TEMA 1. INTRODUCCIÓN Y DESCRIPTIVA

dat=read.table('tiempoviaje.txt', header=T) Leer archivo de datos
names(dat) - head(dat) Para ver los datos

hist(dat$tiempo, breaks=20) Hacer histograma de una de las variables. Breaks es para la numeración del eje x
boxplot(dat$tiempo, horizontal=T) Boxplot horizontal de una de las variables
grid() Para los gráficos tengan cuadricula

summary(dat) Medias y cuartiles
cor(dat[, c(2,4)]) Matriz de correlación de las columnas 2 y 4 de dat
cor(cbind(dat$timepo, dat$hermanos)) Matriz de correlación entre tiempo y hermanos

plot(tiempo~hermanos, data=dat) Gráfico de dispersión de tiempo en función de hermanos

TEMA 2. MODELOS DE REGRESIÓN

Borrar variables: rm(list=ls())

Leer datos: dat=read.table('coches.txt', header = T)

Variables cualitativas: dat$origin = factor(dat$origin , label=c('USA','JAP', 'EU'))
Coge como referencia el primer tipo (USA). Cuidado, a veces las bases de datos tiene mas de una columna cualitativa (por ejemplo, fev.txt tiene el sexo y si es fumador o no fumador). Para coger otra referencia se hace así:

Coger otra referencia →
dat$origen=relevel(dat$origen, ref='JAP')

Nueva variable:
Para crear una nueva columna en dat (a veces lo piden, por ejemplo: "el consumo es 234 entre la potencia", y una de las columnas es la potencia) se hace así:
dat$cons=f(dat$pot)
f() no es un comando de R, si no cualquier función que queramos, por ejemplo: "dat$cons=234/dat$pot"

Hacer una selección de los datos: Si me piden hacer un modelo de regresión de muchas variables pero no de todas, o todas menos una, conviene hacer una selección de las columnas que me interesan:
selec=dat[, c("columnas aqui con comas")]
Conviene comprobar después que tenemos las columnas correctas con names(selec).
Tambien puede ser que me pidan hacer una selección en función del valor de alguna de las variables (muy tipico el que te digan, coge solo los valores de las mujeres y estudia las columnas 2,5,6 y 7). En este caso se haría así:
selec=dat[dat$sexo='m', c(2,5,6,7)]
También se puede hacer en dos pasos, seleccionando primero lo del sexo, y luego las columnas que queramos.

Resumen de hacer una selección →
En resumen, el comando selec=dat[,] lo que hace es coger de tu matriz las filas que le digas antes de la coma, y las columnas que le digas después de la coma. Puedes decirle que coja líneas en concreto que quieras, o que haga una selección con comandos como "==" o "<=".

Modelo de regresión: mod=lm(~ , data=)
mod=lm((variable a estudiar)~(regresor(es)), data=(de donde cogemos los datos, cuidado con el nombre o si hemos hecho una selección))
Para los regresores, se pueden escribir varios regresores escribiendo un + entre ellos, o si escribimos un punto(.) significa coger todos. Si escribimos un punto menos el nombre de un regresor, es coger todos menos ese (. - sexo)

Transformaciones →
También se pueden hacer transformaciones en esta misma función, por ejemplo:
mod=lm(log()~potencia, data=)
Se pueden hacer transformaciones en la variable de estudio o en los regresores.

Summary e interpretación: summary(mod)

⇓

Con esto podemos escribir la ec. de regresión:

$\hat{y}_i = x_{i1}\hat{\beta}_1 + \dots + x_{in}\hat{\beta}_n + e_i$

En la columna de 'Estimate' da las β en orden
En la columna de 'Std.Error' tenemos las desviaciones típicas de cada β. Como eso nos da la ¨presición¨, nos interesa que sean bajas
En la columna 'Pr(t)' si los numeros son menores que 0.05, el parámetro es significativo con un 95% de confianza. Si son menores que 0.01, es significativo con un 99% de confianza y etc.
El 'Residual standart error' nos dice el error del modelo (Sr gorrito)
El 'R-squared' nos da el porcentaje de variabilidad (cuanto más grande, mejor el estudio de RLS)
El 'Adjusted R-squared' es el R-squared para RL Múltiple (más de un regresor)
El p-value es el area a la derecha/izquierda del F-stadistic. Si es menor que 0.05, hay algún parámetro es significativo al 95% de confianza.
abline(m1, lwd=3, col='blue') #Dibuja la recta sobre el gráfico

