Package patchwork et fonction ggMarginal du package ggExtra

Mélissa Rochette et Anne-Marie Sauvageau

2020-03-22

Table des matières

Introduction

`ggplot2` est un package très puissant en R devenu populaire au courant des dernières années grâce à sa capacité de créer des graphiques en utilisant sa propre grammaire graphique. C'est cette approche qui lui permet de créer des graphiques complexes plus simplement. Le package est aussi reconnu pour son automatisation de certaines configurations graphiques comme les légendes ainsi que pour ses sorties plus esthétiques. Cette fiche présente donc deux extensions de `ggplot2` qui visent l'amélioration de l'expérience de l'utilisateur du package en rendant possibles certaines manipulations de graphiques autrement impossibles. Ces deux extensions sont `patchwork`, un package permettant la combinaison de « ggplots » (graphiques créés avec le package `ggplot2`) au sein d'un même graphique, et la fonction `ggMarginal` de `ggExtra`, une fonction pour ajouter des graphiques marginaux à un diagramme de dispersion. Nous documentons dans un premier temps `patchwork` et dans un deuxième, la fonction `ggMarginal` de `ggExtra`.

Description des données

Les exemples présents dans cette fiche seront réalisés à partir du data frame `quakes` du package `datasets` qui est chargé par défaut à l'ouverture d'une session R. Nous retrouvons au sein de ce data frame des

données portant sur des évènements sismiques de magnitude supérieure à 4 sur l'échelle de Richter ayant été enregistrés aux îles Fidji depuis 1964. Ces 1000 observations de séismes sont décrites par cinq variables : latitude, longitude, profondeur, magnitude et le nombre de stations sismiques ayant rapporté l'événement. Nous ajoutons à ce data frame une nouvelle variable catégorique, la région, qui peut prendre deux valeurs : Est ou Ouest.

```
# Ajout de la variable region au jeu de données
quakes$region <- factor(quakes$long >= 175, labels = c("Ouest", "Est"))
str(quakes)
```

```
## 'data.frame':    1000 obs. of  6 variables:
##  $ lat     : num  -20.4 -20.6 -26 -18 -20.4 ...
##  $ long    : num  182 181 184 182 182 ...
##  $ depth   : int  562 650 42 626 649 195 82 194 211 622 ...
##  $ mag     : num  4.8 4.2 5.4 4.1 4 4 4.8 4.4 4.7 4.3 ...
##  $ stations: int  41 15 43 19 11 12 43 15 35 19 ...
##  $ region  : Factor w/ 2 levels "Ouest","Est": 2 2 2 2 2 2 1 2 2 2 ...
```

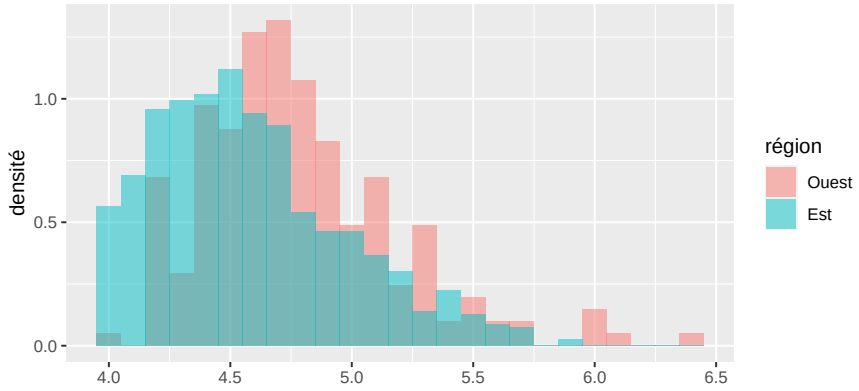
À partir de ces données, nous produisons avec `ggplot2` quatre graphiques qui seront utilisés pour illustrer les différentes fonctionnalités du package `patchwork` ainsi que de la fonction `ggMarginal` du package `ggExtra`.

Nous devons donc commencer par installer et charger le package `ggplot2` disponible à partir de CRAN.

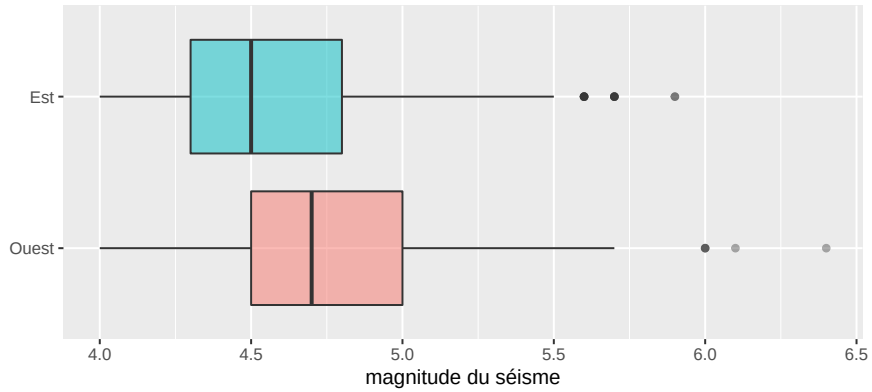
```
# install.packages("ggplot2")
library(ggplot2)
```

Nous créons ensuite les quatre graphiques : un histogramme, un diagramme en boîte, un diagramme quantile-quantile et un diagramme de dispersion.

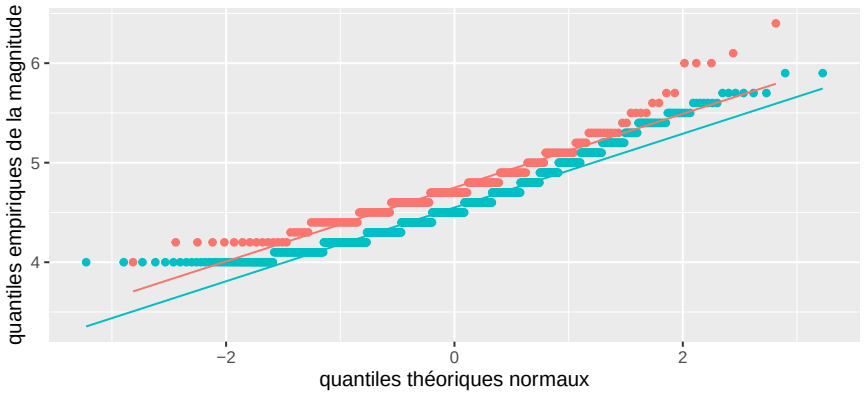
```
 p1 <- ggplot(
   data = quakes,
   mapping = aes(x = mag, fill = region, alpha = 0.5)
) +
  geom_histogram(
    mapping = aes(y=stat(density)),
    position ="identity",
    alpha=.5,
    center = 0,
    binwidth = 0.1
  ) +
  labs(y = "densité") +
  labs(x = NULL ) +
  labs(fill = "région")
p1
```



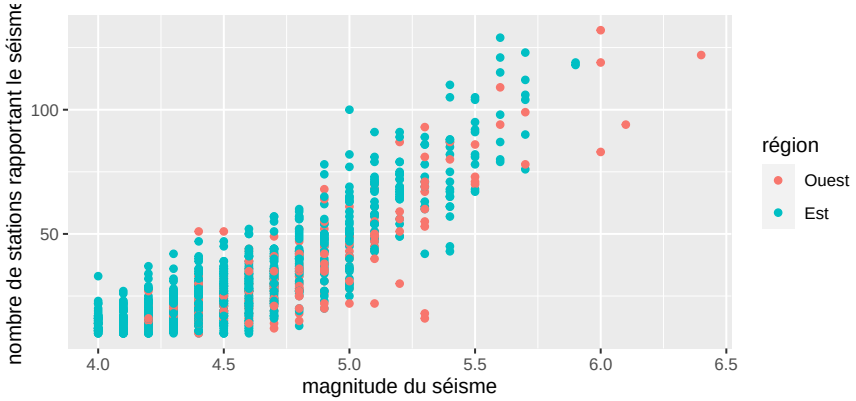

```
p2 <- ggplot(
  data = quakes,
  mapping = aes(x = region, y = mag,fill=region)
) +
  geom_boxplot(
    aes(group = cut_width(region, 0.50)),
    outlier.alpha = 0.4,
    alpha=0.5,
    show.legend = FALSE
  ) +
  coord_flip() +
  labs(x=NULL)+
  labs(y="magnitude du séisme")
p2
```




```
p3 <- ggplot(
  data = quakes,
  mapping = aes(sample = mag, colour = region)
) +
  stat_qq(show.legend = FALSE)+
  stat_qq_line(show.legend = FALSE) +
  labs(y = "quantiles empiriques de la magnitude") +
  labs(x = "quantiles théoriques normaux")
p3
```




```
p4 <-ggplot(
  data = quakes,
  mapping = aes(x = mag, y = stations)
) +
  geom_point(
    mapping = aes(colour = region)
  ) +
  labs(x = "magnitude du séisme") +
  labs(y = "nombre de stations rapportant le séisme") +
  labs(colour = "région")
p4
```



Le package `patchwork`

La production de graphiques en R est très utile lorsque vient le temps de communiquer nos résultats. Le package `ggplot2` a été conçu pour rendre cette opération plus simple et obtenir des graphiques transmettant leurs messages de façon plus efficace. Cependant, une des lacunes de `ggplot2` est qu'il ne prévoit pas une façon simple pour combiner plusieurs graphiques dans une même fenêtre. Le package `patchwork` offre une solution à cette problématique. Il propose une approche intuitive pour facilement combiner des graphiques, et ce même dans une composition complexe. Il existe d'autres packages qui tentent de combler ce besoin en passant par d'autres chemins comme `gridExtra` ou encore `cowplot`.

Démarrage

Le package `patchwork` doit être installé, puis chargé dans la session R.

```
# install.packages("patchwork")
library(patchwork)
```

Éléments de base

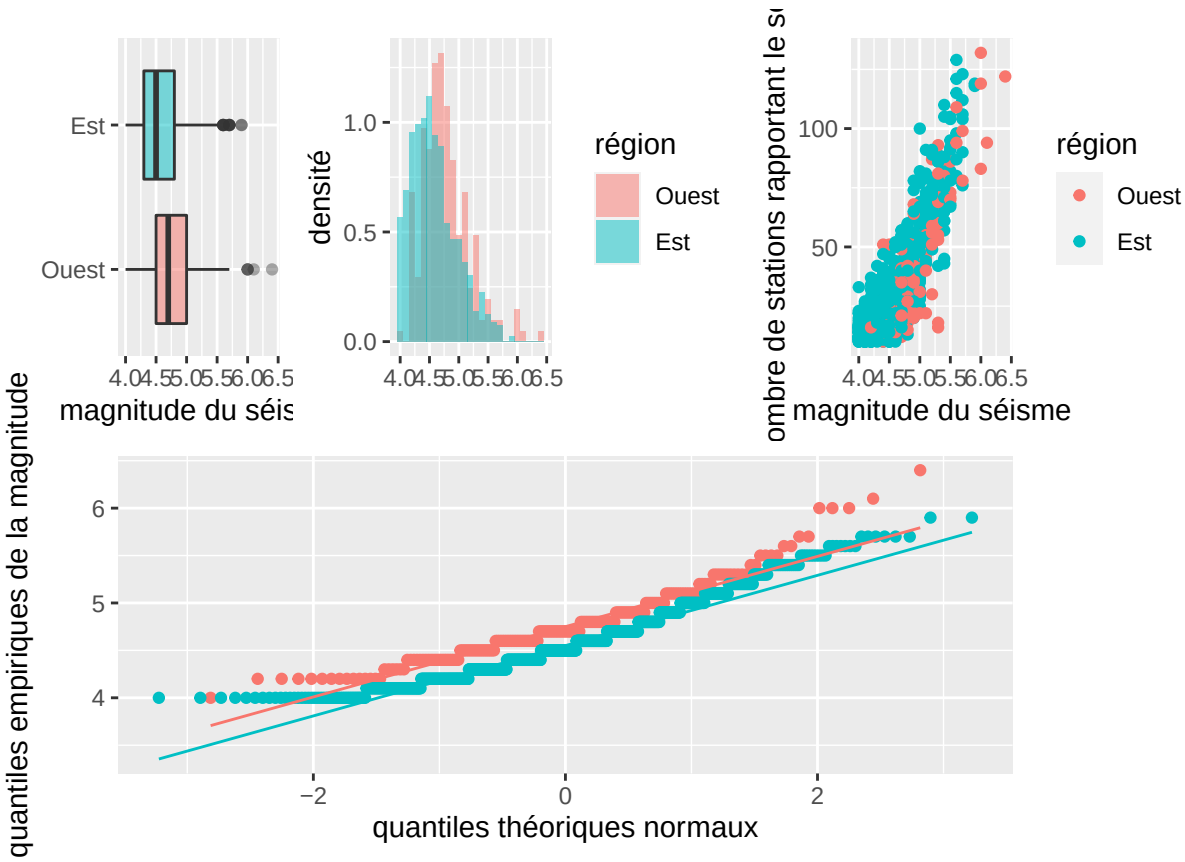
Une fois le package `patchwork` chargé, l'utilisation de l'opérateur `+` permet de placer deux graphes ou plus dans la même fenêtre graphique.

```
p1 + p2
```

Si on ajoute plus de graphiques avec l'opérateur `+`, ils seront ajoutés par défaut à la même ligne puis à la ligne inférieure. Le package `patchwork` essaie de conserver une mise en forme carrée. Il est aussi possible d'ajouter les graphiques colonne par colonne et de modifier la mise en forme de base à l'aide de `plot_layout` dont nous parlerons plus tard dans le document.

