```
library(lattice)
epg = read.table(file.path(pfadu, "epg.txt"))
amp = read.table(file.path(pfadu, "dbdauer.txt"))
head(epg)

#################################################################
# Abbildungen: entweder mit plot()
#################################################################
# 1 Spalte, 3 Reihen
par(mfrow=c(1,3))
# F2: abhängige Variable, COG = unabhängige Variable
plot(F2 ~ COG, data = epg)
# F1: abhängige Variable, COG = unabhängige Variable
plot(F1 ~ COG, data = epg)
# 4 Vokale auswählen
temp = with(epg, V %in% c("i", "I", "E", "a"))
# F1 abhängige Variable, SUM1278 unabhängige Variable
plot(F1 ~ SUM1278, data = epg[temp,])

#################################################################
# Abbildungen: oder mit xyplot() aus library(lattice)
#################################################################
# F2: abhängige Variable, COG = unabhängige Variable
bild1 = xyplot(F2 ~ COG, data = epg)
# F1: abhängige Variable, COG = unabhängige Variable
bild2 = xyplot(F1 ~ COG, data = epg)
# 4 Vokale auswählen
temp = with(epg, V %in% c("i", "I", "E", "a"))
# F1 abhängige Variable, SUM1278 unabhängige Variable
bild3 = xyplot(F1 ~ SUM1278, data = epg[temp,])
# Abbildungen (siehe Vorlesung 2)
plot(bild1, split = c(1, 1, 3, 1), more=T)
plot(bild2, split = c(2, 1, 3, 1), more=T)
plot(bild3, split = c(3, 1, 3, 1))

#################################################################
# Kovarianz
#################################################################
# Kovarianz
# Abhängige Variable
y = epg$F2
# Unabhängige Variable
x = epg$COG
# Anzahl der Stichproben
n = length(y)
# Mittelwert von x
mx = mean(x)
# Mittelwert von y
my = mean(y)
# Stichproben-Abweichung vom Mittelwert
dx = x - mean(x)
dy = y - mean(y)
# Kovarianz = Produkt dieser Abweichungen
```

```
# dividiert durch den Stichprobenanzahl minus 1
covxy = sum(dx * dy)/(n-1)
# das gleiche
cov(x,y)

##################################################################
# Korrelation
##################################################################
xgross = x * 1000
cov(x,y); cov(xgross,y)
# Korrelation: die Kovarianz dividiert durch
# den Produkt der Standardabweichungen der beiden Variablen
r = cov(x,y)/(sd(x) * sd(y))
# Mit der cor() Funktion
cor(x,y)
cor(xgross,y)

##################################################################
# Regression
##################################################################
########################## Die Steigung: entweder
b = r * sd(y)/sd(x)
# oder
b = cov(x,y)/var(x)
########################## Das Intercept
k = my - b * mx
########################## Die eingeschätzen Werte
yhut = b * x + k

########################## Abbildung mit überlagerter Regressionsline
########################## entweder:
par(mfrow=c(1,1))
plot(y ~ x)
abline(k, b)
# die eingeschätzen Werte überlagern
points(x, yhut, col=2)

########################## oder mit xyplot() in library(lattice)

mypanel = function(x, y, ...) {
    panel.xyplot(x, y, ...)
