Le package `tnsproba` : parler des probabilités facilement

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

Version `0.4.0-beta` développée et testée sur Mac OS X.

Christophe BAL

2020-07-31

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. Beta-dépendance

`\tnscom` qui est disponible sur `https://github.com/typensee-latex/tnscom.git` est un package utilisé en coulisse.

3. Packages utilisés

La roue ayant déjà été inventée, le package `tnsproba` réutilise les packages suivants sans aucun scrupule.

- `amsmath`
- `forest`

4. 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.

5. Généralités

a. Probabilité « simple »

Exemple 1

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

Exemple 2 – Choisir le nom de la probabilité

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

b. 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

```
 $\probacond  [P]{B}{A}
= \probacond* [P]{B}{A}
= \eprobacond*[P]{B}{A}
= \eprobacond [P]{B}{A}$
```

$$P_B(A) = P(A \mid B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A \cap B)}{P(B)}$$

c. Évènement contraire

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

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

$$\overline{A}$$

d. 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
$\var[v]{X}$
```

$$\mathrm{V}(X) \text{ ou } v(X)$$

Exemple 4 – Écart-type

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

```
$\stddev   {X}$ ou
$\stddev[s]{X}$
```

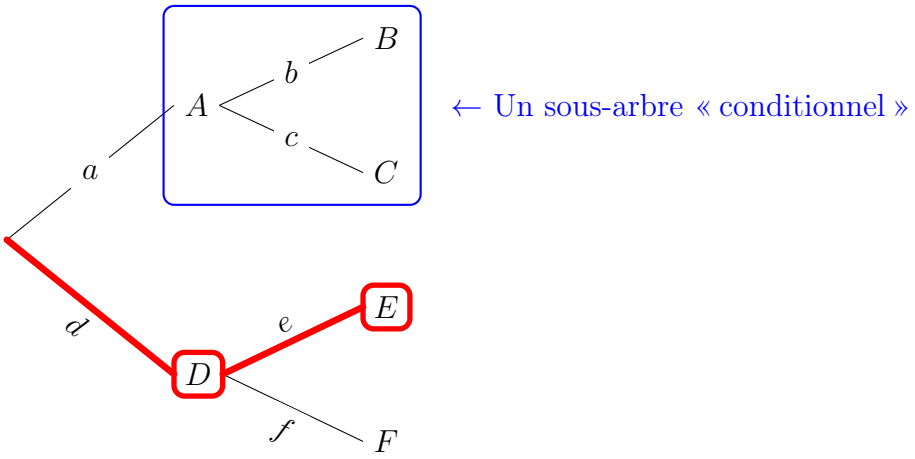
$$\sigma(X) \text{ ou } s(X)$$

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.

6. 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 sympathiques comme ci-dessous à moindre coût neuronal.



Le rendu précédent a été obtenu via le code suivant.

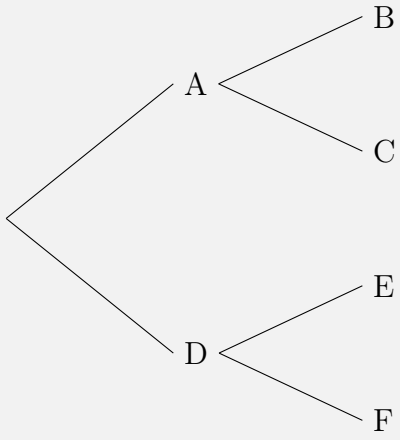
```
\begin{probatree}
    [{}, name = nU
        [$A$, pweight  = $a$,
              pframe   = blue,
              pcomment = \textcolor{blue}%
                                  {$\leftarrow$ Un sous-arbre \og conditionnel \fg}
            [$B$, pweight = $b$]
            [$C$, pweight = $c$]
        ]
        [$D$, name     = nD,
              bpweight = $d$
            [$E$, name     = nE,
                  apweight = $e$]
            [$F$, bpweight = $f$]
        ]
    ]
    \ptreeFocus*[red]{nU}{nD}
    \ptreeFocus*[red]{nD}{nE}
\end{probatree}
```

b. Les bases

Exemple 1 – Le cas type

Commençons par un arbre nu pour voir comment utiliser l'environnement `probatree` qui s'appuie en coulisse sur celui nommé `forest` du package éponyme. L'exemple qui suit utilise juste les réglages spécifiques de mise en forme de l'arbre qui sont propres à `probatree`.

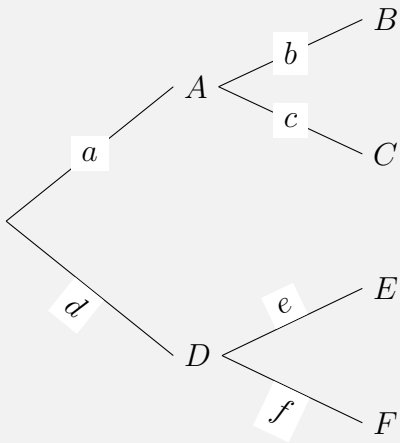
```
\begin{probatree}
    [  % Noeud racine sans texte
        [A  % Sous-noeud nommé
            [B]  % Sous-sous-noeud nommé
            [C]  % Sous-sous-noeud nommé
        ]
        [D  % Sous-noeud nommé
            [E]  % Sous-sous-noeud nommé
            [F]  % Sous-sous-noeud nommé
        ]
    ]
\end{probatree}
```



Exemple 2 – Ajouter des pondérations

Dans le code suivant, ce sont les styles spéciaux supplémentaires `pweight`, `apweight` et `bpweight` qui facilitent l'écriture des pondérations sur les branches [2].

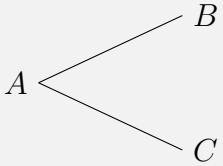
```
\begin{probatree}
    [
        [$A$, pweight = $a$
            [$B$, pweight = $b$]
            [$C$, pweight = $c$]
        ]
        [$D$, bpweight = $d$
            [$E$, apweight = $e$]
            [$F$, bpweight = $f$]
        ]
    ]
\end{probatree}
```



Exemple 3 – 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 *(ceci peut être utile lors de la rédaction d'exercices)*.

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

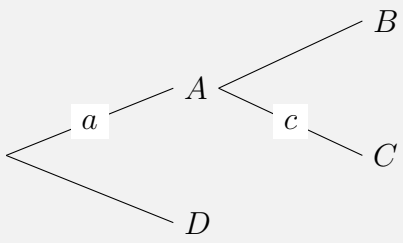


Exemple 4 – Des poids cachés localement

Pour ne cacher que certains poids afin de produire par exemple un arbre à compléter, il faudra utiliser localement le style `pweight*` comme dans l'exemple ci-dessous.

2. `pweight` vient de `p`-robability et `weight` soit *« probabilité »* et *« poids »* en anglais. Quant au `a` et au `b` au début de `apweight` et `bpweight` respectivement, ils viennent de `a`-bove et `b`-elow soit *« dessus »* et *« dessous »* en anglais.

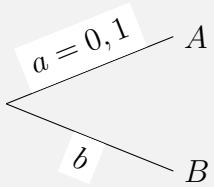
```
\begin{probatree}
    [
        [$A$, pweight = $a$
            [$B$, pweight* = $b$]
            [$C$, pweight  = $c$]
        ]
        [$D$, pweight* = $d$]
    ]
\end{probatree}
```



Exemple 5 – 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 [3]. Pour contourner cette limitation, il suffit de mettre le contenu de l'étiquette entre des accolades.

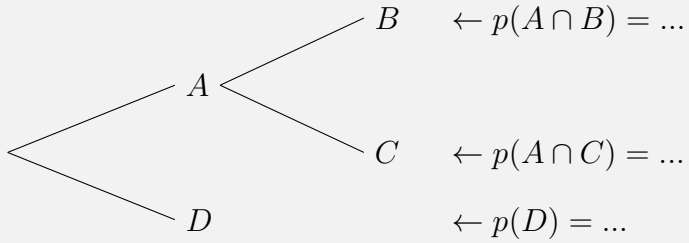
```
\begin{probatree}
    [
        [$A$, apweight = {$a = 0,1$}]
        [$B$, bpweight = $b$]
    ]
\end{probatree}
```



c. Commenter les racines

Que ce soit pour expliquer un arbre de probabilité, ou bien pour raisonner sur ce dernier, l'effet suivant est très utile. Noter qu'ici il a fallu utiliser des accolades pour cacher à TikZ les signes = dans les commentaires.

