The ArcGIS Compendium — Volume D: Spatial Analysis

ON THIS PAGE

The ArcGIS Compendium — Volume D: Spatial Analysis

Asking questions of the map — proximity and overlay, statistics and patterns, street networks, terrain, and time.

One of eight volumes. Approximately 23206 words. Chapters cross-reference the whole set by number.

In this volume:

Proximity and Overlay: The Analysis Workhorses

Nearly every spatial question you will ever ask breaks down into two primitive operations: *how close are these things to each other?* (proximity) and *what happens where these layers overlap?* (overlay). "How many students live within a mile of the proposed school?" is a buffer plus a count. "Which parcels sit inside the flood zone?" is an overlay. "What's the nearest clinic to each patient?" is a closest-facility search. Master the half-dozen tools in this chapter and you can assemble answers to a startling share of real-world questions. The more sophisticated methods in Chapters 17 through 20 are mostly refinements stacked on top of these.

This chapter covers the vector side of analysis — operations on points, lines, and polygons. If the distinction between vector and raster is fuzzy, revisit Chapter 1 (How GIS Thinks). Raster analysis gets its own treatment in Chapter 19.

The Shape of Every Analysis Question

Before any individual tool, adopt a habit that will save you from most analysis mistakes. Every operation in this chapter fits the same four-part template:

1. **The question**, stated in plain English before you touch a tool. If you can't say what you're asking, no tool can answer it.
2. **The inputs** — which layers, and which of them is the "main" layer whose features you care about.
3. **The output** — always a *new* layer. Analysis tools never modify your inputs. Knowing exactly what kind of layer comes out (points? polygons? how many rows?) is half of understanding the tool.
4. **The sanity check** — a quick, concrete way to verify the result before you trust it. Every section below ends with one.

Where do these tools live? In ArcGIS Online's Map Viewer, open the Analysis pane from the toolbar on the right side of the map and browse or search the tool list — the proximity and overlay tools described here all appear there, sometimes under slightly different names than their ArcGIS Pro equivalents. In Pro, the same capabilities live in

the Geoprocessing pane — Pro's home for every analysis tool — reached via `Analysis > Tools`. This chapter names tools by the concept and notes both flavors where the names diverge; Chapter 22 (Geoprocessing in Depth) covers the Pro machinery — environments, licensing, batch runs — in detail.

Outputs behave differently in the two environments. Run a tool in Map Viewer and the result is saved to your content as a new hosted feature layer — a layer stored in and served from ArcGIS Online (Chapter 10 explains what that means for storage and sharing). Run it in Pro and the result lands in your project's geodatabase — the container file where Pro keeps local data — as a feature class. Same analysis, different plumbing.

> **Watch out:** In ArcGIS Online, analysis tools consume credits — the platform's usage currency — and the cost scales with how many features you process. Tools that use Esri's routing services (drive-times, travel-mode distances) cost noticeably more than simple geometry operations. Test on a small selection before running against fifty thousand features. Chapter 34 covers how credits are budgeted and monitored.

Buffers: Rings Around Things

The question: "What area lies within a given distance of these features?"

A buffer is a polygon drawn around a feature at a fixed distance — a circle around a point, a corridor around a line, an expanded ring around a polygon. It is the oldest trick in GIS and still the most used, because "within X distance of" is how an enormous number of policies, regulations, and business rules are written: notification zones around a rezoning parcel, protected strips along streams, trade areas around stores.

Inputs: any layer (points, lines, or polygons) plus a distance. The distance can be a single number applied to everything or read from a field, so each feature gets its own radius — useful when, say, each well has its own regulated setback. In Map Viewer the tool is called Create Buffers; in Pro it's the Buffer tool (recent versions also offer a faster Pairwise Buffer variant that covers most everyday cases).

Output: a polygon layer.

Planar versus geodesic

Here is the first place a wrong default quietly ruins an analysis. A **planar** buffer measures distance on the flattened map — as if you laid a ruler on paper. A **geodesic** buffer measures distance across the curved surface of the Earth. On a small area with a well-chosen local projection, the two are nearly identical. On a web map, they are not, because web maps almost always use the Web Mercator projection, which stretches distances more and more as you move away from the equator. A "10 kilometer" planar buffer drawn in Web Mercator over Alaska is nowhere near 10 real kilometers.

Chapter 3 (Coordinate Systems and Projections) explains why this distortion exists. What you need operationally is simpler:

- Web-based tools generally default to geodesic buffering, which gives true ground distance regardless of projection. This is almost always what you want.
- In Pro, the Buffer tool exposes a method choice (planar or geodesic), and its default behavior depends on the coordinate system of your input. Don't guess — open the tool's parameters and look.
- Planar is the right choice only when your data is in a good local projected coordinate system and you specifically want distances consistent with that projection — common in surveying and engineering contexts.

> **Watch out:** You cannot catch a planar-versus-geodesic mistake by eyeballing shape. Web Mercator preserves local shapes, so at ordinary buffer sizes both kinds draw as tidy circles — what differs is size on the ground. Check with the map's measure tool, which reports true ground distance: a "10 kilometer" buffer in Alaska whose radius measures out at about 5 kilometers was computed planar in Web Mercator units. (Only very large geodesic buffers, hundreds of kilometers across, visibly warp at high latitude — and that warping is the map's distortion, not an error.)

Dissolved versus separate

When buffers overlap — ten stores in a city, each with a 2-mile ring — you must decide whether to keep them as ten separate overlapping polygons or **dissolve** them into one

merged blob. Dissolving means melting shared boundaries away so overlapping shapes become a single feature.

Which you choose depends entirely on the *next* step:

- **Dissolve** when the question is about the covered area as a whole: "what part of the city is within 2 miles of *any* store?" One clean footprint, easy to map, easy to clip against.
- **Keep separate** when each feature needs its own answer: "how many customers are within 2 miles of *each* store?" You need one ring per store, carrying that store's attributes, so a later join or summary can count per ring.

Dissolving throws away per-feature attributes — a merged blob can't remember which store it came from. That's the trade.

> **Tip:** If you're not sure, keep the buffers separate. You can always dissolve later; you cannot un-dissolve.

Variations worth knowing

Three refinements come up often enough to name. **Multiple-ring buffers** create nested distance bands (1, 2, and 5 miles) in one pass — handy for "how does customer density fall off with distance?" **Negative buffers** shrink polygons inward instead of expanding outward, which is how you find the interior of a parcel excluding a setback strip. And when buffering **lines**, Pro lets you buffer one side only or choose flat versus rounded ends — details that matter for street corridors and riverbanks.

Sanity-checking a buffer

Use the map's measure tool to check two or three radii by hand — from a feature to the edge of its ring — in real-world units. If you didn't dissolve, confirm the output feature count equals the input count. And render the buffers semi-transparent so overlaps are visible; opaque fills hide exactly the structure you need to see.

Drive-Time Areas: Buffers That Respect Roads

The question: "What can actually *reach* this place within a given travel time or distance?"

A straight-line buffer assumes travel as the crow flies. People travel on roads, and roads have rivers without bridges, one-way pairs, and highways you can't cross on foot. A **travel area** (you'll also see the names drive-time area or service area) is the road-network equivalent of a buffer: the region reachable within, say, 10 minutes of driving from a point. Map Viewer's tool for this is called Generate Travel Areas in recent interfaces; Pro does the same job through its network analysis tools.

Inputs: point features (stores, fire stations, clinics), a travel limit expressed in time or distance, and a **travel mode** — a preset describing the traveler, such as driving a car, driving a truck, or walking. Some modes can account for typical traffic by time of day.

Output: polygons, usually irregular and spidery, following the road network outward from each point. Spiky, amoeba-like shapes are normal and correct — they trace fast roads outward.

The same dissolve-versus-separate decision from buffers applies here, for the same reasons.

This tool is a doorway into network analysis, which has its own chapter — Chapter 18 covers routing, service areas, and allocation in depth, including how the underlying street network and travel modes work. Use this section's version when you need reachability polygons as an input to further proximity work; go to Chapter 18 when the network itself is the star.

Gotchas. Travel areas run against Esri's routing services, which means credits in ArcGIS Online — appreciably more than plain buffers, so the test-small rule applies doubly. Points far from any road (a rural well, a point dropped in a lake) may fail or snap to a distant road with odd results. And ask honestly whether you need network travel at all: for a quick regional screening, a straight-line buffer is often good enough and much cheaper; save drive-times for when road reality genuinely changes the answer.

Sanity check: pick a spot on the edge of a 10-minute area and get directions from it back to the center point using any routing tool. It should come out near 10 minutes. If it's wildly off, check your travel mode — a walking-mode area mistaken for driving is a classic error, and it's visually obvious once you think to look: walking areas are tiny.

Spatial Joins: Attaching Attributes by Location

The question: "For each feature in layer A, what do I know about the features in layer B that relate to it *spatially*?"

A regular table join matches rows using a shared ID — parcel number to parcel number (see Chapter 12, Schema Design, for keys and relationships). A **spatial join** matches rows using geography instead: this point falls inside that polygon, so give the point the polygon's attributes. No shared ID needed; location is the key. It is arguably the single most useful operation in all of GIS, because it lets datasets that have never heard of each other answer questions together — customer points get a census-tract income figure, inspection reports get a council district, wells get the name of the aquifer beneath them.

Inputs: a **target layer** (the one whose features you're enriching — the output will have this layer's geometry and row structure) and a **join layer** (the one donating attributes). Getting these two straight is the biggest early stumbling block: "join tracts to points" and "join points to tracts" produce completely different outputs. Decide which layer's features you want to end up with, and make that the target. In Map Viewer the tool is Join Features; in Pro it's Spatial Join.

Output: a new layer with the target's geometry and a wider attribute table.

One-to-one versus one-to-many

What happens when one target feature matches *several* join features? A census tract contains hundreds of customer points; which customer's attributes does the tract get? The tools offer two answers:

- **One-to-one:** the output has exactly one row per target feature, and multiple matches are collapsed — either summarized with statistics you choose (count, sum, mean, min, max of numeric fields) or, if you ask for no statistics, reduced to a single arbitrary match. One-to-one with a *count* statistic is how you answer "how many points in each polygon," one of the most common questions in the field.
- **One-to-many:** the output has one row *per match*. A tract containing three hospitals becomes three rows, each with the same tract geometry but a different hospital's attributes. Row count grows, and the same geometry repeats.

One-to-many preserves full detail but sets a trap: those duplicated geometries will double-count in any later summation. If you one-to-many join hospitals to tracts and then sum tract population, every tract with three hospitals contributes its population three times. When you use one-to-many, treat the output as a lookup table, not as a layer to aggregate.

Match options

The spatial relationship that counts as a "match" is configurable, and choosing the right one is where the thinking happens. The core options (exact names vary slightly between Map Viewer and Pro):

MATCH OPTION	MEANS	EXAMPLE USE
Intersects	Features touch or overlap anywhere	Points in polygons; parcels touching a road
Within a distance	Features come within a stated distance	Customers within 500 m of a transit stop
Completely contains	Target wholly contains the join feature	Districts that fully contain a park
Completely within	Target sits wholly inside the join feature	Buildings entirely inside the floodplain
Closest	Nearest join feature, regardless of touching	Each incident tagged with its nearest station

"Intersects" is the usual default and the right choice most of the time. But boundaries are where it bites: a point lying exactly on the line between two polygons can legitimately match both, and two parcels that merely share an edge count as intersecting. If you mean "inside, not just touching," pick a within-style option. "Within a distance" is a buffer and a join fused into one step — often you can skip building buffer rings entirely.

The count field, your best friend

Most spatial join implementations add a field recording how many join features matched each target feature (Pro names it along the lines of a join count). Always look at it. Zero means no match — did you expect that? A huge number on one feature often flags a geometry problem, like a statewide polygon accidentally included among county boundaries.

Sanity checks

Spatial joins reward simple arithmetic. A one-to-one join that keeps all target features must output exactly as many rows as the target has — if it doesn't, stop. A one-to-many join outputs one row per matching *pair*; if 1,000 points each fall in exactly one of your polygons, expect 1,000 rows. Then pick one feature you know personally — your own neighborhood, a store you can picture — click it, and read the joined attributes. If your house got assigned to the wrong school zone, no summary statistic downstream can be trusted. Finally, investigate the non-matches: features with a zero count are either genuinely outside everything (fine) or a symptom of mismatched coordinate systems or bad geometry (not fine — Chapter 39 has the diagnostic trail).

Overlay: Intersect, Union, Erase, Clip

Overlay tools compute new geometry from the way two layers physically overlap — the layers are combined into a new set of shapes cut along every boundary from both. Four operations cover nearly everything; they differ in what they keep and whose attributes survive:

TOOL	GEOMETRY KEPT	ATTRIBUTES KEPT	TYPICAL QUESTION
Clip	Input features, trimmed to the clip area	Input layer's only	"Give me just my study area"
Intersect	Only the areas common to both layers	Both layers, combined	"Where do these conditions coincide?"
Erase (difference)	Input features minus the erase area	Input layer's only	"Everything *except* this zone"

TOOL	GEOMETRY KEPT	ATTRIBUTES KEPT	TYPICAL QUESTION
Union	Everything from both layers, diced	Both layers, blank where absent	"Split space into every combination"

In Map Viewer, intersect, union, and erase are grouped in a single Overlay Layers tool where you pick the method; clip usually doesn't appear as a separate web tool — intersecting with your study-area polygon achieves the same trim, with the study area's few fields tagging along for you to delete afterward. In Pro each operation is its own tool (with faster "pairwise" variants of several in recent versions — and since tool availability in Pro has historically varied with license level, if a tool appears missing, check what your license includes).

Clip: the cookie cutter

Clip trims one layer to the shape of another and keeps *only the input's attributes*, untouched. Nothing from the clip layer transfers. Use it for the everyday chore of cutting a national dataset down to your county so everything draws and processes faster. Because it never mixes attributes, clip is the safest overlay — but it has one famous trap, covered under "stale numbers" below.

