QPAD version 3 documentation

Peter Solymos (solymos@ualberta.ca)

December 13, 2016

Contents

Introduction

The Boreal Avian Modelling (BAM) Project uses statistical offsets to correct for methodology and detectability differences across species, projects, and surveys times/locations as described in Solymos et al. [1]. We refer to this approach as the *QPAD approach*. QPAD version 1 was an internal working version of the parameter estimates. QPAD version 2 was published alongside Solymos et al. [1] and its supporting information. Since closing the data for this manuscript, more data have accumulated from Canada and the United States as part of BAM's point count database. QPAD version 3 represents a major update of the parameter estimates. The methodology is the same as outlined in the paper, but the data set is different in terms of number of surveys, species that met sample size requirements, and some of the predictor variables as outlined in the following sections. This technical report documents and explains the QPAD version 3 estimates.

Recommended citation

If using QPAD version 3 estimates, please cite the original paper:

Solymos, P., Matsuoka, S. M., Bayne, E. M., Lele, S. R., Fontaine, P., Cumming, S. G., Stralberg, D., Schmiegelow, F. K. A. and Song, S. J., 2013. Calibrating indices of avian density from non-standardized survey data: making the most of a messy situation. *Methods in Ecology and Evolution* **4**:1047–1058.

and the technical report:

Solymos, P., 2016. QPAD version 3 documentation. Technical Report, Boreal Avian Modelling Project, URL http://www.borealbirds.ca/library/index.php/technical_reports.

Data

The BAM database version 4 was used. We used 230658 survey events for time-removal, and 230658 for distance sampling models. We originally used data from 211 bird species that were considered as 'singing'

species, i.e. mostly detected by auditory cues in the Boreal. The final number of species for which we report parameter estimates is different from this number because of varying number of detections per species (both considering removal and binned distance design).

We considered the following predictor variables for time-removal sampling:

- `JDAY`: Julian day (based on date of survey, standardized as $x/365$),
- `TSSR`: hours since local sunrise (based on time of survey, standardized as $x/24$),
- `DSLS`: days since local spring (based on date of survey relative to 30-year average of the local start of the growing season, standardized as $x/365$),
- `JDAY2`, `TSSR2`, `DSLS2`: quadratic terms (x^2) for the above 3 variables.

Calculation of `JDAY` and `TSSR` follows Solymos et al. [1] and its supporting material, see McKenney et al. [2] for `DSLS` variable that is now added to v3 QPAD offset models and estimates.

The following model formulae were used as part of the right-hand side of the formula interface in the `cmulti` function (using `type = "rem"`):

0. `~ 1`
1. `~ JDAY`
2. `~ TSSR`
3. `~ JDAY + JDAY2`
4. `~ TSSR + TSSR2`
5. `~ JDAY + TSSR`
6. `~ JDAY + JDAY2 + TSSR`
7. `~ JDAY + TSSR + TSSR2`
8. `~ JDAY + JDAY2 + TSSR + TSSR2`
9. `~ DSLS`
10. `~ DSLS + DSLS2`
11. `~ DSLS + TSSR`
12. `~ DSLS + DSLS2 + TSSR`
13. `~ DSLS + TSSR + TSSR2`
14. `~ DSLS + DSLS2 + TSSR + TSSR2`

Models 0–8 are identical to the removal sampling models used in QPAD v2 and described in Solymos et al. [1] and its supporting material. Models 9–14 include the new variable `DSLS` and used in QPAD v3 only.

We considered the following predictor variables for distance sampling:

- `TREE`: tree cover (see [1] for source, standardized to 0–1 range),
- `LCC2`: land cover as a 4-level factor (`Forest`, `OpenWet` levels) based on the NALCMS land cover product [3],
- `LCC2`: land cover as a 2-level factor (`DecidMixed`, `Conif`, `Open`, `Wet` levels) based on the NALCMS land cover product [3].

The reason we switched from the Canadian land cover product (using 5 land cover categories as described in [1]) to the North American one [3] is that BAM data set includes many data points outside of Canada and we wanted to have consistent land cover classification across Canada and the US Boreal/Hemiboreal (including Alaska, Minnesota, for example).

The following reclassification was used to create for `LCC2` and `LCC4` variables from NALCMS:

Value	Class_name	LCC4	LCC2
0	No data	NA	NA
1	Temperate or sub-polar needleleaf forest	Conif	Forest
2	Sub-polar taiga needleleaf forest	Conif	Forest
3	Tropical or sub-tropical broadleaf evergreen	NA	NA
4	Tropical or sub-tropical broadleaf deciduous	NA	NA

Value	Class_name	LCC4	LCC2
5	Temperate or sub-polar broadleaf deciduous	DecidMixed	Forest
6	Mixed Forest	DecidMixed	Forest
7	Tropical or sub-tropical shrubland	NA	NA
8	Temperate or sub-polar shrubland	Open	OpenWet
9	Tropical or sub-tropical grassland	NA	NA
10	Temperate or sub-polar grassland	Open	OpenWet
11	Sub-polar or polar shrubland-lichen-moss	Open	OpenWet
12	Sub-polar or polar grassland-lichen-moss	Open	OpenWet
13	Sub-polar or polar barren-lichen-moss	Open	OpenWet
14	Wetland	Wet	OpenWet
15	Cropland	Open	OpenWet
16	Barren Lands	Open	OpenWet
17	Urban and Built-up	Open	OpenWet
18	Water	NA	NA
19	Snow and Ice	NA	NA

The following model formulae were used as part of the right-hand side of the formula interface in the `cmulti` function (using `type = "dis"`):

0. `~ 1`
1. `~ TREE`
2. `~ LCC2`
3. `~ LCC4`
4. `~ LCC2 + TREE`
5. `~ LCC4 + TREE`

Parameter estimates

We used the same general approach as in Solymos et al. [1], please consult that for details on method. Estimates can be retrieved using the **QPAD** R package [4]. Here is how the package can be installed:

```
if (!require(devtools))
    install.packages("devtools")
library(devtools)
install_github("psolymos/QPAD")
```

After the package is installed, one can load the estimates. Version 2 or 3 need to be explicitly specified, there is no default version defined. If the version is not defined as part of the call, the function will ask it interactively.

```
library(QPAD)
```

```
## QPAD 0.0-3    2016-10-17
## Estimates based on Solymos et al. 2013
## Use 'load_BAM_QPAD()' to load estimates
```

```
load_BAM_QPAD(version = 3)
```

```
## BAM QPAD parameter estimates loaded, version 3
```

QPAD v2 included results for 75, v3 includes results for 141. As part of QPAD v3, we are providing parameter estimates for the following bird species (* indicates the 66 species that were not part of QPAD v2 species list):

