Appendix III: forest case study

Contents

1 Required packages

In the first step, we load all of the required packages. Note that grainchanger (the package developed for this paper) can be installed using:

```
devtools::install_github("laurajanegraham/grainchanger")
```

Other options in this section set the order and labels for the variables for plotting and tables.

```
library(captioner)
library(jtools)
library(car)
library(perturb)
library(MASS)
library(grainchanger)
library(raster)
library(rgdal)
library(rgeos)
library(cowplot)
library(sf)
library(knitr)
library(GGally)
library(e1071) # optimising transformations (skewness function)
library(broom)
library(MuMIn)
```

```
library(stringr)
library(DHARMa)
library(tidyverse)

source("R/ca_glm.R")

# set up plotting options
theme_set(theme_bw(base_size = 7) + theme(strip.background = element_blank(),
                            panel.grid.major = element_blank(),
                            panel.grid.minor = element_blank()))

varorder <- c("sprich", "winrange500", "elevrange", "elevmean", "bio1", "bio12", "bio15")

varlabel <- c("Species richness", "MW Elevation (500m)", "LS Elevation", "Mean Elevation",
              "Temperature", "Precipitation", "Precip. seasonality")

coefforder <- c("(Intercept)", "winrange500", "elevrange", "elevmean", "elevmean:winvar500",
                "elevmean:elevvar", "bio1", "bio12", "bio15", "I(bio1^2)", "I(bio12^2)")

coefflabel <- c("(Intercept)", "MW elevation\n(500m)", "LS elevation", "Mean elevation",
                "Mean elevation : MW elevation", "Mean elevation : LS elevation",
                "Temperature", "Precipitation", "Precipitation\nseasonality",
                "Temperature\n(quadratic)", "Precipitation\n(quadratic)")

# required for captioning and numbering tables and figures
tabs <- captioner(prefix = "Table AIII.")
figs <- captioner(prefix = "Figure AIII.")
```

2 Data

Note that file paths to data sources are hard coded. These will need updating to match folder structure. A search for `~/` in the document will find these.

2.1 Biological data

First we need to load in and spatialise the EU forest data: available for download from figshare.

```
# we only need this step the first time - once combined with the environmental data the
# output is saved to results/forests_covariates.Rda for further use
forests <- read_csv("~/DATA/BIOLOGICAL/eu_forests/eu_forest_species.csv") %>%
  mutate(sp_name = `SPECIES NAME`) %>%
  dplyr::select(X, Y, sp_name)

eu_crs <- "+proj=laea +lat_0=52 +lon_0=10 +x_0=4321000 +y_0=3210000 +ellps=GRS80 +units=m +nodefs"
forests <- SpatialPointsDataFrame(forests[,1:2], forests[,3],
                                  proj4string = crs(eu_crs))
```

2.2 Environmental data

Analyses will be at 0.5 degree resolution. The data to be upscaled are the elevation data from the European Environment Agency. By cropping all datasets to the extent of the EU elevation data, we end up losing some of the data from the EU forest dataset, such as the records from Northern Africa and the Canaries. We aggregate the elevation to 100m (from 25m) for ease of computation.

We will use the moving window to upscale the variation in elevation using a radius of 500 m. We will also calculate variation in elevation at 0.5 degree resolution. As covariates, we calculated average temperature and precipitation at 0.5 degree resolution.

```
# get climate data
wc_bio <- getData('worldclim', var = "bio", res = 10, path = "~/DATA/CLIMATE/worldclim/")
wc_bio_30 <- aggregate(wc_bio, fact = 3)

# load in the elevation data
elev <- raster("~/DATA/PHYSICAL/elev/eu_dem_1.1/eudem_dem_3035_europe.tif")

# get forest and climate data in same projection as elev and crop worldclim data by the EU
# forest dataset
forests <- spTransform(forests, crs(elev))
wc_bio_30 <- projectRaster(wc_bio_30, crs = crs(elev))

# crop climate data
bounds <- extent(elev)
wc_bio_eu <- crop(wc_bio_30, bounds)

# rasterise the EU forest data to create total species richness (the input function gets
# the total number of unique species in a cell)
forests_r <- rasterize(forests, wc_bio_eu,
                       fun=function(x, ...){
                         length(unique(na.omit(x)))
                       })[[2]]

# create a grid for the moving window and elevation aggregation to loop
eu_grid <- as(forests_r, 'SpatialPolygonsDataFrame')

agg_cell <- function(grid, dat, cell_no, fact, fn="mean", ...) {
  cell <- grid[cell_no,]
  dat_cell <- crop(dat, cell, filename='temp_raster/temp_crop.tif', overwrite=TRUE)
  dat_cell_agg <- aggregate(dat_cell, fact, fun = fn,
                            filename='temp_raster/temp_agg.tif', overwrite=TRUE)
  zero_to_na <- function(x) ifelse(x==0, NA, x)
  dat_cell_na <- raster::calc(dat_cell_agg, zero_to_na, ...)
}

elev_cell_100 <- list()

for(cell_no in 1:length(eu_grid)) {
  fname <- paste0('temp_raster/temp_agg_',cell_no, '.tif'
  elev_cell_100[[cell_no]] <- agg_cell(eu_grid, elev, cell_no, 4,
                                       filename=fname), overwrite=TRUE)
}

# need to add the arguments to mosaic as attributes of the raster list.
```

```
rasters.mosaicargs <- elev_cell_100
names(rasters.mosaicargs) <- NULL
rasters.mosaicargs$fun <- mean
rasters.mosaicargs$filename <- '~/DATA/PHYSICAL/elev/eu_dem/eudem_100m_aggregated.tif'
rasters.mosaicargs$overwrite <- TRUE
rasters.mosaicargs$na.rm <- TRUE

elev_100 <- do.call("mosaic", rasters.mosaicargs)

# create the grid level covariates
eu_grid$elevvar <- nomove_agg(grid = eu_grid, dat = elev_100,
                                  fn = "var", na.rm = TRUE)

eu_grid$elevmean <- nomove_agg(grid = eu_grid, dat = elev_100,
                                   fn = "mean", na.rm = TRUE)

eu_grid$elevrange <- nomove_agg(grid = eu_grid, dat = elev_100,
                                fn = "var_range", na.rm = TRUE)

# upscale the eu elevation data using the moving window approach
strt <- Sys.time()
eu_grid$winvar500 <- winmove_agg(eu_grid, elev_100, 500,
                                         "rectangle", "var")
save(eu_grid, file = "temp.Rda")
run_time <- difftime(Sys.time(), strt, units = "mins")

save(run_time, file = "results/forests_runtime.Rda")

strt <- Sys.time()
eu_grid$winrange500 <- winmove_agg(eu_grid, elev_100, 500,
                                         "rectangle", "var_range")
save(eu_grid, file = "temp.Rda")
range_run_time <- difftime(Sys.time(), strt, units = "mins")

save(range_run_time, file = "results/forests_range_runtime.Rda")

forests_pts <- spTransform(gCentroid(eu_grid, byid=TRUE), crs(elev_100))

forests_sp <- st_as_sf(eu_grid) %>%
  bind_cols(raster::extract(wc_bio_eu, forests_pts) %>% as.tibble) %>%
  rename(sprich = sp_name)

save(forests_sp, file="results/forests_covariates.Rda")
```

Upscaling 100m resolution elevation data for Europe to 0.5 degree resolution using a 500 m radius window took: 479.18 minutes.

