# Supplemental material for ‘Comparing theory-driven and data-driven attractiveness models using images of real women’s faces’

## *Deriving shape and color scores*

## Housekeeping

### Load libraries

```
library(tidyverse)
library(broom)
library(geomorph)
```

### Custom functions

```
# Select PCs according to Broken Stick Model (MacArthur 1957)
# Takes output from plotTangentSpace or prcomp as argument
# Returns selected PCs and saves number of selected PCs in variable called "n.PCs.[argument name]"

# Some of the following code has been adapted from function "evplot" (Francois Gillet, http://adn.biol.umontreal.
ca/~numericalecology/numecolR/)
selectPCs<-function(PCA.output){

  ev <- PCA.output$sdev^2
  n.ev <- length(ev)
  bsm <- data.frame(j=seq(1:n.ev), p=0)

  bsm$p[1] <- 1/n.ev
  for (i in 2:n.ev) bsm$p[i] <- bsm$p[i-1] + (1/(n.ev + 1 - i))
  bsm$p <- 100*bsm$p/n.ev

  test<-cbind(100*ev/sum(ev), bsm$p[n.ev:1])
  n.PCs<-sum(test[,1] >= test[,2])

  arg_name <- deparse(substitute(PCA.output)) # Get argument name
  var_name <- paste("n.PCs", arg_name, sep=".")
  assign(var_name, n.PCs, .GlobalEnv)

  if (!is.null(PCA.output$pc.summary$importance)) {

    return(PCA.output$pc.summary$importance[,1:n.PCs])

  } else {

    temp<-summary(PCA.output)
    return(temp$importance[,1:n.PCs])

  }

}
```

## Shape

### Data prep

#### Load data

Read shape data (n=594 women)

```
data<-readland.tps("dataFiles/sample_n594.tps", specID = "imageID",warnmsg = FALSE)
ids<-dimnames(data)[[3]]

# Defining dimensions of data
k<-dim(data)[1] # number of landmarks
d<-dim(data)[2] # number of dimensions
n<-dim(data)[3] # number of specimen
```

Read shape data of separate set of images for sexual dimorphism scores (n=50 men and 50 women)

```
data.sexdim<-readland.tps("dataFiles/sexdim.tps", specID = "imageID",warnmsg = FALSE)
ids.sexdim<-dimnames(data.sexdim)[[3]]

n.sd<-dim(data.sexdim)[3]
n.all <-n+n.sd
```

## Generalized Procrustes analysis

### GPA of original faces

```
gpa.data<-gpagen(data,print.progress=FALSE)

# geomorph flips templates: rotate
rotate<-gpa.data$coords
temp<-array(0,c(k,d,n))
data.aligned<-array(0,c(k,d,n))

for (i in 1:n){
  temp[,,i]<-rotate[,,i] %*% matrix(c(-1,0,0,1),2,2,byrow=T)
  data.aligned[,,i] <- temp[,c(2,1),i]
}

dimnames(data.aligned)[[3]]<-ids
```

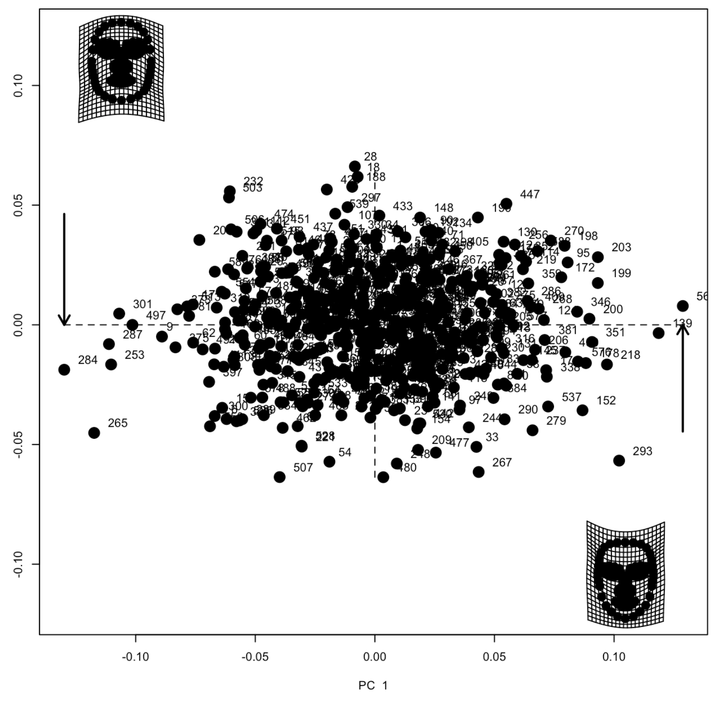
### GPA of faces incl. sexual dimorphism prototype faces

```
data.all<-array(c(data,data.sexdim), dim = c(k, d, n.all), dimnames=list(NULL,NULL,c(ids,ids.sexdim)))

gpa.sexdim<-gpagen(data.all,print.progress = FALSE)

# rotate data
rotate<-gpa.sexdim$coords
temp<-array(0,c(k,d,n.all))
sexdim.aligned<-array(0,c(k,d,n.all))

for (i in 1:n.all){
  temp[,,i]<-rotate[,,i] %*% matrix(c(-1,0,0,1),2,2,byrow=T)
  sexdim.aligned[,,i] <- temp[,c(2,1),i]
}

dimnames(sexdim.aligned)[[3]]<-c(ids,ids.sexdim)
```

## Principal component analysis

### Original data

```
PCA <- plotTangentSpace(data.aligned, verbose = T,label=T)
```



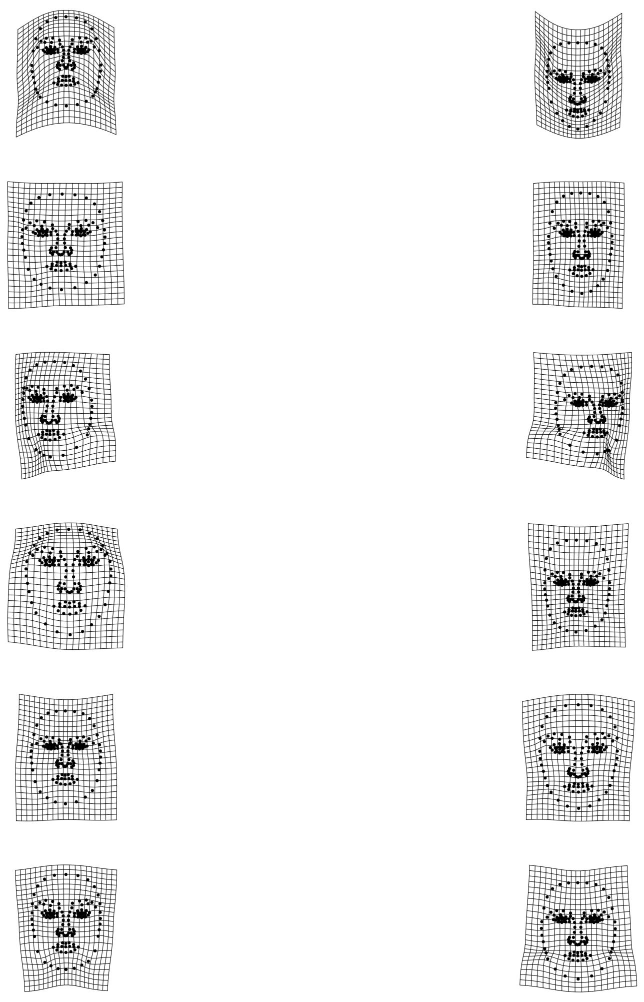
```
selectPCs(PCA)
```

```
##                                PC1        PC2        PC3        PC4
## Standard deviation     0.03866493 0.02247149 0.02136332 0.01806544
## Proportion of Variance 0.32617000 0.11017000 0.09957000 0.07120000
## Cumulative Proportion  0.32617000 0.43634000 0.53591000 0.60712000
##                                PC5       PC6        PC7        PC8
## Standard deviation     0.01475336 0.0131439 0.01218302 0.01072943
## Proportion of Variance 0.04749000 0.0376900 0.03238000 0.02512000
## Cumulative Proportion  0.65461000 0.6923000 0.72468000 0.74980000
##                                 PC9        PC10        PC11        PC12
## Standard deviation     0.009600971 0.009136584 0.008805167 0.007643237
## Proportion of Variance 0.020110000 0.018210000 0.016920000 0.012750000
## Cumulative Proportion  0.769910000 0.788120000 0.805040000 0.817780000
```

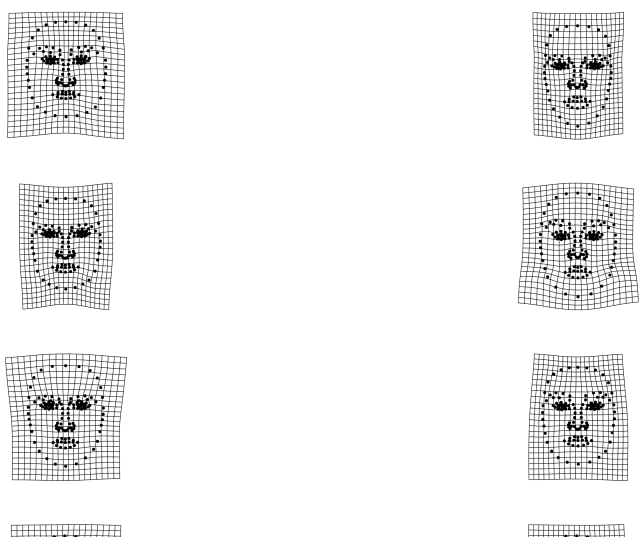
Hence retaining 12 principal components, explaining 81.8% of variance

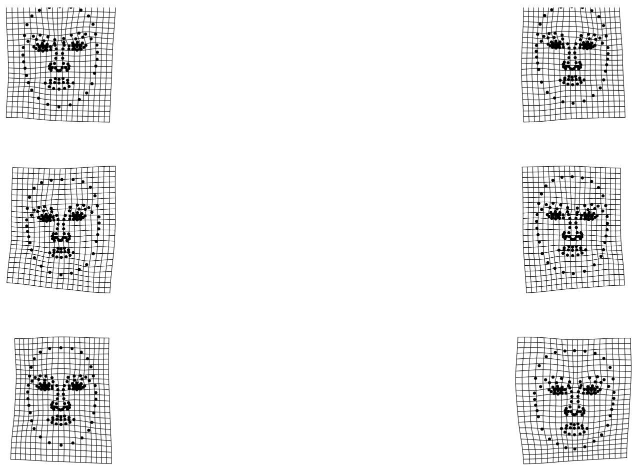
## Visualize PCs

PCs 1-6



PCs 7-12





## Save PC scores for subsequent analysis

```
scores<-PCA$pc.scores

col_names<-dimnames(scores)[[1]]
col_names <- gsub("_nc_1","", col_names)
row_names<-dimnames(scores)[[2]]
PC.scores<-matrix(scores,ncol=n,byrow = TRUE, dimnames=list(row_names,col_names)) %>%
  t() %>%
  as.data.frame() %>%
  rownames_to_column(var="oc_id") %>%
  dplyr::select(1:(n.PCs.PCA+1))
```

