Introduction to the R package *sugaR*

Plots for presenting a diabetes diary

Steffen Möller
steffen_moeller@gmx.de

June 7, 2009

Abstract: *Diabetics cannot produce the hormone insulin (aka type I if caused by an autoimmune disorder) or the self-produced insulin is no longer effective (type II). For the treatment of either form of the disease one will eventually externally apply insulin, commonly as subcutaneous injections.*

It is crucial to determine parameters for the optimal treatment of diabetes, i.e. the insulin dosage for the basal supply and for the carbohydrates taken up with the food. But there is much more information to be taken into account, which is difficult to read out from the notes every diabetic takes. The graphs produced by this packages shall help in communicating those notes - to oneself and to one's treatment advisers.

Contents

1 Motivation

Diabetes is a complex disease - in its molecular etiology and for its impact on the patient's life. While the disease and the patient develop, continuous updates of the treatment are required. Whenever things become complicated, which is when the first diagnosis is made, when the remission ("honeymoon") ends, and with all infections or holiday-associated changes in lifestyle, one needs to think - and talk between doctors, the patient and/or the parents. One needs a diagram where the key information is collected, if possible on a single page.

The Internet offers several tools to display sugar levels and insulin dosage over time. Such are helpful, but in their presentation of sugar levels, i.e. the temporal development of those values, are commonly taken out of their respective context. This may be a result of the apparent emphasis of such tools on the collection of data. And with much data being available, the tracking of measurements over the day becomes difficult if not impossible. When the response to insulin however is changing rapidly, one is likely to solely concentrate on the past week or a shorter time for the analysis - older values are obsolete. Also, one will take as much information into account as the self-prepared notes allow to derive from them. This is more than the typical (time, blood sugar concentration, carbs uptake, units of insulin injected) quadruplets, i.e. the activity can be estimated, the onset of sleep, stress, ... arbitrary regular and irregular events.

The focus of this package is on the presentation, not on data gathering. Consequently, in order to fully exploit the functionality of this package one may need to add data to the system that is not (yet) available electronically in routine. This package is hence less likely to be used in routine. But by ignoring the manual labor for entering the data this package is also freed from all data-dependent constraints and thus allowed to investigate what the linking of a diabetes diary with a state-of-the-art statistics suite may bring for the treatment. The time required for the manual transformation of the diary's past week into the electronic form should not last longer than the travel to the doctor, the waiting time and the time the doctor is actually seen.

The package today offers to add *symbols* to the typical line plots to help investigating the interplay of glucose and insulin. The *connecting lines* allow for the tracking of the development over the day. The *thickness* of these lines represent the physical activity. This supports finding explanations for the variability of measurements and a more holistic modelling. The current version of this package is already useful to some degree, but many more ideas are already in the pipeline to be implemented (see section 4.3). And possibly there is somebody out there (you?) who can contribute something that was not yet thought about.

Note: these plots shall help communicating the contents of your diabetes diary. No more than that. Consult a doctor just the way you did before, just possibly with sugaR-crafted plots in your pocket. And: the current patient's sensations should be trusted more than yesterday's statistics.

2 The data - and a basic introduction to R data types

Admittedly, the input of data for the moment is still too tedious. Later versions of this packages shall be able to interface with other software, of which there is plenty, to collect the data manually with a graphical user interface or to read out glucose meters. And with the steadily improving mobile technology, one may soon expect a sufficiently capable handheld device.

For now, the data is expected in a form that can be directly given to the statistics suite R. This is intuitive for those who are already good at R programming, but demands some training for novices. This section presents some technical background to help preparing your own input. The actual data formats are explained together with the respective plots in section 3.

2.1 Installation

This package needs the statistics suite R to run. The R shell can be started like every other program and is available for all modern computing platforms from `http://www.r-project.org`. The package can be installed like others in the CRAN archive by the R command

```
install.packages("sugaR")
```

. There are no dependencies to other non-standard packages.

2.2 Lists and Arrays

Lists are one-dimensional arrays, where every entry is accessible under a special name - or its index. Arrays can be of any dimension. This package uses two-dimensional arrays, i.e. tables to pair the time of the day with the sugar concentrations and/or other traits of the disease. The tables don't need to be entered directly. Instead, routines u and h perform transformations from easier to edit string representations.

$u(string)$	conversion of a string in "$HH : MM$" format into a floating point number of the value $HH + (MM/60)$
$h(string)$	converts a string of comma-separated rows of a table into a matrix, for which every row first defines a time in HH:MM format and an arbitrary number of values.

