Estimation de paramètres démographiques à partir d'individus non marqués

Marc J. Mazerolle

Centre d'étude de la forêt, Institut de recherche sur les forêts, Université du Québec en Abitibi-Témiscamingue, QC

Jour 2

Utilisation du package `unmarked`

- préparation du jeu de données
- modèles d'occupation à une saison
- modèles d'occupation dynamiques
- modèles N-mixture à une saison
- modèles N-mixture dynamiques
- sélection de modèles et inférence multimodèle avec `AICcmodavg`
- tests d'ajustement

Une approche unifiante

Peu importe l'analyse, les mêmes étapes reviennent :

- importer les données
- exécuter les modèles
- vérifier les suppositions
- visionner les résultats (β, SE)
- représenter graphiquement les résultats

Possible d'exécuter les analyses dans des programmes indépendants, sauvegarder les projets, copier-coller ensuite pour traitement ultérieur (graphiques, inférence multimodèle)

- pénible si les données changent et risque d'erreur
- peut être difficile à répéter (superviseur)

Il y a une meilleure façon d'organiser le travail ...

Plan

Utilisons R

Rédiger un script de codes

- transparent
- une feuille de route qui documente tout ce qu'on a fait
- permet de facilement répéter le traitement de A à Z

Utiliser un éditeur de texte intelligent

- petit logiciel qui reconnaît la syntaxe de R
- envoie les codes directement à R
- coloration de différents termes, reconnaît les parenthèses, fournit de l'aide, complétion automatique des commandes
- plusieurs disponibles (e.g., `Tinn-R`, `RStudio`, `Emacs`)

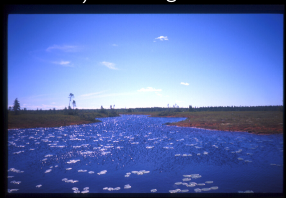
Sauvegarder les commandes complètes (importation, chargement de packages, graphiques) dans un fichier `.R` (un script)

- ajouter des commentaires (#)
- peut être recyclé pour différentes analyses

Exemple 1 — Modèle d'occupation à une saison

Occupation à une saison

Grenouilles vertes (*Lithobates clamitans*) en étangs de tourbières



Échantillonnage à l'aide de points d'écoute

- 34 étangs
- 5 visites

Utilisation de unmarked

À l'aide du package `unmarked`

Les étapes principales :

- installer et charger package
- importer les données
- formater les données à l'aide des fonctions
- exécuter l'analyse
- sélection de modèle et inférence multimodèle
- vérifier l'ajustement
- faire des graphiques

Utilisation de unmarked

Installer le package

```
> install.packages(pkgs = "unmarked")
```

Charger le package afin d'utiliser les fonctions lors de la session de travail en R

```
> library(unmarked)
```


Importation des données

Importer les données de détection

```
> gfrog <- read.table("Single_season_detection.csv",
     header = TRUE)
> head(gfrog)
  Pond Visit1 Visit2 Visit3 Visit4 Visit5
1    1      0      0      1      0      0
2    2      0      0      1      1      0
3    3      0      0      0      0      1
4    4      0      0      1      1      1
5    5      0      0      0      1      0
6    6      0      0      0      0      0
```


Importation des données

Importer variables de sites

```
> site.covs <- read.table("Single_season_siteCovs.csv",
     header = TRUE)
> head(site.covs)
  Pond Perimeter Dist.pond
1    1        79        27
2    2       159         2
3    3       105         8
4    4       628         3
5    5        75         3
6    6       141        58
```


Importation des données

Importer les variables d'échantillonnage

```
> airtemp <- read.table("Single_season_samplingCovs.csv",
     header = TRUE)
> head(airtemp)
  Pond Visit1 Visit2 Visit3 Visit4 Visit5
1    1    4.5   11.0   22.0   16.5     21
2    2    5.0   10.0   20.0   23.0     21
3    3    4.0    9.5   19.0   22.0     20
4    4    4.0   10.0   19.5   24.0     20
5    5    4.0   10.0   19.5   20.5     21
6    6    6.0   10.0   21.5   17.0     20
```

Importation des données

Vérifier la moyenne des variables de sites

```
> colMeans(site.covs[, c("Perimeter", "Dist.pond")])
Perimeter Dist.pond
  190.206    29.676
```

Vérifier la moyenne des variables d'échantillonnage

```
> mean(as.vector(as.matrix(airtemp[, -1])))
[1] 15.397
```

Préférable d'avoir des moyennes près de 0

- centrer
- centrer et réduire
- diviser par une constante
- utiliser le log

Formatage des données

Centrer les variables de sites

```
> moy.sites <- colMeans(site.covs[, c("Perimeter",
      "Dist.pond")])
> site.covs$Perim.cent <- site.covs$Perimeter -
      moy.sites[1]
> site.covs$Dist.pond.cent <- site.covs$Dist.pond -
      moy.sites[2]
```

Centrer la température de l'air

```
> air.mean <- mean(as.vector(as.matrix(airtemp[,
      -1])))
> airtemp.cent <- airtemp[, -1] - air.mean
```


Formatage des données

Formater les données avec `unmarkedFrameOccu`

```
> gfrog.data <- unmarkedFrameOccu(y = gfrog[, -1],
                                  siteCovs = site.covs,
                                  obsCovs = list(Airtemp.cent =
                                    airtemp.cent))
```

Points importants :

- `y` doit être une matrice ou un `data.frame` constitué des observations (0, 1)
- `siteCovs` doit être un `data.frame`
- `obsCovs` doit être une liste de `data.frame`

Objet `unmarkedFrameOccu`

Résumé des données

```
> summary(gfrog.data)

unmarkedFrame Object

34 sites
Maximum number of observations per site: 5
Mean number of observations per site: 5
Sites with at least one detection: 18

Tabulation of y observations:
   0    1 <NA>
 138   32    0

Site-level covariates:
      Pond         Perimeter       Dist.pond
 Min.   : 1.00   Min.   : 50.0   Min.   :  0.0
 1st Qu.: 9.25   1st Qu.: 91.2   1st Qu.:  8.0
...
```

Utilisation de occu

Exécutons un modèle nul

```
> ##null
> m0 <- occu(~ 1 ~ 1, data = gfrog.data)
```

Formule double ∼ `Airtemp.cent` ∼ `Dist.pond.cent`

- première formule pour la détection
- deuxième formule pour l'occupation

```
> summary(m0)
Call:
occu(formula = ~1 ~ 1, data = gfrog.data)

Occupancy (logit-scale):
 Estimate    SE    z P(>|z|)
     0.59 0.517 1.14   0.254

Detection (logit-scale):
 Estimate    SE     z P(>|z|)
   -0.883 0.282 -3.13 0.00176

AIC: 162.8
Number of sites: 34
```


Utilisation de occu

Les objets de unmarked sont plus complexes et de type S4

```
> str(m0)


Formal class 'unmarkedFitOccu' [package "unmarked"] with 13 slots
  ..@ knownOcc        : logi [1:34] FALSE FALSE FALSE FALSE F
  ..@ fitType         : chr "occu"
  ..@ call            : language occu(formula = ~1 ~ 1, data = gfro
  ..@ formula         :Class 'formula' length 3 ~1 ~ 1
  .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
  ..@ data            :Formal class 'unmarkedFrameOccu' [package "u
  .. .. ..@ y       : int [1:34, 1:5] 0 0 0 0 0 0 0 0 0 0 ...
  .. .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. .. ..$ : NULL
...
```

On accède au sous-éléments à l'aide de `@` (au lieu de `$`)

Utilisation de occu

Réalisons les mêmes modèles qu'avec PRESENCE

```
> ##psi(.) p(Airtemp.cent)
> m1 <- occu(~ Airtemp.cent  ~ 1, data = gfrog.data)


> ##psi(Dist.pond.cent) p(.)
> m2 <- occu(~ 1 ~ Dist.pond.cent, data = gfrog.data)


> ##psi(Dist.pond.cent) p(Airtemp.cent)
> m3 <- occu(~ Airtemp.cent ~ Dist.pond.cent, data = gfrog.data)
```


Occupation à une saison

Important d'inspecter les sorties

```
> summary(m1)
Call:
occu(formula = ~Airtemp.cent ~ 1, data = gfrog.data)

Occupancy (logit-scale):
 Estimate   SE    z P(>|z|)
    0.475 0.46 1.03   0.302

Detection (logit-scale):
             Estimate     SE     z  P(>|z|)
(Intercept)    -1.164 0.3307 -3.52 0.000434
Airtemp.cent    0.215 0.0562  3.83 0.000130

AIC: 141.99
Number of sites: 34
optim convergence code: 0
optim iterations: 42
Bootstrap iterations: 0
```

Inspection de l'objet unmarkedFitOccu

Important d'inspecter les sorties

```
> summary(m2)
Call:
occu(formula = ~1 ~ Dist.pond.cent, data = gfrog.data)

Occupancy (logit-scale):
               Estimate     SE     z P(>|z|)
(Intercept)      0.6824 0.6511  1.05   0.295
Dist.pond.cent  -0.0347 0.0226 -1.54   0.124

Detection (logit-scale):
 Estimate    SE     z P(>|z|)
   -0.908 0.294 -3.08 0.00204

AIC: 161.16
Number of sites: 34
optim convergence code: 0
optim iterations: 22
Bootstrap iterations: 0
```


Inspection de l'objet unmarkedFitOccu

Important d'inspecter les sorties

```
> summary(m3)
Call:
occu(formula = ~Airtemp.cent ~ Dist.pond.cent, data = gfrog.data)

Occupancy (logit-scale):
               Estimate     SE      z P(>|z|)
(Intercept)      0.4889 0.5106  0.958   0.338
Dist.pond.cent  -0.0311 0.0195 -1.593   0.111

Detection (logit-scale):
             Estimate    SE     z  P(>|z|)
(Intercept)    -1.160 0.328 -3.53 0.000415
Airtemp.cent    0.213 0.056  3.80 0.000144

AIC: 140.57
Number of sites: 34
optim convergence code: 0
optim iterations: 59
Bootstrap iterations: 0
```


Test d'ajustement

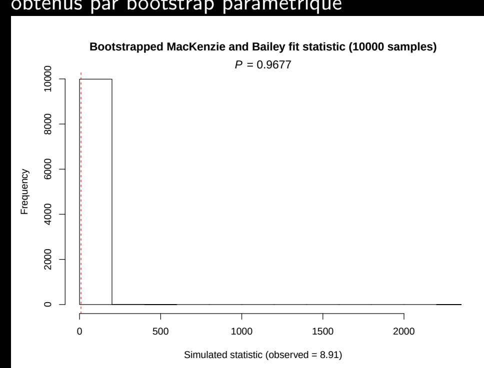
Test de MacKenzie et Bailey pour vérifier l'ajustement du modèle global dans le package AICcmodavg

```
> library(AICcmodavg)
> gof <- mb.gof.test(m3, nsim = 10000, plot.hist = TRUE)

> gof$chisq.table
      Cohort Observed Expected Chi-square
00000      0       16 15.88443  0.0008408
00001      0        3  1.73238  0.9275466
00010      0        3  2.63258  0.0512790
00011      0        1  1.59623  0.2227048
00100      0        2  2.87143  0.2644622
00101      0        1  1.70712  0.2929045
00110      0        3  2.52541  0.0891859
00111      0        3  1.50934  1.4722128
01100      0        1  0.32393  1.4109786
01110      0        1  0.33727  1.3022534
> gof$c.hat.est
[1] 0.32928
```


