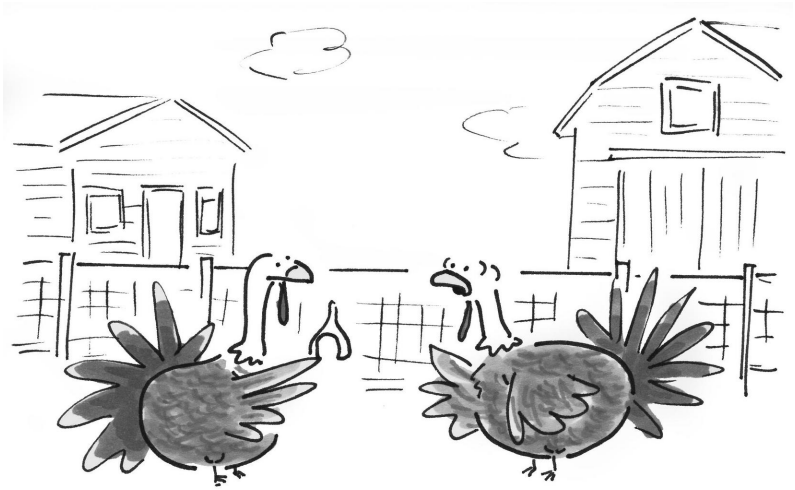
Lost confidence

Confidence games · Techniques



credit [1]

CONFIDENCE GAMES

Many prioritization frameworks include a measure of confidence*—how sure we are that we can execute, at more-or-less the predicted estimated effort, resulting in more-or-less the predicted impact. This seems rational; if two projects generate equal value for equal effort, but we're confident we can execute the first and unsure about the second, we should select the first.

This is not, however, how confidence scores are used. If it were, the process would look like this:

1. Score projects somehow.
2. If there's one clear winner, do it.
3. If there's a tie, pick the one we are more confident in.

That's not a bad idea. But popular frameworks like RICE include "confidence" in step one:

$$\text{Score} = \frac{\text{Reach} \times \text{Impact} \times \text{Confidence}}{\text{Effort}}$$

Or RPS:

$$\text{Score} = \text{Reach} \times \text{Potential} \times \text{Solution Confidence}$$

Which means, for example, the following two scenarios are deemed equally strong:

---

* Or a measure of risk. Whether `risk` is equal to `1-confidence` is left to the discretion of the reader.

1. A small incremental feature, that we're sure we can execute.
2. A large feature, with large impact, that carries some risk.

This equality is false. Especially when you remember that small projects almost always carry higher confidence, and rightly so.* But that systematically skews the prioritization away from delivering as much value as possible—the opposite of what a prioritization framework ought to do.

I don't believe your confidence score anyway. First, because it's ill-defined. What does "30%" mean? What it should mean,[3] is you track your confidence scores and measure how accurate they were after the fact, and determine how good you are at it with mathematical precision.[4] But you don't do that, do you? And if you only ship a few major features per year, you don't have enough data to know.

Second, I don't believe you because we all know that projects are nearly always late, and often have less impact, less quickly, than we wanted. No matter how confident we were. Indeed, *everything* we choose to do, we have at least "pretty good confidence in," or we wouldn't do it at all! So what weight should we place in "confidence?"

> **Hofstadter's Law**
> It always takes longer than you expect, even when you take Hofstadter's Law into account.

To prove this, find any experienced Product Manager[5] and ask: "Can you recall a feature you were certain would be well-received, but wasn't?" Perhaps they had evidence from customer conversations, explicit requests, or purchase commitments—yet after building it, almost no one used it, including those who promised they would. Their eyes will roll as they share multiple stories. This doesn't make them a bad PM. Everyone who has built products, regardless of skill, has these experiences. The best PMs have techniques to mitigate this problem,* but none will claim they can eliminate it entirely.

* If you disagree, consider that the entire motivation of the Agile movement was that we should *always* have low confidence that large projects will be successful, despite our best techniques of planning, analysis, and estimation. And consider this theory of Rocks, Pebbles, and Sand.[2]

Similarly, ask content creators about their most successful work. Often it's something they hastily produced—a trivial piece they almost didn't publish because it seemed uninsightful or trite—yet it generated more views and engagement than anything else that year. Conversely, pieces they spent dozens of hours crafting, work they're genuinely proud of and consider their best, generate minimal interest (Figure 1).