Some basics for dealing with R:

`a <- 5`	– assignment of the value 5 to a variable named a
`a <- c(1,2)`	– assignment of the tuple (1,2)
`list("a"=1, "b"=c(1,2))`	– list with two heterogeneous entries
`u("18:10")`	– call of function u with string argument, function u is provided by this package to convert the string into a numeric value

The *sugar.over.time* function expects all parameters to be passed as lists. For most types of data, these are lists of days, and for every day a matrix collects values over time. To ease data entry, the function h transforms a single string into a matrix. The string shall separate measurements by commas, and multiple values measured shall be separated by blanks. The first value is always the time, converted by the above mentioned function u. This way, the string
`"8 180,9 190,10 200,11 190,12 180"`
is converted by h to what would otherwise be produced by

`matrix(c(8,180,9,190,10,200,11,190,12,180),2,byrow=TRUE)`. Here the presentation of the prior defined matrix as a table:

time	values
8	180
9	190
10	200
11	190
12	180

2.3 Invocation and documentation

To load the package into the R shell, call

```
library(sugaR)
```

with or without quotes around the package name. This is required once for every R shell started. The example data presented with figures 4 etc. can be loaded with the R command

```
data(diabetesDiary)
```

. An invocation of the *sugar.over.time* function can be observed with

```
example(sugar.over.time)
```

with further examples shown in section 3.

Documentation on R in general is available on the R project's web site. Online documentation can be spawned from within the R shell:

?*command*	– look-up of man page of that command
??*string*	– search for string in available man pages
help.search("*string*")	– like ??string
vignette()	– lists vignettes (like this document) and can open them

This package has man pages with examples for the functions *u, h* and *sugar.over.time*. The latter is the function that produces the figures. The following sections explain the preparation of the data set. The function is then invoked just the way the *help.search* function is invoked, following the schema sugar.over.time(data.$datatype$=$someValue$,...).

3 The plot

This package offers some helper functions for the data input, the core functionality however is gathered in the *sugar.over.time* function. The plot it produces shows

- the sugar concentration,
- carbohydrate consumption,
- insulin dosage
- physical activity
- and special events

over time in one single graph. The basal rates may be shown in a second plot underneath.

Once accustomed to the way the data is represented, the figure is comparatively straight-forward to interpret. The following sections introduce every graphical feature one at a time.

3.1 Sugar measurements

The sugar values are given as $(time, concentration)$ points and subsequent measurements are connected by lines.

Providing measurements of glucose concentrations

Sugar measurements are presented as lists of arrays of measurements. Every list entry shall represent a day. The name of the list entry is the date in the YYYYMMDD format. Example:

```
myGlucose<-list(
  "20090426"=h("8:47 139,11:12 75,16:18 112,19:46 71,21:48 75",ncol=2),
  "20090427"=h("8:00 212,12:12 291,13:49 236,15:42 291,17:42 112,19:15
                    74,20:35 202,22:39 139",ncol=2)
)
```

The h function needs an extra argument to learn that there should always be pairs of time and a single value to be expected. This sums up to two columns, hence the argument $ncol = 2$. To invoke the sugar.over.time function with this data, one would call

```
sugar.over.time(data.glucose=myGlucose)
```

with the effect shown as figure 1.

Days and colours

In the current implementation, continuous measurements are presented with the same colour. Every day has its very own colour, though[1]. On the plot, all colours are shown in the same

[1] Future versions will allow not only the nights, but also other events, i.e. the change of the catheter to change the colour.

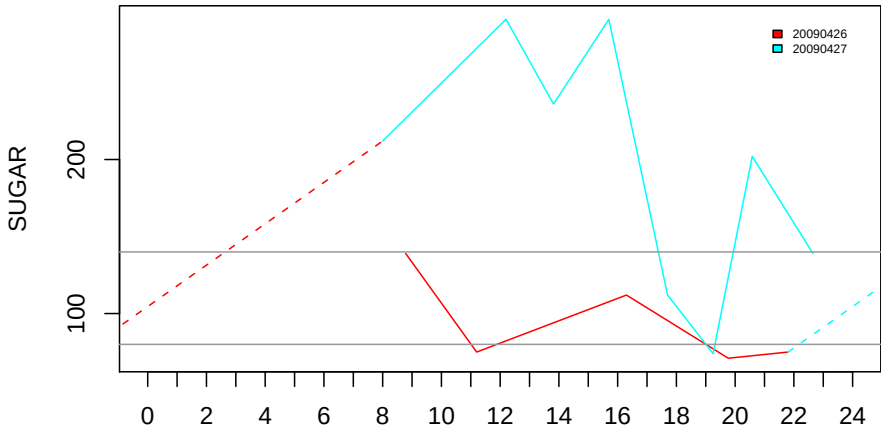

Figure 1: Blood glucose concentration - lines connect measurements, dotted between days

intensity. In the same order as the data appears in the input lists, days are first shown in colours of long wavelength (red) and then with continuously shorter wavelengths towards blue. This way, besides the time of the day, also the date is coded into the plot - albeit less obviously for the non-physicists. The plots all feature a legend in their upper-right corner to visualise the assignment of days to colours.

The lines between days are dotted. This is since over the night there are commonly fewer measurements than over the day, thus the confidence in the interpolation is much lower - if there is any. While every day in represented by a different colour, the dotted lines are presented in the colour of the respective upcoming or previous day.

3.2 Activities and Events

A strong activity may help explaining subsequently observed drops in blood glucose levels. This package may be unique in displaying this information[2].

The degree of activity is represented by the thickness of the connecting lines. The default is 1 (weak). The max is 5.

