Spatial and Functional Diversity

Ian G. Brennan

12/11/2019

Load a whole bunch of packages we (mostly) need

```r
library(phytools); library(ALA4R);
library(raster); library(rgeos);
library(sp); library(vegan)
library(FD); library(raster);
library(rgdal); library(ggmap);
library(broom); library(dplyr)
library(wesanderson); library(fields)
library(metricTester); library(picante)
library(fields); library(RColorBrewer)
library(FD)
```

We're going to turn our focus over to the Australian radiation of monitor lizards now, so we can load some existing files.

```r
#tutorial <- readRDS("~/Documents/GitHub/MonitorPhylogenomics/Trait_Evo/Goanna_Walkthrough.RDS")
tutorial <- readRDS("~/Documents/GitHub/MonitorPhylogenomics/Spatial_Walkthrough.RDS")
names(tutorial)
```

```
## [1] "distribution.data" "size.data"
```

Create a tibble from the distribution data, turn it into Site x Species tibble

```r
ygridded <- tutorial$distribution.data %>%

  ## bin into 0.5-degree bins
  dplyr::mutate(longitude=round(Longitude*2)/2, latitude=round(Latitude*2)/2) %>%

  #  ## average environmental vars within each bin
  group_by(longitude,latitude) %>%
  #  mutate(precipitationAnnual=mean(precipitationAnnual, na.rm=TRUE),
  #         temperatureAnnualMaxMean=mean(temperatureAnnualMaxMean, na.rm=TRUE)) %>%

  ## subset to vars of interest
  dplyr::select(longitude, latitude, Name_in_Tree) %>%

  ## take one row per cell per species (presence)
  distinct() %>%

  ## calculate species richness
  dplyr::mutate(richness=n()) %>%

  ## convert to wide format (sites by species)
  dplyr::mutate(present=1) %>%
  do(tidyr::spread(data=., key=Name_in_Tree, value=present, fill=0)) %>%
  ungroup()
```

Have a quick look at the Site x Species tibble, then translate it to a data frame we can manipulate normally.

```
gridded.dist <- as.data.frame(ygridded)
gridded.dist[1:5, 1:7]
```

```
##   longitude latitude richness Varanus_gouldii Varanus_brevicauda
## 1     113.0    -25.0        1               1                 NA
## 2     113.5    -26.0        6               1                  1
## 3     113.5    -25.5        2               1                 NA
## 4     113.5    -25.0        2               1                 NA
## 5     113.5    -24.5        1              NA                 NA
##   Varanus_caudolineatus Varanus_eremius
## 1                    NA              NA
## 2                     1               1
## 3                    NA              NA
## 4                    NA               1
## 5                    NA               1
```

Lots of sites don't have any records, and are listed as NAs. This won't jibe with our code, so switch NA to 0.

```
gridded.dist[is.na(gridded.dist)] <- 0 # make NAs 0
gridded.dist <- filter(gridded.dist, !richness==1) # remove sites with just one taxon
gridded.dist <- filter(gridded.dist, latitude <= -11);
gridded.dist <- filter(gridded.dist, longitude >= 113.5)
gdist <- gridded.dist[ , 4:ncol(gridded.dist)]
```

Make the order of the trait dataframe match the order of the Site x Species DF

```
goanna.trait <- tutorial$size.data[match(colnames(gdist),
                                   tutorial$size.data$Name_in_Tree),]
goanna.frame <- data.frame(SVL = goanna.trait$Body_Length);
rownames(goanna.frame) <- goanna.trait$Name_in_Tree
```

Run the Functional Diversity function and extract two estimates of fuctional diversity: the Rao's Quadratic value, and FDis

```
best <- dbFD(log(goanna.frame), gdist)
```

```
## FEVe: Could not be calculated for communities with <3 functionally singular species.
## FRic: Only one continuous trait or dimension in 'x'. FRic was measured as the range, NOT as the conve
## FDiv: Cannot not be computed when 'x' contains one single continuous trait or dimension.
```

```
RQ.scores <- best$RaoQ
FDis.scores <- best$FDis
res.table <- cbind.data.frame(latitude=gridded.dist$latitude,
                              longitude=gridded.dist$longitude,
                              RaoQ=best$RaoQ, FDis=best$FDis)
```

Read in your shapefile

```
oz <- shapefile("~/Documents/GitHub/MonitorPhylogenomics/Map_Shapefiles/Australia.shp")
plot(oz)
```

Set up a raster "template" for a 0.5 degree grid

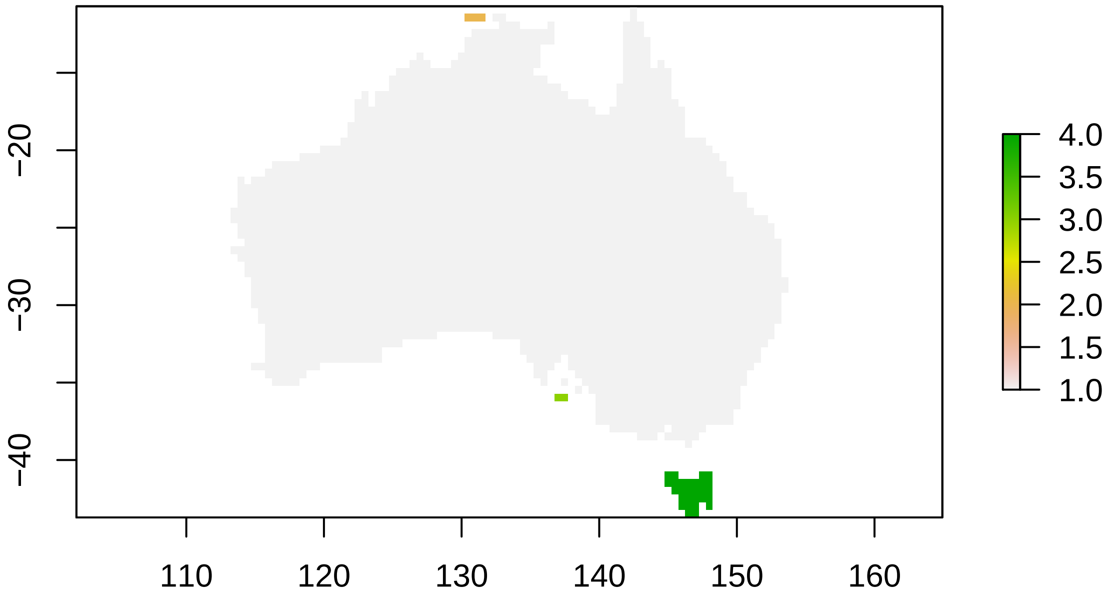
```
ext <- extent(113.2244, 153.6242, -43.64806, -10.70667)
gridsize <- 0.5
r <- raster(ext, res=gridsize)
```

Rasterize the shapefile

```
rr <- rasterize(oz, r)
```

Plot raster

```
plot(rr)
```

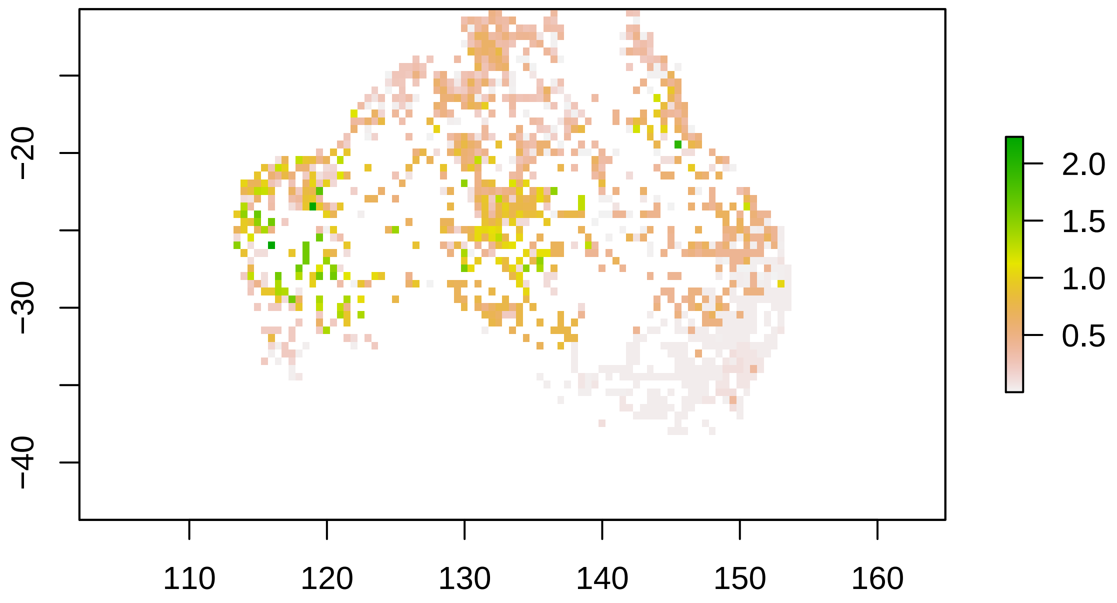


```
rr.cells <- xyFromCell(rr, 1:length(rr));
rr.cells <- as.data.frame(rr.cells)
rr.cells$x <- round(rr.cells$x*2)/2;
rr.cells$y <- round(rr.cells$y*2)/2
colnames(rr.cells) <- c("longitude", "latitude")
head(rr.cells)
```

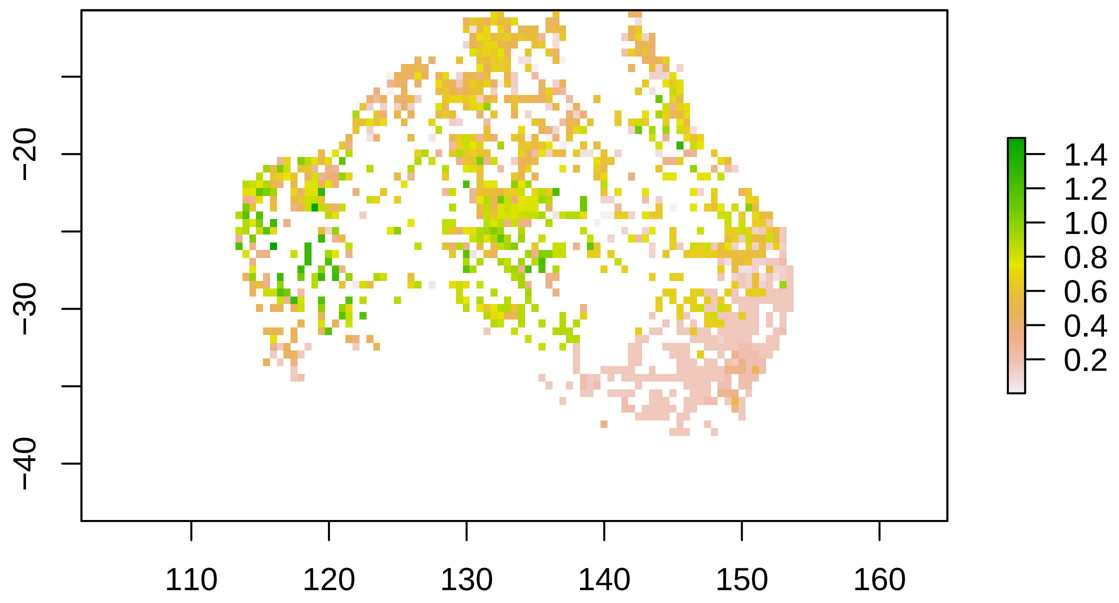
```
##   longitude latitude
## 1      113.5      -11
## 2      114.0      -11
## 3      114.5      -11
## 4      115.0      -11
## 5      115.5      -11
## 6      116.0      -11
```

Fill raster cells by FD values (Rao's Q, FDis), and visualize it.

```
combo.Q <- left_join(rr.cells,
                     res.table,
                     by=c("longitude", "latitude"))
FDras <- rr
values(FDras) <- combo.Q$RaoQ
plot(FDras)
```

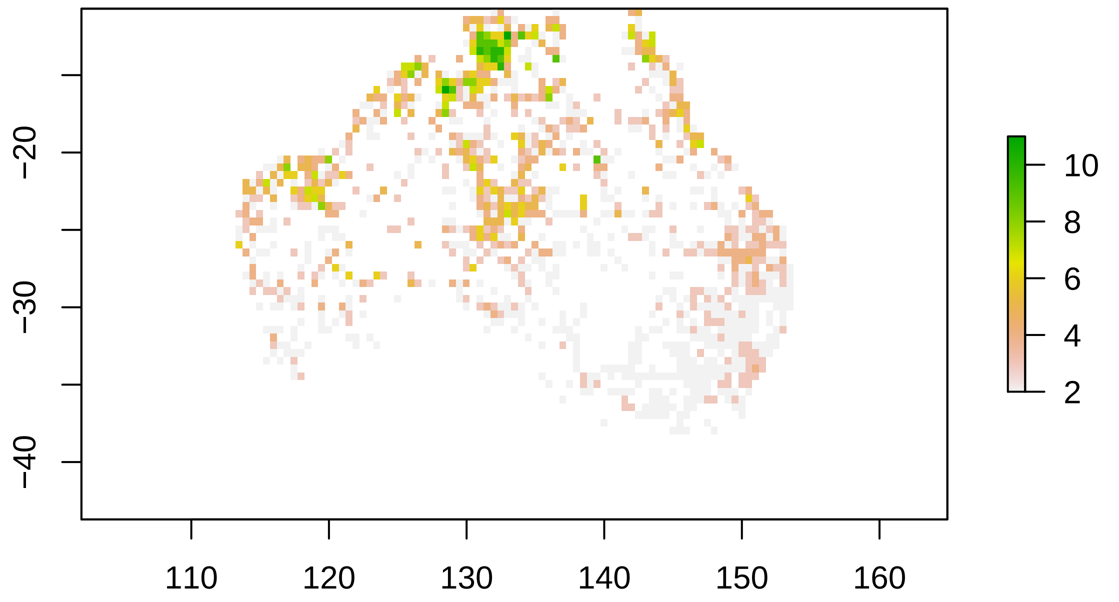


```
FDisras <- rr
values(FDisras) <- combo.Q$FDis
plot(FDisras)
```



Fill raster cells by richness and visualize it.

```
combo.R <- left_join(rr.cells,
                     gridded.dist[,1:3],
                     by=c("longitude", "latitude"))
RICHras <- rr
values(RICHras) <- combo.R$richness
plot(RICHras)
```



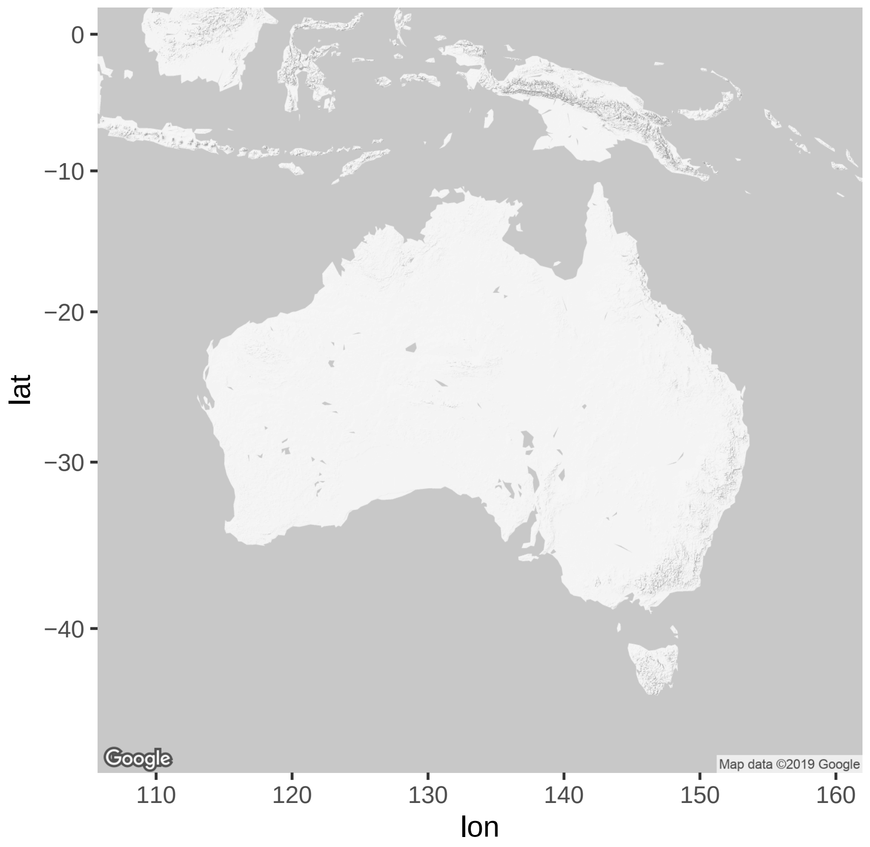
We can do this a bit prettier, start by establishing a map

```
graymap <- get_googlemap(center = "Australia", zoom = 4, style = 'https://maps.googleapis.com/maps/api/
```

```
## Source : https://maps.googleapis.com/maps/api/staticmap?center=Australia&zoom=4&size=640x640&scale=2
```

```
## Source : https://maps.googleapis.com/maps/api/geocode/json?address=Australia&key=xxx
```

```
ggmap(graymap)
```



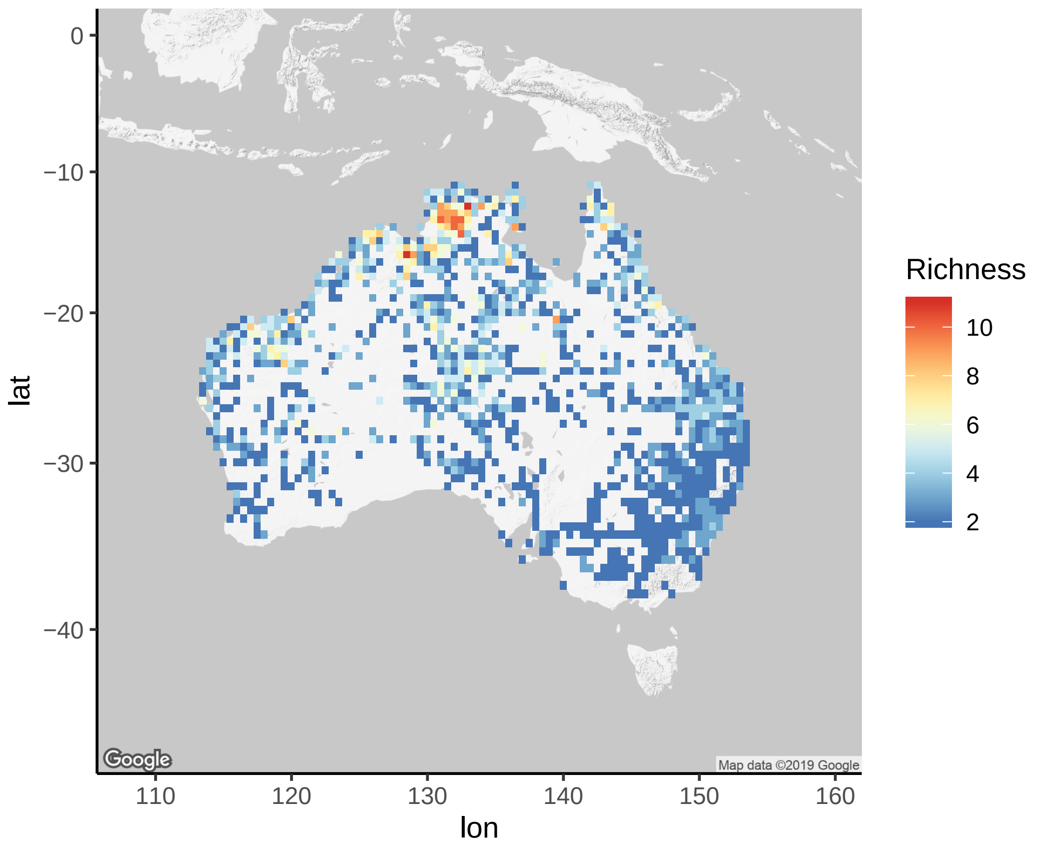
Create richness polygons

```
RICHpoly <- rasterToPolygons(RICHras);
max.colors <- length(unique(RICHpoly$layer));
filled.RICH <- rep(RICHpoly$layer, each=5)
# 'each' is important, otherwise the polygon values get screwed up
```

Set the color palette length and the breakpoints

```
RICHras@data@values[is.na(RICHras@data@values)] <- 0
pal.length <- abs(min(RICHras@data@values) - max(RICHras@data@values)) * 10
myBreaks <- c(seq(min(RICHras@data@values), 0, length.out=ceiling(pal.length/2) + 1),
              seq(max(RICHras@data@values)/pal.length, max(RICHras@data@values),
                  length.out=floor(pal.length/2)))
```

```
ggmap(graymap) +
  geom_polygon(data = RICHpoly,
               aes(x = long, y = lat,
                   group = group,
                   fill = filled.RICH),
               size = 0, alpha = 1)  +
  scale_fill_gradientn("Richness",
                       colors = rev(colorRampPalette(
                         brewer.pal(9, "RdYlBu"))(max.colors))) +
  theme_classic()
```



Do the same for functional diversity (choose either FDras or FDisras)

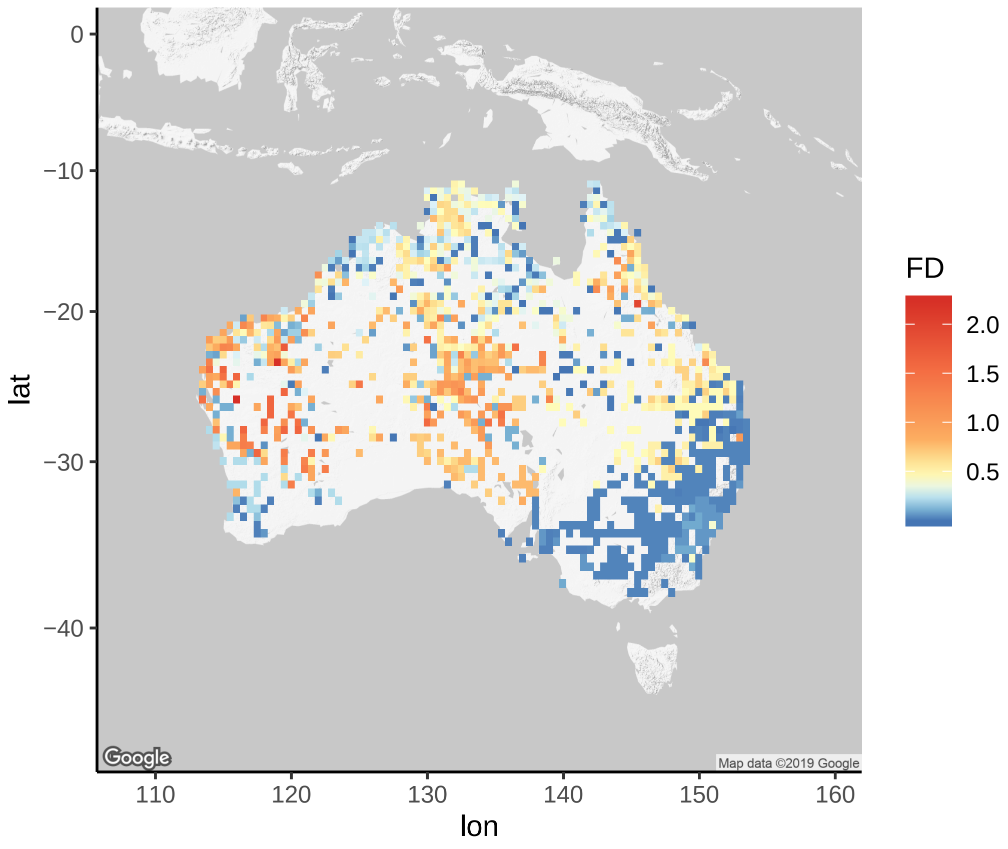
```r
FDpoly <- rasterToPolygons(FDras);
#FDpoly <- rasterToPolygons(FDisras)
max.colors <- length(unique(FDpoly$layer));
filled.FD <- rep(FDpoly$layer, each=5)
# 'each' is important, otherwise the polygon values get screwed up
```

