# NLNP Praktikum 3

*Robin Baudisch, Merlin Kopfmann, Maximilian Neudert*

## Inhaltsverzeichnis

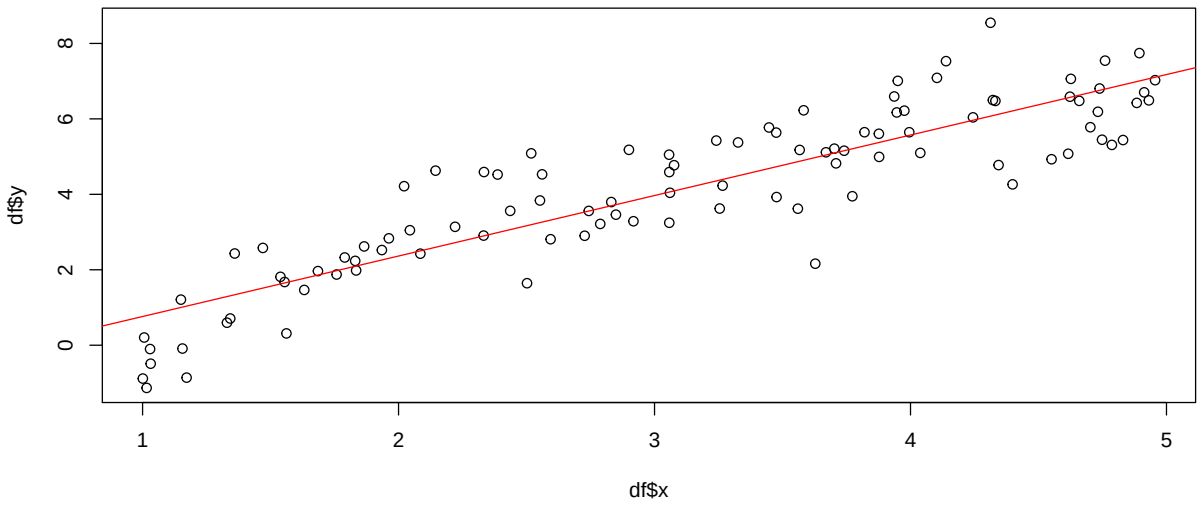
## A1

### a)

```
gendata = function(x, beta) {
    return(beta * log(x) + rnorm(x))
}
x <- runif(100, 1, 5)
y <- gendata(x, 4)
df = data.frame(x, y)
```

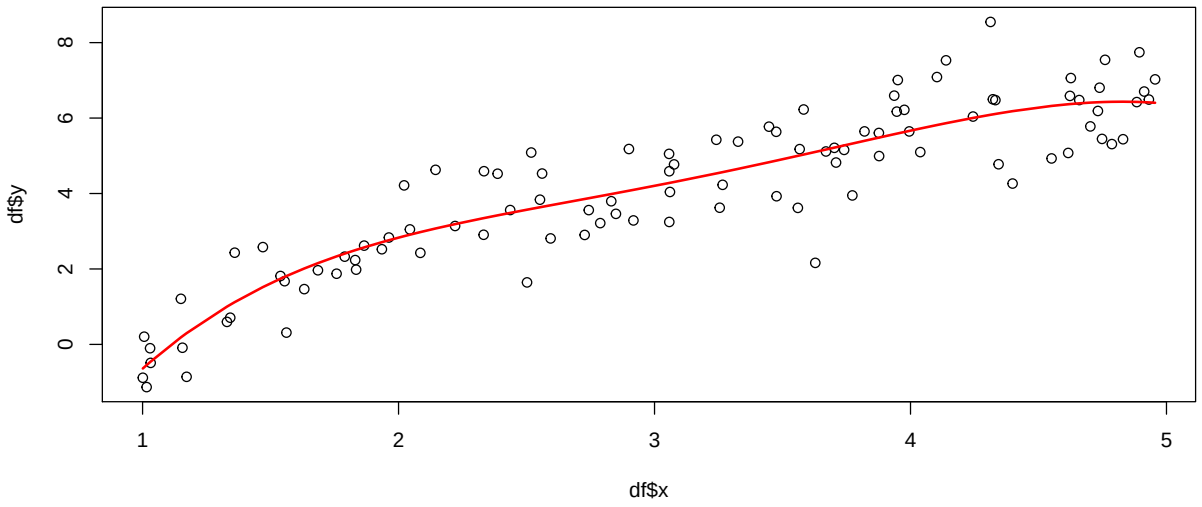
### b)

```
lmlinear = lm(y ~ x, data = df)
fdachlin <- as.vector(lmlinear$coefficients %*% c(1, 3))
plot(df$x, df$y)
abline(lmlinear, col = "red")
```