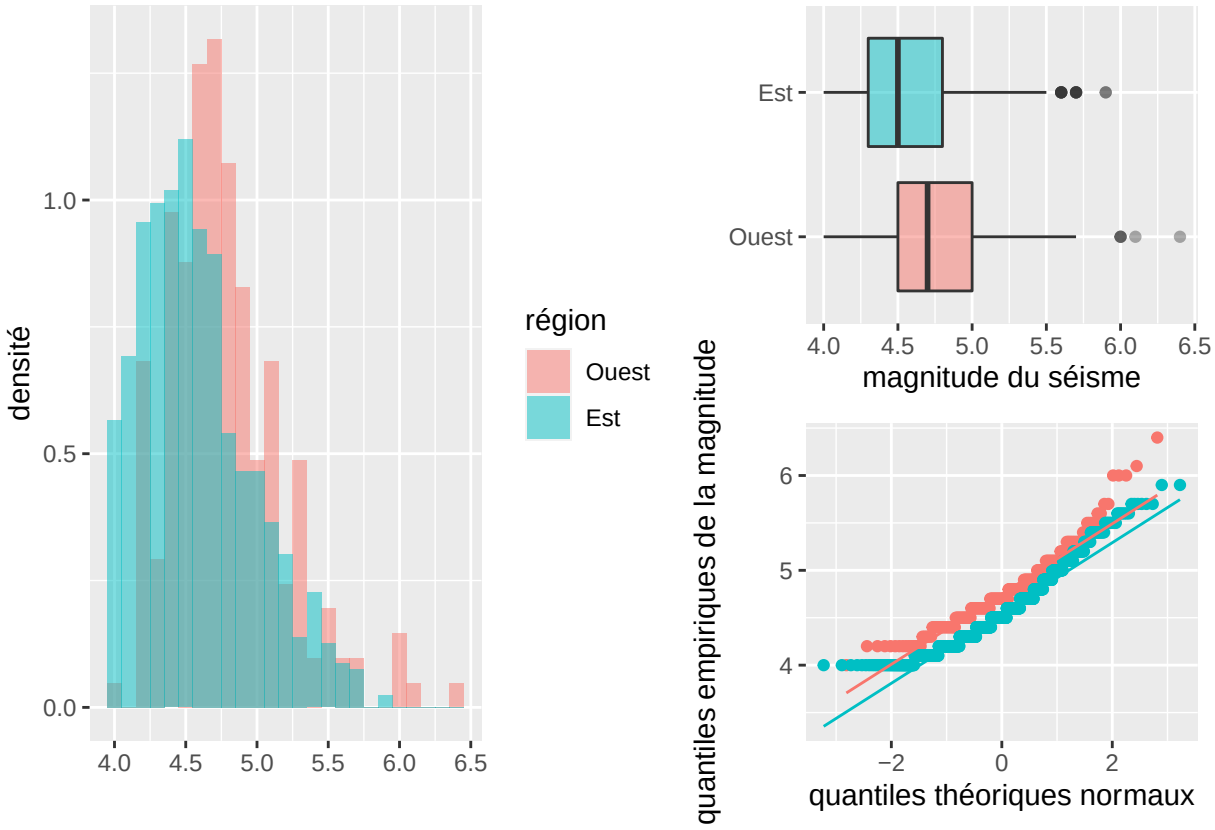
L'utilisation des opérateurs `/` et `|` permettent aussi à l'utilisateur d'avoir plus de contrôle, et ce de façon très intuitive.

```
(p2 | p1 | p4 ) /
      p3
```



L'opérateur `|` ajoute le graphique ou le groupe de graphiques le suivant en ligne, alors que l'opérateur `/` ajoute le graphique ou le groupe de graphiques le suivant en colonne. En effet, dans l'exemple précédent, le graphique `p1` suivait un opérateur `|`, donc il s'ajoute sur la même ligne que le précédent (`p2`). Il en est de même pour le graphique `p4` qui suit un opérateur `|` et donc, il est sur la même ligne que les graphiques `p1` et `p2`. Les parenthèses servent à faire un groupe de graphique. Ce groupe sera ensuite traité comme un seul «graphique». Puis, l'opérateur `/` précède le graphique `p3`, donc `p3` est mis en colonne en dessous du groupe de graphiques comportant `p1`, `p2` et `p4`.

```
p1 | (p2 / p3)
```



Dans cet exemple, nous devons en premier lieu porter notre attention sur les parenthèses tel qu'il est fait normalement en terme de priorité d'opérations. Dans ces parenthèses, nous avons `p3` précédé d'un opérateur `/`, donc `p3` est ajouté en colonne en dessous de `p2`. Ensuite, `p2` et `p3` forment un groupe et est traité comme un seul «graphique». Ce groupe est précédé de l'opérateur `|`, donc le groupe est mis en ligne à droite de `p1`.

Ajouts à un `patchwork` existant

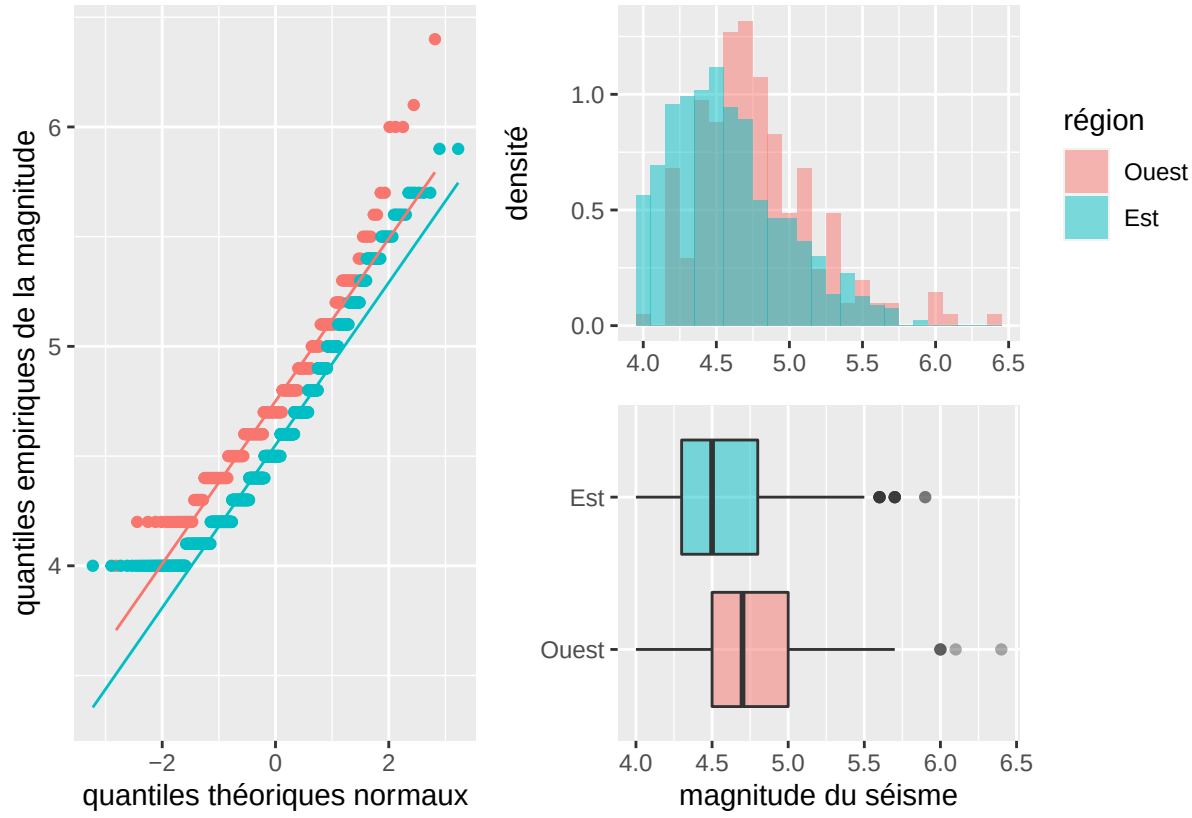
Le mot «patchwork» réfère à une technique de couture où l'on assemble plusieurs morceaux de tissus hétérogènes afin de réaliser différents types d'ouvrages comme la courtepointe par exemple. Les graphiques étant ici nos morceaux de tissus, il est possible de continuer le travail en ajoutant un simple morceau de tissu à un assemblage existant.

Ajout d'un graphique `ggplot2`

Si le morceau de tissu en question est un autre graphique, il est facile de le lier à un «patchwork» existant. L'ajout d'un nouveau graphique se fait toujours du côté gauche avec l'opérateur `+`.

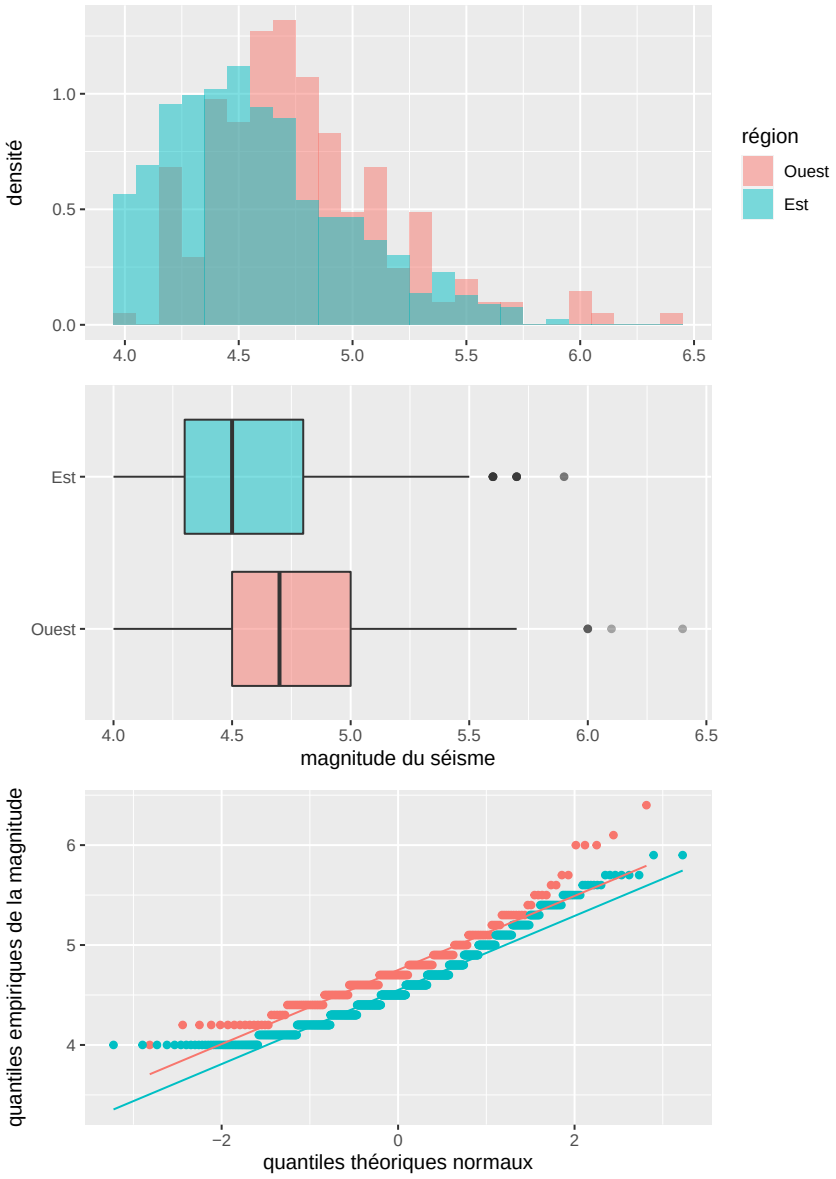
```
courtepointe <- p1 / p2

p3 + courtepointe
```



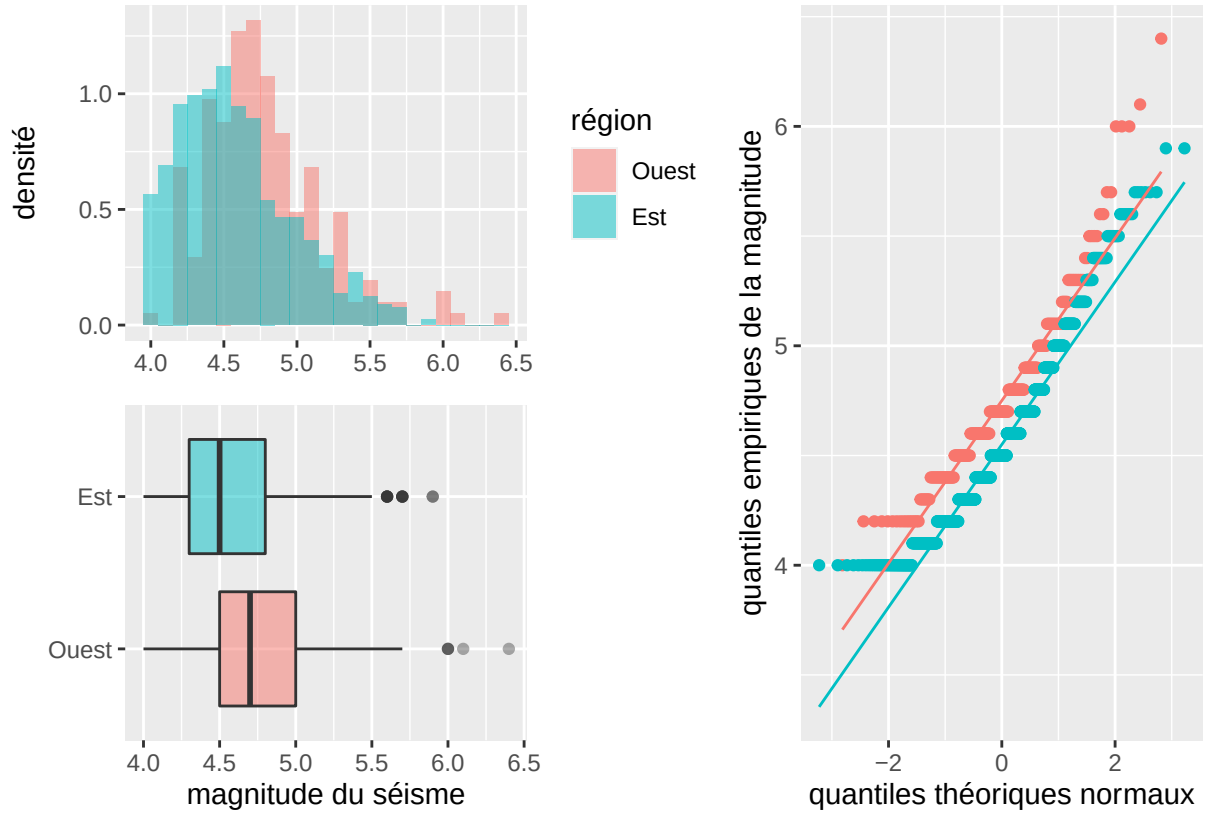
Si l'on souhaite ajouter notre nouveau graphique du côté droit, c'est un peu plus compliqué.

```
courtepointe + p3
```



On remarque que le nouveau graphique a été ajouté en dessous plutôt que du côté droit et il est de la même taille que chacun des graphiques de notre assemblage existant. Nous voulions plutôt avoir ce nouveau graphique `p3` du côté droit de `courtepointe` et que l'espace de `p3` soit de la même taille que l'espace de `courtepointe`. Pour ce faire, il faut utiliser l'opérateur `-` plutôt que `+`. L'opérateur `-` jouera ici le rôle d'un trait d'union.

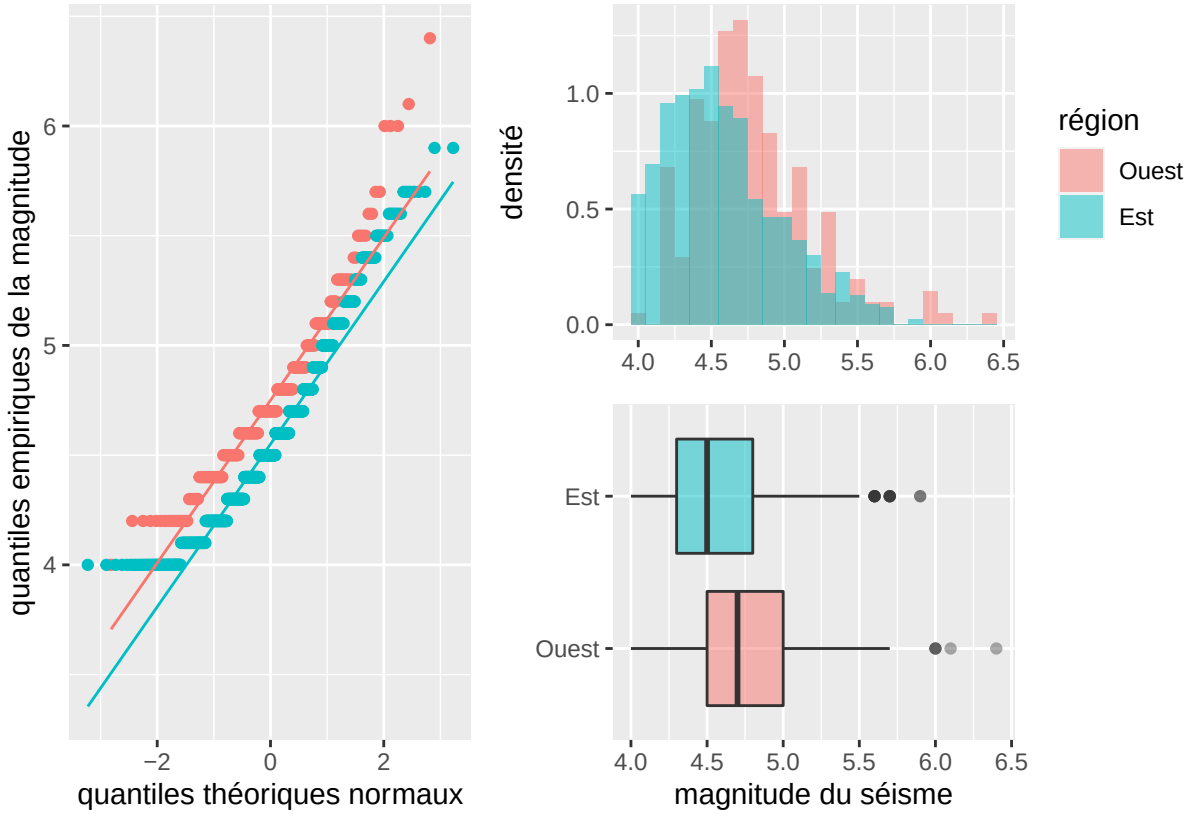
```
courtepointe - p3
```



Il est aussi possible d'utiliser la fonction `wrap_plots` pour combiner plusieurs graphiques. Cette fonction va conserver les tailles (proportions) des graphiques mis en entrée, et ce peu importe leur ordre contrairement à l'opérateur `+`.

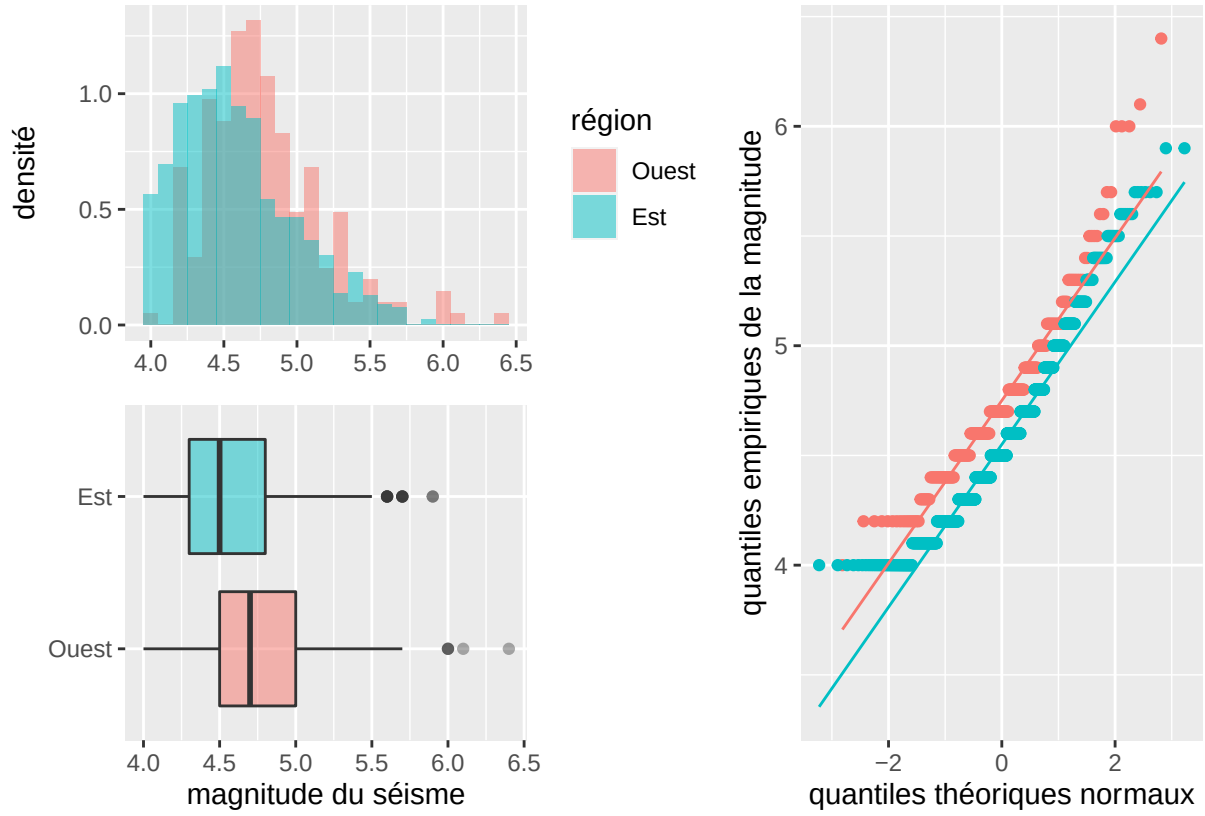
Du côté gauche :

```
# Équivalent à la commande : p3 + courtpointe
wrap_plots(p3, courtepointe)
```



Du côté droit :

```
# Équivalent à la commande : courtpointe - p3
courtepointe2 <- wrap_plots(courtepointe, p3)
courtepointe2
```



Ajout de contenus autres que des graphiques `ggplot2`

Il est également possible de composer notre «patchwork» d'autres éléments que des graphiques `ggplot2`. Nous pouvons ajouter :

- des graphiques produits avec le système graphique R de base en utilisant la fonction `wrap_elements` (à la condition que le package `gridGraphics` soit installé) ;
- des zones de textes avec la fonction `textGrob` du package `grid` ;
- des tableaux avec le fonction `tableGrob` du package `gridExra`.

Des exemples de tels ajouts sont présentés ici : https://patchwork.data-imaginist.com/articles/guides/assembly.html

Modifications d'un `patchwork` existant

Le package `patchwork` permet de modifier des graphiques `ggplot2` parmi un «patchwork» existant. En d'autres mots, nous pouvons modifier des morceaux de tissus de notre courtepointe.

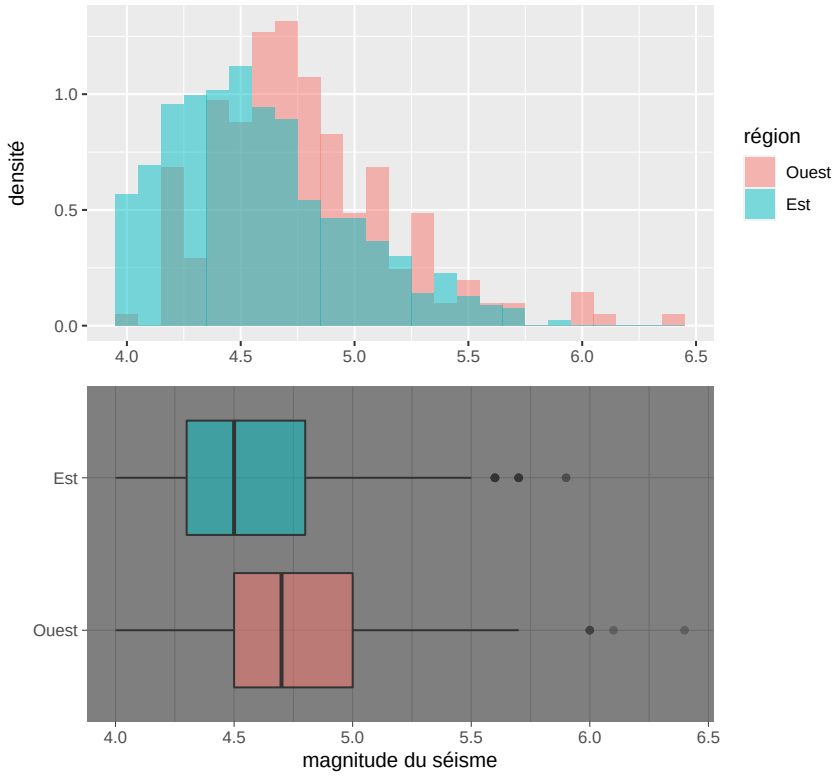
Modifications d'un graphique

Avec l'opérateur `+`, il est possible de modifier un graphique d'un «patchwork». Si nous ajoutons une commande pour modifier le thème d'un graphique à un «patchwork», cette commande sera appliquée sur le dernier

graphique. Dans notre première courtepointe produite, le dernier graphique joint est le diagramme en boîte.

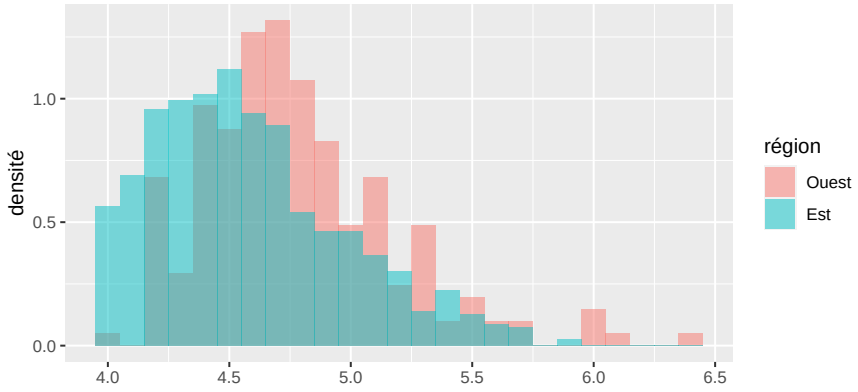
Si nous désirons modifier le dernier graphique inclus dans un «patchwork», il est possible d'utiliser l'opérateur `+` comme suit :

```
courtepointe + theme_dark()
```



Il est également possible de modifier les autres graphiques dans notre courtepointe en utilisant l'opérateur doubles crochets `[[`. En effet, il est possible d'extraire un graphique d'un «patchwork» avec cet opérateur comme il est possible pour extraire un élément d'un objet.

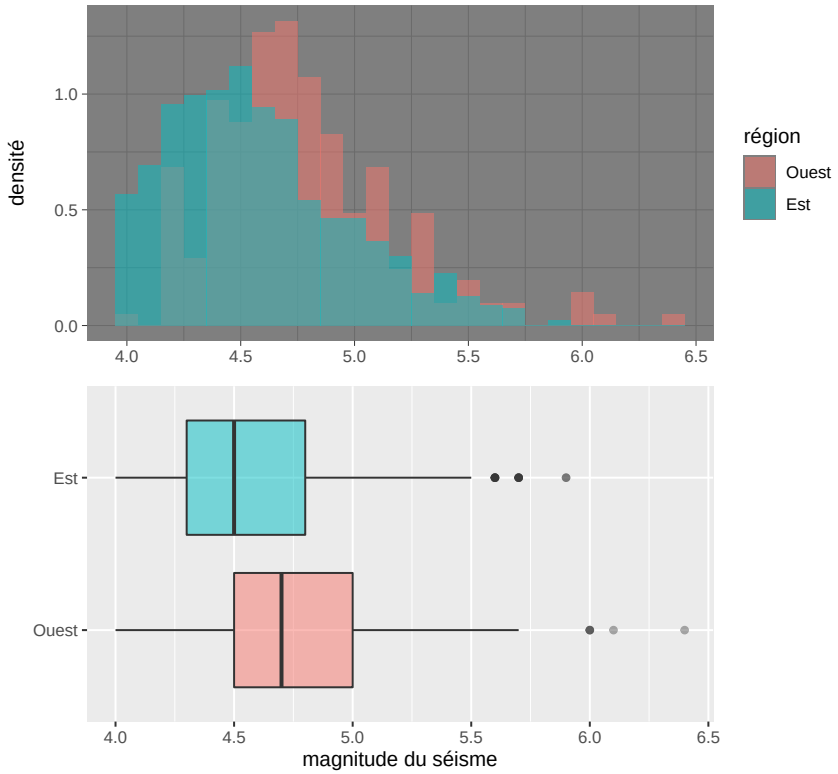
```
courtepointe[[1]]
```



Puis, il est possible de faire des modifications à un graphique et de le remplacer dans le `patchwork`.