We can summarize the relationship between our confidence and actual results in a handy two-by-two table:

	Was confident	**Was not confident**
They loved it	Lots of things	Lots of things
Nobody cared	Lots of things	Lots of things

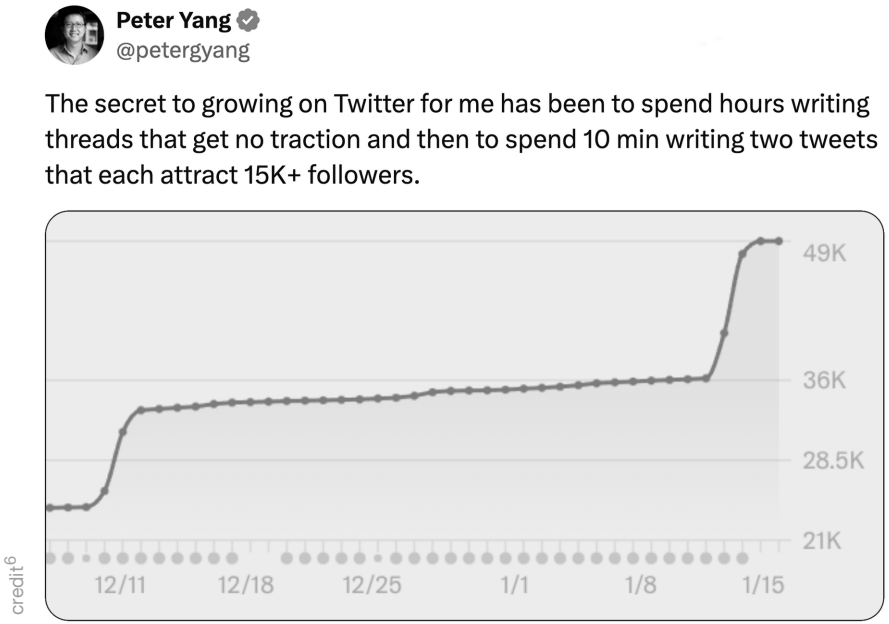
So, if "confidence" is too nebulous to define, and we shouldn't trust ourselves with it anyway, what should we do?

WHAT TO USE IN PLACE OF CONFIDENCE AND RISK

The answer lies in the realm of uncertainty, rather than of probability.

Probability presumes you know the underlying distribution, enabling mathematical predictions about future events. You can predict that flip-

* Some techniques to improve prediction include asking customers to describe exactly how they would use a feature in their normal workflow. Often people genuinely think they would use something, but when forced to walk through it step-by-step, they realize, "Oh wait, this would require me to rewrite this code, we probably wouldn't do that." Or, "I'd need to export it into another system—actually, never mind."



credit[6]

Figure 1

ping a fair coin 100 times is highly likely to result in between 40 and 60 heads, because you know the underlying distribution. If predicting whether a feature will create defensible differentiation were like coin-flipping, you could use probabilities.*

Almost nothing in a startup is like that. Outcomes cannot be assigned meaningful probabilities because things like startup success, strategy, and features are unprecedented, or too complex to model accurately, or we have no precision on the input variables. This is the domain of uncertainty.**

* If you're tempted to claim that Bayesian methods could still work, remember that you need numeric priors and conditional probabilities, both of which we established above are unknowable and ill-defined.

** Formally called "Knightian Uncertainty" after economist Frank Knight in his 1921 work "Risk, Uncertainty, and Profit."

In this domain, we ask: What actions are wise regardless of the probability distribution?

I've previously written about embracing uncertainty in overall product strategy.[7] Below, I'll address a more specific question: How should we prioritize individual items in an uncertain world?

Here are some techniques.

True always

What is always true under any circumstance? This is Bezos's principle of focusing on long-term constants.* For instance, users universally appreciate fast, responsive software. They value web apps that feel native, with background synchronization and instant interactions, that work well on all their devices. At worst, they might not consciously notice; at best (in web-apps like Notion, Miro, Gmail, and Google Docs), performance becomes a key differentiator that customers explicitly value.

