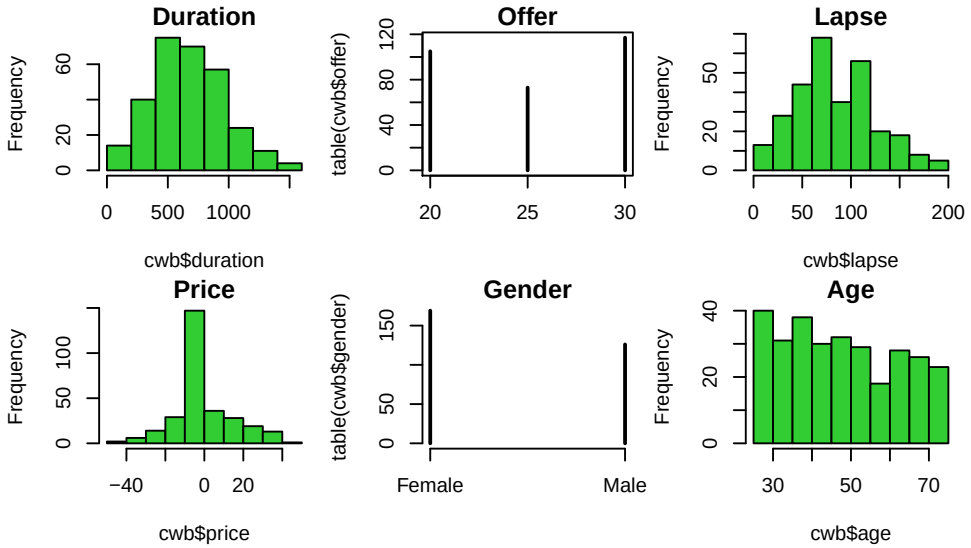
Solution to Series 7

1. **a)** We begin the analysis by plotting histograms and barplots for all variables.

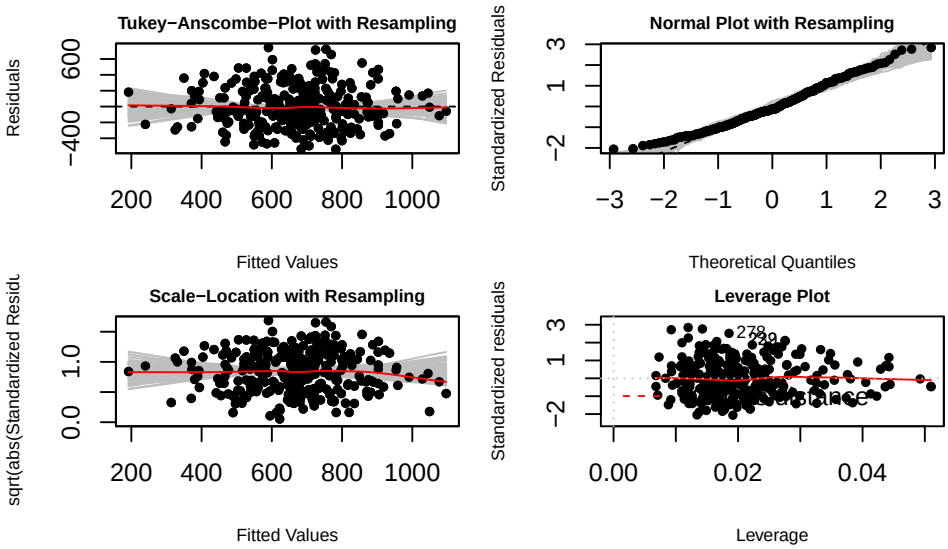
```
> ## load data
> load("CustomerWinBack.rda")
> ## create factor variable for gender
> cwb$gender <- factor(cwb$gender, levels=c(0,1), labels=c("Female", "Male"))
> ## histograms und barplots
> par(mfrow=c(2,3))
> hist(cwb$duration, col="limegreen", main="Duration") ## log ?
> plot(table(cwb$offer), main="Offer")                 ## change to factor variable ?
> hist(cwb$lapse, col="limegreen", main="Lapse")       ## log ?
> hist(cwb$price, col="limegreen", main="Price")
> plot(table(cwb$gender), main="Gender")
> hist(cwb$age, col="limegreen", main="Age")
```



First, we need to create a factor variable for `gender`. The variables `duration` and `lapse` are candidates for a log-transformation. We shall not do these transformations for now. Instead, we fit a first model with all untrasformed variables as predictors and assess the fit.

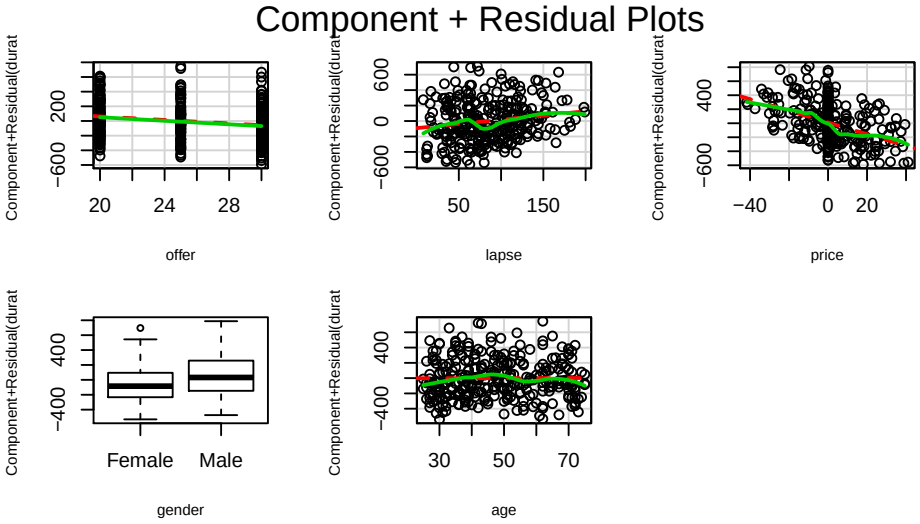
OLS with all variables

```
> ## fit OLS
> fit.ols <- lm(duration ~ offer + lapse + price + gender + age, data=cwb)
> par(mfrow=c(2,2))
> source("../../series6/ex1/resplot.R")
> resplot(fit.ols)
```



The Tukey-Anscombe plot shows a perfect fit and there are no issues with respect to high leverage points or non-constant variance. Merely, the distribution of the residuals is somewhat short-tailed on the left but this deviation is tolerable. So it seems like we have found a well-fitting model for the target variable `duration`. Lastly, we check the partial residual plots.

```
> library(car)
> crPlots(fit.ols, layout=c(2,3), cex.lab = .75)
```



These plots do not suggest a necessary transformation, either. Applying transformations, we notice that the residual plots look worse (not shown here). So our final conclusion is to work with the untransformed variables.

Model chosen by AIC/BIC model selection

```
> ## variable selection with AIC und BIC
> fit.aic <- step(fit.ols)
Start:  AIC=3294.27
duration ~ offer + lapse + price + gender + age

         Df Sum of Sq      RSS    AIC
- age     1       246 20041387 3292.3
<none>                20041141 3294.3
- lapse   1    548224 20589366 3300.2
- offer   1    718009 20759150 3302.6
- gender  1    886259 20927400 3305.0
- price   1   4500960 24542101 3352.0
```

```
Step:  AIC=3292.27
duration ~ offer + lapse + price + gender

         Df Sum of Sq      RSS    AIC
<none>                20041387 3292.3
- lapse   1    552534 20593922 3298.3
- offer   1    733612 20774999 3300.9
- gender  1    888951 20930338 3303.1
- price   1   4503240 24544627 3350.1
```

```
> fit.bic <- step(fit.ols, k=log(nrow(cwb)))
```

```
Start:  AIC=3316.39
duration ~ offer + lapse + price + gender + age

         Df Sum of Sq      RSS    AIC
- age     1       246 20041387 3310.7
<none>                20041141 3316.4
- lapse   1    548224 20589366 3318.7
- offer   1    718009 20759150 3321.1
- gender  1    886259 20927400 3323.5
- price   1   4500960 24542101 3370.5

Step:  AIC=3310.7
duration ~ offer + lapse + price + gender

         Df Sum of Sq      RSS    AIC
<none>                20041387 3310.7
- lapse   1    552534 20593922 3313.0
- offer   1    733612 20774999 3315.6
- gender  1    888951 20930338 3317.8
- price   1   4503240 24544627 3364.8
```

AIC and BIC model selection yield the same model.

