Modèles de comptage avec R

Tuto@Mate, 21 mai 2024

Joseph Larmarange

Variable de type comptage

- *outcome* correspondant à un nombre entier positif
- souvent, nombre d'occurrences d'un évènement
- modèles linéaires et logistiques non adaptés

Modèle de Poisson

Un premier exemple

Descendance atteinte par des femmes à l'âge de 30 ans

- jeu de données `fecondite` fourni par le package `{questionr}`
- contient 3 tables : `menages`, `femmes` et `enfants`

Aperçu des données

```r
library(tidyverse)
library(labelled)
data("fecondite", package = "questionr")
enfants |> look_for()
```

```
 pos variable       label                        col_type missing values
 1   id_enfant      Identifiant de l'enfant      dbl      0
 2   id_femme       Identifiant de la mère       dbl      0
 3   date_naissance Date de naissance            date     0
 4   sexe           Sexe de l'enfant             dbl+lbl  0       [1] masculin
                                                                  [2] féminin
 5   survie         L'enfant est-il toujours en~ dbl+lbl  0       [0] non
                                                                  [1] oui
 6   age_deces      Age au décès (en mois)       dbl      1442
```

Aperçu des données

```
library(tidyverse)
library(labelled)
data("fecondite", package = "questionr")
enfants |> look_for()
```

```
 pos variable       label                        col_type missing values
 1   id_enfant      Identifiant de l'enfant      dbl      0
 2   id_femme       Identifiant de la mère       dbl      0
 3   date_naissance Date de naissance            date     0
 4   sexe           Sexe de l'enfant             dbl+lbl  0       [1] masculin
                                                                  [2] féminin
 5   survie         L'enfant est-il toujours en~ dbl+lbl  0       [0] non
                                                                  [1] oui
 6   age_deces      Age au décès (en mois)       dbl      1442
```

Les données sont labellisées –> conversion en facteurs avec `labelled::unlabelled()`

```
femmes <-
  femmes |>
  unlabelled()
enfants <-
  enfants |>
  unlabelled()
```

Préparation des données

Calcul de l'âge exact des mères à la naissance avec `lubridate::time_length()`

```
enfants <-
  enfants |>
  left_join(
    femmes |>
      select(id_femme, date_naissance_mere = date_naissance),
    by = "id_femme"
  ) |>
  mutate(
    age_mere = time_length(
      date_naissance_mere %--% date_naissance,
      unit = "years"
    )
  )
```

Préparation des données

Calcul de l'âge exact des mères à la naissance avec `lubridate::time_length()`

```
enfants <-
  enfants |>
  left_join(
    femmes |>
      select(id_femme, date_naissance_mere = date_naissance),
    by = "id_femme"
  ) |>
  mutate(
    age_mere = time_length(
      date_naissance_mere %--% date_naissance,
      unit = "years"
    )
  )
```

Comptons, par femme, le nombre d'enfants nés avant l'âge de 30 ans

```
femmes <-
  femmes |>
  left_join(
    enfants |>
      filter(age_mere < 30) |>
      group_by(id_femme) |>
      count(name = "enfants_avt_30"),
    by = "id_femme"
  ) |>
  tidyr::replace_na(list(enfants_avt_30 = 0L))
```

Préparation des données (2)

Calcul de l'âge des femmes au moment de l'enquête et recodage du niveau d'éducation

```
femmes <-
  femmes |>
  mutate(
    age = time_length(
      date_naissance %--% date_entretien,
      unit = "years"
    ),
    educ2 = educ |>
      fct_recode(
        "secondaire/supérieur" = "secondaire",
        "secondaire/supérieur" = "supérieur"
      )
  )
```

Préparation des données (2)

Calcul de l'âge des femmes au moment de l'enquête et recodage du niveau d'éducation

```
femmes <-
  femmes |>
  mutate(
    age = time_length(
      date_naissance %--% date_entretien,
      unit = "years"
    ),
    educ2 = educ |>
      fct_recode(
        "secondaire/supérieur" = "secondaire",
        "secondaire/supérieur" = "supérieur"
      )
  )
```

Enfin, nous n'allons garder que les femmes âgées d'au moins 30 ans au moment de l'enquête.

```
femmes30p <-
  femmes |>
  filter(age >= 30)
```

Statistiques descriptives

- Représentation des moyennes (avec intervalle de confiance à 95%), écart-type (avec `t.test()`) et nombre d'observations (astuce en utilisant `length()`) + test de comparaison (*one-way ANOVA*).

```
library(gtsummary)
theme_gtsummary_language("fr",
  decimal.mark = ",", big.mark = " "
)
mean_cil <- function(x, conf.level = 0.95) {
  t.test(x, conf.level = conf.level)$conf.int[1]
}
mean_cih <- function(x, conf.level = 0.95) {
  t.test(x, conf.level = conf.level)$conf.int[2]
}
femmes30p |>
  tbl_continuous(
    variable = enfants_avt_30,
    include = c(educ2, milieu, region),
    statistic = ~ "{mean} [{mean_cil} - {mean_cih}] ({sd}) [n={length}]",
    digits = ~ c(2, 2, 2, 2, 0)
  ) |>
  add_p(test = ~ "aov") |>
  bold_labels()
```

Caractéristique	**N = 804**	**p-valeur**
Niveau d'éducation		0,036

Calcul du modèle de Poisson

- fonction `stats::glm()` en précisant `family = poisson`
- réduction par minimisation de l'AIC avec `stats::step()`
- fonction de lien logarithmique (*log*) –> exponentielle des coefficients s'interprète comme un risque relatif

```
mod1_poisson <- glm(
  enfants_avt_30 ~ educ2 + milieu + region,
  family = poisson,
  data = femmes30p
)
mod1_poisson <- step(mod1_poisson)
```

```
Start:  AIC=1013.81
enfants_avt_30 ~ educ2 + milieu + region

         Df Deviance    AIC
- region  3   686.46 1010.6
<none>        683.62 1013.8
- milieu  1   686.84 1015.0
- educ2   2   691.10 1017.3

Step:  AIC=1010.65
enfants_avt_30 ~ educ2 + milieu

         Df Deviance    AIC
<none>        686.46 1010.6
- milieu  1   691.30 1013.5
- educ2   2   693.94 1014.1
```

Tableau des coefficients

```
mod1_poisson |>
  tbl_regression(exponentiate = TRUE) |>
  bold_labels()
```

Caractéristique	**IRR**	**95% IC**	**p-valeur**
Niveau d'éducation			
aucun	—	—	
primaire	1,25	0,90 – 1,72	0,2
secondaire/supérieur	0,53	0,27 – 0,96	0,052
Milieu de résidence			
urbain	—	—	
rural	1,42	1,04 – 1,98	0,032

Tableau des coefficients

```
mod1_poisson |>
  tbl_regression(exponentiate = TRUE) |>
  bold_labels()
```

