Tagging with LaTeX — Part 1: author considerations

Ross Moore*

Abstract

Successful tagging within PDF files generated from LaTeX source encourages a change in viewpoint on the nature and intent of the LaTeX coding. Using explicit examples from a real-world document, we illustrate how to capture such a change within the LaTeX source, for various structural elements. Other issues for creating archival and accessible PDF documents are discussed.

1 Introduction

With 'Tagged PDF' being the accepted method for creating PDF documents enriched to satisfy Accessibility requirements [5, 6], this article is intended to address the main issues that authors and editors should be aware of, with regards to tagging and LaTeX usage. Examples are taken from a real-world fully-tagged research report, prepared in LaTeX but also employing extra coding written by the author, in a package named `tpdf` that handles the technical aspects of producing 'Tagged PDF'. That research report is the one used by the author in the talk [7] at TUG 2019, and delivered remotely using Zoom collaboration software. It is based on a publication from the U.S. National Parks Service [8].

This article is not meant to be an introduction to the use of the `tpdf` package, but more about the kind of extra considerations that authors and editors alike should be making, to allow tagging to be performed successfully and usefully.

1.1 Requirements for U.S. federally funded research publications

As some justification as to why 'Tagged PDF' is both relevant and desirable, we note some U.S. Government guidelines and requirements.

- United States Access Board; Information and Communication Technology (ICT) Final Standards and Guidelines. [1]
  Section 508 ICT Refresh, §504.2.2:

  > ... be capable of exporting PDF files that conform to ANSI/AIIM/ISO 14289 -1:2016 (PDF/UA-1) ...

- National Science Foundation, Q&A: public access policy. [2]

  > ... possess a minimum set of machine-readable metadata elements ...;
  >
  > be managed to ensure long-term preservation; ...

By producing documents conforming to published standards both PDF/A [3] and PDF/UA [4], these obligations can be met in full, if not surpassed.

2 Tagging commands for special content

It is a common typographical practice to use different styling to present names of books, magazines, or publications which have a particular relevance to the topic under discussion. Certainly the fact of a different style being used indicates to a fully-sighted reader that there is a special significance, but not what that significance actually is. That has to be deduced from context. With tagging, that significance can be made explicit. And with LaTeX source this is very easy to do.

For example, the Night Skies Project report has a page which mentions the various Acts of Congress which underpin their work; see Figure 1. The names of these acts appear in italics. This was originally done by simply specifying

```
\textit{Organic Act of 1916} ...
```

Changing this by inventing a macro `\nrpsAct` the true intention is captured, at least within the LaTeX source. For typesetting purposes the expansion of this new macro is given by:

```
% for names of Acts of Congress
\newcommand{\nrpsAct}[1]{\textit{#1}}
```

which typesets exactly as `\textit` does. But now, when it comes to generating tagging for a Tagged PDF version, as seen on the left-hand side of Figure 1, one can use coding as in Figure 2.

In that coding we see that firstly a new macro is created, named `\NRPS@Act`, which refers to the same code-block as currently does `\nrpsAct`, by using TeX's `\let` primitive. Then a new code-block is defined under the name `\TPDF@NRPS@Act`. When this block is executed, after reading the parameter text as `#1` a group is started with `\begingroup`. Structure tagging named as `CongressAct` is implemented, along with a counter for such structure elements. The tagging that would otherwise be performed by `\textit`, through the style change macro `\itshape`, is suppressed since we are using the `CongressAct` structure instead. Then `\TPDF@NRPS@Act` is called with the `#1` parameter text, to do the typesetting and generate the associated content tagging structures that would normally have been done when `\textit` is called. The grouping is closed using `\endgroup` after which normal tagging of the paragraph content is resumed. To have this new code

* Macquarie University, Sydney, Australia

block activated at the correct time, we re-assign the macro name `\nrpsAct` to point to the modified expansion, again using `\let`. A final command, using `\TPDF@appendto@RoleMapDict`, allows PDF reader software to treat the non-standard structure tag name `/CongressAct` (which we just invented) in the same way as the standard `/Span` tag name.

2.1 Aside on how TEX macros work

We all make great use of TEX/LATEX commands, but how well do we really understand how they work? Upon reading a string beginning with the `\` character, TEX (the program, which underlies LATEX processing) creates a control-sequence token (also called a 'macro'); but what is it really? Essentially there is the name token (such as `\title`, `\date`, `\section`, etc.) and a block of coding that is to be executed, or data to be inserted into the processing stream, when the name token is encountered while working through a document's source. This block of coding resides somewhere in the computer's memory, associated with the running job. The memory location has an address, specified by a number which allows the code to be found and used. In other computing languages, one refers to use of such address locations as *pointers* to data or code-blocks. Thus effectively a macro is just a *named pointer* to a block of code or content to be used.

When a new macro is first defined (e.g., using LATEX's `\newcommand` or similar, or with TEX's `\def`, `\edef` or `\let`) one can think of a key–value pair consisting of the new name token and its associated memory address, being pushed onto the top of a stack of all the currently defined such name tokens. Now when a command is encountered, the processor starts at the top of this stack, reading downwards until a token having the same name is encountered. Then the corresponding address is used to find the data or code that is to be handled next. Frequently the macro definition will occur within an environment or grouping (using `{...}`, or `\begingroup` ... `\endgroup` or similar). Upon entering the grouping, the current location of the stack is recorded as a *stack-pointer*, say. When the grouping closes, any name token entries added later than that recorded level are discarded — except for any that have been declared as *global* (using TEX's `\gdef`, or `\xdef` or `\global\let`). All of LATEX's counters are globally defined and updated. This is the kind of mechanism that allows the same macro name to refer to different values, at various stages of processing; a concept known as *scoping* of variable values.

With this interpretation we can better understand how the coding in Figure 2 works. Firstly a new pointer named `\NRPS@Act` is made, pointing to the value of `\nrpsAct`, since that is going to be redefined to point instead to the coding of `\TPDF@NRPS@Act`. But part of the coding of this is to use `\NRPS@Act` itself. This is done within a grouping, because the command

```
\TPDF@style@structure@suppress
```

makes some changes to other macros, which changes need to be scoped to within that grouping only. In particular a command

```
\TPDF@maybe@taggedparagraphmiddle
```

is set to `\relax`, (i.e., to do nothing) and

```
\aftergroup
 \TPDF@maybe@taggedparagraphmiddle
```

issued. This causes the coding for resumption of tagging within the surrounding paragraph to be delayed until `\endgroup`, when otherwise it would have occurred upon completion of `\textit{...}`. It works since the named pointer's associated value reverts back to what it was prior to `\begingroup`, as the pointer to the `\relax` value has been removed from the stack.

2.2 patching as 'hacking' or 'enhancement'

This technique, of capturing a pointer and using it within a new code-block, to replace what LATEX would do evaluating a particular macro-name, can be considered as a trick for code-hacking. Or it can be considered as a legitimate technique for enhancing the results of typesetting with other structures that may be important for the job as a whole. Such patching is used in the nameref package, part of the hyperref bundle of packages, for enhancing LATEX commands to produce named destinations to act as anchors for hyperlinks to section titles, figure/table captions, and other usages of the `\label` command. Other packages employ this technique to enhance footnotes, cross-references, citations and more, with hypertext features.

Later in Section 6.1 we give an example of how pointers can be used to resolve an apparent inconsistency created by the same LATEX macro being patched by two different packages [20, 21]. And in Section 7.2 we use them to resolve a difficulty with mathematical source initially intended for LuaTEX. Furthermore these are fundamental to how the tpdf package works, to produce Tagged PDF output; as will be discussed in more detail in a later paper in this series.

3 LATEX environments indicate structure

Common LATEX environments, such as `flushleft`, `flushright`, `center`, `quotation`, etc. adjust the

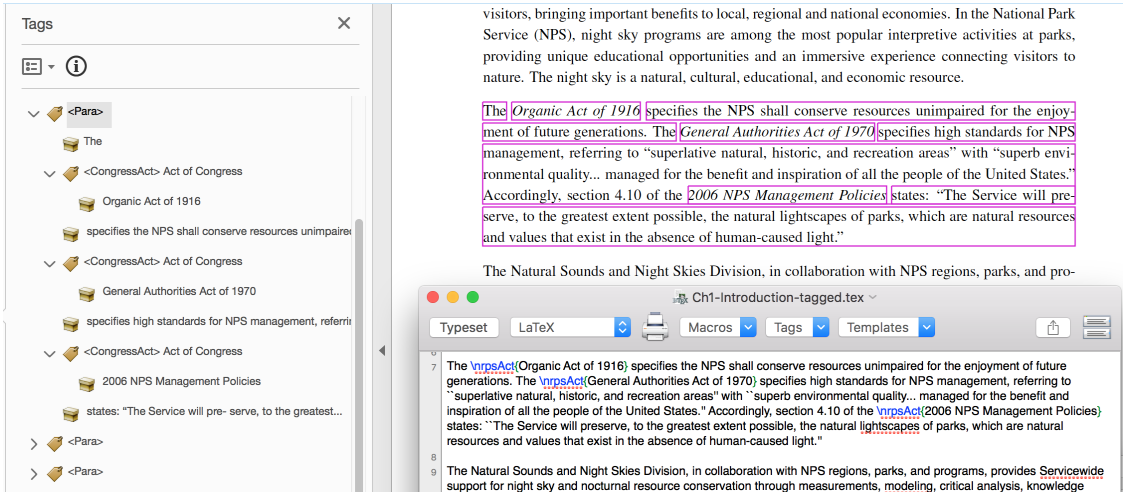


Figure 1: Tagging of Acts of Congress, using a distinctive macro name `\nrpsAct`.

```
\let\NRPS@Act\nrpsAct
\def\TPDF@NRPS@Act #1{%
 \begingroup
  \TPDF@advancecounter{CongressAct}%
  \edef\TPDF@theseparams{{CongressAct}{CongressAct.\TPDF@counter@CongressAct}}%
  \expandafter\TPDF@newstructnode\TPDF@theseparams
   {}{}{}{Act of Congress}{}{}{}%
  \TPDF@style@structure@suppress{\itshape}%
  \NRPS@Act{#1}%
 \endgroup
}
\let\nrpsAct\TPDF@NRPS@Act
\TPDF@appendto@RoleMapDict{/CongressAct /Span}
```

Figure 2: Code to initiate generation of the structure tagging as seen in Figure 1.

