# Monitor Phylogenomics

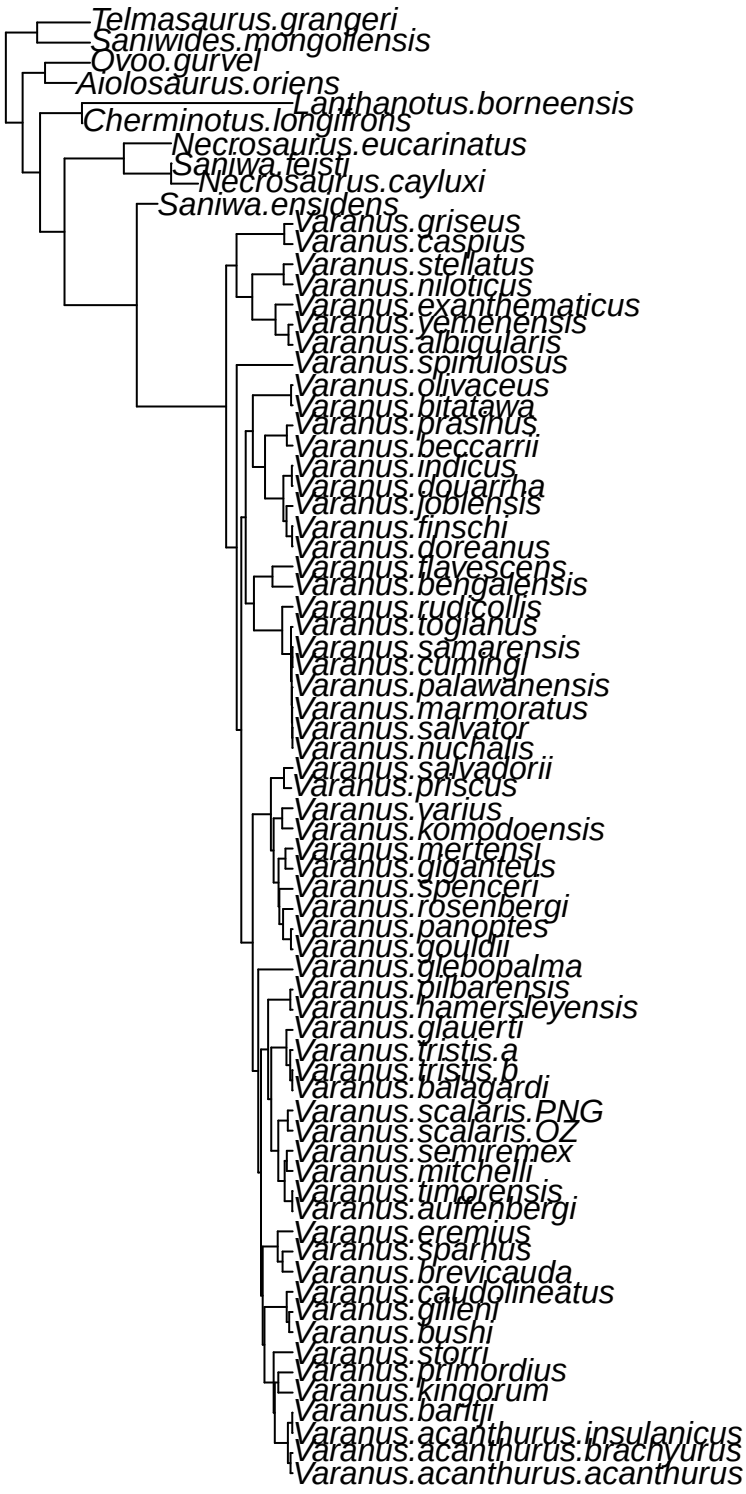
*Ian G. Brennan*

*25/10/2019*

## Look at our data

```
library(dplyr)
library(treeplyr)
library(RCurl)
library(phytools)
library(RColorBrewer)
# remember 'plyr' and 'dplyr' conflict, so don't load 'plyr'
```

```
gtree <- read.tree("~/Documents/GitHub/MonitorPhylogenomics/Varanidae_STRICT_HKY_270_con.newick");
plot.phylo(gtree)
```




```
alldata <- read.csv("~/Documents/GitHub/MonitorPhylogenomics/All_Size_Data.csv", header=T)
head(alldata)
```

```
