model

Johnson T.F.

The models require approximately 500GB RAM to run, and will take in excess of 14 days. With that in mind, we have hash-tagged out the code to run the models (which are stored in source script) within the Rmarkdown document. We run the models in a variety of scenarios (described below) and within each model we store (for each dataset seperately) the model output (e.g. model summary, predicted values etc.). Each source script is annotated, but the core strucutre is most thoroughogly described here:

```
library(INLA)
library(brinla)

#This script would be ran for each dataset.
message("   Model 1 - Intercept model")

#For all random effects, we set an improper uniform hyper prior.

prior_prec = "expression:
  log_dens = 0 - log(2) - theta / 2;
  return(log_dens);
"

m1 = inla(log_abundance ~ #log abundance is the ressonse
            year_centre + #centred year
            f(site_spec_code,  model = "iid", constr = F, hyper = prior_prec) + #independent random int
            f(tips_code, model = "iid", constr = F, hyper = prior_prec) + #independent random intercept
            f(genus_code, model = "iid", constr = F, hyper = prior_prec) + #independent random intercep
            # Note: INLA does not require explicitly defining a nested structure in the model formula.
            f(site_code, model = "iid", constr = F, hyper = prior_prec) +
            #independent random intercept for each site
            f(region_code, model = "iid", constr = F, hyper = prior_prec), #independent random intercep
            data = analysis_list[[a]][[1]],  #Using the data from 'manipulate.Rmd'
            family = "gaussian", #A gaussian error distribution
            control.predictor=list(compute=TRUE), #Save the predicted values
            control.fixed = list(mean.intercept = 0, prec.intercept = 0.001, #Set a prior on the interc
                                 mean = 0, prec = 1), #Set a prior on the fixed effect coefficient of N
            num.threads = 4) #Number of cores to run the model on. Lots of cores can actually slow thin

message("   Model 2 - Slope model")
m2 = inla(cent_abundance ~ #Each abundance time series is centered now too. Every line passes through z
            year_centre +
            f(site_spec_code, year_centre, model = "iid", constr = F, hyper = prior_prec) + #All random
            f(tips_code, year_centre, model = "iid", constr = F, hyper = prior_prec) +
            f(genus_code, year_centre, model = "iid", constr = F, hyper = prior_prec) +
            f(site_code, year_centre, model = "iid", constr = F, hyper = prior_prec) +
            f(region_code, year_centre, model = "iid", constr = F, hyper = prior_prec),
            data = analysis_list[[a]][[1]],  family = "gaussian",
```

```
            control.predictor=list(compute=TRUE),
            control.fixed = list(mean.intercept = 0, prec.intercept = 0.001,
                                 mean = 0, prec = 1),
            num.threads = 4)

m2_sig = bri.hyperpar.summary(m2)[1,4] #Residual standard deviation
ar_prior = list(theta1 = list(prior="pc.prec", param=c(m2_sig*3, 0.01)), #penalised complexity prior st
                theta2 = list(prior="pc.cor1", param=c(0, 0.9), initial = 0)) #Here we state the odds o

#Need aadditional indicator variables
message("   Model 3 - Correlation model")
  m3 = inla(cent_abundance ~
              year_centre +
              f(year3, model = "ar1", replicate = site_spec_code2, hyper = ar_prior) + #We specify that
              f(site_spec_code, year_centre, model = "iid",
                constr = F, hyper = prior_prec) +
              f(tips_code2, model = "generic0",
                constr = F, Cmatrix = analysis_list[[a]][[2]], hyper = prior_prec) + #We specify that s
              f(tips_code, year_centre, model = "iid",
                constr = F, hyper = prior_prec) +
              f(genus_code, year_centre, model = "iid",
                constr = F, hyper = prior_prec) +
              f(site_code2, year_centre, model = "generic0",
                constr = F, Cmatrix = analysis_list[[a]][[3]], hyper = prior_prec) + #We specify that s
              f(site_code, year_centre, model = "iid",
                constr = F, hyper = prior_prec) +
              f(region_code, year_centre, model = "iid",
                constr = F, hyper = prior_prec),
            data = analysis_list[[a]][[1]], family = "gaussian",
            control.predictor=list(compute=TRUE),
            control.fixed = list(mean.intercept = 0, prec.intercept = 0.001,
                                 mean = 0, prec = 1),
            num.threads = 4)
```

The 'core' models are those presented in the main text

```
source("../code/modelling_core.R")
```

The 'phylogeny' models explore how inference in the correlated effect model changes as you move from the Open Tree of Life phylogeny (which lacks branch lengths) to the TimeTree phylogeny (which has brench lengths, but less species).

```
source("../code/modelling_phylo.R")
```

The 'sensitivity to structure' models explore how the addition of each correlative component (e.g. spatial covariance, phylogenetic covariance, temporal correlation) impact inference (particularly uncertainty around the collective trend)

```
source("../code/modelling_structure.R")
```

The 'abundance' models explore how well each model can predict the final (missing) abundance values within half of the time-series

```
source("../code/modelling_abundance.R")
```

The 'trends' models explore how well each model can predict missing population trends in each dataset

```
source("../code/modelling_trends.R")
```

The 'predict' models are used to produce the site-level trend estimate for BioTIME

```
source("../code/modelling_predict.R")
```

The 'predict2' models are used to generate predictions across space and phylogenies within the BioTIME dataset

```
source("../code/modelling_predict2.R")
```