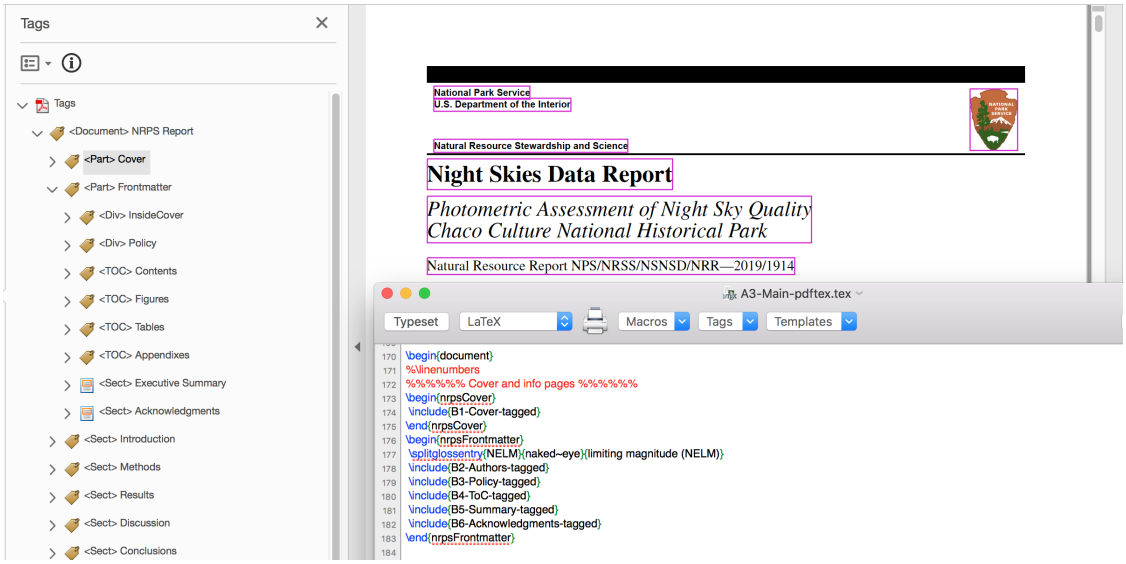


Figure 3: Structure tagging of the front cover and front-matter section of the research report.

```
% declare some environments, to allow hooks for structural tagging
\newenvironment{nrpsCover}{%
 \pseudochapter{Cover}%
}{\unskip}

\newenvironment{nrpsFrontmatter}{%
 \pseudochapter{Titlepage}%
}{\unskip}

%%  RRM:  use this to get a bookmark, at the  chapter level
%%   without making a ToC entry
\newcommand{\pseudochapter}[1]{%
 \begingroup
  \renewcommand{\H@old@schapter}[1]{}%
  \NR@schapter{#1}%
  \addbookmarksline{chapter}{#1}%
 \endgroup
}
\newcommand{\addbookmarksline}[3][toc]{%
 {\renewcommand{\addtocontents}[2]{}%
  \addcontentsline{#1}{#2}{#3}%
 }%
}
```

Figure 4: Code for environments named for the structure tagging as seen in Figure 3, and the extra bookmarks as in Figure 5.

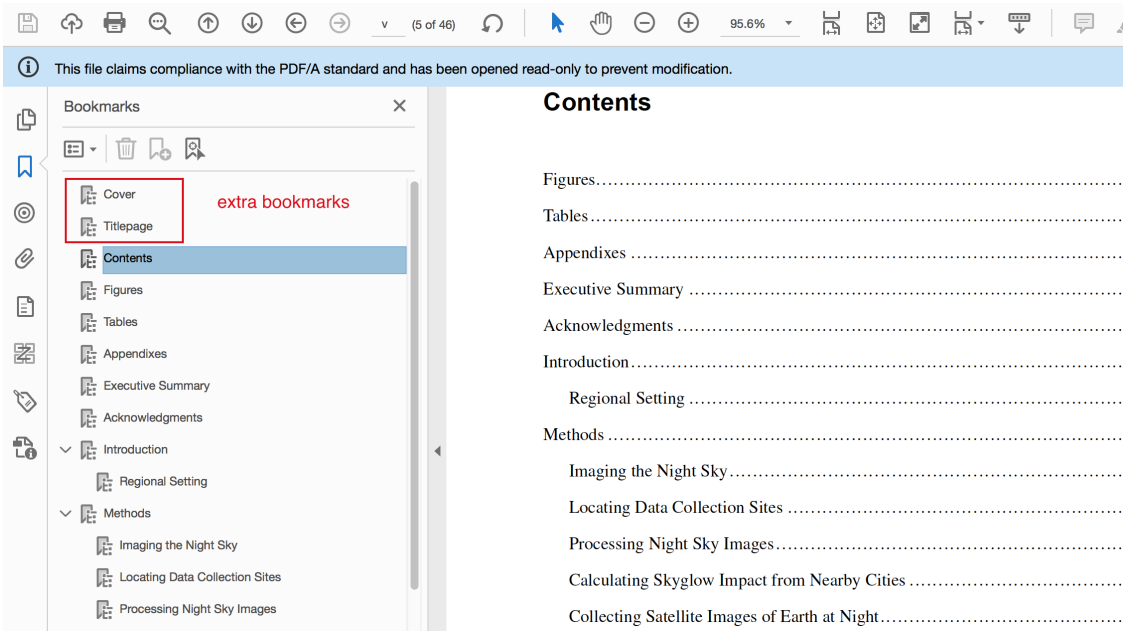


Figure 5: Extra Bookmarks generated for front cover and title-page of the research report, resulting from the coding shown in Figure 4.

way paragraph content is displayed. This is a visual hint to a sighted reader that there is a special meaning attached to the enclosed content. But that is exactly one of the main reasons for tagging, to attach a name to a block of content. That is, a screen-reader will not do anything special with how the content is presented, but it can alert a visually-impaired user to the fact that it has been tagged specially. The `tpdf` package, when used to create Tagged PDF, automatically produces the appropriate tagging for such environments. It also specifies suitable attributes which hint at the desired visual layout, for tools that need to construct a layout; e.g., for small screens, say.

Other environments, `tabular`, `verbatim` and common list-like environments `itemize`, `enumerate`, `description` and more, indicate that their content may involve special typesetting and/or layouts. For the list-like environments, in particular, the question arises as to whether the listed information remains as part of the preceding (perhaps surrounding) paragraph, or whether it constitutes a separate (vertical) block with the paragraph having finished already. LaTeX has no formal way to differentiate between these cases, however when coded such as follows with a blank line,

```
... satisfying conditions in this list.

\begin{enumerate}
 \item
```

one would expect the paragraph to finish prior to commencement of the list, as triggered by the blank line. Whereas with the following coding

```
... satisfying the conditions:
%
\begin{enumerate}
 \item
```

there is no (uncommented) blank line; indeed the list items might each be supplying a possible end to the preceding incomplete sentence. A commented blank line is only for clarity in the LaTeX source; it can be omitted altogether, with no change to processing.

3.1 New environments declaring structure

Just as new commands can be defined to convey the intention associated with styling, so also can new environments be defined, doing no extra typesetting, but conveying a name to be associated with definite structure within the document. The first two pages of the Night Skies Project report are meant to be printed on special stiffer paper, and serve as the 'cover' for a printed version. Similarly the last two pages are for the back cover, printed on the same stiff paper. Thus in the PDF, it makes sense to regard these 2-page blocks as separate parts of a structured document, tagged as `Part`.

Figure 3 shows an environment `nrpsCover`, that wraps the first two pages, but otherwise produces no visual content. Similarly there is an environment `nrpsFrontmatter` which encloses the title-page, the Table of Contents, List of Figures, Lists of Tables and Appendices, and further information generic to the organisation of the project, rather than being unique to the project report and results. These two structures do not use a separate sectioning command, but they are sufficiently important that a *bookmark* is appropriate within an electronic document.

Coding as shown in Figure 4 achieves this, using a macro `\pseudochapter` which itself uses macros from other packages. The `\NR@schapter` is a patch of `\@schapter` resulting from `\chapter*` for starting an un-numbered chapter. But there is no anchor text, so the prior redefinition

```
\renewcommand{\H@old@schapter}[1]{}%
```

causes that part to be skipped, leaving just the specification of a named destination. Then we have

```
\addbookmarksline{chapter}{#1}
```

to add a bookmark without also creating a Table-of-Contents entry. Its definition, shown in Figure 4, calls up `\addcontentsline`, but with a disabled `\addtocontents` command, which will just gobble its arguments.

These two new commands `\pseudochapter` and `\addbookmarksline` are essentially patching macros `\NR@schapter` and `\addcontentsline` respectively; but this time *removing* functionality, rather than adding extra coding. The result is an invisible chapter heading with a bookmark but no ToC entry. This also provides a place for structure to be attached, as shown in Figure 3.

3.2 Structure destinations

Observe in Figure 5 that the bookmarks use the icon shown here at right below. This indicates that the bookmark includes a specified *structure destination* (see [14, Table 8.4] or [16, Table 151]), as well as the usual destination area of a printed page.

normal bookmark with structure destination

In the visual view there is no noticeable difference in the result upon clicking on a bookmark. But when the PDF file is exported to XML from Acrobat Pro

DC [17], then targets and links are automatically produced for each bookmark; *viz.*

```
<bookmark title="Introduction">
<destination structID="LinkTarget_591"/>
<bookmark title="Regional Setting">
<destination structID="LinkTarget_622"/>
</bookmark>
</bookmark>
  ...
<Chap>
<H2 id="LinkTarget_591" >Introduction</H2>
 ...
<Sect>
<H3 id="LinkTarget_622" >Regional Setting</H3>
 ...
```

With a *structure destination* the target is the first structure having textual content, as a (perhaps) substructure child of a structure element specified in the bookmark's dictionary, or indeed the specified structure itself. Usually this will be the `<H2>` or `<H3>` title text for a chapter or section rather than their parent `<Chap>` or `<Sect>` structures, but it can be different. For example the '`Cover`' bookmark in Figure 5 goes to the first piece of text on the Cover page, namely '**National Park Service**', as seen in Figure 3.

Having a *structure destination* as just described, when exported for other technologies (in particular Assistive Technology for screen-readers, etc.), a bookmark now behaves as would be expected in XML or HTML web-based pages, say. On the other hand, when using 'ordinary' bookmarks, Acrobat Pro DC [17] uses a heuristic to try the deduce the best piece of text to be chosen as the target, given the area of the page specified by its destination key. While working well in most cases, this can come earlier than one would expect from the visual view, and the same target may be found for multiple bookmarks, which is clearly incorrect.

This concept of *structure destination* can be used for hyperlinks as cross-references, ToC entries, citations, etc., at least with PDF 2.0 [16, §12.3.2.3 and Table 202]. These must be specified explicitly, by adding XML attributes directly to the PDF dictionary for the structure elements of both the link and its target. Unless it is also target for a bookmark, the target needs to associate an explicit `id="..."`, with `...` replaced by a unique name. This is achieved with `/id (...)` as key–value entry, while the link structure needs `xlink:target="..."`, and include also a namespace declaration as follows.

```
xmlns:xlink="http://www.w3.org/1999/xlink"
```

This latter is best done using a `ClassMap` entry for the structure, as `/C /XLink`. While structure destinations were only introduced with PDF 2.0 [16], by including the `/SD` key with value, as well as an ordinary `/D` key and value, in a *go-to action* (`/A`) dictionary, one achieves both forward- and backward-compatibility with all versions of PDF for internal hyperlinks within the document. Reader software is supposed to respect implement the `/SD` action in preference to the `/D`, if able to do so. Otherwise, as with older software, it will be ignored and `/D` used.

4 Combining environments and commands

As well as the first two pages being cover-page material to be printed on special paper stock, so also are the last two pages. Figure 6 shows the result, with special tagging of the content appearing on the Back cover page, using specially defined commands `\nrpsService`, `\nrpsDepartment` named to convey the intention of the material in their arguments, as described in Section 2. Also `\nrpsPlaceLogo`, and specially named environments are used.

The original coding for those last two pages is given in Figure 7, which can be seen as a quite complicated mixture of styling and layout commands, interspersed with the content to be displayed. That kind of coding has been simplified by absorption into macro and environment definitions as shown in Figure 8. Note that `\nrpsIssue` is Metadata that identifies the particular publication, so should be set at the beginning of the document source where such data is easily found; e.g., by (in this case):

```
%% MetaData that is re-used
\providecommand{\theissue}{310/152635}
\providecommand{\theissueII}{\theyear/1914}
\providecommand{\theyear}{2019}
\providecommand{\thedate}{April \theyear}
\providecommand{\thejournal}{Natural
 Resource Report NPS/NRSS/NSNSD/NRR}
\edef\nrpsIssue{\theissue, \thedate}
```

Use of `\providecommand` is recommended for these, since it only has an effect if the macro is currently undefined. This means that it does not interfere with a more sophisticated workflow in which values for `\theissue`, `\thejournal`, `\thedate`, etc. may have been supplied already. The document's title and subtitle could also be provided this way. However, in practice these are set at different sizes in different places within the document, which can require some extra markup to create the best visual layout. So we do not do this here.

Another aspect visible in Figure 6 is the use of a tabular environment simply for the purpose of visual layout. There is no semantic meaning attached to this environment usage, so there is no associated structure tagging; whereas the content of each cell

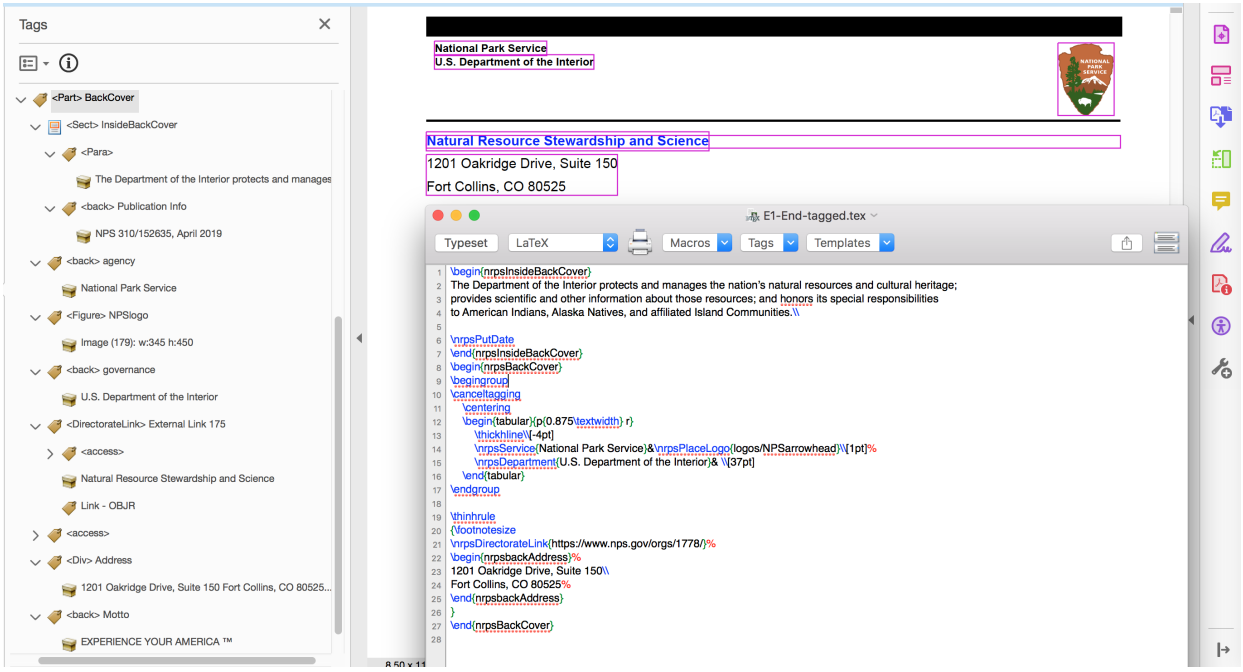


Figure 6: Back cover pages, and the coding for its structures.

```
\clearpage
\pagestyle{empty}
\strut
\vfill
The Department of the Interior protects and manages the nation's natural resources and cultural heritage;
provides scientific and other information about those resources;
and honors its special responsibilities to American Indians, Alaska Natives, and affiliated Island Communities.
\\

NPS \hl{XXXXXX}, March 2019

\clearpage
\newgeometry{lmargin=0.75in, rmargin=0.75in, tmargin=0.75in, bmargin=1in}
\renewcommand{\arraystretch}{0.65} % Default value: 1
\begin{table}
    \centering
    \begin{tabular}{p{0.875\textwidth} r}
        \thickhline
        \\[-4pt]
        \scriptsize{\textsf{\textbf{National Park Service}}}
   & \multirow{3}{*}{\includegraphics[width=0.075\textwidth]{logos/NPSarrowhead}}\\[1pt]
        \scriptsize{\textsf{\textbf{U.S. Department of the Interior}}}     & \\[5pt]
        %\noalign{\hrule height 1.2pt}
    \end{tabular}
\end{table}

\hrule height 1.2pt
%\vspace{4pt}
\footnotesize{
\textsf{\href{https://www.nps.gov/orgs/1778/}{\color{blue}
 \textbf{Natural Resource Stewardship and Science}}}\\
\textsf{1201 Oakridge Drive, Suite 150}\\
\textsf{Fort Collins, CO 80525}\\
}

\vfill
\textsf{\textbf{EXPERIENCE YOUR AMERICA $\rm^{\textsf{TM}}$}}
```

Figure 7: Original coding for the content displayed on the Back-Cover pages.

```
\newcommand{\thickhline}{\noalign{\hrule height 14.15pt}}
\newcommand{\thinhline}{\noalign{\hrule height 1.2pt}}
\newcommand{\thinhrule}{\hrule height 1.2pt}

\def\nrpsPutDate{\noindent\nrpsTheDate}
\def\nrpsTheDate{NPS \nrpsIssue}
\def\nrpsDirectorateName{Natural Resource Stewardship and Science}

\newcommand{\nrpsService}[1]{\scriptsize{\textsf{\textbf{#1}}}}
\newcommand{\nrpsDepartment}[1]{\scriptsize{\textsf{\textbf{#1}}}}
\newcommand{\nrpsDirectorate}[1]{\scriptsize{\textsf{\textbf{#1}}}}
\newcommand{\nrpsDirectorateLink}[1]{{\footnotesize\bfseries\sffamily\color{blue}%
 \href{#1}{{\nrpsDirectorateName}}}}
\newcommand{\nrpsPlaceLogo}[1]{\multirow{3}{*}{\includegraphics[width=0.08\textwidth]{#1}}}

\newenvironment{nrpsbackAddress}{\\\noindent\sffamily\ignorespaces}{\unskip}%
\newenvironment{nrpsInsideBackCover}{\clearpage \strut \thispagestyle{empty}\vfill}{}
\newenvironment{nrpsBackCover}{\clearpage
 \newgeometry{lmargin=0.75in, rmargin=0.75in, tmargin=0.75in, bmargin=1in}
 \renewcommand{\arraystretch}{0.65}% Default value: 1
 \thispagestyle{empty}}%
 {\vfill \begin{nrpsMotto}
  EXPERIENCE YOUR AMERICA\texttrademark
 \end{nrpsMotto}}
\newenvironment{nrpsMotto}{\sffamily\bfseries}{}
```

Figure 8: Macro and environment definitions for styling the content on the Back-Cover pages.

does have sematic meaning, associated with the specially defined macro names, as mentioned above. This is an important counter-example to the discussion of Section 3. Thus within a tagging context, as with the tpdf package, there is no need for automatic tagging at this point; hence the use of `\canceltagging`, which affects also the styling commands in the macro expansions; these revert to having just their usual LaTeX expansions. In a future version, this could be incorporated into the tagging expansion of the nrpsBackCover environment, as there is really nothing else there that is associated with implicit semantic macros or environments. It is not done in this article to illustrate the adaptability of tagging to particular semantic requirements within a specific document or class of documents. To allow the document to be processed *without* having the tpdf package loaded, use coding such as follows.

```
\makeatletter
\AtBeginDocument{%
 \@ifpackageloaded{tpdf}{}%
  {% coding to cancel commands
   \let\canceltagging\relax
  }%
}% end of \AtBeginDocument
\makeatother
```

Use of `\AtBeginDocument` delays testing in case the package is loaded later within the preamble section of the document source.

Also observe in Figures 6 and 7 that the original use of `\begin{table} ... \end{table}` has become simply `\begingroup ... \endgroup`, since there is no intention of this material floating elsewhere, and there is no need for a caption or any numbering or other associated constructs.

This use of a tabular environment to control the visual layout occurs also on the front cover, as can be seen in Figure 3. Indeed the coding is identical apart from how the `\nrpsDirectorateText` is displayed, as the name 'Natural Resource Stewardship and Science' of the Directorate [9]. On the front cover, this is part of the tabular, just above a thin rule that finishes this material. On the back cover it appears in larger type below the rule, as anchor text to the website of the Directorate. With this double usage, the structure is declared specially as `/DirectorateLink`, which uses a `RoleMap` entry to exhibit the behaviour of a `/Link`. The coding shown in Figure 8 has macro definitions for both the front- and back-cover instances.

