CONSEL program user's guide (V0.1i)

Hidetoshi Shimodaira

Department of Mathematical and Computing Sciences
Tokyo Institute of Technology
2-12-1 Ookayama, Meguroku, Tokyo 152-8552, Japan
shimo@is.titech.ac.jp
http://www.is.titech.ac.jp/~shimo/

1 Introduction

CONSEL is a program package for assessing the confidence in selection. The main program `consel` comes with the utility programs `makermt`, `catpv`, `treeass`, `seqmt`, etc. All the programs are written in C language. The confidence is expressed in terms of various "p-values" of statistical testings such as the bootstrap probability, the multiple comparisons tests, and the approximately unbiased tests. The program is designed primarily for phylogeny analysis, but can be used for any other selection problems. The core program of the approximately unbiased tests is based on the multi-scale bootstrap technique, and it is written for general purpose of the problem of regions.

The programs are explained in Section 2. For quick summary, just look at the sample sessions given in Section 3.

The source code and the binary of CONSEL for UNIX and DOS are available from Hidetoshi Shimodaira at Tokyo Institute of Technology. Please contact him by email at shimo@is.titech.ac.jp. Any related information will be found at the web site www.is.titech.ac.jp/~shimo/.

2 Programs

CONSEL consists of small programs written in C language; `seqmt`, `makermt`, `consel`, `catpv`, `catci`, `treeass`, `catass`, `makerep`, `catmt`, `catrep`, `randrep`. The following sections describe these programs.

2.1 raw-data converter: `seqmt`

Let M be the number of items to be compared for selection, and N be the sample size of the data. Let $X_{i,n}$, $i = 1, \ldots, M$, $n = 1, \ldots, N$, be elements of $M \times N$

data matrix X. We are interested in finding which of the row sums $Y_i = \sum_{n=1}^{N} X_{i,n}$, $i = 1, \ldots, M$, has the largest value in the population, not in the observation. In other words, we would like to assess the possibility for each item i that the expected value of Y_i, denoted μ_i, is the largest among the candidates. We consider the item with the largest μ_i as the best item among the candidates, whereas the item with the largest Y_i might have got that value by chance. Since we do not know the values of μ_i's, our inference will be based on the p-values of the statistical hypothesis testings calculated from the data matrix X.

In the case of phylogeny analysis, M is the number of tree topologies and N is the length of the aligned molecular sequences, i.e., the number of sites. $X_{i,n}$ is the log-likelihood of tree-i at site-n. The matrix X is produced by several phylogeny program packages, but they may have different file formats.

In CONSEL, the matrix X is called `mt` file, and the file name ends with extension `mt` after the dot. It is a simple text format starting with two integer numbers indicating M and N. Then $M \times N$ floating point numbers follow in the order $X_{1,1}, \ldots, X_{1,N}, X_{2,1}, \ldots, X_{2,N}, \ldots, X_{M,1}, \ldots, X_{M,N}$, in which the column index n runs fast.

The program `seqmt` in CONSEL converts the formats of phylogeny packages Molphy, PAML, PAUP*, and TREE-PUZZLE to that of CONSEL.

- Molphy produces `lls` file for the site-wise log-likelihoods of trees. It is converted to `mt` file by `seqmt`. For example, `foo.lls` is converted to `foo.mt` as follows.

  ```
  seqmt --molphy foo
  ```

- PAML produces `lnf` file for the log-likelihoods of site-patterns of trees. For example, `foo.lnf` is converted to `foo.mt` as follows.

  ```
  seqmt --paml foo
  ```

  Note that `lnf` file was called `lfh` file in the older versions of PAML.

- PAUP produces a text file for the site-wise log-likelihoods of trees; the file may look like below.

  ```
  Tree    -lnL
  1       2345.033

  Single-site ln likelihoods for tree 1
  1       -2.3434
  ```

```
2       -1.5679
3       -2.3223
...
```

Otherwise the later version may output

```
Tree    -lnL    Site    -lnL
1       2345.033
                1       2.3434
                2       1.5679
                3       2.3223
...
```

or

```
Tree    -lnL    Site    -lnL
                1       2.3434
                2       1.5679
                3       2.3223
...
1       2345.033
...
```

Let us call it `foo.txt` here. It is converted to `foo.mt` by the following command.

```
seqmt --paup foo
```

- TREE-PUZZLE produces `sitelh` file for the log-likelihoods of site-patterns of trees. For example, `foo.sitelh` is converted to `foo.mt` as follows.

  ```
  seqmt --puzzle foo
  ```

2.2 makermt

CONSEL calculates the p-values of the confidence in selection from the multi-scale bootstrap replicates of X. The replicates of the row sums are stored in `rmt` file, which has the file extension `rmt`. The program `makermt` generates the replicates from `mt` file. For example,

```
makermt foo
```

generates `foo.rmt` from `foo.mt`. If you want to have a different name for the output, then

```
makermt foo goo
```

generates `goo.rmt` from `foo.mt`; i.e., the second argument determines the name of the output file.

Not only the `mt` file, but also the outputs of the phylogenetic packages are read by `makermt`. Thus we can skip the use of `seqmt` for the file conversion, and directly feed the matrix to `makermt`. The same options as `seqmt` should be given to specify which file type to be read. For example,

```
makermt --paup foo
```

reads `foo.txt` of the paup file to generate `foo.rmt`. The options `--molphy`, `--paml`, and `--puzzle` are also used.

