Statistik für Naturwissenschaftler 431.010
Woche 11-13: Regression basics (mit R Unterstützung)

Ass.-Prof. Dr. Wolfgang Trutschnig
Fachbereich Mathematik
Universität Salzburg
www.trutschnig.net

Salzburg, Juni 2015

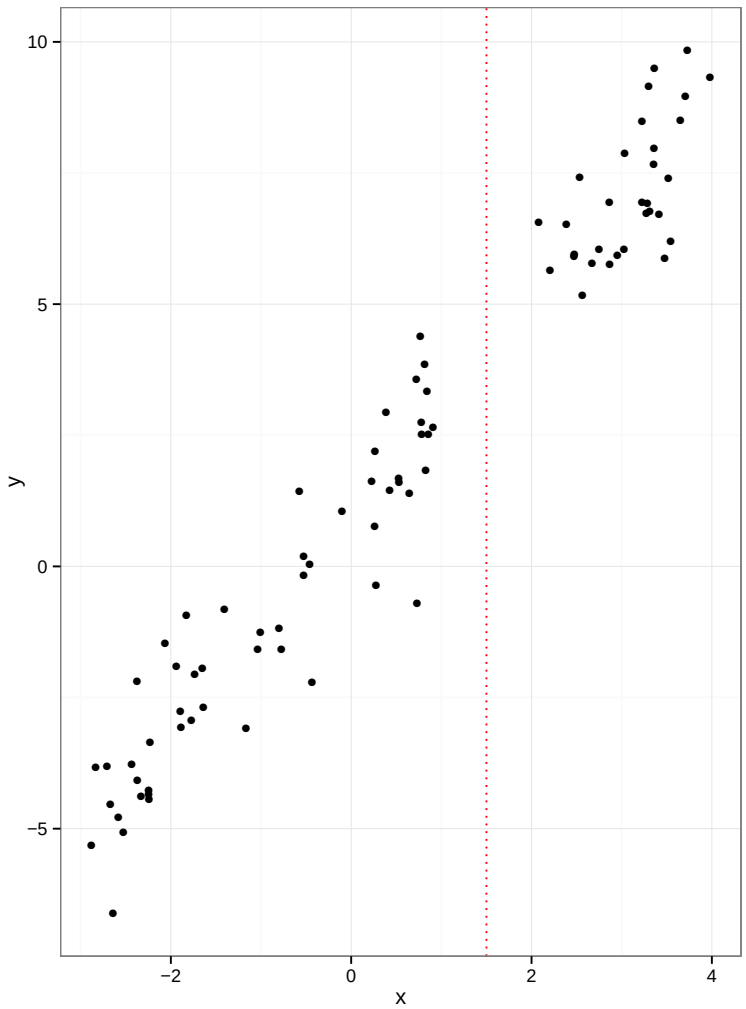


Abbildung: Prognose an der Stelle $x = 1.5$?

- Die Grafik zeigt Messwerte $(x_1, y_1), \ldots, (x_n, y_n)$.
- Es ist bekannt, dass ein linearer Zusammenhang zwischen den y- und den x-Werten besteht.
- Die Messungen der y-Werte sind fehlerbehaftet (in der Praxis immer der Fall).
- Modell: $y_i = ax_i + b + \varepsilon_i$ für $i \in \{1, \ldots, n\}$
- ε_i... zufällige Fehler mit $\mathbb{E}(\varepsilon_i) = 0$, die sich nicht gegenseitig beeinflussen.
- Gesucht: Prognose für den y-Wert an der Stelle $x = 1.5$

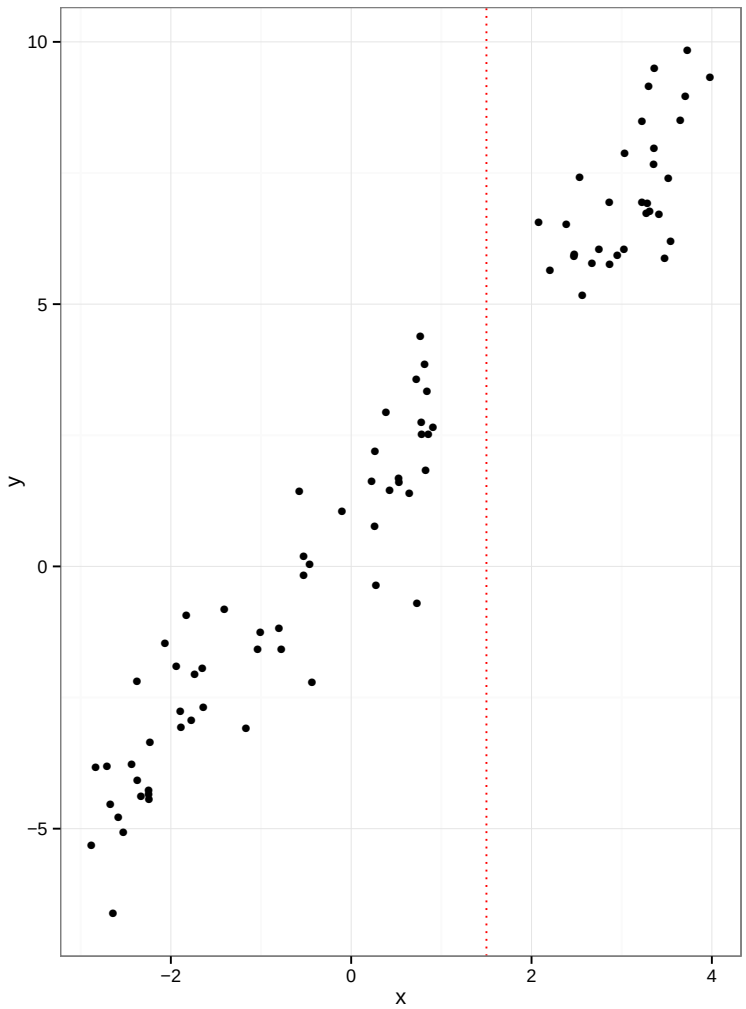
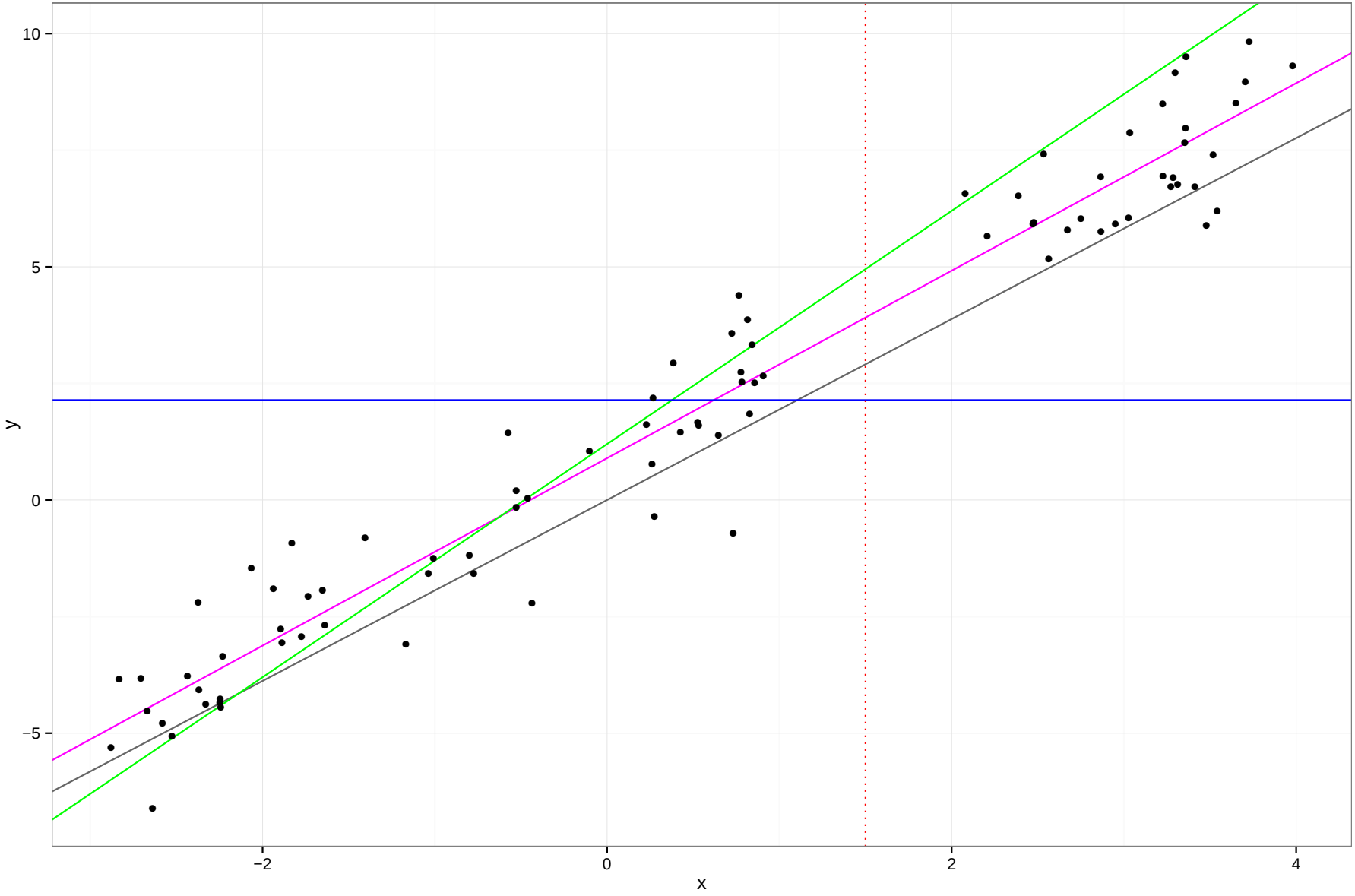


Abbildung: Prognose an der Stelle $x = 1.5$?

