RevBayes_Setup.r will generate RevBayes scripts and appropriate data files that partition characters by numbers of states and general character evolution models. However, the generated RevBayes scripts will require some editing and/or verification before they can be used on your computer. There are two sources of editing that you will need to do. The first concerns the directories invoked by the scripts. Open the file "Cincta_FBD_Analysis_Strict_Clock.Rev." The first line:

```
clear();
```

This simply clears all information from the RevBayes program. The second line includes the command:

```
Setwd("/Users/user_name/Documents/RevBayes_Projects/");
```

"user_name" in your file should not be user_name, but instead be the name of your user folder on your computer. This should have been entered in the Cincta.setup.R in the variable **set_wdir**. If not, then change it to your user folder name now.

RevBayes_Projects should contain three folders:

```
RevBayes_Projects/scripts
RevBayes_Projects/data
RevBayes_Projects/output
```

"Cincta_FBD_Analysis_Strict_Clock.Rev" will look in the first folder for source files that are additional RevBayes scripts for performing MCMC analyses. It will look in the 2nd folder for the relevant data files. It will send output to the third folder.

"Cincta_FBD_Analysis_Strict_Clock.Rev" will call on three types of RevBayes source files. The type is denoted by the first word of the files, which are named after appropriate Harry Potter spells:

Imperio_Source_Name.Rev: these scripts instruct RevBayes how to "think" about the data while conducting analyses.

Accio_Source_Name.Rev: these scripts fetch (summon) data such as taxonomic data with stratigraphic ages and the different partitions of the character data.

Expecto_Source_Name.Rev: these execute the analyses and thus (hopefully!) summon plausible possible histories of your organism.

The first of these source files that is executed is "Imperio_Default_Settings.Rev." As the name indicates, this provides default values for variables that provide a simple strict clock analysis. (Scripts for more complex models will require changing one or more of these variables in the script.) Among other things, the defaults set the clock model (**clock_model**) to “strict” and the number of rate partitions to 1. This later will result in all character partitions and all branches having the same general rate of character change

The next set of commands inform RevBayes about the character data. The first one

```
        filenames <- v(…);
```

inform the analysis there are three nexus files, which were created by the RevBayes_Setup.r program. The subsequent commands:

```
        partition_states <- v(2,3,4);
        partition_ordering <- v("unordered","unordered","unordered");
```

will tell RevBayes the number of states shared by characters in each partition and that there is no “ordering” to the characters. This information will be used below to establish simple Mk (M2, M3 and M4) models logically equivalent to the Jukes-Cantor model.

The next variable informs RevBayes about ascertainment bias within each matrix. Here, Smith & Zamora included some characters that are invariant among all of the cinctans. These are treated as invariant binary characters, so:

```
        coding_bias[1] = “all”;
```

If this is changed to either “variable” or “informative”, then the MCMC analyses will ignore the invariant characters. If there were no invariant binary characters but some autapomorphic binaries (i.e., binary characters with one state shown by only one analyzed taxon), then **coding_bias**[1] would equal “variable.” If there are no invariant or autapomorphic characters, then **coding_bias**[1] would be “informative.” For either “variable” or “informative” characters, the likelihood component of the MCMC analyses will perform a correction akin to Chao2 inferences of unsampled species to infer some number of “unsampled” characters that *could* have varied among the observed species, but that simply have not changed among the sampled taxa.

In this example, **coding_bias**[2] and **coding_bias**[3], are set to “variable.” This is because 3-state, 4-state, etc., characters logically demand the possibility of autapomorphic character states. That is, if 10 species have “circular” and 10 species have “squared” to describe the shape of some feature, then a species with a triangular shape demands its own state. So, if we observe 3+ states, we know that we have to code unique features in order to retain an informative character. (In contrast, autapomorphic binary characters are not informative in traditional phylogenetic analyses and thus might be deliberately excluded by authors.)

The final variable here is:

```
	max_age <- 7.3;
```
This is not important only for more complex models considering early-bursts in character change rates. However, it is included in the generic script simply because RevBayes_Setup.R can provide the information and thus save the user from looking it up separately.

The next set of variables also are read from the original nexus file. This sets the outgroup taxon to the genus *Ctenocystis*. The ingroup is set to the rest of the clade. Because four cinctan species were excluded from Smith & Zamora's original analysis, these are stored in the vector **unscored_taxa**. (This will affect the tree priors but not the tree likelihoods.)

The final variable here has to be set by the user. RevBayes_Setup.r will return:
```
among_char_var <- 
"ENTER_THE_AMONG_CHARACTER_RATE_DISTRIBUTION_YOU_WISH_TO_USE_HERE";  # enter 
"gamma" or "lognormal"
```
This establishes the model of rate variation among characters within any one ratee-partition and should be set to either:
```
	among_char_var <- "lognormal";
```
or
```
	among_char_var <- "gamma";
```
The scripts below will use this to establish models of rate variation among characters.

