Patagonia parrots density analysis

Francisco Dénes and Peter Sólymos

June 8, 2018

Motivation

We want to model the density of the austral parakeet *Enicognathus ferrugineus* and the slender-billed parakeet *Enicognathus leptorhynchus* from count data obtained with road transect surveys in Patagonia. Because these parrots are gregarious, we want to assess whether group size and size of groups vary across habitats (classified as 'urban', 'agropastoral' and 'other' (i.e. various natural forest formations), and between breeding (Nov-Dec) and non-breeding seasons (Jan-Oct).
We use distance sampling methods to account for detectability of parrots.

These animals tend to form smaller or larger groups, but groups of size 2 are also often observed as a result of mating pairs, as shown in Figure 1.

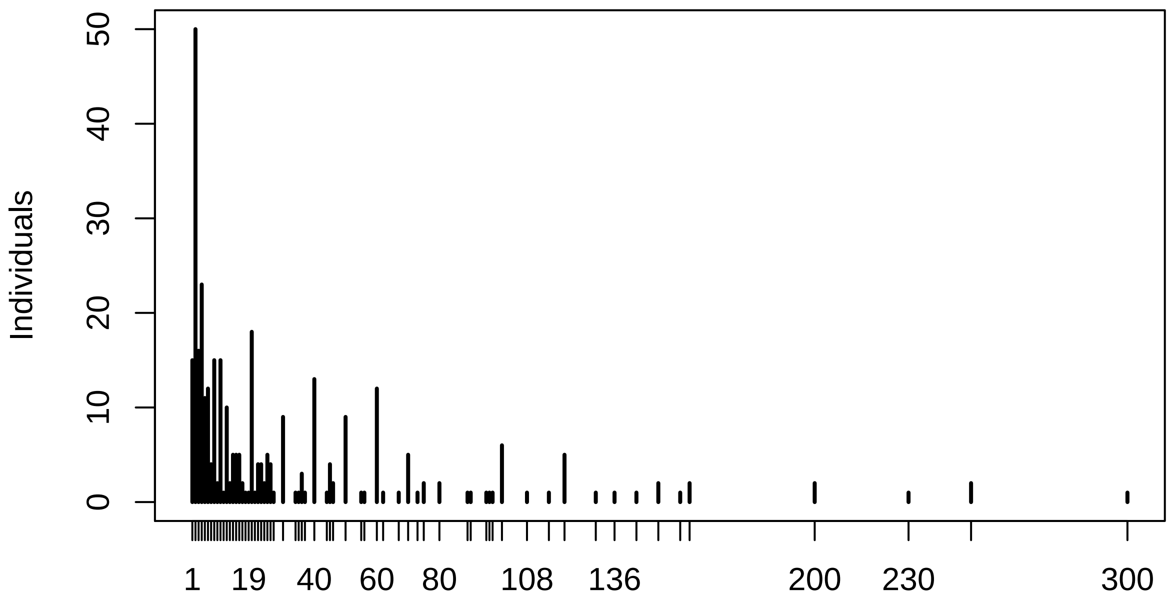

Figure 1: *Enicognathus ferrugineus* count frequencies

Austral parakeet *Enicognathus ferrugineus*

Estimating effective detection radius (EDR)

We fit distance sampling models with detection predictor variables, including the average group size, the number of groups that were observed (to account for the possibility that grouping behaviour influences detectability), and also habitat type. We used forward-stepwise model selection, starting with single covariate models and eliminating covariates that do not improve model parsimony (i.e. result in dAIC < 2) in relation to the null model.

Table 1: *E. ferrugineous* EDR models AIC

	df	AIC	dAIC
EDR.habitatype	3	1434.532	0.00
EDR.null	1	1438.166	3.63
EDR.avggroupsize	2	1439.144	4.61
EDR.numbergroups	2	1440.166	5.63

The model (`EDR.habitatype`) has the lowest AIC (Table 1), indicating that habitat type affects the effective detection radius (EDR) of *Enicognathus ferrugineus.* Specifically, detection radius is higher in agropastoral than urban and other habitats (Table 2). This means that detectability is higher in agropastoral habitats, and also that the area surveyed in a given sample unit is larger if the habitat therein is agropastoral vs. urban or other, presubaply because it is possible to see further in pastures and planted fields than in forest or urban environments.

Table 2: *E. ferrugineous* top-ranked EDR model estimates

	Estimate	Std. Error	z value	Pr(>|z|)
log.tau_(Intercept)	4.589	0.038	119.365	0.000
log.tau_Urban	0.057	0.058	0.988	0.323
log.tau_Agropastoral	0.263	0.103	2.556	0.011

Table 3: *E. ferrugineous* habitat-specific EDR (m)

Habitat	EDR
Other	98.36847
Urban	104.18432
Agropastoral	127.92056

Models for number of groups

We then model the number of groups as a function of covariates. The model for number of groups is $G_i \sim \text{Poisson}(D_i A_i)$, where D_i = covariates and A_i = area sampled in site *i*. A_i is calculated using the habitat-specific estimated EDR, and is added to the model as an offset.

