# Patagonia parrots density analysis

*Francisco Denes and Peter Solymos*

*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.

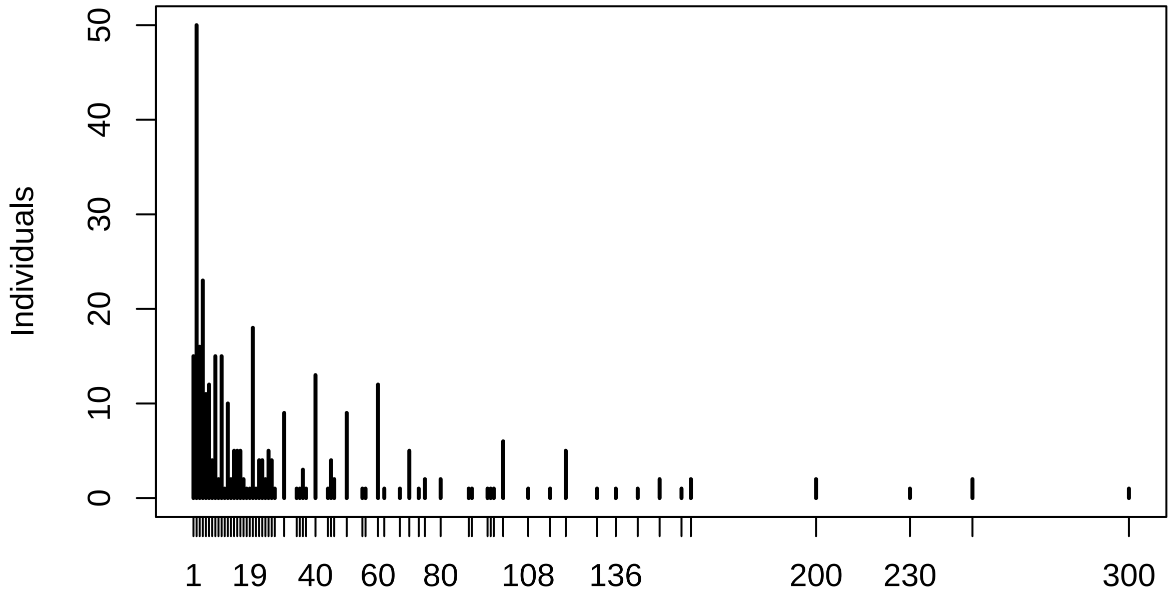


Figure 1: *Enicognathus ferrugineus* count frequencies

# 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, presumably 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 $\mathrm{G_i} \sim \mathrm{Poisson}(\mathrm{D_i A_i})$, where $\mathrm{D_i}$ = covariates and $\mathrm{A_i}$ = area sampled in site *i.* $\mathrm{A_i}$ is calculated using the habitat-specific estimated EDR as $A_i = L_i * (2EDR_i)$, where $L$ is the length of each site (i.e. the transect). $\mathrm{A_i}$ is added to the model as an offset.

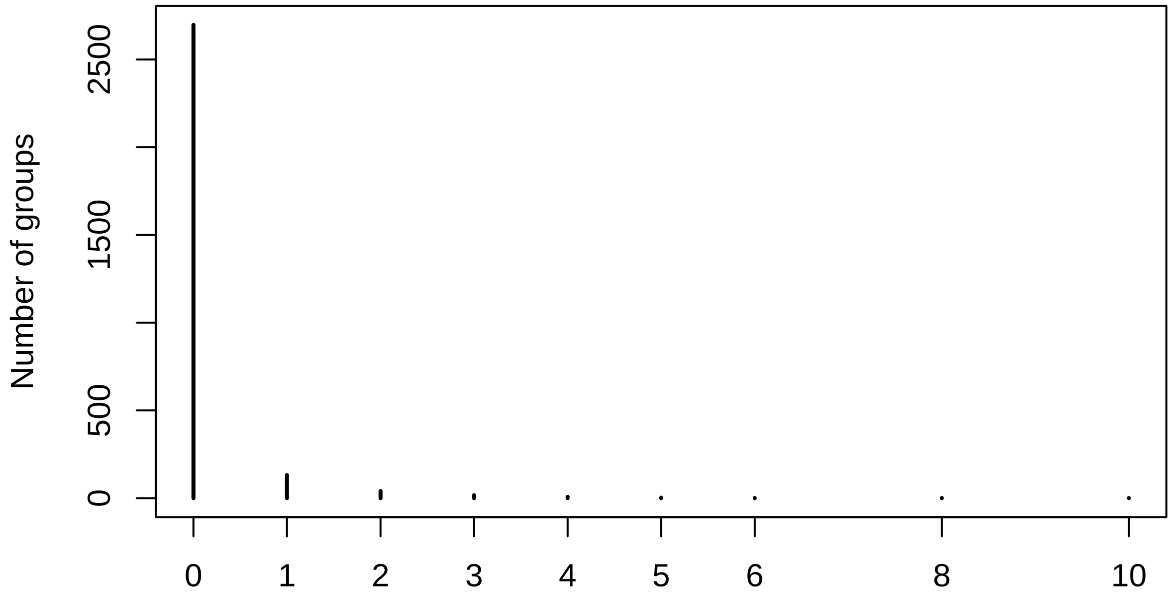


Figure 2: *Enicognathus ferrugineus* group numbers

**Model selection**

For number of groups, we use a stage-wise selection procedure. First, 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(elevationˆ2) | 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(elevationˆ2) | 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 parsimonious 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(elevationˆ2) | 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 type, 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 reference to the "Z" representing 0 in a zero-inflated Poisson model, or ZIP), then we add the >0 condition.

