The dataref package

Christian Dietrich 2013-2016 stettberger@dokucode.de https://github.com/stettberger/dataref

2016/11/15 v0.6

1 Introduction

Writing scientific texts is a craft. It is the craft of communicating your results to your colleagues and to the curious world public. Often your conclusions are based upon facts and numbers that you gathered during your research for the specific topic. You might have done many experiments and produced lot of data. The craft of writing is to guide your reader through a narrative that is based upon that data. But there may be many versions of that data. Perhaps you found a problem in your experiment, while already writing, that forces you back into the laboratory. After a while, the moon has done its circle many times, you return from that dark place and your methodology has improved as significantly as your data has. But now you have to rewrite that parts of the data that reference the old data points.

The dataref is here to help you with managing your data points. It provides you with macro style keys that represent symbolic names for your data points. You can reference those symbolic names with \dref, use them in calculations to have always up-to-date percentage values, define projections between sets of data points and document them. dataref also introduces the notion of assertions (\drefassert) for your results to ensure that your prosa text references fit the underlying data.

2 Usage (or 32 mice)

From the 32 mice in the experiment, 12 died.

\drefset{/med A/mice count}{32}
\drefset{/med A/recovered}{20}
From the \dref{/med A/mice count} mice in the experiment,
\drefcalc[prefix=/med A]{d(/mice count)-d(/recovered)} died.

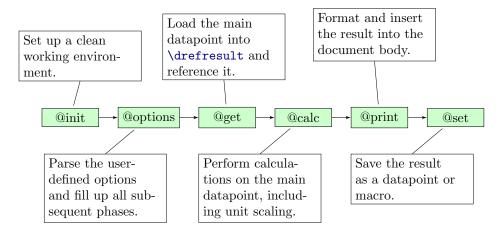


Figure 1: The DATAREF Pipeline

2.1 Design Principles

Before we jump into the description of DATAREF, let us look a little bit into the design principles of DATAREF. By understanding the principles, you will be more productive and embedding data into your document will become easier.

First of all, DATAREF is built on top of pgfkeys and pgfmath from the PGF/TiKZ macro packages. While the former provides a usable user interface to provide options to DATAREF, the later is used to perform computation on your datapoints. If you are interested into these two excellent TEX packages, please look at texdoc pgfmanual for further information.

There are two aspects of DATAREF: setting datapoints and referencing datapoints. While setting datapoints is kind of boring, we have a wide variety of options when it comes to referencing. The expansion of datapoints is done in multiples phases (see Figure 1).

The DATAREF macros are different regarding the phases they include or omit and in their default settings. In the following, we will discuss all options and macros you can use to reference your datapoints. By default, the \drefresult is always set to the result of the pipeline.

2.2 Setting Datapoints

The \drefset command is used to define the symbolic datapoints. The name of a datapoint may contain virtually all characters, including spaces and slashes. It is good practice to use a hierarchy to structure your data point names. The value might be any string, while the focus of DATAREF is on numerical datapoints. The \drefset command works outside the pipeline model (Figure 1) for performance reasons. It virtually consists only of the @options and @set stage.

```
\label{eq:continuous} $$ \drefsave[\langle options \rangle] {\langle value \rangle} $$ (@set)
```

Besides setting the datapoint, \drefsave writes it to the AUX file and, therefore, makes it available in the next LATEX run from the very beginning. The ignoremissing option is useful to reference the saved keys at an earlier place in the document.

```
/dref/set = \langle key \rangle (no default)
```

Inserts a \drefset to the @set that captures the current result and saves it to the given datapoint.

```
/dref/save = \langle key \rangle (no default)
```

Inserts a \drefsave to the @set that captures the current result and saves it to the given datapoint. Since \drefsave is used the result is available from the beginning of the document.

```
/dref/to macro = \langle macro \rangle (no default)
```

The current result is saved to the given macro.

```
32 32 \dref[set=/foo]{/med A/mice count} \dref{/foo}\\
32 32 \dref[ignoremissing]{/bar} \dref[save=/bar]{/med A/mice count}\\
5.00000000000 \drefcalc*[to macro=\myresult]{3+2} \myresult
```

```
/dref/prefix=\langle key prefix\rangle (no default, initially "")
```

On every key retrieval or setting of a datapoint this prefix is added. It is a value-typed PGF key, so operations like prefix/.append are possible.

```
\drefinput[\langle prefix \rangle] \{\langle filename \rangle\}
```

Uses \input to read the file. During the processing of this file prefix is set to the given value. This is useful to include several files that resulted from different experiments but include equal datapoint keys.

