Auto Multiple Choice - Feature # 214: Insert question from group without replacement and repopulate the group once empty

Status:	Closed	**Priority:**	Normal
Author:	Loïc Cerf	**Category:**	
Created:	09/26/2013	**Assignee:**	
Updated:	10/31/2014	**Due date:**	
Description:	It would be nice to have, in the LaTeX package, a modified \insertgroup command that would not only insert the questions but also remove them from the group. If this modified command is asked to insert more questions than now available in the group, it would insert the remaining ones, repopulate the group (the original group must therefore be kept in memory) and insert the additional questions. In this way, all questions are used the same amount of times and there is a guarantee that no two consecutive exams have the same question in the group.		

History

09/26/2013 06:12 pm - Loïc Cerf

If this feature is implemented, the \shufflegroup command would probably have to be modified as well: both the reducing group and the original one (used to repopulate once empty) should be shuffled. If the latter is not done the questions to be picked in addition to those that remained (in the now exhausted group) would always be the same.

09/26/2013 08:17 pm - Loïc Cerf

The solution I proposed is not correct when inserting more than one question and shuffling the questions: the same question could be repeated (for instance when inserting two questions and only one has not been asked yet).

The implementation of the feature would instead require:
the group with the still unasked questions;
the *complementary* group with the questions that have already been asked (not *all* questions).

When the users asks for less questions than available in the first group, those questions are inserted and then moved to the second group.

When more questions are to be inserted than available in the first group, all questions in this group are inserted, the two groups are swapped and the process is repeated for the additional questions.

Again, the \shufflegroup command would have to apply on both the group of unasked questions and the complementary group.

Sorry for the multiple messages: I should have thought it through before posting the feature request.

09/26/2013 11:22 pm - Alexis Bienvenüe

Or perhaps, a command that can insert a given number of items from a group, starting from where the last call stopped, and recycling if necessary. Here we need only a counter to keep in mind the index of the last item inserted.
Do you think this could make the job?

09/26/2013 11:25 pm - Loïc Cerf

It won't work if \shufflegroup is used as well.

09/26/2013 11:49 pm - Alexis Bienvenüe

So you also need a modified @shufflegroup@ that shuffles the items before the current counter and the items after.
However, if you shuffle at each exam, sometimes (perhaps not so often, I agree) two consecutive exams will have same questions (for example when you finish to insert all group's questions and then shuffle, the next to insert can be the same as the ones that has been inserted right before). I think this is the same with your solution.

09/27/2013 01:29 am - Loïc Cerf

I agree: having a counter and shuffling before and after it has the exact same effect as that of the solution I proposed. And, yes, there can be two consecutive exams with the same questions (although less often than today). The advantage of having all questions used the same amount of times (+/- 1) remains intact. In a context where there are at least as many elements in the group as there are students, that means a different question for everyone. A neat feature in my opinion.

The design problem with this feature would be the integration with the normal \insertgroup command applied on the same group. With the solution I proposed, it would insert questions after your counter... and there may not be enough remaining questions. With your solution, \insertgroup would insert questions before the counter... and they can go up to or even beyond the counter.

Using, on a same group, both an \insertgroup with replacement and an \insertgroup without replacement is a weird use case. However, if the user is allowed to do so, I believe you would like it to work in a sane way. As far as I can see, there is no easy solution that would involve a modified \insertgroup command.

However, I think I have a valid idea to circumvent the problem: every group where questions are to be inserted without replacement have to be specifically defined in this way (a \noreplacement command that would take the group as argument). \insertgroup would then behave accordingly (no need for an additional version of it). \copygroup would also have to behave accordingly: after the copy, the questions are considered consumed (the counter is moved). The nature of the group where the questions are copied should not depend on the group where they come from. It should depend on whether it was declared with or without replacement, i.e., whether it was in argument of a \noreplacement command.

It looks like the implementation of the feature suddenly became much more complicated that I thought it would be!

09/27/2013 04:37 pm - Loïc Cerf

Still thinking out loud:
* When \noreplacement is used, the counter could initially point to the end of the group. In this way, the \shufflegroup command only needs to shuffle what is before the counter/iterator (the first time: the whole group).
* For even better performances (in case of very large groups), the \shufflegroup command applied on a group with no replacement could simply set a flag. When \insertgroup (or \copygroup) is to insert (or move) more questions than available after the counter, the flag would be checked and, if present, the shuffle applies before the counter. Then, the command inserts (or move) what is after the counter, moves the counter to the beginning of the group and finishes the work to reach the number of questions the user wants.
* I do not see why a user would prefer the current behavior to the behavior I propose (no replacement in the groups). After all, if she wrote questions, she wants them all to be asked. And because exams are usually distributed in a class, it is nice to have a lower probably of two consecutive exams having the same questions. Also, as explained in the previous items, the generation of the DVIs can be faster in case of large groups that are shuffled (probably an insignificant point though). In the end, you may want to rewrite the current commands to change their default behaviors. That is less work than trying to make both behaviors available. And that would mean no additional \noreplacement command.