R functions for the VIP model are presented at the end of this document, including simulations to check the estimating procedure.

### 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.

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.

### Goodness-of-fit

Model goodness-of-fit can be evaluated visually, inspecting the proportions of fitted values against the count distribution. Deviation from the 1:1 line can be used as a goodness-of-fit metric, with the better model showing smaller deviation.

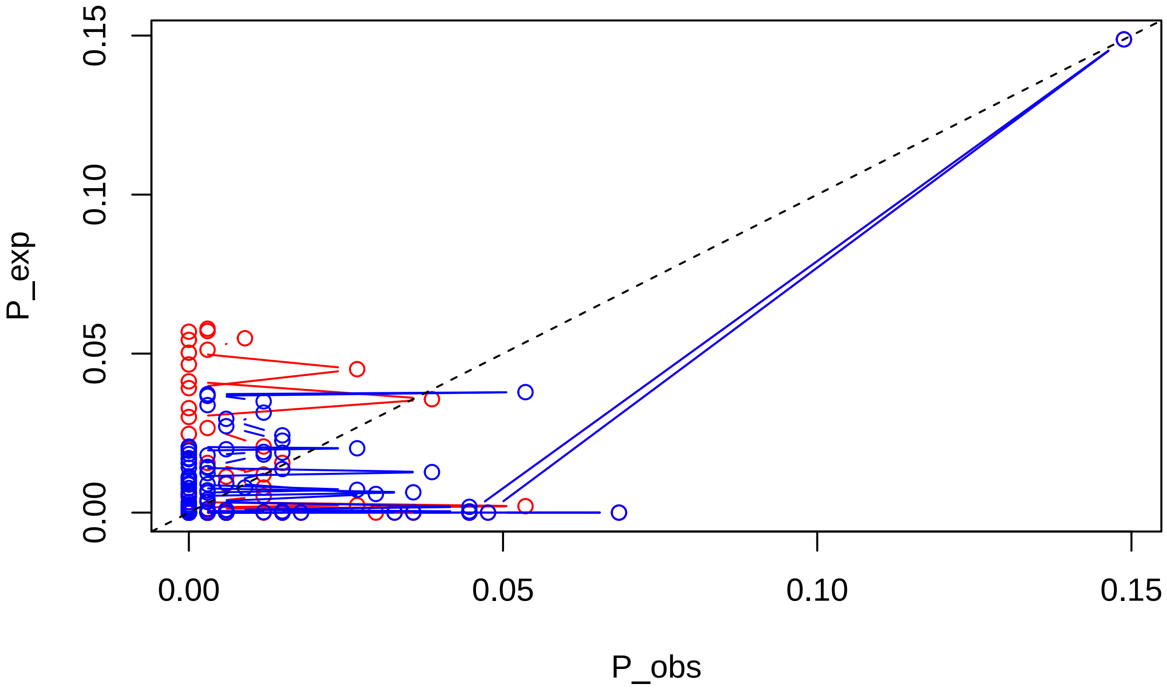


Figure 3: *Enicognathus ferrugineus* group size model goodness-of-fit (red: null model; blue: habitat.season model). Better model is closer to 1:1 line.

The deviation of null model is 1.38, and of the VIP.habitat.season model is 1.17.

### Confidence intervals for model estimates

Estimated model coefficients (Table 13) indicate that group sizes are larger in urban ($\beta = 0.877$) and agropastoral ($\beta = 0.622$) habitats when compared to other natural habitats. Moreover, the probability of there being extra pairs (i.e. the V-inflation probability, $\phi$) is smaller in the non-breeding season ($\beta =$ -2.058).

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

| | Estimate | Std. Error | z value | Pr(>\|z\|) |
|---|---|---|---|---|
| P_(Intercept) | 3.027 | 0.019 | 161.500 | 0.000 |
| P_Urban | 0.877 | 0.023 | 38.534 | 0.000 |
| P_Agropastoral | 0.622 | 0.036 | 17.167 | 0.000 |
| V_(Intercept) | 0.223 | 0.671 | 0.333 | 0.739 |
| V_seasonnon-breeding | -2.058 | 0.690 | -2.984 | 0.003 |

Confidence intervals (CI) based on estimated standard errors can be obtained (Table 14):

Table 14: *E. ferrugineous* group size model CI

| | 2.5% | 97.5% |
|---|---:|---:|
| P_(Intercept) | 2.990 | 3.064 |
| P_Urban | 0.832 | 0.922 |
| P_Agropastoral | 0.551 | 0.694 |
| V_(Intercept) | -1.092 | 1.538 |
| V_seasonnon-breeding | -3.410 | -0.706 |

Alternatively, we can estimate confidence intervals based on quantiles using bootstrap samples (with n=250) for the estimated coefficients (Table 15).

