Poisson-Regression und ihre Leiden

Lukas Burk

Stand: 06. March 2020 23:34 Uhr (CET)

Inhaltsverzeichnis

1. Einführung

Dieses Dokument soll einen Überblick zum Umgang mit Zähldaten (*count data*) liefern.
Zähldaten können im Allgemeinen mittels Poissonverteilung im Rahmen des GLM modelliert werden, allerdings sind in der Praxis einige Komplikationen zu erwarten:

Over-/underdispersion (siehe Abschnitt 2.2): Die Poissonverteilung besitzt nur einen Parameter für sowohl Erwartungswert als auch Varianz und nimmt somit Gleichheit zwischen den beiden an (*equidispersion*) – diese Annahme ist meist in Form von overdispersion verletzt

Zero-Inflation (siehe Abschnitt 4.2): Aus einem Modell lässt sich die erwartete Anzahl an Beobachtungen mit Anzahl 0 bestimmen – wenn die beobachtete Anzahl (bzw. der Anteil) an Nullen deutlich größer ist, spricht man von *zero-inflation*. Verwandte Probleme sind die (seltenere) *zero-deflation*, oder das strukturelle Fehlen von Nullen

Diese Umstände benötigen in der Regel Generalisierungen der einfach Poisson-Regression, entweder durch Erweiterung der Verteilung um zusätzliche Parameter (siehe z.B. *Negative Binomialverteilung* für overdispersion in Abschnitt 3.1, *Generalized Poisson* für underdispersion in Abschnitt 3.3) oder die Konstruktion von *mixture models* oder *hurdle models* für zero-inflation (Abschnitte 3.6 und B.3).

1.1. Beispieldaten

Als Anwendungsbeispiele werden einige Datensätze verwendet, die verschiedene Probleme illustrieren:

- `azprocedure`: Patienten kardiovaskulärer Behandlungen in Arizona
  - Count: `los`, Dauer eines Krankenhausaufenthalts in Tagen (1 - 83)
  - Kovariablen:
    - `sex`: Male (1), female(0)
    - `admit`: Type of admission. Urgent/emergency (1), elective (0)
- `rwm5yr` (`rwm1984`): „German health registry“ 1984-1988 (bzw. nur 1984)
  - Count: `docvis`: Anzahl der Arztbesuche (0 - 121)
  - Kovariablen:

    * `outwork`: Arbeitslos (1), arbeitend (0)
    * `age`: Alter (25 - 64)

- `fish`: Geangelte Fische an einem Campingwochenende
  - Count: `count`, Anzahl der Fische
  - Kovariablen:
    * `child`: Anzahl der Kinder in der Campergruppe
    * `persons`: Anzahl der Personen in der Campergruppe
    * `camper`: `[0, 1]` Hat die Gruppe einen Campingwagen mitgebracht?

Diese Datensätze finden sich entweder in R-packages oder auf der website der UCLA IDRE:

Dataset	R Package	Quelle
azprocedure	COUNT	Hilbe (2014)
rwm5yr	COUNT	Hilbe (2014)
rwm1984	COUNT	Hilbe (2014)
nuts	COUNT	Hilbe (2014)
fish	–	UCLA IDRE (https://stats.idre.ucla.edu)

Daten aus Hilbe (2014) sind zusätzlich verfügbar als CSV (`HILBE-MCD-CVS-data`) auf der Website des Autors[1].

Die Datensätze des UCLA IDRE können wie folgt eingelesen werden:

```
fish <- haven::read_sas("https://stats.idre.ucla.edu/stat/sas/code/fish.sas7bdat")

fish <- within(fish, {
    nofish <- factor(nofish)
    livebait <- factor(livebait)
    camper <- factor(camper)
})

# Cache locally
saveRDS(fish, "data/fish.rds")

# Read from cache later:
# fish <- readRDS("data/fish.rds")
```

Um Daten aus `R` direkt in SAS-freundlichem `sas7bdat` zu speichern, kann folgender Code unter Verwendung des packages `haven`[2] verwendet werden:

[1] https://works.bepress.com/joseph_hilbe/58/
[2] https://haven.tidyverse.org/

```r
# Install package 'haven' if required
if (!("haven" %in% installed.packages())) {
   install.packages("haven")
}
# Load some example data
data("azprocedure", package = "COUNT")
# Write in SAS-format
haven::write_sas(azprocedure, "path/to/saved/file.sas7bdat")
```

1.2. Verwendete Software

Um Code-Beispiele (und Output) übersichtlich zu halten werden einige R-Packages und Hilfsfunktionen verwendet, die hier kurz beschrieben werden um Code in späteren Abschnitten nachvollziehbar zu halten.
Siehe dazu auch Anhang C.

1.2.1. R-Packages

Zur Reproduktion der Beispiele sind insbesondere die folgenden `R` packages notwendig, die durch den angegebenen Code installiert werden, sofern sie nicht bereits verfügbar sind:

```r
# Data transformation / modelling
if (!("dplyr" %in% installed.packages())) install.packages("dplyr")
if (!("purrr" %in% installed.packages())) install.packages("purrr")
if (!("broom" %in% installed.packages())) install.packages("broom")

# Plots
if (!("ggplot2" %in% installed.packages())) install.packages("ggplot2")

# Modelling
if (!("lmtest" %in% installed.packages())) install.packages("lmtest")
if (!("msme" %in% installed.packages())) install.packages("msme")
if (!("VGAM" %in% installed.packages())) install.packages("VGAM")
if (!("gamlss" %in% installed.packages())) install.packages("gamlss")

# Data (and maybe modelling)
if (!("COUNT" %in% installed.packages())) install.packages("COUNT")

# Output formatting (for RMarkdown/pandoc markdown documents)
if (!("pander" %in% installed.packages())) install.packages("pander")
if (!("kableExtra" %in% installed.packages())) install.packages("kableExtra")
```

Siehe auch Anhang C zu verwendeten Packages.

1.2.2. Funktionen

Weiterhin werden einige Hilfsfunktionen im Laufe des Dokuments verwendet, die primär der Abkürzung und/oder der Formatierung des Outputs dienen:

```
#' Simple descriptive stats for count variables
#' @param x A count variable, presumed to be a non-negative integer.
#' @param digits Number of digits to round statistics to.
#' @return Nichts, nur print output.
describe_counts <- function(x, digits = 2) {
   require(kableExtra)

   tibble::tibble(
      n = length(x),
      missing = sum(is.na(x)),
      mean = round(mean(x, na.rm = TRUE), digits),
      var = round(var(x, na.rm = TRUE), digits),
      range = paste0("[", paste0(range(x, na.rm = TRUE), collapse = ", "), "]")
   ) %>%
   setNames(c("N", "Missing", "Mittelwert", "Varianz", "Range")) %>%
   kable(booktabs = TRUE, escape = FALSE, linesep = "") %>%
   kable_styling(position = "center", protect_latex = TRUE)
}
# Example usage
# Define random count variable...
x <- rpois(100, 5)

# ...with some missings
x[sample(100, 10)] <- NA

# A basic summary
describe_counts(x)
```

N	Missing	Mittelwert	Varianz	Range
100	10	5.21	5.43	[1, 13]

Die Pearson-Dispersionsstatistik (eingeführt in Abschnitt 2.2, Definition 2.5):

```
#' Pearson-Dispersion
#' @param model Ein `glm`-Objekt mit Methoden für `resid()` und `df.residual`-Komponente
#' @param type Entweder 'pearson' (default) oder 'deviance'.
#' @return Invisible: Liste mit chi^2-Statistik, Freiheitsgraden und Dispersion.
```

```
#'          Printed: Formatiertes output.
dispersion <- function(model, type = "pearson") {
  chisq <- sum(resid(model, type = type)^2)
  disp  <- chisq / model$df.residual

  invisible(list(chi2 = chisq, df = model$df.residual, dispersion = disp))
  cat(sprintf("X-squared(%i) = %.2f\n%s Dispersion = %.3f", model$df.residual,
              chisq, chartr("pd", "PD", type), disp))
}
```

Der *Lagrange Multiplier Test*:

```
#' Lagrange Multiplier Test according to Hilbe (2014)
#' @param model A `glm` or similiar object.
#' @return Chi^2 statistic and corresponding p-value in a list of class "htest".
lagrange_test <- function(model) {
   mu     <- predict(model, type = "response")
   n_ybar <- length(model$y) * mean(mu)
   mu2    <- mean(mu^2) * length(model$y)
   chisq  <- (mu2 - n_ybar)^2 / (2 * mu2)
   names(chisq) <- "X-squared"
   pval   <- pchisq(chisq, df = 1, lower.tail = FALSE)
   df <- 1
   names(df) <- "df"

   rval <- list(
      statistic = chisq, parameter = df, p.value = pval,
      alternative = "Data is overdispersed",
      method = "Lagrange Multiplier Test",
      data.name = deparse(mod$call)
   )

   class(rval) <- "htest"
   rval
}
```

2. Das Poisson-Modell

Das Poisson-Modell ist die allgemeine Grundlage für die Modellierung von Zählvorgängen / Counts, und auch wenn es in seiner „reinen“ Form in der Praxis meist nicht ausreicht, bauen alle weiteren Methoden auf die eine oder andere Art darauf auf. Auch die häufig verwendete Negative Binomialverteilung ist letztlich eine Poisson-Verteilung mit Gamma-verteilter Varianz,

also eine Erweiterung der Poisson um einen zusätzlichen Parameter. Das gleiche Prinzip findet sich in allen hier besprochenen Verteilungen.

Für eine ausführliche Diskussion der Eigenschaften, siehe Winkelmann (2010) (p. 7-20).

Grundlage zur Modellierung ist das GLM mit den dazugehörigen Voraussetzungen:

- Die abhängige Variable Y kommt aus der Exponentialfamilie (Normal-, Poisson-, Gamma-, Binomialverteilung)
- Modelliert wird der lineare Prädiktor $\eta_i = x_i'\beta$.
- Response- und Linkfunktionen $h(x)$ und $g(x) = h^{-1}(x)$

Definition 2.1 (Poisson-Verteilung)**.** Eine Poisson-verteilte Zufallsvariable $Y \sim \text{Poi}(\mu)$ [3] hat die Dichte

$$\text{P}(Y = y) = \frac{\mu^y \exp(-\mu)}{y!} \quad , y \in \mathbb{N}_0$$

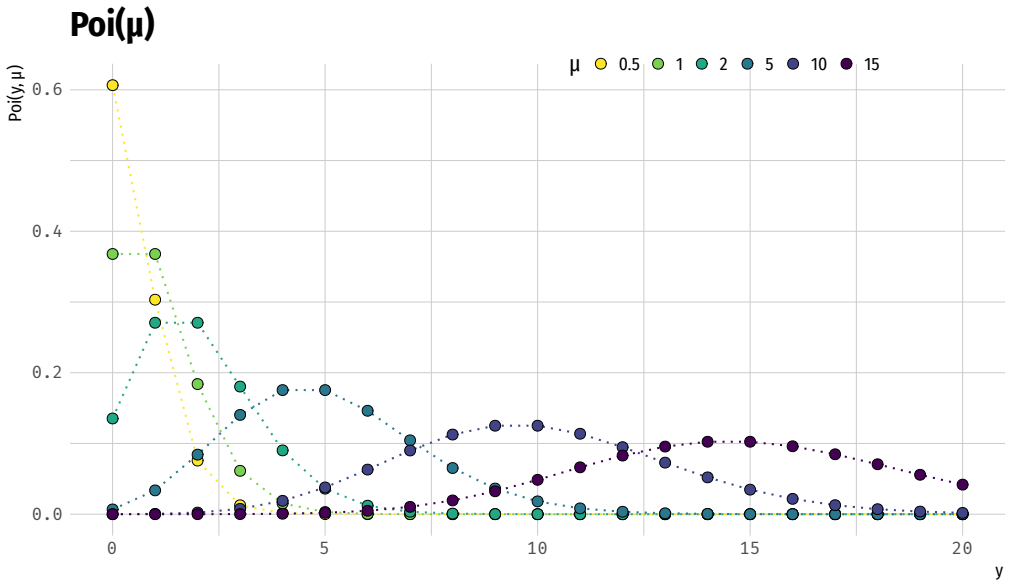


Abbildung 1: Poisson-Verteilungen mit ausgewählten Parametern.

Definition 2.2 (Poisson-Modell)**.** Die Zielvariablen $y_i \in \mathbb{N}_0$ sind (bedingt) unabhängig gegeben der Kovariablen $x_{i1}, x_{i2}, \ldots, x_{ik}$.

[3] Der Poisson-Parameter wird von unterschiedlichen Autoren als λ oder μ bezeichnet, und ich habe mich noch nicht für eine Variante entschieden).

Die Rate $\mu_i = \mathbb{E}(y_i \mid \mathbf{x}_i)$ der Poissonverteilung wird in der Regel log-linear modelliert als

$$\log(\mu_i) = \eta_i = \mathbf{x}_i'\beta = \beta_0 + \beta_1 x_1 + ... + \beta_k x_{ik}$$
$$\mu_i = \exp(\eta_i) = \exp(\beta_0) \cdot \exp(\beta_1 x_1) \cdot ... \cdot \exp(\beta_k x_k)$$

Vgl. Fahrmeir et al. (2009)

Für das log-lineare Poisson-Modell entsprechen die resultierenden Koeffizienten der Veränderung der log-counts – durch Exponentiation lassen diese sich als *incidence rate ratios* (*IRR*) interpretieren.

Um auf ungleiche Expositionsdauern oder -gebiete zu adjustieren wird ein **offset** (oder auch *exposure*) benötigt. Dazu dient der Koeffizient t, der die Länge der Zeit unter Exposition angibt:

$$f(y, \mu) = \frac{\exp(\mu)(t\mu)^y}{y!} \tag{1}$$

Damit entspricht $t\mu$ der Inzidenzrate des Outcomes adjustiert auf e.g. die geographische Lage oder Expositionsdauer. Ohne Offset entspräche $t = 1$. Für als Offset wird in der Regel $\log(t)$ verwendet, womit gelten:

$$\hat{\mu} = \exp x\beta + \log(t)$$
$$\Leftrightarrow \exp(x\beta) = \frac{\hat{\mu}}{t}$$
$$\Leftrightarrow \hat{\mu} = t \exp(x\beta)$$

Ein offset kann z.B. in `R` via `+ offset(log(variable))` in der model `formula` oder über das Argument `offset = log(variable)` in `glm` und verwandten Funktionen angegeben werden.

