Focus-Glue-Context Fisheye Transformations for Spatial Visualization

by Thanh Cuong Nguyen, Michael Lydeamore, and Dianne Cook

Abstract The mapycusmaximus package introduces a Focus-Glue-Context (FGC) fisheye transformation for numeric coordinates and `sf` geometries. Fisheye views magnify local detail while preserving context, by warping radially around a chosen center. The transformation makes a zoomed in focus zone, a smooth transitional glue zone, and a fixed outer context. Distances expand or compress via a zoom factor and a power-law squeeze, with an optional angular twist that enhances continuity. The method is projection-conscious: lon/lat inputs are re-projected to suitable coordinate reference system, normalized for stable parameter control, and restored to the original scale to finish. A geometry-safe engine supports all feature types, maintaining polygon closure and metadata. The resulting data objects integrate with tidy workflow, and can be visualized using ggplot2, enabling spatial exploration that refocuses local structure without losing the global context.

1 Introduction

Maps commonly have important information in small polygons that can be hidden when displayed in regular geographic units. The challenge is to reveal fine local structure without losing broader context. Traditional solutions such as insets and multi-panel displays break spatial continuity and increase cognitive load (Cockburn et al., 2008).

We introduce `mapycusmaximus`, an R package which implements a Focus-Glue-Context (FGC) fisheye transformation that continuously warps geographic space. The transformation magnifies a chosen focus region, compresses surrounding areas into a transitional glue zone, and maintains stability in the outer context. The approach operates directly on vector geometry coordinates, preserves topology, and supports a reproducible, pipeline-oriented cartography within the R `sf` and `ggplot2` ecosystem. An optional glue-zone twist (the `revolution` parameter) can gently rotate features to aid continuity.

The development of focus+context visualization traces back to Furnas (1986)'s *degree-of-interest* (DOI) function, which introduced a formal method to rank information elements by combining intrinsic importance with distance from the user's focus. In this model, items with low DOI are de-emphasized or hidden, enabling emphasis on salient regions without losing global structure. Sarkar and Brown (1992; Sarkar and Brown, 1994) extended this to geometric distortion, demonstrating smooth magnification transitions for graph visualization. Other approaches to this problem include hyperbolic geometry for hierarchies (Lamping et al., 1995), distortion-view frameworks (Carpendale and Montagnese, 2001), and "magic lens" overlays (Bier et al., 1993). Cockburn et al. (2008)'s provides a review of two decades of research across **overview+detail** and **focus+context** paradigms.

In cartography, the need for nonlinear magnification emerged independently. Snyder (1987) developed "magnifying-glass" azimuthal projections with variable radial scales. Harrie et al. (2002) created variable-scale functions for mobile devices where user position appears large-scale against small-scale surroundings. The **Focus+Glue+Context model**, described in Yamamoto et al. (2009) and Yamamoto et al. (2012), introduced the intermediate "glue" region to absorb distortion, preventing excessively warped roads and boundaries that were problematic in earlier fisheye maps. This three-zone architecture proved particularly effective for pedestrian navigation and mobile web services.

Despite the Focus+Glue+Context model being introduced more than a decade ago, there is currently no native R solution for implementing these models. The package `mapycusmaximus` fills this niche by providing a general function to perform these transformations, as well as integrations to work natively within the R spatial ecosystem with `sf` (Pebesma, 2018).

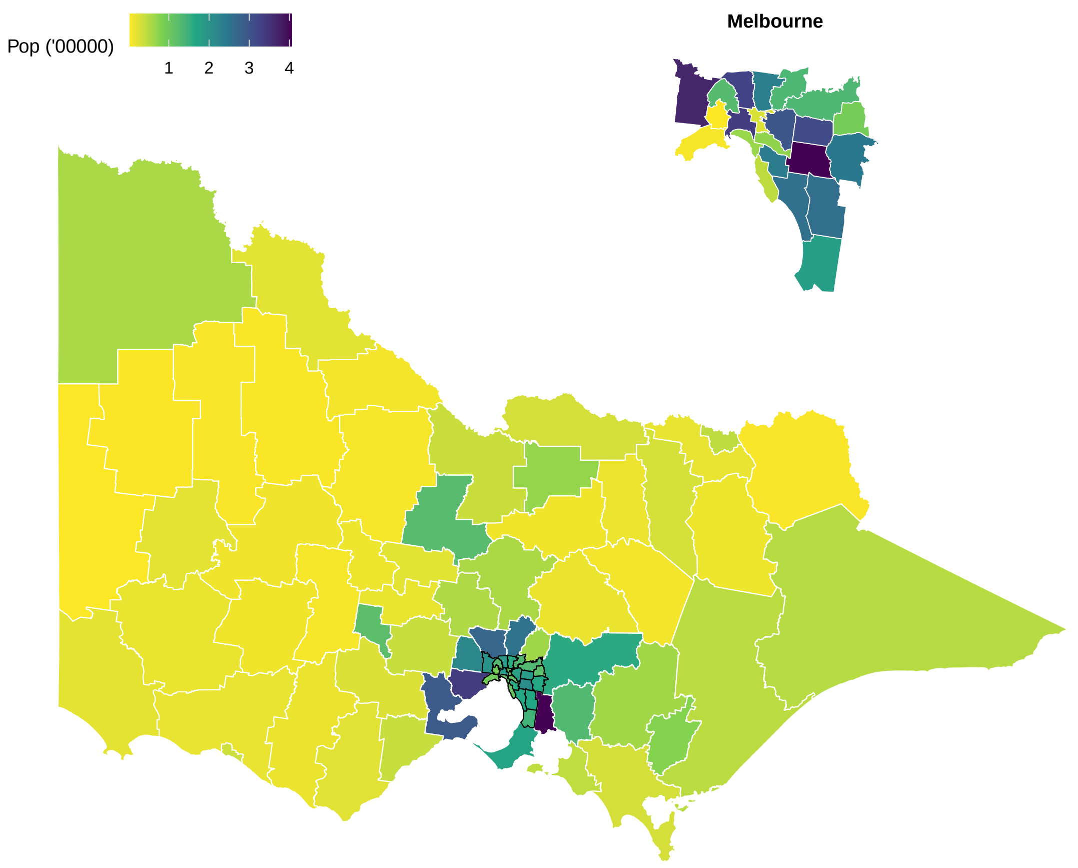


Figure 1: Map of Victorian local government areas (LGAs) colored by population, with an inset showing metropolitan Melbourne at a larger scale. The separation highlights local detail but requires the reader to mentally integrate focus and context across panels.

The paper is organised as follows. Section 2 describes current software available in R with similar functionality. Section 3 provides the basics on getting started using the package, and Section 4 provides the details on the algorithm and how it is implemented. The user can explore the parameters of the transformation using the Shiny app described in Section 5. An application to ambulance transfers in Victoria is shown in Section 6, and the last section describes potential new directions for the package.

2 Resources currently available in R

There are three possible ways to accomplish visualizing maps with re-focused attention on small areas: multi-panel approaches, hexagon tile maps and cartograms. Here we describe the capabilities and limitations of each.

Tools like `cowplot::ggdraw()`(Wilke, 2025) can create multi-panel plots, with one panel showing an overview, and another showing zoomed detail (Figure 1). This is the most common current practice to the detail-versus-context tradeoff, and most familiarfor every day readers. However, they require viewers to mentally integrate separate views, and lose the *embedded* relationship between focus and context with a single continuous geography. Here population is used to color the polygons, with darker indicating higher values. When insets are used to show zoomed detail, it can be more difficult to interpret the color values, and the designer needs to ensure the inset is produced with a scale common to the main plot.