- Was tun ?
- Problem: Wir kennen a und b nicht
- Wählen a und b so, dass die Gerade $y = ax + b$ die Daten 'möglichst gut beschreibt'
- Die optimalen Werte bezeichnen wir mit $\hat{a}$ und $\hat{b}$
- Sobald wir $\hat{a}$ und $\hat{b}$ haben, prognostizieren wir $y = \hat{a} \cdot 1.5 + \hat{b}$.
- Es reicht also, $\hat{a}$ und $\hat{b}$ optimal aus den Daten zu schätzen.
- Welche der folgenden Geraden beschreiben die Daten gut (Ausgleichsgerade)?

Motivation + eindimensionale lineare Regression



- Wählen jene Werte für a und b, die den Prognosefehler minimieren
- Wenn wir a und b als Parameter wählen, dann ist der prognostizierte Wert an der Stelle x_i offensichtlich $ax_i + b$
- Der Fehler, den wir damit machen ist $y_i - (a \cdot x_i + b) = y_i - ax_i - b$
- Die Summe aller quadrierten Prognosefehler ist damit

$$F(a, b) := \sum_{i=1}^{n} \left(y_i - ax_i - b\right)^2 \tag{1}$$

- Wähle a und b so, dass $F(a, b)$ minimal ist.
- Analytische Berechnung liefert die folgenden optimalen Werte

$$\hat{a} = \frac{\sum_{i=1}^{n}(x_i - \overline{x}_n)(y_i - \overline{y}_n)}{\sum_{i=1}^{n}(x_i - \overline{x}_n)^2} \tag{2}$$

$$\hat{b} = \overline{y}_n - \hat{a}\,\overline{x}_n \tag{3}$$

- Für den konkreten Datensatz erhalten wir $\hat{a} = 2.010$ und $\hat{b} = 0.897$.
- Die Prognose an der Stelle $x = 1.5$ ist damit $y = 2.01 \cdot 1.5 + 0.897 = 3.912$

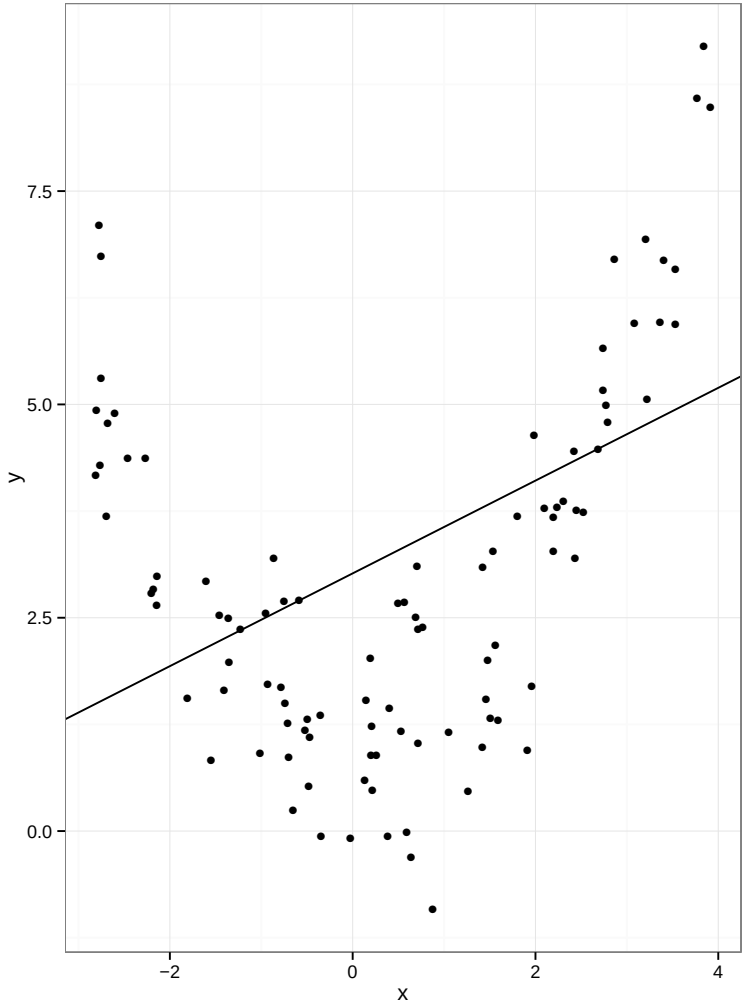
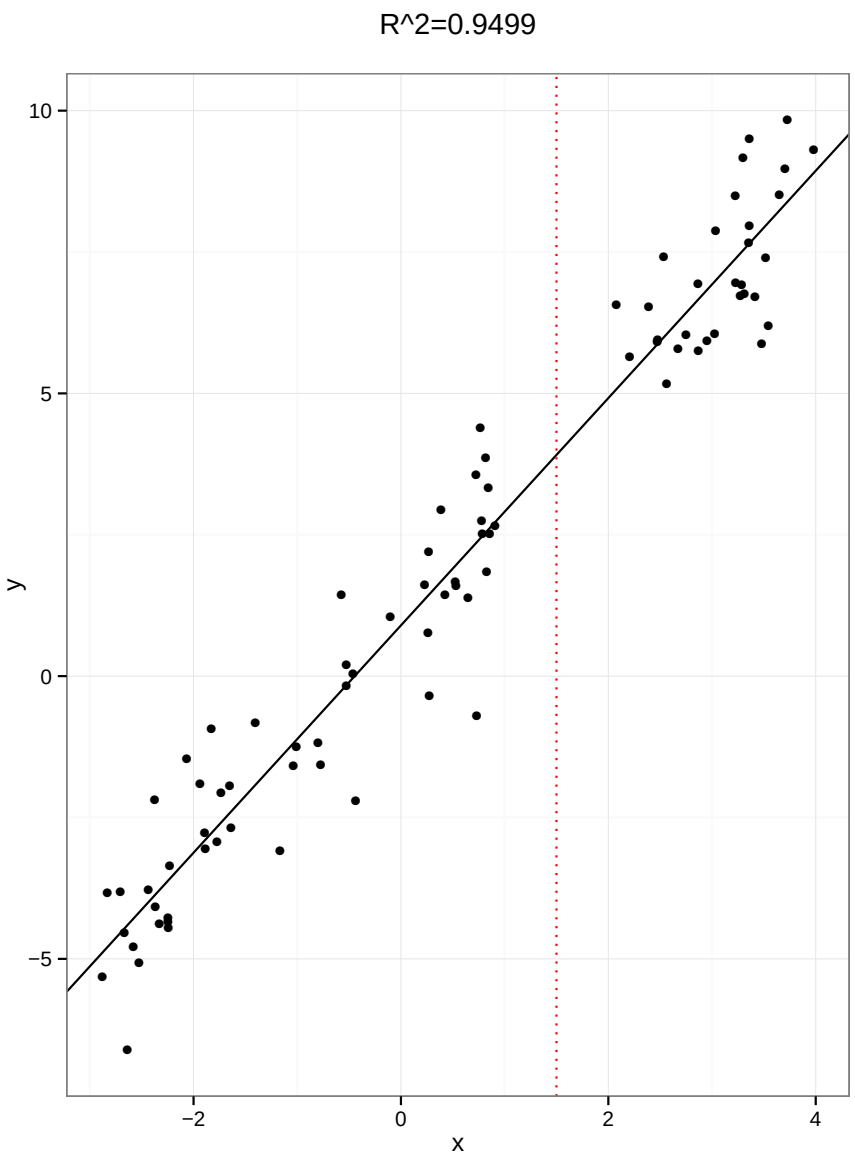


