Das csvsimple-Paket

Version 1.03 (2011/11/04)

Thomas F. Sturm[1]

Abstract

csvsimple provides a simple LaTeX interface for the processing of files with comma separated values (CSV). csvsimple relies heavily on the key value syntax from pgfkeys which results (hopefully) in an easy way of usage. Filtering and table generation is especially supported. Since the package is considered as a lightweight tool, there is no support for data sorting or data base storage.

Inhaltsverzeichnis

1 Einführung

Das csvsimple-Paket dient zur Verarbeitung von CSV[2]-Dateien mit einer einfachen Schnittstelle. Die Verarbeitung wird hauptsächlich durch Schlüssel-Wert-Zuweisungen mit der pgfkeys-Syntax gesteuert. Als Anwendungen kommen z. B. tabellarische Listen, Serienbriefe oder Diagramme in Frage.

Eine Alternative zu csvsimple ist das csvtool-Paket bzw. das datatool-Paket, welche wesentlich mehr Funktionen bieten wie z. B. Änderung der Separatoren und Begrenzer in der CSV-Datei oder Sortierung der Datenzeilen. Im Gegensatz dazu beschränkt sich csvsimple auf wenige Grundfunktionen und ist damit ressourcenschonender.

[1]Prof. Dr. Dr. Thomas F. Sturm, WE Mathematik und Informatik, Universität der Bundeswehr München, 85577 Neubiberg, Germany; email: thomas.sturm@unibw.de

[2]CSV-Datei: Datei mit kommaseparierten Werten.

Jede Zeile einer verarbeitbaren CSV-Datei muss aus der identischen Anzahl von kommaseparierten Werten bestehen, wobei die TEX-Gruppenklammern `{}` verwendet werden können, um einen Bereich als Block zu markieren, der dann auch nicht zu verarbeitende Kommas enthalten kann.
Die erste Zeile einer solchen CSV-Datei ist üblicherweise (aber nicht notwendig) eine Überschriftzeile, die die Bezeichnungen der Spalten enthält.

CSV-Datei `noten.csv`

```
Name,Vorname,Matrikelnummer,Geschlecht,Note
Maier,Hans,12345,m,1.0
Huber,Anna,23456,w,2.3
Weißbäck,Werner,34567,m,5.0
```

Die „billigste“ Methode, die CSV-Datei tabellarisch darzustellen, ist Verarbeitung mit dem `\csvautotabular`-Befehl.

```
\csvautotabular{noten.csv}
```

Name	Vorname	Matrikelnummer	Geschlecht	Note
Maier	Hans	12345	m	1.0
Huber	Anna	23456	w	2.3
Weißbäck	Werner	34567	m	5.0

In der Regel wird man nicht `\csvautotabular` auf Seite 6, sondern `\csvreader` auf Seite 5 verwenden. Durch eine Zuordnungsvorschrift lassen sich die Spaltenbezeichnungen aus der Überschriftzeile eigenen Makros zuordnen, die man dann frei verwenden kann.

```
\begin{tabular}{|l|c|}\hline%
\bfseries Person & \bfseries Matr.~Nr.
\csvreader{noten.csv}{Name=\Name,Vorname=\Vorname,Matrikelnummer=\Nummer}%
  {\\\Vorname~\Name & \Nummer}%
  \\\hline
\end{tabular}
```

Person	**Matr. Nr.**
Hans Maier	12345
Anna Huber	23456
Werner Weißbäck	34567

`\csvreader` kann über Optionen beeinflusst werden. In Tabellen kann man z. B. Zeilenumbrüche und Linien am bequemsten über `late after line` steuern. Dies definiert eine Ausführung knapp vor Ausgabe der nachfolgenden Zeile.

```
\begin{tabular}{|r|l|c|}\hline%
& Person & Matr.~Nr.\\\hline\hline
\csvreader[late after line=\\\hline]%
  {noten.csv}{Name=\Name,Vorname=\Vorname,Matrikelnummer=\Nummer}%
  {\thecsvrow & \Vorname~\Name & \Nummer}%
\end{tabular}
```

	Person	Matr. Nr.
1	Hans Maier	12345
2	Anna Huber	23456
3	Werner Weißbäck	34567

Die gesamte Erzeugung der Tabelle kann auch direkt als Option gesetzt werden:

```
\csvreader[tabular=|r|l|c|,
    table head=\hline & Person & Matr.~Nr.\\\hline\hline,
    late after line=\\\hline]%
  {noten.csv}{Name=\Name,Vorname=\Vorname,Matrikelnummer=\Nummer}%
  {\thecsvrow & \Vorname~\Name & \Nummer}%
```

	Person	Matr. Nr.
1	Hans Maier	12345
2	Anna Huber	23456
3	Werner Weißbäck	34567

Bei wiederkehrenden Anwendungen kann man über die `pgfkeys`-Syntax eigene Stilvereinbarungen vornehmen, die eine einheitliche und zentrale Gestaltung erlauben. Das nachfolgende Beispiel lässt sich leicht durch noch mehr oder weniger Vereinbarungen modifizieren.

```
\csvset{meine Personenliste/.style={%
  tabular=|r|l|c|,
  table head=\hline & Person & #1\\\hline\hline,
  late after line=\\\hline,
  column names={Name=\Name,Vorname=\Vorname}
  }}

\csvreader[meine Personenliste={Matr.~Nr.}]{noten.csv}{Matrikelnummer=\Nummer}%
  {\thecsvrow & \Vorname~\Name & \Nummer}%
\hfill%
\csvreader[meine Personenliste={Ergebnis}]{noten.csv}{Note=\Note}%
  {\thecsvrow & \Vorname~\Name & \Note}%
```

