

Le package `tnsproba` : parler des probabilités facilement

Code source disponible sur <https://github.com/typensee-latex/tnsproba.git>.

Version 0.2.0-beta développée et testée sur Mac OS X.

Christophe BAL

2020-07-23

Table des matières

1. Introduction	2
2. Ensembles probabilistes	2
3. Généralités	2
a. Probabilité « simple »	2
b. Probabilité conditionnelle	2
c. Évènement contraire	3
d. Espérance, variance et écart-type	3
4. Arbres pondérés	3
a. Que se passe-t-il en coulisse ?	3
b. Sans cadre	3
c. Avec des cadres	5
5. Historique	7
6. Toutes les fiches techniques	8
a. Généralités	8
i. Probabilité « simple »	8
ii. Probabilité conditionnelle	8
iii. Évènement contraire	8
iv. Espérance, variance et écart-type	8
b. Arbres pondérés	8

1. Introduction

Le package `tnsproba` propose des macros utiles quand l'on parle de probabilités. La saisie se veut sémantique et simple.

2. Ensembles probabilistes

Le package `tnssets` propose le macro `\setproba` pour indiquer des ensembles de type probabiliste. Se rendre sur <https://github.com/typensee-latex/tnssets.git> si cela vous intéresse.

3. Généralités

a. Probabilité « simple »

Exemple 1

<code>\backslashproba{A}</code>	$p(A)$
--	--------

Exemple 2 – Choisir le nom de la probabilité

<code>\backslashproba[P]{A}</code>	$P(A)$
---	--------

b. Probabilité conditionnelle

Exemple 1 – Les deux écritures classiques

La 1^{re} notation, qui est devenue standard, permet de comprendre l'ordre des arguments.

<code>\backslashprobacond {B}{A}</code> <code>= \probacond*{B}{A}</code>	$p_B(A) = p(A \mid B)$
--	------------------------

Exemple 2 – Obtenir la formule de définition

Le préfixe `e` est pour `e`-xpannd soit « *développer* » en anglais¹.

<code>\backslasheprobacond {B}{A}</code> <code>= \eprobacond*{B}{A}</code>	$\frac{p(A \cap B)}{p(B)} = \frac{p(A \cap B)}{p(B)}$
--	---

Exemple 3 – Choisir le nom de la probabilité

<code>\backslashprobacond [P]{B}{A}</code> <code>= \probacond* [P]{B}{A}</code> <code>= \eprobacond*[P]{B}{A}</code> <code>= \eprobacond [P]{B}{A}</code>	$P_B(A) = P(A \mid B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A \cap B)}{P(B)}$
---	--

1. Pour ne pas alourdir l'utilisation de `\probacond`, il a été choisi d'utiliser un préfixe au lieu d'un système de multi-options.

c. Évènement contraire

`\nevent` vient de `n-ot event` qui est une pseudo-traduction de « *évènement contraire* » en anglais.

<code>\$\nevent{A}\$</code>	\overline{A}
-----------------------------	----------------

d. Espérance, variance et écart-type

Exemple 1 – Espérance

`\expval` vient de `exp-ected val-ue` soit « *espérance* » en anglais.

<code>\$\expval{X}\$</code>	$E(X)$
-----------------------------	--------

Exemple 2 – Choisir le nom de l'espérance

<code>\$\expval[E_1]{X}\$</code>	$E_1(X)$
----------------------------------	----------

Exemple 3 – Variance

<code>\$\var{X}\$</code> ou <code>\$\var[v]{X}\$</code>	$V(X)$ ou $v(X)$
--	------------------

Exemple 4 – Écart-type

`\stddev` vient de `st-andar-d dev-iation` soit « *écart-type* » en anglais.

<code>\$\stddev{X}\$</code> ou <code>\$\stddev[s]{X}\$</code>	$\sigma(X)$ ou $s(X)$
--	-----------------------

4. Arbres pondérés

a. Que se passe-t-il en coulisse ?

Le gros du travail est fait par le package `forest` qui utilise `Tikz`. On peut donc faire appel à la machinerie de ce dernier et obtenir des choses sympatiques comme nous allons le voir ci-dessous.

b. Sans cadre

Exemple 1 – Le cas type

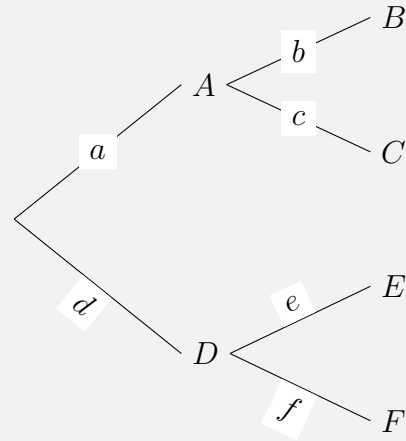
Dans le code suivant l'environnement `probatree` utilise en coulisse celui nommé `forest` du package `forest`. Des réglages spécifiques sont faits pour obtenir le résultat ci-après. À cela s'ajoutent les styles spéciaux `pweight`, `apweight` et `bpweight` qui facilitent l'écriture des pondérations sur les branches².

2. `pweight` vient de « *probability* » et « *weight* » soit « *probabilité* » et « *poids* » en anglais. Quant à `a` et `b` au début de `apweight` et `bpweight` respectivement, ils viennent de « *above* » et « *below* » soit « *dessus* » et « *dessous* » en anglais.

```

\begin{probatree}
[
  [$A$, pweight = $a$
    [$B$, pweight = $b$]
    [$C$, pweight = $c$]
  ]
  [$D$, bpweight = $d$
    [$E$, apweight = $e$]
    [$F$, bpweight = $f$]
  ]
]
\end{probatree}

```



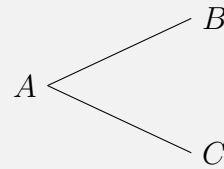
Exemple 2 – Des poids cachés partout

On peut cacher tous les poids via l'environnement étoilé `probatree*` sans avoir à les effacer partout dans le code \LaTeX .

```

\begin{probatree*}
  [$A$, pweight = $a$
    [$B$, pweight = $b$]
    [$C$, pweight = $c$]
  ]
\end{probatree*}

```



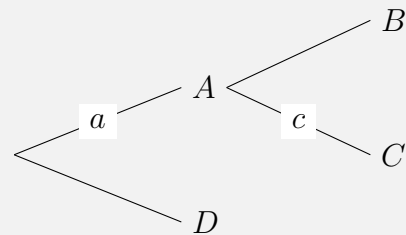
Exemple 3 – Des poids cachés localement

Pour ne cacher que certains poids, il faudra utiliser localement le style `pweight*` comme dans l'exemple ci-dessous.

```

\begin{probatree}
[
  [$A$, pweight = $a$
    [$B$, pweight* = $b$]
    [$C$, pweight = $c$]
  ]
  [$D$, pweight* = $d$]
]
\end{probatree}

```



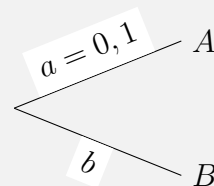
Exemple 4 – Un signe = et/ou une virgule dans les étiquettes

Vous ne pouvez pas utiliser directement un signe = ou une virgule dans les étiquettes des branches. Pour contourner cette limitation, il suffit de mettre le contenu de l'étiquette dans des accolades.

```

\begin{probatree}
[
  [$A$, apweight = {$a = 0,1$}]
  [$B$, bpweight = $b$]
]
\end{probatree}

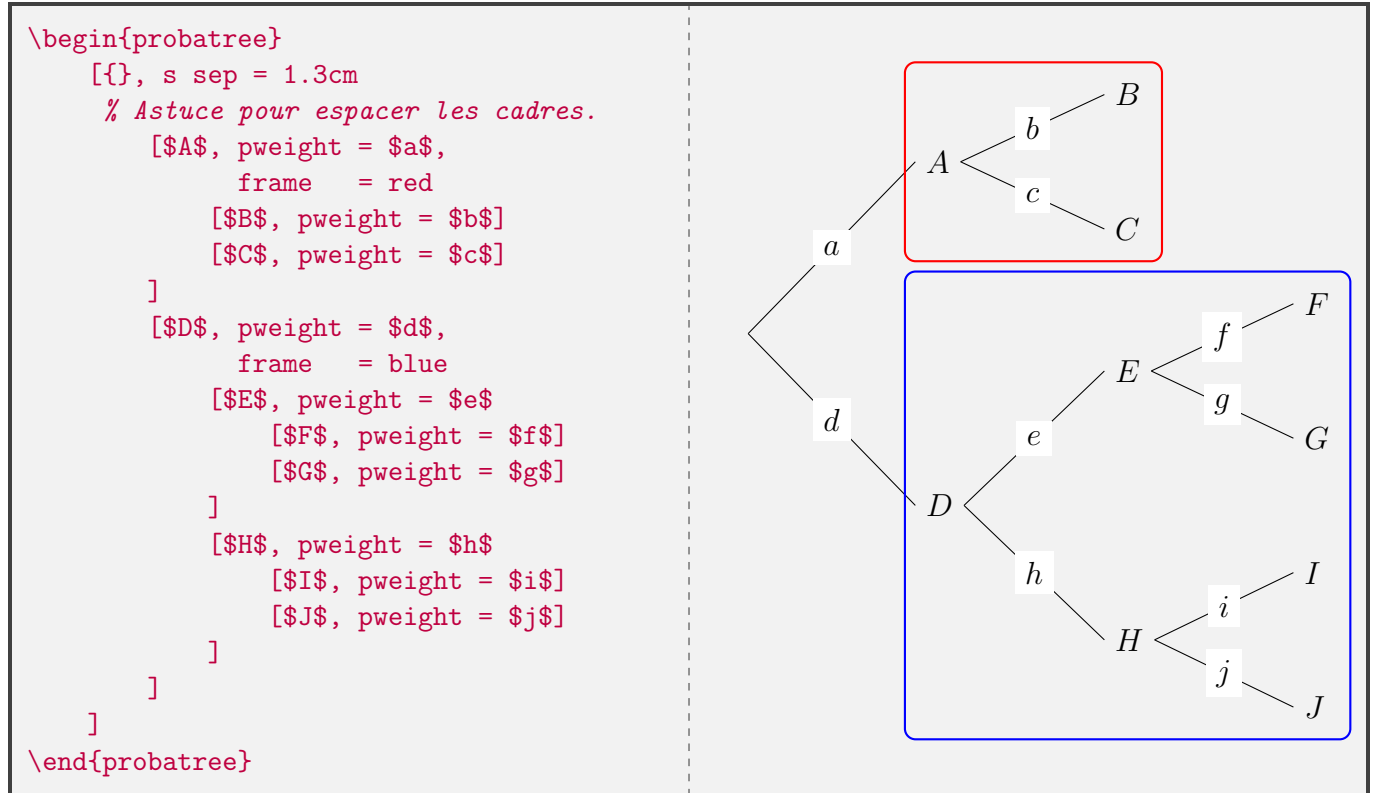
```



c. Avec des cadres

Exemple 1 – Des cadres facilement

Via la clé `frame`, il est très aisé d'encadrer un sous-arbre final³ comme le montre l'exemple suivant. Dans l'exemple ci-après nous utilisons la bidouille `{},s sep = 1.3cm` qui évite que les cadres se superposent.



Exemple 2 – Des cadres faits à la main

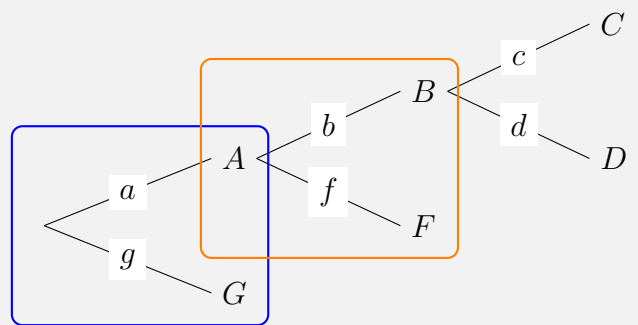
La macro `\ptreeFrame` permet facilement d'encadrer un sous-arbre non final. Ceci nécessite de nommer les noeuds mais c'est facile à faire. Voici un exemple où la macro `\ptreeFrame` attend les noms de la racine et des deux noeuds finaux le plus haut et le plus bas.

3. Un sous-arbre sera dit final si toutes ses feuilles correspondent à des feuilles de l'arbre initial.

```

\begin{probatree}
  [{}], name = nU
    [$A$, pweight = $a$,
      name      = nA
    [$B$, pweight = $b$,
      name      = nB
    [$C$, pweight = $c$]
    [$D$, pweight = $d$]
  ]
  [$F$, pweight = $f$,
    name      = nF]
  ]
  [$G$, pweight = $g$,
    name      = nG]
]
\ptreeFrame      {nU}-{nA}-{nG}
\ptreeFrame[orange] {nA}-{nB}-{nF}
\end{probatree}

```



5. Historique

Nous ne donnons ici qu'un très bref historique récent ⁴ de **tnsproba** à destination de l'utilisateur principalement. Tous les changements sont disponibles uniquement en anglais dans le dossier **change-log** : voir le code source de **tnsproba** sur **github**.

2020-07-23 Nouvelle version mineure 0.2.0-beta.

- **ARBRE** : ajout de la macro `\ptreeFrame` pour tracer facilement des sous cadres non « finaux ».
-

2020-07-22 Nouvelle version mineure 0.1.0-beta.

- **PROBABILITÉ CONDITIONNELLE** : `\probacondexp` renommée en `\eprobacond`.
 - **ÉVÈNEMENT CONTRAIRE** : ajout de `\nevent`.
 - **VARIANCE ET ÉCART-TYPE** : ajout de `\var` et `\stddev`.
-

2020-07-10 Première version 0.0.0-beta.

4. On ne va pas au-delà de un an depuis la dernière version.

6. Toutes les fiches techniques

a. Généralités

i. Probabilité « simple »

`\proba[#opt]{#1}`

— Option: le nom de la probabilité. La valeur par défaut est p .

— Argument: l'ensemble dont on veut calculer la probabilité.

ii. Probabilité conditionnelle

`\probacond [#opt]{#1..#2}`

`\probacond* [#opt]{#1..#2}`

`\eprobacond [#opt]{#1..#2}`

`\eprobacond* [#opt]{#1..#2}`

— Option: le nom de la probabilité. La valeur par défaut est p .

— Argument 1: l'ensemble qui donne la condition.

— Argument 2: l'ensemble dont on veut calculer la probabilité.

iii. Évènement contraire

`\nevent{#1}`

— Argument: l'ensemble dont on veut indiquer le contraire.

iv. Espérance, variance et écart-type

`\expval[#opt]{#1}`

— Option: le nom de la fonction espérance. La valeur par défaut est E obtenue via `\mathrm{E}`.

— Argument: la variable aléatoire dont on veut calculer l'espérance.

`\var[#opt]{#1}`

— Option: le nom de la fonction variance. La valeur par défaut est V obtenue via `\mathrm{V}`.

— Argument: la variable aléatoire dont on veut calculer la variance.

`\stddev[#opt]{#1}`

— Option: le nom de la fonction écart-type. La valeur par défaut est σ obtenue via `\sigma`.

— Argument: la variable aléatoire dont on veut calculer l'écart-type.

b. Arbres pondérés

`\begin{probatree }{#1}`

...

`\end{probatree }`

`\begin{probatree*}{#1}`

...
`\end{probatree*}`

- **Contenu**: un arbre codé en utilisant la syntaxe supportée par le package **forest**.
- Clé **"pweight "**: pour écrire un poids sur le milieu d'une branche.
- Clé **"apweight"**: pour écrire un poids au-dessus le milieu d'une branche.
- Clé **"bpweight"**: pour écrire un poids en-dessous du milieu d'une branche.
- Clé **"frame"**: pour encadrer un sous-arbre depuis un noeud vers toutes les feuilles de celui-ci.

`\ptreeFrame [#opt] {#1..#3}` p = p-robabilty

- **Option**: la couleur au format TikZ. La valeur par défaut est **blue**.
- **Arguments 1..3**: noms de la sous-racine (à gauche), du noeud final en haut (à droite) et du noeud final en bas (à droite). Ici l'ordre n'est pas important.