2
Probability Theory

Before even mentioning networks, I need to lay the groundwork for a basic understanding of a few concepts necessary to make you a good network analyst. These concepts are part of four broad subjects: probability theory, statistics, linear algebra, and machine learning. We start with probability theory here, because that is the foundation on top of which this pyramid is built. Then, we deal with more specialized statistical concepts in Chapter 3, with machine learning in Chapter 4, and linear algebra in Chapter 5.

These chapters deal with entire fields of human knowledge that are much more vast a complicated than the extremely simplified picture I present here. As with many other chapters, my coverage of the subject is the bare minimum I can get away with. The wisest course of action for you would be to skip this book part entirely and take entire courses on these subjects and then come back to this book and start directly with Part II. Alas, that's not an option for everybody, and that is why this book part covers the basics you need to know to enjoy the rest of the book.

If you want to dive deep into probability theory, there are good books on the subject you should check out[1,2]. I will attempt, where possible, to give forward references to later topics in this book, to let you know why understanding probability theory is important for a network scientist.

[1] Willliam Feller. *An introduction to probability theory and its applications*, volume 2. John Wiley & Sons, 1968

[2] Rick Durrett. *Probability: theory and examples*, volume 49. Duxbury Press, 1996

2.1 *Frequentism and Bayesianism*

Probability theory is the branch of mathematics that allows you to work with uncertain events. It gives you the tools to make inferences in cases of uncertainty.

Probability theory is grounded in mathematical axioms. However, there are different ways to interpret what we really mean with the term "probability". With a very broad brush, we can divide the main interpretations into two camps: the frequentist and the Bayesian.

There are more subtleties to this, but since these are the two main approaches we will see in this book, there is no reason to make this picture more complex than it needs to be.

To understand the difference, let's suppose you have Mrs. Frequent and Mr. Bayes experimenting with coin tosses. They toss a coin ten times and six out of ten times it turns heads up. Now they ask themselves the question: what is the probability that, if we toss the coin, it will turn heads up again?

Mrs. Frequent reasons as follows: "An event's probability is the relative frequency after many trials. We had six heads after ten tosses, thus my best guess about the probability it'll come out as heads is 60%". Note that Mrs. Frequent doesn't really believe that ten tosses gave him a perfect understanding of that coin's odds of landing on heads. Mrs. Frequent knows that he will get the answer wrong a certain number of times, that is what confidence intervals are for, but for the sake of this example we need not to go there.

"Hold on a second," Mr. Bayes says, "Before we tossed it, I examined the coin with my Coin Examiner^{TM} and it said it was a fair coin. Of course my Coin Examiner^{TM} might have malfunctioned, but that rarely happens. We haven't performed enough experiments to say it did, but I admit that the data shows it might have. So I think the probability we'll get heads again is 51%". Just like Mrs. Frequent, also Mr. Bayes is uncertain, and he has a different procedure to estimate such uncertainty – in this case dubbed "credible intervals" – which again we leave out for simplicity.

Herein lies the difference between a frequentist and a Bayesian. For a frequentist only the outcome of the physical experiment matters. If you toss the coin an infinite number of times, eventually you'll find out what the true probability of it landing on heads is. For a Bayesian it's all about degrees of beliefs. The Bayesian has a set of opinions about how the world works, which they call "priors". Performing enough new experiments can change these priors, using a standard set of procedures to integrate new data. However a Bayesian will never take a new surprising event at face value if it is wildly off its priors, because those priors were carefully obtained knowledge coherent with how the world worked thus far.

Figure 2.1 shows the difference between the mental processes between a frequentist and a Bayesian. The default mode for this book is taking a frequentist approach. However, here and there, Bayesian interpretations are going to pop up, thus you have to know why we're doing things that way.

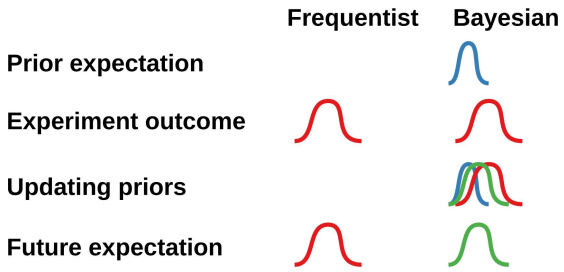


Figure 2.1: Schematics of the mental processes used by a frequentist and a Bayesian when presented with the results of an experiment.

2.2 *Notation*

Probability theory is useful because it gives us the instruments to talk about uncertain processes. For instance, a process could be tossing a die. The first important thing is to understand the difference between *outcome* and *event*. An *outcome* is a *single* possible result of the experiment. A die landing on 2 is an outcome. An *event* is a *set* of possible outcomes on which we're focusing. In our convention, we use X to refer to outcomes. X is a random variable and it can take many values and forms, and we don't know which of them it will be before actually running the process. As for events, they are the focus of all questions in probability theory: you can sum up probability theory as the set of instruments that allow you to ask and answer questions about events (sets of X) such as: "What is the probability that X, the outcome of the process, is this and/or this but not that and/or that?"

Mathematically one writes such a question as $P(X \in S)$, where S is a set of the values that X takes in our question. X is an outcome, $X \in S$ is an event. For instance, if we were asking about the event "will the die land on an even number?", $S = \{2, 4, 6\}$. So, $P(X \in S)$ asks what's the probability of the "die lands on an even number" event – or for X to take either of the $2, 4, 6$ values. Note that elements in S are all possible alternatives: if we write $P(X \in \{2, 4, 6\})$, we're asking about the probability of landing on 2 *or* 4 *or* 6. If you want to have the probability of two events happening simultaneously, you have to explicitly specify it with set notation: $P(X \in \{2, 4, 6\}) \cap P(X \in \{1\})$ asks the probability of landing on an even side and on 1 at the same time.

We also need to consider special questions. For instance, there is the case in which no event happens: $P(X \in \emptyset)$ (here $\emptyset$ refers to the empty set, a set containing no elements). The converse is also important: the probability of any event happening. In the case of the die, there are a total of six possible outcomes. Notation-wise, we define the set of all possible outcomes as $\Omega = \{1, 2, 3, 4, 5, 6\}$. So this is represented as $P(X \in \{1, 2, 3, 4, 5, 6\})$, or $P(X \in \Omega)$. Figure 2.2 shows how the mathematical notation corresponds to our visual

intuition.

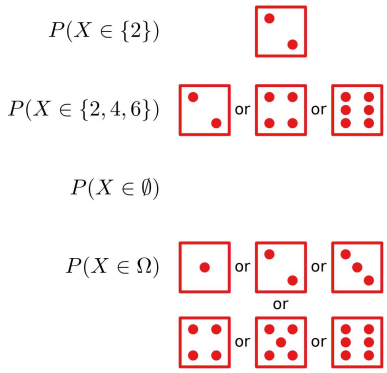


Figure 2.2: A visual shorthand for understanding the mathematical notation of probabilities (left) and the possible outcomes of the "tossing a die" event.

To be more concise, we can skip the explicit reference to the variable X. For instance, we can codify the outcome "the die lands on 3" with the symbol 3. In this way, we can write $P(3)$ to refer to the probability of the die landing on 3, $P(\{2,4,6\})$ for the probability of landing on an even number, $P(\{2,4,6\}) \cap P(1)$ for landing on an even number and on 1, $P(\emptyset)$ for the probability of nothing happening, and $P(\Omega)$ for the probability of anything possible happening.

2.3 *Axioms*

When building probability theory we need to establish a set of axioms: unprovable and – hopefully – self-evident statements that allow you to derive all other statements of the theory. Probability theory rests on three of such axioms.

First, the probability of an event is a non-negative number. Or: talking about a "negative probability" doesn't make any sense. Worst case scenario, an event A is impossible, therefore $P(A) = 0$ – for instance, this is the "nothing happens" case from the previous section when $A = \emptyset$. If A is possible, $P(A) > 0$. In a borderless coin toss, there are only two possible outcomes: heads (H) or tails (T). The coin cannot land on the non-existing rim. Thus, the probability of landing on the rim is zero. It cannot be negative.

Second, certain events occur with probability equal to one. That is, if A is an absolutely certain event, $P(A) = 1$. Using the notation from the previous section: $P(\Omega) = 1$, with $\Omega = \{H, T\}$ for a coin toss. Note that there isn't anything magical about the number 1, we could have said that the maximum probability is equal to 42, π, or "meh". It's just a convenient convention to define your units.

Third, the probability of happening for mutually exclusive events is the sum of their probabilities, or $P(\{H, T\}) = P(H) + P(T)$. A coin cannot land on heads and tails at the same time[3], thus the probability

[3] Get those Schrödinger coins out of my classroom!

that it lands on heads or tails is the sum of the probability of landing on heads and the probability of landing on tails.

As a corollary, you can also multiply probabilities. If A and B are *independent* events, then $P(A)P(B)$ – their multiplication – tells you the probability of *both* events happening. Independent events are events that have no relation to each other, such as you getting a promotion and the appearance of a new spot on the sun. For *dependent* events, you need to take into account this dependence before applying the multiplication. In a fair die, $P(1) = P(2) = 1/6$, but we know that you can't get a 1 if you are getting a 2, so $P(1)P(2)$ is actually zero, not $1/36$. How to perform this check leads us to the world of conditional probabilities.

2.4 *Conditional Probability*

Events do not usually happen in isolation. Things that have happened in the past might influence what will happen in the future. There is a certain probability that the coin will land on heads: $P(H)$. But if I know something happened to the coin before the toss – maybe I put some weights in it, event W – then the probability of heads will change. To handle this scenario, we introduce the concept of "conditional probability". In our scenario, the notation is $P(H|W)$. $P(H|W)$ is the probability of the coin landing on heads – H – given that event W happened.

This view of probability is particularly in line with the Bayesian interpretation, as what you call "prior" is really a synthesis of everything that happened in the past. That is not to say that a frequentist cannot understand conditional probabilities: they can, they just take the usual approach of simply observing what happen before/after something and be done with it.

Conditional probabilities enable you to make a nice set of inferences. Figure 2.3 shows the most basic ones. If you measure $P(H|W)$,

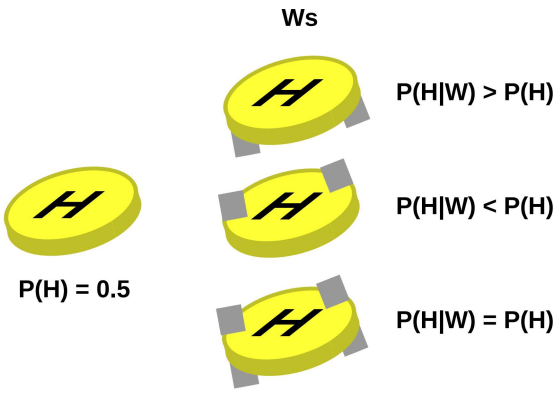


Figure 2.3: The baseline probability of H is 0.5. When you add feet to the coin (W) the coin is more likely to land on the opposite side. Thus, $P(H|W) \neq P(H)$ and the two events are not independent – unless you add feet on both sides as in the bottom example.

you can figure out what event W did to the coin. If $P(H|W) > P(H)$, it means that adding the weight to the coin made it more likely to land on heads. $P(H|W) < P(H)$ means the opposite: your coin is loaded towards tails. The $P(H|W) = P(H)$ case is equally interesting: it means that you added the weight uniformly and the odds of the coin to land on either side didn't change.

This is a big deal: if you have two events and this equation, then you can conclude that the events are independent – the occurrence of one has no effect on the occurrence of the other[4]. This should be your starting point when testing a hypothesis: the null assumption is that there is no relation between an outcome (landing on heads) and an intervention (adding a weight). "Unless," Mr. Bayes says, "You have a strong prior for that to be the case."

Reasoning with conditional probabilities is trickier than you might expect. The source of the problem is that, typically, $P(H|W) \neq P(W|H)$, and often dramatically so. Suppose we're tossing a coin to settle a dispute. However, I brought the coin and you think I might be cheating. You know that, if I loaded the coin, the probability of it landing on heads is $P(H|W) = 0.9$. However, you can't see nor feel the weights: the only thing you can do is tossing it and – presto! – it lands on heads. Did I cheat?

Naively you might rush and say yes, there's a 90% chance I cheated. But that'd be wrong, because the coin already had a 50% chance of landing on heads without any cheating. Thus $P(H|W) \neq P(W|H)$, and what you really want to estimate is the probability I cheated given that the coin landed on heads: $P(W|H)$. How to do so, using what you know about coins ($P(H)$) and what you know about my integrity ($P(W)$), is the specialty of Bayes' Theorem.

[4] Note that here I'm talking about *statistical* independence, which is not the same as *causal* independence. Two events could be statistically dependent without being causally dependent. For instance, the number of US computer science doctorates is statistically dependent with the total revenue of arcades (`http://www.tylervigen.com/spurious-correlations`). This is what the mantra "correlation does not imply causation" means: correlation is mere statistical dependence, causation is causal dependence, and you shouldn't confuse one with the other. You should check [Pearl and Mackenzie, 2018] to delve deeper into this.

2.5 *Bayes' Theorem*

Bayes' Theorem is an almost magical formula that allows you to estimate the probability of an event based on your priors. Keeping the example of cheating on a coin toss, we want to estimate the probability I cheated and rigged the coin so it lands on heads after we tossed it and it indeed landed on heads – in mathematical notation: $P(W|H)$. To do so, you need to have priors. You need to know: what's the probability of heads for all coins in the world (whether they are rigged or not, $P(H)$), what's the probability I rigged the coin ($P(W)$), and what is the probability of obtaining heads on a rigged coin ($P(H|W)$). Without further ado, here's one of the most important formulas in human history:

$$P(W|H) = \frac{P(H|W)P(W)}{P(H)}.$$

Figure 2.4 shows a graphical proof of the theorem. When trying to derive $P(W|H)P(H)$, we realize that's identical to $P(H|W)P(W)$, from which Bayes' theorem follows.

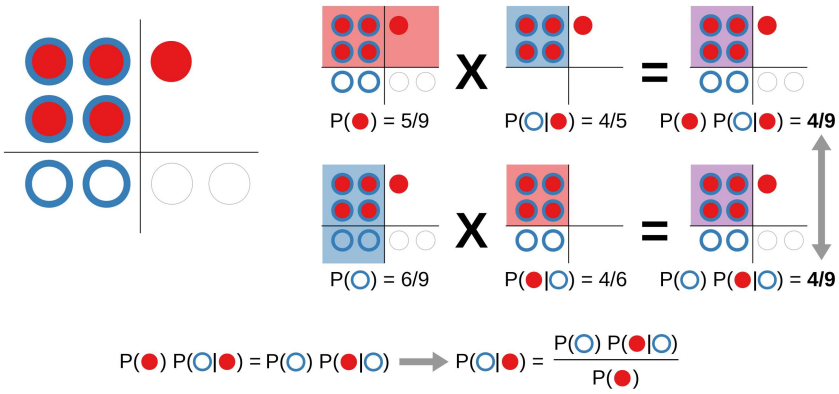


Figure 2.4: The table on the left shows the occurrence of all possible events: red circles (5), blue borders (6), red circles with blue borders (4) and neither (2).

I already told you that I'm a pretty good coin rigger ($P(H|W) = 0.9$). For the sake of the argument, let's assume I'm a very honest person: the probability I cheat is fairly low ($P(W) = 0.3$).

Now, what's the probability of landing on heads ($P(H)$)? $P(H)$ is trickier than it appears, because we're in a world where people might cheat. Thus we can't be naive and saying $P(H) = 0.5$. $P(H)$ is 0.5 if rigging coins is impossible. It's more correct to say $P(H|-W) = 0.5$: a non rigged coin (if W didn't happen, which we refer to as $-W$) is fair and lands on heads 50% of the times. The real $P(H)$ is $P(H|-W)P(-W) + P(H|W)P(W)$. In other words: the probability of the coin landing on heads is the non rigged heads probability if I didn't rig it ($P(H|-W)P(-W)$) plus the rigged heads probability if I rigged it ($P(H|W)P(W)$).

The probability of not cheating $P(-W)$ is equal to $1 - P(W)$. This is because cheating and non cheating are mutually exclusive and either of the two *must* happen. Thus we have $\Omega = \{W, -W\}$. Since $P(\Omega) = 1$ and $P(W) = 0.3$, the only way for $P(W, -W)$ to be equal to 1 is if $P(-W) = 0.7$.

This leads us to: $P(H) = P(H|-W)P(-W) + P(H|W)P(W) = 0.5 \times 0.7 + 0.9 \times 0.3 = 0.62$. Shocking.

The aim of Bayes' theorem is to update your prior about me cheating ($P(W)$) given that, suspiciously, the toss went in my favor ($P(W) \rightarrow P(W|H)$). Plugging in the numbers in the formula:

$$P(W|H) = \frac{0.9 \times 0.3}{0.62} = 0.43.$$

A couple of interesting things happened here. First, since the event went in my favor, your prior about me possibly cheating got updated. Specifically, the event became more likely: from 0.3 to 0.43. Second, even if my success probability after cheating is very high, it is still more likely that I didn't cheat, because your prior about my lack of integrity was low to begin with.

This second aspect is absolutely crucial and it's easy to get it wrong in everyday reasoning. The textbook example is the cancer diagnosing machine. Let's say that 0.1% of people develop a cancer, and we have this fantastic diagnostic machine with an accuracy of 99.9%: the vast majority of people will be diagnosed correctly (positive result for people with cancer and negative for people without). You test yourself and the test is positive. What's your chance of having cancer? 99.9% accuracy is pretty damning, but before working on your last will, you apply Bayes' Theorem:

$$P(C|+) = \frac{0.999 \times 0.001}{0.999 \times 0.001 + 0.001 \times 0.999} = 0.5.$$

The probability you have cancer is *not* 99.9%: it's a coin toss! (Still bad, but not *that* bad).[5]

The real world is a large and scary environment. Many different things can alter your priors and have different effects on different events. The way a Bayesian models the world is by means of a Bayesian network: a special type of network connecting events that influence each other. Exploring a Bayesian network allows you to make your inferences by moving from event to event. I talk more about Bayesian networks in Section 6.4.

[5] Of course, in the real world, if you took the test it means you thought you might have cancer. Thus you were not drawn randomly from the population, meaning that you have a higher prior that you had cancer. Therefore, the test is more likely right than not. Bayes' theorem doesn't endorse carelessness when receiving a bad news from a very accurate medical test.

