Esenciais USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

O PROGRAMA ESTATÍSTICO R
guía rápida das principais utilidades e funcións

María Isabel Borrajo García, Mercedes Conde Amboage e Rosa María Crujeiras Casais (Departamento de Estatística, Análise Matemática e Optimización)

R é unha linguaxe de programación orientada ao ámbito da estatística. Trátase dunha linguaxe interpretada que se emprega maioritariamente de xeito interactivo, distribuída baixo licenza GNU, e que dispón dunha serie de características no módulo base que se pode ir ampliando por medio de librarías. O proceso de instalación e toda a información relativa ao programa pode atoparse en http://www.r-project.org/.

Para poder empregar calquera das librarías ou paquetes os pasos que cómpre seguir son:

- Instalar a libraría dende o repositorio CRAN (precísase de conexión á Rede), para o que se emprega a orde `install.packages("nome_da_libraría")`.
- Cargar a libraría na sesión de R actual para poder empregar as funcións que contén, isto farase coa sentenza `library(nome_do_paquete)`.

Para calquera dúbida sobre estas ou outras funcións do programa, pode consultarse a súa páxina de axuda tecleando na consola `?nome_da_función ou help(nome_da_función)`.

O programa estatístico R dispón de diversas interfaces gráficas máis amigables para as persoas usuarias non expertas. Unha das máis habitualmente empregadas é RStudio, que pode obterse de xeito libre dende https://rstudio.com/, outra opción dispoñible é *R Commander*, unha interface incluída dentro do propio R mediante a libraría `Rcmdr`.

BÁSICOS

O primeiro que debemos de saber para manexar R é que a asignación (gardar un valor nunha variable) faise mediante o símbolo `=` ou `<-`, por exemplo, `x=5` ou `x<-5` para gardar o valor 5 nunha variable denominada `x`.

O seguinte que se debe coñecer son as distintas clases de obxectos que R permite manexar:

- `logical`: consta unicamente de dous valores de tipo lóxico que son `TRUE` (denotado tamén por `T`) ou `FALSE` (denotado tamén por `F`).
- `integer`: almacena valores de tipo enteiro (números naturais e opostos, sen decimais).
- `numeric`: almacena valores de tipo real.
- `complex`: almacena valores de tipo complexo (parte real e parte imaxinaria).
- `character`: almacena valores de tipo carácter, é dicir, secuencias de letras ou texto.
- `factor`: almacena variables de tipo categórico, onde as categorías constitúen os niveis do factor.

```
> x
[1] 1 2 3 4 5
>
> A=matrix(1:6,nr=2,nc=3)
> A
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
>
```

A continuación verase unha colección de comandos básicos para traballar cos distintos tipos de obxectos, así como coas diferentes estruturas de datos que se poden manexar en R : `vector`, `matrix` (matriz), `list` (lista), `data.frame` (base de datos),...

Creación de datos

`c(...)`	función para combinar argumentos separados por comas e formar un vector.
`dende:ata`	xera unha secuencia que comeza en `dende` e remata en `ata` con paso unha unidade.
`factor(x,levels=lev)`	crea un obxecto de tipo factor con categorías dadas en `lev`.
`matrix(datos,nrow=n, ncol=m, byrow=F)`	crea unha matriz a partir de datos de dimensión `nxm` onde os elementos se colocan por columnas; colocaranse enchendo filas se o argumento `byrow=T`.
`rbind(x1,x2,...); cbind(x1,x2,...)`	combina os elementos `x1,x2,...` (vector, matriz ou base de datos) concatenándoos por filas (`r`) ou columnas (`c`).
`rep(x,times=v)`	constrúe un vector repetindo `v` veces o elemento `x`, que pode ser un valor, un vector, unha matriz, unha lista,...
`seq(dende,ata,by=p, length=l)`	xera unha secuencia que comeza en `dende` e remata en `ata`, ben con paso `p` ou ben de lonxitude `l`.

BÁSICOS (cont.)

