The caper package: methods benchmarks.

David Orme

November 29, 2013

This vignette presents benchmark testing of the implementation of various methods in the package against other existing implementations. The output from other implementations have been stored in standard .rda data files in the package 'data' directory. Details of the creation of the benchmark data sets and of suitably formatted input files for other program implementations, along with the original outputs and logs of those programs, are held in the 'benchmark' directory in the package Subversion repository on http://r-forge.r-project.org.

The main benchmark dataset ('benchTestInputs.rda') contains the following objects:

`benchTreeDicho` A 200 tip tree grown under a pure-birth, constant-rates model using the `growTree()` function.

`benchTreePoly` A version of the tree in which six polytomies have been created by collapsing the shortest internal branches.

`benchData` A data frame of tip data for the trees containing the following columns, either evolved using `growTree()` or modified from the evolved variables:

- `node` Identifies which tip on the tree each row of data relates to.
- `contResp, contExp1, contExp2` Three co-varying continuous variables evolved under Brownian motion along the tree.
- `contExp1NoVar` A version of `contExp1` which has identical values for five more distal polytomies. This is present to test the behaviour of algorithms at polytomies which have no variation in a variable.
- `contRespNA, contExp1NA, contExp2NA` As above but with a small proportion (~5%) of missing data.
- `biFact, triFact` A binary and ternary categorical variable evolving under a rate matrix across the tree.
- `biFactNA, triFactNA` As above, but with a small proportion (~5%) of missing data.
- `sppRichTaxa` Integer species richness values, with clade sizes taken from a broken stick distribution of 5000 species among the 200 extant tips, but with richness values distributed arbitrarily across tips.
- `sppRichTips` A column of ones representing each tip as a single species.

`testTree` A dichotomous ultrametric tree of 64 tips

`testData` A complete dataset of two variables for each of the 64 tips

```
> library(caper)
> library(xtable)
> data(benchTestInputs)
> test <- comparative.data(testTree, testData, tips, vcv=TRUE, vcv.dim=3)
> benchDicho <- comparative.data(benchTreeDicho, benchData, node, na.omit=FALSE)
> benchPoly <- comparative.data(benchTreePoly, benchData, node, na.omit=FALSE)
> print(test)
```

```
Comparative dataset of 64 taxa:
Phylogeny: testTree
   64 tips, 63 internal nodes
   chr [1:64] "BH" "BD" "BM" "BT" "AJ" "BO" "CA" "AC" "CB" ...
VCV matrix present:
   VCV.array [1:64, 1:64, 1:11] 0.464 0.177 0.177 0.177 0.177 ...
Data: testData
   $ V1: num [1:64] -7.12 1.35 1.78 -3.41 -2.51 ...
   $ V2: num [1:64] -4.655 4.152 3.384 -2.985 -0.824 ...
```

> print(benchDicho)

```
Comparative dataset of 200 taxa:
Phylogeny: benchTreeDicho
   200 tips, 199 internal nodes
   chr [1:200] "165" "166" "181" "182" "100" "183" "184" "39" ...
Data: benchData
   $ contResp     : num [1:200] -5.81 -5.2 -3.74 -3.94 -5.03 ...
   $ contExp1     : num [1:200] 8.43 8.54 6.84 6.57 6.98 ...
   $ contExp2     : num [1:200] 5.16 5.51 8.63 8.38 8.75 ...
   $ contExp1NoVar: num [1:200] 8.43 8.54 6.84 6.57 6.98 ...
   $ contRespNA   : num [1:200] -5.81 -5.2 -3.74 -3.94 -5.03 ...
   $ contExp1NA   : num [1:200] 8.43 NA 6.84 6.57 6.98 ...
   $ contExp2NA   : num [1:200] 5.16 5.51 8.63 8.38 8.75 ...
   $ binFact      : Factor w/ 2 levels "A","B": 2 1 2 2 1 2 2 1 1 2 ...
   $ triFact      : Ord.factor w/ 3 levels "A"<"B"<"C": 3 3 3 3 3 3 3 3 3 3 ...
   $ binFactNA    : Factor w/ 2 levels "A","B": 2 1 2 2 1 2 2 1 1 2 ...
   $ triFactNA    : Ord.factor w/ 3 levels "A"<"B"<"C": 3 3 3 3 3 3 3 3 3 3 ...
   $ sppRichTaxa  : int [1:200] 59 51 6 7 1 8 19 25 43 6 ...
   $ sppRichTips  : int [1:200] 1 1 1 1 1 1 1 1 1 1 ...
```

> print(benchPoly)

```
Comparative dataset of 200 taxa:
Phylogeny: benchTreePoly
   200 tips, 193 internal nodes
   chr [1:200] "165" "166" "181" "182" "100" "183" "184" "39" ...
Data: benchData
   $ contResp     : num [1:200] -5.81 -5.2 -3.74 -3.94 -5.03 ...
   $ contExp1     : num [1:200] 8.43 8.54 6.84 6.57 6.98 ...
   $ contExp2     : num [1:200] 5.16 5.51 8.63 8.38 8.75 ...
   $ contExp1NoVar: num [1:200] 8.43 8.54 6.84 6.57 6.98 ...
   $ contRespNA   : num [1:200] -5.81 -5.2 -3.74 -3.94 -5.03 ...
   $ contExp1NA   : num [1:200] 8.43 NA 6.84 6.57 6.98 ...
   $ contExp2NA   : num [1:200] 5.16 5.51 8.63 8.38 8.75 ...
   $ binFact      : Factor w/ 2 levels "A","B": 2 1 2 2 1 2 2 1 1 2 ...
   $ triFact      : Ord.factor w/ 3 levels "A"<"B"<"C": 3 3 3 3 3 3 3 3 3 3 ...
   $ binFactNA    : Factor w/ 2 levels "A","B": 2 1 2 2 1 2 2 1 1 2 ...
   $ triFactNA    : Ord.factor w/ 3 levels "A"<"B"<"C": 3 3 3 3 3 3 3 3 3 3 ...
   $ sppRichTaxa  : int [1:200] 59 51 6 7 1 8 19 25 43 6 ...
   $ sppRichTips  : int [1:200] 1 1 1 1 1 1 1 1 1 1 ...
```

1 Benchmarking pgls.

The programs Continuous and BayesTraits both implement similar phylogenetic generalised least squares models, but Continuous is no longer readily available[1] and has been superseded by BayesTraits. The tests below compare the output of `pgls` with the same models fitted using BayesTraits. In order to avoid convergence problems, the lower limit of the parameter bounds for `pgls` have been raised slightly. Model names indicate the optimised parameters (lower case = fixed at 1, upper case = optimised). The test models fit ML estimates of all combinations of branch length transformations and also fit three fixed point sets of parameters: all zero (equivalent to a standard linear model), all set to 0.5 and all set to 1.

```
> bnds <- list(lambda=c(0.01,1), kappa=c(0.01,3), delta=c(0.01,3))
> bnds <- list(lambda=c(0,1), kappa=c(0,3), delta=c(0,3))
> nul <- pgls(V1 ~ V2, test, bounds=bnds, lambda=0, kappa=0, delta=0)
> fix <- pgls(V1 ~ V2, test, bounds=bnds, lambda=0.5, kappa=0.5, delta=0.5)
> kld <- pgls(V1 ~ V2, test, lambda=1, delta=1, kappa=1)
> Kld <- pgls(V1 ~ V2, test, lambda=1, delta=1, kappa="ML")
> kLd <- pgls(V1 ~ V2, test, lambda="ML", delta=1, kappa=1)
> klD <- pgls(V1 ~ V2, test, lambda=1, delta="ML", kappa=1)
> kLD <- pgls(V1 ~ V2, test, lambda="ML", delta="ML", kappa=1)
> KlD <- pgls(V1 ~ V2, test, lambda=1, delta="ML", kappa="ML")
> KLd <- pgls(V1 ~ V2, test, lambda="ML", delta=1, kappa="ML")
> KLD <- pgls(V1 ~ V2, test, lambda="ML", delta="ML", kappa="ML")
>
```