Test d'ajustement

Graphique de la distribution du χ^2 observé par rapport aux χ^2 obtenus par bootstrap paramétrique



Sélection de modèles

Effectuer la sélection de modèle à l'aide des fonctions du package `AICcmodavg`

```
> ##set up models in list
> Candidates <- list(m0, m1, m2, m3)
> ##assign names to the models
> Names <- c("m0", "m1", "m2", "m3")
```

Ajuster la sélection de modèle par $\hat{c}$ si $\hat{c} > 1$

```
> ##model selection
> aictab(cand.set = Candidates, modnames = Names, c.hat = 1)
Model selection based on AICc :

   K   AICc Delta_AICc AICcWt Cum.Wt     LL
m3 4 141.95       0.00    0.6    0.6 -66.28
m1 3 142.79       0.84    0.4    1.0 -67.99
m2 3 161.96      20.01    0.0    1.0 -77.58
m0 2 163.19      21.25    0.0    1.0 -79.40
```


Inférence multimodèle

Estimé pondéré ($\bar{\hat{\theta}}$) de la température de l'air

```
> modavg(cand.set = Candidates, modnames = Names,
         c.hat = 1, parm = "Airtemp.cent",
         parm.type = "detect")
```

Ajuster les *SE* par $\hat{c}$ si $\hat{c} > 1$

```
Multimodel inference on " p(Airtemp.cent) " based on AICc

 AICc table used to obtain model-averaged estimate:

   K   AICc Delta_AICc AICcWt Estimate   SE
m1 3 142.79       0.84    0.4     0.21 0.06
m3 4 141.95       0.00    0.6     0.21 0.06

Model-averaged estimate: 0.21
Unconditional SE: 0.06
 95 % Unconditional confidence interval: 0.1 , 0.32
```


Inférence multimodèle – prédictions

Pour faire des prédictions pondérées en utilisant tous les modèles :

On crée un nouveau jeu de données

```
> range(airtemp.cent)
[1] -11.897  10.603
> ##create newdata data frame
> ##centered air temperature
> new.dats <- data.frame(Airtemp.cent =
                         seq(from = min(airtemp.cent),
                             to = max(airtemp.cent),
                             length.out = 30))
> ##uncentered air temperature
> new.dats$Airtemp <- new.dats$Airtemp.cent + air.mean
```


Inférence multimodèle – prédictions

On peut ensuite faire les prédictions avec `modavgpred( )`

```
> preds <- modavgpred(cand.set = Candidates, modnames = Names,
                      c.hat = 1, newdata = new.dats,
                      type = "response", parm.type = "detect")

> preds

Model-averaged predictions based on entire model set:

   mod.avg.pred uncond.se
1          0.02      0.02
2          0.03      0.02
3          0.03      0.02
4          0.04      0.03
...
```

On peut ajouter les valeurs prédites et *SE* inconditionnelles

```
> new.dats$mod.avg.pred <- preds$mod.avg.pred
> new.dats$uncond.se <- preds$uncond.se
```

Inférence multimodèle – prédictions

Pour calculer les intervalles de confiance, on utilise l'échelle logit

```
> preds.logit <- modavgpred(cand.set = Candidates,
                            modnames = Names,
                            c.hat = 1, newdata = new.dats,
                            type = "link", parm.type = "detect")
> ##add to data set
> new.dats$mod.avg.pred.logit <- preds.logit$mod.avg.pred
> new.dats$uncond.se.logit <- preds.logit$uncond.se
```

On calcule les limites et on convertit sur l'échelle originale avec `plogis( )`

```
> ##95% CI
> new.dats$low95 <- plogis(new.dats$mod.avg.pred.logit -
                           1.96 * new.dats$uncond.se.logit)
> new.dats$upp95 <- plogis(new.dats$mod.avg.pred.logit +
                           1.96 * new.dats$uncond.se.logit)
```

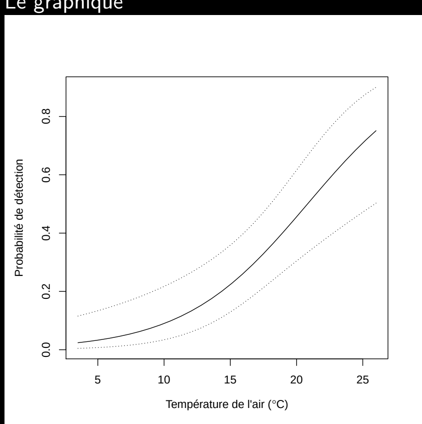

Inférence multimodèle – prédictions

On peut présenter les résultats dans un graphique

```
> ##create plot
> plot(new.dats$mod.avg.pred ~ new.dats$Airtemp,
       ylab = "Probabilité de détection",
       xlab = expression(paste("Température de l'air (",
           degree, "C)")),
       ylim = range(c(new.dats$low95, new.dats$upp95)),
       type = "l")
> ##add confidence bands
> lines(y = new.dats$low95, x = new.dats$Airtemp,
        lty = "dotted")
> lines(y = new.dats$upp95, x = new.dats$Airtemp,
        lty = "dotted")
```


Inférence multimodèle – prédictions

Le graphique



Modèles avec `unmarked`

Plusieurs autres types de modèles dans `unmarked` pour des individus non marqués

- occupation dynamique (multiples saisons)
- modèle d'hétérogénéité de Royle-Nichols
- modèle N-mixture pour décomptes à une saison
- modèle N-mixture dynamique pour décomptes (multiples saisons)
- occupation avec identification erronée lorsqu'espèce n'est pas bien identifiée (*false-positive occupancy model*)
- échantillonnage de la distance (*distance sampling*)

Exemple 2 — Modèle d'occupation dynamique

Occupation dynamique

Le modèle d'occupation dynamique (MacKenzie et al. 2003) permet de quantifier les changements d'occupation entre saisons

Adaptation du modèle robuste de Pollock de CMR pour le cas de l'occupation (périodes primaires, périodes secondaires)

Paramétrisation principale comprend 4 groupes de paramètres

- probabilité d'occupation à la première saison (ψ_{init})
- probabilité de colonisation (γ)
- probabilité d'extinction (ε)
- probabilité de détection (p)

Dans `unmarked`, la fonction `colext( )` estime les paramètres de ce modèle

Occupation dynamique

Ouaouaron (*Lithobates catesbeianus*) et roseau commun


Points d'écoute en 2009 et 2010

- 50 milieux humides de la région de Montréal
- 3 visites par année

Importation des données

On importe les données de détection

```
> bfrog <- read.table("OUA_bin.csv", header = TRUE)
> head(bfrog)
          Location                   Nom V1 V2 V3 V4 V5 V6
1     Arbo Mc gill                 Gill1  0  0  0  0  0  0
2 Beauharnois, cic beau cic 1 bassin haut  0  0  0  0  0  0
3 Beauharnois, cic       beau cic2 chemin  0  0  0  0  0  0
4   Bois de liesse              B-L  elec  0  0  0  0  0  0
5   Bois de liesse              B-L grand  0  0  0  0  0  0
6       ile bizard                barbara  0  0  0  0  0  0
```

Importation des données

On importe les variables de sites

```
> site.covs <- read.table("Site_covariates_centered.csv",
                          header = TRUE)
> head(site.covs)
         Location                    Nom Reed.presence
1     Arbo Mc gill                  Gill1             0
2 Beauharnois, cic beau cic 1 bassin haut             1
3 Beauharnois, cic       beau cic2 chemin             1
4   Bois de liesse              B-L  elec             1
5   Bois de liesse              B-L grand             0
6       ile bizard                barbara             0
  Wetland.area.ha Wetland.area.cent
1        1.011761           0.22707
2        0.314233          -0.47046
3        0.317195          -0.46750
4        0.070684          -0.71401
5        2.425139           1.64045
6        0.476673          -0.30802
```


Importation des données

On importe les variables d'échantillonnage déjà centrées

```
> ##airtemp centered and ordered according to site name
> airtemp <- read.table("Airtemp_centered.csv",
                        header = TRUE)
> ##effort centered and ordered according to site name
> effort <- read.table("Effort_centered.csv",
                       header = TRUE)
> head(airtemp, 3)
  Airtemp1 Airtemp2 Airtemp3 Airtemp4 Airtemp5 Airtemp6
1  -5.7804  -1.5804   6.2796 -5.93044  0.33956   4.9596
2   4.5896   4.2096   4.6096 -0.12044  1.15956   6.5596
3   5.4896   3.9096   4.4396  0.65956  3.17956   8.3496
> head(effort, 3)
  Effort1 Effort2 Effort3 Effort4 Effort5 Effort6
1    0.88    0.88    0.88    0.88    0.88    0.88
2   -1.12   -1.12   -1.12   -1.12   -1.12   -1.12
3   -1.12   -1.12   -1.12   -1.12   -1.12   -1.12
```


Importation des données

On peut créer une variable catégorique année

```
> ##set up yearly site covariate
> year <- matrix(data = c("2009", "2010"),
                 nrow = nrow(site.covs), ncol = 2, byrow = TRUE)
> head(year)
     [,1]   [,2]
[1,] "2009" "2010"
[2,] "2009" "2010"
[3,] "2009" "2010"
[4,] "2009" "2010"
[5,] "2009" "2010"
[6,] "2009" "2010"
```


Formatage des données

On formate les données à l'aide de `unmarkedMultFrame( )`

```
> bfrog.data <- unmarkedMultFrame(y = bfrog[, 3:8],
                                  siteCovs = site.covs[, -c(1,2)],
                                  obsCovs = list(Airtemp =
                                    airtemp, Effort = effort),
                                  yearlySiteCovs =
                                  list(Year = year),
                                  numPrimary = 2)
```

Points importants :

- `siteCovs` doit être un `data.frame`
- `obsCovs` doit être une liste avec des `data.frame`
- `numPrimary` indique le nombre de saisons (périodes primaires)
- `yearlySiteCovs` indique une matrice avec les années

Objet `unmarkedMultFrame`

On peut résumer le jeu de données

```
> summary(bfrog.data)

unmarkedFrame Object

50 sites
Maximum number of observations per site: 6
Mean number of observations per site: 5.88
Number of primary survey periods: 2
Number of secondary survey periods: 3
Sites with at least one detection: 20

Tabulation of y observations:
   0    1 <NA>
 262   32    6

Site-level covariates:
 Reed.presence Wetland.area.ha  Wetland.area.cent
 Min.   :0.0   Min.   :0.0707   Min.   :-0.714
 1st Qu.:0.0   1st Qu.:0.3521   1st Qu.:-0.433
...
```


Avant de lancer les modèles

S'assurer que les variables ne sont pas corrélées entre elles

- éviter d'inclure des variables corrélées ($|r| > 0.7$) sur le même paramètre (ψ_{init}, γ, ε, p)
- possible d'ajouter la même variable sur différents paramètres

On peut formuler différents modèles :

nul : $\psi_{init}.\gamma.\varepsilon.p_{Airtemp+Effort}$

effet additif sur ψ_{init} : $\psi_{init\ Wetland.area+Reed.presence}\gamma.\varepsilon.p_{Airtemp+Effort}$

interaction sur ψ_{init} :

$\psi_{init\ Wetland.area+Reed.presence+Wetland.area:Reed.presence}\gamma.\varepsilon.p_{Airtemp+Effort}$

effet additif sur ε : $\psi_{init}.\gamma.\varepsilon_{Wetland.area+Reed.presence}p_{Airtemp+Effort}$

interaction sur ε :