5 'alternative text' for Figures

Accessibility guidelines require that figures be accompanied by 'alternative text' [11] that can help a visually or cognitively disabled person to understand the semantic content associated with the inclusion of a figure. By a 'figure' here, we mean non-textual content that has a definite semantic meaning within the context of the electronic document. The alternative text is read by screen readers, and other Assistive technology, in place of the figure itself, which

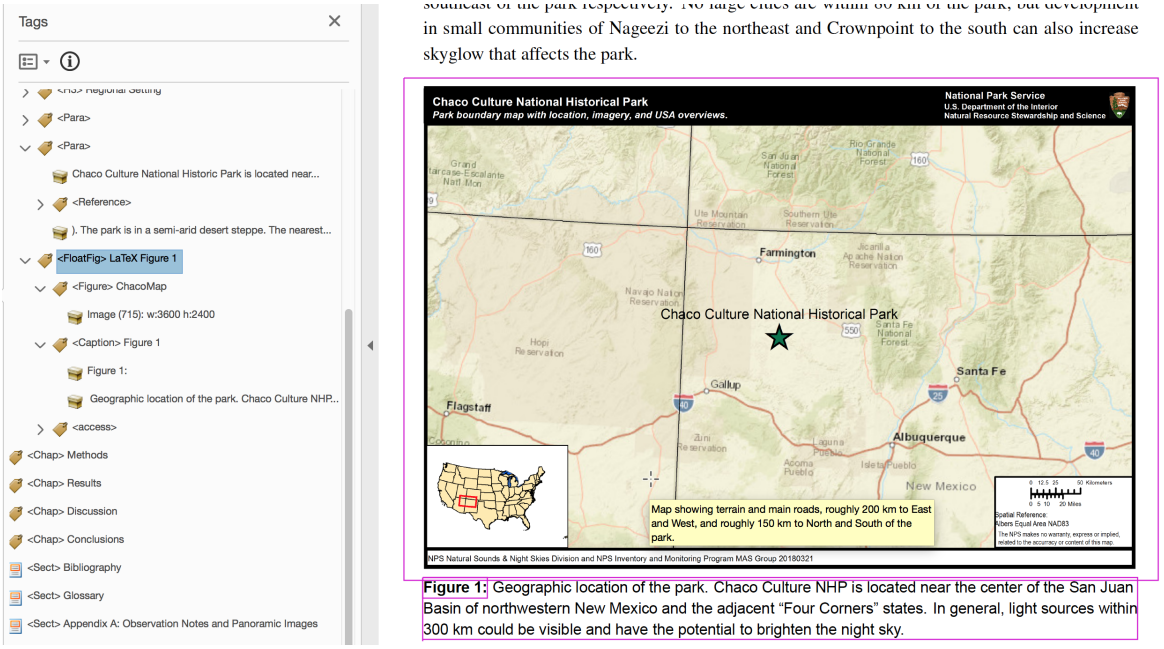


Figure 9: A 'floating' figure with numbered caption following immediately after the image.

may be a photograph or other image, but need not necessarily be so.

Note that this is logically different from LaTeX's figure environment, which is semantically a grouped block of content within the document which needs to be shown together. For pagination considerations this group, or `/Div`, may need to 'float' to a page later than where the information might otherwise logically be read, or an image first viewed. Usually there will be a caption which describes some of the semantic meaning of the non-textual content, which is often an image or graphic, but need not always be. For example, within this article many of the 'figures' are code listings, which in a Tagged PDF document would be tagged as either `/Code` or `/BlockQuote`, or something equivalent.

A common misconception is that the alternative text for an image is just the same as the 'caption' of a figure. This is quite wrong, especially as any caption should also be available to the Assistive Technology (see Recommendation [6, §4.3.1.3]). Rather the 'alternative text' should complement any discussion in the surrounding text, which could well include a caption. Thus it can explain the 'What?' (is in the image/figure) while the caption is giving the 'Why?' (is an image used here). See Figure 9 for a typical example, where the alternative text is seen in a popup as the mouse hovers over the image. The automatically-generated 'Figure 1:' is in a separate text container to the bulk of the caption.

See [6, §4.3.1] for a discussion and examples of tagging figures and captions, where the main expectation is for a `/Caption` tag to follow immediately after the `/Figure`. However, it is also noted there [6, §4.3.1.1, Example D] that an actual `/Caption` need not always occur.

As a first example of how a non-captioned figure can be implemented, consider the logo of the National Parks Service, as displayed on the front cover and seen in Figure 3, and also on the back cover in Figure 6. When building the Tagged PDF version of the report the `\nrpsPlaceLogo` macro is replaced by an alternative declared as follows.

```
\def\TPDF@nrpsPlaceLogo #1{%
 \multirow{3}{*}{\tagFigure[NPSlogo]{%
  National Park Service logo, in shape of
  stone arrowhead with Sequoia tree, bison,
  lake and mountain scenery}{%
 \includegraphics[width=0.08\textwidth]%
 {#1}}}}
```

Here the `\tagFigure` macro is defined in the `tpdf` package; it has an argument for the alternative text, as well as an optional name for the figure, and final argument for placing the figure content itself; in this case as an included image of specified width.

As there is no caption, the alternative text is providing a brief answer to the question 'What is the content conveyed by the image?' [12]. In the logo, one sees a collection of graphic elements symbolising vegetation and wildlife, as well as scenic, recreational, historical and archeological values [10].

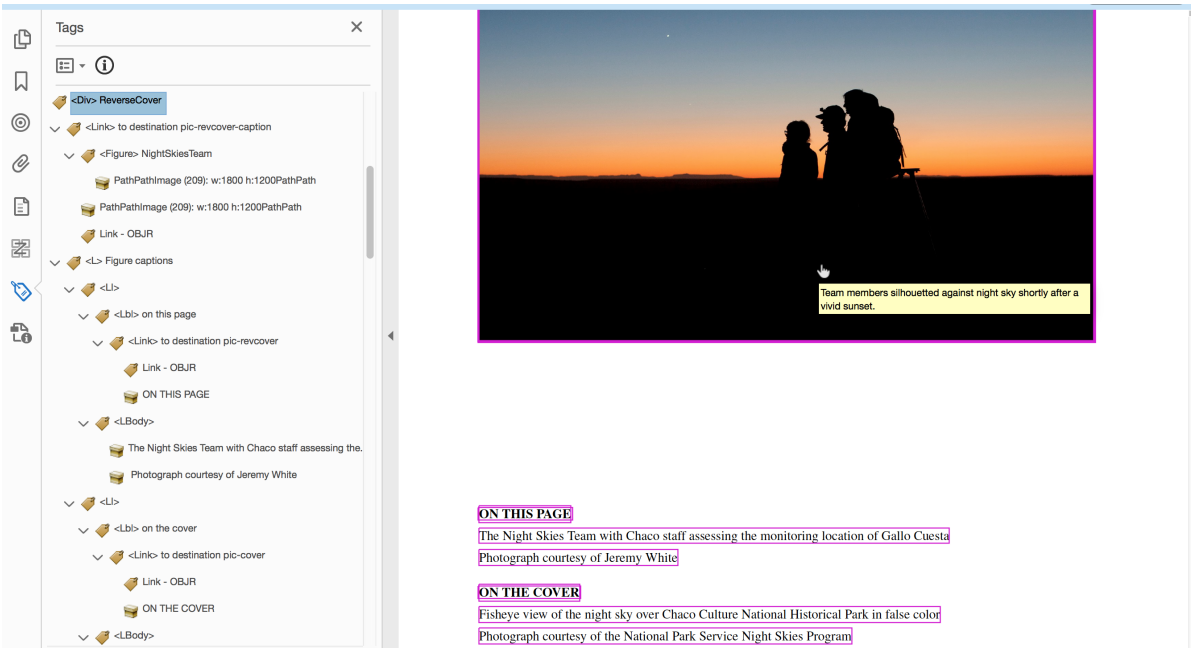


Figure 10: Photo appearing on reverse side of the cover page, along with captions and attributions, for that image and the one appearing on the cover page itself. The image is the anchor for a hyperlink to the caption.

```
\begin{nrpsReverseCover}

% Second page photo
\begingroup
 \hypertarget{pic-revcover}{}% creates anchor-point above the image
 \centerline{\hyperlink{pic-revcover-caption}{%
  \tagFigure[NightSkiesTeam]{Team members silhouetted against night sky shortly after
   a vivid sunset.}%
  {\frame{\includegraphics[width=\textwidth]{photos/fig02-cover2.jpg}}%
     }}% end of  \hyperlink
 }% end of  \centerline
\endgroup
\vfill

% Figure captions
\begin{figurecaptionlist}%
\begin{figurecaption}{pic-revcover}{on this page}% capitalised by the environment
The Night Skies Team with Chaco staff assessing the monitoring location of Gallo Cuesta\\
Photograph courtesy of Jeremy White
\end{figurecaption}

%\newline
\begin{figurecaption}{pic-cover}{on the cover}% capitalised by the environment
Fisheye view of the night sky over Chaco Culture National Historical Park in false color\\
Photograph courtesy of the National Park Service Night Skies Program
\end{figurecaption}%
\end{figurecaptionlist}%

\end{nrpsReverseCover}%
```

Figure 11: Coding for the reverse cover-page photograph and attributions. The image is linked to its caption, and vice-versa.

When tagging is not implemented, one can simply define the expansion as follows, to just place the image normally.

```
\providecommand{\tagFigure}[3][]{#3}
```

This is best delayed using `\AtBeginDocument` as described earlier in Section 4.

For a second example where now there is some surrounding context, Figure 10 shows a photograph used inside the front cover of the research report. Notice how the alternative text appears within a popup 'tool-tip'. The image is followed, down the page, by a 'caption' and attribution of the source of the photograph, as well as similar information for the image used on the front cover. As such, we actually have a list of figure captions, rather than a single caption following each figure. Indeed these 'captions' are structurally more like 'end-notes' for the two images on either side of the cover page.

The coding to place this material is shown in Figure 11, using new commands and environments defined as follows.

```
\newenvironment{nrpsReverseCover}{%
 \newgeometry{margin=1in}\strut\vfill}{}

\newenvironment{figurecaptionlist}%
 {}{\unskip}%

\makeatletter
\newenvironment{figurecaption}[2]%
 {\noindent\footnotesize
  \Hy@raisedlink{%
   \hypertarget{#1-caption}{}}%
  \hyperlink{#1}{{\bfseries
   \MakeUppercase{#2}}}\\}%
 {\unskip}
\makeatother
```

As seen in this coding, commands `\hypertarget` and `\hyperlink` from the hyperref package, are used to link each caption to a target destination located just above the corresponding image (e.g., named `pic-revcover`), as well as linking from the image to caption (with destination `pic-revcover-caption`). This explains the detailed tagging as seen in Figure 10, which is in accordance with the recommendations in [6, §4.2.7.1], and is analogous to Example C found there. Another way to see the alternative text is upon export as text, or into XML as follows.

```
<Div>
<Link><Figure Alt="Team members silhouetted against
 night sky shortly after a vivid sunset.">

<ImageData src="images/A3-Main-pdftex_img_2.jpg"/>
</Figure>
</Link>

<L>
<LI>
<Lbl>
<Link>ON THIS PAGE</Link>
</Lbl>

<LBody>The Night Skies Team with Chaco staff
 assessing the monitoring location of Gallo Cuesta
 Photograph courtesy of Jeremy White</LBody>
</LI>

<LI>
<Lbl>
<Link>ON THE COVER</Link>
</Lbl>

<LBody>Fisheye view of the night sky over Chaco
 Culture National Historical Park in false color
 Photograph courtesy of the National Park Service
  Night Skies Program</LBody>
</LI>
</L>
</Div>
```

The above XML version gives an idea of the kind of tagged information that would be passed to Assistive Technology applications.

6 Metadata & Titlepage

One cannot downplay the rôle that Metadata plays in helping to allow documents to be found on the internet, via search-engines, and otherwise located in document repositories. The more accurate and detailed the Metadata, the easier it becomes to decide whether a given document is the one that most appropriately provides the knowledge that one may be seeking. In the case of PDF documents, by Metadata we mean not only the information displayed visually on the title-page, but also keywords, copyright status and internally generated structural information, such as creation/modification dates and more.

For documents conforming to PDF/A archival specifications, the XMP packet [13] is the method that allows all such pieces of Metadata to be included in a way that can be easily extracted in XML format. It is as though the PDF document is carrying along its own Library of Congress Catalogue card. Figure 12 shows a view of part of this information, with many fields filled. Most of these fields are *not* provided automatically by LaTeX. Even more fields are shown in Figure 13, along with coding that writes information into an external file named `\jobname.xmpdata` based on the name of the LaTeX source file (in this case A3-main-pdftex.tex). This file is used by the pdfx package to create the full XMP packet, by inserting the information into a template (named pdfa.xmp), creating a new file pdfa.xmpi that is then embedded uncompressed into the PDF/A file being built using the pdfTeX processing engine. The package options given as `\pdfxopts` are used via

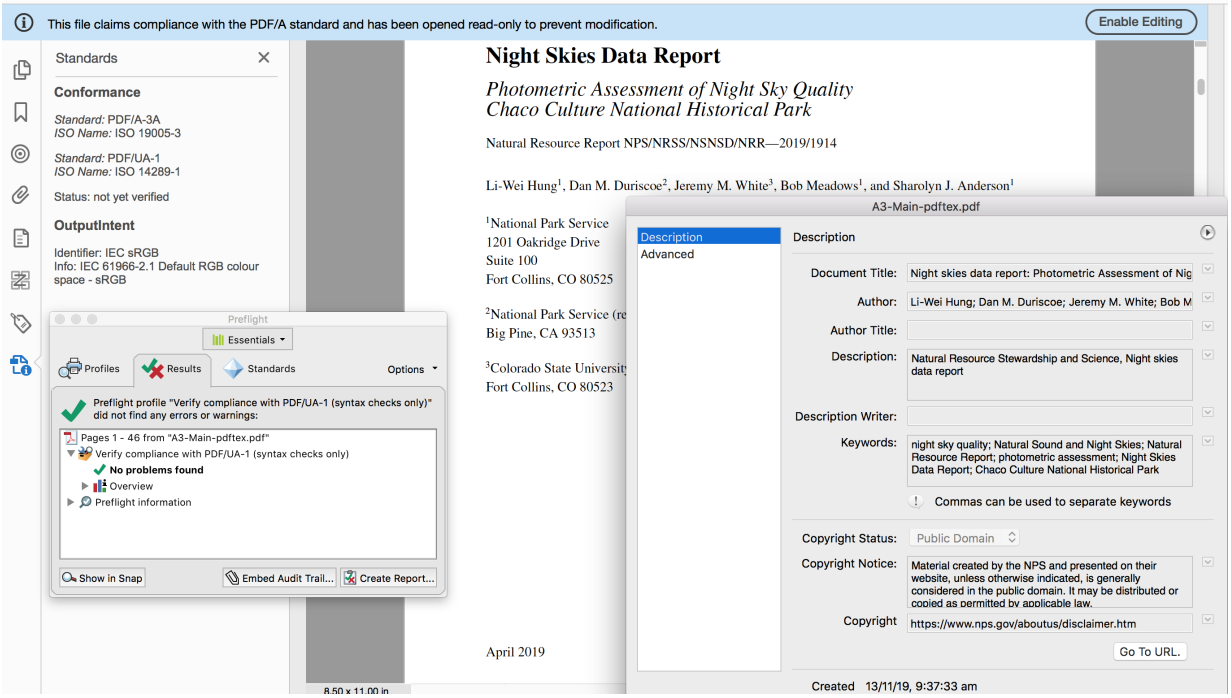
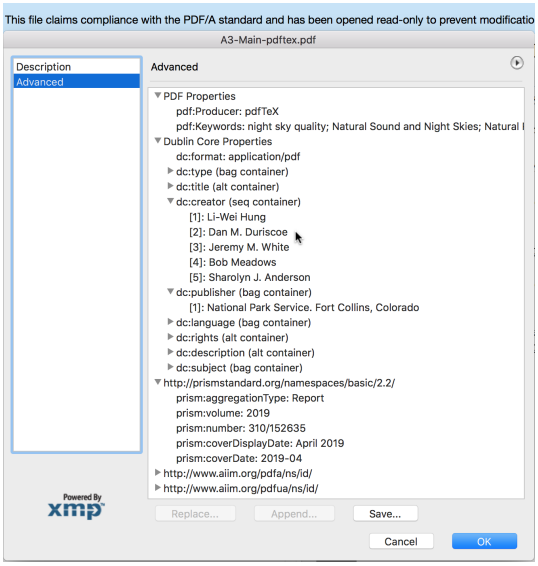


Figure 12: Metadata as shown on the title-page, as well as extra PDF `/Info` entries; conformance for PDF/A-3A and PDF/UA-1 can be checked using Preflight.




```
\providecommand{\pdfxopts}{a-3a,ua-1,pdf17,nocharset}

\begin{filecontents*}{\jobname.xmpdata}
\Title{Night skies data report: Photometric Assessment
 of Night Sky Quality \textemdash\ Chaco Culture
 National Historical Park}
\Author{Li-Wei Hung\sep Dan M. Duriscoe\sep
 Jeremy M. White\sep  Bob Meadows\sep
 Sharolyn J. Anderson}
\Publisher{National Park Service. Fort Collins, Colorado}
\Subject{Natural Resource Stewardship and Science, Night
 skies data report}
\Keywords{night sky quality\sep Natural Sound and Night
 Skies\sep Natural Resource Report\sep photometric
 assessment\sep Night Skies Data Report\sep Chaco Culture
 National Historical Park}
\Copyrighted{False}
\Copyright{Material created by the NPS and presented
 on their website, unless otherwise indicated, is
 generally considered in the public domain. It may be
 distributed or copied as permitted by applicable law.}
\WebStatement{https://www.nps.gov/aboutus/disclaimer.htm}
\PublicationType{Report}
\Journaltitle{\thejournal}
\Volume{\theyear}
\Issue{\theissue}
\CoverDate{\theyear-04}
\CoverDisplayDate{\thedate}
\Creator{pdfTeX + pdfx.sty with \pdfxopts\space option}
\Language{en-us}
\end{filecontents*}
```

Figure 13: Advanced Metadata pane, with LaTeX source that provides information for the XMP packet.

```
% Report title
\begin{nrpsInsideCover}
\nrpsTitle{Night Skies Data Report}\relax\\[-0.1in]%
\nrpsSubtitle{Photometric Assessment of Night Sky Quality \\[2pt]Chaco Culture National Historical Park}%
\nrpsSeries{Natural Resource Report NPS/NRSS/NSNSD/NRR---\hl{\theissueII}}%
\vspace{0.2in}

% Authors
\begin{nrpsAuthors}
\author{Li-Wei Hung\thanks{\nrpsmultidest
 National Park Service\\1201 Oakridge Drive\\Suite 100\\Fort Collins, CO 80525}},
\author{Dan M. Duriscoe\thanks{National Park Service (retired)\\Big Pine, CA 93513}},
\author{Jeremy M. White\thanks{Colorado State University\\Fort Collins, CO 80523}},
\author{Bob Meadows\setcounter{mpfootnote}{0}\mpfootnotemark}, and
\author{Sharolyn J. Anderson\setcounter{mpfootnote}{0}\mpfootnotemark}%
\end{nrpsAuthors}%

\nrpsDate{April 2019}
\begin{nrpsAddress}%
U.S. Department of the Interior\\
National Park Service\\
Natural Resource Stewardship and Science\\
Fort Collins, Colorado%
\end{nrpsAddress}

\end{nrpsInsideCover}
```

Figure 14: LaTeX coding for the title-page, using well-named environments as defined in Figure 16.

`\usepackage[\pdfxopts]{pdfx}`

resulting in the appropriate Metadata to declare the document to be valid for ISO standards PDF/A-3A (ISO 19005-3:2012) [3] and PDF/UA (ISO 14289-1:2012) [4], as can be seen in the left-most panel of Figure 12. These standards are built upon the PDF 1.7 (ISO 32000-1:2008) specification [14]. The claims of validation can be checked, using the Preflight utility [18] of Acrobat Pro DC [17].

It is worth remarking here that use of the pdfx package requires loading hyperref early, to be able to use some of its coding structures for writing directly into the PDF file being built. This can have consequences for the order in which other packages need to be loaded, since hyperref patches many existing LaTeX commands, to include hypertext features. One example of this is discussed and resolved, later in Section 6.1. Doubtless others will occur with other document classes, and the packages used.

Some of the Metadata that is used both on the title-page and within the XMP packet is given using macros, as discussed already in Section 4. With multiple authors and keywords, the `\sep` macro is used as a delimiter; it expands differently in various contexts. For example, the keywords ultimately occur as follows, using RDF syntax [19].

```
<dc:subject><rdf:Bag>
  <rdf:li>night sky quality</rdf:li>
  <rdf:li>Natural Sound and Night Skies</rdf:li>
  <rdf:li>Natural Resource Report</rdf:li>
  <rdf:li>photometric assessment</rdf:li>
  <rdf:li>Night Skies Data Report</rdf:li>
  <rdf:li>Chaco Culture National Historical Park
    </rdf:li></rdf:Bag></dc:subject>
```

If more, or different, Metadata fields are required for a particular class of documents, then a pdfa.xmp template can be edited appropriately. If you have a need to do this, contact the author of this article.

6.1 Authors and footnotes

In many LaTeX document classes a construction like

`\author{ ... \thanks{ ... }}`

provides an author name and affiliation. Normally the argument of `\thanks` is separated from the author and is typeset as a footnote. When provided in the document preamble, each use of `\author` causes the information to be stored away, awaiting use in a command like `\maketitle`. With the research report, the title-page is built separately, after the cover pages. There is no need for storing the author names, but the syntax for providing that information can still be used, for consistency with authoring in other documents.

Figure 14 shows the coding for the title-page, using macros and environments named according to the principles discussed in Section 2, 3 and 4. Those names can be mapped to structure tagging as evident in Figure 15. The visual layout is determined by the macro definitions shown in Figure 16. Of note is the nrpsAuthors environment which uses

a minipage to ensure the correct style and placement of footnotes created from `\thanks` commands. The counter `mpfootnote`, used for these footnotes, avoids interference with those outside the minipage. Also within the nrpsAuthors environment we see that `\author` is defined to just typeset its argument, with `\thanks` becoming just `\footnote`.

Observe how the order of structural tagging need not coincide with the visual order of information on the page. Rather the order for tagging follows the logical order of occurrence in the LaTeX source, with corresponding address following each author. A hyperlink, with anchor text being the raised footnote number, connects to each author's address. This is in accordance with [6, §4.2.7.1 Example C], but without back links as the author and address are adjacent in the tagging structure.

Where different authors share the same address (in this case authors 1, 4 and 5), the anchor text looks the same but there are separate hyperlinks to named destinations `Hfootnote.1`, `Hfootnote.4` and `Hfootnote.5`. These latter two are placed together with the former, using a macro `\nrpsmultidest` which occurs at the beginning of the first `\thanks` (see Figure 14) and is defined as follows.

```
\def\nrpsmultidest{\Hy@raisedlink{%
 \hyper@@anchor{Hfootnote.4}{\relax}%
 \hyper@@anchor{Hfootnote.5}{\relax}}}%
```

Here `\Hy@raisedlink` and `\hyper@@anchor` are internally defined by the hyperref package, and are used when automatically placing hyperlink anchors. For authors 4 and 5 the `\thanks` is replaced by

```
\setcounter{mpfootnote}{0}\mpfootnotemark
```

to get the correct raised number, using a command `\mpfootnotemark` from the footmisc package. This affects just the hyperlink anchor text, as the target destination is generated using yet another counter. Loading the footmisc package is not as easy as it would seem, since it patches the LaTeX command `\@footnotetext`, which has been patched already by hyperref to generate the hyperlinks. Without proper care, one gets messages such as the following, for non-minipage footnotes.

```
pdfTeX warning (dest): name{Hfootnote.6}
has been referenced but does not exist,
replaced by a fixed one
```

This indicates that a hyperlink cannot work as intended. It is essentially the same issue as reported previously [20, 21]. To resolve the incompatibility use the ideas from Sections 2.1 and 2.2. Consider the following coding when loading the package.

```
\makeatletter
 \let\LTX@footnotetext\H@@footnotetext
 \let\HYP@footnotetext\@footnotetext
 \usepackage{footmisc}
 \let\FM@footnotetext\@footnotetext
 \let\H@@footnotetext\FM@footnotetext
 \let\@footnotetext\HYP@footnotetext
\makeatother
```

The resulting expansion for `\@footnotetext` is the same as if footmisc had been loaded first. It works since hyperref created a pointer `\H@@footnotetext` to the expansion of `\@footnotetext` before patching, and uses this within its own patch. What is needed is to make `\H@@footnotetext` point instead to footmisc's version as `\FN@footnotetext`, while still using hyperref's coding which has been captured as `\HYP@footnotetext`. This is what the sequence of three `\let` instances achieves, without defining any new code blocks, and we have a pointer to each defined code block. If we wish to retain pointers to only expansions that will actually be used, then this coding can be reduced by two lines. (How?)

6.2 More front-matter structures

Other pages in the front-matter section of the report have paragraphs, or other structures, that have a special semantic meaning. For example Figure 17 has a few ordinary paragraphs, but also an example citation, and one paragraph with hyperlinks to the NPS website and email address. The whole page can be wrapped in an environment nrpsPolicy that resets the page-numbering and its style, and finishes with the publication date. Although no special formatting is required for the example citation, it is worth tagging this for its semantic meaning.

```
\newenvironment{nrpsPolicy}{%
 \pagenumbering{roman}%
 \setcounter{page}{2}%
 }{\unskip \vfill \nrpsPutDate}

\newenvironment{nrpsCitation}{}{\unskip}
```

Consider again Figure 5. There is a bookmark to the 'Table of Contents' (ToC) page, but no corresponding entry within the ToC itself. Achieve this via a macro `\suppresscontents` which is defined within the following block of coding.

```
\makeatletter
 \def\contentssuppress{%
  \protect\contentsline{chapter}%
  {\contentsname}{\thepage}%
  {\@currentHref}\protected@file@percent}
\makeatother

\let\LTXaddtocontents\addtocontents
\newcommand{\addtocontentssuppressed}[2]{%
 \begingroup \def\thisarg{#2}%
  \ifx\thisarg\contentssuppress\else
```

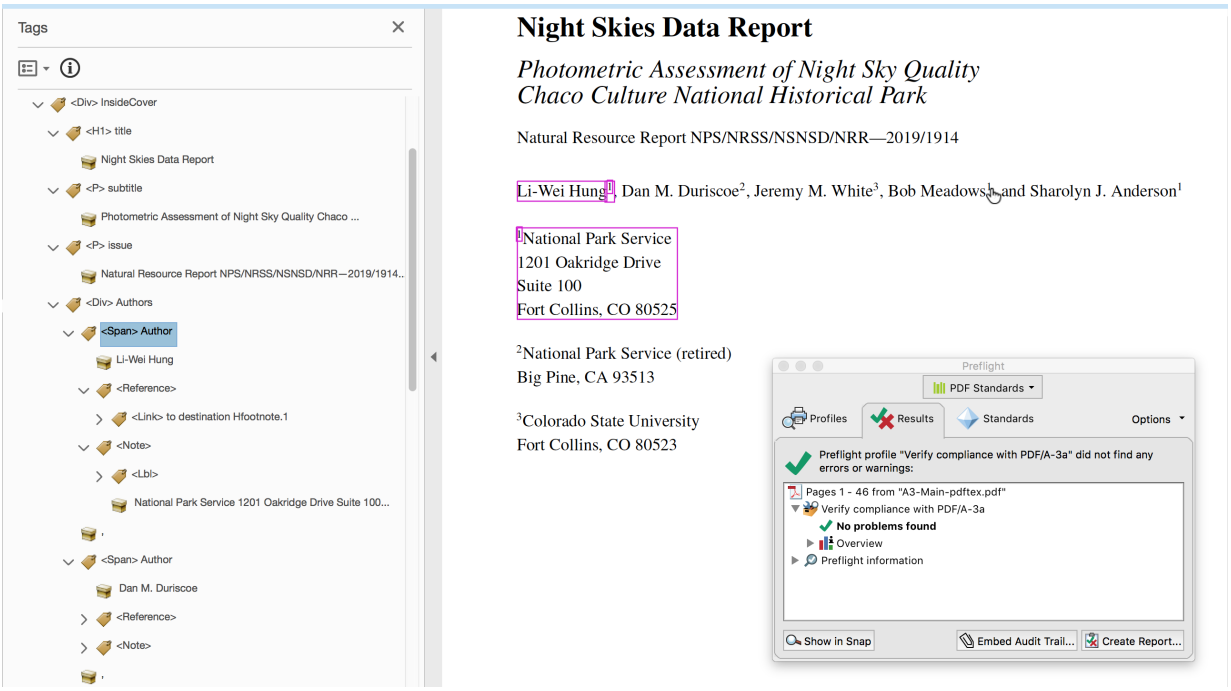


Figure 15: Tagging and layout of author information, produced using the coding shown in Figure 14. The Preflight window shows validation for PDF/A-3a.

```
\newenvironment{nrpsInsideCover}{%
 \vspace{2pt}%
 \rule{\textwidth}{1pt}\\[0.2in]}{}

% Recreations of NRR MS Word styles
\newcommand{\nrpsTitle}[1]{\noindent\fontsize{20}{24}\selectfont\textbf{#1}}
\newcommand{\nrpsSubtitle}[2][{\\[4pt]}]{\fontsize{18}{21}\selectfont\textit{#1#2}}
\newcommand{\nrpsSeries}[2][{\\[12pt]}]{\fontsize{12}{14}\selectfont{#1#2}}

% provide macro as a place-holder for structural tagging
\newcommand\nrpsAuthor[1]{#1}%

\newenvironment{nrpsAuthors}{%
% Do all the work inside a minipage,
% this allows footnotes to come immediately afterwards
% rather than at the bottom of the page.
 \begin{minipage}{\textwidth}%
  % number the footnotes with numerals
  \renewcommand{\thempfootnote}{\arabic{mpfootnote}}%
  \let\footnotesize\normalsize            % write footnotes at normal size
  \def\footnoterule{\vskip-\medskipamount}%  remove the rule above footnotes
  \footnotesep=26pt      % set spacing between footnotes
  \footnotemargin=4pt    % indent the footnote marker, else it overlaps the margin
  \parindent=0pt         % ensure no paragraph indent
  % allow \author  to be used for declaring author names
  \let\author\nrpsAuthor
  \let\thanks\footnote
 }{\end{minipage}}

% set Date and Address, with appropriate space before
\newcommand{\nrpsDate}[1]{\vfill{#1}}
\newenvironment{nrpsAddress}{\\\\}{\unskip}%
```

Figure 16: Macro and environment definitions, for the title-page as coded in Figure 14.

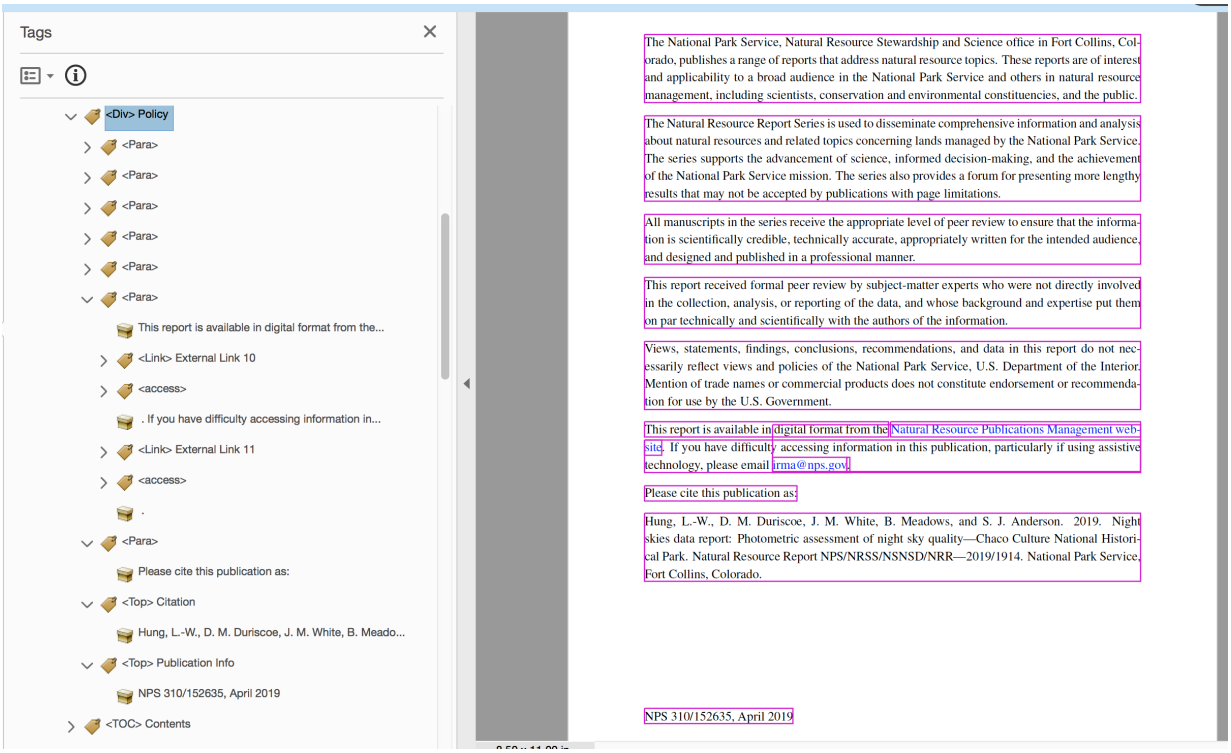


Figure 17: The page following the title-page has a paragraph with external hyperlinks, the publication date, and gives the recommended way to cite the report as a publication.

```
  \LTXaddtocontents{#1}{#2}%
 \fi \endgroup}%  use it locally only

\def\suppresscontents{%
 \let\addtocontents\addtocontentssuppressed}
```

The effect of this is that whenever `\addtocontents` is encountered when writing into the `.aux` file, then the argument is checked to see whether it matches what is produced by the following.

```
\addcontentsline{toc}{chapter}{\contentsname}
```

If so, then writing this ToC entry is skipped, but the `\Bookmark` command is still written into the `.out` file. On the next processing run, the desired result is produced provided the ToC is generated using coding as follows.

```
{% suppress ToC entry, but keep the bookmark
 \suppresscontents
 \maintoc
\addcontentsline{toc}{chapter}{\contentsname}
}\clearpage
```

The tpdf package patches many LaTeX macros including `\tableofcontents`, `\listoffigures` and `\listoftables`. It also patches `\@starttoc` from the tocloft package. This produces tagging in accordance with Recommendation [6, §4.1.4.1, Example B]. This includes having hyperlinks from the section titles to where the (sub-)section starts within the body of the PDF, and tagging of dot leaders as `/Artifact`, as recommended in §4.1.4.3. This can be seen, for example, in Figure 18. Page numbers are *not* hyperlinked, and the word 'Page' above the numbers is also an `/Artifact`.

Note how subsections have a nested `/TOC` structure, child of their enclosing section's `/TOCI`. This tagging is all handled automatically with no extra input required from a document's author. However there is one practical consideration of which authors and editors should be aware, when using tpdf for generating Tagged PDF documents.

The tpdf package maintains a 'tag history' file which, among other things, records the page number on which each structural and content tag occurs; or rather, on which it did occur on the completed LaTeX run. On the next run the history is read, which helps in determining internal aspects of how the tagging is structured for the pages. Also, the page number is compared with what is happening when the same tagging and content is encountered on the subsequent run. A warning message is written when there is a difference. This is similar to

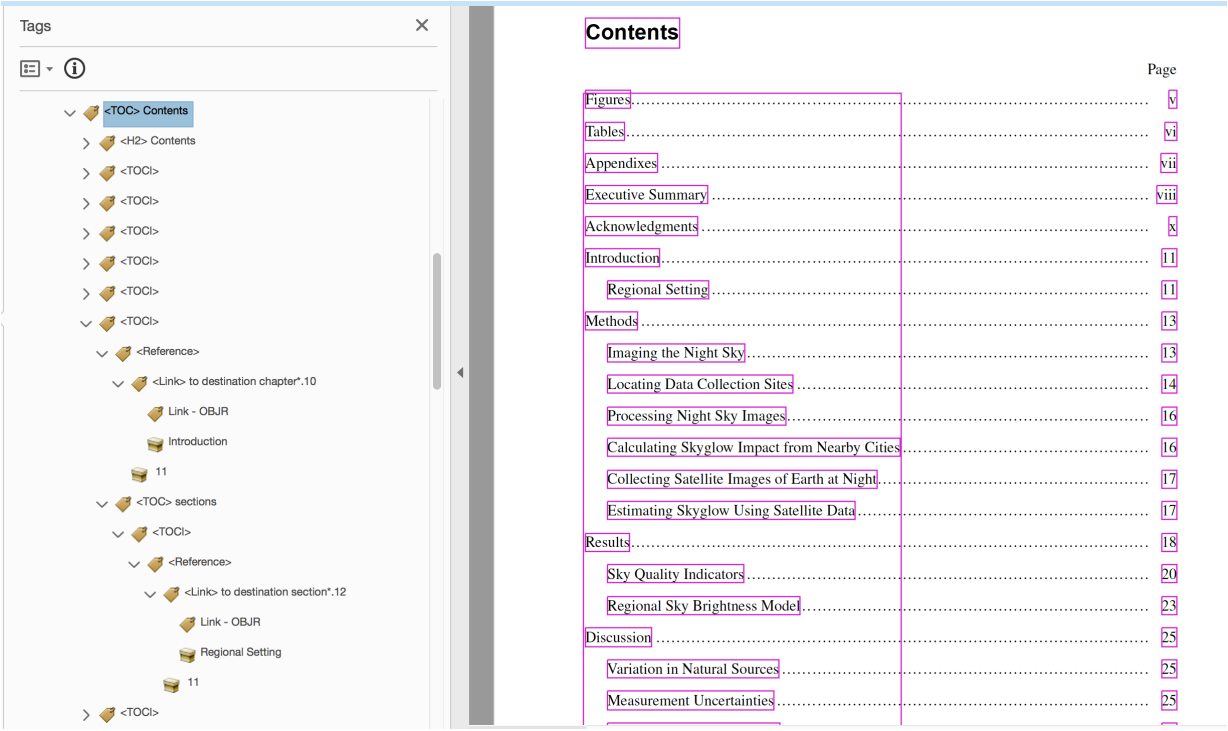


Figure 18: Fully-tagged 'Table of Contents' page, hyperlinked and with nested `/TOC` structure according to section levels.

warnings about missing or changed cross-reference or citation labels in ordinary LaTeX jobs. However the number of structural tags and content snippets is far, far greater.

Now as document source is being written, or edited, it is inevitable that tagging information will change, and differences in the 'history' content encountered. Thus at least one extra LaTeX run will be needed, to ensure that tagging has stabilised. In particular, as extra material is added, the length of the Table of Contents can grow, perhaps requiring an extra page. With such an extra page, there will be changes in the recorded history for all subsequent structure and content. As many as three extra runs may be required before the history has stabilised. Even then it is best to do another run 'just to be sure'. Authors and editors should get used to using the 'Console' window for interactive control of a running TeX job. As well as hitting the `return` key to continue after a pause displaying an error or warning message, one can also use the '`q`' key (followed by `return`) to switch into so-called *quiet* mode. Now there are no further messages for warnings, and the job can proceed to completion without pauses. This is most convenient on the second or third subsequent run after significant edits, when the pauses are due to detected history changes, as these will be updated without need for further edits. In case of real typos, in a macro name say, there is also '`i`' to allow insertion of the intended macro name. This is preferable to '`x`' as it can allow the job to complete without the need for multiple extra runs before stabilisation can be achieved. Just one more run, with the typo corrected, may be sufficient.

Then, of course, when you think the document is complete or you want to check the results of significant editing, use the Preflight utility [18] to check validation (as shown in Figures 12 and 15), and examine the tagging tree within Acrobat Pro DC [17] (as in Figures 1, 3, 6, 9, 10, 15 and 17).

7 pdfLaTeX *vs.* LuaLaTeX

With reports written collaboratively by multiple authors, it is understandable that some may prefer LuaLaTeX to process the document, while another may prefer pdfLaTeX. LuaLaTeX and pdfLaTeX are built upon the LuaTeX and pdfTeX 'engines' respectively, each loading the LaTeX format. Mostly the same (at least visual) output is able to be produced, but there can be significant differences in configuration options that are needed to actually achieve this. Here we discuss such differences relevant to the 'Night Skies' research report.

7.1 Font formats

The research report [8] style is an adaptation for LaTeX following styles developed for Microsoft Word documents. As such it uses fonts 'Times' and 'Arial' in several sizes and faces. LuaTeX is able to work directly with both OpenType (`.otf`) and TrueType (`.ttf`) font formats. OpenType supports a large number of characters including accented letters for different languages, as well as mathematical symbols and much more. On the other hand pdfTeX (generally) addresses at most 256 characters in a font, and can use the TrueType (`.ttf`) font format. The winfonts package [22] provides the support needed to use the Windows' TrueType fonts with pdfTeX, but it does not supply the fonts themselves. These are presumed to be available already on a Windows (or other) system. Thus for the research report, there is coding in the 'settings' file read from the LaTeX preamble as follows.

```
\usepackage{ifluatex}
\ifluatex
 \usepackage{fontspec}
\fi

\ifluatex
%This is Overleaf Specific (or if the fonts
% are not installed in your system)
%-----------------------------------------------
% Times New Roman
\setromanfont[
BoldFont=Font-timesbd.ttf,
ItalicFont=Font-timesi.ttf,
BoldItalicFont=Font-timesbi.ttf,
]{Font-times.ttf}

% Arial
\setsansfont[
BoldFont=Font-arialbd.ttf,
ItalicFont=Font-ariali.ttf,
BoldItalicFont=Font-arialbi.ttf
]{Font-arial.ttf}

%% those TTF fonts do not validate for PDF/A :
%%  incomplete CIDSet
\else
% but all is well with pdfTeX and corresponding
%  Type-1 fonts
 \usepackage[T1]{fontenc}
 \usepackage{times}%  actually  NimbusRomNo9L
 \usepackage{winfonts}
 \renewcommand{\sfdefault}{arial}
\fi
```

This assumes that fonts named Font-times.ttf and Font-arial.ttf are available on the local system when using LuaTeX, at least via the Overleaf online system [26]. Using other systems the font file names will be different. With pdfTeX as the typesetting engine, this coding assumes packages times and winfonts are available. Packages ifluatex and fontspec come with TeXLive [25] distributions and times also, as part of LaTeX's NFSS support, whereas winfonts does not.

There is, however, a small complication with winfonts, in the form of a mistake in the virtual font for '*Arial Bold Italic*', which is used for sub-section headings. The virtual font for this actually refers to '*Arial Bold*', omitting the '*Italic*' part. A fix for this is available trough a package fix-winf [23].

Alternatively, there is a package urw-arial [24] which provides Type 1 fonts based on the 'Arial' glyph shapes, provided freely by URW. But there are slight differences in glyph metrics between these and Microsoft's own TrueType fonts for 'Arial'. Furthermore, the package winfonts also loads the textcomp package, whose utility is discussed within the next sub-section.

7.2 Inline mathematics

There is one feature in LuaTeX that is different to all other TeX engines, with regard to mathematical symbols, in particular for an inline mathematical expression. A macro token like `\pm` normally only works smoothly in so-called 'math-mode', otherwise there is an error message in the Console window.

```
Missing $ inserted.
<inserted text>
                $
1.40 token like \pm
                    normally only works
?
```

Furthermore, TeX has switched into math-mode for the benefit of further mathematical symbols that may be following. The reason for this behaviour is that spacing between letters and symbols tends to be different in math-mode to with ordinary textual input into paragraphs.

On the other hand, with LuaTeX it is perfectly OK to use the '±' *character* directly in the input, as it lies within the range of characters supported by most 8-bit fonts. This behaviour, which is similar to what one might expect from other word- or text-processing software, can be very convenient where the math-expressions are short and uncomplicated. It does however bypass the very fine-tuning of spacing that otherwise results from using math-mode.

Provided the textcomp package is loaded this approach also works with pdfTeX, otherwise there is an error message as shown in Figure 19. The textcomp package sets up a mapping which produces `\IeC{\textpm }` upon reading the special character. Here `\textpm` specifies the desired character from a re-encoded (TS1) version of the current text font.

```
./testinput.tex:53: Package inputenc Error: Unicode character ± (U+00B1)
(inputenc)                not set up for use with LaTeX.

See the inputenc package documentation for explanation.
Type  H <return>  for immediate help.
 ...

l.53 pdf\TeX\ and the ±
                       character used directly on
?
```

Figure 19: Error message, using pdfTEX, resulting from input of a UTF-8 mathematical symbol, unless the textcomp package has been loaded.

Now when it comes to accessibility, it's not at all clear how a screen-reader will handle the resulting '±' character. So we add an extra layer which specifies *alternative text* for that symbol, and uses proper math-mode where appropriate. For example, the research report's preamble has coding as follows.

```
\ifluatex
\else
% account for math chars in text
 \let\LTXarcdeg\arcdeg
 \let\LTXcdot\cdot
 \let\LTXsim\sim
 \let\LTXtextpm\textpm

 \def\NRRarcdeg{\TPDFensuremath{degree}
   {\LTXarcdeg}}
 \def\NRRcdot{\TPDFensuremath{times}{\LTXcdot}}
 \def\NRRsim{\TPDFensuremath{approximately}
   {\LTXsim}}
 \def\NRRtextpm{\TPDFensuremath{plus or minus}
   {\pm}}

\AtBeginDocument{%
 \let\arcdeg\NRRarcdeg
 \let\cdot\NRRcdot
 \let\sim\NRRsim
 \let\textpm\NRRtextpm
}
\fi  %  end of  \ifluatex
```

As a default expansion, we have that

```
 \def\TPDFensuremath #1#2{\ensuremath{#2}}
```

where TEX's \ensuremath handles switching both into math-mode and back out again. With the tpdf package being used to generate Tagged PDF, the adapted command \TPDFensuremath does more by also establishing the contents of the #1 parameter as 'alternative text'. Notice how pointers have been used to capture the expansion of an existing LATEX macro, then to change the name so as to point to an enhanced code-block, as was discussed earlier in Section 2.2.

Throughout the research report such commands are used for measurements and units; *viz.*

```
...in units of W\cdot cm$^{-2}$\cdot sr$^{-1}$
...angular sensitivity is \sim 42$^\circ$.
...darker than \sim 21.5 mag/arcsec$^2$.
...brightness measurement is ±4\%.
...accurate to ±5\%, or 0.05 magnitudes.
```


Acknowledgements

Great thanks go to members of the U. S. National Parks Service science team based at Fort Collins, Colorado:

- Kurt M. Fristrup (Senior Scientist), Natural Sounds and Night Skies Division;
- Chalmers-Fagan Johnson (Web and Report Specialist);
- Li-Wei Hung (Night Skies Research Scientist), main author of the report [8];
- Damon Joyce (Night Skies Research Scientist);
- Dan M. Duriscoe, Jeremy M. White, Robert Meadows, Sharolyn J. Anderson; authors of the report [8].

for allowing use of their report as a development document for Tagged PDF using LATEX.

Thanks also to Thomas E. Price, University of Akron (emeritus), for reading an early draft and providing useful comments and suggestions.

References

[1] United States Access Board; Section 508 ICT Refresh (January 2017). §E205.4 Electronic Content. https://www.access-board.gov/guidelines-and-standards/communications-and-it/about-the-ict-refresh/final-rule/text-of-the-standards-and-guidelines#504-authoring-tools.

[2] National Science Foundation — FAQ; What is NSF's public access policy? https://www.nsf.gov/pubs/2016/nsf16009/nsf16009.jsp#q1.

[3] ISO 19005-3:2012; Document Management — Electronic document file format for long term preservation — Part 3: Use of ISO 32000-1 with

support for embedded files (PDF/A-3). https://www.iso.org/standard/57229.html.

[4] ISO 14289-1:2012; Document management applications – Electronic document file format enhancement for accessibility – Part 1: Use of ISO 32000-1 (PDF/UA-1). Technical Committee ISO/TC 171/SC 2 http://www.iso.org/iso/catalogue_detail.htm?csnumber=64599.

[5] WCAG 2.1; 'Web Content Accessibility Guidelines'. W3C Recommendation 05 June 2018. W3C Consortium. https://www.w3.org/TR/WCAG21/.

[6] 'Tagged PDF Best Practice Guide: Syntax.' PDF Association. For developers implementing ISO 14289-1 (PDF/UA). Version 1.0 (June 2019). https://www.pdfa.org/resource/tagged-pdf-best-practice-guide-syntax/

[7] Moore, R.; 'LaTeX 508 — creating accessible PDFs'. Conference talk presented at TUG 2019 in Palo Alto, July 2019, delivered remotely via Zoom. Movies and other related material can be found at http://web.science.mq.edu.au/~ross/TaggedPDF/TUG2019-movies/.

[8] Hung, L.-W., D. M. Duriscoe, J. M. White, B. Meadows, and S. J. Anderson. 2019. Night skies data report: Photometric assessment of night sky quality – Chaco Culture National Historical Park. Natural Resource Report NPS/NRSS/NSNSD/NRR – 2019/1914. National Park Service, Fort Collins, Colorado. https://irma.nps.gov/DataStore/Reference/Profile/2260171.

[9] Natural Resource Stewardship and Science Directorate. National Park Service. https://www.nps.gov/orgs/1778/index.htm.

[10] 'History of the NPS Arrowhead'. National Park Service. https://www.nps.gov/glac/learn/news/history-of-the-nps-arrowhead.htm.

[11] Web AIM; Web accessibility in mind: 'Alternative Text' page. https://webaim.org/techniques/alttext/.

[12] Accessible U: 'Alt Text'. University of Minnesota web site. https://accessibility.umn.edu/core-skills/alt-text.

[13] Adobe Systems Inc.; Extensible Metadata Platform (XMP). ISO standard (16684-1). https://www.adobe.com/products/xmp.html.

[14] Adobe Systems Inc.; PDF Reference 1.7, November 2006. Also available as [15]. http://www.adobe.com/devnet/pdf/pdf_reference.html.

[15] ISO 32000-1:2008; Document management — Portable document format (PDF 1.7); Technical Committee ISO/TC 171/SC 2 (July 2008). http://www.iso.org/iso/catalogue_detail?csnumber=51502.

[16] ISO 32000-2:2017; Document management — Portable document format — Part 2: PDF 2.0 Technical Committee ISO/TC 171/SC 2 (July 2017). https://www.iso.org/standard/63534.html.

[17] Adobe Systems Inc.; Acrobat Pro DC. The complete PDF solution for any device. (Windows and Mac. only) https://acrobat.adobe.com/au/en/acrobat.html.

[18] Adobe Systems Inc.; 'Analyzing documents with the Preflight tool (Acrobat Pro)'. https://helpx.adobe.com/au/acrobat/using/analyzing-documents-preflight-tool-acrobat.html.

[19] W3C; Resource Description Framework (RDF). W3C Recommendation 25 February 2014. RDF 1.1 Concepts and Abstract Syntax. https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/ RDF 1.1 XML Syntax. https://www.w3.org/TR/rdf-syntax-grammar/.

[20] TeX StackExchange; 'Package footmisc causes pdfTeX error'. question asked October 2015. link

[21] TeX StackExchange; 'Footnotes misbehaving in report go to front page but behave correctly in other report'. question asked September 2014. link

[22] CTAN archive: winfonts — a package to use the basic Windows TrueType fonts. Package and documentation by Paul Pichaureau, Paris, January 2006. Comprehensize TeX Archive Network. https://ctan.org/pkg/winfonts.

[23] CTAN archive: fix-winf — fixes for Windows font usage with pdfLaTeX. Package and documentation by Ross Moore, December 2019. (to appear) Comprehensize TeX Archive Network. https://ctan.org/pkg/fix-winf.

[24] CTAN archive: urw-arial — URW Arial font pack for use with LaTeX. Package and documentation by Walter Schmidt, March 2006. https://ctan.org/pkg/urw-arial.

[25] TeX User's Group; TeX Live — document production system. Distributed by TUG for TeX user groups worldwide, since 1996. Updated annually. https://tug.org/texlive/.

[26] Overleaf, Online LaTeX Editor. https://www.overleaf.com/about.