Kapitel 8

Regressionsmodelle für kategoriale Daten und Zähldaten

Das Modell der linearen Regression und Varianzanalyse (vgl. Abschn. 6.3, 7.3, 12.9.1) lässt sich zum verallgemeinerten linearen Modell (GLM, *generalized linear model*) erweitern, das auch für Daten einer kategorialen vorherzusagenden Variable Y geeignet ist.[1] Als Prädiktoren lassen sich sowohl kontinuierliche Variablen als auch Gruppierungsfaktoren einsetzen. Ein Spezialfall ist die logistische Regression für dichotome Y (codiert als 0 und 1). Im Vergleich zur Vorhersage quantitativer Variablen in der linearen Regression wird an diesem Beispiel zunächst folgende Schwierigkeit deutlich (für Details vgl. Agresti, 2007 sowie Fox & Weisberg, 2011):

Eine lineare Funktion von p Prädiktoren X_j der Form $\beta_0 + \beta_1 X_1 + \cdots + \beta_j X_j + \cdots + \beta_p X_p$ kann bei einem unbeschränkten Definitionsbereich beliebige Werte im Intervall $(-\infty, +\infty)$ annehmen. Es können sich durch die Modellgleichung also Werte ergeben, die weder von Y selbst, noch vom Erwartungswert $E(Y)$ angenommen werden können. Dabei ist $E(Y)$ für dichotome Y die Wahrscheinlichkeit eines Treffers $P(Y = 1)$, die im Intervall $[0, 1]$ liegt.

Dagegen ist die lineare Modellierung von $\text{logit}(P) = \ln \frac{P}{1-P}$ möglich, dem natürlichen Logarithmus des Wettquotienten, der Werte im Intervall $(-\infty, +\infty)$ annimmt. Setzt man $\ln \frac{P}{1-P} = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$ als gültiges Modell voraus und fasst die rechte Seite der Gleichung als *linearen Prädiktor* $\boldsymbol{X\beta}$ zusammen (vgl. Abschn. 12.9.1), so ist P mit der logistischen Funktion als Umkehrfunktion der Logit-Funktion als $P = \frac{\mathrm{e}^{\boldsymbol{X\beta}}}{1+\mathrm{e}^{\boldsymbol{X\beta}}} = \frac{1}{1+\mathrm{e}^{-\boldsymbol{X\beta}}}$ identifizierbar. Solche Transformationen des eigentlich vorherzusagenden Parameters, die eine lineare Modellierung ermöglichen, heißen *Link-Funktion* $g(\cdot)$. Sie müssen u. a. eine Umkehrfunktion $g^{-1}(\cdot)$ besitzen, so dass $g(E(Y)) = \boldsymbol{X\beta}$ und $E(Y) = g^{-1}(\boldsymbol{X\beta})$ gilt.

Verkürzt gesprochen wird im GLM anders als im allgemeinen linearen Modell nur $E(Y)$ über die Link-Funktion linear modelliert, nicht Y selbst. Es ist deshalb notwendig, die angenommene Form der bedingten Verteilung von Y für gegebene Prädiktorwerte explizit anzugeben (vgl. Abschn. 12.9). Mit dieser Form liegt auch die Varianz der Verteilung in Abhängigkeit von $E(Y)$ fest. Die bedingte Verteilung muss aus der natürlichen Exponentialfamilie stammen und wird mit einer Link-Funktion kombiniert, die $E(Y)$ mit dem linearen Prädiktor $\boldsymbol{X\beta}$ in Beziehung setzt. Sind mehrere Link-Funktionen mit einer bedingten Verteilungsform kombinierbar, ist eine davon die *kanonische* Link-Funktion mit besonderen statistischen Eigenschaften.

Die lineare Regression ist ein Spezialfall des GLM, bei dem von bedingter Normalverteilung von Y ausgegangen wird (vgl. Abschn. 12.9.4, Abb. 12.5) und $E(Y)$ zudem nicht durch eine Link-Funktion transformiert werden muss, um linear modellierbar zu sein. Anders als in der

[1] Abschnitt 6.6.4 gibt Hinweise auf gemischte Regressionsmodelle und verallgemeinerte Schätzgleichungen (GEE) für abhängige Daten – etwa durch Messwiederholung oder Clusterung, die analog auf kategoriale Zielgrößen übertragen werden können.

linearen Regression werden die Parameter β_j im GLM über die Maximum-Likelihood-Methode geschätzt (vgl. Abschn. 8.1.7). In den folgenden Abschnitten sollen die logistische, ordinale, multinomiale und Poisson-Regression ebenso vorgestellt werden wie log-lineare Modelle. Für die Visualisierung kategorialer Daten vgl. Abschn. 14.4 sowie das Paket `vcd` (Meyer, Zeileis & Hornik, 2014).

8.1 Logistische Regression

In der logistischen Regression für dichotome Daten wird als bedingte Verteilung von Y die Binomialverteilung angenommen, die kanonische Link-Funktion ist die Logit-Funktion. Die bedingten Verteilungen sind dann durch $E(Y) = P$ vollständig festgelegt, da ihre Varianz gleich $n_i\, P\,(1 - P)$ ist. Dabei ist n_i die Anzahl der dichotomen Messwerte für dieselbe Kombination von Prädiktorwerten.

8.1.1 Modell für dichotome Daten anpassen

Die Anpassung einer logistischen Regression geschieht mit der `glm()` Funktion, mit der allgemein GLM-Modelle spezifiziert werden können.[2] Hier sei zunächst der Fall betrachtet, dass die erhobenen Daten als dichotome Variable vorliegen.

```
> glm(formula=⟨Modellformel⟩, family=⟨Verteilungsfamilie⟩, data=⟨Datensatz⟩)
```

Unter `formula` wird eine Modellformel ⟨`AV`⟩ `~` ⟨`Prädiktoren`⟩ wie mit `lm()` formuliert, wobei sowohl quantitative wie kategoriale Variablen als Prädiktoren möglich sind. Die AV muss ein Objekt der Klasse `factor` mit zwei Stufen sein, die Auftretenswahrscheinlichkeit P bezieht sich auf die zweite Faktorstufe. Weiter ist unter `family` ein Objekt anzugeben, das die für die AV angenommene bedingte Verteilung sowie die Link-Funktion benennt (vgl. `?family`). Im Fall der logistischen Regression ist dieses Argument auf `binomial(link="logit")` zu setzen.

Als Beispiel sei jenes aus der Kovarianzanalyse herangezogen (vgl. Abschn. 7.8), wobei hier vorhergesagt werden soll, ob die Depressivität nach der Behandlung über dem Median liegt. Die Beziehung zwischen der Depressivität vor und nach der Behandlung in den drei Gruppen soll dabei zunächst über ein Diagramm mit dem nonparametrisch geschätzten Verlauf der Trefferwahrscheinlichkeit veranschaulicht werden, das `cdplot(`⟨`Modellformel`⟩`)` erzeugt (*conditional density plot*, Abb. 8.1).

```
# Median-Split zur Umwandlung der quantitativen AV in dichotomen Faktor
> dfAncova <- transform(dfAncova, postFac=cut(DVpost,
+                       breaks=c(-Inf, median(DVpost), Inf),
+                       labels=c("lo", "hi")))

# conditional density plot der AV in den drei Gruppen
> cdplot(postFac ~ DVpre, data=dfAncova, subset=IV == "SSRI",
+        main="Geschätzte Kategorien-Wkt. SSRI")
```

[2] Für die bedingte logistische Regression bei Stratifizierung der Beobachtungen vgl. `clogit()` aus dem Paket `survival` (Therneau, 2014).

```
> cdplot(postFac ~ DVpre, data=dfAncova, subset=IV == "Placebo",
+        main="Geschätzte Kategorien-Wkt. Placebo")

> cdplot(postFac ~ DVpre, data=dfAncova, subset=IV == "WL",
+        main="Geschätzte Kategorien-Wkt. WL")
```

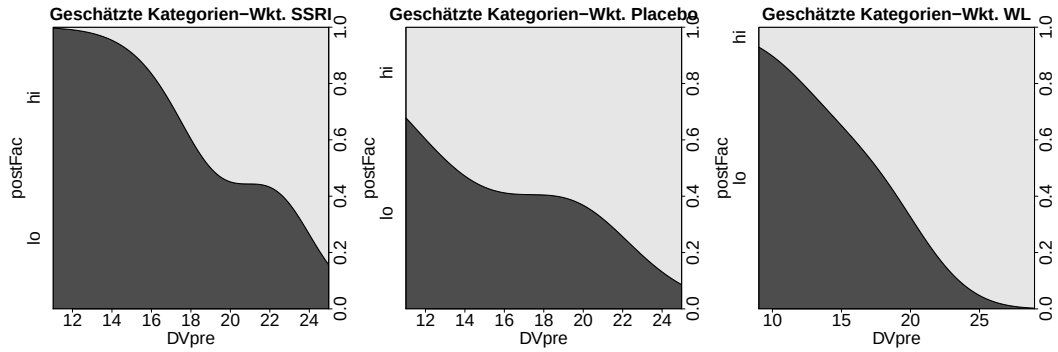

Abbildung 8.1: Nonparametrisch geschätzte Kategorienwahrscheinlichkeiten in Abhängigkeit vom Prädiktor in drei Behandlungsgruppen

```
# Anpassung des logistischen Regressionsmodells
> (glmFit <- glm(postFac ~ DVpre + IV, family=binomial(link="logit"),
+                data=dfAncova))
Call:  glm(formula = postFac ~ DVpre + IV,
           family = binomial(link = "logit"), data = dfAncova)

Coefficients:
(Intercept)   DVpre  IVPlacebo    IVWL
    -8.4230  0.4258     1.7306  1.2027

Degrees of Freedom: 29 Total (i.e. Null);  26 Residual
Null Deviance:      41.46
Residual Deviance: 24.41    AIC: 32.41
```

Die Ausgabe nennt unter der Überschrift `Coefficients` zunächst die Schätzungen b_j der Modellparameter β_j der logistischen Regression, wobei der in der Spalte `(Intercept)` aufgeführte Wert die Schätzung b_0 ist. Der Parameter eines Prädiktors ist als Ausmaß der Änderung der Vorhersage $\ln \frac{\hat{P}}{1-\hat{P}}$ zu interpretieren, wenn der Prädiktor X_j um eine Einheit wächst, also als Differenz der logits (vgl. Abschn. 7.8.1 zur Bedeutung der zwei mit dem Faktor `IV` assoziierten Parameter).

Einfacher ist die Bedeutung eines exponenzierten Parameters e^{b_j} zu erfassen: Dieser Koeffizient gibt an, um welchen Faktor der vorhergesagte Wettquotient $\frac{\hat{P}}{1-\hat{P}}$ zunimmt, wenn sich X_j um eine Einheit vergrößert.[3] Dies ist das Verhältnis des vorhergesagten Wettquotienten nach der Änderung um eine Einheit zum Wettquotienten vor dieser Änderung, also ihr odds ratio (vgl.

[3] Für einen Prädiktor X: $\left(\frac{\hat{P}}{1-\hat{P}}\right)_{X+1} = \mathrm{e}^{b_0+b_1(X+1)} = \mathrm{e}^{b_0}\,\mathrm{e}^{b_1(X+1)} = \mathrm{e}^{b_0}\,\mathrm{e}^{b_1 X}\,\mathrm{e}^{b_1} = \mathrm{e}^{b_1}\,\mathrm{e}^{b_0+b_1 X} = \mathrm{e}^{b_1}\left(\frac{\hat{P}}{1-\hat{P}}\right)_X$.

Abschn. 10.2.6). Dagegen besitzt e^{b_0} keine intuitive Bedeutung. Wie bei linearen Modellen extrahiert `coef(⟨glm-Modell⟩)` die Parameterschätzungen.

```
> exp(coef(glmFit))                    # exponenzierte Koeffizienten
 (Intercept)         DVpre     IVPlacebo          IVWL
0.0002197532  1.5308001795  5.6440022784  3.3291484767
```

Wie bei der linearen Regression lassen sich die Konfidenzintervalle für die wahren Parameter mit `confint(⟨glm-Modell⟩)` berechnen.[4] Für die Konfidenzintervalle der odds ratios e^{β_j} sind die Intervallgrenzen zu exponenzieren.

```
> exp(confint(glmFit))                 # zugehörige Konfidenzintervalle
                    2.5 %      97.5 %
(Intercept)  1.488482e-07   0.0251596
DVpre        1.193766e+00   2.2446549
IVPlacebo    5.343091e-01  95.1942030
IVWL         2.916673e-01  52.2883653
```

8.1.2 Modell für binomiale Daten anpassen

Logistische Regressionen können mit `glm()` auch dann angepasst werden, wenn pro Kombination i von Prädiktorwerten mehrere dichotome Werte erhoben und bereits zu Ausprägungen einer binomialverteilten Variable aufsummiert wurden ($n_i \geq 1$). In diesem Fall ist das Kriterium in Form einer Matrix an die Modellformel zu übergeben: Jede Zeile der Matrix steht für eine Kombination von Prädiktorwerten i, für die n_i dichotome Werte vorhanden sind. Die erste Spalte der Matrix nennt die Anzahl der Treffer, die zweite Spalte die Anzahl der Nicht-Treffer. Beide Spalten summieren sich pro Zeile also zu n_i.

```
> N      <- 100                            # Anzahl Kombinationen
> x1     <- rnorm(N, 100, 15)              # Prädiktor 1
> x2     <- rnorm(N, 10, 3)                # Prädiktor 2
> total  <- sample(40:60, N, replace=TRUE) # ni
> hits   <- rbinom(N, total, prob=0.4)     # Treffer pro Kombination
> hitMat <- cbind(hits, total-hits)        # Matrix Treffer, Nicht-Treffer

# logistische Regression für absolute Häufigkeiten anpassen
> glm(hitMat ~ x1 + x2, family=binomial(link="logit"))        # ...
```

Liegt die AV als Vektor relativer Häufigkeiten eines Treffers vor, sind zusätzlich die n_i als Vektor an das Argument `weights` von `glm()` zu übergeben.

```
> relHits <- hits/total                    # relative Häufigkeiten ...
> glm(relHits ~ x1 + x2, weights=total, family=binomial(link="logit"))
```

[4] Die so ermittelten Konfidenzintervalle basieren auf der Profile-Likelihood-Methode und sind asymmetrisch. Demgegenüber berechnet `confint.default(⟨glm-Modell⟩)` symmetrische Wald-Konfidenzintervalle, die asymptotische Normalverteilung der Parameterschätzungen voraussetzen.

