

# Webscraping für MAXQDA mit R

*Thomas Zapf-Schramm, 26. August 2020*

## 1 Überblick

### 1.1 Texte aus dem Internet

MAXQDA bietet eine Vielzahl von Optionen für den Import von Textdaten in den verschiedensten Formaten. Für den Import von Texten aus dem Internet steht so zum Beispiel die Möglichkeit des Downloads von Webseiten mit Hilfe des MAXQDA Web Collectors als PDF oder Text-Datei zur Verfügung. Twitter- und YouTube-Texte können mit spezialisierten Importfiltern direkt in MAXQDA geladen werden.

Der Import von Webseiten über den Web Collector ist für Projekte mit nicht allzu vielen herunterzuladenden Seiten praktisch und adäquat. Er wird aber – v.a. wenn eine größere Zahl von Web-Pages oder partielle Seiteninhalte importiert werden sollen – schnell etwas mühselig. Jede einzelne Seite muss im Webbrowser manuell angesteuert und gesichert werden.

Darüber hinaus ist der Import von PDF-Dateien nicht für jede gewünschte Analyse vorteilhaft. Beispielsweise kann die automatische Codierfunktion von MAXQDA in PDF-Dokumenten keine Absätze codieren, sondern nur Worte und Sätze. Es gibt in MAXQDA auch keine einfache Möglichkeit, die heruntergeladenen PDF-Dokumente vor dem Import automatisch in kleinere Teile zu splitten, oder Teile automatisch vorzucodieren, wie dies beim Import strukturierter Texte möglich ist.

Möchte man beispielsweise ein beliebiges Diskussionsforum aus dem Internet herunterladen, kann die Datenerfassung für MAXQDA sehr aufwändig werden.

Deshalb möchte ich hier eine Methode für den automatisierten Import größerer Textmengen aus dem Internet am Beispiel des Nutzerforums von MAXQDA vorstellen.

### 1.2 R für Datentransformationen

Das wichtigste Werkzeug ist dabei die Programmiersprache R, die quasi als "Schweizer Taschenmesser" für die Lösung einer Vielzahl datentechnischer Probleme im Rahmen sozialwissenschaftlicher Projekte dienen kann. In R existiert eine große Menge von "Paketen", mit denen Datentransformationen und Formatkonvertierungen relativ einfach werden. Außerdem lassen sich mit R auch eine ganze Reihe interessanter Dinge im Bereich der Textanalyse bewerkstelligen, die sich auf unterschiedliche Weise mit MAXQDA kombinieren lassen.

Mit R ist es möglich, Webseiten seriell herunterzuladen und in datenbankähnlichen Dateien speichern. Es gibt verschiedene spezialisierte Module, die dazu dienen, Informationen aus

Webseiten zu extrahieren und in klar strukturierte Tabellen zu überführen (z.B. Datum, Uhrzeit, Verfasser, Titel, Untertitel, Artikeltext bei Nachrichtenseiten). Diese Tabellen sind mit anderen R-Paketen in verschiedene Formate - z.B. Excel-Tabellen - konvertierbar, die ihrerseits von MAXQDA gelesen werden können.

### **1.3 Der Plan**

Das MAXQDA-Technikforum besteht aus mehreren (z.Z. sieben) Seiten der obersten Ebene (im Folgenden Basisseiten genannt). Von diesen Basisseiten wird auf Threadseiten verlinkt (insgesamt 348), die den Diskussionsfäden (d.h. den Nutzerfragen) entsprechen. Sie enthalten Text aller Postings zu einem Thema (z.Z. 1128). Diese Datenstruktur soll in eine Form gebracht werden, die MAXQDA analysieren kann.

#### **1.3.1 Daten downloaden und speichern**

Unser Ziel ist es, eine Tabelle zu erzeugen, die für jedes Posting im Forum eine Zeile enthält. In den Spalten der Tabelle sollen alle das Posting betreffenden Informationen abgelegt werden. Dazu zählen:

- das Thema zu dem das Posting gehört (auch Thread oder Topic genannt),
- das Datum des Beginns des Topics,
- die URL der Webseite, auf dem sich das Topic befindet,
- die Zahl der Antworten in diesem Topic,
- die Zahl der Views dieses Topics,
- das Datum und die Uhrzeit für jedes Posting,
- den Verfasser des Postings,
- den Typ des Postings (Frage oder Antwort),
- ist das Posting das Erste im Thread?,
- der volle Text des Postings,
- die Version des Betriebssystems auf dem das Posting verfasst wurde, soweit feststellbar,
- die Version von MAXQDA auf die sich das Posting bezieht, soweit feststellbar.

Um zu diesem Ziel zu gelangen müssen bestimmte Schritte ausgeführt werden:

- Die URLs der Basisseiten des Forums müssen ermittelt werden.
- Diese Seiten müssen heruntergeladen werden.
- Aus diesen Seiten müssen inhaltliche Informationen über die vorhandenen Topics extrahiert werden.

- Aus den Seiten müssen die URLs der einzelnen Threadseiten herausgefiltert werden.
- Die Threadseiten müssen heruntergeladen werden.
- Auf diesen Seiten müssen die Detailinformationen zu den Einzelpostings lokalisiert werden.

Um aus den Basisseiten und den Threadseiten die benötigten Informationen zu gewinnen, muss der HTML-Sourcecode der Seiten inspiziert werden, um festzustellen wo auf einer Seite sich die gesuchte Information befindet, und wie man sie ausschneiden kann. Wie das funktioniert, wird an Beispielen gezeigt.

### 1.3.2 Daten an MAXQDA übergeben

Der erstellte “data.frame” wird anschließend in eine Excel-Datei exportiert und danach in MAXQDA eingelesen. Vor dem Export nach Excel müssen in der Exporttabelle noch geeignete Dokumentnamen und Dokumentgruppennamen für MAXQDA erzeugt werden. Außerdem müssen die Datums- und Zeitangaben in ein Format gebracht werden, das MAXQDA korrekt erkennt. Die Dokumentnamen werden aus Datum, Uhrzeit und Verfasser\*innennamen gebildet, die Dokumentgruppennamen werden aus Merkmalen des Threads gebildet (Datum des ersten Postings und Titel).

### 1.3.3 Vorbemerkung zur benutzten Syntax

Dieser Text will keine allgemeine Einführung in die Programmiersprache R oder in ihre Anwendungen im Bereich der Textanalyse geben. Dafür gibt es im Buchhandel und im Internet hervorragende Materialien:<sup>1 2 3 4 5</sup> Stattdessen soll lediglich verdeutlicht werden, dass die Kombination von R mit einer flexiblen Programmiersprache auch für ein prinzipiell GUI-gesteuertes, interaktives Programm wie MAXQDA von erheblichem Vorteil sein kann.

Obwohl für diesen Text keine R-Kenntnisse vorausgesetzt werden sollen, habe ich versucht, die Programmbeispiele so zu formulieren, dass sie auch für nicht Eingeweihte in den Grundzügen nachvollziehbar bleiben.

Wir stützen uns bei den Programmschnipseln in den folgenden Abschnitten stark auf eine Schreibweise für R-Code, bei der viel mit dem Pipeline-Operator “%>%” gearbeitet wird. Ausdrücke mit diesem Operator sehen meistens so aus:

---

<sup>1</sup> Garrett Golemund, Hadley Wickham, R for Data Science, <https://r4ds.had.co.nz>

<sup>2</sup> Cornelius Puschmann, Automatisierte Inhaltsanalyse mit R, <http://inhaltsanalyse-mit-r.de>

<sup>3</sup> Benoit, Kenneth, Kohei Watanabe, Haiyan Wang, Paul Nulty, Adam Obeng, Stefan Müller, and Akitaka Matsuo, 2018, Quanteda: An R package for the quantitative analysis of textual data, Journal of Open Source Software. 3(30), 774

<sup>4</sup> <https://doi.org/10.21105/joss.00774>, <https://tutorials.quanteda.io>

<sup>5</sup> Julia Silge and David Robinson, Text Mining with R — A Tidy Approach, <https://www.tidytextmining.com>

**Zielobjekt** <- **Quellobjekt** %>% **Funktion1**(ParameterA,  
ParameterB) %>% **Funktion2()** %>% **Funktion3**(ParameterX)

Das Ganze ist folgendermaßen zu interpretieren:

- Wir haben ein Quellobjekt (z.B. eine Datentabelle, eine Zeichenkette oder eine Liste von Objekten), aus dem wir ein Zielobjekt berechnen wollen.
- Dieses Quellobjekt wird in eine Funktion "eingefüttert". Die Funktion bearbeitet das Objekt. Sie verändert oder erzeugt Tabellenzellen, modifiziert Zeichenketten usw. Die Parameter der Funktionen — falls welche vorhanden sind — beeinflussen die Art und Weise der Verarbeitung innerhalb der Funktion.
- Das Ergebnis der Bearbeitung wird am Ende von der Funktion zurückgegeben und wird über den Pipeline-Operator in die nächste Funktion eingespeist. Diese Pipeline hintereinander geschalteter Funktionen kann beliebig lang sein.
- Wenn die letzte Funktion in der Kette fertig ist, wird das Ergebnis dem Zielobjekt zugewiesen. Die Zuweisung erfolgt über den Zuweisungsoperator "<-" (oder in bestimmten Situationen über das Gleichheitszeichen "="). "<-" soll ein Pfeil sein, der auf das Objekt zeigt, das das Ergebnis erhalten soll.