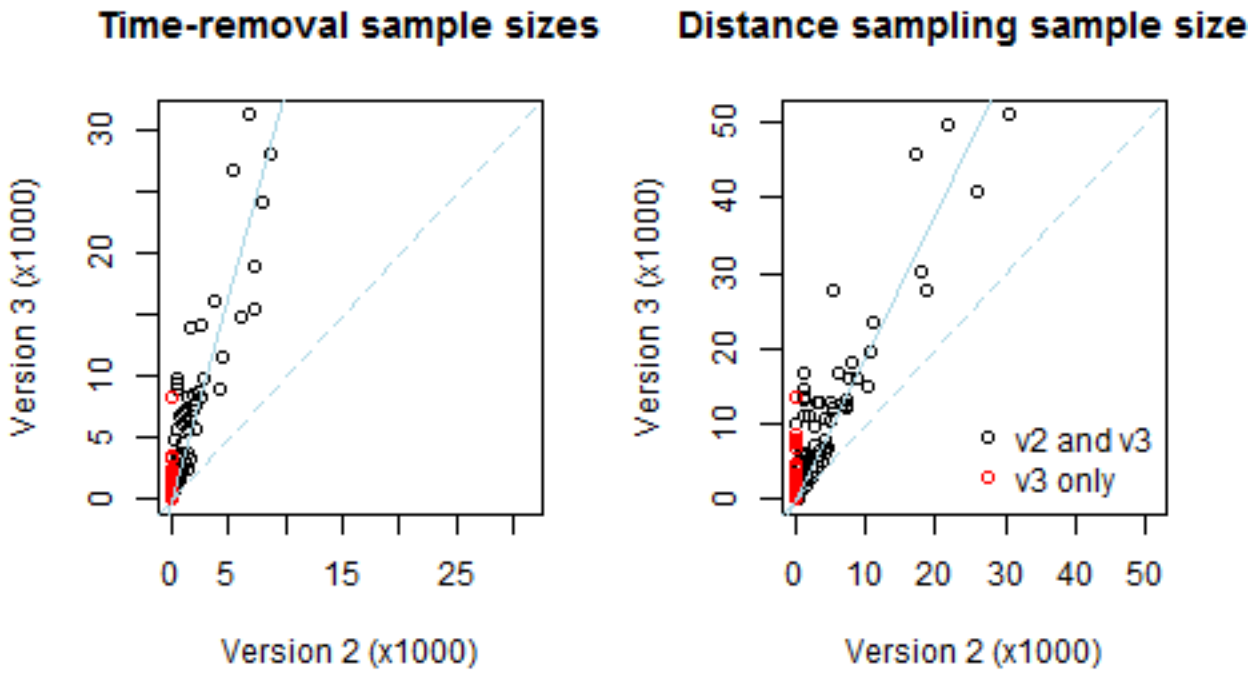
1. Alder Flycatcher (*Empidonax alnorum*, ALFL)

2. American Crow (*Corvus brachyrhynchos*, AMCR)
3. American Goldfinch (*Spinus tristis*, AMGO)
4. American Pipit (*Anthus rubescens*, AMPI)*
5. American Redstart (*Setophaga ruticilla*, AMRE)
6. American Robin (*Turdus migratorius*, AMRO)
7. American Tree Sparrow (*Spizella arborea*, ATSP)
8. American Three-toed Woodpecker (*Picoides dorsalis*, ATTW)*
9. Bank Swallow (*Riparia riparia*, BANS)*
10. Baltimore Oriole (*Icterus galbula*, BAOR)*
11. Barn Swallow (*Hirundo rustica*, BARS)*
12. Black-and-white Warbler (*Mniotilta varia*, BAWW)
13. Black-billed Cuckoo (*Coccyzus erythropthalmus*, BBCU)*
14. Black-billed Magpie (*Pica hudsonia*, BBMA)*
15. Bay-breasted Warbler (*Setophaga castanea*, BBWA)
16. Black-backed Woodpecker (*Picoides arcticus*, BBWO)*
17. Black-capped Chickadee (*Poecile atricapillus*, BCCH)
18. Belted Kingfisher (*Megaceryle alcyon*, BEKI)*
19. Brown-headed Cowbird (*Molothrus ater*, BHCO)
20. Blue-headed Vireo (*Vireo solitarius*, BHVI)
21. Blackburnian Warbler (*Setophaga fusca*, BLBW)
22. Blue Jay (*Cyanocitta cristata*, BLJA)
23. Blackpoll Warbler (*Setophaga striata*, BLPW)
24. Boreal Chickadee (*Poecile hudsonicus*, BOCH)
25. Bohemian Waxwing (*Bombycilla garrulus*, BOWA)*
26. Brewer's Blackbird (*Euphagus cyanocephalus*, BRBL)*
27. Brown Creeper (*Certhia americana*, BRCR)
28. Brown Thrasher (*Toxostoma rufum*, BRTH)*
29. Black-throated Blue Warbler (*Setophaga caerulescens*, BTBW)
30. Black-throated Green Warbler (*Setophaga virens*, BTNW)
31. Blue-winged Warbler (*Vermivora cyanoptera*, BWWA)*
32. Canada Warbler (*Cardellina canadensis*, CAWA)
33. Clay-colored Sparrow (*Spizella pallida*, CCSP)
34. Cedar Waxwing (*Bombycilla cedrorum*, CEDW)
35. Chipping Sparrow (*Spizella passerina*, CHSP)
36. Cliff Swallow (*Petrochelidon pyrrhonota*, CLSW)*
37. Cape May Warbler (*Setophaga tigrina*, CMWA)
38. Common Grackle (*Quiscalus quiscula*, COGR)*
39. Connecticut Warbler (*Oporornis agilis*, CONW)
40. Common Raven (*Corvus corax*, CORA)
41. Common Yellowthroat (*Geothlypis trichas*, COYE)
42. Chestnut-sided Warbler (*Setophaga pensylvanica*, CSWA)
43. Dark-eyed Junco (*Junco hyemalis*, DEJU)
44. Downy Woodpecker (*Picoides pubescens*, DOWO)*
45. Dunlin (*Calidris alpina*, DUNL)*
46. Eastern Bluebird (*Sialia sialis*, EABL)*
47. Eastern Kingbird (*Tyrannus tyrannus*, EAKI)*
48. Eastern Phoebe (*Sayornis phoebe*, EAPH)*
49. Eastern Towhee (*Pipilo erythrophthalmus*, EATO)*
50. Eastern Wood-Pewee (*Contopus virens*, EAWP)*
51. European Starling (*Sturnus vulgaris*, EUST)*
52. Evening Grosbeak (*Coccothraustes vespertinus*, EVGR)
53. Field Sparrow (*Spizella pusilla*, FISP)*
54. Fox Sparrow (*Passerella iliaca*, FOSP)
55. Great Crested Flycatcher (*Myiarchus crinitus*, GCFL)*

