Die Dauer von Verfahren vor dem Bundesverfassungsgericht

Compilation Report

Version 1.0.0

DOI: 10.5281/zenodo.7133364

Titel	Die Dauer von Verfahren vor dem Bundesverfassungsgericht
Autor	Seán Fobbe
Version	1.0.0
Download	https://doi.org/10.5281/zenodo.7133364
Lizenz	MIT No Attribution (MIT-0)

Zitiervorschlag

Seán Fobbe (2022). Die Dauer von Verfahren vor dem Bundesverfassungsgericht. Version 1.0.0. Zenodo. DOI: 10.5281/zenodo.7133364.

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 1.0.0. 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.7133363. Die »Concept DOI« verlinkt immer die aktuellste Version.

Lizenz: MIT No Attribution (MIT-0)

Inhaltsverzeichnis

1 Vorbemerkungen

1.1 Bericht erstellen

Mit folgendem Befehl können Sie aus dem bereitgestellten R-Skript diesen Bericht kompilieren. Er muss separat in eine R-Konsole eingegeben werden, weil er bei der Erstellung des Berichts nicht mit ausgeführt wird.

Denken Sie daran, vorher renv zu aktivieren, damit die package-Versionen konstant gehalten werden.

```
source("run_project.R")
```

1.2 Datum und Uhrzeit (Anfang)

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

```
## [1] "2022-12-15 18:45:02 CET"
```

2 Vorbereitung

2.1 Packages Laden

```
library(RcppTOML)      # TOML-Verarbeitung
library(kableExtra)    # Verbesserte Kable Tabellen
library(knitr)         # Professionelles Reporting
library(viridis)       # Viridis-Farbpalette
```

```
## Loading required package: viridisLite
```

```
library(igraph)        # Netzwerk-Analytik
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```
library(circlize)      # Chord-Diagramme
```

```
## ========================================
## circlize version 0.4.15
## CRAN page: https://cran.r-project.org/package=circlize
## Github page: https://github.com/jokergoo/circlize
## Documentation: https://jokergoo.github.io/circlize_book/book/
##
## If you use it in published research, please cite:
## Gu, Z. circlize implements and enhances circular visualization
##   in R. Bioinformatics 2014.
##
## This message can be suppressed by:
##   suppressPackageStartupMessages(library(circlize))
## ========================================
```

```
##
## Attaching package: 'circlize'
```

```
## The following object is masked from 'package:igraph':
##
##     degree
```

```
library(zip)          # Cross-Platform ZIP und UNZIP
```

```
##
## Attaching package: 'zip'
```

```
## The following objects are masked from 'package:utils':
##
##     unzip, zip
```

```
library(ggplot2)      # Datenvisualisierung
library(data.table)   # Fortgeschrittene Datenverarbeitung
```

```
## data.table 1.14.2 using 1 threads (see ?getDTthreads).  Latest news: r-
    datatable.com
```

2.2 Output aus vorherigen Runs bereinigen

```
unlink(c("output",
         "figures"),
       recursive = TRUE)
```

2.3 Verzeichnisse anlegen

```
dir.create("data")
dir.create("output")

fig.dir <- paste0(getwd(), "/figures/") # Muss mit einem Schrägstrich enden!
dir.create(fig.dir)
```

2.4 Vollzitate statistischer Software schreiben

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

2.5 Parameter

2.5.1 Parameter einlesen und anzeigen

```
config <- parseTOML("config.toml")
print(config)
```

```
## List of 6
##  $ doi     :List of 1
##   ..$ software:List of 2
##   .. ..$ concept: chr "10.5281/zenodo.7133363"
##   .. ..$ version: chr "10.5281/zenodo.7133364"
##  $ download:List of 1
##   ..$ timeout: int 6000
##  $ fig     :List of 3
##   ..$ align : chr "center"
##   ..$ dpi   : int 300
##   ..$ format: chr [1:2] "pdf" "png"
##  $ license :List of 1
##   ..$ code: chr "MIT-0"
##  $ project :List of 3
##   ..$ author   : chr "Seán Fobbe"
##   ..$ fullname : chr "Die Dauer von Verfahren vor dem Bundesverfassungsgericht
    "
##   ..$ shortname: chr "BVerfG-Verfahrensdauer"
##  $ version :List of 2
##   ..$ dash    : chr "1-0-0"
##   ..$ semantic: chr "1.0.0"
```

2.5.2 Knitr Optionen

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

2.5.3 Download Timeout

```
options(timeout = config$download$timeout)
```

2.5.4 Legende für Diagramme

```
caption <- paste("Fobbe | DOI:",
                 config$doi$software$version)
```

2.5.5 Präfix für Output-Dateien

```
prefix.files <- paste0("output/",
                       config$project$shortname,
                       "_",
                       config$version$dash)
```

3 Datenquelle

3.1 Erläuterungen

Für diese Analyse nutze ich den Corpus der Entscheidungen des Bundesverfassungsgerichts (CE-BVerfG) in der Version 2022-08-24.

Der CE-BVerfG enthält den gesamten am jeweiligen Stichtag auf der offiziellen Website des Bundesverfassungsgerichts veröffentlichen Bestand an Entscheidungen. Für Details zu dem Datensatz lesen Sie bitte das Codebook.

Die konkrete Version des Datensatzes und das zugehörige Codebook können dauerhaft über folgenden Link abgerufen werden: https://doi.org/10.5281/zenodo.7011305

Alle von mir veröffentlichten juristischen open access Datensätze finden Sie hier: www.seanfobbe.de/data

3.2 PDF-Datensatz herunterladen

```
if (file.exists("data/CE-BVerfG_2022-08-24_DE_PDF_Datensatz.zip") == FALSE){
    download.file("https://zenodo.org/record/7011305/files/CE-BVerfG_2022-08-24_
    DE_PDF_Datensatz.zip?download=1",
                  "data/CE-BVerfG_2022-08-24_DE_PDF_Datensatz.zip")
    }
```

3.3 CSV-Datensatz herunterladen

```
if (file.exists("data/CE-BVerfG_2022-08-24_DE_CSV_Datensatz.zip") == FALSE){
    download.file("https://zenodo.org/record/7011305/files/CE-BVerfG_2022-08-24_
    DE_CSV_Datensatz.zip?download=1",
                  "data/CE-BVerfG_2022-08-24_DE_CSV_Datensatz.zip")
    }


unzip("data/CE-BVerfG_2022-08-24_DE_CSV_Datensatz.zip",
      exdir = "data")
```

3.4 CSV-Datensatz einlesen

```
bverfg.gesamt <- fread("data/CE-BVerfG_2022-08-24_DE_CSV_Datensatz.csv")
```

4 Deduplikation

Nicht wenige Entscheidungen enthalten dasselbe Aktenzeichen und beziehen sich deshalb auf dasselbe Verfahren. Um die Analyse nicht zu verfälschen wird jeweils nur die späteste Entscheidung pro Aktenzeichen berücksichtigt.

4.1 Anzahl aller Entscheidungen im Korpus

```
bverfg.gesamt[,.N]
```

```
## [1] 8407
```

4.2 Entfernung von untergeordneten Entscheidungen

Zunächst werden alle Entscheidungen auf Festsetzung des Gegenstandswertes, Festsetzung der notwendigen Auslagen, Gegenvorstellungen, Erinnerungen und Berichtigungen entfernt.

```
index1 <- grep("Antr[äa]ge? auf Festsetzung des Gege?nstandswerte?s",
               bverfg.gesamt$text,
               ignore.case = TRUE)

length(index1)
```

```
## [1] 117
```

```
index2 <- grep("h *i *e *r *: *Festsetzung des Gege?nstandswerte?s",
               bverfg.gesamt$text,
               ignore.case = TRUE)


length(index2)
```

```
## [1] 37
```

```
index3 <- grep("h *i *e *r *: *Erstattung notwendiger Auslagen",
               bverfg.gesamt$text,
               ignore.case = TRUE)

length(index3)
```

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

```
index4 <- grep("wird (dahin|dahingehend|(wie folgt)) berichtigt",
               bverfg.gesamt$text,
               ignore.case = TRUE)

length(index4)
```

```
## [1] 11
```

```
index5 <- grep("h *i *e *r *: *(Berichtigung|Gegenvorstellung|Erinnerung)",
               bverfg.gesamt$text,
               ignore.case = TRUE)

length(index5)
```

```
## [1] 26
```

```
index <- unique(c(index1, index2, index3, index4, index5))

length(index) # Anzahl der zu entfernenden Entscheidungen
```

```
## [1] 185
```

```
bverfg.reduziert <- bverfg.gesamt[-index]
```

4.3 Mehrfach auftauchende Aktenzeichen quantifizieren

Nun berechnen wir wieviele Aktenzeichen wie oft vorkommen. Beispielsweise kommt eine bestimmte Zahl Aktenzeichen einmal vor, dann eine geringere Anzahl zweimal, und so weiter.

```
bverfg.reduziert[,
                 .N,
                 keyby = "aktenzeichen"][order(-N)
                                         ][,.N, keyby = "N"]
```

```
##    N    N
## 1: 1 7335
## 2: 2  289
## 3: 3   41
## 4: 4   16
## 5: 5    5
## 6: 6    7
## 7: 7    3
## 8: 8    2
## 9: 9    2
```

4.4 Deduplikation durchführen

Wir behalten für jedes einzigartige Aktenzeichen nur die spätestmögliche Entscheidung, um die Gesamtdauer des Verfahrens abzuschätzen.

```
bverfg.bereinigt <- unique(bverfg.reduziert[order(datum)],
                           fromLast = TRUE,
                           by = "aktenzeichen")
```

Die Anzahl der Entscheidungen im bereinigten Korpus ist:

```
bverfg.bereinigt[,.N]
```

```
## [1] 7700
```

4.5 Späteste Entscheidungen herauskopieren

Die spätesten Entscheidungen für jedes Verfahren werden in einen separaten Ordner sortiert, damit sie untersucht werden können.

```
ids <- setdiff(bverfg.reduziert$doc_id, bverfg.bereinigt$doc_id)
az <- bverfg.reduziert[doc_id %in% ids]$aktenzeichen

late <- bverfg.bereinigt[aktenzeichen %in% az]$doc_id


late.pdf <- gsub("\\.txt", "\\.pdf", late)


unzip(zipfile = "data/CE-BVerfG_2022-08-24_DE_PDF_Datensatz.zip",
      files = late.pdf,
      exdir = "output/spaeteste-entscheidungen/")
```

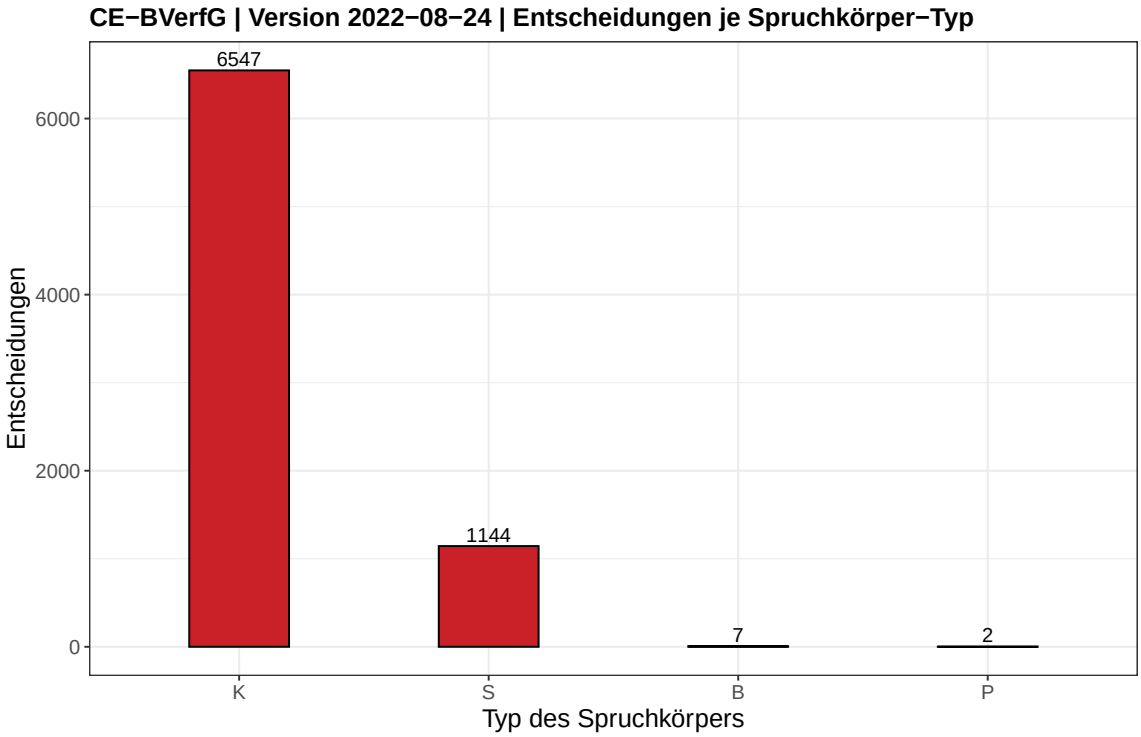
5 Deskriptiver Überblick über den Korpus