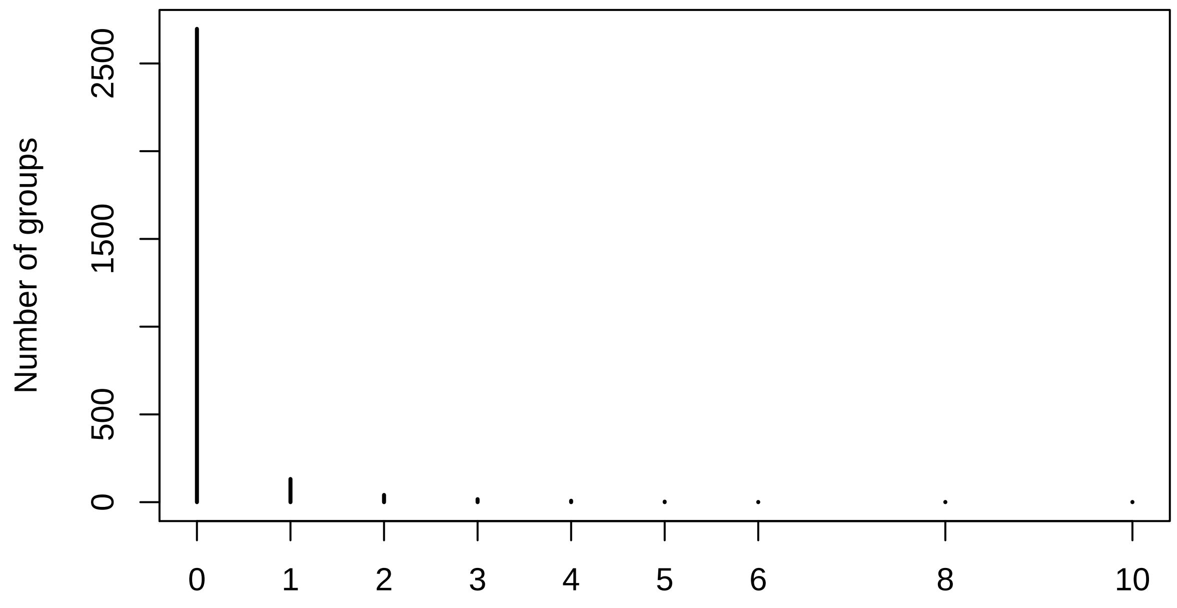

Figure 2: *Enicognathus ferrugineus* group numbers

Model selection

We build a first set of models to evaluate the effect of covariates related to habitat (habitat type and elevation).

Table 4: *E. ferrugineous* number of group models

	df	AIC	dAIC
ngroup.hab.ele2	5	2082.963	0.00
ngroup.hab.ele	4	2114.437	31.47
ngroup.hab	3	2133.717	50.75
ngroup.ele2	3	2275.565	192.60
ngroup.ele	2	2323.806	240.84

The model with both habitat type and elevation (quadratic effect) has the lowest AIC (Table 4), indicating that both covariates affect the number of groups:

Table 5: *E. ferrugineous* 'ngroup.hab.ele2' model estimates

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-3.108	0.233	-13.356	0

	Estimate	Std. Error	z value	Pr(>|z|)
habitatOther	1.507	0.202	7.455	0
habitatUrban	2.390	0.202	11.823	0
elevation	-0.004	0.001	-7.032	0
I(elevation^2)	0.000	0.000	5.947	0

Table 6: Deviance partitioning of 'ngroup.hab.ele2' model for *E. ferrugineous*

	Df	Deviance	Resid. Df	Resid. Dev
NULL	NA	NA	2900	1865.338
habitat	2	207.965	2898	1657.373
elevation	1	21.279	2897	1636.094
I(elevation^2)	1	33.475	2896	1602.619

We then proceed by adding within-year temporal covariates (breeding/non-breeding season and julian date) and their interactions with habitat.

Table 7: *E. ferrugineous* number of group models (within-year temporal predictors) AIC

	df	AIC	dAIC
ngroup.habXseason.ele2	8	2065.594	0.00
ngroup.hab.season.ele2	6	2067.555	1.96
ngroup.habXjdate.ele2	8	2081.546	15.95
ngroup.hab.ele2	5	2082.963	17.37
ngroup.hab.jdate.ele2	6	2084.491	18.90

The model with the season*habitat interaction (Table 8) is equally parcimonious with the model with only the additive effects (Table 9). Given these are nested models, this indicates the interaction only marginally improves the model fit, so will continue with model with season and no interaction:

Table 8: *E. ferrugineous* 'ngroup.habXseason.ele2' interaction model estimates

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-18.187	558.012	-0.033	0.974
elevation	-0.004	0.001	-6.547	0.000
I(elevation^2)	0.000	0.000	5.500	0.000
habitatOther	15.797	558.012	0.028	0.977
habitatUrban	2.008	908.375	0.002	0.998
seasonnon-breeding	15.186	558.012	0.027	0.978
habitatOther:seasonnon-breeding	-14.357	558.012	-0.026	0.979
habitatUrban:seasonnon-breeding	0.266	908.375	0.000	1.000

Table 9: *E. ferrugineous* ‘ngroup.hab.season.ele2’ additive model estimates

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-4.250	0.412	-10.318	0.000
habitatOther	1.510	0.204	7.411	0.000
habitatUrban	2.314	0.203	11.424	0.000
elevation	-0.004	0.001	-6.408	0.000
I(elevation^2)	0.000	0.000	5.365	0.000
seasonnon-breeding	1.175	0.340	3.451	0.001

Finally, we assess year effects by adding a year covariate (2013-2016).

Table 10: *E. ferrugineous* number of group models (year predictor) AIC table

	df	AIC	dAIC
ngroup.hab.season.ele2.year	9	2022.265	0.00
ngroup.hab.season.ele2	6	2067.555	45.29

The model with lowest AIC indicates that the number of groups is affected by habitat ype, elevation, season (breeding/non breeding) and year.

Table 11: *E. ferrugineous* top-ranked (ngroup.hab.season.ele2.year) model estimates

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-4.716	0.417	-11.311	0.000
habitatOther	1.739	0.206	8.434	0.000
habitatUrban	2.460	0.204	12.075	0.000
elevation	-0.003	0.001	-5.712	0.000
I(elevation^2)	0.000	0.000	4.588	0.000
seasonnon-breeding	1.022	0.348	2.939	0.003
as.factor(year)2014	0.092	0.168	0.549	0.583
as.factor(year)2015	0.918	0.161	5.688	0.000
as.factor(year)2016	0.612	0.205	2.979	0.003

Table 12: Deviance partitioning of top-ranked ‘ngroup.hab.season.ele2.year’ model for *E. ferrugineous*

	Df	Deviance	Resid. Df	Resid. Dev
NULL	NA	NA	2900	1865.338
habitat	2	207.965	2898	1657.373
elevation	1	21.279	2897	1636.094
I(elevation^2)	1	33.475	2896	1602.619
season	1	17.408	2895	1585.211
as.factor(year)	3	51.290	2892	1533.921

Models for group size

The count distribution is characterized with a spike at 2 (Figure 1), and by the absence of 0s due to group size being conditional on having >0 birds to consider it a group.

We start developing a general V-Inflated Poisson (VIP) model ("V" stands for "variable"", in refence to the"Z" representing 0 in a zero-inflated Poisson model, or ZIP), then we add the >0 condition.

Maximum likelihood

Let Y be a random variable, and y are observations, V is the count value that has some extra probability mass ($V = 0$ is the ZIP model), $f(y; \lambda)$ is the Poisson density ($f(y; \lambda) = e^{-\lambda} \frac{\lambda^y}{y!}$).

The V-Inflated density can be written as $P(Y = y) = \phi I(Y = V) + (1 - \phi) f(y; \lambda)$ which is $\phi + (1 - \phi) f(V; \lambda)$ when $Y = V$ and $(1 - \phi) f(y; \lambda)$ otherwise.

R functions for the VIP model are presented at the end of this document. Simulations are done to check the estimating procedure.

We define the extra probability mass at `V=2` to account for the group size peak in pairs. The model is also truncated at 0, because there are no groups with 0 individuals. We include habitat type as a covariate for the count (Poisson) component, and compare models with and without season (breeding/non-breeding) as a covariate for the V-inflation probability.

Table 13: *E. ferrugineous* group size model estimates

	Estimate	Std. Error	z value	Pr(>|z|)
P_(Intercept)	1.000	0.019	52.686	0.000
P_Urban	1.754	0.023	76.483	0.000
P_Agropastoral	0.612	0.037	16.744	0.000
V_(Intercept)	1.316	0.816	1.612	0.107
V_seasonnon-breeding	-3.474	0.839	-4.142	0.000

Estimated model coefficients (Table 13) indicate that group sizes are larger in urban ($\beta = 1.754$) and agropastoral ($\beta = 1.754$) habitats when compared to other natural habitats. Moreover, the probability of there being extra pairs (i.e. the V-inflation probability, ϕ) is smaller in the non-breeding season ($\beta = 1.754$).

Alternatively, we can estimate confidence intervals based on quantiles using bootstrap samples (with n=30) for the estimated coefficients (Table 14).