Table 15: Estimated coefficients and 95% CI based on bootstrap sample (n= 250) quantiles for *E. ferrugineous* group size models

| | Estimate | 2.5% | 97.5% |
|---|---:|---:|---:|
| P_(Intercept) | 3.027 | 2.924 | 2.924 |
| P_Urban | 0.877 | 0.956 | 0.956 |
| P_Agropastoral | 0.622 | 0.512 | 0.512 |
| V_(Intercept) | 0.223 | 0.693 | 0.693 |
| V_seasonnon-breeding | -2.058 | -2.554 | -2.554 |

## Density Predictions

The mean expected abundance ($E$) can be predicted for each surveyed site $i$ as $E_i = (\phi_i 2 + (1 - \phi_i)\mu_i)N_i$, where $N$ is the mean number of groups, $\mu$ is the mean group size and $\phi$ is the V(=2)-inflation probability (i.e. the probability that a given group will be a pair).

We can then obtain the expected density stratified by habitat type, year and season, as $D = \frac{\sum E}{\sum A}$, where $A$ is the area surveyed.

Table 16: *E. ferrugineous* mean density predictions for each habitat type, year and season

| Habitat | Year | Season | Density (individuals/ha) | Density (2.5% CI) | Density (97.5% CI) |
|---|---|---|---:|---:|---:|
| Agropastoral | 2013 | breeding | 0.001 | 0.000 | 0.002 |
| Other | 2013 | breeding | 0.002 | 0.001 | 0.004 |
| Urban | 2013 | breeding | 0.007 | 0.003 | 0.017 |
| Agropastoral | 2015 | breeding | 0.002 | 0.001 | 0.004 |
| Other | 2015 | breeding | 0.005 | 0.002 | 0.011 |
| Urban | 2015 | breeding | 0.022 | 0.008 | 0.054 |
| Agropastoral | 2013 | non-breeding | 0.002 | 0.001 | 0.004 |
| Other | 2013 | non-breeding | 0.011 | 0.008 | 0.015 |
| Urban | 2013 | non-breeding | 0.041 | 0.028 | 0.057 |
| Agropastoral | 2014 | non-breeding | 0.003 | 0.002 | 0.004 |
| Other | 2014 | non-breeding | 0.012 | 0.009 | 0.016 |
| Urban | 2014 | non-breeding | 0.048 | 0.038 | 0.062 |
| Agropastoral | 2015 | non-breeding | 0.008 | 0.005 | 0.011 |
| Other | 2015 | non-breeding | 0.027 | 0.021 | 0.036 |
| Urban | 2015 | non-breeding | 0.111 | 0.085 | 0.143 |

| Habitat | Year | Season | Density (individuals/ha) | Density (2.5% CI) | Density (97.5% CI) |
|---|---|---|---|---|---|
| Agropastoral | 2016 | non-breeding | 0.006 | 0.003 | 0.009 |
| Other | 2016 | non-breeding | 0.019 | 0.013 | 0.027 |
| Urban | 2016 | non-breeding | 0.082 | 0.057 | 0.115 |

