Spatiophylogenetic Modelling Tutorial

Russell Dinnage

2 July 2019

In this tutorial, we will introduce you to the methods used to simultaneously model space and phylogeny when analysing species' traits, which we are calling a spatiophylogenetic model. It doesn't exactly roll off the tongue, we admit, but we wanted to be consistent by analogy with spatiotemporal modelling, which also simultaneously models two sources of potentially confounding covariance in data, in that case, space and time.

The main tricky bit of spatiophylogenetic modelling of species data is that the response is typically measured only at the species level, however, a species spatial distribution is characterized over a large spatial extent, often incorporating many occurrence points. We can still model space in this situation by what we might call spatial averaging, that is, we average any spatial effects across a species' full range to attain a single linear predictor in our regression model. This method uses a Bayesian specification of the model, and we specifically implement it in the R package `INLA` (Rue, Martino, and Chopin 2009; Lindgren, Rue, and Lindström 2011; Martins et al. 2013). Here we will go through how to use the method to model the species level trait of maximum height in a Genus of plant species found throughout Australia, called Hakea. Let's start by loading some required packages and getting our data together.

```
library(INLA)
library(ape)
library(dplyr)
library(readr)
library(ALA4R)
library(sp)
library(ggplot2)
library(ggtree, quietly = TRUE)
library(patchwork)
```

We will need a number of different pieces of data for our analysis. Mainly, we will need data on our species' traits, their phylogenetic relationships, and their spatial distributions, which we load next. These data files will be made available online somewhere when the accompanying paper is published, and I will update this tutorial then to point to them.

```
hakea_traits <- read_csv("data/Hakea_data.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   species = col_character(),
##   phy_name = col_character(),
##   perianth_length_mn = col_number(),
##   flowering_times = col_character(),
##   biome = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
hakea_occ <- read_csv("data/Hakea_occ_clean.csv")
```

```
## Parsed with column specification:
## cols(
##   Species = col_character(),
##   latitude = col_double(),
##   longitude = col_double(),
##   year = col_double(),
##   dup = col_logical(),
##   record_basis = col_character(),
##   data_from = col_character(),
##   ALA_id = col_character(),
##   spat_dup = col_logical(),
##   id = col_double()
## )
```

```
hakea_tree <- read.tree("data/hakea_tree_final.tre")
```

Generally speaking there are two mutually non-exclusive things you may wish to do with a spatiophylogenetic model. One thing you may wish to do is 'account' for covariance in your data created by these known sources of structure, space and phylogeny, while estimating some other relationship. For example, you may want to model the maximum height of your species as a function of say, rainfall within its range, but you are afraid that your response may covary through space or phylogeny, reducing your effective degrees of freedom. That is, you want to try and statistically remove the influence of space and phylogeny, such that you can be a little more sure that any other effects in the model are not unduly driven by spatial or phylogenetic non-independence. This is a valid thing to do, although it should be done carefully, as there is a risk of "throwing the baby out with the bathwater" when doing this. This is a discussion that deserves a lot more attention, but this isn't the place to have it (do read Uyeda et al. 2018 (Uyeda, Zenil-Ferguson, and Pennell 2018) for a great discussion of the issue specifically with respect to phylogeny; to boil it down: think carefully about the causal model you are trying to infer). So, to end that digression, we may want to statistically remove spatial and phylogenetic covariance when modelling something else entirely. The second major thing you may want to do, is to estimate the spatial and phylogenetic structure of your trait in a model-based way. In my opinion, the best thing you can do, is to do both of these things simultaneously. Don't just account for space, and phylogeny, estimate them! And then, visualise them! This will give you a much better understanding of your data and what your model is doing.

Estimating Spatial and Phylogenetic Effects

In this tutorial we will start by just estimating the spatial and phylogenetic effects of our trait of interest, maximum height, to get an idea of the context under which we might ask about what factors could potentially drive it (as an aside: "drive" is just another way of saying "cause", which allows us to avoid actually saying "cause", because we are not allowed to say that word when regression modelling). In this case simultaneously modelling space and phylogeny is advantageous, because it is highly possible that these two things could be confounded with each other. Modelling them at the same time attempts to try and separate as much as possible their independent contributions (that is each of their effects after accounting for the other). What do I mean, they may be confounded? Well, species in the same clade are often found in similar areas for various reasons, including biogeography, and environmental filtering on similar traits. This creates a situation where a clade-level effect could be driven by shared evolutionary history ("phylogenetic constraint"), or due to shared living conditions ("spatial autocorrelation"). It is debatable whether these things are ultimately possible to separate (my leaning is towards "no"), however, we can do our best, or rather, our computers

can do their best. In any event, there is probably at least some independent signal of each factor in a given dataset, and it is our aim to find these signals in this model. Let's start!

The way that we will model space and phylogeny is through a covariance model. That is, we will model our species-level response as an average effect (intercept), plus the sum of two factors, space and phylogeny. These factors we will call "random effects", because they have a hyperdistribution which we will also model. Both factors will be modeled as having been drawn from a multivariate normal distribution with some mean vector and some covariance matrix. Our mean vectors will be fixed to zero, so that they can be considered deviations from the average (the intercept of the model). The magic happens with the two covariance matrices. These will be modeled as a function of the spatial configuration and phylogenetic positions of the species being modeled. Out of the two, the phylogenetic effect is much easier, so we will start by modelling this by itself.

The Phylogenetic Model

Let's get our data in a format where it can be analysed by `INLA`. We will need several different pieces. First we will make a "fixed effect" matrix, with data on our traits and environment for each species (we have 152 *Hakea* species in this analysis). As described in the accompanying paper, the spatial averaging procedure results in the environmental variables simply being modeled as a fixed effect coefficient times the average values of the variable across the range of the species (e.g. the average across all occurrence points). To get this we will first join our trait data to our occurrence data, then resummarise the data back to the species-level. We will make use of two species traits in this tutorial, the maximum height and the number of months the species flowers for.

```
hakea_traits <- hakea_traits %>%
  dplyr::select(species = species, max_height = hieght_mx, flower_months)

hakea_traits_occ <- hakea_occ %>%
  dplyr::select(species = Species, longitude, latitude) %>%
  ## make sure species names use same format as traits
  transform(species = gsub(" ", "_", species)) %>%
  left_join(hakea_traits)
```

```
## Joining, by = "species"
```

```
hakea_traits_occ %>% as_tibble
```

```
## # A tibble: 46,730 x 5
##    species        longitude latitude max_height flower_months
##    <chr>              <dbl>    <dbl>      <dbl>         <dbl>
##  1 Hakea_eriantha      149.    -37.3          5             6
##  2 Hakea_sericea       149     -37.3          4             4
##  3 Hakea_eriantha      149     -37.3          5             6
##  4 Hakea_sericea       149.    -37.5          4             4
##  5 Hakea_sericea       149     -37.5          4             4
##  6 Hakea_sericea       149.    -37.4          4             4
##  7 Hakea_eriantha      149.    -37.4          5             6
##  8 Hakea_eriantha      149     -37.5          5             6
##  9 Hakea_eriantha      149.    -37.5          5             6
## 10 Hakea_eriantha      149.    -37.2          5             6
## # ... with 46,720 more rows
```

We will collect environmental data on rainfall and temperature using the awesome R package `ALA4R`, which interfaces with the Atlas of Living Australia (ALA), an incredible resource for ecologists. I should note that the occurrence points I am using were also downloaded from ALA, but I have manually cleaned them, which is why I do not download them as part of this script.

```
ALA_layers <- c("el874", "el893") ## The ALA ID codes for rainfall and temperature
env_dat <- intersect_points(as.data.frame(hakea_traits_occ[ , c("latitude", "longitude")]),
                            layers = ALA_layers) ## function to get env data from ALA
hakea_traits_occ <- hakea_traits_occ %>%
  ungroup %>%
  mutate(rainfall = env_dat$precipitationAnnualBio12,
         temperature = env_dat$temperatureAnnualMeanBio01)

hakea_traits_occ %>% as_tibble
```

```
## # A tibble: 46,730 x 7
##    species      longitude latitude max_height flower_months rainfall temperature
##    <chr>            <dbl>    <dbl>      <dbl>         <dbl>    <dbl>       <dbl>
##  1 Hakea_erian~      149.    -37.3          5             6      917        12.2
##  2 Hakea_seric~      149     -37.3          4             4     1219         9.7
##  3 Hakea_erian~      149     -37.3          5             6     1219         9.7
##  4 Hakea_seric~      149.    -37.5          4             4     1028        13.8
##  5 Hakea_seric~      149     -37.5          4             4     1063        13.6
##  6 Hakea_seric~      149.    -37.4          4             4      954        13.5
##  7 Hakea_erian~      149.    -37.4          5             6      954        13.5
##  8 Hakea_erian~      149     -37.5          5             6     1063        13.6
##  9 Hakea_erian~      149.    -37.5          5             6     1028        13.8
## 10 Hakea_erian~      149.    -37.2          5             6      829        11.1
## # ... with 46,720 more rows
```

Now we can resummarise that back down to species-level data, taking the mean of the environmental variables, and just the first element for the traits, since they should all be the same for each species.

```
hakea_traits_env <- hakea_traits_occ %>%
  group_by(species) %>%
  summarise(max_height = max_height[1], flower_months = flower_months[1],
            rainfall = mean(rainfall, na.rm = TRUE),
            temperature = mean(temperature, na.rm = TRUE))
hakea_traits_env
```

```
## # A tibble: 152 x 5
##    species             max_height flower_months rainfall temperature
##    <chr>                    <dbl>         <dbl>    <dbl>       <dbl>
##  1 Hakea_actites              5              5    1320.        20.4
##  2 Hakea_aculeata             3              2     356.        17.6
##  3 Hakea_acuminata            1.8            1     434.        16.6
##  4 Hakea_adnata               3.5            3     465.        16.6
##  5 Hakea_aenigma              2.5            3     760.        14.5
##  6 Hakea_ambigua              3              2     535.        14.8
##  7 Hakea_amplexicaulis        3              6     919.        16.4
##  8 Hakea_anadenia             2              4     519.        18.4
##  9 Hakea_arborescens          7              6     844.        26.5
## 10 Hakea_archaeoides          7              3    1405.        17.2
## # ... with 142 more rows
```