#### 1.3.4 Anmerkung zur Vorgehensweise

Am einfachsten führt man R-Programme in der Entwicklungsumgebung R-Studio aus. Dort kann man seine Programme direkt nach dem Schreiben ablaufen lassen. Dazu muss man nicht — wie in vielen anderen Programmiersprachen — ein komplettes Programm haben, man kann auch einzelne Zeilen und Programmstatements ausführen und interaktiv ihr Ergebnis betrachten. Auf diese Weise kann man seine Programme schrittweise schreiben und testen. Nach dieser Methode wollen wir im Folgenden vorgehen. Es werden jeweils ein paar Befehle geschrieben und soweit erklärt, dass ungefähr nachvollziehbar wird, was sie tun, und dann ihr Ergebnis betrachtet. Die Befehlsblöcke werden ausgeführt und ihr Output gezeigt.

## 2 Datendownload und Informationsextraktion

### 2.1 Vorbereitung

Der erste Schritt in so gut wie jedem R-Programm besteht im Laden der notwendigen Funktionen für die Bearbeitung des jeweiligen Problems. Diese Funktionen befinden sich in sogenannten “Packages” oder “Libraries” und können von der Plattform CRAN kostenlos heruntergeladen werden. Es gibt Libraries für jedes denkbare statistische oder methodische Problem (z.Z. ca.16.000).

#### 2.1.1 Laden der notwendigen Befehle<sup>6</sup>

Zunächst benötigen wir die Libraries “tidyverse” und “rvest”. Später kommen die Packages “lubridate” und “openxlsx” hinzu.

“tidyverse” stellt eine Vielzahl von Datenverarbeitungsfunktionen zur Verfügung (Manipulation von Tabellen, Zeichenkettenverarbeitung, Grafiken), “rvest” enthält Funktionen für “webscraping”, für das “Ernten” von Internetinhalten, d.h. v.a. für den Download und die Zerlegung von Webseiten im HTML-Format und “openxlsx” ist eine Schnittstelle zu EXCEL, die wir am Ende für den Datenexport brauchen werden.

```
library(tidyverse)
library(rvest)
library(openxlsx)
```

#### 2.1.2 Benutzte Funktionen

Aus dem “tidyverse”-Paket nutzen wir als erstes die Funktion “str\_split” zum Zerlegen von Zeichenketten, dann die Funktion “tibble” zum Erzeugen von Tabellen, die Funktionen “rowwise”, “mutate” und “unnest” zum Bearbeiten dieser Tabellen. Außerdem stellt dieses Paket den “Pipeline”-Operator zur Verfügung, mit dem das Ergebnis einer Funktion als Input an die nächste Funktion übergeben wird.

Aus dem Paket “rvest” werden die Befehle “read\_html”, “html\_nodes”, “html\_attr” und “html\_text” benutzt. “read\_html” lädt eine HTML-Seite aus dem Internet und “parsed” den Quelltext der Seite, d.h. er wird in eine Datenstruktur umgewandelt, mittels der R direkt auf die einzelnen Elemente des Dokuments zugreifen kann. “html\_nodes” extrahiert einzelne oder mehrere Knoten (HTML-Elemente) aus dieser Baumstruktur. Um die benötigten Elemente zu finden, stehen wahlweise “CSS-Selektoren” oder “X-Path Ausdrücke” zur Verfügung. Hier wird von CSS-Selektoren Gebrauch gemacht, die internetaffinen MAXQDA-Nutzer\*innen, die vielleicht selbst schon eine Website gestaltet haben, wahrscheinlich ein Begriff sind. Mit “html\_attr” können die Attribute der gefundenen

---

<sup>6</sup> Die Programmabschnitte in diesem Text wurden mit der R-Version 4.0.1 geschrieben, das Paket “tidyverse” lag in Version 1.3.0 vor, das darin enthaltene zentrale Paket “dplyr” in Version 1.0.0 (aktuellste Version). Die richtige Version von “dplyr” ist im letzten Programmabschnitt des Textes relevant (Funktion “across”).

HTML\_Elemente ausgelesen werden, im vorliegenden Fall v.a. die "href" Attribute der Anchor-Elemente, d.h. die URLs der Seiten der verschiedenen Ebenen. Die Funktion "html\_text" extrahiert Textinhalte aus HTML-Elementen.

## 2.2 Download der Forums-Basisseiten

### 2.2.1 Ausgangs-URLs ermitteln

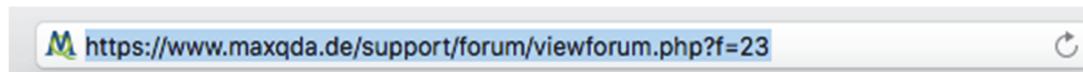
Wir gehen im Webbrowser auf das deutsche MAXQDA-Support Forum in den Zweig "Technische Fragen". Dieses enthält im Kopfbereich der Seite eine mit den Zahlen 1 .. 7 durchnummerierte Liste der Forumsseiten.

Zuerst werden die URLs dieser sieben Seiten der obersten Ebene im Technikforum ermittelt, in dem wir jede Seite zunächst in der Seitenliste am oberen Bildschirmrand anklicken



*Seitenauswahl im Technikforum*

und anschließend die URL aus der Suchzeile des Browsers kopieren (Es ginge auch programmatisch, für die 7 Seiten lohnt hier der Aufwand aber nicht).



*Technikforum in der Browser Suchzeile*

Wir kopieren die Adressen in den R-Editor und fassen sie dort zur Zeichenkette "base\_urls\_str" zusammen. Diese Zeichenkette zerlegen wir direkt anschließend in einen Vektor (eine Tabellenspalte) mit sieben Elementen, den wir sofort durch Eingabe seines Namens anzeigen. Schließlich definieren wir noch eine "root\_url" mit dem Pfad zur Oberebene der Foren. Wir werden sie später benötigen, um unvollständige URLs zu ergänzen.

#### **Im Folgenden gilt durchgängig:**

- Die grau hinterlegten Texte ohne "##" am Zeilenanfang sind unser Programm.
- Die grau hinterlegten Zeilen mit "##" am Anfang sind die Antworten des R-Systems auf diese Befehle.

```
base_url_str =  
  "https://www.maxqda.de/support/forum/viewforum.php?f=23  
  https://www.maxqda.de/support/forum/viewforum.php?f=23&start=50  
  https://www.maxqda.de/support/forum/viewforum.php?f=23&start=100  
  https://www.maxqda.de/support/forum/viewforum.php?f=23&start=150  
  https://www.maxqda.de/support/forum/viewforum.php?f=23&start=200
```

```
https://www.maxqda.de/support/forum/viewforum.php?f=23&start=250
https://www.maxqda.de/support/forum/viewforum.php?f=23&start=300"
```

```
base_urls <- str_split(base_url_str, "\\s+") %>% unlist()
base_urls
```

```
## [1] "https://www.maxqda.de/support/forum/viewforum.php?f=23"
## [2] "https://www.maxqda.de/support/forum/viewforum.php?f=23&start=50"
## [3] "https://www.maxqda.de/support/forum/viewforum.php?f=23&start=100"
## [4] "https://www.maxqda.de/support/forum/viewforum.php?f=23&start=150"
## [5] "https://www.maxqda.de/support/forum/viewforum.php?f=23&start=200"
## [6] "https://www.maxqda.de/support/forum/viewforum.php?f=23&start=250"
## [7] "https://www.maxqda.de/support/forum/viewforum.php?f=23&start=300"
root_url <- "https://www.maxqda.de/support/forum"
```

## 2.2.2 Basisseiten laden

Mit der Funktion “tibble” erzeugen wir eine Tabelle namen “base\_pages” und fügen in sie die Spalte mit den sieben URLs ein. Anschließend berechnen wir mit “mutate” eine zweite Spalte namens “doc”, in die wir mit “read\_html” den geparsten Quelltext der sieben Basisseiten einlesen. Seiten-Download, Parsing und Sichern in der Tabelle finden in einem einzigen Befehl statt. Durch die Eingabe ihres Namens werden die base\_pages am Bildschirm angezeigt.

```
base_pages <-
  tibble(baseurl=base_urls) %>%
  rowwise() %>%
  mutate(doc=list(read_html(baseurl)))
```

