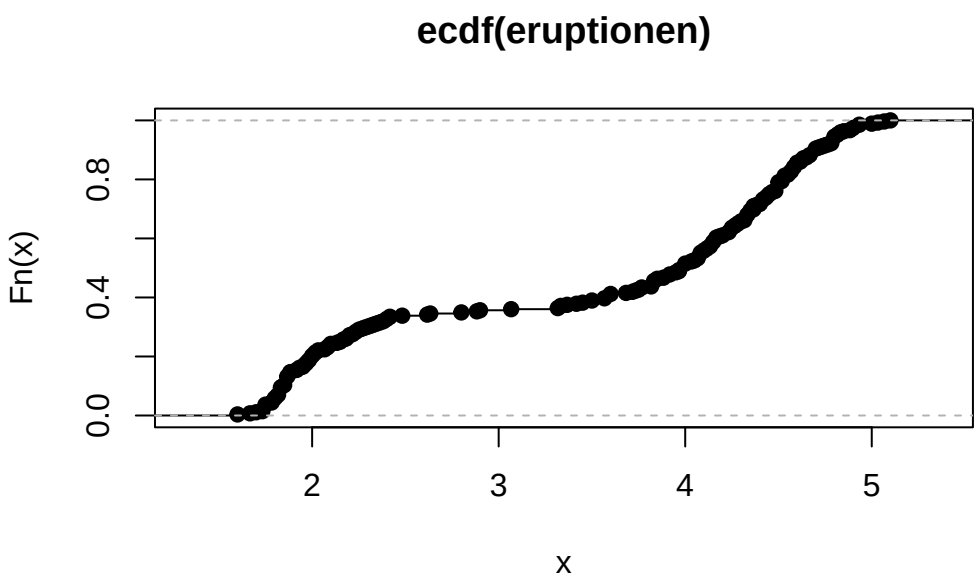
Tutorial 2

Lars Abt

Aufgabe 1: ECDF und Quantile

Wir laden die Daten zum Old Faithful, welche standardmäßig in R enthalten sind.

```
eruptionen <- faithful$eruptions
plot(ecdf(eruptionen))
```

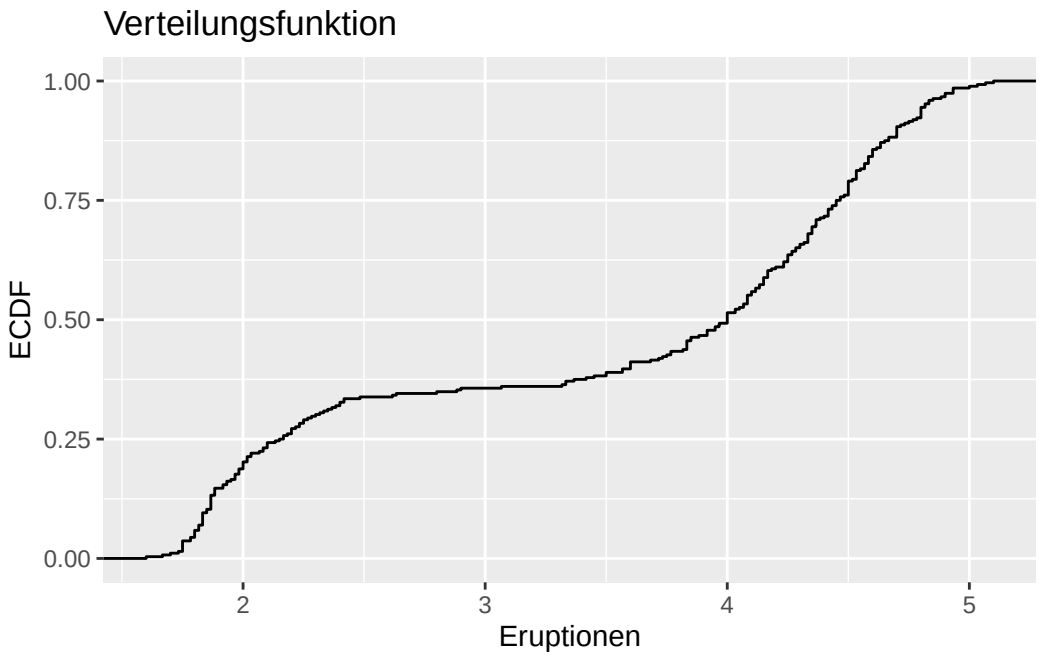



```
summary(eruptionen)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  1.600   2.163   4.000   3.488   4.454   5.100 
```

Alternativ mit ggplot2:

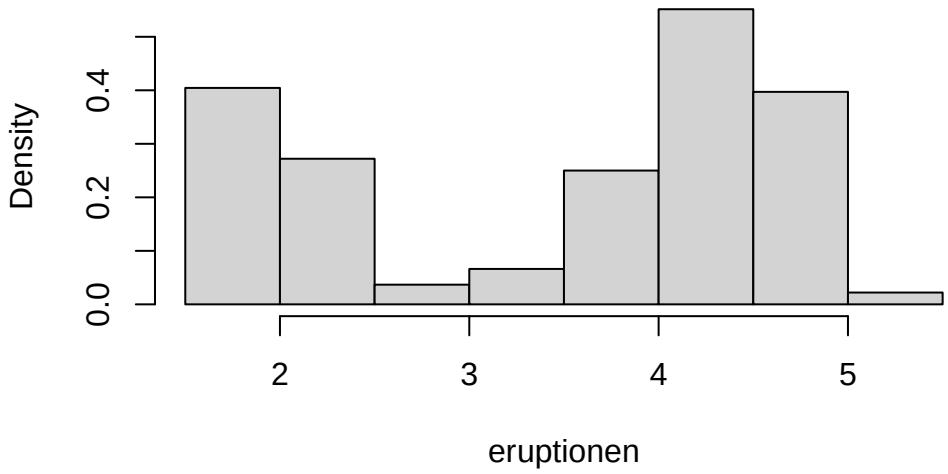
```
library(data.table)
library(ggplot2)

ggplot(data.table(x = eruptionen), aes(x = x)) +
  stat_ecdf(geom = "step") +
  labs(title = "Verteilungsfunktion", x = "Eruptionen", y = "ECDF")
```



Aufgabe 1b

Nun betrachten wir Histogramme:

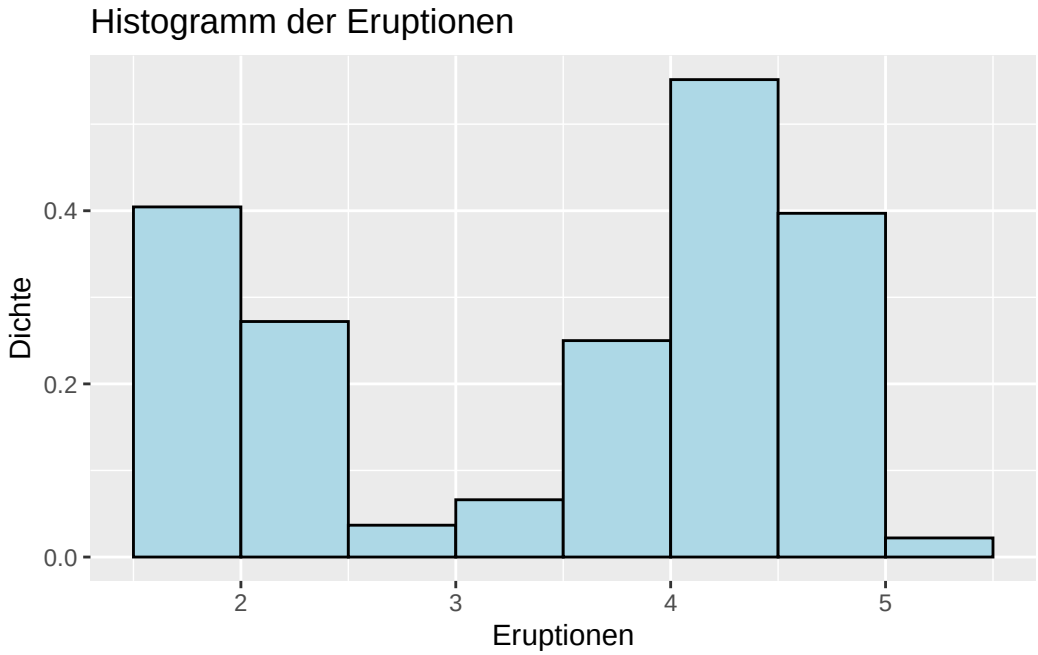
```
hist(eruptionen, breaks = seq(1.5, 5.5, by = 0.5), prob = TRUE)

hist(eruptionen, breaks = seq(1.5, 5.5, length = 8 + 1), prob = TRUE)
```