Ridge regression

```
> ## ridge regression
> library(MASS)
> fit.ridge.cand <- lm.ridge(duration ~ ., lambda=seq(0,100,0.1), data=cwb)
> select(fit.ridge.cand)
```

```
modified HKB estimator is 9.038797
modified L-W estimator is 9.69619
smallest value of GCV  at 15.1
```

```
> fit.ridge <- lm.ridge(duration ~ ., lambda=15.1, data=cwb)
> coef(fit.ridge)
```

```
                     offer         lapse         price
825.445430780 -11.067127182   1.027530005  -7.920981031
   genderMale           age
106.491845495   0.009569645
```

Lasso regression

```
> ## Lasso
> library(glmnet)
> ## Lasso does not work with factor variables
> xx        <- model.matrix(duration~0+., cwb)[,-4]
> yy        <- cwb$duration
> fit.lasso <- glmnet(xx,yy)
> cvfit     <- cv.glmnet(xx,yy)
> coef(cvfit)
```

```
6 x 1 sparse Matrix of class "dgCMatrix"
                      1
(Intercept) 703.5405553
offer        -2.5377140
lapse         0.2467096
price        -6.4896765
genderMale   39.4691513
age           .
```

The Lasso chooses the same variables as the model selection procedure with AIC and BIC. On the other hand, Ridge regression yields a non-zero coefficient for age.

For the cross validation, we need the following code:

```
> ## cross validation preparation
> pre.ols   <- c()
> pre.aic   <- c()
> pre.bic   <- c()
> pre.rr <- c()
> pre.las <- c()
> folds      <- 5
> sb         <- round(seq(0,nrow(cwb),length=(folds+1)))
```

First, we define an object for each method that will serve to save the respective predictions. To perform 5-fold cross validation, the object sb contains the split points for the samples.

```
> ## cross validation Loop
> for (i in 1:folds)
 {
   ## define training and test datasets
   test    <- (sb[((folds+1)-i)]+1):(sb[((folds+2)-i)])
   train   <- (1:nrow(cwb))[-test]

   ## fit models
   fit.ols <- lm(duration ~ ., data=cwb[train,])
   fit.aic <- step(fit.ols)
   fit.bic <- step(fit.ols, k=log(nrow(cwb[train,])))
   fit.rc  <- lm.ridge(duration ~ ., lambda=seq(0,100,0.1), data=cwb[train,])
   lambda  <- as.numeric(attributes(which.min(fit.rc$GCV))$names)
   fit.rr  <- lm.ridge(duration ~ ., lambda=lambda, data=cwb[train,])
   ## Lasso does not work with factor variables
   xx      <- model.matrix(duration~0+., cwb[train,])[,-4]
   yy      <- cwb$duration[train]
   fit.las <- cv.glmnet(xx,yy)

   ## create predictions
   pre.ols[train] <- predict(fit.ols, newdata=cwb[test,])
   pre.aic[train] <- predict(fit.aic, newdata=cwb[test,])
   pre.bic[train] <- predict(fit.bic, newdata=cwb[test,])
   pre.rr[train]  <- model.matrix(duration~., cwb[test,])%*%coef(fit.rr)
   pre.las[train] <- model.matrix(duration~., cwb[test,])%*%as.numeric(coef(fit.las))
 }
>
```

In each round, the OLS model and the variable selection with AIC and BIC are performed in the respective training data set. In the case of ridge regression, we first determine the optimal value for λ for the respective training data set and we then fit the ridge regression model with this value of λ. This procedure is similar for the Lasso and is already implemented in `cv.glmnet()`. Note that the Lasso cannot handle factor variables. However, since the variable gender has two levels only, we can treat it as a numerical variable. We solve this by deleting the unnecessary fourth column from the design matrix which contains the dummy variable for "female".

In the next step, the predictions are computed on the respective test dataset. For the OLS, AIC,

and BIC models, this is straightforward and achieved by calling `predict()`. For ridge regression and the Lasso, we need to create the design matrix for the test dataset and extract the coefficients from the respective fits in order to perform a matrix multiplication.

Finally, we compute the mean squared prediction error:

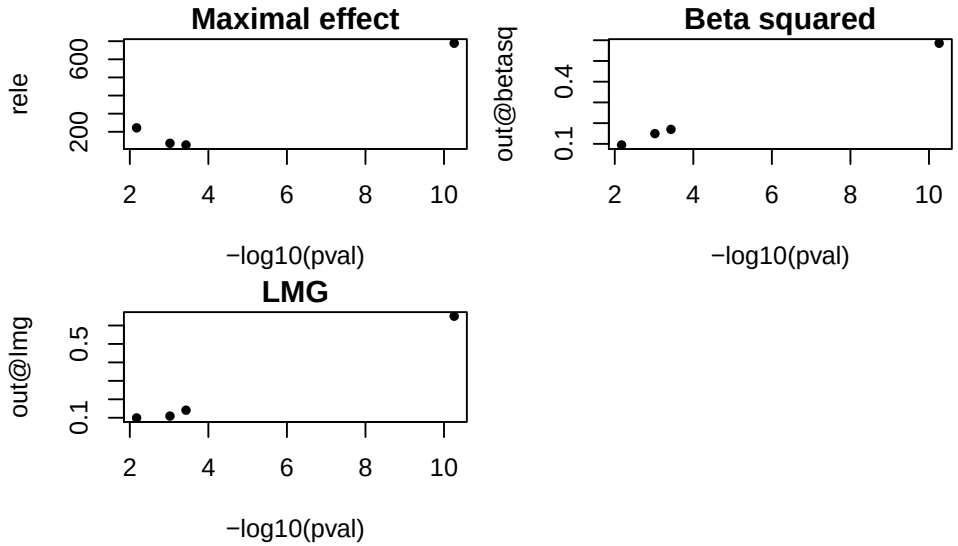
```
> mean((cwb$duration-pre.ols)^2)
[1] 106336.3
> mean((cwb$duration-pre.aic)^2)
[1] 106002
> mean((cwb$duration-pre.bic)^2)
[1] 105201.3
> mean((cwb$duration-pre.rr)^2)
[1] 103577
> mean((cwb$duration-pre.las)^2)
[1] 92889.42
```

By far, the best model is the Lasso. The R-squared of the OLS model is quite low, with a value of 0.24 when using the entire data set. This means that the signal-to-noise ratio is small and the prediction accuracy benefits from the regularization that is achieved by the Lasso.

b)

```
> ## relevance of the predictors
>
> ## use AIC model from above
> pval <- (summary(fit.aic)$coefficients[,4])[-1]
> # negative common logarithm of p-values
> -log10(pval)
     offer      lapse      price genderMale
  3.021323   2.172485  10.258500   3.430056
> par(mfrow=c(2,2))
> ## maximal effect
> mami        <- function(col) max(col)-min(col)
> ranges      <- apply(model.matrix(fit.aic)[,-1],2,mami)
> rele        <- abs(ranges*coef(fit.aic)[-1])
> ranges
     offer      lapse      price genderMale
      10.0      191.0       82.8        1.0
> rele
     offer      lapse      price genderMale
  136.5637   221.7632   688.7541   127.3950
> plot(-log10(pval), rele, pch=20, main="Maximal effect")
> ## beta squared
> library(relaimpo)
> out <- calc.relimp(fit.aic, type="betasq", rela=TRUE); out
Response variable: duration
Total response variance: 88033.6
Analysis based on 236 observations

4 Regressors:
offer lapse price gender
Proportion of variance explained by model: 24.31%
Metrics are normalized to sum to 100% (rela=TRUE).

Relative importance metrics:

           betasq
offer  0.14956402
lapse  0.09485298
price  0.58558849
gender 0.16999451
```

