Simulated data

for the study: The relationship between eating and concentration

Christoph Bamberg

```r
library(truncnorm)
```

Hunger

- Hunger = hungry: around -1, full: around +1
- unobserved variance already accounted for since drawing two normal distributions around timing / could also be interpreted as measurement error
- spread_H is used to shape the bimodal distribution - smaller values mean more distinct shape&less in the middle

```r
make_hunger<-function(N,spread_H=1,plotit=FALSE){
  #function to create a distribution of hunger values that roughly follow a bimodal distribution

  #Hunger<-c(rep(0,length.out=N/2),rep(1,length.out=N/2))

  #half hungry
  hungry<-rtruncnorm(n=N/2,a=-2,b=0.5,mean=-1,sd=spread_H)
  #half full
  full<-rtruncnorm(n=N/2,a=-0.5,b=2,mean=1,sd=spread_H)
  Hunger<-c(hungry,full)

  #### plotting:
  if (plotit==TRUE){
    hist(Hunger,breaks=10,main="Hunger values")

    plot(hungry,col="red",pch=21,main="Hunger",ylim=c(-2,2))
    points(full,col="blue",pch=20)
    legend(-1,2,legend=c("hungry","full"),col=c("red","blue"),pch=21:20)

    }

  return(Hunger)}

#Hi_example<-make_hunger(N,spread_H=1,plotit=TRUE)
#print(c("mean, min, max hunger",mean(Hi_example),min(Hi_example),max(Hi_example)))
```

Expectations

- put together from unobserved variance + belief in challenge's statement + interaction channelge * hunger
- Belief constructed much like hunger
- Belief that hunger is good around -1, belief that fullnes is good around +1

```
make_expectation<-function(N,Hi_in,spread_B=1,
                           interaction_strength=0.2, #need to add the hunger values for interaction
                           reduce_factor=200,
                           plotit=FALSE){

  # function to create expectation values.
  # requires results of make_hunger function
  # combines a bimodal with an interaction with hunger

  # for plots: better have N above 200, or change reduce_factor to smaller values


  Hi=Hi_in
  ## first the degree of belief in the expectation challenge
  #1/2 in the hunger-condition believe that hunger is good to some degree
  Belief_Hgood_Hcond<-rtruncnorm(n=N/4,a=-2,b=0.1,mean=1,sd=spread_B)
  #1/2 in the hunger-condition believe that fullness is good to some degree
  Belief_Fgood_Hcond<-rtruncnorm(n=N/4,a=-2,b=0.1,mean=-1,sd=spread_B)

  #1/2 in the full-condition believe that hunger is good to some degree
  Belief_Hgood_Fcond<-rtruncnorm(n=N/4,a=-0.1,b=2,mean=-1,sd=spread_B)
  #1/2 in the full-condition believe that fullness is good to some degree
  Belief_Fgood_Fcond<-rtruncnorm(n=N/4,a=-0.1,b=2,mean=1,sd=spread_B)

  Belief<-c(Belief_Hgood_Hcond,Belief_Fgood_Hcond,Belief_Hgood_Fcond,Belief_Fgood_Fcond)

  ## adding interaction
  #taking the absolute of hunger to multiply with, adding some random term so that interaction isn't so
  interHE<-  Belief * abs(Hi) + rnorm(N,mean=0, sd=spread_B)


  ## putting both together
  Ei<-Belief + interaction_strength * interHE



  ##### plots
  if (plotit==TRUE){
    ## plot 1: density
    plot(density(Belief),col="red",main="density Expectations")
    lines(density(Ei),col="blue")
    legend(0,0.1,legend=c("w/o interaction","with interaction"),col=c("red","blue"),lty=1)
    ## plot 2: scatter plot
    plot(Belief,col="red",pch=21,main="Expectations")
    points(Ei,col="blue",pch=20)
    legend(-1,2,legend=c("w/o interaction","with interaction"),col=c("red","blue"),pch=21:20)
    ##plot 2b scatter plot but shuffled
    B.shuff<-Belief[sample(1:length(Belief))]
    Ei.shuff<-Ei[sample(1:length(Ei))]
    plot(B.shuff,col="red",pch=21,main="Expectations but shuffled index")
    points(Ei.shuff,col="blue",pch=20)
    legend(-1,2,legend=c("w/o interaction","with interaction"),col=c("red","blue"),pch=21:20)


    ##pllot 3: more detailed scatter plot
    # only taking subsample
    reduceby=N/reduce_factor
    Belief.sub<-Belief[seq(1, length(Belief), reduceby)]
    Hi.sub<-Hi[seq(1, length(Hi), reduceby)]
    Ei.sub<-Ei[seq(1, length(Ei), reduceby)]
    #recalculating expectations for subsample:
    #Ei.sub<-Belief.sub + interaction_strength * (Belief.sub*abs(Hi.sub))
    #plotting w/o interaction
    plot(Belief.sub,col="red",pch=21,
         xlim=c(0,length(Belief.sub)),ylim=c(min(Ei.sub),max(Ei.sub)),main="sample of Expectations")
    #plotting w/ interaction
    points(Ei.sub,col="blue",pch=20)
    # adding lines between with and w/o interaction
    for ( i in 1:length(Belief.sub) ) lines( x=c(i,i),y=c(Ei.sub[i],Belief.sub[i]),col="grey",lwd=0.5 )
    legend(-1,max(Ei.sub)-2,legend=c("w/o interaction","with interaction"),col=c("red","blue"),pch=21:2

  return(Ei)}

#Ei_example<-make_expectation(N,Hi_in=Hi_example,interaction_strength=0,plotit=TRUE)

#Ei_example<-make_expectation(N,Hi_in=Hi_example,interaction_strength=0.2,plotit=TRUE)
#print(c("mean, min, max belief for example simulation",mean(Ei_example),min(Ei_example),max(Ei_example
```

cognitive performance

```
create_outcome<-function(N,
                         b_H,b_E, #coefficients for hnger and expectations
                         spread_H, #for hunger function
                         spread_B, #for expectation function
                         interaction_strength, #for expectations function
                         spread_perf=1, #for intercept of cognitiv performance
                         plotit=FALSE,
                         reduce_factor=200){

  # puts together the hunger & belief coefficients with an intercept
  # returns a dataframe with the values


  # making intercept
  a=rnorm(N,mean=0,sd=spread_perf)

  #making hunger
  Hi<-make_hunger(N,spread_H,plotit)

  #making expectations
  Ei<-make_expectation(N,Hi, spread_B, interaction_strength,reduce_factor, plotit)

  yi=a + b_H * Hi + b_E * Ei

  if (plotit==TRUE){
    plot(yi, main="Outcome")
    plot(density(yi),main="Density of Outcome values")}

  Data<-data.frame(yi=yi,Hi=Hi,Ei=Ei)

  Data$hunger_intervention<-c(rep(0,length.out=N/2),rep(1,length.out=N/2))
  Data$expectation_intervention<-c(rep(0,length.out=N/4),rep(1,length.out=N/4),rep(0,length.out=N/4),rep
  Data$condition<-c(rep(1,length.out=N/4),rep(2,length.out=N/4),rep(3,length.out=N/4),rep(4,length.out=N

  return(Data)}
```

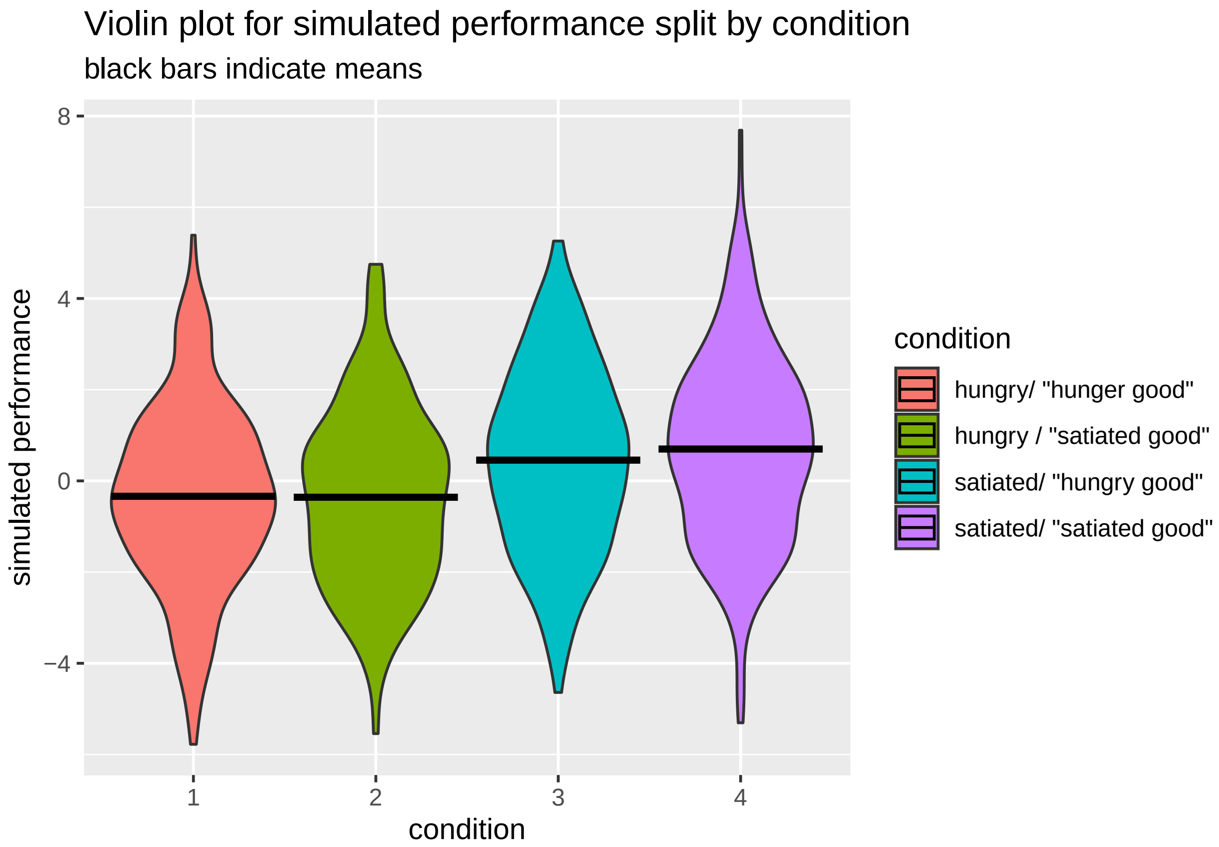
```
# example values
N=1000

b_H=0.3

b_E=0.2

b_HE=-0.15

#creating data:
DF<-create_outcome(N,b_H=b_H,b_E=b_E,spread_perf=2,spread_H=1,spread_B=2,interaction_strength = b_HE,plo
#View(DF)
```

coding of index variables

order of categories was preserved throughout creating the simulations 0 = hungry/ hunger is good, 1 = full / full is good for "condition": - 1 = hungry & hungry good - 2 = hungry & full good - 3 = full & hungry good - 4 = full & full good

more plots

```
library(ggplot2)
DF$Hunger<-as.numeric(DF$Hi)
DF$Expectation<-as.numeric(DF$Ei)
DF$perf<-as.numeric(DF$yi)
DF$condition<-as.factor(DF$condition)
ggplot(data=DF, aes(x=condition , y=yi, fill=condition))+
  geom_violin(scale="area")+
  stat_summary(fun="mean",
               geom="crossbar") +
  labs(x="condition",y="simulated performance",title="Violin plot for simulated performance split by co
```



saving DF to csv & saving specifications

```
write.csv(DF,"Eating_Expectations_simulated_data.csv")
specs<-data.frame(N=N,coeff_hunger=b_H,coeff_expectation=b_E,coeff_interaction=b_HE)
write.csv(specs,"Parameters_simulated_data.csv")
```

same but by calling the functions from remote

```
#source("simulation_eating_expectation_functions.R")
#N=1000
#b_H=0.3
#b_E=0.2
#b_HE=-0.15
```

```
#DF<-create_outcome(N,b_H=b_H,b_E=b_E,spread_perf=1.5,spread_H=0.8,spread_B=1.5,interaction_strength =

#write.csv(DF,"Eating_Expectations_simulated_data.csv")
#specs<-data.frame(N=N,coeff_hunger=b_H,coeff_expectation=b_E,coeff_interaction=b_HE)
#write.csv(specs,"Parameters_simulated_data.csv")
```