Intersect: where conditions coincide

Intersect keeps only the regions where features from both layers overlap, and the output rows carry attributes from *both* parents. That combination is the point: intersect soils with zoning and every output polygon knows both its soil type and its zoning class, which is exactly what a suitability question needs. Two subtleties. First, the output geometry defaults to the "lowest" dimension of the inputs — intersecting a line layer with polygons yields lines; points with polygons yields points (Pro lets you choose the output type explicitly). Second, if a feature overlaps several features in the other layer, it appears once per overlap, so rows multiply just as in a one-to-many join, with the same double-counting hazard.

Erase: the anti-clip

Erase (called difference in some interfaces) keeps the parts of the input *outside* the erase layer — clip's mirror image. It answers "everything except": developable land is

the parcel layer with wetlands, steep slopes, and existing buildings successively erased away. Chained erases like that are a natural fit for ModelBuilder (Chapter 25).

Union: every combination

Union keeps everything from both layers, slicing space along every boundary so each output polygon represents one unique combination of inputs. Where only one layer exists, the other layer's fields are blank. Union is generally a polygon-on-polygon operation, and its output feature count can be startlingly large — that's not a bug, it's the definition. Reach for it when you truly need the full combinatorial fabric ("classify every patch of the county by both flood risk and zoning, including areas in one but not the other"); reach for intersect when you only care about the overlaps.

Slivers and stale numbers

Two failure modes haunt all overlay work.

Slivers are the confetti of tiny, needle-thin polygons produced when two layers have boundaries that *almost* coincide — a coastline digitized twice by two agencies never matches exactly, and every near-miss becomes a sliver. Detect them by sorting the output's area column ascending: a swarm of absurdly small areas at the top is your evidence. The honest fixes are upstream — use layers derived from a common source where possible, and see Chapter 14 (Data Quality) for topology, the enforceable rules about how boundaries must relate, which is how you reconcile boundaries properly. Deleting slivers below a size threshold is a common pragmatic cleanup, but decide deliberately rather than letting them silently pollute statistics.

Stale numbers are subtler. Overlay changes geometry, but ordinary numeric fields don't know that. Clip a 100-acre parcel in half and the output still says 100 acres in its user-entered acreage field, because that's just a number in a table. Any field derived from geometry — area, length, population estimated from area — is untrustworthy after overlay until recalculated.

> **Watch out:** After any overlay, recompute geometry-derived fields before using them. The built-in shape area and length fields that geodatabase feature classes and hosted layers maintain do update automatically — though they're measured in the layer's own coordinate-system units, so treat them with the same planar-

distance caution as any projected measurement — but any acreage, length, or population field *you* created is never auto-updated. In Pro, recalculate with Calculate Geometry on the attribute table. The 100-acres-says-the-half-parcel error has embarrassed more analysts than any other in this chapter.

Sanity checks

Overlay obeys conservation laws you can test. The clipped output's total area can't exceed the input's. Clip plus erase of the same input against the same layer should reassemble, in total area, to the original input. Union's footprint should equal the combined footprint of both inputs. A minute with the attribute table's statistics summary verifies any of these. Then click one output polygon at a known location and confirm its combined attributes match what you'd expect standing there.

Summarize Within and Summarize Nearby

The question: "For each polygon (or each zone around a feature), what are the statistics of another layer's features inside it?"

You could answer this with buffers plus a spatial join plus statistics — and under the hood, that's roughly what these tools do. They exist because the pattern is so common it deserves a one-step wrapper.

Summarize Within takes a polygon layer (counties, districts, hexagonal bins) and a layer to summarize (crime points, road lines, land-cover polygons), and returns the polygons with new statistic fields: count of points, total road length, mean parcel value, and so on. A group-by option splits the statistics by a category field — crimes per county *by offense type* — which otherwise takes tedious manual work.

Summarize Nearby does the same, except the summary zones are generated on the fly around each input feature — a straight-line ring or a travel-time area — so "population within a 10-minute drive of each store" is one tool run instead of three. Travel-mode zones consume routing credits, as with travel areas above.

Gotchas. Three deserve respect. First, *partial features*: a road crossing a county line contributes only the length inside each county — good — but how the tools treat a partial feature's other numeric attributes (proportionally split versus counted whole)

varies by tool and option. When a sum matters, verify against one polygon computed by hand before trusting the batch. Second, *overlapping zones* in Summarize Nearby: if two stores' 10-minute areas overlap, people in the overlap are counted toward both stores. Per-store, that's usually correct; summed across stores, it's not a total market count. Third, the *density mistake*: a raw count per polygon mostly maps polygon size and population, not the phenomenon. Divide by area or population before drawing conclusions — Chapter 7 (Styling and Smart Mapping) covers normalization in mapping, and Chapter 17 picks up the statistical side.

Sanity check: pick one polygon, select the summarized layer's features inside it by hand (a selection rectangle or interactive select is fine), and compare the manual count to the tool's answer. Then check the grand total: if your polygons tile the area without overlap and all features fall inside one, the column total should equal the layer's feature count. A shortfall means features are falling outside every polygon — often a projection or geometry symptom worth chasing.

Find Closest

The question: "For each feature in one layer, which feature in another layer is nearest — and how far away is it?"

Assign every incident to its nearest fire station, every customer to the nearest branch, every school to the nearest hospital. In web tools the operation appears as Find Closest (older interfaces called it Find Nearest); Pro reaches the same answers through Near-style geoprocessing tools or, for road-based versions, the network analysis tools of Chapter 18.

Inputs: an origin layer, a candidate layer, and the measurement choice — straight-line distance or a travel mode (which, again, means routing credits). Two options keep output manageable: how many nearest candidates to find per origin (just the single closest, or the top few), and an optional cutoff so origins farther than a limit from everything simply return no match.

Output: typically the matched pairs with distance and rank attributes, plus — usefully — a line layer connecting each origin to its found candidates. Those connector lines are the best diagnostic in the toolset: put them on the map and misassignments jump out visually in a way no table can match.

Gotchas. Straight-line and travel distance disagree exactly where it matters most — rivers, ridgelines, highway barriers. The clinic 400 meters away across an unbridged river is not the closest clinic by any measure a patient cares about. Use straight-line for screening and physical phenomena; use a travel mode whenever a person or vehicle does the traveling and barriers exist. Also watch near-ties: when two candidates are almost equidistant, tiny data errors decide the winner, so avoid building brittle rules on razor-thin distance differences.

Sanity check: pick one origin, measure from it to its assigned candidate and to the two or three plausible rivals with the measure tool, and confirm the tool's pick really is shortest. Confirm the distance field's units (meters and feet mixed up survive every visual check). And scan the connector lines for any that look absurdly long — those are usually origins that matched a distant candidate because nearer ones were missing from the candidate layer.

Choosing the Right Tool

YOU WANT TO...	REACH FOR
Draw fixed-distance zones around features	Buffers
Draw realistic reach-by-road zones	Travel areas (drive-time)
Attach one layer's attributes to another by location	Spatial join
Count points in polygons	Spatial join (one-to-one, count) or Summarize Within
Trim data to a study area	Clip
Find where two conditions coincide, with both sets of attributes	Intersect
Remove zones from consideration	Erase
Split space into every combination of two layers	Union

YOU WANT TO...	REACH FOR
Compute statistics per polygon	Summarize Within
Compute statistics around each feature	Summarize Nearby
Pair each feature with its nearest counterpart	Find Closest

Notice how many rows are near-duplicates. That's the deeper lesson of this chapter: these tools overlap, and most questions have two or three valid tool paths. "Customers within 500 m of a stop" is a buffer-then-join, a within-a-distance join, or a Summarize Nearby. When paths disagree in convenience but agree in answer, pick the fewest steps; when you're unsure of a result, running a second path and comparing is itself a powerful check.

A Sanity-Check Discipline

Pulling the per-tool checks into one habit loop:

1. **Test small first.** Run against a selection or a small clipped area, inspect thoroughly, then scale up. This protects both your credits and your confidence.
2. **Do the row arithmetic.** Before running, predict the output feature count from the tool's logic. After running, explain any difference. Unexplained counts are unfinished analysis.
3. **Spot-check one known feature.** Pick a place you know personally and verify the tool's answer there by hand with the measure tool and pop-ups.
4. **Recalculate geometry-derived fields** after anything that cuts or merges shapes.
5. **Mind the coordinate systems.** The tools reproject on the fly, but distance-based operations are only as honest as the measurement method — reread the planar-versus-geodesic section whenever a distance looks off, and Chapter 3 when it stays off.

Tip: Encode the parameters in the output layer's name — something like `stores_buffer_2mi_geodesic_dissolved`. Three weeks later, "Buffer_output_7"

tells you nothing, and re-running an analysis because you can't reconstruct its settings is the most avoidable waste in GIS work.

With these workhorses in hand, the rest of Volume D reads as extensions: Chapter 17 adds statistical rigor to the summaries you can now compute, Chapter 18 deepens the road-network side of proximity, and Chapters 19 and 20 carry the same question-inputs-check discipline into rasters and time. When a result refuses to make sense despite every check here, Chapter 39 (The Troubleshooting Encyclopedia) is the place to take it.

Statistical and Pattern Analysis

Look at any map of points long enough and you will see patterns. A knot of burglaries near the highway. A run of asthma cases along the river. The trouble is that the human eye finds patterns in everything, including pure randomness — scatter confetti on the floor and some of it will land in clumps. Spatial statistics exists to answer one deceptively hard question: **is the pattern I see real, or is it the kind of clumping that random chance produces all the time?**

This chapter walks through the pattern-analysis tools you will actually use — hot spot analysis, cluster and outlier analysis, density surfaces, and introductory regression — with special attention to the two places people get hurt: misreading what "statistically significant" means, and letting the boundaries you drew shape the answer you got. By the end you should be able to run these tools, defend the parameters you chose, and — most importantly — describe the results out loud without saying something the analysis does not support.

The mechanics of *where* these tools live are simple. In Map Viewer, open the Analysis pane from the map's toolbar and browse or search its tools for Find Hot Spots, Find Outliers, or Calculate Density. In ArcGIS Pro, they live in the Spatial Statistics and Spatial Analyst toolboxes, reachable through `Analysis > Tools` and the search box.

Chapter 22 (Geoprocessing in Depth) covers the general machinery of running tools; this chapter is about what the pattern tools mean.

> **Tip:** Analysis run in Map Viewer against hosted layers consumes credits — the usage currency of ArcGIS Online, covered in Chapter 34 (Administration). The same statistical tools in ArcGIS Pro run locally and consume no credits. For exploratory work where you will rerun a tool a dozen times while tuning parameters, Pro is usually the more economical home.

Spatial autocorrelation: the idea underneath everything

Nearly every tool in this chapter rests on one concept, so it is worth getting comfortable with it before touching any buttons.

Spatial autocorrelation is the tendency of nearby things to be more alike than distant things. Home prices on the same street resemble each other. Rainfall at two weather stations a mile apart is more similar than at stations five hundred miles apart. Geographers summarize this as Tobler's first law of geography: *everything is related to everything else, but near things are more related than distant things.*

Why does this matter? Two reasons.

First, it is the thing pattern tools measure. When you ask "are high crime counts clustered?", you are literally asking "do high values sit next to other high values more often than chance would produce?" That is a question about spatial autocorrelation.

Second, it breaks the assumptions of ordinary statistics. Classical statistics assumes your observations are independent — that one measurement tells you nothing about the next. Spatial data violates this constantly. If you treat two hundred neighboring census tracts as two hundred independent samples, you are effectively counting the same information multiple times, and your confidence in any conclusion will be inflated. Spatial statistics is, at heart, ordinary statistics rebuilt to stop lying about spatial data.

A pattern can land in one of three broad states:

- **Clustered** — high values near high values, low near low, more than chance would produce.

- **Dispersed** — high and low values alternate more regularly than chance, like squares on a checkerboard. Rare in nature, but it happens (competing retail stores spacing themselves apart, for instance).
- **Random** — no detectable structure. The null hypothesis, the boring default every test is trying to reject.

Tools that measure this come in two flavors. **Global** statistics (the classic is a tool called Spatial Autocorrelation, based on a measure named Moran's I) give you a single number for the whole dataset: "yes, overall, this data is clustered." They are a useful sanity check but they draw nothing on the map. **Local** statistics — the hot spot and outlier tools that make up the rest of this chapter — compute a value for *every feature*, telling you *where* the clustering is. The global tool answers "is there a pattern?"; the local tools answer "where is it?"

> **Tip:** Distance-based statistics need honest distances. Run these tools on data in a projected coordinate system appropriate to your study area, not raw latitude-longitude, where a "degree" is a different real-world distance north-south versus east-west. Chapter 3 (Coordinate Systems and Projections, Complete) explains how to choose and apply one.

Hot spot analysis, demystified

Hot spot analysis is the most-run and most-misread tool in spatial statistics. The classic implementation is called **Getis-Ord Gi*** (pronounced "G-i-star", named for the statisticians Arthur Getis and J.K. Ord). In Map Viewer the tool is called Find Hot Spots; in Pro it appears as Hot Spot Analysis and an assistant version called Optimized Hot Spot Analysis that picks reasonable settings for you.

What the tool actually computes

Here is the whole algorithm in plain language. For every feature in your data — every census tract, every grid cell, every point — the tool:

1. **Defines a neighborhood** around that feature: everything within a certain distance, or its adjacent polygons, or its nearest handful of neighbors, depending on your settings.

2. **Adds up the values** in that neighborhood, *including the feature itself* (that is what the asterisk in Gi* signifies).
3. **Compares that local sum to what a random arrangement of the same values would produce.** Imagine shuffling all the values across all the features thousands of times and asking how often a neighborhood sum this high (or this low) turns up by pure luck. (The tool computes that expectation mathematically rather than literally shuffling, but the logic is the same.)

The output for each feature is a **z-score** — a measure of how many standard deviations the local sum sits above or below the random expectation — and a **p-value**, the probability of seeing a result at least that extreme if the data were actually random. Big positive z-score with a tiny p-value: a **hot spot**, a statistically unusual concentration of high values. Big negative z-score: a **cold spot**, an equally real concentration of *low* values. Everything else: not distinguishable from noise.