2.1. Annahmen

1. Die Abhängige / Zielvariable Y muss eine Zählung sein, i.e. die Verteilung ist diskret mit einem einzelnen Parameter μ der Poisson-Verteilung für Erwartungswert und Varianz.
2. $y \in \mathbb{N}_0$, insbesondere: Y muss 0 enthalten *können* (siehe auch B.2 Truncation und Censoring)
3. Beobachtungen sind *unabhängig*, i.e. weder longitudinal noch gepoolt.
    - Möglicher Test durch Vergleich der Modell-SE und der SE adjustiert durch robuste sandwich-estimators (siehe 4.1.3.1): Große Unterschiede implizieren korrelierte Daten
4. Balanced: Die Zellen sind in etwa so besetzt wie es aufgrund der Poissonverteilung erwartet wird.

5. Erwartungswert und Varianz sind identisch (*equidispersion*), i.e. ein größerer Erwartungswert impliziert auch eine größere Varianz (siehe 2.2).
6. Die χ^2-Statistik hat einen Wert nahe 1, i.e. beobachtete und erwartete Varianzen der response sind gleich (Dispersionsindex).

(Nach Tabelle in (Hilbe, 2014))

Alternative Formulierung nach Winkelmann (2010) (p. 64):

1. $f(y \mid \mu) = \frac{e^{-\mu}\mu^y}{y!} \quad \mu > 0, y = 0, 1, 2, ...$
2. $\mu = \exp(\mathbf{x}'\beta)$.
3. Beobachtungspaare (y_i, x_i) sind unabhängig verteilt.

2.2. Dispersion

Definition 2.3 (Equi-, Extra-, Over-, Underdispersion)**.** Für Zählvariablen $y_i \in \mathbb{N}_0$ und erklärenden Variablen $\mathbf{x}_i$ gilt innerhalb eines Modells:

$$\begin{aligned} \text{Equdispersion:} &\quad \operatorname{Var}(y_i) = \operatorname{Var}(y_i \mid \mathbf{x}_i) \\ \text{Extradispersion:} &\quad \operatorname{Var}(y_i) \neq \operatorname{Var}(y_i \mid \mathbf{x}_i) \\ \text{Overdispersion:} &\quad \operatorname{Var}(y_i) > \operatorname{Var}(y_i \mid \mathbf{x}_i) \\ \text{Underdispersion:} &\quad \operatorname{Var}(y_i) < \operatorname{Var}(y_i \mid \mathbf{x}_i) \end{aligned}$$

Definition 2.4 (Poisson-Overdispersion)**.** Innerhalb eines Poisson-Modells (vgl. Abschnitt 2) mit der Annahme

$$y_i \mid \mathbf{x}_i \sim \text{Poi}(\mu_i)$$

$$\mu = \mathbb{E}(y_i \mid \mathbf{x}_i) = \operatorname{Var}(y_i \mid \mathbf{x}_i) \quad \text{(Equidispersion)}$$

spricht man von **Poisson-Overdispersion** wenn die Varianz der Beobachtungen die erwartete Varianz des Poisson-Modells übersteigt:

$$\operatorname{Var}(y_i \mid \mathbf{x}_i) > \mathbb{E}(y_i \mid \mathbf{x}_i)$$

in einem Modell mit overdispersion gilt die Annahme:

$$\operatorname{Var}(y_i \mid \mathbf{x}_i) = \theta \cdot \mu_i$$

Mit *Dispersionsparameter* θ (vgl. Fahrmeir et al., 2009, p. 210)

Als Dispersionsstatistik können *deviance dispersion* oder *Pearson dispersion* berechnet werden. Laut Hilbe (2014) ist die *Pearson dispersion* zu bevorzugen, da sie für echte Poisson-Modelle gleich 1 ist, wohingegen die *deviance dispersion* nach oben verzerrt ist.

Definition 2.5 (Pearson-Dispersion)**.** Nach Hilbe (2014) (p. 77ff):

Die Pearson χ^2-Statistik ist die Summe der quadrierten (Pearson-)Residuen gewichtet mit der Modellvarianz:

$$\chi^2_{\text{Pearson}} = \sum_{i=1}^{n} \frac{(y_i - \hat{\mu}_i)^2}{\text{Var}(\hat{\mu}_i)}$$

und die **Pearson-Dispersionsstatistik**:

$$D = \frac{\chi^2_{\text{Pearson}}}{\text{df}}$$

Mit der Interpretation

$$\text{D} = \begin{cases} < 1 & \Longrightarrow \text{Underdispersion} \\ 1 & \Longrightarrow \text{Equidispersion (Poisson)} \\ > 1 & \Longrightarrow \text{Overdispersion} \end{cases}$$

Für Modelle moderater Größe kann man ab Werten über 1.25 von overdispersion sprechen, wobei für große Stichproben auch schon ab 1.05 overdispersion vorliegen kann – zumindest nach Hilbe (2014) (p. 82), der aber leider keine konkreten Angaben für seine Definition von „moderaten“ oder „großen“ Stichproben macht.

In `R` kann die Pearson-Dispersion wie folgt berechnet werden:

```
# Model fit
mod <- glm(y ~ x1 + x2 + x3, data = sim, family = poisson(link = "log"))

# Pearson dispersion
sum(resid(mod, type = "pearson")^2) / mod$df.residual
```

...wofür wir in Abschnitt 1.2.2 eine Hilfsfunktion `dispersion()` definiert haben.

2.2.1. Overdispersion

Der vermutlich häufigste Fall für Count-Daten: Die Varianz der abhängigen Variable ist größer als ihr Erwartungswert, bzw. größer als ihre erwartete Varianz innerhalb eines Modells. Hilbe

(2014) unterscheidet zwischen echter und scheinbarer (*apparent*) Overdispersion, wobei letztere oft durch geeignete Korrekturen kompensiert werden kann, wobei *echte* Overdisperion sowohl Parameterschätzung als auch Modellanpassung im Allgemeinen beeinträchtigt.

Nach Hilbe (2014) (p. 82) entsteht *echte* Overdispersion durch:

- Positive Korrelation zwischen responses
- Große Variation zwischen response-probabilities und counts
- Verletzungen der Verteilungsannahme (i.e. Poissonverteilung)
- „Proneness“: Frühere Ereignisse beeinflussen das Auftreten späterer Ereignisse [4]

Ursachen für *scheinbare* (*apparent*), und damit (bedingt) korrigierbare Overdispersion nach Hilbe (2014) (p. 41, 82):

1. Fehlende explanatorische Prädiktoren
2. Ausreißer
3. Fehlende Interaktionsterme
4. Ein Prädiktor muss transformiert werden
5. Die Daten sind zu dünn besetzt (*sparse*)
6. Fehlende Werte, die nicht zufällig sind (missing not at random, *MNAR* – siehe auch Anhang B.1

Ein einfaches simuliertes Beispiel zur Auswirkung von fehlenden Prädiktoren:

```
# Generate binary variable in [0, 1] with a given proportion of 1's
rbinary <- function(n, prob = 0.5) {
  sample(0:1, size = n, replace = TRUE, prob = c(1 - prob, prob))
}

set.seed(436)
n <- 1000

sim <- tibble(
  x1 = rbinary(n, .1),
  x2 = rbinary(n, .2),
  x3 = rbinary(n, .3),
  eta = 0.5 + 1 * x1 + 2 * x2 + 0.5 * x3,
  mu = exp(eta),
  py = rpois(n, mu)
)

# Korrektes modell:
```

[4] Diese Annahme (Unabhängigkeit der Ereignisse) der Poissonverteilung ist auch der Grund, warum sich die Poissonverteilung prinzipiell **nicht** eignet um Epidemien wie Ebola zu modellieren. Freundliche Grüße an Frau Pigeot, „Statistische Modellierung I“, WiSe 18/19.

```r
mod <- glm(py ~ x1 + x2 + x3, data = sim, family = poisson(link = "log"))
dispersion(mod)
```

```
#> X-squared(996) = 1029.70
#> Pearson Dispersion = 1.034
```

```r
# Modell mit fehlendem Prädiktor:
mod2 <- glm(py ~ x1 + x3, data = sim, family = poisson(link = "log"))
dispersion(mod2)
```

```
#> X-squared(997) = 7669.65
#> Pearson Dispersion = 7.693
```

2.2.2. Underdispersion

Underdispersion ist der Fall, wenn vorliegende Daten eine geringere Varianz aufweisen, als auf Basis eines Poisson-Modells erwartet würde, das heißt die Daten sind „enger zusammengeklumpt". Bei underdispersion werden die Standardfehler des Modells *über*schätzt, im Gegensatz zur overdispersion, bei der Standardfehler *unter*schätzt werden (Hilbe, 2014, p. 210).

Im Allgemeinen wird für diese Situation die generalized Poisson empfohlen[5] (Hilbe, 2014), da diese Erweiterung der Poisson-Verteilung nicht nur einen zusätzlichen Parameter für die Varianz hat (analog NB, PIG), sondern dieser Parameter auch negativ sein kann.

Weiterhin taucht im Kontext von hurdle models (siehe B.3) folgende Bemerkung auf:

> [...] that underdispersion occurs if **zeros are less frequent than the parent distribution would predict**. The higher the expected value of the Poisson distribution, the lower the predicted probability of zero outcome and the lower the scope for underdispersion. – (Winkelmann, 2010, p. 180 (eigene Hervorhebung))

3. Mehrparametrische Modelle

Zweiparametrische Modelle haben neben dem Parameter für den Erwartungswert einen weiteren Parameter für die Dispersion, was wir natürlich insbesondere im Kontext der Dispersionsproblematik sehr nützlich finden.
Alle hier aufgeführten Modelle (inklusive der zero-inflation models) können als Verallgemeinerung der Poisson-Verteilung um (mindestens) einen weiteren Parameter aufgefasst werden. Die Unterschiede liegen hauptsächlich in der Parametrisierung und den damit zusammenhängenden

[5] https://stats.stackexchange.com/a/237177/80056

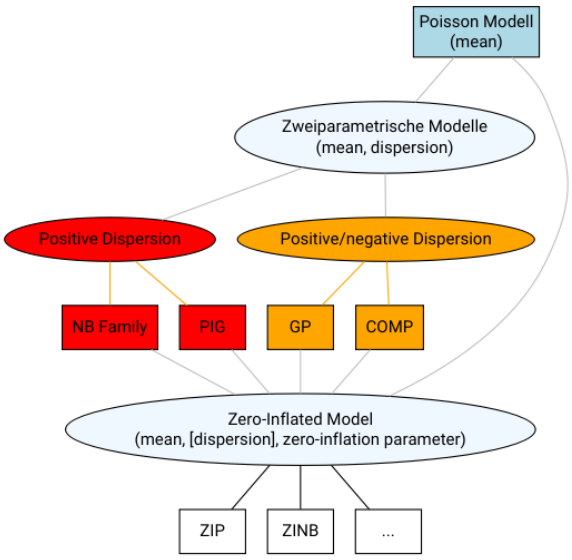


Abbildung 2: Hierarchie ausgewählter Count-Modelle. In Klammern: Zu schätzende Parameter der zugrundeliegenden Verteilung

Einschränkungen. Die *Negative Binomialverteilung* und die *Poisson Inverse Gaussian* zum Beispiel erweitern beide die Poisson um einen Dispersionsparameter, machen aber unterschiedliche Annahmen zur Verteilung der Varianz (gamma- vs. invers-normalverteilt).

Tabelle 1 zeigt eine Reihe von Verteilungen mit entsprechend parametrisierter Varianz abhängig vom Erwartungswert μ und einem Dispersionsparameter α.

3.1. Negativ Binomial (NB)

Die Negative Binomialverteilung (*NB*) kann als zweiparametrische Erweiterung der *Poisson* betrachtet werden, wobei die Varianz (separat vom Erwartungswert) als Gamma-verteilte Zufallsvariable betrachtet wird (*Poisson-Gamma Mischmodell*).
Für eine ausführliche Beschreibung, siehe Hilbe (2014) (p. 126ff).

Die Wahrscheinlichkeitsfunktion mit der α-Parametrisierung:

Definition 3.1 (Negative Binimialverteilung 2, alpha)**.** Nach (Hilbe, 2014, p. 130):

Für $Y \sim NB(\mu, \alpha)$ gilt

$$P(Y = y_i) = \begin{pmatrix} y_i + \frac{1}{\alpha} - 1 \\ \frac{1}{\alpha} - 1 \end{pmatrix} \left(\frac{1}{a + \alpha\mu_i} \right)^{\frac{1}{\alpha}} \left(\frac{\alpha\mu_i}{1 + \alpha\mu_i} \right)$$

Tabelle 1: Ein Überblick einiger Modelle mit Varianzparametrisierung (nach @hilbeModelingCountData2014, p. 12).