8.1.3 Anpassungsgüte

Als Maß für die Güte der Modellpassung wird von `glm()` die Residual-Devianz D als Summe der quadrierten Devianz-Residuen ausgegeben.[5] Mit $\hat{L}$ als geschätzter likelihood des Modells, die an der likelihood eines Modells mit perfekter Vorhersage normalisiert wurde, gilt $D = -2 \ln \hat{L}$. Durch die Maximum-Likelihood-Schätzung der Parameter wird $\hat{L}$ maximiert, D also minimiert – analog zur Fehlerquadratsumme in der linearen Regression (vgl. Abschn. 6.2).[6] Weiter erhält man den Wert des Informationskriteriums AIC $= D + 2(p + 1)$, bei dem ebenfalls kleinere Werte für eine bessere Anpassung sprechen (vgl. Abschn. 6.3.3). Dabei ist $p + 1$ die Anzahl zu schätzender Parameter (p Gewichte β_j sowie β_0).[7]

Die Residual-Devianz eines Modells ermittelt `deviance(`⟨`glm-Modell`⟩`)`, die logarithmierte geschätzte likelihood eines Modells `logLik(`⟨`glm-Modell`⟩`)` und den AIC-Wert `AIC(`⟨`glm-Modell`⟩`)`.

```
> (Dev <- deviance(glmFit))                    # Devianz
[1] 24.40857

> sum(residuals(glmFit)^2)                     # Devianz
[1] 24.40857

> as.vector(-2 * logLik(glmFit))               # Devianz
[1] 24.40857

> all.equal(AIC(glmFit), Dev + 2*(3+1))        # AIC
[1] TRUE
```

Weitere Maße der Anpassungsgüte sind pseudo-R^2-Koeffizienten, die an den Determinationskoeffizienten in der linearen Regression angelehnt sind (vgl. Abschn. 6.2.2).[8] Die Varianten nach McFadden, Cox & Snell (Maddala) und Nagelkerke (Cragg-Uhler) können auf Basis der Ausgabe von `glm()` manuell ermittelt werden.[9] Hier soll $\hat{L}_0$ die geschätzte likelihood des 0-Modells ohne Prädiktoren X_j mit nur dem Parameter β_0 sein. Analog sei $\hat{L}_f$ die geschätzte likelihood des Modells mit allen berücksichtigten Prädiktoren. Ferner bezeichne D_0 bzw. D_f die jeweils zugehörige Devianz.

- $R^2_{\text{McFadden}} = 1 - \frac{\ln \hat{L}_f}{\ln \hat{L}_0} = 1 - \frac{D_f}{D_0}$

[5] In der Voreinstellung gibt `residuals(`⟨`glm-Modell`⟩`, type="`⟨`Typ`⟩`")` Devianz-Residuen aus. Für andere Residuen-Varianten kann das Argument `type` verwendet werden (vgl. `?residuals.glm`).

[6] Für die gewöhnliche lineare Regression stimmen Devianz und Fehlerquadratsumme überein.

[7] Bei der gewöhnlichen linearen Regression wie auch bei der logistischen Regression mit der quasi-binomial Familie (s. u.) ist zusätzlich ein Varianzparameter zu schätzen. Hier beträgt die Anzahl also $p + 1 + 1$.

[8] Anders als in der linearen Regression lassen sich die pseudo-R^2-Maße jedoch nicht als Verhältnis von Variabilitäten verstehen. Ihre Vergleichbarkeit über verschiedene Datensätze hinweg ist zudem eingeschränkt – so beziehen etwa $R^2_{\text{Cox \& Snell}}$ sowie $R^2_{\text{Nagelkerke}}$ neben der absoluten Anpassung auch die Stichprobengröße ein.

[9] Für weitere Gütemaße der Modellanpassung vgl. die Funktion `lrm()` aus dem Paket `rms`, die neben Nagelkerkes pseudo-R^2 die Fläche unter der ROC-Kurve (vgl. Abschn. 10.2.7) ebenso bestimmt wie etwa Somers' d, Goodman und Kruskals γ sowie Kendalls τ für die vorhergesagten Wahrscheinlichkeiten und beobachteten Werte (vgl. Abschn. 10.3.1).

- $R^2_{\text{Cox \& Snell}} = 1 - \left(\frac{\hat{L}_0}{\hat{L}_f}\right)^{\frac{2}{N}} = 1 - e^{(\ln \hat{L}_0 - \ln \hat{L}_f)\,\frac{2}{N}}$

  Das Maximum von $R^2_{\text{Cox \& Snell}}$ beträgt $1 - \hat{L}_0^{\frac{2}{N}} < 1$.

- $R^2_{\text{Nagelkerke}} = R^2_{\text{Cox \& Snell}} / (1 - \hat{L}_0^{\frac{2}{N}})$

Wie bei linearen Modellen lässt sich ein bereits angepasstes Modell mit `update(⟨glm-Modell⟩)` ändern (vgl. Abschn. 6.3.2), etwa alle Prädiktoren bis auf den absoluten Term entfernen.

```
> glm0 <- update(glmFit, . ~ 1)                    # 0-Modell
> LL0  <- logLik(glm0)        # gesch. log-likelihood 0-Modell
> LLf  <- logLik(glmFit)      # gesch. log-likelihood vollständiges Modell
> as.vector(1 - (LLf / LL0))                       # R^2 McFadden
[1] 0.411209

> N <- nobs(glmFit)           # Anzahl der Beobachtungen
> as.vector(1 - exp((2/N) * (LL0 - LLf)))          # R^2 Cox & Snell
[1] 0.4334714

# R^2 Nagelkerke
> as.vector((1 - exp((2/N) * (LL0 - LLf))) / (1 - exp(LL0)^(2/N)))
[1] 0.578822
```

Eine Alternative zu pseudo-R^2-Koeffizienten ist der Diskriminationsindex von Tjur. Er berechnet sich als Differenz der jeweils mittleren vorhergesagten Trefferwahrscheinlichkeit für die Beobachtungen mit Treffer und Nicht-Treffer. Die vorhergesagte Wahrscheinlichkeit $\hat{P}$ erhält man etwa mit `fitted(⟨glm-Modell⟩)` (vgl. Abschn. 8.1.4). Der Diskriminationsindex nimmt maximal den Wert 1 an, ist jedoch anders als die pseudo-R^2-Koeffizienten nicht auf andere Regressionsmodelle für diskrete Kriterien verallgemeinerbar.

```
> Phat <- fitted(glmFit)    # vorhergesagte Trefferwahrscheinlichkeiten

# mittlere vorhergesagte Wahrscheinlichkeit für wFac = lo und wFac = hi
> (PhatLoHi <- aggregate(Phat ~ postFac, FUN=mean, data=dfAncova))
  postFac      Phat
1      lo 0.2521042
2      hi 0.7118809

# Tjur Diskriminationsindex
> abs(diff(PhatLoHi$Phat))  # Differenz mittlere vorherges. Wkt.
[1] 0.4597767
```

Für die Kreuzvalidierung verallgemeinerter linearer Modellen vgl. Abschn. 13.2. Methoden zur Diagnose von Ausreißern in den Prädiktoren können aus der linearen Regression ebenso übernommen werden wie Cooks Distanz und der Index DfBETAS zur Identifikation einflussreicher Beobachtungen (vgl. Abschn. 6.5.1). Anders als im Modell der linearen Regression hängt im GLM die Varianz vom Erwartungswert ab. Daher sollte eine Residuen-Diagnostik mit

spread-level oder scale-location plots (vgl. Abschn. 6.5.2) standardisierte Devianz- oder Pearson-Residuen verwenden, wie es mit `glm.diag(⟨glm-Modell⟩)` aus dem Paket `boot` Canty & Ripley, 2014 möglich ist. Für Methoden zur Einschätzung von Multikollinearität der Prädiktoren vgl. Abschn. 6.5.3.

8.1.4 Vorhersage, Klassifikation und Anwendung auf neue Daten

Die vorhergesagte Wahrscheinlichkeit $\hat{P} = \frac{1}{1+e^{-\boldsymbol{X}\hat{\boldsymbol{\beta}}}}$ berechnet sich für jede Beobachtung durch Einsetzen der Parameterschätzungen b_j in $\boldsymbol{X}\hat{\boldsymbol{\beta}} = b_0 + b_1 X_1 + \cdots + b_p X_p$. Man erhält sie mit `fitted(⟨glm-Modell⟩)` oder mit `predict(⟨glm-Modell⟩, type="response")`. In der Voreinstellung `type="link"` werden die vorhergesagten Logit-Werte ausgegeben.

```
> Phat <- fitted(glmFit)                # vorhergesagte Wahrscheinlichkeit
> predict(glmFit, type="response")      # äquivalent ...
> logitHat <- predict(glmFit, type="link")          # vorhergesagte Logits
> all.equal(logitHat, log(Phat / (1-Phat)))         # Kontrolle: log-odds
[1] TRUE
```

Mit der gewählten Herangehensweise ist die mittlere vorhergesagte Wahrscheinlichkeit gleich der empirischen relativen Häufigkeit eines Treffers.[10]

```
> mean(Phat)                            # mittlere vorhergesagte W'keit
[1] 0.4666667

# relative Trefferhäufigkeit
> prop.table(xtabs(~ postFac, data=dfAncova))
postFac
       lo        hi
0.5333333 0.4666667
```

Die vorhergesagte Trefferwahrscheinlichkeit soll hier als Grundlage für eine dichotome Klassifikation mit der Schwelle $\hat{P} = 0.5$ verwendet werden. Diese Klassifikation kann mit den tatsächlichen Kategorien in einer Konfusionsmatrix verglichen und etwa die Rate der korrekten Klassifikation berechnet werden.[11]

```
# Klassifikation auf Basis der Vorhersage
> thresh <- 0.5                         # Schwelle bei P=0.5
> facHat <- cut(Phat, breaks=c(-Inf, thresh, Inf), labels=c("lo", "hi"))

# Kontingenztafel: tatsächliche vs. vorhergesagte Kategorie
> cTab <- xtabs(~ postFac + facHat, data=dfAncova)
> addmargins(cTab)
```

[10] Dies ist der Fall, wenn die kanonische Link-Funktion und Maximum-Likelihood-Schätzungen der Parameter gewählt werden und das Modell einen absoluten Term β_0 beinhaltet.

[11] Vergleiche Abschn. 13.2 für die Kreuzvalidierung zur Abschätzung der Vorhersagegüte in neuen Stichproben sowie Abschn. 10.2.6, 10.2.7, 10.3.3 für weitere Möglichkeiten, Klassifikationen zu analysieren. Vergleiche Abschn. 12.8 für die Diskriminanzanalyse sowie die dortige Fußnote 44 für Hinweise zu weiteren Klassifikationsverfahren.

```
       facHat
postFac lo hi Sum
    lo  12  4  16
    hi   4 10  14
    Sum 16 14  30
```

```
> (CCR <- sum(diag(cTab)) / sum(cTab))  # Rate der korrekten Klass.
[1] 0.7333333
```

Inwieweit die Vorhersage der logistischen Regression zutrifft, kann auf unterschiedliche Weise grafisch veranschaulicht werden. Eine Möglichkeit stellt die vorhergesagten logits $\ln \frac{\hat{P}}{1-\hat{P}}$ dar und hebt die tatsächlichen Treffer farblich hervor (Abb. 8.2). Vorhergesagte logits > 0 sind bei einer Schwelle von $\hat{P} = 0.5$ äquivalent zur Vorhersage eines Treffers, so dass die tatsächlichen Treffer hier bei einer guten Klassifikation oberhalb einer Referenzlinie bei 0 liegen sollten.

```
> plot(logitHat, pch=c(1, 16)[unclass(regDf$wFac)], cex=1.5, lwd=2,
+      main="(Fehl-) Klassifikation durch Vorhersage",
+      ylab="vorhergesagte logits")

> abline(h=0)                              # Referenzlinie
> legend(x="bottomright", legend=c("lo", "hi"), pch=c(1, 16), cex=1.5,
+        lty=NA, lwd=2, bg="white")
```

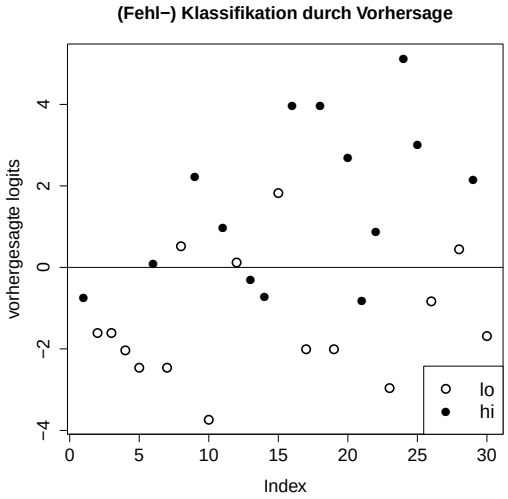

Abbildung 8.2: Vorhersage und Daten einer logistischen Regression. Daten, die auf Basis der Vorhersage falsch klassifiziert würden, sind farblich hervorgehoben

An das Argument `newdata` von `predict()` kann zusätzlich ein Datensatz übergeben werden, der neue Daten für Variablen mit denselben Namen, und bei Faktoren zusätzlich denselben Stufen wie jene der ursprünglichen Prädiktoren enthält. Als Ergebnis erhält man die vorhergesagten Trefferwahrscheinlichkeiten für die neuen Prädiktorwerte (vgl. Abschn. 6.4).

```
> Nnew  <- 3                               # Anzahl neuer Daten
> dfNew <- data.frame(DVpre=rnorm(Nnew, 20, sd=7),
```

```
+                         IV=factor(rep("SSRI", Nnew),
                                   levels=levels(dfAncova$IV)))

> predict(glmFit, newdata=dfNew, type="response")     # Vorhersage
          1           2           3
0.947324911 0.008185519 0.123296946
```

8.1.5 Signifikanztests für Parameter und Modell

Die geschätzten Gewichte b lassen sich mit `summary(⟨glm-Modell⟩)` einzeln einem Wald-Signifikanztest unterziehen. In der Ausgabe finden sich dazu unter der Überschrift `Coefficients` die Gewichte b in der Spalte `Estimate`, deren geschätzte Streuungen $\hat{\sigma}_b$ in der Spalte `Std. Error` und die zugehörigen z-Werte $\frac{b}{\hat{\sigma}_b}$ in der Spalte `z value`. Der Test setzt voraus, dass z unter der Nullhypothese asymptotisch standardnormalverteilt ist. Die zugehörigen p-Werte stehen in der Spalte `Pr(>|z|)`.

```
> summary(glmFit)
Call:
glm(formula = postFac ~ DVpre + IV, family = binomial(link = "logit"),
    data = dfAncova)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9865  -0.5629  -0.2372   0.4660   1.5455

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -8.4230     2.9502  -2.855   0.0043 **
DVpre         0.4258     0.1553   2.742   0.0061 **
IVPlacebo     1.7306     1.2733   1.359   0.1741
IVWL          1.2027     1.2735   0.944   0.3450

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 41.455  on 29  degrees of freedom
Residual deviance: 24.409  on 26  degrees of freedom
AIC: 32.409

Number of Fisher Scoring iterations: 5
```

Geeigneter als Wald-Tests sind oft Likelihood-Quotienten-Tests der Parameter, die auf der asymptotisch χ^2-verteilten Devianz-Differenz zweier nested Modelle mit demselben Kriterium beruhen:[12] Der Prädiktorensatz des eingeschränkten Modells ist dabei vollständig im Prädiktorensatz

[12] Bei Wald-Tests kann etwa das *Hauck-Donner-Phänomen* auftreten: Bei starken Effekten (sehr große β_j) sind die berechneten Streuungen $\hat{\sigma}_b$ dann deutlich zu groß, wodurch Wald-Tests der Parameter fälschlicherweise nicht signifikant werden.

des umfassenderen Modells enthalten, das zusätzlich noch weitere Prädiktoren berücksichtigt (vgl. Abschn. 6.3.3). Solche Modellvergleiche können mit `anova(⟨fitR⟩, ⟨fitU⟩, test="Chisq")` durchgeführt werden, wobei ⟨`fitR`⟩ das eingeschränkte und ⟨`fitU`⟩ das umfassendere Modell ist. Zusätzlich lässt sich wie in der linearen Regression `drop1(⟨glm-Modell⟩, test="Chisq")` verwenden (vgl. Abschn. 6.3.3).

