Price Sensitivity Meter con R

Karel L.

14 de enero de 2016

Objetivos.

1- Presentar la implementación en R de una técnica de investigación de mercados para realizar test de precios que tiene por nombre Price Sensitivity Meter (PSM).

2- Personalización gráfica.

¿Cuánto debería cobrar por este producto?

Contexto.

- El precio es un elemento crítico en el éxito de un producto o servicio. Ej: rentabilidad, capacidad, etc.
- Impacto psicológico del precio y posicionamiento del producto o servicio. Ej: interpretación de la relación calidad/precio.
- Buena parte del economía se desarrolla fuera de los modelos teóricos de competencia perfecta o competencia monopolística.
- Las decisiones sobre la fijación de precios están sujetas a gran incertidumbre, principalmente cuando no existe una referencia previa.
- La batalla por el excedente del consumidor.
- El valor percibido es difícil de cuantificar.

La investigación de mercados y los test de precios.

- Hace lo que puede: intención vs. evidencia. Ej: encuestas de intención de voto.
- Existen varias técnicas para el análisis de precios, … ninguna espectacular.
- Requiere una descripción clara, sencilla y neutra del producto o servicio. Evitar la sugestión.
- Barreras defensivas de los encuestados y profesionalización de los mismos.

Price Sensitivity Meter.

- Es una técnica de análisis de precios basada en parte en percepciones psicológicas.
- Fue presentada en el Congreso de ESOMAR de 1976 por el economista holandés Van Westendorp.
- Muy extendida, pero tampoco está exenta de críticas.
- Consta de dos fases, una de diseño y otra de análisis.

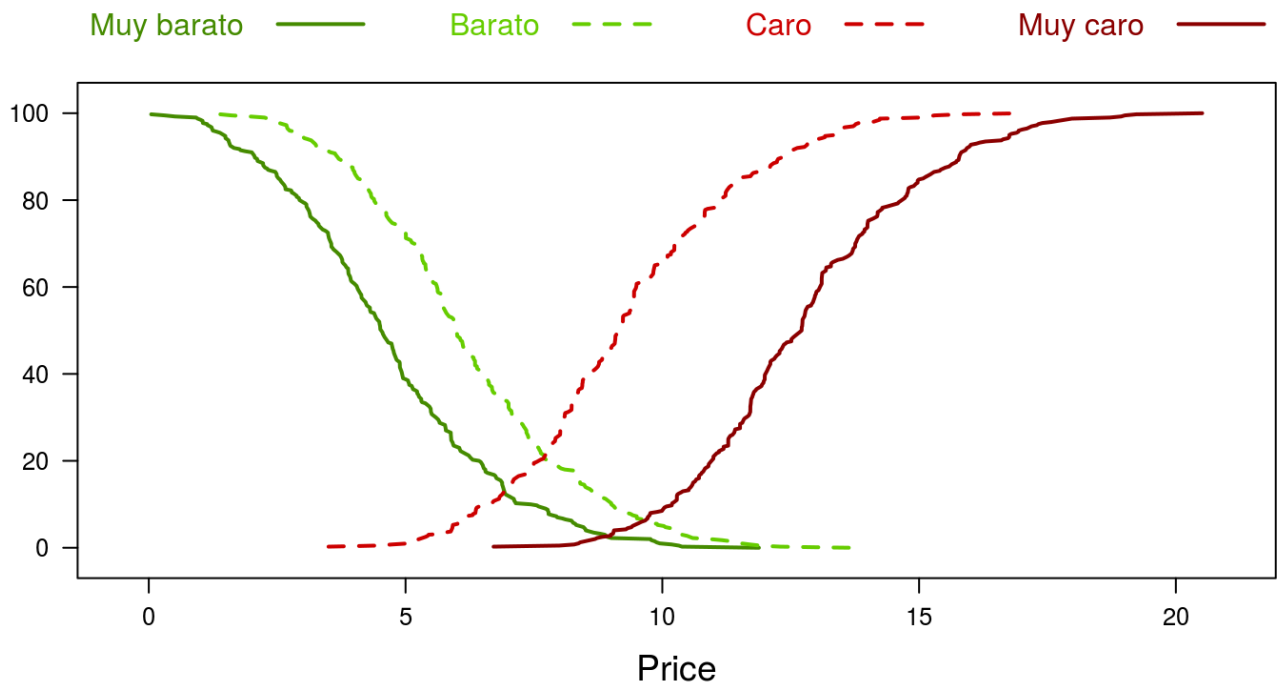
Fase de diseño y captura de la información.

Batería de 4 preguntas:

- ¿A qué precio dejarías de comprar el producto/servicio … por considerar que es tan barato que su calidad se resentiría?
- ¿A qué precio consideras que el producto/servicio … es barato?
- ¿A qué precio consideras que el producto/servicio … empieza a ser caro, pero aún así lo comprarías?
- ¿A qué precio dejarías de comprar el producto/servicio … por considerar que es demasiado caro?

Fase de análisis.

Con las 4 variables numéricas obtenidas se construyen 4 curvas de distribución empíricas acumuladas (ecdf) con la salvedad de que las relacionadas con el precio barato se construyen como ecdf inversas.



