# Nichtlineare Effekte in der linearen Regression

## Splines

Jan-Philipp Kolb

Freitag, 20.06.2014

# Einleitung

## Polynome

- **Vorteil:** Glättung wird durchgeführt
  $\rightarrow$ Gute Anpassung an Daten
- **Nachteil:** jeder Punkt beeinflusst den Fit

## Segmented regressions

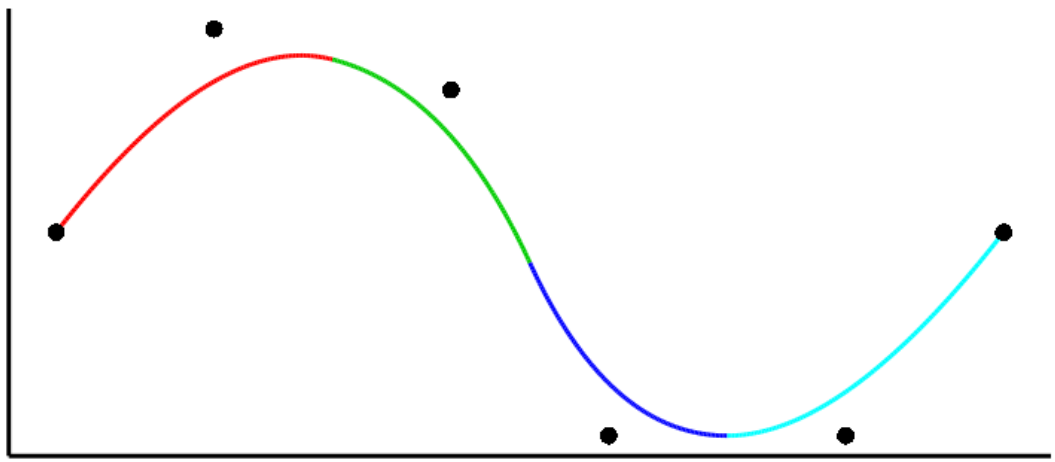
- Einfluss eines Punktes wird auf bestimmten Bereich begrenzt.
- Aber der Fit ist nicht ganz so gut.

Mit **Splines** können die Vor- und Nachteile dieser beiden Verfahren miteinander in Einklang gebracht werden.

# Was sind Splines?

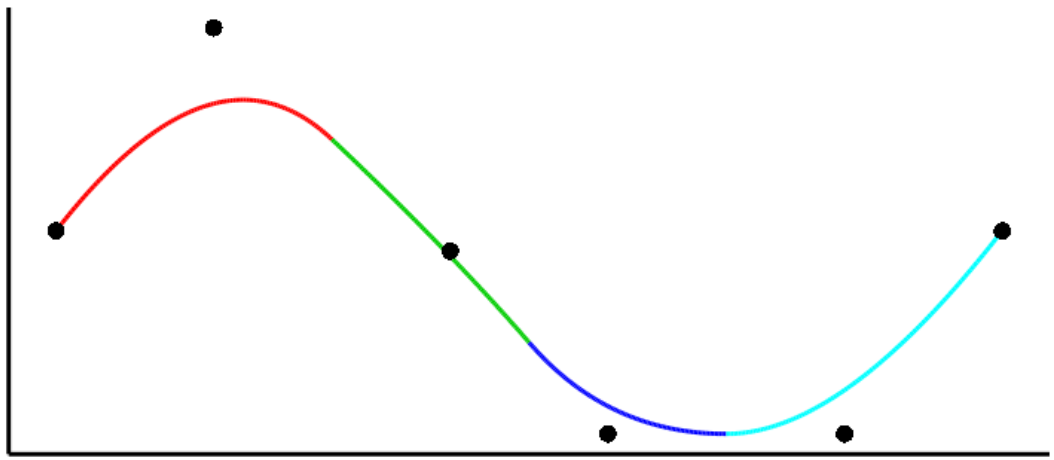
- Ein Spline Grades ist eine Funktion, die stückweise aus Polynomen zusammengesetzt ist.
- Bei einem Spline *n*-ten Grades haben die Polynome höchstens den Grad *n*
- Ein B-Spline ist ein *Basis*-Spline
- Knoten sind die Stellen, an denen zwei Polynomstücke zusammenstoßen

# Was sind Splines?



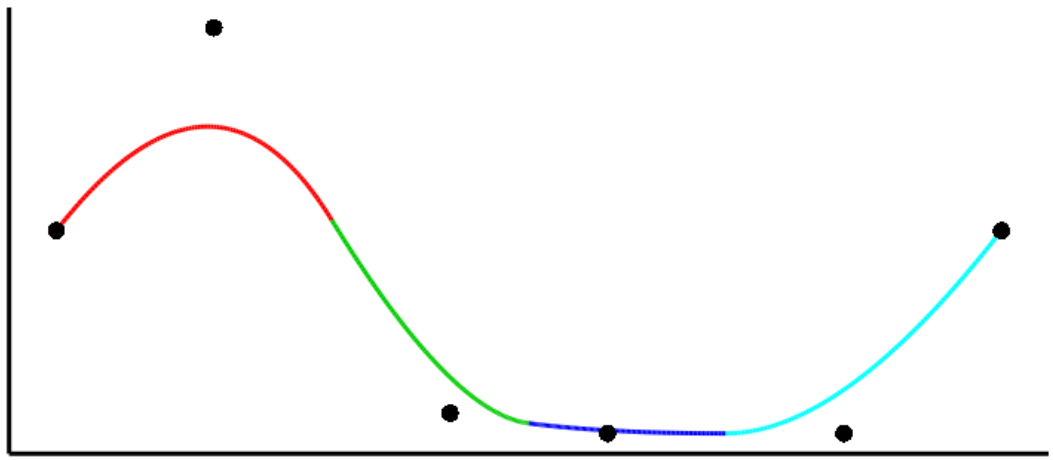
https://team.inria.fr/imagine/pierre-luc-manteaux/

# Was sind Splines?



https://team.inria.fr/imagine/pierre-luc-manteaux/

# Was sind Splines?



https://team.inria.fr/imagine/pierre-luc-manteaux/

# Splines - Umsetzung in R

Die Bibliothek `splines` ist eine von vielen Bibliotheken, mit der Splines berechnet werden können

```
library(splines)
```

Datensatz einlesen:

```
li<-"http://data.princeton.edu/wws509/datasets/effort.dat"
fpe <- read.table(li)
```

So werden die Splines berechnet:

```
# Basic Spline
setting.bs <- bs(setting, knots = c(66,74,84) )
# natural cubic splines
setting.ns <- ns(setting, df=5)
```

# Splines - Beispieldaten

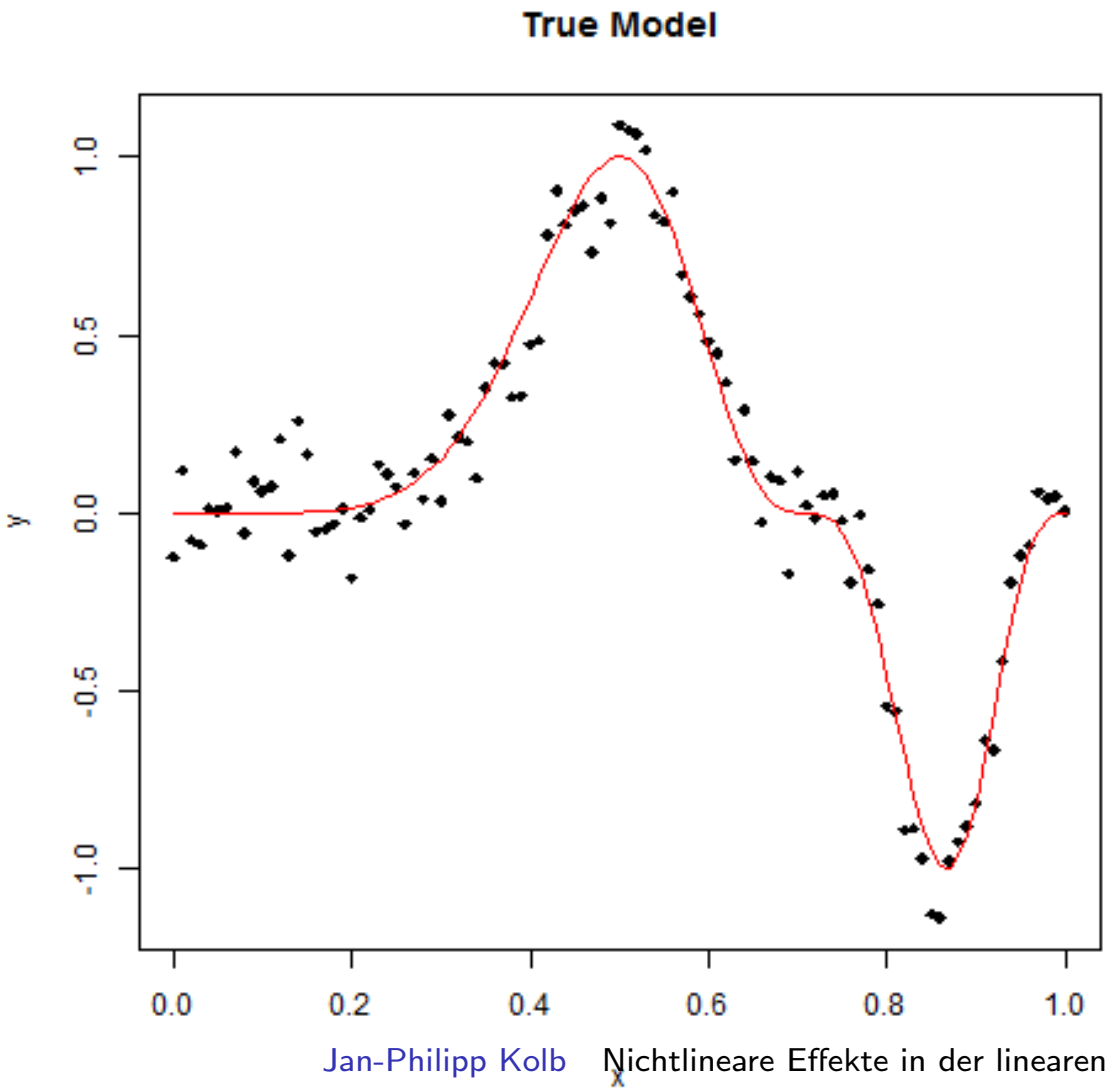
- Wir erzeugen synthetische Daten
- Diese Daten haben den Vorteil, dass wir überprüfen können wie nah die Methoden der Wahrheit kommen.

```
funky  <- function(x) sin(2*pi*x^2)^3
x <- seq(0,1,0.01)
y <- funky(x)+0.1*rnomr(101)
```

- Die erzeugten Daten werden graphisch dargestellt:

```
%par(mfrow=c(2,2))
matplot(x,cbind(y,funky(x)),type="pl",ylab="y",pch=18,
lty=1,main="True Model")
```

# Splines - Beispieldaten



# Polynomiale Regression auf den Daten

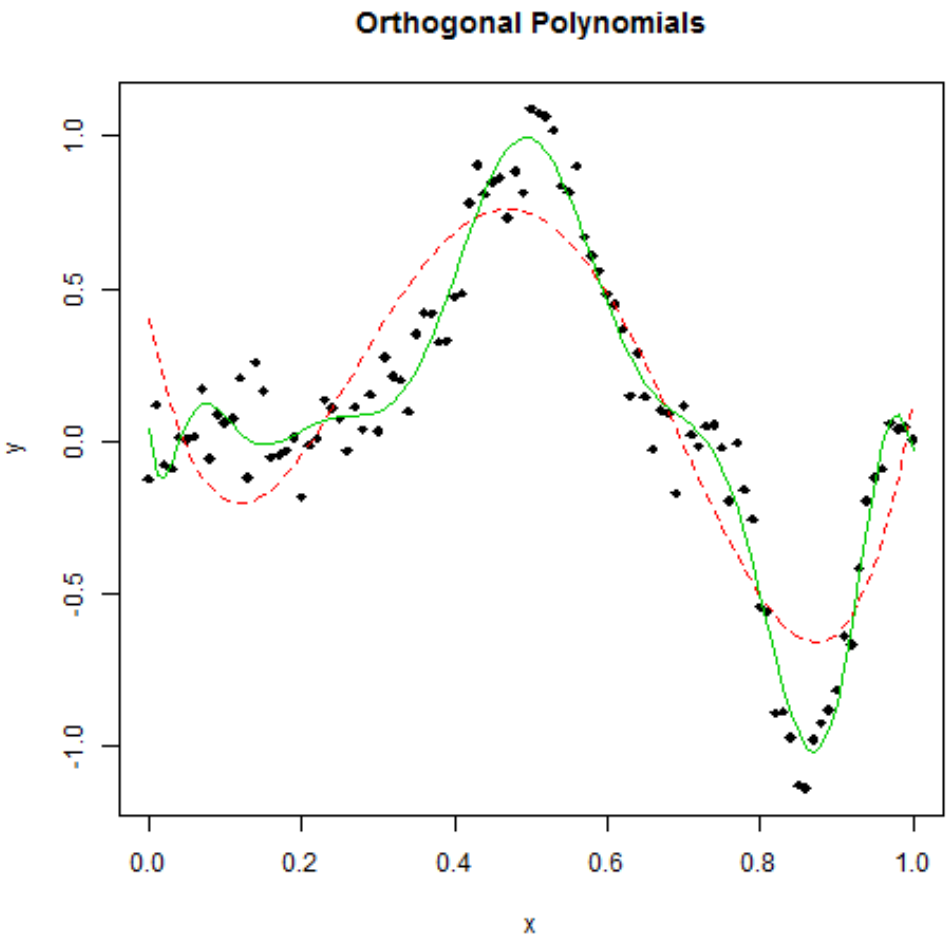
Zwei verschiedene Polynomiale Regressionen:

```
g4 <- lm(y~poly(x,4))
g12 <- lm(y~poly(x,12))
```

Und der entsprechende Plot dazu:

```
matplot(x,cbind(y,g4$fit,g12$fit),type="pll",ylab="y",
pch=18,lty=c(1,2),main="Orthogonal Polynomials")
```

# Polynomiale Regression auf den Daten




Polynom vierten Grades
Polynom zwölften Grades

# Splines

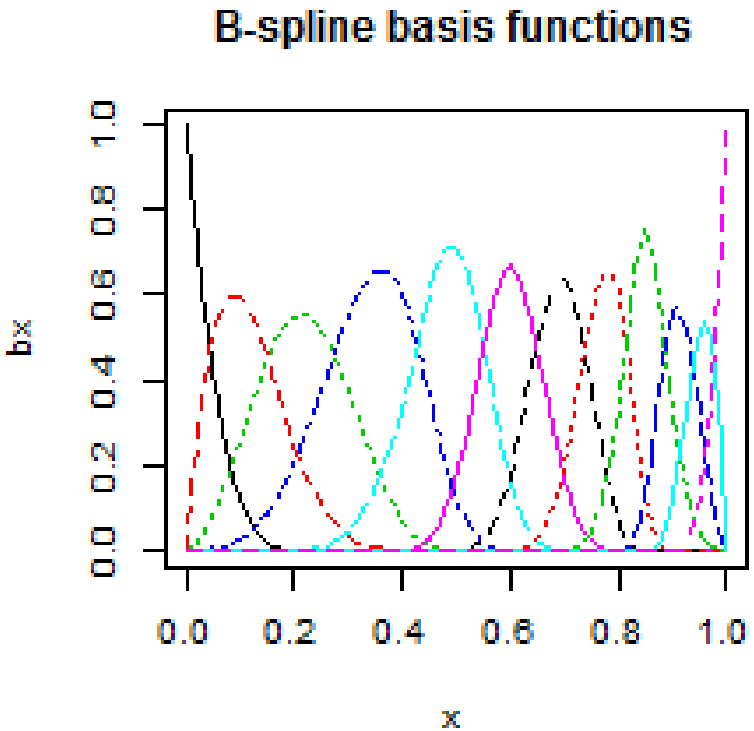
- Die Spline-Regression mit dem Paket splines
- Zunächst müssen Knoten definiert werden
- `splineDesign` erzeugt ein entsprechendes Objekt

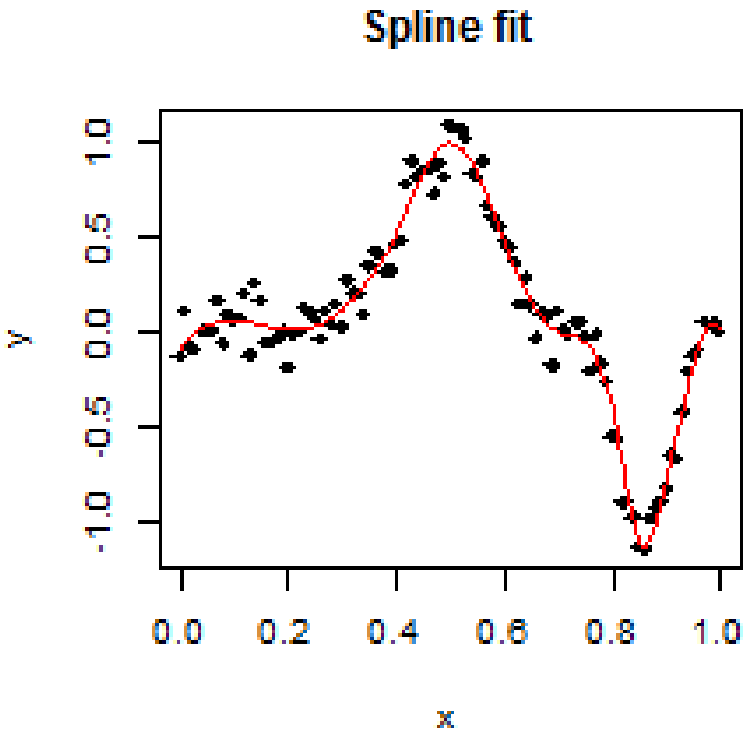
```
library(splines)
knots <- c(0,0,0,0,0.2,0.4,0.5,0.6,0.7,0.8,0.85,0.9,
1,1,1,1)
bx <- splineDesign(knots,x)
gs <- lm(y~bx)
```

Das Ergebnis wieder plotten:

```
matplot(x,bx,type="l",main="B-spline basis functions")
matplot(x,cbind(y,gs$fit),type="pl",ylab="y",pch=18,lty=1,
main="Spline fit")
```

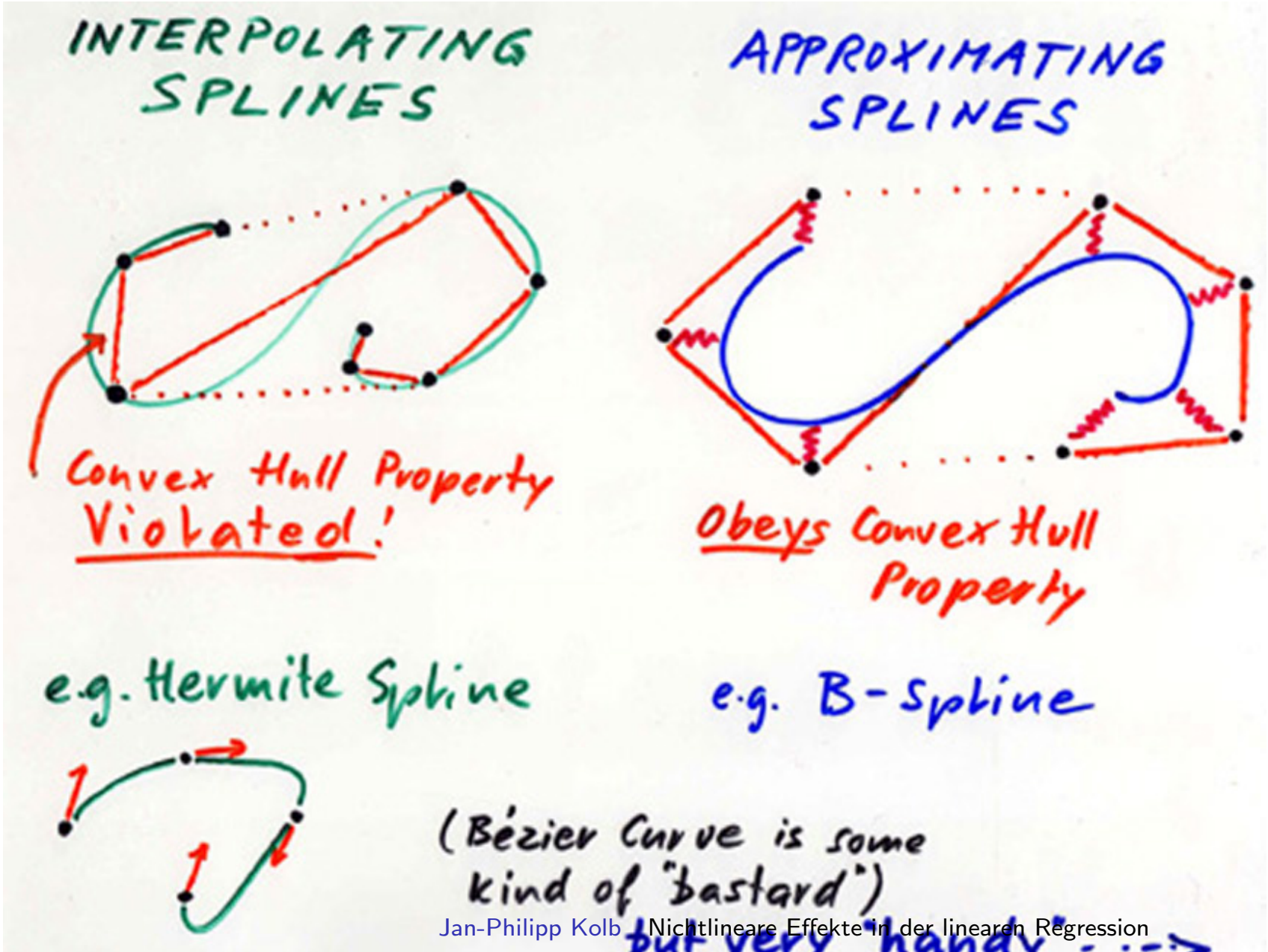
# Spline-Regression auf den Daten



# Splines - Linkliste

- http://www.r-bloggers.com/non-linear-regression-in-r/
- http://www.r-bloggers.com/r-for-ecologists-simulating-species-area-curves-linear
- http://www.r-bloggers.com/some-heuristics-about-local-regression-and-kernel-smoo
- http://data.princeton.edu/R/introducingR.pdf


INTERPOLATING SPLINES
APPROXIMATING SPLINES
Convex Hull Property Violated!
Obeys Convex Hull Property
e.g. Hermite Spline
e.g. B-Spline
(Bézier Curve is some kind of "bastard")