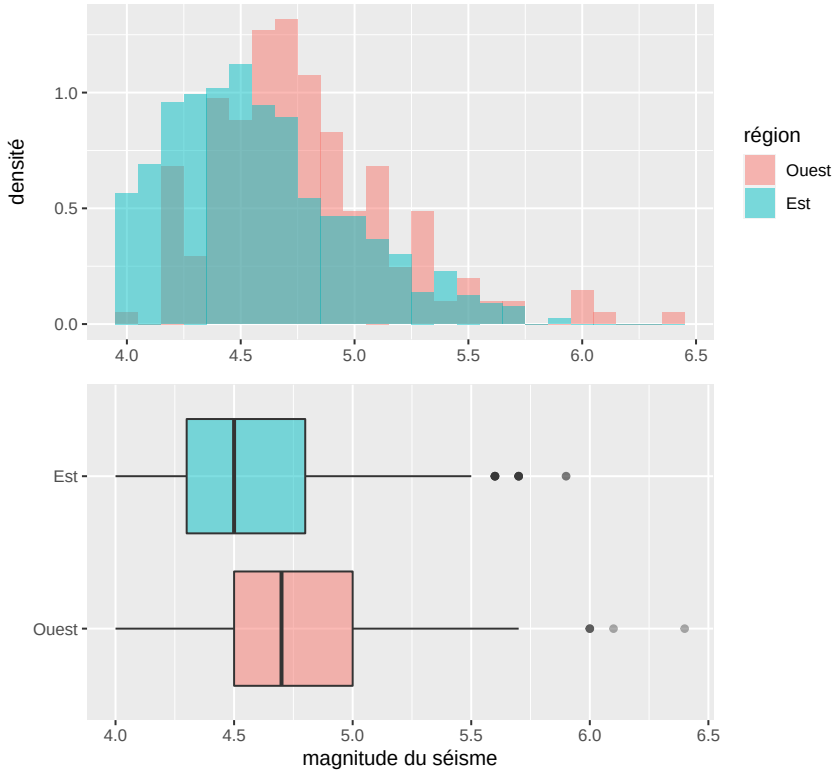
```
courtepointeCopie <- courtepointe
courtepointeCopie[[1]] <- courtepointeCopie[[1]] + theme_dark()
```

```
courtepointeCopie
```



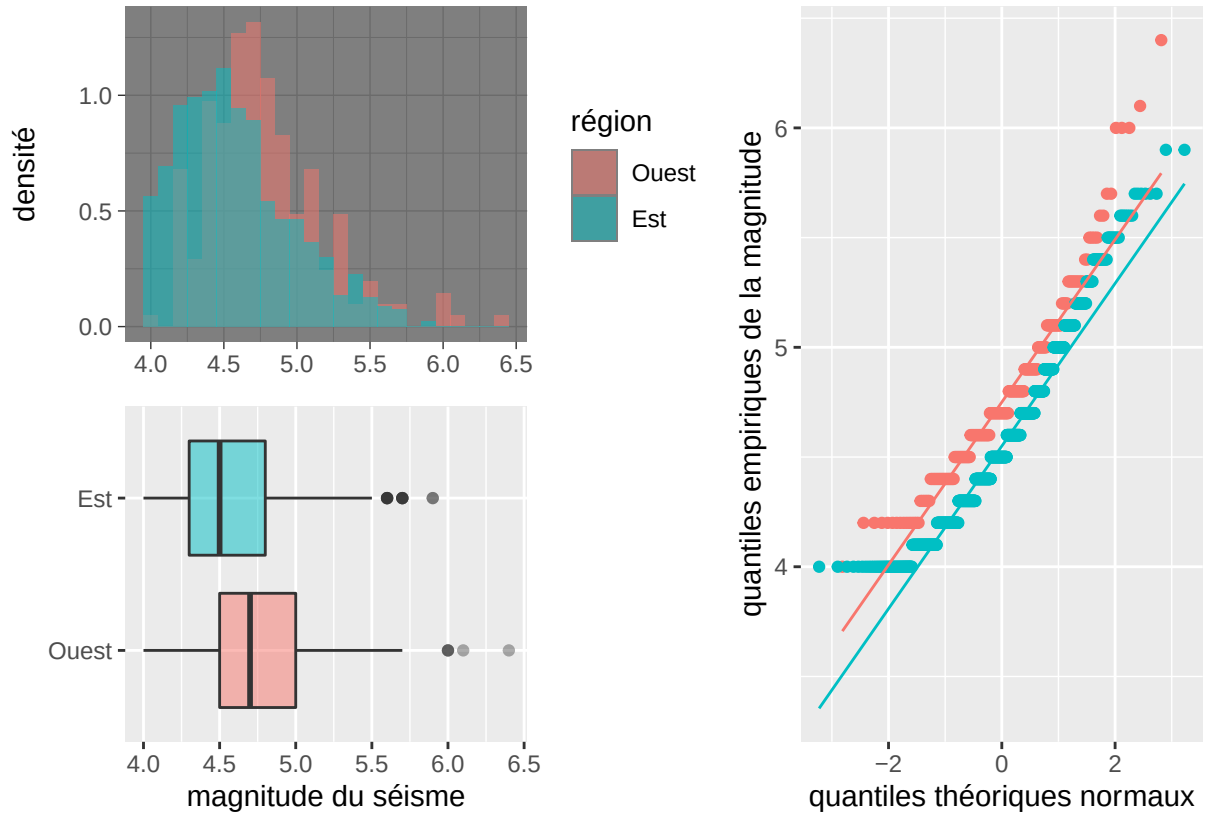
Dans le cas où le «patchwork» est plus complexe, cela peut nécessiter plusieurs extractions en séquence. En effet, supposons que nous souhaitons modifier le thème du graphique en haut à gauche de `courtepointe2`, l'extraction du premier élément extrait les deux graphiques de gauche.

```
#Extraction du premier élément
courtepointe2[[1]]
```



Ainsi, pour extraire le graphique en haut à gauche de `courtepointe2`, il faudra extraire à nouveau le premier élément du dernier «patchwork».

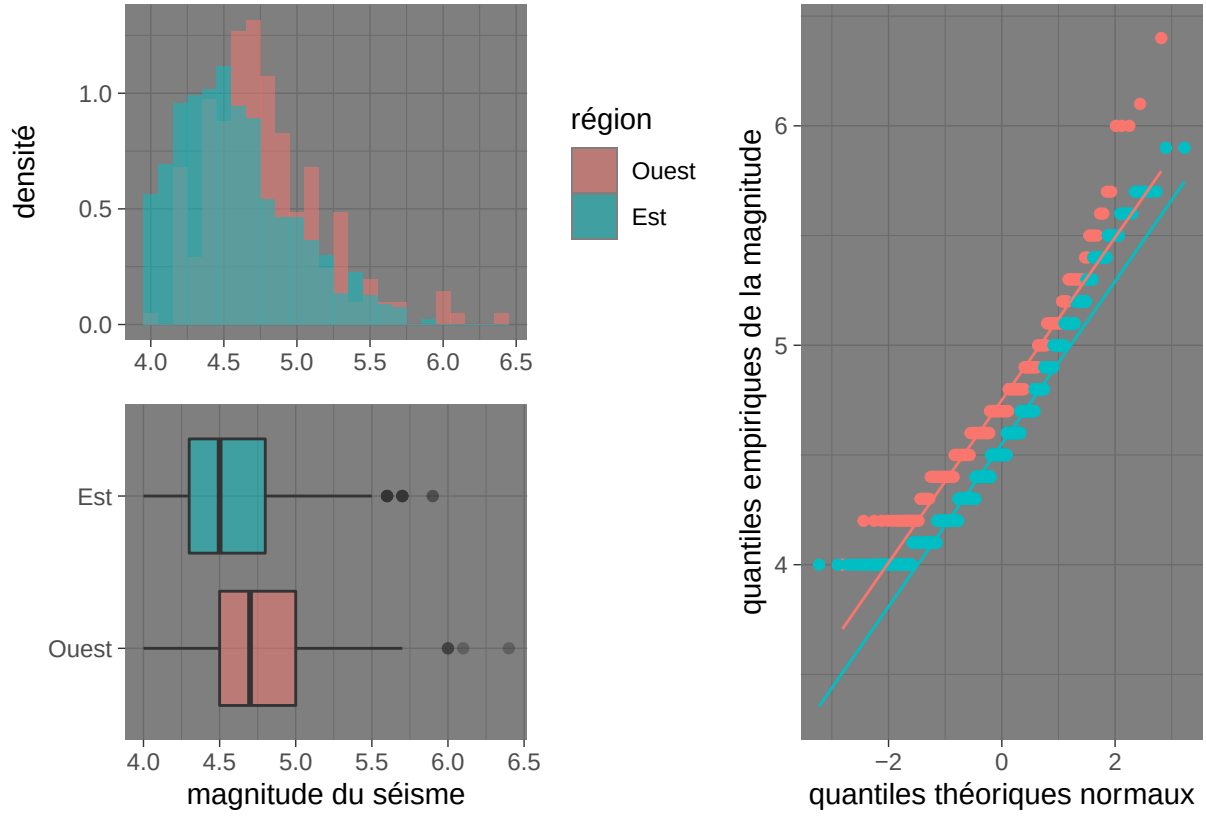
```
courtepointe2Copie <- courtepointe2
courtepointe2Copie[[1]][[1]] <- courtepointe2Copie[[1]][[1]] + theme_dark()
courtepointe2Copie
```



Modifier l'ensemble des graphiques

Pour modifier tous les graphiques en même temps, l'opérateur `&` doit être utilisé.

```
courtepointe2 & theme_dark()
```



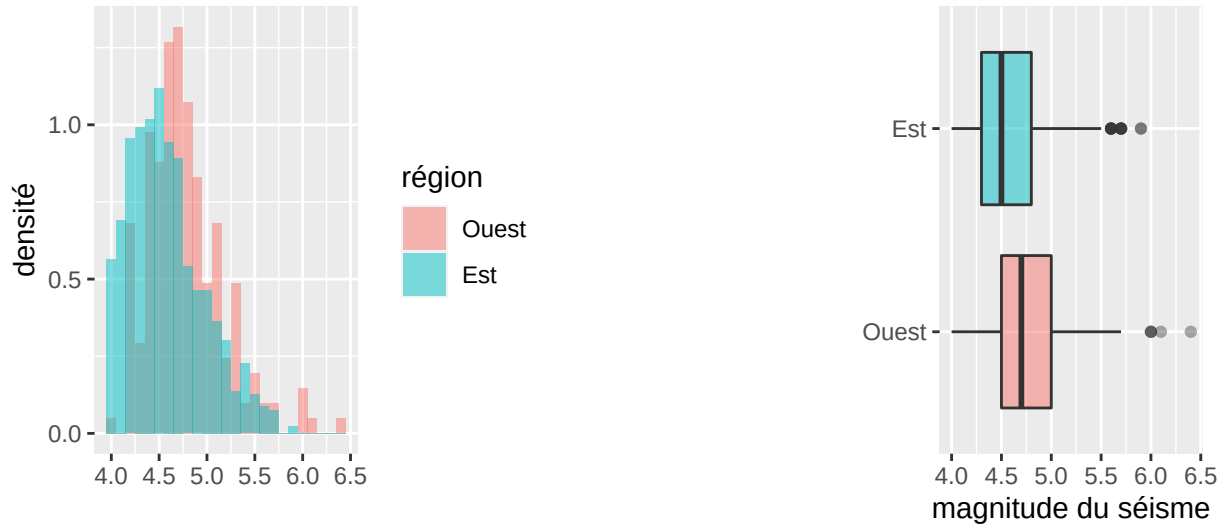
Contrôler la mise en forme d'un `patchwork`

Le package `patchwork` permet également de modifier l'affichage et l'organisation des éléments du «patchwork».

La fonction `plot_spacer`

Il est possible d'ajouter des espaces vides à un «patchwork» avec la fonction `plot_spacer`. La taille de l'espace correspond à la taille d'un graphique.

```
p1 + plot_spacer() + p2
```



Nous détaillons dans les prochaines lignes des façons de moduler davantage la taille des fenêtres pour chacun des graphiques ou des espaces vides d'un «patchwork».

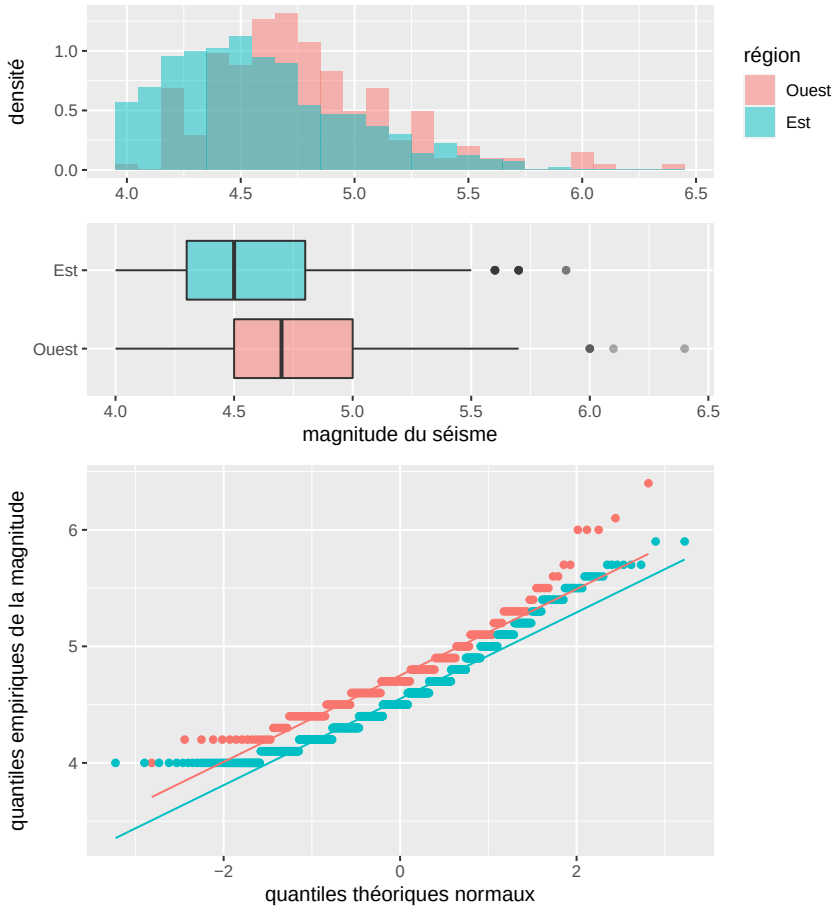
La fonction `plot_layout`

Comme mentionné précédemment, par défaut, `patchwork` rend les fenêtres et les graphiques le plus carrés possible. La fonction `plot_layout` permet à l'utilisateur de davantage contrôler la mise en forme des «patchwork».

