Le package `tnsproba` : parler des probabilités facilement

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

Version `0.12.0-beta` développée et testée sur Mac OS X.

Christophe BAL

2020-09-28

Table des matières

I. Introduction

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

II. Beta-dépendance

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

III. Packages utilisés

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

- `forest`
- `nicematrix`
- `simplekv`
- `tcolorbox`
- `xstring`

IV. Ensembles probabilistes

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

V. Juste pour rédiger

1. Probabilité « simple »

Exemple 1

`$\proba{A}$`	$\mathrm{P}(A)$

Exemple 2 – Choisir le nom de la probabilité

`$\proba[p]{A}$`	$\mathrm{p}(A)$

2. Probabilité conditionnelle

Exemple 1 – Les deux écritures classiques

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

`$\probacond {B}{A}` `= \probacond*{B}{A}$`	$\mathrm{P}_{\mathrm{B}}(A) = \mathrm{P}(A \mid B)$

Exemple 2 – Obtenir la formule de définition

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

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.

```
 $\eprobacond {B}{A}
= \eprobacond*{B}{A}$
```

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

Exemple 3 – Choisir le nom de la probabilité

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

$$\mathrm{p}_{\mathrm{B}}(A) = \mathrm{p}(A \mid B) = \frac{\mathrm{p}(A \cap B)}{\mathrm{p}(B)} = \frac{\mathrm{p}(A \cap B)}{\mathrm{p}(B)}$$

3. Évènement contraire

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

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

$$\overline{A}$$

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

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

$$\mathrm{E}_1(X)$$

Exemple 3 – Variance

Notez la possibilité d'utiliser `\mathit` pour obtenir des lettres en italique.

```
$\var   {X}$ ou
$\var[v]{X}$ ou
$\var[\mathit{v}]{X}$
```

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

Exemple 4 – Écart-type

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

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

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

VI. Calculer l'espérance – Cas fini

Il est possible de définir facilement les valeurs d'une variable aléatoire finie (v.a.f.) et d'obtenir si souhaiter le calcul détaillé de son espérance [2].

1. Pour un usage direct – Notations par défaut

Dans la section 2. page 7 nous verrons qu'il est possible de définir une loi à un endroit puis de la réutiliser à d'autres [3]. Nous verrons aussi dans la section 3. page 8 que certaines notations sont modifiables.

Dans la présente section nous reproduirons à chaque fois les paramètres de la v.a.f. même si ceci produit des exemples un peu lourds à lire.

Exemple 1 – Loi d'une v.a.f.

La définition de la loi d'une v.a.f. avec la macro `\calcexpval` se fait avec une syntaxe semblable à celle d'un tableau. Notez dans l'exemple suivant que par défaut un tableau centré est affiché ce qui correspond à l'option par défaut `disp = table`. Ce n'est pas le seul comportement possible comme vont le montrer les autres exemples à venir.

```
\calcexpval{
    0      & 1   & 2   & 3    & 4    & 5   & 6   \\
    0.2000 & 0.1 & 0.2 & 0.05 & 0.15 & 0.1 & 0.2
}
```

x_k	0	1	2	3	4	5	6
p_k	0.2000	0.1	0.2	0.05	0.15	0.1	0.2

Remarque. Il est possible de ne rien afficher comme ci-dessous où `disp` est pour `disp`-lay soit *« afficher »* en anglais. Cette option sera utile pour définir une loi à un endroit et l'utiliser à d'autres.

```
\calcexpval[disp = none]{
    0      & 1   & 2   & 3    & 4    & 5   & 6   \\
    0.2000 & 0.1 & 0.2 & 0.05 & 0.15 & 0.1 & 0.2
}
```

Exemple 2 – Calculs expliqués et décorés de l'espérance d'une v.a.f.

La macro `\calcexpval` propose un système de clé valeur pour des réglages personnalisé. L'exemple ci-dessous montre comment obtenir un calcul détaillé et décoré de l'espérance.

```
\calcexpval[disp = all]{
    0      & 1   & 2   & 3    & 4    & 5   & 6   \\
    0.2000 & 0.1 & 0.2 & 0.05 & 0.15 & 0.1 & 0.2
}
```

2. Pour l'affichage une limitation importante est que le tableau et les calculs doivent tenir sur la largeur de la ligne. Concrètement pour un usage pédagogique cette limitation ne devrait jamais poser de problème.

3. La réutilisation se fera avec un argument vide de `\calcexpval`. C'est pour cela qu'une macro est proposée et non un environnement.

x_k	0	1	2	3	4	5	6
p_k	0.2000	0.1	0.2	0.05	0.15	0.1	0.2

$$\mathrm{E}(X) = \sum_{k=1}^{7} p_k \cdot x_k$$
$$\mathrm{E}(X) = 0.2000 \cdot 0 + 0.1 \cdot 1 + 0.2 \cdot 2 + 0.05 \cdot 3 + 0.15 \cdot 4 + 0.1 \cdot 5 + 0.2 \cdot 6$$

L'exemple suivant montre qu'il est facile de changer le couleurs qui sont utilisées de façon cyclique *(on peut indiquer une seule couleur)*.

```
\calcexpval[disp   = all,
            colors = orange - olive]{
    0      & 1   & 2   & 3    & 4    & 5   & 6   \\
    0.2000 & 0.1 & 0.2 & 0.05 & 0.15 & 0.1 & 0.2
}
```

x_k	0	1	2	3	4	5	6
p_k	0.2000	0.1	0.2	0.05	0.15	0.1	0.2

$$\mathrm{E}(X) = \sum_{k=1}^{7} p_k \cdot x_k$$
$$\mathrm{E}(X) = 0.2000 \cdot 0 + 0.1 \cdot 1 + 0.2 \cdot 2 + 0.05 \cdot 3 + 0.15 \cdot 4 + 0.1 \cdot 5 + 0.2 \cdot 6$$

Exemple 3 – Juste détailler les calculs de l'espérance d'une v.a.f.

Ci-dessous `exp` est pour `exp`-and soit *« développer »* en anglais. Cette option sera très utile lors de la réutilisation d'une loi définie précédemment.

```
\calcexpval[disp   = exp]{
    0      & 1   & 2   & 3    & 4    & 5   & 6   \\
    0.2000 & 0.1 & 0.2 & 0.05 & 0.15 & 0.1 & 0.2
}
```

$$\mathrm{E}(X) = \sum_{k=1}^{7} p_k \cdot x_k$$
$$\mathrm{E}(X) = 0.2000 \cdot 0 + 0.1 \cdot 1 + 0.2 \cdot 2 + 0.05 \cdot 3 + 0.15 \cdot 4 + 0.1 \cdot 5 + 0.2 \cdot 6$$

Exemple 4 – Des bouts du calcul

Lors de la réutilisation d'une loi définie précédemment les deux options `formal` et `eval` peuvent rendre service.

```
Somme formelle :
\calcexpval[disp = formal]{
    0      & 1   & 2   & 3    & 4    & 5   & 6   \\
    0.2000 & 0.1 & 0.2 & 0.05 & 0.15 & 0.1 & 0.2
}
```

```
Somme développée :
\calcexpval[disp = eval]{
    0      & 1   & 2   & 3    & 4    & 5   & 6   \\
    0.2000 & 0.1 & 0.2 & 0.05 & 0.15 & 0.1 & 0.2
}
```

Somme formelle :

$$\sum_{k=1}^{7} p_k \cdot x_k$$

Somme développée :

$$0.2000 \cdot 0 + 0.1 \cdot 1 + 0.2 \cdot 2 + 0.05 \cdot 3 + 0.15 \cdot 4 + 0.1 \cdot 5 + 0.2 \cdot 6$$

Exemple 5 – Calculs expliqués sans Σ

L'option `nosigma` permet de ne pas afficher le symbole Σ pour des raisons pédagogiques ou d'efficacité.

```
\calcexpval[disp = all,
            nosigma]{
    0      & 1   & 2   & 3    & 4    & 5   & 6   \\
    0.2000 & 0.1 & 0.2 & 0.05 & 0.15 & 0.1 & 0.2
}

\medskip

Juste les calculs :
\calcexpval[disp = exp,
            nosigma]{
    0      & 1   & 2   & 3    & 4    & 5   & 6   \\
    0.2000 & 0.1 & 0.2 & 0.05 & 0.15 & 0.1 & 0.2
} .
```

x_k	0	1	2	3	4	5	6
p_k	0.2000	0.1	0.2	0.05	0.15	0.1	0.2

$$\mathrm{E}(X) = \boxed{0.2000 \cdot 0} + 0.1 \cdot 1 + \boxed{0.2 \cdot 2} + 0.05 \cdot 3 + \boxed{0.15 \cdot 4} + 0.1 \cdot 5 + \boxed{0.2 \cdot 6}$$

Juste les calculs :

$$\mathrm{E}(X) = 0.2000 \cdot 0 + 0.1 \cdot 1 + 0.2 \cdot 2 + 0.05 \cdot 3 + 0.15 \cdot 4 + 0.1 \cdot 5 + 0.2 \cdot 6 \ .$$

2. Pour un usage différé – Notations par défaut

Rien de bien compliqué à utiliser comme le montre l'exemple suivant. Notez bien qu'il faut indiquer un argument vide à la macro `\calcexpval` lors de la réutilisation.

```
Voici ma loi de dingue.

\calcexpval[name = maloidedingue]{
    0      & 1   & 2   & 3    & 4    & 5   & 6   \\
    0.2000 & 0.1 & 0.2 & 0.05 & 0.15 & 0.1 & 0.2
}
```

```
Je dis des choses, bla, bla...

Et puis finalement j'indique le calcul de l'espérance.

\calcexpval[disp  = exp,
            reuse = maloidedingue]{}
```

Voici ma loi de dingue.

x_k	0	1	2	3	4	5	6
p_k	0.2000	0.1	0.2	0.05	0.15	0.1	0.2

Je dis des choses, bla, bla...
Et puis finalement j'indique le calcul de l'espérance.

$$\mathrm{E}(X) = \sum_{k=1}^{7} p_k \cdot x_k$$

$$\mathrm{E}(X) = 0.2000 \cdot 0 + 0.1 \cdot 1 + 0.2 \cdot 2 + 0.05 \cdot 3 + 0.15 \cdot 4 + 0.1 \cdot 5 + 0.2 \cdot 6$$

Remarque. Si cela vous convient, il est possible d'utiliser des noms de type « chemin de fichiers » comme ci-après [4].

```
Définissons de façon cachée la loi...
\calcexpval[disp = none,
            name = ma/loi/de/dingue]{
    0      & 1   & 2   & 3    & 4    & 5   & 6   \\
    0.2000 & 0.1 & 0.2 & 0.05 & 0.15 & 0.1 & 0.2
}

Bla, bla... Affichons la loi.

\calcexpval[disp  = table,
            reuse = ma/loi/de/dingue]{}
```

Définissons de façon cachée la loi...
Bla, bla... Affichons la loi.

x_k	0	1	2	3	4	5	6
p_k	0.2000	0.1	0.2	0.05	0.15	0.1	0.2

3. Notations personnalisées

Exemple 1 – Toutes les options

L'exemple ci-après utilise toutes les options liées à l'affichage textuel. Il faut noter que la clé `com` sert juste à ajouter un court texte sur une seule ligne [5].

```
\calcexpval[disp = all,
            E    = Esp, X  = Y,
```

4. Ceci permet d'utiliser des espaces de noms très pratiques à l'usage.
5. Si besoin utiliser séparément l'affichage par défaut avec `name = maloi` suivi d'un long texte puis enfin employer `disp = exp, reuse = maloi`.

```
            k    = i  , xk = y, pk = q,
            com  = Le tableau précédent aboutit aux calculs suivants.,
            ope  = \times]{
    0      & 1   & 2   & 3    & 4    & 5   & 6   \\
    0.2000 & 0.1 & 0.2 & 0.05 & 0.15 & 0.1 & 0.2
}
```

y_i	0	1	2	3	4	5	6
q_i	0.2000	0.1	0.2	0.05	0.15	0.1	0.2

Le tableau précédent aboutit aux calculs suivants.

$$\mathrm{Esp}(Y) = \sum_{i=1}^{7} q_i \times y_i$$

$$\mathrm{Esp}(Y) = 0.2000 \times 0 + 0.1 \times 1 + 0.2 \times 2 + 0.05 \times 3 + 0.15 \times 4 + 0.1 \times 5 + 0.2 \times 6$$

Exemple 2 – En réutilisant une loi définie précédemment

Lors de la création d'une loi nommée via la clé `name` les options textuelles, autre que le commentaire court, sont mémorisées. Ceci est très utile comme le montre le 1er affichage de l'exemple suivant tout en étant modifiable à tout moment comme dans le dernier affichage ci-après.

```
DÉFINITION CACHÉE.
\calcexpval[name = option/texte/loi,
            disp = none,
            E    = Esp, X  = Y,
            k    = i  , xk = y, pk = q,
            ope  = \times]{
    0      & 1   & 2   & 3    & 4    & 5   & 6   \\
    0.2000 & 0.1 & 0.2 & 0.05 & 0.15 & 0.1 & 0.2
}

\bigskip

RÉUTILISATION DIRECTE.
\calcexpval[reuse = option/texte/loi,
            disp  = all]{}

\bigskip

RÉUTILISATION AVEC DES OPTIONS MODIFIÉES (E, k et ope = $\cdot$).
\calcexpval[reuse = option/texte/loi,
            disp  = all,
            E     = E,
            k     = k,
            com   = Le tableau précédent aboutit aux calculs suivants.,
            ope   = \cdot]{}
```

DÉFINITION CACHÉE.

RÉUTILISATION DIRECTE.

y_i	0	1	2	3	4	5	6
q_i	0.2000	0.1	0.2	0.05	0.15	0.1	0.2

$$\mathrm{Esp}(Y) = \sum_{i=1}^{7} q_i \times y_i$$

$$\mathrm{Esp}(Y) = 0.2000 \times 0 + 0.1 \times 1 + 0.2 \times 2 + 0.05 \times 3 + 0.15 \times 4 + 0.1 \times 5 + 0.2 \times 6$$

RÉUTILISATION AVEC DES OPTIONS MODIFIÉES (E, k et ope = ·).

y_k	0	1	2	3	4	5	6
q_k	0.2000	0.1	0.2	0.05	0.15	0.1	0.2

Le tableau précédent aboutit aux calculs suivants.

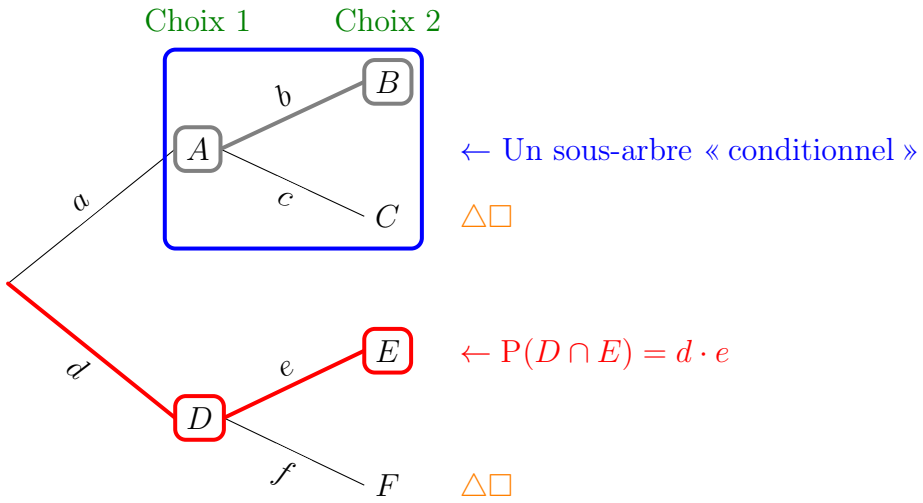
$$\mathrm{E}(Y) = \sum_{k=1}^{7} q_k \cdot y_k$$

$$\mathrm{E}(Y) = 0.2000 \cdot 0 + 0.1 \cdot 1 + 0.2 \cdot 2 + 0.05 \cdot 3 + 0.15 \cdot 4 + 0.1 \cdot 5 + 0.2 \cdot 6$$

VII. Arbres pondérés

1. Au commencement était la forêt...

Le gros du travail est fait par le package `forest` qui s'appuie `TikZ` dont on peut utiliser toute la machinerie afin d'obtenir des choses sympathiques comme ci-dessous et ceci à moindre coût neuronal comme vont le montrer les explications données dans les sections suivantes.



Le rendu précédent a été obtenu via le code suivant.

```
\begin{probatree}
    [{}, name = nU              % Noeud racine sans texte via {} mais avec un nom.
        [A, apweight = a,
            name     = nA,
            pframe   = blue    % Encadrement d'ici à la fin.
            [B, name     = nB,
                apweight = b]
            [C, name     = nC,
```

```
                bpweight = c]  % Pas besion de nommer ce noeud.
        ]
        [D, bpweight = d,
            name     = nD
            [E, apweight = e,
                name     = nE]
            [F, name     = nF,
                bpweight = f]  % Pas besion de nommer ce noeud.
        ]
    ]
    % Étiquettes pour les niveaux.
    \ptreeTagLevel[tcol = green!50!black, dy = .75mm]
                  {nA}{Choix 1}
    \ptreeTagLevel[tcol = green!50!black, dy = .75mm]
                  {nB}{Choix 2}
    % Mettre en valeur un chemin.
    \ptreeFocus[lcol = gray, frame = start]
               {nA | nB}
    % Mettre en valeur un chemin et le commenter.
    \ptreeComment[tcol = blue]
                 {nA}{$\leftarrow$ Un sous-arbre \og conditionnel \fg}
    % Comme avant.
    \ptreeFocus[lcol = red]
               {nU | nD | nE}
    \ptreeComment[tcol = red]
                 {nE}{$\leftarrow \proba{D \cap E} = d \cdot e$}
    % Décorer des feuilles.
    \ptreeTagLeaf[tcol = orange]
                  {nC | nF}{$\triangle \square$}
\end{probatree}
```

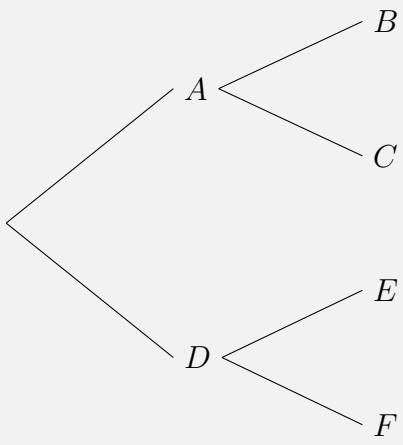
Remarque. Jusqu'à la section 9. page 27, nous nommerons à la main les noeuds des arbres via `name = ...` lorsque cela sera nécessaire. Dans la section indiquée nous verrons comment utiliser les noms automatiques donnés par le package `forest`.

2. 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`.

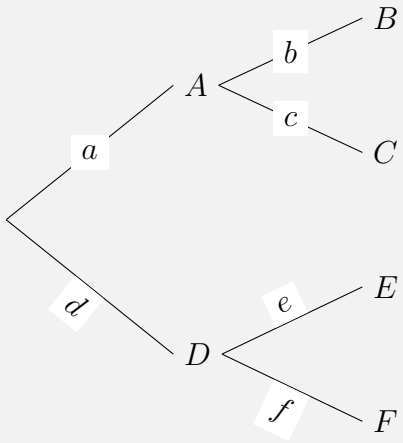
```
\begin{probatree}
    [           % Noeud racine sans texte
        [A      % Sous-noeud nommé
            [B] % Sous-sous-noeud nommé
            [C] % Sous-sous-noeud nommé
        ]
        [D      % Sous-noeud nommé
            [E] % Sous-sous-noeud nommé
            [F] % Sous-sous-noeud nommé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



Exemple 2 – Ajouter des pondérations

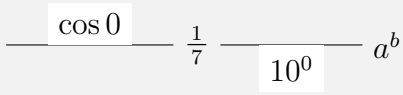
Dans le code suivant, ce sont les clés[6] `pweight`, `apweight` et `bpweight` qui facilitent l'écriture des pondérations sur les branches. Indiquons que `pweight` vient de `p`-robability et `weight` soit *« probabilité »* et *« poids »* en anglais. Quant au `a` et au `b` au début de `apweight` et `bpweight` 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, bpweight = d
            [E, apweight = e]
            [F, bpweight = f]
        ]
    ]
\end{probatree}
```



Remarque. Notez que le mode mathématique est activé par défaut pour les noms des noeuds et les poids comme le montre l'exemple improbable[7] ci-dessous.

```
\begin{probatree}
    [
        [\frac{1}{7}, apweight = \cos 0
            [a^b, bpweight = 10^0]
        ]
    ]
\end{probatree}
```



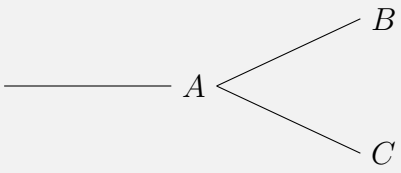
Exemple 3 – Des poids cachés partout

On peut cacher tous les poids sans avoir à les effacer partout dans le code LaTeX *(ceci peut être utile lors de la rédaction d'exercices)*. Il suffit pour cela d'utiliser une option `hideall` de l'environnement `probatree`. Comme les codes des arbres utilisent des crochets, l'option s'indique via `<hideall>` et

6. En fait du point de vue de TikZ, ce sont des styles.
7. Quoique...

non le traditionnel `[hideall]` [8]. L'option utilisée par défaut est `asit` qui demande de respecter les indications données pour les poids dans le code de l'arbre.

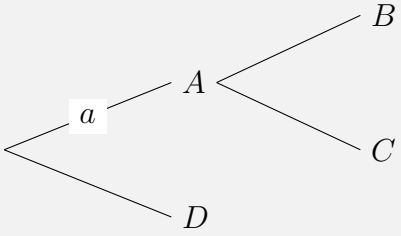
```
\begin{probatree}<hideall>
    [
        [A, pweight = a
            [B, apweight = b]
            [C, bpweight = c]
        ]
    ]
\end{probatree}
```



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.

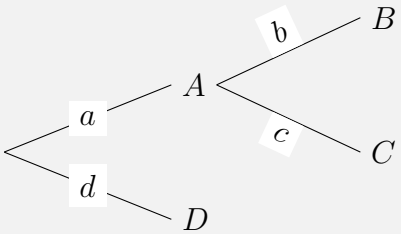
```
\begin{probatree}
    [
        [A, pweight = a
            [B, apweight* = b]
            [C, bpweight* = c]
        ]
        [D, pweight* = d]
    ]
\end{probatree}
```



Exemple 5 – Forcer l'affichage des poids cachés localement

Imaginons que nous voulions donner l'arbre précédent avec tous ses poids visibles pour une correction par exemple. Il suffit dans ce cas de passer via l'option `showall` de l'environnement `probatree` qui affichera absolument tous les poids *(on tape une fois et on réutilise le même code dans deux contextes différents)*.

```
\begin{probatree}<showall>
    [
        [A, pweight = a
            [B, apweight* = b]
            [C, bpweight* = c]
        ]
        [D, pweight* = d]
    ]
\end{probatree}
```



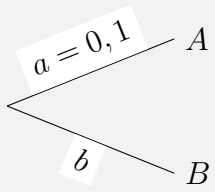
Exemple 6 – Un signe = et/ou une virgule dans les étiquettes

Vous ne pouvez pas utiliser directement un signe = ou une virgule dans les étiquettes des branches [9]. Pour contourner cette limitation, il suffit de mettre le contenu de l'étiquette entre des accolades.

8. Ceci a été rendu possible grâce à un code proposé dans l'excellent livre *« Apprendre à programmer en TEX »* de Christian Tellechea.

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

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

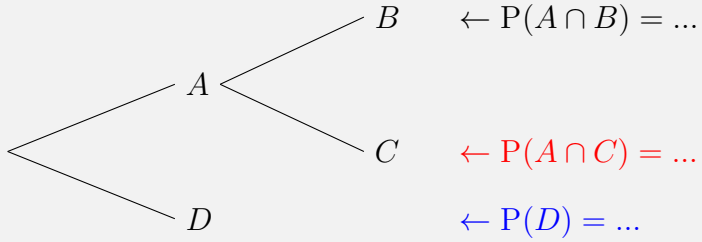


3. Commenter les feuilles

Exemple 1 – Tout aligner

Que ce soit pour expliquer un arbre de probabilité, ou bien pour raisonner sur ce dernier, l'effet suivant est très utile [10]. Noter l'utilisation possible de la clé `tcol` pour `t`-ext et `col`-or afin d'indiquer la couleur du texte au format TikZ. La couleur par défaut est le noir.

```
\begin{probatree}
    [
        [A
            [B, name = nB]
            [C, name = nC]
        ]
        [D, name = nD]
    ]
    %
    \ptreeComment              {nB}{$\leftarrow \proba{A \cap B} = ...$}
    \ptreeComment[tcol = red] {nC}{$\leftarrow \proba{A \cap C} = ...$}
    \ptreeComment[tcol = blue]{nD}{$\leftarrow \proba{D} = ...$}
\end{probatree}
```

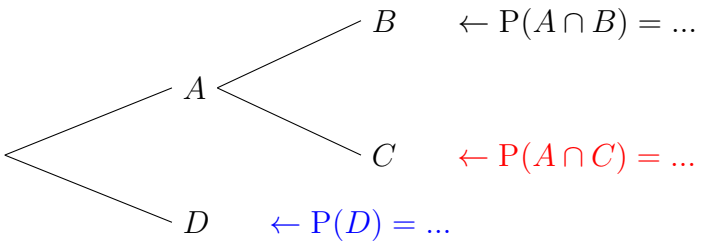


Remarque. Commenter un noeud interne ne provoquera pas d'erreur même si `\ptreeComment` n'a pas été conçu pour ceci. Ceci a été utilisé dans l'exemple d'introduction mais ça reste un petit hack.

Exemple 2 – Coller au plus près

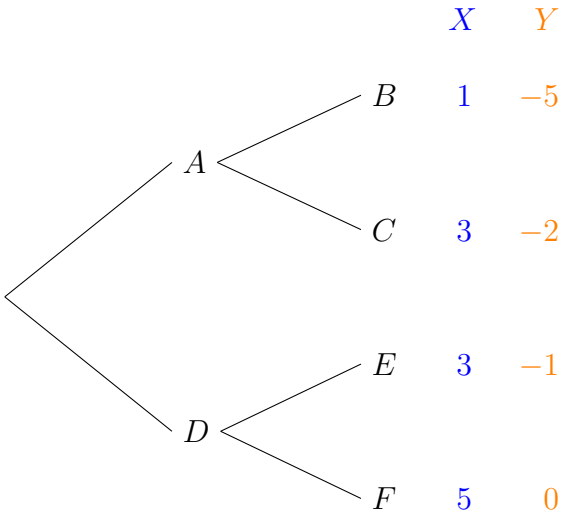
En utilisant `\ptreeComment*` au lieu de `\ptreeComment`, les commentaires seront proches des noeuds et donc non alignés verticalement. Avec l'exemple précédent on obtient la mise en forme qui suit.

10. Le package `forest` permet d'indiquer directement des mises en forme dans le code de l'arbre. L'auteur du présent package trouve bien plus efficace à l'usage de ne pas toucher au code minimal d'un arbre. Ceci explique donc le choix retenu de donner les décorations supplémentaires après le code de l'arbre.



Exemple 3 – Décalages horizontal et vertical – Avec des variables aléatoires

Grâce aux clés `dx` et `dy` il est possible d'ajouter des décalages horizontal et vertical. Ceci permet d'obtenir ce qui suit sans trop se fatiguer les méninges.



Le code utilisé est le suivant. Notez qu'ici il y a des réglages à faire au doigt mouillé. Dans l'exemple suivant nous allons voir comment se passer des horribles copier-coller.

```
\begin{probatree}
    % ------------------------------------------- %
    % -- Code de l'arbre seul non reproduit ici -- %
    % ------------------------------------------- %
    % Valeurs de X
    \ptreeComment[tcol = blue,
                  dx   = -.25em,
                  dy   = 1cm]{nB}{$X$}
    %
    \ptreeComment[tcol = blue]{nB}{$1$}
    \ptreeComment[tcol = blue]{nC}{$3$}
    \ptreeComment[tcol = blue]{nE}{$3$}
    \ptreeComment[tcol = blue]{nF}{$5$}
    %
    % Valeurs de Y
    \ptreeComment[tcol = orange,
                  dx   = 2em+.5em,
                  dy   = 1cm]{nB}{$Y$}
    %
    \ptreeComment[tcol = orange,
                  dx   = 2em]{nB}{$-5$}
    \ptreeComment[tcol = orange,
                  dx   = 2em]{nC}{$-2$}
```

```
    \ptreeComment[tcol = orange,
                  dx   = 2em]{nE}{$-1$}
    \ptreeComment[tcol = orange,
                  dx   = 2em]{nF}{$\phantom{-}0$}
\end{probatree}
```

Exemple 4 – Commenter via une boucle – Avec des variables aléatoires

Dans le code précédent nous avons dû faire des copier-coller. La macro `\foreach` de TikZ permet d'éviter cela afin d'obtenir un code très facile à maintenir et à comprendre comme celui ci-après. Ceci étant indiqué, il y a des pièges à éviter comme nous l'expliquons juste après.

```
\begin{probatree}
    % ------------------------------------------- %
    % -- Code de l'arbre seul non reproduit ici -- %
    % ------------------------------------------- %
    % Valeurs de X
    \ptreeComment[tcol = blue,
                  dx = -.25em, dy = 1cm] {nB}{$X$}
    %
    \foreach \name/\val in {
        nB/$1$,
        nC/$3$,
        nE/$3$,
        nF/$5$
    }{
        \ptreeComment[tcol = blue]{\name}{\val}
    }
    % Valeurs de Y
    \ptreeComment[tcol = orange,
                  dx   = 2em+.5em,
                  dy   = 1cm]{nB}{$Y$}
    %
    \foreach \name/\val in {
        nB/$-5$,
        nC/$-2$,
        nE/$-1$,
        nF/$\phantom{-}0$
    }{
        \ptreeComment[tcol = orange,
                      dx   = 2em]{\name}{\val}
    }
\end{probatree}
```

Voici les pièges à éviter.

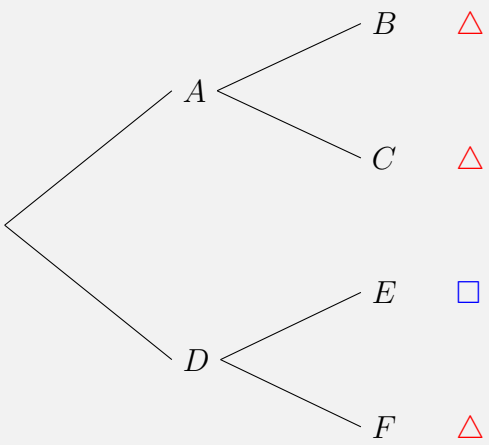
1. `\foreach` ignore les espace initiaux mais pas les finaux. Si vous utilisez `nB /$1$` au lieu de `nB/$1$` alors la macro croira que le nom se finit par un espace et `forest` ne pourra rien faire.
2. Comme les noms `\color`, `\tcol`, `\dx` et `\dy` sont utilisés en coulisse, il n'est pas possible de les utiliser dans les boucles.

4. Décorer les feuilles

Il peut être utile de décorer de la même façon différents chemins pour indiquer des évènements étudiés. C'est la raison d'être de `\ptreeTagLeaf` et `\ptreeTagLeaf*` qui produisent le même commentaire pour différents noeuds. Notez dans les exemples que les noms des noeuds sont séparés par le symbole `|` avec la possibilité d'ajouter des espaces pour améliorer la lisibilité du code.

Exemple 1 – Décorer au même niveau

```
\begin{probatree}
    [
        [A
            [B, name = nB]
            [C, name = nC]
        ]
        [D
            [E, name = nE]
            [F, name = nF]
        ]
    ]
    %
    \ptreeTagLeaf[tcol = red]
                 {nB | nC | nF}{$\triangle$}
    \ptreeTagLeaf[tcol = blue]
                 {nE}{$\square$}
\end{probatree}
```

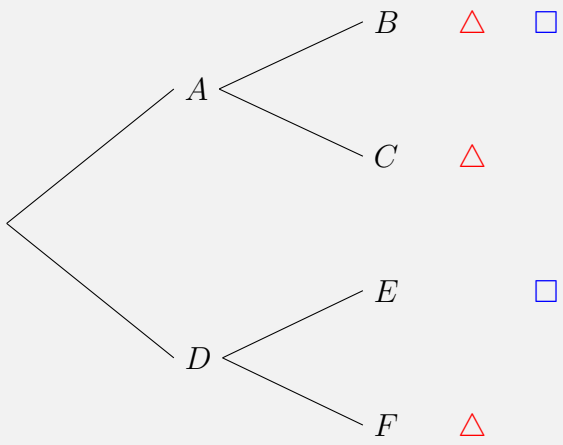


Exemple 2 – Décorer sur des niveaux décalés

La démo. suivante n'utilise que `dx` mais bien entendu `dy` est aussi disponible.

```
\begin{probatree}
    [
        [A
            [B, name = nB]
            [C, name = nC]
        ]
        [D
```

```
            [E, name = nE]
            [F, name = nF]
        ]
    ]
    %
    \ptreeTagLeaf[tcol = red]
                 {nB | nC | nF}{$\triangle$}
    \ptreeTagLeaf[tcol = blue,
                  dx   = 1cm]
                 {nB | nE}{$\square$}
\end{probatree}
```

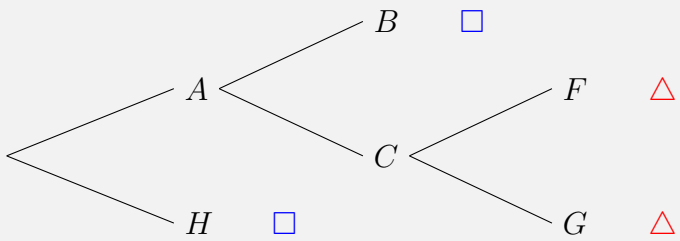


Exemple 3 – Décorer au plus près

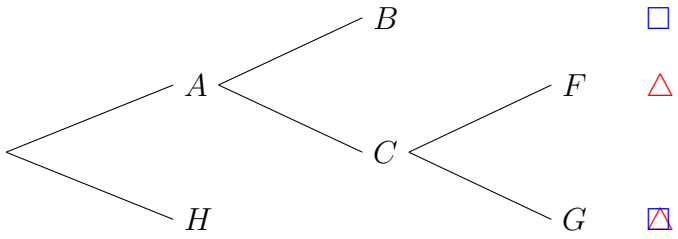
Voici un exemple avec un arbre dissymétrique[11].

```
\begin{probatree}
    [
        [A
            [B, name = nB]
            [C,
                [F, name = nF]
                [G, name = nG]
            ]
        ]
        [H, name = nH]
    ]
    \ptreeTagLeaf*[tcol = red]
                  {nF | nG}{$\triangle$}
    \ptreeTagLeaf*[tcol = blue]
                  {nB | nH}{$\square$}
\end{probatree}
```

11. Est-ce vraiment pertinent comme usage ?

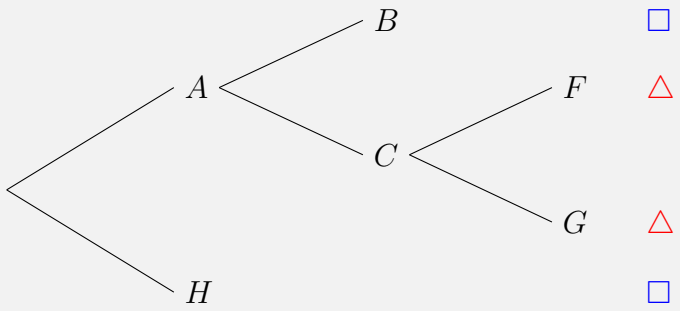


Remarque. Sans utiliser la version étoilée de `\ptreeTagLeaf`, on obtient ce qui suit qui est mauvais.



Le package `forest` propose `fit = band` qui est très utile dans ce type de situation. Voici comment utiliser ceci.

```
\begin{probatree}
    [, s sep = 0.35cm      % <-- Un peu de design aux doigts mouillés.
        [A
            [B, name = nB]
            [C,
                [F, name = nF]
                [G, name = nG]
            ]
        ]
        [H, name = nH,
            fit  = band]  % <-- Pour placer H seul sur son niveau.
    ]
    \ptreeTagLeaf[tcol = red]
                 {nF | nG}{$\triangle$}
    \ptreeTagLeaf[tcol = blue]
                 {nB | nH}{$\square$}
\end{probatree}
```

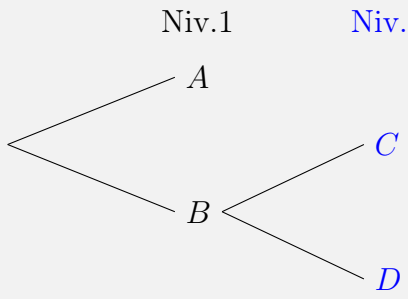


5. Expliquer les niveaux

Grâce aux deux sections précédentes les exemples devraient se suffire à eux-mêmes pour comprendre les fonctionnements de `\ptreeTagLevel` et `\ptreeTagLevel*`.

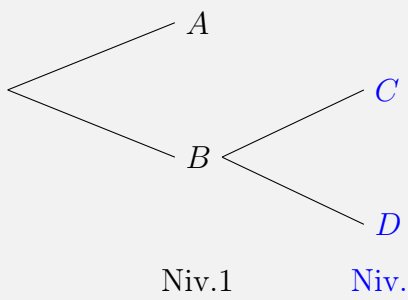
Exemple 1 – Expliquer au-dessus

```
\begin{probatree}
    [
        [A, name = nA]
        [B
            [C, name = nC,
                blue]
            [D, blue]
        ]
    ]
    %
    \ptreeTagLevel            {nA}{Niv.1}
    \ptreeTagLevel[tcol=blue]{nC}{Niv.2}
\end{probatree}
```



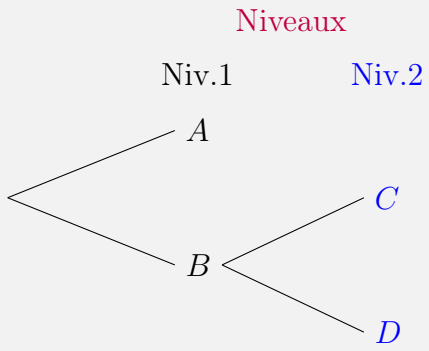
Exemple 2 – Expliquer en dessous

```
\begin{probatree}
    [
        [A, name = nA]
        [B
            [C, name = nC,
                blue]
            [D, blue]
        ]
    ]
    %
    \ptreeTagLevel*            {nA}{Niv.1}
    \ptreeTagLevel*[tcol=blue]{nC}{Niv.2}
\end{probatree}
```



Exemple 3 – Décalages horizontal et vertical

```
\begin{probatree}
    [
        [A, name = nA]
        [B
            [C, name = nC,
                blue]
            [D, blue]
        ]
    ]
    %
    \ptreeTagLevel[tcol = purple,
                   dx   = 3em,
                   dy   = 1.75em]{nA}{Niveaux}
    %
    \ptreeTagLevel           {nA}{Niv.1}
    \ptreeTagLevel[tcol=blue]{nC}{Niv.2}
\end{probatree}
```

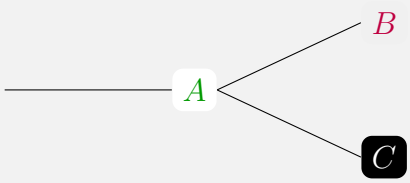


6. Textes des noeuds

Exemple 1 – Changer les couleurs

On peut a posteriori changer les couleurs du texte et du fond d'un noeud via `\ptreeNodeColor` présentée ci-dessous.

```
\begin{probatree}
    [
        [A, name = nA
            [B, name = nB]
            [C, name = nC]
        ]
    ]
    \ptreeNodeColor{nA}{tcol = green!60!black}
    \ptreeNodeColor{nB}{tcol = purple,
                        bcol = black!6!white}
    \ptreeNodeColor{nC}{tcol = white,
                        bcol = black}
\end{probatree}
```



Voici ce qu'il faut retenir du code précédent.

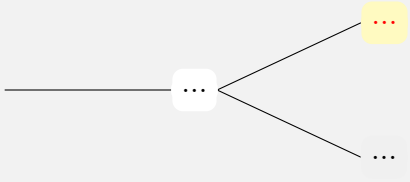
1. Par défaut le blanc sert de couleur de fond. Ceci se voit dans la mise en forme du noeud A pour lequel `tcol` change juste la couleur du texte qui par défaut sera le noir.
2. Pour changer la couleur de fond, on passe via `bcol`. Ceci permet d'avoir le rendu souhaité pour la mise en forme du noeud B [12].
3. Les préfixes `t` et `b` de `bcol` et `tcol` sont pour `t`-exte et `b`-ackground, ce dernier mot signifiant *« fond »* en anglais. Quant à `col` c'est pour `col`-or soit *« couleur »* en anglais.

Remarque. Techniquement toutes les macros présentées dans cette section cachent l'ancien texte d'un noeud par superposition de ce texte en utilisant une couleur identique pour le texte et le fond.

Exemple 2 – Changer le texte

On peut a posteriori changer le texte d'un noeud, avec un choix des couleurs, via `\ptreeNodeNewText` présentée ci-après. Noter que les couleurs par défaut du texte et du fond restent le noir et le blanc respectivement.

```
\begin{probatree}
    [
        [A, name = nA
            [B, name = nB]
            [C, name = nC]
        ]
    ]
    \ptreeNodeNewText{nA}{...}
    \ptreeNodeNewText[tcol = red,
                      bcol = yellow!30!white]%
                     {nB}{...}
    \ptreeNodeNewText[bcol = black!6!white]%
                     {nC}{...}
\end{probatree}
```



Exemple 3 – Récupérer le texte d'un noeud

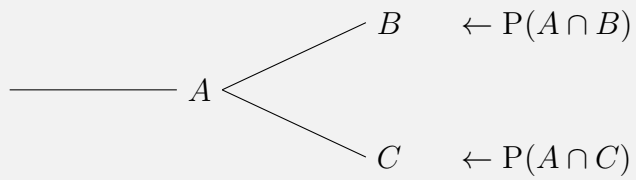
Considérons l'arbre un peu plat suivant.

12. Ainsi le fond du noeud et celui du cadre ont la même couleur.

```
\begin{probatree}
    [
        [texte.de.A, name = nA
            [B]
        ]
    ]
\end{probatree}
```

Une fois ce code inséré il est possible de récupérer *texte.de.A* juste en tapant `\ptreeTextOf{nA}`. Voici un exemple concret [13] d'utilisation.

```
\begin{probatree}
    [
        [A, name = nA
            [B, name = nB]
            [C, name = nC]
        ]
    ]
    %
    \ptreeComment{nB}%
                 {$\leftarrow \proba{\ptreeTextOf{nA} \cap \ptreeTextOf{nB}}$}
    \ptreeComment{nC}%
                 {$\leftarrow \proba{\ptreeTextOf{nA} \cap \ptreeTextOf{nC}}$}
\end{probatree}
```



Remarque. Comme la macro `\ptreeNodeNewText` utilise le dernier noeud rencontré, il faudra veiller à ne pas vouloir utiliser les textes de noeuds présents dans deux arbres différents et qui ont en même temps le même nom.

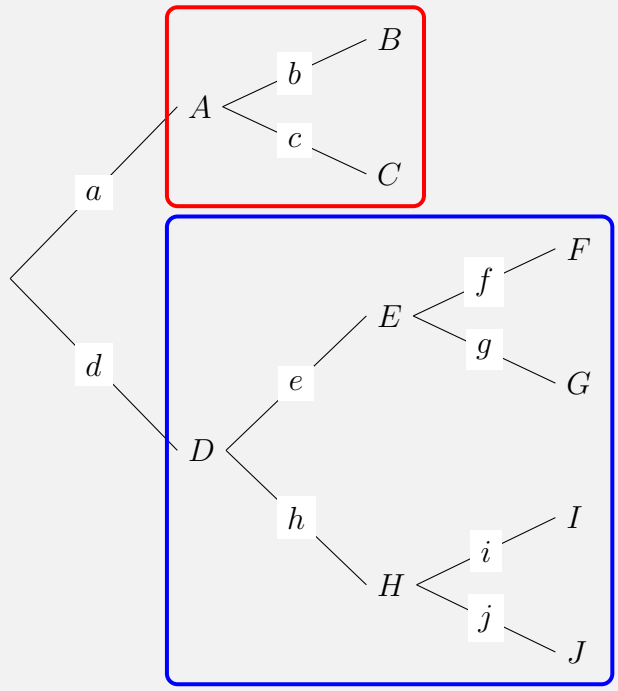
7. Avec des cadres

Exemple 1 – Des cadres finaux

Via la clé `pframe` il est très aisé d'encadrer un sous-arbre final [14] comme le montre l'exemple suivant [15]. Dans l'exemple ci-après nous utilisons la bidouille `{},s sep = 1.3cm` qui évite que les cadres se superposent.

13. Mais pas forcément pertinent... L'exemple peut être intéressant dans le cadre de contenus produits de façon automatisée.
14. Un sous-arbre sera dit final si toutes ses feuilles correspondent à des feuilles de l'arbre initial.
15. Ce type de cadre est très utile d'un point de vue pédagogique.

```
\begin{probatree}
    [{}, s sep = 1.3cm
     % Espacment pour éviter que
     % les cadres se superposent.
        [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}
```

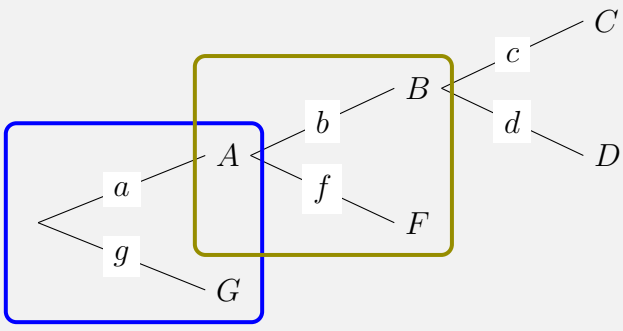


Remarque. La clé `pframe` est un cas particulier de décoration car les autres décorations se font en dehors de la définition de l'arbre

Exemple 2 – Des cadres non finaux

La macro `\ptreeFrame` permet facilement d'encadrer un sous-arbre non final. Ceci nécessite d'utiliser des noms de noeuds. 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.

```
\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[lcol = olive]{nA}{nB}{nF}
\end{probatree}
```

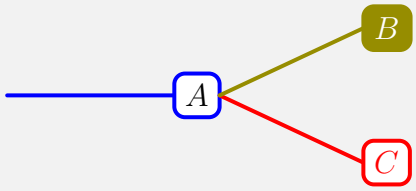


8. Mettre en valeur des chemins

Exemple 1 – Avec deux noeuds – Choix des couleurs

La macro `\ptreeFocus` rend facile la mise en valeur d'un chemin particulier comme le montre l'exemple ci-après qui est une simple démo. Notez que les noms des noeuds sont séparés par des barres verticales `|` et qu'il est possible d'utiliser des espaces pour améliorer la lisibilité du code.

```
\begin{probatree}
    [{}, name = nU
        [A, name = nA
            [B, name = nB]
            [C, name = nC]
        ]
    ]
    %
    \ptreeFocus{nU | nA}
    \ptreeFocus[tcol = red,
                lcol = red]{nA | nC}
    \ptreeFocus[tcol = white,
                bcol = olive,
                lcol = olive]{nA | nB}
\end{probatree}
```



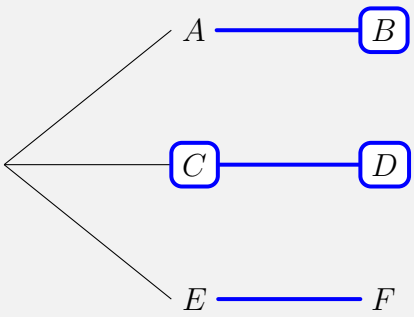
Voici ce qu'il faut retenir pour les couleurs qui doivent être du type TikZ.

1. `lcol` sert à indiquer la couleur des arêtes et des cadres éventuels. Par défaut `lcol = blue`. Indiquons que le préfixe `l` est pour `l`-igne.
2. `tcol` sert à indiquer la couleur du texte. Par défaut `lcol = black`. Indiquons que le préfixe `t` est pour `t`-exte.
3. `bcol` sert à indiquer la couleur du fond. Par défaut `lcol = white`. Indiquons que le préfixe `b` est pour `b`-ackground soit *« fond »* en anglais.

Exemple 2 – Avec deux noeuds – Choix des cadres

Par défaut le 1er noeud n'est pas encadré car il est courant de vouloir indiquer un noeud partant de la racine qui traditionnellement ne contient aucun texte. Il est possible d'obtenir deux autres mises en forme comme ci-après.

```
\begin{probatree}
    [
        [A, name = nA
            [B, name = nB]
        ]
        [C, name = nC
            [D, name = nD]
        ]
        [E, name = nE
            [F, name = nF]
        ]
    ]
    %
    \ptreeFocus                 {nA | nB}
    \ptreeFocus[frame = start]{nC | nD}
    \ptreeFocus[frame = none] {nE | nF}
\end{probatree}
```



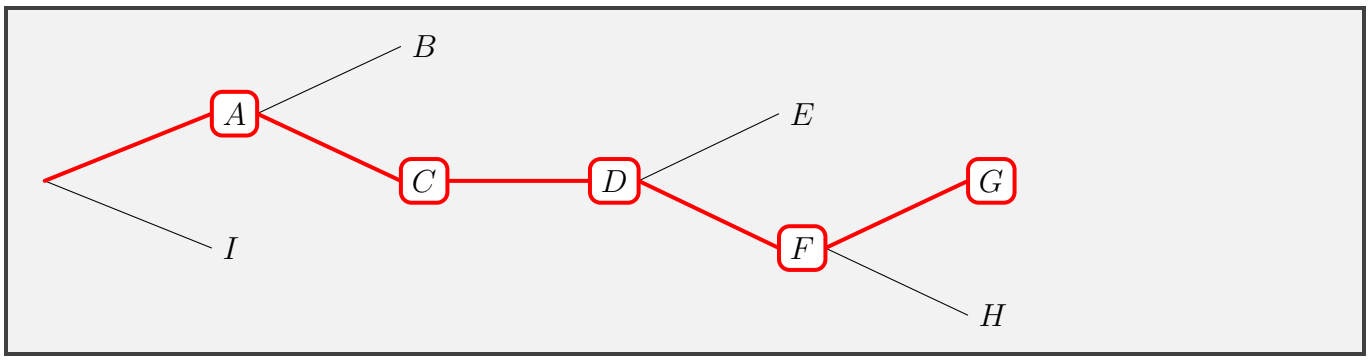
Voici ce qu'il faut retenir à propos des encadrements.

1. `frame = nostart`, le réglage par défaut, demande d'encadrer tous les noeuds sauf le 1er.
2. `frame = start` demande d'encadrer tous les noeuds.
3. `frame = none` demande de n'encadrer aucun noeud.

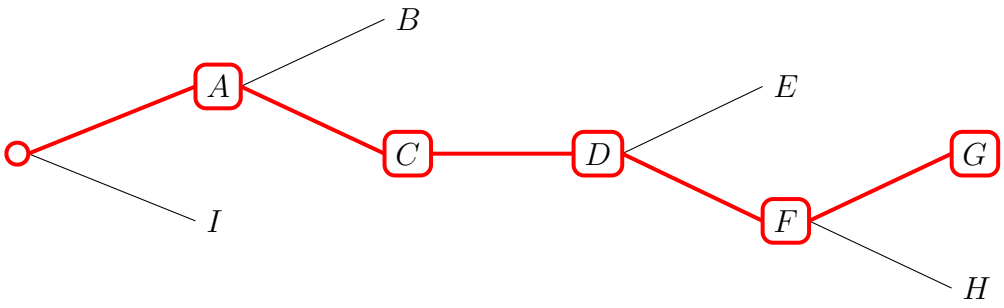
Exemple 3 – Plusieurs noeuds d'un coup

Rien de bien compliqué à condition de bien respecter l'ordre de saisie des noeuds.

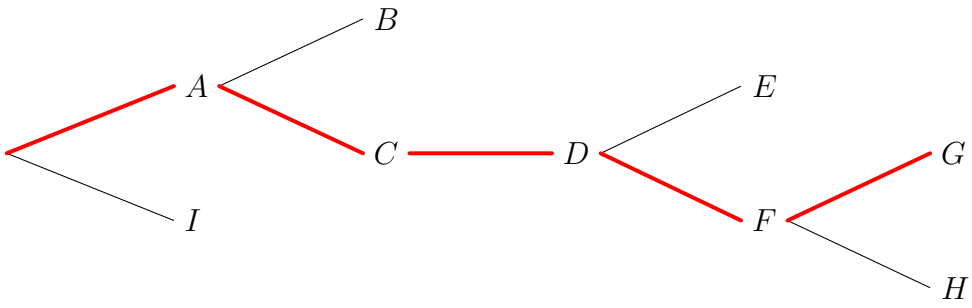
```
\begin{probatree}
    [{}, name = nU
        [A, name = nA
            [B]
            [C, name = nC
                [D, name = nD
                    [E]
                    [F, name = nF
                        [G, name = nG]
                        [H]
                    ]
                ]
            ]
        ]
        [I]
    ]
    %
    \ptreeFocus[lcol = red]{nU | nA | nC | nD | nF | nG}
\end{probatree}
```



Avec `frame = start` on obtient l'arbre suivant où le mini disque initial [16] n'est pas forcément souhaité.

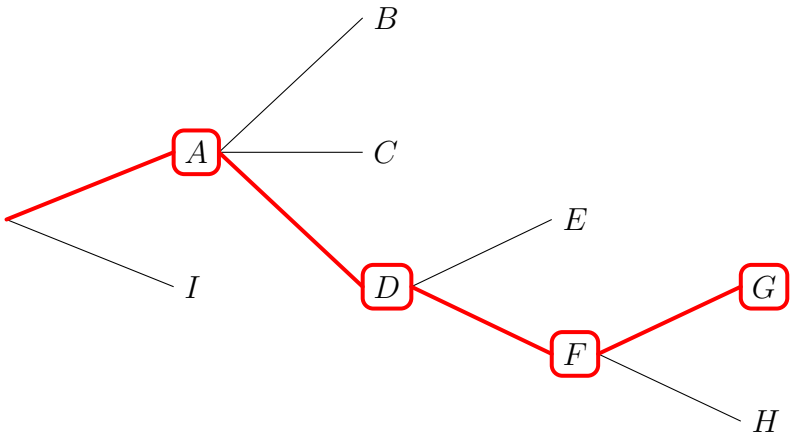


Avec `frame = none` on obtient l'arbre ci-dessous.



9. Utiliser les noms automatiques donnés par `forest`

Voyons comment obtenir le résultat suivant en indiquant tous les noeuds via les noms automatiques fabriqués par `forest`.



Le rendu précédent a été obtenu via le code suivant.

16. Ce disque est en fait un carré aux coins arrondis autour d'un texte vide.

```
\begin{probatree}
    [
        [A
            [B]
            [C]
            [D
                [E]
                [F
                    [G]
                    [H]
                ]
            ]
        ]
        [I]
    ]
    %
    \aptreeFocus[lcol = red]{ | 1 | 13 | 132 | 1321}
\end{probatree}
```

Voici comment s'y prendre.

1. On utilise `\aptreeFocus` au lieu de `\ptreeFocus` où le préfixe `a` est pour `a`-uto.

   ATTENTION ! *Il n'est pas possible de modifier les couleurs du texte et du fond des noeuds car* `\aptreeTextOf`*,* `\aptreeNodeColor` *et* `\aptreeNodeNewText` *n'ont pas pu être implémentées. Par contre les macros* `\aptreeComment` *et* `\aptreeFrame` *ainsi que* `\aptreeTagLevel` *et* `\aptreeTagLeaf` *existent sans limitation.*

2. Chaque nom automatique[17] fabriqué par `forest` commence par `!` . Ce caractère spécial n'est pas à indiquer car il sera ajouté automatiquement en coulisse.
3. La racine est nommée `!` par `forest` d'où le `|` seul au début de l'argument de `\aptreeFocus*` ci-dessus afin d'indiquer un texte vide comme nom du tout premier noeud.
4. Pour voir ce qu'il faut faire pour un noeud autre que la racine, considérons par exemple `1321`. On indique en fait le chemin à suivre après la racine pour arriver au noeud voulu.

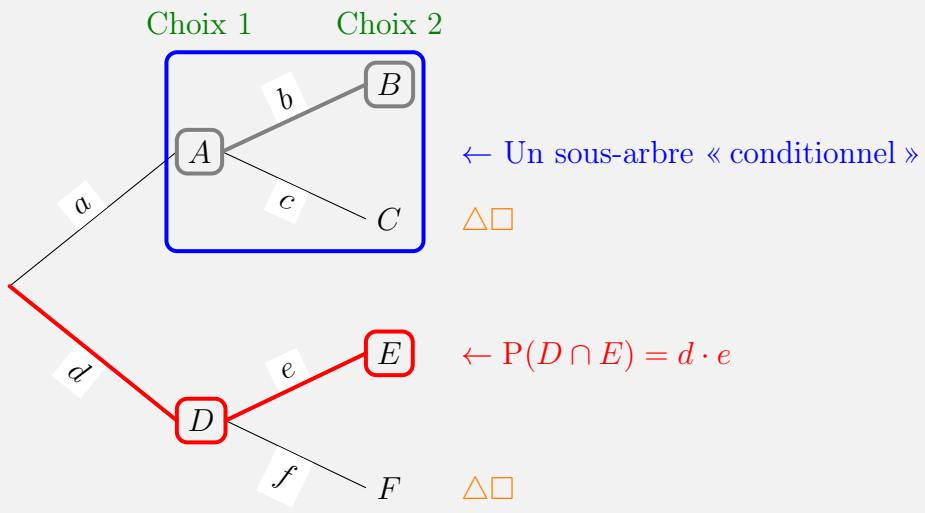
   - Aller d'abord au $\boxed{1}^{\text{er}}$ noeud du niveau 1 qui ici est A.
   - Aller ensuite au $\boxed{3}^{\text{e}}$ noeud du niveau 2 qui ici est D.
   - Aller après au $\boxed{2}^{\text{e}}$ noeud du niveau 3 qui ici est F.
   - Aller enfin au $\boxed{1}^{\text{er}}$ noeud du niveau 4 qui ici est G.
   - On obtient finalement notre noeud nommé $\boxed{1321}$.

Remarque. Dans la mesure du possible, utiliser les noms automatiques facilite la maintenance des arbres sur le long terme. Si on reprend le tout premier exemple d'arbre décoré, il est bien plus simple de faire comme suit car on ne touche pas à la structure minimale du code de l'arbre. On a même utilisé `\ptreeFrame` au lieu de la clé `pframe` dans l'arbre.

```
\begin{probatree}
    [
        [A, apweight = a
            [B, apweight = b]
            [C, bpweight = c]
        ]
```

17. Ce sont en fait des noms relativement à la racine de l'arbre.

```
        [D, bpweight = d
            [E, apweight = e]
            [F, bpweight = f]
        ]
    ]
    % Étiquettes pour les niveaux.
    \aptreeTagLevel[tcol = green!50!black, dy = .75mm]
                   {1}{Choix 1}
    \aptreeTagLevel[tcol = green!50!black, dy = .75mm]
                   {11}{Choix 2}
     % Mettre en valeur un chemin.
    \aptreeFocus[lcol = gray,frame = start]
                {1 | 11}
   % Mettre en valeur un chemin et le commenter.
    \aptreeFrame{1}{11}{12}
    \aptreeComment[tcol = blue]%
                  {1}{$\leftarrow$ Un sous-arbre \og conditionnel \fg}
    % Comme avant.
    \aptreeFocus[lcol = red]
                { | 2 | 21}
    \aptreeComment[tcol = red]
                  {21}{$\leftarrow \proba{D \cap E} = d \cdot e$}
     % Décorer des feuilles.
    \aptreeTagLeaf[tcol = orange]
                  {12 | 22}{$\triangle \square$}
\end{probatree}
```



← Un sous-arbre « conditionnel »

△□

← $\mathrm{P}(D \cap E) = d \cdot e$

△□

VIII. Historique

Nous ne donnons ici qu'un très bref historique récent [18] 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`.

2020-09-28 Nouvelle version mineure `0.12.0-beta`.

- **Arbre de probabilités.**
  - Les macros `\ptreeTag`, `\ptreeTag*`, `\aptreeTag` et `\aptreeTag*` ont été renommées `\ptreeTagLevel`, `\ptreeTagLevel*`, `\aptreeTagLevel` et `\aptreeTagLevel*`.
  - `\ptreeTagLeaf`, `\ptreeTagLeaf*`, `\aptreeTagLeaf` et `\aptreeTagLeaf*` permettent de décorer des racines avec le même texte.

2020-09-23 Nouvelle version mineure `0.11.0-beta`.

- **Arbre de probabilités :** ajout de `\ptreeTag`, `\ptreeTag*`, `\aptreeTag` et `\aptreeTag*` pour décrire individuellement les niveaux de l'arbre.

2020-09-03 Nouvelle version mineure `0.10.0-beta`.

- **Généralités.**
  - P est le nom par défaut d'une probabilité.
  - Les noms des probabilités, des espérances, des variances et des écarts-types utilisent tous une police droite via `\mathrm` en coulisse.
- **Espérance d'une variable aléatoire finie.**
  - `\expval` est utilisée en coulisse pour rédiger les espérances dans les détails des calculs.
  - L'ordre des produits dans le calcul numérique est le même que celui dans la somme formelle.

2020-09-02 Nouvelle version mineure `0.9.0-beta`.

- **Calculs de l'espérance dans le cas fini :** `\calcexpval` rend facile la définition d'une variable aléatoire finie avec la possibilité de détailler le calcul de son espérance.
- **Arbre de probabilités :** pas mal de changements dans l'API et quelques nouveautés.
  - `\begin{probatree}<hideall> ... \end{probatree}` remplace l'usage de l'environnement `probatree*`.
  - `\begin{probatree}<showall> ... \end{probatree}` remplace l'usage de l'environnement `probatree**`.
  - De nouvelles macros permettent d'agir sur le texte des noeuds.
    1. `\ptreeTextOf` renvoie le texte d'un noeud.
    2. `\ptreeNodeColor` change les couleurs du texte et/ou du fond d'un noeud.

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

3. `\ptreeNodeNewText` change le texte en plus éventuellement les couleurs du texte et/ou du fond d'un noeud.

— `\ptreeFocus` a évolué.

1. `\ptreeFocus[frame = start]` remplace l'ancien `\ptreeFocus`.
2. `\ptreeFocus[frame = none]` remplace `\ptreeFocus**` qui n'existe plus.
3. `\ptreeFocus[frame = nostart]` est utilisé par défaut et remplace `\ptreeFocus*` qui n'existe plus. On obtient dans ce cas une mise en valeur encadrant tous les noeuds sauf le tout 1er.
4. Les clés optionnelles `lcol`, `tcol` et `bcol` permettent de choisir la couleur des arrêtes et des cadres, celle du texte et enfin celle du fond.

— `\aptreeFocus` a évolué presque comme `\ptreeFocus` car pour la couleur seule `lcol` est disponible.

— `tcol` remplace l'ancien `col` de `\ptreeComment` et `\aptreeComment`.

— `lcol` remplace l'ancien `col` de `\ptreeFrame` et `\aptreeFrame`.

2020-08-10 Nouvelle version mineure `0.8.0-beta`.

- **Arbre de probabilités :** le mode mathématique est activé par défaut pour les noms des noeuds et les poids *(plus besoin de taper plein de `$`)*.

2020-08-09 Nouvelle version mineure `0.7.0-beta`.

- **Arbre de probabilités.**
  — Utilisation obligatoire de `col=...` pour indiquer une couleur à toutes les macros de décoration.
  — Les macros `\ptreeComment`, `\ptreeComment*`, `\aptreeComment` et `\aptreeComment*` ont deux clés `dx` et `dy` pour indiquer un décalage relatif.
  — Ajout de l'environnement `probatree**` qui force l'affichage de tous les poids !

2020-08-08 Nouvelle version mineure `0.6.0-beta`.

- **Arbre de probabilités.**
  — `\aptreeFocus`, `\aptreeFocus*` et `\aptreeFocus**` permettent d'utiliser le système de nommage automatique des noeuds proposé par `forest`.
  — Il en va de même pour `\aptreeComment` et `\aptreeFrame`.

2020-08-05 Nouvelle version mineure `0.5.0-beta`.

- **Arbre de probabilités.**
  — `\ptreeFocus`, `\ptreeFocus*` et `\ptreeFocus**` fonctionnent avec un multi-argument pour pourvoir indiquer un chemin sur plusieurs noeuds.
  — Suppression de la clé `\pcomment`.

— Ajout des macros `\ptreeComment` et `\ptreeComment*` qui simplifient la saisie.

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`.

IX. Toutes les fiches techniques

1. Juste pour rédiger

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. valeur par défaut est `E`.

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

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

— `Option:` le nom de la fonction variance. valeur par défaut est `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 `\sigma`.

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

2. Calculer l'espérance – Cas fini

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

```
calc = calc-ulate
expval = exp-ected val-ue
```

— `Option:` un système de type `clé = valeur`.

1. `disp` indique ce qu'il faut afficher. La valeur par défaut est `table`. Voici toutes les valeurs possibles.
   (a) `table` : un tableau donnant la loi est affiché seul.
   (b) `all` : un tableau donnant la loi est affiché suivi du début du calcul de l'espérance.
   (c) `none` : rien n'est affiché.
   (d) `exp` : le début du calcul de l'espérance sans le tableau donnant la loi.
   (e) `formal` : la formule avec Σ .
   (f) `eval` : la somme des produits du type $p_k \cdot x_k$.
2. `nosigma` : utilisée seule cette option demande de ne pas afficher la formule avec Σ .
3. `E` : la macro utilisée pour écrire l'espérance. Par défaut `E = \expval` .
4. `X` : le nom de la variable aléatoire. Par défaut `X = X` .
5. `k` : l'indice. Par défaut `k = k` .
6. `xk` : la lettre à indicer pour les valeurs de la variable aléatoire. Par défaut `xk = x` .
7. `pk` : la lettre à indicer pour les valeurs des probabilités. Par défaut `pk = p` .
8. `com` : un texte court éventuel à ajouter entre le tableau donnant la loi et le début du calcul de l'espérance.
9. `ope` : le symbole de multiplication. Par défaut `ope = \cdot` .
10. `name` : un nom éventuel de la loi pour une utilisation ultérieure.
11. `reuse` : un nom indiquant ce qui doit être réutilisé.
12. `colors` : le cycle des couleurs au format TikZ séparées par des tirets.
    Par défaut `colors = red - blue - orange - green!70!black` .

— `Argument:` définition de la loi de la variable aléatoire finie avec une syntaxe semblable à celle d'un tableau.

3. Arbres pondérés

```
\begin{probatree}[#opt]
   ...
\end{probatree}
```

— `Option:` la valeur par défaut étant `asit`.

ATTENTION ! *L'option s'indique via `<...>` et non le traditionnel `[...]`.*

1. `asit` : pour les poids, on suit les indications données dans le code.
2. `hideall` : on cache tous les poids quelque soient les indications données dans le code.
3. `showall` : on montre tous les poids quelque soient les indications données dans le code.

— `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 "pweight*":` pour indiquer un poids sans l'imprimer. Avec l'option `showall` le poids sera affiché comme si on avait utilisé `pweight`.

— `Clé "apweight*":` pour indiquer un poids sans l'imprimer. Avec l'option `showall` le poids sera affiché comme si on avait utilisé `apweight`.

— Clé "bpweight*": pour indiquer un poids sans l'imprimer. Avec l'option showall le poids sera affiché comme si on avait utilisé bpweight.

— Clé "pframe": pour encadrer un sous-arbre depuis un noeud vers toutes les feuilles de celui-ci.

\ptreeComment [#opt]{#1..#2}
\ptreeComment* [#opt]{#1..#2}

— Option: un système de type clé = valeur.

1. tcol : la couleur au format TikZ du texte. La valeur par défaut est black.
2. dx : une distance horizontale relative de décalage. La valeur par défaut est 0cm.
3. dy : une distance verticale relative de décalage. La valeur par défaut est 0cm.

— Argument 1: le nom de la feuille.

— Argument 2: le texte du commentaire.

\aptreeComment [#opt]{#1..#2}
\aptreeComment* [#opt]{#1..#2}

Voir les indications précédentes excepté qu'ici on utilise le système de nommage automatisé dérivé de celui de forest.

\ptreeFocus [#opt]{#1}

— Option: un système de type clé = valeur.

1. lcol : la couleur au format TikZ des lignes des arrêtes et des cadres éventuels. La valeur par défaut est blue.
2. tcol : la couleur au format TikZ du texte. La valeur par défaut est black.
3. bcol : la couleur au format TikZ du fond. La valeur par défaut est white.
4. frame : le type d'encadrement souhaité. La valeur par défaut est nostart. Voici les valeurs disponibles.
   (a) nostart : le 1er noeud n'est pas encadré mais les autres le sont.
   (b) start : tous les noeuds sont encadrés.
   (c) none : aucun des noeuds n'est encadré.

— Argument: les noms des noeuds dans le bon ordre séparés par des barres verticales | .

\aptreeFocus [#opt]{#1}

— Option: un système de type clé = valeur plus réduit que celui de \ptreeFocus.

1. lcol : la couleur au format TikZ des lignes des arrêtes et des cadres éventuels. La valeur par défaut est blue.
2. frame : voir les indications précédentes.

— Argument: : voir les indications précédentes excepté qu'ici on utilise le système de nommage automatisé dérivé de celui de forest.

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

— Option: un système de type clé = valeur.

1. `lcol` : la couleur au format TikZ du cadre. La valeur par défaut est `blue`.

— `Arguments 1..3:` les noms de la sous-racine (à gauche), du noeud final en haut (à droite) et du noeud final en bas (à droite) tous indiqués via `name = ...` *(en fait l'ordre n'est pas important ici).*

`\aptreeFrame[#opt]{#1..#3}` `a = a-auto`

Voir les indications précédentes excepté qu'ici on utilise le système de nommage automatisé dérivé de celui de `forest`.

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

— `Option:` un système de type `clé = valeur`.

1. `tcol` : la couleur au format TikZ du texte. La valeur par défaut est `black`.
2. `dx` : une distance horizontale relative de décalage. La valeur par défaut est `0cm`.
3. `dy` : une distance verticale relative de décalage. La valeur par défaut est `0cm`.

— `Argument 1:` les noms des noeuds séparés par des barres verticales `|` .

— `Argument 2:` le texte commun décrivant chaque feuille.

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

Voir les indications précédentes excepté qu'ici on utilise le système de nommage automatisé dérivé de celui de `forest`.

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

— `Option:` un système de type `clé = valeur`.

1. `tcol` : la couleur au format TikZ du texte. La valeur par défaut est `black`.
2. `dx` : une distance horizontale relative de décalage. La valeur par défaut est `0cm`.
3. `dy` : une distance verticale relative de décalage. La valeur par défaut est `0cm`.

— `Argument 1:` le nom d'un noeud du niveau souhaité.

— `Argument 2:` le texte décrivant le niveau.

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

Voir les indications précédentes excepté qu'ici on utilise le système de nommage automatisé dérivé de celui de `forest`.

`\ptreeNodeColor{#1..#2}`

— `Argument 1:` le nom du noeud.

— `Argument 2:` chacun des paramètres présentés ci-dessous est optionnel et par défaut la macro utilise `tcol = black` et `bcol = white`.

1. `tcol = ...` sert à indiquer la couleur du texte.

2. `bcol = ...` pour à indiquer la couleur du fond *(`b = b-ackground`)*.

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

— `Option:` chacun des paramètres présentés ci-dessous est optionnel et par défaut la macro utilise `tcol = black` et `bcol = white`.

1. `tcol = ...` sert à indiquer la nouvelle couleur du texte.
2. `bcol = ...` pour à indiquer la nouvelle couleur du fond *(`b = b-ackground`)*.

— `Argument 1:` le nom du noeud.

— `Argument 2:` le nouveau texte.

`\ptreeTextOf{#1}`

— `Argument:` le nom du noeud.