The following code constructs a summary table for comparison with a similar summary table from BayesTraits (Table 1):

```
> pglsMods <- list(nul=nul, fix=fix, kld = kld, Kld = Kld, kLd = kLd,
+                       klD = klD, kLD = kLD, KlD = KlD, KLd = KLd, KLD = KLD)
> pglslogLik <- sapply(pglsMods, logLik)
> pglsCoefs <- t(sapply(pglsMods, coef))
> pglsSigma <- sapply(pglsMods, '[[', 'RMS')
> pglsR.sq <- sapply(pglsMods, function(X) summary(X)$r.squared)
> pglsParam <- t(sapply(pglsMods, '[[', 'param'))
> pglsModTab <-  data.frame(mods=names(pglsMods), pglslogLik, pglsCoefs, pglsSigma, pglsR.sq, pgls
> data(benchBayesTraitsOutputs)
```

2 Benchmarking crunch() and brunch().

These benchmarks test the implementation of independent contrast calculations (Felsenstein, 1985) using the `crunch()` and `brunch()` algorithms against the implementations in the Mac Classic program CAIC v2.6.9 (Purvis and Rambaut, 1995), running in Mac OS 9.2.2 (emulated in Mac OS 10.4.10).

2.1 The crunch algorithm

Five analyses were performed in CAIC to benchmark the following situations. The numbers in the object names refer to the column numbers in the `BenchData` data frame.

CrDi213 A dichotomous tree with complete data in three continuous variables.

[1] Discontinued?

Model	logLik	Intercept	Slope	Variance	r^2	κ	λ	δ
a) Models fitted using `pgls`								
nul	-164.17	0.66	0.49	10.22	0.25	0.00	0.00	0.00
fix	-145.88	0.55	0.64	5.72	0.38	0.50	0.50	0.50
kld	-135.07	0.12	0.82	7.24	0.49	1.00	1.00	1.00
Kld	-134.60	0.21	0.80	5.70	0.48	0.84	1.00	1.00
kLd	-134.28	0.18	0.79	6.28	0.48	1.00	0.99	1.00
klD	-133.98	0.18	0.76	1.60	0.46	1.00	1.00	2.18
kLD	-133.71	0.21	0.76	2.04	0.46	1.00	0.99	1.89
KlD	-133.81	0.25	0.75	1.42	0.47	0.90	1.00	2.05
KLd	-134.19	0.13	0.80	7.40	0.48	1.19	0.98	1.00
KLD	-133.70	0.19	0.76	2.35	0.46	1.05	0.99	1.84
b) Models fitted using BayesTraits								
nul	-164.17	0.66	0.49	9.90	0.25	0.00	0.00	0.00
fix	-148.58	0.55	0.60	4.57	0.35	0.50	0.50	0.50
kld	-134.86	0.12	0.82	7.01	0.49	1.00	1.00	1.00
Kld	-134.37	0.21	0.80	5.49	0.48	0.84	1.00	1.00
kLd	-134.08	0.18	0.79	6.09	0.48	1.00	0.99	1.00
klD	-133.91	0.17	0.76	2.15	0.47	1.00	1.00	1.97
kLD	-133.61	0.20	0.76	2.63	0.47	1.00	0.99	1.70
KlD	-133.71	0.24	0.75	1.94	0.47	0.89	1.00	1.83
KLd	-134.01	0.13	0.80	7.06	0.48	1.17	0.98	1.00
KLD	-133.60	0.18	0.77	2.96	0.47	1.05	0.98	1.66

Table 1: Model outputs from `pgls` and BayesTraits.

CrDi657 A dichotomous tree with incomplete data in three continuous variables.

CrPl213 A polytomous tree with complete data in three continuous variables.

CrPl413 A polytomous tree with complete data in three continuous variables, but with no variation in the reference variable at some polytomies.

CrPl657 A polytomous tree with incomplete data in three continuous variables.

These calculations are reproduced below using the `crunch()` function. Note that the default internal branch length used in calculations at a polytomy (`polytomy.brlen`) needs to be changed from the 0, which is the default in `crunch()`, to 1, which was the default in CAIC. The `caic.table()` function is used to extract a contrast table from the `crunch()` output, including CAIC style node labels.

```
> crunch.CrDi657 <- crunch(contRespNA ~ contExp1NA + contExp2NA,  data=benchDicho, polytomy.brlen=
> crunch.CrDi213 <- crunch(contResp ~ contExp1 + contExp2, data=benchDicho, polytomy.brlen=1)
> crunch.CrPl413 <- crunch(contResp ~ contExp1NoVar + contExp2, data=benchPoly, polytomy.brlen=1)
> crunch.CrPl213 <- crunch(contResp ~ contExp1 + contExp2, data=benchPoly, polytomy.brlen=1)
> crunch.CrPl657 <- crunch(contRespNA ~ contExp1NA + contExp2NA, data=benchPoly, polytomy.brlen=1)
```

The outputs of contrast calculations in CAIC are saved as the data frames `CAIC.CrDi213`, `CAIC.CrDi657`, `CAIC.CrPl213`, `CAIC.CrPl413` and `CAIC.CrPl657` in the data file `benchCrunchOutputs.rda`. Each of the data frames contains the standard CAIC contrast table consisting of: the CAIC code for the node, the contrast in each variable, the standard deviation of the contrast, the height of the node, the number of subtaxa descending from the node; and the nodal values of the variables. The CAIC codes can now be used to merge the datasets from the two implementations in order to compare the calculated values. The contrasts in the response variable are plotted in Fig. **??**, with data from CAIC shown in black and overplotting of data from `crunch()` in red. The correlations between these contrasts calculated with each implementation are shown in Table 2.

```
> crunch.CrDi657.tab <- caic.table(crunch.CrDi657, CAIC.codes=TRUE)
> crunch.CrDi213.tab <- caic.table(crunch.CrDi213, CAIC.codes=TRUE)
> crunch.CrPl413.tab <- caic.table(crunch.CrPl413, CAIC.codes=TRUE)
> crunch.CrPl213.tab <- caic.table(crunch.CrPl213, CAIC.codes=TRUE)
> crunch.CrPl657.tab <- caic.table(crunch.CrPl657, CAIC.codes=TRUE)
> data(benchCrunchOutputs)
> crunch.CrDi213.tab <- merge(crunch.CrDi213.tab,  CAIC.CrDi213, by.x="CAIC.code", by.y="Code", su
> crunch.CrDi657.tab <- merge(crunch.CrDi657.tab,  CAIC.CrDi657, by.x="CAIC.code", by.y="Code", su
> crunch.CrPl213.tab <- merge(crunch.CrPl213.tab,  CAIC.CrPl213, by.x="CAIC.code", by.y="Code", su
> crunch.CrPl413.tab <- merge(crunch.CrPl413.tab,  CAIC.CrPl413, by.x="CAIC.code", by.y="Code", su
> crunch.CrPl657.tab <- merge(crunch.CrPl657.tab,  CAIC.CrPl657, by.x="CAIC.code", by.y="Code", su
```

analysis	RespCor	Exp1Cor	Exp2Cor
CrDi213	1.0000	1.0000	1.0000
CrDi657	1.0000	1.0000	1.0000
CrPl213	1.0000	1.0000	1.0000
CrPl413	0.9442	1.0000	0.9670
CrPl657	0.9908	0.9862	0.9998

Table 2: Correlations between values of crunch and CAIC contrasts.