Model	Mean	Variance
Poisson	μ	μ
Negative Binomial I (NB1)	μ	$\mu(1+\alpha) = \mu + \alpha\mu$
Negative Binomial II (NB2)	μ	$\mu(1+\alpha\mu) = \mu + \alpha\mu^2$
Negative Binomial-P (NBP)	μ	$\mu(1+\alpha\mu^p) = \mu + \alpha\mu^p$
Poisson Inverse Gaussian (PIG)	μ	$\mu(1+\alpha\mu^2) = \mu + \alpha\mu^3$
Generalized Poisson (GP)	μ	$\mu(1+\alpha\mu)^2 = \mu + 2\alpha\mu^3 + \alpha^2\mu^3$

Mit $\mathbb{E}(Y) = \mu$ und $\text{Var}(Y) = \mu + \alpha\mu^2$

Eine Alternative unter Verwendung von $\theta = \frac{1}{\alpha}$:

Definition 3.2 (Negative Binomialverteilung 2, theta)**.** Nach Perumean-Chaney et al. (2013) (p. 1675):

Für $Y \sim NB(\mu, \theta)$ gilt

$$P(Y = y) = \frac{\Gamma(\theta + y)}{\Gamma(\theta)\Gamma(y+1)} \frac{\theta^\theta \mu^y}{(\theta+\mu)^{(\theta+y)}}, \quad y = 0, 1, 2, ...$$

Mit $\mathbb{E}(Y) = \mu$ und $\text{Var}(Y) = \mu + \frac{\mu^2}{\theta}$

Der gängigste Anwendungsfall der *NB* findet sich bei Daten mit nicht korrigierbarer overdispersion (siehe Abschnitt 2.2.1), da der Parameter θ bzw. auch $\alpha = \frac{1}{\theta}$ als *Dispersionsparameter* dient:

Definition 3.3 (NB-Dispersionsparameter)**.** Meist wird der Parameter als α bezeichnet, mit der Interpretation (relativ zum Poisson-Modell):

$$\text{Var}(Y) = \mu + \alpha\mu^2$$

$$\begin{aligned} \alpha = 0 \quad &\Longrightarrow \text{ Äquivalent zur Poissonverteilung} \\ \alpha > 0 \quad &\Longrightarrow \text{ Overdispersion} \end{aligned}$$

Je nach Quelle (und unter Anderem in R (z.B. `MASS::glm.nb`)) wird die inverse Variante $\theta = \frac{1}{\alpha}$ verwendet, womit gilt:

$$\text{Var}(Y) = \mu + \frac{\mu^2}{\theta}$$

$$\begin{aligned} \theta > 0 \quad &\Longrightarrow \text{ Overdispersion} \\ \theta \to \infty \quad &\Longrightarrow \text{ Äquivalent zur Poissonverteilung} \end{aligned}$$

Siehe dazu auch Hilbe (2014) (p. 131).

Das GLM sieht keinen weiteren zu schätzenden Parameter vor, weshalb θ bzw. α in der Praxis separat geschätzt wird [6].

Der wohl wichtigste Aspekt des Dispersionsparameters, unabhängig davon ob α oder θ verwendet wird, ist sein Vorzeichen: Er ist *immer* positiv, das heißt die Varianz der NB ist entweder *größer oder gleich* der Poisson, oder mit anderen Worten:

> „The negative binomial model adjusts for Poisson overdispersion; it **cannot be used to model underdispersed** Poisson data"
>
> — (Hilbe, 2014, (p. 11), eigene Hervorhebung)

Je nach Software/Algorithmus ist es auch möglich, dass ein NB-fit nicht möglich ist, wenn $\alpha \approx 0$ (die Verteilung zu nah an Poisson) bzw. die Daten Poisson-underdispersed sind.

Weiterhin gibt es zwei verschiedene Formulierungen der Negativen Binomialverteilung, NB1 und NB2, deren Nummerierung auf den Exponenten im zweiten Term ihrer Varianzen zurückzuführen ist:

Definition 3.4 (NB1 und NB2)**.** Nach Hilbe (2014) (p. 126f):

Die **NB1**, oder auch *lineare* Negative Binomialverteilung hat die Varianz

$$\text{Var}(Y) = \mu + \alpha\mu$$

...und die gängigere **NB2**, oder auch *quadratische* Negative Binomialverteilung hat (wie oben beschrieben) die Varianz

$$\text{Var}(Y) = \mu + \alpha\mu^2$$

Beide Varianten können mittels MLE geschätzt werden, wobei NB2 zusätzlich im Kontext des GLM via IRLS[7] geschätzt werden kann.

[6] Das ist auch der Grund, warum in R nicht `glm()` verwendet werden kann, sondern eine separate Implementation mit expliziter Schätzung von θ benötigt wird (`MASS::glm.nb`)

[7] https://en.wikipedia.org/wiki/Iteratively_reweighted_least_squares

Als link wird $\log \mu$ verwendet, wobei der *canonical link* eigentlich $\log\left(\frac{\alpha\mu}{1+\alpha\mu}\right)$ ist [8].

Für ein gut angepasstes NB-Modell gilt analog eines Poisson-Modells, dass die Dispersionsstatistik approximativ 1 ist (siehe 2.5).

Zur Anwendung in R gibt es zwei Möglichkeiten: `MASS::glm.nb` oder `msme::nbinomial`, wobei letztere mehrere Optionen zur Parametrisierung hat und zusätzlich die Dispersionsstatistik ausgibt:

```
mod_nb <- MASS::glm.nb(docvis ~ outwork + age,
                       data = rwm1984)
summary(mod_nb)
```

```
#>
#> Call:
#> MASS::glm.nb(formula = docvis ~ outwork + age, data = rwm1984,
#>     init.theta = 0.4353451729, link = log)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -1.5316  -1.2816  -0.4895   0.1043   5.0468
#>
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept) -0.039550   0.106669  -0.371    0.711
#> outwork      0.414662   0.055451   7.478 7.55e-14 ***
#> age          0.022146   0.002396   9.242  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for Negative Binomial(0.4353) family taken to be 1)
#>
#>     Null deviance: 4100.8  on 3873  degrees of freedom
#> Residual deviance: 3909.6  on 3871  degrees of freedom
#> AIC: 16674
#>
#> Number of Fisher Scoring iterations: 1
#>
#>
#>               Theta:  0.4353
#>           Std. Err.:  0.0135
```

[8] Daher können Standardfehler nicht wie für canonical models üblich via OIM (*observed information matrix*) geschätzt werden, sondern via EIM (*expected information matrix*). Die EIM-Methode liefert weniger genaue Resultate für kleine Stichproben ($n < 30$). So jedenfalls Hilbe (2014).

```
#>
#>  2 x log-likelihood:  -16665.5250
```

```
mod_nb2 <- msme::nbinomial(docvis ~ outwork + age,
                           data = rwm1984)
summary(mod_nb2)
```

```
#>
#> Call:
#> ml_glm2(formula1 = formula1, formula2 = formula2, data = data,
#>     family = family, mean.link = mean.link, scale.link = scale.link,
#>     offset = offset, start = start, verbose = verbose)
#>
#> Deviance Residuals:
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> -1.5320 -1.2816 -0.4896 -0.4553  0.1035  5.0497
#>
#> Pearson Residuals:
#>      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
#> -0.637147 -0.607737 -0.376257  0.000055  0.108975 21.659334
#>
#> Coefficients (all in linear predictor):
#>               Estimate      SE      Z          p     LCL    UCL
#> (Intercept)    -0.0443 0.10570 -0.419     0.675 -0.2514 0.1629
#> outwork         0.4141 0.05532  7.485  7.15e-14  0.3056 0.5225
#> age             0.0223 0.00237  9.401  5.42e-21  0.0176 0.0269
#> (Intercept)_s   2.2971 0.07098 32.364 8.85e-230  2.1580 2.4362
#>
#> Null deviance: 4100.786  on  3872 d.f.
#> Residual deviance: 3909.522  on  3870 d.f.
#> Null Pearson: 5835.302  on  3872 d.f.
#> Residual Pearson: 5458.2  on  3870 d.f.
#> Dispersion: 1.410388
#> AIC:  16673.53
#>
#> Number of optimizer iterations:  82
```

3.2. Poisson Inverse Gaussian (PIG)

So wie die *NB* als mixture aus Poisson-Verteilung mit Gamma-verteilter Varianz betrachtet werden kann, ist die *Poisson Inverse Gaussian* (*PIG*) eine mixture aus Poisson-Verteilung mit invers-normalverteilter[9] Varianz.

[9] https://en.wikipedia.org/wiki/Inverse_Gaussian_distribution

Die Verteilung hat Erwartungswert und Varianz mit gleicher Form wie die NB2, unterscheidet sich aber in ihrer Form – sie ist linkssteiler und hat längere tails.

Die *PIG* kann also alternativ zur *NB* verwendet werden, insbesondere für den Fall sehr linkssteiler Daten.

> „Simply put, PIG models can better deal with highly overdispersed data that can negative binomial regression, particularly data clumped heavily at 1 and 2“
>
> – Hilbe (2014) (p. 163)

Siehe auch

Software:

- R: `gamlss` (CRAN)
- SAS: Machbar via `NLMIXED`, aber unschön (siehe High, 2018).
- Stata: `pigreg`

Vergleichen wir ein PIG Modell mit unserem vorigen NB2-Modell:

```
# Poisson Modell
mod_pois <- glm(docvis ~ outwork + age, family = poisson(),
              data = rwm1984)
# NB2 Modell
mod_nb <- MASS::glm.nb(docvis ~ outwork + age,
                       data = rwm1984)

# PIG Modell
mod_pig <- gamlss::gamlss(docvis ~ outwork + age, family = gamlss.dist::PIG,
                          data = rwm1984)
```

```
#> GAMLSS-RS iteration 1: Global Deviance = 16757.45
#> GAMLSS-RS iteration 2: Global Deviance = 16757.15
#> GAMLSS-RS iteration 3: Global Deviance = 16757.11
#> GAMLSS-RS iteration 4: Global Deviance = 16757.11
#> GAMLSS-RS iteration 5: Global Deviance = 16757.11
```

```
summary(mod_pig)
```

```
#> ******************************************************************
#> Family:  c("PIG", "Poisson.Inverse.Gaussian")
#>
#> Call:
#> gamlss::gamlss(formula = docvis ~ outwork + age, family = gamlss.dist::PIG,
#>     data = rwm1984)
```

```
#>
#> Fitting method: RS()
#>
#> ------------------------------------------------------------------
#> Mu link function:  log
#> Mu Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -0.285765   0.112128  -2.549   0.0109 *
#> outwork      0.527637   0.058037   9.091   <2e-16 ***
#> age          0.026861   0.002514  10.684   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> ------------------------------------------------------------------
#> Sigma link function:  log
#> Sigma Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  1.34417    0.05564   24.16   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> ------------------------------------------------------------------
#> No. of observations in the fit:  3874
#> Degrees of Freedom for the fit:  4
#>       Residual Deg. of Freedom:  3870
#>                       at cycle:  5
#>
#> Global Deviance:     16757.11
#>             AIC:     16765.11
#>             SBC:     16790.16
#> ******************************************************************
```

```
cbind(
  exp(coef(mod_pois)),
  exp(coef(mod_nb)),
  exp(coef(mod_pig))
) %>%
  as_tibble(rownames = ".id") %>%
  setNames(c("Koeffizient", "Poisson", "NB2", "PIG"))
```

```
#> # A tibble: 3 x 4
#>   Koeffizient Poisson   NB2   PIG
#>   <chr>         <dbl> <dbl> <dbl>
```

```
#> 1 (Intercept)   0.967 0.961 0.751
#> 2 outwork       1.50  1.51  1.69
#> 3 age           1.02  1.02  1.03
```

Zum Modellvergleich kann das AIC verwendet werden:

```
AIC(mod_pois, mod_nb, mod_pig)
```

```
#>          df      AIC
#> mod_pois  3 31278.78
#> mod_nb    4 16673.52
#> mod_pig   4 16765.11
```

3.3. Generalized Poisson (GP)

Die Verteilung besitzt auch einen zusätzlichen Dispersionsparameter (analog NB, PIG), allerdings kann dieser hier auch negative Werte annehmen, womit die GP für alle Dispersionsarten (bzw. insbesondere underdispersion) geeignet ist.

Definition 3.5 (Generalized Poisson)**.** Aus Hilbe (2014) (p. 211) und Harris et al. (2012):

Eine Variable Y ist *generalized Poisson* verteilt mit der PMF:

$$P(Y = y_i) = \frac{\theta_i \left(\theta_i + \delta y_i\right)^{y_i - 1}}{y_i!} \exp(-\theta_i - \delta y_i), \quad y_i = 1, 2, 3, ...$$

$$\theta_i > 0, \quad \max(-1, \frac{-\theta_i}{4}) < \delta < 1$$

$$\mu_i = \mathbb{E}(Y_i) = \frac{\theta_i}{1 - \delta}$$
$$\text{Var}(Y_i) = \frac{\theta}{(1-\delta)^3} = \frac{1}{(1-\delta)^2}\mathbb{E}(Y_i) = \Phi\mathbb{E}(Y_i)$$

Wobei $\Phi = \frac{1}{(1-\theta)^2}$ als Dispersionsparameter dient.

Es gilt außerdem

$$\delta = 0 \Longrightarrow \text{Equidispersion (Poisson)}$$
$$\delta < 0 \Longrightarrow \text{Underdispersion}$$
$$\delta > 0 \Longrightarrow \text{Overdispersion}$$

Auch hier kann ein Likelihood-Ratio Test auf $\delta = 0$ durchgeführt werden, analog $\alpha = 0$ für die NB – wobei δ hier keine Restriktion hat (siehe BLR, Definition 4.3).

Eine alternative Darstellung unter Verwendung von Parametern μ und *sigma* findet sich in Stasinopoulos and Rigby (2007), der Grundlage für das R-package `gamlss` (*„Generalized additive models for location, scale and shape"*):

Definition 3.6 (Generalized Poisson (GAMLSS))**.**

$$f(y \mid \mu, \sigma) = \left(\frac{\mu}{(1+\sigma+\mu)}\right)^{y} \frac{(1+\sigma y)^{(y-1)}}{y!} \exp\left(-\mu \frac{(1+\sigma y)}{(1+\sigma+\mu)}\right)$$

Für $y = 0, 1, 2, ...$ mit $\mu > 0$ und $\sigma > 0$.

Software:

- R: `VGAM`: `vgam(..., family = genpoisson())]`
  - Hier wird das obige δ als λ bezeichnet (Verwechslungsgefahr mit Poisson-Parameter)
  - Numerisch instabil für λ nahe 0 oder 1
  - Siehe `?VGAM::genpoisson`[10]
- Alternativ via gamlss[11]: `gamlss(..., family = GPO)`

In beiden R-Varianten habe ich bisher nicht geschafft θ, den Dispersionsparameter, zu extrahieren – auch wenn die Koeffizientien aus dem Stata-Beispiel in (Hilbe, 2014, p. 213f) reproduzierbar sind.

- SAS: `NLMIXED` oder `FMM`, siehe auch SAS support document[12]

Allgemein scheint die Parametrisierung (und Notation der Parameter) zu variieren.