Um das Gesamtmodell mit einem Likelihood-Quotienten-Test auf Signifikanz zu prüfen, muss somit `anova(⟨0-Modell⟩, ⟨glm-Modell⟩, test="Chisq")` aufgerufen werden (vgl. Abschn. 6.3.3). Der Test beruht auf dem Vergleich des angepassten Modells mit dem 0-Modell, das nur eine Konstante als Prädiktor beinhaltet. Teststatistik ist die Devianz-Differenz beider Modelle mit der Differenz ihrer Freiheitsgrade als Anzahl der Freiheitsgrade der asymptotisch gültigen χ^2-Verteilung.

```
> glm0 <- update(glmFit, . ~ 1)                 # 0-Modell
> anova(glm0, glmFit, test="Chisq")
Analysis of Deviance Table

Model 1: postFac ~ 1
Model 2: postFac ~ DVpre + IV
  Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
1        29     41.455
2        26     24.409  3   17.047 0.0006912 ***

# manuelle Kontrolle
> chisqStat <- glmFit$null.deviance - deviance(glmFit)
> chisqDf   <- glmFit$df.null       - df.residual(glmFit)
> (pVal     <- pchisq(chisqStat, chisqDf, lower.tail=FALSE))
[1] 0.0006912397
```

Da der hier im Modell berücksichtigte Faktor `IV` mit mehreren Parametern β_j assoziiert ist, muss seine Signifikanz insgesamt über einen Modellvergleich gegen das vollständige Modell getestet werden.

```
# eingeschränktes Modell ohne Faktor IV
> glmPre <- update(glmFit, . ~ . - IV)
> anova(glmPre, glmFit, test="Chisq")           # Modellvergleich
Analysis of Deviance Table

Model 1: postFac ~ DVpre
Model 2: postFac ~ DVpre + IV
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1        28     26.566
2        26     24.409  2   2.1572   0.3401
```

Analog erfolgt für Quadratsummen vom Typ I (vgl. Abschn. 7.5.2) der Test des kontinuierlichen Prädiktors `DVpre` als Vergleich des Modells nur mit `DVpre` mit dem 0-Modell.

```
> anova(glm0, glmPre, test="Chisq")             # Modellvergleich
Analysis of Deviance Table
```

```
Model 1: postFac ~ 1
Model 2: postFac ~ DVpre
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1        29     41.455
2        28     26.566  1    14.89 0.000114 ***
```

8.1.6 Andere Link-Funktionen

Eine meist zu ähnlichen Ergebnissen führende Alternative zur logistischen Regression ist die Probit-Regression. Sie verwendet $\Phi^{-1}(\cdot)$, die Umkehrfunktion der Verteilungsfunktion $\Phi(\cdot)$ der Standardnormalverteilung als Link-Funktion, wofür das Argument `family` von `glm()` auf `binomial(link="probit")` zu setzen ist. Hier ist also $P = \Phi(\boldsymbol{X}\boldsymbol{\beta})$. Eine weitere Link-Funktion, die mit bedingter Binomialverteilung von Y kombiniert werden kann, ist die komplementäre log-log-Funktion $\ln(-\ln(1-P))$. Man erhält sie mit `binomial(link="cloglog")`. Mit ihrer Umkehrfunktion, der Verteilungsfunktion der Gumbel-Verteilung, gilt $P = 1 - \exp(-\exp(\boldsymbol{X}\boldsymbol{\beta}))$ – dabei steht $\exp(\cdot)$ aus typografischen Gründen für $\mathrm{e}^{(\cdot)}$.

Mitunter streuen empirische Residuen stärker, als dies bei bedingter Binomialverteilung mit der Varianz $n_i\,P\,(1-P)$ zu erwarten wäre (*overdispersion*). Ein Hinweis auf overdispersion ist ein Verhältnis von Residual-Devianz zu Residual-Freiheitsgraden, das deutlich von 1 abweicht. Für diesen Fall kann ein Binomial-ähnliches Modell verwendet werden, das einen zusätzlichen, aus den Daten zu schätzenden Streuungsparameter ϕ besitzt, mit dem für die bedingte Varianz $\sigma^2 = \phi\,n_i\,P\,(1-P)$ gilt: Hierfür ist `family` auf `quasibinomial(link="logit")` zu setzen. Die Parameterschätzungen b sind dann identisch zur logistischen Regression, die geschätzten Streuungen $\hat{\sigma}_b$ jedoch unterschiedlich, was zu anderen Ergebnissen der inferenzstatistischen Tests führt.

Da die bedingte Verteilung der Daten in der quasi-binomial Familie bis auf Erwartungswert und Varianz unspezifiziert bleibt, ist die Likelihood-Funktion der Daten für gegebene Parameter unbekannt. Trotzdem kann jedoch die IWLS-Methode Parameter schätzen (vgl. Abschn. 8.1.7), wobei dann alle likelihood-basierten Kennwerte wie AIC oder pseudo-R^2 nicht zur Verfügung stehen. Durch die Schätzung des Streuungs-Parameters sind die Wald-Teststatistiken keine z-, sondern t-Werte. Analog sollten Modellvergleich mit einem F-Test über das Argument `anova(..., test="F")` durchgeführt werden statt über einen χ^2-Test wie bei bekannter Varianz.

Den Spezialfall einer linearen Regression erhält man mit `gaussian(link="identity")`. Die Maximum-Likelihood-Schätzungen der Parameter stimmen dann mit den Schätzern der linearen Regression überein.

8.1.7 Mögliche Probleme bei der Modellanpassung

Die Maximum-Likelihood-Schätzung der Parameter ist nicht in geschlossener Form darstellbar, sondern muss mit einem numerischen Optimierungsverfahren gefunden werden – typischerweise über die IWLS-Methode (*iterative weighted least squares*). Diese numerische Suche nach dem

Maximum der Likelihood-Funktion kann in seltenen Fällen fehlschlagen, auch wenn es ein eindeutiges Maximum gibt. Ursache einer nicht konvergierenden Suche ist häufig, dass sie an einem Punkt von Schätzungen beginnt, der zu weit vom tatsächlichen Maximum entfernt ist.

Konvergenz-Probleme können durch Warnmeldungen angezeigt werden (etwa *algorithm did not converge*), sich in sehr großen Schätzungen bei gleichzeitig sehr großen Standardfehlern äußern oder in einer Residual-Devianz, die größer als die Devianz des Null-Modells ist. In diesen Fällen ist das Ergebnis von `glm()` nicht gültig. Ob Konvergenz erreicht wurde, speichert die von `glm()` zurückgegebene Liste in der Komponente `converged`.

Über das Argument `start` lassen sich manuell Start-Werte für alle Parameter vorgeben, um die Konvergenz der Suche zu begünstigen. Eine Strategie für die Wahl guter Start-Werte besteht darin, eine gewöhnliche lineare Regression der logit-transformierten Variable Y durchzuführen und deren Parameter-Schätzungen zu wählen. Konvergenzprobleme lassen sich u. U. auch über eine höhere maximale Anzahl von Suchschritten beheben, die in der Voreinstellung 25 beträgt und so geändert werden kann:

```
> glm(..., control=glm.control(maxit=⟨Anzahl⟩))
```

Die Maximum-Likelihood-Schätzung der Parameter setzt voraus, dass keine (quasi-) vollständige Separierbarkeit von Prädiktoren durch die Kategorien von Y vorliegt (D, 2008). Dies ist der Fall, wenn Y stellenweise perfekt aus den Daten vorhersagbar ist. Das zugehörige geschätzte odds ratio $e^{\hat{\beta}_j}$ müsste dann ∞ sein. Ein Symptom für (quasi-) vollständige Separierbarkeit ist die Warnmeldung, dass geschätzte Trefferwahrscheinlichkeiten von 0 bzw. 1 aufgetreten sind. Ein weiteres Symptom für Separierbarkeit sind sehr große Parameterschätzungen, die mit großen Standardfehlern assoziiert sind und daher im Wald-Test kleine z-Werte liefern. In diesem Fall kann auf penalisierte Verfahren ausgewichen werden, etwa auf die logistische Regression mit Firth-Korrektur, die im Paket `logistf` (Heinze, Ploner, Dunkler & Southworth, 2013) umgesetzt wird (vgl. auch Abschn. 6.6.2). Eine andere Möglichkeit besteht darin, Kategorien in relevanten Prädiktorvariablen zusammenzufassen (D, 2008).

8.2 Ordinale Regression

In der ordinalen Regression soll eine kategoriale Variable Y mit k geordneten Kategorien $1, \ldots, g, \ldots, k$ mit p Prädiktoren X_j vorhergesagt werden. Dazu führt man die Situation auf jene der logistischen Regression zurück (vgl. Abschn. 8.1), indem man zunächst $k-1$ dichotome Kategorisierungen $Y \geq g$ vs. $Y < g$ mit $g = 2, \ldots, k$ vornimmt. Mit diesen dichotomen Kategorisierungen lassen sich nun $k-1$ *kumulative* Logits bilden:

$$\text{logit}(P(Y \geq g)) = \ln \frac{P(Y \geq g)}{1 - P(Y \geq g)} = \ln \frac{P(Y = g) + \cdots + P(Y = k)}{P(Y = 1) + \cdots + P(Y = g - 1)}$$

Die $k-1$ kumulativen Logits geben jeweils die logarithmierte Chance dafür an, dass Y mindestens die Kategorie g erreicht. Sie werden in $k-1$ separaten logistischen Regressionen mit dem Modell $\text{logit}(P(Y \geq g)) = \beta_{0_g} + \beta_1 X_1 + \cdots + \beta_p X_p$ linear vorhergesagt ($g = 2, \ldots, k$). Die Parameterschätzungen erfolgen dabei für alle Regressionen simultan mit der Nebenbedingung, dass die Menge der β_j für alle g identisch ist und $\beta_{0_2} > \cdots > \beta_{0_k}$ gilt. In diesem Modell führt

ein höherer Prädiktorwert X_j bei positivem β_j zu einer höheren Chance, dass eine höhere Kategorie von Y erreicht wird.[13]

Kern des Modells ist die Annahme, dass eine additive Erhöhung des Prädiktorwerts X_j um den Wert c dazu führt, dass die Chance für eine höhere Kategorie unabhängig von g um den festen Faktor $e^{\beta_j c}$ wächst: $\frac{P(Y \geq g|X_j+c)}{P(Y<g|X_j+c)} = e^{\beta_j c} \frac{P(Y \geq g|X_j)}{P(Y<g|X_j)}$. Aus diesem Grund heißt es auch *proportional odds* Modell. Wie in der logistischen Regression ist damit e^{β_j} das odds ratio, also der Faktor, um den der vorhergesagte Wettquotient zunimmt, wenn X_j um eine Einheit wächst.[14]

Wegen $P(Y = g) = P(Y \geq g) - P(Y \geq g+1)$ hat die Link-Funktion sowie ihre Umkehrfunktion hier eine zusammengesetzte Form. Mit der logistischen Funktion als Umkehrfunktion der Logit-Funktion sind die theoretischen Parameter $P(Y = g)$ identifizierbar als:

$$P(Y = g) = \frac{e^{\beta_{0_g}+\beta_1 X_1+\cdots+\beta_p X_p}}{1+e^{\beta_{0_g}+\beta_1 X_1+\cdots+\beta_p X_p}} - \frac{e^{\beta_{0_{g+1}}+\beta_1 X_1+\cdots+\beta_p X_p}}{1+e^{\beta_{0_{g+1}}+\beta_1 X_1+\cdots+\beta_p X_p}}$$

Die Modellannahmen lassen sich grafisch veranschaulichen (Abb. 8.3): Bei einem Prädiktor X sind die $k-1$ kumulativen Logits lineare Funktionen von X mit derselben Steigung und geordneten y-Achsenabschnitten. Die Wahrscheinlichkeit dafür, dass Y mindestens die Kategorie $g = 2, \ldots, k$ erreicht, ergibt sich aus $k-1$ parallelen logistischen Funktionen mit der horizontalen Verschiebung $\frac{\beta_{0_{g+1}}-\beta_{0_g}}{\beta_1}$, d. h. $P(Y \geq g+1|X) = P(Y \geq g|X - \frac{\beta_{0_{g+1}}-\beta_{0_g}}{\beta_1})$.

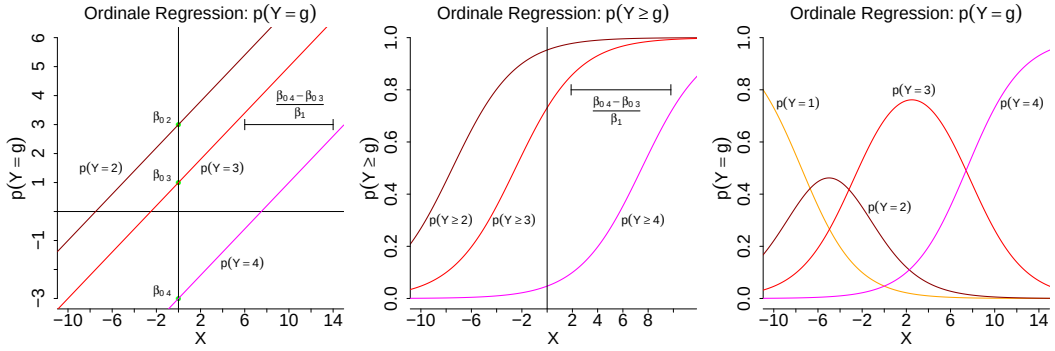

Abbildung 8.3: Modellannahmen der ordinalen Regression bei einem Prädiktor und 4 Kategorien des Kriteriums Y: Linearität der 3 kumulativen Logits mit identischer Steigung; parallel verschobene logistische Funktionen jeweils für die Wahrscheinlichkeit, mindestens Kategorie g zu erreichen; stochastisch geordnete Funktionen für die Wahrscheinlichkeit von $Y = g$.

