densify: An R package to prune sparse data frames of typological linguistic data

8 February 2024

Summary

The R package `densify` provides a procedure to prune input data frames containing empty cells (or cells with values {?} or {NA}) to denser sub-matrices with fewer rows, columns and empty cells. The pruning process, split into three functions, trades off a series of variably weighted concerns, including data retention, coding density (proportion of non-empty cells) and taxonomic diversity of rows (if the entities are represented in a taxonomic structure). Users can adapt the relative weights given to these concerns through various parameters for the densification process to best fit their needs. As such, the software is useful for several purposes, including the densification of sparse input matrices and the subsampling of large input matrices according to a procedure that is sensitive to taxonomic structure.

Statement of Need

While the software will run on any data frame (with rows representing any entities with or without taxonomic structure and columns representing variables), it was primarily designed to prune data frames of linguistic typological data, where the rows are languages (taxa) and the columns typological variables (sometimes also called characters, parameters or features).

Linguistic typological data is increasingly available in large-scale databases, and many analyses that aim at exploring diversity or testing hypotheses rely on such databases. Some of these resources have information for nearly all variables across nearly all languages in the database (i.e. they have complete or near-complete coding density), but the dataframe may be too large for certain computationally intensive analyses (e.g., PHOIBLE (Moran and McCloy 2019), Grambank (Skirgård et al. 2023)). Other databases (e.g., WALS (Dryer and Haspelmath 2013), AUTOTYP (Bickel et al. 2023), Lexibank (List et al. 2023)) exhibit variables that are coded for different sets of languages, resulting in sparse language-variable matrices. Combining data from various databases via language identifiers like glottocodes usually increases sparsity because the language samples do not match.

When datasets are too large or too sparse for computational applications, it can thus be necessary to generate and subsequently operate on a subset of the data represented in a smaller and denser matrix. Thereby, researchers might be particularly interested in maintaining taxonomic diversity in the languages represented, preferentially removing languages belonging to clades represented by many other languages in the sample and penalizing the removal of language isolates or languages which represent small language families.

While certain packages exist to generate sub-matrices from varying input matrices according to principled criteria (e.g., `admmDensestSubmatrix` (Ames and Bombina 2019), which identifies the densest sub-matrix of an input graph of a specified size; or `FSelector` (Romanski, Kotthoff, and Schratz 2023) and `varrank` (Kratzer and Furrer 2022), which perform attribute subset selection based on various tests and entropy measures to identify the most relevant attributes of a data input), `densify` provides a flexible, explicit, and taxonomy-sensitive pruning algorithm that focuses both on the removal of rows and columns and does not require the size of the sub-matrix to be specified a priori.

Usage

The package `densify` provides the data from The World Atlas of Language Structures (WALS) (Dryer and Haspelmath 2013) and the language taxonomy provided by Glottolog v. 4.8 (Hammarström et al. 2023) as example data. The accompanying package vignette features a detailed demonstration of the utility and flexibility of `densify` to subsample an input matrix according to varying needs, using this data.

Preparing input

The data frame that requires subsetting must have rows representing taxa or observations (with taxon names provided in a dedicated column) and columns representing variables (and variable names as column names). Any cells with empty entries, not applicable or question marks must be coded as NA. If matrix densification should be sensitive to taxonomic structure, a taxonomy must be provided as a `phylo` object (cf. Paradis and Schliep 2019) or as an adjacency table (i.e. a data frame containing columns `id` and `parent_id`, with each row encoding one parent-child relationship). Every taxon in the input data frame must be included in the taxonomy (as a tip or node). If a language taxonomy is to be provided, glottolog (`glottolog_languoids`) can be used directly.

```
install.packages("densify")
library(densify)

# prepare data: WALS and Glottolog
data(WALS)
data(glottolog_languoids)

# any question marks, empty entries, "NA"s must be coded as NA
WALS[WALS=="?"] <- NA
WALS[WALS=="NA"] <- NA
head(WALS)

# all taxa must be present in the taxonomy used for pruning
WALS <- WALS[which(WALS$Glottocode %in% glottolog_languoids$id), ]
```

Pruning

Iterative pruning of the input matrix is performed by the `densify()` function, which requires the following information:

- The original data frame with observations in rows and variables in columns (`data`). No default.
- `A specification of which columns should be densified (`cols`). Default: all columns densified.`
- A taxonomy tree as a phylo object or a data frame with columns `id` and `parent_id` (`taxonomy`). No default.
- `The name fo the column identifying taxa (`taxon_id`).`
- `A string specifying the scoring type used for calculating row-wise importance weights (`scoring`).`
- An optional integer specifying the threshold for variability in variables (`min_variability`). Default: `1`.
- An optional list denoting conditions for densification to end (`limits`). Available conditions are `min_coding_density` (denoting target matrix coding density), `min_prop_rows` (denoting the minimal proportion of rows that have to be retained in the data) and `min_prop_cols` (denoting the minimal proportion of columns that have to be retained in the data). Default: `min_coding_density = 1`.
- An optional list denoting additional weighting factors during importance score calculation. Available parameters are `coding` and `taxonomy`, $\in (0, 1)$. They tweak the relative importance of coding density

and taxonomic diversity in the pruning process. Setting the value to 0, `NA` or `NULL` disables the corresponding weight calculation. Default: `coding = 1, taxonomy = 1`.