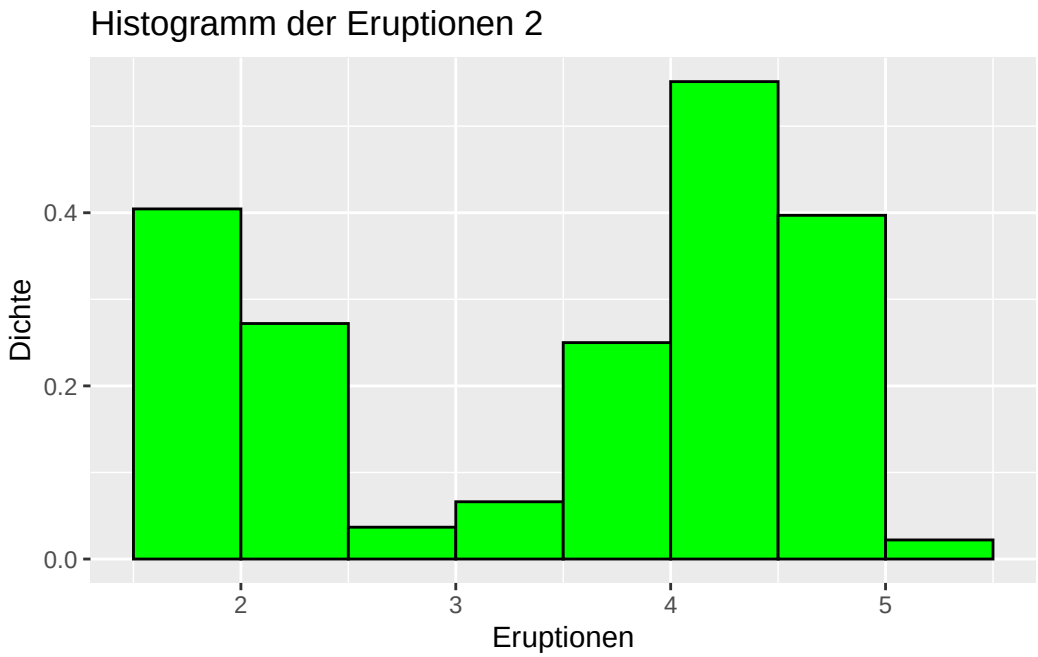
Hier sehen wir, dass beide Befehle ein klassisches Histogramm mit jeweils 8 Klassen liefern, welche komplett identisch aussehen. Nun betrachten wir auch, wie wir ähnliche Histogramme mit ggplot2 erstellen können:

```
# Variante 1: Mit "..density..":
ggplot(data.table(x = eruptionen), aes(x = x)) +
  geom_histogram(aes(y = ..density..),
                 binwidth = 0.5,
                 boundary = 1.5,
                 color = "black",
                 fill = "lightblue") +
  labs(title = "Histogramm der Eruptionen", x = "Eruptionen", y = "Dichte")
```

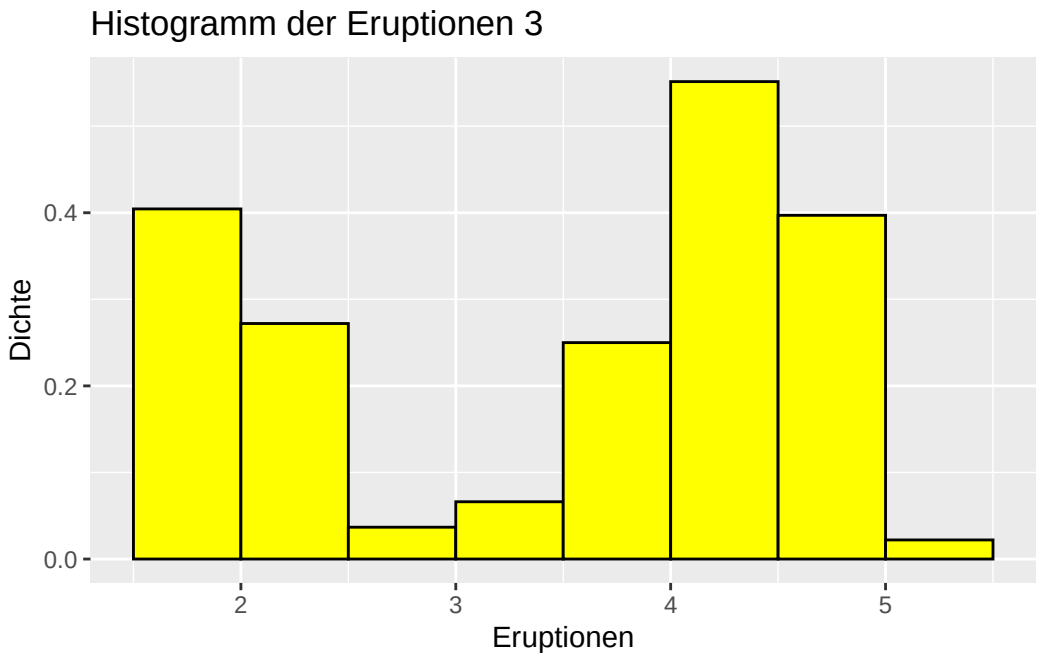
```
Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
i Please use `after_stat(density)` instead.
```