Table 14: Estimated coefficients and 90% CI based on bootstrap sample (n= 30) quantiles for *E. ferrugineous* group size models

	Estimate	5%	95%
P_(Intercept)	1.000	-10.681	18.873
P_Urban	1.754	-22.459	21.211
P_Agropastoral	0.612	-17.279	20.264
V_(Intercept)	1.316	-7.805	20.915
V_seasonnon-breeding	-3.474	-11.655	21.395

Slender-billed parakeet *Enicognathus leptorhynchus*

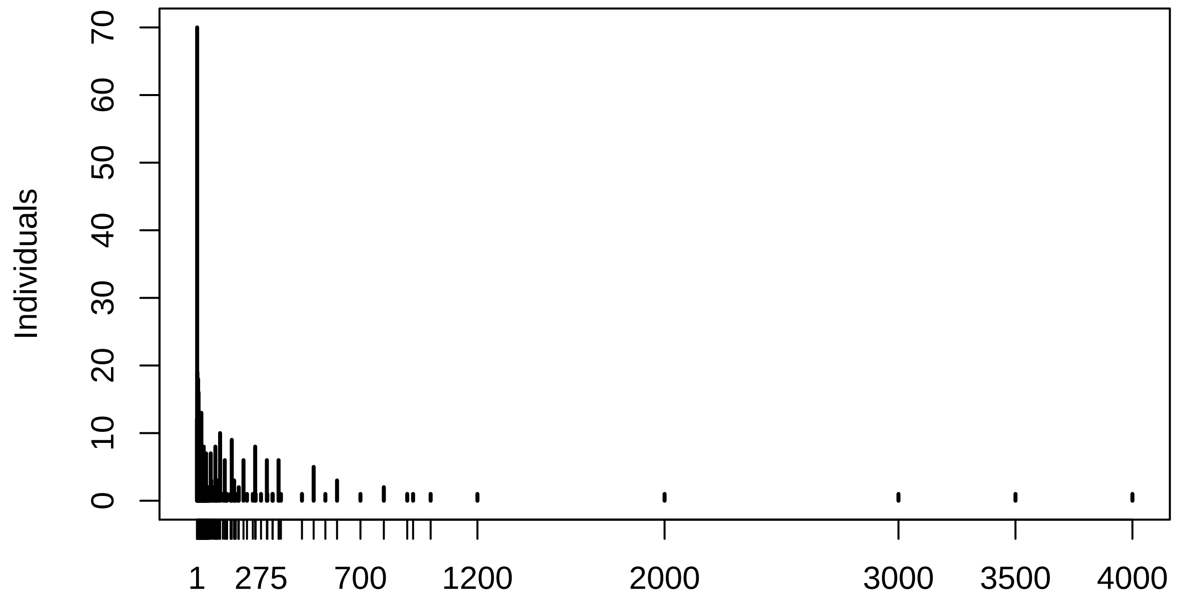

Figure 3: *Enicognathus leptorhynchus* count frequencies

Estimating effective detection radius (EDR)

Table 15: *E. leptorhynchus* EDR models AIC

	df	AIC	dAIC
EDR.avggroupsize.habitat	4	1054.464	0.00
EDR.habitat.avggroupsize.numbergroups	5	1059.062	4.60
EDR.avggroupsize	2	1063.875	9.41
EDR.avggroupsize.numbergroups	3	1064.553	10.09
EDR.habitat.numbergroups	4	1069.376	14.91
EDR.habitatype	3	1070.755	16.29
EDR.null	1	1086.330	31.87
EDR.numbergroups	2	1087.172	32.71

The model (`EDR.avggroupsize.habitat`) has the lowest AIC (Table 7), indicating that habitat type and average group size affect the effective detection radius (EDR) of *Enicognathus leptorhynchus*. The mean EDR for each habitat, predicted using model coefficients and the habitat-specific means of average group size, is shown in Table 16.

Table 16: *E. leptorhynchus* top-ranked EDR model estimates

	Estimate	Std. Error	z value	Pr(>|z|)
log.tau_(Intercept)	5.361	0	1.273654e+51	0
log.tau_gavg	0.001	0	9.610881e+50	0
log.tau_Urban	-0.266	0	-6.308165e+49	0
log.tau_Agropastoral	-0.029	0	-4.843080e+48	0

Table 17: *E. leptorhynchus* habitat-specific mean EDR (m)