Die Datensätze enthalten jeweils eine Entscheidung pro Zeile. Die folgende Frequenztabelle gibt Aufschluss über die Anzahl Entscheidungen je Spruchkörpertyp (Senat, Kammer, Beschwerdekammer, Plenum). Diese Frequenztabelle visualisieren wir als Säulendiagramm.

```
freqtable.typ <- bverfg.bereinigt[,
                                  .N,
                                  keyby = "spruchkoerper_typ"]

print(freqtable.typ)
```

```
##    spruchkoerper_typ    N
## 1:                 B    7
## 2:                 K 6547
## 3:                 P    2
## 4:                 S 1144
```

```
ggplot(data = freqtable.typ) +
    geom_bar(aes(x = reorder(spruchkoerper_typ,
                             -N),
                 y = N),
             stat = "identity",
             fill = "#ca2129",
             color = "black",
             width = 0.4) +
    geom_text(aes(x = reorder(spruchkoerper_typ,
                              -N),
                  y = N,
                  label = N),
              vjust = -0.3)+
    theme_bw() +
    labs(
        title = paste("CE-BVerfG",
                      "| Version",
                      unique(bverfg.bereinigt$version),
                      "| Entscheidungen je Spruchkörper-Typ"),
        caption = caption,
        x = "Typ des Spruchkörpers",
        y = "Entscheidungen"
    )+
    theme(
        text = element_text(size = 14),
        plot.title = element_text(size = 14,
                                  face = "bold"),
        legend.position = "none",
        plot.margin = margin(10, 20, 10, 10)
    )
```


CE−BVerfG | Version 2022−08−24 | Entscheidungen je Spruchkörper−Typ
6547
1144
7
2
6000
4000
2000
0
Entscheidungen
K
S
B
P
Typ des Spruchkörpers
Fobbe | DOI: 10.5281/zenodo.7133364

6 Analyse der Verfahrensdauer

Um die Verfahrensdauer zu bestimmen stehen für jede Entscheidung zwei Angaben zur Verfügung: das Eingangsjahr und das Entscheidungsjahr. Beide wurden aus der ECLI extrahiert. Die Berechnung ist denkbar einfach: Entscheidungsjahr minus Eingangskahr. Eine Verfahrensdauer von »0« bedeutet, dass das Verfahren im gleichen Jahr entschieden wurde. Zur Robustheit dieser Berechnungsmethode werden am Ende des Berichts verschiedene Simulationen durchgeführt.

- **Einschränkung 1:** Es werden nur veröffentlichte und begründete Entscheidungen analysiert. Diese machen einen Bruchteil der gesamten Arbeitslast des Gerichtes aus. So hat das BVerfG beispielsweise im Jahr 2018 insgesamt 6.231 Verfahren erledigt, es wurden davon aber nur 276 begründete Entscheidungen veröffentlicht, die im CE-BVerfG dokumentiert werden konnten.
- **Einschränkung 2:** Eine genauere Bestimmung der Verfahrenslänge als in Jahren ist nicht möglich, weil zwar das Entscheidungsdatum, nicht aber das Eingangsdatum bekannt ist.
- **Einschränkung 3:** Manche Entscheidungen beenden mehrere Verfahren mit verschiedenen Aktenzeichen gleichzeitig. Es ist durchaus diskussionswürdig, ob diese Tatsache die Analyse nicht spürbar verzerrt. Ich meine es ist vertretbar, bei einer parallelen Verhandlung und Entscheidung von mehreren Aktenzeichen ex-post von *einem* Verfahren und nicht von mehreren zu sprechen.
- **Einschränkung 4:** Im Falle mehrerer Aktenzeichen, die durch eine Entscheidung beendet wurden, ist das Eingangsjahr maßgeblich, welches in der ECLI dokumentiert ist. Dieses richtet sich nach dem Pilotverfahren.
- **Einschränkung 5:** Für jedes einzigartige Aktenzeichen wähle ich die jeweils späteste Entscheidung aus. Es werden dabei grundsätzlich alle Entscheidungen miteinbezogen, ich entferne aber zuvor Entscheidungen, die nur die Festsetzung des Gegenstandswertes, die Erstattung notwendiger Auslagen, oder die Berichtigung vergangener Entscheidungen betreffen, sowie Erinnerungen und Gegenvorstellungen hierzu. Die Entfernung basiert auf eng formulierten *regular expressions* und kann daher durch Tippfehler, ungewöhnliche Formulierungen oder plötzliche Silbentrennungen im Text nicht alle solche Entscheidungen erkennen. In der Regel sollten die untergeordneten Entscheidungen aber zeitlich in der Nähe der Sachentscheidung ergangen sein und die Ergebnisse nicht spürbar verzerren.

6.1 Einzelne Verfahrensdauern berechnen

```
bverfg.bereinigt$verfahrensdauer <- bverfg.bereinigt$entscheidungsjahr - bverfg.
    bereinigt$eingangsjahr_iso
```

6.2 Alle Spruchkörper-Typen

6.2.1 Zusammenfassung anzeigen

```
summary(bverfg.bereinigt$verfahrensdauer)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   1.000   1.572   2.000  21.000
```

6.2.2 Entscheidungen mit Verfahrensdauer ab 10 Jahren anzeigen

```
dt.langedauer <- bverfg.bereinigt[verfahrensdauer >= 10
                                  ][order(-verfahrensdauer)
                                    ][,.(doc_id, datum, aktenzeichen,
    verfahrensdauer)]


kable(dt.langedauer[,.(datum, aktenzeichen, verfahrensdauer)],
      format = "latex",
      align = "r",
      booktabs = TRUE,
      longtable = TRUE) %>% kable_styling(latex_options = "repeat_header")
```

datum	aktenzeichen	verfahrensdauer
2012-12-30	1 BvR 1237/91	21
1994-10-06	2 BvR 856/81	13
1997-10-10	1 BvR 310/84	13
1998-03-30	1 BvR 1172/85	13
2001-05-08	1 BvF 3/88	13
1998-02-18	1 BvR 1318/86	12
1998-02-26	1 BvR 1114/86	12
1998-06-10	1 BvR 1485/86	12
2002-02-06	1 BvR 952/90	12
2004-03-03	1 BvF 3/92	12
1998-01-27	1 BvL 15/87	11
1998-04-09	1 BvR 415/87	11
1998-04-09	1 BvR 416/87	11
1998-07-08	1 BvR 851/87	11
1999-02-18	1 BvR 1367/88	11
2000-11-14	1 BvL 9/89	11
2001-10-24	1 BvR 1190/90	11
2002-06-26	1 BvR 558/91	11

(continued)

datum	aktenzeichen	verfahrensdauer
2002-06-26	1 BvR 670/91	11
2005-07-26	1 BvR 782/94	11
2006-02-23	2 BvR 2335/95	11
2006-05-02	1 BvR 1351/95	11
1993-05-25	1 BvR 345/83	10
1996-04-24	1 BvR 712/86	10
1997-10-29	1 BvR 780/87	10
1998-04-24	1 BvR 587/88	10
1998-05-26	1 BvR 180/88	10
1999-06-23	1 BvR 984/89	10
1999-09-08	1 BvR 301/89	10
1999-12-02	1 BvR 335/89	10
2000-10-10	1 BvR 586/90	10
2000-10-24	2 BvR 756/90	10
2001-01-19	1 BvR 1759/91	10
2003-10-07	1 BvR 246/93	10
2004-02-25	1 BvR 1564/94	10
2004-07-13	1 BvR 1298/94	10
2005-07-26	1 BvR 80/95	10
2006-01-09	1 BvR 756/96	10
2006-02-15	1 BvR 1317/96	10
2006-03-16	1 BvR 1311/96	10
2019-01-15	2 BvL 1/9	10
2019-03-16	1 BvR 1782/9	10
2021-03-25	2 BvL 1/11	10
2021-10-27	2 BvL 12/11	10

6.2.3 Entscheidungen mit Verfahrensdauer ab 10 Jahren in separaten Ordner speichern

```
langedauer.id <- gsub("\\.txt",
                      "\\.pdf",
                      dt.langedauer$doc_id)


unzip(zipfile = "data/CE-BVerfG_2022-08-24_DE_PDF_Datensatz.zip",
      files = langedauer.id,
      exdir = "output/Entscheidungstexte_Verfahrensdauer-über-10-Jahre")
```

6.2.4 Frequenztabelle erstellen

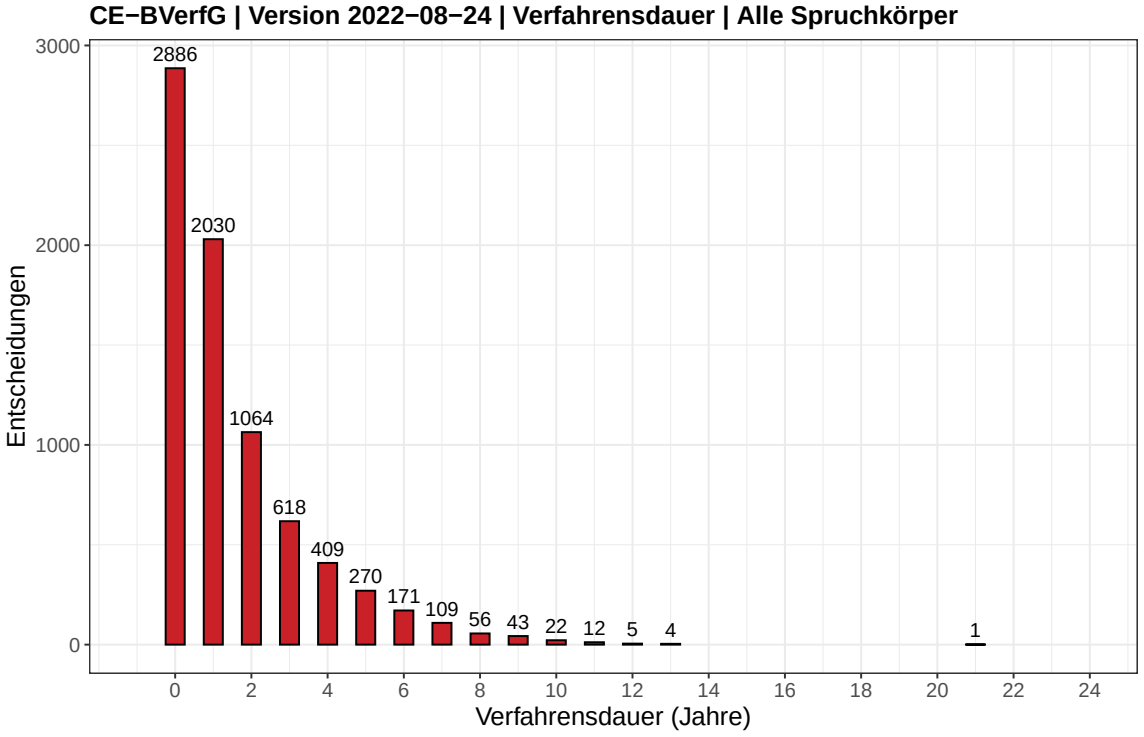
```
bverfg.bereinigt.freqtable <- bverfg.bereinigt[,
                                               .N,
                                               keyby = "verfahrensdauer"]
print(bverfg.bereinigt.freqtable)
```

```
##     verfahrensdauer    N
##  1:               0 2886
##  2:               1 2030
##  3:               2 1064
##  4:               3  618
##  5:               4  409
##  6:               5  270
##  7:               6  171
##  8:               7  109
##  9:               8   56
## 10:               9   43
## 11:              10   22
## 12:              11   12
## 13:              12    5
## 14:              13    4
## 15:              21    1
```

6.2.5 Frequenztabelle visualisieren

```
ggplot(data = bverfg.bereinigt.freqtable) +
    geom_bar(aes(x = verfahrensdauer,
                 y = N),
             stat = "identity",
             fill = "#ca2129",
             color = "black",
             width = 0.5) +
    geom_text(aes(x = verfahrensdauer,
                  y = N,
                  label = N),
              vjust = -0.5)+
    theme_bw() +
    scale_x_continuous(limits = c(-1, 24), breaks = seq(0, 24, by = 2))+
```

```
    labs(
        title = paste("CE-BVerfG",
                      "| Version",
                      unique(bverfg.bereinigt$version),
                      "| Verfahrensdauer | Alle Spruchkörper"),
        caption = caption,
        x = "Verfahrensdauer (Jahre)",
        y = "Entscheidungen"
    )+
    theme(
        text = element_text(size = 14),
        plot.title = element_text(size = 14,
                                  face = "bold"),
        legend.position = "none",
        plot.margin = margin(10, 20, 10, 10)
    )
```



Fobbe | DOI: 10.5281/zenodo.7133364

6.3 Nur Senate

6.3.1 Zusammenfassung anzeigen

```
summary(bverfg.bereinigt[spruchkoerper_typ == "S"]$verfahrensdauer)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.000   2.000   2.866   4.000  13.000
```

6.3.2 Frequenztabelle erstellen

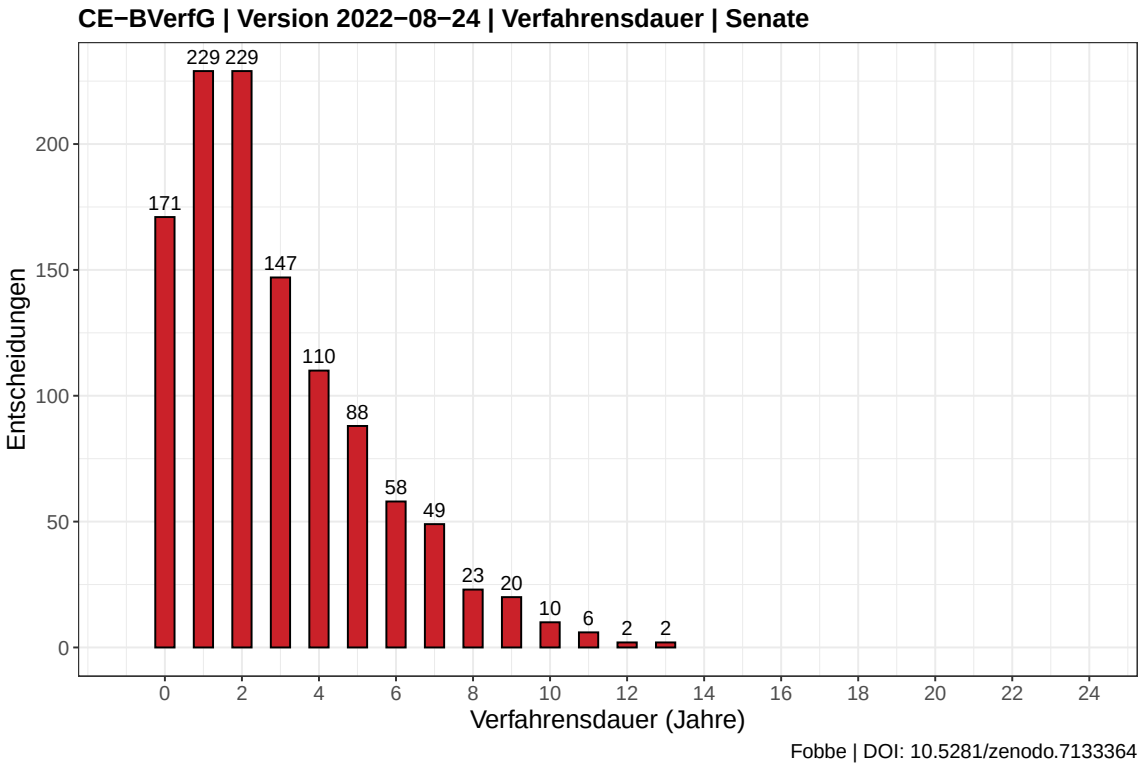
```
bverfg.bereinigt.freqtable.senat <- bverfg.bereinigt[spruchkoerper_typ == "S", .N
    , keyby = "verfahrensdauer"]
print(bverfg.bereinigt.freqtable.senat)
```

```
##     verfahrensdauer   N
##  1:               0 171
##  2:               1 229
##  3:               2 229
##  4:               3 147
##  5:               4 110
##  6:               5  88
##  7:               6  58
##  8:               7  49
##  9:               8  23
## 10:               9  20
## 11:              10  10
## 12:              11   6
## 13:              12   2
## 14:              13   2
```

6.3.3 Frequenztabelle visualisieren

```
ggplot(data = bverfg.bereinigt.freqtable.senat) +
    geom_bar(aes(x = verfahrensdauer,
                 y = N),
             stat = "identity",
             fill = "#ca2129",
             color = "black",
             width = 0.5) +
    geom_text(aes(x = verfahrensdauer,
                  y = N,
                  label = N),
              vjust = -0.5)+
    theme_bw() +
    scale_x_continuous(limits = c(-1, 24), breaks = seq(0, 24, by = 2))+
    labs(
        title = paste("CE-BVerfG",
                      "| Version",
                      unique(bverfg.bereinigt$version),
                      "| Verfahrensdauer | Senate"),
        caption = caption,
        x = "Verfahrensdauer (Jahre)",
        y = "Entscheidungen"
    )+
    theme(
        text = element_text(size = 14),
        plot.title = element_text(size = 14,
                                  face = "bold"),
        legend.position = "none",
```

```
        plot.margin = margin(10, 20, 10, 10)
    )
```



6.4 Nur Kammern

6.4.1 Zusammenfassung anzeigen

```
summary(bverfg.bereinigt[spruchkoerper_typ == "K"]$verfahrensdauer)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   1.000   1.347   2.000  21.000
```

6.4.2 Frequenztabelle erstellen

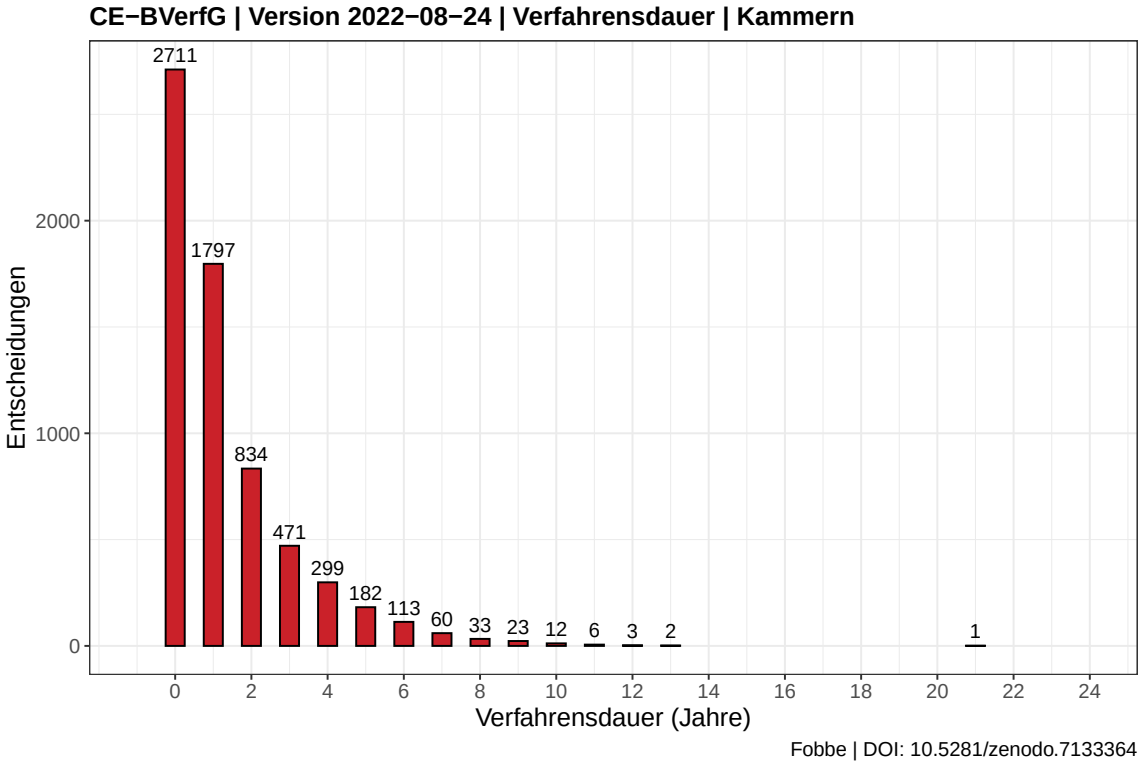
```
bverfg.bereinigt.freqtable.kammer <- bverfg.bereinigt[spruchkoerper_typ == "K", .
    N, keyby = "verfahrensdauer"]
print(bverfg.bereinigt.freqtable.kammer)
```

```
##     verfahrensdauer    N
##  1:               0 2711
##  2:               1 1797
##  3:               2  834
```

```
##  4:                3  471
##  5:                4  299
##  6:                5  182
##  7:                6  113
##  8:                7   60
##  9:                8   33
## 10:                9   23
## 11:               10   12
## 12:               11    6
## 13:               12    3
## 14:               13    2
## 15:               21    1
```

6.4.3 Frequenztabelle visualisieren

```
ggplot(data = bverfg.bereinigt.freqtable.kammer) +
    geom_bar(aes(x = verfahrensdauer,
                 y = N),
             stat = "identity",
             fill = "#ca2129",
             color = "black",
             width = 0.5) +
    geom_text(aes(x = verfahrensdauer,
                  y = N,
                  label = N),
              vjust = -0.5)+
    theme_bw() +
    scale_x_continuous(limits = c(-1, 24), breaks = seq(0, 24, by = 2))+
    labs(
        title = paste("CE-BVerfG",
                      "| Version",
                      unique(bverfg.bereinigt$version),
                      "| Verfahrensdauer | Kammern"),
        caption = caption,
        x = "Verfahrensdauer (Jahre)",
        y = "Entscheidungen"
    )+
    theme(
        text = element_text(size = 14),
        plot.title = element_text(size = 14,
                                  face = "bold"),
        legend.position = "none",
        plot.margin = margin(10, 20, 10, 10)
    )
```


CE−BVerfG | Version 2022−08−24 | Verfahrensdauer | Kammern
Entscheidungen
Verfahrensdauer (Jahre)
2711
1797
834
471
299
182
113
60
33
23
12
6
3
2
1
0
1000
2000
0
2
4
6
8
10
12
14
16
18
20
22
24
Fobbe | DOI: 10.5281/zenodo.7133364

7 Chord-Diagramme

7.1 Erläuterung der Funktion

Diese Funktion nimmt als Input einen Vektor der Eingangsjahre, einen Vektor der Entscheidungsjahre, ein maximales Entscheidungsjahr, sowie Farbwunsch und Tickmarks.

Zunächst werden aus den beiden Vektoren (Eingangsjahr, Entscheidungsjahr) ein Data Frame gebildet und dieses auf Entscheidungen der Entscheidungsjahre 1998 bis 2021 (maximales Entscheidungsjahr) begrenzt, weil der CE-BVerfG das Jahr 2022 noch nicht vollständig abdeckt. Zwar sind auch Entscheidungen aus früheren Jahren als 1998 enthalten, diese sind aber sehr vereinzelt und würden die Analyse verzerren.

Anschließend behandeln wir das Data Frame wie eine Edge List (also immer A -> B bzw. Eingangsjahr -> Entscheidungsjahr)

Chord-Diagramme sind sehr komplex und eine Erläuterung der Konstruktion im Detail würde den Rahmen dieser Analyse sprengen. Ich kann aber die R Graph Gallery[1] zum Einstieg und für die Details das Buch zu **circlize**[2] sehr empfehlen.

7.2 Funktion erstellen

```
circleflow.dauer <- function(Eingangsjahr,
                             Entscheidungsjahr,
                             grenze = 2021,
                             colorscheme = "dark",
                             palette,
                             alpha = 0.8,
                             ticks){

    ## === Edge List erstellen ===

    ## Data Table erstellen
    el <- data.table(Eingangsjahr, Entscheidungsjahr)

    ## Entscheidungsjahr Grenze
    el <- el[Entscheidungsjahr >= 1998 & Entscheidungsjahr <= grenze]

    ## Eingangsjahr Grenze
    el <- el[Eingangsjahr >= 1950 & Eingangsjahr <= grenze]

    ## Nach Eingangsjahr sortieren
    el <- el[order(Eingangsjahr)]


    ## === Adjacency Matrix erstellen ===

    g  <- graph.data.frame(el, directed = TRUE)
    data <- get.adjacency(g, edges = F)
    data <- as.matrix(data)
```

[1] https://www.r-graph-gallery.com/chord-diagram.html
[2] https://jokergoo.github.io/circlize_book/

```r
## === Farbpalette ===

if(colorscheme == "dark"){

    col.background <- "black"
    col.legend  <- "white"

}else{

    col.background <- "white"
    col.legend  <- "black"

}


colors <- viridis(nrow(data),
                  alpha = alpha,
                  begin = 0,
                  end = 1,
                  option = palette)


## === Parameter ===
circos.clear()
circos.par(start.degree = 90,
           gap.degree = 2,
           track.margin = c(-0.1, 0.1),
           points.overflow.warning = FALSE)

par(mar = c(3, 3, 5, 3), #c(0.1, 0.1, 3, 0.1)
    bg = col.background,
    col =  col.legend,
    col.main = col.legend,
    col.sub = col.legend)

## === Chord-Diagramm erstellen ===
chordDiagram(
    x = data,
    grid.col = colors,
    transparency = 0.1,
    directional = 1,
    direction.type = c("arrows", "diffHeight"),
    diffHeight  = -0.04,
    annotationTrack = "grid",
    annotationTrackHeight = c(0.05, 0.1),
    link.arr.type = "big.arrow",
    link.sort = FALSE,
    link.largest.ontop = FALSE)

## === Texte und Achse einfügen ===
circos.trackPlotRegion(
    track.index = 1,
    bg.border = NA,
    panel.fun = function(x, y) {
        xlim = get.cell.meta.data("xlim")
```

```
            xplot = get.cell.meta.data("xplot")
            sector.index = get.cell.meta.data("sector.index")

            circos.text(mean(xlim),
                        y = 2.2,
                        labels = sector.index,
                        facing = "clockwise",
                        niceFacing = TRUE,
                        cex = 0.8,
                        adj = c(0, 0.25)
                        )

            circos.axis(
                h = "top",
                major.at = seq(from = 0,
                               to = xlim[2],
                               ticks),
                labels.cex = 0.5,
                labels.niceFacing = TRUE)
        }
    )
}
```

7.3 Korpus filtern

7.3.1 Kammer- und Senatsentscheidungen filtern

```
bverfg.senat  <- bverfg.bereinigt[spruchkoerper_typ == "S"]
bverfg.kammer <- bverfg.bereinigt[spruchkoerper_typ == "K"]
```

7.3.2 Eingangsjahr filtern

```
Eingangsjahr.senat <- bverfg.senat$eingangsjahr_iso
Eingangsjahr.kammer <- bverfg.kammer$eingangsjahr_iso
```

7.3.3 Entscheidungsjahr filtern

```
Entscheidungsjahr.senat <- bverfg.senat$entscheidungsjahr
Entscheidungsjahr.kammer <- bverfg.kammer$entscheidungsjahr
```

7.4 Chord-Diagramme erstellen

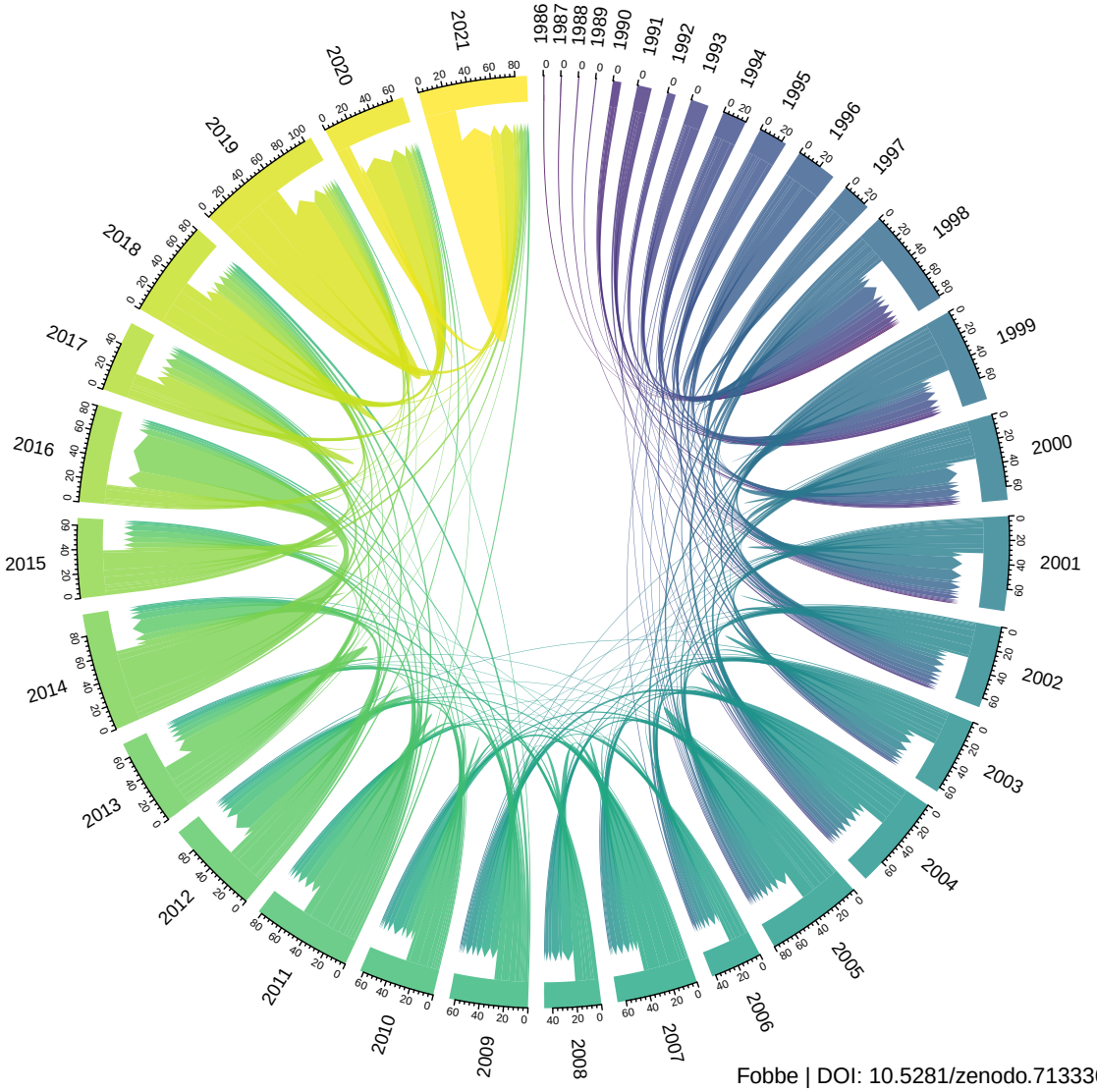
7.4.1 Senate

```
circleflow.dauer(Eingangsjahr = Eingangsjahr.senat,
                 Entscheidungsjahr = Entscheidungsjahr.senat,
                 grenze = 2021,
                 colorscheme = "light",
                 palette = "viridis",
                 alpha = 0.8,
                 ticks = 20)
```

```
## Warning in get.adjacency(g, edges = F): The `edges` argument of
## `as_adjacency_matrix` is deprecated; it will be removed in igraph 1.4.0
```

```
title(main = "CE-BVerfG | Version 2022-08-24 | Verfahrensdauer 1998-2021 | Senate
    ")
legend("bottomright", legend = caption, bty = "n")
```

CE−BVerfG | Version 2022−08−24 | Verfahrensdauer 1998−2021 | Senate

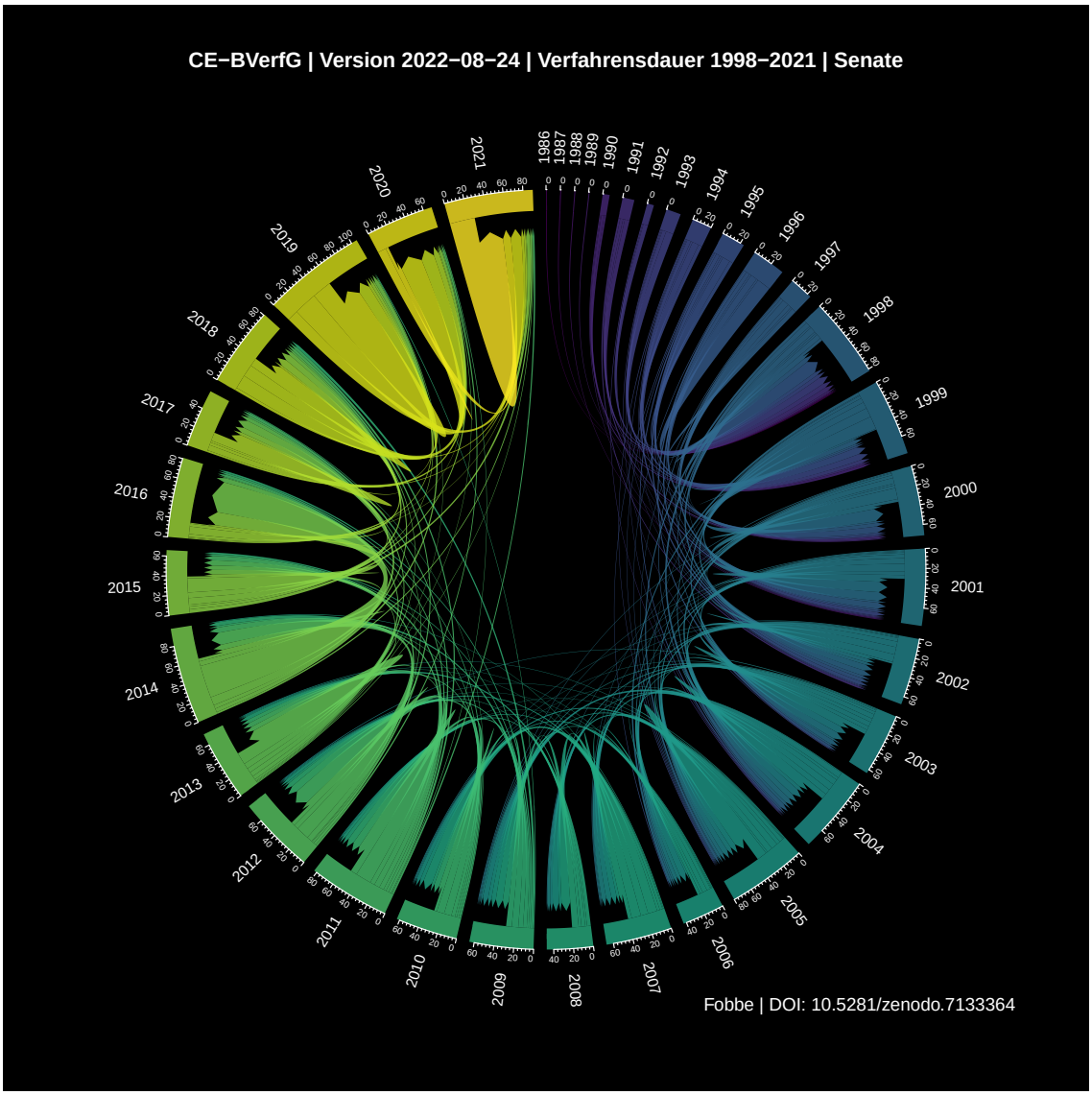


Fobbe | DOI: 10.5281/zenodo.7133364

```
circleflow.dauer(Eingangsjahr = Eingangsjahr.senat,
                 Entscheidungsjahr = Entscheidungsjahr.senat,
                 grenze = 2021,
                 colorscheme = "dark",
                 palette = "viridis",
                 alpha = 0.8,
                 ticks = 20)
```

```
## Warning in get.adjacency(g, edges = F): The `edges` argument of
## `as_adjacency_matrix` is deprecated; it will be removed in igraph 1.4.0
```

```
title(main = "CE-BVerfG | Version 2022-08-24 | Verfahrensdauer 1998-2021 | Senate
    ")
legend("bottomright", legend = caption, bty = "n")
```



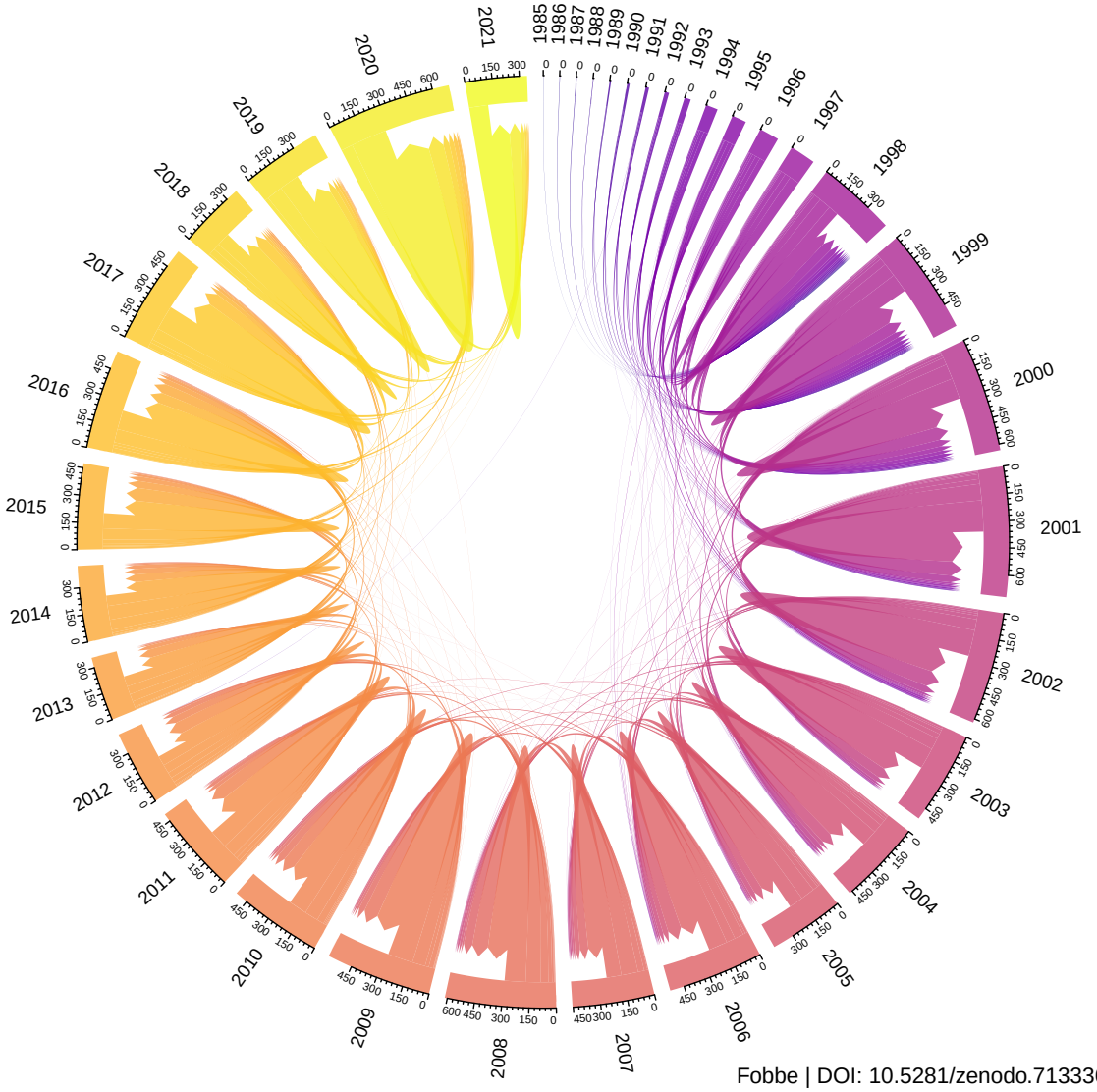
7.4.2 Kammern

```
circleflow.dauer(Eingangsjahr = Eingangsjahr.kammer,
                 Entscheidungsjahr = Entscheidungsjahr.kammer,
                 grenze = 2021,
                 colorscheme = "light",
                 palette = "plasma",
                 alpha = 0.8,
                 ticks = 150)
```

```
## Warning in get.adjacency(g, edges = F): The `edges` argument of
## `as_adjacency_matrix` is deprecated; it will be removed in igraph 1.4.0
```

```
title(main = "CE-BVerfG 2022-08-24 | Verfahrensdauer 1998-2021 | Kammern")
legend("bottomright", legend = caption, bty = "n")
```

CE−BVerfG 2022−08−24 | Verfahrensdauer 1998−2021 | Kammern

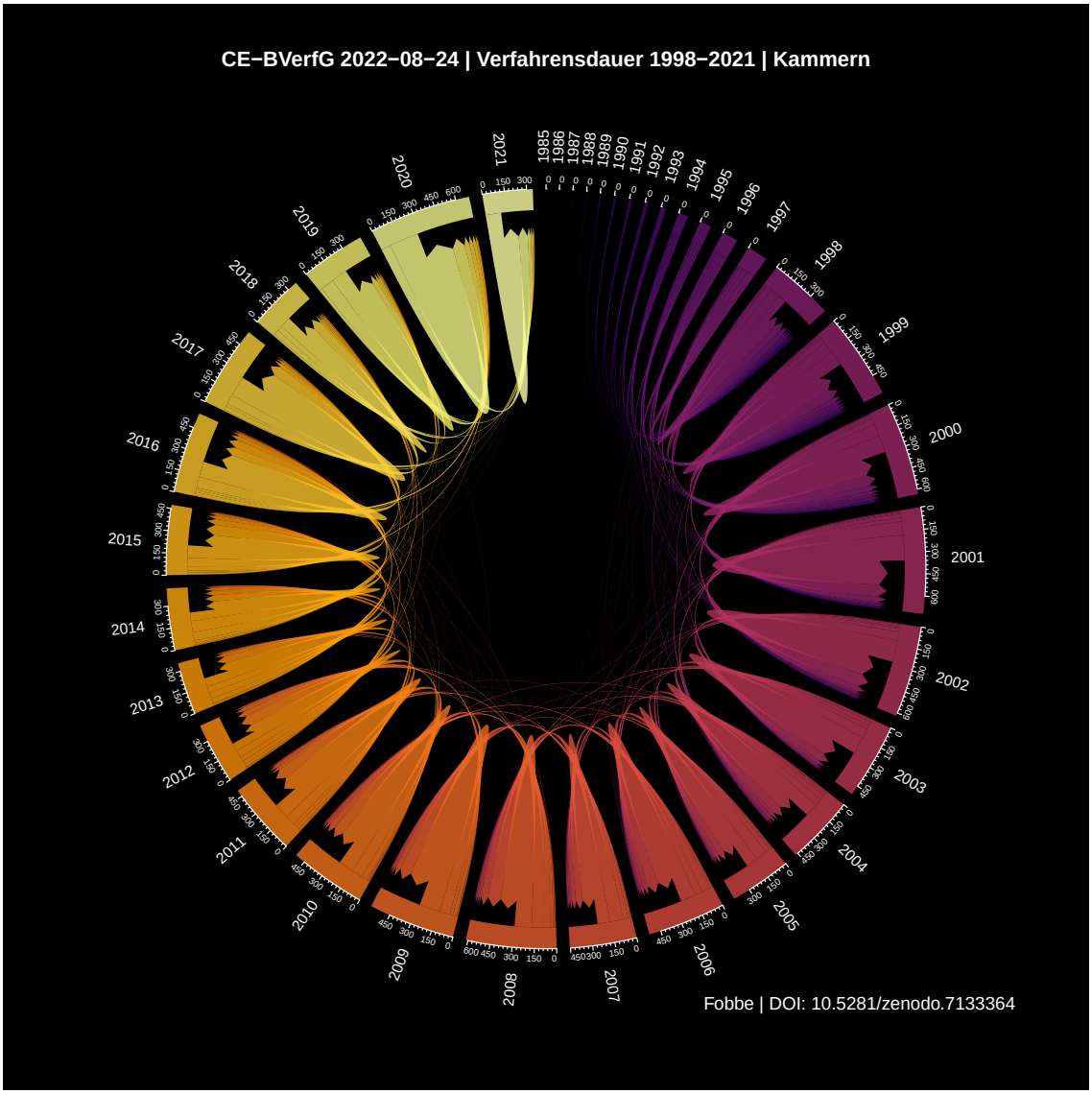



```
circleflow.dauer(Eingangsjahr = Eingangsjahr.kammer,
                 Entscheidungsjahr = Entscheidungsjahr.kammer,
                 grenze = 2021,
                 colorscheme = "dark",
                 palette = "inferno",
                 alpha = 0.8,
                 ticks = 150)
```

```
## Warning in get.adjacency(g, edges = F): The `edges` argument of
## `as_adjacency_matrix` is deprecated; it will be removed in igraph 1.4.0
```

```
title(main = "CE-BVerfG 2022-08-24 | Verfahrensdauer 1998-2021 | Kammern")
```

```
legend("bottomright", legend = caption, bty = "n")
```



8 Validierung der Genauigkeit

Durch die Verwendung allein der Eingangs- und Entscheidungsjahre geht Genauigkeit verloren. Es folgen daher einige Monte Carlo-Simulationen um zu zeigen, dass die Genauigkeit für das arithmetische Mittel dennoch sehr hoch bleibt, auch bei stark schiefen (asymmetrischen) Verteilungen der Verfahrensdauern. Die Genauigkeit für den Median ist immernoch gut genug um nützlich zu sein.

8.1 Funktion definieren: Monte Carlo-Simulation der Abweichungen

Diese Funktion generiert für N synthethische Verfahren aus dem gewählten Zeitraum ein zufälliges Eingangsdatum und eine zufällige Verfahrensdauer (in Tagen) entsprechend einer vorgegebenen Wahrscheinlichkeitsverteilung.

Dann werden die Daten wie oben beschrieben auf Eingangsjahre und Entscheidungsjahre reduziert und die Verfahrensdauer (in Jahren) berechnet.

Schließlich berechnet die Funktion für die wahren und geschätzten Verfahrensdauern jeweils das arithmetische Mittel, den Median und die Summe. Die Abweichung des wahren vom geschätzten Wert wird absolut (in Tagen) und relativ wiedergegeben.

Achtung: es handelt sich hier um eine Zufalls-Simulation. Die Änderung des Seeds ändert auch die Ergebnisse, idR aber nur geringfügig.

```
f.errormargin <- function(distr.expr = "rgamma(N, shape = 1, rate = 0.6)",
                          N = 1000,
                          date.min = "1980-01-01",
                          date.max = "2020-12-31",
                          seed = 23452345){

    set.seed(seed)

    dates.all <- as.Date(date.min):as.Date(date.max)
    dates.begin <- as.Date(sample(dates.all,
                                  N,
                                  replace = TRUE),
                           origin = "1970-01-01")

    distr <- ceiling(eval(parse(text = distr.expr)) * 365)

    dates.end <- dates.begin + distr

    year.begin <- year(dates.begin)
    year.end <- year(dates.end)

    duration.real.days <- as.integer(dates.end - dates.begin)
    duration.real.years <- duration.real.days / 365.25

    duration.estimate <- year.end - year.begin

    ## Absolute Abweichung der Mittelwerte (in Tagen)
    dev.days.mean <- mean(duration.real.days) - (mean(duration.estimate) *
    365.25)
```

```
    ## Absolute Abweichung der Mediane (in Tagen)
    dev.days.median <- median(duration.real.days) - (median(duration.estimate) *
    365.25)

    ## Absolute Abweichung der Summen (in Tagen)
    dev.days.sum <- sum(duration.real.days) - (sum(duration.estimate) * 365.25)

    ## Relative Abweichungen der Mittelwerte
    dev.rel.mean <- (mean(duration.real.years) - mean(duration.estimate)) / mean(
    duration.real.years)

    ## Relative Abweichungen der Mediane
    dev.rel.median <- (median(duration.real.years) - median(duration.estimate)) /
     median(duration.real.years)

    ## Relative Abweichung der Summen
    dev.rel.sum <- (sum(duration.real.years) - sum(duration.estimate)) / sum(
    duration.real.years)

    dt <- data.table(N,
                     distr.expr,
                     date.min,
                     date.max,
                     dev.days.mean,
                     dev.days.median,
                     dev.days.sum,
                     dev.rel.mean,
                     dev.rel.median,
                     dev.rel.sum)

    return(dt)

}
```

8.2 Gamma-Verteilung

Die Gamma-Verteilung ist eine stetige Wahrscheinlichkeitsverteilung mit zwei Parametern. Sie ist in R durch einen Shape und einen Rate-Parameter definiert.

Die Gamma-Verteilung kann mit entsprechenden Parameter-Werten eine asymmetrische Verteilung bilden, die der beobachteten Verteilung von BVerfG-Verfahrensdauern nahe kommt. Als Paramter-Raum wählen ich alle Kombinationen des Shape-Parameters von 1 bis 3 und dem Rate-Parameter von 0.1 bis 0.9.

Ein Shape-Paramter von 1 oder 2 mit jeweils einem Rate-Parameter ab 0.6 wirken überzeugend, wenn alle Entscheidungen des BVerfG in Frage kommen. Höhere Werte des Shape-Parameters sind hier unplausibel, da die meisten Verfahren in unter einem Jahr entschieden werden. Ein Shape-Parameter von 3 und einem Rate-Parameter ab 0.6 kommt in Frage, wenn nur Senats-Entscheidungen betrachtet werden.

Ich wähle eine Stichprobe von 1100, da in etwa soviele Senats-Entscheidungen im Korpus vorhanden sind.

Im Ergebnis sehen wir, dass der geschätzte Mittelwert bei allen Parameter-Sätzen nur um

wenige Tage vom wahren Wert abweicht. Der Mittelwert ist also in diesen Fällen eine sehr genaue und sinnvolle Schätzgröße.

Der Median ist weniger genau. Im interessanten Parameter-Raum beträgt der Messfehler zwischen einem und vier Monaten. Der Median bleibt ein sinnvoller Schätzwert, ist aber mit einer spürbaren Ungenauigkeit behaftet.

Die Ungenauigkeit bei den Summen beträgt weniger als 1%, ist also akzeptabel. Der absolute Wert ist nur zur Vollständigkeit enthalten, dieser steigt mit steigendem N monoton an.

Die Ergebnisse sind plausibel. Der maximale theoretische Messfehler beträgt zwischen plus/minus 1 Jahr bei einer maximal asymmetrischen Verteilung. Bei einer vollständig symmetrischen Verteilung (z.B. Normalverteilung) wäre er bei 0. Die hier betrachteten teilweise asymmetrischen Gamma-Verteilungen liegen innerhalb dieses Spektrums. ### Parameter-Raum definieren

```
shape  <- 1:3

rate <- seq(0.1, 0.9, 0.1)


combinations <- expand.grid(shape, rate)
setDT(combinations)
setnames(combinations, new = c("shape", "rate"))
combinations <- combinations[order(shape, rate)]

print(combinations)
```

```
##     shape rate
##  1:     1  0.1
##  2:     1  0.2
##  3:     1  0.3
##  4:     1  0.4
##  5:     1  0.5
##  6:     1  0.6
##  7:     1  0.7
##  8:     1  0.8
##  9:     1  0.9
## 10:     2  0.1
## 11:     2  0.2
## 12:     2  0.3
## 13:     2  0.4
## 14:     2  0.5
## 15:     2  0.6
## 16:     2  0.7
## 17:     2  0.8
## 18:     2  0.9
## 19:     3  0.1
## 20:     3  0.2
## 21:     3  0.3
## 22:     3  0.4
## 23:     3  0.5
## 24:     3  0.6
```

```
## 25:     3  0.7
## 26:     3  0.8
## 27:     3  0.9
##     shape rate
```

8.2.1 Exakte Abweichungen berechnen

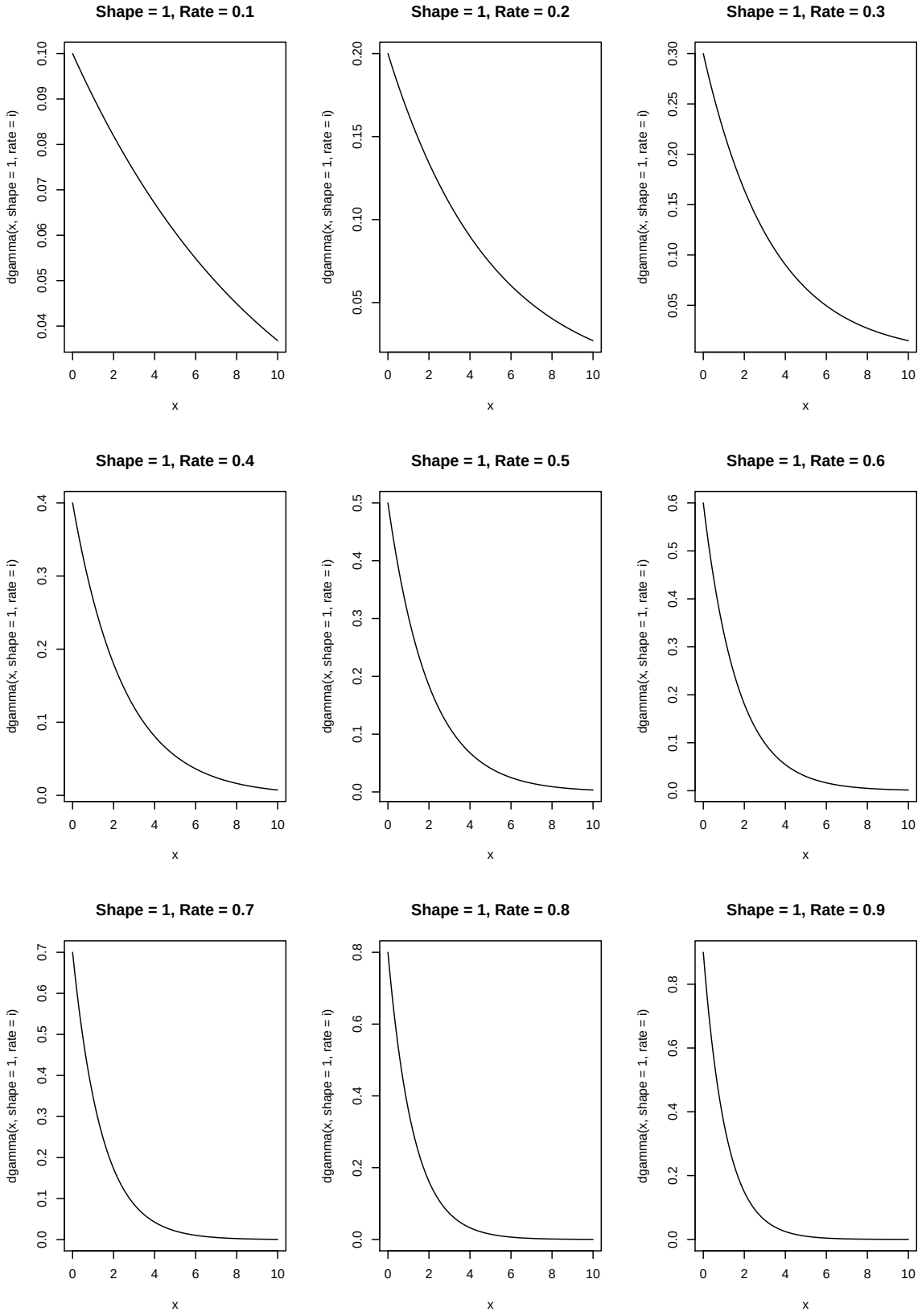
```
distrs.gamma <- paste0("rgamma(N, ",
                       "shape = ",
                       combinations$shape,
                       ", rate = ",
                       combinations$rate,
                       ")")


list <- lapply(distrs.gamma,
               f.errormargin,
               N = 1100)


rbindlist(list)
```

```
##        N                       distr.expr   date.min   date.max dev.days.mean
##  1: 1100 rgamma(N, shape = 1, rate = 0.1) 1980-01-01 2020-12-31   -2.64772727
##  2: 1100 rgamma(N, shape = 1, rate = 0.2) 1980-01-01 2020-12-31   -2.06090909
##  3: 1100 rgamma(N, shape = 1, rate = 0.3) 1980-01-01 2020-12-31   -2.98545455
##  4: 1100 rgamma(N, shape = 1, rate = 0.4) 1980-01-01 2020-12-31    1.20363636
##  5: 1100 rgamma(N, shape = 1, rate = 0.5) 1980-01-01 2020-12-31   -1.50977273
##  6: 1100 rgamma(N, shape = 1, rate = 0.6) 1980-01-01 2020-12-31    3.41409091
##  7: 1100 rgamma(N, shape = 1, rate = 0.7) 1980-01-01 2020-12-31   -4.07068182
##  8: 1100 rgamma(N, shape = 1, rate = 0.8) 1980-01-01 2020-12-31    0.17954545
##  9: 1100 rgamma(N, shape = 1, rate = 0.9) 1980-01-01 2020-12-31    1.33954545
## 10: 1100 rgamma(N, shape = 2, rate = 0.1) 1980-01-01 2020-12-31   -2.31886364
## 11: 1100 rgamma(N, shape = 2, rate = 0.2) 1980-01-01 2020-12-31   -1.40340909
## 12: 1100 rgamma(N, shape = 2, rate = 0.3) 1980-01-01 2020-12-31   -6.31181818
## 13: 1100 rgamma(N, shape = 2, rate = 0.4) 1980-01-01 2020-12-31    1.04977273
## 14: 1100 rgamma(N, shape = 2, rate = 0.5) 1980-01-01 2020-12-31   -2.05931818
## 15: 1100 rgamma(N, shape = 2, rate = 0.6) 1980-01-01 2020-12-31   -3.23295455
## 16: 1100 rgamma(N, shape = 2, rate = 0.7) 1980-01-01 2020-12-31    1.38000000
## 17: 1100 rgamma(N, shape = 2, rate = 0.8) 1980-01-01 2020-12-31   -1.04272727
## 18: 1100 rgamma(N, shape = 2, rate = 0.9) 1980-01-01 2020-12-31   -6.73977273
## 19: 1100 rgamma(N, shape = 3, rate = 0.1) 1980-01-01 2020-12-31   -3.31727273
## 20: 1100 rgamma(N, shape = 3, rate = 0.2) 1980-01-01 2020-12-31   -0.08227273
## 21: 1100 rgamma(N, shape = 3, rate = 0.3) 1980-01-01 2020-12-31   -2.77500000
## 22: 1100 rgamma(N, shape = 3, rate = 0.4) 1980-01-01 2020-12-31   -2.77681818
## 23: 1100 rgamma(N, shape = 3, rate = 0.5) 1980-01-01 2020-12-31    6.44295455
## 24: 1100 rgamma(N, shape = 3, rate = 0.6) 1980-01-01 2020-12-31    1.18318182
## 25: 1100 rgamma(N, shape = 3, rate = 0.7) 1980-01-01 2020-12-31   -0.08613636
## 26: 1100 rgamma(N, shape = 3, rate = 0.8) 1980-01-01 2020-12-31   -5.13659091
## 27: 1100 rgamma(N, shape = 3, rate = 0.9) 1980-01-01 2020-12-31    3.50659091
##        N                       distr.expr   date.min   date.max dev.days.mean
##     dev.days.median dev.days.sum  dev.rel.mean dev.rel.median   dev.rel.sum
```

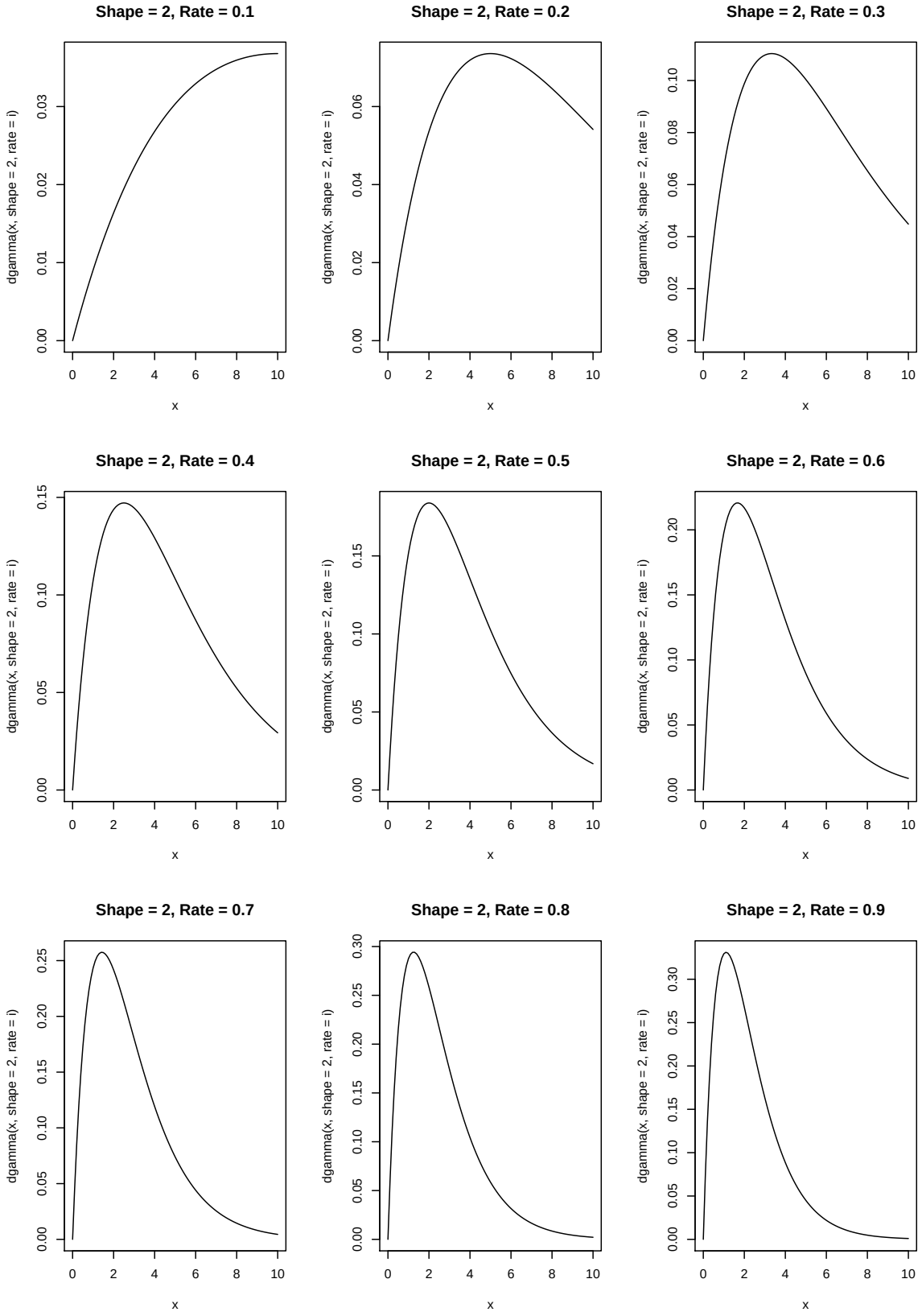
```
##  1:          176.00     -2912.50 -7.258303e-04   0.0743400211 -7.258303e-04
##  2:           88.25     -2267.00 -1.129767e-03   0.0745354730 -1.129767e-03
##  3:           59.00     -3284.00 -2.454567e-03   0.0747308423 -2.454567e-03
##  4:         -138.00      1324.00  1.319290e-03  -0.2329113924  1.319290e-03
##  5:          108.75     -1660.75 -2.068220e-03   0.2294303797 -2.068220e-03
##  6:           29.75      3755.50  5.611581e-03   0.0753164557  5.611581e-03
##  7:          -26.25     -4477.75 -7.804861e-03  -0.0774336283 -7.804861e-03
##  8:          -68.75       197.50  3.933863e-04  -0.2318718381  3.933863e-04
##  9:         -101.75      1473.50  3.301236e-03  -0.3861480076  3.301236e-03
## 10:          -23.00     -2550.75 -3.253013e-04  -0.0039512111 -3.253013e-04
## 11:          -11.00     -1543.75 -3.937258e-04  -0.0037787702 -3.937258e-04
## 12:          114.75     -6943.00 -2.655990e-03   0.0591190108 -2.655990e-03
## 13:           -5.50      1154.75  5.889415e-04  -0.0037787702  5.889415e-04
## 14:           68.75     -2265.25 -1.444053e-03   0.0590382138 -1.444053e-03
## 15:         -124.75     -3556.25 -2.720247e-03  -0.1284757981 -2.720247e-03
## 16:          101.50      1518.00  1.354578e-03   0.1219951923  1.354578e-03
## 17:           -2.50     -1147.00 -1.169638e-03  -0.0034340659 -1.169638e-03
## 18:          -83.00     -7413.75 -8.504513e-03  -0.1281853282 -8.504513e-03
## 19:           -5.00     -3649.00 -3.069629e-04  -0.0005267871 -3.069629e-04
## 20:           -2.25       -90.50 -1.522547e-05  -0.0004740834 -1.522547e-05
## 21:         -123.25     -3052.50 -7.702825e-04  -0.0389538559 -7.702825e-04
## 22:         -183.25     -3054.50 -1.027665e-03  -0.0772066568 -1.027665e-03
## 23:           72.25      7087.25  2.980435e-03   0.0380563603  2.980435e-03
## 24:          121.50      1301.50  6.567637e-04   0.0767772512  6.567637e-04
## 25:         -104.50       -94.75 -5.577851e-05  -0.0770364910 -5.577851e-05
## 26:           91.25     -5650.25 -3.801309e-03   0.0768744735 -3.801309e-03
## 27:          -40.75      3857.25  2.919253e-03  -0.0386255924  2.919253e-03
##     dev.days.median dev.days.sum  dev.rel.mean dev.rel.median   dev.rel.sum
```

8.2.2 Verteilungen visualisieren

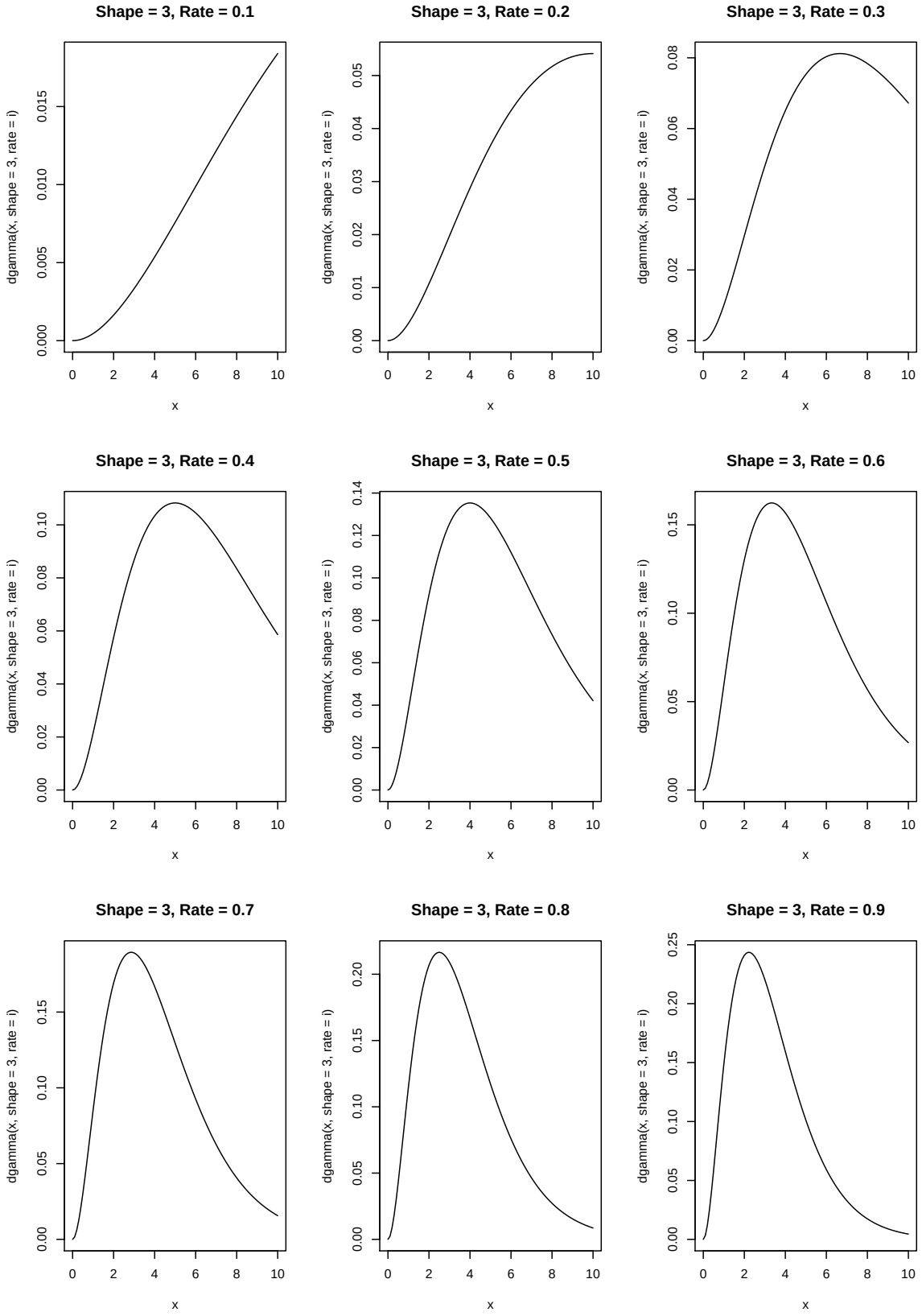
```
par(mfrow=c(3,3))

for (i in rate){
    curve(dgamma(x, shape = 1, rate = i), from = 0, to = 10,
         main = paste("Shape = 1,", "Rate =", i))

}
```


Shape = 1, Rate = 0.1
Shape = 1, Rate = 0.2
Shape = 1, Rate = 0.3
Shape = 1, Rate = 0.4
Shape = 1, Rate = 0.5
Shape = 1, Rate = 0.6
Shape = 1, Rate = 0.7
Shape = 1, Rate = 0.8
Shape = 1, Rate = 0.9
dgamma(x, shape = 1, rate = i)
x

```r
par(mfrow=c(3,3))

for (i in rate){
    curve(dgamma(x, shape = 2, rate = i), from = 0, to = 10,
         main = paste("Shape = 2,", "Rate =", i))

}
```


Shape = 2, Rate = 0.1
Shape = 2, Rate = 0.2
Shape = 2, Rate = 0.3
Shape = 2, Rate = 0.4
Shape = 2, Rate = 0.5
Shape = 2, Rate = 0.6
Shape = 2, Rate = 0.7
Shape = 2, Rate = 0.8
Shape = 2, Rate = 0.9
dgamma(x, shape = 2, rate = i)
x

```r
par(mfrow=c(3,3))

for (i in rate){
    curve(dgamma(x, shape = 3, rate = i), from = 0, to = 10,
         main = paste("Shape = 3,", "Rate =", i))

}
```


Shape = 3, Rate = 0.1
Shape = 3, Rate = 0.2
Shape = 3, Rate = 0.3
Shape = 3, Rate = 0.4
Shape = 3, Rate = 0.5
Shape = 3, Rate = 0.6
Shape = 3, Rate = 0.7
Shape = 3, Rate = 0.8
Shape = 3, Rate = 0.9
dgamma(x, shape = 3, rate = i)
x

8.3 Beta-Verteilung

Die Beta-Verteilung ist eine stetige Wahrscheinlichkeitsverteilung mit zwei Shape-Parametern, die im Intervall von 0 bis 1 definiert ist. Ich multipliziere die Verteilungen jeweils mit 10, damit die Verteilung als Dauer in Jahren interpretierbar ist.

Wenn ein Parameter konstant gehalten und der andere erhöht wird, lässt sich eine asymmetrische Verteilung erstellen, die der beobachteten Verteilung von BVerfG-Verfahrensdauern nahe kommt. Die vermutlich besten Parameter-Werte bei konstantem Shape1 = 1 sind Werte von 5 bis 7 für Shape2.

Ich wähle eine Stichprobe von 1100, da in etwa soviele Senats-Entscheidungen im Korpus vorhanden sind.

Im Ergebnis sehen wir, dass der geschätzte Mittelwert bei allen Parameter-Sätzen nur um wenige Tage vom wahren Wert abweicht. Der Mittelwert ist also in diesen Fällen eine sehr genaue und sinnvolle Schätzgröße.

Der Median ist weniger genau. Im interessanten Parameter-Raum beträgt der Messfehler auch hier zwischen einem und vier Monaten. Der Median bleibt ein sinnvoller Schätzwert, ist aber mit einer spürbaren Ungenauigkeit behaftet.

Die Ungenauigkeit bei den Summen beträgt ungefähr 1% oder weniger, ist also akzeptabel. Der absolute Wert ist nur zur Vollständigkeit enthalten, dieser steigt mit steigendem N monoton an.

Die Ergebnisse sind plausibel. Der maximale theoretische Messfehler beträgt zwischen plus/minus 1 Jahr bei einer maximal asymmetrischen Verteilung. Bei einer vollständig symmetrischen Verteilung (z.B. Normalverteilung) wäre er bei 0. Die hier betrachteten teilweise asymmetrischen Beta-Verteilungen liegen innerhalb dieses Spektrums. ### Parameter-Raum definieren

```
shape1  <- 1 # alpha

shape2 <- 1:9 # beta


combinations <- expand.grid(shape1, shape2)
setDT(combinations)
setnames(combinations, new = c("shape1", "shape2"))
combinations <- combinations[order(shape1, shape2)]

print(combinations)
```

```
##    shape1 shape2
## 1:      1      1
## 2:      1      2
## 3:      1      3
## 4:      1      4
## 5:      1      5
## 6:      1      6
## 7:      1      7
## 8:      1      8
```

```
## 9:      1      9
```

8.3.1 Exakte Abweichungen berechnen

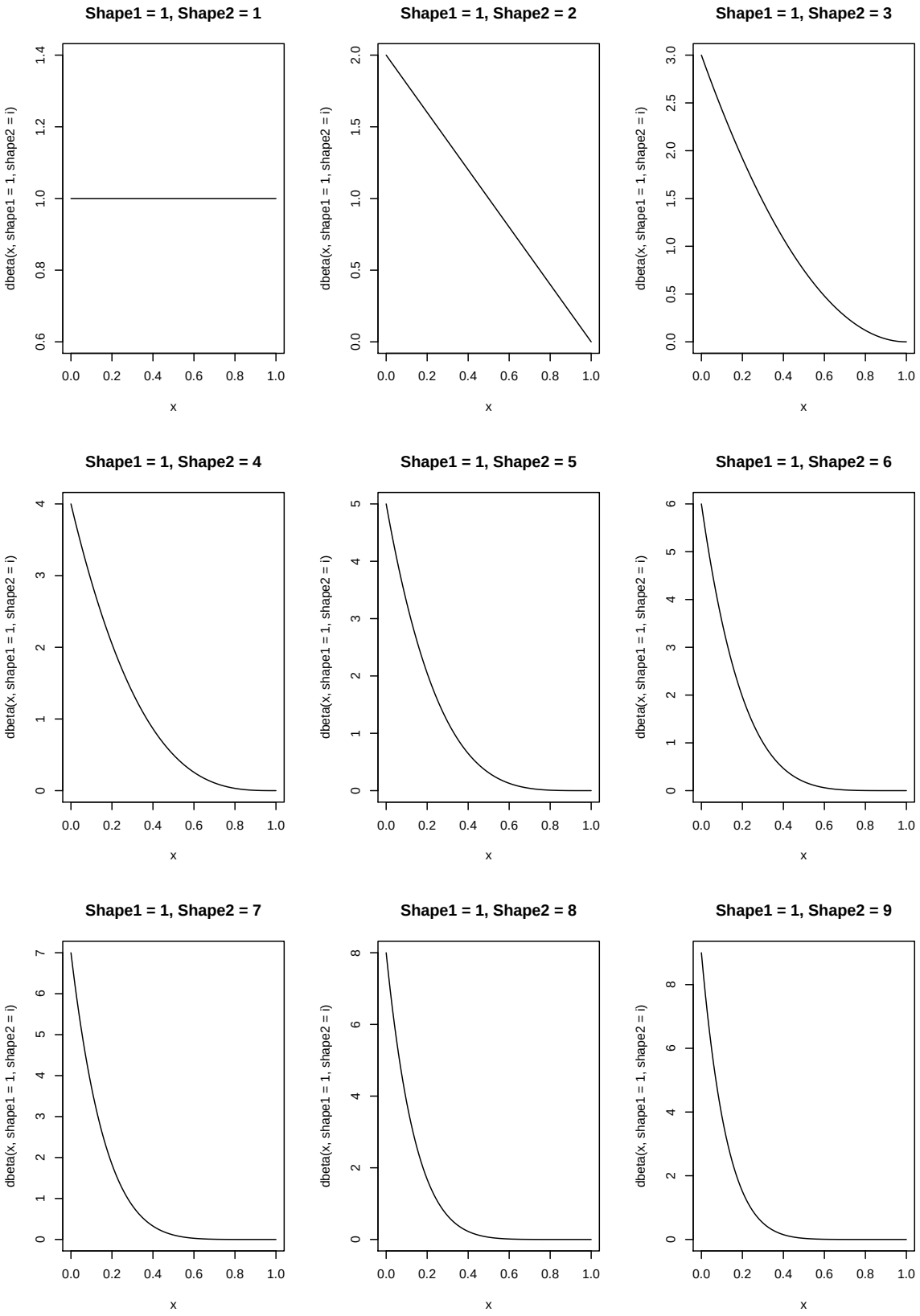
Hinweis: die Beta-Verteilungen werden jeweils mit 10 multipliziert um einer Verfahrensdauer in Jahren möglichst nahezukommen.

```
distrs.beta <- paste0("rbeta(N, ",
                      "shape1 = ",
                      combinations$shape1,
                      ", shape2 = ",
                      combinations$shape2,
                      ") * 10")


list <- lapply(distrs.beta,
               f.errormargin,
               N = 1100)


rbindlist(list)
```

```
##       N                              distr.expr   date.min   date.max
## 1: 1100 rbeta(N, shape1 = 1, shape2 = 1) * 10 1980-01-01 2020-12-31
## 2: 1100 rbeta(N, shape1 = 1, shape2 = 2) * 10 1980-01-01 2020-12-31
## 3: 1100 rbeta(N, shape1 = 1, shape2 = 3) * 10 1980-01-01 2020-12-31
## 4: 1100 rbeta(N, shape1 = 1, shape2 = 4) * 10 1980-01-01 2020-12-31
## 5: 1100 rbeta(N, shape1 = 1, shape2 = 5) * 10 1980-01-01 2020-12-31
## 6: 1100 rbeta(N, shape1 = 1, shape2 = 6) * 10 1980-01-01 2020-12-31
## 7: 1100 rbeta(N, shape1 = 1, shape2 = 7) * 10 1980-01-01 2020-12-31
## 8: 1100 rbeta(N, shape1 = 1, shape2 = 8) * 10 1980-01-01 2020-12-31
## 9: 1100 rbeta(N, shape1 = 1, shape2 = 9) * 10 1980-01-01 2020-12-31
##    dev.days.mean dev.days.median dev.days.sum  dev.rel.mean dev.rel.median
## 1:      1.704318          -19.75      1874.75  0.0009508447    -0.01093274
## 2:     -2.017045          -31.75     -2218.75 -0.0016720978    -0.02984023
## 3:     -4.152045           20.00     -4567.25 -0.0046167508     0.02664890
## 4:     -4.717727         -138.00     -5189.50 -0.0064705934    -0.23291139
## 5:     -2.667727          118.75     -2934.50 -0.0043523142     0.24535124
## 6:     -5.752045           43.25     -6327.25 -0.0108835317     0.10587515
## 7:     -2.500455          -13.25     -2750.50 -0.0054471063    -0.03764205
## 8:     -5.588636          -56.25     -6147.50 -0.0136929396    -0.18203883
## 9:      0.550000          -88.25       605.00  0.0014927141    -0.31859206
##      dev.rel.sum
## 1:  0.0009508447
## 2: -0.0016720978
## 3: -0.0046167508
## 4: -0.0064705934
## 5: -0.0043523142
## 6: -0.0108835317
## 7: -0.0054471063
## 8: -0.0136929396
## 9:  0.0014927141
```

8.3.2 Verteilungen visualisieren

```
par(mfrow=c(3,3))

for (i in shape2){
    curve(dbeta(x, shape1 = 1, shape2 = i),
         main = paste("Shape1 = 1,", "Shape2 =", i))

}
```


Shape1 = 1, Shape2 = 1
Shape1 = 1, Shape2 = 2
Shape1 = 1, Shape2 = 3
Shape1 = 1, Shape2 = 4
Shape1 = 1, Shape2 = 5
Shape1 = 1, Shape2 = 6
Shape1 = 1, Shape2 = 7
Shape1 = 1, Shape2 = 8
Shape1 = 1, Shape2 = 9
dbeta(x, shape1 = 1, shape2 = i)
x

9 ZIP-Archive erstellen

9.1 Verpacken der Diagramme

```
zip(paste0(prefix.files,
           "_",
           "Diagramme.zip"),
    list.files(fig.dir, full.names = TRUE),
    mode = "cherry-pick")
```

9.2 Verpacken der Entscheidungstexte

```
zip(paste0(prefix.files,
           "_",
           "Entscheidungstexte_Verfahrensdauer-über-10-Jahre",
           ".zip"),
    "output/Entscheidungstexte_Verfahrensdauer-über-10-Jahre",
    mode = "cherry-pick")
```

9.3 Verpacken der Source-Dateien

```
files.source <- c(list.files(pattern = "\\.R$|\\.toml$|\\.R?md"),
                  "buttons",
                  "tex",
                  list.files(pattern = "renv\\.lock|\\.Rprofile",
                             all.files = TRUE),
                  list.files("renv",
                             pattern = "activate\\.R",
                             full.names = TRUE))




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

zip(paste0(prefix.files,
           "_",
           "Source_Code.zip"),
    files.source)
```

10 Abschluss

10.1 Datum und Uhrzeit (Anfang)

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

```
## [1] "2022-12-15 18:45:02 CET"
```

10.2 Datum und Uhrzeit (Ende)

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

```
## [1] "2022-12-15 18:49:25 CET"
```

10.3 Laufzeit des gesamten Skriptes

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

```
## Time difference of 4.375576 mins
```

10.4 Warnungen

```
warnings()
```

11 Parameter für strenge Replikationen

```
sessionInfo()
```

```
## R version 4.0.4 (2021-02-15)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Debian GNU/Linux 11 (bullseye)
##
## 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-r0.3.13.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 datasets  utils     methods   base
##
## other attached packages:
##  [1] data.table_1.14.2 ggplot2_3.3.6     zip_2.2.0         circlize_0.4.15
##  [5] igraph_1.3.4      viridis_0.6.2     viridisLite_0.4.0 knitr_1.39
##  [9] kableExtra_1.3.4  RcppTOML_0.1.7    rmarkdown_2.14
##
## loaded via a namespace (and not attached):
##  [1] shape_1.4.6          xfun_0.35            lattice_0.20-41
##  [4] colorspace_2.0-3     vctrs_0.5.1          htmltools_0.5.4
##  [7] yaml_2.3.5           utf8_1.2.2           rlang_1.0.6
## [10] pillar_1.8.0         glue_1.6.2           withr_2.5.0
## [13] lifecycle_1.0.3      stringr_1.5.0        munsell_0.5.0
## [16] gtable_0.3.0         rvest_1.0.2          GlobalOptions_0.1.2
## [19] evaluate_0.15        labeling_0.4.2       fastmap_1.1.0
## [22] fansi_1.0.3          highr_0.9            Rcpp_1.0.9
## [25] renv_0.15.5          scales_1.2.0         webshot_0.5.3
## [28] magick_2.7.3         farver_2.1.1         systemfonts_1.0.4
## [31] gridExtra_2.3        digest_0.6.31        stringi_1.7.8
## [34] grid_4.0.4           cli_3.4.1            tools_4.0.4
## [37] magrittr_2.0.3       tibble_3.1.7         pkgconfig_2.0.3
## [40] ellipsis_0.3.2       Matrix_1.3-2         xml2_1.3.3
## [43] svglite_2.1.0        httr_1.4.3           rstudioapi_0.13
## [46] R6_2.5.1             compiler_4.0.4
```

Literaturverzeichnis

Allaire, JJ, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone. 2022. *Rmarkdown: Dynamic Documents for R.* https://CRAN.R-project.org/package=rmarkdown.

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. 2021. *Zip: Cross-Platform Zip Compression.* https://github.com/r-lib/zip#readme.

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

Eddelbuettel, Dirk. 2020. *RcppTOML: Rcpp Bindings to Parser for Tom's Obvious Markup Language.* http://dirk.eddelbuettel.com/code/rcpp.toml.html.

file., See AUTHORS. 2022. *Igraph: Network Analysis and Visualization.* https://CRAN.R-project.org/package=igraph.

Garnier, Simon. 2021a. *Viridis: Colorblind-Friendly Color Maps for R.* https://CRAN.R-project.org/package=viridis.

———. 2021b. *ViridisLite: Colorblind-Friendly Color Maps (Lite Version).* https://CRAN.R-project.org/package=viridisLite.

Gu, Zuguang. 2022. *Circlize: Circular Visualization.* https://CRAN.R-project.org/package=circlize.

Gu, Zuguang, Lei Gu, Roland Eils, Matthias Schlesner, and Benedikt Brors. 2014. "Circlize Implements and Enhances Circular Visualization in R." *Bioinformatics* 30 (19): 2811–2.

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

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. 2022. *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/.

———. 2022. *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.