MAT188 LEARNING STANDARD LaTeX PACKAGES DOCUMENTATION

SAMUEL P. DUMAS, CAMELIA KARIMIANPOUR

Contents

1. Overview

The following three packages were written for the course **MAT188 - Linear Algebra for Engineering** at the **University of Toronto** during the Summer of **2023**. The primary purpose of this library is to facilitate the creation of standard-based grading material for mathematics courses. Standard-based grading is an approach to assessing students' work, where marks are earned by demonstrating the required learning standards. The package `mathbib.sty` enables the user to manage relevant concepts to the course (e.g., definitions, theorems) in a similar way to a `.bib` file. The next package, `learningstandards.sty`, is utilized for learning-standard management. The package `mathbib.sty` is automatically called within `learningstandards.sty`, and the two packages are compatible. Lastly, the package `questionbank.sty` allows the user to store questions and their learning standards, rubrics, and solutions. Thereafter, the stored questions can be easily called to form worksheets.

1.1. Required Files.

Packages (Included in Overleaf)

- `expl3`
- `geometry`
- `xparse`
- `multicol`
- `graphicx`
- `tikz`
- `hyperref`
- `forloop`
- `tcolorbox`
- `xcolor`

Custom Packages

- `mathbib.sty`
- `learningstandards.sty`
- `questionbank.sty`

TeX Files (To create for storage)

- `concepts.tex`
- `standards.tex`
- `questions.tex`

2. mathbib.sty

2.1. Package Functions.

2.1.1. *\SaveConcept.*

```
\begin{SaveConcept}{<type>}[
        key = <reference key>,
        title = <concept title>
    ][<chapter>]
    <description>
\end{SaveConcept}
```

This environment is used to save desired concepts for future referencing. Moreover, it is quintessential that a concept be saved first before any subsequent functions can be utilized within this package. Concepts should be saved in a separate TEX file titled `concepts.tex`. The type of concept is indicated in the `<type>` parameter, with either `definition` or `theorem` being recognized as valid concept types. To access the metadata of a saved concept, an identification key is set in `<reference key>` and a title to the concept is set in `<concept title>`. Additionally, if the concept is from a particular textbook chapter, this can be specified in `<chapter>`, though this is optional. Lastly, a description of the concept can be written as plain body text (including mathematics mode) for `<description>`. Included below is an example where the definition of a real-valued column vector.

```
\begin{SaveConcept}{definition}[
        key = cvec,
        title = {Column Vector}
    ][0]
    A real \textbf{column vector} is a $n \times 1$ matrix $\begin{bmatrix} v_1\\
    \vdots \\v_n \end{bmatrix}$, where $v_i \in \mathbb{R}$, $1 \leq i \leq n$.
\end{SaveConcept}
```

2.1.2. *\MathCite.*

```
\MathCite{<key>}[<title>]
```

This command is used to cite a previously saved concept using the `SaveConcept` environment. The most simple syntax of this command requires only the `<key>` of the desired concept to be entered, such as `\MathCite{<key>}`. However, by default `\MathCite` uses the key of the concept, which produces **cvec** for the column vector concept saved above. Using the optional parameter `[<title>`, we can specify what text should be displayed instead of the `<key>`. Thus, `\MathCite{cvec}[column vector]` would produce a citation for **column vector**. Lastly, if you want a concept to appear in the bibliography without citing the concept in-text, the addition of an asterisk `\MathCite*{<key>}` achieves this.

2.1.3. *\PrintSavedRender.*

```
\PrintSavedRender
```

This command is used before the end of the document to print out the bibliography. By default, the bibliography printed using `\PrintSavedRender` begins on a new page, however, using `\PrintSavedRender*` negates this setting. At the end of the section, we use `\PrintSavedRender*` to load the bibliography.

```
\begin{document}
...
\PrintSavedRender
\end{document}
```

```
\begin{document}
...
\PrintSavedRender*
\end{document}
```

2.1.4. *\PullConcept.*

```
\PullConcept{<key>}
```

This command is used to include a printout of a saved concept. The concept accessed using `<key>` will be printed using `\PullDef{<key>}` or `\PullThm{<key>}` depending on the type of the concept. The next two functions detail both outcomes and further describe what occurs within the \PullConcept command.

2.1.5. *\PullDef.*

```
\PullDef{<key>}
```

This command is used to include a printout of a definition type concept. The definition will be printed within a teal tcolorbox, though the colour can be changed in the mathbib.sty file by modifying the \definecolor{188teal}{cmyk}{<cyan>,<magenta>,<yellow>,<key/black>} on line 78. Included below is the printout \PullDef{cvec} for the **column vector** definition created earlier.
NOTE: This command will printout theorem type concepts as a definition.

Definition (Column Vector)

A real **column vector** is a $n \times 1$ matrix $\begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}$, where $v_i \in \mathbb{R}$, $1 \leq i \leq n$.

2.1.6. *\PullThm.*

```
\PullThm{<key>}
```

This command is used to include a printout of a theorem type concept. The theorem will be printed within a teal tcolorbox, though the colour can be changed in the mathbib.sty file by modifying the \definecolor{188orange}{cmyk}{<cyan>,<magenta>,<yellow>,<key/black>} on line 79. Included below is the printout \PullThm{ranknull} for the **Rank-Nullity Theorem**.
NOTE: This command will printout definition type concepts as a theorem.

Theorem (Rank-Nullity Theorem)

Let A be a $m \times n$ matrix. Then the sum between the rank of A and the null of A is equal to n, the number of columns.

$$\text{rank}(A) + \text{null}(A) = n$$

2.2. **Saving Concepts.** To store concepts within this package, it is best practice to create a separate file labelled concepts.tex. Within this file, use the SaveConcept environment to save all the necessary concepts. Now, in the document you wish to write (i.e., main document), input the file using the command \input{concepts.tex}. Notably, this assumes that concepts.tex is within the same repository as the main document, and thus the repository of the concepts.tex file may need specification within the \input command. Following the completion of these steps, every feature from the package mathbib.sty will be functional.

```
\documentclass{amsart}
\usepackage{mathbib}
...
\input{<file directory>/concepts.tex}
...
\begin{document}
```

```
\end{document}
```

2.3. Definitions.

Definition (Column Vector). *A real* ***column vector*** *is a* $n \times 1$ *matrix* $\begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}$, *where* $v_i \in \mathbb{R}$, $1 \leq i \leq n$.

Theorem (Rank-Nullity Theorem). *Let* A *be a* $m \times n$ *matrix. Then the sum between the rank of* A *and the null of* A *is equal to* n*, the number of columns.*

$$rank(A) + null(A) = n$$

3. LEARNINGSTANDS.STY

3.1. Package Functions.

3.1.1. *\BuildLSKey.*

```
\begin{BuildLSKey}[
    key = <reference key>,
    stan = <learning standard>
][<PCE>]
\end{BuildLSKey}
```

This is environment is used to build and store a learning standard. All subsequent commands require the desired learning standard to be built using the `BuildLSKey` environment. Learning standards should be saved in a separate TEXfile title `standards.tex`. An identification key is set in `<reference key>` and the learning standard is set in `<learning standard>`.

There is a specific syntax that should be used when assigning the `<reference key>` parameter in the `BuildLSKey` environment. Identification keys consists of three components: chapter number, learning standard type, and a sub-key.

```
ch<chapter number>-<learning standard type>-<sub-key>
```

Within this library, there exists four types of learning standards: computational (COM), conceptual (CON), visual/geometry (VG), and writing (WRIT). The logistics behind this division in further explain in . Below, we provide example code where we save two computational learning standard concerned with determining when two vectors are parallel and when two vectors are perpendicular. Additionally, we use the `\MathCite` command from `mathbib.sty` within the code to cite two new definitions `parallel` and `perpendicular`.

```
\begin{BuildLSKey}[
    key=ch0-COM-par,
    stan={I can decide whether given vectors are \MathCite{parallel}.}
][1]
\end{BuildLSKey}

\begin{BuildLSKey}[
    key=ch0-COM-per,
    stan={I can decide whether given vectors are \MathCite{perpendicular}.}
][2]
```

```
\end{BuildLSKey}
```

Lastly, building a learning standard within any chapter requires the previous chapter to have an existing learning standard (i.e., a chapter 3 learning standard cannot be saved without the creation of chapter 2 learning standard). However, this can be voided by including an asterisk following the `\begin{BuildLSKey}` in the `BuildLSKey` environment. In this situation, the assigned `<reference key>` should take the form `ch<chapter number>-void` and the remaining parameters should be left blank. Below is an example where chapter 1 does not include any learning standards.

```
\begin{BuildLSKey}*[
    key=ch1-void,
    stan={ }
]
\end{BuildLSKey}
```

3.1.2. *\PullLS.*

```
\PullLS[<key list>]
```

This command is used to printout a list of learning standards. The input for `<key list>` can be as simple as a singular learning standard key, and at most contain nine learning standard keys. Keys should be comma separated without any extra spaces added between commas. For example, using `\PullLS[ch0-COM-par,ch0-COM-per]` to printout the learning standards saved in the previous section yields the following.

LS* I can decide whether given vectors are **parallel**. (*Computation*)

LS† I can decide whether given vectors are **perpendicular**. (*Computation*)

The reason this command is restricted to nine learning standards relates to the use of `\fnsymbol` to mark each `LS` bullet, which is only defined for integers one through nine.

$$\begin{array}{ccc} 1 \equiv * & 2 \equiv \dagger & 3 \equiv \ddagger \\ 4 \equiv \S & 5 \equiv \P & 6 \equiv \| \\ 7 \equiv ** & 8 \equiv \dagger\dagger & 9 \equiv \ddagger\ddagger \end{array}$$

Moreover, learning standards are meant to isolate specific goals within a question, and thus utilizing more than nine learning standards within a question would be counter-intuitive.

Additionally, specific aspects of a question associated with a learning standard can be labelled using the `\PullLS*[<integer>`. This command produces a print in superscript of the form `LS\fn{symbol}`. For example, it may be the case that this sentence pertains to the second learning standard$^{LS^\dagger}$, and this one the third$^{LS^\ddagger}$. In our experience, this command was primarily used within the solutions of questions to emphasize to students where a learning standard is present within the question.

3.1.3. *\PulledLS.*

```
\PulledLS{<key>}
```

This command is used to print a learning standard. Conversely to the command `\PullLS`, this command prints a singular learning standard in in-line text, rather than a custom `itemize`. Using `\PulledLS{ch0-COM-par}` prints the learning standard `ch0-COM-par` in plain text, such as, I can decide whether given vectors are **parallel**., for easy referencing.

3.1.4. *\PullChap.*

`\PullChap{ngoals<chapter number>}`

This command is used to print every learning standard within the chapter specified in `<chapter number>`. Recall that the chapter of a learning standard is determined by its `<key>` in the initial section `ch<chapter num>-...`. Standards are printed out using an `enumerate` environment. Currently, the only two learning standards we have saved in Chapter 0 are `ch0-COM-par` and `ch0-COM-per`. Thus, using `\PullChap{ngoals0}` should produce an `enumerate` with prints of both those standards, as seen below.

Chapter 0 Learning Standards

(1) I can decide whether given vectors are **parallel**. (*Computation*)
(2) I can decide whether given vectors are **perpendicular**. (*Computation*)

3.1.5. *\PullPCE.*

`\PullPCE{<PCE number>}`

This command is used to print every learning standard within the PCE section specified in `<PCE number>`. Recall that the chapter of a learning standard is determined by its input for the `<PCE>` parameter. Standards are printed out using an `enumerate` environment. Briefly, we define two new standards, `ch1-CON-rref` and `ch1-COM-rref`, which are included in the second PCE section, as is `ch0-COM-per`. Thus, using `\PullPCE{2}` should produce an `enumerate` of those prints exclusively, as seen below.

PCE 2 Learning Standards

(1) I can decide whether given vectors are **perpendicular**.
(2) I can perform row reduction on any matrix and reduce it to **REF** and **RREF**.
(3) I can determine when a matrix is in **REF** or **RRFF** form.

3.1.6. *\AllChap.*

`\AllChap`

This command is used to print every learning standard saved within the file `standards.tex`. Standards are printed out using an `enumerate` environment.

All Learning Standards

(1) I can decide whether given vectors are **parallel**. (*Computation*)
(2) I can decide whether given vectors are **perpendicular**. (*Computation*)
(3) I can perform row reduction on any matrix and reduce it to **REF** and **RREF**. (*Computation*)
(4) I can determine when a matrix is in **REF** or **RRFF** form. (*Conceptual*)
(5) I can correctly use mathematical notation. (*Writing*)

3.2. **Saving Learning Standards.** To store learning standards within this package, it is best practice to create a separate file labelled `standards.tex`. Within this file, use the `BuildLSKey` environment to save all the necessary learning standards. Now, in the document you wish to write (i.e., main document), input the file using the command `\input{standards.tex}`. Notably, this assumes that `standards.tex` is within the same repository as the main document, and thus the repository of the `standards.tex` file may need specification within the `\input` command.

Following the completion of these steps, every feature from the package `learningstandards.sty` will be functional. If you are using the package `learningstandards.sty` alongside `mathbib.sty`, it is quintessential that you input the `concepts.tex` file before the `standards.tex` file,

```
\documentclass{amsart}
\usepackage{learningstandards.sty}
...
\input{<file directory>concepts.tex} %IF USING MATHBIB
\input{<file directory>standards.tex}
...
\begin{document}
\end{document}
```

3.3. Definitions.

Definition (Column Vector)**.** *A real **column vector** is a* $n \times 1$ *matrix* $\begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}$, *where* $v_i \in \mathbb{R}$, $1 \leq i \leq n$.

Theorem (Rank-Nullity Theorem)**.** *Let* A *be a* $m \times n$ *matrix. Then the sum between the rank of* A *and the null of* A *is equal to* n*, the number of columns.*

$$\mathit{rank}(A) + \mathit{null}(A) = n$$

Definition (Parallel Vectors)**.** *We say two vectors* $\vec{v}$ *and* $\vec{w}$ *are **parallel** if one is a scalar multiple of the other. That is if* $\vec{v} = k\vec{w}$ *for some* $k \in \mathbb{R}$ *or* $\vec{w} = k\vec{v}$ *for some* $k \in \mathbb{R}$.

Definition (Perpendicular)**.** *We say two vectors* $\vec{v}$ *and* $\vec{w}$ *are **perpendicular** or **orthogonal** if* $\vec{v} \cdot \vec{w} = 0$.

Definition (REF and RREF)**.** *A matrix is in **row echelon form** (REF) if it has the following properties*

(1) All zero rows are in the bottom.
(2) The leading entry in each row is to the right of the leading entry of the row above.
(3) All entries below a leading entry are zero.

*A matrix is in **reduced row echelon form** (RREF) if it is in REF form and in addition*

4. All the leading entries are 1*. In this case, we call them "leading ones".*
5. Each leading 1 *is the only nonzero entry in its column.*

4. QUESTIONBANK.STY

4.1. Package Functions.

4.2. Saving Learning Standards.