Tutorial 5

Lars Abt

Aufgabe 1: Nadaraya-Watson

Wir laden unsere Pakete mit Hilfe von 'pacman'.

```
pacman::p_load(data.table, ggplot2)
```

Nun laden wir unsere Daten herunter, über den angegebenen Link:

```
miete <- fread("https://tinyurl.com/yn9ttfcu", header = TRUE)
miete <- miete[miete$wfl <= 200, ]
```

Letztlich verschaffen wir uns einen kleinen Überblick über die Daten.

```
# Schauen wir uns die Struktur der Daten an
str(miete)
```

```
Classes 'data.table' and 'data.frame':  3061 obs. of  13 variables:
 $ nm      : num  608 780 823 500 595 ...
 $ nmqm    : num  12.67 13 7.48 8.62 8.5 ...
 $ wfl     : int  48 60 110 58 70 81 97 50 71 76 ...
 $ rooms   : int  2 2 5 2 3 3 3 2 3 3 ...
 $ bj      : num  1958 1983 1958 1958 1972 ...
 $ bez     : chr  "Untergiesing" "Bogenhausen" "Obergiesing" "Schwanthalerhöhe" ...
 $ wohngut : int  0 1 0 0 0 0 1 1 0 0 ...
 $ wohnbest: int  0 0 0 0 0 0 0 0 0 0 ...
 $ ww0     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ zh0     : int  0 0 1 0 0 0 0 0 0 1 ...
 $ badkach0: int  1 1 1 1 0 1 1 0 1 1 ...
 $ badextra: int  0 0 1 0 0 0 1 0 0 1 ...
 $ kueche  : int  0 1 0 1 0 0 1 1 0 0 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

```r
# Betrachten wir die ersten 6 Zeilen
head(miete)
```

```
      nm  nmqm   wfl rooms     bj              bez wohngut wohnbest   ww0   zh0
   <num> <num> <int> <int>  <num>           <char>   <int>    <int> <int> <int>
1: 608.4 12.67    48     2 1957.5     Untergiesing       0        0     0     0
2: 780.0 13.00    60     2 1983.0      Bogenhausen       1        0     0     0
3: 822.6  7.48   110     5 1957.5      Obergiesing       0        0     0     1
4: 500.0  8.62    58     2 1957.5 Schwanthalerhöhe       0        0     0     0
5: 595.0  8.50    70     3 1972.0        Aubing...       0        0     0     0
6: 960.0 11.85    81     3 2006.5 Schwanthalerhöhe       0        0     0     0
   badkach0 badextra kueche
      <int>    <int>  <int>
1:        1        0      0
2:        1        0      1
3:        1        1      0
4:        1        0      1
5:        0        0      0
6:        1        0      0
```

Aufgabe 1a

Nun erstellen wir ein Streudiagramm der Wohnfläche (wfl) gegen den Nettomietpreis pro Quadratmeter (nmqm).

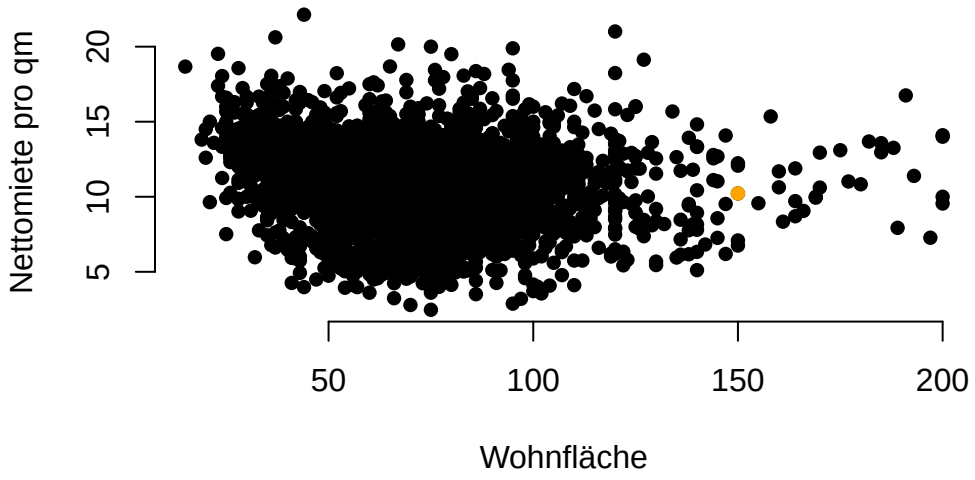
```r
# Ein Streudiagramm mit Hilfe von R-Base:
plot(miete$wfl, miete$nmqm, pch = 16, bty = "n",
xlab = "Wohnfläche", ylab = "Nettomiete pro qm")

# Die Werte von "wfl" aus dem Intervall [140,160] identifizieren, da uns die Stelle wfl = 150
ind <- which(140 <= miete$wfl & miete$wfl <= 160)

# Werte der Zielvariable "nm" auf dem Intervall:
head(miete$nmqm[ind])
```

```
[1]  8.93  8.16  5.11  6.75 12.69 10.22
```

```r
# Dort den lokalen Mittelwert berechnen:
m150 <- mean(miete$nmqm[ind])
m150
```

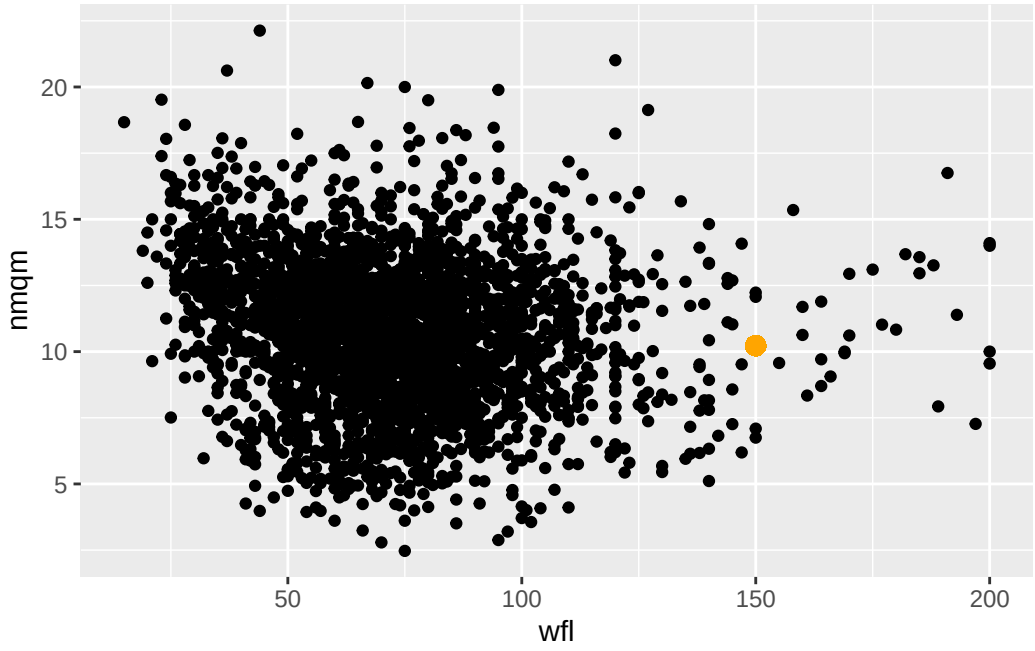
```
[1] 10.22276
```

```
# Und einzeichnen:
points(150, m150, col = "orange", pch = 16)
```




```
# Jetzt noch einmal mit ggplot2:
ggplot(miete, aes(x = wfl, y = nmqm)) +
  geom_point() +
  geom_point(aes(x = 150, y = m150), color = "orange", size = 3)
```

```
Warning in geom_point(aes(x = 150, y = m150), color = "orange", size = 3): All aesthetics hav
i Please consider using `annotate()` or provide this layer with data containing
  a single row.
```




```
  labs(x = "Wohnfläche", y = "Nettomiete pro qm") +
  theme_minimal()
```

```
NULL
```

Kurze Erklärung zu unserem Vorgehen: Wir betrachten lediglich die Werte zwischen 140 und 160, da der Rechteckkern lediglich Werte in Betracht zieht, welche für den Ausdruck (x-xi)/b kleiner als 0.5 sind. Dies ist der Fall, wenn x in [x-b/2, x+b/2] liegt. Für den Punkt xi = 150 und b = 20 ergibt sich also das Intervall [140,160]. Somit schauen wir zuerst, welche Wohnungen eine Wohnfläche in diesem Bereich aufweisen, und bilden anschließend den Mittelwert des Nettopreises pro Quadratmeter über genau diese Werte.

Aufgabe 1b

Wir übernehmen die Funktion von unserem Übungsblatt:

```
K <- function(z){ifelse(abs(z)>0.5, 0, 1)}
```

Nun probieren wir diese Funktion für einige Werte aus:

```r
K(10)
```

```
[1] 0
```

```r
K(2)
```

```
[1] 0
```

```r
K(0.1)
```

```
[1] 1
```

```r
K(0.5)
```

```
[1] 1
```

Nun berechnen wir den Nadaraya-Watson-Schätzer für die Stelle wfl = 150:

```r
b <- 20 # Bandweite definieren

sum(K((150 - miete$wfl)/b) * miete$nmqm) /
sum(K((150 - miete$wfl)/b))
```

```
[1] 10.22276
```

Wir sehen, dass der Nadaraya-Watson-Schätzer für die Stelle wfl = 150 den Wert 10.22276 berechnet.

Aufgabe 1c

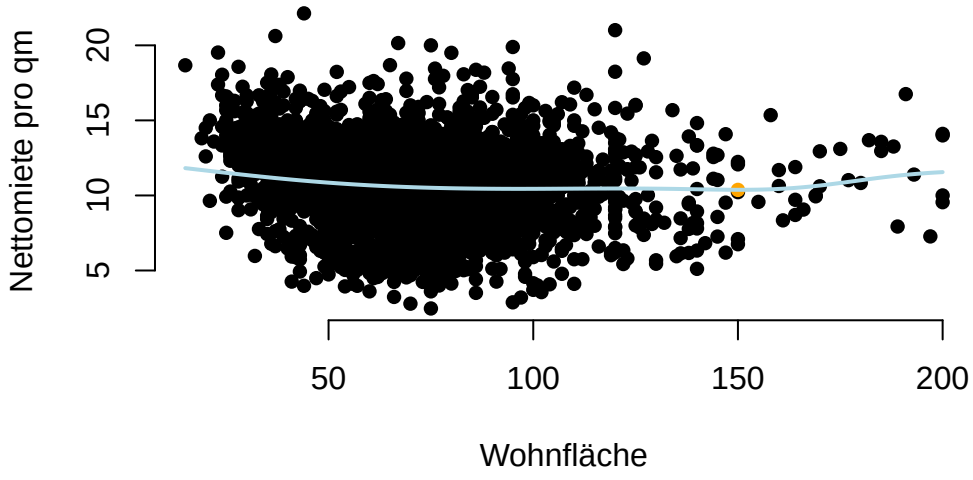
Nun betrachten wir den Gaußkern (statt des Rechteckskerns), und betrachten unseren Plot, ergänzt durch unsere Schätzungen.