2.3 Exploration and transformation

```
load("results/forests_covariates.Rda")
forests_df <- forests_sp %>% as.tibble %>% select(varorder) %>% na.omit %>%
  mutate(bio1 = bio1/10)
```

```
# /10 due to the way worldclim stores temperature data
```

What do the variables look like spatially?

```
forests_narrow <- forests_sp %>%
  select(varorder) %>%
  mutate_at(.vars = vars(-sprich, -geometry), .funs = funs(scale)) %>%
  gather(variable, value, -geometry) %>%
  mutate(facet = "Tree species richness") %>% na.omit

sp_plot <- ggplot(forests_narrow %>% filter(variable == "sprich")) +
  geom_sf(aes(fill = value), colour = NA) +
  coord_sf(crs = st_crs(forests_narrow), datum = NA) +
  scale_fill_viridis_c(name = "", option = "magma") +
  facet_wrap(~facet) +
  theme(axis.text = element_blank(),
        axis.line = element_blank(),
        axis.ticks = element_blank(),
        legend.position = "bottom",
        legend.title.align = 0.5,
        legend.key.height=unit(6,"points"),
        legend.key.width = unit(1.5, "line"),
        panel.border = element_blank())

covs <- forests_narrow %>%
  filter(variable %in% varorder[-1]) %>%
  mutate(variable = factor(variable, levels = varorder, labels = varlabel))

cov_plot <- ggplot(covs) +
  geom_sf(aes(fill = value), colour = NA) +
  coord_sf(crs = st_crs(forests_narrow), datum = NA) +
  scale_fill_viridis_c(name = "") +
  facet_wrap(~variable) +
  theme(axis.text = element_blank(),
        axis.line = element_blank(),
        axis.ticks = element_blank(),
        legend.position = "bottom",
        legend.title.align = 0.5,
        legend.key.height=unit(6,"points"),
        legend.key.width = unit(2, "line"),
        panel.border = element_blank())

plot_grid(sp_plot, cov_plot,
          labels = c("a)", "b)"),
          label_size = 7,
          rel_widths = c(1, 1.4))
```

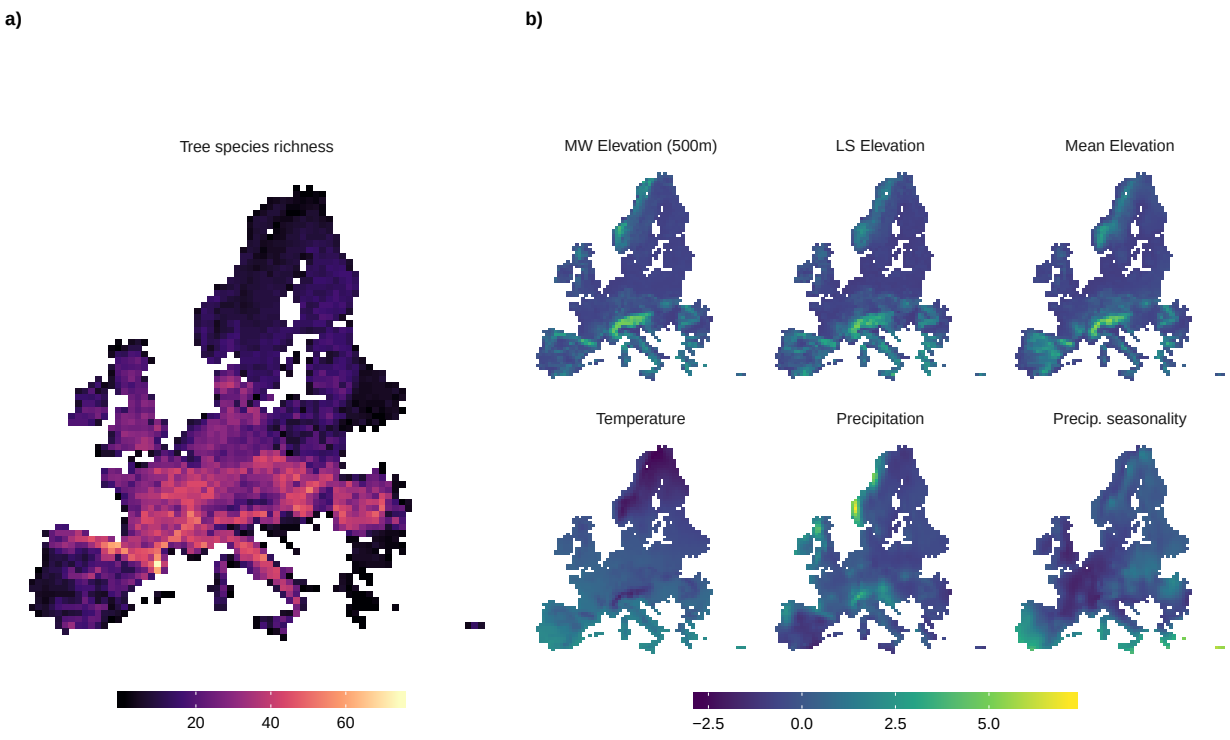

Figure AIII. 1: A case study of the effect of topographic heterogeneity on tree species richness. Study area showing the spatial distribution of (a) tree species richness and (b) scaled covariates (mean = 0; standard deviation = 1).

How are the variables distributed and where are the correlations?

```
ggpairs(
  forests_df %>% select(varorder),
  upper = list(
    continuous = wrap('cor', method = "spearman")
  ),
  lower = list(
    continuous = wrap('points', alpha = 0.3, size =0.1)
  ),
  columnLabels = varlabel
)
```

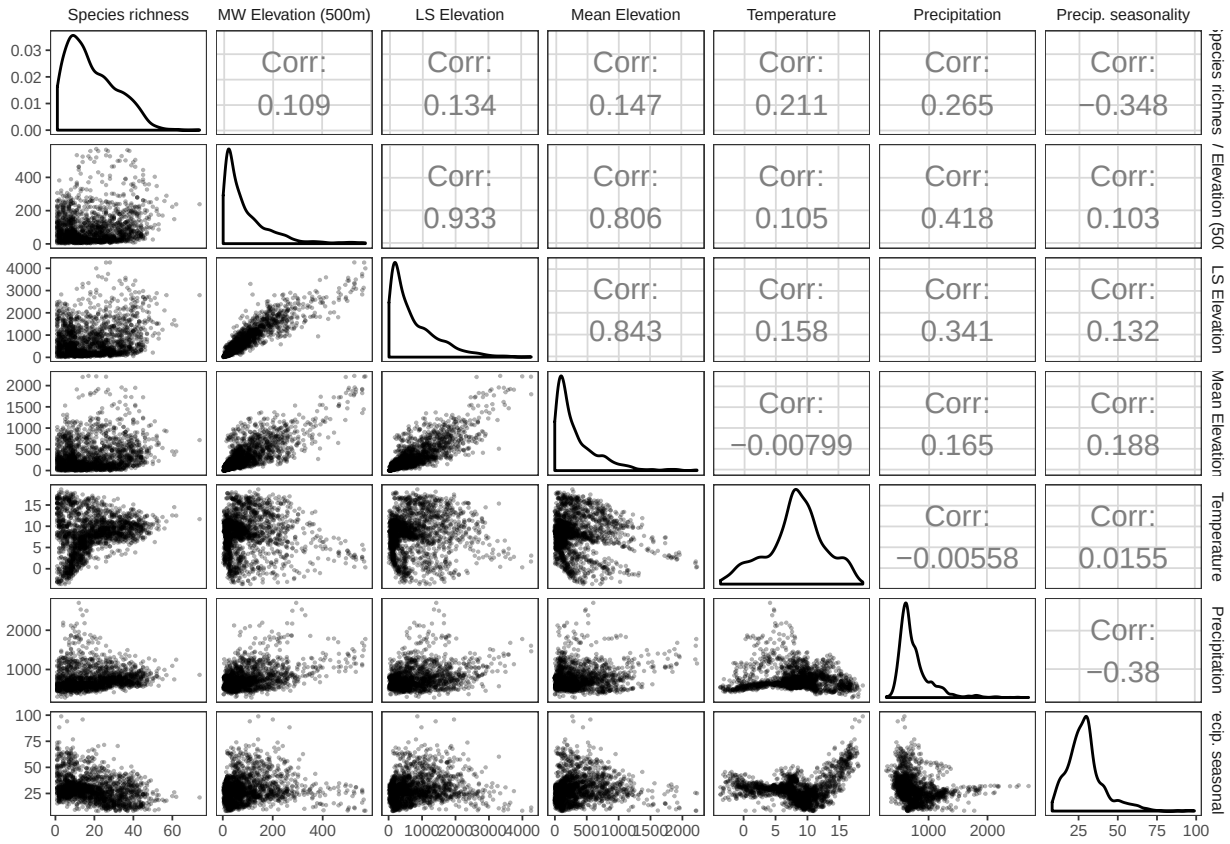

Figure AIII. 2: Pairs plot of *Garrulus glandarius* relative abundance and covariates. Correlations shown are Spearman's ρ.