By default, `makermt` generates 10 sets of bootstrap replicates; each set consists of 10000 replicates of the row sums. These sets of replicates have different "scale" parameters. Let $r_1, \ldots, r_K$ be the scales for the K sets of replicates, and $B_1, \ldots, B_K$ be the numbers of replicates corresponding to them. The default values are

$$r_1 = 0.5, r_2 = 0.6, r_3 = 0.7, r_4 = 0.8, r_5 = 0.9, r_6 = 1.0, r_7 = 1.1,$$

$$r_8 = 1.2, r_9 = 1.3, r_{10} = 1.4; B_1 = \cdots = B_{10} = 10000; K = 10.$$

In the k-th set of replicates, $N_k = r_k N$ sites are randomly chosen from $1, \ldots, N$ with replacement to calculate the row sums. In other words, each replicate of Y_i is written as

$$Y_i^* = (N/N_k) \sum_{j=1}^{N_k} X_{i,n_j^*}$$

where $n_1^*, \ldots, n_{N_k}^*$ are randomly chosen from $1, \ldots, N$ with replacement. The factor N/N_k makes Y_i^* comparable to the original row sum Y_i. Although $N_k = N$ in the ordinary non-parametric bootstrap resamplings, we allow the sample size N_k to differ from N in the scaled bootstrap resamplings.

The default value can be changed by `pa` file; the option `-p NAME` specifies `NAME.pa` as the parameter file. For example, let `short.pa` be a text file with the following content.

```
11
0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2 1.3 1.4 1.5
11
1000 1000 1000 1000 1000 10000 1000 1000 1000 1000 1000
```

This specifies

$$r_1 = 0.5, r_2 = 0.6, \ldots, r_{11} = 1.5;$$

$$B_1 = \cdots = B_5 = B_7 = \cdots = B_{11} = 1000, B_6 = 10000; K = 11.$$

Then

```
makermt -s 234 -p short foo fooshort
```

generates `fooshort.rmt` with 10 sets of 1000 replicates and one set of 10000 replicates; the total number of replicates is $\sum_{k=1}^{K} B_k = 20000$.

The option `-b VAL` multiply the number of replicates by `VAL`. For example, `-b 10` is good for getting the final result with small sampling error.

The option `-f` changes the default values to

$$r_1 = 1; \quad B_1 = 10000; \quad K = 1.$$

This may be useful when the multiscale bootstrap is not needed, or when the rescaling approximation is used in `consel`.

The option `-s VAL` specifies the random seed to integer `VAL` ≥ 0. By default, `VAL` $= 0$, and the radom seed is taken from the system clock.

2.3 consel

Once we get a `rmt` file, the main program `consel` calculates various p-values for each item to assess the possibility that the item has the largest row sum in the population. For example,

```
consel foo
```

calculates the p-values from `foo.rmt`, and stores them in `foo.pv`. The `pv` file contains p-values as well as other auxiliary information such as the values of the test statistics. The content of `foo.pv` can be seen by

```
catpv foo
```

as explained later.

Not only the p-values, but also the confidence intervals of the test statistics are automatically calculated by `consel` and stored in `ci` file. It is seen by

```
catci foo
```

where `foo.ci` contains the confidence intervals.

The second argument to `consel`, if given, specifies the base-name of the output files. For example,

```
consel foo goo
```

produces `goo.pv` and `goo.ci` from `foo.rmt`.

`consel` will produce two other files with extensions `.rep` and `.cnt` by specifying the option `-r` and `-c` respectively. For example,

```
consel -r -c foo goo
```

produces `goo.rep` and `goo.cnt` as well as `goo.pv` and `goo.ci`. The `rep` file contains the replicates of the test statistics, and the `cnt` file contains the counts as to how many times each item has been chosen for the largest Y_i in the replicates. As we will discuss later, `consel` is able to calculate the approximately unbiased p-values from these files. For example,

```
consel -R goo hoo
```

produces `hoo.pv` from `goo.rep`, and

```
consel -C goo hoo
```

produces `hoo.pv` from `goo.cnt`.

Now maximum likelihood method is used for interanl curve fitting of the asymptotic theory. The option `--wls` changes the fitting from maximum likelihood method to weighted least squares method. The option `--mle` does nothing, but assures that maximum likelihood method is used.

The option `-f` changes the resampling algorithm to the rescaling approximation. This uses only the replicates with $r_k = 1$ in `rmt` file to generate replicates with all r_k values. The option `-f` should be used for `makermt` as well. The rescaling approximation may give practically equivalent results as those without approximation, but the standard error becomes larger if the same B_k is used. Note that the standard error calculation of `consel` does not take account of the approximation, so the output of `catpv` with `-e` option shows underestimates of the error.

