Corpus der Entscheidungen des Bundespatentgerichts

(CE-BPatG-Source)

Compilation Report

Version 2023-04-02

DOI: 10.5281/zenodo.7767296

Titel	Source Code des »Corpus der Entscheidungen des Bundespatentgerichts«
Abkürzung	CE-BPatG-Source
Autor	Seán Fobbe
Version	2023-04-02
Download	https://doi.org/10.5281/zenodo.7767296
Lizenz	MIT No Attribution (MIT-0)

Zitiervorschlag

Seán Fobbe (2023). Source Code des »Corpus der Entscheidungen des Bundespatentgerichts« (CE-BPatG-Source). Version 2023-04-02. Zenodo. DOI: 10.5281/zenodo.7767296.

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 2023-04-02. 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 README: Corpus der Entscheidungen des Bundespatentgerichts (CE-BPatG)

1.1 Überblick

Das **Corpus der Entscheidungen des Bundespatentgerichts (CE-BPatG)** ist eine möglichst vollständige Sammlung der vom Bundespatentgericht veröffentlichten Entscheidungen. Der Datensatz nutzt als seine Datenquelle die Entscheidungsdatenbank des Bundespatentgerichts und wertet diese vollständig aus.

Alle mit diesem Skript erstellten Datensätze werden dauerhaft kostenlos und urheberrechtsfrei auf Zenodo, dem wissenschaftlichen Archiv des CERN, veröffentlicht. Alle Versionen sind mit einem separaten und langzeit-stabilen (persistenten) Digital Object Identifier (DOI) versehen.

Aktuellster, funktionaler und zitierfähiger Release des Datensatzes: https://doi.org/10.5281/zenodo.3954850

1.2 Funktionsweise

Primäre Endprodukte des Skripts sind folgende ZIP-Archive:

- Der volle Datensatz im CSV-Format
- Die reinen Metadaten im CSV-Format (wie unter 1, nur ohne Entscheidungstexte)
- Alle Entscheidungen im TXT-Format (reduzierter Umfang an Metadaten)
- Alle Entscheidungen im PDF-Format (reduzierter Umfang an Metadaten)
- Alle Analyse-Ergebnisse (Tabellen als CSV, Grafiken als PDF und PNG)
- Der Source Code und alle weiteren Quelldaten

Alle Ergebnisse werden im Ordner `output` abgelegt. Zusätzlich werden für alle ZIP-Archive kryptographische Signaturen (SHA2-256 und SHA3-512) berechnet und in einer CSV-Datei hinterlegt.

1.3 Systemanforderungen

- Docker
- Docker Compose
- 10 GB Speicherplatz auf Festplatte
- Multi-core CPU empfohlen (8 cores/16 threads für die Referenzdatensätze).

In der Standard-Einstellung wird das Skript vollautomatisch die maximale Anzahl an Rechenkernen/Threads auf dem System zu nutzen. Die Anzahl der verwendeten Kerne kann in der Konfigurationsatei angepasst werden. Wenn die Anzahl Threads auf 1 gesetzt wird, ist die Parallelisierung deaktiviert.

1.4 Anleitung

1.4.1 Schritt 1: Ordner vorbereiten

Kopieren Sie bitte den gesamten Source Code in einen leeren Ordner (!), beispielsweise mit:

```
$ git clone https://github.com/seanfobbe/ce-bpatg
```

Verwenden Sie immer einen separaten und *leeren* Ordner für die Kompilierung. Die Skripte löschen innerhalb von bestimmten Unterordnern (`files/`, `temp/`, `analysis` und `output/`) alle Dateien die den Datensatz verunreinigen könnten — aber auch nur dort.

1.4.2 Schritt 2: Docker Image erstellen

Ein Docker Image stellt ein komplettes Betriebssystem mit der gesamten verwendeten Software automatisch zusammen. Nutzen Sie zur Erstellung des Images einfach:

```
$ bash docker-build-image.sh
```

1.4.3 Schritt 3: Datensatz kompilieren

Falls Sie zuvor den Datensatz schon einmal kompiliert haben (ob erfolgreich oder erfolglos), können Sie mit folgendem Befehl alle Arbeitsdaten im Ordner löschen:

```
$ Rscript delete_all_data.R
```

Den vollständigen Datensatz kompilieren Sie mit folgendem Skript:

```
$ bash docker-run-project.sh
```

1.4.4 Ergebnis

Der Datensatz und alle weiteren Ergebnisse sind nun im Ordner `output/` abgelegt.

1.5 Pipeline visualisieren

Sie können die Pipeline visualisieren, aber nur nachdem sie die zentrale .Rmd-Datei mindestens einmal gerendert haben:

```
> targets::tar_glimpse()     # Nur Datenobjekte
> targets::tar_visnetwork()  # Alle Objekte
```

1.6 Troubleshooting

Hilfreiche Befehle um Fehler zu lokalisieren und zu beheben.

```
> tar_progress()  # Zeigt Fortschritt und Fehler an
> tar_meta()      # Alle Metadaten
> tar_meta(fields = "warnings", complete_only = TRUE)  # Warnungen
> tar_meta(fields = "error", complete_only = TRUE)  # Fehlermeldungen
> tar_meta(fields = "seconds")  # Laufzeit der Targets
```

1.7 Projektstruktur

Die folgende Struktur erläutert die wichtigsten Bestandteile des Projekts. Während der Kompilierung werden weitere Ordner erstellt (`files/`, `temp/` `analysis` und `output/`). Die Endergebnisse werden alle in `output/` abgelegt.

```
.
├ buttons                    # Buttons (nur optische Bedeutung)
├ CHANGELOG.md               # Alle Änderungen
├ compose.yaml               # Konfiguration für Docker
├ config.toml                # Zentrale Konfigurations-Datei
├ data                       # Datensätze, auf denen die Pipeline aufbaut
├ delete_all_data.R          # Löscht den Datensatz und Zwischenschritte
├ docker-build-image.sh      # Docker Image erstellen
├ Dockerfile                 # Definition des Docker Images
├ docker-run-project.sh      # Docker Image und Datensatz kompilieren
├ functions                  # Wichtige Schritte der Pipeline
├ gpg                        # Persönlicher Public GPG-Key für Seán Fobbe
├ old                        # Alter Code aus früheren Versionen
├ pipeline.Rmd               # Zentrale Definition der Pipeline
├ README.md                  # Bedienungsanleitung
├ reports                    # Markdown-Dateien
├ requirements-python.txt    # Benötigte Python packages
├ requirements-R.R           # Benötigte R packages
├ requirements-system.txt    # Benötigte system dependencies
├ run_project.R              # Kompiliert den gesamten Datensatz
└ tex                       # LaTeX-Templates
```

1.8 Weitere Open Access Veröffentlichungen (Fobbe)

Website — https://www.seanfobbe.de

Open Data — https://zenodo.org/communities/sean-fobbe-data/

Source Code — https://zenodo.org/communities/sean-fobbe-code/

Volltexte regulärer Publikationen — https://zenodo.org/communities/sean-fobbe-publications/

1.9 Kontakt

Fehler gefunden? Anregungen? Kommentieren Sie gerne im Issue Tracker auf GitHub oder schreiben Sie mir eine E-Mail an fobbe-data@posteo.de

2 Packages laden

```
library(targets)
library(tarchetypes)
library(RcppTOML)
library(future)
library(data.table)
library(quanteda)
#> Package version: 3.2.4
#> Unicode version: 14.0
#> ICU version: 70.1
#> Parallel computing: 16 of 16 threads used.
#> See https://quanteda.io for tutorials and examples.
library(knitr)
library(kableExtra)
library(igraph)
#>
#> Attaching package: 'igraph'
#> The following objects are masked from 'package:future':
#>
#>     %->%, %<-%
#> The following objects are masked from 'package:stats':
#>
#>     decompose, spectrum
#> The following object is masked from 'package:base':
#>
#>     union
library(ggraph)
#> Loading required package: ggplot2

tar_unscript()
```

3 Vorbereitung

3.1 Definitionen

```
## Datum
datestamp <- Sys.Date()
print(datestamp)
#> [1] "2023-04-02"

## Datum und Uhrzeit (Beginn)
begin.script <- Sys.time()

## Konfiguration
config <- RcppTOML::parseTOML("config.toml")
print(config)
#> List of 9
#>  $ cores   :List of 2
#>   ..$ max   : logi TRUE
#>   ..$ number: int 8
#>  $ debug   :List of 3
#>   ..$ cleanrun: logi FALSE
#>   ..$ pages   : int 50
#>   ..$ toggle  : logi FALSE
#>  $ doi     :List of 4
#>   ..$ aktenzeichen : chr "10.5281/zenodo.4569564"
#>   ..$ data         :List of 2
#>   .. ..$ concept: chr "10.5281/zenodo.3954850"
#>   .. ..$ version: chr "10.5281/zenodo.7767295"
#>   ..$ personendaten: chr "10.5281/zenodo.4568682"
#>   ..$ software     :List of 2
#>   .. ..$ concept: chr "10.5281/zenodo.6667305"
#>   .. ..$ version: chr "10.5281/zenodo.7767296"
#>  $ download:List of 1
#>   ..$ timeout: int 600
#>  $ fig     :List of 3
#>   ..$ align : chr "center"
#>   ..$ dpi   : int 300
#>   ..$ format: chr [1:2] "pdf" "png"
#>  $ license :List of 2
#>   ..$ code: chr "MIT-0"
#>   ..$ data: chr "Creative Commons Zero 1.0 Universal"
#>  $ parallel:List of 3
#>   ..$ extractPDF   : logi TRUE
#>   ..$ lingsummarize: logi TRUE
#>   ..$ multihashes  : logi TRUE
#>  $ project :List of 3
#>   ..$ author   : chr "Seán Fobbe"
#>   ..$ fullname : chr "Corpus der Entscheidungen des Bundespatentgerichts"
#>   ..$ shortname: chr "CE-BPatG"
#>  $ quanteda:List of 1
#>   ..$ tokens_locale: chr "de_DE"


# Analyse-Ordner
```

```
dir.analysis <- paste0(getwd(),
                       "/analysis")
```

3.2 Aufräumen

Löscht Dateien im Output-Ordner, die nicht vom heutigen Tag sind.

```
unlink(grep(datestamp,
            list.files("output",
                       full.names = TRUE),
            invert = TRUE,
            value = TRUE))
```

3.3 Ordner erstellen

```
#unlink("output", recursive = TRUE)
dir.create("output", showWarnings = FALSE)
dir.create("temp", showWarnings = FALSE)

dir.create(dir.analysis, showWarnings = FALSE)
```

3.4 Vollzitate statistischer Software schreiben

```
knitr::write_bib(renv::dependencies()$Package,
                 "temp/packages.bib")
#> Finding R package dependencies ... Done!
```

4 Globale Variablen

4.1 Packages definieren

```
tar_option_set(packages = c("tarchetypes",
                            "RcppTOML",     # TOML-Dateien lesen und schreiben
                            "fs",           # Verbessertes File Handling
                            "zip",          # Verbessertes ZIP Handling
                            "mgsub",        # Vektorisiertes Gsub
                            "httr",         # HTTP-Werkzeuge
                            "rvest",        # HTML/XML-Extraktion
                            "testthat",     # Unit Tests
                            "knitr",        # Professionelles Reporting
                            "kableExtra",   # Verbesserte Kable Tabellen
                            "pdftools",     # Verarbeitung von PDF-Dateien
                            "ggplot2",      # Fortgeschrittene
    Datenvisualisierung
                            "ggraph",       # Visualisierung von Graphen
                            "scales",       # Skalierung von Diagrammen
                            "data.table",   # Fortgeschrittene Datenverarbeitung
                            "readtext",     # TXT-Dateien einlesen
                            "quanteda",     # Fortgeschrittene Computerlinguistik
                            "future",       # Parallelisierung
                            "future.apply"))# Funktionen höherer Ordnung für
    Parallelisierung

tar_option_set(workspace_on_error = TRUE) # Save Workspace on Error
tar_option_set(format = "qs")

#> Establish _targets.R and _targets_r/globals/global-packages.R.
```

4.2 Konfiguration

```
datestamp <- Sys.Date()

config <- RcppTOML::parseTOML("config.toml")

dir.analysis <- paste0(getwd(),
                       "/analysis")

## Caption for diagrams
caption <- paste("Fobbe | DOI:",
                 config$doi$data$version)


## Prefix for figure titles
prefix.figuretitle <- paste(config$project$shortname,
                            "| Version",
                            datestamp)

## File prefix
```

```
prefix.files <- paste0(config$project$shortname,
                       "_",
                       datestamp)


if (config$cores$max == TRUE){
    fullCores <- future::availableCores()
}


if (config$cores$max == FALSE){
    fullCores <- as.integer(config$cores$number)
}

#> Establish _targets.R and _targets_r/globals/global-config.R.
```

4.3 Funktionen definieren

```
lapply(list.files("functions", pattern = "\\.R$", full.names = TRUE), source)

#> Establish _targets.R and _targets_r/globals/global-functions.R.
```

4.4 Metadaten für TXT-Dateien definieren

```
docvarnames <- c("gericht",
                 "senatsgruppe",
                 "leitsatz",
                 "datum",
                 "spruchkoerper_az",
                 "registerzeichen",
                 "eingangsnummer",
                 "eingangsjahr_az",
                 "zusatz_az",
                 "kollision")

#> Establish _targets.R and _targets_r/globals/global-txtvars.R.
```

4.5 ZIP-Datei für Source definieren

```
files.source.raw <-  c(system2("git", "ls-files", stdout = TRUE),
                       ".git")

#> Establish _targets.R and _targets_r/globals/global-sourcefiles.R.
```

5 Pipeline: Konstruktion

5.1 File Tracking Targets

Mit diesem Abschnitt der Pipeline werden Input-Dateien getrackt. Mit der Option »format = "file"« werden für Input-Dateien Prüfsummen berechnet. Falls sich diese verändern werden alle von ihnen abhängigen Pipeline-Schritte als veraltet markiert und neu berechnet.

5.1.1 Variablen

Die Variablen des Datensatzes, inklusive ihrer Erläuterung.

```
list(
    tar_target(file.variables.codebook,
               "data/CE-BPatG_Variables.csv",
               format = "file"),
    tar_target(variables.codebook,
               fread(file.variables.codebook))
)
#> Establish _targets.R and _targets_r/targets/tar.filevars.R.
```

5.1.2 Aktenzeichen der Bundesrepublik Deutschland (AZ-BRD)

Die Tabelle der Registerzeichen und der ihnen zugeordneten Verfahrensarten stammt aus dem folgenden Datensatz: »Seán Fobbe (2021). Aktenzeichen der Bundesrepublik Deutschland (AZ-BRD). Version 1.0.1. Zenodo. DOI: 10.5281/zenodo.4569564.«

```
list(
    tar_target(file.az.brd,
           "data/AZ-BRD_1-0-1_DE_Registerzeichen_Datensatz.csv",
           format = "file"),
     tar_target(az.brd,
                fread(file.az.brd))
     )
#> Establish _targets.R and _targets_r/targets/tar.file.azbrd.R.
```

5.1.3 Source Code

Dies sind alle Dateien, die den Source Code für den Datensatz bereitstellen.

```
tar_target(files.source,
           files.source.raw,
           format = "file")

#> Establish _targets.R and _targets_r/targets/tar.file.source.R.
```

5.1.4 Changelog

```
tar_target(changelog,
           "CHANGELOG.md",
           format = "file")
#> Establish _targets.R and _targets_r/targets/tar.file.changelog.R.
```

5.2 Download Targets

In diesem Abschnitt der Pipeline wird die Datenbank des BPatG abgerufen, die Links zu Volltexten und Metadaten zusammengeführt und schließlich abgerufen.

5.2.1 Maximalen Umfang der Datenbankabfrage bestimmen

```
tarchetypes::tar_age(dt.scope,
                     f.scope(debug.toggle = config$debug$toggle,
                             debug.pages = config$debug$pages),
                     age = as.difftime(1, units = "days"))

#> Establish _targets.R and _targets_r/targets/tar.download.scope.R.
```

5.2.2 Datenbankseiten als HTML abrufen

```
tar_target(files.html,
                f.download(url = dt.scope$url,
                           filename = dt.scope$filename,
                           dir = "files/html",
                           sleep.min = 0.2,
                           sleep.max = 0.5,
                           retries = 3,
                           retry.sleep.min = 2,
                           retry.sleep.max = 5,
                           timeout = config$download$timeout,
                           debug.toggle = FALSE,
                           debug.files = 500),
                format = "file")

#> Establish _targets.R and _targets_r/targets/tar.download.html.R.
```

5.2.3 Vorläufige Download-Tabelle erstellen

Abfrage der Datenbank des BPatG und erstellen der vorläufigen Download-Tabelle.

```
tar_target(dt.download,
           f.download_table_make(files.html = files.html))

#> Establish _targets.R and _targets_r/targets/tar.download.make.R.
```

5.2.4 Download-Tabelle bereinigen

Die Aktenzeichen und Senatsgruppen werden bereinigt und gespeichert.

```
list(
    tar_target(az_clean,
               f.clean_az_bpatg(dt.download$az)),
    tar_target(senatsgruppe_clean,
              f.clean_senatsgruppe_bpatg(dt.download$senatsgruppe))
    )


#> Establish _targets.R and _targets_r/targets/tar.download.clean.R.
```

5.2.5 Finale Download-Tabelle erstellen

Die bereinigten Aktenzeichen und Senatsgruppen werden hier mit der vorläufigen Download-Tabelle zusammengeführt, weitere Bereinigungen durchgeführt, weitere Variablen aus den Bemerkungen extrahiert und schließlich auf ihre Konformität mit der Beschreibung im Codebook getestet.

```
tar_target(dt.download.final,
               f.download_table_finalize(x = dt.download,
                                         az = az_clean,
                                         senatsgruppe = senatsgruppe_clean))
#> Establish _targets.R and _targets_r/targets/tar.download.final.R.
```

5.2.6 Download durchführen

```
tar_target(files.pdf,
               f.download(url = dt.download.final$url,
                          filename = dt.download.final$doc_id,
                          dir = "files/pdf",
                          sleep.min = 0,
                          sleep.max = 0.1,
                          retries = 3,
                          retry.sleep.min = 2,
                          retry.sleep.max = 5,
                          timeout = config$download$timeout,
                          debug.toggle = FALSE,
                          debug.files = 500),
               format = "file")

#> Establish _targets.R and _targets_r/targets/tar.download.pdf.R.
```

5.3 Convert Targets

Durch diesen Abschnitt der Pipeline werden die PDF-Dateien in TXT konvertiert und mitsamt den Variablen in ihren Dateinamen eingelesen. Beim Einlesen werden die in PDF-Dateien üblichen über Zeilen gebrochene Wörter wieder zusammengefügt.

```
list(tar_target(files.txt,
                f.tar_pdf_extract(files.pdf,
                                  outputdir = "files/txt",
                                  cores = fullCores),
                format = "file"),
     tar_target(dt.bpatg,
                f.readtext(x = files.txt,
                           docvarnames = docvarnames))
     )

#> Establish _targets.R and _targets_r/targets/tar.convert.R.
```

5.4 Enhance Targets

Dieser Abschnitt der Pipeline berechnet diverse Verbesserungen für den Datensatz und führt diese am Ende zusammen.

5.4.1 Daten standardisieren

Das Datum wird im ISO-Format standardisiert und die Variablen »entscheidungsjahr« und »eingangsjahr_iso« hinzugefügt.

```
tar_target(dt.bpatg.datecleaned,
                f.clean_dates(dt.bpatg))
#> Establish _targets.R and _targets_r/targets/tar.enhance.dateclean.R.
```

5.4.2 Variable erstellen: »verfahrensart«

Die Variable »verfahrensart« wird aus den Registerzeichen berechnet.

```
tar_target(var_verfahrensart,
                f.var_verfahrensart(dt.bpatg.datecleaned$registerzeichen,
                                    az.brd = az.brd,
                                    gericht = "BPatG"))
#> Establish _targets.R and _targets_r/targets/tar.enhance.verfahrensart.R.
```

5.4.3 Variable erstellen: »aktenzeichen«

Das Aktenzeichen wird aus seinen Komponenten berechnet.

```
tar_target(var_aktenzeichen,
                f.var_aktenzeichen(dt.bpatg.datecleaned,
                                   az.brd = az.brd,
                                   gericht = "BPatG"))
#> Establish _targets.R and _targets_r/targets/tar.enhance.az.R.
```

5.4.4 Variable erstellen: »entscheidung_typ«

Der Typ der Entscheidung wird aus dem Text extrahiert.

```
tar_target(var_entscheidung_typ,
                f.var_entscheidung_typ(dt.bpatg.datecleaned$text))
#> Establish _targets.R and _targets_r/targets/tar.enhance.entschtyp.R.
```

5.4.5 Variable erstellen: »ecli«

Die ECLI wird aus ihren Komponenten berechnet. Achtung: weil die offizielle Kollisionsnummer nicht in der Datenbank dokumentiert ist, können Entscheidungen vom gleichen Tag mit dem gleichen Aktenzeichen potentiell fehlerhafte ECLIs aufweisen.

```
tar_target(var_ecli,
                f.var_ecli_bpatg(dt.bpatg.datecleaned,
                                 entscheidung_typ = var_entscheidung_typ))
#> Establish _targets.R and _targets_r/targets/tar.enhance.ecli.R.
```

5.4.6 Variablen erstellen: »zeichen, token, typen, saetze«

Berechnung klassischer linguistischer Kennzahlen.

```
tar_target(var_lingstats,
                f.lingstats(dt.bpatg.datecleaned,
                            multicore = config$parallel$lingsummarize,
                            cores = fullCores,
                            germanvars = TRUE))
#> Establish _targets.R and _targets_r/targets/tar.enhance.lingstats.R.
```

5.4.7 Konstanten erstellen

Konstanten die dem Datensatz wichtige Herkunftsinformationen hinzufügen. Darunter sind die Versionsnummer, die Version DOI, die Concept DOI und die Lizenz.

```
tar_target(var_constants,
           data.frame(version = as.character(datestamp),
                      doi_concept = config$doi$data$concept,
                      doi_version = config$doi$data$version,
                      lizenz = as.character(config$license$data))[rep(1,
                                                                      nrow(dt.
    bpatg.datecleaned)),])
#> Establish _targets.R and _targets_r/targets/tar.enhance.constants.R.
```

5.4.8 Zusätzliche Variablen zusammenführen

```
tar_target(vars_additional,
           data.table(verfahrensart = var_verfahrensart,
                      aktenzeichen = var_aktenzeichen,
                      ecli = var_ecli,
                      entscheidung_typ = var_entscheidung_typ,
                      var_lingstats,
                      var_constants))

#> Establish _targets.R and _targets_r/targets/tar.enhance.unify.R.
```

5.4.9 Finalen Datensatz erstellen

Die Verbesserungen der vorherigen Schritte werden in dieser Funktion zusammengefügt um den finalen Datenatz herzustellen.

```
tar_target(dt.bpatg.final,
           f.dataset_finalize(x = dt.bpatg.datecleaned,
                              download.table = dt.download.final,
                              vars.additional = vars_additional,
                              varnames = variables.codebook$varname))

#> Establish _targets.R and _targets_r/targets/tar.enhance.final.R.
```

5.4.10 Variante erstellen: Nur Metadaten

Hier wird nur die Text-Variable entfernt um eine deutlich platzsparendere Variante des Datensatzes nur mit den Metadaten zu erstellen.

```
tar_target(dt.bpatg.meta,
                dt.bpatg.final[, !"text"])

#> Establish _targets.R and _targets_r/targets/tar.enhance.meta.R.
```

5.5 Write Targets

Dieser Abschnitt der Pipeline schreibt den Datensatz und die jeweiligen Prüfsummen auf die Festplatte.

5.5.1 CSV schreiben: Voller Datensatz

```
tar_target(csv.full,
           f.tar_fwrite(x = dt.bpatg.final,
                        filename = file.path("output",
                                             paste0(prefix.files,
                                                    "_DE_CSV_Datensatz.csv"))
```

```
                        )
            )
#> Establish _targets.R and _targets_r/targets/tar.write.full.R.
```

5.5.2 CSV schreiben: Metadaten

```
tar_target(csv.meta,
           f.tar_fwrite(x = dt.bpatg.meta,
                        filename = file.path("output",
                                             paste0(prefix.files,
                                                    "_DE_CSV_Metadaten.csv"))
                        )
           )
#> Establish _targets.R and _targets_r/targets/tar.write.meta.R.
```

5.5.3 CSV schreiben: Kryptographische Hashes

```
tar_target(csv.hashes,
           f.tar_fwrite(x = hashes,
                        filename = file.path("output",
                                             paste0(prefix.files,
                                                    "_KryptographischeHashes.csv"
    ))
                        )
           )
#> Establish _targets.R and _targets_r/targets/tar.write.hashes.R.
```

5.6 ZIP Targets

Diese Abschnitt der Pipeline erstellt ZIP-Archive für alle zentralen Rechenergebnisse und speichert diese im Ordner »output«.

5.6.1 ZIP erstellen: PDF-Dateien (alle Entscheidungen)

```
tar_target(zip.pdf.all,
                f.tar_zip(files.pdf,
                          filename = paste(prefix.files,
                                           "DE_PDF_Datensatz.zip",
                                           sep = "_"),
                          dir = "output",
                          mode = "cherry-pick"),
                format = "file")
#> Establish _targets.R and _targets_r/targets/tar.zip.pdf.all.R.
```

5.6.2 ZIP erstellen: PDF-Dateien (nur Leitsatzentscheidungen)

```
tar_target(zip.pdf.leitsatz,
                f.tar_zip(grep("_LE_", files.pdf, value = TRUE),
                                filename = paste(prefix.files,
                                                 "DE_PDF_Leitsatz-Entscheidungen.
    zip",
                                                 sep = "_"),
                                dir = "output",
                                mode = "cherry-pick"),
                format = "file")
#> Establish _targets.R and _targets_r/targets/tar.zip.pdf.leitsatz.R.
```

5.6.3 ZIP erstellen: TXT-Dateien

```
tar_target(zip.txt,
                f.tar_zip(files.txt,
                                filename = paste(prefix.files,
                                                 "DE_TXT_Datensatz.zip",
                                                 sep = "_"),
                                dir = "output",
                                mode = "cherry-pick"),
                format = "file")
#> Establish _targets.R and _targets_r/targets/tar.zip.txt.R.
```

5.6.4 ZIP erstellen: Analyse-Dateien

```
tar_target(zip.analysis,
           f.tar_zip("analysis/",
                           filename = paste(prefix.files,
                                            "DE_Analyse.zip",
                                            sep = "_"),
                           dir = "output",
                           mode = "cherry-pick",
                           report.codebook,
                           report.robustness),
           format = "file")
#> Establish _targets.R and _targets_r/targets/tar.zip.analysis.R.
```

5.6.5 ZIP erstellen: CSV-Datei (voller Datensatz)

```
tar_target(zip.csv.full,
                f.tar_zip(csv.full,
                                filename = gsub("\\.csv", "\\.zip", basename(csv.
    full)),
                                dir = "output",
                                mode = "cherry-pick"),
                format = "file")
#> Establish _targets.R and _targets_r/targets/tar.zip.csv.full.R.
```

5.6.6 ZIP erstellen: CSV-Datei (nur Metadaten)

```
tar_target(zip.csv.meta,
                f.tar_zip(csv.meta,
                              filename = gsub("\\.csv", "\\.zip", basename(csv.
    meta)),
                              dir = "output",
                              mode = "cherry-pick"),
                format = "file")
#> Establish _targets.R and _targets_r/targets/tar.zip.csv.meta.R.
```

5.6.7 ZIP erstellen: Source Code

```
tar_target(zip.source,
                f.tar_zip(files.source,
                              filename = paste0(prefix.files,
                                                "_Source_Code.zip"),
                              dir = "output",
                              mode = "mirror"),
                format = "file")
#> Establish _targets.R and _targets_r/targets/tar.zip.source.R.
```

5.6.8 Kryptographische Hashes für alle ZIP-Archive berechnen

```
tar_target(zip.all,
                c(zip.pdf.all,
                  zip.pdf.leitsatz,
                  zip.txt,
                  zip.csv.full,
                  zip.csv.meta,
                  zip.analysis,
                  zip.source))
#> Establish _targets.R and _targets_r/targets/tar.zip.all.R.
```

```
tar_target(hashes,
           f.tar_multihashes(c(zip.all,
                               report.codebook[1],
                               report.robustness[1]),
                             multicore = config$parallel$multihashes,
                             cores = fullCores))
#> Establish _targets.R and _targets_r/targets/tar.zip.hashes.R.
```

5.7 Report Targets

Dieser Abschnitt der Pipeline erstellt die finalen Berichte (Codebook und Robustness Checks).

5.7.1 LaTeX-Definitionen schreiben

Um gewisse Variablen aus der Pipeline in die LaTeX-Kompilierung einzuführen müssen diese als .tex-Datei auf die Festplatte geschrieben werden.

```
tar_target(latexdefs,
                f.latexdefs(config,
                            dir = "temp",
                            version = datestamp),
           format = "file")

#> Establish _targets.R and _targets_r/targets/tar.report.latexdefs.R.
```

5.7.2 Zusammenfassungen linguistischer Kennwerte berechnen

```
tar_target(lingstats.summary,
                f.lingstats_summary(dt.bpatg.final,
                                    germanvars = TRUE))

#> Establish _targets.R and _targets_r/targets/tar.report.lingstats.R.
```

5.7.3 Report erstellen: Robustness Checks

```
tarchetypes::tar_render(report.robustness,
                        file.path("reports",
                                  "RobustnessChecks.Rmd"),
                        output_file = file.path("../output",
                                                paste0("CE-BPatG_", datestamp, "_
    RobustnessChecks.pdf")))

#> Establish _targets.R and _targets_r/targets/tar.report.robustness.R.
```

5.7.4 Report erstellen: Codebook

```
tarchetypes::tar_render(report.codebook,
                        file.path("reports",
                                  "Codebook.Rmd"),
                        output_file = file.path("../output",
                                                paste0("CE-BPatG_", datestamp, "_
    Codebook.pdf")))

#> Establish _targets.R and _targets_r/targets/tar.report.codebook.R.
```

6 Pipeline: Kompilierung

6.1 Durchführen der Kompilierung

```
tar_make()
```

6.2 Zwischenergebnisse archivieren

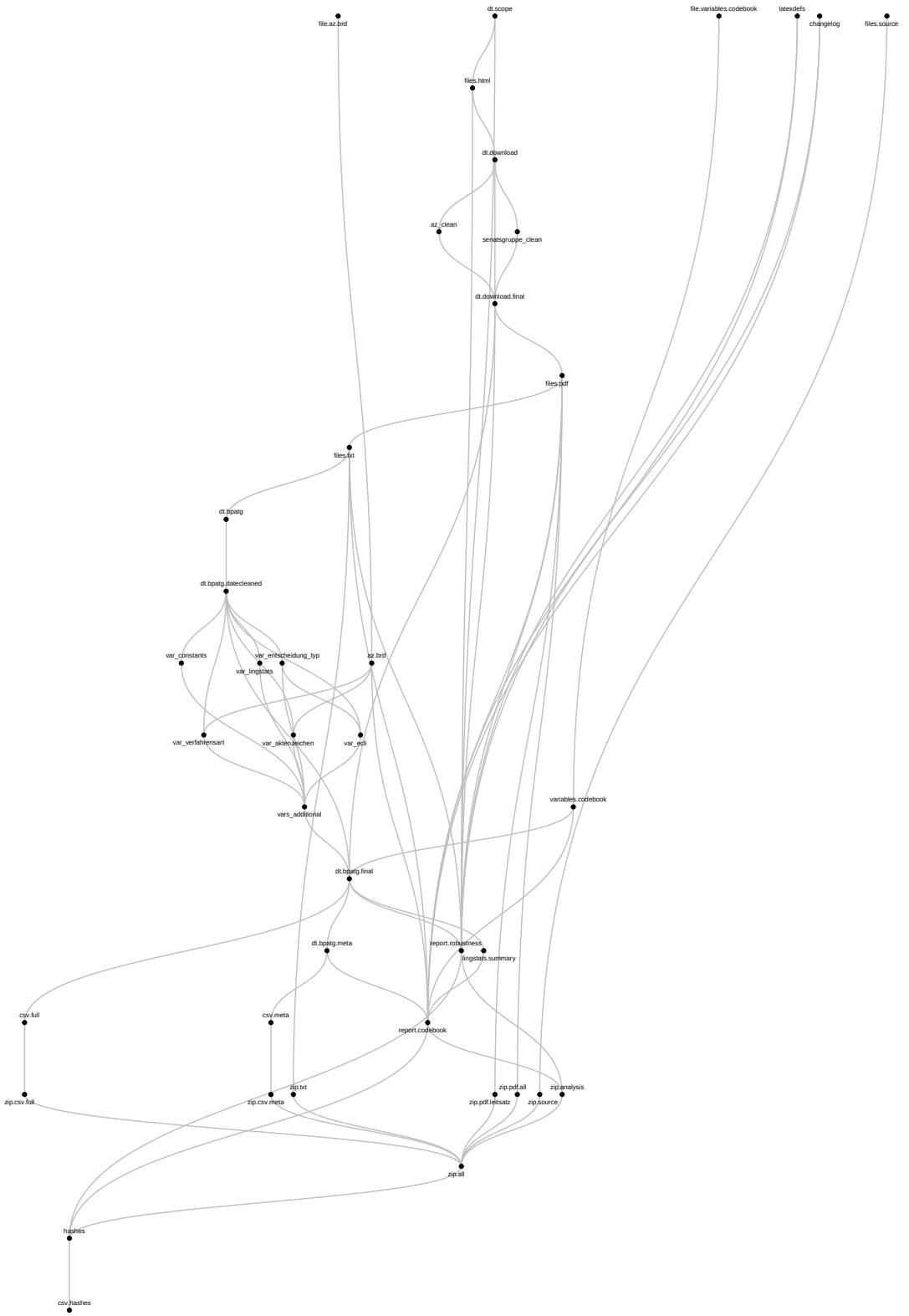
```
zip(paste0("output/CE-BPatG_",
           datestamp,
           "_Targets_Storage.zip"),
    "_targets/")
```

6.3 Visualisierung

```
edgelist <- tar_network(targets_only = TRUE)$edges
setDT(edgelist)

g  <- igraph::graph.data.frame(edgelist,
                               directed = TRUE)


ggraph(g,
       'sugiyama') +
    geom_edge_diagonal(colour = "grey")+
    geom_node_point()+
    geom_node_text(aes(label = name),
                   size = 2,
                   repel = TRUE)+
    theme_void()
#> Warning: Using the `size` aesthetic in this geom was deprecated in ggplot2
    3.4.0.
#> i Please use `linewidth` in the `default_aes` field and elsewhere instead.
#> This warning is displayed once every 8 hours.
#> Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
#> generated.
```


dt.scope
file.az.brd
file.variables.codebook
latexdefs
changelog
files.source
files.html
dt.download
az_clean
senatsgruppe_clean
dt.download.final
files.pdf
files.txt
dt.bpatg
dt.bpatg.datecleaned
var_constants
var_entscheidung_typ
az.brd
var_lingstats
var_verfahrensart
var_aktenzeichen
var_ecli
vars_additional
variables.codebook
dt.bpatg.final
dt.bpatg.meta
report.robustness
lingstats.summary
csv.full
csv.meta
report.codebook
zip.txt
zip.pdf.all
zip.analysis
zip.csv.full
zip.csv.meta
zip.pdf.leitsatz
zip.source
zip.all
hashes
csv.hashes

7 Pipeline: Profiling

7.1 Gesamte Liste

Die vollständige Liste aller Targets, inklusive ihres Types und ihrer Größe. Targets die auf Dateien verweisen (z.B. alle PDF-Dateien) geben die Gesamtgröße der Dateien auf der Festplatte an.

```
meta <- tar_meta(fields = c("type", "bytes", "format"), complete_only = TRUE)
setDT(meta)
meta$MB <- round(meta$bytes / 1e6, digits = 2)

# Gesamter Speicherplatzverbrauch
sum(meta$MB, na.rm = TRUE)
#> [1] 8102.65

kable(meta[order(type, name)],
      format = "latex",
      align = "r",
      booktabs = TRUE,
      longtable = TRUE) %>% kable_styling(latex_options = "repeat_header")
```

name	type	bytes	format	MB
az.brd	stem	5509	qs	0.01
az_clean	stem	105825	qs	0.11
changelog	stem	1583	file	0.00
csv.full	stem	87	qs	0.00
csv.hashes	stem	93	qs	0.00
csv.meta	stem	87	qs	0.00
dt.bpatg	stem	142644543	qs	142.64
dt.bpatg.datecleaned	stem	147605391	qs	147.61
dt.bpatg.final	stem	143353449	qs	143.35
dt.bpatg.meta	stem	1033224	qs	1.03
dt.download	stem	421197	qs	0.42
dt.download.final	stem	618599	qs	0.62
dt.scope	stem	5012	qs	0.01
file.az.brd	stem	36533	file	0.04
file.variables.codebook	stem	7113	file	0.01
files.html	stem	54870943	file	54.87

(continued)

name	type	bytes	format	MB
files.pdf	stem	3495089411	file	3495.09
files.source	stem	985832	file	0.99
files.txt	stem	558107878	file	558.11
hashes	stem	1418	qs	0.00
latexdefs	stem	1279	file	0.00
lingstats.summary	stem	395	qs	0.00
report.codebook	stem	571434	file	0.57
report.robustness	stem	369595	file	0.37
senatsgruppe_clean	stem	21752	qs	0.02
var_aktenzeichen	stem	104144	qs	0.10
var_constants	stem	17592	qs	0.02
var_ecli	stem	184926	qs	0.18
var_entscheidung_typ	stem	3437	qs	0.00
var_lingstats	stem	198514	qs	0.20
var_verfahrensart	stem	12481	qs	0.01
variables.codebook	stem	2647	qs	0.00
vars_additional	stem	498361	qs	0.50
zip.all	stem	175	qs	0.00
zip.analysis	stem	2340718	file	2.34
zip.csv.full	stem	149565240	file	149.57
zip.csv.meta	stem	1338419	file	1.34
zip.pdf.all	stem	3065725990	file	3065.73
zip.pdf.leitsatz	stem	143238655	file	143.24
zip.source	stem	847733	file	0.85
zip.txt	stem	192698000	file	192.70

7.2 Timing

7.2.1 Gesamte Laufzeit

```
meta <- tar_meta(fields = c("time", "seconds"), complete_only = TRUE)
setDT(meta)
meta$mins <- round(meta$seconds / 60, digits = 2)

runtime.sum <- sum(meta$seconds)

## Sekunden
print(runtime.sum)
#> [1] 14063.32

## Minuten
runtime.sum / 60
#> [1] 234.3886

## Stunden
runtime.sum / 3600
#> [1] 3.906477
```

7.2.2 Laufzeit einzelner Targets

Der Zeitpunkt an dem die Targets berechnet wurden und ihre jeweilige Laufzeit in Sekunden.

```
kable(meta[order(-seconds)],
      format = "latex",
      align = "r",
      booktabs = TRUE,
      longtable = TRUE) %>% kable_styling(latex_options = "repeat_header")
```

name	time	seconds	mins
files.pdf	2023-04-03 03:12:29	11997.306	199.96
files.html	2023-04-02 23:51:48	1327.931	22.13
var_lingstats	2023-04-03 03:20:29	157.032	2.62
lingstats.summary	2023-04-03 03:23:13	152.667	2.54
files.txt	2023-04-03 03:13:45	98.495	1.64
zip.pdf.all	2023-04-03 03:15:47	94.784	1.58
dt.bpatg	2023-04-03 03:17:19	85.250	1.42
dt.download	2023-04-02 23:52:32	43.943	0.73
zip.csv.full	2023-04-03 03:23:43	29.840	0.50
zip.txt	2023-04-03 03:17:46	27.252	0.45

(continued)

name	time	seconds	mins
hashes	2023-04-03 03:24:07	13.333	0.22
report.codebook	2023-04-03 03:23:54	11.012	0.18
report.robustness	2023-04-03 03:20:40	8.174	0.14
dt.scope	2023-04-02 23:29:40	5.694	0.09
zip.pdf.leitsatz	2023-04-03 03:15:52	4.380	0.07
var_aktenzeichen	2023-04-03 03:17:52	3.379	0.06
files.source	2023-04-02 23:29:20	1.277	0.02
dt.download.final	2023-04-02 23:52:33	0.332	0.01
zip.csv.meta	2023-04-03 03:23:54	0.244	0.00
csv.full	2023-04-03 03:20:40	0.200	0.00
dt.bpatg.final	2023-04-03 03:20:32	0.189	0.00
az_clean	2023-04-02 23:52:32	0.153	0.00
var_entscheidung_typ	2023-04-03 03:17:52	0.106	0.00
zip.analysis	2023-04-03 03:23:54	0.092	0.00
var_ecli	2023-04-03 03:20:29	0.084	0.00
zip.source	2023-04-02 23:29:40	0.055	0.00
senatsgruppe_clean	2023-04-02 23:52:32	0.039	0.00
dt.bpatg.datecleaned	2023-04-03 03:17:49	0.017	0.00
latexdefs	2023-04-02 23:29:40	0.016	0.00
csv.meta	2023-04-03 03:23:43	0.014	0.00
var_constants	2023-04-03 03:17:52	0.012	0.00
dt.bpatg.meta	2023-04-03 03:20:40	0.006	0.00
az.brd	2023-04-02 23:51:48	0.003	0.00
var_verfahrensart	2023-04-03 03:20:29	0.002	0.00
vars_additional	2023-04-03 03:20:29	0.002	0.00
changelog	2023-04-02 18:24:41	0.001	0.00
csv.hashes	2023-04-03 03:24:07	0.001	0.00
zip.all	2023-04-03 03:23:54	0.001	0.00
file.az.brd	2022-07-16 22:40:26	0.000	0.00

(continued)

name	time	seconds	mins
file.variables.codebook	2023-04-02 18:24:41	0.000	0.00
variables.codebook	2023-04-02 23:51:48	0.000	0.00

8 Pipeline: Warnungen

```
meta <- tar_meta(fields = "warnings", complete_only = TRUE)
setDT(meta)
meta$warnings <- gsub("(\\.pdf|\\.html?|\\.txt)", "\\1 \n\n", meta$warnings)
meta <- meta[name != "dt.annotated"] # de_core_news_sm does not work correctly
    with lemmatization; warnings are not reported because they flood the document
    ; should be inspected directly

if (meta[,.N > 0]){

    for(i in 1:meta[,.N]){

        cat(paste("##", meta[i]$name), "\n\n")
        cat(paste(meta[i]$warnings, "\n\n"))

    }

}else{

    cat("No warnings to report.")

}
```

8.1 report.codebook

Package microtype Warning Unable to apply patch footnote on input line 197.

8.2 report.robustness

Package microtype Warning Unable to apply patch footnote on input line 206.

9 Pipeline: Fehlermeldungen

```
meta <- tar_meta(fields = "error", complete_only = TRUE)
setDT(meta)

if (meta[,.N > 0]){

    for(i in 1:meta[,.N]){

        cat(paste("##", meta[i]$name), "\n\n")
        cat(paste(meta[i]$error, "\n\n"))

    }

}else{

    cat("No errors to report.")

}
#> No errors to report.
```

10 Dateigrößen

10.1 ZIP-Dateien

```
files <- list.files("output", pattern = "\\.zip", full.names = TRUE)

filesize <- round(file.size(files) / 10^6, digits = 2)

table.size <- data.table(basename(files),
                         filesize)


kable(table.size,
      format = "latex",
      align = c("l", "r"),
      format.args = list(big.mark = ","),
      booktabs = TRUE,
      longtable = TRUE,
      col.names = c("Datei",
                    "Größe in MB"))
```

Datei	Größe in MB
CE-BPatG_2023-04-02_DE_Analyse.zip	2.34
CE-BPatG_2023-04-02_DE_CSV_Datensatz.zip	149.57
CE-BPatG_2023-04-02_DE_CSV_Metadaten.zip	1.34
CE-BPatG_2023-04-02_DE_PDF_Datensatz.zip	3,065.73
CE-BPatG_2023-04-02_DE_PDF_Leitsatz-Entscheidungen.zip	143.24
CE-BPatG_2023-04-02_DE_TXT_Datensatz.zip	192.70
CE-BPatG_2023-04-02_Source_Code.zip	0.85
CE-BPatG_2023-04-02_Targets_Storage.zip	437.19

10.2 CSV-Dateien

```
files <- list.files("output", pattern = "\\.csv", full.names = TRUE)

filesize <- round(file.size(files) / 10^6, digits = 2)

table.size <- data.table(basename(files),
                         filesize)


kable(table.size,
      format = "latex",
      align = c("l", "r"),
      format.args = list(big.mark = ","),
      booktabs = TRUE,
      longtable = TRUE,
      col.names = c("Datei",
                    "Größe in MB"))
```

Datei	Größe in MB
CE-BPatG_2023-04-02_DE_CSV_Datensatz.csv	568.59
CE-BPatG_2023-04-02_DE_CSV_Metadaten.csv	17.56
CE-BPatG_2023-04-02_KryptographischeHashes.csv	0.00

11 Kryptographische Signaturen

11.1 Signaturen laden

```
tar_load(hashes)
```

11.2 Leerzeichen hinzufügen um bei SHA3-512 Zeilenumbruch zu ermöglichen

Hierbei handelt es sich lediglich um eine optische Notwendigkeit. Die normale 128 Zeichen lange Zeichenfolge von SHA3-512-Signaturen wird ansonsten nicht umgebrochen und verschwindet über die Seitengrenze. Das Leerzeichen erlaubt den automatischen Zeilenumbruch und damit einen für Menschen sinnvoll lesbaren Abdruck im Codebook. Diese Variante wird nur zur Anzeige verwendet und danach verworfen.

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

11.3 In Bericht anzeigen

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

index	filename
1	output/CE-BPatG_2023-04-02_DE_PDF_Datensatz.zip
2	output/CE-BPatG_2023-04-02_DE_PDF_Leitsatz-Entscheidungen.zip
3	output/CE-BPatG_2023-04-02_DE_TXT_Datensatz.zip
4	output/CE-BPatG_2023-04-02_DE_CSV_Datensatz.zip
5	output/CE-BPatG_2023-04-02_DE_CSV_Metadaten.zip
6	output/CE-BPatG_2023-04-02_DE_Analyse.zip
7	output/CE-BPatG_2023-04-02_Source_Code.zip
8	output/CE-BPatG_2023-04-02_Codebook.pdf
9	output/CE-BPatG_2023-04-02_RobustnessChecks.pdf

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

index	sha2.256
1	c650578256d01932e574ca5c0d201c68594b8a156798a1de67eb9f17d6bf2426
2	8ecc0d1a1778fe19af4b5de9fe4d288be5c9d9f0ff32af7cd550472dcb5641d7
3	b96376d54a8b36601ef5a15096bc24fa88905b313fa5180afc0bbb3a1a1d5c04
4	69717dc3473303ccd8e6b04c2b46b541a35aa8813183a796ab5077c9c78132cc
5	e7401112c3d8a3f57c752fb086a422411d23a374ad0c31ca8630977da381419e
6	907212b8ef4a845e6a551a19c65c58d5e13e9ecee14e63b58641ba8d4f57f385
7	dce3151bf8dfdeaea7236b7a792b905b3ae1489a78513869254166cfbe9dcb9b
8	c1a5cfdd538a6bc85a4989cb878263b2806bedb6e9791e505a42a8bc5dfbd17d
9	dfc05b852bfea22f9215ed793752b5356c072442f2f10ba7b26d6bf8fd1150a9

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

index	sha3.512
1	5f6fd61687189b750ac7902aeb1f35db623b7fb18b4c6b286eccfe90a72bb535 0098238f539e0f1891b05af7008158cdaa2ba883570239e9edf51e28a8d34aa4
2	b76c2ab10ab7c52838a76b4c8e3a296caf72dbff4133170f97d4d1d1f9f2b858 e502dd489ff837cbdb1275dfd94461ba5305d3ae467980ba2cb3131fba4d06d6
3	e0467cafbe5b6884c14796067165521d29d432b39d7c9f32c89a3a9b22b19892 c0b7774df3a9dd0fb881b6309e1567fcf4eac0ce176b40ba9d81e20bf047d019
4	1dddc6aed7677b135ddd949e6a256c0c2004754c6e814cc5bff1fe4a596a8623 5ed9e1b020cf193adfb671b36eb943bae1977744e09854eeee3fe65c2b9e7d30

5	ea7691dd83be5b62cbcf977b4a4f826258dea03c5c4c3a94bb8f52cb6e91c0e8 7c3ac1df077f49b80358ba25b438aae4e158cfb8240cb67b2d56694c3087b7ff
6	0bae19259e065684a28cf13cdc97bc9669fd1bfc960705af4506ca204ab0c6c3 d8bff6c74fa9db2a101afebc84a42640ad7c0d8451869de546b8e0483aef5cf8
7	3cfc41b9877fc6e3dc5ddb37f54518e8f79f1a9818e5e198cf31f031c9dce93a 70e74c5687cb50c51f7bfc15f5768f241653b94e316ddcbef229908864882080
8	7853b6979bfde6b251a6a5867298b0cf008357852389bc3736a7888f1b0e2d68 2d6122ccfe5210baa8542f5a51e2f6fbbfd1de68f1e036e946457a8167b3d833
9	649c33f790261f5f94aff3894438f94a1ac334a80d43b394f4d9a246a24b6798 fbf5b72e935f9d6958e85ab0558e7bef6530bd7bc0c2ad4526ae390377ef26ed

12 Changelog

12.1 Version 2023-04-02

- Vollständige Aktualisierung der Daten
- Gesamte Laufzeitumgebung mit Docker versionskontrolliert
- Aktenzeichen aus dem Eingangszeitraum 2000 bis 2009 nun korrekt mit führender Null formatiert (z.B. 1 BvR 44/02 statt 1 BvR 44/2)
- Vereinfachung der Konfigurationsdatei
- Run- und Delete-Skripte aktualisiert
- Neue Funktion für automatischen clean run (Löschung aller Zwischenergebnisse)
- Neuorganisation des Repositories
- Inhalt des ZIP-Archivs mit dem Source Code orientiert sich nun an der Versionskontrolle mit Git und enthält auch die gesamte Git-Historie
- Proto-Package Mono-Repo entfernt, alle Funktionen nun fest projektbasiert versionskontrolliert
- Update der Download-Funktion
- Überflüssige Warnung in f.future_lingsummarize-Funktion entfernt
- Zusätzliche Unit-Tests
- Alle Roh-Dateien werden nun im Ordner "files/" gespeichert
- Verbesserung des Robustness Check Reports
- Verbesserung des Codebooks
- Alle Diagramme neu nummeriert
- Verbesserte Formatierung von Profiling, Warnungen und Fehlermeldungen im Compilation Report
- README im Hinblick auf Docker überarbeitet
- Alle Zwischenergebnisse der Pipeline werden automatisch im Ordner "output/" archiviert
- Umfang der Datenbankabfrage ist nun vollständig automatisiert
- Zwischenergebnisse werden im qs-Format gespeichert um Speicherplatz zu sparen

12.2 Version 2022-07-11

- Vollständige Aktualisierung der Daten
- Neuer Entwurf des gesamten Source Code im {targets} Framework
- Veröffentlichung des Source Codes

12.3 Version 2020-07-20

- Erstveröffentlichung

13 Abschluss

```
## Datumsstempel
print(datestamp)
#> [1] "2023-04-02"

## Datum und Uhrzeit (Anfang)
print(begin.script)
#> [1] "2023-04-02 23:29:31 UTC"


## Datum und Uhrzeit (Ende)
end.script <- Sys.time()
print(end.script)
#> [1] "2023-04-03 03:24:21 UTC"


## Laufzeit des gesamten Skriptes
print(end.script - begin.script)
#> Time difference of 3.913765 hours
```

14 Parameter für strenge Replikationen

```
system2("openssl", "version", stdout = TRUE)
#> [1] "OpenSSL 3.0.2 15 Mar 2022 (Library: OpenSSL 3.0.2 15 Mar 2022)"

sessionInfo()
#> R version 4.2.2 (2022-10-31)
#> Platform: x86_64-pc-linux-gnu (64-bit)
#> Running under: Ubuntu 22.04.2 LTS
#>
#> Matrix products: default
#> BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
#> LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.20.so
#>
#> locale:
#>  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
#>  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
#>  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
#>  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
#>  [9] LC_ADDRESS=C               LC_TELEPHONE=C
#> [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
#>
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
#>  [1] ggraph_2.1.0      ggplot2_3.4.1     igraph_1.4.1      kableExtra_1.3.4
#>  [5] knitr_1.42        quanteda_3.2.4    data.table_1.14.8 future_1.32.0
#>  [9] RcppTOML_0.2.2    tarchetypes_0.7.5 targets_0.14.3
#>
#> loaded via a namespace (and not attached):
#>  [1] viridis_0.6.2       httr_1.4.5          tidyr_1.3.0
#>  [4] bit64_4.0.5         tidygraph_1.2.3     viridisLite_0.4.1
#>  [7] RcppParallel_5.1.7  highr_0.10          future.callr_0.8.1
#> [10] base64url_1.4       renv_0.17.2         yaml_2.3.7
#> [13] ggrepel_0.9.3       globals_0.16.2      pillar_1.8.1
#> [16] backports_1.4.1     lattice_0.20-45     glue_1.6.2
#> [19] digest_0.6.31       polyclip_1.10-4     rvest_1.0.3
#> [22] stringfish_0.15.7   colorspace_2.1-0    htmltools_0.5.4
#> [25] Matrix_1.5-1        pkgconfig_2.0.3     listenv_0.9.0
#> [28] purrr_1.0.1         scales_1.2.1        webshot_0.5.4
#> [31] processx_3.8.0      svglite_2.1.1       tweenr_2.0.2
#> [34] RApiSerialize_0.1.2 ggforce_0.4.1       tibble_3.2.1
#> [37] generics_0.1.3      farver_2.1.1        withr_2.5.0
#> [40] furrr_0.3.1         cli_3.6.0           crayon_1.5.2
#> [43] magrittr_2.0.3      evaluate_0.20       ps_1.7.2
#> [46] stopwords_2.3       fs_1.6.1            fansi_1.0.4
#> [49] parallelly_1.34.0   MASS_7.3-58.1       xml2_1.3.3
#> [52] tools_4.2.2         lifecycle_1.0.3     stringr_1.5.0
#> [55] munsell_0.5.0       callr_3.7.3         compiler_4.2.2
#> [58] qs_0.25.5           systemfonts_1.0.4   rlang_1.1.0
#> [61] grid_4.2.2          rstudioapi_0.14     labeling_0.4.2
#> [64] rmarkdown_2.20      gtable_0.3.2        codetools_0.2-18
#> [67] graphlayouts_0.8.4  R6_2.5.1            gridExtra_2.3
#> [70] dplyr_1.1.0         bit_4.0.5           fastmap_1.1.1
```

```
#> [73] utf8_1.2.3           fastmatch_1.1-3     stringi_1.7.12
#> [76] parallel_4.2.2       Rcpp_1.0.10         vctrs_0.6.0
#> [79] tidyselect_1.2.0     xfun_0.37
```

Literaturverzeichnis

Allaire, JJ, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone. 2023. *Rmarkdown: Dynamic Documents for R.*

Bengtsson, Henrik. 2021. "A Unifying Framework for Parallel and Distributed Processing in R Using Futures." *The R Journal* 13 (2): 208–27. https://doi.org/10.32614/RJ-2021-048.

———. 2022. *Future.apply: Apply Function to Elements in Parallel Using Futures.*

———. 2023. *Future: Unified Parallel and Distributed Processing in R for Everyone.*

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, and William Lowe. 2022. *Quanteda: Quantitative Analysis of Textual Data*. https://quanteda.io.

Csardi, Gabor, and Tamas Nepusz. 2006. "The Igraph Software Package for Complex Network Research." *InterJournal* Complex Systems: 1695. https://igraph.org.

Csárdi, Gábor, Kuba Podgórski, and Rich Geldreich. 2022. *Zip: Cross-Platform Zip Compression*. https://github.com/r-lib/zip#readme.

Dowle, Matt, and Arun Srinivasan. 2023. *Data.table: Extension of 'Data.frame'.*

Eddelbuettel, Dirk. 2023. *RcppTOML: Rcpp Bindings to Parser for "Tom's Obvious Markup Language"*. http://dirk.eddelbuettel.com/code/rcpp.toml.html.

Ewing, Mark. 2021. *Mgsub: Safe, Multiple, Simultaneous String Substitution.*

file., See AUTHORS. 2023. *Igraph: Network Analysis and Visualization.*

Gagolewski, Marek. 2022. "stringi: Fast and Portable Character String Processing in R." *Journal of Statistical Software* 103 (2): 1–59. https://doi.org/10.18637/jss.v103.i02.

Gagolewski, Marek, Bartek Tartanus, others; Unicode, Inc., and others. 2023. *Stringi: Fast and Portable Character String Processing Facilities.*

Landau, William Michael. 2021a. *Tarchetypes: Archetypes for Targets.*

———. 2021b. "The Targets R Package: A Dynamic Make-Like Function-Oriented Pipeline Toolkit for Reproducibility and High-Performance Computing." *Journal of Open Source Software* 6 (57): 2959. https://doi.org/10.21105/joss.02959.

———. 2023a. *Tarchetypes: Archetypes for Targets.*

———. 2023b. *Targets: Dynamic Function-Oriented Make-Like Declarative Pipelines.*

Ooms, Jeroen. 2023. *Pdftools: Text Extraction, Rendering and Converting of Pdf Documents.*

Pedersen, Thomas Lin. 2022. *Ggraph: An Implementation of Grammar of Graphics for Graphs and Networks.*

Ushey, Kevin. 2023. *Renv: Project Environments*. https://rstudio.github.io/renv/.

Wickham, Hadley. 2022. *Rvest: Easily Harvest (Scrape) Web Pages*.

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.

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

———. 2023. *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*.