The gravy package

Dylan Yang

Contents

1 Introduction

The gravy package is a collection of my personal LaTeX styles and commands.

2 Usage

Typically, gravy should be loaded as a package (i.e. with `\usepackage`) after specifying a KOMA-Script document class. Note that gravy currently only supports pdfTeX.

For convenience, the gravyartcl and gravyreprt document classes are also available. These classes simply load the corresponding KOMA-Script document class and the gravy package.

gravy has two options. The `nodate` option disables the printing of the date when using `\maketitle`. The `minted` option loads custom styles for use with the minted package. Options can be passed to the gravy package or document classes.

3 Typography

3.1 Fonts

The serif font is Linux Libertine, while the sans serif font is Linux Biolinum. The monospace font is `Fira Mono`. The math font is the libertine math font provided by the newtxmath package.

Here is a demonstration of some of the supported ligatures and kerning.

ff fi fl ffi ffl

ff fi fl ffi ffl

AV WA Ay Kw Te fo w.

Note that additional ligatures such as 'Th' and 'Qu' are not currently supported given the use of `T1` font encoding; see this question on the TeX StackExchange. Additionally, the current monospace font, Fira Mono, does not support certain characters such as '\' when using `OT1` encoding instead. Future versions may add an option for XeLaTeX or LuaLaTeX compatibility, which support OpenType fonts and should avoid these issues.

3.2 Line Width

One lowercase alphabet is 133.37076pt in width; the line width is 350.295pt. This should follow the general typographic advice that the line width should be about two to three lowercase alphabets.

3.3 Fonts and Styles

- The quick brown fox jumps over the lazy dog
- **The quick brown fox jumps over the lazy dog**
- *The quick brown fox jumps over the lazy dog*
- ***The quick brown fox jumps over the lazy dog***
- The quick brown fox jumps over the lazy dog
- **The quick brown fox jumps over the lazy dog**
- *The quick brown fox jumps over the lazy dog*
- ***The quick brown fox jumps over the lazy dog***
- The quick brown fox jumps over the lazy dog
- **The quick brown fox jumps over the lazy dog**
- *The quick brown fox jumps over the lazy dog*
- ***The quick brown fox jumps over the lazy dog***
- `The quick brown fox jumps over the lazy dog`
- **`The quick brown fox jumps over the lazy dog`**
- *`The quick brown fox jumps over the lazy dog`*
- ***`The quick brown fox jumps over the lazy dog`***

4 Commands

4.1 Delimiter Sizing

`\auto` `\auto [⟨opening delimiter⟩⟨closing delimiter⟩] ⟨delimited expression⟩`

When typesetting math equations, it is common to use `\left` and `\right` to automatically size delimiters, but this can become cumbersome in more complicated use cases. `\auto` automatically detects the next pair of delimiters and sizes them appropriately. For instance, `\auto(\frac{1}{2})` produces

$$\left(\frac{1}{2}\right).$$

The delimiters that can be automatically detected are:

- `(` and `)`
- `\lparen` and `\rparen`
- `[` and `]`
- `\lbrack` and `\rbrack`
- `\{` and `\}`
- `\lbrace` and `\rbrace`
- `<` and `>`
- `\langle` and `\rangle`
- `|` and `|`
- `\vert` and `\vert`
- `\lvert` and `\rvert`
- `\|` and `\|`
- `\Vert` and `\Vert`
- `\lVert` and `\rVert`

Note that under the hood, `\auto` only looks for the next supported opening delimiter and assumes that the closing delimiter based on that, so it only supports matching pairs of delimiters. For instance, `\auto.\dv{f}{t}\rvert_{t=0}` does not work. Instead, to handle arbitrary pairs of delimiters, `\auto` can also take one optional argument, which should be two tokens representing the opening and closing delimiters. `\auto[.\rvert].\dv{f}{t}\rvert_{t=0}` works instead, producing

$$\left.\frac{\mathrm{d}f}{\mathrm{d}t}\right|_{t=0}.$$

4.2 Derivatives

`\dd` `\dd {⟨variable⟩}`

The `\dd` command is used to typeset the differential operator. It produces an upright 'd' with math operator spacing preceding it, as shown in the example below:

$$\int x \, \mathrm{d}x.$$

`\deriv` `\deriv * [⟨power⟩] {⟨numerator⟩} {⟨denominator⟩}`
`\dv` `\dv * [⟨power⟩] {⟨numerator⟩} {⟨denominator⟩}`
`\pderiv` `\pderiv * [⟨power⟩] {⟨numerator⟩} {⟨denominator⟩}`
`\pdv` `\pdv * [⟨power⟩] {⟨numerator⟩} {⟨denominator⟩}`

Derivatives can be typeset using the `\deriv` and `\dv` commands (which are equivalent), while partial derivatives can be typeset using the `\pderiv` and `\pdv` commands (which are also equivalent). They take two mandatory arguments, the ⟨*numerator*⟩ and ⟨*denominator*⟩, as well as an optional argument ⟨*power*⟩ which specifies the power to which the derivative is raised and an optional star to produce an inline fraction instead of a display fraction.

4.3 Semantic Delimiters

`\abs` `\abs * [⟨size⟩] {⟨expression⟩}`
`\norm`
`\set`
`\floor`
`\ceil`

`\abs`, `\norm`, `\set`, `\floor`, and `\ceil` offer more semantic names to wrap an ⟨*expression*⟩ in the appropriate delimiters. By default, the size of the delimiters is automatically determined using `\left` and `\right`, but this can be overridden by specifying a specific size using the optional ⟨*size*⟩ argument or by including an optional star to avoid sizing the delimiters.

Note that `\set` also adds a `\,` space after the opening brace and before the closing brace. The set on the left below uses `\set` while the set on the right does not:

$$\{\, 1, 2, 3 \,\} \qquad \{1, 2, 3\}$$

`\innerp` `\innerp * [⟨size⟩] {⟨expression⟩} {⟨expression⟩}`

`\innerp` typesets inner products, e.g. $\langle 3, 4 \rangle$. It likewise has automatically sized delimiters by default which can be manually sized with an optional argument, but it takes two arguments instead of one.

`\conj` `\conj {⟨expression⟩}`

`\conj` is an alias for `\overline` that also works outside of math mode. It is a more semantic name for the (complex) conjugate.

4.4 Sets

`\given` `\given`
`\suchthat` `\suchthat`

The `\given` command (alias: `\suchthat`) is used to specify the condition for a set. If it is preceded by a (yet-to-be-closed) `\left`, then it is equivalent to `\middle\vert` with math relation spacing on both sides; otherwise, it is equivalent to `\mid`.

`\RR` $\RR$
`\CC` $\CC$
`\NN` $\NN$
`\FF` $\FF$
`\QQ` $\QQ$
`\ZZ` $\ZZ$

Some shorthands for common blackboard bold (`\mathbb`) letters are provided. These also work outside of math mode, or rather, will automatically enter math mode in that case.

4.5 Vectors

`\bvec` `\bvec {⟨expression⟩}`

The `\bvec` command is a shorthand for `\mathbf` that works outside of math mode as well. It is a shorter and more semantic way to typeset vectors in bold.

```
\mtx  \mtx [⟨alignment⟩] {⟨contents⟩}
\pmtx \pmtx [⟨alignment⟩] {⟨contents⟩}
\bmtx \bmtx [⟨alignment⟩] {⟨contents⟩}
\Bmtx \Bmtx [⟨alignment⟩] {⟨contents⟩}
\vmtx \vmtx [⟨alignment⟩] {⟨contents⟩}
\Vmtx \Vmtx [⟨alignment⟩] {⟨contents⟩}
```

Matrices can also be typset with a shorthand command instead of an environment; this is useful for smaller matrices. For instance:

$$\begin{bmatrix} 33 & 4 \\ 5 & 6 \end{bmatrix}$$

An optional argument can be used to specify the alignment of the columns, using the starred matrix environment variants provided in mathtools. For instance:

$$\begin{bmatrix*[r] 33 & 4 \\ 5 & 6 \end{bmatrix*}$$

The similarly named `\smtx`, `\psmtx`, etc. refer to the small matrix variants. They likewise can take an optional argument for the alignment.

5 Theorems

The gravy package provides a range of colored environments for theorems, lemmas, definitions, remarks, etc. These environments are loosely color-coded by their general semantic meaning.

These environments are defined with the thmtools package, and thus take one optional argument which can either be a name or a key-value list of a name and a label.

5.1 Blue

Blue environments semantically refer to factual claims that are believed or proven to be true. The following environments, along with starred versions of each, are provided:

- `theorem`
- `lemma`
- `corollary`
- `proposition`
- `conjecture`
- `criterion`

- `assertion`

By default, these environments are numbered; the starred versions produce unnumbered theorems. An example of the styling is shown below.

Theorem. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

5.2 Red

Red environments semantically refer to definitions or algorithms, which by definition accurately describe their subject matter. The following environments, along with starred versions of each, are provided:

- `definition`
- `algorithm`

By default, these environments are numbered; the starred versions produce unnumbered theorems. An example of the styling is shown below.

Definition. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

5.3 Orange

Orange environments semantically refer to tangential or additional information. The following environments, with no starred versions, are provided:

- `remark`
- `note`

These environments are not numbered. An example of the styling is shown below.

Remark. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus

et netus et malesuada fames ac turpis egestas.

5.4 Green

Green environments semantically refer to exercises for the reader. The following environments, along with starred versions of eaach, are provided:

- `example`
- `problem`
- `question`

By default, these environments are numbered; the starred versions produce unnumbered theorems. An example of the styling is shown below.

Example. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

5.5 Proofs

Proof. This is a proof, created using the `proof` environment. □

Proof. This is a proof with no QED symbol, created using the `proof*` environment.

Solution. This is a solution, created using the `solution` environment. Note that it has no QED symbol.

6 Vocabulary

`\vocab` `\vocab {⟨term⟩}`

Vocabulary terms can be emphasized with the `\vocab` command. For instance, **this** is a vocabulary word. The red color used is the same as for definitions and algorithms, and is not used for any types of links.

7 Code

Styles for code snippets are not loaded by default, and are instead loaded by passing the `minted` option to the gravy package or document classes. This option loads the minted package and defines a custom style for it. The minted package is not loaded if the `minted` option is not passed.

In order to function, these styles require the `pygments-gravy` custom style for the Pygments syntax highlighter. For more information and installation instructions, see the Gravitonic/pygments-gravy GitHub repository.