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: rowsCount = 5, columnsCount = 7

In [3]:
```
redes_df.columnNames()
// redes_df.red_01_0
// redes_df.getColumn(0)
```

Out[3]:
```
[meses, red_A, red_B, red_C, red_D, red_E, red_F]
```

In [4]:
```
redes_df.describe()
```

Out[4]:

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: rowsCount = 7, columnsCount = 12

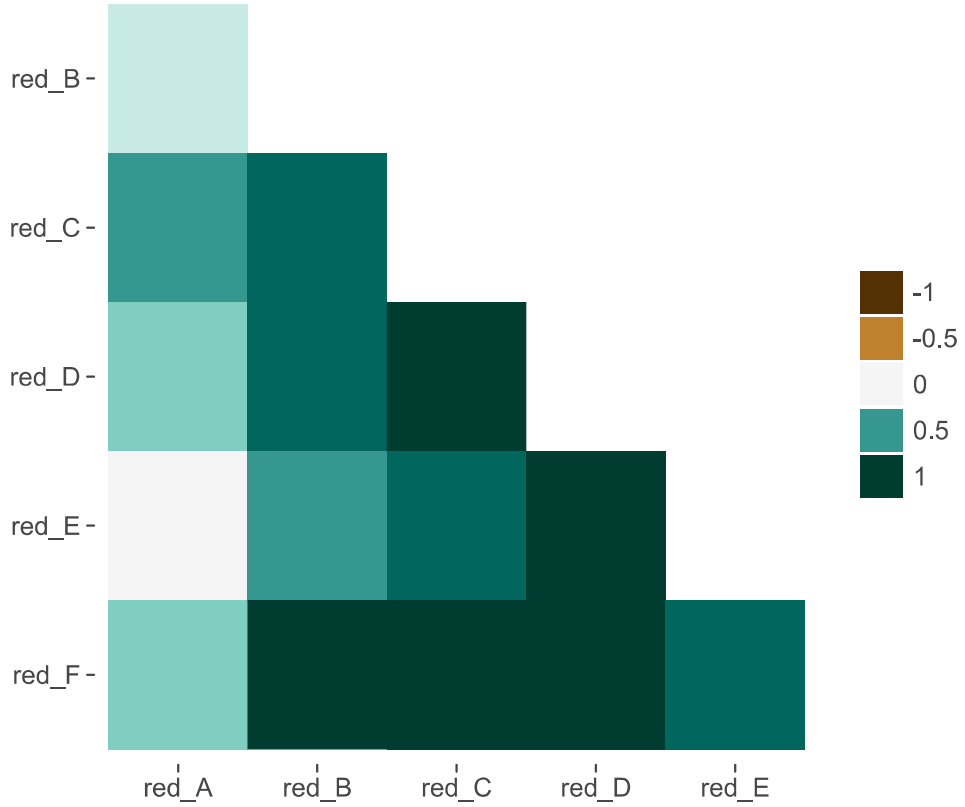
Correlación de varias variables

El parámetro adjustSize, permite cambiar el tamaño de visualización, por ejemplo:

adjustSize = 0.5 reducirá el tamaño del gráfico a la mitad. adjustSize = 2.0 duplicará el tamaño del gráfico. adjustSize = 1.0 mantendrá el tamaño por defecto (sin cambios).

In [5]:
```
CorrPlot(redes_df.toMap(), adjustSize = 2.0)
    .tiles(type="lower")
    .paletteBrBG()
    .build()
```

