Datenanalyse

Abgabe 2

Lukas Mahler (Matr. Nr. 11908553)

16.05.2021

Vorinformationen:

Der Fließtext der Abgabe wurde in deutscher Sprache verfasst, der R-Code selbst wurde nach Konvention in englisch geschrieben. Aus diesem Grund sind auch die Grafiken englisch beschriftet. Folgende Pakete werden zur einfacheren Bearbeitung und Dartstellung der Daten genutzt:

- **data.table:** Vereinfacht das Laden der Daten von einem "CSV" File in ein Dataframe.
- **ggplot2:** Erleichtert das Produzieren aussagekräftiger Plots.
- **zoo:** Genutzt, um das Datum, da kein Tag vorhanden ist, korrekt einzulesen.
- **ggseas:** Erlaubt das einfache Plotten von seasonal data in ggplot.
- **ggfortify:** Ermöglicht es ggplot timerseries Objekte zu interpretieren.

Nach Möglichkeit wurden die Grafiken mit ggplot gezeichnet, da die dadurch produzierten Grafiken aussagekräftiger sind und der R-Code lesbarer ist als mit den nativen Methoden von R. Die verwendeten Methoden nutzen dieselben Modelle (teilweise sogar direkt denselben sourcecode) wie die vorgeschlagenen, der produzierte Output sollte sich daher inhaltlich nicht mit dem der vorgeschlagenen Methoden unterscheiden.

```
# install.packages("data.table")
# install.packages("ggplot2")
# install.packages("zoo")
# install.packages("robfilter")
# install.packages("ggseas")
# install.packages("ggfortify")

library(data.table) # used for simplified reading into data frame
library(ggplot2) # used for simplified plotting
library(zoo) # for simpler date reading
library(robfilter) # for robust filtering of data
library(ggseas) # for saisonal trends directly in ggplot
library(ggfortify) # for ggplot to interpret timeseries objects
set.seed(24) # seed for reproducibility
```

1. Beispiel

Als Datensatz dient die Primärenergieproduktion der USA, abrufbar unter link, aufgeschlüsselt nach Energiequellen. Nach Laden der Daten ergibt sich folgendes Bild:

```
# Data from:
# https://www.eia.gov/totalenergy/data/browser/index.php?tbl=T01.02#
# Primary Energy Production by Source in the USA
if (grepl("Datenanalyse$", getwd())) setwd("exercises/exercise2/working")
```

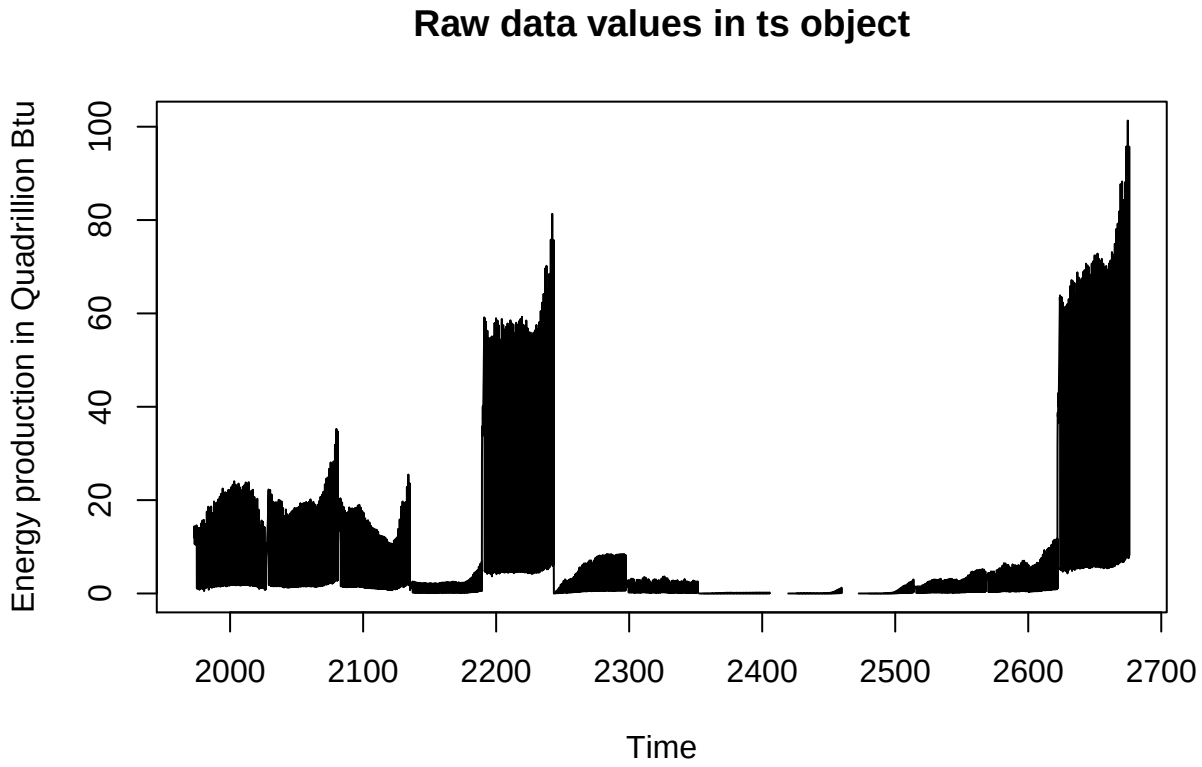
```
data_raw <- fread(
    file = "../data/MER_T01_02.csv", sep = ",",
    col.names = c("MSN", "Date", "Energy", "Source", "Description"),
    select = c(1, 2, 3, 4, 5),
    colClasses = c(YYYYMM = "character")
)
str(data_raw) # view basic structure of the data
```

```
## Classes 'data.table' and 'data.frame':   8437 obs. of  5 variables:
##  $ MSN        : chr  "CLPRBUS" "CLPRBUS" "CLPRBUS" "CLPRBUS" ...
##  $ Date       : chr  "194913" "195013" "195113" "195213" ...
##  $ Energy     : chr  "11.973882" "14.060135" "14.419325" "12.734313" ...
##  $ Source     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Description: chr  "Coal Production" "Coal Production" "Coal Production" "Coal Production" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
raw_ts <- ts(data_raw$Energy, start = c(1973, 01), frequency = 12)

plot(raw_ts, main = "Raw data values in ts object",
    ylab = "Energy production in Quadrillion Btu")
```

```
## Warning in xy.coords(x, NULL, log = log, setLab = FALSE): NAs introduced by
## coercion
```

```
## Warning in xy.coords(x, y): NAs introduced by coercion
```




```
data_cleaned <- data_raw
# Convert Value to num and set NA args
data_cleaned$Energy <- type.convert(data_cleaned$Energy,
    na.strings = c("Not Available")
)
```

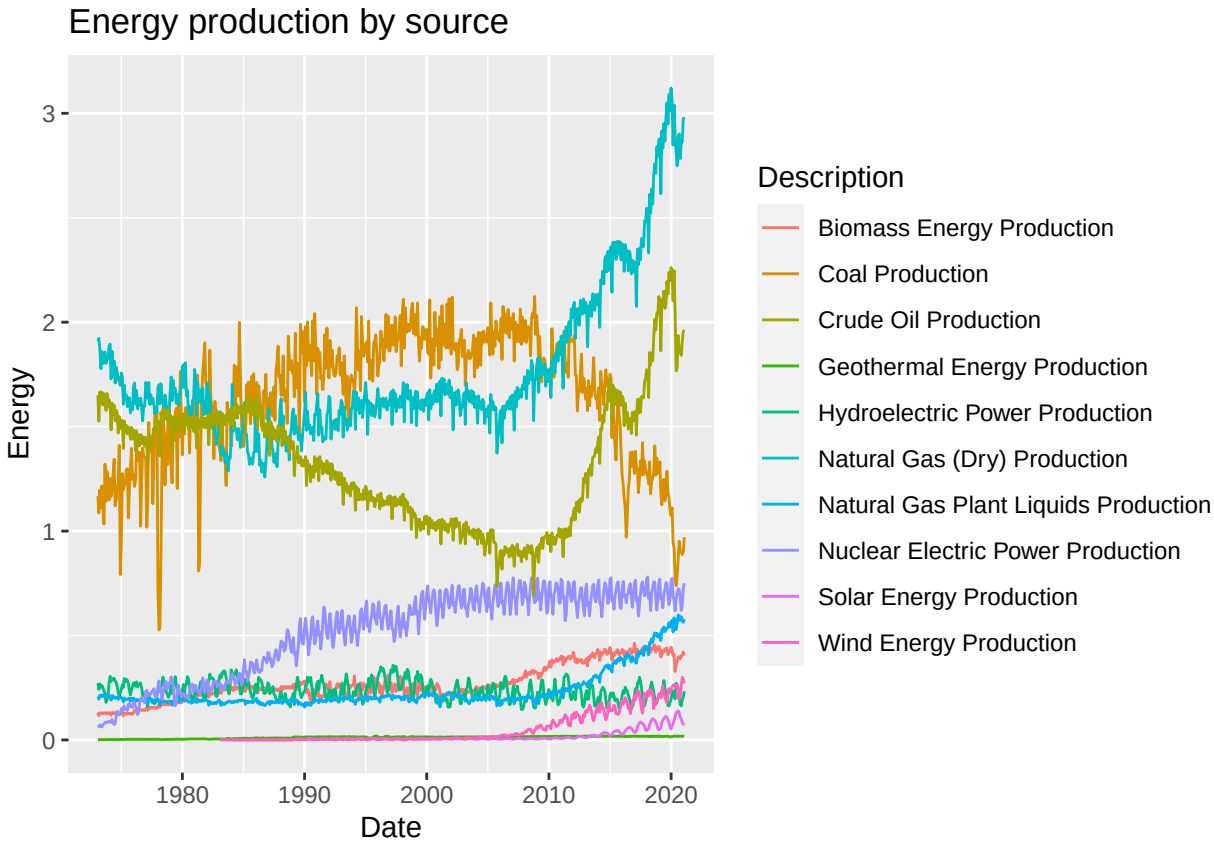
```r
# Convert into date format
data_cleaned$Date <- as.Date(as.yearmon(data_cleaned$Date, "%Y%m"), frac = 1)
str(data_cleaned)
```

```
## Classes 'data.table' and 'data.frame':   8437 obs. of  5 variables:
##  $ MSN        : chr  "CLPRBUS" "CLPRBUS" "CLPRBUS" "CLPRBUS" ...
##  $ Date       : Date, format: NA NA ...
##  $ Energy     : num  12 14.1 14.4 12.7 12.3 ...
##  $ Source     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Description: chr  "Coal Production" "Coal Production" "Coal Production" "Coal Production" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Wie aus den Daten klar ersichtlich ist, werden nun die unterschiedlichen Energiequellen in der Zeit hintereinander aufgereiht. Nun wird das Dataframe in seine unterschiedlichen Energiequellen aufgeteilt und diese Teile werden dann als einzelne Zeitreihen behandelt. Auch kann man eine starke Fluktuation in den Datenwerten feststellen. Diese kommt dadurch zustande, dass für jedes Jahr ein Monat 13 gespeichert wurde, das die insgesamt produzierte Energie pro Jahr angibt. Dieses dritte Monat muss zusätlich entfernt werden, da es die Daten ansonsten immer um ein Monat pro Jahr verschiebt und zusätzlich falsche Daten einbringt.

```r
# Remove all summaries for a year (month 13) and all Total sources
data_monthly <- subset(data_cleaned, !is.na(Date))
data_monthly_total <- subset(data_monthly, grepl("^Total", Description))
data_monthly <- subset(data_monthly, !grepl("^Total", Description))
data_sources <- split(data_monthly, data_monthly$Source)

ggplot(data = data_monthly,
    aes(x = Date, y = Energy, group = MSN, colour = Description)) +
    geom_line(size = 0.5) +
    ggtitle("Energy production by source")
```