Extraer valores

`x[k]`	accede ao elemento `k`-ésimo do vector `x`.
`x[-k]`	accede a todos os elementos do vector `x` salvo ao `k`-ésimo.
`x[c(1,3,7)]`	accede ao primeiro, terceiro e sétimo elementos do vector `x`.
`x[x<2]`	accede a todos os elementos de `x` que cumpran a condición de seren menores que 2.
`x[i,k]`	accede ao elemento da fila `i` e columna `k` da matriz ou base de datos `x`.
`x[i, ]`	accede á fila `i`-ésima da matriz ou base de datos `x`.
`x[ ,k]`	accede á columna `k`-ésima da matriz ou base de datos `x`.
`x[ ,c(2,4)]`	accede ás columnas segunda e cuarta da matriz ou base de datos `x`.
`x[[k]]`	accede ao elemento `k`-ésimo da lista `x`.
`x[["nome"]]`	accede ao elemento da lista `x` denominado `nome`.
`x$nome`	accede ao elemento da lista ou base de datos denominado `nome`.

Información sobre variables

`class(x)`	devolve o tipo de obxecto que é `x`.
`dim(x)`	devolve a dimensión (nº de filas e nº de columnas) da matriz ou base de datos `x`.
`is.numeric(x)`	devolve cun valor lóxico indicando se o obxecto `x` é (`T`) ou non é (`F`) de tipo numérico. Análogo para `is.character, is.na, is.complex, is.data.frame,...`
`length(x)`	devolve o número de elementos do obxecto `x`.
`nrow(x);ncol(x)`	devolve o número de filas (`nr`) ou columnas (`nc`) da matriz ou base de datos `x`.
`summary(x)`	resume a información contida no obxecto `x`; dependendo da clase de `x` proporciona diferente información.

Selección e manipulación de datos

`choose(x,k)`	calcula as posibles combinacións de coller `k` elementos en `n` repeticións.
`cut(x,breaks=b)`	crea a partir do obxecto `x` un novo obxecto de tipo factor mediante a división de `x` en intervalos (categorías); `b` pode ser o número de intervalos, ou os propios puntos de corte que delimitan os intervalos.
`det(x)`	calcula o determinante da matriz `x`.
`diag(x)`	accede á diagonal da matriz `x`.
`eigen(x)`	calcula os autovalores e autovectores da matriz `x`.
`match(x,y)`	define un vector da mesma lonxitude de `x`, cos elementos de `x` que están en `y` e `NA` (valor co que R codifica os datos ausentes) noutro caso.
`round(x,digits=nd)`	redondea os valores numéricos de `x` ao número de decimais dado por `nd`.
`rowsum(x);colsum(x)`	calcula a suma por filas (`row`) ou columnas (`col`) da matriz ou base de datos `x`.
`rowMeans(x);colMeans(x)`	calcula a media por filas (`row`) ou columnas (`col`) da matriz ou base de datos `x`.
`solve(x)`	calcula a matriz inversa da matriz `x`.
`sort(x, decreasing=F)`	ordena de xeito crecente (ou decrecente se `decreasing=T`) os elementos de `x`. Empregando o argumento `index.return=T` pódese obter o valor da posición dos elementos segundo a súa orde (crecente ou decrecente).
`t(x)`	calcula a matriz trasposta da matriz `x`.
`x%*%y`	calcula o produto escalar ou matricial dos vectores ou matrices `x` e `y`.
`which(x<2)`	determina a/as posición/s dos elementos de `x` que cumpran a condición de ser menores que 2.
`which.min(x);which.max(x)`	determina a posición que ocupa en `x` o seu elemento de menor (`min`) ou maior (`max`) valor.

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}$$

Lectura e escritura de datos

`data(nome)`	carga o conxunto de datos `nome` gardado en R; se non se especifica ningún argumento, lista os conxuntos de datos dispoñibles na sesión actual de R.
`load(...)`	carga datos dende unha área de traballo previamente gardada (en xeral con `save`).
`read.table(file="nome.ext", sep=";",dec="",header=T)`	carga os datos almacenados no arquivo `nome`, de extensión *.ext (.txt, .dat).* Co argumento `sep` determínase o elemento que serve para separar valores no arquivo, con `dec` indícase o símbolo empregado para o separador decimal e `header` é un argumento lóxico que indica se a primeira fila da base de datos contén os nomes das variables (`T`) ou contén os valores do primeiro individuo (`F`). Análogo `read.csv` para extensións *.csv*.

