library(lattice)

###########################################################################

###### 0. Einleitung: Logistische Regression

###########################################################################

# Mit der logistischen Regression wird geprüft, ob

# Proportionen (abhängige Variable) von einem (oder mehreren) unabhängigen

# Faktoren beeinflusst werden.

#

# Die abhängige Variable ist immer kategorial und immer binär.

# Die unabhängige Variable

# kann numerisch oder kategorial (auch mehrstufig) sein.

#

# Beispiele:

# 1. Inwiefern wird die Vokalieriung

# von einem final /l/ im Englischen (feel vs. 'feeu')

# vom Dialekt beeinflusst?

# Abhängige Variable: Vokalisiert (kategorial, 2 Stufen: ja, nein)

# Unabhängige Variable: Dialekt (kategorial: 2 oder mehrere Stufen)

#

# 2. Wird 'passt' in Augsburg im Vgl. zu München eher mit /ʃ/ produziert?

# Abhängige Variable: Frikativ (kategorial, 2 Stufen: /s/, /ʃ/)

# Unabhängige Variable: Dialekt (kategorial, 2 Stufen: Augsburg, München)

#

# 3. Ein offener Vokal in /lVm/ wird mit unterschiedlichen Dauern synthetisiert.

# Nimmt die Wahrnehmung von 'lahm' vs. 'Lamm' mit zunehmender Dauer zu?

# Abhängige Variable: Vokal (kategorial, 2 Stufen: /a/, /a:/)

# Unabhängige Variable: Dauer (kontinuierlich)

# In der linearen (least-squares) Regression (Vorlesung voriger Woche)

# wird geprüft, inwiefern eine lineare Beziehung zwischen

# zwei Variablen, x und y vorliegt.

# Um dies zu tun, wird eine Regressionslinie

# mit Steigung m und Intercept k an die Stichproben angepasst.

# yhut in:

#

# yhut = m x + k

#

# sind dann die eingeschätzen Werte, die auf der Regressionslinie liegen

#

# Wir können aber eine solche Linie nicht an Proportionen anpassen,

# weil Proportionen zwischen 0 und 1 begrenzt sind (und die lineare

# Regression erwartet Werte zwischen ±unendlich).

# Stattdessen wird eine gerade Linie an sogenannte Log-Odds angepasst

#

# log(Odds) = m x + k

#

#

# Odds: P/Q. (Odds = Gewinnchancen).

# P ist 'Erfolg' - z.B. in Bsp. 3 Häufigkeit der 'lahm'-Urteile

# Q ist 'Misserfolg':Häufigkeit der 'lamm'-Urteile

# log(Odds) ist dann einfach log(P/Q)

# log(Odds) haben Werte zwischen ±unendlich

# Wenn P = Q dann ist log(P/Q) = log(1) = 0

# Dies ist auch der sogenannte 'Umkipppunkt': der

# Dauerwert (in diesem Fall) zu dem Hörer 'lahm' und 'Lamm'

# mit derselben Wahrscheinlichkeit wahrnehmen.

###########################################################################

###### 1. Erfolg (P), Misserfolg (Q), Log-Odds log(P/Q), und Proportionen (p)

###########################################################################

ovokal = read.table(file.path(pfadu, "ovokal.txt"))

head(ovokal)

# Zwischen 1950 und 2005 sollen Wörter wie 'lost' in

# einer aristokratischen Form der Standardaussprache von England

# immer weniger mit einem hohen Vokal /lo:st/

# und zunehmend mit einem tiefen Vokal /lɔst/ produziert.

# Ist dies Der Fall?

# Also:

# Wird der Vokal (hoch vs. tief) = abhängige Variable

# vom Jahr (1950... 2005) = unabhängige numerische Variable beeinflusst?

################################ P ist 'Erfolg'

# Wir nehmen den zweiten Wert von

levels(ovokal$Vokal)

# als P

P = ovokal$Vokal == "tief"

################################ Q ist 'Misserfolg' (= nicht Erfolg)

Q = !P

# P, Q in den Data-Frame einbinden

ovokal = cbind(ovokal, P, Q)

################################# Die Summe von P, Q pro Jahr berechnen

ovokal.m = aggregate(cbind(P, Q) ~ Jahr, sum, data = ovokal)

# z.B. Reihe 1 bedeutet: es gab 5 Mal Erfolg (tief) und 30 Mal

# Misserfolg (hoch) in 1950

head(ovokal.m)

