artcile

Ismail Guennouni

2024-11-06

Introduction

Methods

```r
library(papaja)
```

```
## Loading required package: tinylabels

## Registered S3 methods overwritten by 'effectsize':
##   method              from
##   standardize.Surv    datawizard
##   standardize.bcplm   datawizard
##   standardize.clm2    datawizard
##   standardize.default datawizard
##   standardize.mediate datawizard
##   standardize.wbgee   datawizard
##   standardize.wbm     datawizard
```

```r
library(kableExtra)
require(knitr)
```

```
## Loading required package: knitr
```

```r
# using some functions dplyr, ggpubr, PairedData and sjPlot. Need to be loaded.
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter()     masks stats::filter()
## x dplyr::group_rows() masks kableExtra::group_rows()
## x dplyr::lag()        masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(afex)
```

```
## Loading required package: lme4
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## ************
## Welcome to afex. For support visit: http://afex.singmann.science/
## - Functions for ANOVAs: aov_car(), aov_ez(), and aov_4()
## - Methods for calculating p-values with mixed(): 'S', 'KR', 'LRT', and 'PB'
## - 'afex_aov' and 'mixed' objects can be passed to emmeans() for follow-up tests
## - NEWS: emmeans() for ANOVA models now uses model = 'multivariate' as default.
## - Get and set global package options with: afex_options()
## - Set orthogonal sum-to-zero contrasts globally: set_sum_contrasts()
## - For example analyses see: browseVignettes("afex")
## ************
##
## Attaching package: 'afex'
##
## The following object is masked from 'package:lme4':
##
##     lmer
```

```r
library(PairedData)
```

```
## Loading required package: MASS
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
##
## Loading required package: gld
## Loading required package: mvtnorm
## Loading required package: lattice
##
## Attaching package: 'PairedData'
##
## The following object is masked from 'package:Matrix':
##
##     summary
##
## The following object is masked from 'package:base':
##
##     summary
```

```r
library(multcompView)
library(lsmeans)
```

```
## Loading required package: emmeans
## The 'lsmeans' package is now basically a front end for 'emmeans'.
## Users are encouraged to switch the rest of the way.
## See help('transition') for more information, including how to
## convert old 'lsmeans' objects and scripts to work with 'emmeans'.
```

```r
library(depmixS4)
```

```
## Loading required package: nnet
## Loading required package: Rsolnp
## Loading required package: nlme
##
## Attaching package: 'nlme'
##
## The following object is masked from 'package:lme4':
##
##     lmList
##
## The following object is masked from 'package:dplyr':
##
##     collapse
```

```r
library(flextable)
```

```
##
## Attaching package: 'flextable'
##
## The following object is masked from 'package:purrr':
##
##     compose
##
## The following objects are masked from 'package:kableExtra':
##
##     as_image, footnote
```

```r
library(grid)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(forcats)
library(ggsignif)
library(magick)
```

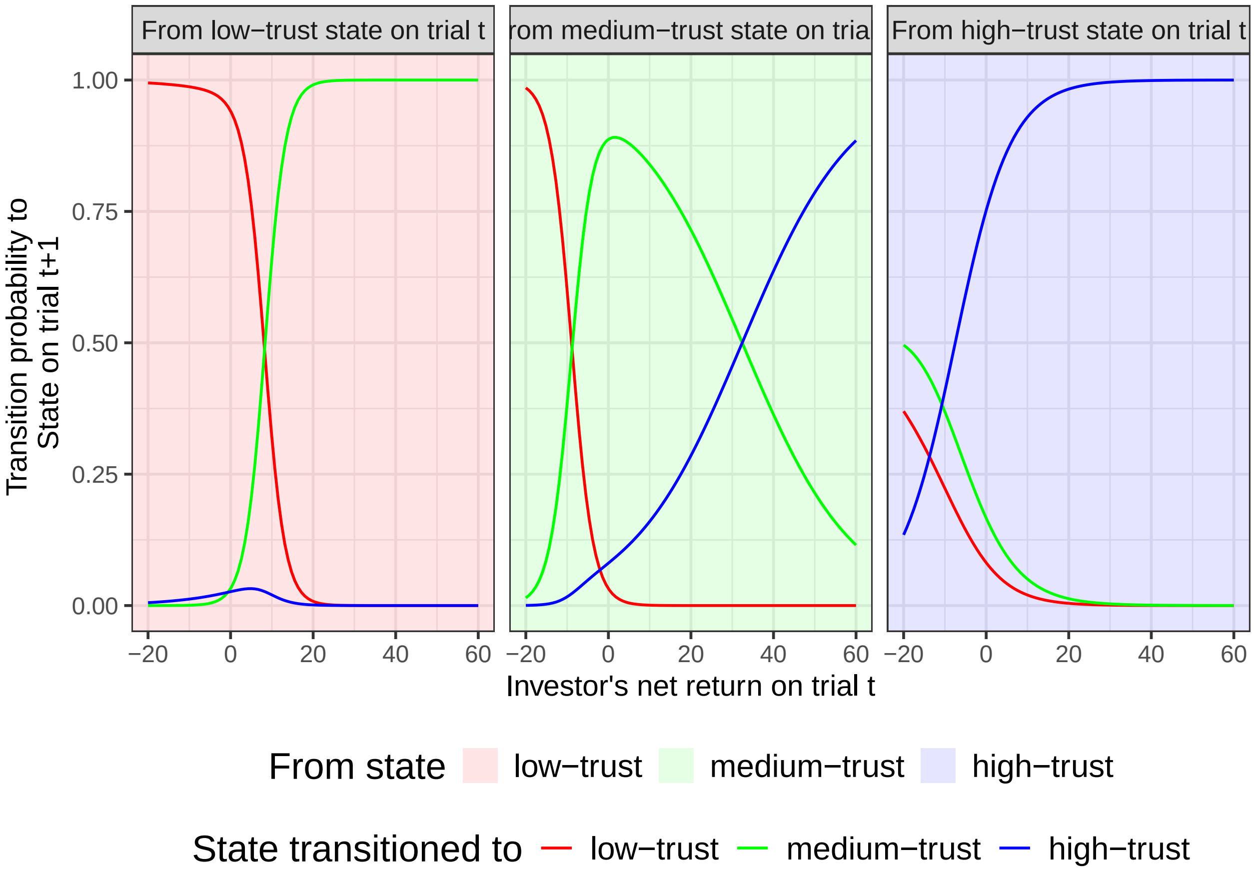
```
## Linking to ImageMagick 6.9.12.3
## Enabled features: cairo, fontconfig, freetype, heic, lcms, pango, raw, rsvg, webp
## Disabled features: fftw, ghostscript, x11
```

```
# Parameters for HMM human-like Transition Function
unhappy_pars <- rbind(c(0,0), c(-3.366027, 0.40910797 ), c(-3.572619,0.08137274))
neutral_pars <- rbind(c(0,0), c(3.3142637, 0.3763408), c(0.9169736, 0.4502838))
happy_pars   <- rbind(c(0,0), c(0.7134085, 0.02101626), c(2.2215478 ,0.16162964))

pars_inv <- list(unhappy_pars, neutral_pars, happy_pars)
```

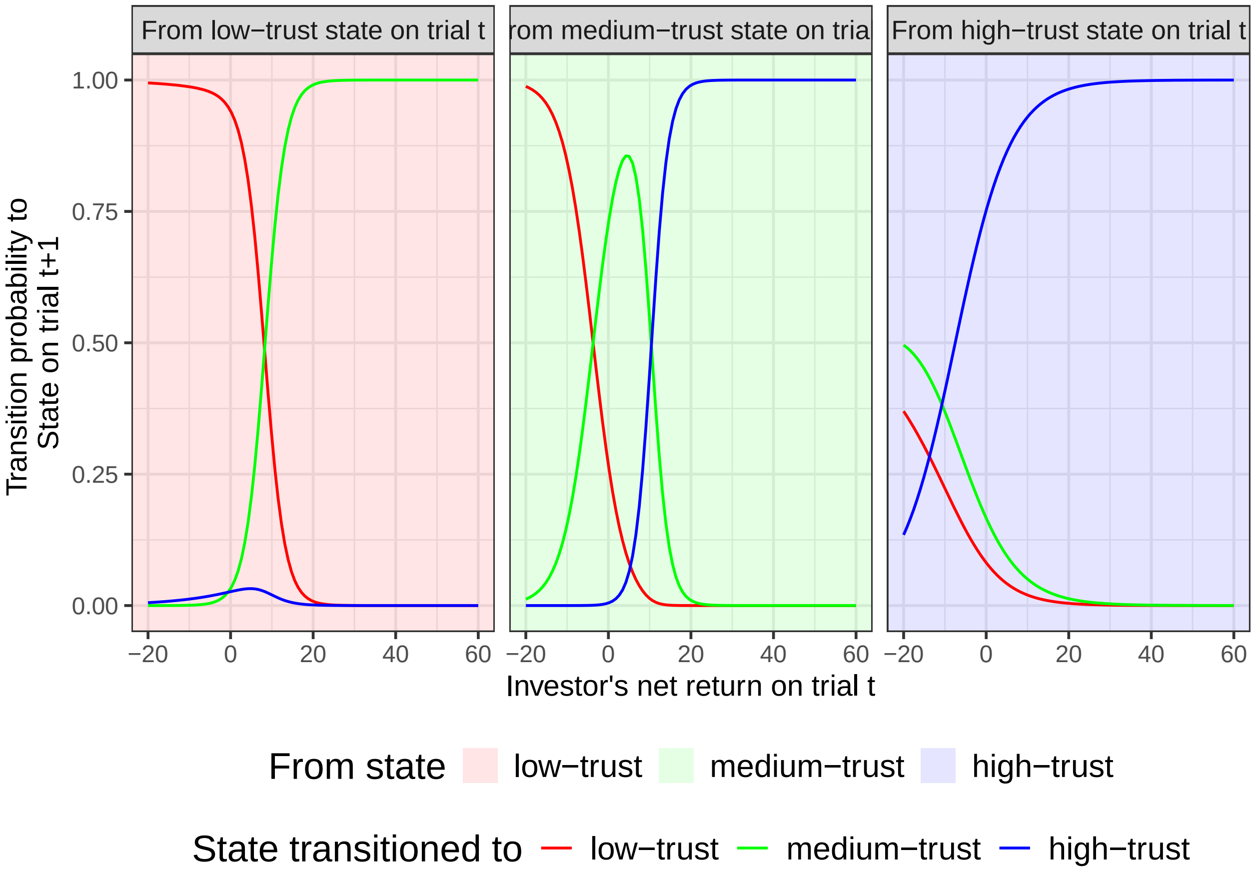
```
plot_HMM_transitions <- function(ns, pars_mat) {

  trans_prob <- data.frame(
    from = rep(1:ns, each=100*ns),
    to = rep(1:ns, each=100),
    ret = seq(-20,60,length=100),
    probs = 0
  )


  y <- matrix(0.0,ncol=ns, nrow=100)

  for(from in 1:ns) {
  pars <- matrix(pars_mat[[from]], ncol=2)
  # print(pars)

    for(to in 1:ns) {
        x <- trans_prob[trans_prob$from == from & trans_prob$to == to,"ret"]
        y[,to] <- exp(pars[to,1] + pars[to,2]*x)
    }
    y <- y/rowSums(y)


    for(to in 1:ns) {
      trans_prob$probs[trans_prob$from == from & trans_prob$to == to] <- y[,to]
    }
  }

  df <- as.data.frame(trans_prob) %>%
    mutate(from = recode(from, "1" = "low-trust", "2" = "medium-trust", "3" = "high-trust"),
           to = recode(to, "1" = "low-trust", "2" = "medium-trust", "3" = "high-trust") ) %>%
    mutate(across(from, factor, levels=c("low-trust","medium-trust","high-trust"))) %>%
    mutate(across(to, factor, levels=c("low-trust","medium-trust","high-trust")))


    # Create a separate data frame with the background colors
  bg_colors <- data.frame(
    from = factor(c("low-trust", "medium-trust", "high-trust"), levels=c("low-trust","medium-trust","hig
  )

  # plotting code...
  ggplot() +
    geom_rect(data = bg_colors, aes(xmin = -Inf, xmax = Inf, ymin = -Inf, ymax = Inf, fill = from), alph
```

```r
    geom_line(data = df, aes(x = ret, y = probs, colour = as.factor(to))) +
    facet_wrap(~from, labeller = labeller(from = function(x) paste("From", x, "state on trial t"))) +
    ylim(c(0,1)) +
    scale_fill_manual(values = c("low-trust" = "red", "medium-trust" = "green", "high-trust" = "blue"),
                    name = "From state") +  # Changed legend title for 'fill' here
    scale_color_manual(values = c("low-trust" = "red", "medium-trust" = "green", "high-trust" = "blue")
                       labels = c("low-trust", "medium-trust", "high-trust"),
                       name = "State transitioned to") +
    labs(x = "Investor's net return on trial t", y = "Transition probability to \nState on trial t+1", 
    theme_bw() +
    theme(legend.position = "bottom",
          legend.text = element_text(size = 12),
          legend.key.size = unit(1, 'lines'),
          legend.spacing.x = unit(0.1, 'in'),
          legend.title = element_text(size = 14),
          legend.margin = margin(t = 0.2, b = 0, unit = 'cm'),
          plot.margin = margin(t = 0, r = 0, b = 0, l = 0, unit = "cm"),
          strip.text = element_text(size = 10),
          legend.box = "vertical" # Arrange legends vertically
    ) +
    guides(fill = guide_legend(order = 1, title.position = "left", title.hjust = 0.3),
           color = guide_legend(order = 2, title.position = "left", title.hjust = 0.3))
}
```

```r
# Parameters for HMM human-like Transition Function
unhappy_pars <- rbind(c(0,0), c(-3.366027, 0.40910797 ), c(-3.572619,0.08137274))
neutral_pars <- rbind(c(0,0), c(3.3142637, 0.3763408), c(0.9169736, 0.4502838))
happy_pars   <- rbind(c(0,0), c(0.7134085, 0.02101626), c(2.2215478 ,0.16162964))

pars_inv <- list(unhappy_pars, neutral_pars, happy_pars)
plotInvTran <- plot_HMM_transitions(3, pars_inv)
print(plotInvTran)
```




```r
unhappy_pars_vol <- rbind(c(0,0), c(-3.366027, 0.40910797 ), c(-3.572619,0.08137274))
neutral_pars_vol <- rbind(c(0,0), c(1.0,0.27), c(-4, 0.75))
happy_pars_vol   <- rbind(c(0,0), c(0.7134085, 0.02101626), c(2.2215478 ,0.16162964))

pars_inv_vol <- list(unhappy_pars_vol, neutral_pars_vol, happy_pars_vol)
plotInvVol <- plot_HMM_transitions(3, pars_inv_vol)
print(plotInvVol)
```




```
#0.3763408
```

Results

```
final_data <- read_csv("data/final_data.csv")
```

```
## Rows: 9150 Columns: 22
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (10): playerId, id, gameNum.f, gameOpponent, Turing.choice, Turing.justi...
## dbl (10): roundNum, investment, return, return_pct, rating_cooperative, rati...
## lgl  (2): volatile_first, isLastRound
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
num_participants <- length(unique(final_data$playerId))
num_participants
```

```
## [1] 183
```

```r
# Count D factor groups
d_factor_counts <- final_data %>%
  dplyr::select(playerId, d_level) %>%
  distinct() %>%
  count(d_level) %>%
  print()
```

```
## # A tibble: 2 x 2
##   d_level     n
##   <chr>   <int>
## 1 high_D     91
## 2 low_D      92
```

```r
# Count order of play (volatile first vs HMM first)
order_counts <- final_data %>%
  dplyr::select(playerId, volatile_first) %>%
  distinct() %>%
  count(volatile_first) %>%
  print()
```

```
## # A tibble: 2 x 2
##   volatile_first     n
##   <lgl>          <int>
## 1 FALSE             88
## 2 TRUE              95
```

```r
final_data <- final_data %>% mutate(ret_pct_na = ifelse(investment==0,NA,return/(3*investment)),
                                    opponent.f = factor(gameOpponent, levels = c("AI_HMM", "AI_HMM_vol")
                                    investorState.f = factor(investorState, levels = c("unhappy","neutra
                                    d_level = as.factor(d_level),
                                    roundNum = as.numeric(as.character(roundNum)),
                                    inv_scaled = as.vector(scale(investment))) %>%
                              dplyr::select(-c("gameOpponent","investorState"))
```

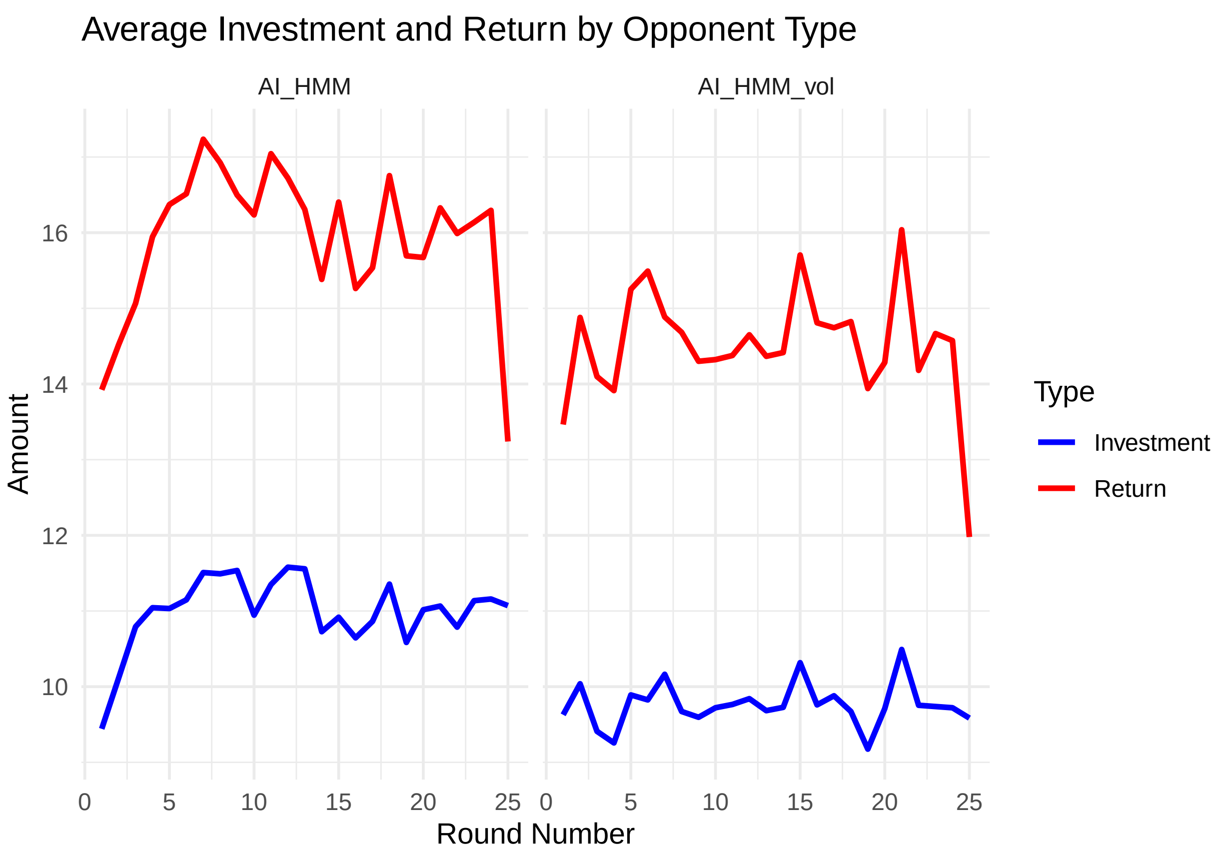
```r
# Calculate means by round and opponent type

# Calculate means
means_by_round_opp <- final_data %>%
  group_by(opponent.f, roundNum) %>%
  summarise(
    mean_investment = mean(investment),
    mean_return = mean(return)
  )
```

```
## `summarise()` has grouped output by 'opponent.f'. You can override using the
## `.groups` argument.
```

```r
# Create plot
ggplot(means_by_round_opp, aes(x = roundNum)) +
  geom_line(aes(y = mean_investment, color = "Investment"), size = 1) +
  geom_line(aes(y = mean_return, color = "Return"), size = 1) +
```

