

Chapter 10

Awful errors and how to amend them

Network data, like all data, are imperfect measures of objects of study. There may be missing information or false information. For networks, these measurement errors can lead to missing nodes or links (network elements that exist in reality but are absent from the network data),¹ or spurious nodes or links (nodes or links present in the data but absent in reality). More troubling is that these conditions exist in a continuum, and there is a spectrum of scenarios where nodes or links may exist but not be “meaningful” in some way.

In this chapter, we describe how such errors can appear and affect network data, and introduce some ways these errors can be handled in the data processing steps. (For more on the theory of errors and uncertainty in networks, see Ch. 24.) Such error fixes can lead to different networks, before and after processing, for example. Chapter 14 discusses techniques for comparing networks that may be useful to guide any processing you may wish to do.

i **Errors are not mistakes.** Here we use *error* in the sense of scientific uncertainty. Errors in the network come from either measurement uncertainty or processing uncertainty. We distinguish these issues from systematic mistakes or blunders made by the researcher, such as creating buggy computer code. Such blunders are truly awful. (Of course, buggy code used in the past may be *causing* errors in the data you are presently working with.)

10.1 Errors in data: omission and commission

Broadly speaking, and certainly not limited to networks, errors in data fall into three classes:

¹ We previously encountered missingness not in the network but in attributes associated with the network; Sec. 9.5.

Missingness where an observation should have been captured in the data but was not.

Spuriousness where an observation appears in the data but did not actually exist.

Uncertainty where an observation was made but its value was recorded incorrectly.

A scientist will always have to contend with the specter of such errors, thinking whether their data are affected and why.

For example, suppose you are gathering data by polling individuals about their political views. You may deal with missing values due to *survey nonresponse*, where some individuals preferentially avoid participating in your poll. Were this nonresponse to be uncorrelated with the data you are gathering, you could deal with it by considering your data as randomly sampled from a larger population. Unfortunately, this is seldom the case. Instead, individuals who avoid your poll are much more likely to hold unusual or non-mainstream views, or to distrust scientists, or the like. Your missingness is now non-random, and the conclusions you will draw about the distribution of political views is likely to be skewed.

Spuriousness and uncertainty can also appear within these polling data. False observations can be recorded due to mistakes or (unfortunately) fraud. Individuals who wish to skew political views may attempt to participate in the poll multiple times or workers canvassing for your poll may accidentally survey the same individuals multiple times. And mixups can occur when the surveyors take notes, leading to uncertainty due to incorrectly recorded survey responses.

This polling example highlights some errors that occur based on the data generating mechanism, how the data are observed and recorded. In this case, people are involved directly, creating the data (by participating in the poll) and recording the data (by conducting the poll). But such problems can still occur, and may even be worse, when people are not in the equation, as automation—automated data gathering and automated processing—can lead to such errors as well.

Take care when addressing errors

Errors in data should be addressed to ensure your conclusions from those data are reliable and well supported. This can lead data scientists toward methods to fix errors. In principle, fixes are great, but often in practice fixes are infeasible and it is best to only identify errors and reason about their sources. In fact, poor fixes can make things worse. Let's discuss some consequences that occur when trying to deal with missing values.

Have you ever started analyzing a data table and discovered there are a bunch of “N/A” or “NaN”² values in your table? These symbols are being used to represent missing values and you can't compute even basic summary statistics with them in place.³ Although it is all too common to immediately drop rows with missing values⁴

² “NaN” or “Not a Number” represents an ill-defined number in standard floating point numeric convention.

³ Averaging, for example, a collection of numbers containing a single “NaN” will make the average itself be “NaN.”

⁴ Dropping any rows in a table of data that contain at least one missing value is sometimes known as *listwise deletion*.

so you can proceed with calculations, in practice, this can be dangerous. The reason is that the mechanism for missingness can severely bias what we can see in the data.

For example, say we are interested in the relationship between income and education by using a survey dataset that contains income and education information for a sample of people. We may be tempted to drop all rows with missing income or education values. What happens then? Let's think about why someone might have not reported their income. It could be that they are embarrassed about their income, or do not have a consistent source of income. If this is the case, then we would expect that people who do not report their income are more likely to have lower income than people who do report their income. In other words, by dropping the datapoints without income values, we are introducing a strong bias in our analysis.

In any serious data science practice, we should never simply drop rows with missing values without investigating it. Instead, we should try to understand the mechanism of missingness and try to address it. The first criterion to think about is whether the missingness is *missing completely at random (MCAR)* or not. If the missingness is MCAR, that means that the probability of missing data is same for all datapoints and therefore it is safe to drop the rows with missing values. Of course, it is very rare that the missingness is MCAR and that we can confirm that the missingness is MCAR.

If the missingness is not MCAR, then there are several possibilities. The first case is *missingness at random (MAR)* (without "completely"). This means that the probability of missingness is not the same for all datapoints, but it only depends on the observed data. If this is the case, we can drop the rows with missing values as long as we can control for the observed characteristics that affect the missingness. Then there can be the case of missingness that depends on unobserved data (or even the missing value itself), or *missing not at random (MNAR)*. One common example is the case of clinical trials, where a patient becomes more likely to drop from the trial if they feel discomfort from the treatment. If the level of discomfort is not measured, then there is no observed variable about the patient that affected the missingness, although the missingness is definitely affected by a specific variable and not random. In this case, the missingness should be explicitly modeled.

While our discussion has centered on missing values, spurious and uncertain values need to be addressed too. In many ways, these are even more challenging, as a spurious value is less obviously spurious than a missing value is missing. Further, attempts to fix these problems can increase uncertainty (a value meant to correct for a noisy observation may be even farther from the truth) or missingness (say, by incorrectly removing a supposedly spurious observation). Once again, the best bet is often to address errors only by identifying them.