[13] Andere Formulierungen des Modells sind möglich. So legt etwa SPSS das Modell $\text{logit}(P(Y \leq g)) = \beta_{0_g} - (\beta_1 X_1 + \cdots + \beta_p X_p)$ mit der Nebenbedingung $\beta_{0_1} < \cdots < \beta_{0_{k-1}}$ zugrunde, das jedoch nur zu umgedrehten Vorzeichen der Schätzungen für die β_{0_g} führt. Mit derselben Nebenbedingung ließe sich das Modell auch als $\text{logit}(P(Y \leq g)) = \beta_{0_g} + \beta_1 X_1 + \cdots + \beta_p X_p$ formulieren. In diesem Modell führt ein höherer Prädiktorwert X_j bei positivem β_j zu einer höheren Chance, dass eine *niedrigere* Kategorie von Y erreicht wird. Entsprechend haben hier die Schätzungen für alle Parameter umgekehrte Vorzeichen.

[14] Alternative proportional odds Modelle sind zum einen mit *adjacent category* Logits $\ln \frac{P(Y=g)}{P(Y=g-1)}$ möglich, zum anderen mit *continuation ratio* (*sequentiellen*) Logits $\ln \frac{P(Y=g)}{P(Y<g)}$.

8.2.1 Modellanpassung

Das proportional odds Modell mit kumulativen Logits kann mit `vglm()` aus dem Paket `VGAM` (Yee, 2010) angepasst werden. Diese Funktion hat gegenüber anderen, später ebenfalls erwähnten Funktionen den Vorteil, dass sie sich für alle in diesem Kapitel behandelten Modelle eignet und damit eine konsistente Herangehensweise an verwandte Fragestellungen ermöglicht.

```
> vglm(⟨Modellformel⟩, family=⟨Modell⟩, data=⟨Datensatz⟩)
```

Die Modellformel ist wie bei `glm()` zu formulieren, ihre linke Seite muss ein geordneter Faktor sein (vgl. Abschn. 2.6.4). Für das proportional odds Modell ist `family=propodds` zu setzen.[15] Schließlich benötigt `data` den Datensatz mit den Variablen aus der Modellformel.

Als Beispiel soll eine Variable mit $k = 4$ geordneten Kategorien anhand von $p = 2$ Prädiktoren vorhergesagt werden. Die kategoriale AV soll sich dabei aus der Diskretisierung einer kontinuierlichen Variable ergeben.

```
> N     <- 100
> X1    <- rnorm(N, 175, 7)                       # Prädiktor 1
> X2    <- rnorm(N,  30, 8)                       # Prädiktor 2
> Ycont <- 0.5*X1 - 0.3*X2 + 10 + rnorm(N, 0, 6)  # kontinuierliche AV

# geordneter Faktor aus Diskretisierung der kontinuierlichen AV
> Yord <- cut(Ycont, breaks=quantile(Ycont), include.lowest=TRUE,
+             labels=c("--", "-", "+", "++"), ordered=TRUE)

# zugehöriger ungeordneter Faktor für spätere Auswertungen
> Ycateg <- factor(Yord, ordered=FALSE)
> dfOrd  <- data.frame(X1, X2, Yord, Ycateg)      # Datensatz

> library(VGAM)                             # für vglm(), lrtest()
> vglmFit <- vglm(Yord ~ X1 + X2, family=propodds, data=dfOrd)
```

Die $k - 1$ Schätzungen der Parameter β_{0_g} sowie die p Schätzungen der Parameter β_j können mit `coef(⟨vglm-Modell⟩)` extrahiert werden. Die b_j sind wie in der logistischen Regression zu exponenzieren, um die geschätzten odds ratios zu erhalten (vgl. Abschn. 8.1.1): e^{b_j} gibt an, um welchen Faktor die vorhergesagte Chance wächst, eine höhere Kategorie zu erreichen, wenn sich X_j um eine Einheit erhöht. Die proportional odds Annahme bedeutet, dass sich die Wechselchance bei allen Kategorien im selben Maße ändert.

```
> exp(coef(vglmFit))                                # odds ratios
(Intercept):1 (Intercept):2 (Intercept):3           X1           X2
 3.554552e-11  9.225479e-12  2.195724e-12 1.170903e+00 9.355142e-01
```

[15] Mit `vglm()` ist es möglich, auch die proportional odds Modelle mit adjacent category Logits bzw. continuation ratio Logits anzupassen (vgl. Abschn. 8.2, Fußnote 14). Dazu ist `family` auf `acat(parallel=TRUE)` bzw. auf `sratio(parallel=TRUE)` zu setzen. Eine weitere Option für `acat()` bzw. `sratio()` ist dabei das Argument `reverse`, das die Vergleichsrichtung dieser Logits bzgl. der Stufen von Y kontrolliert und auf `TRUE` oder `FALSE` gesetzt werden kann.

8.2.2 Anpassungsgüte

Wie in der logistischen Regression dienen verschiedene Maße als Anhaltspunkte zur Anpassungsgüte des Modells (vgl. Abschn. 8.1.3). Dazu zählt die Devianz ebenso wie das Informationskriterium AIC sowie die pseudo-R^2 Kennwerte. Letztere basieren auf dem Vergleich der geschätzten Likelihoods von 0-Modell und vollständigem Modell und sind manuell zu ermitteln. Das 0-Modell ist dabei das Regressionsmodell ohne Prädiktoren X_j.[16]

```
> deviance(vglmFit)             # Devianz
[1] 244.1915

> AIC(vglmFit)                  # Informationskriterium AIC
[1] 254.1915

# 0-Modell für pseudo-R^2 Kennwerte
> vglm0 <- vglm(Yord ~ 1, family=propodds, data=dfOrd)
> LL0   <- logLik(vglm0)        # gesch. log-likelihood 0-Modell
> LLf   <- logLik(vglmFit)      # gesch. log-likelihood vollst. Modell

> as.vector( 1 - (LLf / LL0))                       # R^2 McFadden
[1] 0.1192653

> as.vector( 1 - exp((2/N) * (LL0 - LLf)))          # R^2 Cox & Snell
[1] 0.2815605

# R^2 Nagelkerke
> as.vector((1 - exp((2/N) * (LL0 - LLf))) / (1 - exp(LL0)^(2/N)))
[1] 0.3003312
```

Für potentiell auftretende Probleme bei der Modellanpassung vgl. Abschn. 8.1.7.

8.2.3 Signifikanztests für Parameter und Modell

Mit `summary(`⟨`vglm-Modell`⟩`)` werden neben den Parameterschätzungen b auch ihre geschätzten Streuungen $\hat{\sigma}_b$ sowie die zugehörigen z-Werte $\frac{b}{\hat{\sigma}_b}$ berechnet. Mit der Annahme, dass z unter der Nullhypothese asymptotisch standardnormalverteilt ist, stehen die zweiseitigen p-Werte in der Spalte `Pr(>|z|)`. Die genannten Werte lassen sich mit `coef(summary(`⟨`vglm-Modell`⟩`))` extrahieren.

```
> sumOrd   <- summary(vglmFit)
> (coefOrd <- coef(sumOrd))
                  Estimate Std. Error   z value     Pr(>|z|)
(Intercept):1 -24.06020695 5.82571121 -4.130003 3.627579e-05
(Intercept):2 -25.40905205 5.87912090 -4.321914 1.546818e-05
(Intercept):3 -26.84450928 5.94234461 -4.517495 6.257564e-06
X1              0.15777487 0.03385696  4.660042 3.161443e-06
```

[16] Weitere Gütemaße der Modellanpassung erzeugt `lrm()` aus dem Paket `rms` (vgl. Abschn. 8.1.3, Fußnote 9).

```
X2             -0.06665895 0.02499298 -2.667107 7.650735e-03
```

Ist von asymptotischer Normalverteilung der Schätzungen b_{j_g} auszugehen, können approximative $(1-\alpha)$-Wald-Konfidenzintervalle $[b - z_{1-\alpha/2}\,\hat{\sigma}_b, b + z_{\alpha/2}\,\hat{\sigma}_b]$ für die β_{j_g} berechnet werden, wobei $z_{\alpha/2}$ das $\frac{\alpha}{2}$-Quantil der Standardnormalverteilung ist.[17]

```
# 1-alpha/2 alpha/2 Quantile der Standardnormalverteilung
> zCrit <- qnorm(c(1 - 0.05/2, 0.05/2))

# Wald-Konfidenzintervalle für die Parameter (ohne intercepts b_0g)
> (ciCoef <- t(apply(coefOrd[-(1:3), ], 1, function(x) {
+              x["Estimate"] - zCrit*x["Std. Error"] } )))
          [,1]        [,2]
X1  0.09141645  0.22413329
X2 -0.11564429 -0.01767361
```

Eine oft geeignetere Alternative zu Wald-Tests sind Likelihood-Quotienten-Tests eines umfassenderen Modells ⟨`fitU`⟩ gegen ein eingeschränktes Modell ⟨`fitR`⟩ mit `lrtest(`⟨`fitU`⟩`,` ⟨`fitR`⟩`)` aus dem Paket `VGAM` (vgl. Abschn. 8.1.5).

```
# eingeschränktes Modell ohne Prädiktor X2
> vglmR <- vglm(Yord ~ X1, family=propodds, data=dfOrd)
> lrtest(vglmFit, vglmR)                  # Likelihood-Quotienten-Test
Likelihood ratio test

Model 1: Yord ~ X1 + X2
Model 2: Yord ~ X1
  #Df  LogLik Df  Chisq Pr(>Chisq)
1 295 -122.10
2 296 -125.87  1 7.5568   0.005978 **
```

Auf diese Weise lässt sich auch das Gesamtmodell gegen das 0-Modell ohne Prädiktoren X_j testen.

```
> lrtest(vglmFit, vglm0)                  # Test des Gesamtmodells
Likelihood ratio test

Model 1: Yord ~ X1 + X2
Model 2: Yord ~ 1
  #Df  LogLik Df  Chisq Pr(>Chisq)
1 295 -122.10
2 297 -138.63  2 33.067    6.6e-08 ***
```

Ohne die proportional odds Annahme, dass die Menge der β_j in allen $k-1$ separaten Regressionen der kumulativen Logits identisch ist, erhält man ein umfassenderes Modell. Die

[17] Für Konfidenzintervalle der Parameter kann die ordinale Regression auch zunächst mit `polr()` aus dem Paket `MASS` angepasst werden. Die dann von `confint(`⟨`polr-Modell`⟩`)` erzeugten Konfidenzintervalle basieren auf der Profile-Likelihood-Methode.

Parameter β_{j_g} sind dann von g abhängig, die linearen Funktionen für die Logits also nicht mehr parallel (Abb. 8.3). Dieses Modell lässt sich ebenfalls mit `vglm()` anpassen, indem man `family` auf `cumulative(parallel=FALSE)` setzt. Der Likelihood-Quotienten-Test des umfassenderen Modells gegen das eingeschränkte Modell mit proportional odds Annahme erfolgt wieder mit `lrtest()`. Fällt der Test signifikant aus, ist dies ein Hinweis darauf, dass die Daten gegen die proportional odds Annahme sprechen.

```
# eingeschränktes Modell - äquivalent zu vglm(..., family=propodds)
> vglmP <- vglm(Yord ~ X1 + X2, family=cumulative(parallel=TRUE,
+               reverse=TRUE), data=dfOrd)

# umfassenderes Modell ohne proportional odds Annahme
> vglmNP <- vglm(Yord ~ X1 + X2, family=cumulative(parallel=FALSE,
+                reverse=TRUE), data=dfOrd)

> lrtest(vglmNP, vglmNP)                  # Likelihood-Quotienten-Test
Likelihood ratio test

Model 1: Yord ~ X1 + X2
Model 2: Yord ~ X1 + X2
  #Df  LogLik Df  Chisq Pr(>Chisq)
1 291 -119.89
2 295 -122.10  4 4.4033   0.35422
```

8.2.4 Vorhersage, Klassifikation und Anwendung auf neue Daten

Die vorhergesagten Kategorienwahrscheinlichkeiten $\hat{P}(Y = g)$ erhält man wie in der logistischen Regression mit `predict(⟨vglm-Modell⟩, type="response")`. Die ausgegebene Matrix enthält für jede Beobachtung (Zeilen) die vorhergesagte Wahrscheinlichkeit für jede Kategorie (Spalten).

```
> PhatCateg <- predict(vglmFit, type="response")
> head(PhatCateg, n=3)
    Yord=--    Yord=-    Yord=+    Yord=++
1 0.1433775 0.2486796 0.3383702 0.26957271
2 0.2785688 0.3194630 0.2640555 0.13791266
3 0.5278107 0.2837526 0.1360684 0.05236823
```

Die vorhergesagten Kategorien selbst lassen sich aus den vorhergesagten Kategorienwahrscheinlichkeiten bestimmen, indem pro Beobachtung die Kategorie mit der maximalen vorhergesagten Wahrscheinlichkeit herangezogen wird. Das zeilenweise Maximum einer Matrix gibt `max.col(⟨Matrix⟩)` aus.

```
> categHat <- levels(dfOrd$Yord)[max.col(PhatCateg)]
> head(categHat)
[1] "+"  "-"  "--"  "+"  "-"  "+"
```

Die Kontingenztafel tatsächlicher und vorhergesagter Kategorien eignet sich als Grundlage für die Berechnung von Übereinstimmungsmaßen wie der Rate der korrekten Klassifikation. Hier ist darauf zu achten, dass die Kategorien identisch geordnet sind.

```
# Faktor mit gleich geordneten Kategorien
> facHat <- factor(categHat, levels=levels(dfOrd$Yord))

# Kontingenztafel tatsächlicher und vorhergesagter Kategorien
> cTab <- table(dfOrd$Yord, facHat, dnn=c("Yord", "facHat"))
> addmargins(cTab)                        # Randsummen hinzufügen
     facHat
Yord   --   -   +  ++ Sum
  --   10   7   7   1  25
  -    10   7   4   4  25
  +     5   7   7   6  25
  ++    0   2   9  14  25
  Sum  25  23  27  25 100

> (CCR <- sum(diag(cTab)) / sum(cTab))  # Rate der korrekten Klass.
[1] 0.38
```

An das Argument `newdata` von `predict()` kann zusätzlich ein Datensatz übergeben werden, der neue Daten für Variablen mit denselben Namen, und bei Faktoren zusätzlich denselben Stufen wie jene der ursprünglichen Prädiktoren enthält. Als Ergebnis erhält man die vorhergesagten Kategorienwahrscheinlichkeiten für die neuen Prädiktorwerte (vgl. Abschn. 6.4).

```
> Nnew  <- 3
> dfNew <- data.frame(X1=rnorm(Nnew, 175, 7),         # neue Daten
+                     X2=rnorm(Nnew,  30, 8))

> predict(vglmFit, newdata=dfNew, type="response")    # Vorhersage
     Yord=--    Yord=-    Yord=+    Yord=++
1 0.10563202 0.2071139 0.3438480 0.3434060
2 0.17385833 0.2739122 0.3253090 0.2269204
3 0.31326643 0.3240997 0.2433696 0.1192643
```

8.3 Multinomiale Regression

In der multinomialen Regression soll eine kategoriale Variable Y mit k ungeordneten Kategorien $1, \ldots, g, \ldots, k$ mit p Prädiktoren X_j vorhergesagt werden. Dazu führt man die Situation auf jene der logistischen Regression zurück (vgl. Abschn. 8.1), indem man zunächst eine Referenzkategorie r von Y festlegt und $k-1$ Logits der Form $\ln \frac{P(Y=g)}{P(Y=r)}$ bildet. Diese $k-1$ *baseline category* Logits geben die logarithmierte Chance dafür an, dass Y die Kategorie g annimmt – verglichen mit der Referenzkategorie r. Für r wird in der Voreinstellung typischerweise die erste (so hier im folgenden) oder die letzte Kategorie von Y gewählt.

