R-lecture-notes-2023-12-06-chi-squared-logistic-regression

Adam Tallman

2023-12-04

knitr::opts\_chunk$set(echo = TRUE)  
library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.3 ✔ readr 2.1.4  
## ✔ forcats 1.0.0 ✔ stringr 1.5.0  
## ✔ ggplot2 3.4.4 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.3 ✔ tidyr 1.3.0  
## ✔ purrr 1.0.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(ggdag)

##   
## Attaching package: 'ggdag'  
##   
## The following object is masked from 'package:stats':  
##   
## filter

library(V8)

## Using V8 engine 11.8.172.13

library(dagitty)  
library(Rling)  
library(rms)

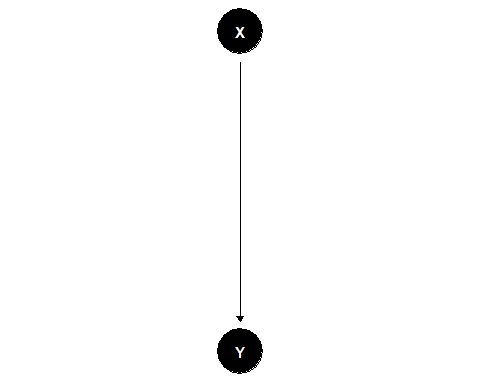
## Loading required package: Hmisc  
##   
## Attaching package: 'Hmisc'  
##   
## The following objects are masked from 'package:ggdag':  
##   
## label, label<-  
##   
## The following objects are masked from 'package:dplyr':  
##   
## src, summarize  
##   
## The following objects are masked from 'package:base':  
##   
## format.pval, units

## Warning in .recacheSubclasses(def@className, def, env): undefined subclass  
## "ndiMatrix" of class "replValueSp"; definition not updated

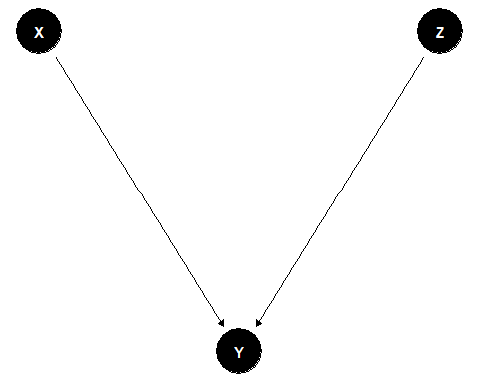
library(visreg)  
library(car)

## Loading required package: carData  
##   
## Attaching package: 'car'  
##   
## The following objects are masked from 'package:rms':  
##   
## Predict, vif  
##   
## The following object is masked from 'package:dplyr':  
##   
## recode  
##   
## The following object is masked from 'package:purrr':  
##   
## some

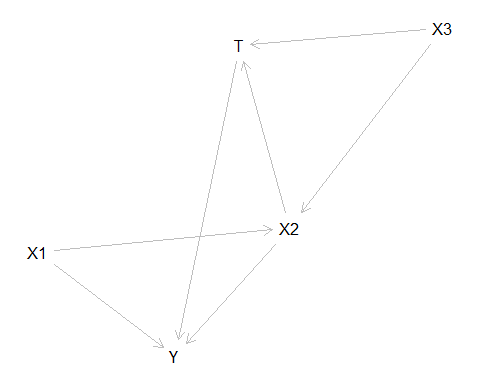
coord\_dag <- list(  
 x = c(X = 2, Y = 2),  
 y = c(X = 3, Y = 1)  
)  
our\_dag <- ggdag::dagify(Y ~ X,  
 coords = coord\_dag)  
ggdag::ggdag(our\_dag) + theme\_void()



coord\_dag <- list(  
 x = c(X = 1, Y = 2, Z = 3),  
 y = c(X = 3, Y = 1, Z = 3)  
)  
our\_dag <- ggdag::dagify(Y ~ X,  
 Y ~ Z,  
 coords = coord\_dag)  
ggdag::ggdag(our\_dag) + theme\_void()



dag <- dagitty("dag {  
 X1 -> X2  
 X1 -> Y  
 X3 -> X2  
 X2 -> Y  
 X2 -> T -> Y  
 X3 -> T  
 }")  
plot( graphLayout( dag ) )



## Chi-squared test

## Logistic regression

mcf <- read.csv("/Users/Adan Tallman/Desktop/StatisticsinLinguistics\_FSU\_2023\_2024/07\_data/chacobo.mcf.df.csv")

Recode data as 0 vs. 1 and then extract the simuli. After this we pull out the pitch and the duration values using strsplit.

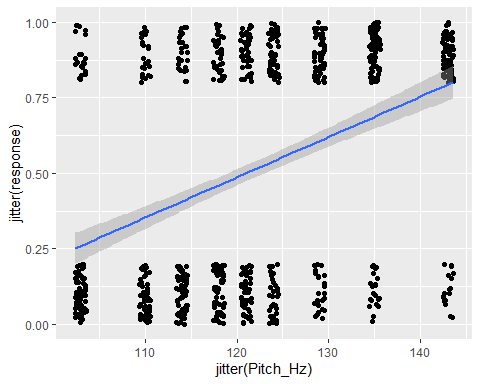
response <- ifelse(mcf$response=="janáquë vomitó", 1, 0)  
stimulus <- str\_remove\_all(mcf$stimulus, "janaquë\_")  
df <- as.data.frame(cbind(response, stimulus))  
df[c("Pitch\_Hz", "Duration\_ms")] <- str\_split\_fixed(df$stimulus, "\_", 2)  
df$Pitch\_Hz<-as.numeric(df$Pitch\_Hz)  
df$response <- as.numeric(df$response)  
head(df)

## response stimulus Pitch\_Hz Duration\_ms  
## 1 0 103\_80 103 80  
## 2 1 143\_40 143 40  
## 3 1 118\_110 118 110  
## 4 0 114\_100 114 100  
## 5 1 118\_90 118 90  
## 6 0 110\_100 110 100

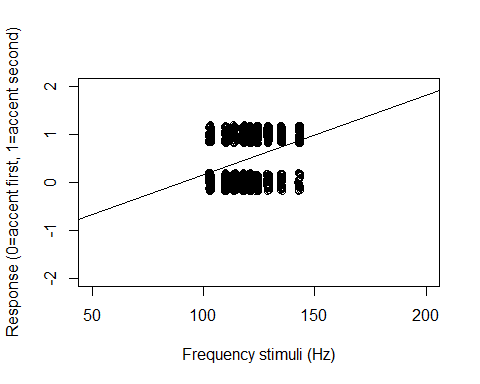
ggplot(df, aes(jitter(Pitch\_Hz), jitter(response))) +   
 geom\_point()+  
 stat\_smooth(method="lm",  
 formula = y ~ x,  
 geom= "smooth")+  
 ylim(0,1)

## Warning: Removed 815 rows containing non-finite values (`stat\_smooth()`).

## Warning: Removed 801 rows containing missing values (`geom\_point()`).



plot(jitter(df$response)~jitter(df$Pitch\_Hz), ylab="Response (0=accent first, 1=accent second)", xlab ="Frequency stimuli (Hz)", xlim=c(50,200), ylim=c(-2,2)) + abline(lm(response ~ Pitch\_Hz, data = df))



## integer(0)

There’s certain aspects of this model that are not realistic. For instance, it makes predictions about when the response variable is lower than or higher than 0. What type of line would we want to best fit the data and to give an accurate assessment of the probability of 0 or 1 of the response variable given a value of x?

bernouilli\_data <- rbinom(20, 1, 0.5)  
print(bernouilli\_data)

## [1] 1 1 1 0 1 1 0 0 1 0 1 0 0 0 1 1 1 1 1 0

## Log / Exponent function

Logistic regression model is about telling us where the probability changes conditional on other variables.

