Introduction to the `surface` package

Travis Ingram

This tutorial walks through the steps of a SURFACE analysis using a simple demonstration data set. The main use of SURFACE is to construct a macroevolutionary adaptive landscape for a clade, given only a phylogenetic tree and measurements of one or more continuous traits for each member species. This is done by using stepwise AIC algorithms to fit a series of 'Hansen' models using `ouch` functions, with two distinct phases. In the *forward* phase, new selective regimes are added to the model, and in the *backward* phase multiple regimes may be 'collapsed' into convergent regimes discovered independently by different lineages. The final model provides an estimate of the macroevolutionary adaptive landscape, and can be compared to simulated data to evaluate whether the clade contains exceptional phenotypic convergence.

The `surface` package can be installed locally from source files, and will soon be available for download from CRAN. Also required are the packages `ape`, `ouch`, `geiger`, `igraph` and `pmc`, and their dependencies. First, load the `surface` package along with its dependencies, then import the data object `surfaceDemo` contained in the package. This object consists of a 25-taxon tree in `phylo` format, and a simulated data set that includes measurements of three continuous traits for each species.

```
> library(surface)
> data(surfaceDemo)
> tree<-surfaceDemo$tree
> dat<-surfaceDemo$sim$dat
```

At this point, the `tree` and `dat` objects could be provided as input to the wrapper function `runSurface`, which carries out both phases of the analysis in a single step. Instead, let's look at the two steps separately to make the sequence of events clear. The function `nameNodes` ensures that each node in the tree has a unique label, for easier conversion between formats later (if nodes are already labeled, as is the case for the demo tree, the function returns the tree unchanged). `convertTreeData` converts the tree and data objects into a format ready for analysis with the `ouch` function `hansen`.

```
> tree<-nameNodes(tree)
> olist<-convertTreeData(tree,dat)
> otree<-olist[[1]]; odata<-olist[[2]]
```

The data set is now ready for analysis with SURFACE. We use the function `surfaceForward` to add one selective regimes at a time to the model, by repeatedly calling the function `addRegime`. By default, the starting model is a single-regime OU model, although one can optionally start an analysis with a model containing some regimes by specifying a `starting_list`.

The default arguments should be suitable for many analyses, but one can specify options such as a different `aic_threshold` for accepting model improvements, a limit to the `max_steps` of the algorithm, whether to `sample_shifts` within `sample_threshold` AIC units of the best model rather than choosing the best model at each step, and whether to `exclude` a fraction of the worst candidate models from the previous step (see documentation for `surfaceForward` for more). By default progress is not printed out, but one can choose to print (fairly extensive) output to the console with `verbose=TRUE`, to view a visual representation of the change in AIC at each step in the graphics device with `plotaic=TRUE`, or to save the output to a file `filename` at each step to guard against crashes or exceeding run time limits using `save_steps=TRUE`. This step should take a little under a minute, but can be quite long for larger data sets.

```
> fwd<-surfaceForward(otree, odata, aic_threshold = 0, exclude = 0,
+ verbose = FALSE, plotaic = FALSE)
> k<-length(fwd)
```

The object returned by `surfaceForward`, which has been named `fwd`, is a list of length `k`, each element of which is another list containing the output of one call to `addRegime`. The object `fwd` is unwieldy to view all at once, but `surfaceSummary` can be used to retrieve information about the forward phase, such as the sequence of AIC values:

```
> fsum<-surfaceSummary(fwd)
> names(fsum)

[1] "n_steps"       "aics"          "shifts"        "n_regimes"
[5] "alpha"         "sigma_squared" "theta"

> fsum$aics

       1        2        3        4        5        6
254.6531 247.0146 241.2537 237.5858 218.8915 184.9636
```

`surfaceSummary` also returns the parameter estimates and regime placements from `fwd[[k]]`, the model returned from the forward phase of SURFACE, which is also the starting model for the backward phase. `surfaceBackward` repeatedly calls the function `collapseRegimes`, identifying cases where the same (or very similar) regimes are found independently on different branches of the tree, and where the model simplification obtained by collapsing them into single regimes results in a further improvement in the AIC. By default multiple regimes can be collapsed at each step if they are mutually compatible (see documentation), but will be limited to one collapse per step if the option `only_best=TRUE` is specified. Other defaults and options are similar to the forward phase. This step will run in a few seconds, but the time taken increases quickly with the number of taxa and the number of regimes found in the forward phase.

