La boîte à outils ProjLib

Manuel d'utilisation

Jinwen XU
ProjLib@outlook.com

Octobre 2021, à Paris

Résumé

La boîte à outils ProjLib est conçue pour simplifier la préparation avant d'écrire des documents LaTeX. Avec le package ProjLib ajouté, vous n'avez plus besoin de créer des environnements de type théorème, ni de configurer les paramètres multilingues. De plus, une série de fonctionnalités auxiliaires sont introduites.

Table des matières

Avant de commencer

Pour utiliser la boîte à outils, vous devez :

— installer TeX Live ou MikTeX de la dernière version possible, et vous assurer que `projlib` est correctement installé dans votre système TeX.
— être familiarisé avec l'utilisation de base de LaTeX, et savoir comment compiler vos documents avec pdfLaTeX, XeLaTeX ou LuaLaTeX.

1 Introduction

Le nom ProjLib peut être considéré comme l'abréviation de *Project Library* en anglais ou de *Projet Libre* en français (l'auteur préfère l'interprétation française). Son objectif principal est de fournir un support multilingue et des environnements de type théorème avec des références intelligentes. De plus, certaines fonctionnalités supplémentaires sont fournies, telles que le bloc auteur amélioré, les marques de brouillon, les symboles mathématiques et les raccourcis, etc.

La boîte à outils ProjLib est composée du package principal ProjLib et d'une série de composants dont les noms commencent par l'abréviation « PJL ». Vous pouvez apprendre à l'utiliser à travers les exemples d'utilisation dans la section suivante.

Correspondant à : ProjLib 2021/10/24

2 Exemple d'utilisation

2.1 Comment l'ajouter

Ajoutez simplement la ligne suivante à votre préambule :

```
\usepackage{ProjLib}
```

Attention

Comme cleveref est utilisé en interne, ProjLib doit être placé après varioref et hyperref.

2.2 Exemple - Un document complet

Regardons d'abord un document complet.

```
\documentclass{article}
\usepackage[a4paper,margin=.75in]{geometry}
\usepackage[hidelinks]{hyperref}
\usepackage[palatino]{ProjLib} % Load the toolkit and use font Palatino

\UseLanguage{French} % Use French from here

\begin{document}

\title{⟨title⟩}
\author{⟨author⟩}
\date{\PLdate{2022-04-01}}

\maketitle

\begin{abstract}
    ⟨abstract text⟩ \dnf<⟨some hint⟩>
\end{abstract}

\section{Un théorème}

\begin{theorem}\label{thm:abc}
    Ceci est un théorème.
\end{theorem}

Référence du théorème: \cref{thm:abc} % It is recommended to use clever reference

\end{document}
```

Si vous trouvez cela un peu compliqué, ne vous inquiétez pas. Examinons maintenant cet exemple pièce par pièce.

2.2.1 Initialisation

```
\documentclass{article}
\usepackage[a4paper,margin=.75in]{geometry}
\usepackage[hidelinks]{hyperref}
\usepackage[palatino]{ProjLib}
```

Dans les classes standard, il suffit généralement de configurer la taille de la page, les liens hypertexte et d'ajouter ProjLib avant de commencer à écrire le document. L'option de police `palatino` de ProjLib est utilisée ici. Pour toutes les options disponibles de ProjLib, veuillez vous référer à la section suivante.

Bien sûr, vous pouvez également utiliser la classe de document amsart, les configurations sont les mêmes.

2.2.2 Choisir la langue

```
\UseLanguage{French}
```

Cette ligne indique que le français sera utilisé dans le document (d'ailleurs, si seul l'anglais apparaît dans votre article, alors il n'est pas nécessaire de choisir la langue). Vous pouvez également changer de langue de la même manière plus tard au milieu du texte. Les langues prises en charge sont : chinois simplifié, chinois traditionnel, japonais, anglais, français, allemand, espagnol, portugais, portugais brésilien et russe[1].