2.3 Referencing Datapoints

```
\label{lem:continuous_set} $$ \dref^*[\langle options \rangle][\langle name \rangle] $$ (@options=\{print=default\}, @get, @calc, @print, @set) $$ \\ dref[\langle options \rangle][\langle name \rangle] $$ (@options=\{print=raw\}, @get, @calc, @print, @set) $$ $$ (@options=raw\}, @get, @calc, @print, @set) $$ (@options=raw\}, @get, @calc, @print, @set) $$ (@options=raw), @get, @calc, @calc, @print, @set) $$ (@options=raw), @get, @calc, @calc, @print, @set) $$ (@options=raw), @get, @calc, @print, @set) $$ (@options=raw), @get, @calc, @print, @get, @calc, @print, @get, @g
```

This macro is used to reference a single symbolic data point. The value stored in that datapoint is inserted into the text. \dref additionally marks the data point as used; it will appear in the datagraphy (see Section 2.8). The starred variant does not attempt to parse the datapoint as a numerical value, but outputs the saved string.

```
Black Six 32 2.00 \cdot 10^{1}
```

```
\dref*{/control group/mice race}\\
\dref*{/control group/mice count}\\
\dref[sci,precision=2,zerofill=true]{/med A/recovered}
```

```
\delta ref value of \{\langle name \rangle\}
\delta ref \{\langle name \rangle\}
```

(@get, @print)

(@get)

(drefrer(\name)

Black Six

\drefvalueof{/med A/mice race}

Since \dref is not expandable, it cannot be used in all circumstances. Therefore, \drefvalueof bypasses all internal bookkeeping and provides access to the raw datapoint value. \drefref can be used to mark the datapoint as used to let it appear in the datagraphy.

```
/ dref/ignoremissing = \langle true\text{-}or\text{-}false \rangle / dref/default value = \langle value \rangle
```

(default **true**, initially **false**)

(no default, initially 1.0)

By default, DATAREF produces an error if the referenced datapoint is undefined. If ignoremissing is given, the defaultvalue is used. This key is useful in combination with \drefsave. Furthermore, it is possible to extract the missing keys from the AUX file and to retrieve it from a secondary source (e.g. database, wikidata, etc).

undefined

\dref*[ignoremissing,defaultvalue=undefined]{blah}

```
\label{eq:drefsethelp} $$ \drefhelp{\langle pattern \rangle} {\langle text \rangle} $$ \\ drefhelp{\langle name \rangle} $$
```

dataref comes with a simple method for defining documentation for data points. This help can for example be used to communicate what is the concrete semantics of the data point. This is of special interest when writter and data gatherer are not the same person. \drefsethelp takes two arguments: first a regular expression that matches the symbolic data point, second the help text.

\drefsethelp{.*/mice race}{The mice race used for experiments heavily influences the outcome of the results}

The documentation for a datapoint is obtained by using the \drefhelp macro. It checks all defined documentation strings (in linear order, first defined, first matched), and prints the first matching help text. With LuaTeX: only Lua (string.find) regular expressions are supported as patterns.

The mice race used for experiments heavily influences the outcome of the results

\drefhelp{/med A/mice race}

\drefresult

Is set in the @set phase to the result of the currently executed pipeline.

2.4 Calculations and Math Tools

```
\label{eq:calc_options} $$ \drefcalc^{(options)}_{(expression)} $$ (@calc, @print, @set) $$ \\ \drefcalc^{(options)}_{(expression)} $$ (@calc, @set) $$ \\ \drefformat^{(options)}_{(number)} $$ (@print) $$
```

The \drefcalc is the core function of calculating with data points. It is based on the pgfmath engine, but allows also the usage of symbolic datapoints within mathematical expressions. Datapoints can either are inserted into the calculations with the $\langle (path) \rangle$ or the \ata(" $\langle path \rangle$ ") notation. The starred variant of \drefcalc does not print the result, but only sets the result macros.

It is important to note, that \drefcalc always uses the /pgf/fpu environment. The FPU feature of pgfmath is used to handle large numbers, which may occur often when handling experiment data points.

```
91.67
3,200
3,200
4.00000000000
```

```
\drefcalc{(4+7)/12 * 100}\\
\drefcalc{d(/med A/mice count) * 100}\\
\drefcalc{data("/med A/mice count") * 100}\\
\drefcalc*{1+3}\\
\drefcalc*sult
```

Since the default printing mechanism of DATAREF utilizes PGF, all options from /pgf/number format can be directly used in the options. \drefformat does only the printing. For documentation on the available options, please consult the PGF manual.

```
\begin{array}{c} 0.3333 \\ 1.23 \cdot 10^8 \\ \frac{5}{8} \\ 0.63 \end{array}
```

```
\drefcalc[precision=4]{1/3}\\
\drefcalc[sci]{123456789}\\
\drefcalc[prefix=/med A/,frac]{d(recovered)/d(mice count)}\\
\drefformat[fixed zerofill, precision=2]{\drefresult}
```

