Jeroen Claes (Ghent University/KU Leuven)

Análisis de regresión: Taller

Antes de hacer un análisis de regresión: estadísticas descriptivas

1. Cambiar la carpeta de trabajo (solo una vez): Rstudio> Tools > Global options > R general > Default working directory (when not in a Project). Lo más fácil es que la carpeta de trabajo sea el escritorio.
2. A R no le gustan los espacios. Si los niveles de tus variables o los nombres de tus variables contienen espacios, hay que reemplazarlos por puntos (.).
3. Antes de empezar, vamos a guardar los datos en formato CSV para que R pueda leerlos.

En Excel: guardar como > CSV (Windows)

4. Ahora vamos a cargar el archivo y guardar la base de datos en el objeto *datos*:

```
#seleccionar el archivo de datos

datos<-read.csv2(file.choose(), header=T, fileEncoding = "iso-8859-1")
```

Para ver las primeras 6 filas de nuestro archivo de datos, introducimos:

```
head(datos)
```

5. Tenemos que bajar e instalar algunos paquetes que nos van a permitir hacer los análisis.

```
install.packages(c("lme4", "MuMIn", "car", "Hmisc", "effects", "lattice", "snow", "ggplot2", "vcd", "plyr", "grid"))
```

6. Ahora vamos a cargar algunas funciones que escribí yo para que podamos obtener las estadísticas sin tener que insertar mucho código. Después vamos a guardar el espacio de trabajo para que esas funciones se queden ahí la próxima vez que abramos R Studio.

```
source("http://www.jeroenclaes.be/scripts/taller.R")

save.image()
```

7. Para hacer una tabla de contingencia vamos a usar mi función *tabler (nombre objeto de datos, "nombre de la columna", "nombre de la columna que contiene variable dependiente")*

```
tabler(datos, "Co.2.Pr", "Tipo")
```

El *output* de esta función se muestra abajo. Consta de dos tablas: una de frecuencias relativas, otra de frecuencias absolutas. Bajo *Significance* se ofrece el valor *p* para la tabla. Cuando p < 0.05 se suele decir que el resultado es "significante" estadísticamente. Bajo *Effect size* se ofrecen varios estimados del tamaño del efecto de la variable independiente, cuanto más alto sean esos números, más grande es el efecto de la variable.

```
---Tables---
            dep
var          Pluralized Singular
  Pluralized        151      113
  Singular           63      122
  No.Priming        646      747
            dep
var          Pluralized  Singular
  Pluralized  0.5719697 0.4280303
  Singular    0.3405405 0.6594595
  No.Priming  0.4637473 0.5362527

---Significance---

        Fisher's Exact Test for Count Data with simulated p-value (based on 2000 replicates)

data:  tab
p-value = 0.0004998
alternative hypothesis: two.sided


---Effect size---
                   X^2 df   P(> X^2)
Likelihood Ratio 23.881  2 6.5213e-06
Pearson          23.632  2 7.3844e-06

Phi-Coefficient   : NA
Contingency Coeff.: 0.113
Cramer's V        : 0.113
>
```

Si en este momento se dan cuenta de que varios niveles de una variable producen más o menos el mismo resultado, es mejor juntar los niveles de esa variable, con tal de que de esa manera se obtenga una agrupación que haga sentido desde el punto de vista teórico/lingüístico.

Para hacerlo, podríamos volver a Excel, hacer los cambios, guardar el archivo y volver a cargar la base de datos o podemos recodificar los datos en R. Para hacerlo, necesitamos el paquete *plyr*.

```
install.packages("plyr")

library("plyr")
```

Para recodificar una columna tenemos que especificar el objeto de datos y la columna mediante el operador $: *nombre del objeto $ nombre de la columna*. Luego viene el

llamado a la función *revalue* del paquete *plyr* y otra vez el objeto de datos y la columna. Después de la coma especificamos los valores que tienen que reemplazarse y sus valores nuevos.

```
datos$Co.2.Pr<-revalue(datos$Co.2.Pr, c("Singular"="Singular.no.priming",
"No.Priming"="Singular.no.priming"))
```

Cuando volvemos a hacer las tablas, ahora vemos que la variable *Co.2.Pr* ya no tiene tres, sino dos niveles: *Pluralized* y *Singular.no.priming*

```
tabler(datos, "Co.2.Pr", "Tipo")
```

```
---Tables---
                     dep
var                   Pluralized Singular
  Pluralized                 151      113
  Singular.no.priming        709      869
                     dep
var                   Pluralized  Singular
  Pluralized           0.5719697 0.4280303
  Singular.no.priming  0.4493029 0.5506971
```

Para guardar el archivo recodificado, usamos la función *write.csv2()*. El archivo datos.csv se encuentra en el escritorio.

```
write.csv2(datos, file="datos.csv", fileEncoding = "iso-8859-1")
```

8. Repitamos estos pasos para todas las variables independientes.

2. Análisis de regresión logística de efectos mixtos

1) Primer paso: cargar paquetes y el archivo de datos

Instalar e cargar paquetes:

```
#Sólo la primera vez

install.packages(c("lme4", "MuMIn", "car", "Hmisc", "effects", "lattice", "snow"))

#Siempre:

library("lme4"); library("MuMIn"); library("car"); library("Hmisc"); library("effects")
```