# Symmetry

Create empty array for next step

```
original<-data.aligned
mirrored<-array(0,c(k,d,n))
```

Mirror landmark template by changing sign of x-coords

```
for (i in 1:n){
  mirrored [,,i]<- original[,,i] %*% matrix(c(-1,0,0,1),2,2,byrow=T)}
```

Relabel landmarks

```
mirr.lm.order<-as.matrix(read.table("./dataFiles/mirr.lm.order.txt",header=F))
mirrored<-mirrored[order(mirr.lm.order),,]
```

Put original & mirrored data in one array

```
org.mirr<-array(0,c(k,d,n*2))
org.mirr[,,1:n]<-original[,,1:n]
org.mirr[,,(n+1):(n*2)]<-mirrored[,,1:n]
```

Run GPA

```
gpa.asymmetry<-gpagen(org.mirr,print.progress = FALSE)
```

Separate into two arrays

```
data.org.aligned<-gpa.asymmetry$coords[,,1:n]
data.mirr.aligned<-gpa.asymmetry$coords[,,(n+1):(n*2)]
```

Calculate Procrustes distance between original and mirrored faces

```
asym<-numeric(n)
for (i in 1:n) {
  asym[i]<-sqrt(sum((data.org.aligned[,,i]-data.mirr.aligned[,,i])**2))
}
```

# Averageness

Calculated as Euclidean distance between PCs of individual faces and average face

```
pc.matrix <- PC.scores[,2:ncol(PC.scores)] %>% as.matrix()
sample.mean<-t(as.matrix(colMeans(pc.matrix)))
avg<-numeric(n)
for (i in 1:n) {
  avg[i]<-sqrt(sum((pc.matrix[i, ]-sample.mean)**2))
}
```

# Sexual dimorphism

## Data prep

Convert array to matrix for PCA using prcomp

```
# Data needs to be in format id, x1, y1, x2, y2, ...x132, y132
temp<-matrix(sexdim.aligned,ncol=(k*2),byrow=T)
x.coords<-temp[,1:k]
y.coords<-temp[,(k+1):(k*2)]
sexdim.matrix<-matrix(rbind(x.coords,y.coords),n.all, dimnames=list(dimnames(sexdim.aligned)[[3]],NULL))

sexdim.matrix.prototypes<-sexdim.matrix[(n+1):n.all,]
sexdim.matrix.faces<-sexdim.matrix[1:n,]
```

PCA only of those faces that will constitute sexual dimorphism prototypes

```
PCA.sexdim <- prcomp(sexdim.matrix.prototypes)
selectPCs(PCA.sexdim)
```

```
##                               PC1        PC2        PC3        PC4
## Standard deviation     0.02994476 0.02447292 0.02051732 0.01730292
## Proportion of Variance 0.21420000 0.14307000 0.10056000 0.07152000
## Cumulative Proportion  0.21420000 0.35727000 0.45783000 0.52935000
##                               PC5        PC6        PC7       PC8
## Standard deviation     0.01575885 0.01402555 0.01118872 0.0108423
## Proportion of Variance 0.05932000 0.04699000 0.02990000 0.0280800
## Cumulative Proportion  0.58867000 0.63567000 0.66557000 0.6936500
##                               PC9        PC10
## Standard deviation     0.01019717 0.009992825
## Proportion of Variance 0.02484000 0.023850000
## Cumulative Proportion  0.71849000 0.742340000
```

10 PCs explaining 74.2% of variance

Saves scores of faces that will constitute sexual dimorphism prototypes

```
PC.scores.prototypes<-PCA.sexdim$x[,1:n.PCs.PCA.sexdim] %>%
  as.data.frame() %>%
  rownames_to_column() %>%
  separate(rowname, c("sex"), sep = "_", extra = "drop", remove=FALSE) %>%
  mutate(sex = recode(sex, f = 0, m = 1)) %>%
  column_to_rownames("rowname")
```

Predict OCMATE PC scores based on sexdim PCA model

```
PC.scores.data <- predict(PCA.sexdim, newdata=sexdim.matrix.faces) %>%
  as.data.frame() %>%
  dplyr::select(1:n.PCs.PCA.sexdim)
```

## Calculating vector score

Pull out faces that define female and male prototypes, respectively

```
fem <- PC.scores.prototypes %>%
  filter(sex==0) %>%
  dplyr::select(-sex) %>%
  as.matrix()

mal <- PC.scores.prototypes %>%
  filter(sex==1) %>%
  dplyr::select(-sex) %>%
  as.matrix()
```

Calulate male and female average for each PC (=column)

```
fem.mean <- t(as.matrix(colMeans(fem)))
mal.mean <- t(as.matrix(colMeans(mal)))
```

Normalize difference vector between female and male

```
temp <- mal.mean - fem.mean
norm.dist <- sqrt(sum(temp ^ 2))
norm.vec <- temp / norm.dist
```

For each face, subtract average fem PC score, divide by norm.distance, multiply result by norm.vector, sum elements

```
R <- nrow(PC.scores.data)
pc.matrix <- as.matrix(PC.scores.data)
sd.vector <- numeric(R)
for (i in 1:R) {
  z1 <- pc.matrix[i, ] - fem.mean
  z2 <- z1 / norm.dist
  z3 <- z2 * norm.vec
  sd.vector[i] <- sum(z3)
}
```

# Save shape scores

```
shapeScores<-data.frame(
  "oc_id"=PC.scores[,1],
  asym,
  avg,
  sd.vector,
  PC.scores[,2:ncol(PC.scores)]
)

write.csv(shapeScores, file ="./dataFiles/0_originalFiles/shapeScores.csv",row.names=FALSE)

shapeScores %>%
  mutate_at(vars(PC1:PC12),funs(as.numeric(scale(.)))) %>%
  write.csv(file ="./dataFiles/0_originalFiles/shapeScores_z.csv",row.names=FALSE)
```

# Colour

## Deriving colour PCs

– separate script –

### Load data

```
load('./_preppingData_Colour/OCMATE_set/colourPCs_OCMATE.Rdata')

n.PCs.col <- 60
n.col <- NROW(colourPCs)

col_names <- dimnames(colourPCs)[[1]]
col_names <- gsub("_nc_1_calibrated.png","", col_names)
row_names <- dimnames(colourPCs)[[2]]
PC.scores.col <- matrix(colourPCs,ncol=n.col,byrow = FALSE, dimnames=list(col_names,row_names)) %>%
  as.data.frame() %>%
  rownames_to_column(var="oc_id") %>%
  dplyr::select(1:(n.PCs.col+1))

names(PC.scores.col) <- gsub("PC", "PC.col", names(PC.scores.col))
```

## Averageness

Calculated as distance between PCs of individual faces and average face

```
pc.matrix <- PC.scores.col[,2:ncol(PC.scores.col)] %>% as.matrix()
sample.mean<-t(as.matrix(colMeans(pc.matrix)))
avg.col<-numeric(n.col)
for (i in 1:n.col) {
  avg.col[i]<-sqrt(sum((pc.matrix[i, ]-sample.mean)**2))
}
```

## Sexual dimorphism

### Data prep

Load sexual dimorphism PCA model (needed to predict PC scores of OCMATE faces). Note that colour-calibrated images were only available for 36 of the 50 men; the sexual dimorphism colour PC model was thus built on a subset of 36 men and 36 women.

```
# ON MAC
load('./_preppingData_Colour/CP2_set/colourPCA_CP.Rdata') # name of prcomp output: "colour.pc.CP"
PCA.sexdim.col<-colour.pc.CP # rename to keep naming consistent
selectPCs(PCA.sexdim.col)
```

```
##                              PC1      PC2      PC3      PC4      PC5
## Standard deviation     47.97244 25.06412 18.42228 16.51244 15.30431
## Proportion of Variance  0.33069  0.09027  0.04877  0.03918  0.03366
## Cumulative Proportion   0.33069  0.42096  0.46973  0.50891  0.54257
##                              PC6      PC7      PC8      PC9     PC10
## Standard deviation     13.76112 12.48656 11.67339 11.14693 10.75952
## Proportion of Variance  0.02721  0.02240  0.01958  0.01785  0.01664
## Cumulative Proportion   0.56978  0.59218  0.61176  0.62962  0.64625
##                             PC11     PC12     PC13     PC14     PC15
## Standard deviation     10.27568 9.928083 9.613953 9.399682 8.957973
## Proportion of Variance  0.01517 0.014160 0.013280 0.012700 0.011530
## Cumulative Proportion   0.66143 0.675590 0.688870 0.701570 0.713100
##                             PC16
## Standard deviation     8.797169
## Proportion of Variance 0.011120
## Cumulative Proportion  0.724220
```

Hence retaining 16 PCs explaining 72% of variance

Saves scores of faces that will constitute sexual dimorphism prototypes

```
PC.scores.col.SDprototypes<-PCA.sexdim.col$x[,1:n.PCs.PCA.sexdim.col] %>%
  as.data.frame() %>%
  rownames_to_column() %>%
  rename(id=rowname) %>%
  left_join(read.table("./_preppingData_Colour/CP2_set/CP_sex.txt", sep="\t", header=TRUE, stringsAsFactors=FALSE
), by="id") %>%
  column_to_rownames(var="id")
n.SD.col<-NROW(PC.scores.col.SDprototypes)
```

Load OCMATE colour values (“colour”)

```
load('./_preppingData_Colour/OCMATE_set/colourValues_OCMATE.Rdata')
```