10.2 Errors in networks

Our problems compound when confronted with a network structure. Here errors such as missingness or spuriousness can interact through the network structure. Broadly speaking, we can have missing nodes (a node in the network fails to appear in our data), missing links (a link between observed nodes is missing), spurious nodes (a node appears in our data which is not present in reality), and spurious links (a link appears

in the data between observed nodes that is not present in reality).

But these categories intersect. A link can be missing because both nodes it connects are missing. A spurious node can be introduced by a duplication error where a node is incorrectly included twice in the network. And if this occurs, what about the links connecting those nodes: are they divvied up somehow between the real and spurious node? Or, are they each duplicated so both nodes have every link?

Missingness in network data

In the context of network data, missingness can arise in multiple contexts. First of all, the missingness can be about the nodes. When we examine a social network obtained from a social media platform, we will miss many of those who do not use the platform. And then we may not see those who have not interacted with anyone in the network or simply were inactive during our observation period. If we examine a protein–protein interaction network, we may miss the proteins that have not been catalogued yet or those that are simply too difficult to test. Missing nodes, especially if they are very central nodes or hubs of high degree, can drastically alter our view of the network's structure.

Missingness can also be about the links. Even if neither node is missing, we may be unaware of the link that connects them. This may come from limitations of the experimental techniques (e.g., a protein–protein interaction network), or limitations of the data collection procedure (e.g., the subjects in a social network survey cannot recall all of their friends or acquaintances). And missing links can also have a profound effect on our view of the network.

Note that there is a difference between missing data (usually links) and measured absence. If the existence of a link was tested or measured, then the lack of the link is not missing data; the absence was measured.

In addition to missing the nodes or links themselves, we can also be missing information about non-missing nodes or links: the missingness can be about the attributes of the nodes or links that we can use for statistical analyses. For instance, missing income can bias our analyses of the relationship between income and social network structure, as described above.

Spuriousness in networks

On the other hand, we may have spurious data in the network. As no experiment or data collection is perfect, spurious nodes or links can find their way into the data. A survey respondent may confuse someone with another; an experiment may yield a false positive; someone on social media may have mistyped their friend's username. Even when there are no such errors in the basic measurements, there still can be spurious links.

One prominent example is *indirect correlation*. When it is difficult to observe interactions or links directly, we often rely on correlational measurements. For instance, if we measure how two genes respond to various perturbations (experimental conditions), we can calculate a correlation matrix between genes. This correlation matrix can then be interpreted as a network. However, this poses a challenge. Not only are we in the situation of multiple hypothesis testing (and therefore some of the *significant* correlation

values would be spurious), but also we will have indirect correlation. Let's say gene *A* strongly affect gene *B*—*B*'s expression is strongly correlated with *A*'s. At the same time, *A* also affects another gene *C*—*C*'s expression is also strongly correlated with *A*'s. In this situation, what do we observe between *B* and *C*? Most likely—although not guaranteed—*B* and *C* will show significant correlation with each other, even if they do not directly interact. This positive, indirect correlation may be strong enough to even wash out weak, *negative* direct interaction between *B* and *C*! This problem exists even if the data collection or experiments are perfect.

What effects can spuriousness have on the network structure? Of course, the predominant issue is links appearing in the network that are either absent in reality or are otherwise meaningless. But compound effects can occur. We may encounter incorrect node *merging*—one node in the data is more than one node in reality. Or we may have the inverse, incorrect node *splitting*—multiple nodes in the data are one node in reality. Such errors arise when nodes are not clearly identified, for example, a social network where users are pseudonymous or only partial identifying information is available can quickly lead to one person appearing multiple times in the network and multiple people appearing as a single node in the network. Node splitting can also lead to link splitting: if a link exists between *i* and *j* and *j* is split into *j*₁ and *j*₂, then we may expect link *i, j* to become two links, *i, j*₁ and *i, j*₂. And likewise, node merging can lead to link merging. These node errors can be especially insidious compared to simpler missing or spurious link errors.

10.3 Sources of network errors


Whenever network data are gathered, one must consider whether the gathering mechanism has introduced errors. Often this requires having more than a passing familiarity with the mechanism itself, how it works and how it might fail. Consider link errors. Will false positive links occur, links that appear in your dataset but are not truly present? What about false negatives, where nodes appear disconnected in your data but are linked in reality? Often a given data generating process is more likely to create one type of error than another.

Experimental and observational data will come with errors. Experimental protocols, for instance assays that test protein interactions, will never be perfect, although good methods and good researchers will always put in the effort to understand and if possible mitigate such issues. Observational data, likewise, may fail to capture all facets underlying the network structure and if a computational or statistical method is used to infer links from those data, that method may itself be the source of errors. Many datasets are also generated manually, through surveying or laborious human coding, and such manual efforts are prone to human error.

Based on your understanding of the data you are working with, can you tell what are the main sources of errors? What are the net results of these errors, many false or spurious nodes or links? Or perhaps too few links are present, an error of omission? When many links are included that should not be, you may wish to thin your data. Otherwise, if you suspect much of your network is missing, consider trying to thicken your network. We turn our attention to such fixes now.

10.4 Fixes

Given errors are possible, even likely, what can we do? Is it possible to identify errors in the network structure and correct them? What methods might we use for such *fixes*?

 Fixing errors is non-trivial, sometimes impossible. Often the best we can do or should do is *identify* and *understand* the errors that are likely present. Doing so gives us more confidence in the data because we can anticipate failure modes for problems relying on those data. Unreliable fixes, in comparison, may pollute the data, leaving us worse off than if we had instead done nothing at all.

Having discussed a bit about errors in network data, both the kinds of errors and their possible causes, we now turn our attention to what to do about them.

10.5 Thinning spurious data

At times, the network you extract from the underlying data will be overloaded with too many links, nodes, or both. These elements may be entirely spurious, or they may be deemed extraneous junk to be eliminated.⁵ Sometimes, the data are too big to even fit into memory, which makes it tedious to analyze the network. Highly dense networks often appear when considering weighted networks. For instance, networks constructed from co-occurrences or correlations can be extremely dense, even fully connected. When the network is extremely dense, network visualization does not work well and even simple measures like degree can become meaningless.

In these cases, it is worth removing the least important elements of the network in order to facilitate visualization, computation, and analysis. Here we discuss ways to thin out these elements, forming a reduced or “thinned” network.

10.5.1 Thresholding by degree

Probably the simplest way to reduce the size of a network is to remove nodes with degree less than a certain threshold. The implicit assumption here is that the degree of a node reflects its importance and thus removing the least connected nodes is a good way to reduce the network while losing the least amount of information. The simplest case is removing nodes with degree one (also known as “dangling nodes”). Nodes with degree one are nodes that are connected to only one other node and thus it is often reasonable to consider them to belong to the node that they are connected to. For instance, if we imagine community structure in the network, it is natural to assume that all dangling nodes belong to the same community of the node that they are dangling from. Many real-world networks, because they tend to be a sample of a larger population, tend to have numerous dangling nodes that do not contribute a lot of information to the network, and removing them often leads to much smaller networks that are amenable to visualization and computational analyses. And of course, we can and should also consider thresholds other than 1, the challenge being choosing the threshold appropriately.

⁵ Keep in mind that care should be taken declaring anything “junk” (cf. junk DNA [287, 134]).

Degree thresholding is even more useful in *directed networks*. In directed networks, either in-degree or out-degree reflects the importance of the node in the network better than the total degree and therefore these filtering methods can be used more effectively. For instance, let's say we have obtained a retweet network about a certain topic from Twitter. In this network, where the nodes represent users and the directed edges represent retweets, the in-degree captures how many times a user's tweets have been retweeted by others, which can be one of the simplest proxy of "influence" on Twitter. Therefore, filtering nodes based on their in-degrees is a good way to remove nodes that are less important to the conversation.

10.5.2 Thresholding by cores

A slightly more sophisticated variation of degree-based thresholding is to use the concept of *k-cores*. A *k-core* is a subnetwork of a network containing every node that is connected to at least *k* other nodes in the subnetwork. For instance, a 2-core subnetwork of a given network is a subnetwork where every node has at least two neighbors. This 2-core subnetwork can be obtained by iteratively removing all nodes with degree less than two. Any *k-core* subnetwork can be obtained similarly. Although not guaranteed, often the cores of the network contain the most important structure to analyze.

10.5.3 Weight thresholding

When the edges are weighted, using the weight of the edge usually provides a higher-resolution proxy of importance. We can not only filter out nodes based on their importance, but also filter out edges based on their weight or other derived measures of importance.

The simplest approach is removing edges based on a weight threshold. We simply set a minimum weight threshold—call it w_{thr} —and ignore all edges that have smaller weight than the weight threshold. The idea behind this method is that more important edges are heavier weight and choosing the right threshold will let us retain those important edges.

Figure 10.1 shows an example of some weight thresholds in practice. Here we have taken the Malawi Sociometer Network,⁶ removed nodes outside the largest connected component, and then retained only those edges (i, j) with weight $w_{ij} > w_{thr}$. (After removing edges, any nodes of degree zero were also removed.)

Although this is probably the most common approach, it often fails to reveal meaningful structure and is generally a poor method to reduce the complexity of the network. The reasons that a simple threshold does not work well are (1) real networks often exhibit core-periphery structure or so-called "rich-club" structure (see Ch. 12), and (2) there is seldom a single, universal threshold that, across the entire network, can meaningfully separate edges worth keeping from edges best discarded.

Rich-club structure is characterized by stronger and denser connections among a "rich club" of nodes while the density decreases as we move towards the periphery. If

⁶ Here we convert the temporal network to a weighted network by summing the number of contacts between participants over time.



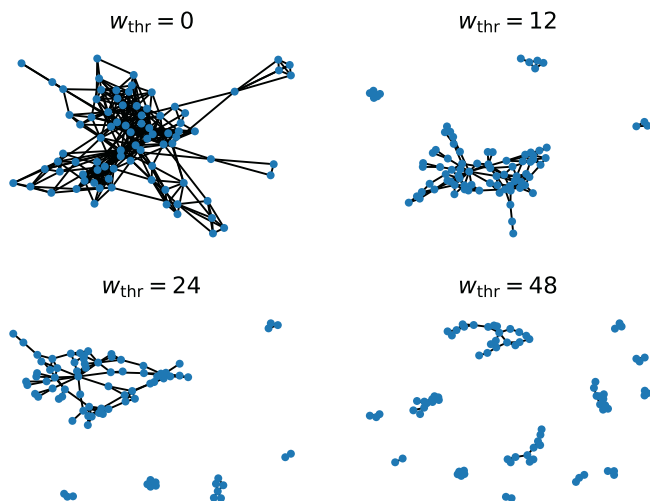


Figure 10.1 Visualizing the Malawi Sociometer Network under different weight thresholds. See Ch. 13 for more on such *network visualizations*.

the network possesses this structure, a simple thresholding will result in the removal of dangling periphery nodes, which does not meaningfully reduce the density at the core. Increasing the threshold value may not help, because it will gradually remove only the most peripheral nodes that are remaining in the network, leaving dense cores unscathed. We see evidence of this happening in Fig. 10.1.

Nevertheless, a single threshold is the simplest way to make use of edge weights and can be useful when a concrete, well-reasoned threshold value can be picked and when the network does not exhibit strong rich-club structure. In biological applications, often the choice of threshold is done by considering the false discovery rate⁷ or, for co-occurrence/correlation networks, setting a p-value threshold of the correlation.

10.5.4 Windowing

Windowing is a particular kind of thresholding most useful for dynamic networks (Ch. 15).