Set the color palette length and the breakpoints

```r
FDras@data@values[is.na(FDras@data@values)] <- 0
pal.length <- abs(min(FDras@data@values) - max(FDras@data@values)) * 10
myBreaks <- c(seq(min(FDras@data@values), 0, length.out=ceiling(pal.length/2) + 1),
              seq(max(FDras@data@values)/pal.length, max(FDras@data@values),
                  length.out=floor(pal.length/2)))
#FDras@data@values[which(FDras@data@values == 0)] <- "NA"
```

```r
ggmap(graymap) +
  geom_polygon(data = FDpoly,
               aes(x = long,
                   y = lat,
                   group = group,
                   fill = filled.FD),
               size = 0, alpha = 1)  +
  scale_fill_gradientn("FD",
                       values=scales::rescale(c(min(res.table$RaoQ),
                                                mean(res.table$RaoQ)/2,
                                                mean(res.table$RaoQ),
                                                mean(res.table$RaoQ)*2,
                                                max(res.table$RaoQ))),
                       colors = rev(colorRampPalette(
                         brewer.pal(9, "RdYlBu"))(max.colors))) +

  theme_classic()
```



We've plotted richness and functional diversity, but we'd like to know if either is significantly different than scores from random communities.
We've already got a community matrix ('gridded.dist'), so just copy that.

```
cm <- gridded.dist
```

Create an empty raster or two

```
richness.raster <- rr; richness.raster@data@values[] <- 0
fd.raster <- rr; fd.raster@data@values[] <- 0
```

Add the FD and Richness scores to your community matrix

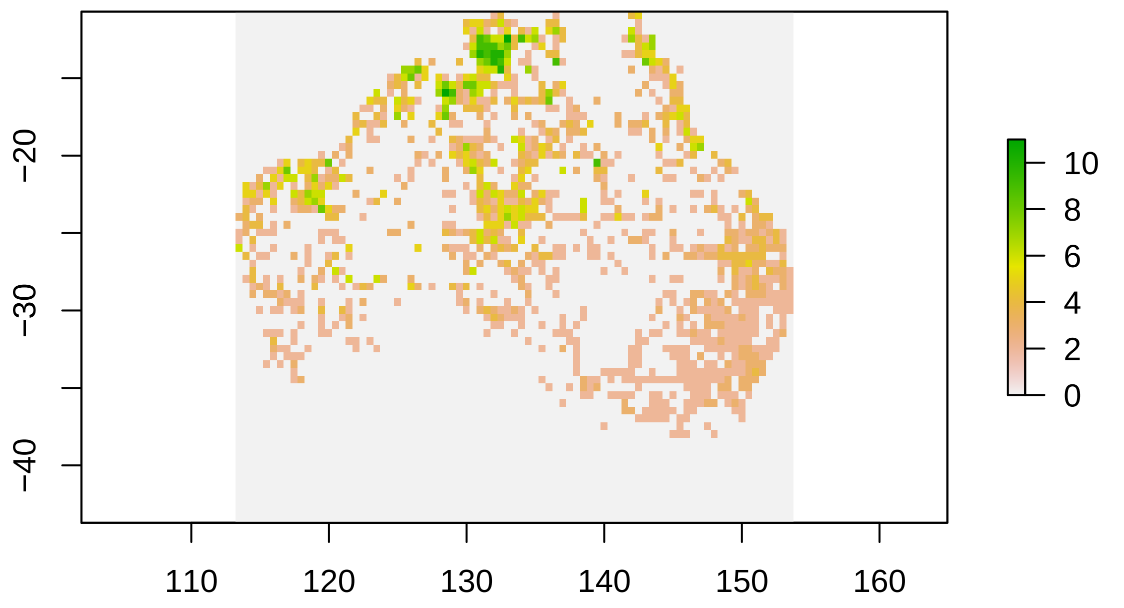
```
pre.rr <- left_join(rr.cells, cm, by=c("latitude", "longitude")); pre.rr[is.na(pre.rr)] <- 0
pre.fd <- left_join(rr.cells, res.table, by=c("latitude", "longitude"))
    pre.fd <- left_join(pre.fd, cm, by=c("latitude", "longitude")); pre.fd[is.na(pre.fd)] <- 0
```

Pass along the values from the matrices to your rasters

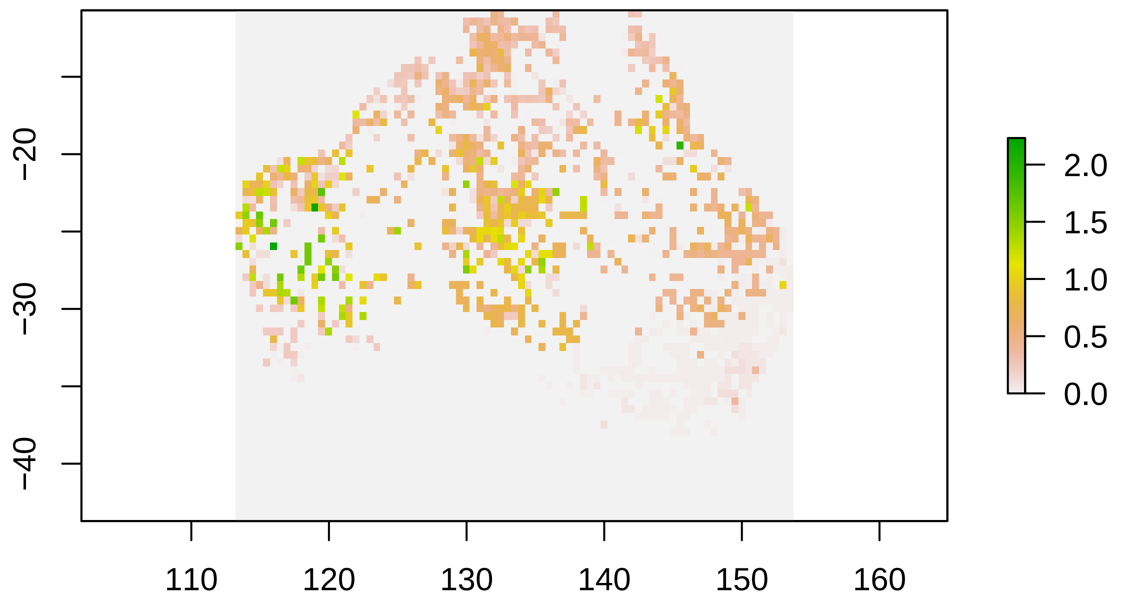
```
richness.raster@data@values <- pre.rr$richness
fd.raster@data@values <- pre.fd$RaoQ
#fd.raster@data@values <- pre.fd$FDis
```

Quickly plot them again to make sure they make sense and nothing funny happened

```r
plot(richness.raster)
```



```r
plot(fd.raster)
```



Identify which cells have richness values > 1 (more than one taxon occupying it)
Or identify which cells have functional diversity values > 0 (so we can compare)

```r
cells.rich <- which(richness.raster@data@values > 1)
cells.fd <- which(fd.raster@data@values > 0)
```

Make your blank site x species matrices by choosing cells with richness (>1) and FD (>0)

```r
input.rr <- pre.rr[,4:ncol(pre.rr)];
input.rr <- input.rr[which(rowSums(input.rr) > 1),]
input.fd <- pre.fd[which(pre.fd$RaoQ > 0), 6:ncol(pre.fd)]
#input.fd <- pre.fd[which(pre.fd$FDis > 0), 6:ncol(pre.fd)]
```

We can check this quickly by showing how many sites there were (including those with no observations), and how many we now have (including only those with observations)

```
## [1] "5346 total sites"
```

```
## [1] "1210 sites have >1 species present"
```

Get the x (longitude) y (latitude) coordinates of those cells

```
coords.rich <- xyFromCell(richness.raster, cells.rich)
coords.fd <- xyFromCell(fd.raster, cells.fd)
```

Now create the greater circle distance (in meters) for each raster. This is an important input step for our

```
# for richness
gc.dist.rich <- rdist.earth(coords.rich);
rownames(gc.dist.rich) <- cells.rich;
colnames(gc.dist.rich) <- cells.rich;
diag(gc.dist.rich) <- 0

# for functional diversity
gc.dist.fd <- rdist.earth(coords.fd);
rownames(gc.dist.fd) <- cells.fd;
colnames(gc.dist.fd) <- cells.fd;
diag(gc.dist.fd) <- 0
```

We'll need to source the dispersal null metric function

```
source("~/Documents/GitHub/MonitorPhylogenomics/DispersalNullModel.R")
```

And create an additional function to run this null model repeatedly

```
library(parallel)
nullFD <- function(n.model, n.iter,
                   method=c("randomizeMatrix", "DNM"),
                   cores, trait.data, measure=c("RaoQ", "FDis", "Richness"),
                   great.circle){

  beginning <- Sys.time()
  Rao.table <- NULL

  if(method=="randomizeMatrix"){
    swap <- mclapply(1:n.iter, function(x) {
      randomizeMatrix(input.fd,
                      null.model=n.model,
                      iterations=10)},
      mc.cores=cores)
    swap.res <- mclapply(1:length(swap), function(x) {
      dbFD(trait.frame, swap[[x]])}, mc.cores=8)

    for(j in 1:length(swap.res)){
      Rao.table <- cbind(Rao.table, swap.res[[j]]$RaoQ)
    }
  }
  else if(method=="DNM"){
    swap <- mclapply(1:n.iter, function(x) {
      DNM(input.fd, tree=NA,
          great.circle, abundance.matters=F,
          abundance.assigned="directly")}, mc.cores=cores)
    swap <- Filter(function(x) length(x)>1, swap)
    # Get FD
    if (measure=="RaoQ"){
      swap.res <- mclapply(1:length(swap), function(x) {
        dbFD(trait.data, swap[[x]])}, mc.cores=8)
      for(j in 1:length(swap.res)){
        Rao.table <- cbind(Rao.table, swap.res[[j]]$RaoQ)
      }
    }
    else if (measure=="FDis"){
      swap.res <- mclapply(1:length(swap), function(x) {
        dbFD(trait.data, swap[[x]])}, mc.cores=8)
      for(j in 1:length(swap.res)){
        Rao.table <- cbind(Rao.table, swap.res[[j]]$FDis)
      }
    }
    # or Get RICHNESS
    else if (measure=="Richness"){
      swap.res <- mclapply(1:length(swap), function(x) {
        rowSums(swap[[x]])}, mc.cores=8)
      for (j in 1:length(swap.res)){
        Rao.table <- cbind(Rao.table, swap.res[[j]])
      }
    }

    print(paste("you attempted", n.iter,
                "iterations, and you got",
                length(swap), "simulations"))

  }

  end <- Sys.time()
  duration <- format(end-beginning)
  print(paste("Computation time to fit", n.iter,
              method, "null models:", duration))

  Rao.table <- as.data.frame(Rao.table);
  Raw.table <- Rao.table
  Rao.table <- cbind(Rao.table,
                     sim.mean=rowMeans(Rao.table))
  Rao.table <- cbind(Rao.table,
                     sim.sd=apply(Raw.table, 1, sd))
  #Rao.table <- cbind(Rao.table, emp.val=) # I could add in the empirical values (FD)
  #Rao.table <- cbind(Rao.table, ses=apply(Rao.table, 1, (Rao.table[,"mean"]))) # then I could calculat
  return(Rao.table)
}
```

Run the function a lot. I'll just quickly do 50 simulations here, but we should do many many more.

```
RQ <- nullFD(n.model=NULL,
             n.iter=50,
             method="DNM",
             cores=6,
             trait.data=log(goanna.frame),
             measure="RaoQ",
             great.circle = gc.dist.fd)
```

If you don't have time to run those functions above, you'll want to read in the files

```
RQ <- readRDS(file="~/Documents/GitHub/MonitorPhylogenomics/SimulatedGoanna_RaoQ_logData.RDS")
SESras <- readRDS(file="~/Documents/GitHub/MonitorPhylogenomics/SimulatedGoanna_RaoQ_logData_SES_raster
```

Now we need to add the empirical FD (or richness) values to this data frame

```
#RQ <- cbind(RQ, emp.val=res.table$RaoQ)
RQ <- cbind(RQ, emp.val=res.table$FDis)
```

Then get standard effect sizes (SES) for each sell across all simulations

```
ses.vec <- NULL
for(k in 1:nrow(RQ)){
  curr <- RQ[k,]
  ses <- (curr$emp.val - curr$sim.mean) / curr$sim.sd
  ses.vec <- append(ses.vec, ses)
}
# bind it to the simulation dataframe
RQ <- cbind(RQ, ses=ses.vec)
```

Make a table of the ses values with the coordinates of each cell

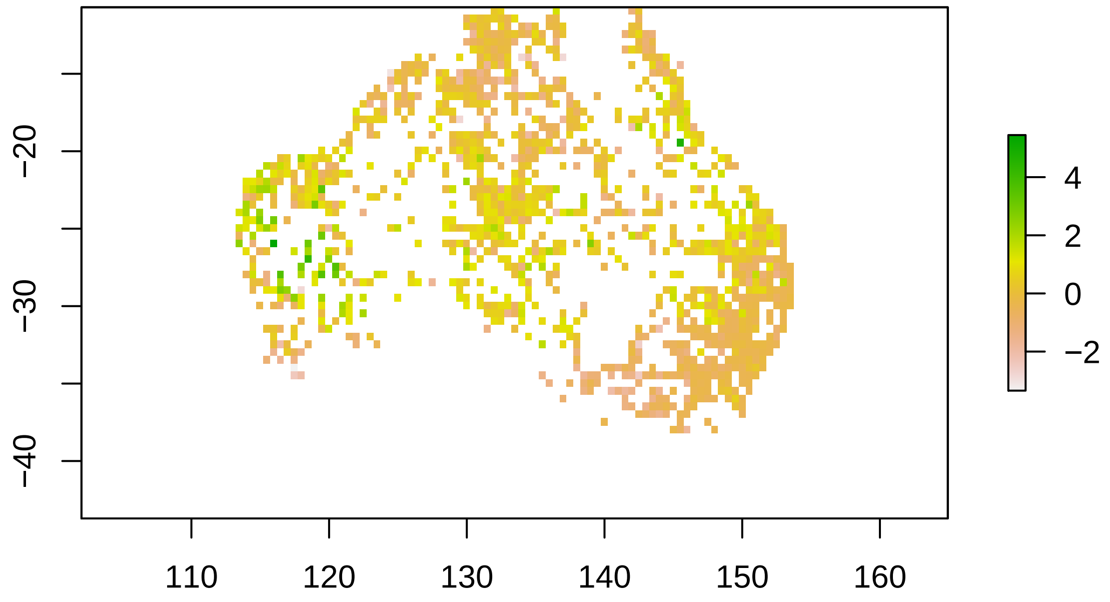
```
ses.table <- cbind.data.frame(latitude=gridded.dist$latitude, longitude=gridded.dist$longitude, SES=RQ$
```

Bind the table with the empty raster cells we set up earlier, and make any NA values 0.

```
combo.SES <- left_join(rr.cells,
                       ses.table,
                       by=c("latitude", "longitude"))
```

Make an empty raster frame for the ses values to go into
Dump them into the raster
And plot it to make sure it makes sense

```
SESras <- rr;
SESras@data@values[] <- 0
SESras@data@values <- combo.SES$SES
#values(SESras) <- combo.SES$SES
plot(SESras)
```

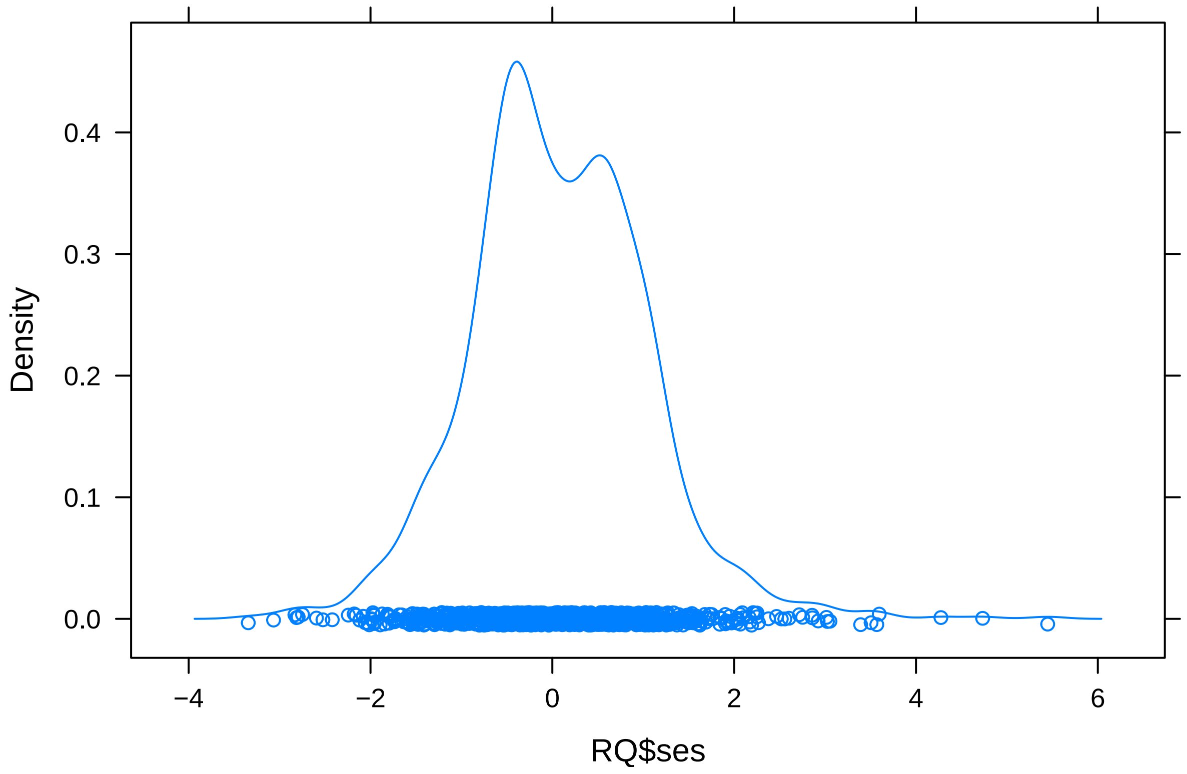


If you've been running all these steps, you'll want to save the output

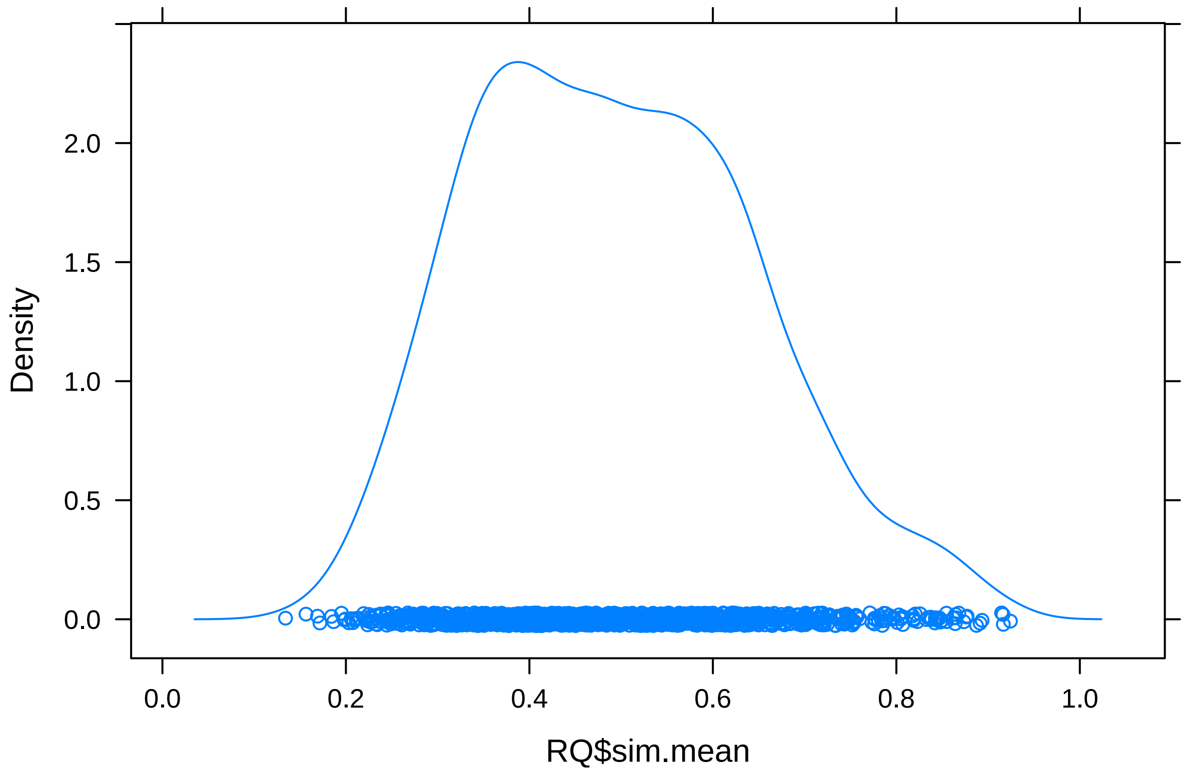
```
saveRDS(RQ, file="~/Documents/GitHub/MonitorPhylogenomics/SimulatedGoanna_RaoQ_logData.RDS")
saveRDS(SESras, file="~/Documents/GitHub/MonitorPhylogenomics/SimulatedGoanna_RaoQ_logData_SES_raster.RI
```

Have a quick look at some of the parameters

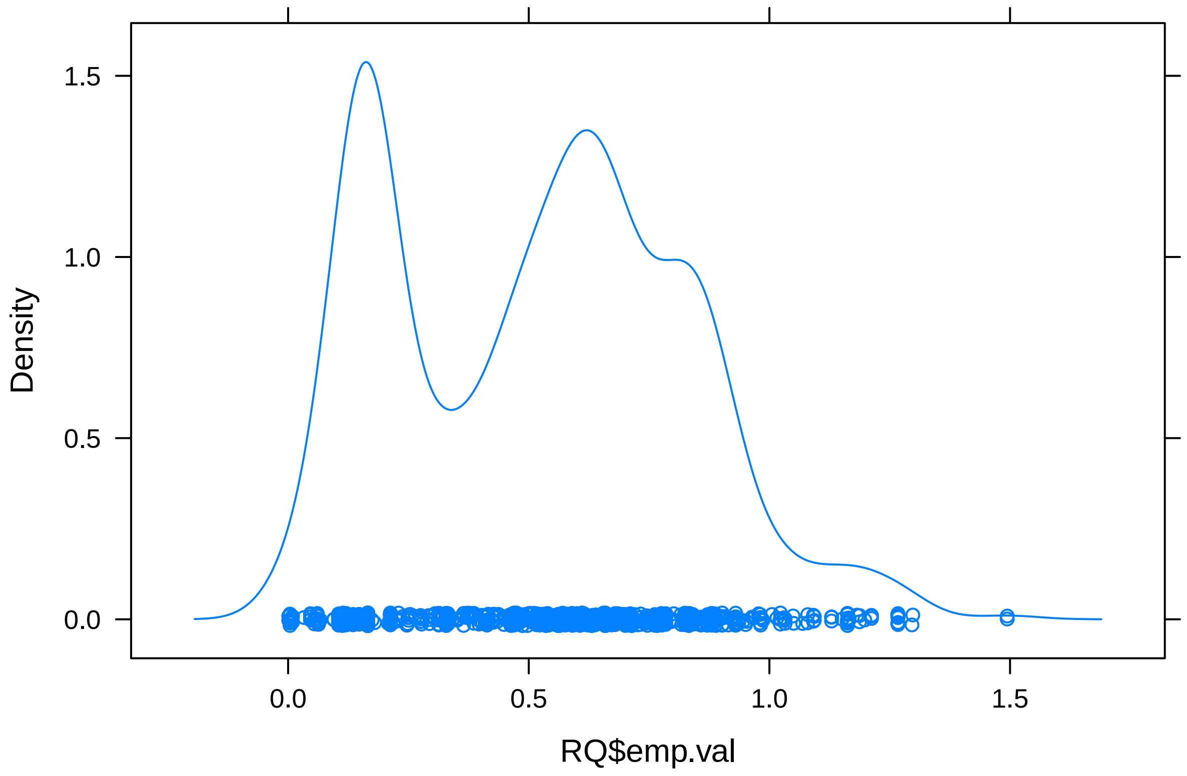
```
densityplot(RQ$ses)
```



```
densityplot(RQ$sim.mean)
```



```
densityplot(RQ$emp.val)
```



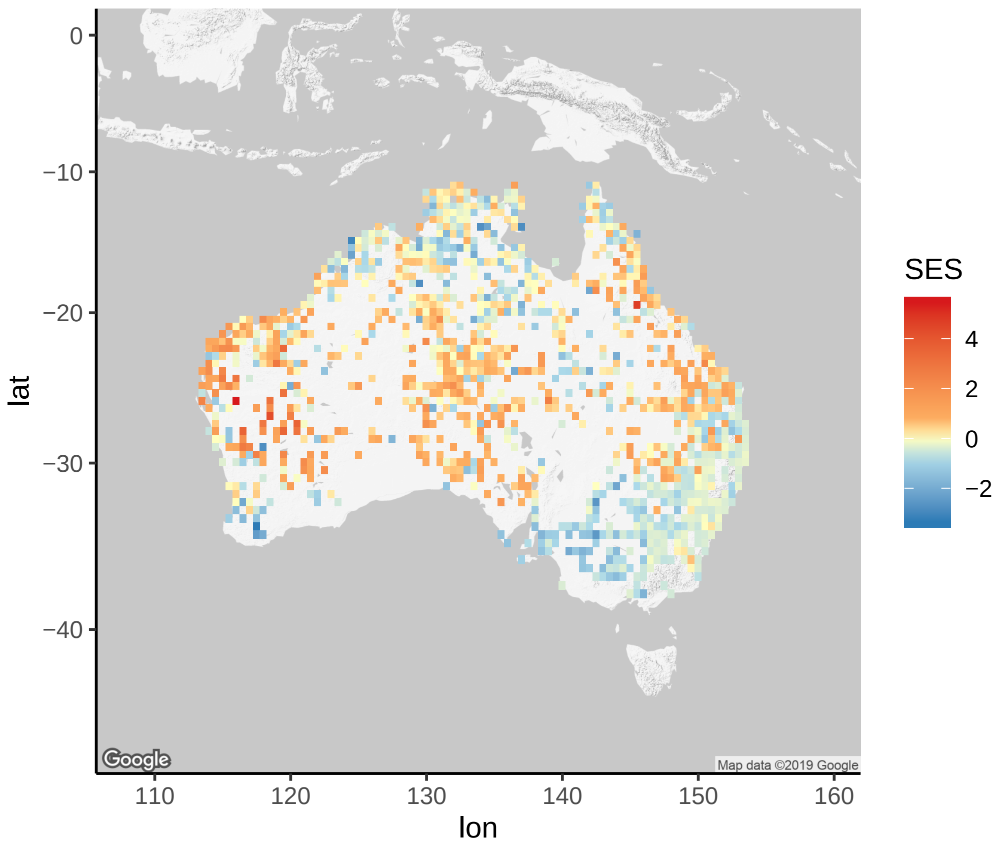
Translate the SES raster into polygons for plotting with ggmap

```r
SESpoly <- rasterToPolygons(SESras);
max.colors <- length(unique(SESpoly$layer));
filled.SES <- rep(SESpoly$layer, each=5)
```

Lastly plot the map of SES (functional diversity)

```r
ggmap(graymap) +
  geom_polygon(data = SESpoly,
               aes(x = long, y = lat, group = group,
                   fill = filled.SES), size = 0, alpha = 1)  +
  scale_fill_gradientn("SES", values=scales::rescale(c(min(ses.table$SES),
                                                       #min(ses.table$SES)/2,
                                                       -0.8,
                                                       0,
                                                       #max(ses.table$SES)/2,
                                                       0.8,
                                                       max(ses.table$SES))),
                      colors = rev(brewer.pal(5, "RdYlBu"))) +
  theme_classic()
```

```
## Regions defined for each Polygons
```



We want to know if the difference in simulated and observed FD values is signficant. So we'll create a function to calculate the confidence interval of the SES.

```r
confidence_interval <- function(vector, interval) {
  # Standard deviation of sample
  vec_sd <- sd(vector)
  # Sample size
  n <- length(vector)
  # Mean of sample
  vec_mean <- mean(vector)
  # Error according to t distribution
  error <- qt((interval + 1)/2, df = n - 1) * vec_sd / sqrt(n)
  # Confidence interval as a vector
  result <- c("lower" = vec_mean - error, "upper" = vec_mean + error,
              "error" = error, "mean" = vec_mean, "sd" = vec_sd, "N" = n)
  return(result)
}
```

Can also be calculated as:

upper = mean + (error * 1.96)

lower = mean - (error * 1.96)

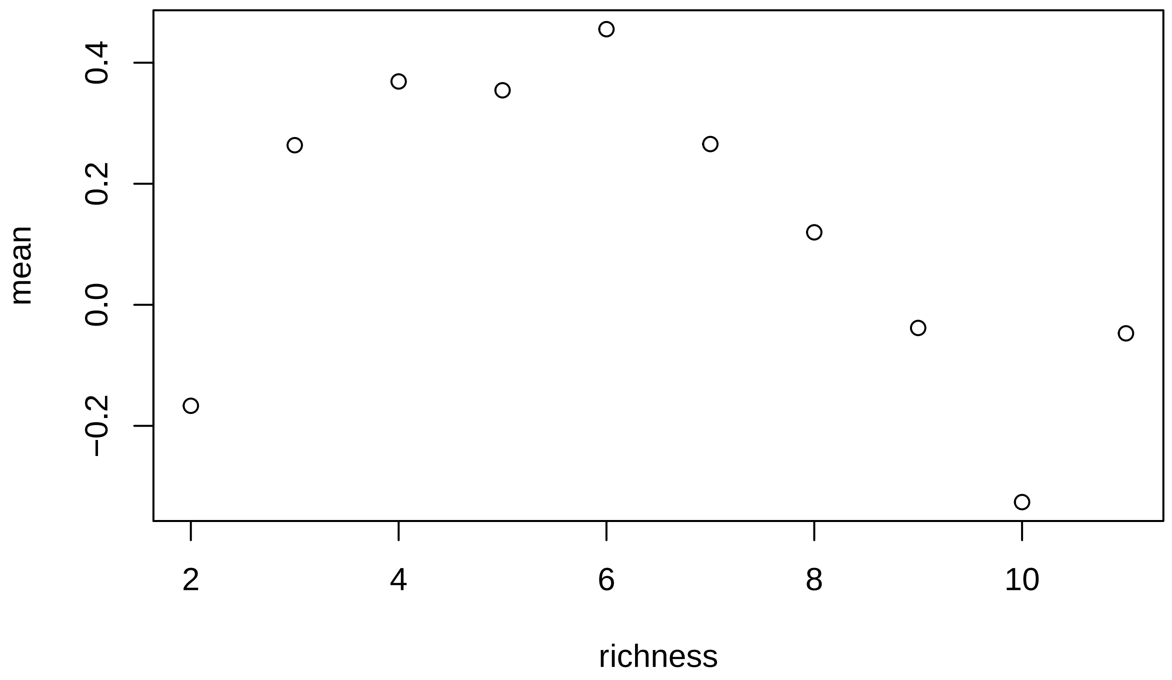
```
CIall <- confidence_interval(ses.table$SES, 0.95); CIall
```

```
##        lower        upper        error         mean           sd
## 1.765771e-02 1.275356e-01 5.493893e-02 7.259663e-02 9.740691e-01
##            N
## 1.210000e+03
```

```
CIall["richness"] <- 1
```

```
siteRICH <- left_join(ses.table,
                      gridded.dist[,1:3],
                      by=c("longitude", "latitude"))
```

```
CIses <- NULL
for (i in min(siteRICH$richness):max(siteRICH$richness)){
  curr.rich <- filter(siteRICH, richness == i)
  CIses <- rbind(CIses, confidence_interval(curr.rich$SES, 0.95))
}
CIses <- data.frame(CIses)
CIses$richness <- 2:11
plot(data=CIses, mean ~ richness)
```

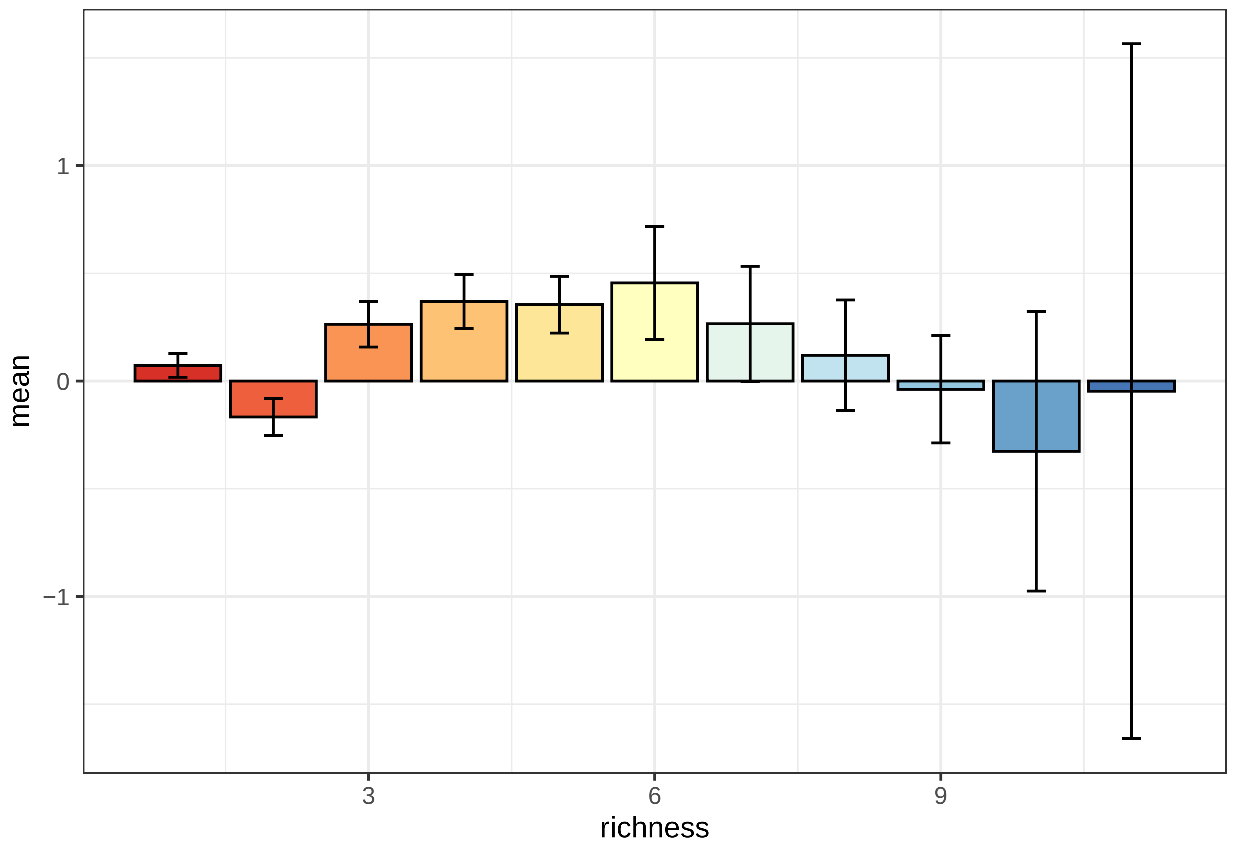


Add the confidence interval for SES across the whole continent to the individual communities

```r
CIses <- rbind(CIses, CIall); CIses
```

```
##          lower       upper      error        mean        sd    N richness
## 1  -0.25265258 -0.08109514 0.08577872 -0.16687386 1.0689687  599        2
## 2   0.15781727  0.36964206 0.10591239  0.26372966 0.8588049  255        3
## 3   0.24372979  0.49436858 0.12531940  0.36904918 0.8026238  160        4
## 4   0.22269865  0.48620457 0.13175296  0.35445161 0.5996291   82        5
## 5   0.19333878  0.71750885 0.26208504  0.45542381 0.9877488   57        6
## 6  -0.00146655  0.53272406 0.26709530  0.26562875 0.6612764   26        7
## 7  -0.13678194  0.37627128 0.25652661  0.11974467 0.4442921   14        8
## 8  -0.28734231  0.21086400 0.24910315 -0.03823916 0.3482222   10        9
## 9  -0.97490634  0.32296756 0.64893695 -0.32596939 0.5226349    5       10
## 10 -1.65996651  1.56567795 1.61282223 -0.04714428 0.1795088    2       11
## 11  0.01765771  0.12753556 0.05493893  0.07259663 0.9740691 1210        1
```

```r
library(RColorBrewer)
ggplot(CIses, aes(x=richness, y=mean)) +
  geom_bar(stat="identity", color="black",
           position=position_dodge(),
           fill = colorRampPalette(brewer.pal(9, "RdYlBu"))(11)) +
  geom_errorbar(aes(ymin=lower, ymax=upper), width=.2,
                position=position_dodge(.9)) +
  theme_bw()
```


1
0
−1
mean
3
6
9
richness

Spatial and Function Diversity of Marsupial Carnivores

We can do run the same analyses to see what the patterns of richness and functional diversity are for dasyuromorph and peramelamorph marsupials.

```
marsupial.tutorial <- readRDS("~/Documents/GitHub/MonitorPhylogenomics/Marsupial_Walkthrough.RDS")
names(marsupial.tutorial)
```

```
## [1] "marsupial.distribution" "marsupial.sizes"
```

Create a tibble from the distribution data, turn it into Site x Species tibble

```
mgridded <- marsupial.tutorial$marsupial.distribution %>%

  ## bin into 0.5-degree bins
  dplyr::mutate(longitude=round(Longitude*2)/2, latitude=round(Latitude*2)/2) %>%

  #  ## average environmental vars within each bin
  group_by(longitude,latitude) %>%
  #  mutate(precipitationAnnual=mean(precipitationAnnual, na.rm=TRUE),
  #          temperatureAnnualMaxMean=mean(temperatureAnnualMaxMean, na.rm=TRUE)) %>%

  ## subset to vars of interest
  dplyr::select(longitude, latitude, Name_in_Tree) %>%

  ## take one row per cell per species (presence)
  distinct() %>%

  ## calculate species richness
  dplyr::mutate(richness=n()) %>%

  ## convert to wide format (sites by species)
  dplyr::mutate(present=1) %>%
  do(tidyr::spread(data=., key=Name_in_Tree, value=present, fill=0)) %>%
  ungroup()
```

Have a quick look at the Site x Species tibble, then translate it to a data frame we can manipulate normally.

```
gridded.dist <- as.data.frame(mgridded)
gridded.dist[1:5, 1:7]
```

```
##   longitude latitude richness Sminthopsis.dolichura Perameles.bougainville
## 1     113.0    -25.5        1                     1                     NA
## 2     113.0    -25.0        1                    NA                      1
## 3     113.5    -26.5        1                     1                     NA
## 4     113.5    -26.0        2                     1                      1
## 5     113.5    -25.5        2                     1                     NA
##   Sminthopsis.hirtipes Sminthopsis.crassicaudata
## 1                   NA                        NA
## 2                   NA                        NA
## 3                   NA                        NA
## 4                   NA                        NA
## 5                    1                        NA
```

Lots of sites don't have any records, and are listed as NAs. This won't jibe with our code, so switch NA to 0.

```
gridded.dist[is.na(gridded.dist)] <- 0 # make NAs 0
gridded.dist <- filter(gridded.dist, !richness==1) # remove sites with just one taxon
gridded.dist <- filter(gridded.dist, latitude <= -11);
gridded.dist <- filter(gridded.dist, longitude >= 113.5)
gdist <- gridded.dist[ , 4:ncol(gridded.dist)]
```

Make the order of the trait dataframe match the order of the Site x Species DF

```
marsupial.trait <- marsupial.tutorial$marsupial.sizes[match(colnames(gdist),
                                   marsupial.tutorial$marsupial.sizes$Name_in_Tree),]
marsupial.frame <- data.frame(SVL = marsupial.trait$Body_Length);
rownames(marsupial.frame) <- marsupial.trait$Name_in_Tree
```

Run the Functional Diversity function and extract two estimates of fuctional diversity: the Rao's Quadratic value, and FDis

```
best <- dbFD(log(marsupial.frame), gdist)
```

```
## FEVe: Could not be calculated for communities with <3 functionally singular species.
## FRic: Only one continuous trait or dimension in 'x'. FRic was measured as the range, NOT as the conve
## FDiv: Cannot not be computed when 'x' contains one single continuous trait or dimension.
```

```
RQ.scores <- best$RaoQ
FDis.scores <- best$FDis
res.table <- cbind.data.frame(latitude=gridded.dist$latitude,
                              longitude=gridded.dist$longitude,
                              RaoQ=best$RaoQ, FDis=best$FDis)
```

Read in your shapefile

```
oz <- shapefile("~/Documents/GitHub/MonitorPhylogenomics/Map_Shapefiles/Australia.shp")
plot(oz)
```

Set up a raster "template" for a 0.5 degree grid

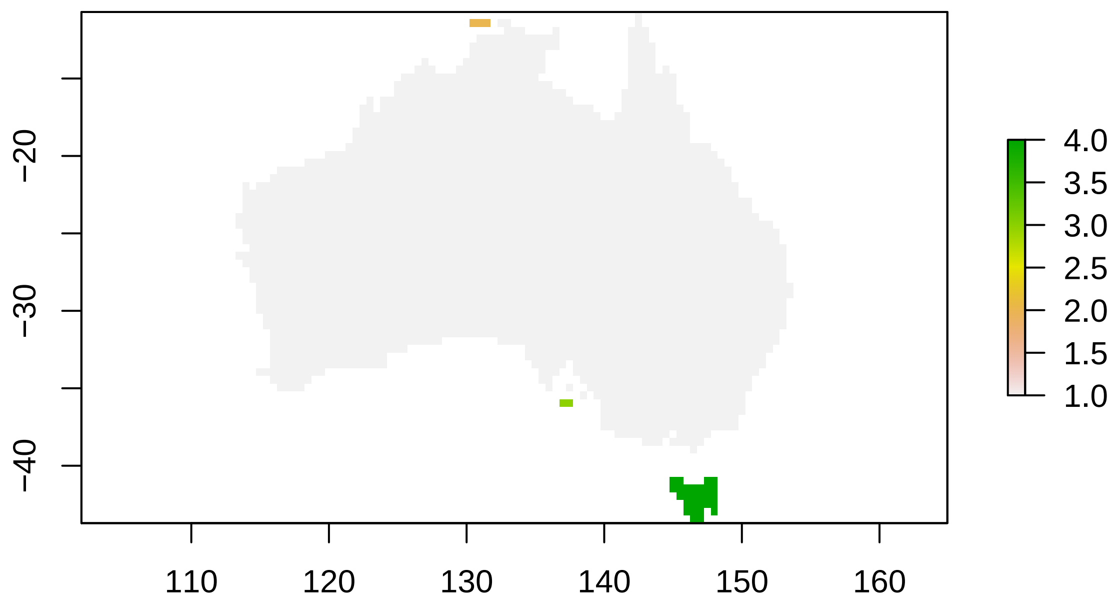
```
ext <- extent(113.2244, 153.6242, -43.64806, -10.70667)
gridsize <- 0.5
r <- raster(ext, res=gridsize)
```

Rasterize the shapefile

```
rr <- rasterize(oz, r)
```

Plot raster

```
plot(rr)
```

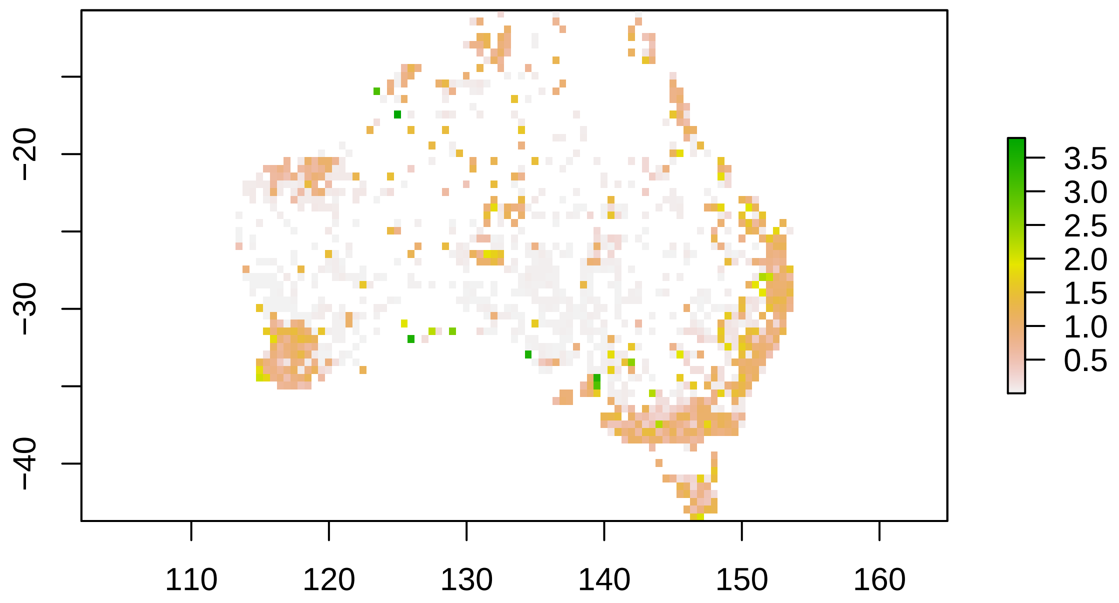


```
rr.cells <- xyFromCell(rr, 1:length(rr));
rr.cells <- as.data.frame(rr.cells)
rr.cells$x <- round(rr.cells$x*2)/2;
rr.cells$y <- round(rr.cells$y*2)/2
colnames(rr.cells) <- c("longitude", "latitude")
head(rr.cells)
```

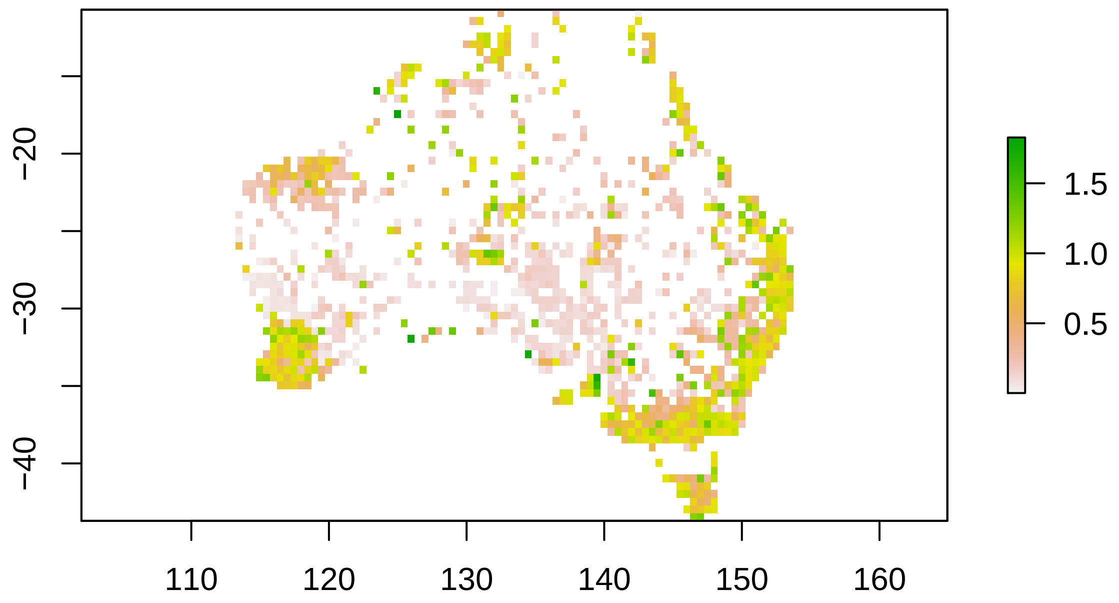
```
##   longitude latitude
## 1     113.5      -11
## 2     114.0      -11
## 3     114.5      -11
## 4     115.0      -11
## 5     115.5      -11
## 6     116.0      -11
```

Fill raster cells by FD values (Rao's Q, FDis), and visualize it.

```
combo.Q <- left_join(rr.cells,
                     res.table,
                     by=c("longitude", "latitude"))
FDras <- rr
values(FDras) <- combo.Q$RaoQ
plot(FDras)
```

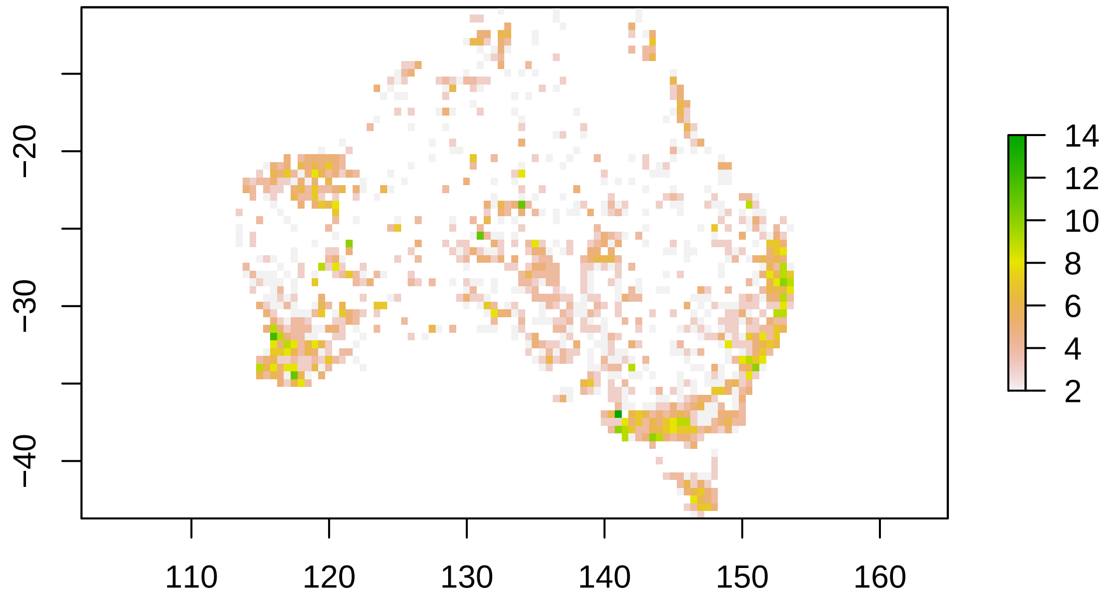


```
FDisras <- rr
values(FDisras) <- combo.Q$FDis
plot(FDisras)
```



Fill raster cells by richness and visualize it.

```
combo.R <- left_join(rr.cells,
                     gridded.dist[,1:3],
                     by=c("longitude", "latitude"))
RICHras <- rr
values(RICHras) <- combo.R$richness
plot(RICHras)
```



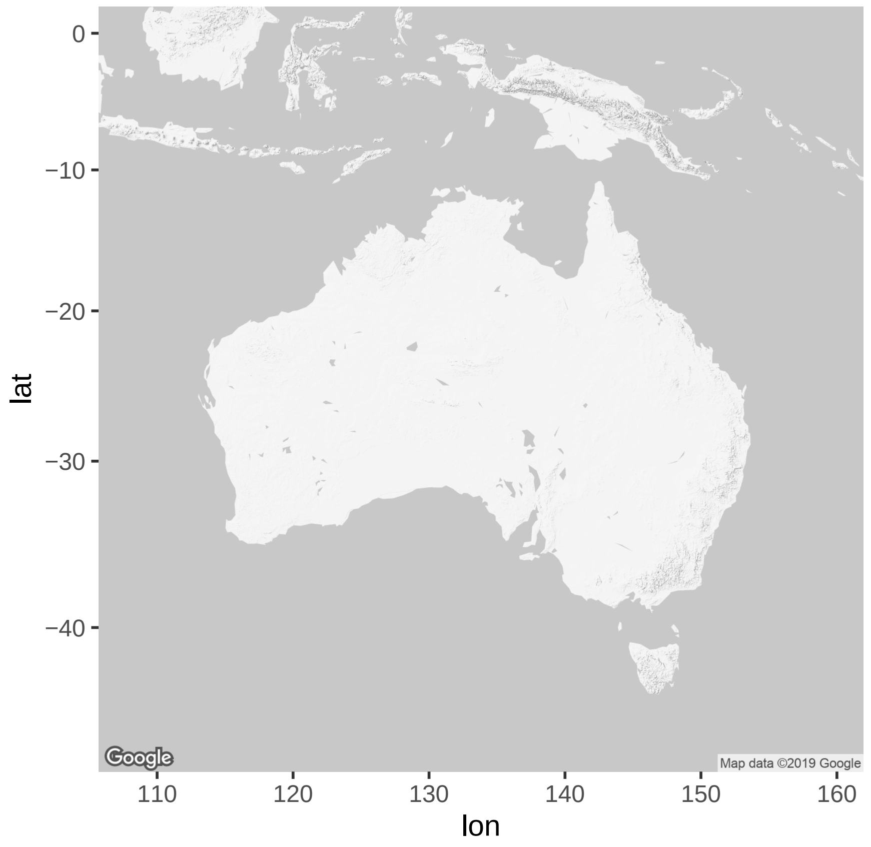
We can do this a bit prettier, start by establishing a map

```
graymap <- get_googlemap(center = "Australia", zoom = 4, style = 'https://maps.googleapis.com/maps/api/s
```

```
## Source : https://maps.googleapis.com/maps/api/staticmap?center=Australia&zoom=4&size=640x640&scale=2&

## Source : https://maps.googleapis.com/maps/api/geocode/json?address=Australia&key=xxx
```

```
ggmap(graymap)
```



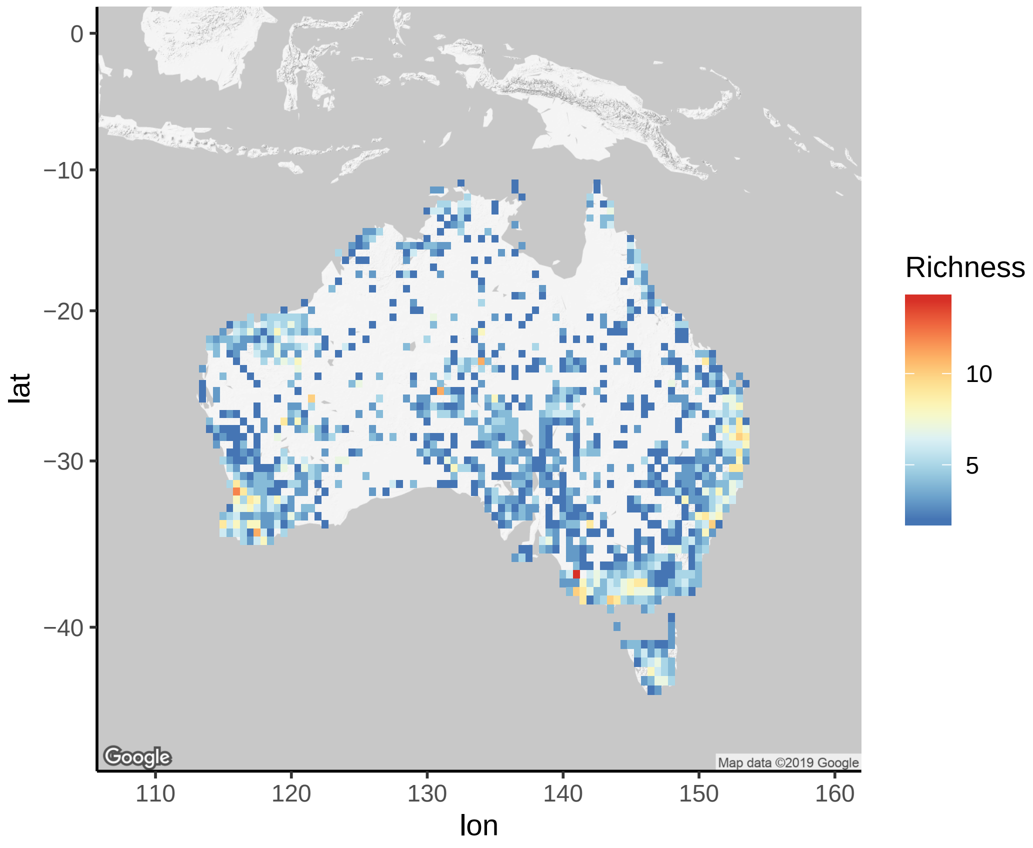
Create richness polygons

```r
RICHpoly <- rasterToPolygons(RICHras);
max.colors <- length(unique(RICHpoly$layer));
filled.RICH <- rep(RICHpoly$layer, each=5)
# 'each' is important, otherwise the polygon values get screwed up
```

Set the color palette length and the breakpoints

```r
RICHras@data@values[is.na(RICHras@data@values)] <- 0
pal.length <- abs(min(RICHras@data@values) - max(RICHras@data@values)) * 10
myBreaks <- c(seq(min(RICHras@data@values), 0, length.out=ceiling(pal.length/2) + 1),
              seq(max(RICHras@data@values)/pal.length, max(RICHras@data@values),
                  length.out=floor(pal.length/2)))
```

```r
ggmap(graymap) +
  geom_polygon(data = RICHpoly,
               aes(x = long, y = lat,
                   group = group,
                   fill = filled.RICH),
               size = 0, alpha = 1)  +
  scale_fill_gradientn("Richness",
                       colors = rev(colorRampPalette(
                         brewer.pal(9, "RdYlBu"))(max.colors))) +
  theme_classic()
```



Do the same for functional diversity (choose either FDras or FDisras)

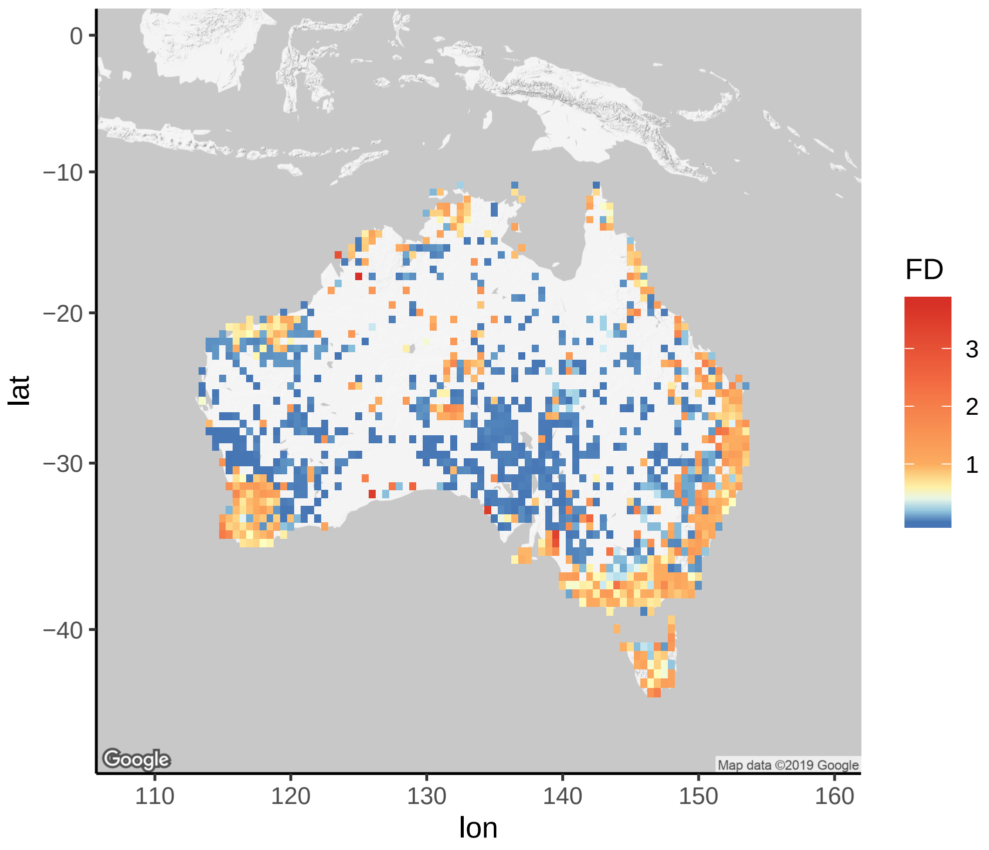
```r
FDpoly <- rasterToPolygons(FDras);
#FDpoly <- rasterToPolygons(FDisras)
max.colors <- length(unique(FDpoly$layer));
filled.FD <- rep(FDpoly$layer, each=5)
# 'each' is important, otherwise the polygon values get screwed up
```

Set the color palette length and the breakpoints

```r
FDras@data@values[is.na(FDras@data@values)] <- 0
pal.length <- abs(min(FDras@data@values) - max(FDras@data@values)) * 10
myBreaks <- c(seq(min(FDras@data@values), 0, length.out=ceiling(pal.length/2) + 1),
              seq(max(FDras@data@values)/pal.length, max(FDras@data@values),
                  length.out=floor(pal.length/2)))
#FDras@data@values[which(FDras@data@values == 0)] <- "NA"
```

```r
ggmap(graymap) +
  geom_polygon(data = FDpoly,
               aes(x = long,
                   y = lat,
                   group = group,
                   fill = filled.FD),
               size = 0, alpha = 1)  +
  scale_fill_gradientn("FD",
                       values=scales::rescale(c(min(res.table$RaoQ),
                                                mean(res.table$RaoQ)/2,
                                                mean(res.table$RaoQ),
                                                mean(res.table$RaoQ)*2,
                                                max(res.table$RaoQ))),
                       colors = rev(colorRampPalette(
                         brewer.pal(9, "RdYlBu"))(max.colors))) +

  theme_classic()
```



We've plotted richness and functional diversity, but we'd like to know if either is significantly different than scores from random communities.
We've already got a community matrix ('gridded.dist'), so just copy that.

```
cm <- gridded.dist
```

Create an empty raster or two

```
richness.raster <- rr; richness.raster@data@values[] <- 0
fd.raster <- rr; fd.raster@data@values[] <- 0
```

Add the FD and Richness scores to your community matrix

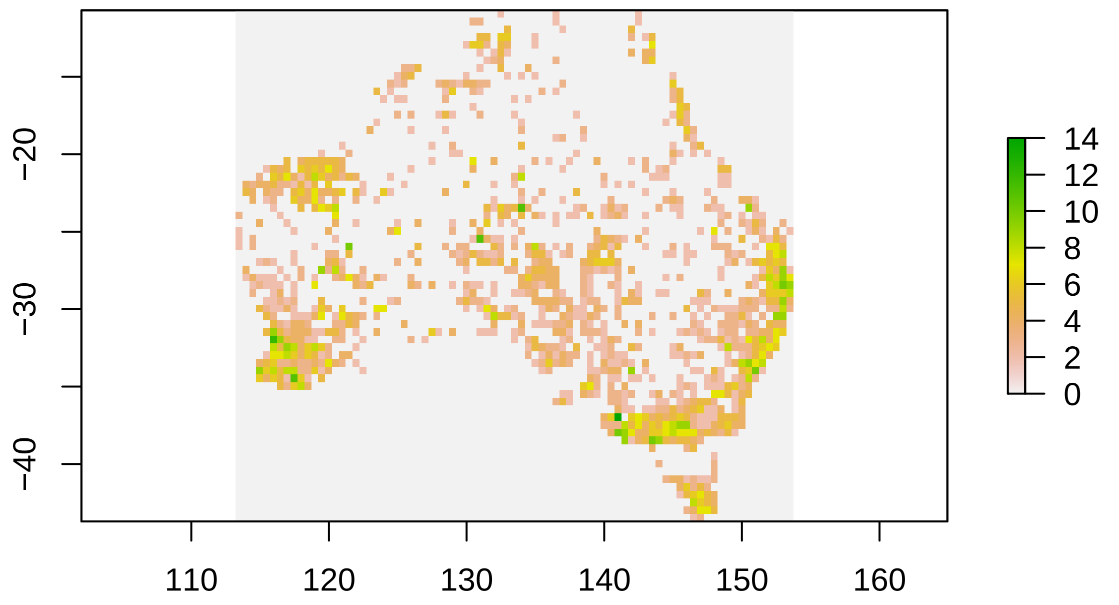
```
pre.rr <- left_join(rr.cells, cm, by=c("latitude", "longitude")); pre.rr[is.na(pre.rr)] <- 0
pre.fd <- left_join(rr.cells, res.table, by=c("latitude", "longitude"))
    pre.fd <- left_join(pre.fd, cm, by=c("latitude", "longitude")); pre.fd[is.na(pre.fd)] <- 0
```

Pass along the values from the matrices to your rasters

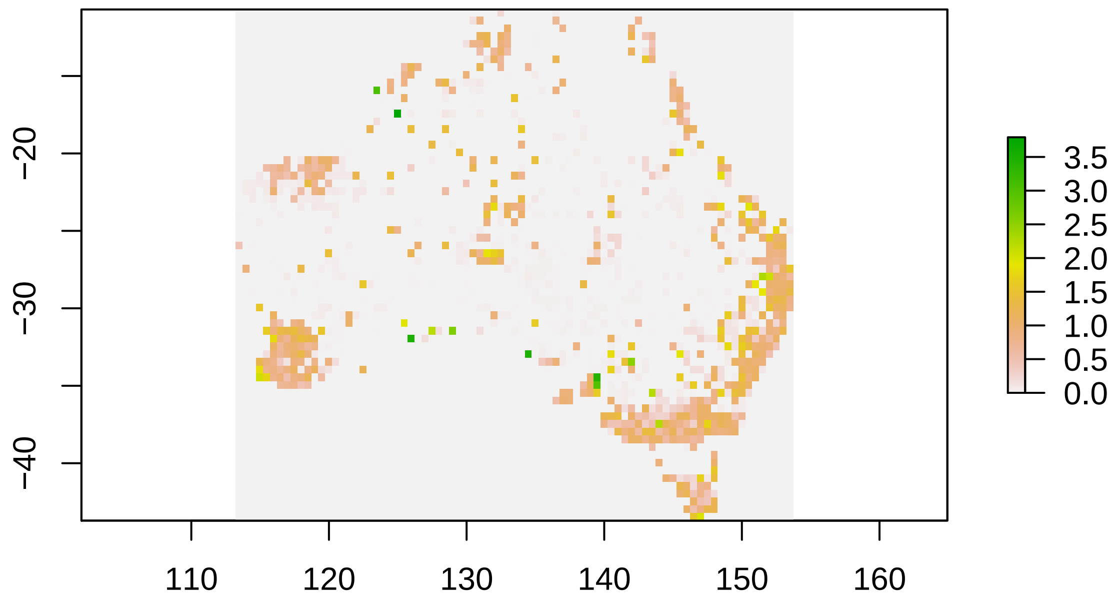
```
richness.raster@data@values <- pre.rr$richness
fd.raster@data@values <- pre.fd$RaoQ
#fd.raster@data@values <- pre.fd$FDis
```

Quickly plot them again to make sure they make sense and nothing funny happened

```
plot(richness.raster)
```



```r
plot(fd.raster)
```



Identify which cells have richness values > 1 (more than one taxon occupying it)
Or identify which cells have functional diversity values > 0 (so we can compare)

```r
cells.rich <- which(richness.raster@data@values > 1)
cells.fd <- which(fd.raster@data@values > 0)
```

Make your blank site x species matrices by choosing cells with richness (>1) and FD (>0)

```r
input.rr <- pre.rr[,4:ncol(pre.rr)];
input.rr <- input.rr[which(rowSums(input.rr) > 1),]
input.fd <- pre.fd[which(pre.fd$RaoQ > 0), 6:ncol(pre.fd)]
#input.fd <- pre.fd[which(pre.fd$FDis > 0), 6:ncol(pre.fd)]
```

We can check this quickly by showing how many sites there were (including those with no observations), and how many we now have (including only those with observations)

```
## [1] "5346 total sites"
```

```
## [1] "1159 sites have >1 species present"
```

```
## [1] "1159 sites have >0 functional diversity"
```

Get the x (longitude) y (latitude) coordinates of those cells

```
coords.rich <- xyFromCell(richness.raster, cells.rich)
coords.fd <- xyFromCell(fd.raster, cells.fd)
```

Now create the greater circle distance (in meters) for each raster. This is an important input step for our

```
# for richness
gc.dist.rich <- rdist.earth(coords.rich);
rownames(gc.dist.rich) <- cells.rich;
colnames(gc.dist.rich) <- cells.rich;
diag(gc.dist.rich) <- 0

# for functional diversity
gc.dist.fd <- rdist.earth(coords.fd);
rownames(gc.dist.fd) <- cells.fd;
colnames(gc.dist.fd) <- cells.fd;
diag(gc.dist.fd) <- 0
```

We'll need to source the dispersal null metric function

```
source("~/Documents/GitHub/MonitorPhylogenomics/DispersalNullModel.R")
```

And create an additional function to run this null model repeatedly

```
library(parallel)
nullFD <- function(n.model, n.iter,
                   method=c("randomizeMatrix", "DNM"),
                   cores, trait.data, measure=c("RaoQ", "FDis", "Richness"),
                   great.circle){

  beginning <- Sys.time()
  Rao.table <- NULL

  if(method=="randomizeMatrix"){
    swap <- mclapply(1:n.iter, function(x) {
      randomizeMatrix(input.fd,
                      null.model=n.model,
                      iterations=10)},
      mc.cores=cores)
    swap.res <- mclapply(1:length(swap), function(x) {
      dbFD(trait.frame, swap[[x]])}, mc.cores=8)

    for(j in 1:length(swap.res)){
      Rao.table <- cbind(Rao.table, swap.res[[j]]$RaoQ)
    }
  }
  else if(method=="DNM"){
    swap <- mclapply(1:n.iter, function(x) {
      DNM(input.fd, tree=NA,
          great.circle, abundance.matters=F,
          abundance.assigned="directly")}, mc.cores=cores)
    swap <- Filter(function(x) length(x)>1, swap)
    # Get FD
    if (measure=="RaoQ"){
      swap.res <- mclapply(1:length(swap), function(x) {
        dbFD(trait.data, swap[[x]])}, mc.cores=8)
      for(j in 1:length(swap.res)){
        Rao.table <- cbind(Rao.table, swap.res[[j]]$RaoQ)
      }
    }
    else if (measure=="FDis"){
      swap.res <- mclapply(1:length(swap), function(x) {
        dbFD(trait.data, swap[[x]])}, mc.cores=8)
      for(j in 1:length(swap.res)){
        Rao.table <- cbind(Rao.table, swap.res[[j]]$FDis)
      }
    }
    # or Get RICHNESS
    else if (measure=="Richness"){
      swap.res <- mclapply(1:length(swap), function(x) {
        rowSums(swap[[x]])}, mc.cores=8)
      for (j in 1:length(swap.res)){
        Rao.table <- cbind(Rao.table, swap.res[[j]])
      }
    }

    print(paste("you attempted", n.iter,
                "iterations, and you got",
                length(swap), "simulations"))

  }

  end <- Sys.time()
  duration <- format(end-beginning)
  print(paste("Computation time to fit", n.iter,
              method, "null models:", duration))

  Rao.table <- as.data.frame(Rao.table);
  Raw.table <- Rao.table
  Rao.table <- cbind(Rao.table,
                     sim.mean=rowMeans(Rao.table))
  Rao.table <- cbind(Rao.table,
                     sim.sd=apply(Raw.table, 1, sd))
  #Rao.table <- cbind(Rao.table, emp.val=) # I could add in the empirical values (FD)
  #Rao.table <- cbind(Rao.table, ses=apply(Rao.table, 1, (Rao.table[,"mean"]))) # then I could calculat
  return(Rao.table)
}
```

Run the function a lot. I’ll just quickly do 50 simulations here, but we should do many many more.

```
RQ <- nullFD(n.model=NULL,
             n.iter=50,
             method="DNM",
             cores=6,
             trait.data=log(marsupial.frame),
             measure="RaoQ",
             great.circle = gc.dist.fd)
```

If you don't have time to run those functions above, you'll want to read in the files

```
RQ <-     readRDS(file="~/Documents/GitHub/MonitorPhylogenomics/SimulatedMarsupial_RaoQ_logData.RDS")
SESras <- readRDS(file="~/Documents/GitHub/MonitorPhylogenomics/SimulatedMarsupial_RaoQ_logData_SES_ras
```

Now we need to add the empirical FD (or richness) values to this data frame

```
#RQ <- cbind(RQ, emp.val=res.table$RaoQ)
RQ <- cbind(RQ, emp.val=res.table$FDis)
```

Then get standard effect sizes (SES) for each sell across all simulations

```
ses.vec <- NULL
for(k in 1:nrow(RQ)){
  curr <- RQ[k,]
  ses <- (curr$emp.val - curr$sim.mean) / curr$sim.sd
  ses.vec <- append(ses.vec, ses)
}
# bind it to the simulation dataframe
RQ <- cbind(RQ, ses=ses.vec)
```

Make a table of the ses values with the coordinates of each cell

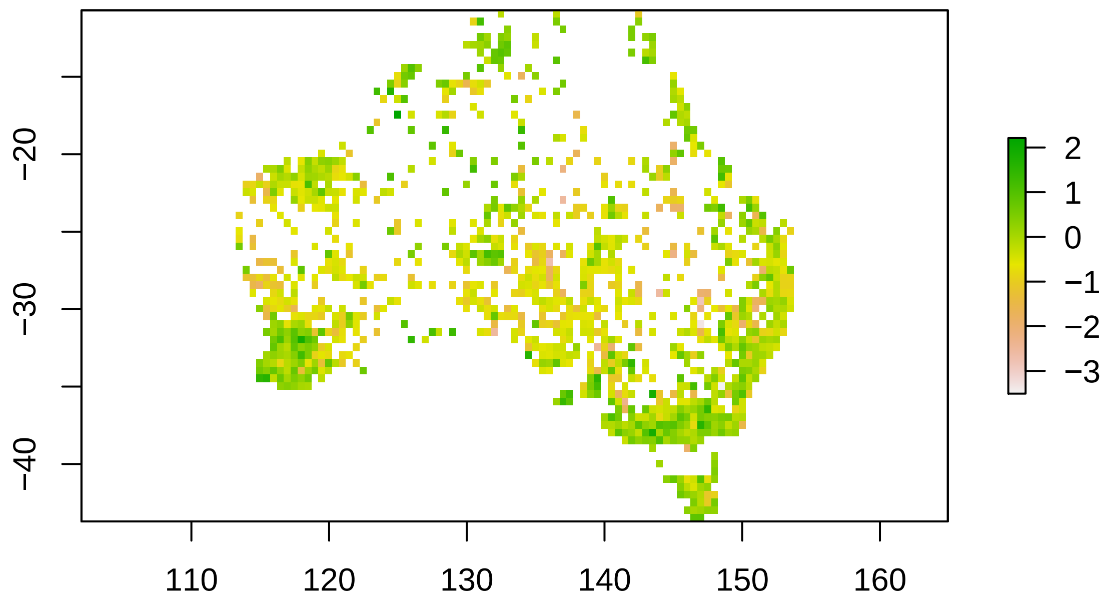
```
ses.table <- cbind.data.frame(latitude=gridded.dist$latitude, longitude=gridded.dist$longitude, SES=RQ$
```

Bind the table with the empty raster cells we set up earlier, and make any NA values 0.

```
combo.SES <- left_join(rr.cells,
                       ses.table,
                       by=c("latitude", "longitude"))
```

Make an empty raster frame for the ses values to go into
Dump them into the raster
And plot it to make sure it makes sense

```
SESras <- rr;
SESras@data@values[] <- 0
SESras@data@values <- combo.SES$SES
#values(SESras) <- combo.SES$SES
plot(SESras)
```

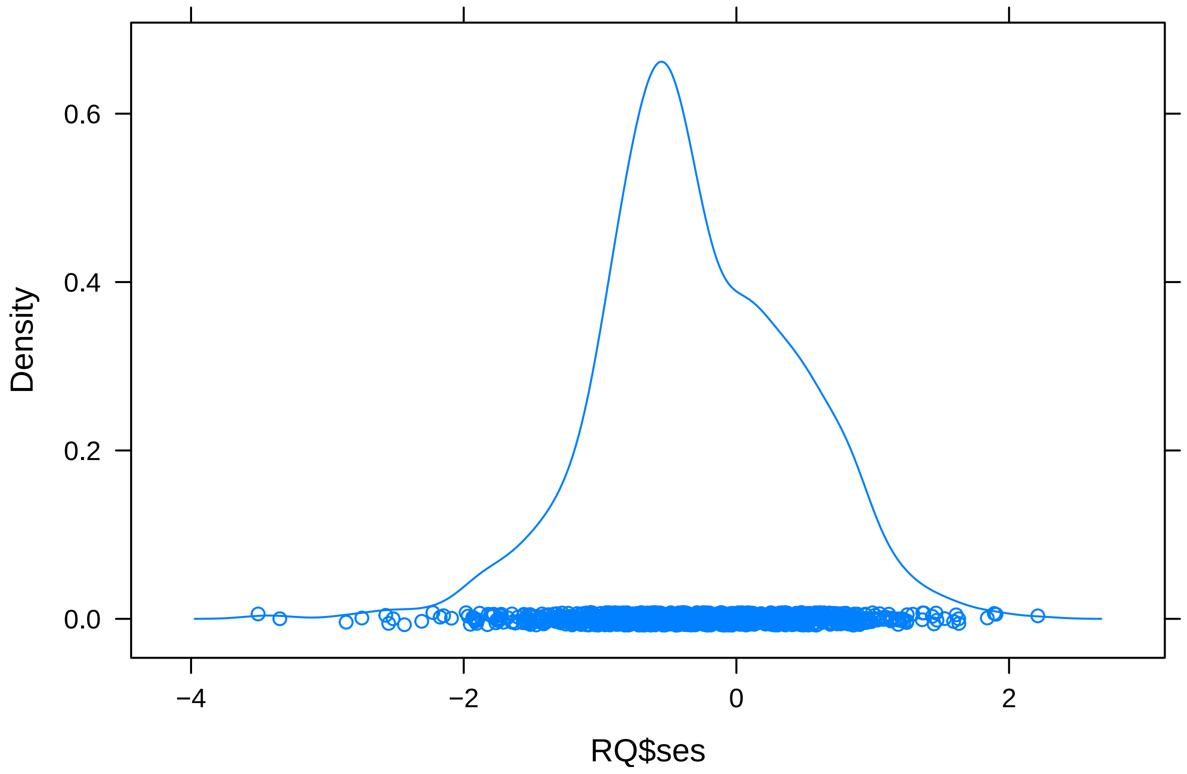


If you've been running all these steps, you'll want to save the output

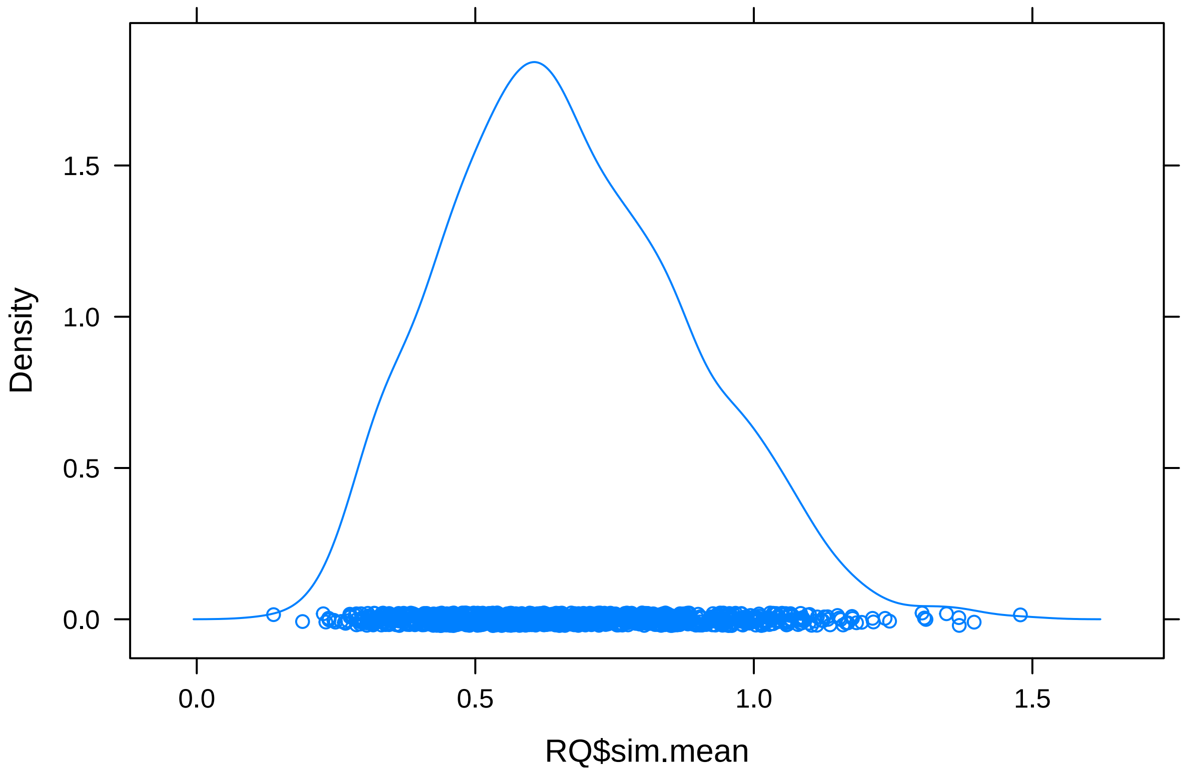
```
saveRDS(RQ, file="~/Documents/GitHub/MonitorPhylogenomics/SimulatedMarsupial_RaoQ_logData.RDS")
saveRDS(SESras, file="~/Documents/GitHub/MonitorPhylogenomics/SimulatedMarsupial_RaoQ_logData_SES_raste
```

Have a quick look at some of the parameters

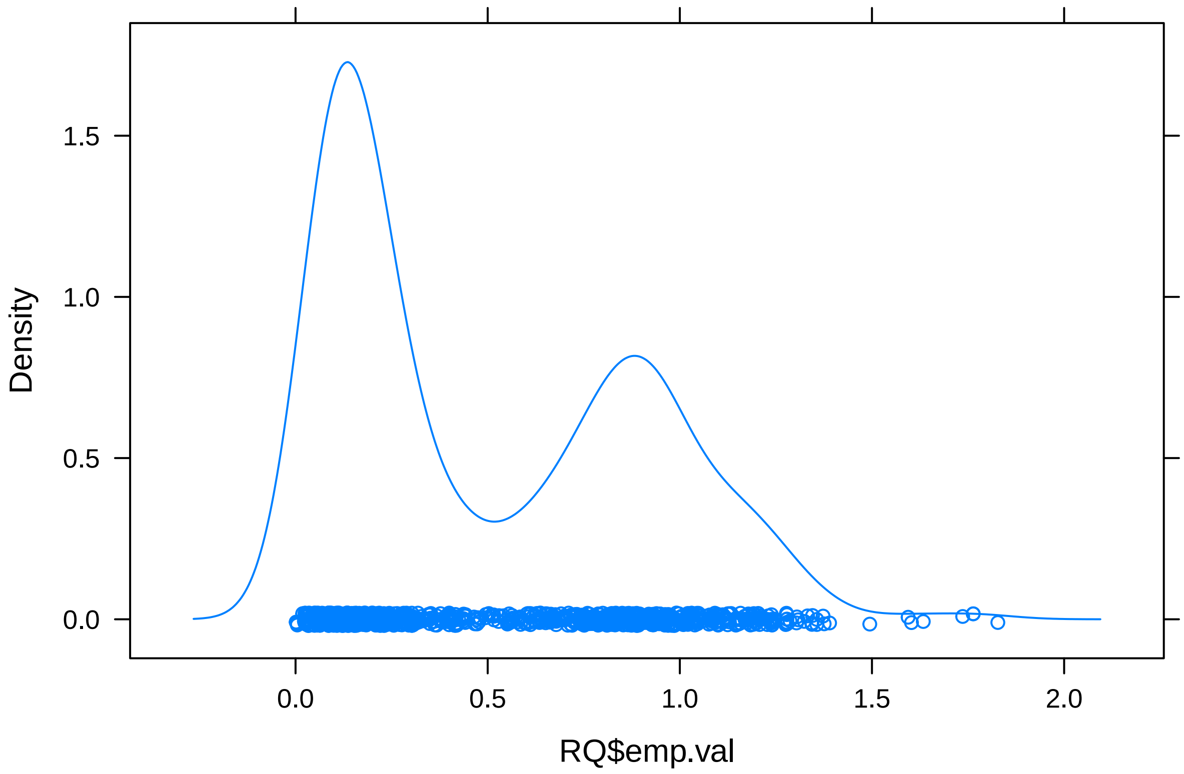
```
densityplot(RQ$ses)
```



```
densityplot(RQ$sim.mean)
```



```
densityplot(RQ$emp.val)
```



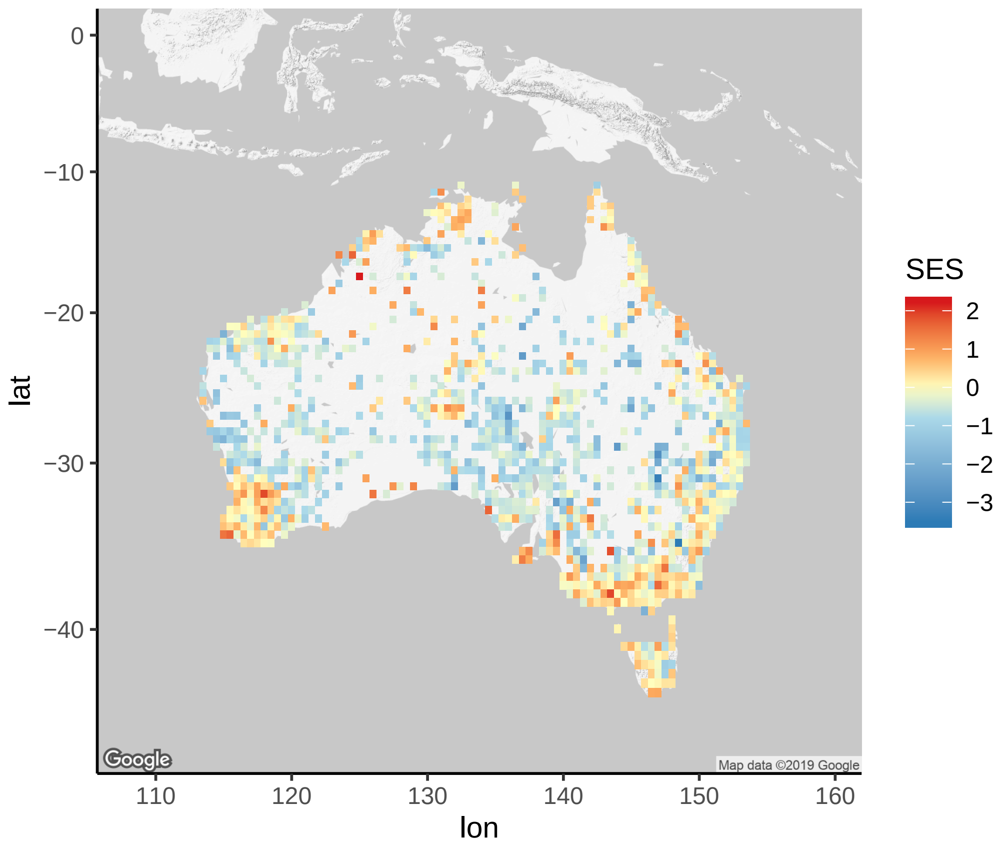
Translate the SES raster into polygons for plotting with ggmap

```
SESpoly <- rasterToPolygons(SESras);
max.colors <- length(unique(SESpoly$layer));
filled.SES <- rep(SESpoly$layer, each=5)
```

Lastly plot the map of SES (functional diversity)

```
ggmap(graymap) +
  geom_polygon(data = SESpoly,
               aes(x = long, y = lat, group = group,
                   fill = filled.SES), size = 0, alpha = 1)  +
  scale_fill_gradientn("SES", values=scales::rescale(c(min(ses.table$SES),
                                                       #min(ses.table$SES)/2,
                                                       -0.8,
                                                       0,
                                                       #max(ses.table$SES)/2,
                                                       0.8,
                                                       max(ses.table$SES))),
                     colors = rev(brewer.pal(5, "RdYlBu"))) +
  theme_classic()
```

```
## Regions defined for each Polygons
```



We want to know if the difference in simulated and observed FD values is signficant. So we'll create a function to calculate the confidence interval of the SES.

```
confidence_interval <- function(vector, interval) {
  # Standard deviation of sample
  vec_sd <- sd(vector)
  # Sample size
  n <- length(vector)
  # Mean of sample
  vec_mean <- mean(vector)
  # Error according to t distribution
  error <- qt((interval + 1)/2, df = n - 1) * vec_sd / sqrt(n)
  # Confidence interval as a vector
  result <- c("lower" = vec_mean - error, "upper" = vec_mean + error,
              "error" = error, "mean" = vec_mean, "sd" = vec_sd, "N" = n)
  return(result)
}
```

Can also be calculated as:

upper = mean + (error * 1.96)

lower = mean - (error * 1.96)

```
CIall <- confidence_interval(ses.table$SES, 0.95); CIall
```

```
##         lower         upper         error          mean            sd
##   -0.34121480   -0.25649242    0.04236119   -0.29885361    0.73503433
##             N
## 1159.00000000
```

```
CIall["richness"] <- 1
```

```
siteRICH <- left_join(ses.table,
                      gridded.dist[,1:3],
                      by=c("longitude", "latitude"))
```

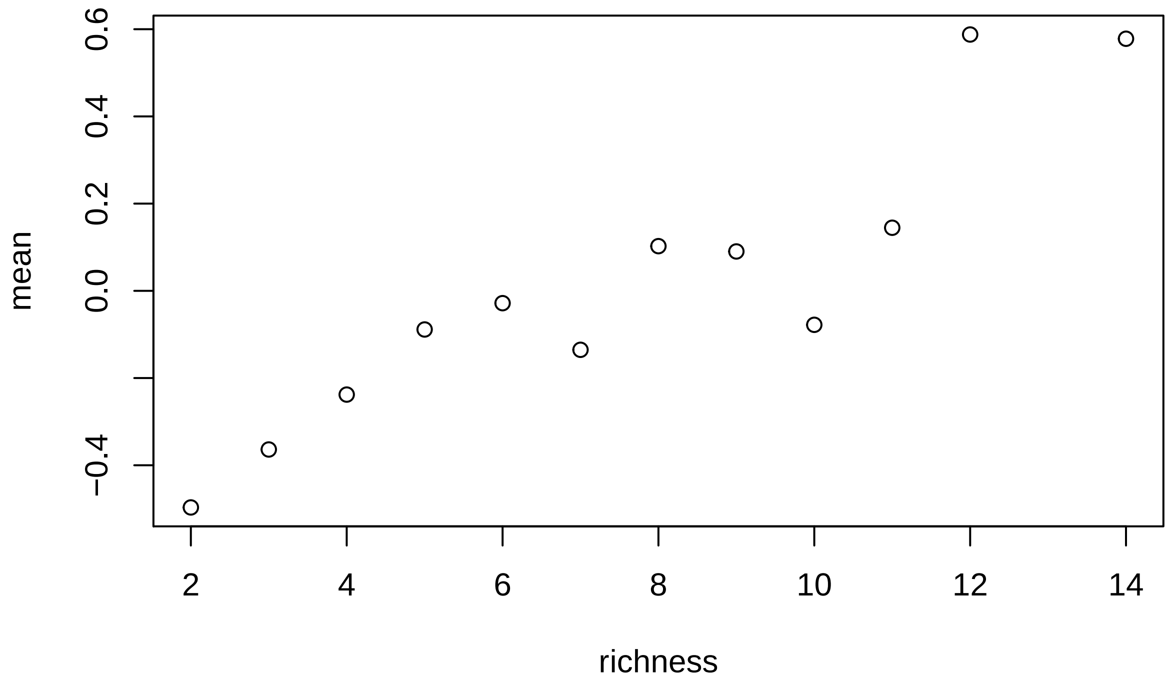
```
CIses <- NULL
for (i in min(siteRICH$richness):max(siteRICH$richness)){
  curr.rich <- filter(siteRICH, richness == i)
  CIses <- rbind(CIses, confidence_interval(curr.rich$SES, 0.95))
}
```

```
## Warning in qt((interval + 1)/2, df = n - 1): NaNs produced

## Warning in qt((interval + 1)/2, df = n - 1): NaNs produced

## Warning in qt((interval + 1)/2, df = n - 1): NaNs produced
```

```
CIses <- data.frame(CIses)
CIses$richness <- 2:14
plot(data=CIses, mean ~ richness)
```