##     SVL Tail         Name_in_Tree Location     Group Habitat
## 1  90.0   NA Antechinomys.laniger          Marsupial    <NA>
## 2  94.5   NA    Antechinus.agilis          Marsupial    <NA>
## 3 134.5   NA    Antechinus.bellus          Marsupial    <NA>
```

```
## 4 129.0   NA  Antechinus.flavipes          Marsupial    <NA>
## 5 133.0   NA   Antechinus.godmani          Marsupial    <NA>
## 6 151.0   NA       Antechinus.leo          Marsupial    <NA>
```

Use treeplyr to combine the data, then remove any missing

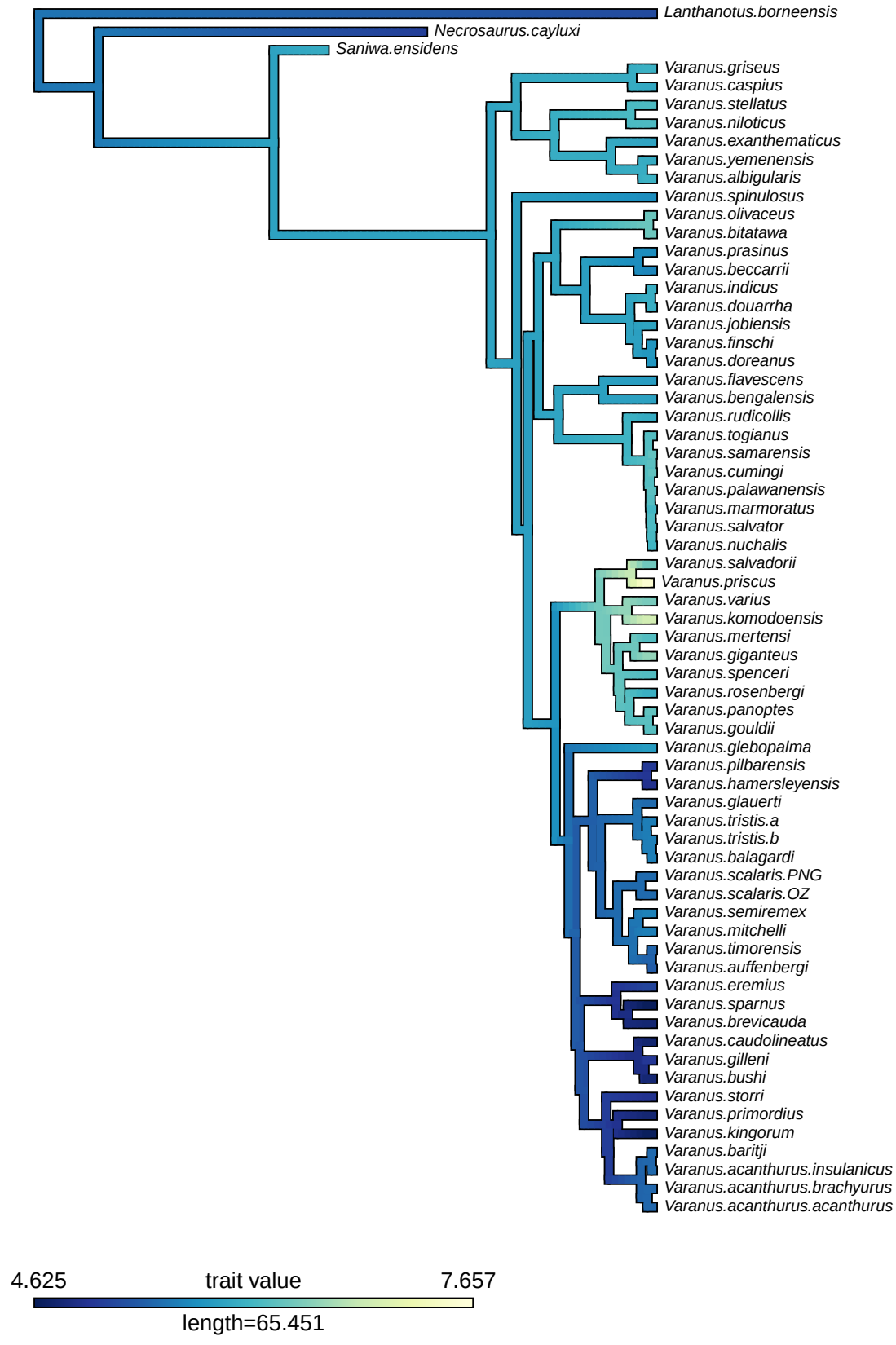
```
goanna <- make.treedata(gtree, alldata); # summary(goanna)
gf <- filter(goanna, is.na(Tail)==FALSE, is.na(SVL)==FALSE);
```

