Praxislabor der AG Digitale Geschichtswissenschaft

RStudio – Geschichtswissenschaft zwischen Stoppwörtern und Plots

DONNERSTAG, 21 SEP 2023, 16:00–18:00 Uhr, Seminargebäude S420

Liebe Teilnehmende des Workshops,

Ich freue mich sehr über Ihre Teilnahme an meinem Workshop „RStudio – Geschichtswissenschaft zwischen Stoppwörtern und Plots“! In nur 90 Minuten möchte ich Ihnen bzw. Euch RStudio als geschichtswissenschaftliches Werkzeug präsentieren und Sie bzw. Euch dazu befähigen, eigene Analysen damit durchzuführen. Grundlage ist ein Quellenkorpus, der aus 16428 Tweets besteht. Sie beinhalten bestimmte Worte rund um den 60. Jahrestag des Mauerbaus und wurden zwischen dem 6.8.2021 und dem 14.8.2021 veröffentlicht. Sie wurden mithilfe eines Suchbefehls über RStudio automatisch „abgegriffen“. Der damalige Suchbefehl wurde am Abend des 14.08.2021 um ca. 19:12 Uhr „abgeschickt“ und lautete:

```
tweets_search1 <- search_tweets(q = "#Mauerbau OR #60JahreMauerbau OR #DDR OR #Mauer OR #Berlin OR #BerlinerMauer OR Mauerbau", n = 18000, include_rts = FALSE)
```

Die quellenkritische Einordnung dieser Suchanfrage und seiner Ergebnisse findet im Workshop statt. Auf den über RStudio erhaltenen Datensatz mit dem Titel „60JahreMauerbau.csv“ können Sie bzw. könnt Ihr auf der Internetseite, die uns den gesamten Workshop über begleitet und über die Sie bzw. Ihr diese Datei heruntergeladen haben bzw. habt, zugreifen. Der Dateityp „CSV“ steht für „Comma-separated values“, was meint, dass es sich beim Inhalt um eine gewisse Menge an Daten handelt, die in einer Tabellenstruktur unterschiedliche Spalten durch Kommata getrennt aufweist. Für die Tweets, die wir über RStudio und die API-Schnittstelle von Twitter abgreifen, ergeben sich (derzeit) 90 Spalten einer jeden Zeile, also jeden einzelnen Tweets.

Damit Sie bzw. Ihr dies sehen können bzw. könnt, müssen Sie bzw. müsst Ihr zunächst R und RStudio auf Ihrem eigenen Endgerät installieren. Ich bin „Windows-sozialisiert“ und dementsprechend mit der Installation auf Windows-Geräten am ehesten vertraut. Bei Fragen zu Mac- oder Linux-Installationen kann ich also ebenfalls nur auf die digitalen Suchmaschinen zurückgreifen oder lediglich rudimentär helfen! R und RStudio finden Sie bzw. findet Ihr über die beiden folgenden Links, die ebenfalls auf „unserer“ Website auffindbar sind:

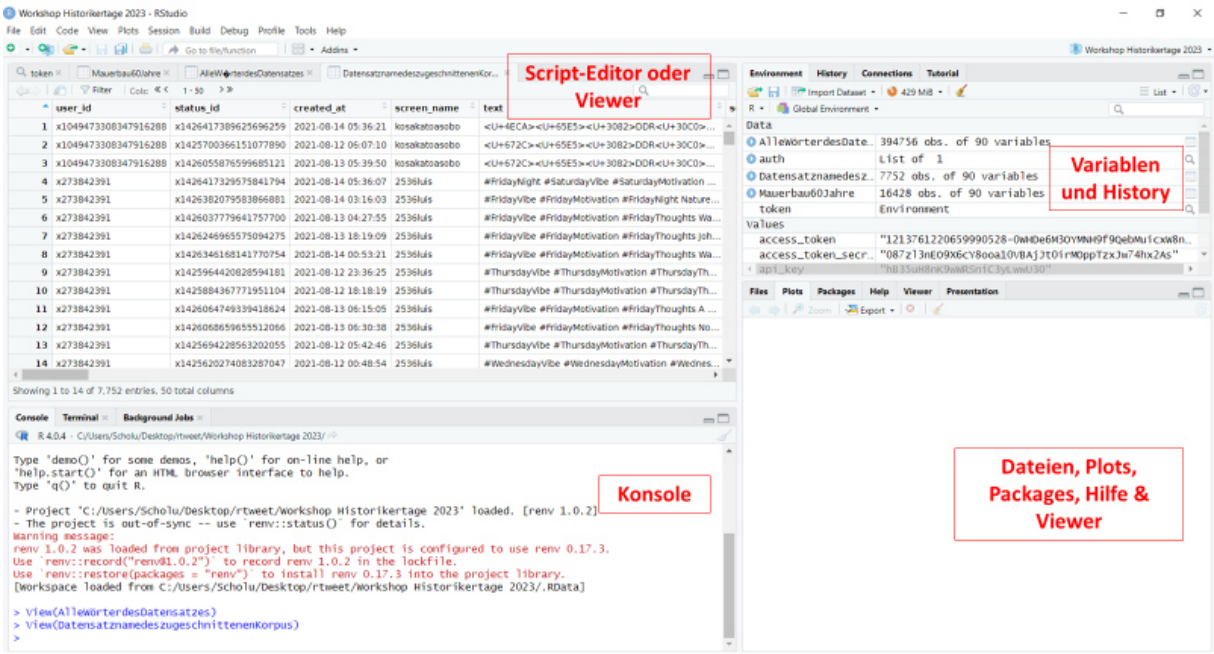
R: https://ftp.gwdg.de/pub/misc/cran/ (Oben auf der Seite sind mittig die Optionen „Download R for“ verschiedene Betriebssysteme auffindbar)

RStudio: https://posit.co/download/rstudio-desktop/?utm_source=downloadrstudio&utm_medium=Site&utm_campaign=home-hero-cta (Auf der rechten Seite kann unter 2: Install RStudio der blaue Button „Download RStudio for

Windows“ gefunden werden. Weitere Betriebsversionen sind weiter unten auf der Website aufgeführt!)

Beide Dateien können über eine Standard-Installation auf Ihren bzw. Euren Endgeräten ganz einfach installiert werden. Ein Shortcut auf dem jeweiligen Desktop vereinfacht das Arbeiten mit RStudio weiter. R selbst kann im Hintergrund „arbeiten“. Danach sind wir startklar. Starten Sie bzw. startet bitte RStudio. Die Bedienungsoberfläche befremdet auf den ersten Blick, ist jedoch klar strukturiert. Eine Einführung finden Sie bzw. findet Ihr auf YouTube unter dem folgenden Link: https://www.youtube.com/watch?v=tyvEHQszZJs

Für den Workshop genügt, die vier großen Arbeitsbereiche unterscheiden zu können.



[Quelle: Eigener Screenshot]

Grundlage für das Arbeiten mit diesen Tabellen bzw. Datensätzen ist das Anreichern von Ihrem bzw. Eurem recht rudimentären RStudio-Arbeitsplatz (*Workspace*) mit einzelnen Erweiterungspaketen (*Packages*), die zusätzliche Funktionen mit sich bringen. Installieren Sie zunächst die folgenden „*Packages*“:

```
install.packages("rtweet")
install.packages("tidyverse")
install.packages("tidytext")
install.packages("dplyr")
install.packages("tm")
install.packages(„ggplot2“)
install.packages(„wordcloud“)
install.packages(„RColorbrewer“)
install.packages(„cluster“)
install.packages(„text2vec“)
```

Sie können bzw. Ihr könnt diese „Codes“ einfach kopieren und in die Konsole Eures RStudio-Arbeitsplatzes einfügen. Durch das Drücken der Enter-Taste führt Ihr bzw. führen Sie die

Befehle aus. Während die Installation von „*Packages*“ lediglich einmal ausgeführt werden muss, müssen Sie die jeweiligen Packages bei jedem Neustart von RStudio erneut aus der Bibliothek (*library*) laden bzw. „reaktivieren“:

```
library(rtweet)
library(tidyverse)
library(tidytext)
library(dplyr)
library(tm)
library(ggplot2)
library(wordcloud)
library(RColorBrewer)
library(cluster)
library(text2vec)
```