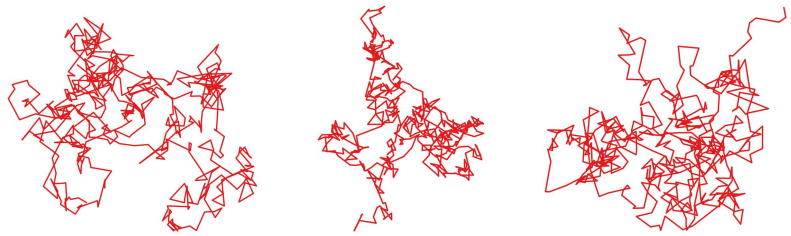
2.6 *Stochasticity*

Colloquially, a stochastic process is one or more random variables that change their values over time. The quintessential stochastic process is Brownian motion. Brown observed very light pollen particles on water changing directions, following a *stochastic* path that seemed governed purely by randomness. Interestingly, this problem was later solved by Einstein in one of his first contributions to science[6], working off important prior work[7]. He explained the seemingly random changes of direction as the result of collision between the pollen and water molecules jiggling in the liquid.

When you have a stochastic process, there is an almost infinite set of results. The pollen can follow potentially infinite different paths. When you observe an actual grain, you obtain only one of those paths. The observed path is called a realization of the process. Figure 2.5 shows three of such realizations, which should help you visualize

[6] Albert Einstein. Über die von der molekularkinetischen theorie der wärme geforderte bewegung von in ruhenden flüssigkeiten suspendierten teilchen. *Annalen der physik*, 4, 1905

[7] Louis Bachelier. Théorie de la spéculation. In *Annales scientifiques de l'École normale supérieure*, volume 17, pages 21–86, 1900



(a) (b) (c)

Figure 2.5: Three realizations of a Brownian stochastic motion on a two dimensional plane.

the intrinsic randomness of the change of direction.

Whenever you encounter the word "stochastic" in this book or in a paper, we're referring to a process governed by these dynamics. For instance, a stochastic matrix is a matrix whose rows and/or columns sum up to one. We call it stochastic, because such matrices are routinely used to describe stochastic processes. By having their rows to sum to one, you can interpret each entry of the row as the *probability* of its corresponding event. The row in which you are tells you the current state of the process, the column tells you the next possible state, and the cell value tells you the probability of transitioning to each of the next possible states (column) given the current state (row). In other words, it is the probability of one possible realization of a single step in a stochastic process. In network science, you normally have stochastic adjacency matrices, which are the topic of Section 8.2.

0.3	0.2	0.08	0	0.07	0.1	0	0.07	0.2
0.1	0.3	0.04	0.04	0.05	0.2	0.04	0.09	0.1
0.08	0.06	0.3	0.06	0	0.2	0.06	0.06	0.1
0	0.08	0.08	0.2	0	0.2	0.08	0.2	0.08
0.07	0.07	0	0	0.3	0.1	0.08	0.2	0.2
0.07	0.1	0.1	0.06	0.07	0.3	0.04	0.09	0.08
0	0.08	0.08	0.08	0.1	0.1	0.2	0.08	0.2
0.05	0.09	0.04	0.1	0.1	0.1	0.04	0.3	0.1
0.1	0.1	0.08	0.03	0.1	0.1	0.08	0.1	0.3

Figure 2.6: A right stochastic matrix.

Figure 2.6 is a stochastic matrix[8]. The rows tell you your current state and the columns tell you your next state. If you are in the first row, you have a 30% probability of remaining in that state (the value of the cell in the first row and first column is 0.3). You have a 20% probability of transitioning to state two (first row, second column), 8% probability of transitioning to state three, and so on.

[8] Specifically, it is a right stochastic matrix: the rows sum to one, although there's a bit of rounding going on. In a left stochastic matrix, the columns sum to one.

2.7 *Markov Processes*

It should be clear now that, even if the next state is decided by a random draw, a stochastic process isn't necessarily uniformly random. In Brownian motion, the next position is determined by your

previous position as well as a random kick. This observation is at the basis of a fundamental distinction between three flavors of stochastic processes, which are the most relevant for network science. The three flavors are: Markov processes, non-Markov processes, and higher-order Markov processes.

In a Markov process, the next state is exclusively dependent on the current state and nothing else. No information from the past is used: only the present state matters. That is why a Markov process is usually called "memoryless". The stochastic process I described when discussing Figure 2.6 is a typical Markov process. The only thing we needed to know to determine the next state was the current state: in which row are we?

The classical Markov process in network science is the random walk. A random walker simply chooses the next node it wants to occupy, and its options are determined solely by the node it is currently occupying. Rather surprisingly, random walks are one of the most powerful tools in network science and have been applied to practically everything. I'm going to introduce them properly in Chapter 11, but they will pop up throughout the book – for instance, in community discovery (Part X) and in network sampling (Chapter 29). Figure 2.7 shows an example of a random walk. As you can see, we start from the leftmost node. From that state, reaching the two rightmost ones is impossible because the nodes are not connected. Only when you transition to another state, new states become available.

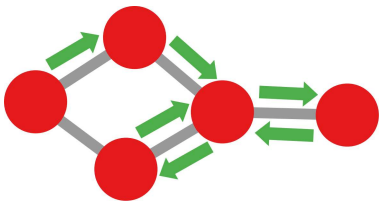


Figure 2.7: A random walk. The green arrows show the state transitions.

A bit more formally, let's assume you indicate your state at time t with X_t. You want to know the probability of this state to be a specific one, let's say x. x could be the id of the node you visit at the t-th step of your random walk. If your process is a Markov process, the only thing you need to know is the value of X_{t-1} – i.e. the id of the node you visited at $t-1$. In other words, the probability of $X_t = x$ is $P(X_t = x|X_{t-1} = x_{t-1})$. Note how $X_{t-2}, X_{t-3}, ..., X_1$ aren't part of this estimation. You don't need to know them: all you care about is X_{t-1}.

On the other hand, a non-Markov process is a process for which knowing the current state doesn't tell you anything about the next possible transitions. For instance, a coin toss is a non-Markov process. The fact that you toss the coin and it lands on heads tells you nothing

about the result of the next toss – under the absolute certainty that the coin is fair. The probability of $X_t = x$ is simply $P(X_t = x)$: there's no information you can gather from your previous state.

Finally, we have higher-order Markov processes. Higher-order means that the Markov process now has a memory. A Markov process of order 2 can remember one step further in the past. This means that, now, $P(X_t = x|X_{t-1} = x_{t-1}, X_{t-2} = x_{t-2})$: to know the probability of $X_t = x$, you need to know the state value of X_{t-2} as well as of X_{t-1}. More generally, $P(X_t = x|X_{t-1} = x_{t-1}, X_{t-2} = x_{t-2}, ..., X_{t-m} = x_{t-m})$, with $m \leq t$.

The classical network examples of a higher order Markov process is the non-backtracking random walk (Figure 2.8). In a non-backtracking random walk, once you move from node u to node v, you are forbidden to move back from v to u. This means that, once you are in v, you also have to remember that you came from u. Higher order Markov processes are the bread and butter of higher order network problems, which is the topic of Chapter 34.

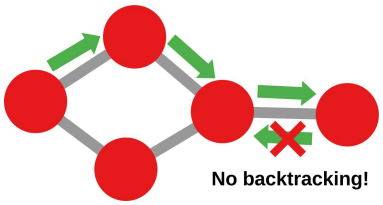


Figure 2.8: A non-backtracking random walk. The green arrows show the state transitions.

2.8 *Alternatives to Probability Theory*

When it comes to dealing with uncertainty, probability theory is not the only game in town. Here I briefly present two alternatives. These will be handy when we focus specifically on probabilistic networks in Chapter 28.

Dempster-Shafer's Theory of Evidence

In Dempster-Shafer's theory of evidence (DST) we take the key insight from Bayes and we turn it up to eleven. One thing that underlies Bayesian thought is that probabilities are subjective. If two people have different priors, say A and B such that $P(A) \neq P(B)$, then they will disagree on the probability of event C, which will depend on the priors. This doesn't happen with a frequentist framework, because there are no priors and $P(C)$ is based on objective data available to everyone. However, besides this subjectivity, Bayes still uses the axioms and the rules of probability theory.

DST is a generalization of probability theory, which moves from exact probabilities to probability intervals. Its central parts are *beliefs* and *plausibilities*, and these two things don't necessarily behave like probabilities[9,10].

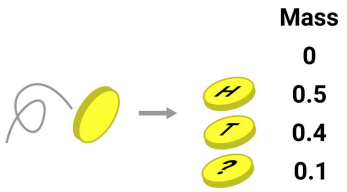


Figure 2.9: An illustration of Mass in DST. After tossing a coin (left) each each subset of Ω obtains a value.

Starting with beliefs, the first thing you need is a *degree of belief*, which estimates your ability to prove a set of beliefs[11]. This degree of belief is quantified by a function which is conventionally called its Mass function. Figure 2.9 shows a relatively simple example when tossing a coin – slightly loaded on heads. The distinction between Mass in DST and classical probability is that it considers the case "we don't know whether heads or tail" as distinct from "heads" and "tail". In probability theory, you wouldn't make this distinction, because no other outcome than heads or tails can happen, even if for some reason you don't know the outcome. But in DST you want to model this, because we're talking about the ability of proving our statement, so we need to specifically take into account the situation in which we don't actually know the result – e.g., if the coin rolled under the sofa and we can't see it. In that case, we don't have any evidence to say that the coin landed on heads or tail.

In summary, if $\Omega = \{H, T\}$, then $p(\Omega) = p(H) + p(T) = 1$, but $Mass(\Omega) \neq Mass(H) + Mass(T)$: $Mass(\Omega)$ is less trivial and actually informative – it is the amount of uncertainty we have about the outcome of the event given the imperfection of our evidence. So the Mass function is basically giving all the available evidence a probability and obeys the following two rules:

1. $Mass(\emptyset) = 0$, and
2. $\sum_{X \in 2^\Omega} Mass(X) = 1$.

Here, 2^Ω means all possible subsets of Ω, which in my simple case are: $\emptyset$, $\{H\}$, $\{T\}$, and $\{H, T\}$. The first property means that it's impossible to prove that nothing happened – we know we tossed the coin, it must have landed on something. That's why the ugly coin toss drawing on the left of Figure 2.9 is necessary: to show we performed the tossing. The second property means that Ω contains all possible answers to our question – so what's actually true must be a subset of Ω.

[9] Arthur Dempster. Upper and lower probabilities induced by a multi-valued mapping. *Annals of Mathematical Statistics*, 38, 1967

[10] Glenn Shafer. *A mathematical theory of evidence*, volume 42. Princeton university press, 1976

[11] Judea Pearl. Reasoning with belief functions: An analysis of compatibility. *International Journal of Approximate Reasoning*, 4(5-6):363–389, 1990

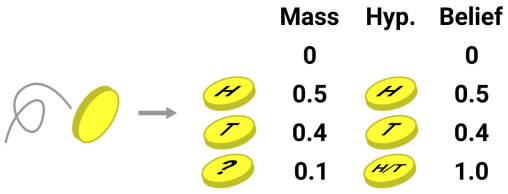


Figure 2.10: An illustration of Belief in DST, based on the Mass from Figure 2.9. Note how Belief of a subset of size 1 is equal to its Mass.

Now that you know the probabilities of all possible outcomes, you must make a hypothesis which is a set of potential outcomes. To estimate your ability to prove your hypothesis, you sum all the Mass values of all the subsets of your hypothesis. This is the Belief function, and you can see how it works in Figure 2.10. If your hypothesis has no support in the gathered evidence then its Belief value is zero, while if it is absolutely certain then Belief evaluates to one. So Belief tells you how likely your hypothesis – or any of its subsets – is to be proven given the available evidence.

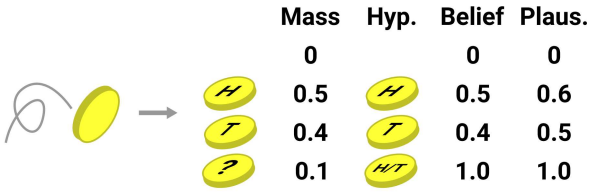


Figure 2.11: An illustration of Plausibility in DST, based on the Mass and Belief Figure 2.10.

DST also allows to compute the Plausibility function, which is an upper bound of Belief. In practice, you can tally up all the evidence of your hypothesis not containing the truth and take the inverse of it. Figure 2.11 shows how it works. Plausibility is basically estimating how much your hypothesis can survive an attempt to prove it false. A handy rule to remember is that $Plausibility(X) = 1 - Belief(\bar{X})$, where $\bar{X}$ is the complementary set of X – the plausibility of something is the opposite of your belief of that something being proven false.

We go through the trouble of defining these things because DST has some advantages. For instance, there are some operations you can do with the Belief and Plausibility functions – which we are not going to see here – but allow you to work with different hypotheses in conflict. Classical probability theory is ill suited to handle these cases. For instance, suppose that the ice cream shop has three flavors: chocolate, strawberry, and vanilla. We want to share an ice cream and my preference is 99% chocolate and 1% vanilla, while yours are 99% strawberry and 1% vanilla. We should obviously go for vanilla, and DST agrees, but if we took a probabilistic approach ignoring DST, you might instead say that vanilla is the least likely solution, and we would end up with either chocolate of strawberry, much

to the distress of either of us. For instance by naively aggregating probabilities as 49.5% for each strawberry and chocolate. To be fair, DST also ends up saying weird things occasionally, which has led researchers to formulate ways to turn it into computable functions that are different from the ones I explained[12].

[12] Kari Sentz and Scott Ferson. Combination of evidence in dempster-shafer theory. 2002

Moreover, probability theory must assign a probability to an event, even if there is no evidence for it, while DST can simply give it zero Mass. For instance, if we have a potentially loaded die, in probability theory using a Bayesian approach we must start by assigning a prior probability of 1/6 to all outcomes *even if we have zero evidence for it*. In DST, you'd give them Mass zero instead, and Mass one to Ω and then start gathering evidence.

Fuzzy Logic

In probability theory, you only deal with boolean events, whose truth values can either be zero or one. Either something is false or it is true. The coin either landed on heads or on tails. In fuzzy logic, you work with something different. Things can have degrees of truthiness, which is to say we assign them a truth value between zero and one. If it is zero, we're certain that a statement is false, if it is one we're certain it is true, and if it is a value in between then there is some vagueness about whether it is true or false[13].

[13] Petr Hájek. *Metamathematics of fuzzy logic*, volume 4. Springer Science & Business Media, 2013

For instance, at the moment of writing this paragraph I am 39 years old. Is that young or old? Well, you could line up 100 people and ask them this question. Maybe 60 will say that I'm young, 39 will say that I'm old (I'm so insecure I am disrespected even in my thought experiments), and 1 will say something else. In probability theory you could model this as something like: there's a 39% chance a random person will call me old (hey!). But in fuzzy logic you'd do something different. You could say that I belong to both the sets of young people and old people, with different strengths. I'm 60% young and 39% old – I frankly don't know if that's an improvement over the alternative.

The consequences of this difference lead to different outcomes when working with fuzzy logic. We'll see a basic common example[14], but know that there are alternative ways to implement fuzzy logic[15]. Let's assume that the degree to which a person belongs to an age class depends on their age, following the function I draw in Figure 2.12.

[14] Ebrahim H Mamdani. Application of fuzzy algorithms for control of simple dynamic plant. In *Proceedings of the institution of electrical engineers*, volume 121, pages 1585–1588. IET, 1974

[15] Tomohiro Takagi and Michio Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE transactions on systems, man, and cybernetics*, (1):116–132, 1985

For the age I highlight, probability theory can say the following things:

- The probability of picking somebody who could call me both young and old (assuming I ask multiple time and people can

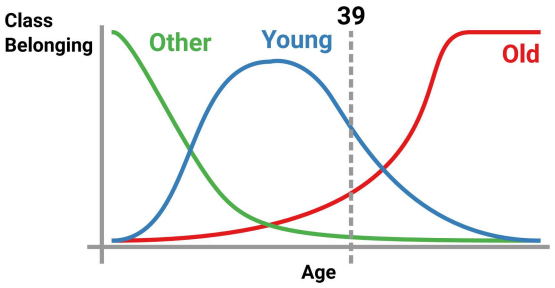


Figure 2.12: An illustration of fuzzy logic. The degrees of belonging (y axis) to different age classes (line color) for a given age (x axis).

change their mind independently from what they said the first time) is $P(Y \cap O) = P(Y)P(O) = 0.234$;

- The probability of picking somebody who will call me either young or old is $P(Y \cup O) = P(Y) + P(O) = 0.99$;
- The probability of somebody not calling me young is $P(\bar{Y}) = 1 - P(Y) = 0.61$.

But in fuzzy logic we have:

- My belonging to the class of people who are both young and old is $P(Y \cap O) = \min(P(Y), P(O)) = 0.39$ – it can't be any higher than my minimum belonging, because to fully belong to the young-old class I must be fully young and fully old;
- My belonging to the class of people who are either young or old is $P(Y \cup O) = \max(P(Y), P(O)) = 0.6$ – it can't be any lower than my maximum belonging, because if I am fully old then I am also fully-young-or-fully-old;
- My belonging to the class of people who are not young is $P(\bar{Y}) = 1 - P(Y) = 0.61$.

2.9 *Summary*

1. Probability theory gives you the tools to make inferences about uncertain events. We often use a frequentist approach, the idea that an event's probability is approximated by the aggregate past tests of that event. Another important approach is the Bayesian one, which introduces the concept of priors: additional information that you should use to adjust your inferences.
2. Probabilities are non-negative estimates. The set of all possible outcomes has a probability sum of one. Summing two probabilities tells you the probability of either of two independent outcomes to happen.

3. The conditional probability $P(A|B)$ tells you the probability of an outcome A given that you know another outcome B happened. If $P(A|B) \neq P(A)$ then the two outcomes are not independent. Bayes' Theorem allows you to infer $P(A|B)$ from $P(B|A)$.

4. When we track the change over time of one or more random variables, we're observing a stochastic process. Markov processes are stochastic processes whose status exclusively depends on the status of the system in the previous time step.

5. There are alternative to probability theory when working with uncertainty. Dempster-Shafer's theory of evidence allows to work with the degrees of beliefs in specific hypotheses, while fuzzy logic allows for multiple things to be a little bit true at the same time.

2.10 *Exercises*

1. Suppose you're tossing two coins at the same time. They're loaded in different ways, according to the table below. Calculate the probability of getting all possible outcomes:

$p_1(H)$	$p_2(H)$	H-H	H-T	T-H	T-T
0.5	0.5				
0.6	0.7				
0.4	0.8				
0.1	0.2				
0.3	0.4				

2. 60% of the emails hitting my inbox is spam. You design a phenomenal spam filter which is able to tell me, with 98% accuracy, whether an email is spam or not: if an email is not spam, the system has a 98% probability of saying so. The filter knows 60% of emails are spam and so it will flag 60% of my emails. Suppose that, at the end of the week, I look in my spam box and see 963 emails. Use Bayes' Theorem to calculate how many of those 963 emails in my spam box I should suspect to be non-spam.

3. You're given the string: "OCZ XJMMZXO VINRZM". Each letter follows a stochastic Markov process with the rules expressed by the table at `http://www.networkatlas.eu/exercises/2/3/data.txt`. Follow the process for three steps and reconstruct the correct answer. (Note, this is a Caesar cipher[16] with shift 7 applied three times, because the Caesar cipher is a Markov process).

