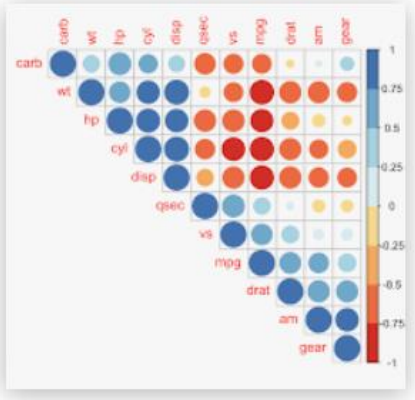
Curso R / Rstudio 2022

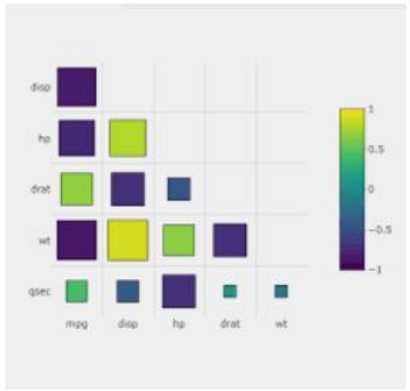
ESTADÍSTICA INFERENCIAL

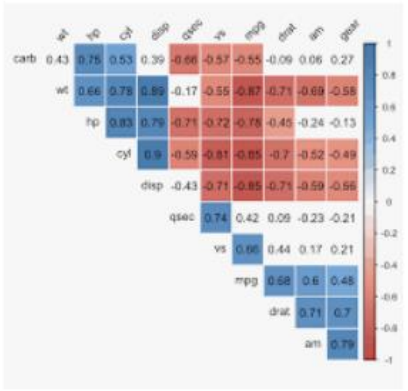
Correlaciones

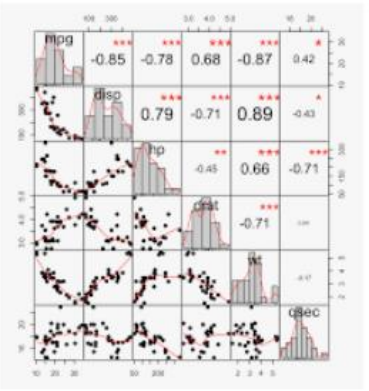
- R tiene paqueterías gráficas muy útiles para ver matrices de correlación

- La función para calcular una matrix de correlación es cor()
  - El argumento method = c(“pearson”, “kendall”, “spearman”) nos permite elegir entre los tres métodos









➢ Vamos al código “Correlaciones.R”

Pruebas de hipótesis

- A veces resulta útil hacer pruebas respecto a las distribuciones de nuestros datos.
- En R se usa la función **t.test()** para comparar las medias de distribuciones. Esto puede ser comparar la media vs cierto dato, o comparar las medias de dos distribuciones.

- ?t.test

```
t.test(x, …)

# S3 method for default
t.test(x, y = NULL,
       alternative = c("two.sided", "less", "greater"),
       mu = 0, paired = FALSE, var.equal = FALSE,
       conf.level = 0.95, …)
```

➢ Vamos al código “Prueba_hipotesis.R”

Más pruebas de hipótesis

#	Name of Test	in R?	Package	Function
1	One Mean T-test	Yes	pwr	pwr.t.test
2	Two Means T-test	Yes	pwr	pwr.t.test
3	Paired T-test	Yes	pwr	pwr.t.test
4	One-way ANOVA	Yes	pwr	pwr.anova.test
5	Single Proportion Test	Yes	pwr	pwr.p.test
6	Two Proportions Test	Yes	pwr	pwr.2p.test
7	Chi-Squared Test	Yes	pwr	pwr.chisq.test
8	Simple Linear Regression	Yes	pwr	pwr.f2.test
9	Multiple Linear Regression	Yes	pwr	pwr.f2.test
10	Correlation	Yes	pwr	pwr.r.test
11	One Mean Wilcoxon Test	Yes*	pwr	pwer.t.test + 15%
12	Mann-Whitney Test	Yes*	pwr	pwer.t.test + 15%
13	Paired Wilcoxon Test	Yes*	pwr	pwer.t.test + 15%
14	Kruskal Wallace Test	Yes*	pwr	pwr.anova.test + 15%
15	Repeated Measures ANOVA	Yes	WebPower	wp.rmanova
16	Multi-way ANOVA (1 Category of interest)	Yes	WebPower	wp.kanova
17	Multi-way ANOVA (>1 Category of interest)	Yes	WebPower	wp.kanova
18	Non-Parametric Regression (Logistic)	Yes	WebPower	wp.logistic
19	Non-Parametric Regression (Poisson)	Yes	WebPower	wp.poisson
20	Multilevel modeling: CRT	Yes	WebPower	wp.crt2arm/wp.crt3arm
21	Multilevel modeling: MRT	Yes	WebPower	wp.mrt2arm/wp.mrt3arm
22	GLMM	Yes^	Simr & lme4	n/a

*-parametric test with non-parametric correction

Regresión lineal

- El caballo de batalla de la econometría aplicada (Applied Econometrics in R).

- La función **lm()** (linear model) es la función más común para estimar estos modelos.
  - El input pueden ser vectores guardados separados, o un data frame
  - El valor (output) es una lista que contiene los elementos necesarios para las distintas pruebas (coeficientes, fitted values, residuales, etc.)
- La función **summary()** devuelve un resumen tal como el output de Eviews o Stata

- Vamos al código “Regresion_lineal.R”
- Vamos al código “Regresion_dinámica.R”
- Vamos al código “Diagnosticos.R”

Selección de variables

- La paquetería Leaps fue diseñada para elegir el modelo con base en la R^2
- En específico usaremos la función `regsubsets()`
  - Esta función devuelve el mejor modelo de tamaño 1, 2, …, k (k especificado por el usuario)
  - Resta comparar con algún estadístico qué tamaño es el mejor

- Otra técnica muy usada para la elección de modelo es el LASSO.
  - Se trata de minimizar el error cuadrático, con la restricción del tamaño de los coeficientes
  - El parámetro lambda es un *tuning parameter*

$$L_{lasso}(\hat{\beta}) = \sum_{i=1}^{n}(y_i - x_i^T\hat{\beta})^2 + \lambda\sum_{j=1}^{m}|\hat{\beta}_j|$$

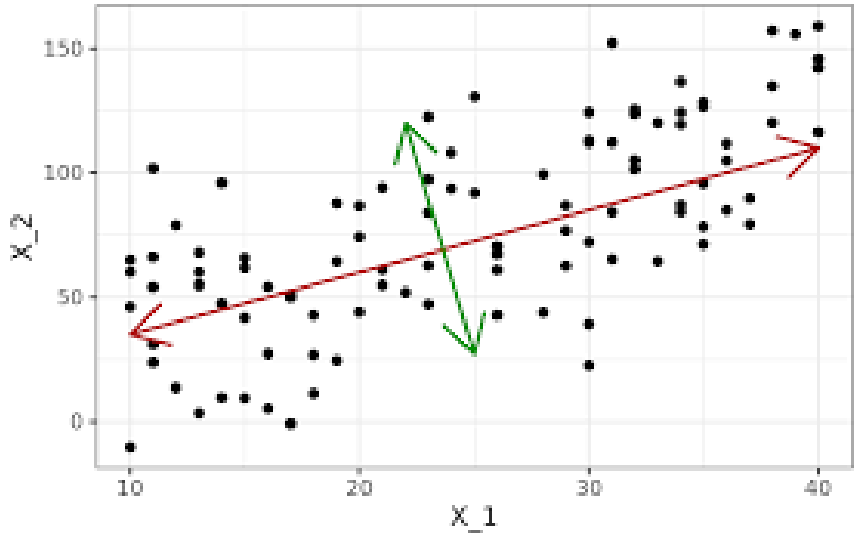
➢ Vamos al código “Subset selection.R”

➢ Vamos al código “LASSO.R”

Análisis de componentes principales (pca)

- El análisis de componentes principales es una técnica de reducción de variables que consiste en explicar un conjunto de variables en un conjunto de combinaciones lineales independientes.

- En R vamos a usar la función `prcomp()`
  - Buscaremos la matriz de rotaciones
  - Decidiremos el número de componentes con base en la proporción de la varianza que explica cada uno.

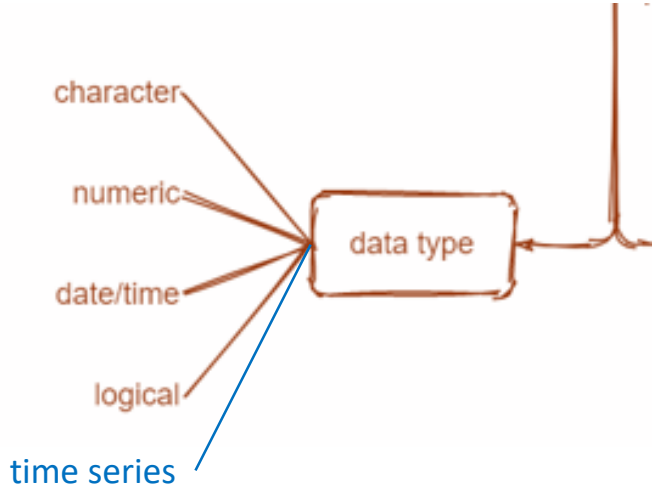


➢ Vamos al código “PCA.R”

Series de tiempo

- Hasta ahora hemos tomado variables de serie de tiempo, pero no le hemos especificado a R que estamos trabajando con ellas, ni hemos hecho análisis típicos de las series de tiempo.
- Veremos un nuevo tipo de variable: time series

```
> ts(X, start = c(1993,1),
      end = c(2021,11), frequency = 12)
```



- Vamos al código “time series.R”
- Vamos al código “Filtros.R”

Desestacionalización

- La paquetería **seasonal** es una interfaz al X-13-ARIMA-SEATS del US Census Bureau. Gracias a esto podemos replicar series desestacionalizadas por agencias oficiales como el INEGI, BEA, DANE, entre otros.
- Por default, muchos de los parámetros de la desestacionalización los encuentra automáticamente, pero podemos especificar explícitamente los que buscamos.

➢ Vamos al código “seasonal.R”

Modelos de series de tiempo: Arima

Function	Package	Description
`ar()`	stats	Fits univariate autoregressions via Yule-Walker, OLS, ML, or Burg's method and unrestricted VARs by Yule-Walker, OLS, or Burg's method. Order selection by AIC possible.
`arima()`	stats	Fits univariate ARIMA models, including seasonal (SARIMA) models, models with covariates (ARIMAX), and subset ARIMA models, by unconditional ML or by CSS.
`auto.arima()`	forecast	Order selection via AIC, BIC, or AICC within a user-defined set of models. Fitting is done via `arima()`.
`StructTS()`	stats	Fits structural time series models: local level, local trend, and basic structural model.

Fuente: Applied Econometrics with R

- Vamos al código "arimas.R"
- Vamos al código "fanplots.R"

Modelos dinámicos lineales (Orden 1)

- Hay varias paqueterías en R para correr modelos dinámicos lineales. Todas corren la paquetería dlm.
- Caminata aleatoria (modelo polinomial de primer orden):

$$\begin{aligned} y_t &= \theta_t + v_t\,, & v_t &\sim N(0,V) \\ \theta_t &= \theta_{t-1} + w_t\,, & w_t &\sim N(0,W) \end{aligned} \Rightarrow F = (1) = G$$

- Vamos a crear el modelo con la función dlm()
  - *dlm(FF = 1, V = 1, GG = 1, W = 0.01, m0 = 0, C0 = 0)*
- Aunque es lo mismo si lo creamos con la función dlmModPoly()
  - *dlmModPoly(**order = 1**, dV = 1, dW = 0.01, m0 = 0, C0 = 0)*
- Una vez que declaras el modelo, y eliges los priors, solo queda aplicar las funciones para utilizarlo.

➢ Vamos al código “DLM_order1.R”

➢ Vamos al código “DLM_KFilter.R”

Modelos dinámicos lineales (Orden 2)

- Modelo polinomial de segundo orden:

$$\begin{aligned} y_t &= \theta_{t,1} + v_t\,, & v_t &\sim N(0,V) \\ \theta_{t,1} &= \theta_{t-1,1} + \theta_{t-1,2} + w_{t,1}\,, & w_{t,1} &\sim N(0,W) \\ \theta_{t,2} &= \theta_{t-1,2} + w_{t,2}\,, & w_{t,2} &\sim N(0,W) \end{aligned} \quad \Rightarrow F = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, G = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

- función dlmModPoly()
  - *dlmModPoly(**order = 2**)*

➢ Vamos al código “DLM_order2.R”

➢ Vamos al código “DLM_Comb.R”