ARTRAGIS

CC BY

Geste de savoir

Récupérer les données à partir d'un graphique, avec R, en 5 minutes

25 janvier 2021

Table des matières

Ce billet est une adaptation de la vidéo postée par lebiostatisticien , accompagnée de son article de blog. La publication est faite avec l'autorisation de l'auteur.

Élément externe (Video) —
Consultez cet élément à l'adresse `https://www.youtube.com/embed/AG9EZ0aV-zo?feature=oembed`.
La vidéo originale

Imaginez, vous trouvez un graph dans un vieux papier, une vieille publi, et vous voudriez récupérer les données de ce graph pour pouvoir les retravailler sous R. Il y a plein d'outils qui permettent de faire ça, mais il y a surtout un package R, qui est assez fabuleux: digitize .

1. Cas d'étude : L'effet Dunning-Kruger

1.1. Ce qu'on veut faire

Partons d'un exemple réel : le papier sur l'effet Dunning-Kruger.

Vous connaissez sans doute cette effet au travers de cette figure erronée[1]:

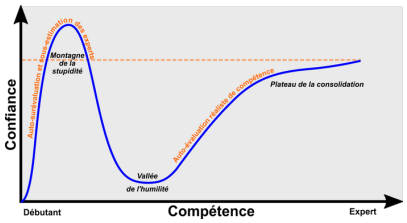

FIGURE 1.1. – L'effet Dunning Kruger tel que fréquemment vulgarisé : d'abord la montagne de la stupidité (les gens se surévaluent), puis la vallée de l'humilité (ils se sous-évaluent) puis un plateau de consolidation (ils s'évaluent correctement)

En gros l'effet Dunning Kruger explique qu'on a du mal a évaluer sa propre compétence et qu'on a tendance à se surévaluer quand on n'est pas compétent dans un domaine.

Sur l'article publié en 1999, nous aimerions refaire un graphique *propre* par rapport à l'original.

.

Pour cela nous devons extraire les données du graphique pour ensuite le redessiner.

Pour cela nous allons utiliser un package dans R qui s'appelle **digitize** et qui est vraiment chouette.

1.2. Préparation

La première étape consiste extraire le graphique et sauvegarder l'image en local. On peut utiliser l'outil de capture d'écran de l'ordinateur. Sur Windows, deux possibilités:

- dans le menu démarrer taper "Capture" et il vous proposera d'utiliser l'outil
- appuyer simultanément sur Windows+Maj+S, votre fenêtre se grisera et vous pourrez prendre la capture qui vous plaît.

Vous pouvez enregistrer votre capture dans le dossier de votre choix.

2. Utiliser digitize pour récupérer les données du graphe

(Pour éditer le code R et explorer les données, le logiciel rstudio est utilisé.)

Dans le dossier qui contient l'image, créer un fichier de script `digitize_script.R`. Vous pouver ensuite l'ouvrir avec RStudio.

Pour vous faciliter la vie, vous allez annoncer à RStudio que votre répertoire de travail sera celui du script. Pour cela, cliquez sur `Session` puis `Set Working Directory` et enfin `To Sourcefile Location`