2.5 Units and Unit Scaling

DATAREF allows to give the unit of a datapoint and enforces the correct combination of units when using them in calculations. DATAREF units can be arbitrary combinations of macros and strings, which allows the combination with the SIUNITX package.

```
/dref/unit = \langle unit \rangle
```

(no default)

The unit of a datapoint is loaded in the @get phase, and stored in the @set phase of the DATAREF pipeline.

```
\drefset[unit=ms]{/duration}{5555}
\drefset[unit=\joule]{/power}{1234}
```

```
/dref/unit/format = \langle formatting \ style \rangle
```

(no default, choice)

 $/dref/unit/format default = \langle formatting style \rangle$

(initially **plain**)

If a datapoint with unit is referenced, the unit is printed after the value. The formatting mechanism can be exchanged, in order to omit the unit, or to use SIUNITX for properly print it. By default, the unit/format default is set in the @init phase. If you are using SIUNITX in your document, it is safe to set the default value accordingly. Currently, the values false, plain, and siunitx are valid formatting styles.

5,555; 5,555 ms; 1,234 J

```
\drefkeys{unit/format default=siunitx}
\dref[unit/format=false]{/duration}; \dref{/duration};
\dref{/power}
```

 $/dref/unit/new scala = \langle scala \rangle$

(no default)

DATAREF allows to define scaled units and conversion between the members of the scala. A scala definition is a list of units with scaling factors between them.

```
\drefkeys{
  unit/new scala={
    1/y, 365/d, 24/h, 60/m, 60/s, 1000/ms, 1000/us, 1000/ns
  },
  unit/new scala={
    1/\kilo\joule, 1000/\joule, 1000/\milli\joule,
    1000/\micro\joule, 1000/\nano\joule
  }
}
```

 $/dref/unit/scale to = \langle unit \rangle$

(no default)

With a defined scala, you can scale to any unit on that scala automatically. In the example, we use unit to set the unit of plain value to nano joule, and scale everything to milli joule.

```
1.23 \cdot 10^{-3} \text{ mJ}
4.14 \text{ mJ}
2.14 \cdot 10^5 \text{ mJ}
2.4 \cdot 10^{-5} \text{ mJ}
```

```
\foreach \x in {1234, 4135413, 213516513245, 24} {%
  \drefformat[
    unit=\nano\joule,
    unit/scale to=\milli\joule]{\x}\\
}
```

/dref/unit/scale to auto= $\langle optimum\ number \rangle$

(default 50)

With scale to auto, the appropriate unit is chosen automatically. The algorithm tries every unit on the scala and chooses the unit, where the numerical value after

scaling is nearest to the $\langle optimum\ number \rangle$. So with a optimum number of 50, 5000 seconds are scaled to 83.33 m instead of 1.39 h.

1.23 s 68.92 m 6.77 y 24 ms

```
\foreach \x in {1234, 4135413, 213516513245, 24} {% \drefformat[unit=ms, unit/scale to auto]{\x}\\}
```

2.6 Relating Datapoints

```
\label{lem:continuous} $$ \drefrel*[\langle options \rangle] {\langle key\ or\ value \rangle} $$ (@calc, @set) $$ \\ \drefrel[\langle options \rangle] {\langle key\ or\ value \rangle} $$ (@calc, @print, @set) $$
```

Often, datapoints are set in relation to each other. This can either be done with \drefcalc or, more explicitly, with \drefrel. The different options that come along \drefrel, add steps to the @calc phase and can, therefore, be combinated with any other dref macro that includes the @calc phase.

All operations operate on the current result, which is initially the given key or value from the mandatory argument.

32.52

```
\drefrel[percent of=123]{40}
```

The different relation operations try to have a speaking name, such that the TeX code is easily understandable by the writer. This explicit notation aims to prevent common mistakes, like confusing denominator and numerator.

The starred version of \drefrel does not print any number, but only saves the result in \drefresult. Instead of only taking datapoint keys, \drefrel, as well as the relating operations, take either a key or a bare number as you can see from the example.

```
/dref/scale by = \langle key \ or \ value \rangle
```

(no default)

Scales the current value by the given factor, which must be unit less. The resulting unit is untouched.

/dref/percent

(no default)

Scales the current value by 100. The unit is not changed.

/dref/negate

(no default)

Scales the current value by -1. The resulting unit is untouched.

 $/dref/divide by = \langle key \ or \ value \rangle$

(no default)

Divides the current value by the given factor, which must be unit less. The resulting unit is untouched.

