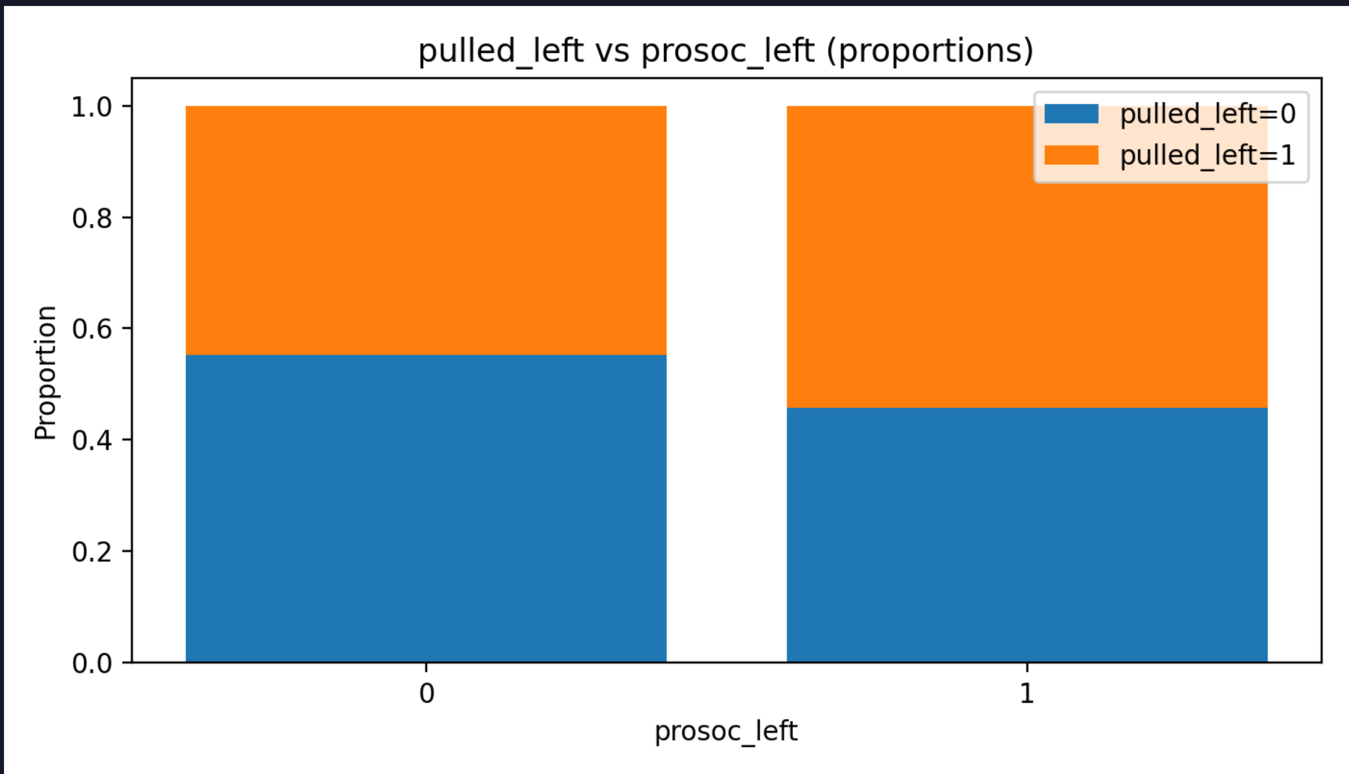
Bayesian Logistic Regression (Chimpanzees)

Step 1 — Exploratory Data Analysis (EDA)

This section checks how pulled_left changes across prosoc_left and condition using stacked proportion bars.

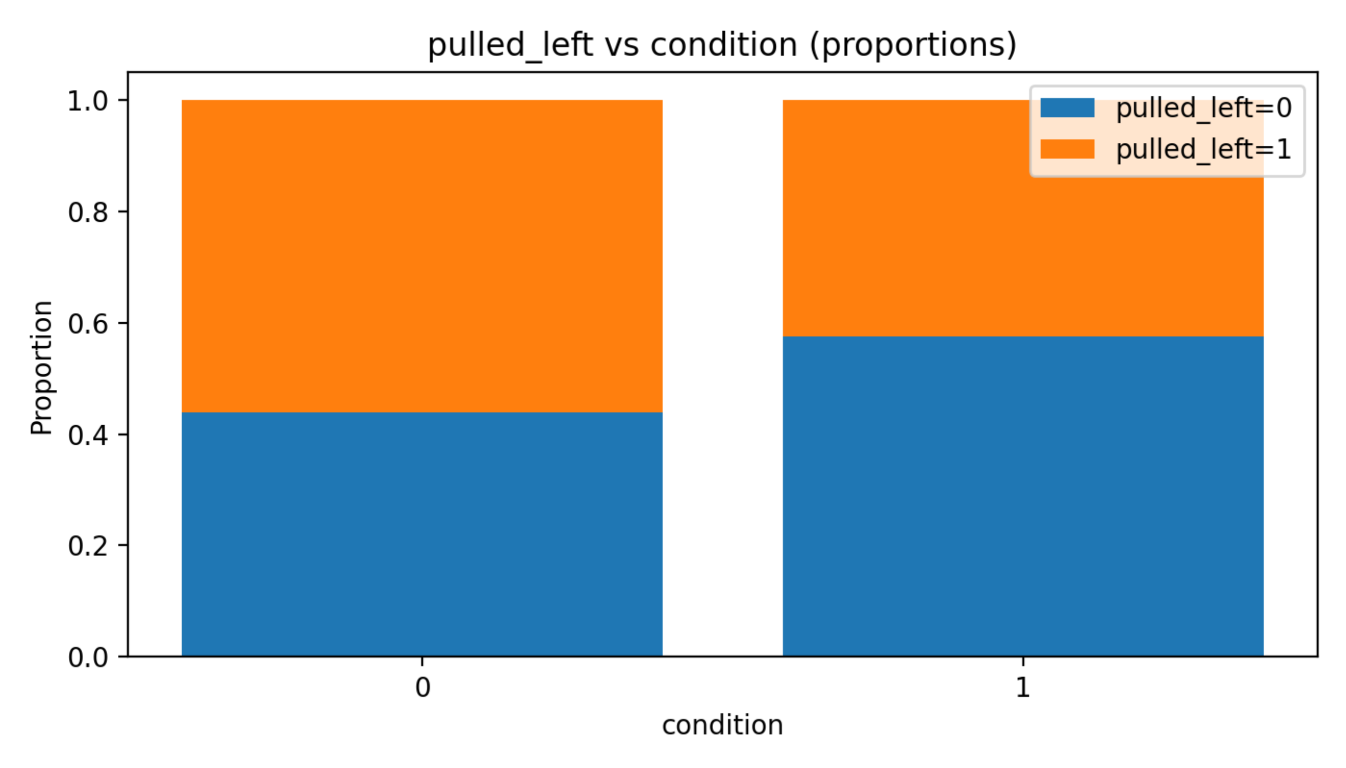
Generated by the block:

```
p1 <- ggplot(datos, aes(x = factor(prosoc_left), fill = factor(pulled_left))) +
  geom_bar(position = "fill")
```



Generated by the block:

```
p2 <- ggplot(datos, aes(x = factor(condition), fill = factor(pulled_left))) +
  geom_bar(position = "fill")
```



Step 2 — Bayesian Models (Stan)

Two Bayesian logistic regressions are fit in Stan: a baseline model and an interaction model where the effect of prosoc_left depends on condition .

```
logit(pulled_left) = alpha + betaP · prosoc_left

logit(pulled_left) = alpha + (beta_P + beta_PC · condition) · prosoc_left
```

Step 3 — Model Comparison

Models are compared with a Bayes Factor computed using bridge sampling (bridgesampling), providing evidence for or against the interaction effect.

Full Reproducible R Script (copy/paste)

Place chimpanzees.csv next to the script (or change data_path) and run in R/RStudio.

```
# =========================================
# Bayesian Logistic Regression (Chimpanzees)
# EDA + Stan models (basic vs interaction) + Bayes Factor
# =========================================

# ---- Libraries ----
library(rstan)
library(tidyverse)
library(readr)
library(bayesplot)
library(coda)
library(bridgesampling)

# ---- Stan performance ----
options(mc.cores = parallel::detectCores())
rstan_options(auto_write = TRUE)

# ---- Data ----
# Your CSV uses ';' as delimiter.
data_path <- "chimpanzees.csv"   # Put the file next to this script or set a full path
```

```r
datos <- read_delim(data_path, delim = ";", show_col_types = FALSE) %>%
  as.data.frame()

stopifnot(all(c("pulled_left", "prosoc_left", "condition") %in% names(datos)))

# ===========================================
# (A) Exploratory Data Analysis (EDA)
# ===========================================

# 1) Contingency tables + column-wise proportions
tabla_prosoc <- table(datos$pulled_left, datos$prosoc_left)
prop_prosoc  <- prop.table(tabla_prosoc, margin = 2)

tabla_condition <- table(datos$pulled_left, datos$condition)
prop_condition  <- prop.table(tabla_condition, margin = 2)

# 2) Stacked proportion bars (equivalent to ggplot position="fill")
# --- Figure 1 is generated by this block ---
p1 <- ggplot(datos, aes(x = factor(prosoc_left), fill = factor(pulled_left))) +
  geom_bar(position = "fill") +
  labs(
    x = "prosoc_left",
    y = "Proportion",
    fill = "pulled_left",
    title = "pulled_left vs prosoc_left (proportions)"
  ) +
  theme_minimal()

# --- Figure 2 is generated by this block ---
p2 <- ggplot(datos, aes(x = factor(condition), fill = factor(pulled_left))) +
  geom_bar(position = "fill") +
  labs(
    x = "condition",
    y = "Proportion",
    fill = "pulled_left",
    title = "pulled_left vs condition (proportions)"
  ) +
  theme_minimal()

# Uncomment to export images:
# ggsave("pulled_left_vs_prosoc_left.png", p1, width = 7, height = 4, dpi = 200)
# ggsave("pulled_left_vs_condition.png",  p2, width = 7, height = 4, dpi = 200)

# 3) Simple descriptive summary
est_descrip <- datos %>%
  group_by(prosoc_left, condition) %>%
  summarise(
    n = n(),
    p_pulled_left = mean(pulled_left),
    .groups = "drop"
  )

# ===========================================
# Stan data objects
# ===========================================
stan_basic_data <- list(
  N = nrow(datos),
  L = as.integer(datos$pulled_left),
  P = as.integer(datos$prosoc_left)
)

stan_int_data <- list(
  N = nrow(datos),
  L = as.integer(datos$pulled_left),
  P = as.integer(datos$prosoc_left),
  C = as.integer(datos$condition)
)

# ===========================================
# (B) Basic model in Stan
# logit(pulled_left) = alpha + betaP * prosoc_left
# ===========================================
```

```
stan_basic_code <- "
data {
  int<lower=0> N;
  array[N] int<lower=0,upper=1> L;
  array[N] int<lower=0,upper=1> P;
}
parameters {
  real alpha;
  real betaP;
}
model {
  alpha ~ normal(0, 10);
  betaP ~ normal(0, 10);
  for (i in 1:N) {
    L[i] ~ bernoulli_logit(alpha + betaP * P[i]);
  }
}
generated quantities {
  vector[N] log_lik;
  for (i in 1:N) {
    log_lik[i] = bernoulli_logit_lpmf(L[i] | alpha + betaP * P[i]);
  }
}
"

writeLines(stan_basic_code, "model_basic.stan")
mod_basic <- rstan::stan_model(file = "model_basic.stan")

set.seed(2542)
fit_basic <- rstan::sampling(
  object = mod_basic,
  data = stan_basic_data,
  iter = 6000,
  warmup = 1500,
  chains = 4,
  seed = 2542,
  control = list(adapt_delta = 0.95)
)

print(fit_basic, pars = c("alpha", "betaP"), probs = c(0.1, 0.5, 0.9))

bayesplot::mcmc_intervals(as.array(fit_basic), pars = c("alpha", "betaP")) +
  ggtitle("Credible intervals (basic model)")

basic_draws <- as.matrix(fit_basic)
coda::heidel.diag(coda::as.mcmc(basic_draws))

# ==========================================
# (C) Interaction model in Stan
# logit(pulled_left) = alpha + (beta_P + beta_PC * condition) * prosoc_left
# ==========================================
stan_int_code <- "
data {
  int<lower=0> N;
  array[N] int<lower=0,upper=1> L;
  array[N] int<lower=0,upper=1> P;
  array[N] int<lower=0,upper=1> C;
}
parameters {
  real alpha;
  real beta_P;
  real beta_PC;
}
model {
  alpha ~ normal(0, 10);
  beta_P ~ normal(0, 10);
  beta_PC ~ normal(0, 10);

  for (i in 1:N) {
    L[i] ~ bernoulli_logit(alpha + (beta_P + beta_PC * C[i]) * P[i]);
  }
}
```

```
generated quantities {
  vector[N] log_lik;
  for (i in 1:N) {
    log_lik[i] = bernoulli_logit_lpmf(L[i] | alpha + (beta_P + beta_PC * C[i]) * P[i]);
  }
}
"

writeLines(stan_int_code, "model_interaction.stan")
mod_int <- rstan::stan_model(file = "model_interaction.stan")

fit_int <- rstan::sampling(
  object = mod_int,
  data = stan_int_data,
  iter = 6000,
  warmup = 1500,
  chains = 4,
  seed = 2542,
  control = list(adapt_delta = 0.95)
)

print(fit_int, pars = c("alpha", "beta_P", "beta_PC"), probs = c(0.1, 0.5, 0.9))

bayesplot::mcmc_intervals(as.array(fit_int), pars = c("alpha", "beta_P", "beta_PC")) +
  ggtitle("Credible intervals (interaction model)")

int_draws <- as.matrix(fit_int)
coda::heidel.diag(coda::as.mcmc(int_draws))

# ========================================
# (D) Bayes Factor via Bridge Sampling
# ========================================
bridge_basic <- bridgesampling::bridge_sampler(fit_basic, silent = TRUE)
bridge_int   <- bridgesampling::bridge_sampler(fit_int, silent = TRUE)

BF_10 <- bridgesampling::bf(bridge_int, bridge_basic)  # Interaction vs Basic
print(BF_10)
```