Die von mir zur Verfügung gestellte Datei können Sie bzw. könnt Ihr nun mithilfe des folgenden Befehls in Ihre bzw. Eure jeweiligen RStudio Programme „hochladen“:

Befehl in der Theorie

```
Datensatzname <- read_csv("Dateipfad zur Datei")
```

Praxisbeispiel

```
Mauerbau60Jahre <- read_csv("C:/Users/scholu/Desktop/rtweet/TDE2020/60JahreMauerbau.csv")
```

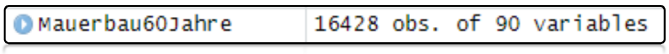
Sie müssen Ihren eigenen Dateipfad finden!

!Achten Sie darauf, ~~dass~~ Ihr gewählter Datensatzname nicht mit einer Zahl beginnt!

Ein Video-Tutorial für andere Wege des Einlesens finden Sie bzw. findet Ihr unten dem folgenden Link: https://www.youtube.com/watch?v=Ia5Q1LlCCFc. Darüber hinaus kann dieser Link helfen: https://statologie.de/csv-importieren-r/.

Hatten Sie bzw. hattet Ihr Erfolg, erscheint ein Datensatz oben rechts bei „Environment“:

[Quelle: Eigener Screenshot]



Diesen Datensatz können Sie sich bzw. könnt Ihr Euch nun in aller Ruhe angucken – einige der 90 Spalten sind überaus interessant, andere absolut redundant. Machen Sie sich selbst ein Bild! Durch Upgrades der Packages kann es sein, dass Sie lediglich 50 der 90 Variablen explorieren können. Das ist eine Einschränkung, aber kein Weltuntergang.

Untersuchungszeitraum begrenzen

Bevor die Spalten einen Erkenntnisgewinn für wie auch immer geartete geschichtswissenschaftliche Fragestellungen ermöglichen können, gilt es erst einmal, den Datensatz auf einen bestimmten Zeitraum zu begrenzen, ist er doch vom Abend des 14.8.21 ausgehend für die letzten sieben Tage erhoben worden. Der Befehl zum „Beschneiden“ eines solchen Korpus lautet wie folgt:

```
DatensatznamedeszugeschnittenenKorpus <- filter(Datensatzname, "JJJJ-MM-TT 00:00:00" < created_at & created_at < "JJJJ-MM-TT 00:00:00")
```

Frequenz der Tweets

Nun geht es darum, die Frequenz der Tweets in diesem Zeitraum abzubilden, um einen Überblick über die Intensität potentieller Diskurse zum 60. Jahrestag des Mauerbaus zu erhalten. Dafür gibt es den folgenden Befehl (Nur die fettgedruckten Begriffe müssen von Ihnen bzw. von Euch modifiziert werden)

:

```
ts_plot(DatensatznamedeszugeschnittenenKorpus, "hours") +
  labs(x = NULL, y = NULL,
       title = "Frequenz von Tweets am 60. Jahrestag des Mauerbaus mit Suchworten",
       subtitle = paste0(format(min(DatensatznamedeszugeschnittenenKorpus $created_at), "%d %B %Y"), " to ", format(max(DatensatznamedeszugeschnittenenKorpus $created_at),"%d %B %Y")),
       caption = "Daten abgerufen mit Twitters API via rtweet") +
  theme_minimal()
```

Nur die **fettgedruckten** Begriffe müssen Sie selbst aussuchen bzw. einsetzen! Den Rest können Sie kopieren.

Häufigste Hashtags

Auch die Zahl der häufigsten Hashtags verrät viel über die Tweets dieses Tages. Sie können mithilfe der folgenden Eingabe als eigener Datensatz angezeigt werden:

```
HashtagDatensatznamedeszugeschnittenenKorpus <- DatensatznamedeszugeschnittenenKorpus %>% unnest_tokens(hashtag, text, "tweets", to_lower = FALSE) %>% filter(str_detect(hashtag, "^#")) %>% count(hashtag, sort = TRUE)
```

Dieser Befehl kann mittlerweile dysfunktional sein. Er verdeutlicht den Unerfolg, der zum Arbeiten mit RStudio, besonders aber Tweets dazugehört. Keine Sorge, das ist nicht der Regelfall!

Häufigste Wörter

Nun sagt eine Tabelle der häufigsten Hashtags etwas, aber nicht alles über unseren Korpus aus. Dafür lohnt der Blick auf alle Wörter, die über alle Tweets hinweg verzeichnet werden konnten. Der Weg dahin ist etwas komplizierter. Zunächst kreieren wir einen neuen Datensatz, in dem jedem Wort eine einzelne Zeile geschenkt wird:

```
AlleWörterdesDatensatzes <- DatensatznamedeszugeschnittenenKorpus %>% unnest_tokens(word, text)
```

Um daraus valide Aussagen ableiten zu können, müssen wir diesen neuen Datensatz aller Stoppwörter entledigen, die uns ansonsten nur aufhalten würden. Dankenswerterweise gibt es dafür bereits Vorlagen. Sie müssen sie nur „aktivieren“:

```
data(stop_words)
```

Dieses Mal ersetzen wir den Datensatz einfach mit seinem „bereinigten“ Pendant:

```
AlleWörterdesDatensatzes <- AlleWörterdesDatensatzes %>% anti_join(stop_words)
```

Darauf aufbauend lassen wir uns die in diesem „bereinigten“ Datensatz häufigsten Wörter anzeigen:

```
HäufigsteWörter <- AlleWörterdesDatensatzes %>% count(word, sort = TRUE)
```

Alternativ kann auch der folgende Befehl genutzt werden:

```
word_counts <- AlleWörterdesDatensatzes %>% group_by(word) %>% summarise(count = n())
```

Ihnen bzw. Euch wird schnell auffallen, dass hier noch immer Worte auftauchen, die wir als Stoppwörter erachten, die aber nicht von der vorgegebenen Liste „stop_words“ abgedeckt werden. Wir müssen also eine Liste schaffen, die aus den bereits existierenden, aber auch unseren eigenen Stoppwörtern besteht. Sie können Sie nennen, wie Sie wollen:

```
custom.stop.words <- bind_rows(data_frame(word = c("https", "t.co", "amp"),
                                          lexicon = c("custom", "custom", "custom")),
                               stop_words)
```

Nun wiederholen wir „Bereinigung“ und Auflistung, allerdings mit den neuen Stoppwörtern:

```
AlleWörterdesDatensatzes <- AlleWörterdesDatensatzes %>% anti_join(custom.stop.words)

HäufigsteWörter <- AlleWörterdesDatensatzes %>% count(word, sort = TRUE)
```

Auch hier kann der alternative Code von vorher genutzt werden. Jetzt haben Sie einen „wahrhaftigen“ Überblick über die Wörter, die innerhalb des zugeschnittenen Datensatzes am häufigsten vorkamen. Weil hier zunächst alle Wörter des Datensatzes abgezählt werden, müssen wir für die grafische Umsetzung eine Grenze setzen. Nehmen wir 20:

```
word_counts <- AlleWörterdesDatensatzes %>% group_by(word) %>% summarise(count = n()) %>% top_n(20, count)
```

Eine grafische Darstellung erfolgt über den folgenden Code:

```
ggplot(word_counts, aes(x = reorder(word, -count), y = count)) +
  geom_bar(stat = "identity") +
  labs(x = "Wörter", y = "Häufigkeit") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  ggtitle("Häufigkeit der verwendeten Wörter")
```

Um hier etwas Farbe reinzubringen, müssen wir den Code erweitern:

```
ggplot(word_counts, aes(x = reorder(word, -count), y = count, fill = count)) +
  geom_bar(stat = "identity") +
  scale_fill_gradient(low = "lightblue", high = "darkblue") +  # Farbpalette anpassen
  labs(x = "Wörter", y = "Häufigkeit") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  ggtitle("Balkendiagramm mit aufgeschlüsselter Häufigkeit")
```

Wir können auch die Frequenz der häufigsten Wörter untersuchen. Dafür müssen wir zunächst die Zeitstempel der Wörter und ihre Häufigkeit zusammenrechnen:

```
word_data3 <- AlleWörterdesDatensatzes2 %>%
  mutate(Hour = format(as.POSIXct(created_at), "%Y-%m-%d %H:00:00")) %>%
  group_by(Hour, word) %>%
  summarise(count = n())
```