treeplyr is great because we can use all the tidyverse terms we already love.
Log-transform the SVL data, and make it a new column

```
gf <- mutate(gf, logSVL = log(gf$dat$SVL))
```

Start visualizing the trait using 'contMap' from 'phytools'

```
obj1 <- contMap(gf$phy, getVector(gf, logSVL), plot=FALSE, outline=F);
n<-length(obj1$cols);
obj1$cols[1:n] <- rev(colorRampPalette(brewer.pal(9, "YlGnBu"))(n));
plot(obj1, legend=0.7*max(nodeHeights(obj1$tree)),
     fsize=c(0.7,0.9), lwd=5, border=F); axisPhylo(1, backward=T)
```


Lanthanotus.borneensis
Necrosaurus.cayluxi
Saniwa.ensidens
Varanus.griseus
Varanus.caspius
Varanus.stellatus
Varanus.niloticus
Varanus.exanthematicus
Varanus.yemenensis
Varanus.albigularis
Varanus.spinulosus
Varanus.olivaceus
Varanus.bitatawa
Varanus.prasinus
Varanus.beccarrii
Varanus.indicus
Varanus.douarrha
Varanus.jobiensis
Varanus.finschi
Varanus.doreanus
Varanus.flavescens
Varanus.bengalensis
Varanus.rudicollis
Varanus.togianus
Varanus.samarensis
Varanus.cumingi
Varanus.palawanensis
Varanus.marmoratus
Varanus.salvator
Varanus.nuchalis
Varanus.salvadorii
Varanus.priscus
Varanus.varius
Varanus.komodoensis
Varanus.mertensi
Varanus.giganteus
Varanus.spenceri
Varanus.rosenbergi
Varanus.panoptes
Varanus.gouldii
Varanus.glebopalma
Varanus.pilbarensis
Varanus.hamersleyensis
Varanus.glauerti
Varanus.tristis.a
Varanus.tristis.b
Varanus.balagardi
Varanus.scalaris.PNG
Varanus.scalaris.OZ
Varanus.semiremex
Varanus.mitchelli
Varanus.timorensis
Varanus.auffenbergi
Varanus.eremius
Varanus.sparnus
Varanus.brevicauda
Varanus.caudolineatus
Varanus.gilleni
Varanus.bushi
Varanus.storri
Varanus.primordius
Varanus.kingorum
Varanus.baritji
Varanus.acanthurus.insulanicus
Varanus.acanthurus.brachyurus
Varanus.acanthurus.acanthurus
4.625
trait value
7.657
length=65.451

Trim the data just down to Australian goannas

```
oz.g <- filter(gf, Location == "Australia")
#summary(oz.g)
```

We need to load a bunch of additional packages and custom scripts, I'll try and explain what they're for briefly.

```
library(phytools); library(parallel)
library(RPANDA); library(deSolve)
library(rase)
```

This is a collection of additional functions for RPANDA, and includes the new models we'll use later

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

These functions make a *Geo Object* from spatial data and your *rase* output

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

A collection of functions for extracting the AIC values and weights for a set of models

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

A function for plotting distribution maps for a set of taxa. This also translates spatial data into spatial geometries and OWin objects.