3.4. Conway-Maxwell Poisson (COMP)

Definition 3.7 (Conway-Maxwell-Poisson Modell (CMP))**.** Nach Shmueli et al. (2005) (p. 129):

$$P(X = x) = \frac{\lambda^x}{(x!)^\nu} \frac{1}{Z(\lambda, \nu)}, \quad x = 0, 1, 2, ... \tag{2}$$

$$Z(\lambda, \nu) = \sum_{j=0}^{\infty} \frac{\lambda^j}{(j!)^\nu} \tag{3}$$

$$\tag{4}$$

$$\lambda > 0, \nu \geqslant 0 \tag{5}$$

[10] https://rdrr.io/cran/VGAM/man/genpoisson.html
[11] https://www.gamlss.com/
[12] http://support.sas.com/kb/56/549.html

Mit (im Unterschied zur Poisson) nichtlinearer Zerfallsrate ν *rate of decay*, so dass:

$$\frac{P(X = x - 1)}{P(X = x)} = \frac{x^\nu}{\lambda} \tag{6}$$

Der Zusammenhang zwischen ν und λ nach Sellers and Shmueli (2010) (p. 946) legt Nahe, dass ν ähnlich α in NB-Modellen die Dispersion bestimmt:

$$\begin{aligned}\text{Var}(Y_i) =& \lambda_i \frac{\partial}{\partial \lambda_i} \mathbb{E}(Y_i) \approx \lambda_i \frac{\partial}{\partial \lambda_i} \left(\lambda_i^{\frac{1}{\nu}} - \frac{\nu - 1}{2\nu} \right) \\ =& \frac{1}{\nu} \lambda^{\frac{1}{\nu}} \approx \frac{1}{\nu} \mathbb{E}(Y_i)\end{aligned}$$

Als link wird $\log \lambda$ verwendet, da so die links für Poisson- und logistische Regression Spezialfälle der COMP sind.

Die Verteilung hat als Spezialfälle:

- $\nu = 1 \implies Z(\lambda, \nu) = exp(\lambda)$: Poisson
- $\nu \to \infty \implies Z(\lambda, \nu) \to 1 + \lambda$: Bernoulli mit $P(X = 1) = \frac{\lambda}{1+\lambda}$
- $\nu = 0,\ \lambda < 1$: Geometrisch mit $P(X = x) = \lambda^x (1 - \lambda)$

Vorteile:

- „Brücke“ zwischen logistischer und Poisson-Regression (Sellers and Shmueli, 2010)

  > Although the logistic regression is a limiting case ($\nu \to \infty$), in practice, fitting a COM-Poisson regression to binary data yields estimates and predictions that are practically identical to those from a logistic regression.
  >
  > — Sellers and Shmueli (2010), p. 945

- „Low cost“, i.e. „nur“ ein zusätzlicher Parameter
- Einfach zu handhaben (wenn in GLM mit MLE statt MCMC estimation)
- Over- und underdispersion abbildbar
- Zero-inflation abbildbar

Software:

- `CompGLM::glm.comp`: Funktioniert, aber crasht RStudio wenn `nuFormula` spezifiziert wird
- `compoisson::com.fit`: Entweder generell nicht geeignet oder unfassbar langsam (und deshalb nicht geeignet)
- `COMPoissonReg` (GitHub: `lotze/COMPoissonReg`): Scheint zu funktionieren

Da die COMP-Wahrscheinlichkeitsfunktion eine unendliche Summe $Z(\lambda, \nu)$ enthält, kann es bei der Anwendung in Software durchaus dazu kommen, dass keine Konvergenz erreicht wird. Wie viele Iterationen für die Konvergenz der Summe benötigt werden hängt zum einen von ν ab: Je größer, desto schneller die Konvergenz. Das Gegenteil gilt für λ – je größer, desto langsamer die Konvergenz (High, 2018, p. 4).

Siehe auch

- Sellers and Shmueli (2010) für eine gute Übersicht
- Lord et al. (2010), Lord et al. (2008) für eine Anwendung
- Shmueli et al. (2005)

3.5. Diagnostische Modelle

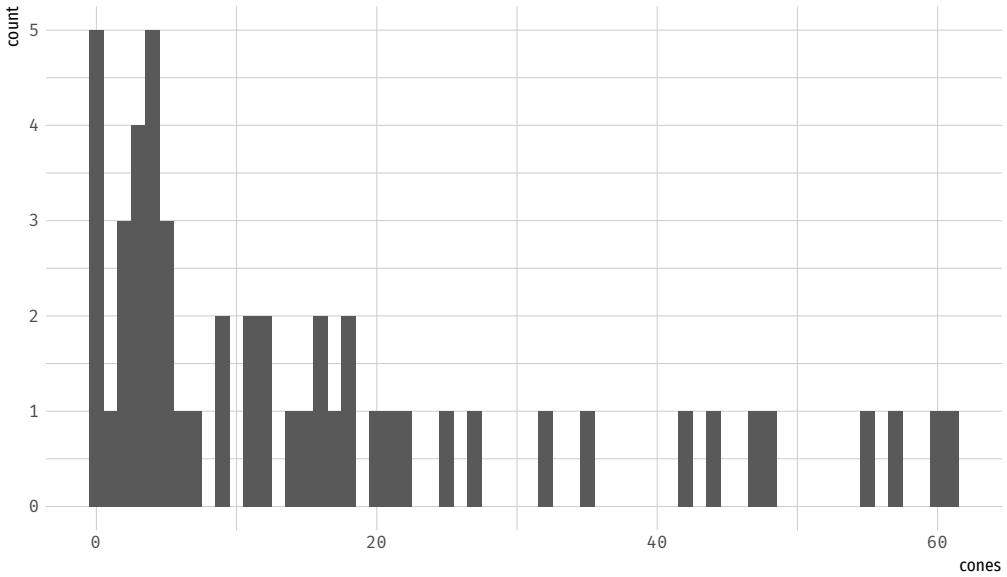
3.5.1. Heterogenous Negative Binomial (NBH)

- Selling feature: Erlaubt Parametrisierung von α
- -> Ursache für under-/overdispersion modellierbar

Hilbe (2014) (p. 156)

```
data(nuts, package = "COUNT")
nuts <- subset(nuts, dbh < 0.6)

ggplot(data = nuts, aes(x = cones)) +
  geom_histogram(binwidth = 1)
```




```
# Poisson-Modell als Startpunkt
m_pois <- glm(cones ~ sntrees + sheight + scover, family = poisson(), data = nuts)
dispersion(m_pois) # -> overdispersion
```

```
#> X-squared(47) = 643.01
#> Pearson Dispersion = 13.681
```

```
# NB-Modell
m_nb <- msme::nbinomial(cones ~ sntrees + sheight + scover, data = nuts)
summary(m_nb) # Dispersion < 1 -> NB-underdispersed
```

```
#>
#> Call:
#> ml_glm2(formula1 = formula1, formula2 = formula2, data = data,
#>     family = family, mean.link = mean.link, scale.link = scale.link,
#>     offset = offset, start = start, verbose = verbose)
#>
#> Deviance Residuals:
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> -2.4506 -1.0501 -0.4199 -0.3294  0.3910  1.6464
#>
#> Pearson Residuals:
```

```
#>       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
#> -1.0061432 -0.7292929 -0.3653465  0.0003247  0.4433920  2.6100790
#>
#> Coefficients (all in linear predictor):
#>               Estimate    SE     Z        p      LCL   UCL
#> (Intercept)      2.627 0.142 18.53 1.24e-76  2.34948 2.905
#> sntrees          0.306 0.161  1.90   0.0573 -0.00945 0.621
#> sheight          0.158 0.159  0.99    0.322 -0.15463 0.470
#> scover           0.536 0.179  3.00  0.00274  0.18516 0.886
#> (Intercept)_s    0.927 0.199  4.65 3.37e-06  0.53584 1.318
#>
#> Null deviance: 77.26177  on  49 d.f.
#> Residual deviance: 59.07252  on  46 d.f.
#> Null Pearson: 58.54636  on  49 d.f.
#> Residual Pearson: 41.19523  on  46 d.f.
#> Dispersion: 0.8955486
#> AIC:  383.8132
#>
#> Number of optimizer iterations:  53
```

```
# NB-H: Ursache der Dispersion?
m_nbh <- msme::nbinomial(cones ~ sntrees + sheight + scover,
                         formula2 = cones ~ sntrees + sheight + scover,
                         scale.link = "log_s",
                         family = "negBinomial",
                         data = nuts)
summary(m_nbh)
```

```
#>
#> Call:
#> ml_glm2(formula1 = formula1, formula2 = formula2, data = data,
#>     family = family, mean.link = mean.link, scale.link = scale.link,
#>     offset = offset, start = start, verbose = verbose)
#>
#> Deviance Residuals:
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> -2.7908 -0.9137 -0.3023 -0.3208  0.4419  1.5671
#>
#> Pearson Residuals:
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#> -1.14680 -0.70958 -0.23947  0.03088  0.31546  3.20165
#>
#> Coefficients (all in linear predictor):
```

```
#>                Estimate    SE      Z        p     LCL   UCL
#> (Intercept)     2.6147 0.144 18.159  1.1e-73  2.3325 2.897
#> sntrees         0.2731 0.110  2.478   0.0132  0.0571 0.489
#> sheight         0.0744 0.144  0.516    0.606 -0.2081 0.357
#> scover          0.5217 0.158  3.300 0.000967  0.2118 0.832
#> (Intercept)_s  -0.1950 0.238 -0.821    0.412 -0.6608 0.271
#> sntrees_s      -0.3834 0.323 -1.186    0.236 -1.0168 0.250
#> sheight_s       0.3312 0.321  1.033    0.302 -0.2973 0.960
#> scover_s        0.2723 0.417  0.652    0.514 -0.5459 1.091
#>
#> Null deviance: 85.07187  on  49 d.f.
#> Residual deviance: 59.49816  on  43 d.f.
#> Null Pearson: 45.30266  on  49 d.f.
#> Residual Pearson: 46.82167  on  43 d.f.
#> Dispersion: 1.088876
#> AIC:  385.7116
#>
#> Number of optimizer iterations:  54
```

Signifikanz der Dispersionsprädiktoren (Suffix `_s`) kann nun verwendet werden, um Ursachen der overdispersion zu identifizieren – auch wenn in diesem Beispiel keiner der Koeffizienten signifikant wird.

3.5.2. Negative binomial-P (NB-P)

Eine Erweiterung der NB1 und NB2, die den Exponenten im Varianzterm parametrisiert (ρ), womit die Dispersion nun nicht mehr statisch für alle Beobachtungen ist:

Definition 3.8 (NB-P)**.**

$$\begin{aligned} \text{NB-P:} \quad & \text{Var}(Y) = \mu + \alpha\mu^{\rho} \\ \text{NB1:} \quad & \text{Var}(Y) = \mu + \alpha\mu \\ \text{NB2:} \quad & \text{Var}(Y) = \mu + \alpha\mu^2 \end{aligned}$$

Damit ist NB-P äquivalent zu NB2 für $\rho = 2$ bzw. NB1 für $\rho = 1$.
Um zu evaluieren, ob ein NB1 oder NB2 Modell einen besseren Fit liefert, kann man nun ein NB-P Modell zum Vergleich fitten und Likelihood Ratio Tests zum Vergleich heranziehen.

Siehe auch Hilbe (2014) (p. 155) – wo der Autor zwar den obigen Ansatz beschreibt, aber zur Entscheidung zwischen NB1/2 zusätzlich auf die sowieso etablierten Informationskriteren (AIC/BIC) zurückgreift. Dazu gehört auch, dass das NB-P kein rein diagnostisches Werkzeug sein muss: Wenn ein NB-P ein *deutlich* kleineres AIC aufweist als NB1/2-Modelle, kann das NB-P auch als Modell der Wahl in Erwägung gezogen werden.

3.6. Zero-Inflated Models (*mixture models*)

Zero-inflated Modelle sind *mixtures*, das heißt, sie entstehen durch Mischung zweier Wahrscheinlichkeitsfunktionen. In der Regel wird eine Nullverteilung mit einer regulären Zählverteilung (Poisson, NB, PIG, ...) „gemischt", abhängig von einem *mixture parameter* $\pi \in [0, 1]$, der z.B. logistisch modelliert wird. Die Nullen im Modell werden dementsprechend durch beide Verteilungen (Nullverteilung, Zählverteilung) beschrieben – im Gegensatz zu *hurdle models* (vgl. B.3), die Nullen logistisch und die positiven counts über eine *truncated* distribution modellieren würden.

Zero-inflated distributions wie ZIP und ZINB wurden hergeleitet als zweiteilige mixture distributions. Die allgemeine Form für mixture distributions ist

Definition 3.9 (Mixture distribution und zero-inflated distribution)**.**

$$P(Y = y) = p \cdot g_1(y) + (1 - p) \cdot g_2(y)$$

mit p als *mixture proportion* und g_1, g_2 als Dichte-/Massefunktionen der beiden Komponenten.

Für zero-inflation setzt man p als *rate of zero-inflation* π, g_1 als degenerierten 0-Verteilung und g_2 als Poisson-PMF oder ähnliche Zähl-Verteilung wie NB, PIG, etc.

Eine alternative Darstellung für eine *zero-inflated* Verteilung (siehe[13]):

$$f_{\text{zeroinfl}}(y; x, \beta, \pi) = \pi \cdot I_{\{0\}}(y) + (1 - \pi) \cdot f_{\text{count}}(y; x, \beta)$$

Mit zu modellierendem Erwartungswert

$$\mu_i = \pi_i \cdot 0 + (1 - \pi_i) \cdot \exp(x_i^T \beta)$$

Wobei π in der Regel binomial, bzw. mittels logistischer Regression geschätzt wird.

