Curso básico R studio - Clase extra

Stephanie Hereira Pacheco

Contents

PCA - Análisis de componentes principales

- El PCA o Análisis de componentes principales es una de las técnicas de aprendizaje no supervisado, las cuales suelen aplicarse como parte del análisis exploratorio de los datos.
- Una de las aplicaciones de PCA es la reducción de dimensionalidad (variables), perdiendo la menor cantidad de información (varianza) posible, esto, cuando contamos con un gran número de variables cuantitativas posiblemente correlacionadas (indicativo de existencia de información redundante).
- El PCA permite reducirlas a un número menor de variables transformadas (componentes principales) que expliquen gran parte de la variabilidad en los datos. Cada dimensión o componente principal generada por PCA será una combinación lineal de las variables originales.
- El PCA también sirve como herramienta para la visualización de datos.
- El PCA puede considerarse como una rotación de los ejes del sistema de coordenadas de las variables originales a nuevos ejes ortogonales, de manera que estos ejes coincidan con la dirección de máxima varianza de los datos.
- En un PCA nos interesa conocer la proporción de varianza explicada por cada uno de los componentes principales, o dicho de otra manera, cuanta información presente en los datos se pierde por la proyección de las observaciones sobre los primeros componentes principales.

PCA en R

Existen diversas librerías y/o funciones que podemos usar en R para obtener nuestro PCA, tales como:

PCA:

library(stats)

- `prcomp()` -> Forma rápida de implementar PCA sobre una matriz de datos.

library(vegan)

- `rda()`

library(FactoMineR)

- `PCA()` -> PCA con resultados más detallados. Los valores ausentes se reemplazan por la media de cada columna. Pueden incluirse variables categóricas suplementarias. Estandariza automáticamente los datos.

library(factoextra)

- `get_pca()` -> Extrae la información sobre las observaciones y variables de un análisis PCA.
- `get_pca_var()` -> Extrae la información sobre las variables.
- `get_pca_ind()` -> Extrae la información sobre las observaciones.

Visualizaciones:

library(FactoMineR)

- `fviz_pca_ind()` -> Representación de observaciones sobre componentes principales.
- `fviz_pca_var()` -> Representación de variables sobre componentes principales.
- `fviz_screeplot()` -> Representación (gráfico barras) de eigenvalores.
- `fviz_contrib()` -> Representa la contribución de filas/columnas de los resultados de un pca.

Ejemplo de PCA

Función prcomp()

Utilizaremos la data de iris para estos ejemplos, data que ya conocemos:

```
data(iris)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

Ahora aplicaremos la función de R base prcomp(), quitando los datos categóricos o factores y si observamos los elementos de este archivo de salida (en forma de lista), podemos notar:

```
pca_prcomp<- prcomp(iris[,-5], scale. = TRUE)
names(pca_prcomp)
```

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

Los elementos `"center"` y `"scale"` se corresponden con las medias y las desviaciones estándar originales de las variables previo escalado e implementación del PCA. La matriz `"rotation"`proporciona los loadings de los componentes principales (cada columna contiene el vector de loadings de cada componente principal). La función los denomina matriz de rotación ya que si multiplicáramos la matriz de datos por rotation, obtendríamos las coordenadas de los datos en el nuevo sistema rotado de coordenadas. Estas coordenadas se

corresponden con los scores de los componentes principales. Los vectores de los scores son x y la desviación estándar de cada componente principal es sdev.

En términos prácticos `"rotation"` son nuestras variables proyectadas en nuestro PCA y`"x"` son las observaciones. Si queremos extraer estos datos sencillmente usamos el operador '$'.

```
head(pca_prcomp$x)
```

```
##            PC1        PC2         PC3          PC4
## [1,] -2.257141 -0.4784238  0.12727962  0.024087508
## [2,] -2.074013  0.6718827  0.23382552  0.102662845
## [3,] -2.356335  0.3407664 -0.04405390  0.028282305
## [4,] -2.291707  0.5953999 -0.09098530 -0.065735340
## [5,] -2.381863 -0.6446757 -0.01568565 -0.035802870
## [6,] -2.068701 -1.4842053 -0.02687825  0.006586116
```

```
head(pca_prcomp$rotation)
```

```
##                     PC1         PC2        PC3        PC4
## Sepal.Length  0.5210659 -0.37741762  0.7195664  0.2612863
## Sepal.Width  -0.2693474 -0.92329566 -0.2443818 -0.1235096
## Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
## Petal.Width   0.5648565 -0.06694199 -0.6342727  0.5235971
```

La varianza explicada por cada componente principal la obtenemos elevando al cuadrado la desviación estándar:

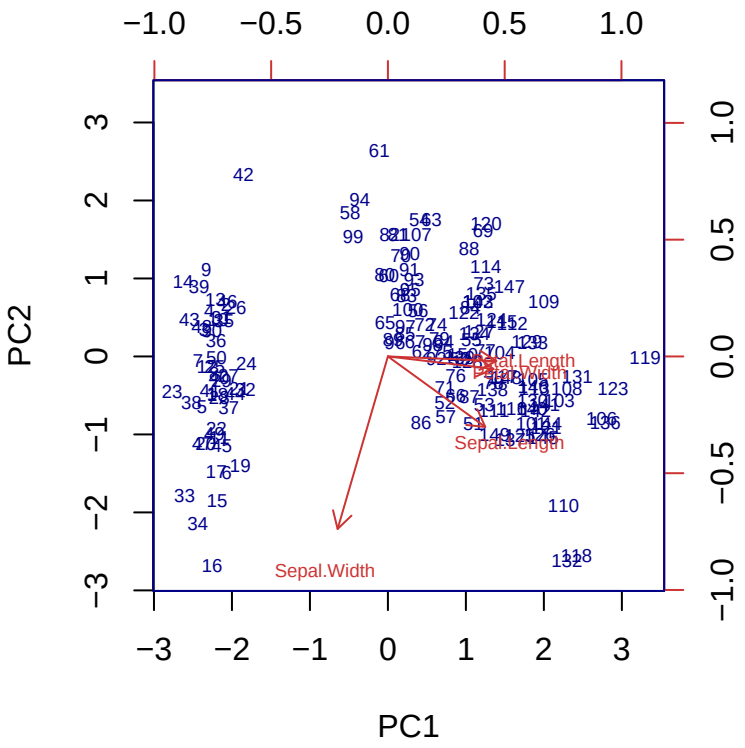
```
pca_prcomp$sdev^2
```