Caractéristique	IRR	95% IC	p-valeur
Niveau d'éducation			
aucun	—	—	
primaire	1,25	0,90 – 1,72	0,2
secondaire/supérieur	0,53	0,27 – 0,96	0,052
Milieu de résidence			
urbain	—	—	
rural	1,42	1,04 – 1,98	0,032

```
library(ggstats)
mod1_poisson |>
  ggcoef_table(exponentiate = TRUE)
```

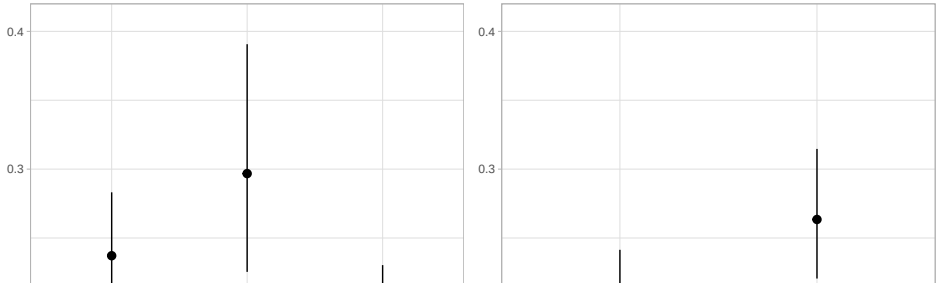
Interprétation des coefficients

- Le modèle de Poisson modélise le nombre moyen d'évènements.
- Le RR pour la modalité *secondaire/supérieur* est de 0,5 : indépendamment des autres variables du modèle, la descendance atteinte moyenne de ces femmes est moitié moindre que celle des femmes de la modalité de référence.

Interprétation des coefficients

- Le modèle de Poisson modélise le nombre moyen d'évènements.
- Le RR pour la modalité *secondaire/supérieur* est de 0,5 : indépendamment des autres variables du modèle, la descendance atteinte moyenne de ces femmes est moitié moindre que celle des femmes de la modalité de référence.
- Vérification visuelle avec un graphique des prédictions marginales moyennes.

```
mod1_poisson |>
  broom.helpers::plot_marginal_predictions() |>
  patchwork::wrap_plots() &
  ggplot2::scale_y_continuous(limits = c(0, .4))
```

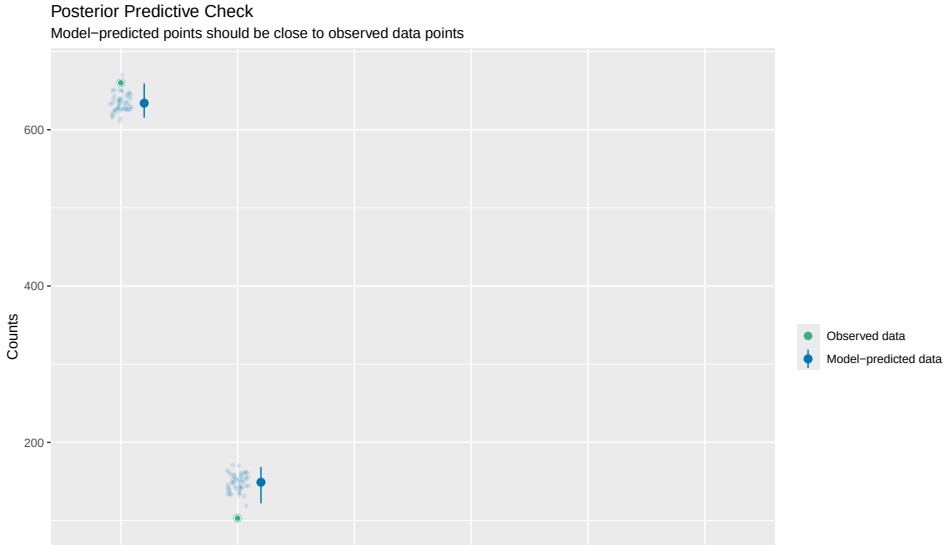


Évaluation de la surdispersion

- Le modèle de Poisson suppose que la variance est égale à la moyenne
- Or, la variance est souvent supérieure à la moyenne
- On parle alors de **surdispersion**
- Comparons distribution observée et distribution prédite/théorique du modèle

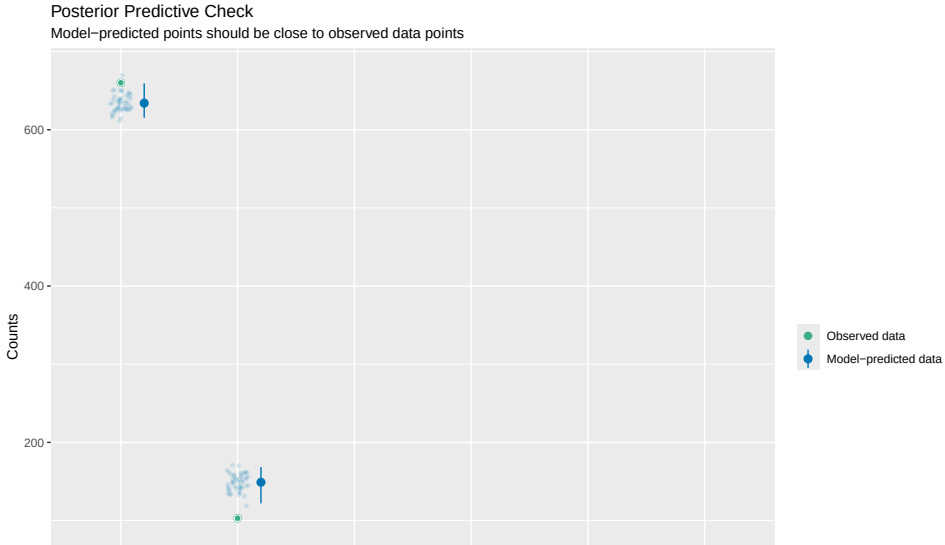
Visualiser / Tester la surdispersion

```
mod1_poisson |>
  performance::check_predictions(type = "discrete_both")
```



Visualiser / Tester la surdispersion

```
mod1_poisson |>
  performance::check_predictions(type = "discrete_both")
```



Modèle de quasi-Poisson

Modèle de quasi-Poisson

- Similaire au modèle de Poisson
- Fonction de lien logarithmique
- Plus de souplesse : variance modélisée comme une relation linéaire de la moyenne

Modèle de quasi-Poisson

- Similaire au modèle de Poisson
- Fonction de lien logarithmique
- Plus de souplesse : variance modélisée comme une relation linéaire de la moyenne
- S'obtient toujours avec `glm()` mais en indiquant `family = quasipoisson`

```
mod1_quasi <- glm(
  enfants_avt_30 ~ educ2 + milieu,
  family = quasipoisson,
  data = femmes30p
)
```

Modèle de quasi-Poisson

- Similaire au modèle de Poisson
- Fonction de lien logarithmique
- Plus de souplesse : variance modélisée comme une relation linéaire de la moyenne
- S'obtient toujours avec `glm()` mais en indiquant `family = quasipoisson`

```
mod1_quasi <- glm(
  enfants_avt_30 ~ educ2 + milieu,
  family = quasipoisson,
  data = femmes30p
)
```

