Datenanalyse

Abgabe 2

Lukas Mahler (Matr. Nr. 11908553)

01.04.2021

Vorinformationen:

Der Fließtext der Abgabe wurde in Deutscher Sprache verfasst, der R-Code selbst ist wie üblich in Englischer Sprache verfasst. 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.

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

1. Beispiel

Als Datensatz dient die Primärenergieproduktion der USA, abrufbar unter https://www.eia.gov/totalenergy/data/browser/index.php?tbl=T01.02#, aufgeschlüsselt nach Energiequelle. Nach Laden der Daten ergibt sich folgendes Bild:

```
set.seed(69) # same seed for reproduceability
library(data.table) # used for simplified reading into data frame
library(ggplot2) # used for simplified plotting
library(zoo) # for date reading

# Data from:
# https://www.eia.gov/totalenergy/data/browser/index.php?tbl=T01.02#
# Primary Energy Production by Source in the USA
setwd("Z:/Benutzer/OneDrive - TU Wien/Uni/4.Semester/Datenanalyse/exercises/exercise2")
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 = "chr")
)
```

```
## Warning: Column 'YYYYMM' was requested to be 'chr' but fread encountered the following error:
##  no method or default for coercing "character" to "chr"
## so the column has been left as type 'character'
```

```
str(data_raw) # view basic structure of the data
```