2.2 The `brunch` algorithm

Eight analyses were performed in CAIC using the 'brunch' algorithm to benchmark the following tests:

BrDi813 and BrPl813 A binary factor as the primary variable with two continuous variables on both the dichotomous and polytomous tree.

BrDi913 and BrPl913 An ordered ternary factor as the primary variable with two continuous variables on both the dichotomous and polytomous tree.

BrDi1057 and BrPl1057 A binary factor and two continuous variables, all with missing data, on both the dichotomous and polytomous tree.

BrDi1157 and BrPl1157 A ordered ternary factor and two continuous variables, all with missing data, on both the dichotomous and polytomous tree.

These analyses are duplicated below using the `brunch()` function. Note that `brunch()` does not calculate contrasts at polytomies and the tests using the polytomous tree are to check that the algorithms are drawing the same contrasts from the data. The `caic.table()` function is again used to extract a contrast table from the `brunch()` output.

```
> brunch.BrDi813  <- brunch(contResp ~ binFact + contExp2, data=benchDicho)
> brunch.BrDi913  <- brunch(contResp ~ triFact + contExp2, data=benchDicho)
> brunch.BrDi1057 <- brunch(contRespNA ~ binFactNA + contExp2NA, data=benchDicho)
> brunch.BrDi1157 <- brunch(contRespNA ~ triFactNA + contExp2NA, data=benchDicho)
> brunch.BrPl813  <- brunch(contResp ~ binFact + contExp2, data=benchPoly)
> brunch.BrPl913  <- brunch(contResp ~ triFact + contExp2, data=benchPoly)
> brunch.BrPl1057 <- brunch(contRespNA ~ binFactNA + contExp2NA, data=benchPoly)
> brunch.BrPl1157 <- brunch(contRespNA ~ triFactNA + contExp2NA, data=benchPoly)
>
```

The CAIC codes can again now be used to merge the output of these analyses with the outputs from the original CAIC to compare the calculated values. The contrasts from each test are overplotted in Fig. 2 and Fig. 3 and the correlation between the contrasts for each variable is shown in Table 3.

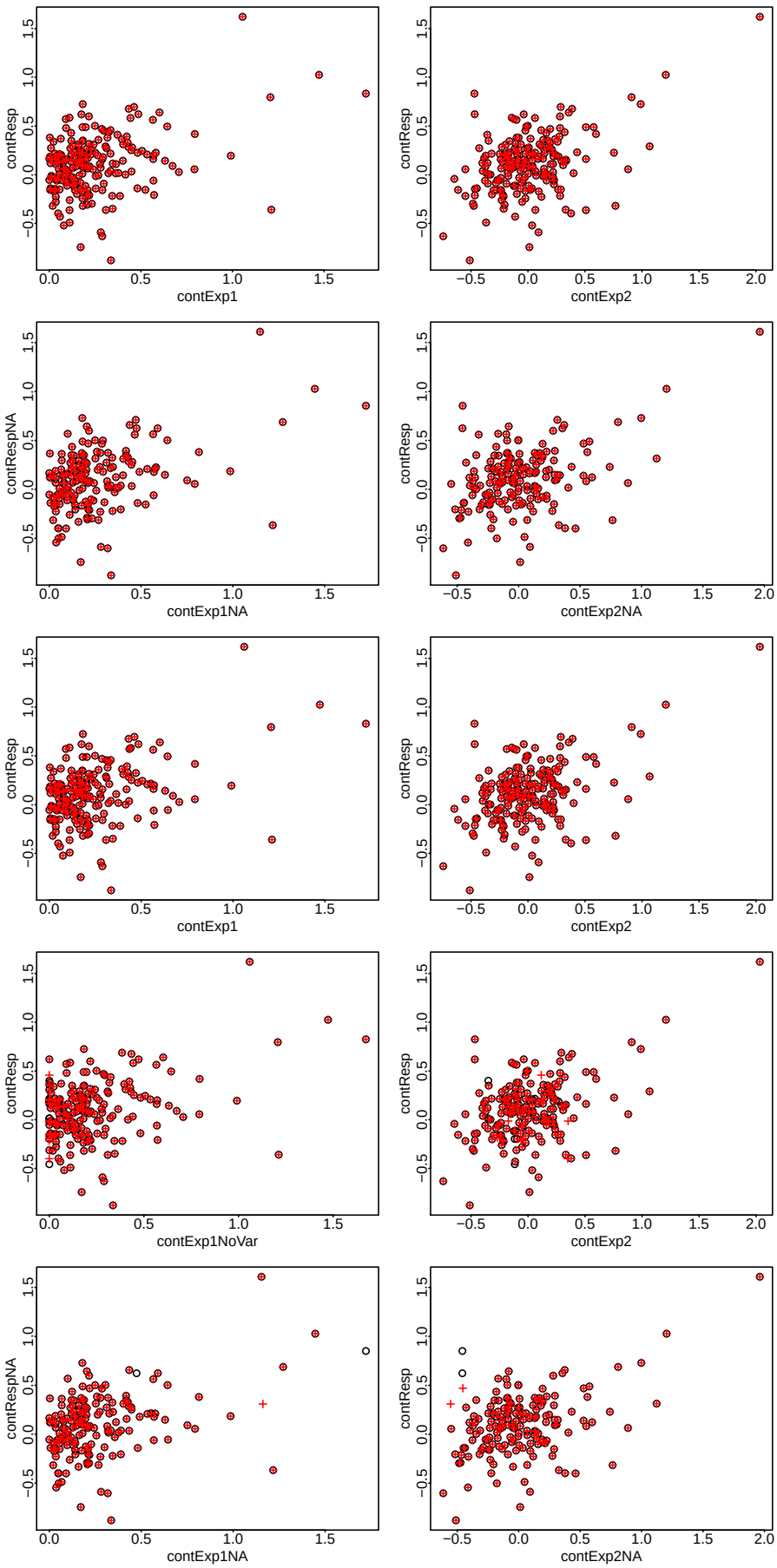

Figure 1: Overplotting of results from CAIC (black) and `crunch()` (red) analyses.

```
> brunch.BrDi813.tab  <- caic.table(brunch.BrDi813, CAIC.codes=TRUE)
> brunch.BrDi913.tab  <- caic.table(brunch.BrDi913, CAIC.codes=TRUE)
> brunch.BrDi1057.tab <- caic.table(brunch.BrDi1057, CAIC.codes=TRUE)
> brunch.BrDi1157.tab <- caic.table(brunch.BrDi1157, CAIC.codes=TRUE)
> brunch.BrPl813.tab  <- caic.table(brunch.BrPl813, CAIC.codes=TRUE)
> brunch.BrPl913.tab  <- caic.table(brunch.BrPl913, CAIC.codes=TRUE)
> brunch.BrPl1057.tab <- caic.table(brunch.BrPl1057, CAIC.codes=TRUE)
> brunch.BrPl1157.tab <- caic.table(brunch.BrPl1157, CAIC.codes=TRUE)
> data(benchBrunchOutputs)
> brunch.BrDi813.tab  <- merge(brunch.BrDi813.tab , CAIC.BrDi813 , by.x="CAIC.code", by.y="Code",
> brunch.BrDi913.tab  <- merge(brunch.BrDi913.tab , CAIC.BrDi913 , by.x="CAIC.code", by.y="Code",
> brunch.BrDi1057.tab <- merge(brunch.BrDi1057.tab, CAIC.BrDi1057, by.x="CAIC.code", by.y="Code",
> brunch.BrDi1157.tab <- merge(brunch.BrDi1157.tab, CAIC.BrDi1157, by.x="CAIC.code", by.y="Code",
> brunch.BrPl813.tab  <- merge(brunch.BrPl813.tab , CAIC.BrPl813 , by.x="CAIC.code", by.y="Code",
> brunch.BrPl913.tab  <- merge(brunch.BrPl913.tab , CAIC.BrPl913 , by.x="CAIC.code", by.y="Code",
> brunch.BrPl1057.tab <- merge(brunch.BrPl1057.tab, CAIC.BrPl1057, by.x="CAIC.code", by.y="Code",
> brunch.BrPl1157.tab <- merge(brunch.BrPl1157.tab, CAIC.BrPl1157, by.x="CAIC.code", by.y="Code",
```

analysis	RespCor	Exp1Cor	Exp2Cor
BrDi813	1.00000		1.00000
BrPl813	1.00000		1.00000
BrDi1057	0.78625	1.00000	0.64672
BrPl1057	0.93942	1.00000	0.99186
BrDi913	0.99999		1.00000
BrPl913	1.00000		1.00000
BrDi1157	0.72425	1.00000	0.46912
BrPl1157	0.93522	1.00000	0.97660

Table 3: Range in the differences between brunch and CAIC contrasts.