Auch hier begrenzen wir es auf die häufigsten, dieses Mal zehn Wörter:

```
top_10_words_per_hour <- word_data3 %>%
  group_by(Hour) %>%
  arrange(desc(count)) %>%
  slice(1:10)
```

sadasd

```
ggplot(top_10_words_per_hour, aes(x = as.POSIXct(Hour), y = count, color = word)) +
  geom_line() +
  labs(x = "Datum und Uhrzeit", y = "Häufigkeit") +
  scale_color_manual(values = rainbow(length(unique(top_10_words_per_hour$word)))) +
  theme_minimal() +
  ggtitle("Entwicklung der Top-10-Wortnutzung pro Stunde")
```

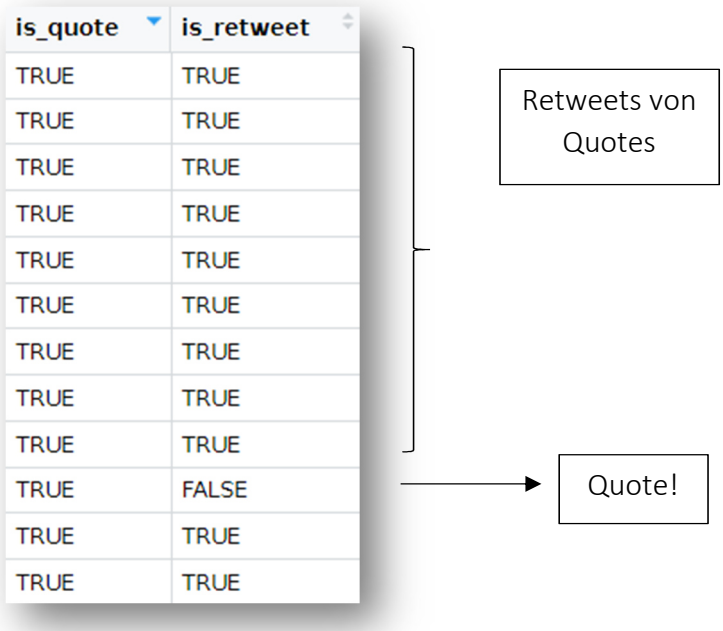
Sie werden bzw. Ihr werdet an dieser Stelle möglicherweise irritiert sein: Den einen Tweet gibt es ja nicht, sondern unterschiedliche Typen. Dem müssen wir natürlich ebenfalls Rechnung tragen, indem wir uns auf die Suche nach Merkmalen machen, die bei der automatisierten Unterscheidung helfen. Denn im Datensatz stehen Retweets neben Quotes und originären ohne Kennzeichnung. Sie können bzw. Ihr könnt natürlich Merkmale wie z.B. häufigste Hasthags über alle Tweets hinweg erheben. Für eine genauere Analyse brauchen wir allerdings die unterschiedlichen Tweet-Sorten.

Den einzigen Hinweis auf den Tweet-Typen liefern die drei Variablen „reply_to_status_id“, „is_retweet“ und „is_quote“. Hier finden Sie bzw. findet Ihr unter „reply_to_status_id“ entweder NA (→ keine Antwort) oder eine lange Zahl (→Antwort). Bei den Retweets bzw. Quotes verhält es sich etwas komplexer: hier finden Sie die entweder „FALSE“ oder „TRUE“ als Spalteninhalte vor. Auf den ersten Blick können wir mithilfe einer simplen Addierung uns anzeigen lassen, wie viele Zeilen welches dieser drei Merkmale aufweisen:

```
Versuch1 <- count(Ihr Datensatzname, is_retweet)
Versuch2 <- count(Ihr Datensatzname, is_quote)
Versuch3 <- count(Ihr Datensatzname, reply_to_status_id)
```