Implementación en R.

Hay un repositorio en github.

```
# Instalación
library(devtools)
devtools::install_github("kintero/psmr")

# Carga
library(psmr)
```

Implementación en R.

Hay función principal llamada psm() y que cuenta con los siguientes parámetros:

- **data**: data.frame con los datos.
- **toocheap**: nombre de la variable/columna con las respuestas del precio muy barato.
- **cheap**: nombre de la variable/columna con las respuestas del precio barato.
- **expensive**: nombre de la variable/columna con las respuestas del precio caro.
- **tooexpensive**: nombre de la variable/columna con las respuestas del precio muy caro.
- **by**: nombre de la variable/columna para segmentar, por defecto NULL.

El objetivo de último parámetro es que sirva para identificar diferencias en perfiles de cara a posibles políticas de discriminación de precios.

Implementación en R.

La función anterior devuelve un objeto de clase psm que viene a ser lista con una estructura específica donde se almacenan los resultados.

Implementación en R.

Función summary().

Genera una tabla con los principales resultados.

Función plot().

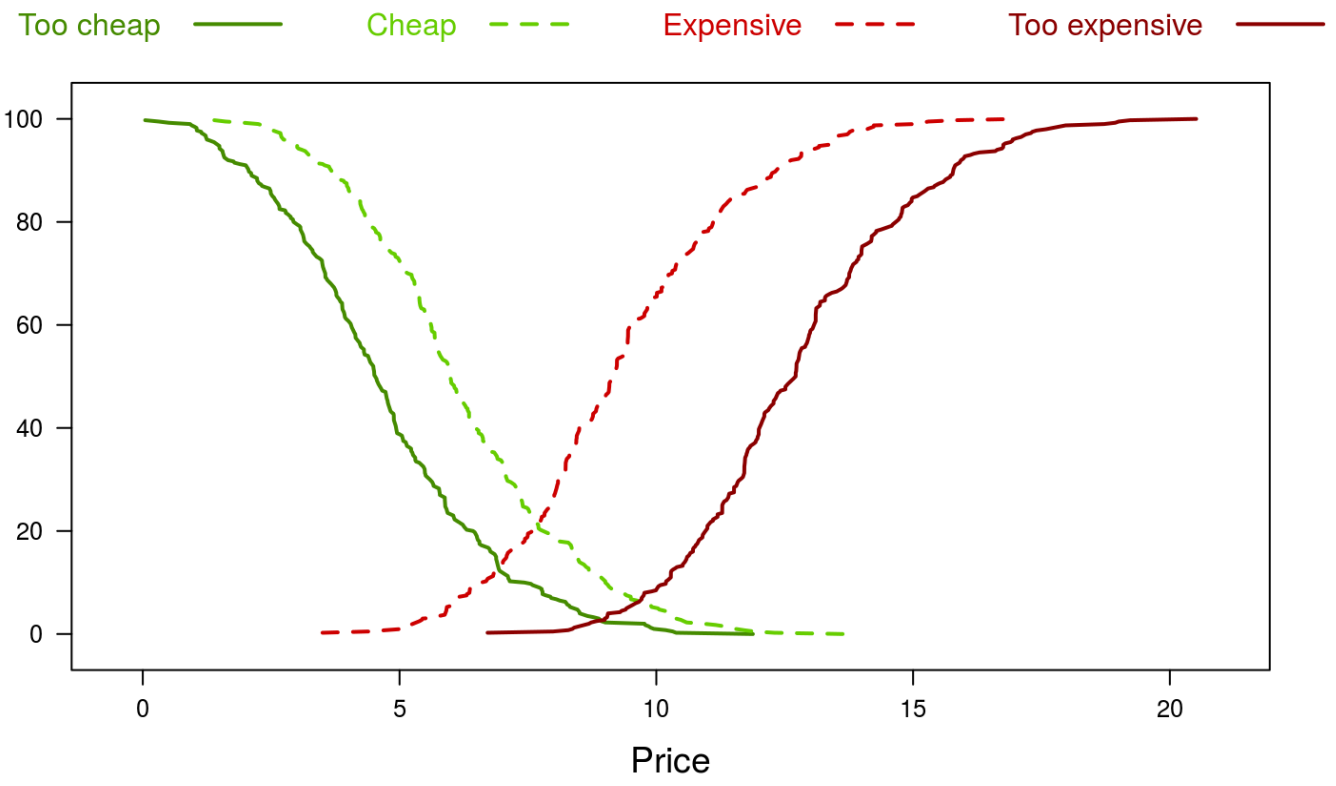
Utiliza el framework de lattice para poder representar los resultados de forma precocinada.