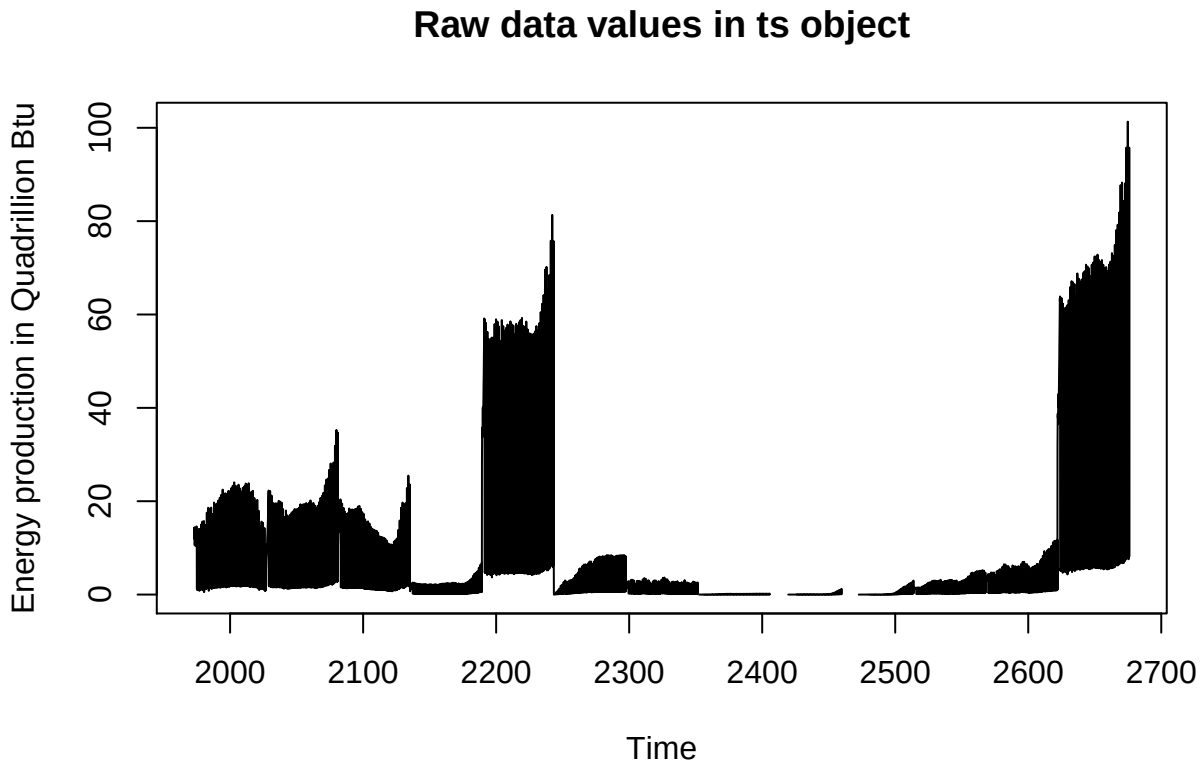
```
## Classes 'data.table' and 'data.frame':   8437 obs. of  5 variables:
##  $ MSN        : chr  "CLPRBUS" "CLPRBUS" "CLPRBUS" "CLPRBUS" ...
```

```
##  $ 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>
```

```
plot(ts(data_raw$Energy, start = c(1973, 1), frequency = 12),
    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")
)
# 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.

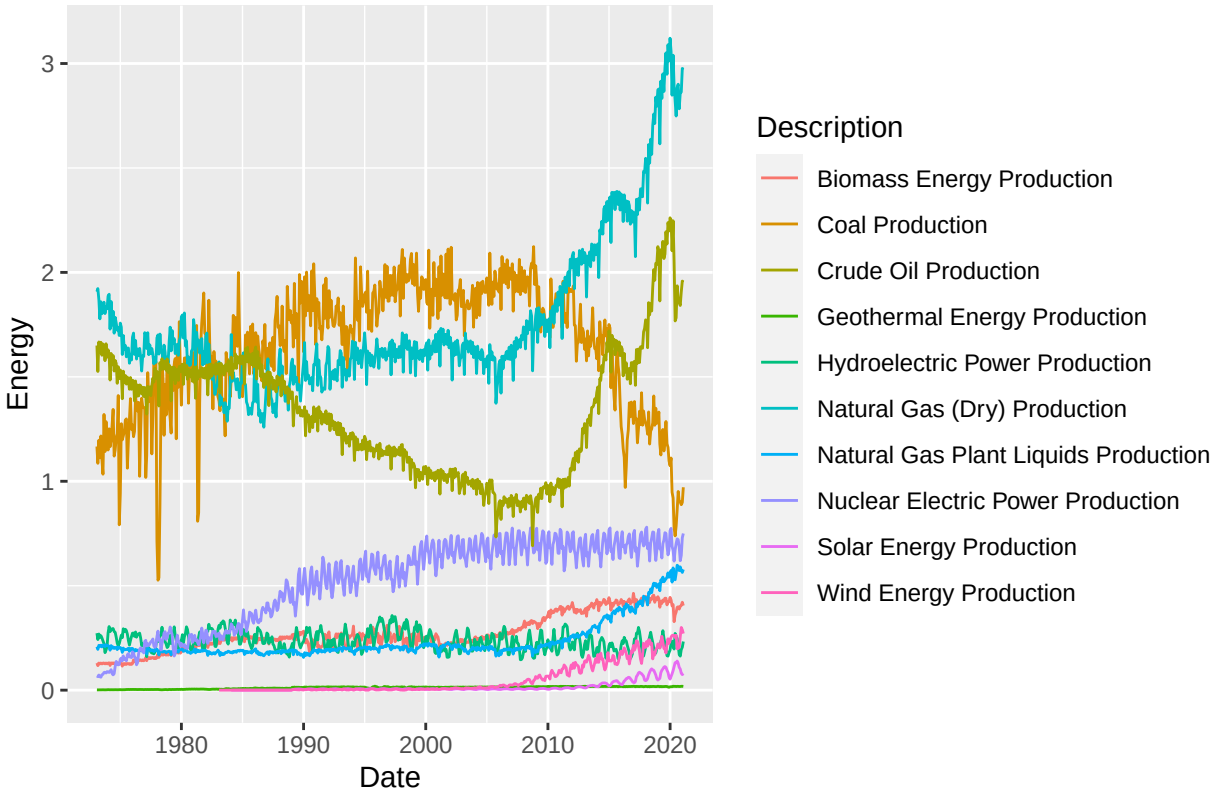
```
# Remove all summaries for a year
data_monthly <- subset(data_cleaned, !is.na(Date))
data_monthly_total <- subset(data_monthly, grepl("^Total", Description))
data_monthly <- subset(data_monthly, !grepl("^Total", Description))