1. Cet article vous expliquera en quoi la figure est erronée.

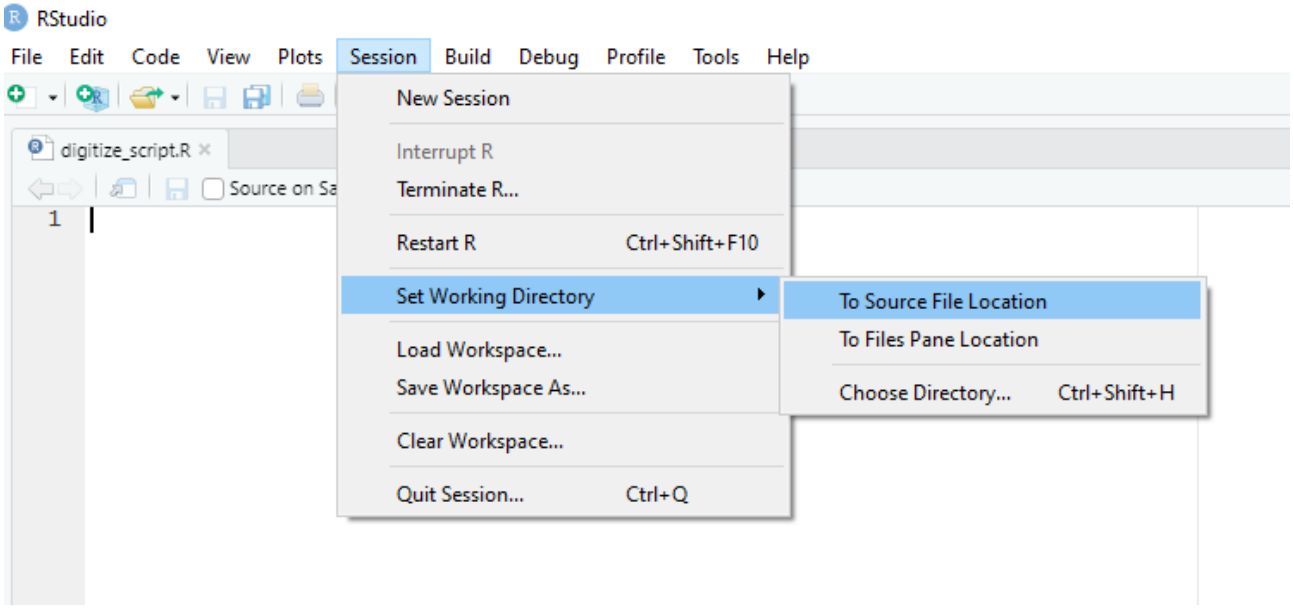

FIGURE 2.2. – Configuration du répertoire de travail

Pour pouvoir nous amuser avec les données, nous allons maintenant charger deux bibliothèques: `digitize` et `tidyverse`.

Si c'est la première fois que vous utilisez ces packages, vous pouvez les installer en sélectionnant l'onglet "Packages" puis en cliquant sur install.

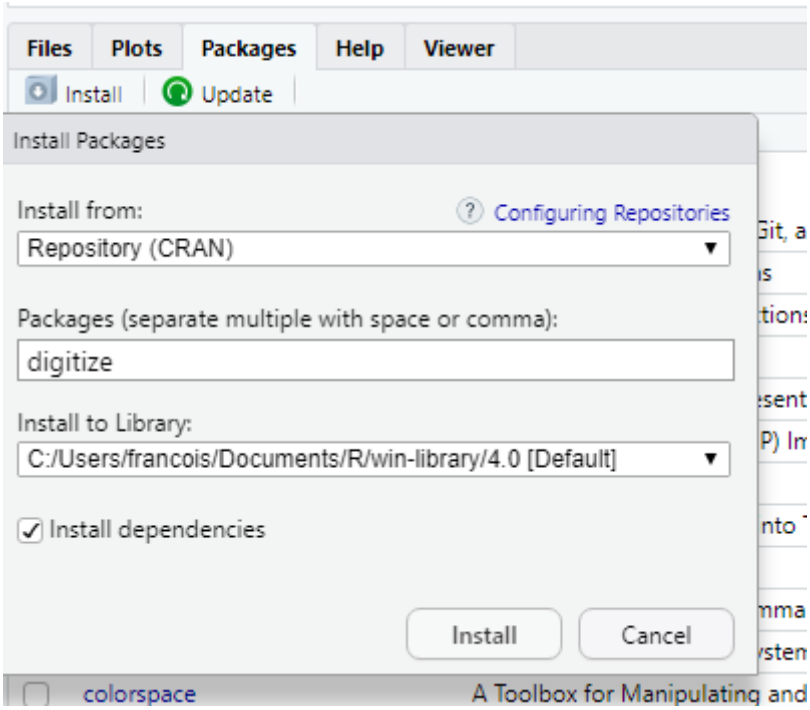

FIGURE 2.3. – Installation du package digitize

Une fois l'aide chargé, nous allons regarder l'aide de digitize dans la console.

```
Usage
digitize(image_filename, ..., x1, x2, y1, y2)

Arguments

image_filename: the image file you wish to digitze

...: pass parameters col or type to change data calibration points

x1: (optional) left-most x-axis point
x2: (optional) right-most axis point
y1: (optional) the lower y-axis point
y2: (optional) the upper y-axis point
```

Listing 1 – Aide de la fonction digitize

Il nous suffit donc de passer le nom de l'image en tant que premier argument. Comme nous connaissons le nom du fichier, nous devons le passer entre guillemets. Nous stockerons le résultat dans une variable.

```r
library(digitize)
library(tidyverse)

digitize::digitize("capture.png")
```

Listing 2 – appel de `digitize`, ne pas hésiter à utiliser la touche tabulation pour autocompléter les noms de fichiers et aller plus vite

Dans la console, digitize nous invite à calibrer notre graphique, en face l'onglet "plot" a affiché l'image que nous voulons extraire.

La calibration consiste à définir 2 points:

- le point avec l'abscisse et l'ordonnée les plus petites
- le point avec l'abscisse et l'ordonnée les plus grandes

La sélection se fait dans cet ordre: on clique sur l'abscisse la plus petite, puis la plus grande, ensuite on clique sur l'ordonnée la plus petite et la plus grande. Pour se faciliter la tâche on peut s'aider soit des points de la courbe soit des graduations des axes.

Dans notre exemple, les abscisses sont des classes de données (les quartiles), chaque point est placé au milieu de la classe, milieu qui n'est pas gradué. Nous allons donc utiliser les points du graphique. On vise le milieu de ceux-ci.

A l'opposé, l'axe des ordonnées est gradué d'une manière exploitable (et chose importante, il commence à 0 !), nous allons donc utiliser les graduations.

Une fois nos quatre clics réalisés, dans la console, RStudio nous demande les valeurs des points que nous avons sélectionnés.

Dans notre cas, dans l'ordre, on aura 25, 100, 0, 100.

Maintenant que nous avons terminé notre calibration, nous pouvons sélectionner les données. Pour rester organisé, nous allons cliquer un par un sur les points en respectant l'ordre des séries.

Une fois qu'on a cliqué sur les huit points, on peut appuyer sur la touche ESC. Dans la console

on obtient donc un tableau qui ressemble à ça:

.	x	y
1	25.18610	57.65306
2	50.12407	61.22449
3	74.87593	70.91837
4	100.18610	76.53061
5	25.00000	12.50000
6	50.31017	37.24490
7	74.68983	62.75510
8	100.00000	88.26531

TABLE 2.2. – Résultat de la fonction digitize

Comme on le voit, il a produit deux colonnes avec les 8 valeurs qu'on lui a passées.
Maintenant il va falloir travailler les données.

3. Travailler sur les données

3.1. Recréer les séries

On va commencer par distinguer les deux séries. Pour cela, nous allons ajouter une troisième colonne qui permettra de donner un *label* à nos points.
Pour ce faire, nous utilisons du R standard: `mydata$group <- c(rep("Perceived", 4), rep("Actual", 4))`.
Désormais, `mydata` a une colonne "group" qui reprend les séries du graphique original:

.	x	y	group
1	25.18610	57.65306	Perceived
2	50.12407	61.22449	Perceived
3	74.87593	70.91837	Perceived
4	100.18610	76.53061	Perceived
5	25.00000	12.50000	Actual
6	50.31017	37.24490	Actual
7	74.68983	62.75510	Actual
8	100.00000	88.26531	Actual

3.2. Nettoyer les abscisses

On remarque que les abscisses sont sensées tomber pile sur les classes mais qu'à cause de l'imprécision au clic, nous avons quelques ratés. On peut donc commencer par faire un arrondi sur cette colonne.

```
mydata$x <- round(mydata$x)
```

3.3. Afficher les données

Nous allons maintenant pouvoir refaire le même graphique que l'original, en plus propre et avec des couleurs.
Pour cela, nous allons envoyer nos données dans `ggplot` en demandant à ce que la couleur des lignes et des points soit pilotée par la colonne `group`.

```
library(digitize)
library(tidyverse)

mydata <- digitize::digitize("capture.png")

mydata$group <- c(rep("Perceived", 4), rep("Actual", 4))
mydata$x <- round(mydata$x)

mydata %>%
  ggplot() +
  geom_point(aes(x, y, color = group)) +
  geom_line(aes(x, y, color = group))
```

Listing 3 – Le code complet pour produire un graphique identique à l'original, en plus propre
Le résultat s'affiche dans l'onglet plot:

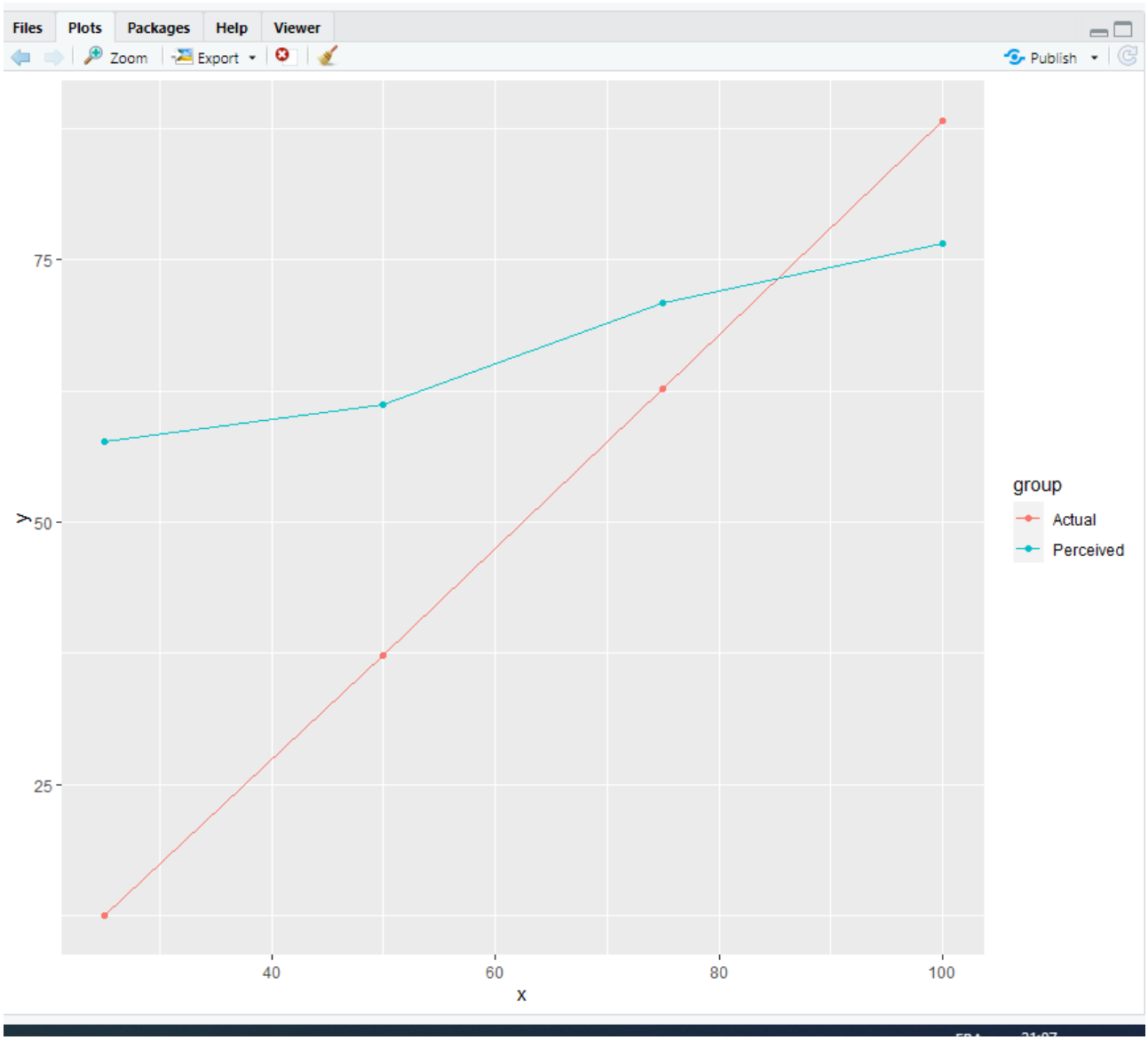

FIGURE 3.4. – Le graphique final

3.4. Extraire plus des données

Plus intéressant que refaire le graphe original, maintenant qu'on a toutes les données, on peut produire d'autres visualisations. Par exemple nous pourrions générer le vrai graphe qui évalue la "compétence perçue" par rapport à la "compétence réelle".
Il faut donc changer un peu la forme des données pour obtenir une colonne "Perçu" et une colonne "Réel" plutôt que deux séries.
Ici nous allons utiliser la commande `pivot_wider` à qui nous allons spécifier d'aller chercher les nouvelles colonnes dans `group` et les valeurs qui seront utilisées dans `y`.
Une fois cela fait on pourra afficher la courbe final avec en abscisse la compétence réelle, en ordonnées la compétence perçue. Il faudra prendre garde de bien faire commencer l'axe des ordonnées à 0.

```r
data_wide <- mydata %>%
  pivot_wider(names_from = group, values_from = y)

data_wide %>%
  ggplot() +
  geom_point(aes(x = Actual, y = Perceived )) +
  geom_line(aes(x = Actual, y = Perceived )) +
  ylim(0, 100)
```

Listing 4 – Le code pour tenter de retrouver la montagne de la stupidité

Le graphique final ne permet donc d'observer une surévaluation, mais pas de montagne.

