databrickscovidBinned

Reading covid19.csv

```
path = "/FileStore/tables/"
fc = path + "covid19.csv"

dfc1 = spark.read.csv( fc, header=True, inferSchema=True )
dfc1.printSchema()
dfc1.show(2)
```

```
root
 |-- Province/State: string (nullable = true)
 |-- Country/Region: string (nullable = true)
 |-- Lat: double (nullable = true)
 |-- Long: double (nullable = true)
 |-- Date: string (nullable = true)
 |-- Confirmed: integer (nullable = true)
 |-- Deaths: integer (nullable = true)
 |-- Recovered: integer (nullable = true)

+--------------+--------------+-------+-------+-------+---------+------+---------+
|Province/State|Country/Region|    Lat|   Long|   Date|Confirmed|Deaths|Recovered|
+--------------+--------------+-------+-------+-------+---------+------+---------+
|          null|   Afghanistan|   33.0|   65.0|1/22/20|        0|     0|        0|
|          null|       Albania|41.1533|20.1683|1/22/20|        0|     0|        0|
+--------------+--------------+-------+-------+-------+---------+------+---------+
only showing top 2 rows
```

```
print( dfc1.rdd.getNumPartitions() )
print( sc.defaultParallelism )
dfc1 = dfc1.repartition( 24 )
print( dfc1.rdd.getNumPartitions() )
```

```
1
8
24
```

Schema modification, aggregation on states for each

```
from pyspark.sql import functions as fun
dfc2 = dfc1.select( dfc1["Country/Region"].alias("country"),
                   fun.to_timestamp( "Date", "MM/dd/yy" ).alias("date"),
                   dfc1.Confirmed.alias("confirmed"),
                   dfc1.Deaths.alias("deaths"),
                   dfc1.Recovered.alias("recovered"))\
            .groupby("country", "date")\
            .agg( fun.sum("confirmed").alias("confirmed"),
                  fun.sum("deaths").alias("deaths"),
                  fun.sum("recovered").alias("recovered"))
```

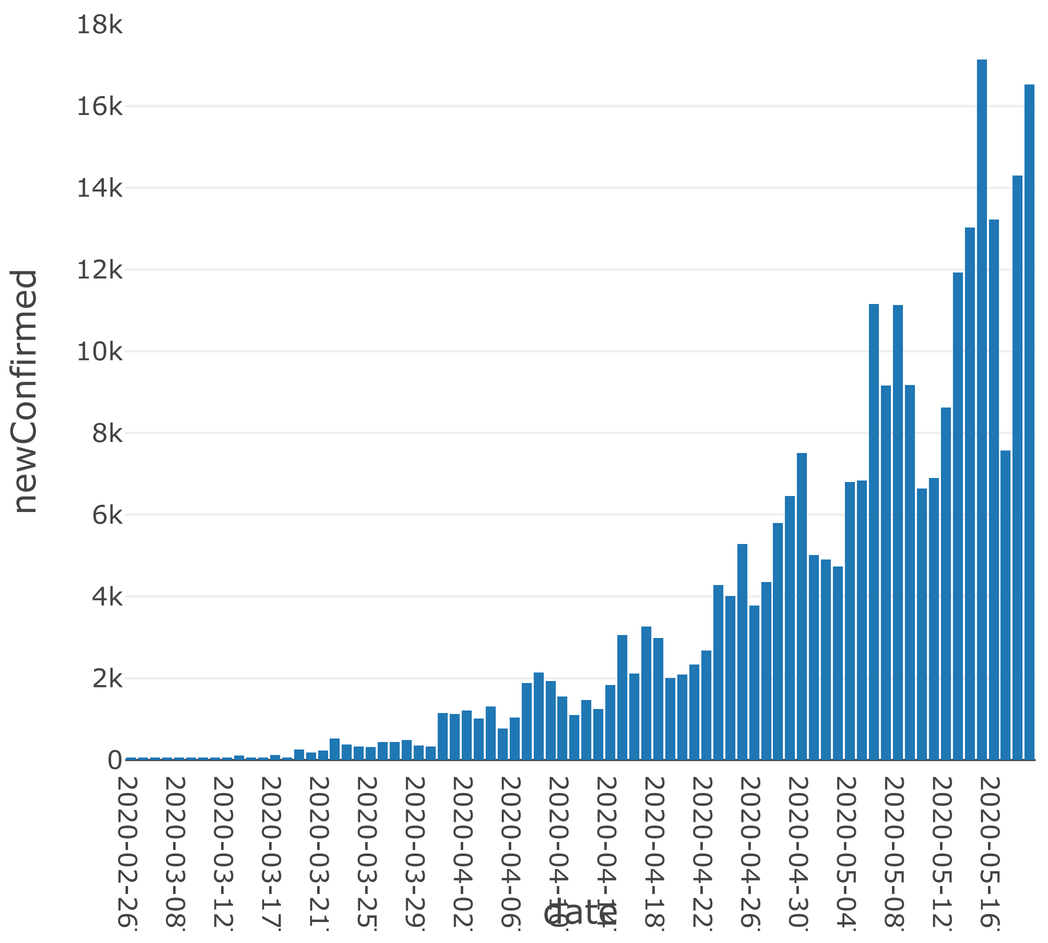
```
dfc2.filter( dfc2.date > fun.lit("2020-04-12") ).show(5)
```

```
+--------------+-------------------+---------+------+---------+