In order to fit our phylogenetic model, the only other piece of data we need is the phylogeny. We've already imported the Hakea phylogeny as a `phylo` object using the package `ape`. `INLA` models covariance using "precision" matrices, which are actually the inverse of covariance matrices (it turns out using precision matrices has many benefits in terms of the underlying computations, for more information read the INLA papers cited in the References at the end of this document). Typically phylogenetic covariance is modeled by assuming the residuals of our regression are drawn from a multivariate Gaussian distribution, with means equal to zero and a covariance matrix constructed from the phylogeny. Assuming a Brownian motion model of evolution, this entries of covariance matrix are simply proportional to the amount of shared branch length between each pair of species on the phylogeny. First we will calculate the phylogenetic covariance matrix from our phylogeny. To then use it in `INLA` we will invert it, to form the 'phylogenetic precision matrix'. We will then use `INLA` to include a phylogenetic 'random effect' in our model, which will have means of zero and the equivalent of a covariance matrix equal to the phylogenetic covariance matrix multiplied by some scaling factor, which will be estimated in the model. Note that we will also standardize the phylogenetic covariance matrix to make it comparable to other models using different phylogenies (we aren't doing that here, but it is good to try and standardise as much as possible). See this article for details on the standardisation: https://cran.r-project.org/web/packages/pez/vignettes/pez-pglmm-overview.pdf.

```
phylo_covar_mat <- ape::vcv(hakea_tree)
phylo_covar_mat <- phylo_covar_mat/max(phylo_covar_mat)
phylo_covar_mat <- phylo_covar_mat / exp(determinant(phylo_covar_mat)$modulus[1] /
                                         nrow(phylo_covar_mat))

phylo_prec_mat <- solve(phylo_covar_mat)
phylo_prec_mat[1:5, 1:5]
```

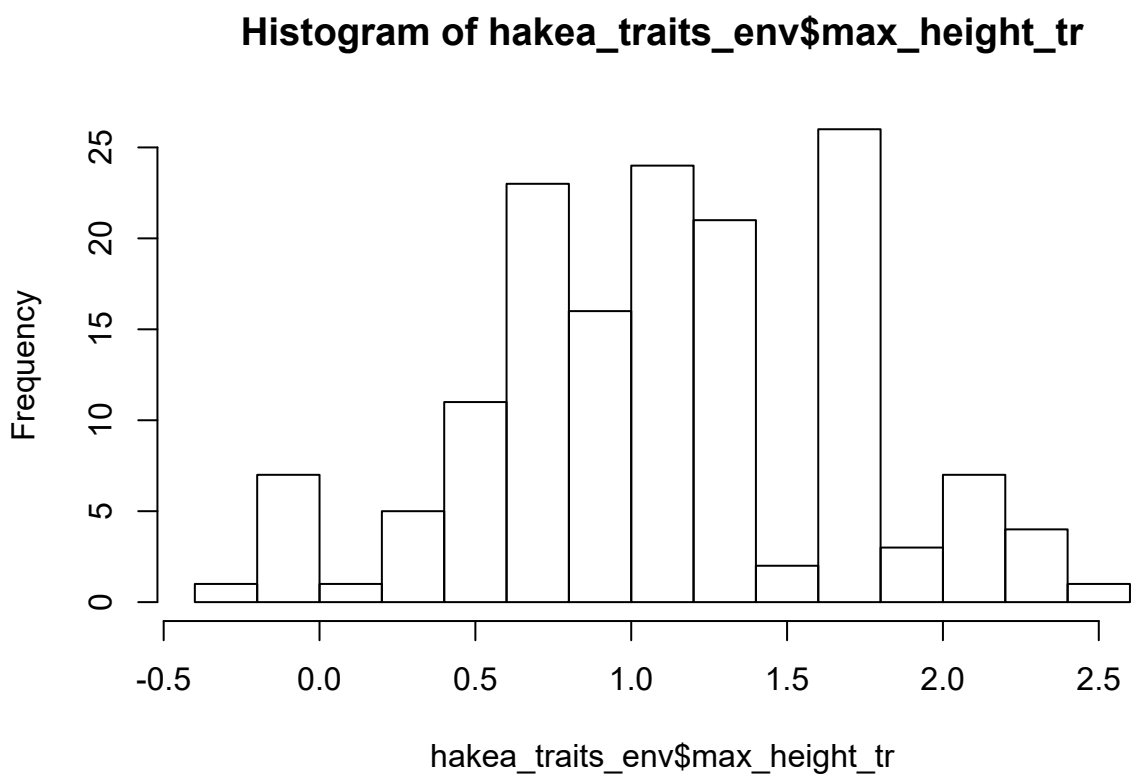
```
##                     Hakea_archaeoides Hakea_trineura Hakea_minyma
## Hakea_archaeoides          0.43834947    -0.09967147  -0.05398885
## Hakea_trineura            -0.09967147     0.43834947  -0.05398885
## Hakea_minyma              -0.05398885    -0.05398885   0.42491947
## Hakea_multilineata        -0.03979680    -0.03979680  -0.04634537
## Hakea_francisiana         -0.03480378    -0.03480378  -0.04053075
##                     Hakea_multilineata Hakea_francisiana
## Hakea_archaeoides          -0.03979680       -0.03480378
## Hakea_trineura             -0.03979680       -0.03480378
## Hakea_minyma               -0.04634537       -0.04053075
## Hakea_multilineata          0.48646862       -0.11040891
## Hakea_francisiana          -0.11040891        0.52903580
```

Now to run the model! First, we set up a prior for our phylogenetic random effect, which is to say a prior probability density on the scaling factor of the phylogenetic effect. As in the manuscript, we use a "PC" prior, which stand for "Penalizing Complexity". This is a standard prior developed by the developers of `INLA`, which is "weakly informative". It places slightly more of the prior probability density on values close to zero, but has a "long tail", which allows the data to push the parameter away from zero if there is good evidence (e.g. the likelihood of the data is higher). The PC prior has two parameters, p1 and p2: p2 is the the proportion of the prior probability density that falls above values greater than p1. Choosing sensible values here is important, and the values chosen will depend greatly on the type of response data, how its errors are being modeled, and whether it has been standardised. Let's model the maximum height of the *Hakea* species using log-transformed maximum height as our response, and Gaussian errors.

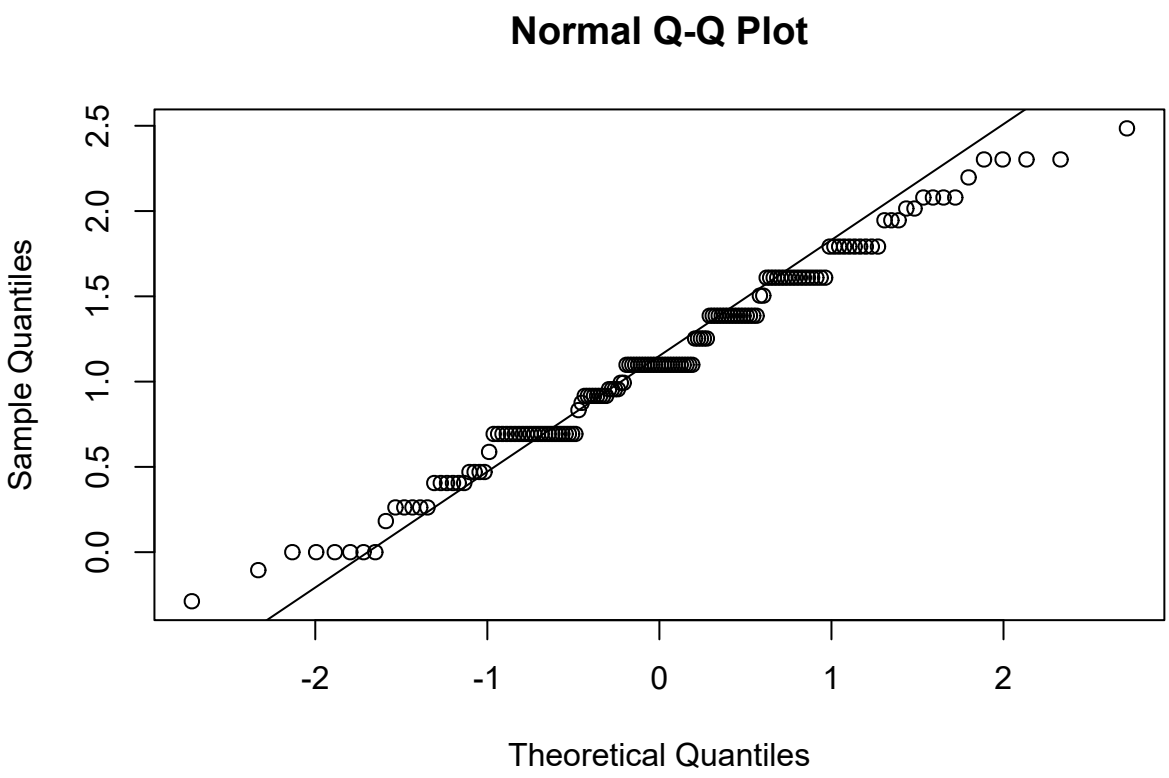
```
hakea_traits_env <- hakea_traits_env %>%
  mutate(max_height_tr = log(max_height),
         max_height_tr_st = (max_height_tr - mean(max_height_tr)) / sd(max_height_tr),
         rainfall_st = (rainfall - mean(rainfall)) / sd(rainfall),
         temperature_st = (temperature - mean(temperature)) / sd(temperature))
```

We can look at the transformed variable to see how Gaussian looking it is, but more importantly, whether there is any major skew which could lead to high leverage point. The distribution is a little funny looking due to some discreteness in the maximum height values (presumably because they were rounded to the nearest half meter or so), but overall doesn't look bad, a Gaussian error model will likely do fine.

```
## Look at transformed variables
hist(hakea_traits_env$max_height_tr, breaks = 10)
```




```
qqnorm(hakea_traits_env$max_height_tr)
qqline(hakea_traits_env$max_height_tr)
```



Now set the PC prior and run the model. Given we are modelling Gaussian data that we have standardised to a variance of 1, we would not expect any random factor to have a variance greater than one. So we will set our prior to only have about 10% of its prior probability density above 1. We will guestimate the total variance possibly explained by the phylogenetic effect based on it's diagonal entries, which are all equal to 2.7373648. So to get to 1, the scaling factor would have to be about 0.36. The only other thing we need to do to run the model is to create a column in the data that indexes into the phylogenetic precision matrix, so `INLA` knows how to match the rows of data to the covariance structure.

```
pcprior <- list(prec = list(prior="pc.prec", param = c(0.36, 0.1)))
hakea_traits_env <- hakea_traits_env %>%
  left_join(tibble(species = rownames(phylo_prec_mat),
                   phy_id = 1:nrow(phylo_prec_mat)))
```

```
## Joining, by = "species"
```

```
max_height_model_phylo <- inla(max_height_tr_st ~ rainfall + temperature +
                                 f(phy_id, model = "generic0", Cmatrix = phylo_prec_mat,
                                   constr = TRUE, hyper = pcprior),
                               data = hakea_traits_env)
```

The model specification is fairly simple for this model. Once we want to add space, things will get more complicated, and specifying the model will require special data structures. More on that in the next section. A few thing to note about the model specification. The `f()` function allows you to specify "random effects" in INLA, though the term random effect is not that meaningful in Bayesian statistics (technically all parameters of the model are "random" because they have prior probabilities). Instead we can think of them as sub-models

of the data, which have their own set of "hyper-parameters". In this can we are specifying a `"generic0"` model, which simply means use a specified precision matrix and use one hyperparameter which is just a scaling factor to multiple the covariance matrix by (in this case `"Cmatrix"` stands for concentration matrix, another term for precision matrix, not to be confused with covariance matrix, which also starts with "C"!). The last parameter of `f()` that requires some explanation is `constr = TRUE`. This means that `INLA` will attempt to constrain the integral of the random effect to equal roughly zero. This is equivalent to making your random effects have a mean of zero, which will be necessary to make your model identifiable if it also includes an intercept, which `INLA` does by default. This also means that the random effects can be roughly interpreted as a deviation from the linear predictor of the "fixed effects". Let's get a summary of the model.

```
summary(max_height_model_phylo)
```

```
##
## Call:
##    c("inla(formula = max_height_tr_st ~ rainfall + temperature + f(phy_id,
##    ", " model = \"generic0\", Cmatrix = phylo_prec_mat, constr = TRUE, ",
##    " hyper = pcprior), data = hakea_traits_env)")
## Time used:
##     Pre = 0.315, Running = 1.35, Post = 0.14, Total = 1.81
## Fixed effects:
##               mean    sd 0.025quant 0.5quant 0.975quant   mode kld
## (Intercept) -0.683 0.626     -1.912   -0.684      0.548 -0.685   0
## rainfall     0.001 0.000      0.000    0.001      0.001  0.001   0
## temperature  0.014 0.032     -0.048    0.014      0.076  0.015   0
##
## Random effects:
##   Name     Model
##     phy_id Generic0 model
##
## Model hyperparameters:
##                                          mean    sd 0.025quant 0.5quant
## Precision for the Gaussian observations 1.52 0.258       1.06     1.51
## Precision for phy_id                    7.68 3.937       3.03     6.71
##                                         0.975quant mode
## Precision for the Gaussian observations       2.08 1.49
## Precision for phy_id                         17.91 5.28
##
## Expected number of effective parameters(stdev): 32.79(11.88)
## Number of equivalent replicates : 4.64
##
## Marginal log-Likelihood:  -240.47
```

Interpreting the fixed effects from `INLA` is fairly straightforward. `0.025quant` and `0.975quant` represent the lower and upper values of the estimated 95% credible interval. From this, we can see there is reasonable evidence that rainfall has an effect on maximum height, since its credible interval does not overlap zero. On the other hand, temperature does not seem to have a strong effect, since there is a strong overlap with zero. Interpreting the "random effects" is a little less straightforward. If you are used to thinking in terms of variance and covariance (as I am), we now have to think in terms of precision and precision matrices. Luckily, there is a fairly straightforward transformation to convert a scaling factor ϕ for a precision matrix into a scaling factor on the equivalent covariance matrix" $\sigma = 1/\phi$. We can transform the marginal posterior distribution of any parameter using the `INLA` function `inla.tmarginal()`, then we can resummarise (using `inla.qmarginal()`) to credible intervals to get something more interpretable if you are more familiar with variance thinking.

```
phylo_effect_var <- inla.tmarginal(function(x) 1 / x,
                                   max_height_model_phylo$marginals.hyperpar$`Precision for phy_id`,
                                   method = "linear") %>%
  inla.qmarginal(c(0.025, 0.5, 0.975), .)
phylo_effect_var
```

```
## [1] 0.05538867 0.14841313 0.32837803
```

So the median of the posterior on our phylogenetic scaling factor is 0.14, and the 95% credible interval is reasonably symmetrical and shifted away from zero, providing good evidence that phylogeny is a good predictor of maximum height. Let's have a look at a plot of the phylogenetic effect on the phylogeny. For this we will use the `ggtree` package.

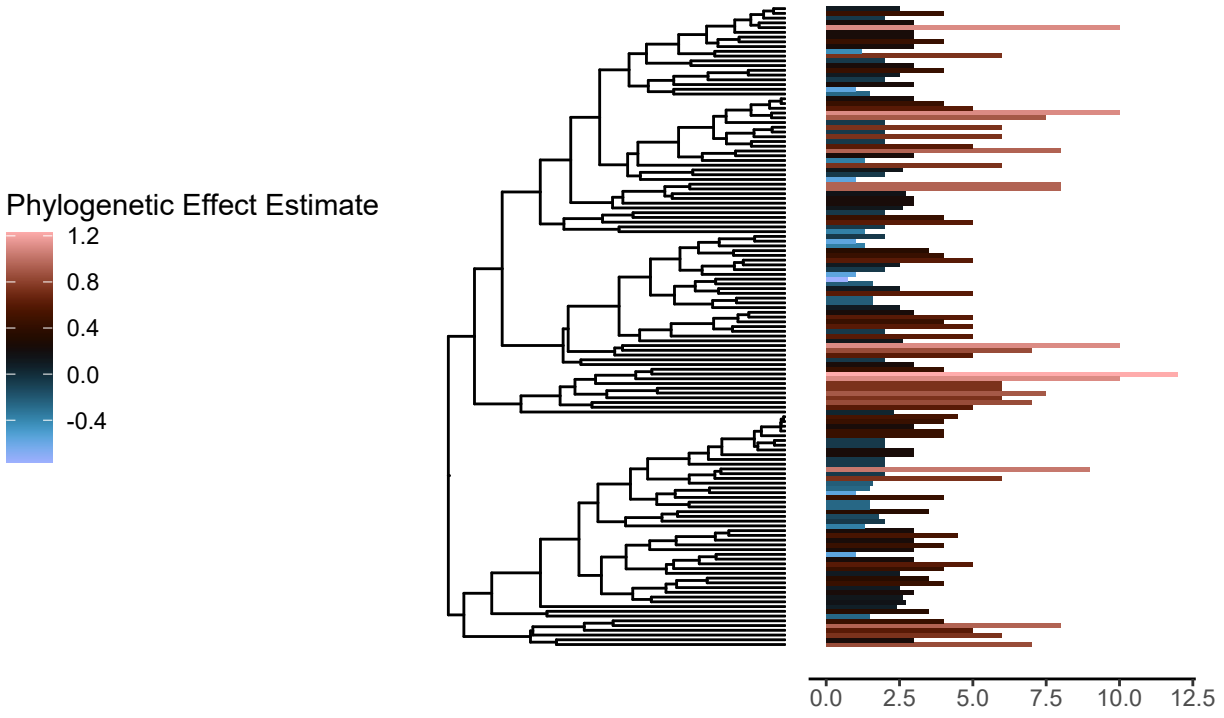
```
tree_pred <- max_height_model_phylo$summary.random$phy_id %>%
  as_tibble %>%
  rename(phy_id = ID) %>%
  dplyr::left_join(hakea_traits_env %>% dplyr::select(phy_id, species, max_height_tr_st, max_height))
```

```
## Joining, by = "phy_id"
```

```
tree_dat <- fortify(hakea_tree) %>% dplyr::left_join(tree_pred, by = c("label" = "species"))
```

```
## Warning: `data_frame()` is deprecated as of tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
## add an offset to plot phylogenetic effects easily
tree_wid <- max(tree_dat$x)
offset <- 1
tree_dat <- tree_dat %>%
    mutate(phylo_effect_x = tree_wid + offset)

height_bars <- ggplot(tree_dat %>% filter(isTip), aes(y, max_height)) +
  geom_bar(aes(fill = max_height_tr_st), stat = "identity") +
  scico::scale_fill_scico(palette = "berlin") +
  coord_flip() +
  theme_tree() +
  theme(legend.position = "none",
        axis.line.x = element_line(),
        axis.text.x = element_text(),
        axis.ticks.x = element_line())

(ggtree(tree_dat) +
  geom_tile(aes(phylo_effect_x, y, fill = mean), colour = NA,
            data = tree_dat %>% filter(isTip)) +
    scico::scale_fill_scico(name = "Phylogenetic Effect Estimate", palette = "berlin") +
  theme(legend.position = "left",
        plot.margin = unit(c(0, -0.025, 0.1, 0), "npc"))) +
  height_bars
```



The boxes immediately right of the phylogeny are the predicted deviations estimated from the phylogenetic random effect in the model. To the right of that is the observed maximum height of each species. We can see that the phylogenetic estimate is following the observed data, while being smoothed by the phylogeny. We can also see that the phylogenetic effect is not exactly overwhelming, in other words, the fit is not especially strong. Most of the effect is probably driven by that one large clade near the middle with very tall Hakeas. However, it is believable that phylogeny is explaining some of the variation, which is why we probably saw a reasonable estimate in the `INLA` model.

We also can conclude from the model summary that mean rainfall across the species' range is a contributing factor to maximum height. However, we know that there is likely to be spatial autocorrelation in rainfall, which means we may want to account for the fact that some of our species' ranges are closer together than others, and so they are not entirely independent data points. It may also be true that certain phylogenetic clades may tend to occur in spatially restricted areas. This could lead to confounding between space and phylogeny that we would like to at least try and disentangle. To do this, we can model spatial covariance in our response using another type of "random effect" in `INLA`. Specifically, we can model the expected covariance between spatially explicit data points using the Matern covariance function, which described how covariance decays with increasing spatial distance between points. Its parameters can be estimated with data. I will not go into details on the Matern covariance function, I will leave that up to the interested reader to research themselves. The main contribution of this tutorial will be to describe a way we can use this spatial covariance model when we have many spatial occurrence records for each species, but we only have a single response value for each, such as is the case for maximum height.

In brief, the idea is this: we calculate a spatial "random effect", which in this case is a "random field" across a spatial mesh. We then take the estimated "random effects" at each of the data points within each single species, and average them to get a single random effect estimate for each species. This then gives a single spatial predictor for each species we can include in the linear model of the species-level response. In principle the concept is fairly simple but it can be a little tricky to set up in `INLA` (and difficult to do at all in most other modelling frameworks, though it would be possible to implement in a very flexible statistical

programming language such as Stan). This type of random effect is sometimes called a multi-membership random effect (for example the R package `brms` can implement multi-membership random effects using the `mm()` formula helper function, however, only Gaussian distributions on the grouping terms are currently supported, so spatial or phylogenetic random effects with multi-membership cannot currently be done in `brms`).

To setup the spatial random effect we need to generate a special data structure for `INLA` to use. We will add the phylogenetic effects and fixed effects into this data structure along with the spatial. The first thing to do is setup a Stochastic Partial Differential Equation (SPDE) data structure, which tells `INLA` how to model the Matern covariance internally (it uses SPDEs to approximate the integral). What this needs is an object containing the spatial mesh to be used, and some parameters describing the priors on the Matern covariance function terms.