Ejemplo sin segmentar.

toocheap	cheap	expensive	tooexpensive
8.5	10.4	12.8	16.8
5.5	6.5	9.8	12.9
4.6	6.2	8.9	12.6
5.9	7.6	11.0	14.5
8.9	10.4	13.6	16.8
6.9	8.4	12.4	15.8

Ejemplo sin segmentar.

```
psmObject00<-psm(dat, "toocheap", "cheap", "expensive", "tooexpensive")
plot(psmObject00)
```



Ejemplo sin segmentar.

```
summary(psmObject00)
```

```
##     Grupo      Mín      Máx Indiferencia Óptimo
## 1 Generic 6.926399 9.659121     7.702996 8.9163
```

Ejemplo con segmentación.

toocheap	cheap	expensive	tooexpensive	group
10.5	10.3	12.2	16.1	Group_1
7.8	5.9	12.9	15.9	Group_1
8.9	9.1	14.0	14.6	Group_1
8.9	16.9	9.0	9.1	Group_1
7.8	6.8	13.2	10.7	Group_1
1.6	12.7	13.9	22.8	Group_1

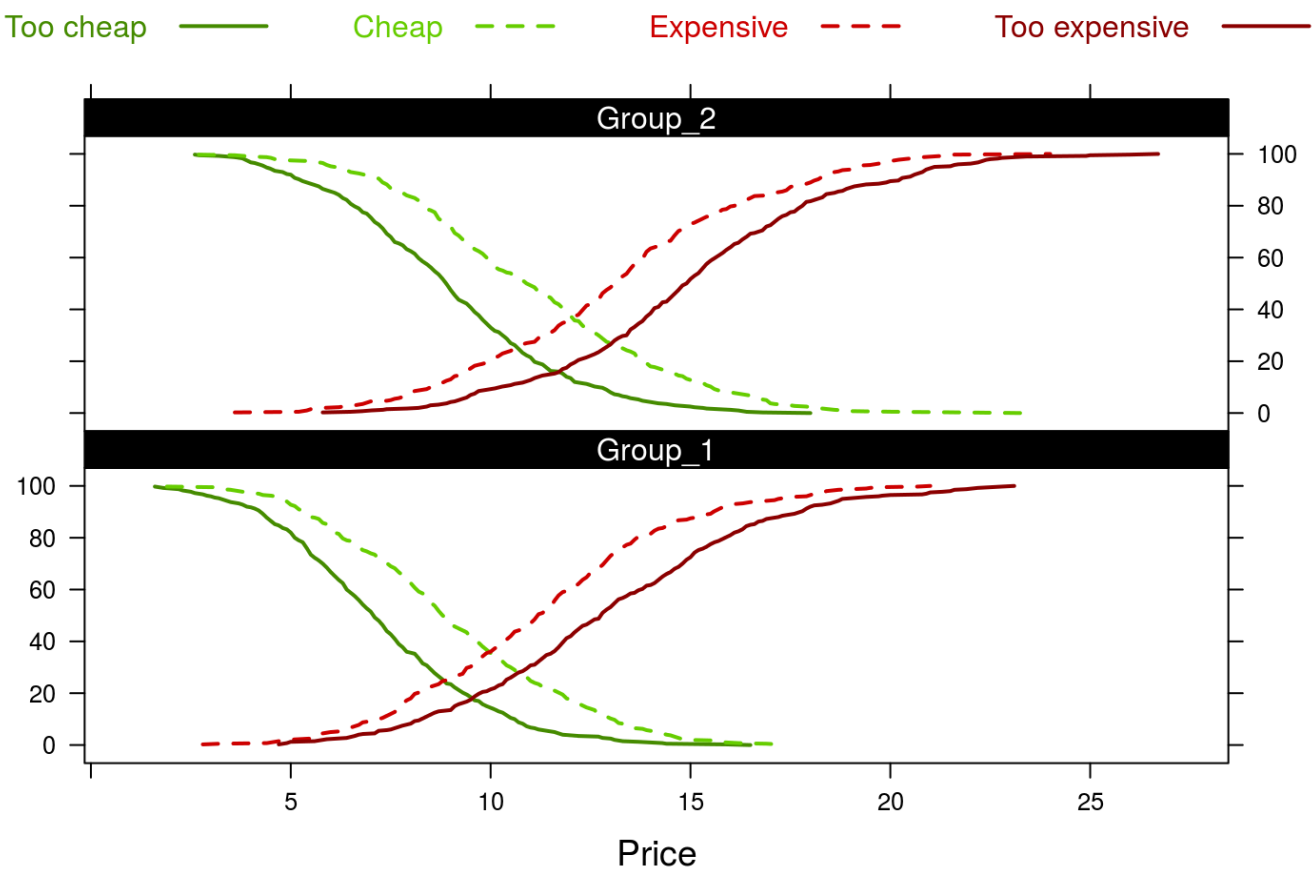
Ejemplo con segmentación.

```
psmObject<-psm(dat1, "toocheap", "cheap", "expensive", "tooexpensive", by= "group")
summary(psmObject)
```

```
##     Grupo       Mín      Máx Indiferencia  Óptimo
## 1 Group_1  8.828571 10.67273     9.971429  9.5400
## 2 Group_2 10.655556 13.02222    12.053846 11.7125
```

Ejemplo con segmentación.

```
plot(psmObject)
```

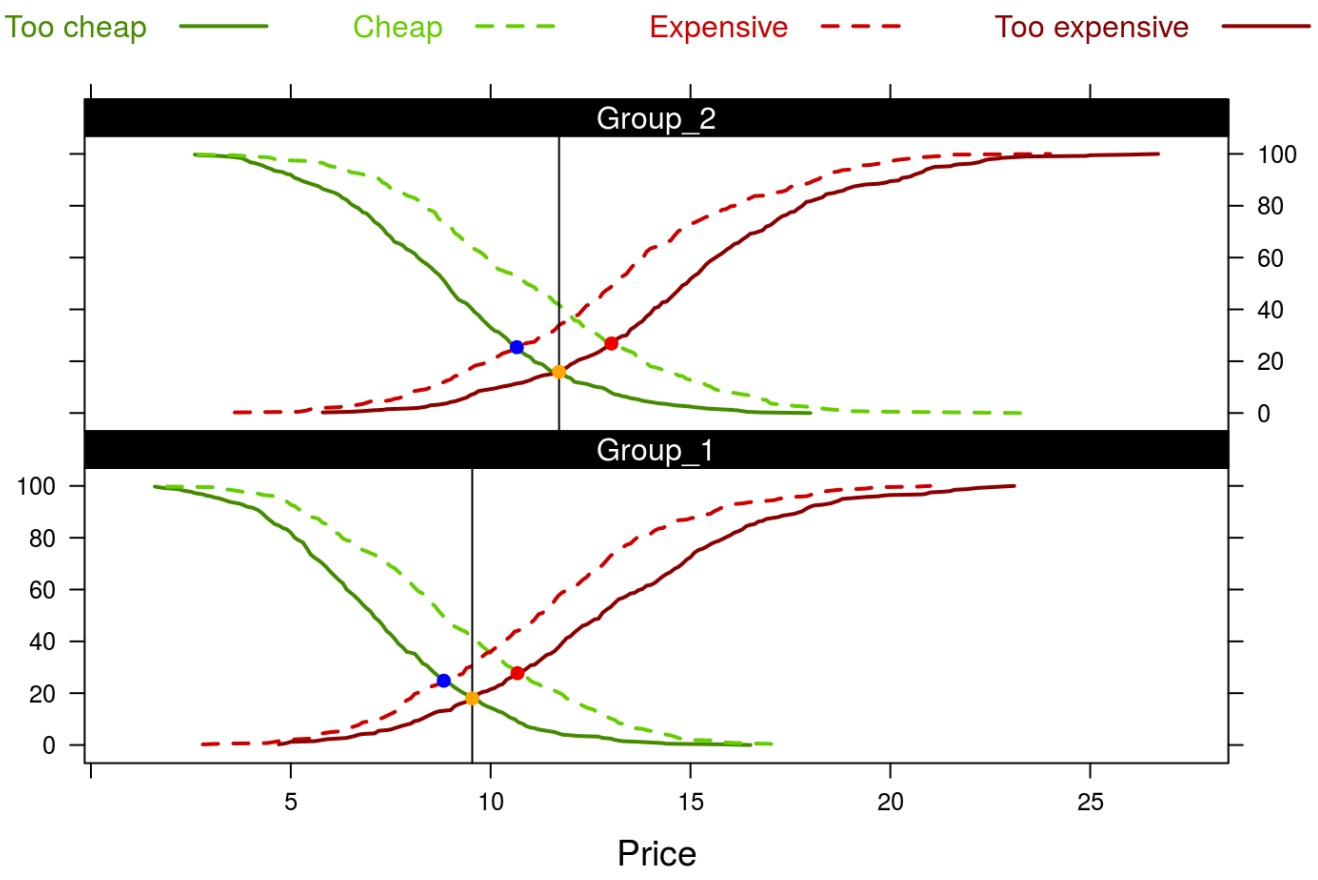


Personalización gráfica.

En lattice es posible trabajar con los objetos trellis para realizar cambios específicos. En este caso añadiré los puntos de intersección.

```
plot(psmObject)
paneles  = trellis.currentLayout()
for( i in seq_along(paneles)) {
  idx = which(paneles == i, arr.ind=TRUE)
  trellis.focus("panel", column = idx[2], row = idx[1])
  panel.abline(v=opprice[i,1], col="black")
  panel.points(x=minprice[i,1], y=100*minprice[i,2], pch=19, col="blue")
  panel.points(x=maxprice[i,1], y=100*maxprice[i,2], pch=19, col="red2")
  panel.points(x=opprice[i,1], y=100*opprice[i,2], pch=19, col="orange")
}
trellis.unfocus()
```

Personalización gráfica.



Personalización gráfica.

En este caso estoy utilizando una función para generar un gráfico precocinado.

```
args(plot.psm)
```