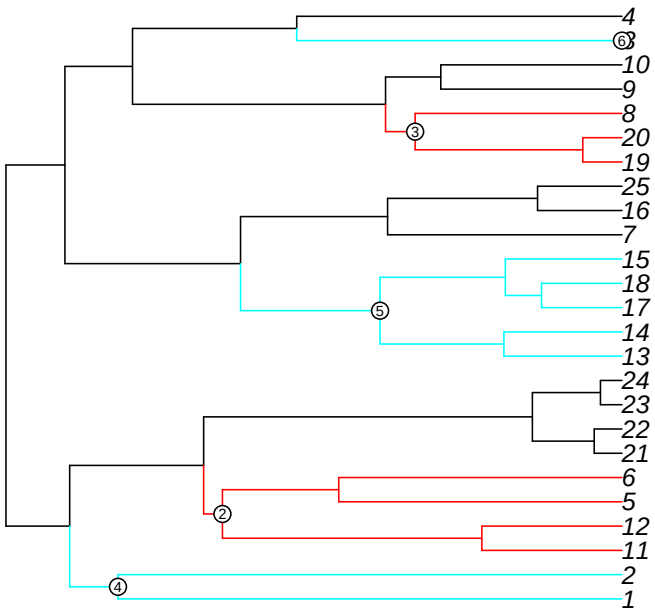
```
> bwd<-surfaceBackward(otree, odata, starting_model = fwd[[k]], aic_threshold = 0,
+ only_best = FALSE, verbose = FALSE, plotaic = FALSE)
> bsum<-surfaceSummary(bwd)
> kk<-length(bwd)
```

`surfaceBackward` returns another list of lists, and again the final element (`bwd[[kk]]`) is generally the element of interest. `surfaceSummary` again summarizes the steps and final parameter values, and also displays measures of the extent of convergence in the data set (`deltak` and `c`).

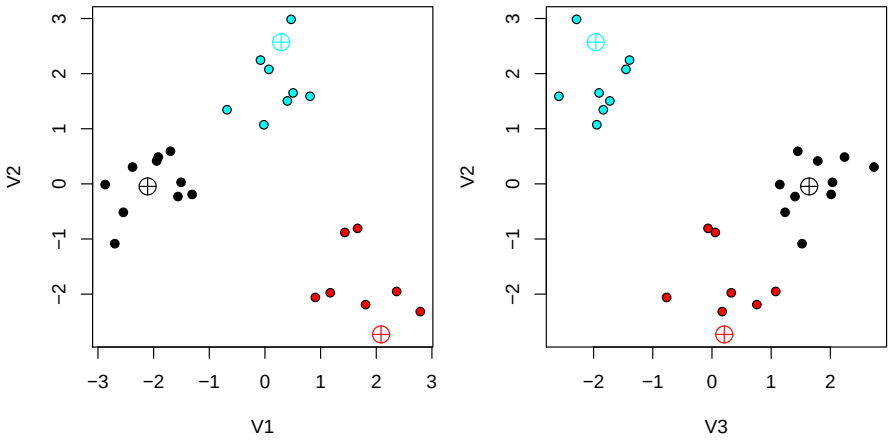
```
> bsum$alpha

       V1        V2        V3
0.4953761 0.2062759 0.6356926

> bsum$sigma_squared

       V1        V2        V3
0.2569052 0.1071173 0.2672484

> bsum$theta

          V1          V2         V3
a -2.1074803 -0.04566228  1.6418668
b  2.0867557 -2.73101221  0.2075315
d  0.2913039  2.57074245 -1.9619587

> bsum$n_regimes

             k         kprime         deltak              c    kprime_conv
             6              3              3              5              2
kprime_nonconv
             1
```

The fitted SURFACE model can be visualized either as paintings of regimes on the tree, or as color-coded points in trait space. The first two traits are shown here, and by default `convcol = TRUE`, meaning convergent regimes are colorful and non-convergent regimes are grey or black.

```
> surfaceTreePlot(tree, bwd[[kk]], labelshifts = T)
```




```
> par(mfrow=c(1,2), mai=c(0.8,0.8,0.2,0.2))
> surfaceTraitPlot(dat, bwd[[kk]], whattraits = c(1,2))
> surfaceTraitPlot(dat, bwd[[kk]], whattraits = c(3,2))
```



The function `propRegMatch` can be used to compare the fitted model to the 'true', generating model based on the proportion of pairs of branches (either all branches or only taxa at the tip branches) that are correctly assigned to either the same regime or to different regimes. In this simple case, SURFACE recovers the generating model perfectly, so both values are `1`.

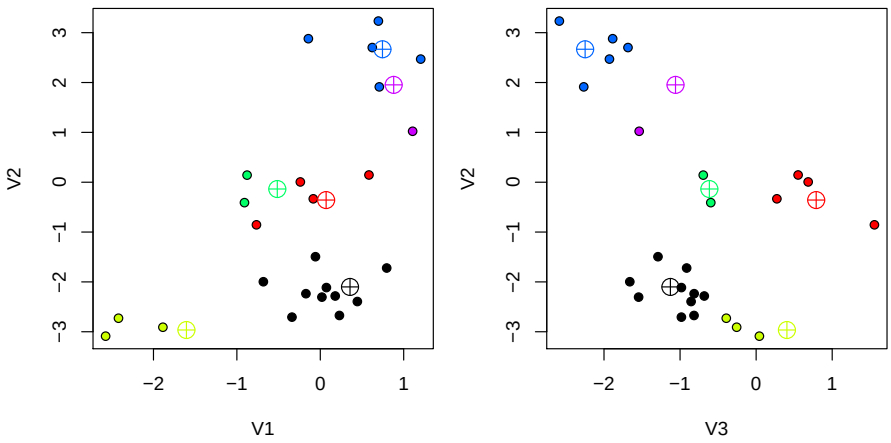
```
> truefit<-surfaceDemo$sim$fit
> propRegMatch(truefit, bwd[[kk]]$fit, internal = FALSE)

[1] 1

> propRegMatch(truefit, bwd[[kk]]$fit, internal = TRUE)

[1] 1
```

