Corona-Rechtsprechung des Bundesverfassungsgerichts

(BVerfG-Corona-Source)

Compilation Report

Version 2021-05-20

DOI: 10.5281/zenodo.4778814

Titel	Source Code der »Corona-Rechtsprechung des Bundesverfassungsgerichts«
Abkürzung	BVerfG-Corona-Source
Autor	Seán Fobbe
Version	2021-05-20
Download	https://doi.org/10.5281/zenodo.4778814
Lizenz	MIT No Attribution (MIT-0)

Zitiervorschlag

Seán Fobbe (2021). Source Code der »Corona-Rechtsprechung des Bundesverfassungsgerichts« (BVerfG-Corona-Source). Version 2021-05-20. Zenodo. DOI: 10.5281/zenodo.4778814.

Digital Object Identifier (DOI): Concept DOI und Version DOI

Soweit nicht anders angegeben ist die DOI immer eine »Version DOI« und bezieht sich nur auf eine bestimmte Version der Software. Sie verlinkt daher nur Version 2021-05-20. Für das Gesamtkonzept der Software steht eine »Concept DOI« zur Verfügung, die auf der Zenodo-Seite jeder Version unter »Cite all versions?« zu finden ist. Sie lautet 10.5281/zenodo.4459415. Die »Concept DOI« verlinkt immer die aktuellste Version.

Lizenz: MIT No Attribution (MIT-0)

Disclaimer

Dieser Datensatz ist eine private wissenschaftliche Initiative und steht in keiner Verbindung zu Behörden, Gerichten oder anderen amtlichen Stellen der Bundesrepublik Deutschland.

Inhaltsverzeichnis

1 Einleitung

1.1 Überblick

Dieses R-Skript lädt den Corpus der Entscheidungen des Bundesverfassungsgerichts (CE-BVerfG) herunter, untersucht ihn auf mit SARS-CoV-2 assoziiertem Vokabular und speichert relevante Entscheidungen. Es ist die Grundlage für den Datensatz **Corona-Rechtsprechung des Bundesverfassungsgerichts (BVerfG-Corona)**.

Alle mit diesem Skript erstellten Datensätze werden dauerhaft kostenlos und urheberrechtsfrei auf Zenodo, dem wissenschaftlichen Archiv des CERN, veröffentlicht. Alle Versionen sind mit einem persistenten Digital Object Identifier (DOI) versehen. Die neueste Version des Datensatzes ist immer über den Link der Concept DOI erreichbar: https://doi.org/10.5281/zenodo.4459405

1.2 Funktionsweise

Primäre Endprodukte des Skripts sind folgende ZIP-Archive:

1. Alle Corona-relevanten Entscheidungen im PDF-Format
2. Alle Corona-relevanten Entscheidungen im TXT-Format
3. Alle Analyse-Ergebnisse (Tabellen als CSV, Grafiken als PDF und PNG)
4. Der Source Code und alle weiteren Quelldaten

Zusätzlich werden für alle ZIP-Archive kryptographische Signaturen (SHA2-256 und SHA3-512) berechnet und in einer CSV-Datei hinterlegt. Es kann optional ein PDF-Bericht erstellt werden (siehe unter »Kompilierung«).

1.3 Systemanforderungen

Das Skript in seiner veröffentlichten Form kann nur unter Linux ausgeführt werden, da es Linux-spezifische Optimierungen (z.B. Fork Cluster) und Shell-Kommandos (z.B. OpenSSL) nutzt. Das Skript wurde unter Fedora Linux entwickelt und getestet. Die zur Kompilierung benutzte Version entnehmen Sie bitte dem **sessionInfo()**-Ausdruck am Ende dieses Berichts.

In der Standard-Einstellung wird das Skript vollautomatisch die maximale Anzahl an Rechenkernen/Threads auf dem System zu nutzen. Wenn die Anzahl Threads (Variable »fullCores«) auf 1 gesetzt wird, ist die Parallelisierung deaktiviert.

Auf der Festplatte sollten 4 GB Speicherplatz vorhanden sein.

Um die PDF-Berichte kompilieren zu können benötigen Sie das R package **rmarkdown**, eine vollständige Installation von LaTeX und alle in der Präambel-TEX-Datei angegebenen LaTeX Packages.

1.4 Kompilierung

Mit der Funktion **render()** von **rmarkdown** kann der **vollständige Datensatz** kompiliert und das Skript mitsamt seinen Rechenergebnisse in ein gut lesbares PDF-Format überführt werden.

Alle Kommentare sind im roxygen2-Stil gehalten. Das Skript kann daher auch **ohne render()** regulär als R-Skript ausgeführt werden. Es wird in diesem Fall kein PDF-Bericht erstellt und Diagramme werden nicht abgespeichert.

Um den vollständigen Datensatz zu kompilieren und einen PDF-Bericht zu erstellen, kopieren Sie bitte alle im Source-Archiv bereitgestellten Dateien in einen leeren Ordner und führen mit R diesen Befehl aus:

```
rmarkdown::render(input = "BVerfG-Corona_Source_CorpusCreation.R",
                  output_file = paste0("BVerfG-Corona_2021-01-08_
    CompilationReport.pdf"),
                  envir = new.env())
```

2 Parameter

2.1 Name des Datensatzes

```
datasetname <- "BVerfG-Corona"
```

2.2 Datumsstempel

Dieser Datumsstempel wird in alle Dateinamen eingefügt. Er richtet sich nach der Version des Stamm-Datensatzes.

```
datestamp <- "2021-05-20"
```

2.3 DOI der konkreten Datensatz-Version (CE-BVerfG)

Aus diesem Datensatz werden die Entscheidungen bezogen.

```
doi.version.cebverfg <- "10.5281/zenodo.4765622" # checked
```