```
## Warning: Removed 252 row(s) containing missing values (geom_path).
```

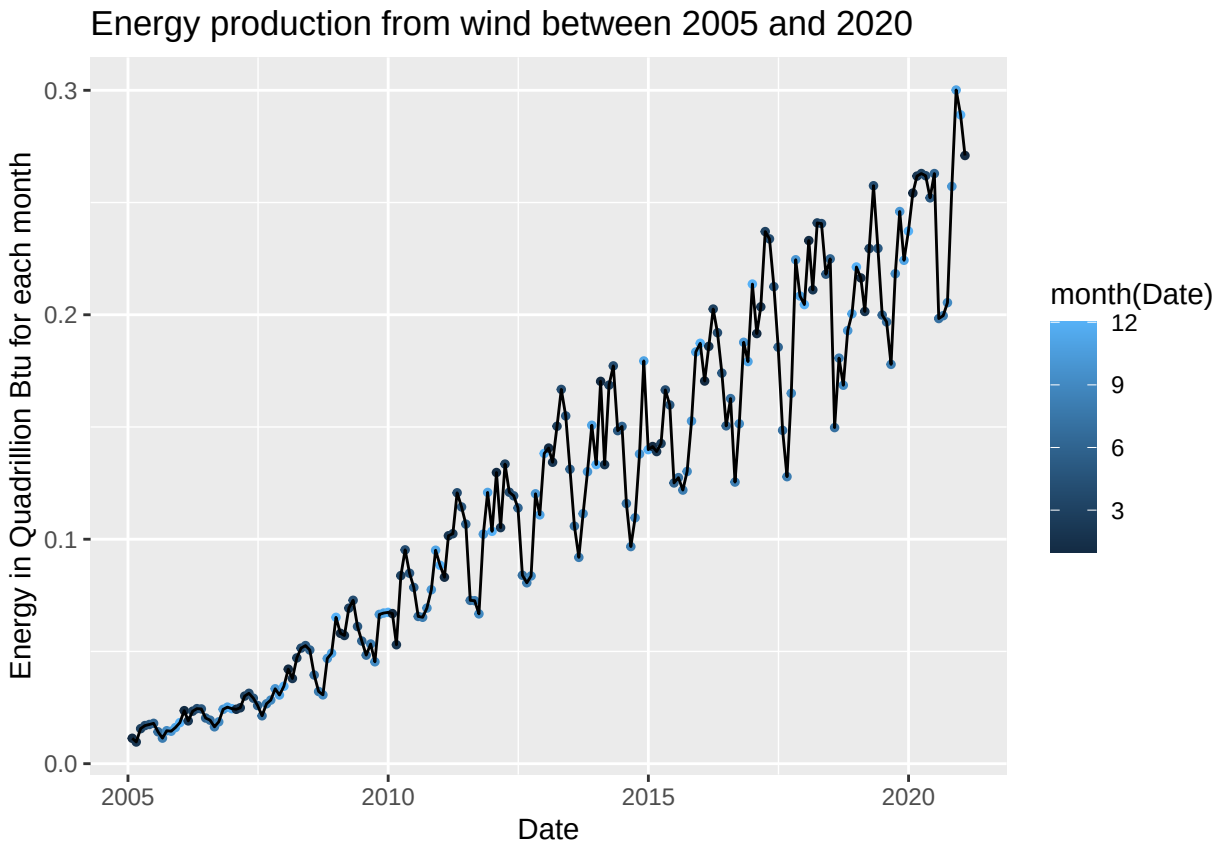


Nun sind die Daten aufbereitet, um als Timeseries Objekte eingelsesen und in Modelle verarbeitet zu werden. Für die weiteren Aufgaben werden wir die produzierte Energie durch Wind ab 2005 betrachten.

```
wind <- subset(data_sources$"10", Date > as.Date("2005-01-01"))
wind_ts <- ts(wind$Energy, start = c(2005, 01), frequency = 12)
str(wind)
```

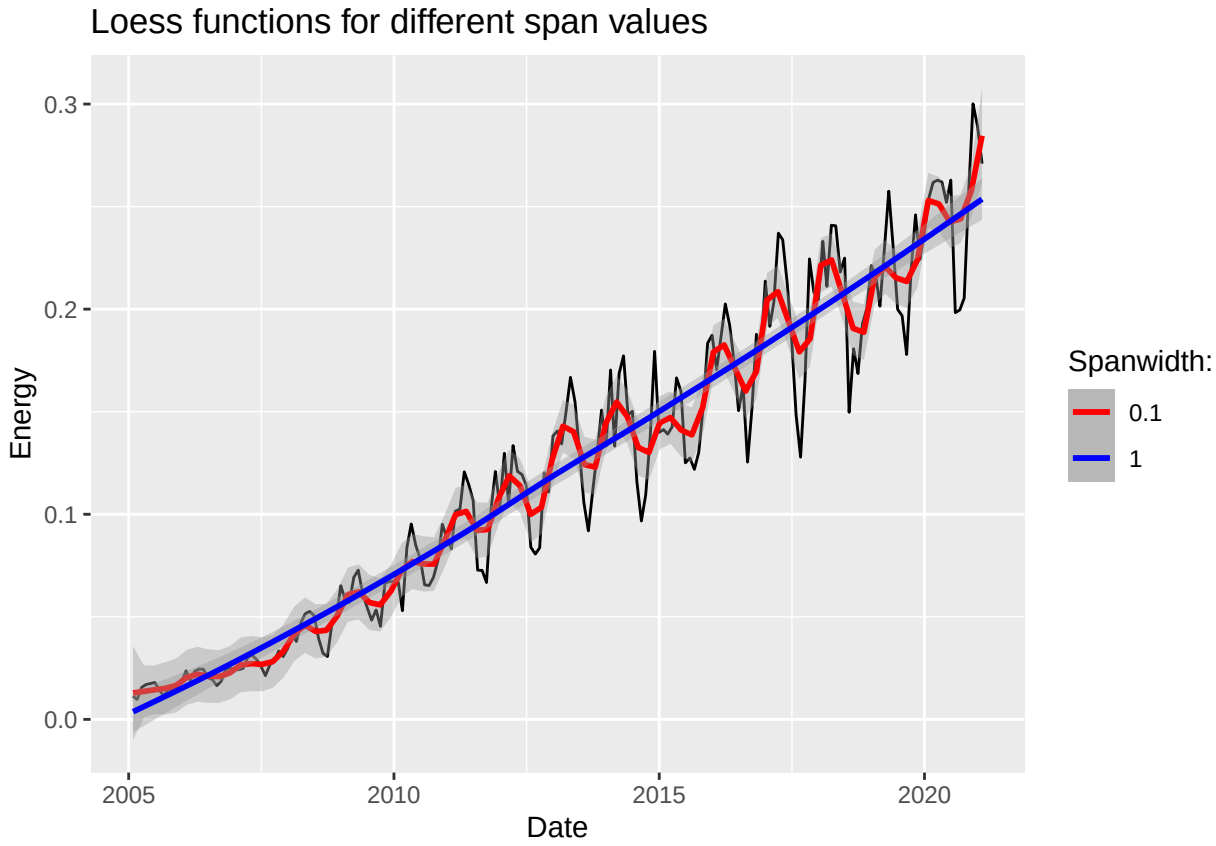
```
## Classes 'data.table' and 'data.frame':   193 obs. of  5 variables:
##  $ MSN        : chr  "WYTCBUS" "WYTCBUS" "WYTCBUS" "WYTCBUS" ...
##  $ Date       : Date, format: "2005-01-31" "2005-02-28" ...
##  $ Energy     : num  0.01132 0.00966 0.01561 0.01697 0.01746 ...
##  $ Source     : int  10 10 10 10 10 10 10 10 10 10 ...
##  $ Description: chr  "Wind Energy Production" "Wind Energy Production" "Wind Energy Production" "Wind
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
ggplot(wind, aes(Date, Energy)) +
    geom_point(size = 1, aes(colour = month(Date))) +
    geom_line() +
    ggtitle("Energy production from wind between 2005 and 2020") +
    ylab("Energy in Quadrillion Btu for each month")
```




```
spans <- data.frame(val = c(0.1, 1), col = c("red", "blue"))
ggplot(wind, aes(Date, Energy)) +
    geom_line() +
    geom_smooth(method = "loess", span = spans[1, 1],
        aes(col = as.character(spans[1, 1]))) +
    geom_smooth(method = "loess", span = spans[2, 1],
        aes(col = as.character(spans[2, 1]))) +
    scale_colour_manual(name = "Spanwidth:", values = spans[, 2]) +
    ggtitle("Loess functions for different span values")
```

```
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
```



- **Einfluss der Spannweite (span):** Wie in der oberen Grafik zu erkennen ist, sorgt eine kleinere span für eine höhere Annäherung an die einzelnen Datenpunkte, wodurch es zu einer geringeren Glättung kommt. Bei höherer span kommt es zum gegenteiligen Effekt. Die Glättung ist sehr stark und bei diesem Datensatz reicht ein Wert von 1 aus, dass die resultierende Lowess Funktion einer Linearfunktion ähnelt.
- **Inhaltliche Interpretation der Trends:** Wie ersichtlich ist, folgt die Lowess Kurve mit der Spannweite 0.1 einem jährlichen Zyklus (ein lokales Minimum / Maximum pro Jahr), während die Lowess Kurve mit der Spannweite von 1 dem generellen Trend über mehrere Jahre relativ linear folgt. Je nachdem welches Modell erwünscht ist wird die Spannweite entsprechend gewählt.
- **Nutzen von upper / lower smoothing:** Upper und Lower smoothing bietet zusätzlich zur Lowess Glättung Information bezüglich der Streuung. So wird ein Korridor (wie in der Abbildung ersichtlich) gezeichnet, in dem Kurve sehr wahrscheinlich bei gegebener Datenlage liegt, vergleichbar mit einem Konfidenzintervall. Upper und Lower smoothing ist in der Abbildung bereits für beide Lowess Glättungen eingezeichnet (graue Bänder).

2.Beispiel:

```
wind_rob <- wind
wind_outliers <- wind
rob_filter <- robust.filter(wind$Energy, width = 5)
str(rob_filter)
```

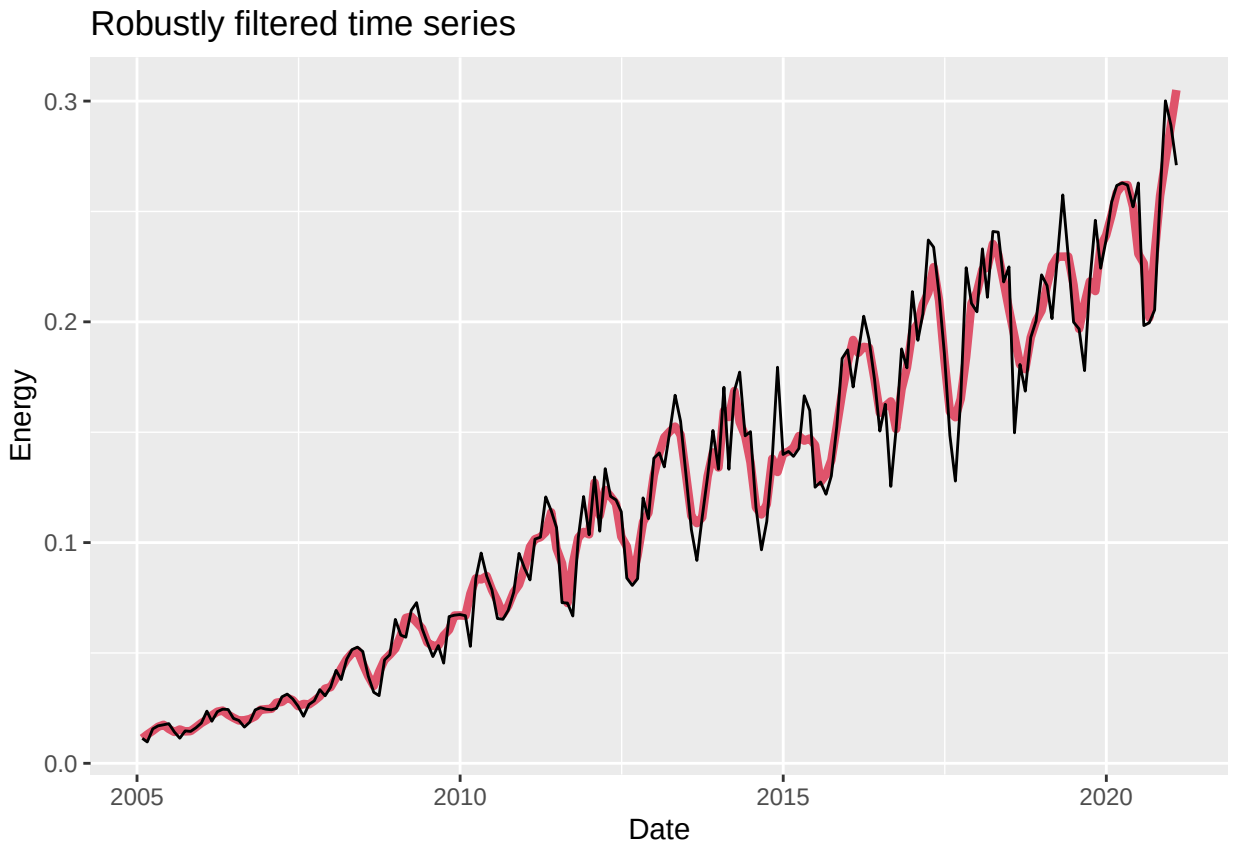
```
## List of 19
##  $ level      : num [1:193] 0.0113 0.013 0.0147 0.0165 0.0175 ...
##  $ slope      : num [1:193] 0.001711 0.001711 0.001711 0.000932 0.000497 ...
```

```
##  $ sigma      : num [1:193] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 ...
##  $ ol         : num [1:193] 0 0 0 0 0 0 0 0 0 0 ...
##  $ level.shift: num [1:193] 0 0 0 0 0 0 0 0 0 0 ...
##  $ y          : num [1:193] 0.01132 0.00966 0.01561 0.01697 0.01746 ...
##  $ width      : num 5
##  $ trend      : chr "RM"
##  $ scale      : chr "QN"
##  $ outlier    : chr "T"
##  $ shiftd     : num 2
##  $ wshift     : num 2
##  $ lbound     : num 0.1
##  $ p          : num 0.9
##  $ adapt      : num 0
##  $ max.width  : num 5
##  $ online     : logi FALSE
##  $ extrapolate: logi TRUE
##  $ ts.name    : chr "wind$Energy"
##  - attr(*, "class")= chr "robust.filter"
```

```
rob_filter$ol # outliers
```

```
##   [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [38] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [112] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [149] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [186] 0 0 0 0 0 0 0 0
```

```
wind_rob$Energy <- rob_filter$level
wind_outliers$ol <- rob_filter$ol
wind_outliers <- subset(wind_outliers, ol != 0)

ggplot() +
geom_line(data = wind_rob, aes(Date, Energy), colour = 2, size = 1.5) +
geom_line(data = wind, aes(Date, Energy)) +
geom_point(data = wind_outliers, aes(Date, Energy), size = 1) +
ggtitle("Robustly filtered time series")
```

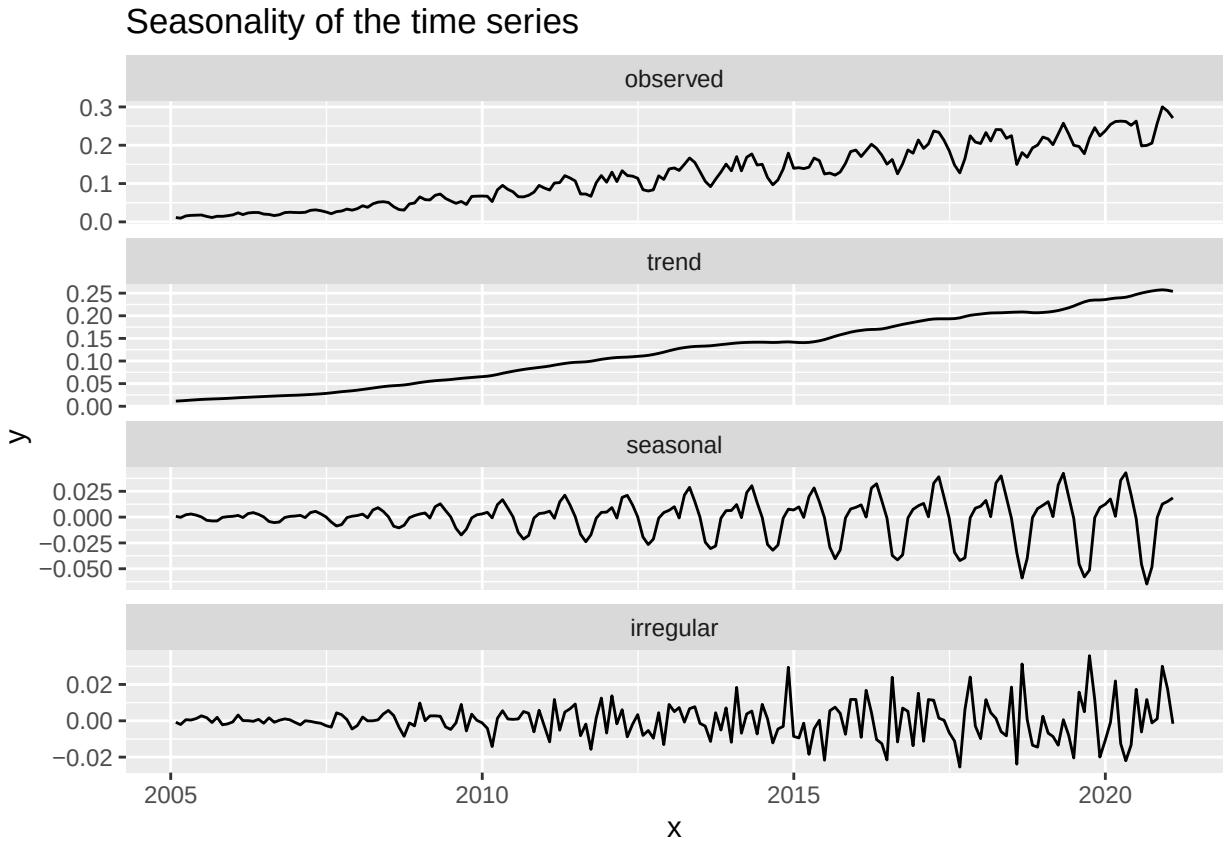


Auch nach mehreren Versuchen war rob_filter$ol immer ein 0-Array, was bedeutet, dass keine Outlier erkannt wurden und es somit keine Outlier in dieser Reihe gibt. Würden Outlier erkannt, würden sie automatisch durch meinen R-Code eingezeichnet.

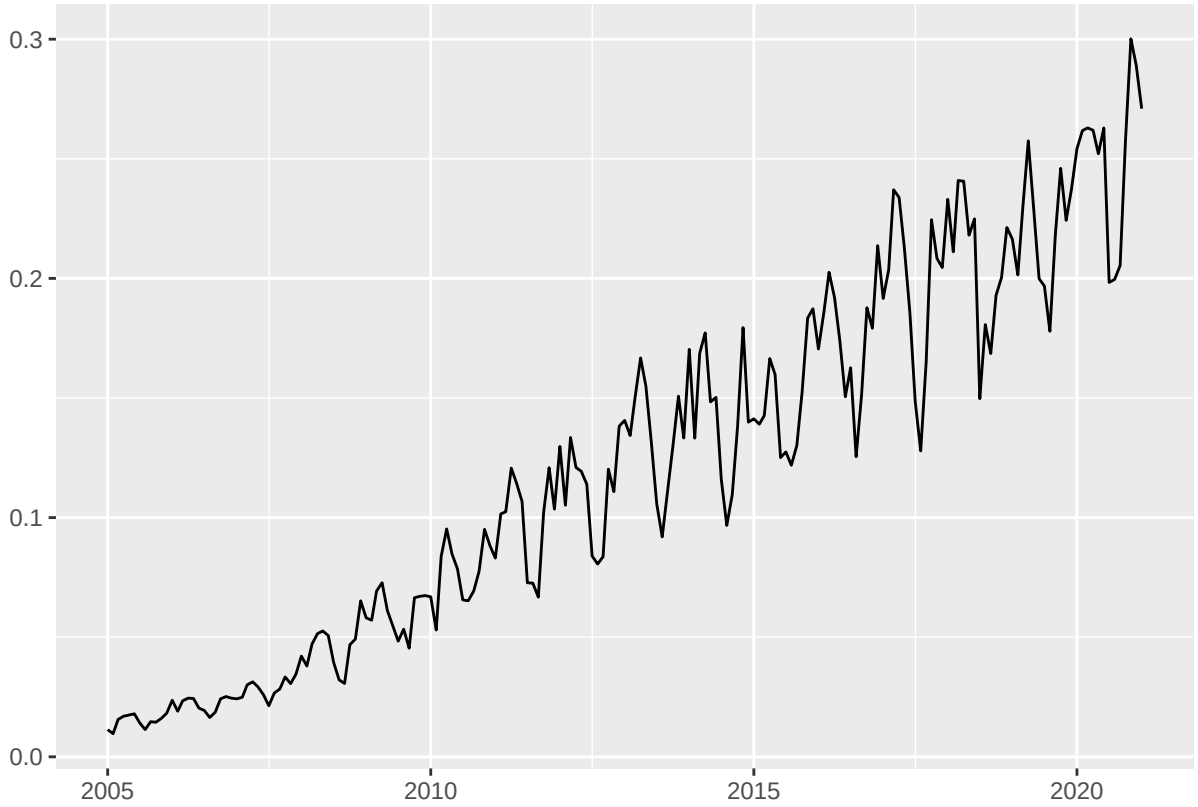
- **Schätzer verwendet von robust.filter():** robust.filter nutzt zur Berechnung den Repeated Mean, da dieser relativ einfach zu berechnen ist und zusätzlich auch recht aussagekräftig ist. Hierbei werden lokal, also in einer Umgebung mit der Länge 2 * Breite (width) + 1, Repeated Means gebildet und so eine Linearfunktion für jeden Punkt geschätzt (Mittelwert und Steigung).
- **Prinzip der Ausreißererkennung:** Ein Ausreißer wird bei dieser Berechnung erkannt, falls er signifikant, in diesem Fall $>$ 2 * /sigma, der Erwartung widerspricht. Aus diesem Grund wurden die Steigung und der Mittelwert mittels Repeated Mean berechnet. Es werden damit all jene Residuen entfernt, die außerhalb dieses Konfidenzintervalls für ihren jeweiligen Index liegen.
- **Interpretation der Ausreißer:** Ausreißer sind jene Werte, die in der festgelegten Umgebung 2 * width + 1, also width Werte davor und width Werte danach signifikant vom Mittelwert abweichen und damit das Modell verfälschen würden. Die Ausreißer sind also Werte, die aufgrund anderer nicht im Modell dargestellter Gründe von der Erwartung stark abweichen.

3.Beispiel:

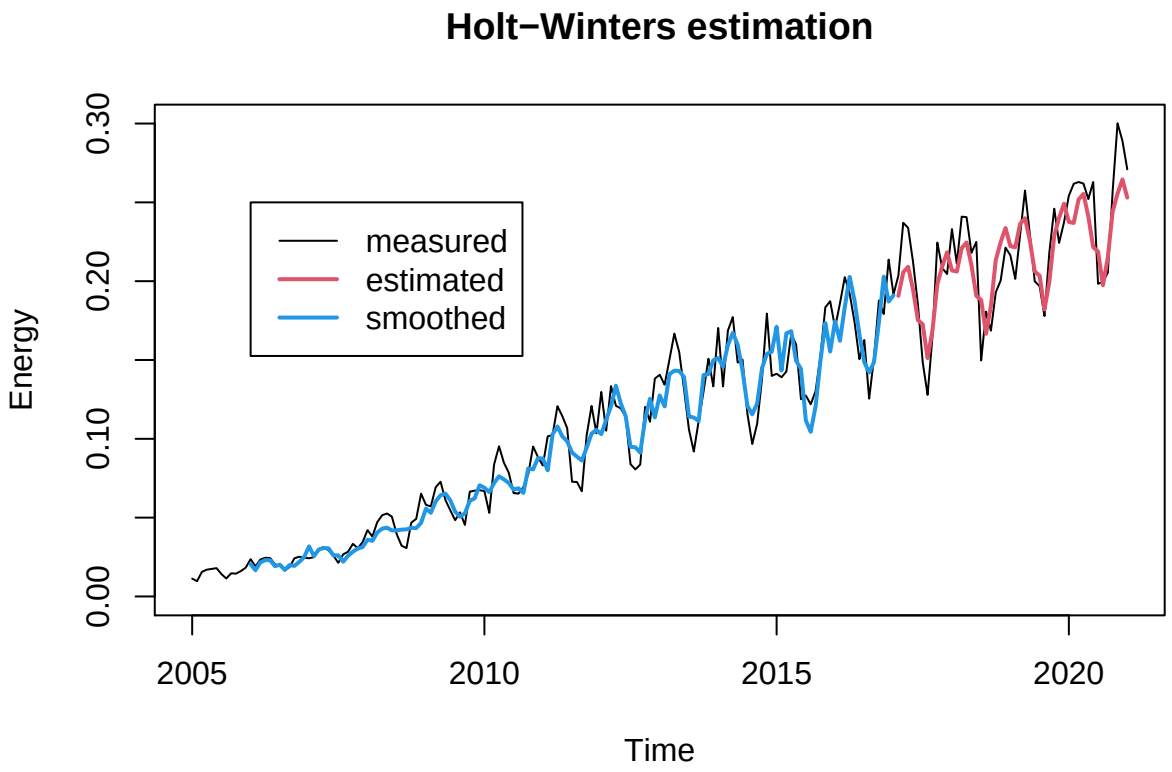
```
ggsdc(wind, aes(Date, Energy), method = "seas", start = "2005", frequency = 12) +
    geom_line() +
    ggtitle("Seasonality of the time series")
```




```
autoplot(wind_ts)
```



```r
ts_base <- window(wind_ts, end = c(2017, 1))
ts_topredict <- window(wind_ts, start = c(2017, 1))
hw <- HoltWinters(ts_base)
predicted <- predict(hw, n.ahead = 48)

plot(wind_ts, xlim = c(2005, 2021), ylim = c(0, 0.3), lty = 1, lwd = 1,
    ylab = "Energy", main = "Holt-Winters estimation")
lines(predicted, col = 2, lwd = 2)
lines(hw$fitted[, 1], col = 4, lwd = 2)
legend(2006, 0.25, col = c(1, 2, 4),
    legend = c("measured", "estimated", "smoothed"),
    lty = c(1, 1, 1), lwd = c(1, 2, 2))
```



- **Saisonale Schwankungen und deren Interpretation:** Saisonale Schwankungen sind klar erkennbar, wobei in den Wintermonaten wesentlich mehr Energie produziert wird als in den Sommermonaten. Dies war auch zu erwarten, da in den Wintermonaten mehr Energie benötigt wird (Heizung, Licht, etc.) als in den Sommermonaten und die Windräder je nach Bedarf auch recht flexibel Strom produzieren können. Sollte also kein Strom benötigt werden, können die Windräder problemlos abgestellt werden. Aus diesem Grund kommt es zu saisonalen Schwankungen, die sich mit unserer Erwartung decken.
  Ein Trend ist deutlich erkennbar, dieser folgt einem relativ linearen Anstieg, der etwa auf die Forcierung auf erneuerbare Energiequellen und den allgemein höheren Strombedarf erklärbar ist.
  Die Residualwerte, also die Restkomponente folgt keinem klaren Muster und dürfte daher nur noch white noise sein. Auf den ersten Blick erkennbar ist allerdings, dass die Abweichungen zu späteren Zeitpunkten höher sind als am Anfang. Die Restkomponente selbst kann auch noch auf mittels der oben genannten Verfahren untersucht werden, um sicherzugehen, dass keine Muster oder Trends mehr vorhanden sind.
- **Prognose mit Holt-Winters:** Die Prognose für diese Zeitreihe funktioniert recht gut. Dies ist dem annähernd linearen Trend und der regelmäßigen Saisonalität zu verdanken.
  Die nächsten 4 Jahre werden vergleichsweise gut geschätzt (rote Linie), wenn auch der Peak Anfang

2021 doch recht weit vom Modell entfernt liegt. Abgesehen von dieser Abweichung liegt die Schätzung immer recht nahe an den gemessenen Werten. Dasselbe gilt für die Glättung (blaue Linie) vor der Prognose. Auch hier liegt die Glättung sehr nahe an den realen Werten (schwarze Linie).

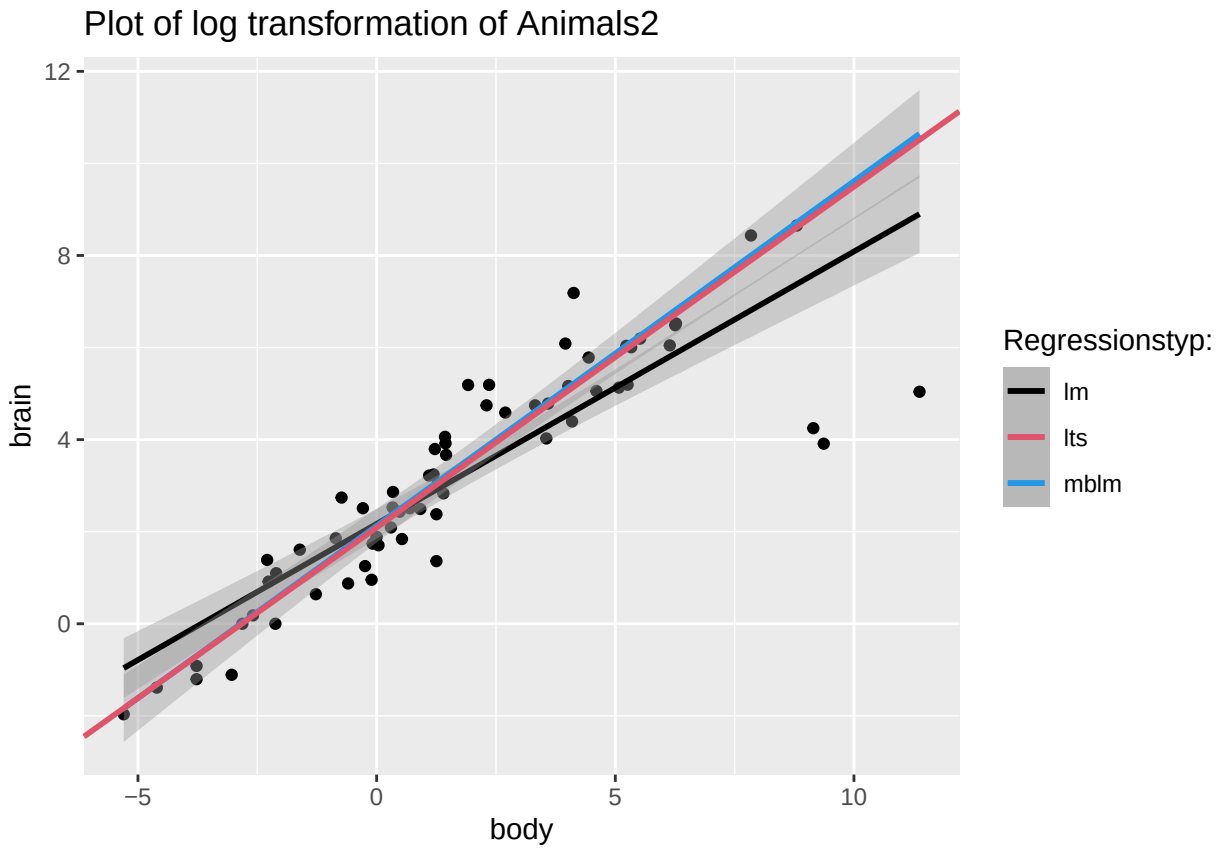
4.Beispiel:

```
data(Animals2)
library(MASS)

anims_log <- log(Animals2)
str(anims_log)
```

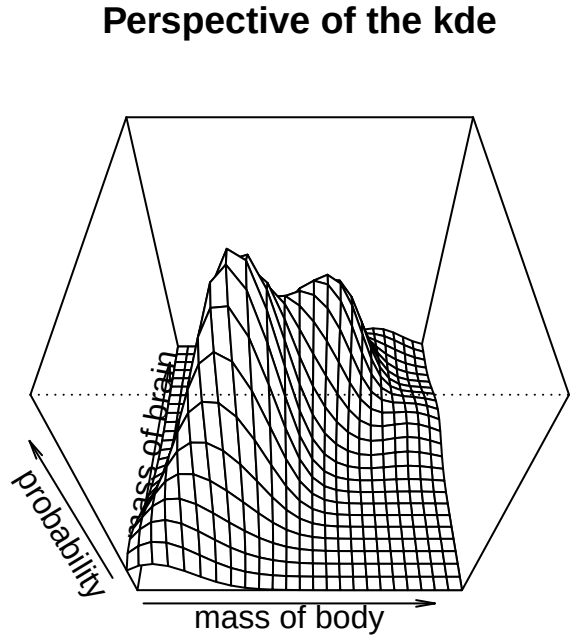
```
## 'data.frame':    65 obs. of  2 variables:
##  $ body : num  0.3001 6.142 3.5926 3.32 0.0392 ...
##  $ brain: num  2.09 6.05 4.78 4.74 1.7 ...
```

```
mblm_gg <- function(..., weights = NULL) mblm::mblm(...)
lts <- ltsReg(anims_log$brain ~ anims_log$body)

ggplot(anims_log, aes(body, brain)) +
    geom_point() +
    geom_smooth(method = "lm", aes(col = "lm")) +
    geom_smooth(method = mblm_gg, aes(col = "mblm")) +
    geom_abline(
        aes(intercept = coef(lts)[1],slope = coef(lts)[2], col = "lts"),
        show.legend = FALSE, size = 1) +
    scale_colour_manual(name = "Regressionstyp:", values = c(1, 2, 4)) +
    ggtitle("Plot of log transformation of Animals2")
```

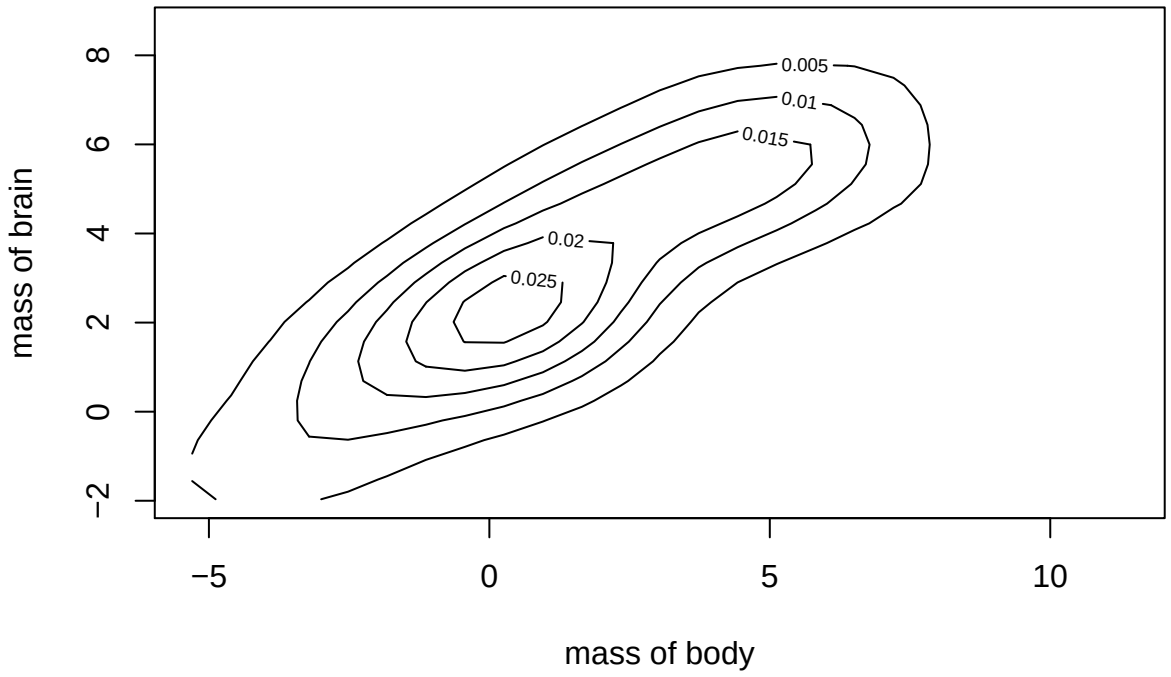
```
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
```




```
kde <- kde2d(anims_log$body, anims_log$brain)
persp(kde, theta = 0, phi = 50, xlab = "mass of body", ylab = "mass of brain",
    zlab = "probability", main = "Perspective of the kde")
```




```
contour(kde, xlab = "mass of body", ylab = "mass of brain",
    main = "Contour of the kde")
```



- **Eignung der Geraden:** Die herkömmliche Least-Squares Gerade ist weniger gut für diese Datenlage geeignet, da sie relativ stark von den Hebelpunkten verzerrt wird. Dies macht sich vor allem durch eine geringere Steigung bemerkbar.
  Im Gegenteil dazu sind die Unterschiede zwischen mblm und lts vergleichsweise gering, die Auswahl zwischen diesen beiden ist daher für diesen Datensatz nebensächlich.
- **Berechnungsart der bestgeigneten Gerade:** Die bestgeeignete Gerade ist jene nach Siegel. Das Berechnungsmodell dahinter: Die Gerade nach Theil mit Wiederholter Medianberechnung. Dies erfolgt folgendermaßen: Für jeden Punkt wird der Median der Verbindungen zu allen anderen Punkten gewählt. Hier bereits mit einem "einfachen" Median abzubrechen würde der Gerade nach Theil entsprechen. Für einen höheren Bruchkpunkt kann nun die Gerade nach Siegel verwendet werden. Hier wird der Median über den Medianen der der Steigungen für den jeweiligen Punkt i berechnet. Dadurch fließt jede Steigung 2 mal in die Berechnung mit ein. Alpha wird über den Median von allen y Werten minus den Steigungen mal der zugehörigen x Werte berechnet. Da bei dieser Methode der Median über mehrere Mediane gebildet wird, heißt diese Methode auch Wiederholte Median Gerade. Diese Methode ist sehr beliebt, da sie einen sehr hohen Bruchpunkt von 50% Prozent besitzt. Das beudeutet es darf bis zu 50% Ausreißer in den Daten geben, was das Maximum für robuste Geraden darstellt.