	Person	Matr. Nr.
1	Hans Maier	12345
2	Anna Huber	23456
3	Werner Weißbäck	34567

	Person	Ergebnis
1	Hans Maier	1.0
2	Anna Huber	2.3
3	Werner Weißbäck	5.0

Als Alternative kann zur Vereinbarung der Spaltennamen der Abkürzungsbefehl \csvnames auf Seite 7 und für Stilvereinbarungen der Abkürzungsbefehl \csvstyle auf Seite 6 eingesetzt werden. Das Beispiel sieht dann wie folgt aus:

```
\csvnames{meine Namen}{Name=\Name,Vorname=\Vorname}
\csvstyle{meine Personenliste}{tabular=|r|l|c|,
  table head=\hline & Person & #1\\\hline\hline,
  late after line=\\\hline, meine Namen}

\csvreader[meine Personenliste={Matr.~Nr.}]{noten.csv}{Matrikelnummer=\Nummer}%
  {\thecsvrow & \Vorname~\Name & \Nummer}%
\hfill%
\csvreader[meine Personenliste={Ergebnis}]{noten.csv}{Note=\Note}%
  {\thecsvrow & \Vorname~\Name & \Note}%
```

	Person	Matr. Nr.
1	Hans Maier	12345
2	Anna Huber	23456
3	Werner Weißbäck	34567

	Person	Ergebnis
1	Hans Maier	1.0
2	Anna Huber	2.3
3	Werner Weißbäck	5.0

Die Einträge der CSV-Datei können auch gefiltert werden. Im nachfolgenden Beispiel wird ein Schein für alle Personen ausgestellt, die eine Prüfung bestanden haben.

```
\csvreader[filter not equal={\Note}{5.0}]%
  {noten.csv}{Name=\Name,Vorname=\Vorname,Matrikelnummer=\Nummer,
              Note=\Note,Geschlecht=\gender}%
  {\begin{center}\Large\bfseries Schein in Mathematik\end{center}
  \large\ifthenelse{\equal{\gender}{w}}{Frau}{Herr}
  \Vorname~\Name, Matrikelnummer \Nummer, hat die Prüfung in Mathematik
  mit der Note \Note\ bestanden.\par\ldots\par
  }%
```

Schein in Mathematik

Herr Hans Maier, Matrikelnummer 12345, hat die Prüfung in Mathematik mit der Note 1.0 bestanden.

...

Schein in Mathematik

Frau Anna Huber, Matrikelnummer 23456, hat die Prüfung in Mathematik mit der Note 2.3 bestanden.

...

2 Makros zur Verarbeitung von CSV-Dateien

`\csvreader[`⟨*Schlüsselliste*⟩`]{`⟨*Dateiname*⟩`}{`⟨*Zuweisungsliste*⟩`}{`⟨*Befehlsfolge*⟩`}`

Das Kommando `\csvreader` liest die Datei ⟨*Dateiname*⟩ zeilenweise ein. Jede Zeile der Datei muss aus der gleichen Anzahl von kommaseparierten Werten bestehen, wobei die TEX-Gruppenklammern `{}` verwendet werden können, um einen Bereich als Block zu markieren, der dann auch nicht zu verarbeitende Kommas enthalten kann.

Die erste Zeile einer solchen CSV-Datei darf vorzugsweise eine Überschriftzeile sein. Die Einträge dieser Zeile können in der ⟨*Zuweisungsliste*⟩ verwendet werden, um Spalten zu TEX-Makros zuzuordnen. Die Anzahl der Einträge dieser ersten Zeile bestimmt zudem die geforderte Anzahl der Einträge pro Datenzeile. Eine Datenzeile, die mehr oder weniger Einträge enthält, wird im Standardverhalten ignoriert.

Die ⟨*Zuweisungsliste*⟩ enthält kommaseparierte Schlüssel-Wert-Paare ⟨*Name*⟩=⟨*Makro*⟩. Dabei steht ⟨*Name*⟩ für einen Wert aus der Überschriftzeile der Datei und ⟨*Makro*⟩ für ein TEX-Makro, in welches jeweils der Eintrag der zugehörigen Spalte abgelegt wird.

Die ⟨*Befehlsfolge*⟩ wird für jede akzeptierte Dateizeile ausgeführt. Verwendet werden dürfen in der ⟨*Befehlsfolge*⟩:

- `\thecsvrow` bzw. der Zähler `csvrow`, der die Nummer der aktuellen Datenzeile enthält (startend bei 1).
- `\csvcoli`, `\csvcolii`, `\csvcoliii`, ..., die den jeweiligen Inhalt der Einträge der Datenzeile enthalten. Alternativ dazu:
- ⟨*Makro*⟩ aus der ⟨*Zuweisungsliste*⟩, welches eine logische Zuordnung zu einem Eintrag der Datenzeile darstellt.

Die ⟨*Befehlsfolge*⟩ wurde für die Verwendung in Tabellen vorbereitet, d.h. die Makrodefinitionen sind global. Zudem dürfen absatzüberschreitende Befehle verwendet werden. Optional kann die Verarbeitung der CSV-Datei durch eine übergebene ⟨*Schlüsselliste*⟩ gesteuert werden. Die zulässigen Schlüssel sind in Abschnitt 3 ab Seite 9 beschrieben.

```
\csvreader[tabular=|r|l|l|, table head=\hline, late after line=\\,
           late after last line=\\\hline]{noten.csv}%
  {Name=\Name,Vorname=\Vorname,Note=\Note}%
  {\Note & \Vorname~\Name & \csvcoliii}
```

1.0	Hans Maier	12345
2.3	Anna Huber	23456
5.0	Werner Weißbäck	34567

Das `\csvreader`-Kommando besteht im Wesentlichen aus dem Aufruf des `\csvloop`-Kommandos mit den Parametern

`\csvloop{`⟨*Schlüsselliste*⟩`, file=`⟨*Dateiname*⟩`, column names=`⟨*Zuweisungsliste*⟩`,`
`command=`⟨*Befehlsfolge*⟩`}`

Daher ist die Verwendung von `file` und `command` innerhalb der ⟨*Schlüsselliste*⟩ von `\csvreader` wirkungslos. Bei `\csvreader` dürfen für die ⟨*Befehlsfolge*⟩ auch Absätze verwendet werden.