|       country|               date|confirmed|deaths|recovered|
+--------------+-------------------+---------+------+---------+
|United Kingdom|2020-05-17 00:00:00|   244995| 34716|     1058|
|        Zambia|2020-05-10 00:00:00|      267|     7|      117|
|        Kosovo|2020-05-04 00:00:00|      855|    26|      403|
|      Bulgaria|2020-04-12 00:00:00|      675|    29|       68|
|         Egypt|2020-04-29 00:00:00|     5268|   380|     1335|
+--------------+-------------------+---------+------+---------+
only showing top 5 rows
```

Computing the daily statistical changes

```
# Adding newConfirmed ...
from pyspark.sql import Window
win = Window.partitionBy("country").orderBy("date")
dfc3 = dfc2.withColumn( "newConfirmed", dfc2.confirmed - fun.lag( dfc2.confirmed, 1
).over(win) )\
           .withColumn( "newDeaths", dfc2.deaths - fun.lag( dfc2.deaths, 1
).over(win) )\
           .withColumn( "newRecovered", dfc2.recovered - fun.lag( dfc2.recovered, 1
).over(win) )
```

```
display( dfc3.filter( (dfc3.country=='Brazil') & (dfc3.newConfirmed > 0) ) )
# >"2020-03-15"
```



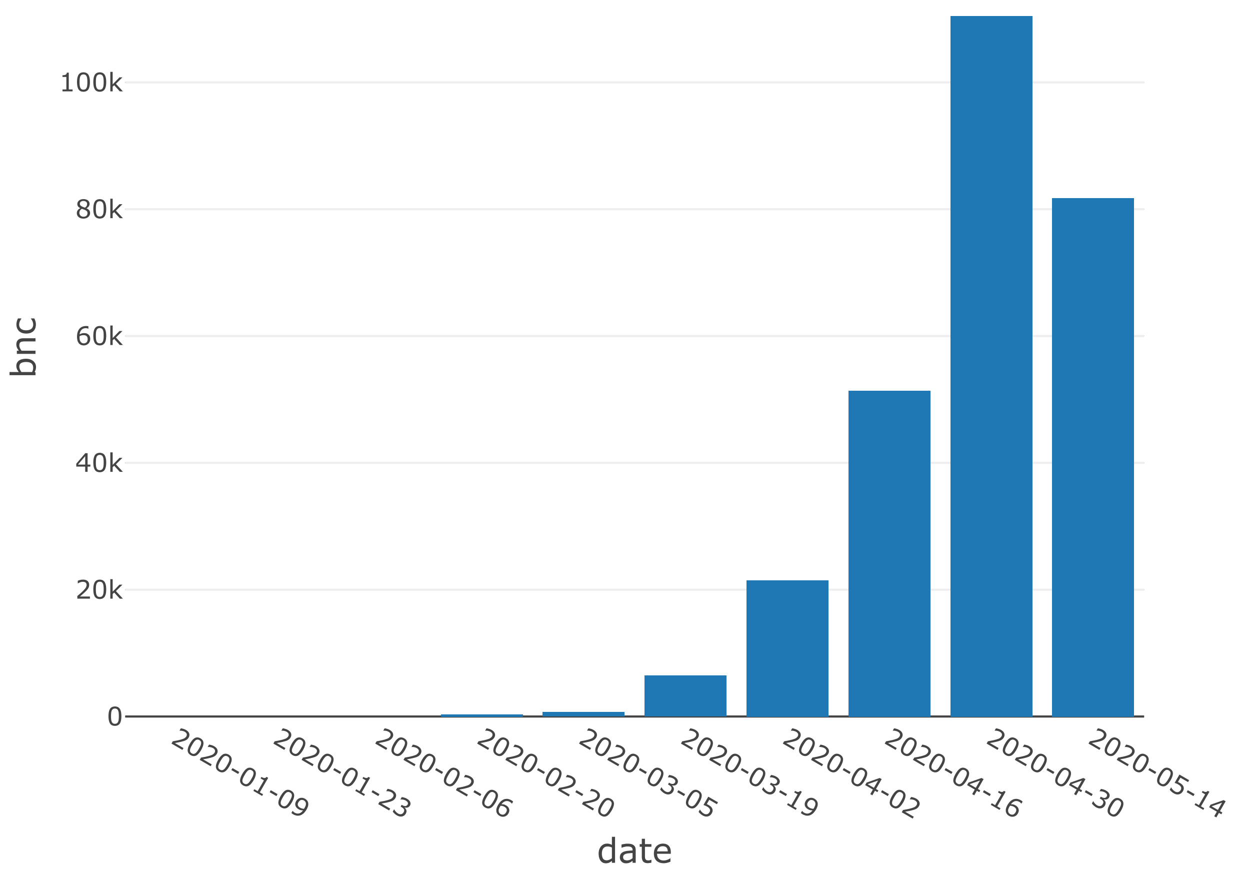
2-week binning over newConfirmed

```
# date binning
dfc4 = dfc3.groupby("country",
                    fun.window("date", "14 days").alias("timeInterval") )\
           .agg( fun.sum("newConfirmed").alias("bnc") )


dfc4.printSchema()
dfc4.filter(dfc4.country=='United Kingdom').show(10, False)
```

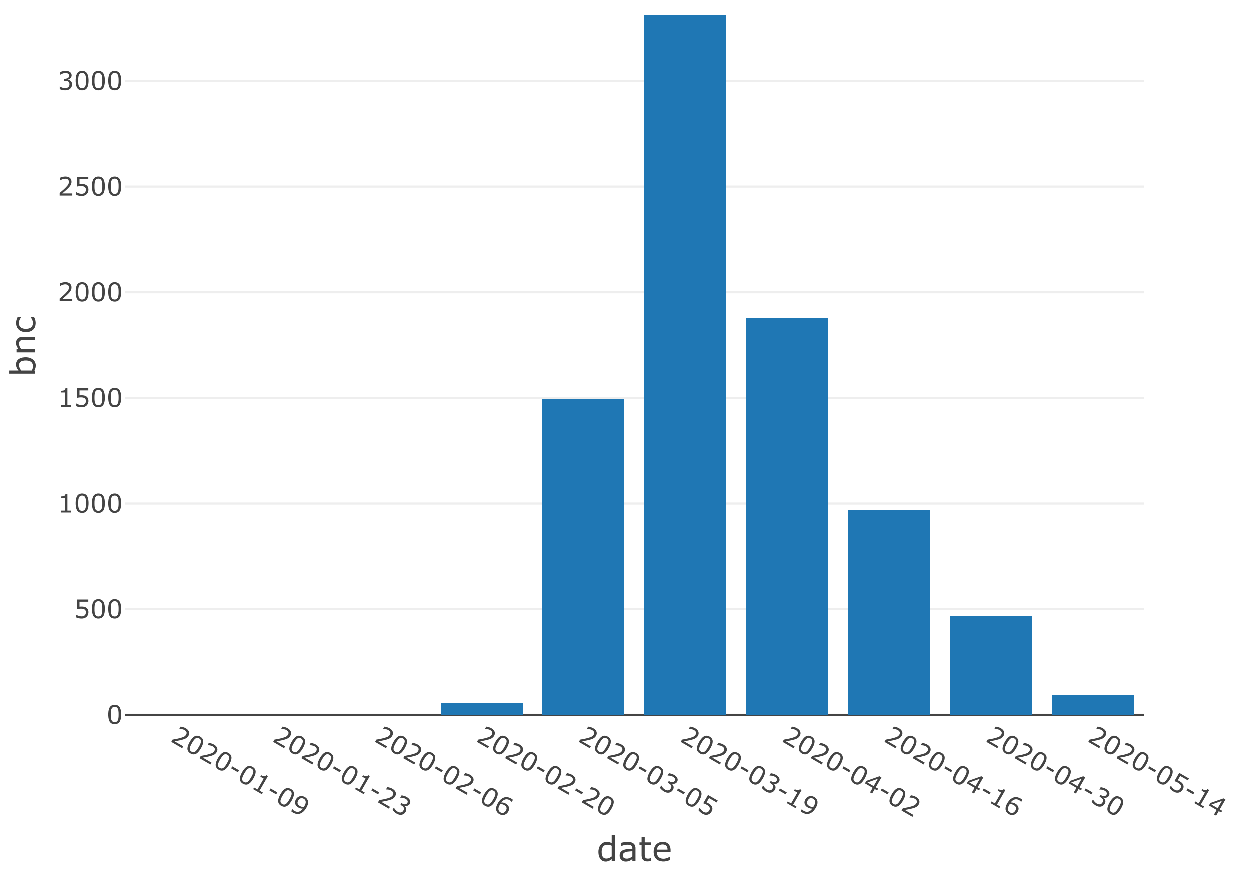
```
root
 |-- country: string (nullable = true)
 |-- timeInterval: struct (nullable = false)
 |    |-- start: timestamp (nullable = true)
 |    |-- end: timestamp (nullable = true)
 |-- bnc: long (nullable = true)

+--------------+------------------------------------------+-----+
|country       |timeInterval                              |bnc  |
+--------------+------------------------------------------+-----+
|United Kingdom|[2020-01-09 00:00:00, 2020-01-23 00:00:00]|null |