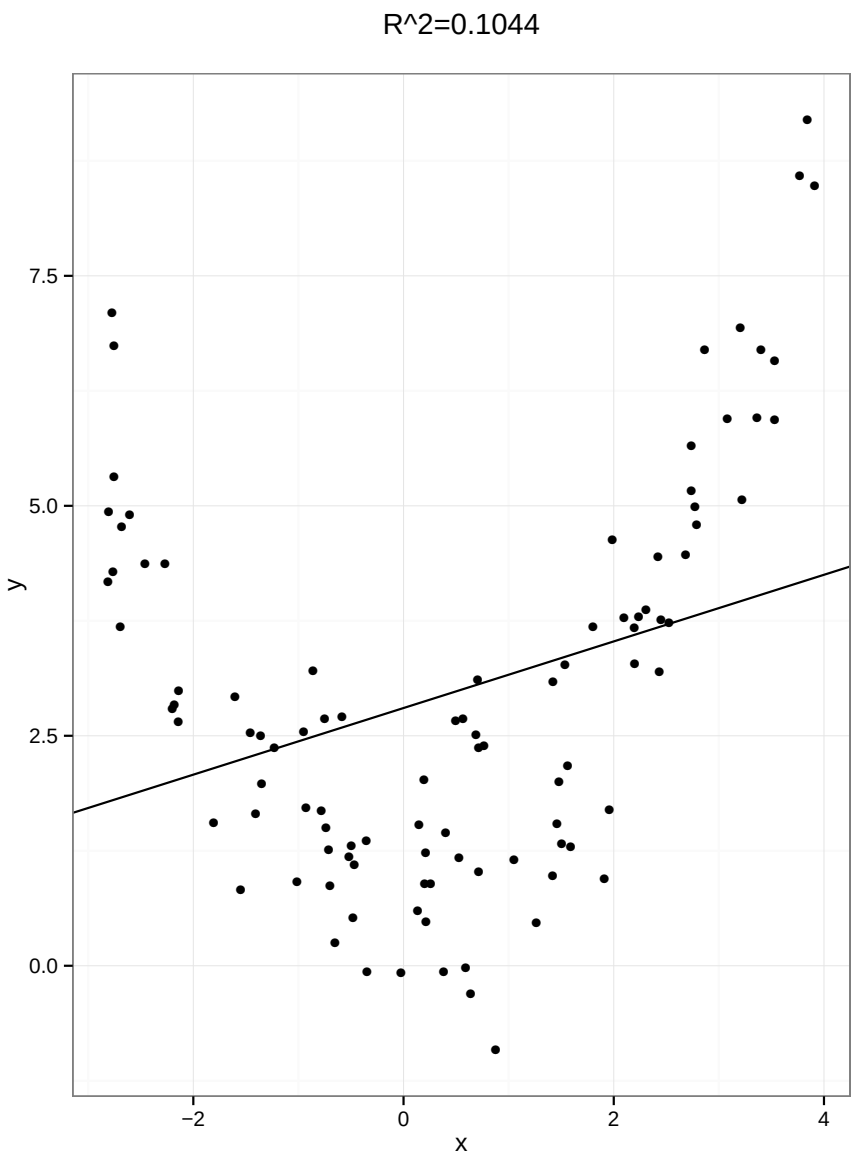
Abbildung: Auch linear ?

- Woher wissen wir, ob unser (optimales) Modell die Daten gut beschreibt ?
- Definiere $\hat{y}_i := \hat{a}x_i + \hat{b}$...Prognosewert an Stelle x_i
- $y_i - \hat{y}_i$ heißt *Residuum*
- Bestimmheitsmaß R^2 - Grundidee: Berechne, um wie viel die Varianz der Originaldaten $y_1, \ldots, y_n$ durch das Modell reduziert wird

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y}_n)^2} \quad (4)$$

Motivation + eindimensionale lineare Regression





- NB: **Je größer R^2 umso besser passt das Modell**, i.e. umso besser erklärt das Modell die Daten.
- Ist R^2 nahe bei 1 passt das Modell gut.
- Ist R^2 nahe bei 0 taugt das Modell nicht zur Erkärung der Daten.
- Im ersten Beispiel passt das linear Modell sehr gut.

- NB: Praktische Berechnung direkt in R - liefert die optimalen Parameter und R^2.
- Keinerlei Kenntnis der Formeln für $\hat{a}$ und $\hat{b}$ notwendig.

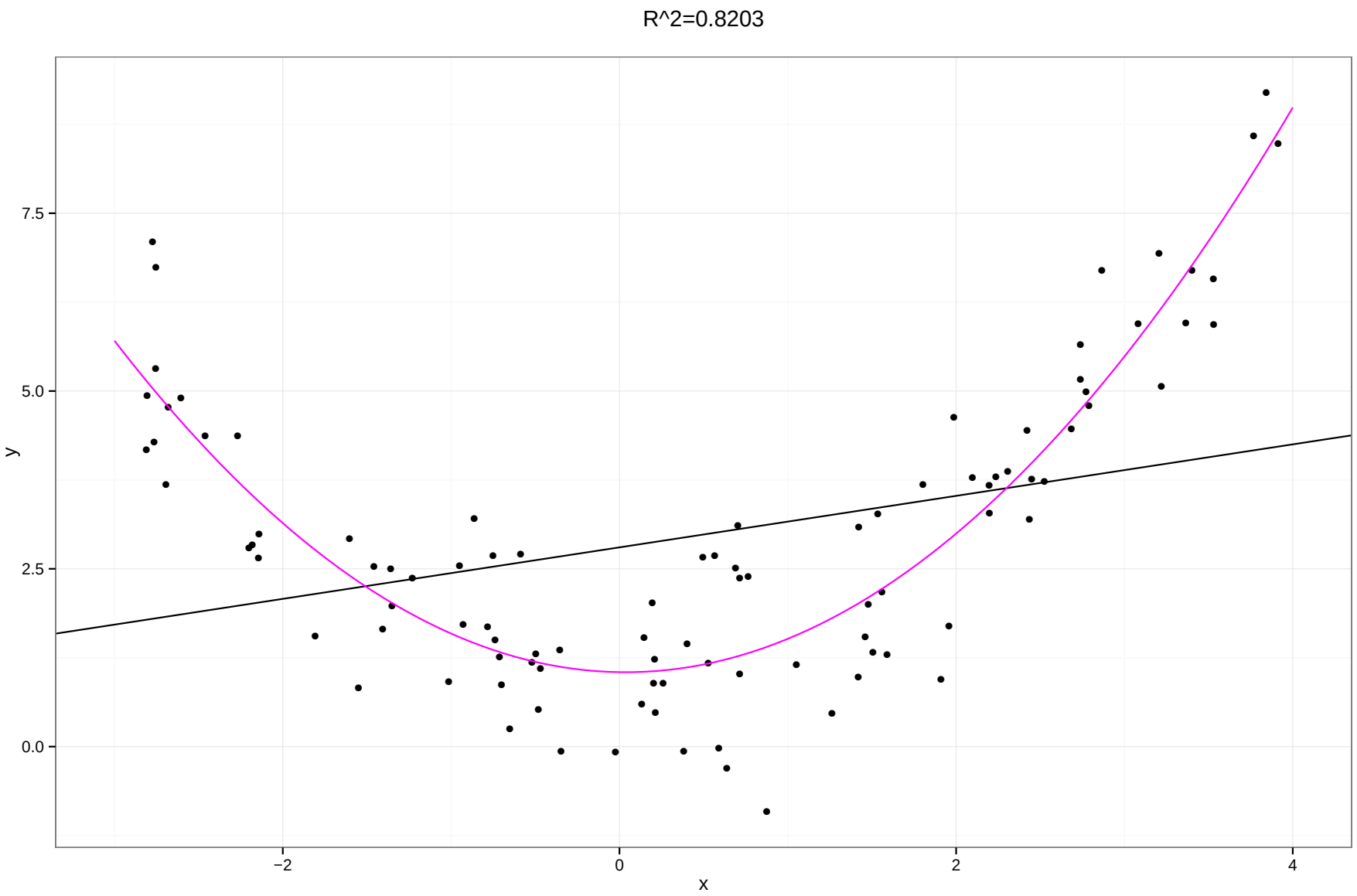
- Bevor wir uns die Handhabung in R anschauen
- Im zweiten Beispiel passt das Modell nicht (auch graphisch offensichtlich) $\rightarrow$ was tun ?
- Naheliegende Idee: Vermuten Zusammenhang der Form $y_i = b + a_1 x_i + a_2 x_i^2 + \varepsilon_i$, i.e. quadratischen Zusammenhang.