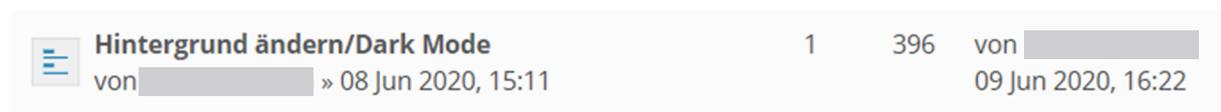
```
base_pages
```

```
## # A tibble: 7 x 2
## # Rowwise:
##   baseurl                                doc
##   <chr>                                  <list>
## 1 https://www.maxqda.de/support/forum/viewforum.php?f=23    <xml_dcmn>
## 2 https://www.maxqda.de/support/forum/viewforum.php?f=23&start=50 <xml_dcmn>
## 3 https://www.maxqda.de/support/forum/viewforum.php?f=23&start=100 <xml_dcmn>
## 4 https://www.maxqda.de/support/forum/viewforum.php?f=23&start=150 <xml_dcmn>
## 5 https://www.maxqda.de/support/forum/viewforum.php?f=23&start=200 <xml_dcmn>
## 6 https://www.maxqda.de/support/forum/viewforum.php?f=23&start=250 <xml_dcmn>
## 7 https://www.maxqda.de/support/forum/viewforum.php?f=23&start=300 <xml_dcmn>
```

## 2.3 Extraktion themenspezifischer Informationen von den Basisseiten und Download der Threadseiten

Auf jeder der sieben gespeicherten Seiten befinden sich die Links zu jeweils 50 Diskussionsthemen (threads oder topics). Jede neue Nutzerfrage startet einen Thread, die Antworten der MAXQDA-Expert\*innen, der anderen Nutzer\*innen und Folgebeiträge der Fragesteller\*innen bilden den Rest des Diskussionsfadens. Jeder Thread befindet sich auf einer eigenen Seite. Im nächsten Schritt ermitteln wir die URLs dieser Thread-Seiten.

Ein Link zu einer Threadseite sieht im Browser so aus:



### Thread-Link auf der Basisseite

Wenn wir mit der rechten Maustaste auf einen solchen Link klicken, dann können wir die “Elementinformationen” (im Safari-Browser) zu diesem Teil der HTML-Seite öffnen, beziehungsweise das Element “untersuchen” (Chrome / Firefox) (sofern wir die Entwicklereinstellungen des Browsers aktiviert haben).

```
<li class="row bg1">
  <dl class="icon" style="background-image: url(./styles/hawiki/imageset/topic_read.svg); background-repeat: no-repeat;">
    <dt title="Keine ungelesenen Beiträge">
      <a href="/viewtopic.php?f=23&t=2022" class="topicitle">Hintergrund ändern/Dark Mode</a> == $0
    <br>
    "
      von "
      <a href="/memberlist.php?mode=viewprofile&u=3508">[redacted]</a>
      " » 08 Jun 2020, 15:11
    "
  </dt>
  <dd class="posts">
    "1 "
    <dfn>Antworten</dfn>
  </dd>
  <dd class="views">
    "396 "
    <dfn>Zugriffe</dfn>
  </dd>
  <dd class="lastpost">
    <span>...</span>
  </dd>
  ::after
</dl>
</li>
```

### Thread Links in der Entwickleransicht

Von den hier angezeigten Informationen interessiert uns zunächst die Seiten-URL des Threads im **<a>** Element im Attribut **“href”**. Im Beispiel lautet sie:

**“./viewtopic.php?f=23&t=2022”**. Dort finden sich alle Postings zu diesem Thread.

Außerdem finden sich im gewählten Ausschnitt Angaben zum Titel des Threads – **dt a class=“topicitle”** –, über die Zahl der Antworten – **dd class=“posts”** – zu diesem

Thema und über die Abrufzahl – **dd class=“views”**. Alle diese Informationen wollen wir

sichern. Den Namen der ersten Poster\*in und das Startdatum des Threads ignorieren wir hier. Wir gewinnen sie später auf andere Weise.

### 2.3.1 Listenspalten

Für diesen Schritt spielen "Listenspalten" in der Tabelle eine Rolle. Normalerweise lässt sich in jeder Zelle einer R-Datentabelle ein einzelner Wert speichern. Bei Tabellen von Typ "tibble" ist in einer Zelle auch eine Werteliste oder eine ganze Subtabelle speicherbar. Von Wertelisten in Tabellenfeldern machen wir umfänglich Gebrauch.

Wir extrahieren aus den Dokumenten Listen mit den Thread-URLs ("html\_nodes" und "html\_attr"), den Threadtiteln, den Poster\*innen und Thread-Startdaten, den Antwort-Anzahlen und Zugriffszahlen (mit "html\_nodes" und "html\_text"), wobei wir die gesuchten Informationen mittels CSS-Selektoren (z.B. "**ul.topiclist.topics li.row dt a.topictitle**") finden. Wir speichern sie in entsprechend benannten Listenspalten. Dann lassen wir die "thread\_pages"-Tabelle anzeigen.

```
thread_pages <- base_pages %>%
  rowwise() %>%
  mutate(
    thread_url = list(doc %>% html_nodes("ul.topiclist.topics li.row dt a.topictitle") %>% html_attr("href")),
    thread_title = list(doc %>% html_nodes("ul.topiclist.topics li.row dt a.topictitle") %>% html_text()),
    antworten = list(doc %>% html_nodes("ul.topiclist.topics li.row dd.posts") %>% html_text()),
    views = list(doc %>% html_nodes("ul.topiclist.topics li.row dd.views") %>% html_text())
  ) %>%
  select(-doc) %>%
  unnest(c(thread_url:views)) %>%
  rowwise() %>%
  mutate(
    thread_url=paste0(root_url,str_remove(thread_url, "\\."),
    doc=list(read_html(thread_url)),
  )
thread_pages
```

```
## # A tibble: 350 x 6
## # Rowwise:
##   baseurl      thread_url      thread_title      antworten views doc
##   <chr>        <chr>          <chr>            <chr>    <chr> <lis>
## 1 https://www.m... https://www.maxqda... Erw. lexikalische ... 1 Antwor... 102 Z... <xml...
## 2 https://www.m... https://www.maxqda... Spalten von Excel-... 1 Antwor... 1301 ... <xml...
## 3 https://www.m... https://www.maxqda... Verknüpfung Video ... 1 Antwor... 73 Zu... <xml...
## 4 https://www.m... https://www.maxqda... Projekt wird als [... 2 Antwor... 321 Z... <xml...
## 5 https://www.m... https://www.maxqda... Text-Dokument DOCX... 5 Antwor... 2959 ... <xml...
## 6 https://www.m... https://www.maxqda... Word Formatierung ... 2 Antwor... 388 Z... <xml...
## 7 https://www.m... https://www.maxqda... Media-Browser spie... 13 Antwo... 18101... <xml...
## 8 https://www.m... https://www.maxqda... Fehlermeldungen be... 1 Antwor... 376 Z... <xml...
## 9 https://www.m... https://www.maxqda... Codes verrutschen   1 Antwor... 345 Z... <xml...
## 10 https://www.m... https://www.maxqda... Codeliste wird nic... 3 Antwor... 1430 ... <xml...
## # ... with 340 more rows
```

### **2.3.2 Unnest**

Mit “unnest” können solche verschachtelten Tabellen in normale “flache” Tabelle umgewandelt werden, die für jedes Element einer eingebetteten Liste oder für jede Zeile einer eingebetteten Tabelle eine eigene Zeile hat. In diesen Zeilen werden Werte der anderen Spalten der übergeordneten Tabelle wiederholt.

Wir expandieren also mit “unnest” die siebenzeilige Tabelle in eine neue Tabelle mit 348 Zeilen für die 348 gefundenen Themen.

Sobald dieser Vorgang erfolgreich abgeschlossen ist, sind die geparsten HTML-Texte in der doc-Spaltenun nicht mehr erforderlich und werden aus der Tabelle entfernt “(select - doc)”.

### **2.3.3 Rowwise**

Diese Funktion bewirkt, dass eine Tabelle zeilenweise bearbeitet wird, und nicht – wie normalerweise mit “tidyverse”– jeweils eine Spalte auf einen Rutsch. Die übliche sehr effiziente Methode ist mit den HTML-Extraktionsfunktionen nicht möglich.

In jeder der 348 Zeilen der Threadtabelle (die bis zu diesem Punkt ausschließlich aus den Einträgen der sieben Top-Seiten generiert wurde) wird nun die URL der Threadseite (thread\_url) genutzt, um die separate HTML-Seite des entsprechenden Themendiskussionsstrangs zu laden. Da die extrahierten Thread-URLs nur den hinteren Teil der URL enthalten muss vorher die “root\_url”, die wir am Anfang definiert haben, mit “paste” davor geklebt werden. Der Download der Threadseiten ist der langwierigste Schritt in der der ganzen vorgestellten Prozedur. Er kann mehrere Minuten in Anspruch nehmen, während alle vorherigen und späteren Schritte nur Sekunden brauchen.