## Slender-billed parakeet *Enicognathus leptorhynchus*

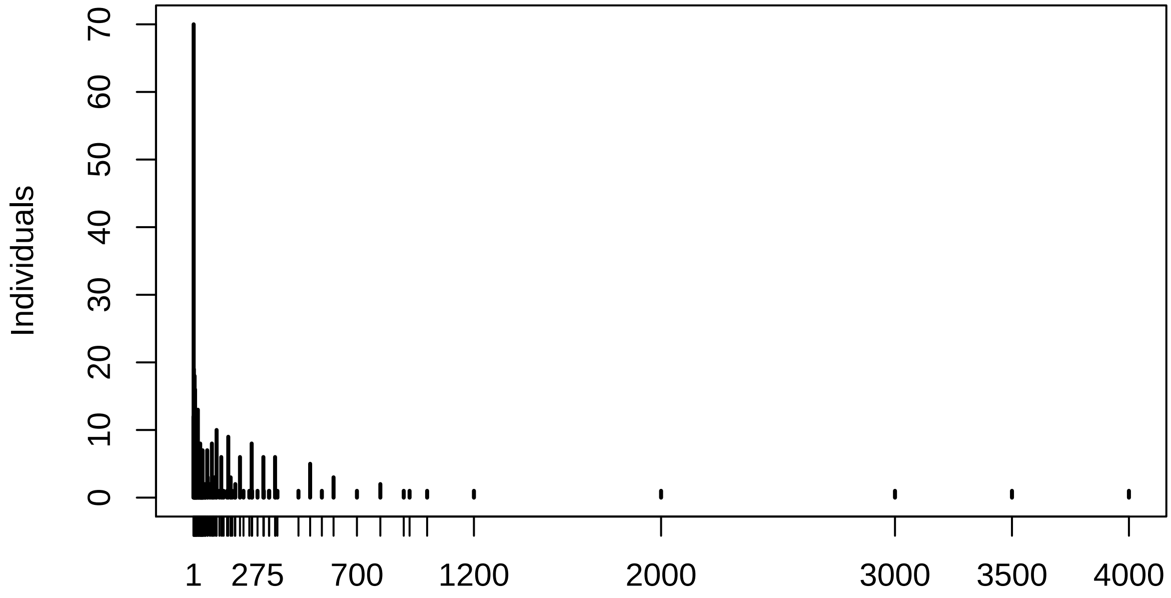


Figure 4: *Enicognathus leptorhynchus* count frequencies

### Estimating effective detection radius (EDR)

We follow the same procedure used for the austral parakeet above.

Table 17: *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 |

| | df | AIC | dAIC |
|---|---|---|---|
| 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 16), 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 17.

Table 18: *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 19: *E. leptorhynchus* habitat-specific mean EDR (m)

| Habitat | EDR |
|---|---|
| Other | 235.4183 |
| Urban | 224.2124 |
| Agropastoral | 180.7002 |

## Models for number of groups

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.

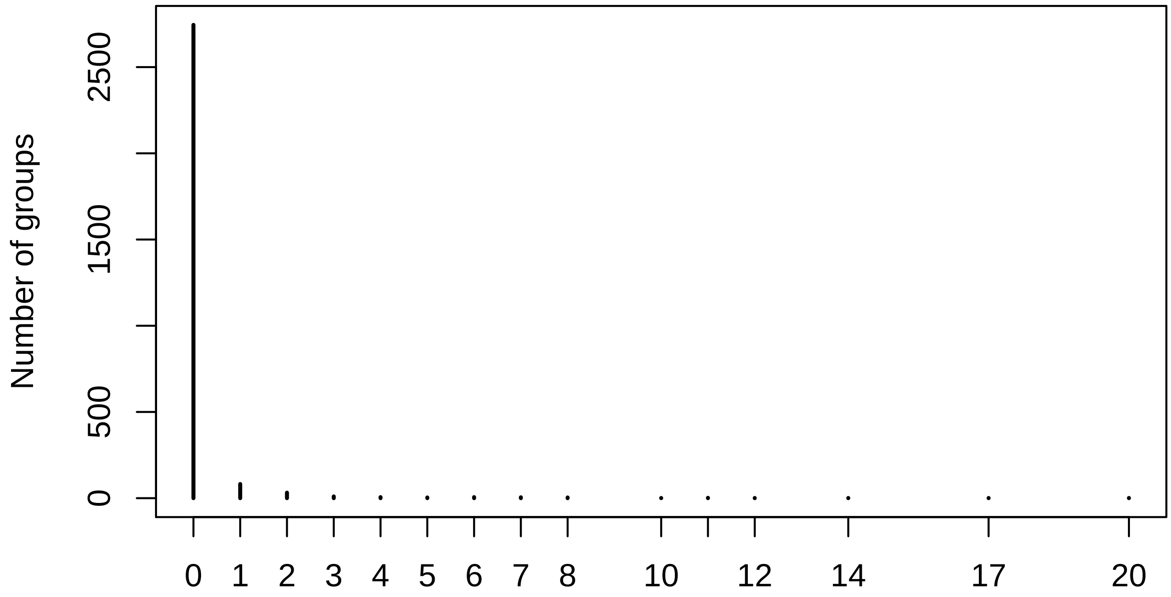


Figure 5: *Enicognathus leptorhynchus* group numbers

**Model selection**

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

Table 20: *E. leptorhynchus* number of group models

| | df | AIC | dAIC |
|---|---|---|---|
| ngroup.hab.ele2 | 5 | 2486.990 | 0.00 |
| ngroup.hab.ele | 4 | 2487.834 | 0.84 |
| ngroup.hab | 3 | 2493.346 | 6.36 |
| ngroup.ele2 | 3 | 2746.031 | 259.04 |
| ngroup.ele | 2 | 2761.122 | 274.13 |

The models ‘ngroup.hab.ele’ (with linear elevation effect) and ‘ngroup.hab.ele2’ (with quadratic effect) are equally parsimonious (Table 18), but the quadratic term in the latter is mostly uninformative (Tables 19-20). Given these are nested models, we drop the quadratic term and continue with the model with habitat and linear elevation effects.

Table 21: *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 22: 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 23: *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 24: *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 |

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

Table 25: *E. leptorhynchus* number of group models (year predictor) AIC table

| | df | AIC | dAIC |
|---|---|---|---|
| ngroup.habXjdate.ele.year | 10 | 1907.516 | 0.00 |
| ngroup.habXjdate.ele | 7 | 2110.653 | 203.14 |

The best-ranked model indicates that the number of groups is affected by habitat type, elevation, Julian date and year (Tables 23-24).

