1 Semi- und Nonparametrische Regression (II)

Aufgabe 1

Vorbereitungen

```
# Einlesen der Daten
fakesoep <- read.table("fakesoep.dat",header=T)
# Package mgcv laden
library(mgcv)
```

a) Datensatz aufteilen

```
fakesoepW <- fakesoep[fakesoep$geschl == 1,]
fakesoepM <- fakesoep[fakesoep$geschl == 0,]
```

In Blatt 11 wurde ein Additives Modell (AM) betrachtet. Nun haben wir einen Gamma-verteilten Response, also brauchen wir Generalisierte Additive Modelle (GAM)!

In der gam()-Funktion können dieselben Link-Funktionen sowie Families spezifiziert werden wie in glm().

```
modelW <- gam(beink ~ s(groesse,bs="cr") + s(alter,bs="cr") + s(dauer,bs="cr")
+ verh + deutsch + abitur, family=Gamma(link="log"), data=fakesoepW)
modelM <- gam(beink ~ s(groesse,bs="cr") + s(alter,bs="cr") + s(dauer,bs="cr")
+ verh + deutsch + abitur, family=Gamma(link="log"), data=fakesoepM)

# Ergebnis betrachten
summary(modelW)


##
## Family: Gamma
## Link function: log
##
## Formula:
## beink ~ s(groesse, bs = "cr") + s(alter, bs = "cr") + s(dauer,
##     bs = "cr") + verh + deutsch + abitur
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.57156    0.04657 162.593   <2e-16 ***
## verh        -0.22448    0.02653  -8.462   <2e-16 ***
## deutsch     -0.04411    0.04441  -0.993    0.321
## abitur       0.28801    0.02558  11.258   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##              edf Ref.df     F  p-value
## s(groesse) 1.001  1.001 10.91 0.000978 ***
## s(alter)   6.469  7.651  2.31 0.018774 *
```

```
## s(dauer)   6.158  7.282 17.07  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.208   Deviance explained = 22.3%
## GCV = 0.17816  Scale est. = 0.17406   n = 1280
```

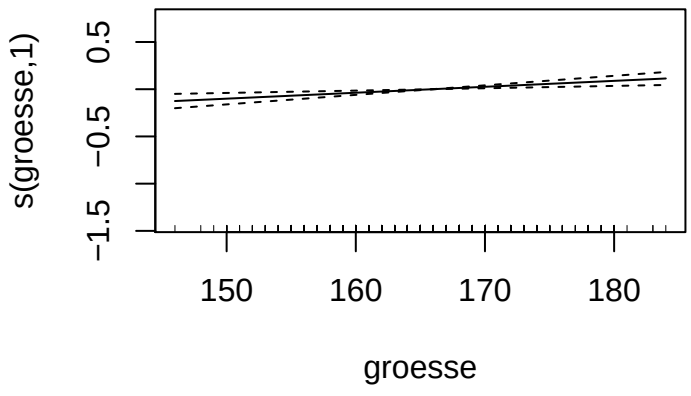
```
summary(modelM)
```

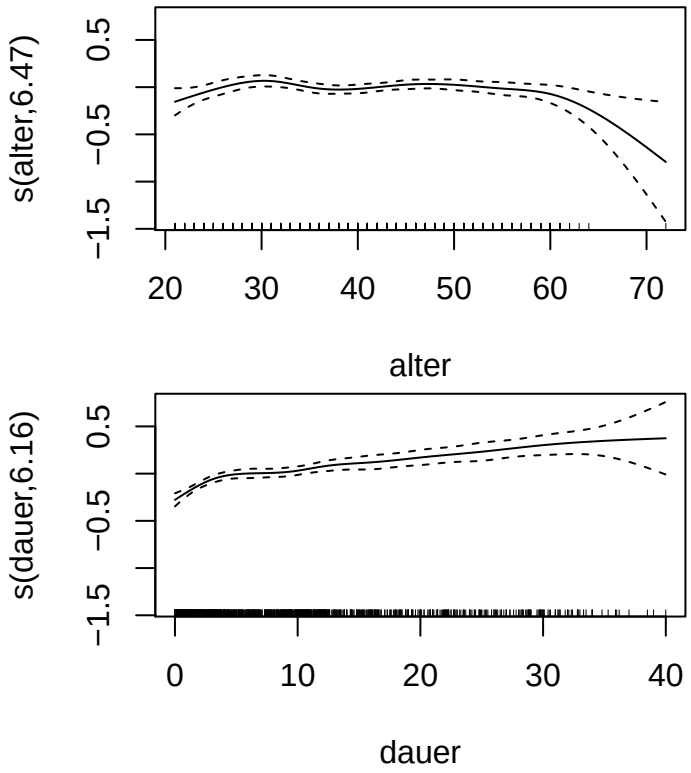
```
##
## Family: Gamma
## Link function: log
##
## Formula:
## beink ~ s(groesse, bs = "cr") + s(alter, bs = "cr") + s(dauer,
##     bs = "cr") + verh + deutsch + abitur
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.71378    0.03831 201.366  < 2e-16 ***
## verh         0.09797    0.02323   4.216 2.61e-05 ***
## deutsch      0.03077    0.03469   0.887    0.375
## abitur       0.35975    0.02125  16.925  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##              edf Ref.df      F  p-value
## s(groesse) 1.002  1.005 38.475 6.64e-10 ***
## s(alter)   4.362  5.411  6.314 6.22e-06 ***
## s(dauer)   2.799  3.512 19.425 1.36e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =   0.25   Deviance explained =   27%
## GCV = 0.14392  Scale est. = 0.15679   n = 1720
```

Nach den geschätzen Modellen wirkt sich bei Frauen eine Hochzeit im Mittel offenbar negativ auf das Einkommen aus, bei Männern hingegen positiv. Der Einfluss der Variablen Abitur ist für Männer und Frauen positiv. D.h. das Einkommen steigt im Mittel, wenn man Abitur hat.

