Appendix
The Zipfian Challenge: Learning the statistical fingerprint of natural languages

Christian Bentz
Department of General Linguistics
University of Tübingen
chris@christianbentz.de

Limitations

This study is a proof of concept, using well-known classification algorithms. KNNs and MLPs are still applied today, and are competitive on general classification leaderboards (https://paperswithcode.com/task/classification). More complex models such as LSTMs, and pre-trained transformers might further increase classification results, especially if more training strings become available. However, a general question is whether the complexity of models should be arbitrarily increased for marginal gains, when simpler methods reach almost the ceiling of performance.

The collection of texts is ongoing. More data, especially on the "non-writing" side is certainly needed. Currently, animal communication is only represented by bird songs. It should be possible to retrieve transcribed songs also of cetaceans, and communicative interactions of apes. Other non-writing symbolic systems produced by humans, such as barn stars and totem poles (Sproat, 2014) could also be integrated. On the writing side, transcribed sign languages are heavily undersampled compared to other languages in the TeDDi sample.

Also, note that the results reported here hold for the general distinction between *all* strings in the category "writing" and *all* strings in the category "non-writing". Of course, it can be a more challenging problem to tease apart *one particular* non-writing system from all of writing. For example, if the KNN algorithm ($k = 5$) is applied to just python code strings (of length 100) contrasted with writing, then the F1 score drops to 0.53. This suggests that python code by itself is (almost) indistinguishable from writing with the current methods. This is not surprising given that python code extensively uses English words for being more readable (see Table 1 in main paper). On the other hand, if KNNs are applied to only morse code or bird song we get near perfect discrimination with writing (F1 scores of 1.00 and 0.98 respectively). Once more data on different non-writing systems is collected, more specific hypotheses for individual systems can be tested.

Lastly, another limitation is the focus on single UTF-8 characters as basic units of analysis. In future studies, rather than using fixed units, flexible "subword" vocabularies could be generated (Gutierrez-Vasques et al., 2021).

References

Ximena Gutierrez-Vasques, Christian Bentz, Olga Sozinova, and Tanja Samardzic. 2021. From characters to words: the turning point of BPE merges. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3454–3468.

Richard Sproat. 2014. A statistical comparison of written language and nonlinguistic symbol systems. *Language*, 90(2):457–481.

A Sampling procedure

The sampling procedure to retrieve strings of length 10, 100, 1000 from the subcorpora is visualized in Figure 1. For each of the eleven subcorpora, a maximum of 10 files is sampled. From these files, in turn, a maximum of 10 strings of given length are sampled.

B Hyperparameter Tuning

The change of performance (F1) with increase of a given hyperparameter for the KNN and MLP algorithms tested in this study. The performance is stable across k-values. The optimal MLP depth is two or three layers. At four layers the performance slightly drops. Moreover, the performance of the networks is relatively stable at sizes between five to ten units, with a potential slight increase for a number of units > 10. However, very few of the networks with more than ten units converge.