\csvautotabular{⟨*Dateiname*⟩}

\csvautotabular ist ein Abkürzungsbefehl für die Verwendung des Schlüssel autotabular auf Seite 13. Das Makro liest die gesamte CSV-Datei mit ⟨*Dateiname*⟩ und formatiert diese automatisch.

```
\csvautotabular{noten.csv}
```

Name	Vorname	Matrikelnummer	Geschlecht	Note
Maier	Hans	12345	m	1.0
Huber	Anna	23456	w	2.3
Weißbäck	Werner	34567	m	5.0

\csvautolongtable{⟨*Dateiname*⟩}

\csvautolongtable ist ein Abkürzungsbefehl für die Verwendung des Schlüssel autolongtable auf Seite 13. Das Makro liest die gesamte CSV-Datei mit ⟨*Dateiname*⟩ und formatiert diese automatisch. Für die Ausführung wird das Paket longtable benötigt.

\csvloop{⟨*Schlüsselliste*⟩}

Meist wird man \csvreader auf Seite 5 statt \csvloop verwenden. \csvreader beruht aber auf dem allgemeineren \csvloop, für welches die ⟨*Schlüsselliste*⟩ verpflichtend ist. Diese ⟨*Schlüsselliste*⟩ enthält alle Vorgaben zur Verarbeitung, insbesondere muss hierüber auch der Dateiname übergeben werden.

```
\csvloop{file={noten.csv}, column names={Name=\Name}, command=\Name,
  before reading={Namensliste:\ }, late after line={{,}\ }, late after last line=.}
```

Namensliste: Maier, Huber, Weißbäck.

\csvset{⟨*Schlüsselliste*⟩}

Ausführung der übergebenen ⟨*Schlüsselliste*⟩ außerhalb von \csvreader und \csvloop. Dieses Kommando kann z. B. zur Definition eigener Stile dienen.

```
\csvset{notenliste/.style={column names={Name=\Name,Vorname=\Vorname,
          Matrikelnummer=\Nummer,Note=\Note}},
  bestanden/.style={filter not equal={\Note}{5.0}} }

Die Mathematik-Prüfung haben bestanden:
\csvreader[notenliste,bestanden]{noten.csv}{}%
  {\Vorname\ \Name\ (\Note); }%
```

Die Mathematik-Prüfung haben bestanden: Hans Maier (1.0); Anna Huber (2.3);

\csvstyle{⟨*Stilname*⟩}{⟨*Schlüsselliste*⟩}

Abkürzung für \csvset{⟨*Stilname*⟩.style={⟨*Schlüsselliste*⟩}} zur Vereinbarung eines neuen Stils.

`\csvnames{`⟨*Stilname*⟩`}{`⟨*Zuweisungsliste*⟩`}`

Abkürzung für `\csvset{`⟨*Stilname*⟩`.style={column names={`⟨*Zuweisungsliste*⟩`}}}` zur Vereinbarung weiterer Spaltenzuordnungen.

```
\csvnames{notenliste}{Name=\Name,Vorname=\Vorname,Matrikelnummer=\Nummer,Note=\Note}
\csvstyle{bestanden}{filter not equal={\Note}{5.0}}

Die Mathematik-Prüfung haben bestanden:
\csvreader[notenliste,bestanden]{noten.csv}{}%
  {\Vorname\ \Name\ (\Note); }%
```

Die Mathematik-Prüfung haben bestanden: Hans Maier (1.0); Anna Huber (2.3);

`\csvheadset{`⟨*Zuweisungsliste*⟩`}`

In Spezialfällen kann dieses Kommando zur Änderung der ⟨*Zuweisungsliste*⟩ innerhalb von `\csvreader` und `\csvloop` dienen, d. h. die Spalten-Makro-Zuordnung kann nachträglich geändert werden.

```
\csvreader{noten.csv}{}%
  { \csvheadset{Name=\n} \fbox{\n}
    \csvheadset{Vorname=\n} \ldots\ \fbox{\n}  }%
```

Maier … Hans Huber … Anna Weißbäck … Werner

`\csviffirstrow{`⟨*Dann-Befehle*⟩`}{`⟨*Sonst-Befehle*⟩`}`

Innerhalb der Befehlsfolge von `\csvreader` auf Seite 5 werden die ⟨*Dann-Befehle*⟩ ausgeführt, falls die erste Datenzeile vorliegt, anderenfalls die ⟨*Sonst-Befehle*⟩.

```
\csvreader[tabbing, table head=\hspace*{3cm}\=\kill]{noten.csv}
   {Name=\Name,Vorname=\Vorname}{%
  \Vorname~\Name \> (\csviffirstrow{Erster Eintrag!!}{nachfolgend})
}
```

Hans Maier (Erster Eintrag!!)
Anna Huber (nachfolgend)
Werner Weißbäck (nachfolgend)

`\csvifoddrow{`⟨*Dann-Befehle*⟩`}{`⟨*Sonst-Befehle*⟩`}`

Innerhalb der Befehlsfolge von `\csvreader` auf Seite 5 werden die ⟨*Dann-Befehle*⟩ ausgeführt, falls eine ungeradzahlige Datenzeile vorliegt, anderenfalls die ⟨*Sonst-Befehle*⟩.

```
\csvreader[tabular=|l|l|l|l|,
  table head=\hline\bfseries \# & \bfseries Name & \bfseries Note\\\hline,
  late after line=\\, late after last line=\\\hline]%
  {noten.csv}{Name=\Name,Vorname=\Vorname,Note=\Note}{%
  \csvifoddrow{\slshape\thecsvrow & \slshape\Name, \Vorname & \slshape\Note}%
    {\bfseries\thecsvrow & \bfseries\Name, \Vorname & \bfseries\Note}}
```