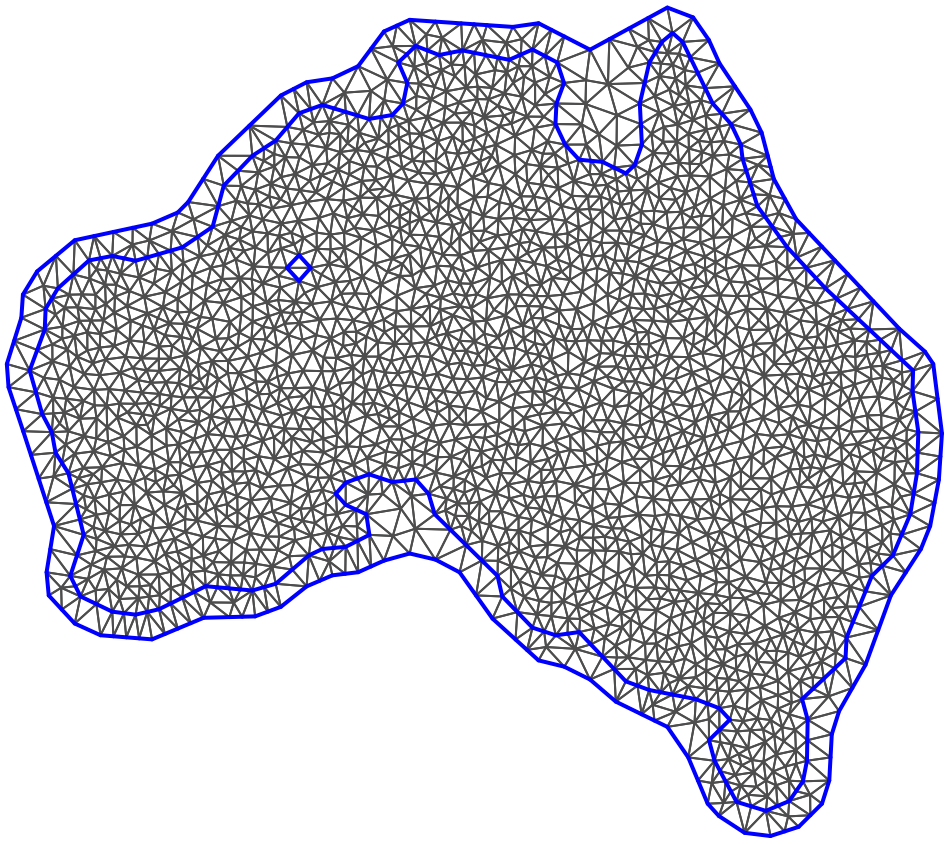
The Spatial Mesh

The spatial mesh is key component to `INLA`'s approach to spatial modelling. It is a triangular mesh across the landscape of interest. Each point of the mesh is where the random effects are being estimated. Data can be anywhere on the landscape however, since they values of the random effects at any point can be simply estimated using triangular interpolation. Essentially, any point in you data receives a set of weights across every mesh point, consisting of three non-zero values, which sum to one and are calculated based on the distance to the three nearest mesh points. All other mesh points receive a weight of zero. This is another example of a multi-membership random effect, where each individual data point has membership in exactly three groups, represented by the three closest mesh points. In our model we will additionally average these weights across each species, such that each species will have "membership" in all mesh points which fall within its range. `INLA` provides a useful Shiny app that allows users to pick parameters to help design a good mesh for their particular dataset (`INLA::meshbuilder()`). Here, we will just show the mesh we ended up designing for Australia to use in this study. We use the our *Hakea* occurrence points to make a convex hull, which will serve as the borders of the mesh.

```
coordsy <- SpatialPoints(hakea_traits_occ[ , c("longitude", "latitude")],
                         proj4string = CRS("+proj=longlat +ellps=WGS84 +datum=WGS84"))

boundary.loc <- coordsy
boundary <- list(
  inla.nonconvex.hull(coordinates(boundary.loc), 0.5, crs=inla.CRS("longlat")),
  inla.nonconvex.hull(coordinates(boundary.loc), 1.6, crs=inla.CRS("longlat"))
  )
```

```
## Warning in inla.nonconvex.hull(coordinates(boundary.loc), 0.5, crs = inla.CRS("longlat")): Resolutio
## Resolution >=(86,71) required for more accurate results.
```

```
  ## Build the mesh:
spatial_mesh <- inla.mesh.2d(boundary=boundary,
                             max.edge=c(1, 2),
                             min.angle=26,
                             cutoff=0.5, ## Filter away adjacent points.
                             offset=c(0.5, 1.6)
                             )

plot(spatial_mesh)
```




```
dim(spatial_mesh$loc)
```

```
## [1] 2219    3
```

So you can definitely see the outline of Australia there. The larger triangles around the edges are a way to try and reduce edge effects. Now we can create our `spde.pcmatern` object (spde being stochastic partial differential equations, pc being penalizing complexity prior, and matern for the Matern covariance).

```
spde <- inla.spde2.pcmatern(
      mesh = spatial_mesh, alpha = 2, ### mesh and smoothness parameter
      prior.range = c(2, 0.1), ### P(practic.range<2)=0.1
      prior.sigma = c(1, 0.1),
      constr = TRUE)
```

I won't go into the details of the parameters and their priors here, but only say that the `range` parameter effects how quickly the covariance decays to zero with increasing spatial distance, and `sigma` is a scaling factor. `alpha` is a third parameter which effects how smooth the function is, typically this is fixed to a generally good value, in this case 2. We can now use this to setup our random effects for the model.

```
coords <- as.matrix(hakea_traits_occ[ , c("longitude", "latitude")])
A <- inla.spde.make.A(spatial_mesh, loc = coords) ## this creates a sparse matrix
dim(A)
```

```
## [1] 46730  2219
```

The dimensions of this matrix are 46730 rows, so one for each of our location data points, and 2219 columns, one for each of the spatial mesh points. The model then will have a 2219 level random effect accounting for space. We can see that each row has exactly three non-zero entries that add to one. These triangulate between the three nearest points in the spatial mesh.

```
apply(A, 1, function(x) sum(x > 0)) %>% head
```

```
## [1] 3 3 3 3 3 3
```

```
apply(A, 1, sum) %>% head
```

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

```
apply(A, 1, function(x) x[x > 0]) %>% t %>% head
```

```
##           [,1]       [,2]      [,3]
## [1,] 0.1450908 0.05075631 0.8041529
## [2,] 0.0784957 0.22007584 0.7014285
## [3,] 0.0784957 0.22007584 0.7014285
## [4,] 0.3569754 0.14477908 0.4982455
## [5,] 0.4407605 0.03831469 0.5209248
## [6,] 0.2801646 0.04434305 0.6754923
```

`INLA` expects the `A` matrix for random effects to have a number of rows equal to the number of data points, and a number of columns equal to the number of random effects levels. Each element then is the "membership" of a data point to each of the random effect grouping levels. In a standard random effect, each row would only have a single nonzero value, a 1 in the column of the grouping level it belongs to. If each row is standardised to sum to one, the values can then be considered weights, and the linear predictor for that random effect for a particular data point is a weighted average of the random effects levels it has membership in. For those who think better in matrix math, the linear predictor for the random effect is simply the dot product between a column vector of the random effects (this is estimated in the model I realised I haven't mentioned yet) and the `A` matrix. In this analysis, our "true" data points are only measured at the species-level. So we have 152 data points, and not 46730. So, we need an `A` matrix with 152 rows instead. We do this by simply averaging the full `A` we just calculate across the rows within each *Hakea* species. A simple way to do this is:

```
specs <- hakea_traits_env$species

spec_A_rows <- lapply(specs, function(x) { ## apply to each species one at a time
```

```
  A[which(hakea_traits_occ$species == x), ] %>% ## get rows associated with the species
    apply(2, mean) ## average each column to return a single row with the same number of columns

}) %>%
  do.call(rbind, .) ## rbind the species back together into one matrix

## convert back to sparse matrix
spec_A_rows <- Matrix(spec_A_rows, sparse = TRUE)

dim(spec_A_rows)
```

```
## [1]  152 2219
```

```
apply(spec_A_rows, 1, sum) %>% head
```

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

```
apply(spec_A_rows, 1, function(x) sum(x > 0)) %>% head
```

```
## [1] 36 10  6 21  5 19
```

That last command just gives an idea of the range of how many mesh points influence each species. Note that if the value is 3, it does not mean the species is represented by a single occurrence point, only that all of its occurrence point fall within one triangle of the mesh (easily possible given some Hakea species have very restricted ranges, and the mesh is quite large).

And now, we can finally make out special `INLA` data structure, which they call a data "stack". It is just a big list, with entries for the response, the "effects" in the model, and `A` matrices that define membership in "effects". One thing to note that to use this framework, all effect must be specified in the object, and so we also need to include the intercept term (just a vector of 1s). Later we will suppress the automatic intercept term in the `inla` model specification. Here is the stack for our model:

```
stack <- inla.stack(data = list(resp = hakea_traits_env$max_height_tr_st),
                    A = list(spec_A_rows, 1, 1, 1), ## one A for each effect, 1 is shorthand, see below
                       effects=list(spat_id = 1:spde$n.spde,
                                    phy_id = hakea_traits_env$phy_id,
                                    intercept = rep(1, nrow(hakea_traits_env)),
                                    list(rainfall_st = hakea_traits_env$rainfall_st,
                                         temperature_st = hakea_traits_env$temperature_st)),
                      tag='est') ## this just gives the object a label we can refer to later, useful
                                 ## if we need to bind multiple stacks together
```

In the `A` part of the object, 1 is shorthand for a matrix with a single one in each row, in the column associated with the random effect level for that data point (e.g. a standard effect weight matrix). So that is all we now need to run the spatial model, however, another useful thing to do with any statistical model is make predictions for it. `INLA` will automatically sample predictions from its marginal posterior distribution if the response is `NA` in the data, simultaneously while fitting the model, a super useful and convenient feature. So, we will make some more data stacks for the sole purpose of getting some useful predictions. In particular, we would like to predict our spatial and phylogenetic random effects for each species. It would also be useful to predict spatial random effects in a grid to generate maps, however, `INLA` has some specialised methods to do that in a computationally efficient manner we will have a look at a bit later. Here we go:

```
## estimate spatial random effect for each species
stck.loc <- inla.stack(data = list(resp = NA),
                       A = list(spec_A_rows),
                       effects = list(spat_id = 1:spde$n.spde), tag='loc')
## estimate phylogenetic random effect for each species
stck.phy <- inla.stack(data = list(resp = NA),
                        A = list(1),
                        effects=list(phy_id = hakea_traits_env$phy_id), tag='phy')

big_stack <- inla.stack(stack, stck.loc, stck.phy)
```

Leaving some effects out of a data stack means that INLA will treat them as missing data, which means they will be set to zero. So predictions for each effect will be evaluated at a reference level of zero for all other effects, which in properly standardised data is the mean, which is usually the most sensible value to evaluate independent effects at.

