Redução de Dimensionalidade

PCA

- Permite reduzir a dimensionalidade
- Permite eliminar a correlação entre as variáveis
- Permite visualizar dados com alta dimensionalidade
- Permite identificar semelhanças ou disparidades entre os dados
- Permite identificar que variáveis fazem com que um grupo seja diferente dos outros
- Variáveis fortemente correlacionadas irão contribuir fortemente para o mesmo componente principal
- Cada componente tenta explicar o máximo de variância possível
- Todos os componentes são ortogonais
- Permite comprimir os dados

Vantagens e desvantagens

Vantagens

- Fácil de calcular
- Aumenta a eficiência dos outros algoritmos
- Minora os efeitos de dados com alta dimensionalidade

Desvantagens

- É difícil de interpretar os componentes principais
- Perde-se informação com a redução de dimensionalidade
- É sensível à escala das variáveis
- Não é robusto a outliers
- Só funciona quando há relação linear entre as variáveis
- Não deteta relações não lineares

Procedimentos de instalação

```
library(devtools)
install_github("vqv/ggbiplot")
library(ggbiplot)
```

Resultado do PCA sobre o dataset iris

```
names(iris) = c("SL", "SW", "PL", "PW", "Species")
PCA = iris[,-5] %>% prcomp(scale = TRUE)
PCA
```

```
Standard deviations (1, .., p=4):
[1] 1.7083611 0.9560494 0.3830886 0.1439265

Rotation (n x k) = (4 x 4):
          PC1         PC2        PC3        PC4
SL  0.5210659 -0.37741762  0.7195664  0.2612863
SW -0.2693474 -0.92329566 -0.2443818 -0.1235096
PL  0.5804131 -0.02449161 -0.1421264 -0.8014492
PW  0.5648565 -0.06694199 -0.6342727  0.5235971
```

```
PCA %>% summary
```

```
Importance of components:
                          PC1    PC2     PC3     PC4
Standard deviation     1.7084 0.9560 0.38309 0.14393
Proportion of Variance 0.7296 0.2285 0.03669 0.00518
Cumulative Proportion  0.7296 0.9581 0.99482 1.00000
```

Correlação do dataset iris

```
iris[,-5] %>% cor %>% round(digits = 2)
```

```
      SL    SW    PL    PW
SL  1.00 -0.12  0.87  0.82
SW -0.12  1.00 -0.43 -0.37
PL  0.87 -0.43  1.00  0.96
PW  0.82 -0.37  0.96  1.00
```

```
PCA %>% .$x %>% cor %>% round(digits = 2)
```

```
    PC1 PC2 PC3 PC4
PC1   1   0   0   0
PC2   0   1   0   0
PC3   0   0   1   0
PC4   0   0   0   1
```

Reconstrução dos dados

```
iris[,-5] %>% scale %>% head
```

```
             SL          SW        PL        PW
[1,] -0.8976739  1.01560199 -1.335752 -1.311052
[2,] -1.1392005 -0.13153881 -1.335752 -1.311052
[3,] -1.3807271  0.32731751 -1.392399 -1.311052
[4,] -1.5014904  0.09788935 -1.279104 -1.311052
[5,] -1.0184372  1.24503015 -1.335752 -1.311052
[6,] -0.5353840  1.93331463 -1.165809 -1.048667
```

```
head(PCA$x %*% t(PCA$rotation))
```

```
             SL          SW        PL        PW
[1,] -0.8976739  1.01560199 -1.335752 -1.311052
[2,] -1.1392005 -0.13153881 -1.335752 -1.311052
[3,] -1.3807271  0.32731751 -1.392399 -1.311052
[4,] -1.5014904  0.09788935 -1.279104 -1.311052
[5,] -1.0184372  1.24503015 -1.335752 -1.311052
[6,] -0.5353840  1.93331463 -1.165809 -1.048667
```

Previsão

```
scale(iris[1:4,-5], center = PCA$center, scale = PCA$scale) %*% PCA$rotation
```

```
        PC1        PC2        PC3         PC4
1 -2.257141 -0.4784238  0.1272796  0.02408751
2 -2.074013  0.6718827  0.2338255  0.10266284
3 -2.356335  0.3407664 -0.0440539  0.02828231
4 -2.291707  0.5953999 -0.0909853 -0.06573534
```

```
predict(PCA, iris[1:4, -5])
```

```
        PC1        PC2        PC3         PC4
1 -2.257141 -0.4784238  0.1272796  0.02408751
2 -2.074013  0.6718827  0.2338255  0.10266284
3 -2.356335  0.3407664 -0.0440539  0.02828231
4 -2.291707  0.5953999 -0.0909853 -0.06573534
```

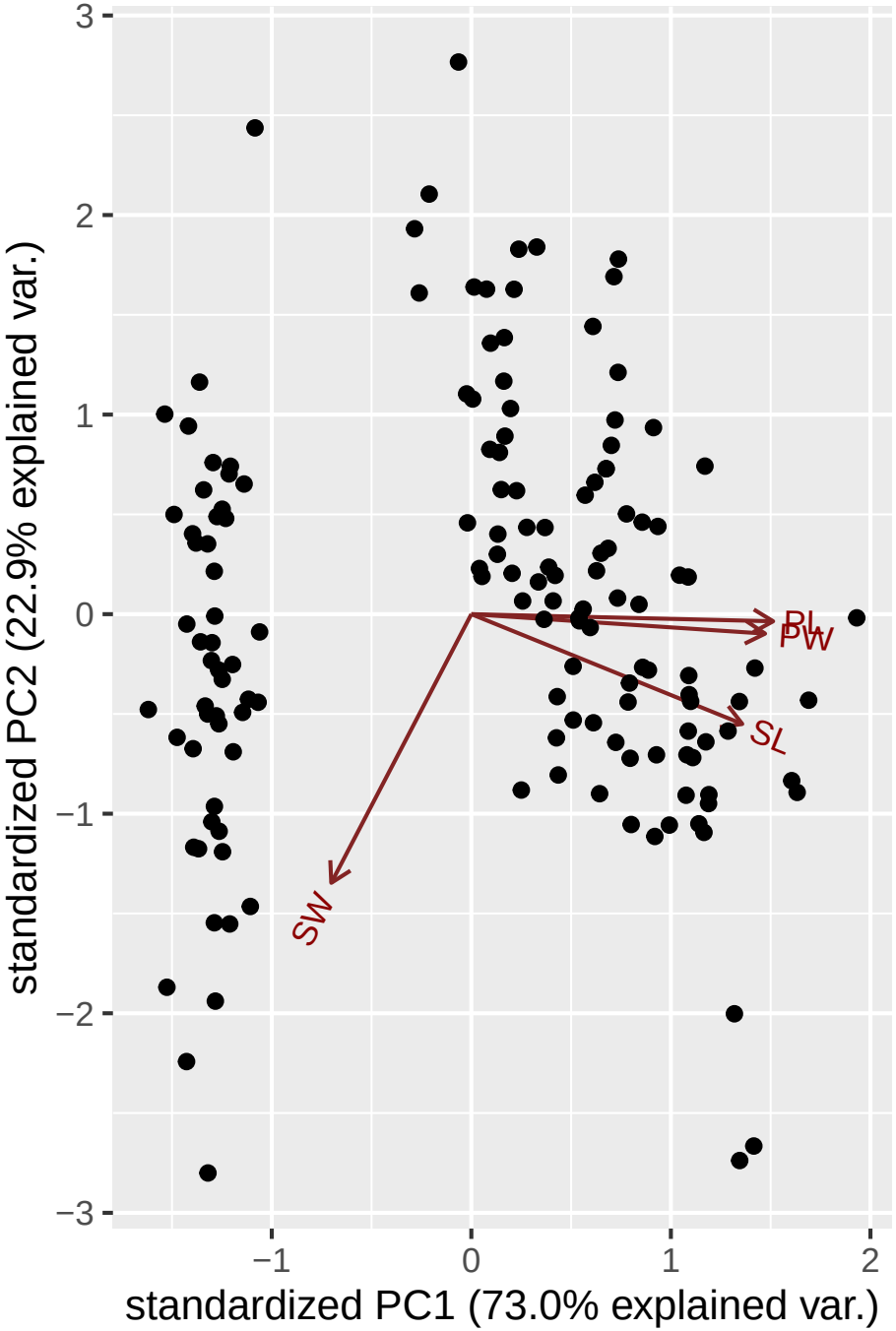
```
PCA$x[1:4,]
```

```
           PC1        PC2        PC3         PC4
[1,] -2.257141 -0.4784238  0.1272796  0.02408751
[2,] -2.074013  0.6718827  0.2338255  0.10266284
[3,] -2.356335  0.3407664 -0.0440539  0.02828231
[4,] -2.291707  0.5953999 -0.0909853 -0.06573534
```

Biplot

- Forma de apresentar resultados do PCA
- Pontos próximos correspondem a dados parecidos
- Eixos correspondem a variáveis
- Variáveis altamente correlacionadas correspondem a eixos com direções muito semelhantes
- Cada eixo é um vetor e a sua norma corresponde à sua contribuição para os componentes principais
- Direção de cada eixo corresponde a correlação positiva ou negativa

Biplot




```
PCA %>% ggbiplot
```

```
iris[,-5] %>% cor %>% round(digits = 2)
```

```
      SL    SW    PL    PW
SL  1.00 -0.12  0.87  0.82
SW -0.12  1.00 -0.43 -0.37
PL  0.87 -0.43  1.00  0.96
PW  0.82 -0.37  0.96  1.00
```