There are two functions - the logarithmic function and the exponential function. They can reverse.

t <- log(10)  
t

## [1] 2.302585

exp(t)

## [1] 10

## Logistic regression

# Odds plot

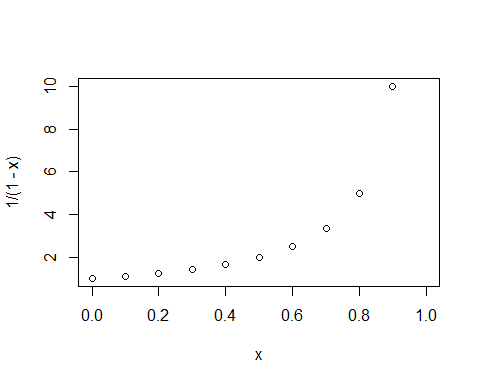
If you want to understand the relationship between normal numbers and logs. You can plot the relationship with the following code.

First make a bunch of numbers from 0 to 1.

par(mfrow=c(1,1))  
x <- seq(0,1,by=0.1)

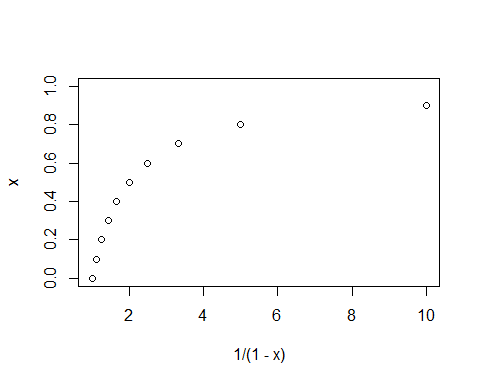
The odds are calculated with the formula

plot(x, 1/(1-x))



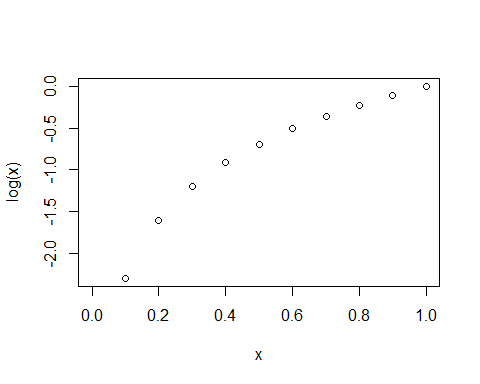
Or if its easier reverse the relationship.

plot(1/(1-x),x)

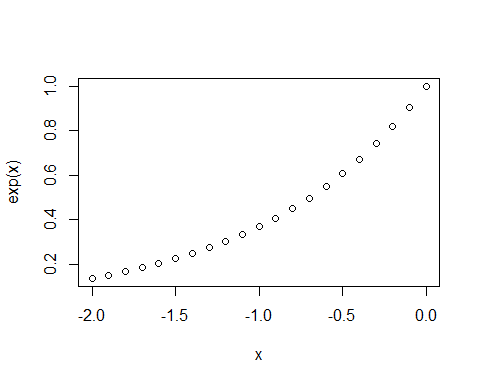


# Plot logs

x <- seq(0,1,by=0.1)  
plot(x,log(x))

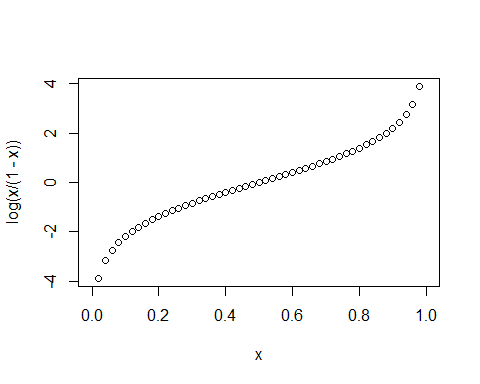
 # Plot exponentials

x <- seq(-2,0, by =.1)  
plot(x, exp(x))

