# The `biocon` package

Pieter Edelman

August 18, 2001

**Abstract**

The `biocon` package handles the typesetting of scientific species names. There are different modes of showing these names. Multiple occurances in the same document are taken care of.

## Contents



## 1 The built-up of the package

According to [2], each package foolows a standard built-up, which is:

⟨*biocon.sty*⟩≡
⟨*Identification part*⟩
⟨*Declaration of options*⟩
⟨*Execution of options*⟩
⟨*Package loading*⟩
⟨*Main code*⟩

The identification part is easy. This package is designed exclusively for LaTeX 2$_\varepsilon$, and it provides the `biocon` package.

⟨*Identification part*⟩≡
```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{biocon}[2001/08/18]
```

## 2 The user interface

### 2.1 Brief syntax

⟨*Main code*⟩≡
⟨*Declarations*⟩
⟨*The user interface*⟩
⟨*Internal workings*⟩

The `biocon` package automates the process of typesetting biological species names. Different conventions are followed for animals, bacteria, plants and fungi (the latter two follow the same conventions). These four main groups are used in this package. The amount of information the user provides is variable, for example the user should be able to typeset the full name with genus, old genus, author etc., but also just the normal binomial name.

Two kinds of commands are provided to reach this goal, the first is for quick-n-dirty use-only-once names. The other is a set of commands were first the species is declared, and then used throughout the document.

The quick-n-dirty commands look like:

- `\plantlike [`*how*`]{`*parameters*`}`
- `\funguslike[`*how*`]{`*parameters*`}`
- `\animallike[`*how*`]{`*parameters*`}`
- `\bactlike  [`*how*`]{`*parameters*`}`

The declaration commands look like:

- `\newplant {`*identifier*`}{`*parameters*`}`
- `\newfungus{`*identifier*`}{`*parameters*`}`
- `\newanimal{`*identifier*`}{`*parameters*`}`
- `\newbact  {`*identifier*`}{`*parameters*`}`

Declarated species can be used with

- `\plant [`*how*`]{`*identifier*`}`
- `\fungus[`*how*`]{`*identifier*`}`
- `\animal[`*how*`]{`*identifier*`}`
- `\bact  [`*how*`]{`*identifier*`}`

In these commands, *parameters* define the actual species. They can consist of genus, oldgenus, epithet, oldepithet, author, year, oldauthor, and oldyear, and follow a `key=value` syntax. Furthermore, more parameters can be defined by the user. The *identifier* is the identifier by which a species is recognized by the user. The commands to declare species are discussed below, while the commands to use them are discussed later.

The declaration commands are all piped through a single command. The quick-n-dirty commands also declare a new species, but always with the same identifier. This command is the `\n@wsp@cies{`*type*`}{`*identifier*`}{`*parameters*`}` command, where *type* is P, F, A, B for plant, fungus, animal or bacterium.

## 2.2 The implementation for the declaration part

As described above, all the declaration commands are piped through a single command (the quick-n-dirty commands don't get a *type* identifier, only a single identifier is needed):

⟨*The user interface*⟩≡

```
\newcommand{\newplant}[2]{\n@wsp@cies{P}{#1}{#2}}
\newcommand{\newfungus}[2]{\n@wsp@cies{F}{#1}{#2}}
\newcommand{\newanimal}[2]{\n@wsp@cies{A}{#1}{#2}}
\newcommand{\newbact}[2]{\n@wsp@cies{B}{#1}{#2}}

\newcommand{\plantlike}[2][]
  {\n@wsp@cies{}{Q@D}{#2}\plant[#1]{Q@D}\expandafter\Q@DCleanup\Q@DCleanList+}
\newcommand{\funguslike}[2][]
  {\n@wsp@cies{}{Q@D}{#2}\fungus[#1]{Q@D}\expandafter\Q@DCleanup\Q@DCleanList+}
\newcommand{\animallike}[2][]
  {\n@wsp@cies{}{Q@D}{#2}\animal[#1]{Q@D}\expandafter\Q@DCleanup\Q@DCleanList+}
\newcommand{\bactlike}[2][]
  {\n@wsp@cies{}{Q@D}{#2}\bact[#1]{Q@D}\expandafter\Q@DCleanup\Q@DCleanList+}
```

# 3 The internal workings

For now, we ignore the commands to use the declared species, and instead define the internal workings of the `\n@wsp@cies` command.

Since the `key=value` syntax is used, the `keyval` package is needed:

⟨*Package loading*⟩≡
```
\RequirePackage{keyval}
```

The `\n@wsp@cies` command creates a command for all the different parameters it gets, which name is of *TypeIdentifier@Parametername*, for example, if the command would be `\n@wsp@cies{A}{Hs}{genus=homo,epithet=sapiens}`, it would define the commands `\AHs@genus` and `\AHs@epiteth`.

⟨*Internal workings*⟩≡
```
⟨The n@wsp@cies command⟩
```

⟨*The n@wsp@cies command*⟩≡
```
\newcommand{\n@wsp@cies}[3]{%
```

First this function determines what the first part of all the commands should be. It stores this in a parameter called `\curr@ntid`.

⟨*Declarations*⟩≡
```
\newcommand{\curr@ntid}{}
```

⟨*The n@wsp@cies command*⟩+≡
```
\renewcommand{\curr@ntid}{#1#2@}%
```

This command also adds a counter with name *Curr@ntID*`counter`, which will be used to track whether this command has been used before (0 for no, 1 for yes) (for Q@D an exception is made).

⟨*Declarations*⟩+≡
```
\newcounter{Q@D@counter}
```

The `\n@wsp@cies` command uses the `ifthen` package.

⟨*Package loading*⟩+≡
```
\RequirePackage{ifthen}
```

⟨*The n@wsp@cies command*⟩+≡
```
\ifthenelse{\equal{#2}{Q@D}}%
  {\relax}%
  {\newcounter{\curr@ntid counter}}%
\setcounter{\curr@ntid counter}{0}%
```

Then it goes forth by processing all the parameters. According to [1], for every key an apart function should exist. For the quick-n-dirty commands a `\Q@DCleanList` command is created to which holds all the possible keys with `\relax` associated. The `\Q@DCleanup` commands sets these keys (this is to prevent old values from being used when a value is not given).

⟨*Internal workings*⟩+≡
```
⟨The key=value functions⟩
```

⟨*The key=value functions*⟩≡
```
\def\Q@DCleanup#1+{\n@wsp@cies{}{Q@D}{#1}}

\newcommand{\add@species@key}[2]{%
  \define@key{SpeciesParams}{#1}{#2}
  \expandafter\ifx\csname Q@DCleanList\endcsname\relax%
     \def\Q@DCleanList{#1=\relax}%
  \else%
     \edef\Q@DCleanList{\Q@DCleanList,#1=\relax}%
  \fi%
}
```

⟨*The n@wsp@cies command*⟩+≡
```
\setkeys{SpeciesParams}{#3}%
}
```

The handler for the genus should make sure it is written capitalized. Therefore a function is made which splits the first letter of a word. It used more often in this package.

⟨*Declarations*⟩+≡
```
\newcommand{\T@mpFirst}{}
\newcommand{\T@mpRest}{}
```

⟨*The key=value functions*⟩+≡
```
\def\SplitG@nusL@tters(#1#2){%
  \uppercase{\renewcommand{\T@mpFirst}{#1}}\lowercase{\renewcommand{\T@mpRest}{#2}}}
```

Then the genus is processed and stored in the right way.

⟨*The key=value functions*⟩+≡
```
\add@species@key{genus}{%
  \SplitG@nusL@tters(#1)%
  \expandafter\edef\csname\curr@ntid genus\endcsname{\T@mpFirst\T@mpRest}%
}
```

Of course, this also goes for the old genus.

⟨*The key=value functions*⟩+≡
```
\add@species@key{oldgenus}{%
  \SplitG@nusL@tters(#1)%
  \expandafter\edef\csname\curr@ntid oldgenus\endcsname{\T@mpFirst\T@mpRest}%
}
```

The epithet and old epithet all have to be completely lowercase.

⟨*The key=value functions*⟩+≡
```
\add@species@key{epithet}{\lowercase{\expandafter\edef\csname\curr@ntid epithet\endcsname{#1}}}
\add@species@key{oldepithet}{\lowercase{\expandafter\edef\csname\curr@ntid oldepithet\endcsname{#1}}}
```

And there are the author, old author year and the old year.

⟨*The key=value functions*⟩+≡
```
\add@species@key{author}{\expandafter\edef\csname\curr@ntid author\endcsname{#1}}
\add@species@key{year}{\expandafter\edef\csname\curr@ntid year\endcsname{#1}}
\add@species@key{oldauthor}{\expandafter\edef\csname\curr@ntid oldauthor\endcsname{#1}}
\add@species@key{oldyear}{\expandafter\edef\csname\curr@ntid oldyear\endcsname{#1}}
```

As mentioned, the user should also be able to add own taxonomical structures. Herefore a the `\newtaxon{`*name*`}` is used.

⟨*Internal workings*⟩+≡
```
\newcommand{\newtaxon}[1]{\add@species@key{#1}{\expandafter\edef\csname\curr@ntid #1\endcsname{##1}}}
```

Two special keys are for the default full style and the default abbreviation.

⟨*The key=value functions*⟩+≡
```
\define@key{SpeciesParams}{fullstyle}
  {\expandafter\def\csname\curr@ntid fullstyle\endcsname{\csname Sp@cies#1\endcsname}}
\define@key{SpeciesParams}{abbrstyle}
  {\expandafter\def\csname\curr@ntid abbrstyle\endcsname{\csname Sp@cies#1\endcsname}}
```

## 3.1 The implementation for the use part part

There's a lot to do with the actual showing of a species. This package was born from the the desire to automagically show *Genus epithet* the first time a species was used, but us *G. epithet* all subsequent times. However, sometimes an abbreviation should just be *Genus*, and someimes the full name is required. Even more, writing the complete species names with subspecies and old genus stuff etc. is better left to the computer. By default, four different modes of typesetting are provided; the *how* parameter specifies how the name should be typeset, this can be **e**xtended, which gives all available information, **l**ong, which gives genus and epithet, **a**bbreviated, which gives the first letter of the genus followed by the epithet, and **g**enus, which gives the genus only. It is also possible for the user to create typesetting schemes, and to set the default full name and abbreviation per species or globally.

Let's start with the commands used to write out the names stored in LaTeX' memory. These are accesible by the user and are already discussed:

- `\plant [`*how*`]{`*identifier*`}`
- `\fungus[`*how*`]{`*identifier*`}`
- `\animal[`*how*`]{`*identifier*`}`
- `\bact  [`*how*`]{`*identifier*`}`

These command use "style" commands to do the actual typesetting. A "style" command contains text and `\taxon{`*pre*`!`*name*`!`*post*`}` commands. Text is shown verbatim. The `\taxon` command shows the taxon *name* for the current species enclosed by *pre* and *post* if this taxon exists. So the typesetting is done by `\plant`/`Fungus`/`Animal`/`Bact`→*style command*→`\taxon`. Besides the `Taxon` command, there is a similar `FirstTaxon` command, which shows only the first letter of that taxon.

⟨*Internal workings*⟩+≡
  ⟨*The Taxon commands*⟩

Both the `\taxon` and the `\taxonfirst` command pipe through a single `\Sh@wTax@n` command, which takes its argument in the form of `+(`*pre*`!`*name*`!`*post*`)+`*how*`+`. The unique enclosures are needed to prevent interference with the *pre* and *post* from the user. *how* is either *n* for normal or *a* for abbreviated (first letter only).

⟨*The Taxon commands*⟩≡
```
\newcommand{\taxon}[1]{\Sh@wTax@n+(#1)+n+}
\newcommand{\taxonfirst}[1]{\Sh@wTax@n+(#1)+a+}

\def\Sh@wTax@n+(#1!#2!#3)+#4+{%
```

`\curr@ntid` is the ID of the species currently treated. This will be discussed later.

⟨*The Taxon commands*⟩+≡
```
  \expandafter\ifx\csname\curr@ntid#2\endcsname%
    \relax%
```

If the taxon exist, the function checks if it should display normal, and if this is the case it should display *pre*, the taxon, and *post*.

⟨*The Taxon commands*⟩+≡
```
  \else%
    \ifthenelse{\equal{#4}{n}}{%
      #1\csname\curr@ntid#2\endcsname#3%
```

Otherwise, only the first letter should be displayed. This is done by expanding the current taxon into the macro `\T@mpTax@n` which is used as argument for the lettersplitting function.

⟨*The Taxon commands*⟩+≡
```
    }{%
      \edef\T@mpTax@n{\csname\curr@ntid#2\endcsname}%
      #1\expandafter\Sh@wFirst\T@mpTax@n+#3%
    }%
  \fi%
}

\def\Sh@wFirst#1#2+{#1}
```

New style can be created with the `\newtaxastyle{`*name*`}{`*style*`}` command, where *name* is an identifier for that style, and *style* is, well, the style.

⟨ *The user interface* ⟩+≡
  ⟨ *The newtaxastyle command* ⟩

⟨ *The newtaxastyle command* ⟩≡
```
\newcommand{\newtaxastyle}[2]{\expandafter\def\csname Sp@cies#1\endcsname{#2}}
```

It is often the case that some names should be printed in italics if the rest of the text is upright, or vice versa. Herefore the command `\taxit{}` is provided, which is the same as `\em` in `latex.ltx`.

⟨ *Internal workings* ⟩+≡
```
\DeclareRobustCommand\taxitalics
        {\@nomath\em \ifdim \fontdimen\@ne\font >\z@
                       \upshape \else \itshape \fi}
\DeclareTextFontCommand{\taxit}{\taxitalics}
```

Using this, the default type can be implemented.

⟨ *Internal workings* ⟩+≡
```
\newtaxastyle{ePlant}
  {\taxit{\taxon{!genus!}\taxon{ !epithet!}}\taxon{ (!oldauthor!)}\taxon{ !author!}}
\newtaxastyle{eAnimal}
  {\taxit{\taxon{!genus!}\taxon{ (!oldgenus!)}\taxon{ !epithet!}}\taxon{ (!oldauthor!}%
   \taxon{, !oldyear!)}\taxon{!author!}\taxon{, !year!}}
\newtaxastyle{f}
  {\taxit{\taxon{!genus!}\taxon{ !epithet!}}}
\newtaxastyle{a}
  {\taxit{\taxonfirst{!genus!.}\taxon{ !epithet!}}}
\newtaxastyle{g}
  {\taxit{\taxon{!genus!}}}
```

Now the default styles can be set.

⟨ *Declarations* ⟩+≡
```
\newcommand{\Gl@balF@llStyle}{}
\newcommand{\Gl@bal@bbrStyle}{}
\newcommand{\Gl@balPE@llStyle}{}
\newcommand{\Gl@balFE@llStyle}{}
\newcommand{\Gl@balAE@llStyle}{}
\newcommand{\Gl@balBE@llStyle}{}
```

⟨ *The user interface* ⟩+≡
```
\newcommand{\defaultplante}[1]
  {\renewcommand{\Gl@balPE@llStyle}{\csname Sp@cies#1\endcsname}}
\newcommand{\defaultfunguse}[1]
  {\renewcommand{\Gl@balFE@llStyle}{\csname Sp@cies#1\endcsname}}
\newcommand{\defaultanimale}[1]
  {\renewcommand{\Gl@balAE@llStyle}{\csname Sp@cies#1\endcsname}}
\newcommand{\defaultbacte}[1]
  {\renewcommand{\Gl@balBE@llStyle}{\csname Sp@cies#1\endcsname}}
\newcommand{\defaultfull}[1]
  {\renewcommand{\Gl@balF@llStyle}{\csname Sp@cies#1\endcsname}}
\newcommand{\defaultabbr}[1]
  {\renewcommand{\Gl@bal@bbrStyle}{\csname Sp@cies#1\endcsname}}
```

⟨ *Internal workings* ⟩+≡
```
\defaultplante{ePlant}
\defaultfunguse{ePlant}
\defaultanimale{eAnimal}
\defaultbacte{eAnimal}
\defaultfull{f}
\defaultabbr{a}
```

The calling commands fisrt set `\curr@ntid` to the current ID (duh!). Furthermore, they have a decision tree: if no style is provided, then it is determined whether or not this is the first use in the document, and action is taken appropiately. This is all done using the `\sh@wsp@cies{`*type*`}{`*ID*`}{`*how*`}`.

⟨*The user interface*⟩+≡
  ⟨*The calling commands*⟩

⟨*The calling commands*⟩≡
```
\newcommand{\plant}[2][]{\sh@wsp@cies{P}{#2}{#1}}
\newcommand{\fungus}[2][]{\sh@wsp@cies{F}{#2}{#1}}
\newcommand{\animal}[2][]{\sh@wsp@cies{A}{#2}{#1}}
\newcommand{\bact}[2][]{\sh@wsp@cies{B}{#2}{#1}}
```

First, the command sets `\curr@ntid` (if it's Q@D no *type* should be set).

⟨*Internal workings*⟩+≡
  ⟨*The sh@wsp@cies command*⟩

⟨*The sh@wsp@cies command*⟩≡
```
\newcommand{\sh@wsp@cies}[3]{%
  \ifthenelse{\equal{#2}{Q@D}}%
    {\renewcommand{\curr@ntid}{#2@}}%
    {\renewcommand{\curr@ntid}{#1#2@}}%
```

Then there is checked for a provided style.

⟨*The sh@wsp@cies command*⟩+≡
```
  \ifthenelse{\equal{#3}{}}{%
    \ifnum\value{\curr@ntid counter}=0%
```

If this is not the case, it is checked whether this is the first time the species is used. Of it is, for default full style is checked and used.

⟨*The sh@wsp@cies command*⟩+≡
```
      \setcounter{\curr@ntid counter}{1}%
      \expandafter\ifx\csname\curr@ntid fullstyle\endcsname\relax%
        \csname Gl@balF@llStyle\endcsname%
      \else%
        \csname\curr@ntid fullstyle\endcsname %
      \fi%
```

Otherwise, the default abbreviation is checked and used.

⟨*The sh@wsp@cies command*⟩+≡
```
    \else%
      \expandafter\ifx\csname\curr@ntid abbrstyle\endcsname\relax%
        \csname Gl@bal@bbrStyle\endcsname%
      \else%
        \csname\curr@ntid abbrstyle\endcsname%
      \fi%
    \fi%
```

If a style is provided, use this style. If this style is "extended", select the appropiate style for the kingdom.

⟨*The sh@wsp@cies command*⟩+≡
```
  }{%
    \ifthenelse{\equal{#3}{e}}{%
      \csname Gl@bal#1E@llStyle\endcsname%
    }{%
      \csname Sp@cies#3\endcsname%
    }%
  }%
}
```

# References

[1] David Carlisle. *The keyval package*, 16 March 1999.

[2] Michel Goossens, Alexander Samarin, and Frank Mittelbach. *The LaTeX comapanion*. Addison-Wesley, 1994.