For a more detailed discussion of the parameters, refer to the vignette hosted in the software repository.

The output of the function is a densify_result object, documenting several summary statistics of the sub-matrix resulting from each iteration. These include the number of available data points, the overall proportion of coded data in the matrix, the coding densities of the least well-coded taxon and variable, the coding densities of the most well-coded taxon and variable, the median coding densities of the all taxa and variables, and a taxonomic diversity index (Shannon diversity of the highest taxonomic level).

```r
set.seed(2024)
example_result <- densify(data = WALS,
                          taxonomy = glottolog_languoids,
                          taxon_id = "Glottocode",
                          scoring = "log_odds",
                          min_variability = 3,
                          limits = list(min_coding_density = 1),
                          scoring_weights = list(coding = 1, taxonomy = 1))
head(example_result)
```

Finding the optimal number of iterations and producing the sub-matrix

The functions `prune()` and `rank_results()` can identify the optimal sub-matrix via a quality score, computed via a user-defined function composed of any combination of available statistics from the densify_result object. The default score thereby maximizes the product of the number of available data points and the overall coding density.

The function `prune()` thereby retrieves the optimal sub-matrix, while `rank_results()` returns the relative ranks of all sub-matrices given the score.

```r
# use prune() to obtain the optimum submatrix, given different scores
example_optimum_1 <- prune(example_result,
                           score = n_data_points*coding_density)
example_optimum_2 <- prune(example_result,
                           score = n_data_points*coding_density*taxonomic_index^3)

# use rank_results() to obtain the same optima, alongside further ranking
example_ranks_1 <- rank_results(example_result,
                                score = n_data_points*coding_density)
example_ranks_2 <- rank_results(example_result,
                                score = n_data_points*coding_density*taxonomic_index^3)
```

The relative ranking of sub-matrices given the specified quality score can also be visualized using `visualize()`.

```r
# use visualize() to illustrate quality scores and optimum
# using a first score
visualize(example_result, score = n_data_points*coding_density)

# using a second score
visualize(example_result, score = n_data_points*coding_density*taxonomic_index^3)
```

Conclusions

The R package `densify` provides users with a flexible and explicit method to generate sub-matrices from an input matrix in a mathematically principled way. This paper shows case examples using a standard sparse linguistic dataset (WALS) and the standard linguistic taxonomy provided by Glottolog. Additional examples and usage details are found in the vignette and hosted in the software repository on GitHub.

Acknowledgements

The authors declare that there are no conflicts of interest.

References

Ames, Brendan, and Polina Bombina. 2019. *admmDensestSubmatrix: Alternating Direction Method of Multipliers to Solve Dense Dubmatrix Problem.* https://CRAN.R-project.org/package=admmDensestSubmatrix.

Bickel, Balthasar, Johanna Nichols, Taras Zakharko, Alena Witzlack-Makarevich, Kristine Hildebrandt, Michael Rießler, Lennart Bierkandt, Fernando Zúñiga, and John B Lowe. 2023. "The AUTOTYP Database (V1.1.1)." https://doi.org/10.5281/zenodo.7976754.

Dryer, Matthew S., and Martin Haspelmath, eds. 2013. *WALS Online (V2020.3).* Data set. Zenodo. https://doi.org/10.5281/zenodo.7385533.

Hammarström, Harald, Robert Forkel, Martin Haspelmath, and Sebastian Bank. 2023. "Glottolog/Glottolog: Glottolog Database 4.8." Zenodo. https://doi.org/10.5281/zenodo.8131084.

Kratzer, Gilles, and Reinhard Furrer. 2022. *Varrank: Heuristics Tools Based on Mutual Information for Variable Ranking.* https://CRAN.R-project.org/package=varrank.

List, Johann-Mattis, Robert Forkel, Simon J. Greenhill, Christoph Rzymski, Johannes Englisch, and Russell D. Gray. 2023. "Lexibank Analysed." Zenodo. https://doi.org/10.5281/zenodo.7836668.

Moran, Steven, and Daniel McCloy. 2019. "cldf-datasets/phoible: PHOIBLE 2.0.1 as CLDF dataset." Zenodo. https://doi.org/10.5281/zenodo.2677911.

Paradis, Emmanuel, and Klaus Schliep. 2019. "Ape 5.0: An Environment for Modern Phylogenetics and Evolutionary Analyses in R." *Bioinformatics* 35: 526–28. https://doi.org/10.1093/bioinformatics/bty633.

Romanski, Piotr, Lars Kotthoff, and Patrick Schratz. 2023. *FSelector: Selecting Attributes.* https://CRAN.R-project.org/package=FSelector.

Skirgård, Hedvig, Hannah J. Haynie, Harald Hammarström, Damian E. Blasi, Jeremy Collins, Jay Latarche, Jakob Lesage, et al. 2023. "Grambank V1.0." Zenodo. https://doi.org/10.5281/zenodo.7740140.