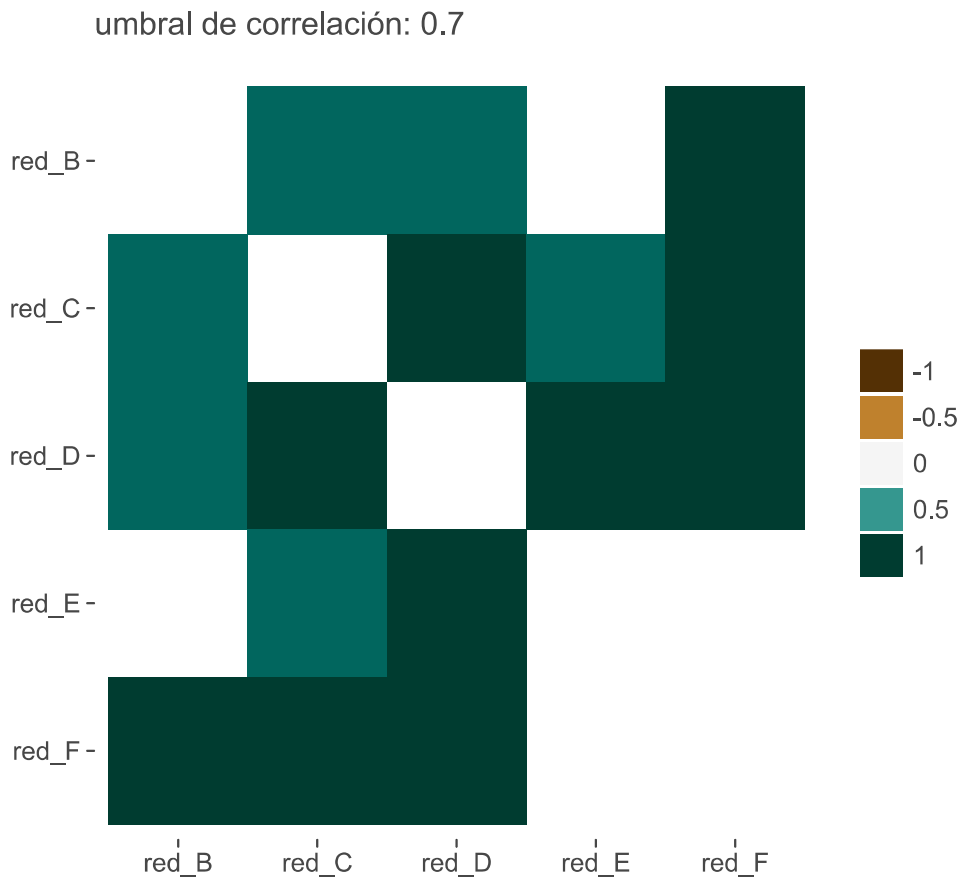
Out[5]:



El parámetro Threshold (umbral) nos permite especificar un nivel de significación, por debajo del cual las variables no se muestran.

In [6]:
```
CorrPlot(redes_df.toMap(), "umbral de correlación: 0.7", threshold = 0.7, adjustSize = 2.0)
    .tiles(type="full", diag=false)
    .paletteBrBG()
    .build()
```

Out[6]:

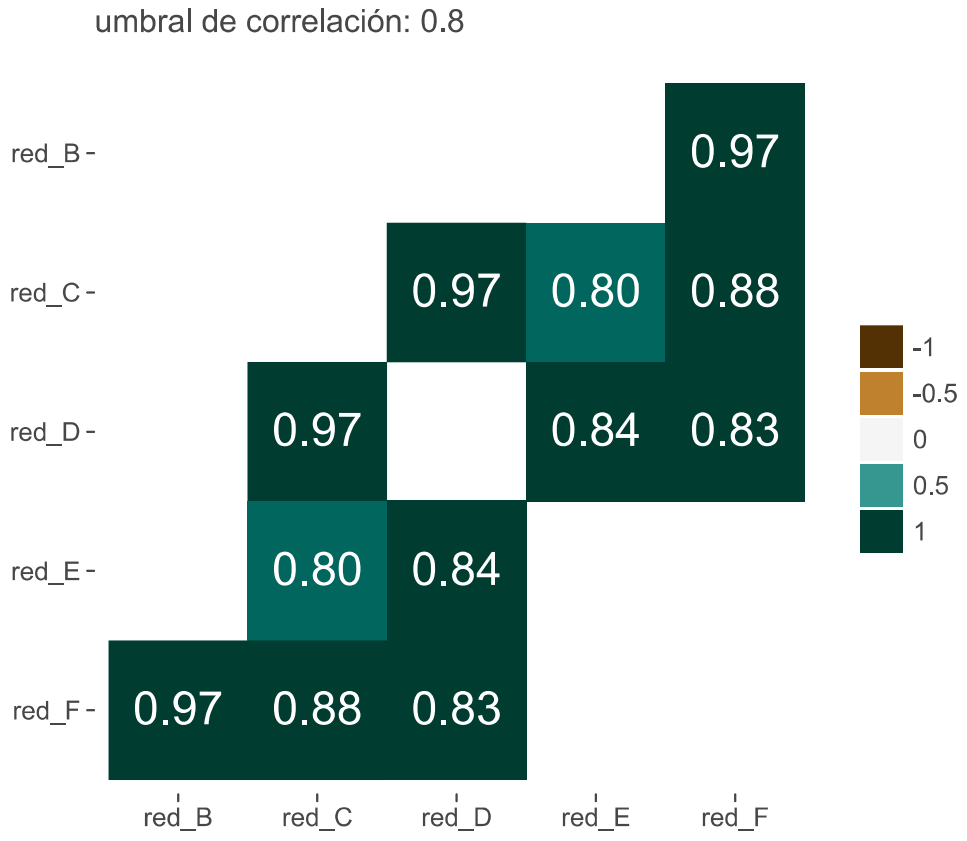


Aumentemos aún más nuestro umbral para ver solo variables altamente correlacionadas.