3 Benchmarking `macrocaic()`.

The function `macrocaic()` is a re-implementation of the program 'MacroCAIC' (Agapow and Isaac, 2002), which calculates standard 'crunch' contrasts (Felsenstein, 1985) for the explanatory variables but species richness contrasts for the response variable. The program calculates two species richness contrast types which are included in the output file. These are 'PDI', the proportion dominance index, and 'RRD', the relative rate difference (Agapow and Isaac, 2002). The benchmark tests included 8 sets of analyses using all combinations of the following:

Di/Poly The benchmark tree used: either dichotomous or polytomous.

Spp/Tax Whether the tips are treated as single species or as taxa containing a known number of species.

23/67 The completeness of the explanatory data used, where 2 and 3 are two complete continuous variables and the variables in 6 and 7 have missing data.

These analyses are repeated below using the function `macrocaic()`.

```
> DichSpp23.RRD <- macrocaic(sppRichTips ~ contExp1  + contExp2, macroMethod = "RRD", data=benchDi
> PolySpp23.RRD <- macrocaic(sppRichTips ~ contExp1  + contExp2, macroMethod = "RRD", data=benchPo
> DichTax23.RRD <- macrocaic(sppRichTaxa ~ contExp1  + contExp2, macroMethod = "RRD", data=benchDi
> PolyTax23.RRD <- macrocaic(sppRichTaxa ~ contExp1  + contExp2, macroMethod = "RRD", data=benchPo
> DichSpp67.RRD <- macrocaic(sppRichTips ~ contExp1NA + contExp2NA, macroMethod = "RRD", data=benc
> PolySpp67.RRD <- macrocaic(sppRichTips ~ contExp1NA + contExp2NA, macroMethod = "RRD", data=benc
> DichTax67.RRD <- macrocaic(sppRichTaxa ~ contExp1NA + contExp2NA, macroMethod = "RRD", data=benc
```

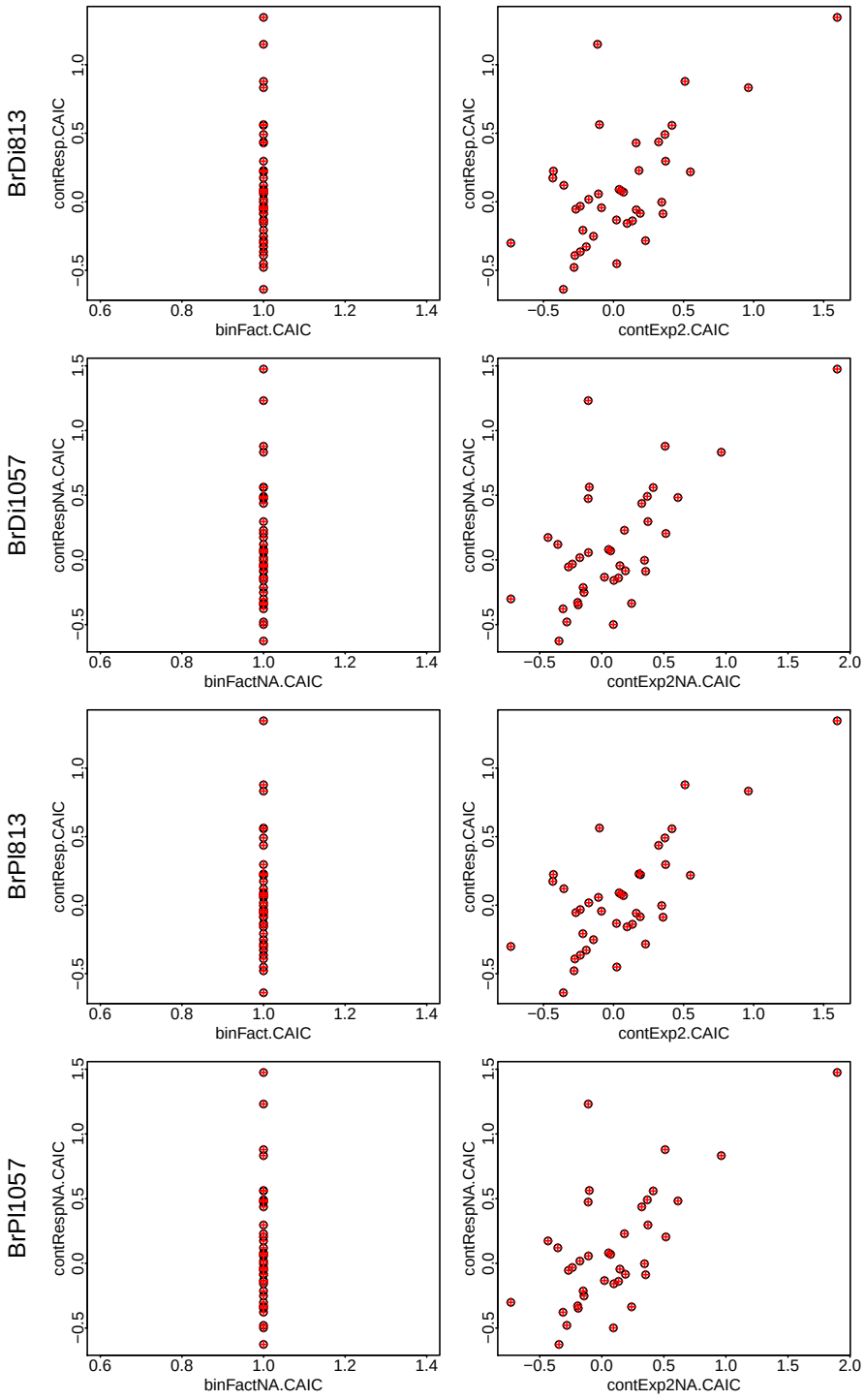

Figure 2: Overplotting of results from CAIC (black) and `brunch()` (red) analyses using a two level factor.

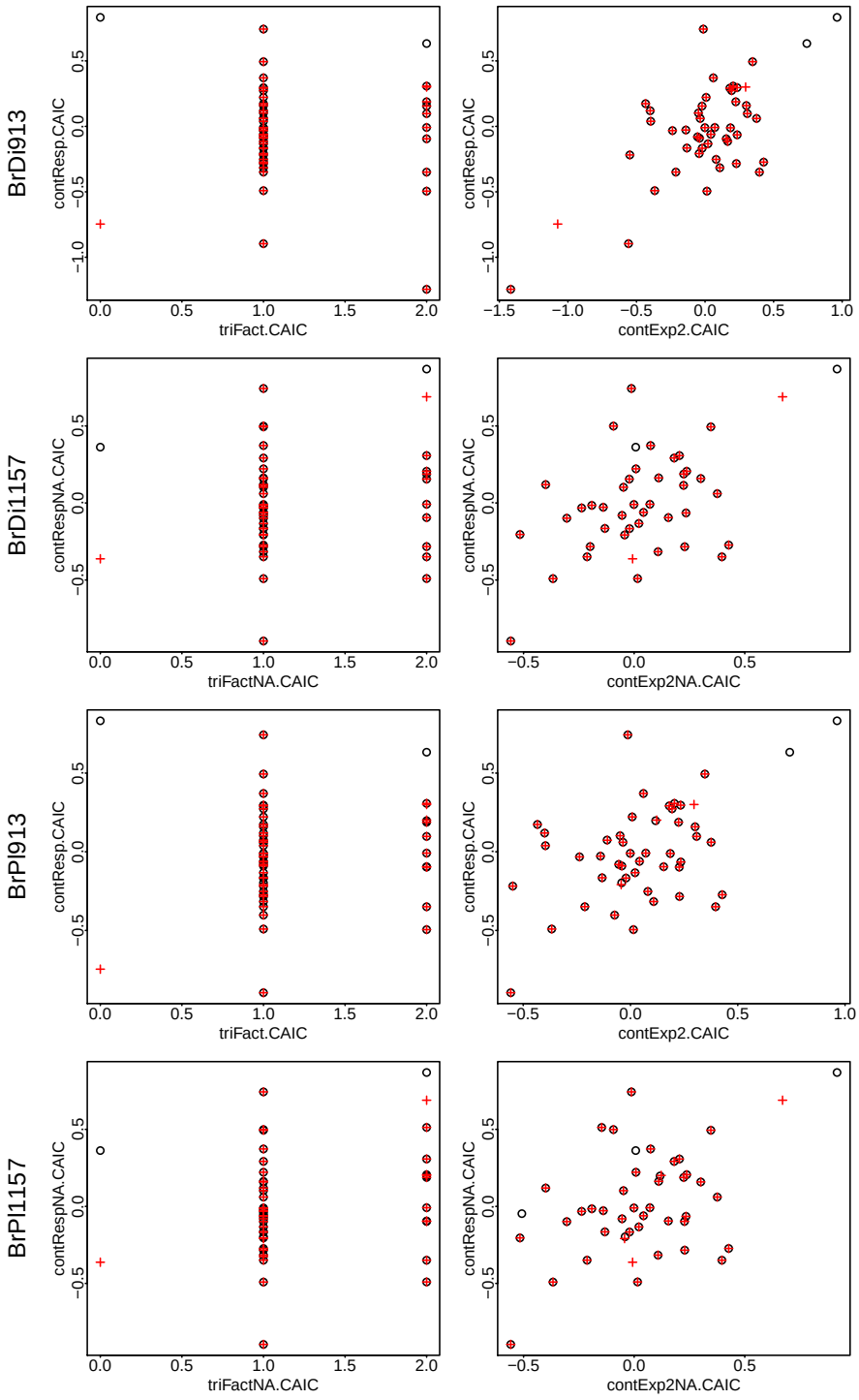

Figure 3: Overplotting of results from CAIC (black) and `brunch()` (red) analyses using a three level factor.

```
> PolyTax67.RRD <- macrocaic(sppRichTaxa ~ contExp1NA + contExp2NA, macroMethod = "RRD", data=benc
> DichSpp23.PDI <- macrocaic(sppRichTips ~ contExp1  + contExp2, macroMethod = "PDI", data=benchDi
> PolySpp23.PDI <- macrocaic(sppRichTips ~ contExp1  + contExp2, macroMethod = "PDI", data=benchPo
> DichTax23.PDI <- macrocaic(sppRichTaxa ~ contExp1  + contExp2, macroMethod = "PDI", data=benchDi
> PolyTax23.PDI <- macrocaic(sppRichTaxa ~ contExp1  + contExp2, macroMethod = "PDI", data=benchPo
> DichSpp67.PDI <- macrocaic(sppRichTips ~ contExp1NA + contExp2NA, macroMethod = "PDI", data=benc
> PolySpp67.PDI <- macrocaic(sppRichTips ~ contExp1NA + contExp2NA, macroMethod = "PDI", data=benc
> DichTax67.PDI <- macrocaic(sppRichTaxa ~ contExp1NA + contExp2NA, macroMethod = "PDI", data=benc
> PolyTax67.PDI <- macrocaic(sppRichTaxa ~ contExp1NA + contExp2NA, macroMethod = "PDI", data=benc
>
```

The code below merges the results for the two different methods and then merges these with the output from MacroCAIC. The resulting contrasts are overplotted on the outputs from 'MacroCAIC' for both RRD and PDI using complete and incomplete data (Figs. 4,5, 6,7). Again, the correlations between the contrasts calculated at each node by each implementation are shown in Table 4.

```
> macro.DichSpp23 <- merge(caic.table(DichSpp23.RRD, CAIC.codes=TRUE), caic.table(DichSpp23.PDI, C
> macro.PolySpp23 <- merge(caic.table(PolySpp23.RRD, CAIC.codes=TRUE), caic.table(PolySpp23.PDI, C
> macro.DichTax23 <- merge(caic.table(DichTax23.RRD, CAIC.codes=TRUE), caic.table(DichTax23.PDI, C
> macro.PolyTax23 <- merge(caic.table(PolyTax23.RRD, CAIC.codes=TRUE), caic.table(PolyTax23.PDI, C
> macro.DichSpp67 <- merge(caic.table(DichSpp67.RRD, CAIC.codes=TRUE), caic.table(DichSpp67.PDI, C
> macro.PolySpp67 <- merge(caic.table(PolySpp67.RRD, CAIC.codes=TRUE), caic.table(PolySpp67.PDI, C
> macro.DichTax67 <- merge(caic.table(DichTax67.RRD, CAIC.codes=TRUE), caic.table(DichTax67.PDI, C
> macro.PolyTax67 <- merge(caic.table(PolyTax67.RRD, CAIC.codes=TRUE), caic.table(PolyTax67.PDI, C
> data(benchMacroCAICOutputs)
> macro.DichSpp23 <- merge(macro.DichSpp23, MacroCAIC.DiSpp23, by.x="CAIC.code", by.y="Code")
> macro.PolySpp23 <- merge(macro.PolySpp23, MacroCAIC.PolySpp23, by.x="CAIC.code", by.y="Code")
> macro.DichTax23 <- merge(macro.DichTax23, MacroCAIC.DiTax23, by.x="CAIC.code", by.y="Code")
> macro.PolyTax23 <- merge(macro.PolyTax23, MacroCAIC.PolyTax23, by.x="CAIC.code", by.y="Code")
> macro.DichSpp67 <- merge(macro.DichSpp67, MacroCAIC.DiSpp67, by.x="CAIC.code", by.y="Code")
> macro.PolySpp67 <- merge(macro.PolySpp67, MacroCAIC.PolySpp67, by.x="CAIC.code", by.y="Code")
> macro.DichTax67 <- merge(macro.DichTax67, MacroCAIC.DiTax67, by.x="CAIC.code", by.y="Code")
> macro.PolyTax67 <- merge(macro.PolyTax67, MacroCAIC.PolyTax67, by.x="CAIC.code", by.y="Code")
>
```

analysis	RichCor.RRD	Exp1Cor.RRD	Exp2Cor.RRD	RichCor.PDI	Exp1Cor.PDI	Exp2Cor.PDI
DichSpp23	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
PolySpp23	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
DichTax23	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
PolyTax23	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
DichSpp67	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
PolySpp67	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
DichTax67	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
PolyTax67	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000

Table 4: Correlations between brunch and CAIC contrasts.

4 Benchmarking `fusco.test()`.

4.1 Testing against the original FUSCO implementation.

This runs tests against Giuseppe Fusco's implementation of the phylogenetic imbalance statistic I (Fusco and Cronk, 1995). The original package consists of two DOS programs 'IMB_CALC' and 'NULL_MDL' which calculate the imbalance of phylogeny and then the expected distribution of imbalance values on a equal rates Markov tree, given the size of the tree. The program contains the

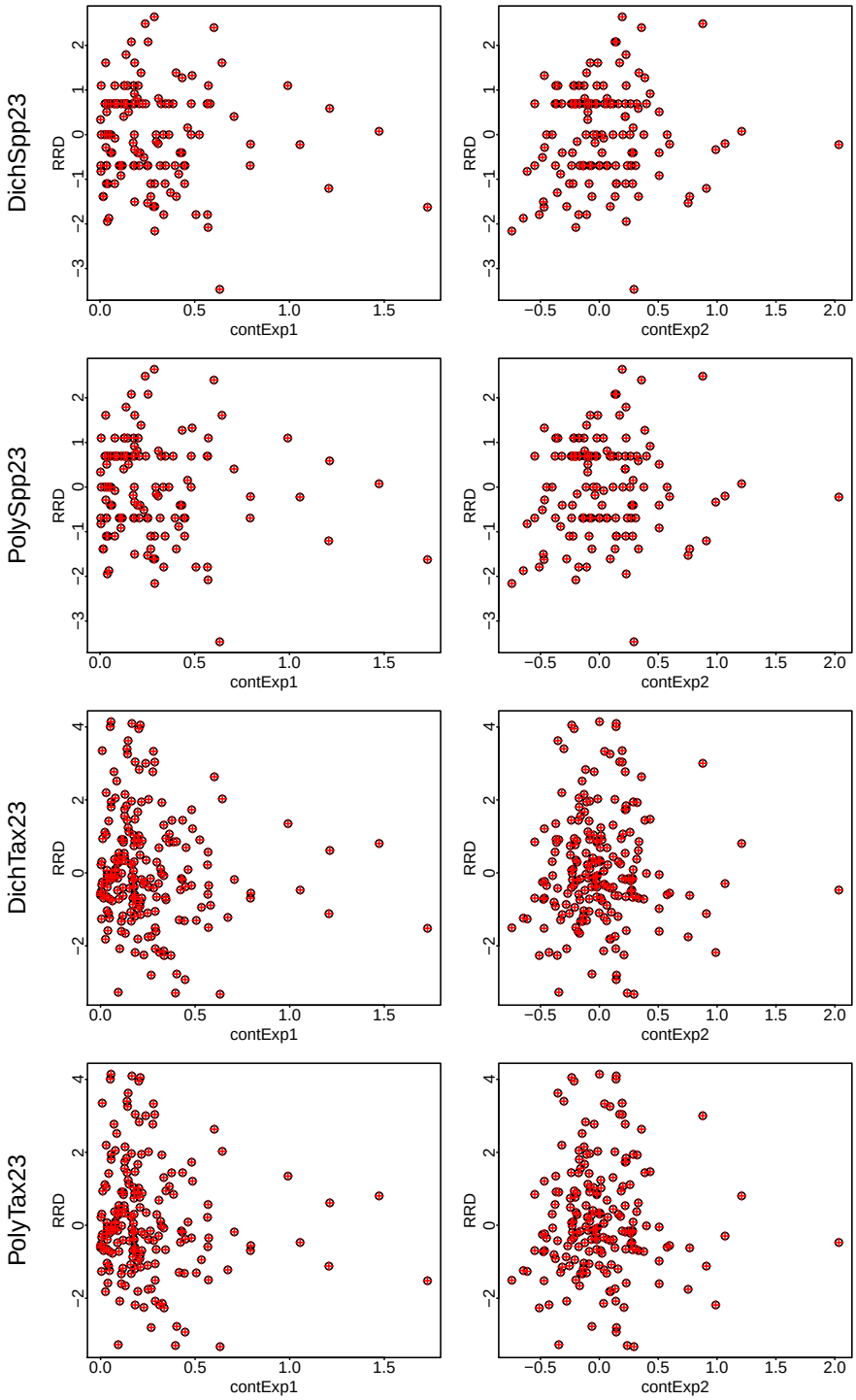

Figure 4: Overplotting of results from MacroCAIC (black) and `macrocaic()` (red) analyses using RRD and complete data.

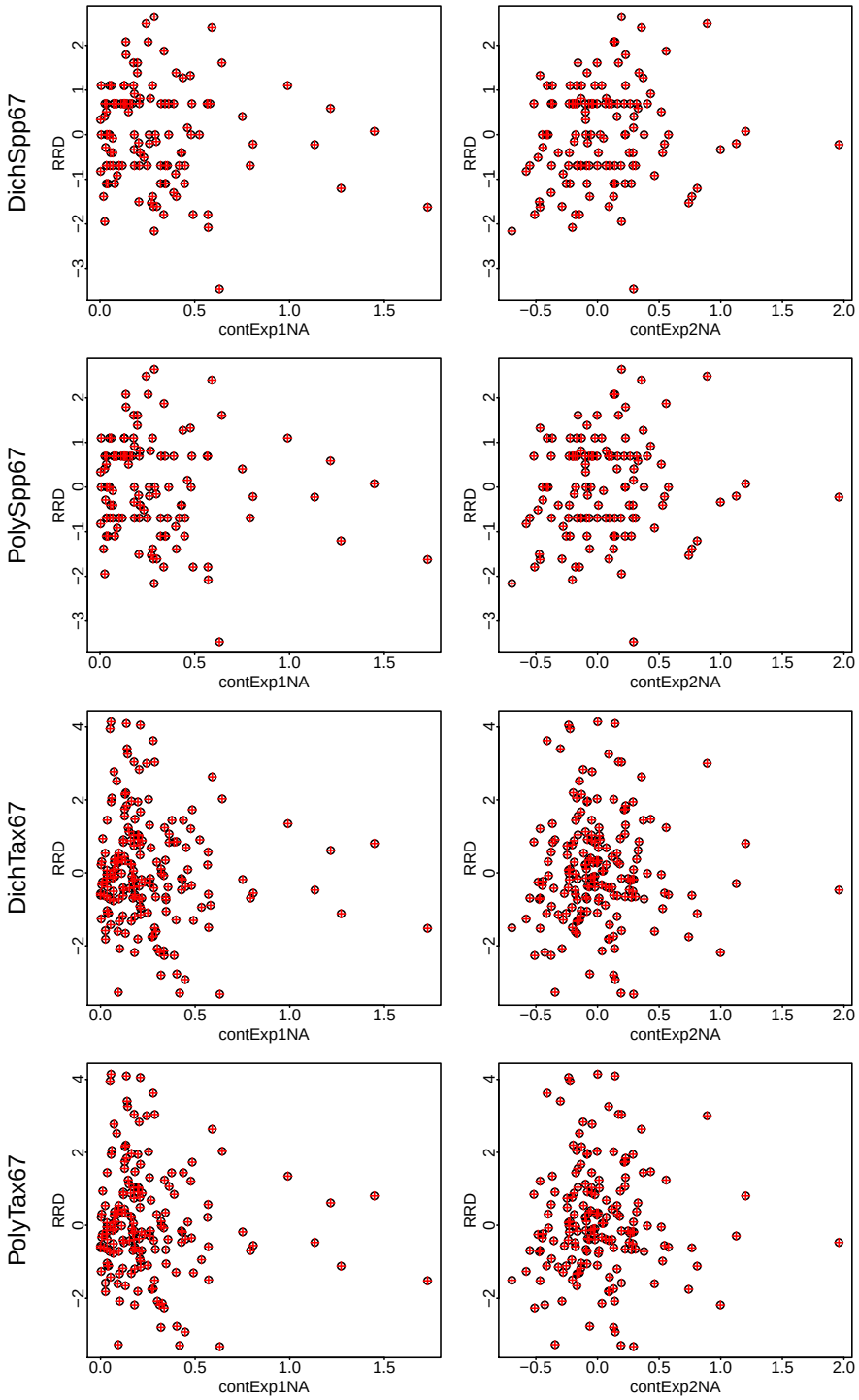

Figure 5: Overplotting of results from MacroCAIC (black) and `macrocaic()` (red) analyses using RRD and incomplete data.

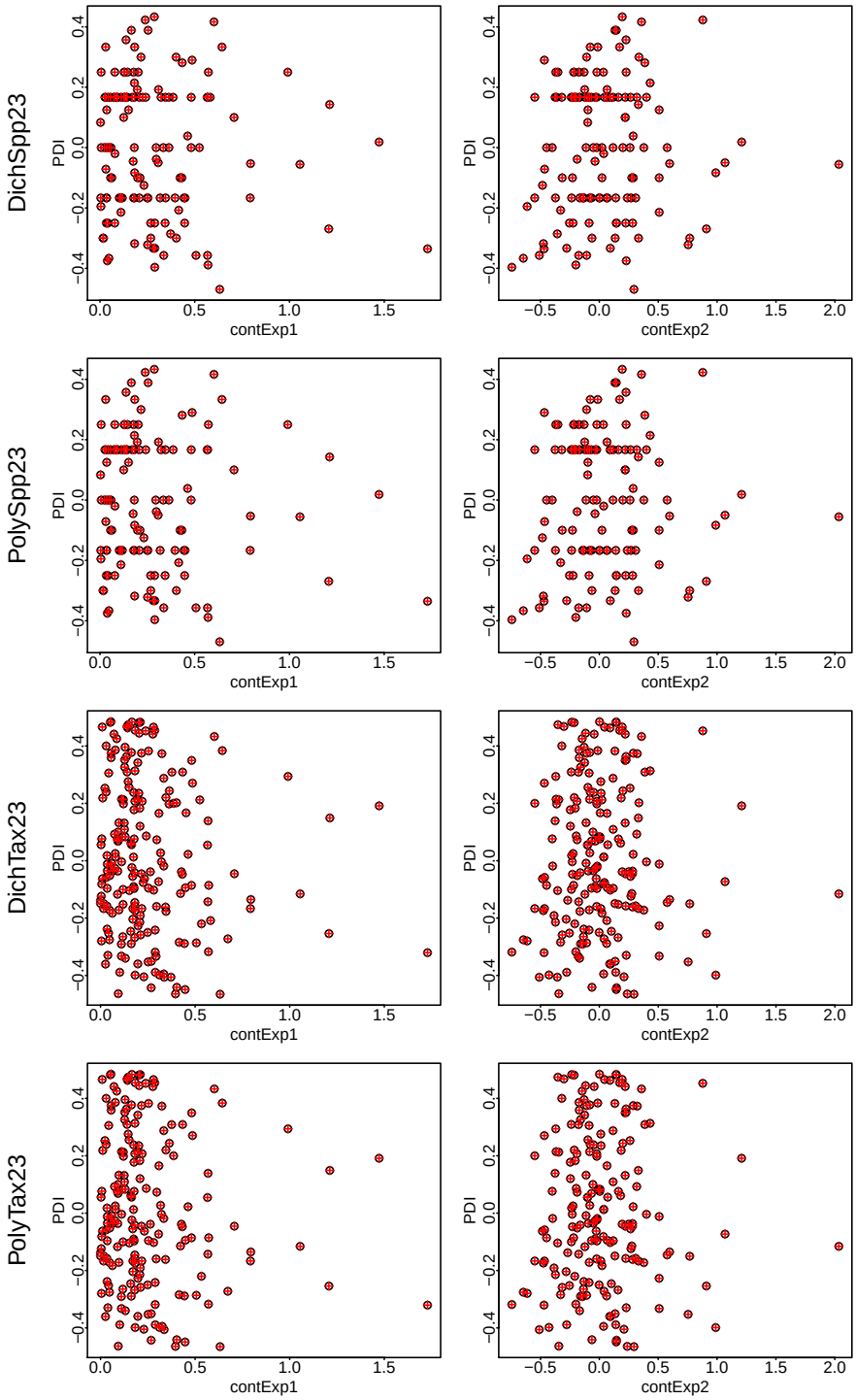

Figure 6: Overplotting of results from MacroCAIC (black) and `macrocaic()` (red) analyses using PDI and complete data.

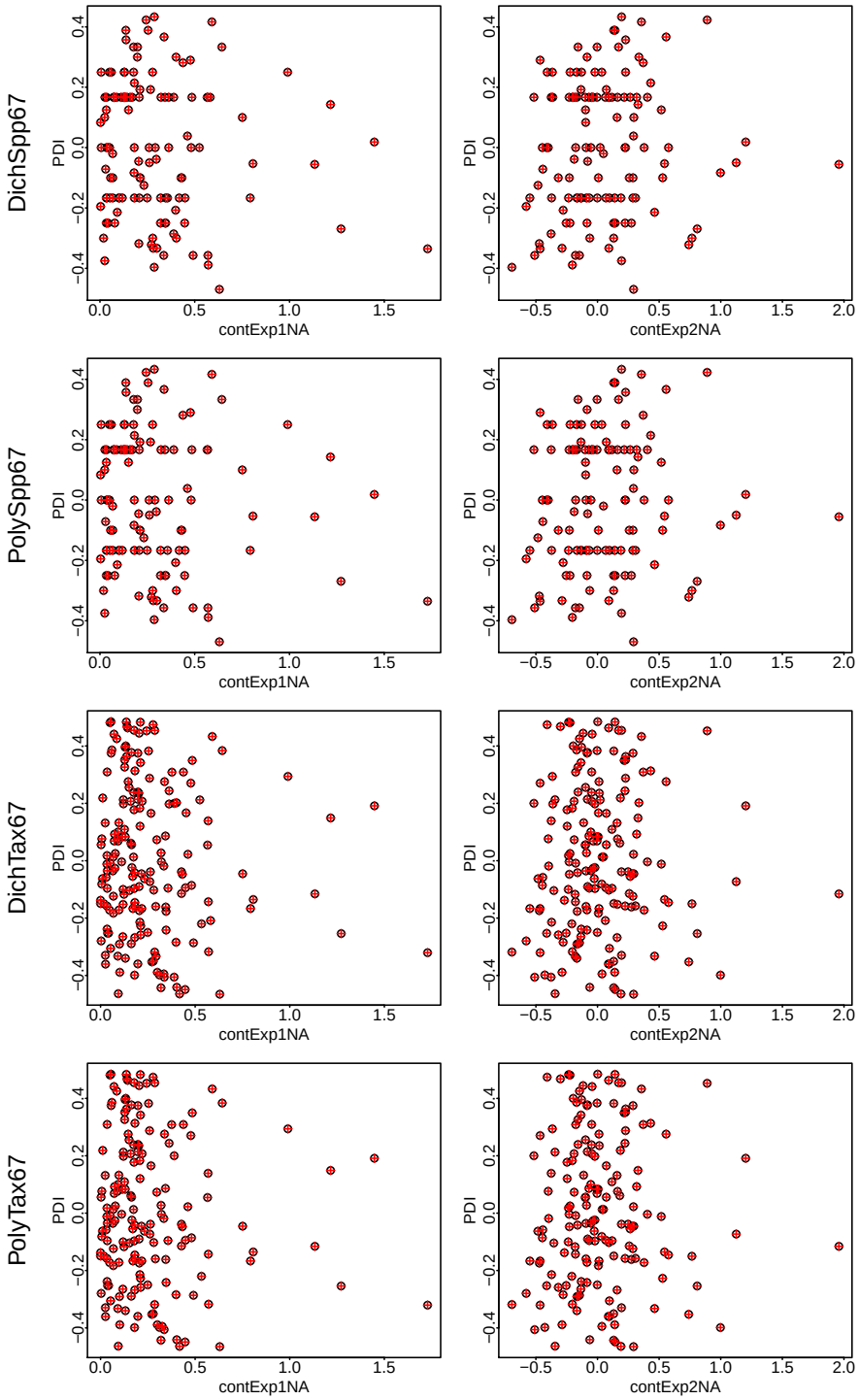

Figure 7: Overplotting of results from MacroCAIC (black) and `macrocaic()` (red) analyses using PDI and incomplete data.

option to consider the imbalance of the topology, treating each tip as a species, or of the distribution of species across tips, treating each tip as a taxon of one or more species. These benchmark tests used both options on both the dichotomous and polytomous benchmark trees. The data provided in the `benchFuscoOutputs` dataset contains, for each combination, a list containing a dataframe of the binned distribution data across nodes, a list of summary data for the tree and then a data frame of the individual node calculations.