Die baseline category Logits werden in $k-1$ separaten logistischen Regressionen mit dem Modell $\ln \frac{P(Y=g)}{P(Y=1)} = \beta_{0_g} + \beta_{1_g} X_1 + \cdots + \beta_{p_g} X_p$ linear vorhergesagt ($g = 1, \ldots, k$).[18] Die Parameterschätzungen erfolgen für alle Regressionen simultan, wobei anders als in der ordinalen Regression sowohl die β_{0_g} als auch die Gewichte β_{j_g} als von g abhängig betrachtet werden (Abb. 8.4). In diesem Modell führt ein höherer Prädiktorwert X_j bei positivem β_j zu einer höheren Chance, dass die Kategorie g von Y angenommen wird – verglichen mit der Referenzkategorie.[19] Mit den gewählten baseline category Logits sind auch alle verbleibenden logarithmierten Chancen beim Vergleich von je zwei Kategorien a, b von Y festgelegt:

$$\begin{aligned} \ln \frac{P(Y=a)}{P(Y=b)} &= \ln \frac{P(Y=a)/P(Y=1)}{P(Y=b)/P(Y=1)} = \ln \frac{P(Y=a)}{P(Y=1)} - \ln \frac{P(Y=b)}{P(Y=1)} \\ &= \beta_{0_a} + \beta_{1_a} X_1 + \cdots + \beta_{p_a} X_p \\ &- \beta_{0_b} + \beta_{1_b} X_1 + \cdots + \beta_{p_b} X_p \\ &= (\beta_{0_a} - \beta_{0_b}) + (\beta_{1_a} - \beta_{1_b}) X_1 + \cdots + (\beta_{p_a} - \beta_{p_b}) X_p \end{aligned}$$

Mit der logistischen Funktion als Umkehrfunktion der Logit-Funktion sind die theoretischen Parameter $P(Y = g)$ identifizierbar als:

$$P(Y = g) = \frac{\mathrm{e}^{\beta_{0_g} + \beta_{1_g} X_1 + \cdots + \beta_{p_g} X_p}}{\sum_{c=1}^{k} \mathrm{e}^{\beta_{0_c} + \beta_{1_c} X_1 + \cdots + \beta_{p_c} X_p}} = \frac{\mathrm{e}^{\beta_{0_g} + \beta_{1_g} X_1 + \cdots + \beta_{p_g} X_p}}{1 + \sum_{c=2}^{k} \mathrm{e}^{\beta_{0_c} + \beta_{1_c} X_1 + \cdots + \beta_{p_c} X_p}}$$

Insbesondere bestimmt sich die Wahrscheinlichkeit der Referenzkategorie 1 als:

$$P(Y = 1) = \frac{1}{1 + \sum_{c=2}^{k} \mathrm{e}^{\beta_{0_c} + \beta_{1_c} X_1 + \cdots + \beta_{p_c} X_p}}$$

8.3.1 Modellanpassung

Als Beispiel sollen die Daten der ordinalen Regression in Abschn. 8.2.1 herangezogen werden, wobei als AV nun der dort bereits erstellte ungeordnete Faktor dient. Die Anpassung des Modells erfolgt wieder mit `vglm()` aus dem Paket `VGAM`. Dafür ist das Argument `family` auf `multinomial(refLevel=1)` zu setzen, womit gleichzeitig die Referenzkategorie `refLevel` auf die erste Stufe von Y festgelegt werden kann. Wie in der logistischen Regression sind die Schätzungen b_{j_g} zu exponenzieren, um die geschätzten odds ratios zu erhalten: $\mathrm{e}^{b_{j_g}}$ gibt bezogen auf die Referenzkategorie an, um welchen Faktor die vorhergesagte Chance wächst, Kategorie g zu erreichen, wenn sich X_j um eine Einheit erhöht (vgl. Abschn. 8.1.1).

[18] Kurz $\ln \frac{P(Y=g)}{P(Y=1)} = \boldsymbol{X}\boldsymbol{\beta}_g$. In der Referenzkategorie 1 sind die Parameter wegen $\ln \frac{P(Y=1)}{P(Y=1)} = \ln 1 = 0$ festgelegt, und es gilt $\beta_{0_1} = \beta_{j_1} = 0$ (mit $j = 1, \ldots, p$) sowie $\mathrm{e}^{\boldsymbol{X}\boldsymbol{\beta}_g} = \mathrm{e}^0 = 1$.

[19] Dabei wird *Unabhängigkeit von irrelevanten Alternativen* angenommen: Für die Chance beim paarweisen Vergleich von g mit der Referenzkategorie soll die Existenz weiterer Kategorien irrelevant sein. Ohne diese Annahme kommen etwa Bradley-Terry-Modelle aus, von denen eine eingeschränkte Variante mit `brat()` aus dem Paket `VGAM` angepasst werden kann.

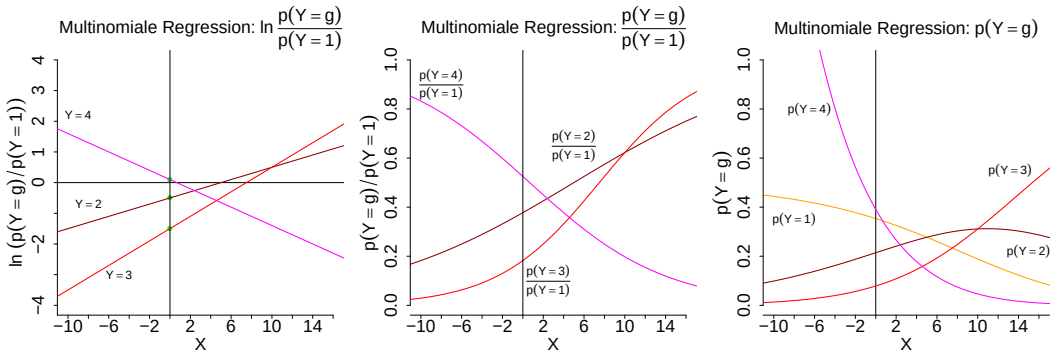

Abbildung 8.4: Multinomiale Regression mit einem Prädiktor und 4 Kategorien der AV Y: Linearität der 3 baseline category Logits mit unterschiedlichen Steigungen; logistische Funktionen für die Chance, verglichen mit der baseline Kategorie eine Kategorie g zu erhalten; zugehörige Funktionen für die Wahrscheinlichkeit einer Kategorie g von Y.

```
> library(VGAM)                                  # für vglm(), lrtest()
> vglmFitMN <- vglm(Ycateg ~ X1 + X2, family=multinomial(refLevel=1),
+                   data=dfOrd)

> exp(coef(vglmFitMN))                           # odds ratios
(Intercept):1 (Intercept):2 (Intercept):3          X1:1          X1:2
 5.812237e-02  1.579953e-11  8.460664e-20  1.019506e+00  1.162001e+00
         X1:3          X2:1          X2:2          X2:3
 1.316564e+00  9.854338e-01  9.636944e-01  8.647246e-01
```

8.3.2 Anpassungsgüte

Wie in der logistischen Regression dienen verschiedene Maße als Anhaltspunkte zur Anpassungsgüte des Modells (vgl. Abschn. 8.1.3). Dazu zählt die Devianz ebenso wie das Informationskriterium AIC sowie die pseudo-R^2 Kennwerte. Letztere basieren auf dem Vergleich der geschätzten Likelihoods von 0-Modell und vollständigem Modell und sind manuell zu ermitteln. Das 0-Modell ist dabei das Regressionsmodell ohne Prädiktoren X_j.

```
> deviance(vglmFitMN)           # Devianz
[1] 237.148

> AIC(vglmFitMN)                # Informationskriterium AIC
[1] 255.148

# 0-Modell für pseudo-R^2 Kennwerte
> vglmMN0 <- vglm(Ycateg ~ 1, family=multinomial, data=dfOrd)
> LL0     <- logLik(vglmMN0)    # gesch. log-likelihood 0-Modell
> LLf     <- logLik(vglmFitMN)  # gesch. log-likelihood vollst. Modell

> as.vector( 1 - (LLf / LL0))                    # R^2 McFadden
```

```
[1] 0.1446692

> as.vector( 1 - exp((2/N) * (LL0 - LLf)))          # R^2 Cox & Snell
[1] 0.3304224

# R^2 Nagelkerke
> as.vector((1 - exp((2/N) * (LL0 - LLf))) / (1 - exp(LL0)^(2/N)))
[1] 0.3524506
```

Für potentiell auftretende Probleme bei der Modellanpassung vgl. Abschn. 8.1.7.

8.3.3 Signifikanztests für Parameter und Modell

Mit `summary(⟨vglm-Modell⟩)` werden neben den Parameterschätzungen b auch ihre geschätzten Streuungen $\hat{\sigma}_b$ sowie die zugehörigen z-Werte $\frac{b}{\hat{\sigma}_b}$ berechnet. Mit der Annahme, dass z unter der Nullhypothese asymptotisch standardnormalverteilt ist, stehen die zweiseitigen p-Werte in der Spalte `Pr(>|z|)`. Die genannten Werte lassen sich mit `coef(summary(⟨vglm-Modell⟩))` extrahieren.

```
> sumMN   <- summary(vglmFitMN)
> (coefMN <- coef(sumMN))
                  Estimate  Std. Error    z value     Pr(>|z|)
(Intercept):1  -2.84520460  8.86963583 -0.3207803 7.483769e-01
(Intercept):2 -24.87104089  9.67909358 -2.5695630 1.018269e-02
(Intercept):3 -43.91627419 11.63358071 -3.7749576 1.600349e-04
X1:1            0.01931815  0.05232332  0.3692072 7.119733e-01
X1:2            0.15014353  0.05662132  2.6517138 8.008439e-03
X1:3            0.27502499  0.06780602  4.0560557 4.990836e-05
X2:1           -0.01467332  0.03810166 -0.3851098 7.001561e-01
X2:2           -0.03698110  0.04095569 -0.9029539 3.665504e-01
X2:3           -0.14534420  0.04957956 -2.9315347 3.372917e-03
```

Ist von asymptotischer Normalverteilung der Schätzungen b_{j_g} auszugehen, können approximative $(1-\alpha)$-Wald-Konfidenzintervalle $[b - z_{1-\alpha/2}\,\hat{\sigma}_b, b + z_{\alpha/2}\,\hat{\sigma}_b]$ für die β_{j_g} berechnet werden, wobei $z_{\alpha/2}$ das $\frac{\alpha}{2}$-Quantil der Standardnormalverteilung ist.

```
# 1-alpha/2 und alpha/2 Quantile der Standardnormalverteilung
> zCrit <- qnorm(c(1 - 0.05/2, 0.05/2))

# Wald-Konfidenzintervalle für die Parameter (ohne intercepts b_0g)
> (ciCoef <- t(apply(coefMN[-(1:3), ], 1, function(x) {
+              x["Estimate"] - zCrit*x["Std. Error"] } )))
            [,1]        [,2]
X1:1 -0.08323367  0.12186997
X1:2  0.03916779  0.26111927
X1:3  0.14212763  0.40792234
X2:1 -0.08935120  0.06000456
```

```
X2:2 -0.11725276  0.04329057
X2:3 -0.24251836 -0.04817005
```

Da jeder Prädiktor mit mehreren Parametern β_{j_g} assoziiert ist, müssen Prädiktoren selbst über Modellvergleiche auf Signifikanz getestet werden (vgl. Abschn. 8.1.5). Dazu dienen Likelihood-Quotienten-Tests, die auf der asymptotisch χ^2-verteilten Devianz-Differenz zweier nested Modelle mit demselben Kriterium beruhen: Der Prädiktorensatz des eingeschränkten Modells ⟨`fitR`⟩ ist dabei vollständig im Prädiktorensatz des umfassenderen Modells ⟨`fitU`⟩ enthalten, das zusätzlich noch weitere Prädiktoren berücksichtigt. Der Test erfolgt dann mit `lrtest(`⟨`fitU`⟩`,` ⟨`fitR`⟩`)` aus dem Paket `VGAM`.

```
# eingeschränktes Modell ohne Prädiktor X2
> vglmFitR <- vglm(Ycateg ~ X1, family=multinomial(refLevel=1), data=dfOrd)
> lrtest(vglmFitMN, vglmFitR)                 # Modellvergleich
Likelihood ratio test

Model 1: Ycateg ~ X1 + X2
Model 2: Ycateg ~ X1
  #Df  LogLik Df  Chisq Pr(>Chisq)
1 291 -118.57
2 294 -124.50  3 11.852   0.007906 **
```

8.3.4 Vorhersage, Klassifikation und Anwendung auf neue Daten

Die vorhergesagten Kategorienwahrscheinlichkeiten $\hat{P}(Y = g)$ erhält man wie in der logistischen Regression mit `predict(`⟨`vglm-Modell`⟩`, type="response")`. Die ausgegebene Matrix enthält für jede Beobachtung (Zeilen) die vorhergesagte Wahrscheinlichkeit für jede Kategorie (Spalten). Mit der gewählten Herangehensweise ist die mittlere vorhergesagte Wahrscheinlichkeit für jede Kategorie gleich der empirischen relativen Häufigkeit der Kategorie.[20]

```
> PhatCateg <- predict(vglmFitMN, type="response")
> head(PhatCateg, n=3)
         --         -         +         ++
1 0.1946834 0.2255730 0.2982816 0.28146203
2 0.3219964 0.3219154 0.2675170 0.08857120
3 0.4473732 0.3769135 0.1596527 0.01606071

# mittlere vorhergesagte Kategorien-Wahrscheinlichkeiten
> colMeans(PhatCateg)
  --    -    +   ++
0.25 0.25 0.25 0.25

# empirische relative Kategorien-Häufigkeiten
> prop.table(table(dfOrd$Ycateg))
```

