Morphological structure can escape reduction effects from mass admixture of second language speakers: evidence from Sino-Tibetan *Supplementary Materials:* Statistical Models[*]

Balthasar Bickel

S1 Data

The data provide for five ways of quantifying morphology:

- odds of presence vs absence of specific morpheme types
- odds of presence vs absence of specific domains, split in phonology vs grammar
- odds and number of domains in which specific cohesion rules/constraints hold, split in phonology vs grammar
- size (in morpheme types) of these domains, split in phonology vs grammar
- odds of presence vs absence of specific inflectional categories (synthesis)

The data on morpheme types are contained in Tables 1, 4, 7, 8, and, as a synopsis, in Table 13 in the main paper. A machine-readable version is available as `morpheme_types.csv`. The data on cohesion domains appear in Tables 2-3, 5-6, 8-9, and 11-12 and are collected in machine-readable form in the file `domains.csv`. The survey of inflectional categories is presented in Table 14 and in `synthesis.csv`.

In each of these data sets the languages are grouped as either L2 = "few" vs L2 = "much". "Few" means that the community includes only a handful of second language speakers (Chintang and Bunan); "much" means that there are several million of second language speakers (Mandarin and Burmese).

```r
morpheme_types <- read.csv('morpheme_types.csv', stringsAsFactors = FALSE)
morpheme_types.long <- gather(morpheme_types, "MorphemeType", "Presence", V2:SA) %>%
  mutate_if(is.character, factor) %>%
  mutate(Presence = ifelse(Presence %in% 'present', TRUE, FALSE))
# head(morpheme_types.long)

domains <- read.csv('domains.csv', strip.white = T, sep = ',',
                    stringsAsFactors = T)

domain_presence <- distinct(domains, Language, Domain, .keep_all = T) %>%
  mutate(Presence = TRUE) %>%
  spread(Domain, Presence, fill = FALSE) %>%
  gather(Domain, Presence, -c(1:4)) %>%
  mutate(Domain = as.factor(Domain))
# clean-up that brms seems to need:
levels(domain_presence$Domain) <- paste("Domain", 1:length(levels(domain_presence$Domain)))

constraint_counts_p <- domains %>%
  filter(Level %in% 'phonology') %>% droplevels() %>%
  group_by(Language, ConstraintType, .drop = F) %>%
```

[*]This document was `rmarkdown::render`'ed from an `R` script available at https://osf.io/mt98r

```
  summarize(Count = n()) %>%
  mutate(L2 = ifelse(Language %in% c('Bunan', 'Chintang'), 'few', 'much')) %>% as.data.frame
   constraint_counts_p$L2 <- as.factor(constraint_counts_p$L2)
# head(constraint_counts_p)

constraint_counts_g <- domains %>%
  filter(Level %in% 'grammar') %>% droplevels() %>%
  group_by(Language, ConstraintType, .drop = F) %>%
  summarize(Count = n()) %>%
  mutate(L2 = ifelse(Language %in% c('Bunan', 'Chintang'), 'few', 'much')) %>% as.data.frame
  constraint_counts_g$L2 <- as.factor(constraint_counts_g$L2)
# head(constraint_counts_g)

domain_size <- domains %>%
  mutate(Size = str_count(Domain, '-') + 1)
# head(domain_size)

synthesis <- read.csv("synthesis.csv")
synthesis.long <- gather(synthesis, "Category", "Presence", A_Agreement:Connectives) %>%
  mutate_if(is.character, factor) %>%
  mutate(Presence = ifelse(Presence %in% 'present', TRUE, FALSE))
# head(synthesis.long)
```

S2 Models and model evaluation methods

We assess the effect of L2 on the odds, counts or sizes of the various patterns in generalized linear multilevel ("mixed effects") model. We assume Bernoulli likelihoods for the odds and Poisson likelihoods for the counts and sizes (counts of morpheme types), each with their canonical links (log odds and log, respectively). For the predictors we use deviation coding so that the intercept represents that grand mean and the L2 coefficient indicates the difference in the log odds/counts/size between high-admixture and low-admixture languages. Here is a helper function to set up the contrast matrix:

```
contr.m <- function(f) { # a factor
  ctr <- contr.treatment(nlevels(f))
  m <- matrix(rep(1/nlevels(f), length(ctr)), ncol = ncol(ctr))
  ctr - m }
```

As explained in the main text, we focus on mean L2 effects and do not include varying ("random") slopes or interaction terms because the distribution of the relevant levels (individual types, domains, constraints and categories) is far too sparse across languages to allow consistent estimates — indeed, many levels occur only in one single language. However, we do include varying intercepts by language in order to achieve better overall estimates (through shrinkage) and to allow factoring out the variation between languages. In the Poisson models, we furthermore include what is sometimes called "observation-level random effects (OLRE)" in order to absorb various sources of overdispersion (Harrison 2014).[1]

Given the fact that we are as interested in the absence as in the presence of L2 effects, we fit the models in a Bayesian framework. We rely on `Stan` (Carpenter et al. 2017) and the `R` package `brms` (Bürkner 2018) for this. We choose a weakly regularizing Student-*t* prior centered on 0 because (i) our null hypothesis is theoretically specific and privileges L2 estimates near 0, and (ii) flatter priors expect too much probability mass at the extremes of a Bernoulli model and unrealistically large counts in Poisson models (McElreath 2020). For the group-level ("random") effects we use a weakly regularizing half-*Cauchy*(0,1) prior.

[1] We also tried negative-binomial likelihoods in the Poisson models, but they didn't improve the fit much and instead yielded many transition divergences in the NUTS sampler.

```
priors <- c(prior(student_t(5, 0, 1), class = b),
            prior(cauchy(0,1), class = sd))
```

We assess the performance of models by how well they predict held out or as yet undocumented observations, i.e. through the *expected log pointwise predictive density* (elpd). We estimate elpds through Pareto-smoothed importance sampling with moment matching (PSIS+) from the posterior log probabilities (Vehtari et al. 2016; Paananen et al. 2019; McElreath 2020). In some models (mentioned below), however, one or more predictor levels occur so sparsely that they cannot be predicted when holding out even just single observations, either through importance sampling or explicit cross-validation (i.e. refitting the model by leaving out one observation at a time). In these cases, we resort to the Watanabe-Akaike Information Criterion (WAIC) as an estimator of the elpd (Vehtari et al. 2016; McElreath 2020).

We compare predictive performance with Aikake weights for $\text{elpd}_{\text{waic}}$ estimates (McElreath 2020) and with Akaike weights corrected by Bayesian bootstrapping (Yao et al. 2018) for $\text{elpd}_{\text{psis+}}$ estimates (also known as "pseudo-BMA+"). We choose this method because it takes into account the full posterior uncertainty and, unlike stacking, regularizes weights away from 0 and 1. For nested binary models, the methods perform equally well (Yao et al. 2018).

In order to assess the variation between languages, we furthermore examine a version of the intra-class (intra-group) correlation coefficient that is generalized to Bayesian non-Gaussian models. The `performance` package (Lüdecke et al. 2020) implements this as the *Variance Partition Coefficient* (VPC) defined as 1 minus the ratio between the variance of posterior predictive distributions not conditioned by the varying intercept and the variance of these distributions conditioned by the varying intercepts. Values close to 0 mean that the two predictive variances are similar to each other, i.e. that the varying intercepts have no impact and that the estimates for each language are virtually the same as the estimate for the overall intercept, i.e. the grand mean ('complete pooling').