Hier sehen wir die generierten Punkte und die lineare Regression und wir erhalten $\hat{f}(3) = 3.77$.

```
lmpoly <- lm(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5))
fdachpoly <- as.vector(lmpoly$coefficients %*% c(1, 3, 3^2, 3^3, 3^4, 3^5))
plot(df$x, df$y)
pts = predict(lmpoly)
points(df$x[order(x)], pts[order(x)], type = "l", col = "red", lwd = 2)
```




Hier sehen wir die generierten Punkte und die polynomiale Regression mit Grad 5 und wir erhalten $\hat{f}(3) = 4.10$.

**c)**

```
y2 <- gendata(3, 4)

quadvolin <- (y2 - fdachlin)^2
quadvopol <- (y2 - fdachpoly)^2
```

Sei $\Delta = (y - \hat{f}(x))^2$. Für die lineare Regression erhalten wir $\Delta = 0.28$ und für die polynomiale Regression erhalten wir $\Delta = 0.04$.

**d)**

```
alllin <- matrix(NA, nrow = 10000, ncol = 6)
colnames(alllin) <- c("fdach", "y", "quadVor", "EBias", "VarFDach", "EQuadVor")
allpol <- matrix(NA, nrow = 10000, ncol = 6)
colnames(allpol) <- c("fdach", "y", "quadVor", "EBias", "VarFDach", "EQuadVor")
for (i in 1:10000) {
    x <- runif(100, 1, 5)
    residuen <- rnorm(100, 0, 1)
    b <- 4
    y <- b * log(x) + residuen
    lmlinear <- lm(y ~ x)
    lmpoly <- lm(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5))
    alllin[i, 1] <- lmlinear$coefficients %*% c(1, 3)
    allpol[i, 1] <- lmpoly$coefficients %*% c(1, 3, 3^2, 3^3, 3^4, 3^5)

    alllin[i, 2] <- b * log(3) + rnorm(1)
    allpol[i, 2] <- alllin[i, 2]
```

```
    alllin[i, 3] <- (alllin[i, 2] - alllin[i, 1])^2
    allpol[i, 3] <- (allpol[i, 2] - allpol[i, 1])^2
}
alllin[, 4] <- mean(alllin[, 1]) - 4 * log(3)
allpol[, 4] <- mean(allpol[, 1]) - 4 * log(3)

alllin[, 5] <- var(alllin[, 1])
allpol[, 5] <- var(allpol[, 1])


x3 <- rep(3, 10000)
y3 <- gendata(x3, 4)
alllin[, 6] <- mean((y3 - alllin[, 1])^2)
allpol[, 6] <- mean((y3 - allpol[, 1])^2)
```

**e)**

Für die lineare Regression erhalten wir:

```
head(alllin[, 4:6], 1)
```

```
          EBias   VarFDach EQuadVor
[1,] -0.3440677 0.01131712 1.151492
```

Für die polynomiale Regression erhalten wir:

```
head(allpol[, 4:6], 1)
```

```
            EBias   VarFDach EQuadVor
[1,] -0.002993156 0.03610161 1.047767
```

**i)**

Der Bias (betragsmäßig) und der erwartete quadratische Vorhersagefehler sind im linearen Modell deutlich größer als polynomialen Modell. Die polynomiale Regression hat eine höhere Varianz als die lineare Regression.