```
\begin{probatree}
    [
        [$A$
            [$B$, pcomment = {$\leftarrow \proba{A \cap B} = ...$}]
            [$C$, pcomment = {$\leftarrow \proba{A \cap C} = ...$}]
        ]
        [$D$, pcomment = {$\leftarrow \proba{D} = ...$}]
    ]
\end{probatree}
```



$\leftarrow p(A \cap B) = ...$

$\leftarrow p(A \cap C) = ...$

$\leftarrow p(D) = ...$

Remarque. Commenter un noeud interne ne provoquera pas d'erreur même si `pcomment` n'a pas été conçu pour ceci. Ceci a été utilisé dans l'exemple d'introduction mais ça reste un petit hack.

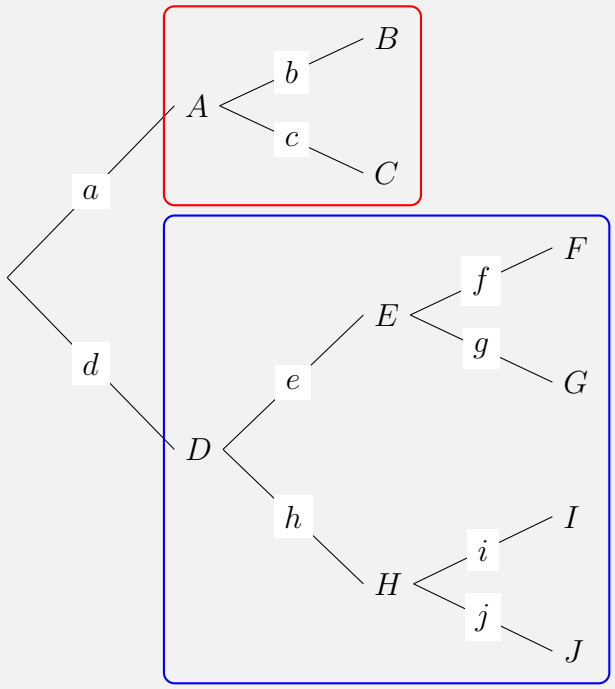
3. Ces deux symboles font partie de la syntaxe TikZ.

d. Avec des cadres

Exemple 1 – Des cadres finaux

Via la clé `pframe` il est très aisé d'encadrer un sous-arbre final [4] 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 = $a$,
              pframe  = red
            [$B$, pweight = $b$]
            [$C$, pweight = $c$]
        ]
        [$D$, pweight = $d$,
              pframe  = blue
            [$E$, pweight = $e$
                [$F$, pweight = $f$]
                [$G$, pweight = $g$]
            ]
            [$H$, pweight = $h$
                [$I$, pweight = $i$]
                [$J$, pweight = $j$]
            ]
        ]
    ]
\end{probatree}
```

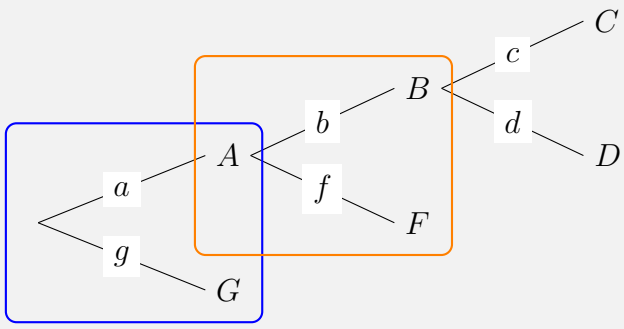


Exemple 2 – Des cadres non finaux

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 qui sont ajoutés via `name = ...` dans le code de l'arbre.

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

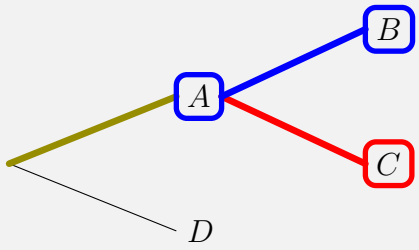
```
\begin{probatree}
    [{}, name = nU
     % La racine a un texte vide.
        [$A$, pweight = $a$,
              name    = nA
            [$B$, pweight = $b$,
                  name    = nB
                [$C$, pweight = $c$]
                [$D$, pweight = $d$]
              ]
            [$F$, pweight = $f$,
                  name    = nF]
        ]
        [$G$, pweight = $g$,
              name    = nG]
    ]
    \ptreeFrame         {nU}{nA}{nG}]
    \ptreeFrame[orange]{nA}{nB}{nF}]
\end{probatree}
```



e. Mettre en valeur des chemins

Il est relativement aisé de mettre en valeur un chemin particulier comme dans l'exemple ci-après qui est une simple démo. montrant les différences entre `\ptreeFocus`, `\ptreeFocus*` et `\ptreeFocus**`.

```
\begin{probatree}
    [{}, name = nU
        [$A$, name = nA
            [$B$, name = nB]
            [$C$, name = nC]
        ]
        [$D$]
    ]
    %
    \ptreeFocus* [red]  {nA}{nC}
    \ptreeFocus**[olive]{nU}{nA}
    \ptreeFocus         {nA}{nB}
\end{probatree}
```



Voici ce qu'il faut retenir.

1. `\ptreeFocus` encadre les noeuds initial et final.
2. `\ptreeFocus*` encadre juste le noeud final.
3. `\ptreeFocus**` n'encadre aucun des deux noeuds.
4. La couleur peut être changée via l'argument optionnel. Par défaut le bleu est utilisé.

7. Historique

Nous ne donnons ici qu'un très bref historique récent [5] 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`.

2020-07-31 Nouvelle version mineure `0.4.0-beta`.

- **Arbre :** possibilité de mettre en valeur un chemin via `\ptreeFocus`, `\ptreeFocus*` ou `\ptreeFocus**`.

2020-07-25 Nouvelle version mineure `0.3.0-beta`.

- **Arbre.**
  - Ajout du style `pcomment` pour placer du texte à la droite d'une feuille.
  - Le style `frame` a été renommé `pframe`.

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`.

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

8. 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}
   ...
\end{probatree}
\begin{probatree*}
```

```
   ...
\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 "pcomment":` pour ajouter un commentaire à la droite d'une feuille en utilisant le même alignement horizontal pour tous les commentaires.

— `Clé "pframe":` 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.

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

— `Option:` la couleur au format TikZ. La valeur par défaut est `blue`.

— `Argument 1:` nom du noeud initial.

— `Argument 2:` nom du noeud final.