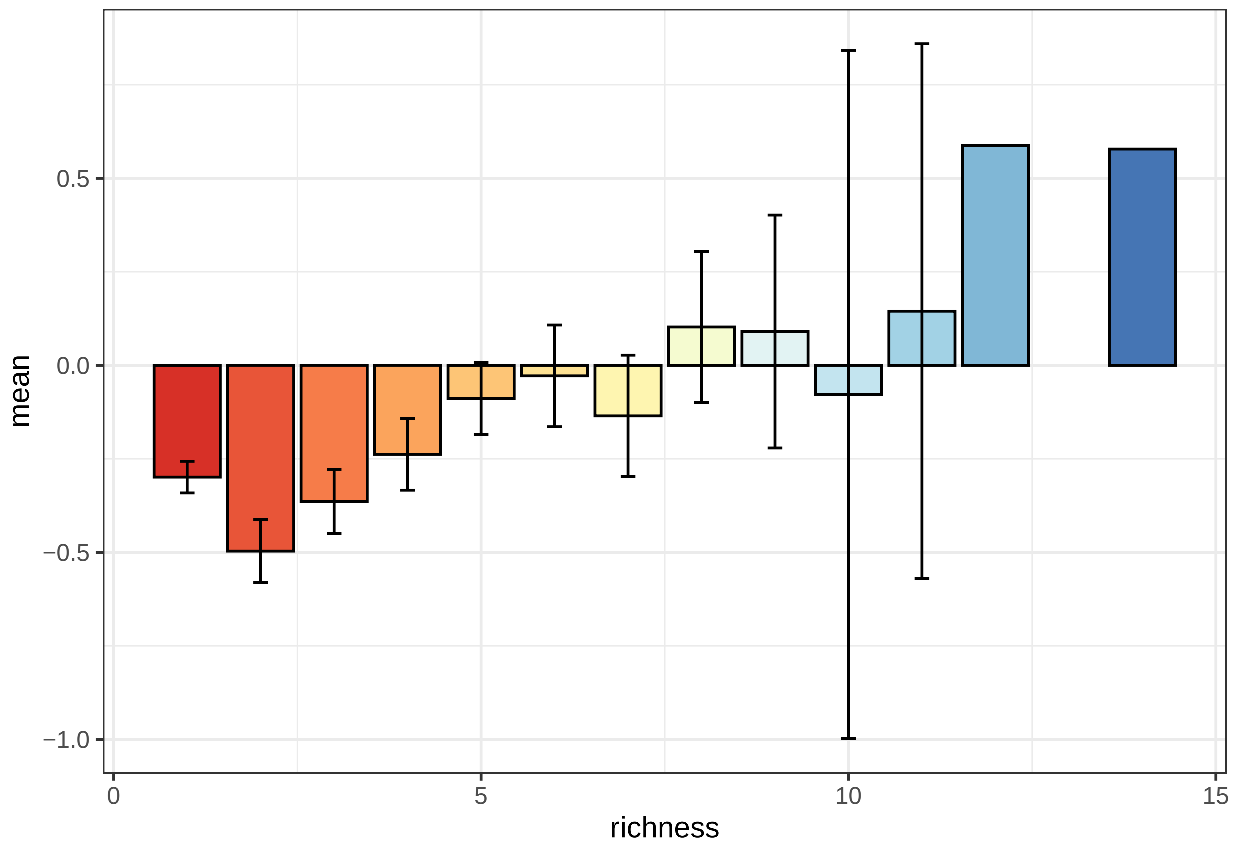


Add the confidence interval for SES across the whole continent to the individual communities

```
CIses <- rbind(CIses, CIall); CIses
```

```
##          lower       upper      error        mean        sd    N richness
## 1  -0.58074685 -0.41272018 0.08401333 -0.49673352 0.8240480  372        2
## 2  -0.44960715 -0.27807951 0.08576382 -0.36384333 0.7211103  274        3
## 3  -0.33382602 -0.14205546 0.09588528 -0.23794074 0.6806567  196        4
## 4  -0.18510308  0.00781287 0.09645798 -0.08864511 0.5958232  149        5
## 5  -0.16428030  0.10775026 0.13601528 -0.02826502 0.5576244   67        6
## 6  -0.29765388  0.02714509 0.16239948 -0.13525440 0.5341600   44        7
## 7  -0.09925506  0.30421346 0.20173426  0.10247920 0.5202561   28        8
## 8  -0.22099487  0.40166454 0.31132970  0.09033484 0.6459329   19        9
## 9  -0.99797118  0.84207908 0.92002513 -0.07794605 0.7409614    5       10
## 10 -0.57021325  0.85959016 0.71490171  0.14468845 0.2877867    3       11
## 11          NA          NA         NA  0.58781462        NA    1       12
## 12         NaN          NA         NA         NaN        NA    0       13
## 13          NA          NA         NA  0.57813845        NA    1       14
## 14 -0.34121480 -0.25649242 0.04236119 -0.29885361 0.7350343 1159        1
```

```
library(RColorBrewer)
ggplot(CIses, aes(x=richness, y=mean)) +
  geom_bar(stat="identity", color="black",
           position=position_dodge(),
           fill = colorRampPalette(brewer.pal(9, "RdYlBu"))(14)) +
  geom_errorbar(aes(ymin=lower, ymax=upper), width=.2,
                position=position_dodge(.9)) +
  theme_bw()
```



mean
0.5
0.0
−0.5
−1.0
0
5
10
15
richness

Spatial Coevolution of *Varanus* and Marsupials

Great, now we want to do the same spatial analyses for both the monitor lizards and the cohabiting marsupials.

```r
cospatial.tutorial <- readRDS("~/Documents/GitHub/MonitorPhylogenomics/CoSpatial_Walkthrough.RDS")
names(cospatial.tutorial)
```

```
## [1] "goanna.distribution"    "goanna.sizes"
## [3] "marsupial.distribution" "marsupial.sizes"
```

Combine the two distribution data frames

```r
co.distribution <- rbind(cospatial.tutorial$goanna.distribution,
                         cospatial.tutorial$marsupial.distribution)
```

Create a tibble from the distribution data, turn it into Site x Species tibble

```r
cogridded <- co.distribution %>%

  ## bin into 0.5-degree bins
  dplyr::mutate(longitude=round(Longitude*2)/2, latitude=round(Latitude*2)/2) %>%

  #  ## average environmental vars within each bin
  group_by(longitude,latitude) %>%
  #  mutate(precipitationAnnual=mean(precipitationAnnual, na.rm=TRUE),
  #          temperatureAnnualMaxMean=mean(temperatureAnnualMaxMean, na.rm=TRUE)) %>%

  ## subset to vars of interest
  dplyr::select(longitude, latitude, Name_in_Tree) %>%

  ## take one row per cell per species (presence)
  distinct() %>%

  ## calculate species richness
  dplyr::mutate(richness=n()) %>%

  ## convert to wide format (sites by species)
  dplyr::mutate(present=1) %>%
  do(tidyr::spread(data=., key=Name_in_Tree, value=present, fill=0)) %>%
  ungroup()
```

Have a quick look at the Site x Species tibble, then translate it to a data frame we can manipulate normally.

```r
gridded.dist <- as.data.frame(cogridded)
gridded.dist[1:5, 1:7]
```

```
##   longitude latitude richness Sminthopsis.dolichura Varanus_gouldii
## 1     113.0    -25.5        1                     1              NA
## 2     113.0    -25.0        2                    NA               1
## 3     113.5    -26.5        1                     1              NA
## 4     113.5    -26.0        8                     1               1
## 5     113.5    -25.5        4                     1               1
##   Perameles.bougainville Varanus_brevicauda
## 1                     NA                 NA
## 2                      1                 NA
## 3                     NA                 NA
## 4                      1                  1
## 5                     NA                 NA
```

Lots of sites don't have any records, and are listed as NAs. This won't jibe with our code, so switch NA to 0.

```
gridded.dist[is.na(gridded.dist)] <- 0 # make NAs 0
gridded.dist <- filter(gridded.dist, !richness==1) # remove sites with just one taxon
gridded.dist <- filter(gridded.dist, latitude <= -11);
gridded.dist <- filter(gridded.dist, longitude >= 113.5)
gdist <- gridded.dist[ , 4:ncol(gridded.dist)]
```

Combine the marsupial and goanna trait data into a single data frame

```
co.trait <- rbind(cospatial.tutorial$goanna.sizes,
                  cospatial.tutorial$marsupial.sizes)
co.trait[1:5,]
```

```
##   Body_Length       Name_in_Tree  Location
## 1       236.0 Varanus_acanthurus Australia
## 2       260.0  Varanus_balagardi Australia
## 3       171.0    Varanus_baritji Australia
## 4       120.0 Varanus_brevicauda Australia
## 5       114.5      Varanus_bushi Australia
```

```
co.trait[33:37,]
```

```
##    Body_Length         Name_in_Tree  Location
## 33        90.0 Antechinomys.laniger Australia
## 34        94.5    Antechinus.agilis Australia
## 35       134.5    Antechinus.bellus Australia
## 36       129.0  Antechinus.flavipes Australia
## 37       133.0   Antechinus.godmani Australia
```

Make the order of the trait dataframe match the order of the Site x Species DF

```
both.trait <- co.trait[match(colnames(gdist),
                                   co.trait$Name_in_Tree),]
both.frame <- data.frame(SVL = both.trait$Body_Length);
rownames(both.frame) <- both.trait$Name_in_Tree
```

Run the Functional Diversity function and extract two estimates of fuctional diversity: the Rao's Quadratic value, and FDis

```
best <- dbFD(log(both.frame), gdist)
```

```
## FEVe: Could not be calculated for communities with <3 functionally singular species.
## FRic: Only one continuous trait or dimension in 'x'. FRic was measured as the range, NOT as the conve
## FDiv: Cannot not be computed when 'x' contains one single continuous trait or dimension.
```

```
RQ.scores <- best$RaoQ
FDis.scores <- best$FDis
res.table <- cbind.data.frame(latitude=gridded.dist$latitude,
                              longitude=gridded.dist$longitude,
                              RaoQ=best$RaoQ, FDis=best$FDis)
```

Read in your shapefile

```
oz <- shapefile("~/Documents/GitHub/MonitorPhylogenomics/Map_Shapefiles/Australia.shp")
plot(oz)
```

Set up a raster “template” for a 0.5 degree grid

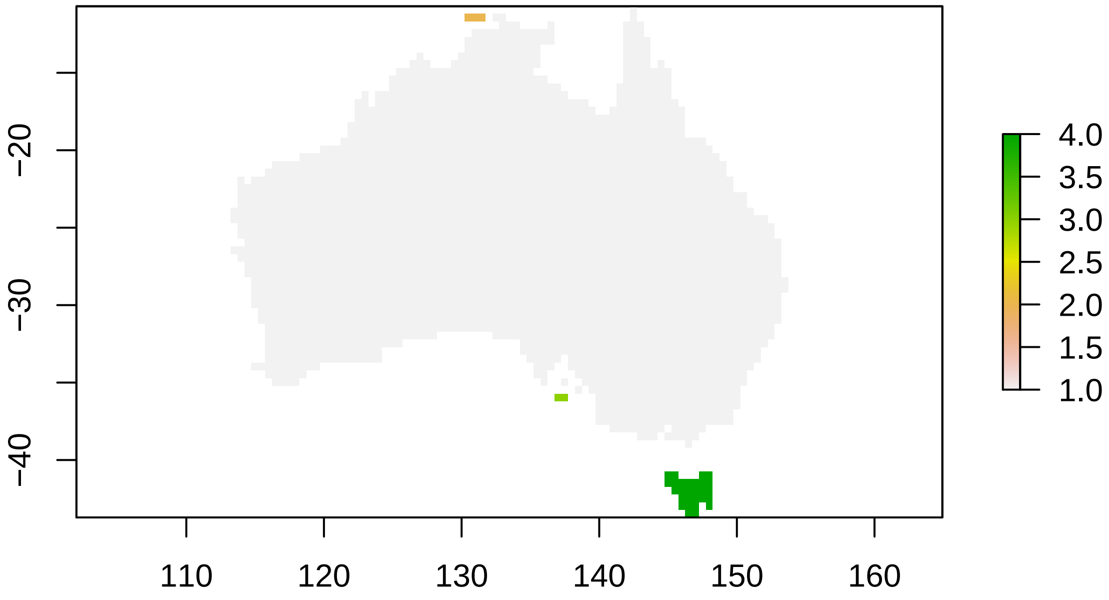
```
ext <- extent(113.2244, 153.6242, -43.64806, -10.70667)
gridsize <- 0.5
r <- raster(ext, res=gridsize)
```

Rasterize the shapefile

```
rr <- rasterize(oz, r)
```

Plot raster

```
plot(rr)
```

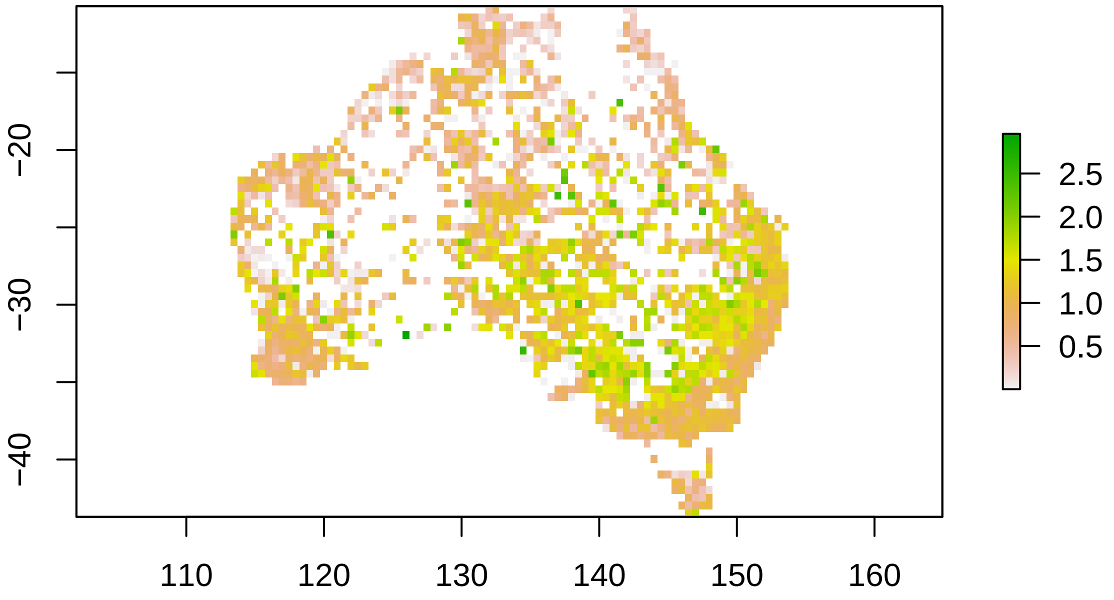


```
rr.cells <- xyFromCell(rr, 1:length(rr));
rr.cells <- as.data.frame(rr.cells)
rr.cells$x <- round(rr.cells$x*2)/2;
rr.cells$y <- round(rr.cells$y*2)/2
colnames(rr.cells) <- c("longitude", "latitude")
head(rr.cells)
```

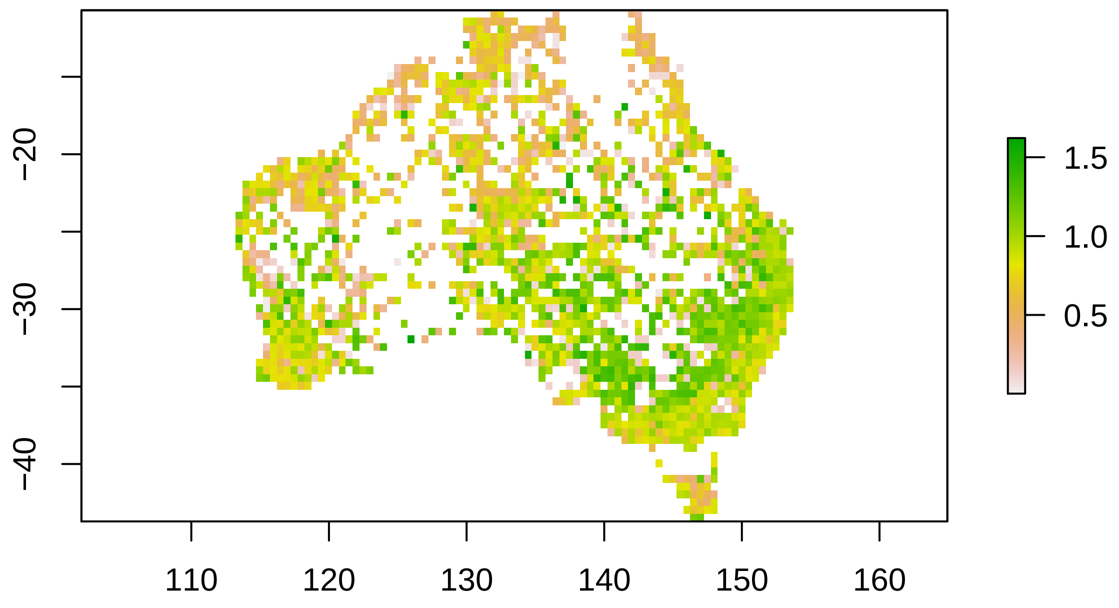
```
##   longitude latitude
## 1     113.5      -11
## 2     114.0      -11
## 3     114.5      -11
## 4     115.0      -11
## 5     115.5      -11
## 6     116.0      -11
```

Fill raster cells by FD values (Rao's Q, FDis), and visualize it.

```
combo.Q <- left_join(rr.cells,
                     res.table,
                     by=c("longitude", "latitude"))
FDras <- rr
values(FDras) <- combo.Q$RaoQ
plot(FDras)
```

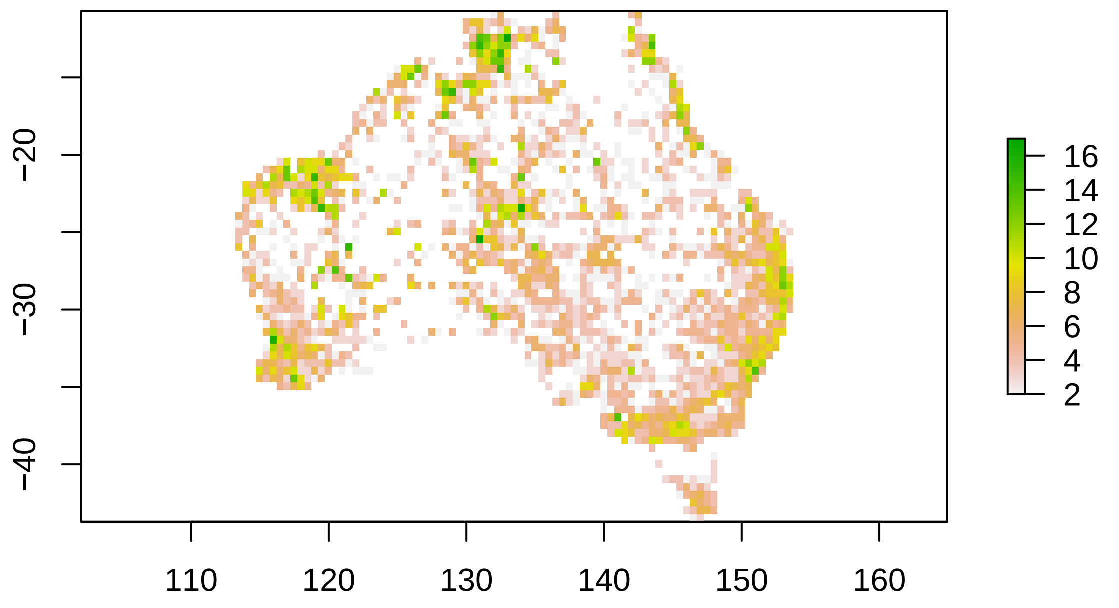


```
FDisras <- rr
values(FDisras) <- combo.Q$FDis
plot(FDisras)
```



Fill raster cells by richness and visualize it.

```
combo.R <- left_join(rr.cells,
                     gridded.dist[,1:3],
                     by=c("longitude", "latitude"))
RICHras <- rr
values(RICHras) <- combo.R$richness
plot(RICHras)
```



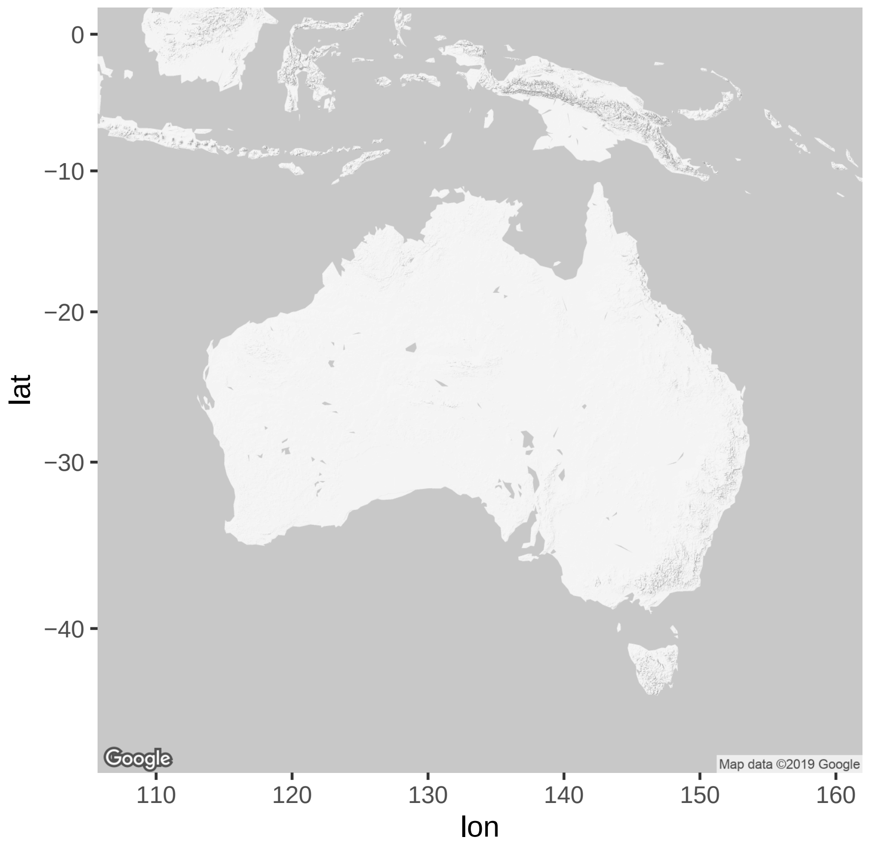
We can do this a bit prettier, start by establishing a map

```
graymap <- get_googlemap(center = "Australia", zoom = 4, style = 'https://maps.googleapis.com/maps/api/s
```

```
## Source : https://maps.googleapis.com/maps/api/staticmap?center=Australia&zoom=4&size=640x640&scale=2&

## Source : https://maps.googleapis.com/maps/api/geocode/json?address=Australia&key=xxx
```

```
ggmap(graymap)
```