A solution to small area problems is the hexagonal tilemap, such as that implemented by the sugarbag package (Kobakian et al., 2023). The algorithm in this package represents each spatial polygon by a hexagon, and places it as close as possible to the centroid of its

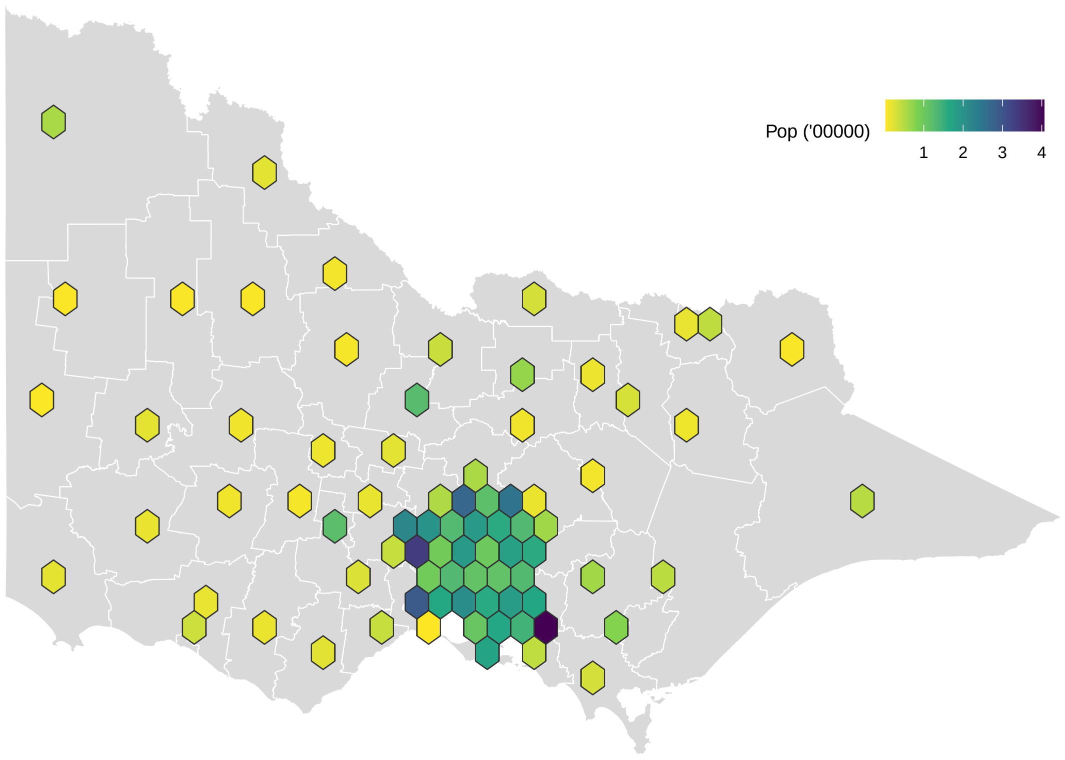


Figure 2: Sugarbag hex tile map illustrating thematic spatial abstraction. Original LGA polygons are replaced by uniform hexagons, with fill indicating poplulation and original geography shown faintly beneath. This representation removes area bias but sacrifices precise geographic location.

respective polygon. Figure 2 shows the hexagon tile map overlaid on a spatial polygon map, constructed on the same data as used in Figure 1. The end result is that information hidden in the small polygons, such as dense populations in inner Melbourne here, are visible. However, the drawback is that it is not possible to place the tiles perfectly aligned with the original geography and preserve neighbourhood structure completely.

Another approach is a cartogram (Gastner and Newman, 2004), where polygons are distorted to reflect the statistic being represented. Figure 3 shows a cartogram for Victorian local government areas (LGAs) where color indicates population, and polygons are distorted proportional to population. Notice how the metropolitan Melbourne LGAs are expanded in size relative to 1. Making the cartogram can be fiddly, and it is not very controllable. Using the defaults for the algorithm generates a plot where Melbourne LGAs completely dominate the rest of the state, and polygons get split. Here we used a very small number of iterations to get a very reasonable view of Melbourne while still getting clear views of the rural LGAs. This is almost accidental!

These approaches fundamentally differ from focus+context methods. Cartograms substitute spatial accuracy for data encoding, often severely disrupting shapes and adjacencies. The reader can still extract geographic features and relationships, as the cartogram preserves some relative positions and topology. In contrast, the FGC fisheye transformation preserves relative positions and topology while magnifying a user-selected spatial region rather than a data-driven variable. The use cases are distinct: cartograms address the dominance of a variable in space, whereas fisheye lenses facilitate exploration of local detail within a broader geographic context.

The FGC 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 applications such as examining hospital networks in Melbourne while maintaining the tsate context, exploring Census tracts in a metro core without losing county boundaries, or studying transit lines with their regional hinterland visible.

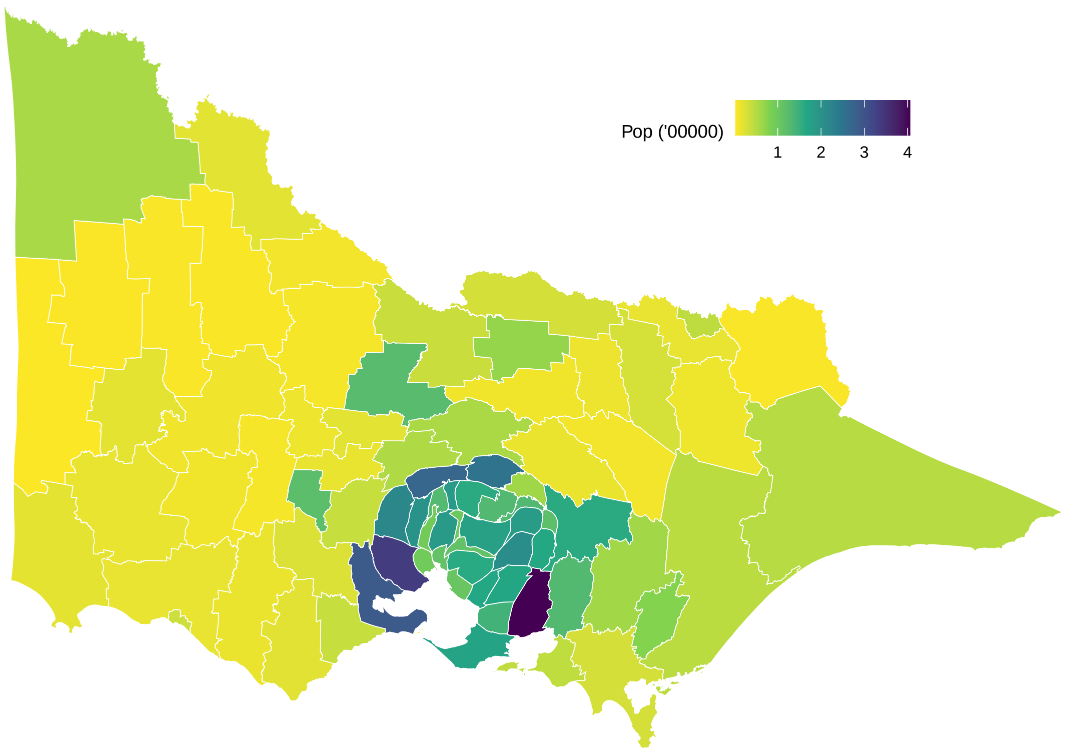


Figure 3: Population of Victorian LGAs shown as a diffusion cartogram. Color (and size) indicates population. We can see that the LGAs for greater Melbourne have the highest population, and these are massively exploded.

3 Usage

The primary functionality is from two functions: `fisheye_fgc` which operates on a `data.frame` or `tibble`, and `sf_fisheye` which operates on an `sf` object. Operationally, both are the same, except that the elements in the object being transformed are different: The first two columns in the former, and the geographic coordinates in the latter.