    panel.lmline(x, y)
}

xyplot(y ~ x, panel = mypanel)


############################ SSE
error =  y - yhut
SSE = sum(error^2)

##################################################################
# Regression und die lm() Funktion
```

```
#################################################################

########################### Regression berechnen
reg = lm(y ~ x)
########################### Regression überlagern
plot(y ~ x)
abline(reg)

########################### Steigung, Intercept
coef(reg)
########################### die eingeschätzten Werte
yhut = predict(reg)

# Der Error (Abstand zwischen den tatsächlichen und eingeschätzen Werten)
residuals(reg)
# SSE
deviance(reg)
# das gleiche (siehe oben)
sum(error^2)

#################################################################
# SSY, SSR, SSE
#################################################################
# SSY, SSR, SSE
SSY = sum( (y - my)^2)
SSR = sum((yhut - my)^2)
# bestätigen, dass SSR + SSE = SSY
SSR + SSE

# R-squared
SSR/SSY
# das gleiche
cor(x, y)^2

#################################################################
# Die Prüfstatistik
#################################################################
# Der Standard-Error von 'r'
rsb = sqrt( (1 - r^2) / (n - 2) )
# Die t-Statistik
tstat = r/rsb
# Die Wahrscheinlichkeit, dass die Werte durch eine Regressionslinie
# modelliert werden können
2 * (1 - pt(tstat, n-2))
# Das gleiche mit der Die F-Statistik
fstat = tstat^2
1 - pf(fstat, 1, n-2)

# Diese Informationen sind auch in summary() enthalten:
summary(reg)
# Residual standard error: 300 on 43 degrees of freedom
# Multiple R-squared:  0.7952,  Adjusted R-squared:  0.7905
# F-statistic:   167 on 1 and 43 DF,  p-value: < 2.2e-16
```

```r
# Es gibt eine signifikante lineare Beziehung zwischen
# COG und F2 (R^2 = 0.80, F[1, 43] = 167.0, p < 0.001).


################################################################
# Kriterien für die Durchführung einer Regression
################################################################
# Die Residuals:
# sollen einer Normalverteilung folgen:
shapiro.test(resid(reg))

#   (a). Shapiro-Wilk normality test
# data:  resid(reg)
# Wenn p > 0.05, ist es OK also die Residuals sind mit einer
# Normalverteilung konsistent
# W = 0.9704, p-value = 0.2987

#   (b). Residuals beobachten
# Die Residuals sollen auf eine randomisierte Weise
# um die Null-Linie verteilt sein - dies ist eventuell nicht der Fall
plot(resid(reg))
abline(h=0, lty=2)

#   (c). Keine Autokorrelation
# Vor allem sollen Werte bei lag 1 und lag 2 innerhalb des
# Konfidenzintervalls liegen. Dies ist nicht der Fall...
acf(resid(reg))

################################################################
# Beispiel
################################################################
# Inwiefern kann die Dauer aus der Amplitude (DB) vorhergesagt werden?
# Verwenden Sie die Regression, um die Dauer für einen DB-Wert
# von 40 dB einzuschätzen.

head(amp); dim(amp)
# Abbildung
plot(Dauer ~ DB, data = amp)
# Regression
reg = lm(Dauer ~ DB, data = amp)
# Linie überlagern
abline(reg)

# oder
mypanel = function(x, y, ...) {
    panel.xyplot(x, y, ...)
    panel.lmline(x, y)
}

xyplot(Dauer ~ DB, data = amp, panel = mypanel)


# Signifikanz-Test
```

```r
summary(reg)
# Es gibt eine lineare Beziehung
# zwischen der Dauer und Amplitude
# (R^2 = 0.79, F[1, 48] = 177.3 p < 0.001).

# Verifizieren, dass die Regression gültig ist
#########################################################
# a. Residuals Normalverteilt? OK.
shapiro.test(resid(reg))

# b.  Residuals sollen um die 0-Linie
# randomisiert verteilt sein
plot(resid(reg))
abline(h=0, lty=2)

# c. Keine Autokorrelation
acf(resid(reg))

#########################################################
# Vorhersage:
predict(reg, data.frame(DB=40))

# Entweder
plot(Dauer ~ DB, data = amp, xlim = c(0, 45))
abline(reg)
points(40, 97.02162, pch=16, col = "red")

# oder
mypanel = function(x, y, ...) {
    panel.xyplot(x, y, ...)
    panel.lmline(x, y)
    panel.points(40, 97.02162, pch=16, col = "red")
}

xyplot(Dauer ~ DB, data = amp, panel = mypanel, xlim = c(0, 45))
```