Table 26: *E. leptorhynchus* 'ngroup.habXjdate.ele.year' model estimates

| | Estimate | Std. Error | z value | Pr(>\|z\|) |
|---|---|---|---|---|
| (Intercept) | -3.424 | 0.553 | -6.189 | 0.000 |
| elevation | 0.001 | 0.000 | 3.208 | 0.001 |
| habitatOther | 0.677 | 0.350 | 1.932 | 0.053 |
| habitatUrban | 0.566 | 0.257 | 2.206 | 0.027 |
| jdate | -0.002 | 0.002 | -1.291 | 0.197 |
| as.factor(year)2014 | -0.201 | 0.352 | -0.570 | 0.569 |
| as.factor(year)2015 | 1.388 | 0.233 | 5.953 | 0.000 |
| as.factor(year)2016 | 1.909 | 0.442 | 4.322 | 0.000 |
| habitatOther:jdate | -0.020 | 0.004 | -5.091 | 0.000 |
| habitatUrban:jdate | -0.008 | 0.002 | -4.327 | 0.000 |

Table 27: Deviance partitioning of 'ngroup.habXjdate.ele.year' model for *E. leptorhynchus*

| | Df | Deviance | Resid. Df | Resid. Dev |
|---|---|---|---|---|
| NULL | NA | NA | 2900 | 2358.856 |
| elevation | 1 | 0.551 | 2899 | 2358.306 |
| habitat | 2 | 277.287 | 2897 | 2081.018 |
| jdate | 1 | 259.490 | 2896 | 1821.528 |
| as.factor(year) | 3 | 239.982 | 2893 | 1581.547 |
| habitat:jdate | 2 | 92.847 | 2891 | 1488.700 |

## Models for group size

### Goodness-of-fit

Model goodness-of-fit can be evaluated visually, inspecting the proportions of fitted values against the count distribution. Deviation from the 1:1 line can be used as a goodness-of-fit metric, with the better model showing smaller deviation.

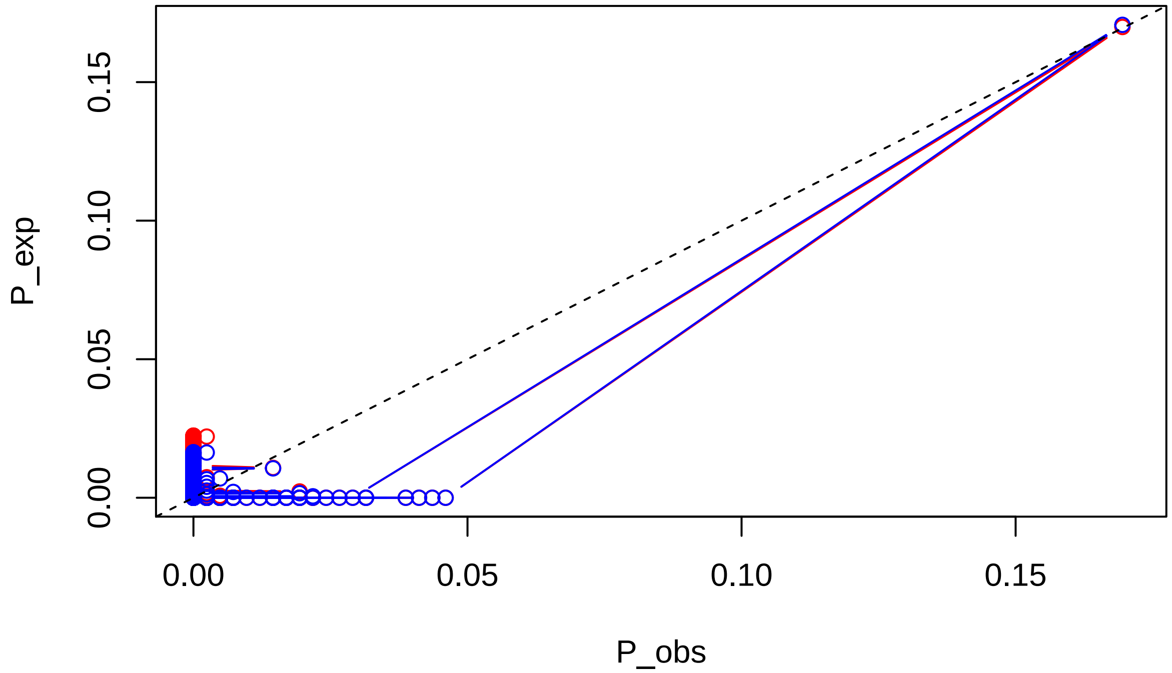


Figure 6: *Enicognathus leptorhynchus* group size model goodness-of-fit (red: null model; blue: habitat.season model). Better model is closer to 1:1 line.

The deviation of null model is 1.62, and of the VIP.habitat.season model is 1.59.

**Confidence intervals for model estimates**

Estimated model coefficients (Table 25) indicate that group sizes are slightly smaller in urban ($\beta$ = -0.055) and larger in agropastoral ($\beta$ = 0.154) habitats when compared to other natural habitats. Moreover, the probability of there being extra pairs (i.e. the V-inflation probability, $\phi$) is smaller in the non-breeding season ($\beta$ = -1.295).