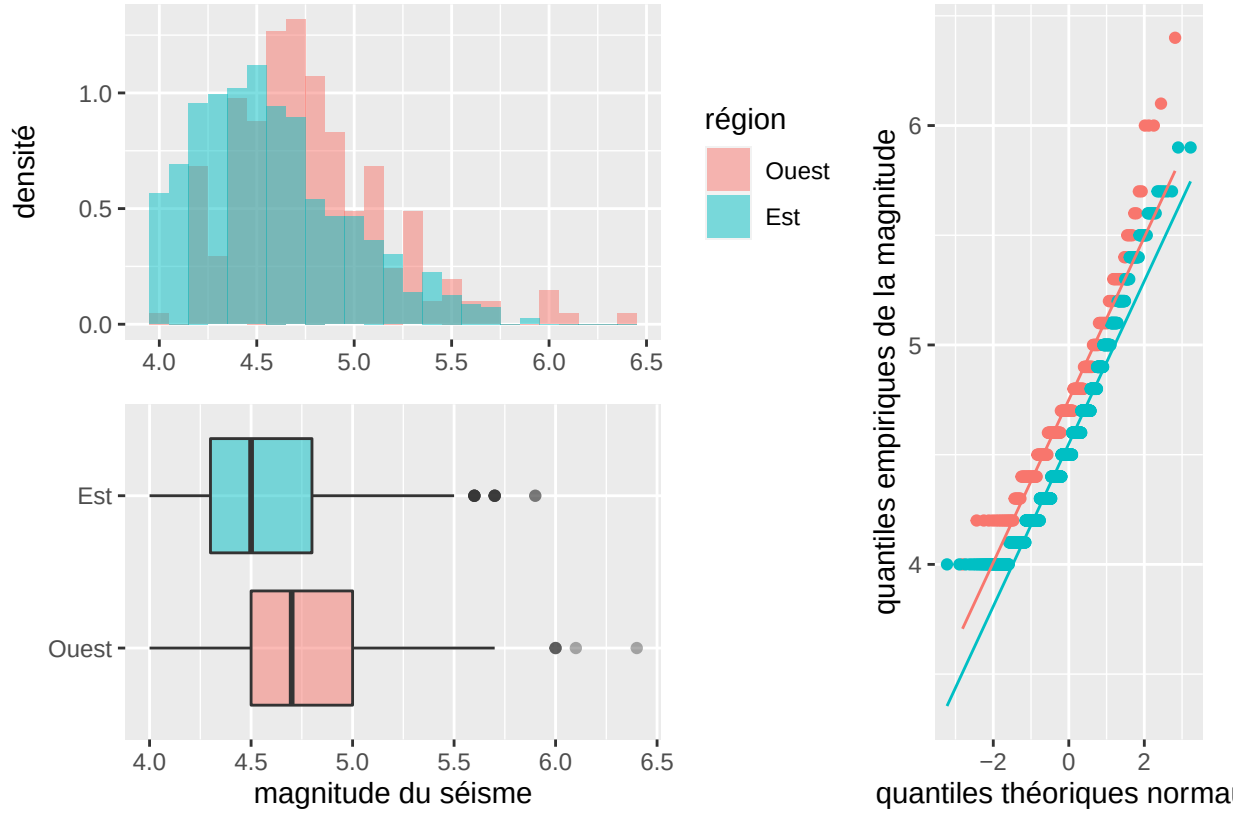
Les arguments `ncol`, `nrow`, `widths` et `heights` de `plot_layout`

Les arguments `ncol` et `nrow` permettent de modifier respectivement le nombre de lignes et de colonnes de la fenêtre. Les arguments `widths` et `heights` permettent de modifier la taille des fenêtres graphiques, soient respectivement la largeur et la hauteur.

```r
# Une colonne plutôt que 2.
courtepointe2 + plot_layout(ncol = 1)
```



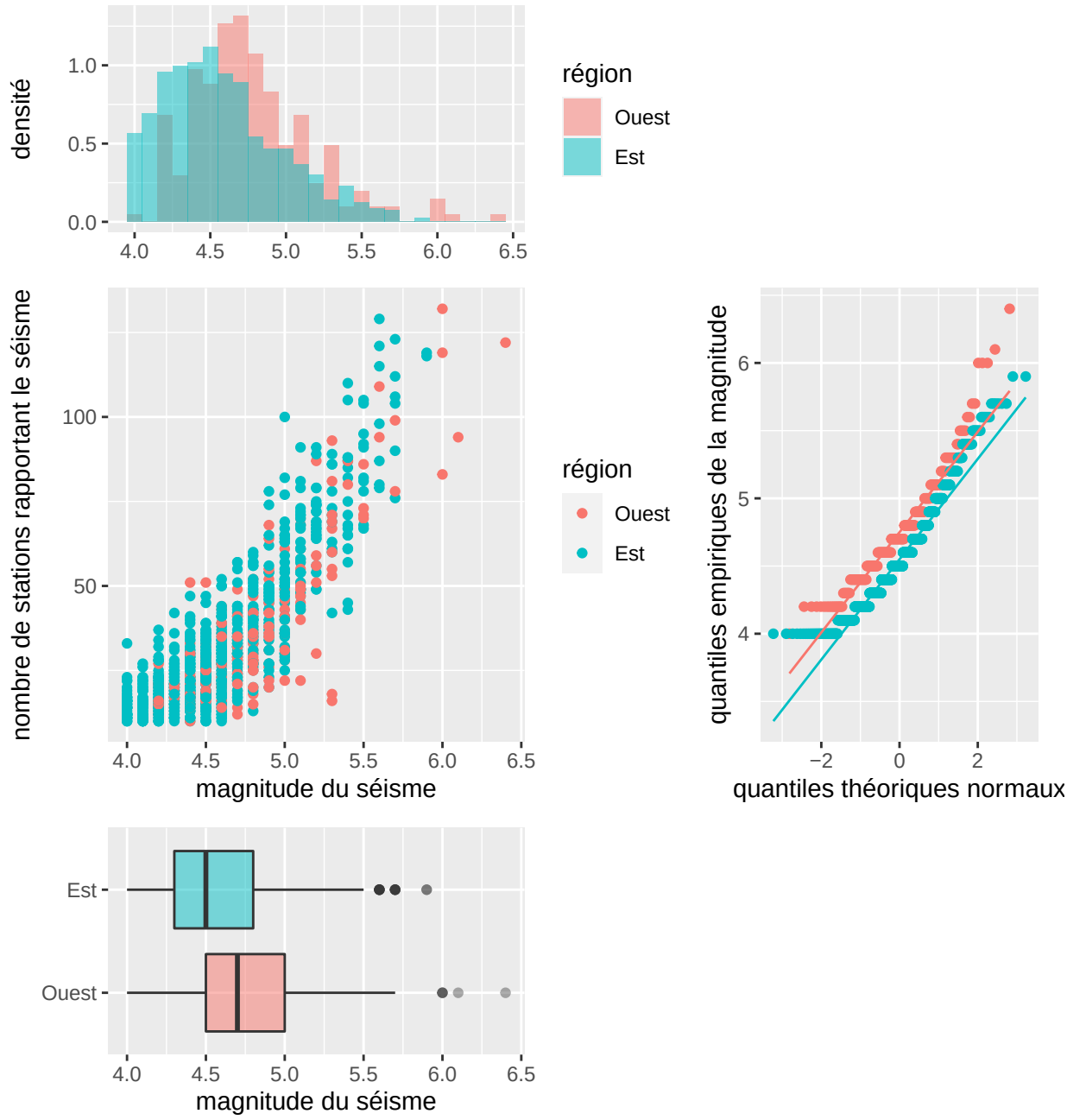
```
# Élargissement de la première colonne par deux fois la deuxième.
courtepointe2 + plot_layout(widths = c(2, 1))
```



L'argument `design`

Il est possible de contrôler davantage la fenêtre des graphiques en créant de façon textuelle ses propres formats de présentation des fenêtres avec l'argument `design` de `plot_layout`. Le cadre d'écriture de ces formats est simple. Des dièses `#` signifient un espace vide comme la fonction `plot_spacer` présenté précédemment et un ensemble de lettres pareil représente l'espace d'un graphique.

```
format1 <- "
AAA##
DDDCC
DDDCC
BBB##
"
p1 + p2 + p3 + p4 + plot_layout(design = format1)
```

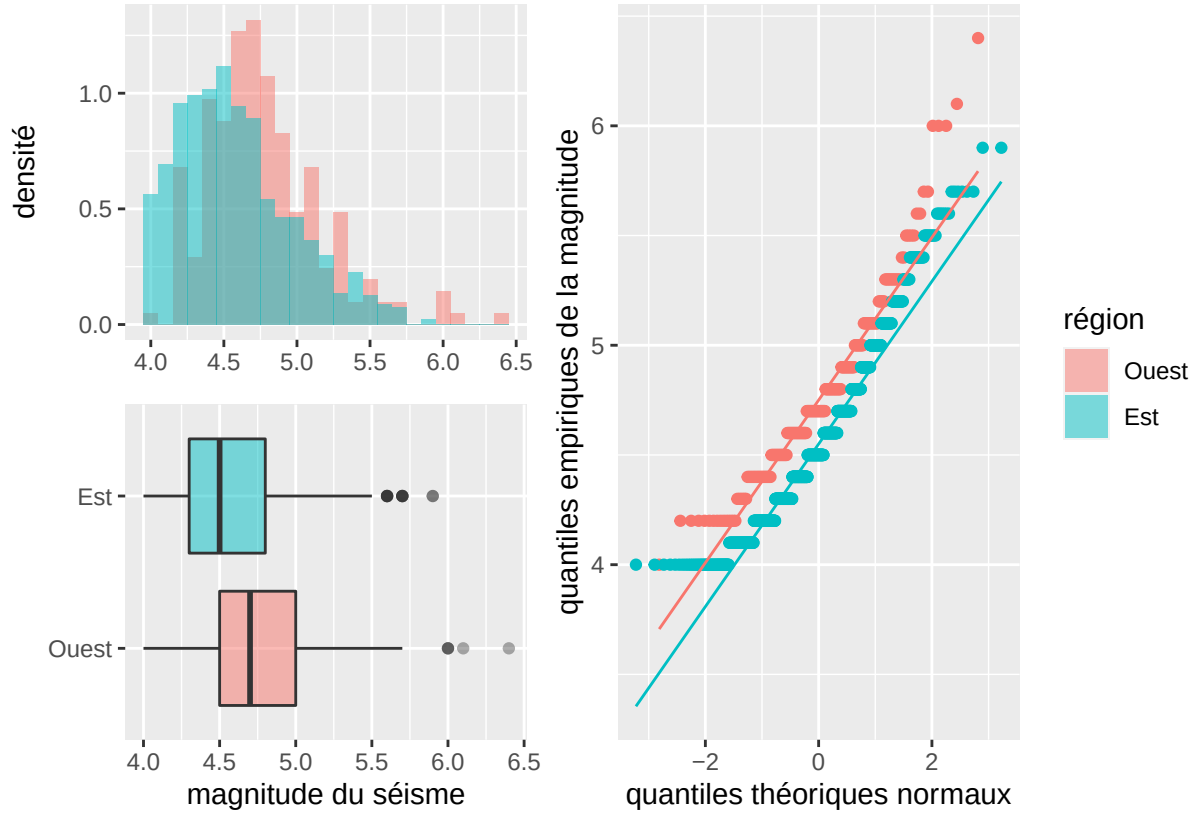


À noter, les graphiques subséquents sont insérés en ordre alphabétique. Dans l'exemple ci-dessus, le premier graphique `p1` a pris la place de l'ensemble des lettres A, le deuxième `p2`, des lettres B, et ainsi de suite jusqu'à D.

L'argument `guides`

L'argument `guides` permet de contrôler l'affichage des légendes.

```
courtepointe2 + plot_layout(guides = "collect")
```

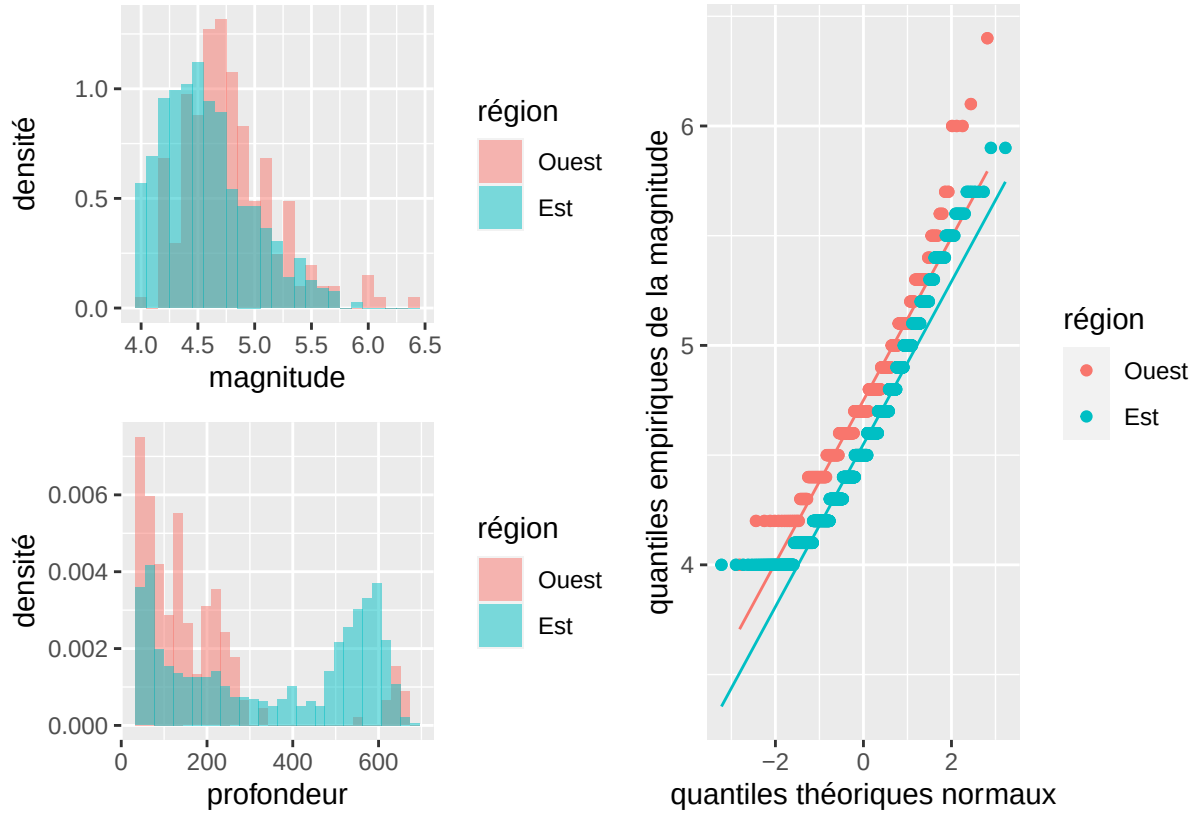


Assigner la valeur `"collect"` à et argument permet de rassembler les légendes du côté droit du «patchwork» et d'en afficher une seule si une légende est répétée.

```
# p1 avec les données de la profondeur plutôt que de la magnitude
p1_profondeur <- ggplot(
  data = quakes,
  mapping = aes(x = depth, fill = region)
) +
  geom_histogram(
    mapping = aes(y = stat(density)),
    alpha = 0.5,
    position ="identity",
    center = 0
  ) +
  labs(y = "densité") +
  labs(x = "profondeur" ) +
  labs(fill = "région")
# p3 avec la légende
p3_avec_legende <- p3 +
  stat_qq(show.legend = TRUE) +
  labs(colour = "région")
```

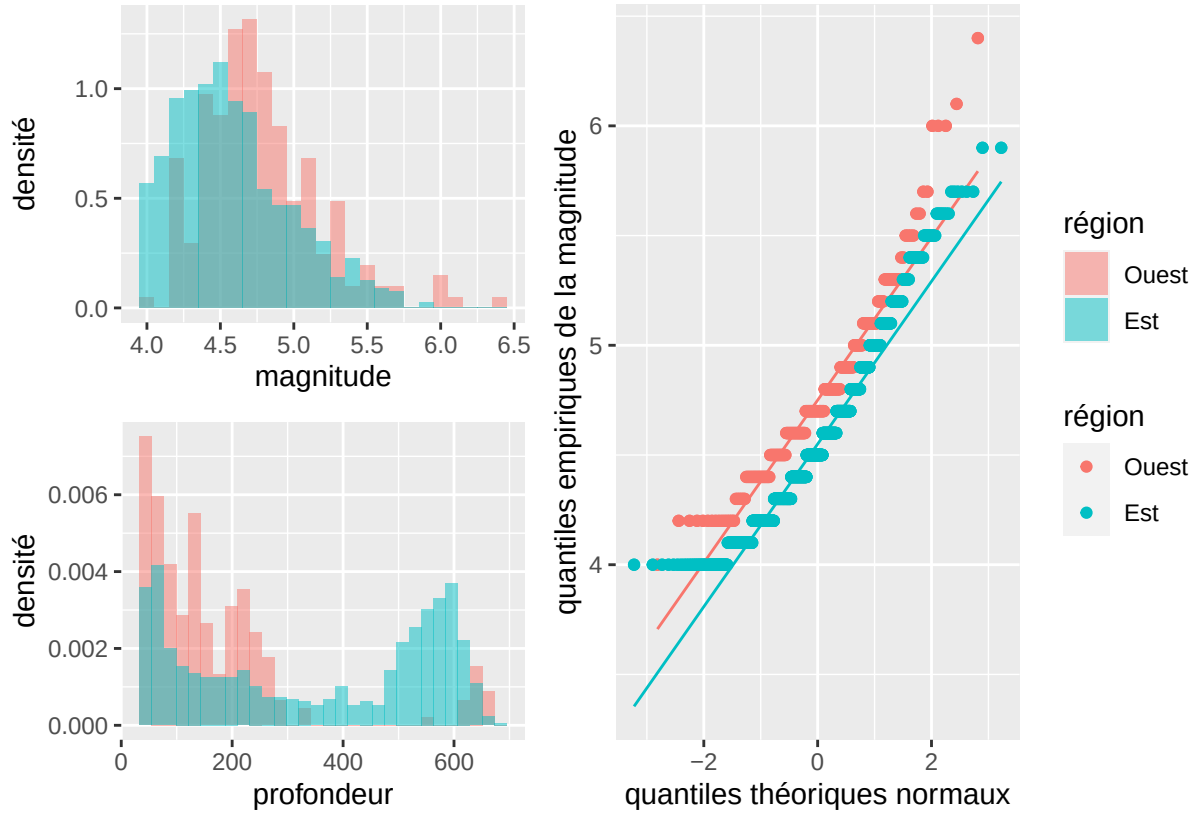
```
# un «patchwork» sans guides = "collect"
(((p1 + labs(x = "magnitude")) / p1_profondeur) | p3_avec_legende)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



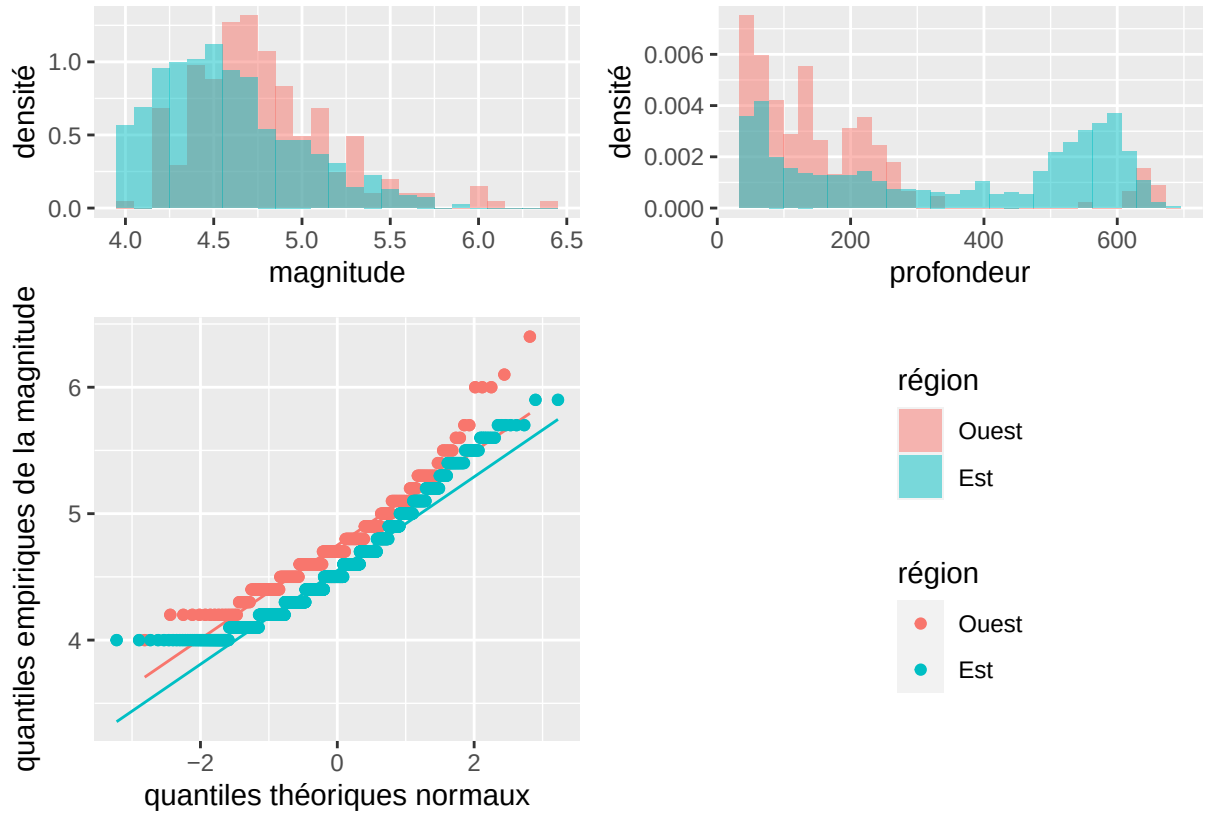
```r
# un «patchwork» avec guides = "collect"
(((p1 + labs(x = "magnitude")) / p1_profondeur) | p3_avec_legende) +
  plot_layout(guides = "collect")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Il est également possible de placer les légendes à l'endroit désiré dans le «patchwork» en utilisant la fonction `guide_area` qui s'utilise similairement à `plot_spacer`. Les légendes présentes dans le «patchwork» seront toutes regroupées à l'endroit où `guide_area` est placé.

```
(p1 + labs(x="magnitude")) +
  p1_profondeur +
  p3_avec_legende +
  guide_area() +
  plot_layout(guides = "collect", heights = c(1,2))
```



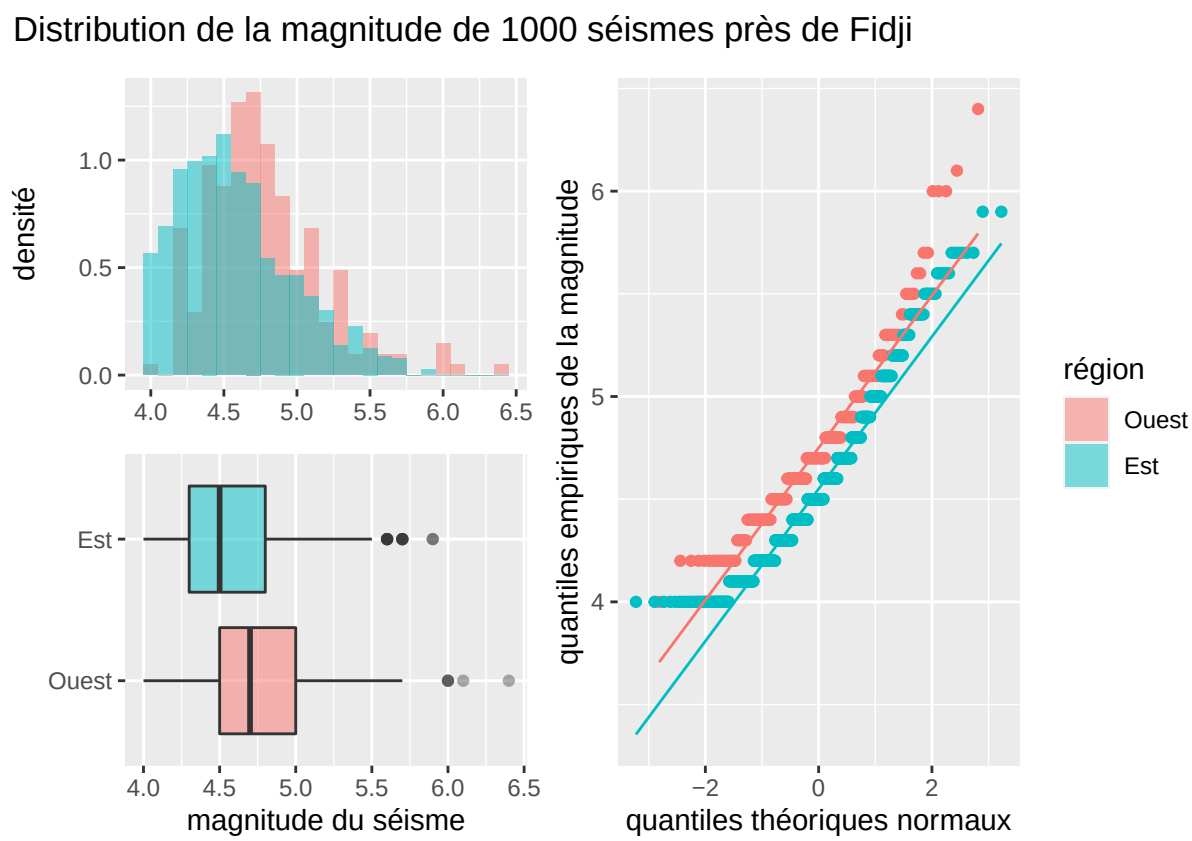
Ajouter des éléments textuels à un `patchwork`

Il est également pertinent de pouvoir ajouter différents éléments textuels à un «patchwork» tels qu'un titre, un sous-titre, des notes et des identifiants aux graphiques. Ceci est possible avec la fonction `plot_annotation` de `patchwork`.

La fonction `plot_annotation`

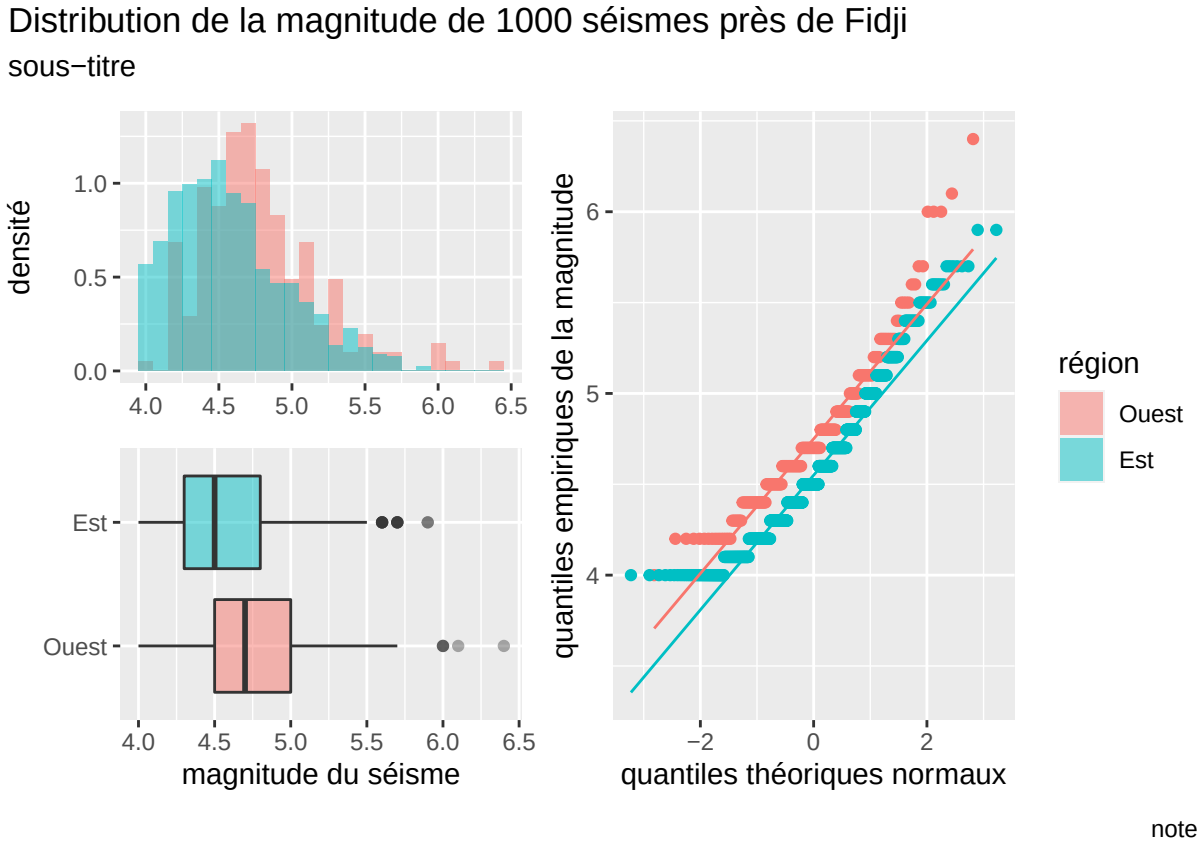
Pour ajouter un titre, il faut simplement utiliser l'argument `title` de `plot_annotation`.

```
courtepointe2T <- courtepointe2 +
  plot_layout(guides ="collect") +
  plot_annotation(title = "Distribution de la magnitude de 1000 séismes près de Fidji")
courtepointe2T
```