Both also require the 'center' of the fisheye transformation to be specified using `cx` and `cy`. By default, these are both 0, which is unlikely to work for most spatial problems. The code snippet below transforms a subset of hospitals in metropolitan Victoria, using data from the `mapycusmaximus` package, where `r_in` and `r_out` specify the radius of the transformed circles, and `zoom_factor` which controls the appearance of the inner-most focus layer.

```
filtered_locations <-  mapycusmaximus::hospital_locations |>
  filter(longitude >= 144,
         longitude <= 146,
         latitude <= -37,
         latitude >= -39)

hospitals_transformed <- fisheye_fgc(filtered_locations[, -1],
                                     cy = 145, cx = -37.8,
                                     r_in = 0.4, r_out = 0.7,
                                     zoom_factor = 1.8)


hospitals_transformed |>
  head()

#>           x_new    y_new
#> [1,] -37.86326 144.9478
#> [2,] -37.19449 145.7153
```

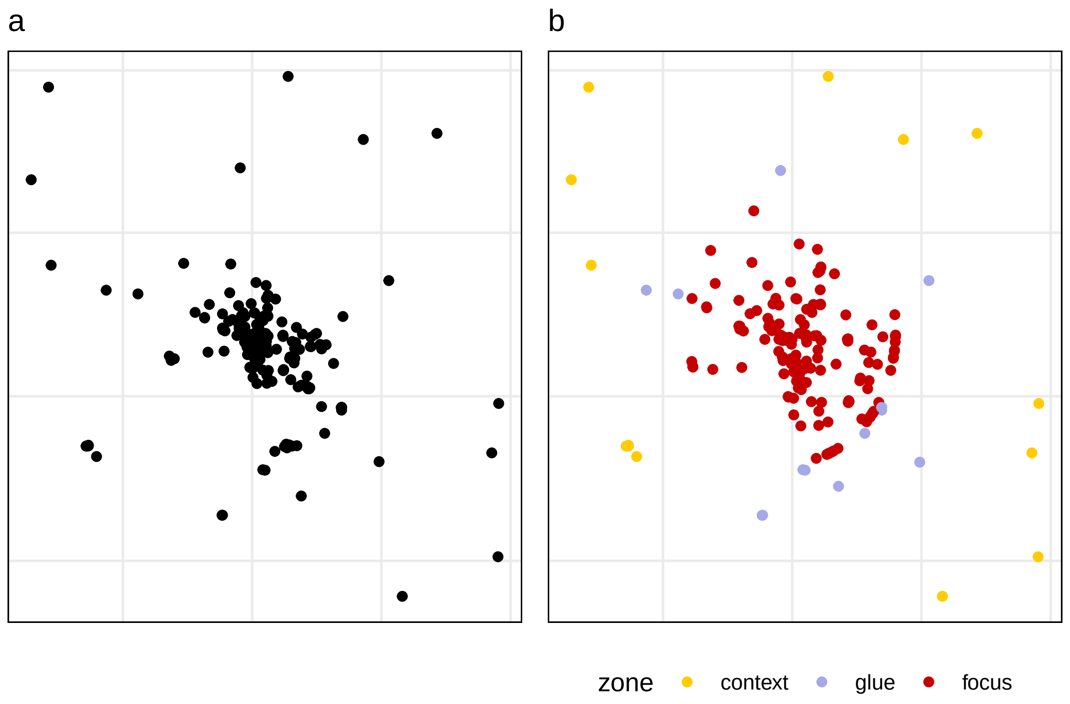


Figure 4: Original (a) and fisheye transformed (b) points representing the geographic location of a subset of hospitals in metropolitan Victoria. In (b), points are colored by whether they are part of the context, glue, or focus layer. The transformed data allows for the individual points near the center to be seen more easily.

```
#> [3,] -37.88071 144.9621
#> [4,] -37.92105 145.3812
#> [5,] -37.72008 145.1054
#> [6,] -37.71986 145.0819
```

The effect of this transformation can be seen in Figure 4. Points near the centre can be seen to be more spread in the transformed data (b). The points are colored based on whether they are part of the context, glue, or focus subsets of the data. Points in the focus are pushed outwards, but points in the context section remain in the original positions. Points in the glue may or may not move to smoothly adjust from the focus to context, depending on the choice of parameter. Here they remain in the original position.

Figure 5 shows the transformation of the spatial coordinates, for the data used in Figures 1, 2 and 3. The result is similar to the cartogram, but more controllable. The polygons are continuous, and boundaries are preserved. Although it may be difficult to see this directly from this plot, it is demonstrated by the coloring by population showing that features such as shading polygons will work as expected.

4 Implementation

Before detailing the algorithm behind the fisheye transforming, we need to establish a baseline notation.

Denote a point in the original (i.e. non-transformed) space to be $P = (x, y)$. The transformation operates around a chosen center $C = (c_x, c_y)$. For the sake of notational simplicity, we will assume $C = (0, 0)$, however the mathematics that follows is true for any C, simply by defining P to be $P + C$.

We will also require 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.

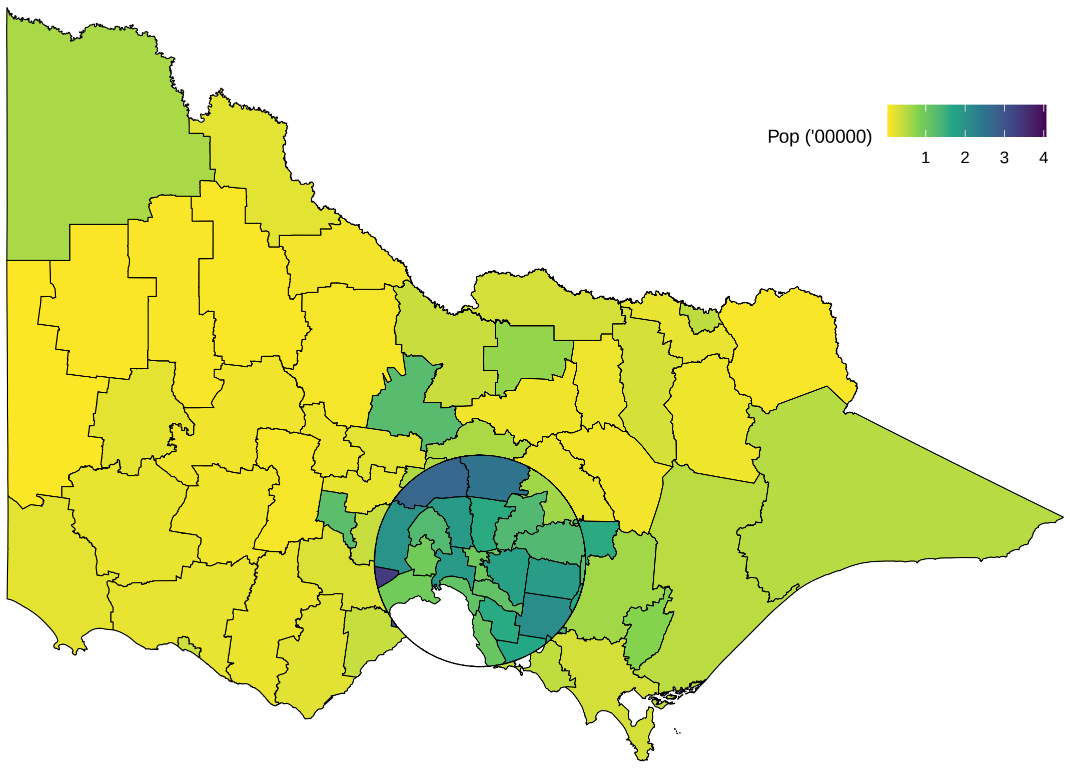


Figure 5: Fisheye transformation applied to the map of Victoria, Australia. The center is focussed on the Melbourne metropolitan region, which cannot typically be seen in standard spatial maps.

