`gridmappr`: an R package for creating small multiple gridmap layouts

Roger Beecham*[1,2]

[1]School of Geography, University of Leeds
[2]Leeds Institute for Data Analytics, University of Leeds

GISRUK 2024

Summary

We present `gridmappr`, an R package that automates the process of generating gridmaps – small multiple data graphics of regular size, laid out with an approximate geographic arrangement. Given a set of real geographic point locations, `gridmappr` allocates points to a regularly-sized grid of stated row-column dimensions. This allocation is constrained such that the distance between points in real and grid space is minimised and with a parameter that affects how compactly points are allocated to the regular grid. Layout examples are presented using different parameterisations and code is demonstrated for generating complex, information-rich gridmaps using standard `ggplot2`.

KEYWORDS: geovisualization, gridmaps, origin-destination, grammar of graphics.

1 Introduction

Gridmaps, sometimes called tilemaps, are maps where spatial units form grid cells of regular size, allocated into an approximate spatial arrangement. The advantage is that complex multivariate structure, rather than single values, can be depicted (Beecham and Slingsby, 2019; Beecham et al., 2020) and tested (Wood et al., 2011; Beecham et al., 2021) within geographic context. Many gridmaps are generated manually, the widely used LondonSquared layout of London boroughs (After the Flood, 2019) or those made available via the `geofacet` package (Hafen, 2024). For automatic allocation of spatial units to grid cells, various constraints might be considered, such as *compactness* and *alignment*, and the preservation of the original data, for example *distance*, *topology* and *shape* (Meulemans et al., 2017). `gridmappr` is an R implementation of Jo Wood's `Observable` notebook on *Grid map allocation* (Wood, 2021) and uses linear programming, and the `ompr` R package, for handling constraints and deriving solutions. Geographic points are allocated to grid cells such that the total of squared distances between geographic and grid locations is minimised. Each point is allocated to one grid cell only and a cell in the grid can contain no more than one geographic

*r.j.beecham@leeds.ac.uk

point. A *compactness* parameter determines whether points should be positioned from the centre or edges of the grid, and fixed spacers – reserved cells that cannot be allocated points – can be introduced to preserve breaks between non-contiguous spatial units. The package is documented at `github.com/rogerbeecham/gridmappr/`.