Alle Postings im Forum befinden sich auf diesen 348 Seiten, es gibt keine weitere Subebene. Daher müssen die 348 Themenstränge weiter zerlegt werden, so dass am Ende aus der 348-Zeilentabelle eine neue Tabelle entsteht, die so viele Zeilen hat, wie Postings im Forum existieren.

## **2.4 Extraktion der Posting-spezifischen Informationen und Posting-Texte von den Threadseiten**

Ein einzelnes Posting auf einer Thread-Seite sieht so aus:

### Hintergrund ändern/Dark Mode

08 Jun 2020, 15:11

Beiträge: 1  
Registriert: 08 Jun 2020,  
14:32

Guten Tag,

lässt sich in MAXQDA ein Dark Mode aktivieren oder irgendwie die Hintergrundfarbe ändern, um den Kontrast zu reduzieren? Ich habe bereits einen Filter für meinen Bildschirm insgesamt eingerichtet, aber das Design von MAXQDA ist so hell und so kontrastreich, dass ich dennoch immer nach relativ kurzer Zeit Probleme mit den Augen bekomme.

Falls nicht, wäre das als zukünftige Verbesserung sehr wünschenswert. Unter den entsprechenden Stichwörtern habe ich leider bisher nirgends etwas gefunden.

Herzliche Grüße,  
[Redacted]

Version: MAXQDA 2020  
System: Windows 10

### Posting im Browser

In der Elementansicht des Entwicklermodus des Webbrowsers wird wieder die Struktur der Seite und die genaue Verortung der relevanten Informationen auf der Seite sichtbar.

```

---
<div id="p5288" class="post bg2">
  <div class="postbody">
    <ul class="profile-icons">...</ul>
    <h2 class="first">
      <a href="#p5288">Hintergrund ändern/Dark Mode</a>
    </h2>
    <p class="author">
      <a href="./viewtopic.php?p=5288#p5288"> = $0
      <i class="fa fa-file">...</i>
    </a>
      "08 Jun 2020, 15:11"
    </p>
    <div class="content">
      "Guten Tag,"
      <br>
      <br>
      "lässt sich in MAXQDA ein Dark Mode aktivieren oder irgendwie die Hintergrundfarbe
      ändern, um den Kontrast zu reduzieren? Ich habe bereits einen Filter für meinen
      Bildschirm insgesamt eingerichtet, aber das Design von MAXQDA ist so hell und so
      kontrastreich, dass ich dennoch immer nach relativ kurzer Zeit Probleme mit den Augen
      bekomme. "
      <br>
      <br>
      "Falls nicht, wäre das als zukünftige Verbesserung sehr wünschenswert. Unter den
      entsprechenden Stichwörtern habe ich leider bisher nirgends etwas gefunden. "
      <br>
      <br>
      "Herzliche Grüße,"
      <br>
      [Redacted]
      <br>
      <br>
      "Version: MAXQDA 2020"
      <br>
      "System: Windows 10"
      <br>
    </div>
  </div>
  <dl class="postprofile" id="profile5288">...</dl>
  <div class="back2top">...</div>
  ::after
</div>
<hr class="divider">

```

### Posting in der Codeansicht

Auch hier finden wir wieder alle notwendigen Informationen, um den Text und die Deskriptoren eines Postings aus der Seite heraus zu schneiden und in unsere Tabelle einzutragen.

### 2.4.1 Extraktion des Posting-Textes

Bisher haben wir die Textinhalte der HTML-Elemente mit der Funktion “html\_text” ausgelesen. Das funktioniert allerdings bei nur bei kurzen Texten gut. “html\_text” klebt nämlich bei längeren Texten, die aus mehreren HTML-Subelementen bestehen, einfach alle Teilstücke ohne Lücke zusammen. Dabei wird gelegentlich das letzte Wort eines Teilstücks dem ersten Wort des nächsten Teilstücks fusioniert.

Deshalb entwickeln wir hier ohne weitere Erläuterung eine kurze Funktion “clean\_html\_text”, die wir zur Extraktion der Posting-Texte einsetzen. Sie setzt Zeilenumbrüche zwischen Textteilen ein und löscht überflüssige HTML-Tags.

```
clean_html_text <- function(nodes) {
  lapply(nodes,
    function(node)
    { node %>%
      xml_contents() %>%
      paste(collapse="\n") %>%
      str_remove_all("<.*?>") %>%
      str_replace_all("\n+", "\n\n")
    }) %>%
  unlist()
}
```

### 2.4.2 Deskriptoren der Postings

Die Informationen zu “titel”, “datum”, “autor” und “content” finden wir – wie schon die Thread-Eigenschaften – wieder einfach mit CSS-Selektoren (z.B. “div.postbody div.content”). Bei der Information “first” weicht das Muster etwas ab.<sup>7</sup>

```
postings <- thread_pages %>%
  rowwise() %>%
  mutate(
    titel = list(doc %>% html_nodes("div.postbody h2") %>% html_text()),
    datum = list(doc %>% html_nodes("div.postbody p.author") %>% html_text()),
    autor = list(doc %>% html_nodes("dl.postprofile dt") %>% html_text()),
    first = list(!is.na(doc %>% html_nodes("div.postbody h2") %>%
html_attr("class"))),
```

---

<sup>7</sup> Wenn wir mit “html\_nodes” und “html\_attr” nach den gewünschten Daten suchen, erhalten wir im Ergebnis eine Tabellenspalte, bei der jedes erste Posting eines Threads den Eintrag “first” hat, alle anderen den Eintrag “NA” (R-Symbol für fehlenden Wert). Wir hätten aber gerne eine Spalte, die für die Erst-Postings “TRUE” und für alle anderen “FALSE” enthält. Der Ausdruck “!is.na()” berechnet genau das (“first” fehlt nicht).

```

    content = list(doc %>% html_nodes("div.postbody div.content") %>%
clean_html_text())
) %>%
select(-doc) %>%
unnest(c(titel:content)) %>%
mutate(
  content_size =str_length(content)
)

```

postings

```

## # A tibble: 1,126 x 11
##   baseurl thread_url thread_title antworten views titel datum autor first
##   <chr>    <chr>         <chr>         <chr>    <chr> <chr> <chr> <chr> <lg1>
## 1 https:... https://w... Erw. lexika... 1 Antwort... 102 ... Erw.... 21 A... "\n\... TRUE
## 2 https:... https://w... Erw. lexika... 1 Antwort... 102 ... Re: ... 25 A... "\n\... FALSE
## 3 https:... https://w... Spalten von... 1 Antwort... 1301... Spal... 29 J... "\n\... TRUE
## 4 https:... https://w... Spalten von... 1 Antwort... 1301... Re: ... 30 J... "\n\... FALSE
## 5 https:... https://w... Verknüpfung... 1 Antwort... 73 Z... Verk... 25 A... "\n\... TRUE
## 6 https:... https://w... Verknüpfung... 1 Antwort... 73 Z... Re: ... 25 A... "\n\... FALSE
## 7 https:... https://w... Projekt wir... 2 Antwort... 321 ... Proj... 10 A... "\n\... TRUE
## 8 https:... https://w... Projekt wir... 2 Antwort... 321 ... Re: ... 12 A... "\n\... FALSE
## 9 https:... https://w... Projekt wir... 2 Antwort... 321 ... Re: ... 24 A... "\n\... FALSE
## 10 https:... https://w... Text-Dokume... 5 Antwort... 2959... Text... 29 N... "\n\... TRUE
## # ... with 1,116 more rows, and 2 more variables: content <chr>,
## #   content_size <int>

```

### 3 Zielprojekte

Als Beispiel für des mögliche Ergebnis eines solchen Downloads sollen zwei sehr unterschiedlich aufgebaute MAXQA-Projekte auf der Basis der Postings im MAQDA-Supportforum aufgesetzt werden.

Das erste Projekt besteht aus längeren vorcodierten Textdokumenten, die den Diskussionsfäden (threads) des Forums entsprechen. Diese Dokumente werden mit einer Reihe von Codes mit Subcodes vorcodiert. Die Codierungen beziehen sich auf Absätze der Texte, die den einzelnen Postings innerhalb des Themenfadens entsprechen. Jedes Posting erhält mehrere überlappende Codierungen. Es ist also ganz im Sinne von Projekten im Stil primär qualitativer Sozialforschung aufgebaut, wo i.d.R. von längere Texte Forschungsgegenstand sind, die dann schrittweise codiert werden, indem Absätzen, Aussagen, Sätzen oder anderen "Coding Units" inhaltliche Codes zugewiesen werden, die dann später zusammengefasst, gesplittet, recodiert und schließlich ausgewertet werden können.