Not all features enjoy universal appeal. Rather than attempting precise numerical breakdowns of potential user interest, identify the features where essentially all customers will either value it, or at least enjoy it. Sometimes this certainty exists because is mandatory, even if mundane. Enterprise requirements like SOC 2 compliance aren't exciting, but they're undeniably valuable when selling to the Enterprise. This certainty compensates for the lack of differentiation.

The caveat: your most innovative, differentiated ideas rarely fall into this "absolutely certain" category. While certainties are valuable, they're unlikely to be your strategic differentiators. This tension is natural—great products require both reliable improvements and innovative leaps of faith.

* Bezos frequently said of Amazon's strategy: When you have something that you know is true, even over the long term, you can afford to put a lot of energy into it. His examples include customers wanting lower prices, faster shipping, and fast, fair customer service.

"Absolutely! 100 percent! Guaranteed! Most of the time."

credit[8]

Quick discovery

I've been a long-time advocate of systematically interviewing potential customers[9] to validate ideas before you start building. Still, I have to admit that this falls into the "confidence" trap. You never really know until you build. (You can, however, invalidate before you build, saving you months if not years of wasted time; therefore this is still the right place to begin.)

The typical solution is to build an SLC[10] (my upgrade to an MVP), i.e. a completed but simple product that generates real feedback. Experience, rather than prediction. For existing products, that means maintaining a balanced[11] portfolio between guaranteed wins and innovative bets, applying different validation methods to each.

For example, consider implementing "dummy features"—buttons that, when clicked, reveal: "This feature isn't built yet. Tell us how you'd use it." This simple test provides real signals: a count of interested users

and potential interview candidates who've demonstrated interest through action rather than words. They can provide insights before you build the feature.

This approach generates 100x better signal than surveys asking hypothetical questions. People easily say "yes" to survey questions about future usage, but taking an action—even clicking a button—requires genuine interest. Observed behavior beats stated intentions every time.

Customer impact

Replace confidence with impact. I define impact in two distinct ways:

Majority rule
When the majority of users regularly use a feature, it's undeniably important—likely a key reason people adopt and retain your software.

Passionate advocates
Features that create passionate advocates among a smaller subset of users. These "magnificent delighters" won't appeal universally, but they inspire deep loyalty in specific segments. Like a piece of music that's someone's all-time favorite (while others merely acknowledge it's objectively good).

These are what determine purchase decisions. Your product rarely satisfies every customer need perfectly, but when users absolutely love certain aspects, they'll tolerate shortcomings elsewhere. We see this with beautifully designed[12] software—users accept missing functionality or limited platform support because the design experience itself is so compelling. There are many other reasons[13] for a customer to love you despite your failings.[14]

These "killer delighters" don't require universal appeal. If 15% of customers identify a feature as a primary reason for purchasing or remaining with your product, that's significant. When 15% feel that strongly, many more likely appreciate it, even if less intensely.

I quantify impact with this definition: A high-impact feature either (1) is regularly used by at least 51% of customers, or (2) is cited by at least

15% of customers as among their top three reasons for choosing or retaining your product.

This sets a high bar, but innovative, risky features demand a high bar. If you're undertaking projects that might exceed timelines or have uncertain outcomes, the potential reward must justify that risk.

Invest in leverage

There are some aspects of the business or product where small, incremental changes yield large results. It sounds too good to be true, but there are mathematical or structural areas where it is almost always true.

These include:

- Top-of-funnel quality
- Retention
- Pricing & pricing terms
- Onboarding
- Strategy

Each of these (and more) are justified in detail in this companion article.[15]

Not included in the list above, is creating delightful, differentiated features. Those are special outliers, and therefore won't be produced by common rules of thumb. Still, it's almost always wise to invest a portion of your time on one of these asymmetric bets.

Maximize optionality

If we don't know how the future will unfold, we can make choices that maximize the options we have when we get there. More than flexibility, more than avoiding lock-in, building systems that are almost always ready to handle anything that arises.

Some examples:

- Keeping costs low enables all kinds of pricing and packaging while still being profitable, allowing for testing today and resilience in future.
- Selecting well-established, actively-development cross-platform libraries and frameworks for building user interfaces, so you're able to handle any evolution in platforms and devices.
- Plug-in systems, so that both you and your community can build things that you cannot imagine today.
- API-first architecture so that you own front-end tools, and your own back-end systems, and customer integrations, survives evolution.
- Wrappers around vendor services, so that you can swap out vendors if one becomes unstable, or too expensive, or lags behind others.

