Exercises for Day 3 – Solution
Applied Statistics & Statistical methods for SCIENCE

Anton Rask Lundborg

December 2023

Exercise 3.1 Data representation & Validation of a probit analysis

Problem

The purpose of this exercise is to compare three different representations of the same dataset. Read the three text files `beetle.txt`, `beetle_long.txt`, and `beetle_verylong.txt` into R using the commands

```
beetle_A <- read.delim("beetle.txt")
beetle_B <- read.delim("beetle_long.txt")
beetle_C <- read.delim("beetle_verylong.txt")
```

Have a look at them, e.g. by clicking on them in the *Environment* window, in order to verify that they encode the same dataset. Suppose we want to fit the probit model presented in the lecture to the dataset. Replace the *?*-signs by the appropriate data frame (either `beetle_A`, `beetle_B` or `beetle_C`) in the following calls to glm() in order to achieve this:

```
glm(cbind(y, n-y) ~ x, data = ?, family = binomial(link = "probit"))
glm(factor(status) ~ x, weights = count, data = ?, family = binomial(link = "probit"))
glm(factor(status) ~ x, data = ?, family = binomial(link = "probit"))
```

An additional question: What is the purpose of the `weights`-option?

Above you fitted a probit analysis to three different organizations of the same dataset. Answer and discuss the following questions:

- Suppose that the correct models for each dataset are called `m_A`, `m_B`, and `m_C` (for `beetle_A`, `beetle_B` and `beetle_C`, respectively). Verify that the same parameter estimates (and associated p-values) are found in all three models, e.g. by executing the R code:

  ```
  summary(m_A)
  summary(m_B)
  summary(m_C)
  ```

- Validate the three models using cumulative residuals. This may be done using the R code:

  ```
  library(gof)
  plot(cumres(m_A))
  plot(cumres(m_B))
  plot(cumres(m_C))
  ```

  Does the validity of the model depend on the organization of the dataset?

Remark: Previously version 0.9.1 of the `gof`-package was available on the CRAN. In that version the validation was only correctly implemented for the `very long'' dataset, that is \verb+beetle_C+. In version 0.9.2 the bug was fixed for the`binomial'' representation, that is `beetle_A`, and if you use the `cumres()` function on models with a `weight` option, as needed for `beetle_B`, then you get an error message stating that the weight-option is not supported.

Presently the status of the package is as follows:

- The `gof`-package has been removed from the CRAN,
- Version 1.0.1 of the `gof`-package is available from GitHub.
- Apparently the `cumres()` function gives an error if the used dataset is too large. This must be due to a bug in the program, which is rather unfortunate.

To install packages from GitHub you must have the `devtools`-package (may be installed from CRAN) and Windows users also need Rtools (available from https://cran.r-project.org/bin/windows/Rtools/). Thereafter you may install from GitHub like this:

```
  library(devtools)
  install_github("kkholst/gof")
```

Try to see if this works on your laptop. If this fails, then do not despair! You can omit this methodology from yuor work. Actually, not using cumulated residuals was the state of the art until 20 years ago (the paper that introduced cumulated residuals for categorical regression models is from 2002).

Solution

We load the data:

```
beetle_A <- read.delim("beetle.txt")
beetle_B <- read.delim("beetle_long.txt")
beetle_C <- read.delim("beetle_verylong.txt")
```

The correct calls are

```
m_A <- glm(cbind(y, n - y) ~ x, data = beetle_A,
           family = binomial(link = "probit"))
m_B <- glm(factor(status) ~ x, weights = count, data = beetle_B,
           family = binomial(link = "probit"))
m_C <- glm(factor(status) ~ x, data = beetle_C,
           family = binomial(link = "probit"))
```

The dataframes `beetle_B` and `beetle_C` have a binary representation of the datasets where the response variable `status` is dead or alive. Be aware that `glm` models the probability of the last level of the response variables. In this case, the probability of `dead` (as dead precedes alive alphabetically).

The `factor` in `m_B` and `m_C` is needed since `read.delim()` loads the `status` variable as a character string. This converts `status` into a factor which is required for `glm`. The primary difference between `beetle_B` and `beetle_C` is that the former contains a `count` variable that provides the number of these observations. Such a variable is often called a frequency variable and should be specified in the `weights` option.

The data frame `beetle_A` provides a binomial representation of the data sets. The response should be given by binding number of successes (here 'dead') together with number of failures (here 'alive) using the `cbind` function.

We verify that the parameter estimates and p-values are identical:

```
summary(m_A)
```

```
##
## Call:
## glm(formula = cbind(y, n - y) ~ x, family = binomial(link = "probit"),
##     data = beetle_A)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5714  -0.4703   0.7501   1.0632   1.3449
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -34.935      2.648  -13.19   <2e-16 ***
## x             19.728      1.487   13.27   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 284.20  on 7  degrees of freedom
## Residual deviance:  10.12  on 6  degrees of freedom
## AIC: 40.318
##
## Number of Fisher Scoring iterations: 4
```

```
summary(m_B)
```

```
##
## Call:
## glm(formula = factor(status) ~ x, family = binomial(link = "probit"),
##     data = beetle_B, weights = count)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -7.2009  -4.5512   0.6233   5.0666   6.6913
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -34.935      2.648  -13.19   <2e-16 ***
## x             19.728      1.487   13.27   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 645.44  on 14  degrees of freedom
## Residual deviance: 371.36  on 13  degrees of freedom
## AIC: 375.36
##
## Number of Fisher Scoring iterations: 5
```

```
summary(m_C)
```

```
##
## Call:
## glm(formula = factor(status) ~ x, family = binomial(link = "probit"),
##     data = beetle_C)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5608  -0.6275   0.1609   0.4500   2.3943
##
```

```
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -34.935      2.648  -13.19   <2e-16 ***
## x             19.728      1.487   13.27   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 645.44  on 480  degrees of freedom
## Residual deviance: 371.36  on 479  degrees of freedom
## AIC: 375.36
##
## Number of Fisher Scoring iterations: 6
```

We can also apply the Goodness-of-Fit tests using cumulative residuals with the `gof`-package (except the for model B since the `weights` option is not supported). The first row contains model validation for `m_A`, the second for `m_C`.

```
library(gof)
```

```
## Loading required package: lava
```

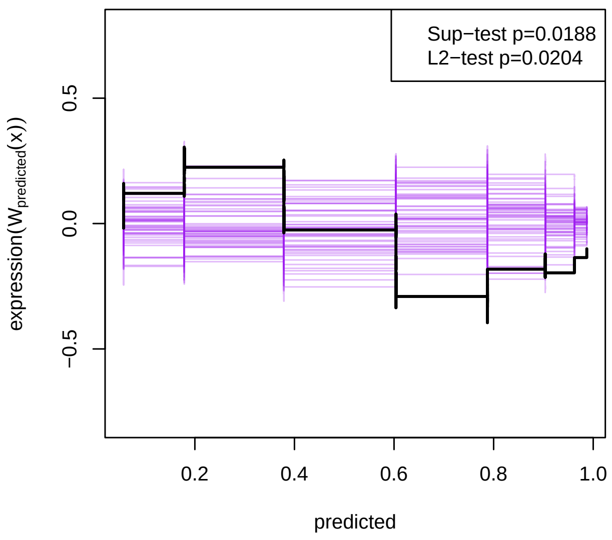
```
##
## Attaching package: 'lava'
```

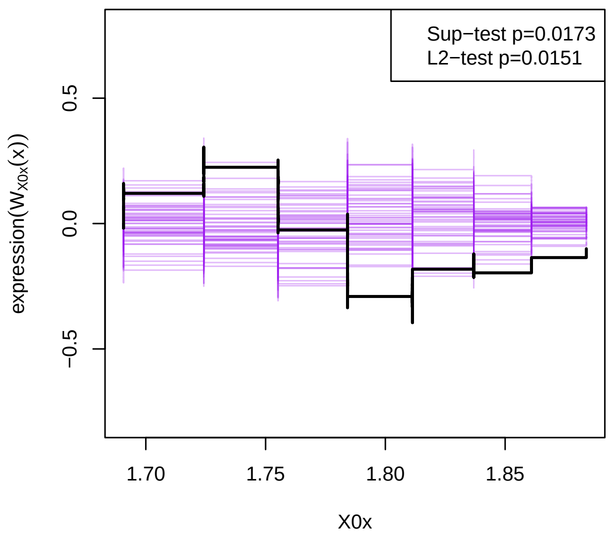
```
## The following object is masked from 'package:dplyr':
##
##     vars
```

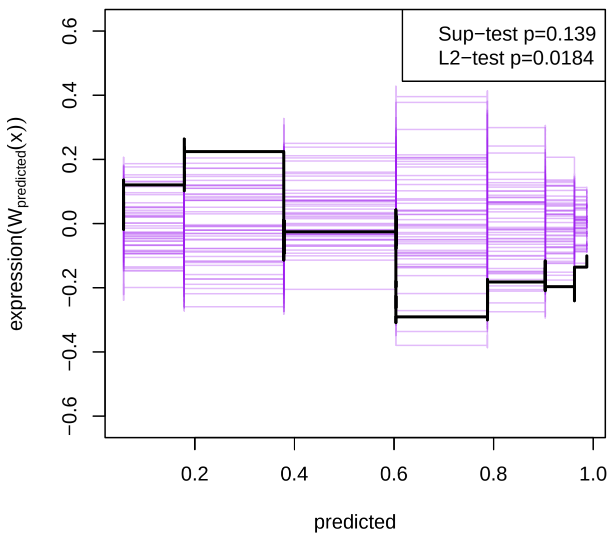
```
## The following object is masked from 'package:ggplot2':
##
##     vars
```

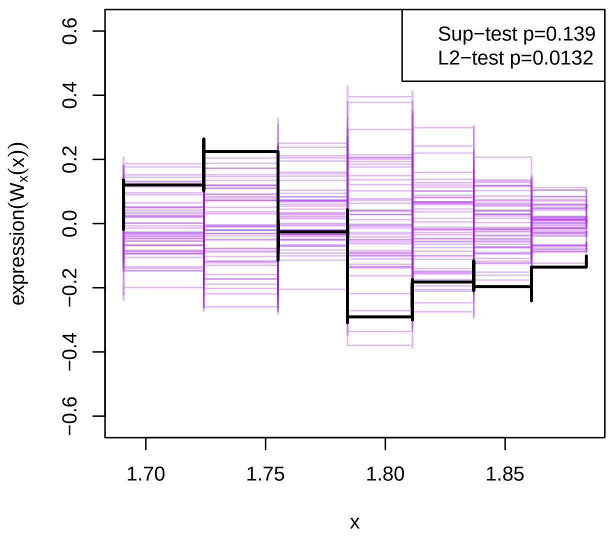
```
## Loading 'gof' version 1.0.1
```

```
par(mfrow = c(2, 2))
plot(cumres(m_A, R = 10000))
plot(cumres(m_C, R = 10000))
```









```
par(mfrow = c(2, 2))
```

The conclusions are similar.

Exercise 3.2 Proportional odds model for the chicken gait score example

Problem

The purpose of this exercise is to reanalyse the chicken gait score example from Day 2 (see Day 2 lecture slides 41–47) using the proportional odds model (see Day 3 lecture slides 42–46). The R script `exercise3_2.R` contains the dataset, the preparation of the dataset and repetition from Day~2, and the exercise questions. Execute the lines in the R script one by one and answer the questions.

Solution

We first load the data (with the code that was provided):

```
library(tidyverse)
```

```
activity <- matrix(c(
  12, 26, 20, 12,
  13, 27, 22, 13,
  25, 25, 18, 8,
  28, 23, 21, 3
), 4, 4, byrow = TRUE)
rownames(activity) <- paste("Treatment", c("A", "B", "C", "D"))
colnames(activity) <- c("0", "1", "2", "3.5")
activity_long <- activity %>%
  as_tibble(rownames = "treat") %>%
  pivot_longer(-"treat", names_to = "gait", values_to = "n") %>%
  uncount(n) %>%
  mutate(treat = factor(treat), gait = factor(gait))
```

We now fit a multinomial regression model:

```
library(ordinal)
```

```
##
## Attaching package: 'ordinal'
```

```
## The following object is masked from 'package:dplyr':
##
##     slice
```

```
m0 <- clm(gait ~ 1, nominal = ~treat, data = activity_long)
```

We also fit a proportional odds model:

```
m1 <- clm(gait ~ treat, data = activity_long)
```

We do a lack-of-fit test:

```
anova(m1, m0)
```

```
## Likelihood ratio tests of cumulative link models:
##
##    formula:     nominal: link: threshold:
## m1 gait ~ treat ~1       logit flexible
## m0 gait ~ 1     ~treat   logit flexible
##
##    no.par    AIC  logLik LR.stat df Pr(>Chisq)
## m1      6 785.71 -386.86
## m0     12 791.91 -383.96   5.799  6     0.4461
```

We see, that the null hypothesis that the proportional odds assumption is true is not rejected ($p = 0.4461$). Thus, we can use the proportional odds model.

We can test the effect of `gait` by using the `drop1` function:

```
drop1(m1, test = "Chisq")
```

```
## Single term deletions
##
## Model:
## gait ~ treat
##        Df    AIC    LRT Pr(>Chi)
## <none>    785.71
## treat   3 792.88 13.166 0.004292 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We get a p-value of the same order as the Kruskal-Wallis test but this model is much more interpretable!

We can test whether it is reasonable to use `treat` as a numeric variable (with values A=1, B=2, C=3, D=4) in the proportional odds model for 'gait':

```
m2 <- clm(gait ~ as.numeric(treat), data = activity_long)
anova(m2, m1)
```

```
## Likelihood ratio tests of cumulative link models:
##
##    formula:                 link: threshold:
## m2 gait ~ as.numeric(treat) logit flexible
## m1 gait ~ treat             logit flexible
##
##    no.par    AIC  logLik LR.stat df Pr(>Chisq)
## m2      4 783.45 -387.72
## m1      6 785.71 -386.86  1.7369  2     0.4196
```

We see that the null hypothesis that `treat` is a numerical variable is not rejected ($p = 0.4196$). Thus, if we want we may choose to use `treat` as a numerical variable in the subsequent analysis. We can again test for the effect of the numeric `treat`:

```
drop1(m2, test = "Chisq")
```

```
## Single term deletions
##
## Model:
## gait ~ as.numeric(treat)
##                   Df    AIC    LRT  Pr(>Chi)
## <none>               783.45
## as.numeric(treat)  1 792.88 11.429 0.0007232 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We get power comparable to the test using Spearman's rank correlation but again we get a much more interpretable model. We perform pairwise comparisons on `m1` using the `emmeans` package.

```
library(emmeans)
pairs(emmeans(m1, ~treat))
```

```
##  contrast                  estimate    SE  df z.ratio p.value
##  Treatment A - Treatment B  -0.0197 0.299 Inf  -0.066  0.9999
##  Treatment A - Treatment C   0.6372 0.303 Inf   2.106  0.1512
##  Treatment A - Treatment D   0.8516 0.303 Inf   2.806  0.0258
##  Treatment B - Treatment C   0.6569 0.298 Inf   2.205  0.1218
##  Treatment B - Treatment D   0.8712 0.299 Inf   2.916  0.0186
##  Treatment C - Treatment D   0.2143 0.299 Inf   0.717  0.8903
##
## P value adjustment: tukey method for comparing a family of 4 estimates
```

We see that using `treat` as categorical permits a precise analysis on the differences between the 4 treatments. Using `treat` as a numerical variable gives higher power in the effect test but perhaps we lose something in terms of interpretability.

Exercise 3.3 Logistic regression

Problem

The data table below shows the result of the glutaraldehyde coagulation test (GLA) on 420 cows from 10 Danish dairy herds. The GLA-test is positive or negative and it is believed that the test may reflect inflammatory disease in the animal. To investigate this hypothesis the result of the test was compared with a score from a clinical examination of the animal concentrating on udder, limbs and external physical injuries. This score was 0, 1, 2 or 3 increasing with severity (infection status). The data table shows the number of cows from the different herds with the eight possible combinations of GLA-test result and clinical score.

Clinical score	0		1		2		3	
GLA	−	+	−	+	−	+	−	+
Herd 1	26	6	4	4	2	1	1	0
2	23	9	5	1	1	1	1	2
3	3	3	5	11	4	0	4	13
4	20	18	4	1	3	1	0	4
5	24	8	3	1	0	0	0	1
6	0	0	5	5	10	7	5	9
7	0	8	5	9	10	6	5	1
8	2	0	4	1	12	4	8	6
9	1	0	0	0	14	7	11	7
10	0	0	1	2	4	16	1	16

Use a logistic linear model to investigate how the GLA-test result relates to the clinical score and to the herd (dataset is available in the text file `GLA.txt`). Estimate an odds-ratio for being GLA-positive for animals with clinical score 1 (respectively 2 and 3) relative to those with clinical score 0.

If you are stuck, consult the hints below:

1. The explanatory variables `herd` and `clin` may be used as categorical variables (instead of continuous variables) by appropriately using the `factor()` inside the model formula.
2. If you use `factor(clin)` as a main effect in the model, then the associated parameters will be log(odds ratio) against clinical score=0.
3. Remember to backtransform the parameter estimates by the exponential function! Why?

(Data from project report by Trine T{ø}lb{ø}ll (1999): Use of Glutaraldehyde test as an indicator of inflammatory diseases in dairy herds, KVL.)

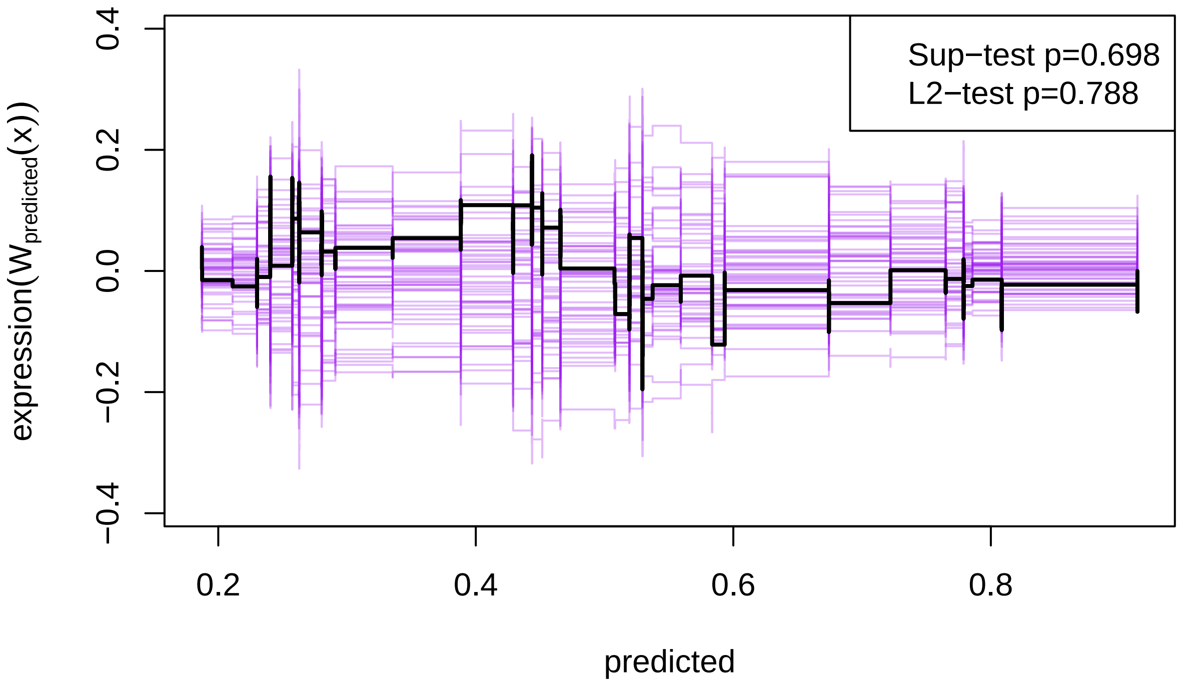
Solution

We load the data and fit a logistic regression:

```
gla <- read.delim("GLA.txt")
m1 <- glm(cbind(positive, negative) ~ factor(herd) + factor(clin),
  data = gla, family = binomial
)
```

To validate the model, we use cumulative residuals:

```
library(gof)
plot(cumres(m1, R = 10000))
```



The model seems valid from the cumulative residuals. We test for effect of `herd` and `clin`:

```
drop1(m1, test = "Chisq")
```

```
## Single term deletions
##
## Model:
## cbind(positive, negative) ~ factor(herd) + factor(clin)
##              Df Deviance    AIC    LRT  Pr(>Chi)
## <none>            30.050 132.34
## factor(herd)  9   76.712 161.01 46.662 4.537e-07 ***
## factor(clin)  3   43.354 139.65 13.304  0.004023 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Both effects are highly significant with `herd` having a p-value of the order 10^{-7} and `clin` of the order 10^{-1}. We can get estimates and confidence intervals for odds-ratios (remembering to back-transform using `exp`):

```
exp(cbind(OR = coef(m1), confint(m1)))
```

```
## Waiting for profiling to be done...
```

```
##                         OR     2.5 %     97.5 %
## (Intercept)      0.3166962 0.1492058  0.6222535
## factor(herd)2    1.2300104 0.4748104  3.2291469
## factor(herd)3    3.4144303 1.2781352  9.5538582
## factor(herd)4    2.5187150 1.0584879  6.2668179
## factor(herd)5    1.1260654 0.4074258  3.0941156
## factor(herd)6    2.0068265 0.7096095  5.8609880
## factor(herd)7    3.5555890 1.4003805  9.4704194
## factor(herd)8    0.7285187 0.2354669  2.2378960
## factor(herd)9    0.8450058 0.2780743  2.5946457
## factor(herd)10 10.2869365 3.1243291 37.9667638
## factor(clin)1    0.9989646 0.5046004  1.9478748
## factor(clin)2    1.2957088 0.5927539  2.8312000
## factor(clin)3    3.2565999 1.4903271  7.2652412
```

We can present the `clin` results nicely by rounding:

```
round(exp(cbind(OR = coef(m1), confint(m1))[11:13, ]), digits = 4)
```

```
## Waiting for profiling to be done...
```

```
##                     OR  2.5 % 97.5 %
## factor(clin)1 0.9990 0.5046 1.9479
## factor(clin)2 1.2957 0.5928 2.8312
## factor(clin)3 3.2566 1.4903 7.2652
```

We get odds-ratios of 0.999, 1.296 and 3.257 for the odds-ratios of groups 1, 2 and 3 versus 0, respectively. We could also have use the `emmeans` package to obtain these:

```
library(emmeans)
confint(pairs(emmeans(m1, ~clin), type = "response", reverse = TRUE))
```

```
##  contrast      odds.ratio    SE  df asymp.LCL asymp.UCL
##  clin1 / clin0      0.999 0.343 Inf     0.413      2.41
##  clin2 / clin0      1.296 0.515 Inf     0.467      3.60
##  clin2 / clin1      1.297 0.486 Inf     0.495      3.40
##  clin3 / clin0      3.257 1.311 Inf     1.157      9.16
##  clin3 / clin1      3.260 1.249 Inf     1.219      8.72
##  clin3 / clin2      2.513 0.852 Inf     1.052      6.00
##
## Results are averaged over the levels of: herd
## Confidence level used: 0.95
## Conf-level adjustment: tukey method for comparing a family of 4 estimates
## Intervals are back-transformed from the log odds ratio scale
```

The results can be seen in rows 1, 2 and 4.

Exercise 3.4 Proportional odds model

Problem

In a survey of the usage of *Oslomarka* (recreational area around Oslo) 365 people were classified by how often they walk in the forest and how far they walk. The dataset listed below is taken from (Haakenstad, 1975):

	Walking distance in km					
Frequency of walks	≤ 2.5	2.5–5	5–10	10–20	≥ 20	Total
F1: Each week	14	29	80	56	16	195
F2: Each month	9	22	30	17	4	82
F3: Sometimes during the season	24	23	30	9	2	88
Total	47	74	140	82	22	365

The dataset is available in the text file `walks.txt`. Please do the following statistical analyses and present their conclusions:

- An ordinal logistic regression of distance on frequency. Is the proportional odds assumption valid? Provide estimates and confidence intervals for the odds-ratio of walking shorter distances relative to frequency group F1.

  Remark: If use want to use `distance` as the response in an ordinal regression, then this variable should be encoded as a factor. This can be done directly in the call to `clm()`:

  ```
  clm(factor(distance) ~ frequency, data = walks, weights = count)
  ```

- An ordinal logistic regression of frequency on distance. Try to use `distance` both as a factor and as a continuous covariate[1]. Is the proportional odds assumption valid? Provide estimates and confidence intervals for the odds-ratio of walking less frequently when walking distance is increased.

  Remark: The variable `frequency` is already encoded as a factor in the data frame, so it can be used as a response in `clm()` without any further ado. However, the following code doing a *multinomial regression* of `frequency` on the numerical variable `distance` used as a categorical factor[2] does not work on my laptop:

  ```
  clm(frequency ~ 1, nominal = ~factor(distance), data = walks, weights = count)
  ```

  This must be seen as a bug in the `ordinal`-package. Luckily there is a fix, namely

  ```
  clm(frequency ~ 1, nominal = ~factor(walks$distance), data = walks, weights = count)
  ```

- What are the differences between the three statistical analysis done above, e.g. in their interpretation and in their power to falsify the hypothesis of no association?

Solution

We load the data and examine it:

```
walks <- read.delim("walks.txt")
str(walks)
```

```
## 'data.frame':    15 obs. of  3 variables:
##  $ frequency: chr  "F1" "F1" "F1" "F1" ...
##  $ distance : num  1 3.5 7.5 15 25 1 3.5 7.5 15 25 ...
##  $ count    : int  14 29 80 56 16 9 22 30 17 4 ...
```

We see that `frequency` is loaded as a character but to use it in the modelling to come, we need it to be a factor. We convert it:

```
walks$frequency <- factor(walks$frequency)
```

We first model distance on frequency and do a lack-of-fit test of the proportional odds assumption:

```
m0 <- clm(factor(distance) ~ 1,
  nominal = ~frequency, data = walks,
  weights = count
)
m1 <- clm(factor(distance) ~ frequency, data = walks, weights = count)
anova(m1, m0)
```

```
## Likelihood ratio tests of cumulative link models:
##
##    formula:                     nominal:   link: threshold:
## m1 factor(distance) ~ frequency ~1         logit flexible
## m0 factor(distance) ~ 1         ~frequency logit flexible
##
##    no.par    AIC  logLik LR.stat df Pr(>Chisq)
## m1      6 1041.2 -514.60
## m0     12 1051.1 -513.54  2.1307  6     0.9073
```

The proportional odds assumption is valid, and we proceed with effect testing:

```
drop1(m1, test = "Chisq")
```

[1]Possibly on a logarithmic scale – the reason that it might be a good idea to use the continuous covariate on this scale is that the odds are also modelled on a log scale!

[2]The multinomial regression is needed as the reference model in the Lack-of-Fit test for the proportional odds assumption.