Das andere Projekt enthält als primäre Dokumente die einzelnen Postings innerhalb der Threads, die übergeordneten Threads werden als Dokumentgruppen abgebildet. Statt vieler Codes mit Subkategorien haben die Postings in diesem Projekt eine längere Liste von Dokumentvariablen, die ihre Eigenschaften und die Eigenschaften der Diskussionsfäden, in die sie eingebettet sind, repräsentieren. Das Projekt entspricht also eher der üblichen Vorgehensweise bei quantitativen Inhaltsanalysen oder Mixed-Method-Ansätzen, es ist prinzipiell aufgebaut wie eine Umfrage mit einer offenen Frage (dem Posting-Text) und einer Reihe von Kontextvariablen. Es kann daher auch am besten ausgewertet werden, wenn in MAXQDA auch das Stats-Modul und MAXDictio installiert sind.

Die Datengewinnung aus dem Internet ist für beide Projekte exakt identisch, lediglich die Aufbereitung der Daten unterscheidet sich.

Es geht in diesem Text hauptsächlich darum, zu zeigen, wie prinzipiell beliebige Textdaten aus dem Internet in eine Form gebracht werden können, die für eine effiziente Bearbeitung mit MAXQDA geeignet ist. Der eigentliche Zweck der Bearbeitung mit MAXQDA - nämlich die inhaltliche Codierung der Textdokumente - ist hier noch nicht geleistet. Die Codierung in der ersten Projektvariante bezieht sich ebenso wie die Variablen der zweiten Variante nur auf die formalen Eigenschaften der Texte. Trotzdem lassen sich auf dieser Basis schon einige interessante Aussagen über die zeitliche Verteilung der Postings und über die Beteiligung der Poster machen.

Da allerdings der eigentliche Zweck von MAXQDA noch nicht genutzt wird, könnten die jetzt schon möglichen Ergebnisse – zumindest in Projektvariante 2 – mit jedem beliebigen Statistikprogramm oder mit EXCEL erzielt werden. Durch die Aufbereitung wird aber der Rahmen bereitgestellt, innerhalb dessen die eigentliche Arbeit der qualitativen Forscher\*innen ansetzen kann.

Bevor die Datengewinnung und Aufbereitung mit R genauer gezeigt werden, sehen wir uns die Zielprojekte kurz näher an.

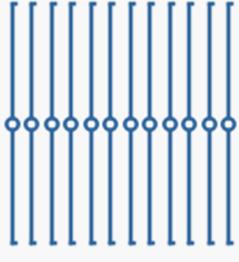
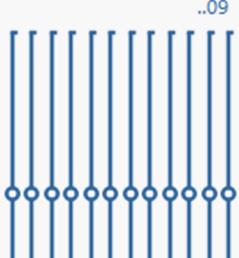
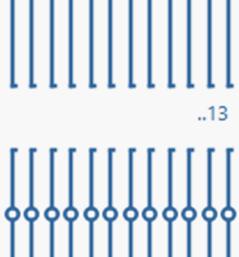
### 3.1 Projekt 1

Projekt 1 besteht aus 348 Dokumenten, 1 Dokument pro Forum-Thread. Die Dokumentnamen werden gebildet aus Datum und Uhrzeit des ersten Postings im Thread und dem Thema des Threads. Die resultierende Zeichenkette muss auf 63 Zeichen beschnitten werden, denn längere Dokumentnamen sind in MAXQDA nicht erlaubt.

•  2020-03-13 - Zwei Gewichtungsparmet...	24
•  2020-03-16 - Subcode zu verschiedenen...	36
•  2020-03-17 - Verschieben von Dokumen...	48
•  2020-03-30 - Dokumentenvariablen änd...	48
•  2020-03-30 - Statistik für Sprechervariab...	72
•  2020-04-03 - Dokument exportieren	24
•  2020-04-05 - Codes und Subcodes über...	60
•  2020-04-26 - Dringend: Aus MaxQDA ex...	24
•  2020-05-04 - Intercoder Übereinstimmu...	24
•  2020-05-07 - Zwei Versionen eines Proje...	24
•  2020-05-11 - Audiowiedergabe stoppt b...	24
•  2020-05-11 - Codebuch mit Farben expo...	36
•  2020-05-15 - Dokumente innerhalb eines...	36
•  2020-05-19 - Qualitative Inhaltsanalyse: F...	24
•  2020-05-29 - Audios spielen nicht ab	24
•  2020-05-29 - Automatisches Codieren	24
•  2020-06-03 - Multimedia-Browser öffnet...	24
•  2020-06-08 - Hintergrund ändern/Dark ...	24
•  2020-06-29 - Spalten von Excel-tabelle w...	24
•  2020-07-01 - Zeitmarken funktionieren ni...	48
•  2020-07-07 - Variablen für einzelne Seg...	24

*Projekt 1: Ausschnitt aus der Dokumentliste*

Jedes Posting ist innerhalb eines Thread-Textes durch mindestens drei Absätze repräsentiert. Der erste Absatz enthält Datum und Uhrzeit des Postings, der zweite Absatz den Namen des Verfassers, und die restlichen Absätze den eigentlichen Text des Postings.

	1	<p><b>2011-12-09 11:01:00</b></p> <p>Hallo,ich habe folgendes Problem:- im Textretrieval (Fenster rechts unten) sind Codings- diese möchte ich feiner kodieren, rufe sie daher auf (Fenster oben rechts) und verschiebe sie per drag &amp; drop in den Codebaum (links unten)- wenn das passiert ist, springt das Textretrieval-Fenster (unten rechts, Liste der Codings) immer wieder zum ersten Coding, so dass ich immer wieder herunterscrollen muss, um die Stelle zu finden, wo ich weitermachen möchte.Das ist sehr lästig.Kann man das irgendwie verhindern?LG und danke im Voraus,Christine</p>
	..09	<p><b>2011-12-13 10:42:00</b></p> <p>Hallo Christine,oh ja, das ist tatsächlich etwas unpraktisch. Wir haben es gleich mal auf die Wunschliste geschrieben - wird allerdings bis Weihnachten nichts mehr Was allerdings für diesen Zweck gut funktioniert, ist die Übersicht Codings. Diese zeigt das Retrieval in Tabellenform an. Aufzurufen ist die Übersicht, indem man auf den gleichnamigen Button in der "Liste der Codings" klickt.Mit dem allerletzten Update für MAXQDA 10 (R111111) ist bereits eine Funktion hinzugekommen, die bei diesem Problem sehr hilfreich sein könnte: Wenn man die Alt-Taste gedrückt hält und einen Codierstreifen (aus dem Fenster rechts oben) oder eine Herkunftsangabe (aus dem Fenster rechts unten) oder eine Zeile aus der Übersicht Codings auf einen Code zieht, dann VERSCHIEBT MAXQDA die Codierung zu diesem Code.Viele GrüßeStefan</p>
	..13	<p><b>2014-05-02 17:32:00</b></p> <p>Hallo!Der Thread ist ja bereits etwas älter - ich habe aktuell aber genau dieses Problem. Daher meine Frage: Gibt es hierzu bereits einen Lösungsvorschlag?Danke und viele Grüße,Sabine</p>
	..Antwort	<p><b>2014-05-05 16:19:00</b></p> <p>Hallo Sabine,das ist nach wie vor ein verständlicher Wunsch, aber in der jetzigen 11-Version für Windows erstmal nicht zu realisieren.Wir haben das aber weiter auf der Agenda!Viele Grüße</p>

### Projekt 1: Postings im Thread

Alle zu einem Posting gehörenden Absätze erhalten die gleiche Codierung, und zwar auf mehreren Codedimensionen (Autor, Jahr, Monat, Tag, Wochentag, Jahr\_Monat, Posting-Typ, Erster, OS\_Version, MAXQDA\_Version).

Jede dieser Codedimensionen hat eine Reihe von Subcodes, die in der Liste der Codes als aufklappbare Liste erscheinen (Autorennamen, Jahre 2011 .. 2020, Monatsnamen, Jahres/Monatskombinationen für Zeitreihenbetrachtung, Frage/Antwort, Ja/Nein, Betriebssystemversionen von Windows und macOS (OS X), MAXQDA-Versionen).

