The Flow Forecasting Pocket Guide

Daniel Vacanti and Colleen Johnson

The Flow Forecasting Pocket Guide

Daniel Vacanti and Colleen Johnson

This book is available at https://leanpub.com/ffpg

This version was published on 2025-07-10

Contents

Preface - A Philosophy of Forecasting **1**
Why This Book Now? 2
Who Is This Book For? 3

Chapter 1 - Introduction **4**
Conclusion 6

Chapter 2 - The Flaw of Averages **7**
What Is An Average? 7
The Flaw of Averages Explained 8
Agile and FoA 10
Conclusion 12

Chapter 3 - Uncertainty and Risk **13**
Conclusion 15

Chapter 4 - Monte Carlo Simulation **16**
Conclusion 27

Chapter 5 - Interpreting Monte Carlo Simulation **28**
The Language of Forecasting 33
Conclusion 33

Chapter 6 - More Monte Carlo Simulation **35**
MCS For A Given Number of Items 35
Forecasts for Multiple Items 37
Continuous (Continual) Forecasting 38
Conclusion 40

Chapter 7 - MCS FAQs **41**

Conclusion . . . 48

Chapter 8 - What About Forecasts for Individual Items? . . . 49
Aging . . . 52
Conclusion . . . 53

Chapter 9 - System Stability . . . 54
Stability Defined . . . 54
Stability and MCS . . . 56
Stability and Single Item Forecasts . . . 57
Conclusion . . . 57

Chapter 10 - Putting It All Together . . . 59
What This Means For You . . . 62
Where To Learn More . . . 63

Appendices . . . 65

Appendix A - What Data To Collect . . . 66
What Data To Collect . . . 66
Flow Metrics Calculation . . . 67
Conclusion . . . 70

Appendix B - The Mechanics of Monte Carlo Simulation . . . 72

Appendix C - Histograms . . . 75
What is a Histogram? . . . 75
Constructing a Histogram . . . 76
Conclusion . . . 77

Endnotes . . . 78

Bibliography . . . 80

Preface - A Philosophy of Forecasting

It is impossible to perfectly predict the future. Anyone who tells you otherwise is lying.

That may seem like a strange opening statement from a book on forecasting.
If true, why would you bother with forecasting at all? If you will pardon the awkward transition, the answer to that question is what we are calling our philosophy of forecasting:

Verbose version: "We believe it is possible to get an economically justifiable, good-enough approximation of future uncertainty to make a context-specific risk assessment so that appropriate action can be taken."

Simpler (equivalent) version: "We believe a simple, cost-effective way to understand future uncertainty can be enough to evaluate risk in your context to make smart decisions."

Either way, there is a lot going on in those statements, so allow us to explain them better by breaking our thoughts up into smaller chunks.

Economically Justifiable

What we want is as accurate of a forecast as we can get by spending as little economic resources (time, labour, money) as possible. Investing a lot of resources does not necessarily make forecasts better (e.g., SAFe's PI planning), but at the same time spending no resources would yield no information. The right answer, then, is somewhere in between. If you are practicing Scrum, and you could come up with a better Sprint plan in minutes (or even seconds), wouldn't you do it (we'll see how to do just that in a later chapter). For us, it is this economically efficient aspect that is the biggest influence on our thinking about the best forecasting approach.

Good-Enough Approximation

The opening sentence says it all: no forecast will ever be perfect. Thankfully perfection is not what we are going for. What we want is a good-enough model that will allow us to make decisions or take action.

Weather forecasting is a perfect example of this. We can make fun of weather forecasts all we want, but which one of us NEVER looks at a weather forecast before heading out for the day—especially if you have plans like going to the park, or to the beach, or to an outdoor concert, etc. In your heart you know that forecast is not perfect, but you look anyway.

Future Uncertainty

As we will see in Chapter 3, the future is full of uncertainty. That uncertainty manifests itself as many possible outcomes. For our day out, will it rain or not? If it rains, will it be a drizzle, a shower, or a monsoon? While all of these futures are possible, we need to understand which ones are probable.

Context-Specific Risk Assessment

As we will also see in Chapter 3, all of those possible futures can be categorized as either good outcomes or bad outcomes, depending on our context. If you are heading out to play baseball in the US Midwest, the absolute last thing you want is rain. If you are a farmer in that same area, then your feelings about rain are probably a little different. But even those contexts can skew. If it hasn't rained for awhile, then maybe the baseball player would like a little rain to soften up the ground a bit. And a farmer fears too much rain just as much as she fears too little. When assessing risk, context is queen.

Appropriate Action

Once we have a understanding of risk in our context, then we can decide what appropriate action to take. Getting back to the weather, by looking at a forecast, we hopefully would have enough information to decide if we take an umbrella, a raincoat, or just simply make other plans.

The purpose of this book is to explore options on how to put into practice all aspects of this forecasting philosophy. Traditional "agile" methods (estimation, story points, etc.) have failed you. Traditional statistical techniques (average, standard deviation, etc.) could lead you astray. What is needed is a more pragmatic, simple approach to produce the answer you need, so that you can get your job done. Anyone who tells you otherwise is lying.

Why This Book Now?

The pace of change has never been faster and the cost of getting it wrong has never been higher.

With the increasing speed of AI, teams are under constant pressure to deliver more, respond to shifting priorities, and make decisions with incomplete information. Yet most organizations still rely on outdated planning techniques that give the illusion of certainty but fail to show risk until it's too late.

With tighter budgets, leaner teams, and higher stakes, leaders and teams can't afford to be surprised by missed deadlines or sunk investments. Flow forecasting offers a simple way to use real data to create simulations that help you spot delivery risk earlier so you can pivot sooner, protect your commitments, and regain trust.

Who Is This Book For?

This book is for anyone responsible for delivering work predictably in uncertain environments, particularly in technology and knowledge work settings. Specifically, it's designed for:

- **Product Managers and Owners** – who need to communicate realistic delivery timelines without relying on guesswork.
- **Delivery Leads, Scrum Masters, and Project Managers** – who are tired of traditional planning and estimating methods that routinely miss the mark.
- **Team Leads and Developers** – who want to push back on unrealistic dates and work within a system that reflects how delivery actually happens.
- **Executives and Stakeholders** – who want better answers to "When will it be done?" and insight into the likelihood of hitting critical deadlines.
- **Agile Coaches and Change Agents** – who need a clear, practical way to introduce flow forecasting that works without overhauling everything.

In short: if you've ever been yelled at for missing a date, this book is for you.

Chapter 1 - Introduction

Here's the scenario:

You are part of a team that is working on a product release that your company is contractually obligated to deliver by March 1. The contract is written in such a way that if you miss the March date, your company will have to pay a significant penalty. The penalty is large enough that a missed date will result in your company losing substantial money on the deal.

Luckily for you, you know several things that make your life somewhat easier:

1. Your release backlog is precisely 1071 "refined" items and that backlog won't change (more on this later).
2. Your team has been working together for quite some time, and you are guaranteed to have your exact team for the duration (i.e., no one will be taken off for other projects).
3. Because you know your team so well, you are also very confident in what they can do. Looking at past data, you know your team regularly finishes about 8 items per day, and you have no reason to expect that performance to significantly change in the future.
4. You are not dependent on any other team to deliver any functionality.
5. Your team will start its work on this release on October 1 which gives you 151 days of runway.
6. Next year is a leap year so you will have 29 days in the upcoming February instead of the usual 28. An extra day of schedule is always helpful.
7. Your entire team is based in the U.S. so you don't have to worry about pesky things like vacations, although you do expect some slowdown over the Thanksgiving and Christmas holidays.

Given this context, the question on everyone's mind is "Will you make the date?" Note that the question is asking you to make a forecast, which is what this book is all about. So let's first consider a more “traditional” approach to answering that question, and see how it plays out.

To come up with your forecast, you start by taking the number of committed items (1071) and dividing that by the number of days in the release (151) to get a required run rate of 7.09 items per day. From above, your team regularly finishes 8 items/day so the dictated timeline seems doable. As there is no meaningful reason to hesitate, you begin work as planned on October 1.

Fast forward to the start of the day on November 1—one month into the release. Your team has finished exactly 293 items. Doing the math, 293 work items completed in 31 days is 9.45 items per day. Not only are you encouraged by the fact that 9.45 items per day is a faster pace than your original estimation, but since you now have 778 work items left in your backlog (1071 − 293 = 778), completing 9.45 work items per day would mean you will be done in 83 days (778 / 9.45 = 83). 83 days from November 1 is January 23. Even if you take into account the expected slowdown in December, you are comfortably ahead of schedule!

We now skip ahead to the start of the day on December 1 and you have exactly 490 work items finished. Doing the math again, 490 work items completed in 61 days is 8.03 items per day. That’s pretty much what you thought your run rate would be and is still way ahead of the required pace. You now have 581 work items left in their backlog which means your new completion date is February 11. This slight slowdown was to be expected since there would have been a dip in productivity at the end of November due to the Thanksgiving holiday. Still, a completion date of February 11 is right where you want to be, given the further expected slowdown this month.

The next significant date for a forecast is January 3, 2012 (New Year’s Day was a Sunday this year which meant the holiday was observed on a Monday). On January 3, the team had 703 items completed which gives a projected completion date of February 22. This is actually good news because you have come through the productivity slowdown of the holiday season and are still on track. In fact, you still have a full week to spare (remember this coming February has 29 days).

Finally, we arrive at February 1. 903 items finished so far. We only have 168 to complete. Even with a slightly shorter month, 168 is a no-brainer target. Our slowest month to date was November when we got

193 items done—and that was largely due to the Thanksgiving holiday. Plus, though not ideal, we could always make people work weekends if needed. There is nothing to worry about!

Your release goes out on March 8.

What went wrong?

While some of you out there might assume this was due to some unexpected event, some unknown complexity, or some unknown dependency. Or maybe the team just slowed down. I can assure you that none of those were the case (on that last point, if anything, the team was working harder in February than they were in October). I can say that with confidence because this data is from a real release from a real team at a real company that I was asked to come in and consult on after the fact[1]. Believe me when I say that this was not a failure of planning or execution. This was a failure of forecasting.

Conclusion

I know many of you are chomping at the bit to get to the mechanics of forecasting itself. I am too. But I would beg your patience for just a bit longer as we explore some concepts crucial to understanding why today's forecasting orthodoxy fails, and why the methods we will describe in the coming chapters are superior.

Chapter 2 - The Flaw of Averages

To put it bluntly, your team in the previous chapter failed because they fell victim to something known as the "The Flaw of Averages" (FoA). FoA is a term coined by Dr. Sam Savage in his book of the same name[1], and can be simply stated as:

"Plans based on average assumptions fail on average."

You might not have realized it at the time, but when you calculated your initial required completion rate of 7.09 items/day, what you really calculated was an *average* completion rate. By doing so, you gave yourself only an average chance of succeeding. But it's worse than that. Because when I say "an average chance of succeeding", most people think that means having at least a 50% chance (or more) of being right. However, as we'll see, "an average chance of succeeding" could in fact be much lower than 50%—and might even be 0%.

[**Authors' Note**: in most contexts when people say "average" what they really mean to say is "arithmetic mean". Thus, most of the examples in this book will use "average" and "arithmetic mean" interchangeably, and we will try to specify a distinction if one is necessary. It's even more complicated than that because native English speakers are more heinous in their use of "average". Many times native English speakers will use the words "average" and "typical" interchangeably. As when they say something like, "the average person can't solve a differential equation." However, we will soon see that "average" is anything but "typical".]

What Is An Average?

It might seem a bit strange to ask what an average (mean) is. After all, we are taught averages fairly early in our mathematics schooling. One of the reasons why we are taught averages so early is because the arithmetic is so easy. But while the arithmetic may be straightforward, the interpretation of the result is not. It is this fundamental lack of understanding of the result that gets advocates for averages into trouble.

When you compute an average, what you are really getting is some notion of the central tendency of your dataset—and not a very good one at that as we will see in a little later. We're not sure where we got this next quote (I think it was from Dr. Wheeler), but we think it perfectly sums up what an average is:

"An average (arithmetic mean) is a measure of location, not of magnitude."

The best way to explain what is meant by this is to get into the flaw of averages itself, but as we go through this discussion, we'd like you to please keep this quote in mind.

The Flaw of Averages Explained

Dr. Savage uses the following example in his book to illustrate the FoA. Let's assume there is a 9:00 AM meeting with 10 people invited, and that all attendees must be present before the meeting can begin. Let's further assume that, on average, all participants have a history of arriving to meetings on time (for this example we'll say that average means a 50% on-time record). What is the chance that the meeting will start on time?

At first blush, the answer to this question might seem easy. If, on average, everyone has a history of arriving on time, then it is reasonable to assume that there is an average chance that the meeting will start on time. That answer, unfortunately, is wrong. If everyone has the same chance of arriving on time as arriving late, then there is actually only a 0.1% chance that the meeting will start on time. Think of it this way: since every invitee has a 50% chance of arriving on time, then you could use the flip of a coin to model if a given attendee will arrive punctually—heads she/he does and tails she/he doesn't. Remember the meeting can only start when all participants arrive. Therefore, the case where the meeting starts on time is the equivalent of flipping 10 heads in a row—flipping only one tails means that the participant is late and the meeting itself starts late. The chance of flipping 10 heads in a row is 0.1% ($1/2^{10}$)—or about 1 in 1,000. There is virtually no chance the meeting starts on time—which is significantly worse than average. (Maybe you understand now why none of the meetings that you schedule actually begin promptly...)

This point can be illustrated by a joke you've probably heard, "If [insert name of world's richest person here] walks into a bar (pub), then, on average, everyone in the bar is a billionaire". The simple truth is that averages don't mean much outside of some very specific use cases. And forecasting isn't one of those exceptions.

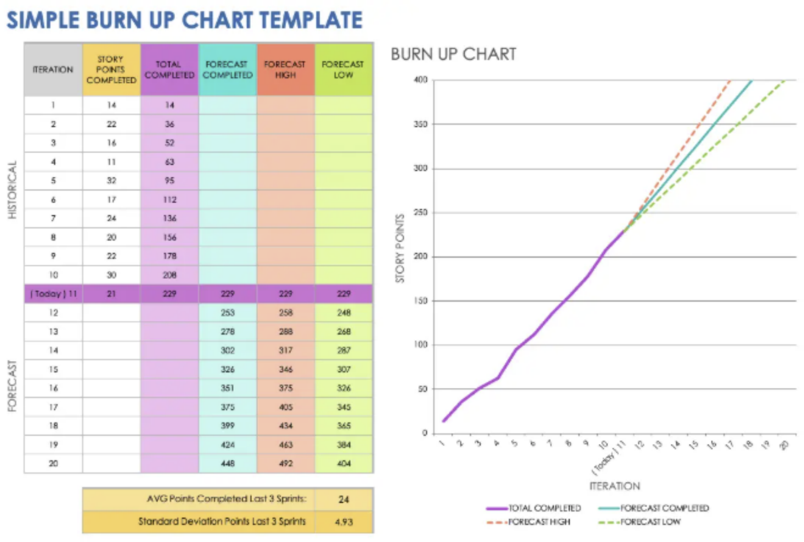
The lesson here is that any time you hear someone say "on average..." your ears should perk up because anything after the "on average..." statement will contain little to no informational value. For example, I currently live in South Florida, and as you may know, Florida is fairly prone to being hit by big storms called hurricanes. Before every hurricane season, forecasters go through their song and dance to try and predict the severity of the upcoming season. You will hear statements like, "The 2025 hurricane season will be more active than average." It should be immediately obvious to you now how that statement is a classic case of the FoA. In other words, it contains no informational value whatsoever. By comparing a single value (the 2025 hurricane season) to an average (the average of all hurricane seasons in the past) then you would expect that about 50% of the time the upcoming season will be more active than average and about 50% of the time the upcoming season will be less active than average. In other words, you would be right about half of the time and you would be wrong about half of the time. So saying the 2025 hurricane season will be more active than average doesn't really tell us anything because that forecast has just as much chance of being wrong as being right.

And we haven't even gotten to the most sinister part of FoA. In many scenarios, it is not even physically possible to achieve an "average" outcome. For example, the average outcome of a single roll of a fair, six-sided die, is 3.5. But we know that it is impossible to roll exactly a 3.5, so if you were to forecast based on an "average" roll, you would have a 0% chance of being right. Another great example of this—depending on what source you cite—the "average" U.S. family has 2.4 children[2]. I don't know about you, but I find that statistic extremely cruel. Obviously, you could meet every single family in the country and never once would you come across one with 2.4 children.

Even when the average is in the realm of possibilities, the average outcome usually isn't very likely. Let's say we want to roll two, fair, six-sided dice. The average outcome in this scenario is (did you have to look it up?) about 16.7%. This is a quirk of averages that most people have a hard time coming to grips with. Even though rolling seven is the average outcome and even though in this case rolling seven is the most likely outcome, the chances of actually rolling a seven are pretty low. To quote my friend and colleague Frank Vega, "The most likely outcome is not very likely". In other words, would you bet on something if ***on average*** you only had a 16.7% (or 0%) chance of succeeding?

