Appendix 2

A.T. Tredennick, A.R. Kleinhesselink, & P.B. Adler

"Ecosystem and community resistance..."

PeerJ

Section A2.1 Random Slopes, Random Intercepts Model

Section A2.1.1 Model Description

Our hierarchical Bayesian model with random intercepts and random slopes allowed us to test for differences among treatments in the relationship between ANPP and growing season precipitation, and allowed us to account for the non-independence of observations through time within a plot. The random effects structure mirrors the nested structure of the data: plot-level parameters are nested within treatment-level parameters, which are nested within overall parameters. In what follows, $\mathbf{X}$ is the design matrix including a column of 1s for intercepts and a column of precipitation values for each observation, $\mathbf{y}$ is a row vector of log(ANPP) values for each observation, i references observations, j references plots, k references treatment, and t references year. The notation *i(j(k(t)))* reads as 'observation i associated with plot j associated with treatment k within year t'. Recall we have three treatments: drought (~50% decrease in precipitation), control (ambient precipitation), and irrigation (~50% increase in precipitation).

We assume the observations are conditionally Gaussian,

$$y_{i(j(k(t)))} \sim \text{Normal}\left(\mu_{i(j(k(t)))}, \sigma_k^2\right), \tag{A2.1}$$

where $\mu_{i(j(k(t)))}$ is the determinstic expectation from the regression model,

$$\mu_{i(j(k(t)))} = \mathbf{x}_i' \boldsymbol{\beta}_{\boldsymbol{j(k)}} + \boldsymbol{\gamma}_{\boldsymbol{t}}. \tag{A2.2}$$

The intercept and slope in the parameter vector β were modeled hierarchically,

$$\boldsymbol{\beta}_{j(k)} \sim \text{MVN}\left(\boldsymbol{\beta}_k, \boldsymbol{\Sigma}(k)\right), \tag{A2.3}$$

$$\boldsymbol{\beta}_k \sim \text{MVN}\left(\boldsymbol{\beta}, \boldsymbol{\Sigma}\right), \tag{A2.4}$$

$$\boldsymbol{\beta} \sim \text{Normal}\left(0, 1\right), \tag{A2.5}$$

where $\beta_{j(k)}$ is the vector of regression coefficients (intercept and slope) for plot j associated with treatment k, β_k is the vector of coefficients for each treatment, and β is the vector of overall coefficients. The plot- and treatment-level coefficients are drawn from multivariate normal

distributions with covariance matrix $\mathbf{\Sigma}$. For the plot-level coefficients, each treatment has its own variance-covariance matrix (i.e., $\mathbf{\Sigma}(k)$). The overall coefficients are drawn from a normal prior with mean 0 and standard deviation 1.

The random year effects (γ) are modeled as intercept offsets centered on zero with a shared variance (σ_{yr}),

$$\gamma \sim \text{Normal}\left(0, \sigma_{yr}\right). \tag{A2.6}$$

We fit the model using MCMC as implemented in the software Stan (Stan Development Team 2016). Our Stan code is below. All code necessary to reproduce our results is in the supplementary code set (Appendix 3).

Section A2.1.2 Stan Code

```
data {
  int<lower=0> Npreds;         # number of covariates, including intercept
  int<lower=0> Nplots;         # number of plots
  int<lower=0> Ntreats;        # number of treatments
  int<lower=0> Nobs;           # number of observations
  int<lower=0> Nppts;          # Number of precip levels to predict
  int<lower=0> Nyears;         # Number of years
  vector[Nobs] y;              # vector of observations
  row_vector[Npreds] x[Nobs]; # design matrix
  matrix[Nppts,Npreds] newx;  # design matrix for predictions
  matrix[Npreds,Npreds] R;    # priors for covariance matrix
  int plot_id[Nobs];           # vector of plot ids
  int treat_id[Nobs];          # vector of treatment ids
  int year_id[Nobs];           # vector of year ids
}

parameters {
  vector[Npreds] beta_plot[Nplots]; # a unique vector matrix for each plot
  vector[Npreds] beta_treat[Ntreats]; # a unique vector matrix for each treatment
    vector[Npreds] beta_mu; # overall coefficients
    vector[Nyears] year_off; # vector of year effects
    cov_matrix[Npreds] Sigma; # covariance matrix for treatment-level coefficients
    cov_matrix[Npreds] Sigma_plot[Ntreats]; # unique covariance matrix for plot-level coeffici
    vector<lower=0>[Ntreats] sd_y; # treatment-level observation std. dev.
```

```
    vector<lower=0>[Nyears] sigma_year; # year std. dev. hyperprior
}

transformed parameters {
  vector[Nobs] yhat; # vector of expected values (predictions)
  for (i in 1:Nobs)
    # regression model for expected values (one for each plot-year)
    yhat[i] = x[i]*beta_plot[plot_id[i]] + year_off[year_id[i]];
}

model {
  ####  PRIORS
  for(i in 1:Nplots)
    # plot-level coefficients vary normally around treatment coefs
        beta_plot[i] ~ multi_normal(beta_treat[treat_id[i]], Sigma_plot[treat_id[i]]);

    for(i in 1:Ntreats){
      # treatment-level coefficients vary normally around overall coefficients
      beta_treat[i] ~ multi_normal(beta_mu, Sigma);
      Sigma_plot[i] ~ inv_wishart(Npreds+1, R); # priors on covariance of effects at plot-leve
    }

    beta_mu ~ normal(0,1); # priors on overall effects
    year_off ~ normal(0,sigma_year); # priors on year effects, shared variance
    Sigma ~ inv_wishart(Npreds+1, R); # priors on covariance of effects at treatment-level
    sd_y ~ weibull(2,1); # priors on observation std. dev. for each treatment
    sigma_year ~ weibull(2,1); # hyperprior on year std. dev.

    ####  LIKELIHOOD
  for(i in 1:Nobs)
    # observations vary normally around expected values
    y[i] ~ normal(yhat[i], sd_y[treat_id[i]]);
}

generated quantities {
  vector[Nppts] ypreds[Ntreats];
  vector[Nppts] ydiff_control_drought;
  vector[Nppts] ydiff_control_irrigate;
  vector[2] inter_diffs;
```

```
  for(i in 1:Ntreats)
    ypreds[i] = newx*beta_treat[i]; # mean predictions for each treatment

  # difference between mean predictions
  ydiff_control_drought = ypreds[1] - ypreds[2];
  ydiff_control_irrigate = ypreds[1] - ypreds[3];

  # difference between control and treatment in avg ppt year
  inter_diffs[1] = beta_treat[1][1] - beta_treat[2][1]; # control - drought
  inter_diffs[2] = beta_treat[1][1] - beta_treat[3][1]; # control - irrigation
}
```

References

Stan Development Team. 2016. Stan: A C++ Library for Probability and Sampling, Version 2.14.1.