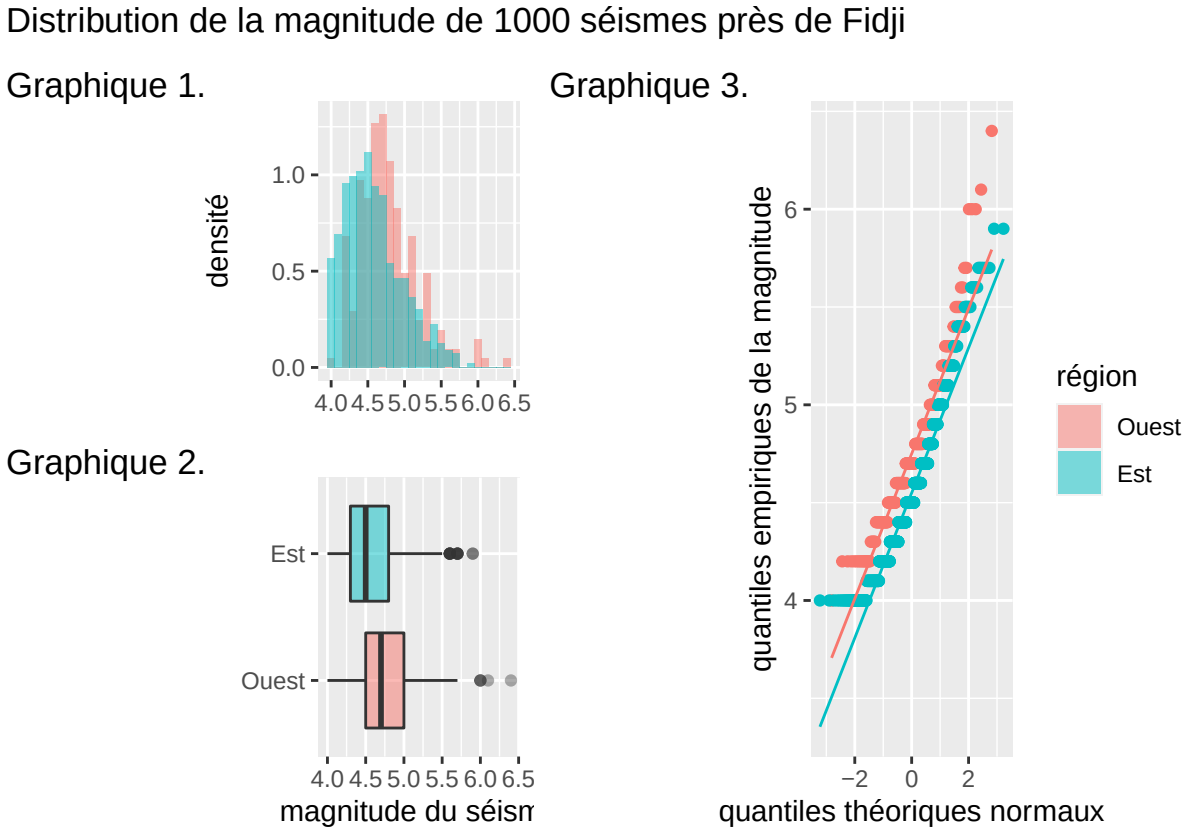
Pour ajouter un sous-titre et une note, les arguments à utiliser sont `subtitle` et `caption`. Le sous-titre apparaît sous le titre principal avec une police de caractère inférieure à celui-ci. Il peut être utilisé pour donner des précisions sur le titre. La note, quant à elle, apparaît avec une police encore plus petite, mais cette fois-ci en bas à droite de la page. Elle peut servir à commenter les graphiques, donner un avertissement ou autre.

```
courtepointe2T + plot_annotation(subtitle = "sous-titre", caption = "note")
```



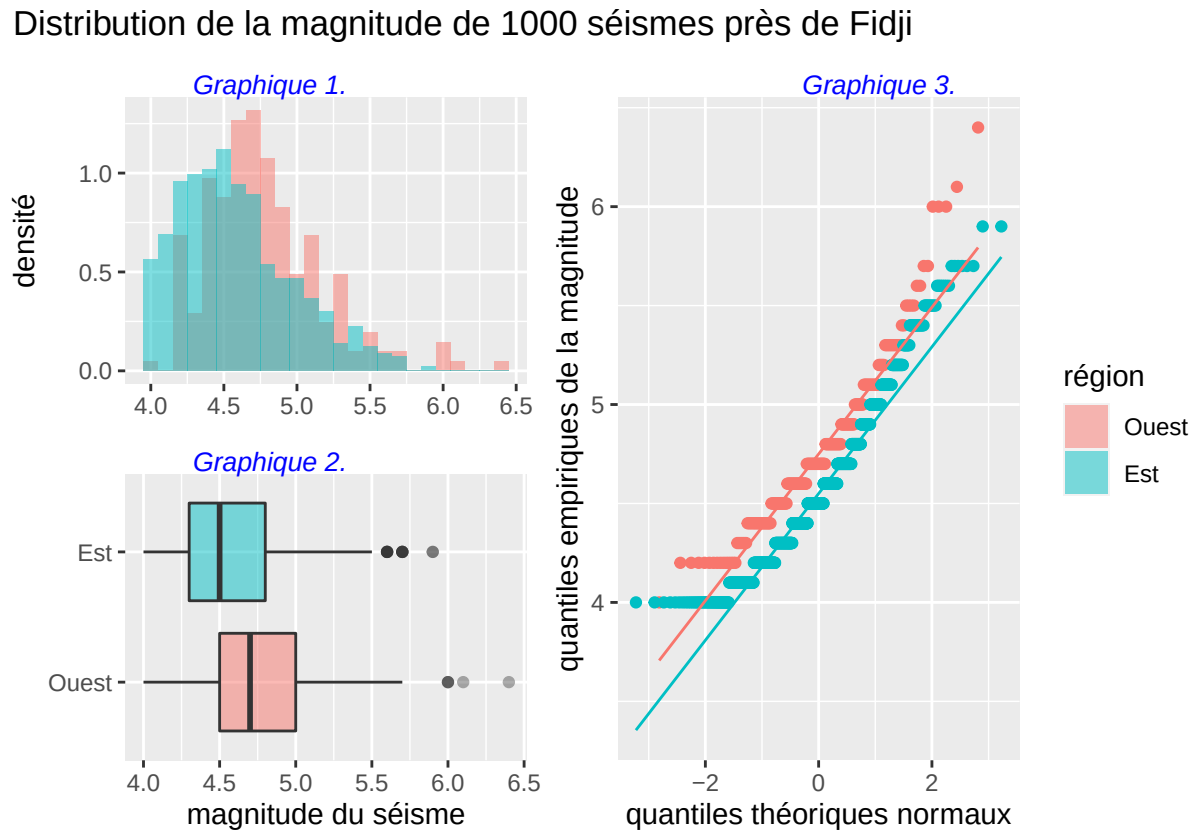
Il est également facilement possible d'identifier les graphiques avec les arguments `tag_levels`, `tag_prefix` et `tag_suffix`. Ce qui est intéressant avec `tag_levels`, c'est que cet argument va automatiquement identifier chacun des graphiques sous forme d'énumération. Il prend en entrée `a` (l'alphabet en minuscule), `A` (l'alphabet en majuscule), `1` (les nombres), `i` (les chiffres romains en minuscule) et `I` (les chiffres romains en majuscule).

```
courtepointe2t_identifier <- courtepointe2T +
  plot_annotation(tag_levels = "1", tag_prefix = "Graphique ", tag_suffix = ".")
courtepointe2t_identifier
```



Par défaut, les identifiants des graphiques sont placés à gauche du graphique en question et dans leur propre colonne. Si nous voulons plutôt qu'ils soient affichés au-dessus de la région graphique, la fonction `theme` doit être utilisée avec l'argument `plot.tag.position`, puis pour modifier la taille et l'apparence de l'identifiant, l'argument `plot.tag` est utilisé.

```
courtepointe2t_identifier & theme(
  plot.tag.position = "top",
  plot.tag = element_text(size = 10, color = "blue", face = "italic")
)
```



La fonction `ggMarginal` de `ggExtra`

La fonction `ggMarginal`est utile pour ajouter des graphiques marginaux en `x` et en `y` à un diagramme de dispersion `ggplot2` existant. Les graphiques marginaux peuvent être de plusieurs types.

Démarrage

Le package `ggExtra` doit être installé, puis chargé dans la session R.

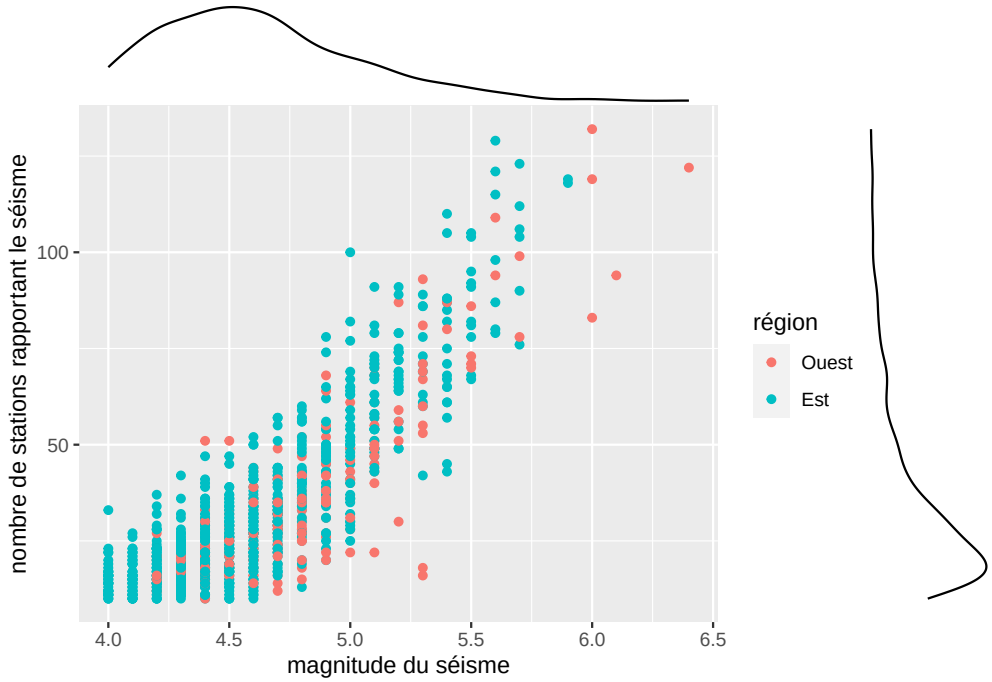
```
# install.packages("ggExtra")
library(ggExtra)
```

Graphiques possibles

La fonction `ggMarginal` est facile d'utilisation. Seulement un argument est nécessaire pour pouvoir l'utiliser. En effet, `ggMarginal` demande seulement un diagramme de dispersion `ggplot2` en entré. Les arguments supplémentaires serviront à modifier le ou les graphiques en marges

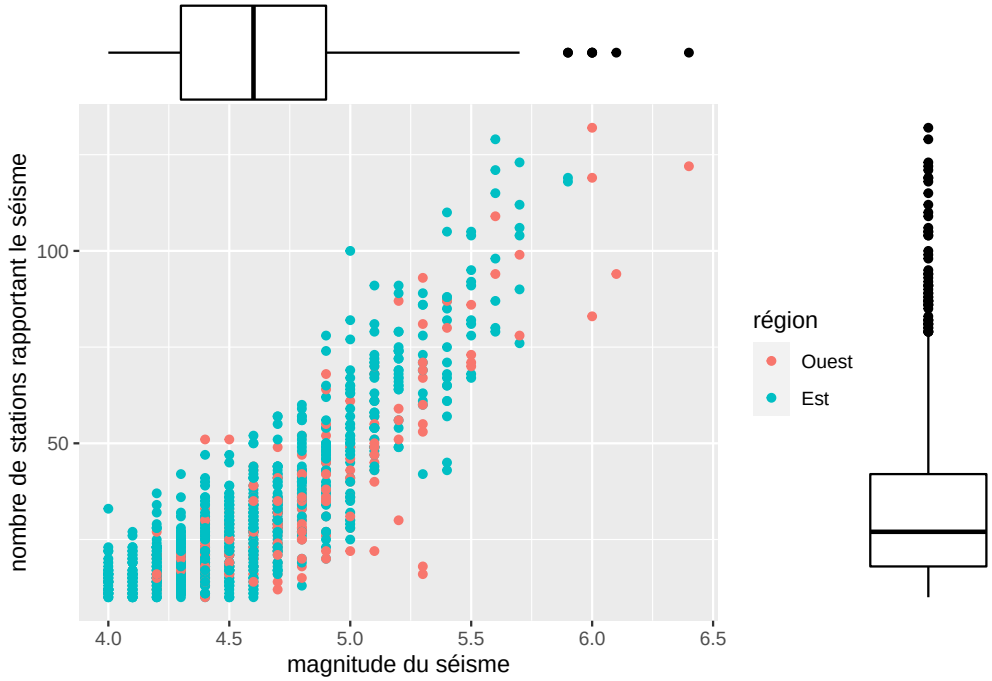
Par défaut, la densité marginale est produite.

```
ggMarginal(p4)
```



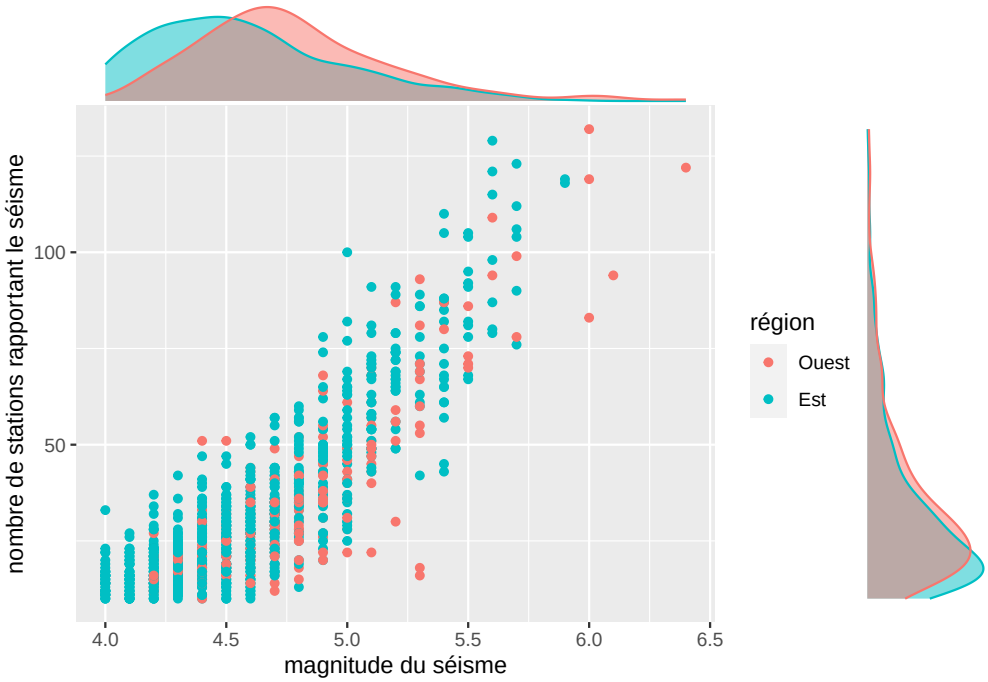
L'argument `type` permet de sélectionner les autres types de densité marginale. Les types possibles sont : densité, histogramme, diagramme en boîte, diagramme en violon et «densigramme». (Un «densigramme» est une juxtaposition de la densité sur l'histogramme).

```
ggMarginal(p4, type = "boxplot")
```



Il est également possible d'afficher les graphiques marginaux selon la catégorie d'un facteur.

