Universidade de Brasília

DEPARTAMENTO DE ESTATÍSTICA

27 abril 2023

Entrega 2 - Lista 3

Prof. Dr. George von Borries

Análise Multivariada 1

Aluno: Bruno Gondim Toledo | Matrícula: 15/0167636

4. Exercício 30 da Lista 3 - Utilize a decomposição espectral $\mathbf{\Sigma} = \mathbf{UDV^T}$ para mostrar que $\sum_{i=1}^{p} Var(\mathbf{X}_i) = \sum_{i=1}^{p} \lambda_i$,em que λ_i são os elementos da matriz diagonal $\mathbf{D}$.

Nota 1: Inclua dois exemplos numéricos com este resultado no R.

Nota 2: Entregas com exemplos iguais serão desconsideradas.

Demonstração:

Como $\sum_{i=1}^{n} Var(\mathbf{X}) = tr(\Sigma)$, segue que $\sum_{i=1}^{n} Var(\mathbf{X}) = tr(\mathbf{UDU^T})$. Dado que $\mathbf{U^TU} = \mathbf{I}$ e $tr(\mathbf{UDU^T}) = tr(\mathbf{U^TUD})$, então é seguro afirmar que $\sum_{i=1}^{n} Var(\mathbf{X}) = tr(\mathbf{ID}) = tr(\mathbf{\Sigma})$. Como $tr(\mathbf{ID}) = tr(\mathbf{D})$, em que o traço de $\mathbf{D}$ é $\sum_{i=1}^{n} \lambda_i$, que são autovalores da matriz de covariâncias $\mathbf{\Sigma}$.

□

```r
# Questão 1

# Comprovando c/ exemplo numérico

set.seed(150167636)
matriz <- matrix(c(sample.int(n=9,replace=T)),
                 nrow = 3, byrow = TRUE)

cov <- cov(matriz)
eig <- eigen(cov)

svar <- sum(diag(cov))
seig <- sum(eig$values)

round(svar) == round(seig)
```

```
## [1] TRUE
```

```r
# E o resultado está comprovado para este caso particular.

# Aumentando agora a matriz e verificando se a igualdade permanece
set.seed(636761051)
matriz2 <- matrix(c(sample.int(n=100,replace=T)),
                  nrow = 10, byrow = TRUE)

cov2 <- cov(matriz2)
eig2 <- eigen(cov2)

svar2 <- sum(diag(cov2))
seig2 <- sum(eig2$values)

round(svar2) == round(seig2)
```

```
## [1] TRUE
```

```r
# O que fortalece a hipótese do resultado.
```

Perceba que foi necessário "arredondar" os resultados para a igualdade ser verdadeira. Isto se deve ao fato de o R utilizar algum tipo de truncamento durante os cálculos. Porém, não fosse este fato e ele realmente utilizasse todas as casas decimais, tal operação não seria necessária visto que as respostas seriam exatamente iguais e a comparação retornaria *TRUE* sem necessidade de arredondamento.

5. Exercício 31 da Lista 3 Reproduza o estudo de redução de dimensão SVD de imagens, utilizando duas imagens. A primeira com poucos detalhes (abstrata, por exemplo) e a segunda com vários detalhes. Justifique sua escolha de dimensão na redução de cada imagem e compare os resultados. Você achou a redução compatível com as imagens utilizadas? Justifique.

Nota 1: Alguns pacotes de leitura de imagem podem ser mais simples de utilizar que o apresentado em aula.

Nota 2: Entregas com imagens e códigos iguais serão desconsideradas.

Exemplo abstrato:

Para este exemplo, utilizaremos um quadro do artista ítalo-brasileiro Alfredo Volpi, conhecido modernista que costumava representar bandeirinhas juninas em suas obras.

```
img <- readImage("rdocs/volpi.jpg")
dim(img)
imageShow(img,clear_viewer = T)
#writeImage(img, file_name = 'rdocs/img.png')
```

Figure 1: Imagem original

```
img_gray <- rgb_2gray(img)
dim(img_gray)
imageShow(img_gray,clear_viewer = T)
#writeImage(img_gray, file_name = 'rdocs/volpi_gray.png')
```

Aplicando SVD na imagem

```
img <- readImage("C:/Users/toled/Documents/Github/multivariada/rdocs/volpi.jpg")
img_gray <- rgb_2gray(img)

imgg.svd <- svd(img_gray)
head(imgg.svd$d,n=54)
```

```
##  [1] 229.859239  61.789964  43.948192  40.760642  36.012312  32.600281
##  [7]  28.910258  19.570303  17.656321  15.850726  13.146486  10.385024
## [13]   9.807348   9.657146   8.744394   8.207525   7.604042   7.317383
## [19]   7.130727   6.824580   6.525574   6.273402   5.997608   5.335333
## [25]   5.080195   4.894382   4.713462   4.251787   4.243742   4.174866
## [31]   3.926617   3.822811   3.637677   3.489109   3.243466   3.129861
```

Figure 2: Imagem cinza

```
## [37]   3.029129   2.969637   2.953226   2.702952   2.691267   2.622958
## [43]   2.489558   2.464112   2.437700   2.323649   2.244852   2.211070
## [49]   2.146597   2.129377   2.068951   2.051561   2.010150   1.926424
```

```
D <- diag(imgg.svd$d)
dim(D)
```

```
## [1] 358 358
```

```
U <- imgg.svd$u
V <- imgg.svd$v
```