2 Generating layouts with gridmappr

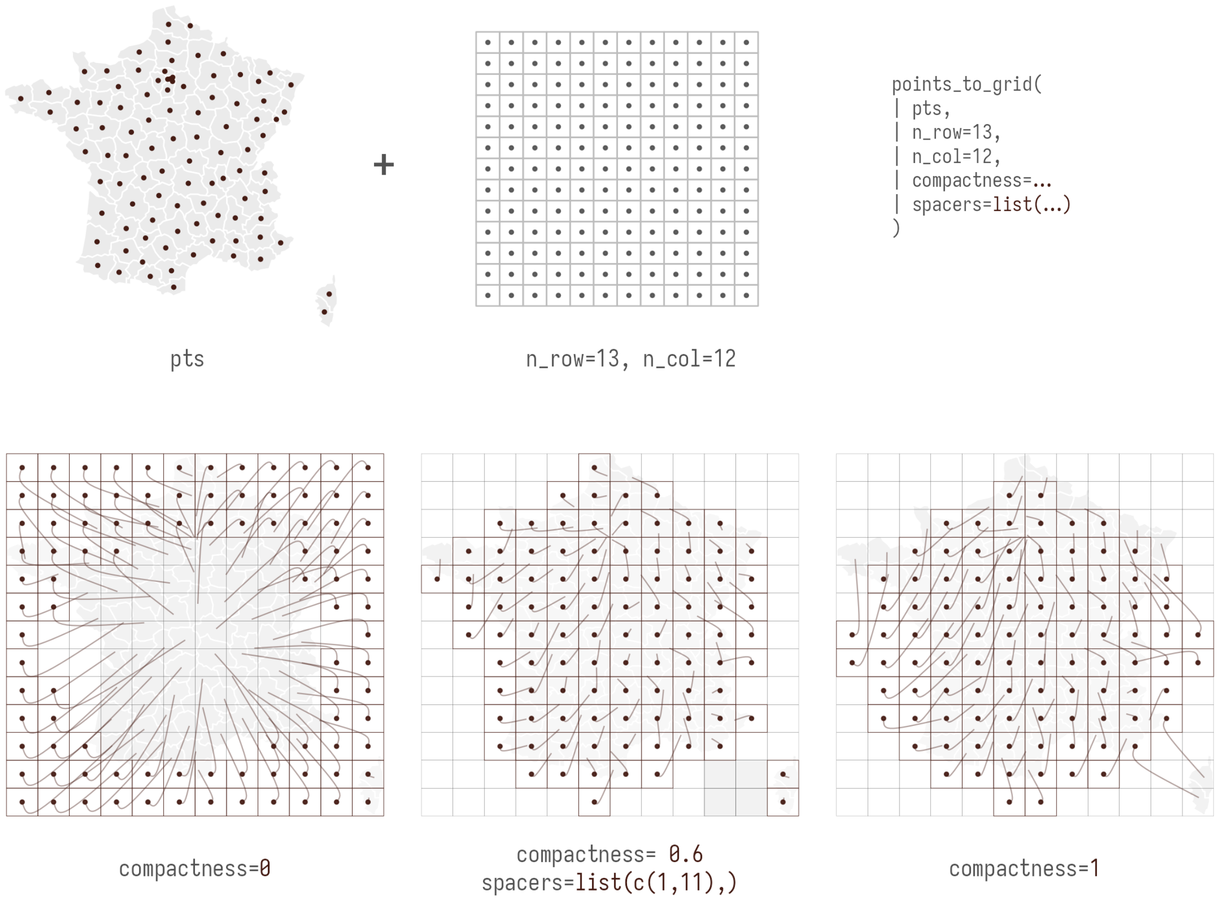


Figure 1: Candidate gridmap layouts for France's 96 departments.

The main allocation function in `gridmappr` is `points_to_grid()`. This will return grid cell positions (row and column identifiers) for a given set of geographic locations. It is parameterised with:

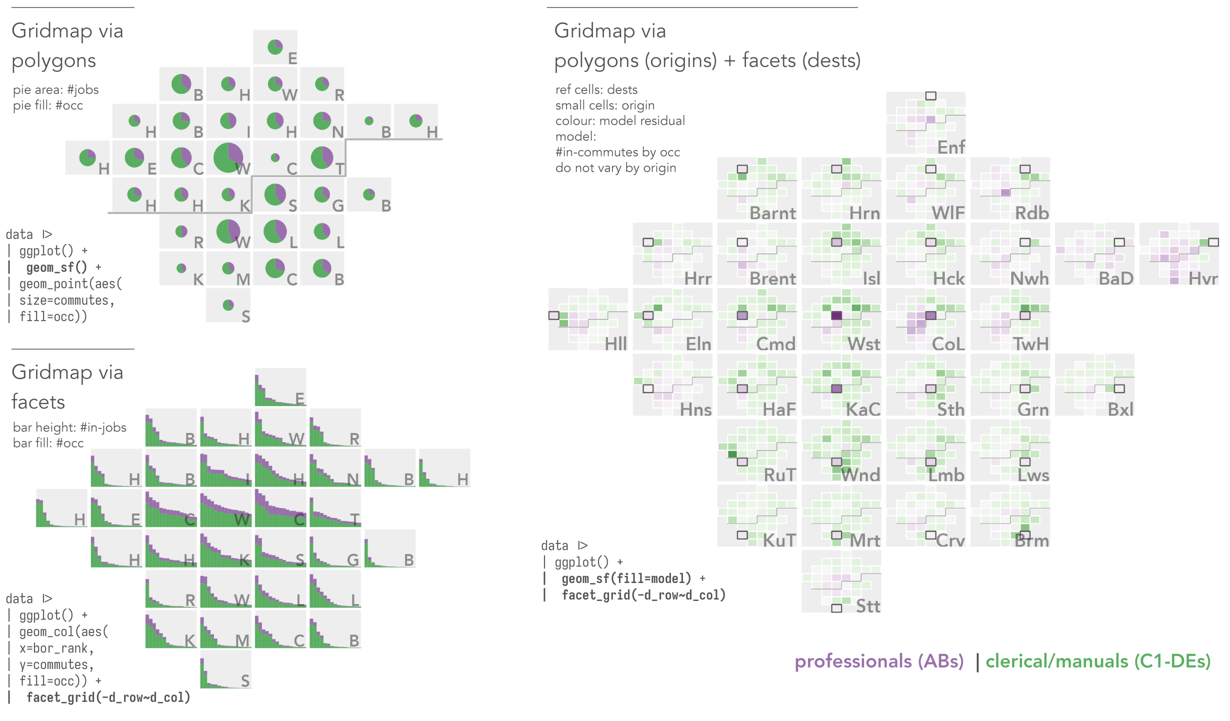
- `pts` A tibble of geographic points (x, y) to be allocated to a grid.
- `n_row` Maximum number of rows in grid.
- `n_col` Maximum number of columns in grid.
- `compactness` Optional parameter between 0 and 1 where 0 allocates towards edges, 0.5 pre-