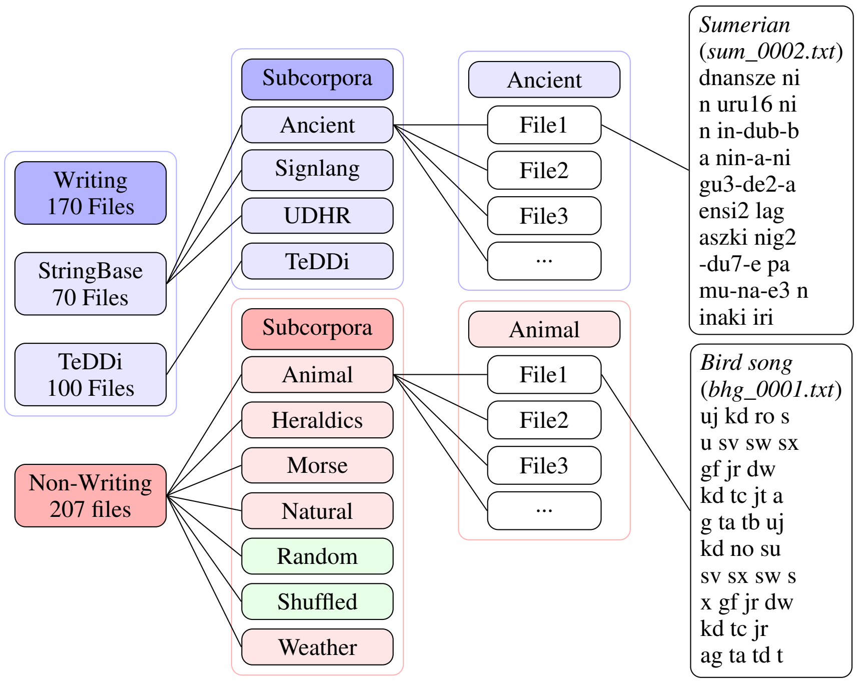

Figure 1: Illustration of database structure and sampling procedure. For each of the 170 "writing" and 207 "non-writing" files, up to a maximum of 10 chunks of a given length (10, 100, 1000 UTF-8 characters) are extracted.

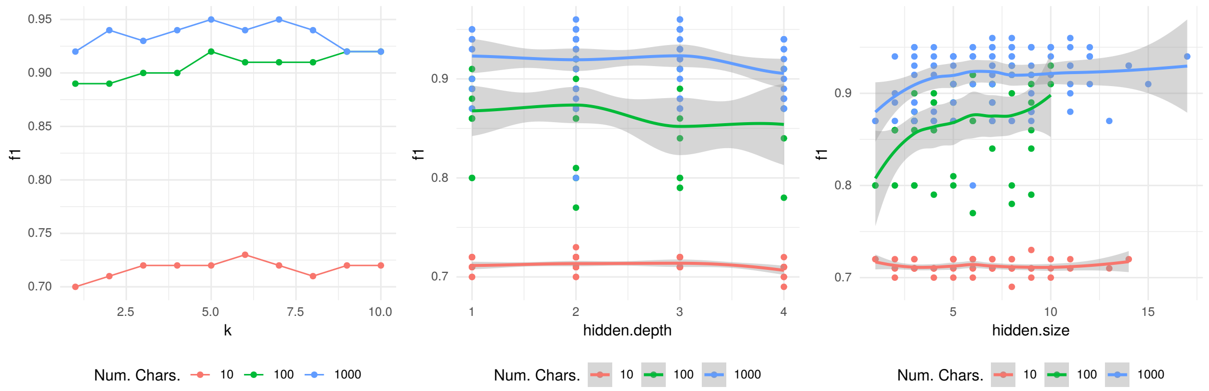

Figure 2: Hyperparameters versus F1 scores for the best algorithms (knn and MLP). The hyperparameter for the knn algorithm (left panel) is k, i.e. the number of neighbours used for classification. The most relevant hyperparameters for the MLP model are the number of hidden layers (i.e. depth, middle panel), and the overall number of hidden units (i.e. size, right panel).

C Stabilization of feature values

The stablization of feature values with the length of strings is illustrated in Figure 3, Figure 4, Figure 5, and Figure 6.

For the majority of strings the values have stabilized when c. 100 characters are provided across the different features (H, h, TTR, R). For highly repetitive systems at the character level (DNA, morse code), this effect is clearest, while natural language writing typically displays growing entropies beyond 100 or even 1000 characters. This speaks to the information-theoretic "productivity" of the system.

D Post hoc experiments with subcorpora

Results for *post hoc* experiments selectively removing particular subcorpora are given in Table 1. The best KNN at 100 characters is used as the baseline here. Generally, the performance is relatively stable, i.e. staying around 0.9 in terms of F1-scores. The removal of strings from the TeDDi corpus has the highest positive impact on the performance (increase by 0.05). The TeDDi sample is a highly diverse sample across languages and writing systems of the world. The effect might be explained partly by unusual strings found in this sample, also due to contamination with non-linguistic strings. The removal of DNA ("natural") has the highest negative impact on the performance (decrease by 0.05).

Table 1: Results of post hoc experiments with removal of strings from specific subcorpora.

