AGENIS-NEVERS Marc
Publication : 11/01/2017
Version : 1.1

Conception d'un prototype de borne de sondage en temps réel, utilisant R.

Contexte

Les bornes de sondages ont commencé à apparaître un peu partout dans l'espace public (gares, magasins, services publics) et remplacent avantageusement des procédures de sondage plus lourdes, tout en rendant possible le sondage d'une large proportion (voire la totalité) de la population concernée.

Les entreprises proposant de tels matériels en France sont nombreuses (Agoraopinion, Smilio, Xifab, Needanswer, Dymension, etc.) et le coût de location d'une borne peut varier entre 30 et 100 euros par mois selon les options, comme l'envoi de données via un réseau *Internet of Things* (Lora, SigFox), la mise à disposition d'un tableau de bord en ligne, etc.

Toutefois, aucune publication scientifique sur la validité de tels sondages n'a été répertoriée, bien que plusieurs brevets existent décrivant de telles bornes (US 4345315 A par exemple). L'absence de littérature scientifique sur ces dispositifs pose la question de la validité des résultats, quand bien même les vendeurs les garantissent très supérieures à des sondages classiques. Certains biais peuvent être imaginés comme l'âge (les jeunes plus enclins à répondre), le sexe, ou la possibilité de réponses multiples/malveillantes. La donnée générée par ces systèmes n'est quasiment pas traitée avant visualisation, seuls des filtres simples (plages horaires ou fréquence de clic trop élevée) sont mis en place.

La note décrit la conception du prototype d'une borne simple utilisant une pédale USB couplée au logiciel d'analyse de données R.

Prototype

Matériel

Le matériel nécessaire pour la conception du prototype:

- Une pédale USB 3 voies de modèle SCYTHE TRIPLE FOOT SWITCH II, dotée de son pilote et interface HIDKEYBOARD 2.31. Prix : 50€
- Un ordinateur portable doté de Windows 7 Entreprise, connecté à un écran LCD.

Les logiciels suivants sont également utilisés:

- RStudio version 0.99.903 couplé à R version 3.2.3 (2015-12-10) disponibles ici et ici.
- Le freeware AutoKeyboard en version 1.3 disponible ici.

La pédale USB est un accessoire ordinairement utilisé dans le jeu vidéo ou pour la réalisation de tâches très répétitives sur ordinateur (transcription audio, copier-coller en masse, etc.), le principe étant d'utiliser le pied pour remplacer le clic souris, une touche ou combinaison de touches au clavier. De ce fait, elle est très solide et résistante au choc, elle fera office ici de boutons customisables qu'on associera, via l'interface utilisateur, aux chiffres 1, 2 et 3.

Le logiciel d'analyse de données R servira à la collecte et au traitement puis affichage des résultats du clic, via un script détaillé plus loin.

Enfin, un logiciel tierce partie est nécessaire pour l'enregistrement : un simple appui sur le bouton écrit le chiffre 1, 2 ou 3 à l'écran mais doit être validé par la touche [ENTREE]. Sans accès au clavier, cette difficulté s'est avérée critique et plusieurs solutions ont été envisagées, comme dédier un des trois boutons à la validation du choix. La solution retenue permet en fait d'émuler l'appui sur la touche [ENTREE] de façon répétée (toutes les secondes).

Mise en place

La Figure 1 montre le dispositif installé sur une table à l'entrée du bureau. Une question simple a été formulée (« How do you feel about your work today ? ») et les boutons associés à des smileys symbolisant les choix "happy", "indifferent", et "unhappy". Une vidéo du fonctionnement est visible.

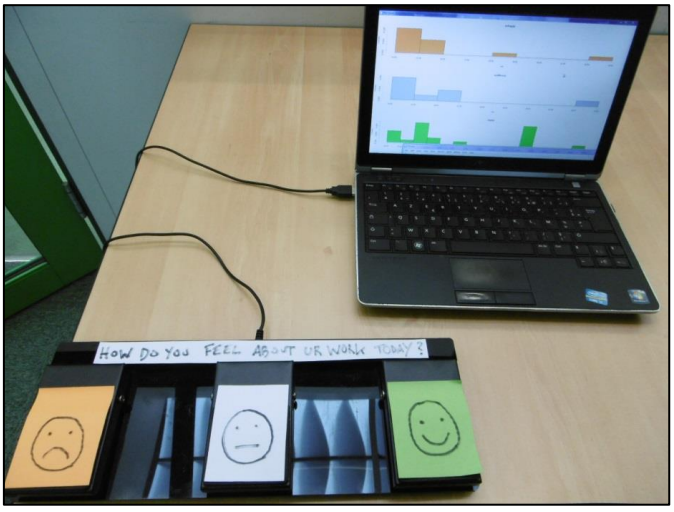

Figure 1: photo du dispositif en fonctionnement

Algorithme

L'algorithme de traitement de la donnée est publié en annexe et ci-dessous (Encadré 1) en pseudocode.

```
INITIALISER un tableau de résultats
LIRE la valeur entrée
        SI la valeur n'est pas vide (touche [ENTREE] appuyée sans clic bouton)
                ENREGISTRER le premier chiffre de la valeur entrée
                INCREMENTER le tableau de résultat avec la valeur et l'horodatage
                SI le tableau est suffisamment grand pour plotter
                        CHOISIR le pas de temps à afficher selon la profondeur d'historique
                        PLOTTER les histogrammes de chaque bouton
                END SI
        END SI
REPETER la boucle indéfiniment (la touche [ECHAP] permet de sortir)
```

