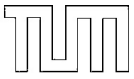

Technische Universität München

Department of Mathematics

Bachelor's Thesis

Fitting generalized linear models in R

Daniela Pohl

Supervisor: Prof. Claudia Czado, Ph.D.

Advisor: Dipl.-Math. oec. Ulf Schepsmeier

Submission Date: 2013-07-31

I hereby declare that I have written the Bachelor's Thesis on my own and have used no other than the stated sources and aids.

Munich,

Zusammenfassung

In dieser Bachelorarbeit wird die Anpassung von generalisierten linearen Modellen in der statistischen Programmiersprache `R` mit Hilfe parametrischer Linkfunktionen beschrieben. Zunächst werden die mathematischen Grundlagen dargestellt. Dafür wird eine Einführung in die Theorie der generalisierten linearen Modelle gegeben und parametrische Linkfunktionen definiert. Darauf basierend werden dann der Maximum Likelihood Schätzer für diese Modelle und die Deviance, eine Methode für das Goodness-of-Fit-Maß, erklärt. Anschließend werden Beispieldatensätze sowohl für den Gauss-verteilten Fall als auch für Binomial- und Poisson-Verteilung vorgestellt. In einer ersten Analyse wird die `R`-eigene `glm` Funktion auf diese Daten angewandt, die ein generalisiertes lineares Modell mit Standard-Linkfunktion erstellt. Dieses Modell ist für die hier vorgestellten Datensätze nicht ausreichend, da vor allem der linke oder rechte Rand der Modelldaten nicht mit den beobachteten Werten übereinstimmt. Deshalb werden im Hauptteil der Arbeit sogenannte Tail-Transformationen beschrieben, die eine Kurve am linken oder rechten Rand oder an beiden Rändern leicht transformieren. Diese Funktionen werden dann in die Linkfunktion eines generalisierten linearen Modells eingebaut. Ich beschreibe in diesem Teil der Bachelorarbeit vier `glm`-Funktionen: Die `glm.model` erstellt ein generalisiertes lineares Modell mit einer solchen Tail-Transformation, während die `glm.profile` die optimalen Parameter der Tail-Transformation für die eingegebenen Daten findet. `glm.fitted` erstellt zwei Plots von modellierten und beobachteten Daten einmal von der Standard-Regression in `R` und einmal vom neuem GLM mit Tail-Transformation für einen direkten Vergleich. Die letzte Funktion `glm.inf` gibt Standardabweichungen sowohl von den Regressions- parametern als auch den Linkparametern, und eine Korrelationstabelle aller Parameter aus. Im Schlussteil wende ich diese vier neuen `glm`-Funktionen auf die zuvor vorgestellten Daten an und zeige die Verbesserungen an den Modellen auf.

Contents

List of Figures

List of Tables

1 Preface

The content of this thesis is divided into four parts. The introductional chapter defines the mathematical properties of generalized linear models and the theory of applying a parametric link to these models. Furthermore it presents the data sets that will be used as examples throughout the thesis. In the main part the `R` functions with their implementation and their peculiarities will be explained. I will set out their features as well as why I have chosen the specific code lines and what they were meant to yield. The fourth chapter shows how to use these functions on the example data sets. Finally a conclusion will be given of the improvements achieved by the new `R` functions.
`R` is a free software and contains programming language and environment for statistical computing. It was inspired by the `S` environment and is therefore very similar to `S` with some important differences. For more information see `www.r-project.org`.
The concept of these transformations and the idea of using these particular link functions come from Czado [4]. There all functions were already implemented in the statistical programming language `S`. While some of the functions only had to be converted to `R` with small parts added to match the requirements of the coding language of `R`, others like the link generating functions, which create own written link functions that can be applied in the `glm`-function, could not be used. That's why all of my generalized linear model with tail transformation functions contain own maximum likelihood estimations where the new link functions were directly applied.
Additionally all functions were combined together in an `R`-package including the example data sets of this thesis and help files for all functions.
The entire `R` code and the complete data sets used for this thesis can be found in the appendix.

2 Introduction

In this chapter the mathematical theory of generalized linear models will be introduced and a short overview of the data sets will be given.

Regression analyses are very important in applied statistics. The purpose is to estimate a relationship, which is assumed to be functional, between the response variable Y and a set of covariates $X = (X_1, X_2, ..., X_p)$ and in predicting further responses for new values of the covariates. (see Dey et al. [6], p. 217)

But linear models require independent normal errors as well as homogenous variances. Since we are also interested in predicting discrete or non-linear continuous responses, we have to use another method for these analyses. This is where the generalized linear models make an appearance. It is a popular class of regression models for non-normally distributed responses. (see Czado et al. [5], p. 21)

Generalized linear models are used for different kinds of applications and are an important tool for data analysing, because they are flexible models. They involve both logistic regression as well as log-linear models for binomial- and poisson-distributed discrete data and Gauss, Gamma and Inverse Gauss models for continuous data. (see Dey et al. [6], p. 4)

2.1 Purpose of the thesis

However, these generalized linear models are not entirely satisfying when applied to certain data and we need some transformations of the analyses.
This thesis focuses on the method to perform the same type of analyses using parametric links to optimize the display of the relation between the response and the covariate once the usual model dissatisfies or fails. These parametric links have tail transformation functions included and are meant to be applied especially on data sets whose standard generalized linear models do not fit at the tails.
In this Bachelor thesis I describe how generalized linear models can be fitted with parametric link in the statistical programming language `R`.

2.2 Theory of Generalized Linear Models and parametric link functions

2.2.1 Exponential family of distributions

In statistical models we observe measurements made on subjects or objects and want to analyse the relationships between these measurements.
We use the terms response or outcome for measurements that are free to vary, regarded as random variables, and we denote the non-random measurements or observations by the term explanatory variable or predictor variable or covariate. (see Dobson and Barnett [7], p. 1)
A linear model or regression model has the following components:
The response random variables $Y = (Y_1, ..., Y_n)$, which have to be identically and indepently normally distributed with expected value μ and variance σ^2 and the explanatory variates $X_1, ..., X_p$, all of length n, which form the $n \times p$ model matrix.

Generalized Linear Models are a class of statistical models where the response does not have to be normally distributed, but is allowed to follow any distribution from the exponential family, for example Normal, Poisson, Binomial or Gamma.
The exponential family as defined by Wood [12] (p. 62):

Definition 2.2.1 (Exponential family)**.**
A distribution belongs to the exponential family of distributions if its probability density function, or probability mass function, can be written as

$$f_\theta(y) = \exp\Big(\frac{y\theta - b(\theta)}{a(\phi)} + c(y, \phi)\Big),$$

where b, a and c are arbitrary functions, ϕ an arbitrary 'scale' parameter, and θ is known as the 'canonical parameter' of the distribution.

Examples of the exponential family are:
The Normal distribution $\mathcal{N}(\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp(\frac{-(y-\mu)^2}{2\sigma^2})$ with

$$\theta = \mu, \phi = \sigma^2, a(\phi) = \phi, b(\theta) = \frac{\theta^2}{2} \text{ and } c(y, \phi) = -\frac{1}{2}(\frac{y^2}{\phi} + \log(2\pi\phi)),$$

the Binomial distribution $B(n, \frac{\mu}{n}) = \binom{n}{y}(\frac{\mu}{n})^y(1 - \frac{\mu}{n})^{n-y}$ with

$$\theta = \log(\tfrac{\mu}{n-\mu}), \phi = 1, a(\phi) = \phi, b(\theta) = n\log(1 + e^\theta) \text{ and } c(y, \phi) = \log\tbinom{n}{y} \text{ and}$$

the Poisson distribution $\mathcal{P}\text{ois}(\mu) = \frac{\mu^y \exp(-\mu)}{y!}$ with

$$\theta = \log(\mu), \phi = 1, a(\phi) = \phi, b(\theta) = \exp(\theta) \text{ and } c(y, \phi) = -\log(y!).$$

These are just some examples of the many distributions, that belong to the exponential family. Other common distributions are Exponential, Gamma, Inverse Gauss, Beta, Chi-Squared, Dirichlet and Wishart. All these distributions can be used in a generalized linear model.

2.2.2 Generalized Linear Models

A generalized linear model consists of three model components, the *random, systematic* and *link component*. For the following definition we refer to Czado et al. [5](p. 21-22):

Definition 2.2.2 (Components of a generalized linear model)**.**
1. *Random Component.* Responses Y_i,$1 \leq i \leq n$, are independent with probability density function or probability mass function belonging to the *exponential family*

$$f(y|\theta,\phi) = exp\Big(\frac{y\theta - b(\theta)}{a(\phi)} + c(y,\phi)\Big)$$

Here $\phi > 0$ is a *dispersion parameter* and the functions $b(\cdot), a(\cdot)$ and $c(\cdot)$ are known. The unknown parameter θ is called the *canonical parameter*.

2. *Systematic Component.* The quantity

$$\eta_i(\beta) := x_i^T\beta = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik}$$

is called the *linear predictor* and $\beta = (\beta_0, \ldots, \beta_k)^T$ are p unknown *regression parameters* to be estimated.

3. *Parametric Link Component.* The random and systematic component are related via a link function

$$g(\mu_i) = \eta_i(\beta) = x_i^T\beta$$

which defines the relationship between the linear predictor η_i and the mean μ_i of Y_i.

One can use generalized linear models in `R` by the `glm` function, which takes a `formula` argument to pass response and covariates and a `family` argument to specify the distribution.

2.2.3 Parametric Link Functions

The parametric link function is the third component of a generalized linear model and relates the linear predictor $\eta = X^T\beta$ to μ, the expected value of y.
In linear models the mean and the linear predictor are identical, so for normal distributed responses mostly the identity function is chosen as a link, $g(\mu_i) = \mu_i$. In the Poisson case however we are dealing with counts and must therefore have $\mu > 0$, so we can't take the identity link. That's why for Poisson distributed response often $g(\mu_i) = \log(\mu_i)$ and $G(\eta_i) = g^{-1}(\eta_i) = exp(\eta_i)$ is taken as the link function, because η may be negative, but μ must not be. For Binomial distribution we have the restriction $0 < \mu < 1$ and we have to choose a link that maps the interval $(0, 1)$ on the whole real line. Because of that most often the binomial link is $g(\mu_i) = \log(\frac{\mu_i}{1-\mu_i})$ for a logistic regression and $g(\mu_i) = \Phi^{-1}(\mu_i)$ for a probit model, where Φ is the probability density function of the standard normal distribution. (see McCullagh [10], p. 31)
The `glm` function in `R` works with these most common link functions from above, if one chooses `gaussian, poisson, binomial` or `binomial(probit)`, respectively, as family.

Sometimes it is appropriate to choose a slightly modified link function from a link family $\mathcal{G} = \{g(\cdot|\psi) : \psi \in \Psi\}$:

Error Distribution	**Parameter Restriction**	**Canonical Link G**	**Link Family $\mathcal{G}$**
Normal	$\mu \in \mathbb{R}$	$G(\eta) = \eta$	$G(\eta, \psi) = h(\eta, \psi)$
Binomial (Logit)	$\mu \in (0, 1)$	$G(\eta) = \frac{\exp(\eta)}{1+\exp(\eta)}$	$G(\eta, \psi) = \frac{\exp(h(\eta,\psi))}{1+\exp(h(\eta,\psi))}$
Binomial (Probit)	$\mu \in (0, 1)$	$G(\eta) = \Phi(\eta)$	$G(\eta) = \Phi(h(\eta, \psi))$
Poisson	$\mu > 0$	$G(\eta) = \exp(\eta)$	$G(\eta, \psi) = \exp(h(\eta, \psi))$

Table 1: Link families for generalized linear models

In this thesis we use tail transformation functions $h(\eta, \psi)$ to modify the link. These functions will be discussed in section 3.1.

2.2.4 Maximum Likelihood Estimation for Generalized Linear Models

The likelihood method is a key concept in statistics. Given a statistical model $f_\theta(y)$ the likelihood is the probability of the observed data set. We write $L(\theta|y)$ for the likelihood and consider it as a function of θ, which gives the probability how "likely" the parameters θ are for the observed data. We want to estimate the values that maximize this probability. These are called the maximum likelihood estimates.
As our observation set $y = (y_1, y_2, ..., y_n)$ is independently distributed with a distribution from the exponential family, i.e. $y \sim f_y(\theta, \phi) = \exp\left(\frac{y\theta - b(\theta)}{a(\phi)} + c(y, \phi)\right)$, we can write the likelihood function as the product of the distributions:

$$L(\theta, \phi) = f(\theta, \phi|y) = \prod_{i=1}^{n} f_i(\theta, \phi|y_i)$$

and on the logarithm scale this independence gives us a an additive property. We write $l(\theta, \phi|y) = \log f_y(\theta, \phi|y)$ for the loglikelihood function and

$$l(\theta, \phi) = \sum_{i=1}^{n} log f_i(\theta, \phi|y_i)$$

(see Lee [9], p. 5)
The loglikelihood functions for the three distributions discussed in this thesis are:

Distribution	**Loglikelihood Function**
Normal	$l(\beta,\psi,\sigma^2\|y) = -\frac{1}{2\sigma^2}\sum_{i=1}^{n}\big(y_i - h(\eta(\beta),\psi)\big)^2 - \frac{n}{2}\log(\sigma^2)$
Binomial (Logit)	$l(\beta,\psi\|y) = \sum_{i=1}^{n}\Big[y_i \log\big(\frac{\exp(h(\eta(\beta),\psi))}{1+\exp(h(\eta(\beta),\psi))}\big) + (n_i - y_i)\log\big(1 - \frac{\exp(h(\eta(\beta),\psi))}{1+\exp(h(\eta(\beta),\psi))}\big)\Big]$
Binomial (Probit)	$l(\beta,\psi\|y) = \sum_{i=1}^{n}\Big[y_i \log\big(\Phi(h(\eta(\beta)),\psi)\big) + (n_i - y_i)\log\big(1 - \Phi(h(\eta(\beta)),\psi)\big)\Big]$
Poisson	$l(\beta,\psi\|y) = \sum_{i=1}^{n}\Big[y_i - \log\big(\exp(h(\eta(\beta),\psi))\big) - \exp(h(\eta(\beta),\psi))\Big]$

In addition the likelihood principle is not just a concept to estimate the parameters, but also provides some methods of model selection.

2.2.5 Deviance

One of this methods for the model selection is the deviance.
When modelling the data we replace the data values y_i by a set of fitted values $\hat{\mu}_i$. Now the question is how discrepant are the μ_i's and y_i's?
For the deviance, a measure of this discrepancy, we take twice the negative logarithm of the ratio of the likelihood functions of the saturated and the fitted model.

Definition 2.2.3 (deviance in a generalized linear model)**.**

$$D(\hat{\mu}, y) = -2\big[l(\hat{\mu}|y) - l(y|y)\big]$$

Note that the factor 2 ensures that for Gaussian observations the deviance equals the sum of squared errors.
Deviance for Normal, Binomial and Poisson data:

Distribution	**Deviance**
Normal	$D(\beta,\psi,y) = \sum_{i=1}^{n}\big(y_i - h(\eta(\beta),\psi)\big)^2$
Binomial (Logit)	$D(\beta,\psi,y) = 2\sum_{i=1}^{n}\Big[y_i \log\big(\frac{y_i(1+\exp(h(\eta(\beta),\psi)))}{n_i \exp(h(\eta(\beta),\psi))}\big) + (n_i - y_i)\log\big(\frac{n_i - y_i}{n_i(\frac{\exp(h(\eta(\beta),\psi))}{1+\exp(h(\eta(\beta),\psi))}) - n_i}\big)\Big]$
Binomial (Probit)	$D(\beta,\psi,y) = 2\sum_{i=1}^{n}\Big[y_i \log\big(\frac{y_i}{n_i\Phi(h(\eta(\beta)),\psi)}\big) + (n_i - y_i)\log\big(\frac{n_i - y_i}{n_i\Phi(h(\eta(\beta)),\psi) - n_i}\big)\Big]$
Poisson	$D(\beta,\psi,y) = 2\sum_{i=1}^{n}\Big[y_i - \log\big(\frac{y_i}{\exp(h(\eta(\beta),\psi))}\big) - y_i + \exp(h(\eta(\beta),\psi))\Big]$

Consider we want to compare models by likelihood ratio testing the hypotheses:

$$H_0 : g(\mu) = X_0\beta_0 \quad \text{against} \quad H_1 : g(\mu) = X_1\beta_1$$

If H_0 is true, then $2\big[l(\beta_0|y) - l(\beta_1|y)\big] \sim \chi^2_{p_0-p_1}$, where $l(\beta_0|y)$ and $l(\beta_1|y)$ are the maximized likelihoods of the two models.
So for the deviance we might expect that, if the model is correct, then $D(\hat{\mu}, y) \sim \chi^2_{n-p}$, where $n - p$ are the degrees of freedom of the model. If we would calculate the deviance of a model with all parameters known, i.e. a saturated model, then $D(y|y) \sim \chi^2_n$. (see Wood [12], p. 69 f.)

2.3 Data sets

This section introduces the examples illustrating the improvement achieved by the extended glm-functions. The following data sets contain different random components: The Gaussian distributed response will be modeled in the PCB Concentration data set, while Binomial random components are given in the Beetle Mortality data set as well as in the Incidence of Byssinosis and the Rotifer Suspension data set. The Coal Mining Fractures data set has a Poisson random component.
These five data sets were already studied in Czado [4] for illustrational purposes.
Every example will be explained and their characteristica shown, e.g. plots of their estimated distribution. In addition we will give some background for every data set and demonstrate changes to the data made for modeling purposes.

2.3.1 PCB Concentration in Trout (Gaussian)

Table 2.1 shows the Polychlorinated biphenyl (short: PCB) concentration in ppm and age in years of lake trout in Lake Cayuga, Ithaca, New York, U.S.A. Every row stands for one trout with the first column showing the PCB concentration found in this very trout and the age of the trout in the third column.
Source: Bates and Watts [1]

pcb	**log(pcb)**	**age**	**age.cen**	**pcb**	**log(pcb)**	**age**	**age.cen**
0.6	−0.5108256	1	−4.5357143	3.4	1.2237754	6	0.4642857
1.6	0.4700036	1	−4.5357143	9.7	2.2721259	6	0.4642857
0.5	−0.6931472	1	−4.5357143	8.6	2.1517622	6	0.4642857
1.2	0.1823216	1	−4.5357143	4.0	1.3862944	7	1.4642857
2.0	0.6931472	2	−3.5357143	5.5	1.7047481	7	1.4642857
1.3	0.2623643	2	−3.5357143	10.5	2.3513753	7	1.4642857
2.5	0.9162907	2	−3.5357143	17.5	2.8622009	8	2.4642857
2.2	0.7884574	3	−2.5357143	13.4	2.5952547	8	2.4642857
2.4	0.8754687	3	−2.5357143	4.5	1.5040774	8	2.4642857
1.2	0.1823216	3	−2.5357143	30.4	3.4144426	9	3.4642857
3.5	1.2527630	4	−1.5357143	12.4	2.5176965	11	5.4642857
4.1	1.4109870	4	−1.5357143	13.4	2.5952547	12	6.4642857
5.1	1.6292405	4	−1.5357143	26.2	3.2657594	12	6.4642857
5.7	1.7404662	5	−0.5357143	7.4	2.0014800	12	6.4642857

Table 2: PCB Concentration Data contained in the data frame `pcb.ex`

To improve the interpretability we center the variable `age`. This has no effect on the model, but it changes the intercept, which is the mean of the dependent variable when the covariate is zero. Modeling with a centered covariate is especially helpful if the covariate has no values around zero as it is the case with `age` in this example. Observing the mean pcb concentration at birth of the trout does not make much sense. By centering the covariate we change the zero point and the intercept is now the average value when the covariate is at its mean. So we create a new variable which is equal to every data point for the variable `age` minus the mean of `age`.

Mean of age: $\frac{\sum \text{age values}}{\text{number of age data points}} = \frac{4{\cdot}1+3{\cdot}2+3{\cdot}3+3{\cdot}4+1{\cdot}5+3{\cdot}6+3{\cdot}7+3{\cdot}8+1{\cdot}9+1{\cdot}11+3{\cdot}12}{28} =$
$= 5.5357143$.
As the logarithm of the random variable pcb is normally distributed, we study the data set with log(pcb) as the response variable which has now a gaussian distribution:

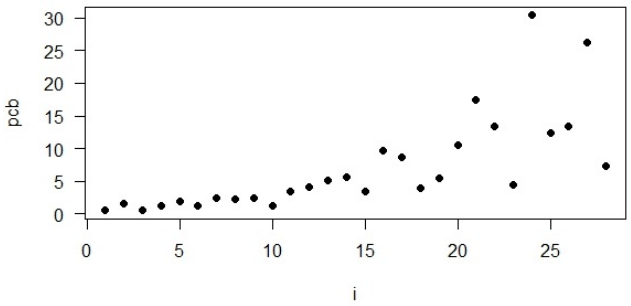

Figure 1: Distribution of pcb

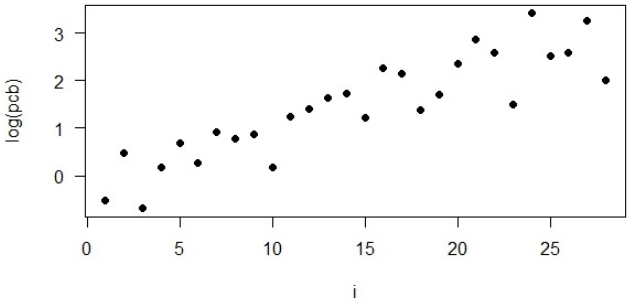

Figure 2: Gauss-distribution of log(pcb)

In the following histogram we can also see the normal distribution of the variable log(pcb):

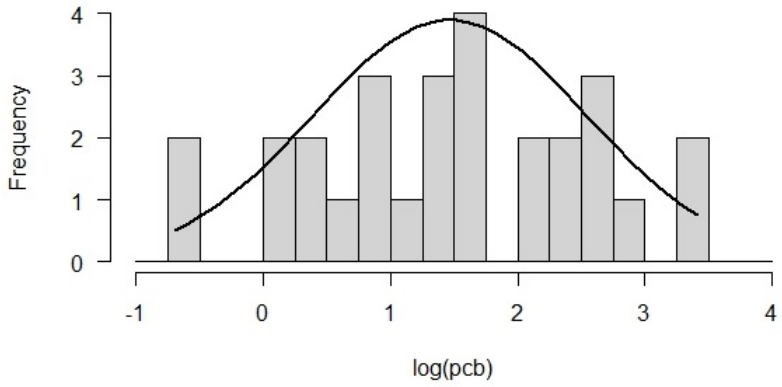

Figure 3: Histogram of the log(pcb)-distribution

We can also have a look at the impact of the variable age on the PCB-concentration in the trout:

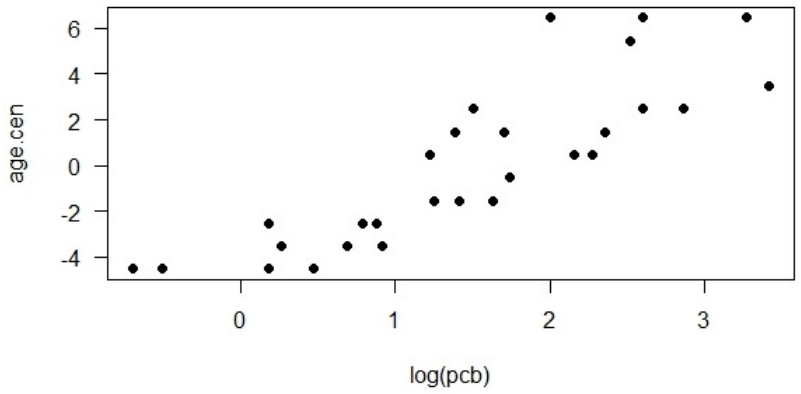

Figure 4: Bivariate Analysis of age.cen versus log.pcb

The plot shows a linear relationship between the age of the trout and the PCB-concentration. That's why we will include `age` as a linear term in the model later.

2.3.2 Beetle Mortality (Binomial)

This example is about the number of beetles that were killed by a five hours' exposure to carbon disulphide. This is a pale, very refractive liquid, that can, in gaseous state, easily be absorbed by the spiracles and leads to poisoning when exposed long enough to the gas. Injected in the ground at the roots or radiated over plants, carbon disulphide is used for pest control.
In the first column Y_i is the number of observed dead beetles. The second column shows n_i, the number of overall observed insects, while the third one gives the concentration of the gas CS_2 in $log_{10}\ mgl^{-1}$.
Source: Bliss [2]

$\mathbf{Y_i}$ **Number killed**	$\mathbf{n_i}$ **Number of Insects**	**Dose** $\log_{10}\mathbf{CS_2 mgl^{-1}}$	**dose.cen**
6	59	1.6907	−0.102725
13	60	1.7242	−0.069225
18	62	1.7552	−0.038225
28	56	1.7842	−0.009225
52	63	1.8113	0.017875
53	59	1.8369	0.043475
61	62	1.8610	0.067575
60	60	1.8839	0.090475

Table 3: Beetle Mortality Data contained in the data frame `beetle.ex`

As before in the PCB data set we center the covariate. Dose is nowhere around the zero point, so we want the intercept to be the average value at the mean dose. The centered variable dose was denoted as `dose.cen`.
The Beetle Mortality data set is Binomial, which means that it describes a series of identical and independent experiments that have only two possible outcomes. In this case it is *beetle dead* or *beetle alive*.
The experiments were run differently often with a specific dose of carbon disulphide. To compare these several test series, we observe the quotient $\frac{Y_i}{n_i}$ and for example look at the percentage of killed beetles:

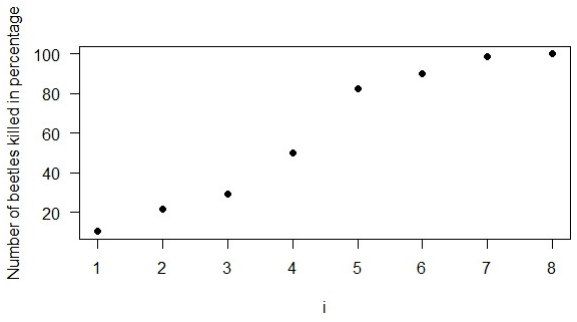

Figure 5: Binomial-distribution of $\frac{Y_i}{n_i}$

We can also have a look at the dependence of the ratio of killed beetles on the centered dose of carbon disulphide.

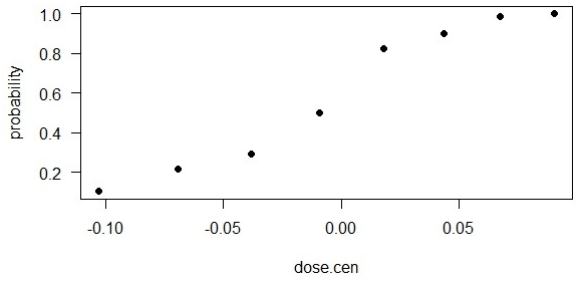

Figure 6: Centered dose against probability of beetles' death

The plot shows a linear effect of the covariate dose. This leads us to use a Binomial logistic regression on the beetle data set and include dose as a linear term.