Leider ist dieser Vorgehen nicht ganz zielführend. Denn es ist nicht so einfach, als dass „TRUE“ bei „is_retweet“ oder „is_quote“ automatisch bedeutet, Retweet oder Quote vor sich zu

haben. Stattdessen liegt ein Quote nur immer dann vor, wenn unter „is_quote“ „TRUE“ und gleichzeitig unter „is_retweet“ „FALSE“ vorliegt. Weisen beide Spalten „TRUE“ vor, handelt es sich um den Retweet eines Quotes – was wirklich nervig ist. Nicht alle „TRUE“ Einträge bei „is_quote“ sind also wirklich Quotes! Bei Retweets hingegen ist klar, dass „TRUE“ einen Retweet anzeigt. Die folgende Grafik verdeutlicht das Problem:



Wie viele der Quotes eigentlich Retweets sind, wissen wir nicht. Denn die Tabelle „Versuch 1“ spiegelt nur die Gesamtzahl der Retweets wieder, ohne Rückschluss auf Retweets von Quotes.

Sie merken bzw. Ihr merkt – wir brauchen einen Befehl, der uns ausgehend von diesen drei Kategorien die Tweet-Typen automatisch auseinander sortiert. Denn diese händische Kalkulation lässt uns nicht wirklich effektiv weiterarbeiten. Dafür gehen wir wie folgt vor:

Zuerst müssen wir alle NA-Angaben der „reply_to_status_id“-Spalte in die Zahl „0“ umwandeln lassen:

```
VersuchOhneNA <- Ihr Datensatzname %>% mutate_at(vars(reply_to_status_id), ~replace(., is.na(.), 0))
```

Anschließend definieren wir die Bedingungen, unter denen in einem Datensatz eine neue (und damit die 91.) Spalte „aufgemacht wird“, in der sich der Typ des Tweets ablesen lässt:

```
VersuchOhneNAmitTyp <- VersuchOhneNA %>% mutate(Typ = case_when(is_retweet == 'TRUE' ~ 'Retweet', is_quote == 'TRUE' ~ 'Quote', reply_to_status_id == '0' ~ 'Original', reply_to_status_id > '0' ~ 'Reply'))
```

Diesen können wir uns nun anzeigen lassen:

```
Versuch4 <- count(VersuchOhneNAmitTyp, Typ)
```

Nun haben wir eine Tabelle, die die Tweet-Typen für uns sortiert – ein wichtiges Unterscheidungsmerkmal bei der schlussendlichen Analyse der Tweets! Dieses Merkmal kann in eine bereits vorgenommene Untersuchung einfließen: die grafische Aufschlüsselung der Tweet-Frequenz:

```
ts_plot(VersuchOhneNAmitTyp, "hours") +
  labs(x = NULL, y = NULL,
       title = " Frequenz von Tweets des Workshops am 21.9.2023",
       subtitle = paste0(format(min(VersuchOhneNAmitTyp $created_at), "%d %B %Y"), " to ", format(max(VersuchOhneNAmitTyp $created_at),"%d %B %Y")),
       caption = "Daten abgerufen mit Twitters API via rtweet") +
  theme_minimal()

VersuchOhneNAmitTyp %>% dplyr::group_by(Typ) %>%  ts_plot("hours")
```

Nun können wir die Frequenz der Tweets am Jahrestag auch abhängig von Ihrem Typ erkennen.

Zum Schluss sollten wir hinterfragen, wie derartige Datenmassen für die weitere wissenschaftliche Arbeit auch außerhalb von RStudio aufbereitet werden können. Die grundlegenden Datensätze lassen sich leicht exportieren:

```
write_as_csv(DatensatznamedeszugeschnittenenKorpus, "230921Workshop.csv", prepend_ids = TRUE, na = "", fileEncoding = "UTF-8")

> saveRDS(DatensatznamedeszugeschnittenenKorpus, file = "230921Workshop.rds")
```

Die Dateien finden sich dann im jeweiligen Ordner, der von RStudio für den jeweiligen Arbeitsplatz angelegt und in der Regel deckungsgleich benannt wurde.

Das gleiche gilt für eine Datei, die nicht mehr alle Merkmale der ursprünglichen Datensätze beinhaltet, aber für die Weiterverwendung sehr wichtig werden kann. Sie stellt das textliche Destillat dar, welches z.B. in MAXQDA weiterverwendet werden könnte. Damit habe ich aber noch keine Erfahrungen sammeln können.

Alles Weitere klären wir dann im Workshop!

Herzliche Grüße,

Robert Scholz