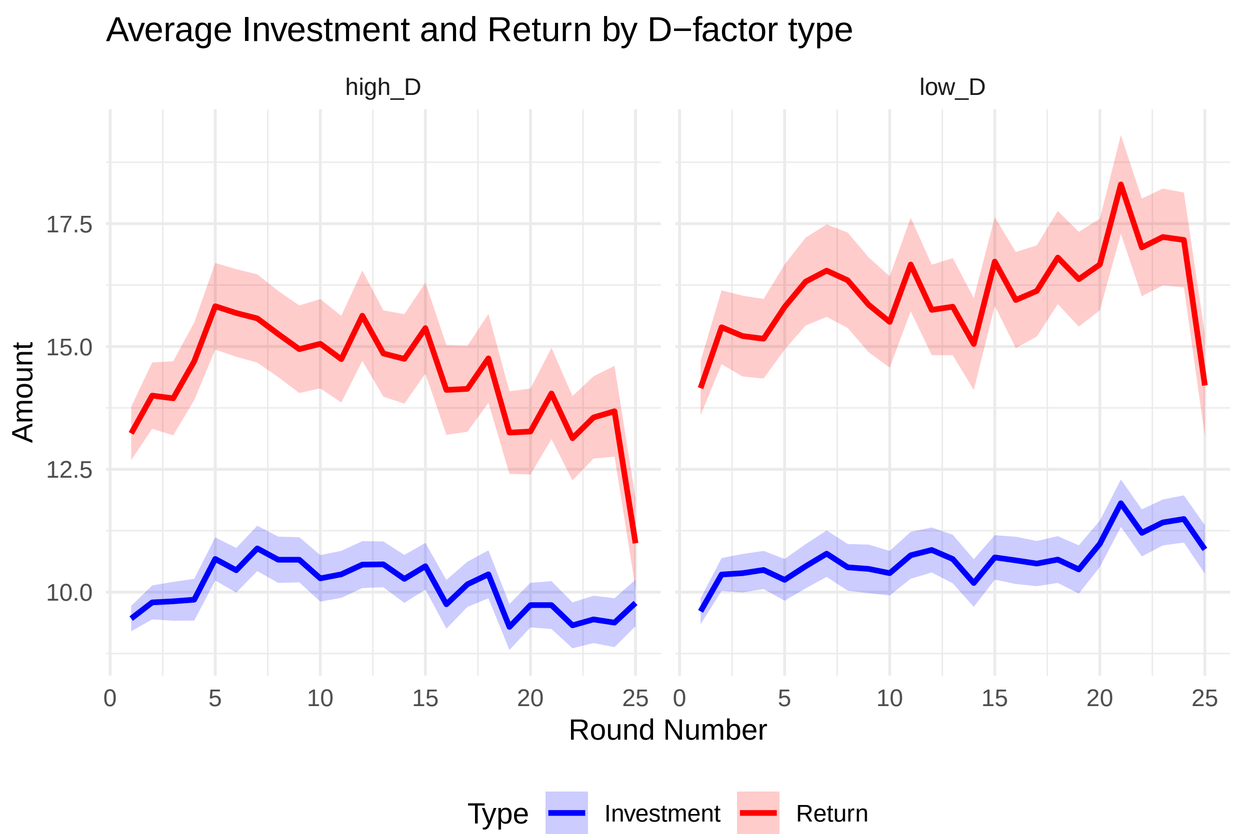
```
  facet_wrap(~opponent.f) +
  scale_color_manual(values = c("Investment" = "blue", "Return" = "red")) +
  labs(x = "Round Number",
       y = "Amount",
       color = "Type",
       title = "Average Investment and Return by Opponent Type") +
  theme_minimal()
```

```
## 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.
```




```
library(dplyr)
library(ggplot2)

# Calculate means and standard errors
means_by_round_d <- final_data %>%
  group_by(d_level, roundNum) %>%
  summarise(
    mean_investment = mean(investment),
    mean_return = mean(return),
    se_investment = sd(investment) / sqrt(n()),
    se_return = sd(return) / sqrt(n()),
```

```r
    .groups = 'drop'
  )

# Create plot with error bands
ggplot(means_by_round_d, aes(x = roundNum)) +
  # Add error bands
  geom_ribbon(aes(ymin = mean_investment - se_investment,
                  ymax = mean_investment + se_investment,
                  fill = "Investment"),
              alpha = 0.2) +
  geom_ribbon(aes(ymin = mean_return - se_return,
                  ymax = mean_return + se_return,
                  fill = "Return"),
              alpha = 0.2) +
  # Add lines
  geom_line(aes(y = mean_investment, color = "Investment"), size = 1) +
  geom_line(aes(y = mean_return, color = "Return"), size = 1) +
  # Facet by D-factor level
  facet_wrap(~d_level) +
  # Set colors
  scale_color_manual(values = c("Investment" = "blue", "Return" = "red")) +
  scale_fill_manual(values = c("Investment" = "blue", "Return" = "red")) +
  # Labels
  labs(x = "Round Number",
       y = "Amount",
       color = "Type",
       fill = "Type",
       title = "Average Investment and Return by D-factor type") +
  # Theme
  theme_minimal() +
  theme(legend.position = "bottom")
```




```
# # Count zero investments
# zero_counts <- final_data %>%
#   filter(investment == 0) %>%
#   group_by(opponent.f, d_level) %>%
#   summarise(
#     zero_count = n(),
#     n_players = n_distinct(playerId)
#   ) %>%
#   arrange(opponent.f, d_level)
#
# # Print results
# print("Number of zero investments by opponent type and D-factor:")
# print(zero_counts)
```

```
# Get last round returns
last_round_data <- final_data %>%
  filter(isLastRound == TRUE) %>%
  dplyr::select(playerId, d_level, investment,return, return_pct)

# Calculate means by D-factor group
means <- last_round_data %>%
  group_by(d_level) %>%
  summarise(
    n = n(),
    mean_return = mean(return),
    sd_return = sd(return)
  )
```

```r
print("Means by D-factor group:")
```

```
## [1] "Means by D-factor group:"
```

```r
print(means)
```

```
## # A tibble: 2 x 4
##   d_level     n mean_return sd_return
##   <fct>   <int>       <dbl>     <dbl>
## 1 high_D    182        11.0      12.4
## 2 low_D     184        14.2      14.2
```

```r
# For absolute returns
t.test(return ~ d_level, data = last_round_data)
```

```
##
##  Welch Two Sample t-test
##
## data:  return by d_level
## t = -2.3113, df = 358.42, p-value = 0.02138
## alternative hypothesis: true difference in means between group high_D and group low_D is not equal to
## 95 percent confidence interval:
##  -5.944979 -0.479054
## sample estimates:
## mean in group high_D  mean in group low_D
##             10.99451             14.20652
```

```r
# For return percentages
t.test(return_pct ~ d_level, data = last_round_data)
```

```
##
##  Welch Two Sample t-test
##
## data:  return_pct by d_level
## t = -2.1782, df = 363.98, p-value = 0.03004
## alternative hypothesis: true difference in means between group high_D and group low_D is not equal to
## 95 percent confidence interval:
##  -0.115721072 -0.005909605
## sample estimates:
## mean in group high_D  mean in group low_D
##            0.3116177            0.3724330
```

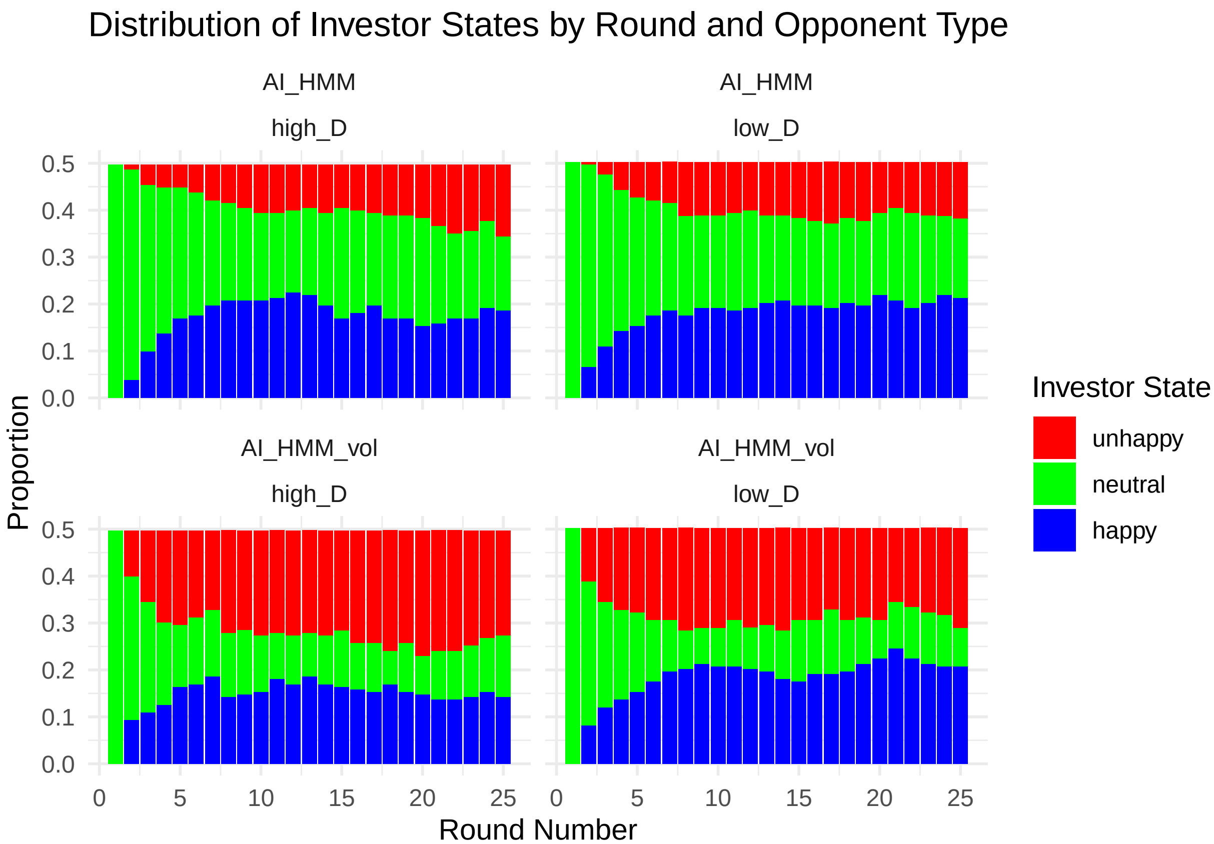
```r
# Mann-Whitney U tests (non-parametric alternative)
# For absolute returns
wilcox.test(return ~ d_level, data = last_round_data)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  return by d_level
## W = 14504, p-value = 0.02505
## alternative hypothesis: true location shift is not equal to 0
```

```r
# For return percentages
 wilcox.test(return_pct ~ d_level, data = last_round_data)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  return_pct by d_level
## W = 14445, p-value = 0.02139
## alternative hypothesis: true location shift is not equal to 0
```

```r
# Calculate proportions of states for each round and opponent
state_props <- final_data %>%
  group_by(opponent.f, roundNum, investorState.f, d_level) %>%
  summarise(count = n(), .groups = 'drop') %>%
  group_by(opponent.f, roundNum) %>%
  mutate(proportion = count / sum(count))

# Create stacked bar plot
ggplot(state_props, aes(x = roundNum, y = proportion, fill = investorState.f)) +
  geom_bar(stat = "identity") +
  facet_wrap(~opponent.f*d_level) +
  scale_fill_manual(values = c("unhappy" = "red", "neutral" = "green", "happy" = "blue")) +
  labs(x = "Round Number",
       y = "Proportion",
       title = "Distribution of Investor States by Round and Opponent Type",
       fill = "Investor State") +
  theme_minimal()
```



Payoff regression

```
# Reshape data to get one row per game per participant
payoff_data <- final_data %>%
  dplyr::select(playerId, d_level, opponent.f, gameNum.f, volatile_first, payoffTrust1, payoffTrust2) %>%
  distinct() %>%
   mutate(
      payoff = case_when(
        gameNum.f == "first game" ~ payoffTrust1,
        gameNum.f == "second game" ~ payoffTrust2
      )
    ) %>%
    dplyr::select(-payoffTrust1, -payoffTrust2)  # Remove unused columns


# fit lmem
mod_payoffs <- mixed( payoff ~ opponent.f*d_level*volatile_first + (1| playerId), payoff_data, REML= TRU
```

```
## Contrasts set to contr.sum for the following variables: opponent.f, d_level, playerId
```

```
## Fitting one lmer() model. [DONE]
## Calculating p-values. [DONE]
```

```r
summary(mod_payoffs)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: payoff ~ opponent.f * d_level * volatile_first + (1 | playerId)
##    Data: data
##
## REML criterion at convergence: 4411.9
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.28404 -0.77333 -0.04065  0.62980  2.66922
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  playerId (Intercept)   590.8   24.31
##  Residual             11150.7  105.60
## Number of obs: 366, groups:  playerId, 183
##
## Fixed effects:
##                                         Estimate Std. Error      df t value
## (Intercept)                              409.875      8.390 179.000  48.851
## opponent.f1                               31.340      7.978 179.000   3.928
## d_level1                                 -11.918      8.390 179.000  -1.420
## volatile_firstTRUE                       -21.800     11.647 179.000  -1.872
## opponent.f1:d_level1                       7.426      7.978 179.000   0.931
## opponent.f1:volatile_firstTRUE            -4.498     11.075 179.000  -0.406
## d_level1:volatile_firstTRUE               19.513     11.647 179.000   1.675
## opponent.f1:d_level1:volatile_firstTRUE    5.857     11.075 179.000   0.529
##                                         Pr(>|t|)
## (Intercept)                              < 2e-16 ***
## opponent.f1                             0.000122 ***
## d_level1                                0.157230
## volatile_firstTRUE                      0.062886 .
## opponent.f1:d_level1                    0.353242
## opponent.f1:volatile_firstTRUE          0.685106
## d_level1:volatile_firstTRUE             0.095621 .
## opponent.f1:d_level1:volatile_firstTRUE 0.597546
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) oppn.1 d_lvl1 v_TRUE op.1:_1 o.1:_T d_1:_T
## opponent.f1  0.000
## d_level1    -0.068  0.000
## vltl_frTRUE -0.720  0.000  0.049
## oppnnt.1:_1  0.000 -0.068  0.000  0.000
## opp.1:_TRUE  0.000 -0.720  0.000  0.000  0.049
## d_lv1:_TRUE  0.049  0.000 -0.720  0.000  0.000   0.000
## o.1:_1:_TRU  0.000  0.049  0.000  0.000 -0.720   0.000  0.000
```

```r
mod_payoffs_lm <- lm( payoff ~ opponent.f*d_level*volatile_first,payoff_data)
summary(mod_payoffs_lm)
```

```
##
## Call:
## lm(formula = payoff ~ opponent.f * d_level * volatile_first,
##     data = payoff_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -259.707  -85.707   -4.368   70.961  305.961
##
## Coefficients:
##                                                Estimate Std. Error
## (Intercept)                                    436.7234    15.8057
## opponent.fAI_HMM_vol                           -77.5319    22.3526
## d_levellow_D                                     8.9839    23.1560
## volatile_firstTRUE                              -0.9279    22.7304
## opponent.fAI_HMM_vol:d_levellow_D               29.7026    32.7475
## opponent.fAI_HMM_vol:volatile_firstTRUE         -2.7181    32.1457
## d_levellow_D:volatile_firstTRUE                -50.7402    32.1446
## opponent.fAI_HMM_vol:d_levellow_D:volatile_firstTRUE  23.4297    45.4594
##                                                t value Pr(>|t|)
## (Intercept)                                     27.631  < 2e-16 ***
## opponent.fAI_HMM_vol                            -3.469 0.000587 ***
## d_levellow_D                                     0.388 0.698266
## volatile_firstTRUE                              -0.041 0.967459
## opponent.fAI_HMM_vol:d_levellow_D                0.907 0.365006
## opponent.fAI_HMM_vol:volatile_firstTRUE         -0.085 0.932662
## d_levellow_D:volatile_firstTRUE                 -1.578 0.115335
## opponent.fAI_HMM_vol:d_levellow_D:volatile_firstTRUE   0.515 0.606592
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 108.4 on 358 degrees of freedom
## Multiple R-squared:  0.09217,    Adjusted R-squared:  0.07442
## F-statistic: 5.193 on 7 and 358 DF,  p-value: 1.189e-05
```

```r
# Get estimated marginal means
# First for opponent type
opponent_emm <- emmeans(mod_payoffs, ~ opponent.f)
```

```
## NOTE: Results may be misleading due to involvement in interactions
```

```r
pairs(opponent_emm)
```

```
##  contrast             estimate   SE  df t.ratio p.value
##  AI_HMM - AI_HMM_vol      58.2 11.1 179   5.253  <.0001
##
## Results are averaged over the levels of: d_level, volatile_first
## Degrees-of-freedom method: kenward-roger
```

```r
# For order (volatile_first)
order_emm <- emmeans(mod_payoffs, ~ volatile_first)
```

```
## NOTE: Results may be misleading due to involvement in interactions
```

```r
pairs(order_emm)
```

```
##  contrast     estimate   SE  df t.ratio p.value
##  FALSE - TRUE     21.8 11.6 179   1.872  0.0629
##
## Results are averaged over the levels of: opponent.f, d_level
## Degrees-of-freedom method: kenward-roger
```

```r
# For d_type
dtype_emm <- emmeans(mod_payoffs, ~ d_level)
```

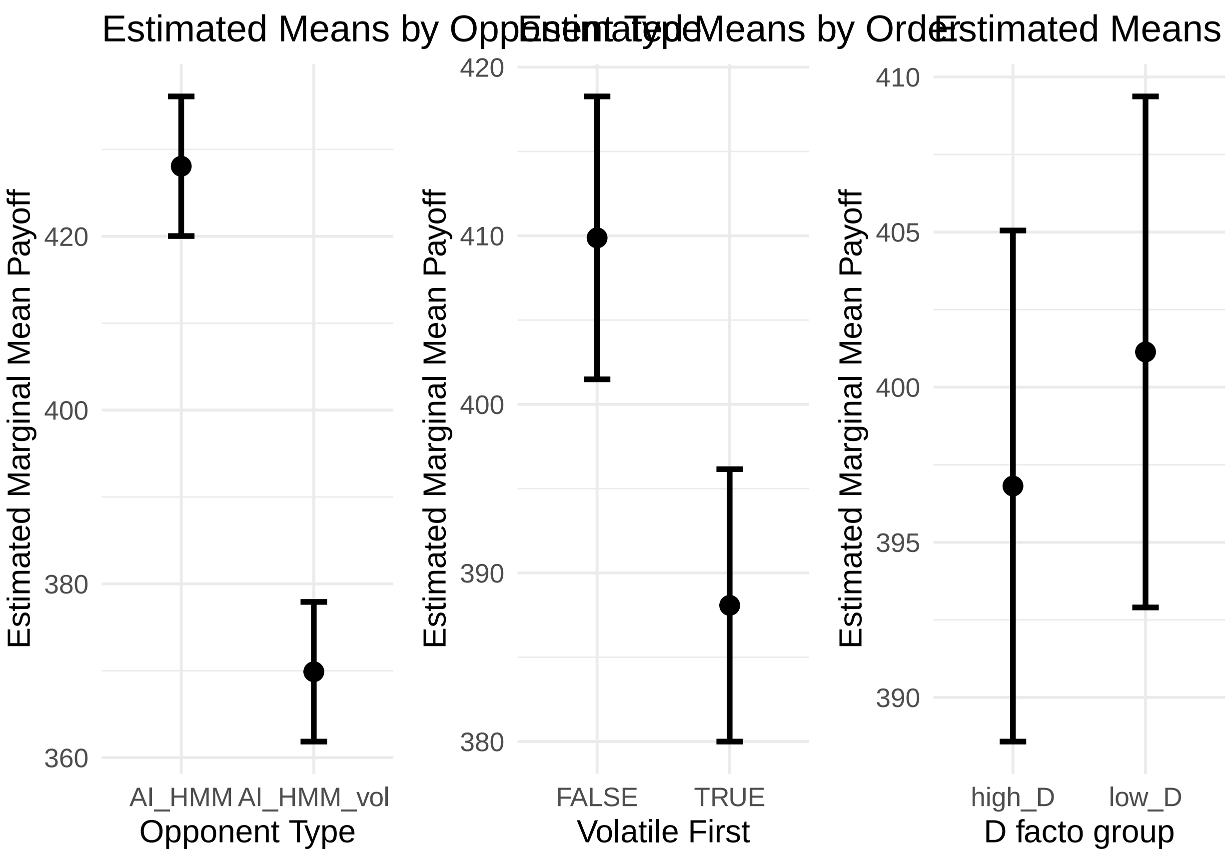
```
## NOTE: Results may be misleading due to involvement in interactions
```

```r
pairs(dtype_emm)
```

```
##  contrast       estimate   SE  df t.ratio p.value
##  high_D - low_D    -4.32 11.6 179  -0.371  0.7110
##
## Results are averaged over the levels of: opponent.f, volatile_first
## Degrees-of-freedom method: kenward-roger
```

```r
# Convert EMMs to data frames for plotting
opponent_plot_data <- as.data.frame(opponent_emm)
order_plot_data <- as.data.frame(order_emm)
dtype_plot_data <- as.data.frame(dtype_emm)

# Plot for opponent effect
p1 <- ggplot(opponent_plot_data, aes(x = opponent.f, y = emmean)) +
  geom_point(size = 3) +
  geom_errorbar(aes(ymin = emmean - SE, ymax = emmean + SE),
                width = 0.2, size = 1) +
  labs(x = "Opponent Type", y = "Estimated Marginal Mean Payoff",
       title = "Estimated Means by Opponent Type") +
  theme_minimal() +
  theme(text = element_text(size = 12))

# Plot for order effect
p2 <- ggplot(order_plot_data, aes(x = volatile_first, y = emmean)) +
  geom_point(size = 3) +
  geom_errorbar(aes(ymin = emmean - SE, ymax = emmean + SE),
                width = 0.2, size = 1) +
  labs(x = "Volatile First", y = "Estimated Marginal Mean Payoff",
       title = "Estimated Means by Order") +
  theme_minimal() +
  theme(text = element_text(size = 12))
```

```r
# Plot for d_type
p3 <- ggplot(dtype_plot_data, aes(x = d_level, y = emmean)) +
  geom_point(size = 3) +
  geom_errorbar(aes(ymin = emmean - SE, ymax = emmean + SE),
                width = 0.2, size = 1) +
  labs(x = "D facto group", y = "Estimated Marginal Mean Payoff",
       title = "Estimated Means by D-factor group") +
  theme_minimal() +
  theme(text = element_text(size = 12))

# Display plots side by side
library(gridExtra)
grid.arrange(p1, p2, p3, ncol = 3)
```