Und im Speziellen, anhand der Beispiele ZIP und ZINB:

Definition 3.10 (Zero-Inflated Poisson Verteilung (ZIP))**.** Nach Perumean-Chaney et al. (2013) (p. 1675)

$$\begin{aligned} P(Y = 0) &= \pi + (1 - \pi) \cdot e^{-\mu} \\ P(Y = y) &= (1 - \pi) \cdot \frac{\mu^y e^{-\mu}}{y!}, \quad y = 1, 2, 3, ... \end{aligned}$$

[13] https://www.statistik.uni-dortmund.de/useR-2008/slides/Kleiber+Zeileis.pdf

Definition 3.11 (Zero-Inflated Negative Binomialverteilung (ZINB))**.** Nach Perumean-Chaney et al. (2013) (p. 1675)

$$P(Y = 0) = \pi + (1 - \pi) \cdot \frac{\theta^\theta}{(\theta + \mu)^\theta}$$
$$P(Y = y) = (1 - \pi) \cdot \frac{\Gamma(\theta + y)}{\Gamma(\theta)\Gamma(y + 1)} \frac{\theta^\theta \mu^y}{(\theta + \mu)^{(\theta+y)}}, \quad y = 1, 2, 3, ...$$

3.6.1. Interpretation

Ein Anwendungsbeispiel der ZIP findet sich auf der Seite der UCLA (SAS[14], R[15], zusätzlich gibt es einmal annotated Output (SAS)[16]), wobei der `fish`-Datensatz verwendet wird.

In diesem Datensatz beobachten wir Nullen mit zwei unterschiedlichen Ursprüngen: Zum Einen haben wir die Nullen der Personen, die geangelt haben, aber keine Fische fingen – und zum Anderen haben wir die Nullen der Personen, die *nicht angelten*, und daher garantiert auch keine Fische fingen („*certain zeros*“) (vgl. auch Abschnitt 4.2.2).

Ein ZIP-Modell erlaubt uns hier also beide Komponenten zu berücksichtigen – sowohl die Personen, die erfolglos angelten, als auch die Personen, die gar nicht angelten.

```
# ZP basierend auf fish-Daten, Modell aus UCLA-Tutorialseite
m_zip <- pscl::zeroinfl(count ~ child + camper | persons, data = fish)

summary(m_zip)
```

```
#>
#> Call:
#> pscl::zeroinfl(formula = count ~ child + camper | persons, data = fish)
#>
#> Pearson residuals:
#>     Min      1Q  Median      3Q     Max
#> -1.2369 -0.7540 -0.6080 -0.1921 24.0847
#>
#> Count model coefficients (poisson with log link):
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  1.59789    0.08554  18.680   <2e-16 ***
#> child       -1.04284    0.09999 -10.430   <2e-16 ***
#> camper1      0.83402    0.09363   8.908   <2e-16 ***
#>
```

[14] https://stats.idre.ucla.edu/sas/dae/zero-inflatedpoisson-regression/
[15] https://stats.idre.ucla.edu/r/dae/zip/
[16] https://stats.idre.ucla.edu/sas/output/zero-inflated-poisson-regression/

```
#> Zero-inflation model coefficients (binomial with logit link):
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)   1.2974     0.3739   3.470 0.000520 ***
#> persons      -0.5643     0.1630  -3.463 0.000534 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Number of iterations in BFGS optimization: 12
#> Log-likelihood: -1032 on 5 Df
```

```
# Exponentierte Koeffizienten
tibble::enframe(exp(coef(m_zip)), name = "coefficient")
```

```
#> # A tibble: 5 x 2
#>   coefficient       value
#>   <chr>             <dbl>
#> 1 count_(Intercept) 4.94
#> 2 count_child       0.352
#> 3 count_camper1     2.30
#> 4 zero_(Intercept)  3.66
#> 5 zero_persons      0.569
```

Wir erhalten zwei Sets an Koeffizienten:

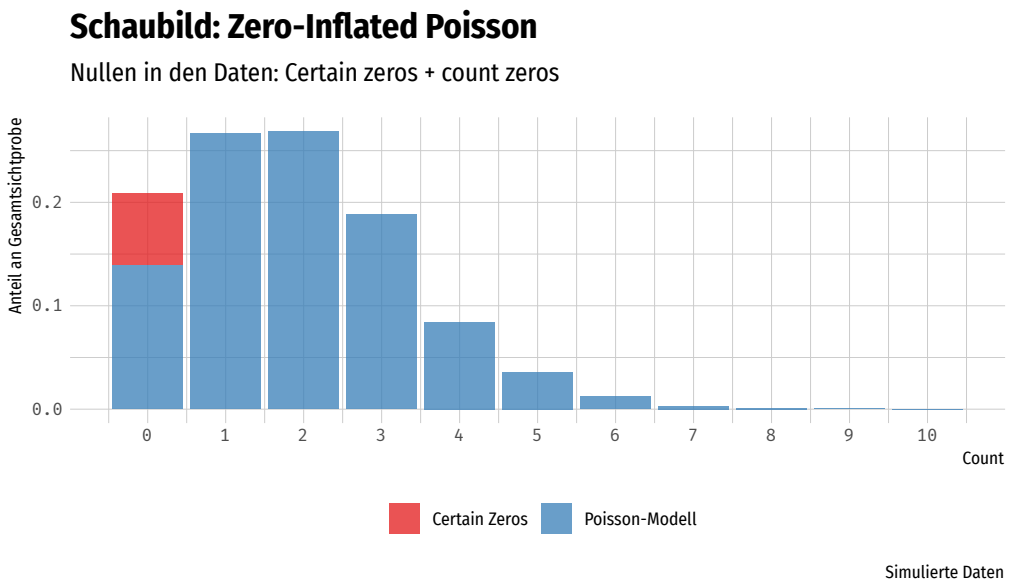
Count-model: Das count model (`count_`), i.e. Poisson. Hier modellieren wir die Anzahl der gefangen Fische ohne Berücksichtigung der *excess zeros*. Diese Koeffizienten werden analog einer herkömmlichen Poisson-Regression interpretiert.

Inflation-model: Die binomiale Komponente (`zeros_`) für die *excess zeros*, womit wir die Wahrscheinlichkeit von „certain zeros" modellieren. Diese Koeffizienten werden analog einer logistischen Regression interpretiert.
Aus dem UCLA DAE:

> **persons**: If a group were to increase its persons value by one, **the odds that it would be in the „Certain Zero" group** would decrease by a factor of exp(-0.5643) = 0.5687581. In other words, the more people in a group, the less likely the group is a certain zero.
>
> **Intercept**: If all of the predictor variables in the model are evaluated at zero, the odds of being a „Certain Zero" is exp(1.2974) = 3.659769. This means that the predicted odds of a group with zero persons is 3.659769 (though remember that evaluating persons at zero is out of the range of plausible values–every group must have at least one person).

```r
nsim <- 10^4
meansim <- 2

tibble::tibble(
  counts = rpois(n = nsim, lambda = meansim),
) %>%
  count(counts, name = "Poisson") %>%
  mutate(
    Zeros = floor(Poisson/2),
    Zeros = ifelse(counts > 0, 0, Zeros)
  ) %>%
  gather(source, freq, Poisson, Zeros) %>%
  mutate(source = forcats::fct_rev(source),
         freq = freq/nsim) %>%
  ggplot(aes(x = counts, y = freq, fill = source)) +
  geom_col(alpha = .75) +
  scale_x_continuous(breaks = seq(0, 12)) +
  scale_fill_brewer(palette = "Set1", labels = c(Zeros = "Certain Zeros", Poisson = "Poisson-Mode
  labs(
    title = "Schaubild: Zero-Inflated Poisson",
    subtitle = "Nullen in den Daten: Certain zeros + count zeros",
    x = "Count", y = "Anteil an Gesamtsichtprobe", fill = "",
    caption = "Simulierte Daten"
  ) +
  theme(legend.position = "bottom")
```



4. Anwendung

Als Beispiel verwenden wir hier `rwm1984`, ein Subset des Datensatz `rwm5yr` (vgl. Abschnitt 1.1, `?COUNT::rwm1984`), der Angaben zur Anzahl der Arztbesuche pro Person mit zusätzlichen demographischen Merkmalen enthält. Für unser Beispielmodell verwenden wir folgende Variablen:

- `docvis`: Abhängige Variable, Anzahl der Arztbesuche im Jahr (0-121)
- `outwork`: Arbeitslos (1), arbeitend (0)
- `age`: Alter (25 - 64)

```
# Daten
data(rwm1984, package = "COUNT")

# Model fit
mod_rwm <- glm(docvis ~ outwork + age, family = poisson(), data = rwm1984)

# Summary display
pander(summary(mod_rwm))
```

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-0.03352	0.03918	-0.8554	0.3923
outwork	0.4079	0.01884	21.65	6.481e-104
age	0.02208	0.0008377	26.36	3.73e-153

(Dispersion parameter for poisson family taken to be 1)

Null deviance:	25791 on 3873 degrees of freedom
Residual deviance:	24190 on 3871 degrees of freedom

Das erste was wir zur Evaluation unseres Modells tun können, ohne direkt andere Modelle zum vergleich heranzuziehen, ist die beobachteten Counts und die auf Basis des Modells erwarteten Counts zu vergleichen, um ein grobes Gefühl für die Situation zu erhalten (Code frei adaptiert nach Hilbe (2014), p. 88f):

```
# Beobachtete und erwartete counts
observed_v_expected <- rwm1984 %>%
  count(docvis, name = "observed") %>%
  mutate(
    expected = purrr::map_dbl(docvis, ~{
                  dpois(.x, fitted(mod_rwm)) %>%
                    sum() %>%
                    round(2)
                }),
    difference = observed - expected
  )

observed_v_expected %>%
  head(5) %>%
  pander(caption = "Observed and expected counts")
```

Tabelle 4: Observed and expected counts

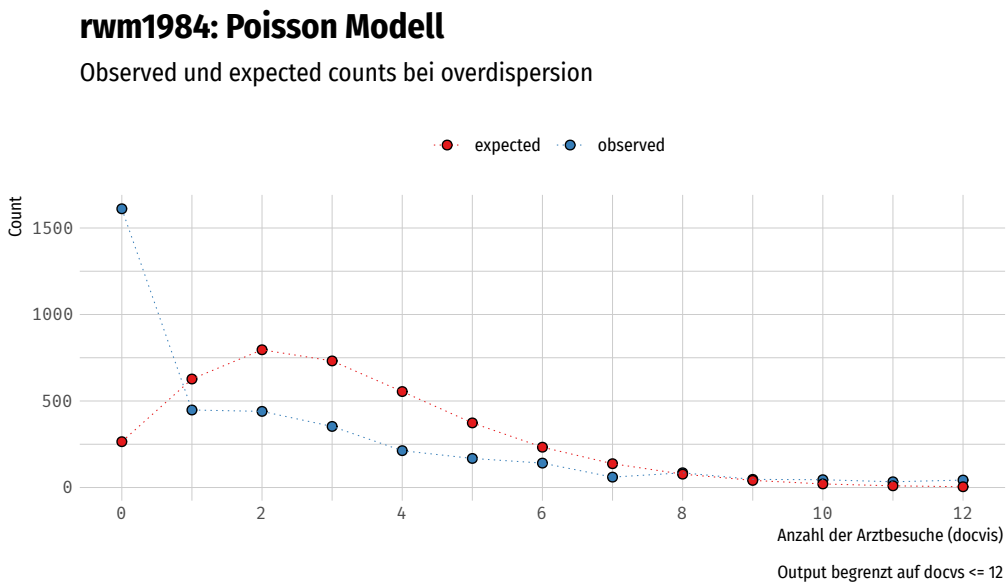
docvis	observed	expected	difference
0	1611	264.8	1346
1	448	627.1	-179.1
2	440	796	-356
3	353	731.6	-378.6
4	213	554.6	-341.6

```r
# Mittelwert und Varianz der jeweiligen counts
# (für "expected" gilt Varianz := Mittelwert)
tribble(
  ~Counts,   ~Mean,                ~Variance,
  "observed", mean(rwm1984$docvis), var(rwm1984$docvis),
  "expected", mean(fitted(mod)),    mean(fitted(mod))
) %>%
  pander(caption = "Mean & variance of observed and expected counts")
```

Tabelle 5: Mean & variance of observed and expected counts

Counts	Mean	Variance
observed	3.163	39.39
expected	5.401	5.401

```r
# Plot: observed vs. expected counts
observed_v_expected %>%
  filter(docvis <= 12) %>%
  gather(type, count, observed, expected) %>%
  ggplot(aes(x = docvis, y = count, fill = type, color = type)) +
  geom_point(shape = 21, color = "black", stroke = .5, size = 2) +
  geom_path(linetype = "dotted", size = .25) +
  scale_x_continuous(breaks = seq(0, 12, 2)) +
  scale_fill_brewer(palette = "Set1", aesthetics = c("color", "fill"), name = "") +
  labs(
    title = "rwm1984: Poisson Modell",
    subtitle = "Observed und expected counts bei overdispersion",
    caption = "Output begrenzt auf docvs <= 12",
    x = "Anzahl der Arztbesuche (docvis)", y = "Count"
  ) +
  theme(legend.position = "top")
```



Anhand der ersten Tabelle können wir recht schnell erkennen, dass wir hier *deutlich* mehr Nullen beobachten als das Modell vorhersagt – mehr dazu in Abschnitt 4.2.
Der Plot veranschaulicht den eher suboptimalen model fit unter diesen Umständen (overdispersion und (bzw. bedingt durch) zero-inflation).

4.1. Overdispersion: Erkennung und Handhabung

Es gibt mehrere Möglichkeiten Poisson-overdispersion zu erkennen, wobei die Pearson-Dispersionsstatistik in der Regel der erste Schritt ist.
Im Folgenden werden zusätzlich einige formale Tests aufgeführt.

4.1.1. Lagrange Mulitplier Test

Definition 4.1 (Lagrange Multiplier Test)**.** Nach Hilbe (2014) (p. 85f):

Eine χ^2-Teststatistik mit einem Freiheitsgrad.

$$\chi^2 = \frac{\left(\sum_{i=1}^{n} \hat{\mu}_i^2 - n\bar{y}\right)^2}{2\sum_{i=1}^{n} \hat{\mu}_i^2} \tag{7}$$

Eine rudimentäre R-Implementation findet sich in Abschnitt 1.2.2.

4.1.2. (Boundary) Likelihood Ratio Test (BLR)

Zum vergleich von zwei geschachtelten (*nested*) Modellen kann der *Likelihood Ratio Test* verwendet werden:

Definition 4.2 (Likelihood Ratio Test)**.**

$$LR = -2(\mathscr{L}_R - \mathscr{L}_F) \tag{8}$$

Mit $\mathscr{L}_F$ als log-likelihood des vollen (oder „größeren") Modells, und $\mathscr{L}_R$ als log-likelihood des reduzierteren Modells.