Now, we can finally run our model! For simplicity, we will use the same PC prior for each random effect.

```
full_mod <- inla(resp ~ 0 + intercept + rainfall_st + temperature_st +
                   f(spat_id, model = spde) + ## spatial random effect
                   f(phy_id, model = "generic0", Cmatrix = phylo_prec_mat,
                     constr = TRUE, hyper = pcprior),
                 data = inla.stack.data(big_stack, spde = spde),
                 control.predictor = list(A = inla.stack.A(big_stack), compute = TRUE,
                                          link = 1),
                 control.compute = list(config = TRUE),
                 num.threads = 10
                 )

summary(full_mod)
```

```
##
## Call:
##    c("inla(formula = resp ~ 0 + intercept + rainfall_st + temperature_st +
##    ", " f(spat_id, model = spde) + f(phy_id, model = \"generic0\", ", "
##    Cmatrix = phylo_prec_mat, constr = TRUE, hyper = pcprior), ", " data =
##    inla.stack.data(big_stack, spde = spde), control.compute = list(config
##    = TRUE), ", " control.predictor = list(A = inla.stack.A(big_stack),
##    compute = TRUE, ", " link = 1), num.threads = 10)")
## Time used:
##     Pre = 0.5, Running = 396, Post = 1.41, Total = 398
## Fixed effects:
##                  mean    sd 0.025quant 0.5quant 0.975quant  mode kld
## intercept       0.423 0.163      0.115    0.418      0.760 0.408   0
## rainfall_st     0.036 0.127     -0.220    0.039      0.280 0.044   0
## temperature_st 0.201 0.112     -0.018    0.200      0.424 0.199   0
##
## Random effects:
##   Name     Model
##     spat_id SPDE2 model
##    phy_id Generic0 model
##
## Model hyperparameters:
```

```
##                                             mean       sd 0.025quant 0.5quant
## Precision for the Gaussian observations    1.884 2.61e-01      1.390     1.88
## Range for spat_id                          3.114 1.22e+00      1.358     2.91
## Stdev for spat_id                          0.951 1.98e-01      0.624     0.93
## Precision for phy_id                    2004.812 2.55e+04     29.051   245.44
##                                         0.975quant   mode
## Precision for the Gaussian observations       2.41  1.887
## Range for spat_id                             6.07  2.533
## Stdev for spat_id                             1.40  0.889
## Precision for phy_id                      12608.32 53.926
##
## Expected number of effective parameters(stdev): 33.85(6.78)
## Number of equivalent replicates : 4.49
##
## Marginal log-Likelihood:  -223.50
## Posterior marginals for the linear predictor and
##  the fitted values are computed
```

This model illustrates a number of interesting things about spatial modelling, including a good reason why you might want to do it. Before I discuss that, let's just see what our phylogenetic scale is on the covariance scale.

```
phylo_effect_var <- inla.tmarginal(function(x) 1 / x, full_mod$marginals.hyperpar$`Precision for phy_id
  inla.qmarginal(c(0.025, 0.5, 0.975), .)
```

```
## Warning in regularize.values(x, y, ties, missing(ties)): collapsing to unique
## 'x' values
```

```
phylo_effect_var
```

```
## [1] 1.149589e-05 1.566725e-04 8.183843e-04
```

Interestingly, the phylogenetic effect has shrunk away to basically nothing. One explanation for this is that maximum height originally showed an effect of phylogeny because closely related species tend to live in similar areas, and that the area a species lives in is ultimately a better explanation of its maximum height. In other words, the phylogenetic effect we found before was due to phylogeny being confounded with an unestimated spatial effect. Now, we still shouldn't conclude phylogeny and hence ancestry is not responsible for the variation in maximum height, only that we cannot rule out that its effect is only due to a correlation with space. Also bear in mind that the spatial model is somewhat more complex than the phylogenetic model, such that in cases where they are saying similar things, it is likely that space will be a better explanation (it can 'soak up' more variation due to its greater flexibility). This is partly due to the simplistic evolutionary model underlying the phylogenetic covariance model, but I digress.

As for the fixed effects, the rainfall effect has also vanished. Given the strong spatial autocorrelation in rainfall across Australia, this is not surprising. It implies that the effect of rainfall on maximum height could potentially just be explained by spatial non-independence. That is, that species with high maximum height are spatially clustered, and some of these spatial clusters happen to be in high rainfall areas. Again, it doesn't mean that rainfall isn't driving the effect, merely that we cannot be sure, because we don't have as many independent data points as we thought. We may be able to get more insight by examining maps of our spatial effect.

Another quite interesting thing is that the effect of temperature has become much stronger in this model, and in fact, the credible interval only overlaps zero very slightly. This highlights something about random effects

models that is often not appreciated. Many people think of using a random effect as away of just 'accounting' for non-independence, and so they typically expect that adding a random effect will tend to make any fixed effect "less significant", because they now have fewer degrees of freedom. But in fact random effect, like any other factor you could add to model, can change other effect in any direction. All other effects are now only interpretable as the effect of x on y, after accounting for all other variables (actually technically not "after", they are all fit simultaneously). This means effects that were previously not prominent can become good predictors after adding a random effect. In this case, we can say that space might be a negative confounder of temperature. Admittedly, the reasons for this are very difficult to wrap your head around. Generally I think most humans, scientists or otherwise, are ill equipped to understand intuitively the multifariousness of multiple regression, whether we use random effects or not. But perhaps we can gain some more insight by making some plots! One of my favourite things to do! First let's get a map of Australia (from Natural Earth), on which we can plot the spatial random effect,

```
library(rnaturalearth)
```

```
## Warning: package 'rnaturalearth' was built under R version 3.6.3
```

```
library(ggplot2)
library(ggthemes)
```

```
## Warning: package 'ggthemes' was built under R version 3.6.3
```

```
aus_coast <- ne_countries(country = "Australia", scale = 10, returnclass = "sf")
ggplot(aus_coast) + geom_sf() + theme_map() + coord_sf(xlim = c(113.1, 153.65), ylim = c(-43.65, -10.68
```

Now we use tools provided by `INLA` to approximate the models estimated spatial field after fitting the model.

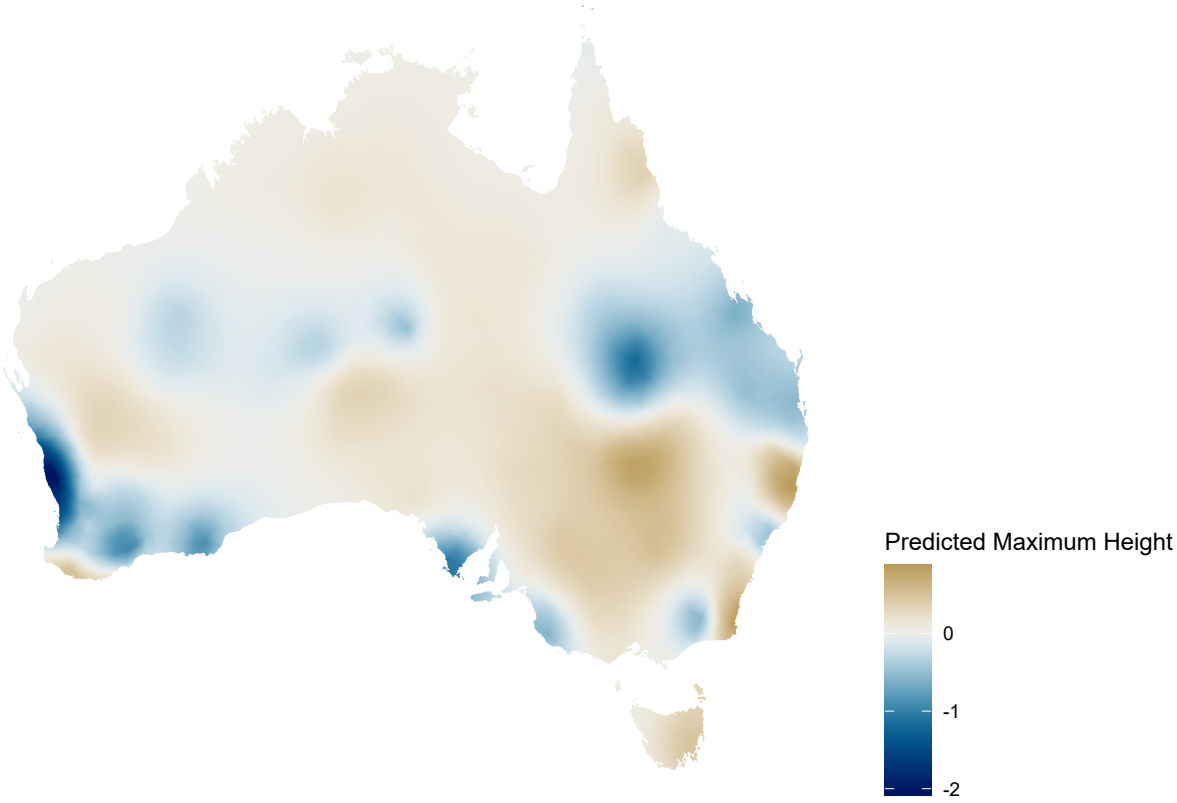
```
grid <- inla.mesh.projector(spatial_mesh, dims=c(501,501))
prd_mean <- inla.mesh.project(grid, full_mod$summary.ran$spat_id$mean)
prd_sd <- inla.mesh.project(grid, full_mod$summary.ran$spat_id$sd)

grid_df <- tibble(long = grid$lattice$loc[ , 1],
                      lat = grid$lattice$loc[ , 2],
                      prediction_mean = as.vector(prd_mean),
                      prediction_sd = as.vector(prd_sd))
```

Now we can make the map showing the estimated spatial field for maximum height across Australia.

```
## code to centre scico colour gradient on zero
low <- min(grid_df$prediction_mean, na.rm = TRUE)
high <- max(grid_df$prediction_mean, na.rm = TRUE)
begin <- 0
end <- 0.5 + 0.5*(abs(high)/abs(low))

##create a "sea mask" to cover values extrapolated into the ocean
sea_mask <- as(raster::extent(c(min(grid_df$long),
                          max(grid_df$long),
                          min(grid_df$lat),
                          max(grid_df$lat))), "SpatialPolygons")
sea_mask <- rgeos::gBuffer(sea_mask, width = 1)
au_sea_shp <- rgeos::gDifference(sea_mask, sf::as_Spatial(aus_coast))
```