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

This function processes a *rase* output object and makes distribution objects for ancestral nodes. More on that later...

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

The likelihood optimization can be hard given the number of parameters we're estimating, so I've created a function 'search.surface' that uses mclapply to fit the model a number of times with different starting parameters. It starts by creating sets of plausible starting parameters from across the surface, fits them and gives you the output either the best model fit, or all of the model fits. This function/method won't be necessary for simpler models like the BM or OU, and as implemented it won't work on models outside of the RPANDA framework.

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

## Spatial Data Processing

```
goanna.dist <- read.csv("~/Documents/GitHub/MonitorPhylogenomics/CoEvo_Goanna_Distributions.csv", heade
head(goanna.dist)
```

```
##         Name_in_Tree  Latitude Longitude
## 1 Varanus.acanthurus -20.50569  140.6596
## 2 Varanus.acanthurus -20.48583  140.6087
## 3 Varanus.acanthurus -21.28000  140.5000
## 4 Varanus.acanthurus -20.70217  140.4910
## 5 Varanus.acanthurus -21.70000  140.4717
## 6 Varanus.acanthurus -18.07472  140.4508
```

We'll be working with spatial data and focusing on the Australian radiation of monitor lizards now, so we'll need to trim the tree down again. We can't use treeplyr this time, so we'll go oldschool.

```
gtree$tip.label[1] <- "Varanus.acanthurus"
gtree$tip.label[18] <- "Varanus.scalaris"
keepers <- intersect(gtree$tip.label, unique(goanna.dist$Name_in_Tree))
gtree <- drop.tip(gtree, setdiff(gtree$tip.label, keepers))
```

Now we can plot the distributions of extant (tip) taxa.
*plot.distmaps* will loop through each taxon in the distribution dataframe (goanna.dist), and return a list three types of objects: • SpatialPoints which have been made from the lat/longs. • ConvexHulls which are distribution shape objects • OWin objects which are another type of distribution shape object

```
tips <- plot.distmaps(goanna.dist, new.directory = NULL, point.width = 0.25)
```

```
## plotting Varanus.acanthurus , 1 of 31
## plotting Varanus.balagardi , 2 of 31
## plotting Varanus.baritji , 3 of 31
## plotting Varanus.brevicauda , 4 of 31
## plotting Varanus.bushi , 5 of 31
## plotting Varanus.caudolineatus , 6 of 31
## plotting Varanus.doreanus , 7 of 31
## plotting Varanus.eremius , 8 of 31
## plotting Varanus.giganteus , 9 of 31
## plotting Varanus.gilleni , 10 of 31
## plotting Varanus.glauerti , 11 of 31
## plotting Varanus.glebopalma , 12 of 31
## plotting Varanus.gouldii , 13 of 31
## plotting Varanus.indicus , 14 of 31
## plotting Varanus.kingorum , 15 of 31
## plotting Varanus.komodoensis , 16 of 31
## plotting Varanus.mertensi , 17 of 31
## plotting Varanus.mitchelli , 18 of 31
## plotting Varanus.panoptes , 19 of 31
## plotting Varanus.pilbarensis , 20 of 31
## plotting Varanus.prasinus , 21 of 31
## plotting Varanus.primordius , 22 of 31
## plotting Varanus.rosenbergi , 23 of 31
## plotting Varanus.scalaris , 24 of 31
## plotting Varanus.semiremex , 25 of 31
## plotting Varanus.sparnus , 26 of 31
## plotting Varanus.spenceri , 27 of 31
## plotting Varanus.storri , 28 of 31
## plotting Varanus.tristis.b , 29 of 31
## plotting Varanus.tristis.a , 30 of 31
## plotting Varanus.varius , 31 of 31
```

We can quickly look at what these things are. Each object is also indexed for each tip taxon.
First check out the *SpatialPoints* objects

```
head(tips$SpatialPoints$Varanus.acanthurus)
```

```
## SpatialPoints:
##    Latitude Longitude
## 1 -20.50569  140.6596
## 2 -20.48583  140.6087
## 3 -21.28000  140.5000
## 4 -20.70217  140.4910
## 5 -21.70000  140.4717
```

```
## 6 -18.07472  140.4508
## Coordinate Reference System (CRS) arguments: NA
```