# TODO (optional): keep dates when converting to Date format
data_yearly <- subset(data_cleaned, is.na(Date))
data_yearly_total <- subset(data_yearly, grepl("^Total", Description))
data_yearly <- subset(data_yearly, !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("")
```

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

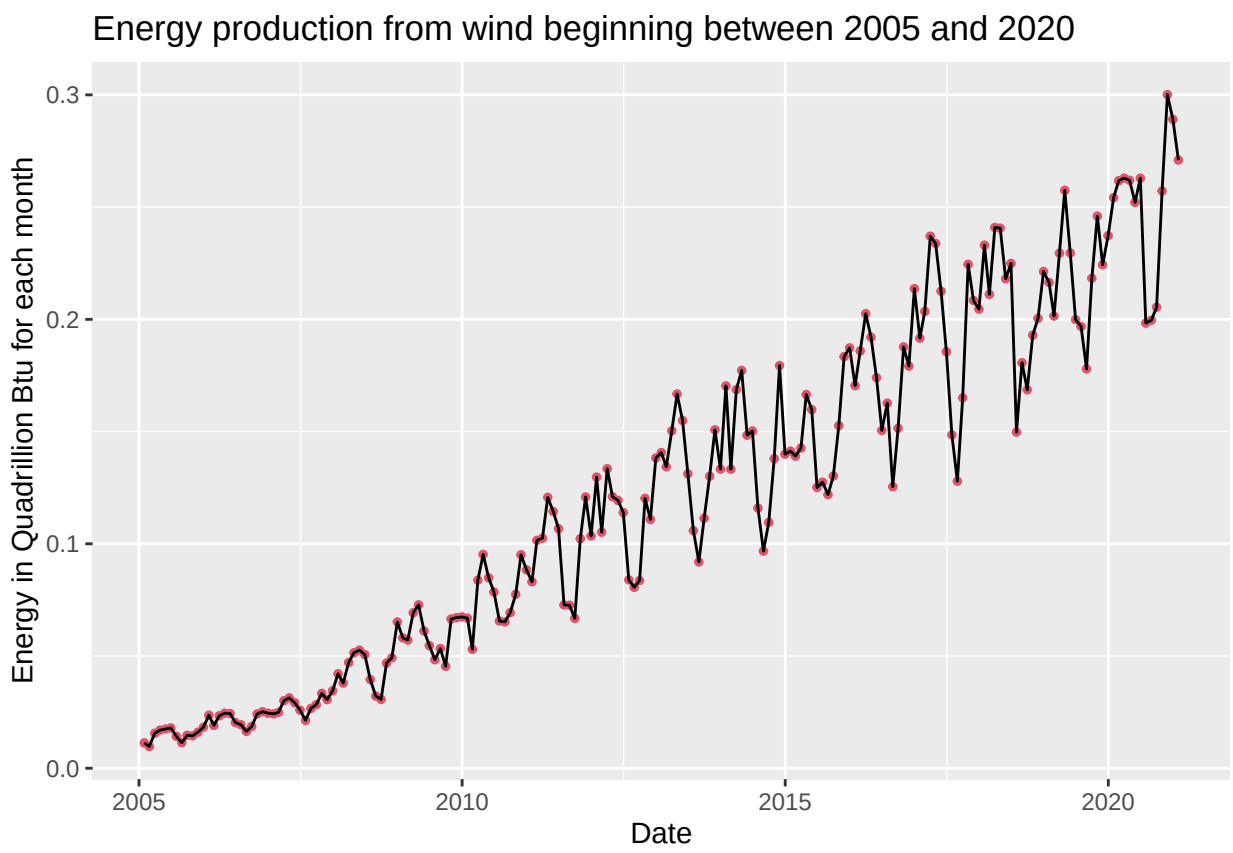


Nun sind die Daten aufbereitet, um als Timeseries Objekte eingelsesen und in Modelle verarbeitet zu werden. Für die weiteren Aufgaben werden wir die produzierte Energie durch Wind ab 2005 betrachten.