2.4 DOI der konkreten Datensatz-Version (BVerfG-Corona)

In diesen Datensatz werden die Corona-relevanten Entscheidungen überführt.

```
doi.version <- "10.5281/zenodo.4778817" # checked
```

2.5 Verzeichnis für Analyse-Ergebnisse

Hinweis: Muss mit einem Schrägstrich enden!

```
outputdir <- paste0(getwd(),
                    "/ANALYSE/")
```

2.6 Optionen: Quanteda

```
tokens_locale <- "de_DE"
```

2.7 Optionen: Knitr

2.7.1 Ausgabe-Format

```
dev <- c("pdf", "png")
```

2.7.2 DPI für Raster-Grafiken

```
dpi <- 300
```

2.7.3 Ausrichtung von Grafiken im Compilation Report

```
fig.align <- "center"
```

2.8 Frequenztabellen: Ignorierte Variablen

Diese Variablen werden bei der Erstellung der Frequenztabellen nicht berücksichtigt.

```
varremove <- c("text",
               "eingangsnummer",
               "datum",
               "doc_id",
               "seite",
               "name",
               "ecli",
               "aktenzeichen",
               "zeichen",
               "tokens",
               "typen",
               "saetze",
               "version")
```

3 Vorbereitung

3.1 Datum und Uhrzeit (Beginn)

```
begin.script <- Sys.time()
print(begin.script)
```

```
## [1] "2021-05-21 20:21:28 CEST"
```

3.2 Ordner für Analyse-Ergebnisse erstellen

```
dir.create(outputdir)
```

3.3 Packages

```
library(doParallel)   # Parallelisierung
```

```
## Loading required package: foreach
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
library(ggplot2)      # Fortgeschrittene Datenvisualisierung
library(rmarkdown)    # Wissenschaftliches Reporting
library(knitr)        # Wissenschaftliches Reporting
library(kableExtra)   # Verbesserte Kable Tabellen
library(data.table)   # Fortgeschrittene Datenverarbeitung
```

```
## data.table 1.14.0 using 8 threads (see ?getDTthreads).  Latest news: r-
    datatable.com
```

```
library(quanteda)     # Fortgeschrittenes Natural Language Processing
```

```
## Package version: 3.0.0
## Unicode version: 12.1
## ICU version: 65.1
```

```
## Parallel computing: 16 of 16 threads used.
```

```
## See https://quanteda.io for tutorials and examples.
```

```
library(quanteda.textplots) # Quanteda: Diagramme
```

3.4 Zusätzliche Funktionen einlesen

Hinweis: Die hieraus verwendeten Funktionen werden jeweils vor der ersten Benutzung in vollem Umfang angezeigt um den Lesefluss zu verbessern.

```
source("General_Source_Functions.R")
```

3.5 Quanteda-Optionen setzen

```
quanteda_options(tokens_locale = tokens_locale)
```

3.6 Knitr Optionen setzen

```
knitr::opts_chunk$set(fig.path = outputdir,
                      dev = dev,
                      dpi = dpi,
                      fig.align = fig.align)
```

3.7 Vollzitate statistischer Software

```
knitr::write_bib(c(.packages()),
                 "packages.bib")
```

```
## tweaking foreach
```

3.8 Parallelisierung aktivieren

Parallelisierung wird zur Datenanalyse mittels **quanteda** und **data.table** verwendet. Die Anzahl Threads wird automatisch auf das verfügbare Maximum des Systems gesetzt, kann aber auch nach Belieben auf das eigene System angepasst werden. Die Parallelisierung kann deaktiviert werden, indem die Variable **fullCores** auf 1 gesetzt wird.

Die hier verwendete Funktion **makeForkCluster()** ist viel schneller als die Alternativen, funktioniert aber nur auf Unix-basierten Systemen (Linux, MacOS).

3.8.1 Anzahl logischer Kerne bestimmen

```
fullCores <- detectCores()
print(fullCores)
```

```
## [1] 16
```

3.8.2 Quanteda

```
quanteda_options(threads = fullCores)
```

3.8.3 Data.table

```
setDTthreads(threads = fullCores)
```

4 Stamm-Datensatz einlesen (CE-BVerfG)

Der Stamm-Datensatz ist der »Corpus der Entscheidungen des Bundesverfassungsgerichts« (CE-BVerfG). Dieser enthält alle vom Bundesverfassungsgericht seit 1998 veröffentlichten Entscheidungen. Dessen **aktuellste** Version ist immer über diesen Digital Object Identifier (DOI) abrufbar: https://doi.org/10.5281/zenodo.3902658

4.1 Download der CSV-Datei

Der Datensatz im CSV-Format wird automatisch über einen verschlüsselten und langzeitstabilen Link aus dem wissenschaftlichen Archiv des CERN heruntergeladen. Dieses Vorgehen garantiert die Verwendung einer authentischen Version des Datensatzes.

```
zip.csv <- paste0("CE-BVerfG_",
                  datestamp,
                  "_DE_CSV_Datensatz.zip")


link.csv <- paste0("https://zenodo.org/record/",
                   gsub("10\\.5281/zenodo\\.([0-9]+)",
                        "\\1",
                        doi.version.cebverfg),
                   "/files/",
                   zip.csv,
                   "?download=1")


if(file.exists(zip.csv) == FALSE){

    download.file(link.csv,
                  zip.csv)

}
```

4.2 CSV-Datei einlesen

```
dt.bverfg <- fread(cmd = paste("unzip -cq",
                               zip.csv))
```

4.3 ZIP-Archiv löschen

```
unlink(zip.csv)
```

4.4 Korpus-Objekt erstellen

```
corpus.bverfg <- corpus(dt.bverfg)
```

5 Keywords in Context (KWIC)

Bei einer KWIC-Analyse (keywords in context) wird nach einer bestimmten Zeichengefolge gesucht und sowohl diese, als auch die angrenzenden Wörter werden angezeigt. Konkret wird an dieser Stelle eine alternative Suche nach den Mustern »Corona«, »COVID« oder »SARS-CoV« durchgeführt. Groß- und Kleinschreibung wird ignoriert um eventuelle Tippfehler zu vernachlässigen. Das Sichtfenster wird auf 15 Tokens vor und nach dem Treffer gesetzt.