56. Golden-crowned Kinglet (*Regulus satrapa*, GCKI)
57. Golden-crowned Sparrow (*Zonotrichia atricapilla*, GCSP)*
58. Gray-cheeked Thrush (*Catharus minimus*, GCTH)*
59. Gray Jay (*Perisoreus canadensis*, GRAJ)
60. Gray Catbird (*Dumetella carolinensis*, GRCA)*
61. Greater Yellowlegs (*Tringa melanoleuca*, GRYE)*
62. Golden-winged Warbler (*Vermivora chrysoptera*, GWWA)*
63. Hammond's Flycatcher (*Empidonax hammondii*, HAFL)
64. Hairy Woodpecker (*Picoides villosus*, HAWO)*
65. Hermit Thrush (*Catharus guttatus*, HETH)
66. Horned Lark (*Eremophila alpestris*, HOLA)*
67. House Sparrow (*Passer domesticus*, HOSP)*
68. House Wren (*Troglodytes aedon*, HOWR)
69. Indigo Bunting (*Passerina cyanea*, INBU)*
70. Killdeer (*Charadrius vociferus*, KILL)*
71. Lapland Longspur (*Calcarius lapponicus*, LALO)*
72. Le Conte's Sparrow (*Ammodramus leconteii*, LCSP)
73. Least Flycatcher (*Empidonax minimus*, LEFL)
74. Lesser Yellowlegs (*Tringa flavipes*, LEYE)*
75. Lincoln's Sparrow (*Melospiza lincolnii*, LISP)
76. Magnolia Warbler (*Setophaga magnolia*, MAWA)
77. Marsh Wren (*Cistothorus palustris*, MAWR)*
78. Mountain Bluebird (*Sialia currucoides*, MOBL)*
79. Mourning Dove (*Zenaida macroura*, MODO)*
80. Mourning Warbler (*Geothlypis philadelphia*, MOWA)
81. Nashville Warbler (*Oreothlypis ruficapilla*, NAWA)
82. Nelson's Sparrow (*Ammodramus nelsoni*, NESP)*
83. Northern Flicker (*Colaptes auratus*, NOFL)*
84. Northern Parula (*Setophaga americana*, NOPA)
85. Northern Waterthrush (*Parkesia noveboracensis*, NOWA)
86. Orange-crowned Warbler (*Oreothlypis celata*, OCWA)
87. Olive-sided Flycatcher (*Contopus cooperi*, OSFL)
88. Ovenbird (*Seiurus aurocapilla*, OVEN)
89. Palm Warbler (*Setophaga palmarum*, PAWA)
90. Philadelphia Vireo (*Vireo philadelphicus*, PHVI)
91. Pine Grosbeak (*Pinicola enucleator*, PIGR)*
92. Pine Siskin (*Spinus pinus*, PISI)
93. Pine Warbler (*Setophaga pinus*, PIWA)*
94. Pileated Woodpecker (*Dryocopus pileatus*, PIWO)*
95. Purple Finch (*Carpodacus purpureus*, PUFI)
96. Rose-breasted Grosbeak (*Pheucticus ludovicianus*, RBGR)
97. Red-breasted Nuthatch (*Sitta canadensis*, RBNU)
98. Ruby-crowned Kinglet (*Regulus calendula*, RCKI)
99. Red Crossbill (*Loxia curvirostra*, RECR)*
100. Red-eyed Vireo (*Vireo olivaceus*, REVI)
101. Rock Sandpiper (*Calidris ptilocnemis*, ROSA)*
102. Ruby-throated Hummingbird (*Archilochus colubris*, RTHU)*
103. Rusty Blackbird (*Euphagus carolinus*, RUBL)
104. Ruffed Grouse (*Bonasa umbellus*, RUGR)*
105. Red-winged Blackbird (*Agelaius phoeniceus*, RWBL)
106. Savannah Sparrow (*Passerculus sandwichensis*, SAVS)
107. Scarlet Tanager (*Piranga olivacea*, SCTA)*
108. Sedge Wren (*Cistothorus platensis*, SEWR)*
109. Solitary Sandpiper (*Tringa solitaria*, SOSA)*

110. Song Sparrow (*Melospiza melodia*, SOSP)
111. Spruce Grouse (*Falcipennis canadensis*, SPGR)*
112. Spotted Sandpiper (*Actitis macularius*, SPSA)*
113. Swamp Sparrow (*Melospiza georgiana*, SWSP)
114. Swainson's Thrush (*Catharus ustulatus*, SWTH)
115. Tennessee Warbler (*Oreothlypis peregrina*, TEWA)
116. Townsend's Solitaire (*Myadestes townsendi*, TOSO)*
117. Townsend's Warbler (*Setophaga townsendi*, TOWA)*
118. Tree Swallow (*Tachycineta bicolor*, TRES)
119. Upland Sandpiper (*Bartramia longicauda*, UPSA)*
120. Varied Thrush (*Ixoreus naevius*, VATH)
121. Veery (*Catharus fuscescens*, VEER)
122. Vesper Sparrow (*Pooecetes gramineus*, VESP)*
123. Warbling Vireo (*Vireo gilvus*, WAVI)
124. White-breasted Nuthatch (*Sitta carolinensis*, WBNU)*
125. White-crowned Sparrow (*Zonotrichia leucophrys*, WCSP)
126. Western Tanager (*Piranga ludoviciana*, WETA)
127. Western Wood-Pewee (*Contopus sordidulus*, WEWP)
128. Willow Ptarmigan (*Lagopus lagopus*, WIPT)*
129. Wilson's Snipe (*Gallinago delicata*, WISN)*
130. Wilson's Warbler (*Cardellina pusilla*, WIWA)
131. Winter Wren (*Troglodytes hiemalis*, WIWR)
132. Wood Thrush (*Hylocichla mustelina*, WOTH)*
133. White-throated Sparrow (*Zonotrichia albicollis*, WTSP)
134. White-winged Crossbill (*Loxia leucoptera*, WWCR)
135. Yellow-billed Cuckoo (*Coccyzus americanus*, YBCU)*
136. Yellow-bellied Flycatcher (*Empidonax flaviventris*, YBFL)
137. Yellow-bellied Sapsucker (*Sphyrapicus varius*, YBSA)*
138. Yellow Warbler (*Setophaga petechia*, YEWA)
139. Yellow-headed Blackbird (*Xanthocephalus xanthocephalus*, YHBL)*
140. Yellow-rumped Warbler (*Setophaga coronata*, YRWA)
141. Yellow-throated Vireo (*Vireo flavifrons*, YTVI)*

Sample sizes have increased by 437.9% for time-removal, and 314.5% for distance-sampling models across the species that were included in both versions.