Encadré 1: pseudocode du script de collecte de données

Ainsi on voit que lorsqu'aucun bouton n'est pressé le script n'enregistre aucune donnée. Afin d'éviter que plusieurs utilisateurs n'appuient dans l'intervalle entre deux émulations automatiques de la touche [ENTREE], et que du coup une valeur invalide soit enregistrée (11, 111,12…), un contrôle est réalisé et seul le premier chiffre du nombre entré est conservé. Cela évite également une partie des appuis malveillants.

Sorties graphiques

Le script R affiche dans une fenêtre séparée les histogrammes de l'historique des clics, en couleur, et met à jour le graphique à chaque fois que le bouton est cliqué de façon valide (Figure 2). Le pas de temps est ajusté selon la profondeur d'historique (par seconde, puis par minute, puis par heure, jour, semaine). Par ailleurs, les données peuvent être récupérées à la fin de l'utilisation.

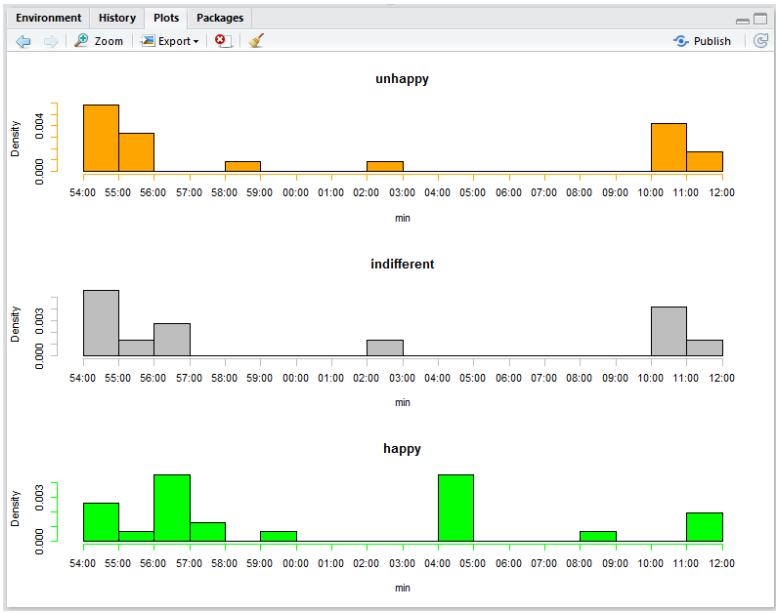

Figure 2: vue de la sortie graphique (histogrammes)

Dans la version actuelle du prototype, aucun traitement statistique n'est réalisé sur les données ; ce peut être l'objet d'une future évolution, ou lorsque suffisamment de données seront accumulées.

Annexe

Le script R du prototype est publié ci-dessous.

```
# details
#########

# This function works in combination with
# - an USB pedal switch tuned to press 1, 1/2 or 1/2/3
# - the freeware Auto Keyboard set to the key [ENTER]
# design: marc.agenis-nevers@veolia.com

# packages
##########
# No package required

# functions
###########

Initialize = function() {
  timer.out <<- data.frame('datetime'=Sys.time(), 'key'=NA)
}

ReadKeyboard = function(script) {
  press <- readline(prompt="# ")
  if (press != "") {
    timer.out <<- rbind(timer.out, list(Sys.time(),
as.numeric(substring(press, 1, 1))))
    if (nrow(timer.out) > 10) {
    PlotSatisfaction(timer.out)
    }
  }
  press <- ReadKeyboard()
  return(n)
}

PlotSatisfaction = function(df) {
  freq <- attributes(diff.POSIXt(range(timer.out$datetime)))$units
  freq <- substr(freq, 1, nchar(freq)-1)
par(mfrow=c(3, 1))
  hist(timer.out[timer.out$key==1, ]$datetime, breaks=freq, main="unhappy",
col="orange", xlab=freq)
  hist(timer.out[timer.out$key==2, ]$datetime, breaks=freq,
main="indifferent", col="grey", xlab=freq)
  hist(timer.out[timer.out$key==3, ]$datetime, breaks=freq, main="happy",
col="green", xlab=freq)
}

# RUN
#####

cat("\n\n\n\n\n\n\n\n\n\n\n\n")
Initialize()
ReadKeyboard()
```