- Wir passen ein Modell der Form $y_i = b + a_1 x_i + a_2 x_i^2 + \varepsilon_i$ an den zweiten Datensatz an.
- Gehen analog zum vorigen Fall vor.
- Wenn wir a_1, a_2, b als Parameter wählen, dann ist der prognostizierte Wert an der Stelle x_i offensichtlich $a_1 x_i + a_2 x_i^2 + b$
- Der Fehler, den wir damit machen ist $y_i - (a_1 \cdot x_i + a_2 x_i^2 + b) = y_i - a_1 \cdot x_i - a_2 x_i^2 - b$
- Die Summe aller quadrierten Prognosefehler (Residuen) ist damit

$$F(a_1, a_2, b) := \sum_{i=1}^{n} (y_i - a_1 x_i - a_2 x_i^2 - b)^2 \tag{5}$$

- Wähle a_1, a_2, b so, dass $F(a_1, a_2, b)$ minimal ist.
- Direkte Berechnung in R liefert $\hat{a}_1 = -0.03688, \hat{a}_2 = 0.50546, \hat{b} = 1.04678$ sowie $R^2 = 0.8203$

Polynomregression



- Siehe www.trutschnig.net/R-codes_GEO_11.R
- **Ersten Datensatz** laden und plotten:

```
file <- url("http://www.trutschnig.net/geo_reg1.RData")

load(file)

head(geo_reg1)

A<-geo_reg1

plot(A)
```

- Lineares Modell anpassen:

```
model<-lm(data=A,y~x)

model
```

- Liefert:

```
Call:
lm(formula = y ~ x, data = A)

Coefficients:
(Intercept)            x
      0.897        2.010
```

- Also $\hat{a} = 2.01$ und $\hat{b} = 0.897$

- `summary(model)` liefert die folgende ausführlichere Information:

```
Call:
lm(formula = y ~ x, data = A)

Residuals:
     Min       1Q   Median       3Q      Max
-3.07477 -0.63681 -0.03544  0.70030  1.95308

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.89704    0.11406   7.865 1.13e-11 ***
x            2.00965    0.05035  39.917  < 2e-16 ***
---
Signif. codes:  0  ***  0.001  **  0.01  *  0.05  .  0.1     1

Residual standard error: 1.03 on 84 degrees of freedom
Multiple R-squared:  0.9499,    Adjusted R-squared:  0.9493
F-statistic:  1593 on 1 and 84 DF,  p-value: < 2.2e-16
```

- Insbesondere also $R^2 = 0.9499$
- Direktes Auslesen von R^2 mittels `summary(model)$r.squared`

- Prognosen mit Hilfe des Modells:
- Prognose an der Stelle $x = 1.5$

```
new<-data.frame(x=1.5)
predict(model,newdata=new)
```

- Liefert

```
3.91152
```

- Prognose an den Stellen $1.25, 1.5, 1.75$

```
new<-data.frame(x=c(1.25,1.5.75)
predict(model,newdata=new)
```

- Liefert

```
3.409106 3.911520 4.413933
```

- Für beliebige andere Stellen analog.

- Siehe www.trutschnig.net/R-codes_GEO_11.R
- **Zweiten Datensatz** laden und plotten:

```
file <- url("http://www.trutschnig.net/geo_reg2.RData")

load(file)

head(geo_reg2)

B<-geo_reg2

plot(B)
```

- Lineares Modell der Form $y = a_1 x + a_2 x^2 + b + \varepsilon$ anpassen:

```
model<-lm(data=B,y~x+I(x^2))

model
```

- Liefert:

```
Call:
lm(formula = y ~ x + I(x^2), data = B)

Coefficients:
(Intercept)            x       I(x^2)
    1.04678     -0.03688      0.50546
```

- Also $\hat{a}_1 = -0.03688$, $\hat{a}_2 = 0.50546$ und $\hat{b} = 1.04678$

- `summary(model)` liefert die folgende ausführlichere Information:

```
Call:
lm(formula = y ~ x + I(x^2), data = B)

Residuals:
     Min       1Q   Median       3Q      Max
-2.31532 -0.68455 -0.04251  0.51362  2.06281

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.04678    0.12970   8.071  1.9e-12 ***
x           -0.03688    0.05232  -0.705    0.483
I(x^2)       0.50546    0.02571  19.660  < 2e-16 ***
---
Signif. codes:  0  ***  0.001  **  0.01  *  0.05  .  0.1     1

Residual standard error: 0.9184 on 97 degrees of freedom
Multiple R-squared:  0.8203,    Adjusted R-squared:  0.8166
F-statistic: 221.4 on 2 and 97 DF,  p-value: < 2.2e-16
```

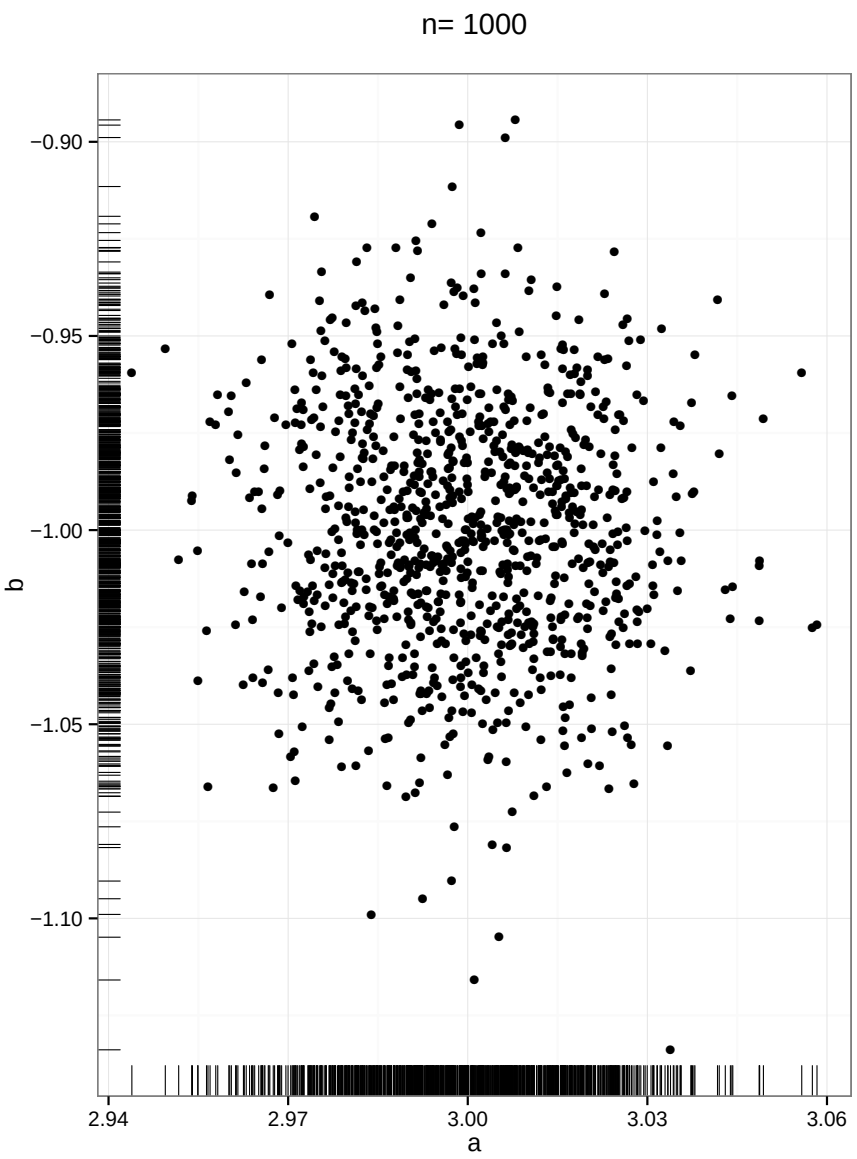
- Insbesondere also $R^2 = 0.8203$
- Direktes Auslesen von R^2 mittels `summary(model)$r.squared`

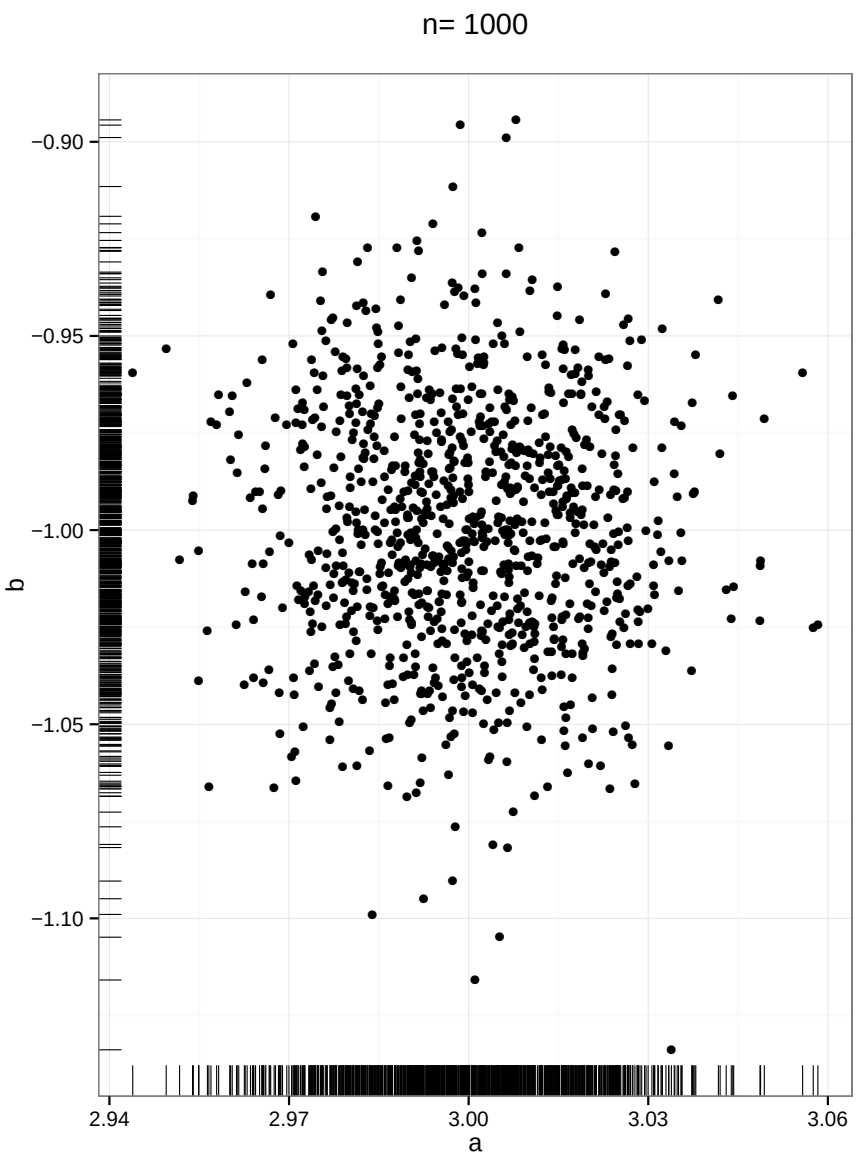
- Vorgangsweise scheint in beiden Beispielen zu funktionieren.
- Wie kann überprüft werden, ob die Vorgangsweise allgemein funktioniert ?
- Am Einfachsten mit Hilfe von Simulationen
- Wählen zum Beispiel $a = 3$ und $b = -1$ und generieren Daten $(x_1, y_1), \ldots (x_n, y_n)$ mit $n = 100$, wobei $y_i = 3x_i - 1 + \varepsilon_i$ und $\varepsilon_i \sim \mathcal{N}(0, 1)$ (normalverteilte Fehler)
- Liefert Schätzer $\hat{a}$ und $\hat{b}$.
- $\hat{a}$ sollte nahe bei $a = 3$ und $\hat{b}$ nahe bei $b = -1$ liegen.
- Umsetzung siehe www.trutschnig.net/R-codes_GEO_11.R
- Liefert

```
 Coefficients:
(Intercept)            x
     -1.084        3.020
```