Habitat	EDR
Other	235.4183
Urban	224.2124
Agropastoral	180.7002

Models for number of groups

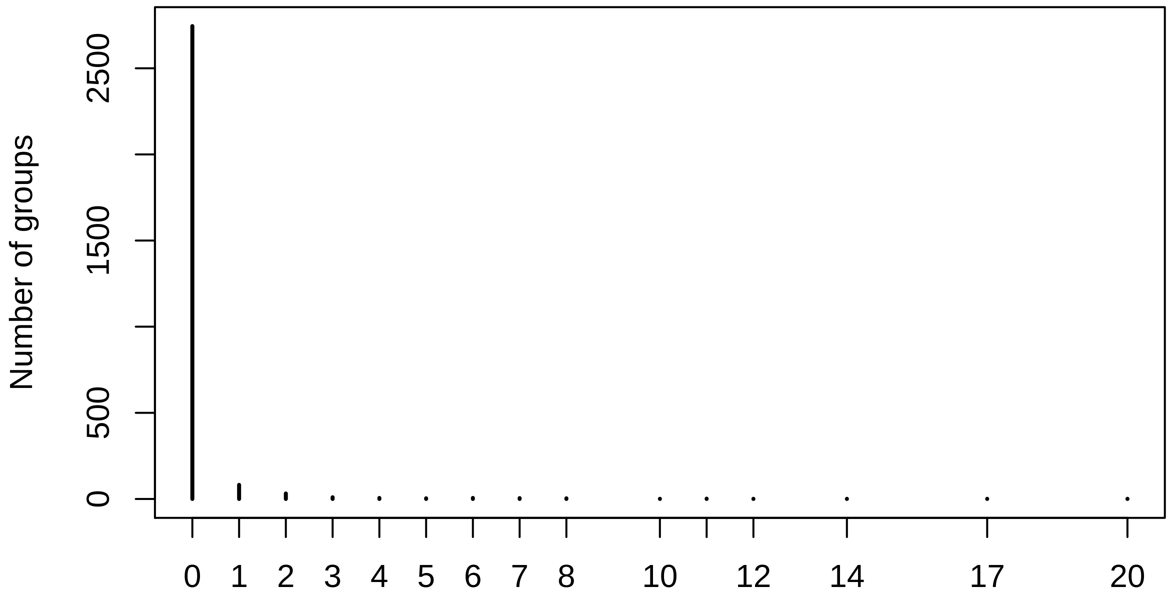

Figure 4: *Enicognathus leptorhynchus* group numbers

Model selection

First set of models to evaluate the effect of habitat type and elevation covariates (Table 18):

Table 18: *E. leptorhynchus* number of group models

	df	AIC	dAIC
ngroup.hab.ele2	5	2486.990	-0.84
ngroup.hab.ele	4	2487.834	0.00
ngroup.hab	3	2493.346	5.51
ngroup.ele2	3	2746.031	258.20
ngroup.ele	2	2761.122	273.29

The models 'ngroup.hab.ele' (linear elevation effect) and 'ngroup.hab.ele' (quadratic effect) are equally parsimonious (Table 18). Given their nestedness, we drop the quadratic term and continue with the model with habitat and linear elevation effects (Tables 19-20).

Table 19: *E. leptorhynchus* 'ngroup.hab.ele' model estimates

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-2.781	0.123	-22.568	0.000
habitatOther	-2.035	0.147	-13.874	0.000
habitatUrban	-0.734	0.143	-5.116	0.000
elevation	0.000	0.000	2.736	0.006

Table 20: Deviance partitioning of 'ngroup.hab.ele' model for *E. leptorhynchus*

	Df	Deviance	Resid. Df	Resid. Dev
NULL	NA	NA	2900	2358.856
habitat	2	270.326	2898	2088.530
elevation	1	7.512	2897	2081.018

Adding within-year temporal covariates (breeding/non-breeding season and julian date) and their interactions with habitat:

Table 21: *E. leptorhynchus* number of group models (within-year temporal predictors) AIC

	df	AIC	dAIC
ngroup.habXjdate.ele	7	2110.653	0.00
ngroup.hab.ele.jdate	5	2230.345	119.69
ngroup.habXseason.ele	7	2472.731	362.08
ngroup.hab.ele	4	2487.834	377.18
ngroup.hab.ele.season	5	2489.059	378.41

The model 'ngroup.habXjdate.ele' has the lowest AIC (Table 21).

Table 22: *E. leptorhynchus* 'ngroup.habXjdate.ele' model estimates

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-1.463	0.167	-8.741	0.000
elevation	0.000	0.000	1.939	0.053
habitatOther	2.234	0.590	3.788	0.000
habitatUrban	0.412	0.310	1.329	0.184
jdate	-0.006	0.001	-10.029	0.000
habitatOther:jdate	-0.040	0.007	-5.589	0.000
habitatUrban:jdate	-0.009	0.002	-4.095	0.000

Assess year effects by adding a year covariate (2013-2016).

Table 23: *E. leptorhynchus* number of group models (year predictor) AIC table

	df	AIC	dAIC
ngroup.habXjdate.ele.year	8	1993.981	0.00
ngroup.habXjdate.ele	7	2110.653	116.67

The model with lowest AIC indicates that the number of groups is affected by habitat type, elevation, julian date and year (Tables 23-24).

Table 24: *E. leptorhynchus* 'ngroup.habXjdate.ele.year' model estimates

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-3.952	0.264	-14.974	0.000
habitatOther	-1.629	0.150	-10.893	0.000
habitatUrban	-0.506	0.144	-3.507	0.000
elevation	0.000	0.000	2.851	0.004
seasonnon-breeding	-0.749	0.201	-3.723	0.000
as.factor(year)2014	0.669	0.276	2.422	0.015
as.factor(year)2015	1.763	0.237	7.452	0.000
as.factor(year)2016	3.280	0.239	13.705	0.000

Table 25: Deviance partitioning of 'ngroup.habXjdate.ele.year' model for *E. leptorhynchus*

	Df	Deviance	Resid. Df	Resid. Dev
NULL	NA	NA	2900	2358.856
habitat	2	270.326	2898	2088.530
elevation	1	7.512	2897	2081.018
season	1	0.776	2896	2080.243
as.factor(year)	3	501.078	2893	1579.165