[20] Dies ist der Fall, wenn die kanonische Link-Funktion und Maximum-Likelihood-Schätzungen der Parameter gewählt werden und das Modell die absoluten Terme β_{0_g} besitzt.

```
  --    -    +   ++
0.25 0.25 0.25 0.25
```

Die vorhergesagten Kategorien selbst lassen sich aus den vorhergesagten Kategorienwahrscheinlichkeiten bestimmen, indem pro Beobachtung die Kategorie mit der maximalen vorhergesagten Wahrscheinlichkeit herangezogen wird. Das zeilenweise Maximum einer Matrix gibt `max.col(⟨Matrix⟩)` aus.

```
> categHat <- levels(dfOrd$Ycateg)[max.col(PhatCateg)]
> head(categHat)
[1] "+"  "--"  "--"  "+"  "--" "+"
```

Die Kontingenztafel tatsächlicher und vorhergesagter Kategorien eignet sich als Grundlage für die Berechnung von Übereinstimmungsmaßen wie der Rate der korrekten Klassifikation. Hier ist darauf zu achten, dass die Kategorien identisch geordnet sind.

```
> facHat <- factor(categHat, levels=levels(dfOrd$Ycateg))
> cTab   <- table(dfOrd$Ycateg, facHat, dnn=c("Ycateg", "facHat"))
> addmargins(cTab)                      # Randsummen hinzufügen
           facHat
Ycateg  --   -   +  ++ Sum
   --    9   5   8   3  25
   -    11   5   5   4  25
   +     5   5   8   7  25
   ++    1   2   8  14  25
   Sum  26  17  29  28 100

> (CCR <- sum(diag(cTab)) / sum(cTab))  # Rate der korrekten Klass.
[1] 0.36
```

An das Argument `newdata` von `predict()` kann zusätzlich ein Datensatz übergeben werden, der neue Daten für Variablen mit denselben Namen, und bei Faktoren zusätzlich denselben Stufen wie jene der ursprünglichen Prädiktoren enthält. Als Ergebnis erhält man die vorhergesagten Kategorienwahrscheinlichkeiten für die neuen Prädiktorwerte (vgl. Abschn. 6.4).

```
> Nnew  <- 3
> dfNew <- data.frame(X1=rnorm(Nnew, 175, 7),         # neue Daten
+                     X2=rnorm(Nnew,  30, 8))

> predict(vglmFitMN, newdata=dfNew, type="response")  # Vorhersage
          --          -         +         ++
1 0.05150391 0.06762012 0.2943623 0.58651367
2 0.45228458 0.39748171 0.1311896 0.01904412
3 0.15948745 0.18932511 0.3110646 0.34012280
```

8.4 Regression für Zähldaten

Das GLM bietet verschiedene Möglichkeiten zur Modellierung einer Variable Y, die nur ganzzahlige nichtnegative Werte annehmen kann und keine obere Schranke aufweist, wie es für Zähldaten charakteristisch ist. Die von Y gezählten Ereignisse sollen dabei unabhängig voneinander eintreten. Handelt es sich bei den Prädiktoren X_j um kontinuierliche Variablen, spricht man von Regressionsmodellen, bei der Modellierung einer festen Gesamtzahl von Ereignissen durch Gruppierungsfaktoren meist allgemein von log-linearen Modellen (vgl. Abschn. 8.5).

8.4.1 Poisson-Regression

Bei der Poisson-Regression wird als bedingte Verteilung von Y eine Poisson-Verteilung $P(Y = y) = \frac{\mu^y e^{-\mu}}{y!}$ angenommen, mit dem Erwartungswert $\mu = E(Y)$. Die kanonische Link-Funktion ist der natürliche Logarithmus, das lineare Modell ist also $\ln \mu = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p = \boldsymbol{X\beta}$. Die bedingten Verteilungen von Y sind dann bereits vollständig festgelegt, da ihre Varianz ebenfalls gleich μ ist. Der Erwartungswert ist als $\mu = e^{\boldsymbol{X\beta}} = e^{\beta_0} e^{\beta_1 X_1} \cdot \ldots \cdot e^{\beta_p X_p}$ identifizierbar. Einem exponenzierten Parameter e^{β_j} kommt deshalb die Bedeutung des multiplikativen Faktors zu, mit dem μ wächst, wenn sich der Prädiktor X_j um eine Einheit vergrößert.[21]

Die Anpassung einer Poisson-Regression geschieht wie in der logistischen Regression mit `glm()`, wobei das Argument `family` auf `poisson(link="log")` zu setzen ist (vgl. Abschn. 8.1.1).[22] Die Daten der vorherzusagenden Variable müssen aus ganzzahligen nichtnegativen Werten bestehen.

Für ein Beispiel sollen zunächst Zähldaten simuliert werden, die mit zwei Prädiktoren korrelieren. Dazu wird die `rmvnorm()` Funktion des `mvtnorm` Pakets verwendet, die Zufallsvektoren einer multinormalverteilten Variable simuliert. Die Verwendung von `rmvnorm()` gleicht der von `rnorm()`, lediglich muss hier das theoretische Zentroid $\boldsymbol{\mu}$ für das Argument `mean` und die theoretische Kovarianzmatrix $\boldsymbol{\Sigma}$ für `sigma` angegeben werden. Die Daten einer der erzeugten Variablen werden zusätzlich gerundet und ihre negativen Werte auf Null gesetzt, damit sie als Zählvariable dienen kann.

```
> library(mvtnorm)                                  # für rmvnorm()
> N     <- 200
> sigma <- matrix(c(4,2,-3, 2,16,-1, -3,-1,8), byrow=TRUE, ncol=3)
> mu    <- c(-3, 2, 4)                              # Erwartungswerte
> XY    <- rmvnorm(N, mean=mu, sigma=sigma)         # Zufallsdaten
> Y     <- round(XY[ , 3] - 1.5)                    # runde Zähldaten
> Y[Y < 0] <- 0                                     # negative Werte -> 0
> dfCount <- data.frame(X1=XY[ , 1], X2=XY[ , 2], Y)    # Datensatz

# Poisson-Regression
> glmFitP <- glm(Y ~ X1 + X2, family=poisson(link="log"), data=dfCount)
```

[21] Für einen Prädiktor X: $\mu_{X+1} = e^{\beta_0+\beta_1(X+1)} = e^{\beta_0} e^{\beta_1(X+1)} = e^{\beta_0} e^{\beta_1 X} e^{\beta_1} = e^{\beta_1} e^{\beta_0+\beta_1 X} = e^{\beta_1} \mu_X$.
[22] Bei der Verwendung von `vglm()` aus dem Paket `VGAM` ist das Argument `family` auf `poissonff` zu setzen.

Wie in der logistischen Regression erhält man die Parameterschätzungen b_j mit `coef()` und die Konfidenzintervalle für die Parameter β_j mit `confint()` (vgl. Abschn. 8.1.1, Fußnote 4). Die geschätzten Änderungsfaktoren für $E(Y)$ ergeben sich durch Exponenzieren als e^{b_j}.

```
> exp(coef(glmFitP))                  # Änderungsfaktoren
(Intercept)          X1          X2
  1.1555238   0.7692037   1.0205773

> exp(confint(glmFitP))               # zugehörige Konfidenzintervalle
                2.5 %    97.5 %
(Intercept) 0.9479450 1.3993779
X1          0.7380968 0.8015816
X2          0.9992731 1.0424328
```

Wald-Tests der Parameter und Informationen zur Modellpassung insgesamt liefert `summary()` (vgl. Abschn. 8.1.3, 8.1.5).

```
> summary(glmFitP)                    # Parametertests + Modellpassung
Call:
glm(formula = Y ~ X1 + X2, family=poisson(link="log"), data=dfCount)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.1695  -1.4608  -0.2707   0.7173   3.4757

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.14455    0.09935   1.455    0.146
X1          -0.26240    0.02105 -12.466   <2e-16 ***
X2           0.02037    0.01079   1.888    0.059 .

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 536.93  on 199  degrees of freedom
Residual deviance: 374.78  on 197  degrees of freedom
AIC: 825.22

Number of Fisher Scoring iterations: 5
```

Wie in der logistischen Regression erhält man die vorhergesagten Häufigkeiten mit `predict(⟨glm-Modell⟩, type="response")` (vgl. Abschn. 8.1.4). Über das Argument `newdata` lässt sich das angepasste Modell dabei auch auf neue Werte für dieselben Prädiktoren anwenden, um Vorhersagen zu gewinnen.

8.4.2 Ereignisraten analysieren

Die Poisson-Regression erlaubt es auch, Ereignisraten zu analysieren. Die absoluten Häufigkeiten Y sind dann auf eine bestimmte Referenzgröße t bezogen, die die Menge der potentiell

beobachtbaren Ereignisse bestimmt. Bei t kann es sich etwa um die Länge eines Zeitintervalls handeln, in dem Ereignisse gezählt werden. In einer anderen Situation kann t die Größe einer Fläche sein, auf der die Anzahl von Elementen mit einer bestimmten Eigenschaft zu zählen sind, die jeweils einem Ereignis entsprechen. t wird als *exposure* bezeichnet und ist echt positiv.

Bezeichnet t die Zeitdauer, nimmt man an, dass der Abstand zwischen zwei aufeinander folgenden Ereignissen unabhängig von t exponentialverteilt mit Erwartungwert $\frac{1}{\lambda}$ ist. Dann folgt $E(Y) = \lambda t$ mit der Konstante λ als echt positiver Grundrate, mit der ein Ereignis pro Zeiteinheit auftritt.

Ist in einer Untersuchung die zu den beobachteten Häufigkeiten Y gehörende exposure t variabel, stellt die Grundrate $\lambda = \frac{\mu}{t}$ die zu modellierende Variable dar. Mit dem Logarithmus als Link-Funktion folgt $\boldsymbol{X\beta} = \ln \lambda = \ln \frac{\mu}{t} = \ln \mu - \ln t$, als lineares Vorhersagemodell ergibt sich $\ln \mu = \boldsymbol{X\beta} + \ln t$. Im linearen Prädiktor $\ln t + \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$ ist $\ln t$ eine additive Konstante wie β_0 – allerdings mit dem Unterschied, dass t kein zu schätzender Parameter, sondern durch die Daten festgelegt ist. $\ln t$ wird auch als *offset* bezeichnet. Die Grundrate λ ist identifizierbar als $\lambda = \mathrm{e}^{\boldsymbol{X\beta}+\ln t} = t\,\mathrm{e}^{\boldsymbol{X\beta}}$.

In der Anpassung der Poisson-Regression für Ereignisraten mit `glm()` ist das Argument `offset=log(⟨t⟩)` zu verwenden. Dabei ist ⟨`t`⟩ ein Vektor mit den Werten der exposure.

```
# simulierte Zähldaten, hier ohne Zusammenhang von Prädiktor und AV
> Nt   <- 100                                  # Beobachtungsobjekte
> Ti   <- sample(20:40, Nt, replace=TRUE)      # exposure
> Xt   <- rnorm(Nt, 100, 15)                   # Prädiktor
> Yt   <- rbinom(Nt, size=Ti, prob=0.5)        # beobachtete Häufigkeiten
> fitT <- glm(Yt ~ Xt, family=poisson(link="log"), offset=log(Ti))
> summary(fitT)                                # ...
```

8.4.3 Adjustierte Poisson-Regression und negative Binomial-Regression

Oft streuen empirische Residuen deutlich stärker, als dies bei bedingter Poisson-Verteilung von Y zu erwarten wäre, bei der die Streuung gleich dem bedingten Erwartungswert ist (*overdispersion*). Ein Hinweis auf overdispersion ist ein Verhältnis von Residual-Devianz zu Residual-Freiheitsgraden, das deutlich von 1 abweicht. Die modellbasierten Streuungsschätzungen unterschätzen dann die wahre Streuung, was zu liberale Signifikanztests der Parameter zur Folge hat. Eine mögliche Ursache für overdispersion ist ein unvollständiges Vorhersagemodell, das tatsächlich relevante Prädiktoren nicht berücksichtigt. Bei overdispersion kommen verschiedene Vorgehensweisen in Betracht:

So kann ein Poisson-ähnliches Modell verwendet werden, das einen zusätzlichen, aus den Daten zu schätzenden Streuungsparameter ϕ besitzt, mit dem für die bedingte Varianz $\sigma^2 = \phi\mu$ gilt. Hierfür ist das Argument `family` von `glm()` auf `quasipoisson(link="log")` zu setzen.[23] Dies führt zu identischen Parameterschätzungen b, jedoch zu anderen geschätzten Streuungen $\hat{\sigma}_b$ und damit zu anderen Ergebnissen der inferenzstatistischen Tests (vgl. Abschn. 8.1.6 für weitere Hinweise zu quasi-Familien).