- AIC non disponible => on ne peut pas utiliser `step()`
- Coefficients identiques au modèle de Poisson mais intervalles de confiance plus larges

Résultats du modèle de quasi-Poisson

```
mod1_quasi |>
  tbl_regression(exponentiate = TRUE) |>
  bold_labels()
```

Caractéristique	**IRR**	**95% IC**	**p-valeur**
Niveau d'éducation			
aucun	—	—	
primaire	1,25	0,85 – 1,81	0,2
secondaire/supérieur	0,53	0,23 – 1,05	0,10
Milieu de résidence			
urbain	—	—	
rural	1,42	0,99 – 2,10	0,067

Résultats du modèle de quasi-Poisson

```
mod1_quasi |>
  tbl_regression(exponentiate = TRUE) |>
  bold_labels()
```

Caractéristique	IRR	95% IC	p-valeur
Niveau d'éducation			
aucun	—	—	
primaire	1,25	0,85 – 1,81	0,2
secondaire/supérieur	0,53	0,23 – 1,05	0,10
Milieu de résidence			
urbain	—	—	
rural	1,42	0,99 – 2,10	0,067

```
list(
  Poisson = mod1_poisson,
  "quasi-Poisson" = mod1_quasi
) |>
  ggcoef_compare(exponentiate = TRUE)
```

Surdispersion du modèle

```
mod1_quasi |>
  performance::check_overdispersion()
```

```
# Overdispersion test

       dispersion ratio =    1.371
  Pearson's Chi-Squared = 1096.575
              p-value =  < 0.001
```

Le modèle de quasi-Poisson n'a pas suffi à régler le problème de surdispersion dans le cadre de notre exemple.

Modèle binomial négatif

Modèle binomial négatif

- Fonction de lien logarithmique
- Spécification quadratique de la variance (i.e. selon la moyenne et le carré de la moyenne)

Modèle binomial négatif

- Fonction de lien logarithmique
- Spécification quadratique de la variance (i.e. selon la moyenne et le carré de la moyenne)
- N'est pas disponible via `glm()`
- Recours à la fonction `MASS::glm.nb()` (syntaxe similaire)
- AIC défini => on peut utiliser `step()`

```
mod1_nb <- MASS::glm.nb(
  enfants_avt_30 ~ educ2 + milieu + region,
  data = femmes30p
)
mod1_nb <- mod1_nb |> step()
```

```
Start:  AIC=979.1
enfants_avt_30 ~ educ2 + milieu + region

         Df Deviance    AIC
- region  3   462.89 975.01
<none>        460.98 979.10
- milieu  1   463.29 979.41
- educ2   2   466.11 980.22

Step:  AIC=975
```

Résultats du modèle négatif binomial

```
mod1_nb |>
  tbl_regression(exponentiate = TRUE) |>
  bold_labels()
```

Caractéristique	IRR	95% IC	p-valeur
Niveau d'éducation			
aucun	—	—	
primaire	1,23	0,82 – 1,82	0,3
secondaire/supérieur	0,53	0,25 – 1,03	0,074
Milieu de résidence			
urbain	—	—	
rural	1,41	0,97 – 2,06	0,076

Résultats du modèle négatif binomial

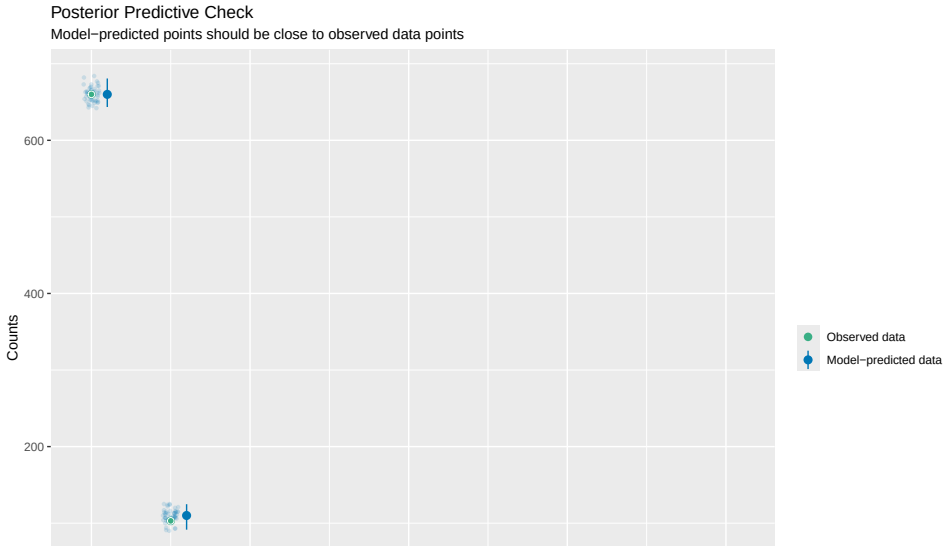
```
mod1_nb |>
  tbl_regression(exponentiate = TRUE) |>
  bold_labels()
```

Caractéristique	IRR	95% IC	p-valeur
Niveau d'éducation			
aucun	—	—	
primaire	1,23	0,82 – 1,82	0,3
secondaire/supérieur	0,53	0,25 – 1,03	0,074
Milieu de résidence			
urbain	—	—	
rural	1,41	0,97 – 2,06	0,076

```
list(
  Poisson = mod1_poisson,
  "quasi-Poisson" = mod1_quasi,
  "Binomial négatif" = mod1_nb
) |>
  ggcoef_compare(exponentiate = TRUE)
```

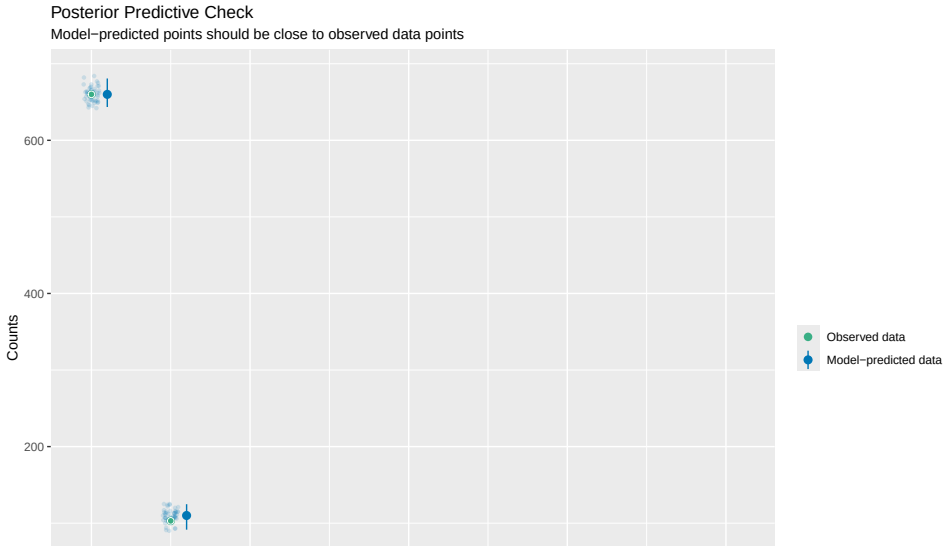
Vérification de la surdispersion

```
mod1_nb |>
  performance::check_predictions(type = "discrete_both")
```