Table 28: *E. leptorhynchus* group size model estimates

| | Estimate | Std. Error | z value | Pr(>\|z\|) |
|---|---:|---:|---:|---:|
| P_(Intercept) | 5.231 | 0 | 6.213682e+54 | 0 |
| P_Urban | -0.055 | 0 | -1.311872e+49 | 0 |
| P_Agropastoral | 0.154 | 0 | 3.659219e+49 | 0 |
| V_(Intercept) | -0.430 | 0 | -7.216786e+49 | 0 |
| V_seasonnon-breeding | -1.295 | 0 | -2.175643e+50 | 0 |

Confidence intervals (CI) based on estimated standard errors can be obtained (Table 26):

Table 29: *E. leptorhynchus* group size model CI

| | 2.5% | 97.5% |
|---|---:|---:|
| P_(Intercept) | 5.231 | 5.231 |
| P_Urban | -0.055 | -0.055 |
| P_Agropastoral | 0.154 | 0.154 |
| V_(Intercept) | -0.430 | -0.430 |
| V_seasonnon-breeding | -1.295 | -1.295 |

We can also calculate confidence intervals based on quantiles using bootstrap (with n=250):

Table 30: Estimated coefficients and 95% CI based on bootstrap quantiles for *E. leptorhynchus* group size models

| | Estimate | 2.5% | 97.5% |
|---|---:|---:|---:|
| P_(Intercept) | 5.231 | 4.478 | 4.478 |
| P_Urban | -0.055 | 0.698 | 0.698 |
| P_Agropastoral | 0.154 | 0.454 | 0.454 |
| V_(Intercept) | -0.430 | -0.062 | -0.059 |
| V_seasonnon-breeding | -1.295 | -1.687 | -1.684 |

## Density Predictions

The mean expected abundance ($E$) can be predicted for each surveyed site $i$ as $E_i = (\phi_i 2 + (1-\phi_i)\mu_i)N_i$, where $N$ is the mean number of groups, $\mu$ is the mean group size and $\phi$ is the V(=2)-inflation probability (i.e. the probability that a given group will be a pair).

We can then obtain the expected density stratified by habitat type, year and season, as $D = \frac{\sum E}{\sum A}$, where $A$ is the area surveyed.

Table 31: *E. ferrugineous* mean density predictions for each habitat type, year and season

| Habitat | Year | Season | Density (individuals/ha) | Density (2.5% CI) | Density (97.5% CI) |
|---|---|---|---:|---:|---:|
| Agropastoral | 2013 | breeding | 0.044 | 0.028 | 0.068 |
| Other | 2013 | breeding | 0.000 | 0.000 | 0.001 |
| Urban | 2013 | breeding | 0.006 | 0.002 | 0.013 |
| Agropastoral | 2015 | breeding | 0.189 | 0.147 | 0.239 |
| Other | 2015 | breeding | 0.001 | 0.000 | 0.005 |
| Urban | 2015 | breeding | 0.022 | 0.010 | 0.047 |
| Agropastoral | 2013 | non-breeding | 0.075 | 0.049 | 0.114 |
| Other | 2013 | non-breeding | 0.000 | 0.000 | 0.002 |
| Urban | 2013 | non-breeding | 0.011 | 0.005 | 0.023 |
| Agropastoral | 2014 | non-breeding | 0.077 | 0.053 | 0.112 |
| Other | 2014 | non-breeding | 0.012 | 0.007 | 0.022 |
| Urban | 2014 | non-breeding | 0.037 | 0.023 | 0.057 |
| Agropastoral | 2015 | non-breeding | 0.302 | 0.239 | 0.377 |
| Other | 2015 | non-breeding | 0.003 | 0.001 | 0.013 |
| Urban | 2015 | non-breeding | 0.048 | 0.026 | 0.091 |
| Agropastoral | 2016 | non-breeding | 0.813 | 0.666 | 0.994 |

| Habitat | Year | Season | Density (individuals/ha) | Density (2.5% CI) | Density (97.5% CI) |
|---|---|---|---|---|---|
| Other | 2016 | non-breeding | 0.426 | 0.319 | 0.580 |
| Urban | 2016 | non-breeding | 0.605 | 0.441 | 0.821 |