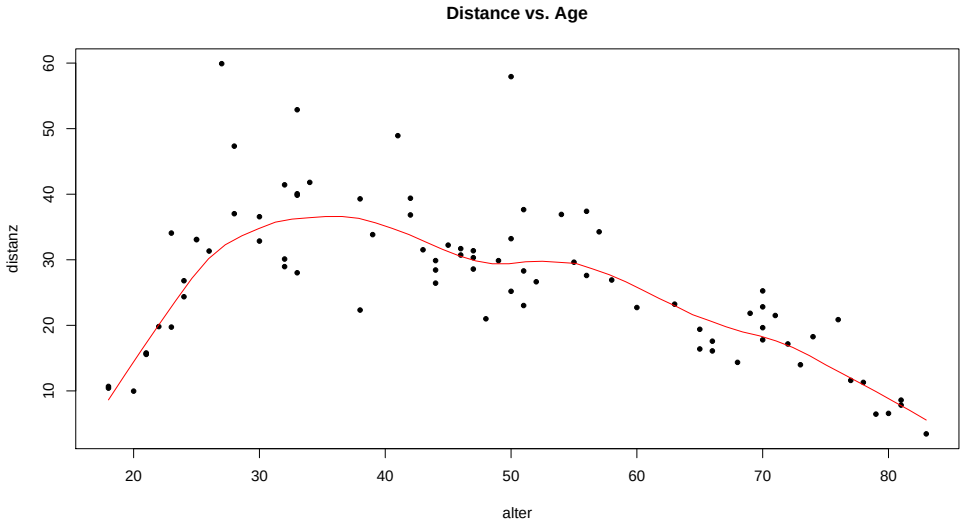
```
> plot(-log10(pval), out@betasq, pch=20, main="Beta squared")
> ## LMG criterion
> out <- calc.relimp(fit.aic, type="lmg", rela=TRUE); out
Response variable: duration
Total response variance: 88033.6
Analysis based on 236 observations

4 Regressors:
offer lapse price gender
Proportion of variance explained by model: 24.31%
Metrics are normalized to sum to 100% (rela=TRUE).

Relative importance metrics:

              lmg
offer  0.10935191
lapse  0.09919038
price  0.65043583
gender 0.14102188

Average coefficients for different model sizes:

              1X        2Xs        3Xs        4Xs
offer  -8.847884 -10.477026 -12.082349 -13.656369
lapse   1.130171   1.128236   1.137795   1.161064
price  -8.450362  -8.415531  -8.373298  -8.318286
gender 97.503824 106.490327 116.622339 127.395003
> plot(-log10(pval), out@lmg, pch=20, main="LMG")
```



The model selection using the AIC critetion removes the variable age from the model. For this model, we study the relevance of the predictors. To visualize the results, we plot the values of the criteria against $-\log_{10}(\cdot)$ of the p-value. So the variables in the bottom left corner of the plot are neither relevant nor significant. On the other hand, the variables in the upper right corner are relevant as well as significant. We see that `price` plays a crucial role. The other three variables are not as important in comparison.

2. **a)**

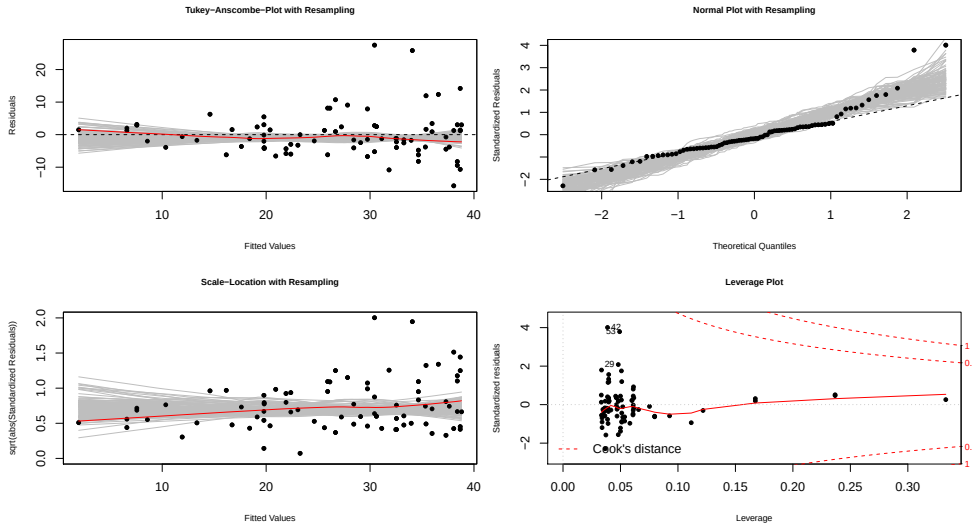
```
> ## load data
> load("autodistanz.rda")
> ## scatter plot and LOESS smoother
> par(mfrow=c(1,1))
> plot(distanz ~ alter, data=autodistanz, pch=20, main="Distance vs. Age")
> lines(loess.smooth(autodistanz$alter, autodistanz$distanz, span=0.3), col="red")
```



The default smoothing parameter span=2/3 is too large since the resulting smoother does not fit the data well. Therefore, we need to choose a smaller value. At the margins, a good fit is achieved when span=0.3. The disadvantage is that this value leads to a "too smooth" fit in the center. This stems from the fact that a LOESS smoother with a fixed span cannot handle a relation well which shows different curvature at different places in predictor space. Splines are less problematic in this respect.

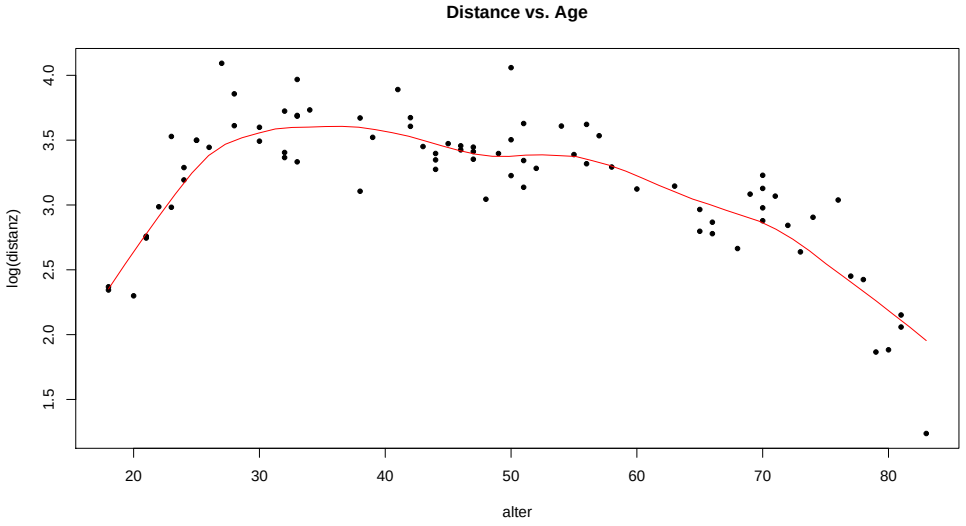
b) The first question we need to answer is what degree the polynomial should have. Since the increase and the decrease in the relation between predictor and response are not symmetric, the order has to be larger than two. Considering the number of inflection points we see that the degree should be at least four.

```
> ## fit polynomial
> fit <- lm(distanz ~ poly(alter,4), data=autodistanz)
> summary(fit)

Call:
lm(formula = distanz ~ poly(alter, 4), data = autodistanz)

Residuals:
    Min      1Q  Median      3Q     Max
-15.722  -4.010  -1.208   2.452  27.505

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      26.8220     0.7726  34.716  < 2e-16 ***
poly(alter, 4)1 -45.9776     6.9963  -6.572 5.32e-09 ***
poly(alter, 4)2 -60.2910     6.9963  -8.618 6.55e-13 ***
poly(alter, 4)3  27.2452     6.9963   3.894 0.000208 ***
poly(alter, 4)4 -25.4200     6.9963  -3.633 0.000502 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.996 on 77 degrees of freedom
Multiple R-squared:  0.6544,        Adjusted R-squared:  0.6365
F-statistic: 36.45 on 4 and 77 DF,  p-value: < 2.2e-16

> par(mfrow=c(2,2))
> source("../../series6/ex1/resplot.R")
> resplot(fit)
```



All terms in the model are significant. Increasing the degree of the polynomial further, this is no longer the case. Therefore, we can assume that a polynomial of degree 4 is sufficient. Looking at the residual plots, we see that the variance of the error is not constant. Therefore, we also see a deviation in the Normal plot. The larger scatter for larger values of `distance` is also visible in the scatter plot. Thus, to model the traveled distance we need to log-transform the target variable. Afterwards we need to redo the above steps and re-assess what order we need to choose for the polynomial.