$\psi_{init}.\gamma.\varepsilon_{Wetland.area+Reed.presence+Wetland.area:Reed.presence}p_{Airtemp+Effort}$

Occupation dynamique – `m0` I

On commence avec le modèle nul

```
> m0 <- colext(psiformula = ~ 1,
               gammaformula = ~ 1,
               epsilonformula = ~ 1,
               pformula = ~ Effort + Airtemp, data = bfrog.data)
> summary(m0)
Call:
colext(psiformula = ~1, gammaformula = ~1, epsilonformula = ~1,
    pformula = ~Effort + Airtemp, data = bfrog.data)

Initial (logit-scale):
 Estimate    SE     z P(>|z|)
    0.129 0.398 0.324   0.746

Colonization (logit-scale):
 Estimate   SE      z P(>|z|)
    -8.93 31.6 -0.282   0.778
```


Occupation dynamique – `m0` II

```
Extinction (logit-scale):
 Estimate   SE     z P(>|z|)
    0.384 0.63 0.608   0.543

Detection (logit-scale):
            Estimate    SE     z P(>|z|)
(Intercept)  -0.9632 0.306 -3.15 0.00164
Effort        0.9070 0.295  3.07 0.00214
Airtemp      -0.0636 0.051 -1.25 0.21232

AIC: 177.24
Number of sites: 50
optim convergence code: 0
optim iterations: 48
Bootstrap iterations: 0
```

Occupation dynamique – m1 I

On fait rouler les autres modèles et on inspecte les sorties

```
> m1 <- colext(psiformula = ~ Wetland.area.cent + Reed.presence,
               gammaformula = ~ 1,
               epsilonformula = ~ 1,
               pformula = ~ Effort + Airtemp, data = bfrog.data)
> summary(m1)
Call:
colext(psiformula = ~Wetland.area.cent + Reed.presence, gammaformul
    epsilonformula = ~1, pformula = ~Effort + Airtemp, data = bfrog

Initial (logit-scale):
                  Estimate   SE     z P(>|z|)
(Intercept)           1.48 1.06  1.40  0.1630
Wetland.area.cent    -1.95 1.03 -1.89  0.0583
Reed.presence        -1.73 1.19 -1.45  0.1468

Colonization (logit-scale):
 Estimate   SE      z P(>|z|)
```


Occupation dynamique – m1 II

```
    -9.33 32.4 -0.288   0.774

Extinction (logit-scale):
 Estimate    SE     z P(>|z|)
    0.419 0.638 0.657   0.511

Detection (logit-scale):
            Estimate     SE     z  P(>|z|)
(Intercept)  -1.1396 0.3234 -3.52 0.000425
Effort        1.0195 0.2811  3.63 0.000287
Airtemp      -0.0594 0.0495 -1.20 0.230086

AIC: 173.53
Number of sites: 50
optim convergence code: 0
optim iterations: 56
Bootstrap iterations: 0
```


Occupation dynamique – m2 I

```
> m2 <- colext(psiformula = ~ Wetland.area.cent + Reed.presence +
               Wetland.area.cent:Reed.presence,
               gammaformula = ~ 1,
               epsilonformula = ~ 1,
               pformula = ~ Effort + Airtemp, data = bfrog.data)
> summary(m2)
Call:
colext(psiformula = ~Wetland.area.cent + Reed.presence + Wetland.ar
    gammaformula = ~1, epsilonformula = ~1, pformula = ~Effort +
        Airtemp, data = bfrog.data)

Initial (logit-scale):
                                Estimate   SE      z
(Intercept)                        1.647 1.26  1.302
Wetland.area.cent                 -2.213 1.32 -1.678
Reed.presence                     -1.875 1.34 -1.401
Wetland.area.cent:Reed.presence    0.844 1.98  0.427
```


Occupation dynamique – m2 II

```
                                P(>|z|)
(Intercept)                      0.1929
Wetland.area.cent                0.0934
Reed.presence                    0.1613
Wetland.area.cent:Reed.presence  0.6697

Colonization (logit-scale):
 Estimate   SE      z P(>|z|)
    -9.33 32.8 -0.284   0.776

Extinction (logit-scale):
 Estimate    SE     z P(>|z|)
    0.422 0.637 0.662   0.508

Detection (logit-scale):
            Estimate     SE     z  P(>|z|)
(Intercept)  -1.1330 0.3197 -3.54 0.000394
Effort        1.0106 0.2814  3.59 0.000329
Airtemp      -0.0593 0.0495 -1.20 0.230664
```

Occupation dynamique – m2 III

```
AIC: 175.35
Number of sites: 50
optim convergence code: 0
optim iterations: 63
Bootstrap iterations: 0
```


Occupation dynamique – m3 I

```
> m3 <- colext(psiformula = ~ 1,
               gammaformula = ~ 1,
               epsilonformula = ~ Wetland.area.cent +
               Reed.presence,
               pformula = ~ Effort + Airtemp, data = bfrog.data)
> summary(m3)
Call:
colext(psiformula = ~1, gammaformula = ~1, epsilonformula = ~Wetlar
    Reed.presence, pformula = ~Effort + Airtemp, data = bfrog.data)

Initial (logit-scale):
 Estimate    SE     z P(>|z|)
    0.175 0.405 0.432   0.666

Colonization (logit-scale):
 Estimate SE      z P(>|z|)
     -8.6 32 -0.268   0.788
```


Occupation dynamique – m3 II

```
Extinction (logit-scale):
                  Estimate    SE      z P(>|z|)
(Intercept)         -0.277 0.902 -0.307   0.759
Wetland.area.cent    0.151 1.459  0.104   0.917
Reed.presence        1.964 1.525  1.287   0.198

Detection (logit-scale):
            Estimate     SE     z P(>|z|)
(Intercept)  -0.9826 0.3105 -3.16 0.00156
Effort        0.9101 0.2964  3.07 0.00213
Airtemp      -0.0599 0.0514 -1.17 0.24359

AIC: 179.16
Number of sites: 50
optim convergence code: 0
optim iterations: 53
Bootstrap iterations: 0
```


Occupation dynamique – m4 I

```
> m4 <- colext(psiformula = ~ 1,
               gammaformula = ~ 1,
               epsilonformula = ~ Wetland.area.cent +
               Reed.presence + Wetland.area.cent:Reed.presence,
               pformula = ~ Effort + Airtemp, data = bfrog.data)
> summary(m4)
Call:
colext(psiformula = ~1, gammaformula = ~1, epsilonformula = ~Wetlar
    Reed.presence + Wetland.area.cent:Reed.presence, pformula = ~Ef
    Airtemp, data = bfrog.data)

Initial (logit-scale):
 Estimate    SE     z P(>|z|)
    0.161 0.406 0.396   0.692

Colonization (logit-scale):
 Estimate  SE       z P(>|z|)
```

Occupation dynamique – m4 II

```
      -22.4 355 -0.0633    0.95

Extinction (logit-scale):
                                Estimate      SE      z
(Intercept)                       -0.203   0.837 -0.243
Wetland.area.cent                 -0.481   1.555 -0.309
Reed.presence                     46.720 104.282  0.448
Wetland.area.cent:Reed.presence  107.027 233.372  0.459
                                P(>|z|)
(Intercept)                       0.808
Wetland.area.cent                 0.757
Reed.presence                     0.654
Wetland.area.cent:Reed.presence   0.647

Detection (logit-scale):
            Estimate     SE     z P(>|z|)
(Intercept)  -0.9655 0.3063 -3.15 0.00162
Effort        0.9160 0.2943  3.11 0.00186
Airtemp      -0.0644 0.0521 -1.24 0.21630
```


Occupation dynamique – m4 III

```
     AIC: 177
     Number of sites: 50
     optim convergence code: 1
     optim iterations: 146
     Bootstrap iterations: 0


Message d'avis :
In function (object)  :
  Model did not converge. Try providing starting values or
  increasing maxit control argment.
```

On remarque le message d'avertissement

- On ne fait pas confiance aux résultats du modèle `m5` (à la poubelle)

Problèmes de convergence

Causes possibles :

- variable sur échelle inappropriée (e.g., 10 000 m^2 au lieu de ha)
- classification parfaite (aucune détection dans certaines classes)
- modèle trop complexe pour les données (ou l'information contenue dans les données)

Signes de problèmes d'estimation ou de convergence :

- β très gros ou très petits ($|20|$)
- *SE* des estimés démesurées par rapport à l'estimé (> 50)
- β ou *SE* avec `NaN`
- message à propos de la matrice hessienne (*Hessian is singular*)

Problèmes de convergence

Autre diagnostic : le nombre de condition (*condition number*, *CN*)

Correspond au ratio des eigenvalues (λ) de la matrice hessienne

$$CN = \max(\lambda)/\min(\lambda)$$

On peut l'obtenir facilement dans R

```
> extract.CN <- function(mod) {
     hess <- mod@opt$hessian
     eigenvals <- eigen(hess)$values
     CN <- max(eigenvals)/min(eigenvals)
     return(CN)
  }
```

CN trop élevé (> 1000000) indique des problèmes

Problèmes de convergence

Vérifions le *CN* pour les modèles

```
> extract.CN(m0)
[1] 389514
> extract.CN(m1)
[1] 437459
> extract.CN(m2)
[1] 449113
> extract.CN(m3)
[1] 402434
> extract.CN(m4)
[1] 48551776
```

On remarque la valeur très élevée pour `m4`

Problèmes de convergence

Possible de modifier le modèle pour réduire les problèmes

- en spécifiant des valeurs initiales avec `starts` en se basant sur valeurs obtenues dans des modèles plus simples
- changer d'algorithme avec `method` (`BFGS`, `Nelder-Mead`, `SANN`)
- en fixant les valeurs de certains paramètres (dans `PRESENCE` et `MARK`, mais pas encore dans `unmarked`)

Toutefois, ne fonctionne pas toujours et on doit mettre le modèle à la poubelle

Test d'ajustement

Tests d'ajustement en développement

- tests d'ajustement sur chaque année séparément à l'aide de modèles à une saison
- test d'ajustement sur chaque année séparément à l'aide du modèle dynamique
- nouvelle fonction à venir dans le package `AICcmodavg` d'ici janvier 2014

Sélection de modèles

On assemble les modèles dans une liste (excluant `m4`)

```
> Cand.mods <- list(m0, m1, m2, m3)
> Mod.names <- c("nul", "psi(Wetland+Reed)", "psi(Wetland*Reed)",
                 "eps(Wetland+Reed)")
```

Sélection de modèles

```
> aictab(cand.set = Cand.mods, modnames = Mod.names)
Model selection based on AICc :

                  K   AICc Delta_AICc AICcWt Cum.Wt     LL
psi(Wetland+Reed) 8 177.04       0.00   0.61   0.61 -78.76
nul               6 179.19       2.16   0.21   0.81 -82.62
psi(Wetland*Reed) 9 179.85       2.82   0.15   0.96 -78.68
eps(Wetland+Reed) 8 182.67       5.63   0.04   1.00 -81.58
```

Le modèle additif sur ψ a le plus de poids

```
> evidence(aictab(cand.set = Cand.mods, modnames = Mod.names))
Evidence ratio between models ' psi(Wetland+Reed) ' and ' nul ':
2.94
```

Inférence multimodèle

Effet de `Wetland.area.cent` sur ψ_{init}

```
> print(modavg(cand.set = Cand.mods, modnames = Mod.names,
         parm = "Wetland.area.cent", parm.type = "psi",
         exclude = list("Wetland.area.cent:Reed.presence")), 4)
Multimodel inference on " psi(Wetland.area.cent) " based on AICc

 AICc table used to obtain model-averaged estimate:

                  K   AICc Delta_AICc AICcWt Estimate
psi(Wetland+Reed) 8 177.04          0      1  -1.9467
                      SE
psi(Wetland+Reed) 1.0281

Model-averaged estimate: -1.9467
Unconditional SE: 1.0281
 95 % Unconditional confidence interval: -3.9617 , 0.0682
```

La probabilité d'occupation initiale du Ouaouaron diminue marginalement avec `Wetland.area.cent`.

