Le package `tnsproba` : parler des probabilités facilement

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

Version `0.7.0-beta` développée et testée sur Mac OS X.

Christophe BAL

2020-08-09

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` réutilise les packages suivants sans aucun scrupule.

- `amsmath`
- `forest`
- `simplekv`
- `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. Généralités

1. Probabilité « simple »

Exemple 1

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

Exemple 2 – Choisir le nom de la probabilité

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

2. Probabilité conditionnelle

Exemple 1 – Les deux écritures classiques

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

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

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

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

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

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

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

$$E_1(X)$$

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

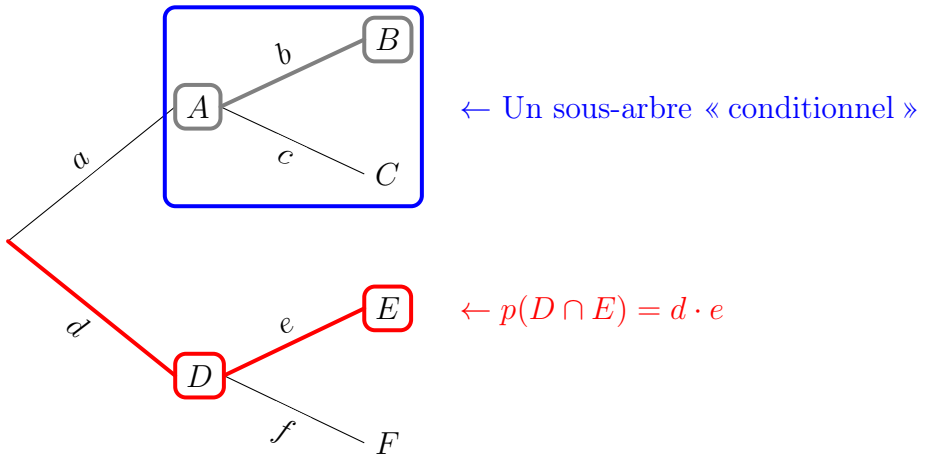
$$\sigma(X) \text{ ou } s(X)$$

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.

VI. 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
            [$B$, name     = nB,
                  apweight = $b$]
            [$C$, bpweight = $c$]   % Pas besion de nommer ce noeud.
        ]
        [$D$, bpweight = $d$,
              name     = nD
            [$E$, apweight = $e$,
                  name     = nE]
            [$F$, bpweight = $f$]   % Pas besion de nommer ce noeud.
        ]
    ]
    %
    \ptreeFocus[col=gray]{nA | nB}
    %
    \ptreeComment[col=blue]%
                 {nA}{$\leftarrow$ Un sous-arbre \og conditionnel \fg}
    %
    \ptreeFocus*[col=red]{nU | nD | nE}
    \ptreeComment[col=red]%
                 {nE}{$\leftarrow \proba{D \cap E} = d \cdot e$}
\end{probatree}
```

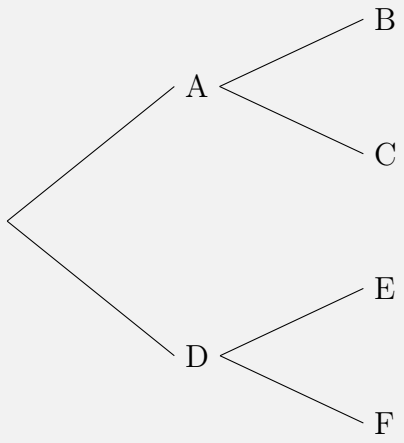
Remarque. Jusqu'à la section 6. page 14, nous nommerons à la main les noeuds des arbres via `name = ...` lorsque cela sera nécessaire. Dans la section indiquée nous verrons comment utiliser les noms automatiques donnés par le package `forest`.

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`.

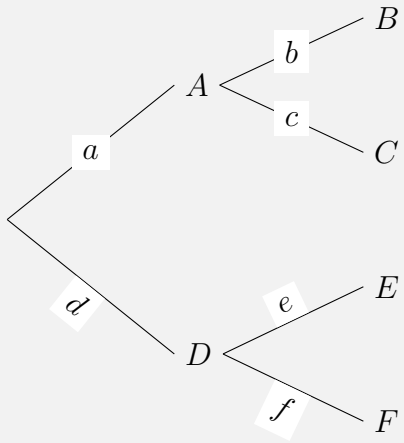
```
\begin{probatree}
    [             % Noeud racine sans
    texte
        [A        % Sous-noeud nommé
            [B]   % Sous-sous-noeud nommé
            [C]   % Sous-sous-noeud nommé
        ]
        [D        % Sous-noeud nommé
            [E]   % Sous-sous-noeud nommé
            [F]   % Sous-sous-noeud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



Exemple 2 – Ajouter des pondérations

Dans le code suivant, ce sont les clés[2] `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}
```

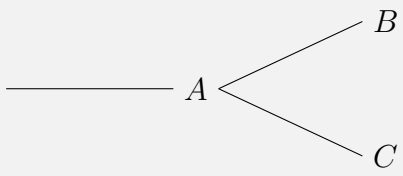


Exemple 3 – Des poids cachés partout

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)*.

2. En fait du point de vue de TikZ, ce sont des styles.

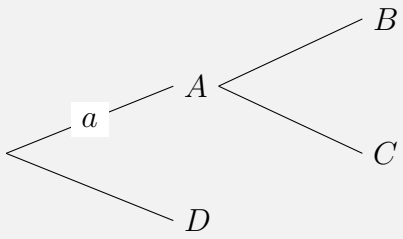
```
\begin{probatree*}
    [
        [$A$, pweight = $a$
            [$B$, apweight = $b$]
            [$C$, bpweight = $c$]
        ]
    ]
\end{probatree*}
```



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 *(ceci aussi peut servir à rédiger des exercices)*.

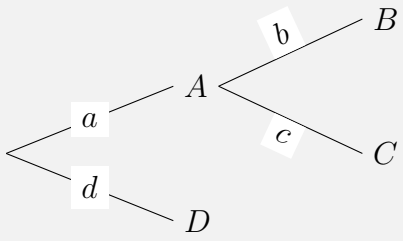
```
\begin{probatree}
    [
        [$A$, pweight = $a$
            [$B$, apweight* = $b$]
            [$C$, bpweight* = $c$]
        ]
        [$D$, pweight* = $d$]
    ]
\end{probatree}
```



Exemple 5 – Forcer l'affichage des poids cachés localement

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

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

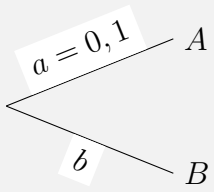


Exemple 6 – Un signe = et/ou une virgule dans les étiquettes

Vous ne pouvez pas utiliser directement un signe = ou une virgule dans les étiquettes des branches [3]. Pour contourner cette limitation, il suffit de mettre le contenu de l'étiquette entre des accolades.

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

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

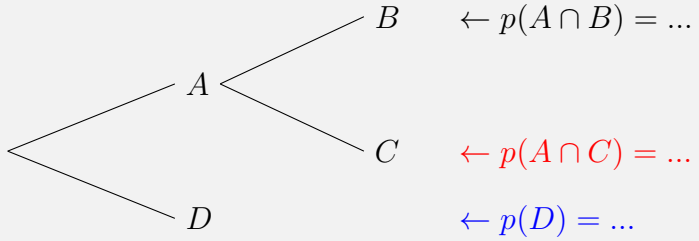


3. Commenter les racines

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[4]. Noter l'utilisation possible de la clé `col` pour `col`-or afin d'indiquer une couleur au format TikZ. La couleur oar défaut est le noir.

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



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

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

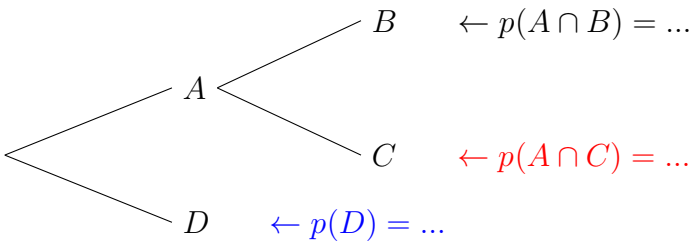
$\leftarrow p(D) = ...$