In [7]:

```
CorrPlot(redes_df.toMap(), "umbral de correlación: 0.8", threshold = 0.8, adjustSize = 2.0)
    .tiles(diag=false)
    .labels(color="white", diag=false)
    .paletteBrBG()
    .build()
```

Out[7]:



Preparamos los datos en dataframes de trabajo

In [8]:

```
// creamos columna de meses
val listMeses : List<Int> = (1..43).toList()
val dfMeses=listMeses.toDataFrame()
val columMeses=dfMeses.getColumn(0)
val columMeses2=dataFrameOf("numeracion_mes")(columMeses)
columMeses2.describe()
```

Out[8]:

	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: rowsCount = 1, columnsCount = 12

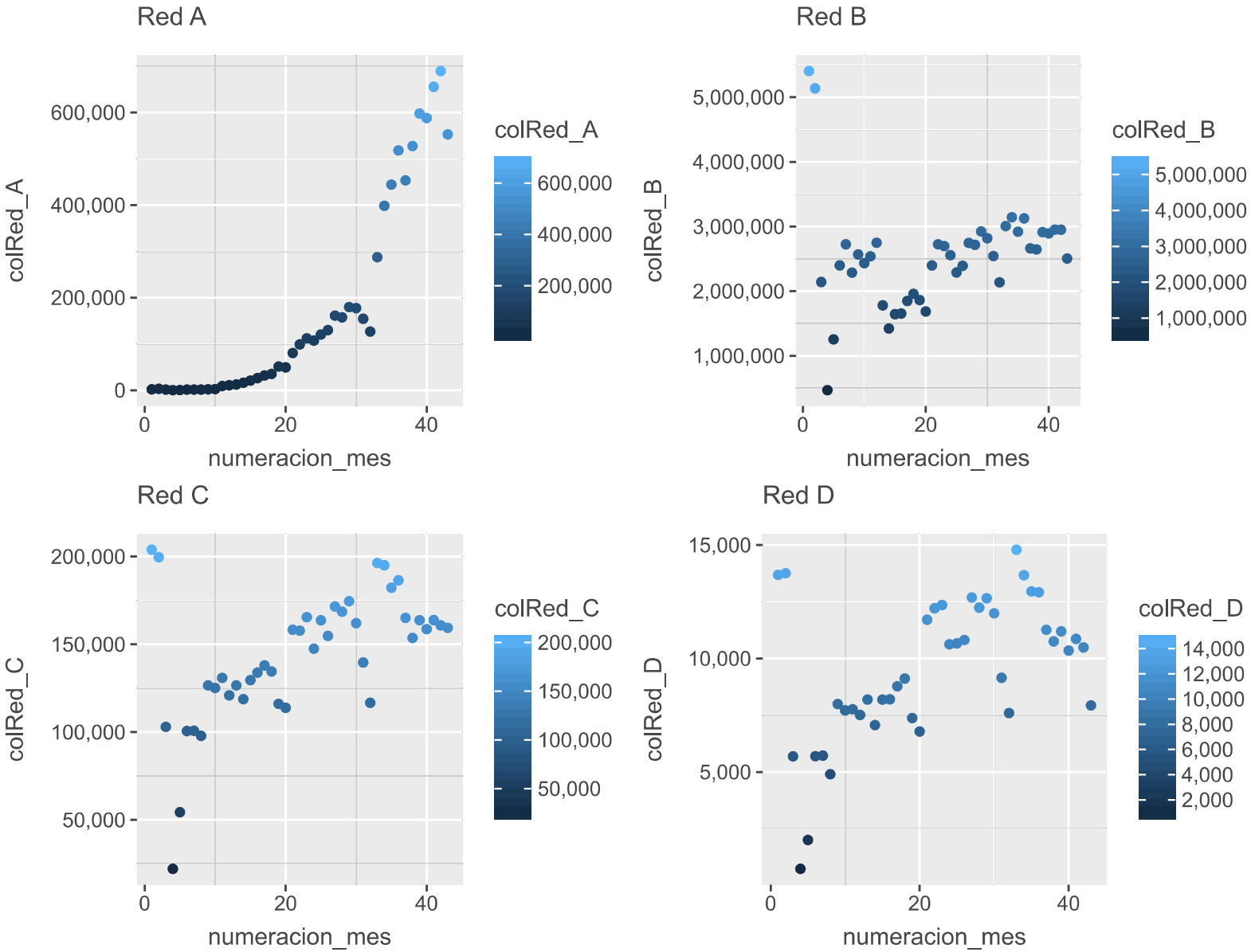
In [9]:
```
// seleccionamos columnas del dataframe origen
val colRed_A by redes_df.red_A
val colRed_B by redes_df.red_B
val colRed_C by redes_df.red_C
val colRed_D by redes_df.red_D
val colRed_E by redes_df.red_E
val colRed_F by redes_df.red_F

// 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()
```