Models for group size

Table 26: *E. leptorhynchus* group size model estimates

	Estimate	Std. Error	z value	Pr(>|z|)
P_(Intercept)	9.348	NA	NA	NA
P_Urban	4.168	NA	NA	NA
P_Agropastoral	-7.553	NA	NA	NA
V_(Intercept)	-0.008	NA	NA	NA
V_seasonnon-breeding	-10.208	NA	NA	NA

Model-fitting function cannot estimate coefficient standard errors due to sigular Hessian matrix (Table 25). We can calculate confidence intervals based on quantiles using bootstrap (with n=30):

Table 27: Estimated coefficients and 90% CI based on bootstrap quantiles for*E. leptorhynchus* group size models

	Estimate	5%	95%
P_(Intercept)	9.348	-19.287	29.098
P_Urban	4.168	-15.801	24.797
P_Agropastoral	-7.553	-19.140	17.682
V_(Intercept)	-0.008	-14.403	25.502
V_seasonnon-breeding	-10.208	-26.835	25.664

VIP model - R functions and simulations (author: Peter Sólymos)

```r
vip <-
function(Y, X, Z, V=0,
offsetx, offsetz, weights, linkz="logit",
truncate=FALSE, hessian=TRUE, method="Nelder-Mead", init=NULL, ...) {
    if (missing(Y))
        stop("C'mon, you must have some data?!")
    if (truncate && any(Y < 1))
        stop("Y must be >0 when truncate=TRUE")
    n <- length(Y)
    id0 <- Y == V
    id1 <- !id0
    if (missing(X)) {
        X <- matrix(1, n, 1)
        colnames(X) <- "(Intercept)"
    }
    if (missing(Z)) {
        Z <- matrix(1, n, 1)
        colnames(Z) <- "(Intercept)"
    }
    kx <- ncol(X)
    kz <- ncol(Z)
    if (missing(offsetx))
        offsetx <- 0
    if (missing(offsetz))
```

```
        offsetz <- 0
    if (missing(weights))
        weights <- rep(1, n)
    linkinvx <- poisson("log")$linkinv
    linkinvz <- binomial(linkz)$linkinv
    good.num.limit <- c(.Machine$double.xmin, .Machine$double.xmax)^(1/3)

    ## VIP model full likelihood
    nll_VIP_ML <- function(parms) {
        mu <- as.vector(linkinvx(X %*% parms[1:kx] + offsetx))
        phi <- as.vector(linkinvz(Z %*% parms[(kx + 1):(kx + kz)] + offsetz))
        loglik0 <- log(phi + (1 - phi) * dpois(V, lambda = mu, log = FALSE))
        loglik1 <- log(1 - phi) + dpois(Y, lambda = mu, log = TRUE)
        loglik <- sum(weights[id0] * loglik0[id0]) + sum(weights[id1] * loglik1[id1])
        if (!is.finite(loglik) || is.na(loglik))
            loglik <- -good.num.limit[2]
        -loglik
    }
    ## 0-truncated VIP model full likelihood
    nll_VIP_TR <- function(parms) {
        mu <- as.vector(linkinvx(X %*% parms[1:kx] + offsetx))
        phi <- as.vector(linkinvz(Z %*% parms[(kx + 1):(kx + kz)] + offsetz))
        loglik0 <- log(phi + (1 - phi) * dpois(V, lambda = mu, log = FALSE) / (1-exp(-mu)))
        loglik1 <- log((1 - phi) * dpois(Y, lambda = mu, log = FALSE) / (1-exp(-mu)))
        loglik <- sum(weights[id0] * loglik0[id0]) + sum(weights[id1] * loglik1[id1])
        if (!is.finite(loglik) || is.na(loglik))
            loglik <- -good.num.limit[2]
        -loglik
    }

    if (is.null(init))
      init <- rep(0, kx+kz)
    opt <- optim(init,
        if (truncate) nll_VIP_TR else nll_VIP_ML,
        hessian=hessian, method=method, ...)
    par <- opt$par
    names(par) <- c(paste0("P_", colnames(X)), paste0("V_", colnames(Z)))
    vc <- if (hessian)
        solve(opt$hessian) else matrix(NA, length(par), length(par))
    dimnames(vc) <- list(names(par), names(par))
    out <- list(call=match.call(),
        coefficients=par, loglik=-opt$value, vcov=vc, nobs=n,
        truncate=truncate)
    class(out) <- "vip"
    out
}
vcov.vip <- function(object, ...) object$vcov
logLik.vip <- function (object, ...)
    structure(object$loglik, df = object$nobs - length(object$coef),
        nobs = object$nobs, class = "logLik")
summary.vip <- function (object, ...) {
    k <- length(object$coefficients)
    coefs <- coef(object)
```

```r
    se <- sqrt(diag(vcov(object)))
    tstat <- coefs/se
    pval <- 2 * pnorm(-abs(tstat))
    coefs <- cbind(coefs, se, tstat, pval)
    colnames(coefs) <- c("Estimate", "Std. Error", "z value", "Pr(>|z|)")
    coefs <- coefs[1:k, , drop = FALSE]
    rownames(coefs) <- names(coef(object))
    out <- list(call = object$call, coefficients=coefs, loglik = object$loglik,
        bic=BIC(object), truncate=object$truncate)
    class(out) <- "summary.vip"
    return(out)
}
print.summary.vip <- function (x, digits, ...)
{
    if (missing(digits))
        digits <- max(3, getOption("digits") - 3)
    cat("\nCall:", deparse(x$call,
        width.cutoff = floor(getOption("width") * 0.85)), "", sep = "\n")
    cat("V-Inflated", if (x$truncate) "(Zero-Truncated)" else "", "Poisson Model\n\n")
    cat(paste("Coefficients:\n", sep = ""))
    printCoefmat(x$coefficients, digits = digits, signif.legend = FALSE)
    if (!any(is.na(array(x$coefficients)))) {
        if (getOption("show.signif.stars") & any(x$coefficients[,4] < 0.1))
            cat("---\nSignif. codes: ", "0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1", "\n")
    }
    cat("\nLog-likelihood:", formatC(x$loglik, digits = digits),
        "\nBIC =", formatC(x$bic, digits = digits), "\n")
    cat("\n")
    invisible(x)
}
confint.vip <-
function (object, parm, level = 0.95, ...)
{
    cf <- coef(object)
    pnames <- names(cf)
    if (missing(parm)) {
        parm <- pnames
    } else {
        if (is.numeric(parm))
            parm <- pnames[parm]
    }
    a <- (1 - level)/2
    a <- c(a, 1 - a)
    pct <- paste(format(100 * a, trim = TRUE, scientific = FALSE, digits = 3), "%", sep="")
    ci <- array(NA, dim = c(length(parm), 2), dimnames = list(parm, pct))
    fac <- qnorm(a)
    ses <- sqrt(diag(vcov(object, model, type)))
    ci[] <- cf[parm] + ses[parm] %o% fac
    ci
}
```