▷	☉	jahr	1129
▲	☉	monat	0
	☉	01 Jan	114
	☉	02 Feb	107
	☉	03 Mrz	139
	☉	04 Apr	83
	☉	05 Mai	84
	☉	06 Jun	94
	☉	07 Jul	73
	☉	08 Aug	82
	☉	09 Sep	59
	☉	10 Okt	87
	☉	11 Nov	115
	☉	12 Dez	92
▷	☉	jahr_monat	1129
▷	☉	tag	1129
▲	☉	wtag	0
	☉	05 Fr	120
	☉	02 Di	267
	☉	01 Mo	219
	☉	04 Do	196

*Projekt 1: Ausschnitt aus Codeliste mit einigen Detailcodes*

Die Arbeit der qualitativ Forschenden läge nach dem Aufbau dieses Projektrahmens darin, weitere inhaltlich relevante Codedimensionen zu finden, sie mit Subcodes auszudifferenzieren und die Coding-Absätze (oder Teile von ihnen) damit zu codieren.

## 3.2 Projekt 2

Projekt 2 besteht aus 1128 Dokumenten, entsprechend der Zahl der Postings im Forum. Sie sind in 348 Dokumentgruppen, entsprechend den Forum-Threads, zusammengefasst. Die Namen der Dokumentgruppen werden genauso gebildet, wie die Dokumentnamen in Projekt 1. Die Dokumentnamen setzen sich aus Datum und Uhrzeit des Postings und dem Posternamen zusammen. Mit der neuen MAXQDA-Version 2020.1 müsste auch eine tiefer verschachtelte Version (z.B. mit einer Ebene Jahr\_Monat oberhalb der Threads) möglich sein, es ist aber noch nicht im Manual dokumentiert, ob und wie das funktioniert.

Dokumente		3387
2011-12-09 - an Anfang springen beim Textr...		15
2011-12-09 11:01:00_...		3
2011-12-13 10:42:00_S...		3
2014-05-02 17:32:00_S...		3
2014-05-05 16:19:00_S...		3
2017-11-07 14:51:00_...		3
2012-08-06 - Codes, Codings und Memos in...		45
2012-08-06 14:27:00_...		3
2012-08-09 08:03:00_S...		3
2012-09-03 10:43:00_...		3
2012-09-03 13:16:00_S...		3
2012-10-12 16:16:00_...		3
2012-10-15 19:38:00_S...		3
2013-04-18 15:00:00_...		3
2013-11-01 17:41:00_...		3
2013-11-01 21:08:00_S...		3
2014-02-19 10:54:00_...		3
2014-02-19 13:04:00_S...		3
2014-03-04 15:25:00_...	nn	3
2014-03-04 15:51:00_...		3

*Projekt 2: Ausschnitt aus der Dokumentliste mit Threads als Dokumentgruppen*

Jeder Posting-Text enthält für die leichtere Lesbarkeit Datum/Zeit, Titel und Autor als Überschrift, dann folgt der eigentliche Text.

titel	1	an Anfang springen beim Textretrieval-Codieren verhindern?
autor	2	.....
content	3	Hallo,ich habe folgendes Problem:- im Textretrieval (Fenster rechts unten) sind Codings- diese möchte ich feiner kodieren, rufe sie daher auf (Fenster oben rechts) und verschiebe sie per drag & drop in den Codebaum (links unten)- wenn das passiert ist, springt das Textretrieval-Fenster (unten rechts, Liste der Codings) immer wieder zum ersten Coding, so dass ich immer wieder herunterscrollen muss, um die Stelle zu finden, wo ich weitermachen möchte.Das ist sehr lästig.Kann man das irgendwie verhindern?LG und danke im Voraus,Christine

### Projekt 2: Text

In der Code-Liste stehen nur drei Codes ohne Subcodes, die den genannten Teilen des Textes entsprechen (Datum, Autor, Content). Die verschachtelten Codelisten aus Projekt 1 entfallen.

Codesystem	3387
titel	1129
autor	1129
content	1129
Sets	0

### Projekt 2: Codeliste

Dafür gibt es in dieser Projektvariante jetzt eine Tabelle mit Dokumentvariablen. Da jetzt jedes Posting ein Dokument ist, können ihm alle Informationen, die im ersten Projekt über Codes den Absätzen des Textes zugeordnet wurden, in die Tabelle der Codevariablen ausgelagert werden. Die Codierung über die Code-Liste wird viel übersichtlicher, wenn die formalen Kontext-Informationen hier ausgelagert werden.

Liste der Dokumentvariablen 29 Variablen

Variablenname	Variablentyp	Sichtbar	Quelle	Fehlender Wert	Kategorial	Favoriten-Varia...
■ Dokumentgruppe	Text	<input checked="" type="checkbox"/>	System		<input checked="" type="checkbox"/>	<input type="checkbox"/>
■ Dokumentname	Text	<input checked="" type="checkbox"/>	System		<input checked="" type="checkbox"/>	<input type="checkbox"/>
■ Erstellt von	Text	<input type="checkbox"/>	System		<input type="checkbox"/>	<input type="checkbox"/>
■ Erstellt am	Datum/Uhrzeit	<input type="checkbox"/>	System		<input type="checkbox"/>	<input type="checkbox"/>
■ Bearbeitet von	Text	<input type="checkbox"/>	System		<input type="checkbox"/>	<input type="checkbox"/>
■ Bearbeitet am	Datum/Uhrzeit	<input type="checkbox"/>	System		<input type="checkbox"/>	<input type="checkbox"/>
■ Codierte Segmente	Ganzzahl	<input checked="" type="checkbox"/>	System		<input type="checkbox"/>	<input type="checkbox"/>
■ Memos	Ganzzahl	<input checked="" type="checkbox"/>	System		<input type="checkbox"/>	<input type="checkbox"/>
■ topic_url	Text	<input checked="" type="checkbox"/>	Benutzer		<input type="checkbox"/>	<input type="checkbox"/>
■ topic_title	Text	<input checked="" type="checkbox"/>	Benutzer		<input type="checkbox"/>	<input type="checkbox"/>
■ topic_datum	Datum/Uhrzeit	<input checked="" type="checkbox"/>	Benutzer		<input type="checkbox"/>	<input type="checkbox"/>
■ antworten	Ganzzahl	<input checked="" type="checkbox"/>	Benutzer		<input type="checkbox"/>	<input type="checkbox"/>
■ views	Ganzzahl	<input checked="" type="checkbox"/>	Benutzer		<input type="checkbox"/>	<input type="checkbox"/>
■ datum	Datum/Uhrzeit	<input checked="" type="checkbox"/>	Benutzer		<input type="checkbox"/>	<input type="checkbox"/>
■ uhrzeit	Text	<input checked="" type="checkbox"/>	Benutzer		<input type="checkbox"/>	<input type="checkbox"/>
■ timestamp	Datum/Uhrzeit	<input checked="" type="checkbox"/>	Benutzer		<input type="checkbox"/>	<input type="checkbox"/>
■ jahr	Ganzzahl	<input checked="" type="checkbox"/>	Benutzer		<input checked="" type="checkbox"/>	<input type="checkbox"/>
■ monat	Text	<input checked="" type="checkbox"/>	Benutzer		<input checked="" type="checkbox"/>	<input type="checkbox"/>
■ jahr_monat	Text	<input checked="" type="checkbox"/>	Benutzer		<input type="checkbox"/>	<input type="checkbox"/>
■ tag	Ganzzahl	<input checked="" type="checkbox"/>	Benutzer		<input checked="" type="checkbox"/>	<input type="checkbox"/>
■ wtag	Text	<input checked="" type="checkbox"/>	Benutzer		<input checked="" type="checkbox"/>	<input type="checkbox"/>
■ stunde	Ganzzahl	<input checked="" type="checkbox"/>	Benutzer		<input checked="" type="checkbox"/>	<input type="checkbox"/>
■ size	Ganzzahl	<input checked="" type="checkbox"/>	Benutzer		<input type="checkbox"/>	<input type="checkbox"/>
■ os_ver	Text	<input checked="" type="checkbox"/>	Benutzer		<input checked="" type="checkbox"/>	<input type="checkbox"/>
■ maxqda_ver	Text	<input checked="" type="checkbox"/>	Benutzer		<input checked="" type="checkbox"/>	<input type="checkbox"/>
■ titel	Text	<input checked="" type="checkbox"/>	Benutzer		<input type="checkbox"/>	<input type="checkbox"/>
■ autor	Text	<input checked="" type="checkbox"/>	Benutzer		<input checked="" type="checkbox"/>	<input type="checkbox"/>
■ first	Boolean (wahr/fal...	<input checked="" type="checkbox"/>	Benutzer		<input checked="" type="checkbox"/>	<input type="checkbox"/>
■ typ	Text	<input checked="" type="checkbox"/>	Benutzer		<input checked="" type="checkbox"/>	<input type="checkbox"/>

*Projekt 2: Variablenliste für die Postings*

## 4 MAXQDA-Import

### 4.1 Vorbereitung für beide Projekte

Für die Ausgabe nach Excel/MAXQDA erzeugen wir eine neue Tabelle "forum\_postings", in die die Spalten der Tabelle "postings" in modifizierten Versionen eingefügt werden.