Inférence multimodèle

Effet de `Reed.presence` sur ψ_{init}

```
> modavg(cand.set = Cand.mods, modnames = Mod.names,
         parm = "Reed.presence", parm.type = "psi",
         exclude = list("Wetland.area.cent:Reed.presence"))
Multimodel inference on " psi(Reed.presence) " based on AICc

 AICc table used to obtain model-averaged estimate:

                  K   AICc Delta_AICc AICcWt Estimate   SE
psi(Wetland+Reed) 8 177.04          0      1    -1.73 1.19

Model-averaged estimate: -1.73
Unconditional SE: 1.19
 95 % Unconditional confidence interval: -4.06 , 0.61
```

ψ_{init} ne varie pas avec `Reed.presence`.

Inférence multimodèle

Effet de `Airtemp` sur p

```
> modavg(cand.set = Cand.mods, modnames = Mod.names,
         parm = "Airtemp", parm.type = "detect")
Multimodel inference on " p(Airtemp) " based on AICc

 AICc table used to obtain model-averaged estimate:

                  K   AICc Delta_AICc AICcWt Estimate   SE
nul               6 179.19       2.16   0.21    -0.06 0.05
psi(Wetland+Reed) 8 177.04       0.00   0.61    -0.06 0.05
psi(Wetland*Reed) 9 179.85       2.82   0.15    -0.06 0.05
eps(Wetland+Reed) 8 182.67       5.63   0.04    -0.06 0.05

Model-averaged estimate: -0.06
Unconditional SE: 0.05
 95 % Unconditional confidence interval: -0.16 , 0.04
```

La probabilité de détection du Ouaouaron ne varie pas avec `Airtemp`.

Inférence multimodèle

Effet de `Effort` sur p

```
> modavg(cand.set = Cand.mods, modnames = Mod.names,
         parm = "Effort", parm.type = "detect")
Multimodel inference on " p(Effort) " based on AICc

 AICc table used to obtain model-averaged estimate:

                  K   AICc Delta_AICc AICcWt Estimate   SE
nul               6 179.19       2.16   0.21     0.91 0.30
psi(Wetland+Reed) 8 177.04       0.00   0.61     1.02 0.28
psi(Wetland*Reed) 9 179.85       2.82   0.15     1.01 0.28
eps(Wetland+Reed) 8 182.67       5.63   0.04     0.91 0.30

Model-averaged estimate: 0.99
Unconditional SE: 0.29
 95 % Unconditional confidence interval: 0.43 , 1.56
```

La probabilité de détection du Ouaouaron augmente avec `Effort`.

Inférence multimodèle – prédictions

On crée un nouveau jeu de données

```
> range(effort)
[1] -5.12  1.88
> ##create newdata data frame
> ##centered effort
> new.dats <- data.frame(Effort =
                         seq(from = min(effort),
                             to = max(effort),
                             length.out = 30),
                         Airtemp = 0)
> ##uncentered effort
> new.dats$Effort.orig <- new.dats$Effort + 9.12
```

On doit donner une valeur à chaque variable apparaissant dans au moins un modèle

- `Airtemp` et `Effort` sur *p*

On fait varier une variable en gardant les autres constantes

- si on a centré les variables, on utilise 0 (la moyenne)

Inférence multimodèle – prédictions

Prédictions avec `modavgpred( )`

```
> preds <- modavgpred(cand.set = Cand.mods, modnames = Mod.names,
                      c.hat = 1, newdata = new.dats,
                      type = "response", parm.type = "detect")
```

Ajouter les valeurs prédites et *SE* inconditionnelles

```
> new.dats$mod.avg.pred <- preds$mod.avg.pred
> new.dats$uncond.se <- preds$uncond.se
```

Pour calculer les intervalles de confiance, on utilise l'échelle logit

```
> preds.logit <- modavgpred(cand.set = Cand.mods,
                            modnames = Mod.names,
                            c.hat = 1, newdata = new.dats,
                            type = "link", parm.type = "detect")
> ##add to data set
> new.dats$mod.avg.pred.logit <- preds.logit$mod.avg.pred
> new.dats$uncond.se.logit <- preds.logit$uncond.se
```


Inférence multimodèle – prédictions

On calcule les limites et on convertit sur l'échelle originale

```
> ##95% CI
> new.dats$low95 <- plogis(new.dats$mod.avg.pred.logit -
                           1.96 * new.dats$uncond.se.logit)
> new.dats$upp95 <- plogis(new.dats$mod.avg.pred.logit +
                           1.96 * new.dats$uncond.se.logit)
```

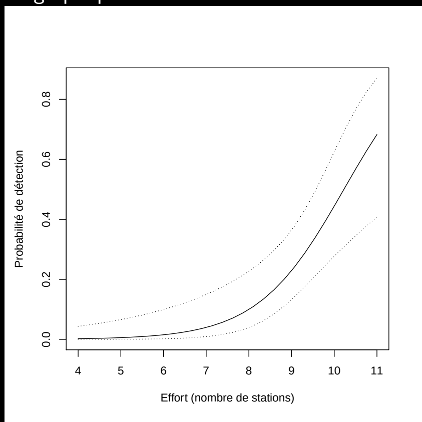

Inférence multimodèle – prédictions

Préparation du graphique

```
> ##create plot
> plot(new.dats$mod.avg.pred ~ new.dats$Effort.orig,
       ylab = "Probabilité de détection",
       xlab = "Effort (nombre de stations)",
       ylim = range(c(new.dats$low95, new.dats$upp95)),
       type = "l")
> ##add confidence bands
> lines(y = new.dats$low95, x = new.dats$Effort.orig,
        lty = "dotted")
> lines(y = new.dats$upp95, x = new.dats$Effort.orig,
        lty = "dotted")
```

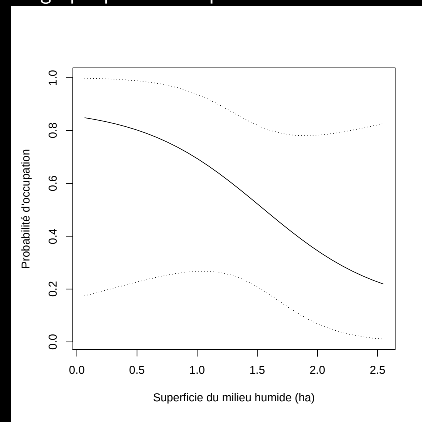
Inférence multimodèle – prédictions

Le graphique de l'effort sur la détection



Inférence multimodèle – prédictions

Le graphique de la superficie du milieu humide sur ψ_{init}



Exemple 3 — Modèle N-mixture

Origine du modèle N-mixture

Développé par Royle (2004)

Principe : le nombre d'individus détectés lors d'un inventaire à un site i au temps t est le résultat d'une expérience binomiale

$$C_{i,t} \sim \text{Binomial}(N_i, p)$$

- où $C_{i,t}$ correspond au nombre d'individus détectés au site i au temps t (c'est le nombre de succès)
- où N_i est le nombre total d'individus présents au site i et correspond au nombre total d'essais
- N_i provient d'une distribution de Poisson ($N \sim Pois(\lambda)$)
- où p correspond à la probabilité de détecter (ou capturer) un individu (P d'observer 1 succès)

Origine du modèle N-mixture

On modélise le processus biologique (N) et le processus d'erreur d'observation (p).

La fonction de vraisemblance :

$$L(p, \theta | n_{i,t}) = \prod_{i=1}^{R} \{ \sum_{N_i = max_t n_{i,t}}^{\infty} (\prod_{t=1}^{T} \text{Binomial}(n_{i,t}; N_i, p) f(N_i; \theta) \}$$

- où on imbrique une distribution $f(N_i; \theta)$ pour les abondances à l'intérieur même d'une distribution binomiale
- on peut modéliser N_i à l'aide la distribution de Poisson, binomiale négative ou Poisson gonflée de 0 (*zero-inflated Poisson*)
- on peut substituer ∞ par une valeur très grande qui représente le nombre maximal possible d'individus qu'on puisse observer à un site (K)

Applications

On estime N_i à partir d'individus non marqués, pourvu qu'on ait au moins deux visites par site.

À chaque visite, on dénombre les individus différents détectés à chacun des sites.

- tout comme pour les modèles d'occupation, une visite peut correspondre à un observateur. Ex. Si trois observateurs échantillonnent indépendamment et en même temps le même site, on a trois visites.

Ce type de modèle nécessite un minimum de sites (> 30) et permet aussi la présence de données manquantes (attention à la sélection de modèles)

Suppositions

Suppositions

- fermeture de la population : il n'y a aucun changement d'abondance entre la première et dernière visite à un site
- la détection des individus est constante à tous les sites, sinon, l'hétérogénéité est modélisée à l'aide de variables appropriées
- la probabilité de détecter (ou capturer) un individu à un site est indépendante pour toutes les visites
- l'abondance est modélisée à l'aide d'une distribution appropriée, soit une distribution de Poisson (`P`), binomiale négative (`NB`) ou Poisson gonflée de 0 (*zero-inflated Poisson*, `ZIP`)

Modèle N-mixture

Abondance du Grand Polatouche (*Glaucomys sabrinus*) en forêt boréale

Piégeage de septembre à décembre 2008 (2 visites)

- 56 sites dans la région de Rouyn-Noranda
- 8 stations par site
- pièges Tomahawk à 1.5 m et 4 m à chaque station

Importation des données

Données de décompte et de sites

```
> ##import data set
> pola <- read.table("pola.txt", header = TRUE)
> head(pola, 3)
  S1V1 S1V2 S1V3 S1V4
1    0    0    0    0
2    0    0    0    0
3    0    0    0    0
> ##import site covariates
> pola.site <- read.table("covsite.txt", header = TRUE)
> head(pola.site, 3)
     Snag  Conifer Boxes
1 -1.7676  1.87572     1
2  5.0368  0.11927     0
3  3.4882 -0.98885     1
```


Importation des variables d'échantillonnage I

Précipitation

```
> ##import prec
> pola.prec <- read.table("obscov.prectot.txt", header = TRUE)
> head(pola.prec, 3)
     S1V1    S1V2    S1V3    S1V4
1 -3.4531 -3.4531 -1.9531 -1.9531
2 -3.4531 -3.4531 -1.9531 -1.9531
3 -3.4531 -3.4531 -1.9531 -1.9531
```

Hauteur du piège

```
> ##import ht
> pola.ht <- read.table("obscov.ht.txt", header = TRUE)
> head(pola.ht, 3)
  S1V1 S1V2 S1V3 S1V4
1    0    1    0    1
2    0    1    0    1
3    0    1    0    1
```