Version	F1
Best KNN at 100 chars (k=5)	0.92
UDHR strings removed	0.93
TeDDi strings removed	0.97
Random strings removed	0.88
Shuffled strings removed	0.94
DNA removed	0.87
Bird Song removed	0.89
Python code removed	0.91
Morse code removed	0.91
Weather symbols removed	0.92
Proto-cuneiform removed	0.92
Heraldics removed	0.91

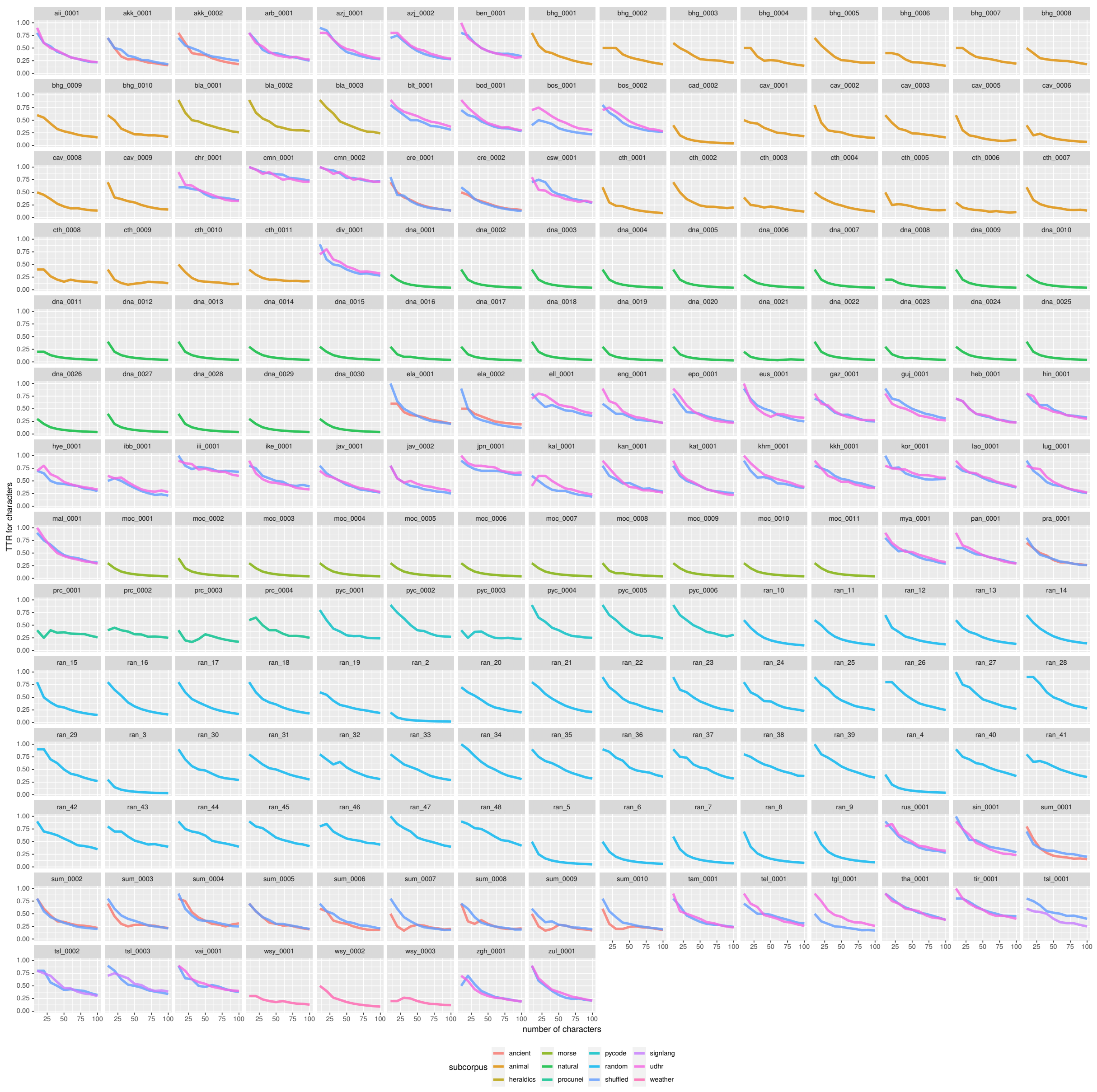

Figure 3: TTR for characters as a function of string length.