Zunächst bringen wir die Datums- und Zeitspalten in eine Form, die Excel versteht (ins ISO-Standardformat). Bevor wir das tun können, müssen wir, falls wir auf Windows arbeiten, eine Korrektur der Datums-Spalte vornehmen, denn die deutsche Windowsversion nutzt ein anderes Monatskürzel für den März als die Forumswebseite. Bei macOS gibt es kein Problem.

```
if (Sys.info()["sysname"]=="Windows") {  
  postings$datum <- postings$datum %>% str_replace("Mär", "Mrz")  
}
```

Danach nutzen wir die Funktionen "as\_datetime()" und "strftime" ("String from Time"). Unsere bisherigen Zeitspalten sind in Wirklichkeit ja nur Textspalten, die Zeitangaben enthalten, allerdings nicht im richtigen Format. Sie werden mit der Funktion "as\_datetime" in einen echten Zeitwert umgewandelt und anschließend mit "strftime()" wieder als drei verschiedene Zeichenketten (datum, uhrzeit, timestamp) ausgegeben. Aus dem Datum extrahieren wir das Jahr und den Monat, letzteren in doppelter Form: Einmal als Zahl 01 .. 12 und einmal als Text Jan .. Dez. Die beiden Komponenten fügen wir zusammen, um später sortierbare Monatsnamen zu haben. Weiter berechnen wir den Tag des Monats, sowie den Wochentag, wobei wir letzteren wieder aus seiner Ordnungszahl 01 .. 07 und seinem Kürzel "Mon" .. "Son" zusammensetzen. Schließlich berechnen wir noch die Tagesstunde des Postings.

In der nächsten Zeile berechnen wir den Umfang jedes Postings in Zeichen.

Die nächsten drei Berechnungen extrahieren bestimmte Informationen aus dem Textinhalt des Postings.

In den nächsten beiden Tabellenspalten speichern wir ab, welches Betriebssystem und welche Version von MAXQDA der Verfasser des Postings benutzt. In diesen beiden Textberechnungen arbeiten wir mit "Regular Expressions" (Regex). Regexes sind Suchmuster, mit denen viele Programmiersprachen oder Texteditoren Texte durchsuchen können. Jeder, der professionell mit der Analyse von Texten zu tun hat, sollte sie beherrschen. Die Regex in der Berechnung von "os\_ver" besagt z.B.:

- Finde beliebig viele (\*)
- beliebige Zeichen (.)
- am Ende einer Zeile (\$)
- sofern unmittelbar davor (?<=) die Zeichenfolge "System:" steht,

- und zwar am Anfang einer Zeile (^).

Den Startzeitpunkt, das Startdatum und die erste Poster\*in innerhalb eines Thread können wir dadurch gewinnen, dass wir die Postings auf ihrer Thread-URL gruppieren und die jeweils ersten Werte des Posting-Datums, der Posting Zeit und des Poster\*innen-Namens innerhalb der so entstehenden Gruppen auswählen.

Sobald wir über diese Angaben verfügen, können wir die Gruppierung wieder aufheben und als Letztes für das MAXQDA-Projekt einen Dokumentnamen und einen Dokumentgruppennamen berechnen. Beide sollen sinnvoll sortierbar sein. Die Dokumentgruppen sollen die Threads abbilden.

- Für die Dokumentgruppe kleben wir das Threaddatum und den Threadtitel (separiert durch "\_") zusammen. Vom Berechnungsergebnis behalten wir die ersten 63 Zeichen, denn länger darf ein Gruppen- oder Dokumentname in MAXQDA nicht sein.
- Den Dokumentnamen fügen wir aus dem Zeitstempel (Datum mit Uhrzeit) und dem Autorennamen zusammen.

Wir beenden das Ganze mit ein paar Aufräumarbeiten:

- wir sortieren die Tabelle mit der Funktion "arrange()" auf Dokumentgruppe und Dokumentname .
- Dann bringen wir mit "select()" die Spalten in die gewünschte Reihenfolge.

```
forum_postings <-
  postings %>%
  mutate(
    antworten = antworten %>% str_extract("^\\d+") %>% as.numeric(),
    views = views %>% str_extract("^\\d+") %>% as.numeric(),
    autor = str_trim(autor),
    datum_zeit = datum %>% strptime(format="%d %b %Y, %H:%M", tz="CET"),
    datum = datum_zeit %>% strftime("%Y-%m-%d"),
    uhrzeit = datum_zeit %>% strftime("%H:%M:%S"),
    timestamp = datum_zeit %>% strftime("%Y-%m-%d %H:%M:%S"),
    jahr = datum_zeit %>% strftime("%Y") %>% as.numeric(),
    monat = datum_zeit %>% strftime("%m %B"),
    jahr_monat = paste(jahr,monat, sep="_"),
    tag = datum_zeit %>% strftime("%d") %>% as.numeric(),
    wtag = datum_zeit %>% strftime("%u %A"),
    stunde=datum_zeit %>% strftime("%H") %>% as.numeric(),
    size = str_length(content),
    maxqda_ver = content %>% str_extract(regex("(?<=^Version: ).*$", multiline = TRUE)),
    os_ver = content %>% str_extract(regex("(?<=^System: ).*$", multiline = TRUE)) %>%
  group_by(thread_url) %>%
  mutate(thread_zeit = first(timestamp),
         thread_datum = first(datum),
         first_poster = first(autor)) %>%
  ungroup() %>%
  mutate(
    Dokumentgruppe = paste(strftime(thread_zeit, format="%Y-%m-%d"),
```

```

        thread_title,
        sep = "-" ) %>%
  str_sub(1, 63),
  Dokumentname = paste(timestamp, autor, sep = "_"),
  typ = if_else(autor==first_poster, "Frage", "Antwort")
) %>%
arrange(Dokumentgruppe,Dokumentname) %>%
select(Dokumentgruppe,Dokumentname,
  thread_datum, thread_zeit,
  thread_url,thread_title,
  antworten, views,
  datum,datum_zeit,
  uhrzeit, timestamp,size,
  jahr,monat, jahr_monat, tag, wtag, stunde,
  os_ver, maxqda_ver, typ,
  autor, titel ,first_poster, first, content)

```

Bevor wir unserer Daten exportieren, setzen wir noch das gewünschte Zielverzeichnis, in dem unsere Exporte landen sollen. Der Verzeichnispfad muss angepasst werden und das Kommentarzeichen # entfernt werden.

```
# setwd("~/unser/zielverzeichnis")
```

## 4.2 Projekt 2: Export als Excel-Datei

Jetzt sind wir soweit, dass wir unser Ergebnis nach Excel – und damit zu MAXQDA – exportieren können. Wir speichern unsere letzte Tabelle in einer EXCEL-Datei namens "MAXQDA\_mit\_R\_V2.xlsx". Vorher sortieren wir allerdings die Spaltenreihenfolge etwas um und lassen überflüssige Berechnungsspalten weg.

- Die Daten auf dem erzeugten Blatt sollen eine sortier- und filterbare Tabelle sein (asTable),
- bei der die Spaltenbreite automatisch festgelegt wird.

```

forum_postings %>%
select(Dokumentgruppe,Dokumentname,
  thread_url,thread_title,thread_datum, antworten, views,
  datum,uhrzeit, timestamp,
  jahr, monat, jahr_monat, tag, wtag, stunde,
  size, os_ver, maxqda_ver,
  titel,autor, first, typ, content) %>%
arrange(Dokumentgruppe, Dokumentname) %>%
write.xlsx(file = "MAXQDA_mit_R_V2.xlsx",
  asTable = TRUE,
  colWidth = "auto")

```

### 4.3 Projekt 2: Import der Excel-Datei

Die Exceldatei kann nun in MAXQDA importiert werden. Sie erzeugt ein Projekt wie es am Anfang als Projekttyp 2 beschrieben wurde.

Beim Import der Excel-Datei erscheint ein Importdialog, bei dem anzugeben ist, in welcher Form die Spalten der Excel-Importtabelle nach MAXQDA übernommen werden. Hier wählen wir für alle Spalten "Variable", lediglich "content" wird nur als Code importiert. Die Spalten "timestamp", "titel" und "autor" werden als "Code" und als "Variable" markiert.

Welche Spalte enthält die Bezeichnungen für ...

... die Dokumentgruppe?

... den Dokumentnamen?

Welche Spalten sollen als Text importiert und automatisch codiert werden (offene Fragen)?  
Welche Spalten sollen als Variablen importiert werden (geschlossene Fragen)?