**ii)**

Eine höhere Varianz führt zu einem niedrigeren Bias (Bias-Varianz-Trade-Off). Bias und erwartete quadratische Vorhersagefehler gehen Hand in Hand (sind positiv korreliert).

**iii)**

```
alllin20 <- matrix(NA, nrow = 10000, ncol = 6)
colnames(alllin20) <- c("fdach", "y", "quadVor", "EBias", "VarFDach", "EQuadVor")
allpol20 <- matrix(NA, nrow = 10000, ncol = 6)
colnames(allpol20) <- c("fdach", "y", "quadVor", "EBias", "VarFDach", "EQuadVor")
for (i in 1:10000) {
    x <- runif(20, 1, 5)
```

```
    residuen <- rnorm(20, 0, 1)
    b <- 4
    y <- b * log(x) + residuen
    lmlinear <- lm(y ~ x)
    lmpoly <- lm(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5))
    alllin20[i, 1] <- lmlinear$coefficients %*% c(1, 3)
    allpol20[i, 1] <- lmpoly$coefficients %*% c(1, 3, 3^2, 3^3, 3^4, 3^5)

    alllin20[i, 2] <- b * log(3) + rnorm(1)
    allpol20[i, 2] <- alllin20[i, 2]

    alllin20[i, 3] <- (alllin20[i, 2] - alllin20[i, 1])^2
    allpol20[i, 3] <- (allpol20[i, 2] - allpol20[i, 1])^2
}
alllin20[, 4] <- mean(alllin20[, 1]) - 4 * log(3)
allpol20[, 4] <- mean(allpol20[, 1]) - 4 * log(3)

alllin20[, 5] <- var(alllin20[, 1])
allpol20[, 5] <- var(allpol20[, 1])

x4 <- rep(3, 20)
y4 <- gendata(x4, 4)
alllin20[, 6] <- mean((y4 - alllin20[, 1])^2)
allpol20[, 6] <- mean((y4 - allpol20[, 1])^2)
```

$n = 20$.

```
head(alllin20[, 4:6], 1)
```

```
          EBias   VarFDach EQuadVor
[1,] -0.3290356 0.06066898  1.13206
```

```
head(allpol20[, 4:6], 1)
```

```
            EBias  VarFDach EQuadVor
[1,] -0.004694322 0.2390739 1.165156
```

$n = 10000$ zum Vergleich.

```
head(alllin[, 4:6], 1)
```

```
          EBias   VarFDach EQuadVor
[1,] -0.3440677 0.01131712 1.151492
```

```
head(allpol[, 4:6], 1)
```

```
            EBias   VarFDach EQuadVor
[1,] -0.002993156 0.03610161 1.047767
```

Wir hätten erwartet, dass die Prognosewerte für die lineare Regression sich nicht großartig ändern. Die für die polynomiale Regression hingegen schon. Wir stellen fest, dass der Bias für die lineare Regression sich nur kaum ändert. Die Lineare Regression hat für ein kleines $n$ einen besseren erwarteten quadratischen Vorhersagewert.

**f)**

Da die Logarithmusfunktions höchstgradig nicht linear ist, ist die polynomiale Regression ein genaueres Modell und liefert bessere Ergebnisse. Dies wird insbesondere im Bias und im erwarteten quadratischen Vorhersagewert widergespiegelt. Man merkte, dass für kleine $n$ die lineare Regression bessere Werte liefert. Dies liegt insbesondere daran, dass bei $n = 20$ der Abstand von Punkten zu einer Geraden sich nicht großartig ändert (generell hoher Bias dafür niedrige Varianz) aber die polynomiale Regression durch die höhere Varianz bei kleinen Stichproben eher schlechtere Werte liefert, da zwar einige Punkte deutlich besser getroffen werden, dafür wiederum andere Punkte deutlich weiter verfehlt werden.