Interpretación de las variables cualitativas →
Si tenemos variables cualitativas, el valor de la ß en realidad es el α, por lo que en realidad la información que nos da es si existen diferencias significativas entre la variable de referencia y la que nos sale en la tabla. Si el Pr(t) es menor a la confianza que queremos, es significativo y por lo tanto hay diferencias. Cuidado cuando nos pregunten si hay diferencias entre dos parámetros que tenemos en la tabla, ahí tendremos que coger uno de ellos de referencia (relevel) y volver a hacer un modelo y summary

Confianza de las β's: confint(mod)

Intervalos de confianza de las β's del modelo. El intervalo estandar es con un 95%, para otros niveles hacemos:

confint(mod, level=0.99)

Predicciones y nuevas observaciones: nuevo=data.frame(,)

En los espacios del data.frame hay que poner los nombres de los nuevos datos recogidos y su valor, por ejemplo: nuevo=data.frame(origin='USA', weight= 2454)

Para predicciones e intervalos de confianza se puede hacer metiendo los datos de la nueva observación en la ecuación de regresión que hemos conseguido con el summary o hacerlo con comandos de R.

predict(mod, nuevo, interval='prediction')

Si ponemos ahí 'prediction' o no ponemos nada, nos da el intervalo para la nueva observación.

predict(mod, nuevo, interval='confidence')

Si ponemos ahí 'confidence' , nos lo da el intervalo para la media.

Si queremos un nivel de confianza diferente a $\alpha=0.05$, ponemos una coma dentro y level=1-α, por ejemplo: predict(mod, nuevo, level=0.99)

TEMA 3. COMPONENTES PRINCIPALES

Borrar variables: rm(list=ls())

Leer datos: dat=read.table('coches.txt', header = T)
dat=read.csv('coches.csv', header=T)

Consideraciones: Es típico que te digan que no hay que coger todos los datos de la tabla. Por ejemplo, puede pasar y es normal que la primera columna o las dos primera sean variables cualitativas, ahí por ejemplo se haría: selec=dat[, 2:12] o selec=dat[, 3:12] para seleccionar solo las numéricas, es recomendable luego dar nombre a las filas de la tabla para localizar mejor los datos en el biplot de la siguiente manera:
row.names(selec)= dat$(columna de dat que queremos poner de nombres)

Conviene comprobar que los datos están correlados, porque puede ser que haya que hacerles transformaciones para que el estudio tenga sentido, o que no estén en las unidades correctas. Para ello cargamos library(corrplot) y hacemos corrplot(cor(basededatos), method='ellipse'). Si no hay buena correlación cambiar los datos, ordenar las columnas, etc, y volver a hacer corrplot para ver si ahora están mejor

Mod. de comp. prin de R: modR=princomp(dat, cor=T)
Este método (estudio de componentes principales con la función de R) es recomendable usarlo solo para el biplot, ya que tiene los nombres de algunas cosas cambiadas

biplot(modR)
Los ejes son las comp. prin. y los elementos están colocados con sus respectivas puntuaciones

Mod. de comp. prin (prinfact): source('prinfact.R')
Ojo, hay que tener la función prinfact en el directorio de trabajo.

