Homework 3

Michael Pena

2024-07-29

Analyis: Problem 1

We aimed to find marginal confidence intervals for scores from a math class of 62 students, covering midterm 1, midterm 2, and the final exam. Our dataset contained missing information (NAs), so we used the Expectation-Maximization (EM) algorithm to handle this issue. The EM algorithm helped us estimate missing values and update parameter estimates iteratively. While this allowed us to proceed with our analysis, the resulting estimates and confidence intervals are not perfect due to the missing data. We computed the marginal confidence intervals for midterm 1, midterm 2, and the final exam scores. The table below summarizes our findings:

var	lower	upper
exam1	73.06095	77.64294
exam2	77.38377	80.00883
final	76.88651	79.40906

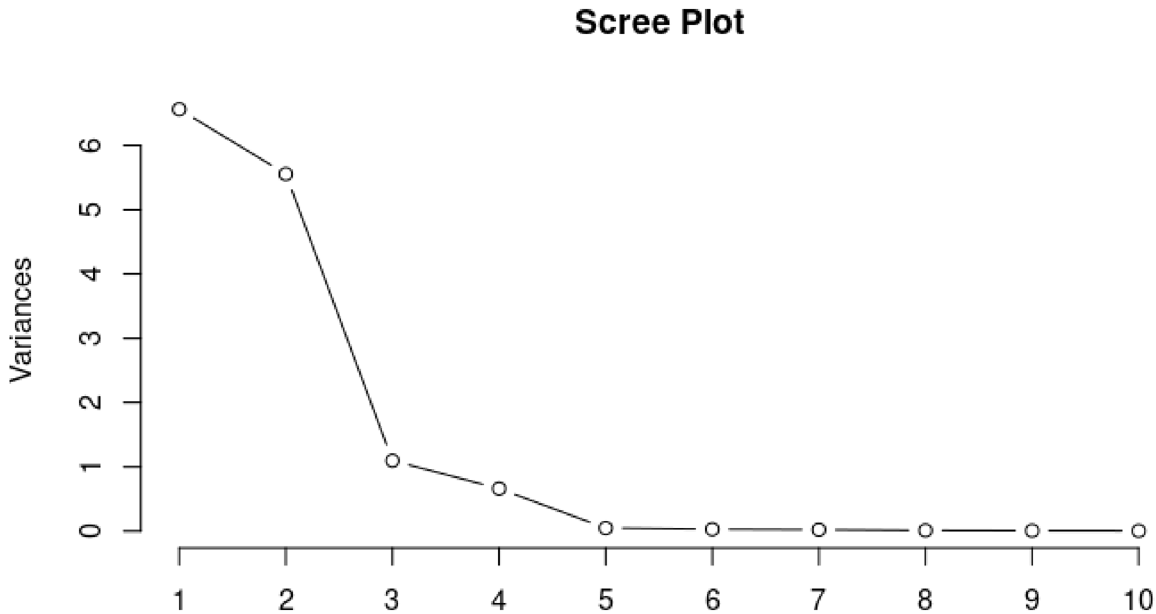
These confidence intervals are based on imputed data and should be seen as suggestive. The missing data introduces uncertainty, highlighting the importance of appropriate handling and cautious interpretation.

Analyis: Problem 2

In this analysis, I am working with graphical image data collected over several days from people driving. The data captures various aspects of facial positioning, including eye position, nose position, mouth movements, general head position, and gaze direction. Our objective is to build a predictive model that determines the direction a person's face is looking based on these variables. I will refer to this prediction as "gaze direction."

Variable Reduction with PCA

Given the dataset comprises 14 different variables, it was crucial to reduce the number of variables to simplify the model and improve interpretability. We employed Principal Component Analysis (PCA) for this purpose. PCA is a dimensionality reduction technique that transforms the original variables into a smaller set of uncorrelated components.



The scree plot reveals that the first few components explain the majority of the variance, suggesting that 4 components might be sufficient for our model. To confirm this, we evaluated the R-squared values for models using 4, 5, and 6 components. I found that using 6 components provided an R-squared value of 0.8096953 which was close to 0.846029 which was the R-squared value from using all 14 variables. Thus, selecting 6 components offers a balance between model simplicity and predictive accuracy, reducing the number of variables by more than half.

