Discrimination Analysis

March 9, 2014

Preamble

This document details the estiamtion procedure for stable isotope dicrimination and fatty acid conversion coefficients. The data used is from Stowasser et al 2006 (JEMBE 333: 97114), and is available as .csv files in the same folder as this document. We group all fish together as we don't know the proportions of individual fish species that were fed in the treatment. The uncertainty in estiamted dicromination coefficients will reflect any variability in species specific coefficients, so will mainly just introduce additional uncertainty in the final analysis.

Stable Isotope discimination

We start with estiamting SI discrimination based on fish and crustacean only diets. First we read the .csv files with raw data:

Data grooming

```
prey.table <- read.csv("Prey_SI.csv", header = T)
pred.table <- read.csv("Predator_SI.csv", header = T, stringsAsFactors = F)
```

We'll need a few items later, such as the number of preys, here set to two - fish and crustacean - and an index of prey samples.

```
# index
prey.type <- as.numeric(prey.table[, 1] == "Grass shrimp") + 1
# number of prey types
n.preys <- 2
# total number of prey samples
n.preys.samps <- length(prey.type)
# final prey table
prey <- prey.table[, 2:3]
```

Similarly, for the predators we'll need a feed index to identify which food each predator was fed on.

```
# feed type index
feed.type <- pred.table[, 1]
# fish
feed.type[pred.table[, 1] == "F"] <- 1
# crustacean
feed.type[pred.table[, 1] == "C"] <- 2
# use only fish and crustations diets, rest will be assessed subset
idx <- which(feed.type %in% c(1, 2))
feed.type <- as.numeric(feed.type[idx])

# subset predator table in the same way
pred <- pred.table[idx, 2:3]
# number of predator samples
n.preds.samps <- length(idx)
```

Priors

We now set up priors for the analysis. We need priors for both the intercept and concentration dependence of the priors. Priors for ΔN follow from Hussey et al 2014 (Ecology Letters Volume 17, Issue 2, pages 239250, February 2014). Priors for ΔC are from Caut et al 2009 (Journal of Applied Ecology, 46, 443453).

```
# beta.not - 95% interval contains 2SD, if symmetric then
bnot.prior.N <- 5.92
sd.bnot.N <- 5.92 - 4.55
bnot.tau.prior.N <- 1/(sd.bnot.N^2)

beta.prior.N <- -0.27
sd.beta.N <- -0.27 - -0.41
beta.tau.prior.N <- 1/(sd.beta.N^2)

# no variance given in Caut, set prior to clearly larger than in N
bnot.prior.C <- -2.85
sd.bnot.C <- 2
bnot.tau.prior.C <- 1/(sd.bnot.C^2)

beta.prior.C <- -0.21
sd.beta.C <- 0.4
beta.tau.prior.C <- 1/(sd.beta.C^2)

# final priors
bnot.prior <- c(bnot.prior.C, bnot.prior.N)
```

```r
bnot.tau.prior <- c(bnot.tau.prior.C, bnot.tau.prior.N)
beta.prior <- c(beta.prior.C, beta.prior.N)
beta.tau.prior <- c(beta.tau.prior.C, beta.tau.prior.N)
```

We simply set the prior mean for the prey mean to the sample mean, but leave the variance vague so as to not force the mean.

```r
require(dplyr)

## Loading required package:  dplyr
##
## Attaching package:  'dplyr'
##
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

# predator mean priors
prior.mu <- data.frame(feed = prey.type, prey) %.% group_by(feed) %.% summarise(mu.C = mean(
    mu.N = mean(X_15N)) %.% arrange(feed)

prior.mu <- as.matrix(prior.mu[, 2:3])
```