MW Elevation is the lowest correlation except for total precipitation (bio12). Highest correlation is with precipitation seasonality (bio15). Our expectation that variation in elevation would be important is supported here by the fact that the species richness correlation with mean elevation is lower. Species richness ideal for Poisson distribution; right skew to MW Elevation and LS elevation, Mean elevation, as well as the precipitation variables - need to transform.

```
# optimise to find optimal value for log transform
skew.score <- function(c, x) (skewness(log(x + c)))^2
cs <- tibble(winrange500 = optimise(skew.score, c(0, 20),
                                    x = forests_df$winrange500)$minimum,
             elevrange = optimise(skew.score, c(0, 20),
                                  x = forests_df$elevrange)$minimum,
             elevmean = optimise(skew.score, c(0, 20),
                                 x = forests_df$elevmean)$minimum,
             bio12 = optimise(skew.score, c(0, 20),
                              x = forests_df$bio12)$minimum,
             bio15 = optimise(skew.score, c(0, 20),
                              x = forests_df$bio15)$minimum)


forests_df_t <- select(forests_df, varorder) %>%
  mutate(winrange500 = log(winrange500 + cs$winrange500),
         elevrange = log(elevrange + cs$elevrange),
         elevmean = log(elevmean + cs$elevmean),
```

```
         bio12 = log(bio12 + cs$bio12),
         bio15 = log(bio15 + cs$bio15))

forests_narrow_t <- gather(forests_df_t, variable, value) %>%
  mutate(variable = factor(variable, levels = varorder, labels = varlabel))

# get mean and sd values from the log-transformed data (for back scaling)
means <- forests_df_t %>%
  summarise_all(funs(mean, sd)) %>%
  gather() %>%
  separate(key, into=c("covariate", "measure")) %>%
  spread(measure, value) %>%
  left_join(gather(cs, covariate, c))
```

We scaled the data (mean = 0, SD = 1) so that the partial regression coefficients are comparable.

```
scale_this <- function(x) as.vector(scale(x))
forests_df_t <- mutate_at(forests_df_t, .vars = vars(-sprich), .funs = funs(scale_this))
```

3 Statistical model

Our sample size is n = 1915.

Our model contains the following variables: MW Elevation, LS Elevation, mean elevation, temperature, temperature quadratic term, precipitation, precipitation quadratic term, and precipitation seasonality. We are including the quadratic terms for temperature and precipitation, due to the shape of the relationship between these variables (and based on some earlier residual diagnostics). We also include the interaction term between mean elevation and the MW & LS elevation variables.

The model is fit to a negative binomial distribution due to overdispersion (based on earlier diagnostics).

```
mod_global <- glm.nb(sprich ~ elevmean + winrange500 + elevrange + bio1 +
                       I(bio1^2) + bio12 + I(bio12^2) + bio15,
                     data = forests_df_t, na.action = na.fail)

res_global <- mod_global %>% coef %>% enframe(name = "variable", value = "coef") %>%
  left_join(mod_global %>% confint %>% as_tibble(rownames = "variable")) %>%
  mutate(fvariable = factor(variable, levels = coefforder,
                            labels = gsub("\n", " ", coefflabel))) %>%
  arrange(fvariable)

globalr2 <- 1 - (mod_global$deviance / mod_global$null.deviance)

res_global %>% select(fvariable, coef, `2.5 %`, `97.5 %`) %>% kable(digits=3)
```

fvariable	coef	2.5 %	97.5 %
(Intercept)	3.146	3.110	3.182
MW elevation (500m)	-0.138	-0.216	-0.060
LS elevation	0.201	0.120	0.282
Mean elevation	0.170	0.115	0.225
Temperature	0.131	0.101	0.161
Precipitation	0.048	0.007	0.088
Precipitation seasonality	-0.160	-0.190	-0.131

fvariable	coef	2.5 %	97.5 %
Temperature (quadratic)	-0.253	-0.278	-0.228
Precipitation (quadratic)	-0.078	-0.095	-0.061

Table AIII. 1: Results of negative binomial GLM for the full model

There is a negative effect of local-scale (500 m) variation in elevation (MW Elevation), but a positive effect of landscape-scale (~50km) variation in elevation (LS Elevation).

This model explains 41.22% of the deviance in tree species richness. This was calculated using D-squared.

4 Model validation

Check the model specification using the DHARMa package.

```
simulationOutput <- simulateResiduals(fittedModel = mod_global, n = 250)

plotSimulatedResiduals(simulationOutput)
```

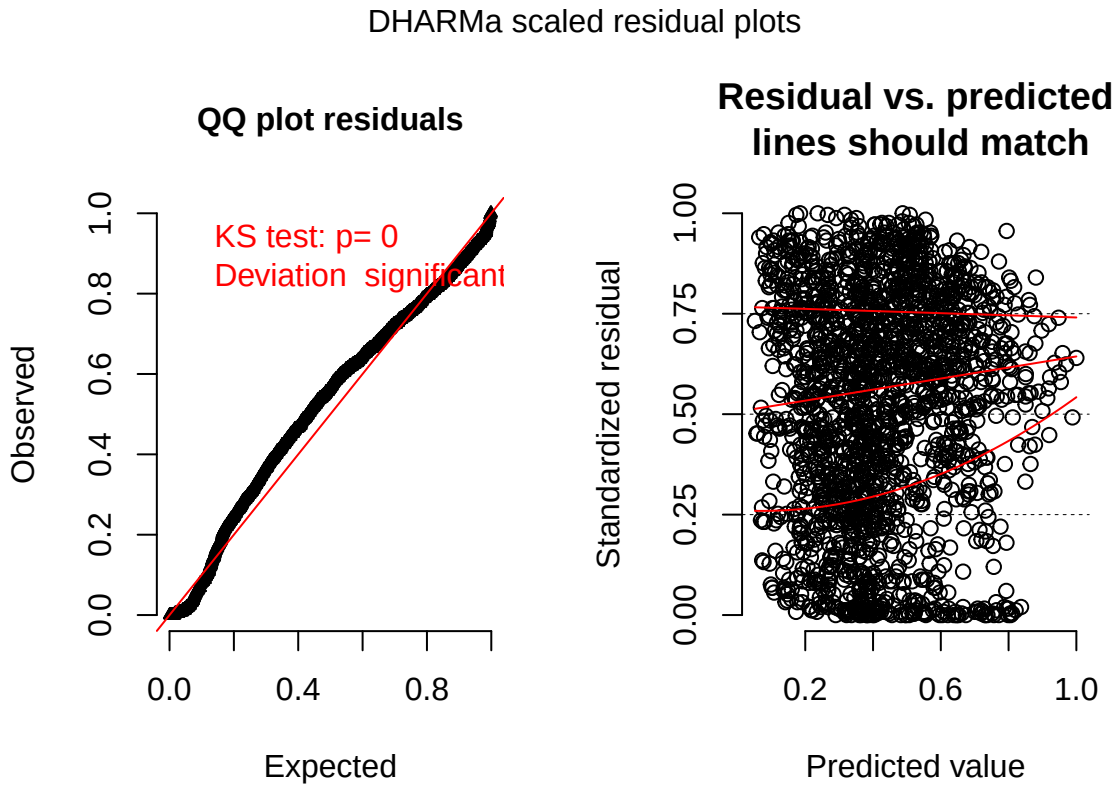

Figure AIII. 3: Residual plots for checking model specification

```
par(mfrow = c(2, 3))
plotResiduals(forests_df_t$winrange500,
              simulationOutput$scaledResiduals,
              xlab = "MW elevation (500m)")
```

```
plotResiduals(forests_df_t$elevrange,
              simulationOutput$scaledResiduals,
              xlab = "LS elevation")

plotResiduals(forests_df_t$elevmean,
              simulationOutput$scaledResiduals,
              xlab = "Mean elevation")

plotResiduals(forests_df_t$bio1,
              simulationOutput$scaledResiduals,
              xlab = "Temperature")

plotResiduals(forests_df_t$bio12,
              simulationOutput$scaledResiduals,
              xlab = "Precipitation")

plotResiduals(forests_df_t$bio15,
              simulationOutput$scaledResiduals,
              xlab = "Precipitation seasonality")
```