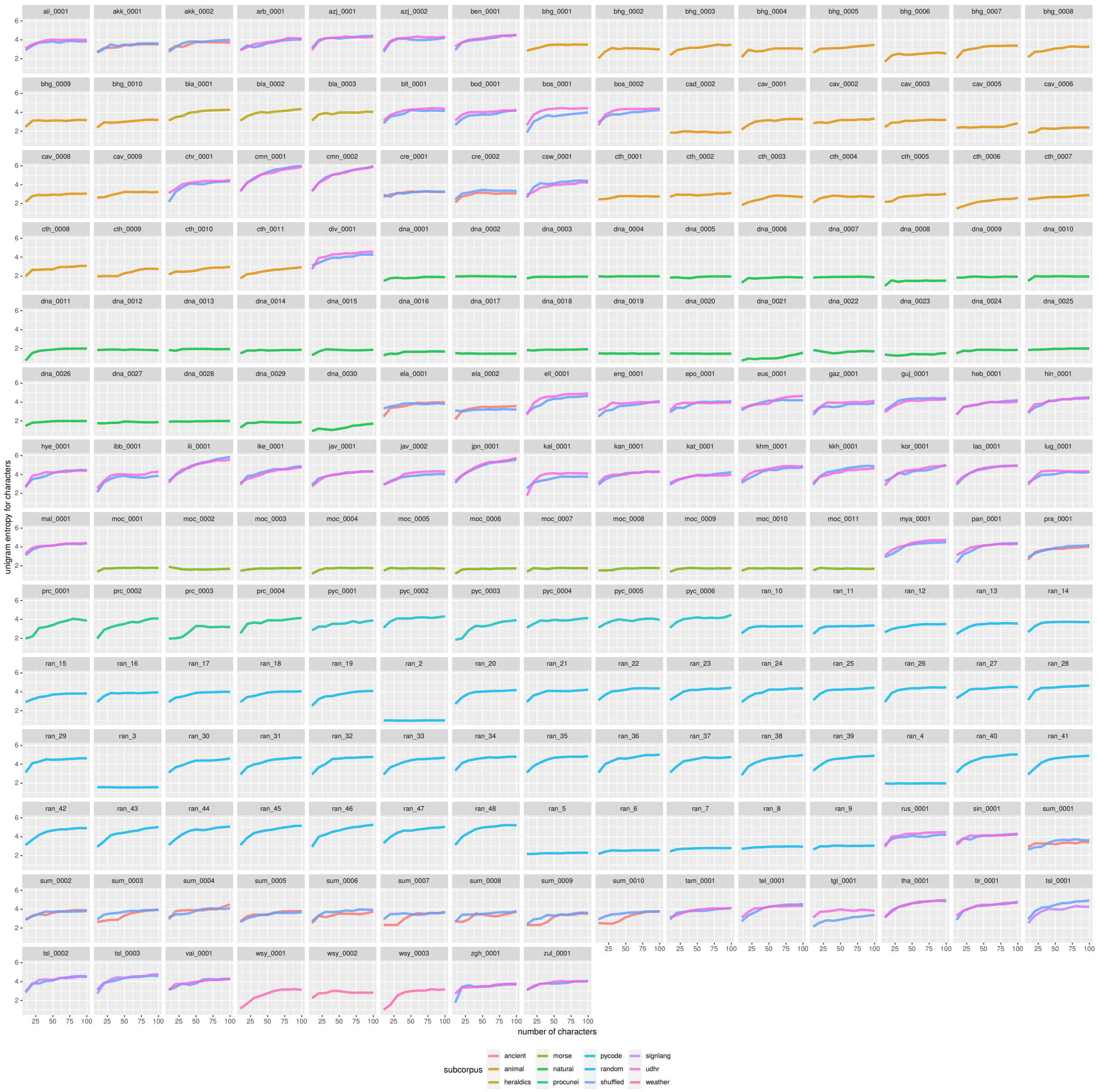
Figure 4: Unigram entropy for characters as a function of string length.