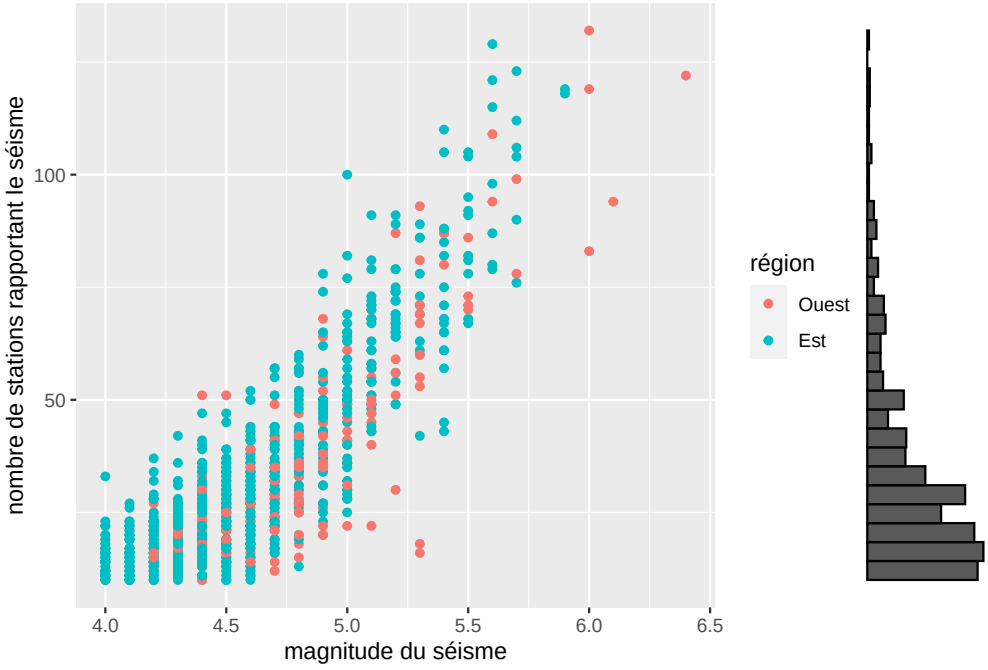
```
graphggM <- ggMarginal(p4, groupColour = TRUE, groupFill = TRUE)
graphggM
```



Modifier ou afficher un seul des graphiques marginaux

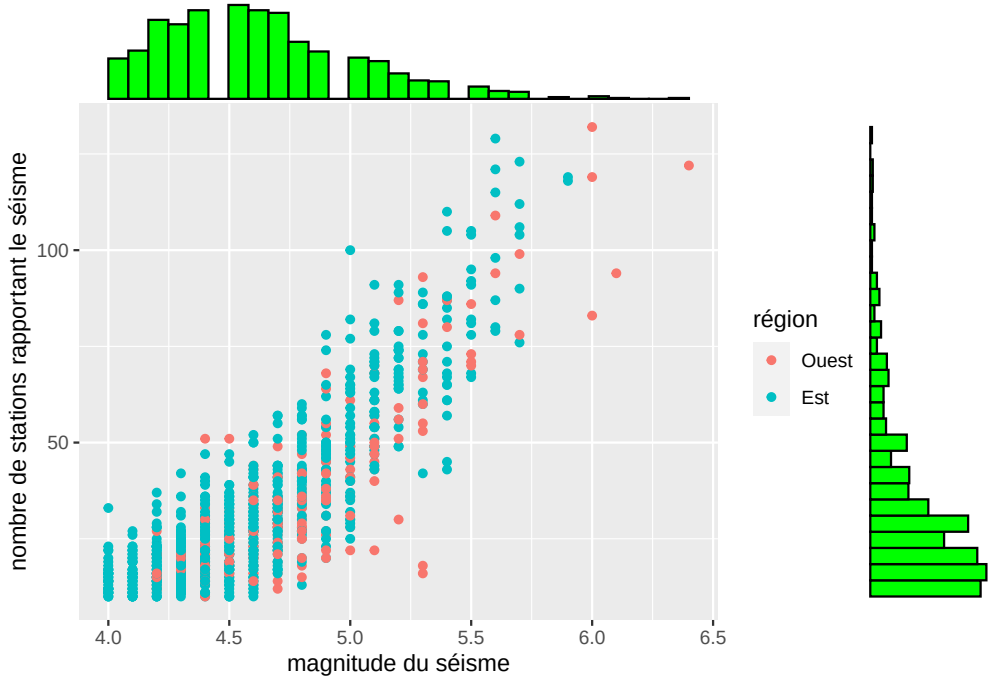
Il est possible d'afficher un seul des graphiques marginaux à la fois. Pour ce faire, il faut donner la valeur `"y"` ou `"x"` à l'argument `margins` pour identifier la marge dans laquelle un graphique marginal doit être affiché

```
ggMarginal(p4, type = "histogram", margins = "y")
```



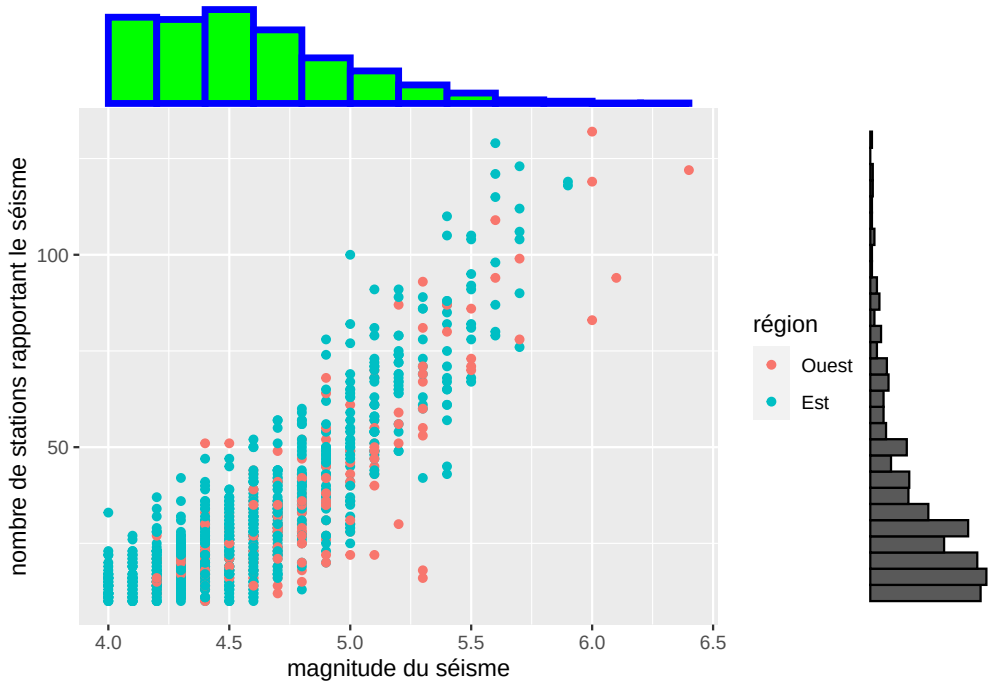
Nous pouvons modifier l'apparence des graphiques marginaux. Par exemple, nous pourrions donner une nouvelle couleur aux histogrammes en utilisant l'argument `fill` .

```
ggMarginal(p4, type="histogram", fill = "green")
```



Si nous voulions afficher les deux graphiques marginaux, mais seulement attribuer cette nouvelle couleur à un des deux graphiques, ce serait aussi possible. En effet, il est possible d'utiliser le `xparams` et le `yparams`. Comme leurs noms laissent entendre, ceux-ci nous permettent de sélectionner les paramètres qui seront appliqués à chacune des marges.

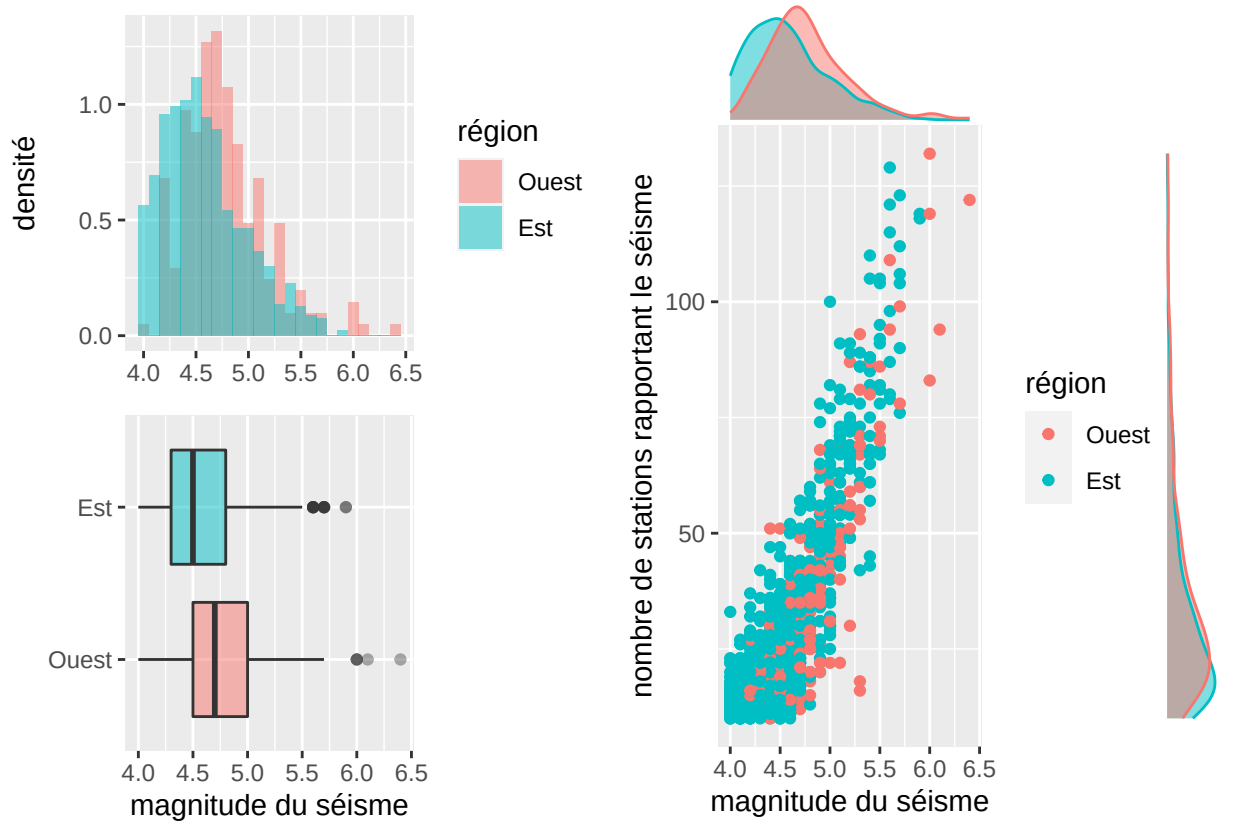
```
ggMarginal(
  p4,
  type = "histogram",
  xparams = list(binwidth = 0.2, size = 1.5, color = "blue", fill = "green")
)
```



Combinaison de `patchwork` et de `ggMarginal` de `ggExtra`

Nous pouvons insérer des graphiques de `ggMarginal` dans un «patchwork».

```
(courtepointe | graphggM ) + plot_layout(widths = c(1, 2))
```



Commentaires et observations sur le package `patchwork` et la fonction `ggMarginal`

Le principal avantage du package `patchwork` est qu'il est très intuitif. Grâce aux opérateurs `|`, `/` et `+`, il est facile de placer les graphiques dans une composition complexe et parfaitement alignée que nous sommes capables de visualiser avant même d'exécuter le code. Nous avons pu comprendre les bases du package rapidement et sans difficulté. Ceci nous aurait permis de l'utiliser directement sans pousser plus nos recherches si nous l'avions désiré. Il ne nécessite pas une compréhension exhaustive de ses fonctionnalités pour pouvoir en retirer de nombreux avantages.

Un désavantage de `patchwork` est qu'il ne permet pas, à lui seul, d'ajouter du contenu «non `ggplot2`» tel que du texte ou des graphiques du système R de base. Le package `cowplot`, qui offre des fonctionnalités équivalentes, a aussi cette limite. Puis, en utilisant `patchwork` en combinaison avec `gridGraphics`, ce n'est pas une tâche aussi intuitive que `patchwork`. Il est assez difficile d'obtenir un alignement exact et aussi esthétique que ceux obtenus lors de l'ajout de contenu `ggplot2`.

De plus, il a été difficile de comprendre le fonctionnement d'ajout de graphiques et de modifications d'un graphique d'un «patchwork». En effet, ces éléments ne sont pas toujours intuitifs. Le résultat obtenu par certaines modifications nous a quelques fois surpris. Par exemple, nous nous attendions qu'une modification que nous avions apportée à un «patchwork» soit appliquée à un certain graphique, mais la modification était appliquée à un ou plusieurs autres graphiques.

Pour ce qui est de la fonction `ggMarginal` du package `ggExtra`, nous avons trouvé qu'elle était facile à utiliser. Il faut simplement porter attention au fait que `ggMarginal`prend le graphique en entrée ce qui diffère de la syntaxe traditionnelle de `ggplot2` qui propose normalement d'ajouter une «couche» à l'aide de l'opérateur `+`.

Références

Sources principales pour le package `patchwork` et la fonction `ggMarginal` de `ggExtra`

https://patchwork.data-imaginist.com/

https://github.com/daattali/ggExtra

Notes du cours STT-4230

https://stt4230.rbind.io/communication_resultats/graphiques_r/

https://stt4230.rbind.io/communication_resultats/redaction_r_markdown/

Références utiles pour la construction de graphiques en `ggplot2`

https://ggplot2.tidyverse.org/reference/geom_boxplot.html

https://ggplot2.tidyverse.org/reference/geom_histogram.html

http://www.cookbook-r.com/Graphs/Plotting_distributions_(ggplot2)/

https://www.datanovia.com/en/fr/blog/gpplot-comment-supprimer-la-legende/#ggplot-sans-legende

https://ggplot2.tidyverse.org/reference/geom_point.html