Corona-Rechtsprechung des Bundesverfassungsgerichts

(BVerfG-Corona-Source)

Compilation Report

Version 2021-01-08

DOI: 10.5281/zenodo.4459416

Titel	Source Code der »Corona-Rechtsprechung des Bundesverfassungsgerichts«
Abkürzung	BVerfG-Corona-Source
Autor	Seán Fobbe
Version	2021-01-08
Download	https://doi.org/10.5281/zenodo.4459416
Lizenz	MIT No Attribution (MIT-0)

Zitiervorschlag

Seán Fobbe (2021). Source Code der »Corona-Rechtsprechung des Bundesverfassungsgerichts« (BVerfG-Corona-Source). Version 2021-01-08. Zenodo. DOI: 10.5281/zenodo.4459416.

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 2021-01-08. 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. 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 (CEBVerfG) 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. Der Corona-Datensatz im PDF-Format
2. Der Corona-Datensatz 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 2 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 DOI der konkreten Datensatz-Version

```
doi.version <- "10.5281/zenodo.4459406"
```

2.3 Verzeichnis für Analyse-Ergebnisse

Hinweis: Muss mit einem Schrägstrich enden!

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

2.4 Optionen: Quanteda

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

2.5 Optionen: Knitr

2.5.1 Ausgabe-Format

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

2.5.2 DPI für Raster-Grafiken

```
dpi <- 300
```

2.5.3 Ausrichtung von Grafiken im Compilation Report

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

2.6 Frequenztabellen: Ignorierte Variablen

Diese Variablen werden bei der Erstellung der Frequenztabellen nicht berücksichtigt.

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

3 Vorbereitung

3.1 Datumsstempel

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

```
datestamp <- "2021-01-08"
print(datestamp)
```

```
## [1] "2021-01-08"
```

3.2 Datum und Uhrzeit (Beginn)

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

```
## [1] "2021-01-24 15:20:41 CET"
```

3.3 Ordner für Analyse-Ergebnisse erstellen

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

3.4 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 1.13.6 using 8 threads (see ?getDTthreads).  Latest news: r-
    datatable.com
```

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

```
## Package version: 2.1.2
```

```
## Parallel computing: 2 of 16 threads used.
```

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

```
##
## Attaching package: 'quanteda'
```

```
## The following object is masked from 'package:utils':
##
##     View
```

3.5 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")
```

```
## Using poppler version 0.84.0
```

```
## Loading required package: xml2
```

3.6 Quanteda-Optionen setzen

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

3.7 Knitr Optionen setzen

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

3.8 Vollzitate statistischer Software

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

```
## tweaking foreach
```

3.9 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).

3.9.1 Logische Kerne

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

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

3.9.2 Quanteda

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

3.9.3 Data.table

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

3.9.4 DoParallel

```
cl <- makeForkCluster(fullCores)
registerDoParallel(cl)
```

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 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.

```
download.file("https://zenodo.org/record/4308215/files/CE-BVerfG_2021-01-08_DE_
    CSV_Datensatz.zip?download=1",
              "Stamm-Datensatz.zip")
```

4.2 ZIP-Archiv entpacken

```
unzip("Stamm-Datensatz.zip")
```

4.3 ZIP-Archiv löschen

```
unlink("Stamm-Datensatz.zip")
```

4.4 CSV-Datei einlesen

```
dt.bverfg <- fread("CE-BVerfG_2021-01-08_DE_CSV_Datensatz.csv")
```

4.5 Korpus-Objekt erstellen

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

5 Keywords in Context (KWIC)

Bei einer KWIC-Analyse (keywords in context) suchen wir nach einem bestimmten Muster und lassen uns die angrenzenden Wörter anzeigen. 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 erfassen. Das Sichtfenster wird auf 15 Tokens vor und nach dem Treffer gesetzt.

5.1 KWIC-Analyse durchführen

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

5.2 KWIC-Tabelle speichern

```
fwrite(data.frame(kwic),
       paste(datasetname,
             datestamp,
             "00_KeywordsInContext.csv",
             sep = "_"))


fwrite(data.frame(kwic),
       paste0(outputdir,
              datasetname,
              "_01_KeywordsInContext.csv"))
```

6 TXT-Datensatz erstellen

6.1 Namen der Corona-Entscheidungen definieren

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

6.2 TXT-Datensatz herunterladen

```
download.file("https://zenodo.org/record/4308215/files/CE-BVerfG_2021-01-08_DE_
    TXT_Datensatz.zip?download=1",
              "TXT.zip")
```

6.3 ZIP-Archiv entpacken

```
unzip("TXT.zip")
```

6.4 Corona-Entscheidungen verpacken

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

6.5 TXT-Dateien löschen

```
unlink(list.files(pattern = ".txt"))
```

6.6 ZIP-Archiv löschen

```
unlink("TXT.zip")
```

7 PDF-Datensatz erstellen

7.1 Namen der Corona-Entscheidungen definieren

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

7.2 PDF-Datensatz herunterladen

```
download.file("https://zenodo.org/record/4308215/files/CE-BVerfG_2021-01-08_DE_
    PDF_Datensatz.zip?download=1",
              "PDF.zip")
```

7.3 ZIP-Archiv entpacken

```
unzip("PDF.zip")
```

7.4 Corona-Entscheidungen verpacken

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

7.5 PDF-Dateien löschen

```
unlink(list.files(pattern = ".pdf"))
```

7.6 ZIP-Archiv löschen

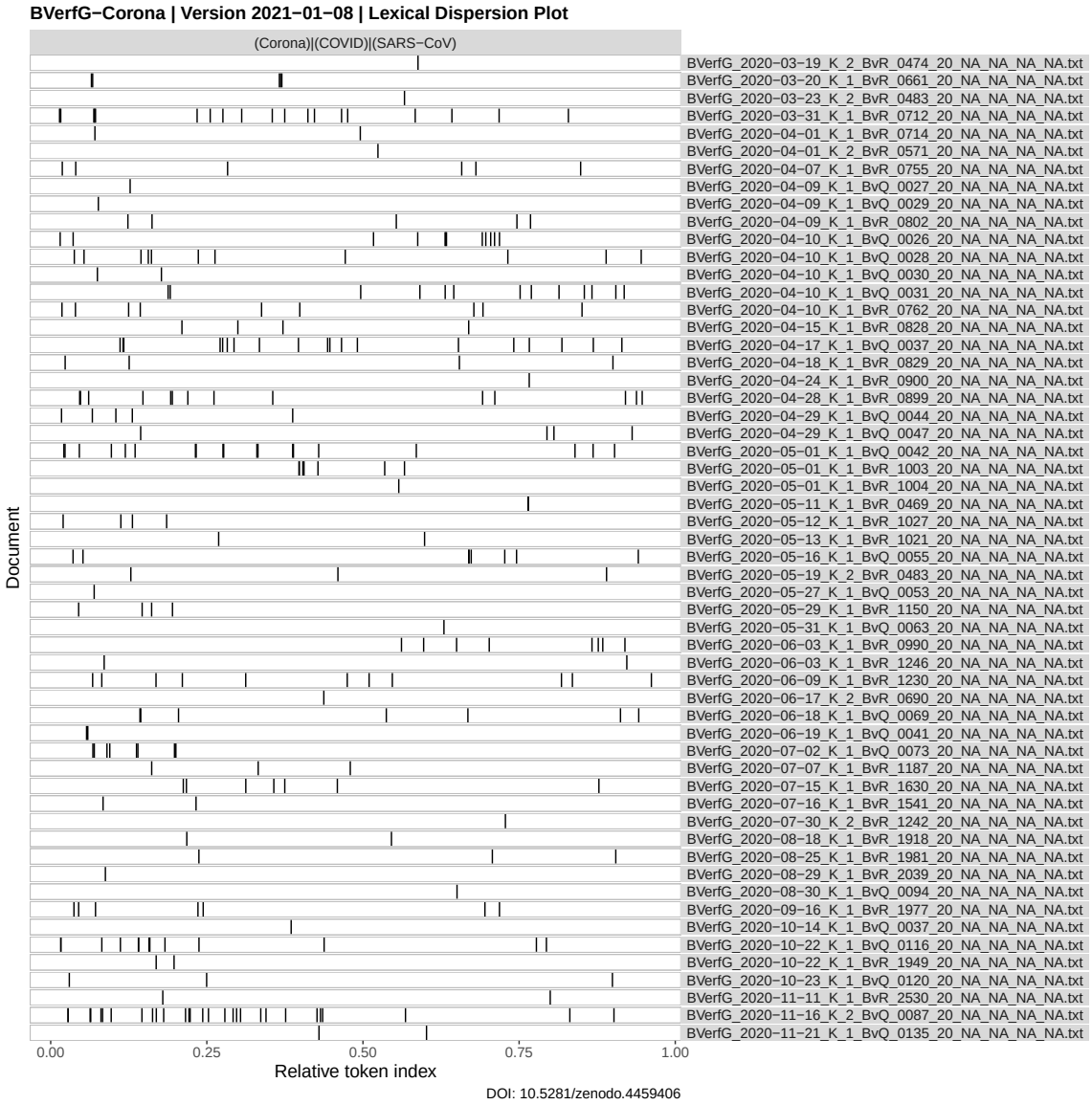
```
unlink("PDF.zip")
```

8 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.

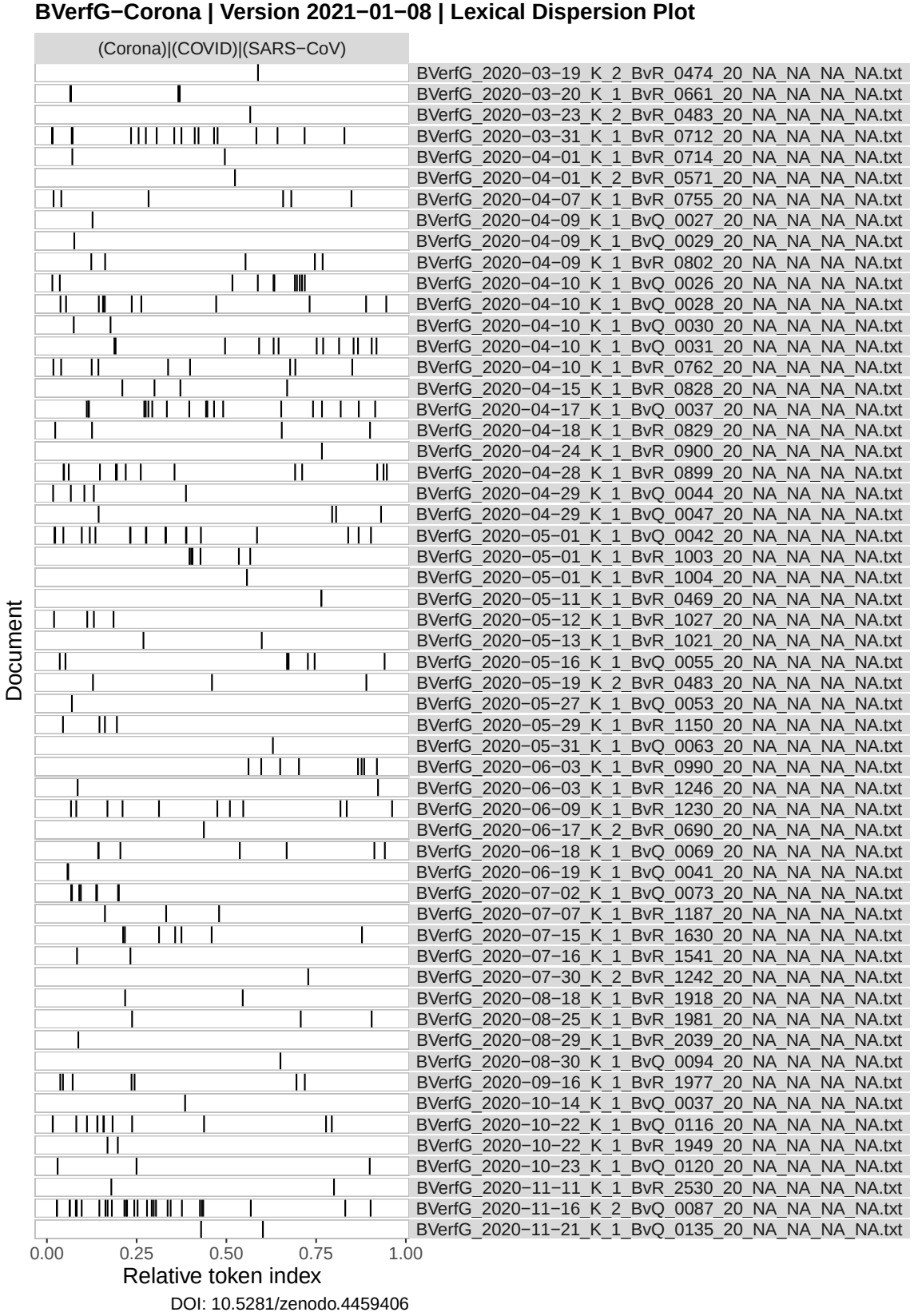
8.1 Quadratisches Format

```
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)
    )
```



8.2 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)
    )
```



DOI: 10.5281/zenodo.4459406

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 =){

```
## 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 = "r",
                    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] "tokens"         "typen"          "saetze"         "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"          "doi_concept"
## [13] "doi_version"
```

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)
```

Frequency Table for Variable: gericht

1 unique value(s) detected.

gericht	N	exactpercent	roundedpercent	cumulpercent
BVerfG	56	100	100	100
Total	56	100	100	100

Frequency Table for Variable: spruchkoerper_typ

1 unique value(s) detected.

spruchkoerper_typ	N	exactpercent	roundedpercent	cumulpercent
K	56	100	100	100
Total	56	100	100	100

Frequency Table for Variable: spruchkoerper_az

2 unique value(s) detected.

spruchkoerper_az	N	exactpercent	roundedpercent	cumulpercent
1	49	87.5	87.5	87.5
2	7	12.5	12.5	100.0
Total	56	100.0	100.0	100.0

Frequency Table for Variable: registerzeichen

2 unique value(s) detected.

registerzeichen	N	exactpercent	roundedpercent	cumulpercent
BvQ	22	39.28571	39.29	39.29
BvR	34	60.71429	60.71	100.00
Total	56	100.00000	100.00	100.00

Frequency Table for Variable: eingangsjahr_az

1 unique value(s) detected.

eingangsjahr_az	N	exactpercent	roundedpercent	cumulpercent
20	56	100	100	100
Total	56	100	100	100

Frequency Table for Variable: kollision

1 unique value(s) detected.

kollision	N	exactpercent	roundedpercent	cumulpercent
NA	56	100	100	100
Total	56	100	100	100

Frequency Table for Variable: band

1 unique value(s) detected.

band	N	exactpercent	roundedpercent	cumulpercent
NA	56	100	100	100
Total	56	100	100	100

Frequency Table for Variable: entscheidungsjahr

1 unique value(s) detected.

entscheidungsjahr	N	exactpercent	roundedpercent	cumulpercent
2020	56	100	100	100
Total	56	100	100	100

Frequency Table for Variable: eingangsjahr_iso

1 unique value(s) detected.

eingangsjahr_iso	N	exactpercent	roundedpercent	cumulpercent
2020	56	100	100	100
Total	56	100	100	100

Frequency Table for Variable: praesi

2 unique value(s) detected.

praesi	N	exactpercent	roundedpercent	cumulpercent
Harbarth	17	30.35714	30.36	30.36
Voßkuhle	39	69.64286	69.64	100.00
Total	56	100.00000	100.00	100.00

Frequency Table for Variable: v_praesi

2 unique value(s) detected.

v_praesi	N	exactpercent	roundedpercent	cumulpercent
Harbarth	39	69.64286	69.64	69.64
König	17	30.35714	30.36	100.00
Total	56	100.00000	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	56	100	100	100
Total	56	100	100	100

Frequency Table for Variable: doi_version

1 unique value(s) detected.

doi_version	N	exactpercent	roundedpercent	cumulpercent
10.5281/zenodo.4308215	56	100	100	100
Total	56	100	100	100

10 Erstellen der ZIP-Archive

10.1 Verpacken der Analyse-Dateien

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

10.2 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)
```

11 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.

11.1 Liste der ZIP-Archive erstellen

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

11.2 Funktion anzeigen

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

function(x){

```
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)
                       }
return(multihashes)
```

}

11.3 Hashes berechnen

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

11.4 In Data Table umwandeln

```
setDT(multihashes)
```

11.5 Index hinzufügen

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

11.6 In Datei schreiben

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

11.7 Leerzeichen hinzufügen um Zeilenumbruch zu ermöglichen

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

11.8 In Bericht anzeigen

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

index	filename
1	BVerfG-Corona_2021-01-08_DE_ANALYSE.zip
2	BVerfG-Corona_2021-01-08_DE_PDF_Datensatz.zip
3	BVerfG-Corona_2021-01-08_DE_TXT_Datensatz.zip
4	BVerfG-Corona_2021-01-08_Source_Files.zip

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

index	sha2.256
1	0fbe6c749396b03a53e66195e08be6d345ca99728bc06b5e2fd926775b2018b1
2	53ce11b228cc990ea513674e0051589105b72f43d01fe5acfe9d6c6d4b37c431
3	7a0379e90a8ae75fc868c8bc9701cd58e5d603e0460584cec3e069662fd1096d
4	35922d2c7eb06ecfd078ac32438baaf8414f09f13ce737ffc42037955641e710

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

index	sha3.512
1	0b2e7b3525a204a35c8fcee79d99d7fec585350593af23f3211d1928d4282076 889ad20170564430820028aa2ea7bb0afa7b95de4fb27a81168f5145ed6507b9
2	f7a2ceb7f75f197f0d23e46cd59fe8d4aa64161ade4a77a4ce330c42f387dc38 f80dffc73969e23289fe461fcb4acdadd9655cb479c54fc0a6dfe7742373115a
3	0c15629bb62a85da512621e35b9421825d4c51f55fca767283fdb1df6f8f0e99 6786fb4154684ad4fdd15c21975c7c75e0f87d0cb9afacc3e510e412b682ec12
4	0bbd6b73f0eafbbda3816deafff9613d413bc988733bfb614373d86f2441bec6 6964f9e518478fa671d556430a83bfa6de2871a3b5b3482fb492fa23527e9b09

12 Abschluss

12.1 Cluster stoppen

```
stopCluster(cl)
```

12.2 Datum und Uhrzeit (Ende)

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

```
## [1] "2021-01-24 15:23:50 CET"
```

12.3 Laufzeit des gesamten Skriptes

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

```
## Time difference of 3.150316 mins
```

12.4 Warnungen

```
warnings()
```

13 Parameter für strenge Replikationen

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

```
## [1] "OpenSSL 1.1.1i FIPS  8 Dec 2020"
```

```
sessionInfo()
```

```
## R version 4.0.3 (2020-10-10)
## Platform: x86_64-redhat-linux-gnu (64-bit)
## Running under: Fedora 32 (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] rvest_0.3.6       xml2_1.3.2        pdftools_2.3.1    quanteda_2.1.2
##  [5] data.table_1.13.6 kableExtra_1.3.1  knitr_1.30        rmarkdown_2.6
##  [9] ggplot2_3.3.3     doParallel_1.0.16 iterators_1.0.13  foreach_1.5.1
##
## loaded via a namespace (and not attached):
##  [1] qpdf_1.1           tidyselect_1.1.0   xfun_0.20          purrr_0.3.4
##  [5] lattice_0.20-41    colorspace_2.0-0   vctrs_0.3.6        generics_0.1.0
##  [9] htmltools_0.5.1    usethis_2.0.0      viridisLite_0.3.0  yaml_2.2.1
## [13] rlang_0.4.10       pillar_1.4.7       glue_1.4.2         withr_2.4.0
## [17] lifecycle_0.2.0    stringr_1.4.0      munsell_0.5.0      gtable_0.3.0
## [21] codetools_0.2-18   evaluate_0.14      labeling_0.4.2     highr_0.8
## [25] Rcpp_1.0.6         scales_1.1.1       RcppParallel_5.0.2 webshot_0.5.2
## [29] magick_2.6.0       farver_2.0.3       fs_1.5.0           fastmatch_1.1-0
## [33] stopwords_2.1      askpass_1.1        digest_0.6.27      stringi_1.5.3
## [37] dplyr_1.0.3        grid_4.0.3         tools_4.0.3        magrittr_2.0.1
## [41] tibble_3.0.5       crayon_1.3.4       pkgconfig_2.0.3    ellipsis_0.3.1
## [45] Matrix_1.3-2       httr_1.4.2         rstudioapi_0.13    R6_2.5.0
## [49] compiler_4.0.3
```

Literaturverzeichnis

Allaire, JJ, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone. 2020. *Rmarkdown: Dynamic Documents for R*. https://github.com/rstudio/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. 2018. "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, Jiong Wei Lua, Jouni Kuha, and William Lowe. 2020. *Quanteda: Quantitative Analysis of Textual Data*. https://quanteda.io.

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. 2020. *Data.table: Extension of 'Data.frame'*. https://CRAN.R-project.org/package=data.table.

Ooms, Jeroen. 2020. *Pdftools: Text Extraction, Rendering and Converting of Pdf Documents*. https://CRAN.R-project.org/package=pdftools.

R Core Team. 2020. *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.

———. 2020. *Rvest: Easily Harvest (Scrape) Web Pages*. https://CRAN.R-project.org/package=rvest.

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.

Wickham, Hadley, Jim Hester, and Jeroen Ooms. 2020. *Xml2: Parse Xml*. https://CRAN.R-project.org/package=xml2.

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/.

———. 2020. *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. 2020. *KableExtra: Construct Complex Table with Kable and Pipe Syntax*. https://CRAN.R-project.org/package=kableExtra.