Profesor: Héctor Bahamonde, PhD.

e:hector.bahamonde@uoh.cl

w:www.hectorbahamonde.com

Curso: MLE.

TA: Gonzalo Barría.

I. Inferencia e Interpretación

Los GLMs no se pueden interpretar usando la tabla. Los coeficientes no sifnifican lo que significaban en nuestro mundo OLS. Para interpretarlos, debemos usar otras herramientas.

Intervalos de confianza Ya sabemos lo que un intervalo de confianza significa: si nuestra confianza es al 95%, sabemos que el "true value" de la estimación caerá el 95% dentro del margen de los intervalos de confianza. Es en base a esto que antes hablabamos de "significancia estadística".

Carguemos los datos

```
# Data
mydata <- read.csv("https://stats.idre.ucla.edu/stat/data/binary.csv")
head(mydata)

##   admit gre  gpa rank
## 1     0 380 3.61    3
## 2     1 660 3.67    3
## 3     1 800 4.00    1
## 4     1 640 3.19    4
## 5     0 520 2.93    4
## 6     1 760 3.00    2

summary(mydata)

##      admit             gre             gpa             rank
##  Min.   :0.0000   Min.   :220.0   Min.   :2.260   Min.   :1.000
##  1st Qu.:0.0000   1st Qu.:520.0   1st Qu.:3.130   1st Qu.:2.000
```

```
##  Median :0.0000   Median :580.0   Median :3.395   Median :2.000
##  Mean   :0.3175   Mean   :587.7   Mean   :3.390   Mean   :2.485
##  3rd Qu.:1.0000   3rd Qu.:660.0   3rd Qu.:3.670   3rd Qu.:3.000
##  Max.   :1.0000   Max.   :800.0   Max.   :4.000   Max.   :4.000
```

Estimemos el primer modelo

```
logit.1 <- glm(admit ~ gre + gpa, data = mydata, family = binomial(link = "logit"))
summary(logit.1)
```

```
##
## Call:
## glm(formula = admit ~ gre + gpa, family = binomial(link = "logit"),
##     data = mydata)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2730  -0.8988  -0.7206   1.3013   2.0620
##
## Coefficients:
##              Estimate Std. Error z value   Pr(>|z|)
## (Intercept) -4.949378   1.075093  -4.604 0.00000415 ***
## gre          0.002691   0.001057   2.544     0.0109 *
## gpa          0.754687   0.319586   2.361     0.0182 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 499.98  on 399  degrees of freedom
## Residual deviance: 480.34  on 397  degrees of freedom
## AIC: 486.34
```

```
##
## Number of Fisher Scoring iterations: 4
```

Y ahora, estimemos los intervalos de confianza. Los intervalos de confianza son sencillos de calcular. Si queremos un 95% de confianza, miramos la tabla con los valores Z. Para ese porcentaje es 1.960 (para un 90% es 1.645).

$$\hat{\theta} \pm 1.960 \times \mathrm{SE}_{\hat{\theta}} \tag{1}$$

donde $\hat{\theta}$ es la estimación (el parámetro), 1.960 es el valor z, DS es la desviación estándard, y N es el largo de la base de datos. Si te fijas, tenemos el signo $\pm$ que implica que debemos restar para obtener el rango mínimo del intervalo de confianza, y sumar para obtener el rango máximo del intervalo de confianza.

Por ejemplo, el coeficiente de "gre" es $\hat{\theta} = 0.0026907$, el $SE = 0.001094$. Entonces, $0.002264426 + 1.96 * 0.001094$ para el intervalo superior, y $0.002264426 - 1.96 * 0.001094$ para el intervalo inferior.

Usemos un paquete.

```
confint(logit.1)

##                     2.5 %       97.5 %
## (Intercept) -7.1092205752 -2.886726963
## gre          0.0006373958  0.004791712
## gpa          0.1347509288  1.390131131
```

Odds Ratios Esta manera de interpretar solo funciona con los *link function* tipo logit (logit, multinomial logit, etc.), no probit links (i.e. probit, multinomial probit). Formalmente,

$$\begin{aligned} ln\Omega(\mathrm{x}) &= \mathrm{x}\beta \\ \Omega(\mathrm{x}) &= \frac{\Pr(\mathrm{y}=1|\mathrm{x})}{\Pr(\mathrm{y}=0|\mathrm{x})} \end{aligned} \tag{2}$$

donde Equation 2 es el *odd ratio* de que y sea 1 dado x, relativo a que y sea 0 dado x. Es un ratio, una fracción. Su interpretación es intuititiva: "cuando x cambia, cuánto cambia el logit estimado ($\hat{\theta}$) manteniendo los otros parametros constantes"?

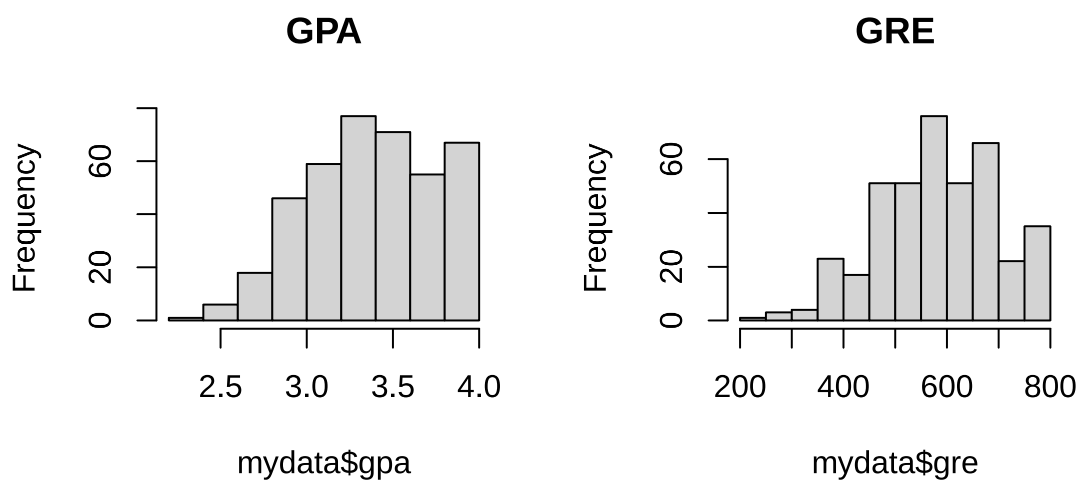
Lo bueno de esta interpretacion es que podemos observar el cambio de la variable dependiente. Ten en cuenta que los *odds ratios* son comparables sólo con la misma variable. **Lo malo** es que seguimos en unidades de la escala logit (que sigue siendo poco interpretable).

```
p_load(oddsratio)
#
or_glm(data = mydata,
       model = logit.1,
       incr = list(gre = 380, gpa = 5))

##   predictor oddsratio ci_low (2.5) ci_high (97.5) increment
## 1       gre     2.780        1.274           6.177       380
## 2       gpa    43.529        1.962        1043.834         5
```

Aquí vemos que la variable dependiente (`admit`) es 55 veces más probable de ocurrir cuando la variable independiente `gpa` incrementa de su media (3.3899) a 5. Como ves, ganamos interpretatibilidad del modelo.

```
par(mfrow=c(1,2))
hist(mydata$gpa, main = "GPA")
hist(mydata$gre, main = "GRE")
```



Como interpretamos `GRE`?

```
mean(mydata$gpa)

## [1] 3.3899

mean(mydata$gre)

## [1] 587.7
```

Si bien es cierto que ganamos interpretación del modelo, lo malo es seguimos hablando con poca precisión. Decir que algo es 55 veces más probable es interesante, pero no sé si tan "científico".

***Partial/Marginal Changes* in y** Los "cambios parciales" son muy parecidos a los *odds ratios*. Son parecidos porque nos muestra "cuando cambia $\hat{y}$ cuando cambia x". Formalmente,

$$\frac{\partial \hat{y}}{\partial x_k} = \beta_k \tag{3}$$

lo que significa "por un cambio en x_k, se espera que $\hat{y}$ cambie β_k" (manteniendo todas las otras variables constantes en su media). Sin embargo, debido a que la varianza de $\hat{y}$ es desconocida (la varianza es un *population parameter*), entonces, esto complica la interpretacion de β_k. Para resolver esto, lo que hacemos es pensar en coeficientes β_k estandarizados. Siguiendo Equation I, lo que pensamos es en "por un cambio en x_k, se espera que $\hat{y}$ cambie β_k **desviaciones estándard**". Formalmente,

$$\beta_k^S = \frac{\sigma_k \beta_k}{\sigma_{\hat{y}}} = \sigma_k \beta_k^{S_{\hat{y}}} \tag{4}$$

Nota que también estandariza $\hat{y}$. Debido a que $\hat{\sigma}_{\hat{y}} = \sigma_{\hat{y}}$ (i.e. podemos estimar la varianza estandarizada de los datos),

$$\hat{\sigma}_{\hat{y}} = \beta^\top \hat{\sigma}(x)\beta + \sigma(\epsilon) \tag{5}$$

Calculemos dos escenarios. Uno donde el estudiante le iba muy mal en el colegio, pero tuvo un buen puntaje en la prueba de admisión de doctorado (GRE).

```
p_load(margins)
summary(margins(logit.1,
        at = list(
          gre = min(mydata$gre),
          gpa = max(mydata$gpa))
        ))

##  factor      gre    gpa    AME     SE      z      p   lower  upper
##     gpa 220.0000 4.0000 0.1242 0.0808 1.5364 0.1244 -0.0342 0.2827
##     gre 220.0000 4.0000 0.0004 0.0001 5.9659 0.0000  0.0003 0.0006
```

Ahora calculemos el opuesto:

```
summary(margins(logit.1,
        at = list(
          gre = max(mydata$gre),
          gpa = min(mydata$gpa):mean(mydata$gpa))
          ))

##  factor      gre    gpa    AME     SE      z      p   lower  upper
##     gpa 800.0000 2.2600 0.1420 0.0325 4.3633 0.0000  0.0782 0.2058
```

```
##     gpa 800.0000 3.2600 0.1834 0.0741 2.4771 0.0132  0.0383 0.3286
##     gre 800.0000 2.2600 0.0005 0.0003 1.6967 0.0898 -0.0001 0.0011
##     gre 800.0000 3.2600 0.0007 0.0003 2.3188 0.0204  0.0001 0.0012
```

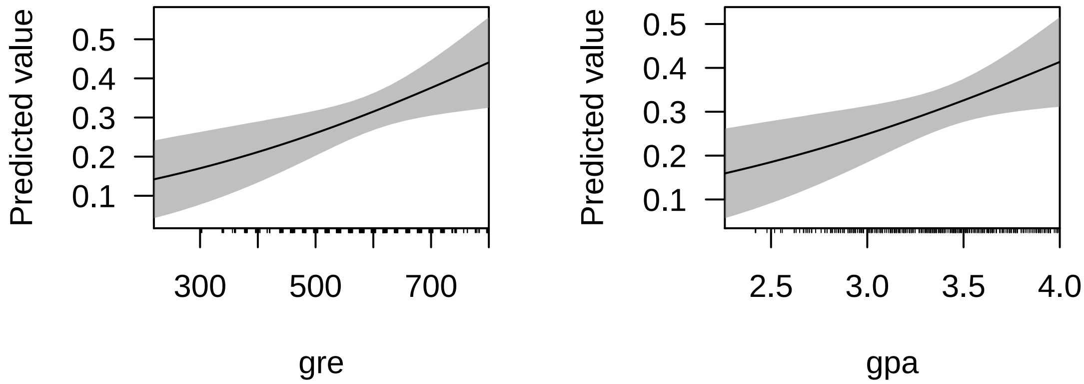
Para entender mejor esto, grafiquemos:

```r
par(mfrow=c(1,2))
cplot(logit.1, "gre")
```

```
##       xvals     yvals     upper      lower
## 1  220.0000 0.1419589 0.2412899 0.04262786
## 2  244.1667 0.1500652 0.2479355 0.05219492
## 3  268.3333 0.1585489 0.2545321 0.06256566
## 4  292.5000 0.1674177 0.2610725 0.07376288
## 5  316.6667 0.1766785 0.2675539 0.08580299
## 6  340.8333 0.1863368 0.2739798 0.09869393
## 7  365.0000 0.1963973 0.2803622 0.11243234
## 8  389.1667 0.2068628 0.2867259 0.12699962
## 9  413.3333 0.2177348 0.2931134 0.14235622
## 10 437.5000 0.2290133 0.2995933 0.15843330
## 11 461.6667 0.2406962 0.3062725 0.17512003
## 12 485.8333 0.2527798 0.3133146 0.19224490
## 13 510.0000 0.2652580 0.3209666 0.20954944
## 14 534.1667 0.2781229 0.3295882 0.22665765
## 15 558.3333 0.2913644 0.3396690 0.24305980
## 16 582.5000 0.3049699 0.3517828 0.25815704
## 17 606.6667 0.3189249 0.3664325 0.27141726
## 18 630.8333 0.3332122 0.3838358 0.28258868
## 19 655.0000 0.3478128 0.4038425 0.29178301
## 20 679.1667 0.3627051 0.4260603 0.29934986
```

```r
cplot(logit.1, "gpa")
```

```
##      xvals     yvals     upper      lower
## 1  2.2600 0.1594306 0.2615309 0.05733021
## 2  2.3325 0.1669004 0.2668014 0.06699931
## 3  2.4050 0.1746474 0.2719973 0.07729752
## 4  2.4775 0.1826752 0.2771187 0.08823181
## 5  2.5500 0.1909867 0.2821696 0.09980371
## 6  2.6225 0.1995839 0.2871600 0.11200782
## 7  2.6950 0.2084684 0.2921072 0.12482964
## 8  2.7675 0.2176409 0.2970392 0.13824262
## 9  2.8400 0.2271012 0.3019984 0.15220395
## 10 2.9125 0.2368481 0.3070478 0.16664844
## 11 2.9850 0.2468798 0.3122801 0.18147962
## 12 3.0575 0.2571932 0.3178294 0.19655702
## 13 3.1300 0.2677843 0.3238891 0.21167944
## 14 3.2025 0.2786478 0.3307294 0.22656631
## 15 3.2750 0.2897777 0.3387074 0.24084799
## 16 3.3475 0.3011665 0.3482440 0.25408903
## 17 3.4200 0.3128058 0.3597384 0.26587323
## 18 3.4925 0.3246859 0.3734345 0.27593734
## 19 3.5650 0.3367961 0.3893290 0.28426321
## 20 3.6375 0.3491245 0.4072024 0.29104653
```



Predicted Probabilities: Gráficos Quizás esta sea la manera más usada para poder interpretar: graficando (o mostrando en una tabla) "la probabilidad de que y sea 1 a distintos valores de x", o de manera más formal,

$$\hat{\Pr}(y_i = 1|\mathbf{x}_i) = f(\mathbf{x}_i\hat{\beta}) \tag{6}$$

Fíjate que $\mathbf{x}$ y $\hat{\beta}$ son matrices. Por que?

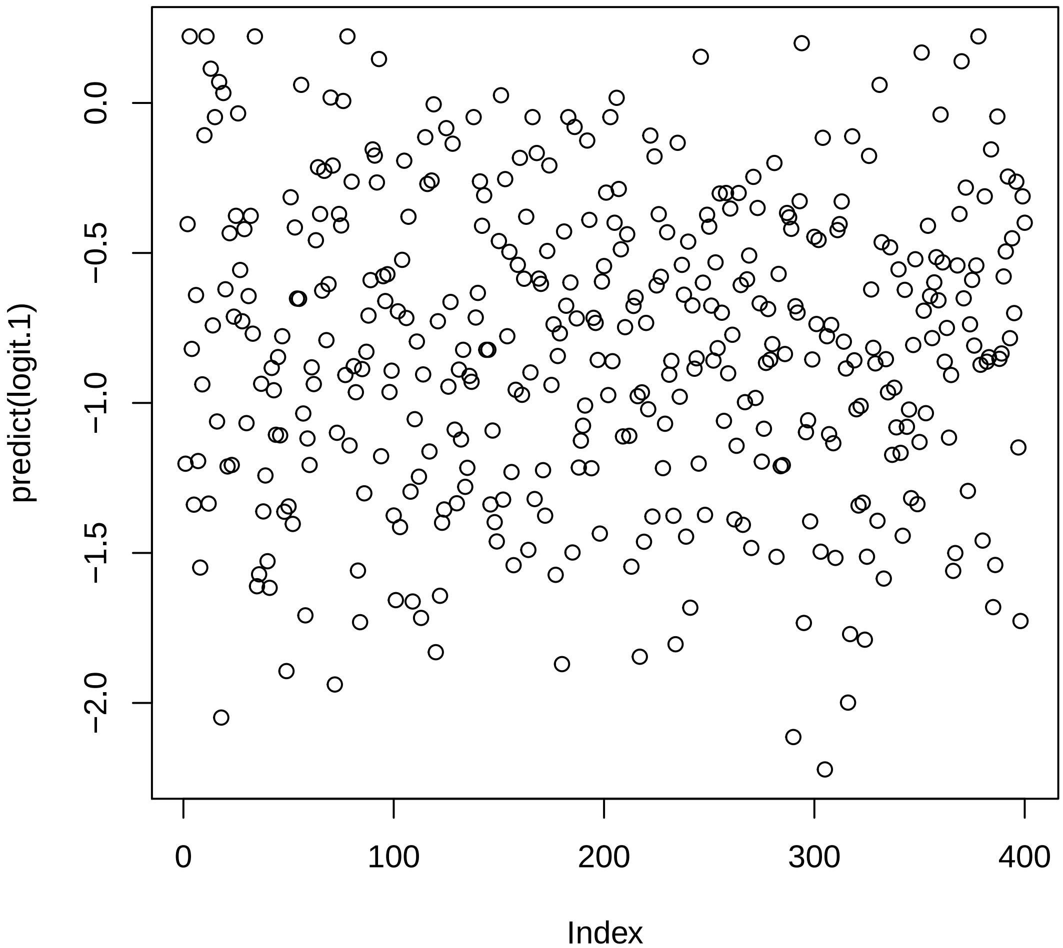
Aqui lo que queremos saber es como se comparta la probabilidad de que $y_i = 1$ a medida que nos movemos en una de las x's. **Lo bueno**, es que tenemos un lenguaje fácil de entender: todos entienden probabilidades, y además, estas varían entre 0 y 1. Pero qué hacemos con el resto de las x's?

Para predecir usaremos el comando `predict`

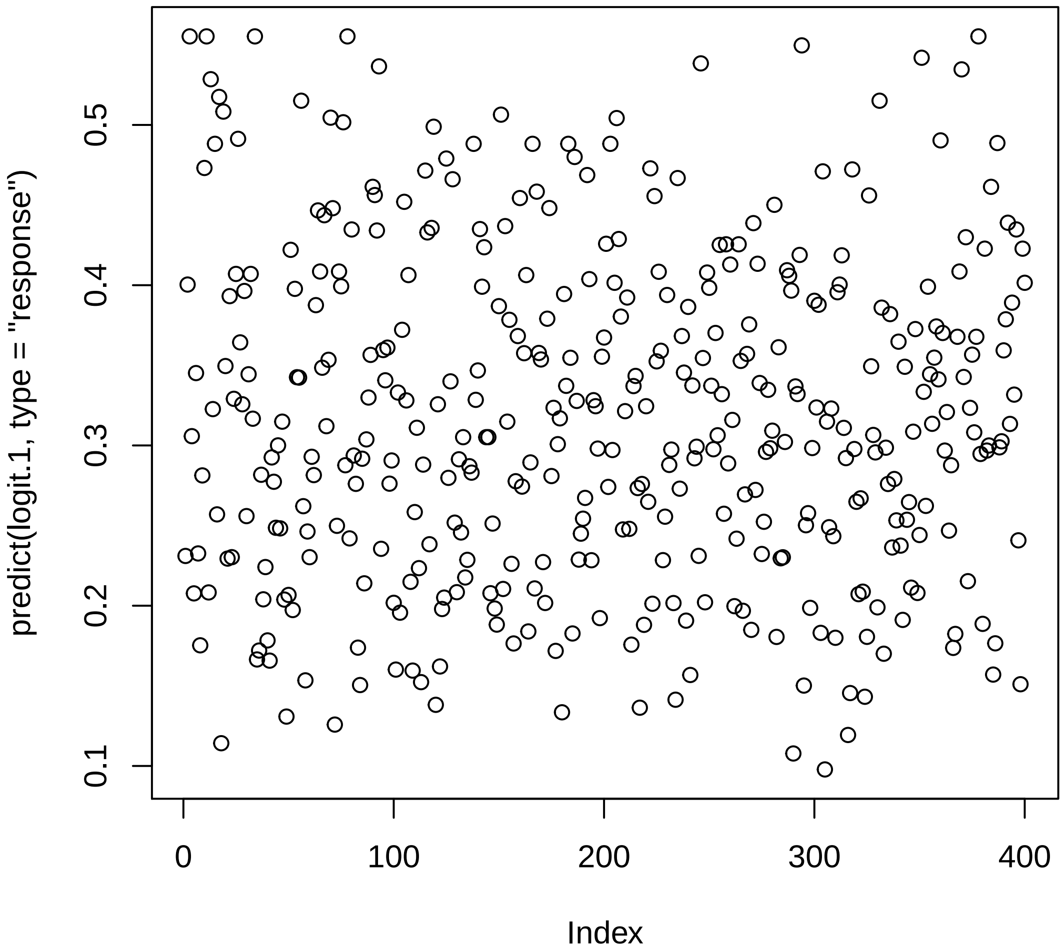
```
head(predict(logit.1))

##          1          2          3          4          5          6
## -1.2024987 -0.4038261  0.2219162 -0.8198895 -1.3389901 -0.6403980

plot(predict(logit.1)) # logit scale
```

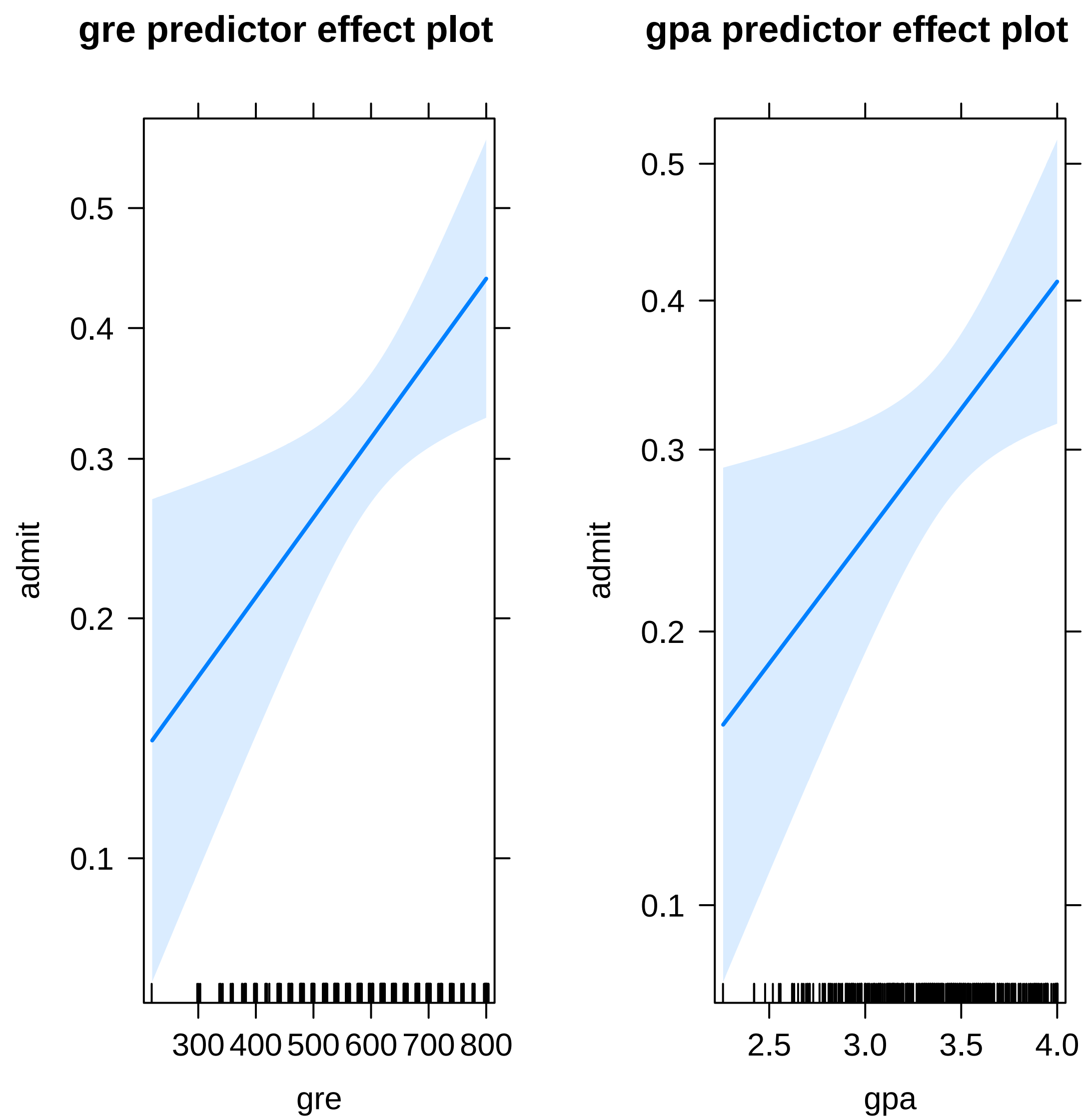



```
plot(predict(logit.1, type="response")) # Predicted Prob
```

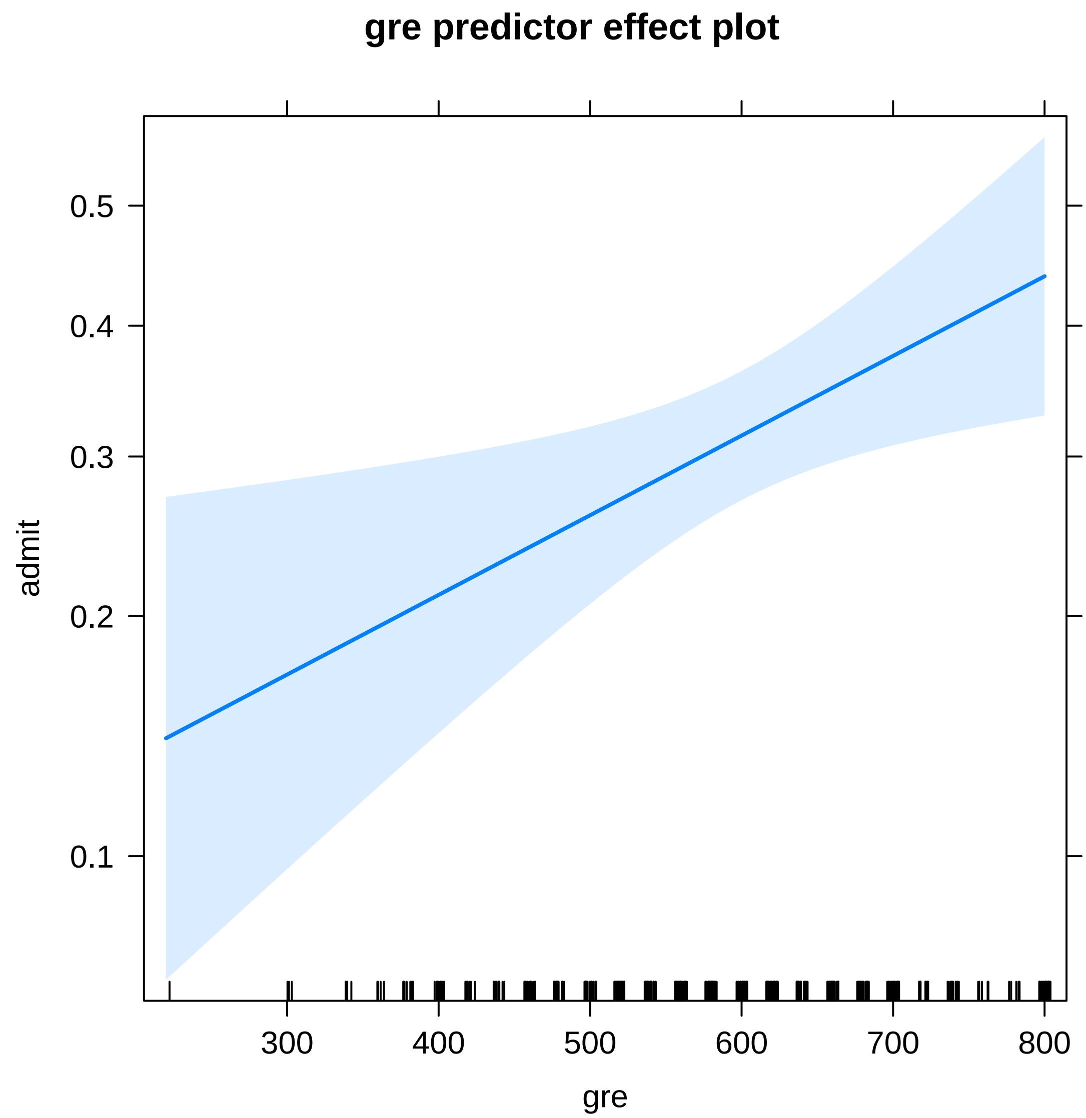


Usando la librería `effects`, podremos graficar.

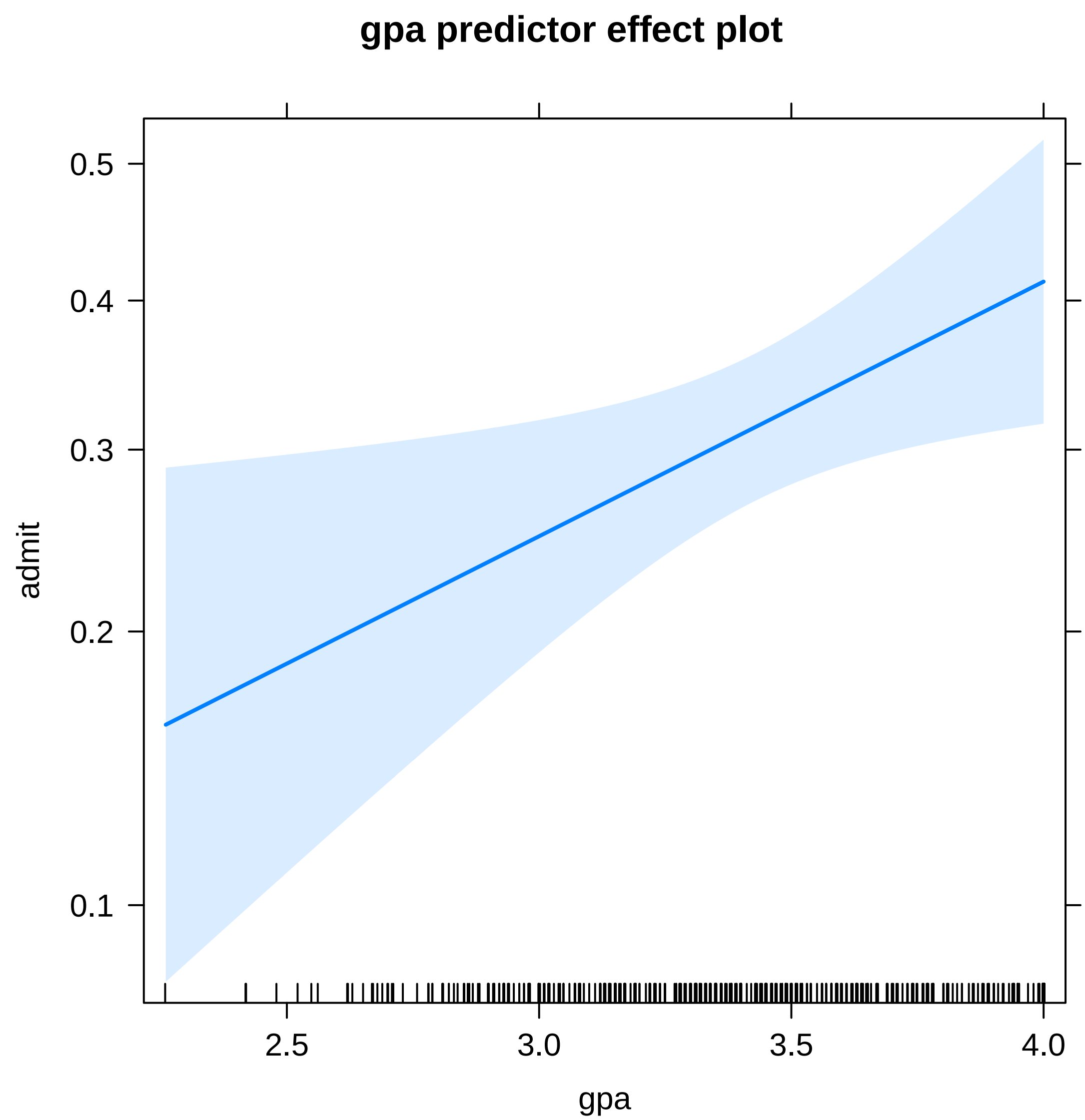
```
p_load("effects")
par(mfrow=c(1,1))
plot(predictorEffects(logit.1)) # Todo el modelo (PLURAL)
```




```
par(mfrow=c(1,2))
plot(predictorEffect("gre", logit.1)) # Solo una variable (SINGULAR)
```




```
plot(gpa.pred.prob <- predictorEffect("gpa", logit.1)) # Solo una variable (SINGULAR)
```



Predicted Probabilities: Tablas La otra alternativa es poder generar tablas con *predicted prob's*. La idea es ir generando "perfiles" que hagan sentido desde un punto de vista substantivo.

```
# generar los rangos de los perfiles deseados
gpa.bajo <- with(mydata,
                 data.frame(
```

```
                        gre = c(min(mydata$gre), max(mydata$gre)),
                        gpa = min(mydata$gpa))
                )
gpa.medio <- with(mydata,
                  data.frame(
                         gre = c(min(mydata$gre), max(mydata$gre)),
                         gpa = mean(mydata$gpa, na.rm = T))
                         )
gpa.alto <- with(mydata,
                      data.frame(
                              gre = c(min(mydata$gre), max(mydata$gre)),
                              gpa = max(mydata$gpa))
                              )
```

```
# usar funcion predict
predict(logit.1, gpa.bajo, type="response")

##          1          2
## 0.06587598 0.25138506

predict(logit.1, gpa.medio, type="response")

##         1         2
## 0.1419589 0.4406515

predict(logit.1, gpa.alto, type="response")

##         1         2
## 0.2077272 0.5552525
```

GPA Bajo		**GPA Medio**		**GPA Alto**	
Gre Min	*Gre Max*	*Gre Min*	*Gre Max*	*Gre Min*	*Gre Max*
0.07	0.25	0.14	0.44	0.21	0.56

Cuál es el problema?

```
knitr::purl('Inferencia_Interpretacion.Rnw')

## Error in parse_block(g[-1], g[1], params.src, markdown_mode):  Duplicate chunk label
'setup20', which has been used for the chunk:
## if (!require("pacman")) install.packages("pacman"); library(pacman)
## p_load(knitr)
## set.seed(2020)
## options(scipen=9999999)

Stangle('Inferencia_Interpretacion.Rnw')

## Writing to file Inferencia_Interpretacion.R
```