Formal specification of activities

Other than for any other kinds of data of this package, the activity needs two time points to be specified, i.e. one for the start time and one for the end:

```
myActivities<-list(
  "20090426"=list(
```

[2]Please send emails with pointers to other fine means to present the data, so they can be referenced properly in this document.

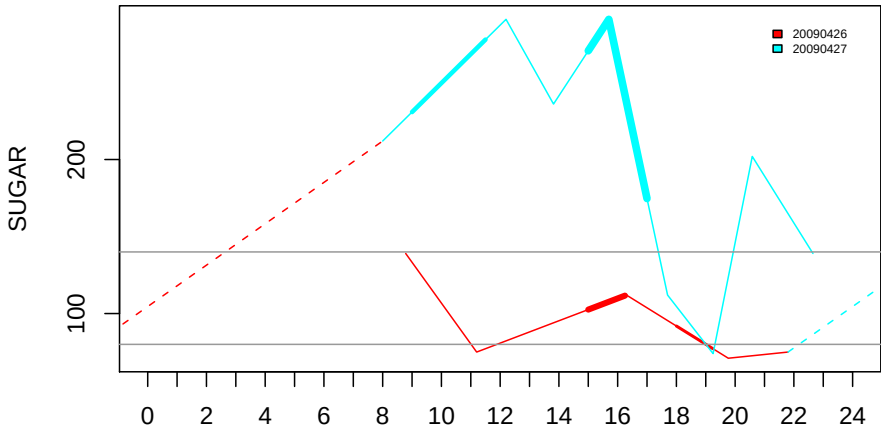

Figure 2: Sugar levels shown together with activity data (line width)

```
      list(f=u("15:00"),t=u("16:15"),a=4,c="medium-crazy"),
      list(f=u("18:00"),t=u("19:15"),a=2,c="walking the dog")
  ),
  "20090427"=list(
      list(f=u("09:00"),t=u("11:30"),a=3,c="speedy inliner"),
      list(f=u("15:00"),t=u("17:00"),a=5,c="wild")
  )
)
```

The a entry denotes the degree of activity, c is the comment. In the current implementation, the comments are not shown in the graph. To invoke the *sugar.over.time* function with this data, one would call

```
sugar.over.time(data.glucose=myGlucose, data.activities=myActivities)
```

and observe what is shown as figure 2.

Formal specification of events

Events are modelled as activities of no duration. The parameter t specifies the time point, the parameter e describes the even. An optional parameter *plot* may be set to FALSE to avoid the appearance of that parameter in the graphical representation. The default is to plot the description right next to the coordinates that represent the time of the event and the (interpolated) sugar concentration. The following code represents a single event on a single day

```
myEvents<-list(
  "20090426"=list(
```

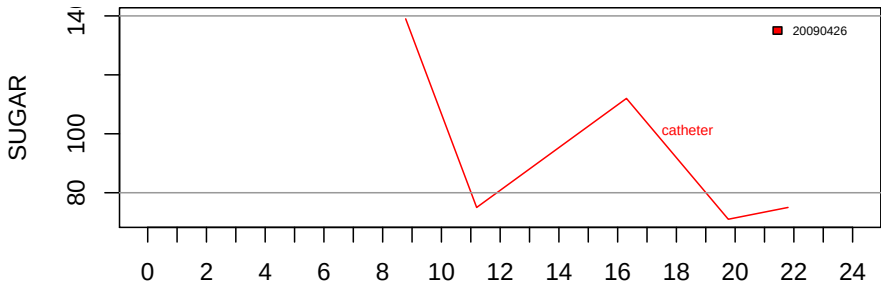

Figure 3: Sugar levels of a single day with a single event (catheter change)

```
      list(t=u("17:15"),e="catheter")
  )
)
```

and the respective plots of

```
sugar.over.time(
    data.glucose=list("20090426"=h("8:47 139,11:12 75,16:18 112,19:46 71,21:48 75",ncol=2)),
    data.events=myEvents)
```

is shown in figure 3.

The colouring of the text is the same as for all other data represented for that day.

Filtering

The filtering of days is not implemented, yet. A future version of this package will allow selecting for days with (or without) a particular event or activity. This shall further help to fine-tune the treatment. There could be sport-days and lazy-days, days with a teflon catheter or such with a needle or dependencies on the positioning of the catheter.

Another intention is to allow for days longer than 24 hours, i.e. the event itself induces the start of a new line. This is motivated to model the caring of catheters for longer than a day. Such events are not explicitly required to be shown in the plot, hence the option to hide them with "$plot = F$".

3.3 Carbohydrates and Insulin

The intake of sugar with food is coupled with the injection of insulin – such strongly that both are commonly noted together.