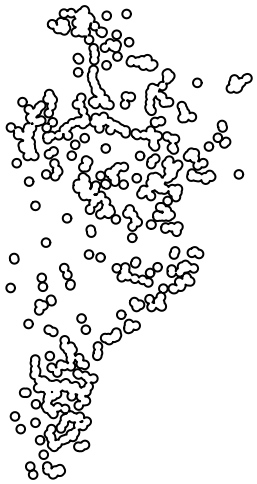
Next we can plot the shape of this distribution, which is an amalgamation of the point data with a buffer (of your choosing) around each point.

```
plot(tips$ConvexHulls$Varanus.acanthurus)
```

Finally, the *OWin* object for a given species should be identical to that species' *ConvexHull*

```
plot(tips$OWin$Varanus.acanthurus)
```

## tips$OWin$Varanus.acanthurus



Ok, now we can move on. Quickly sort the data to make sure the order matches the tree appropriately

```
tree_poly <- name.poly(tips$OWin, gtree, poly.names = unique(goanna.dist$Name_in_Tree))
```

Now we can run the *rase* MCMC sampler. I'm not going to actually run it here because it would take too long, so instead we'll load an object I've already run.

```
res <- rase(goanna.tree, tree_poly, niter=10000, logevery = 100)
```

If you have actually run it, then extract and plot the MCMC output to check for convergence.

```
resmc <- mcmc(res, start=(length(res[,1])*.2)) # remove 20% as burnin
par(mar=c(1,1,1,1))
plot(resmc)
```

If/when you do run *rase* you'll need to process the output so we can use it with the *CreateGeoObject_SP* function. It will do pretty much the same thing that the *plot.distmaps* function did, but for ancestral nodes/distributions. I'll explain how you use it here.

```
process.rase <- function(mcmc.object, distribution, new.directory=NULL,
                         remove.extralimital=NULL, range.shape, point.width=1)
```

Here are the basics:
- *mcmc.object*—is your *rase* mcmc output file
- *distribution*—is the data frame of distribution information
- *new.directory*—if you'd like to see the distributions of ancestral nodes, specify an output folder.
- *remove.extralimital*—because *rase* is a Brownian Motion dispersal process you might end up with samples in the ocean. We can remove these if we have a SHP file of the borders of our region, here Australia. Defaults to NULL if you don't have one.
- *range.shape*—if you set '*remove.extralimital*=TRUE', then the function expects the path to a .shp/x file.
- *point.width*—is the buffer size for your ancestral samples.

We'll read in the processed *rase* object instead of making a new one.

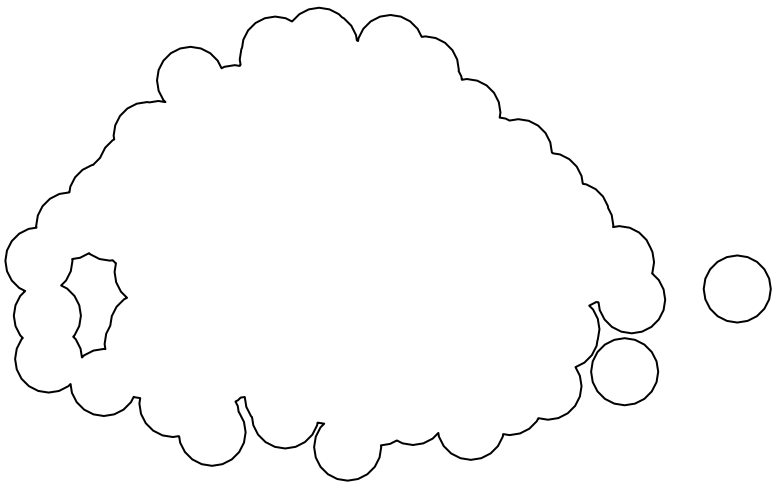
```
goanna.rase <- readRDS("~/Documents/GitHub/MonitorPhylogenomics/CoEvo_Varanus.0.25.RASE.RDS")
```