Pour une description détaillée de cette commande et d'autres commandes associées, veuillez vous référer à la section sur le support multilingue.

2.2.3 Le titre et les informations de l'auteur

```
\title{⟨title⟩}
\author{⟨author⟩}
\date{\PLdate{2022-04-01}}
```

Cette partie est le titre et le bloc d'informations de l'auteur. L'exemple montre l'utilisation la plus fondamentale, mais en fait, vous pouvez également écrire comme :

```
\author{⟨author 1⟩}
\address{⟨address 1⟩}
\email{⟨email 1⟩}
\author{⟨author 2⟩}
\address{⟨address 2⟩}
\email{⟨email 2⟩}
...
```

De plus, si la simulation d'$\mathcal{AMS}$ est activée, alors vous pouvez également écrire à la manière $\mathcal{AMS}$ (la manière originale fonctionne encore). Pour cela, vous devez ajouter l'option de package `amsfashion`[2], c'est-à-dire que la ligne qui introduit ProjLib doit être écrite comme :

```
\usepackage[amsfashion,palatino]{ProjLib}
```

Et en conséquence, vous pourrez également utiliser ces macros :

[1] Cependant, vous devez ajouter vous-même l'encodage et les polices de la langue correspondante. Par exemple, pour le chinois, vous devrez peut-être ajouter le package ctex et choisir les polices. Pour rappel, vous pouvez essayer les classes de documents einfart ou lebhart de l'auteur, dans lesquelles les paramètres correspondants ont été effectués. Pour les détails, exécutez `texdoc minimalist` ou `coloriste texdoc` en ligne de commande.

[2] Étant donné que cette option modifie certaines macros internes de LaTeX, elle peut entrer en conflit avec certains packages ou classes de documents, et n'est donc pas activée par défaut.

```
\dedicatory{⟨dedicatory⟩}
\subjclass{*****}
\keywords{⟨keywords⟩}
```

De plus, vous pouvez également placé le résumé avant \maketitle, comme requis dans les classes $\mathcal{AMS}$:

```
\begin{abstract}
    ⟨abstract text⟩
\end{abstract}
\maketitle
```

2.2.4 Marques de brouillon

```
\dnf<⟨some hint⟩>
```

Lorsque vous avez des endroits qui ne sont pas encore finis, vous pouvez les marquer avec cette commande, ce qui est particulièrement utile lors de la phase de brouillon.

2.2.5 Environnements de type théorème

```
\begin{theorem}\label{thm:abc}
    Ceci est un théorème.
\end{theorem}
Référence du théorème: \cref{thm:abc}
```

Les environnements de type théorème couramment utilisés ont été prédéfinis. De plus, lors du référencement d'un environnement de type théorème, il est recommandé d'utiliser \cref{⟨label⟩} — de cette manière, il ne serait pas nécessaire d'écrire explicitement le nom de l'environnement correspondant à chaque fois.

3 Options du package principal

ProjLib a les options suivantes :

— `draft` ou `fast`
  – Mode brouillon. La fonctionnalité sera réduite de manière appropriée pour obtenir une vitesse de compilation plus rapide, recommandée à utiliser pendant la phase de brouillon.

— `palatino`, `times`, `garamond`, `noto`, `biolinum` | `useosf`
  – Options de police. Comme les noms l'indiquent, la police avec le nom correspondant sera utilisée.
  – L'option `useosf` est pour activer les chiffres à l'ancienne.

— `nothms`, `delaythms`, `nothmnum`, `thmnum` ou `thmnum=`⟨*counter*⟩, `regionalref`, `originalref`
  – Options du composant PJLthm utilisé pour définir des environnements de type théorème, veuillez vous référer à la section sur ce package pour plus de détails.

— `author`
  – Utilisez le composant PJLauthor pour enrichir le bloc d'informations sur l'auteur. Pour plus d'informations sur ses fonctionnalités, consultez la section sur ce package.

— `amsfashion`
  – Permet d'écrire à la manière $\mathcal{AMS}$. En attendant, l'option `author` sera automatiquement activée.

De plus, il existe également certaines options des composants qui doivent être passées en tant qu'options globales de votre classe de document, telles que les options de langue de PJLlang comme `EN` / `english` / `English`, `FR` / `french` / `French` etc., et les options de papier de PJLpaper comme `paperstyle` et `preview`. Pour plus d'informations, veuillez vous référer aux sections correspondantes.

4 Les composants

4.1 Fonctions principales

4.1.1 PJLauthor : bloc auteur amélioré

PJLauthor propose les macros \address, \curraddr et \email, et vous permet de saisir plusieurs groupes d'informations sur l'auteur. L'utilisation standard est comme ceci :

```
\author{⟨author 1⟩}
\address{⟨address 1⟩}
\email{⟨email 1⟩}
\author{⟨author 2⟩}
\address{⟨address 2⟩}
\email{⟨email 2⟩}
...
```

L'ordre mutuel de \address, \curraddr et \email n'est pas important.

De plus, vous pouvez utiliser l'option `amsfashion` pour écrire à la manière $\mathcal{AMS}$. Plus précisément, l'effet est :

— Fournit les macros \dedicatory, \keywords et \subjclass ;
— \thanks peut être écrit en dehors de la macro \author ;
— La environnement `abstract` peut être placé avant \maketitle.

> **Attention**
>
> Ces modifications n'auraient lieu que dans les classes standard. Dans les classes $\mathcal{AMS}$, PJLauthor n'a aucun effet.

4.1.2 PJLlang : support multilingue

PJLlang offre le support multilingue, notamment : chinois simplifié, chinois traditionnel, anglais, français, allemand, japonais et russe (parmi eux, le chinois, le japonais et le russe requièrent des moteurs TEX et des polices appropriés).

PJLlang fournit des options de langue. Les noms de ces options ont trois types, qui sont des abréviations (comme `EN`), des minuscules (comme `english`) et des majuscules (comme `English`). Pour les noms d'options d'une langue spécifique, veuillez vous référer à ⟨*language name*⟩ ci-dessous. Parmi eux, la première langue spécifiée ⟨*first language*⟩ sera considérée comme langue par défaut, ce qui équivaut à spécifier \UseLanguage{⟨*first language*⟩} au début de votre document.

> **Astuce**
>
> Il est recommandé d'utiliser ces options de langue et de les passer en tant qu'options globales. De cette façon, seules les langues spécifiées sont configurées, économisant ainsi la mémoire TEX et améliorant considérablement la vitesse de compilation.

La langue peut être sélectionnée par les macros suivantes :

— \UseLanguage{⟨*language name*⟩} est utilisé pour spécifier la langue. Le réglage correspondant de la langue sera appliqué après celui-ci. Il peut être utilisé soit dans le préambule ou dans le texte. Lorsqu'aucune langue n'est spécifiée, « English » est sélectionné par défaut.
— \UseOtherLanguage{⟨*language name*⟩}{⟨*content*⟩}, qui utilise les paramètres de langue spécifiés pour composer ⟨*content*⟩. Par rapport à \UseLanguage, il ne modifiera pas l'interligne, donc l'interligne restera stable lorsque les textes CJK et occidentaux sont mélangés.

⟨*language name*⟩ peut être (il n'est pas sensible à la casse, par exemple, `French` et `french` ont le même effet) :

— chinois simplifié : `CN`, `Chinese`, `SChinese` ou `SimplifiedChinese`
— chinois traditionnel : `TC`, `TChinese` ou `TraditionalChinese`
— anglais : `EN` ou `English`
— français : `FR` ou `French`
— allemand : `DE`, `German` ou `ngerman`
— italien : `IT` ou `Italian`
— portugais : `PT` ou `Portuguese`
— portugais (brésilien) : `BR` ou `Brazilian`
— espagnol : `ES` ou `Spanish`
— japonais : `JP` ou `Japanese`
— russe : `RU` ou `Russian`

De plus, vous pouvez également ajouter de nouveaux paramètres à la langue sélectionnée :

— `\AddLanguageSetting{`⟨*settings*⟩`}`
  – Ajoutez ⟨*settings*⟩ à toutes les langues prises en charge.
— `\AddLanguageSetting(`⟨*language name*⟩`){`⟨*settings*⟩`}`
  – Ajoutez ⟨*settings*⟩ à la langue ⟨*language name*⟩ sélectionnée.

Par exemple, `\AddLanguageSetting(German){\color{orange}}` peut rendre tout le texte allemand affiché en orange (bien sûr, il faut alors ajouter `\AddLanguageSetting{\color{black}}` afin de corriger la couleur du texte dans d'autres langues).

4.1.3 PJLthm : environnements de type théorème avec référence intelligente et support multilingue

PJLthm offre la configuration d'environnements de type théorème. Il a l'option suivante :

— `nothms`
  – Les environnements de type théorème ne seront pas définis. Vous pouvez utiliser cette option si vous souhaitez appliquer vos propres styles de théorème.
— `delaythms`
  – Reportez la définition des environnements de type théorème à la fin du préambule. Utilisez cette option si vous souhaitez que les environnements soient numérotés dans un compteur personnalisé.
— `nothmnum`, `thmnum` ou `thmnum=`⟨*counter*⟩
  – Les environnements de type théorème ne seront pas numérotés / numérotés dans l'ordre 1, 2, 3... / numérotés dans ⟨*counter*⟩. Ici, ⟨*counter*⟩ doit être un compteur intégré (tel que `subsection`) ou un compteur défini dans le préambule (avec l'option `delaythms` activée). Si aucune option n'est utilisée, ils seront numérotés dans `chapter` (livre) ou `section` (article).
— `regionalref`, `originalref`
  – Lors du référencement, si le nom de l'environnement de type théorème change avec la langue actuelle. Par défaut `regionalref` est activé, c'est-à-dire que le nom correspondant à la langue courante est utilisé ; par exemple, lors du référencement d'un environnement de type théorème dans un contexte français, les noms « Théorème, Définition ... » seront utilisés quel que soit le contexte linguistique dans lequel se trouve l'environnement d'origine. Si `originalref` est activé, alors le nom restera toujours le même que l'environnement d'origine ; par exemple, lors du référencement d'un théorème écrit dans le contexte français, même si l'on est actuellement dans le contexte anglais, il sera toujours affiché comme « Théorème ».
  – En mode `fast`, l'option `originalref` n'aura aucun effet.

Les environnements prédéfinis incluent : `assumption`, `axiom`, `conjecture`, `convention`, `corollary`, `definition`, `definition-proposition`, `definition-theorem`, `example`, `exercise`, `fact`, `hypothesis`, `lemma`, `notation`, `observation`, `problem`, `property`, `proposition`, `question`, `remark`, `theorem`, et la version non numérotée correspondante avec un astérisque `*` dans le nom. Les titres changeront avec la langue actuelle. Par exemple, `theorem` sera affiché comme « Theorem » en mode anglais et « Théorème » en mode français. Pour plus de détails sur la façon de sélectionner une langue, veuillez vous référer à la section sur PJLlang.

Astuce

Lors du référencement d'un environnement de type théorème, il est recommandé d'utiliser `\cref{`⟨*label*⟩`}`. De cette façon, il n'est pas nécessaire d'écrire explicitement le nom de l'environnement correspondant à chaque fois.

Si vous avez besoin de définir un nouvel environnement de type théorème, vous devez d'abord définir le nom de l'environnement dans le langage à utiliser. Il y a deux façons pour cela :

— Paramètres simples : `\NameTheorem[`⟨*language name*⟩`]{`⟨*name of environment*⟩`}{`⟨*name string*⟩`}`

  – Cette approche ne définit qu'un seul nom principal. Les autres noms, tels que ceux utilisés pour référence intelligente, sont définis pour être identiques (en particulier, pour référence intelligente, les noms singulier et pluriel ne seront pas distingués). Lorsque ⟨*language name*⟩ n'est pas spécifié, le nom sera défini pour toutes les langues prises en charge. De plus, les environnements avec ou sans astérisque partagent le même nom, donc, `\NameTheorem{envname*}` a le même effet que `\NameTheorem{envname}` .

— Paramètres détaillés (recommandés) :

```
\NameTheorem{⟨name of environment⟩}{
    ⟨language name 1⟩={
        name=⟨Name⟩,
        crefname={⟨name⟩}{⟨names⟩},
        Crefname={⟨Name⟩}{⟨Names⟩},
        autorefname=⟨name⟩,
        theoremheading=⟨Name⟩,
    },
    ⟨language name 2⟩={...},
}
```

ou

```
\NameTheorem[⟨language name⟩]{⟨name of environment⟩}{
    name=⟨Name⟩,
    crefname={⟨name⟩}{⟨names⟩},
    Crefname={⟨Name⟩}{⟨Names⟩},
    autorefname=⟨name⟩,
    theoremheading=⟨Name⟩,
}
```

  – Cette approche définit tous les noms. Lorsque ⟨*language name*⟩ n'est pas spécifié, l'interface complète sera activée ; lorsqu'il est spécifié, seuls les noms de la langue correspondante sont définis. De même, les environnements avec ou sans astérisque partagent le même nom, donc `\NameTheorem{envname*}` a le même effet que `\NameTheorem{envname}` .

Astuce

De plus, vous pouvez également nommer un environnement de type théorème tout en le définissant, voir la description de \CreateTheorem plus tard.

Ensuite, créez cet environnement de l'une des cinq manières suivantes :

— \CreateTheorem*{⟨*name of environment*⟩}
  – Définir un environnement ⟨*name of environment*⟩ non numéroté
— \CreateTheorem{⟨*name of environment*⟩}
  – Définir un environnement ⟨*name of environment*⟩ numéroté dans l'ordre 1, 2, 3, …
— \CreateTheorem{⟨*name of environment*⟩}[⟨*numbered like*⟩]
  – Définir un environnement ⟨*name of environment*⟩ numéroté, qui partage le compteur ⟨*numbered like*⟩
— \CreateTheorem{⟨*name of environment*⟩}<⟨*numbered within*⟩>
  – Définir un environnement ⟨*name of environment*⟩ numéroté dans le compteur ⟨*numbered within*⟩
— \CreateTheorem{⟨*name of environment*⟩}(⟨*existed environment*⟩)
  \CreateTheorem*{⟨*name of environment*⟩}(⟨*existed environment*⟩)
  – Identifiez ⟨*name of environment*⟩ avec ⟨*existed environment*⟩ ou ⟨*existed environment*⟩*.
  – Cette méthode est généralement utile dans les deux situations suivantes :
    1) Pour utiliser un nom plus concis. Par exemple, avec \CreateTheorem{thm}(theorem), on peut alors utiliser le nom thm pour écrire le théorème.
    2) Pour supprimer la numérotation de certains environnements. Par exemple, on peut supprimer la numérotation de l'environnement remark avec \CreateTheorem{remark}(remark*).

Astuce

Cette macro utilise la fonctionnalité de amsthm en interne, donc le traditionnel theoremstyle lui est également applicable. Il suffit de déclarer le style avant les définitions pertinentes.

Vous pouvez également nommer un environnement de type théorème lors de sa définition, en ajoutant ensuite un groupe de parenthèses contenant les paramètres :

```
\CreateTheorem{⟨name of environment⟩}{
    ⟨language name 1⟩={
        name=⟨Name⟩,
        crefname={⟨name⟩}{⟨names⟩},
        Crefname={⟨Name⟩}{⟨Names⟩},
        autorefname=⟨name⟩,
        theoremheading=⟨Name⟩,
    },
    ⟨language name 2⟩={...},
}
```

Voici un exemple. Le code suivant :

```
\NameTheorem[FR]{proofidea}{Idée}
\CreateTheorem*{proofidea*}
\CreateTheorem{proofidea}<subsection>
```

définit un environnement non numéroté `proofidea*` et un environnement numéroté `proofidea` (numérotés dans la sous-section) respectivement. Ils peuvent être utilisés dans le contexte français. L'effet est le suivant (le style réel est lié à votre classe de document) :

Idée | La environnement `proofidea*` .

Idée 4.1.1 | La environnement `proofidea` .

Bien sûr, vous pouvez également utiliser un ensemble de nom plus détaillé :

```
\NameTheorem{proofidea}{
    FR = {
        name = Idée,
        crefname = {idée}{idées},
        Crefname = {Idée}{Idées},
    }
}
\CreateTheorem*{proofidea*}
\CreateTheorem{proofidea}<subsection>
```

ou définissez les noms en les définissant (pour `proofidea*` et `proofidea`, définir une fois suffit) :

```
\CreateTheorem*{proofidea*}
\CreateTheorem{proofidea}<subsection>{
    FR = {
        name = Idée,
        crefname = {idée}{idées},
        Crefname = {Idée}{Idées},
    }
}
```

4.2 Fonctions secondaires

4.2.1 PJLdate : traitement de date-heure

PJLdate propose la macro \PLdate⟨*yyyy-mm-dd*⟩ (ou \PJLdate⟨*yyyy-mm-dd*⟩) pour convertir ⟨*yyyy-mm-dd*⟩ dans le format de date de la langue actuellement sélectionnée. Par exemple, dans le contexte français actuel, \PLdate{2022-04-01} deviendrait « 1er avril 2022 », tandis que dans le contexte anglais « April 1, 2022 ».

Pour plus de détails sur la façon de sélectionner une langue, veuillez vous référer à la section sur PJLlang.

4.2.2 PJLdraft : marques de brouillon

PJLdraft propose les macros suivantes :

— \dnf ou \dnf<...>. L'effet est : `Pas encore fini #1` ou `Pas encore fini #2: ...`.
Le texte à l'intérieur changera en fonction de la langue actuelle. Par exemple, il sera affiché sous la forme `To be finished #3` en mode anglais.
— \needgraph ou \needgraph<...>. L'effet est :

`Il manque une image ici #1`

ou

`Il manque une image ici #2: ...`

Le texte de l'invite change en fonction de la langue actuelle. Par exemple, en mode anglais, il sera affiché sous la forme

`A graph is needed here #3`

Pour plus de détails sur la façon de sélectionner une langue, veuillez vous référer à la section sur PJLlang.

4.2.3 PJLlogo : le logo ProjLib

PJLlogo propose la macro \ProjLib pour dessiner le logo, qui ressemble à ProjLib. Elle est similaire aux macros de texte ordinaires et peut être utilisée avec différentes macros de taille de texte :

\tiny : ProjLib
\scriptsize : ProjLib
\footnotesize : ProjLib
\normalsize : ProjLib
\large : ProjLib
\Large : ProjLib
\LARGE : ProjLib
\huge : ProjLib
\Huge : ProjLib

4.2.4 PJLmath : symboles et raccourcis mathématiques

PJLmath propose les raccourcis suivants :

i) \mathfrak{·} ⟶ \mf· ou \frak· . Par exemple, \mfA (ou \mf{A}) a le même effet que \mathfrak{A}. Cela fonctionne à la fois pour l'alphabet majuscule et minuscule, produisant :

𝔞𝔟𝔠𝔡𝔢𝔣𝔤𝔥𝔦𝔧𝔨𝔩𝔪𝔫𝔬𝔭𝔮𝔯𝔰𝔱𝔲𝔳𝔴𝔵𝔶𝔷
𝔄𝔅ℭ𝔇𝔈𝔉𝔊ℌℑ𝔍𝔎𝔏𝔐𝔑𝔒𝔓𝔔ℜ𝔖𝔗𝔘𝔙𝔚𝔛𝔜ℨ

ii) \mathbb{·} ⟶ \bb· . Cela ne fonctionne que pour l'alphabet majuscule et le nombre 1.

$$\mathbb{ABCDEFGHIJKLMNOPQRSTUVWXYZ1}$$

Il y a aussi des commandes spéciales pour les structures algébriques bien connues : \N, \Z, \Q, \R, \C, \F, \A.

$$\mathbb{NZQRCFA}$$

iii) \mathcal{·} ⟶ \mc· or \cal· . Cela ne fonctionne que pour l'alphabet majuscule.

$$\mathcal{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$$

iv) \mathscr{·} ⟶ \ms· or \scr· . Cela ne fonctionne que pour l'alphabet majuscule.

$$\mathscr{A\,B\,C\,D\,E\,F\,G\,H\,I\,\,J\,K\,L\,M\,N\,O\,P\,Q\,R\,S\,T\,U\,V\,W\,X\,Y\,Z}$$

De plus, PJLmath fournit également des symboles mathématiques qui ne sont pas inclus par défaut avec LaTeX.

\abs	\abs{a} → $\abs{a}$	symbole de valeur absolue
\norm	\norm{a} → $\norm{a}$	symbole de norme
\injection	\injection → $\injection$	symbole de flèche pour l'injection
\surjection	\surjection → $\surjection$	symbole de flèche pour la surjection
\bijection	\bijection → $\bijection$	symbole de flèche pour la bijection

Ces raccourcis et symboles sont définis de telle manière qu'ils n'entrent pas en conflit avec les commandes existantes ou les commandes définies par l'utilisateur. Ainsi, même dans le cas où vous n'utilisez pas ces raccourcis ou symboles, ne vous inquiétez pas que leur existence apporte des erreurs.

4.2.5 PJLpaper : configuration papier

PJLpaper est principalement utilisé pour ajuster la couleur du papier. Il a les options suivantes :

— `paperstyle =` ⟨*paper style name*⟩
  – Définit le style de couleur du papier. Les options disponibles pour ⟨*paper style name*⟩ sont : `yellow`, `dark` et `nord`.

— `yellowpaper`, `darkpaper`, `nordpaper`
  – Identique à `paperstyle` avec le ⟨*paper style name*⟩ correspondant spécifié.

— `preview`
  – Mode aperçu. Recadrez les bords blancs du fichier pdf pour faciliter la lecture.

Il est recommandé de les passer comme options globales de la classe de document. De cette façon, les paramètres du papier seraient clairs en un coup d'œil.

5 Problèmes connus

— PJLauthor est encore à son stade préliminaire, son effet n'est pas aussi bon que le authblk qui est relativement mature.

— PJLlang : C'est encore assez problématique avec la configuration de polyglossia, donc les fonctionnalités principales sont implémentées via babel pour le moment.

— PJLpaper : l'option `preview` est principalement implémentée à l'aide du package geometry, elle ne fonctionne donc pas aussi bien dans les classes de documents KOMA.

— PJLthm : les paramètres de numérotation et de style théorème des environnements de type théorème ne sont actuellement pas accessibles à l'utilisateur.

— PJLthm : la localisation de cleveref n'est pas encore finie pour toutes les langues prises en charge par PJLlang, en particulier pour le chinois, le japonais et le russe.

— Le mécanisme de gestion des erreurs est incomplet : pas de messages correspondants lorsque certains problèmes surviennent.

— Il y a encore beaucoup de choses qui peuvent être optimisées dans le code. Certains codes prennent trop de temps à s'exécuter, en particulier la configuration d'environnements de type théorème dans PJLthm.