Datenanalyse

Abgabe 1

Maximilian Hagn (Matr. Nr. 11808237)

15.04.2020

1. Beispiel

Praxisbeispiel

Erstellen Sie ein Histogramm (siehe ?hist) der Variable LOI aus der Bodenschicht bhorizon. Auf diese Variable können Sie mit dem $-Operatorzugreifen(bhorizon$LOI).

Was versteht man unter einem Histogramm?

In einem Histogram können absolute oder relative Häufigkeiten von Daten dargestellt werden.

Fügen Sie nun zu diesem Histogramm zwei weitere Histogramme für die (transformierten) Daten hinzu und verwenden Sie dabei einmal die Methode "Friedman-Diaconis" für die Anzahl der Balken (auswählbar mittels Parameter breaks="FD") bzw. einmal eine äquidistante Klasseneinteilung in 6 Klassen (entspricht einer Klassen-Breite von 8.69). Der Parameter breaks der hist-Funktion kontrolliert nur grob die Anzahl der Klassen. Es kann notwendig sein, die Grenzen selbst mittels seq zu erzeugen. Mit dem Befehl par(mfrow = c(3, 1)) können Sie 3 Grafiken untereinander darstellen (siehe auch ?par).

Fügen Sie zwei Kerndichteschätzungen hinzu. Die Kerndichteschätzung kann mit dem Befehl density errechnet werden. Berechnen Sie die erste Kerndichteschätzung mit dem gaussian Kern und einmal mit dem optcosine Kern. Zeichnen Sie beide Kerndichteschätzungen mit dem Befehl lines (lines zeichnet in einen bereits bestehenden Plot) Linien in 2 unterschiedlichen Farben ein (siehe Parameter col des Befehls lines). Eventuell ist es notwendig, den Anzeigebereich des Plots anzupassen (Parameter ylim und xlim).

```
## Loading required package: sgeostat
```

```
## Registered S3 method overwritten by 'geoR':
##   method         from
##   plot.variogram sgeostat
```

Daten laden

```
data("moss")
data("ohorizon")
data("bhorizon")
data("chorizon")
```

Kerndichtefunktion für das Histogram und Methode Friedman-Diaconis

```
gaussian <- density(bhorizon$LOI, kernel="gaussian", adjust=0.0001)
optcosine <- density(bhorizon$LOI, kernel="optcosine", adjust=0.0001)
gaussian
```

```
##
## Call:
##  density.default(x = bhorizon$LOI, adjust = 1e-04, kernel = "gaussian")
```

```
##
## Data: bhorizon$LOI (609 obs.);   Bandwidth 'bw' = 7.527e-05
##
##        x                y
##  Min.   : 0.7898   Min.   : 0.00
##  1st Qu.:11.6524   1st Qu.: 0.00
##  Median :22.5150   Median : 0.00
##  Mean   :22.5150   Mean   :10.35
##  3rd Qu.:33.3776   3rd Qu.:11.32
##  Max.   :44.2402   Max.   :98.81
```

```
optcosine
```

```
##
## Call:
##  density.default(x = bhorizon$LOI, adjust = 1e-04, kernel = "optcosine")
##
## Data: bhorizon$LOI (609 obs.);   Bandwidth 'bw' = 7.527e-05
##
##        x                y
##  Min.   : 0.7898   Min.   : 0.000
##  1st Qu.:11.6524   1st Qu.: 0.000
##  Median :22.5150   Median : 0.000
##  Mean   :22.5150   Mean   : 8.871
##  3rd Qu.:33.3776   3rd Qu.: 9.699
##  Max.   :44.2402   Max.   :84.668
```