Cargar el archivo de datos (si ya está cargado, omite este paso)

```
datos<-read.csv2(file.choose(), header=T, fileEncoding = "iso-8859-1")
```

Para ver las primeras 6 filas de nuestro archivo de datos:

```
head(datos)
```

2) Segundo paso: especificar los contrastes y determinar el nivel de aplicación de la variable dependiente

Cuando vamos a hacer un análisis de regresión, hay varias posibilidades en cuanto al nivel básico con referencia al cual se calculan los efectos de las variables. En R, la manera estándar de establecer contrastes/comparaciones entre los niveles de las variables son los "contrastes de tratamiento". Ese término que deriva de las ciencias médicas indica que el programa calculará el efecto de cada una de las variables independientes (los grupos que recibieron el 'tratamiento') con referencia a uno de los niveles de esas variables (el grupo de control en experimentos médicos)

Por ejemplo: Tenemos una variable *Priming* con dos niveles: A y B. En el contexto descrito por A, la variable dependiente aparece en el 10% de los casos. En el contexto descrito por B, la variable dependiente aparece en el 60% de los casos. Cuando los contrastes son de tratamiento, R calculara el efecto del nivel B con referencia a la distribución que se observa en el contexto A, por lo que el efecto será: 60%-10%= +50%.

Sin embargo, en lingüística, no siempre resulta muy claro cuál es el nivel básico de una variable. Por ejemplo, ¿son los contextos negativos más o menos básicos que los contextos afirmativos? ¿Es el imperfecto más básico que el pretérito? ¿Cuál pueda ser el tratamiento? Por ello es mejor usar los contrastes de suma. Este tipo de contrastes no compara la desviación causada por un nivel de una variable con respecto a otro nivel de la misma variable, sino que expresa el efecto del nivel de la variable independiente con respecto a la tasa global de uso de las variantes.

Por ejemplo: Tenemos una variable *Priming* con dos niveles: A y B. En el contexto descrito por A, la variable dependiente aparece en el 10% de los casos. En el contexto descrito por B, la variable dependiente aparece en el 60% de los casos. Cuando los contrastes son de suma, R calculara el efecto del nivel B con referencia a la distribución que se observa en corpus entero, por lo que el efecto será: 60%-35%= +25%.

Para establecer los contrastes de suma (con etiquetas inteligibles), introducimos el código siguiente:

```
library("car")
```

```
options(contrasts = c("contr.Sum", "contr.Poly"))
```

Ahora solamente nos falta asegurarnos de que el nivel de la variable dependiente para el cual se van a calcular los análisis sea el que nos interesa. Por defecto, R organiza los niveles de las variables alfabéticamente. El primer nivel siempre es el nivel de referencia. Eso implica que se calculan todos los resultados desde la perspectiva del segundo nivel, el nivel de aplicación. Cuando el nivel de la variable independiente que nos interesa (*Pluralized* en nuestro ejemplo) viene primero, hay que cambiar el orden, pues, si no, los resultados expresarán la probabilidad de encontrar el nivel *Singular*.

Para conocer el orden de los niveles:

```
levels(datos$Tipo)
```

Lo que da:

```
[1] "Pluralized" "Singular"
```

Para cambiar el orden:

```
datos$Tipo<-relevel(datos$Tipo, ref=2)
```

Si ahora hacemos:

```
levels(datos$Tipo)
```

Produce:

```
[1] "Singular" "Pluralized"
```

3) Tercer paso: formular el modelo más complejo que se puedan imaginar

En R podemos hacer análisis de regresión logística de efectos mixtos mediante el paquete *lme4* (Bates, Maechler, Bolker, & Walken, 2016). Los modelos de efectos mixtos nos permiten:

1) Calcular el efecto de las variables independientes, tomando en cuenta los efectos de todas las demás variables independientes (= la ventaja básica de cualquier análisis de regresión)

2) Calcular el efecto de las variables independientes, tomando en cuenta las preferencias individuales de los hablantes o ítems léxicos por una u otra variante. Este tipo de variables se conocen como **efectos aleatorios/variables aleatorias.**

Hay dos tipos de variables aleatorias: los interceptos aleatorios y las pendientes aleatorias.

> Los **interceptos aleatorios** nos permiten tomar en cuenta que una de las variantes de nuestra variable dependiente aparece más o menos frecuentemente con algunas palabras o algunos hablantes. El tamaño del efecto de las variables es idéntico para todos los hablantes y todas las palabras (p.ej. +25%), pero el programa toma en cuenta la proporción media de uso de la variable dependiente que se observa para el hablante o el ítem léxico (p. ej. Hablante 1: proporción media de uso = 15%; Hablante 2: proporción media de uso = 25%… Efecto de la variable para grupo 1: 15%+25%=40%; para grupo 2: 25%+25%=50%).
>
> Las **pendientes aleatorias**, en cambio, nos permiten tomar en cuenta tanto las proporciones medias de uso de las variantes como la variación entre los individuos en cuanto a los tamaños de los efectos de las variables independientes. Así, por ejemplo, el tamaño del efecto de una variable puede ser de +25% para el conjunto de los hablantes, pero puede reducirse a +5% para un hablante o amplificarse hasta +60% para otro.

Para especificar un modelo de regresión de efectos mixtos, el paquete *lme4* usa un lenguaje de fórmulas, acá va un ejemplo.

```
modelo<- glmer(Tipo ~ Typical.Action.Chain.Pos + Pr.2.Pr + Co.2.Pr + Age +
Education+ Gender + Tense + Negation + Typical.Action.Chain.Pos * Tense +
Pr.2.Pr*Co.2.Pr + Tense * Co.2.Pr + Tense * Pr.2.Pr + Tense * Negation +
(1+Typical.Action.Chain.Pos + Pr.2.Pr + Co.2.Pr + Tense + Negation | Muestra)+ (1
+Pr.2.Pr + Co.2.Pr + Tense + Negation | Nombrelema), data=datos,
family="binomial", control=glmerControl(optimizer="nloptwrap",
calc.derivs=FALSE))
```

Consideremos esta fórmula paso por paso.

- **modelo:** el nombre del objeto en que vamos a guardar el modelo. Siempre hay que guardar el modelo en un objeto, si no, no vamos a poder hacer cálculos posteriores con él.
- **glmer():** el nombre de la función que lleva a cabo el análisis de regresión
- **Tipo:** nuestra variable dependiente
- **~:** este operador especifica que *Tipo* es la variable dependiente
- Después vienen los nombres de las variables independientes. Son los rótulos que aparecen en las cabeceras de las columnas de nuestro archivo de datos.
- ***:** este operador especifica que el modelo tiene que evaluar una interacción entre dos variables

- **(1 +Pr.2.Pr + Co.2.Pr + Tense + Negation | Nombrelema):** Esta parte describe los efectos aleatorios relacionados con el factor *Nombrelema*. Particularmente, esta parte le dice al programa que el tamaño del efecto de las variables que vienen después de *1+* tiene que variar según los niveles de la variable *Nombrelema*. La estructura es siempre *(1| nombre del intercepto aleatorio)* cuando se trata de un intercepto aleatorio o *(1+nombres variable 1 + nombre variable 2 | nombre del intercepto aleatorio)* cuando se trata de una o más pendientes aleatorias.
- **data=datos:** especifica el objeto que contiene la base de datos
- **family="binomial":** especifica que se trata de un análisis de regresión logística con dos opciones
- **control = glmerControl(optimizer="nloptwrap", calc.derivs = FALSE)):** los parámetros opcionales que se tienen que tomar en cuenta en la calculación. Hacen que los cálculos sean más rápidos. Copiar y pegar.

Cuando termine de calcularse el modelo, podemos ver los resultados mediante la función summary()

```
summary(modelo)
```

Lo siguiente es parte del output de *summary()*

```
Random effects:
 Groups     Name                                Variance Std.Dev. Corr
 Nombrelema (Intercept)                         0.33859  0.5819
            Pr.2.Pr[S.No.Priming]               0.09211  0.3035    0.80
            Pr.2.Pr[S.Pluralized]               0.31113  0.5578   -0.18 -0.71
            Co.2.Pr[S.No.Priming]               0.33215  0.5763   -0.45 -0.63  0.74
            Co.2.Pr[S.Pluralized]               0.35431  0.5952    0.74  0.29  0.46  0.24
            Tense[S.All.Others]                 0.81323  0.9018   -0.84 -0.41 -0.35 -0.08 -0.99
            Negation[S.Absent]                  0.22863  0.4782    0.54  0.93 -0.82 -0.49  0.07 -0.15
 Muestra    (Intercept)                         0.04961  0.2227
            Typical.Action.Chain.Pos[S.Heads]   0.04475  0.2115    0.86
            Pr.2.Pr[S.No.Priming]               0.13113  0.3621   -0.03 -0.24
            Pr.2.Pr[S.Pluralized]               0.04402  0.2098    0.33  0.51 -0.95
            Co.2.Pr[S.No.Priming]               0.20639  0.4543    0.94  0.87 -0.23  0.50
            Co.2.Pr[S.Pluralized]               0.36341  0.6028   -0.33 -0.21  0.69 -0.72 -0.55
            Tense[S.All.Others]                 0.12207  0.3494   -0.51 -0.81  0.60 -0.75 -0.53  0.15
            Negation[S.Absent]                  0.10652  0.3264   -0.03  0.04 -0.94  0.86  0.19 -0.84 -0.32
Number of obs: 1842, groups:  Nombrelema, 252; Muestra, 24

Fixed effects:
                                                  Estimate Std. Error z value Pr(>|z|)
(Intercept)                                       0.735504   0.190495   3.861 0.000113 ***
Typical.Action.Chain.Pos[S.Heads]                -0.598460   0.138826  -4.311 1.63e-05 ***
Pr.2.Pr[S.No.Priming]                             0.094024   0.235088   0.400 0.689194
Pr.2.Pr[S.Pluralized]                            -0.624548   0.206817  -3.020 0.002529 **
Co.2.Pr[S.No.Priming]                             0.277853   0.206916   1.343 0.179328
Co.2.Pr[S.Pluralized]                            -0.293060   0.249289  -1.176 0.239762
Age[S.Oldest.Speakers]                           -0.031604   0.104108  -0.304 0.761458
Education[S.Less]                                -0.032500   0.101067  -0.322 0.747777
Gender[S.Female]                                 -0.115930   0.100076  -1.158 0.246693
Tense[S.All.Others]                              -2.153355   0.222556  -9.676  < 2e-16 ***
Negation[S.Absent]                               -0.154216   0.154958  -0.995 0.319631
Typical.Action.Chain.Pos[S.Heads]:Tense[S.All.Others]  0.204600   0.142630   1.434 0.151437
Pr.2.Pr[S.No.Priming]:Co.2.Pr[S.No.Priming]       0.260147   0.253487   1.026 0.304762
Pr.2.Pr[S.Pluralized]:Co.2.Pr[S.No.Priming]      -0.276933   0.199371  -1.389 0.164822
Pr.2.Pr[S.No.Priming]:Co.2.Pr[S.Pluralized]       0.027922   0.278061   0.100 0.920013
Pr.2.Pr[S.Pluralized]:Co.2.Pr[S.Pluralized]       0.001098   0.240019   0.005 0.996349
Co.2.Pr[S.No.Priming]:Tense[S.All.Others]         0.340215   0.162253   2.097 0.036010 *
Co.2.Pr[S.Pluralized]:Tense[S.All.Others]        -0.011310   0.204255  -0.055 0.955842
Pr.2.Pr[S.No.Priming]:Tense[S.All.Others]        -0.375340   0.171634  -2.187 0.028752 *
Pr.2.Pr[S.Pluralized]:Tense[S.All.Others]         0.279728   0.136774   2.045 0.040836 *
Tense[S.All.Others]:Negation[S.Absent]            0.090680   0.134499   0.674 0.500183
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Arriba, bajo *Random effects* se ofrecen las varianzas y las desviaciones estándar de los efectos aleatorios. Como pueden ver, para algunos, la varianza es limitada (<0.1) , lo que indica que hay poca variación en cuanto al tamaño del efecto de estas variables entre los grupos descritos por los efectos aleatorios.

Bajo *Fixed effects* se resumen los resultados para las variables independientes y sus interacciones. Cuando un efecto es significante, aparecen asteriscos o un punto a su lado derecha. Cuando no es significante, no aparece nada.

Los coeficientes que se obtienen para los efectos fijos se expresan en Log Odds. Los Log Odds negativos indican que el nivel de la variable desfavorece la variante de aplicación. Cuando el valor es positivo, indica que lo favorece. Cero es neutro. Asimismo cuanto más se aleje el coeficiente de cero, mayor es el tamaño del efecto de la variable independiente.

Como usamos contrastes de suma, los coeficientes que se obtienen para una variable independiente siempre suman cero. Por lo tanto, cuando una variable independiente sólo tiene dos niveles, el coeficiente del segundo nivel se puede obtener multiplicando por -1 el coeficiente que aparece en el *summary()*. En cambio, cuando una variable

independiente tiene tres o más niveles, el coeficiente que no aparece en el *summary()* se puede obtener restando de cero la suma de los coeficientes que sí aparecen. Así, para la variable *Pr.2.Pr* (coeficientes: 0.094024, -0.624548), sería:

```
0-sum(0.094024, -0.624548)
```

Lo cual produce:

```
[1] 0.530524
```

Ahora tenemos unos estimados de los efectos de cada una de las variables. Para evaluar fiabilidad de este modelo vamos a usar mi función *sum_stats()*.

```
stats.modelo<-sum_stats(modelo)
```

Esta función produce un *summary()* del modelo y le añade algunas estadísticas sumarias. Las más importantes son:

- $c.full: el índice C expresa cuán adecuadas son las predicciones que formula el modelo. C> 0.9 indica que el modelo hace excelentes predicciones. C>0.8 indica que el modelo hace predicciones fiables, 0.7< C >0.8 indica que las predicciones son aceptables. Cuanto más alto el índice C, mejor.
- $c.fixed: esta estadística expresa cuán adecuadas son las predicciones que formula un modelo más simple, que sólo incluye las variables independientes, sin las variables aleatorias. Cuanto más alto el índice C, mejor.
- $rsquare.full: el R cuadrado indica la proporción de la variación que modela nuestro modelo. Cuanto más alto el R cuadrado, mejor.
- $rsquare.fixed: el R cuadrado indica la proporción de la variación que modelan las variables independientes, sin tomar en cuenta las variables aleatorias. Cuanto más alto el R cuadrado, mejor.
- $vif.fixed.only: Los *variation inflation factors*. GVIF > 5 indica que los efectos de las variables no son independientes (i.e., son colineales) y que habrá que eliminar una de las variables que tienen el GVIF > 5.

Para ver todas las estadísticas, tecleamos:

```
stats.modelo
```

Para ver solamente una de estas estadísticas, tecleamos, por ejemplo

```
stats.modelo$c.full
```

4) Cuarto paso: seleccionar un modelo parsimonioso

Como muestra la fórmula, nuestro modelo es muy complejo: incluye varias pendientes aleatorias y un buen número de interacciones. Cuando consideramos el *summary()*, vemos que varios de estos factores e interacciones no son significantes. Ya que el propósito de un análisis de regresión consiste en identificar los factores que tienen un impacto decisivo en la variación, hay que eliminar las variables superfluas.

A este proceso se le llama 'selección de modelos'. Se considera que la mejor práctica consiste en empezar con el modelo más complicado que podamos formular para luego ir eliminando variables independientes y efectos aleatorios hasta obtener un modelo que ofrezca un buen equilibrio entre, por un lado, su complejidad y, por lado, su ajuste a los datos (Bates, Kliegl, Vasishth, & Baayen, 2015). [1]

La estadística la que usamos para comparar modelos es el *Criterio de Información de Akaike con corrección para muestras pequeñas* (AICc). Este criterio expresa el balance entre la complejidad del modelo y su calidad de ajuste. Si introducimos un parámetro en el modelo que no contribuye información útil para modelar nuestra variable, sube el valor del AICc. Al revés, cuando eliminamos una variable independiente que no contribuye información pertinente para modelar la variable dependiente, baja el AICc. Los valores más bajos indican un mejor equilibrio entre la complejidad del modelo y su ajuste.

Para seleccionar un modelo, vamos a usar mi función *prune.model()*. Esta función empieza por eliminar las pendientes aleatorias una por una y luego elimina las variables independientes una por una. Cuando elimina un parámetro, vuelve a calcular el modelo y luego compara el valor AICc de ese nuevo modelo con el modelo máximo u otro modelo mejor. Ello suele tomar mucho tiempo, pues se calculan y se comparan tantos modelos cuantos parámetros hay.

```
modelo.parsimonioso<-prune.model(modelo)
```

La función produce una lista que incluye el modelo final (modelo.final$model) y una lista de las variables que se eliminaron (modelo.final$dropped.terms). Para visualizar el modelo y/o evaluar su calidad se necesita este código:

```
stats.modelo.parsimonioso<-sum_stats(modelo.parsimonioso$model)
```

[1] Puede que el modelo máximo no converja (i.e., que sea demasiado complejo para la cantidad de datos de que disponemos). En ese caso, habrá que eliminar una o más pendientes aleatorias (las menos probables) para que la función produzca resultados.

5) Quinto paso: Validar el modelo mediante bootstrapping

Gracias a la función prune.model() ahora tenemos un modelo parsimonioso. Pero, todavía no sabemos si los resultados que arroja ese modelo se deben a unas tendencias que sólo se observan en nuestra base de datos o si reflejan unas tendencias reales que se observan a nivel de la población. También nos falta verificar si el modelo no está sobreajustado (i.e., no incluye demasiadas variables en comparación con el número de observaciones). Por eso tenemos que validar nuestro modelo.

Una manera de validar el modelo sería recoger otra muestra nueva y volver a calcular el modelo para ver si los resultados siguen siendo iguales. Como ello no resulta factible en la mayoría de los casos, se suele confeccionar un gran número de muestras nuevas recombinando porciones de la base de datos original (en nuestro caso: 1000). A eso se le llama *bootstrapping*. Para cada una de esas submuestras, se vuelve a calcular el modelo.

Comparando los resultados obtenidos mediante esas 1000 submuestras, se obtienen intervalos de confianza. Los intervalos de confianza expresan la región numérica para la cual se estima con un 95% de probabilidad que incluirá el valor del coeficiente para la media de la población. Si los coeficientes de nuestro modelo no se generalizan a la población, la distancia entre el umbral de confianza del 2.5% y el umbral de confianza del 97.5% será muy amplia (1 Log Odds o más). En ese caso, puede que el modelo incluya demasiados factores (i.e., esté sobreajustado) o que los factores sean colineales. Sea como fuera, en ese caso habrá que verificar a qué se deben los intervalos amplios y habrá que arreglarlo.

Para calcular intervalos de confianza mediante el método *bootstrap* (toma mucho tiempo, háganlo de noche) usamos la función *confint()* del paquete *lme4*:

```
intervalos<-confint(modelo.parsimonioso$model, method="boot", boot.type = "perc", nsim = 1000, parallel="snow", ncpus=8)
```

Para ver los intervalos, tecleen:

```
intervalos
```

```
  (converted from warning) some bootstrap runs failed (1/500)
> intervalos
                                                                2.5 %       97.5 %
.sig01                                                      0.32045509  1.731114227
.sig02                                                     -0.99992448  0.171719952
.sig03                                                      0.42472654  2.022805279
.sig04                                                      0.35663649  0.920094983
.sig05                                                     -1.00000000  0.282816971
.sig06                                                      0.03721259  0.670187638
(Intercept)                                                -1.93613789 -0.351126068
Typical.Action.Chain.Pos[S.Heads]                           0.32098747  1.016852928
Tense[S.All.Others]                                         1.65469638  3.264110310
Negation[S.Absent]                                          0.01514188  0.622676672
Co.2.Pr[S.No.Priming]                                      -0.61554536  0.015783778
Co.2.Pr[S.Pluralized]                                       0.06178750  0.933500789
Pr.2.Pr[S.No.Priming]                                      -0.80009506  0.038336198
Pr.2.Pr[S.Pluralized]                                       0.61195241  1.141599007
Typical.Action.Chain.Pos[S.Heads]:Tense[S.All.Others]      -0.81888496  0.003397187
Tense[S.All.Others]:Co.2.Pr[S.No.Priming]                  -0.52404666  0.114862274
Tense[S.All.Others]:Co.2.Pr[S.Pluralized]                  -0.43528797  0.471675764
Tense[S.All.Others]:Pr.2.Pr[S.No.Priming]                  -0.04032232  0.762917925
Tense[S.All.Others]:Pr.2.Pr[S.Pluralized]                  -0.56197072 -0.082209918
Tense[S.All.Others]:Negation[S.Absent]                     -0.52528292  0.085302904
>
```

Como se deduce de la figura, los intervalos que obtenemos son muy amplios. Como los GVIF < 5, lo más probable es que el modelo esté sobreajustado. Eso sugiere que hay que eliminar una o más variables independientes.

Siempre cuando hay que eliminar variables independientes, hay que mantenerse a este orden y después de cada paso hay que volver a calcular los intervalos de confianza:

1. Eliminar las pendientes aleatorias una por una (primero la pendiente con el mayor número de niveles)
2. Eliminar las interacciones una por una
3. Eliminar las variables independientes una por una
4. Volver a introducir las pendientes aleatorias (con tal de que no se relacionen con variables eliminadas) y verificar si los datos dan para evaluarlas. Si no, sacar la pendiente con el mayor número de niveles y verificar si está resuelto el problema.
5. Volver a insertar las variables independientes (con tal de que no sean interacciones) que eliminó *prune.model()*. Verificar si los datos dan para evaluar esas variables adicionales sin causar sobreajustes.
6. Volver a aplicar la función *prune.model()* para verificar si se deben guardar las variables.
7. Validar ese modelo nuevo mediante intervalos de confianza.

En nuestro caso, vamos a empezar por eliminar la pendiente aleatoria *Tense/Nombrelema*, porque ésta tiene el mayor número de niveles.

```
modelo2<- glmer(Tipo ~ Typical.Action.Chain.Pos  + Pr.2.Pr + Co.2.Pr + Age + Education+ Gender + Tense   + Negation + Typical.Action.Chain.Pos * Tense + Pr.2.Pr*Co.2.Pr + Tense * Co.2.Pr + Tense * Pr.2.Pr + Tense * Negation + (1+Tense | Muestra)+ (1  | Nombrelema), data=datos, family="binomial", control=glmerControl(optimizer="nloptwrap", calc.derivs=F))
```

Volvemos a calcular los intervalos de confianza para averiguar si la eliminación de la pendiente aleatoria solucionó el problema.

```
intervalos2<-confint(modelo2, method="boot", boot.type = "perc", nsim = 1000, parallel="snow", ncpus=8)
```

Después de varios pasos intermedios, finalmente resulta que no disponemos de suficientes datos como para evaluar el efecto de las interacciones y la pendiente aleatoria Tense/Nombrelema. Después de sacar las interacciones, volvimos a introducir la pendiente aleatoria Tense/Muestra. Los intervalos de confianza muestran que nuestros datos dan para evaluar ésta, por lo que la guardamos por ahora.

Después volvimos a introducir las variables sociales, que se eliminaron en la primera aplicación de *prune.model()*. Los intervalos de confianza no sugirieron sobreajustes. Finalmente, volvimos a aplicar la función *prune.model(),* la cual eliminó todas las variables sociales (salvo el género), la negación y la pendiente aleatoria Tense/Muestra.

```
#El modelo

modelo4<- glmer(Tipo ~ Typical.Action.Chain.Pos  + Pr.2.Pr + Co.2.Pr + Age + Education+ Gender + Tense   + Negation + (1+Tense | Muestra)+ (1  | Nombrelema), data=datos, family="binomial", control=glmerControl(optimizer="nloptwrap", calc.derivs=F))

#Los intervalos

intervalos4<-confint(modelo4, method="boot", boot.type = "perc", nsim = 1000, parallel="snow", ncpus=8)

#Eliminar variables superfluas

modelo.final <-prune.model(modelo4)
```

```
#Validar el modelo.final

intervalos5<-confint(modelo.final$model, method="boot", boot.type = "perc", nsim = 1000, parallel="snow", ncpus=8)

#Ver los resultados

summary(modelo.final$model)
```

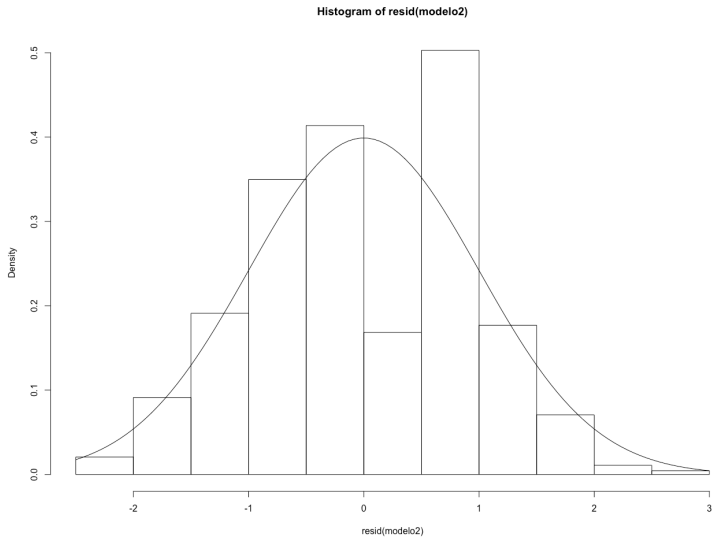
Los resultados quedan resumidos en la figura.

```
Random effects:
 Groups     Name        Variance Std.Dev.
 Nombrelema (Intercept) 0.6116   0.7821
 Muestra    (Intercept) 0.1596   0.3995
Number of obs: 1842, groups:  Nombrelema, 252; Muestra, 24

Fixed effects:
                                Estimate Std. Error z value Pr(>|z|)
(Intercept)                     -0.21746    0.14909  -1.459 0.144669
Typical.Action.Chain.Pos[S.Heads]  0.44341    0.10842   4.090 4.32e-05 ***
Pr.2.Pr[S.No.Priming]           -0.14702    0.11851  -1.241 0.214771
Pr.2.Pr[S.Pluralized]            0.79232    0.09333   8.489  < 2e-16 ***
Co.2.Pr[S.No.Priming]           -0.33793    0.11265  -3.000 0.002702 **
Co.2.Pr[S.Pluralized]            0.50652    0.14458   3.503 0.000459 ***
Gender[S.Female]                 0.14764    0.10243   1.441 0.149461
Tense[S.All.Others]              1.43115    0.08614  16.615  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Correlation of Fixed Effects:
            (Intr) T.A.C. P.2.P[S.N P.2.P[S.P C.2.P[S.N C.2.P[S.P G[S.F]
T.A.C.P[S.H  0.076
P.2.P[S.N.P  0.033  0.064
Pr.2.P[S.P]  0.007 -0.011 -0.657
C.2.P[S.N.P -0.386 -0.090  0.064    -0.078
C.2.Pr[S.P] -0.004  0.092 -0.047    -0.027    -0.248
Gndr[S.Fml]  0.013  0.030  0.012    -0.008     0.010    -0.034
Tns[S.Al.O] -0.093  0.146  0.086     0.016    -0.121     0.025     0.034
```

Finalmente, tenemos que evaluar los residuos de nuestro modelo (la diferencia entre los valores observados y los valores que calcula la función de regresión). El histograma de los residuos debe mostrar una **distribución simétrica en forma de campana**, lo que indica que es probable que sea cierta la hipótesis de normalidad. El código siguiente produce un histograma de los residuos:

```
histograma<-hist(resid(modelo.final$model), freq=F);curve(dnorm, add = TRUE)
```

Como podemos ver los residuos de nuestro modelo final tienen una distribución más o menos normal.



6) Cómo presentar los resultados

Los resultados deben resumirse en sola tabla. En esa tabla no deben aparecer los intervalos de confianza, pero sí deben aparecer:

1) Los números absolutos y los porcentajes

2) Los coeficientes para todos los niveles de todas las variables, redondeados a tres dígitos, así como las varianzas y las desviaciones estándar de los interceptos aleatorios

3) Las estadísticas sumarias: el índice C, el R cuadrado y el AICc para el modelo completo (stats.modelo.final$c.full, stats.modelo.final$r.square.full).

```
stats.modelo.final<-sum_stats(modelo.final$model)
```

El AICc se obtiene de la manera siguiente:

```
AICc(modelo.final$model)
```

He aquí un ejemplo de una tabla de regresión.

Table 9: Logistic generalized linear mixed-effects model of agreement with *haber* in plural existentials, in Puerto Rican Spanish (sum contrasts)

Fixed effects [a]	*N*	*%*	*Coefficient (log-odds)*
(intercept)			-0.974
Verb form			
All others	622/1014	61.3	1.766
Synthetic expressions in present or preterit tense (*hay/hayn* or *hubo/hubieron)*	62/641	9.7	-1.766
Production-to-production priming			
Agreeing *haber*	352/558	63.1	0.597
First occurrence/distance 20+ clauses	88/246	35.8	-0.155
Non-agreeing *haber*	244/851	28.7	-0.442
Comprehension-to-production priming			
Agreeing *haber*	92/175	52.6	0.452
First occurrence/distance 20+ clauses	562/1355	41.5	-0.186
Non-agreeing *haber*	30/125	24.0	-0.266
Typical action-chain position of the noun's referent			
Heads (typical agents)	350/773	45.3	0.418
Other (typical patients and settings)	348/882	37.9	-0.418
Polarity			
Positive	559/1225	45.6	0.341
Negative	125/430	29.1	-0.341
Random intercepts		*Variance*	*Standard Deviation*
Noun		0.378	0.651
Speaker		0.334	0.578
Model summary		*Linguistic fixed*	*Full*
Pseudo-R^2		0.43	0.62 [b]
C-index		0.83	0.89
AIC_c		1626.38	1517.9

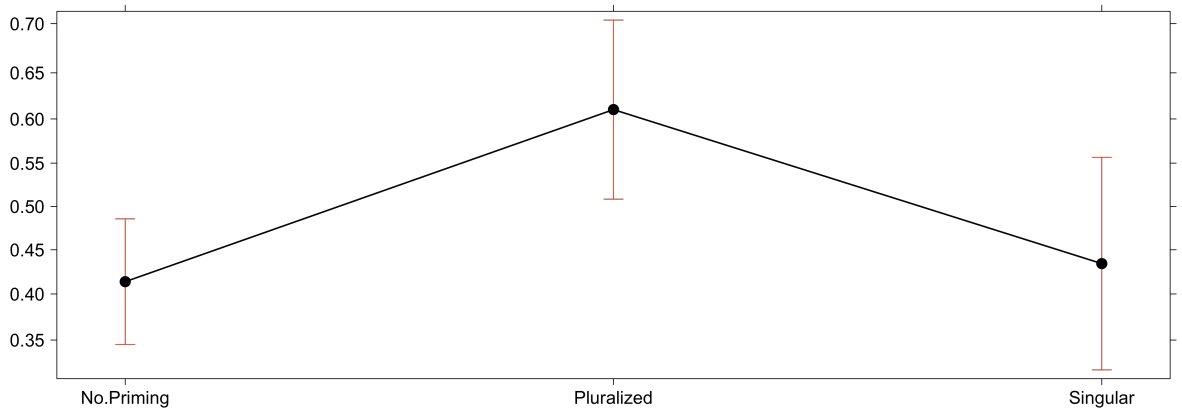
Notes: [a] Also included in the final model: Educational achievement and gender. See Author1 (submitted: Chap. 8). [b] We report the conditional pseudo-R^2 for our mixed-effects models, meaning the R^2 measure takes into account the amount of variance that is explained by the fixed and the random effects.

Además, sus tesinas deben incluir un anejo electrónico donde aparecen los datos, todos los modelos, los intervalos de confianza y el histograma de los residuos.

Durante la discusión, puede resultar útil y cómodo para el lector tener una representación gráfica de los resultados para que no se tenga que volver a las tablas todo el tiempo. Para hacerlo, pueden usar mi función *effectPlot()*. Esta función produce una imagen de alta resolución que representa el efecto de las variables independientes en la escala de probabilidad. Para usarla, usamos el código siguiente:

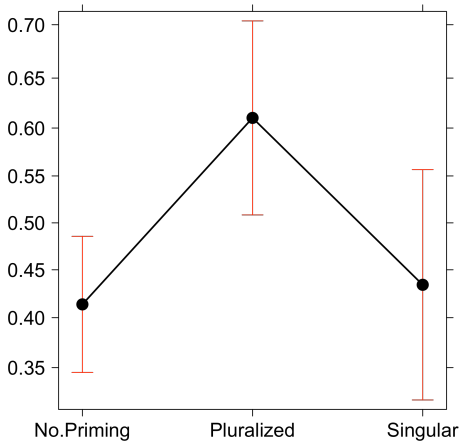
```
#Por alguna razón, la función no encuentra el modelo si no lo creamos directamente, sin la función prune.model()

el.modelo.final<- glmer(Tipo ~ Typical.Action.Chain.Pos + Pr.2.Pr + Co.2.Pr + Gender + Tense + (1 | Muestra)+ (1 | Nombrelema), data=datos, family="binomial", control=glmerControl(optimizer="nloptwrap", calc.derivs=F))

#fíjense en el uso de las comillas

effectPlot("Co.2.Pr", el.modelo.final)

#busquen plot.png en el escritorio
```

Produce la figura siguiente:



Para cambiar las dimensiones de la figura, podemos añadir los argumentos *width* y *height*, que expresan el tamaño de la imagen, en centímetros.

```
effectPlot("Co.2.Pr", el.modelo.final.final, width=15, height=15)
```



7) Guardar nuestro trabajo

Como varios de los cálculos que se tienen que hacer para llevar a cabo un buen análisis de regresión toman mucho tiempo, no los podemos perder al cerrar R. Para guardar esos objetos en formato Rdata se usa:

```
save(datos, modelo, stats.modelo, stats.modelo.parsimonioso, modelo.parsimonioso, intervalos, modelo2, intervalos2, modelo4, intervalos4, modelo.final, intervalos5, el.modelo.final, stats.modelo.final, histograma, file= "resultados.rdata")
```

Para volver a cargar esos objetos, podemos usar:

```
load(file.choose())
```