Vérification de la surdispersion

```
mod1_nb |>
  performance::check_predictions(type = "discrete_both")
```



Comparaison de la performance des 2 modèles

```
performance::compare_performance(
  mod1_poisson,
  mod1_nb,
  metrics = "common"
)
```

```
# Comparison of Model Performance Indices

Name         |  Model |  AIC (weights) |  BIC (weights) | Nagelkerke's R2 |  RMSE
-------------------------------------------------------------------------------
mod1_poisson |    glm | 1010.7 (<.001) | 1029.4 (<.001) |           0.033 | 0.579
mod1_nb      | negbin |  977.0 (>.999) | 1000.5 (>.999) |           0.032 | 0.579
```

Modèle de comptage avec une variable binaire

Une alternative à la régression logistique

- une variable binaire peut-être modélisée avec un modèle de comptage, en considérant que la personne a vécu 0 fois ou 1 fois l'évènement d'intérêt
- permet de calculer des *prevalence ratios* (ou risques relatifs) plutôt que des *odds ratio*

Une alternative à la régression logistique

- une variable binaire peut-être modélisée avec un modèle de comptage, en considérant que la personne a vécu 0 fois ou 1 fois l'évènement d'intérêt
- permet de calculer des *prevalence ratios* (ou risques relatifs) plutôt que des *odds ratio*
- exemple : probabilité de faire du sport (enquête *histoire de vie 2003*)

```
data(hdv2003, package = "questionr")
d <- hdv2003 |>
  mutate(
    groupe_ages = age |>
      cut(
        c(18, 25, 45, 65, 99),
        right = FALSE, include.lowest = TRUE,
        labels = c("18-24 ans", "25-44 ans", "45-64 ans", "65 ans et plus")
      )
  ) |>
  set_variable_labels(
    sport = "Pratique un sport ?",
    sexe = "Sexe",
    groupe_ages = "Groupe d'âges",
    heures.tv = "Heures de télévision / jour"
  )
contrasts(d$sexe) <- contr.treatment(2, base = 2)
```

Calcul des modèles

Régression logistique

```
mod2_binomial <- glm(
  sport ~ sexe + groupe_ages + heures.tv,
  family = binomial,
  data = d
)
```

Calcul des modèles

Régression logistique

```
mod2_binomial <- glm(
  sport ~ sexe + groupe_ages + heures.tv,
  family = binomial,
  data = d
)
```

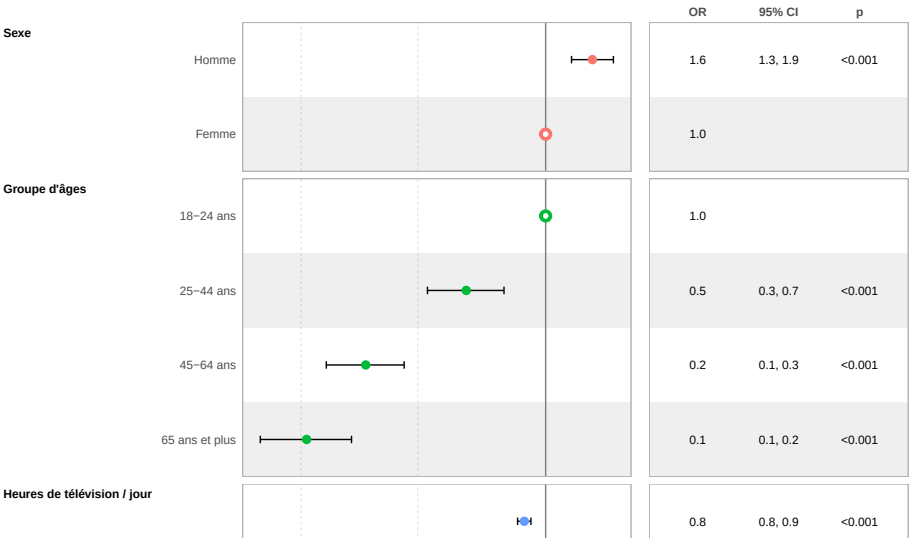
Modèle de Poisson

- Attention : il faut transformer la variable d'intérêt en un entier du type 0/1

```
d$sport2 <- as.integer(d$sport == "Oui")
mod2_poisson <- glm(
  sport2 ~ sexe + groupe_ages + heures.tv,
  family = poisson,
  data = d
)
```

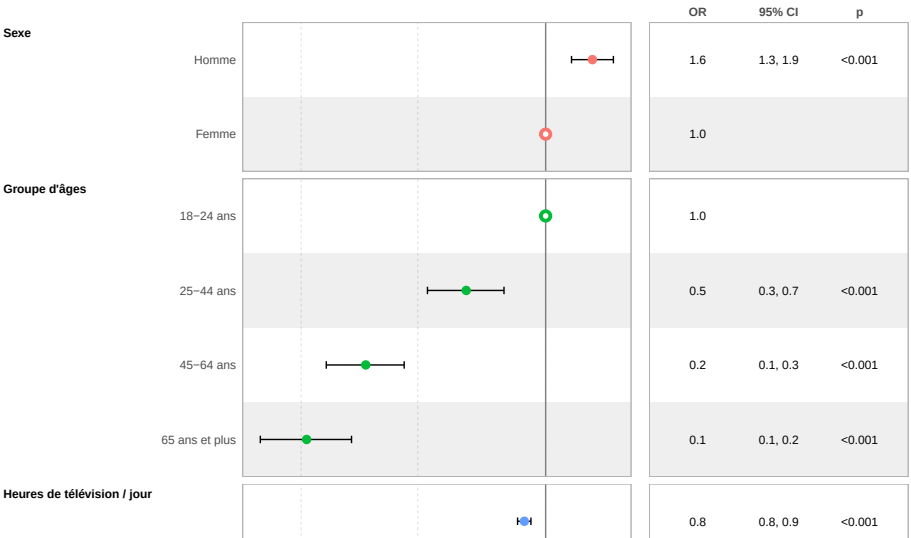
Coefficients des deux modèles

```
mod2_binomial |>
  ggstats::ggcoef_table(exponentiate = TRUE)
```



Coefficients des deux modèles

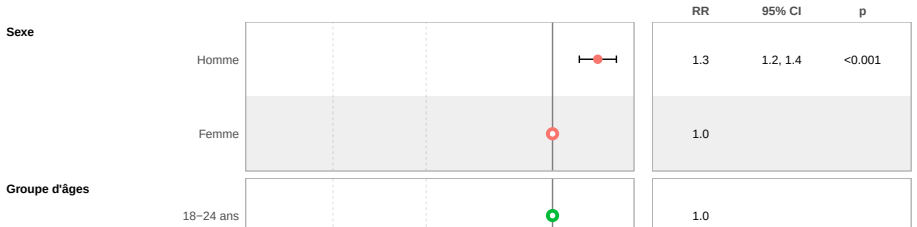
```
mod2_binomial |>
  ggstats::ggcoef_table(exponentiate = TRUE)
```