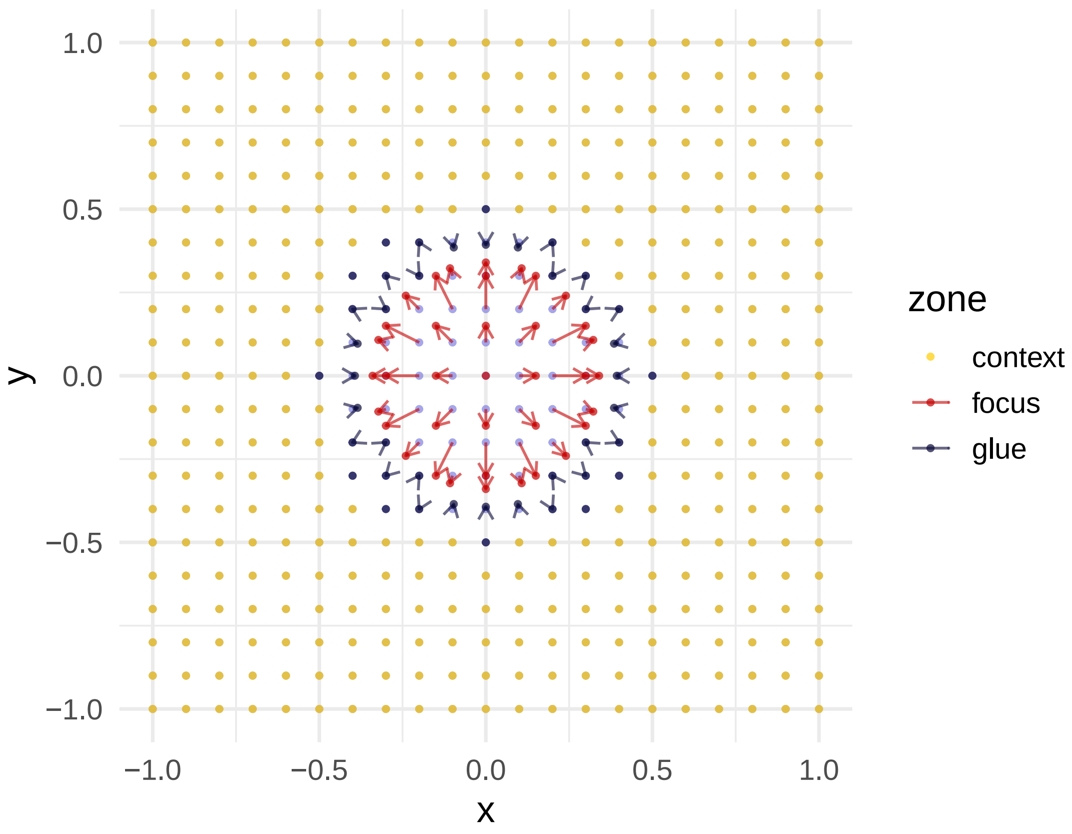


Figure 6: Illustration of Focus-Glue-Context zones in a fisheye transformation. Original grid points are shown alongside their transformed positions, colored by zone, with arrows indicating displacement. Points expand in the focus, compress smoothly (inwards or outwards) in the glue, and remain fixed in the context.

Let (r, θ) denote the polar form of the point P. The transformation defines a new radius r' via a piecewise function:

$$r' = \begin{cases} \min\left(z \cdot r, r_{\text{in}}\right) & \text{if } r \leq r_{\text{in}}, \\ r_{\text{in}} + (r_{\text{out}} - r_{\text{in}}) \cdot h(u; s) & \text{if } r_{\text{in}} < r \leq r_{\text{out}}, \\ r & \text{if } r > r_{\text{out}}, \end{cases} \tag{1}$$

where $z \geq 1$ is the zoom factor within the focus, $s \in (0, 1]$ controls glue compression, and $u = (r - r_{\text{in}})/(r_{\text{out}} - r_{\text{in}})$ normalizes 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 function,

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

which compresses radii near both boundaries and emphasizes the mid-glue region. A visual representation of the function h is shown in Figure 7.

There are two additional options for the transformation. The first biases points towards the outer radius (as opposed to the inner radius).

The second option can introduce 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.

A demonstration of the effect of the transformation on an equal-spaced grid can be seen in Figure 8. Points in the focus region are pushed outwards, those in the glue region are moved less to preserve topographic stability, and those in the context section are not moved at all.

4.1 Integration with sf

We expect that the main use case for this software is in the use of spatial datasets. Spatial datasets vary widely in CRS, extent, feature types, and schema. mapycusmaximus follows a disciplined staged workflow where each step is explicit, auditable, and invariant to input type. The architecture separates numeric mapping, spatial orchestration, and geometry reconstruction, allowing the core transform to remain small and testable while sf-specific concerns are isolated in thin wrappers.

The pipeline broadly follows a process of **normalize** -> **warp** -> **denormalize**. There are additional steps to deal with intricacies of the R `sf` object, such as maintaining the CRS after manipulation, but here we focus only on details relevant to the transformation itself.

The first step in the pipeline is to normalise the spatial data. We choose to normalise, as from a visual perspective it may be easier for analyst to choose a center to be "roughly two thirds across" their data, rather than trying to pick out a specific latitude and longitude. Users can choose whether to keep a fixed aspect ratio (default, `preserve_aspect = TRUE`) or scale each axis independently. In the normalised coordinates framework, we assume the center is at $(0, 0)$, and the spatial dataset is bounded between -1 and 1.

Fundamentally, spatial objects in the `sf` package are just lists of points, with a structure of how they are layered and connected together. Thus, the `sf_fisheye` effectively loops over these layers, each time applying a transformation function to each layer, and then reconstructing the `sf` object based on it's original inputs. Geometries are split by type:

- **POINT**: direct transformation
- **LINESTRING**: transform each vertex, retain order
- **POLYGON**: process each ring independently, each ring becomes a linestring to transform

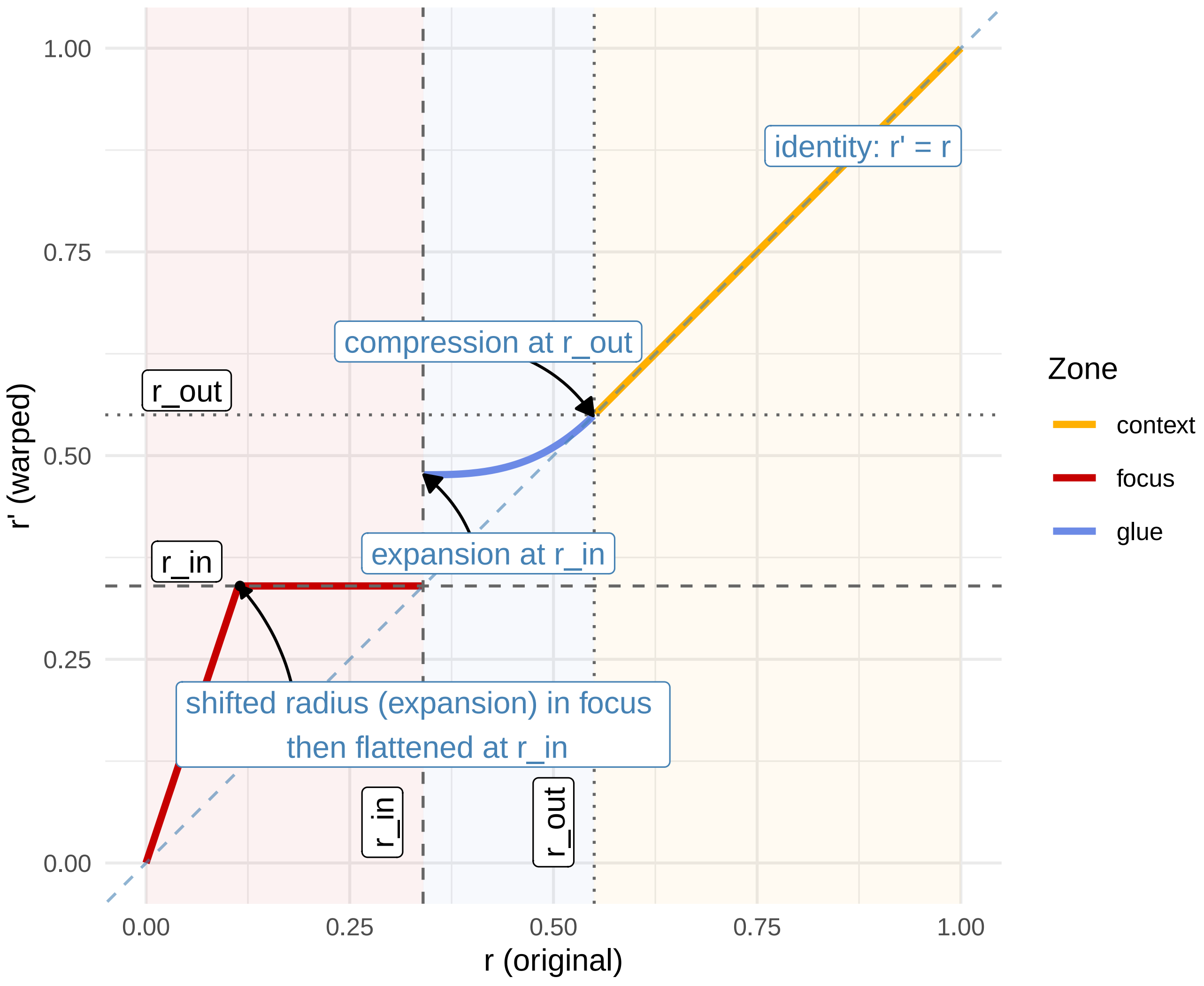


Figure 7: Radial mapping function of the FGC fisheye. The plot shows original radius r against warped radius r′, with shaded focus, glue, and context regions and a reference identity line. The curve demonstrates expansion in the focus, smooth compression in the glue, and identity mapping outside.

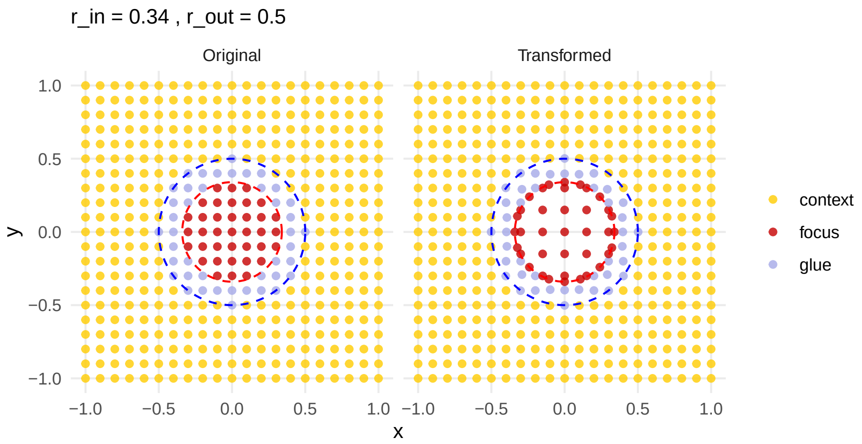


Figure 8: Basic numeric example of an FGC fisheye transformation. A synthetic grid is shown before and after warping. The example isolates the core radial mapping independent of spatial geometry reconstruction.

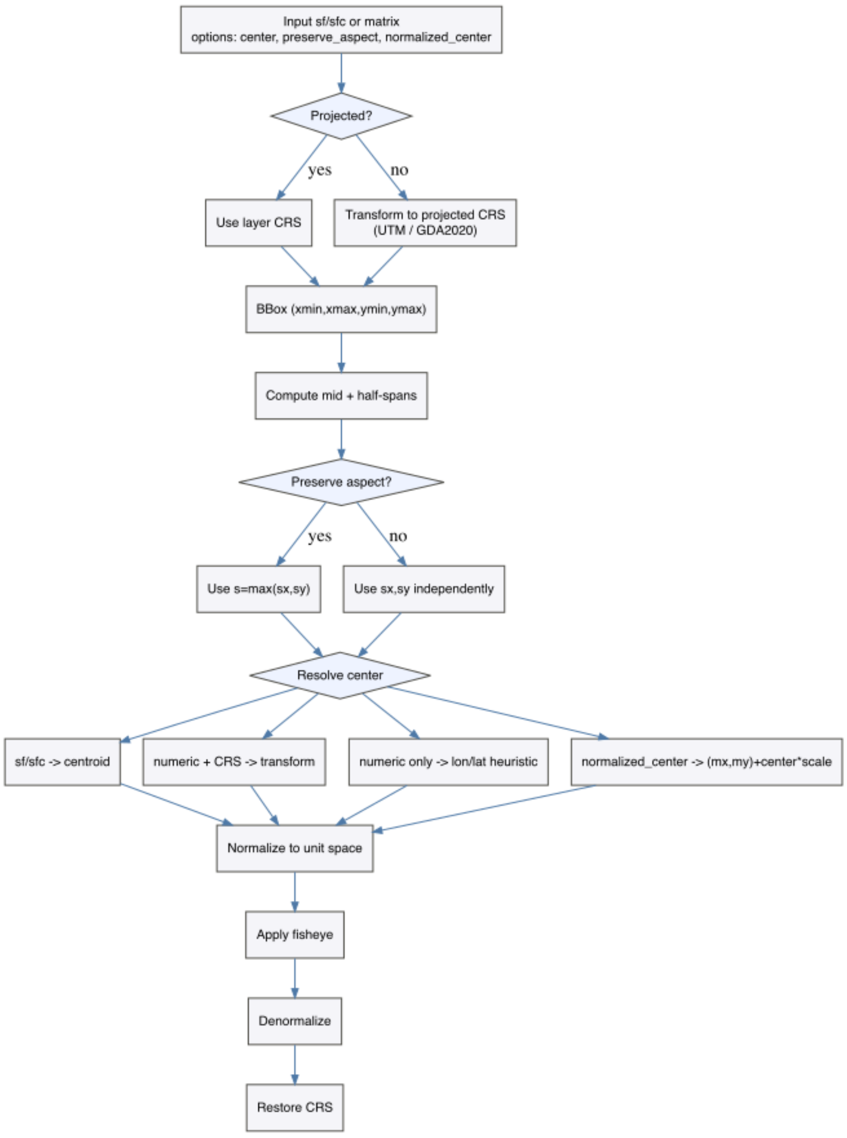


Figure 9: Workflow diagram of the normalization and CRS handling pipeline. The flowchart depicts CRS selection, normalization, center resolution, fisheye application, and CRS restoration. This highlights the staged design ensuring projection awareness and parameter stability.

- **MULTIPOLYGON**: process each polygon separately

After transformation, polygon rings are explicitly closed by forcing first and last vertices to equality. This prevents numerical drift when the warp changes ring curvature. Geometries are rebuilt using sf constructors (`st_point()`, `st_linestring()`, `st_polygon()`, `st_multipolygon()`), combined into an sfc with original CRS, and spliced back into an sf if appropriate. Attributes of the spatial object are preserved because only the geometry column is replaced.

4.2 Parameters

Radial warping. The radii r_{in} and r_{out} define the focus and glue boundaries in the normalized coordinate space and must satisfy $r_{\text{in}} < r_{\text{out}}$. 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_{in}, s in map units; with per-axis scaling, the corresponding shape is an axis-aligned ellipse with semi-axes r_{in}, s_x and r_{in}, s_y. Inside the focus, distances from the center 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 normalized 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 invariant. The following constraints and behaviors 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 center 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 normalized center 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.

4.3 Controlling the transformation

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.

Quick start (synthetic grid). Set r_{in} to 0.30-0.35 and r_{out} to 0.55-0.70. Pair this with

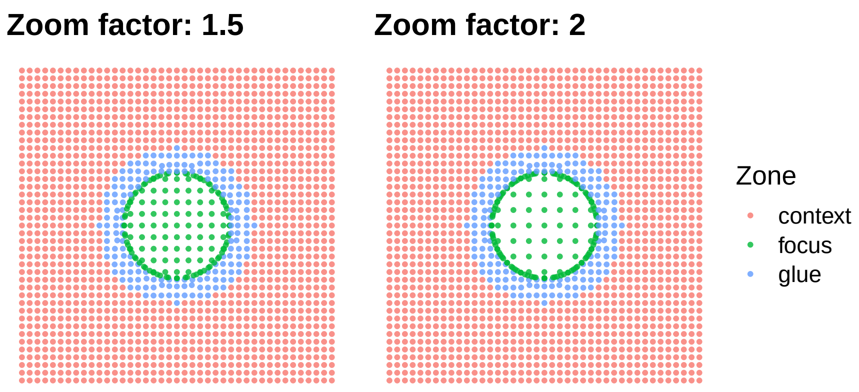


Figure 10: Effect of zoom factor on fisheye distortion. Two panels compare zoom factors 1.5 and 2 applied to a synthetic grid, with points colored by zone. Higher zoom increases magnification in the focus while preserving context stability.