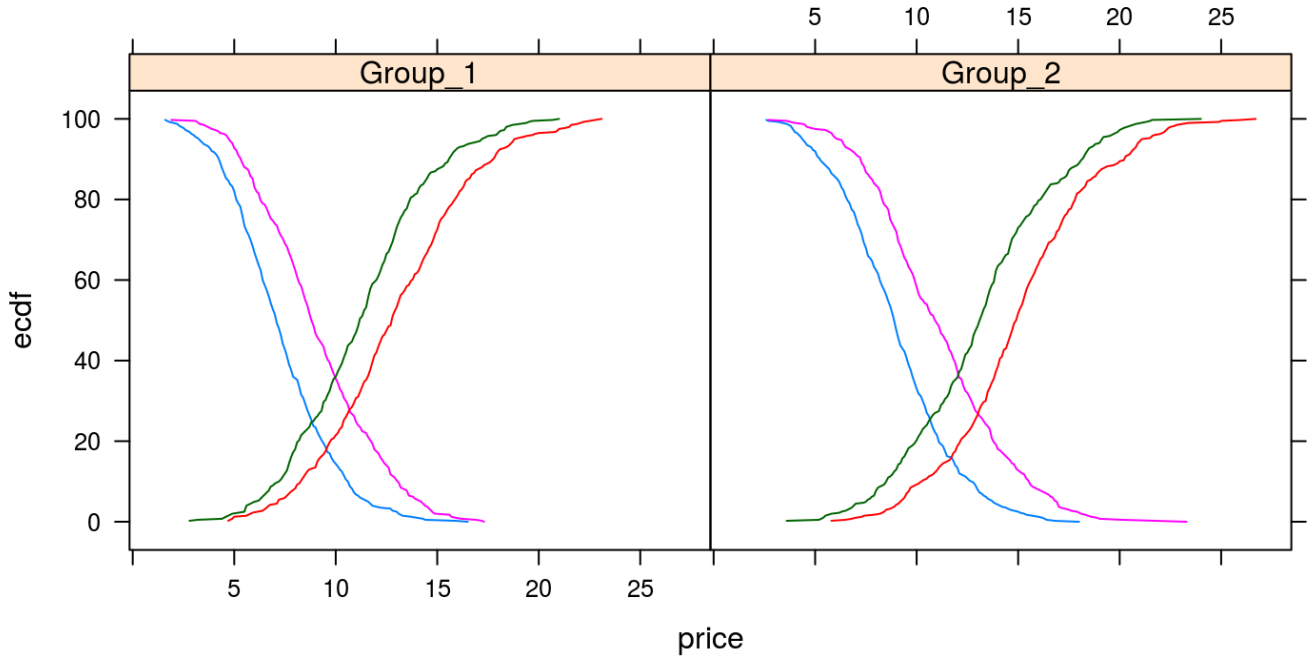
```
## function (psmObject, col = c("chartreuse4", "chartreuse3", "red3",
##     "red4"), names = c("Too cheap", "Cheap", "Expensive", "Too expensive"),
##     main = NULL, ylab = NULL, xlab = "Price", legend.position = "top",
##     legend.columns = 4)
## NULL
```

Sin embargo es interesante poner a disposición del usuario gráficos con sabor vainilla.

Personalización gráfica.

El mismo ejemplo anterior con sabor vainilla utilizando el framework de lattice.

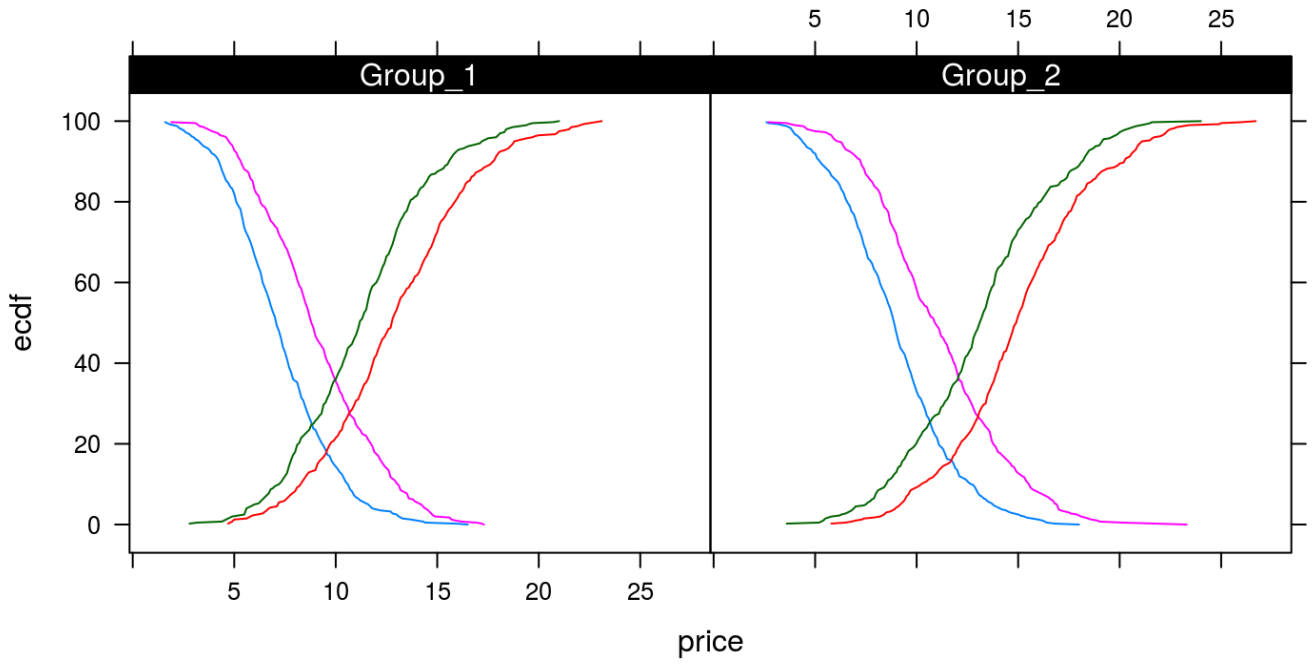
```
vplot(psmObject)
```



Personalización gráfica.

El objetivo de esto es evitar contradicciones en los parámetros y delegar en el usuario la personalización a través de las herramientas del framework.

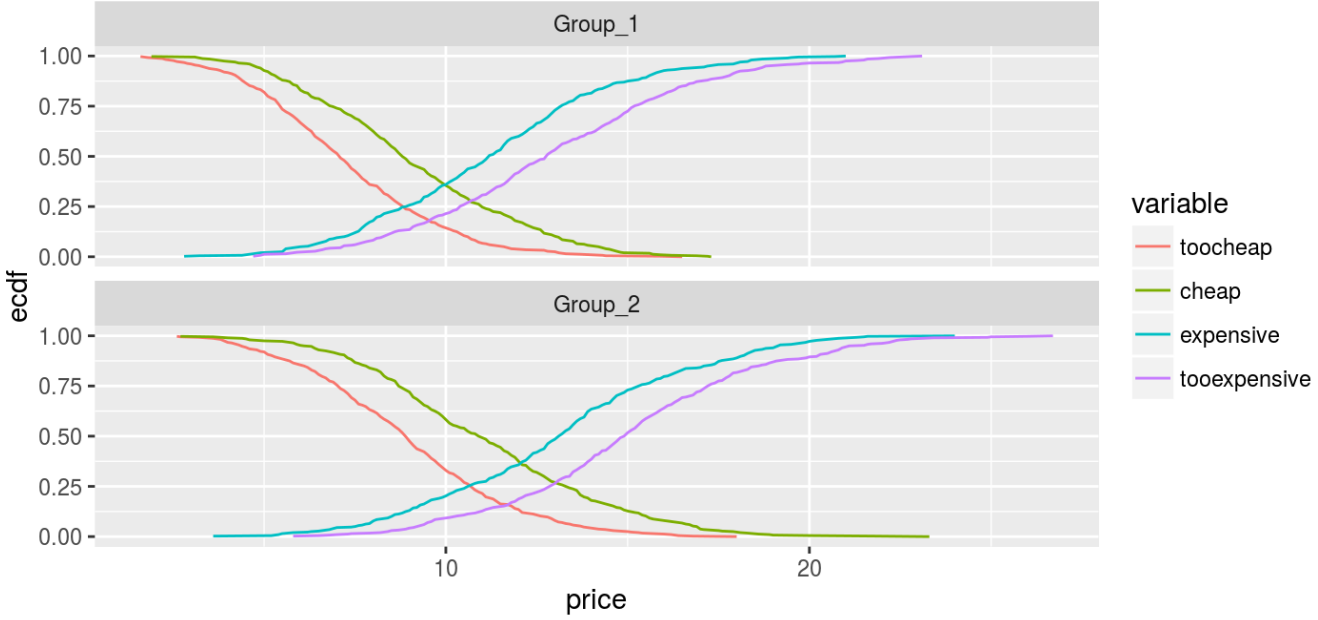
```
vplot(psmObject, par.strip.text=list(col=c("white")), strip=strip.custom(bg="black"))
```



Facilitar objetos adaptables a otras gramáticas.

Dentro del objeto psm está disponible el data.frame con el cual se construyen los gráficos, de forma que se puedan adaptar a otras gramáticas.

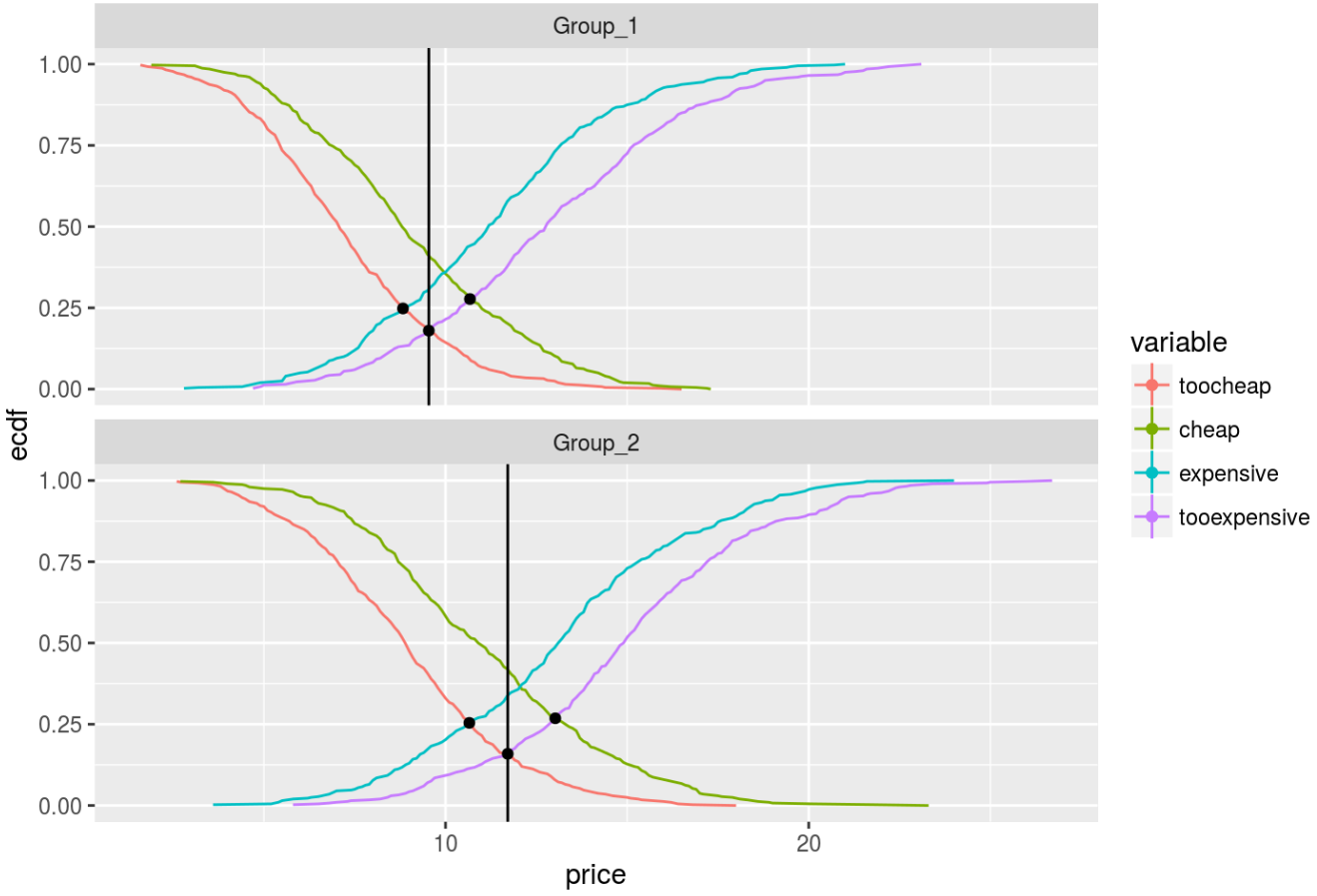
```
library(ggplot2)
ggplot(data=psmObject$data, aes(x=price, y=ecdf, colour=variable))+
  geom_line()+ facet_wrap(~group, ncol = 1)
```



Uso de otras gramáticas.

```
p <-ggplot(data=psmObject$data, aes(x=price, y=ecdf, colour=variable))+
  geom_line()+
  facet_wrap(~group, ncol = 1)
p + geom_point(data = minprice, aes(xintercept = price, yintercept = ecdf, colour=NULL),
               stat = "identity", position = "identity")+
  geom_point(data = opprice, aes(xintercept = price, yintercept = ecdf, colour=NULL),
               stat = "identity", position = "identity")+
  geom_point(data = maxprice, aes(xintercept = price, yintercept = ecdf, colour=NULL),
             stat = "identity", position = "identity")+
  geom_vline(data = opprice, aes(xintercept = price, colour=NULL))
```

Uso de otras gramáticas.



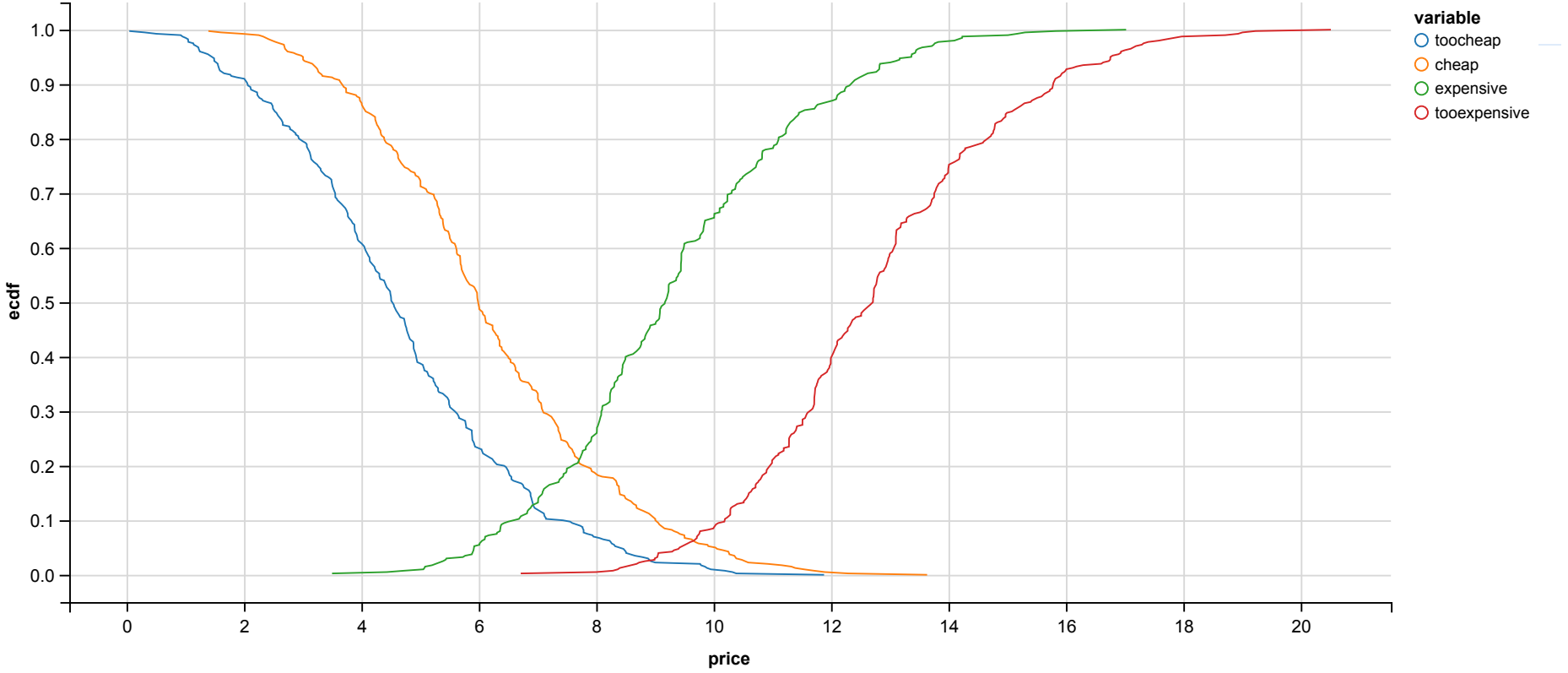
Uso de otras gramáticas.

ggvis

```
library(ggvis)
psmObject00$data %>%
  group_by(variable) %>%
  ggvis(x=~price, y=~ecdf, stroke=~variable) %>%
  layer_lines() %>%
  layer_points(opacity :=0) %>%
  add_tooltip(function(df){df$price}, "hover") %>%
  add_tooltip(all_values, "click")
```

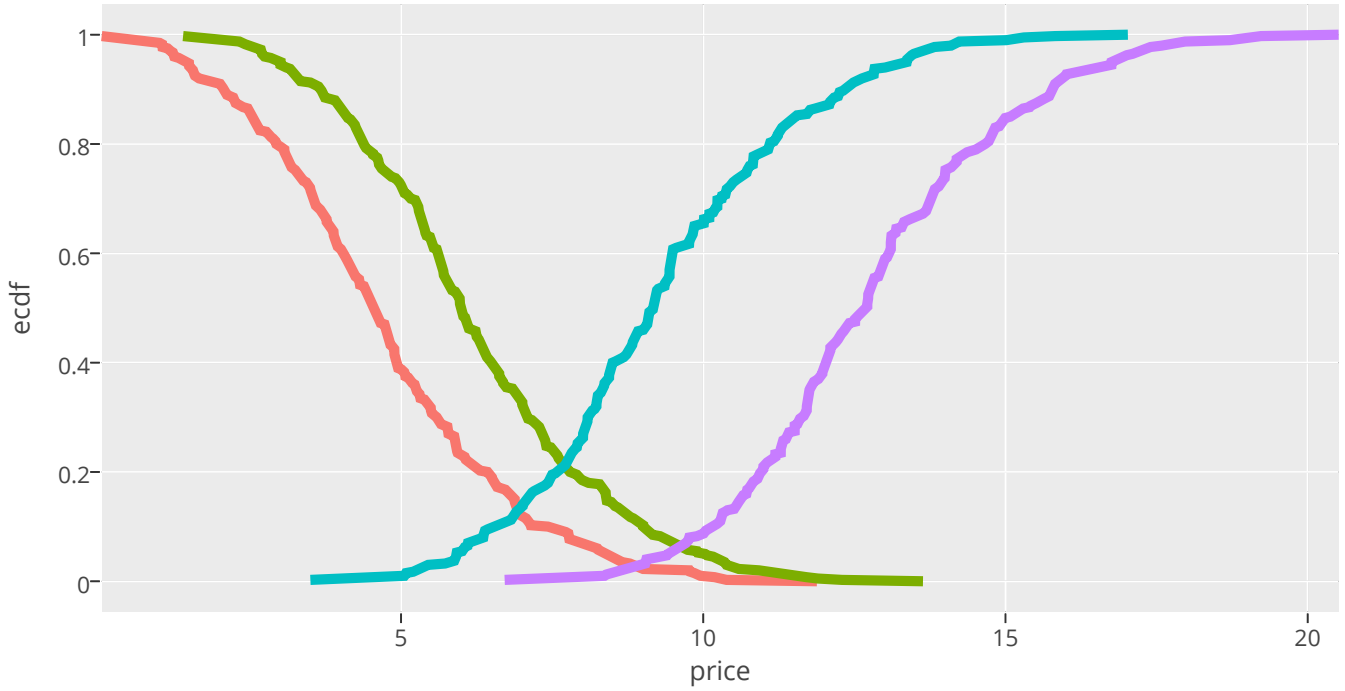
Los nuevos framework gráficos interactivos.

ggvis



Los nuevos framework gráficos interactivos.

```
library(plotly)
x<-ggplot(data=psmObject00$data, aes(x=price, y=ecdf, colour=variable))+
  geom_line()
ggplotly(p=x)
```



Muchas gracias.