Suppose each node or edge of the network has time periods associated with it. Windowing means to retain the subnetwork associated with a particular time period, the data “window,” by keeping only those nodes and edges that appear in the network during that window. Windowing the network can allow researchers to focus on the

⁷ The False Discovery Rate (FDR) is defined based on how often you reject the null hypothesis: $FDR = FP / (FP + TP)$ where FP is the number of false positives (rejecting a null hypothesis which is actually true) and likewise TP is the number of true positives. In the area of hypothesis testing, when repeatedly testing the same data (multiple comparisons) the rate of errors is higher than expected. The FDR was introduced as a better measure to quantify error rates than the significance level α typically used to quantify single hypothesis tests.

most salient time period they wish to study. For example, a social network extracted from online social activity may be most relevant to political scientists during the period leading up to a major national election. Those scientists may want to retain nodes (users) and links (say, retweets) in the period preceding and immediately following the election.

But how to define the start and end of this window of interest? Typically, it will depend on the problem being studied. The political scientists may know that most political activity occurs during the week of an election, or perhaps the preceding month. Likewise, the election under study may have resolved itself quickly and most interest is lost within a few days from the election's conclusion. Conversely, the election may have been highly contested, and interest continued for many weeks, so the scientists should use that information when defining their window's end period.

Window inference. What should be done when a researcher does not have prior information on the time period defining their window of interest? While potentially costly, the best bet is to devise a sweep over time periods. Define a set of window sizes (time durations), slide them over the time span of the data, and examine the network within each period. In other words, suppose your window size is ΔT . Examine the networks extracted from windows $[t_1, t_1 + \Delta T]$, $[t_2, t_2 + \Delta T]$, \dots , $[t_n, t_n + \Delta T]$, where t_1, \dots, t_n are a set of time window starting points you have chosen to iterate over.⁸ As an example, you may wish to look at a window of size ΔT that you slide over one month of data one day at a time, making $t_1 = \text{day 1}$, $t_2 = \text{day 2}$, and so forth. Then, after iterating over many time periods for a ΔT , choose another value of ΔT and repeat the process. This double-loop inference is expensive, and leads to many networks to compare (Ch. 14) but, in essence, this is a burden you should in general expect to bear when you are operating with data but without prior information.

Equipped with a collection of networks of different windows, which window is most appropriate? Again, you ideally turn to prior domain knowledge of those data. Lacking this knowledge, you may be able to specify a reasonable measure based on the network topology. For example, which window has the densest network or which window has the sparsest network? You may need to control these measures, however. For example, a longer window will probably lead to a denser network regardless of other factors, so you may instead want to look at a normalized density such as number of edges per unit time. Likewise, maximizing sparsity may lead to a period with a (nearly) empty network; instead, consider finding the sparsest network that is still connected. Lastly, if you are equipped with a statistical model for the data (e.g., Ch. 23), you may be able to look at the network with the largest likelihood under that model (again, perhaps controlling for model complexity and window size).

Window robustness. Suppose you have decided on a time period $[t_1, t_2]$ to study. You've extracted the network of nodes and edges active or otherwise associated with that window and you are ready to analyze the network. But doubt creeps in: Do I have the right window of interest or should I consider a different window $[t'_1, t'_2]$? Should I look at a wider window ($t'_1 < t_1, t'_2 > t_2$) or a narrower one ($t'_1 > t_1, t'_2 < t_2$)? Perhaps the start of my window is correct but the end is not, or vice versa?

These questions need to be addressed with a *robustness analysis*. As you vary the window of interest, does the network change? If change occurs, when and by how

⁸ Can you determine meaningful starting points for your data?

much? Do your conclusions *about* the network change? We strongly encourage such robustness checks: confirm that small or medium changes to either t_1 , t_2 , or both, do not fundamentally change your results. You can check these changes by comparing (using a suitable measure; Ch. 14) the network taken from window $[t_1, t_2]$ to one taken from a perturbed period $[t_1, t_2 + \delta]$,⁹ where δ is a small perturbation, either additive or subtractive, to the end of the time period.

10.5.5 Backbone extraction

Backbone extraction refers to a set of methods that filter edges by computing their “importance”—usually based on their weights and surrounding network structure. Edges deemed less important by some criteria are removed, reducing the density of the network. The remaining edges become the *backbone* of the network that plays the most critical structural role. Numerous methods exist with different operationalizations of the “backbone.”

We’ve already discussed weight thresholding, a basic thinning step for weighted networks where we retain edges i, j whose weights w_{ij} meet or exceed some given threshold: $w_{ij} \geq w_{\text{thr}}$. This method uses a global threshold w_{thr} . Introducing a global threshold may sometimes be sufficient (try it out and see; Ch. 11), but other times the final extracted network may be inaccurate or otherwise problematic. In particular, a single, global threshold fails to use structural information about the relationships between different edges at different locations in the network. Some nodes may be surrounded primarily by weak (low weight) edges while other nodes sit among many strong edges; a global threshold will not account for such variations across different portions of the network. Thus, it may be better to define an adaptive threshold that accounts for relative edge strengths across different regions of the network.