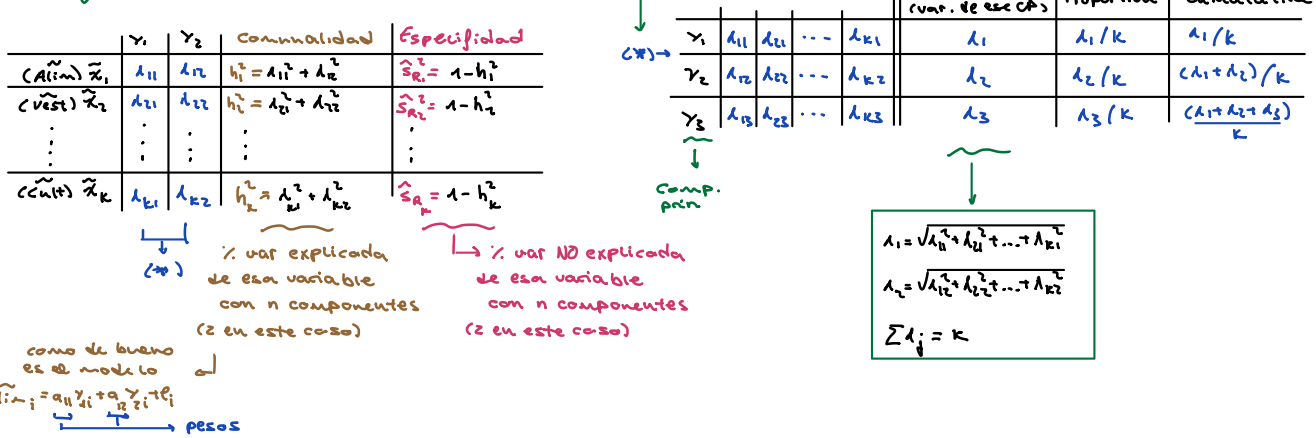
mod=prinfact(,)
Antes de la coma, hay que poner la base de datos de la que queremos hacer el estudio (cuidado si hemos hecho transformaciones o selecciones de los datos). Después de la coma, el número de componentes que queremos utilizar.

mod$loadings
Esto me da las cargas de las componentes, es decir, las λ11, λ12, λ21, λ22..., que son los coeficientes de correlación. También me da las comunalidades (% de variabilidad de cada variable explicado de cada variable) y las especifidades (% de variabilidad NO explicada de cada variable)

mod$variances
Las varianzas son las λ1, λ2... osea, las importancias. La Proportion es el % de variabilidad explicada por cada componente. La Cumulative es el porcentaje de variabilidad explicada acumulada hasta ese componente

mod$scores
Los scores son las coordenadas en el biplot de cada observación

mod$eig
Me da la matriz de cargas diagonalizada, y su correspondiente matriz de autovectores, por lo que las λ que da son las λ1,λ2... (varianzas de cada CP)



TEMA 4. ANÁLISIS DISCRIMINANTE

Borrar variables: rm(list=ls())

Leer datos: dat=read.table('coches.txt', header = T)

Normalmente la variable respuesta hay que pasarla a factor

Cargar paquetes → install.packages('MASS'); library(MASS)
install.packages('multiUS'); library(multiUS)
install.packages('klaR'); library(klaR)

Modelo de análisis discriminante: mod=lda(Variable de estudio~Regresores, data=base de datos)
Poner entre paréntesis para ver el "summary"
-Group means: coordenadas de los centroides
-Coefficients of linear discriminants: coeficiente de cada regresor en la funcion discriminante. NO se pueden interpretar porque no están estandarizados.
Ec. discriminante: Yi=a0+reg_1*coef_reg_1+reg_2*coef_reg_2+ … + reg_n*coef_reg_n
a0=-sum(mod$scaling*colMeans(base de datos[, columnas que usamos como regresores]))

Función discriminante:

$$Y_i = a_0 + a_1 x_{1i} + a_2 x_{2i} + \dots + a_k x_{ki}$$

(pesos al)

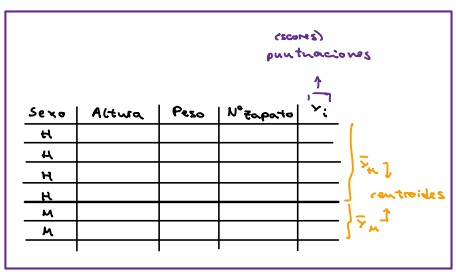
mod$prior Porcentaje de muestras en cada clase
mod$counts Número de observaciones en cada clase
mod$means Medias de cada clase
mod$scaling Pesos de la funcion discriminante (SIN estandarizar)

Como de bueno es el modelo: pred=predict(mod) Predicciones de la muestra
pred$class Clasificación prevista
pred$posterior Como de seguro está el modelo de lo que te ha dicho (probabilidad de pertenecer a cada clase)
pred$x Puntuaciones (scores) previstas

table(prevision=pred$class, real=dat$variable de estudio) Tabla de confusión. El R^2, o cómo de bueno es el modelo serán los daos bien clasificados entre el número total de datos

Para hacer el modelo estandarizado: mod=ldaPlus(x= dat['Regresores'] o dat$'Regresor', grouping=dat$'variable de estudio')
Poner entre paréntesis para ver el "summary"

plotMeans(x= dat['Regresores'] o dat$'Regresor', by =dat$'variable de estudio', xleg='topright') Es el grafico de medias e intervalos de confianza para cada una de las variables. Si se solapan no hay diferencias significativas.



mod$standCoefWithin Coeficientes de la f.discr. estandarizados (se pueden interpretar, cuanto mayor el valor absoluto del peso, más influencia tiene, el signo del peso me dice si influye positiva o negativamente en la armable de estudio)

plot(mod) Estos dos gráficos están "solapados" sobre la función discriminante
Cuanto más aumente una de las variables que influyen negativamente, te vas a ir más hacia la izquierda en el "gráfico total". Básicamente, con el plot y el mod$standCoefWithin podemos interpretar casi toda la información sobre mi f.discr.

mod$centroids Centroides (media de cada grupo)
Un score positivo NO significa que esté en el grupo positivo, lo que hay que medir es la distancia a los centroides -> Del centroide que esté más cerca, es el grupo al que más probable es pertenecer.

mod$sigTest Para comprobar si hay diferencias entre grupos
Si el pvalor es <0.05 (o el nivel de significación que queramos), -> hay diferencias significativas entre grupos

mod$class
orgTab -> matriz de confusion en valores absolutos
perTab -> matriz de confusion en porcentajes
orgPer -> nivel de acierto en la clasificiación
Mirar bien cuáles son las variables originales y cuáles las predicciones

mod$classCV Es lo mismo que mod$class, pero te dice como de bueno es de verdad el modelo: realiza el modelo solo con unos cuantos de los datos y guarda otros para el chequeo de cómo de bueno es (Validación Cruzada)

Simplificación del modelo: mod=greedy.wilks('Variable de clasificación'~ 'Regresor/es', data=Base de datos)
(mod) Esto nos dice cuánto se puede simplificar el modelo, nos da los regresores que son significativos

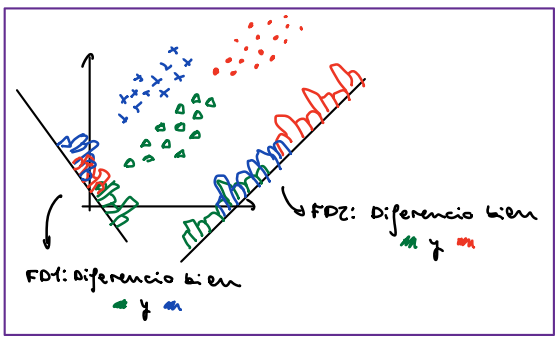
mod1=lda(mod$formula, data=Base de datos)
Modelo simplificado, (mod1) para summary

Para k grupos: mod=lda(Variable de clasificación ~Regresor/es)
Con (mod) me aparecerán los pesos de mis dos funciones discriminantes. Tendremos tantas funciones discriminantes como (tipos-1)
a0=-colMeans(dat[regresores]%*%m1$scaling) a0 de las dos funciones discriminantes. Si solo tenemos un regresor, no hay que poner colMeans(), solo mean()

Dibujo de las puntuaciones:
plot(mod, dimen=1) El plot sobre la FD1
plot(mod, dimen=2, col=as.integer(dat$Species)) Plot en 2D con las dos primeras f.discr

pred=predict(mod)
pred$class Clasificación prevista
pred$x Puntuaciones (coordenadas en el gráfico de dos dimensiones)
pred$posterior Probabilidad de que cada observación pertenezca a un grupo

Con k grupos, se pueden utilizar los comandos anteriormente explicados para el modelo con los coeficientes estandarizados y para la simplificación del modelo a través de eliminar los regresores no significativos, sin cambiar nada, solo hay que tener en cuenta que para todo te saldrán k-1 soluciones.



TEMA 5. ANÁLISIS CLUSTER

Borrar variables: rm(list=ls())

Leer datos: dat=read.table('coches.txt', header = T)

Instalar paquetes:
install.packages('factoextra'); library(factoextra)
install.packages('cluster'); library(cluster)
install.packages('car'); library(car)

Cluster jerárquicos: (dendrogramas)

dat1=scale(dat) Estandarizar las variables (no es obligatorio, lo dirá el enunciado). Por supuesto se puede hacer scale() de una selección de los datos metiendo cuáles quieras.
Se recomienda dar nombre a las variables para poder leerlas mejor en el dendrograma y en el gráfico de comp.prin

mat_dist=dist(Base de datos, method='euclidean') Matriz de distancias
Donde pone euclidean hay que poner el tipo de distancia que queramos utilizar

```
par(mfrow = c(2,2))
boxplot(dat$ALIMENT ~ grupos,
horizontal = T, col = rainbow(4))
boxplot(dat$VESTIDO ~ grupos,
horizontal = T, col = rainbow(4))
boxplot(dat$CULTURA ~ grupos,
horizontal = T, col = rainbow(4))
boxplot(dat$VIVIENDA ~ grupos,
horizontal = T, col = rainbow(4)
```

Ejemplo de boxplot de cada variable separados por grupos → para jerárquico

h1=hclust(mat_dist, method='ward.D2') Modelo de clustering
ward.D2 es el método de encadenamiento. Otros métodos son single, centroid, complete, average, ward.D, ward.D2
plot(h1) Dendrograma
plot(h1, hang=-1) Para que las observaciones queden al mismo nivel
rect.hclust(h1, k=n) Para que en el plot de h1 se vean rodeados los n grupos

fit=eclust(base de datos, "hclust", k=n, graph=T, hc_method='ward.D2')
fviz_dend(h1, k=n, horiz=T)
Este último comando me hace un dendrograma con colores (mejor para visualizarlo). Los argumentos son mi base de datos, el numero de ramas que queremos y el método de encadenamiento.

grupos=cutree(h1, k=n) Para hacer un nuevo modelo con n ramas. Si hago print, me dirá a qué grupo pertenece cada observación
table(grupos) Número de observaciones en cada grupo

row.names(dat[grupos==nombre/valor del grupo]) Observaciones pertenecientes al grupo seleccionado
colMeans(dat[grupos== nombre/valor del grupo]) Medias del grupo seleccionado (centroides)

para k-means

Representar con comp. principales →

fviz_cluster(list(data=base de datos, cluster=grupos)) Componentes principales con clustering (jerárquico)
fviz_cluser(list(data=base de datos, cluster=k$cluster)) Componentes principales con clustering (k-means)
clusplot(base de datos, grupos, color=T, shade=T, labels=2, lines=0) Esto hace lo mismo, pero más cutre

```
par(mfrow = c(3,2))
for(i in 1:6){
  boxplot(dat[,i] ~
k3$cluster, horizontal =
T, col = rainbow(4))
}
```

scatterplotMatrix(dat[], groups=grupos) Una matriz enorme de gráficas plot de todos los datos 2 a 2 pero coloreados en funcion de esos grupos.

Cluster no-jerárquicos: (k-means)

conveniente hacer un set.seed() para lo que es aleatorio sea igual para todos
k=kmeans(base de datos, centers= n, nstart= m) n es el numero de cluster que quiero, y m cuantas veces hago el proceso (de las m me quedaré con la que tenga menor varianza interna)

k$cluster Te dice en que cluster está cada observación
print(k$centers, 2) Centroides de cada cluster con 2 decimales sólo
table(k$cluster) Número de observaciones en cada cluster

N° óptimo de clusters →

¿Número óptimo de clusters?
fviz_nbclust(base de datos, kmeans, method="wss") Esto te da un gráfico de codo, cuando cambie poco el eje y al avanzar en el eje x, cogemos ese número de clusters

Comparar jerárquico: con k-means

table(dendrograma=grupos, kmeans=k$cluster) Viene bien antes de hacer la table poner nombres a los grupos en los modelos de ambos métodos, para facilitar la visualización, y no da lugar a errores de interpretación

Ejemplo de pregunta: indique las observaciones que el modelo jerárquico ha clasificado como flav.alto y el de kmeans como flav.alto.
which(namesk3=='flav.medio' & n3=='flav.alto')

TEMA 6. ÁRBOLES DE REGRESIÓN

Borrar variables: rm(list=ls())

Leer datos: dat=read.table('coches.txt', header = T)

Instalar paquetes:
install.packages('rpart.plot'); library(rpart.plot)
install.packages('plotmo'); library(plotmo)
install.packages('rpart'); library(rpart)

Modelo de árbol:
mod=rpart(variable de estudio~regresor/es, data=base de datos) Modelo de árbol de regresión (variable cuantitativa)/ modelo de árbol de clasificación (variable cualitativa PASAR A FACTOR ANTES)
rpart,plot(mod) Plot del árbol

Probar nueva observación:
nueva=data.frame(var1= , var2= , ...) Nueva observación
predict(mod, nueva)

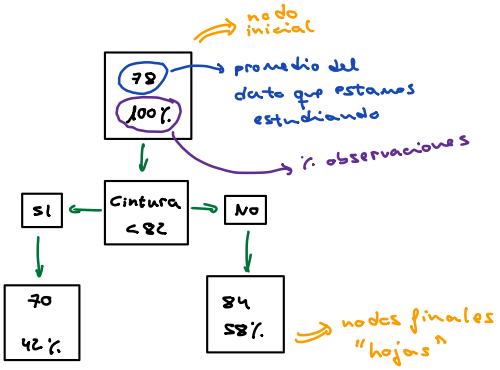
Podar el árbol:
plotmo(mod, degree1=F, degree=2) Te coge los dos regresores que el modelo identifica como más importantes y te hace lo correspondiente a un "plano de regresión" pero en árbol de regresión, donde los datos no tienen que tener una relacion de linealidad
plotcp(mod) Gráfico del parámetro de complejidad (nos ayuda a saber con que cp hay que podar el árbol)
printcp(mod)
- 1a col: número de nodos
- 2a col: cuanto aumentamos el R^2 con cada incremento de CP
- 3a col: numero de divisiones
- 4a col: error relativo (-R^2)

mod2=prune(mod, cp=n) Podar el árbol con el cp que le indiques
rpart.plot Plot del nuevo árbol

mod2$where A qué región pertenece cada observación

Validación cruzada:
Recomendable hacer set.seed()
sel=sample(1: n, round(0.8*n), replace=F) n es el número de observaciones de mi muestra de datos.
train=base de datos[sel,] Con estos datos haremos el modelo
test= base de datos[-sel,] Con estos datos probaremos el modelo

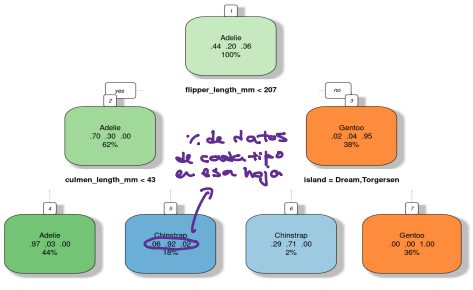
Bondad de ajuste:
y=test$var.estudio
VT=sum((y-mean(y))^2) Variabilidad total
yhat=predict(mod, test) mod tiene que ser un modelo hecho solo con los datos de train
VNE=sum((y-yhat)^2) Variabilidad no explicada (que sea grande es malo)
R^2=1-VNE/VT Que sea grande es bueno



<= CUANTIT.

CUALIT. =>

Árbol pijo: install.packages('rattle'); library(rattle)
fancyRpartPlt(mod, caption=NULL)



TEMA 7. RANDOM FOREST

Borrar variables: rm(list=ls())

Leer datos: dat=read.table('coches.txt', header = T)

Instalar paquetes:
install.packages('randomForest'); library(randomForest)
install.packages('rattle'); library(rattle)
install.packages('rpart'); library(rpart)

Train y test:
Recomendable hacer set.seed()
sel=sample(1: n, round(0.8*n), replace=F) n es el número de observaciones de mi muestra de datos.
train=base de datos[sel,] Con estos datos haremos el modelo
test= base de datos[-sel,] Con estos datos probaremos el modelo

Modelo de random forest:
mod=randomForest(variable~regresor/es, data=train) Modelo de random forest
print(mod) Abajo salen la MSE y la Var.Expl
plot(mod) Gráfico de error frente a número de árboles usados

Importancia:
mod=randomForest(variable~regresor/es, data=train, importance=T)
varImpPlot(mod) Gráficos de IncNodePurity y %IncMSE (para cuantitativas) o de MeanDecreaseArc y MeanDecreaseGini (para cualitativas)