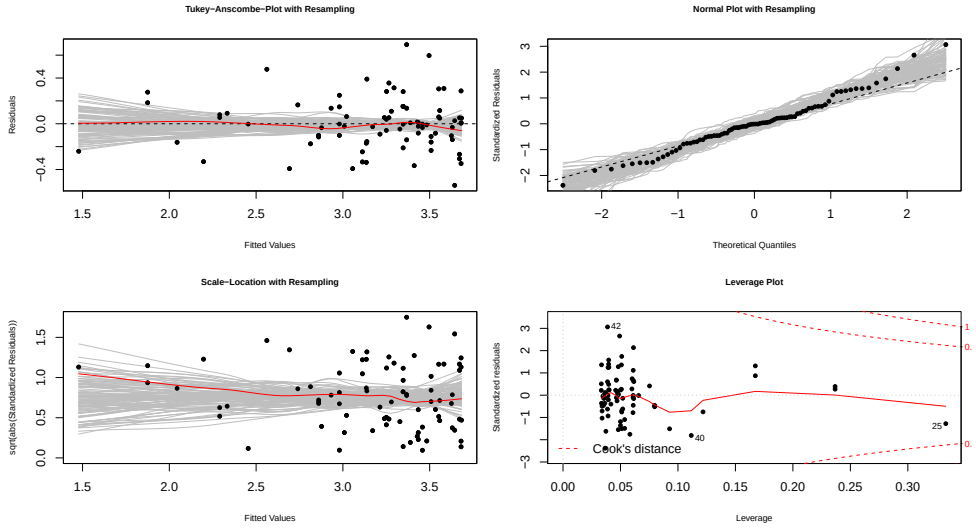
```
> par(mfrow=c(1,1))
> plot(log(distanz) ~ alter, data=autodistanz, pch=20, main="Distance vs. Age")
> lines(loess.smooth(autodistanz$alter, log(autodistanz$distanz), span=0.3), col="red")
```




```
> ## fit polynomial
> fit <- lm(log(distanz) ~ poly(alter,4), data=autodistanz)
> summary(fit)

Call:
lm(formula = log(distanz) ~ poly(alter, 4), data = autodistanz)

Residuals:
     Min       1Q   Median       3Q      Max
-0.53812 -0.13257 -0.00258  0.11400  0.69204

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)     3.17490    0.02544 124.822  < 2e-16 ***
poly(alter, 4)1 -2.20566    0.23033  -9.576 9.35e-15 ***
```

```
poly(alter, 4)2 -3.27382    0.23033 -14.214  < 2e-16 ***
poly(alter, 4)3  0.41314    0.23033   1.794   0.0768 .
poly(alter, 4)4 -1.58031    0.23033  -6.861 1.52e-09 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2303 on 77 degrees of freedom
Multiple R-squared:  0.8171,        Adjusted R-squared:  0.8076
F-statistic: 86.01 on 4 and 77 DF,  p-value: < 2.2e-16
```

```
> par(mfrow=c(2,2))
> resplot(fit)
```



We see that a polynomial of fourth order still yields a good fit and the model assumptions seem to be fulfilled to a sufficient degree.

c) In order to fit a B-spline model with the same degree of complexity as the polynomial, at most five parameters can be estimated, i.e. this is the number of basis functions that can be used. Thus, in addition to the minimum and maximum we can only place one knot. We decide to set it at a value of 35 in order to account for the fact that the curvature of the regression function is larger at a smaller age than at higher values of age.

```
> ## generate and visualize the basis
> par(mfrow = c(1,1))
> library(splines)
> range(autodistanz$alter)
```

```
[1] 18 83
```

```
> xx <- sort(autodistanz$alter)
> yy <- autodistanz$distanz[order(autodistanz$alter)]
> kn <- c(18,18,18,18,35,83,83,83,83)
> bx <- splineDesign (kn, xx)
```

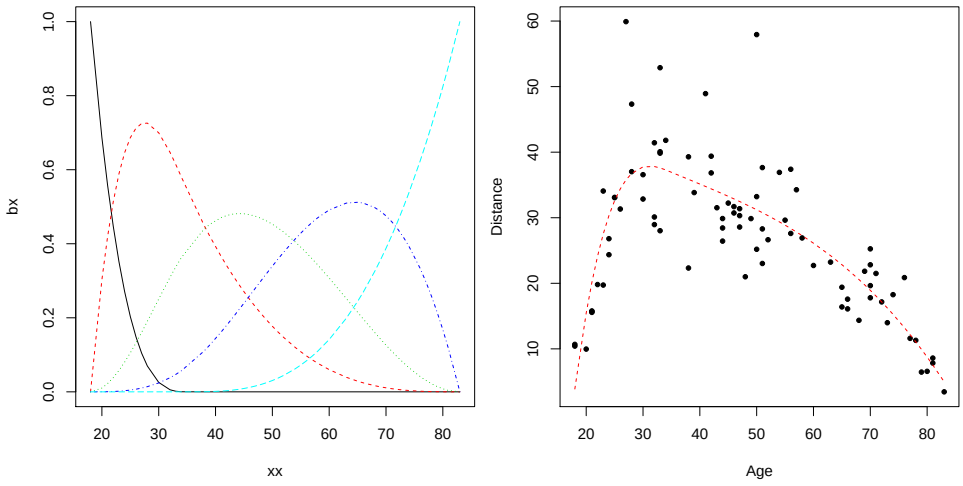
We now perform the regression with these basis functions (cf. plot below). Note that an intercept cannot be used since this would make the design singular. The resulting fit is shown below.

```
> fit.spline <- lm (yy ~ 0+bx)
> summary(fit.spline)
```

```
Call:
lm(formula = yy ~ 0 + bx)

Residuals:
    Min      1Q  Median      3Q     Max
-13.527  -4.216  -1.117   2.870  26.718
```

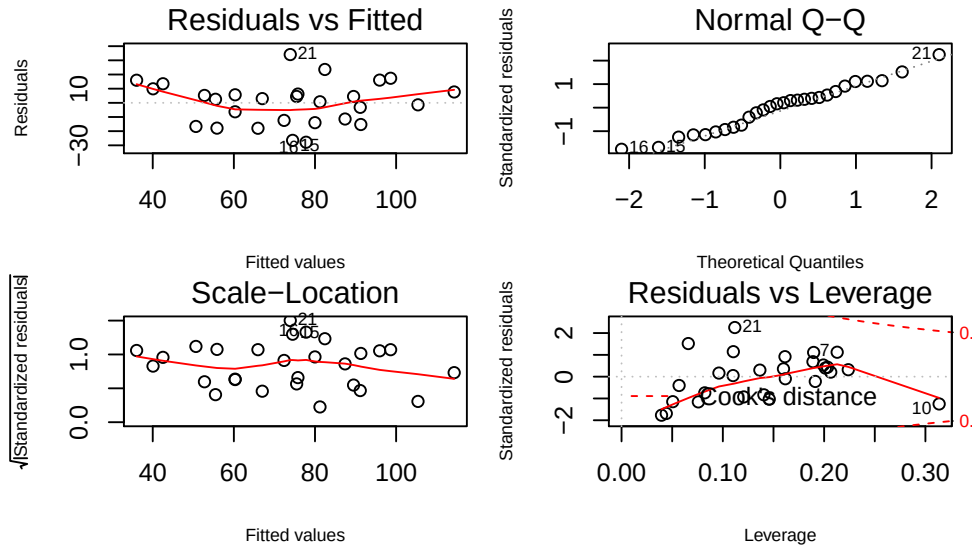
```
Coefficients:
    Estimate Std. Error t value Pr(>|t|)
bx1    3.882      3.858   1.006    0.318
bx2   41.108      3.455  11.897  < 2e-16 ***
bx3   32.970      5.956   5.536 4.13e-07 ***
bx4   26.075      5.728   4.552 1.95e-05 ***
bx5    4.977      3.539   1.406    0.164
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.894 on 77 degrees of freedom
Multiple R-squared:  0.9477,        Adjusted R-squared:  0.9443
F-statistic: 278.8 on 5 and 77 DF,  p-value: < 2.2e-16
```

```
> par(mfrow = c(1,2))
> matplot(xx, bx, type="l")
> matplot(xx, cbind (yy, fit.spline$fitted), type="pl", pch=20, xlab="Age", ylab="Distance")
```