Let us discuss one method for backbone extraction called the *disparity filter* [424], which uses a simple and often effective way to define a relative threshold. It assumes that the importance of an edge i, j can be estimated by examining the nodes that are connected to it and their immediate neighbors. Let’s imagine a node i with $k_i = 4$ neighbors, each of which is connected to i with a different weight: $\{1, 10, 3, 2\}$. Obviously, the edge with weight of 10 (let’s say (i, j)) would be the most important edge to i , but how important? The disparity filter argues that we can think of a simple null model, where we randomly divide the total strength W_i ($1 + 10 + 3 + 2 = 16$) into four parts, representing what the weights would look like if their total was the same and they were distributed over the same number of edges, but no edge was preferred over another. We can visualize this null model as dropping $k_i - 1$ points on the unit (0,1) interval uniformly at random to create k_i random intervals that always sum to 1. Then, what is the probability, call it α_{ij} , for one of the intervals created at random to be at least w_{ij}/W_i long? For an interval of size $x = w_{ij}/W_i$ to occur at the leftmost interval (beginning at 0) we have to make sure a point is dropped at x , which occurs with infinitesimal probability dx , and none of the remaining $k_i - 2$ points land at positions less than x , which occurs with probability $(1 - x)^{k_i - 2}$. (Because points are equally likely anywhere, and the entire interval is of length 1, the probability for a point to fall inside a given interval is the length of that

⁹ Or periods $[t_1 + \delta, t_2]$ or $[t_1 + \delta_1, t_2 + \delta_2]$.

interval.) So the probability for the first interval to be at least length w_{ij}/W_i is given by $\int_{w_{ij}/W_i}^1 (1-x)^{k_i-2} dx$. But there is nothing special about the first interval, created by the leftmost of the $k_i - 1$ points, so the same quantity will hold if we consider the interval created between the leftmost and the second leftmost points, between the second leftmost and third leftmost points, and so on. Thus, the probability we see at least one interval as long or longer than w_{ij}/W_i is one minus the probability we see no intervals longer than w_{ij}/W_i or:

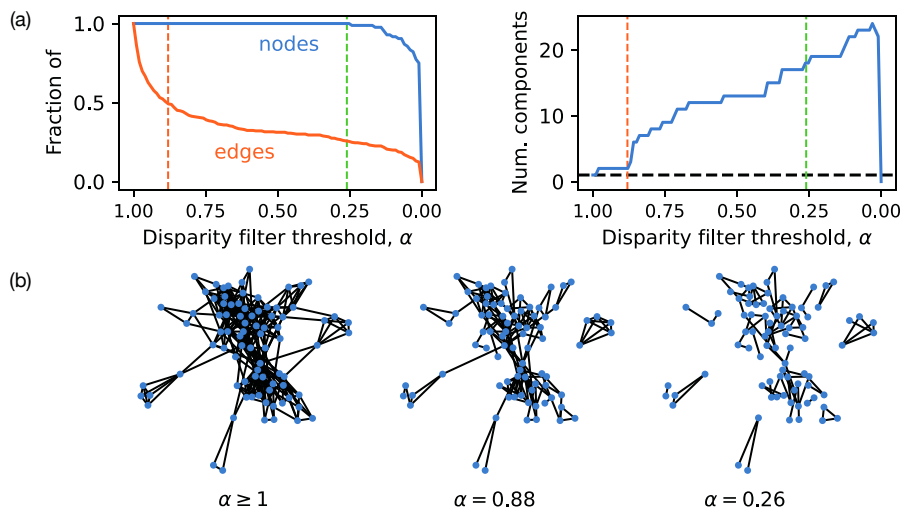
$$\alpha_{ij} = 1 - (k_i - 1) \int_0^{w_{ij}/W_i} (1-x)^{k_i-2} dx = \left(1 - \frac{w_{ij}}{W_i}\right)^{k_i-1}. \quad (10.1)$$

The smaller α , the more unlikely an edge's weight is at random, and therefore the more important that edge is likely to be. The disparity filter works by defining a threshold on the α_{ij} , meaning you keep all edges (i, j) with $\alpha_{ij} < \alpha_{BB}$ for some pre-chosen α_{BB} . Note that the importance of an edge (i, j) can be very different for i and j , meaning, from i 's perspective, the link to j may be very important, whereas from j 's perspective, the link to i may be unimportant compared to other links from j . The edge's weight may be the largest among i 's edges while being the smallest among j 's edges. Thus, we expect $\alpha_{ij} \neq \alpha_{ji}$ and in general it is best to keep an edge if it meets the threshold in either direction, meaning either the link to j is important from node i 's perspective or the link to i is important from node j 's perspective. The power of the disparity filter is that the null model defines a *local* threshold: each node's immediate neighbors are used to determine whether a link is important, unlike a less sophisticated global weight threshold.

A remark on α in the disparity filter method. When proposing the disparity filter, the authors motivated α as a significance threshold (p-value), even using the common choice of $\alpha = 0.05$. Indeed, the derivation of α_{ij} is akin to what one does with hypothesis testing where one defines a test statistic under a null model. However, such an interpretation brings with it a lot of baggage related to significance testing, including multiple comparisons and p-hacking. We have found it instead more fruitful to simply think of α as merely a threshold and not to try to interpret edges as "significant" or not. One effect of this is that a meaningful (i.e., useful) backbone can be had with values of α such as $\alpha = 0.2$ or $\alpha = 0.5$. Such values would be considered too large for hypothesis testing, but here the null model for how link weights are distributed is unlikely to be a plausible generating mechanism for the actual network (and you will surely have more domain knowledge for your network as to why). Were one to desire a proper null hypothesis test, one would need a null model for the network topology and the link weight distribution. The disparity filter works well by sidestepping these interpretive issues.

Example: using the disparity filter Let's apply the disparity filter to one of our focal networks, the Malawi Sociometer Network.¹⁰ For the weighted version of this network, edge weights w_{ij} represent the number of contacts observed between i 's sociometric badge and j 's badge over the course of the study. And, while the authors took care to

¹⁰ Here again we use the weighted version of the network.



⚡ Figure 10.2 Applying the disparity filter to the Malawi Sociometer Network. **(a)** Without guidance on choosing $\alpha_{BB} = \alpha$ for Eq. (10.1), we perform sweeps over a range of values to see where the network structure changes. We’ve flipped the direction of the x axes to mimic the effect of tuning α downward from the original network (at $\alpha = 1$) all the way to removing all edges (at $\alpha = 0$). **(b)** From the α sweeps, we identify a few points of interest and draw the network at those values compared to the original network at $\alpha = 1$.

define time contacts such that brief, likely noisy signals from the badges were excluded, it is still plausible that unimportant edges are captured in the data alongside important edges. So this is a good opportunity to apply the disparity filter using edge weights as inputs to Eq. (10.1).¹¹ However, doing so requires choosing α , the disparity filter’s threshold. What value should it be?