Importation des variables d'échantillonnage II

Jour julien

```
> ##import Jday
> pola.jday <- read.table("Jday.txt", header = TRUE)
> head(pola.jday, 3)
     S1V1    S1V2   S1V3   S1V4
1 -44.571 -44.571 3.4286 3.4286
2 -44.571 -44.571 3.4286 3.4286
3 -44.571 -44.571 3.4286 3.4286
```


Formatage des données I

On utilise la fonction `unmarkedFramePCount( )`

```
> pola.data <- unmarkedFramePCount(y = pola, siteCovs = pola.site,
                                   obsCovs = list(
                                     Prec = pola.prec,
                                     Ht = pola.ht,
                                     Jday = pola.jday))
> summary(pola.data)
unmarkedFrame Object

56 sites
Maximum number of observations per site: 4
Mean number of observations per site: 4
Sites with at least one detection: 33

Tabulation of y observations:
   0    1    2    3    4    5 <NA>
 163   37   17    5    1    1    0
```

Formatage des données II

```
Site-level covariates:
      Snag          Conifer           Boxes
 Min.   :-3.43   Min.   :-1.409   Min.   :0.000
 1st Qu.:-2.53   1st Qu.:-1.235   1st Qu.:0.000
 Median :-0.73   Median :-0.360   Median :1.000
 Mean   : 0.00   Mean   : 0.000   Mean   :0.518
 3rd Qu.: 1.19   3rd Qu.: 0.706   3rd Qu.:1.000
 Max.   : 9.72   Max.   : 4.499   Max.   :1.000

Observation-level covariates:
      Prec               Ht            Jday
 Min.   :-7.953   Min.   :0.0   Min.   :-44.571
 1st Qu.:-5.953   1st Qu.:0.0   1st Qu.:-23.571
 Median :-1.953   Median :0.5   Median :  0.429
 Mean   :-0.868   Mean   :0.5   Mean   :  0.000
 3rd Qu.: 5.172   3rd Qu.:1.0   3rd Qu.: 18.429
 Max.   : 8.547   Max.   :1.0   Max.   : 40.429
```


Formulation du modèle *N*-mixture I

On peut formuler différents modèles :

m0 : $\lambda_. p_.$
m1 : $\lambda_. p_{Prec+Jday+Ht}$
m2 : $\lambda_{Snag} p_{Prec+Jday+Ht}$
m3 : $\lambda_{Conifer} p_{Prec+Jday+Ht}$
m4 : $\lambda_{Snag+Conifer} p_{Prec+Jday+Ht}$
m5 : $\lambda_. p_{Snag+Conifer}$
m6 : $\lambda_{Snag} p_{Snag+Conifer}$
m7 : $\lambda_{Conifer} p_{Snag+Conifer}$
m8 : $\lambda_{Snag+Conifer} p_{Snag+Conifer}$

Formulation du modèle *N*-mixture I

Utilisation de `pcount( )`
Commençons avec un modèle qui utilise une distribution de Poisson

```
> m0 <- pcount(~ 1 ~ 1, K = 40, mixture = "P", data = pola.data)
```

Points importants :

- double formule ($\sim p \sim \lambda$)
- `K` correspond au nombre maximum d'individus qu'on peut voir dans un site
  - on peut faire varier `K` jusqu'à ce que le $\log \mathscr{L}$ se stabilise
- si p est faible, peut donner des abondances trop élevées
- `mixture` correspond à la distribution pour l'abondance (Poisson ici)

Structure de l´objet `unmarkedFitPCount`

Pour le modèle `m0`

```
> str(m0)

Formal class 'unmarkedFitPCount' [package "unmarked"] with 14 slots
  ..@ K             : num 40
  ..@ mixture       : chr "P"
  ..@ fitType       : chr "pcount"
  ..@ call          : language pcount(formula = ~1 ~ 1, data = po
  ..@ formula       :Class 'formula' length 3 ~1 ~ 1
  .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
  ..@ data          :Formal class 'unmarkedFramePCount' [package
  .. .. ..@ y       : int [1:56, 1:4] 0 0 0 0 0 0 1 0 0 1 ...
  .. .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. .. ..$ : NULL
  .. .. .. .. ..$ : chr [1:4] "S1V1" "S1V2" "S1V3" "S1V4"
  .. .. ..@ obsCovs :'data.frame': 224 obs. of  3 variables:
  .. .. .. ..$ Prec: num [1:224] -3.45 -3.45 -1.95 -1.95 -3.45 ...
  .. .. .. ..$ Ht  : int [1:224] 0 1 0 1 0 1 0 1 0 1 ...
  .. .. .. ..$ Jday: num [1:224] -44.57 -44.57 3.43 3.43 -44.57 ...
  .. .. ..@ siteCovs:'data.frame': 56 obs. of  3 variables:
...
```

Inspection de m0

```
> summary(m0)
Call:
pcount(formula = ~1 ~ 1, data = pola.data, K = 40, mixture = "P")

Abundance (log-scale):
 Estimate    SE    z P(>|z|)
    0.618 0.243 2.54  0.0111

Detection (logit-scale):
 Estimate    SE     z  P(>|z|)
    -1.22 0.311 -3.91 9.34e-05

AIC: 391.05
Number of sites: 56
optim convergence code: 0
optim iterations: 22
Bootstrap iterations: 0
```


Lancement des autres modèles

```
> m1 <- pcount(~ Prec + Jday + Ht ~ 1,
               K = 40, mixture = "P", data = pola.data)
> m2 <- pcount(~ Prec + Jday + Ht ~ Snag,
               K = 40, mixture = "P", data = pola.data)
> m3 <- pcount(~ Prec + Jday + Ht ~ Conifer,
               K = 40, mixture = "P", data = pola.data)
> m4 <- pcount(~ Prec + Jday + Ht ~ Snag + Conifer,
               K = 40, mixture = "P", data = pola.data)
> m5 <- pcount(~ Snag + Conifer ~ 1, K = 40,
               mixture = "P", data = pola.data)
> m6 <- pcount(~ Snag + Conifer ~ Snag, K = 40,
               mixture = "P", data = pola.data)
> m7 <- pcount(~ Snag + Conifer ~ Conifer, K = 40,
               mixture = "P", data = pola.data)
> m8 <- pcount(~ Snag + Conifer ~ Snag + Conifer,
               K = 40, mixture = "P", data = pola.data)
```


Inspection de m1 I

```
> summary(m1)
Call:
pcount(formula = ~Prec + Jday + Ht ~ 1, data = pola.data, K = 40,
    mixture = "P")

Abundance (log-scale):
 Estimate    SE    z P(>|z|)
    0.502 0.206 2.43  0.0151

Detection (logit-scale):
            Estimate      SE     z P(>|z|)
(Intercept)  -0.9835 0.32223 -3.05 0.00227
Prec         -0.0812 0.02546 -3.19 0.00143
Jday         -0.0149 0.00613 -2.43 0.01498
Ht           -0.4599 0.25198 -1.83 0.06797

AIC: 372.48
```


Inspection de m1 II

```
Number of sites: 56
optim convergence code: 0
optim iterations: 48
Bootstrap iterations: 0
```

Inspection de m2 I

```
> summary(m2)
Call:
pcount(formula = ~Prec + Jday + Ht ~ Snag, data = pola.data,
    K = 40, mixture = "P")

Abundance (log-scale):
            Estimate    SE     z P(>|z|)
(Intercept)    0.502 0.242  2.07 0.03848
Snag          -0.185 0.057 -3.25 0.00117

Detection (logit-scale):
            Estimate     SE     z P(>|z|)
(Intercept)  -1.1588 0.3530 -3.28 0.00103
Prec         -0.0781 0.0247 -3.16 0.00158
Jday         -0.0159 0.0059 -2.69 0.00706
Ht           -0.4388 0.2462 -1.78 0.07469
```


Inspection de m2 II

```
AIC: 361.11
Number of sites: 56
optim convergence code: 0
optim iterations: 53
Bootstrap iterations: 0
```


Inspection de m3 I

```
> summary(m3)
Call:
pcount(formula = ~Prec + Jday + Ht ~ Conifer, data = pola.data,
    K = 40, mixture = "P")

Abundance (log-scale):
            Estimate     SE    z P(>|z|)
(Intercept)    0.470 0.2085 2.26  0.0241
Conifer        0.121 0.0861 1.41  0.1590

Detection (logit-scale):
            Estimate      SE     z P(>|z|)
(Intercept)  -0.9591 0.32300 -2.97 0.00299
Prec         -0.0776 0.02565 -3.03 0.00248
Jday         -0.0147 0.00617 -2.39 0.01707
Ht           -0.4609 0.25234 -1.83 0.06776
```


Inspection de m3 II

```
AIC: 372.6
Number of sites: 56
optim convergence code: 0
optim iterations: 40
Bootstrap iterations: 0
```

Inspection de m4 I

```
> summary(m4)
Call:
pcount(formula = ~Prec + Jday + Ht ~ Snag + Conifer, data = pola.da
    K = 40, mixture = "P")

Abundance (log-scale):
            Estimate     SE     z P(>|z|)
(Intercept)    0.468 0.2460  1.90 0.05721
Snag          -0.188 0.0572 -3.28 0.00102
Conifer        0.128 0.0853  1.51 0.13187

Detection (logit-scale):
            Estimate      SE     z P(>|z|)
(Intercept)  -1.1350 0.35590 -3.19 0.00143
Prec         -0.0743 0.02491 -2.98 0.00287
Jday         -0.0156 0.00594 -2.62 0.00880
Ht           -0.4393 0.24644 -1.78 0.07462
```


Inspection de m4 II

```
AIC: 360.97
Number of sites: 56
optim convergence code: 0
optim iterations: 47
Bootstrap iterations: 0
```


Inspection de m5 I

```
> summary(m5)
Call:
pcount(formula = ~Snag + Conifer ~ 1, data = pola.data, K = 40,
    mixture = "P")

Abundance (log-scale):
 Estimate    SE    z P(>|z|)
     1.04 0.332 3.14 0.00168

Detection (logit-scale):
            Estimate     SE     z  P(>|z|)
(Intercept)   -1.901 0.4042 -4.70 2.56e-06
Snag          -0.216 0.0620 -3.49 4.84e-04
Conifer        0.158 0.0976  1.62 1.06e-01

AIC: 380.9
Number of sites: 56
```


Inspection de m5 II

```
optim convergence code: 0
optim iterations: 35
Bootstrap iterations: 0
```

Inspection de m6 I

```
> summary(m6)
Call:
pcount(formula = ~Snag + Conifer ~ Snag, data = pola.data, K = 40,
    mixture = "P")

Abundance (log-scale):
            Estimate    SE      z P(>|z|)
(Intercept)   0.8592 0.510  1.686  0.0919
Snag         -0.0921 0.227 -0.406  0.6847

Detection (logit-scale):
            Estimate    SE      z P(>|z|)
(Intercept)   -1.698 0.587 -2.892 0.00382
Snag          -0.119 0.251 -0.474 0.63547
Conifer        0.145 0.105  1.380 0.16771

AIC: 382.78
```


Inspection de m6 II

```
Number of sites: 56
optim convergence code: 0
optim iterations: 59
Bootstrap iterations: 0
```