Simple case

```
set.seed(123)
n <- 1000
lam <- 2 # poisson mean, can be a vector of length n
phi <- 0.4 # V-inflation probability, can be a vector of length n
V <- 2 # V is the count value, can be 0, 2, etc
y <- y0 <- rpois(n, lam)
a <- rbinom(n, 1, phi)
y[a > 0] <- V
table(Poisson=y0, Vinflated=y)
```

```
##        Vinflated
## Poisson   0   1   2   3   4   5   6   8
##       0  81   0  51   0   0   0   0   0
##       1   0 151 126   0   0   0   0   0
##       2   0   0 274   0   0   0   0   0
##       3   0   0  65 112   0   0   0   0
##       4   0   0  39   0  43   0   0   0
##       5   0   0  12   0   0  29   0   0
##       6   0   0   6   0   0   0   9   0
##       7   0   0   1   0   0   0   0   0
##       8   0   0   0   0   0   0   0   1
```

```
mod <- vip(Y=y, V=2)
summary(mod)
```

```
##
## Call:
## vip(Y = y, V = 2)
##
## V-Inflated  Poisson Model
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## P_(Intercept)  0.70472    0.02909  24.224  < 2e-16 ***
## V_(Intercept) -0.33900    0.08824  -3.842 0.000122 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log-likelihood: -1345
## BIC =  9585
```

```
cbind(True=c(log_lam=log(lam), logit_phi=qlogis(phi)),
      Est=coef(mod))
```

```
##                 True        Est
## log_lam    0.6931472  0.7047243
## logit_phi -0.4054651 -0.3389963
```

Covariates for the non-V part

```
set.seed(123)
n <- 10000
```

```
x <- rnorm(n)
df <- data.frame(x=x)
X <- model.matrix(~x, df)
beta <- c(-0.5,-0.5) # Intercept and beta values for covariate
lam <- exp(X %*% beta) # poisson mean, can be a vector of length n
phi <- 0.4 # V-inflation probability, can be a vector of length n
V <- 2 # V is the count value, can be 0, 2, etc
y <- y0 <- rpois(n, lam)
a <- rbinom(n, 1, phi)
y[a > 0] <- V
table(Poisson=y0, Vinflated=y)
```

```
##          Vinflated
## Poisson    0    1    2    3    4    5    6    7    8
##       0 3182    0 2131    0    0    0    0    0    0
##       1    0 1981 1137    0    0    0    0    0    0
##       2    0    0 1088    0    0    0    0    0    0
##       3    0    0  118  226    0    0    0    0    0
##       4    0    0   40    0   57    0    0    0    0
##       5    0    0   14    0    0   17    0    0    0
##       6    0    0    1    0    0    0    3    0    0
##       7    0    0    2    0    0    0    0    1    0
##       8    0    0    1    0    0    0    0    0    1
```

```
mod <- vip(Y=y, X=X, V=2)
summary(mod)
```

```
##
## Call:
## vip(Y = y, X = X, V = 2)
##
## V-Inflated  Poisson Model
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## P_(Intercept) -0.45313    0.02037  -22.24   <2e-16 ***
## P_x           -0.49231    0.01664  -29.58   <2e-16 ***
## V_(Intercept) -0.48770    0.02483  -19.64   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log-likelihood: -1.133e+04
## BIC = 1.147e+05
```

```
cbind(True=c(beta=beta, logit_phi=qlogis(phi)),
      Est=coef(mod))
```

```
##                 True        Est
## beta1     -0.5000000 -0.4531273
## beta2     -0.5000000 -0.4923100
## logit_phi -0.4054651 -0.4876957
```