[23] Bei der Verwendung von `vglm()` aus dem Paket `VGAM` ist das Argument `family` auf `quasipoissonff` zu setzen.

```
> glmFitQP <- glm(Y ~ X1 + X2, family=quasipoisson(link="log"),
+                 data=dfCount)

> summary(glmFitQP)
Call:
glm(formula = Y ~ X1 + X2, family = quasipoisson(link = "log"),
    data = dfCount)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.1695  -1.4608  -0.2707   0.7173   3.4757

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.14455    0.12943   1.117    0.265
X1          -0.26240    0.02742  -9.569   <2e-16 ***
X2           0.02037    0.01405   1.450    0.149

(Dispersion parameter for quasipoisson family taken to be 1.697012)

    Null deviance: 536.93  on 199  degrees of freedom
Residual deviance: 374.78  on 197  degrees of freedom
AIC: NA
```

Für eine adjustierte Poisson-Regression kann auch ein separater robuster Streuungsschätzer verwendet werden, wie ihn etwa das Paket `sandwich` mit `vcovHC()` bereitstellt. Wald-Tests der Parameter lassen sich dann mit `coeftest(`⟨`glm-Modell`⟩`, vcov=`⟨`Schätzer`⟩`)` aus dem Paket `lmtest` durchführen, wobei für das Argument `vcov` das Ergebnis von `vcovHC()` anzugeben ist.

```
> library(sandwich)                          # für vcovHC()
> hcSE <- vcovHC(glmFitP, type="HC0")        # robuste SE-Schätzung
> library(lmtest)                            # für coeftest()
> coeftest(glmFitP, vcov=hcSE)               # Parametertests
z test of coefficients:

             Estimate Std. Error  z value Pr(>|z|)
(Intercept)  0.144554   0.135709  1.0652   0.2868
X1          -0.262399   0.028753 -9.1259   <2e-16 ***
X2           0.020368   0.013097  1.5552   0.1199
```

Eine Alternative ist die Regression mit der Annahme, dass die bedingte Verteilung von Y eine negative Binomialverteilung ist, die einen eigenen Dispersionsparameter θ besitzt. Sie verallgemeinert die Poisson-Verteilung mit demselben Erwartungswert μ, ihre Varianz ist jedoch mit $\mu + \frac{\mu^2}{\theta} = \mu(1 + \frac{\mu}{\theta})$ um den Faktor $1 + \frac{\mu}{\theta}$ größer. Die negative Binomial-Regression lässt sich mit `glm.nb()` aus dem Paket `MASS` anpassen.[24]

[24] Bei der Verwendung von `vglm()` aus dem Paket `VGAM` ist das Argument `family` auf `negbinomial` zu setzen.

```
> library(MASS)                                  # für glm.nb()
> glmFitNB <- glm.nb(Y ~ X1 + X2, data=dfCount)
> summary(glmFitNB)
Call:
glm.nb(formula = Y ~ X1 + X2, data = dfCount, init.theta = 3.852967973,
    link = log)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6146  -1.2540  -0.2149   0.5074   2.7341

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.003337   0.130007   0.026   0.9795
X1          -0.302121   0.029972 -10.080   <2e-16 ***
X2           0.025419   0.014649   1.735   0.0827 .

(Dispersion parameter for Negative Binomial(3.853) family taken to be 1)

    Null deviance: 353.34  on 199  degrees of freedom
Residual deviance: 250.31  on 197  degrees of freedom
AIC: 801.11

Number of Fisher Scoring iterations: 1

              Theta:  3.85
          Std. Err.:  1.12
 2 x log-likelihood:  -793.113
```

Das Ergebnis nennt neben den bekannten Kennwerten unter `Theta` eine Schätzung für den Dispersionsparameter θ. Das von `glm.nb()` erzeugte Objekt kann an die Funktion `odTest()` aus dem Paket `pscl` (Zeileis, Kleiber & Jackman, 2008) übergeben werden. Sie führt dann einen Test auf overdispersion durch, der auf dem Vergleich der Anpassungsgüte von Poisson- und negativem Binomial-Modell beruht.

```
> library(pscl)                                  # für odTest()
> odTest(glmFitNB)                               # overdispersion-Test
Likelihood ratio test of H0: Poisson, as restricted NB model:
n.b., the distribution of the test-statistic under H0 is non-standard
e.g., see help(odTest) for details/references

Critical value of test statistic at the alpha= 0.05 level: 2.7055
Chi-Square Test Statistic = 26.1033 p-value = 1.618e-07
```

8.4.4 Zero-inflated Poisson-Regression

Eine mögliche Quelle für starke Streuungen von Zähldaten sind gehäuft auftretende Nullen. Sie können das Ergebnis eines zweistufigen Prozesses sein, mit dem die Daten zustande kommen. Wird etwa erfasst, wie häufig Personen im Beruf befördert werden oder ihren Arbeitsplatz wechseln, ist zunächst relevant, ob sie überhaupt jemals eine Arbeitsstelle erhalten haben. Diese Eigenschaft wird vermutlich durch qualitativ andere Prozesse bestimmt als jene, die bei tatsächlich Erwerbstätigen die Anzahl der unterschiedlichen Positionen beeinflussen. Die Daten können damit als Ergebnis einer Mischung von zwei Verteilungen verstanden werden: Zum einen kommen Nullen durch Personen zustande, die ein notwendiges Kriterium für einen auch zu echt positiven Häufigkeiten führenden Prozess nicht erfüllen. Zum anderen verursachen Personen, die ein solches Kriterium erfüllen, ihrerseits Nullen sowie zusätzlich echt positive Häufigkeiten. Für die Daten dieser zweiten Gruppe von Personen kann nun wieder angenommen werden, dass sie sich durch eine Poisson- oder negative Binomialverteilung beschreiben lassen.

In der geschilderten Situation kommen *zero-inflated* Modelle in Betracht. Sie sorgen durch die Annahme einer Mischverteilung letztlich dafür, dass im Modell eine deutlich höhere Auftretenswahrscheinlichkeit von Nullen möglich ist, als es zur Verteilung der echt positiven Häufigkeiten passt.

Die zero-inflated Poisson-Regression eignet sich für Situationen, in denen keine starke overdispersion zu vermuten ist und kann mit `zeroinfl()` aus dem Paket `pscl` angepasst werden (für Details vgl. `vignette("countreg")`).[25]

```
> zeroinfl(⟨Modellformel⟩, dist="⟨Verteilung⟩", offset=⟨offset⟩,
+          data=⟨Datensatz⟩)
```

Die Modellformel besitzt hier die Form ⟨`AV`⟩ `~` ⟨`UV`⟩ `|` ⟨`0-Anteil`⟩. Für ⟨`0-Anteil`⟩ ist ein Prädiktor zu nennen, der im Modell separat den Anteil der „festen“ Nullen an den Daten kontrolliert, die von jenen Personen stammen, die prinzipiell keine echt positiven Daten liefern können. Im einfachsten Fall ist dies der absolute Term `1`, der zu einem Binomialmodell passt, das für alle Beobachtungen dieselbe Wahrscheinlichkeit vorsieht, zu dieser Gruppe zu gehören. In komplizierteren Situationen könnte die Gruppenzugehörigkeit analog zu einer separaten logistischen Regression durch einen eigenen Prädiktor vorhergesagt werden. Das Argument `dist` ist auf `"poisson"` zu setzen, `data` erwartet einen Datensatz mit den Variablen aus der Modellformel. Wie bei `glm()` existiert ein Argument `offset` für die Analyse von Ereignisraten (vgl. Abschn. 8.4.2).

```
> library(pscl)                                  # für zeroinfl()
> ziFitP <- zeroinfl(Y ~ X1 + X2 | 1, dist="poisson", data=dfCount)
> summary(ziFitP)
Call:
zeroinfl(formula = Y ~ X1 + X2 | 1, data = dfCount, dist = "poisson")

Pearson residuals:
    Min      1Q  Median      3Q     Max
-1.4805 -0.9447 -0.1574  0.6397  3.8149
```

[25] Bei der Verwendung von `vglm()` aus dem Paket `VGAM` ist das Argument `family` auf `zipoissonff` zu setzen.

```
Count model coefficients (poisson with log link):
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.50809    0.13063   3.890   0.0001 ***
X1          -0.20443    0.02658  -7.692 1.45e-14 ***
X2           0.01781    0.01156   1.541   0.1233

Zero-inflation model coefficients (binomial with logit link):
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -1.4957     0.2627  -5.693 1.25e-08 ***

Number of iterations in BFGS optimization: 10
Log-likelihood: -393.1 on 4 Df
```

Separate Walt-Tests der Parameter lassen sich mit `coeftest()` aus dem Paket `lmtest` durchführen, Likelihood-Quotienten-Tests für nested Modelle können mit `lrtest()` aus demselben Paket vorgenommen werden.

Die zero-inflated negative Binomial-Regression kann in Situationen mit gehäuft auftretenden Nullen und einer deutlichen overdispersion zum Einsatz kommen. Auch für sie eignet sich `zeroinfl()`, wobei das Argument `dist="negbin"` zu setzen ist.[26]

```
> ziFitNB <- zeroinfl(Y ~ X1 + X2 | 1, dist="negbin", data=dfCount)
> summary(ziFitNB)

Call:
zeroinfl(formula = Y ~ X1 + X2 | 1, data = dfCount, dist = "negbin")

Pearson residuals:
    Min      1Q  Median      3Q     Max
-1.4548 -0.9007 -0.1515  0.6482  3.9289

Count model coefficients (negbin with log link):
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.36993    0.17740   2.085   0.0370 *
X1          -0.23375    0.03738  -6.254    4e-10 ***
X2           0.02111    0.01320   1.599   0.1098
Log(theta)   2.65338    0.81911   3.239   0.0012 **

Zero-inflation model coefficients (binomial with logit link):
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -1.7486     0.3924  -4.457 8.33e-06 ***

Theta = 14.2019
Number of iterations in BFGS optimization: 20
Log-likelihood:  -392 on 5 Df
```

[26] Bei der Verwendung von `vglm()` aus dem Paket `VGAM` ist das Argument `family` auf `zinegbinomial` zu setzen.

Der Vuong-Test vergleicht die Anpassungsgüte eines von `glm()` erstellten Poisson-Modells mit jener des durch `zeroinfl()` erstellten zero-inflated Poisson-Modells und kann mit `vuong()` aus dem Paket `pscl` durchgeführt werden. Analog testet `vuong()` auch ein mit `glm.nb()` angepasstes negatives Binomial-Modell gegen das zero-inflated negative Binomial-Modell aus `zeroinfl()`.

```
# Poisson-Modell vs. zero-inflated Poisson-Modell
> library(pscl)                                # für vuong()
> vuong(ziFitP, glmFitP)
Vuong Non-Nested Hypothesis Test-Statistic: -0.4582521
(test-statistic is asymptotically distributed N(0,1) under the
null that the models are indistinguishible)
in this case:
model2 > model1, with p-value 0.32339

# negatives Binomial- vs. zero-inflated negatives Binomial-Modell
> vuong(ziFitNB, glmFitNB)
Vuong Non-Nested Hypothesis Test-Statistic: -1.382096
(test-statistic is asymptotically distributed N(0,1) under the
null that the models are indistinguishible)
in this case:
model2 > model1, with p-value 0.083471
```

Eine Alternative zu zero-inflated Modellen bei gehäuft auftretenden Nullen ist die Hurdle-Regression, die von `hurdle()` aus dem Paket `pscl` umgesetzt wird.

8.4.5 Zero-truncated Poisson-Regression

Im Gegensatz zum Umstand gehäuft auftretender Nullen erzwingt der Aufbau mancher Untersuchungen, dass *keine* Nullen beobachtet werden können. Dies ist etwa der Fall, wenn mindestens ein Ereignis vorliegen muss, damit eine Untersuchungseinheit in der Erhebung berücksichtigt wird: So ließe sich an bereits stationär aufgenommenen Patienten die Anzahl der Tage erheben, die sie im Krankenhaus verbringen, oder es könnte die Anzahl geborener Hundewelpen eines Wurfes erfasst werden. Zur Modellierung solcher Variablen kommen beschränkte (*zero-truncated*) Verteilungen in Frage, die den Wert 0 nur mit Wahrscheinlichkeit Null annehmen. Dazu zählen die zero-truncated Poisson-Verteilung – für Situationen ohne overdispersion – sowie die zero-truncated negative Binomialverteilung – bei overdispersion. Um sie bei der Modellanpassung mit `vglm()` aus dem Paket `VGAM` zu verwenden, ist das Argument `family` auf `pospoisson` bzw. auf `posnegbinomial` zu setzen.

8.5 Log-lineare Modelle

Log-lineare Modelle analysieren den Zusammenhang mehrerer kategorialer Variablen auf Basis ihrer gemeinsamen Häufigkeitsverteilung (Agresti, 2007). Sie sind für Daten geeignet, die sich als mehrdimensionale Kontingenztafeln absoluter Häufigkeiten darstellen lassen und verallgemeinern

damit klassische nonparametrische Tests auf Unabhängigkeit bzw. auf Gleichheit von bedingten Verteilungen (vgl. Abschn. 10.2, 10.3.2).

8.5.1 Modell

Aus Sicht der Datenerhebung können Kontingenztafeln absoluter Häufigkeiten aus drei Situationen entstehen. Sie sollen hier an einer $(P \times Q)$-Kontingenztafel der kategorialen Variablen X_1 und X_2 erläutert werden.

	$x_{2.1}$	$\dots$	$x_{2.q}$	$\dots$	$x_{2.Q}$	Summe
$x_{1.1}$	f_{11}	$\dots$	f_{1k}	$\dots$	f_{1q}	f_{1+}
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$x_{1.p}$	f_{j1}	$\dots$	f_{jk}	$\dots$	f_{jq}	f_{p+}
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$x_{1.P}$	f_{p1}	$\dots$	f_{pk}	$\dots$	f_{pq}	f_{P+}
Summe	f_{+1}	$\dots$	f_{+q}	$\dots$	f_{+Q}	N

- Im *produkt-multinomialen* Erhebungsschema ist X_1 ein fester Faktor mit vorgegebenen Gruppenhäufigkeiten f_{p+}, aus deren Summe sich die feste Anzahl N von Beobachtungsobjekten ergibt – etwa in einem Experiment mit kontrollierter Zuweisung zu Versuchsbedingungen. Erhoben wird die kategoriale Variable X_2, deren Verteilung über die Gruppen j i. S. der Auftretenshäufigkeiten ihrer Kategorien f_{+q} zufällig ist.
- Im *multinomialen* Erhebungsschema werden mehrere kategoriale Variablen X_j gleichzeitig an einer festen Anzahl N von Beobachtungsobjekten erhoben, etwa im Rahmen einer Befragung. Die Kontingenztafel entsteht aus der Kreuzklassifikation der X_j und gibt die Häufigkeiten an, mit der Kombinationen ihrer Kategorien auftreten. Dabei sind die f_{p+} und f_{+q} zufällig. In diesem Schema werden keine Einflussbeziehungen der X_j auf eine separate Variable untersucht, sondern Zusammenhänge zwischen den X_j selbst.
- Im *Poisson*-Erhebungsschema wird gleichzeitig mit den Ausprägungen kategorialer Einflussgrößen X_j eine separate Variable Y als Anzahl bestimmter Ereignisse erhoben. Die Kontingenztafel stellt die Anzahl der Ereignisse in den Gruppen dar, die durch die Kombination der Stufen der X_j entstehen. Dabei ist die Gesamthäufigkeit der Ereignisse N wie auch ihre Verteilung in den Gruppen f_{p+} und f_{+q} zufällig. In diesem Schema nimmt man eine Poisson-Verteilung für N an, womit jedes Ereignis unabhängig voneinander eintritt und als einzelne Beobachtung der Merkmalskombinationen der X_j zählt. Bedingt auf N liegt dann wieder ein multinomiales Schema vor.