Agile and FoA

Sadly, the use of averages is endemic to the Agile world. Don't believe me? Do a quick search—any search—on burnup (or burndown) charts. I just did that very thing and found this example as one of my top results[3] (I spent less than two minutes searching):


SIMPLE BURN UP CHART TEMPLATE

	ITERATION	STORY POINTS COMPLETED	TOTAL COMPLETED	FORECAST COMPLETED	FORECAST HIGH	FORECAST LOW
HISTORICAL	1	14	14			
	2	22	36			
	3	16	52			
	4	11	63			
	5	32	95			
	6	17	112			
	7	24	136			
	8	20	156			
	9	22	178			
	10	30	208			
	(Today) 11	21	229	229	229	229
FORECAST	12			253	258	248
	13			278	288	268
	14			302	317	287
	15			326	346	307
	16			351	375	326
	17			375	405	345
	18			399	434	365
	19			424	463	384
	20			448	492	404

AVG Points Completed Last 3 Sprints:	24
Standard Deviation Points Last 3 Sprints	4.93

Figure 2.1: An Example BurnUp Chart

In Figure 2.1 you'll see that the "Forecast Completed" is based on a historical average—worse, it is based on only the average of the last three sprints (we'll talk about why "Forecast High" and "Forecast Low" are problematic as well a little later). Based on what you know now, what do you think this team's chances are of meeting that forecast?

A final example to try and drive this point home is a bit of a cliché from the math world, but, it seems, virtually unknown in the Agile world:

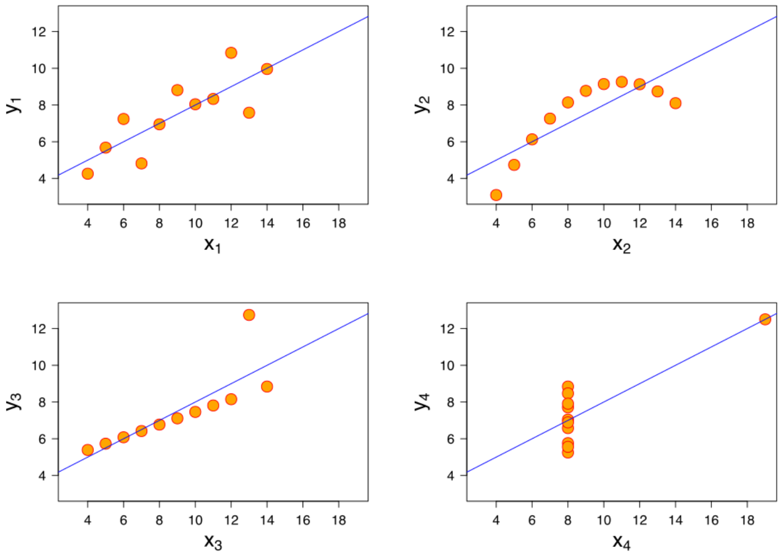


Figure 2.2: Anscombe's Quartet[3]

Figure 2.2 is known as Anscombe's Quartet and is an attempt by Francis Anscombe to illustrate the dangers of simply using descriptive statistics (like averages) to analyze a dataset. Though certainly not obvious, what makes Figure 2.2 so powerful is that all of the datasets have the same average X-value, the same average Y-value, the same standard deviation for X and Y, and the same correlation (amongst other things). Imagine this was data from four teams. Using an average as a forecast, you would expect each team to finish at exactly the same time because each team has exactly the same average. However, when you look at graphs, would you still make that same claim? The brilliance of this example is that it so perfectly demonstrates that the true danger of using descriptive statistics like "average" because those statistics hide all of the underlying uncertainty in your data.

Which takes us back to the example in Chapter 1. When the team in that scenario calculated their average completion rate, what chance do you think that gave them? 50%? 0%? Something else? How would they know if they never analyzed their data graphically, or using something other than "average" to quantify the underlying uncertainty? I can tell you one thing, that team behaved as if it gave them a 100% chance. Is your team any different? Is your company any different?

Conclusion

Are you an average forecaster? Hints might be if you use the word "average" in your forecast, e.g., "our average velocity is…" Or if you use straight-line projections for forecasting as in Figure 2.1. If you are guilty of one or more of these crimes against humanity, then you are an average forecaster.

The fundamental problem of using an average for a forecast is that it masks uncertainty. Unless we can quantify uncertainty, we'll never be able to quantify risk. And if we can't quantify risk, how will we ever know when to take action?

Chapter 3 - Uncertainty and Risk

Do you still commute to work? If you do, how long does it take you each day? If you are like most people, your answer to that question would be "It depends." For example, if your commute involves driving, your actual time might depend on factors like how many other cars are on the road, how many traffic lights you encounter, if there are accidents, if there is construction, etc. Or it is possible that you encounter none of these obstacles and our commute goes smoothly. In essence, when you answer "it depends"what you are really saying is that when you start your commute, there are many different possible future outcomes of how long it will take you.

Figure 3.1 below illustrates this point succinctly. These are drive times for the same commute route measured over several days[1].

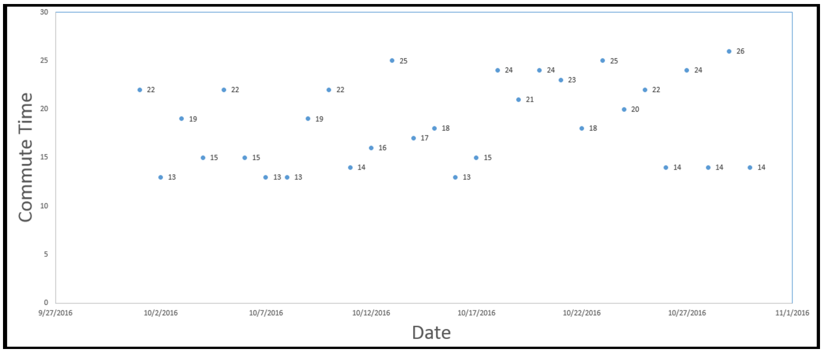


Figure 3.1: Commute Times

If Figure 3.1 looks random to you, then you are reading it correctly. While certain numbers do repeat themselves a few times, you rarely get the same number from one day to the next. The variation in data is down to certain causal factors occurring on a particular day. For example, maybe the 13's are when there are no issues at all. Maybe the 26 was due to an accident. Maybe the difference between getting a 13 and getting a 14 is one red light. The point is that all of the causes of this time variation are potentially present every day, it's just that we don't know on any given day which of these causes will hit (no pun intended). [As an aside, think about how you might use this data to answer how long a commute would take.]

What we have just stumbled across is our working definition of "uncertainty". I'm a Douglas Hubbard fan, so I am going to steal my definition of uncertainty from his book, "How To Measure Anything"[2]:

Uncertainty: "The existence of more than one possibility. The 'true' outcome/state/result/value is unknown."

Which brings us to the first rule of forecasting (no, it's not "Don't talk about forecasting") which is that all statements about the future contain uncertainty. Whether you want to acknowledge it or not, whether your stakeholders want to acknowledge it or not, no one knows with 100% certainty what is going to happen in the future.

What that means is that if we are going to provide a forecast, we must first quantify the uncertainty associated with what we are trying to forecast. This quantification will usually come in the form of some type of probability. Probability is, after all, the mathematics of uncertainty. [Though it should be noted at the outset that the mathematics we are going to use is very simple. If you can do basic arithmetic, then you can use our forecasting techniques. More on that in the coming chapters].

To torture our commute metaphor a bit more, you'll notice that each of the possible future outcomes could readily be categorized as either "bad" or "good" (or, maybe more correctly, "not bad"). Let's say that I work a job where I have to clock in every morning at 09:00. If I clock in before 09:00 then everything is ok. If I clock in after 09:00 then I get penalized—up to and including losing my job. So even though I know my commute times are going to be different, I also know that every time I leave my home and get to work before 09:00 is not a bad outcome and every time I leave my house and get to work after 09:00 is a bad outcome.

Understanding that the universe of all possible future outcomes includes some bad ones gives us our working definition of risk (hat tip again to Dr. Hubbard):

Risk: "The possibility that something bad will happen"[3].

Simply stated, all forecasts should communicate the uncertainty associated with future outcomes so that we can assess the associated risk. Only by understanding risk can we have a true guide as to when and if to take action.

Combining Figure 2.1 with our newfound understanding of risk, if I could lose my job if I don't arrive by 09:00, when should I leave home? Or what about if it is no problem that I am late from time to time, when should I leave home? Or what about if I have a meeting with the CEO at 09:00, when should I leave home?

Getting back to the scenario in Chapter 1, we know that a "not bad" outcome would be for our release to go out any time before March 1. Here I'll repeat my questions from before: when we said that we needed to complete 7.09 items/day, what did that mean in terms of uncertainty? Did that give us a 50% chance? A 16.7% chance? A 0% chance? Something else? The simple fact of the matter is that we don't know.

When most teams don't know about uncertainty, they simply ignore it and assume that there is none. The team in our scenario assumed that as long as they maintained a run rate of at least 7.09 items/day that they would have a 100% chance of being successful (i.e., a 0% chance that something bad would happen). In other words, they assumed they had no risk and therefore took no action. Look how that worked out.

Conclusion

In its most basic sense, a forecast is a statement about the future. The future is full of different possible outcomes—some bad and some not bad. Being able to quantify the uncertainty around the bad outcomes is what forecasting is all about. But we now know traditional forecasting methods (using averages) won't get us there. Stated in clearer terms, we need a forecasting approach that will give us a better than average chance of succeeding while at the same time communicating the risk associated with our prediction. How are we supposed to do that?

We are glad you asked…

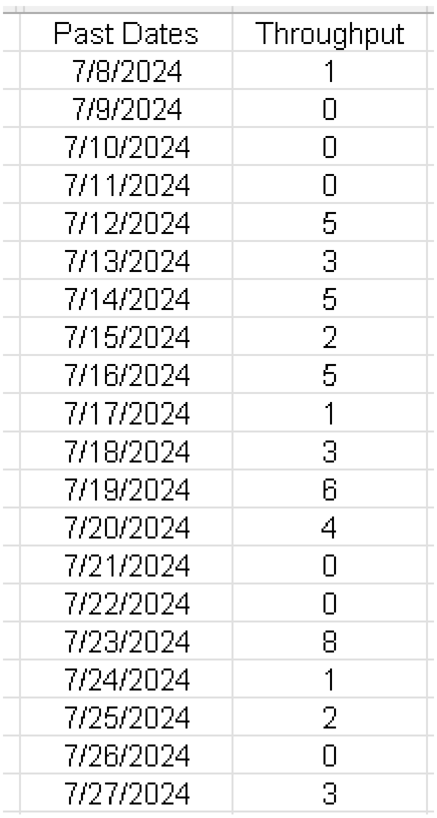
Chapter 4 - Monte Carlo Simulation

Getting back to the dilemma of Chapter 1, what would have been an appropriate method to forecast how many items the team could have completed?

To answer that, I'd like to walk you through a different, simpler scenario. This time, it is July 28th and you are on a team with a release coming up 30 days from today. You want to know how many items you can get done in that time frame.

To come up with your forecast, you decide you are going to start by collecting some data. As this is a pocket guide, I'll cut to the chase and tell you that to forecast how many items you can complete by a given date is going to first require you to have data for the flow metric of Throughput. [What Throughput is and what data you need to calculate it is detailed in Appendix A, so if you are unfamiliar with either of those concepts, please start there. Also, if you are interested in a detailed discussion of why you need Throughput data please reference my book, "When Will It Be Done?"[1]]

So let's say you believe me and you calculate Throughput for the last 20 days. Why 20 days? Because that's all you have data for (we'll discuss what and how much data is needed in a later chapter). Following the procedure outlined in Appendix A, you calculate the following historical Throughput for your process:



Past Dates	Throughput
7/8/2024	1
7/9/2024	0
7/10/2024	0
7/11/2024	0
7/12/2024	5
7/13/2024	3
7/14/2024	5
7/15/2024	2
7/16/2024	5
7/17/2024	1
7/18/2024	3
7/19/2024	6
7/20/2024	4
7/21/2024	0
7/22/2024	0
7/23/2024	8
7/24/2024	1
7/25/2024	2
7/26/2024	0
7/27/2024	3

Figure 4.1: Historical Throughput

Before we go on, I'd like you to take a minute and carefully examine the Throughput data in Figure 4.1. What do you notice? First, you might be struck by the variation in this data. For example, this team does not get exactly the same number of items completed every day. In other words, they are not getting 1 item done every day. Or 2 items done every day. Or 0 items done every day. While obvious, this fact should not be discounted. What might cause this team to differ in its output from day to day? Why on some days were 0 items completed, but on another day 8 items were completed? What might be some reasonable answers to those questions? Let's think for a moment what those 0's might represent Maybe they represent weekends. Maybe they represent days the team was out in all-day training. Maybe the 8 represents the last day of a sprint. Maybe the 1's and 2's are when some team members were out (because of sick or vacation time).

Most importantly, however, consider whether it is reasonable that a team's data would look like this. Is it reasonable that their data would exhibit this amount of variation? Maybe also consider if whatever factors were present to create this variation could be reasonably expected to continue (like the commute example)?

I would argue the answer to all of these considerations is "yes". In the future, there will be weekends and vacations and sick time and ends of sprints and whatever. In most cases, there's no reason to believe that how our data looked in the past will be largely different from how our data looks in the future. But how might that knowledge help us to forecast what is going to happen in the next 30 days? Easy. Let's just make our assumption explicit that how the team worked in the past is going to be roughly similar to how it will work in the future. That means we will be justified in using the data in Figure 4.1 to project what will happen in the future.

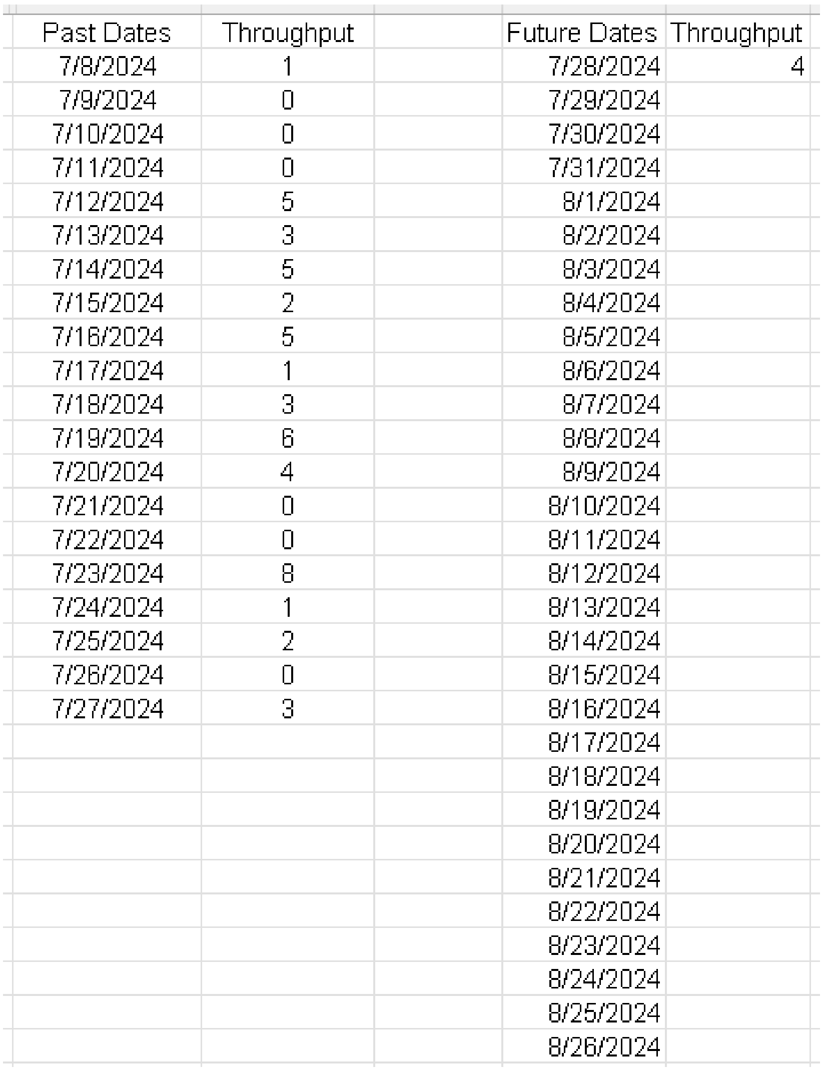
If that's the case, then one thing we could do is simply total the number of items we completed in the past 20 days and say that's what we will complete in the future. While not necessarily a bad place to start, there are at least two problems with this approach as pertains to this scenario. First, we only have 20 days of historical data, so how could summing 20 days tell us how much we can get done in 30 days? That is, how do we make up for that 10-day deficit? Second, as we so painfully demonstrated in the last chapter, simply totaling the items completed gives us a single-point answer that in no way communicates the uncertainty—and therefore the risk—associated with our forecast.

Next, you might think that we could simply average our Throughput data, multiply that average by 30, and use that as our forecast. I'm sure by now I don't need to say anything more about why that solution is deficient. [Note: for those of you keeping score at home, the average of Figure 4.1's data is 2.45, which when multiplied by 30 gives an answer of 73.5. That may become important later—but not for good reasons.]