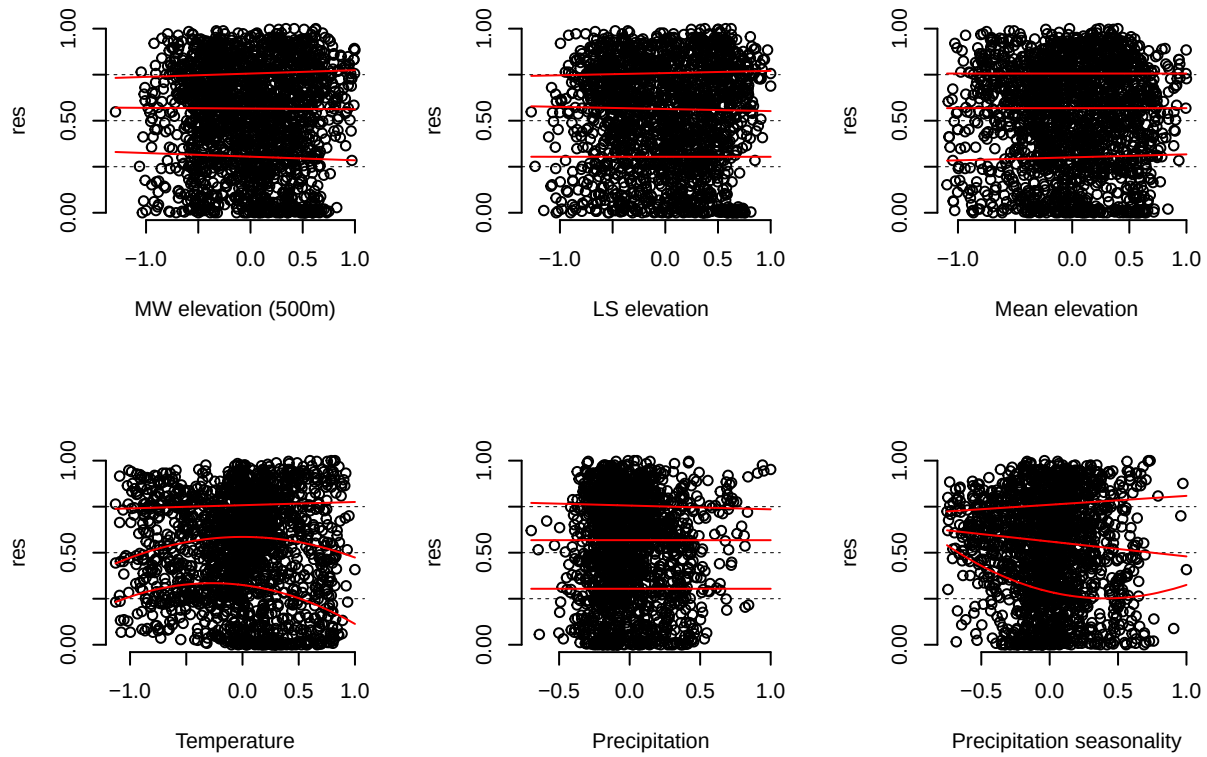

Figure AIII. 4: Residual vs predictor plots for checking model specification

Based on the residual diagnostics, have gone with a negative binomial model due to overdispersion. The earlier version of the diagnostics also found patterns with temperature (bio1) and precipitation (bio12), hence the inclusion of their quadratic terms. The lowest fitted value is 2.27 and the highest is 44.24.

5 Collinearity diagnostics

Due to the high correlation in the data, we will use some in-depth collinearity diagnostics

1. Variance inflation factor (not overly relevant due to large sample size)
2. Condition index and variance decomposition
3. Stability under perturbation analysis
4. Stability under data sub-sampling
5. Collinearity in regression coefficients

5.1 Variance inflation factor

```
vif(mod_global) %>% kable(digits = 2, col.names = c("VIF"))
```

	VIF
elevmean	4.64
winrange500	9.11
elevrange	10.30
bio1	1.28
I(bio1^2)	1.55
bio12	2.35
I(bio12^2)	1.49
bio15	1.43

Table AIII. 2: Variance inflation factors for each term in the global model

Generally low. All below 10 except LS elevation, and even then the large sample size means this is not that large. See O'Brien 2007 for warnings on rule-of-thumb application of multicollinearity diagnostics.

5.2 Condition index and variance decomposition

See Callaghan and Chen 2008 for information on how to interpret condition indices and variance decomposition (but use D. Belsley, E. Kuh, and R. Welsch (1980). Regression Diagnostics. Wiley. as citation). In short, rows in the below table with a high condition index (first column, high is > 10 - moderate to strong collinearity, $>$ 30 - severe collinearity) which are associated with high variance of regression estimate (the rest of the table) will cause a problem in the analysis.

```
mod_diag <- colldiag(mod_global) %>% lapply(as.data.frame)
mod_diag <- do.call("cbind", mod_diag)
names(mod_diag) <- c("CI", "Intercept", "Mean elev", "MW elev", "LS elev",
                     "Temp", "Temp^2", "Precip", "Precip^2", "P season")
kable(mod_diag, digits = 3)
```

CI	Intercept	Mean elev	MW elev	LS elev	Temp	Temp^2	Precip	Precip^2	P season
1.000	0.002	0.016	0.010	0.009	0.000	0.005	0.009	0.012	0.003
1.280	0.081	0.002	0.002	0.001	0.005	0.071	0.010	0.032	0.019
1.431	0.013	0.005	0.000	0.001	0.028	0.007	0.088	0.060	0.144
1.736	0.014	0.010	0.000	0.000	0.584	0.009	0.010	0.020	0.007
2.106	0.108	0.013	0.001	0.002	0.041	0.005	0.071	0.099	0.454
3.124	0.006	0.085	0.007	0.000	0.000	0.214	0.385	0.599	0.011
3.522	0.680	0.047	0.001	0.005	0.079	0.446	0.044	0.161	0.347

CI	Intercept	Mean elev	MW elev	LS elev	Temp	Temp^2	Precip	Precip^2	P season
5.050	0.080	0.748	0.241	0.059	0.202	0.220	0.372	0.015	0.005
7.228	0.016	0.074	0.738	0.922	0.061	0.023	0.012	0.002	0.011

Table AIII. 3: Condition index and variance decomposition

The highest condition index is 7.23, which is in the "weak collinearity" range. Model looks fine by these diagnostics. There is high variance decomposition associated with the MW & LS Elevation coefficient estimates, but given the CI is within "weak collinearity range", this should be okay. We will check this with some tests on the data.