Comentários

- PL e PW estão altamente correlacionados positivamente com PC1
- SL também esta bastante correlacionado positivamente com PC1
- Podemos concluir que PL, PW e SL estão bastante correlacionados entre si
- SL está correlacionado positivamente com PC1 e negativamente com PC2
- SW está correlacionado negativamente com PC1 e PC2

Ordem das variáveis

```
ordem = PCA %>% .$rotation %>% .[,1] %>% order(decreasing = TRUE)
PCA %>% .$rotation %>% round(digits = 2) %>% .[ordem,]
```

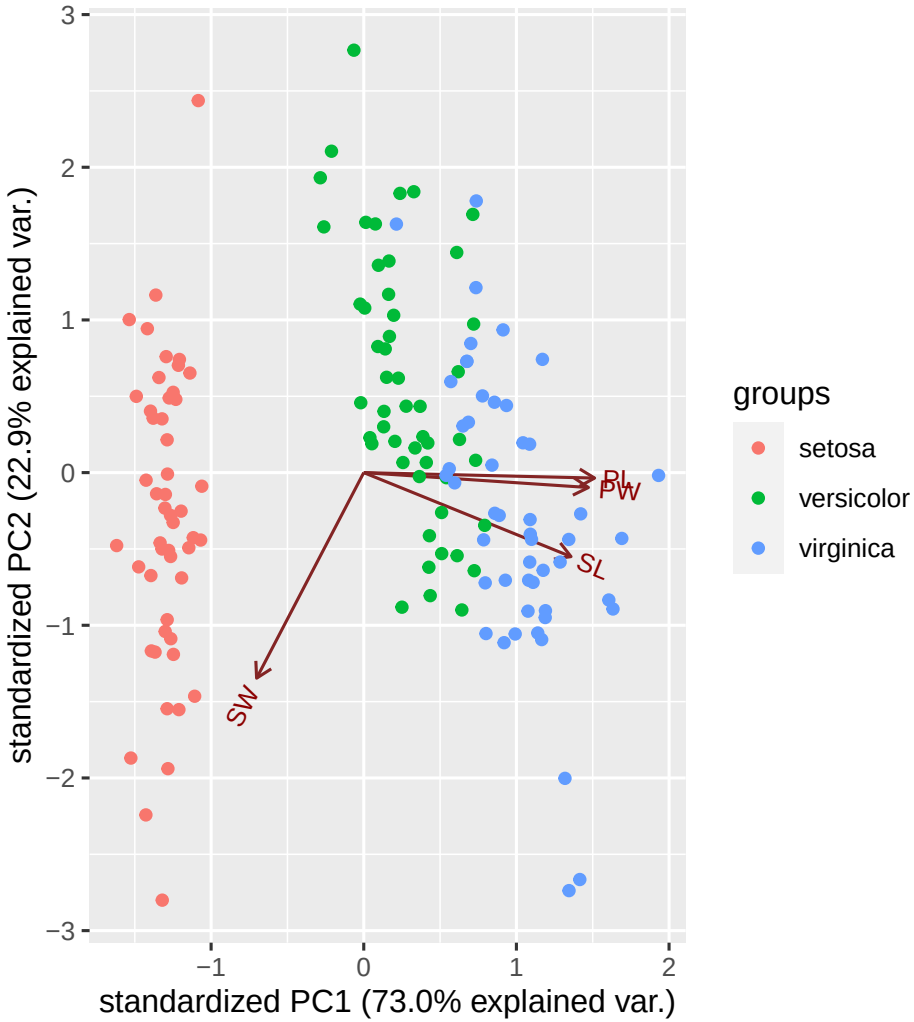
```
     PC1   PC2   PC3   PC4
PL  0.58 -0.02 -0.14 -0.80
PW  0.56 -0.07 -0.63  0.52
SL  0.52 -0.38  0.72  0.26
SW -0.27 -0.92 -0.24 -0.12
```

Biplot

Quando queremos perceber que exemplos estão relacionados

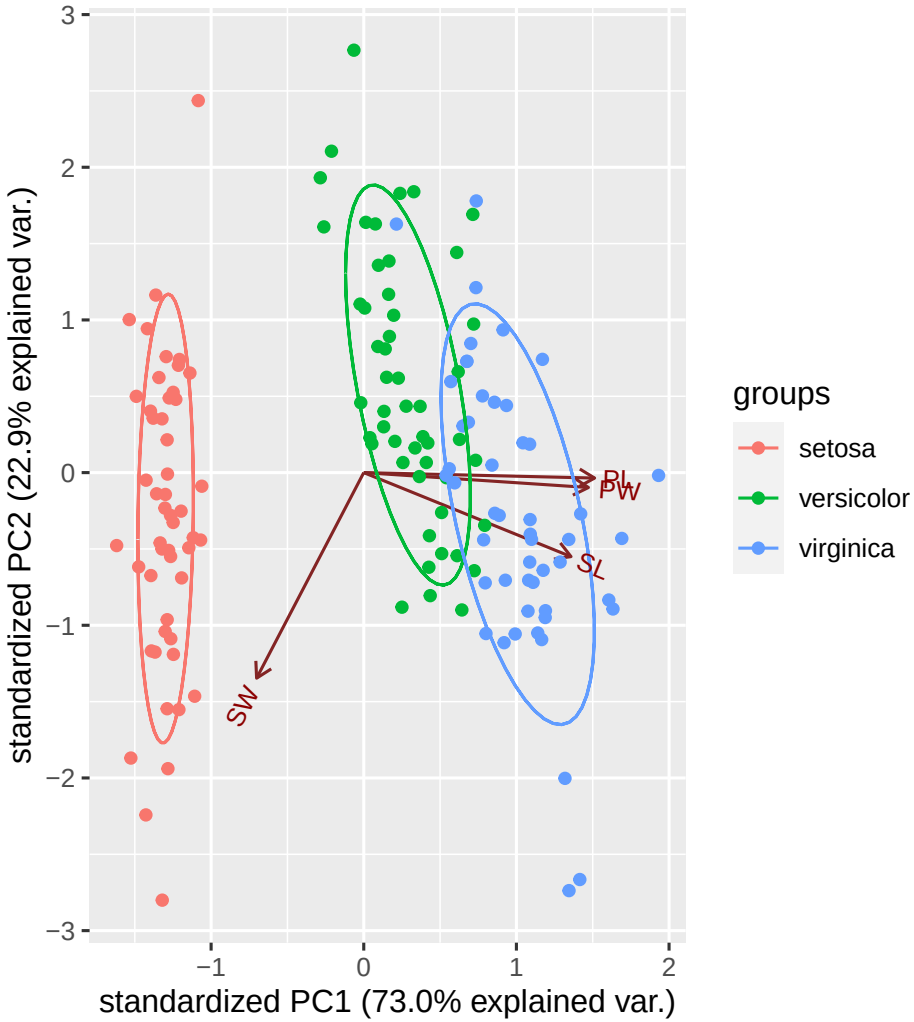
```
ggbiplot(PCA, labels = row.names(iris))
```

Biplot




```
ggbiplot(PCA,groups = iris$Species)
```

Biplots



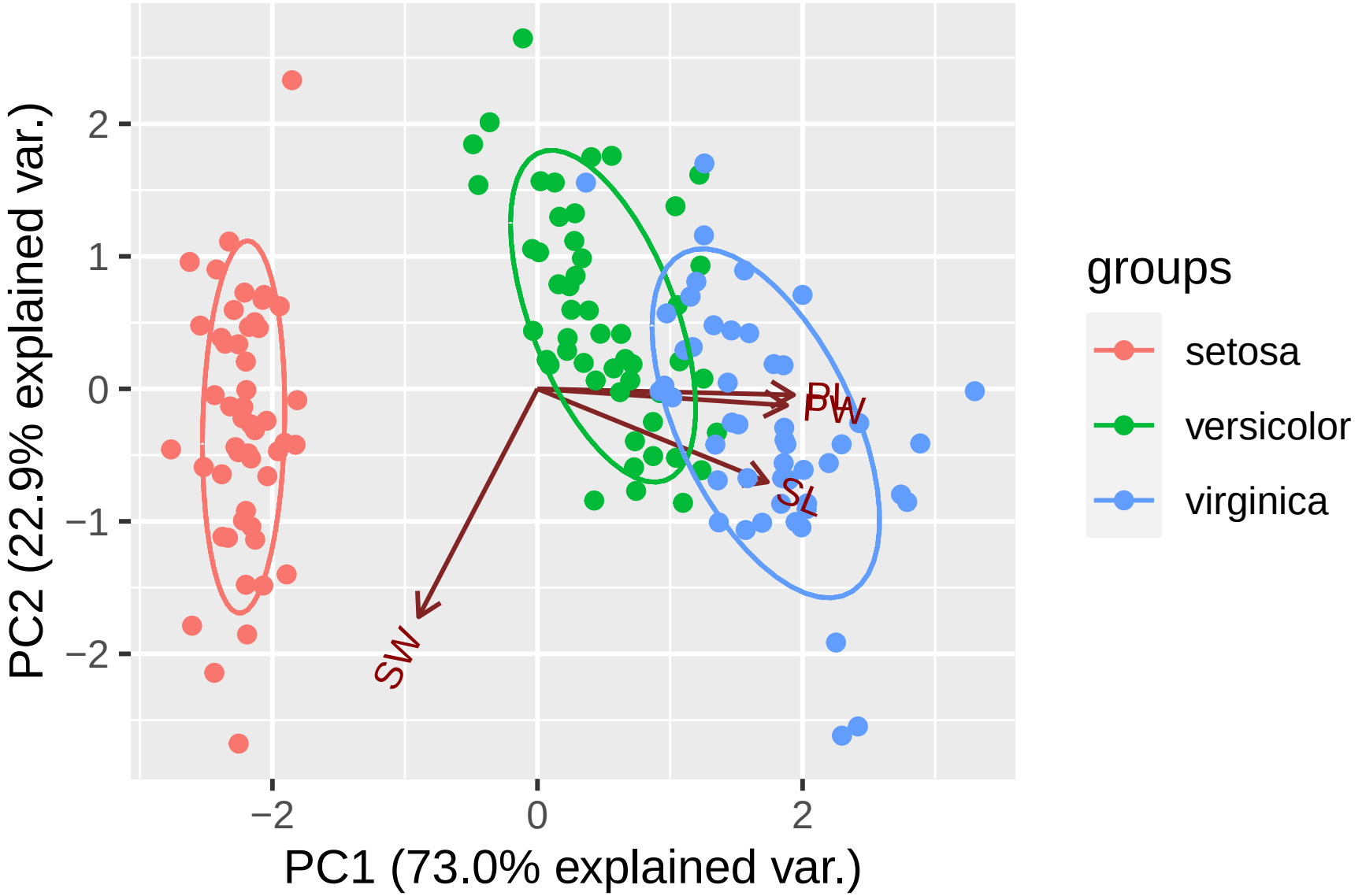

```
ggbiplot(PCA, groups=iris$Species,
  ellipse=T)
```

Conclusões

- Consegue-se separar setosa de virginica e versicolor usando PC1
- É mais complicado de separar virginica de versicolor
- virginica é caracterizado por PL, PW e SL

Mexendo a escala

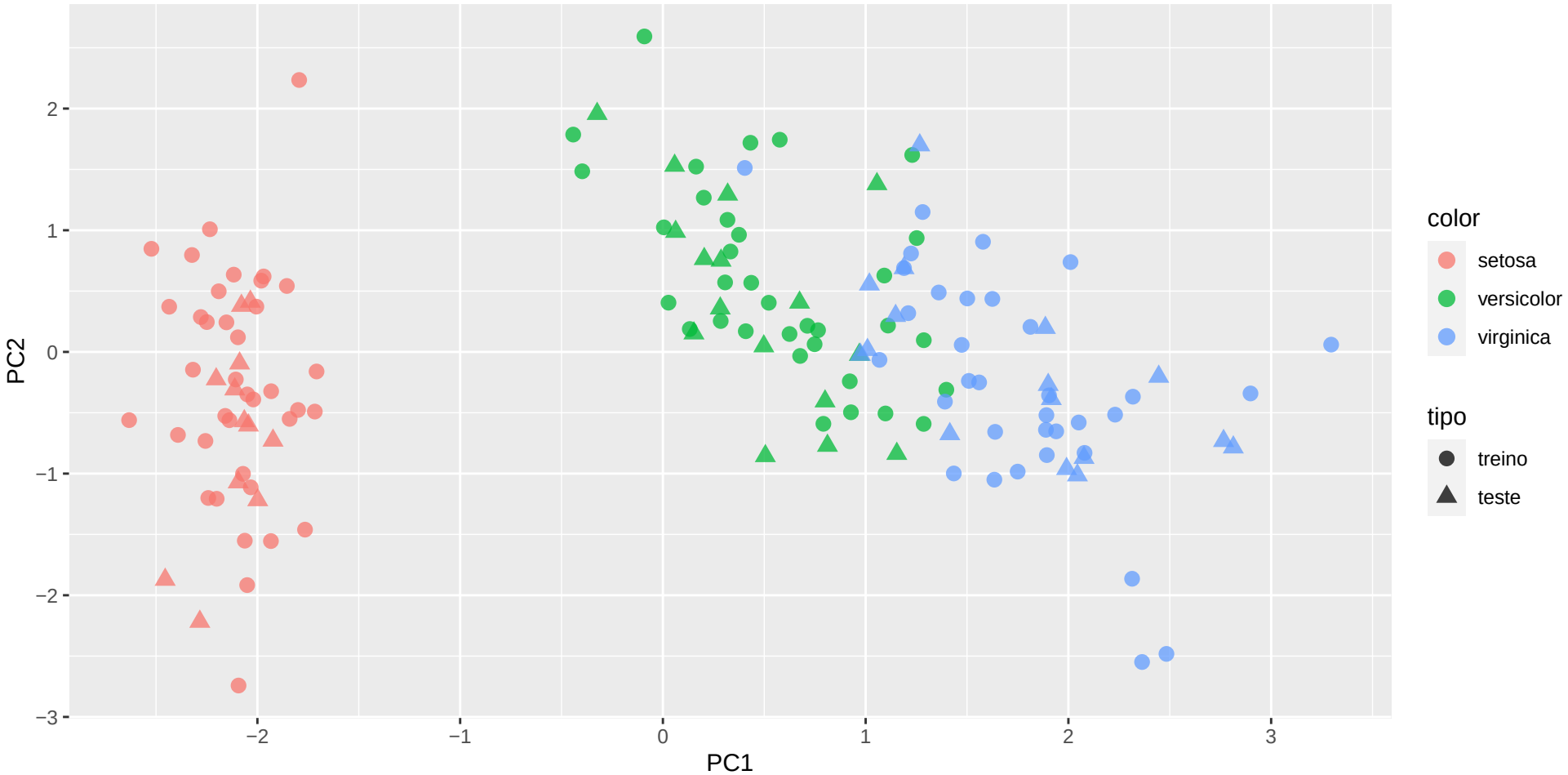
```
ggbiplot(PCA,groups=iris$Species,obs.scale=1,var.scale=1,ellipse=T)
```



Previsão revisitada

```
grupo = sample(1:2, nrow(iris), replace = TRUE, prob=c(0.75, 0.25))
treino = iris[grupo == 1, -5]
teste = iris[grupo == 2, -5]
PCA = prcomp(treino, scale = TRUE)
previsao = predict(PCA, teste)
df.treino = cbind(PCA$x, data.frame(color = iris[grupo == 1, 5], tipo = "treino"))
df.teste = cbind(previsao, data.frame(color = iris[grupo == 2, 5], tipo="teste"))
data=rbind(df.treino, df.teste)
ggplot(data, aes(x=PC1, y=PC2, color = color, shape = tipo)) +
    geom_point(alpha = 0.75, size = 3)
```

Previsão revisitada



mtcars

```
mtcars %>% summary
```

```
      mpg             cyl             disp             hp       
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
 Median :19.20   Median :6.000   Median :196.3   Median :123.0  
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
      drat             wt             qsec             vs        
 Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
 1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
 Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
 Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
 3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
 Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
       am              gear            carb      
 Min.   :0.0000   Min.   :3.000   Min.   :1.000  
 1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
 Median :0.0000   Median :4.000   Median :2.000  
 Mean   :0.4062   Mean   :3.688   Mean   :2.812  
 3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
 Max.   :1.0000   Max.   :5.000   Max.   :8.000  
```

mtcars sem variáveis categoricas

```
MT = mtcars[,-c(8,9)]
MT %>% cor %>% round(digits = 2)
```

```
       mpg   cyl  disp    hp  drat    wt  qsec  gear  carb
mpg   1.00 -0.85 -0.85 -0.78  0.68 -0.87  0.42  0.48 -0.55
cyl  -0.85  1.00  0.90  0.83 -0.70  0.78 -0.59 -0.49  0.53
disp -0.85  0.90  1.00  0.79 -0.71  0.89 -0.43 -0.56  0.39
hp   -0.78  0.83  0.79  1.00 -0.45  0.66 -0.71 -0.13  0.75
drat  0.68 -0.70 -0.71 -0.45  1.00 -0.71  0.09  0.70 -0.09
wt   -0.87  0.78  0.89  0.66 -0.71  1.00 -0.17 -0.58  0.43
qsec  0.42 -0.59 -0.43 -0.71  0.09 -0.17  1.00 -0.21 -0.66
gear  0.48 -0.49 -0.56 -0.13  0.70 -0.58 -0.21  1.00  0.27
carb -0.55  0.53  0.39  0.75 -0.09  0.43 -0.66  0.27  1.00
```

PCA

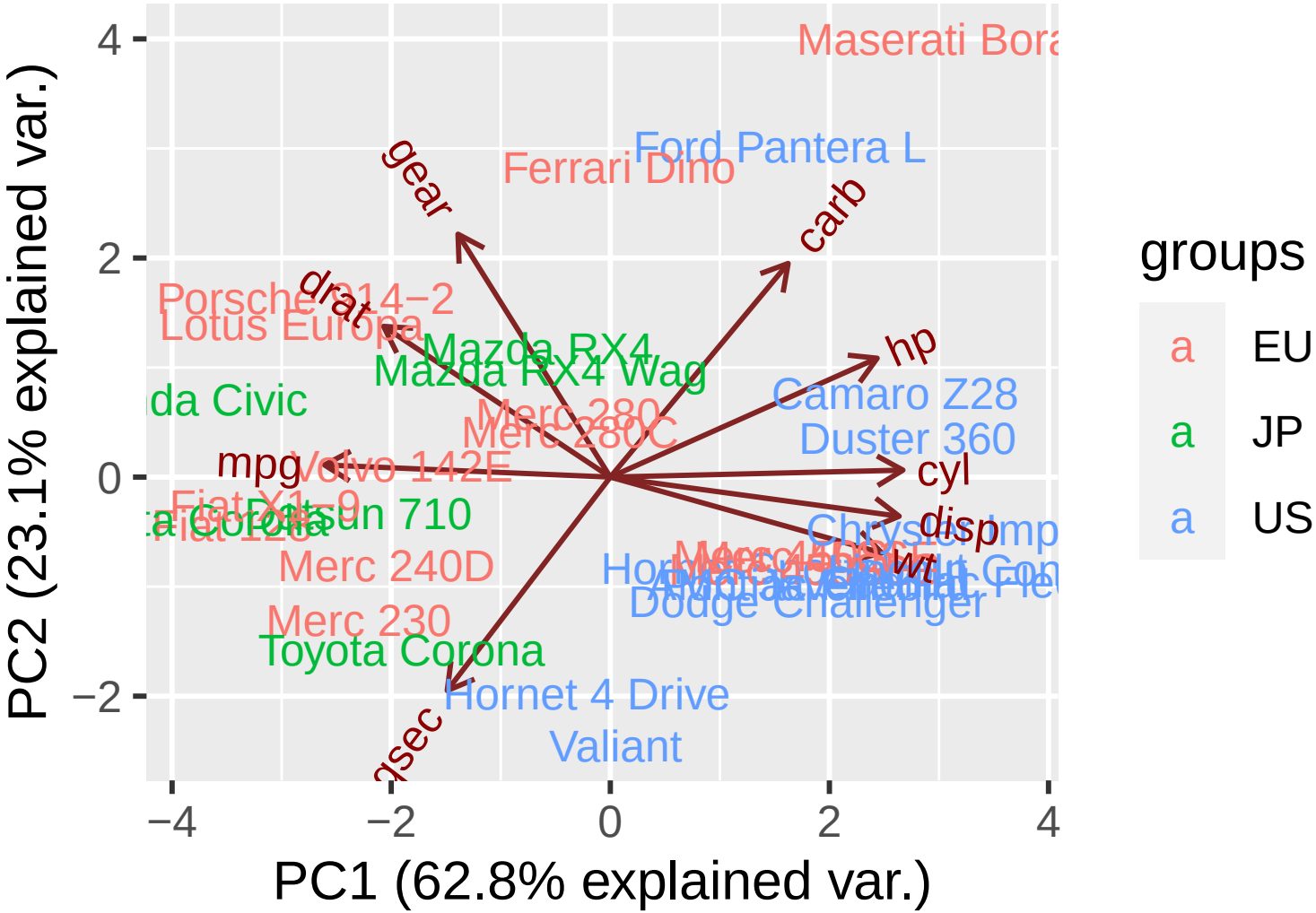
```
PCA = MT %>% prcomp(scale = TRUE)
PCA %>% summary
```

```
Importance of components:
                          PC1    PC2     PC3     PC4     PC5     PC6     PC7
Standard deviation     2.3782 1.4429 0.71008 0.51481 0.42797 0.35184 0.32413
Proportion of Variance 0.6284 0.2313 0.05602 0.02945 0.02035 0.01375 0.01167
Cumulative Proportion  0.6284 0.8598 0.91581 0.94525 0.96560 0.97936 0.99103
                          PC8     PC9
Standard deviation     0.2419 0.14896
Proportion of Variance 0.0065 0.00247
Cumulative Proportion  0.9975 1.00000
```

- 86% da variância é explicada pelos dois primeiros componentes
- 4 primeiros componentes explicam quase 95% da variância
- PC1 explica 62.8% da variância
- PC2 explica 23.1%

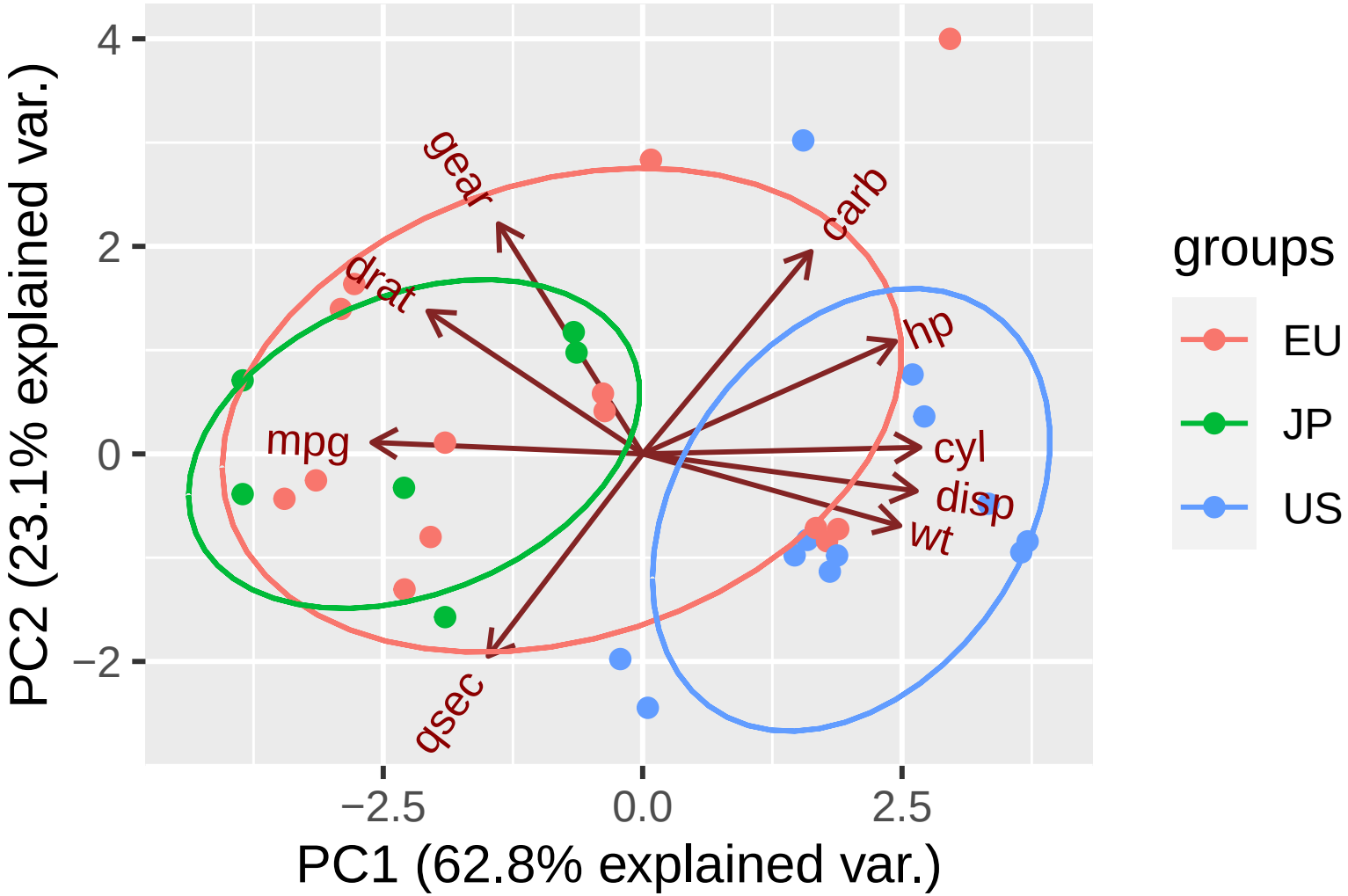
PCA

```
pais=c(rep("JP", 3), rep("US",4), rep("EU", 7),rep("US",3), "EU",
  rep("JP", 3), rep("US",4), rep("EU", 3), "US", rep("EU", 3))
PCA %>% ggbiplot(group=pais, labels = row.names(mtcars),
  obs.scale=1,var.scale=1)
```



Questões sobre o PCA

- Que relação há entre os eixos?
- Que grupos são explicados por que variáveis?



Respostas

- cyl, disp e wt estão altamente correlacionados com PC1
- mpg está altamente correlacionado negativamente com PC1
- drat também está correlacionado negativamente com PC1
- carb e gear são os mais correlacionados positivamente com PC2
- qsec é o mais correlacionado negativamente com PC2, e também está correlacionado negativamente com PC1
- Carros Americanos são caracterizados por valores altos de cyl, disp e wt
- Carros Japoneses são caracterizados por valores altos de mpg

Cachexia

1. Quantos componentes são necessários para explicar 90% da variância?
2. Que percentagem da variância é explicada pelos dois primeiros componentes?
3. Há separabilidade das duas classes pelos dois componentes?

Cachexia

```
url = "http://darwin.di.uminho.pt/dataAnalysisR/cachexia.csv"
cachexia = read.table(url, sep = ",", header = T, row.names = 1)
url = "http://darwin.di.uminho.pt/dataAnalysisR/meta_cachexia.csv"
meta_cachexia = read.table(url, sep = ",", header = T, row.names = 1)
pca_cach = cachexia %>% prcomp(scale = TRUE)
pca_cach %>% summary
```

```
Importance of components:
                          PC1    PC2     PC3     PC4     PC5    PC6     PC7
Standard deviation     5.0467 2.2701 1.83311 1.74728 1.65906 1.6130 1.47304
Proportion of Variance 0.4043 0.0818 0.05334 0.04846 0.04369 0.0413 0.03444
Cumulative Proportion  0.4043 0.4861 0.53941 0.58787 0.63156 0.6729 0.70730
                           PC8     PC9    PC10   PC11    PC12    PC13   PC14
Standard deviation     1.36403 1.24275 1.20650 1.1584 1.05503 1.03620 0.9914
Proportion of Variance 0.02953 0.02451 0.02311 0.0213 0.01767 0.01704 0.0156
Cumulative Proportion  0.73683 0.76135 0.78445 0.8057 0.82342 0.84046 0.8561
                          PC15    PC16    PC17    PC18   PC19    PC20    PC21
Standard deviation     0.96773 0.89551 0.86788 0.83041 0.8133 0.73918 0.72112
Proportion of Variance 0.01487 0.01273 0.01196 0.01095 0.0105 0.00867 0.00825
Cumulative Proportion  0.87093 0.88366 0.89562 0.90656 0.9171 0.92573 0.93399
                          PC22    PC23    PC24   PC25   PC26    PC27    PC28
Standard deviation     0.71053 0.64606 0.63389 0.5830 0.5442 0.50539 0.48743
Proportion of Variance 0.00801 0.00663 0.00638 0.0054 0.0047 0.00405 0.00377
Cumulative Proportion  0.94200 0.94863 0.95500 0.9604 0.9651 0.96916 0.97293
                          PC29    PC30    PC31    PC32    PC33    PC34    PC35
Standard deviation     0.42674 0.42427 0.41483 0.38653 0.35092 0.32424 0.31646
```

Cachexia: Quantos componentes para explicar 90% da variância?

```
pca_cach %>% summary %>% .$importance %>% .[3,]
```

```
    PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9    PC10
0.40427 0.48607 0.53941 0.58787 0.63156 0.67286 0.70730 0.73683 0.76135 0.78445
   PC11    PC12    PC13    PC14    PC15    PC16    PC17    PC18    PC19    PC20
0.80575 0.82342 0.84046 0.85606 0.87093 0.88366 0.89562 0.90656 0.91706 0.92573
   PC21    PC22    PC23    PC24    PC25    PC26    PC27    PC28    PC29    PC30
0.93399 0.94200 0.94863 0.95500 0.96040 0.96510 0.96916 0.97293 0.97582 0.97867
   PC31    PC32    PC33    PC34    PC35    PC36    PC37    PC38    PC39    PC40
0.98141 0.98378 0.98573 0.98740 0.98899 0.99030 0.99158 0.99266 0.99368 0.99465
   PC41    PC42    PC43    PC44    PC45    PC46    PC47    PC48    PC49    PC50
0.99541 0.99602 0.99659 0.99708 0.99753 0.99793 0.99830 0.99860 0.99888 0.99906
   PC51    PC52    PC53    PC54    PC55    PC56    PC57    PC58    PC59    PC60
0.99923 0.99939 0.99951 0.99962 0.99972 0.99979 0.99985 0.99990 0.99994 0.99997
   PC61    PC62    PC63
0.99998 0.99999 1.00000
```

```
pca_cach %>% summary %>% .$importance %>% .[3,] %>%
  map_lgl(~ .x > 0.9) %>% which %>% min
```

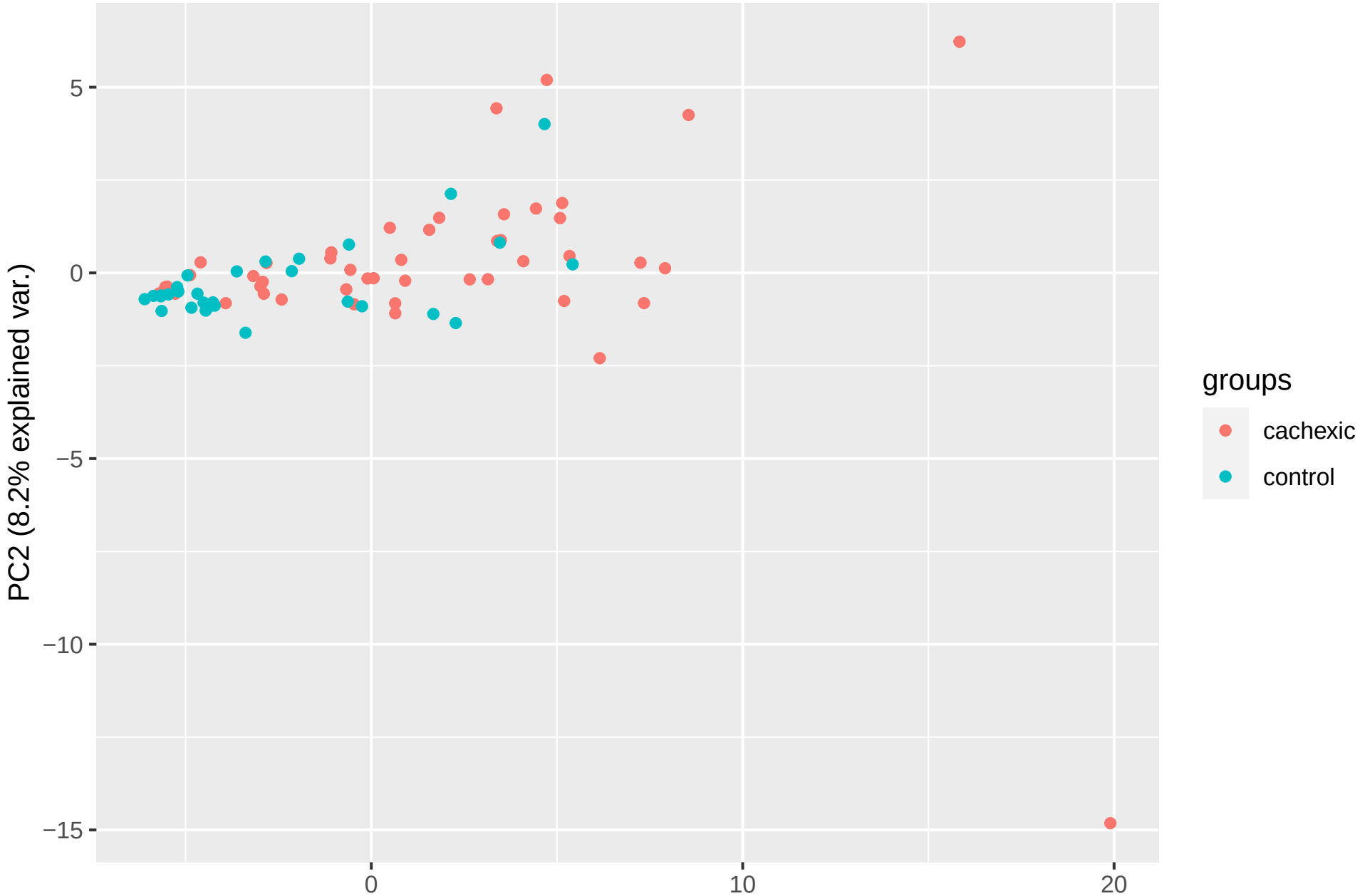
```
[1] 18
```

```
min(which(summary(pca_cach)$importance[3,] > 0.9))
```

```
[1] 18
```

Cachexia

```
ggbiplot(pca_cach,groups=meta_cachexia[,1],var.axes=FALSE,obs.scale=1,var.scale=1)
```



Compressão de imagens

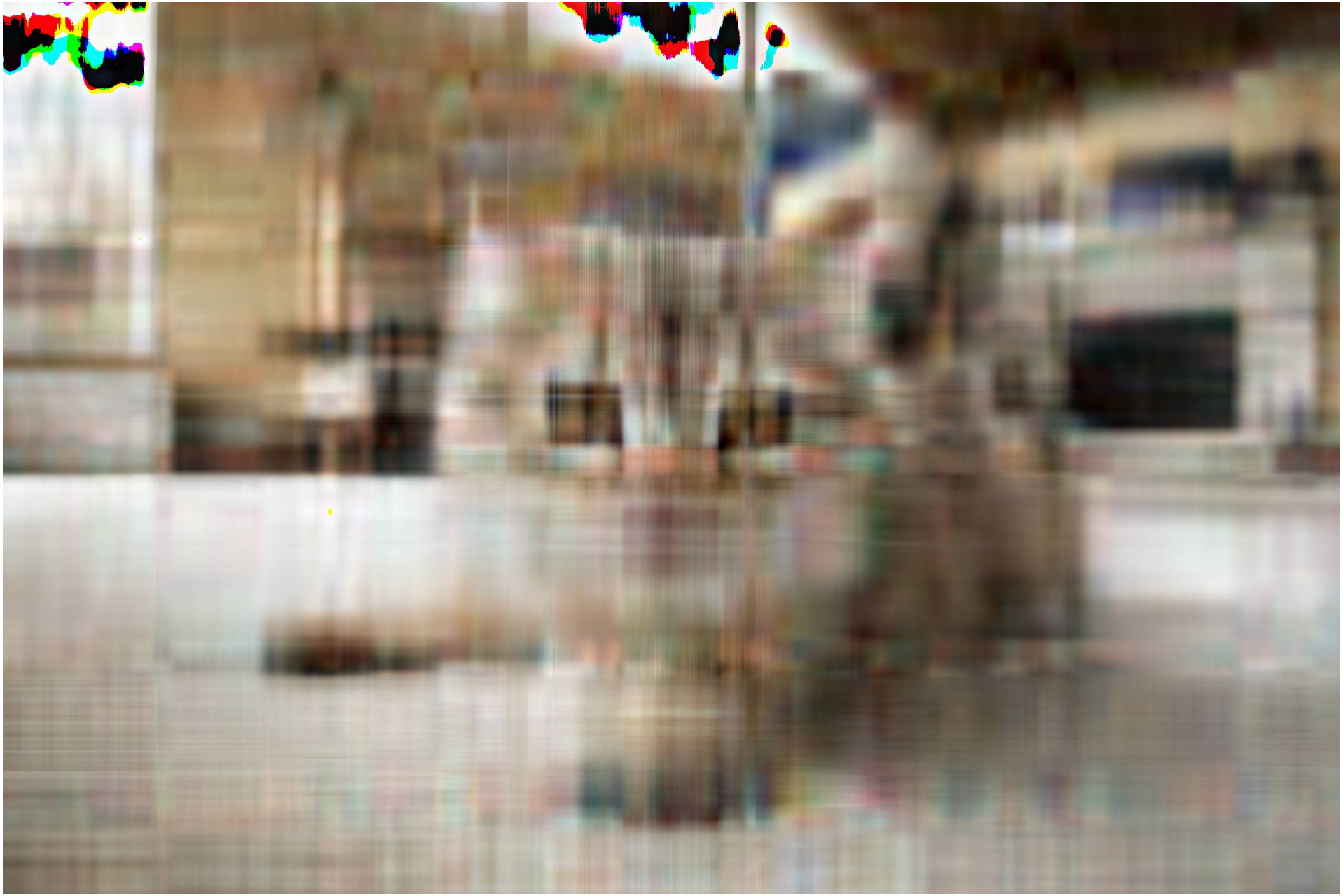
Compressão de imagens

```
library(jpeg)
gato=readJPEG('gato.jpg')
dim(gato)
```

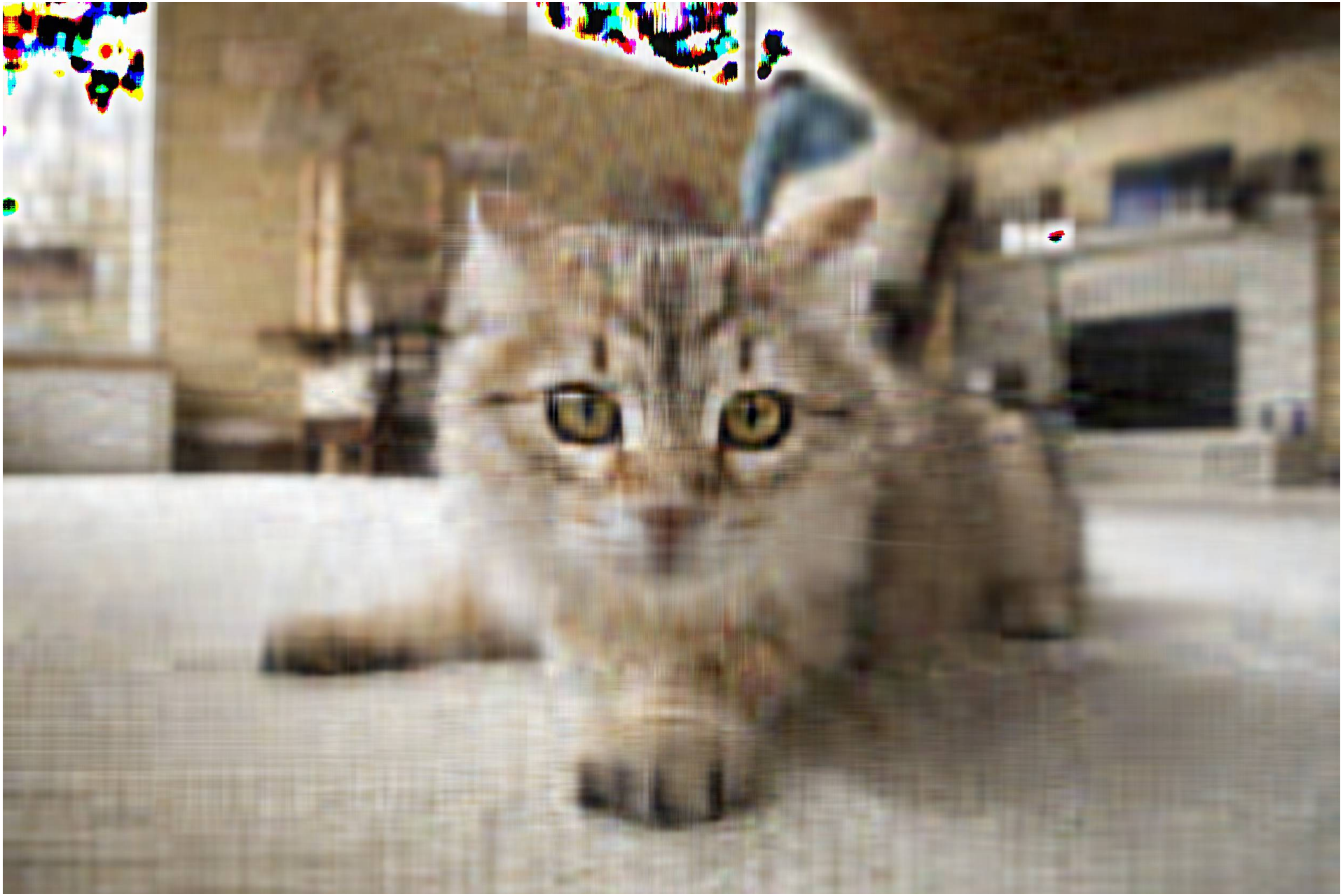
```
[1] 1666 2500    3
```

```
pca.gato=lapply(1:3, function(i)prcomp(gato[,,i], center = FALSE))
pcs = 12
img = sapply(pca.gato, function(j) {j$x[,1:pcs] %*% t(j$rotation[,1:pcs])},
         simplify = "array")
writeJPEG(img, "gato12.jpg")
```

Gato com 12 componentes



Gato com 24 componentes



Gato com 48 componentes

Gato com 128 componentes

Gato com 256 componentes

Gato com 512 componentes

ALL

1. Use os dados de expressão ALL
2. Filtre os genes para só ter aqueles cujo rácio entre máximo e mínimo seja de 2
3. Não se esqueça que os pacientes é que possuem os metadados
4. Quantos componentes são necessários para explicar pelo menos 90% da variância?
5. Consegue uma boa separação para a classe mol.biol?
6. E para a classe BT? (Neste caso só nos interessa a primeira letra)

ALL

```
library(ALL)
data(ALL)
exp = exprs(ALL)
maximos = apply(exp,1,max)
minimos = apply(exp,1,min)
vl = maximos/minimos > 2
ALLm2=ALL[vl,]
exp_m2 = exprs(ALLm2)
exp_pca = t(exp_m2)
dim(exp_pca)
```

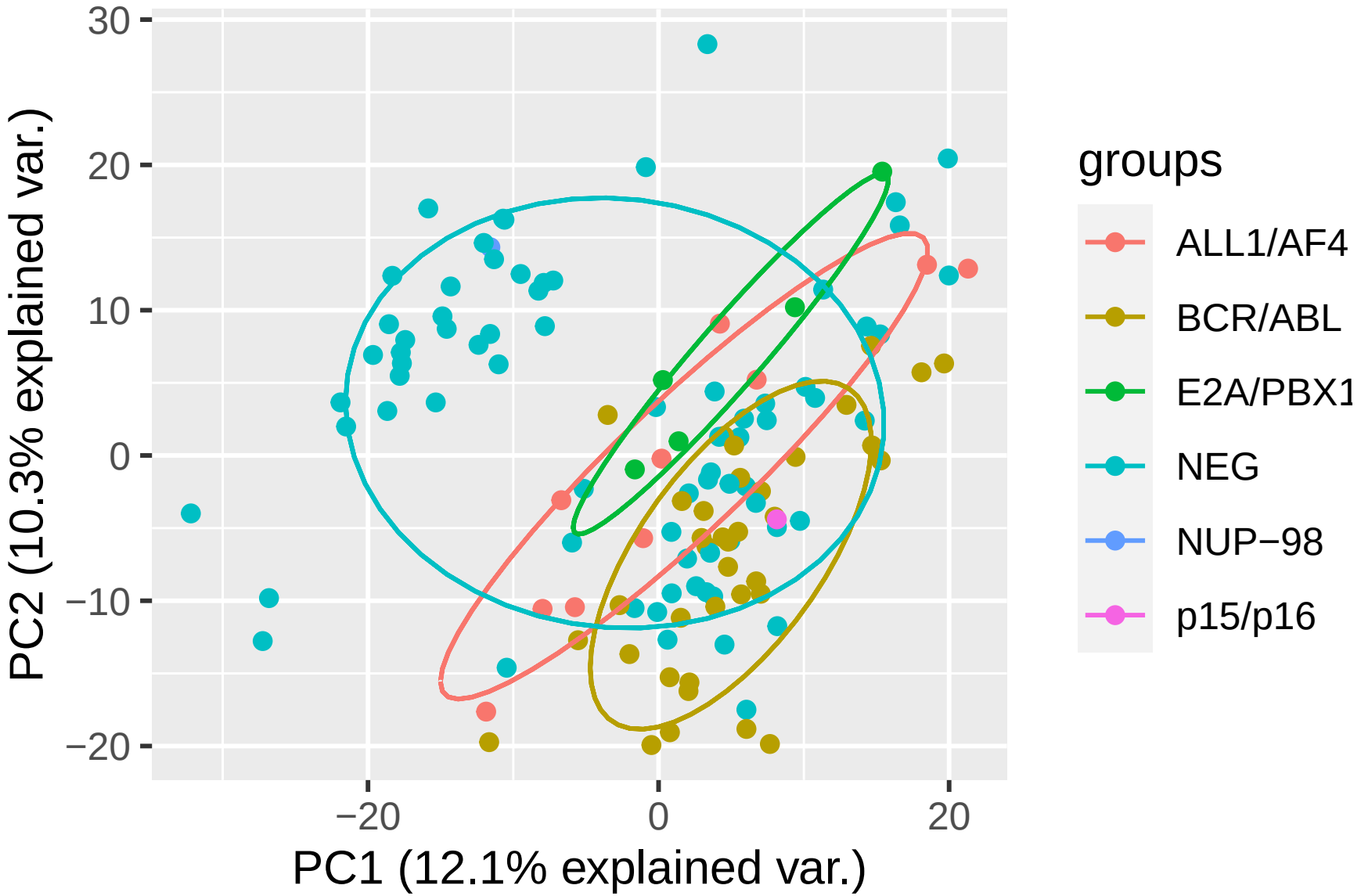
```
[1]  128 1023
```

```
pca_all = prcomp(exp_pca, scale = TRUE)
min(which(summary(pca_all)$importance[3,]>0.9))
```

```
[1] 68
```

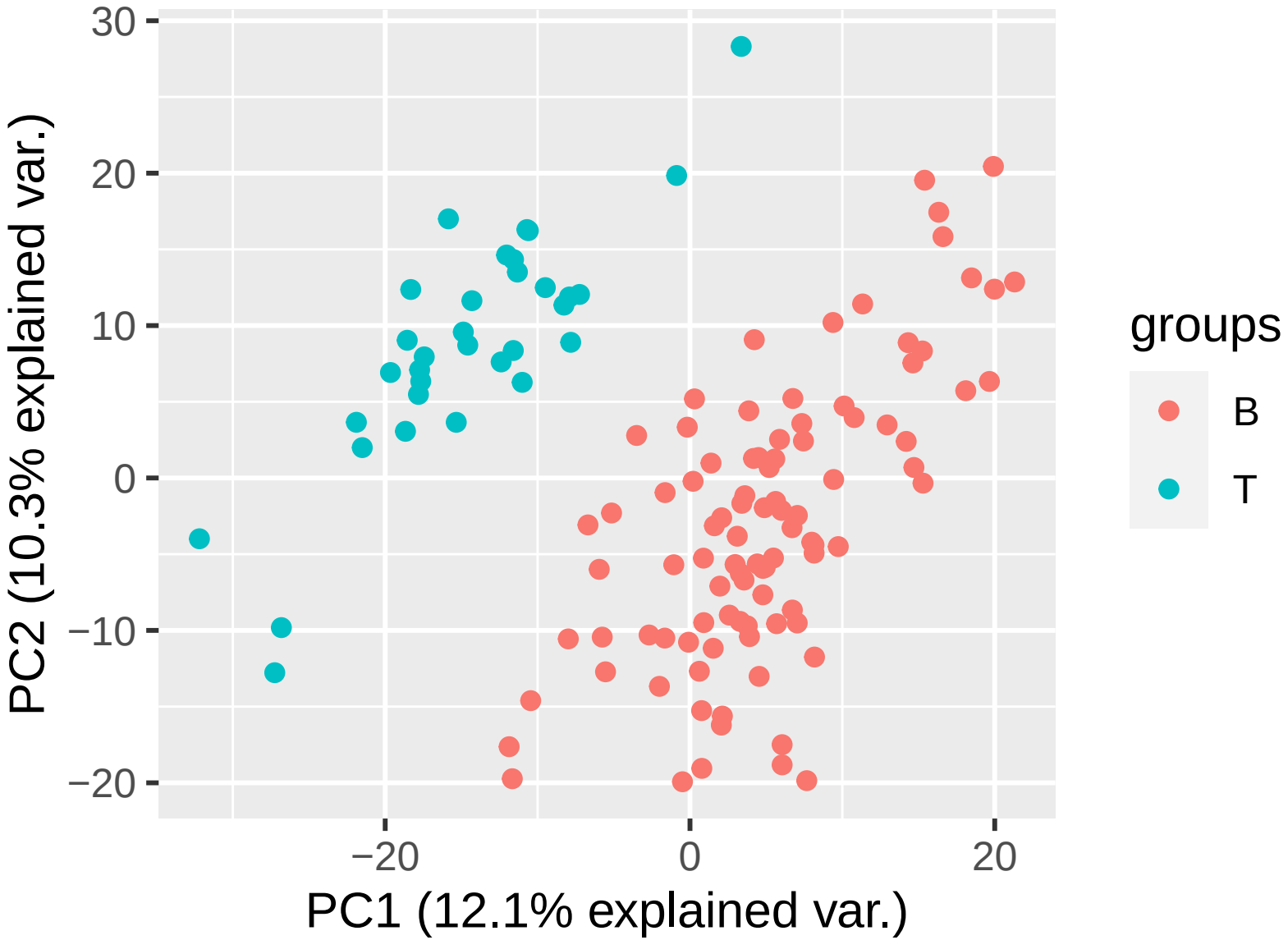
ALL

```
ggbiplot(pca_all, groups = ALL$mol.biol, var.axes = FALSE,
  obs.scale=1,var.scale=1, ellipse=T)
```



ALL

```
BT = sapply(ALL$BT, function(x)x %>% as.character %>% substr(1,1))
ggbiplot(pca_all, groups = BT, var.axes = FALSE,
  obs.scale=1,var.scale=1)
```



SVD

- Método algébrico de fatorização de matrizes
- Pode ser usado em análise de dados para reduzir a dimensionalidade
- Pode ser usado para identificar variáveis dependentes de outras, que podem ser removidas no processo de análise de dados
- Consiste na fatorização de uma matriz M de dimensões $n \times m$ em $U \cdot D \cdot V^T$
  - U é uma matriz $n \times n$, V é uma matriz $m \times m$, D tem dimensões $n \times m$;
  - $U \cdot U^T = I$ e $V \cdot V^T = I$
  - As colunas de U são os vetores **singulares esquerdos** e as de V os vetores **singulares direitos**
  - A matriz D é uma matriz diagonal com os **valores singulares** de M

SVD em R

- Através da função *svd*
- O principal argumento é a matriz de dados M
- O resultado é uma lista com três campos:
  - d : matriz diagonal D
  - u : matriz U
  - v : matriz V
- As colunas de v são equivalentes aos loadings resultantes da PCA (se os dados para esta forem normalizados)

Exemplo

```
iris[,-5] %>% scale %>% svd %>% .$v
```

```
           [,1]        [,2]       [,3]       [,4]
[1,]  0.5210659 -0.37741762  0.7195664  0.2612863
[2,] -0.2693474 -0.92329566 -0.2443818 -0.1235096
[3,]  0.5804131 -0.02449161 -0.1421264 -0.8014492
[4,]  0.5648565 -0.06694199 -0.6342727  0.5235971
```

```
prcomp(iris[,-5], scale = TRUE)$rotation
```

```
          PC1         PC2        PC3        PC4
SL  0.5210659 -0.37741762  0.7195664  0.2612863
SW -0.2693474 -0.92329566 -0.2443818 -0.1235096
PL  0.5804131 -0.02449161 -0.1421264 -0.8014492
PW  0.5648565 -0.06694199 -0.6342727  0.5235971
```

Compressão de imagens com svd

```
svd.gato = lapply(1:3, function(i)svd(gato[,,i]))
j = 48
a=sapply(svd.gato, function(i) {
  i$u[,1:j] %*% diag(i$d[1:j]) %*% t(i$v[,1:j])
}, simplify = 'array')
writeJPEG(a, "gato_svg48.jpg")
```

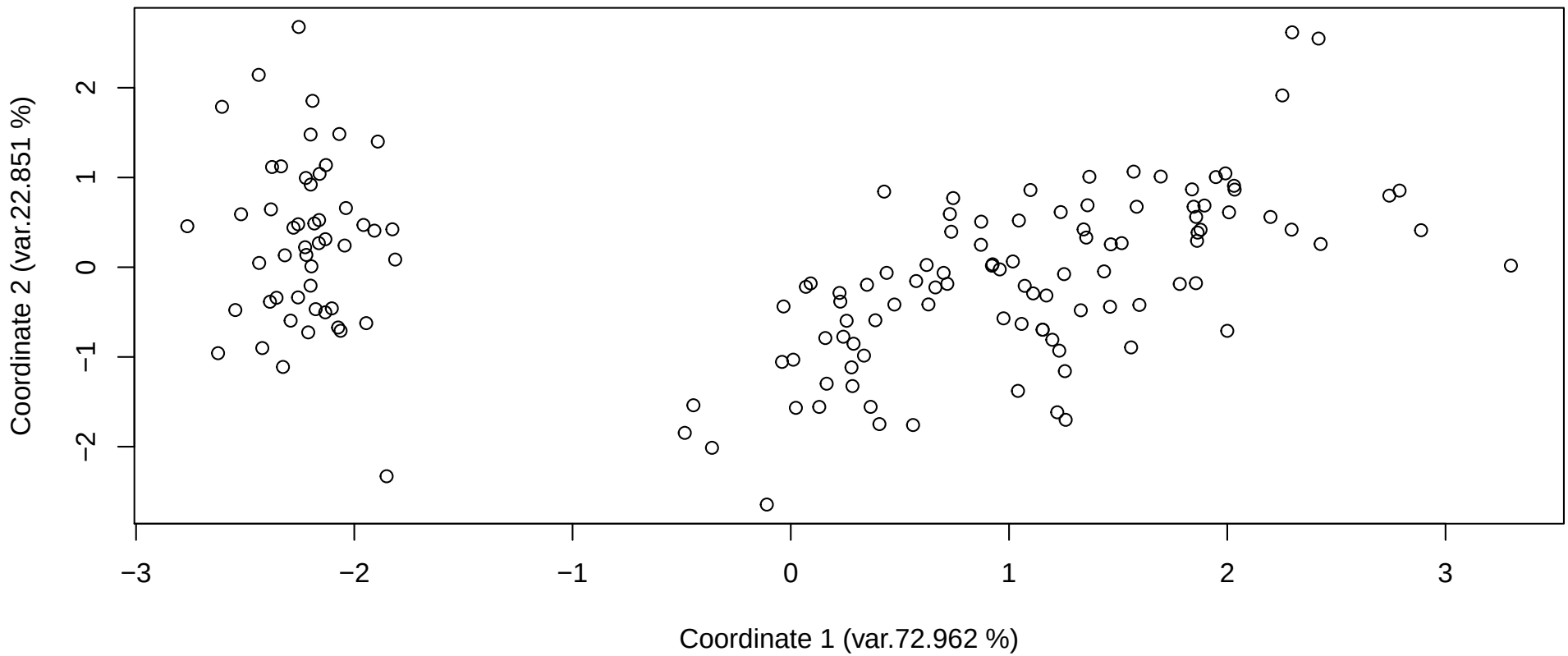
Compressão de imagens com svd

Análise Multi Dimensional - MDS

- Multi-Dimensional Scaling (MDS) é uma técnica de visualização de informação
- Dada uma matriz de distâncias entre objetos
- Tenta colocar cada um dos objetos num espaço de n dimensões de forma a preservar o melhor possível a sua distância original.
- Se n = 2 podemos visualizar os pontos a 2 dimensões
- Existem várias variantes de MDS incluindo algoritmos ditos métricos e não métricos

MDS métrico

```
iris.sc = scale(iris[-5])
dist.iris = dist(iris.sc, method = "euclidean")
cmd.mds <- cmdscale(dist.iris, 3, eig = TRUE)
# Proporção da variância explicada
var.pc1 = sum(cmd.mds$eig[1])/sum(abs(cmd.mds$eig)) * 100
var.pc2 = sum(cmd.mds$eig[2])/sum(abs(cmd.mds$eig)) * 100
var.pc3 = sum(cmd.mds$eig[3])/sum(abs(cmd.mds$eig)) * 100
lab.pc1 = paste("Coordinate 1", sprintf("(var.%.3f", var.pc1), "%)")
lab.pc2 = paste("Coordinate 2", sprintf("(var.%.3f", var.pc2), "%)")
plot(cmd.mds$points[,c(1,2)],xlab=lab.pc1,ylab=lab.pc2,main="")
```



MDS não métrico

```
library(MASS)
# Necessário mudar matriz de distâncias pois há uma réplica perfeita nos dados
dist.iris[dist.iris==0] = 0.00001
ord = isoMDS(dist.iris, k=3) # for 3 dimensions
```

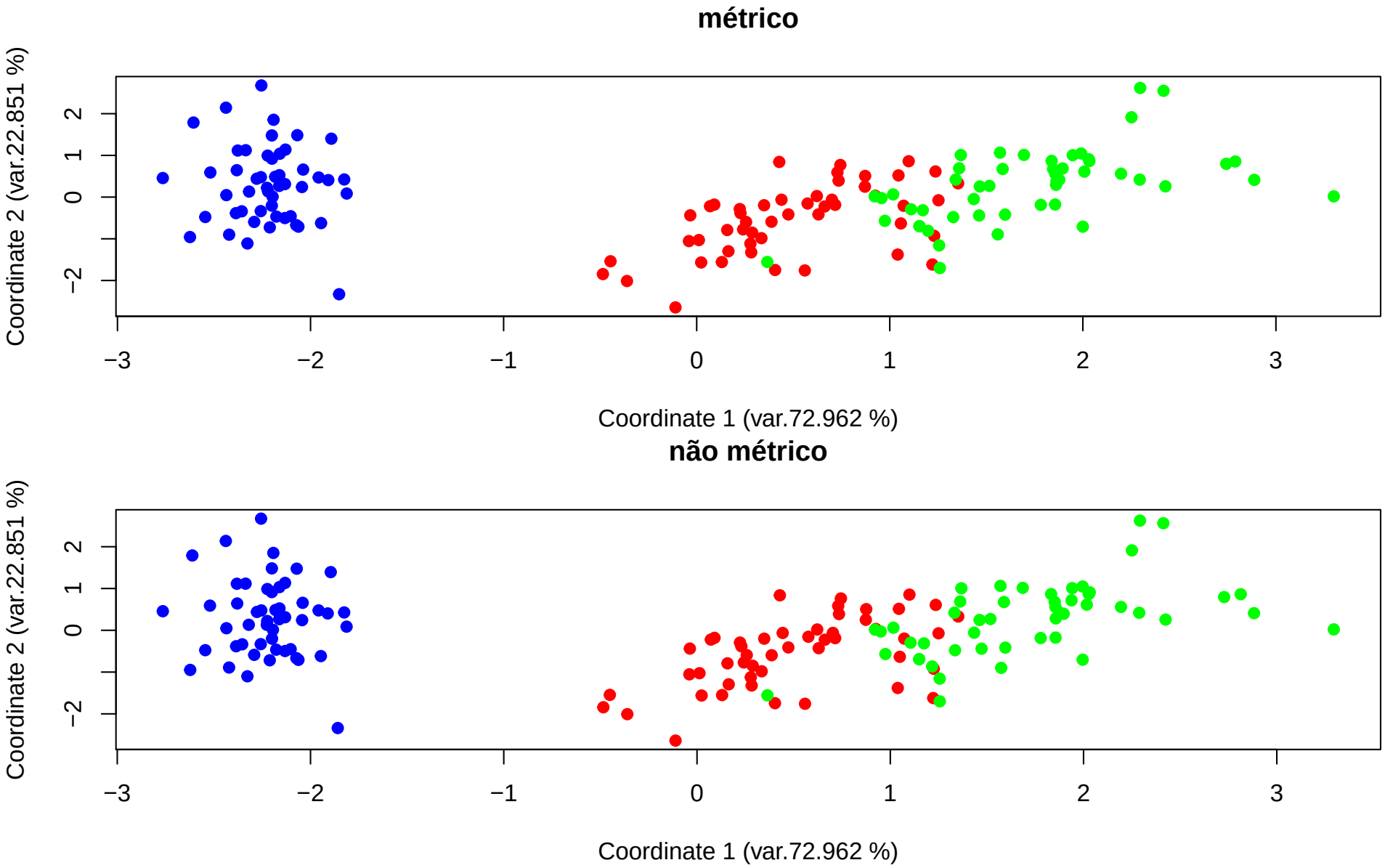
```
initial  value 0.757426
iter   5 value 0.639512
final  value 0.634255
converged
```

MDS não métrico

```
mcols = rep("gray", length(iris[,5]))
mcols[iris[,5]=="setosa"] = "blue"
mcols[iris[,5]=="versicolor"] = "red"
mcols[iris[,5]=="virginica"] = "green"
```

```
plot(cmd.mds$points[,c(1,2)],xlab=lab.pc1,ylab=lab.pc2,
     col= mcols, pch = 19, main="métrico")
plot(ord$points[,c(1,2)],xlab=lab.pc1,ylab=lab.pc2,
     main="não métrico", col= mcols, pch = 19)
```

MDS métrico vs não métrico



T-SNE t-Distributed Stochastic Neighbor Embedding

- Técnica alternativa e mais recente ao PCA e ao MDS para realizar a redução de dimensionalidade e mais voltada para a visualização de dados em 2D ou 3D
- Bastante utilizada para visualização de dados em projetos de Aprendizagem Máquina e Deep Learning
- É uma técnica não linear e que tal como o MDS se baseia na preservação de distâncias em pares de pontos, mas usando técnicas de otimização e funções objetivo distintas

Exemplo

```
install.packages("Rtsne")
library(Rtsne)
# Erro porque há duplicados
Rtsne(iris[1:4])
```

```
iris_nd = iris[!duplicated(iris),]
dim(iris_nd)
```

```
[1] 149   5
```

```
res_tnse = Rtsne(iris_nd[1:4])
plot(res_tnse$Y, col = iris_nd$Species, pch = 19)
```