2.3.3 Incidence of Byssinosis among Cotton Textile Workers (Binomial)

Table 4 shows a health survey on the incidence of Byssinosis among cotton textile workers. Byssinosis is an occupational lung disease and commonly occurs when exposed to cotton dust in not enough ventilated working places, such as fabric manufacture industries.
The workers were classified by the factors type of work place, employment years and smoking.
Source: Higgins and Koch [8]

Employment Years	**Smoking**	**WorkPlace**		
		mostdusty	**lessdusty**	**leastdusty**
< 10	yes	30/233	3/403	11/951
	no	7/126	5/283	7/733
$10 - 19$	yes	16/67	2/94	3/320
	no	3/20	1/51	1/160
> 19	yes	41/155	4/237	15/733
	no	8/72	3/232	5/553

Table 4: Incidence of Byssinosis

For model building the second rearranged table (table 5) has categorical covariates workspace ($-1\widehat{=}$ most dusty, $0\widehat{=}$ less dusty, $1\widehat{=}$ least dusty), smoking ($1\widehat{=}$ yes, $0\widehat{=}$ no) and employment years ($-1\widehat{=} \leq 10$ years, $0\widehat{=}10-19$ years, $1\widehat{=} \geq 20$ years). Y_i is the number of workers complaining about Byssinosis symptons and n_i the number of workers in the class i.

$\mathbf{Y_i}$	$\mathbf{n_i}$	**work**	**smoking**	**employ**	$\mathbf{Y_i}$	$\mathbf{n_i}$	**work**	**smoking**	**employ**
30	233	-1	1	-1	1	51	0	0	0
7	126	-1	0	-1	4	237	0	1	1
16	67	-1	1	0	3	232	0	0	1
3	20	-1	0	0	11	951	1	1	-1
41	151	-1	1	1	7	733	1	0	-1
8	72	-1	0	1	3	320	1	1	0
3	403	0	1	-1	1	160	1	0	0
5	283	0	0	-1	15	733	1	1	1
2	94	0	1	0	5	553	1	0	1

Table 5: Byssinosis data rearranged for binary regression in `bys.ex`

The probability of workers diseased with Byssinosis has the following distribution:

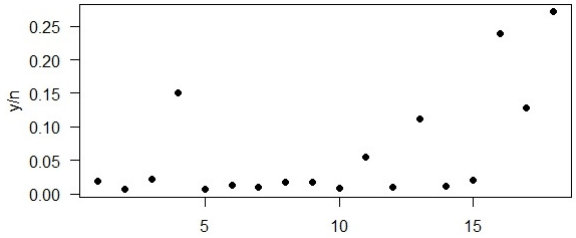

Figure 7: Distribution of $\frac{y_i}{n_i}$

It does not show the typical shape of a Binomial distribution and that's why we will use a Probit generalized linear model on this data set. The difference between the Logit and the Probit model in the link function is that the Logit model has slightly flatter tails while the Probit curve approaches the axes more quickly.

2.3.4 Rotifer Suspension (Binomial)

This example shows the number of rotifers of two species (1 $\widehat{=}$ polyartha major, 0 $\widehat{=}$ keratella cochlearis) that occur at a suspension for different fluid densities.
Source: Collett [3]

$\mathbf{Y_i}$	$\mathbf{n_i}$	**Density**	**Species**	**density.cen**	$\mathbf{Y_i}$	$\mathbf{n_i}$	**Density**	**Species**	**density.cen**
11	58	1.019	1	−2.565	13	161	1.019	0	−2.565
7	86	1.020	1	−2.465	14	248	1.020	0	−2.465
10	76	1.021	1	−2.365	30	234	1.021	0	−2.365
49	212	1.030	1	−1.465	59	573	1.030	0	−1.465
13	29	1.031	1	−1.365	26	167	1.031	0	−1.365
44	75	1.040	1	−0.465	54	403	1.040	0	−0.465
36	56	1.041	1	−0.365	23	162	1.041	0	−0.365
20	27	1.048	1	0.335	7	42	1.048	0	0.335
54	59	1.049	1	0.435	22	48	1.049	0	0.435
29	36	1.050	1	0.535	43	209	1.050	0	0.535
14	17	1.060	1	1.535	71	74	1.060	0	1.535
10	22	1.061	1	1.635	25	45	1.061	0	1.635
64	66	1.063	1	1.835	94	101	1.063	0	1.835
644	667	1.070	1	2.535	395	412	1.070	0	2.535

Table 6: Rotifer Suspension Data contained in the data frame `rotifer.ex`

As already done in the previous data sets, the variable density was centered for modeling purposes and denoted then as `density.cen`. $\frac{y_i}{n_i}$ has a Binomial distribution:

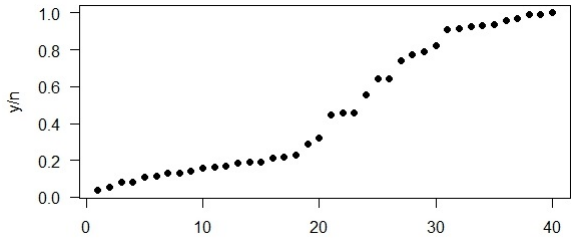

Figure 8: Binomial distribution of $\frac{y_i}{n_i}$ in Rotifer data

We will therefore use a Binomial logistic regression on this data set.

2.3.5 Coal Mining Fractures (Poisson)

Table 7 shows Y_i, the number of injuries in coal mines in the Appalachian region, Virginia, U.S.A. Covariates are `inb`, the inner burden thickness in feet, `extrp`, the percentage of extraction of the lower previously mined seam and `time`, the time that the mine is at work in years. All of these covariates have been centered to ensure the interpretability of our later generalized linear model.
Source: Myers [11]

$\mathbf{Y_i}$	inb	extrp	time	inb.cen	extrp.cen	time.cen	$\mathbf{Y_i}$	inb	extrp	time	inb.cen	extrp.cen	time.cen
2	50	70	1.0	−119.23	−5.93	−6.16	3	65	75	5.0	−104.23	−0.93	−2.16
1	230	65	6.0	60.77	−10.93	−1.16	3	470	90	9.0	300.77	14.07	1.84
0	125	70	1.0	−44.23	−5.93	−6.16	2	300	80	9.0	130.77	4.07	1.84
4	75	65	0.5	−94.23	−10.93	−6.66	2	275	90	4.0	105.77	14.07	−3.16
1	70	65	0.5	−99.23	−10.93	−6.66	0	420	50	17.0	250.77	−25.93	9.84
2	65	70	3.0	−104.23	−5.93	−4.16	1	65	80	15.0	−104.23	4.07	7.84
0	65	60	1.0	−104.23	−15.93	−6.16	5	40	75	15.0	−129.23	−0.93	7.84
0	350	60	0.5	180.77	−15.93	−6.66	2	900	90	35.0	730.77	14.07	27.84
4	350	90	0.5	180.77	14.07	−6.66	3	95	88	20.0	−74.23	12.07	12.84
4	160	80	0.0	−9.23	4.07	−7.16	3	40	85	10.0	−129.23	9.07	2.84
1	145	65	10.0	−24.23	−10.93	2.84	3	140	90	7.0	−29.23	14.07	−0.16
4	145	85	0.0	−24.23	9.07	−7.16	0	150	50	5.0	−19.23	−25.93	−2.16
1	180	70	2.0	10.77	−5.93	−5.16	0	80	60	5.0	−89.23	−15.93	−2.16
5	43	80	0.0	−126.23	4.07	−7.16	2	80	85	5.0	−89.23	9.07	−2.16
2	42	85	12.0	−127.23	9.07	4.84	0	145	65	9.0	−24.23	−10.93	1.84
5	42	85	0.0	−127.23	9.07	−7.16	0	100	65	9.0	−69.23	−10.93	1.84
5	45	85	0.0	−124.23	9.07	−7.16	3	150	80	3.0	−19.23	4.07	−4.16
5	83	85	10.0	−86.23	9.07	2.84	2	150	80	0.0	−19.23	4.07	−7.16
0	300	65	10.0	130.77	−10.93	2.84	3	210	75	2.0	40.77	−0.93	−5.16
5	190	90	6.0	20.77	14.07	−1.16	5	11	75	0.0	−158.23	−0.93	−7.16
1	145	90	12.0	−24.23	14.07	4.84	0	100	65	25.0	−69.23	−10.93	17.84
1	510	80	10.0	340.77	4.07	2.84	3	50	88	20.0	−119.23	12.07	12.84

Table 7: Mining Fracture Data contained in the data frame `mining.ex`

Y_i is a counting response variable and follows a Poisson distribution:

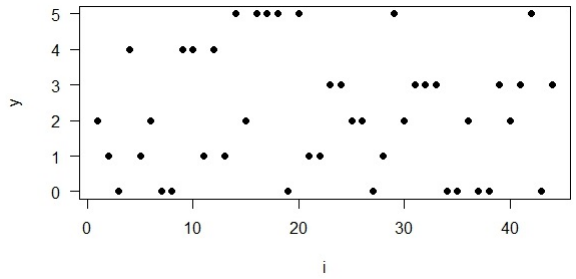

Figure 9: Poisson distribution of y_i

To know how we should include the covariates in the model we observe the impact of the individual covariates on the response y.

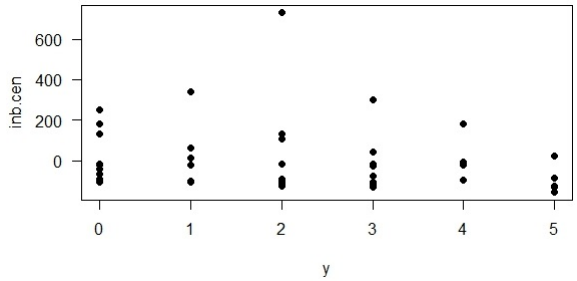

Figure 10: Bivariate Analysis of inb versus y

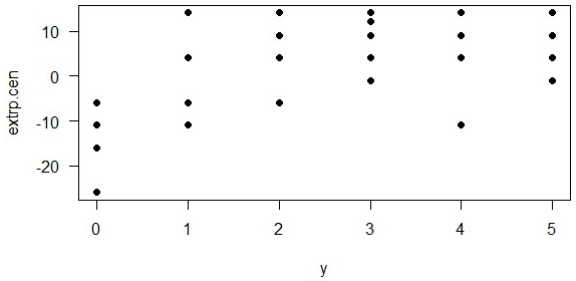

Figure 11: Bivariate Analysis of extrp versus y

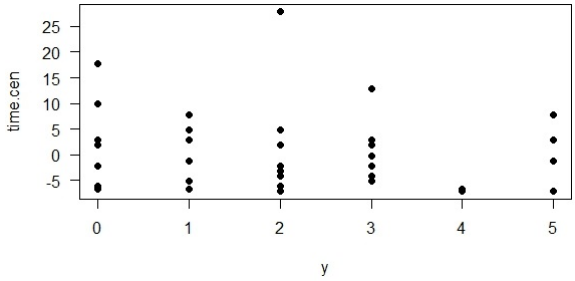

Figure 12: Bivariate Analysis of time versus y

Both the inner burden thickness and the percentage of extraction have a linear impact on the number of injuries in coal mines, but the covariate `time` has no effect on y. Hence we will not include `time` in the model. Our model is: $y \sim \beta_0 + \beta_1 \cdot inb + \beta_2 \cdot extrp$.

2.4 Fitting the data with the standard glm-function in R

For a better comparison to the new functions the example data sets are now explored by the standard `glm` function in `R` in the first place. For this purpose the `glm` function will be called with the appropriate parameters and we then explore the summary. We will also have a look at the significance levels of the covariance and check the deviance for the goodness of fit.

PCB Concentration in Trout
We explore the linear model with log(PCB) as the response variable and centered age as covariate. As log(PCB) was normally distributed we select `gaussian` as the family function.

```
> pcb <- glm(log.pcb ~ age.cen, family = gaussian, data = pcb.ex)
> summary(pcb)

Call: glm(formula = log.pcb ~ age.cen, family = gaussian, data = pcb.ex)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.13953  -0.38791   0.09568   0.43270   1.05082

Coefficients:
           Estimate  Std. Error  t value   Pr(>|t|)
(Intercept) 1.4659      0.1071   13.681   2.18e-13  ***
age.cen     0.2591      0.0308    8.414   6.78e-09  ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.3214991)

    Null deviance: 31.120 on 27 degrees of freedom
Residual deviance:  8.359 on 26 degrees of freedom
AIC: 51.612

Number of Fisher Scoring iterations: 2
```

The output above shows that the intercept and the variable age are both highly significant, i.e. the age of the trout has an impact of the PCB concentration in it.
One can also check the model for the deviance. The null deviance is the deviance for a model with just a constant term, while the residual deviance is the deviance of the fitted model. Here the residual deviance of 8.359 is moderately well on 26 degrees of freedom given only one covariate, which means that the standard `glm` function already fits quite well. But later it will be shown that using a parametric link function will improve even this good fit.

Beetle Mortality

Now we explore the first Binomial data set beetle mortality, where we choose `binomial` as family function. The response is an object of two vectors. In the first column we pass the number of successes, i.e. the number of dead beetles y_i, and in the second column the number of fails, i.e. the number of beetles alive $n_i - y_i$. The covariate is the centered dose of carbon disulphide.

```
> beetle <- glm(cbind(y, n - y) ~ dose.cen, family = binomial, data = beetle.ex)
> summary(beetle)

Call: glm(formula = cbind(y, n - y) ~ dose.cen, family = binomial, data = beetle.ex)

Deviance Residuals:
    Min       1Q  Median      3Q     Max
-1.5941  -0.3944  0.8329  1.2592  1.5940

Coefficients:
          Estimate Std. Error  z value  Pr(>|z|)
(Intercept) 0.7438     0.1379    5.396  6.83e-08 ***
dose.cen    34.2703    2.9121   11.768  < 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: 284.202 on 7 degrees of freedom
Residual deviance:  11.232 on 6 degrees of freedom
AIC: 41.43

Number of Fisher Scoring iterations: 4
```

Unlike the previous example the Beetle Mortality data does not fit well with the standard `glm` function. 11.232 residual deviance on 6 degrees of freedom is a too high value. Notwithstanding intercept and variable dose are highly significant.

Incidence of Byssinosis among Cotton Textile Workers

Our second binomial data set is the Byssinosis one, where we explore the influence of the variables workspace, smoking and employment on the response, the successes and fails of the health survey, i.e. the number of workers with and without Byssinosis, respectively.

```
> bys <- glm(cbind(y,n-y)~workspace+smoking+employment, family=binomial, data=bys.ex)
> summary(bys)

Call: glm(formula=cbind(y,n-y)~workspace+smoking+employment, family=binomial, data=bys.ex)

Deviance Residuals:
    Min      1Q Median     3Q    Max
-3.3356 -0.4823 0.1622 1.1600 2.1055

Coefficients:
            Estimate Std. Error  z value  Pr(>|z|)
(Intercept) -3.76256    0.16518  -22.778  < 2e-16  ***
workspace   -1.46572    0.10578  -13.856  < 2e-16  ***
```

```
smoking      0.67781    0.18871  3.592    0.000328 ***
employment   0.33313    0.08861  3.760    0.000170 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 290.739 on 17 degrees of freedom
Residual deviance:  40.774 on 14 degrees of freedom
AIC: 112.14

Number of Fisher Scoring iterations: 5
```

Every covariate is highly significant, so our model makes sense, although a residual deviance of 40.774 on 14 degrees of freedom is the worst outcome up to now. Remember that the residual deviance should be $\sim \chi^2_{14}$. The probability that 40.774 comes from the χ^2_{14}-distribution is

```
> 1 - pchisq(40.774, 14)
[1] 0.0001930327
```

This small probability confirms the poor fit of the model.

Rotifer Suspension

The last binomial example is about the Rotifer Suspension, where again the response are the combined vectors of successes and fails, in this case the number of rotifers occured and the number of those who did not occure.
Our covariates are both species of rotifer and centered density of the fluids separately as well as species and density.cen together.

```
> rotifer <- glm(cbind(y,n-y)~species+density.cen+species*density.cen,
             family=binomial, data=rotifer.ex)
> summary(rotifer)

Call: glm(formula=cbind(y,n-y)~species+density.cen+species*density.cen,
          family=binomial, data=rotifer.ex)

Deviance Residuals:
    Min      1Q Median     3Q    Max
-6.4708 -2.3380 0.5987 2.4383 6.2355

Coefficients:
                    Estimate Std. Error z value Pr(>|z|)
(Intercept)         -0.75090    0.05460 -13.754   <2e-16 ***
species              1.41441    0.09871  14.329   <2e-16 ***
density.cen          1.08746    0.03857  28.191   <2e-16 ***
species:density.cen -0.03077    0.06329  -0.486    0.627
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3180.99 on 39 degrees of freedom
```

```
Residual deviance:  434.02 on 36 degrees of freedom
AIC: 596.57

Number of Fisher Scoring iterations: 5
```

Only the separate covariates species and density are highly significant, whereas the mixed one is not even slightly significant with a p-value of 0.627.

Figure 13: Bivariate Analysis of species versus density

Also the residual deviance of 434.02 on 36 degrees of freedom is a very bad result and shows that the standard `glm` function does not fit at all. It is obvious that we need an alternate model for this data set, because the standard `glm` function can not be applied on it.

Mining

The last data set to fit with the `glm` function is the Poisson one, so we set `family = poisson`. The response is y, the accidents in coal mines, and our covariates are centered inner burden thickness and percentage of extraction. As seen in chapter 2.3 the covariate `time` has had no influence on the response and will therefore not be included in the model.

```
> mining <- glm(y ~ inb.cen+extrp.cen, family = poisson, data = mining.ex)
> summary(mining)

Call: glm(formula = y ~ inb.cen+extrp.cen, family = poisson, data = mining.ex)

Deviance Residuals:
    Min      1Q  Median      3Q     Max
-1.9257 -0.9483 -0.1884  0.5339  2.0921

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) 0.5992363  0.1237493   4.842 1.28e-06 ***
inb.cen    -0.0017076  0.0007468  -2.286   0.0222 *
extrp.cen   0.0584197  0.0118114   4.946 7.57e-07 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 74.984 on 43 degrees of freedom
Residual deviance: 42.094 on 41 degrees of freedom
AIC: 144.37

Number of Fisher Scoring iterations: 5
```

This is a better result as it was at the binomial examples, but a residual deviance of 42.094 on 41 degrees of freedom is not worthwhile. The probability that the deviance comes from the χ^2_{41}-distribution is

```
> 1 - pchisq(42.094, 41)
[1] 0.4233628
```

and clarifies the failure of the standard link in `glm`.

One can see that the result is not entirely satisfying, because the standard link in the `R`-specific `glm`-function is not appropriate for this data sets. Using the standard link function leads to wide deviations and mistakes in the model. For such data sets we have to find another method to apply a generalized linear model. One of them are the parametric link functions that contain tail transformations.

3 Modifying with tail transformation

Depending on the data either the right or the left or both sides of the curves have to be modified. These transformations must then be included in the link functions.

3.1 h - power transformations

To transform the tails of a graph the three following general h(·)-power functions were considered in Czado [4]. The parameters of the functions are η_0 and ψ_1 or ψ_2 or ψ_1 and ψ_2, respectively. η_0 defines the length of the tail, because it is the starting value for the transformation. For right tail modification every point bigger than η_0 will be transformed and for left tail transformation every point smaller than η_0 will be transformed. The ψ-parameters define how intense the transformation will be, where a ψ-value greater than 1 increases the slope and a ψ-value smaller than 1 decreases the slope. $\psi = 1$ means no transformation.
The functions were given as follows:

hpsi1 - Right tail modification implemented in `hpsi1.r`

$$hpsi1_{\eta_0}(\eta, \psi_1) = \begin{cases} \eta_0 + \ln(\eta - \eta_0 + 1) & \text{if} \quad \eta \geq \eta_0 \quad \text{and} \quad \psi_1 = 0 \\ \eta_0 + \frac{(\eta - \eta_0 + 1)^{\psi_1} - 1}{\psi_1} & \text{if} \quad \eta \geq \eta_0 \quad \text{and} \quad \psi_1 \neq 0 \\ \eta & \text{otherwise} \end{cases}$$

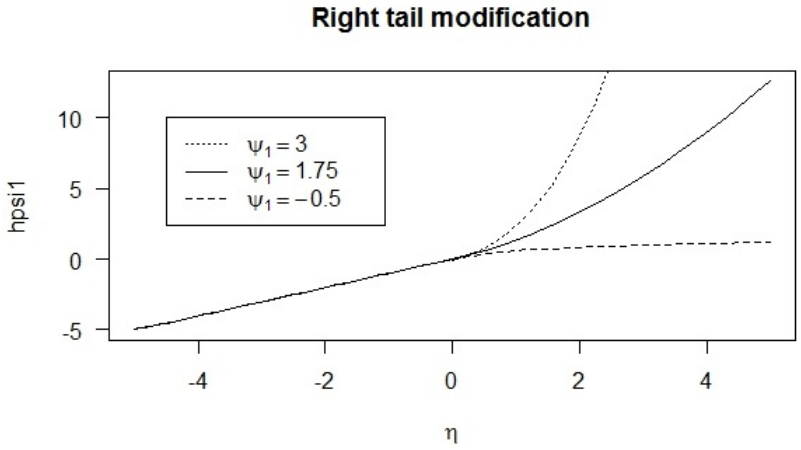

Figure 14: Right tail modification with `hpsi1.r`

hpsi2 - Left tail modification implemented in `hpsi2.r`

$$hpsi2_{\eta_0}(\eta, \psi_2) = \begin{cases} \eta & \text{if} \quad \eta \geq \eta_0 \\ \eta_0 - \ln(-\eta + \eta_0 + 1) & \text{if} \quad \eta < \eta_0 \quad \text{and} \quad \psi_2 = 0 \\ \eta_0 - \frac{(-\eta + \eta_0 + 1)^{\psi_2} - 1}{\psi_2} & \text{otherwise} \end{cases}$$

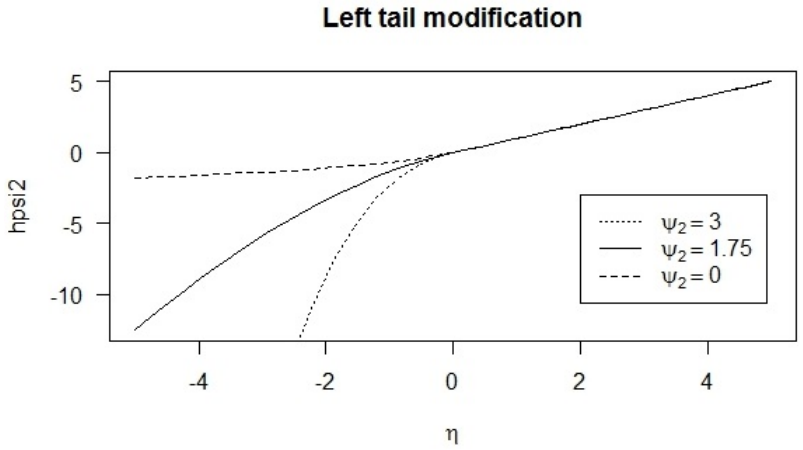

Figure 15: Left tail modification with `hpsi2.r`

hpsi12 - Right and left tail modification implemented in `hpsi12.r`

$$hpsi12_{\eta_0}(\eta, \psi = (\psi_1, \psi_2)) = \begin{cases} \eta_0 + \ln(\eta - \eta_0 + 1) & \text{if} \quad \eta \geq \eta_0 \quad \text{and} \quad \psi_1 = 0 \\ \eta_0 + \frac{(\eta - \eta_0 + 1)^{\psi_1} - 1}{\psi_1} & \text{if} \quad \eta \geq \eta_0 \quad \text{and} \quad \psi_1 \neq 0 \\ \eta_0 - \ln(-\eta + \eta_0 + 1) & \text{if} \quad \eta < \eta_0 \quad \text{and} \quad \psi_2 = 0 \\ \eta_0 - \frac{(-\eta + \eta_0 + 1)^{\psi_2} - 1}{\psi_2} & \text{otherwise} \end{cases}$$

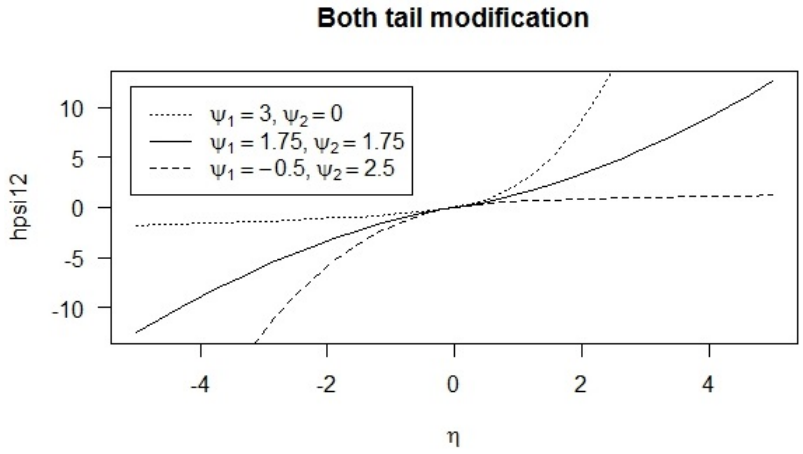

Figure 16: Both tail modification with `hpsi12.r`

In the cases $\psi_1 = 0$ or $\psi_2 = 0$ we take $\lim_{\psi \to \infty} \frac{(\cdot)^{\psi} - 1}{\psi} = \ln(\cdot)$. We need an extra if-statement in the implementation to handle this limiting case. Furthermore, it turned out that a smoother if-statement like `if(any(`$\psi > -1e-14$ `&&` $\psi < 1e-14$`))` avoids numerical problems.