Kerndichtefunktion für 6 Klassen

```
gaussian6 <- density(bhorizon$LOI, kernel="gaussian", adjust=0.00001)
optcosine6 <- density(bhorizon$LOI, kernel="optcosine", adjust=0.00001)
gaussian6
```

```
##
## Call:
##  density.default(x = bhorizon$LOI, adjust = 1e-05, kernel = "gaussian")
##
## Data: bhorizon$LOI (609 obs.);   Bandwidth 'bw' = 7.527e-06
##
##        x               y
##  Min.   : 0.79   Min.   :  0.0
##  1st Qu.:11.65   1st Qu.:  0.0
##  Median :22.52   Median :  0.0
##  Mean   :22.52   Mean   :103.5
##  3rd Qu.:33.38   3rd Qu.:113.3
##  Max.   :44.24   Max.   :988.9
```

```
optcosine6
```

```
##
## Call:
##  density.default(x = bhorizon$LOI, adjust = 1e-05, kernel = "optcosine")
##
## Data: bhorizon$LOI (609 obs.);   Bandwidth 'bw' = 7.527e-06
##
##        x               y
##  Min.   : 0.79   Min.   :  0.00
```

```
##  1st Qu.:11.65   1st Qu.:  0.00
##  Median :22.52   Median :  0.00
##  Mean   :22.52   Mean   : 88.71
##  3rd Qu.:33.38   3rd Qu.: 97.08
##  Max.   :44.24   Max.   :847.34
```

```
par(mfrow=c(3,1))
hist(bhorizon$LOI, breaks=100, main="Histogram von bhorizon", xlim=c(-1.468,46.498), ylim=c(0,100))
lines(gaussian, col="red")
lines(optcosine, col="blue")
hist(bhorizon$LOI, breaks="FD", main="Methode Friedman-Diaconis", xlim=c(0,45), ylim=c(0,100))
lines(gaussian, col="red")
lines(optcosine, col="blue")
hist(bhorizon$LOI, breaks=8.69, main="In 6 Klassen", xlim=c(0,45), ylim=c(0,1000))
lines(gaussian6, col="red")
lines(optcosine6, col="blue")
```

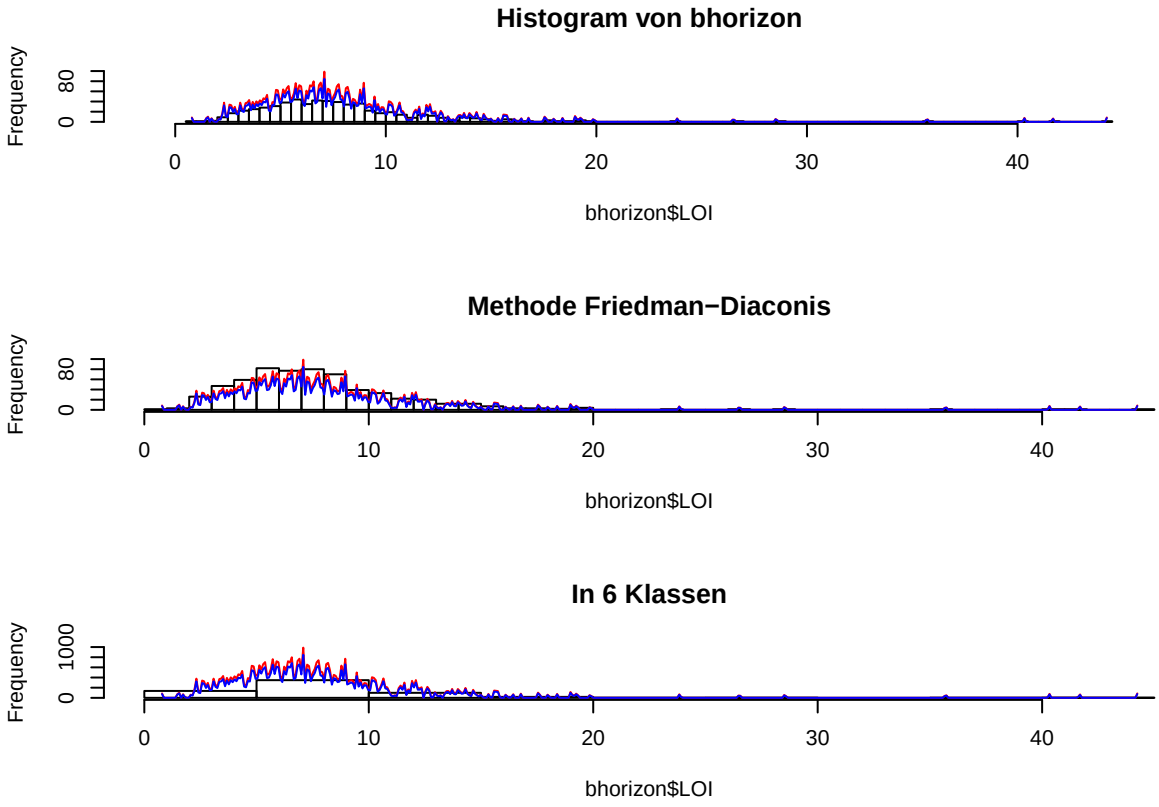

Figure 1: Histogramme von bHorizon Daten

Fragen

Ist es sinnvoll bzw. notwendig die Daten zu transformieren? Falls Sie eine Transformation durchgeführt haben, begründen sie weshalb.