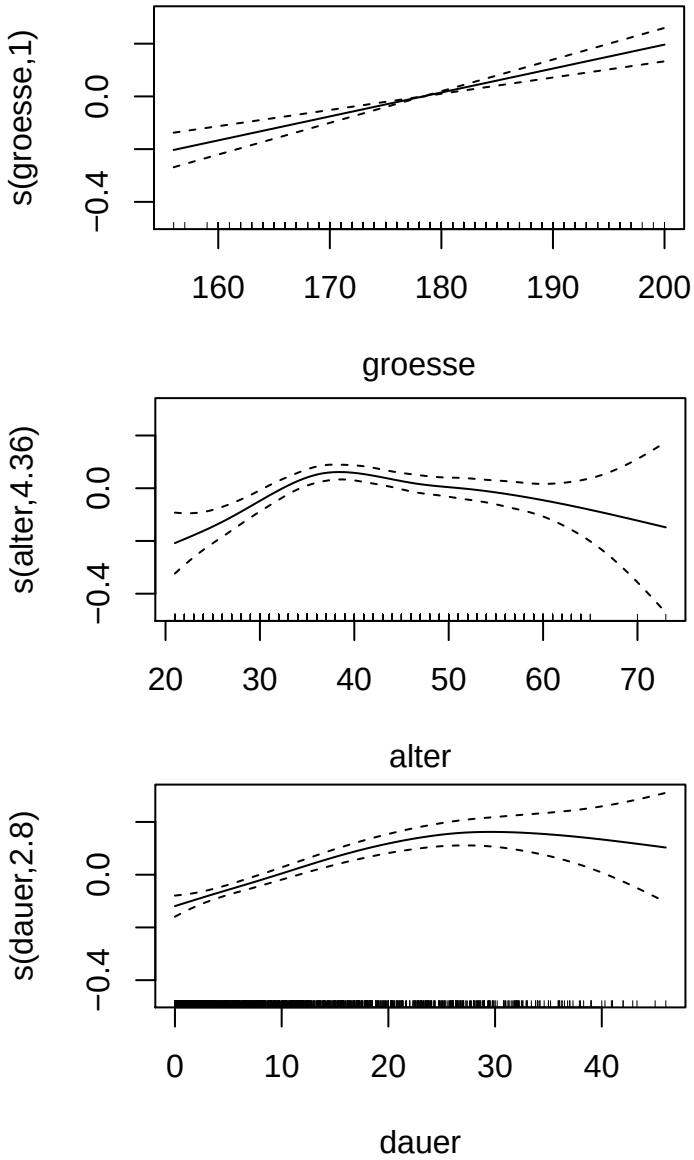
Die glatten Kurven können durch Plots dargestellt werden. Der Einfluss von Körpergröße auf Einkommen erscheint (bei Frauen und Männern) nahezu linear.

```
plot(modelW)
```




s(alter,6.47)
0.5
−0.5
−1.5
20
30
40
50
60
70
alter
s(dauer,6.16)
0.5
−0.5
−1.5
0
10
20
30
40
dauer

```
plot(modelM)
```


s(groesse,1)
0.0
−0.4
160
170
180
190
200
groesse
s(alter,4.36)
0.0
−0.4
20
30
40
50
60
70
alter
s(dauer,2.8)
0.0
−0.4
0
10
20
30
40
dauer

b) Um die beiden obigen Modelle in einem Modell zusammenzufassen, muss die Variable Geschlecht sowie Interaktionen aller anderen Variablen mit Geschlecht ins Modell aufgenommen werden. Hierfür müssen zunächst Geschlechts-Dummies erzeugt werden

```
geschl1 <- fakesoep$geschl
geschl0 <- 1-geschl1

modelAll <- gam(beink ~ s(groesse,bs="cr",by=geschl0) +
s(alter,bs="cr",by=geschl0) + s(dauer,bs="cr",by=geschl0) +
s(groesse,bs="cr",by=geschl1) +
s(alter,bs="cr",by=geschl1) + s(dauer,bs="cr",by=geschl1) +
geschl + verh + deutsch + abitur + verh:geschl + deutsch:geschl + abitur:geschl,
family=Gamma(link="log"),data=fakesoep)

# An den Parameter-Schätzern der parametrisch aufgenommenen Variablen erkennt
# man die Äquivalenz der Modelle (abgesehen von kleineren Ungenauigkeiten)
summary(modelAll)
```

```
##
## Family: Gamma
## Link function: log
##
## Formula:
## beink ~ s(groesse, bs = "cr", by = geschl0) + s(alter, bs = "cr",
##     by = geschl0) + s(dauer, bs = "cr", by = geschl0) + s(groesse,
##     bs = "cr", by = geschl1) + s(alter, bs = "cr", by = geschl1) +
##     s(dauer, bs = "cr", by = geschl1) + geschl + verh + deutsch +
##     abitur + verh:geschl + deutsch:geschl + abitur:geschl
##
## Parametric coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     7.83710    0.06269 125.011  < 2e-16 ***
## geschl         -0.15435    0.16899  -0.913   0.3611
## verh            0.09825    0.02376   4.135 3.65e-05 ***
## deutsch         0.03060    0.03548   0.862   0.3885
## abitur          0.35974    0.02174  16.548  < 2e-16 ***
## geschl:verh    -0.32221    0.03505  -9.193  < 2e-16 ***
## geschl:deutsch -0.07462    0.05582  -1.337   0.1814
## geschl:abitur  -0.07229    0.03301  -2.190   0.0286 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                       edf Ref.df      F  p-value
## s(groesse):geschl0 1.001  1.002 36.805 1.40e-09 ***
## s(alter):geschl0   4.132  5.140  6.012 1.43e-05 ***
## s(dauer):geschl0   2.692  3.381 19.062 3.88e-13 ***
## s(groesse):geschl1 1.001  1.003 11.656 0.000663 ***
## s(alter):geschl1   6.842  7.931  2.729 0.005517 **
## s(dauer):geschl1   6.286  7.398 17.814  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Rank: 62/68
## R-sq.(adj) =  0.356   Deviance explained = 38.5%
## GCV = 0.15839  Scale est. = 0.1641    n = 3000
```

```
summary(modelW)
```

```
##
## Family: Gamma
## Link function: log
```

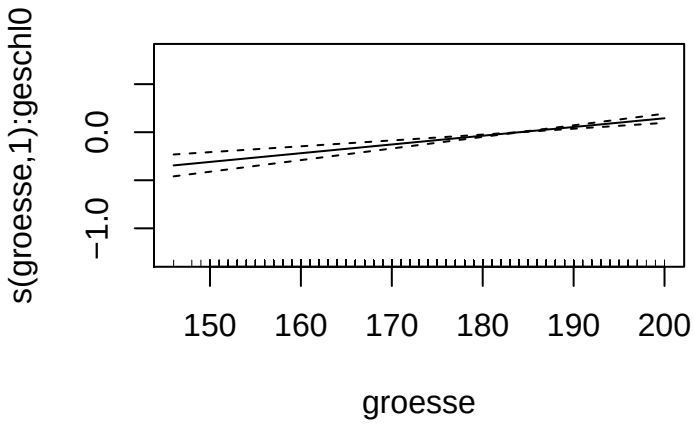
```
##
## Formula:
## beink ~ s(groesse, bs = "cr") + s(alter, bs = "cr") + s(dauer,
##     bs = "cr") + verh + deutsch + abitur
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.57156    0.04657 162.593   <2e-16 ***
## verh        -0.22448    0.02653  -8.462   <2e-16 ***
## deutsch     -0.04411    0.04441  -0.993    0.321
## abitur       0.28801    0.02558  11.258   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##              edf Ref.df     F  p-value
## s(groesse) 1.001  1.001 10.91 0.000978 ***
## s(alter)   6.469  7.651  2.31 0.018774 *
## s(dauer)   6.158  7.282 17.07  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.208   Deviance explained = 22.3%
## GCV = 0.17816  Scale est. = 0.17406   n = 1280
```