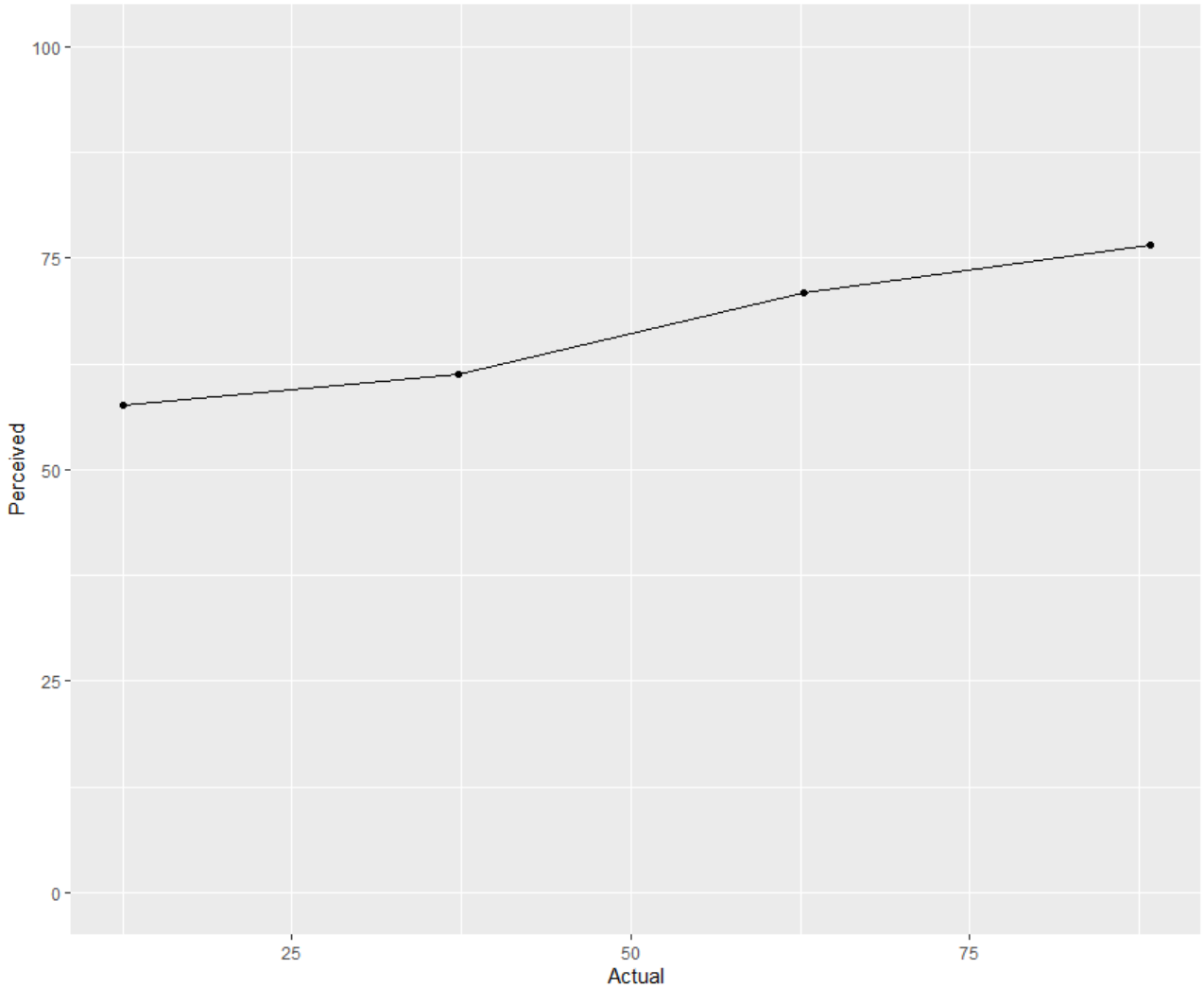

FIGURE 3.5. – Le résultat

Ainsi, en peu de temps nous avons réussi à extraire d'un graphique en basse qualité, les manipuler et générer un graphe qui nous intéressait plus.
Cet astuce peut s'avérer très utile et j'espère qu'elle vous aura plu.

L'auteur de la vidéo originale prévoit de publier une autre vidéo où il réalisera le même exercice mais en extrayant les données d'un tableau. Si le sujet vous intéresse, abonnez vous à sa chaîne Le biostatisticien .