5.1 Tokenisierung

```
tokens <- tokens(corpus.bverfg,
                 what = "word",
                 remove_punct = FALSE,
                 remove_symbols = FALSE,
                 remove_numbers = FALSE,
                 remove_url = FALSE,
                 remove_separators = TRUE,
                 split_hyphens = FALSE,
                 include_docvars = TRUE,
                 padding = FALSE)
```

5.2 KWIC-Analyse durchführen

```
kwic <- kwic(tokens,
             pattern = "(Corona)|(COVID)|(SARS-CoV)",
             window = 15,
             valuetype = "regex",
             case_insensitive = TRUE)
```

5.3 KWIC-Tabelle speichern

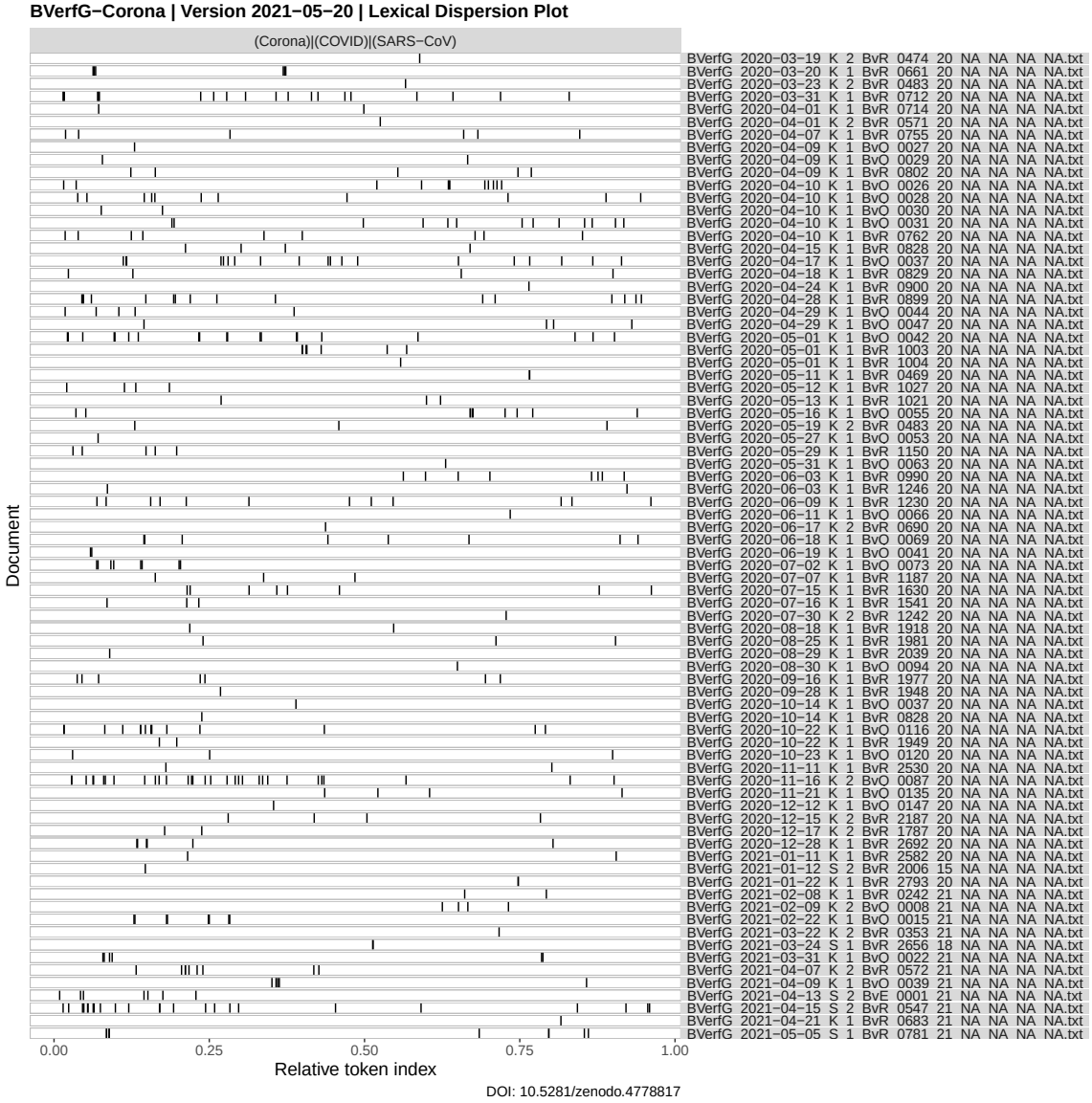
```
file.kwic.sansdate <- paste(datasetname,
                            "02_KeywordsInContext.csv",
                            sep = "_")

file.kwic.date <- paste(datasetname,
                        datestamp,
                        "ANALYSE_02_KeywordsInContext.csv",
                        sep = "_")


fwrite(data.frame(kwic),
       paste0(outputdir,
              file.kwic.sansdate))


fwrite(data.frame(kwic),
       file.kwic.date)
```

6 Lexical Dispersion Plot

Lexical Dispersion Plots zeigen mit einem vertikalen Strich an, an welcher Stelle in einem Dokument sich ein Token befindet. Alle Dokumente sind auf eine Länge von 1.0 normalisiert, d.h. ein Wert von 0.5 heißt immer, dass sich das Token in der Mitte des jeweiligen Dokumentes befindet. Viele und/oder dicke Striche deuten auf eine große Häufigkeit des Tokens hin.