Alternative : le modèle log-binomial

- nativement avec `glm(family = binomial(link = "log"))` mais peut avoir des difficultés à converger
- nécessité d'initialiser les coefficients avec ceux d'un modèle de Poisson
- lien logarithmique => coefficients interprétables comme des risques relatifs

```r
mod2_log <- glm(
  sport ~ sexe + groupe_ages + heures.tv,
  family = binomial(link = "log"),
  start = mod2_poisson$coefficients,
  data = d
)
mod2_log |>
  ggstats::ggcoef_table(exponentiate = TRUE)
```



Comparaison des performances

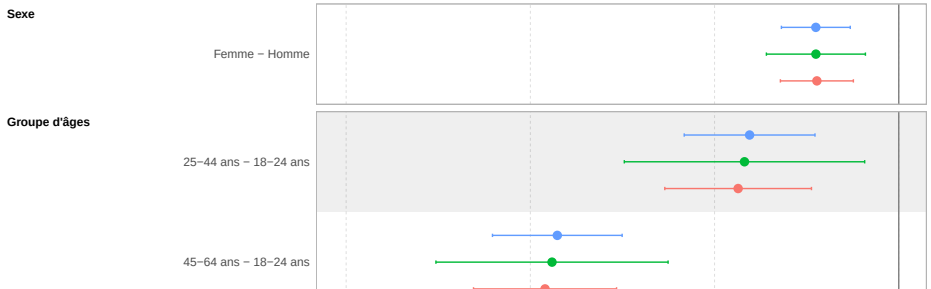
```
performance::compare_performance(
  mod2_binomial,
  mod2_poisson,
  mod2_log,
  metrics = "common"
)
```

```
# Comparison of Model Performance Indices

Name          | Model |  AIC (weights) |  BIC (weights) |  RMSE | Nagelkerke's R2 | Tjur's R2
-----------------------------------------------------------------------------------------
mod2_binomial |   glm | 2346.3 (0.642) | 2379.8 (0.642) | 0.447 |                 |     0.132
mod2_poisson  |   glm | 2748.8 (<.001) | 2782.4 (<.001) | 0.447 |           0.158 |
mod2_log      |   glm | 2347.4 (0.358) | 2381.0 (0.358) | 0.447 |           0.176 |
```

Contrastes marginaux marginaux

- ré-exprime les différences selon l'échelle de la variable d'intérêt, soit ici des différences de probabilité (en points de pourcentage)

```
list(
  "logistique" = mod2_binomial,
  "Poisson" = mod2_poisson,
  "log-binomiale" = mod2_log
) |>
  ggcoef_compare(
    tidy_fun = broom.helpers::tidy_marginal_contrasts
  ) +
  scale_x_continuous(
    labels = scales::label_percent(suffix = "pp")
  )
```



Modèles d'incidence / de taux

Modéliser une incidence / un taux

- En épidémiologie, le taux d'**incidence** rapporte le nombre de nouveaux cas pendant une période donnée à la population exposée pendant cette même période.
- En démographie, le terme de **taux** est utilisé pour désigner la fréquence relative d'un évènement au sein d'une population pendant une période de temps donnée (par exemple : taux de natalité).

Modéliser une incidence / un taux

- En épidémiologie, le taux d'**incidence** rapporte le nombre de nouveaux cas pendant une période donnée à la population exposée pendant cette même période.
- En démographie, le terme de **taux** est utilisé pour désigner la fréquence relative d'un évènement au sein d'une population pendant une période de temps donnée (par exemple : taux de natalité).

- Si l'ensemble des individus sont observés pendant une seule unité de temps, alors cela revient à rapporter le nombre moyen d'évènements à 1 : nous pouvons utiliser un modèle classique de de comptage.
- Le plus souvent, la durée d'observation / d'exposition varie d'un individu à l'autre.
- Pour chaque individu, il nous faut donc connaître le nombre d'évènements vécus (n_{evts}) et la durée d'exposition (d_{exp}). Ce que l'on cherche à modéliser est donc le ratio n_{evts}/d_{exp}.

Modéliser une incidence / un taux

- En épidémiologie, le taux d'**incidence** rapporte le nombre de nouveaux cas pendant une période donnée à la population exposée pendant cette même période.
- En démographie, le terme de **taux** est utilisé pour désigner la fréquence relative d'un évènement au sein d'une population pendant une période de temps donnée (par exemple : taux de natalité).

- Si l'ensemble des individus sont observés pendant une seule unité de temps, alors cela revient à rapporter le nombre moyen d'évènements à 1 : nous pouvons utiliser un modèle classique de de comptage.
- Le plus souvent, la durée d'observation / d'exposition varie d'un individu à l'autre.
- Pour chaque individu, il nous faut donc connaître le nombre d'évènements vécus (n_{evts}) et la durée d'exposition (d_{exp}). Ce que l'on cherche à modéliser est donc le ratio n_{evts}/d_{exp}.

- Une astuce consiste à utiliser un modèle ayant une fonction de lien logarithmique (*log*).
- On cherchera donc à modéliser notre variable sous la forme $log(n_{evts}/d_{exp}) = \beta_i X_i$.

Modéliser une incidence / un taux

- En épidémiologie, le taux d'**incidence** rapporte le nombre de nouveaux cas pendant une période donnée à la population exposée pendant cette même période.
- En démographie, le terme de **taux** est utilisé pour désigner la fréquence relative d'un évènement au sein d'une population pendant une période de temps donnée (par exemple : taux de natalité).

- Si l'ensemble des individus sont observés pendant une seule unité de temps, alors cela revient à rapporter le nombre moyen d'évènements à 1 : nous pouvons utiliser un modèle classique de de comptage.
- Le plus souvent, la durée d'observation / d'exposition varie d'un individu à l'autre.
- Pour chaque individu, il nous faut donc connaître le nombre d'évènements vécus (n_{evts}) et la durée d'exposition (d_{exp}). Ce que l'on cherche à modéliser est donc le ratio n_{evts}/d_{exp}.

- Une astuce consiste à utiliser un modèle ayant une fonction de lien logarithmique (*log*).
- On cherchera donc à modéliser notre variable sous la forme $log(n_{evts}/d_{exp}) = \beta_i X_i$.

Premier exemple (données individuelles)

Prenons un premier exemple à partir du jeux de données `gtsummary::trial` qui contient des informations sur 200 patients atteints d'un cancer. Il contient entre autre les variables suivantes :

- *death* : variable binaire (0/1) indiquant si le patient est décédé
- *ttdeath* : le nombre de mois d'observation jusqu'au décès (si décès) ou jusqu'à la fin de l'étude (si survie)
- *stage* : un facteur indiquant le stade T de la tumeur (plus la valeur est élevée, plus la tumeur est grosse)
- *trt* : le traitement reçu par le patient (A ou B)
- *response* : une variable binaire (0/1) indiquant si le traitement a eu un effet sur la tumeur (diminution)

Nous nous intéressons donc aux facteurs associés au taux de mortalité (*death/ttdeath*) : nous allons donc réaliser un modèle de Poisson sur la variable *death* en ajoutant un décalage (*offset*) correspondant à *log(ttdeath)*.

Statistiques descriptives

- exprimée en personne-mois (pm) ou *person-months* en anglais

```
percent2 <- purrr::partial(style_percent, digits = 1)
trial <- gtsummary::trial |>
  set_value_labels(
    response = c(no = 0, yes = 1)
  ) |>
  unlabelled()
trial |>
  tbl_custom_summary(
    include = c(stage, trt, response),
    stat_fns = ~ ratio_summary("death", "ttdeath"),
    statistic = ~"{ratio}/100pm [{conf.low}; {conf.high}] ({num}/{denom})",
    digits = ~ c(percent2, percent2, percent2, 0, 0),
    overall_row = TRUE,
    overall_row_label = "Overall"
  ) |>
  bold_labels()
```

Caractéristique	**N = 200**
Overall	2,85/100pm [2,35; 3,43] (112/3 925)
T Stage	
T1	2,17/100pm [1,39; 3,23] (24/1 105)
T2	2,48/100pm [1,63; 3,61] (27/1 089)
T3	2,58/100pm [1,62; 3,91] (22/852)

Calcul du modèle

Pour ajouter un décalage, nous avons deux syntaxes équivalentes : soit en ajoutant `offset(log(ttdeath))` directement à l'équation du modèle, soit en passant à `glm()` l'argument `offset = log(ttdeath)`.

```
mod3_poisson <- glm(
  death ~ stage + trt + response + offset(log(ttdeath)),
  family = poisson,
  data = trial
)

mod3_poisson_alt <- glm(
  death ~ stage + trt + response,
  offset = log(ttdeath),
  family = poisson,
  data = trial
)
```

Calcul du modèle

Pour ajouter un décalage, nous avons deux syntaxes équivalentes : soit en ajoutant `offset(log(ttdeath))` directement à l'équation du modèle, soit en passant à `glm()` l'argument `offset = log(ttdeath)`.

```
mod3_poisson <- glm(
  death ~ stage + trt + response + offset(log(ttdeath)),
  family = poisson,
  data = trial
)

mod3_poisson_alt <- glm(
  death ~ stage + trt + response,
  offset = log(ttdeath),
  family = poisson,
  data = trial
)
```

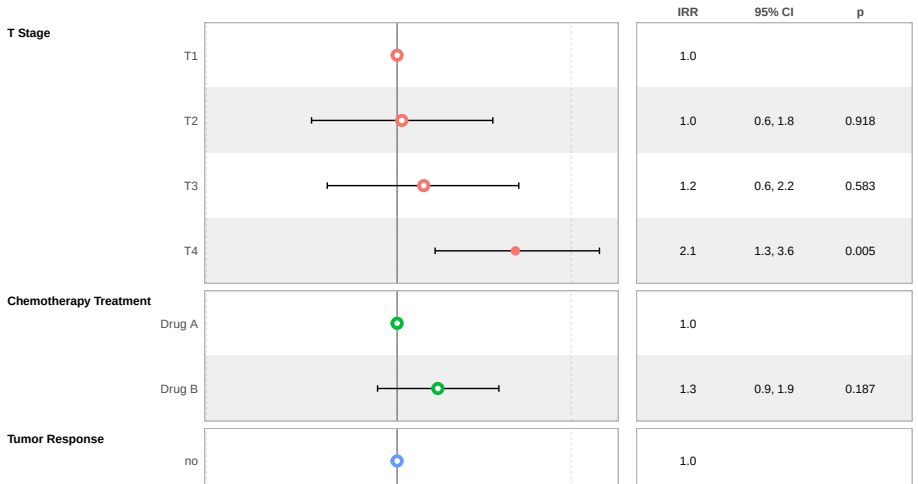
```
mod3_poisson |>
  performance::check_overdispersion()
```

```
# Overdispersion test

       dispersion ratio =   0.850
  Pearson's Chi-Squared = 159.039
              p-value =   0.932
```

Résultats du modèle

L'exponentielle des coefficients s'interprète comme un *incidence risk ratio (IRR)*.

```
mod3_poisson |>
  ggstats::ggcoef_table(exponentiate = TRUE)
```



Modèles univariables

- Calcul de plusieurs modèles univariables pour vérifier les associations avant ajustement

```
trial |>
  tbl_uvregression(
    y = death,
    include = c(stage, trt, response),
    method = glm,
    method.args = list(
      family = poisson,
      offset = log(ttdeath)
    ),
    exponentiate = TRUE
  ) |>
  add_global_p() |>
  bold_labels()
```

Caractéristique	**N**	**IRR**	**95% IC**	**p-valeur**
Chemotherapy Treatment	200			0,4
Drug A		—	—	
Drug B		1,18	0,81 – 1,71	
T Stage	200			0,025
T1		—	—	
T2		1,14	0,66 – 1,99	

Deuxième exemple (données agrégées)

Nous allons considérer le jeu de données `MASS::Insurance` qui provient d'une compagnie d'assurance américaine et porte sur le troisième trimestre 1973.

Il indique le nombre de demande d'indemnisations (*Claims*) parmi les assurés pour leur voiture (*Holders*) en fonction de leur groupe d'âges (*Age*) et de la taille de la cylindrée de la voiture (*Group*).

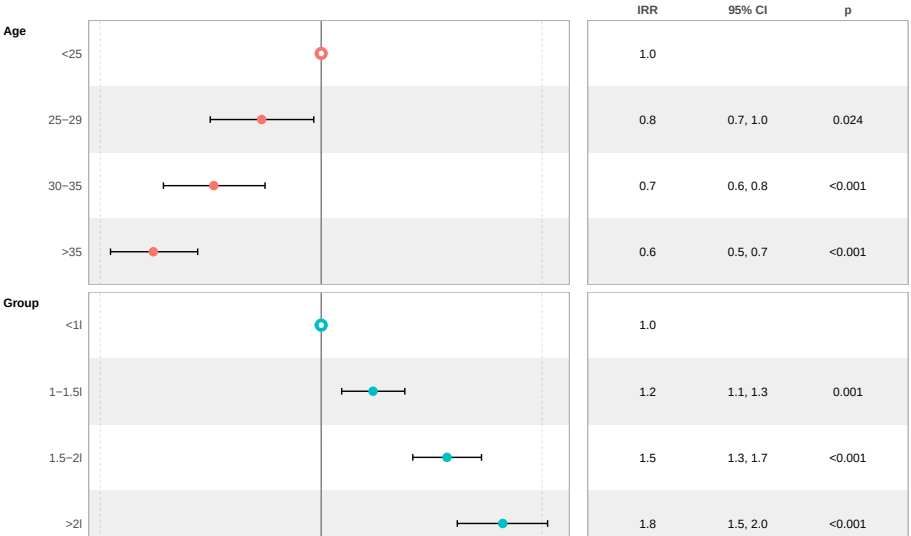
Nous cherchons à identifier les facteurs associés au taux de réclamation.

```
d <- MASS::Insurance
d$Age <- factor(d$Age, ordered = FALSE)
d$Group <- factor(d$Group, ordered = FALSE)

mod4_poisson <- glm(
  Claims ~ Age + Group + offset(log(Holders)),
  family = poisson,
  data = d
)
mod4_poisson |>
  performance::check_overdispersion()
```

```
# Overdispersion test

       dispersion ratio =  1.140
  Pearson's Chi-Squared = 65.003
              p-value =  0.218
```

Résultats

```
mod4_poisson |>
  ggstats::ggcoef_table(exponentiate = TRUE)
```



Modèles zero-inflated et hurdle

Données d'illustration

Nous allons utiliser un jeu de données issu d'un article de Partha Deb et Pravin K. Trivedi. Ce jeu de données porte sur 4406 individus âgés de 66 ans ou plus et couvert par le programme américain *Medicare*.

Données d'illustration

Nous allons utiliser un jeu de données issu d'un article de Partha Deb et Pravin K. Trivedi. Ce jeu de données porte sur 4406 individus âgés de 66 ans ou plus et couvert par le programme américain *Medicare*.

L'analyse va porter sur la demande de soins, mesurée ici à travers le nombre de visites médicales (*ofp*).

Pour les variables explicatives, nous allons considérer le genre du patient (*gender*), le fait de disposer d'une assurance privée (*privins*), la santé perçue (*health*) et le nombre de conditions chroniques de l'assuré.

Données d'illustration

Nous allons utiliser un jeu de données issu d'un article de Partha Deb et Pravin K. Trivedi. Ce jeu de données porte sur 4406 individus âgés de 66 ans ou plus et couvert par le programme américain *Medicare*.

L'analyse va porter sur la demande de soins, mesurée ici à travers le nombre de visites médicales (*ofp*).

Pour les variables explicatives, nous allons considérer le genre du patient (*gender*), le fait de disposer d'une assurance privée (*privins*), la santé perçue (*health*) et le nombre de conditions chroniques de l'assuré.

```
load(url("https://github.com/larmarange/guide-R/raw/main/analyses_avancees/ressources/DebTrivedi.rda"))
d <- DebTrivedi |>
  mutate(
    gender = gender |> fct_recode("femme" = "female", "homme" = "male"),
    privins = privins |> fct_recode("non" = "no", "oui" = "yes"),
    health = health |> fct_recode("pauvre" = "poor", "moyenne" = "average", "excellente" = "excellent")
  ) |>
  set_variable_labels(
    ofp = "Nombre de visites médicales",
    gender = "Genre de l'assuré",
    privins = "Dispose d'une assurance privée ?",
    health = "Santé perçue",
    numchron = "Nombre de conditions chroniques"
  )
contrasts(d$health) <- contr.treatment(3, base = 2)
```

Modèle binomial négatif

```r
mod5_nb <- MASS::glm.nb(
  ofp ~ gender + privins + health + numchron,
  data = d
)
mod5_nb |>
  performance::check_overdispersion()
```

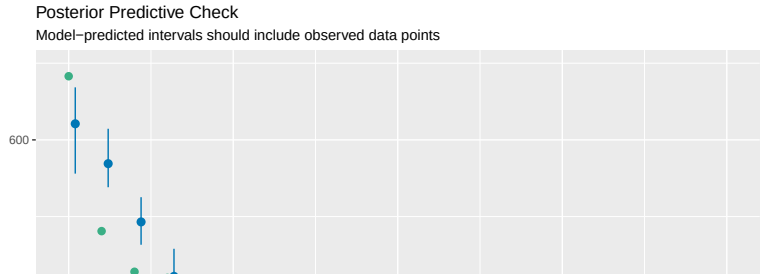
```
# Overdispersion test

 dispersion ratio = 1.050
          p-value =  0.36
```

Modèle binomial négatif

```
mod5_nb <- MASS::glm.nb(
  ofp ~ gender + privins + health + numchron,
  data = d
)
mod5_nb |>
  performance::check_overdispersion()
```

```
# Overdispersion test

 dispersion ratio = 1.050
          p-value =  0.36
```

```
mod5_nb |>
  performance::check_predictions() |>
  plot() +
  xlim(0, 20)
```



Modèle zero-inflated

- Le nombre de 0 prédit par le modèle est inférieur à celui observé.
- Les 0 sont sur-représentés dans nos données par rapport à une distribution négative binomiale.

Modèle zero-inflated

- Le nombre de 0 prédit par le modèle est inférieur à celui observé.
- Les 0 sont sur-représentés dans nos données par rapport à une distribution négative binomiale.

- On pourra alors considérer un modèle *zero-inflated*.
- Un modèle de Poisson *zero-inflated* combine deux modèles : un modèle logistique binaire et un modèle de Poisson.
- Dans un premier temps, on applique le modèle logistique binaire. Si la valeur obtenue est 0, le résultat final est 0. Si la valeur obtenue est 1, alors on applique le modèle de Poisson.

Modèle zero-inflated

- Le nombre de 0 prédit par le modèle est inférieur à celui observé.
- Les 0 sont sur-représentés dans nos données par rapport à une distribution négative binomiale.

- On pourra alors considérer un modèle *zero-inflated*.
- Un modèle de Poisson *zero-inflated* combine deux modèles : un modèle logistique binaire et un modèle de Poisson.
- Dans un premier temps, on applique le modèle logistique binaire. Si la valeur obtenue est 0, le résultat final est 0. Si la valeur obtenue est 1, alors on applique le modèle de Poisson.

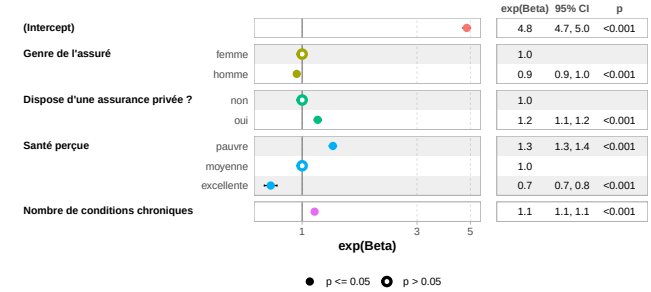
- Un tel modèle se calcule avec `pscl::zeroinfl()`.

```
mod5_zip <- pscl::zeroinfl(
  ofp ~ gender + privins + health + numchron,
  data = d
)
```

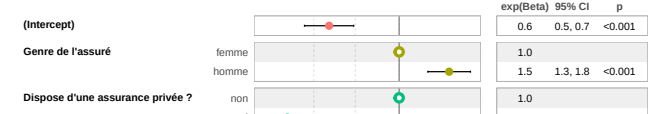
Résultats

```
mod5_zip |>
  ggstats::ggcoef_multicomponents(
    type = "table",
    exponentiate = TRUE,
    intercept = TRUE
  )
```

conditional



zero_inflated



Variantes

- On peut choisir des variables différentes pour chaque sous-modèle (avec `|`), voire faire un modèle simple avec seulement un *intercept* pour la composante logistique.
- On peut également utiliser un modèle négatif binomial avec `dist = "negbin"`.