This object has all the same things as the object from *plot.distmaps*, in that you have *SpatialPoints*, *ConvexHulls*, and *OWin* shapes, as well as the raw distribution data *DistData*. The difference is that all of this information is for ancestral nodes (node numbers according to *ape*).
Quickly look at this information if you're interested. Our tree has 31 tips, so the root node should be n32.

```
# get the node number for the MRCA of V.gilleni and V.bushi
getMRCA(gtree, c("Varanus.gilleni", "Varanus.bushi"))
```

```
## [1] 43
```

```
# we know it's node 43 ('n43' in my notation), so we can plot the distribution of this ancestor
plot(goanna.rase$ConvexHulls$n43)
```



---

Great, we're ready to move on to making our *GeoObject*
In the script *CreateGeoObject_fromSP.R* there are actually two functions:

*CreateGeoObject_SP* works with a single tree, distribution data frame, and processed *rase* object. *CreateCoEvoGeoObject_SP* does much the same, but for the co-evolutionary scenario with two trees, a distribution data frame, and two processed *rase* objects.

```
goanna.geo.object <- CreateGeoObject_SP(gtree, goanna.dist, point.width=0.5)
```

Instead of running that, we'll just read in an existing file, but it shouldn't take long if you do run it (~45 seconds or so).

```
goanna.geo.object <- readRDS("~/Documents/GitHub/MonitorPhylogenomics/Goanna_0.25_geo.object.RDS")
```

The *GeoObject* holds a lot of information, so it's not really useful to go through everything, but I'll try to make a short explanation so it isn't a total black box.

• *geography.object* is a list of the interaction matrices for all taxa that exist at a given time in the tree.

```
goanna.geo.object$geography.object[[3]]
```

```
##                     Varanus.glebopalma .AAB .AAD .AAE
## Varanus.glebopalma                  1    1    1    1
## .AAB                                1    1    1    1
## .AAD                                1    1    1    1
## .AAE                                1    1    1    1
```

```
# this shows the interaction matrix at the third time point (third cladogenetic event)
```

• *times* are the occurrence times of the cladogenetic events, as time since the root (0).

```
goanna.geo.object$times
```

```
##  [1]  0.000000  6.936469  9.839077 11.766469 12.616739 14.045740 15.130049
##  [8] 16.267532 16.382653 17.409324 18.781343 19.360130 20.707024 21.615989
## [15] 22.609906 22.877148 23.680182 23.768804 23.879711 23.899351 24.841652
## [22] 24.969668 25.334848 25.498787 25.755498 27.199174 28.789539 29.053123
## [29] 29.643692 29.731313 30.024877
```

```
# goanna.geo.object$times[[3]] is the time of the matrix above
```

• *spans* are the amount of time between each cladogenetic event.

```
goanna.geo.object$spans
```

```
##  [1] 6.93646905 2.90260825 1.92739208 0.85026958 1.42900062 1.08430957
##  [7] 1.13748316 0.11512033 1.02667170 1.37201875 0.57878699 1.34689416
## [13] 0.90896442 0.99391703 0.26724258 0.80303362 0.08862244 0.11090706
## [19] 0.01963949 0.94230126 0.12801605 0.36518000 0.16393878 0.25671074
## [25] 1.44367643 1.59036531 0.26358380 0.59056876 0.08762114 0.29356387
```

• *name.matrix* is a matrix of all the internal node names and associated information. The left column are node/tip numbers. The center column is the taxon name of the edge which descends from that node, and the right column is the node that the edge ends at. Each node appears in the left column twice because it gives rise to two edges, which end at two different nodes (or tips). This is probably of no interest to anyone.

We now have a *GeoObject* that we can use for the geography-informed models. Next up we'll start fitting some models of trait evolution.