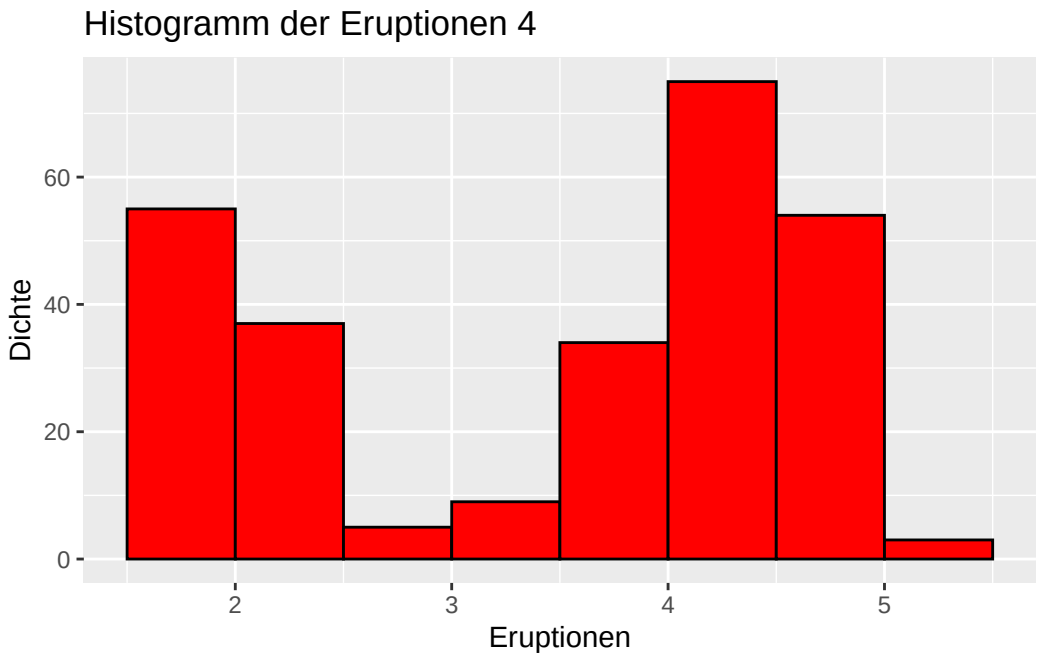

```
# Variante 2: Mit "after_stat(density)":
ggplot(data.table(x = eruptionen), aes(x = x)) +
  geom_histogram(aes(y = after_stat(density)),
                 binwidth = 0.5,
                 boundary = 1.5,
                 color = "black",
                 fill = "green") +
  labs(title = "Histogramm der Eruptionen 2", x = "Eruptionen", y = "Dichte")
```




```
# Variante 3: Mit "after_stat(density)" in ggplot:
ggplot(data.table(x = eruptionen), aes(x = x, y = after_stat(density))) +
  geom_histogram(,
                 binwidth = 0.5,
                 boundary = 1.5,
                 color = "black",
                 fill = "yellow") +
  labs(title = "Histogramm der Eruptionen 3", x = "Eruptionen", y = "Dichte")
```




```
# Variante 4: Ohne "after_stat(density)":
ggplot(data.table(x = eruptionen), aes(x = x)) +
  geom_histogram(,
                 binwidth = 0.5,
                 boundary = 1.5,
                 color = "black",
                 fill = "red") +
  labs(title = "Histogramm der Eruptionen 4", x = "Eruptionen", y = "Dichte")
```



Wir sehen, dass die Argumente '..density', sowie after_stat(density)' die gleichen Auswirkungen haben, allerdings wird '..density..' eigentlich nicht mehr verwendet. Zudem können wir sie im 'aes'-Argument von ggplot oder von geom_histogramm nutzen. Sie sind nötig, um die Fläche des Histogramms auf 1 zu normieren (wie beim Prob=TRUE Befehl im normalen 'Hist'-Befehl in R.) Zudem müssen wir unsere Werte, welche wir in 'eruptionen' gespeichert haben, in ein data.table-Objekt (oder einen data.frame) umwandeln, damit ggplot2 damit arbeiten kann.

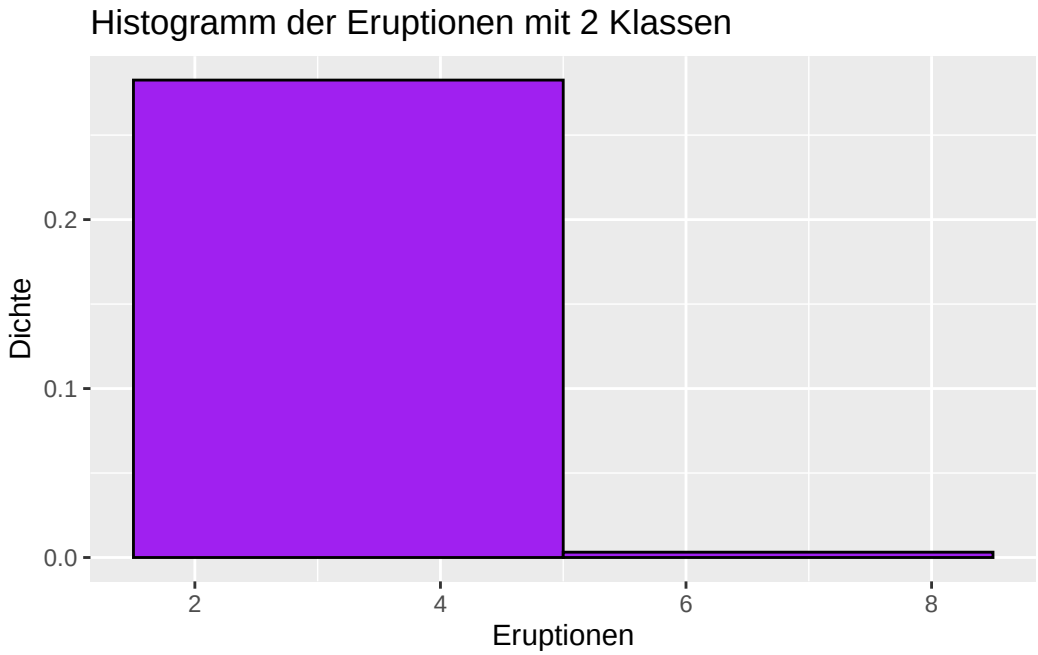
Egal für welche darstellung wir uns entscheiden, die Histogramme sind zeichnen alle das gleiche Bild. Wir sehen, dass wir eine ungewöhnliche (bimodale) Verteilung mit zwei verschiedenen lokalen Maxima haben, sodass wir ein solches Verhalten nicht mit einer klassischen Normalverteilung modellieren können, aber womöglich mit zwei verschiedenen Modi, modelliert durch zwei verschiedene Normalverteilungen.