Data preparation for insulin and carbohydrates

The amount of insulin injected ("units") is commonly noted together with the amount of carbohydrates (gram or Broteinheiten) taken with the food. Extra insulin injected to correct for hyperglycemia is noted separately, almost always being coupled to a sugar measurement. It was nevertheless decided to have all insulin data collected together and separate it from the blood sugar measurements. This led to the following representation:

```
myIntake<-list(
  "20090426"=h("8:47 1.5 1.3,10:00 0.45,11:00 0.45, 11:12 2 0.8,12:17
     2.7 2.1,12:24 1.9 1.5,12:25 0.4 0.3,12:27 1.5 1.2,13:09 0.3
     0.2,13:30 0.45,16:18 2.5 2.0,17:09 2.5 2.0,17:29 2 1.5,18:22
     0.66666,18:28 0.3333333,19:46 0.5,19:56 0.7 0.4,20:00 1.2 0.7,20:10
     2 1.2,20:11 0.5 0.3,21:48 0.7",ncol=4),
  "20090427"=h(paste(
        "6:48 1.5 1.3 0.7,10:15 1 0.6 0.4,10:19 0.27 0.2,",
        "12:51 1.65 1.3,13:22 0.3 0.2,13:37 2 1.6,13:49 1 0.8,",
        "15:04 1 0 0,18:28 2 1.2,18:50 1 0.6,19:38 1.5 0.9,",
        "22:07 0 0 0.6",sep=""),ncol=4)
  # more data omitted
)
```

On real data, the arguments to h will be long. The day 20090427 has its argument broken into multiple lines to help with the presentation of the data in this document. The "paste" of the partial strings to a single line is not required. As it can be seen from the specification of the previous day, the newlines and blanks in the string will be ignored. But when entering your own data, it may be found preferable to use no more than a single line per day. The data produced by h for the first day is of the type

time	carbs/12	bolus	correction
8.78	1.50	1.3	0
10.00	0.45	0.0	0
11.00	0.45	0.0	0
11.20	2.00	0.8	0
12.28	2.70	2.1	0
12.40	1.90	1.5	0
12.42	0.40	0.3	0
12.45	1.50	1.2	0
13.15	0.30	0.2	0
13.50	0.45	0.0	0
16.30	2.50	2.0	0

The table has four columns for time, carbohydrates, insulin units for food and lastly those for corrections of hyperglycemia, hence the argument $ncol = 4$ to the h function.

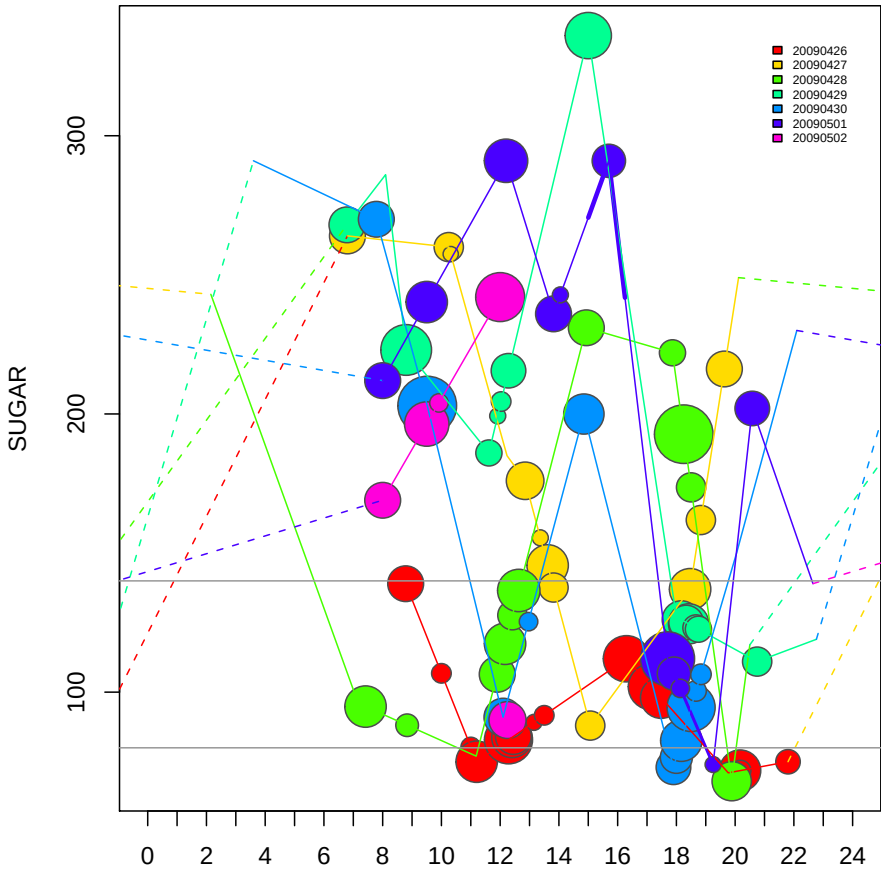

Figure 4: Sugar levels shown together with carbohydrate intake (filled circles) and activity (line width)

Plotting carbohydrates alone

To invoke the sugar.over.time function with this data, one would call
`sugar.over.time(data.glucose=myGlucose, data.intake=myIntake)`.

The intake of food is represented graphically by filled circles. The area or the radius can be set to represent the amount taken in. The size is relative to the maximal value found in the data to be presented. This renders the size invariant to the actual unit – Broteinheiten are just fine ($\frac{\text{carbohydrates [g]}}{12}$).