2.4 catpv

The p-values calculated by `consel` are stored in the `pv` file, and its contents are shown by the command `catpv`. For example,

```
catpv foo
```

reads `foo.pv` and shows the contents like below.

```
# reading foo.pv
# rank item    obs      au      np |      bp      kh      sh     wkh     wsh |
#    1    1   -2.7   0.789   0.575 |   0.579   0.639   0.944   0.639   0.948 |
```

```
#    2    3    2.7  0.516  0.318 |  0.312  0.361  0.799  0.361  0.791 |
#    3    2    7.4  0.114  0.037 |  0.036  0.122  0.575  0.122  0.422 |
#    4    5   17.6  0.076  0.014 |  0.013  0.044  0.178  0.044  0.210 |
#    5    6   18.9  0.129  0.032 |  0.035  0.066  0.149  0.066  0.299 |
#    6    7   20.1  0.029  0.005 |  0.005  0.049  0.114  0.019  0.105 |
#    7    4   20.6  0.102  0.015 |  0.017  0.051  0.112  0.051  0.252 |
#    8   15   22.2  0.012  0.001 |  0.001  0.032  0.073  0.009  0.050 |
#    9    8   25.4  0.001  0.000 |  0.000  0.003  0.032  0.003  0.015 |
#   10   14   26.3  0.028  0.002 |  0.003  0.019  0.034  0.019  0.124 |
#   11   13   28.9  0.019  0.000 |  0.000  0.010  0.018  0.010  0.069 |
#   12    9   31.6  0.004  0.000 |  0.000  0.003  0.006  0.003  0.033 |
#   13   11   31.7  0.010  0.000 |  0.000  0.003  0.006  0.003  0.034 |
#   14   10   34.7  0.007  0.000 |  0.000  0.001  0.003  0.001  0.013 |
#   15   12   36.2  0.009  0.000 |  0.000  0.001  0.002  0.001  0.009 |
```

There are 15 rows of the table, and each row corresponds to one of the items compared for selection. The items are sorted in the decreasing order of Y_i; the first column is the order of the item starting from 1 to $M = 15$. The second column shows the index i of the item; we may use the option `-s 1` to sort the lines by the index i. The third column shows the test statistics

$$T_i = \max_{j=1,\ldots,M} Y_j - Y_i$$

for the items with rank ≥ 2. They are equivalent to

$$T_i = \max_{j \neq i} (Y_j - Y_i), \quad i = 1, \ldots, M, \tag{1}$$

except for the item with the largest Y_i. The subscript j runs through $1, \ldots, M$ but i. The latter definition is better to explain the multiplicity of the comparisons, and thus used for the table. Note that Y_i is the largest among the M items when

$$Y_j - Y_i \leq 0$$

for all $j = 1, \ldots, M$ but i. These $M - 1$ comparisons are explicitly given in the definition (1).

The rest of the columns show various p-values for the items. The larger the p-values are, the chance is higher for the item to have the largest expected value of Y_i. These p-values are derived from different ideas, thus the values can be very different to each other. The first two p-values (au, np) are calculated from all the K sets of the scaled bootstrap replicates, and the rest of five p-values (bp, kh, sh, wkh, wsh) are calculated from one set of replicates with $r_k = 1$. The newly added column pp is calculated from the log-likelihood values.

- **au** The p-value of the approximately unbiased test. This is the main result of `consel`, while rest of the p-values in the table are supplementary. It is derived from the theory of the signed distance and the curvature of the boundary as explained later. Let us denote it AU_i for the item i. We may reject the possibility that item i has the largest expected value of Y_i when $\mathrm{AU}_i < 0.05$ at the significance level 0.05. The confidence set of the items then becomes $\{1, 3, 2, 5, 6, 4\}$ among the fifteen items in `foo.pv`.

- **np** This is the bootstrap probability of the selection; i.e., NP_i is the probability that item i has the largest Y_i^* in the non-scaled bootstrap replicates. However, NP_i is calculated through the same theory as AU_i. This utilizes all the replicates of the multi-scale bootstrap.

- **bp** Same as np, but calculated directly from the replicates with $r_k = 1$. BP_i is the frequency that item i has the largest Y_i^* in the B_k replicates of $r_k = 1$. By definition, $\sum_{i=1}^{M} \mathrm{BP}_i = 1$. NP_i should be very close to BP_i unless the theory breaks down.

- **pp** Bayesian posterior probability (PP) calculated by the BIC approximation.

- **kh** The p-value of the Kishino-Hasegawa (KH) test.

- **sh** The p-value of the Shimodaira-Hasegawa (SH) test.

- **wkh** The p-value of the weighted Kishino-Hasegawa (WKH) test.

- **wsh** The p-value of the weighted Shimodaira-Hasegawa (WSH) test.

There are several options to the command `catpv`.

`-v` prints the auxiliary information. pf: p-value of the diagnostic of the asymptotic theory. d: signed distance. c: curvature of the boundary. th: the actual threshold used for au and np.

`-e` prints the standard errors of the p-values.

`-s 1` sort the lines by the item index.

`-s 6` sort the lines by the SH p-value.

`-s 9` sort the lines by the AU p-value.

`-r` outputs the list of rank, order, and item.

`--no_au` suppresses printing au and np.

`--no_sh` suppresses printing bp, kh, sh, wkh, and wsh.

`--no_print` suppresses printing.

`-o NAME` aggregates the specified `pv` files and writes them into `NAME.out`. This option, combined with `-no_print`, is useful for simulations.