Predict PC scores based on sexdim PCA model

```
PC.scores.col.SDdata <- predict(PCA.sexdim.col, newdata=colour)

# in case of memory problems, below a loop that analyses in batches of 40 rows
PC.scores.col.SDdata<-matrix(NA,nrow=594,ncol=76) # create empty matrix
x<-(seq(nrow(colour))-1) %/% 40                   # create subset indices
i<-0                                              # create subset counter
j<-0                                              # create line index counter

while (i<=max(x)) {

  PC.scores.col.SDdata[(j+1):(j+sum(x==i)),]<-predict(PCA.sexdim.col, newdata=colour[(j+1):(j+sum(x==i)),])

  j=j+sum(x==i)
  i=i+1
}

#save(PC.scores.col.SDdata,file="./_preppingData_Colour/CP2_set/PC.scores.col.SDdata.Rdata")
#load("PC.scores.col.SDdata.Rdata")
```

```
PC.scores.col.SDdata <- PC.scores.col.SDdata %>%
  as.data.frame() %>%
  dplyr::select(1:n.PCs.PCA.sexdim.col)

#write.csv(PC.scores.col.SDdata,"PC_scores_col_SDdata.csv")
```

```
PC.scores.col.SDdata <-read.table("./_preppingData_Colour/CP2_set/PC_scores_col_SDdata.csv", sep=",", header=TRUE
, stringsAsFactors=FALSE) %>%
  column_to_rownames("X")
```

## Calculating vector score

Pull out faces that define female and male prototypes, respectively

```
fem <- PC.scores.col.SDprototypes %>%
  filter(sex==0) %>%
  dplyr::select(-sex) %>%
  as.matrix()
mal <- PC.scores.col.SDprototypes %>%
  filter(sex==1) %>%
  dplyr::select(-sex) %>%
  as.matrix()
```

Calulate male and female average for each PC (=column)

```
fem.mean <- t(as.matrix(colMeans(fem)))
mal.mean <- t(as.matrix(colMeans(mal)))
```

Normalize difference vector between female and male

```
temp <- mal.mean - fem.mean
norm.dist <- sqrt(sum(temp ^ 2))
norm.vec <- temp / norm.dist
```

For each face, subtract average fem PC score, divide by norm.distance, multiply result by norm.vector, sum elements

```
R <- nrow(PC.scores.col.SDdata)
pc.matrix <- PC.scores.col.SDdata
sd.vector.col <- numeric(R)
for (i in 1:R) {
  z1 <- pc.matrix[i, ] - fem.mean
  z2 <- z1 / norm.dist
  z3 <- z2 * norm.vec
  sd.vector.col[i] <- sum(z3)
}
```

## Save colour scores

```
colourScores<-data.frame(
  "oc_id"=PC.scores.col[,1],
  avg.col,
  sd.vector.col,
  PC.scores.col[,2:ncol(PC.scores.col)]
)

write.csv(colourScores, file = "./dataFiles/0_originalFiles/colourScores.csv",row.names=FALSE)

colourScores %>%
  mutate_at(vars(PC.col1:PC.col60),funs(as.numeric(scale(.)))) %>%
  write.csv(file = "./dataFiles/0_originalFiles/colourScores_z.csv",row.names=FALSE)
```

# Supplemental material for ‘Comparing theory-driven and data-driven attractiveness models using images of real women’s faces’

## Modeling

# Housekeeping

## Load libraries

```
library(tidyverse)
library(broom)
library(psych)
library(ICC)
library(caret) # cross-validation
library(viridis) # colour scheme
```

## Raincloud plot theme

Adapted from https://micahallen.org/2018/03/15/introducing-raincloud-plots/ (https://micahallen.org/2018/03/15/introducing-raincloud-plots/)

```
source("https://gist.githubusercontent.com/benmarwick/2a1bb0133ff568cbe28d/raw/fb53bd97121f7f9ce947837ef1a4c65a73
bffb3f/geom_flat_violin.R")

raincloud_theme = theme(
  text = element_text(size = 10),
  axis.text = element_text(size = 10),
  axis.title.x = element_text(size = 12),
  axis.title.y = element_text(size = 12),
  axis.text.x = element_text(angle = 45, vjust = 0.5),
  legend.title = element_text(size = 16),
  legend.text = element_text(size = 16),
  legend.position = "none",
  panel.border = element_blank(),
  panel.grid.minor = element_blank(),
  panel.grid.major = element_blank(),
  axis.line.x = element_line(colour = 'black', size=0.5, linetype='solid'),
  axis.line.y = element_line(colour = 'black', size=0.5, linetype='solid')
)
```

```
options(width = 800)
```

# Predicting female attractiveness with shape and colour

## Prepping data for analyses

### Load data

#### Shape and colour scores

```
shapeScores <- read.csv("dataFiles/shapeScores_z_anon.csv")
colourScores <- read.csv("dataFiles/colourScores_z_anon.csv")
```

#### Sparseness

Images for sparseness analyses were sized 300px x 400px. Note that alpha8, alpha12 and alpha16 refer to receptive field sizes of 8 x 8, 12 x 12 and 16 x 16 respectively

```
sparseness <- read.csv("dataFiles/sparsenessScores_anon.csv")
```

#### Age and anthrompetric data

```
bodymeasures <- read.csv("dataFiles/bodymeasures_anon.csv")
```

#### Attractiveness ratings

```
# For ICC
ratings <- read.csv("dataFiles/ratings_anon.csv") %>%
  rename(oc_id = trial,
         rater_id = user_id)

# For modelling
attr <- read.csv("dataFiles/ratings_anon.csv") %>%
  rename(oc_id = trial,
         rater_id = user_id) %>%
  group_by(rater_id) %>%
  mutate(dv = scale(dv)) %>%
  ungroup() %>%
  group_by(oc_id) %>%
  summarise(att = mean(dv)) %>%
  ungroup() %>%
  left_join(shapeScores, by = "oc_id") %>%
  left_join(colourScores, by = "oc_id") %>%
  left_join(sparseness, by = "oc_id") %>%
  left_join(bodymeasures, by = "oc_id") %>%
  select(oc_id, age, height, weight, bmi, 2:ncol(.))
```

## Descriptive stats

### Face images

```
attr %>%
  dplyr::select(age:bmi) %>%
  summarise_all(funs(
    M = mean,
    SD = sd,
    missing = sum(is.na(.))),
    na.rm = TRUE
  ) %>%
  gather(stat, val) %>%
  mutate(val = round(val, 1)) %>%
  separate(stat, into = c("Measure", "stat"), sep = "_") %>%
  spread(stat, val) %>%
  dplyr::select(Measure, M, SD, missing) %>%
  as.data.frame()
```

```
## Warning: funs() is soft deprecated as of dplyr 0.8.0
## please use list() instead
##
## # Before:
## funs(name = f(.)
##
## # After:
## list(name = ~f(.))
## This warning is displayed once per session.
```

```
##   Measure     M   SD missing
## 1     age  21.5  3.2       1
## 2     bmi  23.1  3.8      49
## 3  height 165.8  6.3      48
## 4  weight  63.8 11.9      48
```

### Facial attractiveness ratings

Cronbach's alpha

```
ratings.wide <- ratings %>%
  select(rater_id, oc_id, dv) %>%
  spread(rater_id, dv) %>%
  select(-oc_id)

ca <- ratings.wide %>%
  psych::alpha()

# Cronbach's alpha
cron.alpha<-ca$total$raw_alpha %>% round(2)

# 95% confidence interval (Feldt, Woodruff & Salih, 1987)
df_1 <- dim(ratings.wide)[1]-1
df_2 <- (dim(ratings.wide)[1]-1)*(dim(ratings.wide)[2]-1)
lci <- 1 - abs((1 - ca$total$raw_alpha) * qf((1 - .05 / 2), df1 = df_1, df2 = df_2)) %>% round(2)
uci <- 1 - abs((1 - ca$total$raw_alpha) * qf((.05 / 2), df1 = df_1, df2 = df_2)) %>% round(2)

tibble(stat = c("Cronbach's alpha","95% CI (lower)","95% CI (upper)"),
       val=c(cron.alpha, lci, uci)) %>%
  as.data.frame()
```

```
##                stat  val
## 1 Cronbach's alpha 0.93
## 2   95% CI (lower) 0.92
## 3   95% CI (upper) 0.94
```

Intraclass correlation coefficients of ratings

```
ICC.all <- ICCest(rater_id, dv, data = ratings)

ratings.m <- ratings %>%
  filter(sex == "male") %>%
  arrange(oc_id, rater_id)

ratings.f <- ratings %>%
  filter(sex == "female") %>%
  arrange(oc_id, rater_id)

ICC.fem <- ICCest(rater_id, dv, data = ratings.f)
ICC.mal <- ICCest(rater_id, dv, data = ratings.m)

ICC.results <- list("All" = ICC.all,
                    "Female" = ICC.fem,
                    "Male" = ICC.mal)
bind_rows(ICC.results, .id="Raters") %>%
  as.data.frame()
```

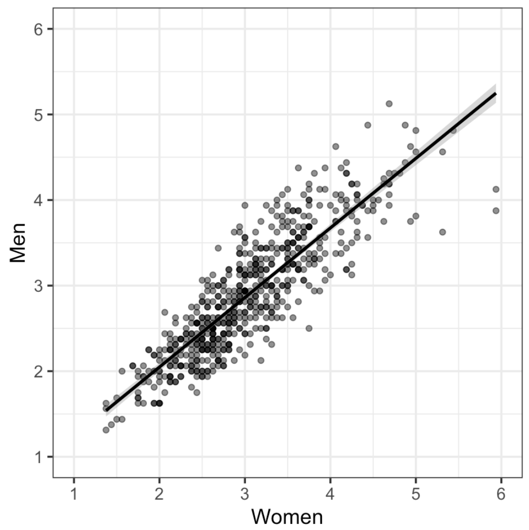
```
##   Raters       ICC   LowerCI   UpperCI  N   k     varw      vara
## 1    All 0.2982422 0.2140927 0.4294523 32 594 1.566607 0.6657972
## 2 Female 0.3213827 0.2047494 0.5320878 16 594 1.547483 0.7328643
## 3   Male 0.2848266 0.1778982 0.4889266 16 594 1.585731 0.6315369
```

Correlation between female and male attractiveness ratings

```
ratings.bysex <- ratings %>%
  group_by(oc_id, sex) %>%
  summarise(att = mean(dv)) %>%
  ungroup %>%
  spread(sex, att) %>%
  as.data.frame()

cor.test(ratings.bysex$female, ratings.bysex$male)
```

```
##
##  Pearson's product-moment correlation
##
## data:  ratings.bysex$female and ratings.bysex$male
## t = 42.323, df = 592, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.8454671 0.8856290
## sample estimates:
##       cor
## 0.8669486
```

```
ratings.bysex %>%
  rename('Women' = 'female',
         'Men' = 'male') %>%
  ggplot(aes(x = Women, y = Men)) +
  geom_point(alpha = .5) +
  geom_smooth(method = "lm", col = "black") +
  scale_x_continuous(limits =  c(1, 6), breaks = seq(1, 6, 1)) +
  scale_y_continuous(limits =  c(1, 6), breaks = seq(1, 6, 1)) +
  coord_fixed(ratio = 1) +
  theme_bw(base_size = 14)
```



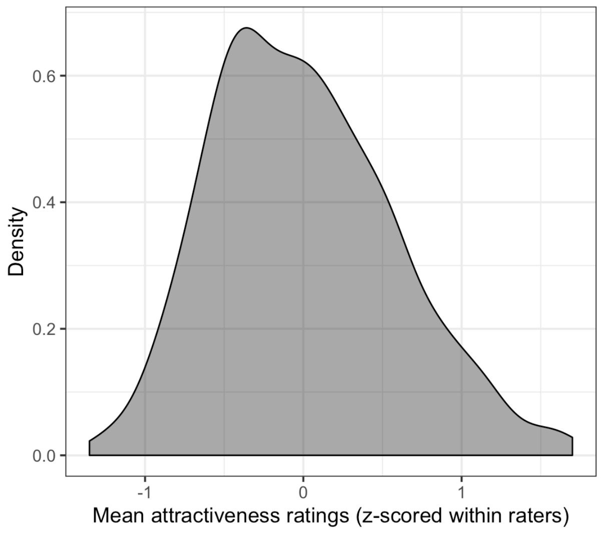
Mean attractiveness ratings

```
attr %>%
  summarise(mean_attractiveness = round(mean(att), 2),
            sd_attractiveness = round(sd(att), 2)) %>%
  as.data.frame()
```

```
##   mean_attractiveness sd_attractiveness
## 1                   0              0.57
```

Distribution of attractiveness ratings

```
attr %>%
  ggplot(aes(x =  att)) +
  geom_density(fill = "black", alpha = .4, trim = FALSE) +
  xlab("Mean attractiveness ratings (z-scored within raters)") +
  ylab("Density") +
  coord_fixed(ratio = 4) +
  theme_bw(base_size = 14)
```



## Shape and colour scores

### Shape

```
shapeScores %>%
  dplyr::select(asym:sd.vector) %>%
  summarise_all(funs(
    M = mean,
    SD = sd,
    min = min,
    max = max)) %>%
  gather(stat, val) %>%
  mutate(val = round(val, 4)) %>%
  separate(stat, into = c("Score", "stat"), sep = "_") %>%
  spread(stat, val) %>%
  dplyr::select(Score,M, SD, min, max)
```

```
##       Score      M     SD     min    max
## 1      asym 0.0561 0.0217  0.0210 0.2021
## 2       avg 0.0582 0.0188  0.0198 0.1371
## 3 sd.vector 0.2023 0.4224 -1.3164 1.4811
```

Note that for both sexual dimorphism scores 0 = female and 1 = male.

### Colour

```
colourScores %>%
  dplyr::select(avg.col:sd.vector.col) %>%
  summarise_all(funs(
    M = mean,
    SD = sd,
    min = min,
    max = max)) %>%
  gather(stat, val) %>%
  mutate(val = round(val, 4)) %>%
  separate(stat, into = c("Score", "stat"), sep = "_") %>%
  spread(stat, val) %>%
  dplyr::select(Score, M, SD, min, max)
```

```
##           Score       M      SD     min      max
## 1       avg.col 67.5932 30.0631 36.8769 324.9975
## 2 sd.vector.col  0.1673  0.6277 -1.7357   3.8629
```

Note that for both sexual dimorphism scores 0 = female and 1 = male.

## Intercorrelations of shape scores

```
corr.test(shapeScores[,2:4], adjust = "none")
```

```
## Call:corr.test(x = shapeScores[, 2:4], adjust = "none")
## Correlation matrix
##           asym  avg sd.vector
## asym      1.00 0.33      0.03
## avg       0.33 1.00      0.06
## sd.vector 0.03 0.06      1.00
## Sample Size
## [1] 594
## Probability values (Entries above the diagonal are adjusted for multiple tests.)
##           asym  avg sd.vector
## asym      0.00 0.00      0.48
## avg       0.00 0.00      0.14
## sd.vector 0.48 0.14      0.00
##
##  To see confidence intervals of the correlations, print with the short=FALSE option
```

### Intercorrelations of colour scores

```
corr.test(colourScores[,2:3], adjust = "none")
```

```
## Call:corr.test(x = colourScores[, 2:3], adjust = "none")
## Correlation matrix
##               avg.col sd.vector.col
## avg.col          1.00          0.31
## sd.vector.col    0.31          1.00
## Sample Size
## [1] 594
## Probability values (Entries above the diagonal are adjusted for multiple tests.)
##               avg.col sd.vector.col
## avg.col             0             0
## sd.vector.col       0             0
##
##  To see confidence intervals of the correlations, print with the short=FALSE option
```

## Modelling female attractiveness

### Theory-driven models

Note that sparseness models uses values for receptive field size of 16x16 (“alpha16”).

```
asym.form <- att ~ asym
asym.model <- lm(asym.form, data = attr)

avg.form <- att ~ avg + avg.col
avg.model <- lm(avg.form, data = attr)

sexdim.form <- att ~ poly(sd.vector, 2) + poly(sd.vector.col, 2)
sexdim.model <- lm(sexdim.form, data = attr)

bmi.form <- att ~ poly(bmi, 2)
bmi.model<- lm(bmi.form, data = filter(attr, !is.na(bmi)))

sparse.form <- att ~ poly(alpha16, 2)
sparse.model <- lm(sparse.form, data = attr)

comb.form <- att ~ avg + avg.col + poly(sd.vector, 2) + poly(sd.vector.col, 2) + asym + poly(bmi, 2) + poly(alpha
16, 2)
comb.model <- lm(comb.form, data = filter(attr, !is.na(bmi)))
```

### Face space model

```
facespace.form <- att ~ poly(PC1,2)+poly(PC2,2)+poly(PC3,2)+poly(PC4,2)+poly(PC5,2)+poly(PC6,2)+poly(PC7,2)+poly(
PC8,2)+poly(PC9,2)+poly(PC10,2)+poly(PC11,2)+poly(PC12,2)+
poly(PC.col1,2)+poly(PC.col2,2)+poly(PC.col3,2)+poly(PC.col4,2)+poly(PC.col5,2)+poly(PC.col6,2)+poly(PC.col7,2)+p
oly(PC.col8,2)+poly(PC.col9,2)+poly(PC.col10,2)+poly(PC.col11,2)+poly(PC.col12,2)+poly(PC.col13,2)+poly(PC.col14,
2)+poly(PC.col15,2)+poly(PC.col16,2)+poly(PC.col17,2)+poly(PC.col18,2)+poly(PC.col19,2)+poly(PC.col20,2)+poly(PC.
col21,2)+poly(PC.col22,2)+poly(PC.col23,2)+poly(PC.col24,2)+poly(PC.col25,2)+poly(PC.col26,2)+poly(PC.col27,2)+po
ly(PC.col28,2)+poly(PC.col29,2)+poly(PC.col30,2)+poly(PC.col31,2)+poly(PC.col32,2)+poly(PC.col33,2)+poly(PC.col34
,2)+poly(PC.col35,2)+poly(PC.col36,2)+poly(PC.col37,2)+poly(PC.col38,2)+poly(PC.col39,2)+poly(PC.col40,2)+poly(PC
.col41,2)+poly(PC.col42,2)+poly(PC.col43,2)+poly(PC.col44,2)+poly(PC.col45,2)+poly(PC.col46,2)+poly(PC.col47,2)+p
oly(PC.col48,2)+poly(PC.col49,2)+poly(PC.col50,2)+poly(PC.col51,2)+poly(PC.col52,2)+poly(PC.col53,2)+poly(PC.col5
4,2)+poly(PC.col55,2)+poly(PC.col56,2)+poly(PC.col57,2)+poly(PC.col58,2)+poly(PC.col59,2)+poly(PC.col60,2)
facespace.model <- lm(facespace.form, data = attr)
```

## Model comparison

### Cross-validation

```
set.seed(1994)
fitControl <- trainControl(method = "repeatedcv",
                           number = 10, # 10 folds
                           repeats = 10, # repeat ten times
                           savePredictions = TRUE,
                           returnResamp = 'all') # saves data on all folds/repetitions
```

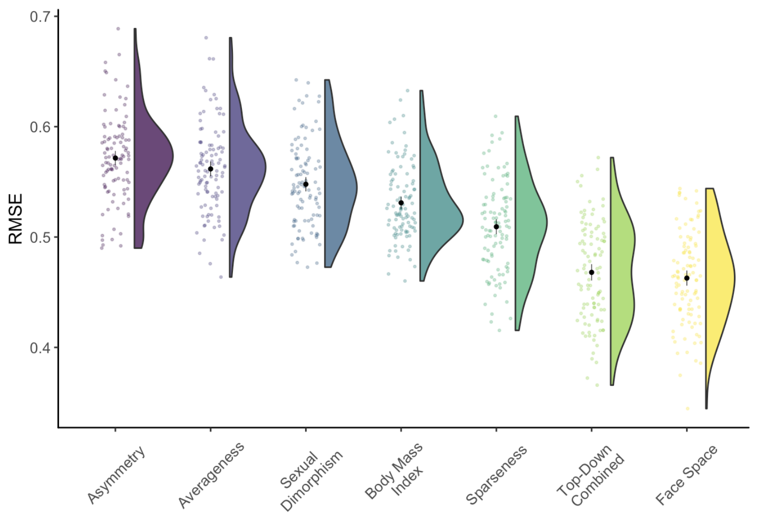
```
set.seed(1994)
asym.cv <- train(asym.form,
                 data = attr,
                 trControl = fitControl,
                 method = "lm")
asym.rsqs <- asym.cv$resample$Rsquared
asym.rmse <- asym.cv$resample$RMSE

set.seed(1994)
avg.cv <- train(avg.form,
                data = attr,
                trControl = fitControl,
                method = "lm")
avg.rsqs <- avg.cv$resample$Rsquared
avg.rmse <- avg.cv$resample$RMSE

set.seed(1994)
sexdim.cv <- train(sexdim.form,
                   data = attr,
                   trControl = fitControl,
                   method = "lm")
sexdim.rsqs <- sexdim.cv$resample$Rsquared
sexdim.rmse <- sexdim.cv$resample$RMSE

set.seed(1994)
bmi.cv <- train(bmi.form,
                data = filter(attr, !is.na(bmi)),
                trControl = fitControl,
                method = "lm")
bmi.rsqs <- bmi.cv$resample$Rsquared
bmi.rmse <- bmi.cv$resample$RMSE

set.seed(1994)
sparse.cv <- train(sparse.form,
                  data = attr,
                  trControl = fitControl,
                  method="lm")
sparse.rsqs <- sparse.cv$resample$Rsquared
sparse.rmse <- sparse.cv$resample$RMSE

set.seed(1994)
comb.cv <- train(comb.form,
                filter(attr, !is.na(bmi)),
                trControl = fitControl,
                method = "lm")
comb.rsqs <- comb.cv$resample$Rsquared
comb.rmse <- comb.cv$resample$RMSE

set.seed(1994)
facespace.cv <- train(facespace.form,
                      data = attr,
                      trControl = fitControl,
                      method = "lm")
facespace.rsqs <- facespace.cv$resample$Rsquared
facespace.rmse <- facespace.cv$resample$RMSE
```

Cross-validation results for each of the six models

```
##              Model intercept      RMSE   Rsquared       MAE     RMSESD RsquaredSD      MAESD
## 1        Asymmetry      TRUE 0.5715637 0.01692690 0.4642664 0.03921507 0.02169106 0.02792397
## 2      Averageness      TRUE 0.5617168 0.05079618 0.4584272 0.04031696 0.04818001 0.03013927
## 3 Sexual dimorphism      TRUE 0.5478403 0.10516446 0.4471678 0.03932173 0.07806421 0.03153971
## 4   Body Mass Index      TRUE 0.5310305 0.13574415 0.4317266 0.03454222 0.06343120 0.02633168
## 5       Sparseness      TRUE 0.5093108 0.22088473 0.4082392 0.04140661 0.09780380 0.03323123
## 6 Top-down combined      TRUE 0.4680041 0.33550246 0.3721261 0.04502389 0.10833511 0.03302573
## 7       Face space      TRUE 0.4627985 0.40213696 0.3627530 0.04187272 0.09269108 0.03178552
```

## Plotting CV results

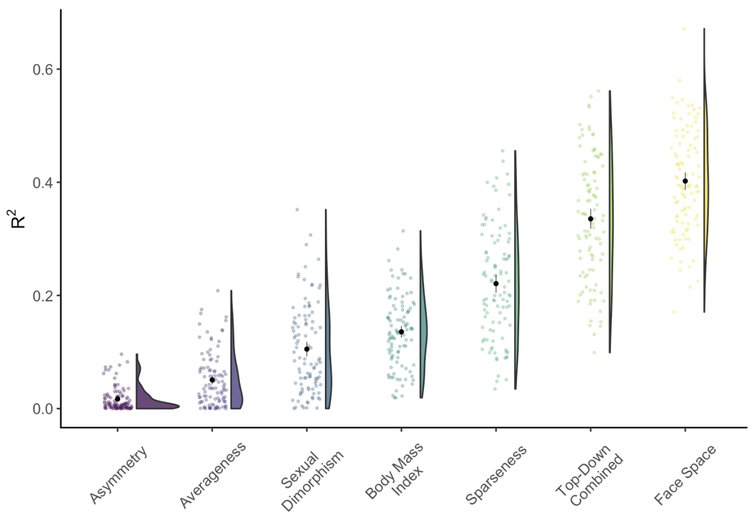
```
rmse.maintext<-tibble(
  "Asymmetry" = asym.rmse,
  "Averageness" = avg.rmse,
  "Sexual \nDimorphism" = sexdim.rmse,
  "Body Mass \nIndex" = bmi.rmse,
  "Sparseness" = sparse.rmse,
  "Top-Down \nCombined" = comb.rmse,
  "Face Space" = facespace.rmse
)

summary_maintext<- rmse.maintext %>%
  gather(model, rmse, 1:7) %>%
  mutate(model.f = factor(model, levels = unique(model))) %>%
  group_by(model.f) %>%
  summarise(
    mean = mean(rmse),
    min = mean(rmse) - qnorm(0.95) * sd(rmse) / sqrt(n()),
    max = mean(rmse) + qnorm(0.95) * sd(rmse) / sqrt(n())
  )

rmse.maintext %>%
  gather(model, rmse,1:7) %>%
  mutate(model.f = factor(model, levels = unique(model))) %>%
  ggplot(aes(x = model.f, y = rmse, fill = model.f)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), alpha = .8) +
  geom_point(aes(y = rmse, color = model.f), position = position_jitter(width = .15), size = .5, alpha = .3) +
  geom_pointrange(
    data = summary_maintext,
    aes(model.f, mean, ymin = min, ymax = max),
    shape = 20,
    size = .2) +
  guides(fill = FALSE, color = FALSE) +
  labs(x = "", y = expression(RMSE)) +
  scale_color_viridis(discrete = TRUE) +
  scale_fill_viridis(discrete = TRUE) +
  theme_bw() +
  raincloud_theme
```



```
ggsave("figure1.pdf", scale = 1, width = 100, height = 100, units = "mm", dpi = 300)
```

R2 plot for comparison with Said and Todorov (2011)

```
rsqs.maintext<-tibble(
  "Asymmetry" = asym.rsqs,
  "Averageness" = avg.rsqs,
  "Sexual \nDimorphism" = sexdim.rsqs,
  "Body Mass \nIndex" = bmi.rsqs,
  "Sparseness" = sparse.rsqs,
  "Top-Down \nCombined" = comb.rsqs,
  "Face Space" = facespace.rsqs
)

summary_maintext<- rsqs.maintext %>%
  gather(model, rsqs, 1:7) %>%
  mutate(model.f = factor(model, levels = unique(model))) %>%
  group_by(model.f) %>%
  summarise(
    mean = mean(rsqs),
    min = mean(rsqs) - qnorm(0.95) * sd(rsqs) / sqrt(n()),
    max = mean(rsqs) + qnorm(0.95) * sd(rsqs) / sqrt(n())
  )

rsqs.maintext %>%
  gather(model, rsqs, 1:7) %>%
  mutate(model.f = factor(model, levels = unique(model))) %>%
  ggplot(aes(x = model.f, y = rsqs, fill = model.f)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), alpha = .8) +
  geom_point(aes(y = rsqs, color = model.f), position = position_jitter(width = .15), size = .5, alpha = .3) +
  geom_pointrange(
    data = summary_maintext,
    aes(model.f, mean, ymin = min, ymax = max),
    shape = 20,
    size = .2) +
  guides(fill = FALSE, color = FALSE) +
  labs(x = "", y = expression(R^2)) +
  scale_color_viridis(discrete = TRUE) +
  scale_fill_viridis(discrete = TRUE) +
  theme_bw() +
  raincloud_theme
```



```
ggsave("figure1_RSQ.pdf", scale = 1, width = 100, height = 100, units = "mm", dpi = 300)
```

## Top-down combined vs face space

### Is face space model over-fitting?

AIC was used as a measure of model fit. Note that top-down combined model included BMI, which was missing for 49 women. To be able to compare AICs for top-down combined and faces space models, all models were run on subset of women for which BMI data was available.

```
# Create empty data frame to populate
AICs <- data.frame(resample = rep(NA, length(comb.cv$control$index)),
                  AIC.comb = rep(NA, length(comb.cv$control$index)),
                  AIC.full = rep(NA, length(comb.cv$control$index)))

for (i in 1:length(comb.cv$control$index)) {
  # indices used to create subsets in top-down combined cross-validation
  index <- comb.cv$control$index[[i]]
  # creating subsets according to indices, and excluding women with missing BMI data
  data <- attr[index,] %>% filter(!is.na(bmi))
  # writing out AICs
  AICs[i,1] <- names(comb.cv$control$index)[[i]]
  AICs[i,2] <- extractAIC(lm(comb.form, data = data))[2]
  AICs[i,3] <- extractAIC(lm(facespace.form, data = data))[2]
}
```

Despite its higher number of predictors, face space was better model of attractiveness than top-down combined model:

```
AICs %>%
  #mutate(full_smaller_comb=ifelse(AIC.full<AIC.comb,1,0)) %>%
  select(-resample) %>%
  summarize_all(funs(mean))
```

```
##    AIC.comb  AIC.full
## 1 -686.5374 -770.9889
```

```
min((AICs$AIC.comb - AICs$AIC.full))
```

```
## [1] 39.09857
```

### How much unique variance is explained by face space/theory combined model?

```
combLast.form <- att ~ poly(PC1, 2) + poly(PC2, 2) + poly(PC3, 2) + poly(PC4, 2) + poly(PC5, 2) + poly(PC6, 2) +
poly(PC7, 2) + poly(PC8, 2) + poly(PC9, 2) + poly(PC10, 2) + poly(PC11, 2) + poly(PC12, 2) + poly(PC.col1, 2) + p
oly(PC.col2, 2) + poly(PC.col3, 2) + poly(PC.col4, 2) + poly(PC.col5, 2) + poly(PC.col6, 2) + poly(PC.col7, 2) +
poly(PC.col8, 2) + poly(PC.col9, 2) + poly(PC.col10, 2) + poly(PC.col11, 2) + poly(PC.col12, 2) + poly(PC.col13,
2) + poly(PC.col14, 2) + poly(PC.col15, 2) + poly(PC.col16, 2) + poly(PC.col17, 2) + poly(PC.col18, 2) + poly(PC.
col19, 2) + poly(PC.col20, 2) + poly(PC.col21, 2) + poly(PC.col22, 2) + poly(PC.col23, 2) + poly(PC.col24, 2) + p
oly(PC.col25, 2) + poly(PC.col26, 2) + poly(PC.col27, 2) + poly(PC.col28, 2) + poly(PC.col29, 2) + poly(PC.col30,
2) + poly(PC.col31, 2) + poly(PC.col32, 2) + poly(PC.col33, 2) + poly(PC.col34, 2) + poly(PC.col35, 2) + poly(PC.
col36, 2) + poly(PC.col37, 2) + poly(PC.col38, 2) + poly(PC.col39, 2) + poly(PC.col40, 2) + poly(PC.col41, 2) + p
oly(PC.col42, 2) + poly(PC.col43, 2) + poly(PC.col44, 2) + poly(PC.col45, 2) + poly(PC.col46, 2) + poly(PC.col47,
2) + poly(PC.col48, 2) + poly(PC.col49, 2) + poly(PC.col50, 2) + poly(PC.col51, 2) + poly(PC.col52, 2) + poly(PC.
col53, 2) + poly(PC.col54, 2) + poly(PC.col55, 2) + poly(PC.col56, 2) + poly(PC.col57, 2) + poly(PC.col58, 2) + p
oly(PC.col59, 2) + poly(PC.col60, 2) +
  avg + avg.col + poly(sd.vector, 2) + poly(sd.vector.col, 2) + asym + poly(bmi, 2) + poly(alpha16, 2)

combFirst.form<-att ~ avg + avg.col + poly(sd.vector, 2) + poly(sd.vector.col, 2) + asym + poly(bmi, 2) + poly(al
pha16, 2) + poly(PC1, 2) + poly(PC2, 2) + poly(PC3, 2) + poly(PC4, 2) + poly(PC5, 2) + poly(PC6, 2) + poly(PC7, 2
) + poly(PC8, 2) + poly(PC9, 2) + poly(PC10, 2) + poly(PC11, 2) + poly(PC12, 2) + poly(PC.col1, 2) + poly(PC.col2
, 2) + poly(PC.col3, 2) + poly(PC.col4, 2) + poly(PC.col5, 2) + poly(PC.col6, 2) + poly(PC.col7, 2) + poly(PC.col
8, 2) + poly(PC.col9, 2) + poly(PC.col10, 2) + poly(PC.col11, 2) + poly(PC.col12, 2) + poly(PC.col13, 2) + poly(P
C.col14, 2) + poly(PC.col15, 2) + poly(PC.col16, 2) + poly(PC.col17, 2) + poly(PC.col18, 2) + poly(PC.col19, 2) +
poly(PC.col20, 2) + poly(PC.col21, 2) + poly(PC.col22, 2) + poly(PC.col23, 2) + poly(PC.col24, 2) + poly(PC.col25
, 2) + poly(PC.col26, 2) + poly(PC.col27, 2) + poly(PC.col28, 2) + poly(PC.col29, 2) + poly(PC.col30, 2) + poly(P
C.col31, 2) + poly(PC.col32, 2) + poly(PC.col33, 2) + poly(PC.col34, 2) + poly(PC.col35, 2) + poly(PC.col36, 2) +
poly(PC.col37, 2) + poly(PC.col38, 2) + poly(PC.col39, 2) + poly(PC.col40, 2) + poly(PC.col41, 2) + poly(PC.col42
, 2) + poly(PC.col43, 2) + poly(PC.col44, 2) + poly(PC.col45, 2) + poly(PC.col46, 2) + poly(PC.col47, 2) + poly(P
C.col48, 2) + poly(PC.col49, 2) + poly(PC.col50, 2) + poly(PC.col51, 2) + poly(PC.col52, 2) + poly(PC.col53, 2) +
poly(PC.col54, 2) + poly(PC.col55, 2) + poly(PC.col56, 2) + poly(PC.col57, 2) + poly(PC.col58, 2) + poly(PC.col59
, 2) + poly(PC.col60, 2)

model.facespace<- lm(facespace.form, data = filter(attr, !is.na(bmi)))
model.comb <- lm(comb.form,data = filter(attr, !is.na(bmi)))
model.combLast <- lm(combLast.form, data = filter(attr, !is.na(bmi)))
model.combFirst <- lm(combFirst.form, data = filter(attr, !is.na(bmi)))

summary(model.facespace)$adj.r.squared
```

```
## [1] 0.5556637
```

```
summary(model.comb)$adj.r.squared
```

```
## [1] 0.3438732
```

```
summary(model.combLast)$adj.r.squared
```

```
## [1] 0.6274946
```

```
summary(model.combFirst)$adj.r.squared
```

```
## [1] 0.6274946
```

Theory combined model explains 34.4%, face space model explains 55.6% of variance. Both models together explain 62.8%. From this, it follows that unique contribution of theory combined model is 7.2% (62.8-55.6); unique variance explained by face space model is 28.4%; shared variance is 55.6-28.4/34.4-7.2 = 27.2 %.

## Additional models

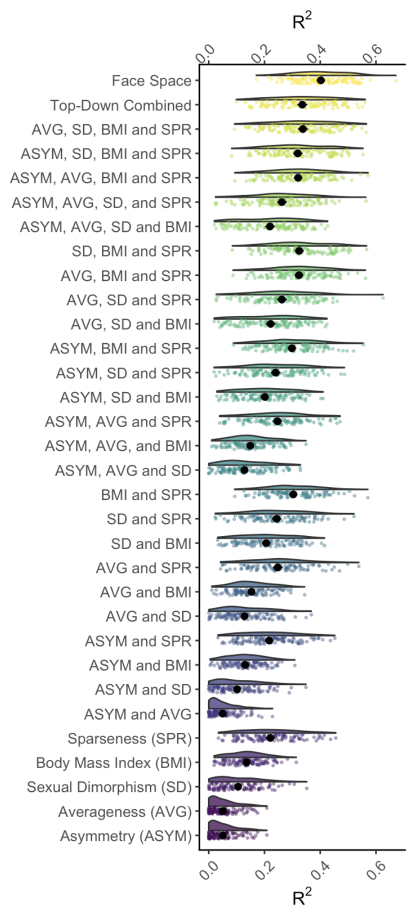
```r
# all possible two-way combinations (10)
asym_avg.form <- att ~ asym + avg + avg.col
asym_SD.form <- att ~ asym + poly(sd.vector, 2) + poly(sd.vector.col, 2)
asym_BMI.form <- att ~ asym + poly(bmi, 2)
asym_sparse.form <- att ~ asym + poly(alpha16, 2)
avg_SD.form <- att ~ avg + avg.col + poly(sd.vector, 2) + poly(sd.vector.col, 2)
avg_BMI.form <- att ~ avg + avg.col + poly(bmi, 2)
avg_sparse.form <- att ~ avg + avg.col + poly(alpha16, 2)
SD_BMI.form <- att ~ poly(sd.vector, 2) + poly(sd.vector.col, 2) + poly(bmi, 2)
SD_sparse.form <- att ~ poly(sd.vector, 2) + poly(sd.vector.col, 2) + poly(alpha16, 2)
BMI_sparse.form <- att ~ poly(bmi, 2) + poly(alpha16, 2)

# all possible three-way combinations (10)
asym_avg_SD.form <- att ~ asym + avg + avg.col + poly(sd.vector, 2) + poly(sd.vector.col, 2)
asym_avg_BMI.form <- att ~ asym + avg + avg.col + poly(bmi, 2)
asym_avg_sparse.form <- att ~ asym + avg + avg.col + poly(alpha16, 2)
asym_SD_BMI.form <- att ~ asym + poly(sd.vector, 2) + poly(sd.vector.col, 2) + poly(bmi, 2)
asym_SD_sparse.form <- att ~ asym + poly(sd.vector, 2) + poly(sd.vector.col, 2) + poly(alpha16, 2)
asym_BMI_sparse.form <- att ~ asym + poly(bmi, 2) + poly(alpha16, 2)
avg_SD_BMI.form <- att ~ avg + avg.col + poly(sd.vector, 2) + poly(sd.vector.col, 2) + poly(bmi, 2)
avg_SD_sparse.form <- att ~ avg + avg.col + poly(sd.vector, 2) + poly(sd.vector.col, 2) + poly(alpha16, 2)
avg_BMI_sparse.form <- att ~ avg + avg.col + poly(bmi, 2) + poly(alpha16, 2)
SD_BMI_sparse.form <- att ~ poly(sd.vector, 2) + poly(sd.vector.col, 2) + poly(bmi, 2) + poly(alpha16, 2)

# all possible four-way combinations (5)
asym_avg_SD_BMI.form <- att ~ asym + avg + avg.col + poly(sd.vector, 2) + poly(sd.vector.col, 2) + poly(bmi, 2)
asym_avg_SD_sparse.form <- att ~ asym + avg + avg.col + poly(sd.vector, 2) + poly(sd.vector.col, 2) + poly(alpha1
6, 2)
asym_avg_BMI_sparse.form <- att ~ asym + avg + avg.col + poly(bmi, 2) + poly(alpha16, 2)
asym_SD_BMI_sparse.form <- att ~ asym + poly(sd.vector, 2) + poly(sd.vector.col, 2) + poly(bmi, 2) + poly(alpha16
, 2)
avg_SD_BMI_sparse.form <- att ~ avg + avg.col + poly(sd.vector, 2) + poly(sd.vector.col, 2) + poly(bmi, 2) + poly
(alpha16, 2)

set.seed(1994)
asym_avg.cv <- train(asym_avg.form,
                     data = attr,
                     trControl = fitControl,
                     method = "lm")
asym_avg.rsqs <- asym_avg.cv$resample$Rsquared

set.seed(1994)
asym_SD.cv <- train(asym_SD.form,
                    data = attr,
                    trControl = fitControl,
                    method = "lm")
asym_SD.rsqs <- asym_SD.cv$resample$Rsquared

set.seed(1994)
asym_BMI.cv <- train(asym_BMI.form,
                     data = filter(attr, !is.na(bmi)),
                     trControl = fitControl,
                     method = "lm")
asym_BMI.rsqs <- asym_BMI.cv$resample$Rsquared

set.seed(1994)
asym_sparse.cv <- train(asym_sparse.form,
                        data = attr,
                        trControl = fitControl,
                        method = "lm")
asym_sparse.rsqs <- asym_sparse.cv$resample$Rsquared

set.seed(1994)
avg_SD.cv <- train(avg_SD.form,
                   data = attr,
                   trControl = fitControl,
                   method = "lm")
avg_SD.rsqs <- avg_SD.cv$resample$Rsquared

set.seed(1994)
avg_BMI.cv <- train(avg_BMI.form,
                    data = filter(attr, !is.na(bmi)),
                    trControl = fitControl,
                    method = "lm")
avg_BMI.rsqs <- avg_BMI.cv$resample$Rsquared

set.seed(1994)
avg_sparse.cv <- train(avg_sparse.form,
                       data = attr,
                       trControl = fitControl,
                       method = "lm")
avg_sparse.rsqs <- avg_sparse.cv$resample$Rsquared

set.seed(1994)
SD_BMI.cv <- train(SD_BMI.form,
                   data = filter(attr, !is.na(bmi)),
                   trControl = fitControl,
                   method = "lm")
SD_BMI.rsqs <- SD_BMI.cv$resample$Rsquared

set.seed(1994)
SD_sparse.cv <- train(SD_sparse.form,
                      data = attr,
                      trControl = fitControl,
                      method = "lm")
SD_sparse.rsqs <- SD_sparse.cv$resample$Rsquared
```

```
set.seed(1994)
BMI_sparse.cv <- train(BMI_sparse.form,
                       data = filter(attr, !is.na(bmi)),
                       trControl = fitControl,
                       method = "lm")
BMI_sparse.rsqs <- BMI_sparse.cv$resample$Rsquared

set.seed(1994)
asym_avg_SD.cv <- train(asym_avg_SD.form,
                        data = attr,
                        trControl = fitControl,
                        method = "lm")
asym_avg_SD.rsqs <- asym_avg_SD.cv$resample$Rsquared

set.seed(1994)
asym_avg_BMI.cv <- train(asym_avg_BMI.form,
                         data = filter(attr, !is.na(bmi)),
                         trControl = fitControl,
                         method = "lm")
asym_avg_BMI.rsqs <- asym_avg_BMI.cv$resample$Rsquared

set.seed(1994)
asym_avg_sparse.cv <- train(asym_avg_sparse.form,
                            data = attr,
                            trControl = fitControl,
                            method = "lm")
asym_avg_sparse.rsqs <- asym_avg_sparse.cv$resample$Rsquared

set.seed(1994)
asym_SD_BMI.cv <- train(asym_SD_BMI.form,
                        data = filter(attr, !is.na(bmi)),
                        trControl = fitControl,
                        method = "lm")
asym_SD_BMI.rsqs <- asym_SD_BMI.cv$resample$Rsquared

set.seed(1994)
asym_SD_sparse.cv <- train(asym_SD_sparse.form,
                           data = attr,
                           trControl = fitControl,
                           method = "lm")
asym_SD_sparse.rsqs <- asym_SD_sparse.cv$resample$Rsquared

set.seed(1994)
asym_BMI_sparse.cv <- train(asym_BMI_sparse.form,
                            data = filter(attr, !is.na(bmi)),
                            trControl = fitControl,
                            method = "lm")
asym_BMI_sparse.rsqs <- asym_BMI_sparse.cv$resample$Rsquared

set.seed(1994)
avg_SD_BMI.cv<- train(avg_SD_BMI.form,
                      data = filter(attr, !is.na(bmi)),
                      trControl = fitControl, method = "lm")
avg_SD_BMI.rsqs <- avg_SD_BMI.cv$resample$Rsquared

set.seed(1994)
avg_SD_sparse.cv <- train(avg_SD_sparse.form,
                          data = attr,
                          trControl = fitControl,
                          method = "lm")
avg_SD_sparse.rsqs <- avg_SD_sparse.cv$resample$Rsquared

set.seed(1994)
avg_BMI_sparse.cv <- train(avg_BMI_sparse.form,
                           data = filter(attr, !is.na(bmi)),
                           trControl = fitControl,
                           method = "lm")
avg_BMI_sparse.rsqs <- avg_BMI_sparse.cv$resample$Rsquared

set.seed(1994)
SD_BMI_sparse.cv <- train(SD_BMI_sparse.form,
                          data = filter(attr, !is.na(bmi)),
                          trControl = fitControl,
                          method = "lm")
SD_BMI_sparse.rsqs <- SD_BMI_sparse.cv$resample$Rsquared

set.seed(1994)
asym_avg_SD_BMI.cv <- train(asym_avg_SD_BMI.form,
                            data = filter(attr, !is.na(bmi)),
                            trControl = fitControl,
                            method = "lm")
asym_avg_SD_BMI.rsqs <- asym_avg_SD_BMI.cv$resample$Rsquared

set.seed(1994)
asym_avg_SD_sparse.cv <- train(asym_avg_SD_sparse.form,
                               data = attr,
                               trControl = fitControl,
                               method = "lm")
asym_avg_SD_sparse.rsqs <- asym_avg_SD_sparse.cv$resample$Rsquared

set.seed(1994)
asym_avg_BMI_sparse.cv <- train(asym_avg_BMI_sparse.form,
                                data = filter(attr, !is.na(bmi)),
                                trControl = fitControl,
                                method = "lm")
asym_avg_BMI_sparse.rsqs <- asym_avg_BMI_sparse.cv$resample$Rsquared
```