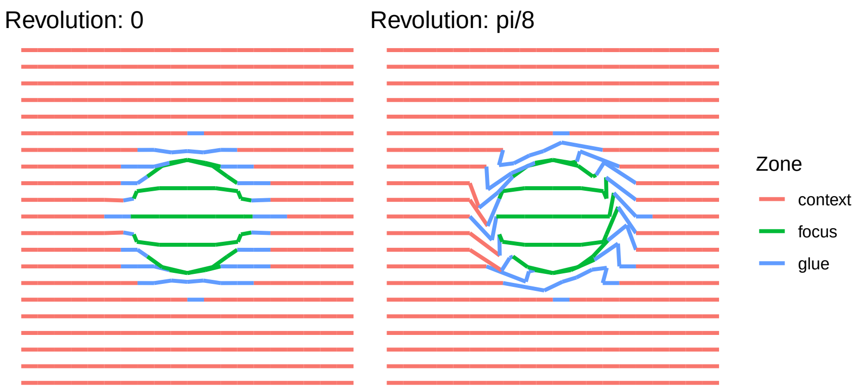


Figure 11: Effect of angular revolution on line geometries. Line paths are shown with revolution set to 0 and pi/8, colored by zone. Introducing revolution produces a visible rotational flow in the glue region without affecting the focus or context radii.

`zoom_factor` between 5 and 10 and `squeeze_factor` near 0.35 for a balanced focus that still shows context. Stick with `method = "expand"` and `revolution = 0` unless you explicitly need outer rigidity or a twist.

In this simple example, we demo the effect of zoom factor 1.5 and 2 on a synthetic grid to demonstrate how the point movement was effected by the zoom factor.

However, as demonstrated below, the revolution effect might make the distortion of the linestring object more obvious, comparing to just only the zoom factor effect.

As we can see, the revolution effect create a vortex or zoom wheel effect into the focus zone, which may be useful in some cases. 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.

Simple tweaks. If linework kinks, raise `squeeze_factor` slightly (for example: 0.45). If the focus feels too tight, lower `zoom_factor` toward 4-6. For reproducible comparisons, keep `normalized_center = TRUE` and reuse the same radii across runs.

5 Exploring the transformation parameter choices interactively

To support interactive exploration of Focus-Glue-Context (FGC) parameters, the package includes a *lens explorer* implemented in a Shiny app. It allows users to experiment with fisheye settings on a realistic spatial example - Victorian LGAs with a synthetic sampled

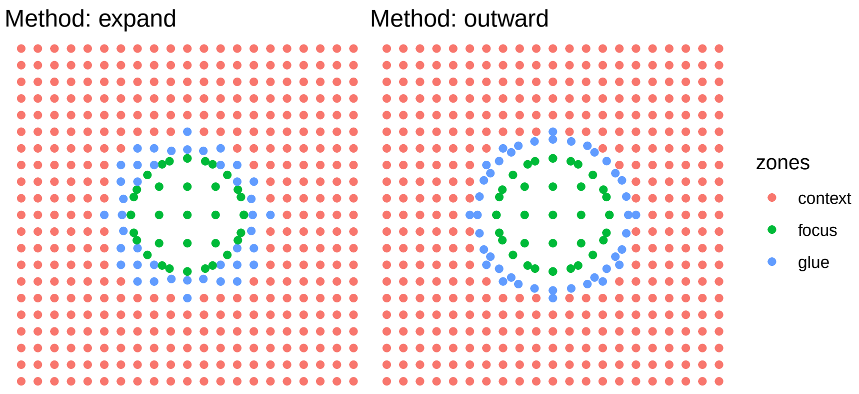
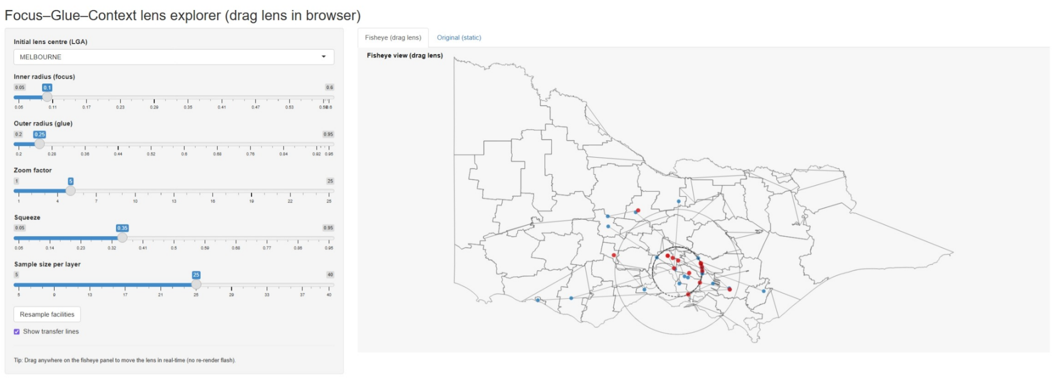


Figure 12: Comparison of glue compression methods. Polygon grids are shown under expand and outward glue modes. The outward method concentrates distortion closer to the focus, while expand distributes compression symmetrically across the glue.


Focus–Glue–Context lens explorer (drag lens in browser)

Initial lens centre (LGA): MELBOURNE

Inner radius (focus): 0.1 (0.05–0.6)

Outer radius (glue): 0.25 (0.2–0.95)

Zoom factor: 5 (1–25)

Squeeze: 0.35 (0.05–0.95)

Sample size per layer: 25 (5–40)

Resample facilities

[✓] Show transfer lines

Tip: Drag anywhere on the fisheye panel to move the lens in real-time (no re-render flash).

Figure 13: Interactive Focus-Glue-Context Shiny application. Users control lens centre, radii, zoom, and compression via sliders and drag the lens directly on the map, with points, lines, and polygons warped together in real time. The app enables rapid exploration of fisheye parameters before committing to static figures.

hospital - Residential Aged Care Facility (RACF) transfer network during COVID 19 - and to observe how points, lines, and polygons respond under a shared geometric warp.

5.1 App structure

The interface is divided into two coordinated panels:

- **Fisheye (drag lens)**: an SVG-based view rendered in the browser. The fisheye center can be repositioned by dragging directly on the map, triggering real-time geometric warping without re-running server-side plotting code.
- **Original (static)**: a conventional `ggplot2` and `sf` map rendered with the *same bounding box and aspect ratio* as the fisheye view, enabling fair visual comparison between warped and unwarped representations.

Spatial data (LGAs, points, and transfer lines) are converted once into plain coordinate lists and sent to the browser. Subsequent interactions update only lens parameters, ensuring smooth response even during continuous dragging.

5.2 Interactive workflow

Selecting the lens center

Users begin by choosing an **Initial lens center (LGA)** from the sidebar. Internally, the selected LGA polygon is reduced to a representative point (`st_point_on_surface()`), which becomes the fisheye center. This mirrors the common scripted workflow of passing a polygon or centroid as the `center` argument to `sf_fisheye()`.

Once initialized, the center can be moved freely by dragging within the fisheye panel, allowing rapid scanning of different regions without changing parameters.

Adjusting focus and glue radii

Two sliders control the spatial extent of distortion:

- **Inner radius (focus)** -> `r_in` Sets the size of the magnified region. Smaller values create a tight focal bubble; larger values expand the magnified area.
- **Outer radius (glue)** -> `r_out` Sets the extent of the transition zone where compression occurs before geometry becomes fixed.

Together, these define the Focus-Glue-Context structure used by both `fisheye_fgc()` and `sf_fisheye()`. Increasing `r_out` spreads distortion more gradually; decreasing it concentrates deformation closer to the focus.

Controlling magnification and compression

Two further sliders adjust distortion strength:

- **Zoom factor** -> `zoom_factor` or `zoom` Controls radial expansion inside the focus. Higher values increase separation of dense features but amplify distortion near the focus boundary.
- **Squeeze** -> `squeeze_factor` or `squeeze` Controls how sharply distances compress within the glue region. Smaller values produce a pronounced 'shoulder' near boundaries; larger values yield a smoother transition.

Users are encouraged to increase zoom until local structure becomes readable, then adjust squeeze to reduce crowding or sharp curvature near the glue boundary.

Sampling and layer visibility

The network is intentionally subsampled to maintain interpretability:

- **Sample size per layer** (`n_fac`) controls how many hospitals and RACFs are included.
- **Resample facilities** draws a new random subset to test robustness of visual conclusions.
- **Show transfer lines** toggles line geometry on and off, allowing users to tune parameters using points alone before validating connectivity.

All layers (polygons, points, and lines) are warped together using identical parameters, ensuring alignment is preserved under distortion.

5.3 What to look for

When using the app, readers should pay attention to:

- Whether dense metropolitan features separate cleanly in the focus.

- Whether transfer lines remain connected to their endpoints under distortion.
- How stable the surrounding context remains as the lens moves or parameters change.

These observations help users choose sensible parameter ranges before producing static figures or scripted analyses.

5.4 Design rationale

The app deliberately separates *parameter exploration* from *final figure generation*. Interactive dragging and sliders provide immediate visual feedback, while the parameter values correspond directly to arguments in `sf_fisheye()`. Once suitable settings are identified, the same values can be reused verbatim in reproducible code pipelines.

5.5 Future work

For the current approach, the shiny app only use the default dataset included in the `mapycusmaximus` package, which are the 2016 Victorian Local Government Areas (LGA) and their boundaries, accompany with the synthetic transfer network between hospital and RACF. In the future stable, we will open up the app for users to upload their own spatial data or piping it directly to the Shiny app from R.

6 Application to Victorian ambulance transfer records

To illustrate the use we use data summarising transfers between residential aged care facilities (RACFs) and Victorian hospitals in Victoria, over a period that includes the COVID-19 pandemic period. This data is provided with the package in the data object `conn_fish`. The transfer data is synthetic, for privacy reasons, but locations of the RACFs and hospitals is publicly available, and accurate.

To simplify the illustration we use a sample 10 hospitals and 10 RACFs. Most of the hospitals are located in the Melbourne metropolitan area, and zooming in on this helps to helps to see the transfer volume.

Figure 14 shows the geographical map of local government areas overlaid by locations of RACFs (blue) and hospitals (red) overlaid with transfers shown as lines. The rural transfers are easy to see, but transfers in the metropolitan area cannot be seen on this map. The fisheye transformation can be used to alleviate this. The plot is generated using the code below. Notice that the polygon data forms the background, on which points for the ACFs and the hospitals, and the connections between points for the transfers. The plot can be made interactive by directly passing to `ggplotly()` (Sievert, 2020).

```
vic_transfers_plot <- ggplot() +
  geom_sf(data = vic, fill = "grey85", color = "white") +
  geom_sf(data = conn_fish_sub,
          aes(size=weight, alpha=weight, label=weight),
          color = "black") +
  geom_sf(data = hosp_pts,
          aes(label = hosp_name),
          color = "#4DAF8F",
          shape = 16,
          alpha = 0.8,
          size = 1.5) +
  geom_sf(data = racf_pts,
          aes(label = racf_name),
          color = "#984EA3",
          shape = 4,
```

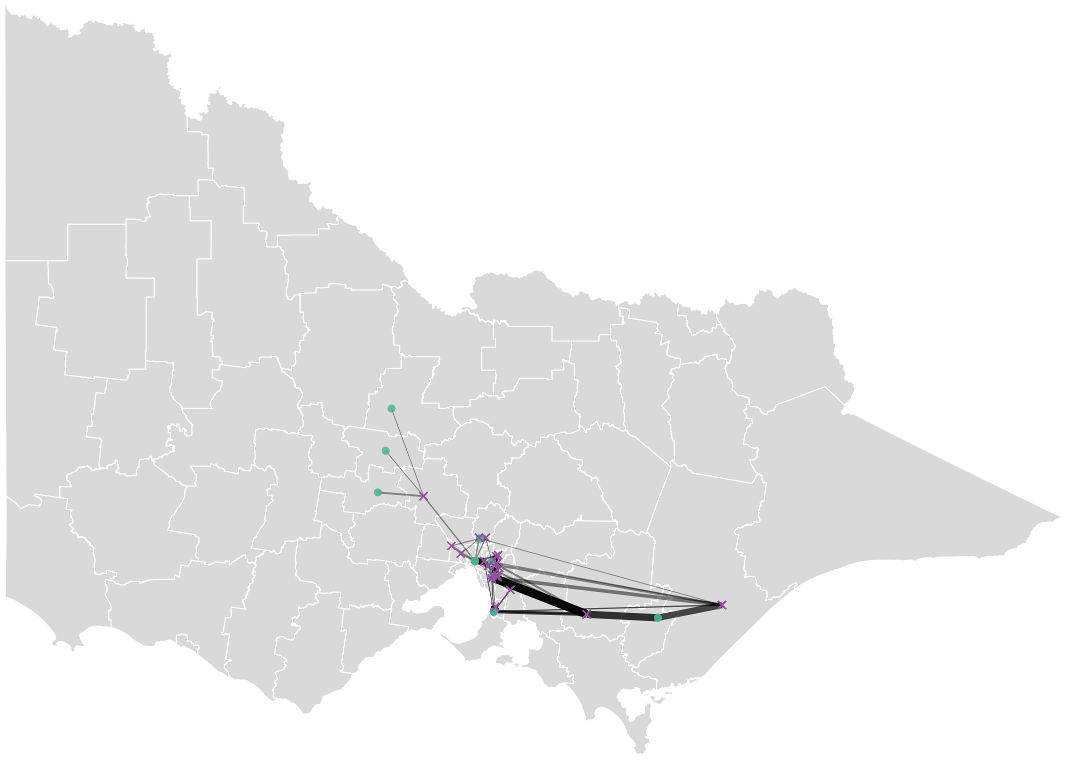


Figure 14: Victorian map overlaid by locations of RACFs (purple cross) and hospitals (green circle), and lines indicating transfers. Line thickness maps to number of transfers, with thicker indicating more. Transfers in the metropolitan region cannot be fully seen because the area is very condensed geographically.

```
          alpha = 0.8,
          size = 1.5) +
  scale_size(range = c(0.2, 2), guide = "none") +
  scale_alpha(range = c(0.4, 1), guide = "none") +
  coord_sf() +
  theme_map() +
  theme(panel.background = element_rect(fill = "white", color = NA))
```

The first step is to decide on a center for the transformation. In the code below, we choose a selection of the LGAs that are in the Melbourne's metropolitan area, but the user can simply input values but it needs to be an `sf` spatial point object.

```
# Select a sample of LGAs in the metropolitan region from which to calculate
# a centre to run the fisheye transformation
metro_names <- c("MELBOURNE", "PORT PHILLIP",
                 "STONNINGTON", "YARRA",
                 "MARIBYRNONG", "MOONEE VALLEY",
                 "BOROONDARA", "GLEN EIRA", "BAYSIDE")
metro_center <- st_union(vic[vic$lga_key %in% metro_names, ]) |>
  st_centroid()
```

The transformation can be applied to multiple layers, if all are in the same object. Thus, the next step is to combine the points, transfers and LGA data into one `sf` object. The following code does this.

```
# Add indicator variable for type of institution, preparing for the join
hosp_point2 <- hosp_pts |>
  mutate(type = "hospital") |>
  rename(id = destination)
racf_point2 <- racf_pts |>
  mutate(type = "racf") |>
  rename(id = source)

# Ensure that all are on the same projection
total_trans <- conn_fish_sub
hosp_point2 <- hosp_point2
racf_point2 <- racf_point2

# Join the different objects
all_points <- bind_rows(hosp_point2, racf_point2)
vic_all <- dplyr::bind_rows(
  vic |> dplyr::mutate(.layer="vic"),
  hosp_point2 |> dplyr::mutate(.layer="hosp_pt"),
  racf_point2 |> dplyr::mutate(.layer="racf_pt"),
  total_trans |> dplyr::mutate(.layer="transfers"))

# Transform all objects simultaneously
vic_all_w <- sf_fisheye(vic_all, center = metro_center,
                        r_in = 0.2, r_out = 0.3,
                        zoom = 4, squeeze = 0.35)

# Extract the layers for plotting control
vic_w   <- vic_all_w |> dplyr::filter(.layer == "vic") |>
  dplyr::select(-.layer)
hosp_w   <- vic_all_w |> dplyr::filter(.layer == "hosp_pt") |>
  dplyr::select(-.layer)
racf_w   <- vic_all_w |> dplyr::filter(.layer == "racf_pt") |>
  dplyr::select(-.layer)
transfers_w <- vic_all_w |> dplyr::filter(.layer == "transfers") |>
  dplyr::select(-.layer)
```