The data should be analysed with sufficiently large printouts. This allows e.g. for hand-written comments. To improve readability, one may decide to let the area of the circles, not their radius, reflect the sugar level (see Figure 5).

An alternative to the representation as circles is the form of *thermometers*. With only single values to be displayed, those will appear as solid boxes (Figure 6). A later plot will use thermometers to display insulin levels together with the glucose intake (Figure 8).

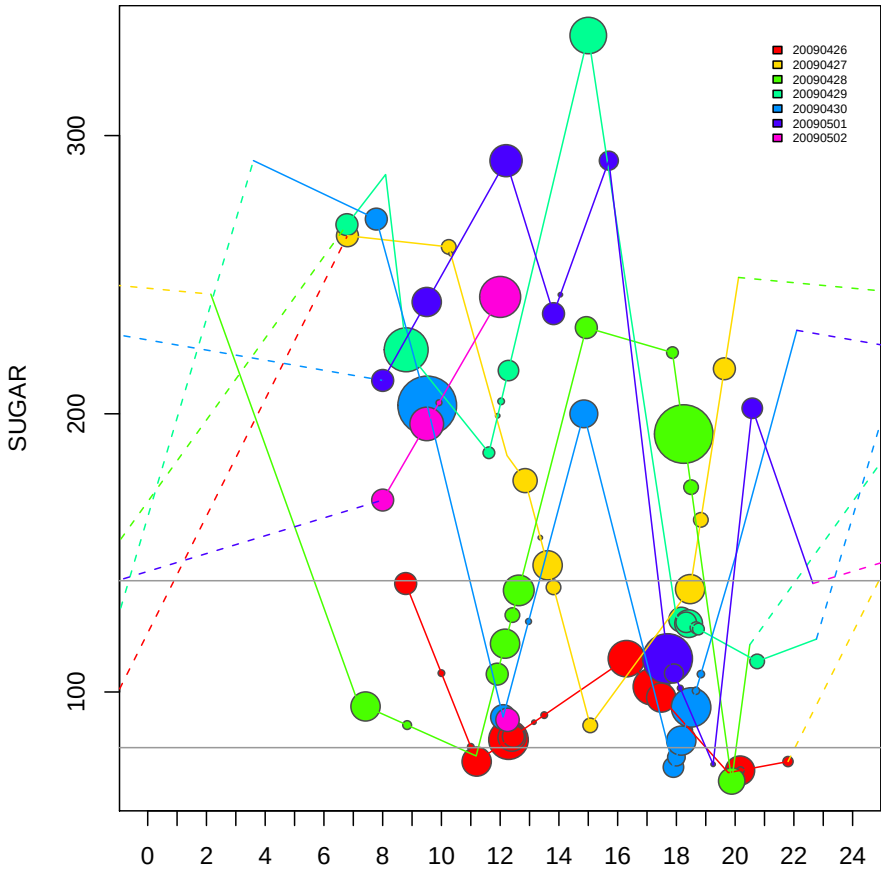

Figure 5: Blood glucose concentration - levels represented by area of circles, not by their diameter

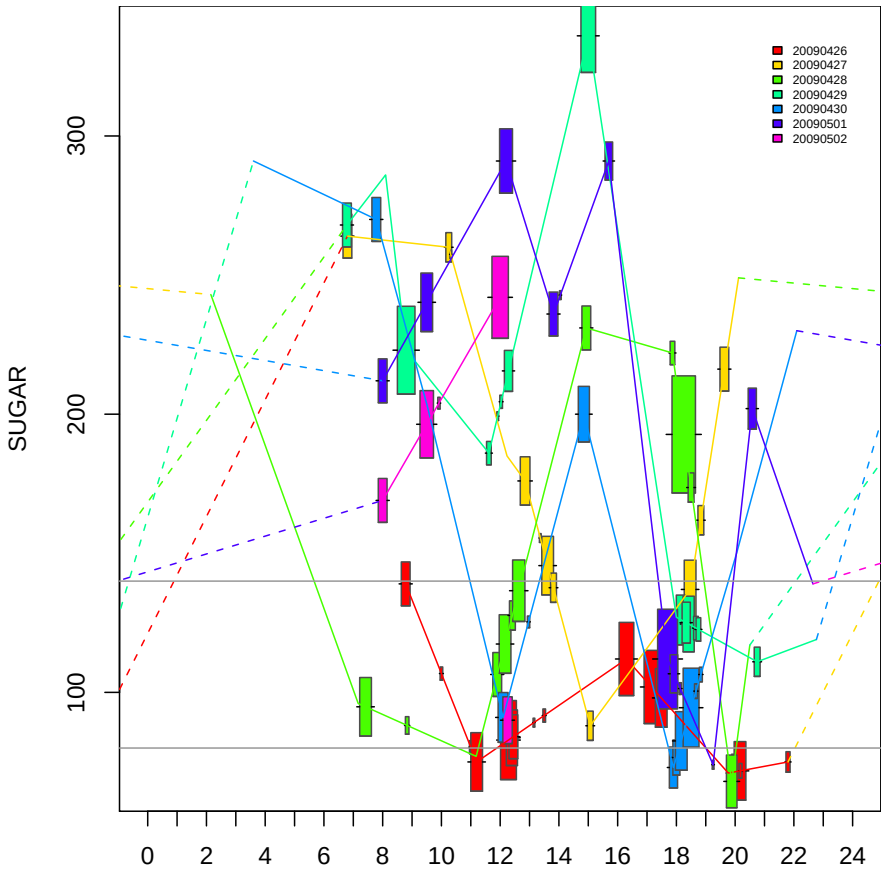

Figure 6: Blood glucose concentration shown with thermometers instead of circles which represent carbohydrate uptake by their size.

