Focus-Glue-Context Fisheye Transformations for Spatial Visualization

by Thanh Cuong Nguyen, Michael Lydeamore, and Dianne Cook

Abstract Fisheye views magnify local detail while preserving context, yet projection-aware, scriptable tools for R spatial analysis remain limited. mapycusmaximus introduces a Focus–Glue–Context (FGC) fisheye transform for numeric coordinates and sf geometries. Acting radially around a chosen center, the transform defines a magnified focus (r_in), a smooth transitional glue zone (r_out), and a fixed exterior. Distances expand or compress via a zoom factor and power-law squeeze, with an optional angular twist enhancing continuity. The method is projection-conscious: lon/lat inputs are reprojected to suitable CRSs (e.g., GDA2020/MGA55), normalized for stable parameter control, and restored afterward. A geometry-safe engine (st_transform_custom) supports all feature types, maintaining ring closure and metadata. The high-level sf_fisheye() integrates with tidyverse, ggplot2, and Shiny, with built-in datasets and tests ensuring reproducibility. By coupling coherent radial warps with tidy, CRS-aware workflows, mapycusmaximus enables spatial exploration that emphasizes local structure without losing global context.

1 Introduction

Maps that reveal fine local structure without losing broader context face a persistent challenge: zooming in hides regional patterns, while small-scale views suppress local detail. Traditional solutions—insets, multi-panel displays, aggressive generalization—break spatial continuity and increase cognitive load. What if we could smoothly magnify a metropolitan core *while keeping it embedded* in its state-level context?

This package implements a Focus–Glue–Context (FGC) fisheye transformation that continuously warps geographic space: a chosen focus region magnifies, surrounding areas compress into a "glue" transition zone, and outer context remains stable. Unlike discrete zoom levels or disconnected insets, the transformation operates directly on vector geometry coordinates, preserving topology and enabling reproducible, pipeline-friendly cartography within R's sf and ggplot2 ecosystem.

The intellectual lineage of focus+context visualization traces back to Furnas (1986)'s *degree-of-interest* function, which formalized how to prioritize salient regions while retaining global structure. Sarkar and Brown (1992) and Sarkar and Brown (1994) extended this to geometric distortion, demonstrating smooth magnification transitions for graph visualization. Subsequent innovations explored diverse lenses: hyperbolic geometry for hierarchies (Lamping et al., 1995), distortion-view frameworks (Carpendale and Montagnese, 2001), and "magic lens" overlays (Bier et al., 1993). By 2008, Cockburn et al. (2008)'s comprehensive review synthesized two decades of research across overview+detail, zooming, and focus+context paradigms.

In cartography, the need for nonlinear magnification emerged independently. Snyder (1987) developed "magnifying-glass" azimuthal projections with variable radial scales—mathematical foundations still cited today. Harrie et al. (2002) created variable-scale functions for mobile devices where user position appears large-scale against small-scale surroundings. The crucial breakthrough came from Yamamoto et al. (2009) and Yamamoto et al. (2012): their **Focus+Glue+Context model** introduced an intermediate "glue" region that absorbs distortion, preventing the excessively warped roads and boundaries that plagued earlier fisheye maps. This three-zone architecture proved particularly effective for pedestrian navigation and mobile web services.

Parallel developments in statistical graphics tackled the "crowding problem"—high-dimensional data collapsing into projection centers. van der Maaten and Hinton (2008)'s t-SNE uses heavy-tailed distributions to spread points, while McInnes et al. (2020)'s UMAP

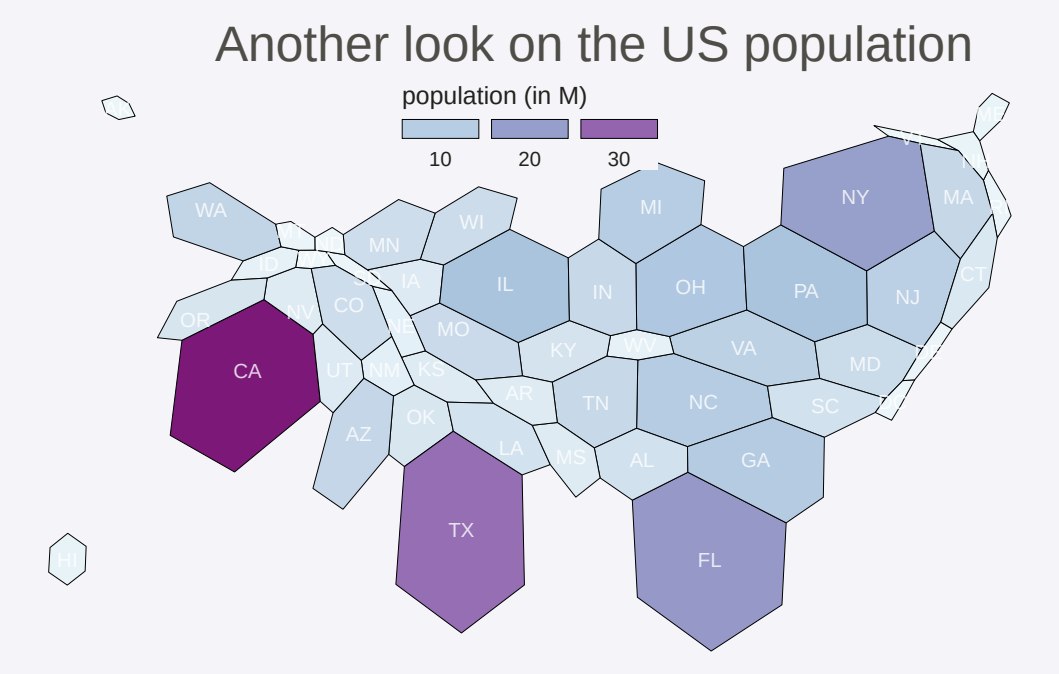
leverages topological methods. Most relevant to our geometric approach: Laa et al. (2020) applies *radial transformations* to tour projections, maintaining the interpretability of linear methods while mitigating overplotting. Implemented in R's tourr package, it demonstrates how well-designed radial warps can reveal structure without the distortions of fully nonlinear embeddings.

Within R's spatial ecosystem, sf (Pebesma, 2018) provides robust vector handling and CRS transformations, while ggplot2 (Wickham, 2016) offers declarative visualization grammar. Yet a gap remained: existing tools addressed *related* distortion needs but not continuous geometric fisheye lenses. This package fills that niche by formalizing an sf-native FGC radial model with controllable zone parameters, optional angular effects, automatic normalization, and safe geometry handling across points, lines, and polygons.

2 Background: Alternative Approaches to the Detail-Context Problem

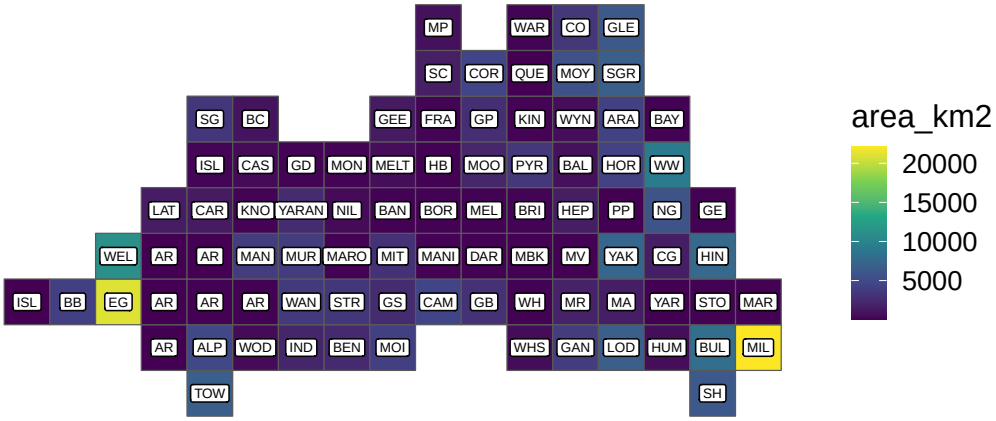
Before diving into fisheye mechanics, it's worth understanding how R's spatial ecosystem currently handles the detail-versus-context tradeoff—and why those solutions, while valuable, leave room for continuous lens-based warping.

Cartograms: Thematic distortion. The cartogram family (Gastner and Newman, 2004) intentionally distorts geographic areas to encode variables—population density reshapes regions so area becomes proportional to demographic weight.



This fundamentally differs from focus+context: cartograms *substitute* spatial accuracy for data encoding, often severely disrupting shapes and adjacencies. A population cartogram makes California balloon while Wyoming shrinks, trading geographic fidelity for thematic insight. FGC fisheye, conversely, preserves relative positions and topology while magnifying a *chosen* spatial region, not a data-driven variable. The use cases diverge: cartograms answer "how does this variable dominate space?" while fisheye lenses answer "what local detail exists within this broader geography?"

Hexagon tile maps: Discrete abstraction. Packages like `geogrid` and visualizations using `sf::st_make_grid()` replace irregular polygons with regular hexagonal or square tiles, each representing an administrative unit.



As seen in the plot above, tile maps *abstracts away* precise geography entirely, treating space as a topology-preserving tessellation where "neighbors touch" matters more than accurate boundaries. Tile maps excel at avoiding size bias (Mildura gets equal visual weight to Yarra) and creating aesthetic, clutter-free layouts. However, they abandon continuous spatial relationships: you cannot identify precise locations, measure distances, or overlay point data meaningfully. Hexbin aggregation for point data (via `ggplot2::geom_hex()`) serves a different purpose—density estimation—rather than focus+context navigation.

Multi-panel approaches: Spatial separation. Tools like `cowplot::ggdraw()`(Wilke, 2025) create side-by-side views: one panel shows overview, another shows zoomed detail.

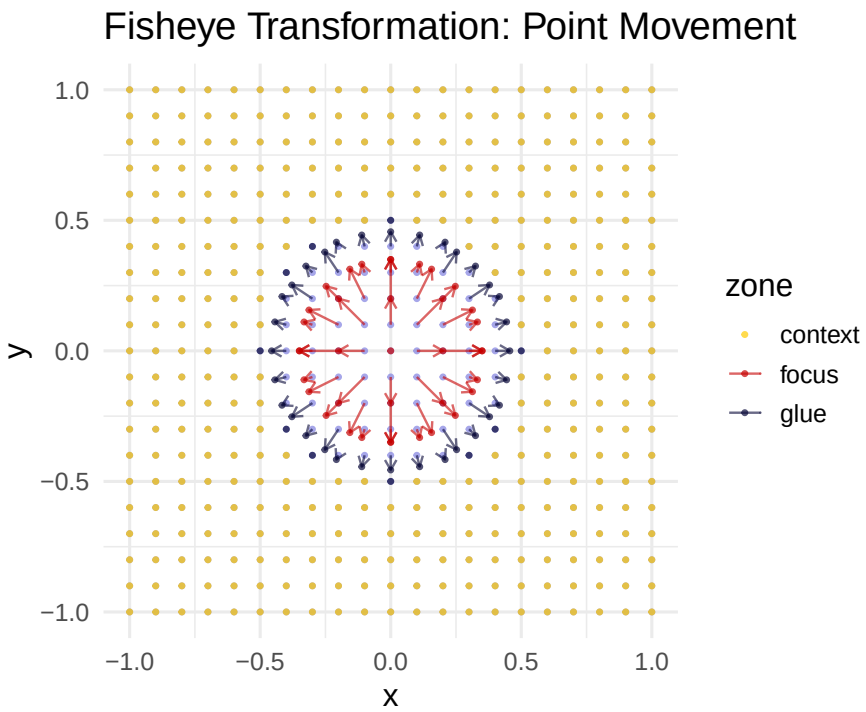
These are effective for static reports but require viewers to mentally integrate separate views, and they don't preserve the *embedded* relationship between focus and context within

a single continuous geography. Futhermore, if you introduce one or more elements into the plot like filling value equal to a variable, the audience will have a hard time identify the zoomed detail.

Why FGC fisheye offers something distinct. None of these approaches provide *continuous geometric magnification within a single, topology-preserving map*. Cartograms distort for data, not user-chosen focus. Tile maps abstract away geography. Multi-panel tools spatially separate context. The fisheye lens keeps everything in one frame—roads bend smoothly, metropolitan detail enlarges, but you still see how the city sits within its state. It's a geometric *warp* rather than a data-driven *substitution* or panel-based *separation*. This matters for use cases like: examining hospital networks in Melbourne while maintaining Victorian context, exploring census tracts in a metro core without losing county boundaries, or analyzing transit lines with their regional hinterland visible.

With this landscape established, we now turn to the technical implementation: how does the Focus–Glue–Context transformation actually work, and how does this package make it accessible within R's spatial workflows?

3 Focus–Glue–Context Transformation



Consider a point $P = (x, y)$ in a projected coordinate system. The analyst chooses a center $C = (c_x, c_y)$ and two radii: r_{in} delineating the focus region and r_{out} marking the glue boundary. Points inside the focus magnify, points between the radii focus on the center and then compress according to a smooth curve, and points outside remain unchanged. This radial scheme keeps angular coordinates intact, thereby preserving bearings and relative direction.

3.1 Algorithm

Let (r, θ) denote the polar form of P relative to C. The transform defines a new radius r' via a piecewise function:

$$r' = \begin{cases} \min(z \cdot r, r_{\text{in}}) & \text{if } r \le r_{\text{in}}, \\ r_{\text{in}} + (r_{\text{out}} - r_{\text{in}}) \cdot h(u; s) & \text{if } r_{\text{in}} < r \le r_{\text{out}}, \\ r & \text{if } r > r_{\text{out}}, \end{cases}$$

where $z > 1$ is the zoom factor within the focus, $s \in (0, 1]$ controls glue compression, and $u = (r - r_{\text{in}})/(r_{\text{out}} - r_{\text{in}})$ normalises the glue radius to $[0, 1]$. The function $h(u; s)$ is chosen so that $h(0; s) = 0$, $h(1; s) = 1$, and both the first derivatives and the radii match at the boundaries. We adopt a symmetric power curve:

$$h(u; s) = \begin{cases} \frac{1}{2} \cdot u^{1/s} & \text{if } 0 \le u \le 0.5, \\ 1 - \frac{1}{2} \cdot (1 - u)^{1/s} & \text{if } 0.5 < u \le 1, \end{cases}$$

which compresses radii near both boundaries and emphasises the mid-glue region. Analysts seeking outward compression can choose alternative methods (e.g., the `"outward"` mode) that bias the curve towards r_{out}.

The transform optionally introduces rotation within the glue zone to accentuate the flow from detail to context. Let $\phi(u)$ denote the angular adjustment. We employ a bell-shaped profile: $\phi(u) = \rho \cdot 4u(1 - u)$, where ρ is the revolution parameter (in radians). This function peaks at the glue midpoint and vanishes at the boundaries, ensuring continuity.

The focus and glue formulas ensure r' and $\frac{\partial r'}{\partial r}$ are continuous at r_{in} and r_{out}. Continuity is essential for maintaining perceptual coherence and avoiding visible creases along the glue boundary. In practice, analysts can tune parameters interactively to obtain the desired amount of magnification and compression, knowing the transform behaves smoothly across the domain.

3.2 Implementation

Coordinate workflows

Spatial datasets vary widely in their coordinate reference systems (CRS), spatial extent, feature types, and attribute schemas. **mapycusmaximus** therefore adopts a disciplined, staged workflow that separates concerns and makes each transformation explicit and auditable. The same pipeline is applied regardless of whether the input is an `sf` data frame or an `sfc` vector; the only difference is whether non-geometry columns are present and preserved.

The preflight step sanitises the input. Empty geometries are removed because downstream operations such as coordinate extraction and ring reconstruction assume at least one vertex. The function `sf::st_zm()` is applied to drop `Z/M` dimensions, ensuring that subsequent matrix operations are performed on two-dimensional coordinates. This choice avoids silent recycling or attribute loss that can occur when three or four column matrices are passed into purely planar formulae. If a user supplies a focal centre, the package validates its form early—accepting either a numeric pair, an `sf/sfc` geometry, or a normalised pair—and records any associated CRS for later transformation. Inputs with missing CRS are handled conservatively: numeric `center` values are assumed to be in working-CRS units only when they do not look like lon/lat; `sf/sfc` centres without CRS produce an explicit error to guard against unit confusion.

Next, a working projected CRS is selected. If `target_crs` is specified, it is used verbatim. Otherwise, when the input is geographic (`sf::st_is_longlat(sf_obj)` is `TRUE`), the package chooses a practical projected system based on the layer's centroid. Datasets geographically centred on Victoria, Australia default to GDA2020 / MGA Zone 55 (EPSG:7855), which balances distortion across the state. For other regions, a UTM zone is chosen using the standard zone index computed from longitude and hemisphere. This deterministic rule avoids project-specific heuristics and makes runs reproducible. Because the fisheye is defined in map units, operating in a metric projection is important for interpretability of distances during normalisation; nonetheless, the original input CRS is retained and restored

after the warp to keep downstream code unchanged.

With a working CRS in place, the function computes a bounding box and derives scale factors for normalisation. Let the half-spans of the bbox be s_x and s_y. When `{preserve_aspect = TRUE}` (the default), a uniform scale $s = \max(s_x, s_y)$ is used so that unit radii have a consistent meaning along both axes, preventing elliptical exaggeration of the focus. When `preserve_aspect = FALSE`, axes are scaled independently by s_x and s_y. Degenerate cases—where a layer collapses to a line or a point—are handled by substituting a unit scale for any zero half-span to avoid division by zero; such inputs are uncommon yet arise in diagnostic pipelines and deserve robust handling.

Centre resolution occurs before normalisation because both numeric and geometry-supplied centres must be expressed in the working CRS. The internal helper `.resolve_center()` implements precedence and conversion rules: if `center` is an `sf/sfc` object, it is combined and reduced to a centroid when necessary, transformed to the working CRS, and converted to a numeric pair; if it is numeric with a declared `center_crs`, a trivial `st_transform()` is applied; if it is numeric without a CRS, a lon/lat heuristic is used; if `normalized_center = TRUE`, the pair is interpreted in $[-1, 1]$ relative to the bbox midpoint with either uniform or per-axis scaling in accordance with `preserve_aspect`. If no centre is given, the bbox midpoint serves as a sensible default that is stable under reprojection.

The core of the pipeline is a coordinate-wise map, but spatial objects require structured handling. `sf_fisheye()` defines a small wrapper `wrapped_fisheye()` that composes three operations: normalise the x, y matrix around the chosen centre, apply `fisheye_fgc()` to produce the warped coordinates in unit space, and denormalise back to map units. This wrapper is then threaded through `st_transform_custom()`, a geometry-aware iterator that extracts coordinates from each POINT, LINESTRING, POLYGON, or MULTIPOLYGON; applies the user-supplied transform function; and rebuilds the geometry. For polygons, rings are processed independently (using ring and part indices `L1` and `L2` from `sf::st_coordinates()`) and explicitly re-closed to ensure that the first and last vertices match. This avoids topological artefacts (e.g., sliver triangles) that can arise when a warp perturbs the final vertex away from the start.

After transformation, the geometry column is reassembled into an `sfc` with the working CRS and then, if necessary, transformed back to the original CRS. Attributes are preserved by design for `sf` inputs because only the geometry column is replaced; row order and feature identity remain unchanged. This round trip affords a convenient invariant for testing—coordinates expressed in the original CRS match inputs in the context zone to within floating-point tolerance and for user mental models—plots accept the same code as before, simply rendering warped shapes.

Two elements of robustness deserve emphasis. First, numeric stability at zone boundaries is addressed in `fisheye_fgc()` by clamping expansions in the focus so that radii do not exceed r_{in}, and by using a smooth power curve in the glue so that derivatives match across boundaries. Second, per geometry error handling in `st_transform_custom()` prevents a single malformed feature from aborting the entire warp: failures yield an empty geometry of the appropriate family with a warning, allowing batch processing to continue while signalling the need for data repair.

Performance concerns guided the choice to operate on plain matrices wherever possible and to minimise object churn. Coordinate extraction and reconstruction are the main overhead for large polygonal layers; the radial mapping itself is vectorised and runs in linear time in the number of vertices. For multi-layer maps, the same parameter set (centre, radii, zoom, squeeze, method, revolution) should be applied consistently to each layer to maintain spatial alignment; the pipeline ensures that doing so yields coincident warps provided the same working CRS and normalisation regime are used. In practice, analysts often capture the parameter set in a list and pass it to `sf_fisheye()` for each layer, simplifying reproducible workflows and parameter sweeps.

Table 1: Comparison of coordinate

x_new	y_new	x	y	zone	r_orig	r_new
-1.000	-1.000	-1.0	-1	context	1.414	1.414
-0.900	-1.000	-0.9	-1	context	1.345	1.345
-0.800	-1.000	-0.8	-1	context	1.281	1.281
-0.111	-0.332	-0.7	-1	focus	0.316	0.350
0.000	-0.350	-0.6	-1	focus	0.300	0.350
0.111	-0.332	-0.5	-1	focus	0.316	0.350
0.000	-0.500	-0.4	-1	glue	0.500	0.500
-0.300	-0.400	-0.3	-1	glue	0.500	0.500
-0.208	-0.416	-0.2	-1	glue	0.447	0.466

Software architecture

The software architecture reflects a clear separation of responsibilities between numeric mapping, spatial orchestration, geometry reconstruction, and user-facing utilities. This separation allows the core transform to remain small, testable, and language-agnostic, while **sf**-specific concerns are isolated in thin wrappers. The following narrative describes the principal modules and their interactions, with attention to extension points and design trade-offs.

At the heart of the package is `fisheye_fgc()`, a self-contained, vectorised function that maps an $n \times 2$ matrix of planar coordinates to a new $n \times 2$ matrix via the Focus-Glue-Context rule. Its contract is intentionally minimal: it accepts only numeric arrays and scalar parameters that define the centre, radii, magnification, compression, method, and revolution.

Internally it computes displacements from the centre, converts to polar form, applies a piecewise radial map with smooth boundary conditions, optionally perturbs the angle by a bell-shaped rotation in the glue, and then converts back to Cartesian coordinates. To support diagnostics, it attaches attributes to the returned matrix: the zone label for each point and the original and new radii. These attributes are consumed by plotting utilities and tests but do not affect downstream geometry reconstruction.

The orchestration of CRS, normalisation, and geometry replacement is handled by `sf_fisheye()`. Its purpose is to present a single, idiomatic R interface for spatial analysts while keeping the numeric core untouched. The function performs: (i) input validation and sanitation; (ii) automatic or user-specified working CRS selection; (iii) centre resolution via the internal helper `.resolve_center()`, which encapsulates precedence rules and CRS conversion; (iv) definition of normalisation and denormalisation closures (uniform or per-axis); (v) construction of a small wrapper that applies normalise-warp-denormalise to any x, y matrix; and (vi) invocation of a geometry walker to apply that wrapper to every feature. The top-level class of the input is preserved, and the original CRS is restored on return. By design, `sf_fisheye()` never reaches into the internals of the numeric mapping; this keeps the transform portable and facilitates future experimentation with alternative warps.

Geometry walking is implemented in `st_transform_custom()` (module `sf_related.R`). The function accepts either an `sf` or `sfc` object plus a user-supplied coordinate transform `transform_fun`. For each feature, it inspects the geometry type and extracts coordinates via `sf::st_coordinates()`. POLYGON and MULTIPOLYGON types are split by ring and part indices (columns `L1`, `L2`) so that each ring is transformed independently. After transformation, the helper enforces explicit ring closure by setting the last vertex equal to the first, which is necessary because not all warps are vertex-preserving and because some upstream datasets omit the closing vertex convention. The output geometries are rebuilt using constructors from **sf** (`st_point()`, `st_linestring()`, `st_polygon()`, `st_multipolygon()`), combined into an `sfc` with the original CRS, and spliced back into an `sf` if appropriate. Error handling is per-geometry: failures raise a warning and yield an empty geometry of the same family, thus preserving list structure and indices while signalling problems to the user.

Utilities live in `utils.R`. `create_test_grid()` generates dense regular grids for diagnostics and performance testing; `classify_zones()` labels points according to the FGC partition; and `plot_fisheye_fgc()` provides a side-by-side visualisation that mirrors the attributes attached by `fisheye_fgc()`. These helpers are deliberately simple and free of side-effects so that users can copy or modify them in downstream projects without incurring hidden dependencies. Dataset documentation in `data.R` accompanies example layers (`vic`, `vic_fish`, `conn_fish`) used in the article and tests.

Two internal design choices warrant discussion. First, the centre-resolution helper `.resolve_center()` concentrates all logic for interpreting `center` in different forms and CRS. Centralising this logic avoids subtle discrepancies that would otherwise arise if different call paths attempted to "do the right thing" in isolation. Second, normalisation occurs in closures that are constructed once per call to `sf_fisheye()`, avoiding repeated conditionals inside the hot loop of coordinate mapping and keeping the path to `fisheye_fgc()` as short as possible. Both choices contribute to clarity and performance without sacrificing flexibility.

The test suite (`tests/testthat/`) mirrors the modular structure. Unit tests for `fisheye_fgc()` cover boundary behaviour (exactly on r_{in}, r_{out}), zone labelling consistency, and the monotonicity of the radial map. Tests for `sf_fisheye()` verify CRS round-trips, centre precedence and conversion, and the invariants associated with aspect preservation. Geometry-walker tests assert that polygon rings are re-closed, that multi-part objects retain part counts, and that attribute columns in `sf` frames are untouched. Performance tests use synthetic grids and the packaged datasets to detect regressions in both runtime and memory allocation. Where appropriate, tests exercise error paths to confirm that warnings are issued and structure is preserved.

From an API-design perspective, naming and argument defaults follow tidyverse conventions while prioritising explicitness. Functions are named with verbs (`sf_fisheye()`, `plot_fisheye_fgc()`) or descriptive nouns (`create_test_grid()`); parameters use snake_case and align across functions so that mental models transfer. Backward compatibility is maintained for users who prefer `cx`, `cy` centres; however, the `center` interface is encouraged because it supports normalized and CRS-aware specifications. The exported surface is intentionally small; the package does not expose low-level ring iterators or CRS heuristics to avoid freezing internals that may evolve. Instead, stability is guaranteed for `fisheye_fgc()`, `sf_fisheye()`, `st_transform_custom()`, and the three utilities documented in the API table.

Finally, the architecture leaves room for extension without breaking existing code. Alternate radial profiles or anisotropic warps can be prototyped by swapping `fisheye_fgc()` in the wrapper while retaining the rest of the pipeline. Additional geometry types (e.g., CURVE, GEOMETRYCOLLECTION) can be supported by extending `st_transform_custom()` with corresponding extraction and rebuild logic. Integration with raster frameworks (e.g., **stars**) would follow a similar pattern: compute grid coordinates, apply the mapping, and resample values as needed. Because the modules are loosely coupled and communicate through simple contracts (matrices in, matrices out), such evolutions can be undertaken incrementally and verified through the existing test scaffolding.

Parameters

The principal user interface is `sf_fisheye()`, which accepts an `sf` or `sfc` object and returns an object of the same top-level class whose geometry has been warped in a projection- aware manner. For clarity, we group arguments into data/CRS handling, centre selection, and radial warping, and we make explicit the invariants enforced by the implementation.

Data and CRS. The argument `sf_obj` supplies the features to be transformed. Before any calculation, empty geometries are removed and Z/M dimensions are dropped using `sf::st_zm()`, so that downstream computation operates on a strict $n \times 2$ coordinate matrix. The optional `target_crs` sets the working projected CRS; if provided, the input is transformed via `sf::st_transform()` and the original CRS is restored on return. When `target_crs = NULL` and the input is geographic (lon/lat), a projected working CRS is chosen

deterministically from the layer's centroid: for Victoria, Australia, GDA2020 / MGA Zone 55 (EPSG:7855) is used; otherwise a UTM zone is inferred by longitude and hemisphere. This choice ensures the fisheye operates in metric units with bounded distortion across the extent of interest. The `preserve_aspect` flag governs normalisation: with `TRUE` (default) a uniform scale $s = \max(s_x, s_y)$ is applied, where s_x, s_y are bbox half-spans; with `FALSE`, independent scales are used per axis. Uniform scaling preserves circular symmetry of the focus and glue; per-axis scaling yields an elliptical interpretation that can be useful for long, narrow extents but should be used deliberately. Degenerate cases ($s_x = 0$ or $s_y = 0$) are handled by substituting a unit scale to avoid division by zero.

Centre selection. The lens centre may be specified in several forms. The preferred interface is `center`, which takes precedence over legacy `cx`, `cy`. If `center` is a numeric pair and `center_crs` is provided (e.g., `"EPSG:4326"`), the point is transformed into the working CRS. If `center_crs` is omitted, a heuristic interprets pairs that lie within $|\text{lon}| \leq 180$, $|\text{lat}| \leq 90$ as WGS84 and transforms them accordingly; otherwise the values are assumed to be already in working-CRS map units. Any `sf/sfc` geometry may be used as `center`; non-point centres are combined and reduced to a centroid and then transformed to the working CRS, which is often convenient when the focal area is a polygon (e.g., a CBD boundary) or a set of points (e.g., incident locations). Finally, when the argument `{normalized_center = TRUE}`, `center` is interpreted as a pair in $[-1, 1]$ relative to the bbox midpoint and the chosen normalisation (uniform or per-axis). Normalised centres make parameter sets portable across datasets of different extents and are a natural fit for parameter sweeps in reproducible pipelines. If no centre is supplied, the bbox midpoint is used; this default is stable under reprojection.

Radial warping. The radii `r_in` and `r_out` define the focus and glue boundaries in the normalised coordinate space and must satisfy `r_out > r_in`. The interpretation of these radii depends on `preserve_aspect`. With uniform scaling, a circle of radius r_{in} in unit space corresponds to a circle of radius $r_{\text{in}}\, s$ in map units; with per-axis scaling, the corresponding shape is an axis-aligned ellipse with semi-axes $r_{\text{in}} s_x$ and $r_{\text{in}} s_y$. Inside the focus, distances from the centre are multiplied by `zoom_factor`; to prevent overshoot, the implementation clamps r' so that points do not cross the r_{in} boundary. Across the glue, `squeeze_factor \in (0,1]` controls how strongly intermediate radii compress: smaller values create tighter compression near the boundaries and a more pronounced "shoulder" in the middle of the glue; larger values approach a linear transition. The `method` selects the family of curves used in the glue. The default `"expand"` applies a symmetrical power law that expands inward and outward halves of the glue to maintain visual balance around the midpoint; `"outward"` biases the map towards r_{out}, keeping the outer boundary steadier and pushing more deformation into the inner portion of the glue. The optional `revolution` parameter adds a bell-shaped angular twist inside the glue of magnitude $\rho\, 4u(1-u)$, where u is the normalised glue radius. This rotation vanishes at both glue boundaries and peaks at the midpoint, preserving continuity. Positive values rotate counter-clockwise, negative values clockwise; values are specified in radians.

Inter-parameter interactions and invariants. The following constraints and behaviours are enforced: $r_{\text{out}} > r_{\text{in}} > 0$; `zoom_factor` ≥ 1 (values close to one yield gentle focus); `squeeze_factor` in $(0, 1]$ ($= 1$ approaches linear); and monotonicity of the radial map so that ordering by distance from the centre is preserved. The choice of `preserve_aspect` affects the physical size of radii and thereby the impact of a given parameter set on different datasets; using uniform scaling with a normalised centre yields the most portable configurations. Twisting via `revolution` is confined to the glue; it does not change radii and therefore does not affect the classification of points into zones. Because angles are modified only in the glue, bearings inside the focus and in the context are preserved.

Return value and side effects. The function returns an object of the same top-level class as its input (`sf` or `sfc`). For `sf` inputs, non-geometry columns are preserved verbatim; only the geometry column is replaced. The original CRS is restored before return so that downstream plotting and analysis code does not need to change. On malformed geometries, the implementation emits a warning and returns an empty geometry of the appropriate family to preserve row count and indices. For exploratory diagnostics, the low-level `fisheye_fgc()`

returns a coordinate matrix with attributes "zones", "original_radius", and "new_radius"; these can be used to plot scale curves and verify parameter effects prior to applying the transform to complex geometries.

Common choices

Although the parameter space is continuous, certain regimes recur in practice and can serve as reliable starting points. We describe these regimes and articulate the trade-offs that motivate each choice. The recommendations assume the default preserve_aspect = TRUE; when per-axis scaling is enabled, translate radii to semi-axes using the bbox half-spans.

Balanced metropolitan focus within a state. A common narrative emphasises a city region while retaining a recognisable state outline. Choose r_{in} to enclose the urban footprint (often 0.30–0.35) and r_{out} to provide a broad glue buffer (0.55–0.70). A zoom_factor of 1.5–2.0 provides visible enlargement without overwhelming the transition. Pair this with squeeze_factor = 0.25\text{--0.40}, which gently compresses surroundings while maintaining smoothness. The "expand" method yields a balanced appearance in which the mid-glue region visibly bridges detail and context. If preserving the outer coastline or boundary is paramount (e.g., for policy maps where the edge must remain stable), "outward" can be substituted to reduce outer drift at the cost of slightly stronger inner squeeze.

Dense line networks and flows. When the layer of interest is line-heavy (transport corridors, flows, hydrology), kink introduction and overplotting are the primary risks. Reduce glue compression and avoid large twists: squeeze_factor \ge 0.35 (ideally 0.40–0.60) coupled with revolution \le 0.2 radians keeps linework legible while still communicating focus. The "expand" method is generally preferable because its symmetric treatment of the glue reduces inflections near r_{in} and r_{out}. When in doubt, plot a radius-vs-radius diagnostic from fisheye_fgc() to confirm that the derivative remains near one at boundaries.

Polygon-dominated maps and choropleths. For administrative regions, land-use parcels, or other polygon-dense layers, slightly stronger compression in the glue is tolerable because viewers rely on silhouette and adjacency rather than precise edge angles. Settings such as {squeeze_factor = 0.25\text{ - }0.40} with zoom_factor = 1.6\text{ - }2.2} and either "expand" or "outward" often work well. We recommend revolution = 0 for publication unless the swirl is part of the intended rhetoric; twists, while visually engaging, can distract from choropleth encoding and complicate legend interpretation.

Small multiples and parameter sweeps. Analysts frequently compare scenarios across maps (e.g., different thresholds or temporal slices). Portability of parameters is maximised by using a normalised centre (normalized_center = TRUE) with preserve_aspect = TRUE. This yields consistent radii across datasets of different extent and makes small multiples directly comparable. A pattern that works well is to fix $r_{\text{in}}, r_{\text{out}}$ and squeeze_factor, and vary zoom_factor over a short range (e.g., 1.3, 1.6, 2.0). Faceting these outputs produces a transparent narrative of how emphasis changes with magnification.

Choosing radii from map scale. When stakeholders communicate distances in kilometres or miles, convert desired physical radii to unit radii using the bbox half-span. With {preserve_aspect = TRUE}, $r_{\text{in}} = d/s$ where d is the intended focus radius in map units (metres for metric projections) and s is the larger half-span of the bbox. This rule allows quick calibration: for a state with half-span 250 km, a desired 75 km focus corresponds to $r_{\text{in}} \approx 0.30$. For per-axis scaling, choose semi-axes independently: $r_{\text{in,x}} = d_x/s_x$, $r_{\text{in,y}} = d_y/s_y$, noting that the current implementation interprets r_{in} as a single scalar and therefore realises an ellipse only through preserve_aspect = FALSE.

Centres for reproducibility. To avoid ambiguity in collaborative settings, prefer specifying center either as an sf geometry (whose CRS is explicit) or as a lon/lat pair with center_crs = "EPSG:4326". Numeric pairs without CRS are accepted but rely on heuristics. When the focal area is itself a polygon or multi-polygon, passing that object as center ensures the centroid is derived from the same dataset used for the map, improving reproducibility and intent.

CRS considerations. Leaving `target_crs = NULL` suffices for most lon/lat inputs because the working CRS is chosen deterministically. Projects that maintain a standard grid (e.g., local government dashboards) should specify `target_crs` to improve cross-report comparability. Avoid switching working CRS between layers that will be overlaid; doing so changes the meaning of normalised radii and will misalign warps.

Publication vs. exploration. For exploratory notebooks and talks, small nonzero `revolution` values (≤ 0.3 radians) can help audiences perceive continuity across the glue. For manuscripts and dashboards, prefer `revolution = 0`. Similarly, start with `"expand"` and adopt `"outward"` only when outer stability is an explicit requirement. Always annotate or at least describe the distortion in figure captions so readers do not mistake warped areas for standard projections.

4 Examples of use

SHOW THE WAYS THAT IT CAN BE USED FOR THE VICTORIAN AMBULANCE DATA: Just the map with hospital locations, map with transfers, map with convex hulls, map with two focal points, then maybe a raster map

5 Discussion

HERE YOU SUMMARISE WHAT THE PAPER CONTRIBUTED IN ONE PARAGRAPH AND SUGGEST NEW WORK THAT MIGHT BE DONE THAT YOU DIDN'T HAVE TIME TO DO

References

E. A. Bier, M. C. Stone, K. Pier, W. Buxton, and T. D. DeRose. Toolglass and magic lenses: The see-through interface. In *Proceedings of SIGGRAPH '93*, pages 73–80, 1993. doi: 10.1145/166117.166126. [p1]

M. S. T. Carpendale and C. Montagnese. A framework for unifying presentation space. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, pages 61–70, 2001. doi: 10.1145/502348.502371. [p1]

A. Cockburn, A. Karlson, and B. B. Bederson. A review of overview+detail, zooming, and focus+context interfaces. *ACM Computing Surveys*, 41(1):1–31, 2008. doi: 10.1145/1456650.1456652. [p1]

G. W. Furnas. Generalized fisheye views. In *Proceedings of CHI '86*, pages 16–23, 1986. doi: 10.1145/22627.22342. [p1]

M. T. Gastner and M. E. J. Newman. Diffusion-based method for producing density-equalizing maps. *Proceedings of the National Academy of Sciences*, 101(20):7499–7504, 2004. doi: 10.1073/pnas.0400280101. URL https://www.pnas.org/doi/abs/10.1073/pnas.0400280101. [p2]

L. Harrie, L. T. Sarjakoski, and L. Lehto. A variable-scale map for small-display cartography. In *Joint International Symposium on Geospatial Theory, Processing and Applications*, pages 1–6, 2002. [p1]

U. Laa, D. Cook, and S. Lee. Burning sage: Reversing the curse of dimensionality in the visualization of high-dimensional data, 2020. URL https://arxiv.org/abs/2009.10979. [p2]

J. Lamping, R. Rao, and P. Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings of CHI '95*, pages 401–408, 1995. doi: 10.1145/223904.223956. [p1]

L. McInnes, J. Healy, and J. Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020. URL https://arxiv.org/abs/1802.03426. [p1]

E. Pebesma. Simple features for r: Standardized support for spatial vector data. *The R Journal*, 10:439–446, 2018. ISSN 2073-4859. doi: 10.32614/RJ-2018-009. https://doi.org/10.32614/RJ-2018-009. [p2]

M. Sarkar and M. H. Brown. Graphical fisheye views of graphs. In *Proceedings of CHI '92*, pages 83–91, 1992. doi: 10.1145/142750.142763. [p1]

M. Sarkar and M. H. Brown. Graphical fisheye views. *Communications of the ACM*, 37(12): 73–84, 1994. doi: 10.1145/198366.198384. [p1]

J. P. Snyder. "magnifying-glass" azimuthal map projections. *The American Cartographer*, 14(1): 61–68, 1987. [p1]

L. van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL http://jmlr.org/papers/v9/vandermaaten08a.html. [p1]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer, 2016. doi: 10.1007/978-3-319-24277-4. [p2]

C. O. Wilke. *cowplot: Streamlined Plot Theme and Plot Annotations for 'ggplot2'*, 2025. URL https://wilkelab.org/cowplot/. R package version 1.2.0. [p3]

D. Yamamoto, S. Ozeki, and N. Takahashi. Wired fisheye lens: A motion-based improved fisheye interface for mobile web map services. In A. S. Carswell, James D.and Fotheringham and G. McArdle, editors, *Web and Wireless Geographical Information Systems*, pages 153–170, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-10601-9. doi: https://doi.org/10.1007/978-3-642-10601-9_11. [p1]

D. Yamamoto, S. Ozeki, N. Takahashi, and S. Takahashi. A fusion of multiple focuses on a focus+glue+context map. In *Advances in Cartography and GIScience*, pages 23–37. 2012. doi: 10.1007/978-3-642-29934-6_2. [p1]

Thanh Cuong Nguyen
Monash University
Department of Econometrics and Business Statistics
Melbourne, Australia
https://alex-nguyen-vn.github.io
ORCiD: 0000-0000-0000-0000
thanhcuong10091992@gmail.com

Michael Lydeamore
Monash University
Department of Econometrics and Business Statistics
Melbourne, Australia
https://www.michaellydeamore.com
ORCiD: 0000-0001-6515-827X
michael.lydeamore@monash.edu

Dianne Cook
Monash University
Department of Econometrics and Business Statistics
Melbourne, Australia
https://www.dicook.org
ORCiD: 0000-0002-3813-7155
dicook@monash.edu