Jags analysis

This is all put into the appropriate format for jags, and off we go...

```r
input <- list(prey = prey, pred = pred, n.preys.samps = n.preys.samps, n.preys = n.preys,
    n.preds.samps = n.preds.samps, prey.type = prey.type, feed.type = feed.type,
    prior.mu = prior.mu, bnot.prior = bnot.prior, beta.prior = beta.prior, bnot.tau.prior =
    beta.tau.prior = beta.tau.prior)

require(rjags)

## Loading required package:  rjags
## Loading required package:  coda
## Loading required package:  lattice
## Linked to JAGS 3.4.0
## Loaded modules:  basemod,bugs

DM <- jags.model("Discrim.model.SI.R", n.chains = 3, inits = list(mu = prior.mu,
    beta.not = bnot.prior, beta.reg = beta.prior), data = input)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
##    Graph Size: 133
##
## Initializing model

update(DM, 10000)

samps <- coda.samples(DM, c("pred.discr", "beta.reg", "beta.not", "mu"), n.iter = 1e+05,
    thin = 100)
```

All done, checking output visually for adequacy (convergence, lack of strong autocorrelation in the Markov Chains) and extract estiamted coefficients for each preys species and isotope and put the resulting matrix into the right format.

```
plot(samps, ask = T)  # need to run this outside of compilation
```

```
summary(samps)

##
## Iterations = 10100:110000
## Thinning interval = 100
## Number of chains = 3
## Sample size per chain = 1000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                   Mean    SD Naive SE Time-series SE
## beta.not[1]     -2.502 1.934  0.03531        0.03970
## beta.not[2]      7.980 0.691  0.01261        0.01231
## beta.reg[1]     -0.235 0.114  0.00209        0.00234
## beta.reg[2]     -0.109 0.130  0.00238        0.00229
## mu[1,1]        -20.005 0.501  0.00915        0.00916
## mu[2,1]        -17.157 0.360  0.00657        0.00655
## mu[1,2]          5.772 0.381  0.00696        0.00688
## mu[2,2]          4.179 0.246  0.00449        0.00463
## pred.discr[1,1]  2.191 0.561  0.01023        0.01054
## pred.discr[2,1]  1.529 0.443  0.00809        0.00783
## pred.discr[1,2]  7.356 0.559  0.01021        0.01021
## pred.discr[2,2]  7.521 0.508  0.00928        0.00928
##
## 2. Quantiles for each variable:
##
```

```
##                   2.5%     25%     50%      75%    97.5%
## beta.not[1]      -6.291  -3.746  -2.473  -1.2021   1.2123
## beta.not[2]       6.627   7.516   7.978   8.4265   9.3339
## beta.reg[1]      -0.457  -0.309  -0.234  -0.1587  -0.0129
## beta.reg[2]      -0.367  -0.196  -0.108  -0.0192   0.1424
## mu[1,1]         -21.008 -20.326 -20.011 -19.7001 -18.9887
## mu[2,1]         -17.834 -17.360 -17.166 -16.9763 -16.4156
## mu[1,2]           5.094   5.521   5.745   5.9910   6.5924
## mu[2,2]           3.591   4.061   4.196   4.3241   4.6145
## pred.discr[1,1]   1.040   1.841   2.196   2.5736   3.2240
## pred.discr[2,1]   0.598   1.282   1.536   1.8120   2.3481
## pred.discr[1,2]   6.255   6.986   7.348   7.7319   8.4126
## pred.discr[2,2]   6.513   7.199   7.513   7.8564   8.5312
```

It seems that despite the informative priors, the poterior for both beta.not[2] and beta.reg[2] (coeffs for ΔN) are higher and lower in their posterior median than their respective priors. The higt intercept leads to ΔN discrimination coefficients that are both high and similar in magniturde for both prey species, indicating that this may be linked to the tissue type or to the predator. For ΔC the posterior medians are close to the respective priors means. Estiamted ΔC discrimination is positive for both species.

```r
# get estiamted discrimintation from all chains:
ix <- grep("pred.discr", colnames(samps[[1]]))

# combine chains
r.samps <- do.call("rbind", samps)[, ix]
dim(r.samps)

## [1] 3000    4

# into format for analysis
discr.means <- matrix(apply(r.samps, 2, mean), 2, 2)
# pretend discrimination is the same for all fish species
discr.means <- rbind(discr.means[1, ], discr.means[1, ], discr.means)

discr.var <- matrix(apply(r.samps, 2, var), 2, 2)
# pretend discrimination is the same for all fish species
discr.var <- rbind(discr.var[1, ], discr.var[1, ], discr.var)

# write to file
colnames(discr.means) <- colnames(prey.table)[2:3]
rownames(discr.means) <- unique(prey.table[, 1])
write.csv(discr.means, file = "discr.means.csv")
discr.means
```

```
##                    X_13C X_15N
## Silverside         2.191 7.356
## Sailfin Molly      2.191 7.356
## Sheepshead minnow  2.191 7.356
## Grass shrimp       1.529 7.521

# write to file
colnames(discr.var) <- colnames(prey.table)[2:3]
rownames(discr.var) <- unique(prey.table[, 1])
write.csv(discr.var, file = "discr.var.csv")
discr.var

##                     X_13C  X_15N
## Silverside         0.3142 0.3125
## Sailfin Molly      0.3142 0.3125
## Sheepshead minnow  0.3142 0.3125
## Grass shrimp       0.1965 0.2585
```

All done for SI, moving on to FAP...

Fatty Acid Conversion Coefficients

Again, we start with data grooming, reading in raw data and preparing it for analysis:

Data grooming

```
require(fastinR)

## Loading required package:  fastinR
##
## Attaching package:  'fastinR'
##
## The following object is masked from 'package:dplyr':
##
##     select_vars
##
## The following object is masked from 'package:utils':
##
##     adist

prey.table.FA <- t(read.csv("Prey_FA.csv", header = F, row.names = 1))
pred.table.FA <- t(read.csv("Predator_FA.csv", header = T, stringsAsFactors = F,
    row.names = 1))
```

```r
# prey index
prey.type <- rep(1, nrow(prey.table.FA))
prey.type[prey.table.FA[, 1] == "Grass Shrimp"] <- 2

# msc
n.preys <- 2
n.preys.samps <- length(prey.type)
n.fats <- 25

# replace zeros in compositions with min recorded for that SI. This is
# somewhat arbitrary.
prey.ix <- prey.table.FA[, 1]
prey.table.FA <- matrix(as.numeric(prey.table.FA[, 2:26]), n.preys.samps, n.fats)
for (i in 1:ncol(prey.table.FA)) {
    prey.table.FA[prey.table.FA[, i] == 0, i] = min(prey.table.FA[prey.table.FA[,
        i] > 0, i])
}
# log ratio transform
prey = alr(prey.table.FA)

for (i in 1:ncol(pred.table.FA)) {
    pred.table.FA[pred.table.FA[, i] == 0, i] = min(pred.table.FA[pred.table.FA[,
        i] > 0, i])
}
# log ratio transform
pred = alr(pred.table.FA)

# feed type index
feed.type <- pred.table.FA[, 1]
FAs <- colnames(pred.table.FA)
# use only samples taht were fed on single diet for >=20days
feed.type[which(regexpr("F", rownames(pred.table.FA)) == 1)[9:15]] <- 1
# take ouyt an early sample
feed.type[which(regexpr("C", rownames(pred.table.FA)) == 1)[c(6:8, 10:13)]] <- 2
# use only fish and crustations diets, rest will be assessed
idx <- which(feed.type %in% c(1, 2))
feed.type <- as.integer(feed.type[idx])

# final predator table
pred <- pred[idx, ]
n.preds.samps <- length(idx)
```

FAP priors

There seems to be little evidence about consistent patterns in conversion coefficients for FAP, so we won't set any informative priors.

```r
# predator mean priors
pt <- aggregate(prey.table.FA, list(prey.type), gmean)
prior.mu <- data.matrix(alr(pt[, 2:length(pt)]))

n.fats <- ncol(prey.table.FA)
m.fats <- n.fats - 1
# reasonably vague priors
S <- diag(0.01, m.fats)
R <- diag(0.01, n.fats)

# uniform prior for Dirichlet on conversion coeffs
p = rep(1/n.fats, n.fats)
zeros = rep(0, n.fats)
```

Ready to run JAGS...

```r
require(rjags)

DM <- jags.model("Discrim.model.FA.R", n.chains = 3)

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
##    Graph Size: 1203
##
## Initializing model

update(DM, 10000)

samps.FA <- coda.samples(DM, c("beta.reg"), n.iter = 1e+05, thin = 100)
```

Again, we examine the output for auto-correlation and convergence. Seems ok, despite the 'long tails'.

```r
plot(samps.FA, ask = T)
crosscorr.plot(samps.FA)
```

```r
# display posterior summary
summary(samps.FA)

##
```

```
## Iterations = 11100:111000
## Thinning interval = 100
## Number of chains = 3
## Sample size per chain = 1000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                    Mean      SD Naive SE Time-series SE
## beta.reg[1,1]   0.01906 0.00453 8.27e-05       3.15e-04
## beta.reg[2,1]   0.01444 0.00327 5.97e-05       1.87e-04
## beta.reg[1,2]   0.02254 0.00486 8.87e-05       3.17e-04
## beta.reg[2,2]   0.02125 0.00403 7.35e-05       1.98e-04
## beta.reg[1,3]   0.02323 0.00579 1.06e-04       4.66e-04
## beta.reg[2,3]   0.02518 0.00449 8.19e-05       2.13e-04
## beta.reg[1,4]   0.04170 0.01297 2.37e-04       1.49e-03
## beta.reg[2,4]   0.03472 0.00647 1.18e-04       3.19e-04
## beta.reg[1,5]   0.02018 0.00475 8.67e-05       3.10e-04
## beta.reg[2,5]   0.01264 0.00264 4.82e-05       1.43e-04
## beta.reg[1,6]   0.03013 0.01010 1.84e-04       1.11e-03
## beta.reg[2,6]   0.00838 0.00272 4.96e-05       2.28e-04
## beta.reg[1,7]   0.03799 0.02006 3.66e-04       2.74e-03
## beta.reg[2,7]   0.01590 0.00815 1.49e-04       8.75e-04
## beta.reg[1,8]   0.04067 0.01693 3.09e-04       2.21e-03
## beta.reg[2,8]   0.03752 0.01632 2.98e-04       1.65e-03
## beta.reg[1,9]   0.06902 0.01596 2.91e-04       1.42e-03
## beta.reg[2,9]   0.06360 0.01338 2.44e-04       8.34e-04
## beta.reg[1,10]  0.02099 0.00594 1.08e-04       4.63e-04
## beta.reg[2,10]  0.01529 0.00329 6.01e-05       1.96e-04
## beta.reg[1,11]  0.02812 0.00597 1.09e-04       3.88e-04
## beta.reg[2,11]  0.01669 0.00337 6.16e-05       2.60e-04
## beta.reg[1,12]  0.01940 0.00583 1.06e-04       5.15e-04
## beta.reg[2,12]  0.01473 0.00276 5.04e-05       1.51e-04
## beta.reg[1,13]  0.01738 0.00801 1.46e-04       1.03e-03
## beta.reg[2,13]  0.01263 0.00350 6.39e-05       2.21e-04
## beta.reg[1,14]  0.01330 0.00366 6.68e-05       3.06e-04
## beta.reg[2,14]  0.00932 0.00175 3.20e-05       8.78e-05
## beta.reg[1,15]  0.01196 0.00460 8.41e-05       3.93e-04
## beta.reg[2,15]  0.01102 0.00318 5.80e-05       1.95e-04
## beta.reg[1,16]  0.13617 0.02540 4.64e-04       2.21e-03
## beta.reg[2,16]  0.10983 0.01757 3.21e-04       8.09e-04
## beta.reg[1,17]  0.03415 0.00791 1.44e-04       6.33e-04
## beta.reg[2,17]  0.07823 0.03007 5.49e-04       2.89e-03
## beta.reg[1,18]  0.11895 0.02217 4.05e-04       1.34e-03
## beta.reg[2,18]  0.07376 0.01277 2.33e-04       5.64e-04
```

```
## beta.reg[1,19] 0.02146 0.00646 1.18e-04        5.07e-04
## beta.reg[2,19] 0.02664 0.00595 1.09e-04        2.62e-04
## beta.reg[1,20] 0.10792 0.02051 3.75e-04        1.51e-03
## beta.reg[2,20] 0.06560 0.01116 2.04e-04        5.15e-04
## beta.reg[1,21] 0.02814 0.00721 1.32e-04        6.15e-04
## beta.reg[2,21] 0.03647 0.01117 2.04e-04        7.33e-04
## beta.reg[1,22] 0.03218 0.01294 2.36e-04        1.65e-03
## beta.reg[2,22] 0.10607 0.02177 3.97e-04        1.31e-03
## beta.reg[1,23] 0.02889 0.00638 1.17e-04        3.80e-04
## beta.reg[2,23] 0.07239 0.01211 2.21e-04        5.25e-04
## beta.reg[1,24] 0.05435 0.00978 1.79e-04        5.08e-04
## beta.reg[2,24] 0.07002 0.01271 2.32e-04        7.04e-04
## beta.reg[1,25] 0.02211 0.00686 1.25e-04        1.30e-03
## beta.reg[2,25] 0.04767 0.00605 1.10e-04        4.13e-04
##
## 2. Quantiles for each variable:
##
##                       2.5%     25%     50%     75%   97.5%
## beta.reg[1,1]   0.01068 0.01626 0.01867 0.02150 0.0291
## beta.reg[2,1]   0.00883 0.01267 0.01421 0.01582 0.0210
## beta.reg[1,2]   0.01287 0.02004 0.02248 0.02495 0.0325
## beta.reg[2,2]   0.01474 0.01917 0.02092 0.02271 0.0310
## beta.reg[1,3]   0.01444 0.01979 0.02277 0.02570 0.0352
## beta.reg[2,3]   0.01666 0.02275 0.02482 0.02711 0.0362
## beta.reg[1,4]   0.02430 0.03704 0.04081 0.04438 0.0605
## beta.reg[2,4]   0.02430 0.03120 0.03408 0.03699 0.0511
## beta.reg[1,5]   0.01162 0.01751 0.01975 0.02230 0.0307
## beta.reg[2,5]   0.00806 0.01141 0.01248 0.01355 0.0181
## beta.reg[1,6]   0.01379 0.02313 0.02910 0.03610 0.0544
## beta.reg[2,6]   0.00431 0.00650 0.00797 0.00976 0.0147
## beta.reg[1,7]   0.01170 0.02327 0.03310 0.04760 0.0891
## beta.reg[2,7]   0.00572 0.00983 0.01400 0.01991 0.0380
## beta.reg[1,8]   0.01678 0.02851 0.03778 0.05029 0.0794
## beta.reg[2,8]   0.01435 0.02659 0.03497 0.04494 0.0773
## beta.reg[1,9]   0.04488 0.06068 0.06683 0.07370 0.1175
## beta.reg[2,9]   0.04206 0.05765 0.06290 0.06790 0.0880
## beta.reg[1,10]  0.01163 0.01762 0.02036 0.02330 0.0380
## beta.reg[2,10]  0.01006 0.01376 0.01510 0.01646 0.0210
## beta.reg[1,11]  0.01720 0.02484 0.02776 0.03072 0.0416
## beta.reg[2,11]  0.01101 0.01487 0.01633 0.01800 0.0252
## beta.reg[1,12]  0.01041 0.01580 0.01855 0.02198 0.0335
## beta.reg[2,12]  0.01005 0.01331 0.01459 0.01584 0.0206
## beta.reg[1,13]  0.00697 0.01202 0.01585 0.02085 0.0369
## beta.reg[2,13]  0.00695 0.01038 0.01212 0.01430 0.0210
## beta.reg[1,14]  0.00736 0.01085 0.01288 0.01519 0.0217
```

```
## beta.reg[2,14] 0.00649 0.00837 0.00911 0.00997 0.0138
## beta.reg[1,15] 0.00581 0.00898 0.01110 0.01365 0.0242
## beta.reg[2,15] 0.00633 0.00869 0.01058 0.01281 0.0182
## beta.reg[1,16] 0.08783 0.12140 0.13481 0.14883 0.1939
## beta.reg[2,16] 0.07274 0.10074 0.10985 0.11874 0.1487
## beta.reg[1,17] 0.01936 0.02973 0.03371 0.03790 0.0515
## beta.reg[2,17] 0.03234 0.05744 0.07416 0.09409 0.1458
## beta.reg[1,18] 0.07366 0.10753 0.11781 0.12866 0.1715
## beta.reg[2,18] 0.04765 0.06677 0.07330 0.08043 0.1013
## beta.reg[1,19] 0.01299 0.01841 0.02077 0.02318 0.0336
## beta.reg[2,19] 0.01626 0.02318 0.02606 0.02947 0.0395
## beta.reg[1,20] 0.06276 0.09652 0.10866 0.12039 0.1484
## beta.reg[2,20] 0.04391 0.05964 0.06508 0.07092 0.0903
## beta.reg[1,21] 0.01442 0.02339 0.02789 0.03257 0.0435
## beta.reg[2,21] 0.01781 0.02874 0.03521 0.04243 0.0626
## beta.reg[1,22] 0.01377 0.02374 0.02979 0.03833 0.0613
## beta.reg[2,22] 0.06629 0.09301 0.10524 0.11742 0.1554
## beta.reg[1,23] 0.01759 0.02536 0.02840 0.03184 0.0422
## beta.reg[2,23] 0.04879 0.06576 0.07183 0.07827 0.1001
## beta.reg[1,24] 0.03540 0.04891 0.05385 0.05894 0.0767
## beta.reg[2,24] 0.04724 0.06318 0.06922 0.07524 0.0989
## beta.reg[1,25] 0.01094 0.01765 0.02207 0.02558 0.0400
## beta.reg[2,25] 0.03700 0.04338 0.04725 0.05172 0.0605
```

And process the output for the diet analysis.

```r
# get estimated discrimintation from all chains:
r.samps <- do.call("rbind", samps.FA)
dim(r.samps)

## [1] 3000    50

fish.cc.samples <- r.samps[, seq(1, 2 * n.fats, 2)]
shrimp.cc.samples <- r.samps[, seq(2, 2 * n.fats, 2)]

fish.cc <- colMeans(fish.cc.samples)
shrimp.cc <- colMeans(shrimp.cc.samples)

# combine and write to file, repeat for 3 fish species in fianl analysis
ccs <- rbind(fish.cc, fish.cc, fish.cc, shrimp.cc)
rownames(ccs) <- unique(prey.ix)[-4]
colnames(ccs) <- colnames(pred.table.FA)
write.csv(ccs, file = "cc_FA.csv")

# using independent ccs seems warranted here given cross-corr plot
fish.cc.var <- apply(fish.cc.samples, 2, var)
```

```r
shrimp.cc.var <- apply(shrimp.cc.samples, 2, var)
# combine
ccs.var <- rbind(fish.cc.var, fish.cc.var, fish.cc.var, shrimp.cc.var)
rownames(ccs.var) <- unique(prey.ix)[-4]
colnames(ccs.var) <- colnames(pred.table.FA)
write.csv(ccs.var, file = "cc_FA_var.csv")
```

All done!