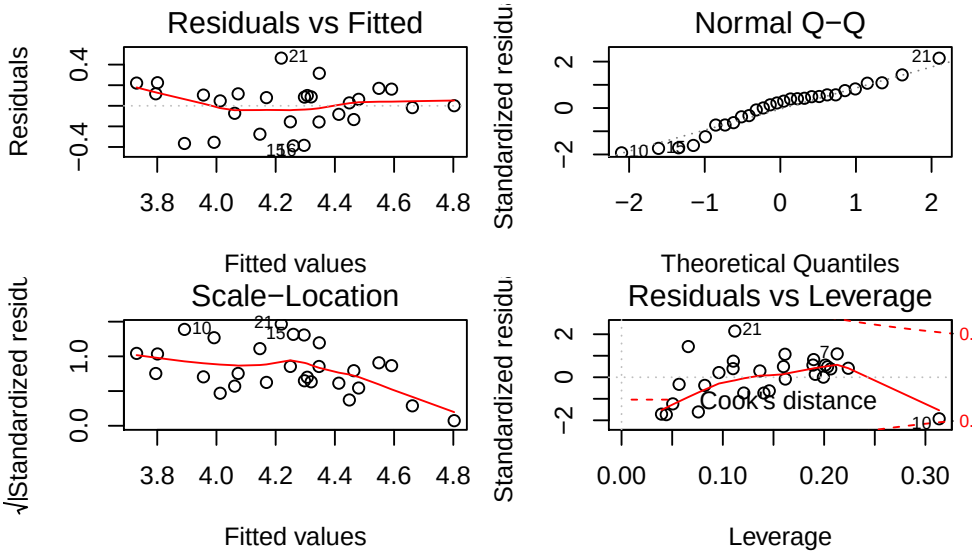
3. **a)** We start with the following model:

```
> ## load data
> load("no2basel.rda")
> ## multiple regression model
> fit1 <- lm(NO2 ~ Tag + Temp + Wind, data=no2basel)
> par(mfrow = c(2,2))
> plot(fit1, cex.lab = 0.75)
```



The Tukey-Anscombe plot shows indications of a systematic error, so the current model is not acceptable. The target variable NO2 is suitable for a log-transformation, so we will apply this transformation while leaving the predictors unchanged:

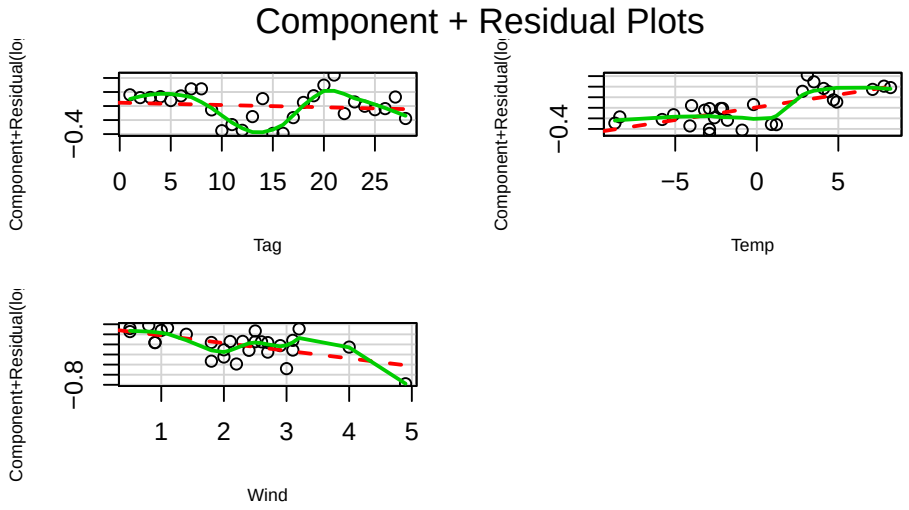
```
> ## with transformation
> fit2 <- lm(log(NO2) ~ Tag + Temp + Wind, data=no2basel)
> par(mfrow=c(2,2))
> plot(fit2)
```



The residual plots improved but they still do not look satisfactory. The deviation in the Tukey-Anscombe plot is less accentuated but is still present. We conclude that we have not yet found a valid and correct model for this data set as the log-transformation could not remove the systematic error entirely.

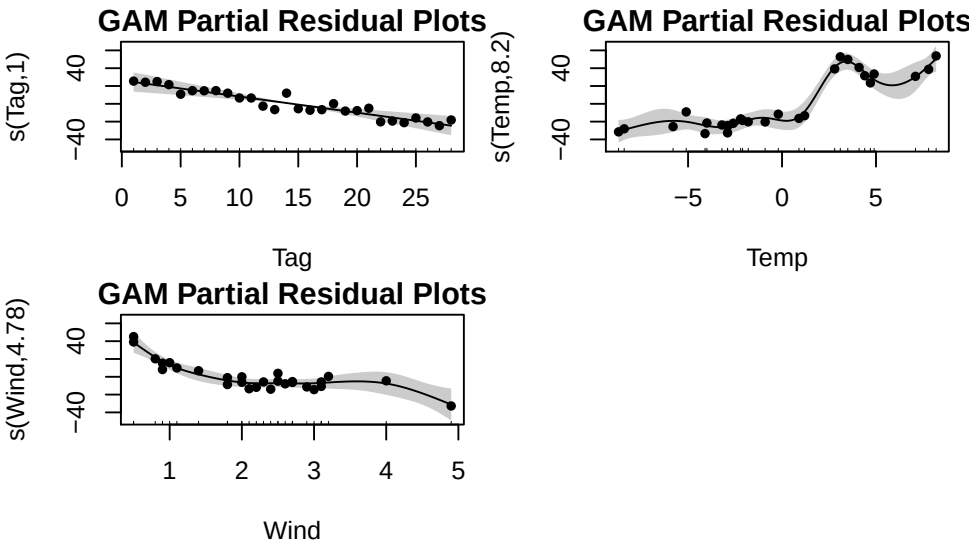
So the systematic error stems from the fact that the predictors do not influence the response linearly. Instead, the relation is more complex so that we need to apply a method that can account for this. The partial residual plots yield some insights with respect to the predictor-response relations:

```
> ## partial residual plots
> library(car)
> crPlots(fit2, cex.lab = .75)
```

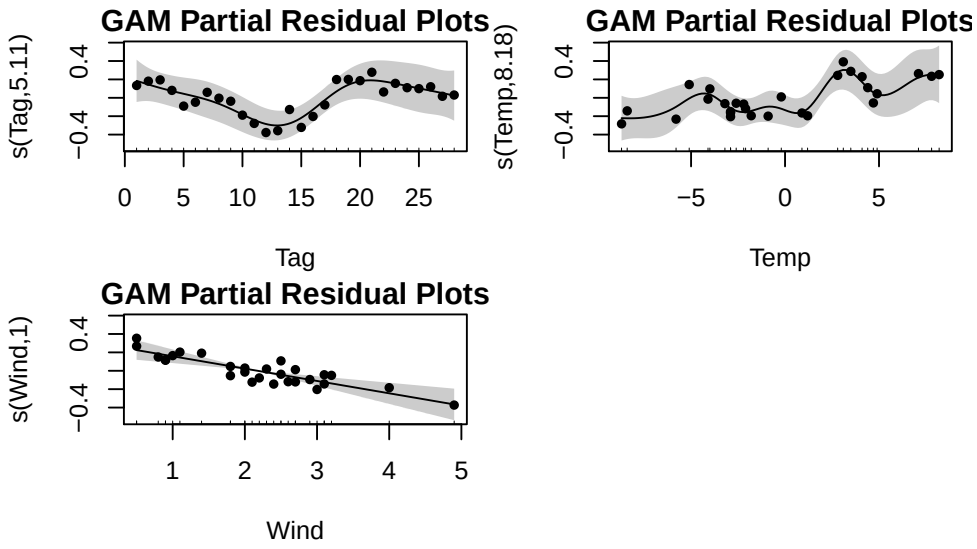


b) Again, we need to decide whether the response variable should be log-transformed. If the relation between the response and the predictors & the error term is multiplicative, then a transformation is mandatory. For the predictors we do not need to consider transformations as these can be estimated by the GAM itself in a non-parametric way. We shall compute both possible models, i.e. the model with the untransformed response and the model with the log-transformed response. The partial residual plots show a large difference:

```
> ## GAM
> ## load package
> library(mgcv)
> ## fit models
> fit3 <- gam(NO2 ~ s(Tag) + s(Temp) + s(Wind), data=no2basel)
> fit4 <- gam(log(NO2) ~ s(Tag) + s(Temp) + s(Wind), data=no2basel)
> ## partial residual plots
> par(mfrow=c(2,2))
> plot(fit3, shade=TRUE, residuals=TRUE, pch=20, main="GAM Partial Residual Plots")
```