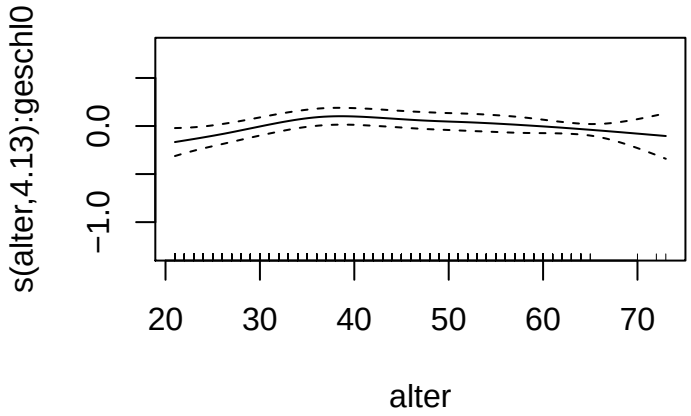
```
summary(modelM)
```

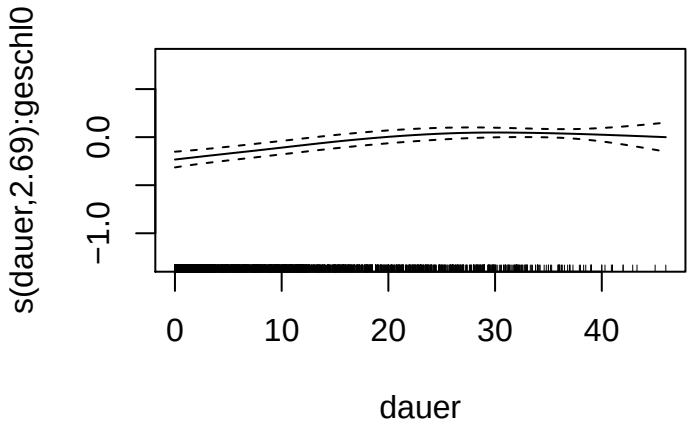
```
##
## Family: Gamma
## Link function: log
##
## Formula:
## beink ~ s(groesse, bs = "cr") + s(alter, bs = "cr") + s(dauer,
##     bs = "cr") + verh + deutsch + abitur
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.71378    0.03831 201.366  < 2e-16 ***
## verh         0.09797    0.02323   4.216 2.61e-05 ***
## deutsch      0.03077    0.03469   0.887    0.375
## abitur       0.35975    0.02125  16.925  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##              edf Ref.df      F  p-value
## s(groesse) 1.002  1.005 38.475 6.64e-10 ***
## s(alter)   4.362  5.411  6.314 6.22e-06 ***
## s(dauer)   2.799  3.512 19.425 1.36e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =   0.25   Deviance explained =   27%
## GCV = 0.14392  Scale est. = 0.15679   n = 1720
```

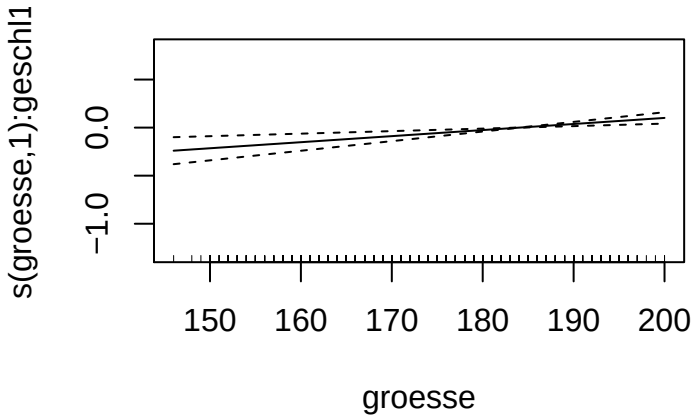
Auch das Ergebnis der glatten Modellierung ist ähnlich.

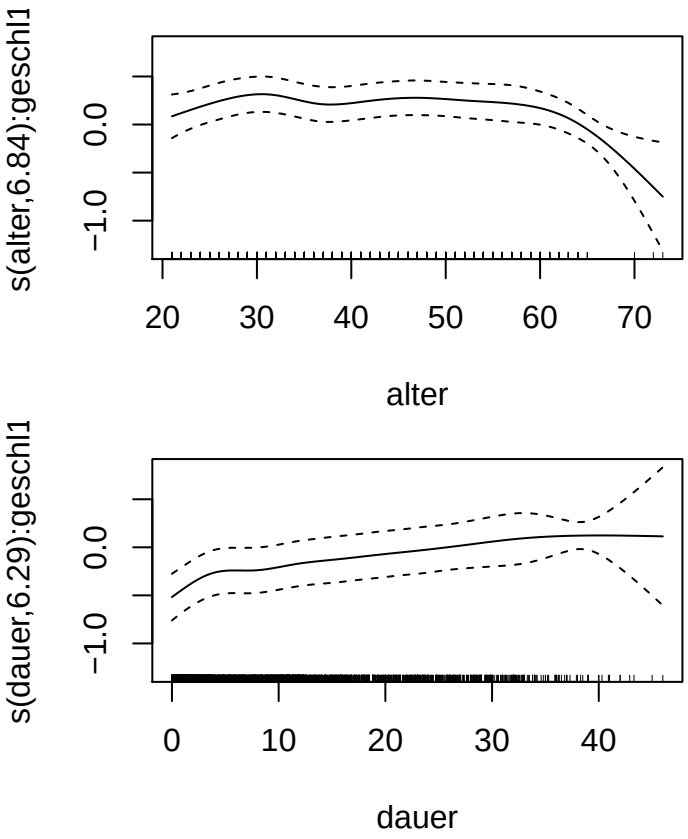
```
plot(modelAll)
```


s(groesse,1):geschl0
−1.0
0.0
150
160
170
180
190
200
groesse


s(alter,4.13):geschl0
−1.0
0.0
20
30
40
50
60
70
alter


s(dauer,2.69):geschl0
−1.0
0.0
0
10
20
30
40
dauer


s(groesse,1):geschl1
−1.0
0.0
150
160
170
180
190
200
groesse

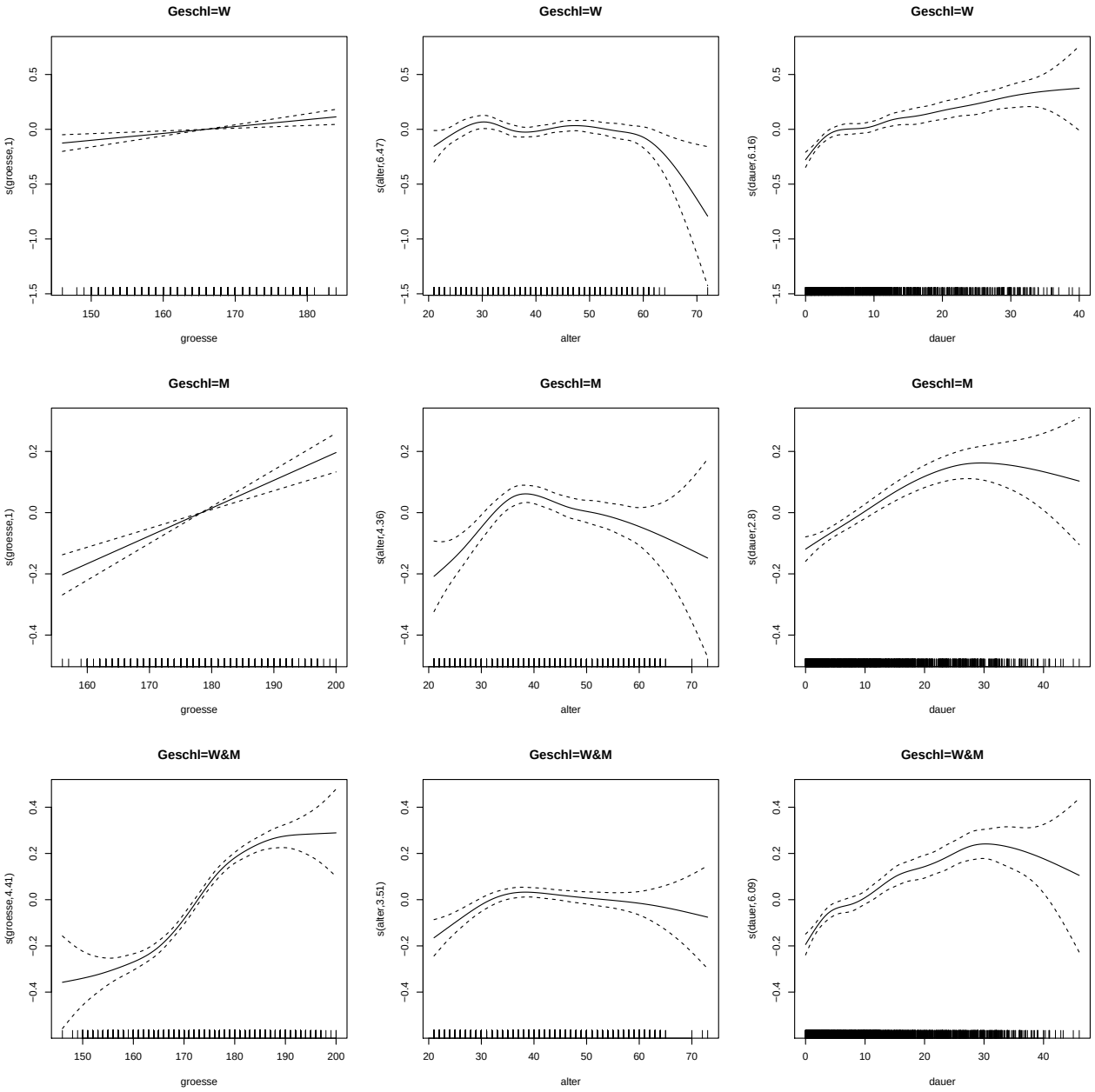


Gemeinsames Modell ohne Trennung nach Geschlecht:

```
modelWM <- gam(beink ~ s(groesse,bs="cr") + s(alter,bs="cr") + s(dauer,bs="cr")
+ verh + deutsch + abitur, family=Gamma(link="log"), data=fakesoep)
```

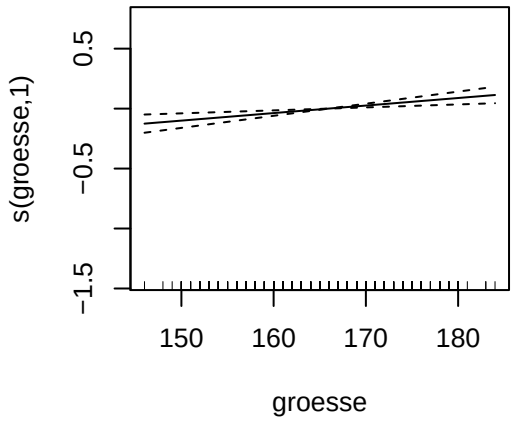
Vergleich

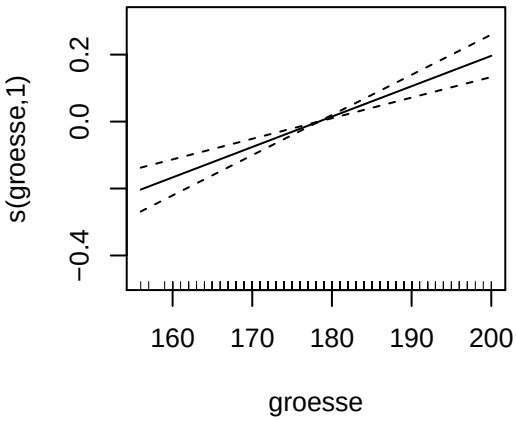
```
par(mfrow=c(3,3))

plot(modelW, main="Geschl=W")
plot(modelM, main="Geschl=M")
# für den gesamten Datensatz (Frauen und Männer zusammen)
plot(modelWM, main="Geschl=W&M")
```

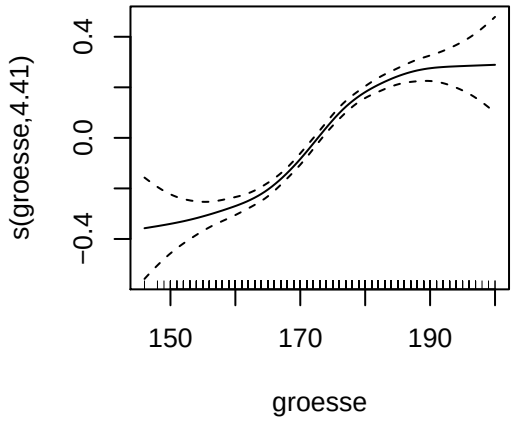


Vergleich des Effekts der Größe zwischen gemeinsamen Modell und den beiden nach Geschlechtern getrennten Modellen:

```
par(mfrow=c(2,2))
plot(modelW, select=1)
plot(modelM, select=1)
plot(modelWM, select=1)
```







Der Effekt der Größe auf das Einkommen ist sowohl für Männer als auch für Frauen eine Gerade, aber das allgemeine Niveau dieses Größeneffekts unter- scheidet sich zwischen den Geschlechtern, er ist bei Männern höher. Diese Niveauunterschiede sieht man in den einzelnen Plots wegen der Zentrierung der Kurven um Null herum nicht. Die nonparametrisch geschätzte Kurve in der dritten Grafik zeigt, wie der Schätzer versucht, von der niedrigeren zur höheren Gerade zu springen. In den extremen Größenbereichen, in denen fast nur eines der Geschlechter vorkommt, reproduziert der Schätzer den Verlauf der jeweiligen Gerade. Fazit: geschätzte Kurven in (G)AMs sind empfindlich gegenüber Interaktionen mit übrigen Einflussgrößen. (Hier dem Geschlecht.)

c) Eine Alternative zur Gamma-Verteilung ist die inverse Gauß-Verteilung (vgl. Blatt 4)

```
modelAllinvG <- gam(beink ~ s(groesse,bs="cr",by=geschl0) +
s(alter,bs="cr",by=geschl0) + s(dauer,bs="cr",by=geschl0) +
s(groesse,bs="cr",by=geschl1) +
s(alter,bs="cr",by=geschl1) + s(dauer,bs="cr",by=geschl1) +
geschl + verh + deutsch + abitur + verh:geschl + deutsch:geschl + abitur:geschl,
family=inverse.gaussian(link="log"),data=fakesoep)
```