```r
b <- 20 # Bandweite definieren

m150norm <- sum(dnorm((150 - miete$wfl) / b) * miete$nmqm) /
sum(dnorm((150 - miete$wfl) / b))
m150norm
```

```
[1] 10.36849
```

Mit dem Gaußkern ergibt sich ein Wert von 10,36849, welcher nahe an unserem anderen berechneten Wert befindet. Nun möchten wir unseren Schätzer auch noch einzeichnen in unser Streudiagramm:

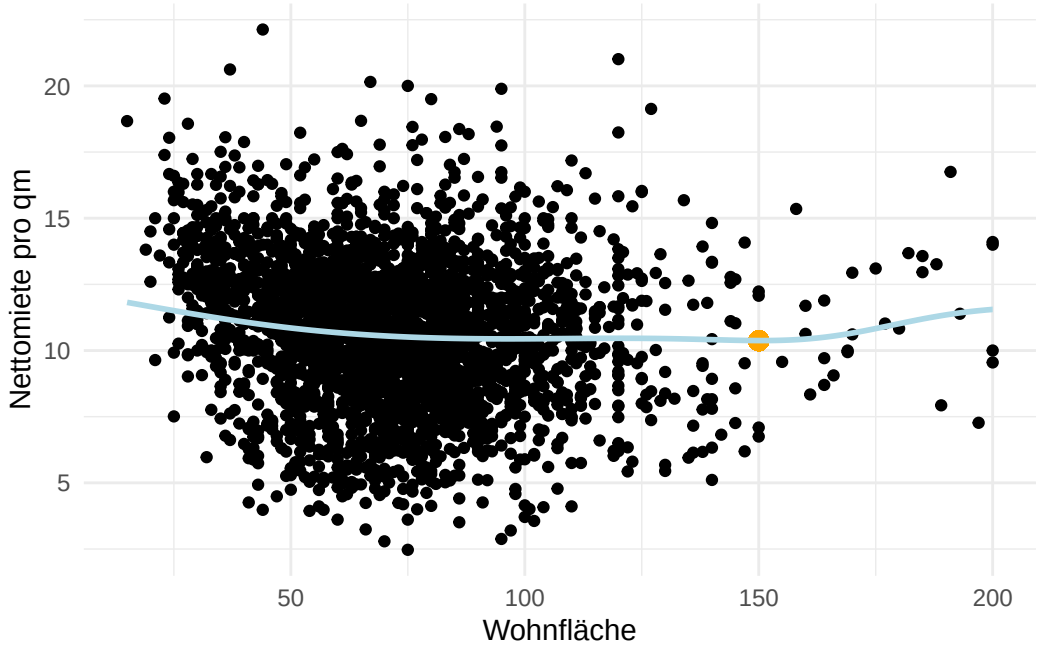
```r
# Mit R-Base:
plot(miete$wfl, miete$nmqm, pch = 16, bty = "n",
     xlab = "Wohnfläche", ylab = "Nettomiete pro qm")

points(150, m150norm, col = "orange", pch = 16)

# Nadaraya Watson Schätzung
NWS <- ksmooth(miete$wfl, miete$nmqm,
               kernel = "normal", bandwidth = 55)
# ?ksmooth

lines(NWS$x,NWS$y,col="lightblue",lwd=2)
```



Und nun noch einmal mit ggplot2:

```r
# Vorbereitung:
NWS <- ksmooth(miete$wfl, miete$nmqm, kernel = "normal", bandwidth = 55)
NWS_data <- data.table(x = NWS$x, y = NWS$y)

# Plot:
ggplot(miete, aes(x = wfl, y = nmqm)) +
  geom_point() +
  geom_point(aes(x = 150, y = m150norm), color = "orange", size = 3) +
  geom_line(data = NWS_data, aes(x = x, y = y),
            color = "lightblue", linewidth = 1) +
  labs(x = "Wohnfläche", y = "Nettomiete pro qm") +
  theme_minimal()
```