# Proportionen berechnen: die Proportion vom Erfolg (0 <= p <= 1)

p = with(ovokal.m, P/(P+Q))

ovokal.m = cbind(ovokal.m, p)

# Log-odds berechnen: log(P/Q)

lodd = with(ovokal.m, log(P/Q))

ovokal.m = cbind(ovokal.m, lodd)

###########################################################################

###### 2. Die logistische Regressionslinie

###########################################################################

# Abbildung im Raum Log-Odds x unabhängige Variable

plot(lodd ~ Jahr, data = ovokal.m, ylab="Log-Odds")

# Eine Regressionslinie in diesem Raum wird mit

# glm(family=binomial) berechnet.

# glm: generalised linear model

# Entweder Anwendung auf den ursprünglichen Data-Frame

head(ovokal)

lreg = glm(Vokal ~ Jahr, family=binomial, data = ovokal)

# Oder Anwendung auf P, Q in dem gemittelten Data-Frame

lreg = glm(cbind(P, Q) ~ Jahr, family=binomial, data = ovokal.m)

# Das Intercept und die Steigung sind hier

coef(lreg)

# Und diese kann auf die Daten im Log-Odds Raum überlagert werden

abline(lreg)

#################################### Alternativ mit lattice

mypanel = function(x, y, ...) {

 panel.xyplot(x, y, ...)

 panel.abline(lreg)

}

xyplot(lodd ~ Jahr, data = ovokal.m, ylab="Log-Odds", panel = mypanel)

###########################################################################

###### 3. Die Prüfstatistik

###########################################################################

# Die Wahrscheinlichkeit wird geprüft,

# dass die Steigung von Null abweicht

# (weil wenn die Steigung Null wäre, dann wären alle Log-odds gleich

# und wir hätten eine gerade linie)

summary(lreg)

# Coefficients:

# Estimate Std. Error z value Pr(>|z|)

# (Intercept) -138.11742 20.99959 -6.577 4.80e-11 \*\*\*

# Jahr 0.07026 0.01066 6.593 4.32e-11 \*\*\*

#

# Vokal (ob hoch oder tief) wird signifikant vom Jahr beeinflusst

# (z = 6.6, p < 0.001).

#

# (Siehe auch <http://www.ats.ucla.edu/stat/r/dae/logit.htm> für mehr zur

# Prüfstatistik in der logistischen Regression).

###########################################################################

###### 4. Die Sigmoid Funktion

###########################################################################

# Diese Funktion kopieren

sig = function(k=0, m=1, xlim = c(-10, 10))

{

 # Funktion um Eigenschaften der Sigmoid zu zeigen

 counter = 1

 for(i in k){

 for(j in m){

 if(counter==1)

 curve(exp(j \* x + i)/(1 + exp(j \* x + i)), ylab="", ylim = c(0, 1), xlim = xlim)

 else

 curve(exp(j \* x + i)/(1 + exp(j \* x + i)), add=T, col = counter)

 counter = counter + 1

 }

 }

}

# Die Regressionslinie wird berechnet im Raum

# Log-Odds x unabhängige Variable (Jahr).

# Dieselben (k, m) Werte können verwendet werden, um

# statt Log-Odds auf der y-Achse Proportionen abzubilden.

# In dem Fall wandelt sich die gerade Linie in eine

# sogenannte Sigmoid-Funktion um.

# Eine Sigmoid-Funktion. Die mathematische Formel dafür:

# f(x) = e^(mx + k)/(1 + e^(mx + k))

# e ist die Exponentialfunktion, m und k sind die Steigung und Intercept

# Umsetzung in R

#

sig()

# m ist die Steigung: je größer m, umso steiler kippt

# die S-Kurve um

# Steigungen von 1, .5, , .25 (schwarz, rot, grün)

sig(m = c(1, .5, .25))

# Daher wenn die Steigung 0 (Null) ist, bekommt

# man eine gerade Linie, um den 0.5 Wert (wenn k = 0)

sig(m = 0)

# Höhere/tiefere k-Werte verschieben die Linie um den 0.5 Wert

sig(k = c(0, 1, -1), m = 0)

# Der Umkipppunkt: das ist wo der Sigmoid am steilsten

# ist; auch auch wo die Proportion = 0.5.

# Bekommt man mit -k/m

sig()

# Der Default: k = 0, m = 1

# Daher u = -k/m = 0

abline(v = 0, lty=2)

###########################################################################

###### 5. Proportionen abbilden

###########################################################################

plot(p ~ Jahr, data = ovokal.m, ylab = "Proportion 'tief'")