3.2 Generalized linear models with tail transformation

glm.model
This function fits a generalized linear model with tail transformation using a specific η_0 and ψ. It has the input parameters `y, X, n, data, family, tail, eta0, psi1` and `psi2`. `y` is the response (`n` only needed with Binomial data) and `X` the design matrix, where the first column contains a vector of ones and the columns left contain the covariates.
Family can be `"gauss", "logit", "probit"` and `"poiss"`, while tail has the three options `"right", "left"` and `"both"`.
`eta0, psi1` and `psi2` are the parameters for the tail transformation function hpsi.
In the first step it matches the `family` with the right `model`-function. For example if `family=logit`, then `model.logit` is called.
There are first of all the loglikelihood functions for right, left and both tail modification provided. These loglikelihoods use the hpsi-power transformations.
Whichever tail is selected the appropriate loglikelihood function is optimized. Therefore we use the `R`-specific function `optim`, which gives us the optimal regression coefficients β_i. Note that `optim` minimizes a function and hence our loglikelihood functions have to calculate the negative loglikelihood.
`glm.model` now prints these optimal β_i's in a matrix at the right covariates.
After that the deviance will be computed and displayed together with the degrees of freedom, which is the length of y minus the number of columns of the design matrix X.

glm.profile
This is a function to find the optimal ψ-value, because one might not know beforehand, what ψ to run the tail transformation glm with. The input parameters are the same as in `glm.model`, but instead of ψ_1 and/or ψ_2, one should put in `psivec`, a vector of ψ-values, from which the optimal value for the model will be selected.
Again the function asks for the `family` in the first step and calls the right `profile`-function. As in the `model`-functions there are the likelihood functions provided. In a for-loop the `optim`-function optimizes the likelihood for every ψ in `psivec` and computes the corresponding deviance. In the case of `tail="both"` we need two for-loops around the `optim`-function and create a deviance-matrix, theirs i,j-entry is the deviance of the model with ψ_1 =`psivec[i]` and ψ_2 =`psivec[j]`. The function then plots a 3D profile plot of ψ_1, ψ_2 and deviance and return the minimal deviance at the corresponding ψ-values.
In right or left tail case we create a deviance-vector. Here we need only one for-loop, which passes through every ψ-value in `psivec`. Afterwards we plot ψ against the deviance-vector and return the ψ-value from the vector that has the lowest deviance.
In addition this plot contains a 95% confidence interval for the optimal ψ in dotted lines.

glm.fitted
This function draws two plots of observed versus fitted means. One for the ordinary GLM regression and the other for the GLM with tail transformation. We fit a generalized linear model by applying the `optim`-function on the loglikelihoodfunction which we provided like in `glm.model` or `glm.profile`. In doing this we get the fitted means

of the generalized linear model with tail transformation. Then we do the whole process again but choose $\psi = 1$ instead, which gives us the fitted means of the standard regression.

glm.inf
The glm.inf function gives standard errors for both the regression parameters and the link parameters and a correlation matrix of all parameters.
First we provide the loglikelihood function for the the generalized linear model with both a fixed ψ-value and an estimated one.
`optim` gives us the optimal parameters as well as the hessian matrix. We invert the hessian matrix to get the Fisher Information matrix. With this data we calculate the standard errors and ratio of estimate for both fixed and estimated link parameters and the variance inflation caused by the additional estimation of the link parameters.
Then we give the link parameter estimates and their standard error as well as the correlation matrix for link and regression parameters, which can be calculated from the Fisher Information matrix.

4 Examples

To show how all the functions of chapter 3 work, we now discuss examples of the new glm-functions that contain parametric links with tail transformations. On that point the data sets introduced in chapter 2.3 and 2.4 will be explored, this time with a for the data suitable tail transformation. The function `glm.profile` helps to find the best parameters to use for the tail transformed model. The results of these new models will be examined by `glm.model` and `glm.inf`. After that they will be compared to the ones in chapter 2.4, where they were fitted with the standard `glm` function of `R`. For that we will not only use the deviance as a measure for the goodness of fit, but also especially the function `glm.fitted`, which gives us two plots of the observed and fitted means for a head-to-head record.

4.1 Gaussian Random Component

First we show how the new functions work with Gauss-Data.

PCB Concentration in Trout
As we have seen in chapter 2.4 the PCB data set needs to be fitted with a right tail transformation. We use the `glm.profile` to see which ψ_1-value we should use for our model.
We set y as the response `log.pcb` and the design matrix X contains a vector of ones in the first column and the covariate `age.cen` in the second column.
First we need to construct this design matrix by building a vector of ones and a vector of the centered age values and combine them in a matrix:

```
> intercept <- rep(1, length(y))
> age.cen <- pcb.ex$age.cen
> X <- as.matrix(cbind(intercept, age.cen))
```

Note that the `glm.model` as well as the `glm.inf` take the names of the columns of the design matrix X to name the covariates in the output. So we named our vectors for the matrix "intercept" and "age.cen". If one does not name the matrix columns the function will name the covariates in the output "intercept", "covariate 1", "covariate 2", etc. automatically.
The family is `"gauss"` and for tail we choose `"right"`.
The output of the standard glm in chapter 2.4 revealed that all observations greater than `age.cen` $= 0$ need the tail transformation, so we select 0 as the η_0-value.
As the observations, that do not fit on the right tail, lie slightly beneath the curve, we want to select ψ-values between zero and one. Note that $\psi = 1$ means no transformation and represents the standard `glm`. We want to have the value 1 in our ψ-vector as well, in order to be able to compare to the standard `glm`.

```
> glm.profile(y, X, family="gauss", tail="right", psivec=seq(-0.3,1.5,by=0.05), eta0=0)
Minimal Deviance 6.327025 at the optimal psi-value 0.2
```

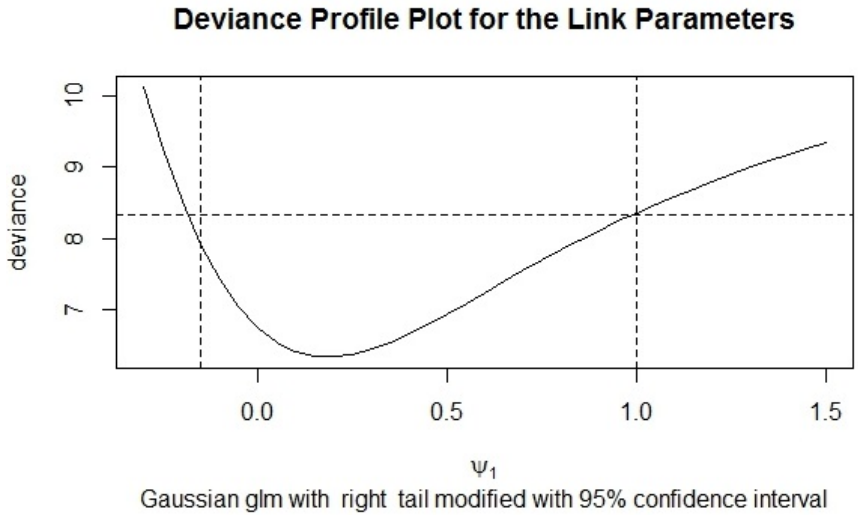

Figure 17: Profile Deviance Plot for ψ_1 for the PCB data

The dotted lines give us a 95% - Confidence Interval for ψ_1. The value $\psi_1 = 1$, which is no transformation at all and therefore indicates the standard model, lies only just within this interval. That explains why the R-specific `glm` already fitted moderately well and had a residual deviance of 8.359.
But with a ψ_1-value of 0.2 we get the minimal deviance 6.327025, which is a pretty good improvement of 25.4%.

So we run the `glm.model` with this particular ψ_1. The parameters `family="gauss", tail="right"` and `eta0=0` stay the same.

```
> glm.model(y, X, family="gauss", tail="right", psi1=0.2, eta0=0)
Fitting Gaussian GLM with  right  tail modified, psi =  0.2

Coefficients:
  (Intercept)   age.cen
     3.361053 0.7728385

Residual Deviance:  6.327025  on  26  degrees of freedom
```

This right tail transformed model has an intercept of 3.361053 and a slope of 0.7728385. The residual deviance 6.327025 on 26 degrees of freedom is an extremely good value. The probability that it follows a χ^2_{26}-distribution is 99.997%:

```
> 1 - pchisq(6.327025, 26)
[1] 0.9999723
```

To see how much our new model has improved to the standard one and how well it fits now, we call the `glm.fitted` function, which gives us the observed versus fitted mean plots of both the ordinary linear regression by `glm` in R and the tail transformed model.

```
> glm.fitted(y, X, family="gauss", tail="right", psi1=0.2 ,eta0=0)
Gauss GLM with  right  modified
Observed vs Fitted Mean Plot
```

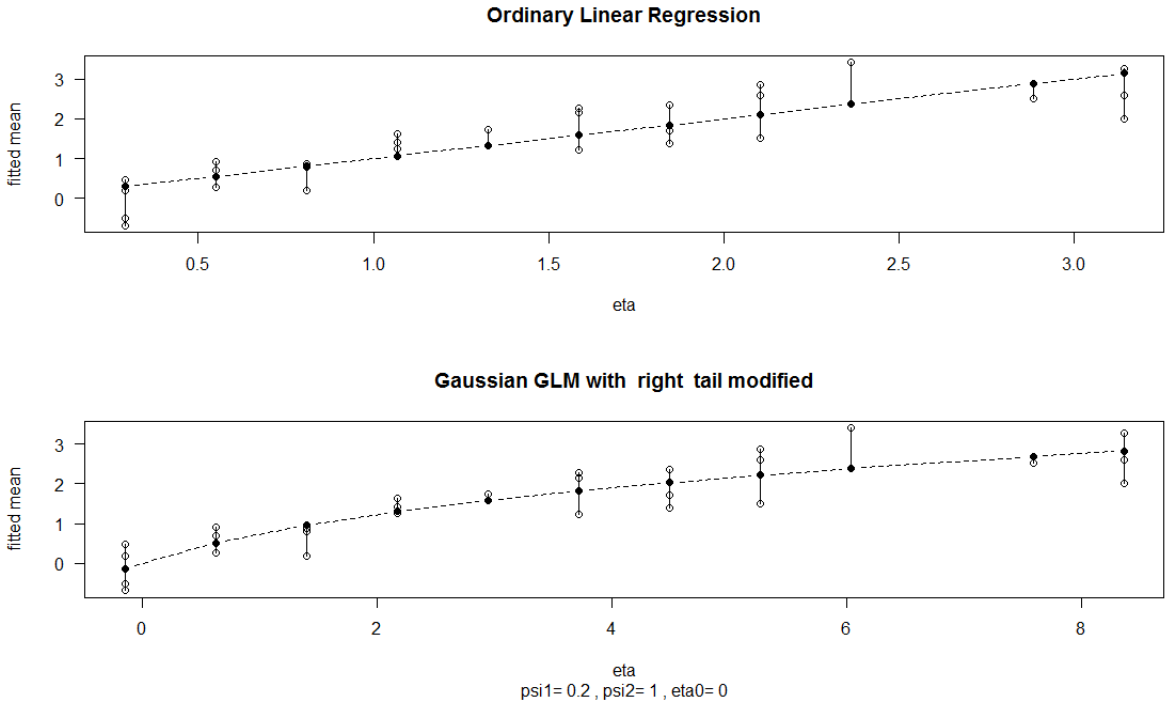

Figure 18: Observed vs Fitted Mean Plots for PCB data

These plots show how the goodness of our model has improved using a right tail transformation. The estimated data points now lie closer to the observed ones.
For more information about the regression parameters of the new generalized linear model, we call `glm.inf`, which gives us some standard errors for both the regression parameters and the link parameter and in addition the correlation matrix.

```
> glm.inf(y, X, family="gauss", tail="right", psi1=0.2, eta0=0)
Gaussian GLM with  right  tail modified, psi1 =  0.2

Regression Parameters:
             coeff  fixed se  fixed z  estimated se  estimated z  var inflation
(Intercept) 3.3611   0.31247   10.757        0.9651        3.483          208.9
age.cen     0.7728   0.08064    9.583        0.2347        3.293          191.0

Link Parameter:
psi1 =  0.2 , standard error of psi1 =  0.1546

Correlation matrix with estimated link:
            (Intercept) age.cen    psi1
(Intercept)      1.0000  0.9807 -0.9414
age.cen          0.9807  1.0000 -0.9346
psi1            -0.9414 -0.9346  1.0000
```

4.2 Binomial Random Component

The second distribution we want to explore is the Binomial one. The model improvements of this distribution will be shown in the three Binomial data sets and we will use both Logit and Probit links.

Beetle Mortality
First we use the profile function and therefore create the design matrix. The left tail of the Beetle data set needs to be transformed so we choose `tail="left"`.

```
> profile.logit(y,n,X,tail="left",psivec=seq(-0.5,1,by=0.05),eta0=0)
Minimal Deviance  3.047763 at the optimal psi-value  0.15
```

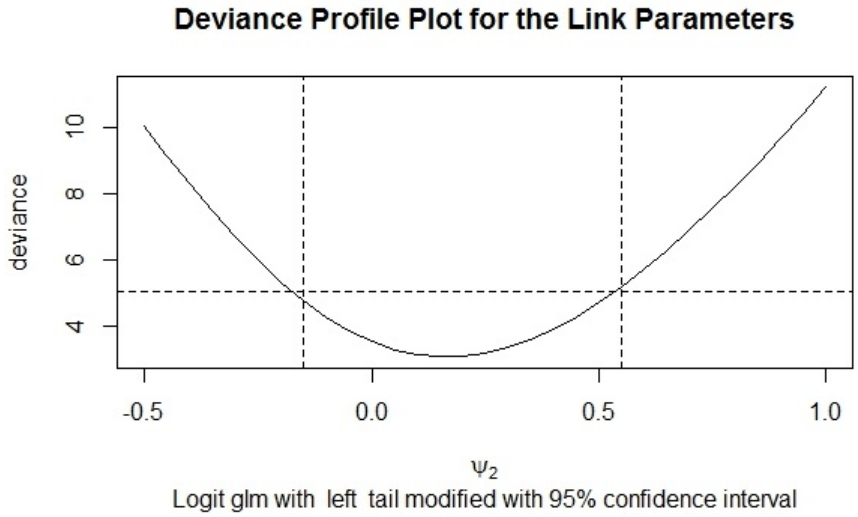

Figure 19: Profile Deviance Plot for ψ_2 for the Beetle data

The minimum deviance is 3.047763 attained at $\psi_2 = 0.15$. Once again note that $\psi = 1$ means no transformation and calling the function with `psi = 1` gives the standard `glm` of `R`. As we can see in the deviance-plot, the value 1 does not lie within the 95% Confidence Interval. That means that it was sensible to use a left tail transformation instead of our standard model from chapter 2.4. We now run the model with the suggested ψ_2-value 0.15:

```
> glm.model(y, X, n, family="logit", tail="left", psi2=0.15, eta0=0)
Fitting Logit GLM with  left  tail modified, psi =  0.15

Coefficients:
  (Intercept) dose.cen
    0.5104924 48.62181

Residual Deviance:  3.047763  on  6  degrees of freedom
```

glm.model gives an intercept of 0.5104924 and slope of centered dose of 48.62181. Let us now have a look at the differences between the standard and the new model by using glm.fitted:

```
> glm.fitted(y, X, n, family="logit", tail="left", psi2=0.15 ,eta0=0)
Logit GLM with  left  modified
Observed vs Fitted Mean Plot
```

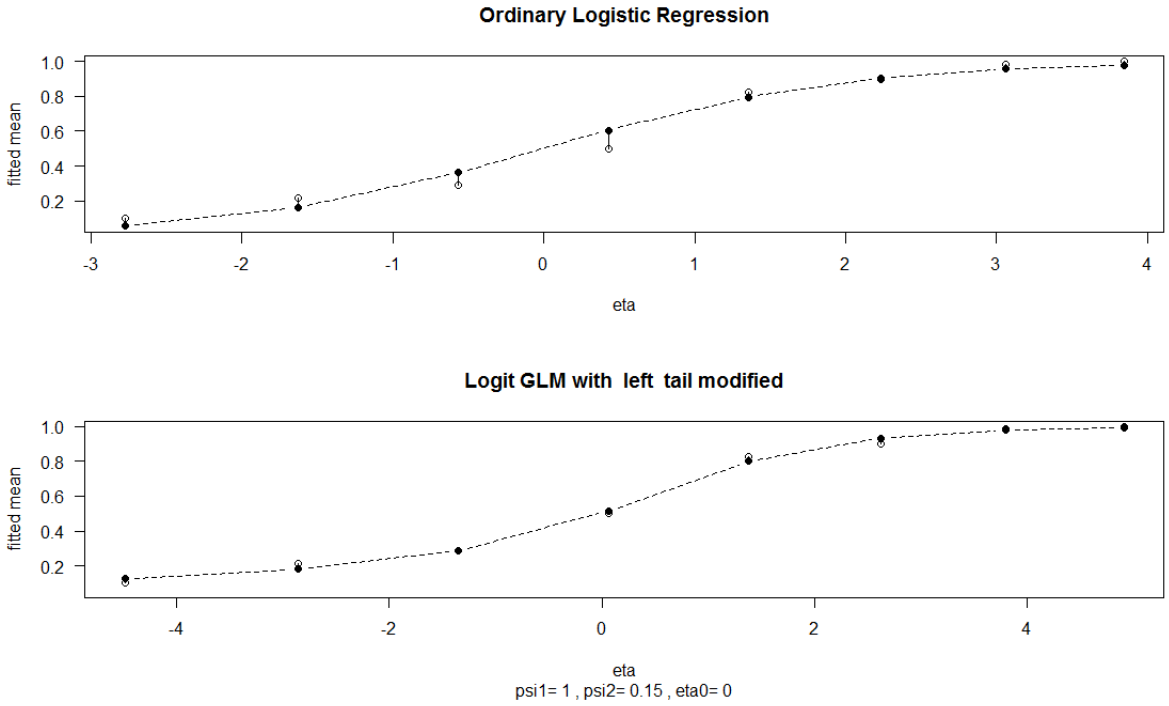

Figure 20: Observed vs Fitted Mean Plots for Beetle data set

While in the first plot of the Ordinary Logistic Regression the white points on the left side, which are the observed means, do not lie on the dotted line, in the second plot of the tail transformed generalized linear model nearly every observation mean lies exactly over the corresponding fitted mean, which are the black points.
More information about regression parameters is provided by glm.inf:

```
> glm.inf(y, X, n, family="logit", tail="left", psi2=0.15, eta0=0)
Logit GLM with  left  tail modified, psi2 =  0.15

Regression Parameters:
             coeff  fixed se  fixed z  estimated se  estimated z  var inflation
(Intercept)  0.5105   0.17368    2.939        0.1906        2.678            9.7
dose.cen    48.6218   5.51787    8.812        7.1157        6.833           29.0

Link Parameter:
psi2 =  0.15 , standard error of psi2 =  0.2458

Correlation matrix with estimated link:
            (Intercept) dose.cen    psi2
(Intercept)      1.0000  -0.3546  0.4210
dose.cen        -0.3546   1.0000 -0.6429
psi2             0.4210  -0.6429  1.0000
```

The correlation matrix shows a relatively low correlation of regression and link parameters.

Incidence of Byssinosis
The second logistic generalized linear model that we arranged in section 2.4 was on the data set of Byssinosis-incidence among Cotton Textile Workers. Here we use a left tail transformation, too.
X is again the design matrix which I created from the covariates before using it in the glm-functions.

```
> glm.profile(y, X, n, family="logit", tail="left", psivec=seq(0.25,0.5,by=0.05),eta0=0)
Minimal Deviance  20.79339 at the optimal psi-value  0.3
```

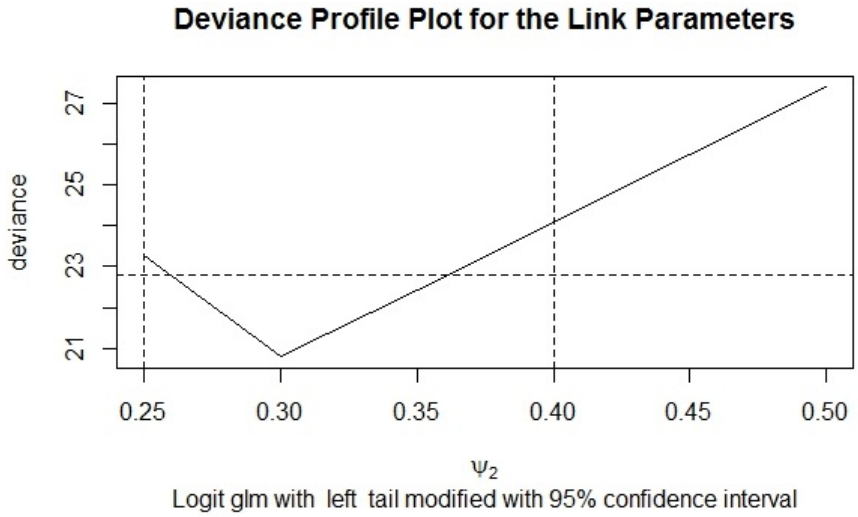

Figure 21: Profile Deviance Plot for ψ_2 for the Byssinosis data

The glm.profile gives us the result that the minimal deviance was attained at $\psi_2 = 0.3$. $\psi_2 = 1$ lies way beyond the 95%-Confidence Interval. This confirms again the poor fit of the standard model from section 2.4.
We fit the model with this ψ_2-value:

```
> glm.model(y, X, n, family="logit", tail="left", psi2=0.3, eta0=0)
Fitting Logit GLM with  left  tail modified, psi =  0.3

Coefficients:
  (Intercept) workspace  smoking employment
    -12.26971 -7.175305 2.648632   1.101333

Residual Deviance:  20.79339  on  14  degrees of freedom
```

A residual deviance of 20.79339 on 14 degrees of freedom is way better than the residual deviance of 40.774 from the standard generalized linear model before.
glm.fitted shows the improvement the left tail transformation has achieved:

```
> glm.fitted(y, X, n, family="logit", tail="left", psi2=0.3 ,eta0=0)
Logit GLM with  left  modified
Observed vs Fitted Mean Plot
```

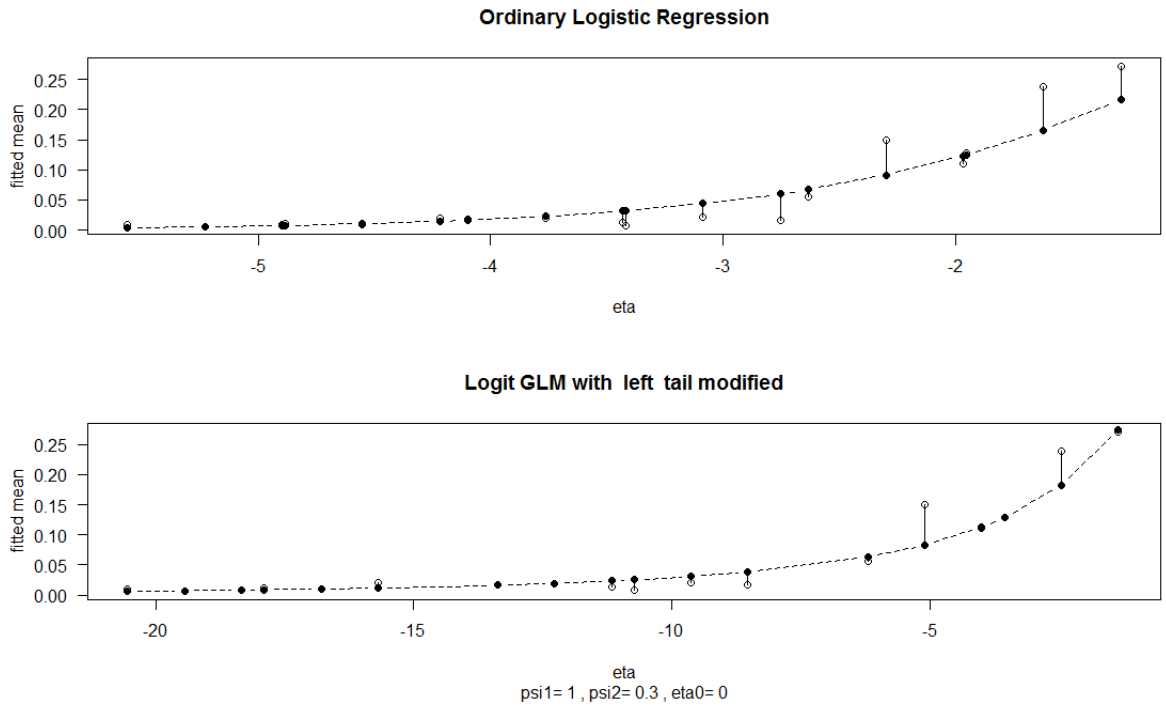

Figure 22: Observed vs Fitted Mean Plots for Byssinosis data set

The observed values smaller than $\eta_0 = 0$ lie nearer to the dotted line in the second plot of the tail transformed generalized linear model than in the first plot of the Ordinary Logistic Regression.
To get the standard errors of the link and regression parameters and their correlation matrix we call glm.inf:

```
> glm.inf(y, X, n, family="logit", tail="left", psi2=0.3, eta0=0)
Logit GLM with  left  tail modified, psi2 =  0.3

Regression Parameters:
               coeff  fixed se  fixed z  estimated se  estimated z  var inflation
(Intercept) -12.2697   0.84588  -14.505      592.2183       -0.021        69912.1
workspace    -7.1753   0.60742  -11.813      605.3709       -0.012        99562.7
smoking       2.6486   0.80229    3.301      449.4200        0.006        55917.2
employment    1.1013   0.28408    3.877      418.1685        0.003       147101.0

Link Parameter:
psi2 =  0.3 , standard error of psi2 =  0.0082

Correlation matrix with estimated link:
            (Intercept) workspace smoking employment    psi2
(Intercept)      1.0000    1.2442  0.2321     0.3855  0.1805
workspace        1.2442    1.0000 -0.0405    -0.3141  0.2752
smoking          0.2321   -0.0405  1.0000    -0.3883 -0.1798
employment       0.3855   -0.3141 -0.3883     1.0000  0.1442
psi2             0.1805    0.2752 -0.1798     0.1442  1.0000
```

Rotifer Suspension
For the Rotifer Suspension data we use the probit family and a both tail transformation because the standard `glm` failed completely on both tails.

```
> glm.profile(y, X, n, family="probit", tail="both", psivec=seq(-0.5,1.2,by=0.05),eta0=0)
Minimal Deviance 253.5838 at the optimal psi-values psi1 = 0 and psi2 = -0.5
```

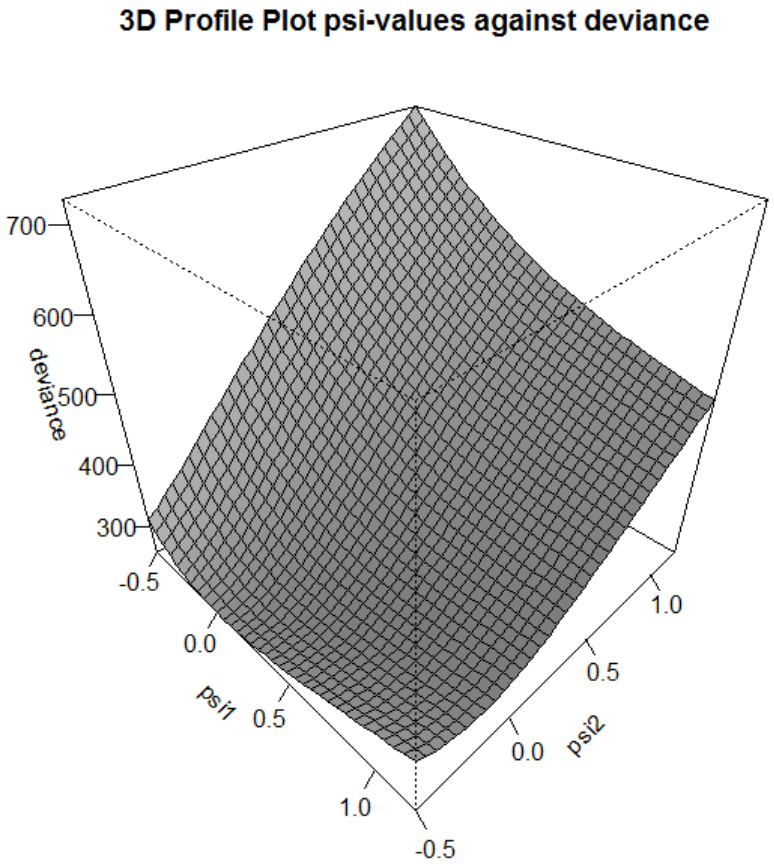

Figure 23: 3D Perspective Plot of ψ_1, ψ_2 and the deviance for the Rotifer data

The profile function tells us that at $\psi_1 = 0$ and $\psi_2 = -0.5$ the deviance attains its minimal value 253.5838. This is a bad result for a model on 36 degrees of freedom but notwithstanding an improvement to the standard model.

```
> glm.model(y, X, n, family="probit", tail="both", psi1=0, psi2=-0.5, eta0=0)
Fitting Probit GLM with  both  tail modified, psi1 =  0  and psi2 =  -0.5

Coefficients:
  (Intercept)  species density.cen speciesanddensity.cen
       -2.606 3.509845    2.729062             -1.229784

Residual Deviance:  253.5838  on  36  degrees of freedom
```

Improvements of the tail transformation can be observed in the following plots:

```
> glm.fitted(y, X, n, family="probit", tail="both", psi1=0, psi2=-0.5, eta0=0)
Probit GLM with  both  modified
Observed vs Fitted Mean Plot
```

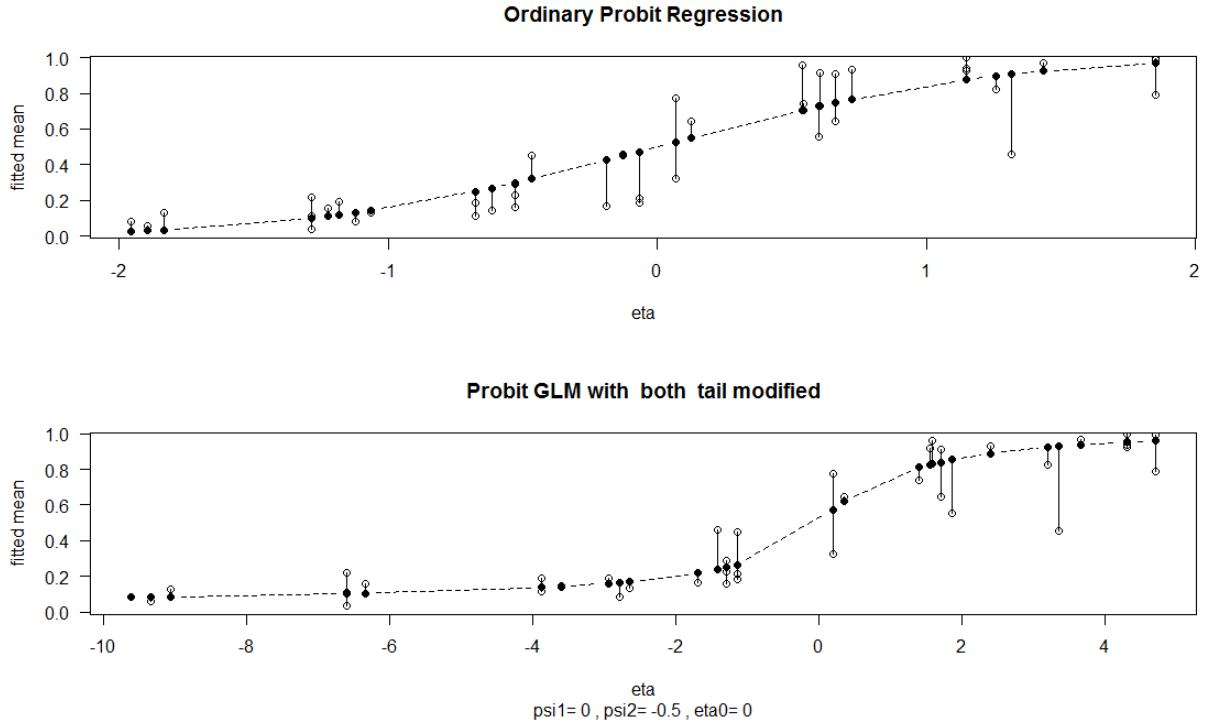

Figure 24: Observed vs Fitted Mean Plots for Rotifer data set

Standard errors and correlation matrix computed by the glm.inf function:

```
 > glm.inf(y, X, n, family="probit", tail="both", psi1=0, psi2=-0.5, eta0=0)
Probit GLM with  both  tail modified, psi1 =  0  and psi2 =  -0.5

Regression Parameters:
                         coeff  fixed se  fixed z  estimated se  estimated z  var inflation
(Intercept)           -2.6060   0.26239   -9.932        0.5057       -5.153   92.7
species                3.5098   0.28851   12.165        0.6045        5.806          109.5
density.cen            2.7291   0.19990   13.652        0.4649        5.870  132.6
speciesanddensity.cen -1.2298   0.23358   -5.265        0.3506       -3.508   50.1

Link Parameters:
psi1 =  0 , standard error of psi1 =  0.0998
psi2 =  -0.5 , standard error of psi2 =  0.058

Correlation matrix with estimated link:
                     (Intercept) species density.cen speciesanddensity.cen    psi1    psi2
(Intercept)               1.0000 -0.9576     -0.8859                0.8218  0.5315  0.7742
species                  -0.9576  1.0000      0.9147               -0.7453 -0.6786 -0.7709
density.cen              -0.8859  0.9147      1.0000               -0.8442 -0.7524 -0.7727
speciesanddensity.cen     0.8218 -0.7453     -0.8442                1.0000  0.3840  0.6502
psi1                      0.5315 -0.6786     -0.7524                0.3840  1.0000  0.5468
psi2                      0.7742 -0.7709     -0.7727                0.6502  0.5468  1.0000
```

4.3 Poisson Random Component

Coal Mining Fractures
The last data set are the Coal Mining Fractures. Family is Poisson and we use right tail transformation.

```
> glm.profile(y, X, family="poiss", tail="right", psivec=seq(-1,1,by=0.065), eta0=0)
Minimal Deviance  30.80365 at the optimal psi-value  -0.61
```

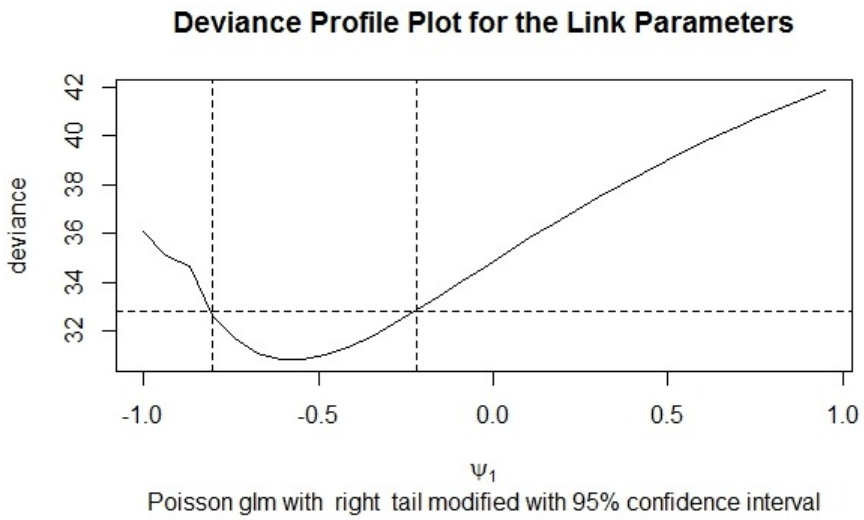

Figure 25: Profile Deviance Plot for ψ_1 for the Mining data

The profile plot shows that a negative ψ_1 of -0.61 was necessary to receive the minimal deviance of 30.80365. The standard parameter $\psi_1 = 1$ lies way out of the 95%-Confidence Interval. This shows how much the tail transformation was needed.
Fitting the model with $\psi_1 = -0.61$ gives us the following results:

```
> glm.model(y, X, family="poiss", tail="right", psi1=-0.61)
Fitting Poisson GLM with  right  tail modified, psi =  -0.61

Coefficients:
  (Intercept)     inb.cen extrp.cen
     3.428951 -0.01112052 0.3964859

Residual Deviance:  30.80365  on  41  degrees of freedom
```

The deviance has now greatly improved compared to the value of 42.094 in the standard generalized linear model. The new model fits better, which can be seen in the Oberved versus Fitted mean Plots:

```
> glm.fitted(y, X, family="poiss", tail="right", psi1=-0.61, eta0=0)
Poisson GLM with right modified
Observed vs Fitted Mean Plot
```

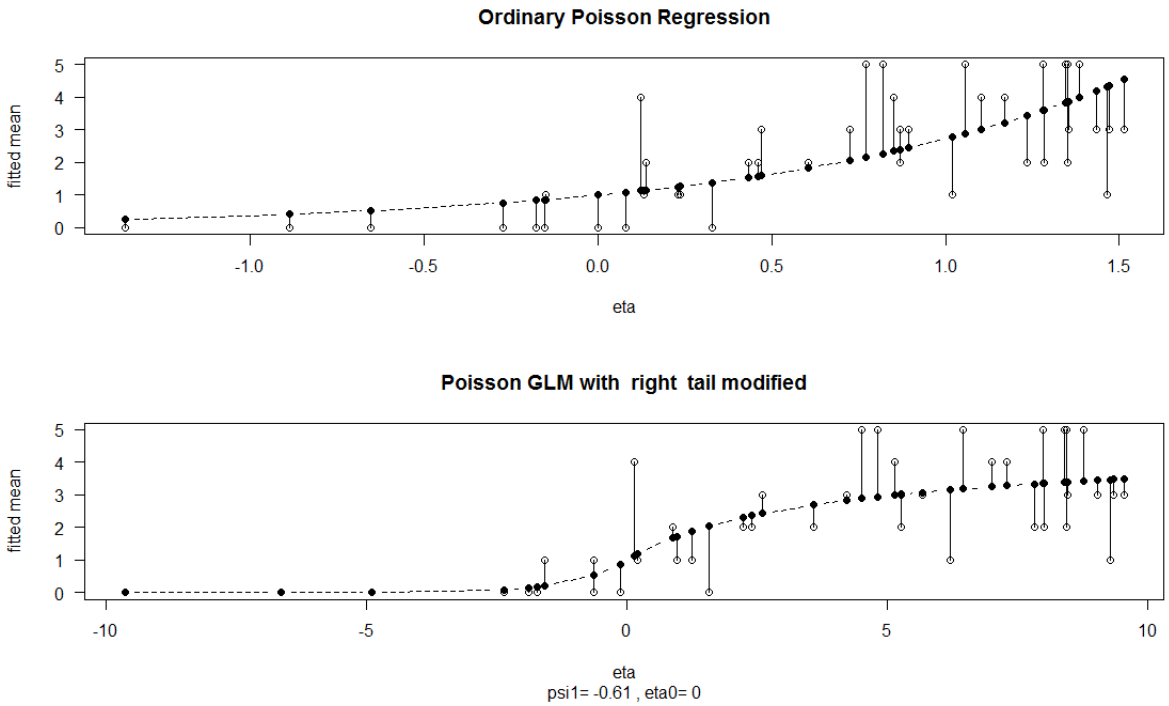

Figure 26: Observed vs Fitted Mean Plots for Mining data set

Standard errors and correlation matrix are provided in glm.inf:

```
> glm.inf(y, X, family="poiss", tail="right", psi1=-0.61, eta0=0)
Poisson GLM with  right  tail modified, psi1 =  -0.61

Regression Parameters:
             coeff  fixed se  fixed z  estimated se  estimated z  var inflation
(Intercept)  3.4290   2.37814    1.442        0.8611        3.982          -63.8
inb.cen     -0.0111   0.00718   -1.546        0.0029       -3.828          -59.6
extrp.cen    0.3965   0.26464    1.498        0.0988        4.013          -62.7

Link Parameter:
psi1 =  -0.61 , standard error of psi1 =  0.052

Correlation matrix with estimated link:
            (Intercept) inb.cen extrp.cen    psi1
(Intercept)      1.0000  1.2108   -1.0663  2.0584
inb.cen          1.2108  1.0000    1.1293 -2.3580
extrp.cen       -1.0663  1.1293    1.0000  2.0823
psi1             2.0584 -2.3580    2.0823  1.0000
```

Applying the tail transformation to the generalized linear models has improved the deviance in each of the five data sets used in this thesis. The generalized linear models with tail transformation always give better results than the standard R-specific `glm`-function.

5 Conclusion

The purpose of this thesis was to implement a family of parametric link functions that perform a tail transformation to improve generalized linear models in R. Generalized linear models are essential models in statistical analyses. There are many data sets that do not satisfy the assumptions for a linear model, i.e. normality, homoscedasticity and linearity. For all of the huge amount of these situations the generalized linear models can be applied. They are based on the exponential family of distributions and use the likelihood principle, which we saw in the first sections where theoretical background on the mathematical properties of a generalized linear model with parametric link function was provided.
All these new functions to fit generalized linear models were implemented in R. R is a highly flexible programming language for data analyses and statistics. It is commonly used because it has several great advantages over other statistical languages: R is an Open Source Software and is executable on nearly every operating system, it provides interfaces to other programming languages and has a huge selection of already implemented algorithm and methods.
One of them is the glm-function, which is installed along with the R-Environment by default. In section 2.4 we applied this R-specific function on some data sets and learned that this standard method is not entirely satisfying and in rare cases even failed by returning bad results. That's why we want to use parametric link functions instead of the commonly used standard links. Functions that create general linear models with tail transformation were presented in chapter 3 along with some pecularities of their implementation.
In the last section, where we applied the new functions on the example data sets, the Observed versus Fitted mean plots have given a good overview over the improvements the tail transformations have achieved.
But also the residual deviance has improved much.
In the following table we compare the residual deviance of the standard GLM with the one of the GLM with tail transformation and the probability that they come from a χ^2-distribution. The last column gives the percentage of improvement of the deviance.

			Standard GLM		**GLM with tail trafo**		**Percentage of**
data set	**dgf**	$\chi^2_{\mathbf{dgf}}$	**Res. dev.**	**prob.**	**Res. dev.**	**prob.**	**deviance – improvement**
pcb.ex	26	χ^2_{26}	8.359	99.959%	6.237025	99.997%	25.4%
beetle.ex	6	χ^2_6	11.232	8.147%	3.04776	80.283%	72.9%
bys.ex	14	χ^2_{14}	40.774	0.019%	20.79339	10.705%	49%
rotifer.ex	36	χ^2_{36}	434.02	0%	253.5838	0%	41.6%
mining.ex	41	χ^2_{41}	42.09	42.353%	30.80365	87.706%	26.8%

Table 8: Deviance table to compare standard GLM and GLM with tail transformation

This table shows that even with data sets, where the R-specific glm function gives moderately good results, we can achieve great improvements by using a tail transformation.
The best example is the Beetle data set. We had a residual deviance of 11.232 and a probability of 8.147% that this comes from the χ^2_6-distribution at the beginning. The generalized linear model with left tail transformation gives us a deviance of 3.04776,

which is an improvement of 72.9%. The probability that the deviance comes from the χ^2_6-distribution rose to 80.283%.
Even a already very good model like the standard model of the PCB data could be improved by 25.4%. But as we have seen there are data sets where the standard `glm` function simply fails and the tail transformations are a good possibility to get a somewhat satisfying model.
If we take the Byssinosis data for example, the `R`-specific `glm` yields absolutely no pleasing result. By applying a left tail transformation to the Byssinosis data set our model is slightly better but still not satisfying. Anyway we achieved a deviance-improvement of 49%, though.
The Rotifer data set had the worst standard model. The probability that the deviance follows a χ^2_{36}-distribution was zero. The both tail transformation was needed. The probability to follow a χ^2-distribution remained 0%, but we achieved a deviance improvement of 41.6% at least. On the other hand the both tail transformation is an elaborate calculation and especially the `glm.profile` takes quite some computing time.
The Poisson data set had a moderate deviance at the beginning, but by the tail transformation we obtained a model that is pretty well fixed.
To sum up, I can say that it depends on the data set if the use of a tail transformation is sensible. The more observations are in a data set the more complex and tough is the tail transformation. However, there exist many data sets that have an offset at the tails when a standard generalized linear model was applied to them. When you come to think of it the tails or the extreme values of an observation rarely match the main measured data. Because of that there is a wide field of application for the tail transformations. They will be useful in nearly every data set one might have to analyse. In addition the tail modifications are relatively easy to use in generalized linear models by the link functions.

References

[1] Bates, D. and Watts, D. (1988). *Nonlinear Regression Analysis and its Applications*. Wiley, New York.

[2] Bliss, C. (1935). The calculation of the dose-mortality curve. *Annals of Applied Biology*, 22:p. 134–67.

[3] Collett, D. (1991). *Modelling Binary Data*. Chapman and Hall, London.

[4] Czado, C. (2007). Fitting generalized linear models with parametric link in s. York University, North York, Ontario, Canada.

[5] Czado, C., Krämer, N., and Brechmann, E. C. (2012). *Generalized linear models with applications*.

[6] Dey, D. K., Ghosh, S. K., and Mallick, B. K. (2000). *Generalized Linear Models - A Bayesian Perspective*. Marcel Dekker, Inc., New York.

[7] Dobson, A. J. and Barnett, A. G. (2008). *An Introduction to Generalized Linear Models*. Chapman and Hall/CRC, Boca Raton, 3rd edition.

[8] Higgins, J. and Koch, G. (1977). Variable selection and generalized chi-square analysis of categorial data applied to a large cross sectional occupational health survey. *Int. Statist. Rev.*, 45:p. 51–62.

[9] Lee, Y. (2006). *Generalized Linear Models with Random Effects*. Chapman and Hal/CRCl, Boca Raton, 2nd edition.

[10] McCullagh, P. (1999). *Generalized Linear Models*. Chapman and Hall, London, 2nd edition.

[11] Myers, R. (1990). *Classical and Modern Regression with Applications*. PWS-Kent Publishing Company, Boston, 2nd edition.

[12] Wood, S. N. (2006). *Generalized Additive Models - An Introduction with R*. Chapman and Hall/CRC, Boca Raton.

A R - Code

A.1 Hpsi - functions

```
############
# hpsi1.r #
############
# general h()-power transformation to induce right tail modification

hpsi1 <- function(psi1, eta, eta0 = 0) {
    h <- 1:length(eta)
    h[eta < eta0] <- eta[eta < eta0]
    if (any(psi1 > -1e-14 && psi1 < 1e-14)) { # limit case
        h[eta >= eta0] <- eta0 + log(eta[eta >= eta0] - eta0 + 1)    }
    else {
        h[eta >= eta0] <- ((eta[eta >= eta0] - eta0 + 1)^psi1 - 1)/psi1
        h[eta >= eta0] <- h[eta >= eta0] + eta0     }
    h
}

############
# hpsi2.r #
############
# general h()-power transformation to induce left tail modification

hpsi2 <- function (psi2, eta, eta0 = 0) {
    h <- 1:length(eta)
    h[eta >= eta0] <- eta[eta >= eta0]
    if (any(psi2 > -1e-14 && psi2 < 1e-14)) {  # limit case
        h[eta < eta0] <- eta0 - log(-(eta[eta < eta0]) + eta0 + 1)  }
    else {
        h[eta < eta0] <- -((- eta[eta < eta0] + eta0 + 1)^psi2 - 1)/psi2
        h[eta < eta0] <- h[eta < eta0] + eta0    }
    h
}

#############
# hpsi12.r #
#############
# general h()-power transformation to induce both tail modification

hpsi12 <- function (psi1, psi2, eta, eta0 = 0) {
    h <- 1:length(eta)
    if (any(psi1 > -1e-14 && psi1 < 1e-14)) {  # 1st limit case
        h[eta >= eta0] <- eta0 + log(eta[eta >= eta0] - eta0 + 1)  }
    else {
        h[eta >= eta0] <- ((eta[eta >= eta0] - eta0 + 1)^psi1 - 1)/psi1
        h[eta >= eta0] <- h[eta >= eta0] + eta0     }
    if (any(psi2 > -1e-14 && psi2 < 1e-14)) {  # 2nd limit case
        h[eta < eta0] <- eta0 - log(-(eta[eta < eta0]) + eta0 + 1) }
    else {
        h[eta < eta0] <- -((- eta[eta < eta0] + eta0 + 1)^psi2 - 1)/psi2
        h[eta < eta0] <- h[eta < eta0] + eta0       }
    h
}
```

A.2 GLM-functions with tail transformation

glm.model

```
###############
# glm.model.r #
###############
# Fits GLM with tail transformation and gives beta-values and residual deviance

glm.model <- function(y, X, n, tail, family, psi1=1, psi2=1, eta0=0){
  family.int <- charmatch(family, c("gauss", "poiss", "logit", "probit"))
if (family.int == 1){ # Gauss
    model.gauss(y=y, X=X, tail=tail, psi1=psi1, psi2=psi2, eta0=eta0) }
  else if (family.int == 2){ # Poisson
    model.poisson(y=y, X=X, tail=tail, psi1=psi1, psi2=psi2, eta0=eta0)}
  else if (family.int == 3){ # Logit
    model.logit(y=y, X=X, n=n, tail=tail, psi1=psi1, psi2=psi2, eta0=eta0)}
  else if (family.int == 4){# Probit
    model.probit(y=y, X=X, n=n, tail=tail, psi1=psi1, psi2=psi2, eta0=eta0) }
  else {
    stop("Please select gauss, poisson, logit or probit family")}
}

#################
# model.gauss.r #
#################
# Gaussian GLM with tail transformation

model.gauss <- function(y, X, tail, psi1, psi2, eta0=0){
  out <- list()
  n <- length(y)
  p <- ncol(X)
  tail.int <- charmatch(tail, c("right", "left", "both"))

  neg.loglike.hpsi1.gauss <- function(pair.beta.sigma, y, X, psi1, eta0){
    beta <- pair.beta.sigma[1:p]
    logsigma2 <- pair.beta.sigma[p+1]
    eta <- X %*% beta
    hpsi1vec <- hpsi1(psi1, eta, eta0)
    temp <- y - hpsi1vec
    out <- (n * 0.5 * log(2 * pi * exp(logsigma2))) + (t(temp) %*% temp)/(2 * exp(logsigma2))
    out
  }
neg.loglike.hpsi2.gauss <- function(pair.beta.sigma, y, X, psi2, eta0){
    beta <- pair.beta.sigma[1:p]
    logsigma2 <- pair.beta.sigma[p+1]
    eta <- X %*% beta
    hpsi2vec <- hpsi2(psi2, eta, eta0)
    temp <- y - hpsi2vec
    out <- (n * 0.5 * log(2 * pi * exp(logsigma2))) + (t(temp) %*% temp)/(2 * exp(logsigma2))
    out
}
neg.loglike.hpsi12.gauss <- function(pair.beta.sigma, y, X, psi1, psi2, eta0){
    beta <- pair.beta.sigma[1:p]
    logsigma2 <- pair.beta.sigma[p+1]
    eta <- X %*% beta
    hpsi12vec <- hpsi12(psi1, psi2, eta, eta0)
    temp <- y - hpsi12vec
    out <- (n * 0.5 * log(2 * pi * exp(logsigma2))) + (t(temp) %*% temp)/(2 * exp(logsigma2))
    out
  }
# compute beta.opt at psi1/psi2 and deviance
  if (tail.int == 1){ # Right tail
    par.opt <- optim(par=rep(0,p+1), fn = neg.loglike.hpsi1.gauss, method="BFGS", y=y, X=X, psi1=psi1, eta0=eta0)$par
    beta.opt <- par.opt[1:p]
    cat("Fitting Gaussian GLM with ",tail," tail modified, psi = ", psi1, "\n\n")
    eta.opt <- X %*% beta.opt
    hpsi1vec.opt <- hpsi1(psi1, eta.opt, eta0)
    temp.opt <- y - hpsi1vec.opt
```

```
    res.dev <- t(temp.opt) %*% temp.opt
  }
  else if (tail.int == 2){ # Left tail
    par.opt <- optim(par=rep(0,p+1), fn = neg.loglike.hpsi2.gauss, method="BFGS", y=y, X=X, psi2=psi2, eta0=eta0)$par
    beta.opt <- par.opt[1:p]
    cat("Fitting Gaussian GLM with ",tail," tail modified, psi = ", psi2, "\n\n")
    eta.opt <- X %*% beta.opt
    hpsi2vec.opt <- hpsi2(psi2, eta.opt, eta0)
    temp.opt <- y - hpsi2vec.opt
    res.dev <- t(temp.opt) %*% temp.opt
  }
  else if (tail.int == 3){ # Both tail
    par.opt <- optim(par=rep(0,p+1), fn = neg.loglike.hpsi12.gauss, method="BFGS", y=y, X=X,
                     psi1=psi1, psi2=psi2, eta0=eta0)$par
    beta.opt <- par.opt[1:p]
    cat("Fitting Gaussian GLM with ",tail," tail modified, psi 1 = ", psi1, " and psi 2 = ", psi2, "\n\n")
    eta.opt <- X %*% beta.opt
    hpsi12vec.opt <- hpsi12(psi1, psi2, eta.opt, eta0)
    temp.opt <- y - hpsi12vec.opt
    res.dev <- t(temp.opt) %*% temp.opt
  }
  else {
    stop("Please select tail")}
cat("Coefficients:","\n")
  mat <- t(as.matrix(cbind(beta.opt)))
  covariates <- rep("0",p)
  covariates[1] <- "(Intercept)"
  if (is.null(dimnames(X)[[2]])){
    for (i in 1:(p-1)){
      covariates[i+1] <- paste("Covariate",i)}
  }
else {
    temp <- dimnames(X)[[2]]
    covariates[2:p] <- temp[2:p]}
  prmatrix(mat, quote=F, rowlab=" ", collab=covariates)
  cat("\nResidual Deviance: ", res.dev, " on ", length(y)-ncol(X), " degrees of freedom")
}

#################
# model.logit.r #
#################
# Logit GLM with tail transformation

model.logit <- function(y, X, n, tail, psi1, psi2, eta0=0){
  out <- list()
  p <- ncol(X)
  fails <- n-y
  fails[fails == 0] <- 10^(-16)
  y[y == 0] <- 10^(-16)
  tail.int <- charmatch(tail, c("right", "left", "both"))

  neg.loglike.hpsi1.logit <- function(b, y, n, X, psi1, eta0){
    eta <- X %*% b
    expon.hpsi1vec <- exp(hpsi1(psi1, eta, eta0))
    Fhpsi1vec <- (expon.hpsi1vec)/(1 + expon.hpsi1vec)
    out <- -sum(y * log(Fhpsi1vec) + (n - y) * log(1 - Fhpsi1vec))
    out
  }
neg.loglike.hpsi2.logit <- function(b, y, n, X, psi2, eta0){
    eta <- X %*% b
    expon.hpsi2vec <- exp(hpsi2(psi2, eta, eta0))
    Fhpsi2vec <- (expon.hpsi2vec)/(1 + expon.hpsi2vec)
    out <- -sum(y * log(Fhpsi2vec) + (n - y) * log(1 - Fhpsi2vec))
    out
  }
neg.loglike.hpsi12.logit <- function(b, y, n, X, psi1, psi2, eta0){
    eta <- X %*% b
    expon.hpsi12vec <- exp(hpsi12(psi1, psi2, eta, eta0))
    Fhpsi12vec <- (expon.hpsi12vec)/(1 + expon.hpsi12vec)
    out <- -sum(y * log(Fhpsi12vec) + (n - y) * log(1 - Fhpsi12vec))
```

```
    out
  }
# compute beta.opt at psi1/psi2 and deviance
  if (tail.int == 1){ # Right tail
    beta.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi1.logit, method="BFGS", y=y, X=X, n=n,
                      psi1=psi1, eta0=eta0)$par
    cat("Fitting Logit GLM with ",tail," tail modified, psi = ", psi1, "\n\n")
    eta.opt <- X %*% beta.opt
    expon.hpsi1.opt <- exp(hpsi1(psi1, eta.opt, eta0))
    F.opt <- (expon.hpsi1.opt)/(1 + expon.hpsi1.opt)
    res.dev <- 2*sum(y * log(y/(n*F.opt)) - (y - n) * log((-fails)/(n*F.opt - n)))
  }
  else if (tail.int == 2){ # Left tail
    beta.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi2.logit, method="BFGS", y=y, X=X, n=n,
                      psi2=psi2, eta0=eta0)$par
    cat("Fitting Logit GLM with ",tail," tail modified, psi = ", psi2, "\n\n")
    eta.opt <- X %*% beta.opt
    expon.hpsi2.opt <- exp(hpsi2(psi2, eta.opt, eta0))
    F.opt <- (expon.hpsi2.opt)/(1 + expon.hpsi2.opt)
    res.dev <- 2*sum(y * log(y/(n*F.opt)) - (y - n) * log((-fails)/(n*F.opt - n)))
  }
  else if (tail.int == 3){ # Both tail
    beta.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi12.logit, method="BFGS", y=y, X=X, n=n,
                      psi1=psi1, psi2=psi2, eta0=eta0)$par
    cat("Fitting Logit GLM with ",tail," tail modified, psi 1 = ", psi1, " and psi 2 = ", psi2, "\n\n")
    eta.opt <- X %*% beta.opt
    expon.hpsi12.opt <- exp(hpsi12(psi1, psi2, eta.opt, eta0))
    F.opt <- (expon.hpsi12.opt)/(1 + expon.hpsi12.opt)
    res.dev <- 2*sum(y * log(y/(n*F.opt)) - (y - n) * log((-fails)/(n*F.opt - n)))
  }
  else {
    stop("Please select tail")}
cat("Coefficients:","\n")
  mat <- t(as.matrix(cbind(beta.opt)))
  covariates <- rep("0",p)
  covariates[1] <- "(Intercept)"
  if (is.null(dimnames(X)[[2]])){
    for (i in 1:(p-1)){
      covariates[i+1] <- paste("Covariate",i)}
  }
  else {
    temp <- dimnames(X)[[2]]
    covariates[2:p] <- temp[2:p]}
  prmatrix(mat, quote=F, rowlab=" ", collab=covariates)
  cat("\nResidual Deviance: ", res.dev, " on ", length(y)-ncol(X), " degrees of freedom")
}

##################
# model.probit.r #
##################
# Probit GLM with tail transformation

model.probit <- function(y, X, n, tail, psi1, psi2, eta0=0){
  out <- list()
  p <- ncol(X)
  fails <- n-y
  fails[fails == 0] <- 10^(-16)
  y[y == 0] <- 10^(-16)
  tail.int <- charmatch(tail, c("right", "left", "both"))

  neg.loglike.hpsi1.probit <- function(b, y, n, X, psi1, eta0){
    eta <- X %*% b
    Fhpsi1vec <- pnorm(hpsi1(psi1, eta, eta0), mean=0, sd=1)
    out <- -sum(y * log(Fhpsi1vec) + (n - y) * log(1 - Fhpsi1vec))
    out
  }
neg.loglike.hpsi2.probit <- function(b, y, n, X, psi2, eta0){
    eta <- X %*% b
    Fhpsi2vec <- pnorm(hpsi2(psi2, eta, eta0), mean=0, sd=1)
    out <- -sum(y * log(Fhpsi2vec) + (n - y) * log(1 - Fhpsi2vec))
```

```
    out
  }
neg.loglike.hpsi12.probit <- function(b, y, n, X, psi1, psi2, eta0){
    eta <- X %*% b
    Fhpsi12vec <- pnorm(hpsi12(psi1, psi2, eta, eta0), mean=0, sd=1)
    out <- -sum(y * log(Fhpsi12vec) + (n - y) * log(1 - Fhpsi12vec))
    out
  }
# compute beta.opt at psi1/psi2 and deviance
  if (tail.int == 1){ # Right tail
    beta.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi1.probit, method="BFGS", y=y, X=X, n=n,
                      psi1=psi1, eta0=eta0)$par
    cat("Fitting Probit GLM with ",tail," tail modified, psi1 = ", psi1, "\n\n")
    eta.opt <- X %*% beta.opt
    F.opt <- pnorm(hpsi1(psi1, eta.opt, eta0), mean=0, sd=1)
    res.dev <- 2*sum(y * log(y/(n*F.opt)) - (y - n) * log((-fails)/(n*F.opt - n)))
  }
  else if (tail.int == 2){ # Left tail
    beta.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi2.probit, method="BFGS", y=y, X=X, n=n,
                      psi2=psi2, eta0=eta0)$par
    cat("Fitting Probit GLM with ",tail," tail modified, psi2 = ", psi2, "\n\n")
    eta.opt <- X %*% beta.opt
    F.opt <- pnorm(hpsi2(psi2, eta.opt, eta0), mean=0, sd=1)
    res.dev <- 2*sum(y * log(y/(n*F.opt)) - (y - n) * log((-fails)/(n*F.opt - n)))
  }
  else if (tail.int == 3){ # Both tail
    beta.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi12.probit, method="BFGS", y=y, X=X, n=n,
                      psi1=psi1, psi2=psi2, eta0=eta0)$par
    cat("Fitting Probit GLM with ",tail," tail modified, psi1 = ", psi1, " and psi2 = ", psi2, "\n\n")
    eta.opt <- X %*% beta.opt
    F.opt <- pnorm(hpsi12(psi1, psi2, eta.opt, eta0), mean=0, sd=1)
    res.dev <- 2*sum(y * log(y/(n*F.opt)) - (y - n) * log((-fails)/(n*F.opt - n)))
  }
  else {
    stop("Please select tail") }
cat("Coefficients:","\n")
  mat <- t(as.matrix(cbind(beta.opt)))
  covariates <- rep("0",p)
  covariates[1] <- "(Intercept)"
  if (is.null(dimnames(X)[[2]])){
    for (i in 1:(p-1)){
      covariates[i+1] <- paste("Covariate",i)}
  }
  else {
    temp <- dimnames(X)[[2]]
    covariates[2:p] <- temp[2:p]}
  prmatrix(mat, quote=F, rowlab=" ", collab=covariates)
  cat("\nResidual Deviance: ", res.dev, " on ", length(y)-ncol(X), " degrees of freedom")
}

###################
# model.poisson.r #
###################
# Poisson GLM with tail transformation

model.poisson <- function(y, X, tail, psi1, psi2, eta0=0){
  out <- list()
  n <- length(y)
  p <- ncol(X)
  y[y == 0] = 10^(-16)
  tail.int <- charmatch(tail, c("right", "left", "both"))

  neg.loglike.hpsi1.poiss <- function(b, y, X, psi1, eta0){
    eta <- X %*% b
    Fhpsi1vec <- exp(hpsi1(psi1, eta, eta0))
    out <- -sum(y * hpsi1(psi1, eta, eta0) - Fhpsi1vec)
    out
  }
# compute beta.opt at psi1 and deviance
  if (tail.int == 1){ # Right tail
```

```
    beta.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi1.poiss, method="BFGS", y=y, X=X, psi1=psi1, eta0=eta0)$par
    cat("Fitting Poisson GLM with ",tail," tail modified, psi = ", psi1, "\n\n")
    eta.opt <- X %*% beta.opt
    F.opt <- exp(hpsi1(psi1, eta.opt, eta0))
    res.dev <- 2*sum(y * log(y/F.opt) - y + F.opt)
  }
else {
    stop("Poisson only available with right tail modification")}
cat("Coefficients:","\n")
  mat <- t(as.matrix(cbind(beta.opt)))
  covariates <- rep("0",p)
  covariates[1] <- "(Intercept)"
  if (is.null(dimnames(X)[[2]])){
    for (i in 1:(p-1)){
      covariates[i+1] <- paste("Covariate",i)}
  }
  else {
    temp <- dimnames(X)[[2]]
    covariates[2:p] <- temp[2:p]
  }
  prmatrix(mat, quote=F, rowlab=" ", collab=covariates)
  cat("\nResidual Deviance: ", res.dev, " on ", length(y)-ncol(X), " degrees of freedom")
}
```

glm.profile

```
##################
# glm.profile.r #
##################
# Plots psi against deviance and finds optimal psi-value to fit the glm

glm.profile <- function(y, X, n, tail, family, psivec=seq(0,2,length=100), eta0=0){
  family.int <- charmatch(family, c("gauss", "poiss", "logit", "probit"))
if (family.int == 1){ # Gauss
    profile.gauss(y=y, X=X, tail=tail, psivec=psivec, eta0=eta0)}
  else if (family.int == 2){ # Poisson
    profile.poisson(y=y, X=X, tail=tail, psivec=psivec, eta0=eta0)}
  else if (family.int == 3){ # Logit
    profile.logit(y=y, X=X, n=n, tail=tail, psivec=psivec, eta0=eta0)}
  else if (family.int == 4){# Probit
    profile.probit(y=y, X=X, n=n, tail=tail, psivec=psivec, eta0=eta0)}
  else {
    stop("Please select gauss, poisson, logit or probit family")}
}

###################
# profile.gauss.r #
###################
# Plots psi against deviance and finds optimal psi-value to fit the glm

profile.gauss <- function(y, X, tail, psivec=seq(0,2,length=100), eta0=0){
  devvec <- rep(0, length(psivec))
  devmat <- matrix(0, length(psivec), length(psivec))
  out <- list()
  n <- length(y)
  p <- ncol(X)
  tail.int <- charmatch(tail, c("right", "left", "both"))

  neg.loglike.hpsi1.gauss <- function(pair.beta.sigma, y, X, psi1, eta0){
    beta <- pair.beta.sigma[1:p]
    logsigma2 <- pair.beta.sigma[p+1]
    eta <- X %*% beta
    hpsi1vec <- hpsi1(psi1, eta, eta0)
    temp <- y - hpsi1vec
    out <- (n * 0.5 * log(2 * pi * exp(logsigma2))) + (t(temp) %*% temp)/(2 * exp(logsigma2))
    out
  }
neg.loglike.hpsi2.gauss <- function(pair.beta.sigma, y, X, psi2, eta0){
    beta <- pair.beta.sigma[1:p]
```

```
    logsigma2 <- pair.beta.sigma[p+1]
    eta <- X %*% beta
    hpsi2vec <- hpsi2(psi2, eta, eta0)
    temp <- y - hpsi2vec
    out <- (n * 0.5 * log(2 * pi * exp(logsigma2))) + (t(temp) %*% temp)/(2 * exp(logsigma2))
    out
  }
neg.loglike.hpsi12.gauss <- function(pair.beta.sigma, y, X, psi1, psi2, eta0){
    beta <- pair.beta.sigma[1:p]
    logsigma2 <- pair.beta.sigma[p+1]
    eta <- X %*% beta
    hpsi12vec <- hpsi12(psi1, psi2, eta, eta0)
    temp <- y - hpsi12vec
    out <- (n * 0.5 * log(2 * pi * exp(logsigma2))) + (t(temp) %*% temp)/(2 * exp(logsigma2))
    out
  }
if (tail.int == 1) { # Right tail Gauss
    for (i in 1:length(psivec)){
      par.opt <- optim(par=rep(0,p+1), fn = neg.loglike.hpsi1.gauss, method="BFGS", y=y, X=X,
                       psi1=psivec[i], eta0=eta0)$par
      beta.opt <- par.opt[1:p]
      sigma2 <- exp(par.opt[p+1])
      eta.opt <- X %*% beta.opt
      hpsi1vec.opt <- hpsi1(psivec[i], eta.opt, eta0)
      temp.opt <- y - hpsi1vec.opt
      devvec[i] <- t(temp.opt) %*% temp.opt #deviance
      whichpsi <- expression(psi[1]) #Which psi for plot axes labels
    }
}
  else if (tail.int == 2) { # Left tail Gauss
    for (i in 1:length(psivec)){
      par.opt <- optim(par=rep(0,p+1), fn = neg.loglike.hpsi2.gauss, method="BFGS", y=y, X=X,
                       psi2=psivec[i], eta0=eta0)$par
      beta.opt <- par.opt[1:p]
      sigma2 <- exp(par.opt[p+1])
      eta.opt <- X %*% beta.opt
      hpsi2vec.opt <- hpsi2(psivec[i], eta.opt, eta0)
      temp.opt <- y - hpsi2vec.opt
      devvec[i] <- t(temp.opt) %*% temp.opt #deviance
      whichpsi <- expression(psi[2]) # Which psi for plot axes labels
    }
  }
  else if (tail.int == 3) { # Both tail Gauss
    for (i in 1:length(psivec)){
      for (j in 1:length(psivec)){
        par.opt <- optim(par=rep(0,p+1), fn = neg.loglike.hpsi12.gauss, method="BFGS", y=y, X=X, psi1=psivec[i],
                 psi2=psivec[j], eta0=eta0)$par
        beta.opt <- par.opt[1:p]
        sigma2 <- exp(par.opt[p+1])
        eta.opt <- X %*% beta.opt
        hpsi12vec.opt <- hpsi12(psivec[i], psivec[j], eta.opt, eta0)
        temp.opt <- y - hpsi12vec.opt
        devmat[i,j] <- t(temp.opt) %*% temp.opt #deviance
      }
    }
  }
  else {
    stop("Please select tail")}
  if (tail.int==3){
    # find optimal psi1 and psi2 and corresponding mininmal deviance
    psi1.opt <- which(devmat == min(devmat), arr.ind = TRUE)[1]
    psi2.opt <- which(devmat == min(devmat), arr.ind = TRUE)[2]
    psi1.optval <- psivec[psi1.opt]
    psi2.optval <- psivec[psi2.opt]
    res.dev <- min(devmat)
    # 3D-Plot psi1 and psi2 against deviance
    persp(psivec, psivec, devmat, theta=45, phi=35, shade=0.75, ltheta=120, xlab = "psi1", ylab = "psi2",
      zlab = "deviance", ticktype="detailed", main = paste("3D Profile Plot psi-values against deviance"))
    # returns the specific psi with minimal deviance
    cat(paste("Minimal Deviance ", round(min(devmat),6), "at the optimal psi-values psi1 = ", psi1.optval,
```

```
            " and psi2 = ", psi2.optval, "\n", sep="\t"))
  }
  else {
    # find optimal psi-value and corresponding minimal deviance
    psi.opt <- psivec[devvec == min(devvec)]
    opt <- which(devvec == min(devvec))[1]
    res.dev <- min(devvec)
    # find 95%-CI for psi
    low.found <- FALSE
    up.found <- FALSE
    psi.ci.low <- psivec[1]
    psi.ci.up <- psivec[length(psivec)]
    for (m in 1:length(psivec)){
      if ((psivec[m] < psi.opt) && (low.found == FALSE)) {
        if (devvec[m] < (min(devvec)+2)){
          psi.ci.low <- psivec[m]
          low.found <- TRUE}
      }
      else if ((psivec[m] > psi.opt) && (up.found == FALSE)) {
        if (devvec[m] > (min(devvec)+2)){
          psi.ci.up <- psivec[m]
          up.found <- TRUE}
}
    }
  # plots psi against deviance with 95%-CI for psi
  plot(psivec, devvec, xlab=whichpsi, ylab="deviance", main="Deviance Profile Plot for the Link Parameters",
     type="l", lty=1, sub=paste("Gaussian glm with ",tail, " tail modified with 95% confidence interval"))
    abline(h = min(devvec) + 2, lty=2)
    abline(v = psi.ci.low, lty=2)
    abline(v = psi.ci.up, lty=2)
    # returns the specific psi with minimal deviance
    cat(paste("Minimal Deviance ", round(min(devvec),6), "at the optimal psi-value ", psi.opt, "\n", sep="\t"))
  }
}

###################
# profile.logit.r #
###################
# Plots psi against deviance and finds optimal psi-value to fit the glm

profile.logit <- function(y, X, n, tail, psivec=seq(0,2,length=100), eta0=0){
  devvec <- rep(0, length(psivec))
  devmat <- matrix(0, length(psivec), length(psivec))
  out <- list()
  m <- length(y)
  p <- ncol(X)
  fails <- n-y
  fails[fails == 0] <- 10^(-16)
  y[y == 0] <- 10^(-16)
  tail.int <- charmatch(tail, c("right", "left", "both"))

  neg.loglike.hpsi1.logit <- function(b, y, n, X, psi1, eta0){
    eta <- X %*% b
    expon.hpsi1vec <- exp(hpsi1(psi1, eta, eta0))
    Fhpsi1vec <- (expon.hpsi1vec)/(1 + expon.hpsi1vec)
    out <- -sum(y * log(Fhpsi1vec) + (n - y) * log(1 - Fhpsi1vec))
    out
  }
  neg.loglike.hpsi2.logit <- function(b, y, n, X, psi2, eta0){
    eta <- X %*% b
    expon.hpsi2vec <- exp(hpsi2(psi2, eta, eta0))
    Fhpsi2vec <- (expon.hpsi2vec)/(1 + expon.hpsi2vec)
    out <- -sum(y * log(Fhpsi2vec) + (n - y) * log(1 - Fhpsi2vec))
    out
  }
  neg.loglike.hpsi12.logit <- function(b, y, n, X, psi1, psi2, eta0){
    eta <- X %*% b
    expon.hpsi12vec <- exp(hpsi12(psi1, psi2, eta, eta0))
    Fhpsi12vec <- (expon.hpsi12vec)/(1 + expon.hpsi12vec)
    out <- -sum(y * log(Fhpsi12vec) + (n - y) * log(1 - Fhpsi12vec))
```

```
  out
}
if (tail.int == 1) { # Right tail Logit
  for (i in 1:length(psivec)){
    beta.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi1.logit, method="BFGS", y=y, n=n, X=X,
                      psi1=psivec[i], eta0=eta0)$par
    eta.opt <- X %*% beta.opt
    expon.hpsi1.opt <- exp(hpsi1(psivec[i], eta.opt, eta0))
    F.opt <- (expon.hpsi1.opt)/(1 + expon.hpsi1.opt)
    temp <- 2*sum(y * log(y/(n*F.opt)) - (y - n) * log((-fails)/(n*F.opt - n)))
    devvec[i] <- temp #deviance
    whichpsi <- expression(psi[1]) # Which psi for plot axes labels
  }
}
else if (tail.int == 2) { # Left tail Logit
  for (i in 1:length(psivec)){
    beta.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi2.logit, method="BFGS", y=y, n=n, X=X,
                      psi2=psivec[i], eta0=eta0)$par
    eta.opt <- X %*% beta.opt
    expon.hpsi2.opt <- exp(hpsi2(psivec[i], eta.opt, eta0))
    F.opt <- (expon.hpsi2.opt)/(1 + expon.hpsi2.opt)
    temp <- 2*sum(y * log(y/(n*F.opt)) - (y - n) * log((-fails)/(n*F.opt - n)))
    devvec[i] <- temp #deviance
    whichpsi <- expression(psi[2]) # Which psi for plot axes labels
  }
}
else if (tail.int == 3) { # Both tail Logit
  for (i in 1:length(psivec)){
    for (j in 1:length(psivec)){
      beta.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi12.logit, method="BFGS", y=y, n=n, X=X,
                psi1=psivec[i], psi2=psivec[j], eta0=eta0)$par
      eta.opt <- X %*% beta.opt
      expon.hpsi12.opt <- exp(hpsi12(psivec[i], psivec[j], eta.opt, eta0))
      F.opt <- (expon.hpsi12.opt)/(1 + expon.hpsi12.opt)
      temp <- 2*sum(y * log(y/(n*F.opt)) - (y - n) * log((-fails)/(n*F.opt - n)))
      devmat[i,j] <- temp #deviance
    }
  }
}
else {
  stop("Please select tail")}
if (tail.int==3){
  # find optimal psi1 and psi2 and corresponding mininmal deviance
  psi1.opt <- which(devmat == min(devmat), arr.ind = TRUE)[1]
  psi2.opt <- which(devmat == min(devmat), arr.ind = TRUE)[2]
  psi1.optval <- psivec[psi1.opt]
  psi2.optval <- psivec[psi2.opt]
  res.dev <- min(devmat)
  # 3D-Plot psi1 and psi2 against deviance
  persp(psivec, psivec, devmat, theta=45, phi=35, shade=0.75, ltheta=120, xlab = "psi1", ylab = "psi2",
        zlab = "deviance", ticktype="detailed", main = paste("3D Profile Plot psi-values against deviance"))
  # returns the specific psi with minimal deviance
  cat("Minimal Deviance ", round(min(devmat),6), "at the optimal psi-values psi1 = ", psi1.optval,
      " and psi2 = ", psi2.optval, "\n", sep="\t")
}
else {
  # find optimal psi-value and corresponding minimal deviance
  psi.opt <- psivec[devvec == min(devvec)]
  opt <- which(devvec == min(devvec))[1]
  res.dev <- min(devvec)
  # find 95%-CI for psi
  low.found <- FALSE
  up.found <- FALSE
  psi.ci.low <- psivec[1]
  psi.ci.up <- psivec[length(psivec)]
  for (m in 1:length(psivec)){
    if ((psivec[m] < psi.opt) && (low.found == FALSE)) {
      if (devvec[m] < (min(devvec)+2)){
        psi.ci.low <- psivec[m]
        low.found <- TRUE}
```

```
      }
      else if ((psivec[m] > psi.opt) && (up.found == FALSE)) {
        if (devvec[m] > (min(devvec)+2)){
          psi.ci.up <- psivec[m]
          up.found <- TRUE}
      }
    }
    # plots psi against deviance with 95%-CI for psi
    plot(psivec, devvec, xlab=whichpsi, ylab="deviance", main="Deviance Profile Plot for the Link Parameters",
         type="l", lty=1, sub=paste("Logit glm with ",tail, " tail modified with 95% confidence interval"))
    abline(h = min(devvec) + 2, lty=2)
    abline(v = psi.ci.low, lty=2)
    abline(v = psi.ci.up, lty=2)
    # returns the specific psi with minimal deviance
    cat("Minimal Deviance ", round(min(devvec),6), "at the optimal psi-value ", psi.opt, "\n", sep="\t")
  }
}

####################
# profile.probit.r #
####################
# Plots psi against deviance and finds optimal psi-value to fit the glm

profile.probit <- function(y, X, n, tail, psivec=seq(0,2,length=100), eta0=0){
  devvec <- rep(0, length(psivec))
  devmat <- matrix(0, length(psivec), length(psivec))
  out <- list()
  m <- length(y)
  p <- ncol(X)
  fails <- n-y
  fails[fails == 0] <- 10^(-16)
  y[y == 0] <- 10^(-16)
  tail.int <- charmatch(tail, c("right", "left", "both"))

  neg.loglike.hpsi1.probit <- function(b, y, n, X, psi1, eta0){
    eta <- X %*% b
    Fhpsi1vec <- pnorm(hpsi1(psi1, eta, eta0), mean=0, sd=1)
    out <- -sum(y * log(Fhpsi1vec) + (n - y) * log(1 - Fhpsi1vec))
    out
  }
  neg.loglike.hpsi2.probit <- function(b, y, n, X, psi2, eta0){
    eta <- X %*% b
    Fhpsi2vec <- pnorm(hpsi2(psi2, eta, eta0), mean=0, sd=1)
    out <- -sum(y * log(Fhpsi2vec) + (n - y) * log(1 - Fhpsi2vec))
    out
  }
  neg.loglike.hpsi12.probit <- function(b, y, n, X, psi1, psi2, eta0){
    eta <- X %*% b
    Fhpsi12vec <- pnorm(hpsi12(psi1, psi2, eta, eta0), mean=0, sd=1)
    out <- -sum(y * log(Fhpsi12vec) + (n - y) * log(1 - Fhpsi12vec))
    out
  }
  if (tail.int == 1) { # Right tail Logit
    for (i in 1:length(psivec)){
      beta.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi1.probit, method="BFGS", y=y, n=n, X=X,
                        psi1=psivec[i], eta0=eta0)$par
      eta.opt <- X %*% beta.opt
      F.opt <- pnorm(hpsi1(psivec[i], eta.opt, eta0), mean=0, sd=1)
      temp <- 2*sum(y * log(y/(n*F.opt)) - (y - n) * log((-fails)/(n*F.opt - n)))
      devvec[i] <- temp #deviance
      whichpsi <- expression(psi[1]) # Which psi for plot axes labels
    }
  }
  else if (tail.int == 2) { # Left tail Logit
    for (i in 1:length(psivec)){
      beta.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi2.probit, method="BFGS", y=y, n=n, X=X,
                        psi2=psivec[i], eta0=eta0)$par
      eta.opt <- X %*% beta.opt
      F.opt <- pnorm(hpsi2(psivec[i], eta.opt, eta0), mean=0, sd=1)
      temp <- 2*sum(y * log(y/(n*F.opt)) - (y - n) * log((-fails)/(n*F.opt - n)))
```

```
devvec[i] <- temp #deviance
      whichpsi <- expression(psi[2]) # Which psi for plot axes labels
    }
  }
  else if (tail.int == 3) { # Both tail Logit
    for (i in 1:length(psivec)){
      for (j in 1:length(psivec)){
        beta.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi12.probit, method="BFGS", y=y, n=n, X=X,
                          psi1=psivec[i], psi2=psivec[j], eta0=eta0)$par
        eta.opt <- X %*% beta.opt
        F.opt <- pnorm(hpsi12(psivec[i], psivec[j], eta.opt, eta0), mean=0, sd=1)
        temp <- 2*sum(y * log(y/(n*F.opt)) - (y - n) * log((-fails)/(n*F.opt - n)))
devmat[i,j] <- temp #deviance }
}
  }
  else {
    stop("Please select tail") }
  if (tail.int==3){
    # find optimal psi1 and psi2 and corresponding mininmal deviance
    psi1.opt <- which(devmat == min(devmat), arr.ind = TRUE)[1]
    psi2.opt <- which(devmat == min(devmat), arr.ind = TRUE)[2]
    psi1.optval <- psivec[psi1.opt]
    psi2.optval <- psivec[psi2.opt]
    res.dev <- min(devmat)
    # 3D-Plot psi1 and psi2 against deviance
    persp(psivec, psivec, devmat, theta=45, phi=35, shade=0.75, ltheta=120, xlab = "psi1", ylab = "psi2",
          zlab = "deviance", ticktype="detailed", main = paste("3D Profile Plot psi-values against deviance"))
    # returns the specific psi with minimal deviance
    cat("Minimal Deviance ", min(devmat), "at the optimal psi-values psi1 = ", psi1.optval, " and psi2 = ",
    psi2.optval, "\n", sep="\t")
  }
  else {
    # find optimal psi-value and corresponding minimal deviance
    psi.opt <- psivec[devvec == min(devvec)]
    opt <- which(devvec == min(devvec))[1]
    res.dev <- min(devvec)
    # find 95%-CI for psi
    low.found <- FALSE
    up.found <- FALSE
    psi.ci.low <- psivec[1]
    psi.ci.up <- psivec[length(psivec)]
    for (m in 1:length(psivec)){
      if ((psivec[m] < psi.opt) && (low.found == FALSE)) {
        if (devvec[m] < (min(devvec)+2)){
          psi.ci.low <- psivec[m]
          low.found <- TRUE}
      }
      else if ((psivec[m] > psi.opt) && (up.found == FALSE)) {
        if (devvec[m] > (min(devvec)+2)){
          psi.ci.up <- psivec[m]
          up.found <- TRUE}
      }
    }
    # plots psi against deviance with 95%-CI for psi
    plot(psivec, devvec, xlab=whichpsi, ylab="deviance", main="Deviance Profile Plot for the Link Parameters",
         type="l", lty=1, sub=paste("Probit glm with ",tail, " tail modified with 95% confidence interval"))
    abline(h = min(devvec) + 2, lty=2)
    abline(v = psi.ci.low, lty=2)
    abline(v = psi.ci.up, lty=2)
    # returns the specific psi with minimal deviance
    cat("Minimal Deviance ", round(min(devvec),6), "at the optimal psi-value ", psi.opt, "\n", sep="\t")
  }
}

######################
# profile.poisson.r #
######################
# Plots psi against deviance and finds optimal psi-value to fit the glm

profile.poisson <- function(y, X, tail, psivec=seq(0,2,length=100), eta0=0){
```

```
  devvec <- rep(0, length(psivec))
  out <- list()
  n <- length(y)
  p <- ncol(X)
  y[y == 0] = 10^(-16)
  tail.int <- charmatch(tail, c("right", "left", "both"))

  neg.loglike.hpsi1.poisson <- function(b, y, X, psi1, eta0){
    eta <- X %*% b
    Fhpsi1vec <- exp(hpsi1(psi1, eta, eta0))
    out <- -sum(y * hpsi1(psi1, eta, eta0) - Fhpsi1vec)
    out
  }
  if (tail.int == 1) { # Right tail Poisson
    for (i in 1:length(psivec)){
      beta.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi1.poisson, method="BFGS", y=y, X=X,
                        psi1=psivec[i], eta0=eta0)$par
      eta.opt <- X %*% beta.opt
      F.opt <- exp(hpsi1(psivec[i], eta.opt, eta0))
      temp <- 2*sum(y * log(y/F.opt) - y + F.opt)
      devvec[i] <- temp #deviance
    }
    # find optimal psi-value and corresponding minimal deviance
    psi.opt <- psivec[devvec == min(devvec)]
    opt <- which(devvec == min(devvec))[1]
    res.dev <- min(devvec)
    # find 95%-CI for psi
    low.found <- FALSE
    up.found <- FALSE
    psi.ci.low <- psivec[1]
    psi.ci.up <- psivec[length(psivec)]
    for (m in 1:length(psivec)){
      if ((psivec[m] < psi.opt) && (low.found == FALSE)) {
        if (devvec[m] < (min(devvec)+2)){
          psi.ci.low <- psivec[m]
          low.found <- TRUE}
      }
      else if ((psivec[m] > psi.opt) && (up.found == FALSE)) {
        if (devvec[m] > (min(devvec)+2)){
          psi.ci.up <- psivec[m]
          up.found <- TRUE}
      }
    }
    # plots psi against deviance with 95%-CI for psi
    plot(psivec, devvec, xlab=expression(psi[1]), ylab="deviance", main="Deviance Profile Plot for the Link Parameters",
         type="l", lty=1, sub=paste("Poisson glm with ",tail, " tail modified with 95% confidence interval"))
    abline(h = min(devvec) + 2, lty=2)
    abline(v = psi.ci.low, lty=2)
    abline(v = psi.ci.up, lty=2)
    # returns the specific psi with minimal deviance
    cat("Minimal Deviance ", round(min(devvec),6), "at the optimal psi-value ", psi.opt, "\n", sep="\t")
  }
  else {
    stop("Poisson only available with right tail modification")}
}
```

glm.fitted

```
#################
# glm.fitted.r #
#################
# Plots observed versus fitted mean plots of the standard glm and the glm with tail transformation

glm.fitted <- function(y, X, n, tail, family, psi1=1, psi2=1, eta0=0){
  family.int <- charmatch(family, c("gauss", "poiss", "logit", "probit"))
  if (family.int == 1){ # Gauss
    fitted.gauss(y=y, X=X, tail=tail, psi1=psi1, psi2=psi2, eta0=eta0)}
  else if (family.int == 2){ # Poisson
    fitted.poisson(y=y, X=X, tail=tail, psi1=psi1, psi2=psi2, eta0=eta0)}
```

```
  else if (family.int == 3){ # Logit
    fitted.logit(y=y, X=X, n=n, tail=tail, psi1=psi1, psi2=psi2, eta0=eta0)}
  else if (family.int == 4){# Probit
    fitted.probit(y=y, X=X, n=n, tail=tail, psi1=psi1, psi2=psi2, eta0=eta0)}
  else {
    stop("Please select gauss, poisson, logit or probit family")}
}

##################
# fitted.gauss.r #
##################
# Observed vs fitted mean plots of Ordinary Linear Regression and Gaussian GLM with tail transformation

fitted.gauss <- function(y, X, tail, psi1=1, psi2=1, eta0=0){
  out <- list()
  n <- length(y)
  p <- ncol(X)
  tail.int <- charmatch(tail, c("right", "left", "both"))

  neg.loglike.hpsi1.gauss <- function(pair.beta.sigma, y, X, psi1, eta0){
    beta <- pair.beta.sigma[1:p]
    logsigma2 <- pair.beta.sigma[p+1]
    eta <- X %*% beta
    hpsi1vec <- hpsi1(psi1, eta, eta0)
    temp <- y - hpsi1vec
    out <- (n * 0.5 * log(2 * pi * exp(logsigma2))) + (t(temp) %*% temp)/(2 * exp(logsigma2))
    out
  }
  neg.loglike.hpsi2.gauss <- function(pair.beta.sigma, y, X, psi2, eta0){
    beta <- pair.beta.sigma[1:p]
    logsigma2 <- pair.beta.sigma[p+1]
    eta <- X %*% beta
    hpsi2vec <- hpsi2(psi2, eta, eta0)
    temp <- y - hpsi2vec
    out <- (n * 0.5 * log(2 * pi * exp(logsigma2))) + (t(temp) %*% temp)/(2 * exp(logsigma2))
    out
  }
  neg.loglike.hpsi12.gauss <- function(pair.beta.sigma, y, X, psi1, psi2, eta0){
    beta <- pair.beta.sigma[1:p]
    logsigma2 <- pair.beta.sigma[p+1]
eta <- X %*% beta
    hpsi12vec <- hpsi12(psi1, psi2, eta, eta0)
    temp <- y - hpsi12vec
    out <- (n * 0.5 * log(2 * pi * exp(logsigma2))) + (t(temp) %*% temp)/(2 * exp(logsigma2))
    out
  }
  # compute beta.opt at psi1/psi2
  if (tail.int == 1){ # Right tail
    par.opt <- optim(par=rep(0,p+1), fn = neg.loglike.hpsi1.gauss, method="BFGS", y=y, X=X,
                     psi1=psi1, eta0=eta0)$par
    beta.opt <- par.opt[1:p]
    eta <- X %*% beta.opt
    eta.sort <- sort(eta)
    mu <- hpsi1(psi1, eta, eta0)
    mu.sort <- hpsi1(psi1, eta.sort, eta0)
    ord.par.opt <- optim(par=rep(0,p+1), fn = neg.loglike.hpsi1.gauss, method="BFGS", y=y, X=X,
                         psi1=1, eta0=eta0)$par
    ord.beta <- ord.par.opt[1:p] # Ordinary GLM}
  else if (tail.int == 2){  # Left tail
    par.opt <- optim(par=rep(0,p+1), fn = neg.loglike.hpsi2.gauss, method="BFGS", y=y, X=X,
                     psi2=psi2, eta0=eta0)$par
    beta.opt <- par.opt[1:p]
    eta <- X %*% beta.opt
    eta.sort <- sort(eta)
    mu <- hpsi2(psi2, eta, eta0)
    mu.sort <- hpsi2(psi2, eta.sort, eta0)
    ord.par.opt <- optim(par=rep(0,p+1), fn = neg.loglike.hpsi2.gauss, method="BFGS", y=y, X=X,
                         psi2=1, eta0=eta0)$par
    ord.beta <- ord.par.opt[1:p] # Ordinary GLM}
  else if (tail.int == 3){ # Both tail
```

```
    par.opt <- optim(par=rep(0,p+1), fn = neg.loglike.hpsi12.gauss, method="BFGS", y=y, X=X,
                     psi1=psi1, psi2=psi2, eta0=eta0)$par
    beta.opt <- par.opt[1:p]
    eta <- X %*% beta.opt
    eta.sort <- sort(eta)
    mu <- hpsi12(psi1, psi2, eta, eta0)
    mu.sort <- hpsi12(psi1, psi2, eta.sort, eta0)
    ord.par.opt <- optim(par=rep(0,p+1), fn = neg.loglike.hpsi12.gauss, method="BFGS", y=y, X=X,
                         psi1=1, psi2=1, eta0=eta0)$par
    ord.beta <- ord.par.opt[1:p] # Ordinary GLM}
  else {
    stop("Please select tail")}
  cat("Gauss GLM with ",tail," modified","\n")
  cat("Observed vs Fitted Mean Plot")
  y.sort <- y[sort.list(eta)]
  ord.eta <- X %*% ord.beta
  ord.eta.sort <- sort(ord.eta)
  ord.mu <- ord.eta
  ord.mu.sort <- ord.eta.sort
  ord.y.sort <- y[sort.list(ord.eta)]
  ry <- range(y, ord.mu, mu)
  rx <- range(eta, ord.eta)
  par(mfrow = c(2,1))
  plot(ord.eta.sort, ord.y.sort, ylim=ry, ylab="fitted mean", xlab="eta", pch=1, las=1,
       main="Ordinary Linear Regression")
  lines(ord.eta.sort, ord.mu.sort, type="b", lty=2, pch=16)
  segments(ord.eta.sort, ord.y.sort, ord.eta.sort, ord.mu.sort)
  plot(eta.sort, y.sort, ylim=ry, xlim=rx, ylab="fitted mean", xlab="eta", pch=1, las=1, main=paste("Gaussian GLM with ",
       tail," tail modified"), sub=paste("psi1=", psi1,", psi2=", psi2, ", eta0=", eta0))
  lines(eta.sort, mu.sort, type="b", lty=2, pch=16)
  segments(eta.sort, y.sort, eta.sort, mu.sort)
}

##################
# fitted.logit.r #
##################
# Observed vs fitted mean plots of Ordinary Logistic Regression and Logit GLM with tail transformation

fitted.logit <- function(y, X, n, tail, psi1=1, psi2=1, eta0=0){
  m <- length(y)
  p <- ncol(X)
  fails <- n-y
  fails[fails == 0] <- 10^(-16)
  y[y == 0] <- 10^(-16)
  tail.int <- charmatch(tail, c("right", "left", "both"))

  neg.loglike.hpsi1.logit <- function(b, y, n, X, psi1, eta0){
    eta <- X %*% b
    expon.hpsi1vec <- exp(hpsi1(psi1, eta, eta0))
    Fhpsi1vec <- (expon.hpsi1vec)/(1 + expon.hpsi1vec)
    out <- -sum(y * log(Fhpsi1vec) + (n - y) * log(1 - Fhpsi1vec))
    out
  }
  neg.loglike.hpsi2.logit <- function(b, y, n, X, psi2, eta0){
    eta <- X %*% b
    expon.hpsi2vec <- exp(hpsi2(psi2, eta, eta0))
    Fhpsi2vec <- (expon.hpsi2vec)/(1 + expon.hpsi2vec)
    out <- -sum(y * log(Fhpsi2vec) + (n - y) * log(1 - Fhpsi2vec))
    out
  }
  neg.loglike.hpsi12.logit <- function(b, y, n, X, psi1, psi2, eta0){
    eta <- X %*% b
    expon.hpsi12vec <- exp(hpsi12(psi1, psi2, eta, eta0))
    Fhpsi12vec <- (expon.hpsi12vec)/(1 + expon.hpsi12vec)
    out <- -sum(y * log(Fhpsi12vec) + (n - y) * log(1 - Fhpsi12vec))
    out
  }
  # compute beta.opt at psi1/psi2
  if (tail.int == 1){ # Right tail
    par.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi1.logit, method="BFGS", y=y, n=n, X=X,
```

```
                    psi1=psi1, eta0=eta0)$par
    beta.opt <- par.opt
    eta <- X %*% beta.opt
    eta.sort <- sort(eta)
    mu <- exp(hpsi1(psi1, eta, eta0))/(1 + exp(hpsi1(psi1, eta, eta0)))
    mu.sort <- exp(hpsi1(psi1, eta.sort, eta0))/(1 + exp(hpsi1(psi1, eta.sort, eta0)))
    ord.par.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi1.logit, method="BFGS", y=y, n=n, X=X,
                         psi1=1, eta0=eta0)$par
    ord.beta <- ord.par.opt # Ordinary GLM}
  else if (tail.int == 2){ # Left tail
    par.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi2.logit, method="BFGS", y=y, n=n, X=X,
                    psi2=psi2, eta0=eta0)$par
    beta.opt <- par.opt
    eta <- X %*% beta.opt
    eta.sort <- sort(eta)
    mu <- exp(hpsi2(psi2, eta, eta0))/(1 + exp(hpsi2(psi2, eta, eta0)))
    mu.sort <- exp(hpsi2(psi2, eta.sort, eta0))/(1 + exp(hpsi2(psi2, eta.sort, eta0)))
    ord.par.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi2.logit, method="BFGS", y=y, n=n, X=X,
                         psi2=1, eta0=eta0)$par
    ord.beta <- ord.par.opt # Ordinary GLM}
  else if (tail.int == 3){ # Both tail
    par.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi12.logit, method="BFGS", y=y, n=n, X=X,
                    psi1=psi1, psi2=psi2, eta0=eta0)$par
    beta.opt <- par.opt
    eta <- X %*% beta.opt
    eta.sort <- sort(eta)
    mu <- exp(hpsi12(psi1, psi2, eta, eta0))/(1 + exp(hpsi12(psi1, psi2, eta, eta0)))
    mu.sort <- exp(hpsi12(psi1, psi2, eta.sort, eta0))/(1 + exp(hpsi12(psi1, psi2, eta.sort, eta0)))
    ord.par.opt <- optim(par=rep(0,p+1), fn = neg.loglike.hpsi12.logit, method="BFGS", y=y, n=n, X=X,
                         psi1=1, psi2=1, eta0=eta0)$par
    ord.beta <- ord.par.opt # Ordinary GLM}
  else {
    stop("Please select tail")}
  cat("Logit GLM with ",tail," modified","\n")
  cat("Observed vs Fitted Mean Plot")
  y.sort <- y[sort.list(eta)]/(n[sort.list(eta)])
  ord.eta <- X %*% ord.beta
  ord.eta.sort <- sort(ord.eta)
  ord.mu <- exp(ord.eta)/(1 + exp(ord.eta))
  ord.mu.sort <- exp(ord.eta.sort)/(1 + exp(ord.eta.sort))
  ord.y.sort <- y[sort.list(ord.eta)]/(n[sort.list(ord.eta)])
  ry <- range((y/(y+n)), ord.mu, mu)
  rx <- range(eta, ord.eta)
  par(mfrow = c(2,1))
  plot(ord.eta.sort, ord.y.sort, ylim=ry, ylab="fitted mean", xlab="eta", pch=1, las=1,
       main="Ordinary Logistic Regression")
  lines(ord.eta.sort, ord.mu.sort, type="b", lty=2, pch=16)
  segments(ord.eta.sort, ord.y.sort, ord.eta.sort, ord.mu.sort)
  plot(eta.sort, y.sort, ylim=ry, xlim=rx, ylab="fitted mean", xlab="eta", pch=1, las=1, main=paste("Logit GLM with ",
       tail," tail modified"), sub=paste("psi1=", psi1,", psi2=", psi2, ", eta0=", eta0))
  lines(eta.sort, mu.sort, type="b", lty=2, pch=16)
  segments(eta.sort, y.sort, eta.sort, mu.sort)
}

####################
# fitted.probit.r #
####################
# Observed vs fitted mean plots of Ordinary Probit Regression and Probit GLM with tail transformation

fitted.probit <- function(y, X, n, tail, psi1=1, psi2=1, eta0=0){
  m <- length(y)
  p <- ncol(X)
  fails <- n-y
  fails[fails == 0] <- 10^(-16)
  y[y == 0] <- 10^(-16)
  tail.int <- charmatch(tail, c("right", "left", "both"))

  neg.loglike.hpsi1.probit <- function(b, y, n, X, psi1, eta0){
    eta <- X %*% b
    Fhpsi1vec <- pnorm(hpsi1(psi1, eta, eta0), mean=0, sd=1)
```

```
  out <- -sum(y * log(Fhpsi1vec) + (n - y) * log(1 - Fhpsi1vec))
  out
}
neg.loglike.hpsi2.probit <- function(b, y, n, X, psi2, eta0){
  eta <- X %*% b
  Fhpsi2vec <- pnorm(hpsi2(psi2, eta, eta0), mean=0, sd=1)
  out <- -sum(y * log(Fhpsi2vec) + (n - y) * log(1 - Fhpsi2vec))
  out
}
neg.loglike.hpsi12.probit <- function(b, y, n, X, psi1, psi2, eta0){
  eta <- X %*% b
  Fhpsi12vec <- pnorm(hpsi12(psi1, psi2, eta, eta0), mean=0, sd=1)
  out <- -sum(y * log(Fhpsi12vec) + (n - y) * log(1 - Fhpsi12vec))
  out
}
# compute beta.opt at psi1/psi2
if (tail.int == 1){ # Right tail
  par.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi1.probit, method="BFGS", y=y, n=n, X=X,
                   psi1=psi1, eta0=eta0)$par
  beta.opt <- par.opt
  eta <- X %*% beta.opt
  eta.sort <- sort(eta)
  mu <- pnorm(hpsi1(psi1, eta, eta0))
  mu.sort <- pnorm(hpsi1(psi1, eta.sort, eta0))
  ord.par.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi1.probit, method="BFGS", y=y, n=n, X=X,
                       psi1=1, eta0=eta0)$par
  ord.beta <- ord.par.opt # Ordinary GLM}
else if (tail.int == 2){ # Left tail
  par.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi2.probit, method="BFGS", y=y, n=n, X=X,
                   psi2=psi2, eta0=eta0)$par
  beta.opt <- par.opt
  eta <- X %*% beta.opt
  eta.sort <- sort(eta)
  mu <- pnorm(hpsi2(psi2, eta, eta0))
  mu.sort <- pnorm(hpsi2(psi2, eta.sort, eta0))
  ord.par.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi2.probit, method="BFGS", y=y, n=n, X=X,
                       psi2=1, eta0=eta0)$par
  ord.beta <- ord.par.opt # Ordinary GLM}
else if (tail.int == 3){ # Both tail
  par.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi12.probit, method="BFGS", y=y, n=n, X=X,
                   psi1=psi1, psi2=psi2, eta0=eta0)$par
  beta.opt <- par.opt
  eta <- X %*% beta.opt
  eta.sort <- sort(eta)
  mu <- pnorm(hpsi12(psi1, psi2, eta, eta0))
  mu.sort <- pnorm(hpsi12(psi1, psi2, eta.sort, eta0))
  ord.par.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi12.probit, method="BFGS", y=y, n=n, X=X, psi1=1,
                       psi2=1, eta0=eta0)$par
  ord.beta <- ord.par.opt # Ordinary GLM}
else {
  stop("Please select tail")}
cat("Probit GLM with ",tail," modified","\n")
cat("Observed vs Fitted Mean Plot")
y.sort <- y[sort.list(eta)]/(n[sort.list(eta)])
ord.eta <- X %*% ord.beta
ord.eta.sort <- sort(ord.eta)
ord.mu <- pnorm(ord.eta)
ord.mu.sort <- pnorm(ord.eta.sort)
ord.y.sort <- y[sort.list(ord.eta)]/(n[sort.list(ord.eta)])
ry <- range((y/(y+n)), ord.mu, mu)
rx <- range(eta, ord.eta)
par(mfrow = c(2,1))
plot(ord.eta.sort, ord.y.sort, ylim=ry, ylab="fitted mean", xlab="eta", pch=1, las=1,
     main="Ordinary Probit Regression")
lines(ord.eta.sort, ord.mu.sort, type="b", lty=2, pch=16)
segments(ord.eta.sort, ord.y.sort, ord.eta.sort, ord.mu.sort)
plot(eta.sort, y.sort, ylim=ry, xlim=rx, ylab="fitted mean", xlab="eta", pch=1, las=1, main=paste("Probit GLM with ",
     tail," tail modified"), sub=paste("psi1=", psi1,", psi2=", psi2, ", eta0=", eta0))
lines(eta.sort, mu.sort, type="b", lty=2, pch=16)
segments(eta.sort, y.sort, eta.sort, mu.sort)
```

```
}

###################
# fitted.probit.r #
###################
# Observed vs fitted mean plots of Ordinary Probit Regression and Probit GLM with tail transformation

fitted.probit <- function(y, X, n, tail, psi1=1, psi2=1, eta0=0){
  m <- length(y)
  p <- ncol(X)
  fails <- n-y
  fails[fails == 0] <- 10^(-16)
  y[y == 0] <- 10^(-16)
  tail.int <- charmatch(tail, c("right", "left", "both"))

  neg.loglike.hpsi1.probit <- function(b, y, n, X, psi1, eta0){
    eta <- X %*% b
    Fhpsi1vec <- pnorm(hpsi1(psi1, eta, eta0), mean=0, sd=1)
    out <- -sum(y * log(Fhpsi1vec) + (n - y) * log(1 - Fhpsi1vec))
    out
  }
  neg.loglike.hpsi2.probit <- function(b, y, n, X, psi2, eta0){
    eta <- X %*% b
    Fhpsi2vec <- pnorm(hpsi2(psi2, eta, eta0), mean=0, sd=1)
    out <- -sum(y * log(Fhpsi2vec) + (n - y) * log(1 - Fhpsi2vec))
    out
  }
  neg.loglike.hpsi12.probit <- function(b, y, n, X, psi1, psi2, eta0){
    eta <- X %*% b
    Fhpsi12vec <- pnorm(hpsi12(psi1, psi2, eta, eta0), mean=0, sd=1)
    out <- -sum(y * log(Fhpsi12vec) + (n - y) * log(1 - Fhpsi12vec))
    out
  }
  # compute beta.opt at psi1/psi2
  if (tail.int == 1){ # Right tail
    par.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi1.probit, method="BFGS", y=y, n=n, X=X,
                     psi1=psi1, eta0=eta0)$par
    beta.opt <- par.opt
    eta <- X %*% beta.opt
    eta.sort <- sort(eta)
    mu <- pnorm(hpsi1(psi1, eta, eta0))
    mu.sort <- pnorm(hpsi1(psi1, eta.sort, eta0))
    ord.par.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi1.probit, method="BFGS", y=y, n=n, X=X,
                         psi1=1, eta0=eta0)$par
    ord.beta <- ord.par.opt # Ordinary GLM}
  else if (tail.int == 2){ # Left tail
    par.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi2.probit, method="BFGS", y=y, n=n, X=X,
               psi2=psi2, eta0=eta0)$par
    beta.opt <- par.opt
    eta <- X %*% beta.opt
    eta.sort <- sort(eta)
    mu <- pnorm(hpsi2(psi2, eta, eta0))
    mu.sort <- pnorm(hpsi2(psi2, eta.sort, eta0))
    ord.par.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi2.probit, method="BFGS", y=y, n=n, X=X,
                         psi2=1, eta0=eta0)$par
    ord.beta <- ord.par.opt # Ordinary GLM
  }
  else if (tail.int == 3){ # Both tail
    par.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi12.probit, method="BFGS", y=y, n=n, X=X, psi1=psi1,
                     psi2=psi2, eta0=eta0)$par
    beta.opt <- par.opt
    eta <- X %*% beta.opt
    eta.sort <- sort(eta)
    mu <- pnorm(hpsi12(psi1, psi2, eta, eta0))
    mu.sort <- pnorm(hpsi12(psi1, psi2, eta.sort, eta0))
    ord.par.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi12.probit, method="BFGS", y=y, n=n, X=X,
                         psi1=1, psi2=1, eta0=eta0)$par
    ord.beta <- ord.par.opt # Ordinary GLM
  }
  else {
```

```
    stop("Please select tail")}
  cat("Probit GLM with ",tail," modified","\n")
  cat("Observed vs Fitted Mean Plot")
  y.sort <- y[sort.list(eta)]/(n[sort.list(eta)])
  ord.eta <- X %*% ord.beta
  ord.eta.sort <- sort(ord.eta)
  ord.mu <- pnorm(ord.eta)
  ord.mu.sort <- pnorm(ord.eta.sort)
  ord.y.sort <- y[sort.list(ord.eta)]/(n[sort.list(ord.eta)])
  ry <- range((y/(y+n)), ord.mu, mu)
  rx <- range(eta, ord.eta)
  par(mfrow = c(2,1))
  plot(ord.eta.sort, ord.y.sort, ylim=ry, ylab="fitted mean", xlab="eta", pch=1, las=1,
       main="Ordinary Probit Regression")
  lines(ord.eta.sort, ord.mu.sort, type="b", lty=2, pch=16)
  segments(ord.eta.sort, ord.y.sort, ord.eta.sort, ord.mu.sort)
  plot(eta.sort, y.sort, ylim=ry, xlim=rx, ylab="fitted mean", xlab="eta", pch=1, las=1, main=paste("Probit GLM with ",
       tail," tail modified"), sub=paste("psi1=", psi1,", psi2=", psi2, ", eta0=", eta0))
  lines(eta.sort, mu.sort, type="b", lty=2, pch=16)
  segments(eta.sort, y.sort, eta.sort, mu.sort)
}

####################
# fitted.poisson.r #
####################
# Observed vs fitted mean plots of Ordinary Poisson Regression and Poisson GLM with tail transformation

fitted.poisson <- function(y, X, tail, psi1, psi2, eta0=0){
  p <- ncol(X)
  y[y == 0] <- 10^(-16)
  tail.int <- charmatch(tail, c("right", "left", "both"))

  neg.loglike.hpsi1.poisson <- function(b, y, X, psi1, eta0){
    eta <- X %*% b
    Fhpsi1vec <- exp(hpsi1(psi1, eta, eta0))
    out <- -sum(y * hpsi1(psi1, eta, eta0) - Fhpsi1vec)
    out
  }
  if (tail.int == 1) { # Right tail Poisson
    # compute beta.opt at psi1
    par.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi1.poisson, method="BFGS", y=y, X=X, psi1=psi1, eta0=eta0)$par
    beta.opt <- par.opt
    eta <- X %*% beta.opt
    eta.sort <- sort(eta)
    mu <- exp(hpsi1(psi1, eta, eta0))
    mu.sort <- exp(hpsi1(psi1, eta.sort, eta0))
    ord.par.opt <- optim(par=rep(0,p), fn = neg.loglike.hpsi1.poisson, method="BFGS", y=y, X=X, psi1=1, eta0=eta0)$par
    ord.beta <- ord.par.opt # Ordinary GLM
    cat("Poisson GLM with right modified","\n")
    cat("Observed vs Fitted Mean Plot")
    y.sort <- y[sort.list(eta)]
    ord.eta <- X %*% ord.beta
    ord.eta.sort <- sort(ord.eta)
    ord.mu <- exp(ord.eta)
    ord.mu.sort <- exp(ord.eta.sort)
    ord.y.sort <- y[sort.list(ord.eta)]
    ry <- range(y, ord.mu, mu)
    rx <- range(eta, ord.eta)
    par(mfrow = c(2,1))
    plot(ord.eta.sort, ord.y.sort, ylim=ry, ylab="fitted mean", xlab="eta", pch=1, las=1,
         main="Ordinary Poisson Regression")
    lines(ord.eta.sort, ord.mu.sort, type="b", lty=2, pch=16)
    segments(ord.eta.sort, ord.y.sort, ord.eta.sort, ord.mu.sort)
    plot(eta.sort, y.sort, ylim=ry, xlim=rx, ylab="fitted mean", xlab="eta", pch=1, las=1,
         main=paste("Poisson GLM with ", tail," tail modified"), sub=paste("psi1=", psi1,", eta0=", eta0))
  lines(eta.sort, mu.sort, type="b", lty=2, pch=16)
  segments(eta.sort, y.sort, eta.sort, mu.sort)
  }
  else {
    stop("Poisson only available with right tail modification")}
```

```
}
```