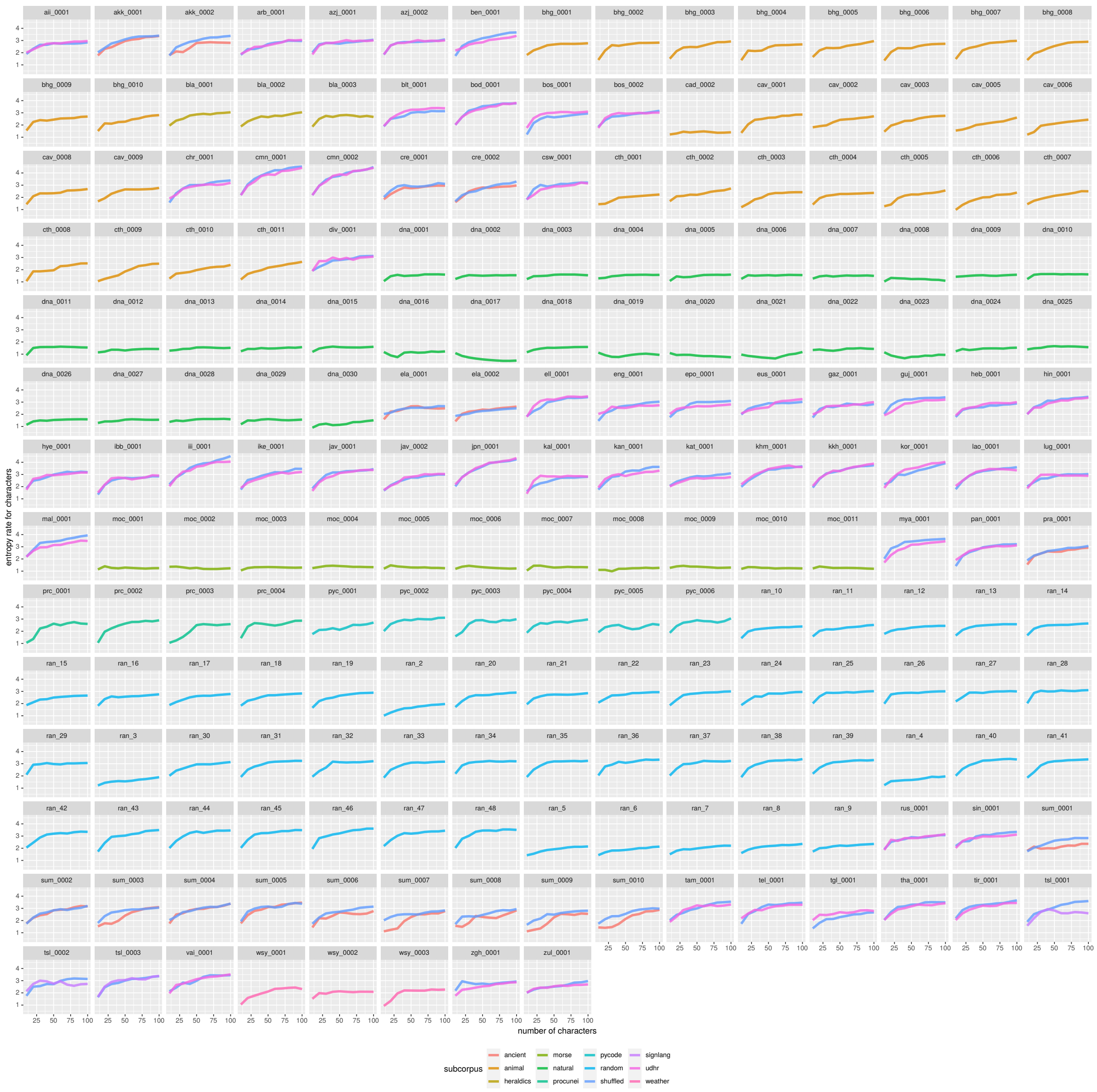
Figure 5: Entropy rate for characters as a function of string length.