The plot of the transformed data (Figure 15) is constructed in the same fashion as the geographic map, using the code below.

```
fgc_transfers_plot <- ggplot() +
  geom_sf(data = vic_w, fill = "grey85", color = "white") +
  geom_sf(data = transfers_w,
          aes(size=weight, alpha=weight, label=weight),
          color = "black") +
  geom_sf(data = hosp_w,
          aes(label = hosp_name),
          color = "#4DAF8F",
          shape = 16,
          alpha = 0.8,
          size = 1.5) +
  geom_sf(data = racf_w,
          aes(label = racf_name),
          color = "#984EA3",
          shape = 4,
          alpha = 0.8,
          size = 1.5) +
  scale_size(range = c(0.2, 2), guide = "none") +
  scale_alpha(range = c(0.4, 1), guide = "none") +
  coord_sf() +
```

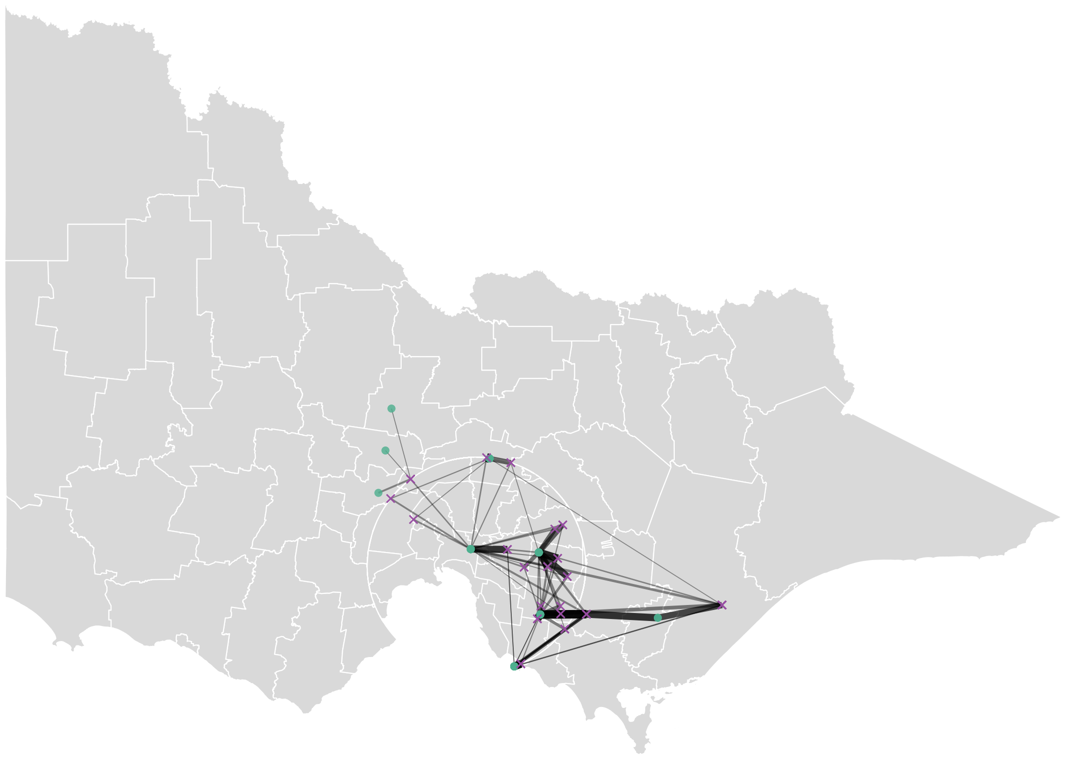


Figure 15: Fisheye magnification of Greater Melbourne with statewide context, allows the transfers within the city region to be more visible. Locations of RACFs (purple cross) and hospitals (green circle), and lines indicating transfers. Line thickness maps to number of transfers, with thicker indicating more.

```
theme_map() +
theme(legend.position = "none",
      panel.background = element_rect(fill = "white", color = NA))
```

The fisheye clarifies dense metro structure without losing state context. Hospitals and RACFs that previously overlapped separate cleanly; transfer lines remain connected to their endpoints because every layer uses the same center and radii. Outside the glue zone, geometry remains stable, so readers can still place Melbourne within Victoria. It can take some adjusting of parameters to get an ideal view, here we have used `zoom = 4` which provides just enough zoom inside the focus layer to spread out the points, and `squeeze = 0.35` to minimize the glue zone. Other parameters such as `center` can shift the focus, e.g. another metroplition region like Geelong, or the `r_in` and `r_out` to set the size of the focus and glue.

We can learn several interesting details about transfers from this transformed view:

- Most transfers occur between an ACF and the closest hospital, such as racf108 and the Northern Hospital, racf506 and St Vincent's Hospital, racf751 and Latrobe Regional Hospital (which can be easily seen with the interactive plot in the html format of the paper).
- Some transfers are occuring to distant hospitals. This would make sense if the medical condition only was treatable at a particular hospital. But the pattern appears more irregular than this, and could be due to congestion at particular hospitals or other reasons. For example, racf152 regularly sends patients to the Northern Hospital or St Vincent's Hospital instead of the closest hospital, Central Highlands Rural Health.

This data is synthetic, so these patterns do not indicate actual operations.

7 Discussion

The mapycusmaximus package provides an sf-native implementation of the FGC fisheye that is projection-aware, parameterized in normalized units, and safe across points, lines, and polygons. The package separates radial mapping from geometry orchestration, exposes explicit controls over focus, glue, and context, and preserves attributes and CRS invariant for reproducible pipelines with ggplot2.

Unlike cartograms (thematic distortion), hex/regular tile maps (discrete abstraction), or inset/multi-panel layouts (spatial separation), the FGC lens delivers continuous magnification within a single map while preserving topology and bearings. This reduces cognitive load for readers who must relate local phenomena to their broader geography.

The fisheye introduces non-metric distortion in the focus and glue; therefore, use it for visual exploration and communication, not for metric analysis. Aggressive `zoom` or `squeeze` can impair legibility near the glue boundary; conservative defaults and `revolution = 0` are recommended for publication maps. When comparing multiple regions, prefer `normalized_center = TRUE` with fixed radii to ensure visual comparability. At present, exact matching of focus and glue radii across separately transformed layers may require a manual step (the user have to manually merge the two or more layers, perform the fisheye transformation, then separated the transformed layers).

Planned extensions include anisotropic or elliptical profiles, multi-focus blending, first-class raster support via warped grids and resampling, and interactive focus selection for exploratory analysis. We also plan an API for shared normalization and radius locking across layers (for example: a `combine_fisheye`) so that multiple layers can be warped with identical scale and then returned transformed. Performance improvements via vectorised geometry walkers or GPU acceleration would benefit dense polygonal datasets. Clear figure captions and scale disclaimers remain essential to communicate the presence and intent of distortion.

8 Conclusion

FGC fisheye transformations offer a concise, CRS-aware way to emphasize local structure without losing geographic context. By starting from a point-wise radial map and integrating carefully with sf for geometry reconstruction, the approach keeps figures continuous and overlays aligned. The examples demonstrate clearer narratives for metropolitan focus while maintaining state or nation-level context.

9 AI use declaration

We used AI tools to assist with code refactoring and drafting portions of the text. All methods, parameter settings, and claims were designed and reviewed by the authors, and we verified outputs with the package's test suite and example renders.

10 Resources

The github repo for this paper is `https://github.com/Alex-Nguyen-VN/paper-mapycusmaximus`.

The mapycusmaximus package is available at `https://github.com/Alex-Nguyen-VN/mapycusmaximus`.

Packages used in this work are XXX.

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. [p3]

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]

S. Kobakian, D. Cook, and E. Duncan. A hexagon tile map algorithm for displaying spatial data. *The R Journal*, 15:6–16, 2023. ISSN 2073-4859. doi: 10.32614/RJ-2023-021. https://doi.org/10.32614/RJ-2023-021. [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]

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. [p1]

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]

C. Sievert. *Interactive Web-Based Data Visualization with R, plotly, and shiny*. Chapman and Hall/CRC, 2020. ISBN 9781138331457. URL https://plotly-r.com. [p14]

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

C. O. Wilke. *cowplot: Streamlined Plot Theme and Plot Annotations for 'ggplot2'*, 2025. URL https://wilkelab.org/cowplot/. R package version 1.2.0. [p2]

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