Frustratingly, all of these methods give a single point number and in no way model the uncertainty that is guaranteed to exist. That is, we know that uncertainty means that we are facing many possible futures, so while we can expect the events that caused variation in the past to continue in the future, the one on thing we can't know is *when* those events will happen. So why not just guess? That's a big statement, and might mean something different than what you are assuming right now, so let's take some time to unpack what I am talking about.

Recall that we are on July 28. We know that today will roughly look like one of the 20 days that we've encountered in the past. But which one? Will today be a day where lots of people are sick? Or out for meetings? Or will today be a day that everyone knocks it out of the park? Since we can't know the exact conditions before we start, why bother trying to come up with a deterministic answer? Why not just randomly pick a day from the past and assume that's what today will look like?

To do this, we might use something like Excel to help us track what we are modeling. We could set up a spreadsheet by first enumerating all of the days in the future that we are going to forecast. In this case, our forecast days would be July 28 through August 26. Then, for the first day of our forecast, today, we could have Excel randomly pick a day from our past Throughput data and "assign" it to be our first day's forecasted Throughput. [Or if you had the right dice handy, you could roll a 20-sided die to pick a day. How you randomly pick that past day is up to you—as long as it random.] I'm doing this experiment in Excel as I write, so I'm going to have Excel do all of the hard work for me. I've set up my spreadsheet to randomly pick a day from our past Throughput, and for this first day, Excel chose a 4. You can see what I've done so far in Figure 4.2:

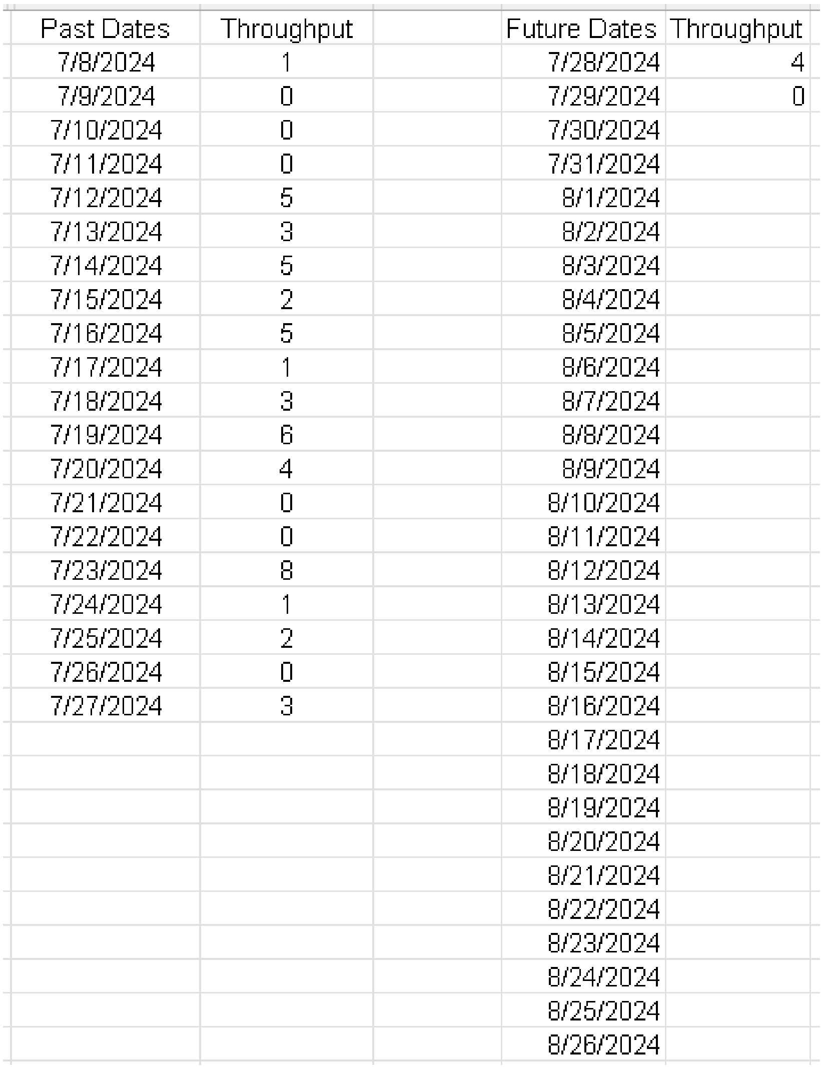


Past Dates	Throughput		Future Dates	Throughput
7/8/2024	1		7/28/2024	4
7/9/2024	0		7/29/2024	
7/10/2024	0		7/30/2024	
7/11/2024	0		7/31/2024	
7/12/2024	5		8/1/2024	
7/13/2024	3		8/2/2024	
7/14/2024	5		8/3/2024	
7/15/2024	2		8/4/2024	
7/16/2024	5		8/5/2024	
7/17/2024	1		8/6/2024	
7/18/2024	3		8/7/2024	
7/19/2024	6		8/8/2024	
7/20/2024	4		8/9/2024	
7/21/2024	0		8/10/2024	
7/22/2024	0		8/11/2024	
7/23/2024	8		8/12/2024	
7/24/2024	1		8/13/2024	
7/25/2024	2		8/14/2024	
7/26/2024	0		8/15/2024	
7/27/2024	3		8/16/2024	
			8/17/2024	
			8/18/2024	
			8/19/2024	
			8/20/2024	
			8/21/2024	
			8/22/2024	
			8/23/2024	
			8/24/2024	
			8/25/2024	
			8/26/2024	

Figure 4.2: Assignment for the First Day of our Forecast

This random assignment means that we are forecasting that our team will complete 4 items today. If you look closely at Figure 4.2, you will see that the 4 we chose corresponds to the Throughput value from July 20. Will today, July 28, indeed look anything like July 20? Again, it is impossible to know, but given the information we have at this time, July 20 seems like as good a day as any to choose for our assignment for today, so we'll stick with that.

Now we move on to the next day, July 29. The procedure is exactly the same. Which day in the past will July 29th look like? Just like before, there is no way to answer this deterministically, so let's randomly pick a past day. Again, I had Excel do this for me and the result is Figure 4.3.

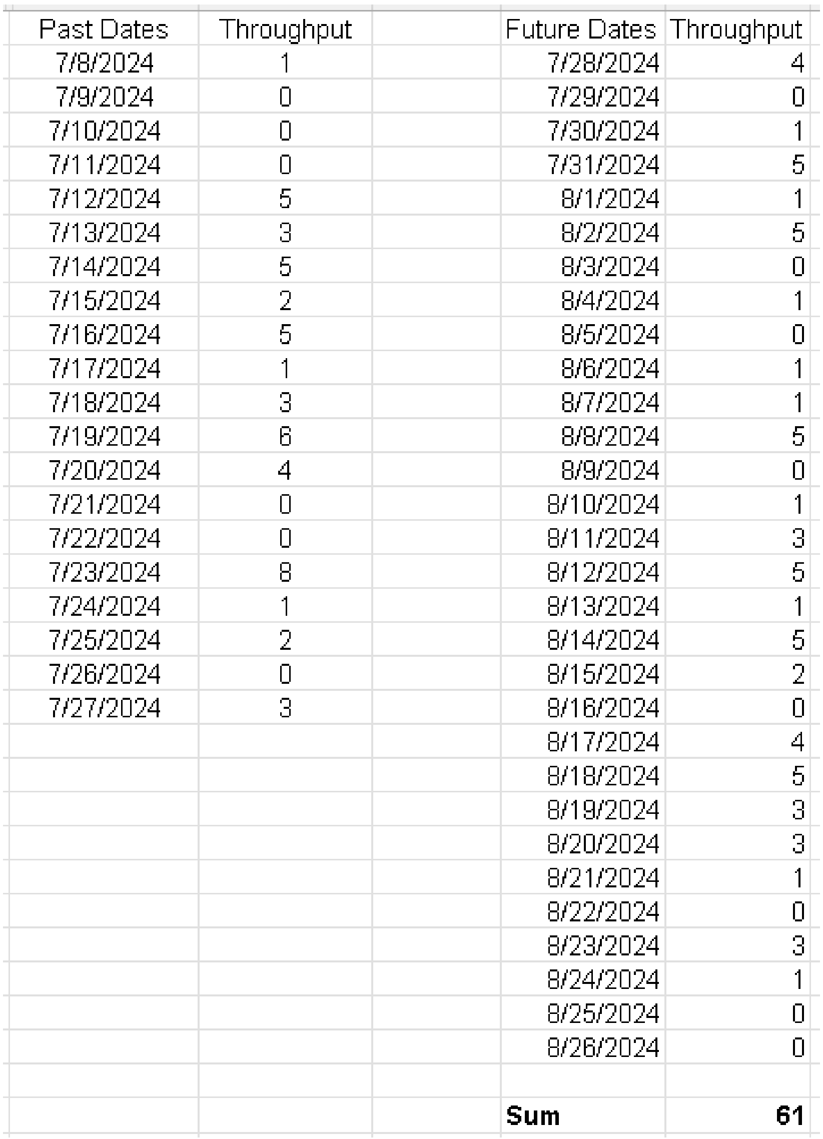


Past Dates	Throughput		Future Dates	Throughput
7/8/2024	1		7/28/2024	4
7/9/2024	0		7/29/2024	0
7/10/2024	0		7/30/2024	
7/11/2024	0		7/31/2024	
7/12/2024	5		8/1/2024	
7/13/2024	3		8/2/2024	
7/14/2024	5		8/3/2024	
7/15/2024	2		8/4/2024	
7/16/2024	5		8/5/2024	
7/17/2024	1		8/6/2024	
7/18/2024	3		8/7/2024	
7/19/2024	6		8/8/2024	
7/20/2024	4		8/9/2024	
7/21/2024	0		8/10/2024	
7/22/2024	0		8/11/2024	
7/23/2024	8		8/12/2024	
7/24/2024	1		8/13/2024	
7/25/2024	2		8/14/2024	
7/26/2024	0		8/15/2024	
7/27/2024	3		8/16/2024	
			8/17/2024	
			8/18/2024	
			8/19/2024	
			8/20/2024	
			8/21/2024	
			8/22/2024	
			8/23/2024	
			8/24/2024	
			8/25/2024	
			8/26/2024	

Figure 4.3: Assignment for the Second Day of our Forecast

Here, you see Excel picked 0. Notice, however, there are a lot of zeros from our past data, so one thing we don't know is if the 0 we just selected for July 20 corresponds to the 0 we saw on July 9, or the 0 we saw on July 10, or whether it corresponds to any other date that had a 0 in the past. But here's the good news: we don't care! All we are trying to say is that July 20 looks like a day in the past. We saw a zero in the past, so it is reasonable that we will see a zero in the future, therefore on July 29th, we will say we will complete 0 items. [The astute reader will realize that since there are a lot of 0's in our past data, that our chances of choosing a 0 for any given day in the future are greater than our chances of choosing, say, a 6. As always, we'll discuss the implications of that in a bit—but at this point can you guess what those implications might be?]

Repeating this process, we can select Throughput values for all of our future forecast days. Once we have all future days assigned, we can then sum up all of those future Throughput values to get a number items that we can complete by the end of the 30th day:



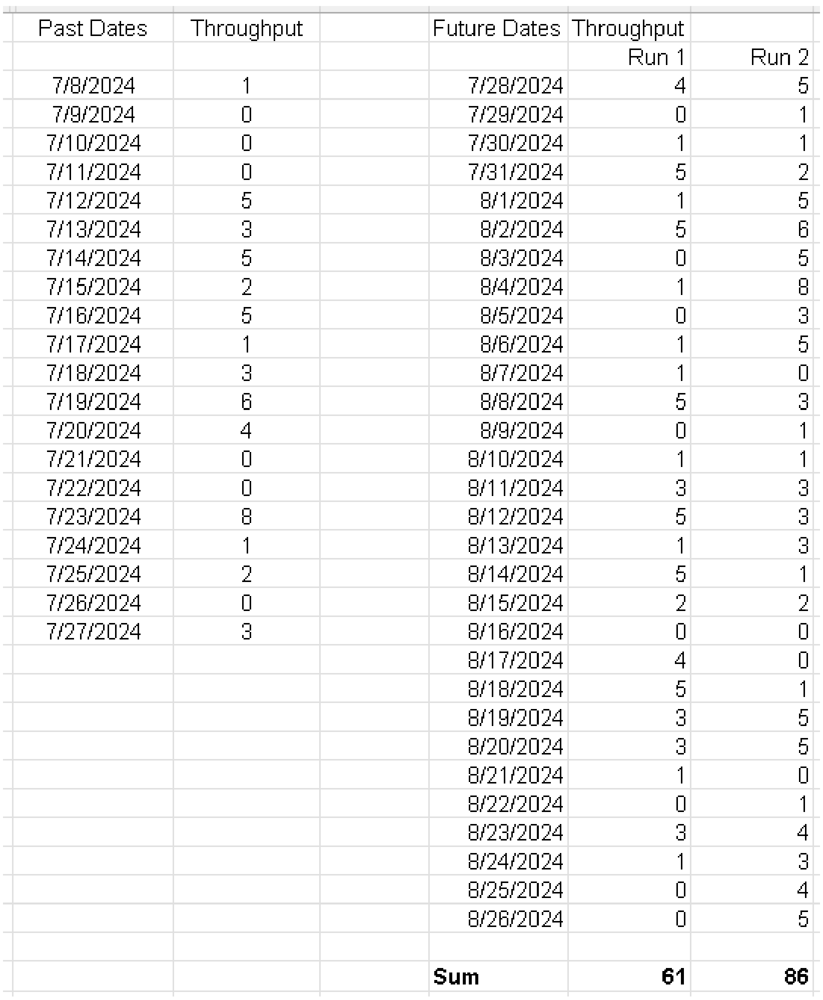
Past Dates	Throughput		Future Dates	Throughput
7/8/2024	1		7/28/2024	4
7/9/2024	0		7/29/2024	0
7/10/2024	0		7/30/2024	1
7/11/2024	0		7/31/2024	5
7/12/2024	5		8/1/2024	1
7/13/2024	3		8/2/2024	5
7/14/2024	5		8/3/2024	0
7/15/2024	2		8/4/2024	1
7/16/2024	5		8/5/2024	0
7/17/2024	1		8/6/2024	1
7/18/2024	3		8/7/2024	1
7/19/2024	6		8/8/2024	5
7/20/2024	4		8/9/2024	0
7/21/2024	0		8/10/2024	1
7/22/2024	0		8/11/2024	3
7/23/2024	8		8/12/2024	5
7/24/2024	1		8/13/2024	1
7/25/2024	2		8/14/2024	5
7/26/2024	0		8/15/2024	2
7/27/2024	3		8/16/2024	0
			8/17/2024	4
			8/18/2024	5
			8/19/2024	3
			8/20/2024	3
			8/21/2024	1
			8/22/2024	0
			8/23/2024	3
			8/24/2024	1
			8/25/2024	0
			8/26/2024	0
			Sum	**61**

Figure 4.4: First Full Run of Our Model

Once again, let's pause and reflect on what's happening in Figure 4.4. What do you observe about our randomly selected values? Did you notice that we never assigned an 8? Or that we also never assigned a 6? How can that be since both an 8 and a 6 were in our past data?

Shouldn't they show up in our future assignments? Not to mention there are more 5's and 0's in our future values than there were in the past. Is that a problem? [Also note that our sum of 61 is lower than our "average" forecast.]

Well, all of those observations would be problems if we simply stopped here. However, what we have modeled up to this point is **one** possible future. But remember that we know that there are **many** possible futures. Since "many" is more than "one", we can't quite consider our forecast finished. We need to model more "futures". So what would happen if we repeated this procedure one more time and tracked it alongside the first "run" of our model? Luckily for you, I just did that in my spreadsheet and have displayed those results below in Figure 4.5:



Past Dates	Throughput		Future Dates	Throughput	
				Run 1	Run 2
7/8/2024	1		7/28/2024	4	5
7/9/2024	0		7/29/2024	0	1
7/10/2024	0		7/30/2024	1	1
7/11/2024	0		7/31/2024	5	2
7/12/2024	5		8/1/2024	1	5
7/13/2024	3		8/2/2024	5	6
7/14/2024	5		8/3/2024	0	5
7/15/2024	2		8/4/2024	1	8
7/16/2024	5		8/5/2024	0	3
7/17/2024	1		8/6/2024	1	5
7/18/2024	3		8/7/2024	1	0
7/19/2024	6		8/8/2024	5	3
7/20/2024	4		8/9/2024	0	1
7/21/2024	0		8/10/2024	1	1
7/22/2024	0		8/11/2024	3	3
7/23/2024	8		8/12/2024	5	3
7/24/2024	1		8/13/2024	1	3
7/25/2024	2		8/14/2024	5	1
7/26/2024	0		8/15/2024	2	2
7/27/2024	3		8/16/2024	0	0
			8/17/2024	4	0
			8/18/2024	5	1
			8/19/2024	3	5
			8/20/2024	3	5
			8/21/2024	1	0
			8/22/2024	0	1
			8/23/2024	3	4
			8/24/2024	1	3
			8/25/2024	0	4
			8/26/2024	0	5
			Sum	**61**	**86**

Figure 4.5: Historical Throughput

What do you observe now? First, you may see that the sum of 86 in this run of the model is higher than the 61 one we got before. So which of these two numbers is more "correct"? And what's causing this second total to be higher? Well notice that we now have sampled an 8 and a 6 where we hadn't before. There are also less zeros. Again I ask, are any of these things problems?