Analisando a quantidade de informação por dimensão

```
png(file="C:/Users/toled/Documents/Github/multivariada/rdocs/plot4.png")
plot(1:length(imgg.svd$d), imgg.svd$d)
abline(v=c(5,10,20,40))
dev.off()
```

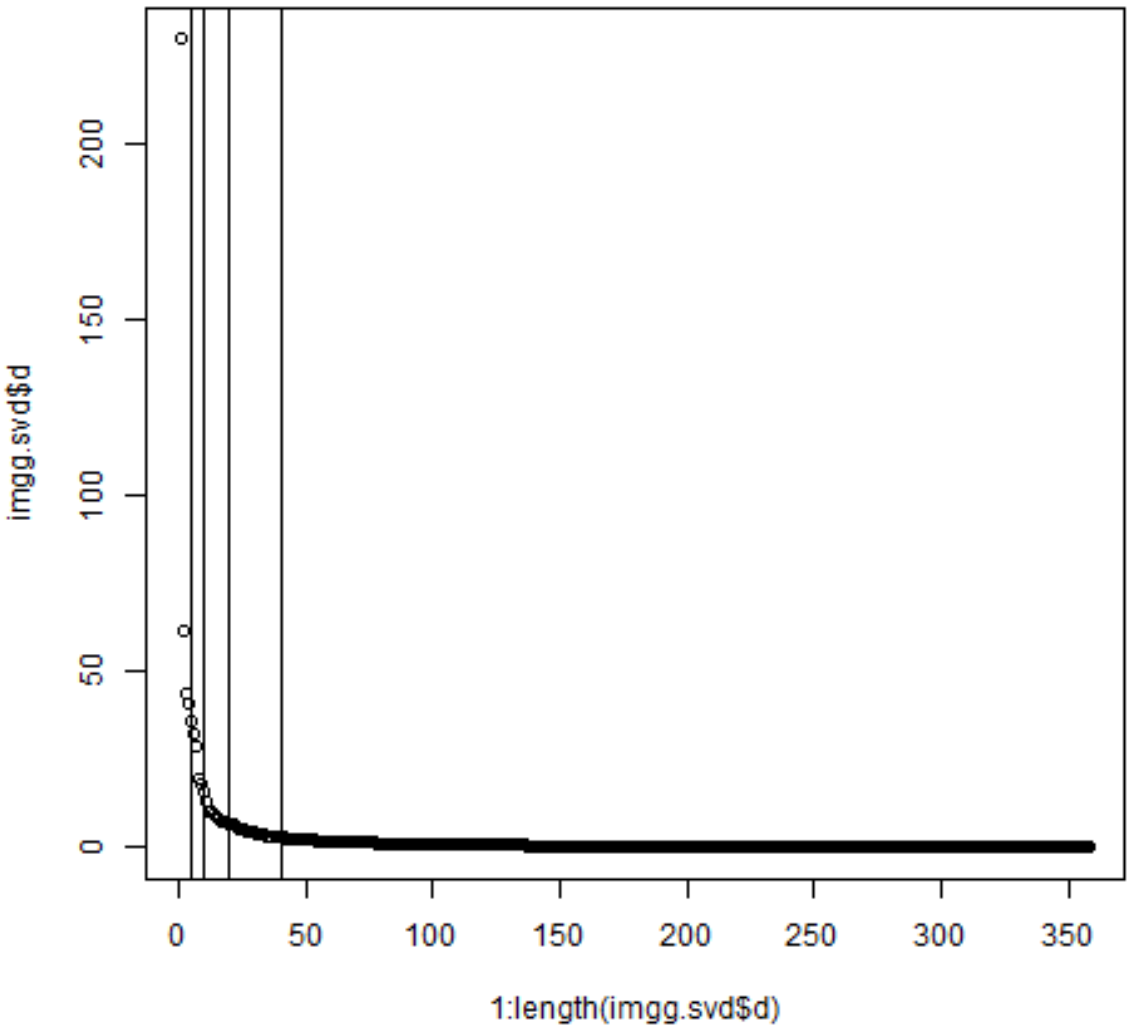
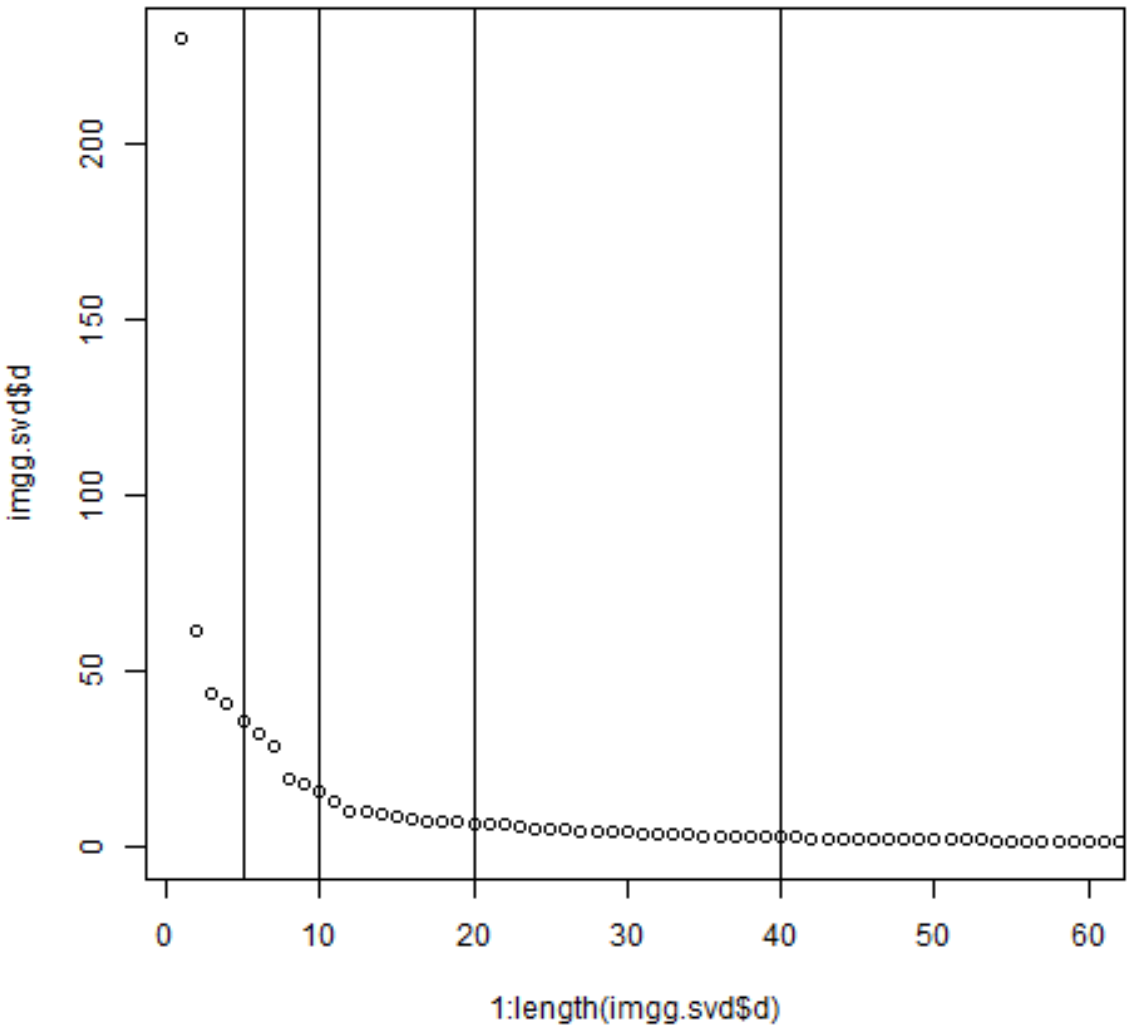
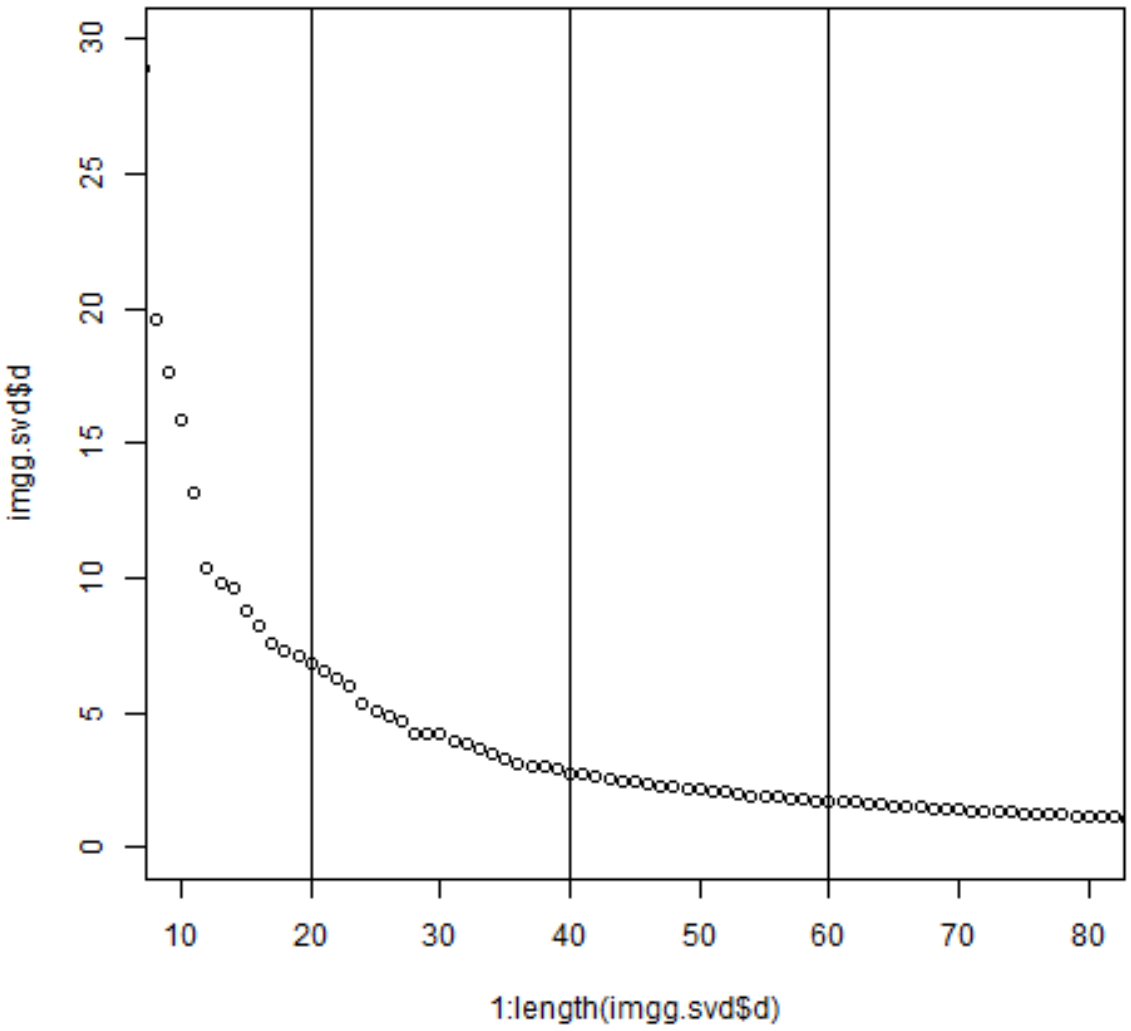