S2.1 Odds of specific morpheme types

```
contrasts(morpheme_types.long$MorphemeType) <- contr.m(morpheme_types.long$MorphemeType)
contrasts(morpheme_types.long$L2) <- contr.m(morpheme_types.long$L2)

mtyp_plus_L2 <- brm(Presence ~ MorphemeType + L2 + (1|Language),
                    family = bernoulli(),
                         prior = priors,
                    data = morpheme_types.long,
                    chains = 4, cores = 4,
                    iter = 2000, warmup = 1000,
                    save_all_pars = T, file = 'models/mtyp_plus_L2',
                    control = list(adapt_delta = .99, max_treedepth = 20))

mtyp_without_L2 <- update(mtyp_plus_L2, ~ . - L2, file = 'models/mtyp_without_L2')
mtyp_without_L <- update(mtyp_plus_L2, ~ . - L2 - (1|Language),
                         file = 'models/mtyp_without_L')

(mtyp_plus_L2.loo <- loo(mtyp_plus_L2, moment_match = T))
```

```
##
## Computed from 4000 by 28 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo    -20.1 3.1
## p_loo         5.5 1.1
## looic        40.2 6.3
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     27    96.4%   1006
##  (0.5, 0.7]   (ok)        1     3.6%   1080
##    (0.7, 1]   (bad)       0     0.0%   <NA>
##    (1, Inf)   (very bad)  0     0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
(mtyp_without_L2.loo <- loo(mtyp_without_L2, moment_match = T))
```

```
##
## Computed from 4000 by 28 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo    -19.8 2.9
## p_loo         4.7 0.8
## looic        39.5 5.9
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     27    96.4%   1744
##  (0.5, 0.7]   (ok)        1     3.6%   1495
##    (0.7, 1]   (bad)       0     0.0%   <NA>
##    (1, Inf)   (very bad)  0     0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
(mtyp_without_L.loo <- loo(mtyp_without_L, moment_match = T))
```

```
##
## Computed from 4000 by 28 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo    -19.0 2.7
## p_loo         4.0 0.7
## looic        38.0 5.3
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

```
mtyp_L2_weights <- loo_model_weights(list(mtyp_plus_L2.loo, mtyp_without_L2.loo),
                                  method = 'pseudobma')
mtyp_L_weights <- loo_model_weights(list(mtyp_without_L2.loo,mtyp_without_L.loo),
                                    method = 'pseudobma')
mtyp_L_vpc <- variance_decomposition(mtyp_without_L2, robust = T, ci = .5)
```

S2.2 Odds of specific domains: phonology

```
pdomains <- droplevels(subset(domain_presence, Level %in% 'phonology'))
contrasts(pdomains$Domain) <- contr.m(pdomains$Domain)
contrasts(pdomains$L2) <- contr.m(pdomains$L2)

pdom_plus_L2 <- brm(Presence ~ Domain + L2 + (1|Language),
                    family = bernoulli(),
                        prior = priors,
                    data = pdomains,
                    chains = 4, cores = 4,
                    iter = 3000, warmup = 1000,
                        save_all_pars = T, file = 'models/pdom_plus_L2',
                    control = list(adapt_delta = .99, max_treedepth = 20))

pdom_without_L2 <- update(pdom_plus_L2, ~ . - L2, file = 'models/pdom_without_L2')
pdom_without_L <- update(pdom_plus_L2, ~ . - L2 - (1|Language),
                         file = 'models/pdom_without_L')

(pdom_plus_L2.loo <- loo(pdom_plus_L2, moment_match = T))
```

```
##
## Computed from 8000 by 588 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -113.0 17.0
## p_loo        18.2  3.4
## looic       226.0 33.9
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     584   99.3%   2399
##  (0.5, 0.7]   (ok)         4    0.7%   1675
##    (0.7, 1]   (bad)        0    0.0%   <NA>
##    (1, Inf)   (very bad)   0    0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
(pdom_without_L2.loo <- loo(pdom_without_L2, moment_match = T))
```

```
##
## Computed from 8000 by 588 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -112.7 16.9
## p_loo        17.7  3.3
## looic       225.4 33.8
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     583   99.1%   2448
##  (0.5, 0.7]   (ok)         5    0.9%   1700
##    (0.7, 1]   (bad)        0    0.0%   <NA>
##    (1, Inf)   (very bad)   0    0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
(pdom_without_L.loo <- loo(pdom_without_L, moment_match = T))
```

```
##
## Computed from 8000 by 588 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -112.3 16.8
## p_loo        16.1  3.0
## looic       224.6 33.7
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     584   99.3%   3269
##  (0.5, 0.7]   (ok)         4    0.7%   1705
##    (0.7, 1]   (bad)        0    0.0%   <NA>
##    (1, Inf)   (very bad)   0    0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
pdom_L2_weights <- loo_model_weights(list(pdom_plus_L2.loo, pdom_without_L2.loo),
                                      method = 'pseudobma')
pdom_L_weights <- loo_model_weights(list(pdom_without_L2.loo, pdom_without_L.loo),
                                     method = 'pseudobma')
pdom_L_vpc <- variance_decomposition(pdom_without_L2, robust = T, ci = .5)
```

S2.3 Odds of specific domains: grammar

```
gdomains <- droplevels(subset(domain_presence, Level %in% 'grammar'))
contrasts(gdomains$Domain) <- contr.m(gdomains$Domain)
contrasts(gdomains$L2) <- contr.m(gdomains$L2)

gdom_plus_L2 <- brm(Presence ~ Domain + L2 + (1|Language),
                    family = bernoulli(),
                    prior = priors,
                    data = gdomains,
                    chains = 4, cores = 4,
                    save_all_pars = T, file = 'models/gdom_plus_L2',
                    iter = 2000, warmup = 1000,
                    control = list(adapt_delta = .999999,  max_treedepth = 25))

gdom_without_L2 <- update(gdom_plus_L2, ~ . - L2, file = 'models/gdom_without_L2')
gdom_without_L <- update(gdom_plus_L2, ~ . - L2 - (1|Language),
                         file = 'models/gdom_without_L')
```

```r
(gdom_plus_L2.loo <- loo(gdom_plus_L2, moment_match = T))
```

```
##
## Computed from 4000 by 504 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -101.7 15.8
## p_loo        17.3  3.3
## looic       203.5 31.5
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     500   99.2%   1170
##  (0.5, 0.7]   (ok)         4    0.8%   1398
##    (0.7, 1]   (bad)        0    0.0%   <NA>
##    (1, Inf)   (very bad)   0    0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```r
(gdom_without_L2.loo <- loo(gdom_without_L2, moment_match = T))
```

```
##
## Computed from 4000 by 504 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -100.9 15.6
## p_loo        16.4  3.1
## looic       201.9 31.2
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     498   98.8%   1378
##  (0.5, 0.7]   (ok)         6    1.2%   1076
##    (0.7, 1]   (bad)        0    0.0%   <NA>
##    (1, Inf)   (very bad)   0    0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```r
(gdom_without_L.loo <- loo(gdom_without_L, moment_match = T))
```

```
##
## Computed from 4000 by 504 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -100.0 15.5
## p_loo        15.5  3.0
## looic       200.1 31.0
## ------
## Monte Carlo SE of elpd_loo is 0.1.
```

```
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     498   98.8%   1120
##  (0.5, 0.7]   (ok)         6    1.2%   1066
##    (0.7, 1]   (bad)        0    0.0%   <NA>
##    (1, Inf)   (very bad)   0    0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
gdom_L2_weights <- loo_model_weights(list(gdom_plus_L2.loo, gdom_without_L2.loo),
                                     method = 'pseudobma')
gdom_L_weights <- loo_model_weights(list(gdom_without_L2.loo, gdom_without_L.loo),
                                       method = 'pseudobma')
gdom_L_vpc <- variance_decomposition(gdom_without_L2, robust = T, ci = .5)
```

S2.4 Odds and count of cohesion constraints: phonology

In order to model both the odds and the counts, we fit a mixture model that combines a Bernoulli likelihood for the odds of a constraint and a Poisson likelihood for counts above 0.[2] This corresponds to what is known as a hurdle model, where the Bernoulli estimator θ_0 (`hu` in `brms`) captures the log odds of 0s (i.e. the "hurdle" that needs to be crossed for the Poisson estimates to kick in). Since some of the constraints have a sparse distribution, leave-one-out cross-validation is problematic, and its approximation through importance sampling violates the Pareto smoothing assumptions (with a large proportion of $\hat{k} > .7$). In response to this, we resort to the WAIC.

```
contrasts(constraint_counts_p$ConstraintType) <- contr.m(constraint_counts_p$ConstraintType)
contrasts(constraint_counts_p$L2) <- contr.m(constraint_counts_p$L2)

constraint_counts_p$obs <- paste0(constraint_counts_p$Language,
                                  row.names(constraint_counts_p))

pconstr_plus_L2 <- brm(bf(Count ~ ConstraintType + L2 + (1|Language/obs),
                          hu ~ ConstraintType + L2 + (1|Language)),
                       family = hurdle_poisson(),
                       prior = c(priors,
                                 # add priors for hurdle:
                                 prior(student_t(5, 0, 1), class = b, dpar = hu),
                                 prior(cauchy(0, 1), class = sd, dpar = hu)),
                       data = constraint_counts_p,
                       chains = 4, cores = 4,
                       iter = 2000, warmup = 1000,
                       save_all_pars = T, file = 'models/pconstr_plus_L2',
                       control = list(adapt_delta = .999))

pconstr_without_L2 <- update(pconstr_plus_L2,
                             bf(Count ~ ConstraintType + (1|Language/obs),
                                hu ~ ConstraintType + (1|Language)),
                             file = 'models/pconstr_without_L2')

pconstr_without_L <- update(pconstr_plus_L2,
                            bf(Count ~ ConstraintType + (1|obs),
```

[2]See McElreath (2020) for a gentle introduction into mixture models.

```
                                hu ~ ConstraintType),
                             file = 'models/pconstr_without_L')

pconstr_L2_weights <- model_weights(pconstr_plus_L2, pconstr_without_L2, weights = 'waic')
pconstr_L_weights <- model_weights(pconstr_without_L2, pconstr_without_L, weights = 'waic')
pconstr_L_vpc <- variance_decomposition(pconstr_without_L2, re_formula = ~ (1|obs),
                                        robust = T, ci = .5)
pconstr_L_vpc.9 <- variance_decomposition(pconstr_without_L2, re_formula = ~ (1|obs),
                                        robust = T, ci = .9)
```

However, closer inspection of the data suggests that the sparse distribution yields separation in some cases (i.e. some (combinations of) predictor levels predict 0 vs non-0 counts perfectly). Although the models fitted above nevertheless describe the data fairly well (see Figures S2 - S3), the separation effect leads to problems in floating point accuracy in the predicted pointwise probability estimates.[3] A solution is better regularizing priors (Gelman et al. 2008). Trying several distinct priors suggests that they need to be very strong to overcome the separation effects — in fact so strong that they risk influencing the posterior too much. But we fit these models in the spirit of a *sensitivity analysis*:

```
pconstr_plus_L2_restr <- brm(bf(Count ~ ConstraintType + L2 + (1|Language/obs),
                                hu ~ ConstraintType + L2 + (1|Language)),
                             family = hurdle_poisson(),
                             prior = c(prior(normal(0, .5), class = b),
                                       prior(normal(0, .1), class = b, dpar = hu),
                                       prior(exponential(1), class = sd),
                                       prior(exponential(6), class = sd, dpar = hu)),
                             data = constraint_counts_p,
                             chains = 4, cores = 4,
                             iter = 2000, warmup = 1000,
                             save_all_pars = T, file = 'models/pconstr_plus_L2_restr',
                             control = list(adapt_delta = .999))

pconstr_without_L2_restr <- update(pconstr_plus_L2_restr,
                                   bf(Count ~ ConstraintType + (1|Language/obs),
                                      hu ~ ConstraintType + (1|Language)),
                                   file = 'models/pconstr_without_L2_restr')

pconstr_without_L_restr <- update(pconstr_plus_L2_restr,
                                  bf(Count ~ ConstraintType + (1|obs),
                                     hu ~ ConstraintType),
                                  file = 'models/pconstr_without_L_restr')

(pconstr_plus_L2_restr.loo <- loo(pconstr_plus_L2_restr, moment_match = T))
```

```
##
## Computed from 4000 by 36 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo    -51.7  7.2
## p_loo         7.7  1.9
## looic       103.4 14.5
## ------
## Monte Carlo SE of elpd_loo is 0.2.
```

[3] I am grateful to Aki Vehtari for this suggestion (https://discourse.mc-stan.org/t/loo-moment-match-and-reloo-errors/17498, August 18, 2020).

```
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     23    63.9%   1549
##  (0.5, 0.7]   (ok)       13    36.1%   135
##    (0.7, 1]   (bad)       0     0.0%   <NA>
##    (1, Inf)   (very bad)  0     0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
(pconstr_without_L2_restr.loo <- loo(pconstr_without_L2_restr, moment_match = T))
```

```
##
## Computed from 4000 by 36 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo    -51.3  7.0
## p_loo         6.7  1.6
## looic       102.6 14.1
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     27    75.0%   997
##  (0.5, 0.7]   (ok)        9    25.0%   163
##    (0.7, 1]   (bad)       0     0.0%   <NA>
##    (1, Inf)   (very bad)  0     0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
(pconstr_without_L_restr.loo <- loo(pconstr_without_L_restr, moment_match = T))
```

```
##
## Computed from 4000 by 36 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo    -52.2  7.0
## p_loo         5.9  1.2
## looic       104.4 13.9
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     25    69.4%   893
##  (0.5, 0.7]   (ok)       11    30.6%   515
##    (0.7, 1]   (bad)       0     0.0%   <NA>
##    (1, Inf)   (very bad)  0     0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
pconstr_L2_restr_weights <- loo_model_weights(list(pconstr_plus_L2_restr.loo,
                                                   pconstr_without_L2_restr.loo),
                                              method = 'pseudobma')
pconstr_L_restr_weights <- loo_model_weights(list(pconstr_without_L2_restr.loo,
                                                  pconstr_without_L_restr.loo),
                                             method = 'pseudobma')
pconstr_L_restr_vpc <- variance_decomposition(pconstr_without_L2_restr,
                                              re_formula = ~ (1|obs),
                                              robust = T, ci = .5)
pconstr_L_restr_vpc.9 <- variance_decomposition(pconstr_without_L2_restr,
                                                re_formula = ~ (1|obs),
                                                robust = T, ci = .9)
```

S2.5 Odds and count of cohesion constraints: grammar

In this data set, ‘absence’ (0) values are limited to one language (Mandarin) and so (unlike in the preceding model) we cannot reasonably fit the Bernoulli component as dependent on L2 admixture. Instead, we take the hurdle parameter θ_0 to be identical across predictors. Note that in this case, `brms` uses an identity link for θ_0, i.e. the estimate is reported on the probability scale (p_0), not on the log odds scale.

```
contrasts(constraint_counts_g$ConstraintType) <- contr.m(constraint_counts_g$ConstraintType)
contrasts(constraint_counts_g$L2) <- contr.m(constraint_counts_g$L2)

constraint_counts_g$obs <- paste0(constraint_counts_g$Language,
                                  row.names(constraint_counts_g))

gconstr_plus_L2 <- brm(Count ~ ConstraintType + L2 + (1|Language/obs),
                       family = hurdle_poisson(),
                       prior = priors,
                       data = constraint_counts_g,
                       chains = 4, cores = 4,
                       iter = 2000, warmup = 1000,
                       save_all_pars = T, file = 'models/gconstr_plus_L2',
                       control = list(adapt_delta = .999))

gconstr_without_L2 <- update(gconstr_plus_L2, ~ . - L2, file = 'models/gconstr_without_L2')
gconstr_without_L <- update(gconstr_plus_L2, ~ ConstraintType + (1|obs),
                            file = 'models/gconstr_without_L')

(gconstr_plus_L2.loo <- loo(gconstr_plus_L2, moment_match = T))
```

```
##
## Computed from 4000 by 16 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo    -20.6 4.4
## p_loo         3.4 0.8
## looic        41.3 8.7
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     7     43.8%   1903
##  (0.5, 0.7]   (ok)       9     56.2%   688
##    (0.7, 1]   (bad)      0      0.0%   <NA>
##    (1, Inf)   (very bad) 0      0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
(gconstr_without_L2.loo <- loo(gconstr_without_L2, moment_match = T))
```

```
##
## Computed from 4000 by 16 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo    -20.1 4.2
## p_loo         3.6 1.0
## looic        40.1 8.5
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     11    68.8%   754
##  (0.5, 0.7]   (ok)        5    31.2%   418
##    (0.7, 1]   (bad)       0     0.0%   <NA>
##    (1, Inf)   (very bad)  0     0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
(gconstr_without_L.loo <- loo(gconstr_without_L, moment_match = T))
```

```
##
## Computed from 4000 by 16 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo    -19.4 4.0
## p_loo         2.8 0.7
## looic        38.7 8.0
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     8     50.0%   1765
##  (0.5, 0.7]   (ok)       8     50.0%   965
##    (0.7, 1]   (bad)      0      0.0%   <NA>
##    (1, Inf)   (very bad) 0      0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
gconstr_L2_weights <- loo_model_weights(list(gconstr_plus_L2.loo, gconstr_without_L2.loo),
                                        method = 'pseudobma')
gconstr_L_weights <- loo_model_weights(list(gconstr_without_L2.loo, gconstr_without_L.loo),
                                       method = 'pseudobma')
gconstr_L_vpc <- variance_decomposition(gconstr_without_L2, re_formula = ~ (1|obs),
                                        robust = T, ci = .5)
```

S2.6 Domain Size: phonology

Here, 0 values are impossible since a domain must minimally include the verb stem. Therefore, we truncate the Poisson distribution with a lower bound of 1. The data show again the same sparsity problem as for the phonological constraints hurdle models. We resort to the WAIC again.

```
psize <- droplevels(subset(domain_size, Level  %in% 'phonology'))
contrasts(psize$ConstraintType) <- contr.m(psize$ConstraintType)
contrasts(psize$L2) <- contr.m(psize$L2)

psize$obs <- paste0(psize$Language, row.names(psize))

psize_plus_L2 <- brm(Size |  trunc(lb = 1) ~ ConstraintType + L2 + (1|Language/obs),
                    family = poisson(),
                    prior = priors,
                    data = psize,
                    chains = 4, cores = 4,
                    iter = 2000, warmup = 1000,
                    save_all_pars = T, file = 'models/psize_plus_L2',
                    control = list(adapt_delta = .999))

psize_without_L2 <- update(psize_plus_L2, ~ . - L2, file = 'models/psize_without_L2')
psize_without_L <- update(psize_plus_L2, ~ ConstraintType + (1|obs),
                          file = 'models/psize_without_L')

psize_L2_weights <- model_weights(psize_plus_L2, psize_without_L2, weights = 'waic')
psize_L_weights <- model_weights(psize_without_L2, psize_without_L, weights = 'waic')
psize_L_vpc <- variance_decomposition(psize_without_L2, re_formula = ~ (1|obs),
                                      robust = T, ci = .5, ntrys = 10)
```

S2.7 Domain Size: grammar

We again truncate the Poisson distribution since sizes cannot be smaller than 1.

```
gsize <- droplevels(subset(domain_size, Level  %in% 'grammar'))
contrasts(gsize$ConstraintType) <- contr.m(gsize$ConstraintType)
contrasts(gsize$L2) <- contr.m(gsize$L2)

gsize$obs <- paste0(gsize$Language, row.names(gsize))

gsize_plus_L2 <- brm(Size | trunc(lb = 1) ~ ConstraintType + L2 + (1|Language/obs),
                     family = poisson(),
                     prior = priors,
                     data = gsize,
                     chains = 4, cores = 4,
                     iter = 2000, warmup = 1000,
                     save_all_pars = T, file = 'models/gsize_plus_L2',
                     control = list(adapt_delta = .999))

gsize_without_L2 <- update(gsize_plus_L2, ~ . - L2, file = 'models/gsize_without_L2')
gsize_without_L <- update(gsize_plus_L2, ~ ConstraintType + (1|obs),
                          file = 'models/gsize_without_L')
```

```r
(gsize_plus_L2.loo <- loo(gsize_plus_L2, moment_match = T))
```

```
##
## Computed from 4000 by 26 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo    -59.4 2.6
## p_loo         6.7 0.9
## looic       118.9 5.2
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     12    46.2%   1286
##  (0.5, 0.7]   (ok)       14    53.8%   510
##    (0.7, 1]   (bad)       0     0.0%   <NA>
##    (1, Inf)   (very bad)  0     0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```r
(gsize_without_L2.loo <- loo(gsize_without_L2, moment_match = T))
```

```
##
## Computed from 4000 by 26 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo    -58.7 2.6
## p_loo         5.7 0.8
## looic       117.4 5.3
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     15    57.7%   869
##  (0.5, 0.7]   (ok)       11    42.3%   741
##    (0.7, 1]   (bad)       0     0.0%   <NA>
##    (1, Inf)   (very bad)  0     0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```r
(gsize_without_L.loo <- loo(gsize_without_L, moment_match = T))
```

```
##
## Computed from 4000 by 26 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo    -58.0 2.8
## p_loo         5.1 0.9
## looic       115.9 5.6
## ------
## Monte Carlo SE of elpd_loo is 0.1.
```

```
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     15    57.7%   1362
##  (0.5, 0.7]   (ok)       11    42.3%   712
##    (0.7, 1]   (bad)       0     0.0%   <NA>
##    (1, Inf)   (very bad)  0     0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
gsize_L2_weights <- loo_model_weights(list(gsize_plus_L2.loo, gsize_without_L2.loo),
                                      method = 'pseudobma')
gsize_L_weights <- loo_model_weights(list(gsize_without_L2.loo, gsize_without_L.loo),
                                     method = 'pseudobma')
gsize_L_vpc <- variance_decomposition(gsize_without_L2, re_formula = ~ (1|obs),
                                      robust = T, ci = .5)
```

S2.8 Synthesis: odds of inflectional categories

In this data set, there is a substantial L2 effect and so we keep this factor when fitting models without varying intercepts.

```
contrasts(synthesis.long$Category) <- contr.m(synthesis.long$Category)
contrasts(synthesis.long$L2) <- contr.m(synthesis.long$L2)

syn_plus_L2 <- brm(Presence ~ Category + L2 + (1|Language),
                   family = bernoulli(),
                   prior = priors,
                   data = synthesis.long,
                   chains = 4, cores = 4,
                   iter = 2000, warmup = 1000,
                   save_all_pars = T, file = 'models/syn_plus_L2',
                   control = list(adapt_delta = .9))

syn_without_L2 <- update(syn_plus_L2, ~ . - L2, file = 'models/syn_without_L2')
syn_without_L <- update(syn_plus_L2, ~ . - (1|Language), file = 'models/syn_without_L')

(syn_plus_L2.loo <- loo(syn_plus_L2, moment_match = T))
```

```
##
## Computed from 4000 by 36 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo    -23.6 3.2
## p_loo         6.7 1.2
## looic        47.1 6.3
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     31    86.1%   1183
##  (0.5, 0.7]   (ok)        5    13.9%   687
##    (0.7, 1]   (bad)       0     0.0%   <NA>
##    (1, Inf)   (very bad)  0     0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```r
(syn_without_L2.loo <- loo(syn_without_L2, moment_match = T))
```

```
##
## Computed from 4000 by 36 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo    -24.6 3.2
## p_loo         7.2 1.2
## looic        49.2 6.4
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     32    88.9%   782
##  (0.5, 0.7]   (ok)        4    11.1%   999
##    (0.7, 1]   (bad)       0     0.0%   <NA>
##    (1, Inf)   (very bad)  0     0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```r
(syn_without_L.loo <- loo(syn_without_L, moment_match = T))
```

```
##
## Computed from 4000 by 36 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo    -22.7 2.8
## p_loo         5.2 0.9
## looic        45.3 5.7
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     35    97.2%   2847
##  (0.5, 0.7]   (ok)        1     2.8%   1157
##    (0.7, 1]   (bad)       0     0.0%   <NA>
##    (1, Inf)   (very bad)  0     0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```r
syn_L2_weights <- loo_model_weights(list(syn_plus_L2.loo, syn_without_L2.loo),
                                    method = 'pseudobma')
syn_L_weights <- loo_model_weights(list(syn_plus_L2.loo, syn_without_L.loo),
                                   method = 'pseudobma')
names(syn_L_weights) <-  c('With Varying Intercepts', 'Without Varying Intercepts')
syn_L_vpc <- variance_decomposition(syn_plus_L2, robust = T, ci = .5)
```

S3 Model evaluation

All models converged well, with no divergent transitions, $\hat{R} = 1$, and large effective sample sizes. See the `models` directory at the repository of the present file https://osf.io/mt98r for details and model summaries.[4] The $\text{elpd}_{\text{psis+}}$ estimates all had Pareto $\hat{k} < .7$, suggesting that assumptions are met and there are no overly influential observations. For models involving phonological constraints (`pconstr.*` and `psize.*`), we used WAIC approximations because of data sparsity. The $\text{elpd}_{\text{waic}}$ estimates come with high variance, and so these results need to be interpreted with caution.

S4 Comparisons

S4.1 L2 effects

The phonological constraints model (`pconstr_plus_L2`) estimates effects both on the log counts (μ or `mu`) and on the log odds (θ_0 or `hu`). We collect the results separately. Note that the log odds in this model are the log odds of 0s, i.e. *absence*, not the odds of presence. In order to make the estimates comparable to the log odds estimates in the other variables, we inverse their sign. For the grammatical constraints model (`gconstr_plus_L2`), we estimated θ_0 independent of the predictors. `brms` reports this on the response scale (identity link). The result is very low, with a posterior median probability of 0s, $\hat{p}_0 = 0.096$, that captures the data well.

```r
# posterior samples:
L2posterior <- bind_rows(lapply(L2model_list,
              # first, extract the effects on mu (odds or counts):
              function(m) { df <- data.frame(posterior_samples(m, "b_L22"),
                                             posterior_samples(m, "r_Language\\["))
                            colnames(df) <- c("Estimate",
                                              "Bunan", "Burmese", "Chintang", "Mandarin")
                            return(df)
                          }
              ), .id = "Variable") %>%
              # second, extract the effects on hu (presence in the hurdle model):
  rbind(., setNames(x <- data.frame(Variable = rep("Constraint Odds (Phonology)",
                                              nrow(posterior_samples(pconstr_plus_L2))),
                    # make the L2 effect comparable to the log odds of presence
                    # like in the other Bernoulli estimates
                    - posterior_samples(pconstr_plus_L2, "b_hu_L22"),
                    - posterior_samples(pconstr_plus_L2, "r_Language__hu"),
                    stringsAsFactors = F),
                 nm = c("Variable", "Estimate",
                        "Bunan", "Burmese", "Chintang", "Mandarin"))
  )

# label and order correctly:
L2posterior$Variable[
  L2posterior$Variable %in% "Constraint Odds/Count (Phonology)"] <-
  "Constraint Count (Phonology)"
L2posterior$Variable <- factor(L2posterior$Variable,
                               levels = c(variables[1:3], "Constraint Count (Phonology)",
                                          "Constraint Odds (Phonology)", variables[5:8]))
```

Next, we bind together the model weights; see Table S1 and the color shading in Figure 2 of the main paper.

[4] DOI `10.17605/OSF.IO/MT98R`

```r
L2_weights <- cbind(Variable = levels(L2posterior$Variable), rbind(
  as.data.frame(t(matrix(mtyp_L2_weights, dimnames = list(c('With', 'Without'))))),
  as.data.frame(t(matrix(pdom_L2_weights, dimnames = list(c('With', 'Without'))))),
  as.data.frame(t(matrix(gdom_L2_weights, dimnames = list(c('With', 'Without'))))),
  as.data.frame(t(matrix(pconstr_L2_weights, dimnames = list(c('With', 'Without'))))),
  # we duplicate this entry because it holds for both the count and the odds in the plot:
  as.data.frame(t(matrix(pconstr_L2_weights, dimnames = list(c('With', 'Without'))))),
  as.data.frame(t(matrix(gconstr_L2_weights, dimnames = list(c('With', 'Without'))))),
  as.data.frame(t(matrix(psize_L2_weights, dimnames = list(c('With', 'Without'))))),
  as.data.frame(t(matrix(gsize_L2_weights, dimnames = list(c('With', 'Without'))))),
  as.data.frame(t(matrix(syn_L2_weights, dimnames = list(c('With', 'Without')))))
  ))
```

The sensitivity analysis of the phonological constraints model is fully in line with these results although the strong priors of course privilege null models and need to be interpreted cautiously. The estimated effect of the L2 factor is $\hat{\theta_1}$ = -0.02, 50%CI = [-0.09, 0.05] on the odds and $\hat{\mu}$ = -0.04, 50%CI = [-0.34, 0.26] on the counts. The model with the L2 factor leverages less weight than the model without the L2 factor (bootstrapped Aikake weight = 0.4).

Table S1: Comparison of predictive performance through Akaike weights

Variable	With L2 factor	Without L2 factor
Morpheme Type Odds	0.425	0.575
Domain Odds (Phonology)	0.428	0.572
Domain Odds (Grammar)	0.311	0.689
Constraint Odds/Count (Phonology)	0.454	0.546
Constraint Odds/Count (Grammar)	0.365	0.635
Domain Size (Phonology)	0.417	0.583
Domain Size (Grammar)	0.328	0.672
Category Odds (Synthesis)	0.731	0.269

For synthesis, we furthermore assess the L2 effect more closely. The probability mass of L2 log odds below 0 is:

```r
round(mean(posterior_samples(syn_plus_L2)$b_L22 < 0),2)
```

```
## [1] 0.94
```

Median log odds and CI:

```r
(overall <- L2posterior[L2posterior$Variable %in% 'Category Odds (Synthesis)',] %>%
  summarise(Median = median(Estimate),
            CI = paste0("[", round(qi(Estimate, .width = .90)[1],2), ", ",
                             round(qi(Estimate, .width = .90)[2],2), "]")))
```

Median	CI
-1.512556	[-3.12, 0.13]

Median odds ratio:

```r
round(median(exp(posterior_samples(syn_plus_L2)$b_L22)), 2)
```

```
## [1] 0.22
```

Figure S1 gives an overview of the estimated variation by language.

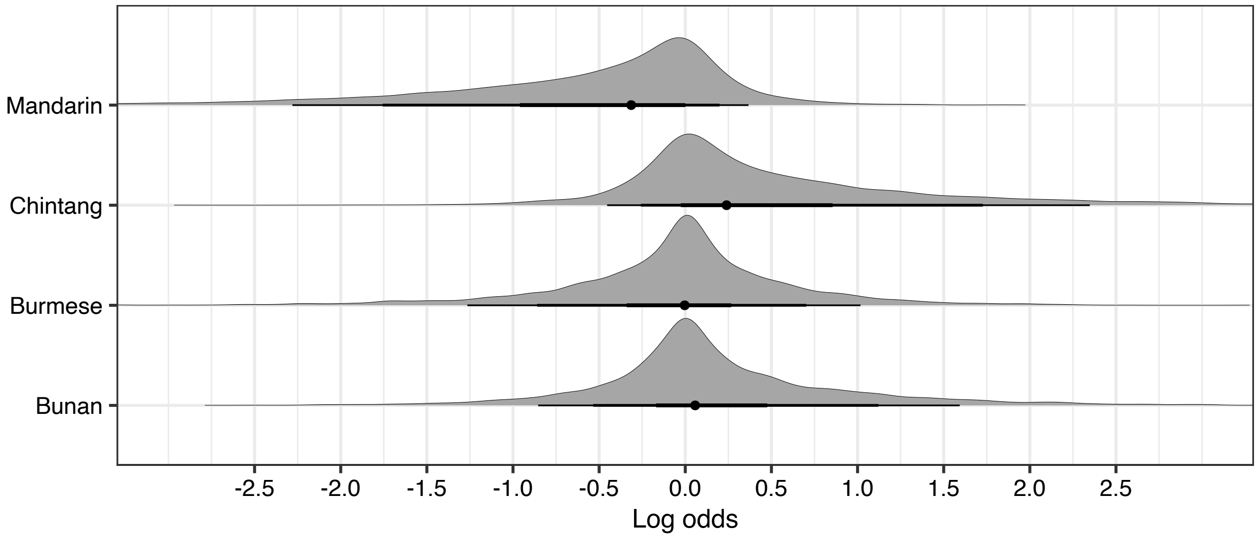

Figure S1: Estimated variation of the grand mean in a model of synthesis with an L2 factor. Black horizontal bars indicate the 90%, 80% and 50% CIs around the median (decreasing bar widths). The variation by language is modest compared to the overall L2 effect (cf. Median = -1.51, 90%CI = [-3.12, 0.13])

Consistent with the estimates in Figure S1, we find that models without varying intercepts have better predictive performance than models with varying intercepts:

```
syn_L_weights
```

```
## Method: pseudo-BMA+ with Bayesian bootstrap
## ------
##                             weight
## With Varying Intercepts    0.299
## Without Varying Intercepts 0.701
```

Also, even the 50% CI of the VPS includes 0:

```
syn_L_vpc
```

```
## # Random Effect Variances and ICC
##
## Conditioned on: all random effects
##
## ## Variance Ratio (comparable to ICC)
## Ratio: 0.00  CI 50%: [-0.03 0.07]
##
## ## Variances of Posterior Predicted Distribution
## Conditioned on fixed effects: 0.24  CI 50%: [0.22 0.25]
## Conditioned on rand. effects: 0.25  CI 50%: [0.23 0.25]
##
## ## Difference in Variances
## Difference: 0.00  CI 50%: [-0.01 0.02]
```

S4.2 Null-effect models

For the variables with evidence against an L2 effect (i.e. all except the synthesis model), we also collect the posterior estimates of the grand mean (the intercept), so we can inspect the spread of the estimates and the variation by language (varying intercepts). The results are show in Figure 3 of the main paper and tabulated in Table S2 below.

```
Lposterior <- bind_rows(lapply(Lmodel_list,
                                # first, extract the effects on mu (presence or counts):
                                function(m) { df <- data.frame(posterior_samples(m, "b_Intercept"),
                                                               posterior_samples(m, "r_Language\\["))
                                colnames(df) <- c("Estimate",
                                                  "Bunan", "Burmese", "Chintang", "Mandarin")
                                return(df)
                                }
  ), .id = "Variable") %>%
  # second, extract the effects on hu (odds of phon. constraints in the hurdle model):
  rbind(., setNames(data.frame(id = rep("Constraint Odds (Phonology)",
                                        nrow(posterior_samples(pconstr_without_L2))),
                               # make the L2 effect comparable to the log odds of presence
                               # like in the other bernoulli estimates:
                               -posterior_samples(pconstr_without_L2, "b_hu_Intercept"),
                               -posterior_samples(pconstr_without_L2, "r_Language__hu"),
                               stringsAsFactors = F),
                    nm = c("Variable", "Estimate",
                           "Bunan", "Burmese", "Chintang", "Mandarin"))
  ) %>%
  # third, we compute the language-specific means/intercepts:
```

```r
  mutate(EstBunan = Estimate + Bunan,
         EstBurmese = Estimate + Burmese,
         EstChintang = Estimate + Chintang,
         EstMandarin = Estimate + Mandarin)

# label and order correctly:
Lposterior$Variable[Lposterior$Variable %in% "Constraint Odds/Count (Phonology)"] <-
  "Constraint Count (Phonology)"
Lposterior$Variable <- factor(Lposterior$Variable,
                              levels = c(variables[1:3], "Constraint Count (Phonology)",
                                         "Constraint Odds (Phonology)",
                                         variables[5:7]))


# reshaping for plotting (main text figure)
Lposterior.long <- gather(
  Lposterior[, c('Variable', 'EstBunan', 'EstBurmese', 'EstChintang', 'EstMandarin')],
  "Language", "Estimate", EstBunan:EstMandarin) %>%
  mutate(Language = factor(gsub('Est', '', Language),
           levels = c('Bunan', 'Chintang', 'Burmese', 'Mandarin')))

# median estimates:
L_rnd_medians <- group_by(Lposterior.long, Variable, Language) %>%
  summarize(m = median(Estimate))
```

Table S2: Intercept and language-specific deviations from it. CI = 90% (quantile-based) credible interval. Note that in the case of grammatical constraints, we estimate a global hurdle estimate (θ). The median posterior estimate for the presence of some constraint is $\theta_1 = 0.906$, with 90%CI = [0.75, 0.98]

Variable	Intercept Median	Intercept CI	Language	Median	CI
Morpheme Type Odds	1.06	[0.11, 2.01]	Bunan	0.00	[-0.75, 0.95]
			Burmese	0.00	[-0.77, 0.96]
			Chintang	0.11	[-0.48, 1.37]
			Mandarin	-0.07	[-1.18, 0.58]
Domain Odds (Phonology)	-3.47	[-4.03, -2.89]	Bunan	-0.06	[-0.81, 0.37]
			Burmese	-0.11	[-0.97, 0.29]
			Chintang	-0.02	[-0.69, 0.48]
			Mandarin	0.16	[-0.24, 1.01]
Domain Odds (Grammar)	-3.46	[-3.96, -2.95]	Bunan	-0.01	[-0.6, 0.35]
			Burmese	0.02	[-0.39, 0.52]
			Chintang	0.00	[-0.47, 0.45]
			Mandarin	-0.02	[-0.59, 0.36]
Constraint Count (Phonology)	-0.11	[-1.39, 0.73]	Bunan	-0.20	[-1.39, 0.52]
			Burmese	-0.16	[-1.38, 0.65]
			Chintang	0.36	[-0.29, 1.84]
			Mandarin	0.18	[-0.54, 1.6]
Constraint Odds (Phonology)	-0.22	[-1.03, 0.62]	Bunan	0.27	[-0.31, 1.64]
			Burmese	0.00	[-0.95, 0.91]
			Chintang	0.00	[-0.95, 0.88]
			Mandarin	-0.28	[-1.72, 0.32]
Constraint Odds/Count (Grammar)	-0.49	[-1.47, 0.28]	Bunan	-0.03	[-0.8, 0.47]
			Burmese	0.02	[-0.51, 0.76]
			Chintang	0.00	[-0.71, 0.59]
			Mandarin	0.00	[-0.75, 0.56]
Domain Size (Phonology)	0.60	[-0.06, 1.14]	Bunan	0.03	[-0.4, 0.68]
			Burmese	0.08	[-0.28, 0.9]
			Chintang	-0.09	[-0.83, 0.29]
			Mandarin	-0.01	[-0.62, 0.56]
Domain Size (Grammar)	1.70	[1.43, 1.96]	Bunan	-0.03	[-0.4, 0.18]
			Burmese	0.05	[-0.15, 0.42]
			Chintang	0.00	[-0.27, 0.29]
			Mandarin	-0.02	[-0.36, 0.22]

These results suggest that the variation by language is minimal, except for the phonological constraints data. This is confirmed by the results from model performance comparison and from variance partitioning:

```
L_comp <- data.frame(Variable = variables[-8],
                     L_weight = c(mtyp_L_weights[2],
                                  pdom_L_weights[2],
                                  gdom_L_weights[2],
                                  pconstr_L_weights[2],
                                  gconstr_L_weights[2],
                                  psize_L_weights[2],
                                  gsize_L_weights[2]),
                     L_VPCCI = c(paste0("[", round(mtyp_L_vpc$ICC_CI[1],2), ", ",
                                        round(mtyp_L_vpc$ICC_CI[2],2), "]"),
                                 paste0("[", round(pdom_L_vpc$ICC_CI[1],2), ", ",
                                        round(pdom_L_vpc$ICC_CI[2],2), "]"),
                                 paste0("[", round(gdom_L_vpc$ICC_CI[1],2), ", ",
                                        round(gdom_L_vpc$ICC_CI[2],2), "]"),
                                 paste0("[", round(pconstr_L_vpc$ICC_CI[1],2), ", ",
                                        round(pconstr_L_vpc$ICC_CI[2],2), "]"),
                                 paste0("[", round(gconstr_L_vpc$ICC_CI[1],2), ", ",
                                        round(gconstr_L_vpc$ICC_CI[2],2), "]"),
                                 paste0("[", round(psize_L_vpc$ICC_CI[1],2), ", ",
                                        round(psize_L_vpc$ICC_CI[2],2), "]"),
                                 paste0("[", round(gsize_L_vpc$ICC_CI[1],2), ", ",
                                        round(gsize_L_vpc$ICC_CI[2],2), "]")))
```

With the exception of phonological constraints again, models without varying intercepts by language consistently leverage more weights (0.59 - 0.71) than models with varying intercepts (Table S3). In line with this, the VPC estimates always include 0 even in their 50% credibility interval by a fair margin.

Table S3: Akaike weight of models without varying intercepts and the 50%CI of the variance partition coefficient (VPC)

Variable	Weight	VPC
Morpheme Type Odds	0.673	[-0.15, 0.17]
Domain Odds (Phonology)	0.588	[-0.29, 0.17]
Domain Odds (Grammar)	0.710	[-0.23, 0.19]
Constraint Odds/Count (Phonology)	0.354	[-0.46, 0.32]
Constraint Odds/Count (Grammar)	0.666	[-0.68, 0.42]
Domain Size (Phonology)	0.666	[-0.35, 0.26]
Domain Size (Grammar)	0.669	[-0.33, 0.24]

The exceptional status of the phonological constraints model is confirmed by the sensitivity analysis. Despite the strong priors against large intercept variation in the Bernoulli component ($Exp(6)$), we find that the hurdle model as a whole leverages *less* weight without than with the varying intercept, with bootstrapped Akaike weight = 0.346. This is the same estimate as the WAIC estimate for model with weaker priors in Table S3. The source of this is both the Bernoulli and the Poisson component (Table S2), although the model with stronger priors (unsurprisingly) reduces the variation in the Bernoulli component and shifts the signal to the Poisson component (Table S4).

```
fixef_restr <- data.frame(
  Variable = c(rep('Count', nrow(posterior_samples(pconstr_without_L2_restr))),
               rep('Odds', nrow(posterior_samples(pconstr_without_L2_restr)))),
  Intercept = c(posterior_samples(pconstr_without_L2_restr)$"b_Intercept",
                -posterior_samples(pconstr_without_L2_restr)$"b_hu_Intercept"))
```

```r
lgs <- c("Bunan", "Burmese", "Chintang", "Mandarin")

ranef_restr <- data.frame(
  Variable = c(rep('Count', nrow(posterior_samples(pconstr_without_L2_restr))),
               rep('Odds', nrow(posterior_samples(pconstr_without_L2_restr)))),
  rbind(setNames(posterior_samples(pconstr_without_L2_restr, "r_Language\\["), nm = lgs),
        setNames(-posterior_samples(pconstr_without_L2_restr, "r_Language__hu"), nm = lgs)
  ))

grands_restr <- group_by(fixef_restr, Variable) %>%
  summarise(`Intercept Median` = median(Intercept),
            `Intercept CI` = paste0("[", round(qi(Intercept, .width = .90)[1],2), ", ",
                                    round(qi(Intercept, .width = .90)[2],2), "]")
  ) %>% slice(rep(1:n(), each = 4))

ranef_summary_restr <- gather(ranef_restr, "Language", "Estimate", Bunan:Mandarin) %>%
  group_by(Variable, Language) %>%
  summarise(Median = median(Estimate),
            CI = paste0("[", round(qi(Estimate, .width = .90)[1],2), ", ",
                        round(qi(Estimate, .width = .90)[2],2), "]"))
```

Table S4: Sensitivity analysis: Intercept and language-specific deviations from it in the phonological constraints model with stronger priors. CI = 90% (quantile-based) credible interval.

Variable	Intercept Median	Intercept CI	Language	Median	CI
Count	0.22	[-0.89, 0.93]	Bunan	-0.25	[-1.36, 0.51]
			Burmese	-0.18	[-1.41, 0.6]
			Chintang	0.43	[-0.21, 1.77]
			Mandarin	0.25	[-0.49, 1.57]
Odds	-0.22	[-0.81, 0.35]	Bunan	0.02	[-0.17, 0.48]
			Burmese	0.00	[-0.3, 0.31]
			Chintang	0.00	[-0.29, 0.29]
			Mandarin	-0.02	[-0.51, 0.2]

However, these results do not support strong conclusions. The actual elpd estimates do not in fact differ much between models with and models without varying intercepts, with overlapping standard errors (Table S5), similar in fact to what we observe in the other models (see Sections S2.1 - S2.7). This is true under both weaker and stronger priors. Additional doubts come from the variance partition (VPC) estimates. Both the model with weaker and the model with stronger priors include 0 in their 50%CI VPC: [-0.46, 0.32] and [-0.39, 0.29], respectively, so the predicted variance between languages seems to be in a similar ballpark as the total variance.

These observations are further confirmed by posterior predictive checks (Gabry et al. 2019). There is no appreciable difference in model fit between models with (Figure S2) and models without (Figure S3) varying intercepts, and this also true for the phonological constraints model. There are only few 90%CIs that fail to include the observed data.[5]

[5]The example in the main text is computed via `log(mean(gsize$Size))` and `group_by(gsize, Language) %>% summarize(log(mean(Size)))`

Table S5: Expected log pointwise predictive density (elpd) estimates for the phonological constraints models. Note that for the models with weaker priors, these estimates are based on the WAIC, but for those with stronger priors, they are based on Pareto-smootheed importance sampling with moment matching (PSIS+; cf. Section S2.4)

Priors	Models	elpd	SE
Weak	With varying intercepts by language	-50.1	7.08
	Without varying intercepts by language	-50.7	7.03
Strong	With varying intercepts by language	-51.3	7.05
	Without varying intercepts by language	-52.2	6.95

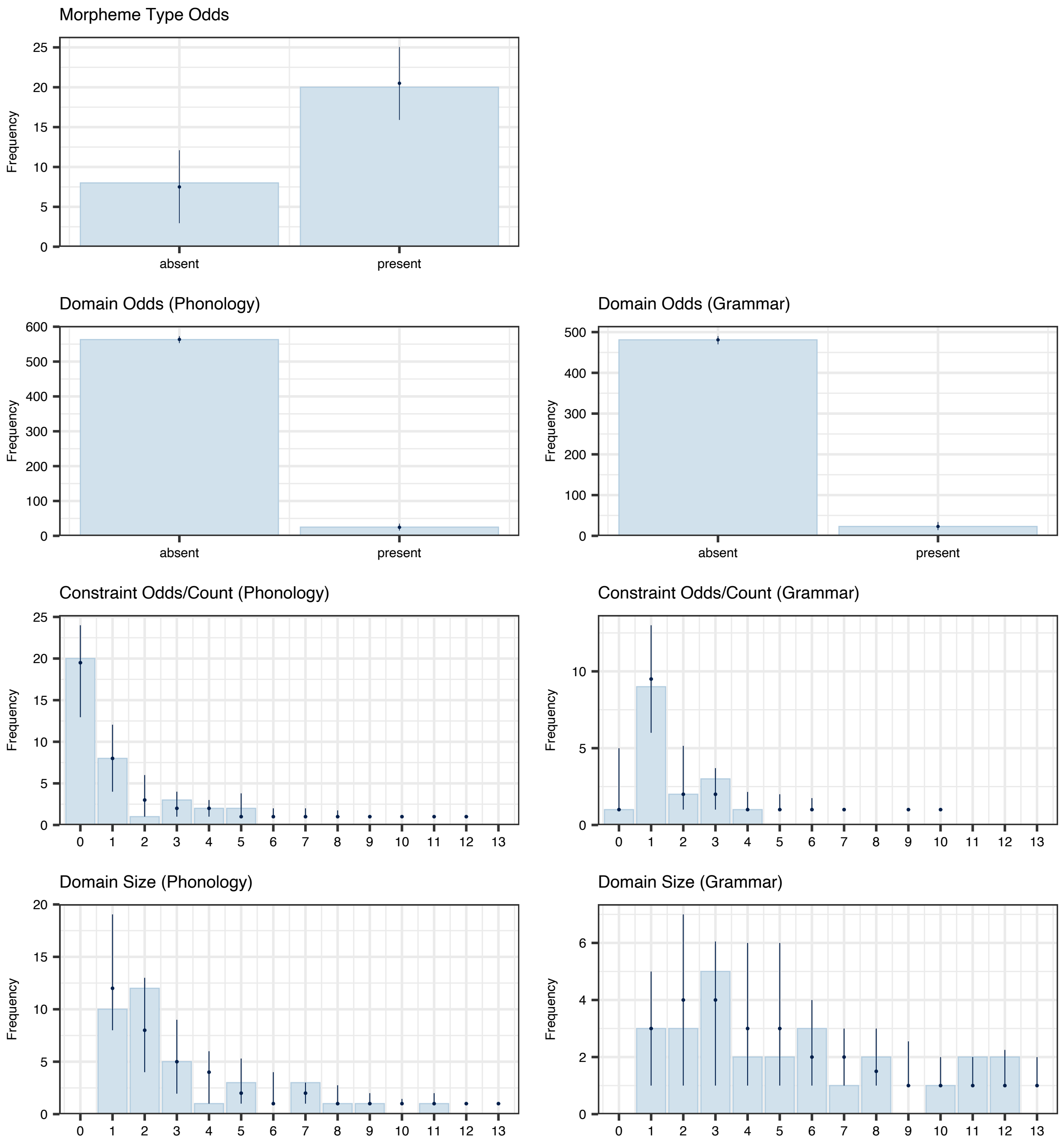

Figure S2: Posterior predictive checks of models without the L2 factor. The blue bars represent the frequencies of presence/absence and the counts/sizes in the observed data. The black dots are the median posterior predictions and the lines capture their 90% CI.

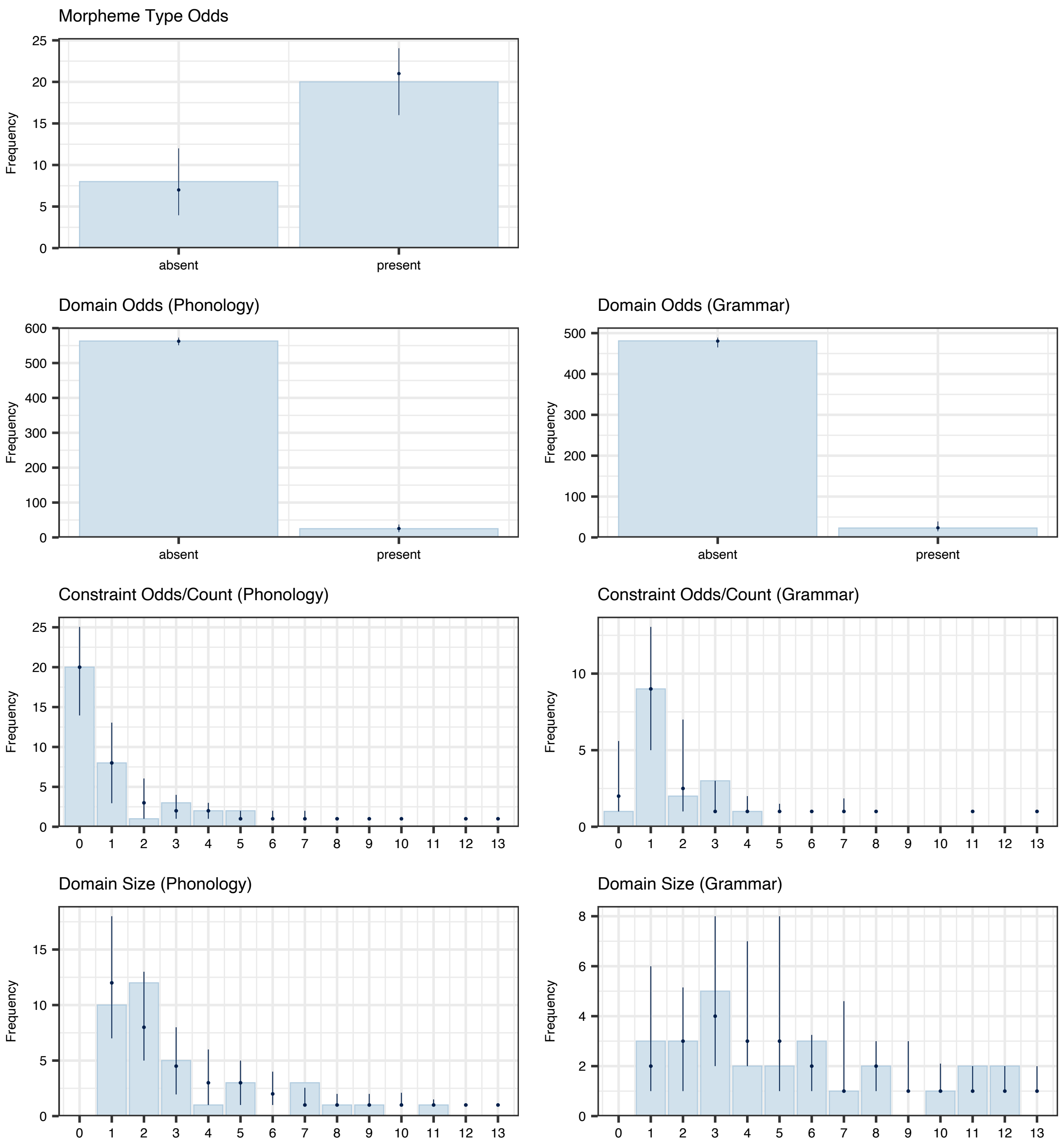

Figure S3: Posterior predictive checks of models without the L2 factor and without varying intercepts. Same plotting conventions as in Figure S2.

References

Bürkner, Paul-Christian. 2018. Advanced Bayesian Multilevel Modeling with the R Package `brms`. *The R Journal* 10. 395–411.

Carpenter, Bob, Andrew Gelman, Matthew D. Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li & Allen Riddell. 2017. Stan: A Probabilistic Programming Language. *Journal of Statistical Software* 76.

Gabry, Jonah, Daniel Simpson, Aki Vehtari, Michael Betancourt & Andrew Gelman. 2019. Visualization in Bayesian Workflow. *J. R. Stat. Soc. A* 182. 389–402.

Gelman, Andrew, Aleks Jakulin, Maria Grazia Pittau & Yu-Sung Su. 2008. A Weakly Informative Default Prior Distribution for Logistic and Other Regression Models. *The Annals of Applied Statistics* 2. 1360–1383.

Harrison, Xavier A. 2014. Using Observation-Level Random Effects to Model Overdispersion in Count Data in Ecology and Evolution. *PeerJ* 2. e616.

Lüdecke, Daniel, Dominique Makowski, Philip Waggoner & Indrajeet Patil. 2020. `performance`: Assessment of Regression Models Performance.. R Package, https://easystats.github.io/performance.

McElreath, Richard. 2020. *Statistical Rethinking (Second Edition).* Boca Raton, FL: CRC Press.

Paananen, Topi, Juho Piironen, Paul-Christian Bürkner & Aki Vehtari. 2019. Implicitly Adaptive Importance Sampling.. *ArXiv* [https://arxiv.org/pdf/1906.08850.pdf].

Vehtari, Aki, Andrew Gelman & Jonah Gabry. 2016. Practical Bayesian Model Evaluation Using Leave-One-Out Cross-Validation and WAIC. *Statistics and Computing* 27(5). 1413–1432.

Yao, Y, A Vehtari, D. Simpson & A. Gelman. 2018. Using Stacking to Average Bayesian Predictive Distributions. *Bayesian Analysis* 10. 10.1214/17–BA1091.