Aufgabe 1c

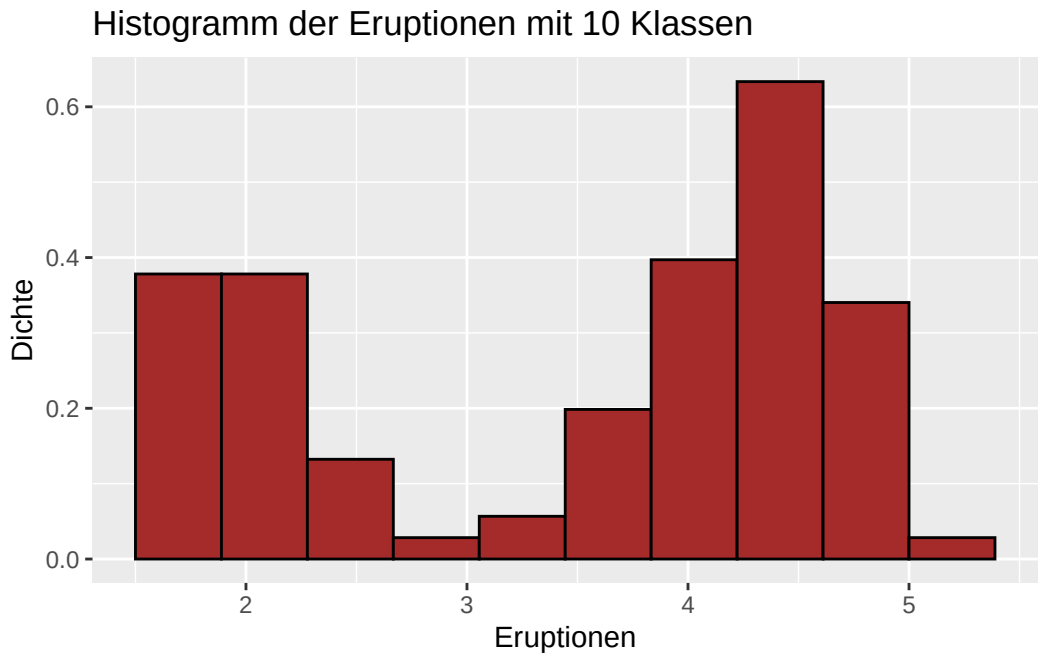
Weitere Histogramme, mit unterschiedlichen Klassenzahlen: Anstatt 'binwidth' können wir auch 'bins' verwenden, um die Anzahl der Klassen zu bestimmen. Dies ist besonders nützlich, wenn wir eine bestimmte Anzahl an Klassen haben wollen, aber nicht wissen, wie breit die Klassen sein sollen.

```
# Mit 2 Klassen:
ggplot(data.table(x = eruptionen), aes(x = x)) +
```

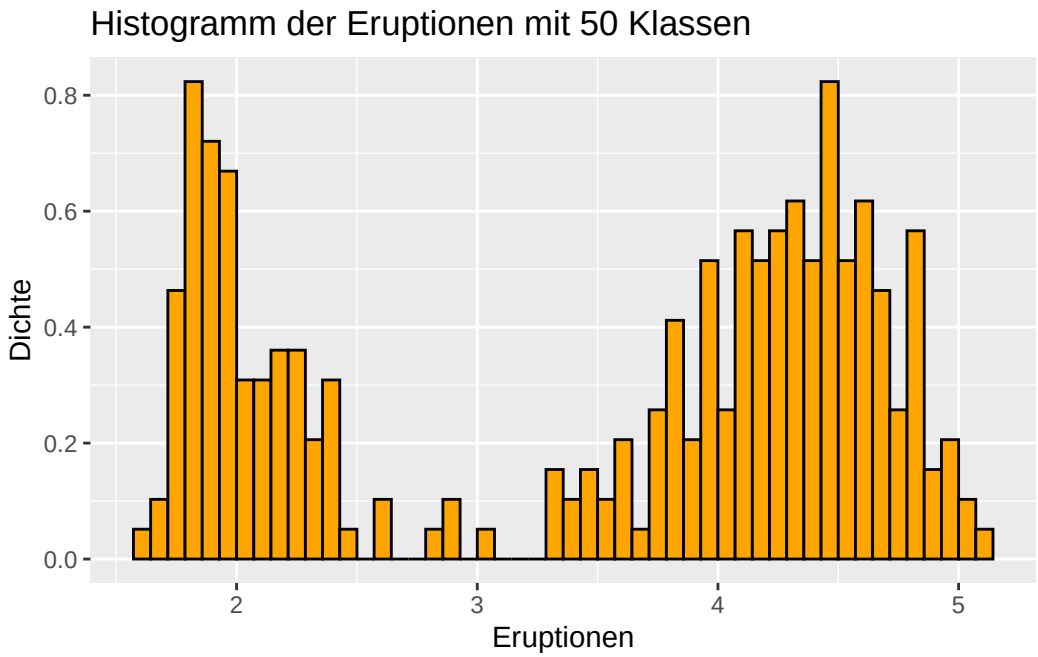
```
  geom_histogram(aes(y = after_stat(density)),
                 bins = 2,
                 boundary = 1.5,
                 color = "black",
                 fill = "purple") +
  labs(title = "Histogramm der Eruptionen mit 2 Klassen",
       x = "Eruptionen", y = "Dichte")
```



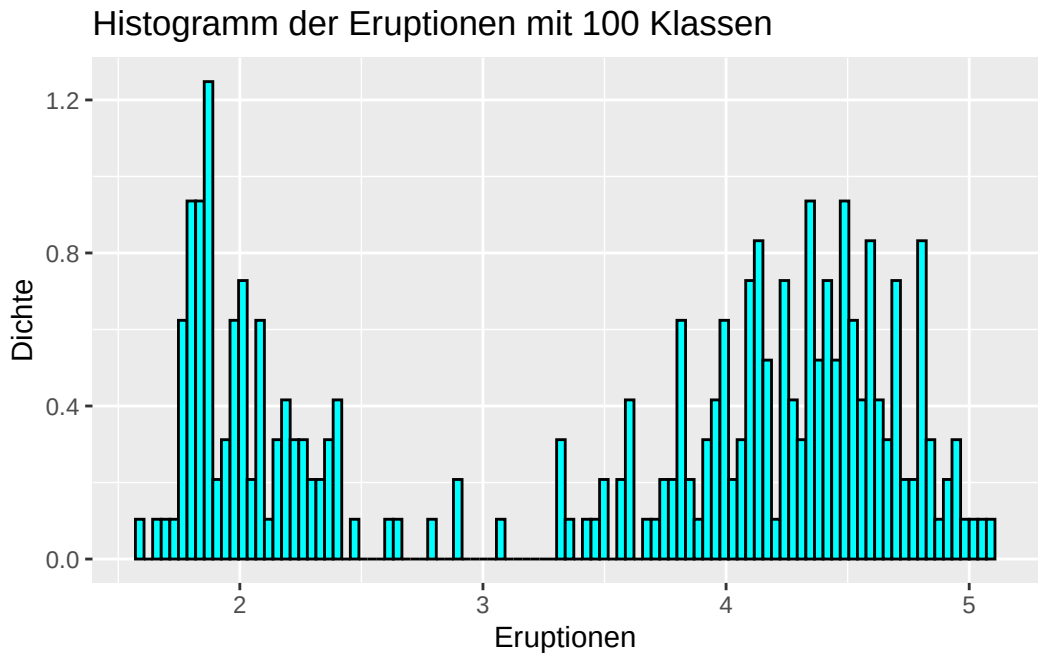

```
# Mit 10 Klassen:
ggplot(data.table(x = eruptionen), aes(x = x)) +
  geom_histogram(aes(y = after_stat(density)),
                 bins = 10,
                 boundary = 1.5,
                 color = "black",
                 fill = "brown") +
  labs(title = "Histogramm der Eruptionen mit 10 Klassen",
       x = "Eruptionen", y = "Dichte")
```




```
# Mit 50 Klassen:
ggplot(data.table(x = eruptionen), aes(x = x)) +
  geom_histogram(aes(y = after_stat(density)),
                 bins = 50,
                 boundary = 1.5,
                 color = "black",
                 fill = "orange") +
  labs(title = "Histogramm der Eruptionen mit 50 Klassen",
       x = "Eruptionen", y = "Dichte")
```




```
# Mit 100 Klassen:
ggplot(data.table(x = eruptionen), aes(x = x)) +
  geom_histogram(aes(y = after_stat(density)),
                 bins = 100,
                 boundary = 1.5,
                 color = "black",
                 fill = "cyan") +
  labs(title = "Histogramm der Eruptionen mit 100 Klassen",
       x = "Eruptionen", y = "Dichte")
```



Diese vier Histogramme sind ausgezeichnet dafür geeignet, um den Bias-Varianz-Trade-off zu demonstrieren. Wir sehen, dass die Histogramme mit 2 und 10 Klassen eine hohe Varianz haben, während die Histogramme mit 50 und 100 Klassen einen hohen Bias haben. Dies ist ein klassisches Beispiel für den Bias-Varianz-Trade-off, bei dem wir zwischen einer hohen Varianz (wenig Bias) und einem hohen Bias (wenig Varianz) wählen müssen. Anders ausgedrückt, die Histogramme mit wenigen Klassen weisen eine hohe Verzerrung auf und haben einen großen Informationsverlust, während Histogramme mit sehr vielen Klassen eine zu starke Anpassung aufweisen, und keine gute Eignung aufweisen, um die vorhandenen Daten gut aufbereitet und mit dem Fokus auf die Verteilung darzustellen.

Aufgabe 2

```
# Festlegen der Werte von x und b
x <- 3.75
b <- 0.5

# Berechnung von fxhat
fxhat <- 1/(length(eruptionen) * b) *
sum(x-b/2 <= eruptionen & eruptionen <= x+b/2)
fxhat
```

```
[1] 0.2647059
```

Dieser Wert wirkt stimmig, wenn wir ihn mit den Histogrammen aus der ersten Aufgabe vergleichen, insbesondere mit dem Histogramm mit einer Klassenbreite von 0,5. Wir sehen, dass die Höhe des Histogramms bei x=3.75 nahe dem berechneten Wert von fxhat ist.

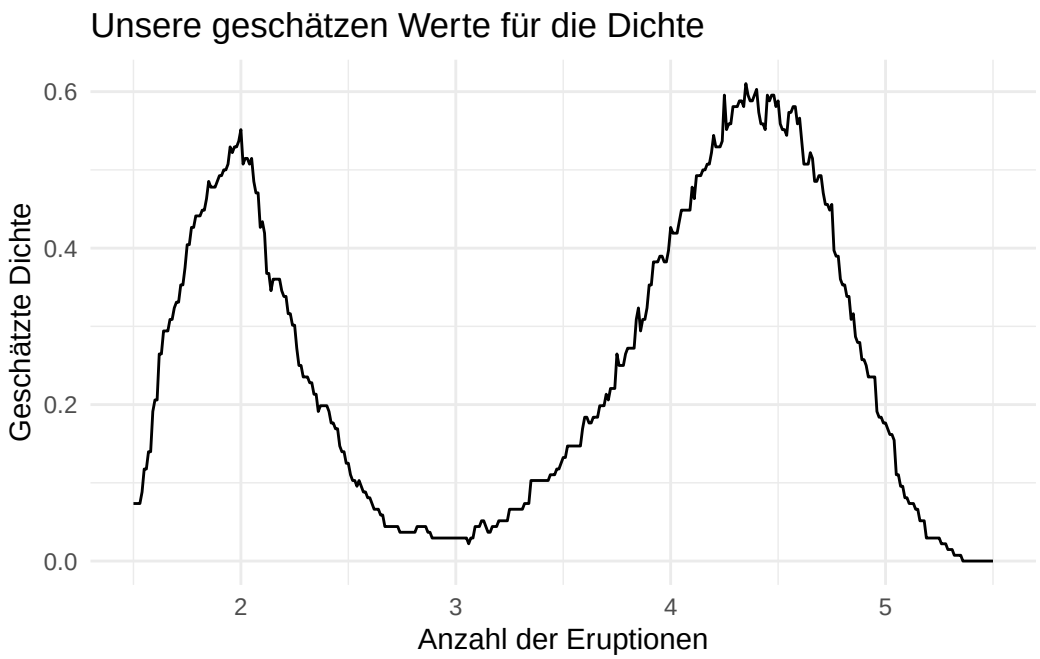
Zusatzaufgabe

```
# Initialisierung der Variablen
b <- 0.5
x <- seq(1.5, 5.5, by = 0.01)

# Berechnung von fxhat
fxhat <- rep(NA, length(x))

for(k in 1:length(x)){
fxhat[k] <- 1/(length(eruptionen)*b)*
sum(x[k]-b/2 <= eruptionen & eruptionen <= x[k]+b/2)
}
```

Wir haben nun Berechnungen für unsere 401 Werte von x durchgeführt. Nun können wir die Werte in einem Plot darstellen:

```
# Mit der normalen Plot Funktion von R:
plot(x, fxhat, type = "l", ylab = "geschätzte Dichte", bty = "n")
```

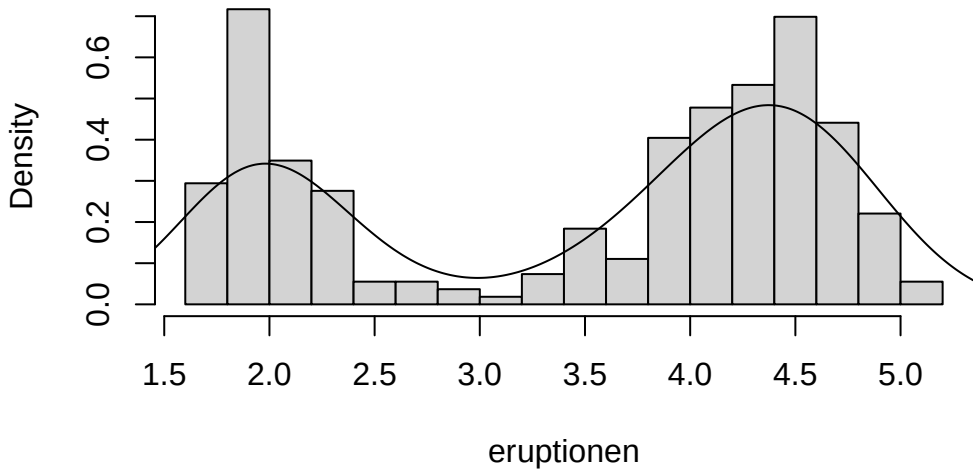

```
# Mit ggplot2:
ggplot(data.table(x = x, fxhat = fxhat), aes(x = x, y = fxhat)) +
  geom_line() +
  labs(title = "Unsere geschätzen Werte für die Dichte",
       x = "Anzahl der Eruptionen",
       y = "Geschätzte Dichte") +
  theme_minimal()
```



Aufgabe 3

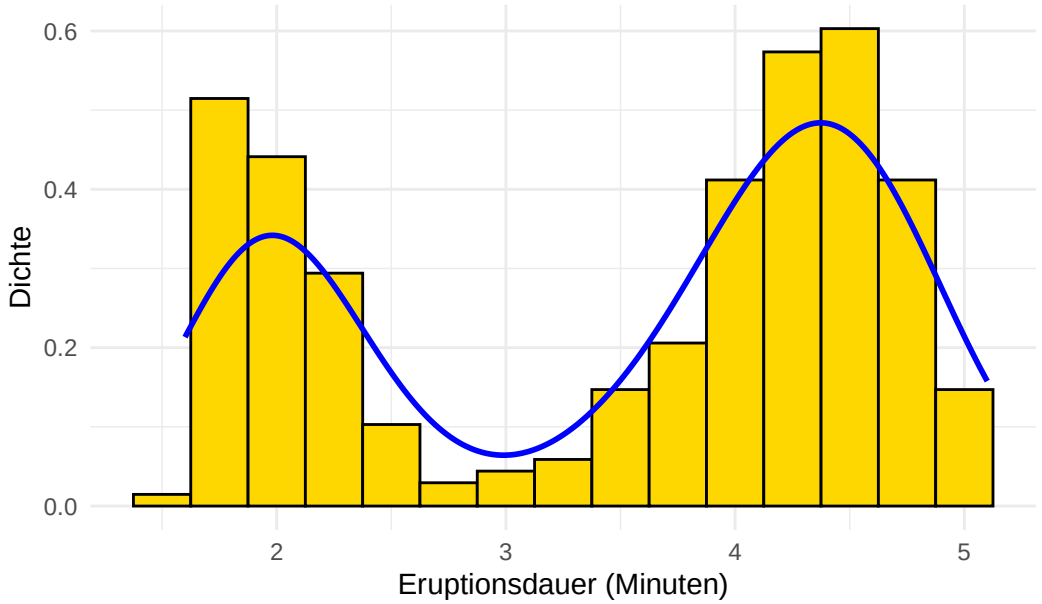
Kerndichteschätzung.

```
hist(eruptionen, breaks = 15, prob = TRUE, main = "")
lines(density(eruptionen))
```




```
# mit GGPlot
ggplot(data.table(x = eruptionen), aes(x = x)) +
  geom_histogram(aes(y = after_stat(density)),
                 bins = 15,
                 fill = "gold",
                 color = "black") +
  geom_density(color = "blue", size = 1) +
  labs(title = "",
       x = "Eruptionsdauer (Minuten)",
       y = "Dichte") +
  theme_minimal()
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```



Vorteile des Kerndichteschätzers sind insbesondere die glatte Darstellung als Graph, welcher, je nach Kern, auch differenzierbar ist. Zudem ist die Darstellung der Dichte nicht mehr von den Klassen abhängig, wie bei einem Histogramm. Dies ist ein großer Vorteil, da wir keine Klassenbreite mehr wählen müssen, und somit auch keine Verzerrung durch die Wahl der Klassenbreite haben.

```
?density
```

```
starte den http Server für die Hilfe fertig
```

Wir erhalten folgende Informationen: Kern: kernel, window a character string giving the smoothing kernel to be used. This must partially match one of “gaussian”, “rectangular”, “triangular”, “epanechnikov”, “biweight”, “cosine” or “optcosine”, with default “gaussian”, and may be abbreviated to a unique prefix (single letter). Also scheint der reguläre Kern der Gauss-Kern zu sein. Wir können auch andere Kerne verwenden, wie z.B. den Rechteckkern, den Dreiecks-Kern, den Epanechnikov-Kern, den Biweight-Kern, den Cosinus-Kern oder den Optcosine-Kern. Diese Kerne sind alle in der Funktion ‘density’ implementiert.

Bandweite: bw
the smoothing bandwidth to be used. The kernels are scaled such that this is the standard deviation of the smoothing kernel. (Note this differs from the reference books cited below.) Die Standardabweichung des Glättungskerns ist die Bandbreite, die wir verwenden. Dies ist ein wichtiger Parameter, da er die Glättung der Dichte beeinflusst. Eine zu kleine Bandbreite führt

zu einer Überanpassung, während eine zu große Bandbreite zu einer Unteranpassung führt. Wir können die Bandbreite auch manuell einstellen, indem wir den Parameter ‘bw’ verwenden. Wir können auch den Parameter ‘adjust’ verwenden, um die Bandbreite anzupassen. Dies ist ein Multiplikator für die Bandbreite, der die Bandbreite vergrößert oder verkleinert.