The rendered map bins these into confidence levels — typically hot and cold at 90%, 95%, and 99% confidence, with everything else drawn as "not significant." A feature in the 99% hot bin means: if the values were randomly arranged, a neighborhood this concentrated would appear less than one time in a hundred.

Notice something important in step 3: the comparison baseline is *the whole study area*. "Hot" does not mean "high." It means "surrounded by values that are high **relative to everything in this dataset**, to a degree random chance rarely produces." A neighborhood of moderately elevated values embedded in a low-value region can be a legitimate hot spot. A single skyscraping value surrounded by low ones usually is not — one loud feature does not make a neighborhood.

The choices that shape the answer

Two decisions matter far more than any other, and you should be able to defend both.

The input value. Gi* needs a numeric value on each feature to analyze. If you have polygons with counts or rates, you are ready. If you have raw incident points — one point per crime, per crash, per sighting — there is nothing to sum, so the points must first be aggregated: counted into a grid of cells or hexagonal bins, or joined to administrative polygons. The online Find Hot Spots tool and Pro's optimized tool will offer to do this aggregation for you. This is convenient, but the choice of cell size

quietly becomes part of your analysis — hold that thought until the section on aggregation effects.

The neighborhood. How far is "nearby"? A fixed distance band is the most defensible default: every feature's neighborhood is everything within, say, a kilometer. Too small a distance and features have no neighbors, producing unstable, speckled results. Too large and everything blurs into one lukewarm smear. The optimized tools estimate a distance from the spacing of your data, which is a reasonable starting point — but if your question implies a scale ("do neighborhoods differ?" versus "do regions differ?"), let the question set the distance and say so in your write-up.

> **Watch out:** Running hot spot analysis on raw *counts* over areas where people live is the single most common blunder in the genre. More people means more of nearly everything — crimes, ambulance calls, coffee shops, dog licenses. A hot spot map of raw counts is usually just a population map wearing a costume. Analyze *rates* (incidents per thousand residents, crashes per vehicle-mile) whenever your phenomenon scales with population, and the tools provide a field for exactly this normalization. Chapter 7 (Styling and Smart Mapping) makes the same point about shading areas by raw counts — it applies double here.

What "statistically significant" actually means

This phrase does more damage than any other in applied statistics, so let us pin it down.

"Statistically significant" means exactly one thing: **the observed pattern would be unusual if the underlying process were random.** That is all. It is a statement about *surprise*, not about *size*, *importance*, or *cause*.

What it does **not** mean:

- **It does not mean the effect is large.** With enough data, trivially small differences become statistically significant. A hot spot can be real and also too mild to matter for any decision anyone will ever make.
- **It does not mean the finding is important.** Importance is a judgment call involving stakes, costs, and context. The p-value has no opinion about whether anyone should

care.

- **It does not explain anything.** The tool has confirmed *concentration*, not *cause*. A hot spot of emergency calls might reflect where emergencies happen — or where phones work, where people trust the system enough to call, or where a records office geocodes (assigns map coordinates to) unknown addresses at a single default point.
- **It is not immune to false positives.** At 95% confidence, roughly one in twenty truly random neighborhoods will light up anyway. Run a test across thousands of features and some "significant" results are statistical noise wearing a badge. Pro's hot spot tools offer a correction for this called FDR (False Discovery Rate) — optional in the standard tool, applied automatically by the optimized one — which tightens the threshold when many tests run at once. Turning it on typically shrinks your hot spots and increases your honesty.

> **Watch out:** The most seductive misreading of all is treating the *edge* of a rendered hot spot as a real boundary — "crime stops at Maple Street." The colored bins are confidence thresholds crossing gradual gradients, not walls. Nothing changes abruptly at the 95% contour except the symbology.

One more subtlety: results depend on the **study area you chose**. Gi* compares each neighborhood to the average across your whole dataset. Analyze one city and a middling neighborhood may look cold; analyze the metro region and the same neighborhood may look hot, because the baseline changed. Neither map is wrong. They answer different questions, and your write-up must say which question you asked. Related to this: features at the edge of your study area have truncated neighborhoods (their neighbors beyond the boundary are invisible to the tool), so treat significance right at the border with extra skepticism.

Cluster and outlier analysis: the sharper sibling

Hot spot analysis paints regions. Its sibling — **Cluster and Outlier Analysis** in Pro, **Find Outliers** in Map Viewer, built on a statistic called Anselin Local Moran's I — asks a subtler question about each feature: *does this feature's value match its neighbors, or clash with them?*

Its output sorts significant features into four categories:

CATEGORY	PLAIN MEANING	EXAMPLE
High-High (HH)	High value surrounded by high values — a cluster core	Expensive house in an expensive neighborhood
Low-Low (LL)	Low value surrounded by low values — a cold cluster core	Vacant parcel in a disinvested block
High-Low (HL)	**Outlier:** high value surrounded by low values	Luxury tower in a struggling district
Low-High (LH)	**Outlier:** low value surrounded by high values	The one cheap holdout on a gentrified street

The HH and LL categories broadly echo what hot spot analysis shows. The outlier categories are the reason this tool exists. Outliers are where the interesting stories live: the clinic whose outcomes beat every neighbor's, the store that underperforms in a thriving corridor, the census tract that broke a regional trend. Outliers are also where the *data errors* live — a High-Low outlier is exactly what a misplaced decimal point or a bad geocode looks like, which makes this tool a quiet workhorse for the quality-assurance workflows in Chapter 14 (Data Quality).

Choosing between the two tools comes down to your question:

YOU WANT TO KNOW...	USE
Where are the broad regions of concentrated high or low values?	Hot Spot Analysis (Gi*)
Which specific features clash with their surroundings?	Cluster and Outlier Analysis (Local Moran's I)
Both — the regions and the exceptions	Run both; they are complementary, not redundant

Everything said above about neighborhoods, rates versus counts, study-area dependence, and the meaning of significance applies here unchanged. Same foundation, different question.

One naming trap worth defusing: ArcGIS also ships tools with "clustering" in the name — Density-based Clustering, for example — that do something different. Those group point *locations* into clumps based purely on geometry, ignoring any value attached to the points. The tools in this section analyze *values* arranged in space. "My points are bunched together" and "my high values are bunched together" are different findings; make sure you know which one you are reporting.

Density surfaces versus heat map styling

Two things in ArcGIS produce glowing blobs, and confusing them is one of the most common mistakes in the entire platform. They look alike. They are not alike.

Heat map styling is a *rendering* choice — one of the smart mapping styles covered in Chapter 7. It redraws your point layer with soft glowing intensity, recalculating as you zoom. It is fast, needs no analysis tool, and consumes no credits. It is also **not analysis**. The glow has no units, no defensible numbers behind it, and it changes shape as the map scale changes — zoom out and separate clumps melt together; zoom in and they split apart. Two colleagues looking at the same heat map at different zoom levels are looking at different pictures.

Density analysis — the Kernel Density tool in Pro, Calculate Density in Map Viewer — is a *geoprocessing operation* that produces a real dataset: a raster (a grid of cells, the data model from Chapter 1) in which every cell holds an actual number with actual units, such as incidents per square kilometer. Conceptually, the tool places a small smooth mound of "influence" over every point, then sums the overlapping mounds into a continuous surface. The mound's width — the **search radius**, or bandwidth — is the parameter that matters: small radius, spiky detailed surface; large radius, smooth general one. There is no single correct value; there is only a value you can justify against the scale of your question, stated plainly in your write-up.

	HEAT MAP STYLING	DENSITY ANALYSIS
What it is	A way of *drawing* points	A computation producing a new raster dataset
Changes with zoom?	Yes — the picture is scale-dependent	No — the data is fixed once computed
Has real units?	No	Yes (e.g., events per unit area)
Cost	Free, instant	Processing time; credits if run online
Defensible in a report?	As illustration only	Yes, with the search radius disclosed
Right use	Quick visual exploration, dense point layers	Any result someone will measure, compare, or act on

A reasonable workflow uses both. Explore with heat map styling to get a feel for the data, then, once you know what question you are asking, run Kernel Density with a deliberate search radius to produce the citable version. If anyone will ever ask "how much, exactly, and where?", you need the raster, not the glow.

Density surfaces also differ from hot spot analysis in a way worth one sentence: density shows you *where values are high*; hot spot analysis shows you *where concentrations are statistically unusual*. A density surface makes no claim about significance at all. For deeper raster mechanics — cell sizes, resampling, map algebra — see Chapter 19 (Terrain and Raster Analysis).

Regression and prediction, at an honest level

Pattern tools tell you *where* something concentrates. Sooner or later someone asks *why* — and whether you can predict it somewhere else. That is regression territory. This section is a map of the terrain and its cliffs, not a full statistics course.

What regression does

Regression builds an equation relating a thing you want to explain (the *dependent variable* — say, house price) to things you believe drive it (*explanatory variables* — square footage, age, distance to transit). The classic method, **ordinary least squares** (OLS, available in Pro under names like Generalized Linear Regression), finds the single equation that best fits all your data at once, then tells you how much each variable contributes and how much of the variation the model explains.

Two outputs deserve your attention even in a first encounter. The **coefficients** say how the dependent variable moves when an explanatory variable moves. The **residuals** — the leftover error for each feature, the gap between what the model predicted and what actually happened — are where a spatial analyst looks first.

Why space complicates it

Map the residuals. If the model has captured the real drivers, its errors should be scattered randomly — no pattern, no story. If instead the residuals *cluster* — the model systematically overpredicts in one region and underpredicts in another — the model is missing something geographic, and its confidence statements are not trustworthy. (You can test this formally: run the Spatial Autocorrelation tool on the residuals. Clustered residuals mean an incomplete model. This one habit — *always map the residuals* — separates careful spatial regression from careless spreadsheet regression.)

The deeper issue is that one global equation assumes the relationship is the *same everywhere*. Often it is not. Proximity to a busy road may depress house prices in a quiet suburb and raise them in a walkable urban core. **Geographically weighted regression** (GWR) addresses this by fitting many local equations instead of one global one — each centered on a location and weighted toward nearby data — producing coefficients that themselves vary across the map. That map of varying coefficients is often the most interesting output: it shows you *where* a relationship is strong, weak, or reversed. The cost is complexity: more parameters to justify, more ways to fool yourself, and a hunger for reasonably large, well-distributed datasets.

The platform also includes machine-learning prediction tools — forest-based regression and its relatives — which learn flexible relationships from training data and predict values where you have none. They can capture patterns straight-line models miss, and they helpfully report which variables mattered most. But they inherit every caveat in

this section plus one of their own: they are excellent at fitting the data you gave them, which is not the same as being right about data you did not.

An honest checklist before you trust any model

- **Correlation is not causation.** Regression finds association. Ice cream sales predict drowning deaths because both follow summer, not because sundaes are dangerous. No output table can tell you which way causation runs, or whether a lurking third variable drives both.
- **Missing variables masquerade as spatial pattern.** Clustered residuals often just mean an important driver was left out — and the things most often left out (history, policy, wealth, infrastructure) are themselves spatially clustered.
- **Beware the too-good fit.** A flexible model can memorize noise. If performance is stellar on the data it trained on, ask how it does on data it has never seen — that is the only measure that counts for prediction.
- **Prediction degrades outside the training data's range** — in variable values, in geography, and in time. A model trained on one city's neighborhoods has earned no opinion about another city's.
- **State what the model is for.** "Explains about half the variation in our study area" is a legitimate, useful finding. It becomes illegitimate the moment it is presented as a causal law or a guarantee.

Aggregation effects: MAUP in plain terms

Here is an uncomfortable truth that sits underneath every polygon-based result in this chapter: **the answer depends on the boundaries, and the boundaries were a choice.**

Statisticians call it the **modifiable areal unit problem** — MAUP. "Areal unit" just means the polygon you aggregated into (tract, grid cell, county); "modifiable" means someone could have drawn it differently. It shows up in two ways:

The scale effect. Aggregate the same incidents into small hexagons and you may see sharp, significant hot spots. Aggregate into counties and the pattern may dilute into nothing — or a different pattern may emerge. Neither is the "true" answer; pattern genuinely exists at some scales and not others.

The zoning effect. Even at a fixed size, *where the lines fall* changes the result. Shift a grid half a cell sideways and a cluster that previously straddled four cells — splitting its strength four ways, none significant — may land inside one cell and blaze. Same reality, different verdict. If this sounds like gerrymandering, it is exactly the same mathematics: gerrymandering is the deliberate weaponization of the zoning effect, and every honest analyst is fighting the accidental version.

A close cousin deserves its own name: the **ecological fallacy** — concluding things about *individuals* from statistics computed over *areas*. "This tract has high average income and high rates of condition X, therefore wealthy people have condition X" does not follow; within that tract, the condition may sit entirely among its poorest residents. Area-level findings describe areas. Full stop.

You cannot eliminate MAUP. You can refuse to let it operate in the dark:

- **Prefer meaningful units when they exist.** School catchments for a school question, watersheds for a water question. When no natural unit exists, regular hexagons at least make the arbitrariness uniform and honest.
- **Run the sensitivity test.** Repeat the analysis at two or three aggregation scales. Findings that survive across scales are robust; findings that appear at exactly one cell size are findings *about that cell size*, and should be reported that way or not at all.
- **Disclose the unit, always.** "Clustered at the tract level" is a complete, honest claim. "Clustered" alone is not.

> **Watch out:** MAUP is the mechanism behind most cases of dueling maps — two analysts, one dataset, opposite conclusions, both technically correct. If a result matters enough to act on, it matters enough to check at a second aggregation scale first.

Saying what you found without overclaiming

Every technique in this chapter converges on a writing problem. The analysis is usually the easy part; the sentence you put under the map is where the trouble starts. Some working rules:

Report the machinery, not just the verdict. A pattern finding is only interpretable alongside its parameters. The honest minimum: the input variable (and whether it was a rate or a count), the aggregation unit, the neighborhood or search radius, the confidence level, the study area, and the time window. That sounds bureaucratic; it is one sentence: "Emergency calls per thousand residents, by hexagonal bin, showed statistically significant clustering (95% confidence, one-kilometer neighborhoods) in two districts during 2025."