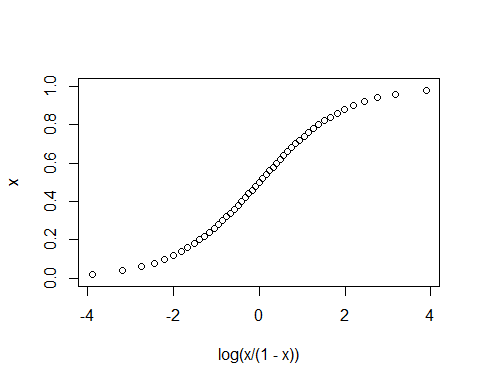


# Log odds

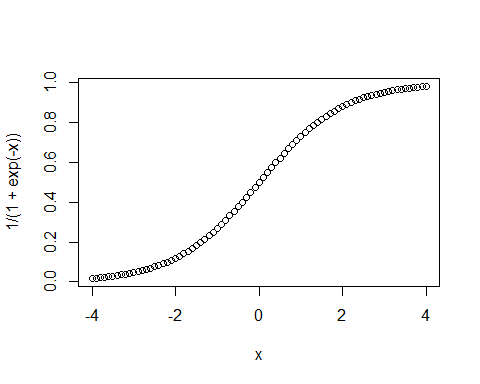
par(mfrow=c(1,1))  
x <- seq(0,1,by=.02)  
plot(x,log(x/(1-x)))



plot(log(x/(1-x)),x)

 # Exponential odds

par(mfrow=c(1,1))  
x <- seq(-4,4,by=.1)  
plot(x, 1/(1+exp(-x)))

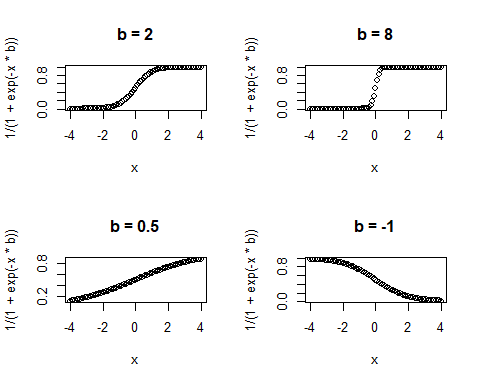


So the trick with log/exponential odds is that it can translate any numbers onto a 0 to 1 scale. This is useful if we want to ask questions about probability.

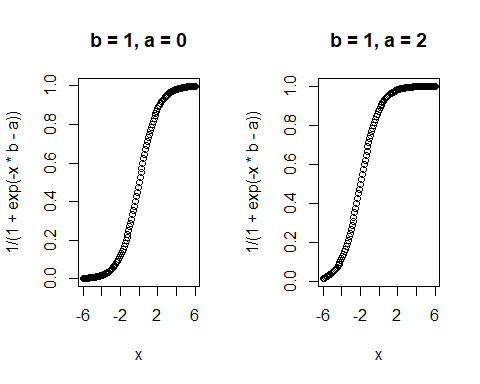
## Adding a coefficient

If you make the slope larger.

par(mfrow=c(2,2))  
x <- seq(-4,4,by=.1)  
b <- 2  
plot(x, 1/(1+exp(-x\*b)), main = "b = 2")  
b <- 8  
plot(x, 1/(1+exp(-x\*b)), main = "b = 8")  
  
b <- 0.5  
plot(x, 1/(1+exp(-x\*b)), main = "b = 0.5")  
  
b <- -1  
plot(x, 1/(1+exp(-x\*b)), main = "b = -1")



par(mfrow=c(1,2))  
x <- seq(-6,6,by=.1)  
b <- 1  
a <- 0  
plot(x, 1/(1+exp(-x\*b-a)), main = "b = 1, a = 0")  
b <- 1  
a <- 2  
plot(x, 1/(1+exp(-x\*b-a)), main = "b = 1, a = 2")



b <- 1  
a <- 1  
y <- 1/(1+exp(-a-x\*b))  
glm(y~x)

##   
## Call: glm(formula = y ~ x)  
##   
## Coefficients:  
## (Intercept) x   
## 0.5822 0.1098   
##   
## Degrees of Freedom: 120 Total (i.e. Null); 119 Residual  
## Null Deviance: 19.5   
## Residual Deviance: 1.711 AIC: -165.9

## Logistic regression practice

Let’s make a model that predicts whether a student is going to class.

set.seed(1)  
coffee <- rnorm(100, 15, 5)  
happiness <- rnorm(100, 10, 2)  
a <- -5  
b1 <- -1  
b2 <- 1

How do we translate the our continuous values onto predictions from 0 (no, they are not a postdoc) to 1 (yes the person is a postdoc).