Figure 3: Gráfico 1

```
png(file="rdocs/plot5.png")
plot(1:length(imgg.svd$d), imgg.svd$d,xlim = c(1,60))
abline(v=c(5,10,20,40))
dev.off()
```



```
png(file="rdocs/plot6.png")
plot(1:length(imgg.svd$d), imgg.svd$d,xlim = c(10,80),ylim = c(0,30))
abline(v=c(20,40,60))
dev.off()
```



Reduzindo a dimensionalidade

```
U5 <- as.matrix(U[,1:5])
V5 <- as.matrix(V[,1:5])
D5 <- diag(imgg.svd$d[1:5])
img_gray5 <- U5 %*% D5 %*% t(V5)
tr(D5)/tr(D) * 100
```

```
## [1] 48.36148
```

```
imageShow(img_gray5,clear_viewer = T)
writeImage(img_gray5, file_name = 'rdocs/volpi_gray5.png')
```

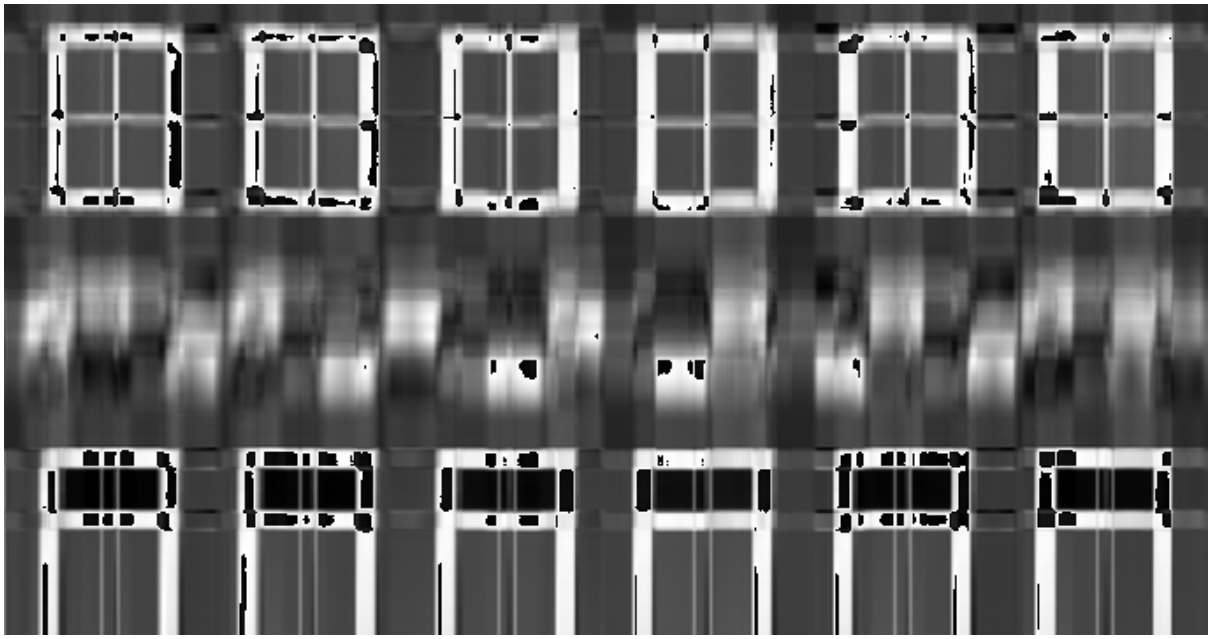


Figure 4: Redução para 5 dimensões

```
U10 <- as.matrix(U[,1:10])
V10 <- as.matrix(V[,1:10])
D10 <- diag(imgg.svd$d[1:10])
img_gray10 <- U10 %*% D10 %*% t(V10)
tr(D10)/tr(D) * 100
```

```
## [1] 61.79998
```

```
imageShow(img_gray10,clear_viewer = T)
writeImage(img_gray10, file_name = 'rdocs/img_gray10.png')
```

```
U20 <- as.matrix(U[,1:20])
V20 <- as.matrix(V[,1:20])
D20 <- diag(imgg.svd$d[1:20])
img_gray20 <- U20 %*% D20 %*% t(V20)
tr(D20)/tr(D) * 100
```

```
## [1] 72.21705
```

```
imageShow(img_gray20,clear_viewer = T)
writeImage(img_gray20, file_name = 'rdocs/img_gray20.png')
```

```
U30 <- as.matrix(U[,1:30])
V30 <- as.matrix(V[,1:30])
D30 <- diag(imgg.svd$d[1:30])
img_gray30 <- U30 %*% D30 %*% t(V30)
```