Some kinds of optionality require additional work today. For example, vendor-wrappers don't add any value today. Those techniques are wise for mature companies where stability and predictability are more important than releasing a feature a month earlier, but might be the wrong choice for early-stage companies who must rely on their velocity to win against incumbents.[16]

Portfolio of bets

Portfolios reduce variability at the expense of reducing maximum upside. That is, you're unlikely to have zero wins (so your downside isn't too bad), but wins have to make up for the losses, so even the occasional massive win isn't as massive as it would have been. The old joke is that the best investment portfolio would have been to buy Amazon at IPO and hold forever. Sure, but if you applied that advice to some other IPOs that year, you'd have $0. A portfolio of stocks means you'll never go to zero, but your maximum growth will be far less than best stock in the portfolio.

Mathematical sidebar

Why do portfolios work regardless of the underlying probability distributions? The Central Limit Theorem[17] makes this precise: When you draw re-

peated samples from any distribution, then plot each sample's mean, the distribution of those sample means is Gaussian—a normal distribution—with a mean equal to the distribution's mean and a variance $\frac{1}{n}$ of the distribution's variance. So, total portfolio results are normally-distributed regardless of the underlying probability distribution, and we expect results near that mean, i.e. not zero, but also not near the maximum value.

Even further, the The Lindeberg—Lévy Central Limit Theorem[18] shows that the same is true even when each sample is drawn from a *different* underlying probability distribution. This holds only under certain conditions (independence, finite variance, and no single variable dominates all others). Arguably these conditions fail with distributions common in startup environments, e.g. some Power Laws have infinite variance.

Portfolios work when you want solid, predictable, but they don't work when you want outlier results. An example of the latter are venture capitalist or angels investor portfolios, where 65% lose money,[19] and only 10%[20] generate returns high enough to justify the risk and illiquidity. When hunting outliers, you need all-in investments,[21] not portfolios.*

Therefore, if the goal of your prioritization exercise is to find features that will be strong differentiators in the market and strong growth drivers, a portfolio is the wrong tool. On the other hand, if you're prioritizing a bunch of smaller things, where you want incremental but reliable results, a portfolio will get you those results. No need to argue about confidence.

Stop pretending you can quantify confidence, or even define it.

Instead, use techniques that work when the future is unpredictable.

Because it is.

* Mathematically, the reason for this breakage is that the underlying distribution of startup returns is a Power Law that violates the Lindeberg criteria mentioned above.

Current version of this article:

https://longform.asmartbear.com/confidence/

More articles & socials:

https://asmartbear.com

REFERENCES

1. https://andertoons.com/thanksgiving/cartoon/704/no-i-dont-want-to-make-wish-where-did-you-get-that
2. https://longform.asmartbear.com/rocks-pebbles-sand/
3. https://longform.asmartbear.com/probability-words/
4. https://longform.asmartbear.com/forecast/
5. https://longform.asmartbear.com/great-product-manager/
6. https://x.com/petergyang/status/1482756170825666562?s=20
7. https://longform.asmartbear.com/predict-the-future/
8. https://andertoons.com/sales/cartoon/8580/absolutely-100-percent-guaranteed-most-of-the-time
9. https://longform.asmartbear.com/customer-development/
10. https://longform.asmartbear.com/slc/
11. https://longform.asmartbear.com/conflicting-choices/
12. https://longform.asmartbear.com/design/
13. https://longform.asmartbear.com/willingness-to-pay/
14. https://longform.asmartbear.com/customer-love/
15. https://longform.asmartbear.com/leverage-points/
16. https://longform.asmartbear.com/startup-beats-incumbent/
17. https://en.wikipedia.org/wiki/Central_limit_theorem
18. https://en.wikipedia.org/wiki/Lindeberg%27s_condition
19. https://www.sethlevine.com/archives/2020/10/vc-fund-returns-are-more-skewed-than-you-think.html
20. https://www.saastr.com/why-only-12-of-vcs-can-be-considered-successful-max/
21. https://longform.asmartbear.com/investment/