Describe concentration, not cause. The tools in this chapter detect *arrangement*. They are silent on *mechanism*. Keep the finding and the interpretation in separate sentences, and label the second one as interpretation.

Keep significance and importance in separate clauses. "Statistically significant" answers "is it likely real?" Whether it is *big enough to matter* is a different question that the analysis cannot answer — so answer it yourself, with effect sizes and context, or leave it explicitly open.

A phrasebook of common overclaims and their honest replacements:

OVERCLAIMED	HONEST
"Crime is concentrated on the east side."	"Reported crime per capita clusters significantly on the east side at the tract level — noting that reporting rates may themselves vary by area."
"The hot spot proves the intersection is dangerous."	"Crashes cluster significantly around this intersection; the analysis identifies concentration, not its cause."
"Poverty causes the disease pattern."	"Disease rates are associated with poverty rates across tracts; area-level association cannot establish cause, or describe individuals."
"The model predicts prices accurately."	"The model explains roughly half of price variation within the study area; accuracy beyond it is untested."
"There is no pattern in the south."	"No significant clustering was detected in the south at this scale and sample size — absence of evidence, not evidence of absence."

That last row deserves its own paragraph, because it is the overclaim people never notice themselves making. A "not significant" result does not mean nothing is happening. It means *this test, at this scale, with this data* could not distinguish the arrangement from chance. Sparse data, a mismatched neighborhood distance, or an unlucky aggregation grid can all hide real structure. Negative findings should be phrased with the same parameter honesty as positive ones.

> **Tip:** Before publishing any pattern map, run the colleague test: show it to someone uninvolved and ask what they think it says. Whatever they answer is what your map *actually* communicates, whatever your methods section claims. If they say "the east side is dangerous" and your finding was "reported incidents per capita cluster at 95% confidence," the map needs a better title, a plainer legend, and probably a caveat in the description — the pop-up and labeling techniques in Chapter 9 are your friends here.

Where to go from here

The tools in this chapter treat time as frozen — one snapshot, analyzed once. Real phenomena drift, pulse, and migrate, and there is a whole family of tools (space-time cubes, emerging hot spot analysis) for finding patterns that move; Chapter 20 (Space-Time Analysis) picks up that thread directly, and much of this chapter's vocabulary — significance, neighborhoods, honest phrasing — carries over unchanged. For the overlay and proximity operations that often *prepare* data for these statistics, see Chapter 16; for automating a pattern-analysis workflow you will rerun monthly, ModelBuilder (Chapter 25) or Python (Chapter 31) will save you from re-clicking every parameter by hand.

The habit to carry forward is smaller than any tool: every pattern claim is really three claims fused together — about the data, about the parameters, and about the world. The analysis only ever certifies the middle one. Say all three out loud, and you will overclaim less than almost everyone else drawing glowing blobs on maps.

Network Analysis: Routing, Service Areas, and Allocation

Draw a circle with a two-mile radius around a fire station and you have made a claim: everything inside this circle is close to the station. Now picture a river running through that circle with one bridge, three miles upstream. Half of your "close" area is actually a twenty-minute drive away. The circle lied, because circles measure the world as the crow flies, and fire trucks are not crows.

Network analysis is the branch of GIS that measures the world the way people and vehicles actually move through it: along streets, obeying one-way signs, slowing for school zones, stopping at the river until there is a bridge. This chapter covers the whole family of network tools in ArcGIS — routing between points, drive-time polygons, finding the closest facility, building travel-cost tables, and choosing where to put new facilities in the first place. It also covers the practical matters that trip people up: travel modes, barriers, and the fact that most network analysis in ArcGIS Online runs against Esri's servers and consumes credits (ArcGIS Online's usage-based currency, which your organization buys and spends on certain services).

If you have read Chapter 16 (Proximity and Overlay), you already know buffers and straight-line distance tools. Everything in this chapter exists because those tools, useful as they are, ignore the street network. When the question is "how far apart are these things, really, for a person traveling between them," this chapter is the answer.

Why Straight Lines Lie

Straight-line distance — sometimes called Euclidean distance, after the geometry you learned in school — treats space as an open field. Network distance treats space as a maze with rules. The gap between the two is not a rounding error. In a city with a tidy grid of streets — the friendliest possible case — network distance still averages roughly a quarter to a third longer than straight-line distance. Add rivers, highways with limited exits, cul-de-sac suburbs, one-way systems, and rush-hour traffic, and the two measurements stop having any reliable relationship at all.

This matters because most of the questions people bring to network analysis are about *access*: Who can reach this clinic within fifteen minutes? Which warehouse should serve this store? Where should the city put its next library? Answering an access

question with straight-line geometry produces answers that look plausible on a map and are wrong on the ground — often wrong in a biased way, because the neighborhoods most poorly served by the street network (cut off by highways, rivers, or rail yards) are exactly the ones a straight-line analysis flatters most.

What a network dataset actually is

Under the hood, every network tool runs against a **network dataset**: a specialized data structure built from line features, typically street centerlines. Three ideas make it more than a pile of lines.

First, **connectivity**. The network knows which street segments actually connect to which. Two lines crossing on the map are not necessarily connected — an overpass crosses the street below it without touching. A network dataset stores junctions (places where you can turn from one segment onto another) explicitly, so the **solver** — the engine that computes answers against the network — never routes you off an overpass into thin air.

Second, **costs**, which network people call **impedance**. Every segment carries one or more costs: its length, and usually the time it takes to traverse it at some assumed speed. When you ask for the "best" route, you are really asking the solver to minimize a cost — usually minutes, sometimes miles. The same pair of points can have a shortest route and a fastest route that differ completely.

Third, **restrictions**. One-way streets, turn prohibitions ("no left turn"), vehicle limits (no trucks over a certain height), pedestrian-only paths. Restrictions tell the solver which movements are forbidden for a given kind of traveler. This is why the network can give you a walking route through a park where it would never send a car.

You will rarely build a network dataset yourself. Esri maintains a global, regularly updated street network — the same data behind its routing services — and for most users the practical choice is between using that hosted network through ArcGIS Online, or licensing street data (Esri's StreetMap Premium, or open data such as OpenStreetMap) to build a local network in ArcGIS Pro. More on that trade-off shortly.

> **Tip:** A quick honesty check for any access analysis you inherit: if the map shows perfect circles around facilities, it was made with buffers, not network analysis.

That is not automatically wrong — buffers are fine for rough screening — but treat any claim about "residents within a 10-minute drive" with suspicion if the shapes are circular. Real drive-time areas are lumpy, stretched along fast roads and pinched at rivers.

Two Ways to Run Network Analysis

Before touring the individual tools, it helps to know that every one of them can run in two fundamentally different ways, and the choice shapes cost, setup effort, and data freshness.

	ESRI'S HOSTED ROUTING SERVICES	YOUR OWN NETWORK DATASET IN PRO
Where it runs	Esri's servers, via ArcGIS Online (or your own ArcGIS Enterprise routing services)	Locally, on your machine
Street data	Esri's global commercial network, kept current, with traffic in many regions	Whatever you build or license (StreetMap Premium, OpenStreetMap, agency centerlines)
Setup	None — the tools just work	Substantial — building a good network dataset is real work
Cost model	Consumes credits per solve (see the credits section below)	Extension license for Pro; no per-solve charge
Best for	Occasional analysis, web maps, anywhere-in-the-world coverage	High-volume solving, custom networks (trails, rail, indoor), offline work

In ArcGIS Online's Map Viewer, the network tools live in the analysis pane (open it from the map's toolbar, then look in the proximity group of tools). In ArcGIS Pro, they live on the `Analysis > Network Analysis` menu, which creates an analysis layer you configure and solve; Pro also offers ready-to-use tools that call Esri's hosted services directly when you would rather not manage a local network. The concepts below are identical in both places; only the buttons differ.

There is a third path worth knowing exists: everything here is also callable from Python and REST, which is how you automate a thousand solves overnight. That story belongs to Chapter 31 (Python for ArcGIS) and Chapter 32 (REST Services and Integration).

Travel Modes: Telling the Solver Who Is Traveling

Every network solve happens *as somebody*: a car, a delivery truck, a pedestrian. That somebody is a **travel mode** — a named bundle of settings that says which impedance to minimize (time or distance), what speeds to assume, which restrictions apply, and how the vehicle behaves at turns.

Esri's hosted network ships with a family of ready-made travel modes, and their names are self-explanatory: driving time, driving distance, trucking time (which respects truck restrictions and slower speeds), walking time and distance, and rural driving variants that assume you are willing to use unpaved roads. When you run any network tool, the travel mode is the first setting to check, because it silently controls everything else.

Two details deserve attention:

Time of travel. Where traffic data is available, driving-time modes can account for it. You can solve for "right now," for a typical Tuesday at 8:15 a.m., or for no particular time (free-flow speeds). A service area computed for rush hour can be dramatically smaller than the same service area at midnight. If your analysis claims to represent commute conditions, set the time of day deliberately rather than accepting the default.

Walking is not driving slowed down. The walking modes ignore one-way restrictions, use pedestrian paths, and assume a modest walking pace. If you analyze access to transit stops or schools, use a walking mode. A common and embarrassing error is publishing a "10-minute walk" map that was actually solved as a 10-minute drive with the label changed.

Barriers: editing reality for one analysis

Sometimes the network is right in general but wrong today: a bridge is closed for repairs, a street fair blocks downtown, a client wants to know what happens to

response times if a level crossing shuts. **Barriers** let you overlay temporary changes on the network without rebuilding anything. They come in three shapes:

- **Point barriers** block a single spot on a street (a crash site), or add a delay there (a checkpoint that costs five minutes).
- **Line barriers** block every street they cross — handy for sketching a parade route or a flooded corridor.
- **Polygon barriers** block or slow everything inside an area. A flood-extent polygon as a barrier turns a routine routing exercise into a genuinely useful emergency-planning tool. Scaled-cost polygon barriers are subtler: instead of forbidding travel, they multiply its cost, modeling "you can drive through this snowstorm area, but at half speed."

Barriers are per-analysis. They live in the analysis layer, not in the network, so your what-if experiments never contaminate anyone else's solves.

> **Watch out:** A barrier placed carelessly can do more than you intended. A point barrier snapped to a junction can block the whole intersection, not just one approach, and a polygon barrier covering a highway severs every route that used it, even routes that merely passed through your study area. After adding barriers, always run one sanity-check solve between two points you understand before trusting the batch results.

Routing: From A to B, and Through Twenty Stops

Routing is the tool everyone already knows from their phone, which is both a blessing (no one needs the concept explained) and a trap (people assume there is nothing more to learn). The GIS version earns its place in your toolkit in two ways: it handles many stops intelligently, and it produces routes as *data* — line features with attributes you can style, analyze, and publish, not just turn-by-turn directions that evaporate.

Point-to-point routes

The simple case: two or more stops, visited in the order given, minimizing time or distance per your travel mode. In Map Viewer, use the directions tool on the map toolbar for quick interactive routing, or the routing tools in the analysis pane when you

want the result saved as a layer. In Pro, `Analysis > Network Analysis > Route` creates a route layer; you load stops, set the travel mode, and click Run.

The result carries total time and distance, per-stop arrival attributes, and the route geometry itself. That geometry is worth pausing on: because it is an ordinary line feature, you can buffer it (which properties fall within 500 feet of the proposed detour?), intersect it with other layers (does the school bus route cross any rail lines?), or symbolize a hundred routes at once to reveal corridor patterns.

Optimized multi-stop routes

Give the solver twenty delivery stops and permission to reorder them, and you have the classic traveling-salesperson problem: what visiting order minimizes total travel? Look for the option to **reorder stops to find the optimal route** (in most interfaces it is a checkbox, with companions to preserve the first stop, the last, or both — because real routes usually start at a depot and end at home).

The improvement from optimization is rarely subtle. Stops entered in the order someone typed them into a spreadsheet routinely produce routes far longer than the optimized order, and the gap widens as the stop count grows. For anyone doing deliveries, inspections, meter reading, or home visits, this one checkbox is the most immediately profitable feature in this whole chapter.

Routes can also respect **time windows** — a promise that stop 7 will only be visited between 1 and 3 p.m. — which brings you to the edge of a bigger tool: the **vehicle routing problem** (VRP) solver, which assigns stops across a whole fleet of vehicles, balancing capacities, driver shifts, and time windows simultaneously. VRP is a deep specialty of its own; know that it exists and that when your question grows from "what order for my stops" to "which of my five trucks takes which stops," you have graduated from Route to VRP.

> **Tip:** Stops have to sit on the network to be routable, and the solver places them by snapping each input point to the nearest street. A stop geocoded — converted from an address to a map point — onto a rooftop in the middle of a large campus may snap to a service road on the wrong side of the property. If a route looks bizarre, inspect where the stops actually landed before blaming the solver — and see

Chapter 11 (Creating Data) for making geocoded points land where you mean them to.

Service Areas: The Honest Version of the Buffer

A **service area** (Map Viewer calls the tool something like Generate Travel Areas; older interfaces said drive-time areas) answers the question the fire-station circle pretended to answer: what territory is genuinely reachable from this point within a given cost? The result is a polygon — usually lumpy, stretched along highways, pinched at rivers — enclosing every street reachable within, say, ten minutes.

The essential settings:

Multiple breaks. You can request several thresholds at once — five, ten, and fifteen minutes — and get nested polygons. Choose whether they come back as overlapping disks (the 15-minute area includes the 10) or as rings, like a bullseye (the 10-to-15-minute band as its own donut-shaped polygon). Rings are usually what you want for mapping and for tabulating "population reached in each band"; overlapping disks are better when each threshold will be analyzed independently.