glm.inf

```
##############
# glm.inf.r #
##############
glm.inf <- function(y, X, n, tail, family, psi1=1, psi2=1, eta0=0){
  family.int <- charmatch(family, c("gauss", "poiss", "logit", "probit"))
  if (family.int == 1){ # Gauss
    inf.gauss(y=y, X=X, tail=tail, psi1=psi1, psi2=psi2, eta0=eta0)}
  else if (family.int == 2){ # Poisson
    inf.poisson(y=y, X=X, tail=tail, psi1=psi1, psi2=psi2, eta0=eta0)}
  else if (family.int == 3){ # Logit
    inf.logit(y=y, X=X, n=n, tail, psi1=psi1, psi2=psi2, eta0=eta0)}
  else if (family.int == 4){# Probit
    inf.probit(y=y, X=X, n=n, tail, psi1=psi1, psi2=psi2, eta0=eta0)}
  else {
    stop("Please select 'gauss', 'poiss', 'logit' or 'probit' family")}
}

###############
# inf.gauss.r #
###############
# Gives standard errors

inf.gauss <- function(y, X, tail, psi1, psi2, eta0=0){
  out <- list()
  n <- length(y)
  p <- ncol(X)
  tail.int <- charmatch(tail, c("right", "left", "both"))

  neg.loglike.hpsi1.gauss.est <- function(triple.beta.psi.sigma, y, X, eta0){
    beta <- triple.beta.psi.sigma[1:p]
    psi <- triple.beta.psi.sigma[p+1]
    logsigma2 <- triple.beta.psi.sigma[p+2]
    eta <- X %*% beta
    hpsi1vec <- hpsi1(psi, eta, eta0)
    temp <- y - hpsi1vec
    out <- (n * 0.5 * log(2 * pi * exp(logsigma2))) + (t(temp) %*% temp)/(2 * exp(logsigma2))
    out
  }
  neg.loglike.hpsi2.gauss.est <- function(triple.beta.psi.sigma, y, X, eta0){
    beta <- triple.beta.psi.sigma[1:p]
    psi <- triple.beta.psi.sigma[p+1]
    logsigma2 <- triple.beta.psi.sigma[p+2]
    eta <- X %*% beta
    hpsi2vec <- hpsi2(psi, eta, eta0)
    temp <- y - hpsi2vec
    out <- (n * 0.5 * log(2 * pi * exp(logsigma2))) + (t(temp) %*% temp)/(2 * exp(logsigma2))
    out
  }
  neg.loglike.hpsi12.gauss.est <- function(triple.beta.psi.sigma, y, X, eta0){
    beta <- triple.beta.psi.sigma[1:p]
    psi <- triple.beta.psi.sigma[p+1:p+2]
    logsigma2 <- triple.beta.psi.sigma[p+3]
    eta <- X %*% beta
    hpsi12vec <- hpsi12(psi[1], psi[2], eta, eta0)
    temp <- y - hpsi12vec
    out <- (n * 0.5 * log(2 * pi * exp(logsigma2))) + (t(temp) %*% temp)/(2 * exp(logsigma2))
    out
  }
  neg.loglike.hpsi1.gauss <- function(pair.beta.sigma, y, X, psi1, eta0){
    beta <- pair.beta.sigma[1:p]
    logsigma2 <- pair.beta.sigma[p+1]
    eta <- X %*% beta
    hpsi1vec <- hpsi1(psi1, eta, eta0)
    temp <- y - hpsi1vec
    out <- (n * 0.5 * log(2 * pi * exp(logsigma2))) + (t(temp) %*% temp)/(2 * exp(logsigma2))
```

```
  out
}
neg.loglike.hpsi2.gauss <- function(pair.beta.sigma, y, X, psi2, eta0){
  beta <- pair.beta.sigma[1:p]
  logsigma2 <- pair.beta.sigma[p+1]
  eta <- X %*% beta
  hpsi2vec <- hpsi2(psi2, eta, eta0)
  temp <- y - hpsi2vec
  out <- (n * 0.5 * log(2 * pi * exp(logsigma2))) + (t(temp) %*% temp)/(2 * exp(logsigma2))
  out
}
neg.loglike.hpsi12.gauss <- function(pair.beta.sigma, y, X, psi1, psi2, eta0){
  beta <- pair.beta.sigma[1:p]
  logsigma2 <- pair.beta.sigma[p+1]
  eta <- X %*% beta
  hpsi12vec <- hpsi12(psi1, psi2, eta, eta0)
  temp <- y - hpsi12vec
  out <- (n * 0.5 * log(2 * pi * exp(logsigma2))) + (t(temp) %*% temp)/(2 * exp(logsigma2))
  out
}
# compute beta.opt and hessian matrix
if (tail.int == 1){ # Right tail
  # estimated
  opt.est <- optim(par=rep(0,p+2), fn = neg.loglike.hpsi1.gauss.est, method="BFGS", y=y, X=X,
                   eta0=eta0, hessian=TRUE)
  par.opt.est <- opt.est$par
  hessian.est <- opt.est$hessian
  psi.est1 <- par.opt.est[p+1]
  # compute Hessian matrix for fixed psi1
  opt.fix <- optim(par=rep(0,p+1), fn = neg.loglike.hpsi1.gauss, method="BFGS", y=y, X=X, psi1=psi1,
                   eta0=eta0, hessian=TRUE)
  par.opt.fix <- opt.fix$par
  beta.opt.fix <- par.opt.fix[1:p]
  hessian.fix <- opt.fix$hessian
  cat("Gaussian GLM with ",tail," tail modified, psi1 = ", psi1, "\n\n")}
else if (tail.int == 2){ # Left tail
  # estimated
  opt.est <- optim(par=rep(0,p+2), fn = neg.loglike.hpsi2.gauss.est, method="BFGS", y=y, X=X,
                   eta0=eta0, hessian=TRUE)
  par.opt.est <- opt.est$par
  hessian.est <- opt.est$hessian
  psi.est2 <- par.opt.est[p+1]
  # compute Hessian matrix for fixed psi2
  opt.fix <- optim(par=rep(0,p+1), fn = neg.loglike.hpsi2.gauss, method="BFGS", y=y, X=X, psi2=psi2,
                   eta0=eta0, hessian=TRUE)
  par.opt.fix <- opt.fix$par
  beta.opt.fix <- par.opt.fix[1:p]
  hessian.fix <- opt.fix$hessian
  cat("Gaussian GLM with ",tail," tail modified, psi2 = ", psi2, "\n\n")}
else if (tail.int == 3){ # Both tail
  # estimated
  opt.est <- optim(par=rep(0,p+3), fn = neg.loglike.hpsi12.gauss.est, method="BFGS", y=y, X=X,
                   eta0=eta0, hessian=TRUE)
  par.opt.est <- opt.est$par
  hessian.est <- opt.est$hessian
  psi.est1 <- par.opt.est[p+1]
  psi.est2 <- par.opt.est[p+2]
  # compute Hessian matrix for fixed psi1 and psi2
  opt.fix <- optim(par=rep(0,p+1), fn = neg.loglike.hpsi12.gauss, method="BFGS", y=y, X=X, psi1=psi1,
                   psi2=psi2, eta0=eta0, hessian=TRUE)
  par.opt.fix <- opt.fix$par
  beta.opt.fix <- par.opt.fix[1:p]
  hessian.fix <- opt.fix$hessian
  cat("Gaussian GLM with ",tail," tail modified, psi1 = ", psi1, " and psi2 = ", psi2, "\n\n")}
else {
  stop("Please select tail")}
# compute fixed and estimated standard error
inf.est <- solve(hessian.est)
se.est <- sqrt(diag(abs(inf.est)))  # se.est[1:p] are the estimated se
inf.fix <- solve(hessian.fix)
```

```
se.fix <- sqrt(diag(abs(inf.fix)))  # se.fix[1:p] are the fixed se
# compute all values needed
coeff <- round(beta.opt.fix,4)
fixed.se <- round(se.fix[1:p],5)
fixed.z <- round(coeff/fixed.se, 3)
est.se <- round(se.est[1:p],4)
est.z <- round(coeff/est.se, 3)
var.infl <- round((est.se*100)/fixed.se - 100, 1)
# values from above in a matrix
mat <- as.matrix(cbind(coeff, fixed.se, fixed.z, est.se, est.z, var.infl))
# Regression parameter
cat("Regression Parameters:","\n")
covariates <- rep("0",p)
covariates[1] <- "(Intercept)"
if (is.null(dimnames(X)[[2]])){
  for (i in 1:(p-1)){
    covariates[i+1] <- paste("Covariate",i)}
}
else {
  temp <- dimnames(X)[[2]]
  covariates[2:p] <- temp[2:p]}
prmatrix(mat, quote=F, rowlab=covariates, collab=c(" coeff"," fixed se"," fixed z"," estimated se",
                                                   " estimated z"," var inflation"))
cat("\n")
# Link parameter
if (tail.int == 1){
  cat("Link Parameter:","\n")
  psi1.est.se <- round(se.est[p+1],4)
  cat("psi1 = ", psi1, ", standard error of psi1 = ", psi1.est.se,"\n\n") }
else if (tail.int == 2){
  cat("Link Parameter:","\n")
  psi2.est.se <- round(se.est[p+1],4)
  cat("psi2 = ", psi2, ", standard error of psi2 = ", psi2.est.se,"\n\n") }
else {
  cat("Link Parameters:","\n")
  psi1.est.se <- round(se.est[p+1],4)
  psi2.est.se <- round(se.est[p+2],4)
  cat("psi1 = ", psi1, ", standard error of psi1 = ", psi1.est.se,"\n")
  cat("psi2 = ", psi2, ", standard error of psi2 = ", psi2.est.se,"\n\n") }
# Correlation matrix
cat("Correlation matrix with estimated link:","\n")
if (tail.int == 1){
  corr.mat <- matrix(rep(0,(p+1)*(p+1)), nrow=p+1, ncol=p+1)
  for (i in 1:(p+1)){
    for (j in 1:(p+1)){
      if (i == j) {corr.mat[i,i] = 1.0000}
      else {corr.mat[i,j] = round(inf.est[i,j]/(sqrt(abs(inf.est[i,i]*inf.est[j,j]))),4)}
    }
  }
  rowlab <- c(covariates, "psi1")
  prmatrix(corr.mat, quote=F, rowlab=rowlab, collab=rowlab)
}
else if (tail.int == 2){
  corr.mat <- matrix(rep(0,(p+1)*(p+1)), nrow=p+1, ncol=p+1)
  for (i in 1:p+1){
    for (j in 1:p+1){
      if (i == j) {corr.mat[i,i] = 1.0000}
      else {corr.mat[i,j] = round(inf.est[i,j]/(sqrt(abs(inf.est[i,i]*inf.est[j,j]))),4)}
    }
  }
  rowlab <- c(covariates, "psi2")
  prmatrix(corr.mat, quote=F, rowlab=rowlab, collab=rowlab)
}
else {
  corr.mat <- matrix(rep(0,(p+2)*(p+2)), nrow=p+2, ncol=p+2)
  for (i in 1:p+2){
    for (j in 1:p+2){
      if (i == j) {corr.mat[i,i] = 1.0000}
      else {corr.mat[i,j] = round(inf.est[i,j]/(sqrt(abs(inf.est[i,i]*inf.est[j,j]))),4)}
    }
```

```
    }
    rowlab <- c(covariates, "psi1", "psi2")
    prmatrix(corr.mat, quote=F, rowlab=rowlab, collab=rowlab)
  }
}

###############
# inf.logit.r #
###############
# Gives standard errors

inf.logit <- function(y, X, n, tail, psi1, psi2, eta0=0){
  out <- list()
  p <- ncol(X)
  fails <- n-y
  fails[fails == 0] <- 10^(-16)
  y[y == 0] <- 10^(-16)
  tail.int <- charmatch(tail, c("right", "left", "both"))

  neg.loglike.hpsi1.logit.est <- function(pair.beta.psi, y, n, X, eta0){
    beta <- pair.beta.psi[1:p]
    psi <- pair.beta.psi[p+1]
    eta <- X %*% beta
    expon.hpsi1vec <- exp(hpsi1(psi, eta, eta0))
    Fhpsi1vec <- (expon.hpsi1vec)/(1 + expon.hpsi1vec)
    out <- -sum(y * log(Fhpsi1vec) + (n - y) * log(1 - Fhpsi1vec))
    out
  }
  neg.loglike.hpsi2.logit.est <- function(pair.beta.psi, y, n, X, eta0){
    beta <- pair.beta.psi[1:p]
    psi <- pair.beta.psi[p+1]
    eta <- X %*% beta
    expon.hpsi2vec <- exp(hpsi2(psi, eta, eta0))
    Fhpsi2vec <- (expon.hpsi2vec)/(1 + expon.hpsi2vec)
    out <- -sum(y * log(Fhpsi2vec) + (n - y) * log(1 - Fhpsi2vec))
    out
  }
  neg.loglike.hpsi12.logit.est <- function(pair.beta.psi, y, n, X, eta0){
    beta <- pair.beta.psi[1:p]
    psi <- pair.beta.psi[p+1:p+2]
    eta <- X %*% beta
    expon.hpsi12vec <- exp(hpsi12(psi[1], psi[2], eta, eta0))
    Fhpsi12vec <- (expon.hpsi12vec)/(1 + expon.hpsi12vec)
    out <- -sum(y * log(Fhpsi12vec) + (n - y) * log(1 - Fhpsi12vec))
    out
  }
  neg.loglike.hpsi1.logit <- function(b, y, n, X, psi1, eta0){
    eta <- X %*% b
    expon.hpsi1vec <- exp(hpsi1(psi1, eta, eta0))
    Fhpsi1vec <- (expon.hpsi1vec)/(1 + expon.hpsi1vec)
    out <- -sum(y * log(Fhpsi1vec) + (n - y) * log(1 - Fhpsi1vec))
    out
  }
  neg.loglike.hpsi2.logit <- function(b, y, n, X, psi2, eta0){
    eta <- X %*% b
    expon.hpsi2vec <- exp(hpsi2(psi2, eta, eta0))
    Fhpsi2vec <- (expon.hpsi2vec)/(1 + expon.hpsi2vec)
    out <- -sum(y * log(Fhpsi2vec) + (n - y) * log(1 - Fhpsi2vec))
    out
  }
  neg.loglike.hpsi12.logit <- function(b, y, n, X, psi1, psi2, eta0){
    eta <- X %*% b
    expon.hpsi12vec <- exp(hpsi12(psi1, psi2, eta, eta0))
    Fhpsi12vec <- (expon.hpsi12vec)/(1 + expon.hpsi12vec)
    out <- -sum(y * log(Fhpsi12vec) + (n - y) * log(1 - Fhpsi12vec))
    out
  }
  # compute beta.opt and hessian matrix
  if (tail.int == 1){ # Right tail
    # estimated
```

```
  opt.est <- optim(par=rep(0,p+1), fn = neg.loglike.hpsi1.logit.est, method="BFGS", y=y, n=n, X=X,
                   eta0=eta0, hessian=TRUE)
  par.opt.est <- opt.est$par
  hessian.est <- opt.est$hessian
  psi.est1 <- par.opt.est[p+1]
  # compute Hessian matrix for fixed psi1
  opt.fix <- optim(par=rep(0,p), fn = neg.loglike.hpsi1.logit, method="BFGS", y=y, n=n, X=X, psi1=psi1,
                   eta0=eta0, hessian=TRUE)
  beta.opt.fix <- opt.fix$par
  hessian.fix <- opt.fix$hessian
  cat("Logit GLM with ",tail," tail modified, psi1 = ", psi1, "\n\n")
}
else if (tail.int == 2){ # Left tail
  # estimated
  opt.est <- optim(par=rep(0,p+1), fn = neg.loglike.hpsi2.logit.est, method="BFGS", y=y, n=n, X=X,
                   eta0=eta0, hessian=TRUE)
  par.opt.est <- opt.est$par
  hessian.est <- opt.est$hessian
  psi.est2 <- par.opt.est[p+1]
  # compute Hessian matrix for fixed psi2
  opt.fix <- optim(par=rep(0,p), fn = neg.loglike.hpsi2.logit, method="BFGS", y=y, n=n, X=X, psi2=psi2,
                   eta0=eta0, hessian=TRUE)
  beta.opt.fix <- opt.fix$par
  hessian.fix <- opt.fix$hessian
  cat("Logit GLM with ",tail," tail modified, psi2 = ", psi2, "\n\n")
}
else if (tail.int == 3){ # Both tail
  # estimated
  opt.est <- optim(par=rep(0,p+2), fn = neg.loglike.hpsi12.logit.est, method="BFGS", y=y, n=n,
                X=X, eta0=eta0, hessian=TRUE)
  par.opt.est <- opt.est$par
  hessian.est <- opt.est$hessian
  psi.est1 <- par.opt.est[p+1]
  psi.est2 <- par.opt.est[p+2]
  # compute Hessian matrix for fixed psi1 and psi2
  opt.fix <- optim(par=rep(0,p), fn = neg.loglike.hpsi12.logit, method="BFGS", y=y, X=X, n=n,
                psi1=psi1, psi2=psi2, eta0=eta0, hessian=TRUE)
  beta.opt.fix <- opt.fix$par
  hessian.fix <- opt.fix$hessian
  cat("Logit GLM with ",tail," tail modified, psi1 = ", psi1, " and psi2 = ", psi2, "\n\n")
}
else {
  stop("Please select tail")}
# compute fixed and estimated standard error
inf.est <- solve(hessian.est)
se.est <- sqrt(diag(abs(inf.est)))  # se.est[1:p] are the estimated se
inf.fix <- solve(hessian.fix)
se.fix <- sqrt(diag(abs(inf.fix)))  # se.fix[1:p] are the fixed se
# compute all values needed
coeff <- round(beta.opt.fix,4)
fixed.se <- round(se.fix[1:p],5)
fixed.z <- round(coeff/fixed.se, 3)
est.se <- round(se.est[1:p],4)
est.z <- round(coeff/est.se, 3)
var.infl <- round((est.se*100)/fixed.se - 100, 1)
# values from above in a matrix
mat <- as.matrix(cbind(coeff, fixed.se, fixed.z, est.se, est.z, var.infl))
# Regression parameters
cat("Regression Parameters:","\n")
covariates <- rep("0",p)
covariates[1] <- "(Intercept)"
if (is.null(dimnames(X)[[2]])){
  for (i in 1:(p-1)){
    covariates[i+1] <- paste("Covariate",i)}
}
else {
  temp <- dimnames(X)[[2]]
  covariates[2:p] <- temp[2:p]}
prmatrix(mat, quote=F, rowlab=covariates, collab=c(" coeff"," fixed se"," fixed z"," estimated se",
                                                   " estimated z"," var inflation"))
```

```
  cat("\n")
  # Link parameters
  if (tail.int == 1){
    cat("Link Parameters:","\n")
    psi1.est.se <- round(se.est[p+1],4)
    cat("psi1 = ", psi1, ", standard error of psi1 = ", psi1.est.se,"\n\n") }
  else if (tail.int == 2){
    cat("Link Parameter:","\n")
    psi2.est.se <- round(se.est[p+1],4)
    cat("psi2 = ", psi2, ", standard error of psi2 = ", psi2.est.se,"\n\n") }
  else {
    cat("Link Parameters:","\n")
    psi1.est.se <- round(se.est[p+1],4)
    psi2.est.se <- round(se.est[p+2],4)
    cat("psi1 = ", psi1, ", standard error of psi1 = ", psi1.est.se,"\n")
    cat("psi2 = ", psi2, ", standard error of psi2 = ", psi2.est.se,"\n\n")}
  # Correlation matrix
  cat("Correlation matrix with estimated link:","\n")
  if (tail.int == 1){
    corr.mat <- matrix(rep(0,(p+1)*(p+1)), nrow=p+1, ncol=p+1)
    for (i in 1:(p+1)){
      for (j in 1:(p+1)){
        if (i == j) {corr.mat[i,i] = 1.0000}
        else {corr.mat[i,j] = round(inf.est[i,j]/(sqrt(abs(inf.est[i,i]*inf.est[j,j]))),4)}
      }
    }
    rowlab <- c(covariates, "psi1")
    prmatrix(corr.mat, quote=F, rowlab=rowlab, collab=rowlab)}
  else if (tail.int == 2){
    corr.mat <- matrix(rep(0,(p+1)*(p+1)), nrow=p+1, ncol=p+1)
    for (i in 1:(p+1)){
      for (j in 1:(p+1)){
        if (i == j) {corr.mat[i,i] = 1.0000}
        else {corr.mat[i,j] = round(inf.est[i,j]/(sqrt(abs(inf.est[i,i]*inf.est[j,j]))),4)}
      }
    }
    rowlab <- c(covariates, "psi2")
    prmatrix(corr.mat, quote=F, rowlab=rowlab, collab=rowlab)}
  else {
    corr.mat <- matrix(rep(0,(p+2)*(p+2)), nrow=p+2, ncol=p+2)
    for (i in 1:(p+2)){
      for (j in 1:(p+2)){
        if (i == j) {corr.mat[i,i] = 1.0000}
        else {corr.mat[i,j] = round(inf.est[i,j]/(sqrt(abs(inf.est[i,i]*inf.est[j,j]))),4)}
      }
    }
    rowlab <- c(covariates, "psi1", "psi2")
    prmatrix(corr.mat, quote=F, rowlab=rowlab, collab=rowlab)}
}

################
# inf.probit.r #
################
# Gives standard errors

inf.probit <- function(y, n, X, tail, psi1, psi2, eta0=0){
  out <- list()
  p <- ncol(X)
  fails <- n-y
  fails[fails == 0] <- 10^(-16)
  y[y == 0] <- 10^(-16)
  tail.int <- charmatch(tail, c("right", "left", "both"))

  neg.loglike.hpsi1.probit.est <- function(pair.beta.psi, y, n, X, eta0){
    beta <- pair.beta.psi[1:p]
    psi <- pair.beta.psi[p+1]
    eta <- X %*% beta
    Fhpsi1vec <- pnorm(hpsi1(psi, eta, eta0), mean=0, sd=1)
    out <- -sum(y * log(Fhpsi1vec) + (n - y) * log(1 - Fhpsi1vec))
    out
```

```
}
neg.loglike.hpsi2.probit.est <- function(pair.beta.psi, y, n, X, eta0){
  beta <- pair.beta.psi[1:p]
  psi <- pair.beta.psi[p+1]
  eta <- X %*% beta
  Fhpsi2vec <- pnorm(hpsi2(psi, eta, eta0), mean=0, sd=1)
  out <- -sum(y * log(Fhpsi2vec) + (n - y) * log(1 - Fhpsi2vec))
  out
}
neg.loglike.hpsi12.probit.est <- function(pair.beta.psi, y, n, X, eta0){
  beta <- pair.beta.psi[1:p]
  psi <- pair.beta.psi[(p+1):(p+2)]
  eta <- X %*% beta
  Fhpsi12vec <- pnorm(hpsi12(psi[1], psi[2], eta, eta0), mean=0, sd=1)
  out <- -sum(y * log(Fhpsi12vec) + (n - y) * log(1 - Fhpsi12vec))
  out
}
neg.loglike.hpsi1.probit <- function(b, y, n, X, psi1, eta0){
  eta <- X %*% b
  Fhpsi1vec <- pnorm(hpsi1(psi1, eta, eta0), mean=0, sd=1)
  out <- -sum(y * log(Fhpsi1vec) + (n - y) * log(1 - Fhpsi1vec))
  out
}
neg.loglike.hpsi2.probit <- function(b, y, n, X, psi2, eta0){
  eta <- X %*% b
  Fhpsi2vec <- pnorm(hpsi2(psi2, eta, eta0), mean=0, sd=1)
  out <- -sum(y * log(Fhpsi2vec) + (n - y) * log(1 - Fhpsi2vec))
  out
}
neg.loglike.hpsi12.probit <- function(b, y, n, X, psi1, psi2, eta0){
  eta <- X %*% b
  Fhpsi12vec <- pnorm(hpsi12(psi1, psi2, eta, eta0), mean=0, sd=1)
  out <- -sum(y * log(Fhpsi12vec) + (n - y) * log(1 - Fhpsi12vec))
  out
}
# compute beta.opt and hessian matrix
if (tail.int == 1){ # Right tail
  # estimated
  opt.est <- optim(par=rep(0,p+1), fn = neg.loglike.hpsi1.probit.est, method="BFGS", y=y, n=n, X=X,
                   eta0=eta0, hessian=TRUE)
  par.opt.est <- opt.est$par
  hessian.est <- opt.est$hessian
  psi.est1 <- par.opt.est[p+1]
  # compute Hessian matrix for fixed psi1
  opt.fix <- optim(par=rep(0,p), fn = neg.loglike.hpsi1.probit, method="BFGS", y=y, n=n, X=X, psi1=psi1,
                   eta0=eta0, hessian=TRUE)
  beta.opt.fix <- opt.fix$par
  hessian.fix <- opt.fix$hessian
  cat("Probit GLM with ",tail," tail modified, psi1 = ", psi1, "\n\n") }
else if (tail.int == 2){ # Left tail
  # estimated
  opt.est <- optim(par=rep(0,p+1), fn = neg.loglike.hpsi2.probit.est, method="BFGS", y=y, n=n, X=X,
                   eta0=eta0, hessian=TRUE)
  par.opt.est <- opt.est$par
  hessian.est <- opt.est$hessian
  psi.est2 <- par.opt.est[p+1]
  # compute Hessian matrix for fixed psi2
  opt.fix <- optim(par=rep(0,p), fn = neg.loglike.hpsi2.probit, method="BFGS", y=y, n=n, X=X, psi2=psi2,
                   eta0=eta0, hessian=TRUE)
  beta.opt.fix <- opt.fix$par
  hessian.fix <- opt.fix$hessian
  cat("Probit GLM with ",tail," tail modified, psi2 = ", psi2, "\n\n")}
else if (tail.int == 3){ # Both tail
  # estimated
  opt.est <- optim(par=rep(0,p+2), fn = neg.loglike.hpsi12.probit.est, method="BFGS", y=y, n=n, X=X,
                   eta0=eta0, hessian=TRUE)
  par.opt.est <- opt.est$par
  hessian.est <- opt.est$hessian
  psi.est1 <- par.opt.est[p+1]
  psi.est2 <- par.opt.est[p+2]
```

```
  # compute Hessian matrix for fixed psi1 and psi2
  opt.fix <- optim(par=rep(0,p), fn = neg.loglike.hpsi12.probit, method="BFGS", y=y, X=X, n=n,
                   psi1=psi1, psi2=psi2, eta0=eta0, hessian=TRUE)
  beta.opt.fix <- opt.fix$par
  hessian.fix <- opt.fix$hessian
  cat("Probit GLM with ",tail," tail modified, psi1 = ", psi1, " and psi2 = ", psi2, "\n\n")}
else {
  stop("Please select tail")}
# compute fixed and estimated standard error
inf.est <- solve(hessian.est)
se.est <- sqrt(diag(abs(inf.est)))  # se.est[1:p] are the estimated se
inf.fix <- solve(hessian.fix)
se.fix <- sqrt(diag(abs(inf.fix)))  # se.fix[1:p] are the fixed se
# compute all values needed
coeff <- round(beta.opt.fix,4)
fixed.se <- round(se.fix[1:p],5)
fixed.z <- round(coeff/fixed.se, 3)
est.se <- round(se.est[1:p],4)
est.z <- round(coeff/est.se, 3)
var.infl <- round((est.se*100)/fixed.se - 100, 1)
# values from above in a matrix
mat <- as.matrix(cbind(coeff, fixed.se, fixed.z, est.se, est.z, var.infl))
# Regression parameter
cat("Regression Parameters:","\n")
covariates <- rep("0",p)
covariates[1] <- "(Intercept)"
if (is.null(dimnames(X)[[2]])){
  for (i in 1:(p-1)){
    covariates[i+1] <- paste("Covariate",i)}
}
else {
  temp <- dimnames(X)[[2]]
  covariates[2:p] <- temp[2:p]}
prmatrix(mat, quote=F, rowlab=covariates, collab=c(" coeff"," fixed se"," fixed z"," estimated se",
                                                   " estimated z"," var inflation"))
cat("\n")
# Link parameter
if (tail.int == 1){
  cat("Link Parameter:","\n")
  psi1.est.se <- round(se.est[p+1],4)
  cat("psi1 = ", psi1, ", standard error of psi1 = ", psi1.est.se,"\n\n") }
else if (tail.int == 2){
  cat("Link Parameter:","\n")
  psi2.est.se <- round(se.est[p+1],4)
  cat("psi2 = ", psi2, ", standard error of psi2 = ", psi2.est.se,"\n\n") }
else {
  cat("Link Parameters:","\n")
  psi1.est.se <- round(se.est[p+1],4)
  psi2.est.se <- round(se.est[p+2],4)
  cat("psi1 = ", psi1, ", standard error of psi1 = ", psi1.est.se,"\n")
  cat("psi2 = ", psi2, ", standard error of psi2 = ", psi2.est.se,"\n\n") }
# Correlation matrix
cat("Correlation matrix with estimated link:","\n")
if (tail.int == 1){
  corr.mat <- matrix(rep(0,(p+1)*(p+1)), nrow=p+1, ncol=p+1)
  for (i in 1:(p+1)){
    for (j in 1:(p+1)){
      if (i == j) {corr.mat[i,i] = 1.0000}
      else {corr.mat[i,j] = round(inf.est[i,j]/(sqrt(abs(inf.est[i,i]*inf.est[j,j]))),4)}
    }
  }
  rowlab <- c(covariates, "psi1")
  prmatrix(corr.mat, quote=F, rowlab=rowlab, collab=rowlab)}
else if (tail.int == 2){
  corr.mat <- matrix(rep(0,(p+1)*(p+1)), nrow=p+1, ncol=p+1)
  for (i in 1:(p+1)){
    for (j in 1:(p+1)){
      if (i == j) {corr.mat[i,i] = 1.0000}
      else {corr.mat[i,j] = round(inf.est[i,j]/(sqrt(abs(inf.est[i,i]*inf.est[j,j]))),4)}
    }
```

```
    }
    rowlab <- c(covariates, "psi2")
    prmatrix(corr.mat, quote=F, rowlab=rowlab, collab=rowlab)}
  else {
    corr.mat <- matrix(rep(0,(p+2)*(p+2)), nrow=p+2, ncol=p+2)
    for (i in 1:(p+2)){
      for (j in 1:(p+2)){
        if (i == j) {corr.mat[i,i] = 1.0000}
        else {corr.mat[i,j] = round(inf.est[i,j]/(sqrt(abs(inf.est[i,i]*inf.est[j,j]))),4)}
      }
    }
    rowlab <- c(covariates, "psi1", "psi2")
    prmatrix(corr.mat, quote=F, rowlab=rowlab, collab=rowlab)}
}

#################
# inf.poisson.r #
#################
# Gives standard errors for poisson GLM

inf.poisson <- function(y, X, tail, psi1, psi2, eta0=0){
  out <- list()
  p <- ncol(X)
  y[y == 0] <- 10^(-16)
  tail.int <- charmatch(tail, c("right", "left", "both"))

  neg.loglike.hpsi1.poiss.est <- function(pair.beta.psi, y, X, eta0){
    beta <- pair.beta.psi[1:p]
    psi <- pair.beta.psi[p+1]
    eta <- X %*% beta
    Fhpsi1vec <- exp(hpsi1(psi, eta, eta0))
    out <- -sum(y * hpsi1(psi, eta, eta0) - Fhpsi1vec)
    out
  }
  neg.loglike.hpsi1.poiss <- function(b, y, X, psi1, eta0){
    eta <- X %*% b
    Fhpsi1vec <- exp(hpsi1(psi1, eta, eta0))
    out <- -sum(y * hpsi1(psi1, eta, eta0) - Fhpsi1vec)
    out
  }
  # compute beta.opt and hessian matrix
  if (tail.int == 1){ # Right tail
    # estimated
    opt.est <- optim(par=rep(0,p+1), fn = neg.loglike.hpsi1.poiss.est, method="BFGS", y=y, X=X,
                     eta0=eta0, hessian=TRUE)
    par.opt.est <- opt.est$par
    hessian.est <- opt.est$hessian
    psi.est1 <- par.opt.est[p+1]
    # compute Hessian matrix for fixed psi1
    opt.fix <- optim(par=rep(0,p), fn = neg.loglike.hpsi1.poiss, method="BFGS", y=y, X=X, psi1=psi1,
                     eta0=eta0, hessian=TRUE)
    beta.opt.fix <- opt.fix$par
    hessian.fix <- opt.fix$hessian
    cat("Poisson GLM with ",tail," tail modified, psi1 = ", psi1, "\n\n")}
  else {
    stop("Poisson only available with right tail modification")}
  # compute fixed and estimated standard error
  inf.est <- solve(hessian.est)
  se.est <- sqrt(diag(abs(inf.est)))  # se.est[1:p] are the estimated se
  inf.fix <- solve(hessian.fix)
  se.fix <- sqrt(diag(abs(inf.fix)))  # se.fix[1:p] are the fixed se
  # compute all values needed
  coeff <- round(beta.opt.fix,4)
  fixed.se <- round(se.fix[1:p],5)
  fixed.z <- round(coeff/fixed.se, 3)
  est.se <- round(se.est[1:p],4)
  est.z <- round(coeff/est.se, 3)
  var.infl <- round((est.se*100)/fixed.se - 100, 1)
  # values from above in a matrix
  mat <- as.matrix(cbind(coeff, fixed.se, fixed.z, est.se, est.z, var.infl))
```

```
  # Regression parameter
  cat("Regression Parameters:","\n")
  covariates <- rep("0",p)
  covariates[1] <- "(Intercept)"
  if (is.null(dimnames(X)[[2]])){
    for (i in 1:(p-1)){
      covariates[i+1] <- paste("Covariate",i)}
  }
  else {
    temp <- dimnames(X)[[2]]
    covariates[2:p] <- temp[2:p]}
  prmatrix(mat, quote=F, rowlab=covariates, collab=c(" coeff"," fixed se"," fixed z"," estimated se",
                                                     " estimated z"," var inflation"))
  cat("\n")
  # Link parameter
  cat("Link Parameter:","\n")
  psi1.est.se <- round(se.est[p+1],4)
  cat("psi1 = ", psi1, ", standard error of psi1 = ", psi1.est.se,"\n\n")
  # Correlation matrix
  cat("Correlation matrix with estimated link:","\n")
  corr.mat <- matrix(rep(0,(p+1)*(p+1)), nrow=p+1, ncol=p+1)
  for (i in 1:(p+1)){
    for (j in 1:(p+1)){
      if (i == j) {corr.mat[i,i] = 1.0000}
      else {corr.mat[i,j] = round(inf.est[i,j]/(sqrt(abs(inf.est[i,i]*inf.est[j,j]))),4)}
    }
  }
  rowlab <- c(covariates, "psi1")
  prmatrix(corr.mat, quote=F, rowlab=rowlab, collab=rowlab)
}
```