Percentage returns model

```r
mod_returns_pct <- mixed( ret_pct_na ~ opponent.f*inv_scaled*d_level*volatile_first + (1+ opponent.f| pl
```

```
## Contrasts set to contr.sum for the following variables: opponent.f, d_level, playerId

## Warning: Due to missing values, reduced number of observations to 8875

## Numerical variables NOT centered on 0: inv_scaled
## If in interactions, interpretation of lower order (e.g., main) effects difficult.
```

```
## Fitting one lmer() model. [DONE]
## Calculating p-values. [DONE]
```

```r
summary(mod_returns_pct)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: ret_pct_na ~ opponent.f * inv_scaled * d_level * volatile_first +
##     (1 + opponent.f | playerId)
##    Data: data
##
## REML criterion at convergence: -7174
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -4.2088 -0.2896  0.0367  0.3586  5.5931
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  playerId (Intercept) 0.01722  0.13124
##           opponent.f1 0.00109  0.03302  -0.12
##  Residual             0.02331  0.15267
## Number of obs: 8875, groups:  playerId, 183
##
## Fixed effects:
##                                                   Estimate Std. Error
## (Intercept)                                      4.423e-01  1.422e-02
## opponent.f1                                      8.548e-04  4.259e-03
## inv_scaled                                       1.778e-02  2.922e-03
## d_level1                                        -9.897e-03  1.422e-02
## volatile_firstTRUE                               1.109e-02  1.974e-02
## opponent.f1:inv_scaled                           3.093e-03  2.867e-03
## opponent.f1:d_level1                            -1.228e-03  4.259e-03
## inv_scaled:d_level1                             -1.204e-02  2.922e-03
## opponent.f1:volatile_firstTRUE                  -1.594e-04  5.907e-03
## inv_scaled:volatile_firstTRUE                   -4.607e-04  4.082e-03
## d_level1:volatile_firstTRUE                      6.698e-03  1.974e-02
## opponent.f1:inv_scaled:d_level1                  4.174e-03  2.867e-03
## opponent.f1:inv_scaled:volatile_firstTRUE       -4.070e-03  3.988e-03
## opponent.f1:d_level1:volatile_firstTRUE         -1.969e-03  5.907e-03
## inv_scaled:d_level1:volatile_firstTRUE           6.687e-03  4.082e-03
## opponent.f1:inv_scaled:d_level1:volatile_firstTRUE  3.062e-04  3.988e-03
##                                                         df t value Pr(>|t|)
## (Intercept)                                      1.772e+02  31.092  < 2e-16
## opponent.f1                                      1.756e+02   0.201    0.841
## inv_scaled                                       8.234e+03   6.085 1.22e-09
## d_level1                                         1.772e+02  -0.696    0.487
## volatile_firstTRUE                               1.772e+02   0.562    0.575
## opponent.f1:inv_scaled                           6.814e+03   1.079    0.281
## opponent.f1:d_level1                             1.756e+02  -0.288    0.773
## inv_scaled:d_level1                              8.234e+03  -4.119 3.84e-05
## opponent.f1:volatile_firstTRUE                   1.749e+02  -0.027    0.979
## inv_scaled:volatile_firstTRUE                    8.292e+03  -0.113    0.910
## d_level1:volatile_firstTRUE                      1.772e+02   0.339    0.735
```

```
## opponent.f1:inv_scaled:d_level1                      6.814e+03   1.456    0.146
## opponent.f1:inv_scaled:volatile_firstTRUE            6.532e+03  -1.021    0.308
## opponent.f1:d_level1:volatile_firstTRUE              1.749e+02  -0.333    0.739
## inv_scaled:d_level1:volatile_firstTRUE               8.292e+03   1.638    0.101
## opponent.f1:inv_scaled:d_level1:volatile_firstTRUE  6.532e+03   0.077    0.939
##
## (Intercept)                                         ***
## opponent.f1
## inv_scaled                                          ***
## d_level1
## volatile_firstTRUE
## opponent.f1:inv_scaled
## opponent.f1:d_level1
## inv_scaled:d_level1                                 ***
## opponent.f1:volatile_firstTRUE
## inv_scaled:volatile_firstTRUE
## d_level1:volatile_firstTRUE
## opponent.f1:inv_scaled:d_level1
## opponent.f1:inv_scaled:volatile_firstTRUE
## opponent.f1:d_level1:volatile_firstTRUE
## inv_scaled:d_level1:volatile_firstTRUE
## opponent.f1:inv_scaled:d_level1:volatile_firstTRUE
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Correlation matrix not shown by default, as p = 16 > 12.
## Use print(x, correlation=TRUE)  or
##     vcov(x)        if you need it
```

```r
# Get estimated means
emm_inv <- emmeans(mod_returns_pct,
                  ~ d_level | inv_scaled,
                  at = list(inv_scaled = c(-1, 0, 1)))
```

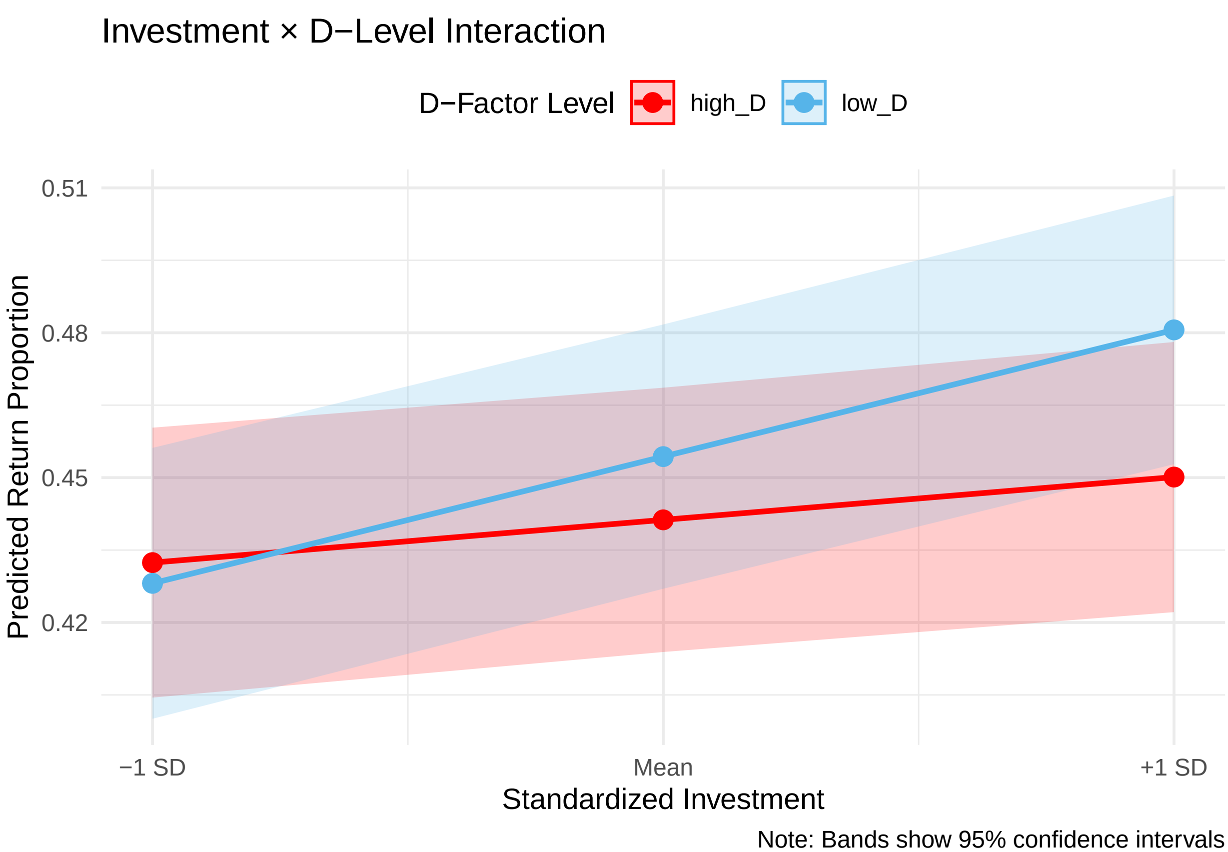
```
## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'pbkrtest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(pbkrtest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.
```

```
## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'lmerTest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(lmerTest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.
```

```
## NOTE: Results may be misleading due to involvement in interactions
```

```r
# Convert to data frame
plot_data <- as.data.frame(emm_inv)

# Create plot with 95% CI instead of SE
ggplot(plot_data, aes(x = inv_scaled, y = emmean, color = d_level)) +
```

```r
  # Add 95% CI bands
  geom_ribbon(aes(ymin = emmean - SE*1.96,
                  ymax = emmean + SE*1.96,
                  fill = d_level,
                  color = NULL),
              alpha = 0.2) +
  geom_line(linewidth = 1) +
  geom_point(size = 3) +
  scale_x_continuous(breaks = c(-1, 0, 1),
                     labels = c("-1 SD", "Mean", "+1 SD")) +
  scale_color_manual(values = c("high_D" = "red", "low_D" = "#56B4E9"),
                     name = "D-Factor Level") +
  scale_fill_manual(values = c("high_D" = "red", "low_D" = "#56B4E9"),
                     name = "D-Factor Level") +
  labs(x = "Standardized Investment",
       y = "Predicted Return Proportion",
       title = "Investment × D-Level Interaction",
       caption = "Note: Bands show 95% confidence intervals") +
  theme_minimal() +
  theme(legend.position = "top")
```

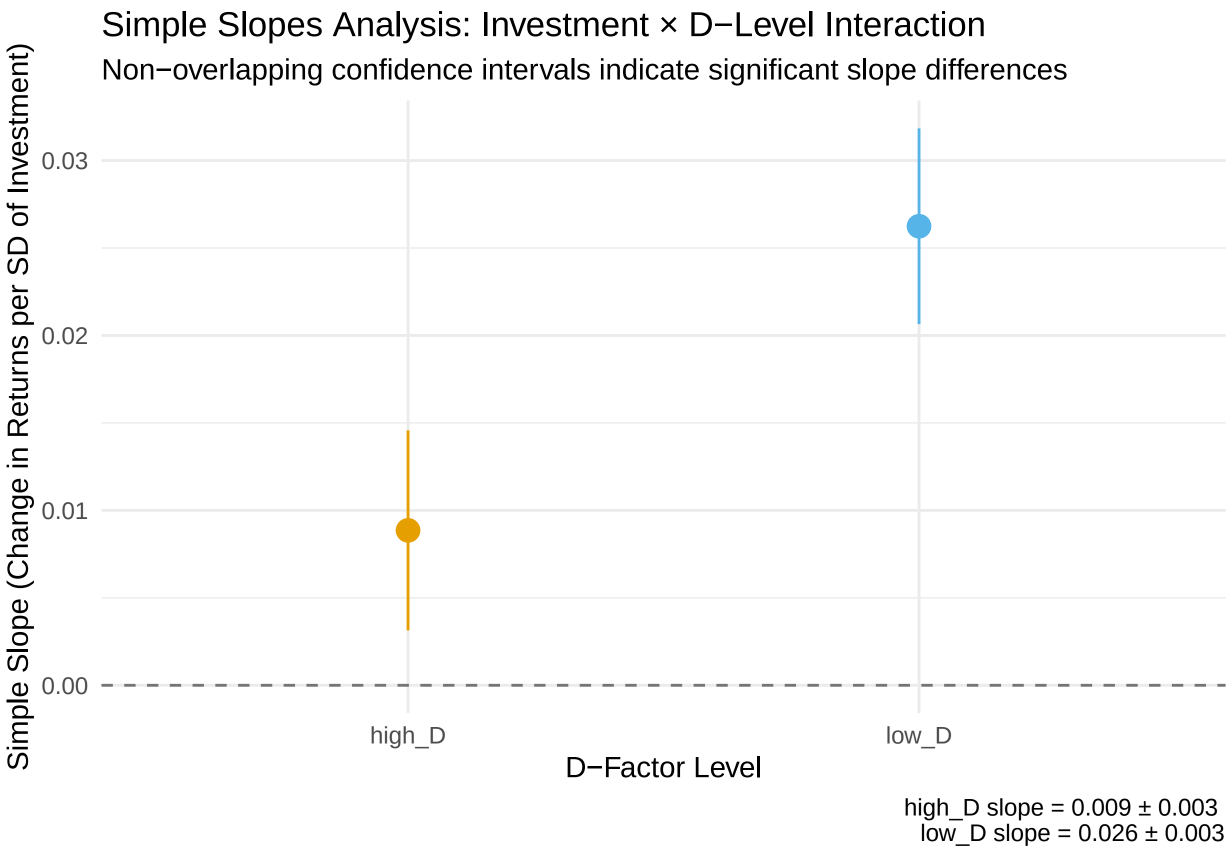



```r
# Get simple slopes for each d_level
slopes <- emtrends(mod_returns_pct, ~ d_level, var = "inv_scaled")
```

```
## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'pbkrtest.limit = 8875' (or larger)
```

```
## [or, globally, 'set emm_options(pbkrtest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.

## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'lmerTest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(lmerTest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.

## NOTE: Results may be misleading due to involvement in interactions
```

```r
slopes_df <- as.data.frame(slopes)

# Create data for plotting slope lines
slope_lines <- slopes_df %>%
  group_by(d_level) %>%
  summarise(
    slope = inv_scaled.trend,
    se = SE,
    ci_lower = slope - (1.96 * se),
    ci_upper = slope + (1.96 * se)
  )

# Create plot
ggplot(slope_lines, aes(x = d_level, y = slope, color = d_level)) +
  # Add error bars for 95% CI
  geom_pointrange(aes(ymin = ci_lower, ymax = ci_upper),
                  size = 1,
                  fatten = 3) +
  # Customize appearance
  scale_color_manual(values = c("high_D" = "#E69F00", "low_D" = "#56B4E9")) +
  labs(x = "D-Factor Level",
       y = "Simple Slope (Change in Returns per SD of Investment)",
       title = "Simple Slopes Analysis: Investment × D-Level Interaction",
       subtitle = "Non-overlapping confidence intervals indicate significant slope differences",
       caption = paste("high_D slope =", round(slope_lines$slope[1], 3),
                       "±", round(slope_lines$se[1], 3),
                       "\nlow_D slope =", round(slope_lines$slope[2], 3),
                       "±", round(slope_lines$se[2], 3))) +
  theme_minimal() +
  theme(legend.position = "none") +
  # Add horizontal line at y = 0 for reference
  geom_hline(yintercept = 0, linetype = "dashed", alpha = 0.5)
```




```r
# Print the statistical test of slope differences
print(slopes)
```

```
##   d_level inv_scaled.trend      SE  df asymp.LCL asymp.UCL
##   high_D           0.00886 0.00292 Inf   0.00314    0.0146
##   low_D            0.02624 0.00285 Inf   0.02065    0.0318
##
## Results are averaged over the levels of: opponent.f, volatile_first
## Degrees-of-freedom method: asymptotic
## Confidence level used: 0.95
```

```r
library(ggsignif)  # For adding significance bars
library(emmeans)

# Helper function to add significance labels
add_significance_label <- function(p_value) {
  if(p_value < 0.001) return("***")
  else if(p_value < 0.01) return("**")
  else if(p_value < 0.05) return("*")
  else return("ns")
}

# 1. Game Opponent × D-Level interaction plot
plot_opponent_dlevel_sig <- function(model) {
  # Get emmeans
  emm_od <- emmeans(model, ~ opponent.f| d_level, pbkrtest.limit = 3200)
```

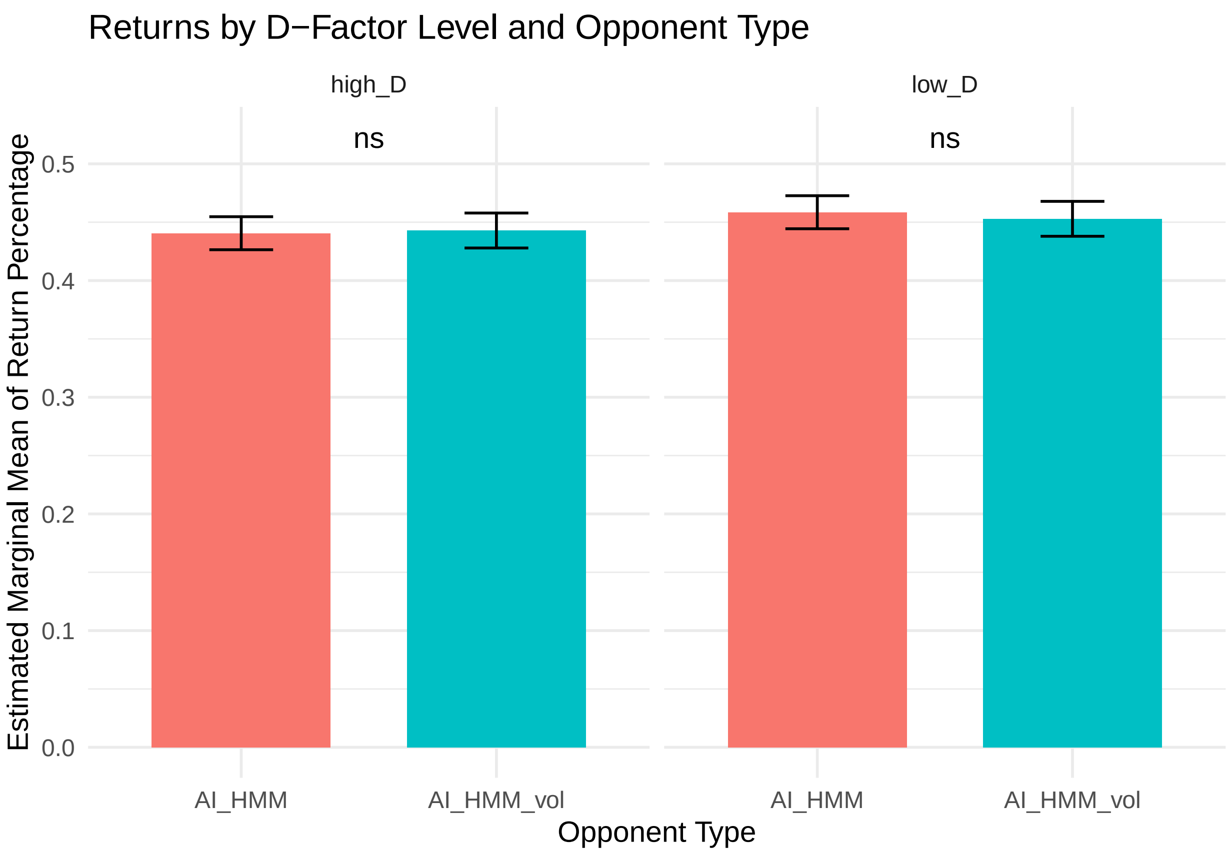
```r
  # Get contrasts and convert to data frame
  pairs_od <- pairs(emm_od)
  pairs_df <- as.data.frame(pairs_od)

  # Convert emmeans to data frame for plotting
  plot_data <- as.data.frame(emm_od)

  # Calculate y positions for significance bars
  max_y <- max(plot_data$emmean + plot_data$SE)
  sig_y_pos <- max_y + 0.05

  # Create annotation data
  anno_data <- data.frame(
    d_level = unique(plot_data$d_level),
    y_pos = sig_y_pos,
    label = sapply(split(pairs_df$p.value, pairs_df$d_level), function(p) add_significance_label(p[1]))
  )

  # Create plot
  p <- ggplot(plot_data, aes(x = opponent.f, y = emmean, fill = opponent.f)) +
    geom_bar(stat = "identity", position = position_dodge(), width = 0.7) +
    geom_errorbar(aes(ymin = emmean - SE, ymax = emmean + SE),
                  position = position_dodge(0.9), width = 0.25) +
    facet_wrap(~d_level) +
    geom_text(data = anno_data, aes(x = 1.5, y = y_pos, label = label),
              inherit.aes = FALSE) +
    labs(title = "Returns by D-Factor Level and Opponent Type",
         x = "Opponent Type",
         y = "Estimated Marginal Mean of Return Percentage") +
    theme_minimal() +
    theme(legend.position = "none")

  return(list(
    plot = p,
    contrasts = pairs_df
  ))
}

results_od <- plot_opponent_dlevel_sig(mod_returns_pct)
```

```
## Note: D.f. calculations have been disabled because the number of observations exceeds 3200.
## To enable adjustments, add the argument 'pbkrtest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(pbkrtest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.
```

```
## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'lmerTest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(lmerTest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.
```

```
## NOTE: Results may be misleading due to involvement in interactions
```

```r
print(results_od$plot)
```




```r
print(results_od$contrasts)
```

```
##              contrast d_level     estimate          SE  df    z.ratio   p.value
## 1 AI_HMM - AI_HMM_vol  high_D -0.002314350 0.008366539 Inf -0.2766197 0.7820721
## 2 AI_HMM - AI_HMM_vol   low_D  0.005635005 0.008331518 Inf  0.6763479 0.4988198
```

```r
library(emmeans)
library(ggplot2)
library(dplyr)

plot_opponent_volatile_sig <- function(model) {
  # Get emmeans
  emm_ov <- emmeans(model, ~ opponent.f| volatile_first)

  # Get contrasts within each volatile_first condition
  pairs_ov <- pairs(emm_ov)
  pairs_df <- as.data.frame(pairs_ov)

  # Convert emmeans to data frame for plotting
  plot_data <- as.data.frame(emm_ov)

  # Calculate y positions for significance bars
  max_y <- max(plot_data$emmean + plot_data$SE)
```

