practica 2

Tabla de contenidos

Principal Component Analysis (PCA)

Introducción

Principal Component Analysis (PCA) es un método estadístico que permite simplificar la complejidad de espacios muestrales con muchas dimensiones a la vez que conserva su informacion. El método de PCA permite por lo tanto "condensar" la información aportada por múltiples variables en solo unas pocas componentes. Esto lo convierte en un método muy útil de aplicar previa utilización de otras técnicas estadísticas tales como regresión, clustering... Aun así no hay que olvidar que sigue siendo necesario disponer del valor de las variables originales para calcular las componentes.

Ejemplo 1 cálculo directo de PCA con R

El set de datos USArrests del paquete básico de R contiene el porcentaje de asaltos (Assault), asesinatos (Murder) y secuestros (Rape) por cada 100,000 habitantes para cada uno de los 50 estados de USA (1973). Además, también incluye el porcentaje de la población de cada estado que vive en zonas rurales (UrbanPoP).

base de datos de los arrestos de Estados Unidos

```
data("USArrests")
head(USArrests)
```

```
           Murder Assault UrbanPop Rape
Alabama      13.2     236       58 21.2
Alaska       10.0     263       48 44.5
Arizona       8.1     294       80 31.0
Arkansas      8.8     190       50 19.5
California    9.0     276       91 40.6
Colorado      7.9     204       78 38.7
```

El promedio de los datos muestra que hay tres veces más secuestros que asesinatos y 8 veces más asaltos que secuestros.

varianzas

```
apply(X = USArrests, MARGIN = 2, FUN = mean)
```

```
  Murder  Assault UrbanPop     Rape 
   7.788  170.760   65.540   21.232 
```

La varianza es muy distinta entre las variables, en el caso de Assault, la varianza es varios órdenes de magnitud superior al resto.

```
apply(X = USArrests, MARGIN = 2, FUN = var)
```

```
   Murder    Assault   UrbanPop       Rape
 18.97047 6945.16571  209.51878   87.72916
```

Si no se estandarizan las variables para que tengan media cero y desviación estándar 1 antes de realizar el estudio PCA, la variable Assault dominará la mayoría de las componentes principales.

##PCA

función prcomp

La función prcomp() es una de las múltiples funciones en R que realizan PCA. Por defecto, prcomp() centra las variables para que tengan media cero, pero si se quiere además que su desviación estándar sea de uno, hay que indicar scale = TRUE.

```
pca <- prcomp(USArrests, scale = TRUE)
names(pca)
```

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

Los elementos center y scale almacenados en el objeto pca contienen la media y desviación típica de las variables previa estandarización (en la escala original).

```
pca$center
```

```
  Murder  Assault UrbanPop     Rape
   7.788  170.760   65.540   21.232
```

```
pca$scale
```

```
   Murder   Assault  UrbanPop      Rape
 4.355510 83.337661 14.474763  9.366385
```

número máximo de componentes principales

rotation contiene el valor de los loadings para cada componente (eigenvector). El número máximo de componentes principales se corresponde con el mínimo(n-1,p), que en este caso es min(49,4)=4 .

```
pca$rotation
```

```
                PC1        PC2        PC3         PC4
Murder   -0.5358995 -0.4181809  0.3412327  0.64922780
Assault  -0.5831836 -0.1879856  0.2681484 -0.74340748
UrbanPop -0.2781909  0.8728062  0.3780158  0.13387773
Rape     -0.5434321  0.1673186 -0.8177779  0.08902432
```

Analizar con detalle el vector de loadings que forma cada componente puede ayudar a interpretar que tipo de información recoge cada una de ellas. Por

ejemplo, la primera componente es el resultado de la siguiente combinación lineal de las variables originales:

PC1=−0.5358995 Murder−0.5831836 Assault−0.2781909 UrbanPop−0.5434321 Rape

Los pesos asignados en la primera componente a las variables Assault, Murder y Rape son aproximadamente iguales entre ellos y bastante superiores al asignado a UrbanPoP, esto significa que la primera componente recoge mayoritariamente la información correspondiente a los delitos. En la segunda componente, es la variable UrbanPoP la que tiene con diferencia mayor peso, por lo que se corresponde principalmente con el nivel de urbanización del estado. Si bien en este ejemplo la interpretación de las componentes es bastante clara, no en todos los casos ocurre lo mismo.

La función prcomp() calcula automáticamente el valor de las componentes principales para cada observación (principal component scores) multiplicando los datos por los vectores de loadings. El resultado se almacena en la matriz x.

```
head(pca$x)
```

```
                PC1        PC2         PC3          PC4
Alabama    -0.9756604 -1.1220012  0.43980366  0.154696581
Alaska     -1.9305379 -1.0624269 -2.01950027 -0.434175454
Arizona    -1.7454429  0.7384595 -0.05423025 -0.826264240
Arkansas    0.1399989 -1.1085423 -0.11342217 -0.180973554
California -2.4986128  1.5274267 -0.59254100 -0.338559240
Colorado   -1.4993407  0.9776297 -1.08400162  0.001450164
```

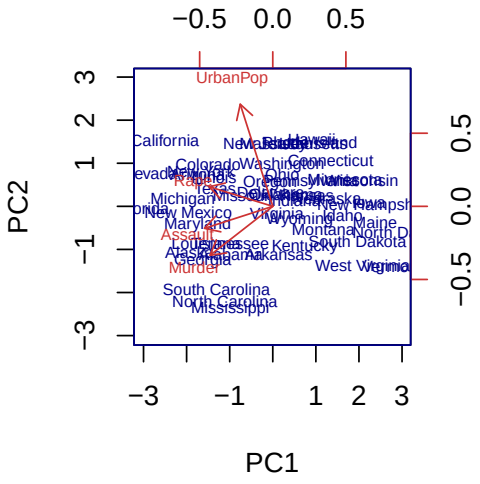
```
dim(pca$x)
```

```
[1] 50  4
```

función biplot

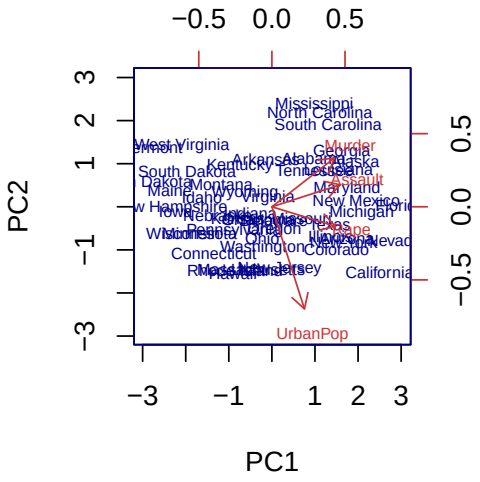
Mediante la función biplot() se puede obtener una representación bidimensional de las dos primeras componentes. Es recomendable indicar el argumento scale = 0 para que las flechas estén en la misma escala que las componentes.

```
biplot(x = pca, scale = 0, cex = 0.6, col = c("blue4", "brown3"))
```



La imagen especular, cuya interpretación es equivalente, se puede obtener invirtiendo el signo de los loadings y de los principal component scores.

```
pca$rotation <- -pca$rotation
pca$x        <- -pca$x
biplot(x = pca, scale = 0, cex = 0.6, col = c("blue4", "brown3"))
```



varianza explicada

Una vez calculadas las componentes principales, se puede conocer la varianza explicada por cada una de ellas, la proporción respecto al total y la proporción de varianza acumulada.

```
library(ggplot2)
pca$sdev^2
```

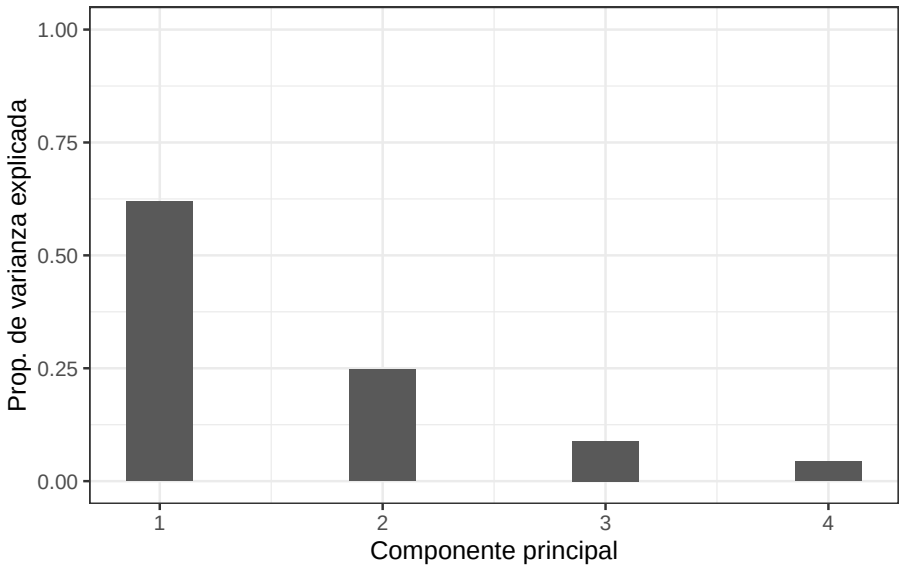
```
[1] 2.4802416 0.9897652 0.3565632 0.1734301
```

```
prop_varianza <- pca$sdev^2 / sum(pca$sdev^2)
prop_varianza
```

```
[1] 0.62006039 0.24744129 0.08914080 0.04335752
```

grafico del PCA con la varianza explicada

```
ggplot(data = data.frame(prop_varianza, pc = 1:4),
       aes(x = pc, y = prop_varianza)) +
  geom_col(width = 0.3) +
  scale_y_continuous(limits = c(0,1)) +
  theme_bw() +
  labs(x = "Componente principal",
       y = "Prop. de varianza explicada")
```



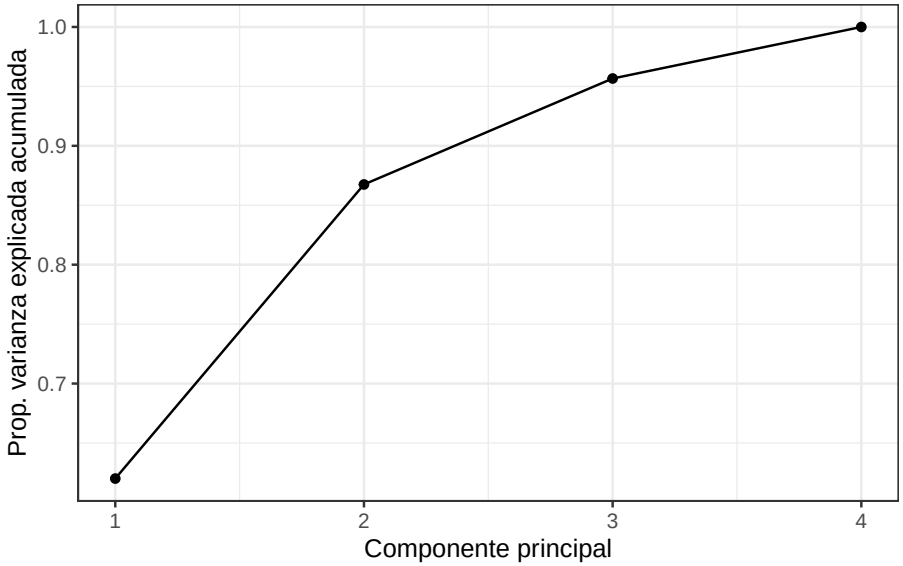
varianza explicada acumulada

```
prop_varianza_acum <- cumsum(prop_varianza)
prop_varianza_acum
```

```
[1] 0.6200604 0.8675017 0.9566425 1.0000000
```

grafico del PCA y la varianza explicada acumulada

```
ggplot(data = data.frame(prop_varianza_acum, pc = 1:4),
       aes(x = pc, y = prop_varianza_acum, group = 1)) +
  geom_point() +
  geom_line() +
  theme_bw() +
  labs(x = "Componente principal",
       y = "Prop. varianza explicada acumulada")
```



En este caso, la primera componente explica el 62% de la varianza observada en los datos y la segunda el 24.7%. Las dos últimas componentes no superan por separado el 1% de varianza explicada. Si se empleasen únicamente las dos primeras componentes se conseguiría explicar el 86.75% de la varianza observada.

Ejemplo 2 PCA aplicado a genómica

El siguiente es un ejemplo de como PCA puede emplearse para encontrar patrones cuando se dispone de muchos de predictores.

Supóngase un equipo de investigación que se encarga de clasificar los tumores de pacientes en 3 subtipos. Dependiendo del subtipo, el paciente recibe una medicación diferente. El proceso de caracterización se hace mediante tinciones y observaciones al microscopio. Este proceso es muy laborioso y lento, lo que incrementa mucho el tiempo de respuesta de los médicos. Nuevos estudios apuntan a que cuantificando la expresión de un grupo de 9 genes se podría clasificar los

tumores con una alta precisión. Se quiere determinar si tal patrón existe dentro de los datos.

Se trata de un ejemplo muy simplificado (en la realidad suele disponerse de varios miles de genes). El método de PCA puede combinarse con técnicas de clustering para crear agrupaciones automáticamente. En este caso, simplemente se muestra como PCA puede ayudar a la identificación de patrones que facilitarán posibles agrupamientos posteriores.

base de datos de tumores de pacientes

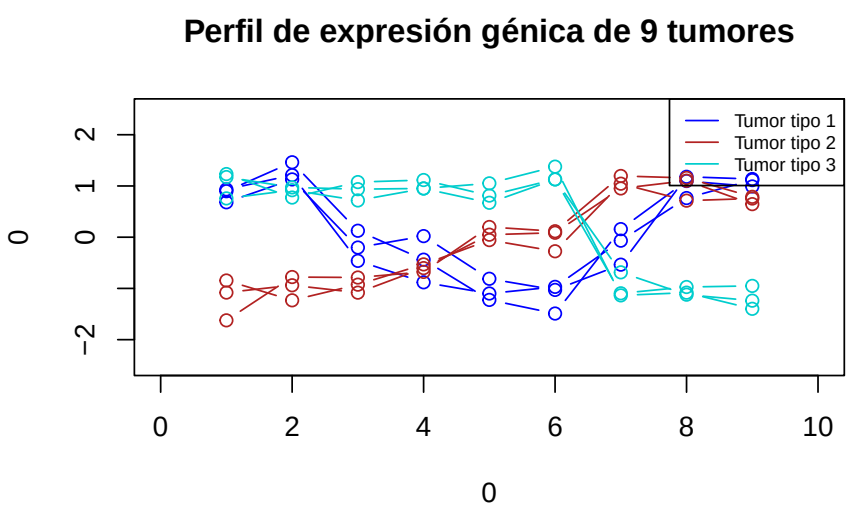
modelo de expresión génica

y

Representación de los perfiles

```
#Se crean 3 perfiles modelo de expresión génica que pueden tener los tumores
tumor_1 <- c(1,1,0,-0.5,-1,-1,0,1,1)
tumor_2 <- c(-1,-1,-1 ,-0.5,0,0,1,1,1)
tumor_3 <- c(1,1,1 ,1,1,1,-1,-1,-1)

#Añadiendo ruido aleatorio se generan 3 muestras a partir de cada perfil modelo
set.seed(755)
tumor_a <- tumor_1 + rnorm(n = 9, mean = 0, sd = 0.2)
tumor_b <- tumor_1 + rnorm(n = 9, mean = 0, sd = 0.2)
tumor_c <- tumor_1 + rnorm(n = 9, mean = 0, sd = 0.2)
tumor_d <- tumor_2 + rnorm(n = 9, mean = 0, sd = 0.2)
tumor_e <- tumor_2 + rnorm(n = 9, mean = 0, sd = 0.2)
tumor_f <- tumor_2 + rnorm(n = 9, mean = 0, sd = 0.2)
tumor_g <- tumor_3 + rnorm(n = 9, mean = 0, sd = 0.2)
tumor_h <- tumor_3 + rnorm(n = 9, mean = 0, sd = 0.2)
tumor_i <- tumor_3 + rnorm(n = 9, mean = 0, sd = 0.2)

# Representación de los perfiles
plot(0,0, xlim = c(0,10), ylim = c(-2.5, 2.5), type = "n",
     main = "Perfil de expresión génica de 9 tumores")
lines(x = 1:9, y = tumor_a, type = "b", col = "blue")
lines(x = 1:9, y = tumor_b, type = "b", col = "blue")
lines(x = 1:9, y = tumor_c, type = "b", col = "blue")

lines(x = 1:9, y = tumor_d, type = "b", col = "firebrick")
lines(x = 1:9, y = tumor_e, type = "b", col = "firebrick")
lines(x = 1:9, y = tumor_f, type = "b", col = "firebrick")

lines(x = 1:9, y = tumor_g, type = "b", col = "cyan3")
lines(x = 1:9, y = tumor_h, type = "b", col = "cyan3")
```

```
lines(x = 1:9, y = tumor_i, type = "b", col = "cyan3")
legend("topright", legend = c("Tumor tipo 1", "Tumor tipo 2", "Tumor tipo 3"),
       col = c("blue", "firebrick", "cyan3"), lty = 1, cex = 0.65)
```



La representación del perfil de expresión muestra que, aunque con variabilidad, cada grupo tiene su propio perfil característico.

dataframe

Se crea un data.frame con los datos para poder empelar ggplot2

```
library(ggplot2)
library(tidyr)
library(dplyr)
```

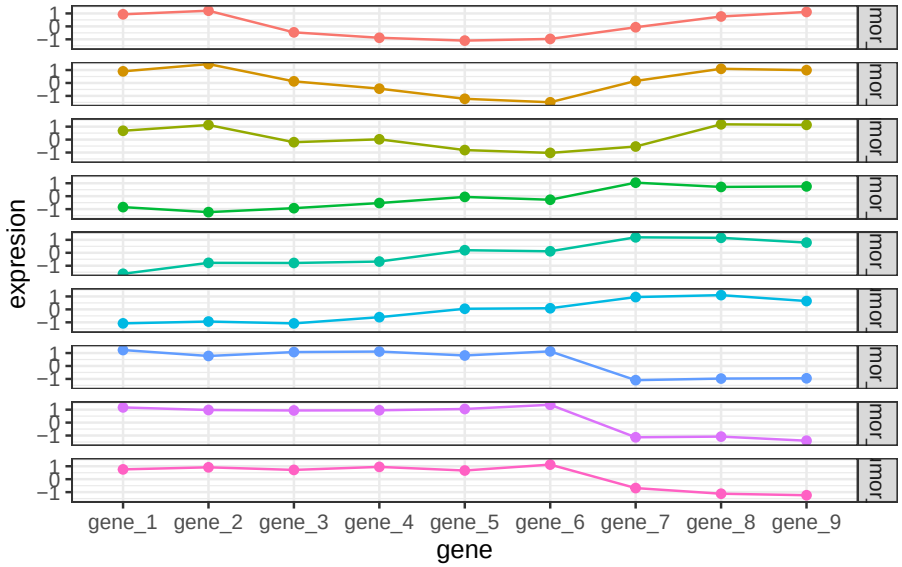
```
Adjuntando el paquete: 'dplyr'
```

```
The following objects are masked from 'package:stats':

    filter, lag
```

```
The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
datos <- data.frame(gene = paste("gene_", 1:9, sep = ""), tumor_a, tumor_b,
                                  tumor_c, tumor_d, tumor_e, tumor_f, tumor_g,
                                  tumor_h, tumor_i)
```

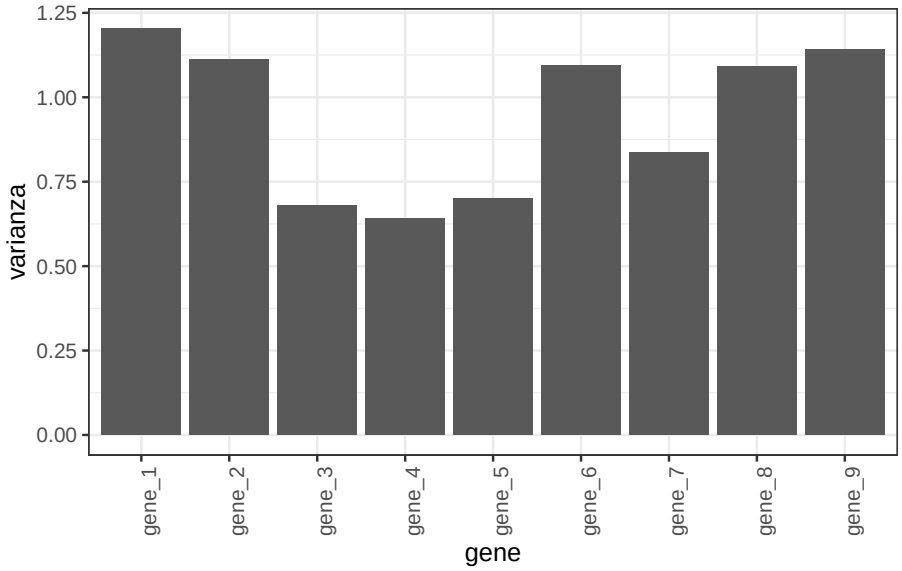
```
datos_tidy <- gather(data = datos, key = tumor, value = expresion, -1)
ggplot(data = datos_tidy, aes(x = gene, y = expresion, color = tumor)) +
  geom_path(aes(group = tumor)) +
  geom_point() +
  theme_bw() +
  facet_grid(tumor~.) +
  theme(legend.position = "none")
```



varianzade los predictores

PCA es dependiente de la escala y varianza de los predictores. En este caso se va a suponer que los genes se han medido con el mismo instrumento y la misma escala. Solo queda entonces comprobar que ninguno de ellos tiene una varianza mucho mayor al resto.

```
datos_tidy %>% group_by(gene) %>% summarise(varianza = var(expresion)) %>%
  ggplot(aes(x = gene, y = varianza)) +
  geom_col() +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



No hay ningún gen cuya varianza sea mucho mayor que la del resto, lo que significa que ninguno de ellos va dirigir el patrón resultante más que los demás.

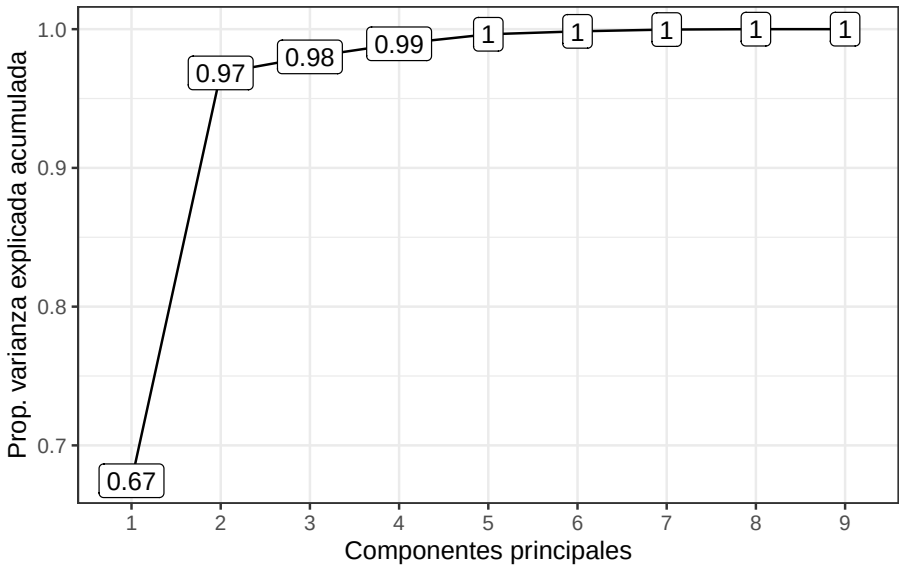
PCA

```
# Dado que se quieren estudiar patrones empleando la expresión de los genes,
# cada gen debe de estar en una columna (predictores)

datos_trans <- datos_tidy %>% spread(key = gene, value = expresion)
rownames(datos_trans) <- datos_trans$tumor
datos_trans <- datos_trans[, -1]
head(datos_trans)
```

```
           gene_1     gene_2     gene_3      gene_4      gene_5      gene_6
tumor_a  0.9323190  1.2113278 -0.4616314 -0.88042240 -1.09941170 -0.96955658
tumor_b  0.8975893  1.4624217  0.1259876 -0.43860412 -1.22377397 -1.49015370
tumor_c  0.6811663  1.1275117 -0.2017863  0.02108253 -0.81164718 -1.02881566
tumor_d -0.8451067 -1.2319309 -0.9265317 -0.53071756 -0.05634140 -0.27472142
tumor_e -1.6228992 -0.7777483 -0.7868679 -0.67712937  0.19922605  0.11591201
tumor_f -1.0805006 -0.9383486 -1.0821545 -0.60355858  0.04880287  0.08683008
            gene_7    gene_8    gene_9
tumor_a -0.06758821 0.7638985 1.1145246
tumor_b  0.15747862 1.0964449 0.9901505
tumor_c -0.53623375 1.1793956 1.1355446
tumor_d  1.04447944 0.7132720 0.7550505
tumor_e  1.19523024 1.1565408 0.7884622
tumor_f  0.95150260 1.1025851 0.6450739
```

varianza explicada acumulada

```
pca <- prcomp(datos_trans)

# Cálculo de la varianza explicada acumulada
prop_varianza <- pca$sdev^2/sum(pca$sdev^2)
prop_varianza_acum <- cumsum(prop_varianza)
ggplot(data = data.frame(prop_varianza_acum, pc = factor(1:9)),
       aes(x = pc, y = prop_varianza_acum, group = 1)) +
  geom_point() +
  geom_line() +
  geom_label(aes(label = round(prop_varianza_acum,2))) +
  theme_bw() +
  labs(x = "Componentes principales",
       y = "Prop. varianza explicada acumulada")
```



El estudio de la varianza explicada acumulada muestra que con solo dos componentes se puede capturar el 97% de la varianza, prácticamente la totalidad de la información contenida en la expresión de los 9 genes. Viendo en detalle el valor de los loadings de las dos primeras componentes se observa que en la primera los 9 genes tienen la misma importancia y lo mismo ocurre en la segunda, a excepción del gen 4 que tiene un peso mucho menor.

```
pca$rotation[, 1:2]
```

```
              PC1         PC2
gene_1 -0.3271796 -0.47294733
gene_2 -0.2321769 -0.55256710
```

```
gene_3 -0.3270151 -0.11554638
gene_4 -0.3203507  0.05025632
gene_5 -0.2356233  0.38672726
gene_6 -0.3286415  0.42850939
gene_7  0.3414044  0.22898673
gene_8  0.4191123 -0.13487735
gene_9  0.4168313 -0.22651416
```

proyección de los datos

Finalmente, la proyección de los datos en las dos primeras componentes muestra que el PCA ha sido capaz de encontrar un claro patrón que diferencia los 3 tipos de tumores.

```
biplot(x = pca, scale = 0, cex = 0.7, col = c("blue4", "brown3"))
```