For example,

```
catpv -v --no_sh foo
```

prints the following output.

```
# reading foo.pv
# rank item    obs     au     np | |     pf    rss df      d      c     th
#    1    1   -2.7  0.789  0.575 | |  0.886  3.668  8 -0.495  0.307  0.000
#    2    3    2.7  0.516  0.318 | |  0.769  4.891  8  0.217  0.257  0.000
#    3    2    7.4  0.114  0.037 | |  0.254 10.164  8  1.498  0.294  0.000
#    4    5   17.6  0.076  0.014 | |  0.629  6.160  8  1.816  0.381  0.000
#    5    6   18.9  0.129  0.032 | |  0.300  9.522  8  1.492  0.359  0.043
#    6    7   20.1  0.029  0.005 | |  0.871  3.848  8  2.234  0.339  0.000
#    7    4   20.6  0.102  0.015 | |  0.080 14.057  8  1.716  0.444  0.000
#    8   15   22.2  0.012  0.001 | |  0.894  2.902  7  2.730  0.478  0.000
#    9    8   25.4  0.001  0.000 | |  0.791  1.043  3  3.583  0.527  0.000
#   10   14   26.3  0.028  0.002 | |  0.242 10.337  8  2.371  0.464  0.000
#   11   13   28.9  0.019  0.000 | |  0.572  5.730  7  2.755  0.672  0.839
#   12    9   31.6  0.004  0.000 | |  0.890  1.693  5  3.149  0.508  2.097
#   13   11   31.7  0.010  0.000 | |  0.209  3.134  2  3.117  0.776  0.861
#   14   10   34.7  0.007  0.000 | |  0.893  1.669  5  3.087  0.609  6.680
#   15   12   36.2  0.009  0.000 | |  0.501  4.343  5  2.909  0.560  9.082
```

Let us denote θ_i for the entry `th` at the last column of the table. The AU test is calculating the p-value AU_i for the null hypothesis represented as the region

$$H_i : \max_{j \neq i}(\mu_j - \mu_i) \leq \theta_i.$$

For the selection problem, θ_i should be zero, since H_i then corresponds to the hypothesis that μ_i is the largest among $\mu_1, \ldots, \mu_M$. However, `consel` has to choose θ_i larger than zero if the bootstrap probabilities are too small. This is avoided by using larger B_k values. You do not have to worry about it when AU_i is already smaller the significance level, because the actual AU_i with $\theta_i = 0$ must be smaller than the obtained AU_i with $\theta_i > 0$.

The standard error of the p-value is shown by the option `-e`. For example,

```
catpv -e --no_sh foo
```

prints the following output.

```
# reading foo.pv
# rank item    obs     au    (se)     np    (se) | |
#    1    1   -2.7  0.789 (0.007)  0.575 (0.001) | |
#    2    3    2.7  0.516 (0.010)  0.318 (0.002) | |
#    3    2    7.4  0.114 (0.008)  0.037 (0.002) | |
#    4    5   17.6  0.076 (0.009)  0.014 (0.002) | |
#    5    6   18.9  0.129 (0.010)  0.032 (0.002) | |
#    6    7   20.1  0.029 (0.006)  0.005 (0.001) | |
#    7    4   20.6  0.102 (0.010)  0.015 (0.002) | |
#    8   15   22.2  0.012 (0.006)  0.001 (0.001) | |
#    9    8   25.4  0.001 (0.004)  0.000 (0.001) | |
#   10   14   26.3  0.028 (0.008)  0.002 (0.002) | |
#   11   13   28.9  0.019 (0.012)  0.000 (0.003) | |
#   12    9   31.6  0.004 (0.006)  0.000 (0.001) | |
#   13   11   31.7  0.010 (0.026)  0.000 (0.007) | |
#   14   10   34.7  0.007 (0.009)  0.000 (0.002) | |
#   15   12   36.2  0.009 (0.009)  0.000 (0.002) | |
```

The numbers of replicates $B_1, \ldots, B_K$ should be increased when the standard errors shown in `se` are so large that it is unclear whether the p-values are above or below the significance level.

Several `pv` files are printed at the same time by `catpv`. For example,

```
catpv foo1 foo2 foo3
```

prints `foo1.pv`, `foo2.pv`, and `foo3.pv`. To aggregate the results of the simulations, the following command will be useful;

```
catpv foo1 foo2 foo3 --no_print -o foos
```

aggregates the three `pv` files and writes `foos.out` in which the p-values are stored as matrices.

2.5 catci

The confidence intervals of μ_i calculated by `consel` are stored in `ci` file, and its contents are shown by the command `catci`. There are two kinds of confidence intervals calculated in `consel`, one associated with AU_i, and the other associated with NP_i. They are derived by inverting the associated p-values. The command

```
catci foo
```

prints the two kinds of confidence limits for the levels 0.05, 0.1, 0.5, 0.9, and 0.95. To avoid lengthy output, one of the two kinds can be suppressed by giving option `--no_np` or `--no_au` as shown in the following examples.

```
catci --no_np foo
```

reads `foo.ci` and produces the output like below.

```
# read foo.ci
#                 --------------- au ---------------- |
# rank item    obs  0.050  0.100  0.500  0.900  0.950 |
#    1    1   -2.7  -17.3  -14.9   -5.9    3.8    6.5 |
#    2    3    2.7  -12.8  -10.6   -0.3   10.7   13.5 |
#    3    2    7.4   -2.1   -0.5    5.9   12.8   15.0 |
#    4    5   17.6   -1.4    1.0   16.2   30.3   35.1 |
#    5    6   18.9   -4.9   -1.5   17.8   33.9   38.3 |
#    6    7   20.1    1.8    4.6   17.1   34.2   39.1 |
#    7    4   20.6   -3.3   -0.2   18.8   35.7   39.9 |
#    8   15   22.2    4.3    6.8   19.4   36.0   40.5 |
#    9    8   25.4    9.2   11.7   22.8   36.3   40.5 |
#   10   14   26.3    3.4    7.2   23.7   41.0   45.9 |
#   11   13   28.9    5.6    9.6   26.0   43.1   48.2 |
#   12    9   31.6   11.3   14.6   28.9   44.6   48.6 |
#   13   11   31.7   10.0   13.9   29.3   45.2   49.7 |
#   14   10   34.7   13.1   17.8   31.5   47.5   51.7 |
#   15   12   36.2   15.6   19.6   33.5   48.8   52.8 |
```