Ja es ist sinnvoll, da wir möglichst viel auf den Diagrammen erkennen möchten. Werden mehr breaks angezeigt, erhalten wir mehr Balken und können so die Verteilung betrachten. Des Weiteren entstehen keine großen Blöcken mit Ausreissern, aus denen schwer Erkenntnisse abgeleitet werden können.

Erläutern Sie kurz die Unterschiede in den Klasseneinteilungen. Welche Einteilung beschreibt am besten die Daten und weshalb?

Wie oben bereits beschrieben, können wir die Histogramme bzw. die darin enthaltenen Daten genauer betrachten. Es muss ein guter Mittelweg gefunden werden, um ein Histogram lesbar zu gestalten. So werden die Werte auf der y-Achse mit steigenden breaks kleiner. Meiner Ansicht nach ist die Friedman-Diaconis Methode mit diesem Datensatz sehr übersichtlich.

Wie äußern sich die beiden unterschiedlichen Kerne der Dichteschätzung?

Die Dichteschätzung mit dem optcosine-Kern ist ein weniger geringer, als mit dem gaussian-Kern. Die beiden Kerne sind lediglich verschiedene Schätzungen des zugrunde liegenden Diagramms

Welcher Kern ist für das konkrete Beispiel besser geeignet?

Meiner Ansicht nach können für dieses konkrete Beispiel beide Kerne verwendet werden.

2. Beispiel

Praisbeispiel

Verwenden Sie für dieses Beispiel die Daten Diamond aus dem R-Paket library(Ecdat).

Erstellen Sie Boxplots mit Kerben für die Preise der Diamanten gruppiert nach der Variable colour. Bedenken Sie, dass der Preis von schwereren Diamanten trivialerweise höher ist, weshalb das Gewicht (Variable carat) herausgerechnet werden muss. Dies könnte z.B. so gemacht werden, dass nicht die Variable price, sondern price/carat genommen wird – aber diese Lösung wird noch nicht optimal sein, und es sollten auch andere Ansätze versucht werden (Tipp: der normierte Preis sollte nicht mehr von carat abhängen).

```
library(Ecdat)
```

```
## Loading required package: Ecfun
```

```
##
## Attaching package: 'Ecfun'
```

```
## The following object is masked from 'package:base':
##
##     sign
```

```
##
## Attaching package: 'Ecdat'
```

```
## The following object is masked from 'package:datasets':
##
##     Orange
```

```
data("Diamond")
boxplot(price/carat ~ colour, Diamond, notch=TRUE)
```

Fragen

Was ist die Bedeutung der Box und der Kerben im Boxplot?

Die Box beschreibt die Länge vom unteren Quartil bis zum oberen Quartil. Der Strich in der Mitte bildet den Median. Die Kerben bilden das 95% Konfidenzinterval ab.

Fallen bestimmte Gruppen als besonders teuer/billig auf?

Die Farbe D ist eher teuerer angesetzt und hat im Gegensatz zu den anderen Farben einen größeren Interquartilsabstand. Die anderen Farben sind sehr ähnlich.

Unterscheiden sich die Mediane der einzelnen Gruppen signifikant (beachten Sie insbesondere die Kerben)?

Nur bei D ist der Unterschied und die Spannweite des Konfidenzintervalls signifikat unterschiedlich zu den anderen Gruppen.