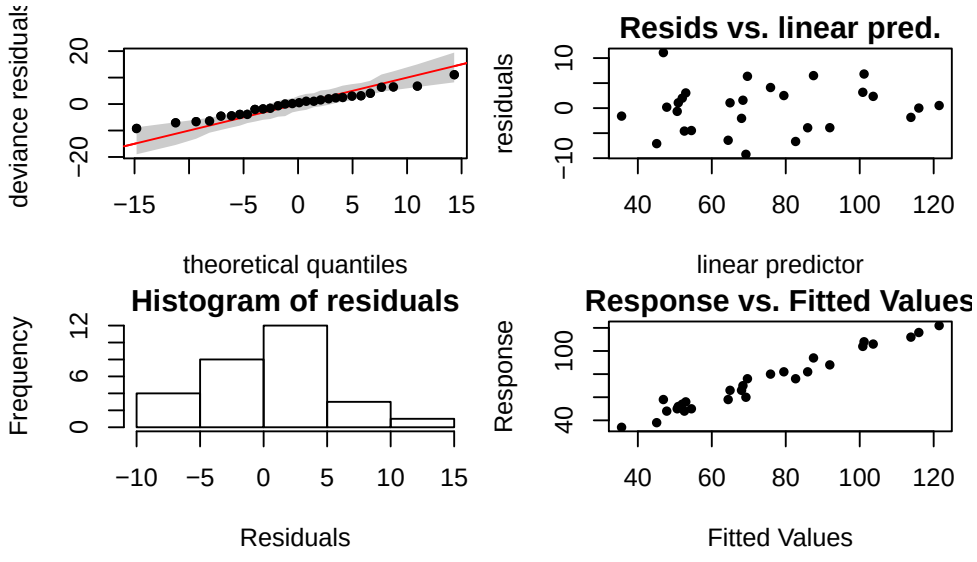



```
> par(mfrow=c(2,2))
> plot(fit4, shade=TRUE, residuals=TRUE, pch=20, main="GAM Partial Residual Plots")
```



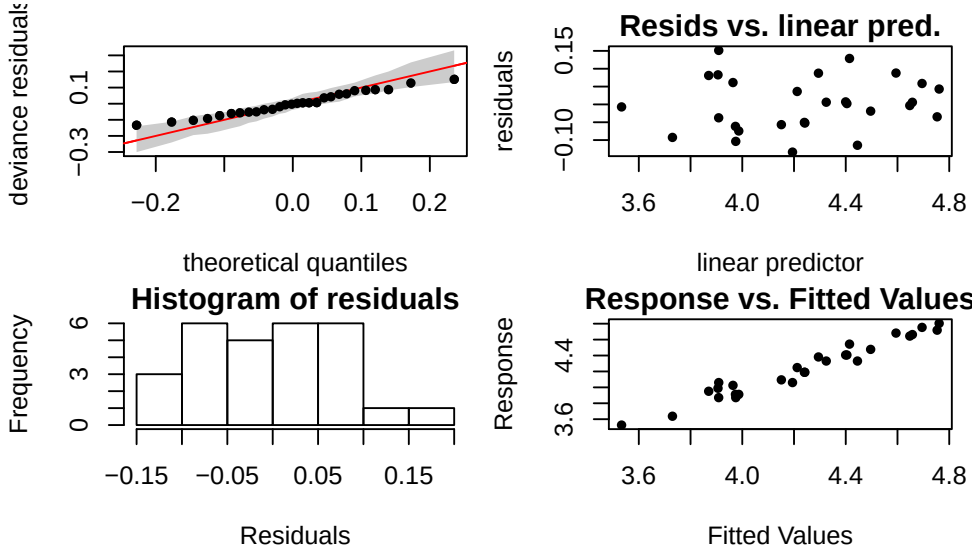
We use the GAM model diagnostic tools to assess which model is better suited.

```
> ##  GAM model diagnostic tools
> gam.check(fit3, pch=20, rep=100)

Method: GCV   Optimizer: magic
Smoothing parameter selection converged after 18 iterations.
The RMS GCV score gradiant at convergence was 7.202621e-06 .
The Hessian was positive definite.
The estimated model rank was 28 (maximum possible: 28)
Model rank =  28 / 28

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(Tag)  9.00 1.00    1.12    0.67
s(Temp) 9.00 8.20    1.44    0.99
s(Wind) 9.00 4.78    1.13    0.68
```




```
> gam.check(fit4, pch=20, rep=100)

Method: GCV   Optimizer: magic
Smoothing parameter selection converged after 23 iterations.
The RMS GCV score gradiant at convergence was 1.277338e-07 .
The Hessian was positive definite.
The estimated model rank was 28 (maximum possible: 28)
```

```
Model rank =  28 / 28

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(Tag)  9.00 5.11    1.03    0.48
s(Temp) 9.00 8.18    1.38    0.96
s(Wind) 9.00 1.00    1.09    0.64
```

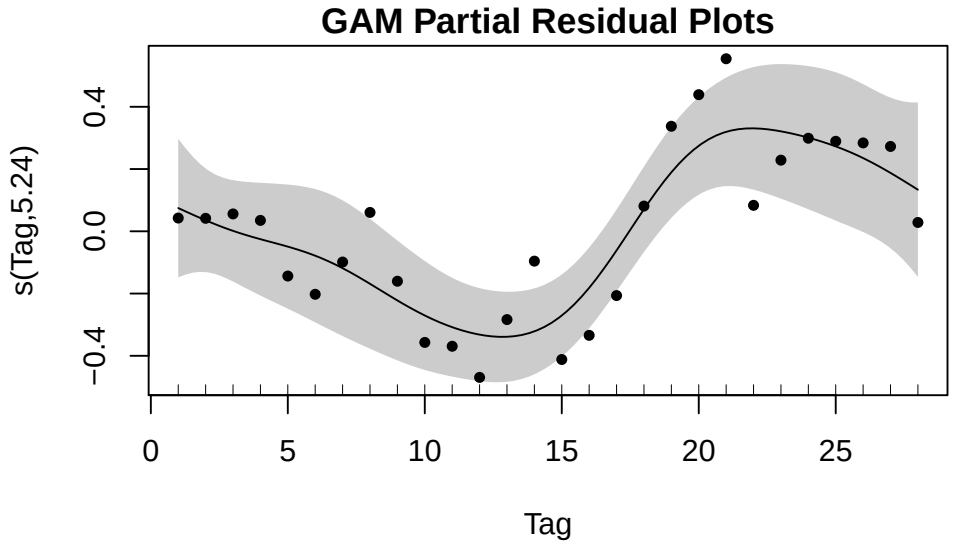


Neither of the two models shows a systematic error. With respect to constant variance and the distribution of the residuals, the differences are not large either. Thus, both models fit well. Looking at the partial residual plots, the second model with the log-transformation seems more suitable. When considering the relation between the temperature predictor and the response variable, a linear function would just fit into the grey confidence band. Additionally, this linear relation is plausible if we take into account the interpretation of the model. Therefore, we suggest a model of the form