Das log-lineare Modell ist ein lineares Modell für $\ln \mu_{jk}$, den logarithmierten Erwartungswert einer Zellhäufigkeit f_{jk}. Formuliert als Regression mit Treatment-Kontrasten hat es dieselbe Form wie das der Poisson-Regression (vgl. Abschn. 8.4.1). Es ist jedoch üblicher, es wie die mehrfaktorielle Varianzanalyse (vgl. Abschn. 7.5) mit Effektcodierung (vgl. Abschn. 12.9.2) zu parametrisieren. Mit der Stichprobengröße N, der Zellwahrscheinlichkeit p_{jk} und den

Randwahrscheinlichkeiten p_{p+}, p_{+q} analog zu den Häufigkeiten gilt dafür zunächst:

$$\begin{aligned} \mu_{jk} = N p_{jk} &= N p_{p+}\, p_{+q}\, \frac{p_{jk}}{p_{p+}\, p_{+q}} \\ \ln \mu_{jk} = \ln N + \ln p_{jk} &= \ln N + \ln p_{p+} + \ln p_{+q} + (\ln p_{jk} - (\ln p_{p+} + \ln p_{+q})) \end{aligned}$$

Das Modell für $\ln \mu_{jk}$ hat nun für eine zweidimensionale Kontingenztafel dieselbe Form wie jenes der zweifaktoriellen Varianzanalyse mit Effektcodierung. Dabei ist $\ln N$ analog zu μ, $\ln p_{p+}$ analog zu α_j (Zeileneffekt von Gruppe j des Faktors X_1), $\ln p_{+q}$ analog zu β_k (Spalteneffekt von Stufe k des Faktors X_2) und $\ln p_{jk} - (\ln p_{p+} + \ln p_{+q})$ analog zu $(\alpha\beta)_{jk}$ (Interaktionseffekt).[27]

$$\begin{aligned} \mu_{jk} &= \mathrm{e}^{\mu}\, \mathrm{e}^{\alpha_j}\, \mathrm{e}^{\beta_k}\, \mathrm{e}^{(\alpha\beta)_{jk}} \\ \ln \mu_{jk} &= \mu + \alpha_j + \beta_k + (\alpha\beta)_{jk} \end{aligned}$$

Als Abweichung der logarithmierten Zellwahrscheinlichkeit von Additivität der logarithmierten Randwahrscheinlichkeiten drückt der Interaktionseffekt $(\alpha\beta)_{jk}$ den Zusammenhang von X_1 und X_2 aus, da bei Unabhängigkeit $p_{jk} = p_{p+}\, p_{+q}$ gilt – logarithmiert also $\ln p_{jk} = \ln p_{p+} + \ln p_{+q}$. In zweidimensionalen Kreuztabellen sind alle beobachtbaren Zellhäufigkeiten mit dem vollständigen Modell (Zeilen-, Spalten- und Interaktionseffekt) verträglich, das deswegen als *saturiert* bezeichnet wird und sich nicht testen lässt. Dagegen sind eingeschränkte Modelle wie das der Unabhängigkeit statistisch prüfbar. Für höherdimensionale Kontingenztafeln gilt dies analog, wobei kompliziertere Abhängigkeitsbeziehungen durch die Interaktionen erster und höherer Ordnung ausgedrückt werden können.

Für den Spezialfall zweidimensionaler Kreuztabellen ist das beschriebene Unabhängigkeitsmodell im multinomialen Erhebungsschema dasselbe, das vom χ^2-Test auf Unabhängigkeit geprüft wird (vgl. Abschn. 10.2.1). Im produkt-multinomialen Erhebungsschema ist es analog dasselbe, das der χ^2-Test auf Gleichheit von bedingten Verteilungen testet (vgl. Abschn. 10.2.2).

8.5.2 Modellanpassung

Log-lineare Modelle lassen sich mit der Funktion `loglm()` aus dem Paket `MASS` testen, in der die Modelle analog zu `lm()` formuliert werden können.[28]

```
> loglm(⟨Modellformel⟩, data=⟨Kreuztabelle⟩)
```

Stammen die für `data` zu übergebenden Daten aus einer Kreuztabelle absoluter Häufigkeiten, kann das erste Argument eine Modellformel ohne Variable links der `~` sein. Besitzt die Kreuztabelle Namen für Zeilen und Spalten, sind diese mögliche Vorhersageterme in der Modellformel. Alternativ stehen die Ziffern $1, 2, \ldots$ für die Effekte der Zeilen, Spalten, etc. `data` kann auch ein Datensatz sein, der Spalten für die kategorialen Variablen X_j sowie für die Auftretenshäufigkeit

[27] Anders als in der Varianzanalyse gibt es jedoch im log-linearen Modell nur eine Beobachtung pro Zelle, die Rolle der abhängigen Variable der Varianzanalyse hat im log-linearen Modell die logarithmierte Auftretenshäufigkeit der zur Zelle gehörenden Kombination von Faktorstufen.

[28] `loglm()` basiert auf `loglin()`, bietet jedoch die von Regression und Varianzanalyse vertraute Methode, eine Modellformel zur Beschreibung des log-linearen Modells zu verwenden.

Y jeder ihrer Stufenkombination besitzt (vgl. Abschn. 2.10.5). Wie gewohnt bildet Y dann die linke Seite der Modellformel und eine Kombination der X_j die rechte Seite.

Als Beispiel soll die $(2 \times 2 \times 6)$-Kreuztabelle `UCBAdmissions` dienen, die im Basisumfang von R enthalten ist. Aufgeschlüsselt nach Geschlecht (Variable `Gender`) vermerkt sie die Häufigkeit, mit der Bewerber für die sechs größten Fakultäten (Variable `Dept`) an der University of California Berkeley 1973 angenommen bzw. abgelehnt wurden (Variable `Admit`).

```
> str(UCBAdmissions)              # Struktur der Kreuztabelle
 table [1:2, 1:2, 1:6] 512 313 89 19 353 207 17 8 120 205 ...
 - attr(*, "dimnames")=List of 3
  ..$ Admit : chr [1:2] "Admitted" "Rejected"
  ..$ Gender: chr [1:2] "Male" "Female"
  ..$ Dept  : chr [1:6] "A" "B" "C" "D" ...

# teste Modell der vollständigen Unabhängigkeit = Additivität
> (llFit <- loglm(~ Admit + Dept + Gender, data=UCBAdmissions))
Call:
loglm(formula = ~Admit + Dept + Gender, data = UCBAdmissions)

Statistics:
                     X^2 df P(> X^2)
Likelihood Ratio 2097.671 16        0
Pearson          2000.328 16        0

# wandle Kontingenztafel in Datensatz um
> UCBAdf <- as.data.frame(UCBAdmissions)
> loglm(Freq ~ Admit + Dept + Gender, data=UCBAdf)  # ...
```

Die Ausgabe von `loglm()` nennt die Kennwerte des Likelihood-Quotienten-Tests sowie des Pearson χ^2-Tests der Hypothese, dass das angegebene Modell stimmt. Der Wert der Teststatistik steht in der Spalte `X^2`, die zugehörigen Freiheitsgrade unter `df` und der p-Wert unter `P(> X^2)`. Mit Hilfe von `coef(`⟨`loglm-Modell`⟩`)` erhält man zusätzlich Schätzungen für die durch die Modellformel spezifizierten Koeffizienten $\alpha_j, \beta_k, \ldots$. Dabei ist zu beachten, dass `loglm()` die Effektcodierung verwendet, die geschätzten Zeilen-, Spalten- und Schichten-Parameter summieren sich also pro Variable zu 0 (vgl. Abschn. 12.9.2).

```
> (llCoef <- coef(llFit))
$`(Intercept)`
[1] 5.177567

$Admit
  Admitted   Rejected
-0.2283697  0.2283697

$Gender
      Male     Female
 0.1914342 -0.1914342
```

```
$Dept
          A           B           C           D           E           F
 0.23047857 -0.23631478  0.21427076  0.06663476 -0.23802565 -0.03704367
```

Die gemeinsamen Häufigkeiten mehrerer kategorialer Variablen lassen sich mit `mosaicplot()` in einem Mosaik-Diagramm darstellen, einer Erweiterung des splineplot für die gemeinsame Verteilung zweier kategorialer Variablen (vgl. Abschn. 14.4.2). Zusammen mit dem Argument `shade=TRUE` sind die Argumente dieselben wie für `loglm()`, die zellenweisen Pearson-Residuen bzgl. des durch die Modellformel definierten Models werden dann farbcodiert dargestellt (Abb. 8.5).

```
# Mosaik-Diagramm der Häufigkeiten und Pearson-Residuen
> mosaicplot(~ Admit + Dept + Gender, shade=TRUE, data=UCBAdmissions)
```

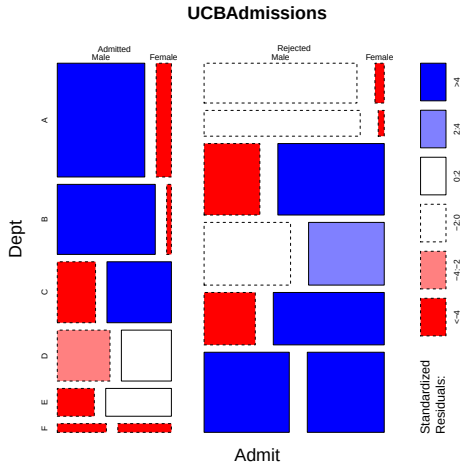

Abbildung 8.5: Mosaik-Diagramm der gemeinsamen Häufigkeiten der Kontingenztafel `UCBAdmissions` inkl. Farbcodierung der Pearson-Residuen vom Modell der Unabhängigkeit.

Wie in der Poisson-Regression erhält man mit `residuals(`⟨`loglm-Modell`⟩`, type="`⟨`Typ`⟩`")` die Residuen, hier allerdings nicht pro Beobachtung, sondern pro Zelle der Kontingenztafel. Mit `type="pearson"` sind dies die Pearson-Residuen, deren Quadratsumme gleich der Pearson-Teststatistik in der Ausgabe von `loglm()` ist. Für zweidimensionale Kontingenztafeln ist diese Quadratsumme gleich der Teststatistik des χ^2-Tests auf Unabhängigkeit mit `chisq.test()` (vgl. Abschn. 10.2.1). Analog erhält man mit `type="deviance"` die Devianz-Residuen, deren Quadratsumme gleich der Likelihood-Ratio-Teststatistik in der Ausgabe von `loglm()` ist.

```
> sum(residuals(llFit, type="deviance")^2)
[1] 2097.671

> sum(residuals(llFit, type="pearson")^2)
[1] 2000.328
```

Anders als in der Poisson-Regression liefert ein mit `loglm()` angepasstes Modell keine Standardfehler der Parameterschätzungen. Um diese zu beurteilen, lässt sich das Modell jedoch als Poisson-Regression mit `glm()` formulieren. Da `glm()` in der Voreinstellung Treatment-Kontraste verwendet, sind die Parameterschätzungen zunächst andere, mit Effektcodierung jedoch dieselben. Vorhersage und Residuen stimmen mit der des log-linearen Modells überein.[29]

```
# Poisson-Regression mit Treatment-Kontrasten
> glmFitT <- glm(Freq ~ Admit+Dept+Gender, family=poisson(link="log"),
+                data=UCBAdf)

# prüfe Gleichheit der Vorhersage mit log-linearem Modell
> all.equal(c(fitted(llFit)), fitted(glmFitT), check.attributes=FALSE)
[1] TRUE

> (glmTcoef <- coef(glmFitT))             # Parameterschätzungen
Coefficients:
(Intercept)  AdmitRejected       DeptB       Dept       DeptD       DeptE
    5.37111         0.45674    -0.46679    -0.01621    -0.16384    -0.46850

      DeptF  GenderFemale
   -0.26752      -0.38287
```

Bei Treatment-Kontrasten ist der absolute Term gleich der Vorhersage in der Zelle, die sich als Kombination aller Referenzkategorien der beteiligten Faktoren (in der Voreinstellung jeweils die erste Faktorstufe) ergibt. Die geschätzten Koeffizienten geben jeweils die geschätzte Abweichung für eine Faktorstufe von der Referenzkategorie an.

```
# Umrechnung mu_jk Effektcodierung aus loglm() in Treatment-Kontraste
# Zelle aus Kombination der Referenz-Kategorien
> glmTcoef["(Intercept)"]                              # Treatment-Kontraste
(Intercept)
    5.37111

> llCoef$`(Intercept)` + llCoef$Admit["Admitted"] +   # Effektcodierung
+     llCoef$Gender["Male"]  + llCoef$Dept["A"]
Admitted
 5.37111

# mu_jk für Admit = "Admit", Gender = "Female", Dept = "C"
# Treatment-Kontraste
> glmTcoef["(Intercept)"] + glmTcoef["DeptC"] + glmTcoef["GenderFemale"]
(Intercept)
   4.972034

> llCoef$`(Intercept)` + llCoef$Admit["Admitted"] +   # Effektcodierung
```

[29] Damit ist auch die Teststatistik des Likelihood-Quotienten-Tests im Prinzip identisch, da `loglm()` und `glm()` jedoch andere numerische Optimierungsverfahren zur Maximum-Likelihood-Schätzung verwenden, sind kleine Abweichungen möglich.

```
+     llCoef$Dept["C"] + llCoef$Gender["Female"]
Admitted
4.972034
```

Auch mit `glm()` lassen sich die Parameter mit Effektcodierung schätzen (vgl. Abschn. 7.5.2). Dabei fehlt die Parameterschätzung für die jeweils letzte Faktorstufe in der Ausgabe, da sie sich aus der Nebenbedingung ergibt, dass sich die Parameter pro Faktor zu Null summieren. Die Standardfehler der Parameterschätzungen sowie den zugehörigen Wald-Test extrahiert man mit `coef(summary(⟨glm-Modell⟩))`. Die Einträge für den absoluten Term `(Intercept)` sind hier ohne Bedeutung, da $\ln N$ im log-linearen Modell fest vorgegeben ist und nicht geschätzt werden muss.

```
# Poisson-Regression mit Effektcodierung
> glmFitE <- glm(Freq ~ Admit + Dept + Gender, family=poisson(link="log"),
+                contrasts=list(Admit=contr.sum,
+                               Dept=contr.sum,
+                              Gender=contr.sum), data=UCBAdf)

# Paramterschätzungen mit Standardfehlern und Wald-Tests
> coef(summary(glmFitE))
              Estimate Std. Error    z value     Pr(>|z|)
(Intercept)  5.17756677 0.01577888 328.132716 0.000000e+00
Admit1      -0.22836970 0.01525343 -14.971696 1.124171e-50
Dept1        0.23047857 0.03071783   7.503086 6.233240e-14
Dept2       -0.23631478 0.03699431  -6.387869 1.682136e-10
Dept3        0.21427076 0.03090740   6.932668 4.129780e-12
Dept4        0.06663476 0.03272311   2.036321 4.171810e-02
Dept5       -0.23802565 0.03702165  -6.429363 1.281395e-10
Gender1      0.19143420 0.01513732  12.646505 1.169626e-36
```