I would propose my help on that... but I am pretty sure I would do a bad job at it (I still do not get how loops work in the .sty!). Also, I am currently busy with some tough C++ code and working in parallel on several programs is not efficient (you spend more time trying to remember how things work than implementing new things). Do you consider the feature is worth the work? Would you rather close the feature request with a "will not be implemented"?

Because you seem to prefer a counter to a group split into two (the asked and the unasked questions), I now explain my thoughts with the counters. However, as far as I understand automultiplechoice.sty, I tend to believe it would be easier to implement the feature with split groups: a variable seems to contain the size of the group (no need for subtractions with the counter to know how many unasked questions are available), the \shufflegroup command could be left unchanged (as explained in the second item above, it is applied on the asked questions only), swapping the two groups (see comment #2) is only swapping their names, etc.

PS: Au fait, je suis français et je constate que le développement semble être majoritairement (sinon exclusivement) fait par des francophones (vous uniquement ?). Il est donc peut-être inutile voire dommageable (malentendus) de continuer cette conversation en anglais. C'est comme vous voulez.

09/27/2013 09:31 pm - Loïc Cerf

The second item is wrong (the end of the group stays unchanged). It should be:
* For even better performances (in case of very large groups), the \shufflegroup command applied on a group with no replacement could simply set a flag. When \insertgroup (or \copygroup) is to insert (or move) more questions than available after the counter, the command inserts (or moves) what is after the counter, moves the counter to the beginning of the group and finishes the work (the counter advances) to reach the number of questions the

user wants. Then the flag would be checked and, if true, the shuffle applies before and after the counter. Finally the flag is set to false.

If you plan to implement the feature (again, I really do not think it would be a good idea that I help with the code), I can write you a formal pseudo-code for every command (either with split groups or counters).

09/27/2013 09:39 pm - Alexis Bienvenüe

> Il est donc peut-être inutile voire dommageable (malentendus) de continuer cette conversation en anglais
Surtout avec mon anglais, en effet.

Pourquoi je préfère un compteur plutôt que la gestion des deux groupes ? J'ai l'impression que ça donnera des opérations plus rapides. Pas besoin de transférer des questions d'un groupe à l'autre à chaque fois...
La question du mélange. Votre idée d'un drapeau est intéressante. Autre variante : après être revenu au début lors de l'utilisation du groupe, si le mélange a été demandé, on mélange le groupe dans sa totalité et on remet le compteur à zéro. Ça fera sans doute des questions qui apparaîtront par malchance plus souvent que d'autres. Mais sinon, si par exemple on ne mélange que le début jusqu'au compteur et la fin après le compteur qui est souvent vers le début, le début reste souvent au début. Est-ce embêtant ? Je n'arrive pas à me faire une idée.
Remplacement plutôt qu'option ? Je n'ai pas envie de changer le comportement, car ça poserait des problème de compatibilité ascendante. Si on reprend un ancien projet, il sera ordonné autrement par exemple lors d'un changement de barème... et tout est cassé. Je préfère laisser le choix, quitte à ajouter une option globale du paquet qui demande à ce que tous les groupes soient sans remplacement par défaut.

> Do you consider the feature is worth the work?
Oui. Et ça me paraît tout à fait faisable. Mais j'essaye de trouver la meilleure option avant de me lancer.

09/27/2013 11:25 pm - Loïc Cerf

Vous avez certainement raison quant aux performances. Je pensais que les temps de copie puis suppression de questions étaient négligeables.

Pour ce qui est du mélange, il s'agit d'un choix de priorité :
* en mélangeant avant et après le compteur il est garanti que chaque élément sort la même quantité de fois (+/- 1) et il y est moins probable d'avoir deux copies consécutives avec les mêmes éléments: ceux avant le compteurs ne sortiront pas dans la copie suivante.
* en mélangeant tout, le mélange est meilleur parce que les éléments au début n'ont pas de raison d'y rester et donc de sortir ensemble (dans une même copie) avec une plus grande probabilité. Par exemple si l'utilisateur veut trois éléments d'un groupe qui en contient quatre (ou plus généralement 3k+1, k>0), les deux premiers éléments, qui sortent ensemble dans la première copie, vont également sortir ensemble dans la seconde (ou la k+1 ième) copie.