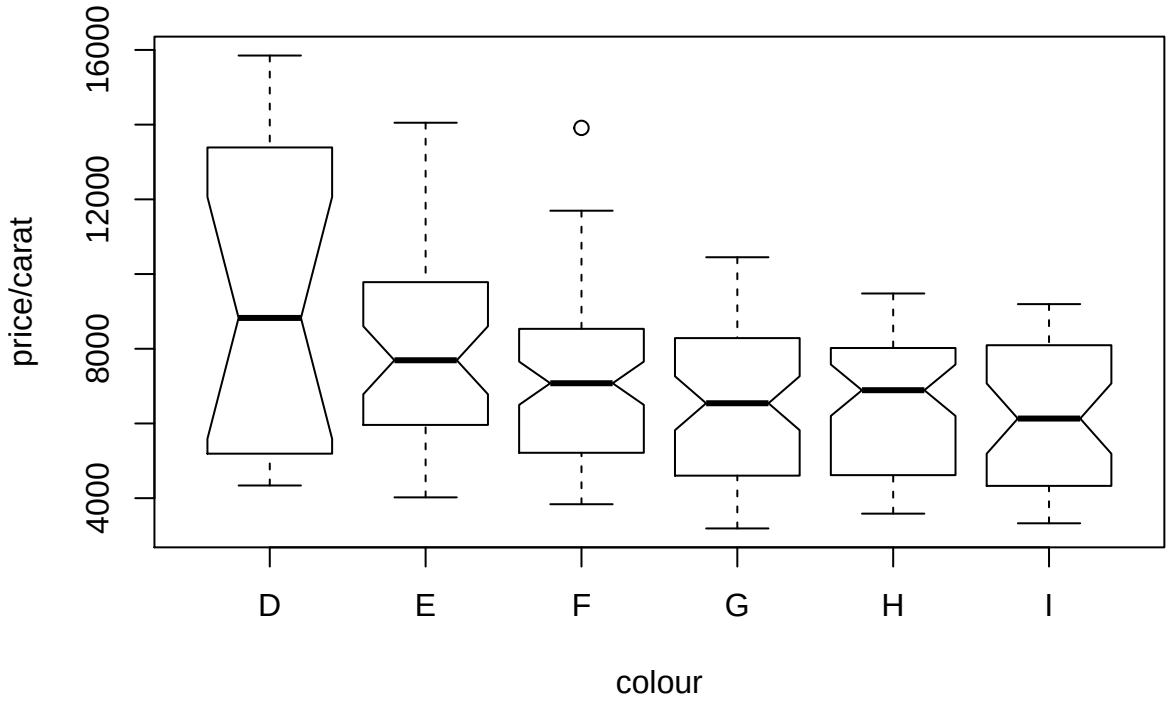

Figure 2: Box-Plot von Diamond Daten

3. Beispiel

Praxisbeispiel A

Generieren Sie sich zuerst 100 Realisierungen zweier Zufallsgrößen die aus einer Poisson-Verteilung mit Parameter lambda = 5.67 bzw. einer Exponentialverteilung mit Parameter rate = 55 stammen. Dies funktioniert mit dem Befehl x.desc <- rpois(100, lambda = 5.67) bzw. x.cont <- rexp(100, rate = 55). Zeichnen Sie nun die empirische Verteilungsfunktion (?ecdf) der beiden Zufallsvektoren mit unterschiedlichen Farben in eine Grafik. Falls die Verteilungsfunktionen aufgrund von sehr unterschiedlichen numerischen Bereiche nicht in einer Grafik vereinbar sind, erstellen Sie separate Plots (begründen Sie dies jedoch in der Dokumentation).

```
x.desc <- rpois(100, lambda = 5.67)
x.cont <- rexp(100, rate = 55)

f0 <- ecdf(x.desc)
f1 <-ecdf(x.cont*50)

plot(f0, col="blue", main="Zufallsverteilungen", xlab="x", ylab="y")
plot(f1, add=TRUE, col='red')
legend("bottomright", legend=c("Poisson-Verteilung", "Exponentialverteilung "), col=c("blue", "red"), l
```

