Introduction to the `scuba` package

Adrian Baddeley

October 2008

This is an introduction to the features of `scuba`, a package in R that performs theoretical calculations about scuba diving. The package supports

- creation, manipulation and plotting of dive profiles
- decompression models
- gas toxicity calculations.

Section 1 gives a quick tour of the package's functionality. Section 2 explains how to install and run the package. Section 3 is a legal disclaimer. Then the remaining sections 4–6 explain the package commands in greater detail.

1 Quick tour

A *dive profile* gives the diver's depth as a function of elapsed time during a scuba dive. See Figure 1.

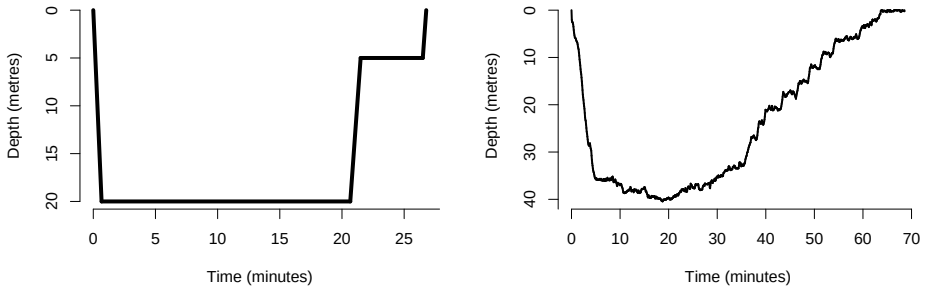


Figure 1: Examples of dive profiles. *Left:* square dive plan with safety stop. *Right:* real dive profile, uploaded from a dive computer.

Using the `dive` function in the `scuba` package, the user can create a dataset that represents any dive profile. A simple dive profile, such as a recreational dive plan or a therapeutic table, can be created by typing the depths and durations of each stage. For example a 'square' dive to 18 metres for 45 minutes (without a safety stop) is specified by:

```
> d <- dive(c(18, 45))
```

Real dive profile data, uploaded from a dive computer, can also be converted into a dive profile dataset.

A dive profile dataset `d` can be plotted simply by typing `plot(d)`. Dive profiles can be manipulated easily, for example they can be cut-and-pasted together.

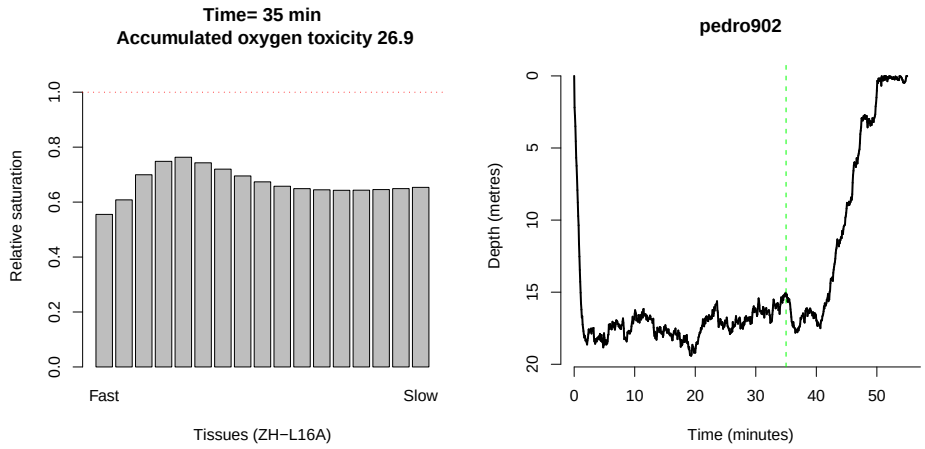
The `scuba` package performs the mathematical calculations of classical decompression theory. For any dive profile `d`, the package can compute the quantity of nitrogen dissolved in the diver's body at the end of the dive, or at each time during the dive, using the command `haldane(d)`. For example, for a dive to 18 metres for 45 minutes,

```
> d <- dive(c(18, 45))
> haldane(d)

        N2
1 2.115516
2 2.106403
3 1.901867
```

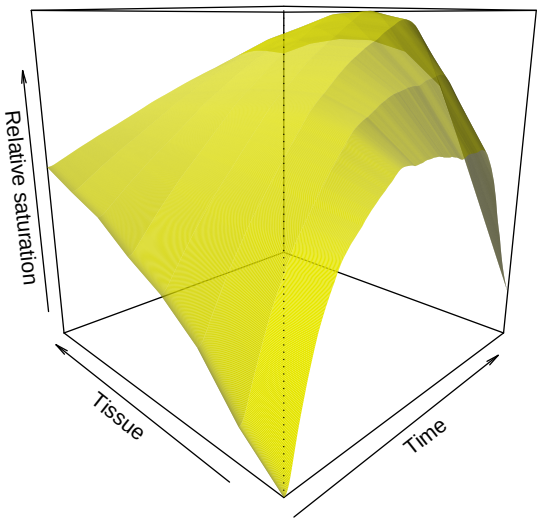
```
4 1.707818
5 1.562281
6 1.370893
7 1.253765
8 1.119592
```

The output gives the nitrogen tension (in atmospheres) in each of the 8 tissue compartments of the PADI DSAT model, at the end of the dive.

The command `showstates(d)` displays an interactive graphical window. When the user clicks on a point in the dive profile, the nitrogen tissue saturation at that time is displayed as a bar chart. Here is a screenshot:



It is also possible to plot the tissue saturations at each time during the dive, as a surface in three dimensions:



The theoretical No Decompression Limit (maximum duration of a no-decompression dive to a specified depth) can be computed by the command `ndl`. For a dive to 24 metres:

```
> ndl(24)

[1] 31.46958
attr(,"controlling")
[1] 3
```

This says that the NDL is 31.5 minutes, and the controlling tissue (the tissue that determines the NDL) is tissue 3.

In the `scuba` package, a **breathing gas** such as air, nitrox or trimix is represented by a special dataset. These gas objects are easy to specify: for example the command `nitrox(0.32)` specifies Nitrox 32 (containing 32% oxygen and 68% nitrogen).

```
> nitrox(0.32)

EANx 32
```

```
> nitrox(1)

100% O2
```

Standard nitrox calculations are available, for example to compute the equivalent air depth, maximum operating depth, and richest nitrox mix for a given depth. To find the maximum operating depth for EAN 32:

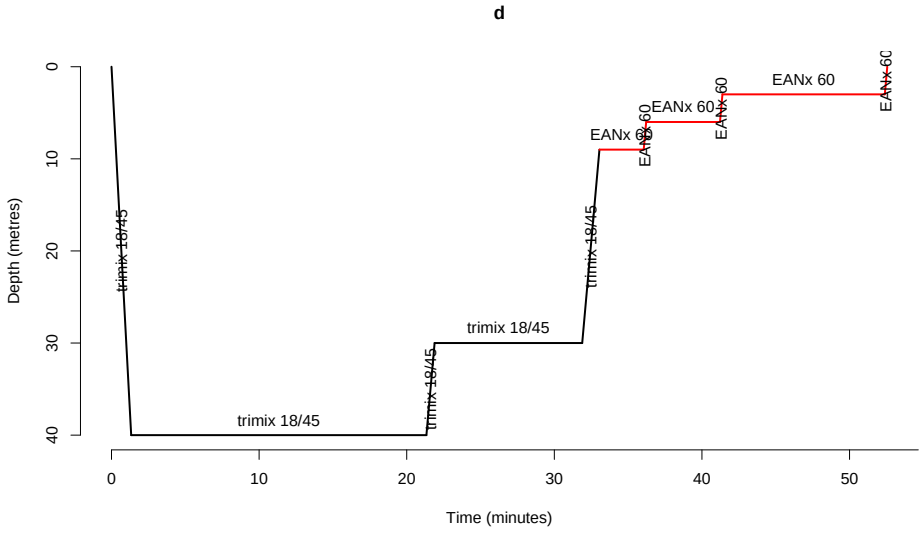
```
> mod(nitrox(0.32))

[1] 33.75
```

A dive profile contains information about the breathing gas or gases used in the dive. For example, we can specify that a dive was conducted on Nitrox EAN 32:

```
> nd <- dive(nitrox(0.32), c(18, 45))
```

The dive can be conducted using multiple tanks (cylinders) containing different breathing gases, and the diver can switch between these tanks at any time.

```
> d <- dive(trimix(0.18, 0.45), c(40, 20), c(30, 10), 9, nitrox(0.6),
+     c(9, 3), c(6, 5), c(3, 11))
> plot(d)
```



The decompression calculations work with nitrox and trimix gases. The total oxygen toxicity incurred during a nitrox or trimix dive can be computed by `oxtox`.

The breathing gas or gases in a dive profile can be changed easily, so it is easy to evaluate how a different choice of breathing gas would have affected nitrogen saturation, helium saturation and oxygen toxicity.

2 Getting started

To use the `scuba` package, you will first need to install the R system on your computer, and then install the `scuba` package within R(follow the installation instructions at `r-project.org`).

With all the software installed, start an R session and type `library(scuba)`. You should get a message like this:

```
> library(scuba)

scuba 1.2-3
Type "help(scuba)" for an introduction
Read the warnings in "help(scuba.disclaimer)"
```

The message asks you to read the disclaimer, so here it is:

3 Disclaimer

The `scuba` software library is intended for use in research and education about the mathematical and statistical basis of decompression theory. It is not designed for actual use in scuba diving and related activities. It is emphatically not suitable for use in actual diving.

Scuba diving is a dangerous activity with inherent risks of death and serious injury. No-one should attempt scuba diving without professional training, certification, supervision and regular medical assessment.

It is also dangerous for trained scuba divers to exceed the limitations of their training. Diving at altitudes above sea level, and breathing mixed gases other than air, carry increased risk and additional types of risk. Divers should seek additional, professional training and certification for such activities.

This software is not suitable for use in actual scuba diving. The software will yield numerical results for any diving activity, without giving any warning if the activity would be dangerous or fatal. Each function in the `scuba` library calculates the predictions of one theoretical model (a law of physics, a decompression model or another empirical relationship). In doing so, it does not take account of safety restrictions, other physical laws, or other important information.

The software is provided for academic interest only. It should not be used to generate diving tables or protocols related to diving. No output from this software should be misconstrued as a diving table. Only persons qualified to

supervise diving activities or qualified in hyperbaric medicine should attempt to design diving tables. Although existing published diving tables are based on theoretical models, such tables have been extensively field-tested and modified before approval. Existing tables are more conservative than the models from which they were originally derived.

The author does not warrant that the software is correct in any sense whatsoever. Even if correctly computed, the predictions of a theoretical physical model may not be correct predictions.

4 Dive profiles

In the rest of this document, we will go through the features of the `scuba` package in more detail.

A *dive profile* gives the diver's depth as a function of elapsed time during a scuba dive. See Figure 1. This section explains how to create and manipulate dive profiles in the `scuba` package.

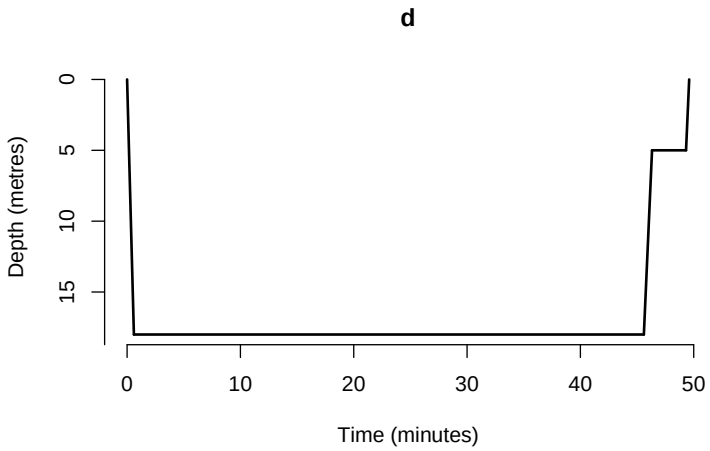
4.1 The `dive` command

The command `dive` creates an object representing a dive profile. For example, the following command creates a dive to 18 metres for 45 minutes with a 3-minute safety stop at 5 metres:

```
> d <- dive(c(18, 45), c(5, 3))
```

The resulting dataset `d` is an object of class `"dive"`. It can be plotted as a conventional dive profile graph by executing the command `plot(d)`.

```
> plot(d)
```



A dive object can be printed as a table of waypoint depths and times by simply typing its name:

```
> d
```

```
Dive profile
gas: air
   time depth
1  0:00     0
2  0:36    18
3 45:36    18
4 46:19     5
5 49:19     5
6 49:36     0
```

A summary of the dive (with such information as the average depth, maximum depth and the main stages of the dive) can be printed by typing `summary(d)`.

```
> summary(d)

Dive to 18 metres on air
Total dive time: 49.6 minutes
Stages:
  depth time
1    18   45
2     5    3
Mean depth 16.9 metres
```

4.2 Creating a synthetic dive profile

A dive profile is piecewise linear: it is a series of *stages* that join successive *waypoints*. Each waypoint is specified by the depth and elapsed time when it is reached. The stage between two waypoints is either a sojourn at a fixed depth, or an ascent or descent at a constant rate.

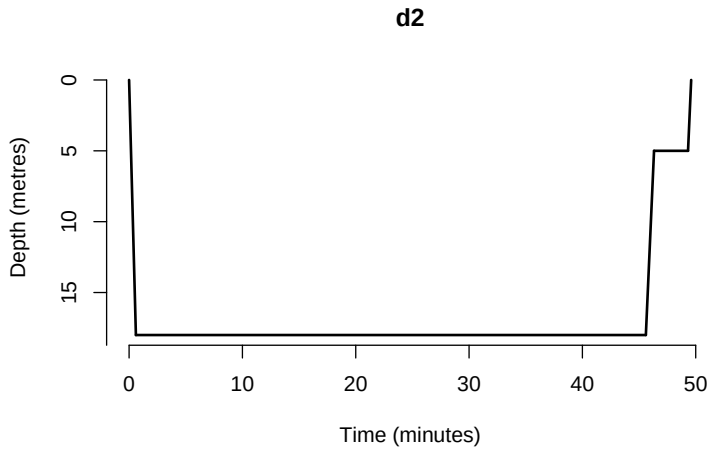
To create a dive plan or a synthetic dive profile, use the function `dive`, typing in the depths of each waypoint and the duration of time between each successive waypoint.

The function `dive` interprets its arguments as a sequence of actions or events occurring during the dive. If an argument is a vector of length 2, it is interpreted as `c(depth,time)` specifying the depth and duration of a stage of the dive. If the argument is a single number, it is interpreted as a depth, meaning that the diver ascends or descends to this depth. For example,

```
> d2 <- dive(c(18, 45), c(5, 3))
```

specifies a dive to 18 metres for 45 minutes followed by a safety stop at 5 metres for 3 minutes:

```
> plot(d2)
> d2

Dive profile
gas: air
   time depth
1  0:00     0
2  0:36    18
3 45:36    18
4 46:19     5
5 49:19     5
6 49:36     0
```

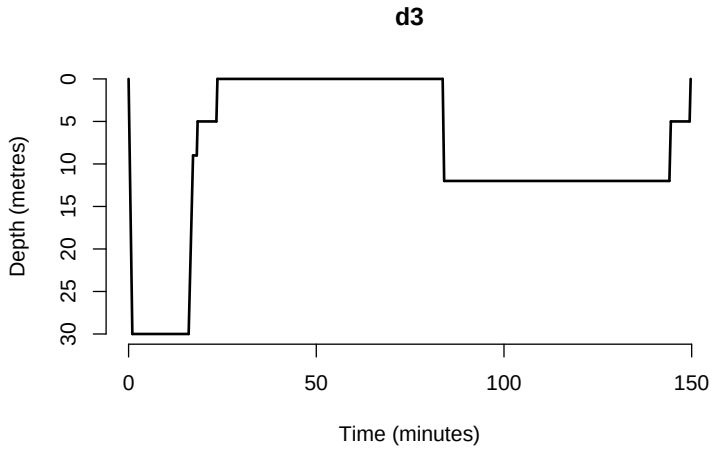


Multilevel dives with any number of stages can be specified in the same way. A dive object may also include periods spent at the surface (depth zero) and may therefore represent a succession of dives separated by surface intervals. For example,

```
> d3 <- dive(c(30, 15), c(9, 1), c(5, 5), c(0, 60), c(12, 60),
+     c(5, 5))
```

represents two dives (with safety stops) separated by a one-hour surface interval:

```
> plot(d3)
```



By default, the function `dive` fills in some details about the dive. It assumes that the diver breathes compressed air; the dive starts and ends at the surface (depth zero); the diver descends at the default descent rate of 30 metres per minute; and the diver ascends at the default ascent rate of 18 metres per minute. These defaults can be changed by giving extra arguments to the function `dive`.

Dive profiles can also be modified after they are created: see below.

4.3 Real Dive Profiles

Dive profiles may also be uploaded from your dive computer and studied in the `scuba` package. There are three steps:

1. read the data from a file into R
2. convert the dataset into the right format in R
3. pass the dataset as an argument to the function `dive`.

Read the data from a file into R

To read data from a file into an R session, we use the basic capabilities of R. If you are not familiar with R, please consult one of the many basic user guides to R.

Typically you will use one of the functions `read.table` or `read.csv` to read data from a text file. If your data are stored in a text file, as columns

of data separated by white space, use `read.table`. If your data are stored as numbers separated by commas, use `read.csv`.

If your data are in a spreadsheet file, use your spreadsheet program to Export or Write the data as a csv (comma-separated values) text file. Then in R use `read.csv` to read the data into R.

For example, suppose your data are stored in a text file `myfile.txt`. The top of the file looks like this:

```
  time   depth   temp   bar    RBT   WL
 "0:00"   0.00   24.8  199.0   99    14
 "0:04"   1.90   24.8  198.8   99    14
 "0:08"   2.76   24.8  198.8   99    14
 "0:12"   3.70   24.8  198.8   99    14
 "0:16"   4.66   24.8  198.5   98    14
 "0:20"   5.50   24.8  198.5   96    14
 "0:24"   6.82   24.8  198.5   93    14
```

To read these data into R, type

```
> mydata <- read.table("myfile.txt", header = TRUE, as.is = TRUE)
```

The argument `header=TRUE` tells R that the first line of the file is a header, containing text labels for the columns. The argument `as.is=TRUE` ensures that the character strings representing the elapsed time (`"0:04"` and so on) will be stored as character strings and not converted to another format.

Convert to the right format in R

The uploaded profile data should now be converted to a `data.frame` with two columns, the first column containing the elapsed time and the second column containing the depth (in metres) recorded at each time.

The elapsed times can be either a vector of character strings in minutes-and-seconds format `mm:ss` or hours-minutes seconds `hh:mm:ss`, or a vector of integer times measured in *seconds* of elapsed time, or an object of class `difftime` containing the elapsed times in any time unit.

Continuing our example, the dataset `mydata` is already a data frame in R, but it has too many columns:

```
> head(mydata)
```

```
  time depth temp   bar RBT WL
1 0:00  0.00 24.8 199.0  99 14
2 0:04  1.90 24.8 198.8  99 14
3 0:08  2.76 24.8 198.8  99 14
4 0:12  3.70 24.8 198.8  99 14
5 0:16  4.66 24.8 198.5  98 14
6 0:20  5.50 24.8 198.5  96 14
```

All we need to do is to extract the first two columns, which contain the elapsed time and the depth:

```
> mydf <- mydata[, 1:2]
```

Note the comma. As a last check:

```
> head(mydf)
```

```
  time depth
1 0:00  0.00
2 0:04  1.90
3 0:08  2.76
4 0:12  3.70
5 0:16  4.66
6 0:20  5.50
```

```
> is.character(mydf[, 1])
```

```
[1] TRUE
```

We have confirmed that the data frame `mydf` is in the required format.

Pass data to the function `dive`

Finally we pass this data frame as an argument to the function `dive`:

```
> d <- dive(mydf)
```

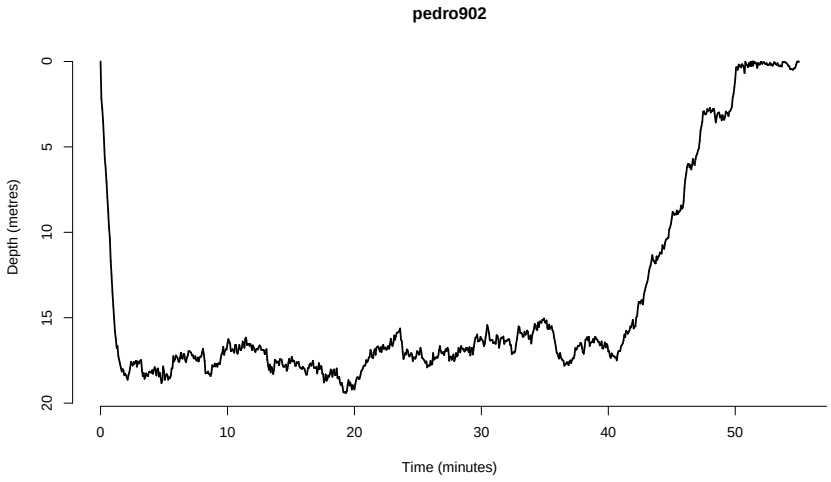
Another example of such a data frame, uploaded from a dive computer, is provided in the `baron` dataset supplied with the package. This is a profile from a dive on the *Baron Gautsch* wreck in Croatia, conducted by Vittorio Broglio. Try the following:

```
> data(baron)
> mydf <- baron[, 1:2]
> baronprof <- dive(mydf[, 1:2])
```

4.4 Installed dive profiles

The package also provides 11 real dive profiles that have already been converted to `"dive"` objects. They were kindly supplied by Pedro Antonio Neves.

```
> data(pedro)
> plot(pedro902)
```



4.5 Manipulating dive profiles

Dive profiles can also be manipulated after they are created. This allows you, for example, to modify the deepest portion of a dive (diving to a deeper depth or for a longer duration), to abort a dive prematurely, to cut-and-paste several dives together, or to consider the tissue saturation incurred by a particular segment of a dive.

The commands `depths.dive` and `times.dive` extract the depths and elapsed times at each waypoint during the dive.

```
> d <- dive(c(30, 20), c(5, 3))
> depths.dive(d)

[1]  0 30 30  5  5  0

> times.dive(d)
```

```
[1]  0.00000  1.00000 21.00000 22.38889 25.38889 25.66667
```

The depths can be modified using `depths.dive<-`. In the example above, `d` is a dive to 30 metres for 20 minutes, starting and finishing at the surface. To change the depth of the bottom stage to 35 metres, we could type

```
> depths.dive(d) <- c(0, 35, 35, 5, 5, 0)
> d
```

```
Dive profile
gas: air
   time depth
1  0:00     0
2  1:00    35
3 21:00    35
4 22:23     5
5 25:23     5
6 25:40     0
```

Thanks to the wonderful features of R, we could alternatively have typed

```
> depths.dive(d)[2:3] <- 35
```

which means that the depths of the second and third waypoints are reset to 35 metres.

Similarly the elapsed times can be modified using `times.dive<-`. It may be more convenient to use the functions `durations.dive` and `durations.dive<-` which give the duration of each stage (the time between two successive waypoints). For example

```
> durations.dive(d)[2] <- 25
```

means that the diver now spends 25 minutes at the bottom instead of 20 minutes.

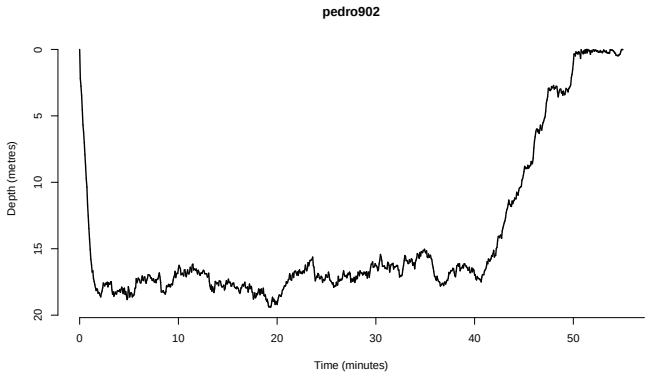
To extract only part of a dive profile, use `chop.dive`:

```
> chop.dive(d, 0, 10)
```

```
Dive profile
gas: air
   time depth
1  0:00     0
2  1:00    35
3 10:00    30
```

To paste together two dive profiles or fragments of dive profiles, simply give them as arguments to `dive`. For example, suppose we want to explore the effect of adding an extra safety stop at 9 metres in the dive `pedro902`.

```
> data(pedro)
> tim <- times.dive(pedro902)
> dep <- depths.dive(pedro902)
> plot(pedro902)
```



We need to determine the time point at which the safety stop should be inserted. This is the last time at which the diver is deeper than 9 metres:

```
> t9 <- max(tim[dep >= 9])
> t9
```

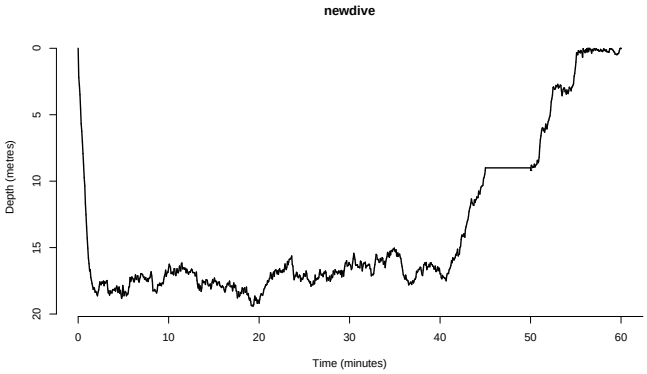
```
[1] 45
```

Cut the dive into two pieces:

```
> before <- chop.dive(pedro902, 0, t9)
> after <- chop.dive(pedro902, t9)
```

Finally paste them together with the new safety stop:

```
> newdive <- dive(before, c(9, 5), after)
> plot(newdive)
```


newdive
Depth (metres)
0
5
10
15
20
0
10
20
30
40
50
60
Time (minutes)

5 Decompression Calculations

5.1 Overview

The `scuba` package performs the mathematical calculations of decompression theory:

- the theoretical No Decompression Limit (maximum duration of a no-decompression dive to a specified depth) can be computed by `ndl(depth)`
- the quantity of nitrogen dissolved in the diver's body after a dive `d` can be computed by `haldane(d)`
- the quantity of nitrogen dissolved in the diver's body at each instant **during** a dive `d` can be computed by `haldane(d, progressive=TRUE)` or plotted interactively by `showstates(d)`.

These calculations are based on the classical theory of decompression originated by Haldane (Boycott et al, 1908).

Bubble theory calculations are not yet implemented.

5.2 Model parameters

In 'Haldane' calculations, the diver's body is idealised as a set of independent compartments, each connected directly to the breathing gas, and governed by classical (exponential) diffusion.

The model parameters (the number of compartments, their diffusion rates, and the maximum tolerated nitrogen tension in each compartment) may be chosen by the user. By default, the model parameters are taken from the DSAT model which is the basis of the PADI Recreational Dive Planner. Alternatively, the user can choose from a variety of standard compartment models using the command `pickmodel`, or construct a new model using `hm`.

```
> m <- pickmodel("USN")
> m

Haldane type decompression model
Name: USN
6 compartments
inert gas: N2
  N2.HalfT    N2.M0 N2.dM
1        5 3.185689  2.27
2       10 2.695583  2.01
```

```
3       20 2.205477  1.67
4       40 1.715371  1.34
5       80 1.592844  1.26
6      120 1.562213  1.19
```

5.3 No-decompression limits

No-decompression limits (the maximum duration of a no-decompression dive to a given depth) can be calculated using the function `ndl`. For example `ndl(30)` gives the theoretical NDL for a dive to 30 metres, predicted by the DSAT model. To use the classical US Navy model instead, type `ndl(30, model="USN")` or `ndl(30, model=pickmodel("USN"))`.

```
> ndl(30, model = "USN")
```

```
[1] 23.51394
attr(,"controlling")
[1] 2
```

The result states that the NDL is 23.5 minutes and the controlling tissue (the tissue which reaches saturation at 23.5 minutes) is tissue number 2 in the USN model.

5.4 Tissue saturations

The nitrogen tension (the quantity of dissolved nitrogen, in atmospheres absolute) in the diver's body after a dive, can be calculated using the function `haldane`. If `d` is a dive object then `haldane(d)` returns a data frame containing the nitrogen tissue tensions (ata) at the end of the dive, in each of the 8 tissue compartments of the DSAT model.

```
> d <- dive(c(18, 60), c(5, 5))
> haldane(d)
```

```
        N2
1 1.647518
2 1.861677
3 1.884928
4 1.777946
5 1.664692
6 1.485508
7 1.361945
8 1.209198
```

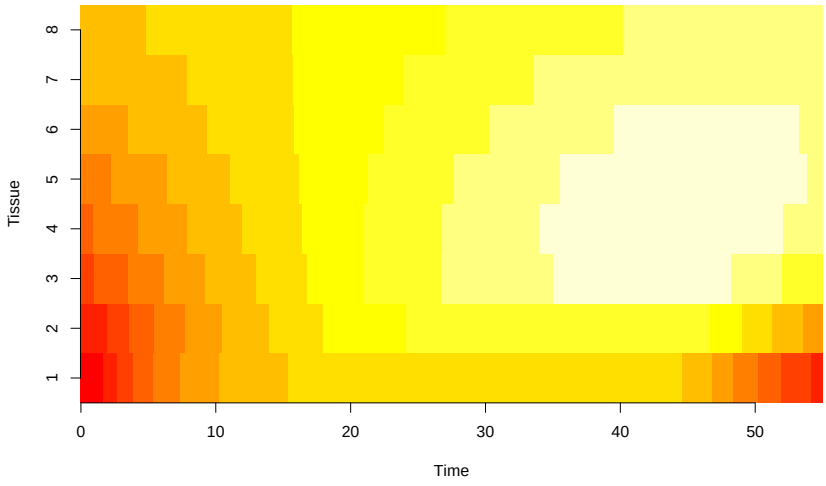
To use the US Navy model instead, type `haldane(d, "USN")` or `haldane(d, pickmodel("USN"))`.

The *relative* tissue tension is the tissue tension expressed as a fraction of the maximum tissue tension tolerated at the surface (the surfacing M-value). To obtain relative tissue tensions, use the argument `relative=TRUE`:

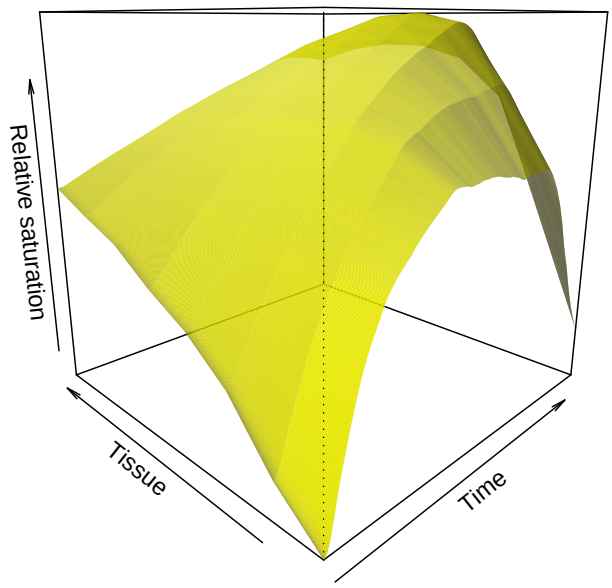
```
> haldane(d, relative = TRUE)

[1] 0.5428430 0.7350787 0.9199488 0.9715908 0.9751578 0.9427660 0.9035170
[8] 0.8411569
```

To compute the nitrogen tissue tensions at each waypoint during the dive, use `haldane(d, progressive=TRUE)`. This produces an array of numbers, which is best visualised as a surface or as a colour image:

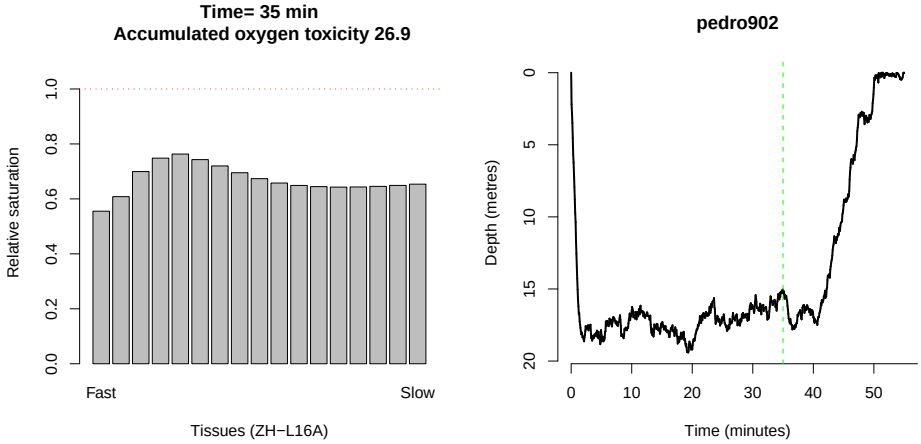
```
> data(pedro)
> h <- haldane(pedro902, progressive = TRUE, relative = TRUE)
> tim <- times.dive(pedro902)
> tiss <- row.names(as.data.frame(pickmodel("D")))
> ntiss <- length(tiss)
> image(tim, 1:ntiss, h, xlab = "Time", ylab = "Tissue", axes = FALSE)
> axis(1)
> axis(2, at = 1:ntiss, labels = tiss)
```



```
> persp(tim, 1:ntiss, h, theta = -45, shade = 0.5, col = "yellow",
+     border = NA, xlab = "Time", ylab = "Tissue", zlab = "Relative saturation")
```



Alternatively, to visualise the nitrogen tissue tensions during the dive, use the interactive function `showstates`. This plots the dive and waits for you to click on a position in the graph. The tissue tensions at that instant are displayed as a bar plot. Here is a screenshot:



5.5 Oxygen toxicity

The total (pulmonary) oxygen toxicity incurred during a dive can be computed by `oxtox`.

```
> oxtox(pedro902)

[1] 33.86758
```

This returns a number in the mysterious OTU (oxygen toxicity units). The maximum tolerable dose per day is usually reckoned as 1500 OTU. Allowing 650 OTU for recompression therapy implies a working maximum of 850 OTU per day.

6 Technical diving

6.1 Gases

A **breathing gas** is represented by an object of class `"gas"`. The object `air` is a representation of compressed air (21% oxygen, 79% nitrogen) as an object of this class. (Don't reassign another value to this object!!!)

Nitrox mixtures (mixtures of oxygen and nitrogen) can be represented using the function `nitrox`.

```
> nitrox(0.36)

EANx 36

> nitrox(1)

100% O2

> nitrox(0.21)

air
```

Trimix (a mixture of oxygen, nitrogen and helium) can also be represented, using the command `trimix`. For example, Trimix 15/50 (containing 15% oxygen, 50% helium and 35% nitrogen) is represented by `trimix(0.15, 0.5)`.

```
> trimix(0.18, 0.45)

trimix 18/45
```

There are methods for `print` and `summary` for gas objects. The print method just prints the name of the gas, as shown above. The summary method is a bit more informative:

```
> summary(nitrox(0.36))

EANx 36 (36% oxygen, 64% nitrogen)
Maximum operating depth 28.9 metres
```

Standard nitrox calculations are also available:

`ead`	equivalent air depth
`mod`	maximum operating depth
`maxmix`	richest nitrox mix for a given depth

To find the equivalent air depth for Nitrox 32 at 24 metres,

```
> ead(24, nitrox(0.32))

[1] 19.26582
```

To find the maximum operating depth for Nitrox 36, with the partial pressure of oxygen at most 1.5 ata,

```
> mod(nitrox(0.36), 1.5)

[1] 31.66667
```

To find the richest Nitrox mix for a dive to 40 metres

```
> maxmix(40, 1.5)

EANx 30
```

6.2 Diving on different gases

Every `"dive"` object contains information about the breathing gas or gases used in the dive. The default breathing gas is air.

As mentioned earlier, the function `dive` interprets its arguments as a sequence of actions or events occurring during the dive. If an argument is a vector of length 2, it is interpreted as `c(depth,time)` specifying the depth and duration of a stage of the dive. If the argument is a single number, it is interpreted as a depth, meaning that the diver ascends or descends to this depth.

Each argument to `dive` may also be a "gas" object, like `nitrox(0.32)`, which means that the diver switches to this gas. For example,

```
> dive(nitrox(0.32), c(30, 20))

Dive profile
gas: EANx 32
   time depth
1  0:00     0
2  1:00    30
3 21:00    30
4 22:40     0
```

means a dive to 30 metres for 20 minutes conducted on EAN 32 (Nitrox 0.32) from start to finish. The command

```
> dive(c(30, 20), 5, nitrox(0.36), c(5, 3))

Dive profile
   time depth     gas
1  0:00     0     air
2  1:00    30     air
3 21:00    30     air
4 22:23     5 EANx 36
5 25:23     5 EANx 36
6 25:40     0 EANx 36
```

creates a dive on air to 30 metres for 20 minutes, ascending to 5 metres while breathing air, then switching to EAN 36 for a safety stop at 5 metres for 3 minutes.

6.3 Important tip

If you specify a dive profile on nitrox or trimix, and if part of the dive profile is at the surface (depth zero), then **the package will not assume you breathe air at the sur**
The package doesn't automatically know whether you continued breathing from the regulator when you reached the surface. It is equally plausible that the diver removed the regulator and began breathing air at the surface, or switched to a snorkel for a surface swim, or breathed from the regulator for a surface swim. It's perfectly sensible for a diver to conduct a decompression stop on pure oxygen at 3 metres, then to surface and continue breathing pure oxygen at the surface. So the following two dive profiles are different:

```
> dive(nitrox(0.25), c(30, 20), c(5, 3), c(0, 20))

Dive profile
gas: EANx 25
   time depth
1  0:00     0
2  1:00    30
3 21:00    30
4 22:23     5
5 25:23     5
6 25:40     0
7 45:40     0

> dive(nitrox(0.25), c(30, 20), c(5, 3), 0, air, c(0, 20))
```

```
Dive profile
   time depth     gas
1  0:00     0 EANx 25
2  1:00    30 EANx 25
3 21:00    30 EANx 25
4 22:23     5 EANx 25
5 25:23     5 EANx 25
6 25:40     0     air
7 45:40     0     air
```

The user must decide whether the breathing gas at the surface is air or some other gas.

6.4 Tank list

A dive object has a *tank list* which is a list of the tanks of breathing gas that were used (or were available to be used) during the dive. The function `tanklist` returns this list, and the function `tanklist<-` changes the list.

For example,

```
> d <- dive(c(30, 20), c(5, 5))
```

is a dive conducted using air. To modify it to a dive that used nitrox EANx 32, simply type

```
> tanklist(d) <- list(nitrox(0.32))
```

Here is a dive conducted using air (tank 1) for the deep section and EANx 50 (tank 2) for the decompression stops at 6 metres and 3 metres.

```
> d <- dive(air, c(30, 40), 6, nitrox(0.5), c(6, 3), c(3, 3))
```

To change the contents of tank 1 to EANx 32, type

```
> tanklist(d) <- list(nitrox(0.32), nitrox(0.5))
```

or just

```
> tanklist(d)[[1]] <- nitrox(0.32)
```

You can also associate a meaningful name with each tank. Just give names to the entries in the tank list, for example

```
> tanklist(d) <- list(deep = nitrox(0.32), deco = nitrox(0.5))
```

or

```
> names(tanklist(d)) <- c("deep", "deco")
```

Perhaps the most readable way to specify the gases in a dive is to give them as arguments to the `dive` command. You specify the tank list as the argument `tanklist`, and switch between tanks by including an argument of the form `tank=number` or `tank=name`.

```
> TL <- list(travel = trimix(0.18, 0.45), deco = nitrox(0.6))
> d <- dive(tanklist = TL, tank = "travel", c(30, 40), 6, tank = "deco",
+     c(6, 3), c(3, 3))
```

6.5 Tank switching

Tank switching and selection, i.e. which tank is actually used at each stage of the dive, is specified by the function `whichtank`. The command `whichtank(d)` returns a vector of integers or character strings, identifying which tank in the tank list is in use at each waypoint during the dive. That is, `whichtank(d)[i]` is the tank in use at the `i`th waypoint during the dive. The vector `whichtank(d)` has the same length as the vectors `depths.dive(d)` and `times.dive(d)`.

```
> whichtank(d)
```

```
[1] travel travel travel deco   deco   deco   deco   deco
Levels: travel deco
```

To change the selection of tanks at each stage during the dive, use the function `whichtank<-`. For example, to change the dive `d` so that the deco gas is only used at the 3-metre stop, type

```
> whichtank(d) <- ifelse(depths.dive(d) < 3, "travel", "deco")
```

Alternatively

```
> whichtank(d)[depths.dive(d) > 3] <- "travel"
```

would select the travel gas for all parts of the dive deeper than 3 metres.

6.6 Decompression calculations

Decompression calculations (`haldane`, `ndl`, `showstates`) also work with nitrox and trimix dives.

Decompression calculations with trimix require a Haldane model that includes parameters for Helium diffusion. Use `pickmodel("Z")` to select the Buehlmann ZH-L16A model, or `hm` to create a new model that includes Helium diffusion.

The total oxygen toxicity incurred during a nitrox or trimix dive can also be computed by `oxtox`.

Acknowledgements

The package was written by Adrian Baddeley `<adrian@maths.uwa.edu.au>` with generous contributions and feedback from Vittorio Broglio and Pedro Antonio Neves.