Remarque. Commenter un noeud interne ne provoquera pas d'erreur même si `\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

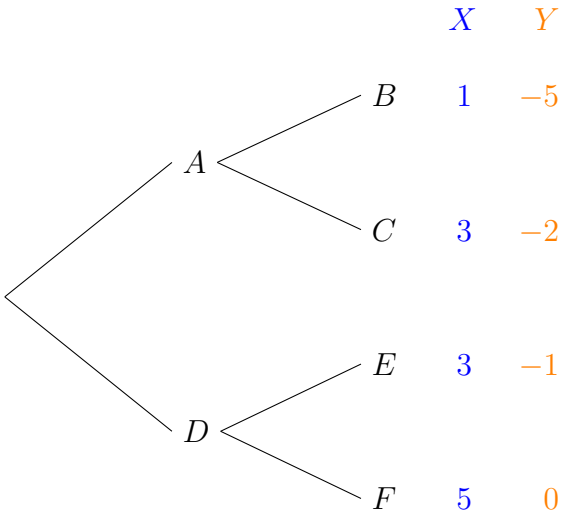
En utilisant `\ptreeComment*` au lieu de `\ptreeComment`, les commentaires seront proches des noeuds et donc non alignés verticalement. Avec l'exemple précédent on obtient la mise en forme qui suit.

4. 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[col=blue, dx=-.25em, dy=1cm]%
                 {nB}{$X$}
    \ptreeComment[col=blue]{nB}{$1$}
    \ptreeComment[col=blue]{nC}{$3$}
    \ptreeComment[col=blue]{nE}{$3$}
    \ptreeComment[col=blue]{nF}{$5$}
    %
    % Valeurs de Y
    \ptreeComment[col=orange, dx=2em+.5em, dy=1cm]%
                 {nB}{$Y$}
    \ptreeComment[col=orange, dx=2em]{nB}{$-5$}
    \ptreeComment[col=orange, dx=2em]{nC}{$-2$}
    \ptreeComment[col=orange, dx=2em]{nE}{$-1$}
    \ptreeComment[col=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[col=blue, dx=-.25em, dy=1cm]%
                 {nB}{$X$}
    \foreach \name/\val in {
        nB/$1$,
        nC/$3$,
        nE/$3$,
        nF/$5$
    }{
        \ptreeComment[col=blue]{\name}{\val}
    }
    % Valeurs de Y
    \ptreeComment[col=orange, dx=2em+.5em, dy=1cm]%
                 {nB}{$Y$}
    \foreach \name/\val in {
        nB/$-5$,
        nC/$-2$,
        nE/$-1$,
        nF/$\phantom{-}0$
    }{
        \ptreeComment[col=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. Les noms `\color`, `\col`, `\dx` et `\dy` sont utilisés en coulisse donc vous ne pourrez pas les utiliser dans votre boucle.

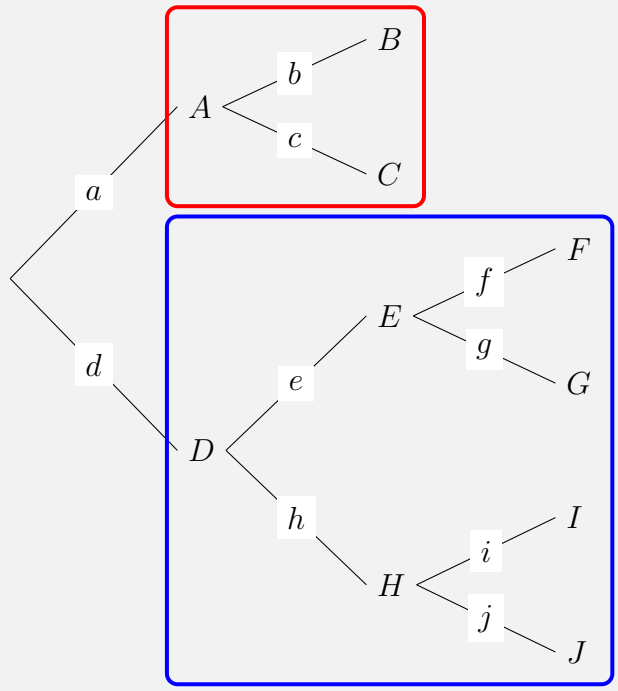
4. Avec des cadres

Exemple 1 – Des cadres finaux

Via la clé `pframe` il est très aisé d'encadrer un sous-arbre final [5] comme le montre l'exemple suivant [6]. Dans l'exemple ci-après nous utilisons la bidouille `{},s sep = 1.3cm` qui évite que les cadres se superposent.

5. Un sous-arbre sera dit final si toutes ses feuilles correspondent à des feuilles de l'arbre initial.
6. 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}
```

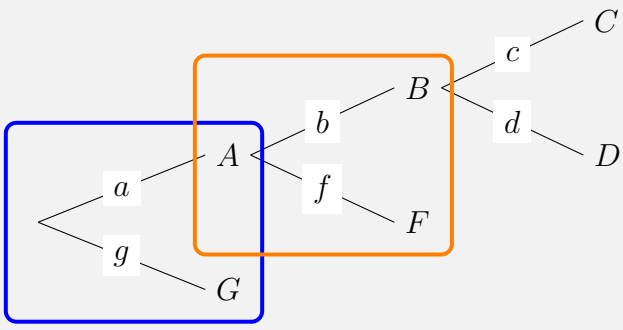


Remarque. La clé `pframe` est un cas particulier de décoration car les autres décorations se font en dehors de la définition de l'arbre

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[col=orange]{nA}{nB}{nF}
\end{probatree}
```

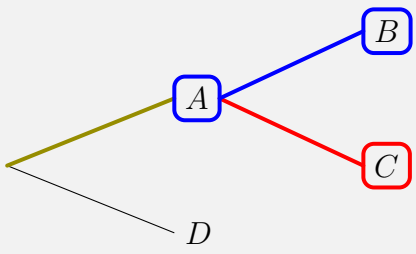


5. Mettre en valeur des chemins

Exemple 1 – Juste avec deux noeuds

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**`. 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]
        ]
        [$D$]
    ]
    %
    \ptreeFocus* [col=red]  {nA | nC}
    \ptreeFocus**[col=olive]{nU | nA}
    \ptreeFocus             {nA | nB}
\end{probatree}
```

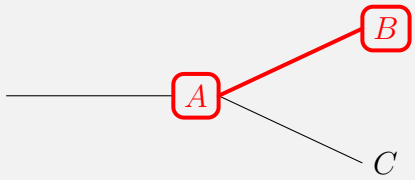


Voici ce qu'il faut retenir.

1. `\ptreeFocus` encadre tous les noeuds.
2. `\ptreeFocus*` n'encadre pas le tout premier noeud *(typiquement cela est utile pour un chemin partant de la racine de l'arbre si celle-ci n'est pas nommée comme on le fait très souvent)*.
3. `\ptreeFocus**` n'encadre aucun des noeuds.
4. La couleur peut être changée via l'argument optionnel en utilisant les couleurs de type TikZ. Par défaut le bleu est utilisé.

Remarque. Le fonctionnement interne de `forest` empêche une coloration automatique des noeuds. Si vous souhaitez obtenir cet effet il faudra ajouter les couleurs à la main pour chaque noeud comme dans l'exemple qui suit.