Direction of travel. Travel *away from* the facility models "what can this facility reach" — right for fire stations and delivery kitchens. Travel *toward* the facility models "who can reach this place" — right for clinics, stores, and polling sites. On a network with one-way streets or asymmetric rush-hour traffic, the two produce genuinely different polygons, so pick the one that matches your question rather than accepting the default.

Detail level. Service-area polygons are generalizations of a reachable set of streets, and most interfaces let you trade smoothness for fidelity. Standard-detail polygons can bridge small gaps and swallow unreachable pockets; high-detail polygons hug the streets more tightly and take longer to compute.

Once you have the polygons, the analysis usually continues with an overlay: intersect the drive-time bands with census data to estimate population served (Chapter 16 covers the overlay mechanics, and Chapter 17 covers doing the demographic math honestly — people are not spread evenly across census polygons, and apportionment matters).

Service areas are the workhorse of **equity analysis** — mapping who lacks access to grocery stores, pharmacies, parks, or early-voting sites — precisely because they capture the network's unfairness. Two neighborhoods equidistant from a hospital as the crow flies can be twenty minutes apart in access, and the service-area polygon shows it where the buffer hides it.

> **Watch out:** A service-area polygon includes territory, not people or addresses. Land inside the polygon that has no street access — the middle of a large park, a gated facility, an island of private roads missing from the network — is technically inside the shape but not actually reachable. For high-stakes claims ("every resident is within 8 minutes of an ambulance station"), validate by solving actual routes to a sample of address points rather than trusting polygon containment alone.

Closest Facility: Which One, and How Far

Service areas start from the facility and look outward. **Closest facility** starts from people — the tool calls them **incidents** — and asks, for each one: which facility is nearest by network travel, and what is the route? Feed it a layer of car crashes and a layer of hospitals, and it returns each crash paired with its nearest hospital, the travel time, and optionally the route line.

Settings that matter:

- **How many facilities to find.** Nearest one, or nearest three? Finding the top few is useful when the nearest facility may be full, closed, or out of the relevant specialty.
- **Cutoff.** A maximum cost, beyond which the solver stops looking. An incident with no facility within the cutoff comes back unassigned — which is often the most important finding in the whole analysis, because those unassigned incidents *are* your underserved population.
- **Direction of travel.** As with service areas: toward the facility for people seeking services, away from it for responders being dispatched.

Closest facility is the right tool whenever assignment is the question: which snow-plow depot covers this street, which school is really nearest to each student, which of our

three warehouses should ship this order. It is also the honest way to compute "distance to nearest X" as an attribute for statistical work — join the travel time back onto your incident layer, and the modeling in Chapter 17 can use real access instead of straight-line proxies.

Origin-Destination Cost Matrices: Everything to Everything

Sometimes you do not want one best route; you want the travel cost from *every* origin to *every* destination — a table with a row for each origin-destination pair. That is an **origin-destination cost matrix** (OD matrix for short), and it is the least glamorous and most quietly indispensable member of the family.

Why a table instead of routes? Because the OD matrix is usually an ingredient, not a final product. Market analysts feed OD matrices into gravity and spatial-interaction models (how much demand will each store capture?). Transportation planners use them to estimate commute burdens. And location-allocation — the next section — runs on OD costs internally. The solver exploits this: because nobody needs the turn-by-turn geometry for a million pairs, the OD tool computes costs without assembling full route shapes, which makes it far faster than solving each pair as a separate route. (If the output draws lines, they are straight connectors for visualization; the *costs* are true network costs.)

The trap is arithmetic. Costs scale with origins times destinations: 200 origins against 150 destinations is 30,000 pairs, and a full county's blocks against a region's facilities gets enormous fast. Use the cutoff setting (ignore pairs beyond a cost you do not care about) and the "closest N destinations" setting to keep the output — and the bill — proportionate to the question.

> **Watch out:** OD matrices are where credit consumption sneaks up on people, because the pair count grows multiplicatively while your mental model grows linearly. Before running a large matrix against the hosted service, do the multiplication, check what the operation costs in credits (your organization's administrator can see rates and remaining balance — see Chapter 34), and run a small pilot first. High-volume OD work is the single strongest argument for a local network dataset in Pro, where solves are free once you have the data and license.

Location-Allocation: Deciding Where Facilities Should Be

Everything so far analyzed a world where the facilities already exist. **Location-allocation** flips the problem: given a set of *candidate* sites and a map of demand, which sites should you pick? It is the difference between "how good is our coverage" and "where should the next library go" — analysis in the service of a decision.

The inputs are three layers and one goal:

- **Facilities**: candidate sites, any of which could be opened. You can mark some as *required* (already built, must stay) and, for market analysis, mark others as *competitors*.
- **Demand points**: where the need is — census-block centroids (the center points of the block polygons) weighted by population, customer locations, historical incident sites.
- **How many to choose**: pick the best 3 of these 20 candidates.
- **A problem type**: the objective the solver optimizes. This choice *is* the analysis, and it deserves its own table.

PROBLEM TYPE	THE QUESTION IT ANSWERS	TYPICAL USER
Minimize impedance	Which sites minimize the total travel burden across all demand?	Public services: libraries, clinics, polling places
Maximize coverage	Which sites cover the most demand within a response threshold?	Emergency services with a response-time standard
Minimize facilities	What is the fewest sites that still cover (nearly) everyone?	Budget hawks: same coverage, fewer buildings
Maximize attendance	Which sites attract the most visitors, assuming willingness to travel decays with distance?	Retail and services people choose to visit

PROBLEM TYPE	THE QUESTION IT ANSWERS	TYPICAL USER
Target market share / maximize market share	Given competitors on the map, which sites capture the share you want?	Businesses entering a contested market

The solver evaluates combinations of candidates against demand — using network costs, via an internal OD computation — and returns the winning sites plus allocation lines showing which demand each chosen facility serves. Because the number of possible combinations explodes combinatorially, the solver uses well-regarded heuristics rather than brute force: the answer is reliably excellent, though not certified mathematically perfect, which for siting decisions is exactly the right trade.

Two pieces of practical wisdom. First, location-allocation is decision *support*, not decision *making*: it knows travel costs and demand weights, and nothing about land prices, zoning, politics, or the parking lot. Treat its output as a shortlist that survives geographic scrutiny, then apply the human criteria. Second, the output is exquisitely sensitive to how you represent demand. Population-weighted block centroids answer a different question than customer points, which answer a different question than nighttime versus daytime population. Run the analysis under two or three demand assumptions; sites that win under all of them are robust picks, and sites that win under only one deserve suspicion.

Location-allocation lives primarily in ArcGIS Pro (`Analysis > Network Analysis > Location-Allocation`) with the Network Analyst extension, though hosted equivalents exist. If you work in Pro, this is also a natural candidate for ModelBuilder automation — rerunning the siting analysis under many scenarios is exactly the repetitive chain Chapter 25 (ModelBuilder, Complete) teaches you to wire up.

Tip: Before your first location-allocation, run a plain service-area analysis on the *existing* facilities. It grounds the conversation ("here is today's coverage and its gaps") and gives you a baseline to measure the proposed sites against. Decision-makers trust "the new site adds this many residents to 10-minute coverage" far more than an optimizer's abstract score.

Credits, Licensing, and Where Your Solves Actually Run

Because network analysis usually runs on Esri's servers, it is one of the corners of ArcGIS where analysis has a visible per-use cost. The details of credit accounting belong to Chapter 34 (Administration), but every analyst should carry this much:

Hosted network services consume credits per solve — whether you call them from Map Viewer, from Pro's ready-to-use tools, or from a script. Different operations bill differently — a simple route, a service area, an OD matrix pair, and a location-allocation run are each metered on their own basis — and the amounts per solve are individually small. The costs that surprise people are the multiplicative ones: batch routing for thousands of records, big OD matrices, repeated dashboard refreshes that quietly re-solve. Check your organization's credit budget before scripting a large batch, and note that some interfaces show an estimated credit cost before you commit to running — read that screen rather than clicking past it.

Local solving trades money for effort. ArcGIS Pro with the Network Analyst extension solves against a local network dataset without consuming credits. You pay instead in licensing (the extension, plus street data such as StreetMap Premium if you do not build from open data) and in the labor of maintaining the network. The crossover point is volume: occasional analysis favors the hosted service; industrial-scale solving favors local.

Enterprise sits in between. Organizations running ArcGIS Enterprise can publish their own routing services from their own network dataset, giving web maps and apps the convenience of a service without per-solve credits. If your organization has Enterprise, ask whether routing services are configured before assuming you must use Esri's — and see Chapter 35 for when standing up Enterprise makes sense at all.

Interactive directions in apps are metered too. Routing embedded in field and web apps (directions in Field Maps, for instance) draws on the same services. App builders should know that a route panel in a widely deployed app is a recurring cost, not a one-time one.

Choosing the Right Tool

The five tools blur together in memory, so here is the sorting logic in one place:

YOUR QUESTION	THE TOOL
What is the best path through these stops?	Route (or VRP, for a whole fleet)
What area can be reached from here within X minutes?	Service area
For each of these points, which facility is nearest, and how far?	Closest facility
What is the travel cost between everything and everything?	OD cost matrix
Which candidate sites should we pick to best serve demand?	Location-allocation

A closing habit worth forming: every network result is a claim about the world that you can spot-check for almost nothing. Pick one route and drive it in your head, or in a street-view tool. Pick one address near the edge of a service area and solve a single route to it. Pick the location-allocation's favorite site and ask a colleague who knows the neighborhood whether it passes the laugh test. The solvers are excellent; the inputs — your travel mode, your demand weights, your snapped stops, the network's currency — are where errors live, and one minute of checking catches most of them before they reach a decision-maker.

Network analysis rewards the effort it demands. Buffers tell you what is nearby; networks tell you what is *reachable*, and nearly every question that matters about people and places — access, equity, response, siting, delivery — is a question about reachability. Once you have seen a real drive-time polygon hugging the highways and stopping dead at the river, the tidy circle never looks honest again.

Terrain and Raster Analysis

A raster is a spreadsheet draped over the landscape. Every cell in the grid holds one number — an elevation, a temperature, a count of something — and because the whole surface is just numbers arranged in rows and columns, you can do math on it. That single idea powers everything in this chapter. Slope is math on elevation. A viewshed is math on elevation plus geometry. A suitability model is math on five or six rasters at once. Once you see raster analysis as arithmetic over a grid, tools that sound exotic become straightforward, and — just as important — you can spot when the arithmetic is being asked to say more than the data supports.

Chapter 1 (How GIS Thinks) introduced the vector/raster split; Chapter 15 (Imagery and Rasters in Practice) covered where rasters come from and how to manage them. This chapter is about what you *compute* from them.

Thinking in Cells

Before the tools, four ideas that every raster operation depends on.

Cell size (resolution). Each cell covers a square of ground — perhaps ten meters on a side, perhaps thirty. Nothing smaller than a cell exists in the data. A ten-meter elevation grid cannot see a drainage ditch two meters wide, and no amount of analysis will conjure it. Every result you compute inherits the resolution of your worst input.

The DEM. The workhorse dataset of this chapter is the *digital elevation model*, or DEM: a raster where every cell's value is the height of the ground at that spot, usually in meters or feet above sea level. Free, good-quality DEMs exist for most of the world; the Living Atlas (see Chapter 5, Finding Data) carries global terrain layers ready to use.

NoData. Cells can be empty — outside the study area, obscured by cloud, simply never measured. GIS calls this *NoData*, and it is contagious: in most operations, any calculation touching a NoData cell produces NoData. A hole in one input becomes a hole in every result downstream. Check your inputs for holes before you blame the tool.

Where the tools live. In ArcGIS Pro, most of what follows sits in the Spatial Analyst toolbox — an extension your license may or may not include — reached through the Geoprocessing pane (`Analysis > Tools`, then search by tool name). In ArcGIS Online's Map Viewer, a subset appears under the Analysis pane as raster analysis tools; running them there consumes credits, the pay-as-you-go currency covered in Chapter 34

(Administration), and raster analysis in Online generally requires an add-on imagery license on top of a standard account. The concepts are identical in both places; this chapter names the Pro tools because Pro exposes the full set.

> **Tip:** Before computing anything from elevation, check the Living Atlas. Esri publishes ready-made world layers for elevation, slope, aspect, and hillshade that stream instantly and cost you nothing to display. Compute your own only when you need the raw numbers for further analysis or a resolution the ready-made layers don't offer.

Reading the Land: Slope, Aspect, and Hillshade

These three tools all answer the same underlying question — *how does elevation change from one cell to its neighbors?* — and each phrases the answer differently. All three take a DEM in and produce a new raster out.

Slope

Slope measures steepness. For each cell, the tool looks at the eight surrounding cells, fits a tilted plane through their elevations, and records how tilted that plane is. The output raster answers "how steep is the ground here?" everywhere at once.

You choose the units: *degrees* (flat is 0°, a cliff approaches 90°) or *percent rise* (rise over run, times one hundred — a 100% slope is a 45° angle, which surprises everyone the first time). Engineering and accessibility standards usually speak percent; scientists usually speak degrees. Pick whichever your audience uses, and label the layer so nobody has to guess.

Slope is quietly one of the most used rasters in all of GIS. Construction suitability, trail difficulty, erosion risk, wildfire behavior, wheelchair access, farm machinery limits — all of them start with a slope raster.

Aspect

Aspect measures direction: which compass bearing does the ground face? The output value at each cell is a bearing in degrees — 0 to 360, where 0 and 360 both mean north,

90 east, 180 south, 270 west. Flat cells, which face no direction at all, get a special flag value.

Aspect matters wherever sunlight or exposure matters. In the northern hemisphere, south-facing slopes get more sun: snow melts earlier, vineyards ripen better, solar panels earn more. North-facing slopes stay cool and damp and grow different forests. One quirk to remember: aspect is *circular* data. The values 1 and 359 are nearly the same direction, but any tool that averages them naively will report 180 — due south, the exact opposite. Never take a plain average of aspect values; work with categories (the eight compass directions) instead.

Hillshade

Hillshade doesn't measure anything — it *draws*. The tool places an imaginary sun in the sky (you set its compass direction and its height above the horizon) and computes how brightly lit each cell would be, producing a grayscale image of light and shadow that makes terrain leap off the screen. The conventional sun position is from the upper left, partway up the sky; oddly, lighting from the south often makes terrain look inverted to human eyes, with valleys reading as ridges, so the upper-left convention persists even where that sun position is physically impossible.

Hillshade is a cartographic ingredient, not an analytical result. The classic recipe is a hillshade layer underneath your thematic layers, with the layers above set to partial transparency or a blend mode, so the data appears painted onto three-dimensional ground. Chapter 4 (Cartographic Design) covers the aesthetics; here, just know the raster comes from this tool, and that Living Atlas hillshades usually save you the trouble.

Watch out: Slope, aspect, and hillshade all compare elevation values against cell-to-cell distances, so the two must be in the same units. If your DEM stores elevation in meters but its coordinate system measures ground distance in decimal degrees (a geographic coordinate system — see Chapter 3), the math silently produces garbage: rise in meters divided by run in fractions of a degree yields near-vertical cliffs almost everywhere. The fix is either to project the DEM into a coordinate system that uses meters, or to supply the tool's *z-factor* — a conversion multiplier between elevation units and ground units — with the correct value. A

wrong z-factor fails in the other direction, flattening real terrain to nothing. If your slope map looks suspiciously vertical or suspiciously flat, check units first.

Seeing and Being Seen: Viewshed

A viewshed answers: standing *here*, what land can I see? The tool takes a DEM and one or more observer points, traces sight lines outward from each observer across the terrain, and marks every cell visible or not visible. The output is a raster of the visible territory — invaluable for siting fire lookouts, radio towers, and wind turbines, for planning scenic trails, and for assessing whether a proposed development will loom over a historic district.

The honest use of viewshed requires attention to three details:

Observer height. A viewer is not lying on the ground. Set the observer offset to eye level, or to the height of the proposed tower. A few meters of offset dramatically changes what's visible across flat country.

What the surface includes. A standard DEM represents *bare earth* — trees and buildings shaved off. A person standing in a forest cannot actually see through the trunks, but a bare-earth viewshed says they can. If vegetation and structures matter to your question, you need a *digital surface model* (DSM) — a raster of the tops of things rather than the ground beneath them — usually derived from lidar, airborne laser scanning (see Chapter 15). Choosing between DEM and DSM is choosing which fiction to accept: bare earth overstates visibility, while a surface model treats every tree canopy as a solid wall, which understates it.

Distance and curvature. Over long distances the Earth's curvature hides low terrain, and the atmosphere bends light slightly. The viewshed tools have options to account for both; leave them off for a valley-scale study, turn them on when sight lines run tens of kilometers.

Watch out: A viewshed is only as trustworthy as its surface. Run one on a coarse bare-earth DEM and present it as "what the neighbors will see of the new warehouse," and you may badly mislead people in both directions — a hedge, a barn, or ten meters of DEM error can flip a cell from visible to hidden. Report

viewsheds as estimates, state what the surface did and didn't include, and spot-check a few sight lines in the field or in street-level imagery when the stakes are real.

For a single line-of-sight question — can point A see point B? — there are simpler sight-line tools that answer with a yes/no and show where the line is blocked. Reach for the full viewshed only when you need the whole territory.

Where Water Goes: Watersheds and Flow

Hydrology tools simulate a simple rule — water flows downhill — cell by cell across a DEM. From that one rule falls a whole family of derived layers, and they build on each other in a fixed order. This is the most sequence-dependent workflow in the chapter, so it's worth walking the standard chain.

Step 1: Fill. Real DEMs contain small artificial pits — single cells or small clusters lower than everything around them, usually data errors rather than real ponds. Simulated water pours into these pits and never leaves, truncating every flow path downstream. The Fill tool raises pit cells until water can escape. Nearly every hydrology workflow starts here.

Step 2: Flow Direction. For each cell, which of its eight neighbors is most steeply downhill? The output raster encodes each cell's single drainage direction. This layer is not much to look at, but everything else derives from it.

Step 3: Flow Accumulation. Now the model releases a virtual drop of rain on every cell and lets it run downhill along the flow directions. Each cell's output value is a count of how many upstream cells drain through it. Ridge cells accumulate nothing; cells in valley bottoms accumulate thousands or millions. Symbolize this layer so only high values draw, and a startlingly realistic stream network appears — extracted from nothing but elevation. Choosing the threshold ("a stream begins where at least this many cells drain through") is a judgment call you should calibrate against a real stream map of the area.

Step 4: Watershed. Choose a point of interest — a water intake, a monitoring station, a culvert — and the Watershed tool traces the flow directions backward to find every cell

that eventually drains through that point. The result is the drainage basin, or *watershed*, of your point. One practical wrinkle: your chosen point must sit exactly on a high-accumulation cell, or the tool will delineate the drainage of a hillside cell three meters from the actual creek and return a comically tiny basin. The Snap Pour Point tool exists precisely to nudge your point onto the stream before you delineate.

> **Tip:** Sanity-check every watershed against a topographic map or hillshade. A correct basin boundary follows ridgelines. If yours cuts across a valley or stops at a straight line, something upstream in the chain went wrong — usually a skipped Fill, a DEM edge, or an unsnapped pour point.

Two limits worth stating plainly. First, this is *surface* hydrology: the model knows nothing about storm drains, culverts, groundwater, or soil absorption, so in cities and in karst country — limestone terrain where streams vanish into sinkholes and underground channels — its answers are approximations at best. Second, the DEM's resolution sets the smallest channel the model can perceive. For rough planning — "roughly what area drains to this pond?" — the standard chain on a public DEM is excellent. For engineering design, hydrologists use specially corrected ("hydro-conditioned") elevation data and dedicated software, and you should hand the problem to them.

Filling the Gaps: Surface Interpolation

Interpolation runs the opposite direction from everything so far. Instead of deriving new rasters from an existing surface, it *creates* a surface from scattered point measurements: rainfall at forty gauges, well depths at ninety boreholes, soil lead at sixty samples. The tool estimates a value for every cell between the points, producing a continuous raster from discrete data.

That word *estimates* is doing heavy lifting, so start with the honesty question.

When interpolation is honest

Interpolation is legitimate when the phenomenon is genuinely continuous — when it truly has a value everywhere and changes gradually between your sample points. Temperature, rainfall, elevation, groundwater depth, and soil chemistry mostly qualify.

It is *not* legitimate for data that only exists at points or within boundaries. Interpolating house sale prices produces a surface implying the middle of a lake has a property value. Interpolating median income from the center points of census polygons manufactures smooth gradients out of what are really administrative cliffs. If a value between your points is a fiction, the surface is a fiction, however smooth it looks.

Three more honesty checks: you need *enough* points (a handful of samples cannot support a detailed surface, whatever the software renders); the points should *cover* the area (interpolation between samples is estimation, but extending beyond the outermost samples is extrapolation — guessing — and most tools will happily do it without warning you); and the samples should be *reasonably distributed* (forty gauges clustered in one valley tell you little about the next ridge over).

IDW: the intuitive method

Inverse Distance Weighting (IDW) estimates each cell as a weighted average of the nearby sample points, where closer points count for more — hence "inverse distance." It's fast, easy to understand, and has no statistical pretensions. Its signature weaknesses: the surface can form bull's-eye rings around individual sample points, and it can never estimate a value higher than your highest sample or lower than your lowest — real peaks and pits between samples are flattened.

Kriging: the statistical method

Kriging (named for a South African mining engineer) starts by studying your data before estimating anything: it measures how quickly values become dissimilar as points get farther apart, fits a model to that pattern, and then uses the model to weight its estimates. Two consequences make it worth the added complexity. First, the weights are tuned to how your particular phenomenon actually varies in space rather than to a generic distance rule. Second — and this is the genuinely valuable part — kriging can produce a *companion error surface* showing where its estimates are confident (near dense samples) and where they are shaky (in the gaps). An estimate that carries its own uncertainty map is a far more honest product than a confident-looking surface with no error bars.

The cost is that kriging asks you to make statistical modeling choices, and bad choices produce bad surfaces with the same authoritative smoothness as good ones. If the interpolated surface is a supporting exhibit, IDW is usually fine. If it *is* the deliverable

— a contamination plume, a groundwater map that drives decisions — invest in kriging or recruit someone comfortable with geostatistics.

METHOD	HOW IT ESTIMATES	STRENGTHS	WEAKNESSES
IDW	Weighted average of nearby samples; closer counts more	Simple, fast, predictable, easy to explain	Bull's-eyes around samples; can't exceed sampled min/max
Kriging	Weighted average tuned by the data's own spatial pattern	Adapts to the phenomenon; produces an uncertainty surface	Requires statistical judgment; easy to misconfigure
Spline	Fits a flexible smooth sheet through the points	Very smooth results; good for gently varying surfaces	Can overshoot wildly between distant or noisy samples

Whatever the method, hold back a few sample points before interpolating, then compare the finished surface's estimates at those locations against the measured truth. Five minutes of validation beats any amount of visual smoothness.

Map Algebra and the Raster Calculator

Here the "grid as spreadsheet" idea pays off directly. *Map algebra* is the practice of writing arithmetic where the variables are entire rasters. The expression runs once per cell, across millions of cells, and the result is a new raster. The Raster Calculator tool (search for it in the Geoprocessing pane) is where you type such expressions.

Some expressions are plain arithmetic. Subtract a lidar bare-earth DEM from a surface model and you get a raster of feature heights — every tree and building's height above ground, in one line. Subtract last decade's elevation from this year's and you map erosion and deposition. Multiply a rainfall raster by a runoff-coefficient raster and you approximate runoff.

Others are logical tests. The expression `"slope" < 5` asks every cell "is your slope under five?" and returns a raster of 1s (true) and 0s (false) — a mask of gentle ground. Conditional functions extend this to "where slope is under five, keep the land-cover

value; elsewhere, NoData," which is how you clip and combine rasters by rules rather than by hand.

The Raster Calculator's simpler sibling deserves equal billing: **Reclassify**. It swaps old cell values for new ones according to a table you define — turn continuous slope into three classes (gentle = 3, moderate = 2, steep = 1), collapse forty land-cover categories into five, convert "distance in meters" into "score from 1 to 5." Reclassify is the standard way to translate raw measurements into common scoring scales, which makes it the backbone of the suitability workflow at the end of this chapter.

> **Watch out:** When two rasters enter one expression, their grids must line up. If cell sizes differ, or the grids are offset by half a cell, the software resamples on the fly — re-estimating values onto a new grid — and cells get compared against slightly-the-wrong neighbors. Before a multi-raster workflow, open the geoprocessing *environment settings* and set three things: a common cell size, a common processing extent, and a *snap raster* (an existing raster whose grid alignment all outputs must copy). Also confirm every input shares one projected coordinate system — Chapter 3 explains why analysis in an unprojected system distorts areas and distances. Five minutes of environment setup prevents subtle misalignment that no error message will ever report.

Zonal Statistics: Raster Answers Summarized by Polygons

Raster analysis produces per-cell answers, but decisions are usually made about *places* — parcels, districts, catchments, counties. Zonal statistics is the bridge. You supply two things: a raster of values and a set of *zones* (usually polygons, though a categorical raster works too). The tool summarizes the raster inside each zone — average, minimum, maximum, range, majority, sum — and hands you the results.

Concrete examples make the pattern click:

- Zones = property parcels, values = slope raster → average and maximum slope per parcel, ready to flag lots too steep to build on.
- Zones = watersheds from the hydrology section, values = a rainfall surface from the interpolation section → total rain falling in each basin.

- Zones = school attendance areas, values = a categorical land-cover raster → the dominant land cover in each area.
- Zones = census tracts, values = summer land-surface temperature from satellite imagery → which neighborhoods run hottest, the standard first step in urban heat-island studies.

The most common form, Zonal Statistics as Table, outputs a plain table with one row per zone, which you then *join* back to the polygon layer by its ID field so the numbers become mappable attributes. (Joins and relationships are covered in Chapter 12, Schema Design.) Two practical cautions: if your zones overlap, check how your tool handles it — when zones are converted to a raster each cell can belong to only one zone and overlaps are silently dropped, though current versions of the table tool can process overlapping polygon zones correctly; and zones smaller than a few cells produce statistics from a sample size of almost nothing, so treat tiny-polygon results with suspicion.

If your summary question is about *vector* data inside polygons — how many crime points per district, total road length per county — that's the aggregation family in Chapter 16 (Proximity and Overlay), not zonal statistics. Zonal statistics is specifically the raster-to-polygon bridge.

Distance Surfaces

Chapter 16 measures distance the vector way: buffers and near-tables between discrete features. The raster way is to compute a *distance surface* — a raster where every cell's value is its distance to the nearest of something. Feed the tool your source features (roads, hospitals, streams) and every cell in the study area learns how far away it is.

Why bother, when buffers exist? Because a distance surface is a *continuous* answer. A buffer says in-or-out at one arbitrary cutoff; a distance surface preserves the full gradient, which you can then reclassify into any scoring scheme you like — near roads scores 5, moderately near scores 3, remote scores 1. When distance is one criterion among several (as in the suitability model coming next), the surface form is what you want.

The straight-line version is only the beginning. A *cost distance* surface replaces "how far" with "how hard to get there." You build a *cost raster* first — each cell's value is the

difficulty of crossing that cell, derived from slope, land cover, barriers, whatever impedes movement in your problem — and the tool accumulates the least possible total cost from the source to every cell, flowing around expensive terrain the way a hiker contours around a cliff. From the accumulated-cost surface, a companion tool can trace the single cheapest route between two points: the *least-cost path*, the classic method for sketching corridors for trails, pipelines, transmission lines, and wildlife movement. In recent versions of Pro, the older separate distance tools have been consolidated; searching the Geoprocessing pane for "distance accumulation" will find the current entry point.

DISTANCE FLAVOR	QUESTION IT ANSWERS	WHEN TO USE IT
Straight-line (Euclidean)	How far, as the crow flies?	Proximity scoring in open terrain; quick screening
Cost distance	How hard to reach, given terrain and barriers?	Corridors, wildlife movement, off-road access
Network distance (Ch. 18)	How far along roads or utilities?	Drive times, service areas, anything vehicle-bound

The choice matters more than beginners expect: a hospital two kilometers away in a straight line may be a twenty-minute drive around a river. If travel happens on a network, use network tools (Chapter 18). If travel happens across terrain, cost distance. Straight-line only when the crow's-eye view is genuinely the right model.

Suitability Modeling: The Weighted Overlay

Everything in this chapter converges here. A *suitability model* answers a "where is best?" question — where should the new park, fire station, solar farm, or vineyard go? — by combining several criteria rasters into one ranked map. The method is old, the arithmetic is simple, and the value lies almost entirely in the judgment you bring to it.

The recipe has five steps:

1. **Choose criteria.** What actually makes a site good or bad for this purpose? Write the list *before* looking at what data you have, then reconcile.

2. **Build a raster for each criterion.** This is where the earlier sections report for duty: a slope raster, a distance-to-roads surface, a land-cover layer, an interpolated soil surface.
3. **Reclassify each to a common scale.** Raw units aren't comparable — degrees, meters, and category codes can't be averaged. Use Reclassify to translate each raster to the same suitability score, say 1 (poor) to 5 (excellent). Cells that are flatly unacceptable are marked *restricted* and will be excluded entirely, not merely scored low.
4. **Weight and combine.** Assign each criterion a percentage of influence, summing to 100, and compute the weighted average cell by cell. The Weighted Overlay tool packages steps 3 and 4 together (note that it accepts only whole-number rasters, which reclassified scores naturally are); the Raster Calculator does the same job transparently as `("slope_score" * 0.3) + ("roads_score" * 0.2) + ...`.
5. **Verify and stress-test.** More on this below — it is not optional.

A worked example: siting a community orchard

Suppose a small town wants a site for a community orchard on public land. Talking with the parks staff and a local grower yields five criteria:

CRITERION	SOURCE RASTER	RECLASSIFIED SCORING (1–5)	WEIGHT
Slope	Slope from the town DEM	Gentle scores high; steep scores low; extremely steep = restricted	30%
Aspect	Aspect from the DEM	South-ish faces score high; north-ish low; flat mid	15%
Soil drainage	County soil survey, converted to raster	Well-drained high; poorly drained low; wetland = restricted	25%
Access	Straight-line distance to roads, reclassified	Close scores high, remote low	20%

CRITERION	SOURCE RASTER	RECLASSIFIED SCORING (1–5)	WEIGHT
Current land cover	Land-cover raster	Open/grass high; forest low (clearing cost); built-up = restricted	10%

Each criterion goes through Reclassify to the 1-to-5 scale with the restrictions flagged, the five scored rasters go into Weighted Overlay with the weights above, and out comes a single raster scored 1 to 5. Symbolize it from red to green, overlay the public-land parcels, and run Zonal Statistics to get each parcel's average suitability — and the shortlist writes itself. Notice how many chapters cooperated: slope and aspect from the terrain section, a distance surface, reclassification from map algebra, zonal statistics to reach a parcel-level decision, and vector overlay from Chapter 16 to constrain it all to public land.

The honest parts

The map that emerges looks precise and objective. It is neither, and the modeler's job is to say so.

The weights are opinions. Nothing in the data says slope deserves 30% and land cover 10% — people decided that, and reasonable people would decide differently. The professional response is *sensitivity analysis*: rerun the model with a few alternative weightings and see which candidate sites stay green under all of them. Sites that survive every reasonable weighting are robust recommendations; sites that appear only under one weighting are artifacts of that opinion. Reporting "these three parcels rank highly under every weighting we tried" is a far stronger claim than any single map.

The scoring breakpoints are opinions too — why does "gentle" end at one slope value and not another? — and should come from domain knowledge (the grower knows what slopes an orchard tolerates) rather than from what makes the map look balanced.

And the model only knows what's in the rasters. Land ownership disputes, a neighbor's objection, a planned road widening — none of it appears in the grid. A suitability model's proper role is to shrink a thousand candidate sites to five worth visiting. The last step is always a person standing on the ground.

Tip: Build any suitability model in ModelBuilder (Chapter 25) rather than by running tools one at a time. The model diagram documents your criteria and weights, and — because sensitivity analysis means rerunning everything repeatedly with small changes — a rebuilt-by-hand workflow gets tedious by the second iteration, while a ModelBuilder chain reruns in one click.

Choosing Your Weapon

A closing orientation, since this chapter has introduced a dozen tools. Slope, aspect, and hillshade are near-instant derivations you'll compute (or stream from the Living Atlas) constantly. Viewshed and the hydrology chain are occasional, purpose-driven, and demand care about what the elevation surface really represents. Interpolation is powerful and easily abused — reserve it for genuinely continuous phenomena and validate against held-back samples. Map algebra, reclassify, and zonal statistics are the connective tissue you'll use in nearly every raster project. Distance and cost surfaces feed corridor and access questions. And suitability modeling is where all of it assembles into a decision — provided you treat the weights as the opinions they are and let sensitivity analysis, not a single confident map, carry the recommendation.

The through-line is the one this chapter opened with: it's all arithmetic on a grid. Master the arithmetic, respect the resolution, and always ask whether the numbers in the cells can honestly support the question you're asking of them.

Space-Time Analysis

A map is a photograph. It shows you where things were at one moment, and it is silent about everything else: whether the cluster of break-ins on the east side is new or has been there for a decade, whether the algae bloom is growing or shrinking, whether the traffic counts you mapped were taken during a festival week. Most of the questions people actually bring to a GIS are not "where" questions at all — they are "where and when" questions. Is it getting worse? Did our intervention work? Where is the problem *emerging*, as opposed to where has it always been?

This chapter is about adding the time dimension to spatial analysis. You have already met time-enabled layers as a display feature — the time slider that animates a map, covered in Chapter 6 (Map Viewer: The Complete Reference). Here we treat time as an analytical variable: something you slice, compare, mine for trends, and — most importantly — something whose measurement flaws can quietly wreck your conclusions. The statistical machinery builds directly on Chapter 17 (Statistical and Pattern Analysis), so if terms like "hot spot" and "statistical significance" are hazy, skim that chapter first.

Time-Enabled Layers, Revisited Analytically

A **time-enabled layer** is simply a layer where ArcGIS knows which field (or fields) holds the timestamp for each feature, so the software can filter, animate, and analyze by time. Enabling it is usually a one-time configuration: for a hosted feature layer, the switch lives in the layer's item page settings, where you point ArcGIS at the relevant date field; in ArcGIS Pro, it lives in the layer's properties under a time tab. The exact labels shift between releases, so think of the goal — "tell the software which field means *when*" — rather than memorizing a menu path.

The two shapes of temporal data

Every temporal dataset answers one of two questions about each feature, and it matters which:

SHAPE	WHAT EACH FEATURE REPRESENTS	EXAMPLE	TYPICAL FIELDS
Instant	Something happened at a moment	A crime report, a water-quality sample, a GPS ping	One date/time field
Interval	Something was true for a duration	A road closure, a lease, a construction permit	A start field and an end field

Instants are the common case and the easy one. Intervals need two fields, and analysis on them has a subtlety: an interval can *span* your analysis periods. A road closure that started in March and ended in June belongs to March, April, May, and June — not just to the period containing its start date. When you filter interval data by time, decide

explicitly whether you mean "started during," "ended during," or "was active during" the window. They give different answers.

Dates that are not really dates

The single most common obstacle to time analysis is a date stored as text. A field containing the characters "03/07/2024" *looks* like a date but, if its field type is string, ArcGIS cannot sort it chronologically, bin it, or put it on a time slider — it is just letters and slashes. Worse, "03/07/2024" is ambiguous: March 7th in the United States, July 3rd in most of the world.

> **Tip:** Before doing anything else with a temporal dataset, check the field type of your date column. If it is a string, convert it to a true date field first — in Pro, the field calculator can parse text into dates; for hosted layers, you can add a new date field and calculate it from the text field. Chapter 12 (Schema Design) covers field types in depth. Ten minutes of conversion now saves hours of confusion later.

Two more things to verify while you are in there. First, **time zones**: date fields in hosted feature layers are stored internally in coordinated universal time (UTC — the global reference clock), and the layer's settings declare what time zone the values should be interpreted in. If that declaration is wrong, every "incidents by hour of day" chart you make will be shifted by several hours, and your "late-night" hot spot may actually be an evening one. Second, **editor tracking dates**: many layers carry automatic fields recording when each row was created or last edited. These record *database* events, not real-world ones. If your field crew entered Monday's inspections on Friday, the created-date field says Friday. Analyze the field that records when the thing *happened*, and be suspicious of any dataset where the "event date" and the "created date" are identical for every row — it usually means nobody recorded the real event time.

From viewing to analyzing

The time slider is for your eyes: drag it, watch dots appear and vanish, build intuition. That is genuinely valuable — always animate your data before analyzing it, because you will spot gaps, bursts, and oddities no summary statistic will show you. But eyes are also easily fooled. An animation of random noise looks like it has patterns; a real trend

can hide inside visual clutter. Analysis means converting the animation into numbers you can defend: counts per period, rates per area per period, statistically tested trends.

Slicing Time: Bins and Why the Width Matters

Almost every space-time analysis begins by chopping continuous time into **bins** — equal-width periods like days, weeks, months, or years — and counting or summarizing what falls into each. This is the temporal cousin of choosing polygon sizes for aggregation, and it carries the same hazard you met in Chapter 17: your choice changes the result.

Bin too finely (say, hourly bins for a dataset with a few events per week) and every bin holds zero or one event; patterns drown in noise. Bin too coarsely (annual bins for something that shifts monthly) and the pattern you care about gets averaged away. There is no universally correct width. Useful rules of thumb:

- Match the bin to the process. Restaurant inspections cycle annually; traffic incidents cycle weekly and daily; construction activity cycles seasonally.
- Aim for bins where the typical count is comfortably above a handful, so that random fluctuation does not dominate.
- If a bin width choice feels arbitrary, run the analysis at two widths. Conclusions that survive both are sturdier; conclusions that appear only at one width deserve suspicion.

Also mind the **partial bin** at each end of your data. If your data runs from January 2021 to mid-March 2024, a yearly binning makes 2024 look like activity collapsed — because 2024 is only ten weeks long. Trim incomplete periods before comparing.

Change Detection Between Periods

The simplest space-time question is a two-photograph comparison: here is period A, here is period B, what changed and where? It sounds trivial. Done carelessly, it produces some of the most confidently wrong maps in the business.

Change detection with features

For point or polygon data, the standard recipe is:

1. **Define two windows.** Say, calendar year 2023 versus calendar year 2024. Equal lengths, and — if the phenomenon is seasonal — covering the same seasons.
2. **Aggregate each window to common areas.** Count the points falling in each neighborhood, grid cell, or district, once for each window. Chapter 16 (Proximity and Overlay) covers the aggregation tools; the summarize-within family is the usual workhorse.
3. **Join and difference.** Attach both counts to the same set of areas and compute the change — either as a new field via the field calculator, or with an Arcade expression at display time (Chapter 8 covers Arcade).
4. **Map the difference**, ideally with a diverging color scheme centered on zero, so increases and decreases read as opposites (see Chapter 7, Styling and Smart Mapping).

The trap hides in step 3: **how** you express change.

MEASURE	FORMULA IN WORDS	STRENGTH	WEAKNESS
Absolute change	B minus A	Honest about magnitude	Big areas dominate just by being big
Percent change	(B − A) ÷ A	Comparable across areas	Explodes when A is small; undefined when A is zero
Rate change	Change per population or per area	Fairest comparison	Needs a reliable denominator

Watch out: Percent change on small numbers is a headline generator, not an analysis. A district that went from one incident to three shows a 200 percent increase; a district that went from 400 to 480 shows 20 percent. The first is statistical noise; the second is a real shift affecting eighty more events. Always map absolute change alongside percent change, and consider suppressing or flagging areas whose baseline count is below some small threshold.

Change detection with rasters

For continuous surfaces — vegetation health from satellite imagery, land cover, elevation — change detection is usually raster math: subtract the earlier raster from the later one, and the result is a change surface. For categorical rasters (land cover classes), the comparison is a **change matrix**: a table showing how many cells moved from each class to each other class, from which "forest became developed" jumps out. The mechanics of raster processing live in Chapter 15 (Imagery and Rasters in Practice) and Chapter 19 (Terrain and Raster Analysis); the design principles here — equal-quality inputs, comparable seasons, honest denominators — apply identically. One raster-specific caution: two images taken with different sensors, sun angles, or atmospheric conditions will show "change" that is really just measurement difference. Change detection on imagery is as much about controlling the inputs as about the subtraction.

Space-Time Pattern Mining: Emerging Hot Spots in Plain Terms

Two-period comparison throws away everything between the endpoints. If you have many periods — monthly data over five years, say — you can ask a much richer question: not just "where is it hot?" but "where is it *becoming* hot, where has it *always* been hot, and where is it cooling off?" This is the territory of **space-time pattern mining**, and its centerpiece in ArcGIS is emerging hot spot analysis.

The space-time cube

The foundation is a data structure with an unusually literal name: the **space-time cube**. Picture your study area divided into a grid of cells, like a checkerboard laid over the map. Now imagine stacking a copy of that checkerboard for every time bin — one layer for January, one for February, and so on — into a three-dimensional block. Each little box in the block (one grid cell at one time period) holds a count: how many incidents happened in that place during that period. That block is the cube.

In ArcGIS Pro, you build one with a geoprocessing tool whose name says exactly what it does — Create Space Time Cube By Aggregating Points — found by searching the geoprocessing toolboxes via `Analysis > Tools`. You feed it your time-enabled point layer, choose a cell size and a time-bin width, and it writes the cube to a file (in a scientific data format called netCDF, which you can treat as a black box). Everything

above about bin width applies double here, because you are choosing two bin sizes at once — one in space, one in time — and both shape the result.