Spalte	Code	Variable
Dokumentgruppe	<input type="checkbox"/>	<input type="checkbox"/>
Dokumentname	<input type="checkbox"/>	<input type="checkbox"/>
topic_url	<input type="checkbox"/>	<input checked="" type="checkbox"/>
topic_title	<input type="checkbox"/>	<input checked="" type="checkbox"/>
topic_datum	<input type="checkbox"/>	<input checked="" type="checkbox"/>
antworten	<input type="checkbox"/>	<input checked="" type="checkbox"/>
views	<input type="checkbox"/>	<input checked="" type="checkbox"/>
datum	<input type="checkbox"/>	<input checked="" type="checkbox"/>
uhrzeit	<input type="checkbox"/>	<input checked="" type="checkbox"/>
timestamp	<input type="checkbox"/>	<input checked="" type="checkbox"/>
jahr	<input type="checkbox"/>	<input checked="" type="checkbox"/>
monat	<input type="checkbox"/>	<input checked="" type="checkbox"/>
jahr_monat	<input type="checkbox"/>	<input checked="" type="checkbox"/>
tag	<input type="checkbox"/>	<input checked="" type="checkbox"/>
wtag	<input type="checkbox"/>	<input checked="" type="checkbox"/>
stunde	<input type="checkbox"/>	<input checked="" type="checkbox"/>
size	<input type="checkbox"/>	<input checked="" type="checkbox"/>
os_ver	<input type="checkbox"/>	<input checked="" type="checkbox"/>
maxqda_ver	<input type="checkbox"/>	<input checked="" type="checkbox"/>
titel	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
autor	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
first	<input type="checkbox"/>	<input checked="" type="checkbox"/>
typ	<input type="checkbox"/>	<input checked="" type="checkbox"/>
content	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Codes:  Variablen:

Optionen

Leere Zellen codieren

MAXQDA Import-Dialog

Das neue Projekt kann nun am einfachsten mit den vom “stats” Modul zur Verfügung gestellten Techniken ausgewertet werden (Häufigkeitsauszählungen und Kreuztabellen). Sobald eigene Codierungen hinzukommen, können die Methoden aus der “mixed methods”-Abteilung eingesetzt werden.

#### 4.4 Projekt 1: Export als strukturierter Text

Für die Projektvariante 1 - längere Texte (Threads) mit codierten Absätzen (Postings) - bereiten wir die Daten so auf, dass sie als “strukturierter Text” importiert werden können. Damit das Ergebnis in MAXQDA etwas formatierter erscheint, schreiben wir die Daten allerdings nicht in eine reine Text-Datei, sondern in eine HTML-Datei, in der wir die Posting-Köpfe optisch etwas hervorheben können. Strukturierter Textimport in MAXQDA funktioniert auch mit HTML-Dateien.

Vor dem Export modifizieren wir einen Teil der Tabellenspalten noch ein wenig, damit wir in MAXQDA nicht für jede Ausprägung einen separaten Code erhalten, sondern sinnvoll gruppierte hierarchische Codes. Dazu fügen wir mit “**mutate(across(...**” vor den entsprechenden Codes den Namen der Spalte ein, separiert durch einen “Backslash”. Den Namen der veränderten Spalten verpassen wir ein “\_c” am Ende. Dann wählen wir diese modifizierten Spalten für den Export aus.

```
forum_postings_1 <-
  forum_postings %>%
  mutate(
    thread_datum_2 = thread_datum,
    across(.cols=c(thread_datum,datum:autor),
           ~paste(names(forum_postings[cur_column()])),.x,sep="\\"),
           .names = "{col}_c") %>%
  arrange(Dokumentgruppe, Dokumentname) %>%
  select(Dokumentgruppe, Dokumentname,
         thread_url,thread_title, thread_datum_2, first_poster, jahr, monat, tag, stunde, antworten, views,
         timestamp, thread_datum_c, datum_c, jahr_c, monat_c, jahr_monat_c, tag_c, wtag_c, stunde_c,
         size, os_ver_c, maxqda_ver_c,
         autor, autor_c, titel, typ_c, content) %>%
  mutate(content = content %>% str_replace_all("\n", "<br>"))
```

Die resultierende Tabelle gruppieren wir nach Dokumentgruppe und schreiben die Werte von Datum/Zeit, Autor\*in und Text des Postings in eine HTML-Datei.

Immer wenn das Thema wechselt wird in dieser Datei mit #TEXT der Anfang eines neuen MAXQDA-Dokuments markiert, das als Namen den Dokumengruppenamen des EXCEL-Projektes erhält. Vor dem Beginn eines neuen Postings werden mit #CODE alle Codes dieses Postings eingefügt, verbunden durch “&&”. Nach der Ausgabe des Postingtextes wird dieser Block mit “#ENDCODE” abgeschlossen. Zeitangabe und Autor des Postings werden mit dem HTML-Tag “<strong>” hervorgehoben.

```
fn <- "MAXQDA_mit_R_V1.html"
if (file.exists(fn))
  file.remove(fn)
```

```
## [1] TRUE
```

```
cn<- file(fn, open = "a")

write(
  '<!DOCTYPE html>
  <html lang="de">
  <head>
  <meta charset="UTF-8" />
  </head>
  <body>',
  file=cn, append = TRUE)
forum_postings_1 %>%
  group_by(Dokumentgruppe) %>%
  group_split() %>%
  walk(function(thread){
    write(paste("<br>#TEXT ", thread$Dokumentgruppe[1]), file=cn, append = T)
    thread %>% split(1:nrow(.)) %>%
      walk(function(posting){
        write(paste("<br>#CODE ", paste(
          posting$thread_datum_c,
          posting$datum_c,
          posting$jahr_c, posting$monat_c, posting$jahr_monat_c, posting$tag_c, posting$wtag_c, posting$stunde_c,
          posting$sos_ver_c, posting$maxqda_ver_c, posting$autor_c, posting$typ_c, sep="\\&&")),
          file=cn, append = TRUE)
        write("<br><strong>", file=cn, append = TRUE)
        write(paste(posting$timestamp, posting$autor, "<br>", sep="<br>"), file=cn, append = TRUE)
        write("</strong>", file=cn, append = TRUE)
        write(posting$content, file=cn, append = TRUE)
        write("<br>#ENDCODE", file=cn, append = TRUE)
        write("<br><br>", file=cn, append = TRUE)
      })
    })
write("</body></html>", file=cn, append = TRUE)

close(cn)
```

Die resultierende HTML-Datei kann in MAXQDA als "strukturiertes Text" übernommen werden. In den Dokumenten dieses Projekts, sind alle Posting-Absätze mit allen Codes codiert. Datum/Zeit und Autor\*innennamen erscheinen im Fettdruck.

Die Auswertung solcher vielfach codierten Absätze in MAXQDA ist am einfachsten über den Menüpunkt "Statistik für Subcodes" im Kontextmenü der Codeliste möglich, hier können Verteilungsdiagramme und univariate Statistiken für die Codes ausgegeben werden. Komplexere Analysen über die Zusammenhänge zwischen den Codes ermöglichen z.B. das visuelle Tool "Code-Relations-Browser" und der Menüpunkt "Codes > Komplexe Codekonfigurationen".

## 4.5 Projekt 1 Erweiterung

Das "Projekt 1"-Design funktioniert zwar prinzipiell, es hat aber ein paar Nachteile.

- Erstens wird die Anzahl der Codings pro Absatz stark aufgebläht, wenn alle Codierungen, die für das ganze Dokument gelten, an jedem Absatz wiederholt werden (im vorliegenden Fall der Code "thread\_datum\_c"). Deshalb haben wir hier einige für die qualitative Analyse nicht ganz so wichtig erscheinende Informationen weggelassen (thread\_url, first\_poster).
- Numerische bzw. nicht kategoriale Daten eignen sich schlecht für diese Art von Codierung. Deshalb haben wir die Spalten "Antworten" und "Views" ebenfalls nicht benutzt.

Es ist aber relativ einfach, diese Informationen unserem Projekt 1 hinzuzufügen, nämlich in Form einer Dokumentvariablen-Tabelle, wie wir sie in Projekt 2 benutzen.

Um dieses Ziel zu erreichen, müssen wir nach dem Einlesen der HTML-Datei als strukturierter Text in unserem neuen MAXQDA-Projekt den von MAXQDA automatisch erzeugten Dokumentgruppennamen ändern. Wir nennen die Dokumentgruppe "Technikforum".

In R exportieren wir dann die thread-bezogenen Informationen in eine Excel-Tabelle.

```
forum_postings_1 %>%
  group_by(thread_url) %>%
  summarize(across(.cols=c(Dokumentgruppe:views),
                    .fns=first)) %>%
  mutate(Dokumentname = Dokumentgruppe,
         Dokumentgruppe = "Technikforum") %>%
  arrange(Dokumentname) %>%
  write.xlsx(asTable = T,
            colwidth = "auto",
            file = "thread_vars.xlsx")
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

Anschließend wechseln wir zurück nach MAXQDA und importieren diese Excel-Tabelle über den Menüpunkt "Variablen > Dokumentvariablen importieren".