Similarly,

```
catci --no_au foo
```

reads `foo.ci` and produces the output like below.

```
# read foo.ci
#                  |--------------- np ----------------
# rank item    obs |  0.050  0.100  0.500  0.900  0.950
#    1    1   -2.7 |  -12.2   -9.9   -1.3    8.0   10.8
#    2    3    2.7 |   -7.7   -5.4    3.4   13.0   15.8
#    3    2    7.4 |    0.7    2.6    9.4   17.0   19.4
#    4    5   17.6 |    3.7    6.5   19.3   33.7   38.1
#    5    6   18.9 |    1.8    5.4   20.9   36.9   41.4
#    6    7   20.1 |    6.7    9.7   21.8   36.0   40.5
#    7    4   20.6 |    4.0    7.4   22.6   38.5   43.0
#    8   15   22.2 |    9.5   12.2   23.9   37.9   42.1
#    9    8   25.4 |   13.8   16.5   27.1   39.6   43.5
```

```
#   10   14   26.3 |   10.4   14.1   28.3   43.6   48.1
#   11   13   28.9 |   13.6   17.1   30.8   45.7   49.9
#   12    9   31.6 |   17.6   20.9   33.6   47.4   51.6
#   13   11   31.7 |   17.5   20.7   33.7   47.8   51.9
#   14   10   34.7 |   21.4   24.5   36.7   50.1   54.0
#   15   12   36.2 |   23.2   26.2   38.2   51.4   55.3
```

2.6 treeass

In phylogeny analysis, we are interested in the history of evolution of species represented as a tree with the labeled leaves corresponding to the taxa. Let M be the number of candidate trees for selection, and Y_i be the log-likelihood of the tree-i. Then `consel` gives the p-values for each tree to see which of the candidate trees represents the true history.

However, we are sometimes interested in the edges of the trees rather than the tree itself. Each edge divides the taxa into two groups, and thus describes monophyly of the group of taxa. In other words, a set of taxa, if it does not contain the "outgroup" species, is monophyletic when it corresponds to one of the edges of the true tree. `consel` can give the p-values for each edge using the information produced by the utility program `treeass`.

The candidate trees are represented in the standard parenthesis format, and stored in `tpl` file. It is a text file starting with M, the number of trees. For example, let `mam15.tpl` be the file given below.

```
15
((Homsa,(Phovi,Bosta)),Orycu,(Musmu,Didvi)); t1
(Homsa,Orycu,((Phovi,Bosta),(Musmu,Didvi))); t2
(Homsa,((Phovi,Bosta),Orycu),(Musmu,Didvi)); t3
(Homsa,(Orycu,Musmu),((Phovi,Bosta),Didvi)); t4
((Homsa,(Phovi,Bosta)),(Orycu,Musmu),Didvi); t5
(Homsa,((Phovi,Bosta),(Orycu,Musmu)),Didvi); t6
(Homsa,(((Phovi,Bosta),Orycu),Musmu),Didvi); t7
(((Homsa,(Phovi,Bosta)),Musmu),Orycu,Didvi); t8
(((Homsa,Musmu),(Phovi,Bosta)),Orycu,Didvi); t9
(Homsa,Orycu,(((Phovi,Bosta),Musmu),Didvi)); t10
(Homsa,(((Phovi,Bosta),Musmu),Orycu),Didvi); t11
((Homsa,((Phovi,Bosta),Musmu)),Orycu,Didvi); t12
(Homsa,Orycu,(((Phovi,Bosta),Didvi),Musmu)); t13
((Homsa,Musmu),Orycu,((Phovi,Bosta),Didvi)); t14
((Homsa,Musmu),((Phovi,Bosta),Orycu),Didvi); t15
```

Then,

```
treeass --outgroup 6 mam15 > mam15.log
```

reads `mam15.tpl` and produces `mam15.ass` and `mam15.log`. The sixth species (Didvi) is the outgroup here. The information regarding associations between the edges and the trees is stored in `ass` file, while auxiliary information is found in `mam15.log` here. Let `foo.rmt` be the `rmt` file for the 15 trees. Then

```
consel -a mam15 foo goo
```

reads `mam15.ass` and `foo.rmt` to produce `goo.pv` and `goo.ci`. The option `-a NAME` specifies the file `NAME.ass` for the associations. The p-values for the edges are shown by

```
catpv goo
```

and the results are given below.

```
# reading goo.pv
# rank item    obs     au     np |     bp     kh     sh    wkh    wsh |
#    1    2  -17.6  0.954  0.931 |  0.927  0.956  0.994  0.934  0.991 |
#    2    1   -2.7  0.749  0.589 |  0.592  0.639  0.910  0.639  0.921 |
#    3    4    2.7  0.469  0.325 |  0.318  0.361  0.754  0.361  0.735 |
#    4    3    7.4  0.111  0.037 |  0.036  0.122  0.567  0.122  0.411 |
#    5    5   17.6  0.076  0.061 |  0.065  0.044  0.177  0.066  0.253 |
#    6    7   18.9  0.088  0.037 |  0.040  0.066  0.147  0.066  0.277 |
#    7    6   20.6  0.071  0.018 |  0.019  0.051  0.112  0.051  0.227 |
#    8    9   22.2  0.016  0.003 |  0.004  0.032  0.072  0.019  0.113 |
#    9    8   25.4  0.001  0.000 |  0.000  0.003  0.032  0.003  0.031 |
#   10   10   31.7  0.002  0.000 |  0.000  0.003  0.006  0.003  0.032 |
```

There are 10 possible edges for the 15 trees of `mam15.tpl` as seen in `goo.pv`. Let m be the number of edges, while M be the number of trees. In this example, $m = 10$ and $M = 15$. The edges $1, \ldots, m$ of the table above are specified in `mam15.log` given below.

```
# $Id: program.tex,v 1.5 2005/09/26 02:04:04 shimo Exp $
# reading mam15.tpl
# 15 trees read
# 15 trees, 6 leaves
# edges total: 18
# reverse edges: 0
# 10 base-edges, 7 common-edges, 1 root-edge

# trees: 15
15
```

```
((1,(2,3)),4,(5,6)); 1
(1,4,((2,3),(5,6))); 2
(1,((2,3),4),(5,6)); 3
(1,(4,5),((2,3),6)); 4
((1,(2,3)),(4,5),6); 5
(1,((2,3),(4,5)),6); 6
(1,(((2,3),4),5),6); 7
(((1,(2,3)),5),4,6); 8
(((1,5),(2,3)),4,6); 9
(1,4,(((2,3),5),6)); 10
(1,(((2,3),5),4),6); 11
((1,((2,3),5)),4,6); 12
(1,4,(((2,3),6),5)); 13
((1,5),4,((2,3),6)); 14
((1,5),((2,3),4),6); 15

# leaves: 6
6
  1 Homsa
  2 Phovi
  3 Bosta
  4 Orycu
  5 Musmu
  6 Didvi

# base edges: 10
10 6
    123456
  1 +++---
  2 ++++--
  3 +--+--
  4 -+++--
  5 ---++-
  6 +--++-
  7 -++++-
  8 +++-+-
  9 +---+-
 10 -++-+-

# common edges: 7
7 6
    123456
```

```
 11 +-----
 12 -++---
 13 -+----
 14 --+---
 15 ---+--
 16 ----+-
 17 +++++-

# tree->edge
15
  1   2  1 2
  2   2  2 3
  3   2  2 4
  4   2  5 6
  5   2  1 5
  6   2  5 7
  7   2  4 7
  8   2  1 8
  9   2  8 9
 10   2  3 10
 11   2  7 10
 12   2  8 10
 13   2  3 6
 14   2  6 9
 15   2  4 9

# edge->tree
10
  1   3  1 5 8
  2   3  1 2 3
  3   3  2 10 13
  4   3  3 7 15
  5   3  4 5 6
  6   3  4 13 14
  7   3  6 7 11
  8   3  8 9 12
  9   3  9 14 15
 10   3  10 11 12

# writing tmp/mam15.ass
# exit normally
```

The 15 trees at the beginning of `mam15.log` are the same as those of `mam15.tpl` but the labels of the leaves are replaced by the numbers. There are 6 leaves numbered $1, \ldots, 6$ instead of the labels Hosma, ..., Didvi. Then ten "base edges" followed by the seven "common edges"are shown. Each row corresponds to one of the seventeen edges, and each column corresponds to one of the six leaves. For each row, the leaves are divided into two groups as indicated by `+` and `-`. We only deal with the "unrooted" trees, so that the two signs are interchangeable. However, the option `--outgroup 6` specifies the sixth species as the outgroup so that it is always denoted by `-`. For example, the first row `+++---` represents the clade of {1, 2, 3} or equivalently {Homsa, Phovi, Bosta}.

We are interested in the base edges, which are included in some of the candidate trees but not all of them. On the other hand, we are not interested in the common edges because they are included in all of the $M = 15$ trees and their p-values are always unity.

There are $m = 10$ base edges for this example. The candidate trees are represented by combinations of these base edges, but not all of the combinations correspond to the trees. For example, base edges 1=`+++---` and 2=`----++` are associated with the tree 1=((1,(2,3)),4,(5,6)). This association is shown in the first row of the entry `# tree->edge`, where the associations from the $M = 15$ trees to the $m = 10$ base edges are shown. In each row, the first number corresponds to the tree index, the second number (2 for all rows) is the number of base edges included in the tree, and the following numbers are the indexes of the edges. For convenience, let us denote B_i the set of base edges for the tree-i; $B_1 = \{1, 2\}$, $B_2 = \{2, 3\}$, ..., $B_{15} = \{4, 9\}$. Only these 15 combinations out of possible 45 combinations of two base edges are allowed to form trees.

The reverse associations are shown at the entry `# edge->tree`. For example, base edge-1 is included in the trees 1, 5, and 8. There are ten rows corresponding to the ten base edges. For $e = 1, \ldots, m$, let A_e be the set of trees including base edge-e; $A_1 = \{1, 5, 8\}$, ..., $A_{10} = \{10, 11, 12\}$.

The test statistics shown in the third column of `goo.pv` are defined by

$$T_e = \min_{i \in A_e} \max_{j \notin A_e} (Y_j - Y_i), \quad e = 1, \ldots, m. \tag{2}$$

The tree-i has the largest Y_i value for some $i \in A_e$ thus edge-e is selected when $T_e \leq 0$. We define NP_e as the bootstrap probability of $T_e \leq 0$. In other words, NP_e is the bootstrap probability of the edge-e. The null hypothesis here is that the edge-e is included in the tree of the largest μ_i value. AU_e and the other p-values are also calculated for each edge.