Again, I would answer they would be problems if we stopped here. You see, what we are demonstrating is the power of randomization.

Because we don't know exactly how the future is going to work out, what we should do is simulate as many futures as possible to get a sense of what reasonably might happen. Imagine, therefore, that you were using a spreadsheet like mine to run these models and instead simulating two "runs", you simulated 1,000 runs. What would that look like? [For the record, we would strongly encourage you to try this out for yourself in your own spreadsheet to see what you come up with.]

Once more I have done the hard work for you and have modified my spreadsheet to show 1,000 runs of our model. But rather than throwing thousands of cells in an Excel file at you (not to mention that I don't think I could fit them all on this page), let's consider an easier way to summarize the data from those 1,000 runs. One of the best tools for this task is to use something called a Histogram. If you don't know what a Histogram is, please see Appendix C. For now, however, I am going to assume you know what a Histogram is and how to construct one. As such, Figure 4.6 is a Histogram I built from my spreadsheet of 1,000 runs of our forecast model:

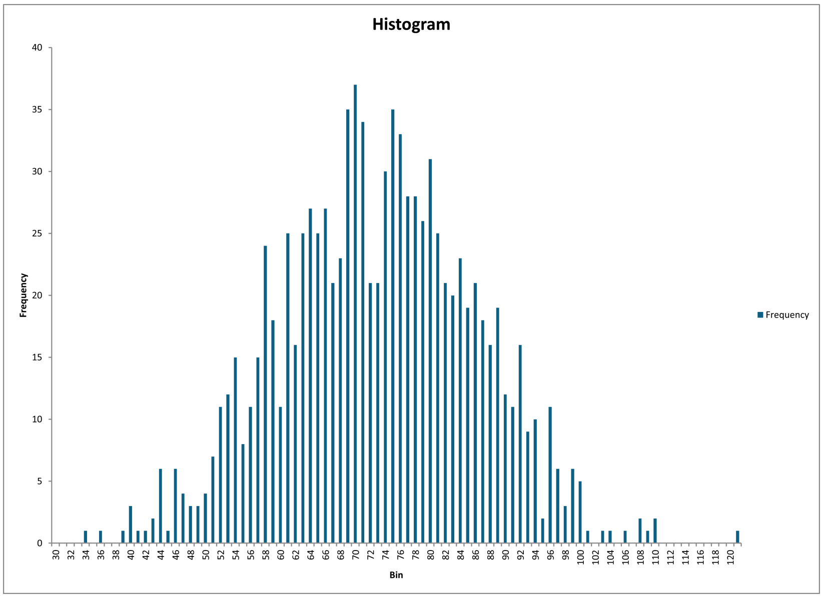


Figure 4.6: 1,000 Runs

Now we are getting somewhere. I believe that Figure 4.6 gets us to a point where we have a much clearer picture of the uncertainty associated with the question we are trying to answer. And while we are desperate to jump to how we might interpret these results with you, let's take a minute to review how we got here:

First, we collected historical Throughput for our team. In this case, that was 20 days worth of data, with Throughput values ranging from 0 to 8. Then we decided how many days out into the future that we wanted to forecast (30 days). Then we made the assumption that how the team will preform over the next 30 days will roughly look like how the team performed over the past 20 days. Keeping that assumption in mind, we randomly "assigned" a Throughput value from one day in the past to each future forecast day. We used this random assignment to capture the uncertainty around the causes of variation in our past Throughput data. Because we don't know what exact days those causes will manifest themselves in the future, randomization seemed like an obvious approach. Once we assigned Throughput values to each future day, we summed up those values to get a candidate forecast result. We called that one "run" of our model. However, we realized that our randomization technique only works if we are able to explore several permutations of future assignments. That is why we performed many "runs"—1,000 to be exact—and plotted the frequency of the results of all of those runs in a Histogram. The result of which is Figure 4.6.

With me so far? We hope so, because in case you were wondering, what we just did was perform a Monte Carlo Simulation (MCS).

Conclusion

A more rigorous definition of MCS is provided in Appendix B. All you really need to know, though, is that whenever you are faced with a problem where you want to model uncertainty in a given problem (as in forecasting how many items you can complete) MCS is a powerful place to start.

Now that we have preformed our MCS, it's time to talk about how to interpret those results.

Chapter 5 - Interpreting Monte Carlo Simulation

To aid our interpretation of the results of the last chapter, let's re-run our simulation with 10,000 runs instead of 1,000. The generated histogram for the new iteration of our model is shown below:

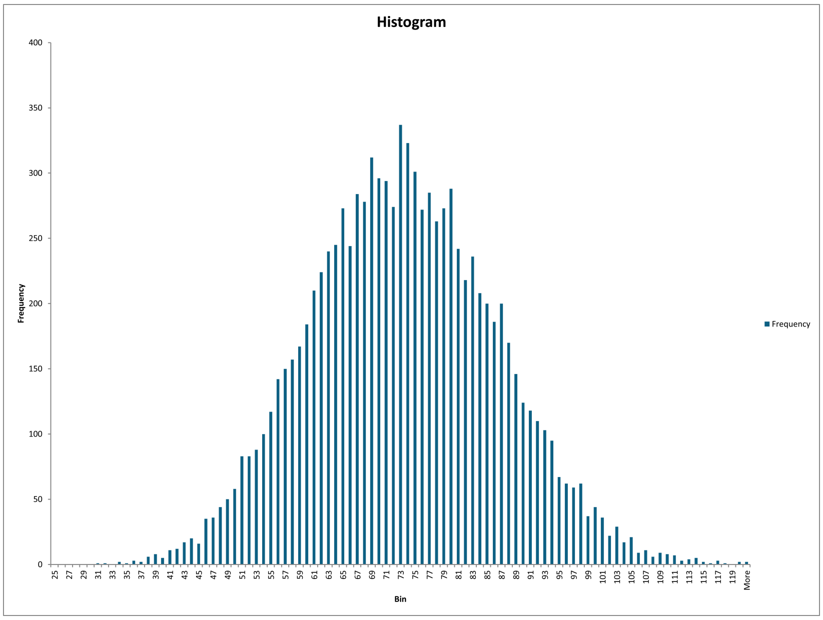


Figure 5.1: 10,000 Run MCS

To be perfectly clear, in this picture, all of the input data has remained the same, and we are still trying to answer the question of how many items we can complete in the next 30 days. The only thing that has changed is that there are 10,000 results on this graph instead of 1,000. Why did we do 10,000 runs? Mostly because we could. But more to the point, more runs have given us a clearer picture of what's going on with our simulation. This clarity should aid in our analysis.

As a first step, let's consider the shape of the curve on our Histogram. You'll notice that there are very few outcomes on the far left, and very few outcomes on the far right. What's going on in those parts? As you've probably guessed, the outcomes on the far left are runs where our model sampled many zeros and other very low numbers—but that didn't happen very often. Likewise, the outcomes on the far right are runs where our model sampled lots of 8's and 6's and other higher numbers—but that didn't happen very often, either. Taking this argument to the extreme, it would be possible to have a run that sampled all 0's as well as a run that sampled all 8's, giving us a theoretical "range" of possible outcomes spanning the interval from 0 to 240 items that we could complete in the next 30 days. The chances of getting those outermost outcomes, however, are astronomically low, which is why you don't see them on our Histogram.

The bulk of our outcomes occur somewhere in the middle of those far left values and far right values. This would make sense as it is much more likely that we get a mix of values in each run rather than the extreme values as was just discussed. In other words, what we have just described is what is commonly known as a bell curve [Why the results Histogram looks like this is due to something called the Central Limit Theorem (CLT). You don't really need to know that, we are just posting it here in case it is of any interest to you.]

It is at this point that you statistics geeks out there are going to want to jump to the conclusion that your outcomes are normally distributed and therefore it is reasonable to break out all of your probability hacks to analyze this curve. We are going to ask that you hold off on that for now. Mostly because it will not be necessary, but also because it's not really a valid approach (more on that later).

Having spent a bit of time looking at the shape of the curve, let's start looking at specific results. For instance, can you tell me which result in Figure 5.1 is the most likely outcome? It might be a little hard to see in the picture, but if you said 73 was the most likely, then you win a prize. We know 73 is the most likely because that is the tallest bar on the graph. Which brings us to a more important question: what is the chance of this team finishing exactly 73 items in the next 30 days? The way we figure out the chance of any specific outcome is to take the height of that result and then divide it by the total number of runs in our simulation. In this case, the height of the bar at 73 is 337. That means out of the total of 10,000 runs, we got the result 73 exactly 337 times which means our chance of finishing 73 items in the next 30 days is 337 / 10,000 or 3.37%. The FoA rears its ugly head again. Would you bet on an outcome if you only had a 3.37% chance of being right? Would your customers? My guess is they would not.

That's what things look like from an uncertainty perspective. What about from a risk perspective? Remember that when we talk about risk, we care about separating not bad outcomes from bad outcomes. Sticking with the 73 number then, what if we told our customers we could complete 73 items in 30 days, but we actually completed 74 items? Would our customers consider that a bad outcome or a not bad outcome? We would say that is not a bad outcome. Would they consider 75 a bad outcome or not bad outcome? What about 76? 77?

I think you get the picture. If we said we could finish 73 items, then any future result that lands to the right of 73 in Figure 5.1 would be considered "not bad". I shaded the area I'm talking about in Figure 5.2:

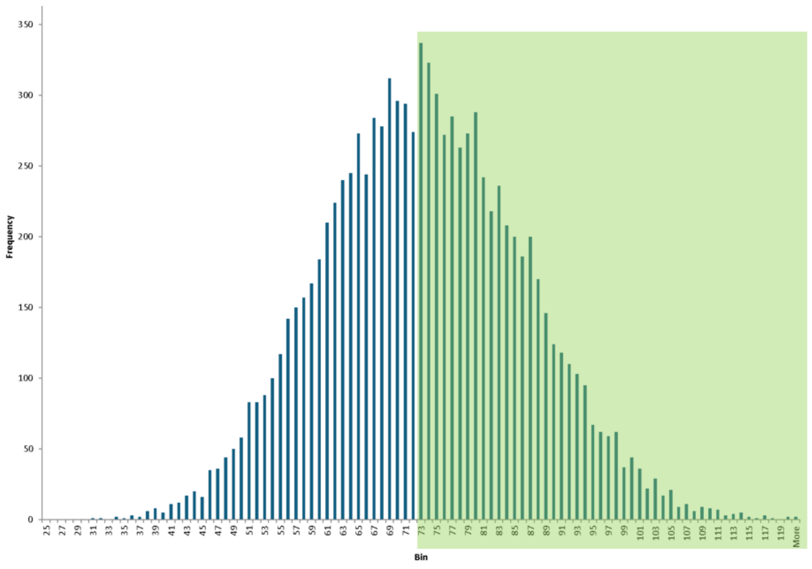


Figure 5.2: MCS 50th Percentile

The way that we figure out the probability of a result occurring in the green shaded area is to add up the heights of all of the bars in the area, and divide by the total amount of simulation runs. Doing that gives us a probability of (about) a 50% chance of a future outcome landing to the right of 73. In other words we have a 50% chance of getting 73 items *or more* finished in the next 30 days. This, now, gives us an understanding of risk. If our reference point is 73 then we have a 50% chance of getting a not bad outcome and a 50% chance of getting a bad outcome.

You might be thinking that a 50% chance still sounds a bit too risky, and we would probably agree. We think our customers would agree, too. The good news for us is that now that we understand how to quantify risk, we can do that for any risk level that our customers find acceptable. Maybe they only want to be wrong only 15% of the time (meaning they would be right 85% of the time), or wrong only 5% of the time. No problem, we can use our results Histogram to quantify what those ranges might be. We'll leave it as an exercise for the reader to figure out how to calculate both of those percentages off of the graph, but what we can do is post a picture of what both of those risk profiles look like:

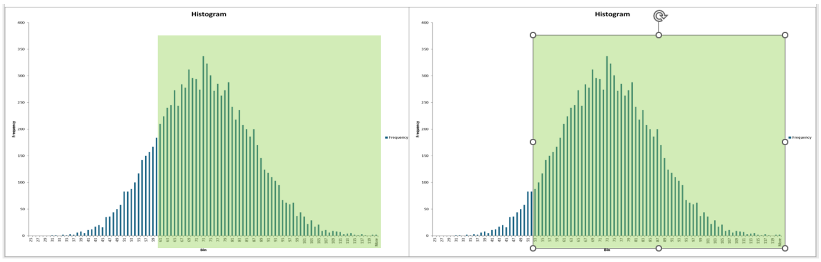


Figure 5.3: MCS 85th (left) and 95th (right) Percentiles

Plugging these values into my magic spreadsheet, we see that to be right 85% of the time (or wrong 15% of the time) would mean communicating a forecast of completing 60 items or more in the next 30 days. To be right 95% of the time would mean a forecast of 53 items or more. Some times you will see these so-called percentile lines plotted directly on the results Histogram as in Figure 5.4 [Note that this graph was generated using completely different data from a completely different team]:

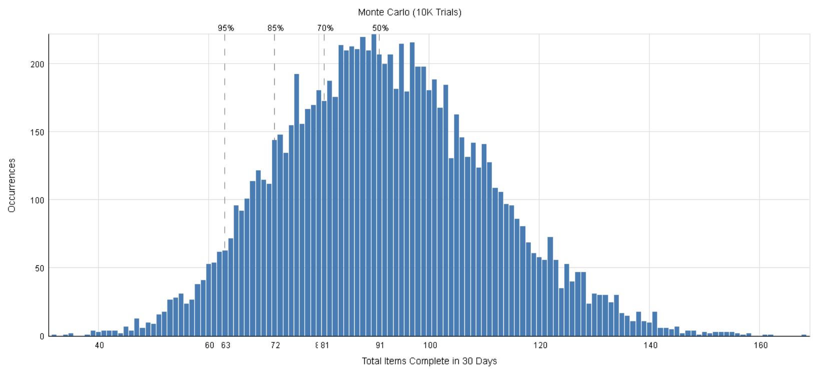


Figure 5.4: Results Histogram With Percentile Lines

In the above picture, you can see the 50th percentile forecast for this team would be 91 or more items completed in the next 30 days. For the 70th, it would 81 or more items, for the 85th it would be 72 or more items and for the 95th it would be 63 or more items completed in 30 days.

Getting back to our original results, allow us to put the percentiles we calculated into a table to make them a bit easier to read:

Percentile	30-day Forecast
50th	73 items or more
85th	60 items or more
95th	53 items or more

Table 5.1: List of Forecasts

What do you notice about our forecasts as our risk tolerance goes down—i.e., as we want more "confidence" in our forecast? It should be plain to see that as those percentiles go up, the number of items we can say we can get done goes down. This should make intuitive sense. If you want me to be more certain about my forecast, I'm going to tell you I can do less. One thing you'll need to work out for your own particular situation is how to strike that right balance between requiring confidence and providing a forecast that is useful. For example, let's say my customers demand 100% certainty from my forecast. What would I tell them? Easy, if you want 100% confidence, I'll tell you that I can get 0 or more items done. I'm sure you see how that forecast does nobody any good. We have to realize that uncertainty creeps in the second we attach a number greater than 0 to a forecast (see Chapter 3), which means any meaningful forecast will be some confidence less than 100%. That's just the way the world works. The sooner both you and your stakeholders accept this, the sooner you'll be on your way to effectively using MCS for proper risk management.

The Language of Forecasting

Which brings me to the final point about the language of forecasting. Because we now know that a single number cannot appropriately capture the uncertainty associated with any prediction, notice the language that we have been using to communicate our forecasts. Every forecast that we've written in this chapter has been in the form of a range and a probability that a "not bad" result will occur in that range. E.g. "We have a 85% chance of getting 60 items or more completed in the next 30 days." The range in that statement is "60 items or more" and the probability is "85%". All forecasts should be communicated in this way. Only by being explicit about the uncertainty associated with our forecast can we begin to have meaningful conversations around what actions to take when a forecast does not line up with your risk tolerance. For the rest of this book we will continue to use this language. We hope you will too.

Conclusion

What we have demonstrated here is a powerful tool that we can share with our stakeholders to quickly and easily come up with a forecast based on an acceptable level of risk. Notice that we did not have to take the team offline to give us estimates that would be useless anyway. Nor did we project using an average that would have us failing as much as 97% of the time. All we needed was some historical Throughput data and Robert's your father's brother. (Ok, so it's not quite *that* easy but you'll have to wait for a later chapter for a discussion around what things you have to consider to get the most out of MCS—but it will still be much easier than traditional deterministic planning techniques).

As great as all of that sounds, we are not done with our discussion of MCS. Turns out there are plenty more questions we can answer with this technique, so let’s discuss some of those next.

Chapter 6 - More Monte Carlo Simulation

The forecasting examples we've examined so far have all been of the form "given a specific date in the future, how many items can we get done?". That's a perfectly valid question, and one that we will continue to explore later in this chapter, but there is also a flip-side to this problem—a flip-side that may be more relevant to your situation. And that is, "given a number of items to complete, what date in the future can we say those items will be completed by?"