```
## Single term deletions
##
## Model:
## factor(distance) ~ frequency
##           Df    AIC    LRT  Pr(>Chi)
## <none>       1041.2
## frequency  2 1073.6 36.433 1.227e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see that `frequency` is highly significant (p of the order 10^{-8}). We summarize the effects of higher frequency on the probability of walking shorter distances:

```
exp(cbind("OR vs F1" = coef(m1)[5:6], confint(m1)[]))
```

```
##              OR vs F1     2.5 %    97.5 %
## frequencyF2 0.5365125 0.3338243 0.8595484
## frequencyF3 0.2378911 0.1467453 0.3824607
```

We now model frequency on distance and check the proportional odds assumption:

```
m0 <- clm(frequency ~ 1,
  nominal = ~ factor(walks$distance), data = walks,
  weights = count
)
m1 <- clm(frequency ~ factor(distance), data = walks, weights = count)
anova(m1, m0)
```

```
## Likelihood ratio tests of cumulative link models:
##
##    formula:                     nominal:                link: threshold:
## m1 frequency ~ factor(distance) ~1                      logit flexible
## m0 frequency ~ 1                ~factor(walks$distance) logit flexible
##
##    no.par    AIC  logLik LR.stat df Pr(>Chisq)
## m1      6 715.61 -351.81
## m0     10 721.18 -350.59  2.4338  4     0.6565
```

The proportional odds assumption is valid, and we proceed with effect testing

```
drop1(m1, test = "Chisq")
```

```
## Single term deletions
##
## Model:
## frequency ~ factor(distance)
##                  Df    AIC    LRT  Pr(>Chi)
## <none>              715.61
## factor(distance)  4 743.74 36.129 2.722e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Effect of distance is highly significant. Let us see if the categorical distance may be used as a continuous variable instead by testing against the proportional odds model:

```
m2 <- clm(frequency ~ distance, data = walks, weights = count)
anova(m2, m1)
```

```
## Likelihood ratio tests of cumulative link models:
##
```

```
##    formula:                     link: threshold:
## m2 frequency ~ distance         logit flexible
## m1 frequency ~ factor(distance) logit flexible
##
##    no.par    AIC  logLik LR.stat df Pr(>Chisq)
## m2      3 716.19 -355.09
## m1      6 715.61 -351.81   6.574  3    0.08679 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We get a valid model fit (by testing) but we increase AIC and the p-value is relatively small (around 0.08). We try distance on the logarithmic scale instead:

```
m3 <- clm(factor(frequency) ~ log(distance), data = walks, weights = count)
anova(m3, m1)
```

```
## Likelihood ratio tests of cumulative link models:
##
##    formula:                           link: threshold:
## m3 factor(frequency) ~ log(distance) logit flexible
## m1 frequency ~ factor(distance)      logit flexible
##
##    no.par    AIC  logLik LR.stat df Pr(>Chisq)
## m3      3 710.10 -352.05
## m1      6 715.61 -351.81   0.489  3     0.9213
```

Now the model fit looks much nicer! We can test the effect of distance as a continuous variable (on the log-scale):

```
drop1(m3, test = "Chisq")
```

```
## Single term deletions
##
## Model:
## factor(frequency) ~ log(distance)
##               Df    AIC   LRT  Pr(>Chi)
## <none>           710.10
## log(distance)  1 743.74 35.64 2.373e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This gives even smaller p-values than above. We can give the estimated odds ratio for walking less frequently when the person usually walks twice as long:

```
exp(log(2) * cbind(estimate = coef(m3)[3], confint(m3)))
```

```
##                estimate     2.5 %    97.5 %
## log(distance) 0.6206576 0.5267625 0.7277036
```

Exercise 3.5 Poisson regression

Problem

In an investigation of the soil profile at *Mejlbjerg Hoved* the number of fine gravel particles categorized in the 4 groups *crystalline*, *sediment grains*, *chalcedony chert*, and *quartz grains* were counted in three depth levels at two different locations. The dataset listed below was taken from (Bl{æ}sild and Granfeldt, 1995):

depth:	4–11 meters		11–14 meters		18–20 meters	
location:	A	B	A	B	A	B
Crystalline	207	205	87	104	159	142
Sediment grains	48	61	23	14	19	12
Chalcedony chert	15	12	46	64	94	100
Quartz grains	30	28	133	127	41	51

The two locations (A and B) serve as blocks and should not be used in interactions. But the interaction of particle type and depth may be relevant. Perform a Poisson regression of the dataset (available in `particles.txt`) and report the estimate and its confidence interval of the relative risk of *sediment grains* against *quartz grains* at 11–14 meters. Consider also the following questions:

- What is the interpretation of the relative risk requested above? Could you think of a more easy way of computing this estimate?
- If the estimate for the relative risk can be computed directly from the dataset, what has then been achieved by the Poisson regression?

If you are unsatisfied with the model validity you might try the option `family=quasipoisson` in the `glm()`-call, cf. Section 10.1.2 in the R guide.

(This exercise was conceived on the basis of exercise 10.1 in Bo Martin Bibby: "Noter til Regressionsanalyse'' (in danish).)

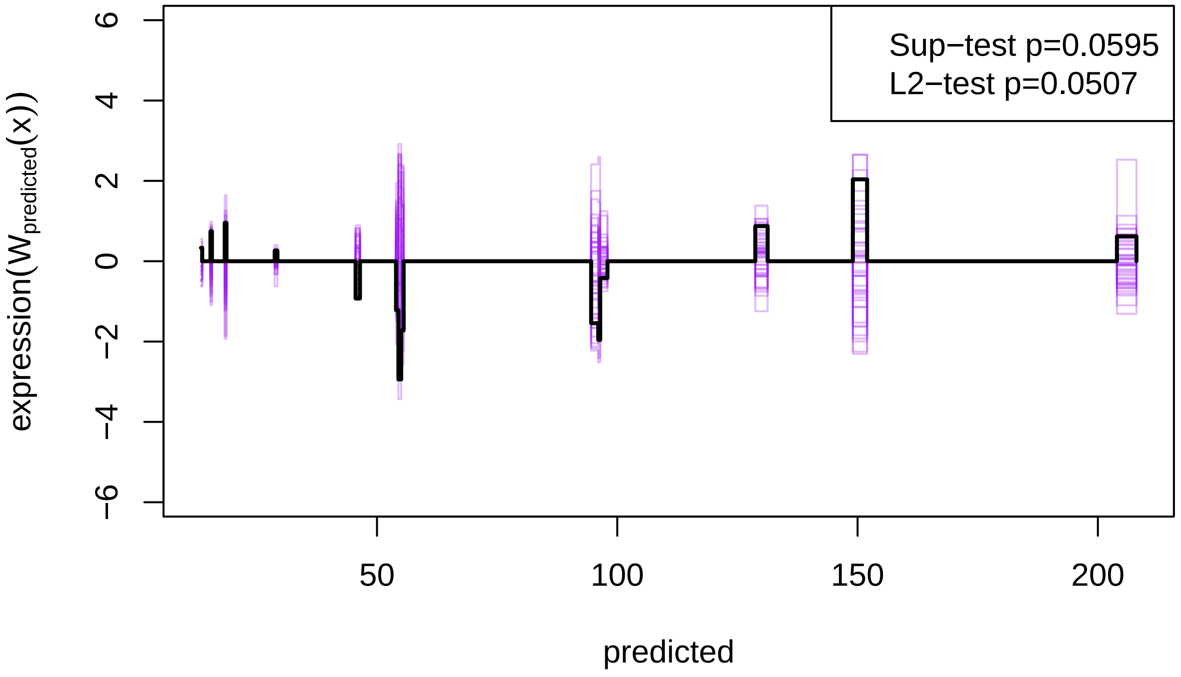
Solution

We load the data and perform a logistic regression:

```
particles <- read.delim("particles.txt")
m1 <- glm(count ~ location + type * depth, data = particles, family = poisson())
```

We validate the model using cumulative residuals:

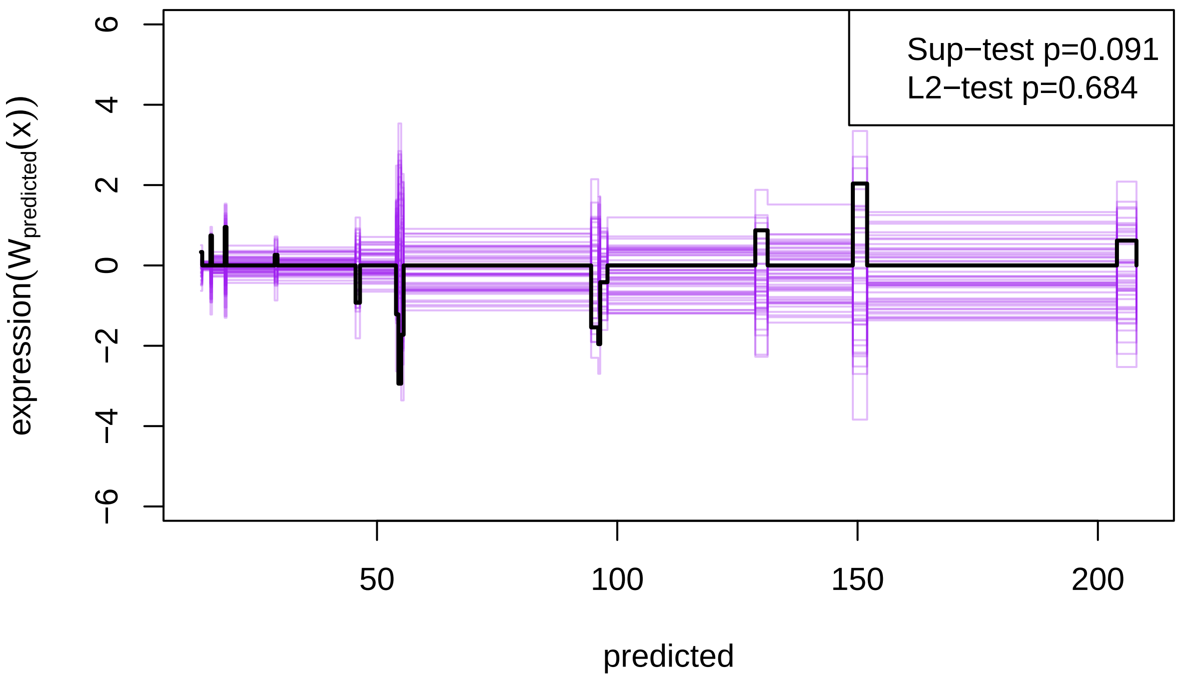
```
library(gof)
plot(cumres(m1, R = 10000))
```



The

model is barely valid. We can perhaps solve this by allowing for more variance than what is native to the Poisson distribution by using a `quasipoisson` family:

```
m2 <- glm(count ~ location + type * depth,
  data = particles,
  family = quasipoisson()
)
plot(cumres(m2, R = 10000))
```



This has improved the model fit from p around 0.05 to around 0.07. We can test for effects:

```
drop1(m2, test = "Chisq")
```

```
## Single term deletions
##
## Model:
## count ~ location + type * depth
##            Df Deviance scaled dev. Pr(>Chi)
## <none>           12.44
## location    1    12.62        0.16   0.6911
## type:depth  6   458.88      396.50   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The interaction between `type` and `depth` is highly significant, i-e. the occurence of the particle types vary with the depth. Despite `location` not being significant ($p = 0.6911$), we keep it in the model. We can extract parameter estimates and confidence intervals:

```
cbind(estimate = coef(m1), confint(m1))
```

```
## Waiting for profiling to be done...
```

```
##                                          estimate       2.5 %     97.5 %
## (Intercept)                            2.59276131  2.18775248  2.9508246
## locationB                              0.01975915 -0.07208172  0.1116278
## typeCrystalline                        2.72518648  2.35652794  3.1383454
## typeQuartz_grains                      0.76460614  0.31865340  1.2360639
## typeSediment_grains                      1.39551102  0.99035505  1.8365079
## depth11-14 meters                        1.40464350  0.99992417  1.8453103
## depth18-20 meters                        1.97202129  1.58830378  2.3967294
## typeCrystalline:depth11-14 meters       -2.17339342 -2.64520120 -1.7329899
## typeQuartz_grains:depth11-14 meters      0.09559512 -0.42392621  0.5957368
## typeSediment_grains:depth11-14 meters -2.48507347 -3.06366901 -1.9366965
## typeCrystalline:depth18-20 meters       -2.28593438 -2.73474020 -1.8733878
## typeQuartz_grains:depth18-20 meters     -1.51067573 -2.04256309 -1.0001937
## typeSediment_grains:depth18-20 meters -3.22938197 -3.81495438 -2.6791593
```

To get the relative risk, we again use `emmeans`:

```
confint(pairs(emmeans(m2, ~ type | depth), reverse = TRUE),
  adjust = "none", type = "response"
)
```

```
## depth = 04-11 meters:
##  contrast                            ratio     SE  df asymp.LCL asymp.UCL
##  Crystalline / Chalcedony_chert     15.259 3.2166 Inf   10.0950    23.065
##  Quartz_grains / Chalcedony_chert    2.148 0.5311 Inf    1.3232     3.487
##  Quartz_grains / Crystalline         0.141 0.0209 Inf    0.1052     0.188
##  Sediment_grains / Chalcedony_chert  4.037 0.9209 Inf    2.5817     6.313
##  Sediment_grains / Crystalline       0.265 0.0302 Inf    0.2115     0.331
##  Sediment_grains / Quartz_grains     1.879 0.3241 Inf    1.3403     2.635
##
## depth = 11-14 meters:
##  contrast                            ratio     SE  df asymp.LCL asymp.UCL
##  Crystalline / Chalcedony_chert      1.736 0.2205 Inf    1.3537     2.227
##  Quartz_grains / Chalcedony_chert    2.364 0.2853 Inf    1.8657     2.994
##  Quartz_grains / Crystalline         1.361 0.1377 Inf    1.1165     1.660
##  Sediment_grains / Chalcedony_chert  0.336 0.0678 Inf    0.2265     0.499
##  Sediment_grains / Crystalline       0.194 0.0369 Inf    0.1333     0.281
##  Sediment_grains / Quartz_grains     0.142 0.0265 Inf    0.0987     0.205
##
## depth = 18-20 meters:
##  contrast                            ratio     SE  df asymp.LCL asymp.UCL
##  Crystalline / Chalcedony_chert      1.552 0.1516 Inf    1.2812     1.879
##  Quartz_grains / Chalcedony_chert    0.474 0.0637 Inf    0.3645     0.617
##  Quartz_grains / Crystalline         0.306 0.0386 Inf    0.2386     0.392
##  Sediment_grains / Chalcedony_chert  0.160 0.0328 Inf    0.1069     0.239
##  Sediment_grains / Crystalline       0.103 0.0206 Inf    0.0696     0.152
##  Sediment_grains / Quartz_grains     0.337 0.0743 Inf    0.2188     0.519
##
## Results are averaged over the levels of: location
## Confidence level used: 0.95
## Intervals are back-transformed from the log scale
```

This gives a relative risk estimate of 0.142 with a confidence interval of $(0.0987, 0.205)$. The relative risk gives the proportion of sediment grains relative to quartz grains. We could have computed this immediately from the given table:

```
(23 + 14) / (133 + 127)
```

```
## [1] 0.1423077
```

However, this estimate comes without confidence intervals or other forms of uncertainty quantification.