

```
In [1]: %useLatestDescriptors
        %use lets-plot
        %use dataframe

        LetsPlot.getInfo()
```

Out[1]: Lets-Plot Kotlin API v.4.4.2. Frontend: Notebook with dynamically loaded JS. Lets-Plot JS v.4.0.0.

```
In [2]: val redes_df = DataFrame.readCSV("operaciones_2020_2023_porRed.csv", delimiter = ';')
        redes_df.head(5)
```

Out[2]:

meses	red_A	red_B	red_C	red_D	red_E	red_F
01/01/2020	2034	5405482	203916	13686	989346	6614464
01/02/2020	3223	5137937	199639	13758	930327	6284884
01/03/2020	1595	2142050	103024	5691	438661	2691021
01/04/2020	276	469272	22150	735	93581	586014
01/05/2020	587	1254988	54408	2006	271591	1583580

DataFrame: rowCount = 5, columnsCount = 7

```
In [3]: redes_df.describe()
```

Out[3]:

	name	type	count	unique	nulls	top	freq	mean	std	min	median	max
meses	String		43	43	0	01/01/2020	1	null	null	01/01/2020	01/06/2021	01/12/2022
red_A	Int		43	43	0	2034	1	176802,465116	218097,345566	276	99066	689709
red_B	Int		43	43	0	5405482	1	2526603,348837	830398,177067	469272	2556821	5405482
red_C	Int		43	43	0	203916	1	143354,069767	36822,071469	22150	153677	203916
red_D	Int		43	43	0	13686	1	9508,255814	3139,901372	735	10352	14795
red_E	Int		43	43	0	989346	1	657767,627907	190450,651159	93581	630778	989346
red_F	Int		43	43	0	6614464	1	3514035,767442	1064607,585153	586014	3478718	6614464

DataFrame: rowCount = 7, columnsCount = 12

Preparamos los datos en dataframes de trabajo

```
In [4]: // creamos columna de meses
val listMeses : List<Int> = (1..43).toList()
val dfMeses=dfMeses.toDataFrame()
val columnMeses=dfMeses.getColumn(0)
val columnMeses2=dataFrameOf("numeracion_mes")(columnMeses)
columnMeses2.describe()
```

Out[4]:

	name	type	count	unique	nulls	top	freq	mean	std	min	median	max
numeracion_mes		Int	43	43	0	1	1	22,0	12,556539	1	22	43

DataFrame: rowCount = 1, columnsCount = 12

```
In [5]: // seleccionamos columnas del dataframe origen
val colRed_A by redes_df.red_A
val colRed_B by redes_df.red_B
val colRed_C by redes_df.red_C
val colRed_D by redes_df.red_D
val colRed_E by redes_df.red_E
val colRed_F by redes_df.red_F

// create dataframes de trabajo
```

```

val df_Red_A = dataframeOf(columMeses2.getColumn(0), colRed_A)
val df_Red_B = dataframeOf(columMeses2.getColumn(0), colRed_B)
val df_Red_C = dataframeOf(columMeses2.getColumn(0), colRed_C)
val df_Red_D = dataframeOf(columMeses2.getColumn(0), colRed_D)
val df_Red_E = dataframeOf(columMeses2.getColumn(0), colRed_E)
val df_Red_F = dataframeOf(columMeses2.getColumn(0), colRed_F)

// creamos el conjunto de datos como Map para poder usarse en objetos LetsPlot
val map_RedA = df_Red_A.toMap()
val map_RedB = df_Red_B.toMap()
val map_RedC = df_Red_C.toMap()
val map_RedD = df_Red_D.toMap()
val map_RedE = df_Red_E.toMap()
val map_RedF = df_Red_F.toMap()

```

Intentamos ajustar curvas de predicciones

```

In [6]: // establecemos los modelos de predicción
var ppA = letsPlot(map_RedA){ x = "numeracion_mes"; y = "colRed_A"; color="colRed_A" }
var ppB = letsPlot(map_RedB){ x = "numeracion_mes"; y = "colRed_B"; color="colRed_B" }
var ppC = letsPlot(map_RedC){ x = "numeracion_mes"; y = "colRed_C"; color="colRed_C" }
var ppD = letsPlot(map_RedD){ x = "numeracion_mes"; y = "colRed_D"; color="colRed_D" }
var ppE = letsPlot(map_RedE){ x = "numeracion_mes"; y = "colRed_E"; color="colRed_E" }
var ppF = letsPlot(map_RedF){ x = "numeracion_mes"; y = "colRed_F"; color="colRed_F" }

```

```

In [7]: gggrid(
  plots = listOf(
    ppA + ggtitle("Red A") + geomPoint() + themeGrey() +
      statSmooth(method="loess", size=1.0, color="green"){color="colRed_A"},

    ppB + ggtitle("Red B") + geomPoint() + themeGrey() +
      statSmooth(method="loess", size=1.0, color="red"){color="colRed_B"},

    ppC + ggtitle("Red C") + geomPoint() + themeGrey() +
      statSmooth(method="loess", size=1.0, color="orange"){color="colRed_C"},

    ppD + ggtitle("Red D") + geomPoint() + themeGrey() +
      statSmooth(method="loess", size=1.0, color="cyan"){color="colRed_D"},

    ppE + ggtitle("Red E") + geomPoint() + themeGrey() +
      statSmooth(method="loess", size=1.0, color="yellow"){color="colRed_E"},
  )
)

```

```
ppF + ggtitle("Red F") + geomPoint() + themeGrey() +  
  statSmooth(method="loess", size=1.0, color="blue"){color="colRed_F"}  
,  
ncol = 2,  
cellWidth = 400,  
cellHeight = 300,  
vGap = 0,  
fit = true  
)
```

Out[7]:



