## Thème : analyse exploratoire des données — TP 1

### Préambule

Le logiciel de statistique qui sera utilisé dans les travaux pratiques est **R**, logiciel libre distribué sous les termes de la *GNU, General Public Licence*, au site web

`http://www.r-project.org`.

Ce logiciel est disponible pour les systèmes d'exploitation Unix, Linux, Windows et Mac OS X. Des exécutables précompilés de la version actuelle R-4.2.2 ("Innocent and Trusting") sont disponibles sur l'un des miroirs du CRAN (*Comprehensive R Archive Network*). Les instructions à suivre pour les installer s'y trouvent.

Pour faciliter votre apprentissage du logiciel, Emmanuel Paradis et Julien Barnier ont écrit de bonnes documentations françaises pour **R**, "**R** pour les débutants" et "Introduction à **R**", qui se trouvent dans la page Moodle du cours à l'adresse

`https://cyberlearn.hes-so.ch/course/view.php?id=22686`.

Le logiciel de statistique **R** fonctionne principalement par commandes. L'attente de commandes, par défaut le symbole >, apparaît au démarrage du logiciel et indique que **R** est prêt à exécuter les commandes. Sous Windows, en utilisant l'interface R-GUI de **R**, certaines d'entre elles (par exemple accès à l'aide et ouverture de fichiers) peuvent être exécutées par les menus.

Le logiciel **R**, créé vers 1994 par Ross Ihaka et Robert Gentleman de l'Université d'Auckland, est davantage qu'un simple logiciel de statistique. Il s'agit non seulement d'un outil d'analyse statistique et graphique mais aussi d'un langage reposant sur le langage **S** créé par AT&T Bell Laboratories. John M. Chambers, l'un des créateurs de **S**, a reçu en 1998 le *Software System Award* de la prestigieuse ACM ("Association for Computing Machinery").


*Tiré du "New York Times" du 6 janvier 2009.*

Les possibilités offertes par **R** sont vastes et permettent à l'utilisateur d'effectuer des analyses de données très pointues. Le logiciel est reconnu pour sa flexibilité. En effet, les résultats d'une analyse sont stockés dans un "objet" ; il est alors possible de n'afficher que la partie des résultats qui intéresse l'utilisateur. Cette facilité n'est pas offerte par la plupart des logiciels classiques. Notons que toutes les actions de **R** sont effectuées sur les objets présents dans la mémoire vive de l'ordinateur. Aucun fichier temporaire n'est utilisé. Pour mieux comprendre le fonctionnement de **R**, il est fortement recommandé de lire le chapitre 2 de l'aide d'Emmanuel Paradis (2005).

Le logiciel de statistique **R** nécessite un apprentissage qui peut paraître pénible et difficile en raison du recours aux commandes plutôt qu'aux menus déroulants.


Rassurez-vous, après s'être rapidement familiarisé avec quelques notions et concepts de base, l'utilisateur pourra employer efficacement le logiciel dont le fonctionnement reste finalement très intuitif. De plus, les commandes vous offrent un horizon de possibilités bien plus large que celui des menus déroulants. Un aide-mémoire des principales commandes de **R** figure dans le fichier "aide_memoire.pdf" qui se trouve dans la page Moodle du cours.

Une aide en ligne existe directement dans **R**. Elle est très utile pour connaître l'utilisation des fonctions du logiciel. Plusieurs méthodes existent pour y accéder : la première en utilisant la commande `help()`, comme par exemple pour la fonction `mean()`, moyenne,

```
help("mean")
```

Une deuxième possibilité consiste à utiliser l'alias de la commande `help()`, un point d'interrogation, `?mean`, et finalement, une dernière variante revient à utiliser simplement le menu de l'interface R-GUI de **R**.

Un moteur de recherche pratique pour obtenir une aide supplémentaire et complète sur **R**, ses fonctions, ses librairies complémentaires et la programmation dans **R** est disponible à l'adresse

`http://www.rseek.org`.

L'utilisation de **R** peut aussi être facilitée en utilisant le **Quick–R**

`http://www.statmethods.net/`.

Pour les utilisateurs de Linux, Windows et Mac OS X, il existe un éditeur, RStudio[1] IDE (*Integrated Development Environment*), encore plus convivial que celui que vous propose **R** par défaut. Il vous permet d'écrire et conserver des scripts dans lesquels figure une suite de commandes qui seront exécutées successivement. Vous pourrez lancer les scripts directement depuis l'éditeur sans avoir besoin de procéder à un "copier–coller". Les scripts de commandes peuvent être archivés et accessibles à tout moment. Il est également possible d'afficher simultanément plusieurs fichiers contenant différents scripts et passer aisément de l'un à l'autre. La possibilité d'écrire des scripts, de les archiver, de les exécuter plusieurs fois en des temps différents est indéniablement un avantage par rapport à ce que vous proposent les logiciels à menus déroulants.

RStudio, outil qui doit impérativement être installé, est disponible à l'adresse

`https://posit.co/downloads/`.

Il est aussi possible d'écrire de manière simple ses propres fonctions. Sans entrer dans les détails, une fonction **R** est écrite dans un fichier sauvé au format ASCII avec extension .R. Comme les autres langages, **R** possède des structures de contrôle qui ne sont pas sans rappeler celles du langage C.

Lorsque vous terminez votre session **R**, n'oubliez pas d'en sauver une image. Elle vous permettra de conserver les objets et de récupérer les dernières commandes utilisées. Dans l'interface R-GUI de **R**, d'autres options très pratiques vous sont offertes comme par exemple charger et sauver l'environnement de travail (utile si vous travaillez sur plusieurs projets distincts), charger et sauver l'historique des commandes.

Un glossaire des notions statistiques avec définition et traduction dans plusieurs langues (français, allemand, anglais, italien et autres) se trouve à l'adresse

`http://isi.cbs.nl/glossary/index.htm`.

L'étudiant doit rédiger un rapport du travail pratique dans lesquels figurent les réponses aux questions posées ainsi que les graphiques tracés. Il sera aussi interrogé sur les travaux pratiques aux travaux écrits.

Les rapports peuvent être rédigés en LaTeX. Un excellent environnement pour écrire des documents en LaTeX est TeXstudio disponible à l'adresse

`http://texstudio.sourceforge.net`.

Un aide-mémoire LaTeX figure dans le fichier "aide-memoire.pdf" qui se trouve dans la page Moodle du cours.

L'extension **rmarkdown** permet de générer des documents de manière dynamique en mélangeant texte et résultats obtenus à l'aide du code **R**. Les documents créés peuvent être notamment au format HTML, PDF, Word. Ainsi, **rmarkdown** est un outil très pratique pour exporter, communiquer et diffuser des résultats d'analyses statistiques. La librairie sous-jacente et fort élégante pour compiler, ou plutôt "tricoter" (*knit* en anglais), un document **R Markdown** afin de visualiser le document généré est **knitr**. Elle crée entre autres un fichier .pdf dans lequel sont insérés les commandes, les sorties ainsi que les graphiques tracés. Vous avez entre vos mains un document confectionné à l'aide de **knitr**[2].

Deux modèles de rapport se trouvent dans la page Moodle du cours. Vous remarquerez que les extensions des fichiers ne sont pas les extensions habituelles .R : .Rnw pour LaTeX et .Rmd pour Microsoft® Word.

1. RStudio, la société qui développe l'interface, s'appelle Posit depuis novembre 2022.
2. Des tutoriaux, démonstrations et exemples se trouvent à l'adresse `http://yihui.name/knitr`.

L'objectif de ce travail pratique consiste à se familiariser avec les commandes de base du logiciel **R** et à utiliser les techniques d'analyse pour une variable statistique. En route !

**Exercice 1**

L'une des forces de **R** est qu'il est capable de traiter de grands jeux de données de manière très rapide. Cet avantage a évidemment son prix : la lecture des données peut paraître ennuyeuse, particulièrement lorsqu'elles se trouvent sur support informatique. Il existe cependant plusieurs possibilités pour lire dans **R** des données contenues dans un fichier. Si elles se présentent sous la forme d'une liste de valeurs telles que chacune d'elles figure sur une ligne ou si elles sont séparées par un espace, on peut utiliser la commande `scan()` qui renvoie un vecteur. Lorsque les données se présentent sous la forme d'une table, i.e. une ligne par observation et une colonne par variable, l'instruction à utiliser est `read.table()` si les données se trouvent dans un fichier texte (ASCII). Des variantes de cette fonction existent comme par exemple `read.csv2()` [3]. Dans cet exercice, nous allons enregistrer dans **R** les données qui seront utilisées dans le travail pratique. Vous pouvez organiser comme vous le souhaitez votre travail. Néanmoins, nous vous suggérons de créer deux répertoires : l'un contenant les données et l'autre votre travail (script, rapport, résultats). Pour être plus structuré, vous pouvez même ajouter un sous-répertoire à votre répertoire de travail pour y stocker vos graphiques et créer un projet pour encore mieux gérer votre travail pratique.

a) Les données que nous allons traiter dans ce travail pratique se trouvent dans la page Moodle du cours. Copiez-les dans votre répertoire de données.

b) Utiliser les commandes suivantes dans **R** pour charger les données :

```
cpus<-scan("cpus.txt")
examen<-read.table("examen.txt", header=T)
```

Les utilisateurs se chargeront d'adapter les chemins à leur répertoire de travail et à leur système d'exploitation. Les fichiers `cpus.txt` et `examen.txt` sont ainsi accessibles dans **R** sous les noms `cpus` et `examen` respectivement.

c) Pour voir le contenu de l'objet `cpus`, taper l'instruction

```
cpus
```

```
##   [1]  77  18  22 144  12 185  38  24  45  38  65 141 208  58  12 636 397  66 915  27  30
##  [22]  66  36  14  26  92   8  36 133  66  24  10  36 100  60  33  40  19  16 140  63  21
##  [43]  32  64  24 110  16  56  46 144
```

Il en est de même pour `examen`. Les objets `cpus` et `examen` sont de nature toute différente. En effet, le premier est un vecteur, le second un tableau de données, data.frame [4] en anglais.

d) Pour accéder à la 10ème composante du vecteur `cpus`, utiliser la commande

```
cpus[10]
```

```
## [1] 38
```

3. Pour de plus amples informations, voir "**R** pour les débutants", E. Paradis, 2005, pages 12–16.
4. Pour de plus amples informations, voir "**R** pour les débutants", E. Paradis, 2005, page 12.

e) Pour obtenir une partie du vecteur `cpus` comme par exemple les éléments du vecteur compris entre la 3ème et la 17ème composante, taper l'instruction

```
cpus[3:17]

##  [1]  22 144  12 185  38  24  45  38  65 141 208  58  12 636 397
```

f) Pour extraire du vecteur `cpus` ses éléments supérieurs à 185, utiliser la commande

```
cpus[cpus>185]

## [1] 208 636 397 915
```

g) Il est possible d'accéder directement aux composantes d'une table par le nom. Par exemple, si on veut afficher la composante `note` de l'objet `examen`, on peut utiliser la commande

```
examen$note

##  [1] 3.3 3.3 3.6 4.8  NA 2.6 3.9 4.4 4.4 4.8 5.2 3.8  NA 1.9 3.9 4.3 3.7 4.0 4.1 4.7 3.6
## [22] 2.7 3.5 4.8
```

h) On peut aussi accéder en profondeur aux composantes comme par exemple par la commande

```
examen$note[7]

## [1] 3.9
```

i) La méthode la plus simple pour créer un vecteur consiste à énumérer ses éléments à l'aide de la fonction `c()`

```
mesdonnees<-c(2.9, 3.4, 3.4, 3.7, 3.7, 2.8, 2.8, 2.5, 2.4)
mesdonnees

## [1] 2.9 3.4 3.4 3.7 3.7 2.8 2.8 2.5 2.4

couleurs<-c("bleu", "vert", "blanc", "noir")
couleurs

## [1] "bleu"  "vert"  "blanc" "noir"
```

j) On peut ôter des composantes d'un vecteur en indiquant entre crochets les indices précédés du signe négatif comme par exemple

```
mesdonnees[-c(3:5)]

## [1] 2.9 3.4 2.8 2.8 2.5 2.4
```

k) Finalement, le contenu de votre environnement de travail est affiché à l'aide de la commande

```
ls()
```

**Exercice 2**

La performance relative au processeur `IBM 370/158-3` de 50 processeurs d'ordinateurs a été relevée.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 77 | 18 | 22 | 144 | 12 | 185 | 38 | 24 | 45 | 38 |
| 65 | 141 | 208 | 58 | 12 | 636 | 397 | 66 | 915 | 27 |
| 30 | 66 | 36 | 14 | 26 | 92 | 8 | 36 | 133 | 66 |
| 24 | 10 | 36 | 100 | 60 | 33 | 40 | 19 | 16 | 140 |
| 63 | 21 | 32 | 64 | 24 | 110 | 16 | 56 | 46 | 144 |

L'objet `cpus` contient les valeurs observées.

a) Construire un diagramme branche-et-feuilles, un histogramme et une boîte à moustaches des données observées à l'aide des commandes ci-dessous.

```
stem(cpus)
par(mfrow=c(1,2), pty="s") # deux graphiques sur la même fenêtre, cadre carré
hist(cpus, xlab="performance relative", ylab="fréquence", main="",
     col="darkslategray4")
boxplot(cpus, xlab="performance relative", col="darkslategray4", horizontal=T)
rug(cpus)
par(mfrow=c(1,1))
```

Observer les résultats obtenus par chaque commande.

b) Commenter la distribution des valeurs observées : valeur(s) atypique(s), asymétrie.

c) Calculer la performance relative médiane, la performance relative moyenne et le(s) mode(s) des valeurs observées en complétant les commandes suivantes :

```
median(cpus)
...(cpus)
n.cpus<-table(...)
as.numeric(names(n.cpus)[n.cpus==max(n.cpus)])
```

Est-il plus approprié d'utiliser la médiane ou la moyenne ?

d) Que fait la commande suivante ?

```
summary(cpus)
```

e) En effectuant aucun calcul, décrire l'effet sur la moyenne et sur la médiane des trois interventions suivantes :

1. ajouter un processeur de performance relative 45 ;
2. soustraire 9 à chaque valeur observée ;
3. diviser chaque observation par 4.

f) Déterminer l'écart-type des performances relatives une fois avec les valeurs atypiques et une fois sans en utilisant les commandes ci-dessous.

```
sd(cpus)
sd(cpus[cpus<=185])
```

Que constate-t-on ? L'écart-type est-il un indicateur robuste ?

**Exercice 3**

Les étudiants suivant un cours de Probabilités et Statistique dans une école d'ingénierie ont passé l'examen de fin d'unité. Le cours était donné par le même professeur à 24 étudiants répartis en deux groupes, celui formé par les étudiants en emploi, EE, et celui constitué par les étudiants à plein temps, PT. Les résultats obtenus figurent dans la TABLE 1 et sont contenus dans l'objet `examen`.

| | mode de formation | note |
|---|---|---|
| 1 | PT | 3.3 |
| 2 | PT | 3.3 |
| 3 | PT | 3.6 |
| 4 | PT | 4.8 |
| 5 | PT | *NA* |
| 6 | PT | 2.6 |
| 7 | PT | 3.9 |
| 8 | PT | 4.4 |
| 9 | PT | 4.4 |
| 10 | PT | 4.8 |
| 11 | PT | 5.2 |
| 12 | PT | 3.8 |
| 13 | PT | *NA* |
| 14 | PT | 1.9 |
| 15 | PT | 3.9 |
| 16 | PT | 4.3 |
| 17 | PT | 3.7 |
| 18 | EE | 4.0 |
| 19 | EE | 4.1 |
| 20 | PT | 4.7 |
| 21 | EE | 3.6 |
| 22 | EE | 2.7 |
| 23 | EE | 3.5 |
| 24 | EE | 4.8 |

TABLE 1 – Notes obtenues par les étudiants à l'examen de fin d'unité du cours de Probabilités et Statistique.

On se demande si une différence significative existe entre les deux groupes à l'examen.

a) Tracer les boîtes à moustaches en parallèle en utilisant les commandes suivantes :

```
note.EE<-split(examen$note, examen$mode_de_formation)$EE
note.PT<-split(examen$note, examen$mode_de_formation)$PT
lblue<-"#528B8B"
par(pty="s")
boxplot(note~mode_de_formation, data=examen, ylim=c(1,6), xlab="mode de formation",
        varwidth=T, col=lblue, main="examen")
abline(h=4, lty=2)
rug(note.EE, side=2)
rug(note.PT, side=4)
```

En se basant sur ce graphique, existe-t-il une différence significative entre les deux groupes à l'examen de fin d'unité ?

b) Observe-t-on sur les boîtes à moustaches une différence entre les dispersions des deux groupes ?

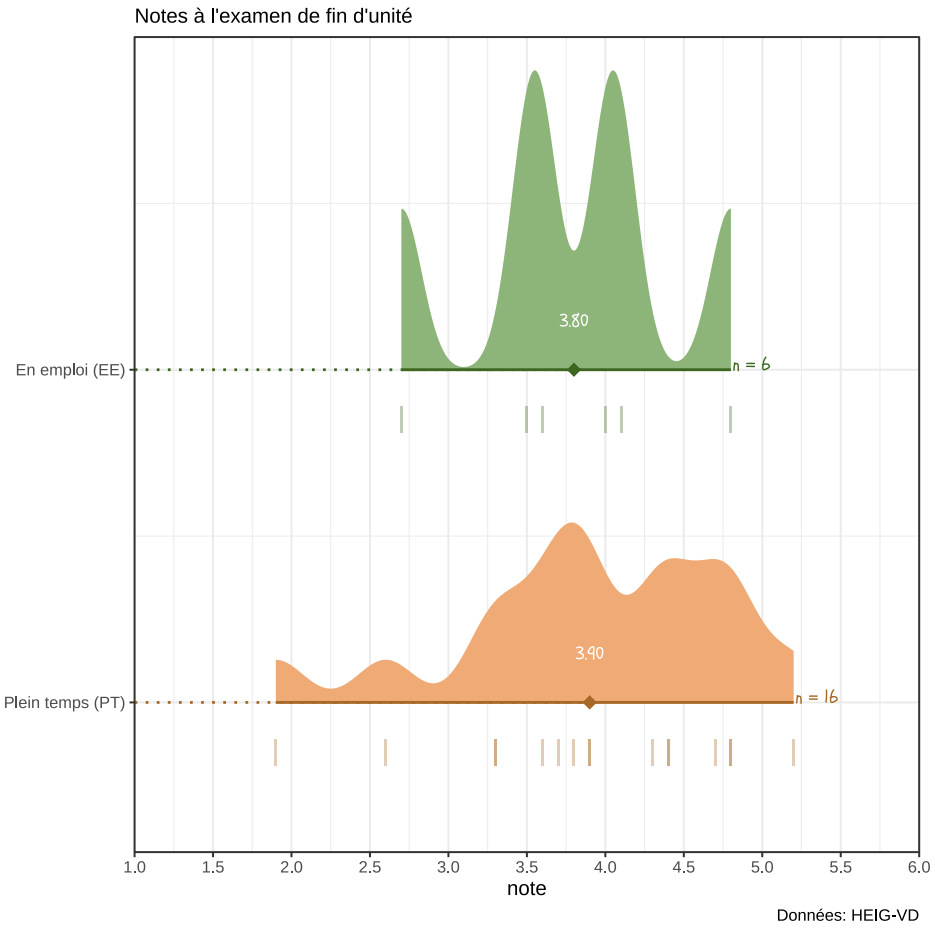
c) Calculer les écarts-types des deux groupes à l'aide de la commande `sd()`.

```
sd(note.EE, na.rm=TRUE)
sd(note.PT, na.rm=TRUE)
```

En se basant sur les écarts-types, existe-t-il une différence en dispersion entre les deux groupes à l'examen de fin d'unité ?

d) Que peut-on déduire en comparant les conclusions établies en b) et en c) ?

e) Un autre graphique pour étudier les éventuelles différences entre les deux groupes à l'examen de fin d'unité se trouve ci-dessous.




À votre avis, entre les boîtes à moustaches en parallèle et le graphique tracé ci-dessus, lequel est le plus approprié ?

**Exercice 4**

Une partie de la base de données du recensement américain [5] de 1994 a été extraite. Elle concerne 48'842 personnes adultes dont on s'intéresse notamment à l'influence que peut avoir le type de scolarité\formation acquise par l'individu sur le nombre d'heures de travail par semaine.

5. Par intérêt, un coup d'œil à la page `https://www.census.gov/`.

Par simplicité et pour préserver l'authenticité du système éducatif américain, le nom des variables n'est pas traduit en français.

a) Nous nous proposons de tracer les boîtes à moustaches en parallèle du temps consacré au travail par les individus recensés. Pour y parvenir, nous utiliserons la librairie **ggplot2** qu'il faudra d'abord installer puis activer dans votre session à l'aide des commandes

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

La librairie **ggplot2** explicite les liens conceptuels entre graphiques et analyses statistiques. Sa syntaxe est particulière mais ingénieuse. Elle se base sur un ensemble de composants indépendants qui peuvent être combinés de différentes manières [6].

Les données du recensement se trouvent dans la librairie **arules** de **R** qui est installée puis activée par les commandes

```
install.packages("arules")
library(arules)
```

Les observations sont lues dans le logiciel à l'aide de la commande

```
data("AdultUCI")
```

et les variables qui nous intéressent sont sélectionnées et stockées dans l'objet `dframe` par les commandes

```
dframe<-AdultUCI[, c("education", "hours-per-week")]
colnames(dframe)<-c("education", "hours_per_week") # à quoi sert cette commande ?
str(dframe)
```

Pour tracer les boîtes à moustaches en parallèle du temps hebdomadaire consacré au travail par les Américains recensés selon leur formation, il convient d'utiliser la commande

```
ggplot(dframe, aes(x=education, y=hours_per_week)) +
  geom_point(colour="lightblue", alpha=0.1, position="jitter") +
  geom_boxplot(outlier.size=0, alpha=0.2) + coord_flip()
```

Commenter le graphique obtenu.

b) Pour quel type de formation observe-t-on la plus grande dispersion du temps de travail ? Existe-t-il une différence entre les médianes des types de formation ? En donner brièvement la raison.

c) Pour chaque type de formation, on peut déterminer puis afficher à l'écran le temps maximal de travail hebdomadaire à l'aide des commandes

```
nx<-tapply(dframe$hours_per_week, dframe$education, max, na.rm=T)
nx
```

6. De plus amples informations se trouvent à l'adresse `https://ggplot2.tidyverse.org/`.

La formation pour laquelle un temps maximal a été observé se détermine par les commandes

```
max(nx)
names(nx)[nx==max(nx)]
```

Est-ce surprenant ?

d) En s'inspirant des commandes utilisées en c), déterminer la formation pour laquelle la distribution des temps de travail se caractérise par le plus petit écart-type.

e) Observe-t-on un résultat similaire en utilisant l'étendue interquartiles à l'aide de la fonction `IQR()` ?

Pour répondre à cette question, utilisez les commandes

```
nx<-aggregate(hours_per_week~education, dframe, IQR, na.rm=T)
min(nx[,2])
nx[,1][nx[,2]==min(nx[,2])]
```

**Exercice 5**

Dans cet exercice, nous vous présentons deux librairies de **R**. La première, **plotly**, permet de construire des boîtes à moustaches accompagnées par les statistiques élémentaires usuelles. La seconde, **magrittr**[7], fournit un mécanisme grâce auquel des commandes peuvent être efficacement enchaînées, principalement à l'aide de l'opérateur %>%.

Des informations sur ces librairies se trouvent respectivement aux adresses

`https://plot.ly/`

et

`https://cran.r-project.org/web/packages/magrittr/vignettes/magrittr.html`.

Avant de construire les boîtes à moustaches en parallèle des notes obtenues à l'examen de Probabilités et Statistique, il faut d'abord installer puis charger les deux librairies à l'aide des commandes

```
install.packages("plotly")
library(plotly)
install.packages("magrittr")
library(magrittr)
```

La librairie **plotly** crée des graphiques interactifs dont l'objectif principal consiste à les diffuser et à les projeter. Ainsi, il est possible que les commandes ci-dessous produisent une erreur dans vos rapports obtenus par compilation. En utilisant directement les commandes dans votre console, les graphiques apparaîtront par exemple dans le "Viewer" de RStudio ou dans votre navigateur préféré.

Les boîtes à moustaches sont tracées à l'aide des commandes enchaînées

```
plot_ly(examen, y=~note, x=~mode_de_formation, type="box") %>%
  layout(
    yaxis=list(range=c(1, 6)))
```

7. Cette librairie porte le nom de **magrittr** en hommage au peintre surréaliste belge René Magritte (1898–1967) qui peignit un tableau dans lequel figure une pipe avec la légende "Ceci n'est pas une pipe".


*Dessin d'Allison Horst.*