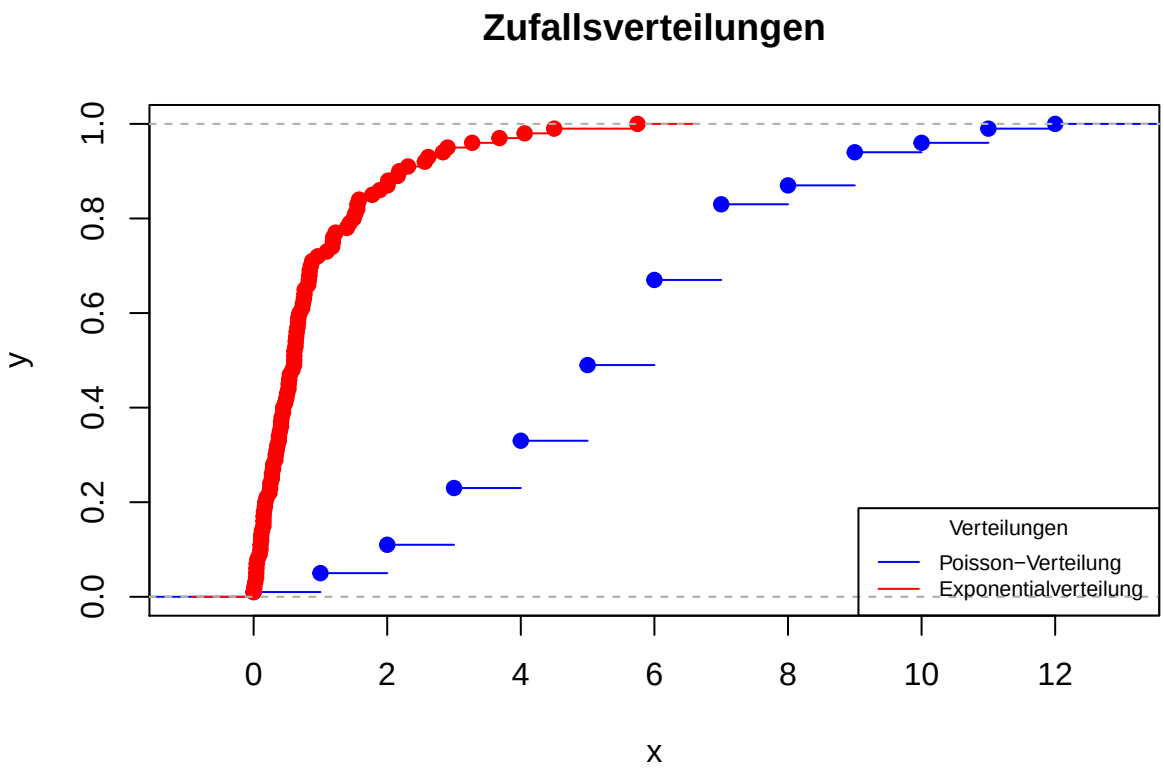

Figure 3: Zufallsverteilung von Poisson und Exponential

Fragen

Wie ist die empirische Verteilungsfunktion definiert?

Die empirische Verteilungsfunktion gibt zu jedem Wert x den Anteil der Werte der Stichprobe, die kleiner oder gleich x sind.

Beschreiben Sie die Unterschiede, die Ihnen zwischen der diskret und der kontinuierlich verteilten Zufallsgröße auffallen.

Die Poisson Verteilung ist diskret. Die Exponentielle Verteilung ist kontinuierlich.

Praxisbeispiel B

Plotten Sie außerdem die empirische Verteilungsfunktion für die Variablen Co, Sr_AA und pH aus der Schicht ohorizon (falls sinnvoll, sollten Sie die Daten zuerst transformieren).

```
co <- ecdf(ohorizon$Co)
Sr_AA <- ecdf(ohorizon$Sr_AA)
pH <- ecdf(ohorizon$pH)

plot(co, col="brown", main="Verteilungsfunktion OHorizon", ylab="y")
plot(Sr_AA, add=TRUE, col="yellow")
plot(pH, add=TRUE, col="blue")
```

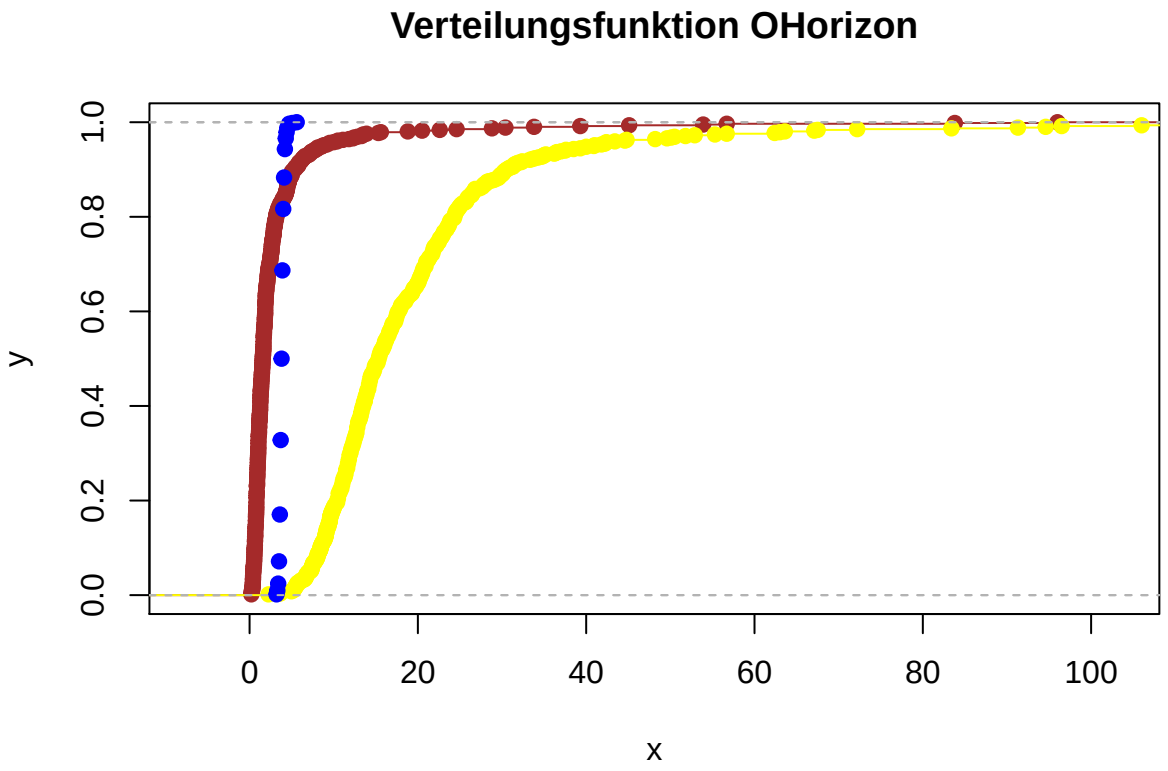

Figure 4: Empirische Verteilungsfunktion für die Daten Co, Sr_AA, ph

Fragen

Interpretieren Sie die Ergebnisse und Unterschiede der empirischen Verteilungsfunktionen.

Co –> Exponentiell; Sr_AA –> Exponentiell; pH –> Chi2

4. Beispiel

Praxisbeispiel

Ziehen Sie zufällig 500 Werte aus einer Normalverteilung mit Mittelwert 98 und Standardabweichung 6 (x <- rnorm(500, 98, 6)). Erstellen Sie für die generierten Werte mit der Funktion qqplot.das einen QQ-Plot.

```
x <- rnorm(500, 98, 6)
qqplot.das(x)
```

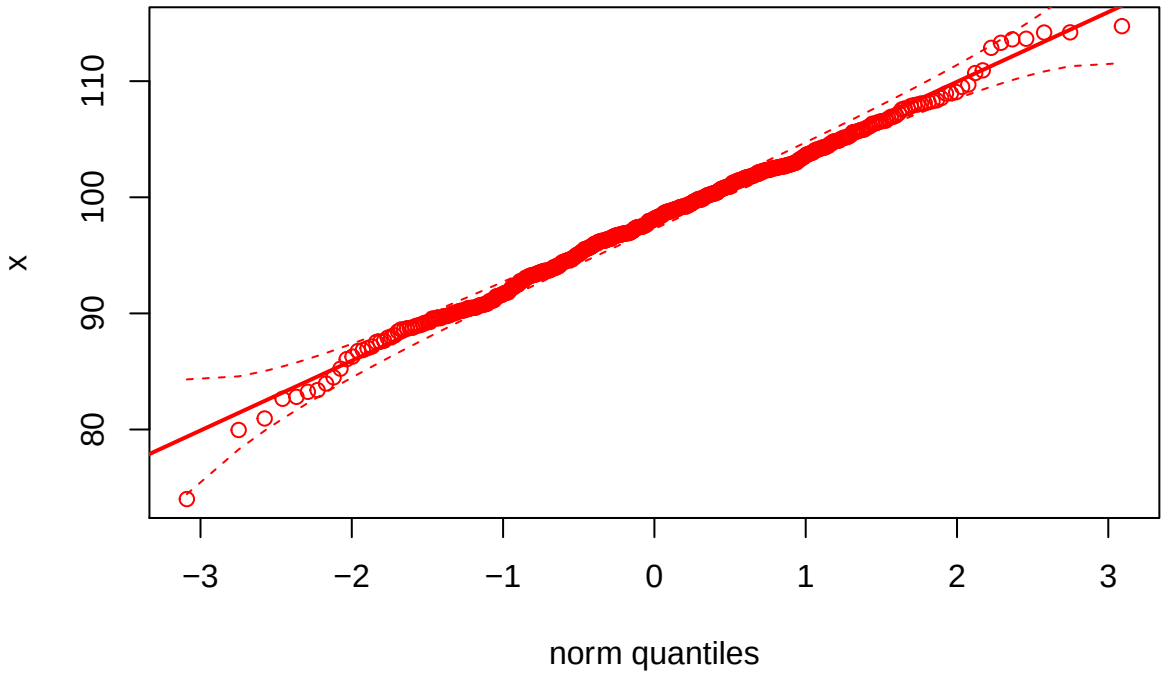

Figure 5: QQ-Plot für Normalverteilung (500,98,6)

Grid Funktion

Sind die Parameter der zugrunde liegenden Normalverteilung im Plot ersichtlich und wenn ja, wie? Hierfür kann die Funktion grid hilfreich sein, welche bei einem bereits bestehenden Plot ein Gitter im Hintergrund einblendet.

Ja die Parameter sind ersichtlich, da der Mittelwert 0 bei 98 liegt und im Bereich [-1, 1] eine Standardabweichung von 6 zu erkennen ist.

```
x <- rnorm(500, 98, 6)
qqplot.das(x)
grid()
```

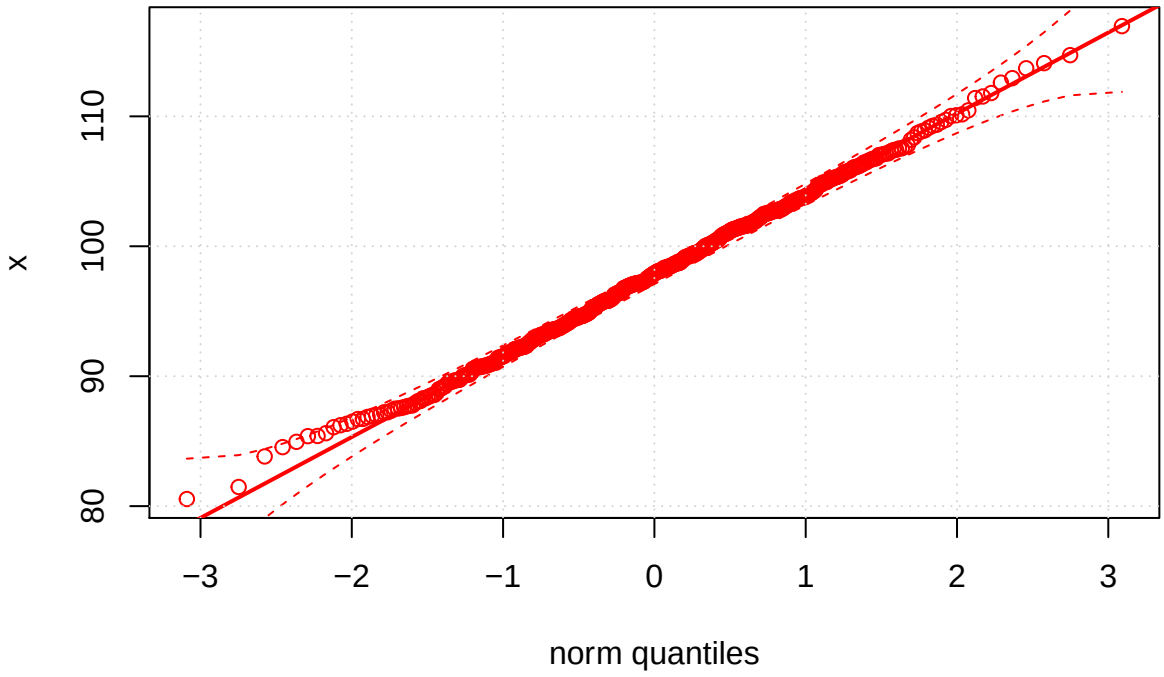

Figure 6: QQ-Plot mit Grid für Normalverteilung (500,98,6)