6.1 Rechteckiges Format

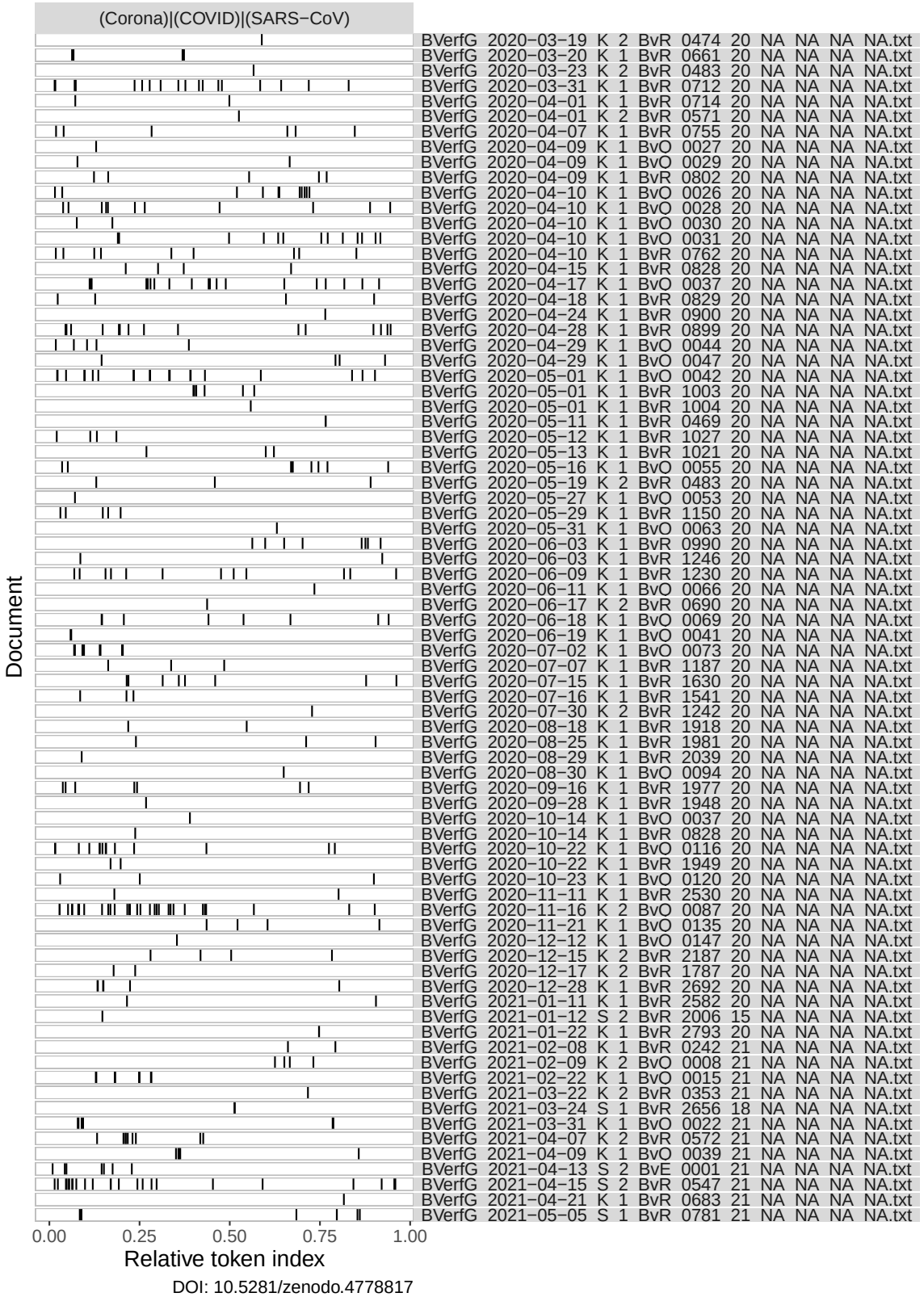
```
textplot_xray(kwic, scale = "relative")+
    labs(
        title = paste(datasetname,
                      "| Version",
                      datestamp,
                      "| Lexical Dispersion Plot"),
        caption = paste("DOI:",
                        doi.version))+
    theme(
        text = element_text(size = 14),
        plot.title = element_text(size = 14,
                                  face = "bold"),
        legend.position = "none",
        plot.margin = margin(10, 20, 10, 10)
    )
```



6.2 A4-Format

```
textplot_xray(kwic, scale = "relative")+
    labs(
        title = paste(datasetname,
                      "| Version",
                      datestamp,
                      "| Lexical Dispersion Plot"),
        caption = paste("DOI:",
                        doi.version))+
    theme(
        text = element_text(size = 14),
        plot.title = element_text(size = 14,
                                  face = "bold"),
        legend.position = "none",
        plot.margin = margin(10, 20, 10, 10)
    )
```

BVerfG−Corona | Version 2021−05−20 | Lexical Dispersion Plot



DOI: 10.5281/zenodo.4778817

7 TXT-Datensatz erstellen

7.1 Namen der Corona-Entscheidungen definieren

```
keep.txt <- unique(kwic$docname)
```

7.2 Anzahl der TXT-Dateien

```
length(keep.txt)
```

```
## [1] 78
```

7.3 TXT-Datensatz herunterladen

```
zip.txt <- paste0("CE-BVerfG_",
                  datestamp,
                  "_DE_TXT_Datensatz.zip")


link.txt <- paste0("https://zenodo.org/record/",
                   gsub("10\\.5281/zenodo\\.([0-9]+)",
                        "\\1",
                        doi.version.cebverfg),
                   "/files/",
                   zip.txt,
                   "?download=1")


if(file.exists(zip.txt) == FALSE){

    download.file(link.txt,
                  zip.txt)

}
```