serves scaled geographic location and 1 allocates towards centre of grid.

- `spacers` Optional list of grid cell locations defining grid location of fixed spacers which cannot be allocated points. Coordinates are in (row, column) order with the origin $(1, 1)$ in the bottom-left. The default is an empty list.

In Figure 1 candidate gridmap layout solutions are shown for France's 96 departments. In this figure a single grid is used with row-column dimensions 13×12. With larger dimensions, the layout would more closely approximate to real geography (assuming a compactness of c.0.5). The consequence of larger grids, however, is gaps and whitespace between spatial units and a reduction data density – the graphic space on which data can be encoded. The three layouts in the bottom row of Figure 1 are created left-to-right with compactness scores of 0, 0.6 and 1. Spacers are introduced in the middle layout to ensure that departments in Corsica are non-contiguous with mainland France. We also annotate layout candidates with displacement vectors. Code and helper functions for plotting these displacement vectors is demonstrated in the `package documentation`. Once a gridmap solution is arrived at, polygon files representing the grid and corresponding cell positions (row and column IDs) is generated with the function within `gridmappr` `make_grid()`.

3 Using `gridmappr` layouts for analysis



```
data |>
| ggplot() +
|  geom_sf() +
| geom_point(aes(
| size=commutes,
| fill=occ))
```

```
data |>
| ggplot() +
| geom_col(aes(
| x=bor_rank,
| y=commutes,
| fill=occ)) +
|  facet_grid(-d_row~d_col)
```

```
data |>
| ggplot() +
|  geom_sf(fill=model) +
|  facet_grid(-d_row~d_col)
```

Figure 2: `ggplot2` code for plotting gridmap layouts.

Figure 2 demonstrates how gridmaps can be drawn with standard `ggplot2`: from the gridmap polygon file directly – top left; by supplying grid cell positions to `facet_grid()` – bottom left; or combining both to effect a map-within-map layout and complex graphics such as OD maps (Wood et al., 2010; Beecham et al., 2023) – right.

The graphics present an analysis of travel-to-work data by London borough, as measured in 2021 Census; each grid cell is a London borough. In the first example, the number of jobs accessed in each London borough is encoded using circle area, with circles positioned at the centroids of gridmap cells and coloured according to high-level occupation class – AB professional and C1-DE clerical/manual occupations. The gridmap layout is drawn with `geom_sf()` and points at the centroids of boroughs parameterised using a variant of `geom_point()`. Next, full frequency distributions of in-commutes to London boroughs – that is, the full origin-destination dataset – are represented using column charts. We have chosen to exclude internal commutes – same origin-destination – and given each borough summary a local scaling. Column charts cannot be straightforwardly spatially arranged in `ggplot2` since they must be parameterised with an x, y position within the data space. A spatial arrangement is instead effected using faceting – row and column IDs for each borough within the gridmap are supplied to `facet_grid()`. Finally the full origin-destination (OD) data (33^2 borough-borough commute pairs) are encoded via a map-within-map layout. The large reference cells are again destinations laid out using `facet_grid()`; the smaller cells are commute origins drawn directly with `geom_sf()` from the gridmap polygon file. Here, we have generated a model that compares the relative number of commutes made by ABs versus C1-DE clerical/manual for any OD pair. Our model depends on commute destination. That is, since the relative number of ABs and C1-DE clerical/manual accessed in boroughs varies – 48% in Richmond upon Thames (RuT) are ABs, whereas in Barking and Dagenham (BaD) that figure is 18%, our model expects a randomly sampled OD pair into- RuT to contain many more ABs than a randomly sampled OD pair into- BaD. The colours in the graphic are model residuals – either more ABs or C1-DEs than expected, and these residuals also emphasise differences from the model that are large in absolute and relative terms (c.f. Beecham and Lovelace, 2023).

