Tests statistiques et régressions logistiques sous R, avec prise en compte des plans d'échantillonnage complexes

par Joseph LARMARANGE version du 29 mars 2007

Ce cours a été développé pour une formation niveau M2 et Doctorat des étudiants du laboratoire PopInter de l'université Paris 5 René Descartes.

Table des Matières

Note :

Une connaissance minimum de R est requise. La lecture du document **R pour les débutants** d'Emmanuel Paradis est donc fortement conseillée. Ce dernier est téléchargeable sur http://cran.r-project.org/other-docs.html.

1. Installation

1.1 Installation de R

Il vous faut d'abord télécharger la dernière version de l'installateur pour Windows sur CRAN (Comprehensive R Archive Network) à cette adresse : http://cran.r-project.org/.

Cliquez sur *Windows (95 or later)*, puis sur *base*, et télécharger le fichier d'installation *R-2.4.1-win32.exe*.

Lors de l'installation :

- Choisissez le dossier de destination (par défaut R sera installé dans *C:\Program Files\R*.
- Composants à installer : choisissez *Installation utilisateur complète*.
- Dans les options de démarrage, nous vous conseillons de choisir D*émarrage personnalisé*.
- Mode d'affichage : sélectionnez le mode *SDI*.
- Style d'aide : si vous ne savez pas quoi choisir, sélectionnez le mode *CHM*.
- Accès internet : si vous avez un doute, sélectionnez *standard*.
- Choisissez où vous souhaitez faire apparaître des raccourcis.

Pour information, il est possible de faire fonctionner R sur une clé USB. Pour cela, copier le répertoire *c:\Program Files\R* sur votre clé USB. Pour démarrer R, cliquez sur *Clé:\R\R-2.4.\bin\Rgui.exe*.

1.2 Installation du package survey

Ce package étant disponible sur CRAN, il peut être installé facilement dans R :

- Lancer R.
- Cliquez sur *Packages > Installer le(s) package(s)*.
- Sélectionnez un site miroir proche de vous.
- Sélectionnez le packages *survey*.

-

1.3 Installation du package TestsFaciles

Il reste à installer TestsFaciles à partir du fichier *zip* disponible sur le site de Joseph Larmarange (http://joseph.larmarange.net/Tests-statistiques-avec-prise-en.html) :

- Télécharger le fichier *zip* et enregistrer le sur votre disque dur.
- Dans R, cliquez sur *Packages > Installer le(s) package(s) des fichiers zip…*
- Sélectionner le fichier adéquat.

2. Remarques générales sur l'utilisation de R

R est un langage, orienté objet, de programmation statistique basé sur le langage S. R est sensible à la casse des caractères. Ainsi, l'objet AbcD sera différent de ABCD ou encore de abcd. Nous vous déconseillons fortement d'utiliser des caractères accentués dans le nom des objets que vous manipulez avec R.

> De nombreuses ressources sont disponibles sur internet pour vous initier à R. Outre une recherche avec Google, nous vous recommandons de consulter cette page http://cran.r-project.org/other-docs.html et en particulier le document **R pour les débutants** d'Emmanuel Paradis.

2.1 Pourquoi utiliser R ?

R est un langage de programmation statistique. Il offre un très grand choix de techniques statistiques et permet une manipulation poussée des données.

Bien que R ne dispose pas encore d'une interface utilisateur développée et intuitive et que la passage à un langage de programmation puisse rebuter certain, il y a tout intérêt à s'initier à ce logiciel du fait de sa puissance de calcul, des possibilités qu'il offre et de sa capacité à réaliser des graphiques entièrement paramétrables.

C'est par ailleurs un projet pérenne et en développement car ce logiciel est utilisé par un grand nombre de personnes à travers le monde (dont bon nombre d'universitaires) et plus d'une centaine de packages additionnels sont d'ores-et-déjà disponibles sur le net.

Enfin et surtout, il s'agit d'un logiciel libre, gratuit et disponible sur toutes les plateformes les plus courantes (Windows, MacOS et Linux).

Pour la problématique qui nous intéresse ici, à savoir la réalisation de tests statistiques et de régressions logistiques avec prise en compte du plan d'échantillonnage, d'autres solutions existantes. Les logiciels commerciaux SAS, Stata ou SPSS disposent de modules complémentaires

permettant de spécifier le plan d'échantillonnage d'une base de données. Cependant, ces modules sont rarement inclus dans la distribution par défaut et nécessite donc l'achat d'une licence spécifique et souvent très coûteuse. R offre ici une alternative efficace, gratuite et performante.

2.2 Charger un package

Lorsqu'un package est installé, les fichiers du package sont copiés dans le répertoire d'installation de R. Cependant, il n'est pas directement utilisable. En effet, lorsque que vous démarrez R, seules les fonctionnalités de base de R sont chargées en mémoire. Pour pouvoir utiliser les fonctions d'un package particulier, il faut au préalable le charger en mémoire, et ce à chaque fois que vous démarrez R.

Pour utilisez un package, vous devez le charger en mémoire. Deux solutions possibles :

- Utiliser le menu *Packages* > *Charger le package…*
- Utiliser la commande library ou la commande require.

Ainsi, pour charger tests-faciles, il suffit de taper :

```
> library(TestsFaciles)
Le chargement a nécessité le package : TestsFaciles
Le chargement a nécessité le package : survey
[1] TRUE
```

Le package survey étant nécessaire pour faire fonctionner TestsFaciles, il est automatiquement chargé quand on charge TestsFaciles.

2.3 Changer le répertoire de travail

Lorsque l'on importe ou que l'on exporte des données, à moins de spécifier le chemin complet d'un fichier, R va rechercher le fichier dans le répertoire de travail en cours. Pour modifier le répertoire de travail actuel, il suffit de cliquer dans le menu sur *Fichier* > *Changer le répertoire courant…*, puis de spécifier le nouveau répertoire de travail et de valider.

2.4 Obtenir de l'aide sur une fonction

Pour obtenir de l'aide sur une fonction dans R, il suffit de taper ? suivi du nom de la fonction ou du package, ou bien la fonction help. Ainsi, pour obtenir une aide générale sur TestsFaciles ou sur la fonction int.conf, il suffit de taper :

```
> ?TestsFaciles
> help('TestsFaciles')
> ?int.conf
```

Chaque fonction documentée fournit des exemples d'utilisation. Il est possible d'exécuter les exemples d'une fonction à l'aide de example(). La fonction mean() de R permet de calculer une moyenne. Pour voir des exemples d'utilisation de cette fonction il suffit de taper :

```
> example(mean)

mean> x <- c(0:10, 50)

mean> xm <- mean(x)

mean> c(xm, mean(x, trim = 0.1))
[1] 8.75 5.50

mean> mean(USArrests, trim = 0.2)
  Murder  Assault UrbanPop     Rape
    7.42   167.60    66.20    20.16
```

Pour rechercher un texte dans la documentation de R et de ses packages, il suffit d'avoir recours à la fonction help.search :

```
> help.search('proportion') # Pour chercher les fonctions travaillant sur des
proportions
```

NB : On notera que des commentaires peuvent être mis dans du code R à l'aide du caractère #.

2.5 Lister les objets en mémoire, sauver son travail et le récupérer

La liste des objets (textes, nombres, tableaux de données, fonctions personnelles, etc…) chargés en mémoire peut être facilement obtenue avec la fonction ls() (disponible également dans le menu *Misc* > *Lister les objets*).

```
> data(api)
> ls()
[1] "apiclus1" "apiclus2" "apipop"   "apistrat"
```

Un objet peut être supprimé à l'aide de la fonction rm() :

```
> rm('apipop')
> ls()
[1] "apiclus1" "apiclus2" "apistrat"
```

La fonction save() permet de sauver des objets dans un fichier de tye *.Rdata*. Pour cela, il suffit de passer à save() l'ensemble des objets que l'on souhaite sauvegarder, et d'ajouter le paramètre file qui spécifie le nom du fichier de sauvegarde. *Pensez à ajouter l'extension !!* La fonction load() permet quant à elle de charger un fichier de données.

```
> save(apiclus1,apiclus2,file='apiclus.Rdata')
> load('apiclus.Rdata')
```

Un fichier de données peut également être chargé à partir du menu à l'aide de la commande *Fichier > Charger l'environnement de travail*.

La commande *Fichier > Sauver l'environnement de travail* permet de sauvegarder l'ensemble des objets en mémoire dans un même fichier. On obtient le même résultat avec la fonction save.image().

À la fermeture de R, le logiciel nous invite à sauvegarder une image de la session. Si l'on acquiesce, l'ensemble des objets en mémoire seront sauvegardés dans un fichier nommé *.Rdata* situé dans le répertoire de travail courant.

2.6 Quelques fonctions de base

Voici une liste, non exhaustive, de quelques fonctions de base particulièrement utiles. Pour plus de détails, voir l'aide de chaque fonction. Les fonctions en italiques dépendent du package survey ou du package tests-faciles.

Nom	Description
as.character	Pour passer un objet en mode texte.
as.factor	Pour passer un objet en mode facteurs.
as.numeric	Pour passer un objet en mode numérique.
c	Permet de créer un vecteur.
compare	Compare des moyennes ou des proportions deux à deux.
edit	Édite un objet.
example	Exécute les exemples fournis dans la documentation d'une fonction.
function	Pour écrire ses propres fonctions.
getwd	Affiche le répertoire de travail courant.
help	Fournit de l'aide sur une fonction.
help.search	Effectue une recherche dans l'aide.
importer.spss	Importer un fichier de données SPSS.
int.conf	Calcule l'intervalle de confiance d'une moyenne ou d'une proportion.
levels	Affiche les étiquettes de valeur d'un objet de type facteurs.
library	Charge un package en mémoire.
list	Pour créer des listes.
load	Charge un fichier de données.
ls	Liste l'ensemble des objets en mémoire.
order	Pour trier des données.
read.dbf	Lire un fichier *dbf* (package *foreign*).
read.table	Importer un fichier texte tabulé.
require	Charge un package en mémoire.
rm	Supprime un objet.