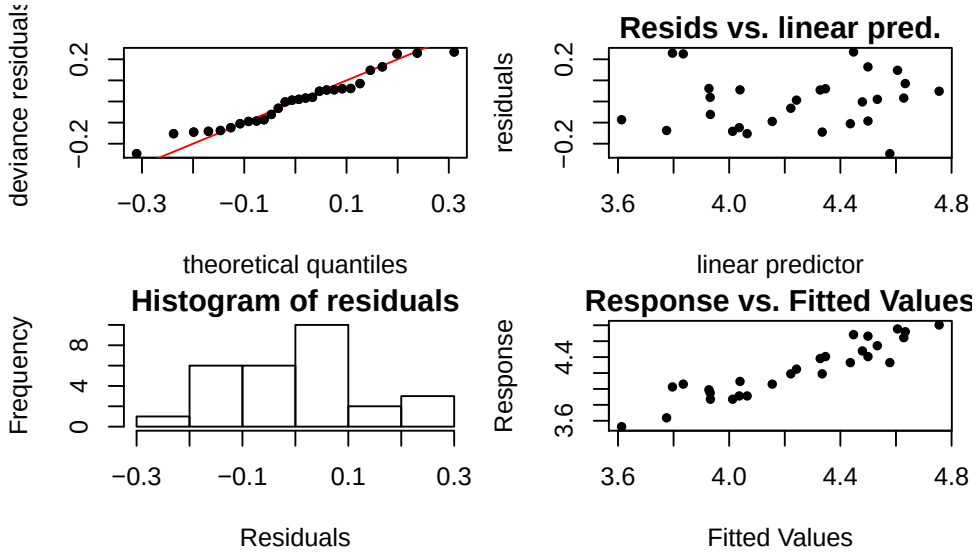
`fit5 <- gam(log(NO2) ~ s(Tag) + Temp + Wind, data=...)`.

As we see below, the model fits well:

```
> ## alternative model
> fit5 <- gam(log(NO2) ~ s(Tag) + Temp + Wind, data=no2basel)
> par(mfrow=c(1,1))
> plot(fit5, shade=TRUE, residuals=TRUE, pch=20, main="GAM Partial Residual Plots")
```



```
> par(mfrow=c(2,2))
> gam.check(fit5, pch=20, shade=TRUE)

Method: GCV   Optimizer: magic
Smoothing parameter selection converged after 5 iterations.
The RMS GCV score gradiant at convergence was 1.405851e-05 .
The Hessian was positive definite.
The estimated model rank was 12 (maximum possible: 12)
Model rank =  12 / 12

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(Tag) 9.000 5.242   0.843    0.15
```



c) A few aspects concerning this questions (e.g. residual and partial residual plots) have been discussed above. In addition, there are formal tests that compare the different models. It is important to note that only those models that have the same form of the target variable can be compared, i.e. transformed or untransformed. We compare three models with log-transformed response:

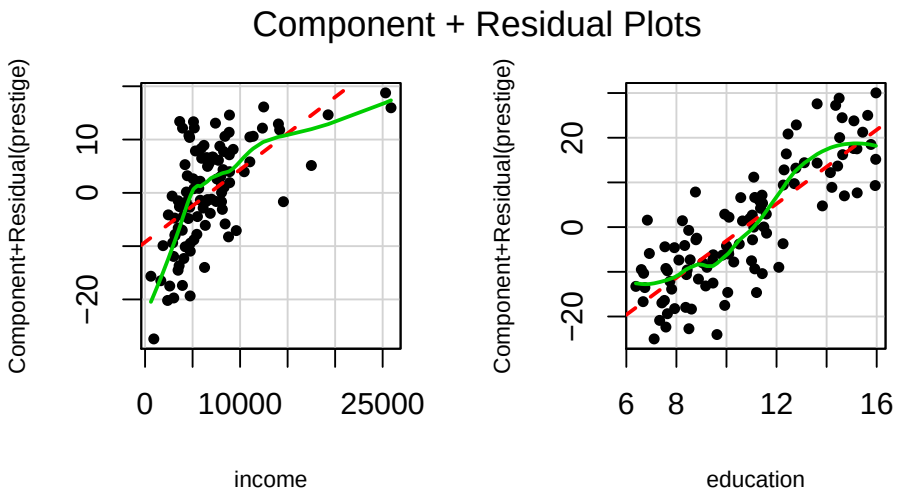
```
> ## model comparison
> fit.ols <- gam(log(NO2) ~ Tag + Temp + Wind, data=no2basel)
> anova(fit.ols, fit5, test="Chisq")

Analysis of Deviance Table

Model 1: log(NO2) ~ Tag + Temp + Wind
Model 2: log(NO2) ~ s(Tag) + Temp + Wind
  Resid. Df Resid. Dev     Df Deviance  Pr(>Chi)
1    24.000    1.25730
2    19.758    0.43186 4.2423  0.82544 1.703e-07 ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> anova(fit5, fit4, test="Chisq")

Analysis of Deviance Table

Model 1: log(NO2) ~ s(Tag) + Temp + Wind
Model 2: log(NO2) ~ s(Tag) + s(Temp) + s(Wind)
  Resid. Df Resid. Dev     Df Deviance Pr(>Chi)
1    19.758    0.43186
2    12.717    0.14995 7.0406  0.28191 0.001218 **
---
```

```
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The last model from b) (`fit5`) is significantly better than the OLS model. `fit4` (containing smooth components only) turned out to be significantly better than `fit5`.

4. a)

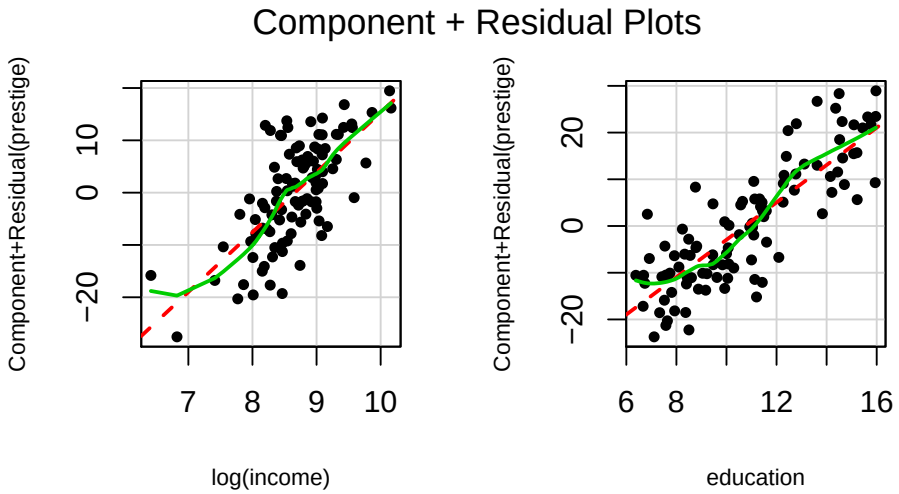
```
> ## load data
> library(car)
> data(Prestige)
> ## fit model
> fit00 <- lm(prestige ~ income + education, data=Prestige)
> summary(fit00)

Call:
lm(formula = prestige ~ income + education, data = Prestige)

Residuals:
     Min       1Q   Median       3Q      Max
-19.4040  -5.3308   0.0154   4.9803  17.6889

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -6.8477787  3.2189771  -2.127   0.0359 *
income       0.0013612  0.0002242   6.071 2.36e-08 ***
education    4.1374444  0.3489120  11.858  < 2e-16 ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.81 on 99 degrees of freedom
Multiple R-squared:  0.798,        Adjusted R-squared:  0.7939
F-statistic: 195.6 on 2 and 99 DF,  p-value: < 2.2e-16

> crPlots(fit00, pch=20, cex.lab = 0.75)
```




```
> ## fit model
> fit0 <- lm(prestige ~ log(income) + education, data=Prestige)
> summary(fit0)

Call:
lm(formula = prestige ~ log(income) + education, data = Prestige)

Residuals:
     Min       1Q   Median       3Q      Max
```

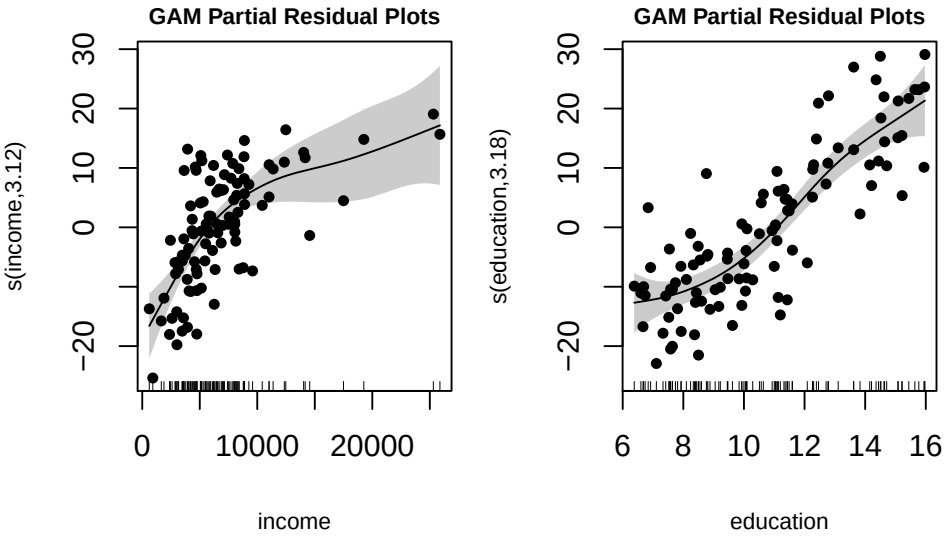
```
-17.0346  -4.5657  -0.1857   4.0577  18.1270

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -95.1940    10.9979  -8.656 9.27e-14 ***
log(income)  11.4375     1.4371   7.959 2.94e-12 ***
education     4.0020     0.3115  12.846  < 2e-16 ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.145 on 99 degrees of freedom
Multiple R-squared:  0.831,        Adjusted R-squared:  0.8275
F-statistic: 243.3 on 2 and 99 DF,  p-value: < 2.2e-16
```

```
> crPlots(fit0, pch=20, cex.lab = 0.75)
```

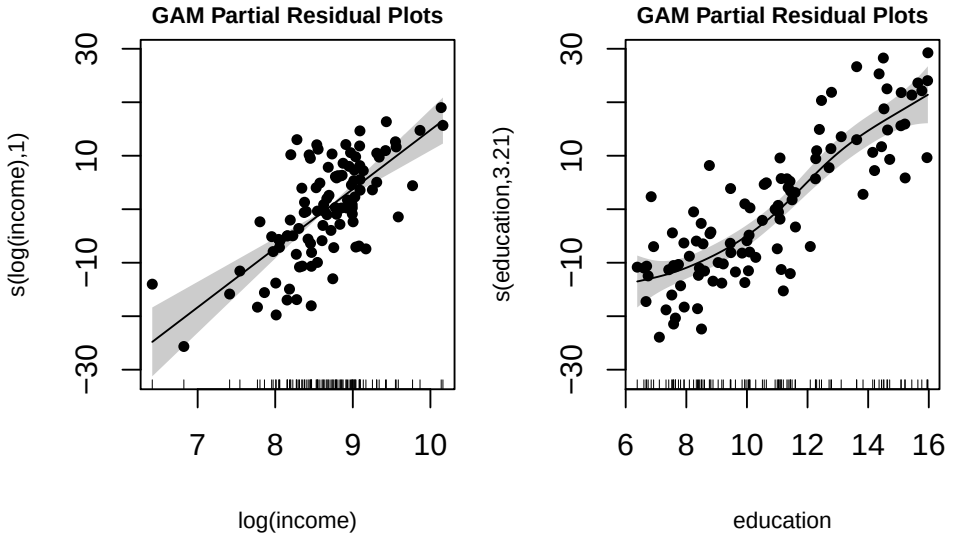


When analyzing the partial residual plots corresponding to the OLS fit, we see that the plots look fine after log-transforming the variable `income`. For `education`, there is an inhomegeneity as there seem to be two groups, namely, one with more than twelve years of education and one with less.

```
> require(mgcv)
> par(mfrow=c(1,2))
> fit11 <- gam(prestige ~ s(income) + s(education), data=Prestige)
> plot(fit11, shade=TRUE, residuals=TRUE, pch=20, main="GAM Partial Residual Plots",
      cex.lab = 0.75, cex.main =0.75)
```

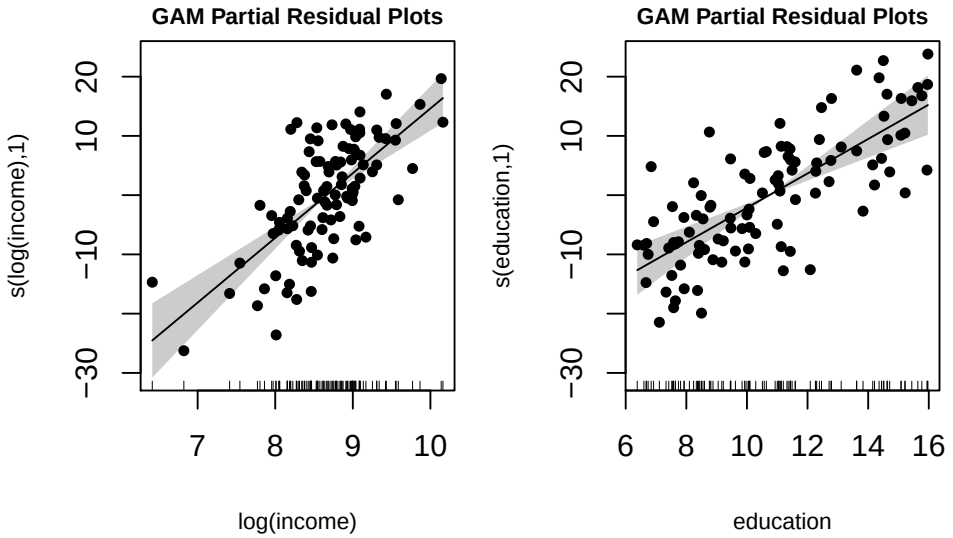


```
> require(mgcv)
> par(mfrow=c(1,2))
> fit1 <- gam(prestige ~ s(log(income)) + s(education), data=Prestige)
> plot(fit1, shade=TRUE, residuals=TRUE, pch=20,
      main="GAM Partial Residual Plots", cex.lab = 0.75, cex.main =0.75)
```



When fitting a GAM with the log-transformed `income` variable, a linear function is chosen. This also indicates that the log-transformation is a good choice in this case. When a variable can be brought into a suitable form by a transformation, then this is benefitial when fitting a GAM as consequently fewer degrees of freedom are needed. For `education`, a smooth component is fitted to accomodate the jump at twelve years of education.

b)
```
> ## add factor variable
> Prestige$uni <- factor(Prestige$education>12, labels=c("nein", "ja"))
> fit2 <- gam(prestige ~ s(log(income)) + s(education) + uni, data=Prestige)
> summary(fit2)

Family: gaussian
Link function: identity

Formula:
prestige ~ s(log(income)) + s(education) + uni

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   44.305      1.096  40.418  < 2e-16 ***
unija          8.058      2.737   2.944  0.00404 **
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
               edf Ref.df     F  p-value
s(log(income))   1      1 61.25 1.76e-12 ***
s(education)     1      1 36.87 1.84e-08 ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =   0.84   Deviance explained = 84.5%
GCV =  49.31  Scale est. = 47.376    n = 102
> par(mfrow=c(1,2))
> plot(fit2, shade=TRUE, residuals=TRUE, pch=20, main="GAM Partial Residual Plots",
      cex.lab = 0.75, cex.main =0.75)
```



As the partial residual plots show, linear functions are fitted for both numerical predictors. Note the effect of the factor variable: If the years of education is larger than twelve, prestige increases on average by 8.058 units. Of course this aspect is not visible in the partial residual plot as it only shows the relation when the other predictors have already been accounted for.

The last model is preferable, i.e. the one with the factor variable and the log-transformed `income` variable. This model only needs four degrees of freedom. When leaving out the transformation as well as the factor variable and using a more complex GAM model instead (one where the predictors are assumed to be smooth), 7.3 degrees of freedom are spent. These are significantly more degrees of freedom while the fit is not improved.

```
> ## model with factor variable and the log-transformed income variable
> fit2

Family: gaussian
Link function: identity

Formula:
prestige ~ s(log(income)) + s(education) + uni

Estimated degrees of freedom:
1 1  total = 4

GCV score: 49.31021

> ## model without factor variable, untransformed income variable and smooth predictors
> fit11

Family: gaussian
Link function: identity

Formula:
prestige ~ s(income) + s(education)

Estimated degrees of freedom:
3.12 3.18  total = 7.3

GCV score: 52.1428
```