Methods

```r
coef(mod)
```

```
## P_(Intercept)           P_x V_(Intercept) 
##    -0.4531273    -0.4923100    -0.4876957
```

```r
vcov(mod)
```

```
##               P_(Intercept)           P_x V_(Intercept)
## P_(Intercept)  0.0004151059  1.815322e-04 -1.454395e-04
## P_x            0.0001815322  2.769780e-04 -5.339019e-05
## V_(Intercept) -0.0001454395 -5.339019e-05  6.165031e-04
```

```r
summary(mod)
```

```
## 
## Call:
## vip(Y = y, X = X, V = 2)
## 
## V-Inflated  Poisson Model
## 
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)    
## P_(Intercept) -0.45313    0.02037  -22.24   <2e-16 ***
## P_x           -0.49231    0.01664  -29.58   <2e-16 ***
## V_(Intercept) -0.48770    0.02483  -19.64   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Log-likelihood: -1.133e+04 
## BIC = 1.147e+05
```

```r
confint(mod)
```

```
##                     2.5%      97.5%
## P_(Intercept) -0.4930599 -0.4131947
## P_x           -0.5249290 -0.4596910
## V_(Intercept) -0.5363606 -0.4390308
```

```r
nobs(mod)
```

```
## [1] 10000
```

```r
logLik(mod)
```

```
## 'log Lik.' -11332.89 (df=9997)
```

```r
AIC(mod)
```

```
## [1] 42659.77
```

```r
BIC(mod)
```

```
## [1] 114741.5
```

Zero-truncated VIP

We can truncate counts to be larger than 0. We also need $V > 0$ (for $V = 0$ case, look into ZIP or conditional Poisson model). Conceptually, the V-Inflation follows the 0-truncation (because we cannot observe 0, real truncated distribution).

The 0-truncated PDF is $P(Y = y \mid Y > 0) = \frac{P(Y=y)}{1-P(Y=0)}$. The 0-truncated V-Inflated density is $P(Y = y \mid Y > 0, V > 0) = \phi I(Y = V) + (1 - \phi)\frac{f(y;\lambda)}{1-f(0;\lambda)}$. This can be achieved in the `vip` call by the argument `truncate=TRUE`.

Here we use covariates for both the V and non-V part.

```
set.seed(1)
n <- 1000
x <- rnorm(n)
z <- runif(n, -1, 1)
df <- data.frame(x=x, z=z)
X <- model.matrix(~x, df)
Z <- model.matrix(~z, df)
beta <- c(-0.5, -0.5)
alpha <- c(0, 0.5)
lam <- exp(X %*% beta)
phi <- plogis(Z %*% alpha)
V <- 2 # V is the count value, cannot be 0
y <- y0 <- rpois(n, lam)
a <- rbinom(n, 1, phi)
keep <- y0>0
y <- y[keep] # conditioning (i.e. exclude 0s)
y0 <- y0[keep]
X <- X[keep,]
Z <- Z[keep,]
y[a[keep] > 0] <- V
table(Poisson=y0, Vinflated=y)
```

```
##        Vinflated
## Poisson   1   2   3   4   6
##       1 155 141   0   0   0
##       2   0 127   0   0   0
##       3   0  21  16   0   0
##       4   0   4   0   7   0
##       5   0   2   0   0   0
##       6   0   0   0   0   1
```

```
mod <- vip(Y=y, X=X, Z=Z, V=2, truncate=TRUE)
summary(mod)
```

```
##
## Call:
## vip(Y = y, X = X, Z = Z, V = 2, truncate = TRUE)
##
## V-Inflated (Zero-Truncated) Poisson Model
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## P_(Intercept) -0.50814    0.14803  -3.433 0.000598 ***
## P_x           -0.57344    0.10170  -5.639 1.71e-08 ***
## V_(Intercept)  0.02131    0.12572   0.170 0.865387
## V_z            0.47041    0.20691   2.273 0.022999 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log-likelihood: -384.1
## BIC =  3664
```

```
cbind(True=c(beta=beta, alpha=alpha),
      Est=coef(mod))
```

```
##        True         Est
## beta1  -0.5 -0.50813933
## beta2  -0.5 -0.57343540
## alpha1  0.0  0.02131236
## alpha2  0.5  0.47040670
```