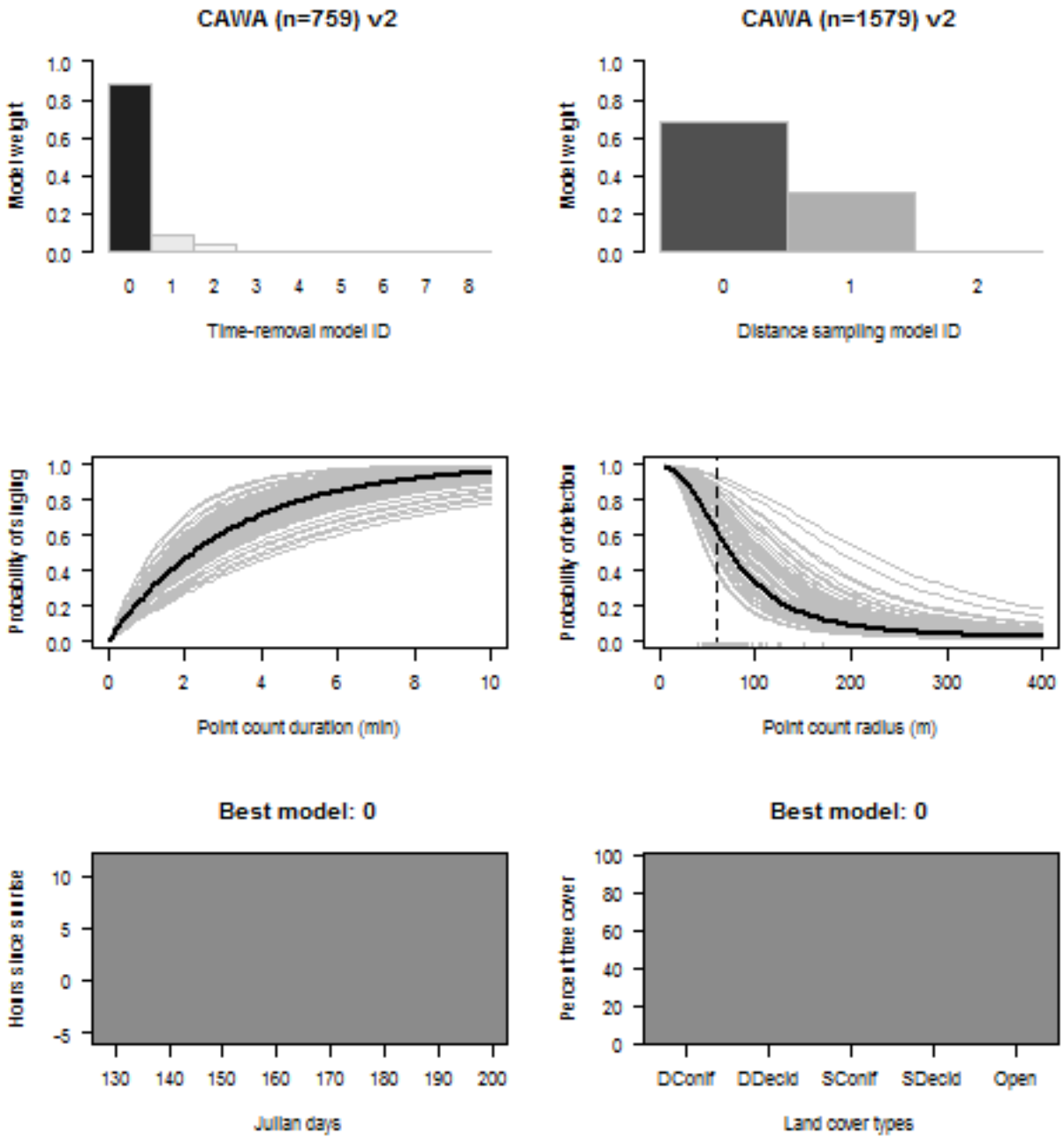


Here is how one can visualize the results for a given species (here Canada Warbler, `CAWA`) and a given version depending on what was specified as part of the `load_BAM_QPAD` call. (Note: loading a different version overwrites the estimates.)

```
load_BAM_QPAD(version = 2)
```

```
## BAM QPAD parameter estimates loaded, version 2
```

```
plot_BAM_QPAD("CAWA")
```



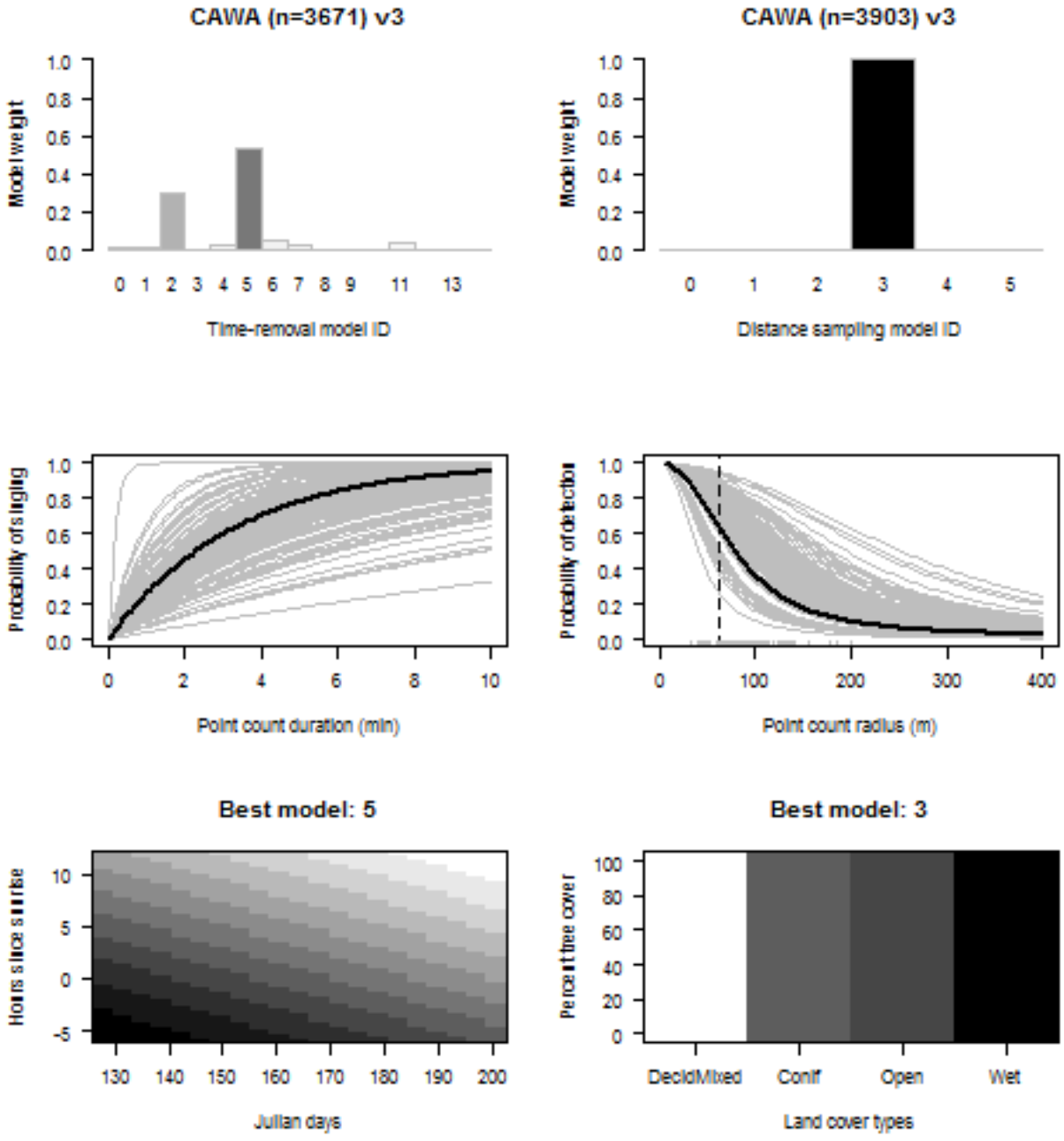
The plots show result summaries for time-removal models (left) and distance sampling (right). The top row shows model selection results for model candidates. The middle row depicts constant singing rate as a function of sampling duration (left) and constant detection distance based distance-decay function (right) (black line is for the actual species, grey lines indicate the other species analyzed). The bottom row shows the date/time relationship for probability of availability based on best supported time-removal model (left), and tree cover vs. land cover relationship based on best supported distance sampling model (right).

We can plot the updated v3 results for the species, to compare sample sizes, model support, general relationships between different versions (note: some species' results might not be available in v2). Sample sizes for time-removal and distance sampling can be different because the conditional maximum likelihood estimation uses different observations depending on methodologies (i.e. availability of multiple duration or distance intervals in a given survey).

```
load_BAM_QPAD(version = 3)
```

```
## BAM QPAD parameter estimates loaded, version 3
```

```
plot_BAM_QPAD("CAWA")
```



A general and through comparison of the results from the different versions is outside of the scope of this report, but it can be performed using the **QPAD** R package. The R functions and their usage described in the supporting information for Solymos et al. [1] are available as part of the **QPAD** package. Please refer to the Appendices of this report for technical details of parameter estimation and offset calculation.

Acknowledgements

This work is a contribution of the Boreal Avian Modelling (BAM) Project, an international research collaboration on the ecology, management, and conservation of boreal birds. We acknowledge the BAM Project's members, avian and biophysical data partners, and funding agencies (including Environment Canada and the U.S. Fish & Wildlife Service), listed in full at www.borealbirds.ca/index.php/acknowledgements.

References

[1] Solymos, P., Matsuoka, S. M., Bayne, E. M., Lele, S. R., Fontaine, P., Cumming, S. G., Stralberg, D., Schmiegelow, F. K. A. and Song, S. J., 2013. Calibrating indices of avian density from non-standardized survey data: making the most of a messy situation. *Methods in Ecology and Evolution* **4**:1047–1058.

[2] McKenney, D., Price, D., Papadapol, P., Siltanen, M., and Lawrence, K., 2006. High-resolution climate change scenarios for North America. Frontline Forestry Research Applications, Canadian Forest Service, Sault Ste. Marie, Technical Note No. 107. 5 pp.

[3] North American Land Change Monitoring System, URL http://landcover.usgs.gov/nalcms.php.

[4] QPAD R package: Calibrating indices of avian density from non-standardized survey data, GitHub repository: https://github.com/psolymos/QPAD.