To compare the extent of convergence in a data set to a null expectation, data can be simulated under a model that lacks true convergence: either a simple stochastic model such as Brownian motion, or a Hansen model based on the fitted model from the forward phase; with multiple evolutionary regime shifts to *non-convergent* adaptive peaks. Either can be done using `surfaceSimulate`: the following code carries out one simulation under the Hansen null model (`set.seed` is used here only for reproducibility).

```
> set.seed(10)
> newsim<-surfaceSimulate(tree, type="hansen-fit", hansenfit=fwd[[k]]$fit,
+ shifts=fwd[[k]]$savedshifts, sample_optima=TRUE)
```

We can then visualize the simulated trait data; as there is no convergence in the true model, setting `convcol = FALSE` makes a more colorful figure.

```
> par(mfrow=c(1,2),mai=c(0.8,0.8,0.2,0.2))
> surfaceTraitPlot(newsim$data, newsim, whattraits = c(1,2), convcol = FALSE)
> surfaceTraitPlot(newsim$data, newsim, whattraits = c(3,2), convcol = FALSE)
```



We can then run SURFACE on the simulated data set, here doing the entire analysis in one step using `runSurface` (this should take less than a minute). We can then use `surfaceSummary` to extract the results, and compare the number of regime shifts (`k`) and the extent of convergence (`deltak` or `c`) to what we saw in the 'real' data set. In this case, one instance of convergence is recovered in the simulated data set where two regimes were relatively close to one another in trait space; as we know that the regimes were nonetheless distinct from one another in the generating model, this represents 'incidental' convergence.

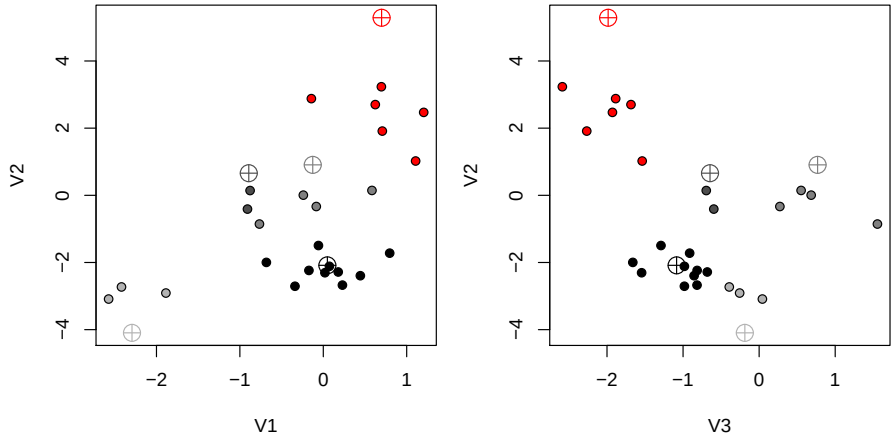
```
> newout<-runSurface(tree, newsim$dat)
> newsum<-surfaceSummary(newout$bwd)
> newkk<-length(newout$bwd)
> newsum$n_regime
```

```
            k          kprime          deltak               c     kprime_conv
            6               5               1               2               1
kprime_nonconv
            4
```

```
> bsum$n_regime
```

```
            k          kprime          deltak               c     kprime_conv
            6               3               3               5               2
kprime_nonconv
            1
```

```
> par(mfrow=c(1,2),mai=c(0.8,0.8,0.2,0.2))
> surfaceTraitPlot(newsim$data, newout$bwd[[newkk]], whattraits = c(1,2))
> surfaceTraitPlot(newsim$data, newout$bwd[[newkk]], whattraits = c(3,2))
```



If we wanted to do a proper hypothesis test of whether the 'real' data set contains more convergence than expected by chance, we would repeat the step of simulating data sets (with different random number seeds) many times, and running SURFACE on them to get a null distribution of `deltak` or `c`. We could calculate the statistical significance of such a test as the proportion of null values that meet or exceed the observed value (note that in a small data set like this statistical power is poor, but that statistical properties are much better given larger trees and at least two trait dimensions).

This tutorial has given an overview of the uses of SURFACE; more information about the method is found in the documentation for each function, and in the manuscript (Ingram and Mahler, currently in revision for *Meth. Ecol. Evol.* that introduces the method.