The next part of Cincta_FBD_Analysis_Strict_Clock.Rev collects basic information about the data set. The first variable,
```
	n_data_subsets <- filenames.size();
```
will be used to tell subsequent scripts how many character data files must be examined. The next command:
```
	taxa <- readTaxonData(file="data/cincta_fossil_intervals_FA.tsv");
```
reads in data about the taxa that will be used in the fossilized birth-death (FBD) analyses. This file was generated by RevBayes_Setup.R and must be in the folder RevBayes_Projects/data (or whatever the equivalent of that fold is on your computer.). Note that this file uses the first appearances for both the min and max ages; the current versions of RevBayes appear to be using the file incorrectly so that the minimum age (last appearance) is a possible divergence time, so that trees with species diverging *after* they first appear are possible!

The variable **taxa.size** provides the number of taxa (**n_taxa**); this is used in scripts below, and also to establish the maximum number of possible branches (**n_branches**).

At this point, the script begins to prepare the actual analyses. The command:

```
moves = VectorMoves();
```

sets up a vector of parameter-alterations that RevBayes will consider in each generation of the MCMC searches. The next two steps will setup the tree & fossilized birth-death (FBD) models, and then the character evolution models.

The program first sets up trees & FBD model for tree-priors using "Accio_Cincta_Range_Based_FBD_Parameterization.Rev." This file was created from PaleoDB data by RevBayes_Setup.r. The first variables that are set are:

```
speciation_rate ~ dnExponential(1.471);
extinction_rate ~ dnExponential(1.471);
```

This sets both speciation (really, cladogenesis) and extinction rates to be sampled from exponential distributions in which the original rate (1.471) represents an estimate of the average per-million-year origination and extinction rates for Cambrian echinoderms given PaleoDB data. Three ways of altering both rates are added to the **moves** vector, with lambda affecting how "big" of a move is expected and "weight" determining how many times the move will be made in an MCMC generation. Here, several small moves will be used for every large move. These moves mean that the final estimates of average origination & extinction could be very different from what we would estimate using paleobiological methods. On the other hand, using numbers that should be fairly realistic might speed up the runs needed to achieve convergence in the MCMC analyses.

(NOTE: I have occasionally run into origination & extinction seed values that seem to "freeze" RevBayes; I have found that just changing the numbers very slightly usually sets things right.)

Finally, note that two deterministic variables are established: diversification and turnover. The ":=" assignment means that these will be automatically updated every time either speciation or extinction changes. Neither of these variables are used in analyses, but they are included in the output because they might be of interest to the user.

The next set of commands defines the sampling parameter, **psi**.

```
psi ~ dnExponential(3.892);
```

Note that as the number in the exponential increases, the log-decay of the exponential distribution increases and the expected number of samples decreases. Here, the original number represents a weighted average of median sampling rates given best-fit uniform, exponential, Beta and lognormal distributions (with the weight proportional to the likelihood of the best-fit distributions given occurrences in different stage-slices in the Cambrian.)  Here, the sampling is based on numbers of *collections* in which we find Cambrian echinoderm species in each stage-slice given the total number of collections in those stage-slices.  (Another alternative would have been to use numbers of rock-units, e.g., formations & members.)  We again assign moves to the MCMC analysis just as we did for origination & extinction; as before, although we seed the analysis with an empirical estimate, other values will be preferred if they increase the posterior probabilities of trees.

An additional deterministic variable is established here, completeness.  This reflects the average sampling and extinction rates; as with diversification & turnover, this variable is automatically updated every time sampling or extinction changes.  It is not used in the analyses but reported in case users are interested.

A final sampling parameter is included, **rho**.  This is the probability of sampling individual taxa in the final interval of the study.  This might seem to be odd for paleontologists, but it is necessary for analyses of fossil & extant taxa in which extant taxa typically have much higher rates of sampling than the fossil taxa.  Here, rho is estimated from the best-fit sampling distribution for the latest interval in which an analyzed cinctan species appears.  This is also why the ages of the taxa in cinctan_intervals_FA.tsv are scaled so that 0 represents when that youngest known species appears: RevBayes will apply **rho** only for age 0.  Here, PaleoDB data suggest **rho**=0.506.

Finally, we use:

```
	origin_time ~ dnUnif(7.3, 12.11);
```

to put limits on plausible origination times for the entire clade.  Here, 7.3 represents the earliest possible appearance time of the oldest known species.  (This represents 7.3 million years before the first appearance of the youngest species.)  The lower bound, 12.11, is based on Bapst's cal-3 calculation given the average origination, extinction & sampling rates, with 12.11 being the time where the log-probability of divergence is 8 less than the most-probable divergence time.  We establish three move commands to adjust this in each MCMC generation, with smaller adjustments being more common than large adjustments.

At this point, we begin setting up the actual tree.  The variable

```
	fbd_dist = dnFBDRP(origin=origin_time, lambda=speciation_rate, mu=extinction_rate, psi=psi,
rho=rho, taxa=taxa);
```
takes all of the information about diversification & sampling to generate a distribution of phylogenies. This then is used to generate a tree, tau:
```
	tau ~ dnConstrainedTopology(fbd_dist,constraints=constraints);
```
where **constraints** is defined in the prior line as the ingroup.