```
wind <- subset(data_sources$"10", Date > as.Date("2005-01-01"))
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>
```

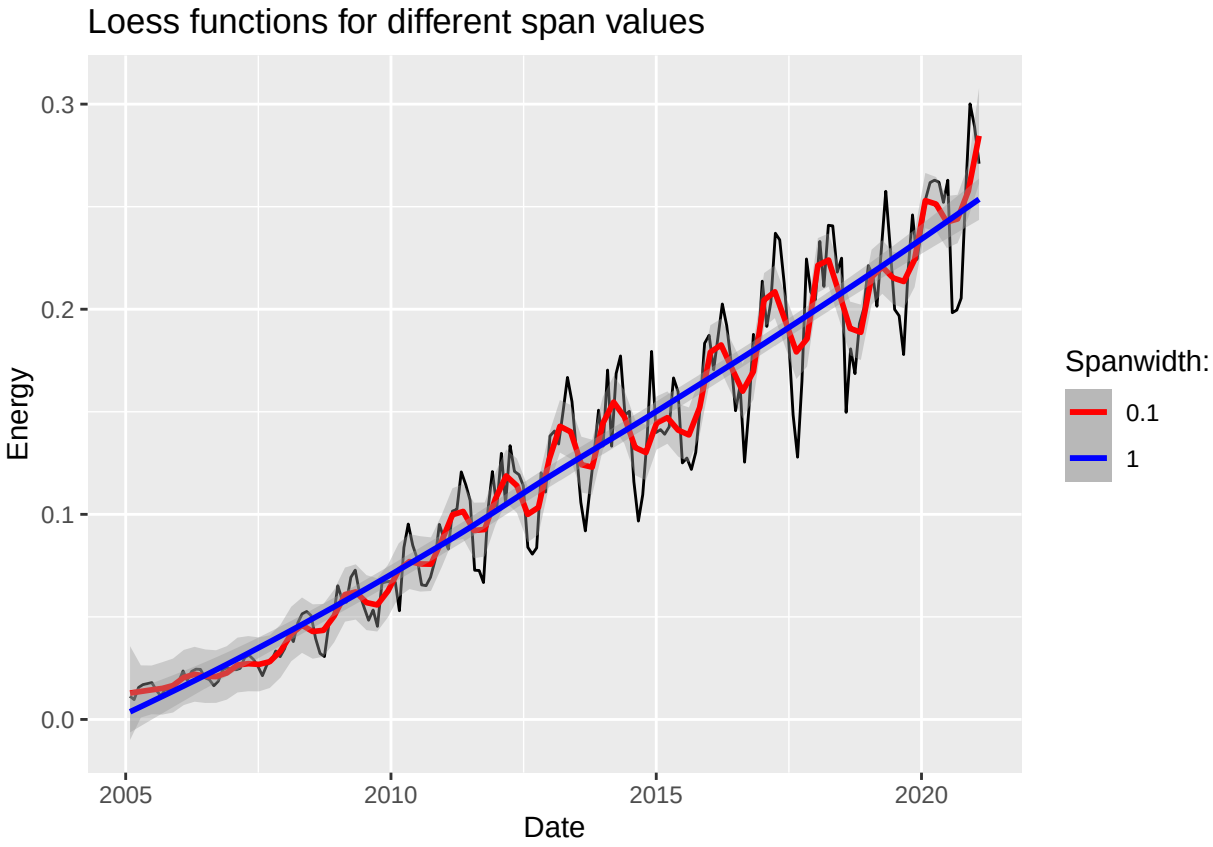
```
ggplot(wind, aes(Date, Energy)) +
    geom_point(size = 1, col = 2) +
    geom_line() +
    ggtitle("Energy production from wind beginning between 2005 and 2020") +
    ylab("Energy in Quadrillion Btu for each month")
```




```
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 kommt reicht ein Wert von 1 aus, dass die Glättung einer Linearfunktion ähnelt.
- **Inhaltliche Interpretation der Trends:** Wie ersichtlich ist, folgt die Lowess Kurve mit der Spannweite `{r} spans[1,1]` einem jährlichen Zyklus (ein lokales max / min pro Jahr), während die Lowess Kurve mit der Spannweite von 1 dem generellen Trend über mehrere Jahre folgt. Je nachdem welches Modell erwünscht ist würde die Spannweite entsprechend gewählt werden.
- **Nutzen von upper / lower smoothing:**

2.Beispiel:

3.Beispiel:

4.Beispiel: