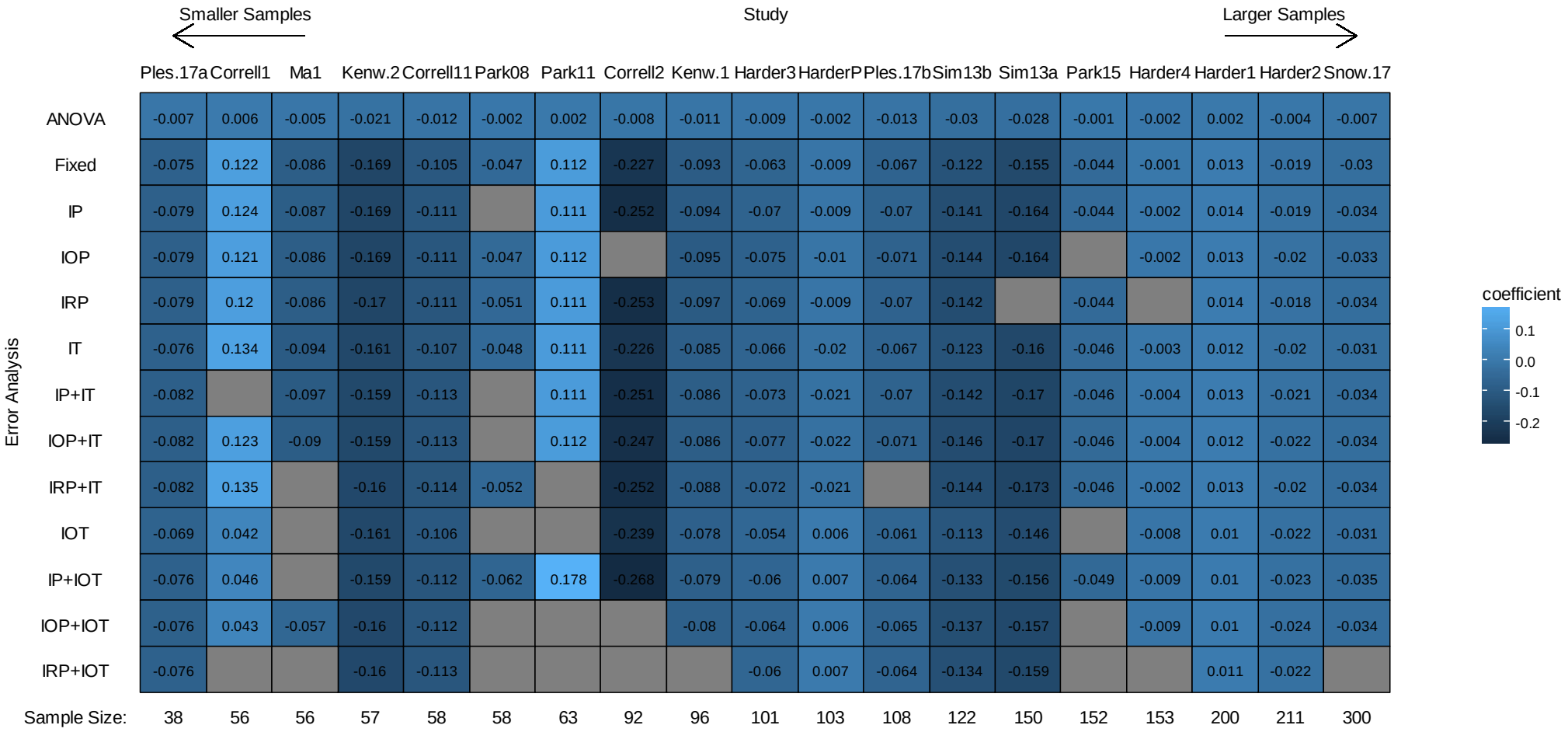

Figure S1. Results from 13 analyses predicting errors in 19 studies, ordered by sample size. Values indicate coefficients of the race-by-object interaction representing racial bias in shooting decisions. Positive values indicate that more errors were made when targets were stereotype-congruent, and negative values indicate that more errors were made when targets were stereotype-incongruent. See Table 1 in the main manuscript for key to abbreviated study citations. See Table 2 for key to abbreviated analysis descriptions. Cells representing nonsignificant results are shaded light gray and unlabeled; otherwise, cells are shaded according to the size of the coefficient.

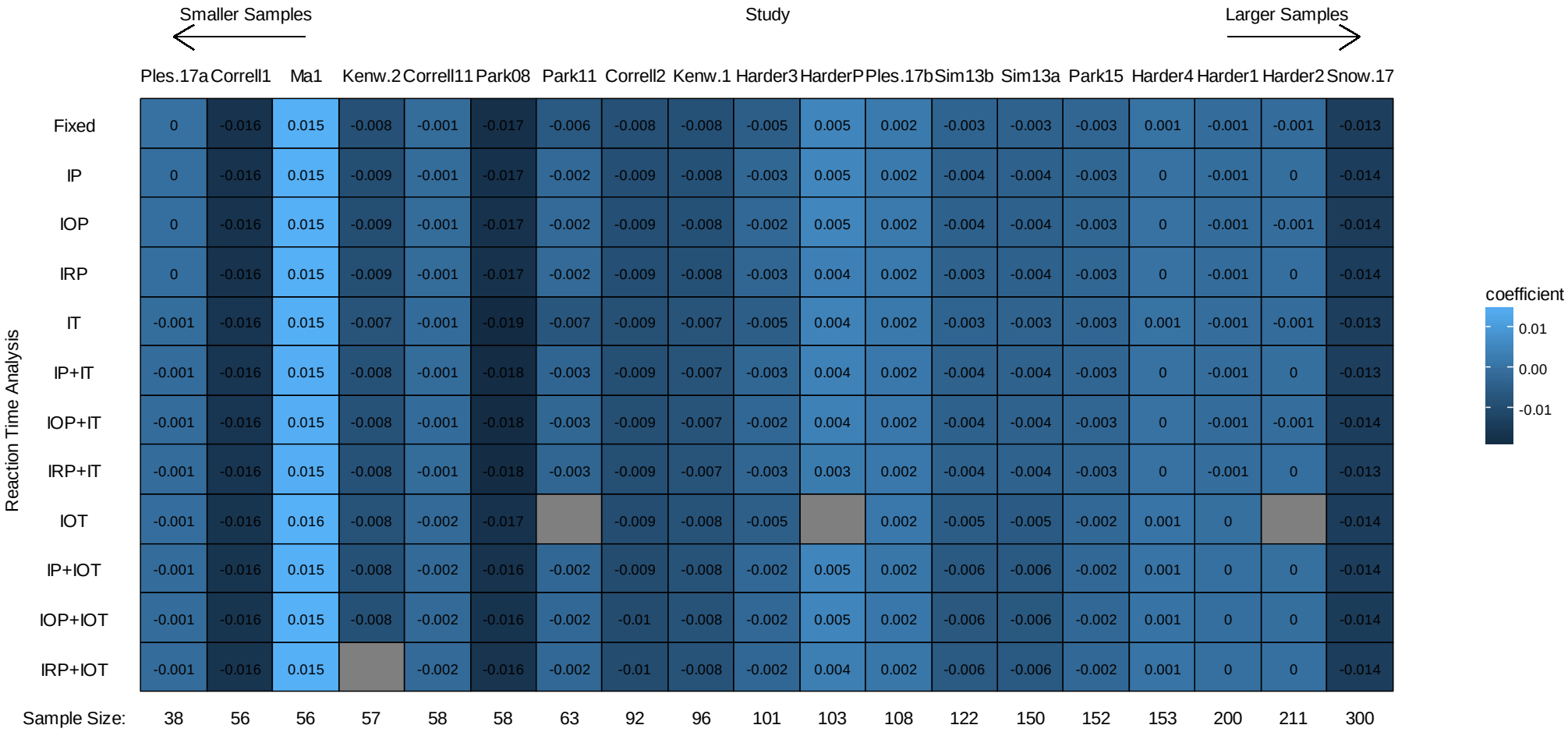

Figure S2. Results from 12 analyses predicting reaction times in 19 studies, ordered by sample size. Values indicate coefficients of the race-by-object interaction representing racial bias in shooting decisions. Positive values indicate that responses were slower when targets were stereotype-congruent, and negative values indicate that responses were slower when targets were stereotype-incongruent. See Table 1 in the main manuscript for key to abbreviated study citations. See Table 2 for key to abbreviated analysis descriptions. Cells representing nonsignificant results are shaded light gray and unlabeled; otherwise, cells are shaded according to the size of the coefficient.

R Code for Multiverse Analysis

```
#Load packages
library(ggplot2)
library(Rmisc)
library(lme4)
library(car)

#Read in data
datafrm <- read.csv("alldata3.csv", stringsAsFactors = F, header = T)

#change settings so that R runs the ANOVAs correctly
options("contrasts")
options(contrasts = c("contr.helmert", "contr.poly"))

#Prep Work####

#create list of formulae
#fixed effects portions:
anteER <- "error_dummy ~ group + object_effects + group:object_effects " #for
error analyses
anteRT <- "RT2 ~ group + object_effects + group:object_effects " #for RT
analyses
anteRT2 <- "RT2 ~ group + object_effects " #for model comparisons to get p-
values for RT analyses using the lme4 package
#participant random effects portions:
PrandomEffects <- c("", "+ (1|s_pnum)", "+ (1+object_effects|s_pnum)", "+
(1+group|s_pnum)")
#^but not the object:race interaction since I want to know average across
participants
#nor obj + group, because it's too computationally intensive: the software
just can't handle it (doesn't converge).
#target random effects portions:
TrandomEffects <- c("", "+ (1|target)","+ (1+object_effects|target)")
#create vector of all the random effects formulas we'll use:
randomEffects <- vector()
for(p in(PrandomEffects)){
  for(t in(TrandomEffects)){
    randomEffects <- c(randomEffects, paste(p, t))
  }
}
randomEffects <- randomEffects[-1]  #the first one is blank so get rid of
that

#create vector of dataset names
datasets<-unique(datafrm$study)

#create data frame with summarized error data, which we'll use to run error
rate anova
anovadf <- data.frame(matrix(nrow=1, ncol=5)) #row of 5 NAs:  we'll remove
this row later
names(anovadf) <- c("s_pnum", "group", "object_effects", "error_rate",
"study")
for(dset in(datasets)) {
  df <- datafrm[which(datafrm$study==dset),]  #isolate data for each dataset
we're summarizing
```

```
  rm(temp)
  #make temporary summary dataset with four rows for each of current
dataset's participants:
  temp <- data.frame(matrix(nrow=length(unique(df$s_pnum))*4, ncol=5)) #make
temporary summary dataset with four rows for each of current dataset's
participants
  names(temp) <- c("s_pnum", "group", "object_effects", "error_rate",
"study")
  temp[,1] <- rep(unique(df$s_pnum), each = 4) #first column: participant
numbers
  temp[,2] <- rep(c(-1, 1), each = 2) #second column:  effects coded levels
of race
  temp[,3] <- rep(c(-1, 1), each = 1) #third column:  effects coded levels of
object
  #calculate mean error rate for each race:object combination for each
participant in dataset:
  for(s in(unique(df$s_pnum))){ #(for each participant)
    for(tr in(c(-1, 1))){ #(for each target race level)
      for(ob in(c(-1, 1))){   #(for each object level)
        temp[which(temp$s_pnum==s & temp$group==tr &
temp$object_effects==ob), "error_rate"] <- mean(df[which(df$s_pnum==s &
df$group==tr & df$object_effects==ob), "error_dummy"], na.rm=T)
      }
    }
  }
  #label which study this was
  temp$study <- unique(df$study)
  #and add it to our larger dataframe (including all the datasets we're
using)
  anovadf <- rbind(anovadf, temp)
}
#you will get a warning saying "In rm(temp) : object 'temp' not found." This
is not a problem.
anovadf <- anovadf[2:nrow(anovadf),] #remove that first row of NAs we put in
earlier


#fix RT data
datafrm$RT <- as.numeric(datafrm$RT) #variable must be numeric
datafrm$RT2 <- ifelse(datafrm$error_dummy==0, datafrm$RT, NA)  #only correct
trials
datafrm$RT2 <- log(datafrm$RT2)  #log-transform RT values

#have to adjust a few more things to make R do the ANOVA correctly:
datafrm$group <- as.factor(datafrm$group)
datafrm$object_effects <- as.factor(datafrm$object_effects)

#get rid of dataless rows
datafrm <- datafrm[which(!is.na(datafrm$error_dummy)),]

#Create Output Dataframe####
      #this is where the coefficients and p-values we generate will go

#one row per analysis per dataset
output <- data.frame(matrix(nrow = ((3 +
2*length(randomEffects))*length(datasets)), ncol = 5))
names(output)<-c("Study", "Analysis", "text", "coefficient", "pval")
```

```
#column naming/numbering each of the analyses
analysesRan <- 1:length(randomEffects)
analyses <- c("rtFixed", paste0("rtRandom", analysesRan), 'errorFixed',
paste0("errorRandom",analysesRan), "errorANOVA")
output$Analysis<-analyses

#column indicating the random effects structure of each analysis
output$text <- c("<fixed effects only>", randomEffects, "<fixed effects
only>", randomEffects, "<n/a>")


#Multiverse####
startTime <- proc.time()
#prep
datasetvec <-  1:length(datasets) #we'll cycle through each dataset
lrf<-length(randomEffects)  #and each of the random effects specifications.
for(dset in(datasetvec)){  #Name each study in the output file
  output$Study[which(is.na(output$Study))[1]:(dset*(lrf*2+3))] <-
datasets[dset]
}
#multiverse
tempmat <- sapply(c(datasetvec), FUN = function(dset){
  #tell the user which dataset we're on, so they can track our progress:
  print(paste(dset, datasets[dset], sep=": "))
  #create empty vectors which we'll populate with coefficients and p-values
for RT and error analyses:
  outputrt <- vector()
  outputer <- vector()
  #select the dataset to analyze:
  df<-datafrm[which(datafrm$study==datasets[dset]),]

  print("rt analyses") #tell the user we're doing RT analyses now

  #run the reaction time analysis with only fixed effects and get coefficient
and p-value
  print("fixed effects") #tell the user which analysis we're on
  #create function to run analysis for this random effects structure:
  fmlaRT <- as.formula(anteRT)
  RTfixed <- function(fmla){
    return(tryCatch( #if we get a warning, it's probably because analysis
didn't converge.
      warning=function(w){
        #so in that case, print warning message:
        print("Regression (RT data) did not converge")
        #and return NAs for that analysis:
        return(c(NA,NA))
      },
      #but if no warning, return coefficient and p-value from object:race
interaction:
      expr = c(summary(lm(fmla, data=df))$coefficients[4,1],summary(lm(fmla,
data=df))$coefficients[4,4])
    ))
  }
  #run the function we just specified to get coefficient and p-value
  #(or NAs, if there was a warning message):
  RTfixedCOEFandPVAL <- RTfixed(fmlaRT)
```

```
  #run the reaction time random effects analyses and get a list of the
cofficients and p-values:
  outputrts <- sapply(1:lrf, FUN=function(f){  #cycling through each random
effects structure
    print(randomEffects[f]) #tell the user which analysis we're on
    #create formulae using the strings of fixed effects and random effects we
created:
    fmlaRT <- as.formula(paste0(anteRT, randomEffects[f]))
    fmlaRT2 <- as.formula(paste0(anteRT2, randomEffects[f]))
    #create function to run analysis for this random effects structure:
    RTval <- function(fmla1, fmla2){
      return(tryCatch( #if we get a warning, it's probably because analysis
didn't converge.
        warning=function(w){
          #so in that case, print warning message:
          print("Regression (RT data) did not converge")
          #and return NAs for that analysis:
          return(c(NA,NA))
        },
        #but if no warning, return object:race interaction coefficient,
        #as well as p-value from comparison of
        #models with and without object:race interaction:
        expr = c(summary(lmer(fmla1, data=df))$coefficients[4,1],
anova((lmer(fmla1, data=df)), (lmer(fmla2, data=df)))[2,8])
      ))
    }
    #run the function we just specified to get coefficient and p-value (or
NA,
    #if there was a warning message):
    vals <- RTval(fmlaRT,fmlaRT2)
    outputrt <- c(outputrt, vals) #add that p-value to our vector of RT p-
values
    return(outputrt)
  }
  )

  print("error analyses") #tell the user we're doing error analyses now

  #run the error analysis with only fixed effects and get coefficient and p-
value
  print("fixed effects") #tell the user which analysis we're on
  #create function to run analysis for this random effects structure:
  fmlaER <- as.formula(anteER)
  ERfixed <- function(fmla){
    return(tryCatch( #if we get a warning, it's probably because analysis
didn't converge.
      warning=function(w){
        #so in that case, print warning message:
        print("Regression (error data) did not converge")
        #and return NAs for that analysis:
        return(c(NA,NA))
      },
      #but if no warning, return coefficient and p-value from object:race
interaction:
```

```
      expr = c(summary(glm(fmla, data = df,
family=binomial(link=logit)))$coefficients[4,1],summary(glm(fmla, data = df,
family=binomial(link=logit)))$coefficients[4,4])
    ))
  }
  #run the function we just specified to get p-value (or NA,
  #if there was a warning message):
  ERfixedCOEFandPVAL <- ERfixed(fmlaER)

  #run error random effects analyses and make a list of the coefficients and
p-values:
  outputers <- sapply(1:lrf, FUN=function(f){ #cycling through each random
effects structure
    print(randomEffects[f]) #tell the user which analysis we're on
    #create formula using the strings of fixed effects and random effects we
created:
    fmlaER <- as.formula(paste0(anteER, randomEffects[f]))
    #create function to run analysis for this random effects structure:
    reg3 <- function(fmla){
      return(tryCatch(  #if we get a warning, it's probably because analysis
didn't converge.
        warning=function(w){
          #so in that case, print warning message:
          print("Logistic regression (error data) did not converge")
          #and return NAs for that analysis:
          return(c(NA, NA))
        },
        #but if no warning, return object:race interaction coefficient and
        #p-value from logistic regression:
        expr = c(summary(glmer(fmla, data = df,

family=binomial(link=logit)))$coefficients[4,1],
                 summary(glmer(fmla, data = df,

family=binomial(link=logit)))$coefficients[4,4])
      ))
    }
    #run the function we just specified to get coefficient and p-value (or
NAs,
    #if there was a warning message):
    vals <- reg3(fmlaER)
    #add those values to our vector of error coefficients and p-values
    outputer <- c(outputer, vals)
    return(outputer)
  }
  )
  #error rate ANOVA:
  anovamod <- lm(error_rate~group*object_effects, data =
anovadf[which(anovadf$study==df$study[1]),])
  anovaoutput <- Anova(anovamod, type=3)  #use Type 3 sums of squares

  #make vector of coefficients and p-values from all the analyses for this
dataset:
  outputvec <- unlist(c(RTfixedCOEFandPVAL, outputrts, ERfixedCOEFandPVAL,
outputers, summary(anovamod)$coefficients[4,1], anovaoutput[4,4])) #make
vector of coefficients and p-values from all the analyses for this dataset
  #put vector in appropriate format to be added to output file:
```

```
  coeffs_pvals <- t(as.matrix(outputvec, nrow=2))
  return(coeffs_pvals)
}
)
#the above sapply() function returns a matrix:
# we want to convert that to a vector and assign it to
# the p-value column of the output dataframe:
output[,c("coefficient")] <- t(matrix(unlist(tempmat), nrow=2))[,1]
output[,c("pval")] <- t(matrix(unlist(tempmat), nrow=2))[,2]
#create column indicating if object:race interaction was significant in each
analysis:
output$sig <- ifelse(output$pval <= 0.05, T, F)
endTime <- proc.time()
print((endTime - startTime)/60)  #third value tells us how long this took us
to run

#took about 5 hours to run 19 datasets
#”output” dataframe should now contain the relevant output
```