Authors and Other Names (`names.dtx`)

Charles Duan

May 2, 2023

Hereinafter offers comprehensive support for lists of names in references, such as authors and editors. Among other things, it provides for both personal and institutional authors, optional list truncation with "et al.," and formation of last-name lists for short citations.

To enter a list of names for a reference, include the appropriate parameter (e.g., `author` or `instauth`) once for each name in the list. The names will automatically be compiled into a formatted list. Every name entered will be listed, although a warning will be issued unless the `noetal` parameter is included. To truncate a list, enter only the initial name with "et al." attached to the end. For example:

```
\defbook{nimmer}{
    author=Melville B. Nimmer, author=David Nimmer,
...
}
\defjrnart{als}{
   noetal, author=John R. Allison, author=Mark A.
Lemley,
   author=David L. Schwartz, ...
}
\defjrnart{als}{author=John R. Allison et al., ...}
```

The remainder of this section provides details on how *Hereinafter* parses and uses name lists, and how to fine-tune those processes.

1 Data Model

A name list is an ordered sequence of one or more names, each of which is either a personal name and an institutional name.[1] An institutional name is simple text, although it is abbreviated in citations. A personal name, by contrast, consists of five parts:

[1]To avoid questions of what constitutes personhood (e.g., AI-authored works), a "personal name" is one that has a family-name component and would not be abbreviated in citations.

- The given name, which includes the middle initials or names. This is the only mandatory part.
- The family name. If only one name is given (e.g., Aristotle), it is treated as a given name, and the short-form name instead uses the given name.
- Suffix to the family name, such as Jr. or III. It should not be included in the last-name-only forms.
- The institution.
- An "et al." indicator for abbreviating the author list.

Separating these parts is necessary, for example, to enable short citations using just the author's last name.

Institutional affiliations attached to human names are distinct from institutional names included in a name list. Consider, for example, a government report written by person *A* at the Federal Trade Commission and *B* at the Department of Justice. If the report expresses the views of the agencies, then the agencies could be included as authors:

> *A* & *B*, Fed. Trade Comm'n & Dep't of Justice

By contrast, if the report represents the views of the individuals in their official capacities but not necessarily the views of the agencies, then only the individuals should be authors, with their affiliations attached to their names:

> *A*, Fed. Trade Comm'n & *B*, Dep't of Justice

Providing these options gives writers more flexibility to cite mixed human and institutional authors to convey accurate and useful information.

Sets a name for this package.

```
\def\hi@names@pkgname{hinames}
```

2 Inputs

For each type of name list, there will be two reference definition parameters, one for human names and one for institutions:

	Personal	**Institutional**
Authors	`author`	`instauth`
Editors	`editor`	`insted`
Letter recipients	`to`	`instto`

Institutional authors are given simply as an unabbreviated name; the package handles abbreviation. If an "et al." is desired after an institutional author, that may be appended to the name. It is treated as if it were part of the institution's name and not processed further.

Because human authors have five component parts that need to be analyzed individually, they must be given in a specific format. An institutional affiliation is always any part of a name following a comma. To separate the given name, family name, and suffix, the most explicit way is to surround the family name and name suffix with curly braces:

> ⟨*given name*⟩ [␣{⟨*family name*⟩ [␣{⟨*suffix*⟩}] }] [, ⟨*institution*⟩] [et al.]

For example:

```
Oliver Wendell {Holmes {Jr.}}, Massachusetts Supreme
Judicial Court
```

would correctly identify the well-known jurist with his one of his judicial positions.[2]

Because including the braces around family names is cumbersome, this package can infer name parts with a few heuristics, such that no braces are required.