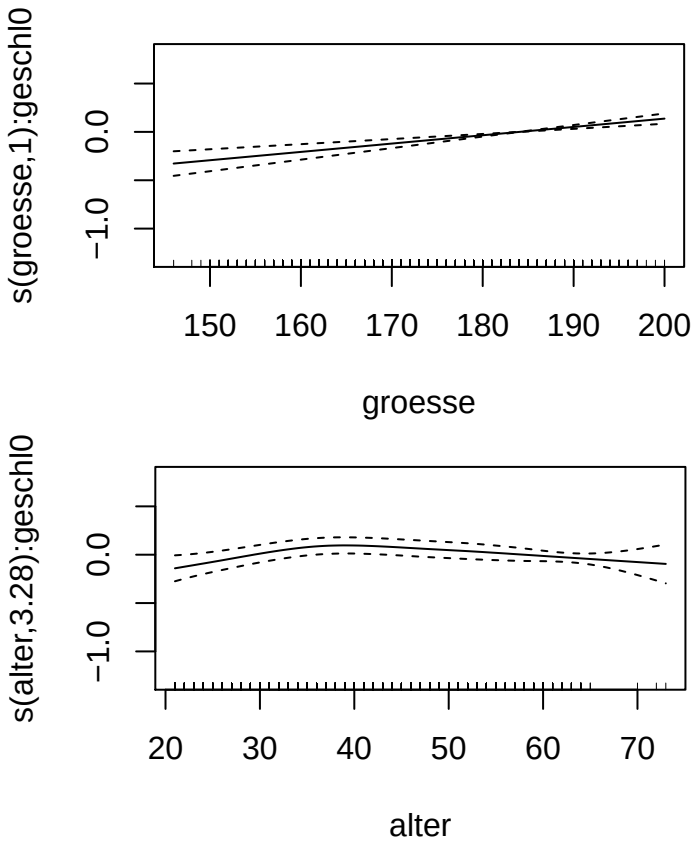
Die Ergebnisse sind sehr ähnlich.

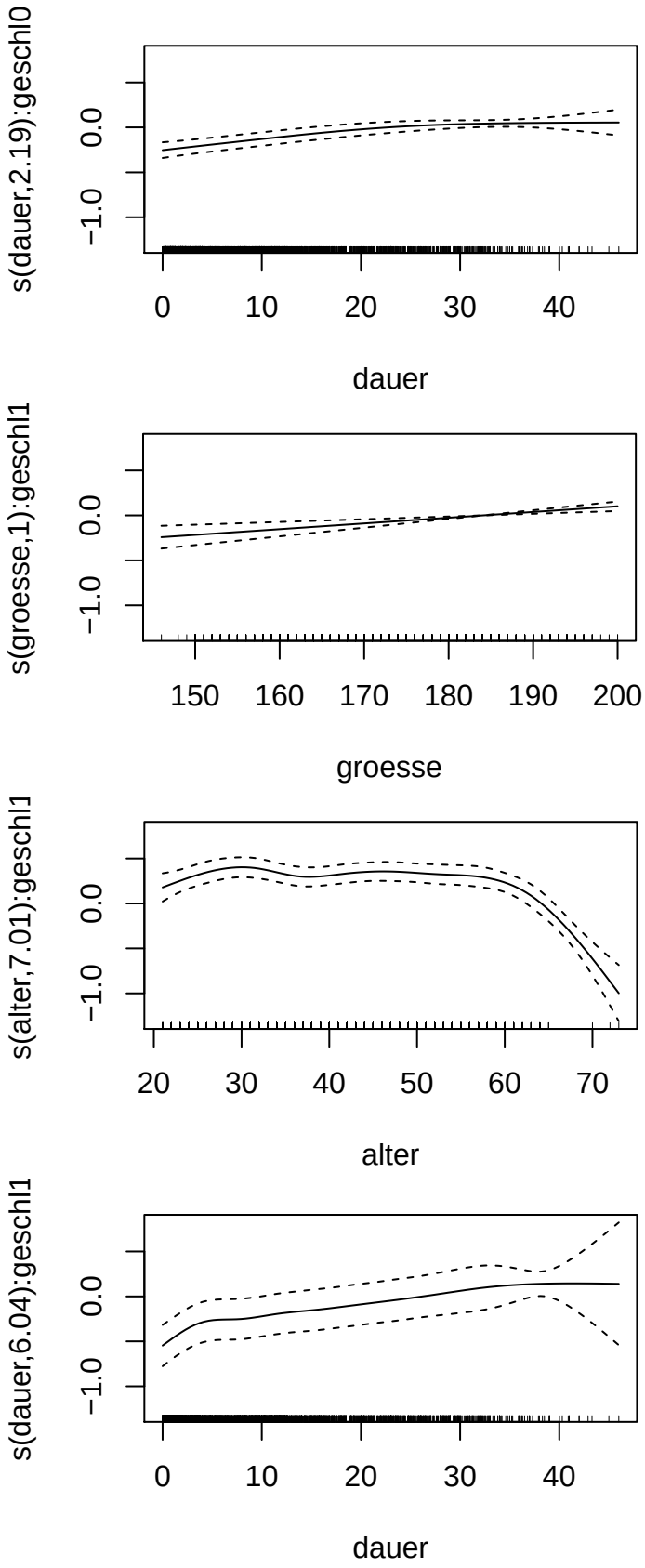
```
summary(modelAllinvG)
```

```
##
## Family: inverse.gaussian
## Link function: log
##
## Formula:
## beink ~ s(groesse, bs = "cr", by = geschl0) + s(alter, bs = "cr",
##     by = geschl0) + s(dauer, bs = "cr", by = geschl0) + s(groesse,
```

```
##     bs = "cr", by = geschl1) + s(alter, bs = "cr", by = geschl1) +
##     s(dauer, bs = "cr", by = geschl1) + geschl + verh + deutsch +
##     abitur + verh:geschl + deutsch:geschl + abitur:geschl
##
## Parametric coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     7.86131    0.06411 122.623  < 2e-16 ***
## geschl         -0.23273    0.14699  -1.583  0.11346
## verh            0.09664    0.02613   3.699  0.00022 ***
## deutsch         0.02230    0.03806   0.586  0.55794
## abitur          0.35337    0.02631  13.431  < 2e-16 ***
## geschl:verh    -0.32207    0.03537  -9.105  < 2e-16 ***
## geschl:deutsch -0.07519    0.05431  -1.384  0.16632
## geschl:abitur  -0.06336    0.03556  -1.782  0.07487 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                       edf Ref.df      F  p-value
## s(groesse):geschl0 1.001  1.003 26.680 2.60e-07 ***
## s(alter):geschl0   3.279  4.115  5.809 0.000103 ***
## s(dauer):geschl0   2.187  2.755 16.749 4.86e-10 ***
## s(groesse):geschl1 1.000  1.001 14.637 0.000134 ***
## s(alter):geschl1   7.013  8.046  7.858 1.57e-10 ***
## s(dauer):geschl1   6.039  7.148 25.376  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Rank: 62/68
## R-sq.(adj) =  0.353   Deviance explained = 34.2%
## GCV = 8.3975e-05  Scale est. = 7.6502e-05  n = 3000
```

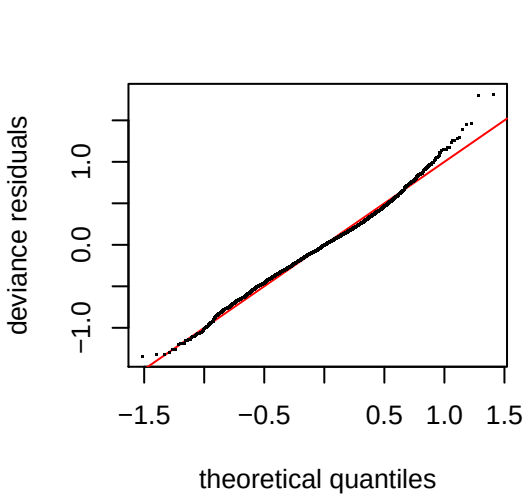
```
plot(modelAllinvG)
```

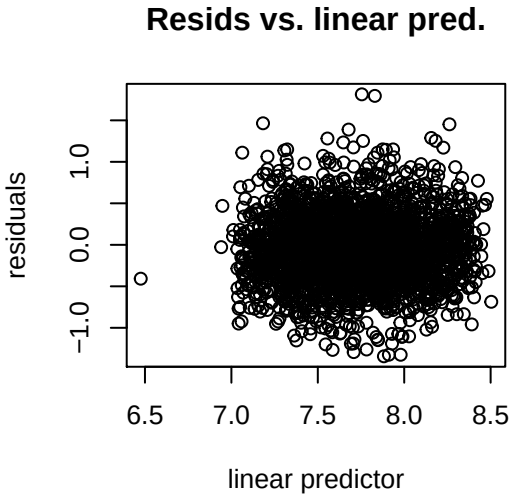


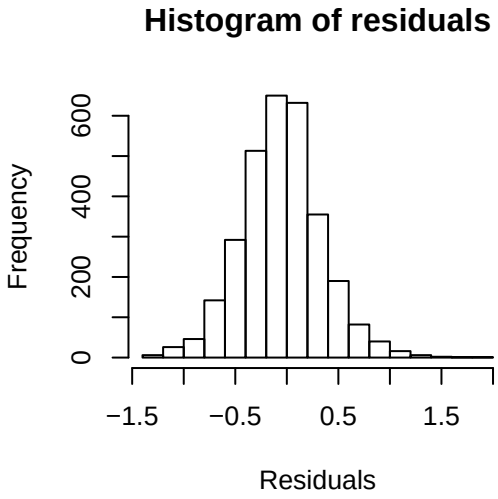


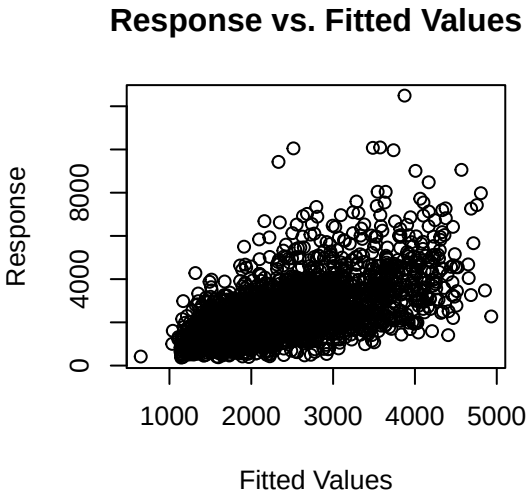
Diagnostik-Plot können mittel gam.check() erzeugt werden.

```
gam.check(modelAll)
```



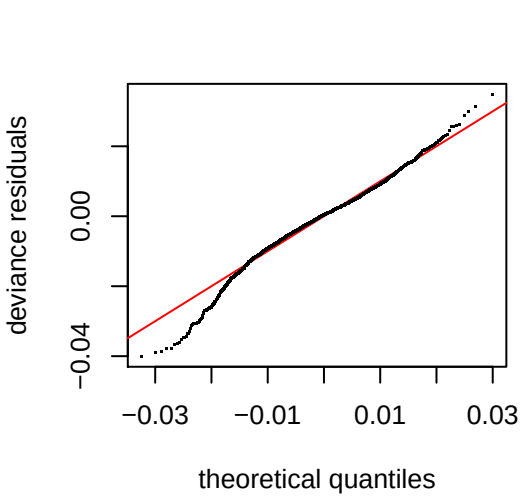


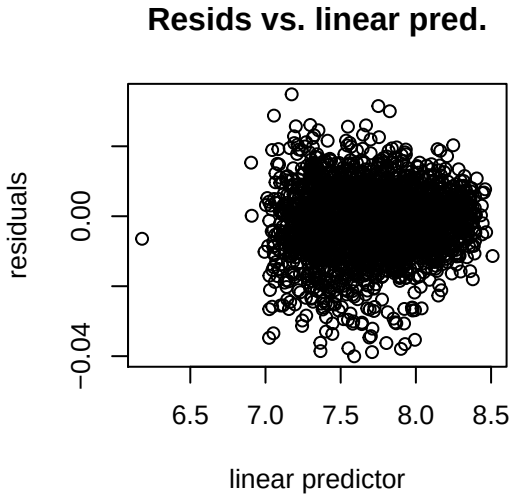


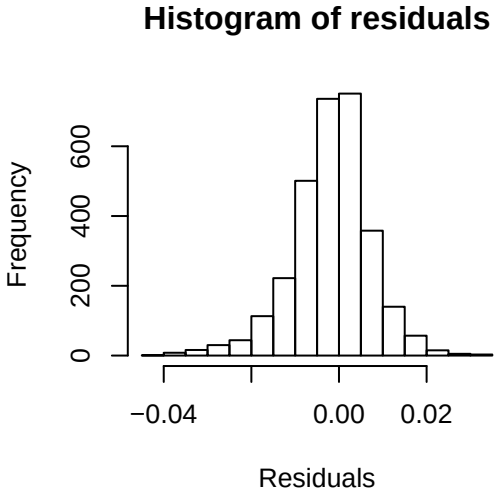


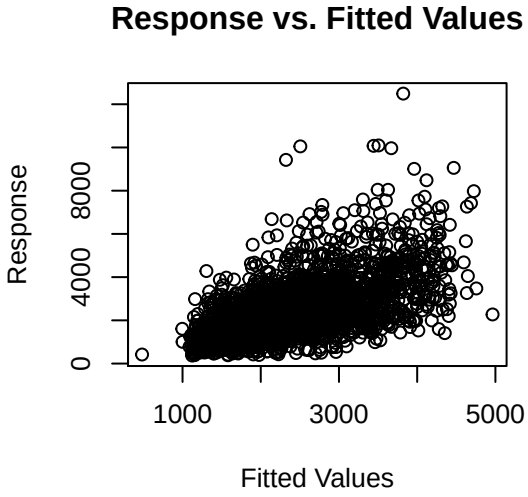

```
##
## Method: GCV   Optimizer: outer newton
## full convergence after 6 iterations.
## Gradient range [-7.537684e-08,1.05339e-09]
## (score 0.1583946 & scale 0.1641047).
## Hessian positive definite, eigenvalue range [2.330142e-08,6.977471e-05].
## Model rank =  62 / 68
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##                         k'    edf k-index p-value
## s(groesse):geschl0 10.000  1.001   0.975    0.34
## s(alter):geschl0   10.000  4.132   0.991    0.66
## s(dauer):geschl0   10.000  2.692   0.971    0.22
## s(groesse):geschl1 10.000  1.001   0.975    0.32
## s(alter):geschl1   10.000  6.842   0.991    0.67
## s(dauer):geschl1   10.000  6.286   0.971    0.24
```

```
gam.check(modelAllinvG)
```










```
##
## Method: GCV   Optimizer: outer newton
## full convergence after 7 iterations.
## Gradient range [-3.973275e-11,3.074378e-13]
## (score 8.397512e-05 & scale 7.65016e-05).
## Hessian positive definite, eigenvalue range [1.098304e-11,5.639196e-08].
## Model rank =  62 / 68
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##                         k'    edf k-index p-value
## s(groesse):geschl0 10.000  1.001   0.956    0.26
## s(alter):geschl0   10.000  3.279   0.967    0.58
## s(dauer):geschl0   10.000  2.187   0.952    0.23
## s(groesse):geschl1 10.000  1.000   0.956    0.32
## s(alter):geschl1   10.000  7.013   0.967    0.57
## s(dauer):geschl1   10.000  6.039   0.952    0.20
```