Inspection de m7 I

```
> summary(m7)
Call:
pcount(formula = ~Snag + Conifer ~ Conifer, data = pola.data,
    K = 40, mixture = "P")

Abundance (log-scale):
            Estimate    SE    z P(>|z|)
(Intercept)    0.921 0.318 2.89 0.00382
Conifer        0.336 0.254 1.32 0.18688

Detection (logit-scale):
            Estimate     SE      z  P(>|z|)
(Intercept)   -1.751 0.3892 -4.498 6.85e-06
Snag          -0.199 0.0621 -3.208 1.34e-03
Conifer       -0.192 0.2838 -0.678 4.98e-01

AIC: 380.57
```


Inspection de m7 II

```
Number of sites: 56
optim convergence code: 0
optim iterations: 53
Bootstrap iterations: 0
```

Inspection de m8 I

```
> summary(m8)
Call:
pcount(formula = ~Snag + Conifer ~ Snag + Conifer, data = pola.data
    K = 40, mixture = "P")

Abundance (log-scale):
            Estimate    SE     z P(>|z|)
(Intercept)    0.685 0.336  2.04  0.0414
Snag          -0.180 0.109 -1.66  0.0979
Conifer        0.458 0.291  1.58  0.1150

Detection (logit-scale):
            Estimate    SE       z  P(>|z|)
(Intercept) -1.47726 0.405 -3.6452 0.000267
Snag        -0.00195 0.126 -0.0155 0.987653
Conifer     -0.34382 0.316 -1.0872 0.276962
```


Inspection de m8 II

```
AIC: 381.06
Number of sites: 56
optim convergence code: 0
optim iterations: 64
Bootstrap iterations: 0
```


Autres diagnostics

On peut vérifier le *CN* pour les modèles

On met les modèles dans une liste

```
> Cands.pois <- list(m0, m1, m2, m3, m4, m5, m6, m7, m8)
> Mod.pois.names <- c("m0", "m1", "m2", "m3", "m4", "m5",
                      "m6", "m7", "m8")
> ##add names to list
> names(Cands.pois) <- Mod.pois.names
> unlist(lapply(Cands.pois, FUN = extract.CN))
      m0       m1       m2       m3       m4       m5
  13.075 4244.329 5391.343 4277.728 5513.127   82.540
      m6       m7       m8
 436.959   88.986  231.858
```

Aucune valeur démesurément élevée

Test d'ajustement

On peut utiliser un test d'ajustement basé sur le χ^2 à l'aide de la fonction `Nmix.gof.test( )`

1. on calcule le χ^2 observé sur les données originales
2. on simule les données à partir du modèle, on lance le modèle, et on calcule un χ^2 pour ces données simulées
3. on répète les étapes 1 – 3 un grand nombre de fois
4. on détermine la $P(\chi^2_{obs} \geq \chi^2_{sim}) = \frac{\text{Nombre}\, \chi^2_{sim} \geq \chi^2_{obs}}{\text{Nombre simulations}}$

Aussi possible de calculer un $\hat{c}$,

$$\hat{c} = \frac{\chi^2_{obs}}{\chi^{\bar{2}}_{obs}}$$

Test d'ajustement I

Ici, nous n'avons pas de modèle global qui englobe toutes les variables (trop petit n)

Dans ce cas, il est possible d'utiliser le modèle au premier rang pour vérifier l'ajustement :

```
> aictab(Cands.pois, Mod.pois.names)
Model selection based on AICc :

   K   AICc Delta_AICc AICcWt Cum.Wt      LL
m2 6 362.82       0.00   0.56   0.56 -174.56
m4 7 363.31       0.48   0.44   1.00 -173.49
m1 5 373.68      10.86   0.00   1.00 -181.24
m3 6 374.32      11.49   0.00   1.00 -180.30
m5 4 381.68      18.86   0.00   1.00 -186.45
m7 5 381.77      18.95   0.00   1.00 -185.29
m8 6 382.77      19.95   0.00   1.00 -184.53
```


Test d'ajustement II

```
m6 5 383.98      21.15   0.00   1.00 -186.39
m0 2 391.28      28.46   0.00   1.00 -193.53
> coef(m2)
 lam(Int) lam(Snag)    p(Int)   p(Prec)   p(Jday)     p(Ht)
 0.501595 -0.185103 -1.158787 -0.078084 -0.015905 -0.438771
```

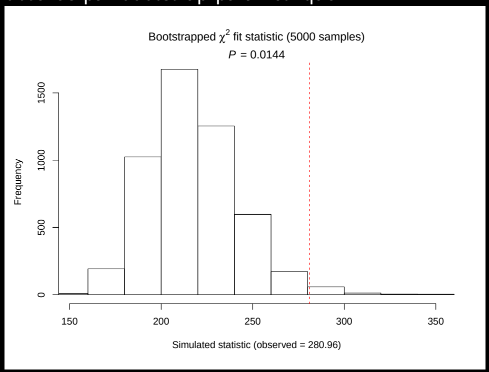
Utilisons le modèle m2 pour vérifier l'ajustement

```
> m2.gof <- Nmix.gof.test(mod = m2, nsim = 5000)


> m2.gof$chi.square
[1] 280.96
> m2.gof$c.hat.est
[1] 1.2915
```


Test d'ajustement III

Graphique de la distribution du χ^2 observé par rapport aux χ^2 obtenus par bootstrap paramétrique



Test d'ajustement IV

On remarque que 72 valeurs de χ^2 simulées sont $\geq \chi^2$ ($P = 0.0144$).

Même si le $\hat{c} = 1.29$, le test d'ajustement indique que le modèle ne s'ajuste pas bien aux données.

On conclut qu'on ne peut continuer avec des modèles qui utilisent la distribution de Poisson avec ce jeu de données.

Modèles N-mixture avec binomiale négative

Essayons des modèles avec la même structure sur p et λ avec une distribution binomiale négative

```
> m0.nb <- pcount(~ 1 ~ 1, K = 40, mixture = "NB",
                  data = pola.data)
> m1.nb <- pcount(~ Prec + Jday + Ht ~ 1, K = 40,
                  mixture = "NB", data = pola.data)
> m2.nb <- pcount(~ Prec + Jday + Ht ~ Snag, K = 40,
                  mixture = "NB", data = pola.data)
> m3.nb <- pcount(~ Prec + Jday + Ht ~ Conifer, K = 40,
                  mixture = "NB", data = pola.data)
> m4.nb <- pcount(~ Prec + Jday + Ht ~ Snag + Conifer, K = 40,
                  mixture = "NB", data = pola.data)
> m5.nb <- pcount(~ Snag + Conifer ~ 1, K = 40,
                  mixture = "NB", data = pola.data)
> m6.nb <- pcount(~ Snag + Conifer ~ Snag, K = 40,
                  mixture = "NB", data = pola.data)
> m7.nb <- pcount(~ Snag + Conifer ~ Conifer, K = 40,
                  mixture = "NB", data = pola.data)
> m8.nb <- pcount(~ Snag + Conifer ~ Snag + Conifer, K = 40,
                  mixture = "NB", data = pola.data)
```


Inspectons la sortie du modèle m0 I

```
> summary(m0.nb)
Call:
pcount(formula = ~1 ~ 1, data = pola.data, K = 40, mixture = "NB")

Abundance (log-scale):
 Estimate    SE    z  P(>|z|)
     1.68 0.432 3.89 9.99e-05

Detection (logit-scale):
 Estimate    SE     z  P(>|z|)
    -2.45 0.454 -5.41 6.34e-08

Dispersion (log-scale):
 Estimate    SE       z P(>|z|)
 -0.00622 0.378 -0.0165   0.987

AIC: 378.35
```


Inspectons la sortie du modèle m0 II

```
Number of sites: 56
optim convergence code: 0
optim iterations: 30
Bootstrap iterations: 0
```

On remarque qu'un paramètre additionnel a été estimé — un paramètre de dispersion pour binomiale négative (permet de modéliser la variance excédentaire par rapport à la Poisson)

Inspectons les sorties des autres modèles I

```
> summary(m1.nb)
Call:
pcount(formula = ~Prec + Jday + Ht ~ 1, data = pola.data, K = 40,
    mixture = "NB")

Abundance (log-scale):
 Estimate    SE    z P(>|z|)
     1.25 0.852 1.47   0.142

Detection (logit-scale):
            Estimate      SE     z P(>|z|)
(Intercept)  -1.9442 1.00169 -1.94 0.05227
Prec         -0.0666 0.02503 -2.66 0.00782
Jday         -0.0127 0.00597 -2.13 0.03324
Ht           -0.3724 0.23072 -1.61 0.10653

Dispersion (log-scale):
```

Inspectons les sorties des autres modèles II

```
 Estimate    SE     z P(>|z|)
    0.118 0.492 0.239   0.811

AIC: 365.52
Number of sites: 56
optim convergence code: 0
optim iterations: 77
Bootstrap iterations: 0
> summary(m2.nb)
Call:
pcount(formula = ~Prec + Jday + Ht ~ Snag, data = pola.data,
    K = 40, mixture = "NB")

Abundance (log-scale):
            Estimate     SE     z P(>|z|)
(Intercept)    1.081 0.7280  1.48 0.13763
Snag          -0.199 0.0668 -2.98 0.00285
```


Inspectons les sorties des autres modèles III

```
Detection (logit-scale):
            Estimate      SE     z P(>|z|)
(Intercept)  -1.9142 0.86673 -2.21 0.02720
Prec         -0.0675 0.02446 -2.76 0.00576
Jday         -0.0141 0.00579 -2.44 0.01467
Ht           -0.3747 0.23025 -1.63 0.10369

Dispersion (log-scale):
 Estimate    SE    z P(>|z|)
     0.57 0.593 0.96   0.337

AIC: 357.73
Number of sites: 56
optim convergence code: 0
optim iterations: 45
Bootstrap iterations: 0
> summary(m3.nb)
```


Inspectons les sorties des autres modèles IV

```
Call:
pcount(formula = ~Prec + Jday + Ht ~ Conifer, data = pola.data,
    K = 40, mixture = "NB")

Abundance (log-scale):
            Estimate    SE     z P(>|z|)
(Intercept)   1.1655 0.787 1.482   0.138
Conifer       0.0946 0.116 0.814   0.415

Detection (logit-scale):
            Estimate     SE     z P(>|z|)
(Intercept)  -1.8547 0.9352 -1.98 0.04735
Prec         -0.0656 0.0251 -2.62 0.00892
Jday         -0.0126 0.0060 -2.11 0.03522
Ht           -0.3770 0.2321 -1.62 0.10437

Dispersion (log-scale):
 Estimate    SE     z P(>|z|)
     0.18 0.519 0.347   0.729
```


Inspectons les sorties des autres modèles V

```
AIC: 366.86
Number of sites: 56
optim convergence code: 0
optim iterations: 43
Bootstrap iterations: 0
> summary(m4.nb)
Call:
pcount(formula = ~Prec + Jday + Ht ~ Snag + Conifer, data = pola.da
    K = 40, mixture = "NB")

Abundance (log-scale):
            Estimate     SE     z P(>|z|)
(Intercept)    0.993 0.6729  1.48 0.14003
Snag          -0.207 0.0678 -3.05 0.00229
Conifer        0.124 0.1071  1.16 0.24788

Detection (logit-scale):
```