```
## [1] 2.91849782 0.91403047 0.14675688 0.02071484
```

```
pca_prcomp$sdev^2 / sum(pca_prcomp$sdev^2)
```

```
## [1] 0.729624454 0.228507618 0.036689219 0.005178709
```

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



Esta es una primera y básica aproximación. Ahora veremos como podemos reproducir esto con **vegan()**.

Función rda()

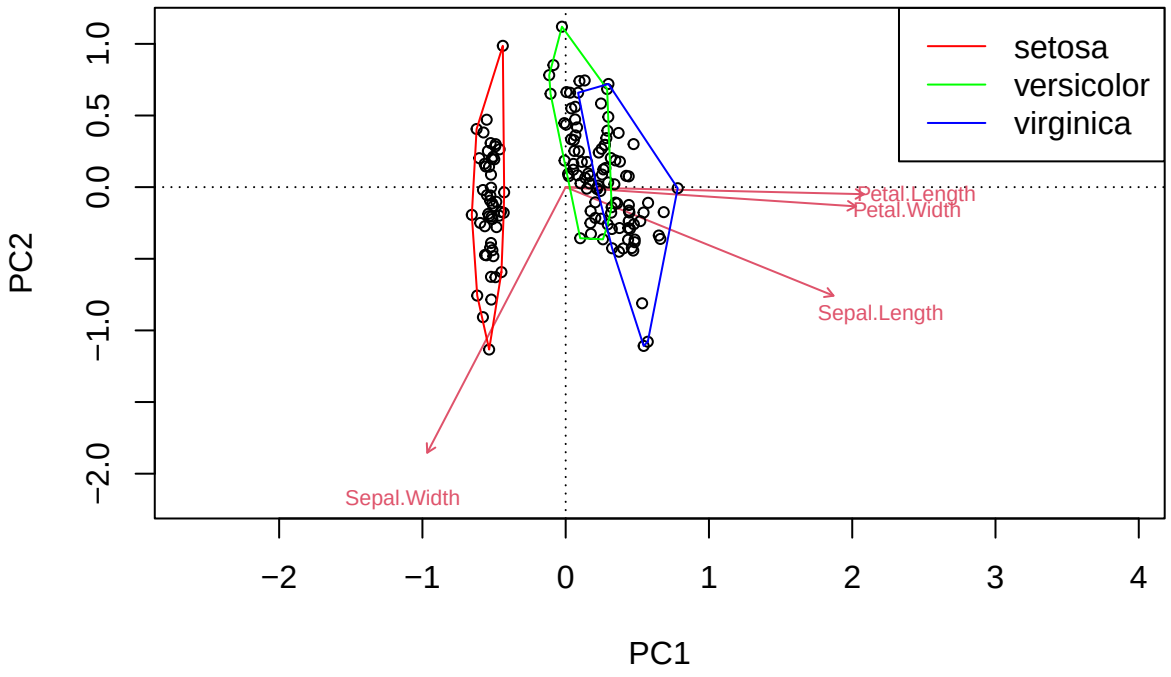
```
#install.packages("vegan")
library(vegan)
pca_vegan<- rda(iris[,-5], scale=TRUE)
#summary(pca_vegan)
pca_vegan_sum<-summary(pca_vegan)
names(pca_vegan_sum)
```

```
##  [1] "species"     "sites"       "call"        "tot.chi"     "unconst.chi"
##  [6] "cont"        "scaling"     "digits"      "inertia"     "method"
```

En términos prácticos `"species"` son nuestras variables proyectadas en nuestro PCA y`"sites"` son las observaciones. Si queremos extraer estos datos sencillmente usamos el operador '$'.

Vamos a graficarlo igual con la función **biplot()**

```
biplot(pca_vegan, type = c("text","points"))
ordihull(pca_vegan, group = iris$Species, col = c("red","green","blue"))
spp.names <- levels(iris$Species)
legend("topright", col = c("red","green","blue"), lty = 1, legend = spp.names)
```



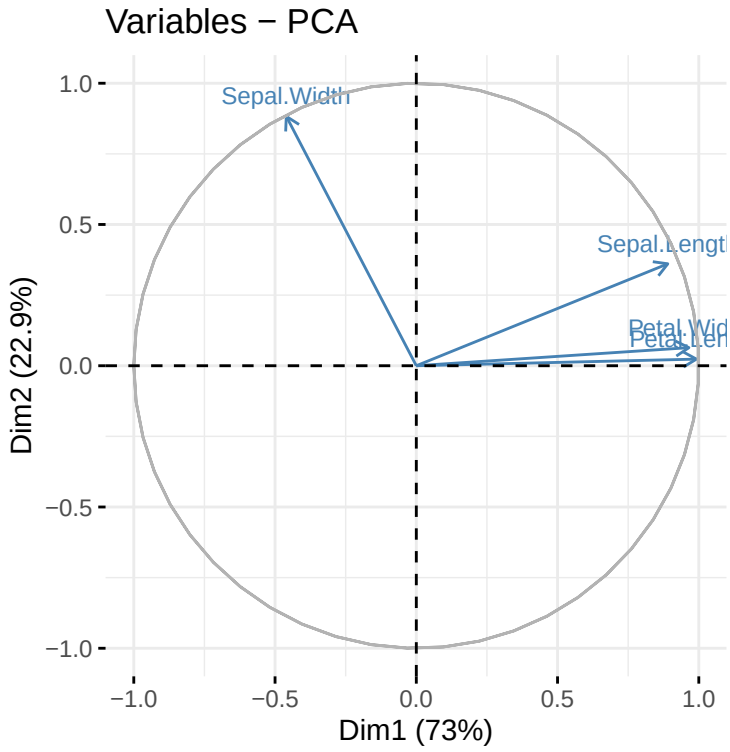
Función PCA()

```
#install.packages("factoextra")
#install.packages("FactoMineR")
library(FactoMineR)

pca_facto <- PCA(iris[,-5], graph = FALSE)
print(pca_facto)
```

```
## **Results for the Principal Component Analysis (PCA)**
## The analysis was performed on 150 individuals, described by 4 variables
## *The results are available in the following objects:
##
##    name               description
## 1  "$eig"             "eigenvalues"
## 2  "$var"             "results for the variables"
## 3  "$var$coord"       "coord. for the variables"
## 4  "$var$cor"         "correlations variables - dimensions"
## 5  "$var$cos2"        "cos2 for the variables"
## 6  "$var$contrib"     "contributions of the variables"
## 7  "$ind"             "results for the individuals"
## 8  "$ind$coord"       "coord. for the individuals"
## 9  "$ind$cos2"        "cos2 for the individuals"
## 10 "$ind$contrib"     "contributions of the individuals"
## 11 "$call"            "summary statistics"
## 12 "$call$centre"     "mean of the variables"
## 13 "$call$ecart.type" "standard error of the variables"
## 14 "$call$row.w"      "weights for the individuals"
## 15 "$call$col.w"      "weights for the variables"
```

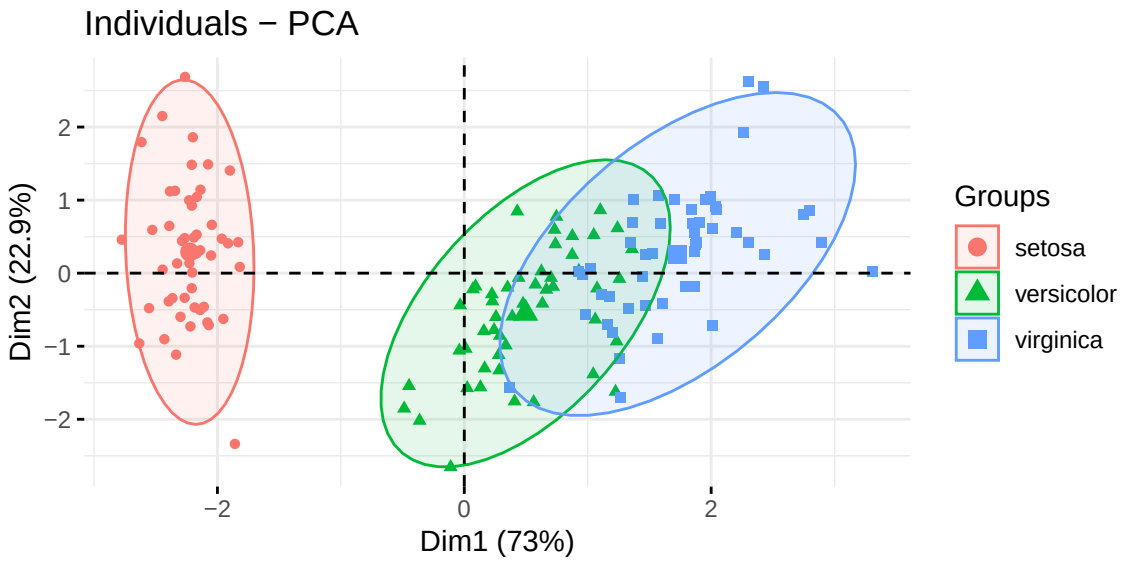
El "eig*coord*"*nosmuestralaexplicaciónde*l*oscomponentesprincipales*, "*ind*coord" los individuale u observaciones y "var" las variables. Sin embargo para extraerlo y visualizarlos de mejor manera se utiliza el paquete **factoextra**. Primero grafiquemos las variables:

```
library(factoextra)

fviz_pca_var(pca_facto, col.var="steelblue", labelsize=3)
```



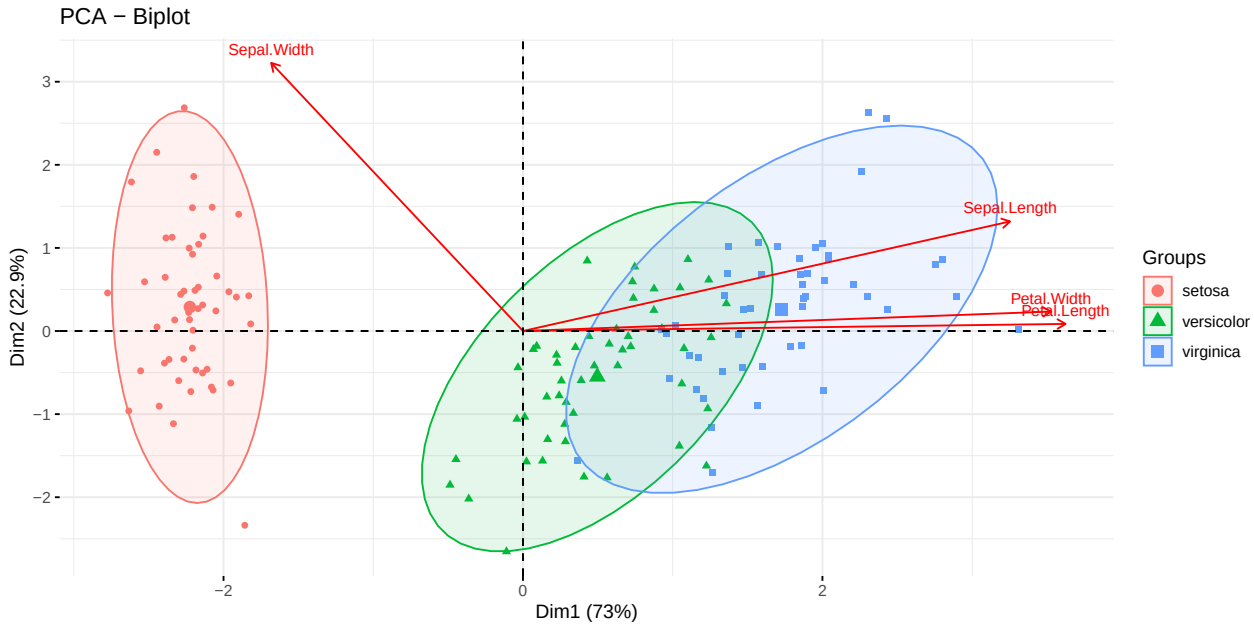
Y ahora las observaciones o indiviudales:

```
fviz_pca_ind(pca_facto, geom.ind = "point",  axes = c(1, 2),  habillage=iris$Species,
             addEllipses=TRUE, ellipse.level=0.95)
```



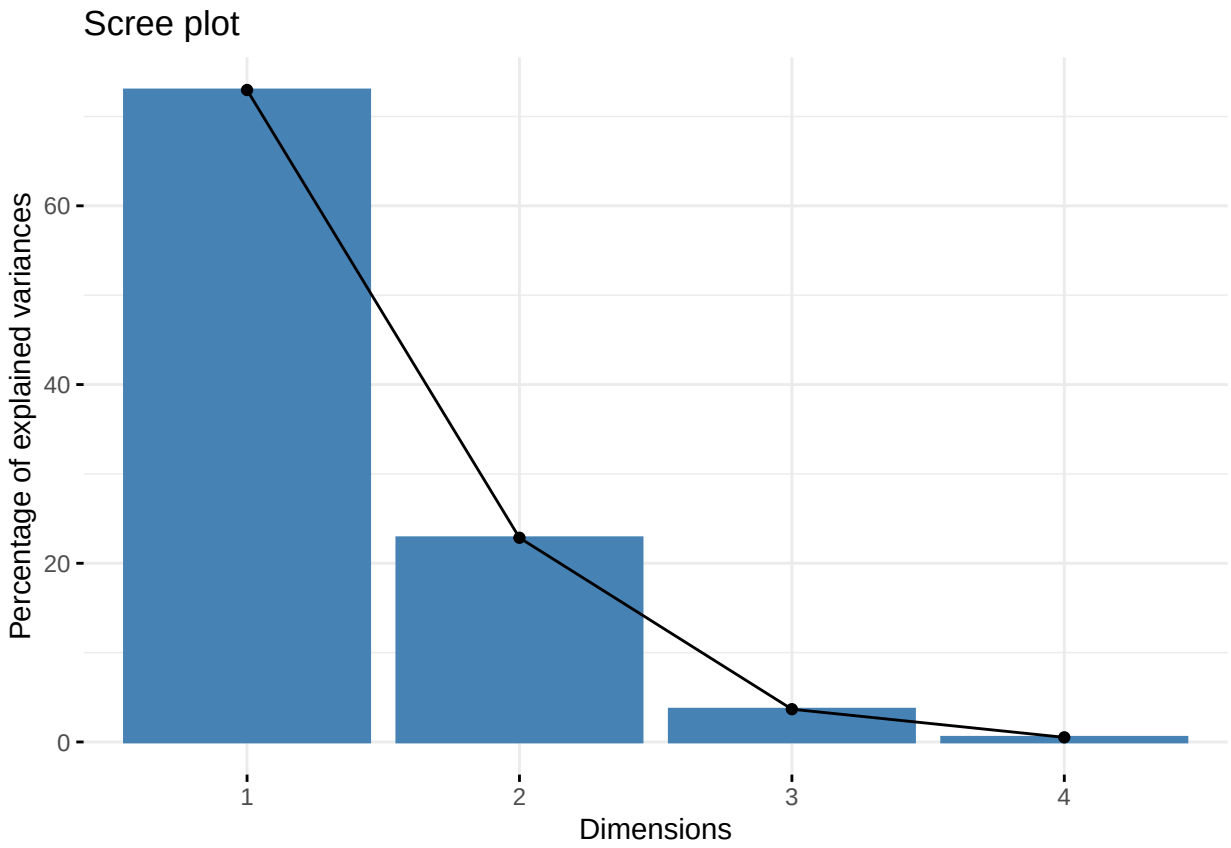
Ahora juntando:

```
fviz_pca_biplot(pca_facto,
  habillage = iris$Species, addEllipses = TRUE,
  col.var = "red",  label = "var", labelsize=3)
```



Otro gráfico que puede ser útil en el PCA es la variación a través de los componentes principales:

```
fviz_screeplot(pca_facto)
```



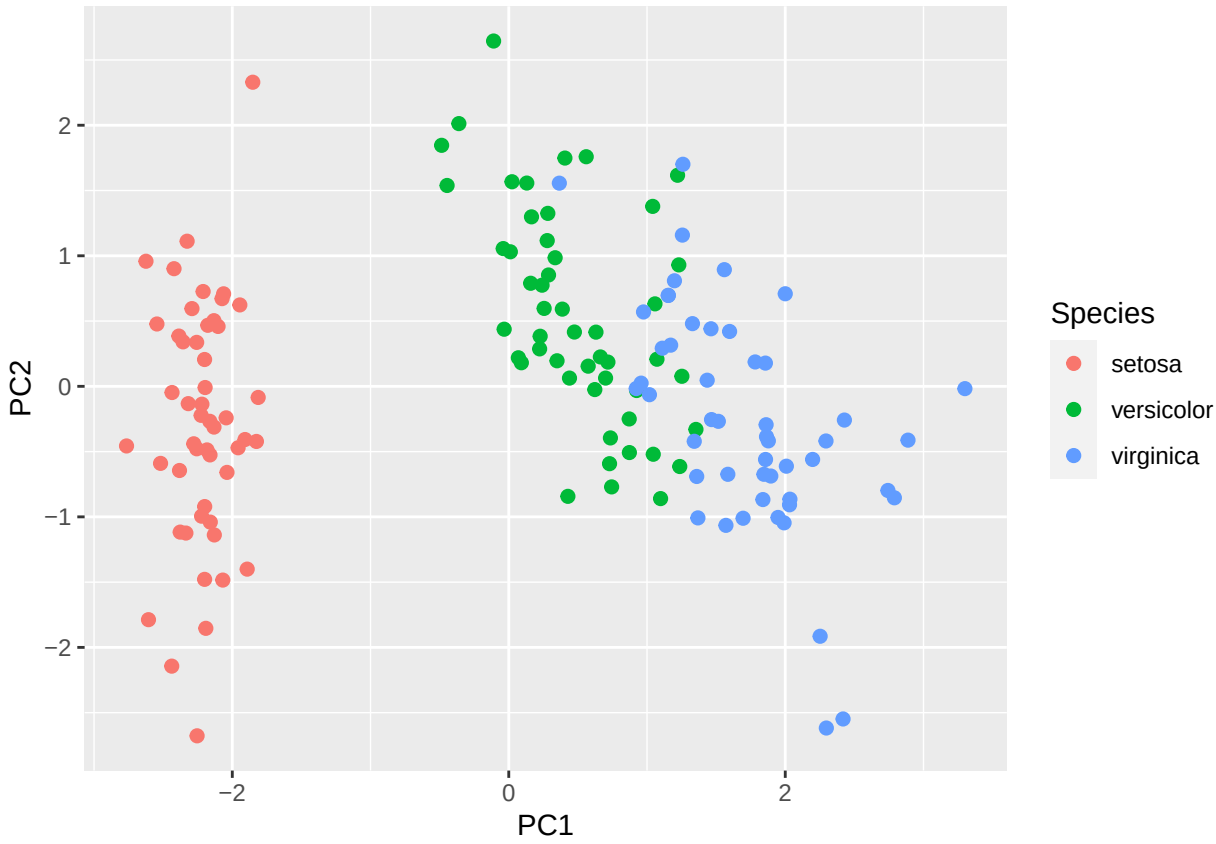
PCA gráfica con ggplot2

Para esta gráfica usaremos el objeto obtenido de **prcomp**, de hecho si extraemos la información de las coordenadas de las variables y observaciones podemos realizar este gráfico, usaré este ejemplo porque es el paquete que viene por defecto en R. Para esto entonces extraemos la información de nuestro objeto en varias datas:

```
library(tidyverse)
individuales <- data.frame(pca_prcomp$x) %>% select(PC1, PC2)%>% rownames_to_column(
  var="inds") %>% inner_join(iris %>% rownames_to_column(var="inds"))
vars<-data.frame(pca_prcomp$rotation) %>% select(PC1, PC2)%>% rownames_to_column(var="vars")
componentes<-pca_prcomp$sdev^2 / sum(pca_prcomp$sdev^2)
PC1_label <- paste("PC1", round(componentes[1]*100,1), "%")
PC2_label <- paste("PC2", round(componentes[2]*100,1), "%")
```

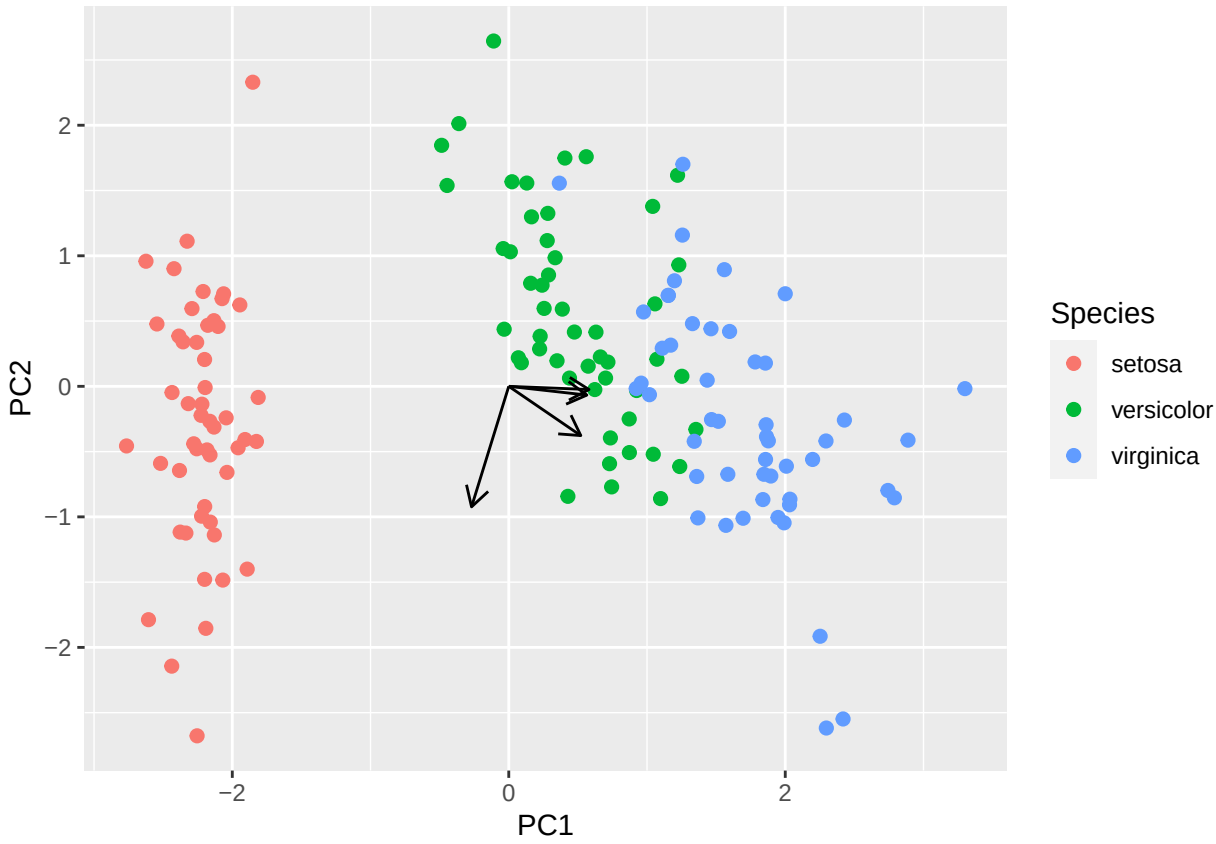
Ahora vamos a constuir la gráfica por capas, primero los individuales u observaciones:

```
ggplot() +
geom_point(data = individuales, aes(x=PC1, y=PC2, color =Species), size=2)
```



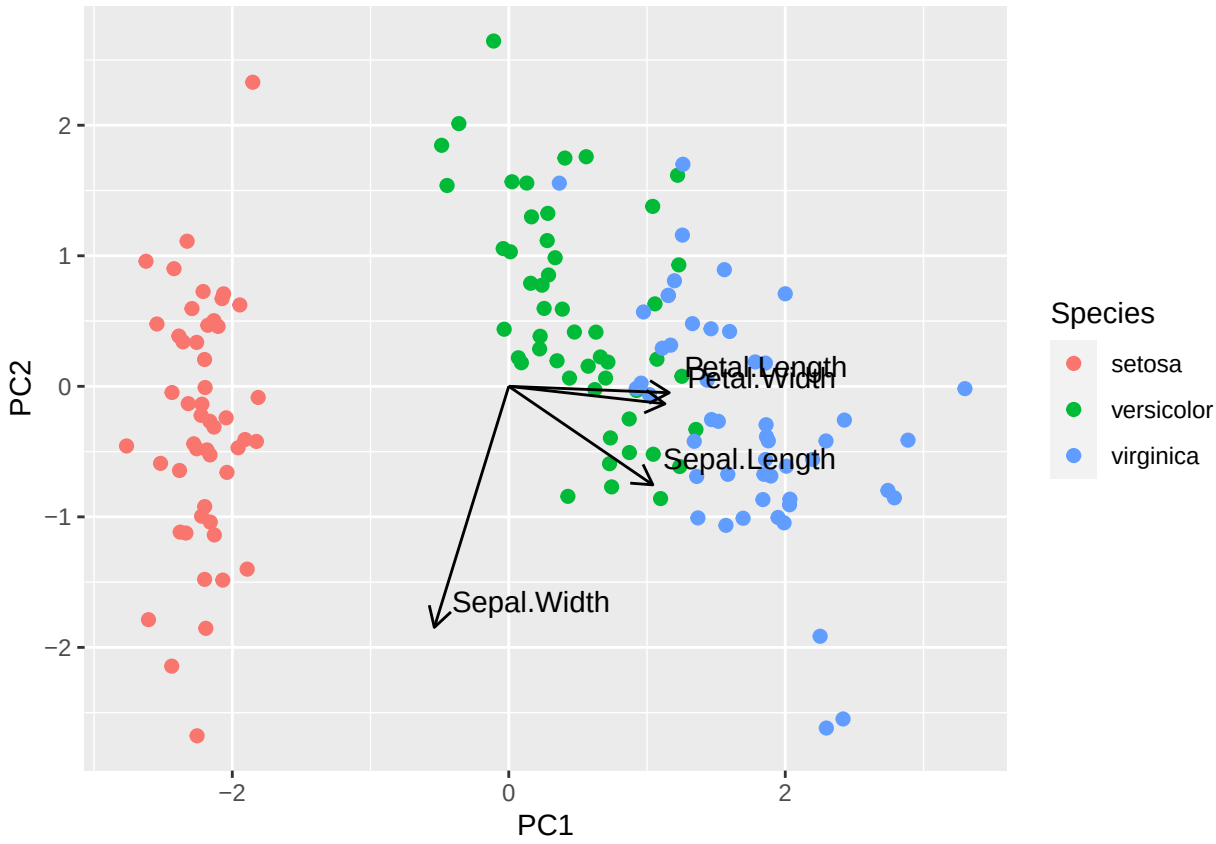
Siguiente agregramos las variables en forma de vectores o segmentos:

```
ggplot() +
geom_point(data = individuales, aes(x=PC1, y=PC2, color =Species), size=2)+
geom_segment(data=vars, aes(x=0, xend=PC1, y=0, yend=PC2),arrow = arrow(length = unit(0.3,"cm")))
```



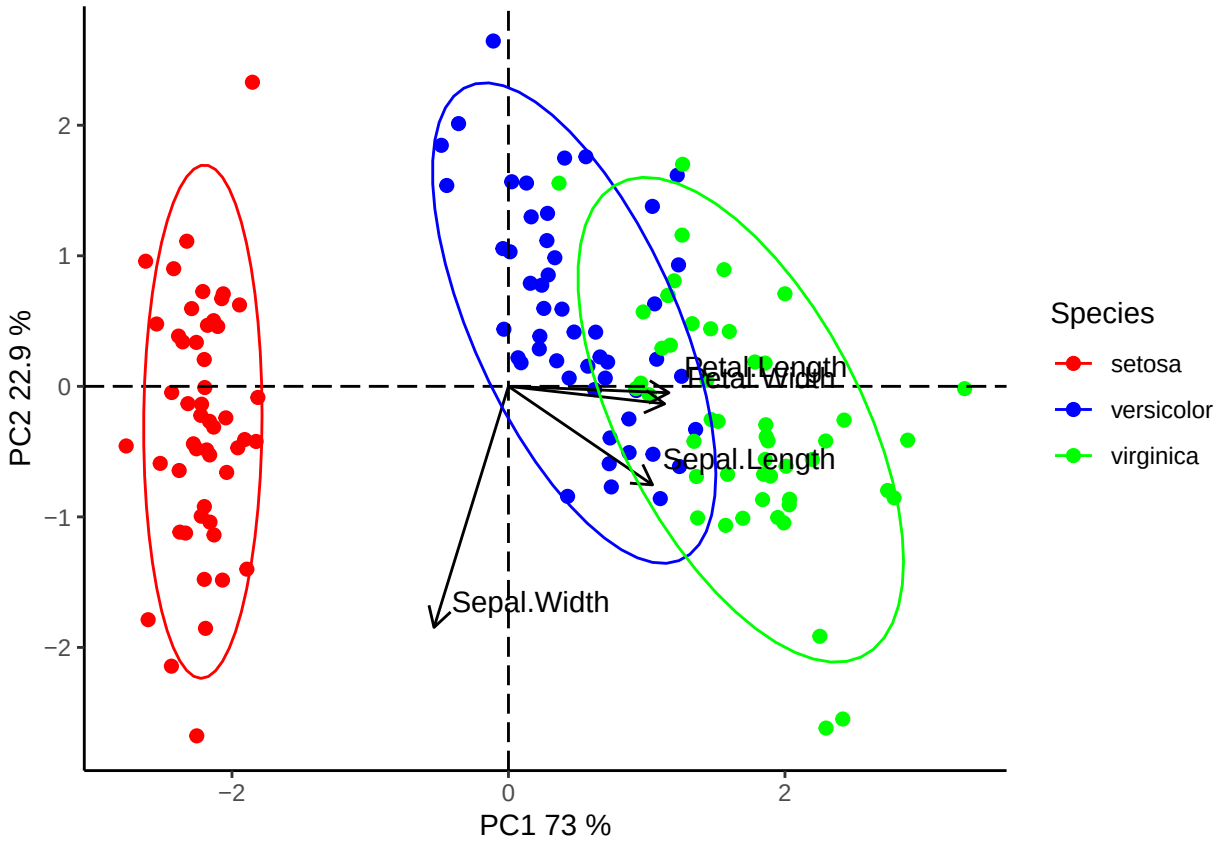
La última capa es el texto de las variables:

```
ggplot() +
geom_point(data = individuales, aes(x=PC1, y=PC2, color =Species), size=2)+
geom_segment(data=vars %>% mutate(PC1=PC1*2, PC2=PC2*2), aes(x=0, xend=PC1, y=0, yend=PC2),
             arrow = arrow(length = unit(0.3,"cm")))+
  geom_text(data=vars %>% mutate(PC1=PC1*2, PC2=PC2*2), aes(x=PC1, y=PC2, label= vars),
            nudge_x = 0.7, nudge_y = 0.2)
```



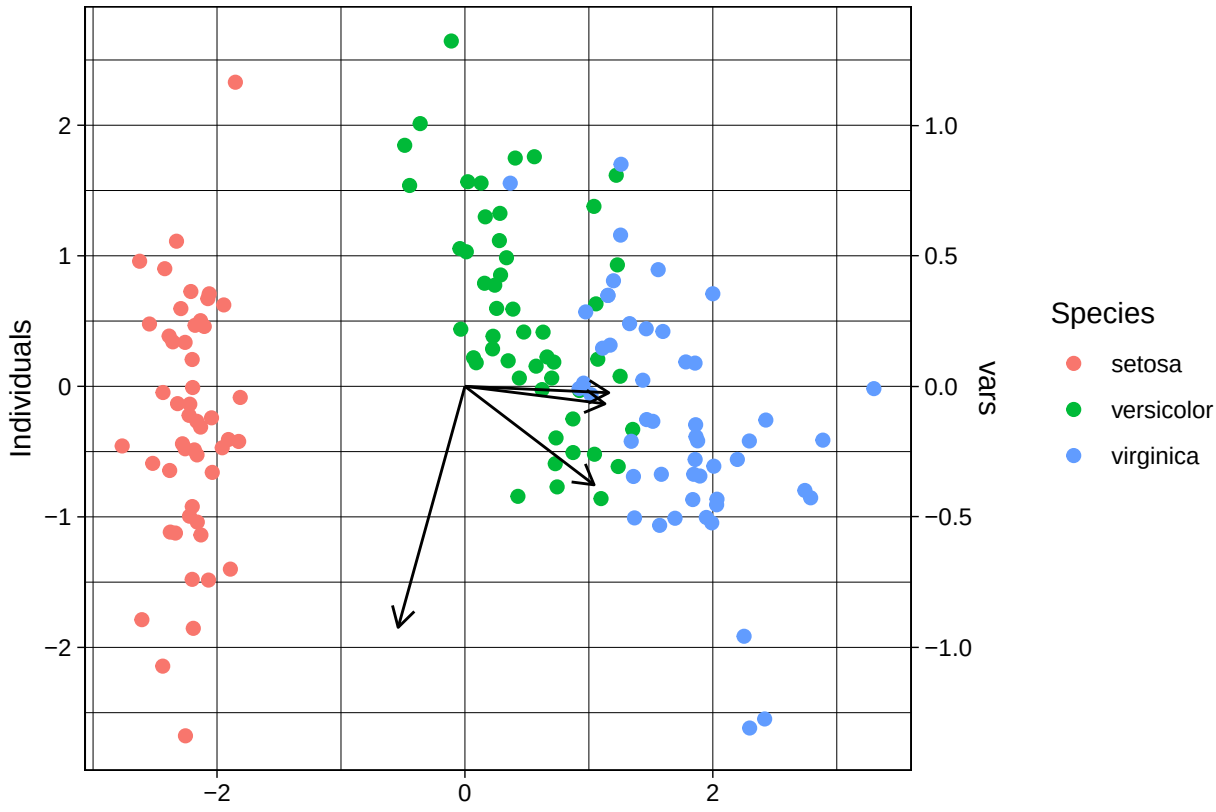
Lo último es lo de más de formato:

```
ggplot() +
geom_point(data = individuales, aes(x=PC1, y=PC2, color =Species), size=2)+
geom_segment(data=vars %>% mutate(PC1=PC1*2, PC2=PC2*2), aes(x=0, xend=PC1, y=0, yend=PC2),
             arrow = arrow(length = unit(0.3,"cm")))+
  geom_text(data=vars %>% mutate(PC1=PC1*2, PC2=PC2*2), aes(x=PC1, y=PC2, label= vars),
            nudge_x = 0.7, nudge_y = 0.2)+
  geom_vline(xintercept = 0, linetype = 5) +
  geom_hline(yintercept = 0, linetype = 5) +theme_classic()+
    stat_ellipse(data = individuales, aes(x=PC1, y=PC2, color =Species))+
  scale_color_manual(values = c("red", "blue", "green"))+
  ylab(PC2_label)+xlab(PC1_label)
```



Si queremos entonces colocar dos axis para cada grupo de datos, sería:

```
coef<- 2
ggplot() +
geom_point(data = individuales, aes(x=PC1, y=PC2, color =Species), size=2)+
geom_segment(data=vars[-1]*coef, aes(x=0, xend=PC1, y=0, yend=PC2),arrow = arrow(length = unit(0.3,"cm")
  scale_y_continuous(name = "Individuals", sec.axis = sec_axis(~./coef, name="vars"))+theme_linedraw()+
```


Individuals
vars
2
1
0
−1
−2
1.0
0.5
0.0
−0.5
−1.0
−2
0
2
Species
setosa
versicolor
virginica