Variantes

- On peut choisir des variables différentes pour chaque sous-modèle (avec |), voire faire un modèle simple avec seulement un *intercept* pour la composante logistique.
- On peut également utiliser un modèle négatif binomial avec `dist = "negbin"`.

```
mod5_zip_simple <- pscl::zeroinfl(
  ofp ~ gender + privins + health + numchron | 1,
  data = d
)
mod5_zinb <- pscl::zeroinfl(
  ofp ~ gender + privins + health + numchron,
  dist = "negbin",
  data = d
)
performance::compare_performance(
  mod5_nb,
  mod5_zip_simple,
  mod5_zip,
  mod5_zinb,
  metrics = "AIC"
)
```

```
# Comparison of Model Performance Indices

Name            |    Model |   AIC (weights)
-----------------------------------------------
mod5_nb         |   negbin | 24505.7 (<.001)
mod5_zip_simple | zeroinfl | 33308.5 (<.001)
mod5_zip        | zeroinfl | 33014.2 (<.001)
mod5_zinb       | zeroinfl | 24380.9 (>.999)
```

Tableau des coefficients

```
tbl_log <- mod5_zinb |>
  tbl_regression(
    tidy_fun = broom.helpers::tidy_zeroinfl,
    component = "zero_inflated",
    exponentiate = TRUE
  )
tbl_nb <- mod5_zinb |>
  tbl_regression(
    tidy_fun = broom.helpers::tidy_zeroinfl,
    component = "conditional",
    exponentiate = TRUE
  )
list(tbl_log, tbl_nb) |>
  tbl_merge(
    c(
      "**OR (régression logistique)**",
      "**RR (négatif binomial)**"
    )
  ) |>
  bold_labels()
```

Caractéristique	**exp(Beta)**	**95% IC**	**p-valeur**	**exp(Beta)**	**95% IC**	**p-valeur**
Genre de l'assuré						
femme	—	—		—	—	
homme	1,82	1,21 –	0,004	0,93	0,88 –	0,031

Interprétation

- Si l'interprétation du modèle de comptage reste classique, celle du modèle logistique binaire est parfois un peu plus complexe.

Interprétation

- Si l'interprétation du modèle de comptage reste classique, celle du modèle logistique binaire est parfois un peu plus complexe.
- En effet, il y a deux sources de 0 dans le modèle *zero-inflated* : si certains sont générés par la composante logistique binaire, le modèle de comptage génère lui aussi des 0.
- Dès lors, le modèle logistique binaire ne suffit pas à lui seul à identifier les facteurs associés de vivre au moins une fois l'évènement.

Interprétation

- Si l'interprétation du modèle de comptage reste classique, celle du modèle logistique binaire est parfois un peu plus complexe.
- En effet, il y a deux sources de 0 dans le modèle *zero-inflated* : si certains sont générés par la composante logistique binaire, le modèle de comptage génère lui aussi des 0.
- Dès lors, le modèle logistique binaire ne suffit pas à lui seul à identifier les facteurs associés de vivre au moins une fois l'évènement.
- Si l'objectif de l'analyse est avant tout d'identifier les facteurs associés avec le nombre moyen d'évènements, on pourra éventuellement se contenter d'un modèle *zero-inflated* simple, c'est-à-dire avec seulement un *intercept* pour la composante *zero-inflated* afin de corriger la sur-représentation des zéros dans nos données.

Interprétation

- Si l'interprétation du modèle de comptage reste classique, celle du modèle logistique binaire est parfois un peu plus complexe.
- En effet, il y a deux sources de 0 dans le modèle *zero-inflated* : si certains sont générés par la composante logistique binaire, le modèle de comptage génère lui aussi des 0.
- Dès lors, le modèle logistique binaire ne suffit pas à lui seul à identifier les facteurs associés de vivre au moins une fois l'évènement.
- Si l'objectif de l'analyse est avant tout d'identifier les facteurs associés avec le nombre moyen d'évènements, on pourra éventuellement se contenter d'un modèle *zero-inflated* simple, c'est-à-dire avec seulement un *intercept* pour la composante *zero-inflated* afin de corriger la sur-représentation des zéros dans nos données.
- Alternativement, on pourra se tourner vers un modèle avec saut qui distingue les valeurs nulles des valeurs positives : les modèles *hurdle* en anglais.
- Les modèles *hurdle* se distinguent des modèles *zero-inflated* dans le sens où l'on combine un modèle logistique binomial pour déterminer si les individus ont vécu au moins une fois l'évènement et un modèle de comptage tronqué (qui n'accepte

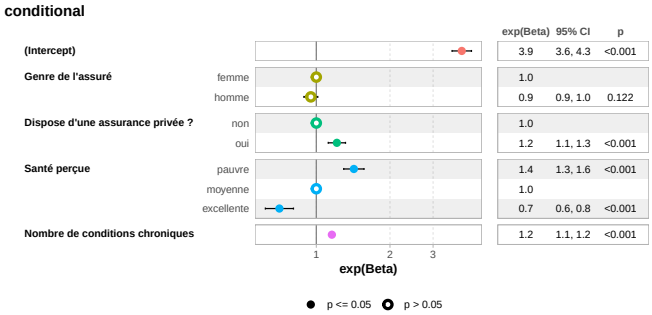
Calcul d'un modèle hurdle

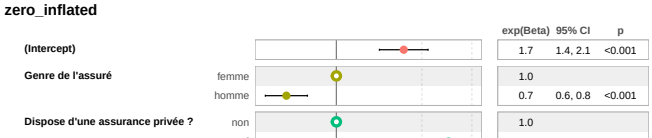
On utilisera simplement `pscl::hurdle()` à la place de `pscl::zeroinfl()`.

```
mod5_hurdle_poisson <- pscl::hurdle(
  ofp ~ gender + privins + health + numchron,
  data = d
)
mod5_hurdle_nb <- pscl::hurdle(
  ofp ~ gender + privins + health + numchron,
  dist = "negbin",
  data = d
)
```

Résultats du modèle hurdle

```
mod5_hurdle_nb |>
  ggstats::ggcoef_multicomponents(
    type = "table",
    exponentiate = TRUE,
    intercept = TRUE
  )
```





Ressources

guide-R

Sur les modèles de comptage :

- Modèles de comptage : https://larmarange.github.io/guide-R/analyses_avancees/modeles-comptage.html
- Modèles d'incidence : https://larmarange.github.io/guide-R/analyses_avancees/modeles-incidence.html
- Modèles zero-inflated et hurdle : https://larmarange.github.io/guide-R/analyses_avancees/modeles-zero-inflated.html

Autres ressources utiles :

- Étiquettes de valeurs : https://larmarange.github.io/guide-R/manipulation/etiquettes-valeurs.html
- Sélection pas à pas d'un modèle : https://larmarange.github.io/guide-R/analyses/selection-modele-pas-a-pas.html
- Prédictions marginales, contrastes marginaux & effets marginaux : https://larmarange.github.io/guide-R/analyses/estimations-marginales.html