Appendix A: R code used for estimation

The code provided here illustrates the process, but is not reproducible without the data itself. The code refers to tables in the `~/repos/bamanalitics/` directory which is equivalent to the GitHub repository `psolymos/bamanalytics`.

```
## Define root folder where data are stored
ROOT <- "c:/bam/May2015"
## Load required packages
library(mefa4)
library(pbapply)
library(detect)
## Load functions kept in separate file
source("~/repos/bamanalytics/R/dataprocessing_functions.R")
## Species table
sppt <- read.csv("~/repos/bamanalytics/lookup/singing-species.csv")
rownames(sppt) <- sppt$Species_ID
spp_singing <- sort(rownames(sppt)[sppt$Singing_birds])
## Load preprocesses data
load(file.path(ROOT, "out", "new_offset_data_package_2016-03-21.Rdata"))

### Removal sampling

## non NA subset for duration related estimates
pkDur <- dat[,c("PKEY","JDAY","TSSR","TSLS","DURMETH","YEAR","PCODE")]
pkDur$TSSR2 <- pkDur$TSSR^2
pkDur$JDAY2 <- pkDur$JDAY^2
pkDur$DSLS <- pkDur$TSLS
pkDur$DSLS2 <- pkDur$DSLS^2
## JDAY outliers
```

```
pkDur$JDAY[pkDur$JDAY > 0.55] <- NA
## strange methodology where all counts have been filtered
## thus this only leads to 0 total count and exclusion
pkDur <- droplevels(pkDur[pkDur$DURMETH != "J",])
pkDur <- droplevels(pkDur[rowSums(is.na(pkDur)) == 0,])
pkDur$TSLS <- NULL

## models to consider
NAMES <- list(
    "0"="(Intercept)",
    "1"=c("(Intercept)", "JDAY"),
    "2"=c("(Intercept)", "TSSR"),
    "3"=c("(Intercept)", "JDAY", "JDAY2"),
    "4"=c("(Intercept)", "TSSR", "TSSR2"),
    "5"=c("(Intercept)", "JDAY", "TSSR"),
    "6"=c("(Intercept)", "JDAY", "JDAY2", "TSSR"),
    "7"=c("(Intercept)", "JDAY", "TSSR", "TSSR2"),
    "8"=c("(Intercept)", "JDAY", "JDAY2", "TSSR", "TSSR2"),
    "9"=c("(Intercept)", "DSLS"),
    "10"=c("(Intercept)", "DSLS", "DSLS2"),
    "11"=c("(Intercept)", "DSLS", "TSSR"),
    "12"=c("(Intercept)", "DSLS", "DSLS2", "TSSR"),
    "13"=c("(Intercept)", "DSLS", "TSSR", "TSSR2"),
    "14"=c("(Intercept)", "DSLS", "DSLS2", "TSSR", "TSSR2"))
ff <- list(
    ~ 1,
    ~ JDAY,
    ~ TSSR,
    ~ JDAY + JDAY2,
    ~ TSSR + TSSR2,
    ~ JDAY + TSSR,
    ~ JDAY + JDAY2 + TSSR,
    ~ JDAY + TSSR + TSSR2,
    ~ JDAY + JDAY2 + TSSR + TSSR2,
    ~ DSLS,
    ~ DSLS + DSLS2,
    ~ DSLS + TSSR,
    ~ DSLS + DSLS2 + TSSR,
    ~ DSLS + TSSR + TSSR2,
    ~ DSLS + DSLS2 + TSSR + TSSR2)
names(ff) <- 0:14

## crosstab for species
xtDur <- Xtab(ABUND ~ PKEY + dur + SPECIES, pc)
#xtDur[["NONE"]] <- NULL
xtDur <- xtDur[spp_singing]

fitDurFun <- function(spp, fit=TRUE, type=c("rem","mix")) {
    rn <- intersect(rownames(pkDur), rownames(xtDur[[spp]]))
    X0 <- pkDur[rn,]
    Y0 <- as.matrix(xtDur[[spp]][rn,])
    ## make sure that columns (intervals) match up
    stopifnot(all(colnames(Y0) == colnames(ltdur$x)))
```

```
    stopifnot(length(setdiff(levels(X0$DURMETH), rownames(ltdur$end))) == 0)
    ## interval end matrix
    D <- ltdur$end[match(X0$DURMETH, rownames(ltdur$end)),]
    ## exclude 0 sum and <1 interval rows
    iob <- rowSums(Y0) > 0 & rowSums(!is.na(D)) > 1
    if (sum(iob)==0)
        return(structure("0 observation with multiple duration (1)",
            class="try-error"))
    if (sum(iob)==1)
        return(structure("1 observation with multiple duration (2)",
            class="try-error"))
    X <- droplevels(X0[iob,])
    Y0 <- Y0[iob,]
    D <- D[iob,]
    n <- nrow(D)
    ## arranging counts into position
    Yid <- ltdur$id[match(X$DURMETH, rownames(ltdur$id)),]
    Y <- matrix(NA, n, ncol(ltdur$id))
    for (i in seq_len(n)) {
        w <- Yid[i,]
        w <- w[!is.na(w)]
        Y[i,seq_len(length(w))] <- Y0[i,w]
    }
    if (fit) {
        res <- list()
        for (i in seq_len(length(ff))) {
            f <- as.formula(paste0("Y | D ", paste(as.character(ff[[i]]), collapse=" ")))
            mod <- try(cmulti(f, X, type=type))
            if (!inherits(mod, "try-error")) {
                rval <- mod[c("coefficients","vcov","nobs","loglik")]
                rval$p <- length(coef(mod))
                rval$names <- NAMES[[i]]
            } else {
                rval <- mod
            }
            res[[names(NAMES)[i]]] <- rval
        }
    } else {
        res <- list(Y=Y, D=D, n=n, pkey=rownames(X))
    }
    res
}

SPP <- names(xtDur)
resDur <- vector("list", length(SPP))
for (i in 1:length(SPP)) {
    cat("Singing rate data check for", SPP[i], "\n")
    flush.console()
    resDur[[i]] <- try(fitDurFun(SPP[i], FALSE))
}
names(resDur) <- SPP
resDurOK <- resDur[!sapply(resDur, inherits, "try-error")]
c(OK=length(resDurOK), failed=length(resDur)-length(resDurOK), all=length(resDur))
```

```
t(sapply(resDurOK, "[[", "n"))
resDurData <- resDurOK

## estimate species with data
SPP <- names(resDurOK)
resDur <- vector("list", length(SPP))
for (i in 1:length(SPP)) {
    cat("Singing rate estimation for", SPP[i], date(), "\n")
    flush.console()
    resDur[[i]] <- try(fitDurFun(SPP[i], TRUE, type="rem"))
}
names(resDur) <- SPP
resDurOK <- resDur[!sapply(resDur, inherits, "try-error")]
c(OK=length(resDurOK), failed=length(resDur)-length(resDurOK), all=length(resDur))
resDur <- resDurOK

save(resDur, resDurData,
    file=file.path(ROOT, "out", "estimates_SRA_QPAD_v2016.Rdata"))

### Distance sampling

## non NA subset for distance related estimates
pkDis <- dat[,c("PKEY","TREE","TREE3","HAB_NALC1","HAB_NALC2","DISMETH")]
pkDis <- droplevels(pkDis[rowSums(is.na(pkDis)) == 0,])
## strange methodology where all counts have been filtered
## thus this only leads to 0 total count and exclusion
pkDis <- droplevels(pkDis[pkDis$DISMETH != "W",])
pkDis$CTREE <- pkDis$TREE3

pkDis$WNALC <- pkDis$HAB_NALC2
levels(pkDis$WNALC)[levels(pkDis$WNALC) %in% c("Agr","Barren","Devel","Grass", "Shrub")] <- "Open"
pkDis$NALC <- pkDis$WNALC
levels(pkDis$NALC)[levels(pkDis$NALC) %in% c("Wet")] <- "Open"

pkDis$WNALCTREE <- pkDis$HAB_NALC1
levels(pkDis$WNALCTREE)[levels(pkDis$WNALCTREE) %in% c("Agr","Barren","Devel",
    "Grass", "Shrub","ConifOpen","DecidOpen","MixedOpen")] <- "Open"
pkDis$NALCTREE <- pkDis$WNALCTREE
levels(pkDis$NALCTREE)[levels(pkDis$NALCTREE) %in% c("Wet")] <- "Open"

pkDis$LCC2 <- as.factor(ifelse(pkDis$WNALC %in% c("Open", "Wet"), "OpenWet", "Forest"))
pkDis$LCC4 <- pkDis$WNALC
levels(pkDis$LCC4) <- c(levels(pkDis$LCC4), "DecidMixed")
pkDis$LCC4[pkDis$WNALC %in% c("Decid", "Mixed")] <- "DecidMixed"
pkDis$LCC4 <- droplevels(pkDis$LCC4)
pkDis$LCC4 <- relevel(pkDis$LCC4, "DecidMixed")

## models to consider

NAMES <- list(
    "0"="(Intercept)",
    "1"=c("(Intercept)", "TREE"),
    "2"=c("(Intercept)", "LCC2OpenWet"),
```

```
    "3"=c("(Intercept)", "LCC4Conif", "LCC4Open", "LCC4Wet"),
    "4"=c("(Intercept)", "LCC2OpenWet", "TREE"),
    "5"=c("(Intercept)", "LCC4Conif", "LCC4Open", "LCC4Wet", "TREE"))
ff <- list(
    ~ 1, # 0
    ~ TREE, # 1
    ~ LCC2, # 2
    ~ LCC4, # 3
    ~ LCC2 + TREE, # 4
    ~ LCC4 + TREE) # 5
names(ff) <- 0:5

## crosstab for species
xtDis <- Xtab(ABUND ~ PKEY + dis + SPECIES, pc)
#xtDis[["NONE"]] <- NULL
xtDis <- xtDis[spp_singing]

fitDisFun <- function(spp, fit=TRUE) {
    rn <- intersect(rownames(pkDis), rownames(xtDis[[spp]]))
    X0 <- pkDis[rn,]
    Y0 <- as.matrix(xtDis[[spp]][rn,])
    ## make sure that columns (intervals) match up
    stopifnot(all(colnames(Y0) == colnames(ltdis$x)))
    stopifnot(length(setdiff(levels(X0$DISMETH), rownames(ltdis$end))) == 0)
    ## interval end matrix: 100 m units
    D <- ltdis$end[match(X0$DISMETH, rownames(ltdis$end)),]
#    D <- D / 100 # use when not 100 m units
    ## exclude 0 sum and <1 interval rows
    iob <- rowSums(Y0) > 0 & rowSums(!is.na(D)) > 1
    if (sum(iob)==0)
        return(structure("0 observation with multiple duration (1)",
            class="try-error"))
    if (sum(iob)==1)
        return(structure("1 observation with multiple duration (2)",
            class="try-error"))
    X <- droplevels(X0[iob,])
    Y0 <- Y0[iob,]
    D <- D[iob,]
    n <- nrow(D)
    ## arranging counts into position
    Yid <- ltdis$id[match(X$DISMETH, rownames(ltdis$id)),]
    Y <- matrix(NA, n, ncol(ltdis$id))
    for (i in seq_len(n)) {
        w <- Yid[i,]
        w <- w[!is.na(w)]
        Y[i,seq_len(length(w))] <- Y0[i,w]
    }
    if (fit) {
        res <- list()
        for (i in seq_len(length(ff))) {
            f <- as.formula(paste0("Y | D ", paste(as.character(ff[[i]]), collapse=" ")))
            mod <- try(cmulti(f, X, type="dis"))
            if (!inherits(mod, "try-error")) {
```

```
                rval <- mod[c("coefficients","vcov","nobs","loglik")]
                rval$p <- length(coef(mod))
                rval$names <- NAMES[[i]]
            } else {
                rval <- mod
            }
            res[[names(NAMES)[i]]] <- rval
        }
    } else {
        res <- list(Y=Y, D=D, n=n, pkey=rownames(X))
    }
    res
}

SPP <- names(xtDis)
resDis <- vector("list", length(SPP))
for (i in 1:length(SPP)) {
    cat("EDR data check for", SPP[i], "\n")
    flush.console()
    resDis[[i]] <- try(fitDisFun(SPP[i], FALSE))
}
names(resDis) <- SPP
resDisOK <- resDis[!sapply(resDis, inherits, "try-error")]
c(OK=length(resDisOK), failed=length(resDis)-length(resDisOK), all=length(resDis))
t(sapply(resDisOK, "[[", "n"))
resDisData <- resDisOK

## estimate species with data
SPP <- names(resDisOK)
resDis <- vector("list", length(SPP))
for (i in 1:length(SPP)) {
    cat("EDR estimation for", SPP[i], date(), "\n")
    flush.console()
    resDis[[i]] <- try(fitDisFun(SPP[i], TRUE))
}
names(resDis) <- SPP
resDisOK <- resDis[!sapply(resDis, inherits, "try-error")]
c(OK=length(resDisOK), failed=length(resDis)-length(resDisOK), all=length(resDis))
resDis <- resDisOK

save(resDis, resDisData,
    file=file.path(ROOT, "out", "estimates_EDR_QPAD_v2016.Rdata"))

### Putting things together

## n.con is threshold above which the 0 constant model is considered
n.con <- 50#25
## n.min is threshold above which all models are considered
n.min <- 50#75

type <- "rem" # can use 'rem' or 'mix'

load(file.path(ROOT, "out", "estimates_SRA_QPAD_v2016.Rdata"))
```

```
load(file.path(ROOT, "out", "estimates_EDR_QPAD_v2016.Rdata"))
e <- new.env()
load(file.path(ROOT, "out", "new_offset_data_package_2016-03-21.Rdata"), envir=e)

## 0/1 table for successful model fit
edr_mod <- t(sapply(resDis, function(z)
    ifelse(sapply(z, inherits, what="try-error"), 0L, 1L)))
sra_mod <- t(sapply(resDur, function(z)
    ifelse(sapply(z, inherits, what="try-error"), 0L, 1L)))
tmp <- union(rownames(edr_mod), rownames(sra_mod))
edr_models <- matrix(0L, length(tmp), ncol(edr_mod))
dimnames(edr_models) <- list(tmp, colnames(edr_mod))
edr_models[rownames(edr_mod),] <- edr_mod
sra_models <- matrix(0L, length(tmp), ncol(sra_mod))
dimnames(sra_models) <- list(tmp, colnames(sra_mod))
sra_models[rownames(sra_mod),] <- sra_mod
## data deficient cases: dropped factor levels
## need to check length of NAMES and length of COEF --> correct factor level
## designation is possible, if != bump up AIC/BIC
for (spp in rownames(sra_mod)) {
    for (mid in colnames(sra_models)) {
        if (!inherits(resDur[[spp]][[mid]], "try-error")) {
            if (type == "rem")
                lcf <- length(resDur[[spp]][[mid]]$coefficients)
            if (type == "mix")
                lcf <- length(resDur[[spp]][[mid]]$coefficients) - 1
            lnm <- length(resDur[[spp]][[mid]]$names)
            if (lcf != lnm) {
                cat("SRA conflict for", spp, "model", mid,
                "( len.coef =", lcf, ", len.name =", lnm, ")\n")
                sra_models[spp,mid] <- 0
            }
        } else {
            resDur[[spp]][[mid]] <- structure("Error", class = "try-error")
            #attributes(resDur[[spp]][[mid]]) <- NULL
            #class(resDur[[spp]][[mid]]) <- "try-error"
        }
    flush.console()
    }
}
for (spp in rownames(edr_mod)) {
    ## data for checking detections in classes
    Dat <- pkDis[resDisData[[spp]]$pkey,c("LCC2","LCC4")]
    for (mid in colnames(edr_models)) {
        if (!inherits(resDis[[spp]][[mid]], "try-error")) {
            lcf <- length(resDis[[spp]][[mid]]$coefficients)
            lnm <- length(resDis[[spp]][[mid]]$names)
            if (lcf != lnm) {
                cat("EDR conflict for", spp, "model", mid,
                "( len.coef =", lcf, ", len.name =", lnm, ")\n")
                edr_models[spp,mid] <- 0
            } else {
                if (mid %in% c("2", "4") && min(table(Dat$LCC2)) < 5) {
```

```
                    cat("EDR LCC2 min issue for", spp, "model", mid, "\n")
                    edr_models[spp,mid] <- 0
                }