Offenbar ist die Anpassung der (Devianz-)Residuen an die Normalverteilung im Gamma-Modell besser, weshalb jenes dem Invers-Gauß-Modell vorzuziehen ist.

d) Wir betrachten aus Gründen der Übersichtlichkeit nun Frauen und Männer wieder getrennt und schätzen zum Vergleich GLMs (vgl. Blatt 4).

```
glmW <- gam(beink ~ groesse + alter + dauer
+ verh + deutsch + abitur,family=Gamma(link=log),data=fakesoepW)
```

```
glmM <- gam(beink ~ groesse + alter + dauer
+ verh + deutsch + abitur,family=Gamma(link=log),data=fakesoepM)
```

Der Vergleich mit den GAMs aus (a) kann mittel LR-Tests erfolgen

```
anova(modelW,glmW,test="F")
```

```
## Analysis of Deviance Table
##
## Model 1: beink ~ s(groesse, bs = "cr") + s(alter, bs = "cr") + s(dauer,
##     bs = "cr") + verh + deutsch + abitur
## Model 2: beink ~ groesse + alter + dauer + verh + deutsch + abitur
##   Resid. Df Resid. Dev      Df Deviance      F    Pr(>F)
## 1    1260.1     221.81
## 2    1273.0     229.54 -12.935  -7.7315 3.4341 3.118e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(modelM,glmM,test="F")
```

```
## Analysis of Deviance Table
##
## Model 1: beink ~ s(groesse, bs = "cr") + s(alter, bs = "cr") + s(dauer,
##     bs = "cr") + verh + deutsch + abitur
## Model 2: beink ~ groesse + alter + dauer + verh + deutsch + abitur
##   Resid. Df Resid. Dev      Df Deviance      F    Pr(>F)
## 1    1706.1     244.06
## 2    1713.0     252.62 -6.9275  -8.5644 7.8848 2.329e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Offenbar sind GAMs in beiden Fällen zur Modellierung wesentlich besser geeignet.

Insbesondere bei der Kovariablen Körpergröße stellt sich aber die Frage, ob der Einfluss nonparametrisch modelliert werden sollte. Nach den Plots in (a) zu urteilen, ist die glatte Modellierung hier nicht notwendig.

```
modelW_k <- gam(beink ~ groesse + s(alter,bs="cr") + s(dauer,bs="cr")
+ verh + deutsch + abitur,family=Gamma(link=log),data=fakesoepW)
modelM_k <- gam(beink ~ groesse + s(alter,bs="cr") + s(dauer,bs="cr")
+ verh + deutsch + abitur,family=Gamma(link=log),data=fakesoepM)
```

Paradoxerweise wird aber bei Verwendung von Tests die nonparametrische Modellierung der parametrischen vorgezogen, obwohl (bzw. weil) der geschätzte Einfluss (fast) linear ist.

```
anova(modelW,modelW_k,test="F")
```

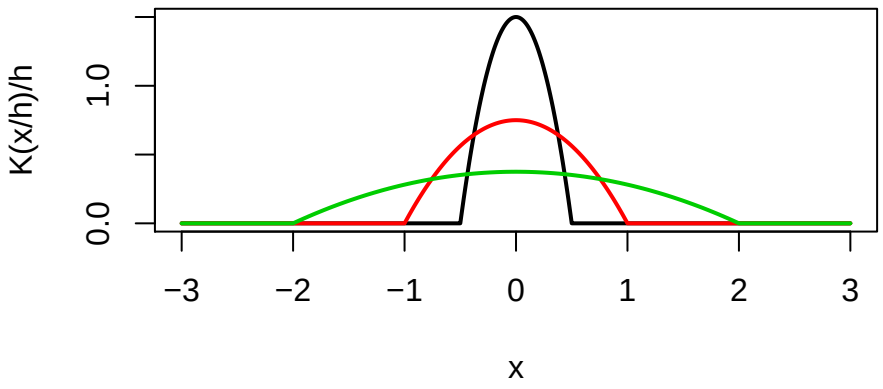
```
## Analysis of Deviance Table
##
## Model 1: beink ~ s(groesse, bs = "cr") + s(alter, bs = "cr") + s(dauer,
##     bs = "cr") + verh + deutsch + abitur
## Model 2: beink ~ groesse + s(alter, bs = "cr") + s(dauer, bs = "cr") +
##     verh + deutsch + abitur
##   Resid. Df Resid. Dev         Df    Deviance      F   Pr(>F)
## 1    1260.1     221.81
## 2    1260.1     221.81 -0.0011437 -8.9877e-05 0.4515 0.004386 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(modelM,modelM_k,test="F")

## Analysis of Deviance Table
##
## Model 1: beink ~ s(groesse, bs = "cr") + s(alter, bs = "cr") + s(dauer,
##     bs = "cr") + verh + deutsch + abitur
## Model 2: beink ~ groesse + s(alter, bs = "cr") + s(dauer, bs = "cr") +
##     verh + deutsch + abitur
##   Resid. Df Resid. Dev         Df    Deviance      F  Pr(>F)
## 1    1706.1     244.06
## 2    1706.1     244.06 -0.0048848 -0.00058527 0.7641 0.01385 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Aufgabe 2

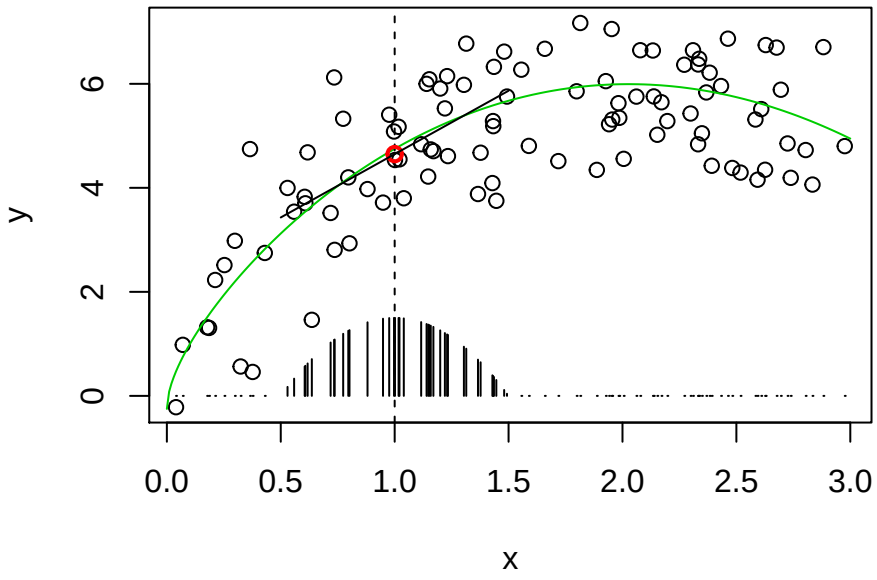
Veranschaulichung von lokaler linearer Regression

```
# Epanechnikov-Kern
epan <- function(x,hvalue)
  {
    ifelse(abs(x/hvalue)<=1,(3/4)*(1-(x/hvalue)^2), 0)/hvalue
  }

xseq <- seq(-3,3,by=0.01)
plot(xseq,epan(xseq,0.5),type="l",xlab="x",ylab="K(x/h)/h",lwd=2)
lines(xseq,epan(xseq,1),col=2,lwd=2)
lines(xseq,epan(xseq,2),col=3,lwd=2)
```