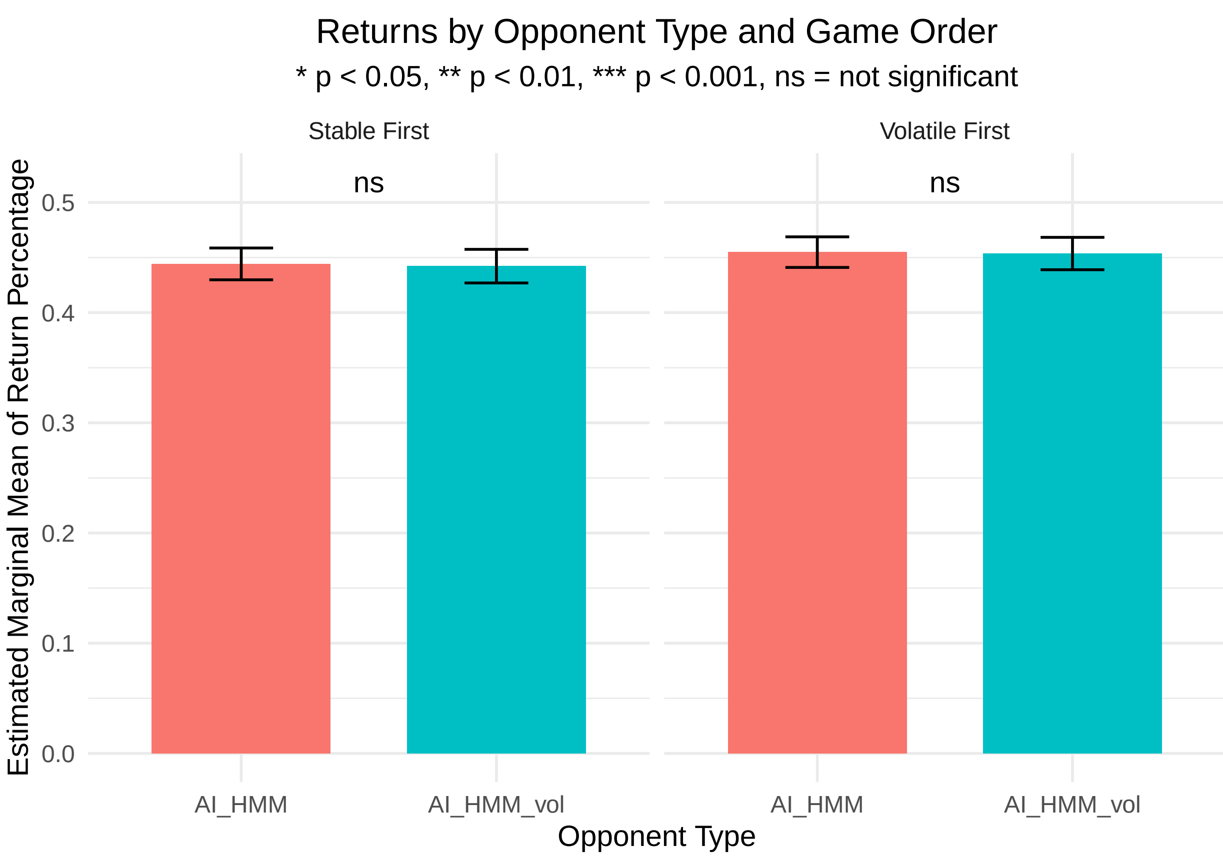
```r
  sig_y_pos <- max_y + 0.05

  # Create annotation data
  anno_data <- data.frame(
    volatile_first = unique(plot_data$volatile_first),
    y_pos = sig_y_pos,
    label = sapply(split(pairs_df$p.value, pairs_df$volatile_first),
                   function(p) {
                     p_val <- p[1]
                     if(p_val < 0.001) return("***")
                     else if(p_val < 0.01) return("**")
                     else if(p_val < 0.05) return("*")
                     else return("ns")
                   })
  )

  # Create plot
  p <- ggplot(plot_data, aes(x = opponent.f, y = emmean, fill = opponent.f)) +
    geom_bar(stat = "identity", position = position_dodge(), width = 0.7) +
    geom_errorbar(aes(ymin = emmean - SE, ymax = emmean + SE),
                  position = position_dodge(0.9), width = 0.25) +
    facet_wrap(~volatile_first, labeller = labeller(volatile_first = c("FALSE" = "Stable First", "TRUE"
    geom_text(data = anno_data, aes(x = 1.5, y = y_pos, label = label),
              inherit.aes = FALSE) +
    labs(title = "Returns by Opponent Type and Game Order",
         subtitle = "* p < 0.05, ** p < 0.01, *** p < 0.001, ns = not significant",
         x = "Opponent Type",
         y = "Estimated Marginal Mean of Return Percentage") +
    theme_minimal() +
    theme(legend.position = "none",
          plot.title = element_text(hjust = 0.5),
          plot.subtitle = element_text(hjust = 0.5))

  # Also look at between-order differences for each opponent
  emm_vo <- emmeans(model, ~ volatile_first | opponent.f)
  pairs_vo <- pairs(emm_vo)

  return(list(
    plot = p,
    within_order_contrasts = pairs_df,
    between_order_contrasts = pairs_vo
  ))
}

# Example usage:
results_ov <- plot_opponent_volatile_sig(mod_returns_pct)
```

```
## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'pbkrtest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(pbkrtest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.
```

```
## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
```

```
## To enable adjustments, add the argument 'lmerTest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(lmerTest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.

## NOTE: Results may be misleading due to involvement in interactions

## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'pbkrtest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(pbkrtest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.

## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'lmerTest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(lmerTest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.

## NOTE: Results may be misleading due to involvement in interactions
```

```r
print(results_ov$plot)
```




```r
print(results_ov$within_order_contrasts)  # Shows opponent differences within each order
```

```
##                contrast volatile_first    estimate          SE  df   z.ratio
## 1 AI_HMM - AI_HMM_vol          FALSE 0.002031804 0.008507814 Inf 0.2388162
```

```
## 2 AI_HMM - AI_HMM_vol           TRUE 0.001288851 0.008187201 Inf 0.1574227
##     p.value
## 1 0.8112481
## 2 0.8749117
```

```r
print(results_ov$between_order_contrasts)  # Shows order differences within each opponent
```

```
## opponent.f = AI_HMM:
##  contrast     estimate     SE  df z.ratio p.value
##  FALSE - TRUE  -0.0107 0.0200 Inf  -0.534  0.5932
##
## opponent.f = AI_HMM_vol:
##  contrast     estimate     SE  df z.ratio p.value
##  FALSE - TRUE  -0.0114 0.0212 Inf  -0.540  0.5892
##
## Results are averaged over the levels of: d_level
## Degrees-of-freedom method: asymptotic
```

```r
  # 1. For each opponent, compare high vs low D
  emm_d <- emmeans(mod_returns_pct, ~ d_level | opponent.f)
```

```
## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'pbkrtest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(pbkrtest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.

## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'lmerTest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(lmerTest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.

## NOTE: Results may be misleading due to involvement in interactions
```

```r
  d_pairs <- pairs(emm_d)
  d_pairs
```

```
## opponent.f = AI_HMM:
##  contrast       estimate     SE  df z.ratio p.value
##  high_D - low_D   -0.018 0.0200 Inf  -0.898  0.3693
##
## opponent.f = AI_HMM_vol:
##  contrast       estimate     SE  df z.ratio p.value
##  high_D - low_D   -0.010 0.0212 Inf  -0.473  0.6359
##
## Results are averaged over the levels of: volatile_first
## Degrees-of-freedom method: asymptotic
```

```r
  # 2. For each opponent, compare volatile first vs stable first
  emm_v <- emmeans(mod_returns_pct, ~ volatile_first | opponent.f)
```

```
## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'pbkrtest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(pbkrtest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.

## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'lmerTest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(lmerTest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.

## NOTE: Results may be misleading due to involvement in interactions
```

```
  v_pairs <- pairs(emm_v)
  v_pairs
```

```
## opponent.f = AI_HMM:
##  contrast     estimate     SE  df z.ratio p.value
##  FALSE - TRUE  -0.0107 0.0200 Inf  -0.534  0.5932
##
## opponent.f = AI_HMM_vol:
##  contrast     estimate     SE  df z.ratio p.value
##  FALSE - TRUE  -0.0114 0.0212 Inf  -0.540  0.5892
##
## Results are averaged over the levels of: d_level
## Degrees-of-freedom method: asymptotic
```

```
# Get emmeans for all combinations
emm_3way <- emmeans(mod_returns_pct, ~ opponent.f*volatile_first | d_level)
```

```
## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'pbkrtest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(pbkrtest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.

## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'lmerTest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(lmerTest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.

## NOTE: Results may be misleading due to involvement in interactions
```

```
# Look at all pairwise comparisons
pairs <- pairs(emm_3way)
print(pairs)
```

```
## d_level = high_D:
##  contrast                            estimate     SE  df z.ratio p.value
##  AI_HMM FALSE - AI_HMM_vol FALSE     9.79e-06 0.0116 Inf   0.001  1.0000
##  AI_HMM FALSE - AI_HMM TRUE         -1.58e-02 0.0283 Inf  -0.558  0.9445
##  AI_HMM FALSE - AI_HMM_vol TRUE     -2.04e-02 0.0292 Inf  -0.700  0.8971
##  AI_HMM_vol FALSE - AI_HMM TRUE     -1.58e-02 0.0291 Inf  -0.542  0.9486
```

```
##   AI_HMM_vol FALSE - AI_HMM_vol TRUE -2.04e-02 0.0300 Inf  -0.682  0.9041
##   AI_HMM TRUE - AI_HMM_vol TRUE      -4.64e-03 0.0120 Inf  -0.386  0.9805
##
## d_level = low_D:
##   contrast                            estimate     SE  df z.ratio p.value
##   AI_HMM FALSE - AI_HMM_vol FALSE     4.05e-03 0.0124 Inf   0.326  0.9880
##   AI_HMM FALSE - AI_HMM TRUE         -5.60e-03 0.0283 Inf  -0.198  0.9973
##   AI_HMM FALSE - AI_HMM_vol TRUE      1.62e-03 0.0291 Inf   0.056  0.9999
##   AI_HMM_vol FALSE - AI_HMM TRUE     -9.65e-03 0.0292 Inf  -0.331  0.9876
##   AI_HMM_vol FALSE - AI_HMM_vol TRUE -2.44e-03 0.0299 Inf  -0.081  0.9998
##   AI_HMM TRUE - AI_HMM_vol TRUE       7.22e-03 0.0111 Inf   0.650  0.9157
##
## Degrees-of-freedom method: asymptotic
## P value adjustment: tukey method for comparing a family of 4 estimates
```

```r
# Get emmeans results in a dataframe for easier viewing
emm_df <- as.data.frame(emm)
print(emm_df)
```

```
## function (...)
## data.frame(value = x(...))
## <bytecode: 0x7fc0bd129b68>
## <environment: 0x7fc0bd1296d0>
```

```r
# Break down the interaction:
# 1. Look at opponent effects for each D-level when volatile_first = TRUE
emm_vol_true <- emmeans(mod_returns_pct, ~ opponent.f| d_level, at = list(volatile_first = TRUE))
```

```
## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'pbkrtest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(pbkrtest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.

## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'lmerTest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(lmerTest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.

## NOTE: Results may be misleading due to involvement in interactions
```

```r
pairs(emm_vol_true)
```

```
## d_level = high_D:
##   contrast            estimate     SE  df z.ratio p.value
##   AI_HMM - AI_HMM_vol -0.00464 0.0120 Inf  -0.386  0.6999
##
## d_level = low_D:
##   contrast            estimate     SE  df z.ratio p.value
##   AI_HMM - AI_HMM_vol  0.00722 0.0111 Inf   0.650  0.5159
##
## Degrees-of-freedom method: asymptotic
```

```r
# 2. Look at opponent effects for each D-level when volatile_first = FALSE
emm_vol_false <- emmeans(mod_returns_pct, ~ opponent.f| d_level, at = list(volatile_first = FALSE))
```

```
## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'pbkrtest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(pbkrtest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.

## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'lmerTest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(lmerTest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.

## NOTE: Results may be misleading due to involvement in interactions
```

```r
pairs(emm_vol_false)
```

```
## d_level = high_D:
##  contrast            estimate     SE  df z.ratio p.value
##  AI_HMM - AI_HMM_vol 9.79e-06 0.0116 Inf   0.001  0.9993
##
## d_level = low_D:
##  contrast            estimate     SE  df z.ratio p.value
##  AI_HMM - AI_HMM_vol 4.05e-03 0.0124 Inf   0.326  0.7442
##
## Degrees-of-freedom method: asymptotic
```

interesting, seems like when volatile is faced first, low_d participants don't change their return, but high_d ones increase their return subsequently when facing human-like HMM.

When human-like is face first, low_d participants subsequently reduce their return against volatile HMM, but high_d subsequently increase them.

This could indicate that:

High D participants are more strategic/adaptive in their behavior, adjusting their returns based on opponent type and order Low D participants are more reactive, particularly showing decreased trust after experiencing the human-like HMM first

Absolute return model

```r
mod_return <- mixed( return ~ opponent.f*inv_scaled*d_level*volatile_first + (1 | playerId), final_data
```

```
## Contrasts set to contr.sum for the following variables: opponent.f, d_level, playerId

## Fitting one lmer() model. [DONE]
## Calculating p-values. [DONE]
```

```r
summary(mod_return)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: return ~ opponent.f * inv_scaled * d_level * volatile_first +
##     (1 | playerId)
##    Data: data
##
## REML criterion at convergence: 55822
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -7.0246 -0.3260  0.0638  0.4286  6.4154
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  playerId (Intercept) 14.56    3.815
##  Residual             24.33    4.933
## Number of obs: 9150, groups:  playerId, 183
##
## Fixed effects:
##                                                Estimate Std. Error
## (Intercept)                                   1.501e+01  4.146e-01
## opponent.f1                                  -2.841e-01  7.560e-02
## inv_scaled                                    9.079e+00  8.572e-02
## d_level1                                     -5.017e-01  4.146e-01
## volatile_firstTRUE                            3.560e-01  5.755e-01
## opponent.f1:inv_scaled                        6.578e-02  8.054e-02
## opponent.f1:d_level1                          2.022e-02  7.560e-02
## inv_scaled:d_level1                          -4.509e-01  8.572e-02
## opponent.f1:volatile_firstTRUE                1.581e-01  1.047e-01
## inv_scaled:volatile_firstTRUE                 1.800e-01  1.195e-01
## d_level1:volatile_firstTRUE                   1.670e-01  5.755e-01
## opponent.f1:inv_scaled:d_level1               9.874e-02  8.054e-02
## opponent.f1:inv_scaled:volatile_firstTRUE    -1.859e-01  1.114e-01
## opponent.f1:d_level1:volatile_firstTRUE      -4.446e-02  1.047e-01
## inv_scaled:d_level1:volatile_firstTRUE        3.174e-01  1.195e-01
## opponent.f1:inv_scaled:d_level1:volatile_firstTRUE -1.796e-03  1.114e-01
##                                                      df t value Pr(>|t|)
## (Intercept)                                   1.772e+02  36.215  < 2e-16
## opponent.f1                                   8.957e+03  -3.758 0.000173
## inv_scaled                                    9.086e+03 105.909  < 2e-16
## d_level1                                      1.772e+02  -1.210 0.227815
## volatile_firstTRUE                            1.772e+02   0.619 0.537007
## opponent.f1:inv_scaled                        9.026e+03   0.817 0.414131
## opponent.f1:d_level1                          8.957e+03   0.268 0.789074
## inv_scaled:d_level1                           9.086e+03  -5.260 1.47e-07
## opponent.f1:volatile_firstTRUE                8.957e+03   1.510 0.131176
## inv_scaled:volatile_firstTRUE                 9.091e+03   1.506 0.132146
## d_level1:volatile_firstTRUE                   1.772e+02   0.290 0.771972
## opponent.f1:inv_scaled:d_level1               9.026e+03   1.226 0.220234
## opponent.f1:inv_scaled:volatile_firstTRUE     9.024e+03  -1.669 0.095219
## opponent.f1:d_level1:volatile_firstTRUE       8.957e+03  -0.425 0.671188
```

```
## inv_scaled:d_level1:volatile_firstTRUE             9.091e+03   2.656 0.007929
## opponent.f1:inv_scaled:d_level1:volatile_firstTRUE  9.024e+03  -0.016 0.987136
##
## (Intercept)                                        ***
## opponent.f1                                        ***
## inv_scaled                                         ***
## d_level1
## volatile_firstTRUE
## opponent.f1:inv_scaled
## opponent.f1:d_level1
## inv_scaled:d_level1                                ***
## opponent.f1:volatile_firstTRUE
## inv_scaled:volatile_firstTRUE
## d_level1:volatile_firstTRUE
## opponent.f1:inv_scaled:d_level1
## opponent.f1:inv_scaled:volatile_firstTRUE          .
## opponent.f1:d_level1:volatile_firstTRUE
## inv_scaled:d_level1:volatile_firstTRUE             **
## opponent.f1:inv_scaled:d_level1:volatile_firstTRUE
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

##
## Correlation matrix not shown by default, as p = 16 > 12.
## Use print(x, correlation=TRUE)  or
##     vcov(x)        if you need it
```

```r
library(emmeans)
library(ggplot2)
library(dplyr)

# For opponent effect
opponent_emm <- emmeans(mod_return, ~ opponent.f)
```

```
## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'pbkrtest.limit = 9150' (or larger)
## [or, globally, 'set emm_options(pbkrtest.limit = 9150)' or larger];
## but be warned that this may result in large computation time and memory use.

## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'lmerTest.limit = 9150' (or larger)
## [or, globally, 'set emm_options(lmerTest.limit = 9150)' or larger];
## but be warned that this may result in large computation time and memory use.

## NOTE: Results may be misleading due to involvement in interactions
```

```r
opponent_pairs <- pairs(opponent_emm)
print(opponent_pairs)
```

```
##  contrast             estimate    SE  df z.ratio p.value
##  AI_HMM - AI_HMM_vol     -0.41 0.105 Inf  -3.916  0.0001
##
## Results are averaged over the levels of: d_level, volatile_first
## Degrees-of-freedom method: asymptotic
```

```r
# For investment × d_level interaction
# First get the slopes for each d_level
slopes <- emtrends(mod_return, ~d_level, var="inv_scaled")
```

```
## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'pbkrtest.limit = 9150' (or larger)
## [or, globally, 'set emm_options(pbkrtest.limit = 9150)' or larger];
## but be warned that this may result in large computation time and memory use.

## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'lmerTest.limit = 9150' (or larger)
## [or, globally, 'set emm_options(lmerTest.limit = 9150)' or larger];
## but be warned that this may result in large computation time and memory use.

## NOTE: Results may be misleading due to involvement in interactions
```

```r
print(slopes)
```

```
##  d_level inv_scaled.trend     SE  df asymp.LCL asymp.UCL
##  high_D              8.88 0.0846 Inf      8.71      9.04
##  low_D               9.46 0.0844 Inf      9.30      9.63
##
## Results are averaged over the levels of: opponent.f, volatile_first
## Degrees-of-freedom method: asymptotic
## Confidence level used: 0.95
```

```r
# Compare slopes between d_levels
slope_pairs <- pairs(slopes)
print(slope_pairs)
```

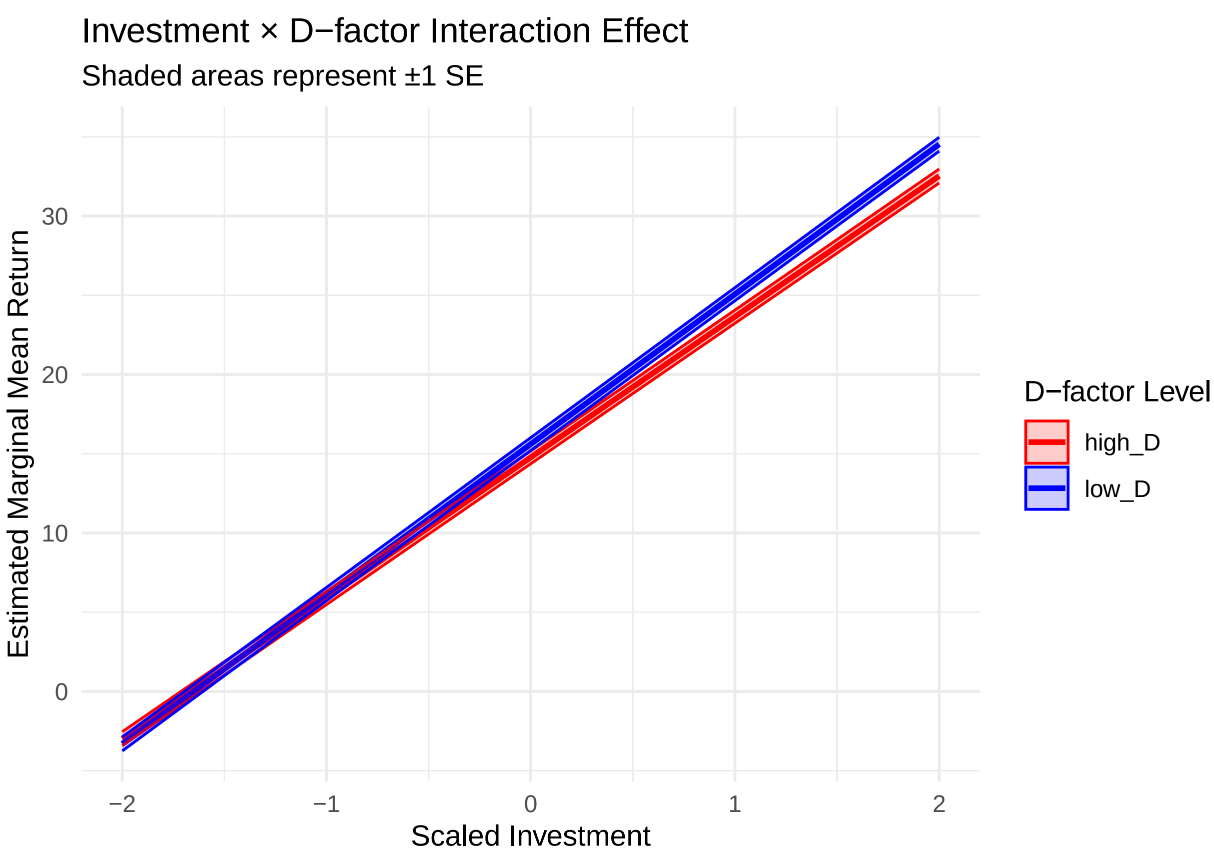
```
##  contrast       estimate   SE  df z.ratio p.value
##  high_D - low_D   -0.584 0.12 Inf  -4.889  <.0001
##
## Results are averaged over the levels of: opponent.f, volatile_first
## Degrees-of-freedom method: asymptotic
```

```r
# Get predicted margins at specific investment values
inv_grid <- seq(from = -2, to = 2, by = 0.5)  # Using standardized values
inv_d_emm <- emmeans(mod_return,
                     ~ d_level | inv_scaled,
                     at = list(inv_scaled = inv_grid))
```

```
## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'pbkrtest.limit = 9150' (or larger)
## [or, globally, 'set emm_options(pbkrtest.limit = 9150)' or larger];
## but be warned that this may result in large computation time and memory use.

## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'lmerTest.limit = 9150' (or larger)
## [or, globally, 'set emm_options(lmerTest.limit = 9150)' or larger];
## but be warned that this may result in large computation time and memory use.
```

```
## NOTE: Results may be misleading due to involvement in interactions
```

```
# Convert to data frame for plotting
inv_d_df <- as.data.frame(inv_d_emm)

# Plot the interaction
ggplot(inv_d_df, aes(x = inv_scaled, y = emmean, color = d_level)) +
  geom_line(size = 1) +
  geom_ribbon(aes(ymin = emmean - SE,
                  ymax = emmean + SE,
                  fill = d_level),
              alpha = 0.2) +
  scale_color_manual(values = c("high_D" = "red", "low_D" = "blue")) +
  scale_fill_manual(values = c("high_D" = "red", "low_D" = "blue")) +
  labs(x = "Scaled Investment",
       y = "Estimated Marginal Mean Return",
       title = "Investment × D-factor Interaction Effect",
       subtitle = "Shaded areas represent ±1 SE",
       color = "D-factor Level",
       fill = "D-factor Level") +
  theme_minimal()
```



Pct return model whith latent inv state

```
mod_returns_latent <- mixed( ret_pct_na ~ opponent.f*investorState.f*d_level*volatile_first + ( 1 + opp
```

```
## Contrasts set to contr.sum for the following variables: opponent.f, investorState.f, d_level, player
```

```
## Warning: Due to missing values, reduced number of observations to 8875

## Fitting one lmer() model. [DONE]
## Calculating p-values. [DONE]
```

```r
anova(mod_returns_latent)
```

```
## Mixed Model Anova Table (Type 3 tests, KR-method)
##
## Model: ret_pct_na ~ opponent.f * investorState.f * d_level * volatile_first +
## Model:     (1 + opponent.f | playerId)
## Data: final_data
##                                                   num Df  den Df       F
## opponent.f                                             1  193.33  0.0565
## investorState.f                                        2 7961.78 18.8252
## d_level                                                1  180.38  0.3148
## volatile_first                                         1  180.38  0.3543
## opponent.f:investorState.f                             2 5980.49  1.0739
## opponent.f:d_level                                     1  193.33  0.0802
## investorState.f:d_level                                2 7961.78  7.8106
## opponent.f:volatile_first                              1  193.33  0.0255
## investorState.f:volatile_first                         2 7961.78  0.8317
## d_level:volatile_first                                 1  180.38  0.0901
## opponent.f:investorState.f:d_level                     2 5980.49  3.7735
## opponent.f:investorState.f:volatile_first              2 5980.49  3.9898
## opponent.f:d_level:volatile_first                      1  193.33  0.1538
## investorState.f:d_level:volatile_first                 2 7961.78  1.3108
## opponent.f:investorState.f:d_level:volatile_first      2 5980.49  0.3972
##                                                      Pr(>F)
## opponent.f                                        0.8123074
## investorState.f                                   6.976e-09 ***
## d_level                                           0.5754305
## volatile_first                                    0.5524305
## opponent.f:investorState.f                        0.3417244
## opponent.f:d_level                                0.7773543
## investorState.f:d_level                           0.0004085 ***
## opponent.f:volatile_first                         0.8734117
## investorState.f:volatile_first                    0.4353479
## d_level:volatile_first                            0.7644529
## opponent.f:investorState.f:d_level                0.0230260 *
## opponent.f:investorState.f:volatile_first         0.0185535 *
## opponent.f:d_level:volatile_first                 0.6953675
## investorState.f:d_level:volatile_first            0.2696502
## opponent.f:investorState.f:d_level:volatile_first 0.6722206
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
summary(mod_returns_latent)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula:
```

```
## ret_pct_na ~ opponent.f * investorState.f * d_level * volatile_first +
##     (1 + opponent.f | playerId)
##    Data: data
##
## REML criterion at convergence: -7084.4
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -4.2886 -0.2912  0.0390  0.3561  5.4828
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  playerId (Intercept) 0.017280 0.13145
##           opponent.f1 0.001124 0.03352  -0.11
##  Residual             0.023369 0.15287
## Number of obs: 8875, groups:  playerId, 183
##
## Fixed effects:
##                                                          Estimate Std. Error
## (Intercept)                                             4.420e-01  1.428e-02
## opponent.f1                                             2.385e-04  4.412e-03
## investorState.f1                                       -2.116e-02  4.952e-03
## investorState.f2                                        1.978e-03  3.807e-03
## d_level1                                               -8.532e-03  1.428e-02
## volatile_firstTRUE                                      1.179e-02  1.981e-02
## opponent.f1:investorState.f1                           -3.400e-03  4.755e-03
## opponent.f1:investorState.f2                           -7.070e-03  3.772e-03
## opponent.f1:d_level1                                    3.322e-04  4.412e-03
## investorState.f1:d_level1                               1.754e-02  4.952e-03
## investorState.f2:d_level1                              -3.458e-03  3.807e-03
## opponent.f1:volatile_firstTRUE                          9.725e-04  6.095e-03
## investorState.f1:volatile_firstTRUE                     8.080e-03  6.672e-03
## investorState.f2:volatile_firstTRUE                    -5.280e-03  5.238e-03
## d_level1:volatile_firstTRUE                             5.946e-03  1.981e-02
## opponent.f1:investorState.f1:d_level1                  -1.917e-03  4.755e-03
## opponent.f1:investorState.f2:d_level1                  -3.268e-03  3.772e-03
## opponent.f1:investorState.f1:volatile_firstTRUE         9.564e-03  6.415e-03
## opponent.f1:investorState.f2:volatile_firstTRUE         6.728e-03  5.206e-03
## opponent.f1:d_level1:volatile_firstTRUE                -2.390e-03  6.095e-03
## investorState.f1:d_level1:volatile_firstTRUE           -9.152e-03  6.672e-03
## investorState.f2:d_level1:volatile_firstTRUE           -1.786e-04  5.238e-03
## opponent.f1:investorState.f1:d_level1:volatile_firstTRUE -3.974e-03  6.415e-03
## opponent.f1:investorState.f2:d_level1:volatile_firstTRUE -1.208e-03  5.206e-03
##                                                                df t value
## (Intercept)                                             1.772e+02  30.956
## opponent.f1                                             1.862e+02   0.054
## investorState.f1                                        7.594e+03  -4.273
## investorState.f2                                        8.536e+03   0.520
## d_level1                                                1.772e+02  -0.597
## volatile_firstTRUE                                      1.768e+02   0.595
## opponent.f1:investorState.f1                            4.549e+03  -0.715
## opponent.f1:investorState.f2                            8.042e+03  -1.874
## opponent.f1:d_level1                                    1.862e+02   0.075
## investorState.f1:d_level1                               7.594e+03   3.542
```

```
## investorState.f2:d_level1                               8.536e+03  -0.908
## opponent.f1:volatile_firstTRUE                          1.832e+02   0.160
## investorState.f1:volatile_firstTRUE                     7.688e+03   1.211
## investorState.f2:volatile_firstTRUE                     8.492e+03  -1.008
## d_level1:volatile_firstTRUE                             1.768e+02   0.300
## opponent.f1:investorState.f1:d_level1                   4.549e+03  -0.403
## opponent.f1:investorState.f2:d_level1                   8.042e+03  -0.866
## opponent.f1:investorState.f1:volatile_firstTRUE         4.682e+03   1.491
## opponent.f1:investorState.f2:volatile_firstTRUE         8.172e+03   1.292
## opponent.f1:d_level1:volatile_firstTRUE                 1.832e+02  -0.392
## investorState.f1:d_level1:volatile_firstTRUE            7.688e+03  -1.372
## investorState.f2:d_level1:volatile_firstTRUE            8.492e+03  -0.034
## opponent.f1:investorState.f1:d_level1:volatile_firstTRUE 4.682e+03  -0.619
## opponent.f1:investorState.f2:d_level1:volatile_firstTRUE 8.172e+03  -0.232
##                                                        Pr(>|t|)
## (Intercept)                                             < 2e-16 ***
## opponent.f1                                              0.9570
## investorState.f1                                        1.95e-05 ***
## investorState.f2                                         0.6034
## d_level1                                                 0.5510
## volatile_firstTRUE                                       0.5524
## opponent.f1:investorState.f1                             0.4747
## opponent.f1:investorState.f2                             0.0609 .
## opponent.f1:d_level1                                     0.9401
## investorState.f1:d_level1                                0.0004 ***
## investorState.f2:d_level1                                0.3638
## opponent.f1:volatile_firstTRUE                           0.8734
## investorState.f1:volatile_firstTRUE                      0.2259
## investorState.f2:volatile_firstTRUE                      0.3135
## d_level1:volatile_firstTRUE                              0.7645
## opponent.f1:investorState.f1:d_level1                    0.6869
## opponent.f1:investorState.f2:d_level1                    0.3864
## opponent.f1:investorState.f1:volatile_firstTRUE          0.1360
## opponent.f1:investorState.f2:volatile_firstTRUE          0.1963
## opponent.f1:d_level1:volatile_firstTRUE                  0.6954
## investorState.f1:d_level1:volatile_firstTRUE             0.1702
## investorState.f2:d_level1:volatile_firstTRUE             0.9728
## opponent.f1:investorState.f1:d_level1:volatile_firstTRUE  0.5356
## opponent.f1:investorState.f2:d_level1:volatile_firstTRUE  0.8166
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

##
## Correlation matrix not shown by default, as p = 24 > 12.
## Use print(x, correlation=TRUE)  or
##     vcov(x)        if you need it
```

```
# First get emmeans object for the interaction
emm <- emmeans(mod_returns_latent, ~ investorState.f | d_level)
```

```
## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'pbkrtest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(pbkrtest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.
```

```
## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'lmerTest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(lmerTest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.

## NOTE: Results may be misleading due to involvement in interactions
```

```r
# Create custom contrasts for state comparisons
# For each D-level we want:
# 1. neutral - unhappy
# 2. happy - unhappy
# 3. happy - neutral

# Define the contrasts
state_contrasts <- list(
  neutral_vs_unhappy = c(-1, 1, 0),  # neutral - unhappy
  happy_vs_unhappy = c(-1, 0, 1),    # happy - unhappy
  happy_vs_neutral = c(0, -1, 1)     # happy - neutral
)

# Apply contrasts within each D-level
results <- contrast(emm,
                   method = state_contrasts,
                   by = "d_level",
                   adjust = "none")  # No adjustment since these are planned comparisons

# Print results
print("Contrast results within each D-level:")
```

```
## [1] "Contrast results within each D-level:"
```

```r
print(results)
```

```
## d_level = high_D:
##  contrast            estimate      SE  df z.ratio p.value
##  neutral_vs_unhappy -5.26e-05 0.00719 Inf  -0.007  0.9942
##  happy_vs_unhappy    1.25e-02 0.00820 Inf   1.528  0.1266
##  happy_vs_neutral    1.26e-02 0.00654 Inf   1.923  0.0544
##
## d_level = low_D:
##  contrast            estimate      SE  df z.ratio p.value
##  neutral_vs_unhappy  3.30e-02 0.00750 Inf   4.393  <.0001
##  happy_vs_unhappy    5.73e-02 0.00814 Inf   7.037  <.0001
##  happy_vs_neutral    2.43e-02 0.00628 Inf   3.870  0.0001
##
## Results are averaged over the levels of: opponent.f, volatile_first
## Degrees-of-freedom method: asymptotic
```

```r
# Test if these contrasts differ between D-levels
contrast_diffs <- contrast(results,
                          method = "revpairwise",
                          by = "contrast",
```

```r
                          adjust = "none")

print("\nD-level differences in contrasts:")
```

```
## [1] "\nD-level differences in contrasts:"
```

```r
print(contrast_diffs)
```

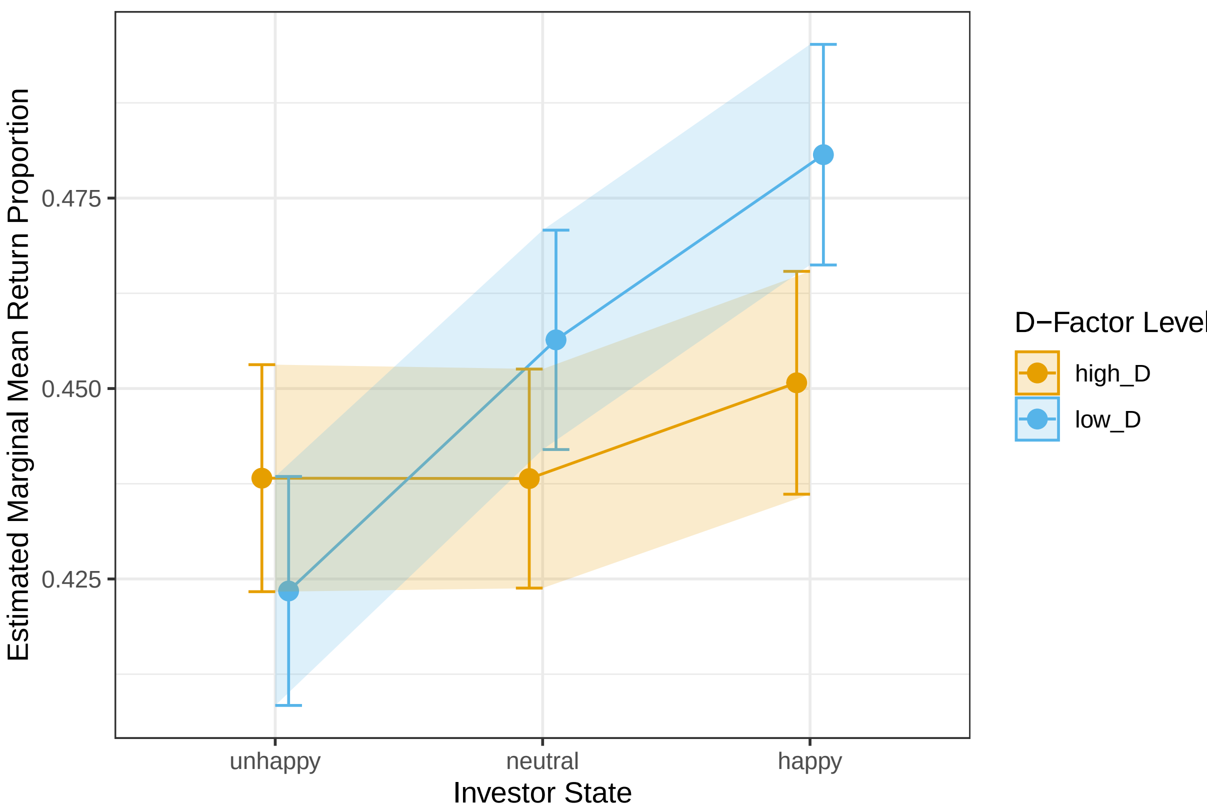
```
## contrast = neutral_vs_unhappy:
##  contrast1      estimate      SE  df z.ratio p.value
##  low_D - high_D   0.0330 0.01039 Inf   3.177  0.0015
##
## contrast = happy_vs_unhappy:
##  contrast1      estimate      SE  df z.ratio p.value
##  low_D - high_D   0.0448 0.01155 Inf   3.874  0.0001
##
## contrast = happy_vs_neutral:
##  contrast1      estimate      SE  df z.ratio p.value
##  low_D - high_D   0.0117 0.00907 Inf   1.294  0.1956
##
## Results are averaged over the levels of: opponent.f, volatile_first
## Degrees-of-freedom method: asymptotic
```

```r
# Get estimated marginal means
emm <- emmeans(mod_returns_latent, ~ investorState.f * d_level)
```

```
## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'pbkrtest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(pbkrtest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.

## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'lmerTest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(lmerTest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.

## NOTE: Results may be misleading due to involvement in interactions
```

```r
emm_df <- as.data.frame(emm)

# Create plot using emmeans
ggplot(emm_df, aes(x = investorState.f, y = emmean, color = d_level, group = d_level)) +
  # Add lines to show pattern
  geom_line(position = position_dodge(width = 0.2)) +
  # Add points
  geom_point(position = position_dodge(width = 0.2), size = 3) +
  # Add error bars using model-based SE
  geom_errorbar(aes(ymin = emmean - SE, ymax = emmean + SE),
                width = 0.2,
                position = position_dodge(width = 0.2)) +
```

```r
  # Add shaded bands to show overlap
  geom_ribbon(aes(ymin = emmean - SE,
                  ymax = emmean + SE,
                  fill = d_level,
                  color = NULL),
              alpha = 0.2) +
  # Customize appearance
  labs(x = "Investor State",
       y = "Estimated Marginal Mean Return Proportion",
       color = "D-Factor Level",
       fill = "D-Factor Level") +
  theme_bw() +
  # Use colorblind-friendly colors
  scale_color_manual(values = c("high_D" = "#E69F00", "low_D" = "#56B4E9")) +
  scale_fill_manual(values = c("high_D" = "#E69F00", "low_D" = "#56B4E9")) +
  # Add note about non-significance
  labs(caption = "Note: Overlapping error bands indicate non-significant differences between groups")
```



Note: Overlapping error bands indicate non−significant differences between groups

The analysis revealed different patterns of responses to investor states between high and low D-factor participants. Low D-factor participants showed significant increases in return rates from unhappy to neutral states (b = 0.033, SE = 0.0075, z = 4.39, p < .001) and from unhappy to happy states (b = 0.057, SE = 0.0081, z = 7.04, p < .001), as well as from neutral to happy states (b = 0.024, SE = 0.0063, z = 3.87, p < .001). In contrast, high D-factor participants showed no significant differences in returns between unhappy and neutral states (b = -0.00005, SE = 0.0072, z = -0.007, p = .994) or unhappy and happy states (b = 0.013, SE = 0.0082, z = 1.53, p = .127), though they showed a marginally significant increase from neutral to happy states (b = 0.013, SE = 0.0065, z = 1.92, p = .054). Direct comparisons between D-factor groups revealed that low D-factor participants showed significantly stronger positive responses than high

D-factor participants between unhappy and neutral states (b = 0.033, SE = 0.0104, z = 3.18, p = .002) and between unhappy and happy states (b = 0.045, SE = 0.0116, z = 3.87, p < .001). However, the groups did not differ significantly in their response to the transition from neutral to happy states (b = 0.012, SE = 0.0091, z = 1.29, p = .196). These results suggest that low D-factor participants were more responsive to improvements in investor state, particularly when recovering from an unhappy state, while high D-factor participants showed more stable returns across investor states.

```r
# First get emmeans for the three-way interaction
emm_three <- emmeans(mod_returns_latent, ~ investorState.f | d_level | opponent.f)
```

```
## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'pbkrtest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(pbkrtest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.

## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'lmerTest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(lmerTest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.

## NOTE: Results may be misleading due to involvement in interactions
```

```r
# Define contrasts for investor states as before
state_contrasts <- list(
  neutral_vs_unhappy = c(-1, 1, 0),  # neutral - unhappy
  happy_vs_unhappy = c(-1, 0, 1),    # happy - unhappy
  happy_vs_neutral = c(0, -1, 1)     # happy - neutral
)

# Apply contrasts within each d_level and opponent combination
results_three <- contrast(emm_three,
                         method = state_contrasts,
                         by = c("d_level", "opponent.f"),
                         adjust = "none")

# Test if these contrasts differ between D-levels for each opponent
contrast_diffs_by_opponent <- contrast(results_three,
                                       method = "revpairwise",
                                       by = c("contrast", "opponent.f"),
                                       adjust = "none")

# Print results
print("Contrast results within each D-level and opponent:")
```

```
## [1] "Contrast results within each D-level and opponent:"
```

```r
print(results_three)
```

```
## d_level = high_D, opponent.f = AI_HMM:
##  contrast            estimate      SE  df z.ratio p.value
##  neutral_vs_unhappy -5.11e-03 0.01102 Inf  -0.464  0.6429
```

```
##  happy_vs_unhappy    2.51e-02 0.01202 Inf   2.092  0.0364
##  happy_vs_neutral    3.03e-02 0.00837 Inf   3.615  0.0003
##
## d_level = low_D, opponent.f = AI_HMM:
##  contrast            estimate      SE  df z.ratio p.value
##  neutral_vs_unhappy  2.78e-02 0.01149 Inf   2.424  0.0154
##  happy_vs_unhappy    4.65e-02 0.01239 Inf   3.755  0.0002
##  happy_vs_neutral    1.87e-02 0.00840 Inf   2.224  0.0261
##
## d_level = high_D, opponent.f = AI_HMM_vol:
##  contrast            estimate      SE  df z.ratio p.value
##  neutral_vs_unhappy  5.00e-03 0.00907 Inf   0.551  0.5814
##  happy_vs_unhappy   -9.61e-05 0.01071 Inf  -0.009  0.9928
##  happy_vs_neutral   -5.10e-03 0.00992 Inf  -0.514  0.6074
##
## d_level = low_D, opponent.f = AI_HMM_vol:
##  contrast            estimate      SE  df z.ratio p.value
##  neutral_vs_unhappy  3.81e-02 0.00926 Inf   4.115  <.0001
##  happy_vs_unhappy    6.80e-02 0.00999 Inf   6.811  <.0001
##  happy_vs_neutral    2.99e-02 0.00931 Inf   3.216  0.0013
##
## Results are averaged over the levels of: volatile_first
## Degrees-of-freedom method: asymptotic
```

```r
print("\nD-level differences in contrasts by opponent:")
```

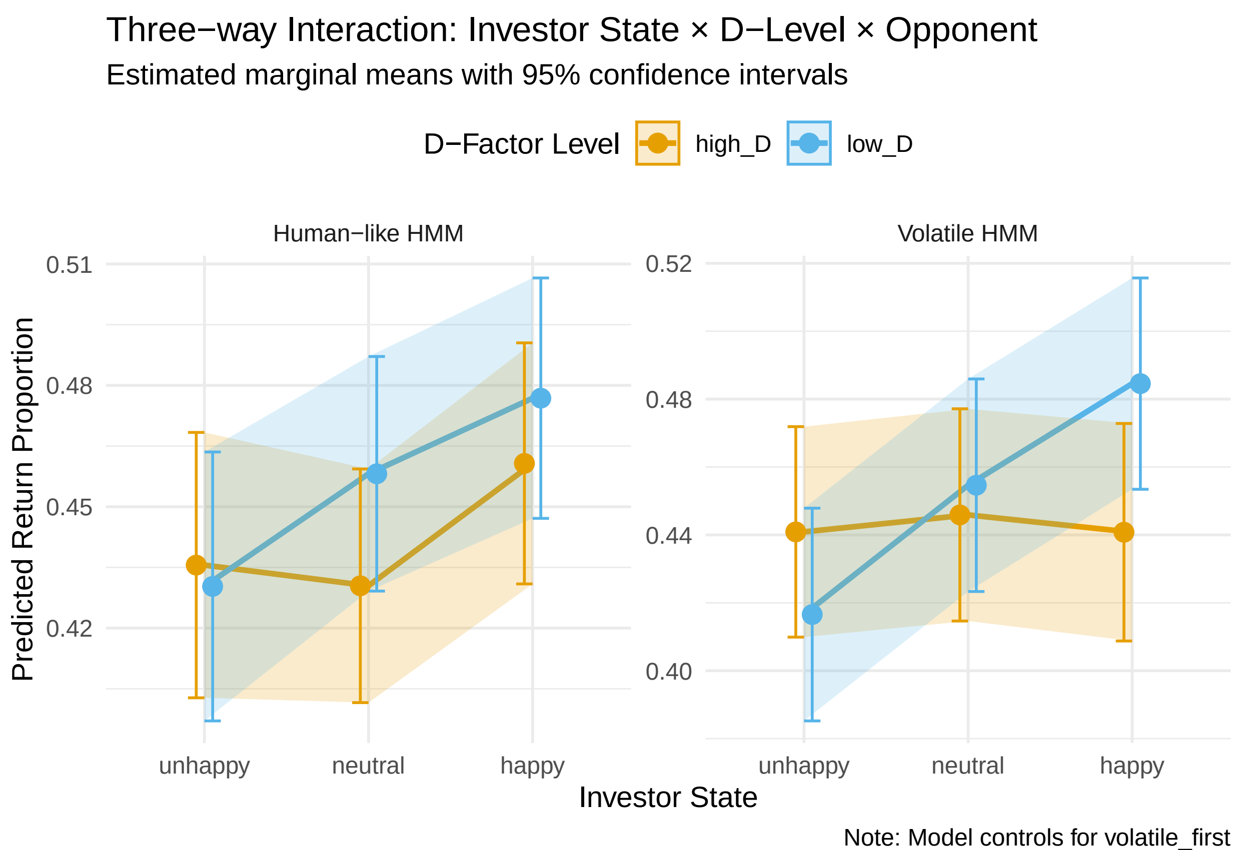
```
## [1] "\nD-level differences in contrasts by opponent:"
```

```r
print(contrast_diffs_by_opponent)
```

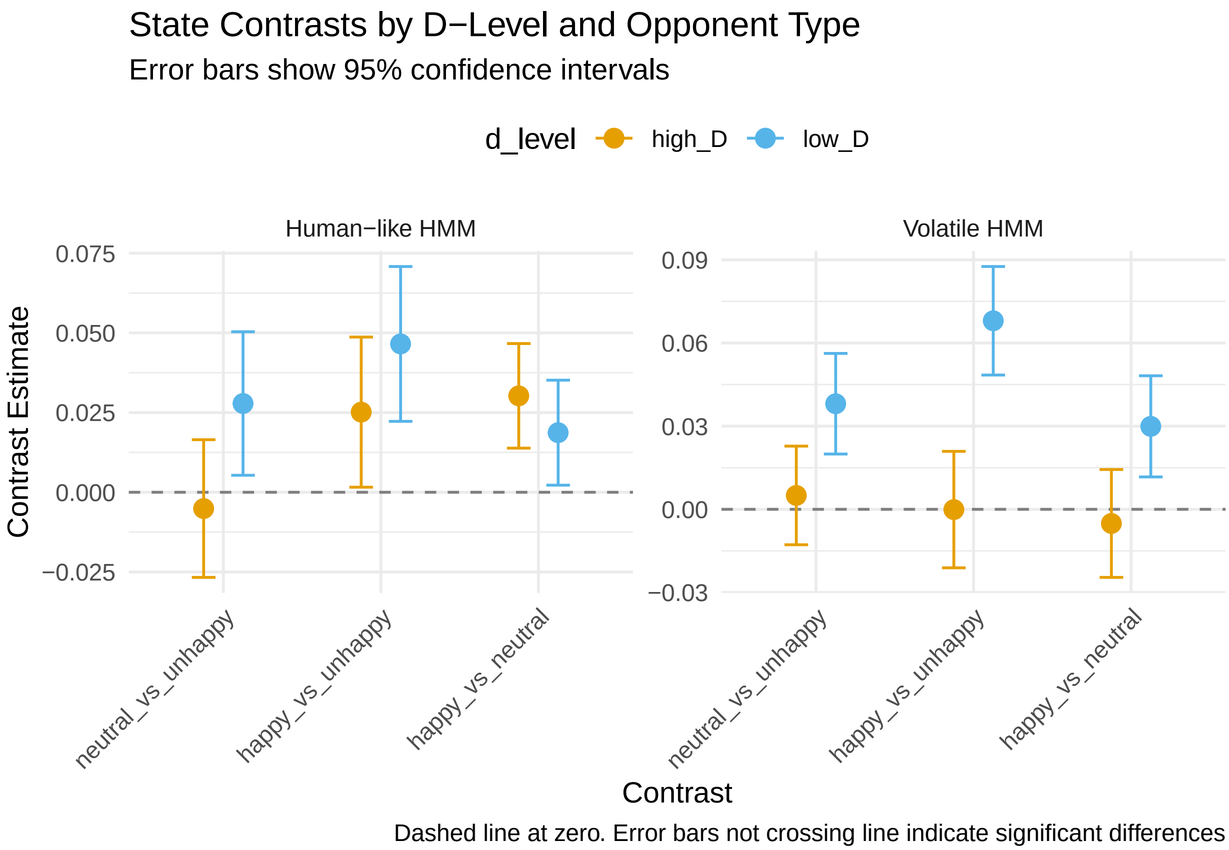
```
## contrast = neutral_vs_unhappy, opponent.f = AI_HMM:
##  contrast1      estimate     SE  df z.ratio p.value
##  low_D - high_D   0.0330 0.0159 Inf   2.070  0.0384
##
## contrast = happy_vs_unhappy, opponent.f = AI_HMM:
##  contrast1      estimate     SE  df z.ratio p.value
##  low_D - high_D   0.0214 0.0173 Inf   1.239  0.2152
##
## contrast = happy_vs_neutral, opponent.f = AI_HMM:
##  contrast1      estimate     SE  df z.ratio p.value
##  low_D - high_D  -0.0116 0.0119 Inf  -0.975  0.3297
##
## contrast = neutral_vs_unhappy, opponent.f = AI_HMM_vol:
##  contrast1      estimate     SE  df z.ratio p.value
##  low_D - high_D   0.0331 0.0130 Inf   2.552  0.0107
##
## contrast = happy_vs_unhappy, opponent.f = AI_HMM_vol:
##  contrast1      estimate     SE  df z.ratio p.value
##  low_D - high_D   0.0681 0.0146 Inf   4.650  <.0001
##
## contrast = happy_vs_neutral, opponent.f = AI_HMM_vol:
##  contrast1      estimate     SE  df z.ratio p.value
##  low_D - high_D   0.0350 0.0136 Inf   2.574  0.0100
```

```
##
## Results are averaged over the levels of: volatile_first
## Degrees-of-freedom method: asymptotic
```

```r
# Create plot of the three-way interaction
# First get estimated marginal means in a format good for plotting
emm_df <- as.data.frame(emm_three)

#Create plot
ggplot(emm_df, aes(x = investorState.f, y = emmean, color = d_level, group = d_level)) +
  # Add lines
  geom_line(linewidth = 1) +
  # Add ribbons for CI
  geom_ribbon(aes(ymin = emmean - SE*1.96,
                  ymax = emmean + SE*1.96,
                  fill = d_level,
                  color = NULL),
              alpha = 0.2) +
  # Add error bars
  geom_errorbar(aes(ymin = emmean - SE*1.96,
                    ymax = emmean + SE*1.96),
                width = 0.2,
                position = position_dodge(width = 0.2)) +
  # Add points
  geom_point(size = 3, position = position_dodge(width = 0.2)) +
  # Separate by opponent
  facet_wrap(~opponent.f, scales = "free_y",
             labeller = labeller(opponent.f = c("AI_HMM" = "Human-like HMM",
                                                "AI_HMM_vol" = "Volatile HMM"))) +
  # Customize appearance
  scale_color_manual(values = c("high_D" = "#E69F00", "low_D" = "#56B4E9"),
                     name = "D-Factor Level") +
  scale_fill_manual(values = c("high_D" = "#E69F00", "low_D" = "#56B4E9"),
                    name = "D-Factor Level") +
  labs(x = "Investor State",
       y = "Predicted Return Proportion",
       title = "Three-way Interaction: Investor State × D-Level × Opponent",
       subtitle = "Estimated marginal means with 95% confidence intervals",
       caption = "Note: Model controls for volatile_first") +
  theme_minimal() +
  theme(legend.position = "top")
```




```r
# Create a plot specifically showing the contrasts
contrast_df <- as.data.frame(results_three)

ggplot(contrast_df, aes(x = contrast, y = estimate, color = d_level)) +
  # Add zero reference line first so it's in the background
  geom_hline(yintercept = 0, linetype = "dashed", color = "gray50") +
  geom_point(position = position_dodge(width = 0.5), size = 3) +
  geom_errorbar(aes(ymin = estimate - SE*1.96,
                    ymax = estimate + SE*1.96),
                position = position_dodge(width = 0.5),
                width = 0.3) +
  facet_wrap(~opponent.f, scales = "free_y",
             labeller = labeller(opponent.f = c("AI_HMM" = "Human-like HMM",
                                                "AI_HMM_vol" = "Volatile HMM"))) +
  scale_color_manual(values = c("high_D" = "#E69F00", "low_D" = "#56B4E9")) +
  labs(x = "Contrast",
       y = "Contrast Estimate",
       title = "State Contrasts by D-Level and Opponent Type",
       subtitle = "Error bars show 95% confidence intervals",
       caption = "Dashed line at zero. Error bars not crossing line indicate significant differences") +
  theme_minimal() +
  theme(legend.position = "top",
        axis.text.x = element_text(angle = 45, hjust = 1))
```

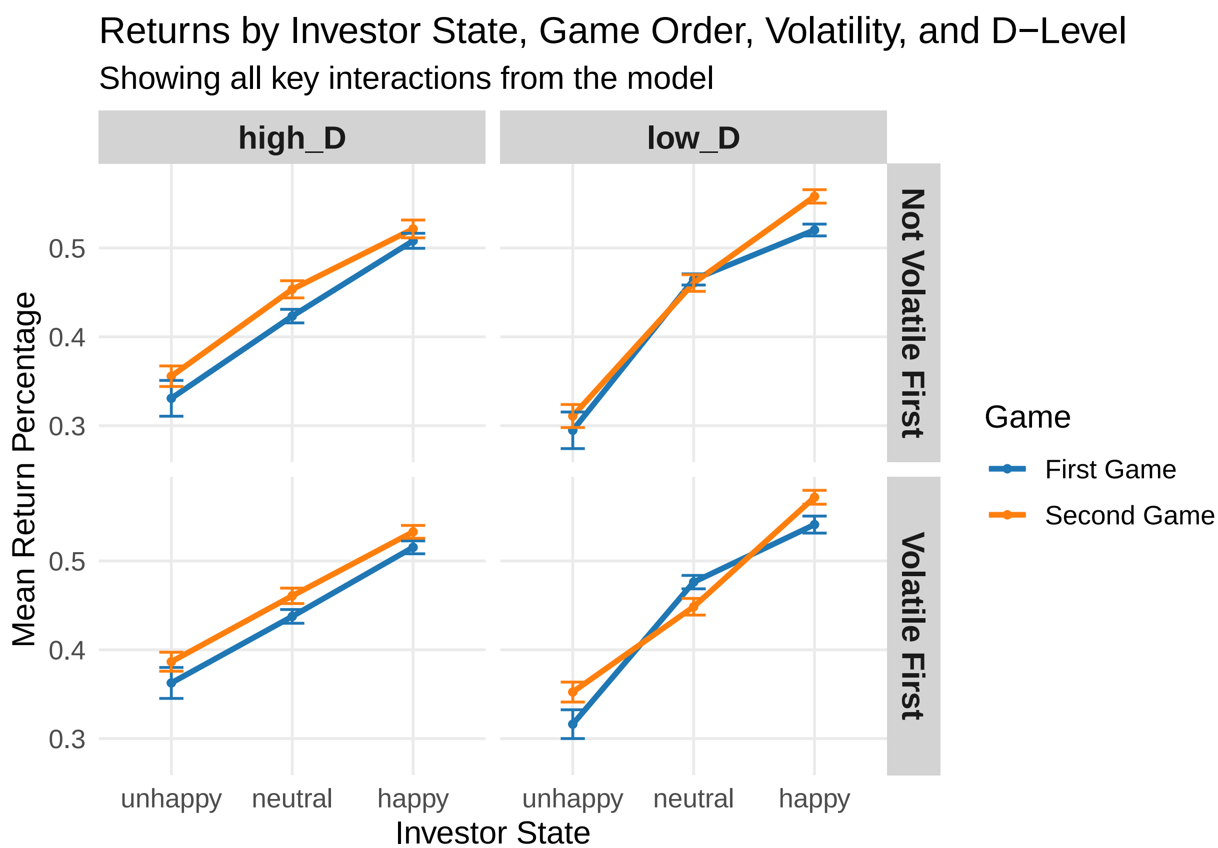


across investor states, with differences between opponent types. With the human-like HMM opponent, high D-factor participants showed higher returns in happy versus neutral states ($b = 0.030$, $SE = 0.008$, $z = 3.62$, $p < .001$) and happy versus unhappy states ($b = 0.025$, $SE = 0.012$, $z = 2.09$, $p = .036$). Low D-factor participants showed higher returns across all state comparisons: neutral versus unhappy ($b = 0.028$, $SE = 0.011$, $z = 2.42$, $p = .015$), happy versus unhappy ($b = 0.047$, $SE = 0.012$, $z = 3.76$, $p < .001$), and happy versus neutral ($b = 0.019$, $SE = 0.008$, $z = 2.22$, $p = .026$).

With the volatile HMM opponent, the pattern diverged markedly. High D-factor participants showed no significant differences in returns between any investor states (all ps $> .58$). In contrast, low D-factor participants maintained significant differences, with higher returns in neutral versus unhappy states ($b = 0.038$, $SE = 0.009$, $z = 4.12$, $p < .001$), happy versus unhappy states ($b = 0.068$, $SE = 0.010$, $z = 6.81$, $p < .001$), and happy versus neutral states ($b = 0.030$, $SE = 0.009$, $z = 3.22$, $p = .001$). The differences between D-factor groups were most pronounced with the volatile opponent, where low D-factor participants showed consistently larger differences in returns between states compared to high D-factor participants (neutral vs unhappy: $b = 0.033$, $SE = 0.013$, $z = 2.55$, $p = .011$; happy vs unhappy: $b = 0.068$, $SE = 0.015$, $z = 4.65$, $p < .001$; happy vs neutral: $b = 0.035$, $SE = 0.014$, $z = 2.57$, $p = .010$).

```r
# Load required libraries
library(dplyr)
library(ggplot2)
library(emmeans)

# Create summary for full interaction including all factors
full_interaction_summary <- final_data %>%
  group_by(investorState.f, opponent.f, volatile_first, d_level) %>%
  summarize(
    mean_return = mean(ret_pct_na, na.rm = TRUE),
    se_return = sd(ret_pct_na, na.rm = TRUE) / sqrt(n()),
```

```r
    n = n(),
    .groups = "drop"
  )

# Create faceted plot showing all interactions
p_full <- ggplot(full_interaction_summary,
                 aes(x = investorState.f, y = mean_return,
                     color = factor(opponent.f), group = opponent.f)) +
  geom_point(size = 1) +
  geom_line(linewidth = 1) +
  geom_errorbar(aes(ymin = mean_return - se_return,
                    ymax = mean_return + se_return),
                width = 0.2) +
  facet_grid(volatile_first  ~ d_level,
             labeller = labeller(
               volatile_first = c("FALSE" = "Not Volatile First",
                                  "TRUE" = "Volatile First"),
               d_level = c("low_d" = "Low D",
                           "high_d" = "High D"))) +
  theme_minimal() +
  theme(
    text = element_text(size = 12),
    panel.grid.minor = element_blank(),
    strip.text = element_text(size = 12, face = "bold"),
    strip.background = element_rect(fill = "lightgray", color = NA)
  ) +
  scale_color_manual(values = c("#1f77b4", "#ff7f0e"),
                     labels = c("First Game", "Second Game")) +
  labs(
    title = "Returns by Investor State, Game Order, Volatility, and D-Level",
    subtitle = "Showing all key interactions from the model",
    x = "Investor State",
    y = "Mean Return Percentage",
    color = "Game"
  )

# Print the plot
print(p_full)
```




```r
# Calculate some key contrasts for interpretation
emm_full <- emmeans(mod_returns_latent,
                    ~ investorState.f | opponent.f:volatile_first:d_level)
```

```
## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'pbkrtest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(pbkrtest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.

## Note: D.f. calculations have been disabled because the number of observations exceeds 3000.
## To enable adjustments, add the argument 'lmerTest.limit = 8875' (or larger)
## [or, globally, 'set emm_options(lmerTest.limit = 8875)' or larger];
## but be warned that this may result in large computation time and memory use.
```

```r
# Print contrasts within each combination
cat("\nState Contrasts within each Condition Combination:\n")
```

```
##
## State Contrasts within each Condition Combination:
```

```r
print(pairs(emm_full))
```

```
## opponent.f = AI_HMM, volatile_first = FALSE, d_level = high_D:
##  contrast          estimate     SE  df z.ratio p.value
##  unhappy - neutral 0.002880 0.0150 Inf   0.192  0.9799
```

```
##  unhappy - happy   -0.029691 0.0170 Inf  -1.749  0.1871
##  neutral - happy   -0.032572 0.0117 Inf  -2.779  0.0151
##
## opponent.f = AI_HMM_vol, volatile_first = FALSE, d_level = high_D:
##  contrast           estimate     SE  df z.ratio p.value
##  unhappy - neutral -0.007162 0.0127 Inf  -0.564  0.8393
##  unhappy - happy    0.012248 0.0146 Inf   0.837  0.6801
##  neutral - happy    0.019410 0.0139 Inf   1.394  0.3440
##
## opponent.f = AI_HMM, volatile_first =  TRUE, d_level = high_D:
##  contrast           estimate     SE  df z.ratio p.value
##  unhappy - neutral  0.007337 0.0161 Inf   0.455  0.8922
##  unhappy - happy   -0.020594 0.0170 Inf  -1.210  0.4469
##  neutral - happy   -0.027931 0.0119 Inf  -2.338  0.0507
##
## opponent.f = AI_HMM_vol, volatile_first =  TRUE, d_level = high_D:
##  contrast           estimate     SE  df z.ratio p.value
##  unhappy - neutral -0.002845 0.0130 Inf  -0.219  0.9738
##  unhappy - happy   -0.012056 0.0157 Inf  -0.770  0.7213
##  neutral - happy   -0.009211 0.0141 Inf  -0.651  0.7917
##
## opponent.f = AI_HMM, volatile_first = FALSE, d_level = low_D:
##  contrast           estimate     SE  df z.ratio p.value
##  unhappy - neutral -0.041813 0.0184 Inf  -2.278  0.0589
##  unhappy - happy   -0.078724 0.0194 Inf  -4.066  0.0001
##  neutral - happy   -0.036911 0.0121 Inf  -3.042  0.0066
##
## opponent.f = AI_HMM_vol, volatile_first = FALSE, d_level = low_D:
##  contrast           estimate     SE  df z.ratio p.value
##  unhappy - neutral -0.046452 0.0142 Inf  -3.274  0.0030
##  unhappy - happy   -0.065188 0.0149 Inf  -4.360  <.0001
##  neutral - happy   -0.018736 0.0134 Inf  -1.394  0.3440
##
## opponent.f = AI_HMM, volatile_first =  TRUE, d_level = low_D:
##  contrast           estimate     SE  df z.ratio p.value
##  unhappy - neutral -0.013877 0.0138 Inf  -1.004  0.5741
##  unhappy - happy   -0.014350 0.0155 Inf  -0.927  0.6229
##  neutral - happy   -0.000473 0.0116 Inf  -0.041  0.9991
##
## opponent.f = AI_HMM_vol, volatile_first =  TRUE, d_level = low_D:
##  contrast           estimate     SE  df z.ratio p.value
##  unhappy - neutral -0.029720 0.0119 Inf  -2.499  0.0333
##  unhappy - happy   -0.070836 0.0132 Inf  -5.350  <.0001
##  neutral - happy   -0.041116 0.0129 Inf  -3.193  0.0040
##
## Degrees-of-freedom method: asymptotic
## P value adjustment: tukey method for comparing a family of 3 estimates
```

```
# Calculate effect sizes for the state differences in each condition
contrasts <- pairs(emm_full)
effect_sizes <- as.data.frame(contrasts) %>%
  mutate(
    effect_size = estimate / SE,
    significant = p.value < 0.05
```

```r
  )

# Print summary of largest effects
cat("\nLargest State Effects:\n")
```

```
##
## Largest State Effects:
```

```r
print(effect_sizes %>%
  dplyr::arrange(desc(abs(effect_size))) %>%
  head(10))
```

```
##              contrast opponent.f volatile_first d_level    estimate         SE
## 1    unhappy - happy AI_HMM_vol           TRUE   low_D -0.07083585 0.01324058
## 2    unhappy - happy AI_HMM_vol          FALSE   low_D -0.06518786 0.01494968
## 3    unhappy - happy     AI_HMM          FALSE   low_D -0.07872403 0.01936388
## 4  unhappy - neutral AI_HMM_vol          FALSE   low_D -0.04645168 0.01418810
## 5    neutral - happy AI_HMM_vol           TRUE   low_D -0.04111612 0.01287530
## 6    neutral - happy     AI_HMM          FALSE   low_D -0.03691126 0.01213289
## 7    neutral - happy     AI_HMM          FALSE  high_D -0.03257176 0.01172105
## 8  unhappy - neutral AI_HMM_vol           TRUE   low_D -0.02971973 0.01189031
## 9    neutral - happy     AI_HMM           TRUE  high_D -0.02793067 0.01194555
## 10 unhappy - neutral     AI_HMM          FALSE   low_D -0.04181277 0.01835595
##     df   z.ratio      p.value effect_size significant
## 1  Inf -5.349907 2.636061e-07   -5.349907        TRUE
## 2  Inf -4.360487 3.859308e-05   -4.360487        TRUE
## 3  Inf -4.065509 1.417836e-04   -4.065509        TRUE
## 4  Inf -3.273989 3.044719e-03   -3.273989        TRUE
## 5  Inf -3.193409 4.017241e-03   -3.193409        TRUE
## 6  Inf -3.042247 6.638861e-03   -3.042247        TRUE
## 7  Inf -2.778911 1.507092e-02   -2.778911        TRUE
## 8  Inf -2.499492 3.328726e-02   -2.499492        TRUE
## 9  Inf -2.338166 5.070449e-02   -2.338166       FALSE
## 10 Inf -2.277886 5.892855e-02   -2.277886       FALSE
```

Question 1: Adaptation to HMM volatility

Creates a moving window correlation between participant returns and investor states Compares adaptation scores between high and low D-factor participants

```r
# Load required libraries
library(dplyr)
library(tidyr)
library(ggplot2)
library(lme4)
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode

## The following object is masked from 'package:purrr':
##
##     some
```

```r
# Question 1: Adaptation to HMM volatility
# We'll measure adaptation by calculating how well participants adjust their
# return rates in response to the investor's state changes

# First, create a function to calculate moving average correlation
calc_moving_correlation <- function(df, window_size = 5) {
  df %>%
    group_by(playerId, opponent.f) %>%
    arrange(roundNum) %>%
    mutate(
      # Create moving averages for returns and investor state
      ma_return = zoo::rollmean(return, k = window_size, fill = NA),
      ma_state = zoo::rollmean(as.numeric(investorState.f), k = window_size, fill = NA)
    ) %>%
    # Calculate correlation between these moving averages
    summarize(
      adaptation_score = cor(ma_return, ma_state, use = "complete.obs"),
      d_level = first(d_level)
    )
}

# Calculate adaptation scores
adaptation_scores <- calc_moving_correlation(final_data)
```

```
## Warning: There were 8 warnings in `summarize()`.
## The first warning was:
## i In argument: `adaptation_score = cor(ma_return, ma_state, use =
##   "complete.obs")`.
## i In group 37: `playerId = "9NZPdsRHMfTffJ86Z"` and `opponent.f = AI_HMM`.
## Caused by warning in `cor()`:
## ! the standard deviation is zero
## i Run `dplyr::last_dplyr_warnings()` to see the 7 remaining warnings.

## `summarise()` has grouped output by 'playerId'. You can override using the
## `.groups` argument.
```

```r
# Compare adaptation between high and low D-factor participants
adaptation_analysis <- mixed(adaptation_score ~ d_level*opponent.f + (1 | playerId), data = adaptation_s
```

```
## Contrasts set to contr.sum for the following variables: d_level, opponent.f, playerId

## Warning: Due to missing values, reduced number of observations to 358

## Fitting one lmer() model. [DONE]
## Calculating p-values. [DONE]
```

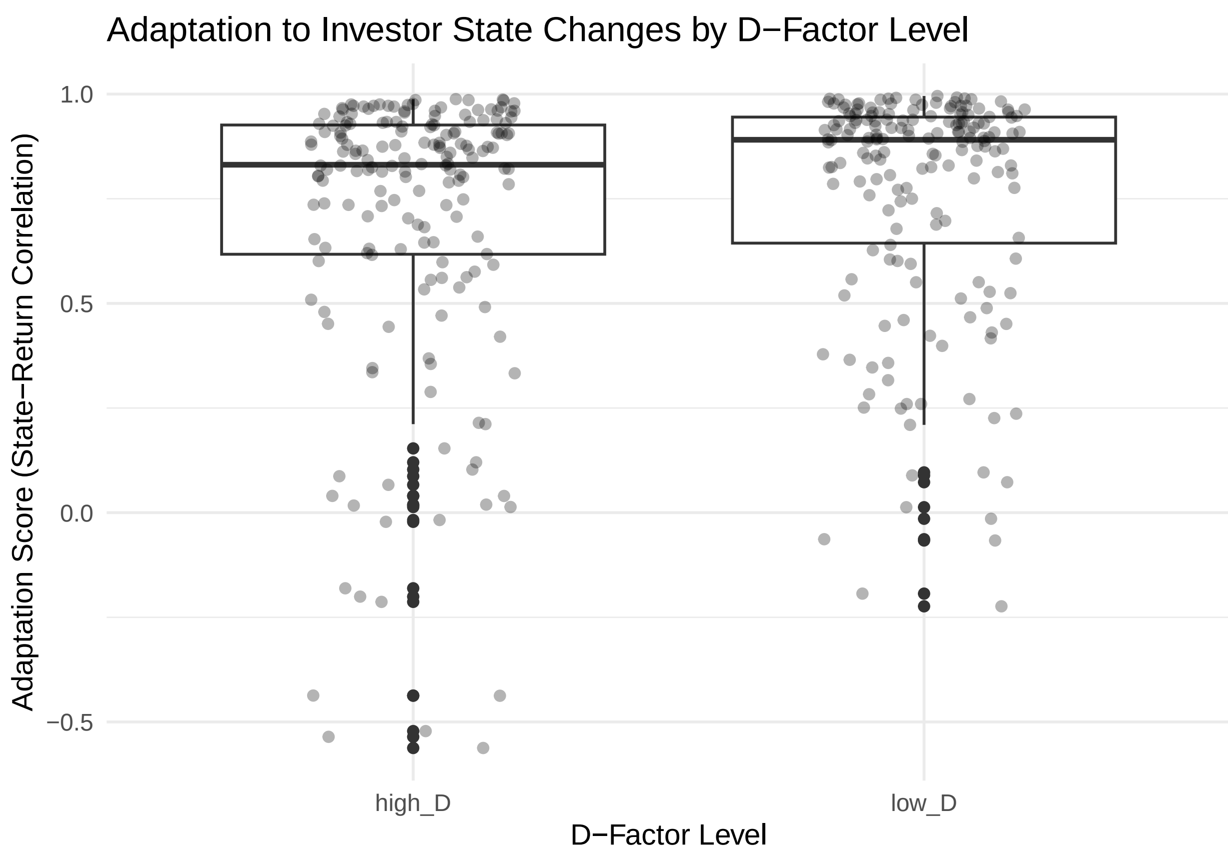
```r
summary(adaptation_analysis)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: adaptation_score ~ d_level * opponent.f + (1 | playerId)
##    Data: data
##
## REML criterion at convergence: 202.7
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.5644 -0.3651  0.3588  0.5711  1.3311
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  playerId (Intercept) 0.01937  0.1392
##  Residual             0.07964  0.2822
## Number of obs: 358, groups:  playerId, 183
##
## Fixed effects:
##                        Estimate Std. Error         df t value Pr(>|t|)
## (Intercept)            0.727852   0.018147 182.723070  40.108   <2e-16 ***
## d_level1              -0.029631   0.018147 182.723070  -1.633    0.104
## opponent.f1            0.004645   0.014950 181.133824   0.311    0.756
## d_level1:opponent.f1   0.011281   0.014950 181.133824   0.755    0.451
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) d_lvl1 oppn.1
## d_level1    -0.004
## opponent.f1  0.016 -0.006
## d_lvl1:pp.1 -0.006  0.016 -0.008
```

```r
# Create visualization
ggplot(adaptation_scores, aes(x = d_level, y = adaptation_score)) +
  geom_boxplot() +
  geom_jitter(width = 0.2, alpha = 0.3) +
  theme_minimal() +
  labs(
    title = "Adaptation to Investor State Changes by D-Factor Level",
    x = "D-Factor Level",
    y = "Adaptation Score (State-Return Correlation)"
  )
```

```
## Warning: Removed 8 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```

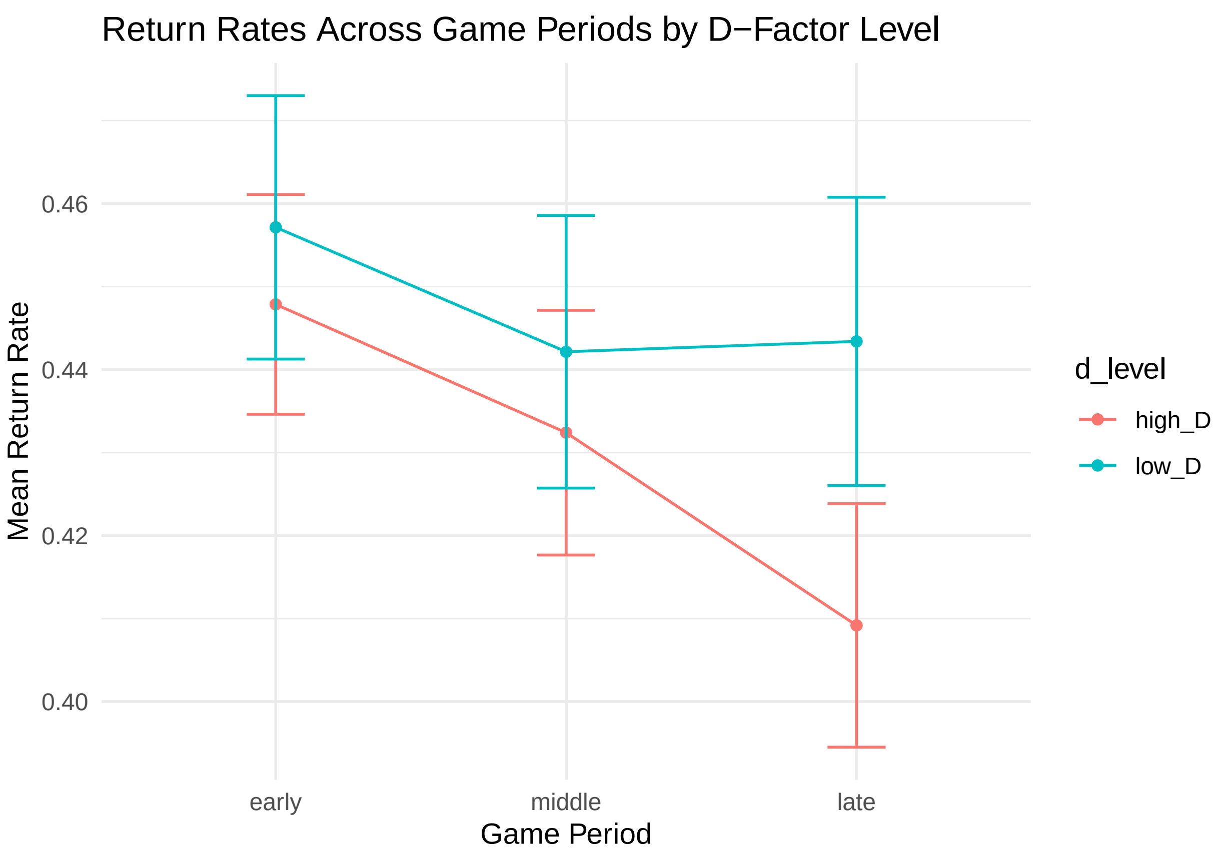
```
## Warning: Removed 8 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



Question 2: Strategic reputation building and exploitation

```
# Analyze how return rates change from early to late rounds

# Create period indicators
data_with_periods <- final_data %>%
  group_by(playerId, gameNum.f) %>%
  mutate(
    game_period = factor(case_when(
      roundNum <= 8 ~ "early",
      roundNum <= 16 ~ "middle",
      TRUE ~ "late"
    ), levels= c("early", "middle", "late"))
  )

# Calculate average returns by period
period_returns <- data_with_periods %>%
  group_by(playerId, d_level, game_period) %>%
  summarize(
    mean_return = mean(return_pct),
    .groups = "drop"
  )

# Run repeated measures ANOVA
period_model <- aov(mean_return ~ d_level * game_period + Error(playerId/game_period),
```

```r
                    data = period_returns)

# Visualization for reputation building
ggplot(period_returns, aes(x = game_period, y = mean_return, color = d_level, group = d_level)) +
  stat_summary(fun = mean, geom = "line") +
  stat_summary(fun = mean, geom = "point") +
  stat_summary(fun.data = mean_se, geom = "errorbar", width = 0.2) +
  theme_minimal() +
  labs(
    title = "Return Rates Across Game Periods by D-Factor Level",
    x = "Game Period",
    y = "Mean Return Rate"
  )
```



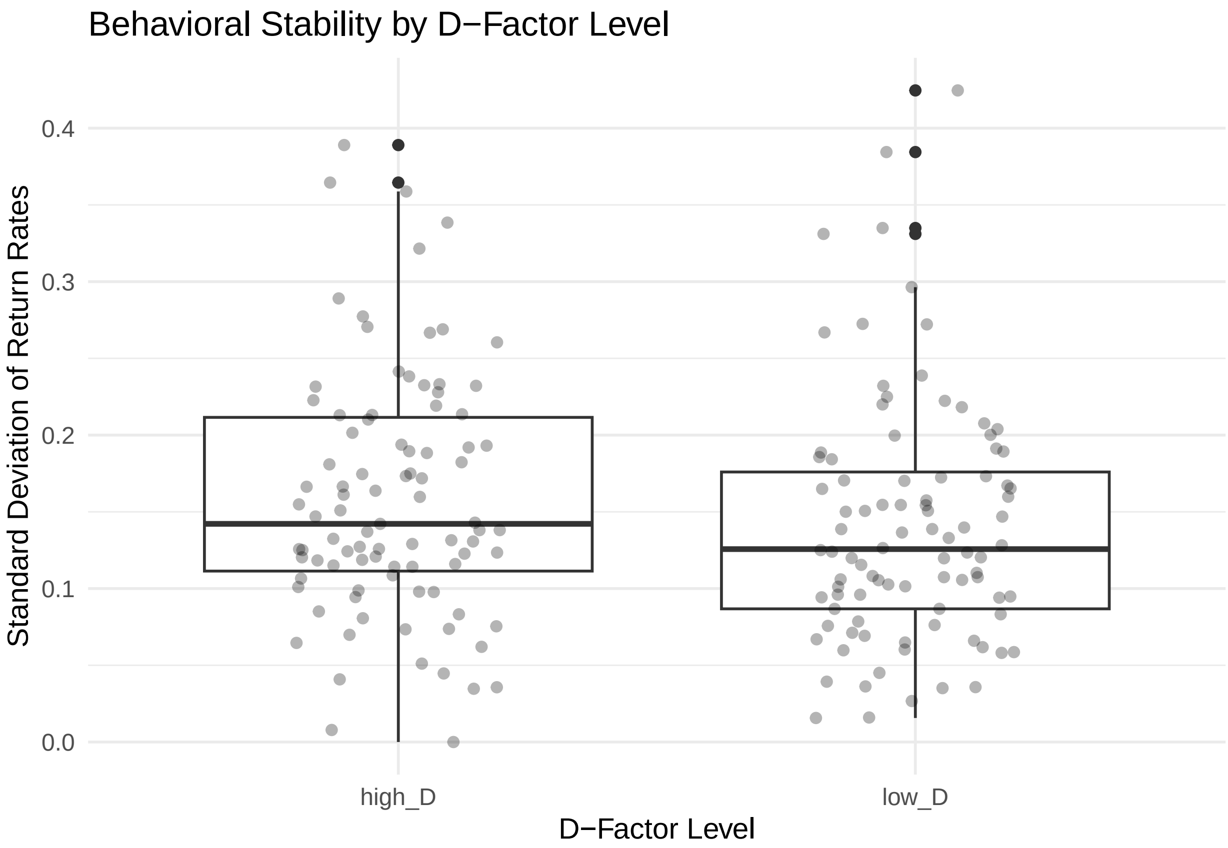
Question 3: Behavioral stability across rounds

```r
# Calculate standard deviation of return rates

stability_scores <- final_data %>%
  group_by(playerId, d_level) %>%
  summarize(
    return_sd = sd(return_pct),
    .groups = "drop"
  )
```

```r
# Test relationship between D-factor and behavioral stability
stability_test <- t.test(return_sd ~ d_level, data = stability_scores)
stability_test
```

```
##
##  Welch Two Sample t-test
##
## data:  return_sd by d_level
## t = 1.5311, df = 180.94, p-value = 0.1275
## alternative hypothesis: true difference in means between group high_D and group low_D is not equal to
## 95 percent confidence interval:
##  -0.005193469  0.041167924
## sample estimates:
## mean in group high_D  mean in group low_D
##            0.1587024           0.1407151
```

```r
# Visualization for behavioral stability
ggplot(stability_scores, aes(x = d_level, y = return_sd)) +
  geom_boxplot() +
  geom_jitter(width = 0.2, alpha = 0.3) +
  theme_minimal() +
  labs(
    title = "Behavioral Stability by D-Factor Level",
    x = "D-Factor Level",
    y = "Standard Deviation of Return Rates"
  )
```

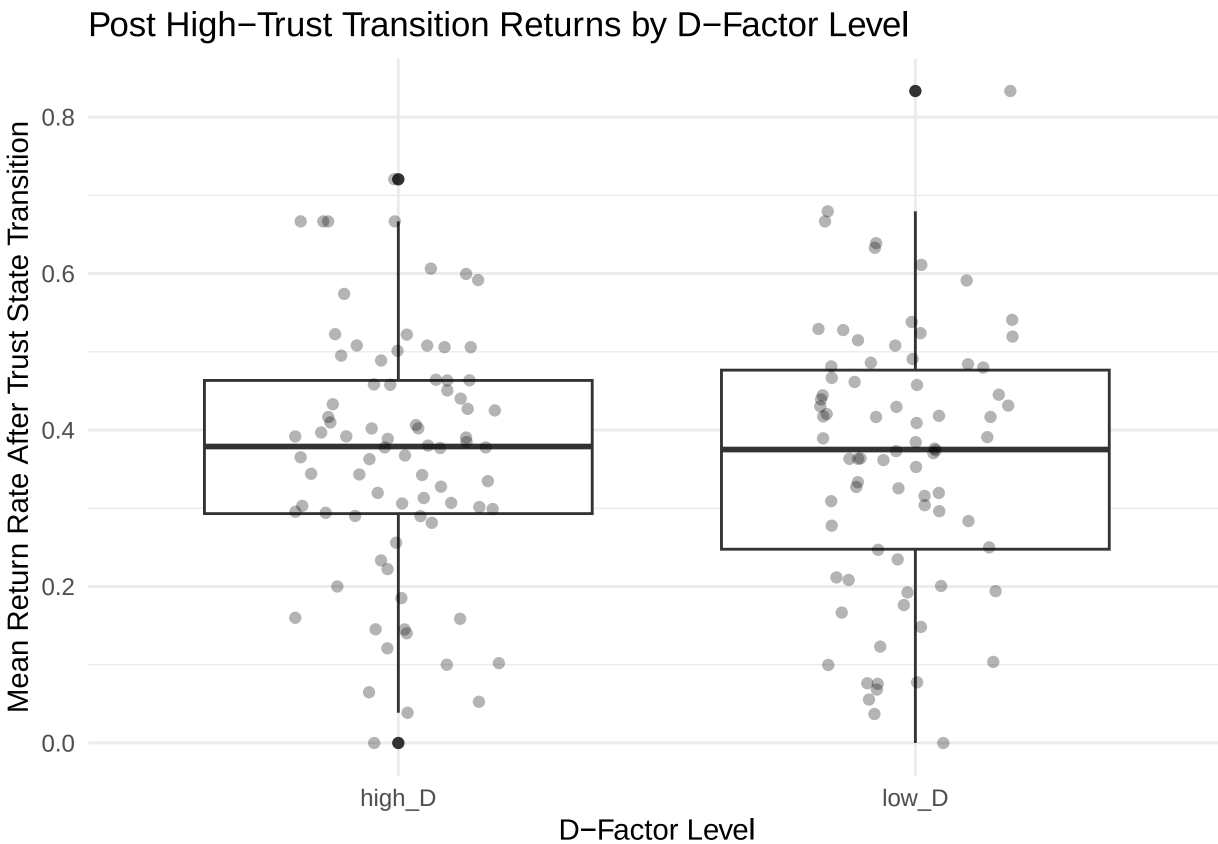


Question 4: Exploitation of high trust states

```r
# Identify transitions to high trust state and subsequent behavior
high_trust_exploitation <- final_data %>%
  group_by(playerId) %>%
  arrange(roundNum) %>%
  mutate(
    state_change = investorState.f != lag(investorState.f),
    to_high_trust = state_change & investorState.f == "happy",
    to_low_trust = state_change & investorState.f == "unhappy",
  ) %>%
  # filter(to_high_trust == TRUE) %>%
  filter(to_low_trust == TRUE) %>%
  group_by(playerId, d_level) %>%
  summarize(
    mean_post_transition_return = mean(return_pct, na.rm = TRUE),
    .groups = "drop"
  )

# Test for exploitation differences
exploitation_test <- t.test(mean_post_transition_return ~ d_level,
                            data = high_trust_exploitation)
exploitation_test
```

```
##
##  Welch Two Sample t-test
##
## data:  mean_post_transition_return by d_level
## t = 0.19009, df = 154.6, p-value = 0.8495
## alternative hypothesis: true difference in means between group high_D and group low_D is not equal t
## 95 percent confidence interval:
##  -0.04674328  0.05669721
## sample estimates:
## mean in group high_D  mean in group low_D
##            0.3675943            0.3626173
```

```r
# Visualization for exploitation analysis
ggplot(high_trust_exploitation, aes(x = d_level, y = mean_post_transition_return)) +
  geom_boxplot() +
  geom_jitter(width = 0.2, alpha = 0.3) +
  theme_minimal() +
  labs(
    title = "Post High-Trust Transition Returns by D-Factor Level",
    x = "D-Factor Level",
    y = "Mean Return Rate After Trust State Transition"
  )
```



Focusing on participants whose score is higher than 55 to define high_d

```
# Read and prepare the data with new D-factor classification
new_data <- final_data %>%
  group_by(playerId) %>%
  mutate(
    avg_total_score = mean(total_score),
    new_d_factor = factor(case_when(
      avg_total_score >= 52 ~ "top_d",
      avg_total_score <= 22 ~ "bottom_d",
      TRUE ~ NA_character_
    ), levels = c("top_d","bottom_d"))
  ) %>%
  ungroup() %>%
  filter(!is.na(new_d_factor))

# Print number of participants in each group after filtering
print("Number of participants in each group after filtering:")
```

```
## [1] "Number of participants in each group after filtering:"
```

```
new_data %>%
  dplyr::select(playerId, new_d_factor) %>%
  distinct() %>%
```

```r
  count(new_d_factor) %>%
  print()
```

```
## # A tibble: 2 x 2
##   new_d_factor     n
##   <fct>        <int>
## 1 top_d           15
## 2 bottom_d        92
```

```r
# Calculate mean scores for each group to verify separation
score_summary <- new_data %>%
  group_by(new_d_factor) %>%
  summarize(
    mean_score = mean(avg_total_score),
    sd_score = sd(avg_total_score),
    n_players = n_distinct(playerId)
  )

print("\nScore summary by group:")
```

```
## [1] "\nScore summary by group:"
```

```r
print(score_summary)
```

```
## # A tibble: 2 x 4
##   new_d_factor mean_score sd_score n_players
##   <fct>             <dbl>    <dbl>     <int>
## 1 top_d              56.5     5.96        15
## 2 bottom_d           19.5     1.87        92
```

```r
mod_returns_new_d <- mixed( ret_pct_na ~ opponent.f*investorState.f*new_d_factor*volatile_first + (1 | p
```

```
## Contrasts set to contr.sum for the following variables: opponent.f, investorState.f, new_d_factor, pl
```

```
## Warning: Due to missing values, reduced number of observations to 5195
```

```
## Fitting one lmer() model. [DONE]
## Calculating p-values. [DONE]
```

```r
summary(mod_returns_new_d)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula:
## ret_pct_na ~ opponent.f * investorState.f * new_d_factor * volatile_first +
##     (1 | playerId)
##    Data: data
##
## REML criterion at convergence: -4166.6
```

```
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -4.3053 -0.3100  0.0488  0.3851  4.9916
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  playerId (Intercept) 0.01803  0.1343
##  Residual             0.02356  0.1535
## Number of obs: 5195, groups:  playerId, 107
##
## Fixed effects:
##                                                               Estimate
## (Intercept)                                                  4.288e-01
## opponent.f1                                                  1.309e-03
## investorState.f1                                            -3.647e-02
## investorState.f2                                             2.052e-02
## new_d_factor1                                               -2.205e-02
## volatile_firstTRUE                                           2.970e-02
## opponent.f1:investorState.f1                                -1.028e-02
## opponent.f1:investorState.f2                                -1.307e-02
## opponent.f1:new_d_factor1                                    3.633e-04
## investorState.f1:new_d_factor1                               7.069e-03
## investorState.f2:new_d_factor1                               1.504e-02
## opponent.f1:volatile_firstTRUE                               1.207e-03
## investorState.f1:volatile_firstTRUE                          2.241e-02
## investorState.f2:volatile_firstTRUE                         -2.641e-02
## new_d_factor1:volatile_firstTRUE                             2.432e-02
## opponent.f1:investorState.f1:new_d_factor1                  -1.763e-02
## opponent.f1:investorState.f2:new_d_factor1                  -6.565e-03
## opponent.f1:investorState.f1:volatile_firstTRUE              1.787e-02
## opponent.f1:investorState.f2:volatile_firstTRUE              1.508e-02
## opponent.f1:new_d_factor1:volatile_firstTRUE                -5.079e-04
## investorState.f1:new_d_factor1:volatile_firstTRUE            4.294e-03
## investorState.f2:new_d_factor1:volatile_firstTRUE           -2.303e-02
## opponent.f1:investorState.f1:new_d_factor1:volatile_firstTRUE  1.469e-02
## opponent.f1:investorState.f2:new_d_factor1:volatile_firstTRUE  4.636e-03
##                                                              Std. Error
## (Intercept)                                                   2.790e-02
## opponent.f1                                                   4.915e-03
## investorState.f1                                              8.106e-03
## investorState.f2                                              6.593e-03
## new_d_factor1                                                 2.790e-02
## volatile_firstTRUE                                            3.809e-02
## opponent.f1:investorState.f1                                  8.247e-03
## opponent.f1:investorState.f2                                  6.720e-03
## opponent.f1:new_d_factor1                                     4.915e-03
## investorState.f1:new_d_factor1                                8.106e-03
## investorState.f2:new_d_factor1                                6.593e-03
## opponent.f1:volatile_firstTRUE                                6.705e-03
## investorState.f1:volatile_firstTRUE                           1.104e-02
## investorState.f2:volatile_firstTRUE                           9.233e-03
## new_d_factor1:volatile_firstTRUE                              3.809e-02
## opponent.f1:investorState.f1:new_d_factor1                    8.247e-03
```

```
## opponent.f1:investorState.f2:new_d_factor1                        6.720e-03
## opponent.f1:investorState.f1:volatile_firstTRUE                   1.057e-02
## opponent.f1:investorState.f2:volatile_firstTRUE                   9.356e-03
## opponent.f1:new_d_factor1:volatile_firstTRUE                      6.705e-03
## investorState.f1:new_d_factor1:volatile_firstTRUE                 1.104e-02
## investorState.f2:new_d_factor1:volatile_firstTRUE                 9.233e-03
## opponent.f1:investorState.f1:new_d_factor1:volatile_firstTRUE  1.057e-02
## opponent.f1:investorState.f2:new_d_factor1:volatile_firstTRUE  9.356e-03
##                                                                       df
## (Intercept)                                                       1.023e+02
## opponent.f1                                                       5.074e+03
## investorState.f1                                                  5.130e+03
## investorState.f2                                                  5.089e+03
## new_d_factor1                                                     1.023e+02
## volatile_firstTRUE                                                1.021e+02
## opponent.f1:investorState.f1                                      5.137e+03
## opponent.f1:investorState.f2                                      5.100e+03
## opponent.f1:new_d_factor1                                         5.074e+03
## investorState.f1:new_d_factor1                                    5.130e+03
## investorState.f2:new_d_factor1                                    5.089e+03
## opponent.f1:volatile_firstTRUE                                    5.082e+03
## investorState.f1:volatile_firstTRUE                               5.145e+03
## investorState.f2:volatile_firstTRUE                               5.094e+03
## new_d_factor1:volatile_firstTRUE                                  1.021e+02
## opponent.f1:investorState.f1:new_d_factor1                        5.137e+03
## opponent.f1:investorState.f2:new_d_factor1                        5.100e+03
## opponent.f1:investorState.f1:volatile_firstTRUE                   5.126e+03
## opponent.f1:investorState.f2:volatile_firstTRUE                   5.101e+03
## opponent.f1:new_d_factor1:volatile_firstTRUE                      5.082e+03
## investorState.f1:new_d_factor1:volatile_firstTRUE                 5.145e+03
## investorState.f2:new_d_factor1:volatile_firstTRUE                 5.094e+03
## opponent.f1:investorState.f1:new_d_factor1:volatile_firstTRUE  5.126e+03
## opponent.f1:investorState.f2:new_d_factor1:volatile_firstTRUE  5.101e+03
##                                                                t value Pr(>|t|)
## (Intercept)                                                     15.368  < 2e-16
## opponent.f1                                                      0.266  0.78998
## investorState.f1                                                -4.499 6.97e-06
## investorState.f2                                                 3.113  0.00186
## new_d_factor1                                                   -0.790  0.43114
## volatile_firstTRUE                                               0.780  0.43735
## opponent.f1:investorState.f1                                    -1.247  0.21247
## opponent.f1:investorState.f2                                    -1.945  0.05183
## opponent.f1:new_d_factor1                                        0.074  0.94108
## investorState.f1:new_d_factor1                                   0.872  0.38322
## investorState.f2:new_d_factor1                                   2.281  0.02260
## opponent.f1:volatile_firstTRUE                                   0.180  0.85721
## investorState.f1:volatile_firstTRUE                              2.031  0.04233
## investorState.f2:volatile_firstTRUE                             -2.861  0.00424
## new_d_factor1:volatile_firstTRUE                                 0.639  0.52452
## opponent.f1:investorState.f1:new_d_factor1                      -2.138  0.03260
## opponent.f1:investorState.f2:new_d_factor1                      -0.977  0.32862
## opponent.f1:investorState.f1:volatile_firstTRUE                  1.690  0.09118
## opponent.f1:investorState.f2:volatile_firstTRUE                  1.612  0.10702
## opponent.f1:new_d_factor1:volatile_firstTRUE                    -0.076  0.93962
```

```
## investorState.f1:new_d_factor1:volatile_firstTRUE               0.389  0.69722
## investorState.f2:new_d_factor1:volatile_firstTRUE              -2.494  0.01267
## opponent.f1:investorState.f1:new_d_factor1:volatile_firstTRUE   1.389  0.16494
## opponent.f1:investorState.f2:new_d_factor1:volatile_firstTRUE   0.495  0.62028
##
## (Intercept)                                                     ***
## opponent.f1
## investorState.f1                                                ***
## investorState.f2                                                **
## new_d_factor1
## volatile_firstTRUE
## opponent.f1:investorState.f1
## opponent.f1:investorState.f2                                    .
## opponent.f1:new_d_factor1
## investorState.f1:new_d_factor1
## investorState.f2:new_d_factor1                                  *
## opponent.f1:volatile_firstTRUE
## investorState.f1:volatile_firstTRUE                             *
## investorState.f2:volatile_firstTRUE                             **
## new_d_factor1:volatile_firstTRUE
## opponent.f1:investorState.f1:new_d_factor1                      *
## opponent.f1:investorState.f2:new_d_factor1
## opponent.f1:investorState.f1:volatile_firstTRUE                 .
## opponent.f1:investorState.f2:volatile_firstTRUE
## opponent.f1:new_d_factor1:volatile_firstTRUE
## investorState.f1:new_d_factor1:volatile_firstTRUE
## investorState.f2:new_d_factor1:volatile_firstTRUE               *
## opponent.f1:investorState.f1:new_d_factor1:volatile_firstTRUE
## opponent.f1:investorState.f2:new_d_factor1:volatile_firstTRUE
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Correlation matrix not shown by default, as p = 24 > 12.
## Use print(x, correlation=TRUE)  or
##     vcov(x)        if you need it
```

```r
# Payoff regression with new d factor
new_payoff_data <- new_data %>%
  dplyr::select(playerId, new_d_factor, opponent.f, gameNum.f, volatile_first, payoffTrust1, payoffTrus
  distinct() %>%
   mutate(
      payoff = case_when(
        gameNum.f == "first game" ~ payoffTrust1,
        gameNum.f == "second game" ~ payoffTrust2
      )
    ) %>%
    dplyr::select(-payoffTrust1, -payoffTrust2)  # Remove unused columns


# fit lmem
mod_payoffs_new <- mixed( payoff ~ opponent.f*new_d_factor*volatile_first + (1| playerId), new_payoff_da
```

```
## Contrasts set to contr.sum for the following variables: opponent.f, new_d_factor, playerId
```

```
## Fitting one lmer() model. [DONE]
## Calculating p-values. [DONE]
```

```r
summary(mod_payoffs_new)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: payoff ~ opponent.f * new_d_factor * volatile_first + (1 | playerId)
##    Data: data
##
## REML criterion at convergence: 2551
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.21315 -0.77114 -0.05279  0.59228  2.64799
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  playerId (Intercept)   939     30.64
##  Residual             10929    104.54
## Number of obs: 214, groups:  playerId, 107
##
## Fixed effects:
##                                              Estimate Std. Error      df
## (Intercept)                                   411.646     16.363 103.000
## opponent.f1                                    30.064     15.115 103.000
## new_d_factor1                                 -10.146     16.363 103.000
## volatile_firstTRUE                            -40.187     22.343 103.000
## opponent.f1:new_d_factor1                       6.150     15.115 103.000
## opponent.f1:volatile_firstTRUE                  8.996     20.640 103.000
## new_d_factor1:volatile_firstTRUE                1.125     22.343 103.000
## opponent.f1:new_d_factor1:volatile_firstTRUE   19.352     20.640 103.000
##                                              t value Pr(>|t|)
## (Intercept)                                   25.158   <2e-16 ***
## opponent.f1                                    1.989   0.0494 *
## new_d_factor1                                 -0.620   0.5366
## volatile_firstTRUE                            -1.799   0.0750 .
## opponent.f1:new_d_factor1                      0.407   0.6850
## opponent.f1:volatile_firstTRUE                 0.436   0.6639
## new_d_factor1:volatile_firstTRUE               0.050   0.9599
## opponent.f1:new_d_factor1:volatile_firstTRUE   0.938   0.3507
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) oppn.1 nw_d_1 v_TRUE op.1:__1 o.1:_T n__1:_
## opponent.f1  0.000
## new_d_fctr1  0.708  0.000
## vltl_frTRUE -0.732  0.000 -0.519
## oppnn.1:__1  0.000  0.708  0.000  0.000
## opp.1:_TRUE  0.000 -0.732  0.000  0.000 -0.519
## nw__1:_TRUE -0.519  0.000 -0.732  0.718  0.000    0.000
## o.1:__1:_TR  0.000 -0.519  0.000  0.000 -0.732    0.718  0.000
```

Discussion

Contrary to our expectations, high D-factor participants did not demonstrate strategic exploitation across different investor states. The literature suggests that individuals high in dark personality traits, particularly the Machiavellian aspect of the D-factor, should show strategic adaptation to maximize personal gain, potentially through initial trust-building followed by exploitation. However, our results reveal an opposing pattern: high D-factor participants showed relatively stable returns across investor states, particularly with the volatile investor, suggesting a form of behavioral rigidity rather than strategic flexibility. This behavioral inflexibility was especially evident in the volatile HMM condition, where high D-factor participants maintained consistent return rates regardless of the investor's state. In contrast, low D-factor participants showed marked sensitivity to investor states, adjusting their returns upward as the investor's state improved from unhappy to happy. This pattern held across both human-like and volatile HMM conditions, though it was more pronounced with the volatile investor. These findings suggest that contrary to the Machiavellian tendency for strategic manipulation, high D-factor individuals might actually be less adept at reading and responding to social cues in economic interactions.

One possible explanation for this unexpected pattern lies in the fundamental nature of the D-factor as "the tendency to maximize one's individual utility at the expense of others with self-justifying beliefs." The behavioral rigidity we observed might represent a form of defensive strategy - by maintaining stable (and relatively lower) returns regardless of the investor's state, high D-factor participants could be prioritizing consistent personal gain over reciprocity. This interpretation aligns with recent work suggesting that dark personality traits might manifest not just as active exploitation, but as a general insensitivity to social cues that would typically motivate cooperative behavior. The finding that low D-factor participants showed greater behavioral flexibility and responsiveness to investor states suggests that the ability to maintain cooperative relationships might require active engagement with partner behavior rather than strategic manipulation. This has important implications for understanding how personality traits influence economic decision-making and challenges the traditional view of dark personality traits as primarily manifesting through strategic exploitation.

Conclusion