5.3 Stability under perturbation analysis

We will add some random noise to the two environmental heterogeneity measures to evaluate collinearity.

```
attach(forests_df_t)
perturb_mod <- perturb(mod_global,
                       pvars = c("winrange500", "elevrange"),
                       prange=c(0.1,0.1))
detach(forests_df_t)

perturb_mod$coeff.table %>% as.tibble %>%
  gather(variable, value) %>%
  group_by(variable) %>%
  summarise(mean_val = mean(value),
            sd_val = sd(value),
            min_val = min(value),
            max_val = max(value)) %>%
  mutate(variable = factor(variable, levels = coefforder,
                           labels = gsub("\n", " ", coefflabel))) %>%
  arrange(variable) %>%
  kable(digits = 3)
```

variable	mean_val	sd_val	min_val	max_val
(Intercept)	3.145	0.001	3.143	3.147
MW elevation (500m)	-0.113	0.012	-0.143	-0.077
LS elevation	0.173	0.013	0.134	0.207
Mean elevation	0.174	0.004	0.165	0.182
Temperature	0.133	0.001	0.130	0.136
Precipitation	0.046	0.002	0.040	0.051
Precipitation seasonality	-0.161	0.001	-0.162	-0.159
Temperature (quadratic)	-0.253	0.001	-0.254	-0.250
Precipitation (quadratic)	-0.078	0.000	-0.079	-0.077

Table AIII. 4: Results of perturbation analysis. Random noise was added to the two Shannon measures to the range of 0.1

Biggest changes are to the MW & LS values, but this does not change the conclusions and the SD is still reasonably small (SD is ~ 10% of the coefficient estimate).

5.4 Stability under data sub-sampling

Now let's test this by checking the model results are stable under subsampling.

NB This will reduce the power to detect significant relationships (increase Type II errors).

```
model_subsample <-function(x, dat, n) {

  dat <- sample_n(dat, n)

  mod <- glm.nb(sprich ~ elevmean + winrange500 + elevrange + bio1 + I(bio1^2) +
                  bio12 + I(bio12^2) + bio15,
                data = dat, na.action = na.fail)

  res <- mod %>% coef %>% enframe(name = "variable", value = "coef") %>%
    left_join(mod %>% confint %>% as_tibble(rownames = "variable")) %>%
    mutate(fvariable = factor(variable, levels = coefforder,
                              labels = gsub("\n", " ", coefflabel))) %>%
    arrange(fvariable)

  r2 <- 1 - (mod$deviance / mod$null.deviance)

  res %>% select(fvariable, coef, `2.5 %`, `97.5 %`) %>% mutate(D = r2)
}

n <- 1500
nsim <- 100
out <- map_dfr(1:nsim, ~model_subsample(.x, forests_df_t, n))

out_narrow <- out %>% gather(measure, value, -D, -fvariable) %>%
  mutate(measure = factor(measure, labels = c("LCI", "UCI", "Coefficient")),
         measure = factor(measure, levels = c("LCI", "Coefficient", "UCI"))) %>%
  filter(grepl("elevation", fvariable))

res_narrow <- res_global %>%
  filter(variable %in% c("winrange500", "elevrange", "elevmean")) %>%
  select(fvariable, Coefficient = coef, LCI = `2.5 %`, UCI = `97.5 %`) %>%
  gather(measure, value, -fvariable)
```

```
ggplot(out_narrow, aes(x = "fvariable", y = value)) +
  geom_jitter() +
  facet_grid(measure~fvariable) +
  geom_hline(data = res_narrow, aes(yintercept = value), colour = "red") +
  labs(x = "", y = "Coefficient estimate") +
  theme(axis.text.x=element_blank())
```

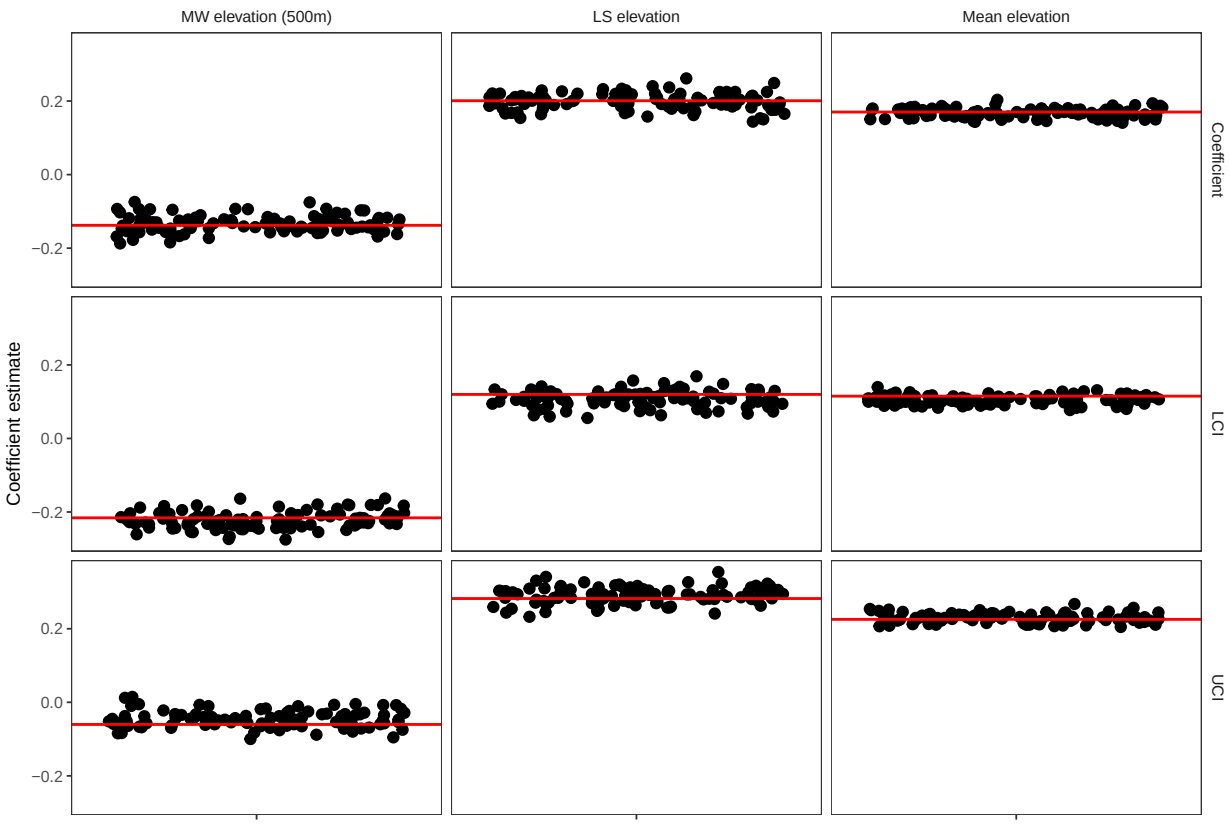

Figure AIII. 5: Results of subsampling analysis. For 100 replicates, 1500 points were subsampled from the total dataset and the regression calculated. Red lines show the value of the coefficient when the regression is calculated using the full dataset

In Figure AIII. 5, each point is the estimate for the model parameterised with one of the 100 samples of n = 1500 from the full dataset. The red line shows the estimate from the model parameterised using the full dataset (n = 1915). The estimates are reasonably robust to sub-sampling the data (not much variation in estimates and very few changes to non-significant - precipitation is the exception here, but this was close to non-significant previously, reducing power has meant this cannot be estimated).

```
ggplot(data = out, aes(x = D)) +
  geom_histogram() +
  geom_vline(xintercept = globalr2)
```

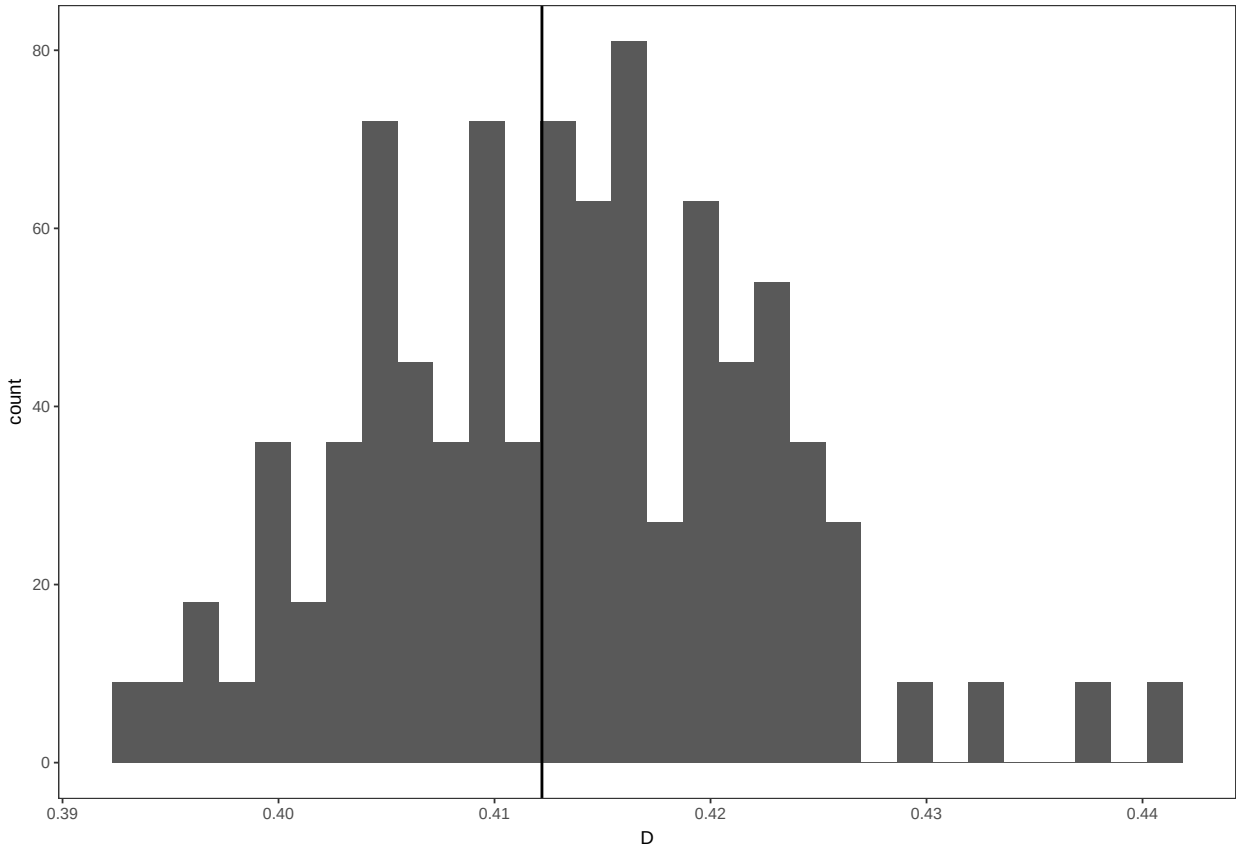

Figure AIII. 6: R^2 distribution for each replicate in the subsampling analysis.

5.5 Collinearity in regression coefficients

Finally, we need to check that we do not have extreme collinearity in the parameter estimates.

```
coef_cov <- cov2cor(vcov(mod_global))

coef_cov%>%
  as.tibble(rownames = "variable") %>%
  kable(digits = 2)
```

variable	(Intercept)	elevmean	winrange500	elevrange	bio1	I(bio1^2)	bio12	I(bio12^2)	bio15
(Intercept)	1.00	-0.12	0.02	0.12	-0.25	-0.58	0.02	-0.38	0.34
elevmean	-0.12	1.00	-0.22	-0.46	0.33	0.18	0.33	-0.03	-0.04
winrange500	0.02	-0.22	1.00	-0.69	0.02	-0.05	-0.35	0.04	-0.07
elevrange	0.12	-0.46	-0.69	1.00	-0.30	-0.18	-0.06	-0.01	0.04
bio1	-0.25	0.33	0.02	-0.30	1.00	0.31	0.18	0.04	-0.19
I(bio1^2)	-0.58	0.18	-0.05	-0.18	0.31	1.00	0.30	-0.13	-0.36
bio12	0.02	0.33	-0.35	-0.06	0.18	0.30	1.00	-0.48	0.24
I(bio12^2)	-0.38	-0.03	0.04	-0.01	0.04	-0.13	-0.48	1.00	-0.15
bio15	0.34	-0.04	-0.07	0.04	-0.19	-0.36	0.24	-0.15	1.00

Table AIII. 5: Pairwise correlations between coefficient estimates from the full model

Correlation between LS elevation and MW elevation (500m), but not super high.

6 Commonality analysis

Although the model does not display multicollinearity despite high correlation between the two Shannon measures, we will apply commonality analysis. This approach allows us to understand the importance of different variables despite any collinearity in the model. See Kraha et al. 2012 and Ray-Mukherjee et al. 2014 for explanation of the approach. For each variable we will calculate:

1. *Beta coefficients* the standardised partial regression coefficients
2. *Structural coefficients* Squared Pearson correlation between the variable and the fitted values from the model
3. *Unique variance* Amount of variance uniquely accounted for by the variable
4. *Common variance* Amount of variance in common with other variables
5. *Total variance* Total amount of variance accounted for by the variable

```
ca_table <- res_global %>%
  filter(fvariable != "Intercept") %>%
  # structural coefficients
  mutate(r_s = map_dbl(variable, function(x) {
    out <- cor(model.matrix(mod_global)[,x], mod_global$fitted.values)
  }),
  r_s2 = r_s^2) %>%
  # unique, common and total variance
  inner_join(calc_commonality(mod_global)$CCTotalbyVar %>%
               as.tibble(rownames = "variable")) %>%
  select(fvariable, beta = coef, `2.5 %`, `97.5 %`, r_s, r_s2, Unique, Common, Total) %>%
  arrange(fvariable)

write_csv(ca_table, "tables/forests_commonality.csv")

kable(ca_table, digits = 3)
```

fvariable	beta	2.5 %	97.5 %	r_s	r_s2	Unique	Common	Total
MW elevation (500m)	-0.138	-0.216	-0.060	0.255	0.065	0.004	0.022	0.026
LS elevation	0.201	0.120	0.282	0.306	0.094	0.007	0.027	0.034
Mean elevation	0.170	0.115	0.225	0.311	0.097	0.011	0.022	0.033
Temperature	0.131	0.101	0.161	0.308	0.095	0.022	0.040	0.062
Precipitation	0.048	0.007	0.088	0.257	0.066	0.002	0.032	0.033
Precipitation seasonality	-0.160	-0.190	-0.131	-0.591	0.349	0.035	0.080	0.115
Temperature (quadratic)	-0.253	-0.278	-0.228	-0.670	0.448	0.116	0.124	0.240
Precipitation (quadratic)	-0.078	-0.095	-0.061	-0.214	0.046	0.022	0.001	0.023