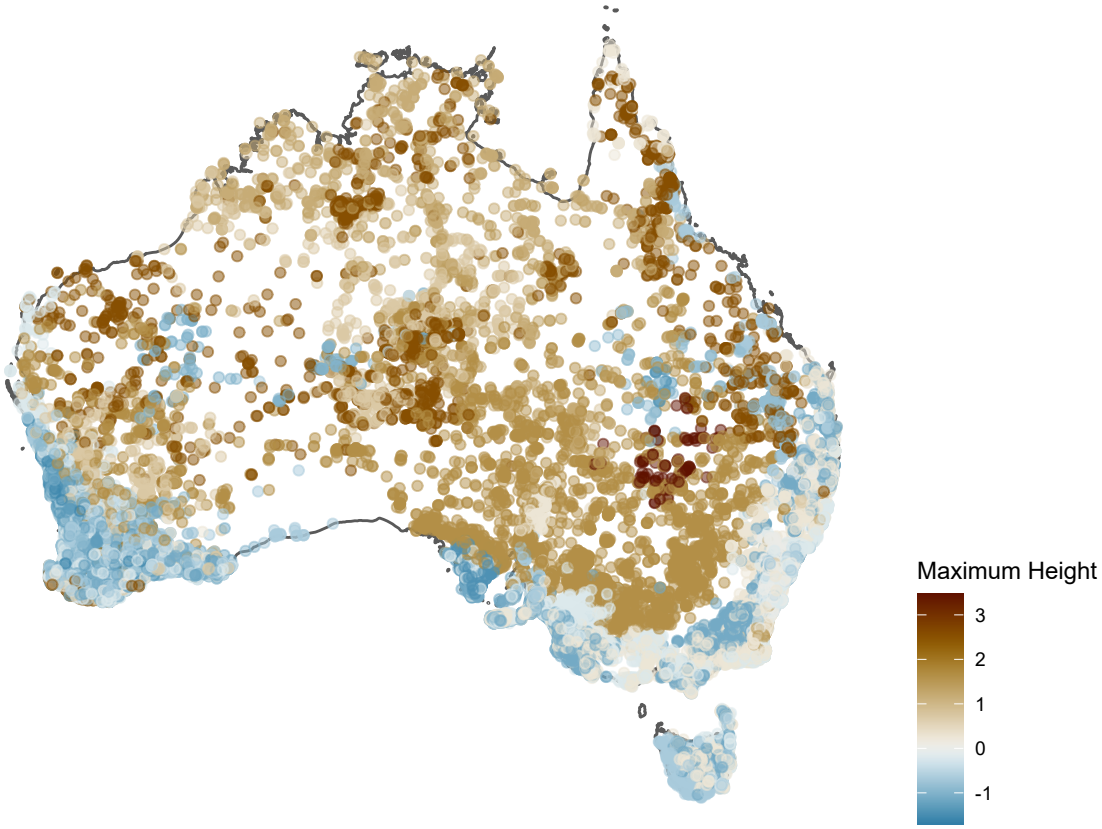
```
## Warning in RGEOSBinTopoFunc(spgeom1, spgeom2, byid, id, drop_lower_td,
## unaryUnion_if_byid_false, : spgeom1 and spgeom2 have different proj4 strings
```

```
au_sea <- fortify(au_sea_shp)

spat_map <- ggplot(aus_coast) +
  geom_sf() +
  geom_raster(aes(long, lat, fill = prediction_mean), data = grid_df) +
  ggpolypath::geom_polypath(aes(long, lat, group = group), fill = "white", data = au_sea) +
  coord_sf(xlim = c(113.1, 153.65),
           ylim = c(-43.65, -10.68)) +
  scico::scale_fill_scico(name = "Predicted Maximum Height", palette = "vik",
                    begin = begin, end = end, expand = c(0, 0)) +
  theme_map() +
  theme(legend.position = "right")

spat_map
```



This shows a fairly complicated spatial pattern of maximum height across Australia. As a sanity check, let's plot the individual Hakea occurrence points along with their maximum heights, and see if we can eyeball whether the above pattern seems reasonable.

```
hakea_traits_occ <- hakea_traits_occ %>%
  mutate(max_height_st = (max_height - mean(max_height)) / sd(max_height))

low <- min(hakea_traits_occ$max_height_st, na.rm = TRUE)
high <- max(hakea_traits_occ$max_height_st, na.rm = TRUE)
begin <- 0.5 - 0.5*(abs(low)/abs(high))
end <- 1

pnt_map <- ggplot(aus_coast) +
  geom_sf(fill = "white") +
  geom_point(aes(longitude, latitude, colour = max_height_st),
             alpha = 0.5,
             data = hakea_traits_occ[sample.int(nrow(hakea_traits_occ)), ]) +
  coord_sf(xlim = c(113.1, 153.65),
           ylim = c(-43.65, -10.68)) +
  scico::scale_colour_scico(name = "Maximum Height", palette = "vik",
                    expand = c(0, 0), begin = begin, end = end) +
  theme_map() +
  theme(legend.position = "right",
        panel.grid = element_blank())

pnt_map
```



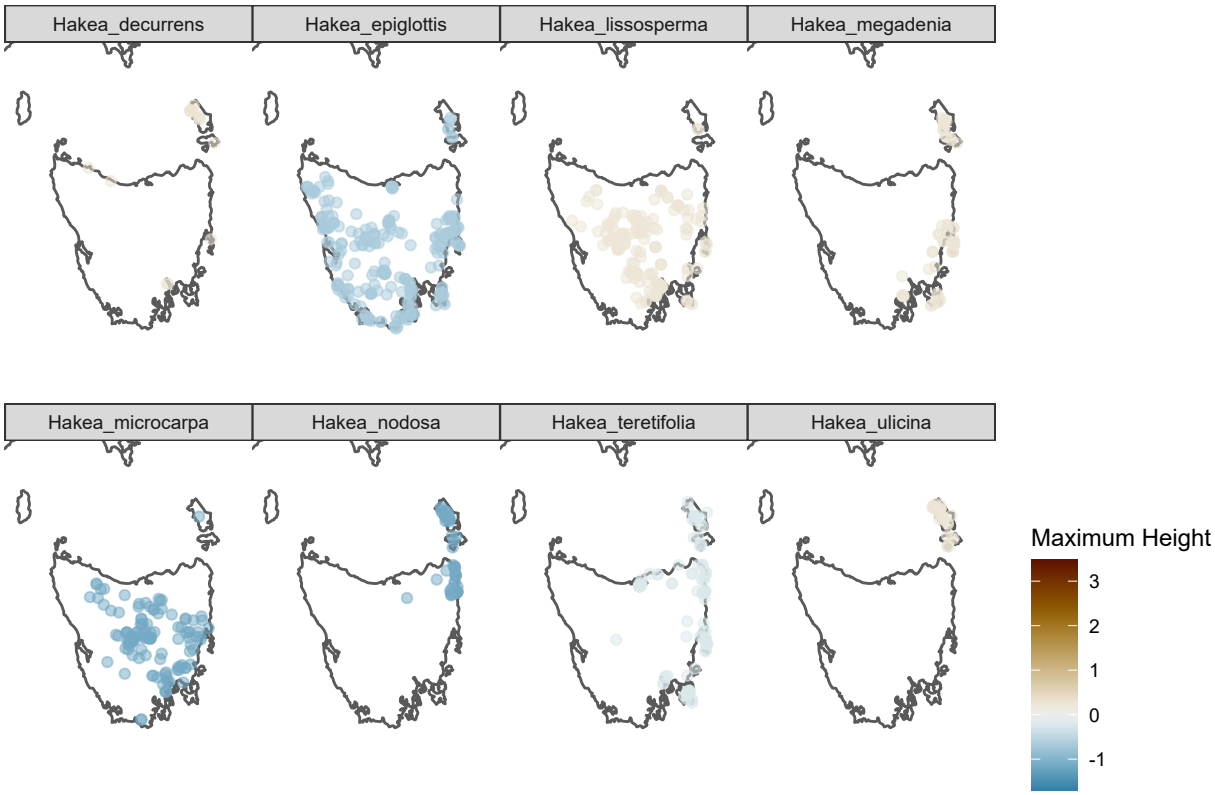
That looks pretty reasonable to me. The only surprising thing is that perhaps I would have expected the spatial field estimate to be lower in Tasmania, given it looks like there are a lot of small Hakeas there from the point figure. It is possible that the points give a slightly misleading portrait for two reasons: 1) over-plotting could be obscuring taller plants in Tasmania and elsewhere, 2) recall each species has many points, but also that some species have many more points than others; it is possible most of the visible points in Tasmania are from a single well sampled species, but rarer species in Tasmania are actually larger and are hard to see. Given our unit of analysis is the species, the model correctly weights each species equally (the well-sampled species does not have a higher weight than less-sampled species, which is what we want). Maybe we should have a closer look?

```
tas_pnts <- hakea_traits_occ %>%
  dplyr::filter(longitude > 143.8189, latitude > -54.6403,
                longitude < 158.9785, latitude < -39.2037)

table(tas_pnts$species)
```

```
##
##   Hakea_decurrens  Hakea_epiglottis Hakea_lissosperma   Hakea_megadenia
##                16               234               170                63
##  Hakea_microcarpa      Hakea_nodosa Hakea_teretifolia     Hakea_ulicina
##               119                46                95                27
```

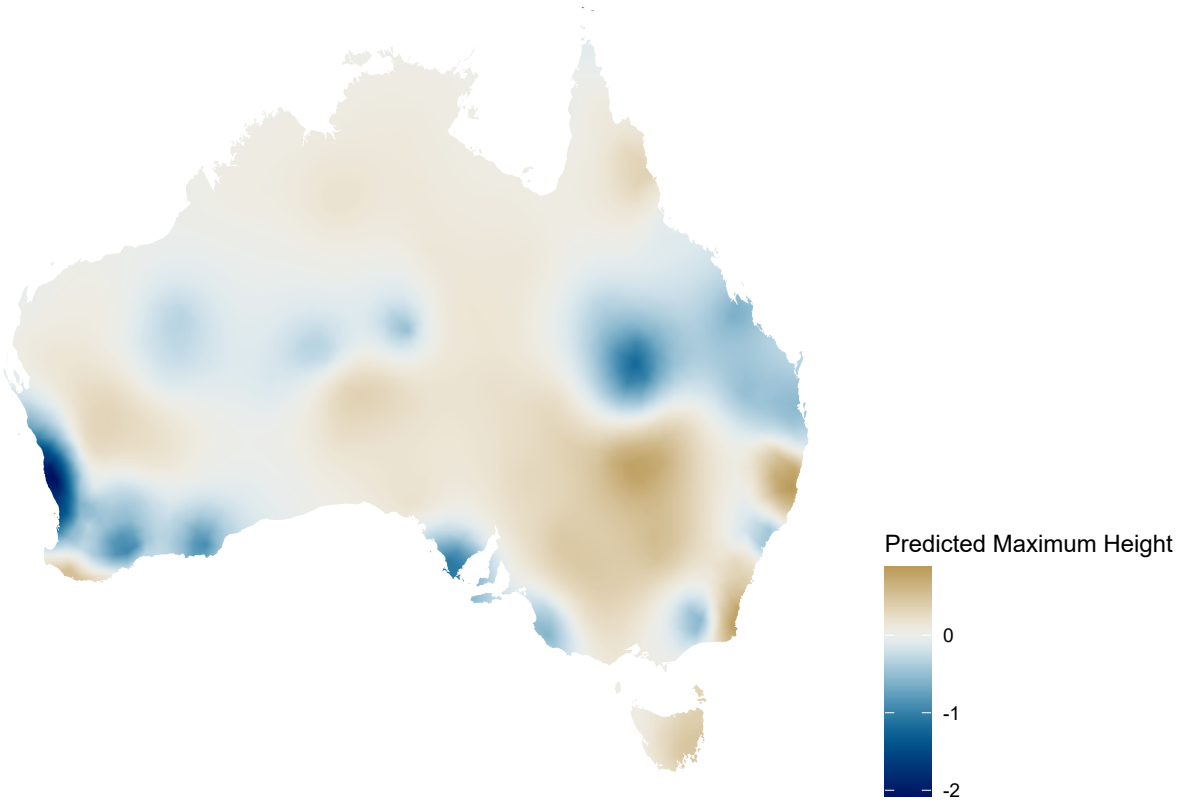
```
pnt_map_tas <- ggplot(aus_coast) +
  geom_sf(fill = "white") +
  geom_point(aes(longitude, latitude, colour = max_height_st),
```

```
            alpha = 0.5,
            data = tas_pnts[sample.int(nrow(tas_pnts)), ]) +
  facet_wrap(~ species, nrow = 2) +
  coord_sf(xlim = c(143.8189, 149),
           ylim = c(-44.6403, -39)) +
  scico::scale_colour_scico(name = "Maximum Height", palette = "vik",
                    expand = c(0, 0), limits = c(low, high), begin = begin,
                    end = end) +
  theme_map() +
  theme(legend.position = "right",
        panel.grid.major = element_line(color = "white"))

pnt_map_tas
```



Indeed we now see that there is one widespread species in Tasmania with a relatively large maximum height, but three widespread species with a smaller maximum height that are probably obscuring the taller one. Additionally, there are three tall Hakea species with restricted ranges that would contribute to Tasmania having a high spatial field estimate overall. We can also see these species are restricted to the East coast. Looking back at the original spatial field estimates, we can even see a slight increase in the spatial field towards the East coast of Tasmania. Here it is again:

```
spat_map
```



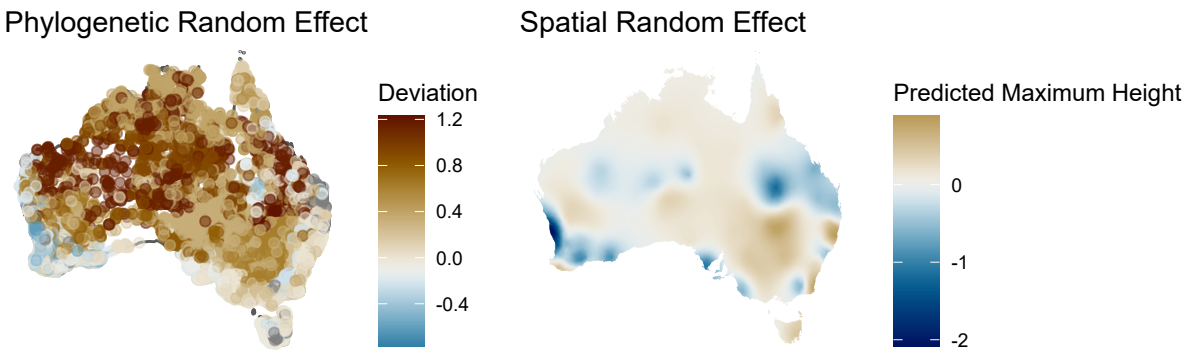
We can now speculate a little more about the reason for our model results. I don't have time to go into detail here, but here is a possible story to explain why adding space made the rainfall effect go away, but strengthened the temperature effect. The spatial effect mapped above would clearly be correlated with rainfall to some degree, in particular the places with high spatial field estimates are mostly wet (for example the far southwest corner of Australia, Tasmania, the wet tropics in Queensland). On the other hand, it is not perfectly correlated with rainfall. All of the centre of Australia is extremely dry, yet there are some areas of predicted high maximum height across the centre. These high maximum height areas not correlated with high rainfall also do not correspond to the highest temperature areas (generally the further North the hotter). Therefore, I suspect that temperature was not a good predictor in the first model because it is correlated with rainfall (rainfall absorbed any effect it had), but in the second model, the spatial effect took over the rainfall effect, which helped to reveal the (weak) potential effect of temperature. Again, that is pretty hand-wavey. Getting more precise would require a lot more work at visualising all these intercorrelated variables and prediction of maximum height based on each, which is beyond the scope of this tutorial. Feel free to explore yourself!

The last thing we can do is try and understand why the phylogenetic effect disappears when including space in the model. One way to do this is to plot the species distributions along with the phylogenetic effect estimate from our original model without space. This should give us an idea about whether the phylogenetic effect appears to correlate with space.

```
hakea_traits_occ <- hakea_traits_occ %>%
  dplyr::left_join(tree_pred)
```

```
## Joining, by = c("species", "max_height")
```

```
ggplot(aus_coast) +
  geom_sf(fill = "white") +
  geom_point(aes(longitude, latitude, colour = mean),
             alpha = 0.5,
             data = hakea_traits_occ[sample.int(nrow(hakea_traits_occ)), ]) +
  coord_sf(xlim = c(113.1, 153.65),
           ylim = c(-43.65, -10.68)) +
  scico::scale_colour_scico(name = "Deviation", palette = "vik",
                    expand = c(0, 0), begin = begin, end = end) +
  ggtitle("Phylogenetic Random Effect") +
  theme_map() +
  theme(legend.position = "right",
        panel.grid = element_blank()) +
  (spat_map + ggtitle("Spatial Random Effect"))
```



So we can see a broad correspondence between the phylogenetic predictions and the spatial model. So it is not surprising the phylogenetic effect was jettisoned in this model.

Another way to look at this is to simply plot the phylogenetic random effect (from the first model) against the estimated spatial random effect (from the second model), averaged at the species-level to see if there is a correspondence. To see the species-level correlation between spatial and phylogenetic random effects, we can use the prediction stacks we included in the original data stack we made with `INLA`, like so:

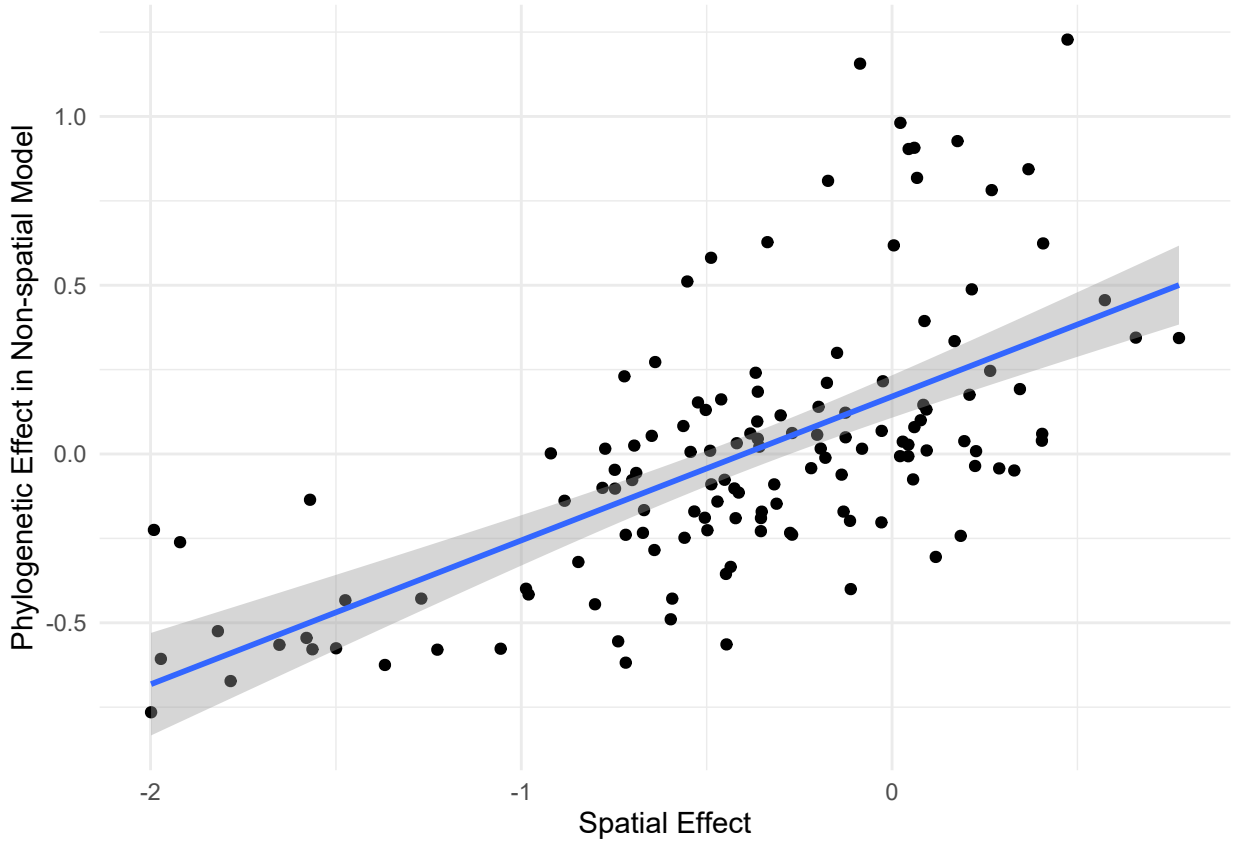
```
spat_index <- inla.stack.index(big_stack, "loc")

spec_spat_effect <- full_mod$summary.fitted.values[spat_index$data, ]
spec_phy_effect <- hakea_traits_env %>%
```

```
  left_join(max_height_model_phylo$summary.random$phy_id,
            by = c("phy_id" = "ID"))

ggplot(tibble(`Spatial Effect` = spec_spat_effect$mean,
              `Phylogenetic Effect in Non-spatial Model` = spec_phy_effect$mean),
       aes(`Spatial Effect`, `Phylogenetic Effect in Non-spatial Model`)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal()
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 17 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 17 rows containing missing values (geom_point).
```



That shows a fairly strong correlation, likely enough to mostly remove the somewhat weak phylogenetic effect. Also bear in mind that the phylogenetic effect appeared to have been largely driven by a single clade with very tall Hakeas, and so the spatial model would have only had to explain the height of these species well in order to mostly remove the phylogenetic effect.

On the other hand, as mentioned before, the spatial model is quite flexible, and so it might just take over the simpler phylogenetic model for this reason alone. Given the plot of maximum height against the phylogeny we saw earlier, I'm inclined to believe that the spatial model is a better description of the data overall, even if it were simpler. We could try a bunch of different priors to look at this if we wanted to be thorough.