\hi@namesplit

`\hi@namesplit{`⟨*name*⟩`}{`⟨*callback*⟩`}`: Splits a name into component parts. The callback should accept five arguments for the name parts.

```
\def\hi@namesplit#1#2{%
    \find@start{parts:}{#1}{\@unbrace{\hi@namesplit@parts{#2}}}{%
        \find@end{ et al.}{#1}{%
            \hi@namesplit@inst{\hi@etal}%
        }{\hi@namesplit@inst{}{#1}}%
        {#2}% Given as an argument no matter what
    }%
}
\make@find@start{parts:}
```

A special input format for hitotoa. If the input starts with parts, then it is expected that four parameters follow: given name, family name, institution, and et-al. The family name is split for a suffix, and the callback is run. #1 is the callback, #2 the given name, #3 the family name (with suffix), #4 the institution, #5 the et al (anything not blank is turned into `\hi@etal`).

```
\def\hi@namesplit@parts#1#2#3#4#5{%
    \ifblank{#5}{%
        \hi@namesplit@desuffix{#3}{#1{#2}}{#4}{}%
    }{%
        \hi@namesplit@desuffix{#3}{#1{#2}}{#4}{\hi@etal}%
    }%
}
```

Removes the institution from the name, then calls `\hi@namesplit@personal`. #1 is the et al., #2 the remaining name, #3 the callback.

```
\def\hi@namesplit@inst#1#2#3{%
    \find@in{, }{#2}{\hi@namesplit@personal}{%
        \hi@namesplit@personal{#2}{}%
    }{#1}{#3}%
}
\make@find@end{ et al.}
\make@find@end{ and others}
\make@find@in{, }
```

Splits a personal name by looking for a group identifying the family name. #1 is the personal name, #2 the institution, #3 the et-al, #4 the callback.

```
\def\hi@namesplit@personal#1#2#3#4{%
    \hi@namesplit@groupend{#1}{\hi@namesplit@group}{%
```

[2] Some style guides prefer a comma before the name suffix. *Hereinafter* does not explicitly support this, but with a preceding comma the name suffix would be treated as an institutional affiliation, which produces desirable results.

```
        \hi@namesplit@nogroup{#1}%
    }%
    {#2}{#3}{#4}% Passed no matter what
}
```

A group was found at the end of the personal name. #1 is the pre-group (given name), #2 the group (family name), #3 the institution, #4 the et-al, #5 the callback.

```
\def\hi@namesplit@group#1#2#3#4#5{%
    \hi@namesplit@desuffix{#2}{#5{#1}}{#3}{#4}%
}
```

2(a) Family names. When no braces delimit the family name, the package first looks for nobiliary particles that indicate a family name: von, van, de, and so on. If that succeeds, it removes suffixes from the family name, thereby separating all the components. If there is no nobiliary particle, then the package first removes suffixes from the whole name, and takes whatever is the last word in the remaining name as the family name.

A name has no group defining the family name. This macro looks for common family name prefixes. #1 is the personal name, #2 the institution, #3 the et-al, #4 the callback.

```
\def\hi@namesplit@nogroup#1#2#3#4{%
    \find@try\find@in{%
        { von }{\hi@namesplit@prefix{von }}%
        { Von }{\hi@namesplit@prefix{Von }}%
        { van }{\hi@namesplit@prefix{van }}%
        { Van }{\hi@namesplit@prefix{Van }}%
        { de }{\hi@namesplit@prefix{de }}%
        { De }{\hi@namesplit@prefix{De }}%
        { du }{\hi@namesplit@prefix{du }}%
        { Du }{\hi@namesplit@prefix{Du }}%
    }{#1}{\hi@namesplit@byspace{#1}}%
    {#2}{#3}{#4}% These are arguments no matter what
}
\make@find@in{ Von }
\make@find@in{ Van }
\make@find@in{ De }
\make@find@in{ Du }
\make@find@in{ von }
\make@find@in{ van }
\make@find@in{ de }
\make@find@in{ du }
```

Having found a nobiliary particle, reassemble the family name, remove suffixes, and run the callback. #1 is the family name prefix, #2 the given name, #3 the rest of the family name, #4 the institution, #5 the et-al, #6 the callback.

```
\def\hi@namesplit@prefix#1#2#3#4#5#6{%
    \hi@namesplit@desuffix{#1#3}{#6{#2}}{#4}{#5}%
}
```