```
Warning in geom_point(aes(x = 150, y = m150norm), color = "orange", size = 3): All aesthetics
i Please consider using `annotate()` or provide this layer with data containing
  a single row.
```

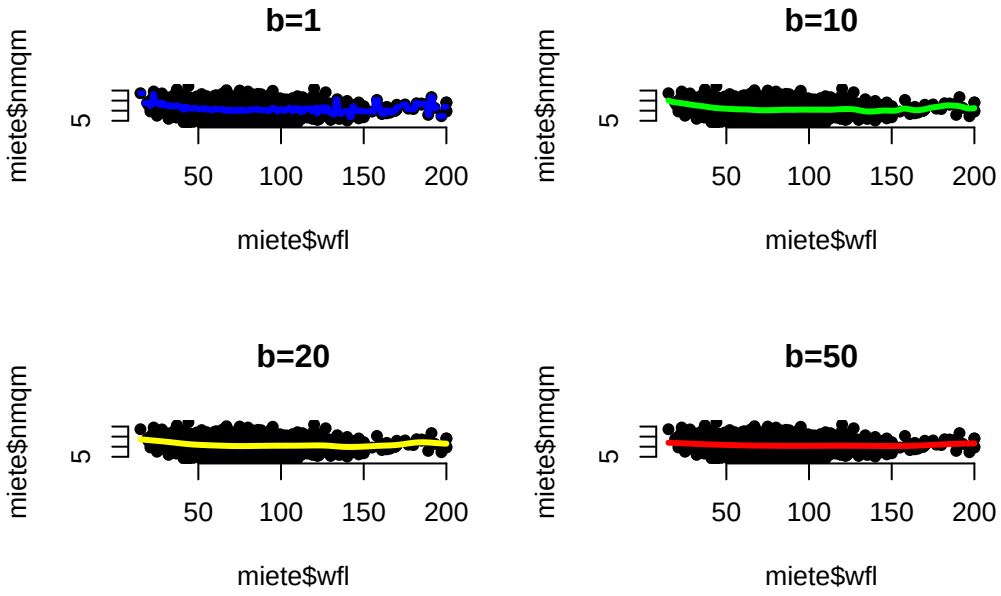


Aufgabe 2: Bandweitenwahl & Kreuzvalidierung

Aufgabe 2a

Nun möchten wir die Bandweite b variieren und uns die Auswirkungen auf den Schätzer anschauen. Wir verwenden dazu die Funktion ksmooth().

```
# So können wir mehrere Plots gemeinsam betrachten:
par(mfrow = c(2, 2))

# Klassisch mit RBase:
plot(miete$wfl, miete$nmqm, pch = 16, bty = "n", main = "b=1")
NWS <- ksmooth(miete$wfl, miete$nmqm, kernel = "normal", bandwidth = 1)
lines(NWS$x, NWS$y, col = "blue", lwd = 3)

plot(miete$wfl, miete$nmqm, pch = 16, bty = "n", main = "b=10")
NWS <- ksmooth(miete$wfl, miete$nmqm, kernel = "normal", bandwidth = 10)
lines(NWS$x, NWS$y, col = "green", lwd = 3)

plot(miete$wfl, miete$nmqm, pch = 16, bty = "n", main = "b=20")
NWS <- ksmooth(miete$wfl, miete$nmqm, kernel = "normal", bandwidth = 20)
lines(NWS$x, NWS$y, col = "yellow", lwd = 3)

plot(miete$wfl, miete$nmqm, pch = 16, bty = "n", main = "b=50")
NWS<-ksmooth(miete$wfl, miete$nmqm, kernel = "normal", bandwidth = 50)
lines(NWS$x, NWS$y, col = "red", lwd = 3)
```

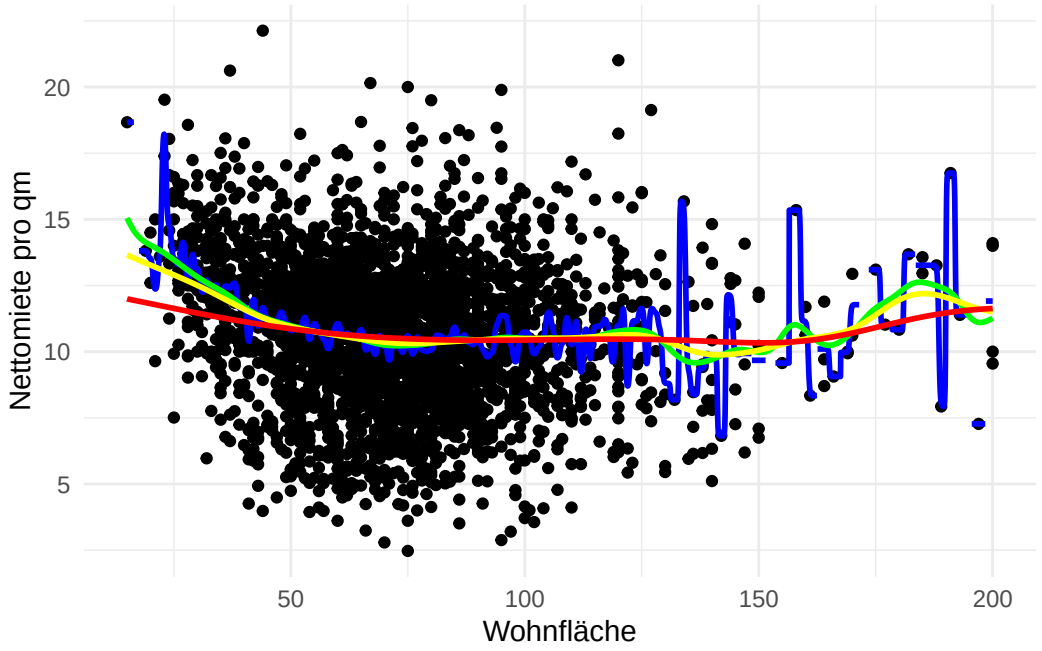


Wir sehen, dass eine Bandweite im Bereich von 10 bis 20 eine gute Wahl scheint.

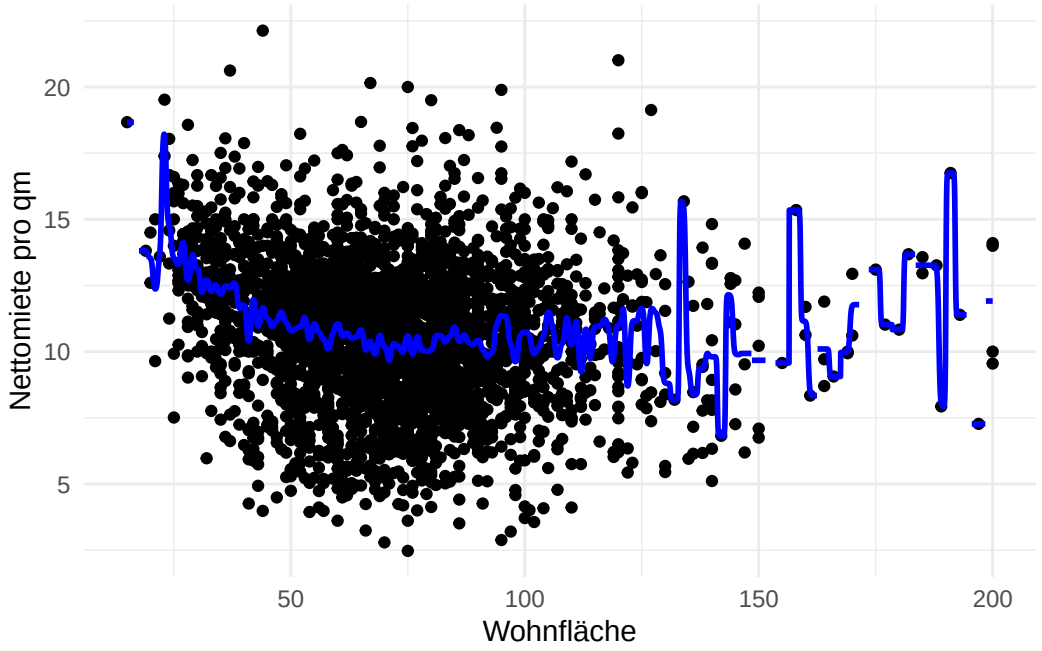
Jetzt schauen wir auch auf die Darstellung mit Hilfe von ggplot2:

```
# Wir bereiten unsere Daten für's plotten vor:
NWS_1 <- ksmooth(miete$wfl, miete$nmqm, kernel = "normal", bandwidth = 1)
NWS_bw_1 <- data.table(x = NWS_1$x, y = NWS_1$y)

NWS_10 <- ksmooth(miete$wfl, miete$nmqm, kernel = "normal", bandwidth = 10)
NWS_bw_10 <- data.table(x = NWS_10$x, y = NWS_10$y)

NWS_20 <- ksmooth(miete$wfl, miete$nmqm, kernel = "normal", bandwidth = 20)
NWS_bw_20 <- data.table(x = NWS_20$x, y = NWS_20$y)

NWS_50 <- ksmooth(miete$wfl, miete$nmqm, kernel = "normal", bandwidth = 50)
NWS_bw_50 <- data.table(x = NWS_50$x, y = NWS_50$y)

# Und plotten anschließend unsere verschiedenen Bandweiten:
ggplot(miete, aes(x = wfl, y = nmqm)) +
  geom_point() +
  geom_line(data = NWS_bw_1, aes(x = x, y = y), color = "blue", linewidth = 1) +
  geom_line(data = NWS_bw_10, aes(x = x, y = y), color = "green", linewidth = 1) +
  geom_line(data = NWS_bw_20, aes(x = x, y = y), color = "yellow", linewidth = 1) +
  geom_line(data = NWS_bw_50, aes(x = x, y = y), color = "red", linewidth = 1) +
```

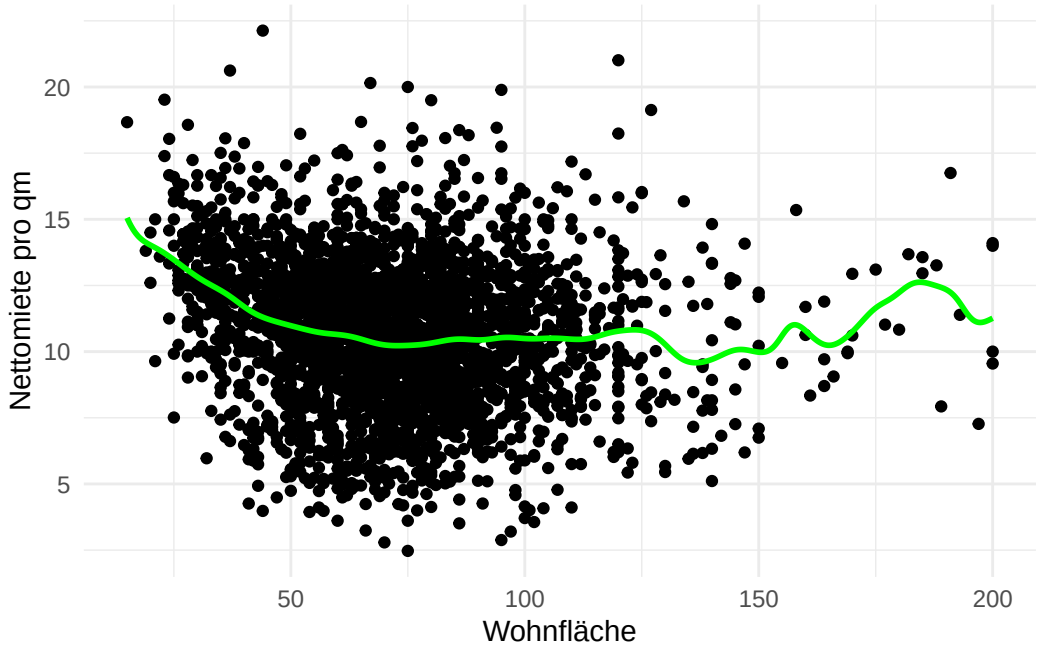
```
  labs(x = "Wohnfläche", y = "Nettomiete pro qm", main = "Bandweitenvergleich") +
  theme_minimal()
```



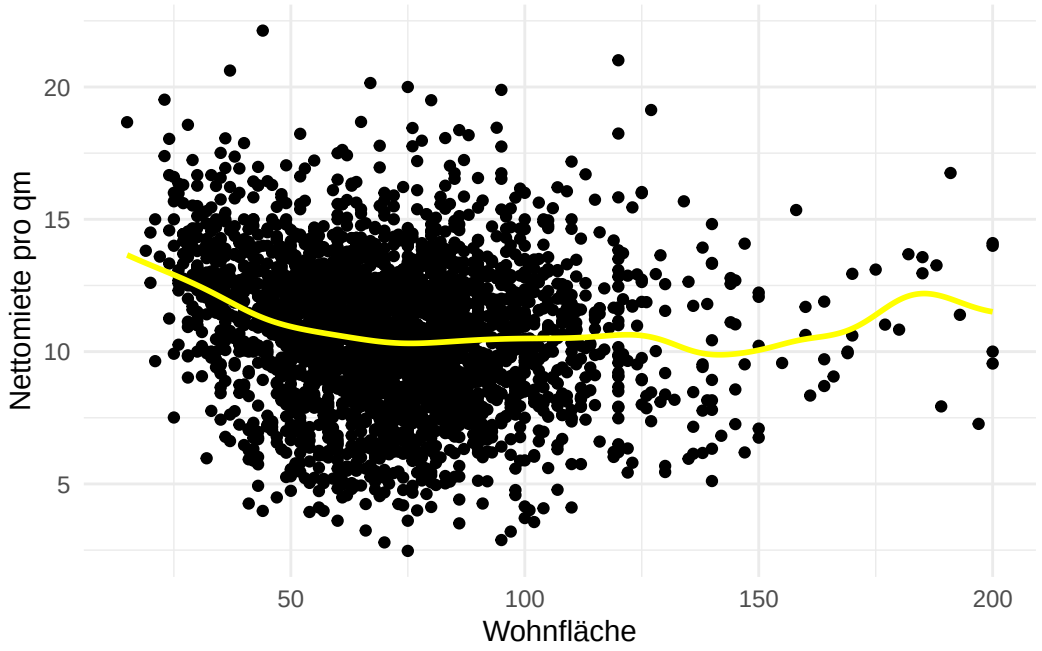

```
# Oder als einzelne Plots:
ggplot(miete, aes(x = wfl, y = nmqm)) +
  geom_point() +
  geom_line(data = NWS_bw_1, aes(x = x, y = y), color = "blue", linewidth = 1) +
  labs(x = "Wohnfläche", y = "Nettomiete pro qm", main = "Bandweite 1") +
  theme_minimal()
```



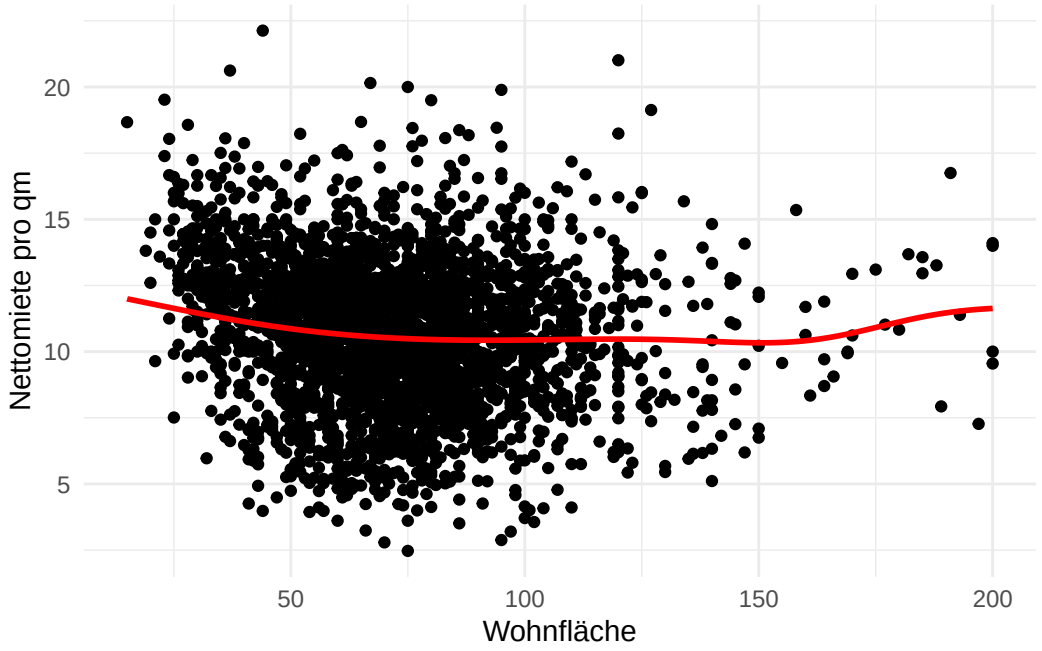

```
ggplot(miete, aes(x = wfl, y = nmqm)) +
  geom_point() +
  geom_line(data = NWS_bw_10, aes(x = x, y = y), color = "green", linewidth = 1) +
  labs(x = "Wohnfläche", y = "Nettomiete pro qm", main = "Bandweite 10") +
  theme_minimal()
```




```
ggplot(miete, aes(x = wfl, y = nmqm)) +
  geom_point() +
  geom_line(data = NWS_bw_20, aes(x = x, y = y), color = "yellow", linewidth = 1) +
  labs(x = "Wohnfläche", y = "Nettomiete pro qm", main = "Bandweite 20") +
  theme_minimal()
```




```
ggplot(miete, aes(x = wfl, y = nmqm)) +
  geom_point() +
  geom_line(data = NWS_bw_50, aes(x = x, y = y), color = "red", linewidth = 1) +
  labs(x = "Wohnfläche", y = "Nettomiete pro qm", main = "Bandweite 50") +
  theme_minimal()
```



Aufgabe 2b

Nun möchten wir uns mit dem Thema der Kreuzvalidierung beschäftigen. Wir beginnen dafür mit dem Kreuzvalidierungskriterium für i = 1.

```
# Berechnen des Kreuzvalidierungskriteriums für i = 1
mcv <- ksmooth(miete$wfl[-1], miete$nmqm[-1],
               kernel = "normal",
               bandwidth = 10,
               x.points = miete$wfl[1])$y

# Berechnen der quadrierten Abweichung
(miete$nmqm[1] - mcv)^2
```

```
[1] 2.561681
```

Nun nutzen wir eine For-Schleife, um CV(10) zu berechnen:

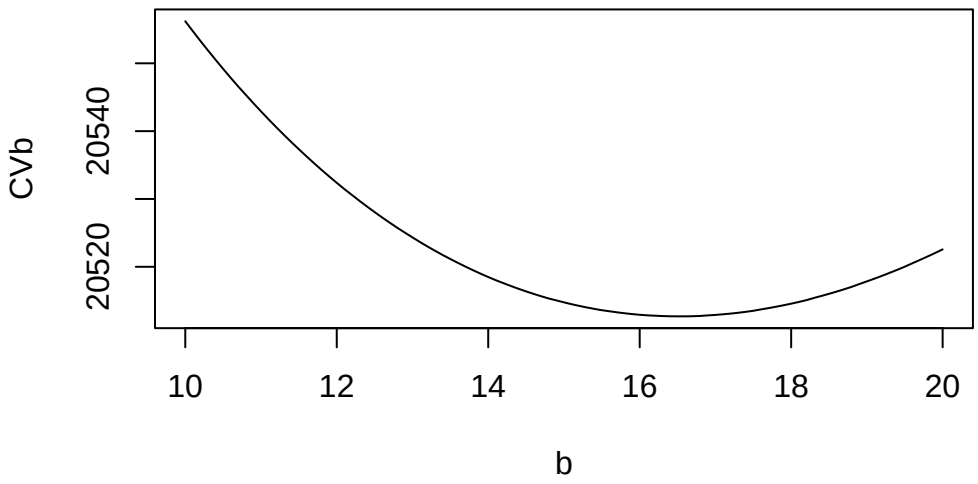
```
# Initialisieren unserer Variablen
n <- nrow(miete)
CV <- rep(NA, n)
```

```r
# For-Schleife zur Berechnung
for(i in 1:n){
  mcv <- ksmooth(miete$wfl[-i], miete$nmqm[-i],
                 kernel = "normal",
                 bandwidth = 10,
                 x.points = miete$wfl[i])$y
  CV[i] <- (miete$nmqm[i] - mcv)^2
}

# Werte der Kreuzvalidierung
sum(CV)
```

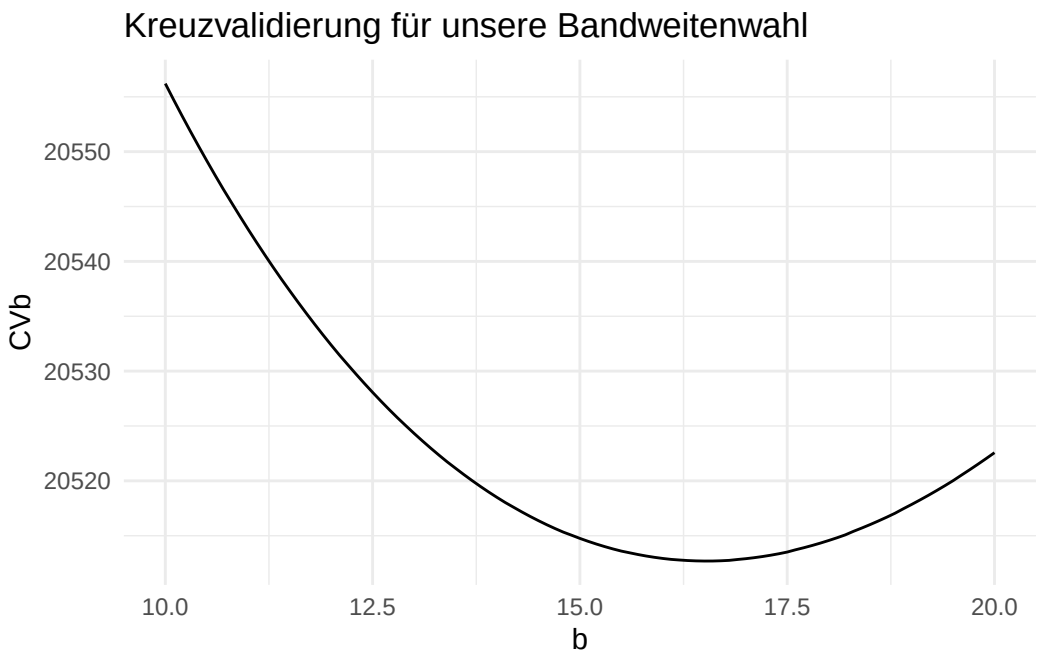
```
[1] 20556.2
```

Nun wollen wir noch die Bandweite variieren, zwischen 10 und 20, in Schritten von 0,1.

```r
# Initialisieren unserer Variablen
b <- seq(10, 20, by = 0.1)
nb <- length(b)
n <- nrow(miete)
CVb <- rep(NA,nb)

# For-Schleife zur Berechnung (etwas zeitintensiv)
for(k in 1:nb){     # äußere For-Schleife für verschiedene Werte von b
  CV <- rep(NA, n)
  for (i in 1:n){   # innere For-Schleife für die Kreuzvalidierung
    mcv <- ksmooth(miete$wfl[-i], miete$nmqm[-i],
                   kernel = "normal",
                   bandwidth = b[k],
                   x.points = miete$wfl[i])$y
    CV[i] <- (miete$nmqm[i] - mcv)^2
  }
  CVb[k] <- sum(CV)
}

# Nun plotten wir unsere Ergebnisse:
par(mfrow = c(1, 1)) # um das Fenster anzupassen
plot(b, CVb, type = "l")
```




```
# Oder mit ggplot2:
# Daten in ein data.table bringen
cv_data <- data.table(b = b, CVb = CVb)

# Linienplot mit ggplot2
ggplot(cv_data, aes(x = b, y = CVb)) +
  geom_line() +
  theme_minimal() +
  labs(x = "b", y = "CVb", title = "Kreuzvalidierung für unsere Bandweitenwahl")
```



Jetzt möchten wir noch einmal exakt nachschauen, an welcher Stelle das Kreuzvalidierungskriterium den niedrigsten Wert aufweist, und welcher das ist.

```
# An welcher Stelle hat der Vektor CVb seinen kleinsten Wert (Index)?
which.min(CVb)
```

```
[1] 66
```

```
# Was ist dieser kleinste Wert (welches b hat den besten CV-Wert)?
b[which.min(CVb)]
```

```
[1] 16.5
```

Wir sehen, das der niedrigste CV Wert bei 66 liegt, und dieser zur Bandweite 16.5 gehört. Dies passt zu unserem Plot, und unserer Vermutung in Aufgabe 1.

Aufgabe 2c

Zum Abschluss werfen wir noch einen kurzen Blick auf den Bias-Varianz-Trade-off. Das Kreuzvalidierungskriterium gibt uns diesbezüglich klare Anhaltspunkte. Der Wert von CV ist hoch für (zu) kleines b, da wir dann eine zu große Varianz haben. Analog dazu ist der Wert von CV

auch hoch für (zu) große b, da wir dann einen hohen Bias haben. Ein mittlerer Wert von b scheint also optimal zu sein, und stellt einen Kompromiss zwischen Varianz (Überanpassung) und Bias (Unteranpassung) dar, welchen auch das Kreuzvalidierungskriterium bevorzugt.

Wenn wir das mit unserem Plot der CV-Werte vergleichen möchten, sehen wir das Tal (den Tiefpunkt) des Graphen als den Kompromiss, welchen das Kreuzvalidierungskriterium als optimalen Wert für b vorschlägt. Dies ist der Punkt, an dem der Bias und die Varianz in einem optimalen Verhältnis zueinander stehen. Links davon, bei kleinerer Bandweite, sehen wir die Gefahr von Overfitting, also einer zu hohen Varianz. Die Auswirkungen dieser geringen Bandweite (eine sehr sprunghafte Linie) haben wir in einer vorangehenden Aufgabe bereits beleuchtet. Rechts vom Tiefpunkt sehen wir die Gefahr von Underfitting, also einer zu hohen Bias. Auch hier haben wir bereits in einer vorangehenden Aufgabe die Auswirkungen (eine zu flache Linie) beleuchtet.