                if (mid %in% c("3", "5") && min(table(Dat$LCC4)) < 5) {
                    cat("EDR LCC4 min issue for", spp, "model", mid, "\n")
                    edr_models[spp,mid] <- 0
                }
                if (mid %in% c("1", "4", "5") &&
                    resDis[[spp]][[mid]]$coefficients["log.tau_TREE"] > 0) {
                    cat("EDR TREE > 0 issue for", spp, "model", mid, "\n")
                    edr_models[spp,mid] <- 0
                }
            }
        } else {
            resDis[[spp]][[mid]] <- structure("Error", class = "try-error")
            attributes(resDis[[spp]][[mid]]) <- NULL
            class(resDis[[spp]][[mid]]) <- "try-error"
        }
    flush.console()
    }
}
## no constant model means exclude that species
edr_models[edr_models[,1]==0,] <- 0L
sra_models[sra_models[,1]==0,] <- 0L

phi0 <- sapply(resDur, function(z) exp(z[["0"]]$coef))
names(phi0) <- names(resDur)
sra_models[names(phi0)[phi0 < 0.01],] <- 0L

sum(edr_models)
sum(sra_models)

colSums(edr_models)
colSums(sra_models)

edr_nmod <- ncol(edr_mod)
sra_nmod <- ncol(sra_mod)

## sample sizes
edr_n <- sra_n <- numeric(length(tmp))
names(edr_n) <- names(sra_n) <- tmp
edr_nn <- sapply(resDis, function(z) ifelse(inherits(z[["0"]], "try-error"),
    NA, z[["0"]]$nobs))
edr_n[names(edr_nn)] <- edr_nn
sra_nn <- sapply(resDur, function(z) ifelse(inherits(z[["0"]], "try-error"),
    NA, z[["0"]]$nobs))
sra_n[names(sra_nn)] <- sra_nn

## exclude all models for species with < n.con observations
sra_models[sra_n < n.con, ] <- 0L
edr_models[edr_n < n.con, ] <- 0L
## exclude non-constant model for species with n.con < observations < n.min
sra_models[sra_n < n.min & sra_n >= n.con, 2:ncol(sra_models)] <- 0L
```

```
edr_models[edr_n < n.min & edr_n >= n.con, 2:ncol(edr_models)] <- 0L
table(rowSums(sra_models))
table(rowSums(edr_models))
table(rowSums(sra_models), rowSums(edr_models))

## spp to keep
spp <- tmp[rowSums(edr_models) > 0 & rowSums(sra_models) > 0]
length(spp)

edr_models <- edr_models[spp,]
sra_models <- sra_models[spp,]
edr_n <- edr_n[spp]
sra_n <- sra_n[spp]

## no. of parameters
edr_df <- sapply(resDis[["OVEN"]][1:edr_nmod], "[[", "p")
sra_df <- sapply(resDur[["OVEN"]][1:sra_nmod], "[[", "p")

## estimates
edr_estimates <- resDis[spp]
sra_estimates <- resDur[spp]

## species table
tax <- e$TAX
tax <- tax[!duplicated(tax$Species_ID),]
rownames(tax) <- tax$Species_ID
spp_table <- data.frame(spp=spp,
    scientific_name=tax[spp, "Scientific_Name"],
    common_name=tax[spp, "English_Name"])
rownames(spp_table) <- spp
spp_table <- droplevels(spp_table)

## get variable names for different models
sra_list <- sapply(sra_estimates[["OVEN"]], function(z) paste(z$names, collapse=" + "))
edr_list <- sapply(edr_estimates[["OVEN"]], function(z) paste(z$names, collapse=" + "))

## loglik values
sra_loglik <- sra_models
sra_loglik[] <- -Inf
for (i in spp) { # species
    for (j in 1:sra_nmod) { # models
        if (sra_models[i,j] > 0)
            sra_loglik[i,j] <- resDur[[i]][[j]]$loglik
    }
}
edr_loglik <- edr_models
edr_loglik[] <- -Inf
for (i in spp) { # species
    for (j in 1:edr_nmod) { # models
        if (edr_models[i,j] > 0)
            edr_loglik[i,j] <- resDis[[i]][[j]]$loglik
    }
}
```

```
## AIC/BIC
sra_aic <- sra_aicc <- sra_bic <- sra_loglik
sra_aic[] <- Inf
sra_aicc[] <- Inf
sra_bic[] <- Inf
edr_aic <- edr_aicc <- edr_bic <- edr_loglik
edr_aic[] <- Inf
edr_aicc[] <- Inf
edr_bic[] <- Inf
for (i in spp) {
    sra_aic[i,] <- -2*sra_loglik[i,] + 2*sra_df
    sra_aicc[i,] <- sra_aic[i,] + (2*sra_df*(sra_df+1)) / (sra_n[i]-sra_df-1)
    sra_bic[i,] <- -2*sra_loglik[i,] + log(sra_n[i])*sra_df
    edr_aic[i,] <- -2*edr_loglik[i,] + 2*edr_df
    edr_aicc[i,] <- edr_aic[i,] + (2*edr_df*(edr_df+1)) / (edr_n[i]-edr_df-1)
    edr_bic[i,] <- -2*edr_loglik[i,] + log(edr_n[i])*edr_df
}

## model ranking
sra_aicrank <- t(apply(sra_aic, 1, rank))*sra_models
sra_aicrank[sra_aicrank==0] <- NA
edr_aicrank <- t(apply(edr_aic, 1, rank))*edr_models
edr_aicrank[edr_aicrank==0] <- NA

sra_aiccrank <- t(apply(sra_aicc, 1, rank))*sra_models
sra_aiccrank[sra_aiccrank==0] <- NA
edr_aiccrank <- t(apply(edr_aicc, 1, rank))*edr_models
edr_aiccrank[edr_aiccrank==0] <- NA

sra_bicrank <- t(apply(sra_bic, 1, rank))*sra_models
sra_bicrank[sra_bicrank==0] <- NA
edr_bicrank <- t(apply(edr_bic, 1, rank))*edr_models
edr_bicrank[edr_bicrank==0] <- NA

sra_aicbest <- apply(sra_aicrank, 1, function(z) colnames(sra_models)[which.min(z)])
sra_aiccbest <- apply(sra_aiccrank, 1, function(z) colnames(sra_models)[which.min(z)])
sra_bicbest <- apply(sra_bicrank, 1, function(z) colnames(sra_models)[which.min(z)])
edr_aicbest <- apply(edr_aicrank, 1, function(z) colnames(edr_models)[which.min(z)])
edr_aiccbest <- apply(edr_aiccrank, 1, function(z) colnames(edr_models)[which.min(z)])
edr_bicbest <- apply(edr_bicrank, 1, function(z) colnames(edr_models)[which.min(z)])

table(edr_aicbest, sra_aicbest)
rowSums(table(edr_aicbest, sra_aicbest))
colSums(table(edr_aicbest, sra_aicbest))

table(edr_aiccbest, sra_aiccbest)
rowSums(table(edr_aiccbest, sra_aiccbest))
colSums(table(edr_aiccbest, sra_aiccbest))

table(edr_bicbest, sra_bicbest)
rowSums(table(edr_bicbest, sra_bicbest))
colSums(table(edr_bicbest, sra_bicbest))
```

```
version <- "3"

bamcoefs <- list(spp=spp,
    spp_table=spp_table,
    edr_list=edr_list,
    sra_list=sra_list,
    edr_models=edr_models,
    sra_models=sra_models,
    edr_n=edr_n,
    sra_n=sra_n,
    edr_df=edr_df,
    sra_df=sra_df,
    edr_loglik=edr_loglik,
    sra_loglik=sra_loglik,
    edr_aic=edr_aic,
    sra_aic=sra_aic,
    edr_aicc=edr_aicc,
    sra_aicc=sra_aicc,
    edr_bic=edr_bic,
    sra_bic=sra_bic,
    edr_aicrank=edr_aicrank,
    sra_aicrank=sra_aicrank,
    edr_aiccrank=edr_aiccrank,
    sra_aiccrank=sra_aiccrank,
    edr_bicrank=edr_bicrank,
    sra_bicrank=sra_bicrank,
    edr_aicbest=edr_aicbest,
    sra_aicbest=sra_aicbest,
    edr_aiccbest=edr_aiccbest,
    sra_aiccbest=sra_aiccbest,
    edr_bicbest=edr_bicbest,
    sra_bicbest=sra_bicbest,
    edr_estimates=edr_estimates,
    sra_estimates=sra_estimates,
    version=version)
.BAMCOEFS <- list2env(bamcoefs)

save(.BAMCOEFS, file=file.path(ROOT, "out", "BAMCOEFS_QPAD_v3.rda"))
toDump <- as.list(.BAMCOEFS)
dump("toDump", file=file.path(ROOT, "out", "BAMCOEFS_QPAD_v3.Rdump"))
```

