The Journal of Open Source Software

bwsample: Processing Best-Worst Scaling data

Ulf A. Hamster[1]

1 Berlin-Brandenburgische Akademie der Wissenschaften, Berlin, Germany

DOI: 10.21105/joss.03324

Software

- Review
- Repository
- Archive

Editor: Mikkel Meyer Andersen

Reviewers:

- @ejhigson
- @jakryd

Submitted: 13 March 2021
Published: 26 July 2021

Summary

bwsample is a Python package that provides algorithms to sample best-worst scaling sets (BWS sets), extract and count pairwise comparisons from user-evaluated BWS sets, and compute rankings and scores.

Statement of need

We are using the bwsample package as part of an *Active Learning* experiment in which linguistics experts and lay people (crowdsourcing) are judging sentences examples with the *Best-Worst Scaling* (BWS) method (Fig. 1). BWS is *"... the cognitive process by which respondents repeatedly choose the two objects in varying sets of three or more objects that they feel exhibit the largest perceptual difference on an underlying continuum of interest"* (Finn & Louviere, 1992, p. 13). In our context, BWS is primarily used as a means of data collection what is economically efficient than using pairwise comparison user interfaces (Hamster, 2021).

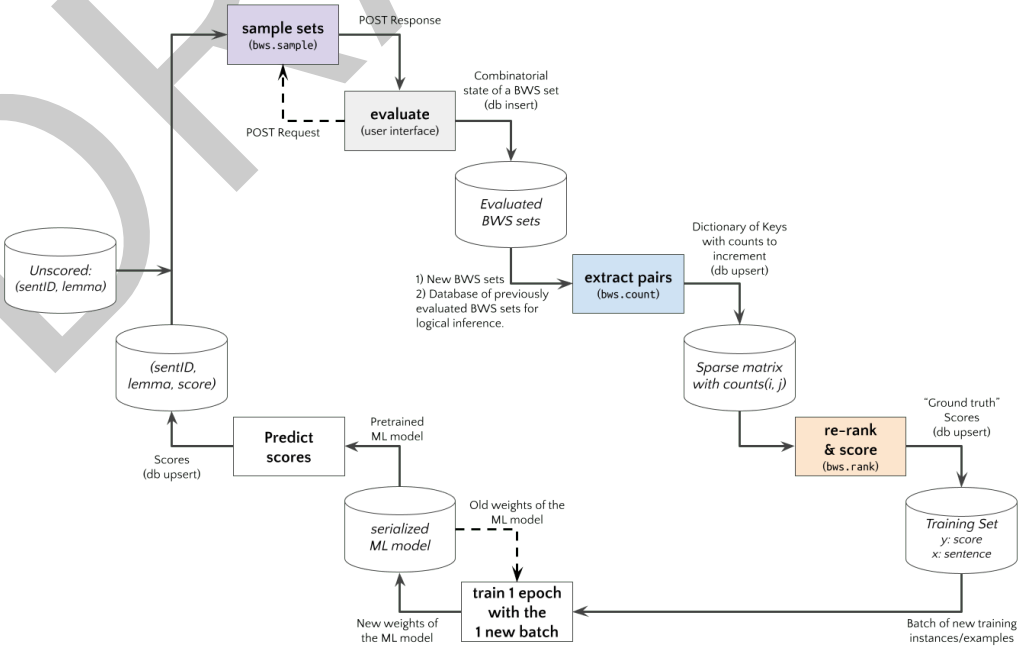

Figure 1: Using bwsample (bws) in an Active Learning experiment.

Software Features

The *sampling* algorithms ensure overlapping BWS sets, and are deployed in the REST API for a Web App. Overlaps are required for counting pairs by logical inference (Hamster, 2021). A possible question is how many items have to be shown twice a) initially, and b) after the pair frequency database grew over time to gather reasonable amounts of counting or resp. frequency data? The implemented *counting* algorithms can distinguish between 3 types of directly extract pairs and 7 types of logically inferred pairs. This opens the opportunity for further analysis, e.g. to detect inconsistent evaluations (Hamster, 2021), or to assign weights to different types of pairs. To compute *rank* items from pairwise comparison data, five algorithms are available: a) Eigenvector estimation of the reciprocal pairwise comparison matrix as scores (Saaty, 2003), b) MLE estimation of the Bradley-Terry-Luce probability model (Hunter, 2004, pp. 386–387), c) Simple ratios for each pair and sum the ratios for each item, d) Chi-Squared based p-value for each pair and sum 1 minus p-values for each item, e) Estimation of the transition probability that the next element is better. All ranking algorithms are implemented based on sparse matrix operations.

Acknowledgements

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 433249742.

Software Citations

`bwsample` is written in Python 3.6+ (Van Rossum & Drake, 2009) and uses the following software packages:

- `NumPy` https://github.com/numpy/numpy (Harris et al., 2020)
- `SciPy` https://github.com/scipy/scipy (Virtanen et al., 2020)
- `scikit-learn` https://github.com/scikit-learn/scikit-learn (Pedregosa et al., 2011)

References

Finn, A., & Louviere, J. J. (1992). Determining the Appropriate Response to Evidence of Public Concern: The Case of Food Safety. *Journal of Public Policy & Marketing*, *11*(2), 12–25. https://doi.org/10.1177/074391569201100202

Hamster, U. (2021). *Extracting Pairwise Comparisons Data from Best-Worst Scaling Surveys by Logical Inference*. OSF Preprints. https://doi.org/10.31219/osf.io/qkxej

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hunter, D. R. (2004). MM algorithms for generalized Bradley-Terry models. *The Annals of Statistics*, *32*(1), 384–406. https://doi.org/10.1214/aos/1079120141

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau,

D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Saaty, T. L. (2003). Decision-making with the AHP: Why is the principal eigenvector necessary. *European Journal of Operational Research*, *145*(1), 85–91. https://doi.org/10.1016/S0377-2217(02)00227-8

Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual*. CreateSpace. ISBN: 1441412697

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., Walt, S. J. van der, Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … Mulbregt, P. van. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2