4 Conclusion

`gridmappr` automates the process of generating gridmaps – maps consisting of grid cells of regular size allocated with an approximate spatial arrangement. We demonstrate how different gridmap layouts can be generated using `gridmappr`'s helper functions and how gridmap data graphics can be drawn with standard `ggplot2` calls. The package currently implements two constraints: *distance* displacement and *compactness*, with the facility to define *spacers* used, for example, to separate spatial units that are non-contiguous. This is handled via Linear Programming and using the `ompr` R package. Future releases of the package will provide options for further constraining layouts using *topology* and *shape* (Meulemans et al., 2017), as well as explore more novel hybrid *data-spatial* layouts (e.g. van Beusekom et al., 2023).

5 Biography

Roger Beecham is Associate Professor in Visual Data Science at School of Geography, University of Leeds.

References

After the Flood (2019). *LondonSquared* `https://github.com/aftertheflood/londonsquared`.

Beecham, R., Dykes, J., Hama, L., & Lomax, N. (2021). On the use of 'glyphmaps' for analysing Covid-19 reported cases. *ISPRS International Journal of Geo-Information*, *10*(4).

Beecham, R. & Lovelace, R. (2023). A framework for inserting visually-supported inferences into geographical analysis workflow: application to road safety research. *55*(3), 345–366, `https://doi.org/10.1111/gean.12338`.

Beecham, R. & Slingsby, A. (2019). Characterising labour market self-containment in London with geographically arranged small multiples. *Environment and Planning A: Economy and Space*, *51*(6), 1217–1224, `https://doi.org/10.1177/0308518X19850580`.

Beecham, R., Williams, N., & Comber, L. (2020). Regionally-structured explanations behind area-level populism: an update to recent ecological analyses. *PLoS ONE*, *15*(3), e0229974, `https://doi.org/10.1371/journal.pone.0229974`.

Beecham, R., Yang, Y., Tait, C., & Lovelace, R. (2023). Connected bikeability in london: which localities are better connected by bike and does this matter? *0*(0), 23998083231165122, `https://doi.org/10.1177/23998083231165122`.

Hafen, R. (2024). *geofacet: 'ggplot2' Faceting Utilities for Geographical Data.* R package version 0.2.1, https://hafen.github.io/geofacet/ `https://github.com/hafen/geofacet`.

Meulemans, W., Dykes, J., Slingsby, A., Turkay, C., & Wood, J. (2017). Small Multiples with Gaps. *IEEE Transctions on Visualization and Computer Graphics*, *23*(1), 381–390.

van Beusekom, N., Meulemans, W., Speckmann, B., & Wood, J. (2023). Data-Spatial Layouts for Grid Maps. In R. Beecham, J. A. Long, D. Smith, Q. Zhao, & S. Wise (Eds.), *12th International Conference on Geographic Information Science (GIScience 2023)*, volume 277 of *Leibniz International Proceedings in Informatics (LIPIcs)* (pp. 10:1–10:17). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik `https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.GIScience.2023.10`.

Wood, J. (2021). *Grid map allocation* `https://observablehq.com/@jwolondon/gridmap-allocation?collection=@jwolondon/utilities`.

Wood, J., Badawood, D., Dykes, J., & Slingsby, A. (2011). BallotMaps: Detecting name bias in alphabetically ordered ballot papers. *IEEE Transactions on Visualization and Computer Graphics*, *17*(12), 2384–91.

Wood, J., Dykes, J., & Slingsby, A. (2010). Visualisation of origins, destinations and flows with OD maps. *The Cartographic Journal*, *47*(2), 117–129.