LaTeX und Tabellen

Sammlung einiger Kniffe

v1.11 vom 25. Februar 2019

Inhaltsverzeichnis

Vorwort

An dieser Stelle möchte ich kurz erklären, dass auch ich nicht darum herumkomme, meine Erfahrungen mit Tabellen in LaTeX, einem berüchtigten Arbeitsfeld mit dieser Software, vorzuweisen. Es handelt sich bislang um eine einfache Sammlung von Tipps, um die meisten ästhetischen und typografischen Hürden beim Erstellen einfacher Tabellen zu überwinden.

Generell ist empfehlenswert, die folgenden Pakete zu laden, sofern man Tabellen in sein Dokument aufnehmen möchte: `booktabs`, `array`, `ragged2e`

1 Grundaufbau

```
\begin{tabular}{|l|cr|}
        \hline
        A & B & C \\
        9 & 22 & a \\
        \hline
\end{tabular}
```

erzeugt

A	B	C
9	22	a

Die `tabular`-Umgebung gilt als grundlegende Tabellenumgebung, zwischen der der Tabelleninhalt eingefasst wird. Im Kopf definiert man die Anzahl der Spalten,

wobei jeder der Buchstaben `l` (linksbündig), `c` (zentriert), `r` (rechtsbündig) einer Spalte entspricht und gleichzeitig die Ausrichtung ihres Inhalts bestimmt. Durch die Eingabe einer Pipe (`|`) kann man vertikale Linien zwischen den Spalten vorgeben, was aber aus ästhetischer Hinsicht verpönt ist; Tabellen sollten wenn möglich nur durch wenige, sinnvoll gesetzte, *horizontale* Linien getrennt werden! Braucht man doch horizontale Linien, kann man diese auf einfache Weise mit einem `\hline`-Befehl erzeugen, der am Zeilenende (oder vor der vorherigen Zeile) sitzt und auch mehrfach in Form von `\hline\hline` benutzt werden darf; im Beispiel oben werden die `\hline` nur als sog. Kopf- und Fußlinie gesetzt. Die Zeilenenden selbst werden mittels `\\` abgeschlossen und die einzelnen Zelleninhalte innerhalb einer Zeile mit `&` getrennt. Damit kann man schon mal einen Großteil an Tabellen setzen.

Man muss beachten, dass insbesondere bei umfangreicheren Tabellen der »Wust« aus Tabelleninhalten, `&` und `\\` dergestalt überhand nehmen kann, dass man rasch den Überblick verliert. Fehlt auch nur ein Trenner an der richtigen Stelle oder sind mit den Buchstaben `l`, `r` usw. zu wenig Spalten definiert, aber für eine Spalte mehr Inhalt hinterlegt, gibt es beim Kompilieren jede Menge Fehlermeldungen, die zu finden sehr aufwendig sein kann!

1.1 Tabelle in Gleitumgebung

Meistens wird man eine Tabelle brauchen, die wie eine Abbildung gleitet, mit Überschrift und der Möglichkeit darauf zu verlinken. Man fasst sie hierfür einfach in eine `table`-Umgebung ein:

```
\begin{table}
        \centering
        \caption{Meine Tabelle.}
        \label{tabelle1}
\begin{tabular}{|l|cr|}
        \hline
        A & B & C \\
        9 & 22 & a \\
        \hline
```

Tabelle 1: Meine Tabelle.

A	B	C
9	22	a

```
\end{tabular}
\end{table}
```

Das Ergebnis (Tab. 1) sieht man am Kopf der Seite.

1.2 Tabellen mit vorgegebener Breite

Hierfür wird die Stern-Version der `tabular`-Anweisung benutzt und in einem zweiten Argument die gewünschte Breite angegeben. Man beachte, dass sich die Spaltenbreite nach wie vor nach dem Inhalt richtet (kein Wortumbruch!).

```
\begin{tabular*}{8cm}{|l|cr|}
        \hline
        A & B & C-Spalte \\
        9 & 22 & mit mehr Text \\
        \hline
\end{tabular*}
```

erzeugt

A	B	C-Spalte
9	22	mit mehr Text

Für eine Tabelle, die exakt genauso breit ist wie der Textblock, kann man auch die `\textwidth`-Anweisung anstatt einer konkreten Zahl benutzen:

```
\begin{tabular*}{\textwidth}{|l|cr|}
```

1.3 Vorgegebene Spaltenbreiten

Will man die Spaltenbreite konkret vorgeben, benutzt man anstelle der Spaltendefinition `l`, `c` oder `r` die Angabe `p{Breite}`, also z. B. `p{4cm}`:

```
\begin{tabular}{|p{2cm}|p{1cm}|p{2cm}|}
        \hline
        A & B & C-Spalte \\ \hline
        9 & 22 & mit noch mehr Text \\
        \hline
\end{tabular}
```

erzeugt

A	B	C-Spalte
9	22	mit noch mehr Text

Wie man sieht, wird der Tabelleninhalt linksbündig ausgerichtet, aber nur unschön umgebrochen (siehe nächster Abschnitt).

1.4 Vorgegebene Spaltenbreiten mit automatischen Wortumbruch

Für diesen Kniff ist das Laden des `array`- und `ragged2e`-Pakets notwendig. Nun kann am Dokument-Anfang ein neuer Spaltentyp `L` definiert werden (statt `L` geht auch ein anderer Buchstabe):

```
\newcolumntype{L}[1]{>{\raggedright\arraybackslash}p{#1}}
```

Nun lässt sich anstatt einer `p`-Spalte eine `L`-Spalte setzen:

```
\begin{tabular}{|p{2cm}|p{1cm}|L{2cm}|}
        \hline
```

```
        A & B & C-Spalte \\ \hline
        9 & 22 & mit noch mehr Text \\
        \hline
\end{tabular}
```

erzeugt

A	B	C-Spalte
9	22	mit noch mehr Text

(vergleiche mit Tabelle im vorherigen Abschnitt!)

Alternativ lässt sich eine Definition namens *K* erzeugen, bei der die Inhalte zentriert ausgerichtet und trotzdem umgebrochen werden. Dafür wird der vorherige Befehl einfach umdefiniert:

```
\newcolumntype{K}[1]{>{\centering\arraybackslash}p{#1}}
```

1.5 Vorgegebene Spaltenbreiten mit manuellen Wortumbruch

Man kann auch manuell mit dem `\newline`-Befehl einen Zeilenumbruch erzeugen, was aber nur in p-Spalten mit festgelegter Breite gelingt:

```
\begin{tabular}{p{2.5cm}lr}
        \hline
        Zeilenumbruch \newline nun m"oglich & B & C \\
        \hline
\end{tabular}
```

ergibt:

Zeilenumbruch nun möglich	B	C

1.6 Übersichtliche Definition von gleichartigen Spaltentypen

Angenommen, man benötigt eine Tabelle mit 19 links ausgerichteten und nochmal 7 recht ausgerichteten Spalten. Dann würde man schreiben:

```
\begin{tabular}{lllllllllllllllllll|rrrrrrr}
```

Einfacher und übersichtlicher ginge es aber wie folgt:

```
\begin{tabular}{*{19}{l}|*{7}{r}}
```

1.7 Drehung der Spaltenköpfe um 90°

Für die Nutzung des `\rotatebox`-Kommandos muss das `rotating`-Paket geladen werden. Dann kann man im 1. Argument den Winkel einstellen (`90` für »90° nach links«), in das 2. Argument kommt der Inhalt. Wie man sieht, wäre etwas mehr Platz gleich unter der Kopflinie nicht verkehrt, siehe Abschnitt 2.1 und 2.7.

```
\begin{tabular}{llr}
        \hline
        \rotatebox{90}{Spalte 1} & \rotatebox{90}{Spalte 2} &%
        \rotatebox{90}{Spalte 3} \\
        A & B & C \\
        \hline
\end{tabular}
```

ergibt:

Spalte 1	Spalte 2	Spalte 3
A	B	C

1.8 Drehung einer ganzen Tabelle um 90°

Hier bedient man sich abermals des `rotating`-Pakets und benutzt anstatt der `table`-Umgebung (für gleitende Tabellen) die `sidewaystable`-Umgebung:

```
\begin{sidewaystable}
        \caption{Text}
        \label{Tabelle1}
                \begin{tabular}{llr}
                ...
                \end{tabular}
\end{sidewaystable}
```