save	Sauve un ou plusieurs objets dans un fichier de données.
save.image	Sauve l'ensemble des objets en mémoire.
seq	Permet de générer une liste de nombres.
setwd	Définit le répertoire de travail.
str	Détaille la structure d'un objet.
subset.survey.design	Extraire un sous-échantillon d'un tableau de données.
summary	Fournit un résumé détaillé du contenu d'un objet.
svychisq	Réalise un test du Chi² sur un tableau croisé.
svydesign	Défini le plan d'échantillonnage d'un tableau de données.
svyglm	Calcule une régression logistique.
svyglm	Calcule une régression logistique.
svymean	Calcule une moyenne ou une proportion.
svyquantile	Calcule des quantiles (dont la médiane)
svytable	Calcule un tableau de contingence, éventuellement croisé.
svyvar	Calcule la variance.
t	Transpose un tableau de données.
write.dbf	Exporter au format *dbf* (package *foreign)*.
write.table	Exporte des données sous la forme d'un fichier texte.

Pour des manipulations avancées des tableaux de données, nous vous conseillons la lecture de l'aide de l'opérateur [:

```
> help('[.data.frame')
```

3. Préparation et importation des données

3.1 Remarques générales

Dans les points suivants (3.2 et 3.3), nous supposerons que la manipulation des données s'effectuera sur un autre logiciel, r servant simplement à la réalisation des tests. Dans ce cas, on n'exportera de la base principale que les variables qui seront nécessaire à l'analyse sous R.

Nous ne détaillerons pas la manipulation de données déjà existantes sous R en raison de la diversité des situations existantes. Cependant, de nombreuses ressources sont disponibles en ligne sur les différentes manières de manipuler des objets sous R.

Pour la suite des analyses, il importera que les données aient la forme suivante :

- Elles seront stockées dans un tableau de données (data.frame) avec une ligne par individu statistique.
- Les variables pour lesquelles des moyennes seront calculées devront être de type numérique.

- Les variables pour lesquelles des proportions seront calculées devront être de type facteur. Si des modalités doivent être réunies en une seule, il faudra recoder la variable de manière à avoir les bons facteurs.
- Les variables déterminant des sous-groupes (pour des comparaisons par exemple) devront être également de type facteurs.
- Pour les régressions logistiques binaires, les variables analysées devront être codées de la forme : 1 si l'évènement a eu lieu, 0 sinon.
- Les valeurs manquantes devront être correctement renseignées à l'aide de la valeur NA.

3.2 Exportation et importation depuis SPSS

Il importe de correctement préparer ses données avant l'exportation. Les variables quantitatives ne doivent pas comporter d'étiquettes de valeurs (colonne *Valeurs* de l'onglet *Affichage des variables*). À l'inverse, les étiquettes de valeurs d'une variable qualitative doivent être renseignées. Si des recodifications sont nécessaires, il est préférable de les réaliser avant l'export.

Les valeurs manquantes doivent être correctement renseignées dans la colonne *Manquant*.

On n'exportera que les variables nécessaires à l'analyse ainsi que, le cas échéant, les variables nécessaires pour spécifier le plan d'échantillonnage (identifiant des clusters et des strates, pondération des individus).

Si l'analyse doit porter sur un sous-échantillon, on préparera une variable de tri permettant de spécifier quels individus appartiennent au sous-échantillon, mais on exportera l'ensemble de l'échantillon. Nous verrons plus loin comment extraire un sous-échantillon (partie 4.4).

Pour réaliser l'exportation, utilisez la commande *Fichier* > *Enregistrer sous…*. Le bouton *Variables* permet de spécifier les variables à inclure dans l'export. Enregistrez votre fichier au format SPSS (*.sav*) dans votre répertoire de travail R.

Dans R, utilisez la fonction importer.spss() pour importer votre fichier *.sav* dans un tableau de données. Par exemple, pour importer le fichier *export.sav* et le placer dans un tableau de données nommé *donnees* :

```
> donnees <- importer.spss('export.sav')
```

Utilisez la fonction str() pour voir la structuration de l'objet *donnees*, summary() pour avoir un résumé des différentes variables et edit() pour éditer les données.

```
> str(donnees)
> summary(donnees)
> edit(donnees)
```

3.3 Exportation et importation depuis Excel ou OpenOffice Calc

Préparez vos données dans une feuille de calcul avec une variable par colonne et un individu statistique par ligne. La première ligne contiendra le nom de chaque variable.

Supprimez les variables inutiles (pensez au préalable à effectuer une sauvegarde de votre base complète). Pour cela, sélectionnez les colonnes entières et utilisez la fonction *supprimer* du menu *Édition*[1].

Les variables numériques seront codées sous forme de chiffres tandis que les variables à modalité seront préférentiellement codée avec du texte. En effet, lors de l'import ultérieur avec R, si une colonne d'un fichier texte contient que des nombres, il considérera cette variable comme étant numérique. Si elle contient des caractères, il la traitera comme étant de type facteur. Si les modalités sont codées avec un nombre, nous verrons plus loin qu'il est possible de la convertir facilement du type numérique au type facteur.