If you believe that the phylogeny should be given more weight (perhaps you believe simple phylogenetic inheritance should be preferred *because* it is simpler), this could be incorporated by adjusting the model's priors to place higher posterior probability on the phylogenetic model. Finding a way to decide on priors in a principled way when you want to compare covariance models of differing complexity is beyond the scope of this tutorial, but it is a really interesting topic which needs more work. I have a project in the works which will delve more into the issue (stay tuned!).

So that is the end of this tutorial. Below is the session info for this run of the tutorial, which shows which versions of each package I am using.

```
devtools::session_info()
```

```
## - Session info ---------------------------------------------------------------
##  setting  value
##  version  R version 3.6.2 (2019-12-12)
##  os       Windows 10 x64
##  system   x86_64, mingw32
##  ui       RTerm
##  language (EN)
##  collate  English_Australia.1252
##  ctype    English_Australia.1252
##  tz       Australia/Sydney
##  date     2020-04-06
##
## - Packages -------------------------------------------------------------------
##  package            * version  date       lib source
##  ALA4R              * 1.7.0    2019-04-02 [1] CRAN (R 3.6.2)
##  ape                * 5.3      2019-03-17 [1] CRAN (R 3.6.2)
##  assertthat           0.2.1    2019-03-21 [1] CRAN (R 3.6.2)
##  backports            1.1.5    2019-10-02 [1] CRAN (R 3.6.1)
##  BiocManager          1.30.10  2019-11-16 [1] standard (@1.30.10)
##  callr                3.4.3    2020-03-28 [1] CRAN (R 3.6.3)
##  class                7.3-15   2019-01-01 [1] CRAN (R 3.6.2)
##  classInt             0.4-2    2019-10-17 [1] CRAN (R 3.6.2)
##  cli                  2.0.2    2020-02-28 [1] CRAN (R 3.6.3)
##  codetools            0.2-16   2018-12-24 [1] CRAN (R 3.6.2)
##  colorspace           1.4-1    2019-03-18 [1] CRAN (R 3.6.1)
##  crayon               1.3.4    2017-09-16 [1] CRAN (R 3.6.2)
##  curl                 4.3      2019-12-02 [1] CRAN (R 3.6.2)
##  DBI                  1.1.0    2019-12-15 [1] CRAN (R 3.6.2)
##  desc                 1.2.0    2018-05-01 [1] CRAN (R 3.6.2)
##  devtools             2.2.2    2020-02-17 [1] CRAN (R 3.6.2)
##  digest               0.6.25   2020-02-23 [1] CRAN (R 3.6.2)
##  dplyr              * 0.8.5    2020-03-07 [1] CRAN (R 3.6.3)
##  e1071                1.7-3    2019-11-26 [1] CRAN (R 3.6.2)
##  ellipsis             0.3.0    2019-09-20 [1] CRAN (R 3.6.2)
##  evaluate             0.14     2019-05-28 [1] CRAN (R 3.6.2)
##  extrafont            0.17     2014-12-08 [1] CRAN (R 3.6.2)
##  extrafontdb          1.0      2012-06-11 [1] CRAN (R 3.6.0)
##  fansi                0.4.1    2020-01-08 [1] CRAN (R 3.6.2)
##  farver               2.0.3    2020-01-16 [1] CRAN (R 3.6.2)
##  foreach            * 1.4.8    2020-02-09 [1] CRAN (R 3.6.3)
##  fs                   1.3.1    2019-05-06 [1] CRAN (R 3.6.2)
##  gdtools              0.2.2    2020-04-03 [1] CRAN (R 3.6.2)
```

```
##  ggplot2           * 3.3.0    2020-03-05 [1] CRAN (R 3.6.3)
##  ggpolypath          0.1.0    2016-08-10 [1] CRAN (R 3.6.3)
##  ggthemes          * 4.2.0    2019-05-13 [1] CRAN (R 3.6.3)
##  ggtree            * 1.16.6   2019-08-26 [1] standard (@1.16.6)
##  glue                1.3.2    2020-03-12 [1] CRAN (R 3.6.2)
##  gtable              0.3.0    2019-03-25 [1] CRAN (R 3.6.2)
##  hms                 0.5.3    2020-01-08 [1] CRAN (R 3.6.2)
##  hrbrthemes          0.8.0    2020-03-06 [1] CRAN (R 3.6.3)
##  htmltools           0.4.0    2019-10-04 [1] CRAN (R 3.6.2)
##  httr                1.4.1    2019-08-05 [1] CRAN (R 3.6.2)
##  INLA              * 20.03.17 2020-03-17 [1] local
##  iterators           1.0.12   2019-07-26 [1] CRAN (R 3.6.3)
##  jsonlite            1.6.1    2020-02-02 [1] CRAN (R 3.6.2)
##  KernSmooth          2.23-16  2019-10-15 [1] CRAN (R 3.6.2)
##  knitr               1.28     2020-02-06 [1] CRAN (R 3.6.3)
##  labeling            0.3      2014-08-23 [1] CRAN (R 3.6.0)
##  lattice             0.20-38  2018-11-04 [1] CRAN (R 3.6.2)
##  lazyeval            0.2.2    2019-03-15 [1] CRAN (R 3.6.2)
##  lifecycle           0.2.0    2020-03-06 [1] CRAN (R 3.6.3)
##  magrittr            1.5      2014-11-22 [1] CRAN (R 3.6.2)
##  Matrix            * 1.2-18   2019-11-27 [1] CRAN (R 3.6.2)
##  memoise             1.1.0    2017-04-21 [1] CRAN (R 3.6.2)
##  mgcv                1.8-31   2019-11-09 [1] CRAN (R 3.6.2)
##  munsell             0.5.0    2018-06-12 [1] CRAN (R 3.6.2)
##  nlme                3.1-142  2019-11-07 [1] CRAN (R 3.6.2)
##  patchwork         * 1.0.0    2019-12-01 [1] CRAN (R 3.6.2)
##  pillar              1.4.3    2019-12-20 [1] CRAN (R 3.6.2)
##  pkgbuild            1.0.6    2019-10-09 [1] CRAN (R 3.6.2)
##  pkgconfig           2.0.3    2019-09-22 [1] CRAN (R 3.6.2)
##  pkgload             1.0.2    2018-10-29 [1] CRAN (R 3.6.2)
##  prettyunits         1.1.1    2020-01-24 [1] CRAN (R 3.6.2)
##  processx            3.4.2    2020-02-09 [1] CRAN (R 3.6.2)
##  ps                  1.3.2    2020-02-13 [1] CRAN (R 3.6.2)
##  purrr               0.3.3    2019-10-18 [1] CRAN (R 3.6.2)
##  R6                  2.4.1    2019-11-12 [1] CRAN (R 3.6.2)
##  raster              3.0-12   2020-01-30 [1] CRAN (R 3.6.2)
##  Rcpp                1.0.4    2020-03-17 [1] CRAN (R 3.6.3)
##  readr             * 1.3.1    2018-12-21 [1] CRAN (R 3.6.2)
##  remotes             2.1.1    2020-02-15 [1] CRAN (R 3.6.2)
##  rgdal               1.4-8    2019-11-27 [1] CRAN (R 3.6.2)
##  rgeos               0.5-2    2019-10-03 [1] CRAN (R 3.6.3)
##  rlang               0.4.5    2020-03-01 [1] CRAN (R 3.6.3)
##  rmarkdown           2.1      2020-01-20 [1] CRAN (R 3.6.3)
##  rnaturalearth     * 0.1.0    2017-03-21 [1] CRAN (R 3.6.3)
##  rnaturalearthhires  0.2.0    2020-04-06 [1] local
##  rprojroot           1.3-2    2018-01-03 [1] CRAN (R 3.6.2)
##  rstudioapi          0.11     2020-02-07 [1] CRAN (R 3.6.2)
##  Rttf2pt1            1.3.8    2020-01-10 [1] CRAN (R 3.6.2)
##  rvcheck             0.1.8    2020-03-01 [1] standard (@0.1.8)
##  scales              1.1.0    2019-11-18 [1] CRAN (R 3.6.2)
##  scico               1.1.0    2018-11-20 [1] CRAN (R 3.6.2)
##  sessioninfo         1.1.1    2018-11-05 [1] CRAN (R 3.6.2)
##  sf                  0.8-1    2020-01-28 [1] CRAN (R 3.6.2)
##  sp                * 1.4-1    2020-02-28 [1] CRAN (R 3.6.3)
```

```
##  stringi             1.4.6    2020-02-17 [1] CRAN (R 3.6.2)
##  stringr             1.4.0    2019-02-10 [1] CRAN (R 3.6.2)
##  systemfonts         0.1.1    2019-07-01 [1] CRAN (R 3.6.3)
##  testthat            2.3.2    2020-03-02 [1] CRAN (R 3.6.3)
##  tibble              3.0.0    2020-03-30 [1] CRAN (R 3.6.2)
##  tidyr               1.0.2    2020-01-24 [1] CRAN (R 3.6.2)
##  tidyselect          1.0.0    2020-01-27 [1] CRAN (R 3.6.2)
##  tidytree            0.3.1    2019-12-14 [1] standard (@0.3.1)
##  treeio              1.8.2    2019-08-21 [1] standard (@1.8.2)
##  units               0.6-5    2019-10-08 [1] CRAN (R 3.6.2)
##  usethis             1.5.1    2019-07-04 [1] CRAN (R 3.6.2)
##  V8                  3.0.1    2020-01-22 [1] CRAN (R 3.6.2)
##  vctrs               0.2.4    2020-03-10 [1] CRAN (R 3.6.3)
##  wellknown           0.6.0    2020-01-10 [1] CRAN (R 3.6.2)
##  withr               2.1.2    2018-03-15 [1] CRAN (R 3.6.2)
##  xfun                0.12     2020-01-13 [1] CRAN (R 3.6.3)
##  yaml                2.2.1    2020-02-01 [1] CRAN (R 3.6.2)
##
## [1] C:/R/R-3.6.2/library
```

References

Lindgren, Finn, Håvard Rue, and Johan Lindström. 2011. "An Explicit Link Between Gaussian Fields and Gaussian Markov Random Fields: The Stochastic Partial Differential Equation Approach." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73 (4): 423–98.

Martins, Thiago G, Daniel Simpson, Finn Lindgren, and Håvard Rue. 2013. "Bayesian Computing with Inla: New Features." *Computational Statistics & Data Analysis* 67: 68–83.

Rue, Håvard, Sara Martino, and Nicolas Chopin. 2009. "Approximate Bayesian Inference for Latent Gaussian Models by Using Integrated Nested Laplace Approximations." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 71 (2): 319–92.

Uyeda, Josef C, Rosana Zenil-Ferguson, and Matthew W Pennell. 2018. "Rethinking Phylogenetic Comparative Methods." *Systematic Biology* 67 (6): 1091–1109.