Eine Variante des Tests kann verwendet werden, um den Dispersionsparameter α eines NB-Modells zu testen. Da eine NB-Verteilung für $\alpha = 0$ äquivalent zur Poisson ist (siehe Abschnitt 3.1), kann ein Poisson-Modell als reduzierte Variante eines NB-Modells betrachtet werden. In diesem Fall verwendet man den *Boundary Likelhihood Ratio* test:

Definition 4.3 (Boundary Likelihood Ratio Test)**.**

$$BLR = -2(\mathscr{L}_{\text{Poisson}} - \mathscr{L}_{\text{NB}}) \tag{9}$$

> It is important, though, to remember that the BLR test has a lower limiting case for the value of α, which is what is being tested. Given that the standard parameterization of the negative binomial variance function is $\mu + \alpha\mu^2$, when $\alpha = 0$, the variance reduces to μ.
>
> — Hilbe (2014), p. 115

Der resultierende Wert ist $\chi^2_{(1)}$-verteilt. Der resultierende p-Wert muss zusätzlich halbiert werden (siehe Hilbe, 2014, p. 115).

> Since the distribution being tested can go no lower than 0, that is the boundary. Only one half of the full distribution is used. Therefore the Chi2 (sic!) test is divided by 2
>
> — Hilbe (2014), p. 115

Am Beispiel der `rwm1984`-Daten:

```
# Poisson-Modell
mod_p <- glm(docvis ~ outwork + age, family = poisson(), data = rwm1984)

# NB-Modell
mod_nb <- MASS::glm.nb(docvis ~ outwork + age, data = rwm1984)

# BLR: Recht eindeutig.
lmtest::lrtest(mod_p, mod_nb)
```

```
#> Likelihood ratio test
#>
#> Model 1: docvis ~ outwork + age
#> Model 2: docvis ~ outwork + age
#>   #Df   LogLik Df Chisq Pr(>Chisq)
#> 1   3 -15636.4
#> 2   4  -8332.8  1 14607  < 2.2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Bei gegebenem BLR-Wert von 4.2 ließe sich der p-Wert wie folgt berechnen:
pchisq(4.2, df = 1, lower.tail = FALSE) / 2
```

```
#> [1] 0.02021199
```

```
# Oder
(1 - pchisq(4.2, df = 1)) / 2
```

```
#> [1] 0.02021199
```

4.1.3. Umgang mit Overdispersion

Zur expliziten Modellierung des Dispersionparameters kann ein NBH Modell (3.5.1) genutzt werden, falls die Quelle der overdispersion von besonderem Interesse ist.

Abseits davon bleiben 2 grobe Strategien :

1. Adjustierung der durch die overdispersion verzerrten Standardfehler (Quasipoisson, Quasi-Likelihood, robuste Varianzschätzer)
2. Wechsel auf ein Modell, das overdispersion (oder allgemeine extradispersion) erlaubt (e.g. NB)

(Bemerke: Standardfehler für IRRs werden i.A. über die *delta method* bestimmt, die ich noch recherchieren und irgendwo einbauen muss)

4.1.3.0.1. Quasipoisson: Skalierung der Standardfehler

Hier werden lediglich die Standardfehler der Koeffizienten eines Poisson-Modells adjustiert, die bei overdispersion i.d.R. unterschätzt werden – die eigentliche Parameterschätzung wird nicht beeinflusst:

$$\text{SE}(\beta_k) \cdot \sqrt{D}$$

Wobei D der Pearson-Dispersionsindex ist (vgl. Hilbe (2014), p. 92ff), den wir in 2.5 als $\frac{\chi^2_{\text{Pearson}}}{\text{df}}$ definiert hatten.

Ein möglicher Nachteil jedoch: Es wird zuerst ein reguläres Poisson-Modell gefittet, Standardfehler und Dispersionsindex bestimmt, und dann dasselbe Modell mit skalierten Standardfehlern erneut gefittet – dementsprechend ist diese Methode vermutlich für größere Datensätze eher ineffizient.

Einmal auf unser voriges Beispiel anhand der `rwm5yr`-Daten angewandt:

```
# Urprüngliches Model
mod <- glm(docvis ~ outwork + age, family = poisson(), data = rwm1984)

mod_qp <- glm(docvis ~ outwork + age, family = quasipoisson(), data = rwm1984)

dispersion(mod)
```

```
#> X-squared(3871) = 43909.62
#> Pearson Dispersion = 11.343
```

```
dispersion(mod_qp)
```

```
#> X-squared(3871) = 43909.62
#> Pearson Dispersion = 11.343
```

An der Dispersion hat sich nichts geändert, nur an den Standardfehlern der Koeffizienten (in folgenden Tabellen auf log-Skala):

```
pander(mod, caption = "Poisson-Modell")
```

Tabelle 6: Poisson-Modell

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-0.03352	0.03918	-0.8554	0.3923
outwork	0.4079	0.01884	21.65	6.481e-104
age	0.02208	0.0008377	26.36	3.73e-153

```
pander(mod_qp, caption = "Quasipoisson-Modell")
```

Tabelle 7: Quasipoisson-Modell

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-0.03352	0.132	-0.254	0.7995
outwork	0.4079	0.06347	6.427	1.457e-10
age	0.02208	0.002821	7.827	6.386e-15

In diesem Fall „lohnt“ sich dieser Ansatz zwar eher nicht, da wir neben overdispersion noch ein Problem mit zero-inflation haben, aber in manchen Fällen kann Quasipoisson-Methode ausreichen.

> „Scaling is an easy, quick-and-dirty method of adjusting standard errors for overdispersion. However, when data are highly correlated or clustered, model slopes or coefficients usually need to be adjusted as well. Scaling does not accommodate that need, but simulations demonstrate that it is quite useful for models with little to moderate overdispersion.“
>
> — Hilbe (2014) (p. 96)

4.1.3.1. Robuste Varianzschätzer (Sandwich Estimators)

(Andere Namen: Huber & White standard errors, empirical standard errors)

Sandwich estimators sind in erster Linie für nicht unabhängige bzw. korrellierte Daten gedacht, zum Beispiel für Daten, die innerhalb mehrere Haushalte, Krankenhäuser, Städte etc. gesammelt wurden.

Robuste Varianzschätzer können (und je nach Quelle: *sollten*) standardmäßig für Count-response Daten verwendet werden, da die resultierenden Schätzer im Falle tatsächlich unkorrellierter Daten äquivalent zu den Standardfehlern des ursprünglichen Modells sind (i.e. „schadet nicht“).

Bootstrapped SEs wiederum erfordern mehr Aufwand, ähneln aber den robusten SEs sowieso sehr stark, weshalb sich der Aufwand ggf. nicht wirklich lohnt.

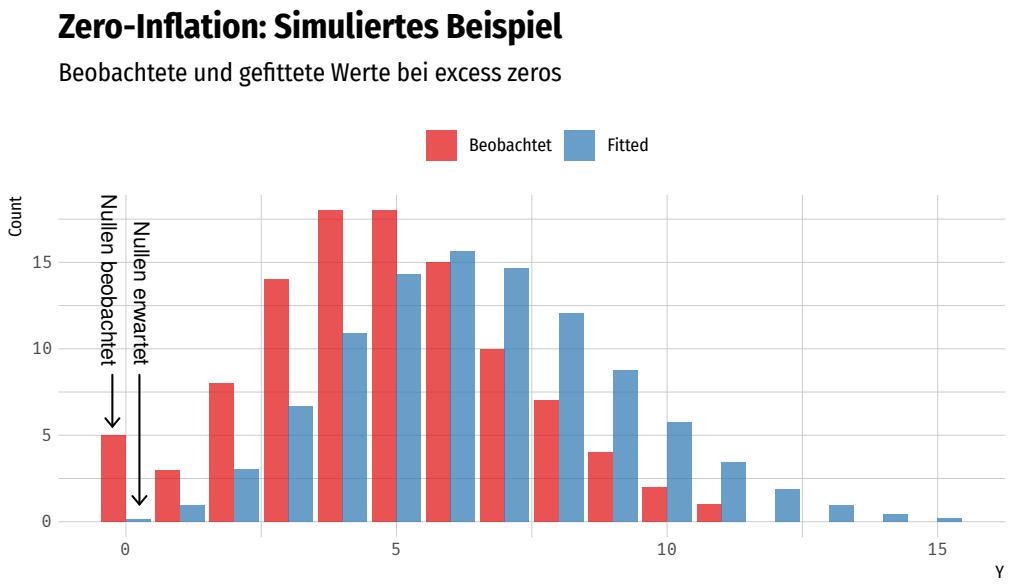
Siehe Hilbe (2014) (p. 99f) für eine ausführlichere Beschreibung. Zudem gibt der Autor explizit den Rat:

> „Unless your Poisson or negative binomial model is well fitted and meets its respective distributional assumptions, use robust or empirical standard errors as a default“
>
> — Hilbe (2014) (p. 133)

4.2. Zero-Inflation (ZI)

Problem: „Excess zeros“, i.e. das Modell sagt für $P(y_i = 0 \mid x_i)$ eine deutlich kleinere Wahrscheinlichkeit vor, als in gegebenen Daten tatsächlich vorliegen.

Zero-inflation hängt eng mit overdispersion zusammen. Sie *kann* Ursache für overdispersion sein, allerdings bedeutet das nicht direkt, dass auch ein entsprechend auf *ZI*-fokussiertes Modell (ZIP, ZINB...) verwendet werden *muss* – ggf. lässt sich die overdispersion bereits durch ein anderes geeignetes Modell „auffangen“. Zum Beispiel zeigt Winkelmann (2010), dass die *NB* allgemein eine größere Anzahl an Nullen erwartet als die Poisson.



4.2.1. Vergleich von Zero-Inflated models

Zum Vergleich zwischen Zählmodellen und ihren zero-inflated Gegenstücken (e.g. Poisson vs. ZIP oder NB vs. ZINB) wird in der Literatur häufig der Vuong-Test (Vuong, 1989; Desmarais and Harden, 2013) verwendet. Dieser Test ist explizit für den Vergleich von **un**geschachtelten Modellen entworfen (im Gegensatz zu z.B. dem Likelihood Ratio test). Wilson (2015) kritisiert dieses Vorgehen mit der Begründung (und Demonstration), dass zero-inflated Modelle *nicht* ungeschachtelt mit ihren ursprünglichen Zählmodellen sind. Auch Perumean-Chaney et al. (2013) und Hilbe (2014), die in diesem Dokument mehrfach zitiert werden, verwenden den Vuong-Test auf diese Weise.

Das Paper (Wilson (2015)) ist recht kurz und definitiv einen Blick Wert, insbesondere da ich (Lukas) mir bisher nicht zutraue zu beurteilen, wer in dieser Frage „Recht“ hat.

> The misuse of the test stems from misunderstanding of what is meant by the terms „non-nested model“ and „nested model“. As is the case with many frequently used terms their meanings are approximately understood by many, but precisely understood by few.
>
> — Wilson (2015), p. 52

4.2.2. Modellierung

Für die Modellierung von zero-inflated Daten stehen primär zwei Möglichkeiten zur Verfügung (vgl. auch Hilbe (2014), p. 19):

1. *Zero-inflated models*: Mixture models, die aus zwei Wahrscheinlichkeitsfunktionen eine neue Wahrscheinlichkeitsfunktion generieren, die sowohl Nullen als auch positive counts beschreibt, allerdings nun auch einen *zero-inflation*-Parameter hat (siehe Abschnitt 3.6).
2. *Hurdle models*: Getrennte Modellierung von Nullen und positiven counts in zwei separaten Modellen (siehe Abschnitt B.3).

Ob hurdle model oder zero-inflated model die bessere Wahl ist hängt mitunter davon ab, welche Annahmen über die *Ursache der Nullen* getroffen werden können.
Wenn es eine echte Trennung der Mechanismen (*„separation of mechanisms“*) gibt, die die Nullen und die positiven counts verursachen, dann wäre ein hurdle model eher angemessen. IDas würde allerdings auch annehmen, dass alle Nullen *die gleiche* Ursache haben.

Wenn sich die Nullen überlappen, es also keine getrennten Prozesse zu geben scheint, dann wären zero-inflated models angemessen (Hilbe, 2014, p. 209).

Als Veranschaulichung für zwei sich überlappende Mechanismen können wir das *fish*-Beispiel auf dieser UCLA-Tutorialseite verwenden[17]. Hier ist die Anzahl der gefangenen Fische an einem Wochenende in einem Park die Zählvariable.

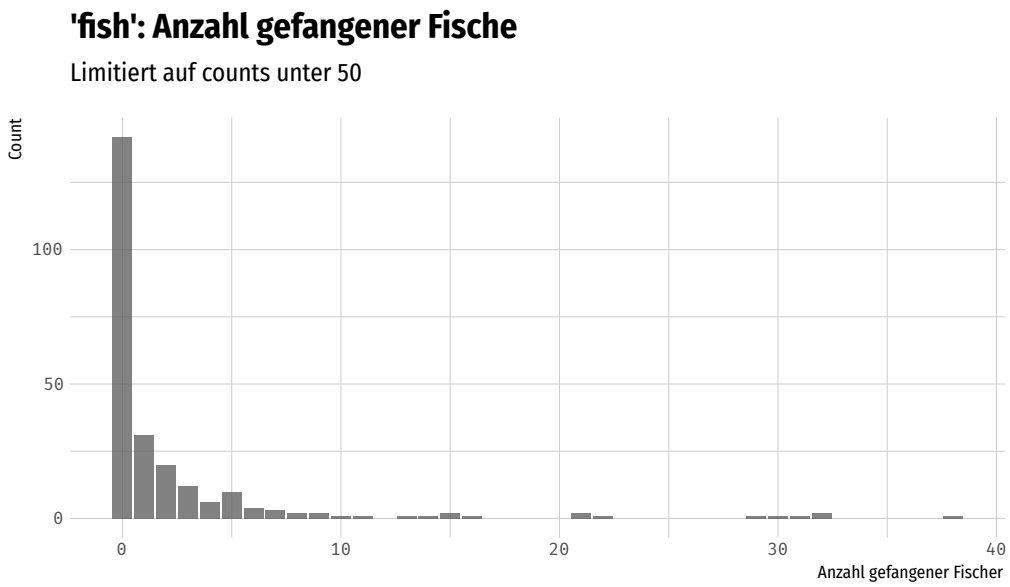
```
# Mittelwert & Varianz von 'count'
describe_counts(fish$count)
```

N	Missing	Mittelwert	Varianz	Range
250	0	3.3	135.37	[0, 149]

```
# Barchart für counts <= 50, Spannweite sehr groß
fish %>%
  filter(count <= 50) %>%
  ggplot(aes(x = count)) +
  geom_bar(alpha = .75) +
```

[17] https://stats.idre.ucla.edu/sas/output/zero-inflated-poisson-regression/

```
  scale_x_continuous() +
  labs(
    title = "'fish': Anzahl gefangener Fische",
    subtitle = "Limitiert auf counts unter 50",
    x = "Anzahl gefangener Fischer", y = "Count"
  )
```

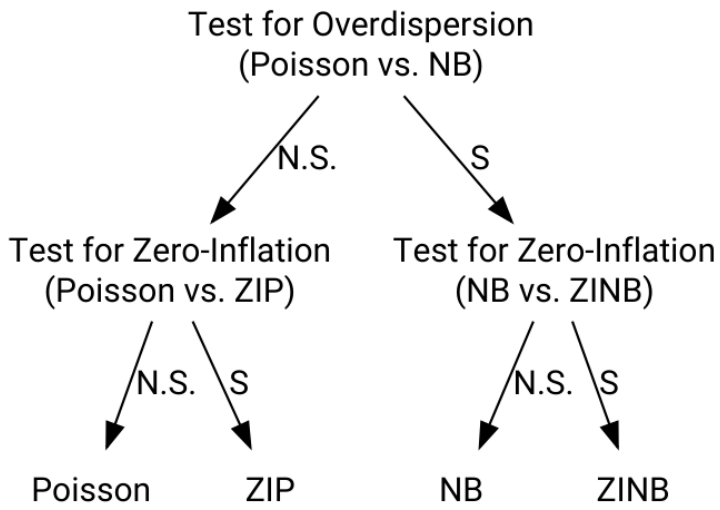


Die Anzahl der Nullen ist hier auffallend groß – allerdings beobachten wir hier auch zwei unterschiedliche Mechanismen, die sich überlappen: Eine Gruppe kann das ganze Wochenende geangelt haben, während eine andere Gruppe gar nicht geangelt hat – beide Gruppen werden jedoch am Ende nach der Anzahl ihrer gefangenen Fische befragt, weshalb wir in den Daten Nullen mit unterschiedlichem „Ursprung“ erhalten

Da es hier nicht angemessen wäre, die Nullen und die positiven counts getrennt zu Modellieren, würden wir hier ein *zero-inflated* Modell (z.B ZIP, ZINB, ...) verwenden.

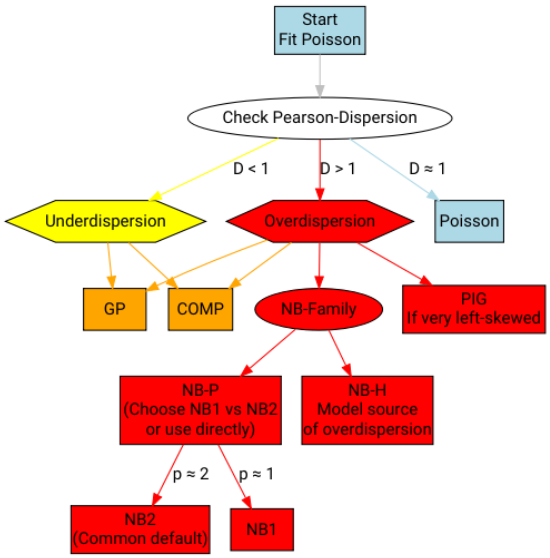
5. Entscheidungshilfen

Perumean-Chaney et al. (2013) schlagen folgendes Vorgehen vor um Overdispersion und Zero-Inflation zu erkennen:



Wobei im ersten Schritt sowohl Poisson als auch NB-Modelle gefittet werden, und dann nach Goodness-of-Fit (via LRT) (NB > Poisson?) entschieden wird.
Im zweiten Schritt werden die jeweiligen Modelle (entweder Poisson oder NB) mit ihren ZI-Counterparts verglichen (model fit, Vuong-Test), analog Schritt 1.
„LRT-Vuong Method“.

Eine allgemeinere Darstellung zur Veranschaulichung der sich ergebenden Möglichkeiten:



A. Verfügbarkeit

Sowohl der Quelltext als auch das Output (sprich, dieses Dokument in mehreren Formaten) sind öffentlich an mehreren Orten zu finden:

	Host	URL
Code	Gitea (self-hosted)	https://git.tadaa-data.de/lukas/poisson-regression
	GitHub	https://github.com/jemus42/poisson-regression
Result	GitHub Pages	https://jemus42.github.io/poisson-regression
	Self-Hosted (automated build)	https://poisson.tadaa-data.de/
	Self-Hosted (manual build)	https://lukas.tadaa-data.de/poisson/

Wobei automatische builds von Travis CI[18] übernommen werden, und das Resultat an die entsprechenden Zielorte gepusht wird. Die „manuelle“ Version auf `https://lukas.tadaa-data.de/poisson/` existiert primär zum schnellen Testen – die finalen Zielorte sollen GitHub Pages und (vermutlich) `https://poisson.tadaa-data.de` sein.

[18] https://travis-ci.org/jemus42/poisson-regression

B. Ergänzungen

Hier finden sich Erläuterungen zu Themen, die nur tangentiell in Verbindung mit dem eigentlichen Thema stehen, aber unter Umständen dennoch von Interesse sein könnten.

B.1. Missing values und missingness patterns

Zu den Begriffen im Kontext von missingness wird folgende Unterscheidung bezüglich der Verteilung der missing values für eine Variable Y gemacht (vgl. auch Little and Rubin (2002), p. 12):

- **MCAR** (missing completely at random): Das Fehlen eines Wertes in Y ist völlig unabhängig von sowohl beobachteten als auch unbeobachtet Daten. Dieser Fall wäre der idealzustand, ist allerdings in der Regel weder realistisch noch verifizierbar.
- **MAR** (missing at random): Es gibt einen Zusammenhang zwischen Ausfallwahrscheinlichkeit und *beobachteten* Daten. Das heißt auch, dass missingness durch beobachtete Daten erklärt werden kann, weshalb dieser Fall noch akzeptabel wäre.
- **MNAR** / **NMAR** (missing not at random): Weder *MCAR* noch *MAR* – die Ausfallwahrscheinlichkeit hängt von *unbeobachten* Daten ab. Das heißt, es gibt ein Ausfallmuster in Y, das nicht durch vorliegende Daten erklärt werden kann, wodurch in der Regel verzerrte Effektschätzungen entstehen.

B.2. Truncation und Censoring

Wenn die Beobachtungen keine Nullen enthalten bzw. im Modell aus inhaltlichen Gründen nicht möglich sind, können *zero-truncated* (ZT) Modelle verwendet werden. Diese Modelle sind insbesondere im Kontext von *hurdle models* interessant (siehe Abschnitt B.3), in denen die positiven counts getrennt von den Null-counts modelliert werden. Eine *truncated* distribution wäre die zero-truncated Poisson (ZTP).

Bemerke, dass *truncation* hier etwas anderes meint als *censoring*:

Truncation bedeutet, dass bestimmte counts (i.e. Merkmalsausprägungen in Y) *nicht möglich* sind. Als Beispiel könnte die Dauer eines Krankenhausaufenthalts dienen, da Aufenthaltstage erst ab einem Tag aufgezeichnet werden. (siehe auch Datensatz `azprocedure`, 1.1).
Unterschieden werden kann zwischen *left truncation* (z.B. keine counts kleiner als 5), *right truncation* (analog, keine counts größer als 5), oder *interval truncation* (nur counts im Intervall $[2, 10]$).

Censoring wiederum bedeutet, dass eine Merkmalsausprägung (wie etwa $Y = 0$ oder auch $Y = 1029$) zwar im Modell prinzipiell möglich ist, aber lediglich nicht in einer konkreten Stichprobe beobachtet wurde.

B.3. Hurdle models

Die nachfolgende Beschreibung dient daher eher der Vollständigkeit, da hurdle models in bestimmten Anwendungsgebieten scheinbar recht populär sind – allerdings ist es vermutlich eher schwierig sie auf binäre outcomes anzuwenden.

Im Allgemeinen kann man zwei Arten von hurdle models unterscheiden, die jeweils aus zwei Modellkomponenten bestehen:

- *Nested* hurdle models: Beide Komponenten nested (e.g. beide Poisson).
- *Non-nested* hurdle models: Hurdle-Komponente als vollständig anderer Prozess betrachtet und via e.g. logit modelliert.

Zwei gängige Komponenten für unnested hurdle models:

1. Binary 0,1 response, (logit oder probit)
    - Modellierung der Wahrscheinlichkeit für die non-zero counts
2. Zero-truncated count model

- Erlauben sowohl under- als auch overdispersion
- (Unnested models) erlauben systematischen Unterschied im Prozess, der zu e.g. Outcomes $= 0$ vs. Outcomes > 0 führt, was durch die Wahl unterschiedlicher Modelle für beide Komponenten abgebildet wird

In diesem Fall entspricht das Resultat eines hurdle models zwei separat gefitteten Modellen (e.g. Pois + Logit), die getrennt interpretierbar sind (im Gegensatz zu zero-inflated models!).

Definition B.1 (Hurdle Model)**.** Nach Winkelmann (2010), p. 179f:

Sei $g_1(0)$ die Wahrscheinlichkeit des Outcomes 0 und $g_1(k), k = 1, 2, 3, ...$ die Wahrscheinlichkeitsfunktion für natürliche Zahlen, dann ist die Wahrscheinlichkeitsfunktion eines *hurdle-at-zero* Modells:

$$
\begin{aligned}
f(y = 0) &= g_1(0) \\
f(y = k) &= (1 - g_1(0))g_2(k), \quad k = 1, 2, 3, ...
\end{aligned}
$$

Bzw. nach Mullahy (1986) mit f_1 und f_2 als PMFs für natürliche Zahlen

$$
\begin{aligned}
f(y = 0) &= f_1(0) \\
f(y = 1) &= \frac{1 - f_1(0)}{1 - f_2(0)} f_2(k) \\
&= \Theta f_2(k), \quad k = 1, 2, 3, ...
\end{aligned}
$$

Wobei

- f_2 als *parent process* bezeichnet wird
- $1 - f_1(0)$ die Wahrscheinlichkeit angibt, die Hürde $(y = 0)$ zu „überqueren“ (*„crossing the hurdle“*).
- $1 - f_2(0)$ zur Normalisierung von f_2 dient, um deren truncation zu berücksichtigen.

Der Erwartungswert des hurdle models ist

$$\mathbb{E}_h(y) = \Theta \sum_{k=1}^{\infty} k f_2(k) = \Theta \mathbb{E}_2(y)$$

Mit $\mathbb{E}_2$ als Erwartunsgwert von f_2.

Mit f_2 = Poisson:

- $0 < \Theta < 1$: Overdispersion
- $1 < \Theta < \frac{\lambda_2 + 1}{\lambda_2}$: Underdispersion

  „By far the most popular hurdle model in practice is the hurdle-at-zero negative bonomial model“ (Winkelmann, 2010, p. 183)

mit $f_1 \sim NB(\beta_1, \alpha_1)$ und $f_2 \sim NB(\beta_2, \alpha_2)$

C. Reproduzierbarkeit

C.1. R-Code

Hier verwendeter Code profitiert stark vom `tidyverse`[19].
Insbesondere wird anstelle der Funktion `data.frame` in der Regel `tibble` (package `tibble`, automatisch re-exportiert von `dplyr`) verwendet. Diese Alternative bietet einige quality of life improvements für schnelle Simulationen, zum Beispiel die Verwendung von Variablen während diese noch definiert werden:

```
# Nicht möglich:
data.frame(
   x1 = rnorm(10),
   x2 = rnorm(10, mean = 5),
   y = 50 + 3 * x1 + 5 * x2
)

# Funktioniert!
tibble(
```

[19] https://www.tidyverse.org/

```
   x1 = rnorm(10),
   x2 = rnorm(10, mean = 5),
   y = 50 + 3 * x1 + 5 * x2
)
```

Zusätzlich kann die Pipe (`%>%`) Verwendung finden, ein function composition operator.
Es gilt:

`f(g(h(x), b = 4), a = 1) = h(x) %>% g(b = 4) %>% f(a = 1)`

```
# Klassisch
x <- rnorm(100, mean = 15)
x_mean <- mean(x)
sqrt(x_mean)

# oder
sqrt(mean(rnorm(100, mean = 15)))

# piped
rnorm(100, mean = 15) %>%
   mean() %>%
   sqrt()

# Klassisch
iris_subset <- subset(iris, Species == "setosa")
head(iris_subset[order(iris_subset$Sepal.Length, decreasing = TRUE), ], n = 5)

# tidyverse-style (incl. filter() als subset()-Analog und top_n() für order() + head())
iris %>%
   filter(Species == "setosa") %>%
   top_n(Sepal.Length, n = 5)
```

Letztlich stellt das Package `broom`[20] eine wichtige Ergänzung dar. Mitunter ist `augment()`[21] eine komfortable Möglichkeit um schnell in tabellarischer Form gefittete Werte mit ihren dazugehörigen x-Werten zu erhalten.

[20] https://broom.tidyverse.org/
[21] https://broom.tidyverse.org/reference/augment.lm.html#examples

C.2. Session Info

Umgebungsvariable	Wert
version	R version 3.6.2 (2017-01-27)
os	Ubuntu 16.04.6 LTS
system	x86_64, linux-gnu
ui	X11
language	en_US.UTF-8
collate	en_US.UTF-8
ctype	en_US.UTF-8
tz	UTC
date	2020-03-06

Package	Version	Quelle
broom	0.5.5	CRAN (R 3.6.2)
DiagrammeR	1.0.5	CRAN (R 3.6.2)
dplyr	0.8.4	CRAN (R 3.6.2)
firasans	0.1.0	Github (hrbrmstr/firasans@52f83c8)
gamlss	5.1-6	CRAN (R 3.6.2)
gamlss.data	5.1-4	CRAN (R 3.6.2)
gamlss.dist	5.1-6	CRAN (R 3.6.2)
ggplot2	3.3.0	CRAN (R 3.6.2)
hrbrthemes	0.8.0	gitlab (hrbrmstr/hrbrthemes@a599f17)
kableExtra	1.1.0	CRAN (R 3.6.2)
MASS	7.3-51.5	CRAN (R 3.6.2)
nlme	3.1-145	CRAN (R 3.6.2)
pander	0.6.3	CRAN (R 3.6.2)
purrr	0.3.3	CRAN (R 3.6.2)
tidyr	1.0.2	CRAN (R 3.6.2)
VGAM	1.1-2	CRAN (R 3.6.2)

D. Unsorted

Hier liegen temporär Opfer der Umstrukturierung, bis sie ein passendes zu Hause gefunden haben, oder auf die farm upstate umziehen.

D.1. General Advice

- Starte mit einem Poisson-Modell und baue darauf auf

- Benutze robuste Varianzschätzer (e.g. sandwich, Bootstrap-SEs sind meist den Aufwand nicht wert) – entweder sie helfen, oder sie schaden nicht.
- Das gängigste Problem ist overdispersion, aber nicht jede overdispersion ist gleich.
- Die erwartete Anzahl an Nullen (unter Poisson) ist $\exp(-\bar{x}) \cdot n$

D.2. Beispiel nach Hilbe (2014 p. 211ff)

(Eigentlich kompliziertes Beispiel, weil 0 counts nicht möglich sind, müsste man truncated drangehen)

Grundlage ist der Datensatz `azprocedure` (siehe Abschnitt 1.1) zur Dauer des Krankenhausaufenthalts.

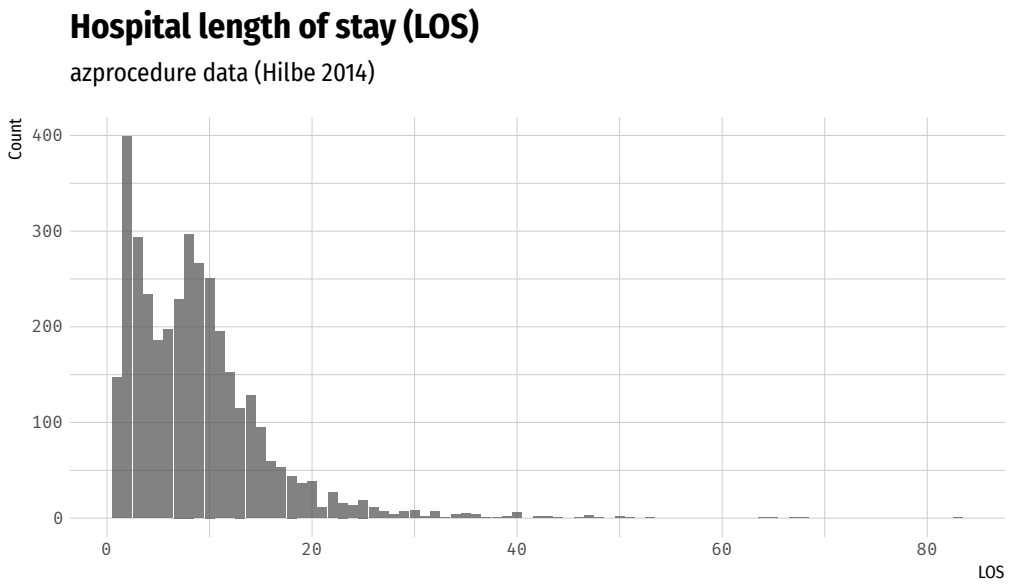
Zuerst werfen wir einen Blick auf die Daten und fitten ein reguläres Poissonmodell.

```
data(azprocedure, package = "COUNT")

describe_counts(azprocedure$los)
```

N	Missing	Mittelwert	Varianz	Range
3589	0	8.83	47.97	[1, 83]

```
# Barchart: So grob poissonverteilt?
ggplot(data = azprocedure, aes(x = los)) +
  geom_bar(alpha = .75) +
  labs(
    title = "Hospital length of stay (LOS)",
    subtitle = "azprocedure data (Hilbe 2014)",
    x = "LOS", y = "Count"
  )
```




```
# Model fit
model_azproc <- glm(los ~ procedure + sex + admit,
                    data = azprocedure, family = poisson())

# Model output (ohne exponentierte Koeffizienten)
pander(model_azproc)
```

Tabelle 8: Fitting generalized (poisson/log) linear model: los ~ procedure + sex + admit

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	1.491	0.01539	96.91	0
procedure	0.9574	0.01218	78.61	0
sex	-0.1302	0.01179	-11.04	2.408e-28
admit	0.3331	0.0121	27.52	1.11e-166

```
# Pearson Dispersion:
dispersion(model_azproc)
```

```
#> X-squared(3585) = 11588.08
#> Pearson Dispersion = 3.232
```

Der Dispersionsindex lässt auf overdispersion schließen.

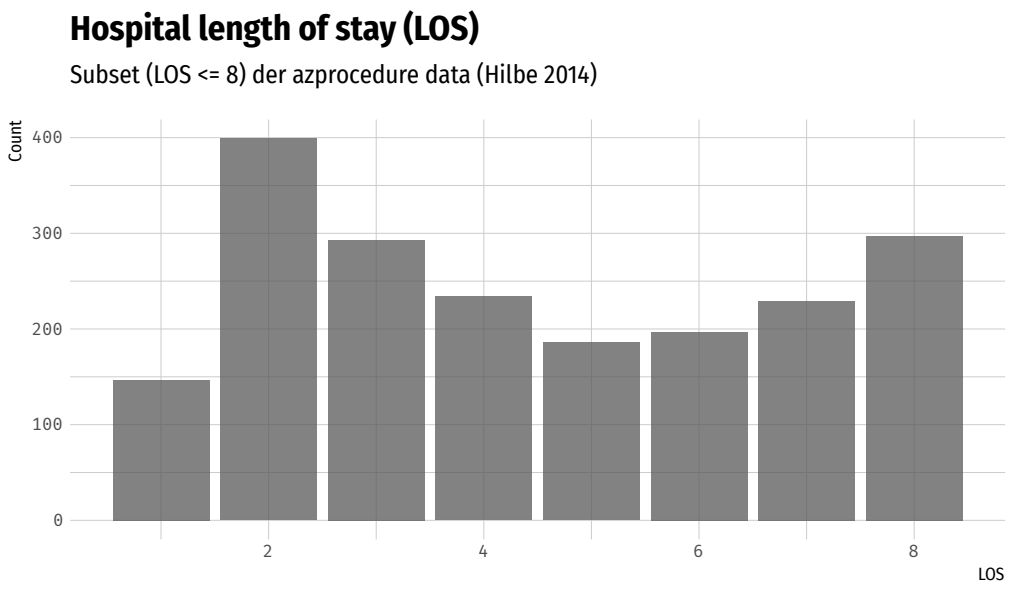
Betrachten wir ein Subset der Daten, indem wir nur Beobachtungen mit `LOS` $\leqslant 8$ betrachten, erhalten wir ein anderes Bild:

```
azprocedure_subset <- subset(azprocedure, los <= 8)

describe_counts(azprocedure_subset$los)
```

N	Missing	Mittelwert	Varianz	Range
1982	0	4.47	5.34	[1, 8]

```
# Barchart
ggplot(data = azprocedure_subset, aes(x = los)) +
  geom_bar(alpha = .75) +
  labs(
    title = "Hospital length of stay (LOS)",
    subtitle = "Subset (LOS <= 8) der azprocedure data (Hilbe 2014)",
    x = "LOS", y = "Count"
  )
```



```
model_azproc_u <- glm(los ~ procedure + sex + admit,
                      data = azprocedure_subset, family = poisson())

pander(model_azproc_u)
```

Tabelle 9: Fitting generalized (poisson/log) linear model: los ~ procedure + sex + admit

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	1.187	0.02498	47.52	0
procedure	0.731	0.02398	30.48	4.848e-204
sex	-0.06892	0.02294	-3.004	0.00266
admit	0.3097	0.02236	13.85	1.286e-43

```
dispersion(model_azproc_u)
```

```
#> X-squared(1978) = 1562.75
#> Pearson Dispersion = 0.790
```

In diesem Fall haben wir es mit underdispersion zu tun, also versuchen wir es mal mit der GP:

```
library(VGAM)
library(gamlss)

mod_gp_vgam <- vglm(los ~ procedure + sex + admit, data = azprocedure_subset, family = genpoisson

mod_gp_gamlss <- gamlss(los ~ procedure + sex + admit, data = azprocedure_subset, family = GPO())
```

```
#> GAMLSS-RS iteration 1: Global Deviance = 7969.342
#> GAMLSS-RS iteration 2: Global Deviance = 7961.567
#> GAMLSS-RS iteration 3: Global Deviance = 7961.567
```

```
summary(mod_gp_vgam)
```

```
#>
#> Call:
#> vglm(formula = los ~ procedure + sex + admit, family = genpoisson(),
#>     data = azprocedure_subset)
#>
#> Pearson residuals:
```

```
#>                         Min      1Q   Median     3Q   Max
#> rhobitlink(lambda) -0.6573 -0.5912 -0.28788 0.2773 6.782
#> loglink(theta)     -2.3158 -0.6304  0.07304 0.8734 1.627
#>
#> Coefficients:
#>               Estimate Std. Error z value Pr(>|z|)
#> (Intercept):1 -0.24021    0.03523  -6.819 9.16e-12 ***
#> (Intercept):2  1.30125    0.02766  47.042  < 2e-16 ***
#> procedure      0.71846    0.02143  33.521  < 2e-16 ***
#> sex           -0.06760    0.02043  -3.308 0.000939 ***
#> admit          0.31193    0.01993  15.652  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Names of linear predictors: rhobitlink(lambda), loglink(theta)
#>
#> Log-likelihood: -3956.379 on 3959 degrees of freedom
#>
#> Number of Fisher scoring iterations: 6
#>
#> No Hauck-Donner effect found in any of the estimates
```

```
summary(mod_gp_gamlss)
```

```
#> ******************************************************************
#> Family:  c("GPO", "Generalised Poisson")
#>
#> Call:  gamlss(formula = los ~ procedure + sex + admit, family = GPO(),
#>     data = azprocedure_subset)
#>
#> Fitting method: RS()
#>
#> ------------------------------------------------------------------
#> Mu link function:  log
#> Mu Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  1.18687    0.02980  39.824   <2e-16 ***
#> procedure    0.73075    0.03959  18.456   <2e-16 ***
#> sex         -0.06892    0.02643  -2.607   0.0092 **
#> admit        0.30974    0.02750  11.264   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
```

```
#> ------------------------------------------------------------------
#> Sigma link function:  log
#> Sigma Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   -36.04    2246.20  -0.016    0.987
#>
#> ------------------------------------------------------------------
#> No. of observations in the fit:  1982
#> Degrees of Freedom for the fit:  5
#>       Residual Deg. of Freedom:  1977
#>                       at cycle:  3
#>
#> Global Deviance:     7961.567
#>             AIC:     7971.567
#>             SBC:     7999.527
#> ******************************************************************
```

Beispiel aus Hilbe 2014. Eigentlich sollte $\delta \approx -0.1195$ und $\theta = \frac{1}{(1-\delta)^2} \approx 0.799$

Literatur

Desmarais, B. A. and Harden, J. J. (2013). Testing for Zero Inflation in Count Models: Bias Correction for the Vuong Test. *The Stata Journal*, 13(4):810–835.

Fahrmeir, L., Kneib, T., and Lang, S. (2009). *Regression: Modelle, Methoden und Anwendungen*. Statistik und ihre Anwendungen. Springer, Berlin, 2. aufl edition. OCLC: 463790744.

Harris, T., Yang, Z., and Hardin, J. W. (2012). Modeling Underdispersed Count Data with Generalized Poisson Regression. *The Stata Journal*, 12(4):736–747.

High, R. (2018). Alternative Variance Parameterizations in Count Data Models with the NLMIXED Procedure.

Hilbe, J. M. (2014). *Modeling Count Data*. Cambridge University Press, Cambridge.

Little, R. J. A. and Rubin, D. B. (2002). *Statistical Analysis with Missing Data*. Wiley Series in Probability and Statistics. Wiley, Hoboken, N.J, 2nd ed edition.

Lord, D., Geedipally, S. R., and Guikema, S. D. (2010). Extension of the Application of Conway-Maxwell-Poisson Models: Analyzing Traffic Crash Data Exhibiting Underdispersion. *Risk Analysis*, 30(8):1268–1276.

Lord, D., Guikema, S. D., and Geedipally, S. R. (2008). Application of the Conway-Maxwell-Poisson generalized linear model for analyzing motor vehicle crashes. *Accident Analysis & Prevention*, 40(3):1123–1134.

Mullahy, J. (1986). Specification and testing of some modified count data models. *Journal of Econometrics*, 33(3):341–365.

Perumean-Chaney, S. E., Morgan, C., McDowall, D., and Aban, I. (2013). Zero-inflated and overdispersed: What's one to do? *Journal of Statistical Computation and Simulation*, 83(9):1671–1683.

Sellers, K. F. and Shmueli, G. (2010). A flexible regression model for count data. *The Annals of Applied Statistics*, 4(2):943–961.

Shmueli, G., Minka, T. P., Kadane, J. B., Borle, S., and Boatwright, P. (2005). A useful distribution for fitting discrete data: Revival of the Conway-Maxwell-Poisson distribution. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 54(1):127–142.

Stasinopoulos, D. M. and Rigby, R. A. (2007). Generalized Additive Models for Location Scale and Shape (GAMLSS) in R. *Journal of Statistical Software*, 23(1):1–46.

Vuong, Q. H. (1989). Likelihood Ratio Tests for Model Selection and Non-Nested Hypotheses. *Econometrica*, 57(2):307.

Wilson, P. (2015). The misuse of the Vuong test for non-nested models to test for zero-inflation. *Economics Letters*, 127:51–53.

Winkelmann, R. (2010). *Econometric Analysis of Count Data*. Springer Berlin, Berlin, 5th ed. edition.