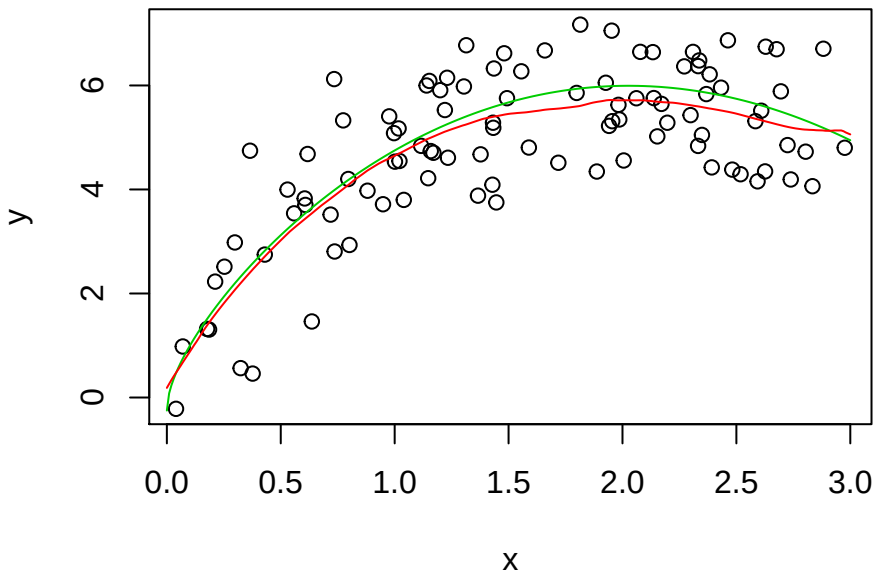



```
# Daten
set.seed(1)
x <- 3*runif(100)
fx <- function(x){2 - (x-1.5)^2 + 3*sqrt(x)}
set.seed(10)
epsilon <- rnorm(100)
y <- fx(x) + epsilon
plot(x,y,xlab="x",ylab="y")
xseq <- seq(0,3,by=0.01)
lines(xseq,fx(xseq),col=3)

# neues x
xnew <- 1
abline(v=xnew,lty=2)

# Gewichte
```

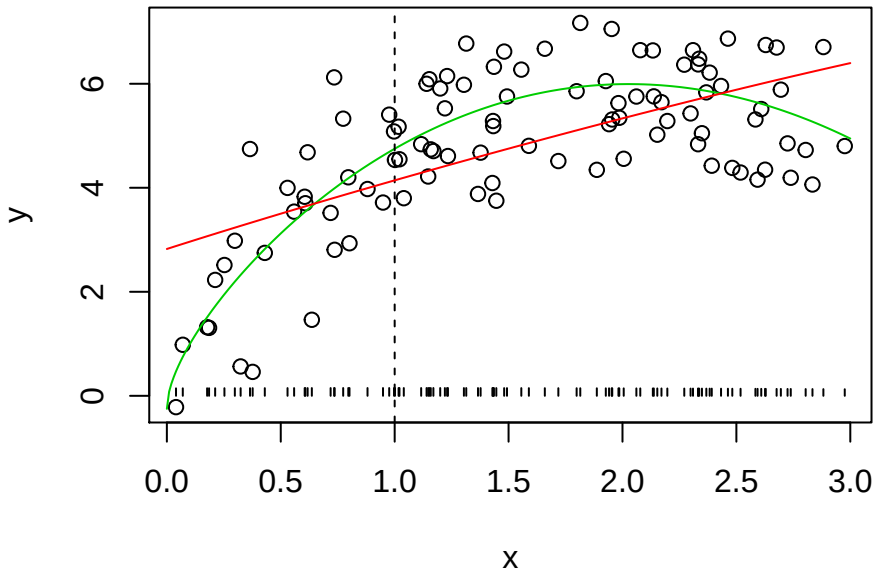
```r
w <- epan(x-xnew,0.5)
points(x,w,type="h")

# Lokale lineare Regression:

xregr <- x-xnew
lmlokal <- lm(y ~ xregr, weights=w)
points(xnew,lmlokal$coef[1],col=2,lwd=2)
lines(xnew+c(-0.5,0.5),lmlokal$coef[1]+ lmlokal$coef[2]*c(-0.5,0.5))
```




```r
#Für eine Kurve Grid von neuen x-Werten; epan(x-xnew,0.5)
xnewseq <- seq(0,3,by=0.01)
fit <- numeric(length(xnewseq))
i <- 1
for (xnew in xnewseq)
  {
    w <- epan(x-xnew,0.5)
    xregr <- x-xnew
    lmlokal <- lm(y ~ xregr, weights=w)
    fit[i] <- lmlokal$coef[1]
    i <- i+1
  }

plot(x,y,xlab="x",ylab="y")
lines(xnewseq,fx(xnewseq),col=3)
lines(xnewseq,fit,col=2)
```




```r
#Für eine Kurve Grid von neuen x-Werten; epan(x-xnew,5)
plot(x,y,xlab="x",ylab="y")
lines(xseq,fx(xseq),col=3)
xnew <- 1
abline(v=xnew,lty=2)
w <- epan(x-xnew,5)
points(x,w,type="h")

xnewseq <- seq(0,3,by=0.01)
fit <- numeric(length(xnewseq))
i <- 1
for (xnew in xnewseq)
{
  w <- epan(x-xnew,5)
  xregr <- x-xnew
  lmlokal <- lm(y ~ xregr, weights=w)
  fit[i] <- lmlokal$coef[1]
  i <- i+1
}

lines(xnewseq,fit,col=2)
```




```r
#Für eine Kurve Grid von neuen x-Werten; epan(x-xnew,0.1)
plot(x,y,xlab="x",ylab="y")
```

```
lines(xseq,fx(xseq),col=3)
xnew <- 1
abline(v=xnew,lty=2)
w <- epan(x-xnew,0.1)
points(x,w,type="h")

xnewseq <- seq(0,3,by=0.01)
fit <- numeric(length(xnewseq))
i <- 1
for (xnew in xnewseq)
{
  w <- epan(x-xnew,0.1)
  xregr <- x-xnew
  lmlokal <- lm(y ~ xregr, weights=w)
  fit[i] <- lmlokal$coef[1]
  i <- i+1
}

lines(xnewseq,fit,col=2)
```