2.7 catass

`catass:` join `ass` files.

The `ass` file generated by `treeass` specifies the associations between the edges and the trees by the sets A_e, $e = 1, \ldots, m$. Any types of associations, however, can be specified by `ass` files. Let `foo1.ass`, `foo2.ass`, and `foo3.ass` be `ass` files of the numbers of associations m_1, m_2, and m_3, respectively. Then

```
catass foo1 foo2 foo3 foos
```

produces `foos.ass` of the number of associations $m_1 + m_2 + m_3$. If only one file name is given, it is taken as the output file. For example,

```
catass -m 15 foo
```

produces `foo.ass` of the identity associations; $A_e = \{e\}$, $e = 1, \ldots, m$, where `-m` option specifies m. This `ass` file represents the trees instead of the edges.

The output associations are restricted by giving `-X file` option. Let `hyp.vt` be the file of

```
2
1 5
```

which represents the vector $(1, 5)$ implying the first and fifth associations. Then

```
catass -X hyp mam15 myhyp
```

produces `myhyp.ass` which consists of two associations; $A_1 = \{1, 5, 8\}$ and $A_2 = \{4, 5, 6\}$. Note that the option `-X` must be captial X.

Set operations can be performed by `catass`. The intersection, the union, and the complement are specified by the options `-i`, `-u`, and `-n`, respectively.

2.8 makerep

`makerep`: generate `rep` file from `mt` file.

This is quite similar to `makermt`, but generates `rep` file instead of `rmt` file. This is useful when the number of trees is large and you are interested in particular hypotheses on phylogeny. The hypotheses are specified by the `ass` file. For example,

```
makerep -a myhyp --paml mam15.lnf myhyp15
```

reads `myhyp.ass` and `mam15.lnf` to produce `myhyp15.rep`. The p-value of AU test is calculated by

```
consel -R myhyp15
```

and the p-value is printed by

```
catpv myhyp15
```

2.9 catmt

`catmt`: join `mt` files.

Let `foo1.mt`, `foo2.mt`, and `foo3.mt` are matrices of $M \times N_1$, $M \times N_2$, and $M \times N_3$, respectively. Then

```
catmt foo1 foo2 foo3 foos
```

produces `foos.mt` of size $M \times (N_1 + N_2 + N_3)$. This is useful to combine the likelihoods of several genes. The number of trees M can be reduced by prescreening using the Kishino-Hasegawa (KH) test. For example,

```
catmt --kht 0.01 foo1 foo2 foo3 foos
```

combines the three `mt` files, then performs the KH test at the level 0.01. The result is stored in `foos.mt`, and the item id's are stored in `foos.vt`. This is useful when M is very large so that `makermt` may become very slow unless the prescreening is done.

2.10 catrep

`catrep`: join and select `rep` and `rmt` files.

Let `foo-i.rep` be the `rep` file for M trees with $K^{(i)}$ scales

$$r_1^{(i)}, \ldots, r_{K^{(i)}}^{(i)}; \quad B_1^{(i)}, \ldots, B_{K^{(i)}}^{(i)}.$$

Then

```
catrep foo-1 foo-2 foo-3 foos
```

combines the three `rep` files to get `foos.rep`. M must be the same for all the `rep` files, but the scale parameters can vary. If there are same r_k values in the `rep` files, then they are aggregated and the corresponding B_k's are summed up in `foos.rep`. Instead of `rep` files, the `rmt` files are also combined by giving `-m` option. For example,

```
catrep -m foo-1 foo-2 foos
```

combines `foo-1.rmt` and `foo-2.rmt` to get `foos.rmt`.

Only the replicates of specified scales can be included in the output by giving `-p NAME` option to specify `NAME.pa` file. For example,

```
3
0.5 1.0 1.4
3
10000 10000 10000
```

is the content of `k3.pa` file, which specifies $r_1 = 0.5$, $r_2 = 1.0$, and $r_3 = 1.4$. Then

```
catrep -p k3 -m foo-1 foo-2 foos
```

combines the two `rmt` files, and picks up only the three scales to produce `foos.rmt`. The B_k values in `k3.pa` are ignored.

2.11 randrep

`randrep`: random generation of `rep` and `rmt` files for simulations.

By default, `randrep` generates the square root of the random variables distributed as the non-central χ^2 distribution;

$$Z^{*b} = \|Y^{*b}\|, \quad b = 1, \ldots, B_k$$

where

$$Y^{*1}, \ldots, Y^{*B_k} \sim N_M(Y, I/r_k); \quad Y \sim N_M(\mu, \lambda^2 I). \tag{3}$$

$\lambda = 0$ thus $Y = \mu$ unless specified by `-f VAL` option. The noncentrality $\|Y\|^2$ is specified by giving μ_1, where $\mu = (\mu_1, 0, \ldots, 0)$ is M-vector. The scales r_k, $k = 1, \ldots, K$ as well as the numbers of replicates $B_1, \ldots, B_K$ are specified by the `-p` option, otherwise the same default value as `makermt` is used.