MCS For A Given Number of Items

Let's say you've done some analysis on a new feature that you want to implement, so you've broken down that feature into 50 items that must be completed. Now your Product Manager comes to you and asks "When will those 100 items be done?" How do you answer her?

Again, as this is a pocket guide, we are going to give you the answer in the interest of expediency: You still will perform an MCS using historical Throughput data, only now the algorithm around how you aggregate results will be a little different. Let's see how.

Following the exact approach as before, to project out how long it will take 50 items to finish, we are going "simulate" how many items get done for each consecutive day into the future. For each day in the future, we are going to select a random Throughput number from our historical (input) Throughput data. To illustrate, let's say it is January 1 and we are starting our simulation today. So we go to our historical input Throughput data and randomly select a 3 (for clarity, let's assume we are working with a completely different team and completely different data than we did in the previous chapter). That means that we are predicting that our team will get 3 things done today (January 1). We then subtract 3 from 50 to get 47 items left to complete. We then repeat the same calculation for tomorrow, January

2 (i.e., the next day of the simulation). Let's say we randomly choose a 5 for January 2. That means we are predicting to get 5 things done on January 2 so we subtract 5 from 47 to get 42. We do this over and over until we either get a 0 (or less than zero—whichever comes first). We then note the date that we got to zero. For this example, let's just say that date is January 28. So far, so good.

You will recall that randomization depends on us simulating **many** futures, not just one. So of course we would not say that January 28 is our forecast date at this point. To come up with our answer, we will need to run our model many more times. And just like before, for each "run" of the simulation, we are can track the results in what we are in our results Histogram. The difference with this new histogram is that instead of number of items across the bottom (the x-axis) as in Figure 5.1, we will list the different completion dates that we get. An example of what this type of Histogram would look like is shown in Figure 6.1:

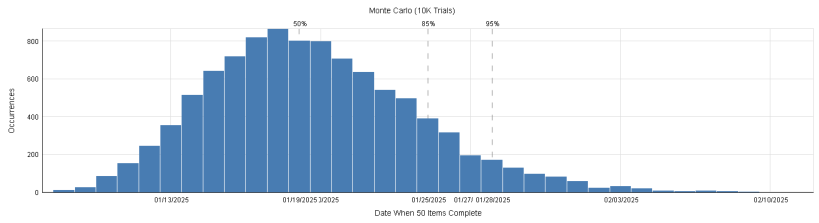


Figure 6.1: MCS for 50 Items

In Figure 6.1, you'll see that we've run the simulation 10,000 times. As before, we do many trials to get a clear picture of what the result set looks like. With each run of the simulation, the probability distribution as described by the Results Histogram will become clearer and clearer—in chapters 4 and 5.

The interpretation of the results in Figure 6.1 are similar to what we did previously, with one important distinction. First, however, let's talk about what's similar. You'll notice the familiar bell-shaped curve which gives us the uncertainties associated with certain outcomes. You've probably noticed as well that we can segment these outcomes into "bad" and "not bad". Which is where the distinction comes in. Let's say we need our feature completed on or by January 19 (again, referring to Figure 6.1). In that case, having the feature complete on January 18 is a not bad outcome, having it completing on January 17 is a not bad outcome, etc. If we were to place a green shaded area on the curve, we would shade all the dates to the *left* of our desired date. The way we would read that result off of the Histogram, then, is “we have a 50% chance of finishing 50 items on or before January 19”. Note this is the exact language of a forecast that we discussed in the last chapter, which is exactly what we are looking for in this example.

Similarly, we can come up with a forecast for any risk level that we are comfortable with. If we want only a 15% chance of being wrong (an 85% chance of being right) we would say we will be finished on or before January 25. A 95% chance of being right would see us finishing on or before January 28. Second verse same as the first.

Forecasts for Multiple Items

Both the “How Many?” question of chapter 4, and the “When?” question of this chapter belong to a general class of problems that we like to call “forecasts for multiple items”. The clue is the plural word “items”. Whenever you have a question phrased in that form, you know you are dealing with a forecast for multiple items, and you further know that using historical Throughput in a Monte Carlo Simulation should be your go-to tool to answer. To drive this point home, let’s explore some of these types of questions that you may come across in your line of work.

Sprint Planning

While there may be several activities to be performed in Sprint Planning, one of most fundamental ones is to answer the question “How many PBIs can we complete in this Sprint?” This is classic form of the “How Many?” forecast for multiple items, so we know we can use MCS with historical Throughput. We won’t go into any great detail here, but if you are interested in specifically how to do this with Scrum, please see the “Flow Metrics for Scrum Teams” book[1].

If you hate the endless conversations and debates associated with traditional “agile” estimation techniques in Sprint Planning (that turn out to be wrong anyway), then you are going to want to check out

MCS. Given the right tool, you can answer “How Many?” in less than a minute (yes, seriously) and get on to the more important aspects of your job like—I don’t know—doing the actual work.

Regulatory Compliance

If you operate in a highly (or even semi) regulated environment, then you know it is not uncommon for some government authority to randomly pass some regulation that you must make your product compliant with. Additionally, the government may give you a set date by which you must be compliant or face a (sometimes significant) fine.

The slightly good news here is that you’ve got two approaches to forecasting this situation. You could do analysis of the regulation, determine how many items you need to complete to be in compliance, and then run an MCS “When?” to determine your chances of meeting the date. Or you could run an MCS “How Many?” and forecast the number of items you can complete by the due date given your acceptable level or risk in being compliant on-time. Either option would give you very quick and cheap insight in terms of what actions to take (focus on the requirement only, add people to the team, work overtime, do nothing, etc.) based on your risk tolerance.

Release Planning

Release Planning is very similar to the compliance situation above. That is, are your releases on a regular cadence, like at the end of every month or every quarter? Or can you release on-demand as needed? If the former, you would run an MCS “How Many?” to approximate your “budget” of items for the release. If the latter, you would run an MCS “When?” to determine a reasonable release date. Again, either way, you have a powerful tool at your disposal to manage your risk throughout the release.

Continuous (Continual) Forecasting

You might think at this point that we have said all that there is to say about MCS. You’d be wrong.

One of the things that makes MCS so powerful is how quick it is to use. In the Sprint Planning example above, we mentioned how you can get your answer using MCS in literally seconds. The reason that is important is because most forecasting is not "set it and forget it." How many times have you spent endless hours or days or weeks coming up with a plan, only for that plan to be invalidated a week into the release by changing scope, or a changing business environment, or whatever (take a second to appreciate how pretty much EVERY single plan *in the world* was invalidated when the 2020 pandemic hit). The ugly truth is that scope with always change, business priorities will always change, governments will always come up with new regulations, people will always take unexpected leave—that is just the world we live in. Thankfully for us, we have a forecasting technique that will embrace that change. Let us explain.

In the book, "When Will It Be Done?"[2] we walk through a detailed example of how hurricanes are forecasted. Long story short, think about how many variables are needed as input to simulate how a hurricane will behave: air temperature, water temperature, wind speed, wind direction—to name a few. But also consider how those inputs are constantly changing. That is, any hurricane forecast that the weather service puts out is almost immediately invalid because several of the input variables have changed since the forecast was generated. Does that mean they don't bother with forecasting? No, what it means is that they at least consider updating their forecasts as they get new information (again, please see my book for a more detailed discussion of how the US weather service does this). The process of constantly updating forecasts as you get new information is what we refer to as Continuous or Continual Forecasting. It is this technique that has allowed hurricane forecasting (and all weather forecasting, really) to become so accurate over the years.

Now think about your own project and what new information you get on virtually a daily basis that may cause you to update your forecast. Hopefully every day—or nearly every day—your team is completing new work. By definition, every time you complete items, you get updated throughput data. Recall from the example data in Chapter 4 how each day the team's throughput changed for different reasons. Again, this is to be totally expected. And we can exploit this knowledge by (at least potentially) including that data in our new forecast. To repeat, please see the book "When Will It Be Done?" for more information on how to do this.

The final thing to say about Continual Forecasting is that we like to refer to it as your "get-out-of-jail-free-card". What we mean by that is if you are ever worried that you don't have the right data, or you don't have enough data, or maybe have some other concerns, but you are still required to give a forecast, you always have continuous forecasting to fall back on. I have never had a case in my professional forecasting career where I did not have to go back and update an initial forecast because something had changed. I'm not going to say that that can't happen, but it probably won't. More to the point, it shouldn't. As we have been saying all along, uncertainty is the name of the game, and as some of that uncertainty is reduced (or increased!) by new information, then that will almost always necessitate a corresponding new forecast.

Conclusion

The purpose of the last three chapters has been to give you a high level overview of Monte Carlo Simulation and how you might apply it in different scenarios. If you have been reading carefully then by now you should have dozens of questions around MCS and its uses. Let's face that problem head-on walk through some of the most common questions directly.

Chapter 7 - MCS FAQs

Note: For this chapter, whenever we say “Monte Carlo Simulation” or “MCS” we are referring specifically to the Monte Carlo methods outlined in this pocket guide.

What assumptions are we making when we us Monte Carlo Simulation?

The fundamental assumption that you are making with MCS is that the future that you are trying to predict roughly looks like the past that you have data for. As we’ve mentioned a few times already, this is usually a safe assumption to make. But there are obviously some exceptions, which we will explore in the following questions.

What is the form of input data that should I use?

For MCS, you should be using your raw Throughput data as detailed in Appendix A. We want to say that again to be clear. Use your **raw** Throughput data. Do not transform it to be “normal”. Do not fit it to some “well-known” probability distribution and then use that new distribution. Your process is speaking to you through the data it produces. To be most successful, you must listen to your process in the language it speaks—its own raw data.

How much data?

Less than you think. The rough rule of thumb is that in most cases you need a dataset of at least 10 data points, and you can usually be quite confident using a dataset of around 20 points. This is a powerful result as it means we can usually get started with forecasting immediately rather than waiting until you have the “perfect” dataset.

More data than 10-20 points may help, but only so long as when you add more data, you are not violating the fundamental assumption of MCS. In other words, any additional past data must still be reflective of your current process, team composition, tooling, and workflow policies. Which means there is a law of diminishing returns when it comes to using larger and larger datasets. That is, by continually adding more data, you will eventually include stale data, or data from a past system state that is no longer valid, or whatever. This means that you will constantly have to analyze your input data for future relevance. Which brings us to our next question.

What historical data?

Matching the right historical data to the future you are trying to predict can be tricky. For example, if you work in Europe, would you use August's data to forecast what you can do in September? Maybe not. Would you use December's data to forecast what you can do in January? Again, maybe not. So what do you do? For September, can we use July's data? For January, can we use November's data? That last question becomes a little trickier to answer if you live in the US. Answering this question is really where forecasting becomes more art than science. Nowhere in this book have we said that you get to stop thinking. Quite the contrary. You know your context. You know your team. By using the principles in this book you should be able to select reasonable historical data to project a completion data. And lest we forget, we always have our get-out-of-jail-free card.

What about outliers or bad data?

As we stated earlier, our very strong recommendation is for you to use your process data in its raw form, without "cleaning it up" by removing what may be considered bad data or outliers. We say this because most times bad data or outliers is really just our process shouting at us that it is hurting. In other words, that's where all the signal is. However, we have to be very careful here. If you truly know that you have bad data, then by all means exclude it. Some examples of bad data might be that we forgot to close the Jira ticket six months ago. Or last month our workflow was not defined properly so that data is invalid. Only in obvious cases like that would we recommend the removal of data. Other that that, you should trust what your process is telling you.

What if you have no data?

Chances are—if you are following the advice in this book—that you have more data than you need rather than too little (or no) data.

However, if you truly have no data, then you have a few options:

1. Start to collect data as soon as possible and use that as input (recall, you don't need many points to get going).
2. Find a recent project or release that is roughly similar to what you are doing now and use that data. Then, as you progress, replace the old data with new data, all the while continually forecasting.
3. Guess. If you truly have no data then any forecasting technique you employ will be equally bad. But, as always, refine your forecasts as you get more data using continuous forecasting (notice a pattern here?)

In most situations, the worst case scenario is that you have to wait a couple weeks to get representative data. If a couple of weeks is too long, then feel free to use one of the other methods above.

Do you need a stable system?

This question is so important that we've dedicated a whole chapter to it. Please see Chapter 9 for more information.

Don't items need to be of the same size or complexity?

No, but this continues to be one of the most persistent myths in software forecasting. The belief that all items must be the same size usually leads teams to spend unnecessary effort trying to artificially normalize work into similarly sized stories. This wastes a ton of time, introduces dependencies, and usually creates bottlenecks in how the work moves to done.

Monte Carlo Simulation thrives on natural variation. It's specifically designed to work with a range of work item sizes and complexities. By using actual completion data where size, effort, and complexity are all baked into the data, you allow the simulation to account for real-world variation. In fact, trying to make all stories the same size usually masks the variability you're supposed to model, potentially reducing forecast accuracy.

How do I scale MCS?

So far, we have implicitly assumed a single team using a single backlog with all of our MCS examples. However, we realize that most contexts are much more complicated than that. Maybe you have multiple teams, multiple shared backlogs, and multiple dependencies (or some permutation thereof). Is it possible to MCS in those scenarios? The good news is that the answer is generally yes. The bad news is that the model itself could quickly become impossibly complex. Usually the answer involves modeling a higher level workflow (feature, epic, initiative, etc.) and using data at those different levels to forecast. As you can imagine there many subtleties and moving parts when it comes to setting up a simulation like that, and as this is a pocket guide, we really don't have the space to get into that here. But a great place to start is "Actionable Agile Metrics for Predictability Volume II"[1]. The whole second half of that book is designed to help you understand the issues of scaled forecasting so you can get started on your own.

MCS assumes I run the project thousands of times, but in the real world it only gets run once?

Yes, your project runs only once. But does that mean that an understanding of what might happen in the future won't help?

Stated differently, MCS simulates thousands of possible outcomes in order to provide a range of what could happen based on the variation seen in your system (rather than only one possible outcome or answer). This helps provide a better understanding of the risk so that you can make better, real-time decisions. For example, if the simulation shows a 50% chance of delivering by September and an 85% chance by October, that tells you something crucial about your risk exposure and what you might want to do next, depending on your comfort level.

What if executives don't understand percentiles?

Then it's your job to explain it in terms that matter to them. Percentiles express risk tolerance. If you say, "We have an 85% chance of finishing this by October 15 or earlier," you're saying, "If we commit to that date, we'll meet it 17 out of 20 times."

Our goal is to drive out a conversation about action and tradeoffs. If I go back to those same stakeholders and say we now have a 50% chance of hitting October 15th I would guarantee they are going to ask me some hard questions about A) what happened and B) what we need to do next.

This framing helps executives understand the likelihood of success and the potential business impacts. Your job is not to dumb things down, but to make them relevant and actionable.

What if T-shirt sizes give me everything I need?

If you're consistently delivering on time, managing risk, and communicating uncertainty well then keep doing what works. But in most cases, T-shirt sizes are just proxies for effort or size, not our actual delivery. They're qualitative guesses that might help you prioritize but they're insufficient for forecasting. They are certainly insufficient to quantify uncertainty in order to assess risk.

How do we use MCS if we've never done this type of work before?

Forecasting without a data history is hard but not impossible. If your team has no precedent for this type of work, consider the following steps:

1. Borrow data from similar teams or projects, ideally within your organization or value stream
2. Reforecast as you start delivering even a few work items. Every data point adds clarity.
3. Use MCS for planning conversations rather than delivery dates in early stages.

Monte Carlo is a learning tool as much as a forecasting one. Its true power lies in helping you confront uncertainty and develop better questions even when you're doing something new.

Can you do MCS with Story Points?

Theoretically you could, but why would you when you have the techniques outlined in this book. Seriously, why?

What about scope changes?

In case you haven't noticed by now, the answer to almost all of these questions is "continual forecasting". Any time that any aspect of your reality changes, it is at least worth considering how that new information might be incorporated into your model and a new simulation run. If scope has changed, then there is little doubt that the quantified uncertainty of your forecast has changed, and there is little doubt that you may want reassess where you are with your understanding of risk. Forecasting is not a "set-it-and-forget-it" mentality.

What if items are not refined?

This question is very similar to the preceding scope one. As items get refined, then your understanding of scope probably changes. And as scope changes, your understanding of uncertainty and of risk changes. In this situation, you can use continual forecasting to determine what the impact of refinement is and/or if scope needs to be adjusted accordingly.

Why do we include weekends and holidays in our data and in our forecast?

There is no hard and fast rule that weekends and holidays should be included with the input data of your forecast—but it is usually recommended. Again, this gets back to our fundamental assumption of MCS. There were weekends in the past, and we expect there to be weekends in the future. Holidays are a little trickier. Some holidays are more disruptive than others. Christmas in North America is much more impactful than Arbour day (more's the pity). Therefore, as was suggested earlier, if your past data includes holidays that are not representative of the future, then you may want to consider how you adjust your input.

Why not use Cycle Time for MCS?