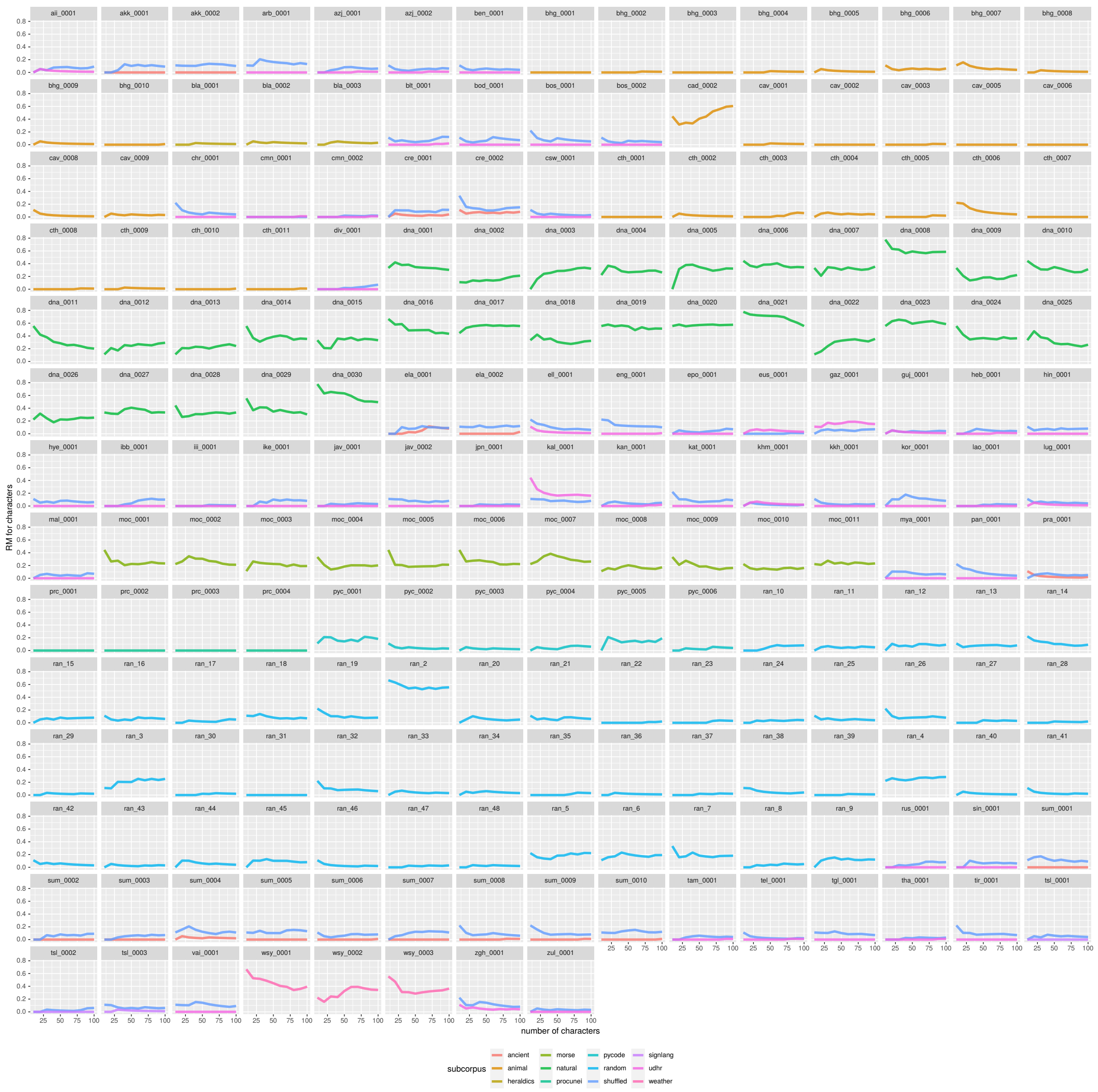

Figure 6: Repetition rate for characters as a function of string length.