Data Import, Conversion, and Export (dice)

Version 0.1

Alex Baker

https://github.com/ambaker1/dice

September 29, 2023

Abstract

The package "dice" provides basic data import/export and conversion utilities. Four datatypes are supported: space-delimited values (txt), comma-separated values (csv), matrices (mat), and tables (tbl).

File Import/Export

The commands *fread* and *fputs* simplify file I/O in Tcl.

```
fread <$option $value ...> <-newline> $file
```

`$option $value ...`	File configuration options, see Tcl *fconfigure* command.
`-newline`	Option to read the final newline if it exists.
`$file`	File to read data from.

```
fputs <$option $value ...> <-nonewline> $file $string
```

`$option $value ...`	File configuration options, see Tcl *fconfigure* command.
`-nonewline`	Option to not write a final newline.
`$file`	File to write data to.
`$string`	Data to write to file.

Example 1: File import/export

Code:

```
# Export data to file (creates or overwrites the file)
fputs example.txt "hello world"
# Import the contents of the file (requires that the file exists)
puts [fread example.txt]
```

Output:

```
hello world
```

Data Conversion

This package also provides conversion utilities for different datatypes. The main datatype is matrix, or **mat**.

Matrix (mat)

The matrix (**mat**) datatype is a nested Tcl list, where each list element represents a row vector of equal length. This definition is compatible with the matrix data type provided by the ndlist package.

An example of a matrix with headers is shown below.

Example 2: Example data (**mat**):

Code:

```
set mat {{step disp force} {1 0.02 4.5} {2 0.03 4.8} {3 0.07 12.6}}
```

This format can be converted from and to all other formats, as is illustrated in the diagram below, with "**a**" and "**b**" acting as placeholders for all other datatypes.

This way, each new datatype only requires the addition of two new conversion commands: one to **mat** and one from **mat**. Then, you can convert between any datatype using **mat** as the intermediate datatype.

Table (tbl)

The table (**tbl**) datatype is a key-value paired list, with keys representing the table header, and values representing the columns. This definition is compatible with the table data type provided by the taboo package. To convert between **mat** and **tbl**, use the commands *mat2tbl* and *tbl2mat*.

```
mat2tbl $mat
```

```
tbl2mat $tbl
```

$mat	Matrix value.
$tbl	Table value.

Example 3: Example data (**tbl**):

Code:

```
puts [mat2tbl $mat]
```

Output:

```
step {1 2 3} disp {0.02 0.03 0.07} force {4.5 4.8 12.6}
```

Space-Delimited Text (txt)

The space-delimited text (**txt**) datatype is simply space-delimited values, where new lines separate rows. Escaping of spaces and newlines is consistent with Tcl rules for valid lists. To convert between **mat** and **txt**, use the commands *mat2txt* and *txt2mat*.

```
mat2txt $mat
```

```
txt2mat $txt
```

$mat	Matrix value.
$txt	Space-delimited values.

Example 4: Example data (**txt**):

Code:

```
puts [mat2txt $mat]
```

Output:

```
step disp force
1 0.02 4.5
2 0.03 4.8
3 0.07 12.6
```

Comma-Separated Values (csv)

The comma-separated values (**csv**) datatype is comma delimited values, where new lines separate rows. Commas and newlines are escaped with quotes, and quotes are escaped with double-quotes. To convert between **mat** and **csv**, use the commands *mat2csv* and *csv2mat*.

```
mat2csv $mat
```

```
csv2mat $csv
```

$mat	Matrix value.
$csv	Comma-separated values.

Example 5: Example data (**csv**):

Code:

```
puts [mat2csv $mat]
```

Output:

```
step,disp,force
1,0.02,4.5
2 0.03,4.8
3,0.07,12.6
```

Derived Conversions

Using the **mat** datatype as the intermediate datatype, data can be converted to and from any datatype. As a convenience, shortcuts are provided for conversions that use **mat** as an intermediate data format.

```
tbl2txt $tbl
tbl2csv $tbl
```

```
txt2tbl $txt
txt2csv $txt
```

```
csv2tbl $csv
csv2txt $csv
```

$tbl	Table value.
$txt	Space-delimited values.
$csv	Comma-separated values.

Example 6: Combining data conversions

Code:

```
# Convert from table to csv, using mat as an intermediate datatype.
set tbl {step {1 2 3} disp {0.02 0.03 0.07} force {4.5 4.8 12.6}}
set csv [mat2csv [tbl2mat $tbl]]; # also could use tbl2csv
puts $csv
```

Output:

```
step,disp,force
1,0.02,4.5
2,0.03,4.8
3,0.07,12.6
```

Command Index