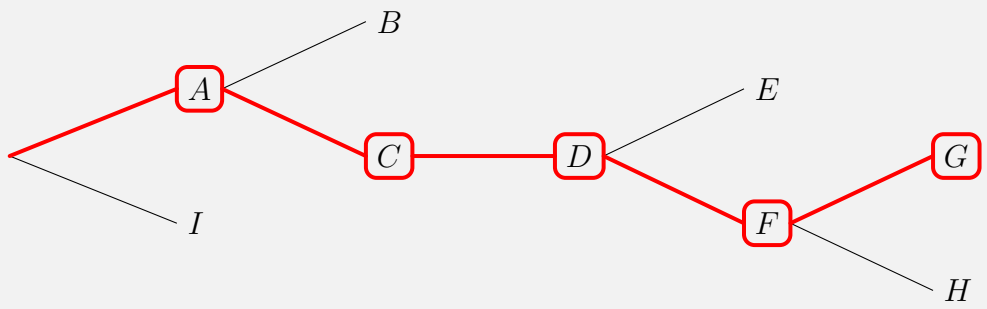
```
\begin{probatree}
    [
        [$A$, name = nA, red
            [$B$, name = nB, red]
            [$C$]
        ]
    ]
    %
    \ptreeFocus[col=red]{nA | nB}
\end{probatree}
```



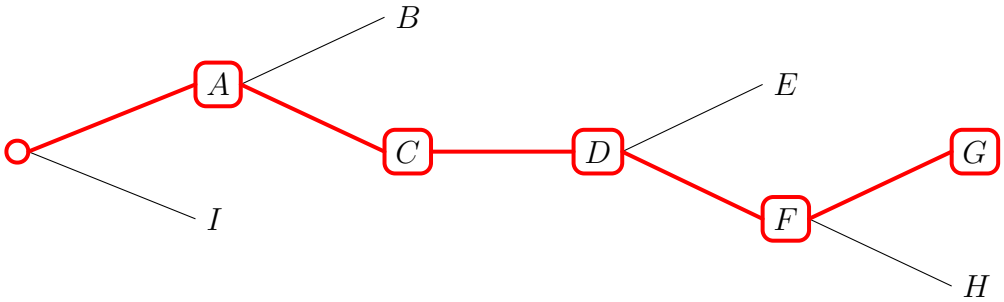
Exemple 2 – Plusieurs noeuds d'un coup

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

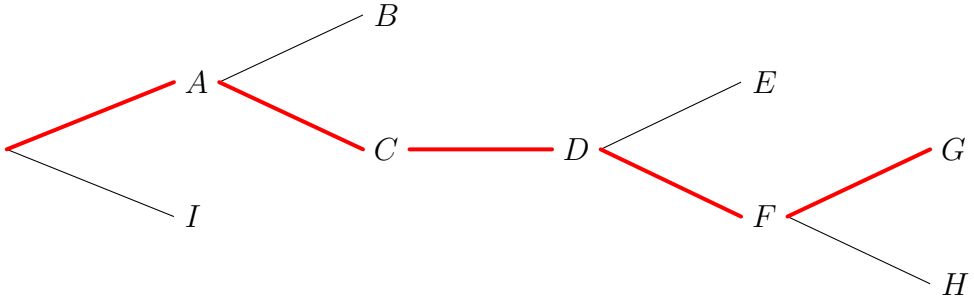
```
\begin{probatree}
    [{}, name = nU
        [$A$, name = nA
            [$B$]
            [$C$, name = nC
                [$D$, name = nD
                    [$E$]
                    [$F$, name = nF
                        [$G$, name = nG]
                        [$H$]
                    ]
  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



Avec `\ptreeFocus` on obtient l'arbre suivant où le mini disque initial [7] n'est pas forcément souhaité.



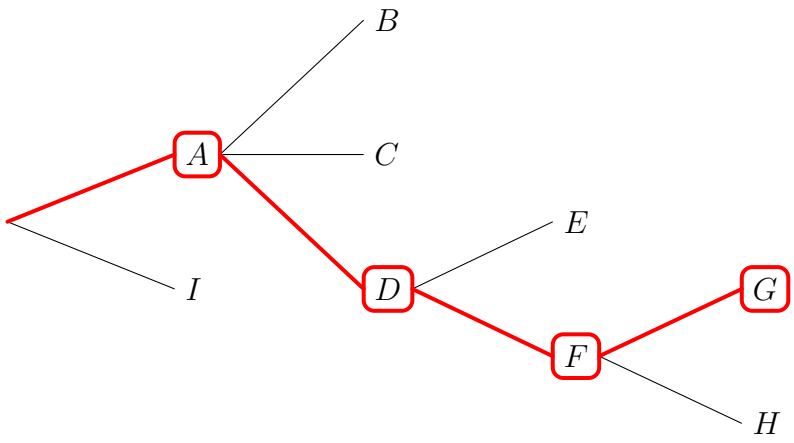
Avec `\ptreeFocus**` on obtient l'arbre ci-dessous.



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

6. 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.

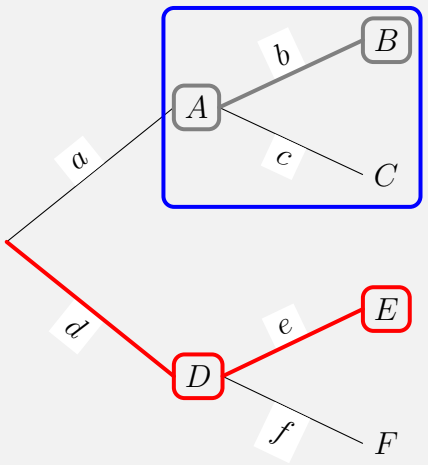
```
\begin{probatree}
    [
        [$A$
            [$B$]
            [$C$]
            [$D$
                [$E$]
                [$F$
                    [$G$]
                    [$H$]
                ]
            ]
        ]
        [$I$]
    ]
    %
    \aptreeFocus*[col=red]{ | 1 | 13 | 132 |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

Voici comment s'y prendre.

1. On utilise `\aptreeFocus` au lieu de `\ptreeFocus` où le préfixe `a` est pour `a`-uto. De façon similaire il existe les macros `\aptreeComment` et `\aptreeFrame`.
2. Chaque nom automatique 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 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. C'est notre noeud nommé $\boxed{1321}$.

Remarque. Utilisez de préférence les noms automatiques car cela facilitera la maintenance de vos 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. A titre illustratif on a utilisé `\ptreeFrame` au lieu de la clé `pframe` directement dans l'arbre.

```
\begin{probatree}
    [
        [$A$, apweight = $a$
            [$B$, apweight = $b$]
            [$C$, bpweight = $c$]
        ]
        [$D$, bpweight = $d$
            [$E$, apweight = $e$]
            [$F$, bpweight = $f$]
        ]
    ]
    %
    \aptreeFocus[col=gray]{1 | 11}
    %
    \aptreeFrame{1}{11}{12}
    \aptreeComment[col=blue]%
                  {1}{$\leftarrow$ Un sous-arbre \og conditionnel \fg}
    %
    \aptreeFocus*[col=red]{ | 2 | 21}
    \aptreeComment[col=red]%
                  {21}{$\leftarrow \proba{D \cap E} = d \cdot e$}
\end{probatree}
```



← Un sous-arbre « conditionnel »

← $p(D \cap E) = d \cdot e$

VII. Historique

Nous ne donnons ici qu'un très bref historique récent [8] 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`.

2020-08-09 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`.

---

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

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`.

VIII. Toutes les fiches techniques

1. 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.

2. Arbres pondérés

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

`\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 "pweight*":` pour indiquer un poids sans l'imprimer. Dans l'environnement `probatree**`, le poids sera affiché comme si on avait utilisé `pweight`.

— `Clé "apweight*":` pour indiquer un poids sans l'imprimer. Dans l'environnement `probatree**`, le poids sera affiché comme si on avait utilisé `apweight`.

— `Clé "bpweight*":` pour indiquer un poids sans l'imprimer. Dans l'environnement `probatree**`, 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.

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

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

1. `col` : une couleur au format TikZ. 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`.

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

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

1. `col` : une couleur au format TikZ. 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}`

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

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

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

1. `col` : une couleur au format TikZ. La valeur par défaut est `blue`.

— `Argument:` les noms des noeuds indiqués via `name = ...` à fournir dans le bon ordre et à séparer par des barres verticales `|`.

```
\aptreeFocus  [#opt]{#1}
\aptreeFocus* [#opt]{#1}
\aptreeFocus**[#opt]{#1}
```

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