A.3 R Code for Plots

```
### Figure 1 - Distribution of pcb ###
plot(seq(1,28,by=1), pcb.ex$pcb, pch=16, xlab="i", ylab="pcb", las=1)

### Figure 2 - Gauss distribution of log(pcb) ###
plot(seq(1,28,by=1), pcb.ex$log.pcb, pch=16, xlab="i", ylab="log(pcb)", las=1)

### Figure 3 - Histogram of log(pcb) ###
x <- pcb.ex$log.pcb
bins <- seq(-1, 4, by=0.25)
xfit<-seq(min(x),max(x),length=40)
yfit<-dnorm(xfit,mean=mean(x),sd=sd(x))
yfit <- yfit*length(bins)*0.5
hist(x, breaks=bins, xlab="log(pcb)", col="lightgrey", main="", las=1)
lines(xfit, yfit, lwd=2)

### Figure 4 - Bivariate Analysis of age.cen versus log.pcb ###
plot(pcb.ex$log.pcb, pcb.ex$age.cen, xlab="log(pcb)", ylab="age.cen", pch=16, las=1)

### Figure 5 - Binomial distribution of Yi/ni ###
plot(seq(1,8,by=1), (beetle.ex$y*100)/(beetle.ex$n), pch=16, xlab="i",
     ylab="Number of beetles killed in percentage", las=1)

### Figure 6 - Centured dose against probability of beetle's death ###
plot(beetle.ex$dose.cen, (beetle.ex$y/beetle.ex$n), ylab="probability",
 xlab="dose.cen", las=1, pch=16)

### Figure 7 - Distribution of yi/ni ####
y.sort <- sort(bys.ex$y)
n.sort <- bys.ex$n[sort.list(bys.ex$y)]
plot(seq(1,18,1),(y.sort)/(n.sort), ylab="y/n", xlab="", las=1, pch=16)

### Figure 8 - Binomial distribution of yi/ni in the rotifer data ###
probability <- rotifer.ex$y/rotifer.ex$n
sortprob <- sort(probability)
plot(seq(1,40,1),sortprob, ylab="y/n", xlab="", las=1, pch=16)

### Figure 9 - Poisson distribution of yi ###
plot(seq(1,44,1),mining.ex$y, ylab="y", xlab="i", las=1, pch=16)

### Figure 10 - Bivariate Analysis of inb versus y ###
plot(mining.ex$y, mining.ex$inb.cen, xlab="y", ylab="inb.cen", pch=16, las=1)

### Figure 11 - Bivariate Analysis of extrp versus y ###
plot(mining.ex$y, mining.ex$extrp.cen, xlab="y", ylab="extrp.cen", pch=16, las=1)

### Figure 12 - Bivariate Analysis of time versus y ###
plot(mining.ex$y, mining.ex$time.cen, xlab="y", ylab="time.cen", pch=16, las=1)

### Figure 13 - Bivariate Analysis of species versus density.cen ###
plot(rotifer.ex$species, rotifer.ex$density.cen, xlab="species",
     ylab="density.cen", pch=16, las=1, xlim=c(0,1), xaxp=c(0,1,1))

### Figure 14 - Right tail modification with hpsi1 ###
plot(eta, hpsi1(eta, psi1=1.75, eta0 = 0), type="l",lty=1, ylab="hpsi1",
     xlab=expression(eta), main="Right tail modification", las=1)
lines(eta, hpsi1(eta,psi1=-0.5, eta0=0), lty=2)
lines(eta, hpsi1(eta,psi1=3, eta0=0), lty=3)
legend(-4.5,10, c(expression(psi[1]==3), expression(psi[1]==1.75),
                  expression(psi[1]==-0.5)), lty = c(3, 1, 2), text.width=2)

### Figure 15 - Left tail modification with hpsi2 ###
plot(eta, hpsi2(eta, psi2=1.75, eta0 = 0), type="l",lty=1, ylab="hpsi2",
     xlab=expression(eta), main="Left tail modification", las=1)
lines(eta, hpsi2(eta,psi2=0, eta0=0), lty=2)
lines(eta, hpsi2(eta,psi2=3, eta0=0), lty=3)
legend(2,-3, c(expression(psi[2]==3),expression(psi[2]==1.75),
               expression(psi[2]==0)), lty = c(3, 1, 2), text.width=1.5)
```

```
### Figure 16 - Both tail modification with hpsi12 ###
plot(eta, hpsi12(eta, psi1 = 1.75,psi2=1.75, eta0 = 0), type="l",lty=1,
     ylab="hpsi12", xlab=expression(eta),main="Both tail modification",las=1)
lines(eta, hpsi12(eta,psi1=-0.5, psi2=2.5, eta0=0), lty=2)
lines(eta, hpsi12(eta, psi1=3, psi2=0, eta0=0), lty=3)
legend(-5.1,12, c(expression(paste(psi[1]==3,", ",psi[2]==0)),expression(paste
                  (psi[1]==1.75,", ",psi[2]==1.75)),expression(paste(psi[1]==-0.5,
                   ", ",psi[2]==2.5))), lty = c(3, 1, 2), text.width=3)
```

A.4 R Code for Chapter 4

```
### GAUSS: PCB-Daten ###
y <- pcb.ex$log.pcb
intercept <- rep(1, length(y))
age.cen <- pcb.ex$age.cen
X <- as.matrix(cbind(intercept, age.cen))
# Right tail
glm.profile(y, X, family="gauss", tail="right", psivec=seq(-0.3,1.5,by=0.05),eta0=0)
glm.model(y, X, family="gauss", tail="right", psi1=0.2, eta0=0)
glm.fitted(y, X, family="gauss", tail="right", psi1=0.2 ,eta0=0)
glm.inf(y, X, family="gauss", tail="right", psi1=0.2, eta0=0)

### LOGIT: Beetle-Daten ###
y <- beetle.ex$y
n <- beetle.ex$n
intercept <- rep(1, length(y))
dose.cen <- beetle.ex$dose.cen
X <- as.matrix(cbind(intercept, dose.cen))
#Left tail
glm.profile(y, X, n, family="logit", tail="left", psivec=seq(-0.5,1,by=0.05),eta0=0)
glm.model(y, X, n, family="logit", tail="left", psi2=0.15, eta0=0)
glm.fitted(y, X, n, family="logit", tail="left", psi2=0.15 ,eta0=0)
glm.inf(y, X, n, family="logit", tail="left", psi2=0.15, eta0=0)

### LOGIT: Byssinosis-Daten ###
y <- bys.ex$y
n <- bys.ex$n
intercept <- rep(1, length(y))
workspace <- bys.ex$workspace
smoking <- bys.ex$smoking
employment <- bys.ex$employment
X <- as.matrix(cbind(intercept, workspace, smoking, employment))
#Left tail
glm.profile(y, X, n, family="logit", tail="left", psivec=seq(0.25,0.5,by=0.05),eta0=0)
glm.model(y, X, n, family="logit", tail="left", psi2=0.3, eta0=0)
glm.fitted(y, X, n, family="logit", tail="left", psi2=0.3 ,eta0=0)
glm.inf(y, X, n, family="logit", tail="left", psi2=0.3 ,eta0=0)

### PROBIT: Rotifer-Daten ###
y <- rotifer.ex$y
n <- rotifer.ex$n
intercept <- rep(1, length(y))
species <- rotifer.ex$species
density.cen <- rotifer.ex$density.cen
speciesanddensity.cen <- rotifer.ex$species * rotifer.ex$density.cen
X <- as.matrix(cbind(intercept, species, density.cen, speciesanddensity.cen))
# Both tail
glm.profile(y, X, n, family="probit", tail="both", psivec=seq(-0.5,1.2,by=0.05),eta0=0)
glm.model(y, X, n, family="probit", tail="both", psi1=0, psi2=-0.5, eta0=0)
glm.fitted(y, X, n, family="probit", tail="both", psi1=0, psi2=-0.5, eta0=0)
glm.inf(y, X, n, family="probit", tail="both", psi1=0, psi2=-0.5, eta0=0)

### POISSON: Mining-Daten ###
y <- mining.ex$y
intercept <- rep(1,length(y))
inb.cen <- mining.ex$inb.cen
extrp.cen <- mining.ex$extrp.cen
X <- as.matrix(cbind(intercept, inb.cen, extrp.cen))
#Right tail
glm.profile(y, X, family="poiss", tail="right", psivec=seq(-1,1,by=0.065), eta0=0)
glm.model(y, X, family="poiss", tail="right", psi1=-0.61)
glm.fitted(y, X, family="poiss", tail="right", psi1=-0.61, eta0=0)
glm.inf(y, X, family="poiss", tail="right", psi1=-0.61, eta0=0)
```

B Data sets

pcb.ex

	pcb	log.pcb	age	age.cen
1	0.6	-0.5108256	1	-4.5357143
2	1.6	0.4700036	1	-4.5357143
3	0.5	-0.6931472	1	-4.5357143
4	1.2	0.1823216	1	-4.5357143
5	2.0	0.6931472	2	-3.5357143
6	1.3	0.2623643	2	-3.5357143
7	2.5	0.9162907	2	-3.5357143
8	2.2	0.7884574	3	-2.5357143
9	2.4	0.8754687	3	-2.5357143
10	1.2	0.1823216	3	-2.5357143
11	3.5	1.2527630	4	-1.5357143
12	4.1	1.4109870	4	-1.5357143
13	5.1	1.6292405	4	-1.5357143
14	5.7	1.7404662	5	-0.5357143
15	3.4	1.2237754	6	0.4642857
16	9.7	2.2721259	6	0.4642857
17	8.6	2.1517622	6	0.4642857
18	4.0	1.3862944	7	1.4642857
19	5.5	1.7047481	7	1.4642857
20	10.5	2.3513753	7	1.4642857
21	17.5	2.8622009	8	2.4642857
22	13.4	2.5952547	8	2.4642857
23	4.5	1.5040774	8	2.4642857
24	30.4	3.4144426	9	3.4642857
25	12.4	2.5176965	11	5.4642857
26	13.4	2.5952547	12	6.4642857
27	26.2	3.2657594	12	6.4642857
28	7.4	2.0014800	12	6.4642857

beetle.ex

	y	n	dose	dose.cen
1	6	59	1.6907	-0.102725
2	13	60	1.7242	-0.069225
3	18	62	1.7552	-0.038225
4	28	56	1.7842	-0.009225
5	52	63	1.8113	0.017875
6	53	59	1.8369	0.043475
7	61	62	1.8610	0.067575
8	60	60	1.8839	0.090475

bys.ex

	y	n	workspace	smoking	employment
1	30	233	-1	1	-1
2	7	126	-1	0	-1
3	16	67	-1	1	0
4	3	20	-1	0	0
5	41	151	-1	1	1
6	8	72	-1	0	1
7	3	403	0	1	-1
8	5	283	0	0	-1
9	2	94	0	1	0
10	1	51	0	0	0
11	4	237	0	1	1
12	3	232	0	0	1
13	11	951	1	1	-1
14	7	733	1	0	-1
15	3	320	1	1	0
16	1	160	1	0	0
17	15	733	1	1	1
18	5	553	1	0	1

rotifer.ex

	y	n	density	species	density.cen
1	11	58	1.019	1	-2.565
2	7	86	1.020	1	-2.465
3	10	76	1.021	1	-2.365
4	19	83	1.030	1	-1.465
5	9	56	1.030	1	-1.465
6	21	73	1.030	1	-1.465
7	13	29	1.031	1	-1.365
8	34	44	1.040	1	-0.465
9	10	31	1.040	1	-0.465
10	36	56	1.041	1	-0.365
11	20	27	1.048	1	0.335
12	54	59	1.049	1	0.435
13	20	22	1.050	1	0.535
14	9	14	1.050	1	0.535
15	14	17	1.060	1	1.535
16	10	22	1.061	1	1.635
17	64	66	1.063	1	1.835
18	68	86	1.070	1	2.535
19	488	492	1.070	1	2.535
20	88	89	1.070	1	2.535
21	13	161	1.019	0	-2.565
22	14	248	1.020	0	-2.465
23	30	234	1.021	0	-2.365
24	10	283	1.030	0	-1.465
25	14	129	1.030	0	-1.465
26	35	161	1.030	0	-1.465
27	26	167	1.031	0	-1.365
28	32	286	1.040	0	-0.465
29	22	117	1.040	0	-0.465
30	23	162	1.041	0	-0.365
31	7	42	1.048	0	0.335
32	22	48	1.049	0	0.435
33	9	49	1.050	0	0.535
34	34	160	1.050	0	0.535
35	71	74	1.060	0	1.535
36	25	45	1.061	0	1.635
37	94	101	1.063	0	1.835
38	63	68	1.070	0	2.535
39	178	190	1.070	0	2.535
40	154	154	1.070	0	2.535

mining.ex

	y	inb	extrp	time	inb.cen	extrp.cen	time.cen
1	2	50	70	1.0	-119.227273	-5.9318182	-6.1590909
2	1	230	65	6.0	60.772727	-10.9318182	-1.1590909
3	0	125	70	1.0	-44.227273	-5.9318182	-6.1590909
4	4	75	65	0.5	-94.227273	-10.9318182	-6.6590909
5	1	70	65	0.5	-99.227273	-10.9318182	-6.6590909
6	2	65	70	3.0	-104.227273	-5.9318182	-4.1590909
7	0	65	60	1.0	-104.227273	-15.9318182	-6.1590909
8	0	350	60	0.5	180.772727	-15.9318182	-6.6590909
9	4	350	90	0.5	180.772727	14.0681818	-6.6590909
10	4	160	80	0.0	-9.227273	4.0681818	-7.1590909
11	1	145	65	10.0	-24.227273	-10.9318182	2.8409091
12	4	145	85	0.0	-24.227273	9.0681818	-7.1590909
13	1	180	70	2.0	10.772727	-5.9318182	-5.1590909
14	5	43	80	0.0	-126.227273	4.0681818	-7.1590909
15	2	42	85	12.0	-127.227273	9.0681818	4.8409091
16	5	42	85	0.0	-127.227273	9.0681818	-7.1590909
17	5	45	85	0.0	-124.227273	9.0681818	-7.1590909
18	5	83	85	10.0	-86.227273	9.0681818	2.8409091
19	0	300	65	10.0	130.772727	-10.9318182	2.8409091
20	5	190	90	6.0	20.772727	14.0681818	-1.1590909
21	1	145	90	12.0	-24.227273	14.0681818	4.8409091
22	1	510	80	10.0	340.772727	4.0681818	2.8409091
23	3	65	75	5.0	-104.227273	-0.9318182	-2.1590909
24	3	470	90	9.0	300.772727	14.0681818	1.8409091
25	2	300	80	9.0	130.772727	4.0681818	1.8409091
26	2	275	90	4.0	105.772727	14.0681818	-3.1590909
27	0	420	50	17.0	250.772727	-25.9318182	9.8409091
28	1	65	80	15.0	-104.227273	4.0681818	7.8409091
29	5	40	75	15.0	-129.227273	-0.9318182	7.8409091
30	2	900	90	35.0	730.772727	14.0681818	27.8409091
31	3	95	88	20.0	-74.227273	12.0681818	12.8409091
32	3	40	85	10.0	-129.227273	9.0681818	2.8409091
33	3	140	90	7.0	-29.227273	14.0681818	-0.1590909
34	0	150	50	5.0	-19.227273	-25.9318182	-2.1590909
35	0	80	60	5.0	-89.227273	-15.9318182	-2.1590909
36	2	80	85	5.0	-89.227273	9.0681818	-2.1590909
37	0	145	65	9.0	-24.227273	-10.9318182	1.8409091
38	0	100	65	9.0	-69.227273	-10.9318182	1.8409091
39	3	150	80	3.0	-19.227273	4.0681818	-4.1590909
40	2	150	80	0.0	-19.227273	4.0681818	-7.1590909
41	3	210	75	2.0	40.772727	-0.9318182	-5.1590909
42	5	11	75	0.0	-158.227273	-0.9318182	-7.1590909
43	0	100	65	25.0	-69.227273	-10.9318182	17.8409091
44	3	50	88	20.0	-119.227273	12.0681818	12.8409091