/dref/abs

(no default)

Calculate the absolute value.

```
/\text{dref/factor of} = \langle key \ or \ value \rangle (no default)

/\text{dref/percent of} = \langle key \ or \ value \rangle (no default)
```

The **factor of** operation gives the portion the current value in relation to the given base value. In easy words: a division. This macro ensures, that base and current vale have the same unit or are unit less. The result of this operation is unit less. The **percent of** operation, furthermore, scales the result with 100 to get a percentage.

In a situation, where a datapoint is the result of a changed experiment setup, the value normally shows an increased or decrease numerical value. This family of operations calculates this delta, assuming it is an increase or decrease. The factor operations scale the result to the base value, and the percent operations give this scaled value as a percentage.

```
25
0.05
5 %
```

```
\drefrel[increase from=500]{525}\\
\drefrel[increase factor from=500,fixed]{525}\\
\drefrel[increase percent from=500]{525}\,\%
```

2.7 Helper Utilities

```
\label{list} $$ \operatorname{\comma-separated\ list} {\langle key\ template \rangle} $$ \operatorname{\comma-separated\ list} {\langle key\ template \rangle} $$
```

Often different columns in a table have to be obtained from your data points. Often those rows and columns are similar. Generating parts of tables within LATEX is very tricky, so DATAREF provides you with \drefrow. This macro iterates over a comma-separated list of values and fills out a macro which is interpreted as a symbolic data point. The entries are separated with & and printed. In the starred variant the resulting text is not interpreted as symbolic name, but as a macro.

Both, unstarred and starred variant take a template (a macro body) that is expanded once for every item in the given list. The first replacement #1 is list item and the second #2 is the current index starting from 0. The unstarred variant interprets the expanded result as a datapoint key and uses \dref to expand it; the optional parameter is passed through to every invocation of \dref. The starred

variant does not wrap the result into \dref, and, therefore, is more flexible.

```
        4
        7
        21

        6
        12
        20
```

 $\drefassert{\langle expr \rangle}$

(@calc)

Sometimes the underlying data changes while you are writing. But what if your prose text relies on certain characteristics of the data. \drefassert uses a pgf-math expression that evaluates to true or false. When the assertion holds (true) nothing happens, only a terminal message is printed. When it does not hold (false) the compilation is aborted.

Of the more than thirty infected mice...

```
\drefassert{data("/control group/mice count") > 30}
Of the more than thirty infected mice...
```

 $/dref/noassert = \langle true \ or \ false \rangle$

(default **true**)

The **noassert** deescalates all assertion errors to mere warnings. This option can also be given at the **\usepackage** invication.

```
      /dref/annotate=\(annotation type\)
      (no default, initially none, choice)

      /dref/annotate=none
      (choice item)

      /dref/annotate=footnote
      (choice item)

      /dref/annotate=pdfcomment
      (choice item)

      /dref/annotate=typeout
      (choice item)
```

While writing a document it is desirable to know, what key is used, while writing the text and generating the document. Therefore DATAREF provides the possibility to annotate values. The default option **none** disables this kind of annotation. The **pdfcomment** option uses pdf annotations. Be aware that those annotations work properlyy only on a few selected PDF readers¹.

```
Black Six<sup>a</sup> 32<sup>b</sup> 33.33

a\dref*[]{/control group/mice race}
b\dref[]{/control group/mice count}
```

```
\drefkeys{annotate=footnote}
\dref*{/control group/mice race}
\dref{/control group/mice count}
\drefcalc[annotate=pdfcomment]{100/3}
```

2.8 Datagraphy

\drefusagereport

¹In doubt use Acrobat

[usagereport] [refall]

With the **usagereport** package option enabled, \drefusagereport generates a usagereport of all referenced keys. The usage report groups the keys by the help texts. If the refall package option is given, all keys are marked as referenced.

Datagraphy

	Page	Value	
/control group/mice race	3, 9	Black Six	
The mice race used for experiments heavily influences the outcome of the results			

	Page	Value
/med A/recovered	1, 3, 5, 8	20
/control group/dead after 24h	8	4
/control group/dead after 48h	8	7
/control group/recovered	8	21
/med A/dead after 24h	8	6
/med A/dead after 48h	8	12

Of all infected mice, a certain number died within a specified period of time. A certain recovered from the infection. The dead categories are cumulative and include all dead mice before.

Keys without Description	Page	Value	
/control group/mice count	1, 3, 9	32	
/med A/mice count	1, 3, 5	32	
/foo	3	32	
/bar	3	32	
blah	4	undefined	
/duration	6	$5555\mathrm{ms}$	
/power	6	1234 J	
For these keys, no description was given			