#	Name	Note
1	*Maier, Hans*	*1.0*
2	**Huber, Anna**	**2.3**
3	*Weißbäck, Werner*	*5.0*

Das `\csvifoddrow`-Kommando kann auch für gestreifte Tabellen verwendet werden:

```
% Das Beispiel benötigt das xcolor Paket
\csvreader[tabular=rlcc,
  table head=\hline\rowcolor{red!50!black}\color{white}\# & \color{white}Person
     & \color{white}Matr.~Nr. & \color{white}Note,
  late after head=\\\hline\rowcolor{yellow!50},
  late after line=\csvifoddrow{\\\rowcolor{yellow!50}}{\\\rowcolor{red!25}}]%
  {noten.csv}{Name=\Name,Vorname=\Vorname,Matrikelnummer=\Nummer,Note=\Note}%
  {\thecsvrow & \Vorname~\Name & \Nummer & \Note}%
```

#	Person	Matr. Nr.	Note
1	Hans Maier	12345	1.0
2	Anna Huber	23456	2.3
3	Werner Weißbäck	34567	5.0

Alternativ kann dafür auch `\rowcolors` aus dem `xcolor`-Paket eingesetzt werden.

```
% Das Beispiel benötigt das xcolor Paket
\csvreader[tabular=rlcc, before table=\rowcolors{2}{red!25}{yellow!50},
  table head=\hline\rowcolor{red!50!black}\color{white}\# & \color{white}Person
     & \color{white}Matr.~Nr. & \color{white}Note\\\hline,
  late after line=\\]%
  {noten.csv}{Name=\Name,Vorname=\Vorname,Matrikelnummer=\Nummer,Note=\Note}%
  {\thecsvrow & \Vorname~\Name & \Nummer & \Note}%
```

#	Person	Matr. Nr.	Note
1	Hans Maier	12345	1.0
2	Anna Huber	23456	2.3
3	Werner Weißbäck	34567	5.0

\csvfilteraccept

Alle nachfolgenden konsistenten Zeilen werden akzeptiert und verarbeitet. Dieser Befehl überschreibt alle Filtervorgaben und kann etwa in `before filter` auf Seite 9 verwendet werden, um im Zusammenspiel mit `\csvfilterreject` eine eigene Filterimplementierung vorzunehmen.

```
\csvreader[autotabular,
  before filter=\ifthenelse{\equal{\csvcoliv}{m}}{\csvfilteraccept}{\csvfilterreject}
  ]{noten.csv}{}{\csvlinetotablerow}%
```

Name	Vorname	Matrikelnummer	Geschlecht	Note
Maier	Hans	12345	m	1.0
Weißbäck	Werner	34567	m	5.0

\csvfilterreject

Alle nachfolgenden Zeilen werden ignoriert. Dieser Befehl überschreibt alle Filtervorgaben.

\csvline

Hier ist die aktuelle unzerlegte Zeile abgespeichert.

```
\csvreader[nohead, tabbing, table head=\textit{Zeile XX:}\=\kill]%
  {noten.csv}{}%
  {\textit{Zeile \thecsvrow:} \> \csvline}%
```

Zeile 1: Name,Vorname,Matrikelnummer,Geschlecht,Note
Zeile 2: Maier,Hans,12345,m,1.0
Zeile 3: Huber,Anna,23456,w,2.3
Zeile 4: Weißbäck,Werner,34567,m,5.0

`\thecsvrow`
Ausgabe der aktuellen Zeilennummer (gezählt nach akzeptierten Zeilen ohne Kopfzeile). Auf den LaTeX-Zähler `csvrow` kann auch mit den üblichen Alternativen zugegriffen werden, z. B. `\roman{csvrow}`.

`\thecsvinputline`
Ausgabe der aktuellen Dateizeilennummer (inklusive Kopfzeile). Auf den LaTeX-Zähler `csvinputline` kann auch mit den üblichen Alternativen zugegriffen werden, z. B. `\roman{csvinputline}`.

```
\csvreader[nohead, filter equal={\thecsvinputline}{3}]%
  {noten.csv}{}%
  {Die Dateizeile Nummer \thecsvinputline\ hat den Inhalt: \csvline}%
```

Die Dateizeile Nummer 3 hat den Inhalt: Huber,Anna,23456,w,2.3

`\csvlinetotablerow`
Ausgabe der zerlegten Zeile mit `&` zwischen den Einträgen. Dieses Kommando wird von Anwendern kaum verwendet werden.

3 Schlüssel zur Verarbeitung von CSV-Dateien

Die nachfolgenden Schlüssel sind jeweils mit ihrem vollen `pgfkeys`-Pfad beschrieben. Für die Anwendung innerhalb von Makros aus dem `csvsimple`-Paket kann man immer das Präfix `/csv/` weglassen.

3.1 Festlegung von Befehlen

`/csv/before reading`=⟨*Befehle*⟩ (Vorgabewert ⟨*leer*⟩)
Legt die ⟨*Befehle*⟩ fest, die vor Einlesen der CSV-Datei auszuführen sind.

`/csv/after head`=⟨*Befehle*⟩ (Vorgabewert ⟨*leer*⟩)
Legt die ⟨*Befehle*⟩ fest, die nach Einlesen der Kopfzeile auszuführen sind.

`/csv/before filter`=⟨*Befehle*⟩ (Vorgabewert ⟨*leer*⟩)
Legt die ⟨*Befehle*⟩ fest, die nach Einlesen und Konsistenzprüfung einer Zeile noch vor Prüfung von Filterbedingungen ausgeführt werden, siehe `filter` auf Seite 12.

`/csv/late after head`=⟨*Befehle*⟩ (Vorgabewert ⟨*leer*⟩)
Legt die ⟨*Befehle*⟩ fest, die nach Einlesen und Zerlegung der ersten akzeptierten Datenzeile, aber noch vor deren weiteren Verarbeitung auszuführen sind.