C'est assez laid d'avoir un mélange imparfait. J'ai donc maintenant tendance à préférer la seconde option (oui, je passe mon temps à changer d'idée !). Par ailleurs, l'implémentation de cette seconde option semble plus simple :
* la commande \shufflegroup met le drapeau à vrai;
* après que \insertgroup (ou \copygroup) ait fait un tour des éléments, le drapeau est consulté et, si il est à vrai, le mélange du groupe entier a lieu (exactement comme il a lieu aujourd'hui) et le compteur est mis à 0 (simplement pour maximiser le temps jusqu'au prochain mélange).

Il faudra ne pas oublier le cas du premier appel à \shufflegroup. Ce que je viens d'écrire ne le prend pas en compte. Plus précisément le code suivant insérerait forcément le premier élément (mais se comporterait correctement si les deux premières commandes sont inversées) :

```
\noreplacement{group}
\shufflegroup{group}
\insertgroup{group}
```

La commande \noreplacement pourrait initialiser le compteur à la fin du groupe mais le problème demeure si des éléments sont ensuite ajoutés au groupe :

```
\noreplacement{group}
\copygroup{other_group}{group}
\shufflegroup{group}
\insertgroup{group}
```

Il faudrait donc en plus que, comme \noreplacement, la commande \copygroup place le compteur à la fin du groupe... sauf que le cas d'utilisation suivant ne fonctionne plus comme l'utilisateur le souhaite (il semble vouloir que les derniers éléments ajoutés restent à la fin) :
\noreplacement{group}
\shufflegroup{group}
\copygroup{other_group}{group}
\insertgroup{group}

Il faut donc, avant de copier les éléments, que la commande \copygroup regarde s'il s'agit d'un groupe sans remise et avec une demande de mélange (drapeau à vrai). Si c'est le cas :
le mélange doit avoir lieu;
le drapeau est remis à faux.
Ensuite et même si le test précédent n'est pas passé parce que le drapeau était à faux :
les éléments sont copiés;
le compteur est mis à la fin du groupe.

Est-ce que cela résout le casse-tête pour tous les cas d'utilisation non absurdes ? Peut-être...

09/28/2013 03:56 am - Loïc Cerf
En fait le problème de mauvais mélange existe dans les deux solutions. Dans l'exemple que je donnais (trois éléments pris dans un groupe à quatre éléments), les deux premiers éléments, qui sortent ensemble dans la première copie, vont également sortir ensemble dans la seconde (ou la k+1 ième) copie. Avec votre solution, ils sortent même dans le même ordre.

La nouvelle solution qui me passe par la tête (d'ici que je comprenne pourquoi elle ne fonctionne pas non plus) est la suivante : lorsqu'il ne reste pas suffisamment d'éléments après le compteur, les éléments avant le compteur sont mélangés, les élément sont insérés (en faisant un tour) et les éléments après le compteur sont mélangés. Je crois que l'on combine là les avantages des deux solutions !

Il restait aussi un cas d'utilisation problématique. Le code suivant (où, cette fois, je n'oublie pas les "[1]" après \insertgroup) insérerait deux fois la même question :
\noreplacement{group}
\insertgroup[1]{group}
\copy_group{other_group}{group}
\insertgroup[1]{group}

Voilà, en détail, ma nouvelle solution :

* \noreplacement sert à définir un groupe (en argument) où les éléments sont pris sans remise. Un drapeau indiquant si le groupe doit être mélangé est défini à faux. Un compteur est défini et pointe après le dernier élément.

* \shufflegroup met le drapeau à vrai.

* \insertgroup va insérer le nombre d'éléments désirés. Avant toute chose, la commande teste si il y a suffisamment d'éléments disponibles à partir du compteur. Si il y a suffisamment d'éléments, le nombre d'éléments demandé est inséré (à partir du compteur) et le compteur est avancé d'autant. Si il n'y a pas suffisamment d'éléments après le compteur, le drapeau est consulté. Si il est à faux :
1. les éléments placés à partir du compteur sont insérés;
2. les éléments pris depuis le début du groupe complète le compte;
3. le compteur est déplacé après le dernier élément inséré.

Si le drapeau est à vrai, il faut ajouter des étapes 0, 4 et 5 :
0. les éléments précédents le compteur sont mélangés;
4. les élément à partir du compteur sont mélangés;
5. le drapeau est mis à faux.

Il est possible d'économiser des mélanges dans le cas (commun) où, lors de l'appel de la commande, le compteurs pointe après le dernier élément : il

n'est pas nécessaire d'effectuer l'étape 4.

* Si le groupe sans remise est le premier argument de \copygroup, le comportement est identique à celui de \insertgroup (décrit ci-dessus) mais les éléments sont copiés dans le second groupe au lieu d'être insérés.