With no braces or known family name prefix, we default to choosing the last non-suffix word as the last name. #1 is the name, #2 the institution, #3 the et-al, #4 the callback.

```
\def\hi@namesplit@byspace#1#2#3#4{%
    \hi@namesplit@desuffix{#1}\hi@namesplit@byspace@{#2}{#3}{#4}%
}
```

#1 is the desuffixed name, #2 the suffixes, #3 the institution, #4 the et-al, #5 the callback.

```
\def\hi@namesplit@byspace@#1#2#3#4#5{%
    \find@last{ }{#1}{\hi@namesplit@byspace@@{#2}{#3}{#4}{#5}}{%
        % With no spaces, the desuffixed name is only a given name.
        #5{#1}{}{#2}{#3}{#4}%
    }
}
\make@find@in{ }
```

#1 is the suffixes, #2 the institution, #3 the et-al, #4 the callback, #5 the given name, #6 the family name.

```
\def\hi@namesplit@byspace@@#1#2#3#4#5#6{%
    #4{#5}{#6}{#1}{#2}{#3}%
}
```

2(b) Name suffixes. When no braces delimit the suffix, the package looks for common suffixes (Jr., Sr., and roman numerals through X). The first such suffix found is treated as a name suffix. As a result, the example above could have been parsed correctly without braces:

```
Oliver Wendell Holmes Jr., Massachusetts Supreme
Judicial Court
```

since everything following the comma would be used as the institution, "Jr." would be identified as a suffix, "Holmes" being the last remaining word would be the family name, and the remaining text "Oliver Wendell" would be the given name.

`\hi@namesplit@desuffix` This macro is used to remove suffixes both from family names with nobiliary particles, and from complete names. Given a string, removes suffixes from the string to the extent they are present, and runs a callback on the parts. If no suffix is found, the empty string is returned as the suffix. #1 is the string, #2 the callback that takes two arguments (non-suffix, suffix).

```
\def\hi@namesplit@desuffix#1#2{%
    \hi@namesplit@groupend{#1}{#2}{%
        \find@try\find@end{%
            { Jr.}{\hi@namesplit@desuffix@{Jr.}}%
            { Sr.}{\hi@namesplit@desuffix@{Sr.}}%
            { I}{\hi@namesplit@desuffix@{I}}%
            { II}{\hi@namesplit@desuffix@{II}}%
            { III}{\hi@namesplit@desuffix@{III}}%
            { IV}{\hi@namesplit@desuffix@{IV}}%
            { V}{\hi@namesplit@desuffix@{V}}%
            { VI}{\hi@namesplit@desuffix@{VI}}%
            { VII}{\hi@namesplit@desuffix@{VII}}%
            { VIII}{\hi@namesplit@desuffix@{VIII}}%
            { IX}{\hi@namesplit@desuffix@{IX}}%
            { X}{\hi@namesplit@desuffix@{X}}%
        }{#1}{\hi@namesplit@desuffix@{}{#1}}{#2}%
    }%
}
```

#1 is the suffix, #2 the pre-suffix string, #3 the callback.

```
\def\hi@namesplit@desuffix@#1#2#3{#3{#2}{#1}}
\make@find@end{ Jr.}
\make@find@end{ Sr.}
\make@find@end{ I}
\make@find@end{ II}
\make@find@end{ III}
\make@find@end{ IV}
\make@find@end{ V}
\make@find@end{ VI}
\make@find@end{ VII}
\make@find@end{ VIII}
\make@find@end{ IX}
\make@find@end{ X}
```

`\hi@namesplit@groupend` This is a utility macro for identifying a group at the end of a string, for example that delimits a last name. The group must be preceded by a space. That is, `abc {def}` would split to `abc` and `def`, but `abc{def}` would find nothing. #1 is the text to split; #2 the callback on successful find; #3 the callback otherwise.

```
\def\hi@namesplit@groupend#1#2#3{%
    \find@group@end{#1}{%
        \hi@namesplit@groupend@{#2}{#3}%
    }{#3}%
}
```

#1 and #2 are callbacks; #3 the pre-group text, #4 the group text. Implementation: The group text is left at the end of the stack. On success, callback #1 is run with the space-chopped text as the first argument (due to `\find@end`) and the group text as the second argument. On failure, `\@firstoftwo` gobbles the group text at the end of the stack, leaving only #2 to be run.

```
\def\hi@namesplit@groupend@#1#2#3#4{%
    \find@end{ }{#3}{#1}{\@firstoftwo{#2}}{#4}%
}
\make@find@end{ }
```

3 Formatting

Once names have been read and separated into component parts, the package arranges and assembles those names into text amenable to inclusion in a citation. There are two matters to address here: Formatting the text of each name, and arranging and punctuating the list.

In terms of name formatting, the primary task is to abbreviate institutional names and affiliations. As a convenience, the package also checks that personal names "look" correct, in that they should not contain institution-like words such as "and" or "Corp."

The following macros are used in reference parameter definitions.

`\hi@nameproc@inst` Processes an institutional name prior to list insertion. #1 is the name, #2 a callback.

```
\def\hi@nameproc@inst#1#2{%
    \hi@abbrev@name{#1}{#2}%
}
```

`\hi@nameproc@person` Processes a personal name prior to list insertion. #1 is the name, #2 a callback (left on the stack). In this case, the callback will take two arguments: first, the processed name, and second, the last name only.

```
\def\hi@nameproc@person#1{%
    \hi@namesplit{#1}\hi@nameproc@person@split
}
```

#1 is the given name, #2 family name, #3 suffix, #4 institution, #5 et-al, #6 the `\hi@nameproc@person` callback.

```
\def\hi@nameproc@person@split#1#2#3#4#5#6{%
    \ifblank{#1}{%
        \PackageWarning\hi@names@pkgname{%
            Name `#1/#2/#3/#4' appears to have no given name.\MessageBreak
            This probably indicates a bug in the name parser.%
        }%
        \def\hi@nameproc@tmp{\ignorespaces}% Hopefully an unnecessary hack
    }{\def\hi@nameproc@tmp{#1}}%
    \notblank{#2}{\appto\hi@nameproc@tmp{ #2}}{}%
    \notblank{#3}{\appto\hi@nameproc@tmp{ #3}}{}%
    \expandafter\hi@nameproc@person@check\expandafter{\hi@nameproc@tmp}%
    %
    % The presence of a suffix also determines whether to put a comma before the
    % et-al.
    %
    \ifblank{#4}{%
        \ifblank{#5}{}{\appto\hi@nameproc@tmp{#5}}%
    }{%
        \appto\hi@nameproc@tmp{, }%
        \hi@nameproc@inst{#4}{\appto\hi@nameproc@tmp}%
        \ifblank{#5}{}{\appto\hi@nameproc@tmp{,#5}}%
    }%
    \ifblank{#2}{%
        \@expand{#6}{\hi@nameproc@tmp}{i}{#1#5}%
    }{%
        \@expand{#6}{\hi@nameproc@tmp}{i}{#2#5}%
    }%
}
\def\hi@nameproc@person@check#1{%
    \find@try\find@in{%
        {Inc.}{\hi@nameproc@person@warn{Inc.}}%
        {Co.}{\hi@nameproc@person@warn{Co.}}%
        {Office}{\hi@nameproc@person@warn{Office}}%
        {Department}{\hi@nameproc@person@warn{Department}}%
        { of }{\hi@nameproc@person@warn{ of }}%
        { to }{\hi@nameproc@person@warn{ to }}%
        { and }{\hi@nameproc@person@warn{ and }}%
    }{#1}{}%
}
\make@find@in{Inc.}
\make@find@in{Co.}
\make@find@in{Office}
\make@find@in{Department}
\make@find@in{ of }
\make@find@in{ to }
\make@find@in{ and }
\def\hi@nameproc@person@warn#1#2#3{%
    \PackageWarning\hi@names@pkgname{%
```

```
        The word `#1' was found in the name\MessageBreak
        #2#1#3\MessageBreak
        but was treated as a personal name.\MessageBreak
        You may want to make it an institutional name.\MessageBreak
        This occurred on
    }%
}
```

With regard to arrangement and punctuation, the following rules apply. First, as noted above, the list of names is separated into two sublists: a list of personal names and a list of institutional names. For each sublist, the last name is delimited by an ampersand and every other name is delimited by a comma. An "et al." can apply to either the personal name list or the institutional name list. Finally, if both lists contain entries, they are joined by a comma. For example:

- Author One, Author Two & Author Three (a coauthored work)
- Author One et al. (alternative to above example)
- Author One et al., Inst. (a work written by several people at a single institution)
- Author One, Inst. et al. (a work written by one person on behalf of several institutions)
- Author One & Author Two, Inst. One & Inst. Two (a joint report)

The following macros define the punctuation itself. They are also used as markers within the data structure implementation to help separate the lists, necessitating the `\iffalse` elements to render the macros distinct. `\hi@namelist@insts` separates the human author list from the institutional author list; `\hi@namelist@and` is the ampersand, and `\hi@namelist@comma` is the comma.

```
\def\hi@namelist@insts{\iffalse\hi@namelist@insts\fi}
\DeclareRobustCommand\hi@namelist@and{%
    \space\if@hi@inline and\else\&\fi\space
}
\def\hi@namelist@comma{\iffalse\hi@namelist@comma\fi,\space}
%
% Appearance of \hi@etal.
\DeclareRobustCommand\hi@etal{%
    \space\if@hi@inline and colleagues\else et al.\@hi@dottrue\fi
}
%
%
% Finders
\make@find@start@cs{hi@namelist@insts}{\hi@namelist@insts}
\make@find@in@cs{hi@namelist@insts}{\hi@namelist@insts}
\make@find@in@cs{hi@namelist@commaandinsts}{, \hi@namelist@insts}
\make@find@in@cs{hi@namelist@and}{\hi@namelist@and}
\make@find@in@cs{hi@namelist@comma}{\hi@namelist@comma}
```

The following macros are used in reference parameter definitions.

`\hi@namelist@addtwo` Adds two names to two lists. #1 and #2 are the lists (typically `\hi@kv@`⟨*parameter*⟩ macros); #3 and #4 are the names to add. If #2 is `\relax` then nothing is added.

```
\def\hi@namelist@addtwo#1#2#3#4{%
    \hi@namelist@add{#3}{#1}%
    \ifx#2\relax\else \hi@namelist@add{#4}{#2}\fi
}
```

`\hi@namelist@add` Receives an author name, assumed to be already processed for addition, and adds the name to the specified list macro.

```
\def\hi@namelist@add#1#2{%
    \hi@ifset#2{%
        \expandafter\hi@namelist@add@\expandafter{#2}{#1}{#2}%
    }{\def#2{#1}}%
}
```

#1 is the current list data, #2, is the item to add, and #3 is the list macro.

```
\def\hi@namelist@add@#1#2#3{%
    \find@in@cs{hi@namelist@commaandinsts}{#1}{%
        \hi@namelist@add@withinsts{#2}{#3}%
    }{%
        \find@start@cs{hi@namelist@insts}{#1}{%
            \def#3{#2, #1}\@gobble
        }{%
            \find@in@cs{hi@namelist@and}{#1}{%
                \hi@namelist@andtocomma{#2}{#3}{}%
            }{%
                \def#3{#1\hi@namelist@and#2}%
            }%
        }%
    }%
}
```

#1 is the item to add, #2 is the list macro, #3 is the list of author names, #4 is the list of institutions.

```
\def\hi@namelist@add@withinsts#1#2#3#4{%
    \hi@namelist@add@{#3}{#1}{#2}%
    \addto@macro#2{, \hi@namelist@insts #4}
}
```

After having split at the `\hi@namelist@and`, reconstructs the list with a new member. #1 is the new member, #2 is the list macro, #3 is a prefix, and #4 and #5 are the results of splitting the old list.

```
\def\hi@namelist@andtocomma#1#2#3#4#5{%
    \ifx\hi@param@set@hooks\relax\else
        \hi@param@addhook{%
            \hi@ifset\hi@kv@noetal{}{%
                \PackageWarning\hi@pkgname{%
                    In reference \@this@case, the\MessageBreak
                    list of names for \noexpand#2 contains\MessageBreak
                    3 or more names. You should specify\MessageBreak
                    `noetal' before adding these authors.\MessageBreak
                    This occured on
                }%
            }%
        }%
    \fi
    \def#2{#3#4\hi@namelist@comma#5\hi@namelist@and#1}%
}
```

Receives an institutional author name, assumed to be already processed for addition, and adds it to the specified macro list.

```
\def\hi@namelist@addinst#1#2{%
    \hi@ifset#2{%
        \expandafter\hi@namelist@addinst@\expandafter{#2}{#1}{#2}%
    }{\def#2{\hi@namelist@insts #1}}%
}
```

Adds the institution only if there is no person named in the list (basically, for last-name lists).

```
\def\hi@namelist@addinst@ln#1#2{%
    \hi@ifset#2{%
        \@expand{\find@start@cs{hi@namelist@insts}}{#2}{i}{%
            \@firstoftwo{\hi@namelist@addinst{#1}{#2}}%
        }{}%
    }{\hi@namelist@addinst{#1}{#2}}%
}
```

#1 is the current list, #2 the element to add, and #3 the list macro.

```
\def\hi@namelist@addinst@#1#2#3{%
    \find@in@cs{hi@namelist@insts}{#1}{%
        \hi@namelist@addinst@more{#2}{#3}%
    }{%
        \def#3{#1, \hi@namelist@insts #2}%
    }%
}
```

Add an institutional name to a list already containing at least one. #1 is the element to add, #2 the list macro, #3 the non-institutional name list, and #4 the current institutional name list.

```
\def\hi@namelist@addinst@more#1#2#3#4{%
    \find@in@cs{hi@namelist@and}{#4}{%
        \hi@namelist@andtocomma{#1}{#2}{#3\hi@namelist@insts}%
    }{%
        \def#2{#3\hi@namelist@insts#4\hi@namelist@and#1}%
    }%
}
```

3(a) Using Partial Lists. Some citation systems place the institutional names in a different place from the personal names. This package provides infrastructure for separating out these sublists, although it is not used in the package itself.

`\hi@namelist@instpart` Retrieves only the institutional part of the given name list macro, and runs the given callback on it. If there is no institutional part, nothing is run.

```
\def\hi@namelist@instpart#1#2{%
    \@expand{\find@in@cs{hi@namelist@insts}}{#1}i{%
        \@tworun\@gobble{#2}%
    }{}%
}
```

`\hi@namelist@personpart` Retrieves only the human author part of the given name list macro, and runs the given callback on it. If there is no human part, then nothing is run.

```
\def\hi@namelist@personpart#1#2{%
    \expandafter\hi@namelist@personpart@\expandafter{#1}{#2}%
}
\def\hi@namelist@personpart@#1#2{%
    \find@in@cs{hi@namelist@commaandinsts}{#1}{%
        \hi@namelist@personpart@chompinsts{#2}%
    }{%
        \find@start@cs{hi@namelist@insts}{#1}{}{%
            #2{#1}%
        }%
    }%
}
\def\hi@namelist@personpart@chompinsts#1#2#3{%
    #1{#2}%
}
```