While the authors of the disparity filter advocate for values $\alpha \approx 0.05$, in our experience these are often too small. Indeed, applying such a value to the Malawi Sociometer Network would lead to few retained edges. Instead, let’s take a more computationally exhaustive approach by sweeping over a range of α and determining what value (or values) may be appropriate. To guide our search, for each value of α we compute the fractions of nodes and edges that remain in the network (nodes with degree zero after removing edges are removed) as well as the number of connected components. Looking at these quantities as functions of α can guide us towards particular points such as when nodes begin to vanish from the network or the number of components start to rise. We present these α sweeps in Fig. 10.2a.

From the curves in Fig. 10.2a we observe a few possibilities. Edges disappear quickly from the network as we increase the filter (i.e., as we lower α downward from $\alpha = 1$) before their removal rate starts to slow around $\alpha \approx 0.9$. Nodes begin to disappear at $\alpha = 0.26$ and that is a good candidate to examine further. Meanwhile, the number of

¹¹ Of course, being a temporal network, we may wish or need to use methods specific to temporal networks, such as windowing (Sec. 10.5.4), discussed in this chapter. See also Ch. 15.

components makes a sudden jump at $\alpha = 0.88$, meaning that parts of the network begin to break apart and this lines up well with the slowdown in removed edges. Taken together, these statistics point to two possible values of α to examine and we draw in Fig. 10.2b the network at both filtering points alongside the unfiltered network. Decreasing α even further may be useful as well, as many small, disconnected groups begin to emerge, pointing to community groups within the social network. And now, having thinned the network (a little or a lot, depending on our needs), we can continue to investigate the social network of the study participants. Finally, remember that, although backbone extraction allows us to simplify and reveal important structure of the network, many network analyses can be—and often should be—done on the original network rather than in the backbone.

While other backbone methods exist (see remarks below), we find the simple disparity filter to be an excellent starting point in practice, and it is often more than sufficient for our needs.

10.5.6 Using reciprocity

For directed networks, another thinning approach is to look for reciprocated ties, pairs of nodes where both (i, j) and (j, i) edges are present, and retain only those links. Reciprocity is a well-studied feature of social networks in particular, and it can be a good signal of the stronger, more important ties. Non-reciprocated ties, depending on the data, may be less meaningful in comparison. Of course, this depends as always on what network question is being addressed; it may be critically important to retain both reciprocated and non-reciprocated ties.

10.5.7 Using side information

The previous methods have all considered thinning the network using information from the network structure. For networks with attributes (Ch. 9), one can also use those attribute values to decide on a thinning criterion. For instance, suppose each node can be placed into one of three categories. We may have additional information that sampling is much better for one or more of the categories than the others. We can then thin the network simply by removing nodes outside of the better-sampled categories.¹²

Depending on the problem under study and the properties of the node or link attributes, we can devise any manner of other thinning actions. A numerical attribute can be thinned using a global threshold (retain all nodes where x exceeds a threshold we introduce). (If we consider edge weights to be an attribute, then we have actually discussed one example of using side information to thin a network.)

Consider the possibility of combining attribute-based thinning with structural thinning. Building off the example of thinning the network by retaining only nodes from well-sampled categories, imagine combining that thinning with a degree-based thinning. Now nodes are retained if they are high-degree (based on a threshold we define) and they fall in the categories we wish to retain. Combining these filtering conditions

¹² Whether this is appropriate of course, is another question. The bias in the sampling process still carries through to the final network.

introduces an order-of-operations complication: if we first filter by node category, the degrees of some nodes will be reduced, changing the results when filtering by degree. (Notice that filtering by attribute is not affected whether we filter first or second by degree.) So is it better to filter first by degree then by attribute? In the end, it will depend on the data at hand; if possible, we encourage you to explore both orders and see what difference—if any!—it makes.

10.6 Thickening missing data

A problem more challenging in many ways than subsetting the non-spurious network nodes and edges, is thickening the data. Here we are trying to grow our network, attempting to infer latent network elements, nodes or links, that were not captured in the data but likely do exist. Such a task is generally far more challenging than taking a known element and deciding whether or not it should be discarded. How can one make such inferences?

10.6.1 Graph union



Suppose you have conducted an experiment to measure protein–protein interactions such as those in HuRI. You have quite a few usable links in your network, but it is still very sparse and you are likely missing many interactions. The experimental procedure you used to measure those links is not the only option, however, and other teams have published networks using other experiments. You decide to thicken your network by *merging it* with other, published networks.¹³

The algorithm underlying this merger is called the graph union. The union of two graphs is defined by taking the unions of their nodes and their edges. Essentially, it defines a new network that contains a node if it exists in either (or both) of the original networks and it defines an edge if that edge exists in either (or both) of the original networks. You can see how this will thicken the network by adding edges where the nodes overlap and by potentially growing the network by adding more nodes.¹⁴ A graph union can represent collating additional data sources into your network, or performing additional experiments.

Computing a graph union is straightforward with one notable but often underappreciated challenge: the nodes need to be *clearly identified*. In other words, given a node i in one graph and a node j in the other graph, you need to be able to tell if they are the same node or not. This is straightforward when nodes have unique, trustworthy identifiers.¹⁵

¹³ Indeed, one of the networks studied as part of HuRI is dubbed “HI-union,” which is the merger of HuRI with all the research group’s previously published PPI screening experiments [283].

¹⁴ Interestingly, the graph *intersection* can be used likewise for thinning spurious data: retain only those edges discovered in multiple networks.