Table AIII. 6: Commonality analysis for the global model

7 Final results and plots

We will now plot the main effects from the full model.

```
# need to predict from the model then convert to the original scales
pred_vals <- bind_rows(
```

```
  make_predictions(mod_global, pred = "elevmean", interval = TRUE)$predicted %>%
    select(sprich, ymax, ymin, pred = elevmean) %>%
    mutate(covariate = "elevmean"),
  make_predictions(mod_global, pred = "elevrange", interval = TRUE)$predicted %>%
    select(sprich, ymax, ymin, pred = elevrange) %>%
    mutate(covariate = "elevrange"),
    make_predictions(mod_global, pred = "winrange500", interval = TRUE)$predicted %>%
    select(sprich, ymax, ymin, pred = winrange500) %>%
    mutate(covariate = "winrange500"),
  make_predictions(mod_global, pred = "bio1", interval = TRUE)$predicted %>%
    select(sprich, ymax, ymin, pred = bio1) %>%
    mutate(covariate = "bio1"),
    make_predictions(mod_global, pred = "bio12", interval = TRUE)$predicted %>%
    select(sprich, ymax, ymin, pred = bio12) %>%
    mutate(covariate = "bio12"),
    make_predictions(mod_global, pred = "bio15", interval = TRUE)$predicted %>%
    select(sprich, ymax, ymin, pred = bio15) %>%
    mutate(covariate = "bio15")) %>%
  as_tibble %>%
  inner_join(means) %>%
  mutate(pred = pred*sd + mean,
         pred = case_when(!is.na(c) ~ exp(pred) - c,
                          TRUE ~ pred))

# plot the results
winrange_plot <- ggplot(pred_vals %>% filter(covariate == "winrange500"),
                        aes(x = pred, y = sprich)) +
  geom_line() +
  geom_ribbon(aes(ymin = ymin, ymax = ymax), alpha = 0.2) +
  scale_y_continuous(limits = c(0, 75), breaks = seq(from = 0, to = 75, by = 15)) +
  labs(x = "MW elevation (500m)",
       y = "")

range_plot <- ggplot(pred_vals %>% filter(covariate == "elevrange"),
                        aes(x = pred, y = sprich)) +
  geom_line() +
  geom_ribbon(aes(ymin = ymin, ymax = ymax), alpha = 0.2) +
  scale_y_continuous(limits = c(0, 75), breaks = seq(from = 0, to = 75, by = 15)) +
  labs(x = "LS elevation",
       y = "")

mean_plot <- ggplot(pred_vals %>% filter(covariate == "elevmean"),
                        aes(x = pred, y = sprich)) +
  geom_line() +
  geom_ribbon(aes(ymin = ymin, ymax = ymax), alpha = 0.2) +
  scale_y_continuous(limits = c(0, 75), breaks = seq(from = 0, to = 75, by = 15)) +
  labs(x = "Mean elevation",
       y = expression("Tree species richness (" %+-% "95% CI)"))

bio1_plot <- ggplot(pred_vals %>% filter(covariate == "bio1"),
                        aes(x = pred, y = sprich)) +
  geom_line() +
  geom_ribbon(aes(ymin = ymin, ymax = ymax), alpha = 0.2) +
```

```
  scale_y_continuous(limits = c(0, 75), breaks = seq(from = 0, to = 75, by = 15)) +
  labs(x = expression("Temperature (" * degree * "C)"),
       y = "")

bio12_plot <- ggplot(pred_vals %>% filter(covariate == "bio12"),
                     aes(x = pred, y = sprich)) +
  geom_line() +
  geom_ribbon(aes(ymin = ymin, ymax = ymax), alpha = 0.2) +
  scale_y_continuous(limits = c(0, 75), breaks = seq(from = 0, to = 75, by = 15)) +
  labs(x = "Total precipitation (mm)",
       y = "")

bio15_plot <- ggplot(pred_vals %>% filter(covariate == "bio15"),
                     aes(x = pred, y = sprich)) +
  geom_line() +
  geom_ribbon(aes(ymin = ymin, ymax = ymax), alpha = 0.2) +
  scale_y_continuous(limits = c(0, 75), breaks = seq(from = 0, to = 75, by = 15)) +
  labs(x = "Precipitation seasonality",
       y = "")

plot_grid(winrange_plot, range_plot, mean_plot,
                    bio1_plot, bio12_plot, bio15_plot,
                    ncol = 2)
```

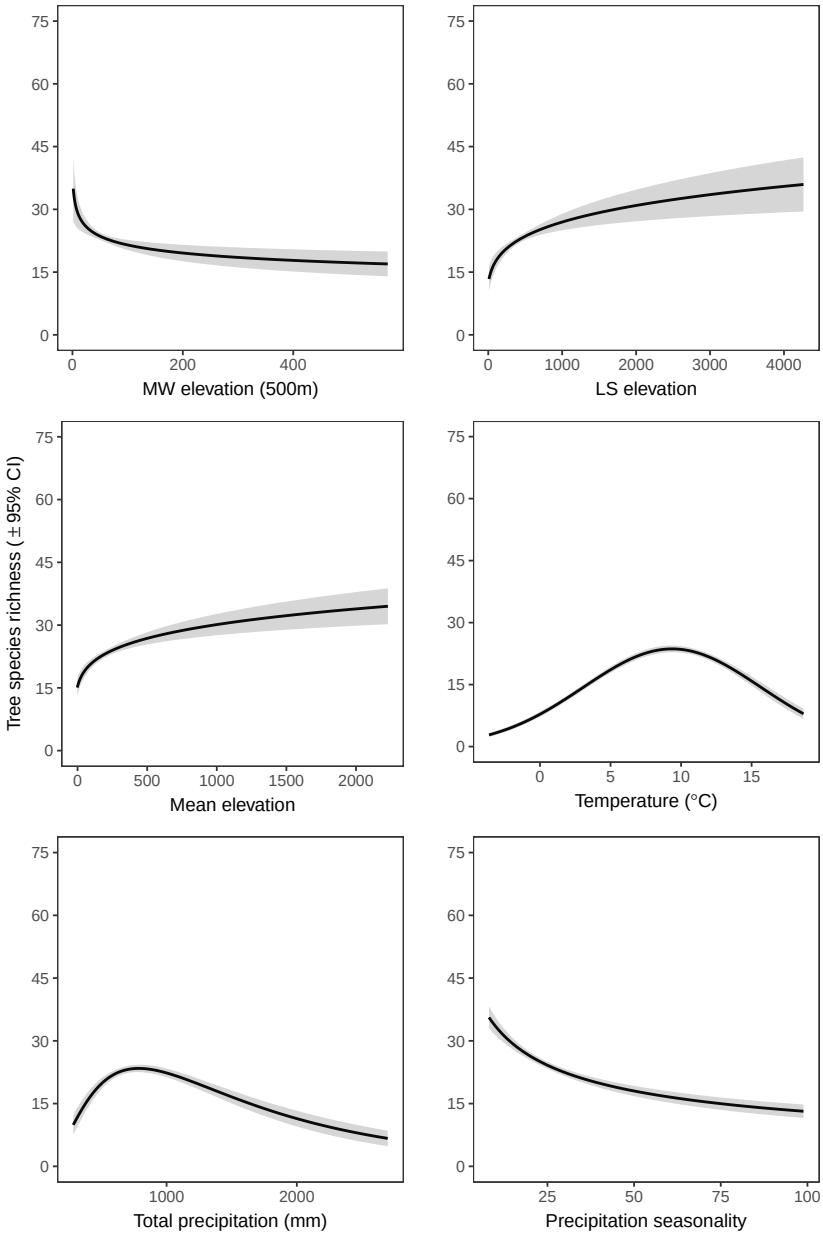

Figure AIII. 7: Main effect estimates for the full model

8 Session Info

```
session <- devtools::session_info()
session[[1]]
```

```
##  setting  value
##  version  R version 3.5.0 (2018-04-23)
##  system   x86_64, mingw32
##  ui       RTerm
##  language (EN)
```

```
##  collate  English_United Kingdom.1252
##  tz       Europe/London
##  date     2018-06-21
```

```
session[[2]] %>% kable
```

package	*	version	date	source
abind		1.4-5	2016-07-21	CRAN (R 3.5.0)
assertthat		0.2.0	2017-04-11	CRAN (R 3.5.0)
backports		1.1.2	2017-12-13	CRAN (R 3.5.0)
base	*	3.5.0	2018-04-23	local
bindr		0.1.1	2018-03-13	CRAN (R 3.5.0)
bindrcpp	*	0.2.2	2018-03-29	CRAN (R 3.5.0)
broom	*	0.4.4	2018-03-29	CRAN (R 3.5.0)
captioner	*	2.2.3.9000	2018-06-19	Github (adletaw/captioner@5f2b435)
car	*	3.0-0	2018-04-02	CRAN (R 3.5.0)
carData	*	3.0-1	2018-03-28	CRAN (R 3.5.0)
cellranger		1.1.0	2016-07-27	CRAN (R 3.5.0)
class		7.3-14	2015-08-30	CRAN (R 3.5.0)
classInt		0.2-3	2018-04-16	CRAN (R 3.5.0)
cli		1.0.0	2017-11-05	CRAN (R 3.5.0)
codetools		0.2-15	2016-10-05	CRAN (R 3.5.0)
colorspace		1.3-2	2016-12-14	CRAN (R 3.5.0)
compiler		3.5.0	2018-04-23	local
cowplot	*	0.9.2	2017-12-17	CRAN (R 3.5.0)
crayon		1.3.4	2017-09-16	CRAN (R 3.5.0)
curl		3.2	2018-03-28	CRAN (R 3.5.0)
data.table		1.11.2	2018-05-08	CRAN (R 3.5.0)
datasets	*	3.5.0	2018-04-23	local
DBI		1.0.0	2018-05-02	CRAN (R 3.5.0)
devtools		1.13.5	2018-02-18	CRAN (R 3.5.0)
DHARMa	*	0.1.6	2018-03-18	CRAN (R 3.5.0)
digest		0.6.15	2018-01-28	CRAN (R 3.5.0)
dplyr	*	0.7.4	2017-09-28	CRAN (R 3.5.0)
e1071	*	1.6-8	2017-02-02	CRAN (R 3.5.0)
evaluate		0.10.1	2017-06-24	CRAN (R 3.5.0)
forcats	*	0.3.0	2018-02-19	CRAN (R 3.5.0)
foreach		1.4.4	2017-12-12	CRAN (R 3.5.0)
foreign		0.8-70	2017-11-28	CRAN (R 3.5.0)
gap		1.1-21	2018-01-24	CRAN (R 3.5.0)
GGally	*	1.4.0	2018-05-17	CRAN (R 3.5.0)
ggplot2	*	2.2.1.9000	2018-05-23	Github (tidyverse/ggplot2@eecc450)
glue		1.2.0	2017-10-29	CRAN (R 3.5.0)
grainchanger	*	0.0.0.9000	2018-06-06	local (laurajanegraham/grainchanger@NA)
graphics	*	3.5.0	2018-04-23	local
grDevices	*	3.5.0	2018-04-23	local
grid		3.5.0	2018-04-23	local
gtable		0.2.0	2016-02-26	CRAN (R 3.5.0)
haven		1.1.1	2018-01-18	CRAN (R 3.5.0)
highr		0.6	2016-05-09	CRAN (R 3.5.0)
hms		0.4.2	2018-03-10	CRAN (R 3.5.0)
htmltools		0.3.6	2017-04-28	CRAN (R 3.5.0)
httr		1.3.1	2017-08-20	CRAN (R 3.5.0)
iterators		1.0.9	2017-12-12	CRAN (R 3.5.0)

package	*	version	date	source
jsonlite		1.5	2017-06-01	CRAN (R 3.5.0)
jtools	*	1.0.0	2018-05-08	CRAN (R 3.5.0)
knitr	*	1.20	2018-02-20	CRAN (R 3.5.0)
labeling		0.3	2014-08-23	CRAN (R 3.5.0)
lattice		0.20-35	2017-03-25	CRAN (R 3.5.0)
lazyeval		0.2.1	2017-10-29	CRAN (R 3.5.0)
lubridate		1.7.4	2018-04-11	CRAN (R 3.5.0)
magrittr		1.5	2014-11-22	CRAN (R 3.5.0)
MASS	*	7.3-49	2018-02-23	CRAN (R 3.5.0)
Matrix		1.2-14	2018-04-13	CRAN (R 3.5.0)
memoise		1.1.0	2017-04-21	CRAN (R 3.5.0)
methods	*	3.5.0	2018-04-23	local
mnormt		1.5-5	2016-10-15	CRAN (R 3.5.0)
modelr		0.1.2	2018-05-11	CRAN (R 3.5.0)
MuMIn	*	1.40.4	2018-01-30	CRAN (R 3.5.0)
munsell		0.4.3	2016-02-13	CRAN (R 3.5.0)
nlme		3.1-137	2018-04-07	CRAN (R 3.5.0)
openxlsx		4.0.17	2017-03-23	CRAN (R 3.5.0)
parallel		3.5.0	2018-04-23	local
perturb	*	2.05	2012-02-19	CRAN (R 3.5.0)
pillar		1.2.2	2018-04-26	CRAN (R 3.5.0)
pkgconfig		2.0.1	2017-03-21	CRAN (R 3.5.0)
plyr		1.8.4	2016-06-08	CRAN (R 3.5.0)
psych		1.8.4	2018-05-06	CRAN (R 3.5.0)
purrr	*	0.2.4	2017-10-18	CRAN (R 3.5.0)
qrnn		2.0.2	2017-12-20	CRAN (R 3.5.0)
R6		2.2.2	2017-06-17	CRAN (R 3.5.0)
raster	*	2.6-7	2017-11-13	CRAN (R 3.5.0)
RColorBrewer		1.1-2	2014-12-07	CRAN (R 3.5.0)
Rcpp		0.12.16	2018-03-13	CRAN (R 3.5.0)
readr	*	1.1.1	2017-05-16	CRAN (R 3.5.0)
readxl		1.1.0	2018-04-20	CRAN (R 3.5.0)
reshape		0.8.7	2017-08-06	CRAN (R 3.5.0)
reshape2		1.4.3	2017-12-11	CRAN (R 3.5.0)
rgdal	*	1.2-20	2018-05-07	CRAN (R 3.5.0)
rgeos	*	0.3-27	2018-06-01	CRAN (R 3.5.0)
rio		0.5.10	2018-03-29	CRAN (R 3.5.0)
rlang		0.2.0	2018-02-20	CRAN (R 3.5.0)
rmarkdown		1.9	2018-03-01	CRAN (R 3.5.0)
RPostgreSQL		0.6-2	2017-06-24	CRAN (R 3.5.0)
rprojroot		1.3-2	2018-01-03	CRAN (R 3.5.0)
rstudioapi		0.7	2017-09-07	CRAN (R 3.5.0)
rvest		0.3.2	2016-06-17	CRAN (R 3.5.0)
scales		0.5.0	2017-08-24	CRAN (R 3.5.0)
sf	*	0.6-2	2018-04-25	CRAN (R 3.5.0)
sp	*	1.2-7	2018-01-19	CRAN (R 3.5.0)
spData		0.2.8.3	2018-03-25	CRAN (R 3.5.0)
stats	*	3.5.0	2018-04-23	local
stats4		3.5.0	2018-04-23	local
stringi		1.1.7	2018-03-12	CRAN (R 3.5.0)
stringr	*	1.3.1	2018-05-10	CRAN (R 3.5.0)
tibble	*	1.4.2	2018-01-22	CRAN (R 3.5.0)

package	*	version	date	source
tidyr	*	0.8.0	2018-01-29	CRAN (R 3.5.0)
tidyselect		0.2.4	2018-02-26	CRAN (R 3.5.0)
tidyverse	*	1.2.1	2017-11-14	CRAN (R 3.5.0)
tools		3.5.0	2018-04-23	local
udunits2		0.13	2016-11-17	CRAN (R 3.5.0)
units		0.5-1	2018-01-08	CRAN (R 3.5.0)
utils	*	3.5.0	2018-04-23	local
viridisLite		0.3.0	2018-02-01	CRAN (R 3.5.0)
withr		2.1.2	2018-05-15	Github (jimhester/withr@79d7b0d)
xml2		1.2.0	2018-01-24	CRAN (R 3.5.0)
yaml		2.1.19	2018-05-01	CRAN (R 3.5.0)