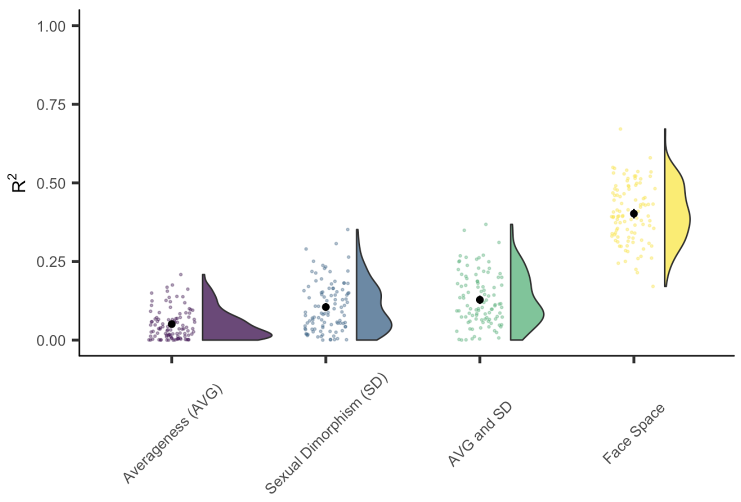
```
set.seed(1994)
asym_SD_BMI_sparse.cv <- train(asym_SD_BMI_sparse.form,
                               data = filter(attr, !is.na(bmi)),
                               trControl = fitControl,
                               method = "lm")
asym_SD_BMI_sparse.rsqs <- asym_SD_BMI_sparse.cv$resample$Rsquared

set.seed(1994)
avg_SD_BMI_sparse.cv <- train(avg_SD_BMI_sparse.form,
                              data = filter(attr, !is.na(bmi)),
                              trControl = fitControl,
                              method = "lm")
avg_SD_BMI_sparse.rsqs <- avg_SD_BMI_sparse.cv$resample$Rsquared
```

```
rsqs.SI<-tibble(
  "Asymmetry (ASYM)" = avg.rsqs,
  "Averageness (AVG)" = avg.rsqs,
  "Sexual Dimorphism (SD)" = sexdim.rsqs,
  "Body Mass Index (BMI)" = bmi.rsqs,
  "Sparseness (SPR)" = sparse.rsqs,
  "ASYM and AVG"  = asym_avg.rsqs,
  "ASYM and SD"  = asym_SD.rsqs,
  "ASYM and BMI" = asym_BMI.rsqs,
  "ASYM and SPR" = asym_sparse.rsqs,
  "AVG and SD" = avg_SD.rsqs,
  "AVG and BMI" = avg_BMI.rsqs,
  "AVG and SPR" = avg_sparse.rsqs,
  "SD and BMI" = SD_BMI.rsqs,
  "SD and SPR" = SD_sparse.rsqs,
  "BMI and SPR" = BMI_sparse.rsqs,
  "ASYM, AVG and SD" = asym_avg_SD.rsqs,
  "ASYM, AVG, and BMI" = asym_avg_BMI.rsqs,
  "ASYM, AVG and SPR" = asym_avg_sparse.rsqs,
  "ASYM, SD and BMI" = asym_SD_BMI.rsqs,
  "ASYM, SD and SPR" = asym_SD_sparse.rsqs,
  "ASYM, BMI and SPR" = asym_BMI_sparse.rsqs,
  "AVG, SD and BMI" = avg_SD_BMI.rsqs,
  "AVG, SD and SPR" = avg_SD_sparse.rsqs,
  "AVG, BMI and SPR" = avg_BMI_sparse.rsqs,
  "SD, BMI and SPR" = SD_BMI_sparse.rsqs,
  "ASYM, AVG, SD and BMI" = asym_avg_SD_BMI.rsqs,
  "ASYM, AVG, SD, and SPR" = asym_avg_SD_sparse.rsqs,
  "ASYM, AVG, BMI and SPR" = asym_avg_BMI_sparse.rsqs,
  "ASYM, SD, BMI and SPR" = asym_SD_BMI_sparse.rsqs,
  "AVG, SD, BMI and SPR" = avg_SD_BMI_sparse.rsqs,
  "Top-Down Combined" = comb.rsqs,
  "Face Space" = facespace.rsqs
)
```

```
summary_SI <- rsqs.SI %>%
  gather(model, rsqs, 1:32) %>%
  mutate(model.f = factor(model, levels = unique(model))) %>%
  group_by(model.f) %>%
  summarise(
    mean = mean(rsqs),
    min = mean(rsqs) - qnorm(0.95) * sd(rsqs) / sqrt(n()),
    max = mean(rsqs) + qnorm(0.95) * sd(rsqs) / sqrt(n())
  )

rsqs.SI %>%
  gather(model,rsqs,1:32) %>%
  mutate(model.f = factor(model, levels = unique(model))) %>%
  ggplot(aes(x = model.f, y = rsqs, fill = model.f)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), alpha = .8) +
  geom_point(aes(y = rsqs, color = model.f), position = position_jitter(width = .15), size = .5, alpha = 0.4) +
  geom_pointrange(
    data = summary_SI,
    aes(model.f, mean, ymin = min, ymax = max),
    shape = 20
  ) +
  guides(fill = FALSE, color=FALSE) +
  scale_y_continuous(sec.axis = sec_axis(~., name = expression(R^2)))+
  labs(x = "", y = expression(R^2)) +
  scale_color_viridis(discrete = TRUE) +
  scale_fill_viridis(discrete = TRUE) +
  theme_bw() +
  raincloud_theme +
  coord_flip()
```



```
summary_SI_SandT <- rsqs.SI %>%
  gather(model, rsqs, 2, 3, 10, 32) %>%
  mutate(model.f=factor(model, levels = unique(model))) %>%
  group_by(model.f) %>%
  summarise(
    mean = mean(rsqs),
    min = mean(rsqs) - qnorm(0.95) * sd(rsqs) / sqrt(n()),
    max = mean(rsqs) + qnorm(0.95) * sd(rsqs) / sqrt(n())
  )

rsqs.SI %>%
  gather(model,rsqs, 2, 3, 10, 32) %>%
  mutate(model.f = factor(model, levels = unique(model))) %>%
  ggplot(aes(x=model.f, y = rsqs, fill = model.f)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), alpha = .8) +
  geom_point(aes(y = rsqs, color = model.f), position = position_jitter(width = .15), size = .5, alpha = 0.4) +
  geom_pointrange(
    data = summary_SI_SandT,
    aes(model.f, mean, ymin = min, ymax = max),
    shape = 20
  ) +
  guides(fill = FALSE, color=FALSE) +
  labs(x = "", y = expression(R^2)) +
  scale_color_viridis(discrete = TRUE) +
  scale_fill_viridis(discrete = TRUE) +
  ylim(0, 1) +
  theme_bw(base_size = 20) +
  raincloud_theme
```



## Visualizing most important predictors

### Extract average coefficient for each PC

```
# Save estimates for each resample
for (i in 1:length(facespace.cv$control$index)) {
  # pull index
  index <- facespace.cv$control$index[[i]]
  # subset data
  data <- attr[index,]
  # run model
  model <- lm(facespace.form, data = data)
  summary <- summary(model)
  # create output table
  if (i==1) {
    output <- summary$coefficients %>%
      as.data.frame() %>%
      rownames_to_column() %>%
      rename(
        effect = rowname,
        estimate = "Estimate",
        p = "Pr(>|t|)"
      ) %>%
      select(effect, estimate, p) %>%
      mutate(sig = ifelse(p <= .05, 1, 0),
             fold = names(facespace.cv$control$index)[i])
  }
  else{
    temp <- summary$coefficients %>%
      as.data.frame() %>%
      rownames_to_column() %>%
      rename(
        effect = rowname,
        estimate = "Estimate",
        p = "Pr(>|t|)"
      ) %>%
      select(effect, estimate, p) %>%
      mutate(sig = ifelse(p <= .05,1,0),
             fold = names(facespace.cv$control$index)[i])

    output<-bind_rows(output, temp)
  }
}

# Calculate average coefficients for each PC
average_coeffs <- output %>%
  mutate(
    # create variable for whether coeff is for linear or quadratic term
    term = ifelse(str_detect(effect, "\\,[[:space:]]2\\)1"), "linear", "quadratic"),
    type = ifelse(str_detect(effect, "PC\\.col"), "colour", "shape"),
    # convert factor "var" into string "PC", strip unwanted bits
    PC = as.character(effect) %>%
      gsub("poly\\(", "", .) %>%
      gsub("\\,[[:space:]]2\\)[[:digit:]]", "", .) %>%
      gsub(".col", "", .)
  ) %>%
  mutate(
    # if PC ends in single digit, insert 0
    PC = ifelse(str_detect(PC,"[0-9][0-9]$"), PC, gsub('^(.*)([0-9])$', "\\10\\2", PC))
  ) %>%
  # averaging coefficients
  group_by(PC, type, term) %>%
  summarise(estimate_mean = mean(estimate),
            sign_mean = mean(sig)) %>%
  ungroup() %>%
  unite("name", c("PC","type"), remove = FALSE) %>%
  select(-PC, -type)

average_linear <- average_coeffs %>%
  filter(term=="linear")

average_quadratic <- average_coeffs %>%
  filter(term=="quadratic")
```

## Choose PCs to visualize based on variable selection using AIC

Save best predictors from each resample into separate .csv file

```r
library(MASS)

for (i in 1:length(facespace.cv$control$index)) {
  # pull index
  index <- facespace.cv$control$index[[i]]
  # subset data
  data <- attr[index,]
  # run model
  model <- lm(facespace.form, data = data)
  # use stepAIC to find best model
  step <- stepAIC(model, direction = "backward", trace = 0)
  # save predictors from final model to .csv
  attributes(step$anova)$heading %>%
    paste0(collapse = '') %>%
    substring(regexpr("\nFinal Model:", .)) %>%
    str_extract_all(regex("PC.*?\\,")) %>%
    unlist() %>%
    gsub("\\,", "", .) %>%
    as.matrix(ncol = 1, byrow = T) %>%
    as.data.frame() %>%
    rename_all((funs(paste0("resample_",i)))) %>%
    write_csv(paste0("./dataFiles/2_bestPreds/resample_", i, ".csv"))
}

for (i in 1:length(facespace.cv$control$index)) {
  # pull index
  index <- facespace.cv$control$index[[i]]
  # subset data
  data<-attr[index,]
  # run model
  model<-lm(facespace.form, data=data)
  # use stepAIC to find best model
  step<-stepAIC(model, direction="backward",trace=0)
  # save predictors from final model to .csv
  step$call[[2]] %>%
    paste0(collapse = '') %>%
    substring(regexpr("~att", .)) %>%
    str_extract_all(regex("PC.*?\\,")) %>%
    unlist() %>%
    gsub("\\,", "", .) %>%
    as.matrix(ncol=1,byrow=T) %>%
    as.data.frame() %>%
    rename_all((funs(paste0("resample_",i)))) %>%
    write_csv(paste0("./dataFiles/2_bestPreds/resample_", i, ".csv"))
}

detach("package:MASS", unload=TRUE)
```

Read .csv files listing best predictors

```r
files <- paste0("./dataFiles/2_bestPreds/", list.files("./dataFiles/2_bestPreds", pattern = "*.csv"))

best.preds <- vector("list", length(files))
for(i in 1:length(files)){
  best.preds[[i]] <- read_csv(files[i], col_names = TRUE, col_types = cols(.default = "c"))
}
```

Find predictors that were selected across *all* resamples

```r
top.AIC <- table(unlist(best.preds)) %>%
  as.data.frame() %>%
  filter(Freq==100) %>%
  select(-Freq) %>%
  mutate(
    type = ifelse(str_detect(Var1, "PC\\.col"), "colour", "shape"),
    # convert factor "var" into string "PC", strip unwanted bits
    PC = as.character(Var1) %>% gsub(".col", "", .)
  ) %>%
  mutate(
    # if PC ends in single digit, insert 0
    PC = ifelse(str_detect(PC, "[0-9][0-9]$"), PC, gsub('^(.*)([0-9])$', "\\10\\2", PC))
  ) %>%
  unite("name",c("PC", "type"), remove = FALSE) %>%
  arrange(type, PC) %>%
  mutate(
    position = gsub('^(.*)([0-9][0-9])$', "\\2", PC) %>% as.numeric()
  ) %>%
  rename(df.id = "Var1") %>%
  mutate(df.id = as.character(df.id)) %>%
  left_join(average_linear, by = "name") %>%
  select(-term) %>%
  rename(estimate_linear = "estimate_mean",
         sign_linear = "sign_mean") %>%
  left_join(average_quadratic, by = "name") %>%
  select(-term) %>%
  rename(estimate_quad = "estimate_mean",
         sign_quad = "sign_mean")

top.AIC
```

## Load functions and data for visualizing

```r
source("./datafiles/visualize.R")

# RGB values/shape coordinates at 1SD of each PC
load("./datafiles/OCMATE_PCs.sd_colour.Rdata")
load("./datafiles/OCMATE_PCs.sd_shape.Rdata")

# Loading base face (sample average) on which visuals will be based
base <- read.webmorph("./datafiles/average.png")
# Removing additional FRL landmarks we did not use in our analyses
base@tem <- remove.outer(base@tem)
```

## Generate images of selected PCs at SD -3 to +3

```r
# Creating shape and colour vectors that keep all PCs at 0 = average
colour_empty<-rep(0, 60)
shape_empty<-rep(0, 12)

# Visualize SDs -3 to +3 in steps of 1
SD = 1
while(SD <= 3){

  # Generate images
  for (i in 1:nrow(top.AIC)){

    shape.values <- shape_empty
    colour.values <- colour_empty
    PC <- top.AIC$name[i]
    pos <- top.AIC$position[i]

    if(top.AIC$type[i]=="colour"){
      colour.values[pos] <- SD
    } else{
      shape.values[pos] <- SD
    }

    out.pos <- generate.stimuli(base,
                                colour.values,
                                shape.values,
                                PCs.sd_colour,
                                PCs.sd_shape)

    out.neg <- generate.stimuli(base,
                                colour.values * -1,
                                shape.values * -1,
                                PCs.sd_colour,
                                PCs.sd_shape)

    writePNG(out.pos, target = paste0("./bestPredictors/", PC, "_", SD, "_pos.png"))
    writePNG(out.neg, target = paste0("./bestPredictors/", PC, "_", SD, "_neg.png"))
  }
  SD = SD + 1
}
```

## Create plot for each PC with images on x-axis

```r
## load the images from filenames
images<-list.files(path = "./bestPredictors", pattern = "\\.png$", full.names = TRUE) %>%
  as.data.frame(stringsAsFactors = FALSE)

# Create vector for reshuffling images into correct order
order.lines=c(6, 4, 2, 1, 3, 5, 7)

for (i in 1:nrow(top.AIC)){
  # Extract images for specific PC plus average
  PC.images <- images %>%
    # select the 6 respective images for each PC + the average
    filter(grepl(top.AIC$name[i], .) | grepl("average",.)) %>%
    mutate(line.number = row_number()) %>%
    # order images from -3 to +3
    arrange(match(line.number, order.lines)) %>%
    select(-line.number)
  # Create empty list for images and labels
  pics  <- vector(mode="list",
                  length = nrow(PC.images))
  # Populate list with corresponding images
  for(j in 1:nrow(PC.images)) {
    pics[[j]] <- magick::image_read(PC.images[j, 1])
  }
  # Populate list with corresponding labls
  names(pics)<-seq(-3, 3, 1) %>%
    as.character()

  # Create scatterplot with images on x-axis
  # if quadratic term was significant in less than 80% of samples, plot linear relationship only
  if (top.AIC$sign_quad[i]<=.8){
    # note - mean intercept = 0
    y = top.AIC$estimate_linear[i] * -3
    yend = top.AIC$estimate_linear[i] * 3

    attr %>%
      # use get() to convert string to variable name
      ggplot(aes(x = get(top.AIC$df.id[i]), y = att)) +
      scale_x_continuous(limits = c(-3, 3), breaks = seq(-3, 3, 1), label = pics, position = "top") +
      scale_y_continuous(limits = c(-3, 3), breaks = seq(-3, 3, 3)) +
      labs(title = "",
           subtitle = "",
           x = "",
           y = "")+
      theme(axis.text.x = my_axis(pics, angle = 0),
            axis.title.x = element_blank(),
            axis.ticks.x = element_blank(),
            axis.title.y = element_blank(),
            panel.border = element_rect(colour = "grey20", fill = NA),
            panel.background = element_rect(fill = "white"),
            panel.spacing = unit(5.5, "lines"),
            panel.grid = element_line(colour = "grey92")
      ) +
      geom_segment(x = -3, y = y, xend = 3, yend = yend, size = .5, colour = "red") +
      coord_fixed(.2)
  } else{
    # if quadratic term was significant in more than 80% of samples, plot curvilinear relationship
    fun <- function(x) {top.AIC$estimate_linear[i] * x + top.AIC$estimate_quad[i] * x^2}

    attr %>%
      ggplot(aes(x = get(top.AIC$df.id[i]), y = att)) +
      scale_x_continuous(limits = c(-3, 3), breaks = seq(-3, 3, 1), label = pics, position = "top") +
      scale_y_continuous(limits = c(-3, 3), breaks = seq(-3, 3, 3))+
      labs(title = "",
           subtitle = "",
           x = "",
           y = "")+
      theme(axis.text.x = my_axis(pics, angle = 0),
            axis.title.x = element_blank(),
            axis.ticks.x = element_blank(),
            axis.title.y = element_blank(),
            panel.border = element_rect(colour = "grey20", fill = NA),
            panel.background = element_rect(fill = "white"),
            panel.spacing = unit(5.5, "lines"),
            panel.grid = element_line(colour = "grey92")
      ) +
      stat_function(fun = fun, colour = "red") +
      coord_fixed(.2)
  }
  ggsave("./bestPredictors/", paste0(top.AIC$name[i], ".png"), width = 3, height = 2.5, dpi = 300)
}
```