* Si le groupe sans remise est le second argument de \copygroup... je n'ai pas de solution parfaite. Serait-ce possible de refuser la "compilation" avec un message ? Quelque chose du genre "groups in argument of \noreplacement cannot receive more elements (you may want to move the \noreplacement command after the group is completed with the \copygroup command)".

09/28/2013 03:32 pm - Loïc Cerf

Je crois avoir une bien meilleure idée : quatre nouvelles commandes au lieu d'une seule ! Comme cela, ce ne paraît pas être une bonne idée... mais voyez plutôt. Les quatre différentes commandes servent à définir de façon simple le comportement du groupe :

\fixed: le comportement du groupe en argument est celui d'aujourd'hui. Les éléments sont toujours pris du premier au dernier (que ce soit pour être insérés ou copiés dans un autre groupe). Lorsque des éléments y sont copiés, c'est après le dernier.

\withreplacement: le comportement du groupe en argument est celui d'aujourd'hui lorsque chaque commande utilisant le groupe est précédé d'un \shufflegroup. Autrement dit (et pour la documentation), les éléments sont comme des boules que l'on met dans un sac. Pour insérer des éléments (ou les copier dans un autre groupe), où tire aléatoirement le nombre désiré de boules et on les remet dans le sac juste après. L'implémentation sera je crois, comme je viens de la décrire : un mélange du groupe entier à l'appel de \insertgroup ou si le groupe est le premier argument de \copygroup.

\withoutreplacement: le comportement que je trouve intéressant. Il y a deux sacs d'éléments. L'un où les éléments sont tirés, l'autre où les éléments sont gardés après leur insertion (ou leur copie dans un autre groupe). Les éléments sont toujours ajoutés au sac où l'on tire aléatoirement. Si ce sac vient à ne plus avoir suffisamment d'éléments, on échange les deux sacs, c'est à dire que l'on se met à tirer dans le sac où les éléments étaient précédemment gardés et à utiliser l'autre sac pour garder les éléments après leur insertion (ou copie).

\cyclic: un comportement qui doit avoir des applications intéressantes. De façon conceptuelle, il s'agirait de la même chose que \withoutreplacement mais, cette fois, il y a deux files bien rangées au lieu de deux sacs opaques. Les éléments sont toujours ajoutés, dans l'ordre, à la file où l'on prend des éléments. Ces nouveaux éléments seront donc pris après ceux qui étaient déjà là. Après être pris dans la file (pour insertion ou copie dans un autre groupe), les éléments sont déplacés (dans le même ordre) dans la seconde file. Si la file vient à ne plus avoir suffisamment d'éléments, on échange les files, c'est à dire que l'on se met à prendre les éléments dans la file où les éléments étaient précédemment gardés et à utiliser l'autre file pour garder les éléments après leur insertion (ou copie).

Les deux derniers types de groupes seraient implémentés comme je le décrivais dans le précédent message (cette fois : le comportement de la copie vers le groupe est bien défini : plus besoin d'un message d'erreur): \withoutreplacement correspond au drapeau toujours à vrai; \cyclic au drapeau toujours à faux. On évite donc les prises de tête correspondant à des usages bizarres où ce drapeau est parfois à vrai et parfois à faux. \shufflegroup continuerait à avoir son comportement actuelle sur un groupe en argument de \fixed. Sur un groupe défini comme \cyclic, elle ne mélangerait que la file où l'on tire (les éléments à partir du compteur). \shufflegroup n'aurait aucun effet sur un groupe d"fini comme \withreplacement ou \withoutreplacement.

01/29/2014 04:31 pm - Alexis Bienvenüe

- File insert.tex added

- Status changed from New to Feedback

À partir de la révision r1508, vous pouvez trouver la commande @\setgroupmode{groupe}{mode}@, avec _mode_ qui peut valoir @fixed@, @withreplacement@, @withoutreplacement@ ou @cyclic@.
Vous pouvez tester avec quelque chose comme le code joint...

03/06/2014 06:29 pm - Loïc Cerf

Désolé pour le délai... et désolé pour l'incompétence : je ne comprends pas comment installer la dernière révision du projet. J'utilise http://ppa.launchpad.net/alexis.bienvenue/test/ubuntu mais il n'y a pas eu de mise à jour depuis décembre. Dois-je importer le dépôt SVN. Si oui, avec quelle commande (SVN est déjà installé sur mon système) ?

10/26/2014 11:39 pm - Alexis Bienvenüe

- Target version set to 1.3.0

10/31/2014 10:14 am - Alexis Bienvenüe

- % Done changed from 0 to 100

- Status changed from Feedback to Closed

Files

insert.tex	1.4 kB	01/29/2014	Alexis Bienvenüe