

The covington Package

Macros for Linguistics

Michael A. Covington Jürgen Spitzmüller*

Version 2.0dev, December 8, 2018

Abstract

This package, initially a collection of Michael A. Covington's private macros, provides numerous minor L^AT_EX enhancements for linguistics, including multiple diacritics on the same letter, interlinear glosses (word-by-word translations), Discourse Representation Structures, and example numbering.

Contents

1	Introduction	2
2	Stacked diacritics	2
3	Numbered examples	3
3.1	Example numbers	3
3.2	The example environment	4
3.3	The examples environment	5
3.4	The subexamples environment	5
3.5	Customizing the numbering	6
3.6	Referring to examples	6
4	Glossing sentences word-by-word	7
4.1	Gloss markup	7
4.2	Customization	8
4.3	Examples	9
5	Phrase structure rules	10
6	Feature structures	10
7	Discourse Representation Structures	11
8	Exercises	12
9	Reference Lists	12
10	Semantik markup	13
11	Big curly brackets (disjunctions)	14
12	Release history	14

*Current maintainer. Please report issues via <https://github.com/jspitz/covington>

1 Introduction

This is the documentation for version 2.0dev of covington (December 8, 2018), which is a \LaTeX package providing macros for typing some special notations common in linguistics.¹

To use covington with \LaTeX 2_ε, load the package as usual by adding the command `\usepackage{covington}` to your document preamble. The package has the following options:

force: Force the redefinition of environments that have already been defined by other packages or the class.

This applies to the **example**, **examples**, **subexamples** and **exercise** environments, which are by default not touched if they are already defined before covington is loaded. See sec. 3.2, 3.3, 3.4 and 8 for details.

keeplayout: Do not tweak the layout.

Covington sets `\raggedbottom` and redefines the value of the `\textfloatsep` length. This just follows the preferences of the original package author and is not necessary for the package’s functionality. Yet for backwards compatibility reasons, we cannot change this. Thus, we provide the option described here to opt out this presetting.

In what follows we presume that you know how to use \LaTeX and have access to \LaTeX manuals. Note that covington does not provide any special fonts or character sets. However, it can be used in combination with other style sheets that do.

If you are using covington and `uga.sty` (UGa thesis style) together, you should load `uga` before covington.

2 Stacked diacritics

\LaTeX provides a generous range of diacritics that can be placed on or below any letter, such as:

˘ ˘ ˘ ˘ ˘ ˘ ˘ ˘ ˘ ˘ ˘ ˘ ˘ ˘

which are typed, respectively, as:

<code>\' {x}</code> <code>\' {x}</code> <code>\^ {x}</code> <code>\" {x}</code> <code>\~ {x}</code> <code>\= {x}</code> <code>\H {x}</code> <code>\t {xx}</code> <code>\c {x}</code> <code>\d {x}</code> <code>\b {x}</code>
--

Out of the box, however, \LaTeX doesn’t give you a convenient way to put *two* diacritical marks on the same letter. To fill this gap, covington provides the following macros:

¹The package has a long history. It started off as a collection of private macros back in the \LaTeX 2.09 days and was initially released as `covingt.n.sty` (following the old 8.3 FAT file name limit). In $\text{em}\text{\LaTeX}$ under MS-DOS, the file was distributed as `covingto.sty`. Eventually, it has been renamed to covington and adapted to \LaTeX 2_ε.

`\twodias{<upper diac.>}{<lower diac.>}{<char>}`
to combine any two diacritics, e. g., `\twodias{\~}{\={}}{a} = \tilde{a}`

`\acm{...}` for acute over macron, e. g., `\acm{a} = \acute{a}`
`\grm{...}` for grave over macron, e. g., `\grm{a} = \grave{a}`
`\cim{...}` for circumflex over macron, e. g., `\cim{a} = \hat{a}`

The first of these is the general case² and the latter three are special cases that are often used in Greek transcription. Now you can type *Koiné* with both accents in place.

The vertical distance between the two diacritics can be adjusted via the macro `\SetDiaOffset{<length>}` which lets you increase or decrease the vertical space that is currently in effect. If you'd use `\SetDiaOffset{-0.25ex}`, the above examples would come out as

`\twodias{<upper diac.>}{<lower diac.>}{<char>}`
to combine any two diacritics, e. g., `\twodias{\~}{\={}}{a} = \tilde{a}`

`\acm{...}` for acute over macron, e. g., `\acm{a} = \acute{a}`
`\grm{...}` for grave over macron, e. g., `\grm{a} = \grave{a}`
`\cim{...}` for circumflex over macron, e. g., `\cim{a} = \hat{a}`

with a slightly better matching distance for the font used here.

Note that not all accent macros work in the `tabbing` environment. Use the `Tabbing` package or refer to [1] for alternative solutions.

3 Numbered examples

Linguistic papers often include numbered examples. With `covington`, generating those is straightforward. In this section, we describe how you can typeset a self-stepping example number (see section 3.1), a single numbered example (sec. 3.2), a consecutive range of numbered examples (sec. 3.3), and alpha-numerically labeled sub-examples (sec. 3.4). All numbered examples can be referred to in the text via `\label` and `\ref` as usual (see sec. 3.6 for details).

3.1 Example numbers

The macro `\exampleno` generates a new example number, stepped by 1. It can be used anywhere you want the number to appear. For example, to display a sentence with a number at the extreme right, do this:

```
\begin{flushleft}
This is a sentence. \hfill (\exampleno)
\end{flushleft}
```

Here's what you get:

²Alternatively, there's also the old syntax `\twoacc[<upper diac.>]{<char with lower diacr.>}`, e. g. `\twoacc{\~}{\={}}{a}` to the same effect, which is however discouraged due to its rather odd form.

This is a sentence. (1)

The example counter is actually the same as L^AT_EX's equation counter, so that if you use equations and numbered examples in the same paper, you get a single continuous series of numbers. If you want to output the number without stepping it, use `\theequation`.

Normally, however, you do not need to manually place `\example` yourself, as in the example above. For the common case where example numbers in parentheses are placed left to the example, covington provides more convenient solutions. These are described in turn.

3.2 The example environment

The `example` environment (alias `covexample`) displays a single example with a generated example number to the left of it. If you type

```
\begin{example}  
This is a sentence.  
\end{example}
```

or

```
\begin{covexample}  
This is a sentence.  
\end{covexample}
```

you get:

(2) This is a sentence.

The example can be of any length; it can consist of many lines (separated by `\\`), or even whole paragraphs.

If you need more space between the example number and the text, you can increase it by means of the length `\examplenumbersep` (which is preset to 0pt). Doing `\setlength\examplenumbersep{1em}`, for instance, will increase the space by 1 em (negative values will decrease the space accordingly).

Note that, as of version 1.1, covington checks if there is already an `example` environment defined (e. g., by the class). If so, covington does not define its own one. However, there is always the alias environment `covexample` which can be used in order to produce covington's example. If you use the package option `force`, covington will override existing `example` environments. In any case, the package will issue a warning if `example` is already defined (this is the case, for instance, if you use covington with the beamer class).

One way to number sub-examples is to use `itemize` or `enumerate` within an example, like this:

```
\begin{example}  
  \begin{itemize}  
    \item[(a)] This is the first sentence.  
    \item[(b)] This is the second sentence.  
  \end{itemize}  
\end{example}
```

This prints as:

- (3) (a) This is the first sentence.
- (b) This is the second sentence.

However, the **examples** and **subexamples** environments, described in turn, are usually more convenient for this task.

3.3 The examples environment

To display a series of examples together, each with its own example number, use **examples** (or **covexamples**) instead of **example** or **covexample**. The only difference is that there can be more than one example, and each of them has to be introduced by `\item`, like this:

```
\begin{examples}
\item This is the first sentence.
\item This is the second sentence.
\end{examples}
```

or, respectively:

```
\begin{covexamples}
\item This is the first sentence.
\item This is the second sentence.
\end{covexamples}
```

This prints as:

- (4) This is the first sentence.
- (5) This is the second sentence.

As for **example**, covington checks if there is already an **examples** environment defined, and if this is the case, covington does not define its own one. The alias environment **covexamples** is always available as a fallback. If you use the package option **force**, covington will override existing **examples** environments. The package will issue a warning if **examples** is already defined (this is the case, for instance, if you use covington with the beamer class), telling you how it has dealt with the situation.

3.4 The subexamples environment

Sometimes a set of (paradigmatic) sub-examples gets only one main example number with alphabetic sub-numbering, as in (6 a). To achieve this most conveniently, covington provides the **subexamples** (or **covsubexamples**) environment. The difference to **examples/covexamples** is the numbering:

```
\begin{subexamples}
\item This is the first sentence.
\item This is the second sentence.
\end{subexamples}
```

or, respectively:

```
\begin{covsubexamples}  
\item This is the first sentence.  
\item This is the second sentence.  
\end{covsubexamples}
```

prints as:

- (6) (a) This is the first sentence.
(b) This is the second sentence.

Again, covington checks if there is already an **subexamples** environment defined, and if this is the case, covington does not define its own one. The alias environment **covsubexamples** is always available as a fallback. If you use the package option **force**, covington will override existing **subexamples** environments. The package will issue a warning if **subexamples** is already defined.

3.5 Customizing the numbering

You can change the display of the example number by redefining (via `\renewcommand*`) the macro `\covexnumber` which has the following default definition:

```
\newcommand*\covexnumber[1]{(#1)}
```

In the same vein, you can customize the display of the subexample letter by redefining (also via `\renewcommand*`) the macro `\covsubexnumber` which has the following default definition:

```
\newcommand*\covsubexnumber[1]{(#1)}
```

The distance between example number and subnumber (letter) can be changed via the length `\examplenumbersp` (which is preset to 0pt). The distance between example subnumber and text can be changed via the length `\subexamplenumbersp` (preset to 0pt as well). In both cases, a positive value will increase, a negative value will decrease the respective distance. Doing

```
\setlength{\examplenumbersp}{-0.5em}  
\setlength{\subexamplenumbersp}{0.5em}
```

for instance, will come out like this:

- (7) (a) This is the first sentence.
(b) This is the second sentence.

3.6 Referring to examples

References to examples and sub-examples can be made the usual way via the `\ref` command (which refers to a `\label` that is placed in the respective example paragraph). The references do not have parentheses by default, i. e., a reference to the example in

section 3.2 would be printed as 2, a reference to the sub-example in section 3.4 as 6 a. For convenience, though, covington provides a command `\pxref` that also prints the parentheses, as in (2) and (6 a). It is defined as follows and can be redefined if needed:

```
\providecommand*\pxref[1]{(\ref{#1})}
```

4 Glossing sentences word-by-word

4.1 Gloss markup

To gloss a sentence is to annotate it word-by-word. Most commonly, a sentence in a foreign language is followed by a word-for-word translation (with the words lined up vertically) and then a smooth translation (not lined up), like this:³

Dit is een Nederlands voorbeeld.
This is a Dutch example.
‘This is an example in Dutch.’

That particular example would be typed as:

```
\gll Dit is een Nederlands voorbeeld.  

      This is a Dutch example.  

\glt ‘This is an example in Dutch.’  

\glend
```

Notice that the words do not have to be typed lining up; instead, \TeX counts them. If the words in the two languages do not correspond one-to-one, you can use curly brackets to group words. For example, to print

Dit is een voorbeeldje in het Nederlands.
This is a little example in Dutch.
‘This is a little example in Dutch.’

you would type:

```
\gll Dit is een voorbeeldje in het Nederlands.  

      This is a {little example} in {} Dutch.  

\glt ‘This is a little example in Dutch.’  

\glend
```

Note that covington locally activates the end of line in glosses in order to identify the different lines of the gloss (via category code change). This does not work inside macros (e. g., if the gloss is in a footnote). To work around this, a special version of the `\gll` macro is provided that does without the character activation: `\xgll`. This can also be used in macro arguments; however, the end of each gloss line needs to be explicitly specified by the `\xgle` macro in this case. If you want to put the above gloss in a footnote, thus, you would type:

³The macros for handling glosses are adapted with permission from `gloss.tex`, by Marcel R. van der Goot.

```
\xgll Dit is een voorbeeldje      in het Nederlands.\xgle
      This is a {little example} in {} Dutch.\xgle
\glt 'This is a little example in Dutch.'
\glend
```

Sometimes, an additional fourth line is needed (for instance to gloss cited forms, morphology, or an additional translation). To support this as well, covington provides `\glll` and `\xglll`, each allowing for, and actually requiring, an additional line (see below for an example).

All together, covington provides eight macros for dealing with glosses:

- `\gll` introduces two lines of words vertically aligned, and activates an environment very similar to `flushleft`. The two lines are separated by a normal line break (carriage return).
- `\glll` is like `\gll` except that it introduces *three* lines of lined-up words.
- `\xgll` is similar to `\gll` except that it does not make the line ending active. It thus works inside macros such as footnotes but requires explicit gloss line termination via `\xgle`.
- `\xglll` is similar to `\glll` except that it does not make the line ending active. It thus works inside macros such as footnotes but requires explicit gloss line termination via `\xgle`.
- `\xgle` is a gloss line ending marker to be used with `\xgll` and `\xglll`.
- `\glt` ends the set of lined-up lines and introduces a line (or more) of translation.
- `\gln` is like `\glt` but does not start a new line (useful when no translation follows but you want to put a number on the right).
- `\glend` ends the special `flushleft`-like environment.

Notice that every glossed sentence begins with either `\gll`, `\xgll`, `\glll` or `\xglll`, then contains either `\glt` or `\gln`, and ends with `\glend`. Layout is critical in the part preceding `\glt` or `\gln`, and fairly free afterward.

4.2 Customization

The font settings of each gloss line can be customized by redefining these macros:

```
\newcommand*\glosslineone{\normalfont\itshape}% font settings 1st gloss line
\newcommand*\glosslinetwo{\normalfont\upshape}% font settings 2nd gloss line
\newcommand*\glosslinethree{\normalfont\upshape}% font settings 3rd gloss line
```

The markup of the translation line has to be done manually.

4.3 Examples

Here are several examples. First, a sentence with three lines aligned, instead of just two:

Hoc est aliud exemplum.
 N.SG.NOM 3SG N.SG.NOM N.SG.NOM
 This is another example.
 ‘This is another example.’

This is typed as:

```
\glll Hoc est aliud exemplum.
      \textsc{n.sg.nom} \textsc{3sg} \textsc{n.sg.nom} \textsc{n.sg.nom}
      This is another example.
\glt  ‘This is another example.’
\glend
```

or, using the font customization possibilities described in sec. 4.2:

```
\begingroup
\renewcommand*\glosslinetwo{\normalfont\scshape}
\glll Hoc est aliud exemplum.
      n.sg.nom 3sg n.sg.nom n.sg.nom
      This is another example.
\glt  ‘This is another example.’
\glend
\endgroup
```

(with `\begingroup... \endgroup` to keep the changes local).

Next, an example with a gloss but no translation, with an example number at the right:

Hoc habet numerum. (8)
 This has number

That one was typed as:

```
\gll Hoc habet numerum.
      This has number
\gln \hfill (\exampleno)
\glend
```

Finally we’ll put a glossed sentence inside the example environment, which is a very common way of using it:

(9) *Hoc habet numerum praepositum.*
 This has number preposed
 ‘This one has a number in front of it.’

This last example was, of course, typed as:

```

\begin{example}
\gll  Hoc habet numerum praepositum.
      This has number preposed
\glt  'This one has a number in front of it.'
\glend
\end{example}

```

5 Phrase structure rules

To print phrase structure rules such as $S \rightarrow NP VP$ you can use covington's macro `\psr{<constituent>}{<sub-constituents>}` (for the given example, `\psr{S}{NP-VP}`).

6 Feature structures

To print a feature structure such as

$$\begin{bmatrix} \textit{case} : \textit{nom} \\ \textit{person} : P \end{bmatrix}$$

you can type:

```
\fs{case:nom \\\ person:P}
```

The feature structure can appear anywhere — in continuous text, in a displayed environment such as `flushleft`, or inside a phrase-structure rule, or even inside another feature structure.

To put a category label at the top of the feature structure, like this,

$$\begin{array}{c} N \\ \begin{bmatrix} \textit{case} : \textit{nom} \\ \textit{person} : P \end{bmatrix} \end{array}$$

here's what you type:

```
\lfs{N}{case:nom \\\ person:P}
```

And here is an example of a ps-rule made of labeled feature structures:

$$\begin{array}{ccc} S & \rightarrow & NP \quad VP \\ \begin{bmatrix} \textit{tense} : T \end{bmatrix} & & \begin{bmatrix} \textit{case} : \textit{nom} \\ \textit{number} : N \end{bmatrix} \quad \begin{bmatrix} \textit{tense} : T \\ \textit{number} : N \end{bmatrix} \end{array}$$

which was obviously coded as:

```

\psr{\lfs{S}{tense:T}}
    {\lfs{NP}{case:nom \\\ number:N}
     \lfs{VP}{tense:T \\\ number:N} }

```

7 Discourse Representation Structures

Several macros in covington facilitate display of discourse Representation Structures (DRSes) in the box notation introduced by Hans Kamp. The simplest one is `\drs`, which takes two arguments: a list of discourse variables joined by \sim , and a list of DRS conditions separated by $\backslash\backslash$. Nesting is permitted. Note that the `\drs` macro itself does not give you a displayed environment; you must use `flushleft` or the like to display the DRS. Here are some examples:

```
\begin{flushleft}
\drs{X}{donkey(X)\green(X)}
\end{flushleft}
```

X
$donkey(X)$
$green(X)$

```
\begin{flushleft}
\drs{X}
{named(X, 'Pedro') \backslash
\drs{Y}{donkey(Y)\owns(X,Y)}\sim
{\large $\Rightarrow$}\sim
\drs{\sim}{feeds(X,Y)}}
\end{flushleft}
```

X				
$named(X, 'Pedro')$				
<table border="1"> <tr><td>Y</td></tr> <tr><td>$donkey(Y)$</td></tr> <tr><td>$owns(X,Y)$</td></tr> </table> \Rightarrow <table border="1"> <tr><td>$feeds(X,Y)$</td></tr> </table>	Y	$donkey(Y)$	$owns(X,Y)$	$feeds(X,Y)$
Y				
$donkey(Y)$				
$owns(X,Y)$				
$feeds(X,Y)$				

To display a sentence above the DRS, use `\sdrs`, as in:

```
\begin{flushleft}
\sdrs{A donkey is green.}{X}{donkey(X)\green(X)}
\end{flushleft}
```

which prints as:

A donkey is green.

X
$donkey(X)$
$green(X)$

Some DRS connectives are also provided (normally for forming DRSes that are to be nested within other DRSes). The macro `\negdrs` forms a DRS preceded by a negation symbol:

```
\negdrs{X}{donkey(X)\green(X)}
```

$$\neg \begin{array}{|l} X \\ \hline donkey(X) \\ green(X) \end{array}$$

Finally, `\ifdrs` forms a pair of DRSes joined by a big arrow, like this:

$$\text{\ifdrs}\{X\}\{donkey(X)\backslash\backslash hungry(X)\} \\ \{\sim\}\{feeds(Pedro,X)\}$$

$$\begin{array}{|l} X \\ \hline donkey(X) \\ hungry(X) \end{array} \Rightarrow \begin{array}{|l} \\ \hline feeds(Pedro,X) \end{array}$$

If you have an “if”-structure appearing among ordinary predicates inside a DRS, you may prefer to use `\alifdrs`, which is just like `\ifdrs` but shifted slightly to the left for better alignment.

8 Exercises

The `exercise` environment (alias `covexercise`) generates an exercise numbered according to chapter, section, and subsection (suitable for use in a large book; in this example, the subsection number is going to come out as o). Here is an example:

Exercise 8.o.1 (Project) *Prove that the above assertion is true.*

This was coded as

```
\begin{exercise}[Project]
Prove that the above assertion is true.
\end{exercise}
```

The argument ([Project] in the example) is optional.

Note that, as of version 1.1, `covington` checks if there is already an `exercise` environment defined (e. g., by the class). If so, `covington` does not define its own one. However, there is always the alias environment `covexercise` which can be used in order to produce `covington`’s exercise. If you use the package option `force`, `covington` will override existing `exercise` environments. In any case, the package will issue a warning if `exercise` is already defined.

9 Reference Lists

To type a simple LSA-style hanging-indented reference list, you can use the `reflist` environment. (*Note: `reflist` is not integrated with BibTeX in any way.*⁴) For example,

⁴For BibTeX, there are several options: the LSA style, as used in the journal *Language*, can be obtained by means of the style files `lsalike.bst` (<http://www.icsi.berkeley.edu/ftp/pub/speech/jurafsky/lsalike>).

```

\begin{reflist}
Barton, G. Edward; Berwick, Robert C.; and Ristad, Eric Sven. 1987.
Computational complexity and natural language. Cambridge,
Massachusetts: MIT Press.

Chomsky, Noam. 1965. Aspects of the theory of syntax. Cambridge,
Massachusetts: MIT Press.

Covington, Michael. 1993. Natural language processing for Prolog
programmers. Englewood Cliffs, New Jersey: Prentice-Hall.
\end{reflist}

```

prints as:

Barton, G. Edward; Berwick, Robert C.; and Ristad, Eric Sven. 1987. Computational complexity and natural language. Cambridge, Massachusetts: MIT Press.

Chomsky, Noam. 1965. Aspects of the theory of syntax. Cambridge, Massachusetts: MIT Press.

Covington, Michael A. 1993. Natural language processing for Prolog programmers. Englewood Cliffs, New Jersey: Prentice-Hall.

By default, the references have a hanging indentation of 3 em. This can be globally changed by altering the length `\reflistindent`. Doing `\setlength\reflistindent{1.5em}`, for instance, will shorten the indentation by half. Likewise, the length `\reflistitemsep` (6 pt by default) and `\reflistparsep` (ca. 4 pt by default) can be adjusted to alter the vertical separation (`\itemsep` and `\parsep`, for that matter) of reference entries.

Notice that within the reference list, “French spacing” is in effect — that is, spaces after periods are no wider than normal spaces. Thus you do not have to do anything special to avoid excessive space after people’s initials.

10 Semantik markup

The macro `\sentence` displays an italicized sentence (it is a combination of `flushleft` and `itshape`). If you type

```
\sentence{This is a sentence.}
```

you get:

This is a sentence.

The font shape can be modified by redefining the following macro:

`\sentence` or `language.bst` (<http://ron.artstein.org/resources/language.bst>); the latter uses `natbib`. The so-called *Unified Style Sheet for Linguistics*, as proposed by the CELXJ (*Committee of Editors of Linguistics Journals*), which slightly differs from the LSA style, is followed by the style file `unified.bst` (available at <http://celxj.org/downloads/unified.bst>). A biblatex style file for the unified style is available at <https://github.com/semprag/biblatex-sp-unified> or on CTAN as part of the `univie-ling` bundle.

```
\newcommand*\sentencefs{\itshape}
```

The following macros provide further markup options common in linguistics:

- `\lexp{word}` is used to mark word forms (italic by default, as in *word*)
- `\lcon{concept}` is used to mark concepts (small caps by default, as in CONCEPT)
- `\lmean{meaning}` is used to mark meaning (single quotes by default, as in ‘meaning’)

Note that for `\lmean`, covington checks at document begin whether the `csquotes` package is loaded. If so, it uses its language-sensitive `\enquote*` macro for quoting. If not, a fallback quotation (using English single quotation marks) is used. The usage of `csquotes` is highly recommended!

Here are the definitions of the macros. They can be redefined via `\renewcommand`:

```
\providecommand*\expression[1]{\textit{#1}}
\providecommand*\concept[1]{\textsc{#1}}
\providecommand*\lmean[1]{\covenquote{#1}}
```

11 Big curly brackets (disjunctions)

Last of all, the two-argument macro `\either` expresses alternatives within a sentence or PS-rule:

the `\either{big}{large}` dog = the $\left\{ \begin{array}{c} \text{big} \\ \text{large} \end{array} \right\}$ dog

$\text{psr}\{A\}\{B\sim\text{either}\{C\}\{D\}\sim E\} = A \rightarrow B \left\{ \begin{array}{c} C \\ D \end{array} \right\} E$

That’s all there is for now. Suggestions for improving covington are welcome, and bug reports are actively solicited (via <https://github.com/jspitz/covington>). Please note, however, that this is free software, and the authors make no commitment to do any further work on it.

12 Release history

2.0 (forthcoming)

- Add possibility to customize `\sentence` font setting. See sec. 10.
- Add `\lexp`, `\lcon` and `\lmean` markup macros. See sec. 10.

1.8 (2018 December 7)

- Fix font markup of second gloss line (do not force rm).
- Add possibility to customize gloss line font setting. See sec. 3.5.
- Add possibility to customize example number display. See sec. 3.5.

1.7 (2018 September 8)

- Fix alignment in **subexamples**.
- Improve manual.

1.6 (2018 September 7)

- Introduce new environment **subexamples** (see sec. 3.4).
- Introduce new command **\pxref** (see sec. 3.6).

1.5 (2018 August 24)

- Introduce new option **keeplayout** which allows to opt-out the layout presets covington does (**\raggedbottom**, **\textfloatsep**).

1.4 (2017 May 23)

- Introduce a new macro **\twodias** that supersedes the rather odd **\twoacc** (which is kept for backwards compatibility). See sec. 2 for details.
- Introduce macro **\SetDiaOffset** for more convenient setting of vertical distance in stacked diacritics. See sec. 2 for details.
- \TeX 2.09 is no longer officially supported (it might continue to work, but is not tested).

1.3 (2017 April 5)

- Gloss variants **\xgll** and **\xglll** that work inside macros (such as footnotes) but require explicit gloss line end markers (**\xgle**). See sec. 4 for details.
- New lengths **\reflistindent**, **\reflistparsep** and **\reflistitemsep** to globally adjust the indentation or vertical space, respectively, of rellist items. See sec. 9 for details.

1.2 (2016 August 26)

- New length **\examplenumbersep** to adjust (increase) the horizontal space between example number and example text. See sec. 3.2 for details.
- Add some more info about bibliography generation.

1.1a (2016 July 7)

- Fix encoding problem in documentation and some typos. No change in functionality.

1.1 (2016 July 6)

- The package now uses NFSS font commands if available (fallback for \LaTeX 2.09 is still provided).
- Work around clash with classes/packages that define their own **example** and **examples** environments (most notably the beamer class) as well as **exercise** environments. The covington package no longer blindly attempts to define these environments. By default, it does not define them if they are already defined (covington's own environments, however, are still available via aliases). By means of a new package option, a redefinition can also be forced. See sec. 3.2 and 3.3 for details.
- New length `\twoaccsep` allows for the adjustment of the distance between stacked accents (see sec. 2).
- Update manual.
- New maintainer: J. Spitzmüller.
- License has been changed to LPPL (in agreement with M. Covington)
- Introduce version numbers. Arbitrarily, we start with 1.1.

2014 May 16

- Patches by Robin Fairbairns:
 - Setting of `\textfloatsep` uses `\setlength` rather than `\renewcommand`
 - Style file converted to un*x line endings

2001 March 27

- It is no longer necessary to type `\it` to get proper italic type in feature structures.
- Instructions have been rewritten with \LaTeX 2_ε users in mind.

Older versions

- Multiple accents on a single letter (e. g., \acute{a}) are supported.
- This package is now called covington (with the o) and is compatible with \LaTeX 2_ε and NFSS as well as \LaTeX 2.09.
- The vertical placement of labeled feature structures has been changed so that the category labels line up regardless of the size of the structures.

References

- [1] Pakin, Scott. The Comprehensive \LaTeX Symbol List. 30 November 2015. <http://www.ctan.org/pkg/comprehensive>.