---

# VIP model - R functions and simulations

**Peter Solymos**

## Functions

The `vip` function does the optimization. `method` argument can take values listed for `optim` and also `"DE"` for differential evolution algorithm. If there are convergence issues with `"Nelder-Mead"`, try `"SANN"` and `"DE"`.

```
library(DEoptim)
library(Matrix)
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
    }
    .solvenear <-
    function(x)
    {
        xinv <- try(solve(x), silent = TRUE)
        if (inherits(xinv, "try-error"))
            xinv <- as.matrix(solve(Matrix::nearPD(x)$mat))
        xinv
    }
    if (is.null(init))

        init <- rep(0, kx+kz)
    nll <- if (truncate) nll_VIP_TR else nll_VIP_ML

    if (method == "DE") {
        DElimit <- 10
        up <- rep(DElimit, length(init))
        lo <- -up
        opt <- DEoptim(fn=nll, lower=lo, upper=up,
            control=list(trace=FALSE, itermax=length(init)*200))
        par <- opt$optim$bestmem
        names(par) <- c(paste0("P_", colnames(X)), paste0("V_", colnames(Z)))
        ll <- -opt$optim$bestval
        if (hessian) {
            hess <- optimHess(opt$optim$bestmem, nll)
            vc <- .solvenear(hess)
        } else {
            matrix(NA, length(par), length(par))
        }
    } else {
        opt <- optim(init, nll,
            hessian=hessian, method=method, ...)
        par <- opt$par
        names(par) <- c(paste0("P_", colnames(X)), paste0("V_", colnames(Z)))
        vc <- if (hessian)
            .solvenear(opt$hessian) else matrix(NA, length(par), length(par))
```

```r
        ll <- -opt$value
    }
    dimnames(vc) <- list(names(par), names(par))
    out <- list(call=match.call(),
        coefficients=par, loglik=ll, vcov=vc, nobs=n,
        truncate=truncate, Y=Y, X=X, Z=Z, V=V,
        offsetx=offsetx, offsetz=offsetz, weights=weights,
        linkz=linkz, method=method, init=init)
    class(out) <- "vip"
    out
}
vcov.vip <- function(object, ...) object$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$coefficients)
    coefs <- coef(object)
    se <- sqrt(diag(vcov(object)))
    tstat <- coefs/se
    pval <- 2 * pnorm(-abs(tstat))
    coefs <- cbind(coefs, se, tstat, pval)
    colnames(coefs) <- c("Estimate", "Std. Error", "z value", "Pr(>|z|)")
    coefs <- coefs[1:k, , drop = FALSE]
    rownames(coefs) <- names(coef(object))
    out <- list(call = object$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 <- 1000
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
##       0 299   0 228   0   0   0   0
##       1   0 201 131   0   0   0   0
##       2   0   0  95   0   0   0   0
##       3   0   0  14  18   0   0   0
##       4   0   0   2   0   6   0   0
##       5   0   0   0   0   0   4   0
##       6   0   0   0   0   0   0   2
```

```
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) -0.39671    0.06382  -6.216 5.11e-10 ***
## P_x           -0.49031    0.05331  -9.198  < 2e-16 ***
## V_(Intercept) -0.42040    0.07813  -5.381 7.42e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log-likelihood: -1141
## BIC =  9169
```

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

```
##                 True        Est
## beta1     -0.5000000 -0.3967122
## beta2     -0.5000000 -0.4903124
## logit_phi -0.4054651 -0.4203965
```

## Methods

```
coef(mod)
```

```
## P_(Intercept)           P_x V_(Intercept)
##    -0.3967122    -0.4903124    -0.4203965
```

```
vcov(mod)
```

```
##               P_(Intercept)           P_x V_(Intercept)
## P_(Intercept)   0.004073534  0.0018566813 -0.0014632502
## P_x             0.001856681  0.0028417922 -0.0006105607
## V_(Intercept)  -0.001463250 -0.0006105607  0.0061042545
```

```
summary(mod)
```

```
##
## Call:
## vip(Y = y, X = X, V = 2)
##
## V-Inflated  Poisson Model
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## P_(Intercept) -0.39671    0.06382  -6.216 5.11e-10 ***
## P_x           -0.49031    0.05331  -9.198  < 2e-16 ***
## V_(Intercept) -0.42040    0.07813  -5.381 7.42e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log-likelihood: -1141
## BIC =  9169
```

```
confint(mod)
```

```
##                     2.5%      97.5%
## P_(Intercept) -0.5218054 -0.2716189
## P_x           -0.5947950 -0.3858297
## V_(Intercept) -0.5735279 -0.2672650
```

```
nobs(mod)
```

```
## [1] 1000
```

```r
logLik(mod)
```

```
## 'log Lik.' -1140.751 (df=997)
```

```r
AIC(mod)
```

```
## [1] 4275.502
```

```r
BIC(mod)
```

```
## [1] 9168.534
```

# 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.

```r
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
```

```r
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
```

```r
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
```

## Goodness of fit

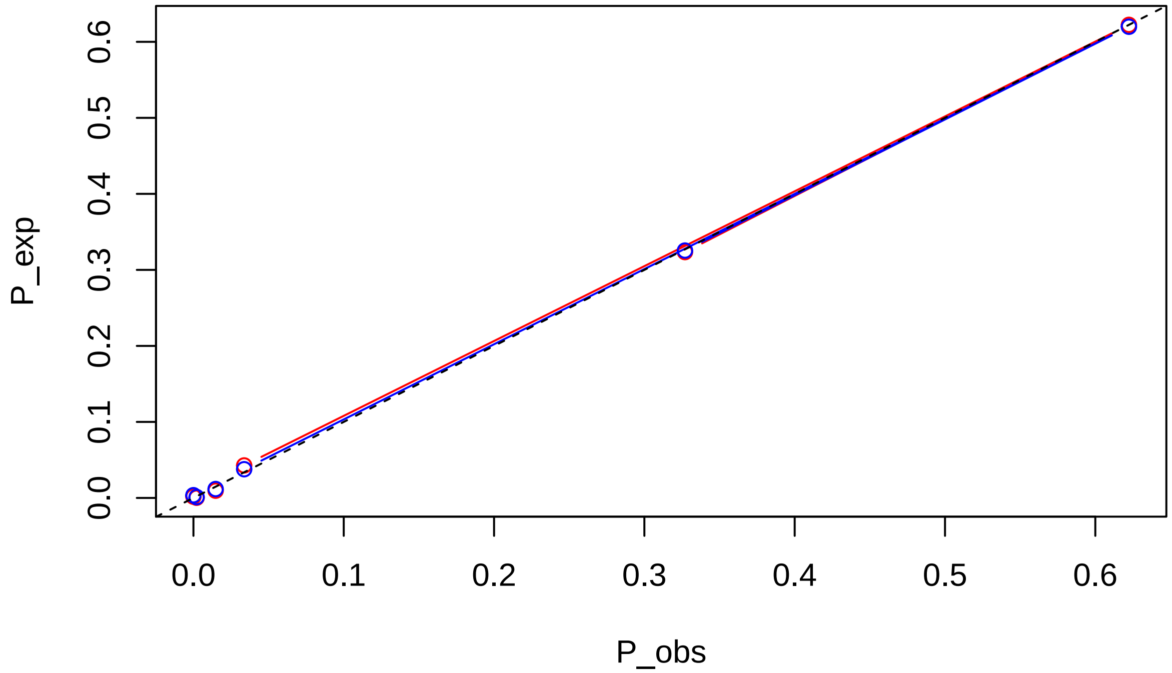
```r
goodness <- function(object, maxcount=NULL) {
    if (is.null(maxcount))
        maxcount <- max(object$Y)
    COUNTS <- if (object$truncate)
        1L:max(object$Y) else 0L:maxcount

    P_obs <- table(object$Y) / nobs(object)
    P_obs <- as.numeric(P_obs[match(COUNTS, names(P_obs))])
    P_obs[is.na(P_obs)] <- 0
    names(P_obs) <- COUNTS

    P_exp <- P_obs
    P_exp[] <- 0

    linkinvx <- poisson("log")$linkinv
    linkinvz <- binomial(object$linkz)$linkinv
    parms <- coef(object)
    kx <- ncol(object$X)
    kz <- ncol(object$Z)
    mu <- as.vector(linkinvx(object$X %*% parms[1:kx] + object$offsetx))
    phi <- as.vector(linkinvz(object$Z %*% parms[(kx + 1):(kx + kz)] + object$offsetz))
    #id0 <- object$Y == object$V
    Pmat <- matrix(0, nobs(object), length(COUNTS))
    colnames(Pmat) <- COUNTS

    PV <- if (object$truncate) {
        log(phi + (1 - phi) * dpois(object$V,
            lambda = mu, log = FALSE) / (1-exp(-mu)))
    } else {
        log(phi + (1 - phi) * dpois(object$V, lambda = mu, log = FALSE))
    }
    for (i in COUNTS) {
        PC <- if (object$truncate) {
            log((1 - phi) * dpois(i, lambda = mu, log = FALSE) / (1-exp(-mu)))
        } else {
            log(1 - phi) + dpois(i, lambda = mu, log = TRUE)
        }
        Pmat[,as.character(i)] <- if (i == object$V)
            exp(PV) else exp(PC)
    }
    P_exp <- colMeans(Pmat)
    cbind(P_obs=P_obs, P_exp=P_exp)
}

## fit null model
mod0 <- vip(Y=y, X=X[,1,drop=FALSE], Z=Z[,1,drop=FALSE], V=2, truncate=TRUE)
## calculate GoF for null and other model
(gof0 <- goodness(mod0))
```

```
##         P_obs        P_exp
## 1 0.327004219 0.3234032597
## 2 0.622362869 0.6223385125
## 3 0.033755274 0.0427550867
## 4 0.014767932 0.0095197408
## 5 0.000000000 0.0016957134
## 6 0.002109705 0.0002517089
```

```r
(gof <- goodness(mod))
```

```
##         P_obs       P_exp
## 1 0.327004219 0.325425662
## 2 0.622362869 0.619861184
## 3 0.033755274 0.037772839
## 4 0.014767932 0.011711772
## 5 0.000000000 0.003622341
## 6 0.002109705 0.001119974
```

```r
## better model is closer to the 1:1 line
plot(gof0, type="b", col=2)
points(gof, type="b", col=4)
abline(0, 1, lty=2)
```



```
## better model will give smaller absolute deviation
sum(abs(apply(gof0, 1, diff)))
```

```
## [1] 0.02142703
```

```
sum(abs(apply(gof, 1, diff)))
```

```
## [1] 0.01576604
```