Think for a second about how a Cycle Time (CT) MCS might work. Let's say you have a backlog of 20 items and you want to know how long to finish all 20 items using CT only. To this, you might select the first item and then randomly select a historical CT to assign to that item. You would add the assigned CT of that item to the current date to come up with a finished date. You would track the date that it finished, select the next item and have it start either on that finished date or on the next day. You would do the same thing for the next item, and so on. When the last item of the simulation finishes you would record that end date, and then start the whole thing over. After several runs, you could build a histogram of all the dates you recorded and then calculate probabilities for each, just like we did in Chapter 6.

Sounds great, right? Maybe not. Did you spot the problem with the above? In that approach what you actually calculated was the sequential completion of 20 single items. But think about how you work? Do you only work on one item at a time? That is, do you always wait for one item to finish before you start another one? Easy, you say, we'll just model an MCS with WIP limits. Yes technically you could do that, but what about pull policies in that simulation? Are you using a FIFO queuing strategy, FIFS, random, something else? What about blocker policies? What about violating WIP limits? While technically you could build an MCS that accounts for all of these things, you can see how the model would become wickedly complex very quickly. Chances are you would get it wrong. But why even bother when the Throughput approach is so much simpler and more accurate?

Should I Fit My Data To A Probability Model?

This topic is a bit too geeky to be included in the regular FAQs, but it is a topic that we regularly come across so feel it must be addressed somewhere. This is as good a place as any.
So-called "experts" often feel the need to try and fit a given dataset—any dataset, really—to a well-known probability curve (e.g., Weibull, Log-Normal, etc.). From the discussion so far you should already realize the ridiculousness of that line of reasoning. Even so, as I never miss a good opportunity for "expert" trouncing, I'll say a few words more about curve-fitting.
The idea that conclusions can be drawn based on how well your data fit a given probability model is known as the Quetelet Fallacy[5]. This fallacy was first exposed by Francis Galton in 1875 and has been well-understood by statisticians ever since. Yet we still have people who preach the need to take your data and fit it to a curve. As Dr. Wheeler says, "Unfortunately, each generation of students of statistics has some individuals who follow in the footsteps of Quetelet. Some of them even write articles [or tweets or blogs]

about their profoundly erroneous insights.” Do a Google search on something like “kanban Weibull shape parameters” and you will see what I mean (Weibull as in Wei-bullshit). People actually publish this stuff with a straight face.

I’ve said it before, and I’ll say it again (paraphrasing Wheeler), your data was not produced by a probability model. Your data was produced by a process. If you assume a specific probability model, you are making erroneous assumptions: “We will *never* have enough data to uniquely identify a specific probability model. Probability models are limiting functions for infinite sequences, and therefore, they can never be said to apply to any finite portion of that sequence. This is why any assumption of a probability model is just that—an unverifiable assumption.”

An even bigger flawed assumption, however, is that—just like with the global standard deviation statistic—all probability models assume you are working with homogeneous data. However because your data is generated by a process, you are very likely to have signals in your dataset. By definition, data that contain signals are non-homogeneous. Therefore, it is time to throw out all of your old statistics textbooks and ignore all of those uninformed blogs. Instead, simply use your raw process data for forecasting. Its what your process would have wanted.

Conclusion

So much of forecasting is not the application of a rigid formula; rather, it is the application of common sense. You now have a good sense of the assumptions that go into a good MCS. You also have a good understanding of your context. Use those two things together to help you get the predictability you want.

But what about the case when I only want to forecast a single item? We’ve been waiting for you to ask that.

Chapter 8 - What About Forecasts for Individual Items?

Up until now we've have been focused on forecasting the completion of multiple items. But if you think about it, a forecast for multiple items is simply a bunch of individual forecasts for single items. Wouldn't it make sense, then, that if we could get really good at forecasting how long it takes us to get individual items done, that our forecasts for multiple items would improve as well?

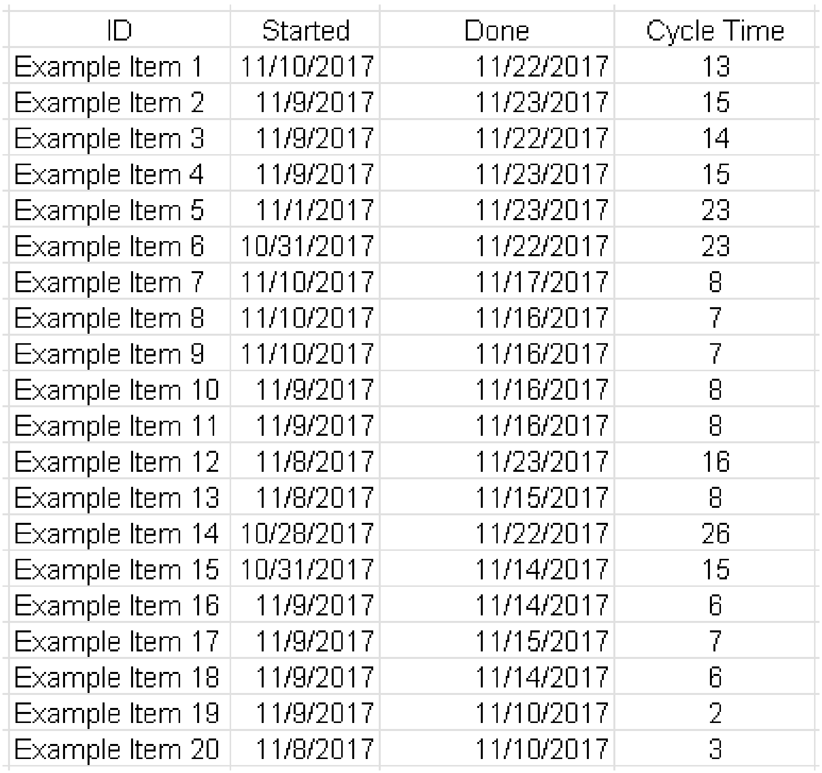
Let's say we are about to start work on an individual item—that is, it could be a single user story or it could be a a single production defect, etc.—but before we start working, a stakeholder wants to know how long it will take for that particular item to finish. Notice, here, that we are faced with all of the same problems with predicting the future that we encountered with forecasts for multiple items. Specifically, we know that the future is full of uncertainty so it is impossible to give a 100% certain answer to the "When Will It Be Done?" question.

Luckily for us, we have two things in our favour:

1. We know how to handle uncertainty.
2. We have the data to quantify that uncertainty (and, ultimately, risk).

Let's start with #2 first. From Appendix A, we know how to set up our process for measurement, and we know what flow data to collect. Because we want to know how long it takes individual items to complete, we are going to use our flow data to calculate the flow metric of Cycle Time.

However, like Throughput, our historical data will exhibit some notion of variation. For example, see the Cycle Time data in Figure 8.1:



ID	Started	Done	Cycle Time
Example Item 1	11/10/2017	11/22/2017	13
Example Item 2	11/9/2017	11/23/2017	15
Example Item 3	11/9/2017	11/22/2017	14
Example Item 4	11/9/2017	11/23/2017	15
Example Item 5	11/1/2017	11/23/2017	23
Example Item 6	10/31/2017	11/22/2017	23
Example Item 7	11/10/2017	11/17/2017	8
Example Item 8	11/10/2017	11/16/2017	7
Example Item 9	11/10/2017	11/16/2017	7
Example Item 10	11/9/2017	11/16/2017	8
Example Item 11	11/9/2017	11/16/2017	8
Example Item 12	11/8/2017	11/23/2017	16
Example Item 13	11/8/2017	11/15/2017	8
Example Item 14	10/28/2017	11/22/2017	26
Example Item 15	10/31/2017	11/14/2017	15
Example Item 16	11/9/2017	11/14/2017	6
Example Item 17	11/9/2017	11/15/2017	7
Example Item 18	11/9/2017	11/14/2017	6
Example Item 19	11/9/2017	11/10/2017	2
Example Item 20	11/8/2017	11/10/2017	3

Figure 8.1: Cycle Time Data

A quick inspection of Figure 8.1 tells us that items in our process have historically finished in as little as 2 days and as many as 26 days with various durations in between. At this point, we should ask, what might some of the causes of this variation be? Maybe some of those items had dependencies, maybe some of them were interrupted, maybe sometimes the people working on them got sick, maybe some items were "harder" than others, and so on. Whatever the reasons, the first question we need to ask is are those causes likely to continue in the future. In most cases, just like with MCS, the answer to that question is usually "yes".

So then the next question would be, how might we go about better understanding that variation? Hopefully by now we've rid you of the habit of wanting to average these numbers (the average is 11.5 days, by the way). Rather, one of the best ways to quantify uncertainty is to first load your Cycle Time data into a Scatterplot, as shown in Figure 8.2

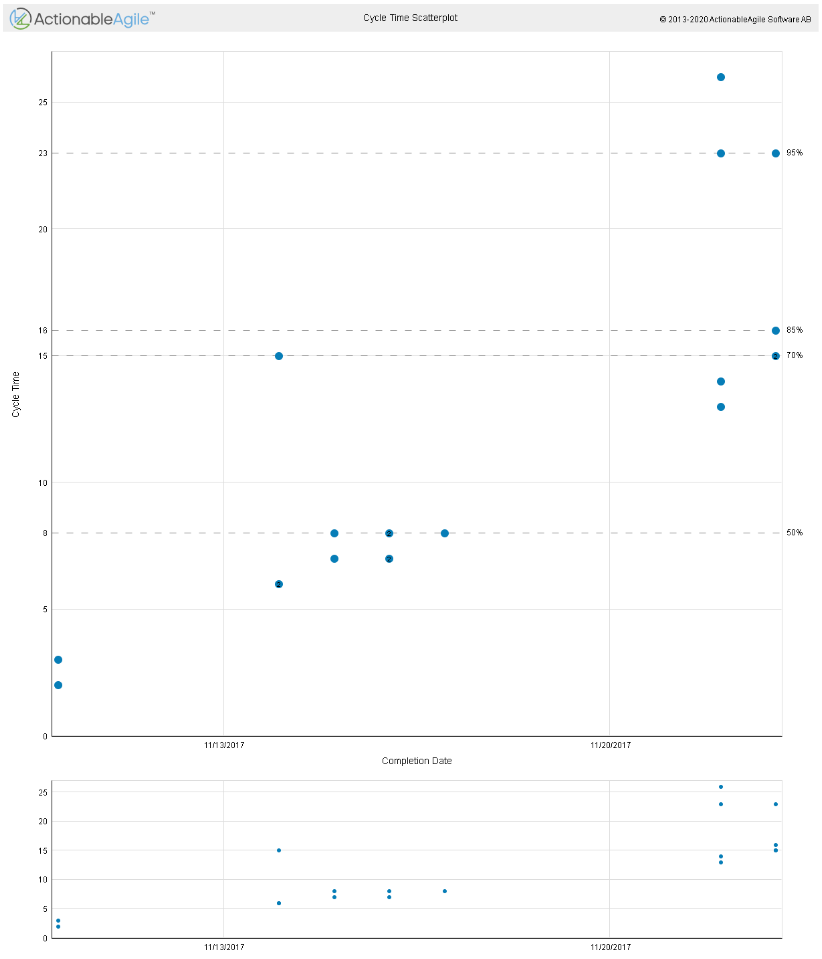


Figure 8.2: Cycle Time Data

Again, at the risk of shirking my duty, I will tell you that what Cycle Time Scatterplots are, how to construct them, and how to analyze them are handled ad nauseam in my book "Actionable Agile Metrics for Predictability"[1], so I won't be going into that detail here.

What we will say, however, is that, once constructed, we can segment our Cycle Time data on the Scatterplot using percentile lines in very much the same way that we did on the MCS Results Histogram.

These percentile lines act to categorize the different levels of uncertainty in our data. Allow me to explain by way of Figure 8.2. There we would say that of all the items that we have data for—and that have *completed*—we know that 50% of those items took 8 days or less to finish, 85% of those items took 16 days or less to finish, and 95% of items took 23 days or less to finish.

Notice that each of those above statements are in the form of a forecast, and, therefore each one would be a candidate for a potential forecast. Just like with MCS, all we need to know in order to produce a forecast is an understanding of what our acceptable level of risk is. If we want to be right only 50% of the time, then we know that any item in the past that finished at or below the 50th percentile line was a "not bad" outcome and any item that finished above the 50th percentile is a "bad" outcome. Thus, for a 50% risk tolerance, our forecast for the next item would be, "We have a 50% chance of of completing the next item that starts in 8 days or less". For a 95% risk tolerance, our forecast would be, "We have a 95% chance of finishing the next item that starts in 23 days or less" [Note that if we want a higher "confidence" we will give a wider range—this should make intuitive sense.]

Those of you who are familiar with Kanban will recognize what I just described as being the same procedure used to come up with a Service Level Expectation (SLE). SLEs work as de facto forecasts in Kanban to help us spot problems with flow. They expose the probabilistic nature of risk management so that a team or organization can take action when appropriate. For more information on how to manage flow using SLEs, please see "The Kanban Pocket Guide."[2]

Aging

So if we know that the best way to impact our multi-item forecast is by managing our individual item forecasts, the next thing we need to see is how old the active work in progress is in relation to the SLE that we have set. Back to your four flow metrics- this is called "Work Item Age" or the the amount of elapse time between when the work item started and today. The magic of this metric is that its telling you that you have a problem with your flow while you still have time to do something about it- but this is only true if you can see it.

Most of the tools out there don't make Work Item Age easy to see, interpret, or even act on so you may need to get creative with how you color, filter, or label the tickets to create the signals that you need to take action on work that has aged past an appropriate amount of time. Like so many things in this guide, what is appropriate or "bad" or "not bad" is really up to your context and should be based on your actual data. The key is to make it actionable which means you will also need to identify the right time and cadence (perhaps a daily sync?) to review this data. If your only looking at your work item age it may be too late.

As with SLEs, if you would like a deeper discussion on how to use Aging to help with predictability, please see "The Kanban Pocket Guide."

Conclusion

We want to reiterate what we stated at the outset as my motivation for including this chapter here. If your goal is to get better multiple items forecasts, then want of the best places to start is to effectively manage the individual items that are flowing through your system. That will require the use of other flow metrics (e.g., Cycle Time and Aging) and other charts (e.g., Scatterplots and Aging Charts). You get those things right and now you've got a shot at better forecasts. Not a guarantee, but a shot.

Chapter 9 - System Stability

We debated whether to make this chapter really short, or really long, as the question of system stability seems to be one that causes the most confusion when applying the forecasting techniques that we have discussed so far. When you get to the end, tell us if the chapter should have been longer or shorter.

Stability Defined

The first problem of stability comes in its definition. Most people think that a stable process is one that doesn't change. That is, you can't have a stable process when team members are added or subtracted, or when requirements are allowed to change change, or where you have items of different sizes—to name a few. In reality, however, a process could exhibit all of those changes (and more!) and still be stable. All of these reasons are why we must take a more rigourous approach to stability, rather than just relying on some random shill. Instead, we will rely on established experts for our definition: Dr. John Little, and Dr. Walter Shewhart.

Little's Law

In 2008, Dr. Little (along with Dr. Stephen Graves) contributed a chapter to a book entitled, "Building Intuition: Insights From Basic Operations Management Models and Principles."[1] In that chapter, Little gives a bit of history of his eponymous law, and clearly defines the conditions needed for the law to hold. We will not go into any of that detail here; rather, we will jump to the section where Little discusses a form of his law as referenced in a textbook by Hopp and Spearman[2]. In that book, Hopp and Spearman state Little's Law as:

TH = WIP / CT (1)

"where they define throughput (TH) as 'the average output of a production process (machine, workstation, line, plant) per unit time,' work in process (WIP) as 'the [average] inventory between the start and end points of a product routing,' and cycle time (CT) as 'the average time from release of a job at the beginning of the routing until it reaches an inventory point at the end of the routing (that is, the time the part spends as WIP).' "[3] As this form of the law is distinct from the form Little first published in 1961[4], Little gives a lengthy discussion about what conditions are necessary for this new form to be valid. [For clarity, we will only address here the situation where you are operating a process where WIP never goes to zero. For a more detailed discussion of how to handle situations where WIP in your system drops to zero from time to time, please see the references included at the end of this book.]

Specifically, Little says that for a system to be stable we need two conditions: "First, we need the size of the WIP to be roughly the same at the beginning and end of the time interval so that there is neither significant growth nor decline in the size of the WIP. Second, we need some assurance that the average age or latency of the WIP is neither growing nor declining."[5]

It doesn't much clearer than that. If you are running a system where WIP fluctuates wildly and/or the average age of items varies wildly, then you don't have a stable system.

Process Behaviour Charts

Another way of looking at stability is through the lens of Dr. Walter Shewhart. In the 1920's, Shewhart developed a method of data analysis that was later called Statistical Process Control (SPC). In particular, he developed a chart for process data that is now called a Process Behaviour Chart (PBC)[6].

As with Little's Law, we are not going to go into all of the sordid details surrounding what a PBC is, how to construct one, or how to interpret one. For that discussion, we would once again point you to the references at the end of this book. For clarity, however, we will say that the discussion here will be centered around a specific type of PBC known as an XmR chart.

The benefit of an XmR chart is that, assuming you have collected your data properly, and assuming you have constructed the chart properly, you can see at an instant if your process data is stable. Take a look at Figure 9.1 below.

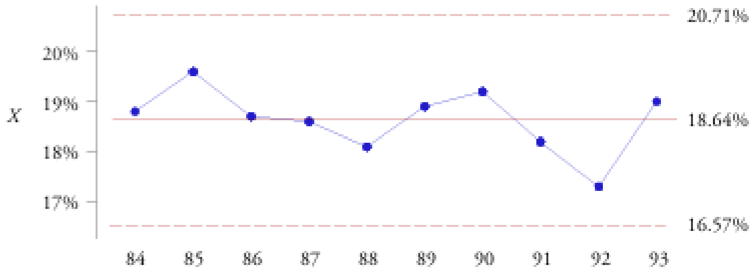


Figure 9.1: Example X-Chart

Figure 9.1 is an example of a properly constructed X-chart. On this chart you will see two dashed lines that "frame" the data. The top line is what is known as an Upper Control Limit (UCL) and the bottom line is what is known as a Lower Control Limit (LCL). If all of your data falls between these two lines then your process is considered stable. Dr. Wheeler would even go a step further and say that your process is predictable[7].

Stability and MCS

Does this mean you have to satisfy both Little's and Shewhart's definitions of stability in order to use Monte Carlo Simulation? Well, yes and no.

Let's start with the no part first. Mathematically speaking, even if your process is unstable by either (or both) definitions, there is nothing stopping you from throwing that data into an MCS to come up with a forecast. The question then becomes how much can you trust that result? If your underlying process is inherently unstable, then we certainly wouldn't lend much credence to any answer you get. That is, your forecast could be much better or much worse than reality, but you simply wouldn't know. In that case, we believe your time would be better spent in "fixing" your process before investing too much time in forecasting. However, there would be nothing stopping you from using MCS as a "back-of-the-envelope" calculation to see where you are at—we just wouldn't make any contract plans based on that.

If, however, your underlying process is stable by both definitions, then our recommendation is easy. Go forth and multiply. Remember, though, that MCS is only an approximation, and continual diligence (and forecasting) is recommended.

What about the case where your process data satisfies one definition but not the other? That's a trickier one. We certainly would want to go to the source and understand why that is happening. But in the meantime, could you use MCS? That would be totally up to you and your understanding of your context. How confident are you in how your process is performing? Do you know of any obvious reasons why your process might be unstable? Or do you think the MCS forecast you are producing is a reasonable reflection of what your process is capable of? Those are questions we can't answer for you. But we can say it would be next to impossible for us to argue with you one way or another depending on which way you decide to go.

Stability and Single Item Forecasts

For completeness, everything we just mentioned about stability for MCS is equally true of Cycle Time and single item forecasts. To gauge whether your SLE is a meaningful forecast, you'll first want to understand if your Cycle Time data is stable. And the process to do that is the same: (1) are you violating the assumptions of Little's Law, and/or (2) does an XmR chart of your Cycle Time show any signal of process changes? If either of those are true, then, as before, you are better off fixing those problems first before worrying about any forecast.

Conclusion

Not commonly well-defined, and even less commonly understood, stability is one of those concepts that either gets completely ignored or blatantly overemphasized when it comes to the topic of forecasting. That is why we believe it is much more important to come up with a definition of stability that meets at least three requirements:

1. Is it logically consistent?
2. Is it mathematically rigourous?
3. Is it experimentally verified? (if needed)

The approaches that we have outlined here certainly meet these criteria. Now ask yourself, can the same be said of the definition provided by that random pundit on LinkedIn? If not, then those definitions are opinions at best. And you know what they say about opinions.

Having exhausted all of the definitions required to proceed, it is time to pull this all together so that you can get started with forecasting.

Chapter 10 - Putting It All Together

Returning to the scenario that we set out in Chapter 1, let's re-run each of those forecasts using a Monte Carlo Simulation instead of Linear Projection and see what we would have come up with.

On November 1st when the team had 778 (1071 - 293) work items left in their backlog, a Monte Carlo Simulation (10,000 runs) that used a historical Throughput data range of October 1 – October 31 would have yielded a Results Histogram that looked like:

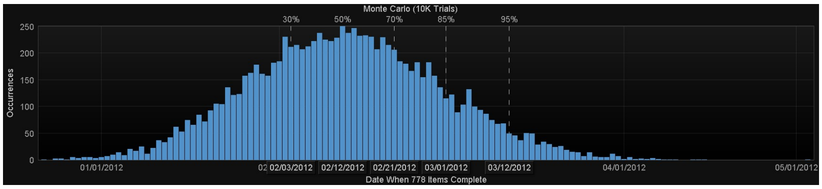


Figure 10.1

The Results Histogram in Figure 10.1 shows that at the time this simulation was run they had an 85% chance of completing on or before the March 1st date. Which is actually not bad. However, given the extreme risk associated with this project, a more conservative projection is probably in order. You can see that the 95% confidence in Figure 10.1 shows a date of on or before March 12. And you will recall that this does not include the inevitable upcoming holiday slowdown. That March 12 date could easily be a week or more later than that. If you were a team member on this project would you have taken action based on this new information? It would certainly have been worth a conversation…

Moving on to December 1st, an MCS that projected forward with updated Throughput numbers and 490 items left in the backlog would have given a Results Histogram that looks like:

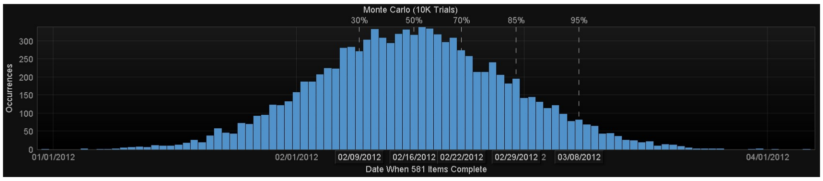


Figure 10.2

Figure 10.2 shows the 95th percentile has not budged from its March 8th prediction. That is good news from the perspective that the Thanksgiving holiday did not slow the team down too much. However, this is still nothing to cheer about as they are still worryingly showing a late date, and they still have the Christmas holidays slowdown to deal with. Ask yourself if now you would have done something?

On January 3rd, the team would have come through its holiday productivity dip. What do you think the MCS would show now? Here it is:

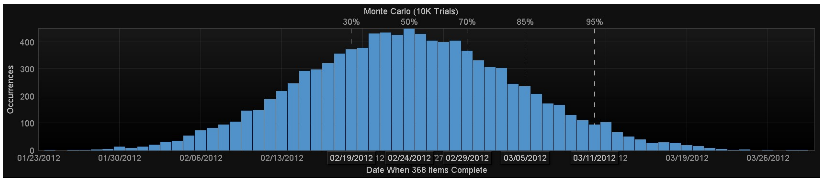


Figure 14.3

Figure 10.3 now shows the 95th percentile at March 11. While they did not take a dramatic hit to the date, they still did take a hit. And, worse, they are still showing late. What would you do now?

The February 1 simulation is much the same story:

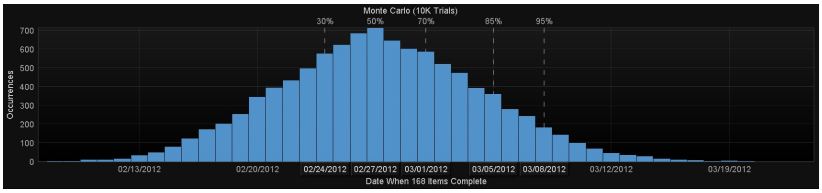


Figure 14.4

You can see the team recovered slightly from their holiday hangover, but they are still showing late at the 95th percentile. Now would it be reasonable to do something?

Let's take a look at those projections side by side:

Date Forecast Made	Linear Projection	MCS Projection (95th Percentile)
Nov 1	1/23/2012	3/8/2012
Dec 1	2/12/2012	3/8/2012
Jan 3	2/22/2012	3/11/2012
Feb 1	2/24/2012	3/8/212

Actual Project Delivery Date: March 8, 2012

There are several things that strike us about these results. The first is how accurate MCS was even at a very early date with very little data. As we have been saying all along, this is without a doubt one of the great strengths of MCS. The second is how consistent the forecasts were even in the face of new Throughput data. Over the course of the project, the team's Throughput was no doubt extremely variable—especially given the multiple holidays. However, even when the MCS model was updated with this updated data, the forecasts did not budge all that much. I have to be careful here because what I am not saying is that these results are typical or even common. But what I am saying is that the power of MCS is its ability to give accurate forecasts even in the face of variability.

Again I want you to ask yourself, had you been using either of these approaches, at what point would you have taken action? Would you have even taken action? My guess is that had you been using the linear projection model you probably would not have made any adjustments to your project schedule. In fact, you might not have even thought to do something until it was too late.

This is the beauty of the MCS approach. You can see in this example that MCS gave you a signal that something was potentially wrong as early as November 1st. As any manager (or customer) will tell you, the earlier you know that a change is needed, the better the chance of that change having an impact. That's just Risk Management 101. For example, some interventions that we talked about to affect the outcome of a forecast are:

1. Change scope
2. Change the date
3. Add people / work longer hours
4. Accept a lower chance of success
5. Some combination of the above

As this was a fixed contract, it eliminated options 1, 2, and 4 above. The only option available to us in this scenario was option 3. However, for option 3 you need enough time for the intervention to take effect. Using the linear projection forecast, we would not have known until very late in February that the project was going to be late. Trying to add resources at that late a date brings Brooks' law into play, "adding people to a late project only makes it later"[1]. Besides, even if the new team members were 100% productive on Day 1, would they have had enough time to make a difference? The same argument goes for working longer hours. There are only so many extra hours available to work which means you need to know well in advance for that strategy to even work (not to mention the realistically muted impact that overtime has on delivery rates anyway).

What This Means For You

The unfortunate reality of giving a first forecast is that once that date is communicated it's usually set in stone despite all of our better efforts to caveat all the things we've talked about here. In our experience what starts as MCS "When", shifts to what is the changing likelihood or risk of hitting that date as we work. Continuously forecasting then becomes a conversation about what actions we want to take to increase the likelihood or decrease the risk of staying on track. When we are constantly changing dates, trust quickly erodes and we take the opportunity to respond and improve the system out of the conversation.

Our recommendation is that when you see the uncertainty increasing or decreasing that that is your chance to talk about the underlying factors contributing to the change.

Questions to consider:

- What's changing in our system as risk increases or decreases?
- Are we taking on too much at once?

- Has the scope quietly expanded?
- Are quality issues causing hidden rework?
- Is capacity shifting due to competing priorities or unplanned work?

Each of these is a risk you can respond to but only if you can see it early enough to act.

Consider this your invitation to get started. You don't need perfect data. You need curiosity, a spreadsheet, and a willingness to learn from what's there.

Begin by exploring the variability in your current system. As we shared, a simple XmR chart is an excellent place to start. Are your fluctuations within expected limits, or are there signs of instability? This kind of visibility will help you understand whether your system is predictable enough to trust your forecasts.

Then, run your Monte Carlo Simulation with your historical throughput data. Don't worry if you only have a few weeks of numbers or if the data isn't stable. Run the simulation anyway. The goal isn't precision, it's perspective. What range of outcomes is possible based on how your team has delivered in the past? What's your current likelihood of hitting the date you've promised? And if that likelihood changes, what should you do next?

As you get going, focus less on getting the "right answer" and more on building your team's ability to see risk and respond to it.

This is where real system improvement starts not in getting the forecast exactly right, but in creating the conditions where delivery conversations are based on reality, not optimism.

Where To Learn More

This is Dan speaking here, and it might seem like an odd thing to admit, but when I get a book, one of my favourite things to read is the bibliography. In fact, I'll often start there as I usually learn more from that one section than I do from the book itself. As such, the bibliography that we have included here is a carefully curated list of some of the books/blogs/articles that have influenced us most when coming up with our philosophy of forecasting. We'll do our best to keep this updated as we uncover more and more references (we are always learning!), but, no matter what, your fallback position should be to come visit us at https://prokanban.org to see all of our latest thinking.

Good luck!

Appendices

Appendix A - What Data To Collect

Calculating flow metrics will first depend on collecting the right data[1]. And collecting the right data will depend on modeling your flow. How to model flow is covered in "The Kanban Pocket Guide" (KPG), so we won't go into that detail here. However, you will note from KPG that a minimally modeled flow has at least one start point and at least one finish points. Before we can proceed, you will need to have those defined for your process.

What Data To Collect

Once you have the start and finish points of your process defined, then the only data you need to collect to calculate flow metrics is a timestamp for when an item crosses your start point and a timestamp for when an item crosses your finish point. That's it. All of the metrics that I will discuss in this book need only those data points. In the interest of clarity, there may be other data you want and/or need to collect for your context, but for the basic metrics of flow, those two data points are all you need.

For example, in your context, your timestamp tracking might look like this:

Work Item ID	Arrived	Departed
1	01/01/2016	01/03/2016
2	02/02/2016	03/03/2016
3	01/02/2016	03/04/2016

Figure A.1 – Sample Timestamp Data

[It is impossible to tell from these data in Figure A.1, but I'm using American-style dates. Not only are American dates represented in Figure A.1, but I will use that style for the rest of the book (unless otherwise noted). It doesn't necessarily matter here, but it might matter later.]

If you are using a tool (like Jira) to help track work, then when it comes to data, there is again good and bad news. The good news is that most Agile tooling tracks timestamps as required above. The bad news is that those tools usually do not make this data easily accessible. You might be able to query a database or code against some API, but almost certainly there will be some nontrivial amount of work required to get the data you need. Thankfully, there are plugins out there to help. The one that I recommend is ActionableAgile® Analytics. But as I said before, I am biased.

Flow Metrics Calculation

Now that you have the right data, it is time to calculate the right metrics. Rigourous definitions for each of basic Flow Metrics can be found in the KPG, so, again, will skip that discussion here. But to perform any forecasting, you will need to know how to calculate the flow metrics from your data, which is what we will discuss now.

Cycle Time

Cycle Time is the measure of how long it takes and individual item to flow through our process from started to finished. Mathematically, Cycle Time equals the finished date minus the started date plus one (CT = FD - SD + 1).

If you are wondering where the "+ 1" comes from in the calculation, it is because we count every day in which the item is worked as part of the total. For example, when a PBI starts and finishes on the same day, we would never say that it took zero time to complete. So we add one, effectively rounding the partial day up to a full day. What about items that don't start and finish on the same day? For example, let's say an item starts on January 1st and finishes on January 2nd. The above Cycle Time definition would give an answer of two days (2 − 1 + 1 = 2). We think this is a reasonable, realistic outcome. Again, from the customers' perspective, if we communicate a Cycle Time of one day, then they could have a realistic expectation that they will receive their item on the same day. If we tell them two days, they have a realistic expectation that they will receive their item on the next day, etc.

You might be concerned that the above Cycle Time calculation is biased toward measuring Cycle Time in terms of days. In reality, you can substitute whatever notion of "time" that is relevant for your context (that is why up until now we have kept saying track a "timestamp" and not a "date"). Maybe weeks are more relevant for your specific situation. Or hours. Or even Sprints. If you are on a Scrum team and you wanted to measure Cycle Time in terms of Sprints, then the calculation would just be Finished Sprint − Start Sprint + 1 (assuming work items cross Sprint boundaries in your context). The point here is that this calculation applies in all contexts. However, as with WIP, my very strong recommendation is to calculate Cycle Time in terms of days. The reasons are too numerous to get into here, so when starting out, calculate Cycle Time in terms of days and then experiment with other time units later should you feel you need them.

Throughput

Throughput is the measure of how many items we finish per unit of time.

To make our Throughput calculation example a bit clearer, let's look at a different set of data than what we viewed previously with Cycle Time:

Work Item Id	Arrived	Departed
1	01/01/2022	03/01/2022
2	01/02/2022	03/03/2022
3	02/02/2022	03/03/2022
4	01/02/2022	03/04/2022
5	03/02/2022	03/04/2022

Figure A.2 - Sample Process Data

To calculate Throughput, begin by noting the earliest date that any item was completed, and the latest date that any item was completed. Then enumerate those dates. In this example, those dates in sequence are:

Completed Date
03/01/2022
03/02/2022
03/03/2022
03/04/2022

Figure A.3 - Consecutive Calendar Days Between First and Last Finished Items

Now for each enumerated date, simply count the number of items that finished on that exact date. For this data, those counts look like this:

Completed Date	Throughput
03/01/2022	1
03/02/2022	0
03/03/2022	2
03/04/2022	2

Figure A.4 - Calculated Throughput

From Figure 3.3 we can see that we had a Throughput of 1 item on 03/01/2016, 0 items the next day, 2 items the third day, and 2 items the last day. Note the Throughput of zero on 03/02/2016—nothing finished that day.

As stated above, you can choose whatever time units you want to calculate Throughput. I advise very strongly that you measure Throughput in terms of days. Again, it would be a book in itself to explain why, but let's consider two quick justifications: (1) using days will provide you much better flexibility and granularity when we start doing things like Monte Carlo simulation for planning activities (explained in detail in my second book "When Will It Be Done?"); and, (2) using consistent units across all of your metrics will save you a lot of headaches. So if you are tracking WIP, Cycle Time, and Age all in days, then you will make your life a whole lot simpler if you track Throughput in days too (not to mention we now know this is a requirement of Little's Law).

You still with me? I hope so because we've saved the most difficult parts for last. These next couple of topics might seem out of place here, but, trust me, you must understand these concepts before we can go any further with metrics and data.

Work Item Age

Work Item Age is a measure of how long an item has been in progress (without finishing). Mathematically, Work Item Age equals the current date minus the started date plus one (Age = CD - SD + 1).

The “plus one” argument is the same as for Cycle Time above. You will never have a work item that has an Age of zero days. Again, start by tracking Age in days. All the reasons why are the same as what I have outlined for CT above.

WIP

Work In Progress is the measure of how many items have entered our system but have not left. [The calculation of Work in Progress (WIP) is not directly needed for forecasting, but we’ve included it here nonetheless for completeness. Not to mention, understanding WIP will come in handy for your process improvement efforts in pursuit of better forecasts. Therefore, it is not wasted time on your part to understand what WIP is and how to calculate it.]

Mathematically, WIP is the count of all work items that have a started timestamp but do not have a finished timestamp for a given interval of time. That last part is a bit difficult for people to grasp. Although technically WIP is an instantaneous metric—that is, at any time you could count all of the work items in your process to calculate WIP—it is usually more helpful to talk about WIP over some time unit: days, weeks, sprints, etc. Our strong recommendation—and this is going to be our strong recommendation for all of these metrics—is that you track WIP per day. Thus, if we wanted to know what our WIP was for a given day, we would just count all the work items that had started but not finished by that date. For Figure A.1, our WIP on January 5th is 3 (work items 3, 4, and 5 have all started before January 5th but have not finished by that day).

Conclusion

The power of the approach outlined in this book is that you need a minimal set of data from which to calculate a minimal set of metrics.

We've set the bar low so as to make it as easy as possible for you to jump over and get started.

Appendix B - The Mechanics of Monte Carlo Simulation

Monte Carlo Simulations (MCS) vary, but in general, they tend to follow a typical pattern:

1. Define a dataset to use as input
2. Randomly select values from the input dataset and perform a computation on the selected inputs
3. Aggregate the results (usually by employing a Histogram)
4. Repeat steps 2-3 an arbitrary number of times until you have a clear picture of what the result set looks like.

To illustrate, imagine an MCS to compute the probability of getting a 7 when you roll two, fair six-sided dice. First, we define our possible inputs—the sum of the numbers of two six-sided dice. Second, we roll the dice. Third, we sum the numbers of each roll and track the result in a histogram (what we will refer to as our Results Histogram). Fourth, we re-run the experiment repeatedly and continue to aggregate each outcome in the Results Histogram. When finished, after 100 rolls, we might get a Results Histogram that looks like the middle graph of Figure B.1:

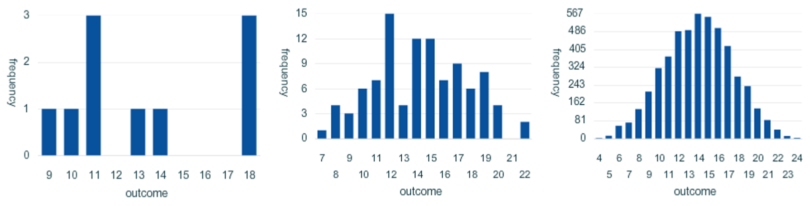


Figure B.1: Rolling Dice Histogram (10 trials, 100 trials, and 5000 trials, respectively)

Once we have this Histogram, we can calculate the probabilities of each outcome just like we did in Chapter 4.

That's all very well for rolling dice, but how might this work in our world? That is, how we might apply this method to forecast the how many items we can complete by a given date for a given team?

Step 1: Define a dataset to use as input. Your candidate dataset is going to be your historical Throughput data that I just showed you how to calculate. However, as was the case when making forecasts for single items, the first thing you need to decide is which past data to include for the simulation. I mentioned before that the dirty little secret of all of the probabilistic forecasting methods is that the future that you predict is going to roughly look like the past that you have data for. Monte Carlo Simulation is no different. That means in order to make meaningful multiple item forecasts, you are going to have to choose a relevant set of historical data that you think will roughly mirror the future that you are trying to predict. Remember that this practice is more art than science and that you can refer to Chapter 11 for more information. Again, you are probably going to want to use a tool that allows you to easily select different sets of past data and to immediately show you the impact of those different selections.

Step 2: Randomly select values from the input distribution and perform a computation on the selected inputs. This is where the MCS rubber meets the road. To project out how long it will take 100 items to finish, we are going "simulate" how many items get done for each consecutive day into the future. The way that works is, say, for example, we are starting our simulation on January 1. For January, 1, then, we are going to select a random Throughput number from our historical (input) Throughput data. Let's say we randomly chose a 3. For the purposes of the simulation, that means that our process got 3 things done on January 1. We then subtract 3 from 100 to get 97 items left to complete. We then repeat the same calculation for the next day, January 2. Let's say we randomly choose a 5 for January 2. That means we got 5 things done this day so we subtract 5 from 97 to get 92. We do this over and over until we either get a 0 (or less than zero whichever comes first). We then note the date that we got to zero—let's say for this example it is February 28.

Just one last thing to note about Step 2 before we move on. The algorithm that we use here where we simply randomly select a historical Throughput input is just one such option for our simulation.

There are many others. We will briefly touch on other options for selecting Throughputs in Chapter 12.

Step 3: Aggregate the results. For each "run" of the simulation, we are going to track the results in what we are going to call a Results Histogram. The Histogram is going to be built using dates across the bottom (the x-axis) and frequency up the side (the y-axis). For each date result we get from step #2, we will find that date on the x-axis and increment its frequency by 1 (so that the height of the histogram bar grows.

Step 4: Repeat steps 1-3 an arbitrary number of times until you have a clear picture of what the result set looks like. With each run of the simulation, the probability distribution as described by the Results Histogram will become clearer and clearer—exactly like what we saw with the rolling two dice example. You can use as many simulations as you like, but it has been my experience that you usually get a pretty good idea of what your distribution looks like after about 1000 runs, and the output has mostly stabilized by 10,000 runs. Once you have finished all of your runs you will get a results histogram that looks something like:

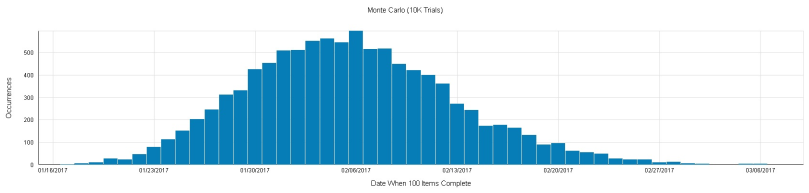


Figure B.2 - Example Results Histogram

Note that unlike your Cycle Time data, your Results Histogram will indeed approximate a Normal Distribution. This is due to something in probability called the Central Limit Theorem (CLT). You need not worry yourself with the specifics of the CLT, only know that in general the Results Histogram will usually resemble a bell curve.

Congratulations! The simulation is complete. The only thing left now is to discuss how to interpret the results.

Appendix C - Histograms

Disclaimer: this appendix is not meant to represent an exhaustive treatment of Histograms. For that I invite you to explore some of the books listed in the Bibliography at the end of this book.

What is a Histogram?

Simply stated, a Histogram is graphical display of data that uses bars of different heights to show the frequency of different data points within an overall dataset. A Histogram is very similar to a bar chart with one important distinction being that a Histogram groups population elements together in ranges. An example Histogram is shown in Figure C.1:

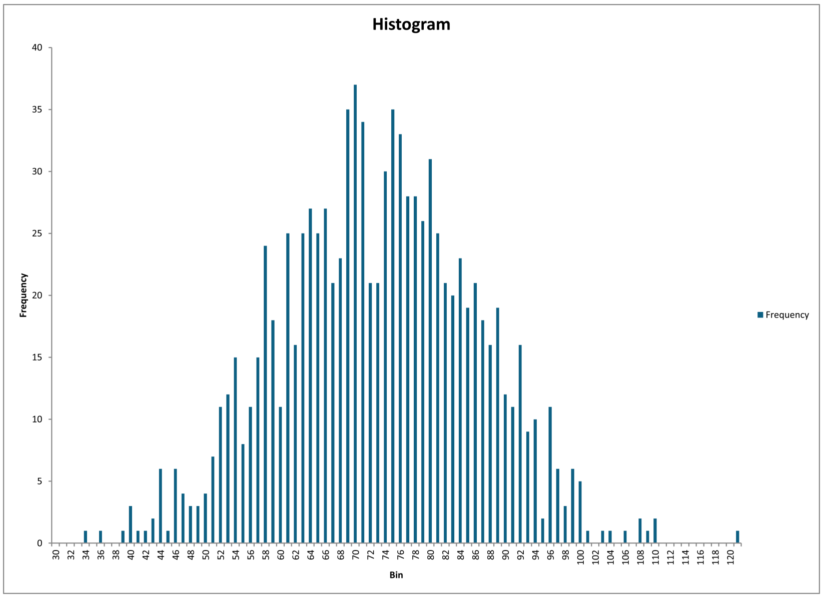


Figure C.1: An Example Histogram

Figure C.1 shows frequency on the vertical (or Y) axis and possible values of data points on the horizontal (or X) axis. The advantage of this chart is that it gives you an overall idea of the shape of the distribution of the uncertainty associated with your data.

Constructing a Histogram

Constructing a Histogram is rather straightforward. As I just mentioned, the vertical axis of this chart is frequency and the horizontal axis represents the ranges of intervals (or bins) that you are interested in. For a given data population, you go through each element and every time a given data point falls within a particular range, you increment the frequency of bin. The height of the bins, therefore, represents the number of times that data points of your dataset occurs within that range.

To illustrate, let's consider the example of rolling four independent (but equal) six-sided dice. In this example we will add up the face value on the dice after each roll and plot them on a Histogram. The bins that we will use for the horizontal axis will therefore be all the possible values for a given roll. That is, since the smallest value for a given roll is four (four times one), our bins will start at four. Since the highest possible value is twenty-four (four times six), then our bins will end at twenty-four. We will have one bin for each possible value between four and twenty four. Figure C.2 shows the Histogram after ten, one hundred, and five thousand rolls, respectively (please note that these Histograms were not generated with the ActionableAgileTM Analytics tool).

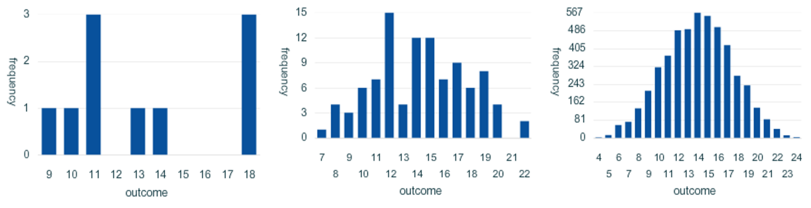


Figure C.2: Rolling Dice Histogram (10 trials, 100 trials, and 5000 trials, respectively)

The thing to note about Figure C.2 is that as the number of trials increases, the shape of the distribution sharpens. In other words, more data is usually better than less data when it comes to visualization (but do not be fooled into thinking you need massive amounts of data to be successful with a statistical approach). Just like the experiment of adding up the results of rolling four, equal, six-sided dice produced random results that could be plotted as the Histograms shown in Figure C.2, so your process will generate random Cycle Times that can be displayed in a similar manner. Figure C.1 is one such example. Again, the lesson here is that the more data you have, the sharper the picture you get.

Conclusion

Let me state for a third time (for those in the cheap seats) that for day-to-day analysis of flow data, I **do not** recommend the use of Histograms. These charts, however, do have their place—the results of a Monte Carlo Simulation being one example. As such, it is helpful to be familiar with them. Just don't be too overly familiar with them. As I mentioned in the preface, if you would like to more fully understand the dangers of Histograms, please see the second book in this series, "Actionable Agile Metrics for Predictability Volume II".

Endnotes

Chapter 1

1. Daniel Vacanti, "When Will It Be Done?". ActionableAgile® Press, 2017.

Chapter 2

1. Sam L. John Savage, *The Flaw of Averages.* Wiley & Sons, Inc., 2009.
2. "Family Size Among Mothers" https://www.pewresearch.org/social-trends/2015/05/07/family-size-among-mothers/ Pew Research Center, May 7, 2015.
3. "Free Burn-Up Chart Templates" https://www.smartsheet.com/content/burn-up-chart-templates SmartSheet, August 29, 2022.
4. https://en.wikipedia.org/wiki/File:Anscombe%27s_quartet_3.svg

Chapter 3

1. "Commute Data", Prateek Singh, 2024.
2. Douglas W. Hubbard, "How to Measure Anything: Finding the Value of Intangibles In Business". John Wiley & Sons, Inc., 2014.
3. Douglas W. Hubbard, "The Failure of Risk Management: Why Its Broken and How to Fix It". John Wiley & Sons, Inc., 2009.

Chapter 4

1. Vacanti, "When Will It Be Done?".

Chapter 6

1. Will Seele and Daniel Vacanti, "Flow Metrics for Scrum Teams" ActionableAgile® Press. 2022.
2. Vacanti, "When Will It Be Done?".

Chapter 7

1. Daniel Vacanti, "Actionable Agile Metrics for Predictability Volume II". ActionableAgile® Press, 2024.

Chapter 8

1. Daniel Vacanti, "Actionable Agile Metrics for Predictability—10th Anniversary Edition". ActionableAgile® Press, 2025.
2. Colleen Johnson, Prateek Singh, Daniel Vacanti, "The Kanban Pocket Guide" https://prokanban.org 2022.

Chapter 9

1. D. Chhajed and TJ. Lowe (eds.) "Building Intuition: Insights From Basic Operations Management Models and Principles." Springer Science +Business Media, LLC 2008.
2. W. J. Hopp and M. L. Spearman "Factory Physics: Foundations of Manufacturing Management, 2nd (ed.)", Irwin/McGraw Hill, New York, NY. 2000.
3. Chhajed, 2008
4. J. D. C. Little, "A Proof of the Queuing Formula: L =AW," Operations Research, 9, (3) 383-387. 1961
5. Chaajed, 2008.
6. Donald Wheeler, https://spcpress.com/pdf/2025/DJW430.pdf 2024.
7. Donald Wheeler, www.spcpress.com/pdf/DJW203.pdf 2009.

Chapter 10

1. Frederick P. Brooks, Jr. "The Mythical Man-Month". Addison-Wesley. 1995 [1975].
2. Donald J. Wheeler, www.spcpress.com/pdf/DJW202.pdf, 2009.

Bibliography

Brooks, Frederick P. Jr., *The Mythical Man-Month.* Addison-Wesley. 1995 [1975].

Deming, W. Edwards. *The New Economics.* 2nd Ed. The MIT Press, 1994

Deming, W. Edwards. *Out of the Crisis.* The MIT Press, 2000.

Duke, Annie. *Thinking In Bets.* Portfolio, 2018.

Hopp, Wallace J., and Mark L. Spearman. *Factory Physics.* Irwin/McGraw-Hill, 2007.

Hubbard, Douglas W. *The Failure of Risk Management: Why Its Broken and How to Fix It.* John Wiley & Sons, Inc., 2014.

Hubbard, Douglas W. *How to Measure Anything: Finding the Value of Intangibles In Business.* John Wiley & Sons, Inc., 2014.

Johnson, Colleen, and Singh, Prateek, and Vacanti, Daniel *The Kanban Pocket Guide* 2021. https://prokanban.org/the-kanban-guide

Little, J. D. C. *A proof for the queuing formula: $L = \lambda W$.* Operations Research. 9(3) 383–387, 1961.

Little, J. D. C., and S. C. Graves. "Little's Law." D. Chhajed, T. J. Lowe, eds. *Building Intuition: Insights from Basic Operations Management Models and Principles.* Springer Science + Business Media LLC, New York, 2008.

Reinertsen, Donald G. *Managing the Design Factory.* Free Press, 1997.

Reinertsen, Donald G. *The Principles of Product Development Flow.* Celeritas Publishing, 2009.

Savage, Sam L. *The Flaw of Averages.* John Wiley & Sons, Inc., 2009.

Seele, Will and Vacanti, Daniel. *Flow Metrics for Scrum Teams* ActionableAgile® Press. 2022.

Shewhart, W. A. *Economic Control of Quality of Manufactured Product*, 1931.

Shewhart, W. A. *Statistical Method from the Viewpoint of Quality Control*, 1939.

Singh, Prateek. *Scaling Simplified* 2024.

Vacanti, Daniel S. "Actionable Agile Metrics for Predictability—10th

Anniversary Edition" *ActionableAgile® Press* February 2025.

Vacanti, Daniel S. and Bennet Vallet. "Actionable Metrics at Siemens Health Services". *AgileAlliance.com. 1* Aug 2014.

Vallet, Bennet. "Kanban at Scale: A Siemens Success Story." *Infoq.com.* 28 Feb 2014.

Vega, Frank. "Are You Just an Average CFD User?" *Vissinc.com.* 21 Feb 2014.

Wheeler, Donald J., and David S. Chambers. *Understanding Statistical Process Control.* 2nd Ed. SPC Press, 1992.

Wheeler, Donald J., https://spcpress.com/pdf/2025/DJW430.pdf, 2024.

Wheeler, Donald J., www.spcpress.com/pdf/DJW202.pdf, 2009.

Wikipedia "Monte Carlo method." *Wikipedia.com* 01 Aug 2014.