No. of Components	R-squared Value
4	0.7561574
5	0.7873039
6	0.8096953
14	0.846029

Examining the loading coefficients of the principal components provides insights into how each original variable influences the components. The analysis showed:

Components 1 and 2: These components did not show significant influences from the variables, indicating that they might not capture substantial patterns related to gaze direction. Component 3: This component was strongly influenced by variables related to the height and width of the face, suggesting it captures significant aspects of facial geometry. Component 4: This component focused primarily on the height of the face, highlighting its importance in gaze direction prediction. Component 5: This component was influenced by the X-position of the face and the X-position of the right corner of the mouth, indicating its role in capturing side-to-side facial positioning.

Regarding Factor Analysis

Factor analysis is another dimensionality reduction technique that helps in clustering similar variables together. While it doesn't directly provide a predictive model, it is valuable for exploratory data analysis. Factor analysis revealed that:

Variables related to the Y-position (up and down) of body parts tend to group together. Variables related to the X-position (side to side) of body parts also cluster. The height of the face forms its own distinct factor. These groupings help us understand how different facial features and positions relate to each other and can guide further analysis or model refinement.

	Factor1	Factor2	Factor3
xF		0.923	0.234
yF	0.994		
wF		-0.337	0.173
hF		0.306	0.910
xRE		0.992	
yRE	0.996		
xLE		0.977	0.174
yLE	0.997		
xN		0.992	
yN	0.996		
xRM		0.976	
yRM	0.997		
xLM	-0.186	0.948	0.202
yLM	0.998		

Code: Problem 1

let's look at dimensions

```
X <- as.matrix(data1)
dim(X)
```

```
## [1] 150   3
```

```
head(X)
```

```
##      exam1 exam2 final
## [1,]    85    86    85
## [2,]    67    77    73
## [3,]    61    61    70
## [4,]    74    82    79
## [5,]    69    82    83
## [6,]    51    66    59
```

Things are missing but we have an EM Algorithm to use from a past assignment.

```
# to.theta function
to.theta <- function(mu,Sig){
  theta = c(0)
  p = length(Sig[,1])
  q = p*(p+1)/2
  v = matrix(c(1,1,2,1,2,2,3,1,3,2,3,3),ncol = 2, nrow = 6, byrow = T)
  for(i in 1:p){theta[i] = mu[i]}
  for(i in 1:q){theta[p+i] = Sig[v[i,1],v[i,2]]}
  return(theta)
}

# EM algorithm
 EMfunc <- function(y,mu,Sig,tolgrad){
  # initials
  p = length(Sig[,1])
  n = length(y[,1])
  th = to.theta(mu,Sig)
  ip = norm(th, type='2')
  it = 1
  # save iterations in here
  ITER = matrix(c(it,mu[1],mu[3],Sig[1,1],Sig[3,3],ip), nrow =1, ncol = 6, byrow = T)
    while(ip >= tolgrad){
      xbar.star = c(0,0,0)
      S.star = matrix(0,p,p)

      for(i in 1:n){

        # noting missing and observed data
        obs = which(!is.na(y[i,]))
        mis = which(is.na(y[i,]))

        # use that info to get mus, sigmas, and data
        mu_o = mu[obs]
        mu_m = mu[mis]
        Sig_oo = Sig[obs,obs]
        Sig_om = Sig[obs,mis]
```

```r
        Sig_mo = Sig[mis,obs]
        Sig_mm = Sig[mis,mis]
        y_o = y[i,obs]
        y_m = y[i,mis]

        # initializing expectations for the xbar and S
        E.xi = c(0)
        E.S = matrix(0,p,p)

        # get mu tilde
        Estar.y_m = mu_m + (Sig_mo %*% solve(Sig_oo)) %*% (y_o - mu_o)
        E.xi[obs] = y_o
        E.xi[mis] = Estar.y_m
        xbar.star = xbar.star + E.xi/n
        mu.tilde = xbar.star

        # get sigma tilde
        E.S[obs,obs] = y_o %*% t(y_o)
        E.S[mis,obs] = Estar.y_m %*% t(y_o)
        E.S[obs,mis] = Estar.y_m %*% t(y_o)
        E.S[mis,mis] = Sig_mm - (Sig_mo %*% solve(Sig_oo) %*% Sig_om) + (Estar.y_m %*% t(Estar.y_m))
        S.star = S.star + E.S/n
        Sig.tilde = S.star - (xbar.star %*% t(xbar.star))
    }

    # finding the gradients
    del.mu = (-1)*n*solve(Sig.tilde) %*% (mu - mu.tilde)
    J = S.star - xbar.star%*%t(mu.tilde) - mu.tilde%*%t(xbar.star) + mu.tilde%*%t(mu.tilde)
    I = diag(3)
    del.Sig = (-n/2) * solve(Sig.tilde)%*%(I - solve(Sig.tilde)%*%J)
    del.theta = to.theta(del.mu,del.Sig)

    # get the innerproduct of del.theta
    ip <- norm(del.theta, type = '2')

    # plug back in for iteration
    mu.tilde -> mu
    Sig.tilde -> Sig
    it = it + 1

    # save into dataframe
    ITER <- rbind(ITER,c(it,mu[1],mu[3],Sig[1,1],Sig[3,3],ip))
    }
# print first three and last three rows of dataframe
u <- length(ITER[,1])
# colnames(ITER) = c("Iteration","mu1","mu2","Sigma_11","Sigma_33","gradnorm")
# print("first 3 iterations")
# print(ITER[1:3,])
# print("last 3 iterations")
# print(ITER[(u-2):u,])
# print final mu and Sigma
return(list("mu estimator" = mu, "Sigma estimator" = Sig))
 }
```

```r
EMfunc(X,c(0,0,0),diag(3),1e-06)
```

```
## $`mu estimator`
## [1] 75.35195 78.69630 78.14778
##
## $`Sigma estimator`
##           [,1]     [,2]     [,3]
## [1,] 197.57479 83.63317 87.83247
## [2,]  83.63317 64.84842 47.27422
## [3,]  87.83247 47.27422 59.88321
```

We can use these estimators to construct the confidence interval. Let's just save them.

```r
EMfunc(X,c(0,0,0),diag(3),1e-06) -> Est
S1 <- Est$`Sigma estimator`
m1 <- Est$`mu estimator`
```

Here we can present the confidence intervals. We can use fomrula from notes...

$$\bar{x}_j \pm \sqrt{\frac{p(n-1)s_j^2}{n(n-p)} \cdot F^{+}_{p,n-p}(\alpha)}$$

```r
confInt <- function(data,xbar,sigma,alpha = .05){
  X = data
  p = dim(X)[2]
  n = dim(X)[1]
  a = alpha
  s = sigma
  CONF <- data.frame()
  for(j in 1:p){
    num = p*(n-1)*(s[j,j])*qf(1-a,n-1,n-1)
    den = n*(n-p)
    CONF[j,1] = names(data)[j]
    CONF[j,2] = xbar[j] - sqrt(num/den)
    CONF[j,3] = xbar[j] + sqrt(num/den)
  }

  names(CONF) = c("var","upper","lower")
 return("confidence intervals" = CONF)
}
confInt(data1,m1,S1,.05)
```

```
##     var    upper    lower
## 1 exam1 73.06095 77.64294
## 2 exam2 77.38377 80.00883
## 3 final 76.88651 79.40906
```

The data given to us was something had had "holes" in it. This was was necessary to obtain estimators from an EM Algorithm.

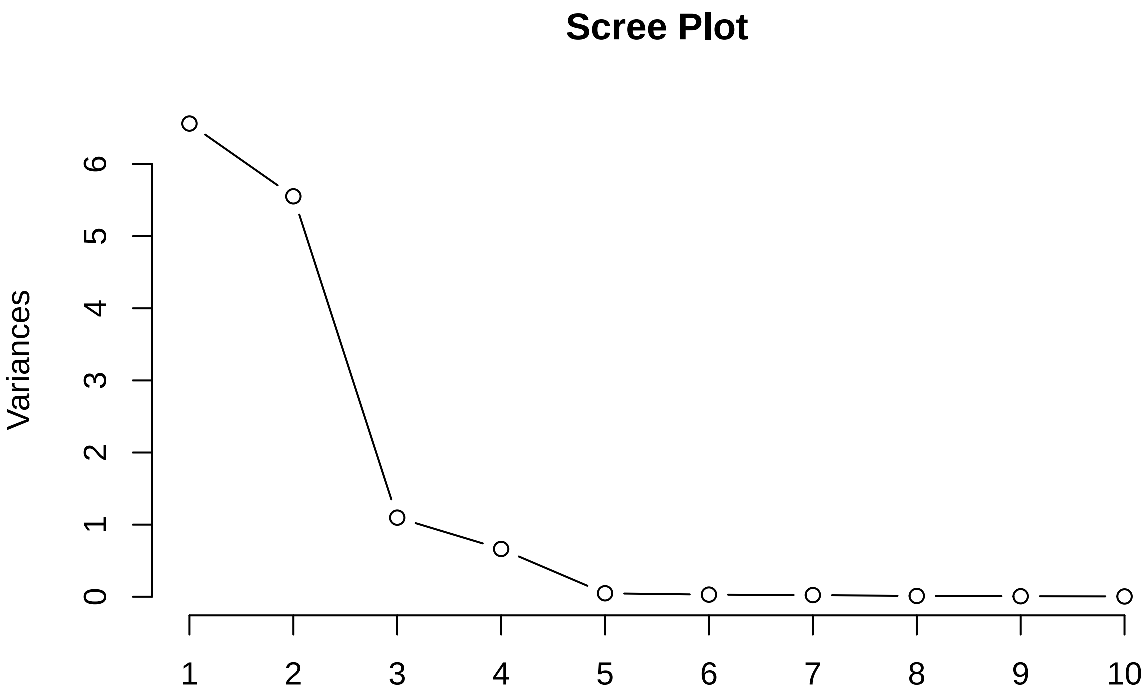
Code: Problem 1

(a).

```
# let's do some PCA work
# we need a GAMMA and LAMBDA
X = as.matrix(data2[,6:19])
Y = as.matrix(data2[,5])
S = cov((data2)[,6:19])
GAM = as.matrix(eigen(S)$vectors)
LAM = eigen(S)$values*(diag(14))
# generate the C columns
C = X%*%GAM
```

It makes much more sense using the pca() function

```
pca <- prcomp(X,scale. = T)
plot(pca, type = 'l', main = "Scree Plot")
```



four components seems to be the ball park.

```
model.all <- lm(Y ~ C)
model.4 <- lm(Y ~ C[,1:4])
model.5 <- lm(Y ~ C[,1:5])
model.6 <- lm(Y ~ C[,1:6])
summary(model.all)$r.squared
```

```
## [1] 0.846029
```

```
summary(model.4)$r.squared
```

```
## [1] 0.7561574
```

```
summary(model.5)$r.squared
```

```
## [1] 0.7873039
```

```r
summary(model.6)$r.squared
```

```
## [1] 0.8096953
```

4 was in the ball park but it would be safer to choose 6 which still eliminates more that half of the components. Let us think about the loading coefficients.

```r
names(pca)
```

```
## [1] "sdev"     "rotation" "center"   "scale"    "x"
```

```r
pca$rotation # these are the loading coefficients
```

```
##            PC1         PC2          PC3          PC4           PC5         PC6
## xF   0.2942435 -0.26070636  0.078744995  0.188899657  0.5119842366  0.35436373
## yF  -0.2794330 -0.29479659  0.046130612 -0.032777586  0.0160763124  0.02506571
## wF  -0.1206052  0.08913905 -0.766422460 -0.568148900  0.1769641420  0.08260637
## hF   0.1585780 -0.10946051 -0.628050184  0.709415013 -0.1994937662 -0.13075896
## xRE  0.2738470 -0.29522750  0.022563377 -0.144893240  0.0251673672 -0.41501621
## yRE -0.2831645 -0.29171738  0.001006468  0.008940596 -0.0115587688 -0.01468576
## xLE  0.2881683 -0.28016465 -0.034533147 -0.078343843  0.3228804485 -0.25441423
## yLE -0.2705769 -0.30511584  0.016749591 -0.011283653 -0.0001209351 -0.18340946
## xN   0.2832853 -0.27525736  0.020665047 -0.246382359 -0.2643976742 -0.40835880
## yN  -0.2768979 -0.29728337 -0.025268870  0.046098948  0.1253570389 -0.06722716
## xRM  0.2788090 -0.28061881 -0.002109732 -0.192831770 -0.6506842274  0.45816308
## yRM -0.2722651 -0.30239743 -0.037238484  0.043792537 -0.0477825779  0.22173192
## xLM  0.3280473 -0.21838995 -0.069718427 -0.082286981  0.2213522507  0.37256407
## yLM -0.2604178 -0.31515751 -0.022807691  0.033737931 -0.0345014284  0.09948744
##             PC7         PC8         PC9        PC10         PC11         PC12
## xF  -0.57436975 -0.16377001  0.21541870 -0.01005612 -0.072638805  0.077754299
## yF   0.05148575  0.02075672  0.12057237 -0.24272488 -0.573827434 -0.638111740
## wF  -0.16418803 -0.02390142  0.02362229 -0.01594534 -0.024164085  0.013045408
## hF   0.03463997 -0.01878499  0.01238614 -0.01943229 -0.070403955 -0.033941825
## xRE -0.19757393 -0.23434198 -0.61506353 -0.36630246  0.116981475 -0.114942861
## yRE  0.02698640 -0.12088418 -0.01155335  0.11836274 -0.248339420  0.339287342
## xLE  0.15672803  0.76832736  0.04138121  0.17713452 -0.001697374 -0.037744312
## yLE  0.00260230  0.02817565 -0.07407274 -0.02693086 -0.329994699  0.578178499
## xN   0.02543036 -0.31167991  0.65548571  0.08838119  0.100064052 -0.007961566
## yN  -0.01891991 -0.21322545 -0.22851070  0.73631401  0.250902566 -0.263559017
## xRM -0.20749259  0.21433647 -0.18747617  0.18588292 -0.111793962  0.011668398
## yRM  0.04886122  0.06846813  0.13346911 -0.22804086  0.450366195 -0.150740884
## xLM  0.72679783 -0.30218757 -0.10233115 -0.04304729 -0.045300167  0.076845146
## yLM  0.01662693  0.16315527  0.08601289 -0.34883346  0.430469428  0.152025623
##             PC13          PC14
## xF  -0.027901293 -0.0232250726
## yF  -0.132758303  0.0172932638
## wF  -0.008456326 -0.0022558845
## hF  -0.008101635  0.0009048581
## xRE  0.096097370  0.0301664358
## yRE  0.522623052  0.5957188421
## xLE  0.103311245  0.0455819628
## yLE -0.215996703 -0.5502648503
## xN  -0.026967038 -0.0110100763
## yN  -0.192856748 -0.0463137723
## xRM -0.040245290 -0.0089404002
## yRM  0.547766490 -0.4209301456
```

```
## xLM -0.068871522 -0.0260861174
## yLM -0.545371628  0.3978526949
```

```r
summary(model.6)
```

```
##
## Call:
## lm(formula = Y ~ C[, 1:6])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -26.7128  -4.1598  -0.2199   3.9353  17.3082
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.361e+02  9.537e+00 -14.270  < 2e-16 ***
## C[, 1:6]1    8.296e-03  2.631e-03   3.153   0.0017 **
## C[, 1:6]2    2.226e-01  5.206e-03  42.755  < 2e-16 ***
## C[, 1:6]3   -4.882e-01  2.652e-02 -18.412  < 2e-16 ***
## C[, 1:6]4    4.367e-01  3.064e-02  14.252  < 2e-16 ***
## C[, 1:6]5    6.007e-01  6.067e-02   9.901  < 2e-16 ***
## C[, 1:6]6    6.163e-01  7.341e-02   8.395 3.37e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.277 on 599 degrees of freedom
## Multiple R-squared:  0.8097, Adjusted R-squared:  0.8078
## F-statistic: 424.8 on 6 and 599 DF,  p-value: < 2.2e-16
```

(b).

```r
fact.model = factanal(X,factors=3,rotation="varimax")
fact.model$loadings
```

```
##
## Loadings:
##     Factor1 Factor2 Factor3
## xF           0.923   0.234
## yF   0.994
## wF          -0.337   0.173
## hF           0.306   0.910
## xRE          0.992
## yRE  0.996
## xLE          0.977   0.174
## yLE  0.997
## xN           0.992
## yN   0.996
## xRM          0.976
## yRM  0.997
## xLM -0.186   0.948   0.202
## yLM  0.998
##
##                Factor1 Factor2 Factor3
## SS loadings      6.005   5.844   1.011
## Proportion Var   0.429   0.417   0.072
```

```
## Cumulative Var   0.429   0.846   0.919
```

With factor analysis we are clustering together variables that are alike. This information does not really give us a model to make any inference. However, this tool may be useful in exploratory data analysis.