# Intercept

k = coef(lreg)[1]

# Steigung

m = coef(lreg)[2]

# Angepasste Sigmoid

curve(exp(m \* x + k)/(1 + exp(m \* x + k)), add=T)

# verifizieren, dass es wirklich ein Sigmoid ist!

plot(p ~ Jahr, data = ovokal.m, xlim = c(1920, 2020), ylim = c(0, 1),

ylab = "Proportion 'tief'")

curve(exp(m \* x + k)/(1 + exp(m \* x + k)), add=T)

# Der Umkipppunkt (das Jahr, zu dem sich laut dem Modell

# die Entscheidungen von hoch auf tief umkippt)

abline(v = -k/m, lty = 2)

# Dasselbe mit xyplot()

mypanel = function(x, y, ...) {

 panel.xyplot(x, y, ...)

 panel.curve(exp(m \* x + k)/(1 + exp(m \* x + k)), add=T)

 panel.abline(v = -k/m, lty = 2)

}

xyplot(p ~ Jahr, data = ovokal.m, xlim = c(1920, 2020), ylim = c(0, 1),

ylab = "Proportion 'tief'", panel = mypanel)

###########################################################################

###### 6. Umkipppunkte in einem synthetischen Kontinuum

###########################################################################

pvp = read.table(file.path(pfadu, "pvp.txt"))

# Ein 11-stufiges Kontinuum wurde synthesisert zwischen /pUp/ und /pYp/.

# Die Stimuli wurden 10 Mal einem Hörer einzeln präsentiert.

# Der Hörer musste pro Stimulus entscheiden: PUPP oder PÜPP?

# Zu welchem F2-Wert kommt der Umkipppunkt vor?

# (= zu welchem F2-Wert kippt die Entscheidung um von PUPP auf PÜPP?)

# P, Q, Proportionen, logodds berechnen

levels(pvp$Urteil)

# Erfolg

P = pvp$Urteil == "Y"

# Misserfolg

Q = !P

# in den Data-Frame einbinden

pvp = cbind(pvp, P, Q)

# summieren pro F2-Wert

pvp.m = aggregate(cbind(P, Q) ~ F2, sum, data = pvp)

# Proportionen

p = with(pvp.m, P/(P+Q))

pvp.m = cbind(pvp.m, p)

plot(p ~ F2, data = pvp.m, ylab = "Proportion /Y/-Urteile")

# (k,m) der Sigmoid berechnen

pvp.glm = glm(Urteil ~ F2, family=binomial, data = pvp)

# oder

pvp.glm = glm(cbind(P, Q) ~ F2, family=binomial, data = pvp.m)

# Koeffiziente

k = coef(pvp.glm)[1]

m = coef(pvp.glm)[2]

curve(exp(m \* x + k)/(1 + exp(m \* x + k)), add=T)

# Umkipppunkte

u = -k/m

abline(v = u, lty=2)

# Die Wahrscheinlichkeit, dass die Urteile

# durch F2-Änderungen beeinflusst werden:

summary(pvp.glm)

# Die Urteile wurden durch F2-Änderungen signifikant

# beeinflusst (z = 4.7, p < 0.001).

###########################################################################

###### 7. Der unabhängige Faktor ist kategorial

###########################################################################

# Die logistische Regression kann auf eine ähnliche Weise

# verwendet werden, wenn der unabhängige Faktor kategorial ist.

# (Der wesentliche Unterschied: man braucht nicht einen Sigmoid abzubilden,

# es wird kein Umkipppunkt berechnet)

sz = read.table(file.path(pfadu, "sz.txt"))

head(sz)

# 20 Vpn. 9 aus Bayern, 11 aus Schleswig-Holstein produzierten 'Sonne'.

# Der initiale Frikativ wurde als /z/ oder /s/ wahrgenommen.

# Wird die Stimmhaftigkeit vom Dialekt beeinflusst?

# Die Abbildung (beide Variablen sind kategorial, daher barchart() )

tab = with(sz, table(Dialekt, Frikativ))

prop = prop.table(tab, 1)

# Proportionen

barchart(prop, auto.key=T, horizontal=F, ylab="Proportion")

# Test

sz.glm = glm(Frikativ ~ Dialekt, family=binomial, data = sz)

summary(sz.glm)

# Der Test prüft die Wahrscheinlichkeit, dass sich die

# proportionale Verteilung s/z in den beiden Dialekten gleich ist.

#

# Die s/z Verteilung wurde signifikant vom Dialekt

# beeinflusst (z = 2.1, p < 0.05)