The non-centrality μ_1^2 and the degrees of freedom M are specified by the `vt` file. For example, `sim1.vt`:

```
3
7 7 7
3
3 4 5
```

specifies three distributions with the non-centrality 7^2 for all, and the degrees of freedom 3, 4, 5. Then,

```
randrep -s 333 -r 10 sim1 tmp/foo_
```

generates `tmp/foo_00.rep`, `tmp/foo_01.rep`, ..., `tmp/foo_09.rep`. The random seed is specified by `-s 333`, and the number of repetitions is specified by `-r 10`. Each of the `rep` files is processed by `consel`;

```
consel --th 5.0 --no_sort -R tmp/foo_01
```

writes `tmp/foo_00.pv` and `tmp/foo_00.ci`. The p-values are calculated for the hypothesis of the region $\|\mu\| \leq \theta$, where $\theta = 5$.

The `rmt` files are also generated by `randrep` if `-m` option is specified. M and μ is specified by the `vt` file. For example, `mu1.vt`:

```
10
0  -1 -2 -3 -4 -5 -6 -7 -8 -9
```

specifies $M = 10$ and $\mu = (0, -1, -2, -3, -4, -5, -6, -7, -8, -9)$. The normal vector Y^*'s are generated by

```
randrep -m -s 345 -r 10 -f 1 mu1 tmp/goo_
```

which outputs `tmp/goo_00.rmt`, `tmp/goo_01.rmt`, ..., `tmp/goo_09.rmt`. The `-s` and `-r` options are the same as before. The difference is the `-f 1` option which specifies $\lambda = 1$.

3 Sample sessions

3.1 Tree analysis

```
# make mam15.rmt from the output of PAML (mam15.lnf)
# this takes several minutes
makermt --paml mam15
# calculate the p-values from mam15.rmt
# this takes only a minute
consel mam15
# print mam15.pv
catpv mam15
```

3.2 Edge analysis

```
# make mam.ass and mam15.log from mam15.tpl
treeass mam15 > mam15.log
# calculate the p-values from mam15.rmt
consel -a mam15 mam15 mam15e
# print mam15e
catpv mam15e
```

3.3 Large data

```
# join foo1.mt, foo2.mt, and foo3.mt and make goo.mt
catmt foo1 foo2 foo3 goo
# select trees from goo.mt with pv>0.01 and make hoo.mt
catmt --kht 0.01 goo hoo
# you can do the above two lines by a single command
catmt --kht 0.01 foo1 foo2 foo3 hoo
```

```
# note: the id's of the selected trees are stored in hoo.vt
# select trees from tree105.tpl using hoo.vt
treeass -p -v hoo tree105 treeout > treeout.log
```

3.4 Only sh-tests

Use `-f option` for `makermt` followed by `consel`.

```
makermt -f --paml mam15 mam15sh
consel mam15sh
catpv mam15sh
```

3.5 Test of particular hypotheses

Let `myhyp.ass` specify the particular hypotheses of your interest. For example, the `ass` file of the hypotheses of monophyly may be easily prepared by using `treeass` and `catass`. Then,

```
makerep -a myhyp --paup manytree.txt fewhyp
```

reads `myhyp.ass` and `manytree.txt` to generate `fewhyp.rep`. The p-value is calculated and printed by

```
consel -R fewhyp
catpv fewhyp
```

3.6 Parallel computing

The calculation of `fewhyp.rep` becomes much faster when a parallel computer is available. Let `r05.pa` be

```
1 0.5 1 10000
```

which represents $K = 1$, $r_1 = 0.5$, $B_1 = 10000$. Similarly, we prepare `r06.pa` to `r14.pa` for $r_2 = 0.6, \ldots, r_{10} = 1.4$. Then,

```
makerep -s 151 -p r05 -a myhyp --paup manytree.txt fewhyp05 &
makerep -s 161 -p r06 -a myhyp --paup manytree.txt fewhyp06 &
...
makerep -s 241 -p r14 -a myhyp --paup manytree.txt fewhyp14 &
```

may generate `fewhyp05.rep`, ..., `fewhyp14.rep`. They are combined by

```
catrep fewhyp??.rep fewhyp
```

which produces `fewhyp.rep`.

4 File formats

Binary files: `rmt`, `rep`.
Text files: `mt`, `vt`, `pv`, `ci`, `pa`, `ass`, `cnt`, `tpl`, `txt`, `lnf`, `lls`.

5 Command reference

6 Theory of the approximately unbiased test

"singed distance" and "curvature"

- Efron, B. (1985) "Bootstrap confidence intervals for a class of parametric problems," *Biometrika,* **72**, 45-58.
- Efron, B., Halloran, E. and Holmes, S. (1996) "Bootstrap confidence levels for phylogenetic trees," *Proc. Natl. Acad. Sci. USA,* **93**, 13429-13434.
- Efron, B. and Tibshirani, R. (1998) "The problem of regions," *Ann. Statist.,* **26**, 1687-1718.
- Shimodaira, H. (2000) "Another calculation of the p-value for the problem of regions using the scaled bootstrap resamplings," Technical Report No. 2000-35, Stanford University.
- Shimodaira, H. (2002) "An approximately unbiased test of phylogenetic tree selection," *Systematic Biology,* **51**, 492–508.
- Shimodaira, H. and Hasegawa, M. (2001) "CONSEL: for assessing the confidence of phylogenetic tree selection," *Bioinformatics,* **17**, 1246–1247.

Please refer to Shimodaira and Hasegawa (2001) for the program CONSEL, and refer to Shimodaira (2002) for the theory of the approximately unbiased test with application to the phylogenetic inference.