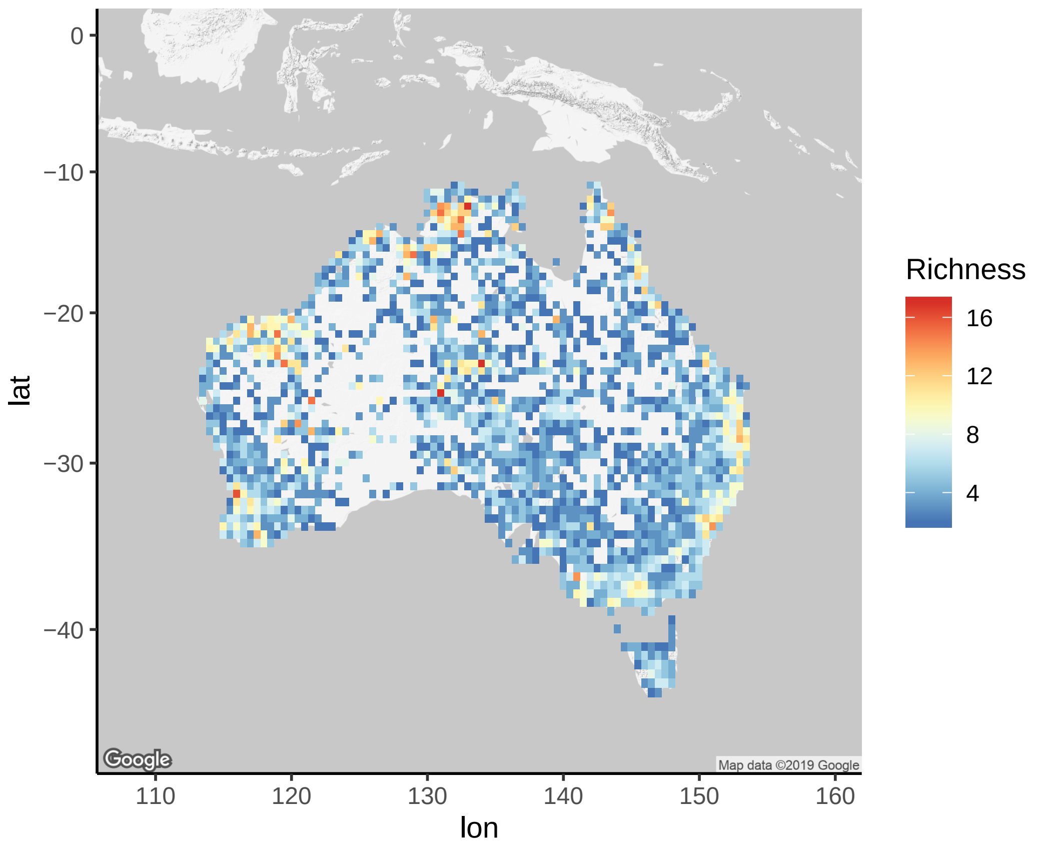
Create richness polygons

```
RICHpoly <- rasterToPolygons(RICHras);
max.colors <- length(unique(RICHpoly$layer));
filled.RICH <- rep(RICHpoly$layer, each=5)
# 'each' is important, otherwise the polygon values get screwed up
```

Set the color palette length and the breakpoints

```
RICHras@data@values[is.na(RICHras@data@values)] <- 0
pal.length <- abs(min(RICHras@data@values) - max(RICHras@data@values)) * 10
myBreaks <- c(seq(min(RICHras@data@values), 0, length.out=ceiling(pal.length/2) + 1),
              seq(max(RICHras@data@values)/pal.length, max(RICHras@data@values),
                  length.out=floor(pal.length/2)))
```

```
ggmap(graymap) +
  geom_polygon(data = RICHpoly,
               aes(x = long, y = lat,
                   group = group,
                   fill = filled.RICH),
               size = 0, alpha = 1)  +
  scale_fill_gradientn("Richness",
                       colors = rev(colorRampPalette(
                         brewer.pal(9, "RdYlBu"))(max.colors))) +
  theme_classic()
```



Do the same for functional diversity (choose either FDras or FDisras)

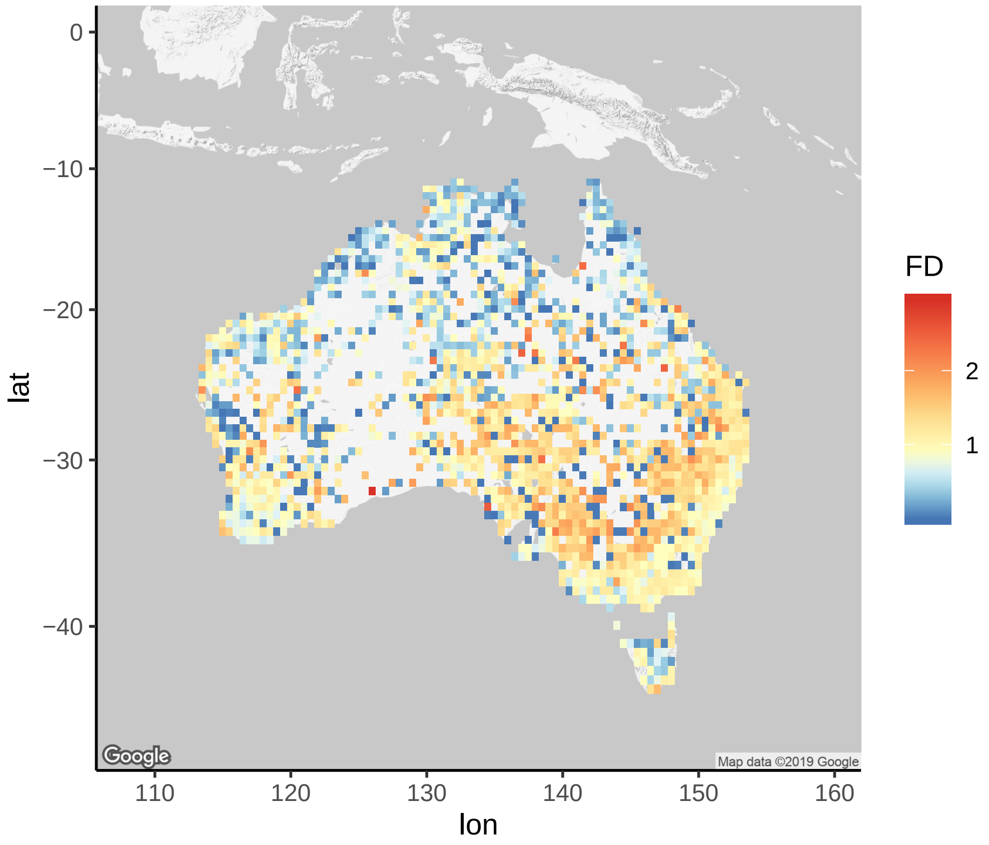
```
FDpoly <- rasterToPolygons(FDras);
#FDpoly <- rasterToPolygons(FDisras)
max.colors <- length(unique(FDpoly$layer));
filled.FD <- rep(FDpoly$layer, each=5)
# 'each' is important, otherwise the polygon values get screwed up
```

Set the color palette length and the breakpoints

```
FDras@data@values[is.na(FDras@data@values)] <- 0
pal.length <- abs(min(FDras@data@values) - max(FDras@data@values)) * 10
myBreaks <- c(seq(min(FDras@data@values), 0, length.out=ceiling(pal.length/2) + 1),
              seq(max(FDras@data@values)/pal.length, max(FDras@data@values),
                  length.out=floor(pal.length/2)))
#FDras@data@values[which(FDras@data@values == 0)] <- "NA"
```

```
ggmap(graymap) +
  geom_polygon(data = FDpoly,
               aes(x = long,
                   y = lat,
                   group = group,
                   fill = filled.FD),
               size = 0, alpha = 1)  +
  scale_fill_gradientn("FD",
                       values=scales::rescale(c(min(res.table$RaoQ),
                                                mean(res.table$RaoQ)/2,
                                                mean(res.table$RaoQ),
                                                mean(res.table$RaoQ)*2,
                                            max(res.table$RaoQ))),
                        colors = rev(colorRampPalette(
                          brewer.pal(9, "RdYlBu"))(max.colors))) +

  theme_classic()
```



We've plotted richness and functional diversity, but we'd like to know if either is significantly different than scores from random communities.
We've already got a community matrix ('gridded.dist'), so just copy that.

```
cm <- gridded.dist
```

Create an empty raster or two

```
richness.raster <- rr; richness.raster@data@values[] <- 0
fd.raster <- rr; fd.raster@data@values[] <- 0
```

Add the FD and Richness scores to your community matrix

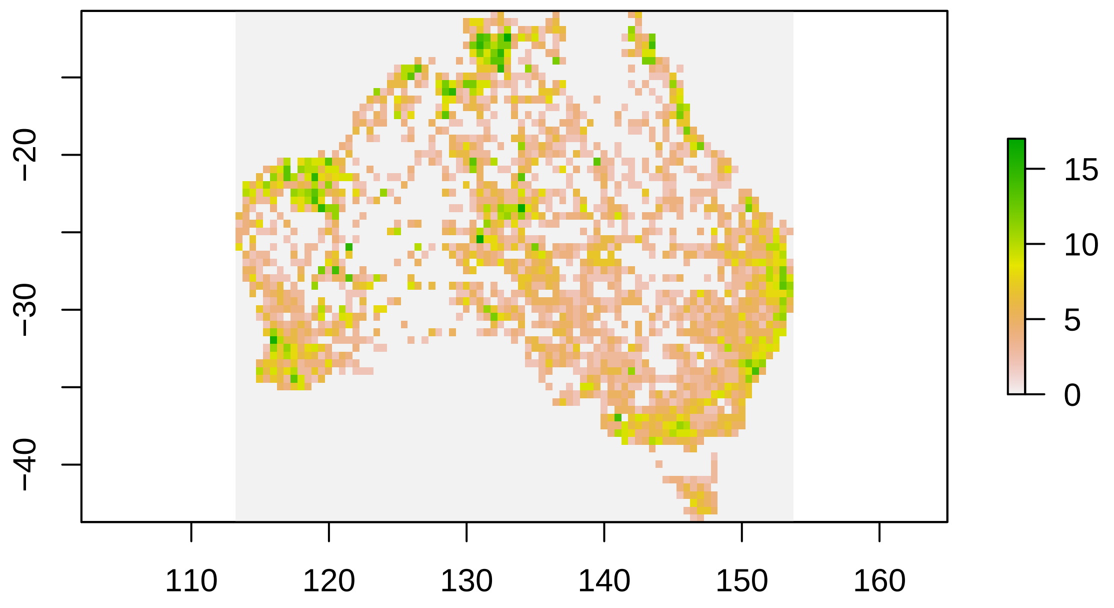
```
pre.rr <- left_join(rr.cells, cm, by=c("latitude", "longitude")); pre.rr[is.na(pre.rr)] <- 0
pre.fd <- left_join(rr.cells, res.table, by=c("latitude", "longitude"))
    pre.fd <- left_join(pre.fd, cm, by=c("latitude", "longitude")); pre.fd[is.na(pre.fd)] <- 0
```

Pass along the values from the matrices to your rasters

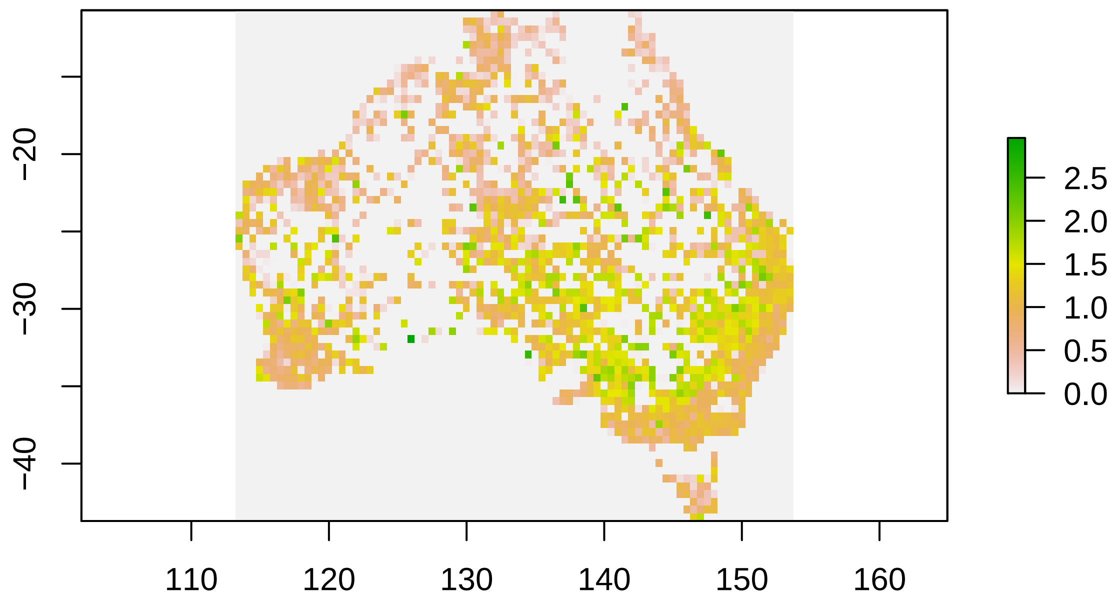
```
richness.raster@data@values <- pre.rr$richness
fd.raster@data@values <- pre.fd$RaoQ
#fd.raster@data@values <- pre.fd$FDis
```

Quickly plot them again to make sure they make sense and nothing funny happened

```
plot(richness.raster)
```



```
plot(fd.raster)
```



Identify which cells have richness values > 1 (more than one taxon occupying it)
Or identify which cells have functional diversity values > 0 (so we can compare)

```
cells.rich <- which(richness.raster@data@values > 1)
cells.fd <- which(fd.raster@data@values > 0)
```

Make your blank site x species matrices by choosing cells with richness (>1) and FD (>0)

```
input.rr <- pre.rr[,4:ncol(pre.rr)];
input.rr <- input.rr[which(rowSums(input.rr) > 1),]
input.fd <- pre.fd[which(pre.fd$RaoQ >= 0), 6:ncol(pre.fd)]
#input.fd <- pre.fd[which(pre.fd$FDis > 0), 6:ncol(pre.fd)]
```

We can check this quickly by showing how many sites there were (including those with no observations), and how many we now have (including only those with observations)

```
## [1] "5346 total sites"

## [1] "1863 sites have >1 species present"

## [1] "5346 sites have >0 functional diversity"
```

Get the x (longitude) y (latitude) coordinates of those cells

```
coords.rich <- xyFromCell(richness.raster, cells.rich)
coords.fd <- xyFromCell(fd.raster, cells.fd)
```

Now create the greater circle distance (in meters) for each raster. This is an important input step for our

```
# for richness
gc.dist.rich <- rdist.earth(coords.rich);
rownames(gc.dist.rich) <- cells.rich;
colnames(gc.dist.rich) <- cells.rich;
diag(gc.dist.rich) <- 0

# for functional diversity
gc.dist.fd <- rdist.earth(coords.fd);
rownames(gc.dist.fd) <- cells.fd;
colnames(gc.dist.fd) <- cells.fd;
diag(gc.dist.fd) <- 0
```

We'll need to source the dispersal null metric function

```r
source("~/Documents/GitHub/MonitorPhylogenomics/DispersalNullModel.R")
```

And create an additional function to run this null model repeatedly

```r
library(parallel)
nullFD <- function(n.model, n.iter,
                   method=c("randomizeMatrix", "DNM"),
                   cores, trait.data, measure=c("RaoQ", "FDis", "Richness"),
                   great.circle){

  beginning <- Sys.time()
  Rao.table <- NULL

  if(method=="randomizeMatrix"){
    swap <- mclapply(1:n.iter, function(x) {
      randomizeMatrix(input.fd,
                      null.model=n.model,
                      iterations=10)},
      mc.cores=cores)
    swap.res <- mclapply(1:length(swap), function(x) {
      dbFD(trait.frame, swap[[x]])}, mc.cores=8)

    for(j in 1:length(swap.res)){
      Rao.table <- cbind(Rao.table, swap.res[[j]]$RaoQ)
    }
  }
  else if(method=="DNM"){
    swap <- mclapply(1:n.iter, function(x) {
      DNM(input.fd, tree=NA,
          great.circle, abundance.matters=F,
          abundance.assigned="directly")}, mc.cores=cores)
    swap <- Filter(function(x) length(x)>1, swap)
    # Get FD
    if (measure=="RaoQ"){
      swap.res <- mclapply(1:length(swap), function(x) {
        dbFD(trait.data, swap[[x]])}, mc.cores=8)
      for(j in 1:length(swap.res)){
        Rao.table <- cbind(Rao.table, swap.res[[j]]$RaoQ)
      }
    }
    else if (measure=="FDis"){
      swap.res <- mclapply(1:length(swap), function(x) {
        dbFD(trait.data, swap[[x]])}, mc.cores=8)
      for(j in 1:length(swap.res)){
        Rao.table <- cbind(Rao.table, swap.res[[j]]$FDis)
      }
    }
    # or Get RICHNESS
    else if (measure=="Richness"){
      swap.res <- mclapply(1:length(swap), function(x) {
        rowSums(swap[[x]])}, mc.cores=8)
      for (j in 1:length(swap.res)){
        Rao.table <- cbind(Rao.table, swap.res[[j]])
      }
    }

    print(paste("you attempted", n.iter,
                "iterations, and you got",
                length(swap), "simulations"))

  }

  end <- Sys.time()
  duration <- format(end-beginning)
  print(paste("Computation time to fit", n.iter,
              method, "null models:", duration))

  Rao.table <- as.data.frame(Rao.table);
  Raw.table <- Rao.table
  Rao.table <- cbind(Rao.table,
                     sim.mean=rowMeans(Rao.table))
  Rao.table <- cbind(Rao.table,
                     sim.sd=apply(Raw.table, 1, sd))
  #Rao.table <- cbind(Rao.table, emp.val=) # I could add in the empirical values (FD)
  #Rao.table <- cbind(Rao.table, ses=apply(Rao.table, 1, (Rao.table[,"mean"]))) # then I could calculat
  return(Rao.table)
}
```

Run the function a lot. I'll just quickly do 50 simulations here, but we should do many many more.

```
RQ <- nullFD(n.model=NULL,
             n.iter=50,
             method="DNM",
             cores=6,
             trait.data=log(both.frame),
             measure="RaoQ",
             great.circle = gc.dist.fd)
```

If you don't have time to run the above functions, you'll want to read in the files

```
RQ <- readRDS(file="~/Documents/GitHub/MonitorPhylogenomics/SimulatedBoth_RaoQ_logData.RDS")
SESras <- readRDS(file="~/Documents/GitHub/MonitorPhylogenomics/SimulatedBoth_RaoQ_logData_SES_raster.R
```

Now we need to add the empirical FD (or richness) values to this data frame

```
#RQ <- cbind(RQ, emp.val=res.table$RaoQ)
RQ <- cbind(RQ, emp.val=res.table$FDis)
```

Then get standard effect sizes (SES) for each sell across all simulations

```
ses.vec <- NULL
for(k in 1:nrow(RQ)){
  curr <- RQ[k,]
  ses <- (curr$emp.val - curr$sim.mean) / curr$sim.sd
  ses.vec <- append(ses.vec, ses)
}
# bind it to the simulation dataframe
RQ <- cbind(RQ, ses=ses.vec)
```

Make a table of the ses values with the coordinates of each cell

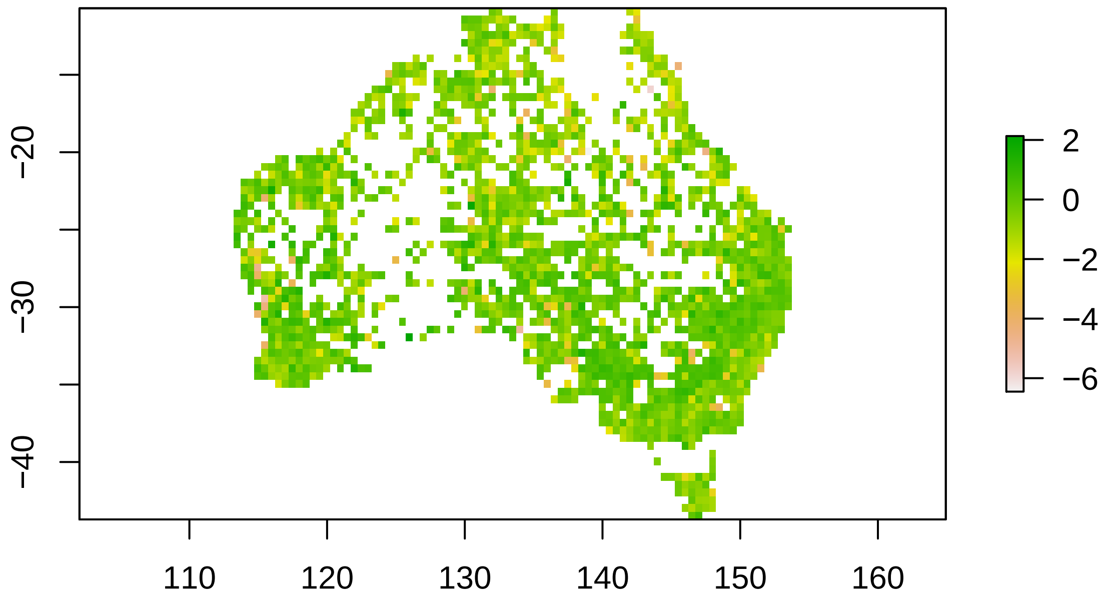
```
ses.table <- cbind.data.frame(latitude=gridded.dist$latitude, longitude=gridded.dist$longitude, SES=RQ$s
```

Bind the table with the empty raster cells we set up earlier, and make any NA values 0.

```
combo.SES <- left_join(rr.cells,
                       ses.table,
                       by=c("latitude", "longitude"))
```

Make an empty raster frame for the ses values to go into
Dump them into the raster
And plot it to make sure it makes sense

```
SESras <- rr;
SESras@data@values[] <- 0
SESras@data@values <- combo.SES$SES
#values(SESras) <- combo.SES$SES
plot(SESras)
```