3.4 Basal rate

Carriers of insulin pumps define a time-dependent automated dosage of insulin. One may specify the start time of a particular dose (units per hour), which is kept until a the time with new instructions was reached.

Data representation

The basal rate is either a list of pairs $(starttime, value)$ or of $(duration, value)$ pairs with initial start time at midnight. This application chose the second way to represent the basal rate. The notation below allows to only specify the hourly rates.

```
myBasal<-list(
  "A'"=cbind(1,
    # hour #   0  1   2   3  4   5  6  7  8  9  10 11 12  13  14 15  16
            c(.3,.25,.25,.3,.4,.45,.5,.5,.5,.4,.15,.1,.1,.15,.15,.3,.45,

    # hour #   17  18  19  20 21 22  23
             23.45,.45,.5,.55,.6,.5,.35))
)
```

The *cbind* (short for *column-bind*) function combines vectors and returns a matrix. The example combines the vector 1 (short for $(1, 1, \ldots, 1)$) with 24 values that are explicitly written as a vector in the '`c(...)`' notation. Once combined, this will be interpreted as 24 levels for the basal rate (right column), which each lasts for one hour (left column). The name of the list entry, `A`, is the name of the profile.

Basal rate is plotted underneath

The basal rate is presented in a separate plot. If the data presented is shown on an hourly or half-hourly basis, and equal for all profiles of basal rate, then the data will be shown in a histogram-like manner. Otherwise, the data will be shown in the form of overlapping boxes. This way, arbitrary times for the starts and ends of the rates can be defined, i.e. one is not dependent on hourly specifications. The increased freedom comes at the costs of a reduced clarity. Please submit suggestions for more suitable displays of the basal rate.

The plots for sugar levels and basal rates are completely independent. I.e., there is yet no direct link between the sugar values and the selection of profiles for basal rates. This becomes an issue when the insulin pump offers multiple such profiles. An idea how to achieve this properly still needs to arise, possibly by colour coding the background or by separating the plots for the basal rate that was selected.

Plotting the basal rate without sugar levels is not supported, though. Figure 7 shows the basal rate underneath the plot already shown alone in figure 4. An important parameter that is missing for the interpretation of the insulin rates and their effect is the onset of sleep.

To invoke the sugar.over.time function with this data, one would call

```
sugar.over.time(data.glucose=myGlucose, data.intake=myIntake, data.basal=myBasal)
```

3.5 Insulin dosage

The amount of insulin given in reaction (or in preparation) of carbohydrate intake was *not* shown in the previous figures. It was omitted with the hindsight that there are established rules (factors) to which one obeys and hence it might be sufficient to know just how much was eaten. Also, the amount of compensation for high sugar concentrations is already determined as soon as one observes an event of hyperglycemia. Consequently, one does not ultimately need to explicitly represent the insulin levels graphically.

On the other hand, there needs to be somebody to first describe these rules. And it would be beneficial to have a display for their evaluation. Another point of view is the request for an optical impression on the degree of correctness with which these instructions were followed.

Factors for determining insulin dosage

Lastly, for the presentation of insulin levels, the reference is important. That reference is dynamic, it depends on the amount of carbs consumed. Commonly, this relation is proportional, time-dependent linear factors are determined. These are again pairs of start times and a single value, the factor.

```
myFactors<-list(
  "20090101"=matrix(
    c(u("0:00"),0.6,
      u("6:00"),0.8,
      u("11:00"),0.8,
      u("16:00"),0.7,
      u("18:00"),0.6),byrow=2)
)
```

The `myFactors` list only shows those days at which factors were changed. Every list element is a matrix with two columns, of which the first denotes the time of the day from which onwards a particular $\frac{\text{insulin units}}{\text{carbohydrate units}}$ ratio shall be employed. For transition days between factor settings, it may be appropriate to provide a separate entry, effective only for a single day, i.e. a recombination of the old an new settings at the time of the day that the change was implemented.

The h function could have been used rather than explicitly writing the $u(..)$ function on every row. The ordering of the days in the list and also of the times within the day is important for the subsequent interpretation of the data. The current implementation performs no checks on the semantics of the data presented - which probably it should.