¹⁵ Identifying nodes without ambiguity is often assumed and taken for granted yet it is not nearly so common in many situations. Consider reconstructing a historical social network from contemporaneous records such as written documents and recorded eyewitness accounts. People may refer to one another in different ways, using nicknames; people may even be unsure what parties were involved in particular actions. Cultural effects can make this worse. Consider the difficulties of distinguishing individuals in the data when studying a society where people do not have surnames.

But if not, if you cannot reliably distinguish the nodes in different networks, trouble can ensue.

While computing the union is straightforward, more difficult is to decide whether it is appropriate. Essentially, by merging two networks, you are deciding that the definitions of links¹⁶ need to be extended. Now you are saying a PPI edge, for instance, exists if it was found by one experimental protocol or another. Perhaps this redefinition is fruitful but what if it is not? The sources of the two networks may be fundamentally incompatible and combining their edge sets may simply not make sense.

10.6.2 Link prediction

While the graph union represents the updating of a network with more data, other approaches are more speculative. Link prediction is the problem of inferring links that probably (hopefully!) exist in a network but are not (yet) seen in the data you have—can you predict these absent links with enough accuracy that you can incorporate them into your network data?

Simply put, we can think of a link prediction method as a function that takes a network (optionally including any associated attributes; Ch. 9) and two disconnected nodes i and j and returns either a binary indicator “yes, link i, j exists” or “no, link i, j does not exist” or a weight such as a probability for the link i, j to exist. This function can be simple or quite complicated. The classic Adamic–Adar function [3] is simple but often reasonably effective; we recommend it as a baseline or starting point if you wish to begin applying link prediction to your data. It uses no metadata, simply examines the observed network structure and computes a measure of similarity between i and j based on how many neighbors they have in common. Let N_i be the set of neighbors of node i , such that $|N_i| = k_i$ is the degree of i and $N_i \cap N_j$ is the set of nodes that are neighbors to both i and j . The Adamic–Adar index is then defined as a sum that runs over these common neighbors:

$$AA(i, j) = \sum_{u \in N_i \cap N_j} \frac{1}{\log k_u}. \quad (10.2)$$

This measures how many common neighbors two nodes have, under the assumption that nodes with many neighbors are more likely to be connected than nodes with few neighbors. But it also down-weights those common neighbors based on their degree, assuming a neighbor with many connections is more likely to be common just by chance and therefore should contribute less compared to a neighbor with few connections when deciding if edge i, j exists. The use of $\log(k)$ in the denominator is meant to mitigate the effects of outliers: very high-degree nodes will still be down-weighted, but not as much as if k_u was used instead.

By itself, Adamic–Adar’s measure, or another scoring function, will not lead to a thicker network. Doing so requires incorporating that function into a decision process that declares “yes, i and j are connected” or “no, i and j are not connected.” One decision process is to define a threshold and insert any links that scored above that cutoff. But what cutoff to choose?

¹⁶ And nodes, if the two networks have different sets of nodes.

Here again the researcher is confronted with a choice and moving forward without enough information may be difficult. We recommend the following. First, determine if you have access to any information or criteria that can help guide you to a threshold. For instance, you may wish to add at most a given number of edges, say m new edges. Then, use that number instead: insert edges into the graph between the m highest rated pairs of nodes. And if you don't have something as simple as the number of new edges to guide you, perhaps you can look for another property of the newly thickened network. Perhaps you want to ensure the average degree is 5, or the diameter is 4. Then add the highest-scored node pairs until that property is achieved. Of course, it may not be possible to achieve such a desired property by adding in those new edges, so some trial-and-error may be involved.

As always, wherever feasible, perform robustness checks to see how any conclusions you wish to draw from the data are affected by the predictions. Comparing different networks under different prediction parameters is helpful (Ch. 14).

Keep in mind that while much of the literature has studied the problem of link prediction, most studies kept known results on hand for testing purposes. Often for instance, studies do not fully address the choices a researcher needs to make in practice, such as what scoring cutoff to use when deciding whether to predict a new link, instead devising a test scheme that either abstracts away this detail or evaluates different methods averaged over all possible cutoffs. In practice, you should not expect to have access to such ground truth when you're making actual predictions, and the results of these more abstract evaluations may not provide much guidance.

Ideally, a link prediction method will have a reasonable notion of confidence or certainty around its predictions: "We are 95% certain Alice and Bob are friends. We are 50% certain Bob and Carole are friends." Use this information, for example by mapping it onto link weights. Or extract a "high-confidence" backbone (discussed above) that retains observed links and those predicted links of which you are most confident. Without measures of certainty, predicted links may not be worth including, as it becomes too difficult to tell the reliability. This may happen with a machine learning "black box" approach to link prediction, although such methods may still be useful if they are deemed sufficiently trustworthy.

Suppose you have thickened a dataset by applying link prediction. It is good practice to distinguish predicted and observed links. First, for data provenance, you do not want subsequent users of the data to misinterpret a predicted link as an observed link, which may be disastrous. Second, for safety, you probably want to compare any analyses you do on the network with and without the predicted links included. How do your conclusions change due to using the predicted links? Therefore, incorporate a binary observed/predicted link attribute (Ch. 9) and retain it with your data moving forward.

Link prediction is fundamentally a machine learning solution; we discuss machine learning with networks in-depth in Ch. 16.

10.7 Other approaches

Combination methods

We have discussed thinning out spurious network elements and thickening up missing network elements as separate problems. In fact, consider applying both practices together. For example, pairing link prediction with a backbone extraction method. What's interesting to explore, but perhaps daunting itself, is the interactions between the different methods. If you first extract the backbone, the links predicted by a given scoring function may be quite different than if you first apply that scoring function. And likewise, if you first thicken the network with predicted links, then extract a backbone, the final backbone may be quite different. Keep in mind Occam's razor: a simple solution, all else being equal, is superior to one more complex. What is the fundamental problem you are addressing with the data and does it lead you to a complex, composite method? Only with an answer in the affirmative is it likely you will need or want the complexity of combining methods.