7.4 ZIP-Archiv entpacken

```
unzip(zip.txt)
```

7.5 Corona-Entscheidungen verpacken

```
zip(paste(datasetname,
          datestamp,
          "DE_TXT_Datensatz.zip",
          sep = "_"),
    keep.txt)
```

7.6 TXT-Dateien löschen

```
files.txt <- list.files(pattern = ".txt")

unlink(files.txt)
```

7.7 ZIP-Archiv löschen

```
unlink(zip.txt)
```

8 PDF-Datensatz erstellen

8.1 Namen der Corona-Entscheidungen definieren

```
keep.pdf <- gsub(".txt",
                 ".pdf",
                 keep.txt)
```

8.2 Anzahl der PDF-Dateien

```
length(keep.pdf)
```

```
## [1] 78
```

8.3 PDF-Datensatz herunterladen

```
zip.pdf <- paste0("CE-BVerfG_",
                  datestamp,
                  "_DE_PDF_Datensatz.zip")


link.pdf <- paste0("https://zenodo.org/record/",
                   gsub("10\\.5281/zenodo\\.([0-9]+)",
                        "\\1",
                        doi.version.cebverfg),
                   "/files/",
                   zip.pdf,
                   "?download=1")


if(file.exists(zip.pdf) == FALSE){

    download.file(link.pdf,
                  zip.pdf)

}
```

8.4 ZIP-Archiv entpacken

```
unzip(zip.pdf)
```

8.5 Corona-Entscheidungen verpacken

```
zip(paste(datasetname,
          datestamp,
          "DE_PDF_Datensatz.zip",
          sep = "_"),
    keep.pdf)
```

8.6 PDF-Dateien löschen

```
files.pdf <- list.files(pattern = ".pdf")

unlink(files.pdf)
```

8.7 ZIP-Archiv löschen

```
unlink(zip.pdf)
```

9 Frequenztabellen erstellen

9.1 CE-BVerfG auf Corona-Entscheidungen reduzieren

```
dt.corona <- dt.bverfg[doc_id %in% keep.txt]
```

9.2 Funktion anzeigen

```
print(f.fast.freqtable)
```

function(x, varlist = names(x), sumrow = TRUE, output.list = TRUE, output.kable = FALSE, output.csv = FALSE, outputdir = »./«, prefix = "», align =«r"){

```
## Begin List
freqtable.list <- vector("list", length(varlist))

## Calculate Frequency Table
for (i in seq_along(varlist)){

    varname <- varlist[i]

    freqtable <- x[, .N, keyby=c(paste0(varname))]

    freqtable[, c("exactpercent",
                  "roundedpercent",
                  "cumulpercent") := {
                      exactpercent  <-  N/sum(N)*100
                      roundedpercent <- round(exactpercent, 2)
                      cumulpercent <- round(cumsum(exactpercent), 2)
                      list(exactpercent,
                           roundedpercent,
                           cumulpercent)}]

    ## Calculate Summary Row
    if (sumrow == TRUE){
        colsums <-  cbind("Total",
                          freqtable[, lapply(.SD, function(x){round(sum(x))}),
                                    .SDcols = c("N",
                                                "exactpercent",
                                                "roundedpercent")
                                    ], round(max(freqtable$cumulpercent)))

        colnames(colsums)[c(1,5)] <- c(varname, "cumulpercent")
        freqtable <- rbind(freqtable, colsums)
    }

    ## Add Frequency Table to List
    freqtable.list[[i]] <- freqtable

    ## Write CSV
```

```
    if (output.csv == TRUE){

        fwrite(freqtable,
               paste0(outputdir,
                      prefix,
                      varname,
                      ".csv"),
               na = "NA")

    }

    ## Output Kable
    if (output.kable == TRUE){

        cat("\n-----------------------------------------------\n")
        cat(paste0("Frequency Table for Variable:   ", varname, "\n"))
        cat("-----------------------------------------------\n")
        cat(paste0("\n ",
                   x[, .N, keyby=c(paste0(varname))][,.N],
                   " unique value(s) detected.\n\n"))


        print(kable(freqtable,
                    format = "latex",
                    align = align,
                    booktabs = TRUE,
                    longtable = TRUE) %>% kable_styling(latex_options = "repeat_
    header"))
    }
}

## Return List of Frequency Tables
if (output.list == TRUE){
    return(freqtable.list)
}
```

}

9.3 Ignorierte Variablen

```
print(varremove)
```

```
##  [1] "text"           "eingangsnummer" "datum"          "doc_id"
##  [5] "seite"          "name"           "ecli"           "aktenzeichen"
##  [9] "zeichen"        "tokens"         "typen"          "saetze"
## [13] "version"
```

9.4 Liste zu prüfender Variablen

```
varlist <- names(dt.corona)

varlist <- setdiff(varlist,
                   varremove)

print(varlist)
```

```
##  [1] "gericht"           "spruchkoerper_typ" "spruchkoerper_az"
##  [4] "registerzeichen"   "eingangsjahr_az"   "kollision"
##  [7] "band"              "entscheidungsjahr" "eingangsjahr_iso"
## [10] "praesi"            "v_praesi"          "verfahrensart"
## [13] "entscheidung_typ"  "doi_concept"       "doi_version"
## [16] "lizenz"
```

9.5 Frequenztabellen erstellen

```
prefix <- paste0(datasetname,
                 "_00_Frequenztabelle_var-")
```

```
f.fast.freqtable(dt.corona,
                 varlist = varlist,
                 sumrow = TRUE,
                 output.list = FALSE,
                 output.kable = TRUE,
                 output.csv = TRUE,
                 outputdir = outputdir,
                 prefix = prefix,
                 align = c("p{5cm}",
                           rep("r", 4)))
```

Frequency Table for Variable: gericht

1 unique value(s) detected.

gericht	N	exactpercent	roundedpercent	cumulpercent
BVerfG	78	100	100	100
Total	78	100	100	100

Frequency Table for Variable: spruchkoerper_typ

2 unique value(s) detected.

spruchkoerper_typ	N	exactpercent	roundedpercent	cumulpercent
K	73	93.589744	93.59	93.59
S	5	6.410256	6.41	100.00
Total	78	100.000000	100.00	100.00

Frequency Table for Variable: spruchkoerper_az

2 unique value(s) detected.

spruchkoerper_az	N	exactpercent	roundedpercent	cumulpercent
1	63	80.76923	80.77	80.77
2	15	19.23077	19.23	100.00
Total	78	100.00000	100.00	100.00

Frequency Table for Variable: registerzeichen

3 unique value(s) detected.

registerzeichen	N	exactpercent	roundedpercent	cumulpercent
BvE	1	1.282051	1.28	1.28
BvQ	28	35.897436	35.90	37.18
BvR	49	62.820513	62.82	100.00
Total	78	100.000000	100.00	100.00

Frequency Table for Variable: eingangsjahr_az

4 unique value(s) detected.

eingangsjahr_az	N	exactpercent	roundedpercent	cumulpercent
15	1	1.282051	1.28	1.28
18	1	1.282051	1.28	2.56
20	65	83.333333	83.33	85.90
21	11	14.102564	14.10	100.00
Total	78	100.000000	100.00	100.00

Frequency Table for Variable: kollision

1 unique value(s) detected.

kollision	N	exactpercent	roundedpercent	cumulpercent
NA	78	100	100	100
Total	78	100	100	100

Frequency Table for Variable: band

1 unique value(s) detected.

band	N	exactpercent	roundedpercent	cumulpercent
NA	78	100	100	100
Total	78	100	100	100

Frequency Table for Variable: entscheidungsjahr

2 unique value(s) detected.

entscheidungsjahr	N	exactpercent	roundedpercent	cumulpercent
2020	63	80.76923	80.77	80.77
2021	15	19.23077	19.23	100.00
Total	78	100.00000	100.00	100.00

Frequency Table for Variable: eingangsjahr_iso

4 unique value(s) detected.

eingangsjahr_iso	N	exactpercent	roundedpercent	cumulpercent
2015	1	1.282051	1.28	1.28
2018	1	1.282051	1.28	2.56
2020	65	83.333333	83.33	85.90
2021	11	14.102564	14.10	100.00
Total	78	100.000000	100.00	100.00

Frequency Table for Variable: praesi

2 unique value(s) detected.

praesi	N	exactpercent	roundedpercent	cumulpercent
Harbarth	38	48.71795	48.72	48.72
Voßkuhle	40	51.28205	51.28	100.00
Total	78	100.00000	100.00	100.00

Frequency Table for Variable: v_praesi

2 unique value(s) detected.

v_praesi	N	exactpercent	roundedpercent	cumulpercent
Harbarth	40	51.28205	51.28	51.28
König	38	48.71795	48.72	100.00
Total	78	100.00000	100.00	100.00

Frequency Table for Variable: verfahrensart

3 unique value(s) detected.

verfahrensart	N	exactpercent	roundedpercent	cumulpercent
Einstweilige Anordnungen	28	35.897436	35.90	35.90
Organstreitverfahren	1	1.282051	1.28	37.18
Verfassungsbeschwerden; Kommunalverfassungsbeschwerden	49	62.820513	62.82	100.00
Total	78	100.000000	100.00	100.00

Frequency Table for Variable: entscheidung_typ

3 unique value(s) detected.

entscheidung_typ	N	exactpercent	roundedpercent	cumulpercent
B	59	75.641026	75.64	75.64
U	1	1.282051	1.28	76.92
V	18	23.076923	23.08	100.00
Total	78	100.000000	100.00	100.00

Frequency Table for Variable: doi_concept

1 unique value(s) detected.

doi_concept	N	exactpercent	roundedpercent	cumulpercent
10.5281/zenodo.3902658	78	100	100	100
Total	78	100	100	100

Frequency Table for Variable: doi_version

1 unique value(s) detected.

doi_version	N	exactpercent	roundedpercent	cumulpercent
10.5281/zenodo.4765622	78	100	100	100
Total	78	100	100	100

Frequency Table for Variable: lizenz

1 unique value(s) detected.

lizenz	N	exactpercent	roundedpercent	cumulpercent
Creative Commons Zero 1.0 Universal	78	100	100	100
Total	78	100	100	100

10 Diagramm Kopieren

```
rechteckig <- list.files(outputdir, pattern = "Rechteckig.*\\.pdf",
                         full.names = TRUE)

rechteckig.path <- gsub("//",
                        "/",
                        rechteckig)

rechteckig.file <- gsub(".+//(.+)",
                        "\\1",
                        rechteckig)

rechteckig.file <- gsub("01",
                        paste0(datestamp,
                               "_ANALYSE_01"),
                        rechteckig.file)

rechteckig.file <- gsub("-1",
                        "",
                        rechteckig.file)


file.copy(rechteckig.path,
          rechteckig.file)
```

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

11 Erstellen der ZIP-Archive

11.1 Verpacken der Analyse-Dateien

```
zip(paste0(datasetname,
           "_",
           datestamp,
           "_DE_",
           basename(outputdir),
           ".zip"),
    basename(outputdir))
```

11.2 Verpacken der Source-Dateien

```
files.source <- c(list.files(pattern = "Source"),
                  "buttons")


files.source <- grep("spin",
                     files.source,
                     value = TRUE,
                     ignore.case = TRUE,
                     invert = TRUE)

zip(paste(datasetname,
           datestamp,
           "Source_Files.zip",
           sep = "_"),
    files.source)
```

12 Kryptographische Hashes

Dieses Modul berechnet für jedes ZIP-Archiv zwei Arten von Hashes: SHA2-256 und SHA3-512. Mit diesen kann die Authentizität der Dateien geprüft werden und es wird dokumentiert, dass sie aus diesem Source Code hervorgegangen sind. Die SHA-2 und SHA-3 Algorithmen gelten derzeit als sicher und ein SHA3-Hash mit 512 bit Länge ist nach derzeitigem Wissen auch gegenüber quantenkryptoanalytischen Verfahren hinreichend resistent.

12.1 Liste der ZIP-Archive erstellen

```
files.zip <- list.files(pattern = "\\.zip$",
                        ignore.case = TRUE)
```

12.2 Funktion anzeigen

```
print(f.dopar.multihashes)
```

function(x, threads = detectCores()){

```
print(paste("Parallel processing using", threads, "threads."))

begin <- Sys.time()

cl <- makeForkCluster(threads)
registerDoParallel(cl)

multihashes <- foreach(filename = x,
                       .errorhandling = 'pass',
                       .combine = 'rbind') %dopar% {

                           sha2.256 <- system2("openssl",
                                               paste("sha256",
                                                     filename),
                                               stdout = TRUE)

                           sha2.256 <- gsub("^.*\\= ",
                                            "",
                                            sha2.256)

                           sha3.512 <- system2("openssl",
                                               paste("sha3-512",
                                                     filename),
                                               stdout = TRUE)

                           sha3.512 <- gsub("^.*\\= ",
                                            "",
                                            sha3.512)
```

```
                            out <- data.frame(filename,
                                              sha2.256,
                                              sha3.512)
                            return(out)
                        }
stopCluster(cl)

end <- Sys.time()
duration <- end - begin

print(paste0("Processed ",
             length(x),
             " files. Runtime was ",
             round(duration,
                   digits = 2),
             " ",
             attributes(duration)$units,
             "."))

return(multihashes)
```

}

12.3 Hashes berechnen

```
multihashes <- f.dopar.multihashes(files.zip)
```

```
## [1] "Parallel processing using 16 threads."
## [1] "Processed 4 files. Runtime was 0.32 secs."
```

12.4 In Data Table umwandeln

```
setDT(multihashes)
```

12.5 Index hinzufügen

```
multihashes$index <- seq_len(multihashes[,.N])
```

12.6 In Datei schreiben

```
fwrite(multihashes,
       paste(datasetname,
             datestamp,
             "KryptographischeHashes.csv",
             sep = "_"),
       na = "NA")
```

12.7 Leerzeichen hinzufügen um Zeilenumbruch zu ermöglichen

```
multihashes$sha3.512 <- paste(substr(multihashes$sha3.512, 1, 64),
                              substr(multihashes$sha3.512, 65, 128))
```

12.8 In Bericht anzeigen

```
kable(multihashes[,.(index,filename)],
      format = "latex",
      align = c("p{1cm}",
                "p{13cm}"),
      booktabs = TRUE,
      longtable = TRUE)
```

index	filename
1	BVerfG-Corona_2021-05-20_DE_ANALYSE.zip
2	BVerfG-Corona_2021-05-20_DE_PDF_Datensatz.zip
3	BVerfG-Corona_2021-05-20_DE_TXT_Datensatz.zip
4	BVerfG-Corona_2021-05-20_Source_Files.zip

```
kable(multihashes[,.(index,sha2.256)],
      format = "latex",
      align = c("c",
                "p{13cm}"),
      booktabs = TRUE,
      longtable = TRUE)
```

index	sha2.256
1	db148459407fa42d232e6b6018e488766648a5baaeba339e64aa34ba2fcfc09f
2	1667b3497fe43d9298e649784b70127479657d5c6e0044ff2de131ce08b6a60a
3	3623445f4a8b106452c59ddf8b7d64e7680c774a4c053c241906cdb0d37f9350
4	aba4fcdf0bff839ff2ea261ff17a4c07b3c3e4bcc4af170f01d3ea99f9a952b2

```
kable(multihashes[,.(index,sha3.512)],
      format = "latex",
      align = c("c",
                "p{13cm}"),
      booktabs = TRUE,
      longtable = TRUE)
```

index	sha3.512
1	820aa90cdd120caee5b3e0047c53cefd2367531e534e32bdf74d61186c32692e ac03a9217d4fa731b8a2f62c05471bc7d487da66764f514997688995020c88a8
2	9e8ac8fdb18266208be071aad4c9ac8269bc7eaa6387a55c4542f3530c09aca5 336d028773662c0e8c3a41c9cde47c1de05114750f1bb406258eda720ab76cfc
3	fcca33d7f5b89b6eff7a434cd6025ea3a355c4ef6bff54744c4e451c724fce9a a3763c31d5593bbb11b073dac236ca6dfb7ba7868dc1af7ceab505d081ba0337
4	ef33cdd236b978e1efd90fd48551ecfe2b88e38b54e0882e51daa20ca63499a1 01952170b91ba1ce6ee4300c04029588446a0d3efb1ed51c326e33dae8d6581d

13 Abschluss

13.1 Datumsstempel

Hinweis: der Datumsstempel weicht vom Zeitpunkt der tatsächlichen Erstellung des Datensatzes ab, weil sich der Datumsstempel nach dem Tag des Abrufs des CE-BVerfG richtet.

```
print(datestamp)
```

```
## [1] "2021-05-20"
```

13.2 Datum und Uhrzeit (Anfang)

```
print(begin.script)
```

```
## [1] "2021-05-21 20:21:28 CEST"
```

13.3 Datum und Uhrzeit (Ende)

```
end.script <- Sys.time()
print(end.script)
```

```
## [1] "2021-05-21 20:23:30 CEST"
```

13.4 Laufzeit des gesamten Skriptes

```
print(end.script - begin.script)
```

```
## Time difference of 2.03394 mins
```

13.5 Warnungen

```
warnings()
```

14 Parameter für strenge Replikationen

```
system2("openssl", "version", stdout = TRUE)
```

```
## [1] "OpenSSL 1.1.1k  FIPS 25 Mar 2021"
```

```
sessionInfo()
```

```
## R version 4.0.5 (2021-03-31)
## Platform: x86_64-redhat-linux-gnu (64-bit)
## Running under: Fedora 33 (Workstation Edition)
##
## Matrix products: default
## BLAS/LAPACK: /usr/lib64/libopenblas-r0.3.12.so
##
## locale:
##  [1] LC_CTYPE=en_US.utf8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.utf8        LC_COLLATE=en_US.utf8
##  [5] LC_MONETARY=en_US.utf8    LC_MESSAGES=en_US.utf8
##  [7] LC_PAPER=en_US.utf8       LC_NAME=C
##  [9] LC_ADDRESS=C              LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.utf8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] quanteda.textplots_0.94 quanteda_3.0.0          data.table_1.14.0
##  [4] kableExtra_1.3.4        knitr_1.33              rmarkdown_2.8
##  [7] ggplot2_3.3.3           doParallel_1.0.16       iterators_1.0.13
## [10] foreach_1.5.1
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_1.1.1   xfun_0.23          purrr_0.3.4        lattice_0.20-41
##  [5] colorspace_2.0-0   vctrs_0.3.8        generics_0.1.0     htmltools
    _0.5.1.1
##  [9] viridisLite_0.4.0  yaml_2.2.1         utf8_1.2.1         rlang_0.4.11
## [13] pillar_1.6.0       glue_1.4.2         withr_2.4.2        lifecycle_1.0.0
## [17] stringr_1.4.0      munsell_0.5.0      gtable_0.3.0       rvest_1.0.0
## [21] codetools_0.2-18   evaluate_0.14      labeling_0.4.2     fansi_0.4.2
## [25] highr_0.9          Rcpp_1.0.6         scales_1.1.1       magick_2.7.2
## [29] RcppParallel_5.1.4 webshot_0.5.2      farver_2.1.0       systemfonts
    _1.0.2
## [33] fastmatch_1.1-0    stopwords_2.2      digest_0.6.27      stringi_1.5.3
## [37] dplyr_1.0.6        grid_4.0.5         tools_4.0.5        magrittr_2.0.1
## [41] tibble_3.1.0       crayon_1.4.1       pkgconfig_2.0.3    ellipsis_0.3.2
## [45] Matrix_1.3-3       xml2_1.3.2         svglite_2.0.0      httr_1.4.2
## [49] rstudioapi_0.13    R6_2.5.0           compiler_4.0.5
```

Literaturverzeichnis

Allaire, JJ, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone. 2021. *Rmarkdown: Dynamic Documents for R*. https://CRAN.R-project.org/package=rmarkdown.

Analytics, Revolution, and Steve Weston. 2020. *Iterators: Provides Iterator Construct*. https://github.com/RevolutionAnalytics/iterators.

Benoit, Kenneth, Kohei Watanabe, Haiyan Wang, Paul Nulty, Adam Obeng, Stefan Müller, and Akitaka Matsuo. 2018a. "Quanteda: An R Package for the Quantitative Analysis of Textual Data." *Journal of Open Source Software* 3 (30): 774. https://doi.org/10.21105/joss.00774.

———. 2018b. "Quanteda: An R Package for the Quantitative Analysis of Textual Data." *Journal of Open Source Software* 3 (30): 774. https://doi.org/10.21105/joss.00774.

Benoit, Kenneth, Kohei Watanabe, Haiyan Wang, Paul Nulty, Adam Obeng, Stefan Müller, Akitaka Matsuo, and William Lowe. 2021. *Quanteda: Quantitative Analysis of Textual Data*. https://quanteda.io.

Benoit, Kenneth, Kohei Watanabe, Haiyan Wang, Adam Obeng, Stefan Müller, and Akitaka Matsuo. 2021. *Quanteda.textplots: Plots for the Quantitative Analysis of Textual Data*. https://CRAN.R-project.org/package=quanteda.textplots.

Corporation, Microsoft, and Steve Weston. 2020. *DoParallel: Foreach Parallel Adaptor for the Parallel Package*. https://CRAN.R-project.org/package=doParallel.

Dowle, Matt, and Arun Srinivasan. 2021. *Data.table: Extension of 'Data.frame'*. https://CRAN.R-project.org/package=data.table.

R Core Team. 2021. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Revolution Analytics, and Steve Weston. n.d. *Foreach: Provides Foreach Looping Construct*.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

Wickham, Hadley, Winston Chang, Lionel Henry, Thomas Lin Pedersen, Kohske Takahashi, Claus Wilke, Kara Woo, Hiroaki Yutani, and Dewey Dunnington. 2020. *Ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. https://CRAN.R-project.org/package=ggplot2.

Xie, Yihui. 2014. "Knitr: A Comprehensive Tool for Reproducible Research in R." In *Implementing Reproducible Computational Research*, edited by Victoria Stodden, Friedrich Leisch, and Roger D. Peng. Chapman; Hall/CRC. http://www.crcpress.com/product/isbn/9781466561595.

———. 2015. *Dynamic Documents with R and Knitr*. 2nd ed. Boca Raton, Florida: Chapman; Hall/CRC. https://yihui.org/knitr/.

———. 2021. *Knitr: A General-Purpose Package for Dynamic Report Generation in R*. https://yihui.org/knitr/.

Xie, Yihui, J. J. Allaire, and Garrett Grolemund. 2018. *R Markdown: The Definitive Guide*. Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown.

Xie, Yihui, Christophe Dervieux, and Emily Riederer. 2020. *R Markdown Cookbook*. Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown-cookbook.

Zhu, Hao. 2021. *KableExtra: Construct Complex Table with Kable and Pipe Syntax*. https://CRAN.R-project.org/package=kableExtra.