To invoke the sugar.over.time function with this data, one would call

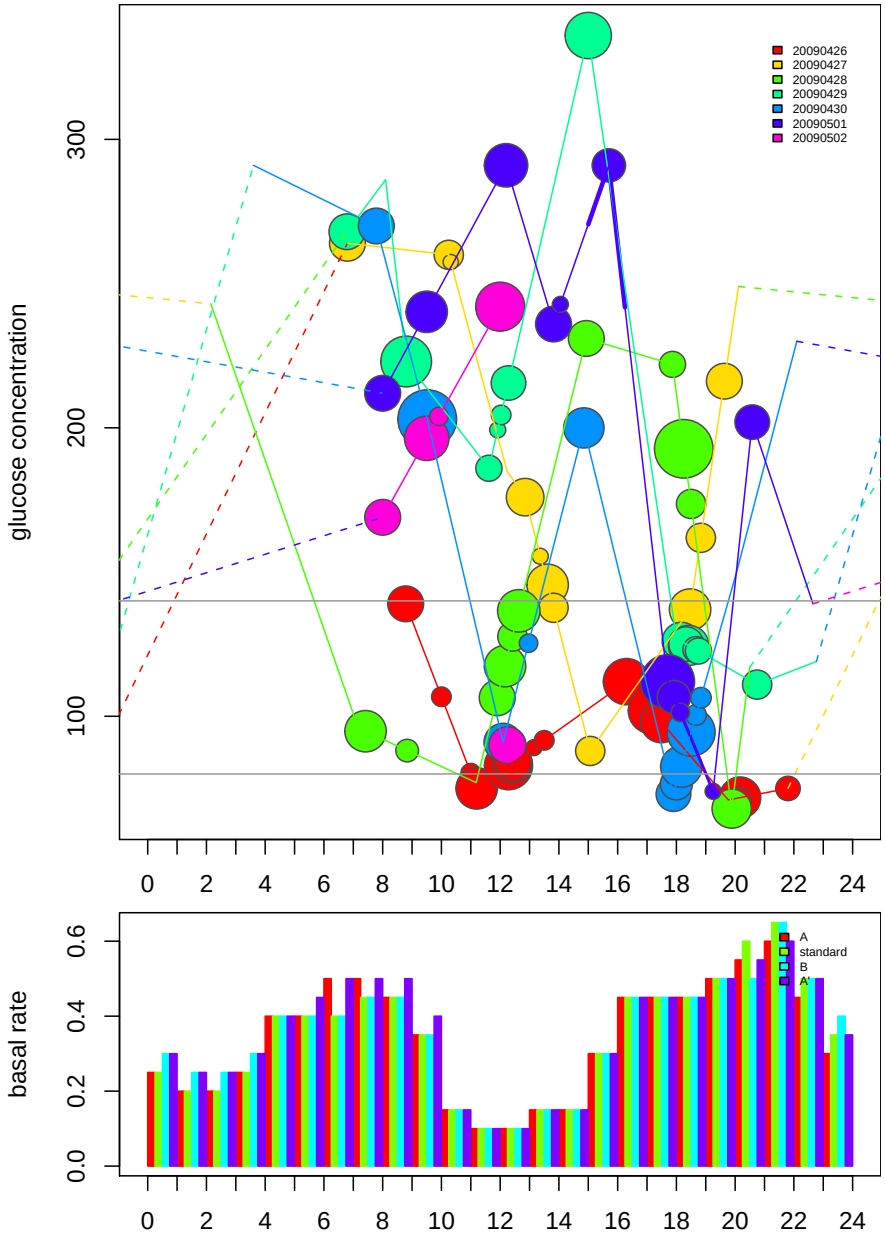

Figure 7: Sugar and food intake on top, basal rates at the bottom

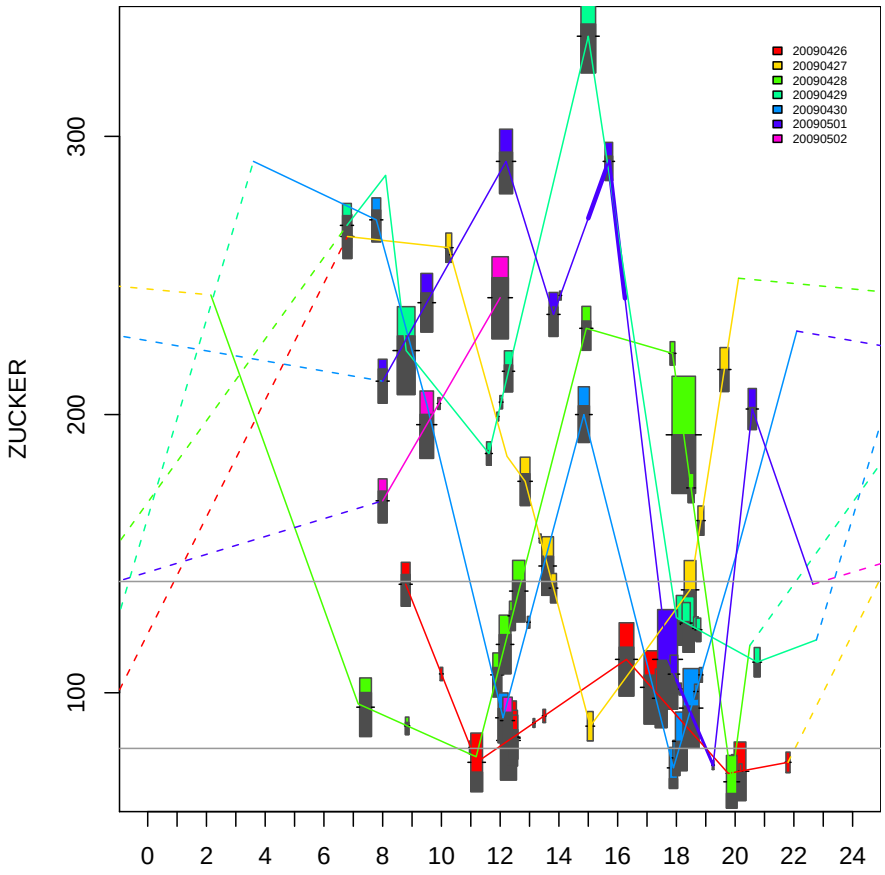

Figure 8: Carbohydrate uptake is represented by the height of the thermometers, the insulin dosage is represented by the "temperature". The Y-axis shows an all capitals simplified wording to help kids reading at least something from these plots - here localised to German (see section 4.1).

```
sugar.over.time(data.glucose=myGlucose, data.intake=myIntake,
                data.activities=myActivities,
                data.factors=myFactors)}.
```

which is shown as Figure 8. It displays the data of Figure 7 with thermometers. Different from Figure 6 these now display in black the "height" of the insulin level. At 50% of the thermometer height, the insulin dosage directly obeys the rule. The factors are passed via the `data.factors` argument.

4 Miscellaneous

4.1 A new program also for the juvenile

For younger patients it may be disturbing to see a crowd of adults stirring at some data that the kids cannot understand at all. And while still in the hospital they might not necessarily escape such conversations easily. Consequently, some little effort was made to support not only multiple languages but also easier wordings for the little ones. There should be some entertaining image placed somewhere, eventually. Kids still won't understand the graph, most likely, but they might then read a word or two and be somewhat content of understanding something, thus gaining confidence that eventually they will understand the whole thing (not that any doctor would fully understand the whole thing, yet). At the same time, the author has some confidence that the graph presented may contribute to the education in how to treat the disease at any age.

The plot of figure 7 explicitly requested the parameter `labels.language` to be set to `"english"` and the parameter `labels.type` to `"adults"`. This concerns the description of the Y axis, invisible with the current settings of the margins is the description of the X axis. Figure 8 instead set `labels.language` to `"german"` and `labels.type` to `="kids"`. One can easily observe the shorter wording and the strict use of capitals only.

4.2 Legal issues

The source code of the diagram is made publicly available under the terms of the Affero GPL-3. This shall allow everyone to improve on the code. But nobody is allowed to hide those improvements when the fruits of the extra developments appear publicly.

The Affero GPL-3 license is already very explicit about it: in no way shall the authors of this program be reliable for anything. The user of this tool is solely responsible for every decision that he or she makes.

4.3 Upcoming development and most recent changes

The following enhancements are on my to-do list,to be implemented as time permits:

- further annotation of the data

  - allowing longer days than 24 hours, trigger new "days" by these events
- ties between basal rate and the sugar levels
  - indication of temporal changes
  - indication of profile changes
- easier input of data
- data reduction
  - local sums of glucose and insulin values
  - filtering for days with similar events/activities
- suggestions for amending basal rates

Contributing with patches

The code for the plots is just above 500 lines long and not overly complicated. As a comparison, the source of this vignette has just close to 750 lines – which took about 3-5 times as long to write, though. If you have some background in programming paired with ideas, you might want to offer your improvements to appear in this package. This would also support the transfer of knowledge between the users of this package and is much appreciated.

The source code of this project is maintained publicly on the pkg-escience subversion repository[3] It can be checked out easily and anonymously with
`svn co svn://svn.debian.org/svn/pkg-escience/r-cran-sugaR` and patches are produced with `svn diff` .

Contributing with money

This program is made available freely, under the assumption that many users of this software are happily adopting some ideas to help their own or someone else's treatment. This is a good thing and every such feedback is strongly motivating. However, the author was reminded from several sides, that he could very much make use of some money, too. Hence, if you think that this work has helped to adjust the treatment, then please forward some money to the paypal account of steffen_moeller@gmx.de[4].

If you are a professional in health care, research or industry and think that this work has helped with the treatment of your patients, then please contact me for an official invoice for an amount that you are suggesting, if this is needed or wanted prior to sending money.

[3] `http://svn.debian.org/wsvn/pkg-escience/r-cran-sugaR/#_r-cran-sugaR_`

[4] `https://www.paypal.com/cgi-bin/webscr?cmd=_donations&business=steffen_moeller%40gmx%2ede&item_name=SugaR&currency_code=EUR&no_shipping=1`

Contributing with insights

Are you someone with a medical background and possibly interested in writing a tutorial on how to adjust the parameters of the treatment? This package would be extended to provide the data and the figures for the examples.