# Results 1/21

```
library(circular)
```

```
##
## Attaching package: 'circular'
```

```
## The following objects are masked from 'package:stats':
##
##     sd, var
```

```
library(tidyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(bpnreg)
library(mltools)
```

```
##
## Attaching package: 'mltools'
```

```
## The following object is masked from 'package:tidyr':
##
##     replace_na
```

```
library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
library(handwriter)

mean_per_cluster=read.csv("mean_per_cluster.csv")

#removing ambi
mean_per_cluster=mean_per_cluster[mean_per_cluster$Dominant.Hand != "ambidextrous", ]
mean_per_cluster=droplevels(mean_per_cluster)
table(mean_per_cluster$Dominant.Hand)
```

```
##
##  left right
##   393  3103
```

```
#didn't include any region
partdata = select(mean_per_cluster, "writer", "cluster","mean_per_cluster", "Age.Group",
                  "Gender", "Dominant.Hand")

summary(is.na(partdata))
```

```
##    writer         cluster        mean_per_cluster Age.Group
##  Mode :logical   Mode :logical   Mode :logical    Mode :logical
##  FALSE:3496      FALSE:3496      FALSE:3496       FALSE:3496
##    Gender        Dominant.Hand
##  Mode :logical   Mode :logical
##  FALSE:3496      FALSE:3496
```

```
#none missing (also have more data points because all of the missing values were from region and
since we aren't using them we have more usable rows)



##########Updating Age groups###########
test <- data.table(partdata) %>%
  .[Age.Group == "25-40", Age.Group := "18-40"]

test <- data.table(test) %>%
  .[Age.Group == "18-24", Age.Group := "18-40"]

test <- data.table(test) %>%
  .[Age.Group == "61+", Age.Group := "41+"]
test <- data.table(test) %>%
  .[Age.Group == "41-60", Age.Group := "41+"]

levels(test$Age.Group)
```

```
## [1] "18-24" "25-40" "41-60" "61+"   "18-40" "41+"
```

```
test=droplevels(test)


mean_per_cluster=test
```

```
y=circular(mean_per_cluster$mean_per_cluster, units='radians', rotation='counter', type='angles'
)
mean_per_cluster$gender=as.factor(mean_per_cluster$Gender)
mean_per_cluster$age=as.factor(mean_per_cluster$Age.Group)
mean_per_cluster$hand=as.factor(mean_per_cluster$Dominant.Hand)
mean_per_cluster$writer=as.numeric(mean_per_cluster$writer)


me_model <- bpnme(pred.I = y ~ age + gender + hand  + (1|writer),
                  data = mean_per_cluster, its=5000, burn=2000)
```

```
## [1] "burn-in iteration 1950"
## [1] "burn-in iteration 1900"
## [1] "burn-in iteration 1850"
## [1] "burn-in iteration 1800"
## [1] "burn-in iteration 1750"
## [1] "burn-in iteration 1700"
## [1] "burn-in iteration 1650"
## [1] "burn-in iteration 1600"
## [1] "burn-in iteration 1550"
## [1] "burn-in iteration 1500"
## [1] "burn-in iteration 1450"
## [1] "burn-in iteration 1400"
## [1] "burn-in iteration 1350"
## [1] "burn-in iteration 1300"
## [1] "burn-in iteration 1250"
## [1] "burn-in iteration 1200"
## [1] "burn-in iteration 1150"
## [1] "burn-in iteration 1100"
## [1] "burn-in iteration 1050"
## [1] "burn-in iteration 1000"
## [1] "burn-in iteration 950"
## [1] "burn-in iteration 900"
## [1] "burn-in iteration 850"
## [1] "burn-in iteration 800"
## [1] "burn-in iteration 750"
## [1] "burn-in iteration 700"
## [1] "burn-in iteration 650"
## [1] "burn-in iteration 600"
## [1] "burn-in iteration 550"
## [1] "burn-in iteration 500"
## [1] "burn-in iteration 450"
## [1] "burn-in iteration 400"
## [1] "burn-in iteration 350"
## [1] "burn-in iteration 300"
## [1] "burn-in iteration 250"
## [1] "burn-in iteration 200"
## [1] "burn-in iteration 150"
## [1] "burn-in iteration 100"
## [1] "burn-in iteration 50"
## [1] "burn-in iteration 0"
## [1] "iteration 50"
## [1] "iteration 100"
## [1] "iteration 150"
## [1] "iteration 200"
## [1] "iteration 250"
## [1] "iteration 300"
## [1] "iteration 350"
## [1] "iteration 400"
## [1] "iteration 450"
## [1] "iteration 500"
## [1] "iteration 550"
## [1] "iteration 600"
## [1] "iteration 650"
```

```
## [1] "iteration 700"
## [1] "iteration 750"
## [1] "iteration 800"
## [1] "iteration 850"
## [1] "iteration 900"
## [1] "iteration 950"
## [1] "iteration 1000"
## [1] "iteration 1050"
## [1] "iteration 1100"
## [1] "iteration 1150"
## [1] "iteration 1200"
## [1] "iteration 1250"
## [1] "iteration 1300"
## [1] "iteration 1350"
## [1] "iteration 1400"
## [1] "iteration 1450"
## [1] "iteration 1500"
## [1] "iteration 1550"
## [1] "iteration 1600"
## [1] "iteration 1650"
## [1] "iteration 1700"
## [1] "iteration 1750"
## [1] "iteration 1800"
## [1] "iteration 1850"
## [1] "iteration 1900"
## [1] "iteration 1950"
## [1] "iteration 2000"
## [1] "iteration 2050"
## [1] "iteration 2100"
## [1] "iteration 2150"
## [1] "iteration 2200"
## [1] "iteration 2250"
## [1] "iteration 2300"
## [1] "iteration 2350"
## [1] "iteration 2400"
## [1] "iteration 2450"
## [1] "iteration 2500"
## [1] "iteration 2550"
## [1] "iteration 2600"
## [1] "iteration 2650"
## [1] "iteration 2700"
## [1] "iteration 2750"
## [1] "iteration 2800"
## [1] "iteration 2850"
## [1] "iteration 2900"
## [1] "iteration 2950"
## [1] "iteration 3000"
## [1] "iteration 3050"
## [1] "iteration 3100"
## [1] "iteration 3150"
## [1] "iteration 3200"
## [1] "iteration 3250"
## [1] "iteration 3300"
## [1] "iteration 3350"
```

```
## [1] "iteration 3400"
## [1] "iteration 3450"
## [1] "iteration 3500"
## [1] "iteration 3550"
## [1] "iteration 3600"
## [1] "iteration 3650"
## [1] "iteration 3700"
## [1] "iteration 3750"
## [1] "iteration 3800"
## [1] "iteration 3850"
## [1] "iteration 3900"
## [1] "iteration 3950"
## [1] "iteration 4000"
## [1] "iteration 4050"
## [1] "iteration 4100"
## [1] "iteration 4150"
## [1] "iteration 4200"
## [1] "iteration 4250"
## [1] "iteration 4300"
## [1] "iteration 4350"
## [1] "iteration 4400"
## [1] "iteration 4450"
## [1] "iteration 4500"
## [1] "iteration 4550"
## [1] "iteration 4600"
## [1] "iteration 4650"
## [1] "iteration 4700"
## [1] "iteration 4750"
## [1] "iteration 4800"
## [1] "iteration 4850"
## [1] "iteration 4900"
## [1] "iteration 4950"
## [1] "iteration 5000"
```

```
me_model
```

```
## Projected Normal Mixed Effects
##
## Model
##
## Call:
## bpnme(pred.I = y ~ age + gender + hand + (1 | writer), data = mean_per_cluster,
##     its = 5000, burn = 2000)
##
## MCMC:
## iterations = 5000
## burn-in = 2000
## lag = 1
##
## Model Fit:
##         Statistic Parameters
## lppd    -2812.264     8.0000
## DIC      5916.913   136.6701
## DIC.alt  5988.266   172.3468
## WAIC     5935.958   155.7148
## WAIC2    5947.652   161.5619
##
##
## Fixed Effects
##
## Linear Coefficients
##
## Component I:
##                    mean       mode         sd     LB HPD     UB HPD
## (Intercept) -1.97141998 -2.0373793 0.2063912 -2.3741598 -1.5670073
## age41+       0.49258356  0.4766567 0.1312755  0.2401041  0.7458294
## gendermale   0.10394846  0.1205782 0.1281175 -0.1507384  0.3457356
## handright    0.03318506  0.0384817 0.2063782 -0.3854544  0.4146826
##
## Component II:
##                   mean       mode         sd      LB HPD    UB HPD
## (Intercept) -0.3438346 -0.3310818 0.3252796 -0.99174141 0.2846663
## age41+       0.7537136  0.7646606 0.2052440  0.35815075 1.1511465
## gendermale   0.4045748  0.4127675 0.2071644  0.01161101 0.8255689
## handright    0.5028739  0.5405753 0.3253566 -0.14272239 1.1336872
##
##
## Circular Coefficients
##
## Continuous variables:
## [1] "There are no numeric predictors in the model"
##
## Categorical variables:
##
## Means:
##                          mean      mode         sd       LB       UB
## (Intercept)         -2.971253 -2.947324 0.15976908 3.014782 3.644061
## age41+               2.878082  2.902995 0.21452232 2.469614 3.302766
## gendermale           3.109754  3.125419 0.17419760 2.758725 3.431940
```

```
## handright            3.060010  3.065286 0.08734150 2.883637 3.224630
## age41+gendermale     2.618969  2.651506 0.20892504 2.227774 3.039550
## age41+handright      2.580999  2.585359 0.09147825 2.413510 2.775026
## gendermalehandright  2.844986  2.852415 0.09084784 2.658928 3.015850
##
## Differences:
##                          mean      mode        sd          LB        UB
## age41+              0.4335104 0.4217937 0.1366821  0.16042759 0.6913939
## gendermale          0.2020540 0.1892774 0.1084581 -0.02021415 0.4048465
## handright           0.2519052 0.2482699 0.1610127 -0.08622788 0.5499665
## age41+gendermale    0.6923297 0.7141534 0.1848159  0.33443525 1.0562245
## age41+handright     0.7309260 0.7802934 0.1854305  0.37414170 1.0903013
## gendermalehandright 0.4670129 0.4804111 0.1899877  0.10069065 0.8505615
##
##
## Random Effects
##
## Linear Coefficients
##
## Component I:
##       mean      mode        sd    LB HPD    UB HPD
## RI 0.30964 0.2780292 0.0555953 0.2073153 0.4199185
##
## Component II:
##         mean      mode        sd    LB HPD   UB HPD
## RI 0.8609729 0.8279964 0.1435687 0.5918366 1.146456
##
##
## Circular Coefficients
##
##         mean      mode         sd       LB       UB
## RI 0.8986063 0.9009289 0.01976535 0.860445 0.935822
```

```
beta1 = me_model$Beta.I
beta2 = me_model$Beta.II

# The yhats are the predicted values for each type of person and each linear
# component. For example yhat000.I is the predicted first component for women
# 18-40, left-handed.  The first 8 are for women and the next set of 8
# are for the men.
yhat000.I =  beta1[,1]
yhat000.II =  beta2[,1]
yhat001.I =  beta1[,1] + beta1[,4]
yhat001.II =  beta2[,1] + beta2[,4]
yhat100.I =  beta1[,1] + beta1[,2]
yhat100.II =  beta2[,1] + beta2[,2]
yhat101.I =  beta1[,1] + beta1[,2] + beta1[,4]
yhat101.II =  beta2[,1] + beta2[,2] + beta2[,4]

yhat010.I =  beta1[,1] + beta1[,3]
yhat010.II =  beta2[,1] + beta2[,3]
yhat011.I =  beta1[,1] + beta1[,3] + beta1[,4]
yhat011.II =  beta2[,1] + beta2[,3] + beta2[,4]
yhat110.I =  beta1[,1] + beta1[,2] + beta1[,3]
yhat110.II =  beta2[,1] + beta2[,2] + beta2[,3]
yhat111.I =  beta1[,1] + beta1[,2] + beta1[,3] + beta1[,4]
yhat111.II =  beta2[,1] + beta2[,2] + beta2[,3] + beta2[,4]

# Check to see whether the first component is always negative.
mean(yhat000.I)
```

```
## [1] -1.97142
```

```
mean(yhat001.I)
```

```
## [1] -1.938235
```

```
mean(yhat100.I)
```

```
## [1] -1.478836
```

```
mean(yhat101.I)
```

```
## [1] -1.445651
```

```
mean(yhat010.I)
```

```
## [1] -1.867472
```

```
mean(yhat011.I)
```

```
## [1] -1.834286
```

```
mean(yhat110.I)
```

```
## [1] -1.374888
```

```
mean(yhat111.I)
```

```
## [1] -1.341703
```

```
# Now check to see whether the second component is always positive
mean(yhat000.II)
```

```
## [1] -0.3438346
```

```
mean(yhat001.II)
```

```
## [1] 0.1590393
```

```
mean(yhat100.II)
```

```
## [1] 0.409879
```

```
mean(yhat101.II)
```

```
## [1] 0.9127529
```

```
mean(yhat010.II)
```

```
## [1] 0.06074024
```

```
mean(yhat011.II)
```

```
## [1] 0.5636141
```

```
mean(yhat110.II)
```

```
## [1] 0.8144539
```

```
mean(yhat111.II)
```

```
## [1] 1.317328
```

```
#  Now compute the predicted angles in degrees for each type of person
theta000 = (atan(yhat000.II/yhat000.I) + pi)*180/pi
theta001 = (atan(yhat001.II/yhat001.I) + pi)*180/pi
theta100 = (atan(yhat100.II/yhat100.I) + pi)*180/pi
theta101 = (atan(yhat101.II/yhat101.I) + pi)*180/pi
theta010 = (atan(yhat010.II/yhat010.I) - pi)*180/pi # yhat010.II was negative
theta011 = (atan(yhat011.II/yhat011.I) + pi)*180/pi
theta110 = (atan(yhat110.II/yhat110.I) + pi)*180/pi
theta111 = (atan(yhat111.II/yhat111.I) + pi)*180/pi

theta = data.frame(theta000, theta001, theta100, theta101, theta010, theta011, theta110, theta11
1)
```

```
####means of theta
mean(theta[,1])
```

```
## [1] 189.7539
```

```
quantile(theta[,1], p=c(0.025, 0.975))
```

```
##     2.5%    97.5% 
## 171.0974 207.2172
```

```
mean(theta[,2])
```

```
## [1] 175.326
```

```
quantile(theta[,2], p=c(0.025, 0.975))
```

```
##     2.5%    97.5% 
## 165.5986 185.2411
```

```
mean(theta[,3])
```

```
## [1] 164.9113
```

```
quantile(theta[,3], p=c(0.025, 0.975))
```

```
##     2.5%    97.5% 
## 141.5056 189.2364
```

```
mean(theta[,4])
```

```
## [1] 147.8816
```

```
quantile(theta[,4], p=c(0.025, 0.975))
```

```
##     2.5%    97.5% 
## 138.0922 158.8752
```

```
mean(theta[,5])
```

```
## [1] -181.8245
```

```
quantile(theta[,5], p=c(0.025, 0.975))
```

```
##      2.5%     97.5% 
## -201.3860 -162.7128
```

```
mean(theta[,6])
```

```
## [1] 163.0062
```

```
quantile(theta[,6], p=c(0.025, 0.975))
```

```
##     2.5%    97.5% 
## 152.8710 173.4994
```

```
mean(theta[,7])
```

```
## [1] 150.0815
```

```
quantile(theta[,7], p=c(0.025, 0.975))
```

```
##     2.5%    97.5% 
## 128.3651 175.2380
```

```
mean(theta[,8])
```

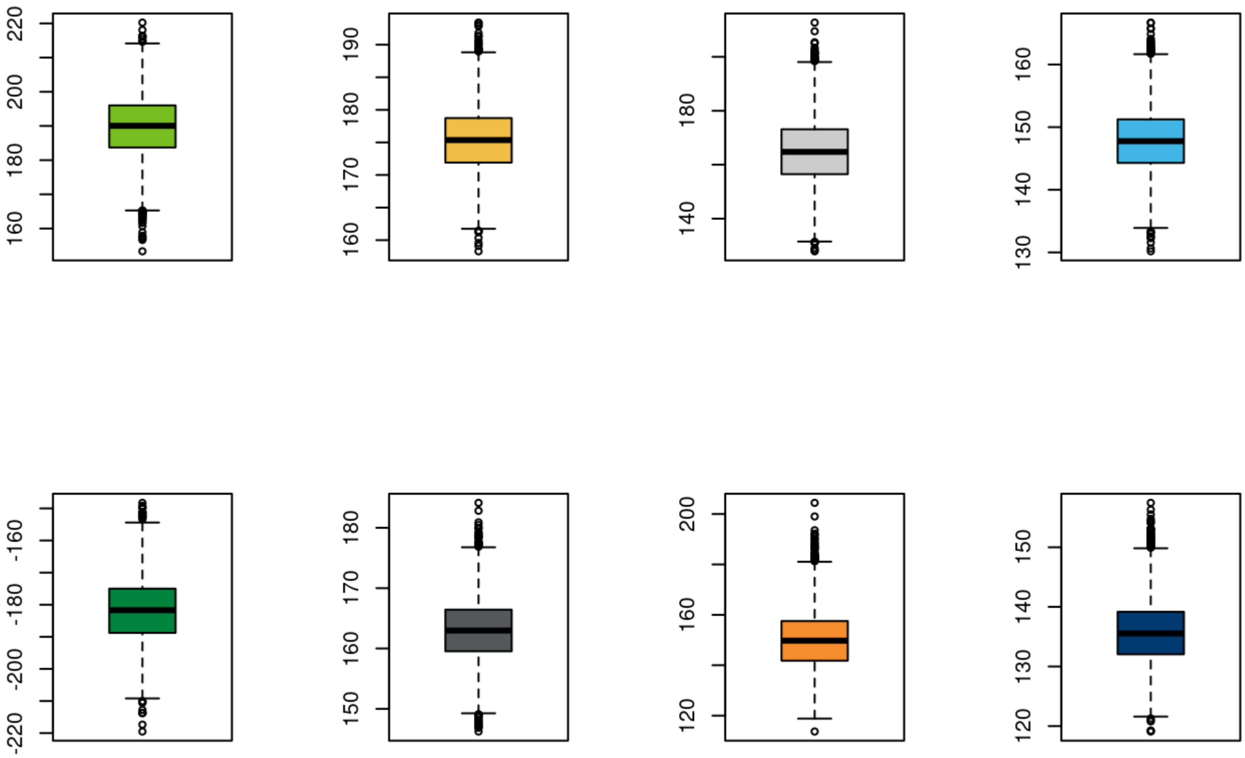
```
## [1] 135.7333
```

```
quantile(theta[,8], p=c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 125.9695 146.8195
```

```
par(mfrow = c(2, 4))


boxplot(theta[,1],  main="",
        xlab="", ylab="", col=(c("#78be20")))
boxplot(theta[,2],  main="",
        xlab="", ylab="",col=(c("#f1be48")) )
boxplot(theta[,3],  main="",
        xlab="", ylab="",col=(c("#CCCCCC")))
boxplot(theta[,4],  main="",
        xlab="", ylab="",col=(c("#41b6e6")))
boxplot(theta[,5],  main="",
        xlab="", ylab="",col=(c("#00843d")))
boxplot(theta[,6],  main="",
        xlab="", ylab="",col=(c("#545859")))
boxplot(theta[,7],  main="",
        xlab="", ylab="",col=(c("#f68d2e")))
boxplot(theta[,8],  main="",
        xlab="", ylab="",col=(c("#003A70")))
```



220
200
180
160
190
180
170
160
180
140
160
150
140
130
-160
-180
-200
-220
180
170
160
150
200
160
120
150
140
130
120