The following code duplicates these analyses using the `fusco.test` function. Summary statistics comparing the two implementations are then show in Table **??**. Note that the median and quartile deviation of I in the summary information produced by 'IMB_CALC' and included in the benchmark files are *not directly reproducible* using `fusco.test`. However, the values presented here are medians and quartile deviations calculated in R directly from the node table generated by 'IMB_CALC'. The corrected nodal distributions of the original I statistic (Fusco and Cronk, 1995), calculated using the `plot` method, are shown plotted over the values from 'IMB_CALC' in Fig. 8.

```
> fstest.DiSpp <- fusco.test(benchDicho, rich=sppRichTips)
> fstest.DiTax <- fusco.test(benchDicho, rich=sppRichTaxa)
> fstest.PlSpp <- fusco.test(benchPoly, rich=sppRichTips)
> fstest.PlTax <- fusco.test(benchPoly, rich=sppRichTaxa)
> fstest.DiSpp.Hist <- plot(fstest.DiSpp, I.prime=FALSE, plot=FALSE)
> fstest.DiTax.Hist <- plot(fstest.DiTax, I.prime=FALSE, plot=FALSE)
> fstest.PlSpp.Hist <- plot(fstest.PlSpp, I.prime=FALSE, plot=FALSE)
> fstest.PlTax.Hist <- plot(fstest.PlTax, I.prime=FALSE, plot=FALSE)
> data(benchFuscoOutputs)
> DiSpp <- cbind(fstest.DiSpp.Hist, FuscoDiSpp$distTab)
> DiTax <- cbind(fstest.DiTax.Hist, FuscoDiTax$distTab)
> PlSpp <- cbind(fstest.PlSpp.Hist, FuscoPolySpp$distTab)
> PlTax <- cbind(fstest.PlTax.Hist, FuscoPolyTax$distTab)
>
```

4.2 Testing against the extended implementation (I, I', I_w)in MeSA.

The original I imbalance statistic showed a bias related to node size, demonstrated and corrected by Purvis et al. (2002). This revised calculation using either weights (I_w) or a modification to the calculation (I') was implemented in the program MeSA (Agapow, 2006). The dataset `benchMesaOutputs` contains the output from MeSA v1.9.23 running under Mac OS 10.5.3, repeating the calculations in Purvis et al. (2002) of weights, I and I' on a genus-level tree of the Syrphidae. These calculations are repeated using `fusco.test` and the calculated mean values for both the original I and I' from both implementations are shown.

```
> data(syrphidae)
> fstest.Syrph <- fusco.test(syrphidaeTree, dat=syrphidaeRich, rich=nSpp, names=genus)
> summary(fstest.Syrph)


Fusco test for phylogenetic imbalance

  Tree with 105 informative nodes and 204 tips.
  Tips are higher taxa containing 5330 species.
  95.0% confidence intervals around 0.5 randomised using 1000 replicates.

  Mean I prime: 0.629 [0.433,0.567]
  Median I: 0.727
  Quartile deviation in I: 0.291

        Wilcoxon signed rank test with continuity correction
```

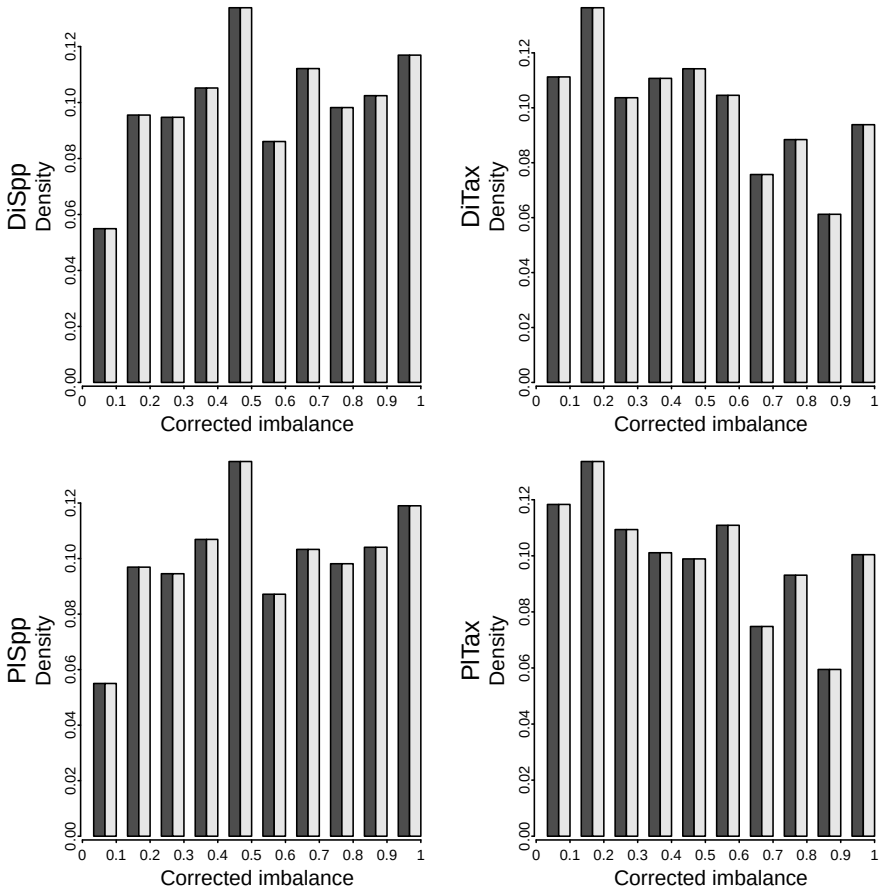

Figure 8: Barplots of nodal imbalance distributions from the program 'IMB_CALC' (grey bars) showing the corresponding values calculated using `fusco.test` in black.

```
data:  object$observed$I.prime
V = 3986, p-value = 0.0001197
alternative hypothesis: true location is not equal to 0.5
```

```
> data(benchMesaOutputs)
> mean(MeSA.I$Iprime)
```

```
[1] 0.6293965
```

```
> median(MeSA.I$I)
```

```
[1] 0.727405
```

References

Paul-Michael Agapow and Nick J B Isaac. Macrocaic: revealing correlates of species richness by comparative analysis. *Diversity and Distributions*, 8:41–43, 2002.

Joseph Felsenstein. Phylogenies and the comparative method. *American Naturalist*, 125:1–15, 1985.

Guiseppe Fusco and Quentin C B Cronk. A new method for evaluating the shape of large phylogenies. *Journal of Theoretical Biology*, 175:235–243, 1995.

Andy Purvis and Andy Rambaut. Comparative analysis by independent contrasts (caic): an apple macintosh application for analysing comparative data. *Computer Applications In the Biosciences*, 11:247–251, 1995.

Andy Purvis, Aris Katzourakis, and Paul-Michael Agapow. Evaluating phylogenetic tree shape: Two modifications to fusco & cronk's method. *Journal of Theoretical Biology*, 214:99–103, 2002. doi: DOI 10.1006/jtbi.2001.2443.