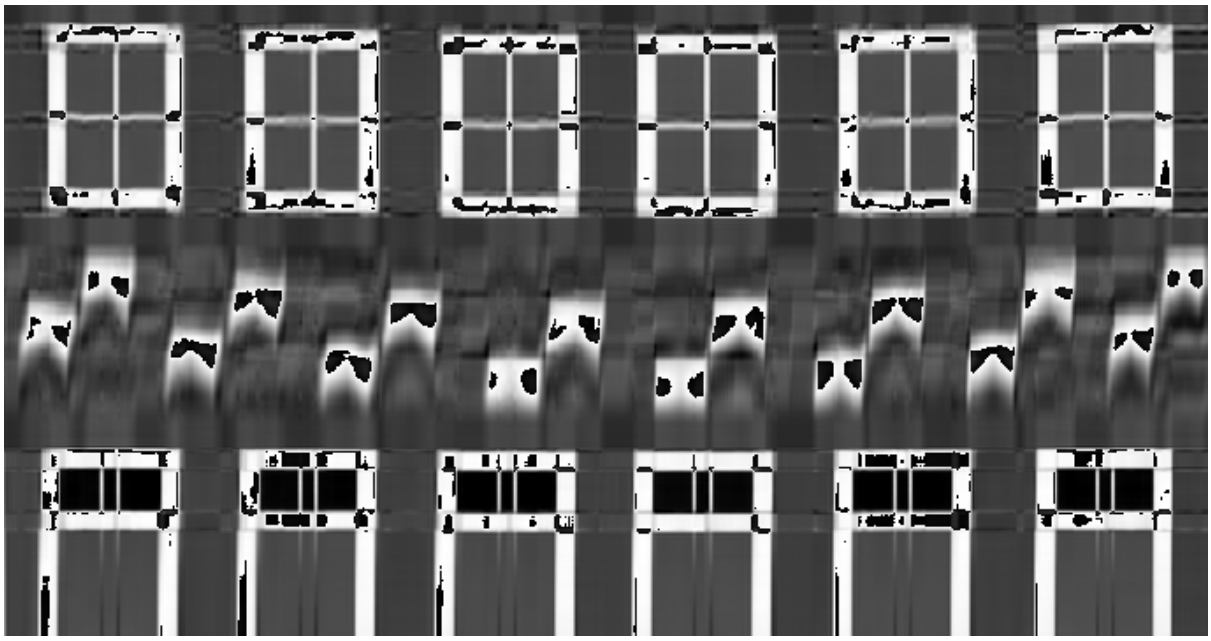


Figure 5: Redução para 10 dimensões

Figure 6: Redução para 20 dimensões

```
tr(D30)/tr(D) * 100
```

```
## [1] 78.25568
```

```
imageShow(img_gray30,clear_viewer = T)
writeImage(img_gray30, file_name = 'rdocs/img_gray30.png')
```

Figure 7: Redução para 30 dimensões

À esta altura, já temos praticamente quase toda a visualização da imagem.

```
U40 <- as.matrix(U[,1:40])
V40 <- as.matrix(V[,1:40])
D40 <- diag(imgg.svd$d[1:40])
img_gray40 <- U40 %*% D40 %*% t(V40)
tr(D40)/tr(D) * 100
```

```
## [1] 82.11461
```

```
imageShow(img_gray40,clear_viewer = T)
writeImage(img_gray40, file_name = 'rdocs/img_gray40.png')
```

```
U60 <- as.matrix(U[,1:60])
V60 <- as.matrix(V[,1:60])
D60 <- diag(imgg.svd$d[1:60])
img_gray60 <- U60 %*% D60 %*% t(V60)
tr(D60)/tr(D) * 100
```

```
## [1] 87.11002
```

```
imageShow(img_gray60,clear_viewer = T)
writeImage(img_gray60, file_name = 'rdocs/volpi_gray60.png')
```

À esta altura, é seguro dizer que já temos definição suficiente para identificar todos os elementos da figura, sem chance de confusão.

Figure 8: Redução para 40 dimensões

Figure 9: Redução para 60 dimensões

Passemos agora para uma imagem com mais detalhes, onde talvez tenhamos de ser mais conservadores quanto a remoção de dimensões para preservar algumas características essenciais.

```
img <- readImage("rdocs/maicon.jpeg")
dim(img)
imageShow(img,clear_viewer = T)
writeImage(img, file_name = 'rdocs/maicon.jpeg')
```

Para este exemplo, usaremos uma fotografia do colega felino de Judytt: o Maicon; gato da minha namorada.

Este é Maicon:

```
img_gray <- rgb_2gray(img)
dim(img_gray)
imageShow(img_gray,clear_viewer = T)
writeImage(img_gray, file_name = 'rdocs/maicon_gray.png')
```

Aplicando SVD na imagem

```
img <- readImage("C:/Users/toled/Documents/Github/multivariada/rdocs/maicon.jpeg")
img_gray <- rgb_2gray(img)

imgg.svd <- svd(img_gray)
head(imgg.svd$d,56)
```

```
##  [1] 662.989256 153.326512 107.579424  63.078489  53.431792  44.587135
##  [7]  34.337830  31.618072  24.895595  22.751035  22.053525  18.765881
## [13]  16.377156  16.039120  14.840268  13.981168  13.350981  12.844415
## [19]  12.324216  11.409597  11.073423  10.793739   9.882916   9.271503
## [25]   8.785698   8.548498   7.980265   7.783329   7.666519   7.280110
## [31]   7.115427   6.677539   6.227357   6.005592   5.704681   5.548171
## [37]   5.365680   5.316002   5.046989   4.925738   4.830321   4.696670
## [43]   4.588491   4.442644   4.378863   4.284502   4.245289   4.206912
## [49]   3.950227   3.896125   3.795058   3.704711   3.644400   3.620115
## [55]   3.552614   3.492629
```

```
D <- diag(imgg.svd$d)
dim(D)
```

```
## [1] 1200 1200
```

```
U <- imgg.svd$u
V <- imgg.svd$v
```

Figure 10: Imagem cinza

Analisando a quantidade de informação por dimensão

```
png(file="rdocs/plot1.png")
plot(1:length(imgg.svd$d), imgg.svd$d)
abline(v=c(5,10,20,40))
dev.off()
```

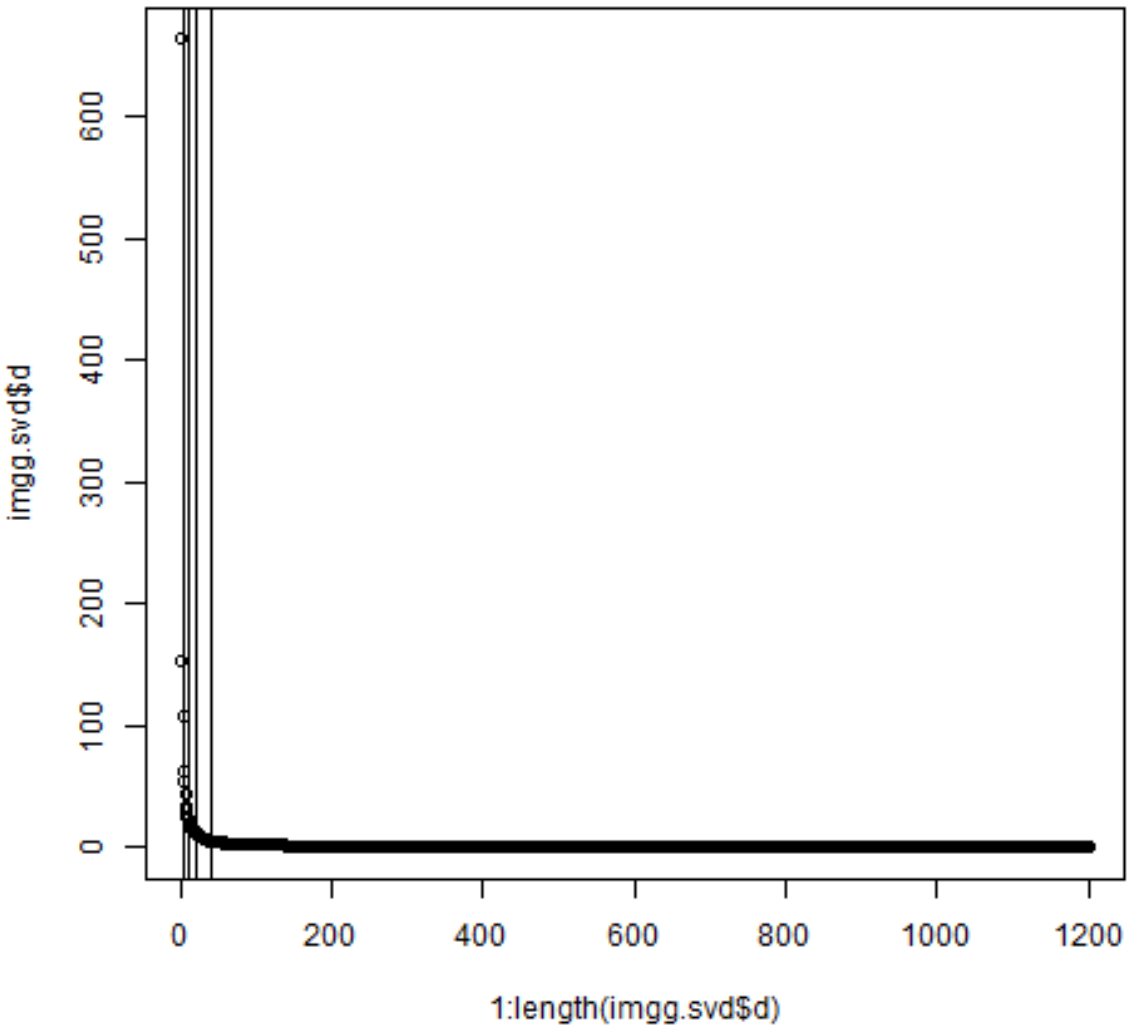
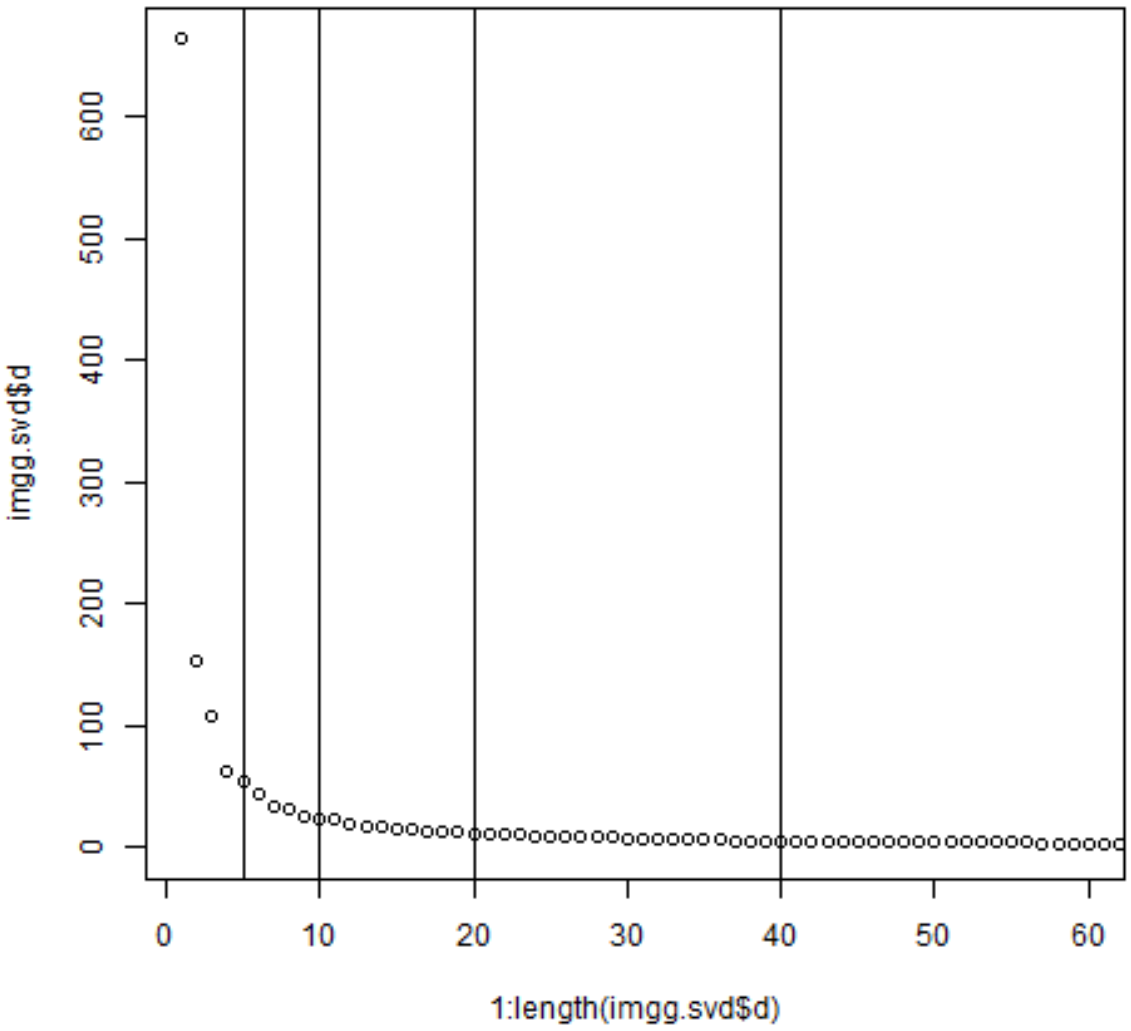
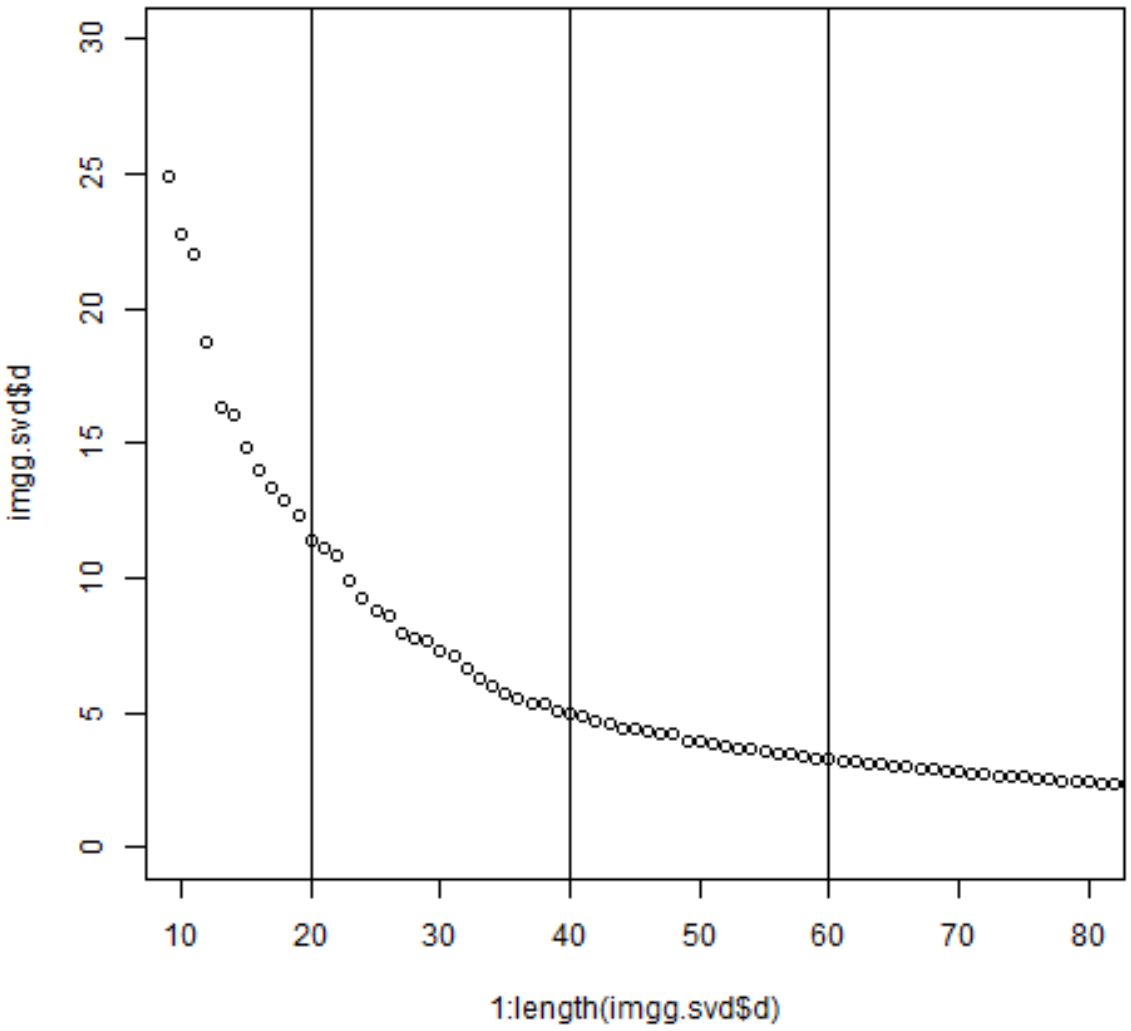


Figure 11: Gráfico 4

```
png(file="rdocs/plot2.png")
plot(1:length(imgg.svd$d), imgg.svd$d,xlim = c(1,60))
abline(v=c(5,10,20,40))
dev.off()
```



```
png(file="rdocs/plot3.png")
plot(1:length(imgg.svd$d), imgg.svd$d,xlim = c(10,80),ylim = c(0,30))
abline(v=c(20,40,60))
dev.off()
```



Reduzindo a dimensionalidade

```
U5 <- as.matrix(U[,1:5])
V5 <- as.matrix(V[,1:5])
D5 <- diag(imgg.svd$d[1:5])
img_gray5 <- U5 %*% D5 %*% t(V5)
tr(D5)/tr(D) * 100
```

```
## [1] 52.97195
```

```
imageShow(img_gray5,clear_viewer = T)
writeImage(img_gray5, file_name = 'rdocs/img_gray5.png')
```

```
U10 <- as.matrix(U[,1:10])
V10 <- as.matrix(V[,1:10])
D10 <- diag(imgg.svd$d[1:10])
img_gray10 <- U10 %*% D10 %*% t(V10)
tr(D10)/tr(D) * 100
```

```
## [1] 61.02613
```

```
imageShow(img_gray10,clear_viewer = T)
writeImage(img_gray10, file_name = 'rdocs/img_gray10.png')
```

```
U20 <- as.matrix(U[,1:20])
V20 <- as.matrix(V[,1:20])
D20 <- diag(imgg.svd$d[1:20])
img_gray20 <- U20 %*% D20 %*% t(V20)
tr(D20)/tr(D) * 100
```

```
## [1] 68.76448
```

```
imageShow(img_gray20,clear_viewer = T)
writeImage(img_gray20, file_name = 'rdocs/img_gray20.png')
```

A imagem ainda está bem ruim, mas já é possível identificar que é um gato, ao menos.

Precisaremos fazer um salto para ganhar alguma definição.

```
U60 <- as.matrix(U[,1:60])
V60 <- as.matrix(V[,1:60])
D60 <- diag(imgg.svd$d[1:60])
img_gray60 <- U60 %*% D60 %*% t(V60)
tr(D60)/tr(D) * 100
```

```
## [1] 80.25788
```

```
imageShow(img_gray60,clear_viewer = T)
writeImage(img_gray60, file_name = 'rdocs/img_gray60.png')
```

```
U99 <- as.matrix(U[,1:99])
V99 <- as.matrix(V[,1:99])
D99 <- diag(imgg.svd$d[1:99])
img_gray99 <- U99 %*% D99 %*% t(V99)
tr(D99)/tr(D) * 100
```

```
## [1] 85.20387
```

```
imageShow(img_gray99,clear_viewer = T)
writeImage(img_gray99, file_name = 'rdocs/img_gray99.png')
```

Conferindo o rank das matrizes, ou seja, vendo o quanto de informação conseguimos retirar e ainda reter uma definição razoável.

Figure 12: Redução para 5 dimensões

Figure 13: Redução para 10 dimensões

Figure 14: Redução para 20 dimensões

Figure 15: Redução para 60 dimensões

Figure 16: Redução para 99 dimensões

```
rankMatrix(img_gray)[1]
```

```
## [1] 1200
```

```
rankMatrix(img_gray99)[1]
```

```
## [1] 99
```

Referências:

- Código disponibilizado pelo prof. George von Borries no aprender3 (svdImagem.R)