Inspectons les sorties des autres modèles VI

```
            Estimate      SE     z P(>|z|)
(Intercept)  -1.8309 0.81228 -2.25 0.02420
Prec         -0.0655 0.02452 -2.67 0.00755
Jday         -0.0139 0.00581 -2.40 0.01644
Ht           -0.3790 0.23144 -1.64 0.10150

Dispersion (log-scale):
 Estimate    SE    z P(>|z|)
    0.672 0.631 1.06   0.287

AIC: 358.41
Number of sites: 56
optim convergence code: 0
optim iterations: 50
Bootstrap iterations: 0
> summary(m5.nb)
```


Inspectons les sorties des autres modèles VII

```
Call:
pcount(formula = ~Snag + Conifer ~ 1, data = pola.data, K = 40,
    mixture = "NB")

Abundance (log-scale):
 Estimate    SE    z  P(>|z|)
     1.95 0.413 4.74 2.18e-06

Detection (logit-scale):
            Estimate     SE     z  P(>|z|)
(Intercept)   -2.907 0.4403 -6.60 4.04e-11
Snag          -0.193 0.0664 -2.91 3.56e-03
Conifer        0.153 0.1145  1.34 1.80e-01

Dispersion (log-scale):
 Estimate   SE     z P(>|z|)
    0.415 0.45 0.922   0.357

AIC: 373.01
```


Inspectons les sorties des autres modèles VIII

```
Number of sites: 56
optim convergence code: 0
optim iterations: 56
Bootstrap iterations: 0
> summary(m6.nb)
Call:
pcount(formula = ~Snag + Conifer ~ Snag, data = pola.data, K = 40,
    mixture = "NB")

Abundance (log-scale):
            Estimate    SE      z P(>|z|)
(Intercept)   1.9021 0.722  2.633 0.00847
Snag         -0.0307 0.293 -0.105 0.91656

Detection (logit-scale):
            Estimate    SE      z  P(>|z|)
(Intercept)   -2.853 0.754 -3.786 0.000153
Snag          -0.163 0.300 -0.544 0.586763
```


Inspectons les sorties des autres modèles IX

```
Conifer        0.153 0.115  1.329 0.183849

Dispersion (log-scale):
 Estimate    SE     z P(>|z|)
    0.411 0.452 0.911   0.362

AIC: 375
Number of sites: 56
optim convergence code: 0
optim iterations: 60
Bootstrap iterations: 0
> summary(m7.nb)
```

Inspectons les sorties des autres modèles X

```
Call:
pcount(formula = ~Snag + Conifer ~ Conifer, data = pola.data,
    K = 40, mixture = "NB")

Abundance (log-scale):
            Estimate    SE     z P(>|z|)
(Intercept)    1.877 0.533 3.521 0.00043
Conifer        0.177 0.233 0.759 0.44775

Detection (logit-scale):
            Estimate     SE       z  P(>|z|)
(Intercept)  -2.8242 0.5711 -4.9457 7.59e-07
Snag         -0.1912 0.0661 -2.8908 3.84e-03
Conifer      -0.0214 0.2389 -0.0895 9.29e-01

Dispersion (log-scale):
 Estimate   SE     z P(>|z|)
     0.42 0.46 0.914   0.361
```


Inspectons les sorties des autres modèles XI

```
AIC: 374.64
Number of sites: 56
optim convergence code: 0
optim iterations: 59
Bootstrap iterations: 0
> summary(m8.nb)
Call:
pcount(formula = ~Snag + Conifer ~ Snag + Conifer, data = pola.data
    K = 40, mixture = "NB")

Abundance (log-scale):
            Estimate    SE     z P(>|z|)
(Intercept)    1.225 0.468  2.62  0.0089
Snag          -0.267 0.143 -1.87  0.0612
Conifer        0.297 0.187  1.59  0.1120

Detection (logit-scale):
            Estimate    SE      z  P(>|z|)
```


Inspectons les sorties des autres modèles XII

```
(Intercept)  -2.1321 0.521 -4.092 4.27e-05
Snag          0.0801 0.150  0.535 5.93e-01
Conifer      -0.1490 0.193 -0.774 4.39e-01

Dispersion (log-scale):
 Estimate    SE    z P(>|z|)
    0.537 0.495 1.09   0.277

AIC: 375.78
Number of sites: 56
optim convergence code: 0
optim iterations: 73
Bootstrap iterations: 0
```


Autres diagnostics

On peut vérifier le *CN* de chaque modèle

```
> Cands.nb <- list(m0.nb, m1.nb, m2.nb, m3.nb, m4.nb,
                       m5.nb, m6.nb, m7.nb, m8.nb)
> Mod.nb.names <- c("m0.nb", "m1.nb", "m2.nb", "m3.nb",
                    "m4.nb", "m5.nb", "m6.nb", "m7.nb", "m8.nb")
> ##add names to list
> names(Cands.nb) <- Mod.nb.names
> unlist(lapply(Cands.nb, extract.CN))
    m0.nb     m1.nb     m2.nb     m3.nb     m4.nb     m5.nb
   22.473 62580.444 48076.937 54249.669 42719.266    94.433
    m6.nb     m7.nb     m8.nb
  615.940   167.546   235.758
```

Certains modèles ont un *CN* un peu élevé.

Test d'ajustement I

On détermine le modèle au premier rang pour vérifier l'ajustement :

```
> aictab(Cands.nb, Mod.nb.names)
Model selection based on AICc :

      K   AICc Delta_AICc AICcWt Cum.Wt      LL
m2.nb 7 360.07       0.00   0.65   0.65 -171.87
m4.nb 8 361.47       1.41   0.32   0.97 -171.21
m1.nb 6 367.24       7.17   0.02   0.99 -176.76
m3.nb 7 369.19       9.13   0.01   1.00 -176.43
m5.nb 5 374.21      14.15   0.00   1.00 -181.51
m7.nb 6 376.35      16.29   0.00   1.00 -181.32
m6.nb 6 376.71      16.64   0.00   1.00 -181.50
m8.nb 7 378.11      18.04   0.00   1.00 -180.89
m0.nb 3 378.82      18.75   0.00   1.00 -186.18
> coef(m2.nb)
```


Test d'ajustement II

```
   lam(Int)    lam(Snag)       p(Int)      p(Prec)
   1.080872    -0.199411    -1.914218    -0.067535
    p(Jday)        p(Ht) alpha(alpha)
  -0.014120    -0.374657     0.569555
```

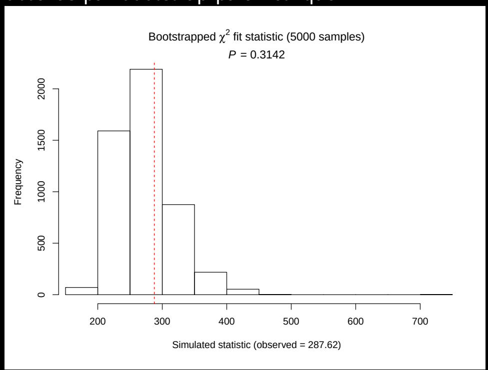
Utilisons le modèle `m2.nb` pour vérifier l'ajustement

```
> m2.nb.gof <- Nmix.gof.test(mod = m2.nb, nsim = 5000)
```

```
> m2.nb.gof$chi.square
[1] 287.62
> m2.nb.gof$c.hat.est
[1] 1.0554
```


Test d'ajustement III

Graphique de la distribution du χ^2 observé par rapport aux χ^2 obtenus par bootstrap paramétrique



Test d'ajustement IV

On remarque que le modèle semble très bien s'ajuster aux données : plusieurs χ^2 simulées sont similaires à ce qu'on obtient avec les données originales ($P = 0.3142$).

De plus, le $\hat{c} = 1.06$, qui est pratiquement 1.

On conclut qu'on peut utiliser la distribution binomiale négative avec ce jeu de données.

Attention :

- la distribution binomiale négative donne parfois des valeurs prédites d'abondance irréalistes (trop grandes)

Test d'ajustement V

- toujours utile de vérifier les valeurs prédites par le modèle pour voir si elles sont réalistes

```
> ##abondances prédites
> abond <- predict(m2.nb, type = "state", backTransform = TRUE)
> head(abond)
  Predicted      SE   lower   upper
1   4.19269 3.07888 0.99407 17.6835
2   1.07948 0.87481 0.22050  5.2848
3   1.47002 1.13263 0.32470  6.6552
4   4.11233 3.01641 0.97662 17.3160
5   3.30441 2.40553 0.79330 13.7641
6   0.63594 0.57400 0.10843  3.7300
> range(abond$Predicted)
[1] 0.42439 5.84532
```


Test d'ajustement VI

Les valeurs prédites d'abondance sont réalistes pour l'espèce : 0 – 6 Grands Polatouches

Sélection de modèles

On effectue la sélection de modèles :

```
> aictab(Cands.nb, Mod.nb.names)
Model selection based on AICc :

      K   AICc Delta_AICc AICcWt Cum.Wt      LL
m2.nb 7 360.07       0.00   0.65   0.65 -171.87
m4.nb 8 361.47       1.41   0.32   0.97 -171.21
m1.nb 6 367.24       7.17   0.02   0.99 -176.76
m3.nb 7 369.19       9.13   0.01   1.00 -176.43
m5.nb 5 374.21      14.15   0.00   1.00 -181.51
m7.nb 6 376.35      16.29   0.00   1.00 -181.32
m6.nb 6 376.71      16.64   0.00   1.00 -181.50
m8.nb 7 378.11      18.04   0.00   1.00 -180.89
m0.nb 3 378.82      18.75   0.00   1.00 -186.18
```

On remarque deux modèles avec un poids d'Akaike cumulatif de 0.97.

Sélection de modèles

Les deux modèles sont constitués des variables suivantes :

```
> coef(m2.nb)
    lam(Int)    lam(Snag)       p(Int)      p(Prec)
    1.080872    -0.199411    -1.914218    -0.067535
     p(Jday)        p(Ht) alpha(alpha)
   -0.014120    -0.374657     0.569555
> coef(m4.nb)
    lam(Int)    lam(Snag) lam(Conifer)       p(Int)
    0.993042    -0.206651     0.123739    -1.830882
     p(Prec)      p(Jday)        p(Ht) alpha(alpha)
   -0.065504    -0.013930    -0.379015     0.671762
```

La variable Snag apparaît dans les deux modèles sur λ.

Inférence multimodèle – Snag

On commence avec Snag sur λ

```
> modavg(cand.set = Cands.nb, modnames = Mod.nb.names,
         parm = "Snag", parm.type = "lambda")
Multimodel inference on " lam(Snag) " based on AICc

 AICc table used to obtain model-averaged estimate:

     K   AICc Delta_AICc AICcWt Estimate   SE
m2.nb 7 360.07       0.00   0.67    -0.20 0.07
m4.nb 8 361.47       1.41   0.33    -0.21 0.07
m6.nb 6 376.71      16.64   0.00    -0.03 0.29
m8.nb 7 378.11      18.04   0.00    -0.27 0.14

Model-averaged estimate: -0.2
Unconditional SE: 0.07
 95 % Unconditional confidence interval: -0.33 , -0.07
```

L'abondance diminue avec Snag.

Inférence multimodèle – Conifer

Pour l'estimé pondéré de Conifer sur λ

```
> modavg(cand.set = Cands.nb, modnames = Mod.nb.names,
         parm = "Conifer", parm.type = "lambda")
Multimodel inference on " lam(Conifer) " based on AICc

 AICc table used to obtain model-averaged estimate:

     K   AICc Delta_AICc AICcWt Estimate   SE
m3.nb 7 369.19       7.72   0.02     0.09 0.12
m4.nb 8 361.47       0.00   0.98     0.12 0.11
m7.nb 6 376.35      14.88   0.00     0.18 0.23
m8.nb 7 378.11      16.63   0.00     0.30 0.19

Model-averaged estimate: 0.12
Unconditional SE: 0.11
 95 % Unconditional confidence interval: -0.09 , 0.33
```

L'abondance ne varie pas avec Conifer.

Inférence multimodèle – Prec

Pour l'estimé pondéré de Prec sur p

```
> modavg(cand.set = Cands.nb, modnames = Mod.nb.names,
         parm = "Prec", parm.type = "detect")
Multimodel inference on " p(Prec) " based on AICc

 AICc table used to obtain model-averaged estimate:

     K   AICc Delta_AICc AICcWt Estimate   SE
m1.nb 6 367.24       7.17   0.02    -0.07 0.03
m2.nb 7 360.07       0.00   0.65    -0.07 0.02
m3.nb 7 369.19       9.13   0.01    -0.07 0.03
m4.nb 8 361.47       1.41   0.32    -0.07 0.02

Model-averaged estimate: -0.07
Unconditional SE: 0.02
 95 % Unconditional confidence interval: -0.11 , -0.02
```

La probabilité de détection diminue avec la quantité de précipitations.

Inférence multimodèle – Ht

Pour l'estimé pondéré de Ht sur p

```
> modavg(cand.set = Cands.nb, modnames = Mod.nb.names,
         parm = "Ht", parm.type = "detect")
Multimodel inference on " p(Ht) " based on AICc

 AICc table used to obtain model-averaged estimate:

     K   AICc Delta_AICc AICcWt Estimate   SE
m1.nb 6 367.24       7.17   0.02    -0.37 0.23
m2.nb 7 360.07       0.00   0.65    -0.37 0.23
m3.nb 7 369.19       9.13   0.01    -0.38 0.23
m4.nb 8 361.47       1.41   0.32    -0.38 0.23

Model-averaged estimate: -0.38
Unconditional SE: 0.23
 95 % Unconditional confidence interval: -0.83 , 0.08
```

La probabilité de détection ne varie pas avec la hauteur des pièges.

Inférence multimodèle – Jday

Pour l'estimé pondéré de Jday sur *p*

```
> print(modavg(cand.set = Cands.nb, modnames = Mod.nb.names,
         parm = "Jday", parm.type = "detect"), digits = 4)
Multimodel inference on " p(Jday) " based on AICc

 AICc table used to obtain model-averaged estimate:

      K   AICc Delta_AICc AICcWt Estimate     SE
m1.nb 6 367.24     7.1704 0.0181  -0.0127 0.0060
m2.nb 7 360.07     0.0000 0.6525  -0.0141 0.0058
m3.nb 7 369.19     9.1281 0.0068  -0.0126 0.0060
m4.nb 8 361.47     1.4092 0.3226  -0.0139 0.0058

Model-averaged estimate: -0.014
Unconditional SE: 0.0058
 95 % Unconditional confidence interval: -0.0254 , -0.0027
```

La probabilité de détection diminue avec le jour julien.

Prédictions - Snag I

On crée un nouveau jeu de données

```
> range(pola.site$Snag)
[1] -3.4340  9.7185
> ##create newdata data frame
> ##centered effort
> new.dats <- data.frame(Conifer = 0,
                         Snag = seq(from = min(pola.site$Snag),
                             to = max(pola.site$Snag),
                             length.out = 30))
> ##uncentered Snag
> new.dats$Snag.orig <- new.dats$Snag + 3.4
```

Prédictions avec `modavgpred( )`

```
> preds <- modavgpred(cand.set = Cands.nb, modnames = Mod.nb.names,
                      c.hat = 1, newdata = new.dats,
                      type = "response", parm.type = "lambda")
```


Prédictions - Snag II

Ajouter les valeurs prédites et *SE* inconditionnelles

```
> new.dats$mod.avg.pred <- preds$mod.avg.pred
> new.dats$uncond.se <- preds$uncond.se
```

Pour calculer les intervalles de confiance, on utilise l'échelle logit

```
> preds.log <- modavgpred(cand.set = Cands.nb,
                          modnames = Mod.nb.names,
                          c.hat = 1, newdata = new.dats,
                          type = "link", parm.type = "lambda")
> ##add to data set
> new.dats$mod.avg.pred.log <- preds.log$mod.avg.pred
> new.dats$uncond.se.log <- preds.log$uncond.se
```

On calcule les limites et on convertit sur l'échelle originale

Prédictions - Snag III

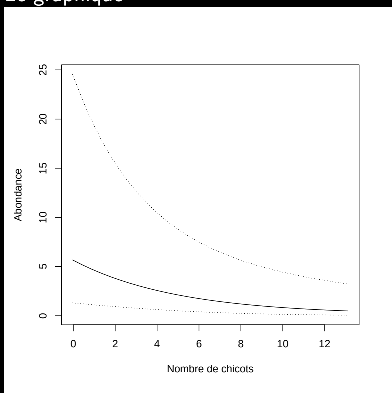
```
> ##95% CI
> new.dats$low95 <- exp(new.dats$mod.avg.pred.log -
                        1.96 * new.dats$uncond.se.log)
> new.dats$upp95 <- exp(new.dats$mod.avg.pred.log +
                        1.96 * new.dats$uncond.se.log)
```

Préparation du graphique

```
> ##create plot
> plot(new.dats$mod.avg.pred ~ new.dats$Snag.orig,
       ylab = "Abondance",
       xlab = "Nombre de chicots",
       ylim = range(c(new.dats$low95, new.dats$upp95)),
       type = "l")
> ##add confidence bands
> lines(y = new.dats$low95, x = new.dats$Snag.orig,
        lty = "dotted")
> lines(y = new.dats$upp95, x = new.dats$Snag.orig,
        lty = "dotted")
```

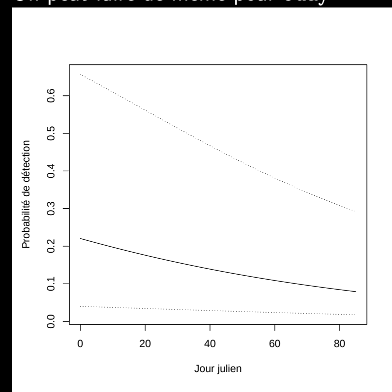
Prédictions - Snag IV

Le graphique



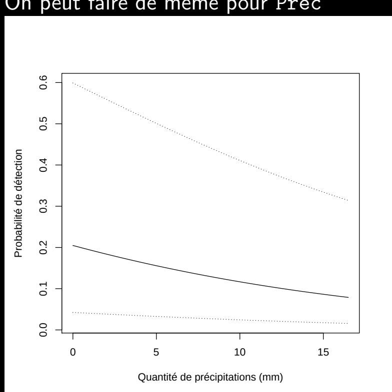
Prédictions - `Jday`

On peut faire de même pour `Jday`



Prédictions - `Prec`

On peut faire de même pour `Prec`



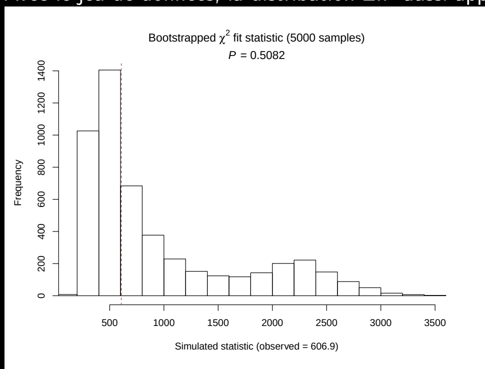
Modèles *N*-mixture avec *zero-inflated Poisson*

Essayons des modèles avec la même structure sur p et λ avec une distribution de Poisson gonflée de zéros (ZIP)

```
> m0.zip <- pcount(~ 1 ~ 1, K = 40, mixture = "ZIP",
                   data = pola.data)
> m1.zip <- pcount(~ Prec + Jday + Ht ~ 1, K = 40,
                   mixture = "ZIP", data = pola.data)
> m2.zip <- pcount(~ Prec + Jday + Ht ~ Snag, K = 40,
                   mixture = "ZIP", data = pola.data)
> m3.zip <- pcount(~ Prec + Jday + Ht ~ Conifer, K = 40,
                   mixture = "ZIP", data = pola.data)
> m4.zip <- pcount(~ Prec + Jday + Ht ~ Snag + Conifer, K = 40,
                   mixture = "ZIP", data = pola.data)
> m5.zip <- pcount(~ Snag + Conifer ~ 1, K = 40, mixture = "ZIP",
                   data = pola.data)
> m6.zip <- pcount(~ Snag + Conifer ~ Snag, K = 40,
                   mixture = "ZIP", data = pola.data)
> m7.zip <- pcount(~ Snag + Conifer ~ Conifer, K = 40,
                   mixture = "ZIP", data = pola.data)
> m8.zip <- pcount(~ Snag + Conifer ~ Snag + Conifer,
                   K = 40, mixture = "ZIP", data = pola.data)
```

Alternative — distribution ZIP

Avec le jeu de données, la distribution ZIP aussi appropriée



L'ajustement du modèle aussi est très satisfaisant ($P = 0.5082$) et le $\hat{c} = 0.65$.

Quelques modèles disponibles

Modèle *N*-mixture dynamique — `pcountOpen( )`

- estime l'abondance initiale (λ), le recrutement (γ), la survie apparente (ω), et la probabilité de détection

Modèle d'hétérogénéité — `occuRN( )`

- estime l'abondance à partir de données binaires

Modèle d'occupation avec faux-positifs et faux négatifs — `occuFP( )`

- estime l'occupation où il y a des faux-positifs

Modèles d'échantillonnage de la distance (*distance sampling*) — `distsamp( )` et `gdistsamp( )`

- estiment l'abondance à partir de données récoltées dans des intervalles discrets

Packages utiles

Occupation

- `hSDM`
- `pom`
- `stocc`
- `unmarked`

Approches bayésiennes

- `R2WinBUGS`
- `R2OpenBUGS`
- `R2jags`

Inférence multimodèle

- `AICcmodavg`
- `MuMIn`

CMR

- `CARE1`
- `marked`
- `mra`
- `Rcapture`
- `RMark`
- `secr`
- `admbsecr`
- `SPACECAP`

Échantillonnage de la distance

- `Distance`
- `dsm`
- `DSpat`
- `hierarchicalDS`
- `mrds`
- `mmds`
- `Rdistance`
- `unmarked`

Le package `RMark`

`RMark` permet de réaliser plusieurs modèles du programme `MARK`

- occupancy
  - une saison, multisaison
  - multi-états
  - co-occurrence
  - hétérogénéité
- *N*-mixture
  - une saison, multisaison
- CMR
  - population ouverte, fermée, design robuste
  - multi-états
  - recovery models
  - known fate
  - POPAN

D'autres avantages à utiliser R

R code, résultats et graphiques intégrés directement dans les documents (e.g., LaTeX et `Sweave` ou `knitr`, OpenOffice et `ODFweave`)

- un fichier comprend tout (texte, résultats, et code R)
- pas besoin de copier-coller
- les modifications du jeu de données se reflètent automatiquement dans les résultats et ls graphiques dans le texte

Utiliser R comme environnement de travail

- permet d'organiser et d'exécuter les analyses
- documente toutes les étapes du début à la fin
- R est cool, gratuit, et une grande communauté d'utilisateurs