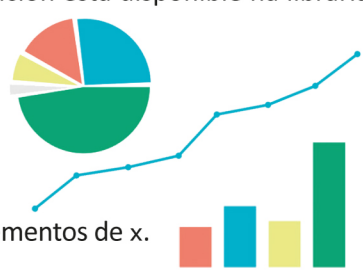
BÁSICOS (cont.)

Lectura e escritura de datos (cont.)

`attach(dat)`	carga no contorno de traballo as variables da base de datos `dat`.
`complete.cases(x,y)`	indica as posicións comúns sen `NA` nos elementos `x` e `y`.
`head(dat)`	amosa as seis primeiras liñas do elemento `dat`.
`names(dat)`	indica os nomes dos elementos almacenados en `dat`.
`save(nome,file=arquivo)`	garda o elemento `nome` en `arquivo`.
`save.image()`	garda todos os elementos que se atopen na memoria da sesión nese momento nun arquivo *.RData*.
`View(dat)`	permite visualizar a base de datos `dat` nun formato de táboa ao estilo dunha folla de cálculo.
`write.table(x, file="nome.ext")`	garda o elemento `x` nun arquivo *.ext (.txt, .csv)* tras convertelo en base de datos.

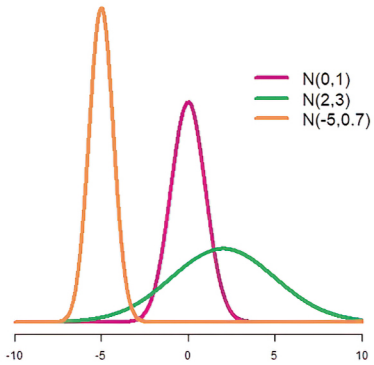
ESTATÍSTICA DESCRITIVA

`addmargins(tab)`	engade a fila e columna dos totais (marxinais) a unha táboa de dobre entrada `tab`.
`barplot(tab)`	representa o diagrama de barras asociado ás frecuencias `tab`, que se poden obter, por exemplo, coa función `table`.
`boxplot(x)`	representa un diagrama de caixa para os valores de `x`.
`boxplot(x~y)`	representa un diagrama de caixa para os valores de `x` agrupados segundo as categorías do factor `y`.
`cov(x,y);cor(x,y)`	calcula a cuasicovarianza (cov) ou correlación (cor) entre os valores de `x` e de `y`.
`cumsum(x);cumprod(x)`	vector no que o elemento `i`-ésimo é a suma (`sum`) ou produto (`prod`) acumulados dos elementos `x[1:i]`.
`describe(x)`	calcula numerosas características dos elementos de `x`: tamaño da mostra, número de valores perdidos, media, cuantís, táboa de frecuencias,... Esta función está dispoñible na libraría `Hmisc`.
`diff(x)`	calcula un vector de lonxitude unha unidade menor que a lonxitude de `x` coas diferenzas iteradas dos elementos de `x`.
`hist(x,breaks=b,freq=T)`	representa un histograma con `b+1` intervalos (ou cos intervalos dados en `b`); se `freq=T` representa as frecuencias absolutas, en caso contrario representa densidades.
`IQR(x)`	calcula o rango intercuartílico (terceiro menos primeiro cuartil) dos elementos de `x`.
`kurtosis(x)`	calcula o coeficiente de curtose para os valores de `x`. Esta función está dispoñible na libraría `moments`.
`max(x)`	calcula o valor máximo dos elementos de `x`.
`mean(x)`	calcula a media da mostra `x`. Engadindo o argumento `trim=p`, calcula a media recortada coa proporción de datos eliminados dada por `p`.
`median(x)`	calcula a mediana da mostra `x`.
`min(x)`	calcula o valor mínimo dos elementos de `x`.
`pie(tab)`	representa o diagrama de sectores asociado ás frecuencias `tab`, que se poden obter, por exemplo, coa función `table`.
`pie3D(tab)`	representa o diagrama de sectores con volume (3D) asociado ás frecuencias `tab`, que se poden obter, por exemplo, coa función `table`.
`proportions(tab,margin=m)`	calcula as proporcións condicionadas da táboa `tab` dadas as marxinais `m`.
`qqplot(x,y)`	representa os cuantís de `x` fronte aos cuantís de `y`.
`quantile(x,probs=p)`	calcula o cuantil dos elementos de `x` definido pola probabilidade en `p`. Tamén é valido que `p` sexa un vector de probabilidades.
`range(x)`	calcula simultaneamente os valores mínimo e máximo dos elementos de `x`.
`skewness(x)`	calcula o coeficiente de asimetría para os valores de `x`. Esta función está dispoñible na libraría `moments`.
`sum(x);prod(x)`	calcula a suma (`sum`) ou produto (`prod`) dos elementos de `x`.
`table(x)`	conta os elementos dos distintos valores de `x`.
`table(x,y)`	realiza unha táboa de dobre entrada dos elementos `x` e `y`.
`unique(x)`	vector cos elementos de `x` eliminando os elementos repetidos.
`var(x);sd(x)`	calcula a cuasivarianza (`var`) ou cuasidesviación típica (`sd`) dos elementos de `x`.
`weighted.mean(x,w)`	calcula a media da mostra `x` ponderada polo vector de pesos `w`.



VARIABLES ALEATORIAS

`dDISTR(x)`



calcula a función de masa de probabilidade (función de densidade) nun punto `x` asociada a unha variable `DISTR` discreta (variable continua). Amais, os argumentos desta función dependen da `DISTR` considerada. As principais variables que podemos empregar con R son:

DISTRIBUCIÓN	DISTR	ARGUMENTOS
Binomial (*n*,*p*)	`binom`	`size=n, prob=p`
Binomial negativa (*n*,*p*)	`nbinom`	`size=n, prob=p`
Poisson (λ)	`pois`	`lambda=λ`
Uniforme (*a*,*b*)	`unif`	`min=a, max=b`
Normal (μ,σ)	`norm`	`mean=μ, sd=σ`
Exponencial (λ)	`exp`	`rate=λ`
Gamma (n,λ)	`gamma`	`shaper=n, rate=λ`
T-Student con *gl* graos de liberdade	`t`	`df=gl`
χ^2 con *gl* graos de liberdade	`chisq`	`df=gl`
F de Snedecor con gl_1 e gl_2 graos de liberdade	`f`	`df1=gl1, df2=gl2`

`density(x)` estima a función de densidade asociada a unha mostra `x` empregando métodos non paramétricos.

`ecdf(x)` calcula a función de distribución empírica asociada a unha mostra `x`.

`fitdistr(x,densfun="NOME")` emprega o método de máxima verosemellanza para axustar os parámetros asociados a unha distribución `NOME` para a mostra `x`. As distintas distribucións compatibles con esta función poden verse en `?fitdistr`. Esta función está dispoñible na libraría `MASS`.

`pDISTR(x)` calcula a función de distribución nun punto `x` asociada a unha variable `DISTR` discreta ou continua. Esta función pódese aplicar para as mesmas distribucións que `dDISTR`.

`qDISTR(τ)` calcula o cuantil de orde `τ` asociado a unha variable `DISTR` discreta ou continua. Esta función pódese aplicar para as mesmas distribucións que `dDISTR`.

`qqPlot(x)` representa os cuantís asociados á mostra `x` fronte aos cuantís dunha distribución normal. Esta función está dispoñible na libraría `car`.

`rDISTR(n)` xera `n` datos asociados a unha variable `DISTR` discreta ou continua. Esta función pódese aplicar para as mesmas distribucións que `dDISTR`.

INFERENCIA ESTATÍSTICA

`ks.test(x,"pDISTR")` realiza un contraste de Kolmogorov-Smirnov para contrastar se a mostra `x` (que procede dunha variable aleatoria continua) segue unha distribución `DISTR`.

`chisq.test(x,y)` realiza un contraste de independencia sobre dúas variables categóricas das que se coñecen as mostras `x` e `y`. Tamén se podería realizar o contraste introducindo simplemente unha táboa de continxencia.

`prop.test`

`(x, n, conf.level=β)` calcula un intervalo de confianza de nivel β para a probabilidade de éxito asociado a un experimento de Bernoulli do que se observaron `n` realizacións das cales `x` foron éxitos. Se `n` e `x` fosen vectores de dimensión 2, calcularía un intervalo de confianza de nivel β para a diferenza das probabilidades de éxito asociadas a dous experimentos de Bernoulli.

`(x, n, p=p, alternative=alt)` realiza un contraste de hipóteses para a probabilidade de éxito asociado a un experimento de Bernoulli do que se observaron `n` realizacións, das cales `x` foron éxitos. O valor da probabilidade de éxito que se quere contrastar será `p` mentres que o tipo de contraste vén determinado polo argumento `alternative`*. Amais, se `n` e `x` fosen vectores de dimensión 2, realizaría un contraste para a diferenza das probabilidades de éxito asociadas a dous experimentos de Bernoulli (neste caso o argumento `p` non se emprega).

`shapiro.test(x)` realiza un contraste de normalidade para unha variable X da que se ten a mostra `x`.

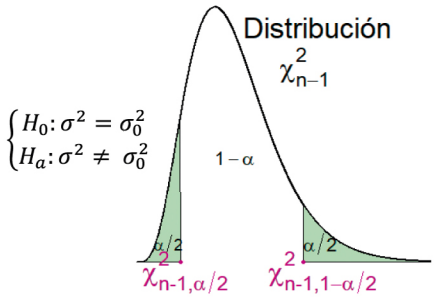
* Nas funcións `prop.test`, `sigma.test`, `t.test`, `var.test` e `z.test` o argumento `alternative` pode ser "`two.sided`" (contraste bilateral), "`less`" (contraste unilateral esquerdo) ou "`greater`" (contraste unilateral dereito).

INFERENCIA ESTATÍSTICA (cont.)

`sigma.test`

`(x, conf.level=β)` — calcula un intervalo de confianza de nivel β para a varianza poboacional dunha distribución normal da que se coñece unha mostra `x`.

`(x, sigmasq = σ², alternative=alt)` — realiza un contrate de hipóteses para a varianza poboacional dunha distribución normal da que se coñece unha mostra `x`. O valor da varianza poboacional que se quere contrastar será σ^2 mentres que o tipo de contraste vén determinado polo argumento `alternative`**.



Esta función está dispoñible na libraría `TeachingDemos`.

`t.test`

`(x, conf.level=β)` — calcula un intervalo de confianza de nivel β para a media poboacional dunha distribución normal da que se coñece unha mostra `x`.

`(x, y, conf.level=β, paired=T)` — calcula un intervalo de confianza de nivel β para a diferenza de medias poboacionais de dúas distribucións normais dependentes (`paired=T`) ou independentes (`paired=F`) das que se coñecen as mostras `x` e `y`. No caso de que `paired=F`, emprégase o argumento `var.equal=T` ou `F` para indicar que as varianzas poboacionais son iguais ou non.

`(x, mu=`μ_0`, alternative=alt)` — realiza un contrate de hipóteses para a media poboacional dunha distribución normal da que se coñece unha mostra `x`. O valor da media poboacional que se quere contrastar será μ_0 mentres que o tipo de contraste vén determinado polo argumento `alternative`**.

`(x, y, mu=`μ_0`, paired=F)` — calcula un contraste de hipóteses para a diferenza de medias poboacionais de dúas distribucións normais dependentes (`paired=T`) ou independentes (`paired=F`) con varianzas coñecidas ou descoñecidas (`var.equal=T` ou `F`) das que se coñecen as mostras `x` e `y`. O valor da diferenza de medias poboacionais que se quere contrastar será μ_0 mentres que o tipo de contraste vén determinado polo argumento `alternative`**.

`var.test`

`(x, y, conf.level=β)` — calcula un intervalo de confianza de nivel β para o cociente das varianzas poboacionais de dúas distribucións normais das que se coñecen as mostras `x` e `y`.

`(x, y, ratio=r, alternative=alt)` — realiza un contrate de hipóteses para o cociente das varianzas poboacionais de dúas distribucións normais das que se coñecen as mostras `x` e `y`. O valor do cociente de varianzas poboacionais que se quere contrastar será `r` mentres que o tipo de contraste vén determinado polo argumento `alternative`**.

`z.test`

`(x, stdev= σ, conf.level=β)` — calcula un intervalo de confianza de nivel β para a media poboacional dunha distribución normal con desviación típica σ coñecida da que se coñece unha mostra `x`.

`(x, mu=`μ_0`, stdev= σ, alternative=alt)` — realiza un contrate de hipóteses para a media poboacional dunha distribución normal con desviación típica σ coñecida da que se coñece unha mostra `x`. O valor da media poboacional que se quere contrastar será μ_0 mentres que o tipo de contraste vén determinado polo argumento `alternative`**.

Esta función está dispoñible na libraría `TeachingDemos`.

** Nas funcións `prop.test`, `sigma.test`, `t.test`, `var.test` e `z.test` o argumento `alternative` pode ser "`two.sided`" (contraste bilateral), "`less`" (contraste unilateral esquerdo) ou "`greater`" (contraste unilateral dereito).

MODELOS DE REGRESIÓN

`aov(y~grupos)`

realiza un contraste de igualdade de medias poboacionais asociadas a unha distribución normal (da que coñecemos unha mostra y) medida en diferentes grupos (determinados polo vector `grupos`).

`boxcox(y~x1+x2+...+xd)`

calcula o parámetro λ óptimo asociado a unha transformación tipo Box-Cox que permite corrixir a falta de cumprimento das hipóteses básicas do modelo de regresión linear múltiple con variable resposta y e variables explicativas `x1,x2,...,xd`. Esta función está dispoñible na libraría `MASS`.

`Box.test(x, lag = γ, type="Ljung-Box"))`

realiza o contraste de Ljung-Box que comproba se as observacións da mostra `x` son non correladas fronte á hipótese alternativa de que exista correlación secuencial de orde γ.

`confint(mod, level = β)`

calcula intervalos de confianza de nivel β asociados aos parámetros do modelo de regresión `mod`.

`glm(y~x1+x2+...+xd, family = NOME)`

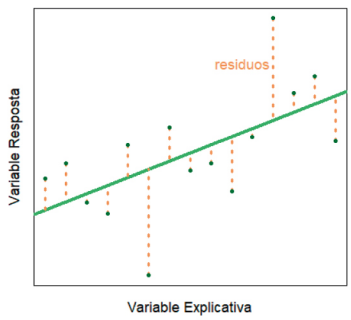
estima un modelo de regresión linear xeneralizado con variable resposta y e variables explicativas `x1,x2,...,xd`. O argumento `family` permite seleccionar a función de enlace adecuada en función da natureza da variable resposta.

`hmctest(mod)`

contrasta a homocedasticidade do modelo de regresión linear múltiple `mod`. Esta función está dispoñible na libraría `lmtest`.

`lm(y~x1+x2+...+xd)`

estima un modelo de regresión linear múltiple con variable resposta y e variables explicativas `x1,x2,...,xd` empregando o método de mínimos cadrados.



`names(mod)`

proporciona os nomes das estimacións asociadas a un modelo de regresión, como os residuos (`mod$residuals`) ou os valores axustados (`mod$fitted.values`).

`nls(y~fun(θ1,θ2,…,θk, start=list((θ1=n1,θ2=n2,…,θk=nk))`

estima un modelo de regresión non linear cuxa expresión vén determinada por `fun` (coñecida) salvo polos parámetros `θ1,θ2,...,θk`. Co argumento `start` introdúcese o punto de arranque do algoritmo iterativo de estimación.

`plot(mod)`

realiza catro representacións gráficas que permiten levar a cabo unha validación e diagnose do modelo de regresión múltiple `mod`.

`predict(mod,data.frame(x1=δ1, x2=δ2,...,xd=δd))`

calcula predicións puntuais da variable resposta dun modelo de regresión `mod` con covariables `x1,x2,...,xd` dada unha nova observación $(\delta_1,\delta_2,\ldots,\delta_d)$ das covariables. Tamén permite calcular intervalos de confianza para a esperanza condicional (`interval="confidence"`) e intervalos de predición (`interval="prediction"`).

`resettest(y ~ x, power=2/3)`

realiza o contraste do modelo linear simple con variable reposta y e variable explicativa x fronte a modelos polinómicos de orde 2/3. Esta función está dispoñible na libraría `lmtest`.

`sm.regression(x,y, model="linear")`

contrasta a linearidade dun modelo de regresión linear múltiple con covariables x (como máximo de dimensión 2) e variable resposta y. Esta función está dispoñible na libraría `sm`.

`summary(mod)`

proporciona unha lista de estatísticos resumo asociados a un modelo de regresión `mod`, como por exemplo, contrastes de significación dos coeficientes, a estimación da desviación típica do erro ou o coeficiente de determinación.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA
XUNTA DE GALICIA
RAGC/ REAL ACADEMIA GALEGA DE CIENCIAS