Statistical models and uncertainty quantification

In principle, although not always helpful in practice, one can build a statistical model (Ch. 23) for the probabilities of nodes or links to be valid, then query the data to see which network elements are missing or spurious. If the statistical model holds, and is computationally tractable for your network, then you can perform an inferential procedure to estimate high-likelihood links that do not exist in the data (even potentially nodes although this is especially tricky), or low-likelihood links (and nodes) that do exist in the data but perhaps should not. Coupled with a decision rule for when to add or remove links (or, again, nodes), you have now used statistical inference to thin and thicken the data by quantifying elements which are very certain or very uncertain (Ch. 24).

Besides relying on the accuracy and appropriateness of the statistical model, another challenge faced by this approach—and indeed, many approaches—is sparsity and imbalance. Most possible links do not actually appear in the network data, the average degree being much less than the number of nodes. This means it's likely to have a high rate of false positives, predicting nodes or links that do not exist, simply because there are so many more pairs of nodes than there are links. Keep this bias in mind.

10.8 Summary

After gathering data and extracting a network, investigating the network may reveal errors of omission—important structure is missing—or errors of commission—spurious structure is present. They are tricky to handle, but there are some techniques that can help. Use thickening techniques to get insights into omissions: link prediction can point you towards unseen links; graph union can allow you to merge your network with another. Use thinning techniques to address commissions. Thresholding by link weight, time window, or other means, can help you retain the strongest, most certain links;

backbone extraction can gather the critical subset of the network structure based on its properties. But remember that you should be careful; although link prediction may point you towards unseen links, you should not simply analyze the network with those predicted links included as if they are all correct. Backbone extraction methods can be useful but they are also *models* with strong assumptions and therefore should be treated as such.

Even when elements of the network structure are not in error, thinning and thickening techniques can be useful preprocessing for a variety of reasons. Algorithms may simply be challenged by too much density or analysis may be too difficult with so much structure to sift through.

There is not necessarily a bright line separating clearly the data gathering and network extraction tasks from thinning and thickening the data. A backbone extraction algorithm can be an element of a larger, iterated upstream task. The data are gathered, a preliminary network is extracted, spurious edges are removed, and a final network is obtained. This recapitulates in many ways the life cycle of a network study.

Bibliographic remarks

Taylor [455] is a truly lovely introduction to uncertainty and error analysis in scientific measurement. One of us (Bagrow) considers the cover photograph to be one of his favorite scientific illustrations.

Imputation is the field of statistics broadly interested in filling in missing values in data. Imputation has a long history dating back to seminal work by Rubin [406] and Dempster et al. [127] (see Rubin [407] for a history). van Buuren [469] is a more recent textbook on the field. Gelman and Hill [178] also has an excellent chapter about missing data.

Numerous backbone extraction and structural sparsification methods, used to thin networks, have been studied. The disparity filter we discussed here was introduced by Serrano et al. [424]. Other approaches include structural measures, especially for social networks, and spectral methods based on spectral properties of matrices such as the graph Laplacian that capture the network structure. Batson et al. [46] provides an overview of spectral methods (see also Ch. 25). There are also many methods that are specifically designed for networks constructed from associations (e.g., correlation matrices). This is a particularly common problem in systems biology. Soranzo et al. [439] provides an early overview of basic methods such as partial correlation.

Link prediction is a well-studied task in networks. Liben-Nowell and Kleinberg [273] provide an overview and Lü and Zhou [282] give a broader survey of work on the task.

Butts [87] discusses statistical aspects of errors in social networks, along with using Bayesian inference to tackle the errors. Other works of interest along these lines include Guimerà and Sales-Pardo [195], Martin et al. [295], and Newman [337].

Exercises

10.1 *Effects of disambiguation.* Suppose you built a network of actors who costarred in films (Ex. 6.1) as extracted from texts of the end credits of films. Doing so required disambiguating different actors. Suppose you are not confident in how the disambiguation was performed. Describe some errors or issues with the final network that may have occurred due to problems with the disambiguation. How likely or unlikely are the different issues?

10.2 (**Focal network**) Consider the Malawi Sociometer Network. Badges worn by participants (the nodes of the network) could measure proximity to other badges. And we add links between nodes when proximity occurs. Each badge, both its hardware and software, is subject to failure.

What happens to the network data if:

- (a) A subset of badges occasionally lose memory and no contacts are recorded?
- (b) Some badges use the wrong time when recording contacts?
- (c) Some badges are running different versions of the software and they cannot detect badges with another software version?

Describe some ways to fix these problems. Does the network being temporal help?

Can you think of any other problems due to the badges? Problems not due to the badges?

10.3 (**Focal network**) Consider a PPI network such as HuRI. These networks are generated by systematically testing pairs of proteins one at a time to see if an interaction exists. Suppose the network has N nodes and M edges. Those M edges were found by testing M_{test} protein pairs. What is the average degree in the observed network and what do you estimate it to be in the “true” network that would be found if every pair of nodes could be tested?

10.4 Describe the disparity filter (Sec. 10.5.5) in pseudocode using data structures and functions from Ch. 8.

10.5 (**Focal network**) Implement the disparity filter. Apply it to the Malawi Sociometer Network, reproducing Fig. 10.2. Compare these results to extracting the backbone with a global threshold. (See also Ex. 23.6.)

10.6 (**Simulation study**) Write a program that samples a graph’s edges uniformly at random. As edges (i, j) are sampled, track how often nodes i and j are observed. Apply this program to two focal networks of interest. (Be sure to average your results over the randomness of sampling.) Use a plot to visualize whether node observations are biased for or against high-degree nodes. How does the sampling rate of nodes relate to their degree?