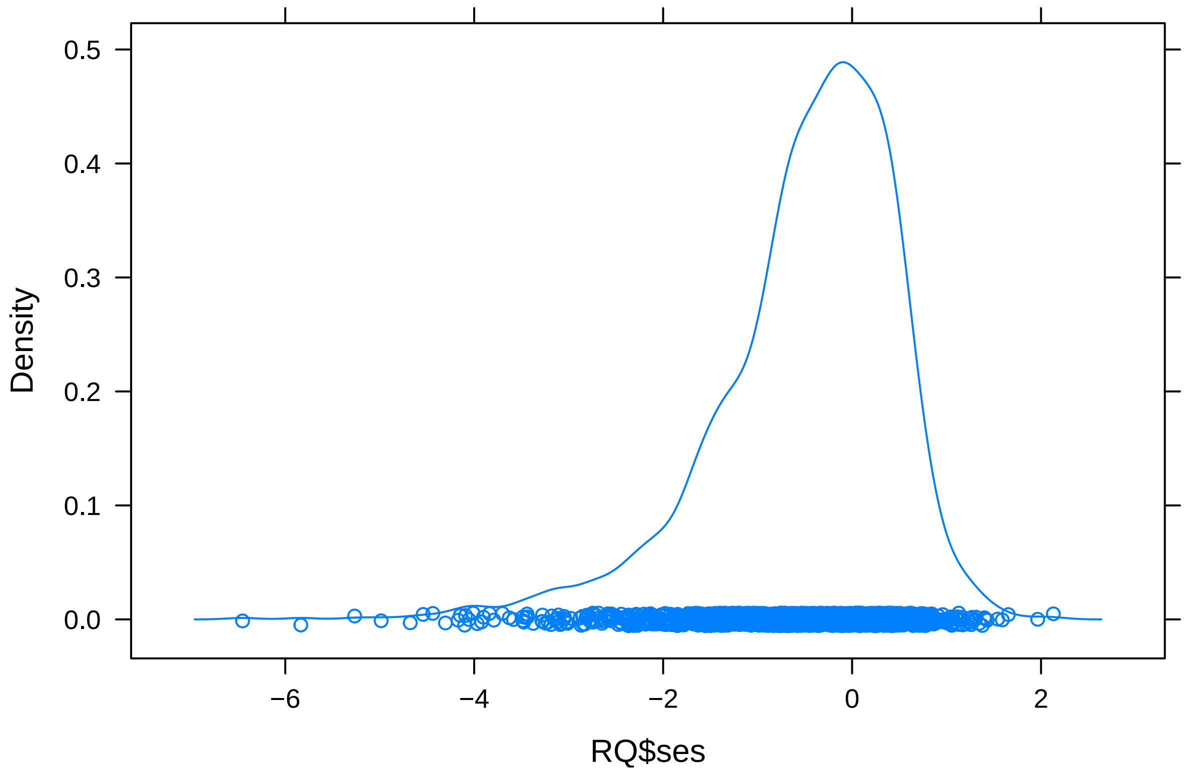


If you've been running all these steps from scratch, you'll want to save the output

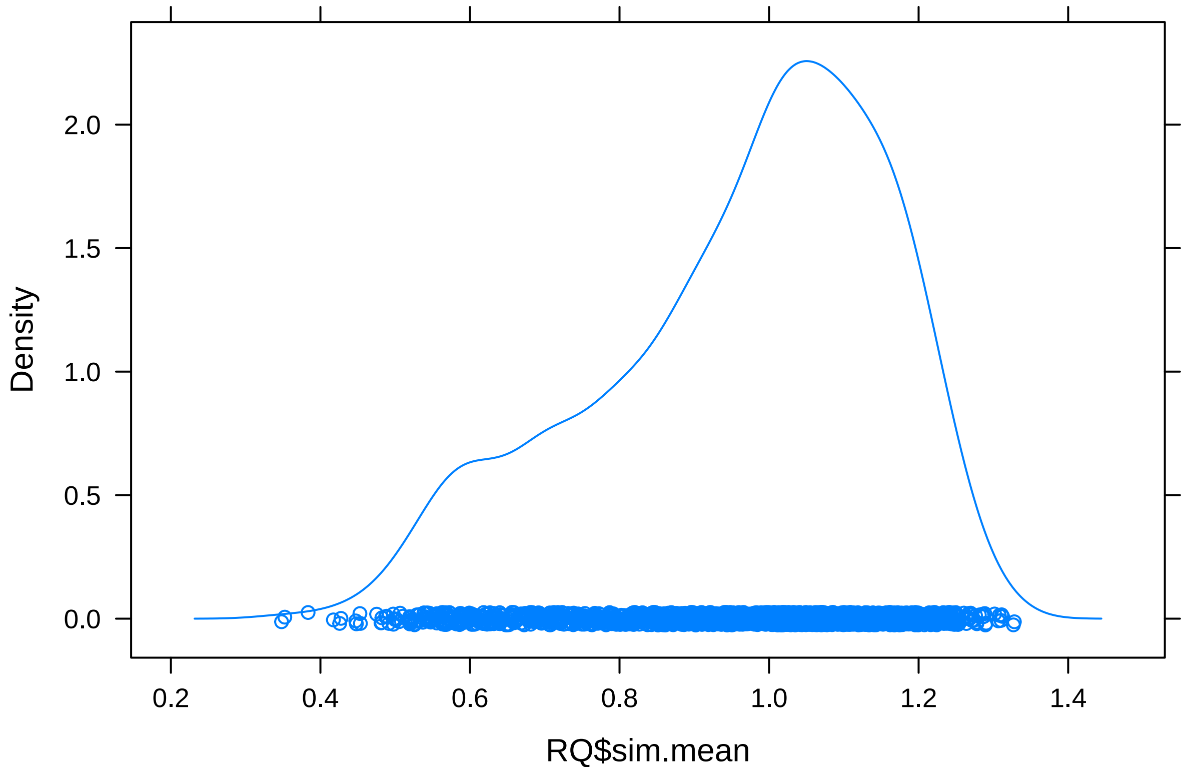
```
saveRDS(RQ, file="~/Documents/GitHub/MonitorPhylogenomics/SimulatedBoth_RaoQ_logData.RDS")
saveRDS(SESras, file="~/Documents/GitHub/MonitorPhylogenomics/SimulatedBoth_RaoQ_logData_SES_raster.RDS
```

Have a quick look at some of the parameters

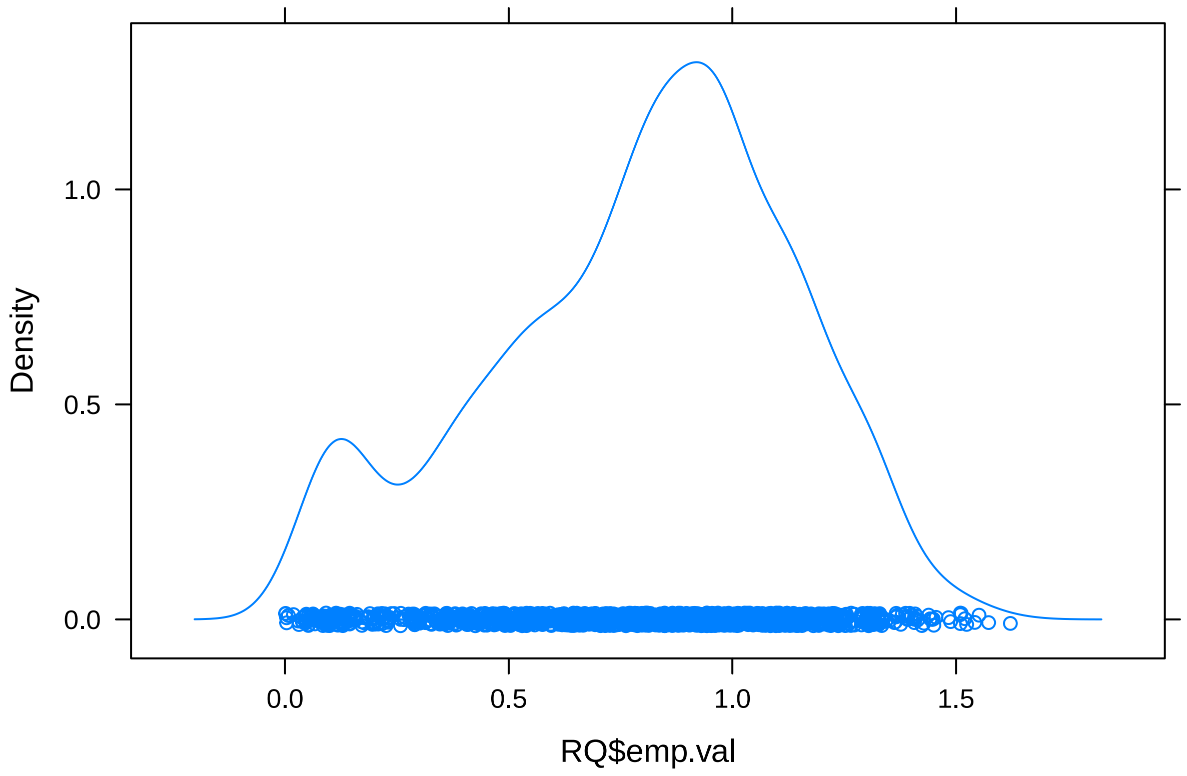
```
densityplot(RQ$ses)
```



```r
densityplot(RQ$sim.mean)
```



```r
densityplot(RQ$emp.val)
```



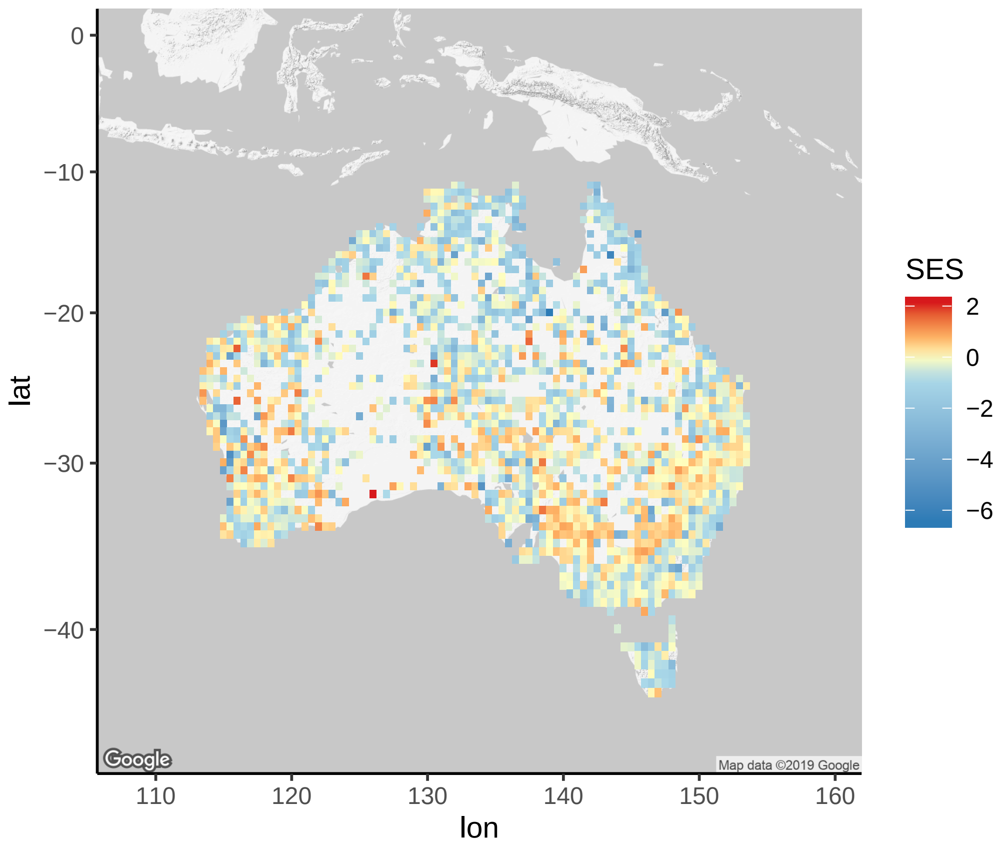
Translate the SES raster into polygons for plotting with ggmap

```
SESpoly <- rasterToPolygons(SESras);
max.colors <- length(unique(SESpoly$layer));
filled.SES <- rep(SESpoly$layer, each=5)
```

Lastly plot the map of SES (functional diversity)

```
ggmap(graymap) +
  geom_polygon(data = SESpoly,
               aes(x = long, y = lat, group = group,
                   fill = filled.SES), size = 0, alpha = 1)  +
  scale_fill_gradientn("SES", values=scales::rescale(c(min(ses.table$SES),
                                                       #min(ses.table$SES)/2,
                                                       -0.8,
                                                       0,
                                                       #max(ses.table$SES)/2,
                                                       0.8,
                                                       max(ses.table$SES))),
                       colors = rev(brewer.pal(5, "RdYlBu"))) +
  theme_classic()
```

```
## Regions defined for each Polygons
```



We want to know if the difference in simulated and observed FD values is signficant. So we'll create a function to calculate the confidence interval of the SES.

```
confidence_interval <- function(vector, interval) {
  # Standard deviation of sample
  vec_sd <- sd(vector)
  # Sample size
  n <- length(vector)
  # Mean of sample
  vec_mean <- mean(vector)
  # Error according to t distribution
  error <- qt((interval + 1)/2, df = n - 1) * vec_sd / sqrt(n)
  # Confidence interval as a vector
  result <- c("lower" = vec_mean - error, "upper" = vec_mean + error,
              "error" = error, "mean" = vec_mean, "sd" = vec_sd, "N" = n)
  return(result)
}
```

Can also be calculated as:

upper = mean + (error * 1.96)

lower = mean - (error * 1.96)

```
CIall <- confidence_interval(ses.table$SES, 0.95); CIall
```

```
##         lower         upper         error          mean            sd
##   -0.53272450   -0.44379796    0.04446327   -0.48826123    0.97853717
##             N
## 1863.00000000
```

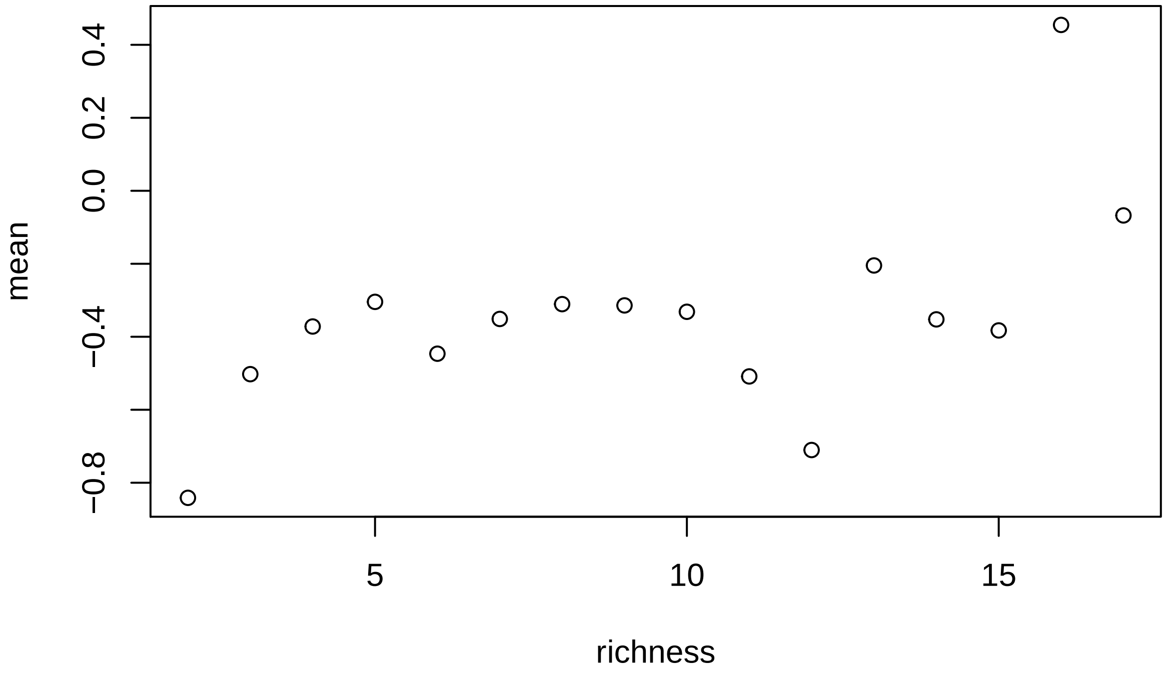
```
CIall["richness"] <- 1
```

```
siteRICH <- left_join(ses.table,
                      gridded.dist[,1:3],
                      by=c("longitude", "latitude"))
```

```
CIses <- NULL
for (i in min(siteRICH$richness):max(siteRICH$richness)){
  curr.rich <- filter(siteRICH, richness == i)
  CIses <- rbind(CIses, confidence_interval(curr.rich$SES, 0.95))
}
```

```
## Warning in qt((interval + 1)/2, df = n - 1): NaNs produced
```

```
CIses <- data.frame(CIses)
CIses$richness <- 2:17
plot(data=CIses, mean ~ richness)
```



There's also an easier way to do this with 'group.CI'

```r
library(Rmisc)
```

```
## Loading required package: plyr

## --------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)

## --------------------------------------------------------------------------

##
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following object is masked from 'package:maps':
##
##     ozone
```

```r
CIses2 <- group.CI(SES ~ richness,
         data=siteRICH,
         ci = 0.90)
```

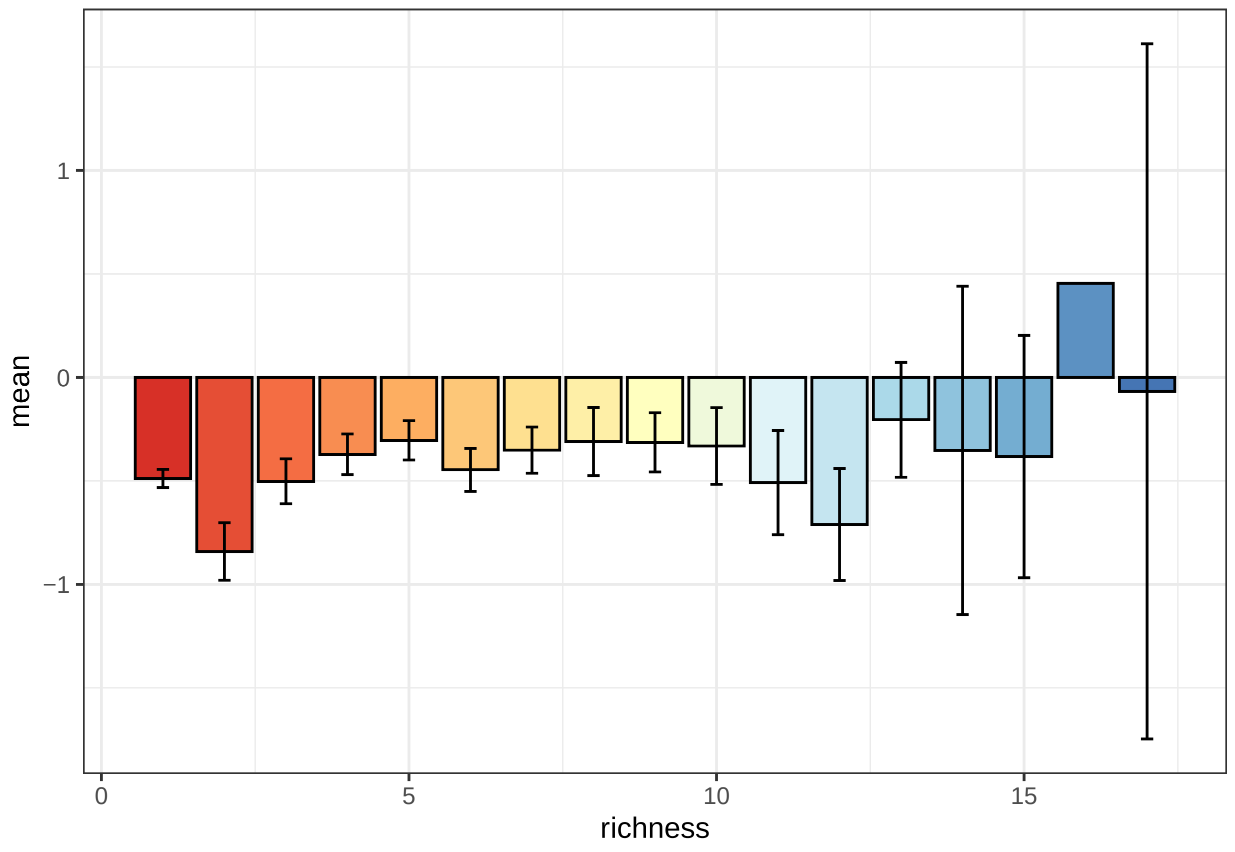
```
## Warning in qt(ci + (1 - ci)/2, df = n - 1): NaNs produced
```

Add the confidence interval for SES across the whole continent to the individual communities

```r
CIses <- rbind(CIses, CIall); CIses
```

```
##         lower       upper      error        mean        sd    N richness
## 1  -0.9799777 -0.70273700 0.13862037 -0.84135737 1.3797493  383        2
## 2  -0.6111262 -0.39385028 0.10863795 -0.50248822 1.0244188  344        3
## 3  -0.4704297 -0.27335183 0.09853892 -0.37189074 0.8643596  298        4
## 4  -0.3990601 -0.20963823 0.09471094 -0.30434917 0.7557160  247        5
## 5  -0.5503313 -0.34245597 0.10393767 -0.44639364 0.7145822  184        6
## 6  -0.4627334 -0.23970379 0.11151482 -0.35121860 0.6324771  126        7
## 7  -0.4751302 -0.14600575 0.16456224 -0.31056800 0.7250327   77        8
## 8  -0.4569476 -0.17121931 0.14286415 -0.31408346 0.5625626   62        9
## 9  -0.5162656 -0.14677388 0.18474588 -0.33151976 0.6568637   51       10
## 10 -0.7607036 -0.25664101 0.25203131 -0.50867232 0.7107791   33       11
## 11 -0.9809260 -0.43973639 0.27059482 -0.71033120 0.6408206   24       12
## 12 -0.4821964  0.07296914 0.27758279 -0.20461365 0.5398841   17       13
## 13 -1.1456556  0.44100276 0.79332918 -0.35232642 0.7559579    6       14
## 14 -0.9684752  0.20328734 0.58588127 -0.38259392 0.6334908    7       15
## 15         NA          NA         NA  0.45452696        NA    1       16
## 16 -1.7472785  1.61220378 1.67974112 -0.06753735 0.6761868    3       17
## 17 -0.5327245 -0.44379796 0.04446327 -0.48826123 0.9785372 1863        1
```

```r
library(RColorBrewer)
ggplot(CIses, aes(x=richness, y=mean)) +
  geom_bar(stat="identity", color="black",
           position=position_dodge(),
           fill = colorRampPalette(brewer.pal(9, "RdYlBu"))(17)) +
  geom_errorbar(aes(ymin=mean-error, ymax=mean+error), width=.2,
                position=position_dodge(.9)) +
  theme_bw()
```


mean
richness