Les valeurs manquantes devront impérativement correspondre à des cases vides.

Exporter vos données à l'aide de la commande *Enregistrer sous…* du menu *Fichier*. Choisissez le format *dBase (.dbf)*[2].

Sous R, pour pouvoir utiliser la fonction read.dbf() qui permet d'importer un fichier *.dbf*, il faut d'abord charger en mémoire le package foreign à l'aide de la commande library().

```
> library(foreign)
> donnees <- read.dbf('export.dbf')
```

[1] Ceci est préférable dans la mesure où une suppression rapide avec la touche supression efface seulement le contenu des colonnes en questions mais Excel ou Calc continuent de considérer qu'il s'agit d'une colonne définie et l'incluse donc dans l'export.

[2] Il est possible de faire un export au format *txt* ou *csv*. Cependant, en raison de la variété des paramètres (séparateur de champ, séparateur des décimales, etc.), il nous semble plus facile de passer par un export *dbf*.

4. Spécifier le plan d'échantillonnage

4.1 Différents types d'échantillons aléatoires

Le texte ci-dessous est repris de
http://www.er.uqam.ca/nobel/r30574/PSY1300/C6P3.html.

Définition : un échantillon est dit aléatoire lorsque la probabilité de sélection de chaque élément de la population est connue et non nulle

Avantage : permet de juger objectivement de la valeur des estimations

Types : aléatoire simple, stratifié et par grappes

4.1.1 Échantillonnage aléatoire simple

Définition : l'échantillonnage aléatoire simple est une méthode pour laquelle tous les échantillons possibles (de même taille) ont la même probabilité d'être choisis et tous les éléments de la population ont une chance égale de faire partie de l'échantillon.

Exemple : dans une classe de 20 personnes, on désire choisir un échantillon aléatoire simple de 5 individus. Par conséquent, chaque personne doit avoir une probabilité de 5/20 = 1/4 de se retrouver dans l'échantillon

4.1.2 Échantillonnage stratifié

Définition : l'échantillonnage stratifié est une méthode qui consiste d'abord à subdiviser la population en groupes homogènes (strates) pour ensuite extraire un échantillon aléatoire de chaque strate. Cette méthode suppose la connaissance de la structure de la population. Pour estimer les paramètres, les résultats doivent être pondérés par l'importance relative de chaque strate dans la population.

Exemple : Pour estimer le revenu annuel moyen des étudiants/es à l'Uqam, on prend un échantillon aléatoire de 10 individus dans chaque programme.

4.1.3 Échantillonnage par grappes

Définition : l'échantillonnage par grappes est une méthode qui consiste à choisir un échantillon aléatoire d'unités qui sont elles-mêmes des sous-ensembles de la population («grappes») Cette méthode suppose que les unités de chaque grappe sont représentatives. Elle possède l'avantage d'être souvent plus économique.

Exemple : Dans l'exemple du revenu des étudiants de l'Uqam, on choisit également un échantillon aléatoire de 30 programmes d'études différents

4.2 Impact sur les tests statistiques

La majorité des techniques statistiques de base enseignées aux étudiants et implémentées dans les logiciels de statistiques correspondent à une situation d'échantillonnage aléatoire simple. D'ailleurs, pour les personnes intéressées par l'utilisation de ces techniques sous R, nous leur recommandons la lecture du cours de *Biostatistiques avec R* de Nicolas jay disponible en ligne à cette adresse : http://www.spieao.uhp-nancy.fr/~jay/files/r/introduction-au-systeme-r.pdf.

À partir du moment où un échantillon présente un plan d'échantillonnage complexe (présence de strates et ou de clusters), ces formules ne sont plus utilisables car, non seulement les observations doivent être pondérées, mais le calcul des variances est plus complexe.

Le poids à attribuer à chaque observation correspond à l'inverse de la probabilité de cette observation d'être incluse dans l'échantillon. Ce qui est avant tout primordial c'est le poids relatif accordé à chaque individu. Cependant, le plus souvent, on multipliera l'ensemble des poids d'un échantillon par une même valeur afin que le total des poids corresponde au nombre total d'observations.

Le package survey permet justement de pouvoir tenir compte du plan d'échantillonnage afin de calculer correctement les différents indicateurs. Mais il faut au préalable lui spécifier correctement le plan d'échantillonnage des données. La simple connaissance des poids n'est pas suffisante. Il faut également préciser quelles sont les strates et quels sont les clusters.

4.3 Spécifier le plan d'échantillonnage avec svydesign()

La fonction svydesign() permet de spécifier le plan d'échantillonnage d'un échantillon et renvoie un objet R comprenant à la fois les données et l'ensemble des informations nécessaires concernant le plan d'échantillonnage.

Il est possible de spécifier des plans de sondage particulièrement complexes avec cette fonction. Pour cela nous vous renvoyons à la document de cette fonction (accessible en tapant help('svydesign')).

L'agument *data* permet de spécifier le tableau de données contenant les observations.
L'argument *ids* est obligatoire et spécifie sous la forme d'une formule les identifiants des différents niveaux d'un tirage en grappe. S'il s'agit d'un échantillon aléatoire simple, on entrera ids=~1. Autre

situation : supposons une étude portant sur la population française. Dans un premier temps, on a tiré au sort un certain nombre de départements français. Dans un second temps, on tire au sort dans chaque département des communes. Dans chaque commune sélectionnée, on tire au sort des quartiers. Enfin, on interroge de manière exhaustive toutes les personnes habitant les quartiers enquêtés. Notre fichier de données devra donc comporter pour chaque observation les variables *id_departement, id_commune* et *id_quartier*. On écrira alors pour l'argument *ids* la valeur suivante : ids=~id_departement+id_commune+id_quartier.

Si l'échantillon est stratifié, on spécifiera les strates à l'aide de l'argument *strata* en spécifiant la variable contenant l'identifiant des strates. Par exemple : strata=~id_strate.

Il faut encore spécifier les probabilités de tirage de chaque cluster ou encore la pondération des individus. Si l'on dispose de la probabilité de chaque observation d'être sélectionnée, on utilisera l'argument *probs*. Si par contre, on connaît la pondération de chaque observation (qui doit être proportionnelle à l'inverse de cette probabilité), on utilisera l'argument *weights*.

Si l'échantillon est stratifié, qu'au sein de chaque strate les individus ont été tirés au sort de manière aléatoire et que l'on connaît la taille de chaque strate, il est possible de ne pas avoir à spécifier la probabilité de tirage ou la pondération de chaque observation. Il est préférable de fournir une variable contenant la taille de chaque strate à l'argument *fpc*. De plus, dans ce cas là, une petite correction sera appliquée au modèle pour prendre en compte la taille finie de chaque strate.

Quelques exemples :

```
# Échantillonnage aléatoire simple
> plan <- svydesign(ids=~1, data=donnees)

# Échantillonnage stratifié à un seul niveau (la taille de chaque strate est connue)
> plan <- svydesign(ids=~1, data=donnees, fpc=~taille)

# Échantillonnage en grappes avec tirages à quatre degrés (departement, commune,
# quartier, individus). La probabilité de tirage de chaque niveau de cluster est connue.
> plan <- svydesign (ids=~id_departement+id_commune+id_quartier, data=donnees,
                     Probs=~proba_departement+proba_commune+proba_quartier)

# Échantillonnage stratifié avec tirage à deux degrés (clusters et individus).
# Le poids statistiques de chaque observation est connue.
> plan <- svydesign(ids=~id_cluster, data=donnees, strata=~id_strate, weights=~poids)
```

4.4 Extraire un sous-échantillon

Si l'on souhaite travailler sur un sous-échantillon tout en gardant les informations d'échantillonnage, il suffit d'utiliser la fonction subset() en lui spécifiant les données et une condition. Si par exemple, on souhaite récupérer le sous-échantillon des individus âgés de moins de 15 ans et allant à l'école (variable *age* de type numérique et *ecole* de type facteur) :

```
> sous <- subset(plan, age < 15 & ecole=='oui')
```

5. Les intervalles de confiance

5.1 Principes d'un intervalle de confiance

On cherche à connaître une grandeur X au sein d'une population. Or, le plus souvent, on ne peut enquêter toute la population. On réalisera donc un échantillon de la population et l'on mesurera dans cet échantillon une valeur expérimentale de X que l'on notera x_e.

X nous restera dans tous les cas inconnu (à moins de faire une enquête exhaustive). Cependant, on peut raisonnablement penser que x_e constitue une bonne approximation de X. Néanmoins, du fait de l'échantillonnage, il y a de fortes chances que x_e diffère légèrement de X. On va alors chercher à trouver un intervalle où, connaissant x_e il y a de forte chance que X s'y trouve. L'intervalle de confiance à 95% de X, connaissant x_e, c'est l'intervalle tel que la probabilité que X appartienne à cet intervalle soit de 95 sur 100.

Cet intervalle est usuellement centré autour de x_e. Plus le niveau de confiance est élevé, plus l'intervalle de confiance est large.

5.2 Moyennes et proportions

La méthode proposée ci-après est la même quelque soit le plan d'échantillonnage. Pour calculer l'intervalle de confiance d'une moyenne ou d'une proportion, on aura recours à la fonction int.conf(). Pour des détails sur cette fonction, voire l'aide en ligne : ?int.conf.

Le premier argument spécifie les variables pour lesquelles on souhaite calculer un intervalle de confiance, le second le fichier de données (après avoir spécifié le plan de sondage à l'aide de svydesign()) et le troisième, optionnel, une variable spécifiant des groupes si l'on souhaite séparer les résultats.

L'argument *confiance* permet de choisir le niveau de confiance désiré (95% par défaut)

Pour afficher plus facilement les résultats, on peut utiliser la fonction t() qui permet de transposer un tableau (les lignes deviennent colonnes et inversement).

La fonction int.conf() renvoie à la fois la valeur de x_e et celles de l'intervalle de confiance

Quelques exemples :

```
# Age moyen à 95%
> int.conf(~age, plan)

# Proportion de femmes à 95%
> int.conf(~sexe, plan)

# Age moyen et proportion de femmes
> int.conf(~sexe+age, plan)

# Age moyen et proportion de femmes selon la region
> int.conf(~sexe+age, plan, ~region)
```

```
# Age moyen et proportion de femmes selon la region avec une confiance de 90%.
> int.conf(~sexe+age, plan, ~region, confiance=0.9)

# Age moyen et proportion de femmes selon la region avec une confiance de 90%.
# avec affichage plus pratique des résultats
> t(int.conf(~sexe+age, plan, ~region, confiance=0.9))
```

5.3 Quantiles

Pour la médiane (quantile à 0.5 ou 50%) et les autres quantiles, il suffit d'utiliser directement la fonction svyquantile(). Il faut lui spécifier les arguments suivants (pour plus de détails voir l'aide de cette fonction) :

- *x* : liste des variables pour lesquelles on veut calculer les quantiles. Variables numériques uniquement.
- *design* : plan d'échantillonnage.
- *quantiles* : liste des quantiles désirés (en valeur décimales, pas de pourcentages). Untilisez la fonction c() pour faire une liste.
- *ci* : il faut préciser *ci=TRUE* pour que la fonction renvoie l'intervalle de confiance.
- *alpha* : correspond au complément à 1 de la confiance. Pour un intervalle à 95%, alpha vaut 1-0.95=0.05, pour un intervalle à 90%, alpha vaut 0.1 et pour un intervalle à 99%, alpha vaut 0.01.

Quelques exemples :

```
# Médiane de l'âge avec une confiance de 95%
> svyquantile(~age, plan, 0.5, ci=TRUE)

# Médiane et quartiles de l'âge avec une confiance de 95%
> svyquantile(~age, plan, c(0.25,0.5,0.75), ci=TRUE)

# Médiane et quartiles de l'âge avec une confiance de 90%
> svyquantile(~age, plan, c(0.25,0.5,0.75), ci=TRUE, alpha=0.1)

# Médiane et quartiles de l'âge avec une confiance de 99%
> svyquantile(~age, plan, c(0.25,0.5,0.75), ci=TRUE, alpha=0.01)

# Médiane et quartiles de l'âge et du nombre d'enfants avec une confiance de 95%
> svyquantile(~age+descandance, plan, c(0.25,0.5,0.75), ci=TRUE)
```

6. Tests de comparaison

6.1 Principes des tests de comparaison

Le but d'un test statistique c'est de tester une hypothèse statistique (notée H_0) versus une autre hypothèse (notée H_1). En fonction des données dont on dispose, on va regarder la probabilité que, si H_0 est vrai, ce qu'on a observé se réalise. Si cette probabilité est faible, on considérera que H_0 n'est pas vérifiée, on la rejettera et on choisira H_1. Si cette probabilité est importante, on conclura que l'on ne peut pas rejeter H_0.

Dans le cas d'un test de comparaison on cherche à savoir si une moyenne ou une proportion, observée dans deux groupes, sont identiques ou différentes. L'hypothèse nulle présuppose donc que les deux moyennes (ou proportions) sont identiques dans la population et que les écarts observés dans l'échantillon ne dépendent que d'effets aléatoires liés à l'échantillonnage.

On va donc calculer la probabilité d'observer, si les deux moyennes ou proportions étaient identiques dans la population, l'écart que l'on a effectivement observé dans l'échantillon. Cette probabilité est notée p.

Si cette probabilité est inférieure à un certain seuil, usuellement 5%, on va considérer que la probabilité que la différence observée soit simplement due à l'échantillonnage est trop faible et donc que la différence observée provient du fait que les deux moyennes (ou proportions) ne sont pas identiques dans la population. On dira alors que la différence est statistiquement significative au seuil de 5%.

À l'inverse, si cette probabilité est supérieure à 5%, on conclura que la différence observée peut n'être due qu'à des effets d'échantillons. On ne montre pas que les deux moyennes (ou proportions) sont identiques dans la population. Simplement on ne peut exclure cette éventualité.

En résumé :

- si p<seuil, on rejette H_0 : la différence est statistiquement significative ;
- si p>seuil, on garde H_0 : la différence peut n'être due qu'à des aléas de l'échantillonnage.

Soyez vigilant vis-à-vis de l'expression *statistiquement significatif*. C'est une expression technique indiquant simplement que l'on a réuni suffisamment de données pour permettre d'établir l'existence d'une différence effective. Elle ne signifie pas que la différence est importante ni même qu'elle est une signification humaine ou scientifique. L'expression *statistiquement significative au seuil de 5%* signifie littéralement *statistiquement discernable avec une marge d'erreur de 5%*.

6.2 Tests unilatéraux et tests bilatéraux

Dans le cadre d'un test unilatéral, on teste si les deux grandeurs sont différentes. L'hypothèse nulle présuppose donc leur égalité et l'hypothèse H_1 qu'elles sont différentes.

Dans le cas d'un test bilatéral, on suppose dans l'hypothèse nulle que la moyenne (ou la proportion) du groupe A est inférieure ou égale à celle du groupe B. On teste donc si la moyenne (ou proportion) du groupe A est supérieure à celle du groupe B.

6.3 Comparer deux moyennes ou deux proportions

Il suffit d'utiliser la fonction compare() en lui précisant trois arguments :

- En premier, la liste des variables que l'on souhaite tester (de type numérique pour des moyennes et de type facteur pour des proportions).
- En second, le plan d'échantillonnage.
- En troisième, une ou plusieurs variables de type facteur précisant les groupes à comparer.

compare() renvoie la valeur de la moyenne ou de la proportion dans chaque groupe, calcule leur différence et renvoie la valeur de *p* à la fois pour un test unilatéral et pour un test bilatéral.

Si le troisième argument défini plusieurs groupes, les variables du premier argument sont testées pour chaque combinaison de deux groupes.

Quelques exemples :

```
# Comparaison de l'âge moyen selon le milieu de résidence
> compare(~age, plan, ~residence)

# Comparaison de l'âge moyen selon le milieu de résidence et la région
> compare(~age, plan, ~residence+region)

# Comparaison de l'âge moyen et de la descendance finale selon le milieu de résidence
> compare(~age+descendance, plan, ~residence)

# Comparaison de la proportion de femmes selon le type emploi
> compare(~sexe, plan, ~type_emploi)

# Comparaison de la proportion de femmes selon chaque type d'emploi et niveau de revenus
> compare(~sexe, plan, ~type_emploi+niveau_revenu)
```

6.4 Test du Chi²

Il s'agit de tester deux variables qualitatives (de type facteur dans R) pour savoir si elles sont indépendantes ou non. Si deux variables qualitatives sont indépendantes, et si l'on fait un tableau croisé de l'une par l'autre, alors les pourcentages en colonnes seraient identiques d'une colonne à une autre et les pourcentages en ligne seraient identiques d'une ligne à autre.

Le test du Chi² va donc calculer le tableau croisé des deux variables comparées et regarder si la répartition des observations diffère *significativement d'un point de vue statistique* de la répartition que l'on aurait si les deux variables étaient indépendantes.

Si le test est positif, cela signifie qu'il y a une corrélation entre les deux variables. Au moins une ligne ou au moins une colonne diffère sensiblement des autres. Il s'agit d'un test sur l'ensemble du tableau. Il ne dit rien sur des comparaisons de deux colonnes deux à deux.

L'observation d'une corrélation statistique entre deux variables signifie juste que les variations observées ne peuvent être imputées uniquement à des aléas de l'échantillonnage. Cela ne veut surtout pas dire qu'il y aurait une relation de cause à effet entre elles, ni même que cette variation ait un sens scientifique.

Si le tableau croisé comporte deux colonnes et deux lignes, alors le test du Chi² peut être utilisé comme un test de comparaison de deux proportions.

svytable() permet de calculer le tableau croisé et svychisq() de faire le test. Leurs arguments sont les suivants :

- En premier, la liste des deux variables à croiser.
- En second, le plan d'échantillonnage.
- Optionnel pour svytable, un nombre spécifiant le total du tableau. Par défaut, la fonction renvoie les effectifs de chaque case. Si on lui transmet en troisième argument la valeur 100, on aura alors des pourcentages du total.
- Pour svychisq uniquement, on passera *'Chisq'* en troisième argument. Cet argument permet de spécifier le type de test à effectuer (cette fonction en propose plusieurs). Le plus courant est celui-ci.

Quelques exemples :

```
# Niveau de salaire selon le sexe
> svytable(~sexe+salaire, plan)
> svychisq(~sexe+salaire, plan, 'Chisq')

# Niveau de scolarité selon le village
> svytable(~village+ecole, plan)
> svychisq(~village+ecole, plan, 'Chisq')
```

7. Régressions logistiques

Un cours complet et accessible au non technicien sur les régressions logistiques est disponible en ligne à cette adresse : http://www.tesser-pro.org/stat/Cours_regression_logistique.pdf. Ce cours de Patrick Taffé est particulièrement bien écrit pour des non statisticiens. Sa lecture est donc fortement conseillée. Nous ne détaillerons donc pas ici les subtilités de l'interprétation d'une régression logistique.

Un autre cours plus technique de Paul Marie Bernard : http://www.uquebec.ca/reglog/.

Sur Wikipédia : http://fr.wikipedia.org/wiki/R%C3%A9gression_logistique.

Un cours sur les régressions logistiques avec R : http://pbil.univ-lyon1.fr/R/archives/br5.pdf[3]. Ce cours ne tient pas compte des plans de sondage complexe et utilise la fonction glm(). Cependant, pour l'adapter aux plans complexes, il suffit le plus souvent de remplacer glm() par svyglm().

[3] Plus généralement, de nombreuses ressources sur les statistiques et R sont disponibles en français sur http://pbil.univ-lyon1.fr/R/enseignement.html.

7.1 Principes d'une régression logistique

La **régression logistique** est une technique statistique qui a pour objectif, à partir d'un fichier d'observations, de produire un modèle permettant de prédire les valeurs prises par une variable catégorielle, le plus souvent binaire, à partir d'une série de variables explicatives continues et/ou binaires.

Dans le cas présent, nous ne regarderons que le cas d'une régression logistique pour une variable binaire (de type oui/non) en ayant recours au modèle *logit* (modèle le plus courant). Nous noterons *X* la variable étudié. Elle prend deux valeurs : 0 si l'évènement ne s'est pas réalisé, 1 s'il s'est réalisé. On modélisera la probabilité que *X* soit égal à 1 connaissant les valeurs d'autres variables, dites variables explicatives et notées Y_1 Y_2 Y_3 … Y_n.[4]

On cherchera les coefficients α, β_1, β_2, β_3 … β_n tels que

$$\text{logit}(P[X=1]) = \alpha + \beta_1 y_1 + \beta_2 y_2 + \ldots + \beta_n y_n$$

$$\text{où } \text{logit}(p) = \log\left(\frac{p}{1-p}\right).$$

La fonction logit() existe dans le package boot. Il faut donc penser à charger ce package en mémoire pour pouvoir utiliser cette fonction (library(boot)). La fonction inverse existe également et se nomme inv.logit().

7.2 Préparer les données

La variable que l'on cherche à modéliser, *X*, doit être de type facteur avec deux modalités. La première modalité ou modalité de référence doit correspondre au cas où le phénomène étudié n'a pas eu lieu[5].

La fonction relevel() est très pratique pour préciser quelle est le facteur de référence. Supposons que l'on souhaite que la modalité de référence de la variable *malade* d'un tableau de données *donnees* soit *non* on entrera sous R :

```
> donnees$malade <- relevel(donnees$malade, ref='non')
```

Si l'on souhaite modifier la même variable mais dans cette fois-ci dans le plan d'échantillonnage, on ne peut faire directement plan$malade. Dans un objet créé avec svydesign(), les données sont accessibles via plan$variabales :

[4] La notation est ici simplifiée par rapport aux notations habituelles. Pour une notation plus rigoureuse, voir les cours de Patrick Taffé ou de Paul Marie Bernard.

[5] Cette variable peut également être de type numérique avec les valeurs 0 et 1, et plus généralement toute valeur comprise entre 0 et 1 correspondant à une probabilité.

```
> plan$variables$malade <- relevel(plan$variables$malade, ref='non').
```

Concernant les variables explicatives du modèle, y_1 y_2..., elles peuvent être soit quantitatives (type numérique) soit qualitatives (type facteurs). Pour les variables qualitatives, on utilisera relevel() de la même manière afin de définir la modalité de référence.

7.3 Calculer une régression avec svyglm()

On aura recours à la fonction svyglm() à laquelle on passera trois arguments :

- Une formule spécifiant le modèle à ajuster. Elle sera de la forme x~y1+y2+y3, avec la variable à modéliser à gauche du ~ et les variables explicatives à droite séparées par +[6].
- Le plan d'échantillonnage contenant les données.
- L'argument family=binomial qui précise qu'il s'agit d'une régression logistique binaire de type *logit*.

Un exemple où l'on souhaite calculer la probabilité d'être atteint par une maladie en fonction de l'âge, du sexe et du milieu de résidence :

```
> reg <- svyglm(malade~age+sexe+residence, plan, family=binomial)
```

7.4 Résultats

Pour afficher les résultats, utilisez la fonction summary() :

```
> summary(reg)
```

La valeur des différents coefficients sont affichés sous la colonne *Estimates*. Par ailleurs, cette fonction fournit le résultat d'un test vérifiant si chaque coefficient diffère de 0 (absence d'effet significatif). La valeur du *p* est affichée dans la colonne *Pr(>|t|)*. Des étoiles et des points indiquent par ailleurs les résultats du test aux seuils usuels.

Le plus souvent, on n'utilise pas les coefficients du modèle mais les *odds-ratios*. Ces derniers correspondent à l'exponentiel des coefficients. Ils peuvent êtres obtenus facilement en faisant :

```
> exp(reg$coefficients)
```

[6] Il est possible d'effectuer des modèles plus complexes. Référez vous à la documentation de glm pour cela.

Pour vérifier si un modèle est efficace, on peut comparer les valeurs observées avec les valeurs prédites par le modèle. predict() permet de récupérer les probabilités prédites.

```
> plan$variables$prob.pred<- predict(reg, newdata=plan$variables, type='response')
```

On considère que si cette probabilité est supérieure à 0.5, alors l'évènement a eu lieu, sinon il n'a pas eu lieu.

```
> plan$variables$malade.pred <- 'non'
> plan$variables$malade.pred[plan$variables$prob.pred>0.5] <- 'oui'
```

Il reste alors à afficher le tableau croisé entre variable observée et variable prédite :

```
> svytable(~malade.pred+malade, plan)
```