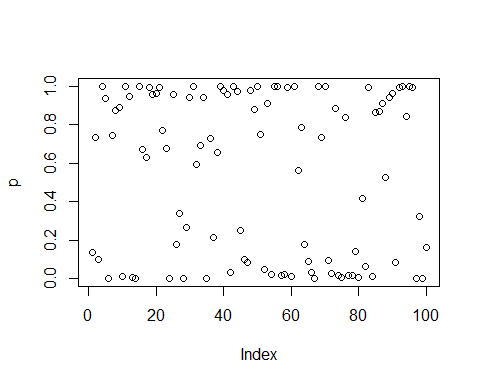
xb <- a + b1 \* happiness + b2 \* coffee + rnorm(100, 0, 0.1)

Then we generate probabilities using logistic regression model.

p <- 1/ (1+exp(-xb))  
summary(p)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0000497 0.0780173 0.7105522 0.5493012 0.9611723 0.9999948

par(mfrow=c(1,1))  
plot(p)



gotoclass <- rbinom(n=100, size=1, prob=p)  
gotoclass

## [1] 0 0 0 1 0 0 1 1 0 0 1 1 0 0 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 1 1 1 1 1 0 1 0  
## [38] 1 1 1 1 0 1 1 1 0 0 1 1 1 1 0 1 0 1 1 0 0 1 0 1 0 1 0 0 0 0 1 1 1 0 0 1 0  
## [75] 0 1 0 0 0 0 1 0 1 0 1 1 1 1 1 1 0 1 1 1 1 1 0 0 0 0

library(glm2)

model.logit <- glm(gotoclass~happiness+coffee, family="binomial")  
summary(model.logit)

##   
## Call:  
## glm(formula = gotoclass ~ happiness + coffee, family = "binomial")  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -4.1157 2.8897 -1.424 0.15438   
## happiness -1.6263 0.5221 -3.115 0.00184 \*\*   
## coffee 1.3600 0.3348 4.062 4.87e-05 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 137.186 on 99 degrees of freedom  
## Residual deviance: 36.172 on 97 degrees of freedom  
## AIC: 42.172  
##   
## Number of Fisher Scoring iterations: 8

## Interpreting logistic regression coefficients

invlogit <- function (x) {1/(1+exp(-x))}  
invlogit(-4.1157)

## [1] 0.01605263

invlogit <- function (x) {1/(1+exp(-x))}  
invlogit(-4.1157 + -1.6263\*mean(happiness) + 1.3600\*mean(coffee))

## [1] 0.707797

-1.6263\*mean(happiness)

## [1] -16.14003

1.3600\*mean(coffee)

## [1] 21.14043

Divide by 4 gives you the maximum difference corresponding to a unit difference in happiness.

-1.6263/4

## [1] -0.406575

## Visualization of logistic regression

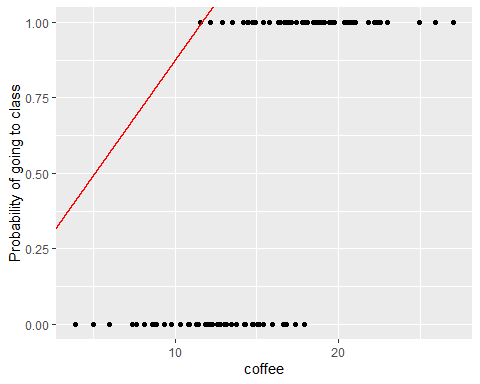
library(ggplot2)  
library(ggpubr)

par(mfrow=c(1,2))  
data <- data.frame(gotoclass, happiness, coffee)

plotline<-ggplot(data, aes(x=coffee, y = gotoclass))+geom\_point()+  
 geom\_abline(intercept = 0.11105, slope = 0.076446, color="red", size=1)+   
 ylab("Probability of going to class")

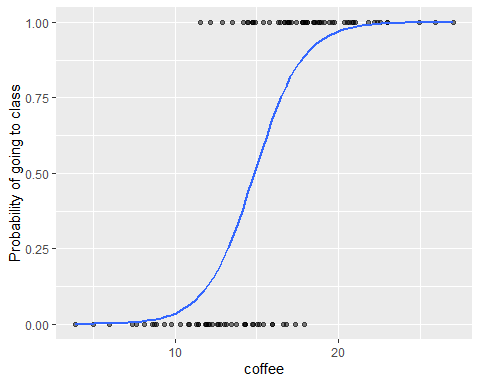
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.  
## ℹ Please use `linewidth` instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

plotline



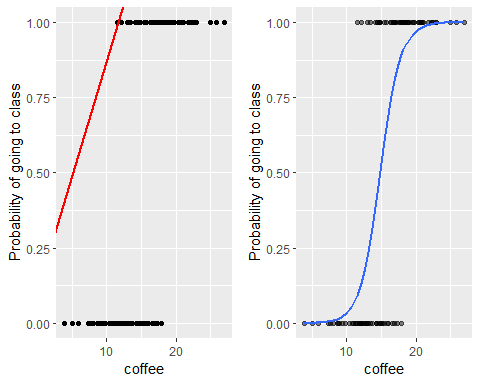
plotS<-ggplot(data, aes(x=coffee, y= gotoclass))+   
 geom\_point(alpha=.5)+  
 stat\_smooth(method="glm", se=FALSE, method.args = list(family=binomial))+   
 ylab("Probability of going to class")   
plotS

## `geom\_smooth()` using formula = 'y ~ x'



ggarrange(plotline, plotS)

## `geom\_smooth()` using formula = 'y ~ x'



## Interpreting logistic regression

library(Rling);library(rms); library(visreg); library(car)

data(doenLaten)  
d <- doenLaten  
head(doenLaten)

## Aux Country Causation EPTrans EPTrans1  
## 1 laten NL Inducive Intr Intr  
## 2 laten NL Physical Intr Intr  
## 3 laten NL Inducive Tr Tr  
## 4 doen BE Affective Intr Intr  
## 5 laten NL Inducive Tr Tr  
## 6 laten NL Volitional Intr Intr

library(tidyverse)  
glimpse(d)

## Rows: 455  
## Columns: 5  
## $ Aux <fct> laten, laten, laten, doen, laten, laten, doen, doen, doen, l…  
## $ Country <fct> NL, NL, NL, BE, NL, NL, NL, NL, NL, BE, BE, BE, BE, BE, BE, …  
## $ Causation <fct> Inducive, Physical, Inducive, Affective, Inducive, Volitiona…  
## $ EPTrans <fct> Intr, Intr, Tr, Intr, Tr, Intr, Intr, Tr, Intr, Intr, Intr, …  
## $ EPTrans1 <fct> Intr, Intr, Tr, Intr, Tr, Intr, Intr, Tr, Intr, Intr, Intr, …

d <- as.data.frame(d)  
model.glm <- glm(Aux~Causation\*EPTrans\*Country, data=d, family="binomial")  
summary(model.glm)

##   
## Call:  
## glm(formula = Aux ~ Causation \* EPTrans \* Country, family = "binomial",   
## data = d)  
##   
## Coefficients: (2 not defined because of singularities)  
## Estimate Std. Error z value Pr(>|z|)  
## (Intercept) 1.8971 0.6191 3.064 0.00218  
## CausationInducive -2.8779 0.7322 -3.931 8.47e-05  
## CausationPhysical 0.5878 0.9618 0.611 0.54110  
## CausationVolitional -3.8666 0.7571 -5.107 3.28e-07  
## EPTransTr -1.6458 0.7983 -2.062 0.03925  
## CountryBE 0.8755 0.9563 0.915 0.35996  
## CausationInducive:EPTransTr -0.9429 1.1420 -0.826 0.40900  
## CausationPhysical:EPTransTr NA NA NA NA  
## CausationVolitional:EPTransTr 1.1303 1.3822 0.818 0.41349  
## CausationInducive:CountryBE -0.9507 1.1117 -0.855 0.39245  
## CausationPhysical:CountryBE -0.4982 1.4088 -0.354 0.72363  
## CausationVolitional:CountryBE -0.2410 1.1094 -0.217 0.82801  
## EPTransTr:CountryBE 0.4137 1.2543 0.330 0.74156  
## CausationInducive:EPTransTr:CountryBE 1.2852 1.6122 0.797 0.42535  
## CausationPhysical:EPTransTr:CountryBE NA NA NA NA  
## CausationVolitional:EPTransTr:CountryBE 0.5205 1.9165 0.272 0.78593  
##   
## (Intercept) \*\*   
## CausationInducive \*\*\*  
## CausationPhysical   
## CausationVolitional \*\*\*  
## EPTransTr \*   
## CountryBE   
## CausationInducive:EPTransTr   
## CausationPhysical:EPTransTr   
## CausationVolitional:EPTransTr   
## CausationInducive:CountryBE   
## CausationPhysical:CountryBE   
## CausationVolitional:CountryBE   
## EPTransTr:CountryBE   
## CausationInducive:EPTransTr:CountryBE   
## CausationPhysical:EPTransTr:CountryBE   
## CausationVolitional:EPTransTr:CountryBE   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 609.05 on 454 degrees of freedom  
## Residual deviance: 330.03 on 441 degrees of freedom  
## AIC: 358.03  
##   
## Number of Fisher Scoring iterations: 6

## Logistic regression on Forced Choice Experiment Chacobo

head(df)

## response stimulus Pitch\_Hz Duration\_ms  
## 1 0 103\_80 103 80  
## 2 1 143\_40 143 40  
## 3 1 118\_110 118 110  
## 4 0 114\_100 114 100  
## 5 1 118\_90 118 90  
## 6 0 110\_100 110 100

logit\_model\_01 <- glm(response~Pitch\_Hz, data=df, family="binomial")  
summary(logit\_model\_01)

##   
## Call:  
## glm(formula = response ~ Pitch\_Hz, family = "binomial", data = df)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -9.109549 0.626233 -14.55 <2e-16 \*\*\*  
## Pitch\_Hz 0.075658 0.005149 14.70 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 2242.6 on 1619 degrees of freedom  
## Residual deviance: 1980.1 on 1618 degrees of freedom  
## AIC: 1984.1  
##   
## Number of Fisher Scoring iterations: 4

0.075658/4

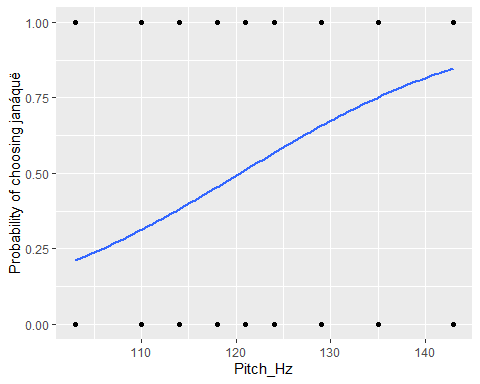
## [1] 0.0189145

invlogit(-9.109549+ 0.075658\*mean(df$Pitch\_Hz))

## [1] 0.5287489

plotS<-ggplot(dat=df, aes(x=Pitch\_Hz, y= response))+   
 geom\_point(alpha=.5)+  
 stat\_smooth(method="glm", se=FALSE, method.args = list(family=binomial))+   
 ylab("Probability of choosing janáquë")   
plotS

## `geom\_smooth()` using formula = 'y ~ x'



## How to make an S shaped curve without data

x <- seq(-4,4,length.out=100)  
p <-1/(1+exp(-x))  
plot(x, p, type="l")