Appendix B: R code for offset calculations

The functions introduced in the supporting information of Solymos et al. [1] are still available as part of the **QPAD** R package. Here we provide an alternative and more transparent algorithm for calculating statistical offsets (that are the log transformed version of correction factors).

```
ROOT <- "e:/peter/bam/Apr2016"
library(mefa4)
library(QPAD)
source("~/repos/bamanalytics/R/dataprocessing_functions.R")
```

```
load(file.path(ROOT, "out", paste0("data_package_2016-04-18.Rdata")))

load_BAM_QPAD(3)
getBAMversion()
sppp <- getBAMspecieslist()

offdat <- data.frame(PKEY[,c("PCODE","PKEY","SS","TSSR","JDAY","MAXDUR","MAXDIS","JULIAN")],
    SS[match(PKEY$SS, rownames(SS)),c("TREE","TREE3","HAB_NALC1","HAB_NALC2", "SPRNG")])
offdat$srise <- PKEY$srise + PKEY$MDT_offset
offdat$DSLS <- (offdat$JULIAN - offdat$SPRNG) / 365
summary(offdat)

offdat$JDAY2 <- offdat$JDAY^2
offdat$TSSR2 <- offdat$TSSR^2
offdat$DSLS2 <- offdat$DSLS^2
offdat$LCC4 <- as.character(offdat$HAB_NALC2)
offdat$LCC4[offdat$LCC4 %in% c("Decid", "Mixed")] <- "DecidMixed"
offdat$LCC4[offdat$LCC4 %in% c("Agr","Barren","Devel","Grass", "Shrub")] <- "Open"
offdat$LCC4 <- factor(offdat$LCC4,
        c("DecidMixed", "Conif", "Open", "Wet"))
offdat$LCC2 <- as.character(offdat$LCC4)
offdat$LCC2[offdat$LCC2 %in% c("DecidMixed", "Conif")] <- "Forest"
offdat$LCC2[offdat$LCC2 %in% c("Open", "Wet")] <- "OpenWet"
offdat$LCC2 <- factor(offdat$LCC2, c("Forest", "OpenWet"))
table(offdat$LCC4, offdat$LCC2)
offdat$MAXDIS <- offdat$MAXDIS / 100

Xp <- cbind("(Intercept)"=1, as.matrix(offdat[,c("TSSR","JDAY","DSLS","TSSR2","JDAY2","DSLS2")]))
Xq <- cbind("(Intercept)"=1, TREE=offdat$TREE,
    LCC2OpenWet=ifelse(offdat$LCC2=="OpenWet", 1, 0),
    LCC4Conif=ifelse(offdat$LCC4=="Conif", 1, 0),
    LCC4Open=ifelse(offdat$LCC4=="Open", 1, 0),
    LCC4Wet=ifelse(offdat$LCC4=="Wet", 1, 0))

OFF <- matrix(NA, nrow(offdat), length(sppp))
rownames(OFF) <- offdat$PKEY
colnames(OFF) <- sppp

#spp <- "OVEN"
for (spp in sppp) {
cat(spp, "\n");flush.console()
p <- rep(NA, nrow(offdat))
A <- q <- p

## constant for NA cases
cf0 <- exp(unlist(coefBAMspecies(spp, 0, 0)))
## best model
mi <- bestmodelBAMspecies(spp, type="BIC")
cfi <- coefBAMspecies(spp, mi$sra, mi$edr)

Xp2 <- Xp[,names(cfi$sra),drop=FALSE]
OKp <- rowSums(is.na(Xp2)) == 0
Xq2 <- Xq[,names(cfi$edr),drop=FALSE]
```

```
OKq <- rowSums(is.na(Xq2)) == 0

p[!OKp] <- sra_fun(offdat$MAXDUR[!OKp], cf0[1])
unlim <- ifelse(offdat$MAXDIS[!OKq] == Inf, TRUE, FALSE)
A[!OKq] <- ifelse(unlim, pi * cf0[2]^2, pi * offdat$MAXDIS[!OKq]^2)
q[!OKq] <- ifelse(unlim, 1, edr_fun(offdat$MAXDIS[!OKq], cf0[2]))

phi1 <- exp(drop(Xp2[OKp,,drop=FALSE] %*% cfi$sra))
tau1 <- exp(drop(Xq2[OKq,,drop=FALSE] %*% cfi$edr))
p[OKp] <- sra_fun(offdat$MAXDUR[OKp], phi1)
unlim <- ifelse(offdat$MAXDIS[OKq] == Inf, TRUE, FALSE)
A[OKq] <- ifelse(unlim, pi * tau1^2, pi * offdat$MAXDIS[OKq]^2)
q[OKq] <- ifelse(unlim, 1, edr_fun(offdat$MAXDIS[OKq], tau1))

ii <- which(p == 0)
p[ii] <- sra_fun(offdat$MAXDUR[ii], cf0[1])

OFF[,spp] <- log(p) + log(A) + log(q)

}

## checks
(Ra <- apply(OFF, 2, range))
summary(t(Ra))
which(!is.finite(Ra[1,]))
which(!is.finite(Ra[2,]))

## save offsets table and covariates used
SPP <- sppp
save(OFF, SPP,
    file=file.path(ROOT, "out", "offsets-v3_2016-04-18.Rdata"))
offdat <- offdat[,c("PKEY","TSSR","JDAY","DSLS","TREE","LCC4","MAXDUR","MAXDIS")]
save(offdat,
    file=file.path(ROOT, "out", "offsets-v3data_2016-04-18.Rdata"))
```