> **Tip:** Let the data suggest the spatial cell size rather than guessing. A reasonable starting point is a cell small enough to separate the distinct places you care about, but large enough that a typical cell-period combination contains more than an occasional stray point. If most boxes in the cube are empty, your cells or bins are too small; the analysis will run, but the results will be brittle. Build the cube twice at different resolutions and compare — agreement is evidence.

What emerging hot spot analysis actually does

Emerging Hot Spot Analysis (a companion tool in the same toolbox) does two things to the cube, and both are ideas you have already met:

1. **For each place at each time**, it runs the same kind of hot spot test you saw in Chapter 17 — asking whether that cell's count is significantly higher (hot) or lower (cold) than you would expect by chance, considering its neighbors in *both* space and time. A cell's "neighborhood" now includes the cells around it on the map *and* the same location in adjacent time periods.
2. **For each place across all times**, it runs a trend test — a standard statistical check (the Mann-Kendall test, if you want the name) for whether that location's hot-spot intensity — how strongly hot or cold it scores — is consistently rising or falling over the whole series.

Then it combines the two into a plain-English category per location. The categories are worth knowing because they are the entire output vocabulary of the tool:

CATEGORY	MEANING IN PLAIN WORDS
New hot spot	Hot only in the most recent period; never hot before
Consecutive hot spot	Hot in an unbroken run of recent periods, but not for most of its history

CATEGORY	MEANING IN PLAIN WORDS
Intensifying hot spot	Hot most of the time, and getting significantly hotter
Persistent hot spot	Hot most of the time, with no significant trend up or down
Diminishing hot spot	Hot most of the time, but cooling significantly
Sporadic hot spot	Hot now, with a history of flickering in and out of being hot; never cold
Oscillating hot spot	Hot recently, but has been *cold* in earlier periods
Historical hot spot	Was hot for much of its history, but not recently

Each category has a cold-spot mirror image (new cold spot, persistent cold spot, and so on), and locations with no significant pattern are left uncategorized. For decision-making, the categories sort themselves into narratives: **new**, **consecutive**, and **intensifying** hot spots are where a problem is arriving or accelerating — the places to investigate first. **Persistent** hot spots are the chronic condition everyone already knows about. **Diminishing** and **historical** hot spots are, tentatively, good news — or a sign your data collection stopped, which we will get to.

Reading the output honestly

Resist the temptation to treat the category map as ground truth. Three cautions. First, every cell's label rests on statistical tests, and with thousands of cells some will cross the significance line by luck; a lone "new hot spot" cell surrounded by nothing deserves less trust than a coherent patch of them. Second, the labels are exquisitely sensitive to the time-bin width and the length of the study period — "recent" literally means the final bins, so a different bin width redefines what counts as new. Third, the tool models counts, not causes. It can tell you *that* the northeast corridor is intensifying; it cannot tell you whether that is more incidents or more reporting.

The same cube supports two sibling analyses worth knowing by name. **Local outlier analysis** finds space-time oddballs — a quiet cell surrounded by hot ones, or a spike in

a normally calm location — which is often the fastest route to "something specific happened here; go look." **Time series clustering** groups locations by the *shape* of their history — places that share a rhythm, whatever their volume — which is a good exploratory step when you suspect there are a few distinct temporal behaviors in your study area but do not know what they are.

Tracking and Movement Data Basics

Everything so far treats events as independent dots. Movement data is different: the dots are *connected*. A delivery van, a tagged animal, a survey crew — each produces a stream of GPS positions that only make sense as a sequence.

The anatomy of tracking data

Tracking data is almost always a point layer with three essential ingredients per row: a location, a timestamp, and a **track identifier** — the field that says which vehicle, animal, or person this ping belongs to. Without the track ID, ten vans' worth of breadcrumbs is just an undifferentiated cloud. If you collect data with mobile apps, Chapter 30 (Field Operations) covers how location tracks get captured in the first place; here we take the breadcrumbs as given.

From points to tracks

The first analytical move is usually to reconnect the dots: group points by track ID, sort each group by timestamp, and connect consecutive points into lines. ArcGIS has tools for exactly this — a simple points-to-line tool in Pro, and richer track-reconstruction tools in some analysis toolsets whose availability depends on your setup — and once you have line segments between consecutive pings, useful quantities fall out of arithmetic:

- **Speed:** segment length divided by the time between its two pings.
- **Dwell:** a run of consecutive pings that stay within a small distance of each other for more than some minimum duration — in other words, a stop. Where do the vans actually stop, and for how long?
- **Corridors:** overlay many tracks and the shared routes emerge as dense braids; aggregate track segments to a grid and you get a usage-intensity map.

- **Origin-destination patterns:** take just the first and last point of each track (or of each detected trip) and you have a from-here-to-there dataset, ready for the flow-analysis ideas in Chapter 18 (Network Analysis).

GPS reality

Raw GPS is messier than the demos suggest. Positions wobble by meters even standing still; in dense downtowns, signals bounce off buildings ("urban canyon" error) and a pedestrian's track can teleport across the street. Devices also record at different rates — one ping per second versus one per two minutes — and a computed "speed" between two pings four minutes apart is an average that can hide a stop entirely. Practical hygiene: filter out physically impossible speeds (a delivery van doing several hundred kilometers per hour is a GPS glitch, not a felony), treat dwell detection thresholds as tunable rather than sacred, and never compare track statistics between devices with very different ping rates as if they were equivalent.

> **Watch out:** Tracking data about people is sensitive in a way most GIS data is not. A week of location breadcrumbs reveals where someone lives, worships, and receives medical care. Before analyzing or — especially — sharing tracks, know your organization's policy, aggregate where possible, and default to the coarsest data that answers the question. Chapter 34 (Administration) covers the sharing and security controls.

Seasonal Patterns, in Plain Terms

Most human and natural phenomena breathe in cycles: more burglaries in summer evenings, more potholes after the spring thaw, more retail foot traffic in December. Any time series measured at a place can be imagined as three ingredients added together:

- a **trend** — the slow underlying drift up or down across years;
- a **seasonal component** — the repeating wiggle that looks roughly the same every cycle;
- and **noise** — the leftover random jitter.

Statisticians call separating these ingredients **decomposition**, and dedicated tools exist for it, but you can get most of the practical value with two disciplined habits that

require no special software.

First, **compare like season with like season**. The honest comparison for this July is *last* July, not this June. Year-over-year comparison silently subtracts the seasonal wiggle, because both sides of the comparison contain the same wiggle. Nearly every "sudden spike" that turns out to be embarrassing is a month-over-month comparison that rediscovered summer.

Second, **plot before you map**. Build a simple chart of counts per time bin — Map Viewer's charting, Pro's charts, or a Dashboards serial chart (Chapter 28) all work — and look at several years side by side. If every year shows the same hump in the same months, that hump is seasonality, not news. What *is* news is a deviation from the usual seasonal shape: a winter that behaves like a summer, a hump arriving two months early. Seasonal decomposition, at its heart, is just formalizing that eyeball comparison — and for many operational decisions, the eyeball version plus year-over-year discipline is enough.

Where this becomes spatial: seasonality often differs by *place*. Coastal districts may peak in tourist season while inland ones peak during the school year. Aggregating your data by both area and month — a simple pivot — reveals whether one seasonal story fits the whole map or each region dances to its own calendar. Time series clustering on a space-time cube, mentioned above, automates exactly this discovery.

Designing Before/After Studies

Sooner or later someone will ask the highest-stakes space-time question there is: *did it work?* Did the new streetlights reduce crime, did the culvert repair stop the flooding, did the road diet — a deliberate reduction in traffic lanes — calm traffic? A before/after study looks simple — count before, count after, subtract — but the design choices you make before touching any tool determine whether the answer means anything.

Pin down the moment of change

Decide exactly when "before" ends and "after" begins, and be honest about fuzziness. Interventions rarely switch on instantly: the streetlights were installed over six weeks; the policy was announced in March but enforced from May. A common, defensible move is to exclude a buffer window around the transition from both periods, so neither side is contaminated by the messy middle.

Choose comparable windows

The before and after windows should be equal in length and cover the same seasons — one full year on each side is the classic choice when data allows, because it neutralizes seasonality entirely. Avoid the tempting shortcut of a long "before" and a short "after"; short windows are noisy, and you will be comparing a stable average against a jittery snapshot.

Use a control area

Here is the single biggest upgrade available to any before/after study: measure the same change in a **control area** — a similar place that did *not* get the intervention. Suppose incidents dropped 15 percent near the new streetlights. Impressive — until you check the control neighborhoods and find they *also* dropped 12 percent, because the whole city was trending down. The honest effect is the difference between the two changes: your area improved about 3 points more than the background trend. Statisticians call this **difference-in-differences**, and while the formal version comes with machinery, the plain version — always subtract the background trend — is a mindset you can adopt today. Choose controls that resemble the treated area in the ways that matter (density, land use, baseline levels), and choose them *before* you look at their results.

Decide the measure before you peek

Write down, in advance, the one or two metrics that will define success: which incident types, which distance from the intervention, which statistic. If you instead compute twenty variations and report the flattering one, you are not analyzing — you are shopping. This pre-commitment feels bureaucratic and is the difference between evidence and advocacy.

> **Tip:** Also write down, in advance, what result would convince you the intervention *failed*. If no conceivable data pattern would count as failure, the study is a press release with maps.

Honest Pitfalls

Every analysis method in this chapter shares one silent assumption: that the data-generating process was steady over time, so changes in the data reflect changes in the

world. That assumption fails constantly, and it fails in ways that produce beautiful, publishable, wrong maps.

Uneven collection effort over time

This is the pitfall that deserves top billing. Reported data measures two things multiplied together: how much happened, and how much of it got recorded. When recording effort changes, the data changes — with no change in reality.

The classic offenders: a city launches a new 311 app (the non-emergency service-request line) and "potholes" triple overnight (reporting got easier, not roads worse); a police department adds officers to a district and recorded incidents rise (more eyes, not more crime); a sensor network gets three new stations in the north and the north "heats up"; a volunteer science project goes viral and species observations explode along the routes volunteers happen to walk. Crowdsourced and opportunistic data — anything where people choose when and whether to report — is soaked in this problem.

Defenses, in rough order of power. **Ask how the data was made**, period by period: were there system changes, staffing changes, app launches, form redesigns, outreach campaigns? The metadata and the people who run the collection know things no algorithm can infer (Chapter 5 covers evaluating data sources). **Normalize by effort where you can**: incidents per patrol-hour, detections per active sensor, observations per participant. **Look for fingerprints of effort change**: a jump that occurs simultaneously across all categories and all districts on a specific date is almost always administrative, not real. **Restrict to consistently collected subsets**: if only the downtown sensors existed for the whole study period, analyze downtown for trends and use the full network only for recent snapshots.

> **Watch out:** Emerging hot spot analysis has no idea your reporting system changed in 2023. It will faithfully label the affected areas "new" or "intensifying" hot spots, wrap them in the authority of statistical significance, and hand you a map that is precisely, rigorously measuring your own IT department's rollout schedule. Statistical significance means "unlikely to be random chance." It does not mean "caused by the thing you care about."

History that quietly disappears

Feature layers store *current* state. When an editor fixes a record, the old value is gone unless something preserved it. If your workflow updates features in place — inspection status flipped from "open" to "closed," geometries corrected, duplicates deleted — then the layer you analyze today is not the layer that existed last year, and a naive time analysis on it is archaeology on a renovated site. Defenses: turn on whatever edit-history or versioning capabilities your setup offers (Chapter 13 discusses editing workflows and Chapter 35 covers the Enterprise-side options), or simply export a dated snapshot of important layers on a schedule. A folder of monthly snapshots is an unglamorous, bulletproof time machine.

Boundaries and categories that shift mid-stream

If police districts were redrawn in the middle of your study period, "District 4 before" and "District 4 after" are different pieces of ground wearing the same name. If the incident-type codes were reorganized, "vandalism" may have absorbed or shed subcategories. Both produce steps in the data that look like real change. When boundaries shift, re-aggregate all periods to a neutral, stable geography — a grid you control is the standard fix. When categories shift, build a crosswalk table mapping old codes to new before comparing anything.

Clocks and calendars

Small, dumb, devastating: time zone misconfiguration shifts every hour-of-day analysis; daylight saving transitions create a phantom missing hour each spring and a doubled one each fall; some source systems stamp midnight on any record whose real time was unknown, creating a fake spike at 12:00 a.m. Before trusting any hour-of-day or day-of-week pattern, chart the raw distribution and look for suspicious spikes at midnight, on the first of the month, and on January 1st — the traditional dumping grounds for unknown dates.

The bin-edge illusion

Any pattern that appears at one bin width and vanishes at another is probably an artifact of where the bin edges fell. Statisticians know this as the **modifiable temporal unit problem** — the time-axis twin of the aggregation hazard you met with polygon boundaries in Chapter 17 — but the practical rule is simpler: robust findings survive re-binning. Make re-binning a standard part of your checks, not an afterthought.

A Working Checklist

Space-time analysis rewards paranoia applied in the right order. Before you trust any temporal finding — yours or anyone's — walk this list:

1. Is the event-date field a true date type, in the right time zone, recording when things *happened* rather than when they were typed in?
2. Are the comparison windows equal in length and seasonally matched, with partial bins trimmed?
3. Did collection effort, system configuration, boundaries, or category definitions change during the study period?
4. Do the conclusions survive a different bin width, and (for percent changes) are the baselines large enough to mean anything?
5. Is there a control comparison — a background trend subtracted — before anyone claims an intervention worked?
6. For anything mined from a space-time cube: does the pattern form coherent patches rather than isolated significant cells, and does it survive a rebuilt cube?

If a finding clears all six, it is worth acting on. From here, the natural next steps are Chapter 28 (Dashboards) for putting live temporal indicators in front of decision-makers, Chapter 22 (Geoprocessing in Depth) and Chapter 25 (ModelBuilder) for automating the period-over-period comparisons you will inevitably be asked to repeat, and Chapter 36's worked project, which threads a time-aware analysis from raw data to published product.