OHorizon Werte

Stellen Sie nun den QQ-Plot der 3 (evtl. transformierten) Variablen aus Beispiel 3 dar. Der QQ-Plot der ersten Variable kann wieder mit qqplot.das erzeugt werden. Die weiteren Variablen sollen im selben Plot mit anderer Farbe aufscheinen (kann mit dem Parameter add.plot der Funktion qqplot.das erzielt werden). Passen Sie eventuell wieder die Bereiche des Plots mit den Parametern xlim und ylim an. Falls die Daten trotz Anpassung der Bereiche nicht in einen Plot passen, erstellen sie separate Plots (mit Begründung in der Dokumentation).

```
qqplot.das(ohorizon$Co, col="black", ylab="value")
qqplot.das(ohorizon$Sr_AA, add.plot=TRUE, col="brown")
qqplot.das(ohorizon$pH, add.plot=TRUE, col="blue")
legend("topleft", legend=c("Co", "Sr_AA", "pH"), col=c("black", "brown", "blue"), lty=1:1, title="Wert")


grid()
```

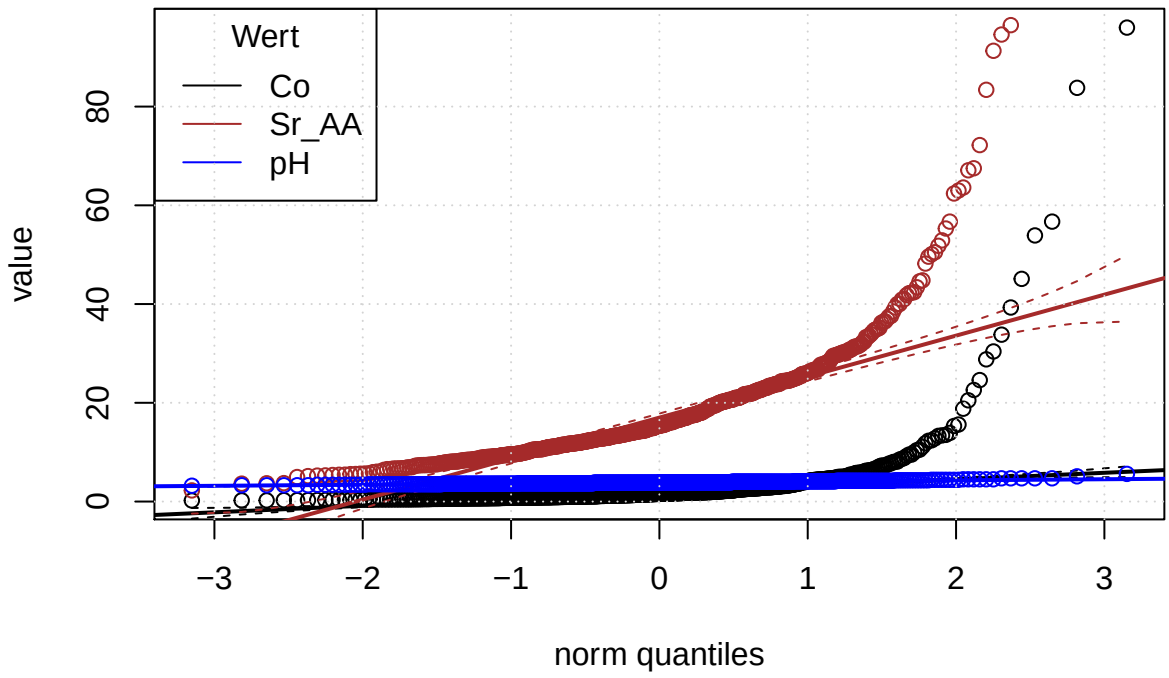

Figure 7: QQ-Plot der 3 Variablen Co, Sr_AA, pH

Fragen

Lassen sich im QQ-Plot Strukturen in den Daten erkennen und unterstützen die QQ-Plots ihre Schlüsse in Beispiel 3?

Ja es lassen sich Strukturen erkennen. Die pH Daten sind in allen Quantilen sehr ähnlich. Die Co und Sr_AA Daten dahingegen sind exponentiell steigend, wobei Sr_AA insgesamt höhere Werte aufweist.