## Model Fitting

We can have a look at the *search.surface* function quickly to see what it does and how it works.

```
search.surface <- function(model, n.iter = 10, traits, n.proc = 8,
                           no.S=1, results=c("best", "all"), start.params=NULL)
```

The function lets us control a few things via the commands:
• *model*—the model of interest, you must have built this already
• *n.iter*—the number of model fittings you'd like completed, defaults to 10
• *traits*—the input traits for model fitting
• *n.proc*—number of processors. this function will fit the model in parallel.
• *no.S*—defaults to 1. if your model requires estimating/fitting more than 1 S parameter, say so
• *results*—would you like the function to report just the best fitting run, or all the results from each fit attempt

Now we need to build & fit a set of models just to the goanna data to compare with previous hypotheses about how body size variation evolved

Let's fit a model for goannas based on habitat partitioning (Collar et al. 2011).
This is a multi-OU model with different "adaptive optima" per habitat/niche type.

```
library(OUwie)
source("/Users/Ian/Google.Drive/R.Analyses/Convenient Scripts/make.OUwie.input_Script.R")
goanna.habitat <- make.OUwie.input(data=oz.g$dat, regime=oz.g$dat$Habitat,
                  taxa=oz.g$phy$tip.label, phy=oz.g$phy, trait=oz.g$dat$logSVL)
fitOUM <- OUwie(goanna.habitat$regime.simm, goanna.habitat$combined, model="OUM", simmap.tree=T)
```

Now the standard Brownian Motion (BM) model (Felsenstein, 1985)

```
modelBM <- createModel(oz.g$phy, keyword="BM")
show(modelBM)
fitBM <- fitTipData(modelBM, getVector(oz.g, logSVL), GLSstyle=T)
show(fitBM)
```

Next an Ornstein-Uhlenbeck (OU) model, BM with a single "adaptive optimum"

```
modelOU <- createModel(oz.g$phy, keyword="OU")
show(modelOU)
fitOU <- fitTipData(modelOU, getVector(oz.g, logSVL), GLSstyle=T)
show(fitOU)
```

Build the PM model, which estimates a single S parameter

```
modelPM <- createModel(oz.g$phy, keyword="PM")
show(modelPM)
fitPM <- fitTipData(modelPM, getVector(oz.g, logSVL), GLSstyle=T)
show(fitPM)
```

The geography-informed models below require a processed rase object,

Build the PM+geo model, which estimates a single S parameter, and includes geography

```
modelPMgeo <- createGeoModel(oz.g$phy, goanna.geo.object, keyword="PM+geo")
fitPM_geo <- search.surface(modelPMgeo, n.iter=8, traits=gtraits,
                            n.proc=4, no.S=1, results="best") # this took ~1 min
```

Build the PM OU-less model, which estimates a single S, no alpha, and includes geography

```
modelPMOU_geo <- createGeoModel(goanna.tree, goanna.geo.object, keyword="PMOU+geo")
fitPMOU_geo <- fitTipData(modelPMOU_geo, gtraits, GLSstyle=T)
```

Summarize the model fits

```
multiphy.AIC(prefix="fit", phylo=goanna.tree, models=c("BM", "OU", "PM_geo",
                                                       "PMOU_geo", "OUM"))
```

If you wanted to simulate some data to see what it looks like under the inferred parameters

```
simulateTipData(modelBM, fitBM$inferredParams, method=2)
simulateTipData(modelPMOU_geo, fitPMOU_geo$inferredParams, method=2)
```

We could also quickly check that the inferred params result in simulated data that can recover the correct model:

```
testPMOUg <- simulateTipData(modelPMOU_geo, fitPMOU_geo$inferredParams, method=2)
test_modelBM <- createModel(goanna.tree, keyword="BM")
    test_fitBM <- fitTipData(test_modelBM, testPMOUg, GLSstyle=T)
test_modelPMOU_geo <- createGeoModel(goanna.tree, goanna.geo.object, keyword="PMOU+geo")
    test_fitPMOU_geo <- fitTipData(test_modelPMOU_geo, testPMOUg, GLSstyle=T)
multiphy.AIC(prefix="test_fit", goanna.tree, c("BM", "PMOU_geo"))
```

Reminder: you can check the composition of a model using '@', e.g.:

```
modelGMM@aAGamma(3, ssGMM$inferredParams);
modelGMM@aAGamma(3, ssGMM$inferredParams)$A / ssGMM$inferredParams[5]
```