- Wiederholen obige Schritte $R = 1000$ mal.

Umsetzung in R





- Naheliegende Verallgemeinerung des linearen Modells auf mehrere Dimensionen
- Betrachten den Fall von zwei erklärenden Variablen x_1 und x_2 (für mehr als zwei analog) und einer erklärten Variable Y.
- Gegeben sind Daten $(x_{1,1}, x_{2,1}, y_1), (x_{1,2}, x_{2,2}, y_2), \ldots (x_{1,n}, x_{2,n}, y_n)$
- Modell: $y_i = a_1 x_{1,i} + a_2 x_{2,i} + b + \varepsilon_i$
- ε_i... zufällige Fehler mit $\mathbb{E}(\varepsilon_i) = 0$, die sich nicht gegenseitig beeinflussen.
- Gehen genau gleich vor wie im eindimensionalen Fall
- Zielfunktion: Summe der quadrierten Prognosefehler (Residuen)

$$F(a_1, a_2, b) := \sum_{i=1}^{n} \left(y_i - a_1 x_{1,i} - a_2 x_{2,i} - b\right)^2 \tag{6}$$

- Wähle Parameter a_1, a_2, b so, dass $F(a_1, a_2, b)$ minimal wird.

- Siehe www.trutschnig.net/R-codes_GEO_12.R, Zeilen 01-09
- Datensatz laden und lineares Modell anpassen

```
file <- url("http://www.trutschnig.net/geo_reg_d2.RData")

load(file)

A<-geo_red_d2; head(A)
```

- Lineares Modell anpassen:

```
model<-lm(data=A,z~x1+x2) model
```

- Liefert:

```
Call:
lm(formula = y ~ x1 + x2, data = A)

Coefficients:
(Intercept)           x1           x2
    0.04404      2.93824      1.96832
```

- Also $\hat{a}_1 \approx 2.93$, $\hat{a}_2 \approx 1.97$ und $\hat{b} \approx 0.04$

- `summary(model)` liefert die folgende ausführlichere Information:

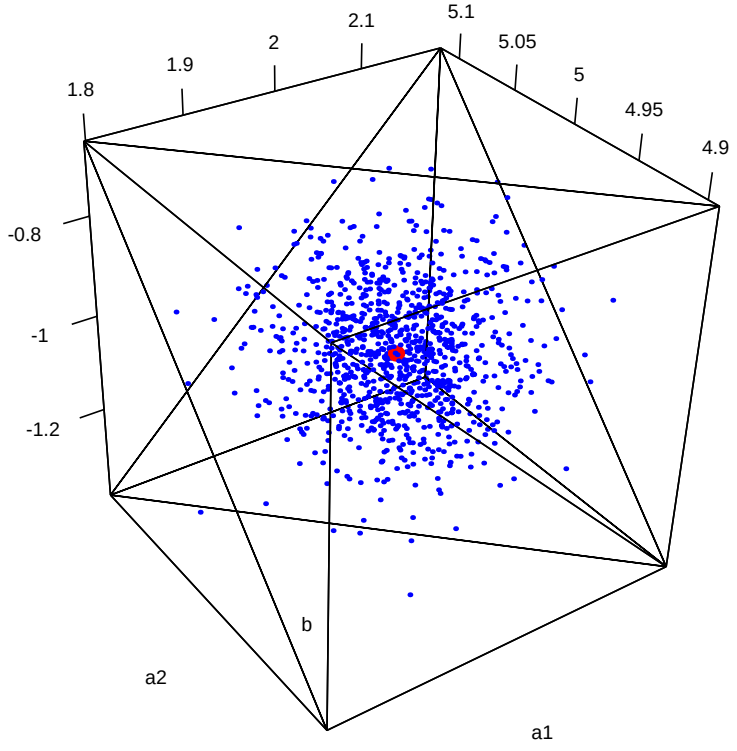
```
Call:
lm(formula = y ~ x1 + x2, data = A)

Residuals:
    Min      1Q  Median      3Q     Max
-3.5631 -0.6376  0.0564  0.9176  2.1860

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.04404    0.11306    0.39    0.698
x1           2.93824    0.04889   60.10   <2e-16 ***
x2           1.96832    0.06229   31.60   <2e-16 ***
---
Signif. codes:  0  ***  0.001  **  0.01  *  0.05  .  0.1     1

Residual standard error: 1.096 on 97 degrees of freedom
Multiple R-squared:  0.9791,    Adjusted R-squared:  0.9787
F-statistic:  2273 on 2 and 97 DF,  p-value: < 2.2e-16
```

- Insbesondere also $R^2 \approx 0.9791$; `summary(model)$r.squared`
- Eigenständige Berechnung: `residuals<-A$y-model$fitted.values`

  `R2<-1-var(residuals)/var(A$y)`

Mehrdimensionale lineare Regression

- Syntax vollkommen analog zum eindimensionalen Fall.
- Systematische Überprüfung mit Hilfe von Zeilen 15-35 in www.trutschnig.net/R-codes_GEO_12.R
- Was passiert, wenn die sample size n erhöht wird ?
- Was passiert, wenn die Varianz der Fehler vergrößert oder verringert wird ?
- **NB:** (i) Je größer n desto besser; (ii) je kleiner die Varianz der Fehler desto besser
- Funktioniert wunderbar $\rightarrow$ Bedenkenlos anwenden ?



Beispiel (Multikollinearität)

- Verwenden wiederum www.trutschnig.net/R-codes_GEO_12.R, Zeilen 37-45
-
```
Coefficients:
(Intercept)           x1           x2
   -0.02729      3.25548      0.87345

Coefficients:
(Intercept)           x1           x2
    0.07343      1.86555      1.56986
```
- Geschätzte Werte variieren extrem stark - warum ?
- Grund: Lt. Code gilt: `x2<-2*x1+runif(n,-0.3,0.3)`
- Die zwei erklärenden Variablen liegen fast auf einer Geraden (i.e. sind kollinear)
- Daher: Korrelation der erklärenden Variablen berechnen:

  `cor(A$x1,A$x2)`
- Falls knapp bei 1 oder $-1 \rightarrow$ nur eine der beiden Variablen verwenden

- In der Praxis häufig auftretendes Problem: viele erkärende Variablen - welche sind relevant ?
- Overfitting (Überanpassung): Modell zu kompliziert, enthält irrelevante erklärende Variable
- Mögliche Strategie: Wähle Modell mit möglichst niedrigem BIC-Wert (Bayes'schen Informationskriterium)
- Implementiert in R, zum Beispiel R-package `leaps` (zusätzliche Werkzeuge: `car` package)

Beispiel (Overfitting)

- Verwenden www.trutschnig.net/R-codes_GEO_12.R, Zeilen 51-61
- Output unabhängig von den Variablen x_3 und x_4
- Trotzdem wird Modell mit allen vier vorhandenen Variablen angepasst.
- `r<-regsubsets(y~x1+x2+x3+x4,data=A,nbest=2); plot(r)` zeigt Modelle mit niedrigsten BIC-Werten (2 pro Anzahl erkärender Variablen)

- Allgemeines Regressionsmodell: $Y = r(X) + \varepsilon$
- ε...zufälliger Fehler mit $\mathbb{E}(\varepsilon) = 0$
- Gegeben: Daten $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
- ZIEL: Möglichst genaue Bestimmung der Regressionsfunktion r
- Alle bisher betrachteten Regressionsmodelle waren *parametrisch*, i.e. r hing (von einer festen Anzahl) von Parametern ab.
- Bsp: $r(x) = ax + b$ (eindimensionale lineare Regression)
- Bsp: $r(x) = a_1 x + a_2 x^2 + b$ (eindimensionale quadratische Regression)
- Bsp: $r(x_1, x_2) = a_1 x_1 + a_2 x_2 + b$ (zweidimensionale lineare Regression)
- **Problem: Woher kennt man die Form von r? In der Praxis ist r oft gänzlich unbekannt.**
- Was tun?
- $\rightarrow$ Nichtparametrische Techniken: Kernregression, local weighted linear/polynomial regression, etc.

- **(Nadaraya-Watson) Kernregression:** Gegeben Daten $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$.
- Wir setzen

$$\hat{r}_n(x) := \frac{\sum_{i=1}^n K\left(\frac{x-x_i}{h}\right) y_i}{\sum_{i=1}^n K\left(\frac{x-x_i}{h}\right)}, \qquad \text{wobei} \tag{7}$$

- K...**Kern**: Wahrscheinlichkeitsdichte (Zum Beispiel: Dichte von $\mathcal{N}(0,1)$ oder Dichte von $\mathcal{U}(-1,1)$)
- $h > 0$...**Bandbreite** (Stärke der Glättung)
- x...Punkt an dem ausgewertet wird
- Entspricht genau einem gewichteten Mittelwert der $y_1, \ldots, y_n$ - je größer $|x - x_i|$ desto weniger Gewicht hat x_i in der Berechnung von $\hat{r}_n(x)$
- Einfache Berechnung - implementiert in R.
- Veranschaulichung der Funktionalität direkt in R.

Beispiel (Kernregression)

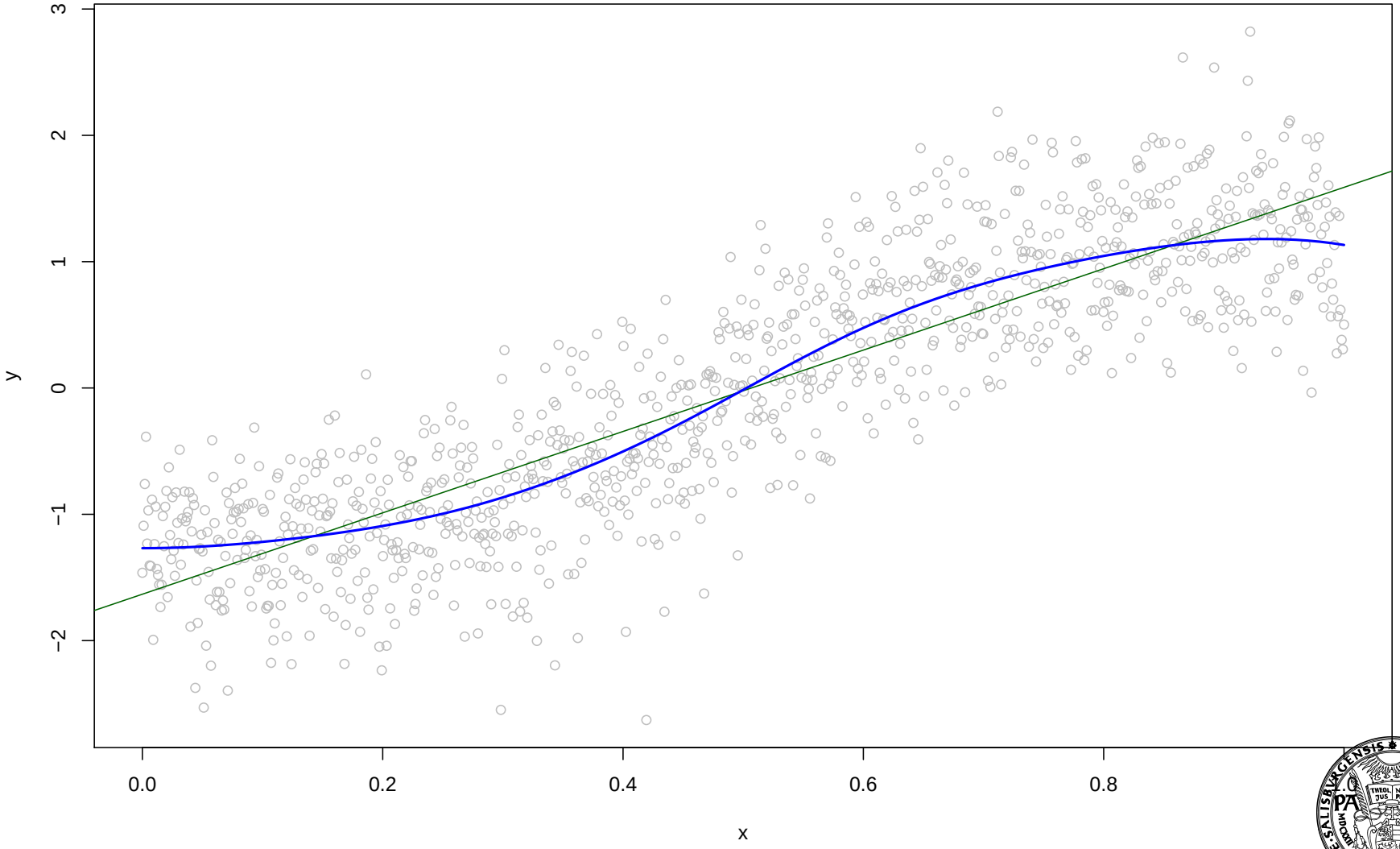
- Sprung ins kalte Wasser - direkt ausprobieren
- Wir betrachten den Datensatz `reg_data.RData` auf www.trutschnig.net
- Verwenden www.trutschnig.net/R-codes_GEO_12.R, Zeilen 69-81
- Kernregression kann mittels des R-packages `sm` (Smoothing Methods) berechnet werden (erweiterte Funktionalität: R package `np`).
- R-Code für $\hat{r}_n(0.6)$:

```
library(sm)
nreg<-sm.regression(A$x,A$y,eval.points=c(0.6),display="none")
nreg$estimate
points(0.6,nreg$estimate,col="blue",cex=1)
```

- Berechne $\hat{r}_n$ auf Gitter und ergänze resultierende Kurve im plot
- R-Code:

```
nreg<-sm.regression(A$x,A$y,eval.points=seq(0,1,length=101),
          display="none")
lines(nreg$eval.points,nreg$estimate,type="l",col="blue",lwd=2)
```

Nichtparametrische Regression



Beispiel (Kernregression, cont.)

- Berechne R^2:

```
> nreg<-sm.regression(A$x,A$y,eval.points=A$x,display="none")
> res<-A$y-nreg$estimate
> R2<-1-var(res)/var(A$y)
> R2
[1] 0.7833464
```

- Die Funktion `sm.regression` wählt automatisch die Bandbreite h
- Kann aber auch händisch eingegeben werden
- Veranschaulichung des Einflusses der Bandbreite mit Shiny
- Grundkonzept: Je kleiner h desto schneller fällt das Gewicht ab, i.e. desto weniger Werte haben Einfluss in die Gewichtung
- Zu großes h liefert zu starke Glättung.
- Zu kleines h liefert zu zittrige (wenig glatte) Kurve
- Berechne R^2 für die von `sm.regression` automatisch berechnet Bandbreite h

- Das vorige Beispiel war rein deskriptiv.
- Wollen die Güte der Schätzungen mittels Kernregression überprüfen.
- Ist es mittels Kernregression möglich, die echte Regressionsfunktion $r(x)$ gut zu schätzen ?
- Arbeiten (wie im parametrischen Fall) wieder mit Simulationen:
  - wählen feste Regressionsfunktion $r(x)$
  - erzeugen Stichproben $(x_1, y_1), \ldots, (x_n, y_n)$ mit $y_i = r(x_i) + \varepsilon$
  - berechnen die Kernregression $\hat{r}_n$
  - schauen wie nahe $\hat{r}_n$ bei r liegt.

Beispiel (Güte I)

- Modell $Y = \arctan(6x - 3) + \varepsilon$
- also $r(x) = \arctan(6x - 3)$
- Verwenden www.trutschnig.net/R-codes_GEO_12.R, Zeilen 93-103
- Beobachtungen: $\hat{r}_n$ liegt für großes n sehr nahe bei r

Beispiel (Güte II)

- Modell $Y = 2X + 3 + \varepsilon$
- also $r(x) = 2x + 3$
- Verwenden www.trutschnig.net/R-codes_GEO_12.R, Zeilen 106-118
- Beobachtungen: $\hat{r}_n$ liegt für großes n sehr nahe bei r

Beispiel (Güte III)

- Modell $Y = \frac{3}{1+2e^{-X}} + \varepsilon$
- also $r(x) = \frac{3}{1+2e^{-x}}$
- Verwenden www.trutschnig.net/R-codes_GEO_12.R, Zeilen 123-137
- Beobachtungen: $\hat{r}_n$ liegt für großes n sehr nahe bei r

Zusammenfassend:

- Kernregression funktioniert sehr gut für hinreichend große sample size n.
- Funktionalität (Konvergenz von $\hat{r}_n$ gegen $r(x)$) auch mathematisch sauber beweisbar.
- Kernregression funktioniert auch für zwei, maximal drei erklärende Variablen gut, in höheren Dimensionen aber problematisch (sample size, Rechenaufwand).
- Oft man es Sinn, mit der Bandbreite h zu spielen und zu beobachten, was passiert (i.e. nicht nur das automatisch gewählte h verwenden).
- Veranschaulichendes abschließendes Beispiel in shiny.

Vorgangsweise in der Praxis (eindimensionaler Fall)

- Gegeben sind Daten $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ des Modells $Y = r(X) + \varepsilon$
- ZIEL: Schätzung der Regressionsfunktion r (meist keinerlei Vorinformation)
- Berechne (für hinreichend großes n) die Kernregression $\hat{r}_n$
- Oftmals versucht man aufgrund der Form von $\hat{r}_n$ (annähernd linear, S-förmig, etc.) ein parametrisches Modell anzupassen
- Bsp: Für die Anpassung eines quadratischen Modells der Form $r(x) = a_1 x + a_2 x^2 + b$ muss die Funktion F, definiert durch

$$F(a_1, a_2, b) := \sum_{i=1}^{n} \left(y_i - a_1 x_i - a_2 x_i^2 - b\right)^2 \tag{8}$$

  minimiert werden.
- In R direkt umsetzbar mittels Funktion `lm` (siehe R-Codes), Minimierungsproblem analytisch lösbar

- In vielen Fällen muss numerisch minimiert werden
- In R machbar mit Hilfe der Funktion `optim`

Beispiel (Lineare regression via `optim`)

- Laden und plotten den Datensatz `geo_reg1.RData`

```
file <- url("http://www.trutschnig.net/geo_reg1.RData")
load(file)
head(geo_reg1)
A<-geo_reg1
plot(A)
```

- Anpassung des linearen Modells mit `lm`

```
model<-lm(data=A,y~x)
model
abline(model)
summary(model)

Call:
lm(formula = y ~ x, data = A)

Coefficients:
(Intercept)            x
      0.897        2.010
```

Beispiel (Lineare regression via `optim`, cont.)

- Anpassung des linearen Modells mit `optim`

```
F <- function(v){
  predict<- v[1]*A$x + v[2]
  r<-sum((A$y-predict)^2)
  return(r)
}

res<-optim(par=c(2,0), fn=F)
res
```

- Liefert

```
$par
[1] 2.0095869 0.8969354

$value
[1] 89.0495

$counts
function gradient
      57       NA
$convergence
[1] 0
```

Beispiel (Lineare regression via `optim`, cont.)

- Prognose an den Stellen $x = 1.25, 1.5, 1.75$ mit `lm`

```
new<-data.frame(x=c(1.25,1.5,1.75))
predict(model,newdata=new)

       1        2        3
3.409106 3.911520 4.413933
```

- Prognose an den Stellen $x = 1.25, 1.5, 1.75$ mit `optim`

```
reg<-function(v,x){
  r<-v[1]*x + v[2]
  return(r)
}
reg(v=res$par,x=c(1.25,1.5,1.75))

3.408919 3.911316 4.413712
```

- `optim` liefert selbe Resultate und ist auch für die Anpassung allgemeiner parametrischer Funktionen geeignet
- $\rightarrow$ Sehr flexibles Werkzeug !

- Für quadratische Regression analog - in diesem Fall ist der Parametervektor v dreidimensional.
- Betrachten anderes Beispiel - Aufgabe 63 mit Hilfe von `optim`

Beispiel (Fitting via `optim`, cont.)

- Laden und plotten den Datensatz `moisture.txt`

```
A <- read.table("http://www.trutschnig.net/moisture.txt",head=TRUE)
A<-moisture
head(A)
plot(A,col="gray")
```

- Anpassung des Modells mit `optim`

```
F <- function(v){
  predict<- 125-110*(1-exp(-v*A$tiefe))
  r<-sum((A$Wassergehalt-predict)^2)
  return(r)
}
res<-optim(par=c(1), fn=F)
res
```

Beispiel (Fitting via `optim`, cont.)

- Liefert

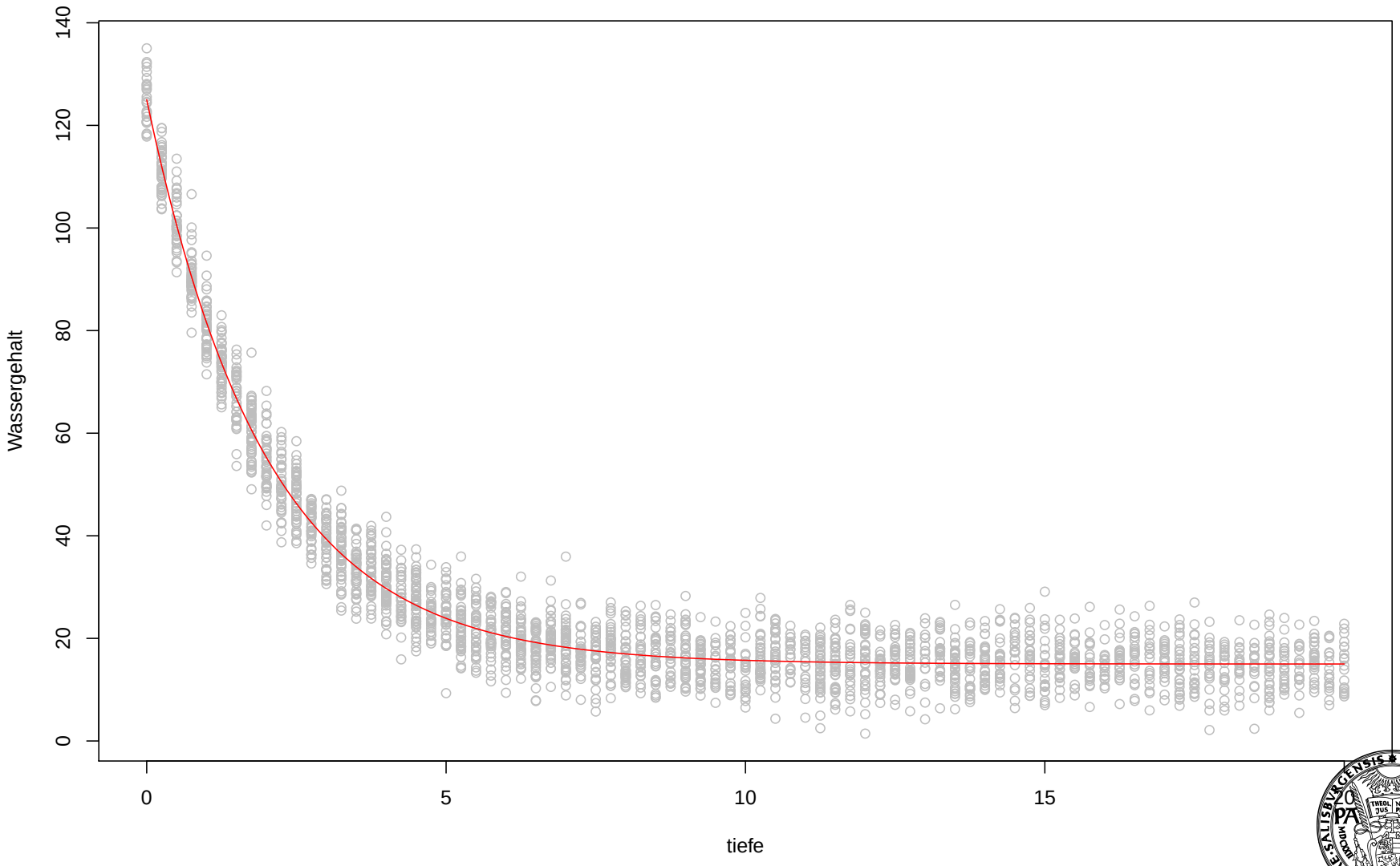
```
$par
[1] 0.5026855
```

- Berechne Regressionsfunktion auf grid und ergänze Graph im plot

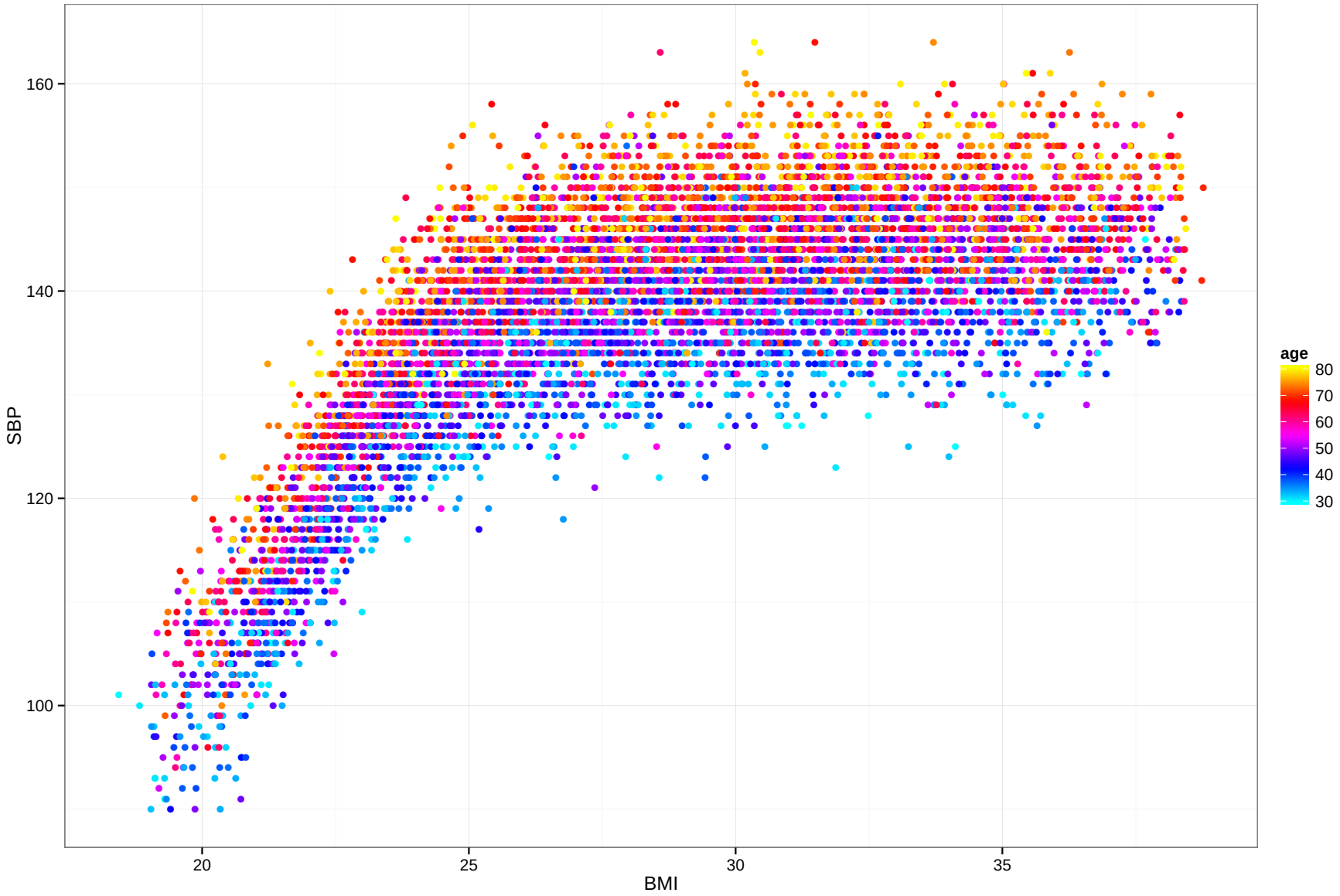
```
reg<-function(v,tiefe){
  r<-125-110*(1-exp(-v*tiefe))
  return(r)
}

tg<-seq(0,20,length=1001)
pred<-reg(v=res$par,tg)
lines(tg,pred,col="red",type="l")
```

- Passt wunderbar
- Wäre es einfacher gegangen ?

Anpassung (nichtlinearer) parametrischer Modelle



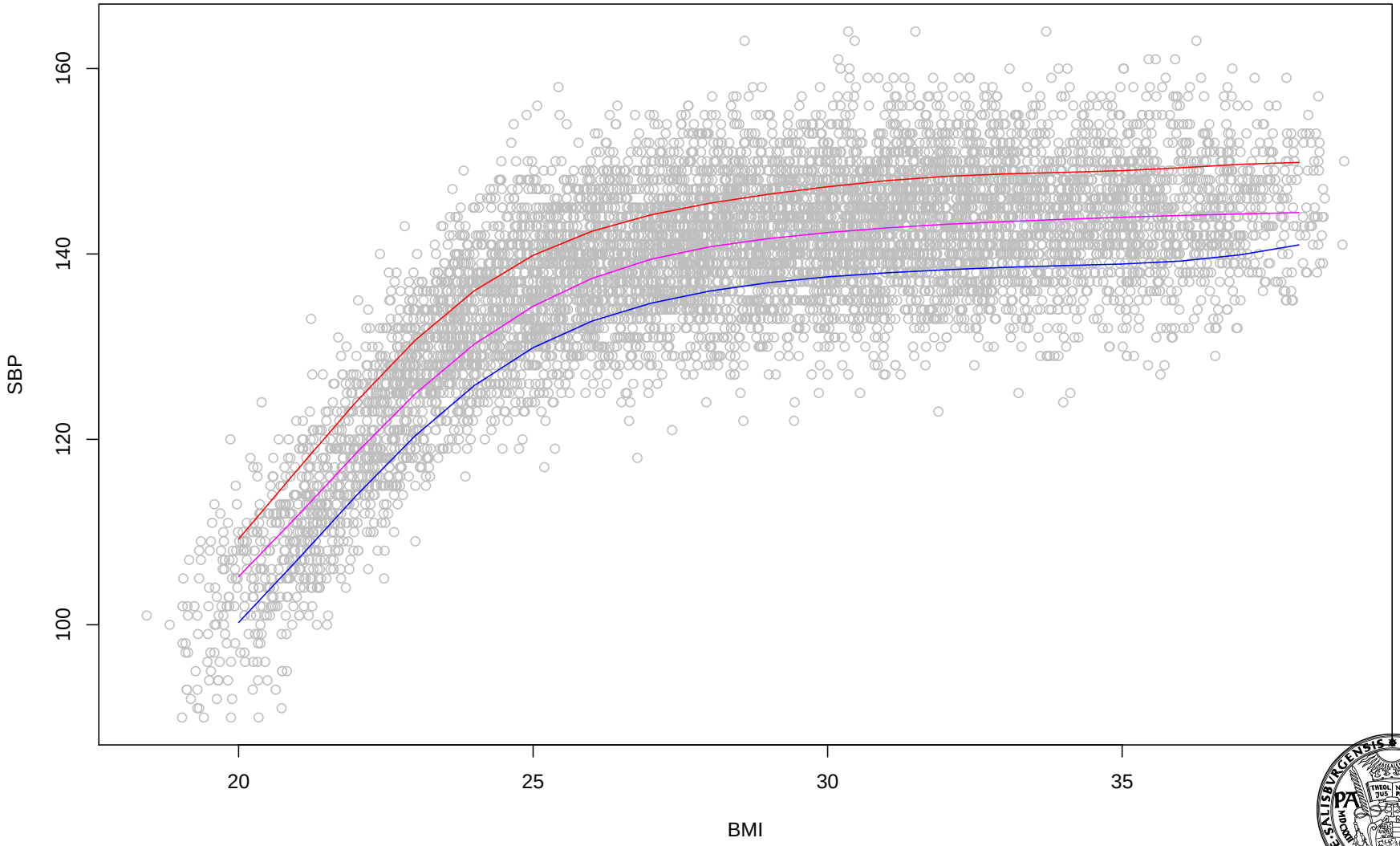
Anpassung (nichtlinearer) parametrischer Modelle



Beispiel (Fitting via `optim`, SBP versus BMI und age)

- Der (generierte) Datensatz `SBP.RData` enthält für ca. 8000 Patienten die folgenden Informationen: Alter, BMI (Mody Mass Index), SBP (Systolic blood pressure)
- Modell $SBP = r(BMI, age) + \varepsilon$, die Regressionsfunktion r soll geschätzt werden.
- Als ersten Schritt betrachten wir drei Gruppen:
  (i) Alter zwischen 30 und 40
  (ii) Alter zwischen 50 und 60
  (iii) Alter zwischen 70 und 80,
  und berechnen für jede der Gruppen den Kernregressionsschätzer.
- Details siehe `R-codes_GEO_13.R`
- Liefert die nachfolgende Grafik

Anpassung (nichtlinearer) parametrischer Modelle



Beispiel (Fitting via `optim`, SBP versus BMI und age)

- Wir passen ein Regressionsmodell der Form

$$r(BMI, age) = 105 + a \cdot age + b \cdot atan(c \cdot BMI + d) + \varepsilon$$

an die Daten an.

```
f <- function(v){
  predict<- 105+v[1]*A$age + v[2]*atan(v[3]*A$BMI+v[4])
  r<-sum((A$SBP-predict)^2,na.rm=TRUE)
  return(r)
}

res<-optim(par=c(0,10,0,10), fn=f)
res
```

- Liefert

```
$par
[1]   0.2479021  18.0918060   0.4976221 -10.9334087
```

- Berechnen und plotten Regressionsfunktion für age=35 und beliebige BMI-Werte

Anpassung (nichtlinearer) parametrischer Modelle