`/csv/late after line`=⟨*Befehle*⟩ (Vorgabewert ⟨*leer*⟩)
Legt die ⟨*Befehle*⟩ fest, die für eine Zeile bereits nach Einlesen und Zerlegung der nächstfolgenden akzeptierten Zeile (nach `before filter` auf Seite 9) für diese nächstfolgende Zeile auszuführen sind. `late after line` überschreibt `late after first line` und `late after last line`.

`/csv/late after first line`=⟨*Befehle*⟩ (Vorgabewert ⟨*leer*⟩)
Legt die ⟨*Befehle*⟩ fest, die statt `late after line` auf Seite 9 nach Verarbeitung der ersten Zeile auszuführen sind. Muss nach `late after line` gesetzt werden.

`/csv/late after last line`=⟨*Befehle*⟩ (Vorgabewert ⟨*leer*⟩)
Legt die ⟨*Befehle*⟩ fest, die statt `late after line` auf Seite 9 nach Verarbeitung der letzten Zeile auszuführen sind. Muss nach `late after line` gesetzt werden.

/csv/before line=⟨*Befehle*⟩ (Vorgabewert ⟨*leer*⟩)
Legt die ⟨*Befehle*⟩ fest, die nach `late after line` auf Seite 9 und vor `command` auf Seite 10 auszuführen sind. `before line` überschreibt `before first line`.

/csv/before first line=⟨*Befehle*⟩ (Vorgabewert ⟨*leer*⟩)
Legt die ⟨*Befehle*⟩ fest, die für die erste Zeile statt `before line` auf Seite 10 auszuführen sind. Muss nach `before line` gesetzt werden.

/csv/command=⟨*Befehle*⟩ (Vorgabewert **\csvline**)
Legt die ⟨*Befehle*⟩ fest, die für jede akzeptierte Zeile der Eingabedatei zwischen `before line` auf Seite 10 und `after line` auf Seite 10 auszuführen sind.

/csv/after line=⟨*Befehle*⟩ (Vorgabewert ⟨*leer*⟩)
Legt die ⟨*Befehle*⟩ fest, die nach `command` auf Seite 10 auszuführen sind. `after line` überschreibt `after first line`.

/csv/after first line=⟨*Befehle*⟩ (Vorgabewert ⟨*leer*⟩)
Legt die ⟨*Befehle*⟩ fest, die für die erste Zeile statt `after line` auf Seite 10 auszuführen sind.

/csv/after reading=⟨*Befehle*⟩ (Vorgabewert ⟨*leer*⟩)
Legt die ⟨*Befehle*⟩ fest, die nach Einlesen der CSV-Datei auszuführen sind.

```
\csvreader[
  before reading        = /A/\\,
  after head            = /B/,
  before filter         = \\/C/,
  late after head       = /D/,
  late after line       = /E/,
  late after first line = /F/,
  late after last line  = \\/G/,
  before line           = /H/,
  before first line     = /I/,
  after line            = /J/,
  after first line      = /K/,
  after reading         = \\/L/
   ]{noten.csv}{Name=\Name}{\textbf{\Name}}%
```

/A/
/B/
/C//D//I/**Maier**/K/
/C//F//H/**Huber**/J/
/C//E//H/**Weißbäck**/J/
/G/
/L/

Weitere vereinbare Befehle stehen optional für die unterstützten Tabellenarten zur Verfügung, siehe Abschnitt 3.4 ab Seite 13.

3.2 Kopfverarbeitung und Namenszuweisungen

/csv/head=⟨*Boolscher Wert*⟩ (Vorgabe- und Defaultwert `true`)
Legt fest, ob die erste Zeile der CSV-Datei als Kopfzeile behandelt werden soll, deren Einträge die Bezeichnungen der Spalten bilden.

/csv/nohead (–)
Entspricht `head=false`, d. h. die erste Zeile der CSV-Datei wird bereits als Datenzeile behandelt.

/csv/column names=⟨*Zuweisungen*⟩ (Vorgabewert ⟨*leer*⟩)
Fügt der Liste der Spalten-Makro-Zuweisungen neue ⟨*Zuweisungen*⟩ hinzu.

/csv/column names reset (–)
Leert die Liste der Spalten-Makro-Zuweisungen.

3.3 Konsistenzprüfung und Filterung

/csv/check column count=⟨*Boolscher Wert*⟩ (Vorgabe- und Defaultwert `true`)
Legt fest, ob die Zahl der Einträge einer Zeile mit dem vorgeschriebenen Wert verglichen werden soll.
Falls `true`, so wird eine nicht-konforme Zeile ohne Meldung verworfen.
Falls `false`, so wird jede Zeile akzeptiert und erzeugt möglichweise einen Fehler in der weiteren Verarbeitung.

/csv/nocheckcolumncount (–)
Entspricht `check columncount=false`, d. h. Zeilen werden nicht auf die korrekte Zahl der Einträge geprüft.

/csv/column count=⟨*Anzahl*⟩ (kein Vorgabewert)
Legt ⟨*Anzahl*⟩ für die Zahl der zulässigen Einträge pro Zeile fest. Diese Festlegung macht nur im Zusammenhang mit `nohead` auf Seite 11 Sinn, da anderenfalls ⟨*Anzahl*⟩ durch die Zahl der Einträge der Kopfzeile ersetzt wird.

/csv/on column count error=⟨*Befehle*⟩ (Vorgabewert ⟨*leer*⟩)
Aktion, die bei unzulässigen Zeilen durchgeführt wird.

/csv/warn on column count error (–)
Ausgabe einer Warnung bei unzulässigen Zeilen.

/csv/filter=⟨*Bedingung*⟩ (–)
Nur Zeilen, die die logische ⟨*Bedingung*⟩ erfüllen, werden akzeptiert. Als ⟨*Bedingung*⟩ dürfen Konstrukte aus dem `ifthen`-Paket verwendet werden. Um den Zeileninhalt vor dem Test der ⟨*Bedingung*⟩ (vor-)zuverarbeiten, kann `before filter` auf Seite 9 eingesetzt werden.

/csv/nofilter (Default)
Löscht einen gesetzten Filter.

/csv/filter accept all (Default)
Entspricht `nofilter`. Alle konsistenten Zeile werden akzeptiert.

/csv/filter reject all (–)
Alle Zeilen werden ignoriert.

/csv/filter equal=`{`⟨*TextA*⟩`}{`⟨*TextB*⟩`}` (–)
Nur Zeilen, für die ⟨*TextA*⟩ mit ⟨*TextB*⟩ nach Expansion übereinstimmt, werden akzeptiert.

/csv/filter not equal=`{`⟨*TextA*⟩`}{`⟨*TextB*⟩`}` (–)
Nur Zeilen, für die ⟨*TextA*⟩ mit ⟨*TextB*⟩ nach Expansion *nicht* übereinstimmt, werden akzeptiert.

3.4 Tabellenunterstützung

/csv/tabular=⟨*Formatvereinbarung*⟩ (–)
Umschließt die CSV-Verarbeitung mit `\begin{tabular}{`⟨*Formatvereinbarung*⟩`}` zu Beginn und `\end{tabular}` am Ende. Zudem werden die über die Schlüssel `before table` auf Seite 13, `table head` auf Seite 13, `table foot` auf Seite 13 und `after table` auf Seite 13 vereinbarten Befehle an den entsprechenden Stellen durchgeführt.

/csv/centered tabular=⟨*Formatvereinbarung*⟩ (–)
Wie `tabular` auf Seite 13, aber zusätzlich innerhalb einer `center`-Umgebung.

/csv/longtable=⟨*Formatvereinbarung*⟩ (–)
Wie `tabular` auf Seite 13, aber für die `longtable`-Umgebung. Benötigt das Paket `longtable`.

/csv/tabbing (–)
Wie `tabular` auf Seite 13, aber für die `tabbing`-Umgebung.

/csv/centered tabbing (–)
Wie `tabbing` auf Seite 13, aber zusätzlich innerhalb einer `center`-Umgebung.

/csv/before table=⟨*Befehle*⟩ (Vorgabewert ⟨*leer*⟩)
Diese ⟨*Befehle*⟩ werden vor `\begin{tabular}` bzw. vor `\begin{longtable}` bzw. vor `\begin{tabbing}` ausgeführt.

/csv/table head=⟨*Befehle*⟩ (Vorgabewert ⟨*leer*⟩)
Diese ⟨*Befehle*⟩ werden nach `\begin{tabular}` bzw. nach `\begin{longtable}` bzw. nach `\begin{tabbing}` ausgeführt.

/csv/table foot=⟨*Befehle*⟩ (Vorgabewert ⟨*leer*⟩)
Diese ⟨*Befehle*⟩ werden vor `\end{tabular}` bzw. vor `\end{longtable}` bzw. vor `\end{tabbing}` ausgeführt.

/csv/after table=⟨*Befehle*⟩ (Vorgabewert ⟨*leer*⟩)
Diese ⟨*Befehle*⟩ werden nach `\end{tabular}` bzw. nach `\end{longtable}` bzw. nach `\end{tabbing}` ausgeführt.

/csv/autotabular=⟨*Dateiname*⟩ (–)
Liest die gesamte CSV-Datei mit ⟨*Dateiname*⟩ und formatiert diese automatisch.

/csv/autolongtable=⟨*Dateiname*⟩ (–)
Liest die gesamte CSV-Datei mit ⟨*Dateiname*⟩ und formatiert diese automatisch mit Hilfe des benötigten `longtable`-Pakets.

3.5 Sonstiges

/csv/every csv (Vorgabewert ⟨*leer*⟩)
Eine Stilvereinbarung, die für jede CSV-Datei verwendet wird und überschrieben werden kann.

```
% Schaltet eine Warnmeldung für unzulässige Zeilen global ein.
\csvset{every csv/.style={warn on column count error}}
% Gleichbedeutend ist:
\csvstyle{every csv}{warn on column count error}
```

/csv/default (–)
Wird vor jedem Einlesen einer CSV-Datei ausgeführt und setzt alle Vorgaben auf die Standardwerte zurück[3]. Diesen Schlüssel sollte man nicht einsetzen oder verändern, wenn man nicht ganz genau weiß, was man tut.

/csv/file=⟨*Datei*⟩ (Vorgabewert `unknown.csv`)
Setzt den Dateinamen der einzulesenden CSV-Datei auf ⟨*Datei*⟩.

4 Beispiele

4.1 Ein Serienbrief

CSV-Datei `adressen.csv`

```
Name,Vorname,Geschlecht,Titel,Strasse und Nummer,PLZ,Ort,Überlast
Maier,Hans,M,,Am Bachweg 17,10010,Hopfingen,20
    Nachfolgend ein eingeklammertes Komma
Huber,Erna,W,Dipl.-Ing.,{Moosstraße 32, Hinterschlag},10020,Örtingstetten,30
Weißbäck,Werner,M,Prof. Dr.,Brauallee 10,10030,Klingenbach,40
    % Diese Zeile wird ignoriert %
  Siebener , Franz,M,  , Blaumeisenweg 12 , 10040 , Pardauz , 50
    % Leerzeichen zu Beginn und Ende der Einträge werden gelöscht %
Schmitt,Anton,M,,{\AE{}lfred-Esplanade, T\ae{}g 37}, 10050,\OE{}resung,60
```

Zunächst verschaffen wird uns einen Überblick über den Dateiinhalt mittels `\csvautotabular`. Wie man sieht, werden die unzulässigen Zeilen ignoriert.

```
\tiny\csvautotabular{adressen.csv}
```

Name	Vorname	Geschlecht	Titel	Strasse und Nummer	PLZ	Ort	Überlast
Maier	Hans	M		Am Bachweg 17	10010	Hopfingen	20
Huber	Erna	W	Dipl.-Ing.	Moosstraße 32, Hinterschlag	10020	Örtingstetten	30
Weißbäck	Werner	M	Prof. Dr.	Brauallee 10	10030	Klingenbach	40
Siebener	Franz	M		Blaumeisenweg 12	10040	Pardauz	50
Schmitt	Anton	M		Ælfred-Esplanade, Tæg 37	10050	Œresung	60

Nun wird ein Serienbrief erzeugt, bei dem für jeden Eintrag eine eigene Tabelle verwendet wird. Die Zuordnung der Spaltennamen zu TEX-Makros wird vorab über `\csvnames` auf Seite 7 vorgenommen, da man diese vielleicht mehrfach benötigt, z. B. zum Bedrucken von Briefumschlägen. Des weiteren wird mittels `\csvstyle{csvnames}` ein neuer Stil `anpassung` generiert, der einen eingelesenen leeren `\Titel` aus optischen Gründen zu `\unskip` umdefiniert. Die männlichen und weiblichen Anredeformen werden über das neue Makro `\ifmale` gesteuert.

[3] `default` wird aufgrund der vielen global wirksamen Einstellungen verwendet.

```
\csvnames{adressdatei}{Name=\Name,Vorname=\Vorname,Geschlecht=\GS,Titel=\Titel,
            Strasse und Nummer=\Strasse,PLZ=\PLZ,Ort=\Ort,Überlast=\Ueberlast}
\csvstyle{anpassung}{before line={\ifthenelse{\equal{\Titel}{}}{\gdef\Titel{\unskip}}{}}}
\newcommand{\ifmale}[2]{\ifthenelse{\equal{\GS}{M}}{#1}{#2}}

\csvreader[adressdatei,anpassung]{adressen.csv}{}{%
  \renewcommand{\arraystretch}{1.2}
  \begin{tabular}{|p{4cm}p{9.5cm}|}\hline
  \ifmale{Herrn}{Frau}~\Titel\newline
  \Vorname~\Name\newline
  \Strasse\newline
  \PLZ~\Ort &
  {\itshape\ifmale{Sehr geehrter Herr}{Sehr geehrte Frau}~\Titel~\Name,}\newline
  wir teilen Ihnen mit, dass eine Überlast von \Ueberlast\%{}
  zu Ihren Gunsten erkannt wurde\ldots\\\hline
  \end{tabular}
}
```

Herrn Hans Maier Am Bachweg 17 10010 Hopfingen	*Sehr geehrter Herr Maier,* wir teilen Ihnen mit, dass eine Überlast von 20% zu Ihren Gunsten erkannt wurde…
Frau Erna Huber Moosstraße 32, Hinterschlag 10020 Örtingstetten	*Sehr geehrte Frau Huber,* wir teilen Ihnen mit, dass eine Überlast von 30% zu Ihren Gunsten erkannt wurde…
Herrn Werner Weißbäck Brauallee 10 10030 Klingenbach	*Sehr geehrter Herr Weißbäck,* wir teilen Ihnen mit, dass eine Überlast von 40% zu Ihren Gunsten erkannt wurde…
Herrn Franz Siebener Blaumeisenweg 12 10040 Pardauz	*Sehr geehrter Herr Siebener,* wir teilen Ihnen mit, dass eine Überlast von 50% zu Ihren Gunsten erkannt wurde…
Herrn Anton Schmitt Ælfred-Esplanade, Tæg 37 10050 Œresung	*Sehr geehrter Herr Schmitt,* wir teilen Ihnen mit, dass eine Überlast von 60% zu Ihren Gunsten erkannt wurde…

4.2 Eine graphische Darstellung

CSV-Datei `daten.csv`

```
Land,Gruppe,Menge
Bayern,A,1700
Baden-Württemberg,A,2300
Sachsen,B,1520
Thüringen,A,1900
Hessen,B,2100
```

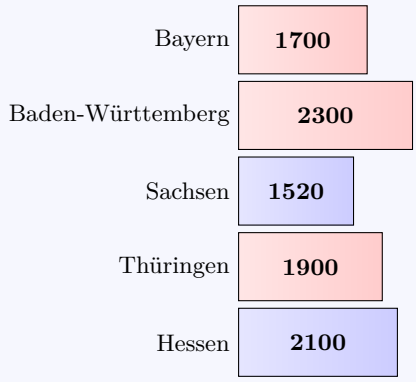
Für den ersten Überblick wird zunächt wieder `\csvautotabular` eingesetzt.

```
\csvautotabular{daten.csv}
```

Land	Gruppe	Menge
Bayern	A	1700
Baden-Württemberg	A	2300
Sachsen	B	1520
Thüringen	A	1900
Hessen	B	2100

Die Mengenangaben werden in nachfolgender Graphik durch Balken veranschaulicht. Die Gruppenzuordnung wird für die Farbgestaltung verwendet.

```
% benötigt das Paket tikz
\begin{tikzpicture}[Gruppe/A/.style={left color=red!10,right color=red!20},
                    Gruppe/B/.style={left color=blue!10,right color=blue!20}]
\csvreader{daten.csv}{Land=\Land,Menge=\Menge,Gruppe=\Gruppe}{%
  \begin{scope}[yshift=-\thecsvrow cm]
  \path [draw,Gruppe/\Gruppe] (0,-0.45)
    rectangle node[font=\bfseries] {\Menge} (\Menge/1000,0.45);
  \node[left] at (0,0) {\Land}  ;
  \end{scope}  }
\end{tikzpicture}
```



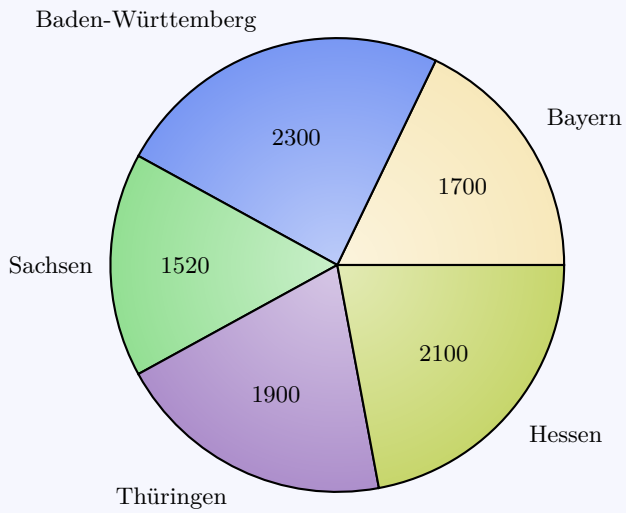
Falls benötigt, so lässt sich auch ein Tortendiagramm durch zweifachen Aufruf von `\csvreader` erzeugen. Im ersten Schritt wird die Gesamtsumme ermittelt und im zweiten Schritt werden die einzelnen Tortenstücke gezeichnet.

```
% Modifiziertes Beispiel von www.texample.net für Tortendiagramme
% Benötigt die Pakete tikz, xcolor, calc
\definecolorseries{myseries}{rgb}{step}[rgb]{.95,.85,.55}{.17,.47,.37}
\resetcolorseries{myseries}%

% Ein Tortenstück
\newcommand{\slice}[4]{
  \pgfmathsetmacro{\midangle}{0.5*#1+0.5*#2}
  \begin{scope}
    \clip (0,0) -- (#1:1) arc (#1:#2:1) -- cycle;
    \colorlet{SliceColor}{myseries!!+}%
    \fill[inner color=SliceColor!30,outer color=SliceColor!60] (0,0) circle (1cm);
  \end{scope}
  \draw[thick] (0,0) -- (#1:1) arc (#1:#2:1) -- cycle;
  \node[label=\midangle:#4] at (\midangle:1) {};
  \pgfmathsetmacro{\temp}{min((#2-#1-10)/110*(-0.3),0)}
  \pgfmathsetmacro{\innerpos}{max(\temp,-0.5) + 0.8}
  \node at (\midangle:\innerpos) {#3};
}

% Summe der Mengenwerte
\csvreader[before reading=\def\mysum{0}]{daten.csv}{Menge=\Menge}{%
  \pgfmathsetmacro{\mysum}{\mysum+\Menge}%
}

% Zeichnung des Tortendiagramms
\begin{tikzpicture}[scale=3]%
\def\mya{0}\def\myb{0}
\csvreader{daten.csv}{Land=\Land,Menge=\Menge}{%
  \let\mya\myb
  \pgfmathsetmacro{\myb}{\myb+\Menge}
  \slice{\mya/\mysum*360}{\myb/\mysum*360}{\Menge}{\Land}
}
\end{tikzpicture}%
```

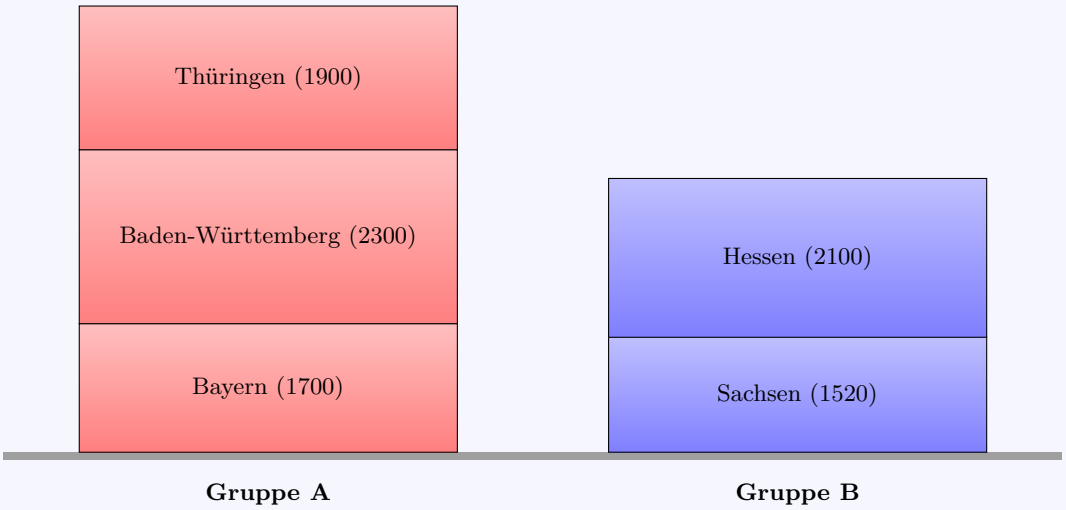


Nachfolgend wird die Filteroption eingesetzt, um zunächst die Länder der Gruppe A und danach die Länder der Gruppe B auf die passenden Säulen zu stellen.

```
\csvnames{laender}{Land=\Land,Menge=\Menge,Gruppe=\Gruppe}

\newcommand{\zeichneGruppe}[2]{%
  \def\mya{0}\def\myb{0}
  \node[below=3mm] at (2.5,0) {\bfseries Gruppe #1};
  \csvreader[laender,filter equal={\Gruppe}{#1}]{daten.csv}{}{%
    \let\mya\myb
    \pgfmathsetmacro{\myb}{\myb+\Menge}
    \path[draw,top color=#2!25,bottom color=#2!50]
      (0,\mya/1000) rectangle node{\Land\ (\Menge)} (5,\myb/1000);
}}

\begin{tikzpicture}
  \fill[gray!75] (-1,0) rectangle (13,-0.1);
  \zeichneGruppe{A}{red}
  \begin{scope}[xshift=7cm]
  \zeichneGruppe{B}{blue}
  \end{scope}
\end{tikzpicture}
```



Index