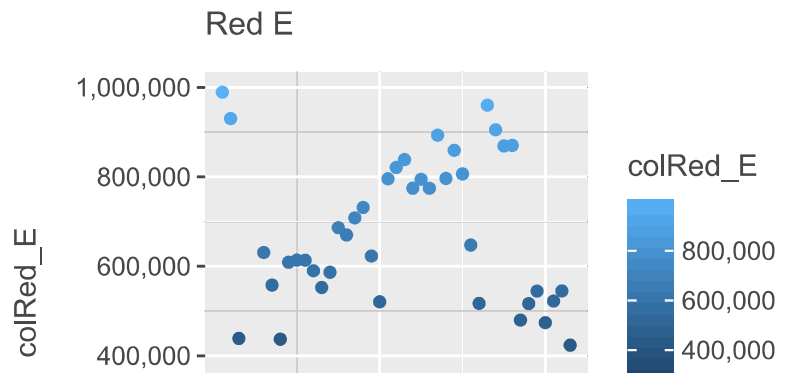
Dibujamos todas las redes respecto al tiempo

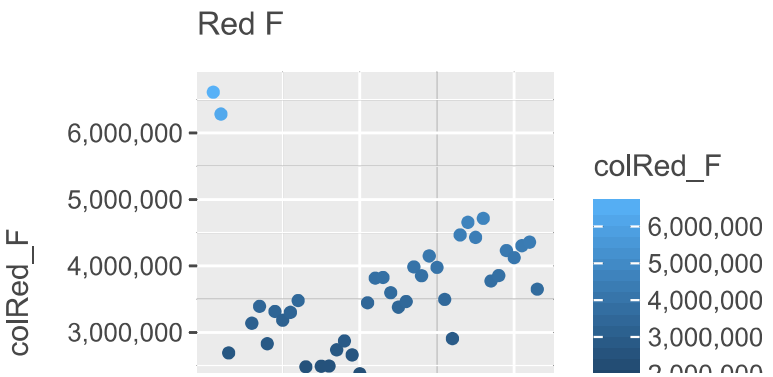
In [10]:
```
// establecemos los modelos de predicción
var ppA = letsPlot(map_RedA){ x = "numeracion_mes"; y = "colRed_A"; color="colRed_A" }
var ppB = letsPlot(map_RedB){ x = "numeracion_mes"; y = "colRed_B" ; color="colRed_B"  }
var ppC = letsPlot(map_RedC){ x = "numeracion_mes"; y = "colRed_C" ; color="colRed_C" }
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 [11]:
```
gggrid(
    plots = listOf(
        ppA + ggtitle("Red A") + geomPoint() + themeGrey(),

        ppB + ggtitle("Red B") + geomPoint() + themeGrey(),

        ppC + ggtitle("Red C") + geomPoint() + themeGrey(),

        ppD + ggtitle("Red D") + geomPoint() + themeGrey(),

        ppE + ggtitle("Red E") + geomPoint() + themeGrey(),

        ppF + ggtitle("Red F") + geomPoint() + themeGrey()
    ),
    ncol = 2,
    cellWidth = 400,
    cellHeight = 300,
    vGap = 0,
    fit = true
)
```

Out[11]:


Red A
colRed_A
600,000
400,000
200,000
0
0
20
40
numeracion_mes
Red B
colRed_B
5,000,000
4,000,000
3,000,000
2,000,000
1,000,000
Red C
colRed_C
200,000
150,000
100,000
50,000
Red D
colRed_D
15,000
10,000
5,000
14,000
12,000
8,000
6,000
4,000
2,000


Red E
1,000,000
800,000
600,000
400,000
colRed_E
colRed_E
800,000
600,000
400,000


Red F
6,000,000
5,000,000
4,000,000
3,000,000
colRed_F
colRed_F
6,000,000
5,000,000
4,000,000
3,000,000

In [ ]: