Le package `tnsproba` : parler des probabilités facilement

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

Version `0.1.0-beta` développée et testée sur Mac OS X.

Christophe BAL

2020-07-22

Table des matières

---

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

3.1 Probabilité « simple »

Exemple 1

`$\proba{A}$`	$p(A)$

Exemple 2 – Choisir le nom de la probabilité

`$\proba[P]{A}$`	$P(A)$

3.2 Probabilité conditionnelle

Exemple 1 – Les deux écritures classiques

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

`$\probacond {B}{A}` `= \probacond*{B}{A}$`	$p_B(A) = p(A \mid B)$

Exemple 2 – Obtenir la formule de définition

Le préfixe `e` est pour `e`-xpand soit *« développer »* en anglais [1].

`$\eprobacond {B}{A}` `= \eprobacond*{B}{A}$`	$\frac{p(A \cap B)}{p(B)} = \frac{p(A \cap B)}{p(B)}$

Exemple 3 – Choisir le nom de la probabilité

`$\probacond [P]{B}{A}`
`= \probacond* [P]{B}{A}`
`= \eprobacond*[P]{B}{A}`
`= \eprobacond [P]{B}{A}$`
	$P_B(A) = P(A \mid B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A \cap B)}{P(B)}$

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.

3.3 Évènement contraire

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

`$\nevent{A}$`	$\overline{A}$

3.4 Espérance, variance et écart-type

Exemple 1 – Espérance

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

`$\expval{X}$`	$\mathrm{E}(X)$

Exemple 2 – Choisir le nom de l'espérance

`$\expval[E_1]{X}$`	$E_1(X)$

Exemple 3 – Variance

`$\var  {X}$ ou` `$\var[v]{X}$`	$\mathrm{V}(X)$ ou $v(X)$

Exemple 4 – Écart-type

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

`$\stddev  {X}$ ou` `$\stddev[s]{X}$`	$\sigma(X)$ ou $s(X)$

4 Arbres pondérés

4.1 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 sympathiques comme nous allons le voir ci-dessous.

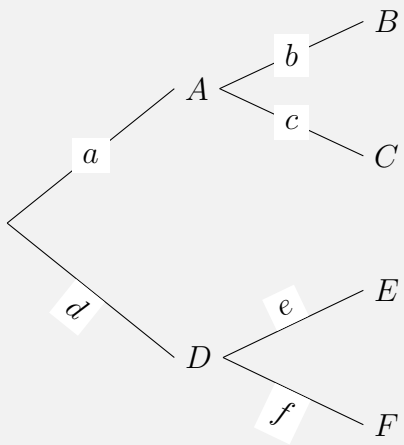
4.2 Sans cadre

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 branches [2].

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.

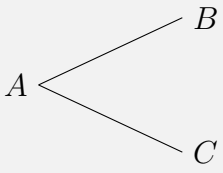
```
\begin{probatree}
    [
        [$A$, pweight = $a$
            [$B$, pweight = $b$]
            [$C$, pweight = $c$]
        ]
        [$D$, bpweight = $d$
            [$E$, apweight = $e$]
            [$F$, bpweight = $f$]
        ]
    ]
\end{probatree}
```



Exemple 2 – Des poids cachés partout

On peut cacher tous les poids via l'environnement étoilé `probatree*` sans avoir à retaper un arbre où les pondérations ont déjà été indiquées.

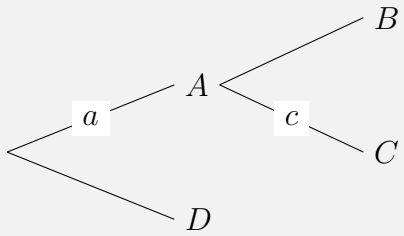
```
\begin{probatree*}
    [$A$, pweight = $a$
        [$B$, pweight = $b$]
        [$C$, pweight = $c$]
    ]
\end{probatree*}
```



Exemple 3 – Des poids cachés localement

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

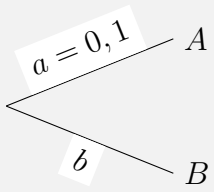
```
\begin{probatree}
    [
        [$A$, pweight = $a$
            [$B$, pweight* = $b$]
            [$C$, pweight  = $c$]
        ]
        [$D$, pweight* = $d$]
    ]
\end{probatree}
```



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. L'astuce pour contourner cette limitation consiste juste à mettre le contenu de l'étiquette dans des accolades.

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

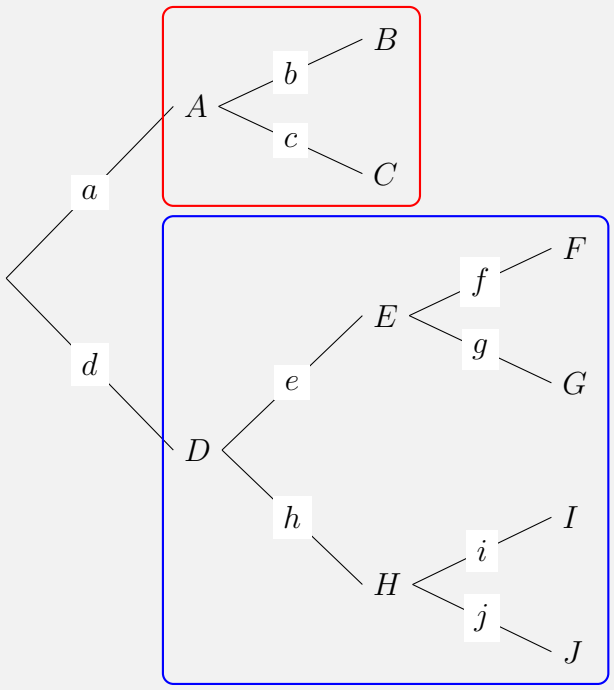


4.3 Avec des cadres

Exemple 1 – Des cadres facilement

Via la clé `frame`, il est très aisé d'encadrer un sous-arbre 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.

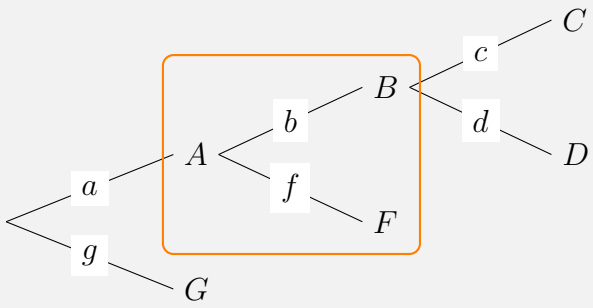
```
\begin{probatree}
    [{}, s sep = 1.3cm
     % Astuce pour espacer les cadres.
        [$A$, pweight = $a$,
              frame   = red
            [$B$, pweight = $b$]
            [$C$, pweight = $c$]
        ]
        [$D$, pweight = $d$,
              frame   = blue
            [$E$, pweight = $e$
                [$F$, pweight = $f$]
                [$G$, pweight = $g$]
            ]
            [$H$, pweight = $h$
                [$I$, pweight = $i$]
                [$J$, pweight = $j$]
            ]
        ]
    ]
\end{probatree}
```



Exemple 2 – Des cadres faits à la main

En utilisant la machinerie de `TiKz` il est facile de décorer un arbre de probabilité comme ci-dessous où le cadre s'appuie sur trois noeuds nommés. Notons que cet exemple est tout simplement infaisable avec la clé `frame`.

```
\begin{probatree}
    [
        [$A$, pweight = $a$,
              name    = nA
            [$B$, pweight = $b$,
                  name    = nB
                [$C$, pweight = $c$]
                [$D$, pweight = $d$]
               ]
            [$F$, pweight = $f$,
                  name    = nF]
        ]
        [$G$, pweight = $g$]
    ]
    \node[draw = orange,
          thick,
          rounded corners,
          fit = (nA)(nB)(nF)] {};
\end{probatree}
```



5 Historique

Nous ne donnons ici qu'un très bref historique récent [3] 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`.

2020-07-22 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`.

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

6 Toutes les fiches techniques

6.1 Généralités

6.1.1 Probabilité « simple »

`\proba <macro> [1 Option] (1 Argument)`

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

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

6.1.2 Probabilité conditionnelle

```
\probacond  <macro> [1 Option] (2 Arguments)
\probacond*  <macro> [1 Option] (2 Arguments)
\eprobacond  <macro> [1 Option] (2 Arguments)
\eprobacond* <macro> [1 Option] (2 Arguments)
```

— `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.

6.1.3 Évènement contraire

`\nevent <macro> (1 Argument)`

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

6.1.4 Espérance, variance et écart-type

`\expval <macro> [1 Option] (1 Argument)`

— `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 <macro> [1 Option] (1 Argument)`

— `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 <macro> [1 Option] (1 Argument)`

— `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.

6.2 Arbres pondérés

```
probatree  <env>
probatree* <env>
```

— `Contenu:` un arbre codé en utilisant la syntaxe supportée par le package `forest`.

— `Option "pweight "`: pour écrire un poids sur le milieu d'une branche.

— `Option "apweight"`: pour écrire un poids au-dessus le milieu d'une branche.

— `Option "bpweight"`: pour écrire un poids en-dessous du milieu d'une branche.

— `Option "frame"`: pour encadrer un sous-arbre depuis un noeud vers toutes les feuilles de celui-ci.