1.9 Spalteninhalte nach Dezimaltrenner ausrichten

Dieser Kniff ist für gefüllte Zahlen-Tabellen geeignet, in denen untereinanderstehende Werte an ihrem Dezimaltrenner (Komma, Punkt) ausgerichtet werden sollen. Die Anwendung ist denkbar einfach: Das Laden des Pakets `siunitx` stellt eine neue Spaltendefinition namens `S` bereit, die alles erledigt. Der Inhalt der Spalten wird dabei insgesamt zentriert ausgerichtet.

```
\begin{tabular}{SSS}
        \hline
        \textbf{Werte 1} & \textbf{Werte 2} & \textbf{Werte 3} \\
        0,91 & 0,23 & 0,3 \\
        2,04 & 1,44 & 23,19 \\
        0,5 & 97,7 & -1,34 \\
        \hline
\end{tabular}
```

ergibt:

Werte 1	Werte 2	Werte 3
0.91	0.23	0.3
2.04	1.44	23.19
0.5	97.7	−1.34

2 Anpassung der `tabular`-Vorgaben

2.1 Schönere Linien

Wie in der Einführung bereits notiert, sollte in Tabellen auf vertikale Linien verzichtet werden, und horizontale Linien sollten, wenn überhaupt, nur sparsam eingesetzt werden. Linien lassen sich, wie bereits gezeigt, mit Pipes: `|` (vertikale Linien) und `\hline`-Kommandos (horizontale Linien) generieren. Leider setzt Standard-`tabular` die Linien unangenehm eng an den Text; etwas Auflockerung wäre wünschenswert. Genau das übernimmt das `booktabs`-Paket und stellt für horizontale Linien einige neue Kommandos bereit:

```
\begin{tabular}{lll}
        \toprule
        \textbf{Nr.} & \textbf{Probe} & \textbf{Fundort} \\
        \midrule
        1 & Dino-Knochen & Museum \\
        2 & Fossiles Holz & Wald \\
        \bottomrule
\end{tabular}
```

ergibt:

Nr.	Probe	Fundort
1	Dino-Knochen	Museum
2	Fossiles Holz	Wald

Man beachte, dass die `\toprule` und `\bottomrule` etwas fetter gezeichnet sind, während die `\midrule` die `\hline` ersetzt. Man vergleiche die Zeilenabstände mit Standard-`tabular` wie folgt:

```
\begin{tabular}{lll}
        \hline
        \textbf{Nr.} & \textbf{Probe} & \textbf{Fundort} \\
        \hline
        1 & Dino-Knochen & Museum \\
        2 & Fossiles Holz & Wald \\
        \hline
\end{tabular}
```

ergibt:

Nr.	**Probe**	**Fundort**
1	Dino-Knochen	Museum
2	Fossiles Holz	Wald

Es zeigt sich eindeutig eine Verbesserung der Tabellen-Ästhetik durch Auflockerung der Zeilenabstände! Die Linien-Kommandos des `booktab`-Pakets sind uneingeschränkt zu empfehlen und sollten die Verwendung von `\hline` immer ersetzen!

2.2 Überstehende Linien stutzen

Wie in den beiden Beispieltabellen unter Abschnitt 2.1 zu sehen, laufen die horizontalen Linien geringfügig weiter als der in der Tabelle befindliche Text. Wer das »abstellen« möchte, definiert als ersten und letzten »Spaltentyp« noch ein `@{}`

```
\begin{tabular}{@{}lll@{}}
        \hline
        \textbf{Nr.} & \textbf{Probe} & \textbf{Fundort} \\
        \hline
```

```
        1 & Dino-Knochen & Museum \\
        2 & Fossiles Holz & Wald \\
        \hline
\end{tabular}
```

ergibt:

Nr.	Probe	Fundort
1	Dino-Knochen	Museum
2	Fossiles Holz	Wald

2.3 Nur teilweise durchgängige, horizontale Linien

Dies wird normalerweise mit dem `\cline`-Kommando realisiert, das als Argument die Start- und End-Spalte aufnimmt. Wie alle Standard-`tabular`-Linien wird auch `cline` viel zu eng gesetzt, sodass sich die Verwendung von `cmidrule` aus dem `booktabs`-Paket (Abschnitt 2.1) empfiehlt und wie `cline` bedienen lässt:

```
\begin{tabular}{llr}
        \toprule
        A & B & C \\
        \cmidrule{2-3}
        A & B & C \\
        \cmidrule{1-2}
        A & B & C \\
        \bottomrule
\end{tabular}
```

ergibt:

A	B	C
A	B	C
A	B	C

2.4 Schriftgröße und Schriftfamilie

Tabellen umfangreichen Inhalts muss man manchmal hinsichtlich ihrer Schriftgröße anpassen, damit sie am Seitenrand nicht überlaufen. Für Tabellen *ohne* Gleitumgebung fasst man die `tabular`-Umgebung in eine weitere Umgebung mit Schriftgrößen-Angabe ein:

```
\begin{tiny}
        \begin{tabular}{lll}
        \toprule
        \textbf{Nr.} & \textbf{Probe} & \textbf{Fundort} \\
        \midrule
        1 & Dino-Knochen & Museum \\
        2 & Fossiles Holz & Wald \\
        \bottomrule
\end{tabular}
\end{tiny}
```

ergibt:

Nr.	Probe	Fundort
1	Dino-Knochen	Museum
2	Fossiles Holz	Wald

Werden Tabellen in Gleitumgebungen benutzt, genügt ein Schriftgrößen-Kommando innerhalb der `table`-Umgebung; es gilt dann auch nur für diesen Bereich:

```
\begin{table}
\scriptsize
\begin{tabular}{lll}
        ...
\end{tabular}
\end{table}
```

2.5 Spaltenabstand einpassen (einheitlich für alle Spalten)

Manchmal möchte man den Standard-Spaltenabstand generell etwas verringern oder erhöhen, z. B. um eine Tabelle zu verkleinern, damit sie nicht am Seitenrand überläuft. Man erreicht dies durch Veränderung des `\tabcolsep`-Befehls und gibt dabei einen Wert ein:

```
\setlength{\tabcolsep}{0.6mm} % Spaltenabstand anpassen
\begin{tabular}{llr}
        \hline
        A & B & C \\
        \hline
\end{tabular}
```

2.6 Spaltenabstand einpassen (nur ausgewählte Spalten)

Nicht in jedem Fall sollen die Spalten gleich breit sein. Mittels `@{\hskip 2mm}`, das man an die Position des jeweiligen Spaltenübergangs setzt (z. B. zwischen Spalte 1 und 2), kann man einen Abstand vorgeben. Im Beispiel wird der Abstand zwischen Spalte 1 und 2 auf 6 mm und derjenige zwischen Spalte 2 und 3 auf 12 mm festgelegt (zur Veranschaulichung mit vertikalen Linien):

```
\begin{tabular}{|l@{\hskip 6mm}|l@{\hskip 12mm}|r|}
        \hline
        A & B & C \\
        \hline
\end{tabular}
```

ergibt:

A	B	C

Man beachte, dass hier der *Abstand der Spalten* und nicht die *Spaltenbreite* (Abschnitt 1.3) angepasst wird!

2.7 Zeilenabstand einpassen (einheitlich für alle Tabellen)

Standard-`tabular` setzt die Zeilen relativ eng. Zur Auflockerung gebrauche man das `booktabs`-Paket und seine Kommandos (Abschnitt 2.1) oder gebe einen Wert manuell vor:

```
\renewcommand{\arraystretch}{1.2} %Zeilenabstand anpassen
\begin{tabular}{llr}
        \hline
        A & B & C \\
        1 & 2 & 3 \\
        \hline
\end{tabular}
```

ergibt:

A	B	C
1	2	3

Das `arraystretch`-Kommando kann man auch am Dokument-Anfang definieren und gilt dann für alle Tabellen im Dokument.

2.8 Zeilenabstand einpassen (für einzelne Zeilen)

Schauen wir noch einmal auf das Beispiel aus Abschnitt 1.7:

```
\begin{tabular}{llr}
        \hline
        \rotatebox{90}{Spalte 1} & \rotatebox{90}{Spalte 2} &%
        \rotatebox{90}{Spalte 3} \\
        A & B & C \\
        \hline
\end{tabular}
```

ergibt:

Spalte 1	Spalte 2	Spalte 3
A	B	C

Man sieht, dass der obere Rand der ersten Zeile sehr gedrängt liegt, die obere Zeilenbegrenzung liegt quasi auf den Buchstaben auf. Nun lassen sich mit folgendem Kniff zusätzliche vertikale Abstände einfügen. Zunächst definiert man zwei neue Kommandos, `\T` und `\B`:

```
\newcommand\T{\rule{0pt}{8ex}} %für einen Abstand nach oben
\newcommand\B{\rule[-1.5ex]{0pt}{0pt}} % für einen Abstand nach unten
```

Beide Befehle definieren im Grunde eine unsichtbare Linie, die keine Länge, sondern nur eine Höhe hat; dieser regelt dann den Abstand. Beide Kommandos fügt man nach Bedarf vor und nach der betreffenden Zeile ein:

```
\begin{tabular}{llr}
        \hline
        \T \rotatebox{90}{Spalte 1} & \rotatebox{90}{Spalte 2} &%
        \rotatebox{90}{Spalte 3} \B \\
        A & B & C \\
        \hline
\end{tabular}
```

ergibt:

Spalte 1	Spalte 2	Spalte 3
A	B	C

Durch Variation der Werte kann man seine Tabellen gut nachbessern.

2.9 Formatdefinition für eine ganze Spalte

Manchmal soll eine ganze Spalte einheitlich formatiert sein, z. B. fett. Um nun nicht jedes Mal den ersten Eintrag mit einem Formatbefehl für fett zu versehen (`\textbf{ }`), setzt man ihn mit einem > und dem gewünschten Format *vor* die Spaltendefinition. Im Beispiel wurde die erste Spalte fett gemacht und die dritte wird kursiviert:

```
\begin{tabular}{>{\bfseries}ll>{\itshape}l}
        \toprule
        Nummer & Gestein & Mineral \\
        \midrule
        1 & Granit & Quarz \\
        2 & Diabas & Pyroxen \\
        3 & Granulit & Feldspat \\
        \bottomrule
\end{tabular}
```

ergibt:

Nummer	Gestein	*Mineral*
1	Granit	*Quarz*
2	Diabas	*Pyroxen*
3	Granulit	*Feldspat*

2.10 Formatdefinition für eine ganze Zeile

In viele Fällen soll die 1. Zeile einer Tabelle (Tabellenkopf) einheitlich formatiert sein, z. B. fett. Um nun nicht jeden Eintrag der 1. Zeile mit einem Format-Kommando wie `\textbf{ }` zu versehen, kann man sich einer Abkürzung bedienen. Hierfür wird das `tabu`-Paket geladen und ein sog. `\head` definiert:

```
\newcommand*\head{\rowfont[c]{\bfseries}}
```

Anstatt über die Option `c` alle Spalten zentriert auszurichten, kann man auch `l` (linksbündig), `r` (rechtsbündig) oder `j` (Blocksatz) nutzen. Als Format für alle Einträge wird *fett* (`\bfseries`) festgelegt. Anstatt einer `tabular`-Umgebung nimmt man nun die `tabu`-Umgebung und bestimmt mit `\head` diejenige Zeile, die nach Festlegung formatiert werden soll:

```
\newcommand*\head{\rowfont[c]{\bfseries}}

\begin{tabu}{lll}
        \toprule
        \head
                Wort 1 & Wort 2 & Wort 3 \\
        \midrule
                A & B & C \\
        \bottomrule
\end{tabu}
```

ergibt:

Wort 1	**Wort 2**	**Wort 3**
A	B	C

3 Zusammenfassung von Zellen

3.1 Zellen horizontal zusammenfassen (`multicolumn`)

Für die Zusammenfassung von horizontalen Zellen kommt meist das `\multicolumn`-Kommando zum Einsatz. Es hat drei Argument, wovon das erste die Anzahl der zusammengefassten Zellen enthält, das zweite die Ausrichtung (`c` für zentriert) und das dritte den Inhalt. Man beachte, dass jeweils die entsprechende Anzahl der Spaltentrenner (&) reduziert werden muss, d. h. wenn bei 3 Spalten zwei davon zusammengefasst werden, steht zwischen diesen beiden (= `\multicol`-Bereich) und der 3. Spalte nur noch *ein* Spaltentrenner &!

```
\begin{tabular}{lll}
        \toprule
        \multicolumn{2}{c|}{\textbf{Gesteine}} & \textbf{Fundort} \\
        \midrule
        Granit & Diabas & Strand \\
        Sandstein & Kalkstein & Fluss \\
        Feuerstein & \multicolumn{2}{|c|}{Berghang} \\
        \multicolumn{3}{c}{\emph{in die Sammlung!}} \\
        \bottomrule
\end{tabular}
```

ergibt:

Gesteine		**Fundort**
Granit	Diabas	Strand
Sandstein	Kalkstein	Fluss
Feuerstein	Berghang	
in die Sammlung!		

3.2 `multicolumn` für Zeilenumbruch auf Breite zwingen

Eine `multicolumn` tut sich von Haus aus schwer damit, bei Überlänge automatisch umzubrechen. Man zwingt sie auf eine gewisse Breite mittels einer `parbox`:

```
\begin{tabular}{ccc}
        \toprule
        \multicolumn{2}{c}{\parbox{2.0cm}{Umbruch w"are sch"on}} & C
        ↪  \\
        \midrule
        1 & 2 & 3 \\
        \bottomrule
\end{tabular}
```

ergibt:

Umbruch wäre schön		C
1	2	3

3.3 Zellen vertikal zusammenfassen (`multirow`)

Die Anwendung erfolgt analog zum `multicolumn`-Paket, wobei die zusammengefassten Zellen mit einem Spaltentrenner »übersprungen« werden. Es muss das Paket `multirow` geladen werden. Das erste Argument enthält die Anzahl der zusammengefassten Zellen, das zweite nimmt eine Breite auf (entweder konkrete Vorgabe oder automatische Anpassung mit `*`), das dritte Argument enthält den eigentlichen Inhalt. Wird eine konkrete Breite vorgegeben, ist ein Zeilenumbruch möglich (2. Beispiel). Häufig wird der `multirow`-Inhalt noch um 90° gedreht, wie in Abschnitt 1.7 beschrieben und in der Beispieltabelle unter Abschnitt 3.4 gezeigt.

```
\begin{tabular}{llr}
        \toprule
        \multirow{2}{*}{Test} & B & C \\
         & B & C \\
         \midrule
         Wort & Satz & Text \\
        \bottomrule
\end{tabular}
```

ergibt:

Test	B	C
	B	C
Wort	Satz	Text

2. Beispiel mit Zeilenumbruch:

```
\begin{tabular}{llr}
        \toprule
        \multirow{2}{2cm}{mit einem Umbruch} & B & C \\
         & B & C \\
         \midrule
         Wort & Satz & Text \\
        \bottomrule
\end{tabular}
```

ergibt:

mit einem	B	C
Umbruch	B	C
Wort	Satz	Text

3.4 Manuelle Anpassungen der `multirow`-Ausrichtung

In vielen Fällen versagt die automatische Positionierung des `multirow`-Bereichs. Einerseits kann man versuchen, den mit `multirow` positionierten Text über ein vorangestelltes `\vfil` mittig zu zentrieren:

```
\multirow{4}{2cm}{\vfil Wort}
```

Andererseits kann man ein optionales Argument (= Option) mit einer konkreten Vorgabe angeben, um die der Text versetzt werden soll (negative vs. positive Werte entsprechen einer Verschiebung nach oben bzw. unten):

```
\begin{tabular}{llr}
        \toprule
        \multirow{4}{1cm}[-0.5cm]{\rotatebox{90}{Test}} & B & C \\
         & B & C \\
```

```
        & 1 & 2 \\
        & Test & Wort \\
      \bottomrule
\end{tabular}
```

ergibt:

Test	B	C
	B	C
	1	2
	Test	Wort

3.5 `multirow` mit geschachtelter Tabelle imitiert

Die häufig besseren Ergebnisse erzielt man, wenn man anstatt einer `multirow` einfach eine geschachtelte Tabelle, d. h. eine Tabelle in einer Tabelle, benutzt. Die »sekundäre« Tabelle wird anstelle des eigentlichen Zelleneintrags gesetzt und stellt in diesem Beispiel 3 Zeilen (Inhalt: »B«) einer 1-spaltigen Tabelle dar:

```
\begin{tabular}{ccc}
      \toprule
            Wort &
            \begin{tabular}{c}
                  B \\ B \\ B \\
            \end{tabular}
            & C \\
            \midrule
      Wort & Satz & Text \\
      \bottomrule
\end{tabular}
```

ergibt:

Wort	B B B	C
Wort	Satz	Text

4 Diagramme mit Tabellen

Auch mit der LaTeX-Tabellenumgebung lassen sich einige einfache Diagramme realisieren, ohne sie über eine externe Software generieren zu lassen und z. B. als PDF einbinden zu müssen. Das folgende Beispiel habe ich dem Buch »Tabellen mit LaTeX« von H. Voß entnommen und etwas angepasst:

Zusammensetzung Probe	
62 %	Quarz
12 %	Feldspat
6 %	Glimmer
20 %	Turmalin

Eine weitere Möglichkeit Diagramme zu erzeugen ist die Nutzung des sog. `datatool`-Pakets, das einerseits den praktischen Import von `csv`-Tabellen ermöglicht und die Daten z. B. als Histogramm, XY-Plot oder Tortendiagramm ausgeben kann (siehe umfangreiche Dokumentation des Pakets).

5 Import/Export

Für viele gerade ungeübte Nutzer im Umgang mit LaTeX-Tabellen ist die nicht ganz unberechtigte Frage danach, ob es nicht einen einfacheren Weg gibt, den relativ unübersichtlichen Tabellencode extern bequem zu formatieren und zu befüllen, um im Anschluss daran die Tabelle zu LaTeX-Code zu exportieren.

Genau das macht die Software Gnumeric, die neben z. B. `Excel` und `Calc` eine weitere Tabellenkalkulation darstellt und von diesen beiden optisch kaum unterscheidbar ist bzw. sich größtenteils genauso bedient. Auch wenn man `Gnumeric` nicht als Tabel-

lenkalkulation benutzen mag, kann man mit ihr doch vorhandene `Excel`- und anders formatierte Tabellen öffnen und sie über den Menüpunkt

```
Daten|Daten exportieren|In anderes Format exportieren ...
```

zu LaTeX-Code exportieren, der dann bequem per Copy & Paste in sein LaTeX-Dokument übernommen werden kann. Zumindest umfangreiche Tabellen lassen sich erst einmal so einfach und schnell erstellen, um an ihnen dann weitere Formatierungen vorzunehmen.

Neuerdings habe ich eine weitere Software, u. a. für Windows, entdeckt, nämlich LaTable, mit der man Tabellen zumindest »vorzeichnen« und befüllen kann, um sie anschließend als LaTeX-Code zu speichern. Ich halte die Nutzung von `Gnumeric`, auch für die Konzeption ganz neuer Tabellen, für sinnvoller.

Für die Nutzer der Kommandozeile gibt es weiterhin die Pakete `excel2latex` und `exceltex` aus dem CTAN-Bestand, die ebenfalls den direkten Import eines Tabellendokuments in sein LaTeX-Dokument ermöglichen. Wer allerdings, gerade als Windows-Nutzer, keine große Affinität zu Kommandozeile und »kryptischen Befehlen« hat, sollte, wie empfohlen, `Gnumeric` benutzen.

Darüber hinaus bringen viele TeX-Editoren (z. B. `Kile`) spezielle »Assistenten« mit, über die sich Größe und Formatierung einer Tabelle über gewohnte Mausklicks einrichten und befüllen lässt.

Für LibreOffice gibt es eine Erweiterung namens `Calc2LaTeX`, mit der man aus dem Tabellenkalkulationsmodul heraus LaTeX-Tabellen erzeugen kann. Leider scheint diese Erweiterung nicht mehr angeboten zu werden, oder ist nur für ältere LibreOffice-Versionen verfügbar.