4. Suppose that we are examining a painting and we're trying to date it with the century when it was produced. Find out the Belief and Plausibility values for all hypotheses given the following Mass

[16] `https://en.wikipedia.org/wiki/Caesar_cipher`

estimation (note that, by definition $\Omega = \{\text{XIV, XV, XVI}\}$ must have Belief and Plausibility equal to one):

Hypothesis	Mass	Belief	Plausibility
$\emptyset$	0.00		
XIV	0.16		
XV	0.04		
XVI	0.21		
{XIV, XV}	0.34		
{XV, XVI}	0.16		
{XIV, XVI}	0.08		
Ω	0.01	1	1

3
Statistics

In Chapter 2 I explained the basic concepts of probability theory you need to understand to be a good network scientist. This chapter focuses on basic statistical concepts that are also necessary to analyze your networks. The disclaimer I put at the beginning of Chapter 2 also holds here: statistics is much more vast and complicated than what I present here. This chapter is emphatically *not* a substitute for a proper statistics textbook[1,2], which you could use to study this stuff further. You'll get a sense of how powerful statistics is[3], in that it can allow you to support any point. As the saying goes: there are lies, damn lies, and statistics.

The main difference between probability and statistics is that you can do a lot of work in probability theory without actually looking at any data. When data takes the center stage, you enter in the world of statistics. Statistics covers more than simply describing your data: you should think in statistical terms also when collecting, cleaning, validating your data. But here we ignore all that and we focus on the tools that allow you to say something interesting about your data, assuming you did a good job collecting, cleaning, and validating it.

[1] Charles Wheelan. *Naked statistics: Stripping the dread from the data*. WW Norton & Company, 2013

[2] Richard McElreath. *Statistical rethinking: A Bayesian course with examples in R and Stan*. Chapman and Hall/CRC, 2018

[3] Darrell Huff. *How to lie with statistics*. Penguin UK, 1954

3.1 Summary Statistics

Mean & Median

One common use of statistics is to give a quick description of what is in the data. The most classical task is to try and figure out what are the values you'd expect to find if you were to look directly at the data. For instance, if you want to know the height of the average human, you might want to calculate the mean height: $\mu(H) = \sum_i H_i/|H|$, where H_i is the height of one human. The mean in this case would tell you what you'd expect to see if you were to measure the height of a random person.

However, that works for height because height is normally dis-

tributed – meaning that the average value is actually the mean and values farther from the mean are progressively more rare. We'll see what a normal distribution looks like in Section 3.2. The same section will also tell you that not all variables distribute like that: in some cases the mean is actually not a great approximation of your average (or "typical") case. Wealth is like that: a tiny fraction of people own vastly more than the majority. In this case, the median gives you a better idea[4]. The median tells you the value that splits the data in two equally populated halves: 50% of the points are below the median and 50% of the points are above.

[4] David J Sheskin. *Handbook of parametric and nonparametric statistical procedures*. Chapman and hall/CRC, 2003

Figure 3.1 shows that the mean and the median can be quite different. In network science, we use the mean extensively – even though maybe we shouldn't. When we'll talk about the number of connections a node has in a network (Chapter 9), we'll see we routinely take its "average" by calculating its mean. But connection counts in real networks typically do not follow a neat normal distribution. These distributions tend to look more like Figure 3.1(b) than Figure 3.1(a). So, perhaps the mean count of connections is not the most meaningful thing you can calculate.

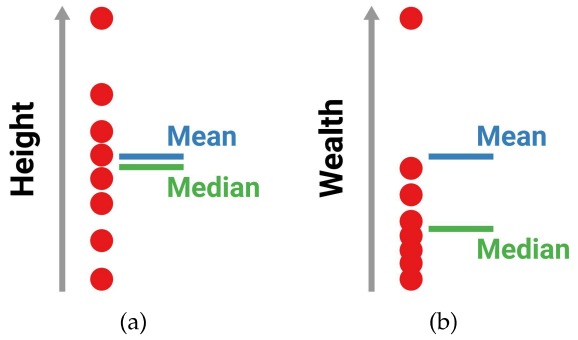


Figure 3.1: The mean (blue) and median (green) or two different variables (y axis): (a) normally distributed height; (b) skewed distribution of wealth.

This disconnect between the mean and the typical case is true for the arithmetic mean I show here, but there are other types of means – such as geometric or harmonic – which can take into account some special properties of the data[5].

[5] Philip J Fleming and John J Wallace. How not to lie with statistics: the correct way to summarize benchmark results. *Communications of the ACM*, 29 (3):218–221, 1986

Variance & Standard Deviation

When we deal with an average observation, we might want to know not only its expected value, but also how much we expect it to differ from the actual average value. Even if the average human height is, let's say, 1.75 meters, we could think of two radically different populations. In the first, almost everyone is more or less 1.75 and heights don't vary much. In the other, the opposite is true: the average is still 1.75, but people could be anything between 1 meter and 2.5 meters. So the heights in this second population vary much more. We

need to have a tool allowing us to distinguish these two populations. Since the difference is all about how much the heights *vary*, we call this measure *variance*. Variance (and standard deviation) helps you quantify how dispersed your values are away from the mean.

Variance is almost literally the mean difference from the mean. The only tweak is that we take the square of this difference: $var(H) = \mu((H - \mu(H))^2)$. We take the square because we don't want values below the mean to cancel out values above the mean. The standard deviation is simply the square root of the variance: $\sigma(H) = \sqrt{var(H)}$. The advantage of the standard deviation is that it ends up having the same units as the original variable. For example, for heights measured in cm, the variance will be in cm^2, but the standard deviation will be in cm again. Figure 3.2 shows what it looks like to have different variances for variables with the same mean.

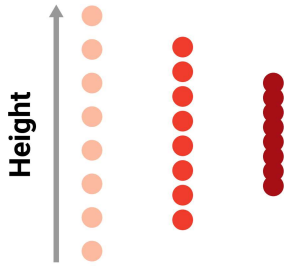


Figure 3.2: The variances of three different populations, going from left (high variance, bright red) to right (low variance, dark red).

The concept of variance will be important when we talk about data dimensionality reduction (Section 5.6) and degree distributions (Section 9.3). We will even see how to modify its definition to create a notion of network variance (Section 47.5)

Distribution & Skewness

Knowing how skewed your data is can be quite important – in wealth distribution, how skewed the data is equals how screwed people are. Variance and standard deviation can help you quantify this. There are different formulas and different terms to talk about skewness[6], but for our purposes we limit ourselves to a bit of terminology.

First: what actually is a distribution? I've used this term in an intuitive way without really defining it. Let's do it here. When you perform an experiment, or observe a stochastic process, you have many possible outcomes. For instance, you're measuring the heights of all people in a country and so your possible outcomes are all the possible heights a person can have. Sometimes, you're not interested on the frequency of measuring a specific outcome – say 175 cm. Sometimes, you want to study all possible outcomes together, to determine which is more likely, what you could expect when you

[6] Paul T Von Hippel. Mean, median, and skew: Correcting a textbook rule. *Journal of statistics Education*, 13(2), 2005

measure more people, if there are maximums and minimums you don't expect to ever exceed, and so on. This is the task of a distribution. A distribution is a function that, for each outcome in the set of all possible ones (called the "sample space"), tells you how many times you measured a given outcome. Figure 3.3 shows a vignette on how to interpret a plot showing you a distribution.

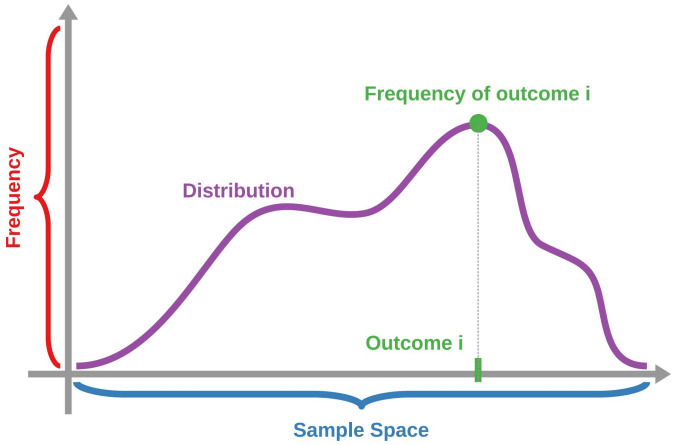


Figure 3.3: A distribution, connecting every possible outcome in the sample space (x axis) to a frequency (y axis).

Skewness is a property of a distribution, it measures its symmetry. A symmetric distribution has no skewness, and any asymmetry will create a non-zero skewness value: positive if the skewedness is on the right and negative if it is on the left – see Figure 3.4.

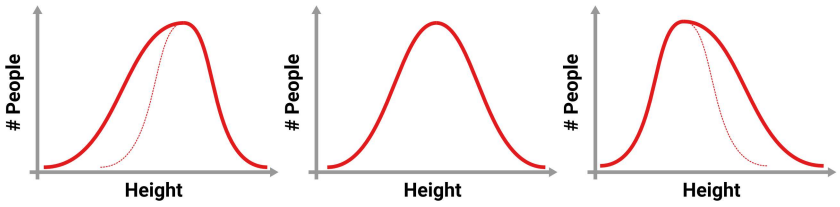


Figure 3.4: Three distributions with different skewness: (a) the values lower than the average are more likely, (b) no skewness, (c) the values higher than the average are more likely.

An important related concept to skewness is the heavy-tailed distribution. There are two types of heavy-tailed distributions that interest network scientists, because they're often observed in real world networks. They are the long tail and the fat tail[7]. In a long tail, you can find arbitrarily large outliers: that means the very highest value can be many times larger than the second-highest one. You might know these from the popular concept of the black swan[8]. In these kinds of distributions, observations can happen that are much more extreme than anything we have seen so far. You'll see a network example in Chapter 9, when we'll see it is common for nodes in real network to have a long tail in the number of connections attached to them. With a fat tail, you still have outliers that can be many times

[7] In case you were wondering: yes, statisticians body-shame distributions.

[8] Nassim Nicholas Taleb. *The black swan: The impact of the highly improbable*, volume 2. Random house, 2007

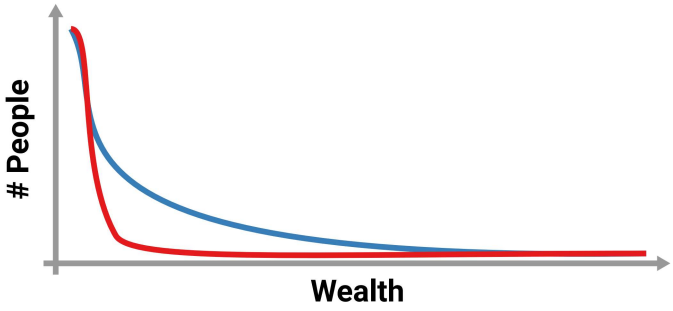


Figure 3.5: A long tail (red) and a fat tail (blue).

over the average value. However, these outliers are more common and less extreme. Figure 3.5 shows a graphical example.

If we're talking wealth, a long tail world is a world with a single Jeff Bezos and everybody else works in an Amazon warehouse. In a fat tail world, Jeff might not be quite as rich, but there are a few billionaire friends to keep him company.

3.2 *Important Distributions*

Chapter 9 will drill in your head how important distributions are for network science, so it pays off to become familiar with a few of them. First, let's make an important distinction. There are two kinds of distributions, depending on the kinds of values that their underlying variables can take. There are discrete distributions – for instance, the distribution of the number of ice cream cones different people ate on a given day. And there are continuous distributions – for instance the distances you rode on your bike on different days. The difference is that the former has specific values that the underlying variable can take (you may have eaten two or three ice cream cones, but 2.5 is not an option), the latter can take any real value as an outcome. In the first discrete case, we call the distribution a "mass function". In the second case, we call it a "density function".

Figure 3.6 shows some stylized representations of the most important distributions you should pay attention to, which are:

- **Uniform**: in this distribution each event is equally likely. This distribution can be both discrete or continuous. In the discrete case, if you have n possible events, each occurs with probability $p = 1/n$. You get a discrete uniform distribution if you look at the number on a ball extracted from an urn, where all balls in the urn are identified by distinct, progressive numbers without gaps. A continuous uniform distribution could be the amount of time you have to wait for the next drop to come from a leaky faucet that drips once a minute: if you haven't seen the last drop falling, the wait time could be any time between 0 and 60 seconds.

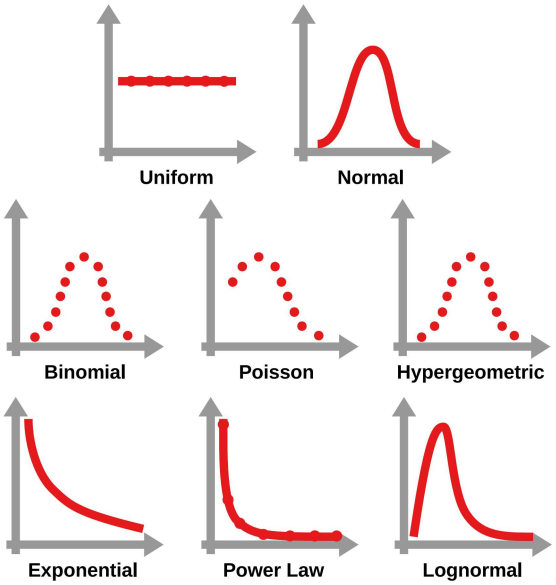


Figure 3.6: A stylized representation of the most common distributions you'll encounter as a network scientist. Solid lines show continuous distributions, while dots show discrete ones – note that some distributions can be both (dotted lines).

- **Normal** (or **Gaussian**): this a very common distribution for continuous variables. The classical example is the distribution of people's heights: most people are of average height, and larger and larger deviations from the average get steadily less likely.

- **Binomial**: this is a discrete distribution, in which you do n experiments, each with success probability p, and you calculate the probability of having n' successes. For instance, suppose you take five balls from an urn containing 50 red and 50 green balls – each time putting the ball you extracted back into the urn. Let's say we count getting a green ball as a "success" here. The number of times you got 0, 1, 2, 3, 4 or 5 green balls over a bunch of trials would follow a binomial distribution.

- **Hypergeometric**: this is yet another discrete probability function. It is very similar to a binomial distribution. Where the binomial described the number of "successes" in an extraction-with-replacement urn game, the hypergeometric describes the more common case of extraction-without-replacement. When you extract a ball from the urn, you don't put it back. It is mathematically less tractable, but much more useful. This is used especially for the task of network backboning (Chapter 27).

- **Poisson**: this is another discrete distribution, which is the number of successes in a given time interval, assuming that each success arrives independently from the previous ones. For instance, the number of meteorites impacting on the moon each year will have a Poisson distribution. Interestingly, many examples commonly

mentioned for explaining a Poisson distribution (number of admittances to a hospital in an hour, number of emails written in an hour, and so on) aren't actually Poisson distributions, because of the "burstiness" of human behavior[9].

[9] Albert-Laszlo Barabasi. The origin of bursts and heavy tails in human dynamics. *Nature*, 435(7039):207, 2005

- **Exponential**: the exponential distribution is a continuous distribution modeling cases in which the probability of something happening is not dependent of how much time has passed since you starting observing the phenomenon. For instance, if there's an epidemics out, the amount of time you have been infection-free bears no weight in determining your probability of being infected, if exposed. This is the reason why this distribution is sometimes called "memoryless" or that it "doesn't age".
- **Power law**: a power law can be both a discrete or a continuous distribution. It describes the relationship between two quantities, the second quantity changes as a power of the first. One practical consequence is that, if you were given a power law plot without axis labels, you would not be able to tell where you are in the distribution, because the slope of the line always looks the same no matter how much you zoom in or out, or whether you're on the head or the tail. An example of discrete power law is Zipf's law[10] recording the frequency of words in a document against their frequency rank. We'll see more than you want to know about power laws when talking about fitting degree distributions in Section 9.3.
- **Lognormal**: a lognormal distribution is the distribution of a continuous random variable whose logarithm follows a normal distribution – meaning the logarithm of the random variable, not of the distribution. This is the typical distribution resulting from the multiplication of two independent random positive variables. If you throw a dozen 20-sided dice and multiply the values of their faces up, you'd get a lognormal distribution. It's very tricky to tell this distribution apart from a power law, as we'll see.

[10] Mark EJ Newman. Power laws, pareto distributions and zipf's law. *Contemporary physics*, 46(5):323–351, 2005b

Sometimes, rather than looking at the mass/density functions, it's more useful to look at their cumulative versions. In practice, you want to ask yourself what is the number of – say – x or *fewer* successes. Each distribution changes in predictable ways, as Figure 3.7 shows.

For instance, a cumulative uniform distribution is a line that goes straight up, because each event adds the same value to the cumulative sum. A cumulative normal distribution has an the shape of a flattened "S". In the power law case, as we'll see, we actually want to see the complement of the cumulative distribution (1 - CDF). This is, interestingly, also a power law.

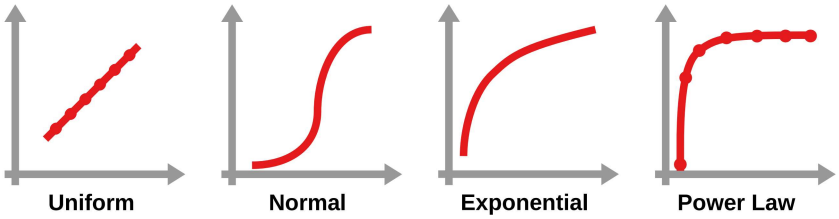


Figure 3.7: A stylized representation of a few cumulative distributions. Same legend as Figure 3.6.

3.3 *p-Values*

One of the key tasks of statistics is figuring out whether what you're observing – a natural phenomena or the result of an experiment – can tell us something bigger about how the world works. The way this is normally done is making an hypothesis, for instance that a specific drug will cause weight loss. To figure out whether it is true, we need to prove that taking the drug actually does something rather than nothing. "The drug does nothing" is what we call the *null hypothesis*, which is what we'd expect – after all, most drugs don't cause weight loss. This is what we colloquially call "burden of proof": the person making the claim that something exists needs to prove that it does, because if we haven't proven that something exists yet there is no reason to believe it does.

We call what you want to prove – "the drug causes weight loss" – the *alternative hypothesis*, because it's the alternative to the null hypothesis.

p-values are among of the most commonly used tools to deal with this problem[11]. The interpretation of p-values is tricky and it is easy to get it wrong. The "p" stands for "probability". Suppose that you give your drug to a bunch of people and, after a few weeks, you see that their weight decreased by 5kg. The p-value tells you the probability that you would be observing an effect this strong – a loss of 5kg – if the null hypothesis was true – i.e. if the drug actually did nothing. Lower p-values mean there is stronger evidence against the null hypothesis. What the p-value does not tell you (but might trick you into thinking it does) is:

[11] Ronald L Wasserstein and Nicole A Lazar. The asa statement on p-values: context, process, and purpose, 2016

- The p-value does **NOT** tell you how likely you are to be right;
- The p-value does **NOT** tell you how strong the effect is;
- The p-value does **NOT** tell you how much evidence you have against your hypothesis.

Etch these bullet points into your brain, because it is so easy to fool yourself. The last of them means that a high p-value does not

mean that the null hypothesis is true. A high p-value just means that the observations we have are compatible with a world where the null hypothesis is true. But it could also mean that our sample is not big enough to draw firm conclusions. Given how tricky it is to get them right, some researchers have called for not using p-values altogether[12].

Figure 3.8 shows a graphical way to understand the p-value. You have a distribution of values that would be produced by the null hypothesis, you pit your measurements against those values, and the more unusual your observations look compared to those that would be produced by the null hypothesis, the lower the p-value. Exactly how to produce this null hypothesis distribution is not something we'll cover here, because it is not so close to the core of network science – although we'll see something similar in Section 19.1.

[12] Raymond Hubbard and R Murray Lindsay. Why p values are not a useful measure of evidence in statistical significance testing. *Theory & Psychology*, 18(1):69–88, 2008

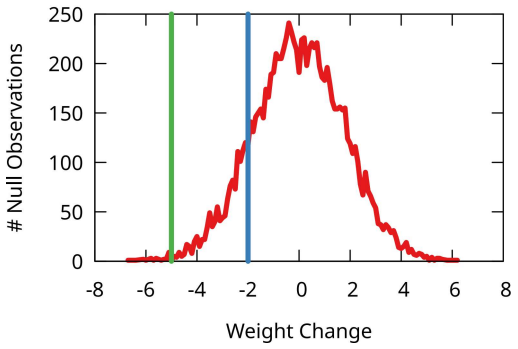


Figure 3.8: In red we have the number of observations (y axis) with a given weight change (x axis) under the null hypothesis. In blue we have an observation of 2kg weight loss after taking the drug ($p = 0.14$). In green we have an observation of 5kg weight loss after taking the drug ($p = 0.003$).

One thing is worth mentioning, though. You will often see some magical p-value thresholds that people use as standards – the most common being $p < 0.05$ and $p < 0.01$. These p-value thresholds say something about the strength of the evidence we want to see before we are willing to reject the null hypothesis of no effect. Beware of these. Not only because – as I said before – they don't mean what you think they mean, but also because of Goodhart's Law[13]. If we say $p < 0.01$ is the gold standard we need to achieve to publish a paper, the natural tendency would be to try and repeat/modify experiments until we get $p < 0.01$. This is know in the literature as p-hacking[14] and has led researchers to publish a flurry of false results.

If the p-value tells you the probability of observing a given result under the null hypothesis, then if you repeat your experiments 100 times the probability that at least one of them will leave to a p-value ≤ 0.01 is actually 63%![15] That is because the probability of getting a $p > 0.01$ is 99% – with this standard of evidence, one percent of the time, you will reject the null hypothesis by accident. You try 100 times, so the formula to know how likely a $p \leq 0.01$ is

[13] "When a measure becomes a target, it ceases to be a good measure."

[14] Megan L Head, Luke Holman, Rob Lanfear, Andrew T Kahn, and Michael D Jennions. The extent and consequences of p-hacking in science. *PLoS biology*, 13(3):e1002106, 2015

[15] https://xkcd.com/882/

becomes $1 - (0.99^{100})$. Sometimes, of course, you do need to run more than one test, and look at more than one p-value. A couple of common options to deal with this are applying the Bonferroni[16,17] or the Holm-Bonferroni[18] corrections to your p-values, systematically lowering the significance threshold to make up for "cheating" by running multiple tests.

p-values are important for network science because we will often have to create null models to figure out whether some property we observe in a network is actually interesting (Section 19.1). Another case is figuring out whether an edge has a weight significantly different from zero (i.e. it actually exists) or not (Chapter 27).

[16] Carlo Bonferroni. Teoria statistica delle classi e calcolo delle probabilita. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commericiali di Firenze*, 8:3–62, 1936

[17] Olive Jean Dunn. Multiple comparisons among means. *Journal of the American statistical association*, 56(293): 52–64, 1961

[18] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70, 1979

3.4 *Correlation Coefficients*

So far we have worked with a single variable and we have done our best to describe how it distributes. However, more often than not, you have more than one variable and you want to know something interesting about how one relates to the other. For instance, you might want to describe how the weight of a person tends to be related to their height.

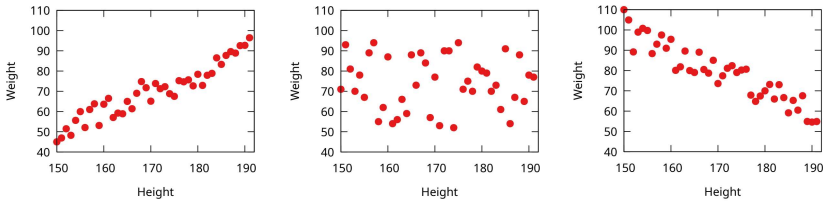
In general, the taller a person is, the more we expect them to weigh. In this sense the two variables vary together. Therefore, we call this concept "covariance[19]." The formula is pretty simple, it is the mean of the product of the deviations from the mean of both variables, so: $cov(H, W) = \mu((H - \mu(H))(W - \mu(W)))$.

[19] John A Rice. *Mathematical statistics and data analysis*. Cengage Learning, 2006

If a person is both a lot taller than the mean and a lot heavier than the mean, the product of their height and weight deviations will be big, and they will contribute a large positive value to the covariance calculation. If a person is both a lot shorter than the mean and a lot lighter than the mean, the product of their height and weight deviations, both negative, will again be a large positive. So, they will also make the covariance turn out bigger. If someone is both tall and light, their positive height deviation and negative weight deviation will be multiplied to become a large negative number, pulling the covariance down. The covariance will be large in total if we observe many tall/heavy and short/light people, and not so many tall/light or short/heavy people (see Figure 3.9, the covariance values are in the figure's caption, first row under $cov(H, W)$).

One problem is that the covariance depends on the units of your variables. That means the size of the covariance can be difficult to interpret – is a covariance of 5.5 high or low? To make sense of it, it helps to normalize it. This is what the Pearson correlation coefficient tries to do[20]. It divides the covariance by the variances of both variables. That means, for example, that if start measuring

[20] Francis Galton. Typical laws of heredity. Royal Institution of Great Britain, 1877



$cov(H, W) = 154.4$
$\rho(H, W) = 0.95$
(a)

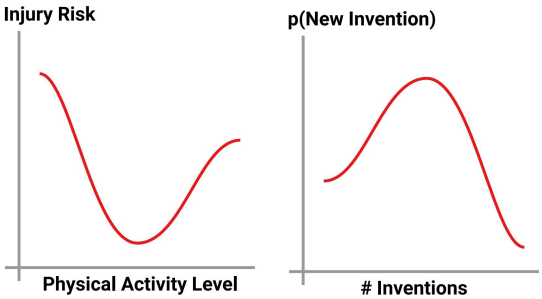
$cov(H, W) = -4.7$
$\rho(H, W) = -0.03$
(b)

$cov(H, W) = -161.3$
$\rho(H, W) = -0.93$
(c)

Figure 3.9: Three examples of datasets with (a) positive, (b) no, and (c) negative covariance between height and weight. Covariance ($cov(H, W)$) and correlation ($\rho(H, W)$) values below the scatter plots.

height in meters rather than centimeters, its covariance with weight will change, but its Pearson correlation with weight will stay the same. The Pearson correlation can only take values between -1 (perfect anticorrelation) and $+1$ (perfect correlation).

The Pearson correlation coefficient is nice and it is used for many things – including in network science to predict links (Section 23.7), project bipartite networks (Section 26.2), estimate assortativity (Section 31.1), ... you get the idea. There are a couple of problems with the Pearson correlation. The first is that it only estimates how monotone a relationship is: this means that it wants variables to always vary together in the same general direction. Figure 3.10 shows a couple of non-monotone relationships – the classic "U" and "inverted U" shapes. In this case, even if there clearly is a relationship, Pearson will return zero and there isn't much you can do about it.

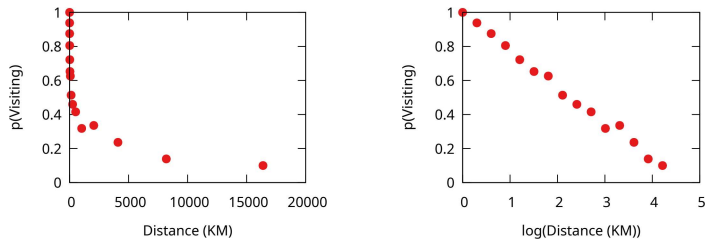


(a) (b)

Figure 3.10: Two non monotone relationships (a) U-shaped: if you never exercise you're frail and prone to injury, and if you exercise too much you have more chances to injury yourself. (b) Inverted U-shaped (or A-shaped): if there are no inventions innovation is hard, if there are too many innovations there's nothing left to innovate.

The second issue is that, even if the variables have a roughly monotone relationship, Pearson only measures it accurately if this relationship is *linear*. Pearson will give us the wrong idea if increases in one variable are associated with changes in another variable, but less and less so. For example, we visit our friends less often if they live further away from us, but moving 20 kilometers away if they lived next door before could make a big difference, whereas moving 20 extra kilometers away if they already lived in another country is not so meaningful.

One neat trick you can often do to fix this is to do a log-transformation



(a) (b)

Figure 3.11: Data with skewed distributions across various orders of magnitude. (a) Linear plot. (b) Same data, with the x axis log-transformed.

of your data before you calculate a correlation. In Figure 3.11(a) you can see a pair of variables with an non-linear relationship – the probability of deciding to visit a friend living at a given distance. Once you take the logarithm of the distance (Figure 3.11(b)), a linear relationship with the visit probability comes to the surface. Often, you can log both variables.

Most of the times you can stop here, but when this fails you still have one tool that allows to calculate correlations of data related in just about any way. As long as a change in one variable monotonically correspond to a change in the other (so, no U- or A-shapes where the direction of the relationship changes midway), you can summarize this with the Spearman rank correlation[21]. You do not assume anything at all about *how much* each variable changes as the other changes. The Spearman rank correlation is still normalized to take values between -1 and $+1$ with the exact same meaning as Pearson.

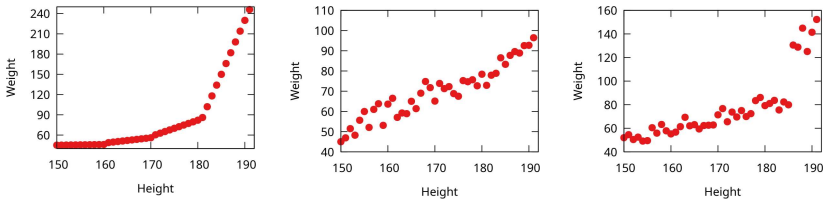
[21] Charles Spearman. The proof and measurement of association between two things. *Am J Psychol*, 15:72–101, 1904

How can you achieve this? It's actually rather simple. You still calculate a Pearson correlation. But rather than calculating it directly on the values of your variable, you do it on their *ranks*. The observation with the highest value becomes a 1, the second largest becomes a 2, and so on. Then, you calculate the Pearson correlation of these ranks. That is why this is called the Spearman *rank* correlation.

Figure 3.12 provides some examples. You can see what a monotonic but not linear relationship looks like (Figure 3.12(a)), that for data fitting the Pearson's assumptions Spearman returns comparable values (Figure 3.12(b)), and the robustness of Spearman to outliers (Figure 3.12(c)).

3.5 *Mutual Information*

Mutual Information (MI) is a key concept in information theory. It is another measure of how related two variables are. You can see it as a sort of a special correlation. Formally, MI quantifies how much information you obtain about one variable if you know the value of the other variable. For example, if we know how tall someone is,



$\rho(H,W) = 0.83$
$\rho_R(H,W) = 1$
(a)

$\rho(H,W) = 0.95$
$\rho_R(H,W) = 0.95$
(b)

$\rho(H,W) = 0.83$
$\rho_R(H,W) = 0.95$
(c)

Figure 3.12: Three pairs of variables showing the difference between the Pearson ($\rho(H,W)$) and the Spearman ($\rho_R(H,W)$) correlation coefficients.

does this allow us to make a better guess at their weight? How much better? This "amount of information" is usually measured in bits.

To understand MI, we need to take a quick crash course on information theory[22,23], which starts with the definition of information entropy. It is a lot to take in, but we will extensively use these concepts when it comes to link prediction and community discovery in Parts VII and X, thus it is a worthwhile effort.

Consider Figure 3.13. The figure contains a representation of a vector of six elements that can take three different values. The first thing we want to know is how many bits of information we need to encode its content.

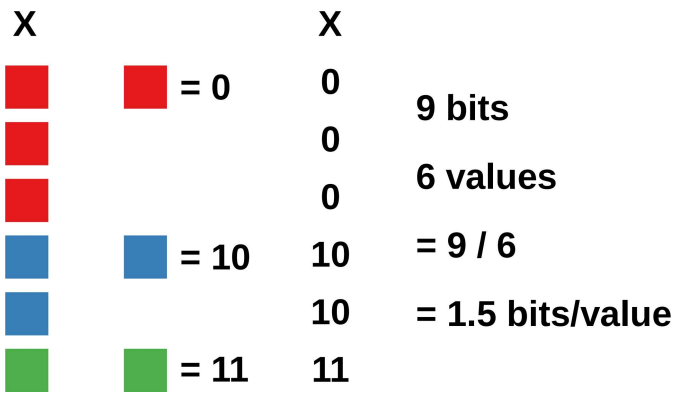


Figure 3.13: A simple example to understand information entropy. From left to right: the vector x has six elements taking three different values. We can encode each value with a sequence of zeros and ones. Doing so allows us to transmit x's six elements using nine bits of information. This means that the number of bits per value is 1.5.

We can be smart and use the shortest codes for the elements that appear most commonly, in this case the red square. Every time we see a red square, we encode it with a zero. If we don't see a red square, we write a one, which means that we need to look at a second bit to know whether we saw a blue or a green square. If it was a blue square, we write a zero, if it was green we write another one. With these rules, we can encode the original vector using nine bits, i.e. we use 1.5 bits per element.

This is close to – but not exactly – the definition of information entropy. In information entropy, we weigh the probability of an event by its logarithm[24,25].

[22] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999

[23] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003

[24] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948

[25] Andrei N Kolmogorov. Three approaches to the quantitative definition ofinformation'. *Problems of information transmission*, 1(1):1–7, 1965

Consider flipping a coin. Once you know the result, you obtain one bit of information. That is because there are two possible events, equally likely with a probability p of 50%. Generalizing to all possible cases, every time an event with probability p occurs, it gives you $-\log_2(p)$ bits of information for... reasons[26]. So, the total information of an event is the amount of information you get per occurrence times the probability of occurrence: $-p\log_2(p)$. Summed over all possible events i in x: $H_x = -\sum_i p_i \log_2(p_i)$, which is Shannon's information entropy – how many bits you need to encode the occurrence of all events.

Mutual information is defined for two variables. As I said, it is the amount of information you gain about one by knowing the other, or how much entropy knowing one saves you about the other. Consider Figure 3.14. It shows the relationship between two vectors, x and y. Note how y has equally likely outcomes: each color appears three times. However, if we observe a green square in x, we know with 100% confidence that the corresponding square in y is going to be purple. This means that, knowing x's values gives us information about y's value. Mathematically speaking, mutual information is the amount of information entropy shared by the two vectors.

It would take $-\log_2(1/3) \sim 1.58$ bits to encode y on its own (it is a random coin with three sides). However, knowing x's values makes you able to use the inference rules we see in Figure 3.14. Those rules are helpful: note how our confidence is almost always higher than 33%, which is the probability of getting y's color right without any further information. The rules will save you around 0.79 bits, which is x and y's mutual information.

The exact formulation of mutual information is similar to the formula of entropy:

[26] The amount of information of an event is a function that only depends on the probability p of the event to happen, e.g. $i_a = f(p_a)$ for event a. If we have two events, a and b, happening with probability p_a and p_b, the event c defined as a and b happening has probability $p_c = p_a p_b$. Now, each event also gives you an amount of information, namely i_a and i_b. When c happens, it means that both a and b happened, thus you got both pieces of information, or $i_c = i_a + i_b$. What we just said can be rewritten as $f(p_c) = f(p_a) + f(p_b)$, given the equation at the beginning. Since $p_c = p_a p_b$, then we can also rewrite the equation as $f(p_a p_b) = f(p_a) + f(p_b)$. The only function f that we can possibly plug into this equation maintaining it true is the logarithm. Since probabilities are lower than 1, the logarithm would be lower than zero, which would be nonsense – you cannot get negative information. Thus we take the negated logarithm: $i_a = -\log(p_a)$.

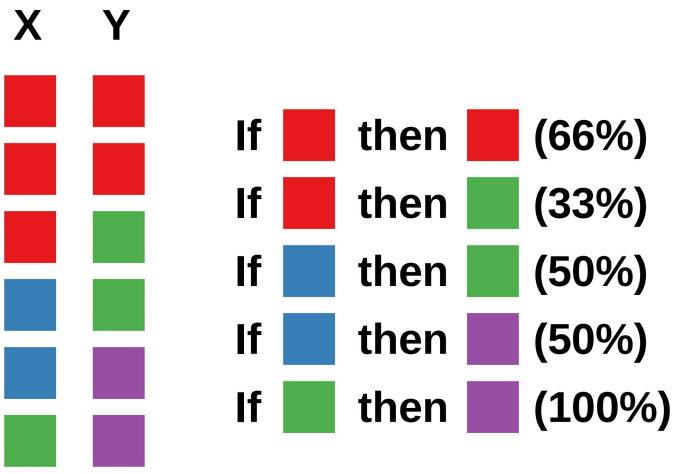


Figure 3.14: An illustration of what mutual information means for two vectors. Vector y has equal occurrences for its values (there is one third probability of any colored square). However, if we know the value of x we can usually infer the corresponding y value with a higher than chance confidence.

$$MI_{xy} = \sum_{j \in y} \sum_{i \in x} p_{ij} \log \left(\frac{p_{ij}}{p_i p_j} \right),$$

where p_{ij} is the joint probability of i and j. The meat of this equation is in comparing the joint probability of i and j happening with what you would expect if i and j were completely independent. If they are, then $p_{ij} = p_i p_j$, which means we take the logarithm of one, which is zero. But any time the happening of i and j is not independent, we add something to the mutual information. That something is the number of bits we save.

3.6 Summary

1. If you want to know what an average observation looks like in a variable, you might want to calculate the mean or the median.

2. The average observation will deviate from the mean: to estimate how far from the mean it will be on average, you can calculate the variance or the standard deviation.

3. Data can be skewed, meaning it distributes asymmetrically around the mean. There could be long or fat tails, depending how extreme and uncommon outliers are. The more skewed your data is, the more mean and median will disagree.

4. There are important distributions one should know: uniform, where all values are equally likely; normal, where values cluster around the mean; and various skewed distributions such as exponential, lognormal, and power law.

5. A cumulative distribution tells you the probability of observing a a values equal to or smaller than a given threshold.

6. The p-value tells you the probability of making a given observation under the null hypothesis (no change, no effect, nothing interesting is happening). A low p-value can prove the null hypothesis wrong but a high p-value does not prove the null hypothesis is true.

7. If you do multiple experiments, you need to correct the p-values you get otherwise you are going to misinterpret them.

8. When you have two variables, covariance tells you how much the two change together. Correlation coefficients are normalized covariances that do not change depending on the scale of the data. If your variables have a non-linear relationship, you need to use a correlation coefficient that can handle it.

9. Another approach to measure of how related two random variables are is mutual information. It tells you how many bits of information you gain about the status of one variable by knowing the other.

3.7 *Exercises*

1. Calculate the mean, median, and standard deviation of the two variables at `http://www.networkatlas.eu/exercises/3/1/data.txt` (one variable per column).

2. Make a scatter plot of the variables used in the previous exercise – with one variable on the x axis and the other on the y axis. Do you think that they are skewed or not? Calculate their skewness to motivate your answer.

3. Draw the mass function and the cumulative distribution of the following outcome probabilities:

Outcome	p
1	0.1
2	0.15
3	0.2
4	0.21
5	0.17
6	0.09
7	0.06
8	0.02

4. Which correlation coefficient should you use to calculate the correlation between the variables used in the exercise 2? Motivate your answer by calculating covariance, and the Pearson and Spearman correlation coefficients (and their p-values). Does the Spearman correlation coefficient agree with the Pearson correlation calculated on log-transformed values?

5. How many bits do we need to independently encode v_1 and v_2 from `http://www.networkatlas.eu/exercises/3/5/data.txt`? How much would we save in encoding v_1 if we knew v_2?

6
Basic Graphs

6.1 Simple Graphs

Every story should start from the beginning and, in this case, in the beginning was the graph[1,2,3,4]. To explain and decompose the elements of a graph, I'm going to use the recurrent example of social networks. The same graph can represent different networks: power grids, protein interactions, financial transactions. Hopefully, you can effortlessly translate these examples into whatever domain you're going to work.

Let's start by defining the fundamental elements of a social network. In society, the fundamental starting point is you. The person. Following Euler's logic that I discussed in the introduction, we want to strip out the internal structure of the person to get to a node. It's like a point in geometry: it's the fundamental concept, one that you cannot divide up into any sub-parts. Each person in a social network is a node – or vertex; in the book I'll treat these two terms as synonyms. We can also call nodes "actors" because they are the ones interacting and making events happen – or "entities" because sometimes they are not actors: rather than making things happen, things happen to them. "Actor" is a more specific term which is not an exact synonym of "node", but we'll see the difference between the two once we complicate our network model just a bit[5], in Section 7.2.

To add some notation, we usually refer to a graph as G. V indicates the set of G's vertices. Since V is the set of nodes, to refer to the number of nodes of a graph we use $|V|$ – some books will use n, but I'll try to avoid it. Throughout the book, I'll tend to use u and v to indicate single nodes.

So far, so good. However, you cannot have a society with only one individual. You need more than one. And, once you have at least two people, you need interactions between them. Again, following Euler, for now we forget about everything that happens in the internal structure of the communication: we only remember that an interaction is

[1] John Adrian Bondy, Uppaluri Siva Ramachandra Murty, et al. *Graph theory with applications*, volume 290. Citeseer, 1976

[2] Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001

[3] Reinhard Diestel. *Graph theory*. Springer Publishing Company, Incorporated, 2018

[4] Jonathan L Gross and Jay Yellen. *Graph theory and its applications*. CRC press, 2005

[5] The understatement of the century.

taking place. We will have plenty of time to make this model more complicated. The most common terms used to talk about interactions are "edge", "link", "connection" or "arc". While some texts use them with specific distinctions, for me they are going to be synonyms, and my preferred term will always be "edge". I think it's clearer if you always are explicit when you refer to special cases: sure, you can decide that "arc" means "directed edge", but the explicit formula "directed edge" is always better than remembering an additional term, because it contains all the information you need. (What the hell are "directed edges"? Patience, everything will be clear)

Again, notation. E indicates the set of G's edges and $|E|$ is the number of edges – some books will use m as a synonym for $|E|$. Usually, when talking about a specific edge one will use the notation (u, v), because edges are pairs of nodes – unless we complicate the graph model. Now we have a way to refer to the simplest possible graph model: $G = (V, E)$, with $E \subseteq V \times V$. A graph is a set of nodes and a set of edges – i.e. node pairs – established among those nodes.

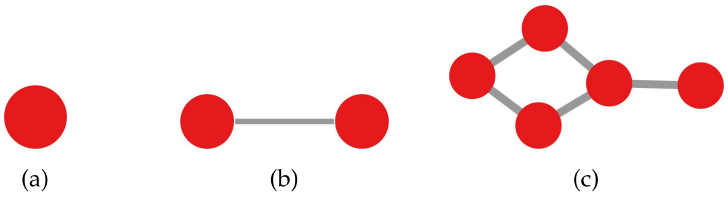


Figure 6.1: (a) A node. (b) An edge. (c) A simple graph.

We're going to talk about how to visualize networks much later in Part XIII, but it's better to introduce some visual elements now, otherwise how are we supposed to have figures before then? Nodes are usually represented as dots, or circles – Figure 6.1(a). Edges are lines connecting the dots – Figure 6.1(b). When all you have is nodes and edges, then you have a simple graph – Figure 6.1(c). Note that these visual elements are basic and widely used, but they are by no means the only way to visualize nodes and edges. In fact, when you want to convey a message about a network of non-trivial size, they're usually not a great idea.

The first famous graph in history is Euler's Königsberg graph, which I show in Figure 6.2. In the graph, each node represents a landmass and each edge represents a bridge connecting two landmasses. Since there were multiple bridges connecting the same landmasses, we have multiple edges between the same two nodes. This seemingly trivial fact is actually rather interesting.

"Simple graph" means *literally* simple: nothing more than nodes and edges – no attributes, no possibility of having multiple connections between the same two nodes. If you add any special feature, it's not a simple graph any more. Under this light, we discover that

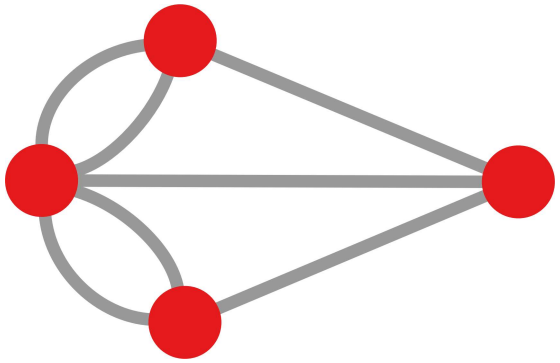
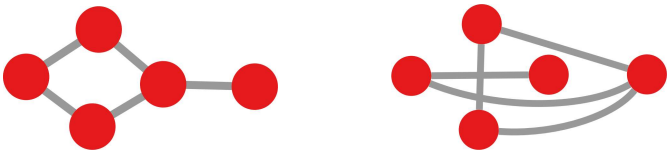


Figure 6.2: The famous Königsberg graph Euler used.

Euler's first graph wasn't simple after all. It allowed for parallel edges: multiple edges between the same two nodes. Euler's first graph was a multigraph. That's so non-standard that we're not even going to talk about it in this chapter: you'll have to wait for the next one, specifically for Section 7.2.

In our simple graph we also assume there are no self loops, which are edges connecting a node with itself. Our assumption is that we aren't psychopaths: everybody is friend with themselves, so we don't need to keep track of those connections.



(a) (b)

Figure 6.3: (a) A simple graph. (b) Its complement.

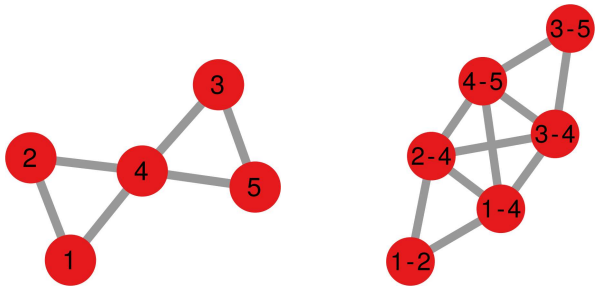
When you have a simple graph G, you can derive a series of special simple graphs related to G. For instance, you can derive the complement of G. This is equivalent to remove all of the original edges of G, and then connect all the unconnected pairs of nodes in G. Figure 6.3 shows an example.

This operation basically views G as a set of edges. If you take this perspective, you can define many operations on graphs as sets. Given two graphs G' and G'', you can calculate their union, intersection, and difference, which are the union, intersection, and difference of their edge sets. The union of G' and G'' is a graph G that has the edges found in either G' or G''; the intersection of G' and G'' is a graph G that has the edges found in both G' and G''; and the difference of G' and G'' is a graph G that has the edges found in G' but not in G''.

Another important special graph is the line graph[6,7]. The line graph of G represents each of G's edges as a node. Two nodes in the line graph are connected to each other if the edges they represent

[6] Hassler Whitney. Congruent graphs and the connectivity of graphs. *American Journal of Mathematics*, 54(1):150–168, 1932

[7] József Krausz. Démonstration nouvelle d'une théoreme de whitney sur les réseaux. *Mat. Fiz. Lapok*, 50(1):75–85, 1943



(a) (b)

Figure 6.4: (a) A graph. (b) Its linegraph version.

are attached to the same node in G. Figure 6.4 shows an example of line graph. We'll see how you can use line graphs to represent high order relationships in Chapter 34, to find overlapping communities in Chapter 38, and to estimate similarities between networks in Chapter 48.

6.2 *Directed Graphs*

Simple graphs are awesome. They allow you to represent a surprising variety of different complex systems. But they are not the end all be all of network theory. There are many phenomena out there that cannot be simply reduced to a set of nodes interacting through a set of edges. Sometimes you really need to complicate stuff. In this and in the next section we're going to see two ways to enhance the simple graph models. They all work in the same way: by slightly modifying the definition of an edge. We're going to see even more fundamental reworkings of the simple graph model in Chapter 7.

The first thing we will do is realizing that not all relations are reciprocal. The fact that I consider you as my friend – and I do, my dear reader – doesn't necessarily mean that you also consider me as your friend – wow, this book is getting very real very fast. We can introduce this asymmetry in the graph model. So far we said that (u, v) is an edge and we implicitly assumed that (u, v) is the same as (v, u). Directed graphs[8] are graphs for which $(u, v) \neq (v, u)$.

In a message passing game, (u, v) – or $u \to v$ – means that node u can pass a message to node v, but v cannot send it back to u. Directed graphs introduce all sorts of intricacies when it comes to finding paths in the network, a topic we're going to dissect in Chapter 10. The use of the arrow is a pretty straightforward metaphor to indicate the lack of reciprocity: relationships flow from the tail to the head of the arrow, not the other way around. It comes as no surprise, then, that we can use the arrow to indicate a directed edge, as we do in Figure 6.5(a). If E contains directed edges, we have a directed

[8] Frank Harary, Robert Zane Norman, and Dorwin Cartwright. *Structural models: An introduction to the theory of directed graphs*. Wiley, 1965

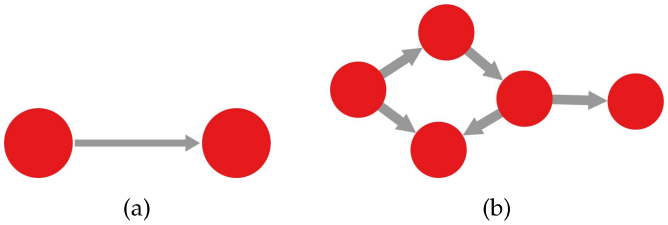


Figure 6.5: (a) A directed edge. (b) A directed graph.

graph – Figure 6.5(b).

Note that, in a directed graph (or digraph) representation, an edge always has a direction. If two nodes have a reciprocal relationship, convention dictates that we draw two directed edges pointing in the two directions, to make such relationship explicit.

In general, when you have a directed graph G, you can calculate its reverse graph by flipping all edge directions.

6.3 *Weighted Graphs*

Another way to make edges more interesting is realizing that two connections are not necessarily equally important in the network. One of the two might be much stronger than another. We are all familiar with the concepts of "best friend" and "Facebook friend". One is a much more tightly knit connection than the other.

For this reason, we can add weights to the edges[9,10]. A weight is simply an additional quantitative information we add to the connection. A possible notation could be (u, v, w): nodes u and v connect to each other with strength w. So our graph definition now changes to $G = (V, E, W)$, where W is our set of possible weights. W is practically always included in the set of real numbers, and most of the times in the set of real positive numbers – i.e. $W \subseteq R^+$. Now we have a weighted graph.

[9] Alain Barrat, Marc Barthelemy, Romualdo Pastor-Satorras, and Alessandro Vespignani. The architecture of complex weighted networks. *Proceedings of the national academy of sciences*, 101(11): 3747–3752, 2004a

[10] Mark EJ Newman. Analysis of weighted networks. *Physical review E*, 70 (5):056131, 2004a

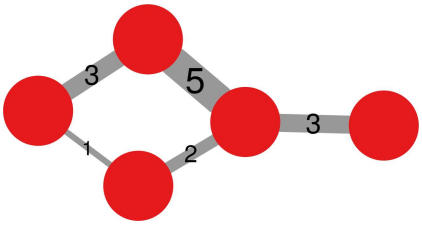


Figure 6.6: A weighted graph. The weight of the edge dictates its label and thickness.

Graphically, we usually represent the weight of a connection either by labeling the edge with its value, or simply by using visual elements such as the line thickness. I do both things in Figure 6.6.

Edge weights can be interpreted in two opposite ways, depending on what the network is representing. They can be considered the

proximity between the two nodes or their *distance*. This can and will influence the results of many algorithms you'll apply to your graph, so this semantic distinction matters. For instance, if you're looking for the shortest path (see Chapter 13) in a road network, your edge weight could mean different things. It could be a distance if it represents the length of the trait of road: longer traits will take more time to cross. Or it can be a proximity: it could be the throughput of the trait of road in number of cars per minute that can pass through it – or the number of lanes. If the weight is a distance, the shortest path should avoid high edge weights. If the weight is a proximity, it should do its best to include them.

To sum up, "proximity" means that a high weight makes the nodes closer together; e.g. they interact a lot, the edge has a high capacity. "Distance" means that a high weight makes the nodes further apart; e.g. it's harder or costly to make the nodes interact.

Edge weights don't have to be positive. Nobody says nodes should be friends! Examples of negative edge weights can be resistances in electric circuits or genes downregulating other genes. This observation is the beginning of a slippery slope towards signed networks, which is a topic for another time (namely, for Section 7.2, if you want to jump there).

The network in Figure 6.6 has nice integer weights. In this case, the edge weights are akin to counts. For instance, in a phone call network, it could be the number of times two people have called each other. Unfortunately, not all weighted networks look as neat as the example in Figure 6.6. In fact, most of the weighted networks you might work with will have continuous edge weights. In that case, many assumptions you can make for count weights won't apply – for instance when filtering connections, as we will see in Chapter 27.

By far, the most common case is the one of correlation networks. In these networks, the nodes aren't really interacting directly with one another. Instead, we are connecting nodes because they are similar to each other, for some definition of similarity. For instance, we could connect brain areas via cortical thickness correlations[11], or currencies according to their exchange rate[12], or correlating the taxa presence in different biological communities[13].

These cases have more or less the same structure. I provide an example in Figure 6.7. In this case, nodes are numerical vectors, which could represent a set of attributes, for instance. We calculate a correlation between the vectors, or some sort of attribute similarity – for instance mutual information (Section 3.5). We then obtain continuous weights, which typically span from -1 to 1. And, since every pair of nodes have a similarity (because any two vectors can be correlated, minus extremely rare degenerate cases), every node is connected to

[11] Boris C Bernhardt, Zhang Chen, Yong He, Alan C Evans, and Neda Bernasconi. Graph-theoretical analysis reveals disrupted small-world organization of cortical thickness correlation networks in temporal lobe epilepsy. *Cerebral cortex*, 21(9):2147–2157, 2011

[12] Takayuki Mizuno, Hideki Takayasu, and Misako Takayasu. Correlation networks among currencies. *Physica A: Statistical Mechanics and its Applications*, 364:336–342, 2006

[13] Jonathan Friedman and Eric J Alm. Inferring correlation networks from genomic survey data. *PLoS computational biology*, 8(9):e1002687, 2012

every other node. So, when working with similarity networks, you will have to filter your connections somehow, a process we call "network backboning" which is far less trivial that it might sound. We will explore it in Chapter 27.

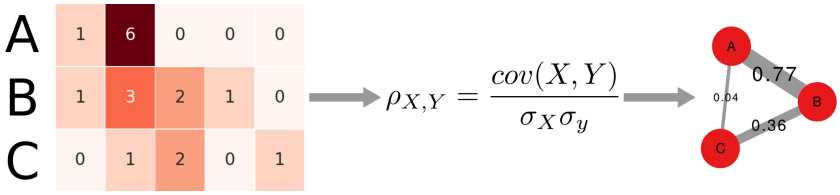


Figure 6.7: A typical workflow for correlation networks: (left to right) from nodes represented as some sort of vectors, to a graph with a similarity measure as edge weigth.

6.4 *Network Types*

Now that you know more about the various features of different network models, we can start looking at different types of networks. I'm going to use a taxonomy for this section. I find this way of organizing networks useful to think about the objects I work with.

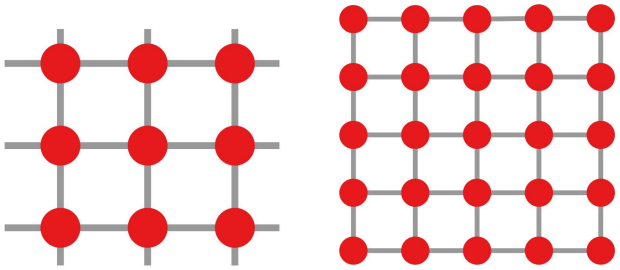
Simple Networks

The first important distinction between network types is between *simple* and *complex* networks. A simple network is a network we can fully describe analytically. Its topological features are exact and trivial. You can have a simple formula that tells you everything you need to know about it. In complex networks that is not possible, you can only use formulas to approximate their salient characteristics.

The difference between a simple network and a complex network is the same between a sphere and a human being. You can fully describe the shape of a sphere with a few formulas: its surface is $4\pi r^2$, its volume is $\frac{4}{3}\pi r^3$. If you know r you know everything you need to know about the sphere. Try to fully describe the shape of a human being, internal organs included, starting from a single number. Go on, I have time.

What do simple networks look like? I think the easiest example conceivable is a square lattice. This is a regular grid, in which each node is connected to its four nearest neighbors. Such lattice can either span indefinitely (Figure 6.8(a)), or it can have a boundary (Figure 6.8(b)). Their fundamental properties are more or less the same. Knowing this connection rule that I just stated allows you to picture any lattice ever. That is why this is a simple topology.

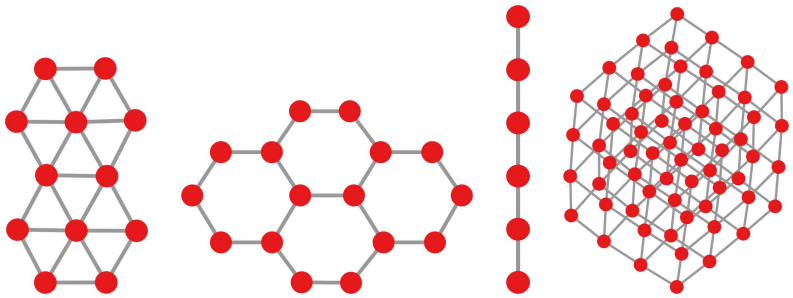
Regular lattices can come in many different shapes besides square, for instance triangular (Figure 6.9(a)) or hexagonal (Figure 6.9(b)).



(a) (b)

Figure 6.8: (a) An infinite lattice without boundaries. (b) A finite lattice with 25 nodes and 40 edges.

They also don't necessarily have to be two dimensional as the examples I made so far: you can have 1D (Figure 6.9(c)) and 3D (Figure 6.9(d)) lattices – the latter might be a bit hard to see, but it is a cube of with four nodes per side.



(a) (b) (c) (d)

Figure 6.9: Different lattice types. (a) Triangular. (b) Hexagonal. (c) One dimensional. (d) Three dimensional cube.

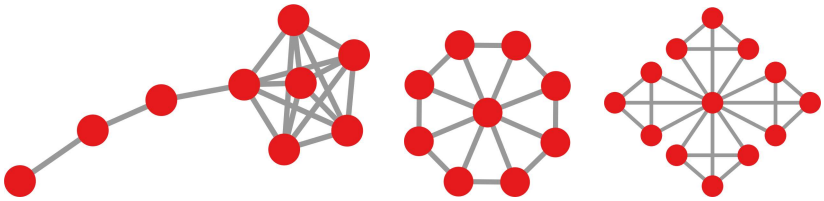
Even if deceptively simple, lattices can be extremely useful and are used as starting point for many advanced tasks. For instance, they are at the basis of the small-world graph generator (Section 17.2) and of our understanding of epidemic spread in society (Chapter 20).

Lattices are not the only simple network out there. There is a wide collection of other network types. These are usually developed as the simplest illustrative examples for explaining new problems or algorithms. A few of my favorites (yes, I'm the kind of person who has favorite graphs) are the lollipop graph[14] (a set of n nodes all connected to each other plus a path of m nodes shooting out of it, Figure 6.10(a)), the wheel graph (which has a center connected to a circle of m nodes, Figure 6.10(b)), and the windmill graph (a set of n graphs with m nodes and all connections to each other, also all connected to a central node, Figure 6.10(c)). Once you figure out what rule determines each topology, you can generate an arbitrary set of arbitrary size of graphs that all have the same properties.

[14] Graham Brightwell and Peter Winkler. Maximum hitting time for random walks on graphs. *Random Structures & Algorithms*, 1(3):263–276, 1990

Complex Networks

If simple networks were the only game in town, this book would not exist. That is because, as I said, you can easily understand all



(a) (b) (c)

Figure 6.10: Different simple networks. (a) Lollipop graph. (b) Wheel graph. (c) Windmill graph.

their properties from relatively simple math. That is not the case when the network you're analyzing is a complex network. Complex networks model complex systems: systems that cannot be fully understood if all you have is a perfect description of all their parts. The interactions between the parts let global properties emerge that are not the simple sum of local properties. There isn't a simple wiring rule and, even knowing all the wiring, some properties can still take you by surprise.

Personally, I like to divide complex networks into two further categories: complex network *with fundamental metadata* and *without fundamental metadata*. We saw that you can have edge metadata, the direction and weight. In Chapter 7 we'll see you can have even more, attached to both nodes and edges. The difference I'm trying to make is that, if the metadata are fundamental, they change the way you interpret some or all the metadata themselves.

To understand *non-fundamental* metadata, think about the fact that social networks, infrastructure networks, biological networks, and so on, model different systems and have different metadata attached to their nodes and edges. They can be age/gender, activation types, up- and down-regulation. However, the algorithms and the analyses you perform on them are the same, regardless of what the networks represent. They have nodes and edges and you treat them as such. You perform the Euler operation: you forget about all that is unnecessary so you can apply standardized analytic steps.

That is emphatically not true for networks with *fundamental* metadata. In that case, you need to be aware of what the metadata represent, because they change the way you perform the analysis and you interpret the results. A few examples:

- *Affiliation networks*. These are networks that, for instance, connect individuals to the groups they belong to. Here it is clear that one node type includes the other – the group includes the individual. This is fundamentally different when you have node types at an equal level – for instance if you connect people to the products they buy.
- *Interdependent networks*. These usually model some sort of physical

system, for instance computers connected to the power plants they control. Edges express the dependencies of one node on the other they connect to. In this case, the removal of one node in one layer has immediate and non-trivial repercussions on all the layers depending into it, often with catastrophic consequences (see Section 22.4) – which may not be true for other networks.

- *Correlation networks*. We saw a glimpse of these networks when we looked at weighted graphs. Here we have constraints on the edge weights, which can also be negative. The interpretation of such edge weights is different from what you would have in regular weighted networks. For instance, edges with very low weights are important here, because a strong negative correlation is interesting, even if its value (-1) is lower than no correlation at all (0).

A special mention for this class of networks should go to Bayesian networks[15,16,17]. In a Bayesian network, each node is a variable and directed edges represent dependencies between variables. If knowing something about the status of variable u gives you information about the status of variable v, then you will connect u to v with a directed (u, v) edge.

In the classical example, you might have three variables: the probability of raining, the probability of having the sprinklers on, and the probability that the grass is wet. Clearly, rain and sprinklers both might cause the grass to be wet, so the two variables point to them. Rain also might influence the sprinklers, because the automatic system to save water will not turn them on when it's raining, since it would be pointless. Obviously, the fact that the sprinklers are on will have no effect on whether it will rain or not.

We can model this system with the simple Bayesian network in Figure 6.11(a) and the corresponding conditional probability tables in Table 6.11(b). Bayesian networks are usually the output of a machine learning algorithm. The algorithm will learn the best network that fits the observations. Then, you can use the network to predict the most likely probability of the state of a variable given a new observation of a subset of variables.

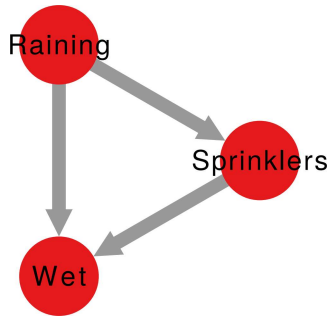
Simple examples like this might seem boring, but when you start having hundreds of variables you can find interesting patterns by applying some of the techniques you will learn later on. For instance, you might discover sets of variables that are independent of each other, even if, at first glance, it might be difficult to tell.

A not so distant relative of Bayesian networks are neural networks, the bread and butter of machine learning these days. Notwithstanding their amazing – and, sometimes, mysterious – power, neural networks are actually much more similar to simple networks than to

[15] Finn V Jensen et al. *An introduction to Bayesian networks*, volume 210. UCL press London, 1996

[16] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine learning*, 29(2-3): 131–163, 1997

[17] Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *IJCAI*, volume 99, pages 1300–1309, 1999



(a)

	Rain	
	T	F
	0.20	0.80

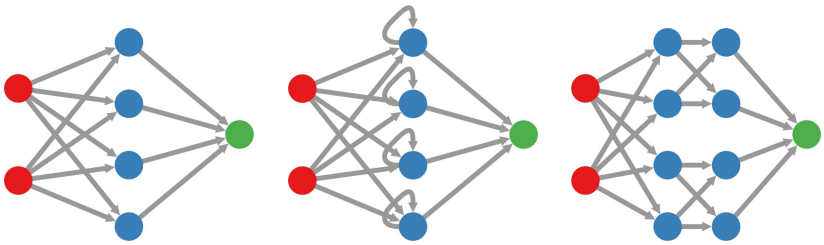
	Sprinkler	
Rain	T	F
T	0.01	0.99
F	0.20	0.80

		Wet	
Rain	Sprinkler	T	F
T	T	0.99	0.01
T	F	0.98	0.02
F	T	0.97	0.03
F	F	0.01	0.99

(b)

Figure 6.11: (a) A Bayesian network. (b) The conditional probability tables for the node states. The tables are referring to, from top to bottom: Rain, Sprinkler, Wet.

complex ones. Differently from Bayesian networks, the wiring rules of neural networks – of which I show some examples in Figure 6.12 – are usually rather easy to understand.



(a) Feedforward. (b) Recurrent. (c) Modular.

Figure 6.12: Different neural networks. The node color determines the layer type: input (red), hidden (blue), output (green).

The way they work is that the weight on each node of the output layer is the answer the model is giving. This weight is directly dependent on a combination of the weights of the nodes in the last hidden layer. The contribution of each hidden node is proportional to the weight of the edge connecting it to the output node. Recursively, the status of each node in the hidden layer is a combination of all its incoming connections – combining the edge weight to the node weight at the origin. The first hidden layer will be directly dependent on the weights of the nodes in the input layer, which are, in turn, determined by the data.

What the model does is simply finding the combination of edge weights causing the output layer's node weights to maximize the desired quality function.

6.5 Summary

1. The mathematical representation of a network is the graph: a collection of nodes – the actors of the network –, and edges – the connections among those actors. In a simple graph, no additional feature can be added, and there is only one edge between a pair of nodes.

2. If connections are not symmetric, meaning that if you consider me your friend I don't necessarily consider you mine, then we have directed graphs. In directed graphs, edges have a direction so relations flow one way, unless there is a reciprocal edge pointing back.

3. In weighted graphs, connections can be more or less strong, indicated by the weight of the edge, a numerical quantity. It doesn't have to be a discrete number, nor necessarily positive: for instance in correlation networks you can have negative continuous weights.

4. Weights can have two meanings: proximity – the edge is the strength of a friendship –, or distance – the edge is a cost to pay to cross from one node to another. Different semantics imply that some algorithms' results should be interpreted differently.

5. Simple networks are networks whose topology can be fully described with simple rules. For instance, in regular lattices you place nodes uniformly in a space and you connect them with their nearest neighbors.

6.6 Exercises

1. Calculate $|V|$ and $|E|$ for the graph in Figure 6.1(c).

2. Mr. A considers Ms. B a friend, but she doesn't like him back. She has a reciprocal friendship with both C and D, but only C considers D a friend. D has also sent friend requests to E, F, G, and H but, so far, only G replied. G also has a reciprocal relationship with A. Draw the corresponding directed graph.

3. Draw the previous graph as undirected and weighted, with the weight being 2 if the connection is reciprocal, 1 otherwise.

4. Draw a correlation network for the vectors in `http://www.networkatlas.eu/exercises/6/4/data.txt`, by only drawing edges with positive weights, ignoring self loops.

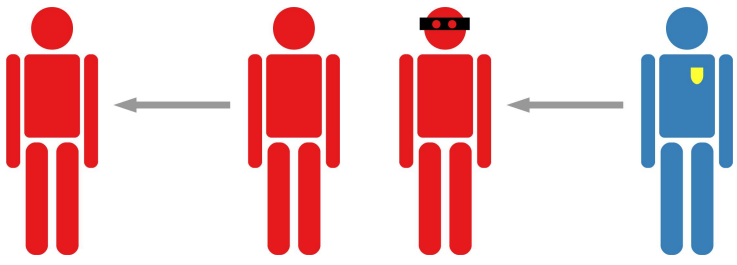
7
Extended Graphs

The world of simple graphs is... well... simple. The only thing complicating it a bit so far was adding some information on the edges: whether they are asymmetric – meaning $(u, v) \neq (v, u)$ – and whether they are strong or weak. Unfortunately, that's not enough to deal with everything reality can throw your way. In this chapter, I present even more graph models, which go beyond the simple addition of edge information.

7.1 Bipartite Graphs

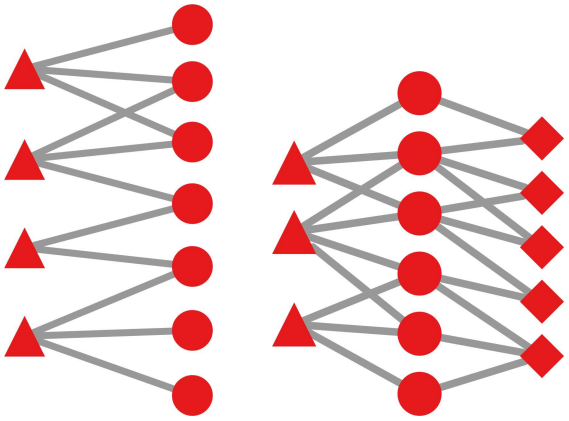
So far we have talked about networks in which relations run between peers: nodes are all the same to us. But nodes might belong to two distinct classes. And connections can only be established between members of different classes. Figure 7.1 provides an example. In a social network without node attributes nor types, anybody can be friend with anybody else and there isn't much to distinguish two nodes. But if we want to connect cops with the thieves they catch, then we are establishing additional connecting rules. Thieves don't catch each other. And, hopefully, cops aren't thieves. Another example could be connecting workers to the buildings hosting their offices.

Stripping down the model to a minimum, bipartite networks are



(a) (b)

Figure 7.1: (a) A simple graph representing a social network with no additional constraints. (b) A cop-thief bipartite network: nodes can be either a cop or a thief, and cops can only catch (connect to) thieves.



(a) (b)

Figure 7.2: (a) An example of a bipartite network. (b) An example of a tripartite network.

networks in which nodes must be part of either of two classes (V_1 and V_2) and edges can only be established between nodes of unlike type[1,2]. Formally, we would say that $G = (V_1, V_2, E)$, and that E can only contain edges like (v_1, v_2), with $v_1 \in V_1$ and $v_2 \in V_2$. Figure 7.2(a) depicts an example.

Bipartite networks are used for countless things, connecting: countries to the products they export[3], hosts to guest in symbiotic relationships[4], users to the social media items they tag[5], bank-firm relationships in financial networks[6], players-bands in jazz[7], listener-band in music consumption[8], plant-pollinators in ecosystems[9], and more. You get the idea. Bipartite networks pop up everywhere.

However, by a curious twist of fate, the algorithms able to work directly on bipartite structures are less studied than their non-bipartite counterparts. For instance, for every community discovery algorithm that works on bipartite networks you have a hundred working on non-bipartite ones. The distinction is important, because the standard assumptions of non-bipartite community discovery do not hold in bipartite networks, as we will see in Part X.

Why would that be the case? Because practically everyone who works on bipartite networks projects them. Most of the times, you are interested only in one of the two node types. So you create a unipartite version of the network connecting all nodes in V_1 to each other, using some criteria to make the V_2 count. The trivial way is to connect all V_1 nodes with at least a common V_2 neighbor. This is so widely done and so wrong that I like to call it the Mercator bipartite projection, in honor of the most used and misunderstood map projection of all times. We'll see in Chapter 26 why that's not very smart, and the different ways to do a better job.

Why stopping at bipartite? Why not go full n-partite? For instance, a paper I cited before actually builds a tri-partite network (Figure

[1] Armen S Asratian, Tristan MJ Denley, and Roland Häggkvist. *Bipartite graphs and their applications*, volume 131. Cambridge University Press, 1998

[2] Jean-Loup Guillaume and Matthieu Latapy. Bipartite structure of all complex networks. *Information processing letters*, 90:Issue–5, 2004

[3] César A Hidalgo and Ricardo Hausmann. The building blocks of economic complexity. *Proceedings of the national academy of sciences*, 106(26):10570–10575, 2009

[4] Brian D Muegge, Justin Kuczynski, Dan Knights, Jose C Clemente, Antonio González, Luigi Fontana, Bernard Henrissat, Rob Knight, and Jeffrey I Gordon. Diet drives convergence in gut microbiome functions across mammalian phylogeny and within humans. *Science*, 332(6032):970–974, 2011

[5] Renaud Lambiotte and Marcel Ausloos. Collaborative tagging as a tripartite network. In *International Conference on Computational Science*, pages 1114–1117. Springer, 2006

[6] Luca Marotta, Salvatore Micciche, Yoshi Fujiwara, Hiroshi Iyetomi, Hideaki Aoyama, Mauro Gallegati, and Rosario N Mantegna. Bank-firm credit network in japan: an analysis of a bipartite network. *PloS one*, 10(5): e0123079, 2015

[7] Pablo M Gleiser and Leon Danon. Community structure in jazz. *Advances in complex systems*, 6(04):565–573, 2003

[8] Renaud Lambiotte and Marcel Ausloos. Uncovering collective listening habits and music genres in bipartite networks. *Physical Review E*, 72(6):066107, 2005

7.2(b) depicts an example): users connect to the social media they tag and with the tags they use. However, the gains you get from a more precise data structure quickly become much lower than the added complexity of the model. Even tripartite networks are a rarity in network science. A couple of examples are the recipe-ingredient-compound structure of the flavor network[10], or the aid organization-country-issue structure[11].

7.2 *Multilayer Graphs*

In this section I describe models you can use to analyze multilayer networks. This should not be confused with the similarly-sounding, but actually completely different, multilevel network analysis[12]. This is a whole different way to analyze social network data using multilevel analysis, of which I know little and I will attempt to cover in future versions of this book.

One-to-One

Traditionally, network scientists try to focus on one thing at a time. If they are interested in analyzing your friendship patterns, they will choose one network that closely approximates your actual social relations and they will study that. For instance, they will download a sample of the Facebook graph. Or they will analyze tweets and retweets.

However, in some cases, that is not enough to really grasp the phenomenon one wants to study. If you want to predict a new connection on Facebook, something happening in another social media might have influenced it. Two people might have started working in the same company and thus first connected on Linkedin, and then became friends and connected on Facebook. Such scenario could not be captured by simply looking at one of the two networks. Network scientists invented multilayer networks[13,14,15,16,17,18] to answer this kind of questions.

There are two ways to represent multilayer networks. The simpler is to use a multigraph. Remember Euler's parallel edges in the Königsberg graph from Figure 6.2? That's what makes a multigraph. Differently from a simple graph (Figure 7.3(a)), in which every pair of nodes is forced to have at most one edge connecting them, in a multigraph (Figure 7.3(b)) we allow an arbitrary number of possible connections.

If that is all, there wouldn't be much difference between multigraphs and weighted networks. If all parallel edges are the same, we could have a single edge with a weight proportional to the number

[9] Colin Campbell, Suann Yang, Réka Albert, and Katriona Shea. A network model for plant–pollinator community assembly. *Proceedings of the National Academy of Sciences*, 108(1):197–202, 2011

[10] Yong-Yeol Ahn, Sebastian E Ahnert, James P Bagrow, and Albert-László Barabási. Flavor network and the principles of food pairing. *Scientific reports*, 1:196, 2011

[11] Michele Coscia, Ricardo Hausmann, and César A Hidalgo. The structure and dynamics of international development assistance. *Journal of Globalization and Development*, 3(2):1–42, 2013a

[12] Emmanuel Lazega and Tom AB Snijders. *Multilevel network analysis for the social sciences: Theory, methods and applications*, volume 12. Springer, 2015

[13] Mikko Kivelä, Alex Arenas, Marc Barthelemy, James P Gleeson, Yamir Moreno, and Mason A Porter. Multilayer networks. *Journal of complex networks*, 2 (3):203–271, 2014

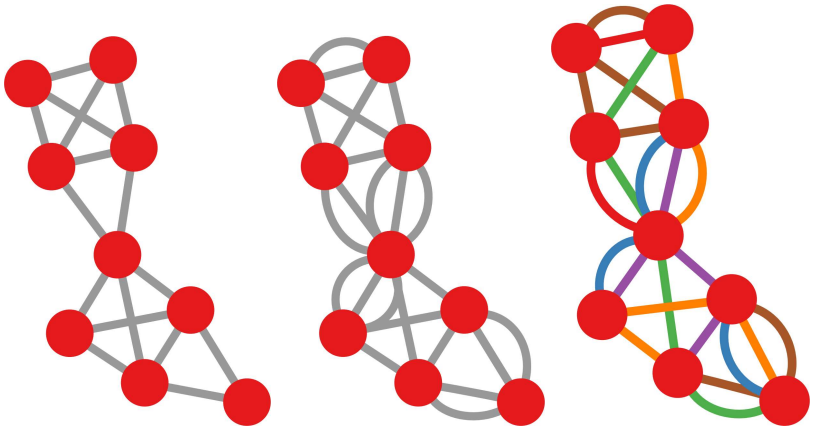
[14] Manlio De Domenico, Albert Solé-Ribalta, Emanuele Cozzo, Mikko Kivelä, Yamir Moreno, Mason A Porter, Sergio Gómez, and Alex Arenas. Mathematical formulation of multilayer networks. *Physical Review X*, 3(4):041022, 2013

[15] Stefano Boccaletti, Ginestra Bianconi, Regino Criado, Charo I Del Genio, Jesús Gómez-Gardenes, Miguel Romance, Irene Sendina-Nadal, Zhen Wang, and Massimiliano Zanin. The structure and dynamics of multilayer networks. *Physics Reports*, 544(1):1–122, 2014

[16] Michele Berlingerio, Michele Coscia, Fosca Giannotti, Anna Monreale, and Dino Pedreschi. Multidimensional networks: foundations of structural analysis. *WWW*, 16(5-6):567–593, 2013a

[17] Mark E Dickison, Matteo Magnani, and Luca Rossi. *Multilayer social networks*. Cambridge University Press, 2016

[18] Matteo Magnani and Luca Rossi. The ml-model for multi-layer social networks. In *ASONAM*, pages 5–12. IEEE, 2011



(a) (b) (c)

Figure 7.3: (a) A simple graph. (b) A multigraph, with multiple edges between the same node pairs. (c) A multilayer network, where each edge has a type (represented by its color).

of connections between the two nodes. However, in this case, we can add a "type" to each connection, making them *qualitatively* different: one edge type for Facebook, one for Twitter, one for Linkedin (Figure 7.3(c)), etc.

In practice, every edge type – or label – represents a different layer of the network. A pair of nodes can establish a connection in any layer, even at the same time. Each layer is a simple graph. In this book – and generally in computer science – the most used notation to indicate a multilayer network is $G = (V, E, L)$. V and E are the sets of nodes and edges, as usual. L is the set of layers – or labels. An edge is now a triple $(u, v, l) \in E$, with $u, v \in V$ as nodes, and $l \in L$ as the layer. This might seem similar to the notation used for weighted edges – which was (u, v, w). The key difference is that w is a quantitative information, while l is a qualitative one: a class, a type. We can make the two co-exist in weighted multigraphs, by specifying an edge as (u, v, l, w).

The model that we introduce in Figure 7.3(c) is but the simplest way to represent multilayer networks. This strategy rests on the assumption that there is a one-to-one node mapping between the layers of the network. In other words, the entities in each layer are always the same: you are always you, whether you manage your Facebook account or your Linkedin one. Such simplified multilayer networks are sometimes called multiplex networks.

Studies have shown how layers in a multiplex network could be complementary[19]. This means that a single layer in the network might not show the typical statistical properties you would expect from a real world network – the types of things we'll see in this book. However, once you stack enough layers one on top of the other, the resulting network does indeed conform to our structural expectations.

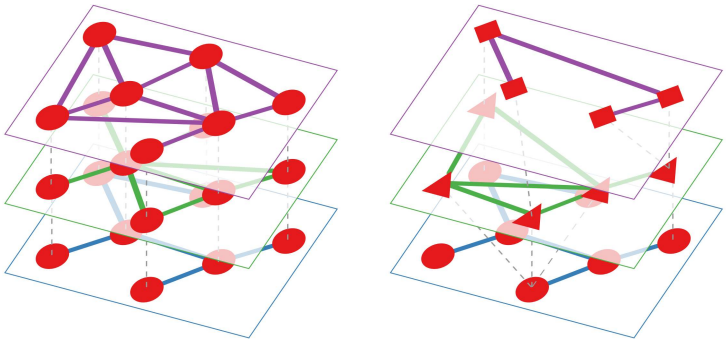
[19] Alessio Cardillo, Jesús Gómez-Gardenes, Massimiliano Zanin, Miguel Romance, David Papo, Francisco Del Pozo, and Stefano Boccaletti. Emergence of network features from multiplexity. *Scientific reports*, 3:1344, 2013

In other words, multilayer networks have *emerging* properties.

Multiplex networks, don't necessarily cover all application scenarios: sometimes a node in one layer can map to multiple nodes – or none! – in another. This is what we turn our attention to next.

Many-to-Many

To fix the insufficient power of multiplex networks to represent true multilayer systems we need to extend the model. We introduce the concept of "interlayer coupling". In this scenario, the node is split into the different layers to which it belongs. In this case, your identity includes multiple personas: you are the union of the "Facebook you", the "Linkedin you", the "Twitter you". Figure 7.4(a) shows the visual representation of this model: each layer is a slice of the network. There are two types of edges: the intra-layer connections – the traditional type: we're friends on Facebook, Linkedin, Twitter –, and the inter-layer connections. The inter-layer edges run between layers, and their function is to establish that the two nodes in the different layers are really the same node: they are *coupled* to – or *dependent* on – each other.



(a) (b)

Figure 7.4: The extended multilayer model. Each slice represents a different layer of the network. Dashed grey lines represent the inter-layer coupling connections. (a) A multilayer network with trivial one-to-one coupling. (b) A multilayer network with complex interlayer coupling.

Formally, our network is $G = (V, E, L, C)$. V is still the set of nodes, but now we split the set of edges in two: E is the set of classical edges, the intra-layer one – connections between different people on a platform –; and C is the set of coupling connections, the inter-layer one, denoting dependencies between nodes in different layers.

Having a full set of coupling connections enables an additional degree of freedom. We can now have nodes in one layer expressing coupling with multiple nodes in other layers. In our social media case, we are now allowing you to have multiple profiles in one platform that still map on your single profile in another. For instance, you can run as many different Twitter accounts as you want, and they are still coupled with your Facebook account. To get a visual sense on what this means, you can look at Figure 7.4(b).

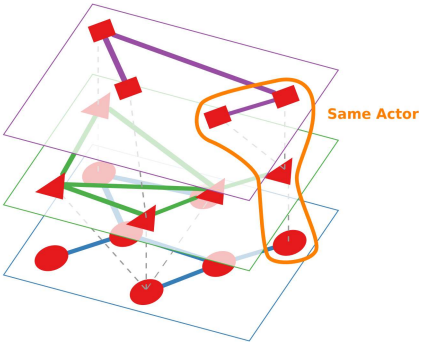
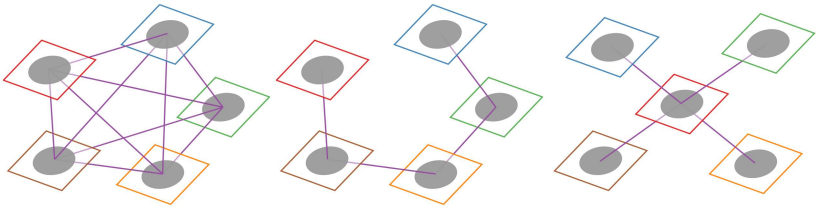


Figure 7.5: An actor in a many-to-many coupled multilayer network. The orange outline surrounds nodes with coupling edges connecting them.

This new freedom comes to a cost. While in the one-to-one mapping it is easy to identify a node among layers, because all identities of a node are concentrated in a single point in a layer, in the many-to-many coupling that is not true any more. So we introduce the term "actor", which is the entity behind all the multiple identities across layers and within a layer. In practice, the actor is a connected component (see Section 10.4), when only considering inter-layer couplings as the possible edges. If my three Twitter profiles all refer to the same person, with maybe two Flickr accounts and one Facebook profile, all these identities belong to the same actor: me. Figure 7.5 should clarify this definition.

Note that there can be many ways to establish inter-layer couplings between the different nodes belonging to the same actor. As far as I know, when analyzing networks people usually use a "cliquey" approach: every node belonging to the same actor is connected to every other node as, for instance, in Figure 7.6(a). This effectively creates a clique of inter-layer coupling connections – for more information about what a clique is, see Section 12.3.



(a) Clique (b) Chain (c) Star

Figure 7.6: Different coupling flavors for your multilayer networks. Showing a network with a single actor and a single node per actor per layer (represented by the border-colored polygon). I color the coupling edges in purple.

However, this is usually too cumbersome to draw. So, for illustration purposes, the convention is to use a "chainy" approach (Figure 7.6(b)): you sort your layers somehow, and you simply place a line representing your coupling connections piercing through the layers. We don't really have to stop there. One could imagine using a "starry" approach: defining one layer as the center of the system, and connecting all nodes belonging to that actor to the node in the cen-

tral layer. To see what I mean, look at Figure 7.6(c). Using different coupling flavors can be useful for computational efficiency: when you start having dozens or even hundreds of layers, creating cliques of layers can add a significant overhead.

Such many-to-many layer couplings are often referred to in the literature as "networks of networks", because each layer can be seen as a distinct network, and the interlayer couplings are relationships between different networks[20,21,22].

[20] Jacopo Iacovacci, Zhihao Wu, and Ginestra Bianconi. Mesoscopic structures reveal the network between the layers of multiplex data sets. *Physical Review E*, 92 (4):042806, 2015

[21] Gregorio D'Agostino and Antonio Scala. *Networks of networks: the last frontier of complexity*, volume 340. Springer, 2014

[22] Dror Y Kenett, Matjaž Perc, and Stefano Boccaletti. Networks of networks–an introduction. *Chaos, Solitons & Fractals*, 80:1–6, 2015

Aspects

Do you think we can't make this even more complicated? Think again. These aren't called "complex networks" by accident. To fully generalize multilayer networks, adding the many-to-many interlayer coupling edges is not enough. To see why that's the case, consider the fact that, up to this point, I considered the layers in a multilayer network as interchangeable. Sure, they represent different relationships – Facebook friendship rather than Twitter following – but they are fundamentally of the same type. That's not necessarily the case: the network can have multiple aspects.

For instance, consider time. We might not be Facebook friends now, but that might change in the future. So we can have our multilayer network at time t and at time $t + 1$. These are two aspects of the same network. All the layers are present in both aspects and the edges inside them change. Another classical example is a scientific community. People at a conference interact in different ways – by attending each other talks, by chatting, or exchanging business cards – and can do all of those things at different conferences. The type of interaction is one aspect of the network, the conference in which it happens is another.

I can't hope to give you here an overview of how many new things this introduces to graph theory. So I'm referring you to a specialized book on the subject[23].

[23] Ginestra Bianconi. *Multilayer Networks: Structure and Function*. Oxford University Press, 2018

Signed Networks

Signed networks are a particular case of multilayer networks. Suppose you want to buy a computer, and you go online to read some reviews. Suppose that you do this often, so you can recognize the reviewers from past reviews you read from them. This means that you might realize you do not trust some of them and you trust others. This information is embedded in the edges of a signed network: there are positive and negative relationships.

Signed networks are not necessarily restricted to either a single positive or a single negative relationship – e.g. "I trust this person"

or "I don't trust this person". For instance, in an online game, you can have multiple positive relationships like being friend or trading together; and multiple reasons to have a negative relationship, like fighting each other, or putting a bounty on each other heads.

A key concept in signed networks is the one of structural balance. Since this is mostly related to the link prediction problem, I expand on this in Section 24.1.

Positive and negative relationships have different dynamics. For instance, in a seminal study looking at interactions between players in a massively multiplayer online game[24], the authors studied the different degree distributions (Section 9.2) for each type of relationship. They uncovered that positive relationships have a marked exponential cutoff, while negative relationships don't. You'll become more accustomed to what a degree distribution is and all the lingo related to it in Chapter 9. For now, the meaning of what I just said is: there is a limit to the number of people you can be friends with, but there is no limit to the number of people that can be mad at you.

[24] Michael Szell, Renaud Lambiotte, and Stefan Thurner. Multirelational organization of large-scale social networks in an online world. *Proceedings of the National Academy of Sciences*, 107 (31):13636–13641, 2010

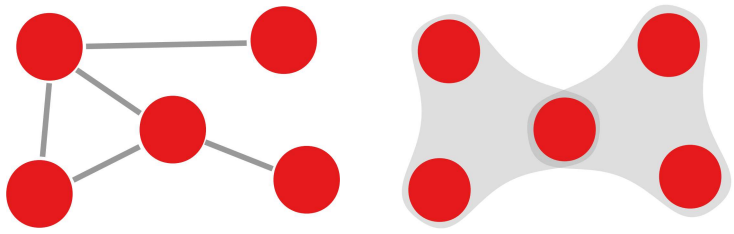
7.3 *Many-to-Many Relationships*

Hypergraphs

In the classical definition, an edge connects two nodes – the gray lines in Figure 7.7(a). Your friendship relation involves you and your friend. If you have a second friend, that is a different relationship. There are some cases in which connections bind together multiple people at the same time. For instance, consider team building: when you do your final project with some of your classmates, the same relationship connects you with all of them. When we allow the same edge to connect more than two nodes we call it a *hyperedge* – the gray area in Figure 7.7(b). A collection of hyperedges makes a *hypergraph*[25,26].

[25] Vitaly Ivanovich Voloshin. *Introduction to graph and hypergraph theory*. Nova Science Publishers Hauppauge, 2009

[26] Alain Bretto. Hypergraph theory: An introduction. *Mathematical Engineering. Cham: Springer*, 2013



(a) (b)

Figure 7.7: (a) Classical Graph. (b) Hypergraph.

To make them more manageable, we can put constraints to hyperedges. We could force them to always contain the same number of nodes. In a soccer tournament, the hyperedge representing a team can only have eleven members: not one more nor one less, be-

cause that's the number of players in the team. In this case, we call the resulting structure a "uniform hypergraph", and have all sorts of interesting properties[27]. In general, when simply talking about hypergraphs we have no such constraint.

It is difficult to work with hypergraphs[28]. Specialized algorithms to analyze them exist, but they become complicated very soon. In the vast majority of cases, we will transform hyperedges into simpler network forms and then apply the corresponding simpler algorithms.

There are two main strategies to simplify hypergraphs. The first is to transform the hyperedge into the simple edges it stands for. If the hyperedge connects three nodes, we can change it into a unipartite network in which all three nodes are connected to each other. In the project team example, the new edges simply represent the fact that the two people are part of the same team. The advantage is a gain in simplicity, the disadvantage is that we lose the ability to know the full team composition by looking at its corresponding hyperedge: we need to explore the newly created structures.

The second strategy is to turn the hypergraph into a bipartite network. Each hyperedge is converted into a node of type 1, and the hypergraph nodes are converted into nodes of type 2. If nodes are connected by the same hyperedge, they all connect to the corresponding node of type 1. In the project team example, the nodes of type 1 represent the teams, and the nodes of type 2 the students. This is an advantageous representation: it is simpler than the hypergraph, but it preserves some of its abilities, for instance being able to reconstruct teams by looking at the neighbors of the nodes of type 1. However, the disadvantage with respect to the previous strategy is that there are fewer algorithms working for bipartite networks than with unipartite networks.

Figure 7.8 provides a simple example on how to perform these two

[27] Shenglong Hu and Liqun Qi. Algebraic connectivity of an even uniform hypergraph. *Journal of Combinatorial Optimization*, 24(4):564–579, 2012

[28] Source: I tried once.

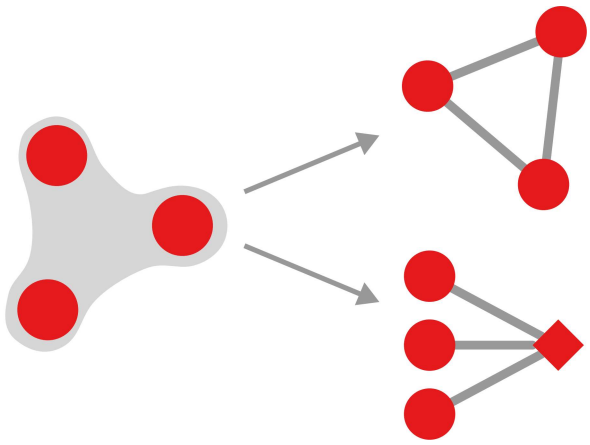


Figure 7.8: The two ways to convert a hyperedge into simpler forms. A hyperedge connecting three nodes can become a triangle (top right), or a bipartite network (bottom right).

conversion strategies on a simple hyperedge connecting three nodes.

When it comes to notation, the network is still represented by the classical node and edge sets: $G = (V, E)$. However, the E set now is special: its elements are not forced to be tuples any more. They can be triples, quartuplets, and so on. For instance, (u, v, z) is a legal element that can be in E, with $u, v, z \in V$.

Simplicial Complexes

Simplicial complexes[29,30,31] are related to hypergraphs. A simplicial complex is a graph containing simplices. Simplicial complexes are like hyperedges in that they connect multiple nodes, but they have a strong emphasis on geometry. Graphically, we normally represent simplicial complexes as fills in between the sides that compose the simplex – as I show in Figure 7.9.

[29] Vsevolod Salnikov, Daniele Cassese, and Renaud Lambiotte. Simplicial complexes and complex systems. *European Journal of Physics*, 40(1):014001, 2018

[30] Jakob Jonsson. *Simplicial complexes of graphs*, volume 3. Springer, 2008

[31] Ginestra Bianconi. *Higher-order networks*. Cambridge University Press, 2021

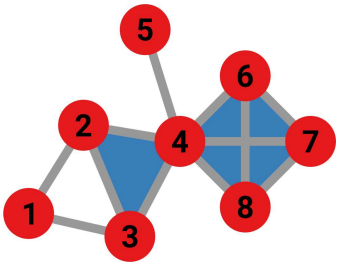


Figure 7.9: An example of simplicial complex. The blue shades represent the two simplices in the complex.

You can think of simplices as hyperedges on steroids. While a hyperedge including, say, four nodes is "just" a group of four nodes all interacting with each other, a simplex of four logically also contains all lower-level simplices, which are taken into account in the analysis. Notation-wise, a node is a 0-simplex, an edge is a 1-simplex, then a 2-simplex connects three nodes, and you can see where this is going. A simplex connecting $n + 1$ nodes is a n-simplex. Figure 7.10 shows you the first entries of the simplicial complex zoo.

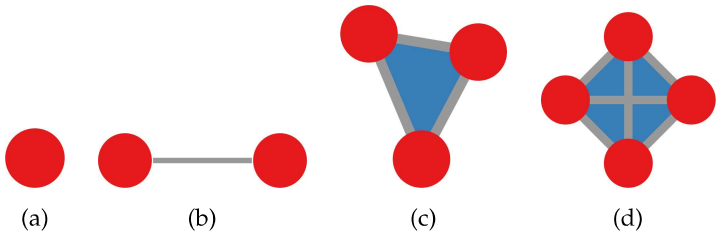


Figure 7.10: The smallest simplices a simplicial complex can have. (a) 0-simplex, (b) 1-simplex, (c) 2-simplex, (d) 3-simplex.

What does it mean for a simplicial to contain all its lower level versions? Consider the 3-simplex in Figure 7.10(d). That simplex contains four 0-simplices – the nodes –, six 1-simplices – the edges –, and four 2-simplices. These are called the *faces* of the simplex – think of them as the faces of a solid in geometry, because that's

what they are. On the other hand, a hyperedge with four nodes only contains those four nodes, it does not logically contain any three-node hyperedge – unless that specific three-node hyperedge is explicitly coded as part of the data, but it is a wholly separate entity. In the paper writing example, four people writing a paper make a four-node hyperedge, but only a *different* paper with three of those authors will generate a hyperedge contained in it – while a 3-simplex will naturally contain all 2-simplices with no extra paper.

A simplicial complex – a network with simplices – has a *dimension*: the largest dimension of the simplices it contains. The simplicial complex in Figure 7.9 has dimension 3. A *facet* of a simplicial complex is a simplex that is not a face of any other larger simplex. The facets in the simplicial complex of Figure 7.9 are: $[\{1,2\},\{1,3\},\{2,3,4\},\{4,5\},\{4,6,7,8\}]$. The sequence of facets of a simplicial complex fully distinguishes it from any other simplicial complex. Just like with uniform hypergraphs, we can also have pure simplicial complexes, which are complexes that contain only facets of the same dimension. Figure 7.9 is not pure because it has facets of dimension three, two and one. Figure 7.10(d) is a 3-pure simplicial complex, because it only contains a simplex of dimension three. If you were to ignore all simplices and analyze the network without them, you'd be working with the *skeleton* of the simplicial complex. In practice, any network is a skeleton of one or more simplicial complexes.

Simplicial complexes are one of the main ways to analyze high-order interactions in networks, and so we're going to look at them extensively in Chapter 34.

7.4 *Dynamic Graphs*

Most networks are not crystallized in time. Relationships evolve: they are created, destroyed, modified over time by all parties involved. Every time we use a network without temporal information on its edges, we are looking at a particular slice of it, that may or may not exist any longer.

For many tasks, this is ok. For others, the temporal information is a key element. Imagine that your network represents a road graph. Nodes are intersections, and edges are stretches of the street connecting them. Roadworks might cut off a segment for a few days. If your network model cannot take this into account, you would end up telling drivers to use a road that is blocked, creating traffic jams and a lot of discomfort. That is why you need dynamic – or temporal – networks[32,33,34,35,36].

Consider Figures 7.11(a) to (d) as an example. Here, we have a

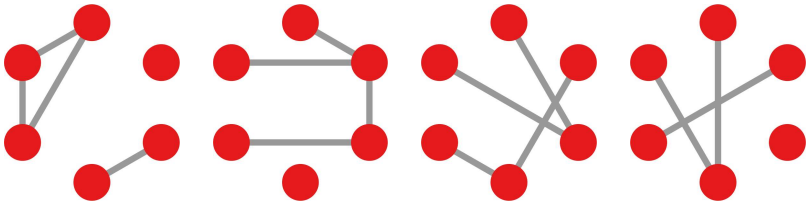
[32] Soon-Hyung Yook, Hawoong Jeong, A-L Barabási, and Yuhai Tu. Weighted evolving networks. *Physical review letters*, 86(25):5835, 2001

[33] Alain Barrat, Marc Barthélemy, and Alessandro Vespignani. Weighted evolving networks: coupling topology and weight dynamics. *Physical review letters*, 92(22):228701, 2004b

[34] Petter Holme and Jari Saramäki. Temporal networks. *Physics reports*, 519 (3):97–125, 2012

[35] Vincenzo Nicosia, John Tang, Cecilia Mascolo, Mirco Musolesi, Giovanni Russo, and Vito Latora. Graph metrics for temporal networks. In *Temporal networks*, pages 15–40. Springer, 2013

[36] Naoki Masuda and Renaud Lambiotte. *A Guidance to Temporal Networks*. World Scientific, 2016



(a) $t = 1$ (b) $t = 2$ (c) $t = 3$ (d) $t = 4$

Figure 7.11: An example of dynamic network. Each figure represents the same network, observed at different points in time.

social network. People are connected only when they are actually interacting with each other. We have four observations, taken at four different time intervals. Suppose that you want to infer if these people are part of the same social group – or community. Do they? Looking at each single observation would lead us to say *no*. In each time step there are individuals that have no relationships to the rest of the group. Adding the observations together, though, would create a structure in which all nodes are connected to each other. Taking into account the dynamic information allows us to make the correct inference. Yes, these nodes form a tightly connected group.

In practice, we can consider a dynamic network as a network with edge attributes. The attribute tells us when the edge is active – or inactive, if the connection is considered to be "on" by default, like the road graph. Figure 7.12 shows a basic example of this, with edges between three nodes.

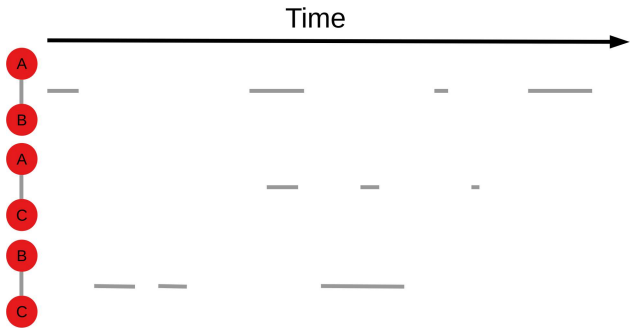
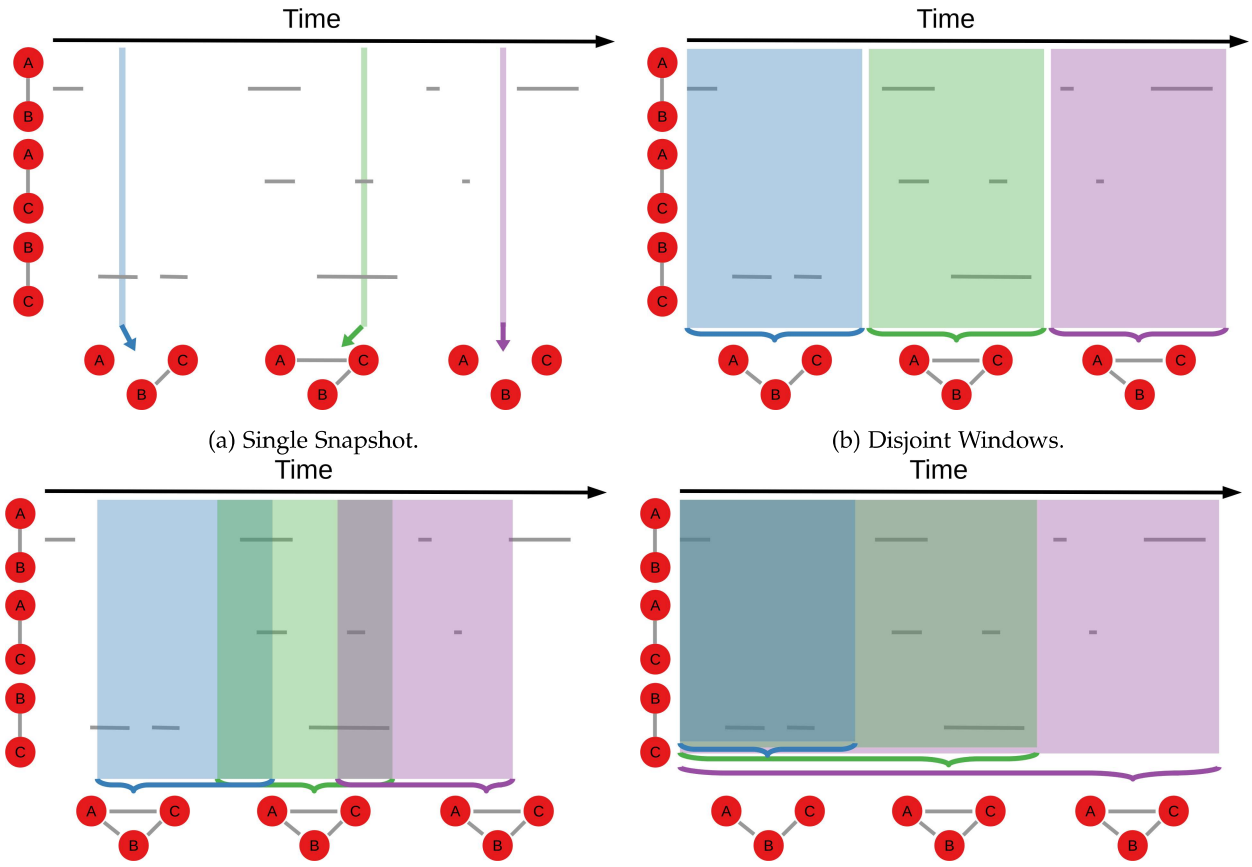


Figure 7.12: An example of dynamic edge information. Time flows from left to right. Each row represents a possible potential edge between nodes A, B, and C. The moments in time in which each edge is active are represented by gray bars.

More formally, our graph can be represented as $G = (G_1, G_2, ..., G_n)$, where each G_i is the i-th snapshot of the graph. In other words, $G_i = (V_i, E_i)$, with V_i and E_i being the set of nodes and edges active at time i.

How do we deal with this dynamic information when we want to create a static view of the network? There are a four standard techniques.

- *Single Snapshot* – Figure 7.13(a). This is the simplest technique. You choose a moment in time and your graph is simply the collection of nodes and edges active at that precise instant. This strategy works well when the edges in your network are "on" by default.



(a) Single Snapshot.

(b) Disjoint Windows.

(c) Sliding Windows.

(d) Cumulative Windows.

Figure 7.13: Different strategies for converting dynamic edges into a graph view.

It risks creating an empty network when edges are ephemeral and/or there are long lulls in the connection patterns, for instance in telecommunication networks at night.

- *Disjoint Windows* – Figure 7.13(b). Similar to single snapshot. Here we allow longer periods of time to accumulate information. Differently from the previous technique, no information is discarded: when a window ends, the next one begins immediately. Works well when it's not important to maintain continuity.

- *Sliding Windows* – Figure 7.13(c). Similar to disjoint windows, with the difference that we allow the observation periods to overlap. That is, the next window starts before the previous one ended. Works well when it is important to maintain continuity.

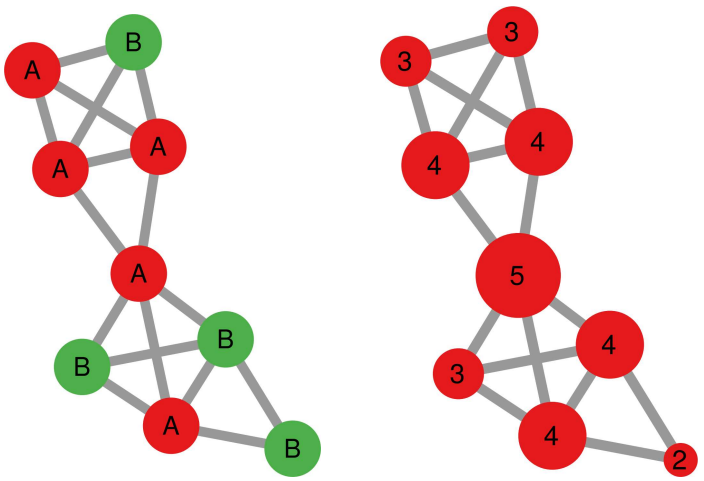
- *Cumulative Windows* – Figure 7.13(d). Similar to sliding windows, but here we fix the beginning of each window at the beginning of the observation period. Information can only accumulate: we never discard edge information, no matter how long ago it was firstly generated. Each window includes the information of all

previous windows. Works well when the effect of an edge never expires, even after the edge has not been active for a long time.

Note how these different techniques generate radically different "histories" for the network in Figure 7.13(a) to (d), even when the edge activation times are identical.

7.5 *Attributes on Nodes*

Earlier I defined what a bipartite network is: a network with two node types and edges connecting exclusively nodes of unlike type. You could consider the node type as a sort of binary attribute on the node. Once you make the step of adding some metadata to the nodes, why stopping at just two values? And why constraining how edges can connect nodes depending on their attributes? Welcome to the world of node attributes!



(a) (b)

Figure 7.14: (a) A network with qualitative node attributes, represented by node labels and colors. (b) A network with quantitative node attributes, represented by node labels and sizes.

Here we do not have the requirement of only establishing edges between nodes with unlike attribute values. Moreover the attributes don't have to be binary. They also don't have to be qualitative at all (as in Figure 7.14(a)): they can be quantitative, as in Figure 7.14(b). For instance, the number of times a user logged into their social media profile. Finally, nodes can have an arbitrary number of attributes attached to them, not just one.

Consider for instance a trade network. The nodes in this network are the various countries. They connect together if one country exports goods to another. We can have multiple quantitative attributes on each country. For instance, it can be its GDP per capita, its population, its total trade volume. On the other hand, we can also put countries in different categories: in which world region are they lo-

cated? Are they democracies or not? Of which trade agreement are they part of?

In this case, our graph changes form again: $G = (V, E, A)$. We can see each $v \in V$ not as a simple entity, but as a vector of attribute values: $v = (a_1, a_2, a_3, ...)$. In this representation, a_1 is the value for v of the first attribute in A. a_1 can be a real, integer, or a category.

Node attributes are important because nodes might have tendencies of connecting – or refusing to connect – to nodes with similar attribute values. We'll explore this topic in the forms of "homophily" in Chapter 30 for qualitative attributes, and "assortativity" in Chapter 31 for quantitative attributes. This is different from bipartite networks because in bipartite networks edges between nodes with the same attribute value are *forbidden*, while in these cases edges are simply *correlated* with attribute values. Moreover, bipartite networks are only defined for qualitative attributes, not quantitative.

To wrap up, no one forces you to use a single of these more complex graph models at a time. You can merge them together to fit your analytical needs. For instance, you can create this monster graph type: $G_n = (V_1, V_2, E, L, W, A)$: a bipartite graph with V_1 and V_2 nodes, each with attributes in A, which is weighted (W) multilayer with $|L|$ layers and – for good measure – is also a hypergraph, allowing edges in E with more than two nodes. And, of course, you can observe it at multiple time intervals (G_1, G_2, ...). Yikes.

7.6 Summary

1. Bipartite networks are networks with two node types. Edges can only connect two nodes of different types. You can generalize them to be n-partite, and have n node types.

2. In multigraphs we allow to have multiple (parallel) edges between nodes. We can have labeled multigraphs when we attach labels to nodes and edges. Labels on nodes can be qualitative or quantitative attributes.

3. If we only allow one edge with a given label between nodes we have a multiplex or multilayer network: the edge label informs us about the layer in which the edge appears.

4. Multilayer networks are networks in which different nodes can connect in different ways. To track which node is "the same" across layers we use inter-layer couplings. Couplings can connect a node in a layer to multiple nodes in another, making a many-to-many correspondence.

5. Signed networks are a special type of multilayer network with two layers: one positive (e.g. friendship) and one negative (e.g. enmity).

6. Hypergraphs are graphs whose (hyper)edges can connect more than two nodes at the same time. You can consider hyperedges as cliques or bipartite edges.

7. Simplicial complexes, like hypergraphs, allow nodes to connect in many-to-many relationships called simplices. Simplices are more powerful than hyperedges because a simplex of 4 nodes logically contain all of its smaller simplices – called "faces".

8. Dynamic graphs are graphs containing temporal information on nodes and edges. This information tells you when the node/edge was present in the network. There are many ways to aggregate this information to create snapshots of your evolving system.

7.7 *Exercises*

1. The network in `http://www.networkatlas.eu/exercises/7/1/data.txt` is bipartite. Identify the nodes in either type and find the nodes, in either type, with the most neighbors.

2. The network in `http://www.networkatlas.eu/exercises/7/2/data.txt` is multilayer. The data has three columns: source and target node, and edge type. The edge type is either the numerical id of the layer, or "C" for an inter-layer coupling. Given that this is a one-to-one multilayer network, determine whether this network has a star, clique or chain coupling.

3. The network in `http://www.networkatlas.eu/exercises/7/3/data.txt` is a hypergraph, with a hyperedge per line. Transform it in a unipartite network in which each hyperedge is split in edges connecting all nodes in the hyperedge. Then transform it into a bipartite network in which each hyperedge is a node of one type and its nodes connect to it.

4. The network in `http://www.networkatlas.eu/exercises/7/4/data.txt` is dynamic, the third and fourth columns of the edge list tell you the first and last snapshot in which the edge was continuously present. An edge can reappear if the edge was present in two discontinuous time periods. Aggregate it using a disjoint window of size 3.