At this point, we add a series of adjustments to phylogeny to the **moves** vector. Here, the weights reflect the number of taxa: the more taxa we have, the more adjustments to phylogeny we should consider to explore comparable proportions of tree space. Two of these moves (mvFNPR and mvNNI) adjust the cladistic topology of the tree with nearest-neighbor exchanges, pruning & grafting, etc. (Note that other tree-changing moves can be used only if they work for *time*-trees.) The other moves alter branch durations (called "branch lengths" but not the same as the expected amount of change along each branch), including collapsing some branches into sampled ancestors.

The script establishes several deterministic variables after this. num_samp_anc gives the number of observed taxa treated as ancestors. Note that **divergence_dates** and **origin_dates** are different. **divergence_dates** for observed taxa is the first-appearance date; for hypothesized ancestors, it is the date at which it diverged into 2+ daughter lineages. For sampled taxa, **divergence_dates** gives the first-appearance date and **origin_dates** gives the time when the lineage leading directly to a sampled species diverged from its closest relatives. For unsampled ancestors, **divergence_dates** gives the time that the ancestor diverged into 2+ daughter taxa and **origin_dates** gives the time that it diverged from the rest of the clade. In both cases, **branch_lengths** gives the duration between **origin_dates** & **divergence_dates**. Finally, **summed_gaps** gives the total amount of "unsampled time" over which character change might have occurred. All of these are included in the output.

The final important variable assignment here is **pruned_tau**; this separates out four species for which we have no character coding. Note that the RevBayes_Setup.r modules will add this only if there are uncoded taxa.

The main script then executes the source code for establishing models of character evolution from the script "Accio_Parameters_for_Analysis_Partitioned_by_States_and_Ordering_and_Class.Rev." This is a more complicated but more flexible script than the one for establishing the tree parameters because it allows for multiple models of character evolution. This example is the simplest possible because all three character partitions are assigned to the same rate group and we are using a strict clock. There are three distinct parts to this script:

1. Establishing per-branch rates of change;
2. Establishing among-character rates of variation;
3. Clamping appropriate data sets and models together for the overall analyses.

Part one sets up the per-branch model of change for each rate-partition. Because **ttl_rate_partitions**==1, this is done only once. Because **clock_model**==“strict” and **big_bang**==F, the first part ultimately executes the script, “Imperio_Strict_Clock_Branch_Rates.Rev.” That script establishes a single variable **branch_rates** from an exponential distribution and adds three types of moves to the **moves** vector (again, varying in the probability of major moves) that are used in each generation of the MCMC search. A deterministic variable, **mean_rt**, is set equal to **branch_rates** here. Although this is redundant in a strict clock analysis, **mean_rt** will be informative under different relaxed clock or rate-shift models and is used so that the same variables can be used in multiple types of analyses.

Part two sets up gamma or lognormal rate variation for all of the different rate partitions. In this case, there is only one (**ttl_rate_partitions**==1), but a for loop is used so that this script can be used for analyses with 2+ rate partitions. This establishes a variable **alpha** from an exponential distribution. As with the prior variables established from exponentials, three types of moves are added to the **moves** vector. **alpha** is not used on its own, but instead to adjust a deterministic variable, **partition_char_rate_var[1]**. If the user opts for a gamma distribution (**among_char_var**=="gamma"), then **partition_char_rate_var[1]** is a gamma distribution in which **alpha** is used for both the $\alpha$ and $\beta$ parameter. Because the mean of a gamma distribution is $\alpha/\beta$, the mean of this distribution will be 1.0; because the variance is $\alpha/\beta^2$, the differences between the maximum and minimum rate decreases as **alpha** increases. If the user opts for a lognormal distribution (**among_char_var**=="lognormal"), then **partition_char_rate_var[1]** is a lognormal distribution with a geometric mean of 0.0 and log-standard deviation=**alpha**. Thus, differences among rates increases as **alpha** increases here. Also, because the geometric mean is set to zero, it is the median rate rather than the (arithmetic) mean rate that is 1.0 when the lognormal is used.

The third part goes through each of the datasets to combine the character evolution models with individual data partitions in order to create a vector **phyMorpho** that links each character partition with the appropriate character evolution models. Here, there are three data partitions separated by numbers of states. The first step is to assign each data partition to the vector **crumplehorn_snorkaxe**. The 2nd step is to create an appropriate Q-matrix (**Q**) for each data partition; because all of the data partitions are unordered, here each **Q[nds]** is a Jukes-Cantor model with k=**partition_states[nds]** for all **nds**=1…3 data partitions

The third step is to assign the character-variation and branch-rate models to deterministic variables **char_rate_var[nds]** and **ind_branch_rates[nds]**. Because there is only one rate-partition here, **char_rate_var[nds]** is the same for all **nds**=1…3. Because we are using a strict clock, **ind_branch_rates[nds]** is a scalar rather than a vector and also the same for all **nds**=1…3.

The final step is to create **phyMorpho[nds]** for each data partition. This uses all of the variables defined in this script plus the tree **tau** created in the prior script and **coding_bias** as the RevBayes_Setup.r script identified them.