

Regressão ordinal - uma introdução

Igor Costa

10/05/2022

Motivação

Esse material pretende fazer uma breve introdução aos chamados modelos de regressão ordinais, ajustados a variáveis dependentes categorias ordinais, ou seja, em que a ordem dos elementos é importante (nível de escolaridade, respostas em escala do tipo Likert, etc). As discussões aqui feitas restringem-se ao campo da psicolinguística experimental.

Recomendações de leitura:

Gelman & Hill (2009). *Data Analysis Using Regression and Multilevel/Hierarchical Models*.

Mais especificamente:

- 6.5 Multinomial regression (119-123)
- 15.2 Ordered categorical regression: storable votes (331-332).
- 5 Logistic regression (p. 79-105).

Veja também:

- Nicenboim, B., Schad, D. & Vasishth, S. An Introduction to Bayesian Data Analysis for Cognitive Science.
- Cap. 23 - Ordinal Predicted Variable. In: Kurz, S. Doing Bayesian Data Analysis in brms and the tidyverse.

Carregando os pacotes adequados

```
require(ggplot2)
require(dplyr)
require(RColorBrewer)
require(tidyr)
require(scales)
require(brms) # Esse o pacote para fazermos o modelo ordinal bayesiano
require(sjPlot) # Fundamental para extrair os efeitos marginais (última etapa deste tutorial)
```

Carregamento e organização dos dados brutos

Carregando os dados de um experimento com Escala Likert:

```
dados <-
  read.csv("https://raw.githubusercontent.com/igordeo-costa/ModelosOrdinais/main/ScalaLikertCosta2022.csv")
```

Transformando as variáveis de modo adequado

```
dados$Ordem<-as.factor(dados$Ordem)
dados$Num<-as.factor(dados$Num)
```

```

dados$answer<-as.factor(dados$answer)
dados$idade<-as.integer(dados$idade)
dados$genero<-as.factor(dados$genero)
dados$escolaridade<-as.factor(dados$escolaridade)
dados$idioma<-as.factor(dados$idioma)
dados$item<-as.factor(dados$item)

```

Mudar os dados respostas para ordenados

```

dados$answer<-as.ordered(dados$answer)

```

Investigando os dados - Análise descritiva

Contagem dos valores brutos:

```

contag <- dados %>%
  group_by(Ordem, Num, answer) %>%
  tally() %>%
  group_by(Ordem, Num) %>%
  spread(answer, n)

colnames(contag) <- c("Ordem", "Num",
                     "Discordo_Totalmente", "Discordo", "Neutro",
                     "Concordo", "Concordo_Totalmente")

contag

```

```

## # A tibble: 4 x 7
## # Groups:   Ordem, Num [4]
##   Ordem  Num  Discordo_Totalmente Discordo Neutro Concordo Concordo_Totalmente
##   <fct> <fct>                <int>   <int> <int>   <int>         <int>
## 1 todo-um PL             156       72   20     27         23
## 2 todo-um SG              27       25   40     53        149
## 3 um-todo PL              66       26   42     66        108
## 4 um-todo SG              20       12   33     53        181

```

Cálculo das porcentagens respectivas (tabela em formato horizontal)

```

porc_horiz <- dados %>%
  group_by(Ordem, Num, answer) %>%
  tally() %>%
  mutate(perc=n/sum(n)) %>%
  dplyr::select(-n) %>%
  group_by(Ordem, Num) %>%
  spread(answer, perc)

colnames(porc_horiz) <- c("Ordem", "Num",
                        "Discordo_Totalmente", "Discordo", "Neutro",
                        "Concordo", "Concordo_Totalmente")

porc_horiz

```

```

## # A tibble: 4 x 7
## # Groups:   Ordem, Num [4]
##   Ordem  Num  Discordo_Totalmente Discordo Neutro Concordo Concordo_Totalmente
##   <fct> <fct>                <dbl>   <dbl> <dbl>   <dbl>         <dbl>

```

## 1 todo-um PL	0.523	0.242	0.0671	0.0906	0.0772
## 2 todo-um SG	0.0918	0.0850	0.136	0.180	0.507
## 3 um-todo PL	0.214	0.0844	0.136	0.214	0.351
## 4 um-todo SG	0.0669	0.0401	0.110	0.177	0.605

Gráfico de barras empilhadas para fácil visualização

Essa parte é difícil de compreender se não temos a noção do resultado final. Você pode rodá-la e voltar depois com calma para entender os códigos. Resumidamente: vamos fazer um gráfico de barras empilhadas centrado na categoria “neutro”, ou seja, no meio da escala. Para isso, vamos criar dois conjuntos de dados: as porcentagens da parte de cima (“concordo” e “concordo totalmente”) e as da parte de baixo (“discordo” e “discordo totalmente”). A categoria “neutro” será dividida em duas e cada pedaço anexado a uma das partes acima.

Dividir o meio da escala (os julgamentos “Neutro”):

```
dados_meio <- porc_horiz %>%
  mutate(c1 = Neutro / 2,
         c2 = Neutro / 2) %>%
  select(Ordem, Num,
         Discordo_Totalmente, Discordo, c1, c2, Concordo, Concordo_Totalmente) %>%
  gather(key = answer, value = perc, 3:8)
```

Separar a escala em dois conjuntos, o “alto” e o “baixo”:

```
meio_alto <- dados_meio %>%
  filter(answer %in% c("Concordo_Totalmente", "Concordo", "c2")) %>%
  # Níveis na ordem normal!
  mutate(answer = factor(answer, levels = c("Concordo_Totalmente", "Concordo", "c2")))

meio_baixo <- dados_meio %>%
  filter(answer %in% c("c1", "Discordo", "Discordo_Totalmente")) %>%
  # Níveis na ordem inversa!
  mutate(answer = factor(answer, levels = c("Discordo_Totalmente", "Discordo", "c1")))
```

Estabelecer uma paleta de cores para organização!

```
# Usar, do pacote RColorBrewer, a paleta de cores "spectral", com 5 cores
legend_pal <- brewer.pal(name = "Spectral", n = 5)
# Duplica a cor do meio manualmente
legend_pal <- c("#2B83BA", "#ABDDA4", "#FFFFBF", "#FFFFBF", "#FDAE61", "#D7191C")
# Substitui a cor do meio por um cinza
legend_pal <- gsub("#FFFFBF", "#9C9C9C", legend_pal)
# Atribui nomes às cores
names(legend_pal) <-
  c("Concordo_Totalmente", "Concordo", "c1", "c2", "Discordo", "Discordo_Totalmente")
```

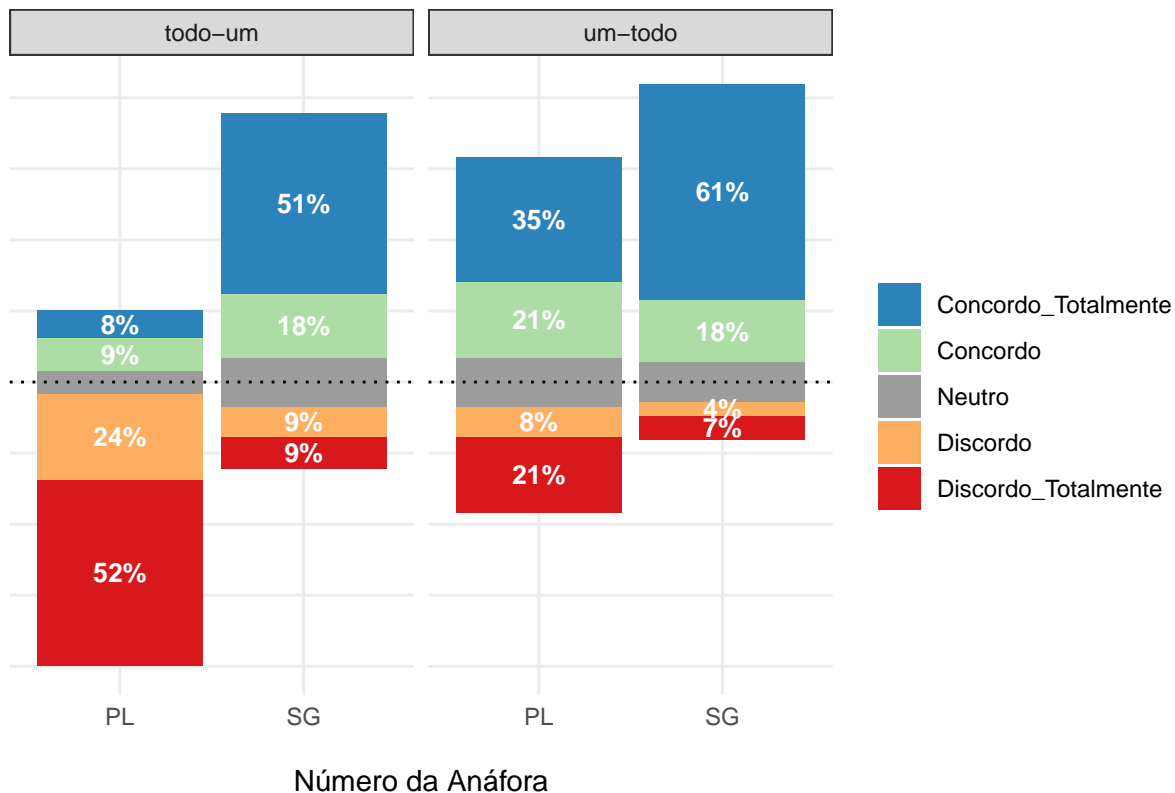
Produzir o gráfico, finalmente!

```
ggplot() +
  geom_bar(data = meio_alto, aes(x = Num, y=perc, fill = answer), stat="identity") +
  geom_bar(data = meio_baixo, aes(x = Num, y=-perc, fill = answer), stat="identity") +
  #-----
  # Essa parte do código é um pouco complicada...
  # Serve apenas para colocar os valores dentro das barras de modo alinhado...
  geom_text(data = meio_alto,
            aes(alpha = answer, x = Num, y=perc, group = answer,
```

```

        label = scales::percent(perc, accuracy = 1)),
        color = "white", size = 3.5, fontface = "bold",
        position = position_stack(vjust = .5)) +
geom_text(data = meio_baixo,
        aes(alpha = answer, x = Num, y=-perc, group = answer,
            label = scales::percent(perc, accuracy = 1)),
        size = 3.5, color = "white", fontface = "bold",
        position = position_stack(vjust = .5),) +
scale_alpha_manual(values = c("c1" = 0, "c2" = 0,
        "Discordo" = 1, "Discordo_Totalmente" = 1,
        "Concordo_Totalmente" = 1, "Concordo" = 1),
        guide = 'none') +
#-----
geom_hline(yintercept = 0, color =c("black"), linetype = "dotted") +
facet_wrap(~Ordem) +
scale_y_continuous(breaks = seq(from = -1, to = 1, by = .2),
        # Definir os valores negativos da legenda como positivos
        labels = function(x) percent(abs(x))) +
scale_fill_manual(values = legend_pal,
        breaks = c("Concordo_Totalmente", "Concordo", "c2",
            "Discordo", "Discordo_Totalmente"),
        labels = c("Concordo_Totalmente", "Concordo", "Neutro",
            "Discordo", "Discordo_Totalmente")) +
labs(x = "\nNúmero da Anáfora", y = "", fill = "") +
ggtitle("") +
theme_bw() +
theme(axis.ticks.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank(),
        plot.background = element_blank(),
        panel.grid.minor = element_blank(),
        panel.border = element_blank())

```



Ajustando um modelo ordinal misto

Agora vamos de fato ajustar um modelo ordinal bayesiano...

Verificando quais os contrastes dos dados:

```
contrasts(dados$Num)
```

```
##      SG
## PL    0
## SG    1
```

```
contrasts(dados$Ordem)
```

```
##          um-todo
## todo-um         0
## um-todo         1
```

Mudar o nível-base para “um-todo” para facilitar a comparação entre as condições “um-todo” SG x PL

```
dados$Ordem<-relevel(dados$Ordem, ref = "um-todo")
```

Ajustando um modelo de regressão ordinal bayesiano sem definição das priors informativas (“flat priors”). Este não é o melhor cenário, mas o objetivo aqui é apenas entender o funcionamento e implementação do modelo.

Vamos rodar o modelo mais completo, pois demora um pouco... ou muito, dependendo da capacidade de processamento do seu computador...

```
# Vamos garantir que o programa use toda a nossa capacidade de computação para acelerar o
# processo usando todos os núcleos do nosso processador
options(mc.cores = parallel::detectCores())
```

```
# Aqui de fato o modelo
```

```
m2 <- brm(answer ~ Ordem*Num + (1+Ordem*Num|subj)+(1+Ordem*Num|item),  
  data = dados,  
  family = cumulative(link = "logit", threshold = "flexible"),  
  silent = 2, refresh = 0) # Apenas para não dar avisos no arquivo .pdf
```

```
## Running /usr/lib/R/bin/R CMD SHLIB foo.c  
## gcc -I"/usr/share/R/include" -DNDEBUG -I"/home/igor/R/x86_64-pc-linux-gnu-library/4.1/Rcpp/include/  
## In file included from /home/igor/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/Core:88,  
## from /home/igor/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/Dense:1,  
## from /home/igor/R/x86_64-pc-linux-gnu-library/4.1/StanHeaders/include/stan/math/prin  
## from <command-line>:  
## /home/igor/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: e  
## 628 | namespace Eigen {  
## | ~~~~~  
## /home/igor/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:17: c  
## 628 | namespace Eigen {  
## | ~~~~~  
## In file included from /home/igor/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/Dense:1,  
## from /home/igor/R/x86_64-pc-linux-gnu-library/4.1/StanHeaders/include/stan/math/prin  
## from <command-line>:  
## /home/igor/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/Core:96:10: fatal error: comple  
## 96 | #include <complex>  
## | ~~~~~  
## compilation terminated.  
## make: *** [/usr/lib/R/etc/Makeconf:168: foo.o] Erro 1
```

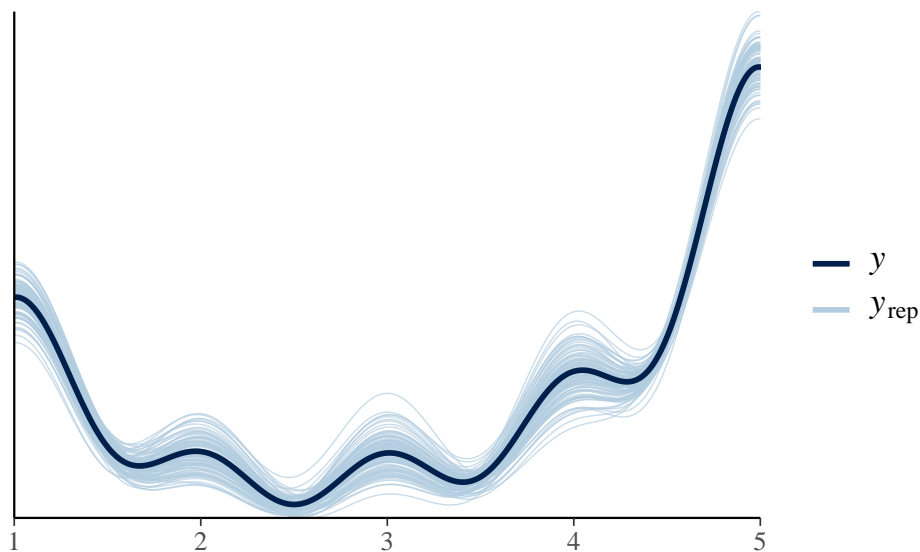
```
# Investigando apenas os fatores fixos
```

```
fixef(m2)[5:7,]
```

```
## Estimate Est.Error Q2.5 Q97.5  
## OrdemtodoMum -2.474556 0.3606946 -3.2144680 -1.770054  
## NumSG 1.613760 0.4158148 0.8302595 2.461932  
## OrdemtodoMum:NumSG 1.671950 0.4210663 0.8490518 2.520467
```

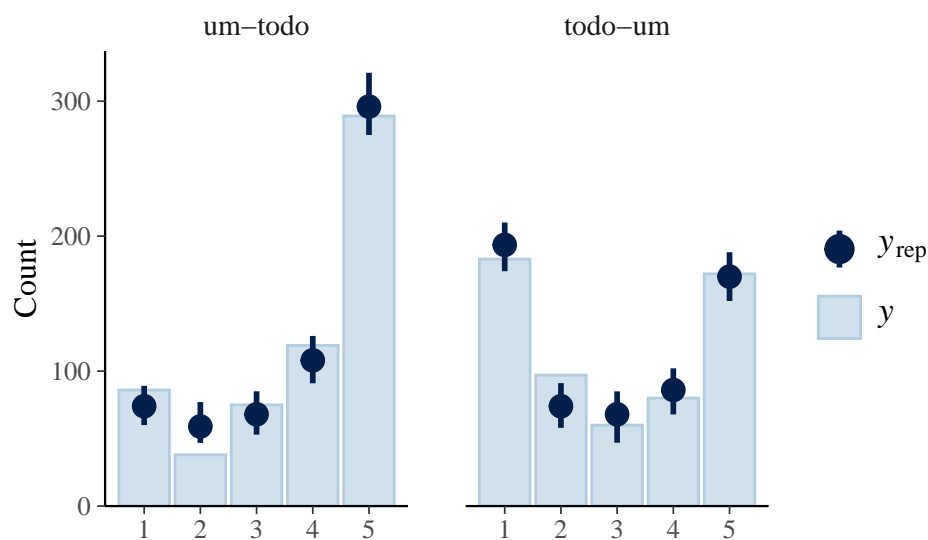
Você pode checar o ajuste do modelo, ou seja, a relação entre as priors e os dados. Neste caso, porém, ele será bom, visto que, com ‘flat priors’ o modelo usava os próprios dados como critério...

```
pp_check(m2, ndraws = 100)
```



```
# Investigue também no formato de barras, se desejar
# pp_check(m2, type = "bars", ndraws = 100)

# Você pode investigar, inclusive, por grupo:
pp_check(m2, type = "bars_grouped", ndraws = 100, group = "Ordem")
```



Visualizando as estimativas

Extraindo os coeficientes dos fatores fixos e intervalos de confiança calculados

```
fixos.m<-fixef(m2)[5:7,]

# Prepara um data frame com esses dados
colnames(fixos.m)<-c("Estimativas", "Est.Error", "lower", "upper")
fixos.m<-as.data.frame(fixos.m)

# Calcula da chance e das probabilidades correspondentes
fixos.m<-fixos.m %>%
  mutate(OddsRatio=exp(Estimativas)) %>%
  mutate(Probs=OddsRatio/(1 + OddsRatio)*100)
```

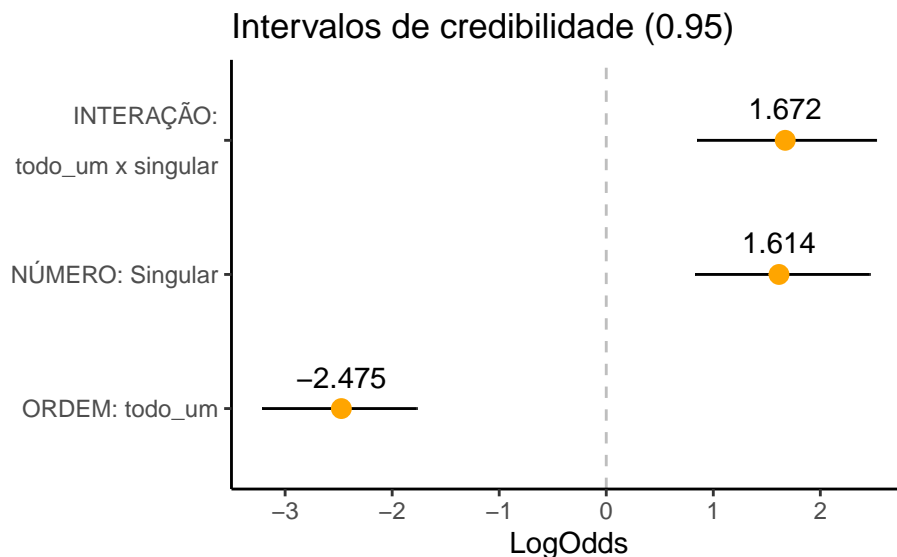
```
rownames(fixos.m)<-c("ORDEM: todo_um", "NÚMERO: singular", "INTERAÇÃO: ordem x número")

round(fixos.m, 3)
```

```
##               Estimativas Est.Error  lower  upper OddsRatio  Probs
## ORDEM: todo_um        -2.475    0.361 -3.214 -1.770    0.084  7.766
## NÚMERO: singular         1.614    0.416  0.830  2.462    5.022 83.393
## INTERAÇÃO: ordem x número    1.672    0.421  0.849  2.520    5.323 84.184
```

Agora vamos plotar o gráfico para publicação...

```
fixos.m %>%
  ggplot(aes(x=reorder(rownames(fixos.m), Estimativas),
              y=Estimativas))+
  geom_hline(yintercept = 0, linetype = "dashed", color = "grey")+
  geom_errorbar(aes(ymin=lower, ymax=upper), width=.0,
               position=position_dodge(.9))+
  geom_point(color="orange", size = 3)+
  geom_text(aes(label=round(Estimativas, 3)), vjust=-1)+
  # ao mudar a ordem é preciso mudar aqui embaixo...
  scale_x_discrete(labels=c("ORDEM: todo_um", "NÚMERO: Singular",
                            "INTERAÇÃO:\n\ntodo_um x singular")) +
  labs(y = "LogOdds", x = "") +
  ggtitle("Intervalos de credibilidade (0.95)") +
  coord_flip()+theme_classic()
```



Uma outra forma de pensar sobre esse gráfico é olhar para a relação entre a distribuição das estimativas dos fatores fixos¹ e os interceptos (tau-cuts τ) estimados pelo modelo:

```
tibble(
  fatores = c("Todo_um", "Singular", "Interação"),
  mu = fixef(m2)[5:7, 1], # Estimativas do modelo
  ci = 1.96*fixef(m2)[5:7, 2]) %>% # Erro padrão x 1.96 (ci = .95)
  expand(nesting(fatores, mu, ci),
        y = seq(from = -5, to = 5, by = .01)) %>%
```

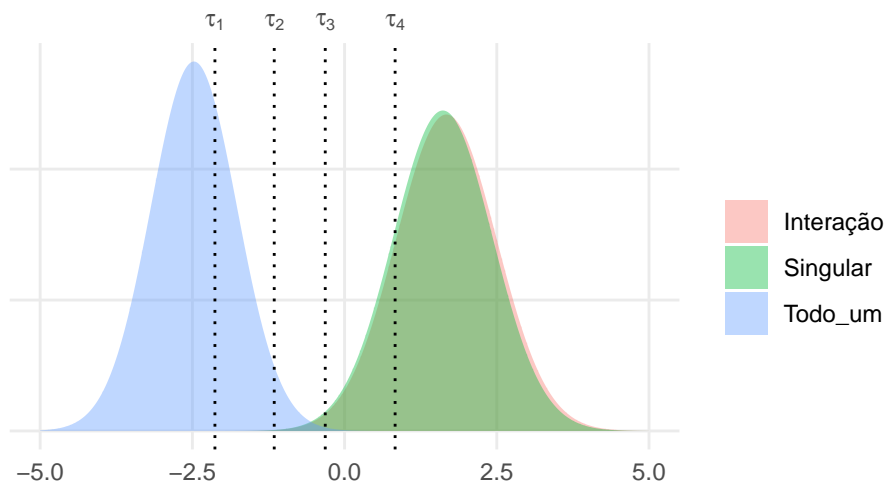
¹Modelos bayesianos apresentam uma série de estimativas para os valores de sigma, de modo que aqui, apenas a título de ilustração, usamos o intervalo de confiança calculado a partir do Erro padrão como a variância da curva.


```

mutate(d = dnorm(y, mu, ci)) %>%
ggplot(aes(x = y, y = d, fill = fatores)) +
geom_area(alpha = 0.4, position = "identity") +
geom_vline(xintercept = fixef(m2)[1:4, 1], linetype = "dotted") +
scale_x_continuous(sec.axis = dup_axis(breaks = fixef(m2)[1:4, 1] %>%
as.double(),
labels = parse(text = stringr::str_c("tau[", 1:4, "]")))) +

labs(y = "", x = "", fill = "") +
theme_bw() +
theme(axis.ticks.y = element_blank(),
axis.ticks.x = element_blank(),
axis.text.y = element_blank(),
plot.background = element_blank(),
panel.grid.minor = element_blank(),
panel.border = element_blank())

```



Interpretando os resultados...

A fim de entender os resultados, primeiro devemos lembrar os contrastes do modelo. Neste caso, estamos usando ‘dummy code’ (0 e 1), também chamado de ‘treatment contrasts’. Não vamos entrar em detalhes aqui sobre outros tipos de contrastes. Para uma discussão introdutória, veja o excelente material disponível aqui.

```
contrasts(dados$Num)
```

```
##      SG
## PL    0
## SG    1
```

```
contrasts(dados$Ordem)
```

```
##      todo-um
## um-todo      0
## todo-um      1
```

As categorias codificadas com 0 (zero) formam a base, a referência contra as quais as demais serão contrastadas. Neste caso, um_todo (0) PL (0) é a categoria de referência. Logo, podemos entender assim as estimativas do modelo:

- O preditor ‘Ordem: todo_um’ = -2.47 logOdds indica o contraste ‘todo_um (1) PL (0)’ × ‘um_todo (0) PL (0)’. Ou seja, mantendo-se o número constante, o efeito da ‘ordem’: todo_um aumenta a

probabilidade de se DESCER na escala, aumenta a probabilidade de se dar uma resposta mais abaixo na escala.

- O preditor 'Número: singular' = +1.62 logOdds indica o contraste 'um_todo (0) SG (1)' x 'um_todo (0) PL (0)'. Ou seja, mantendo-se a ordem constante, o efeito do 'número': singular aumenta a probabilidade de se SUBIR na escala, aumenta a probabilidade de se dar uma resposta mais acima na escala.
- A interação 'Ordem x Número (todo_um singular)' = +1.66 logOdds indica que os níveis da variável 'Ordem' interagem com os níveis da variável 'Número', basicamente indicando que: o nível singular aumenta a probabilidade de SUBIR na escala com todo_um; o nível plural aumenta a probabilidade de DESCER na escala com todo_um.

IMPORTANTE! O fato de haver um efeito de interação aponta para olharmos com cuidado para os efeitos de Ordem e de Número. Se não houvesse interação, o efeito de um nível da variável seria o mesmo para ambos os níveis da outra, mas como há, os efeitos de 'ordem' e de 'número' indicam contrastes particulares. Ver explicação abaixo, se quiser saber mais sobre esse ponto.

Um adendo: de logOdds para Odds para probailidades

Vamos começar com um vetor de probabilidades, de 0 a 1, portanto. E vamos transformá-lo em logaritmo da chance (LogOdds). É esse valor que o modelo com `family = 'logit'` nos dá.

```
a = round(qlogis(c(0,.1,.2,.3,.4,.5,.6,.7,.8,.9,1)), 2)
```

```
# Logaritmo da chance, ou logOdds
print(a)
```

```
## [1] -Inf -2.20 -1.39 -0.85 -0.41 0.00 0.41 0.85 1.39 2.20 Inf
```

Agora vamos transformar esse valor em chance:

```
# Odds ou chances
round(exp(a), 1)
```

```
## [1] 0.0 0.1 0.2 0.4 0.7 1.0 1.5 2.3 4.0 9.0 Inf
```

E, por fim, vamos retornar às probabilidades:

```
# Retornando às probabilidades
round(exp(a)/(1+exp(a)), 2)
```

```
## [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 NaN
```

Olhando com atenção para esses dados, o que percebemos é que:

- Probabilidade de 50% é igual a 0 (zero) logOdds que é igual a chance de 1/1 (1 para 1, ou seja, uma chance de sucesso para uma de fracasso);
- Probabilidade de 80% é igual a 1.39 logOdds que é igual a chance de 4/1 (4 chances de sucesso para uma de fracasso);
- Probabilidade de 10% é igual a -2.20 logOdds que é igual a chance de 0.1 ou 1/10 (1 chance de sucesso para 10 de fracasso).

Mas por que usar logOdds se fica tão difícil de compreender? O motivo é que probabilidade é um valor limitado entre 0 e 1 enquanto LogOdds é um valor que vai de -Inf a +Inf:

```
qlogis(0) # 0%
```

```
## [1] -Inf
```

```
qlogis(1) # 100%
```

```
## [1] Inf
```

```
qlogis(.999999999) # um valor muito próximo de 100% pode ser estimado em logOdds
```

```
## [1] 23.02585
```

```
qlogis(.000000001) # um valor muito próximo de 0% pode ser estimado em logOdds
```

```
## [1] -23.02585
```

Assim sendo, uma outra maneira de ler as estimativas do modelo é interpretar as probabilidades (ou chances) correspondentes na tabela:

```
round(fixos.m, 3)
```

##	Estimativas	Est.Error	lower	upper	OddsRatio	Probs
## ORDEM: todo_um	-2.475	0.361	-3.214	-1.770	0.084	7.766
## NÚMERO: singular	1.614	0.416	0.830	2.462	5.022	83.393
## INTERAÇÃO: ordem x número	1.672	0.421	0.849	2.520	5.323	84.184

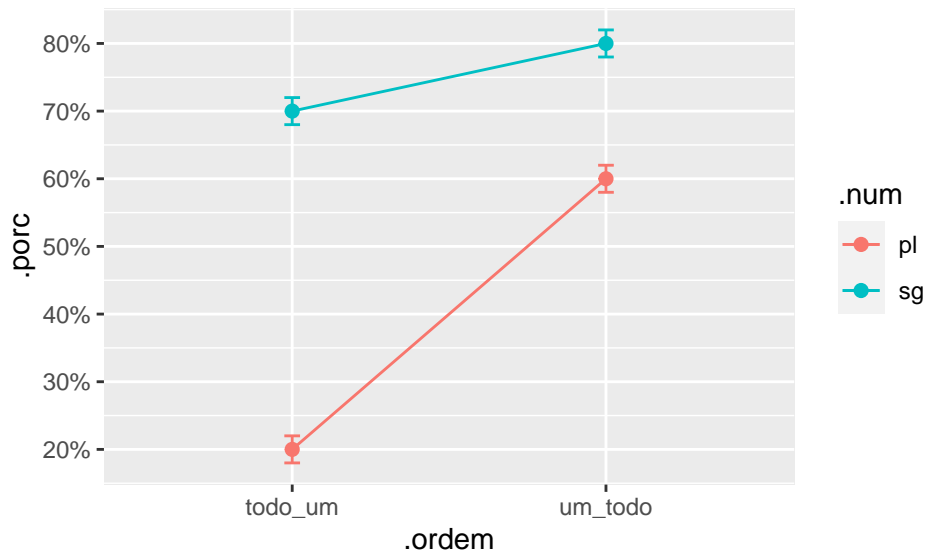
Ao se mudar de uma frase com a ordem ‘um_todo’ para uma frase com a ordem ‘todo_um’, há cerca de 7% de probabilidade (0.08 ou 1 chance para 12) de a resposta ser 1 nível abaixo na escala. Por exemplo, se as pessoas em geral marcam ‘concordo’ em uma frase com ‘um_todo’, há 7% de chance de elas marcarem abaixo, ou seja, ‘neutro’, em uma frase com ‘todo_um’. Apesar de ser uma probabilidade pequena, ela foi significativa.

Se as pessoas em geral marcam ‘concordo’ em uma frase com ‘plural’, há 83% de probabilidade (ou 5 chances para 1) de elas marcarem acima, ou seja, ‘concordo totalmente’ em uma frase com anáfora ‘singular’. *Grosso modo*, é isso que o modelo está dizendo.

Um adendo para explicar o efeito de interação

Observe o gráfico abaixo, cujos valores foram inventados, apenas para ilustração:

```
data.frame(
  .ordem = c("um_todo", "um_todo", "todo_um", "todo_um"),
  .num = c("sg", "pl", "sg", "pl"),
  .porc = c(.8, .6, .7, .2),
  .sd = c(rep(.02, 4))) %>%
  ggplot(aes(x = .ordem, y = .porc, group = .num, color = .num)) +
  geom_errorbar(aes(ymin = .porc-.sd, ymax = .porc+.sd), width = .05) +
  geom_point(size = 2) + geom_line() +
  scale_y_continuous(labels = scales::label_percent(accuracy = 1L),
    breaks = seq(from = 0, to = 1, by = .1))
```



Observe que mudar de plural (0) para singular (1) (ou vice-versa):

- Quando a ordem é 'um_todo', a mudança é de 20% (80 - 60) ou (60-80 = -20)
- Quando a ordem é 'todo_um', a mudança é de 50% (70 - 20) ou (20-70 = -50)
- A diferença é, portanto: 50 - 20 = 30 ou -50 - (-20) = -30

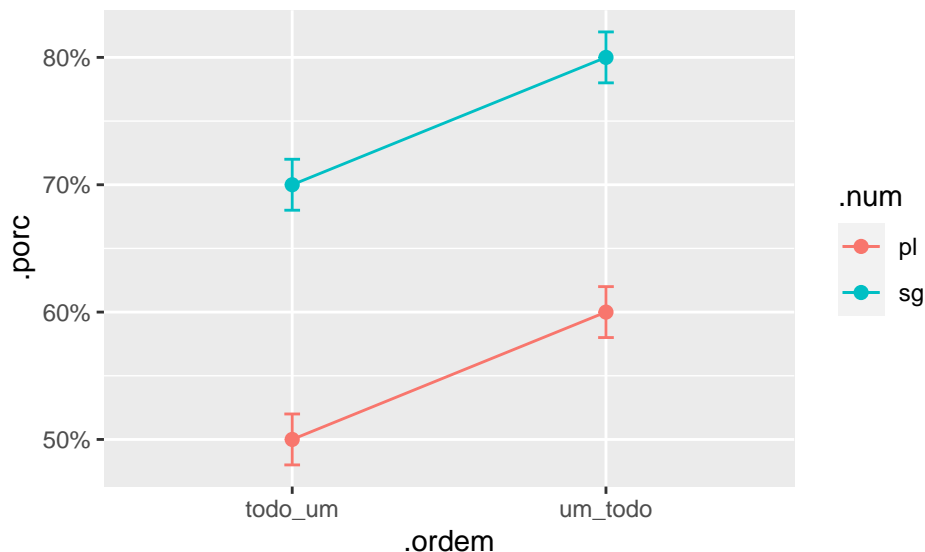
Observe também que mudar de 'um_todo' (0) para 'todo_um' (1) (ou vice-versa):

- Quando o número é 'singular', a mudança é de 10% (80 - 70) ou (70 - 80 = -10)
- Quando o número é plural, a mudança é de 40% (60 - 20) ou (20 - 60 = -40)
- A diferença é, portanto: 40 - 10 = 30 ou -40 - (-10) = -30

Repare que o “efeito de interação” é sempre o mesmo: a diferença das diferenças = 30% (ou -30% se for no sentido contrário, de 1 para 0). Por isso, não importa o modelo que você ajuste para esses dados, com quaisquer contrastes o valor da interação será o mesmo, mudando apenas o sinal em alguns casos!

Vamos fazer uma pequena mudança no gráfico, trocando aquele .2 por .5:

```
data.frame(
  .ordem = c("um_todo", "um_todo", "todo_um", "todo_um"),
  .num = c("sg", "pl", "sg", "pl"),
  .porc = c(.8, .6, .7, .5),
  .sd = c(rep(.02, 4))) %>%
  ggplot(aes(x = .ordem, y = .porc, group = .num, color = .num)) +
  geom_errorbar(aes(ymin = .porc-.sd, ymax = .porc+.sd), width = .05) +
  geom_point(size = 2) + geom_line() +
  scale_y_continuous(labels = scales::label_percent(accuracy = 1L),
    breaks = seq(from = 0, to = 1, by = .1))
```



Ao fazer isso, o que vemos é que o resultado, ao se mudar de ‘plural’ para ‘singular’ não é afetado pela ‘ordem’, que é 20% em ambos os casos: $20 - 20 = 0\%$ de alteração de um nível para outro; O resultado ao se mudar de ‘um_todo’ para ‘todo_um’ não é afetado pelo ‘número’, que é 10% em ambos os casos: $10 - 10 = 0\%$ de alteração de um nível para outro. Logo: não há efeito de interação!

Esse gráfico mostra bem, ainda, o que um modelo de regressão faz ao expor as ‘estimativas’. Se não há interação, o efeito de ‘singular’ (aumento de 20%) é o mesmo tanto em ‘todo_um’ quanto em ‘um_todo’. Esse pensamento é difícil para quem vem de um paradigma mental típico da Análise de Variância (Anova), que pensa sempre em contrastes de todas as categorias contra todas as categorias, como se faz em um teste *post-hoc*. O modelo de regressão contrasta apenas o que for estritamente necessário, não contrastando o que for redundante. Se, por exemplo, temos 3 categorias (A, B e C), ele apenas fará dois contrastes, porque o terceiro é inferível desses dois. Um exemplo pode deixar isso mais claro: se sabemos que Pedro é 5 anos mais velho do que João e que Maria é 2 anos mais velha do que Pedro, então não precisamos calcular a diferença de idade entre João e Maria, pois podemos inferi-la dos contrastes já dados.

No caso de 4 condições nos dados com que estamos trabalhando, ao fixar uma delas no intercepto, o modelo precisa apenas contrastar duas delas e estimar a interação para ter tudo o que é necessário. Se não há interação, o efeito do número obtido do contraste de duas condições é o mesmo para o contraste entre as demais (e o modelo não precisa perder tempo calculando esse valor). Se há interação, então o efeito do número obtido do contraste de duas condições não é o mesmo para as demais e o modelo está mostrando nas estimativas o contraste individual entre elas.

Investigando os dados do Mario

Carregando os dados e fazendo as alterações na tabela:

```
final_table <-
  read.csv("https://raw.githubusercontent.com/igor-de-costa/ModelosOrdinais/main/AnaliseEscalaLikert_Mario.csv",
           stringsAsFactors = T)

# Preparando a tabela
final_table$escala <- as.ordered(final_table$escala) # variável resposta ordenada
```

Preparando a estatística descritiva

Calculando valores absolutos

```

resp_escalabso <- final_table %>%
  filter(!is.na(escala)) %>% # Excluir células vazias NA
  group_by(Tamanho, Número, escala) %>% # agrupa pelas condições relevantes
  tally() %>% # Faz a contagem
  spread(escala, n)

colnames(resp_escalabso) <- c("Tamanho", "Número",
                             "Nada_natural", "Pouco_natural", "Neutro",
                             "Muito_natural", "Totalmente_natural")

```

Calculando valores percentuais

```

resp_escalaperc <- final_table %>%
  filter(!is.na(escala)) %>%
  group_by(Tamanho, Número, escala) %>%
  tally() %>%
  mutate(perc=n/sum(n)) %>%
  select(-n) %>%
  spread(escala, perc)

colnames(resp_escalaperc) <- c("Tamanho", "Número",
                              "Nada_natural", "Pouco_natural", "Neutro",
                              "Muito_natural", "Totalmente_natural")

```

E aqui vamos fazer o gráfico de barras empilhadas:

```

# Dividir o meio da escala (os julgamentos "Neutro"):
dados_meio <- resp_escalaperc %>%
  mutate(c1 = Neutro / 2,
         c2 = Neutro / 2) %>%
  select(Tamanho, Número,
         Nada_natural, Pouco_natural, c1, c2, Muito_natural, Totalmente_natural) %>%
  gather(key = Escolha, value = perc, 3:8)

# Separando a escala em dois conjuntos, o "alto" e o "baixo":
meio_alto <- dados_meio %>%
  filter(Escolha %in% c("Totalmente_natural", "Muito_natural", "c2")) %>%
  # Níveis na ordem normal!
  mutate(Escolha = factor(Escolha,
                          levels = c("Totalmente_natural", "Muito_natural", "c2")))

meio_baixo <- dados_meio %>%
  filter(Escolha %in% c("c1", "Pouco_natural", "Nada_natural")) %>%
  # Níveis na ordem inversa!
  mutate(Escolha = factor(Escolha,
                          levels = c("Nada_natural", "Pouco_natural", "c1")))

# Estabelecimento de uma paleta de cores para organização!
legend_pal<-c("#2B83BA", "#ABDDA4", "#FFFFBF", "#FFFFBF", "#FDAE61", "#D7191C")
legend_pal <- gsub("#FFFFBF", "#9C9C9C", legend_pal)
names(legend_pal) <- c("Totalmente_natural", "Muito_natural", "c1",
                      "c2", "Pouco_natural", "Nada_natural")

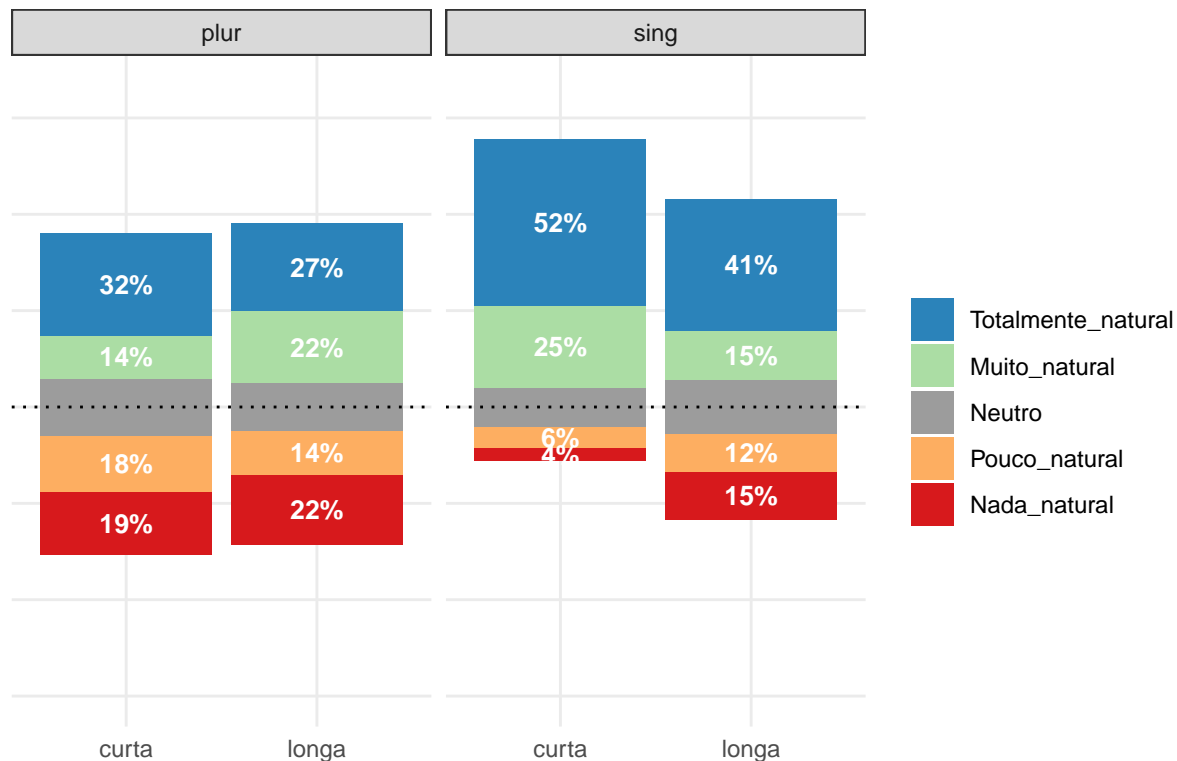
```

Produzindo o gráfico, finalmente!

```

ggplot() +
  geom_bar(data = meio_alto, aes(x = Tamanho,
                                y=perc, fill = Escolha), stat="identity") +
  geom_bar(data = meio_baixo, aes(x = Tamanho,
                                y=-perc, fill = Escolha), stat="identity") +
  #-----
  # Essa parte do código é um pouco complicada...
  # Serve apenas para colocar os valores dentro das barras
  geom_text(data = meio_alto,
            aes(x = Tamanho, y=perc, group = Escolha,
                alpha = Escolha,
                label = scales::percent(perc, accuracy = 1)),
            color = "white",
            size = 3.5, position = position_stack(vjust = .5),
            fontface = "bold") +
  geom_text(data = meio_baixo,
            aes(x = Tamanho, y=-perc, group = Escolha,
                alpha = Escolha,
                label = scales::percent(perc, accuracy = 1)),
            size = 3.5, position = position_stack(vjust = .5),
            color = "white",
            fontface = "bold") +
  # Não mostrar as % no meio da escala!
  scale_alpha_manual(values = c("c1" = 0, "c2" = 0,
                                "Muito_natural" = 1, "Totalmente_natural" = 1,
                                "Nada_natural" = 1, "Pouco_natural" = 1),
                    guide = 'none') +
  #-----
  geom_hline(yintercept = 0, color = c("black"), linetype = "dotted") +
  scale_fill_manual(values = legend_pal,
                    breaks = c("Totalmente_natural", "Muito_natural", "c2",
                                "Pouco_natural", "Nada_natural"),
                    labels = c("Totalmente_natural", "Muito_natural",
                                "Neutro", "Pouco_natural", "Nada_natural")) +
  labs(x = "", y = "", fill="") +
  facet_wrap(~Número) +
  theme_bw() +
  scale_y_continuous(labels = function(x) percent(abs(x)),
                    limits = c(-0.9, 1),
                    breaks = seq(from = -0.9, to = 1, by = .3)) +
  theme(axis.ticks.y = element_blank(),
        axis.ticks.x = element_blank(),
        axis.text.y = element_blank(),
        plot.background = element_blank(),
        panel.grid.minor = element_blank(),
        panel.border = element_blank()) +
  ggtitle("")

```



Exemplos de sentenças usadas pelo autor e suas categorizações respectivas:

- No escritório da firma desapareceu um documento sigiloso. (curta_sg)
- No escritório da firma desapareceu uns documentos sigilosos. (curta_pl)
- No canteiro do jardim desabrochou **repentinamente** uma azaleia belíssima. (longa_sg)
- No canteiro do jardim desabrochou **repentinamente** umas azaleias belíssimas. (longa_pl)

Aplicando um modelo ordinal aos dados usando ‘Flat priors’

```
# Ativando acesso a todos os núcleos do processador
options(mc.cores = parallel::detectCores())

# Ajustando o modelo
ordinal_mario <- brm(escala ~ Tamanho*Número +
  (1+Tamanho*Número|Part) + (1+Tamanho*Número|num),
  data = final_table,
  family = cumulative(link = "logit", threshold = "flexible"),
  silent = 2, refresh = 0)
```

```
## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## gcc -I"/usr/share/R/include" -DNDEBUG -I"/home/igor/R/x86_64-pc-linux-gnu-library/4.1/Rcpp/include"
## In file included from /home/igor/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/Core:88,
## from /home/igor/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/Dense:1,
## from /home/igor/R/x86_64-pc-linux-gnu-library/4.1/StanHeaders/include/stan/math/primitive/
## from <command-line>:
## /home/igor/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: e
## 628 | namespace Eigen {
##      | ~~~~~~
## /home/igor/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:17: c
## 628 | namespace Eigen {
```



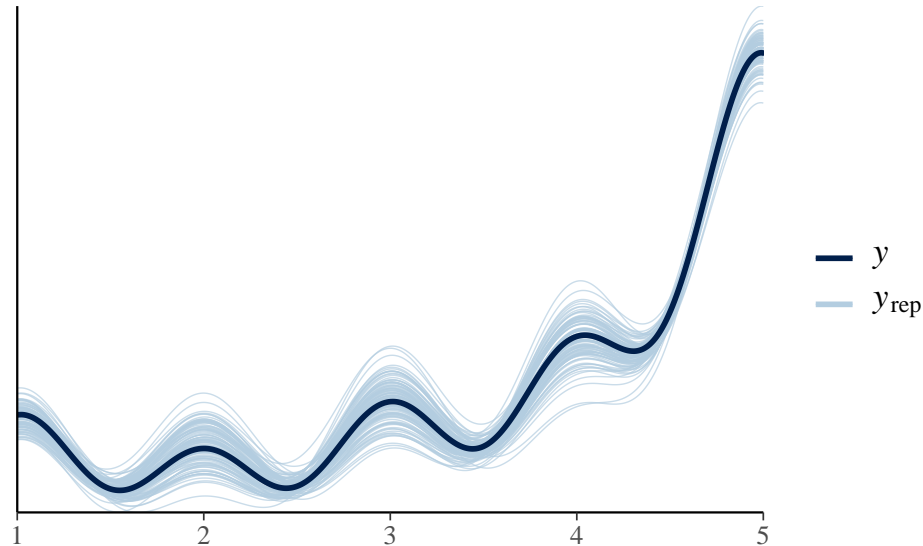
```
## |
## In file included from /home/igor/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/Dense:1,
##      from /home/igor/R/x86_64-pc-linux-gnu-library/4.1/StanHeaders/include/stan/math/pr
##      from <command-line>:
## /home/igor/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/Core:96:10: fatal error: comple
## 96 | #include <complex>
##      |      ^~~~~~
## compilation terminated.
## make: *** [/usr/lib/R/etc/Makeconf:168: foo.o] Erro 1
```

```
fixef(ordinal_mario)[5:7,]
```

```
##              Estimate Est.Error    Q2.5    Q97.5
## Tamanho longa      -0.3622743  1.335081 -3.097868  2.288892
## Número singular     6.2977747  2.393731  2.814354 12.153583
## Tamanho longa:Número singular -3.4247582  2.148716 -8.618445  0.019136
```

Checando a qualidade do ajuste...

```
pp_check(ordinal_mario, ndraws = 100)
```



Extrair os coeficientes dos fatores fixos e intervalos de confiança calculados

```
fixos.m<-fixef(ordinal_mario)[5:7,]
```

```
# Preparar um data frame com esses dados
```

```
colnames(fixos.m)<-c("Estimativas", "Est.Error", "lower", "upper")
```

```
fixos.m<-as.data.frame(fixos.m)
```

```
# Cálculo da chance e das probabilidades
```

```
fixos.m<-fixos.m %>%
```

```
  mutate(OddsRatio=exp(Estimativas)) %>%
```

```
  mutate(Probs=OddsRatio/(1 + OddsRatio)*100)
```

```
rownames(fixos.m)<-c("TAMANHO: longa", "NÚMERO: singular", "INTERAÇÃO: longa x sg")
```

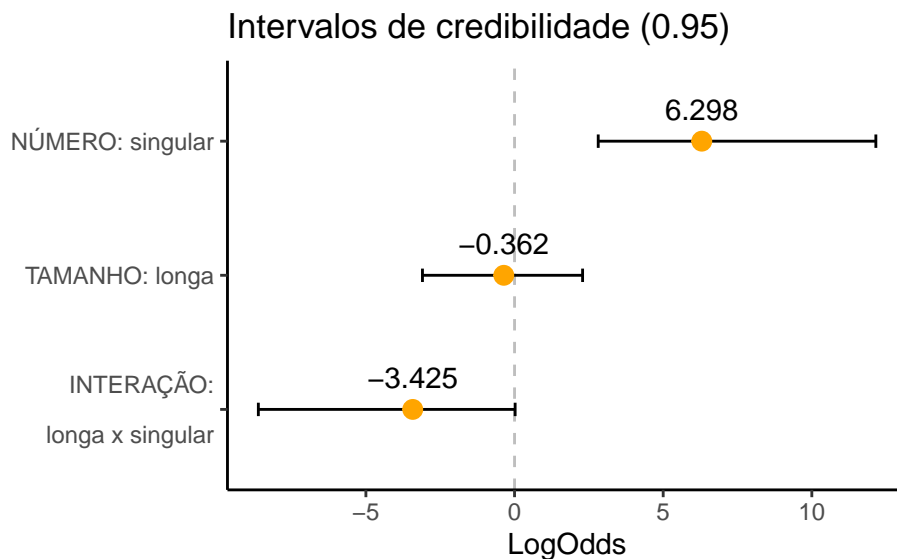
```
round(fixos.m, 3)
```

```
##              Estimativas Est.Error  lower  upper OddsRatio  Probs
```

```
## TAMANHO: longa          -0.362      1.335 -3.098  2.289      0.696 41.041
## NÚMERO: singular        6.298      2.394  2.814 12.154    543.361 99.816
## INTERAÇÃO: longa x sg   -3.425      2.149 -8.618  0.019      0.033  3.153
```

Plotar um gráfico para publicação...

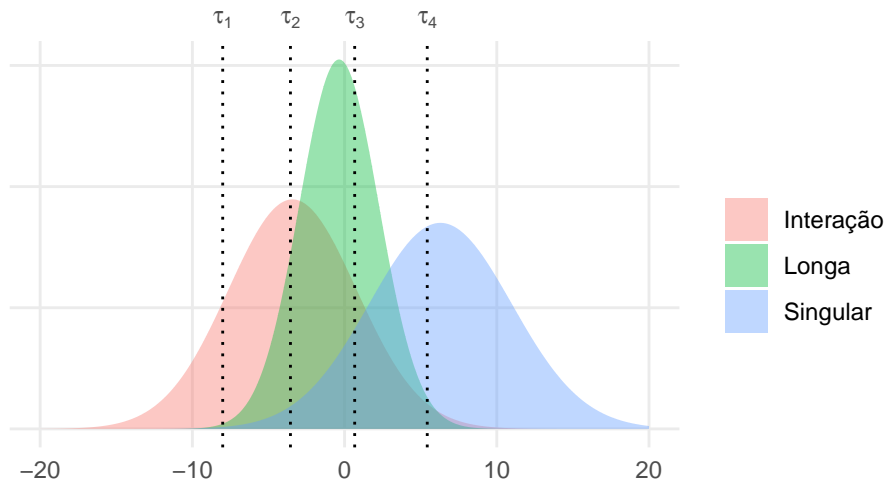
```
fixos.m %>%
  ggplot(aes(x=reorder(rownames(fixos.m), Estimativas),
               y=Estimativas))+
  geom_hline(yintercept = 0, linetype = "dashed", color = "grey")+
  geom_errorbar(aes(ymin=lower, ymax=upper), width=.1,
               position=position_dodge(.9))+
  geom_point(color="orange", size = 3) +
  geom_text(aes(label=round(Estimativas, 3)), vjust=-1)+
  # ao mudar a ordem é preciso mudar aqui
  scale_x_discrete(labels=c("INTERAÇÃO:\n\nlonga x singular",
                           "TAMANHO: longa",
                           "NÚMERO: singular"))+
  labs(y = "LogOdds", x = "") +
  ggtitle("Intervalos de credibilidade (0.95)") +
  coord_flip()+theme_classic()
```



E também o gráfico com a distribuição...

```
tibble(
  fatores = c("Longa", "Singular", "Interação"),
  mu = fixef(ordinal_mario)[5:7, 1], # Estimativas do modelo
  ci = 1.96*fixef(ordinal_mario)[5:7, 2]) %>% # Erro padrão x 1.96 (ci = .95)
  expand(nesting(fatores, mu, ci),
        y = seq(from = -20, to = 20, by = .01)) %>%
  mutate(d = dnorm(y, mu, ci)) %>%
  ggplot(aes(x = y, y = d, fill = fatores)) +
  geom_area(alpha = 0.4, position = "identity") +
  geom_vline(xintercept = fixef(ordinal_mario)[1:4, 1], linetype = "dotted") +
  scale_x_continuous(sec.axis = dup_axis(breaks = fixef(ordinal_mario)[1:4, 1] %>%
                                         as.double(),
                                         labels = parse(text = stringr::str_c("tau[", 1:4, "]")))) +
  labs(y = "", x = "", fill = "") +
```

```
theme_bw() +
  theme(axis.ticks.y = element_blank(),
        axis.ticks.x = element_blank(),
        axis.text.y = element_blank(),
        plot.background = element_blank(),
        panel.grid.minor = element_blank(),
        panel.border = element_blank())
```



Interpretando os resultados...

Lembrando nossos contrastes...

```
contrasts(final_table$Tamanho)
```

```
##      longa
## curta    0
## longa    1
```

```
contrasts(final_table$Número)
```

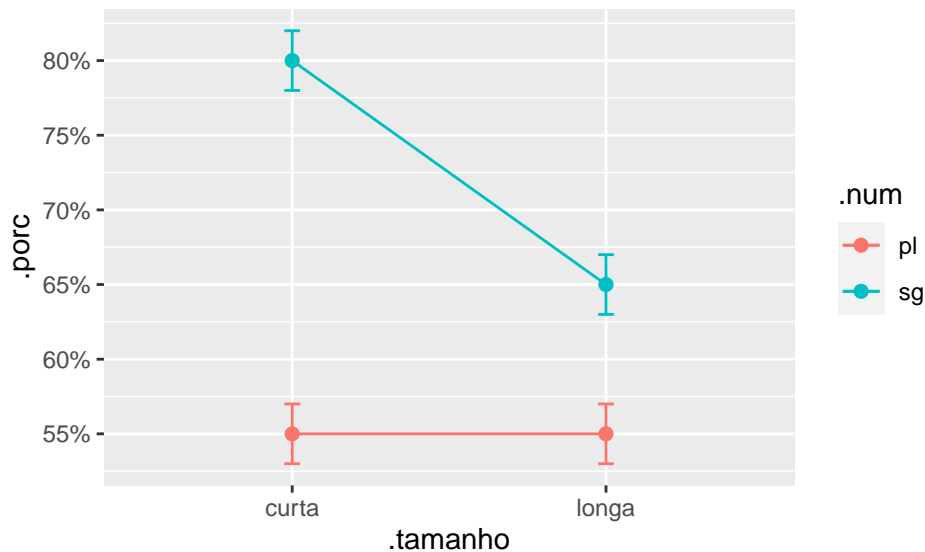
```
##      sing
## plur    0
## sing    1
```

- O fator significativo número singular (+6 logOdds) indica o contraste Curta (0) sg (1) x Curta (0) pl (0): singular aumenta a chance de se SUBIR na escala...
- O efeito não significativo de tamanho indica que nada se pode inferir quanto ao tamanho. O intervalo de credibilidade cortando a linha do zero indica que o parâmetro populacional pode tanto ser menor quanto maior do que zero... ou mesmo zero. Mantendo-se o plural constante, ao se mudar de longa para curta, não se pode inferir que há diferença.
- O intervalo de credibilidade para a interação ora corta a linha do zero ora não. Apenas por motivos pedagógicos, vamos assumir que ele seja significativo. Se é assim, então o nível singular aumenta MUITO a probabilidade de SUBIR na escala com curta e aumenta POUCO ou NADA com longa. Mas por que o valor em logOdds é negativo (-3 logOdds)? Como falamos antes, o resultado da interação é sempre o mesmo, o que muda é apenas o sinal.

De novo um adendo para compreender a interação...

Vamos olhar para o gráfico abaixo:

```
# Valores abaixo são inventados, apenas para ilustração
data.frame(.tamanho = c("curta", "curta", "longa", "longa"),
           .num = c("sg", "pl", "sg", "pl"),
           .porc = c(.8, .55, .65, .55),
           .sd = c(rep(.02, 4))) %>%
  ggplot(aes(x = .tamanho, y = .porc, group = .num, color = .num)) +
  geom_line() +
  geom_errorbar(aes(ymin = .porc-.sd, ymax = .porc+.sd), width = .05) +
  geom_point(size = 2) +
  scale_y_continuous(labels = scales::label_percent(accuracy = 1L),
                     breaks = seq(from = 0, to = 1, by = .05))
```



Quando se muda de plural (0) para singular (1):

- na curta: $55 - 80 = -25$
- na longa: $55 - 65 = -10$
- Diferença = $-25 - (-10) = -15\%$ (Sinal negativo)

Quando se muda de singular (1) para plural (0):

- na curta: $80 - 55 = 25$
- na longa: $65 - 55 = 10$
- Diferença = $25 - 10 = 15\%$ (Sinal positivo)

Quando se muda de curta (0) para longa (1):

- no singular $80 - 65 = 15$
- no plural $60 - 60 = 0$
- Diferença = 15% (Sinal positivo)

Quando se muda de longa (1) para curta (0):

- no singular $65 - 80 = -15$
- no plural $60 - 60 = 0$
- Diferença = -15% (Sinal negativo)

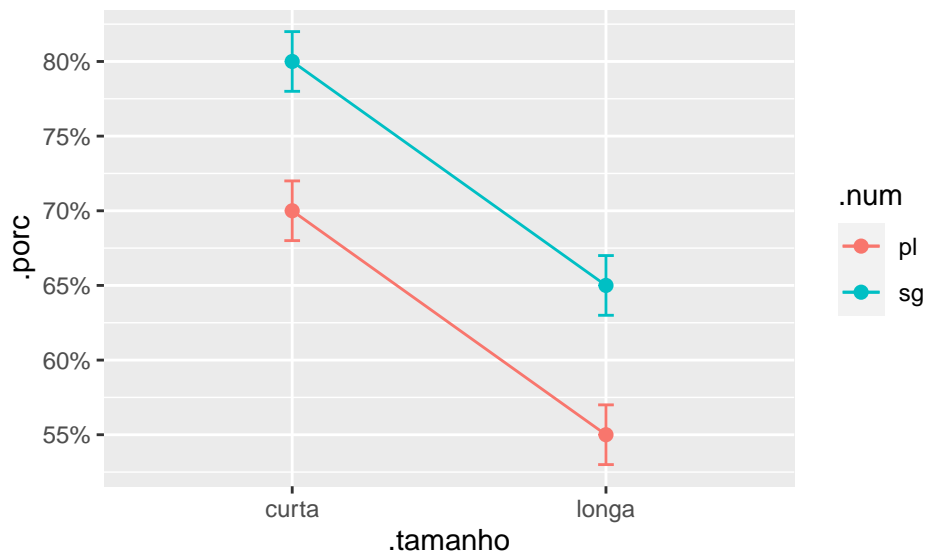
Se você quiser ver como fica sem o efeito de interação, basta mudar o segundo .55 para .7:

```
# Valores abaixo são inventados, apenas para ilustração
data.frame(.tamanho = c("curta", "curta", "longa", "longa"),
           .num = c("sg", "pl", "sg", "pl"),
```

```

.porc = c(.8, .7, .65, .55),
.sd = c(rep(.02, 4))) %>%
ggplot(aes(x = .tamanho, y = .porc, group = .num, color = .num)) +
  geom_line() +
  geom_errorbar(aes(ymin = .porc-.sd, ymax = .porc+.sd), width = .05) +
  geom_point(size = 2) +
  scale_y_continuous(labels = scales::label_percent(accuracy = 1L),
    breaks = seq(from = 0, to = 1, by = .05))

```



Ajustando um modelo com priors informativas (ou nem tanto)

NOTA DO AUTOR: Recomendo não discutirmos esses pontos agora. Acho que poderíamos nos aprofundar mais antes de avançarmos aqui. Comecei meus estudos pelas aulas disponíveis no Youtube do Prof. Shravan Vasishth. A primeira aula me parece fundamental para entender o que é uma função de verossimilhança (*Likelihood function*). Relembrem também as aulas do Gilberto, em que ele trabalhou esse ponto com bastante atenção, dando vários exemplos. Sem esse conhecimento fica muito difícil avançar para a segunda aula, em que ele discute a noção de priors, fundamentando os princípios matemáticos por trás dessa noção. Relembrem também a última aula do Gilberto, em que ele trabalhou a noção de priors, com um exemplo de uma moeda não viciada.

Eu ainda estou estudando o assunto e podemos ter uma conversa informal sobre o tema no dia da apresentação.