|United Kingdom|[2020-01-23 00:00:00, 2020-02-06 00:00:00]|2    |
|United Kingdom|[2020-02-06 00:00:00, 2020-02-20 00:00:00]|7    |
|United Kingdom|[2020-02-20 00:00:00, 2020-03-05 00:00:00]|77   |
|United Kingdom|[2020-03-05 00:00:00, 2020-03-19 00:00:00]|2556 |
|United Kingdom|[2020-03-19 00:00:00, 2020-04-02 00:00:00]|27223|
|United Kingdom|[2020-04-02 00:00:00, 2020-04-16 00:00:00]|69618|
|United Kingdom|[2020-04-16 00:00:00, 2020-04-30 00:00:00]|66958|
|United Kingdom|[2020-04-30 00:00:00, 2020-05-14 00:00:00]|64544|
|United Kingdom|[2020-05-14 00:00:00, 2020-05-28 00:00:00]|19153|
+--------------+------------------------------------------+-----+
```

```
dfc5 = dfc4.select("country", dfc4.timeInterval.start.cast("date").alias("date"), "bnc")
```

```
display( dfc5.filter( (dfc5.country=='Brazil') ) )
```



```
display( dfc5.filter( (dfc5.country=='Norway') ) )
```



```
dfc5.printSchema()
```

```
root
 |-- country: string (nullable = true)
 |-- date: date (nullable = true)
 |-- bnc: long (nullable = true)
```

```
# mean and std per country for binned new confirmmed
dfc5.filter( "date > '2020-03-15' " )\
    .groupby("country")\
    .agg( fun.round( fun.avg("bnc"), 0 ).alias("mean_bnc"),
          fun.round( fun.stddev("bnc"), 0 ).alias("std_bnc") )\
    .orderBy("mean_bnc", ascending=False)\
    .show()
```

```
+--------------+--------+--------+
|       country|mean_bnc| std_bnc|
+--------------+--------+--------+
|            US|304157.0|125969.0|
|        Russia| 59959.0| 54435.0|
|        Brazil| 54303.0| 42662.0|
|United Kingdom| 49499.0| 24254.0|
|         Spain| 43625.0| 37188.0|
|         Italy| 38197.0| 27970.0|
|        France| 34362.0| 29331.0|
|       Germany| 33090.0| 27136.0|
|        Turkey| 30303.0| 19894.0|
|          Iran| 21448.0|  7840.0|
|         India| 21264.0| 16669.0|
|          Peru| 19868.0| 15556.0|
|        Canada| 15967.0|  7694.0|
|  Saudi Arabia| 11937.0|  8989.0|
|       Belgium| 10861.0|  6987.0|
|        Mexico| 10846.0|  8336.0|
|         Chile|  9868.0|  7127.0|
|      Pakistan|  8733.0|  6889.0|
|   Netherlands|  8477.0|  5577.0|
|         Qatar|  7031.0|  5412.0|
+--------------+--------+--------+
only showing top 20 rows
```

```
# median per country for binned new confirmed
from pyspark.sql import Window
win2 = Window.partitionBy( "country" )
dfc5.filter( " date > '2020-03-15' " )\
    .withColumn( "medianBnc", fun.expr( "percentile_approx(bnc, .5)" ).over(win2)
)\
    .show()
```

```
+--------+----------+-----+---------+
| country|      date|  bnc|medianBnc|
+--------+----------+-----+---------+
|    Chad|2020-03-19|    7|       29|
|    Chad|2020-04-02|   16|       29|
|    Chad|2020-04-16|   29|       29|
|    Chad|2020-04-30|  320|       29|
|    Chad|2020-05-14|  173|       29|
|Paraguay|2020-03-19|   58|       89|
|Paraguay|2020-04-02|   92|       89|
|Paraguay|2020-04-16|   78|       89|
|Paraguay|2020-04-30|  501|       89|
|Paraguay|2020-05-14|   89|       89|
|  Russia|2020-03-19|  2630|    57670|
|  Russia|2020-04-02| 21713|    57670|
|  Russia|2020-04-16| 74909|    57670|
|  Russia|2020-04-30|142872|    57670|
|  Russia|2020-05-14| 57670|    57670|
|   Yemen|2020-03-19|     0|        5|
|   Yemen|2020-04-02|     1|        5|
|   Yemen|2020-04-16|     5|        5|
|   Yemen|2020-04-30|    64|        5|
|   Yemen|2020-05-14|    97|        5|
+--------+----------+------+---------+
only showing top 20 rows
```

```
dfc3.printSchema()
```

```
root
 |-- country: string (nullable = true)
 |-- date: timestamp (nullable = true)
 |-- confirmed: long (nullable = true)
 |-- deaths: long (nullable = true)
 |-- recovered: long (nullable = true)
 |-- newConfirmed: long (nullable = true)
 |-- newDeaths: long (nullable = true)
 |-- newRecovered: long (nullable = true)
```

```
# keeping the list of aggragation on 14 days
dfc6 = dfc3.groupby( "country", fun.window( "date", "14 days" ).start.alias("date") )\
    .agg( fun.sum("newConfirmed").alias("bnc"), fun.collect_list( "newConfirmed" ).alias("listBnc") )
```

```
dfc6.filter( "country='France'" ).show(10, False)
dfc3.filter( "country='France'" ).show(10, False)
```

```
+-------+-------------------+-----+-----------------------------------------------------------------------------------------+
|country|date               |bnc  |listBnc                                                                                  |
+-------+-------------------+-----+-----------------------------------------------------------------------------------------+
|France |2020-01-09 00:00:00|null |[]                                                                                       |
|France |2020-01-23 00:00:00|6    |[0, 2, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0]                                               |
|France |2020-02-06 00:00:00|6    |[0, 0, 5, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
|
|France |2020-02-20 00:00:00|276  |[0, 0, 0, 0, 0, 2, 4, 20, 19, 43, 30, 61, 13, 8
4]                                      |
|France |2020-03-05 00:00:00|8836 |[92, 276, 303, 177, 83, 575, 499, 0, 1388, 815,
36, 2151, 1032, 1409]                  |
|France |2020-03-19 00:00:00|48625|[1846, 1788, 1705, 1780, 3880, 2499, 2978, 3951
, 3851, 4703, 2603, 4462, 7657, 4922] |
|France |2020-04-02 00:00:00|75836|[2180, 5273, 4298, 1912, 3931, 3820, 3894, 4309
, 4372, 3125, 26849, 3682, 4971, 3220]|
```

```python
# writting a udf to compute line slop for 14-day list (listBnc)
import numpy as np
from sklearn.linear_model import LinearRegression

@udf( "float" )
def computeSlope( ll ) :
  try :
    zcount = len( [ 0 for i in ll if i==0  ] )
    # return none if there are not enough data points to fit
    if zcount > int(len(ll)*1./2.) : return None

    xx, yy = [], []
    for i in range( len(ll) ) :
      if ll[i] > 0 :
        xx.append( i )
        yy.append( ll[i] )
    X = np.array( xx ).reshape(-1,1) # time
    y = np.array( yy, ) # newConfirmed

    reg = LinearRegression()
    reg.fit(X, y)
    s = reg.score(X,y)
    # to filter the fit quality (0 : no filter)
    if s > .0 :
      # return float( reg.coef_[0] )
      # line slop computation
      return float( round( np.arctan(reg.coef_[0])*180./np.pi, 1 ) )
      # you should call the float function to convert fron numpy-float to Python-
float

    else : return None

  except ValueError :
    return None
```

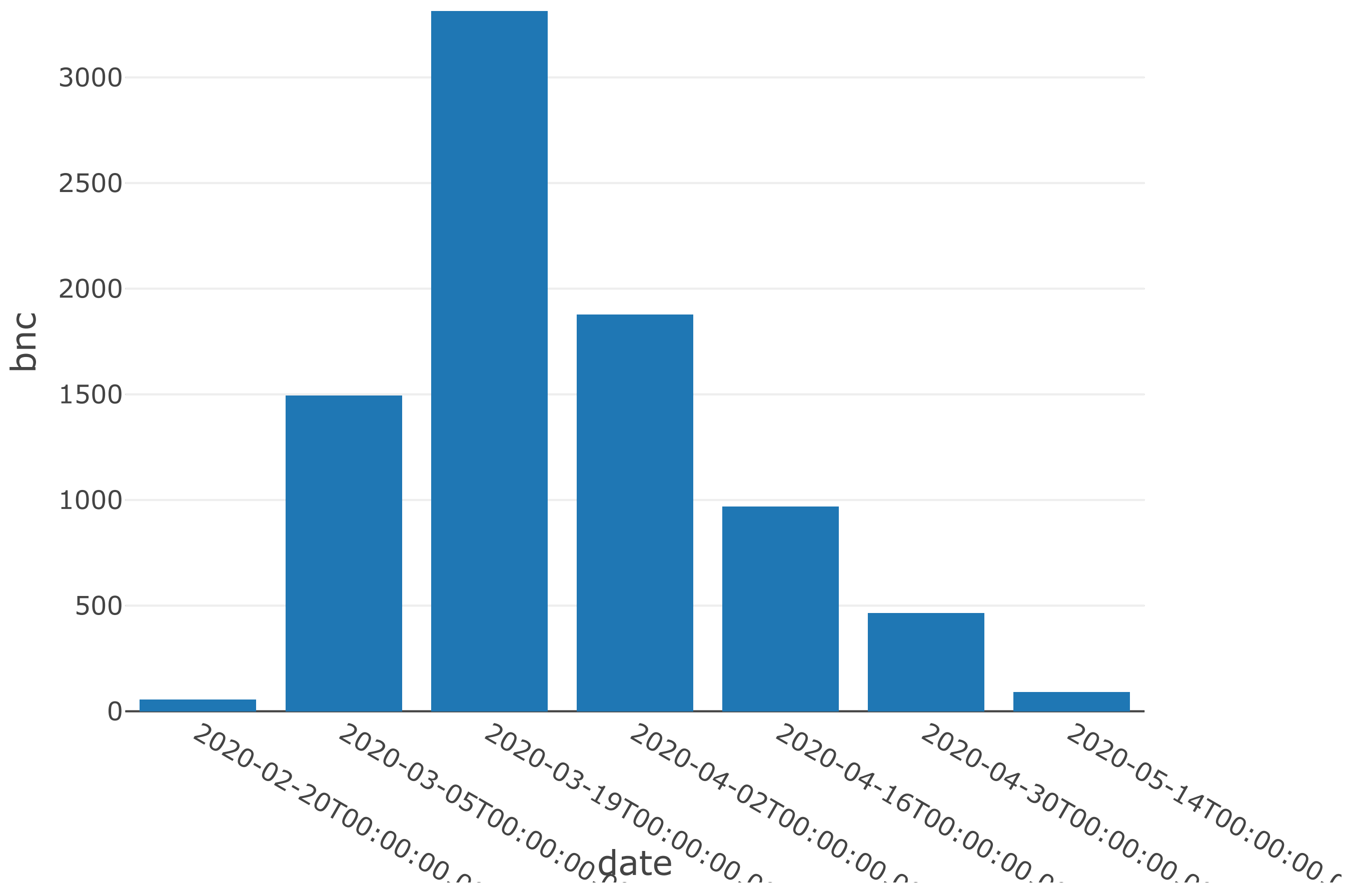
```
dfc7 = dfc6.withColumn( "slope", computeSlope( "listBnc" ) )


# Printng for brazil
dfc7.filter("country='Brazil'").show(100, False)

+-------+-------------------+------+--------------------------------------------------
----------------------------------------+-----+
|country|date               |bnc   |listBnc
|slope|
+-------+-------------------+------+--------------------------------------------------
----------------------------------------+-----+
|Brazil |2020-01-09 00:00:00|null  |[]
|null |
|Brazil |2020-01-23 00:00:00|0     |[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
|null |
|Brazil |2020-02-06 00:00:00|0     |[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
|null |
|Brazil |2020-02-20 00:00:00|4     |[0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 2]
|null |
|Brazil |2020-03-05 00:00:00|368   |[0, 9, 0, 7, 5, 6, 7, 14, 99, 0, 11, 38, 121, 5
1]                                      |81.2 |
|Brazil |2020-03-19 00:00:00|6464  |[249, 172, 228, 525, 378, 323, 307, 431, 432, 4
87, 352, 323, 1138, 1119]               |88.8 |
|Brazil |2020-04-02 00:00:00|21484 |[1208, 1012, 1304, 770, 1031, 1873, 2136, 1922,
1546, 1089, 1465, 1238, 1832, 3058]    |89.3 |
|Brazil |2020-04-16 00:00:00|51365 |[2105, 3257, 2976, 1996, 2089, 2336, 2678, 4279
, 4007, 5281, 3776, 4346, 5789, 6450]   |89.8 |
|Brazil |2020-04-30 00:00:00|110452|[7502, 5015, 4898, 4726, 6794, 6835, 11156, 916
2, 11121, 9167, 6638, 6895, 8620, 11923]|89.8 |
|Brazil |2020-05-14 00:00:00|81748 |[13028, 17126, 13220, 7569, 14288, 16517]
|89.4 |
+-------+-------------------+------+--------------------------------------------------
----------------------------------------+-----+


# Adding the slop sign ( increase/decrease in binned newConfirmed growth)
# groupin by cantry and sign then counting
dfc8 = dfc7\
    .withColumn( "slopeSign", fun.when( dfc7.slope<0., "-" ).otherwise("+") )\
    .groupby( "country", "slopeSign" )\
    .agg( fun.count(dfc7.slope).alias("count") )\
    .orderBy("country")
#     .groupby("country")\
#     .agg( fun.collect_list("count").alias("count-+") )\
dfc8.show(500)
dfc8.filter( "country = 'Norway'" ).show()
```

```
+--------------------+---------+-----+
|             country|slopeSign|count|
+--------------------+---------+-----+
|         Afghanistan|        +|    6|
|             Albania|        +|    1|
|             Albania|        -|    5|
|             Algeria|        -|    2|
|             Algeria|        +|    4|
|             Andorra|        -|    2|
|             Andorra|        +|    2|
|              Angola|        -|    1|
|              Angola|        +|    1|
| Antigua and Barbuda|        +|    0|
|           Argentina|        +|    6|
|             Armenia|        +|    5|
|             Armenia|        -|    1|
|           Australia|        -|    4|
|           Australia|        +|    2|
|             Austria|        -|    4|
|             Austria|        +|    3|
|          Azerbaijan|        +|    3|
```

```
# considering the increase numbers
dfc8.filter("slopeSign='+'").orderBy("count", "country", ascending=False).show(200)
```

```
+--------------------+---------+-----+
|             country|slopeSign|count|
+--------------------+---------+-----+
|           Singapore|        +|    6|
|        Saudi Arabia|        +|    6|
|            Pakistan|        +|    6|
|              Mexico|        +|    6|
|              Kuwait|        +|    6|
|                Iraq|        +|    6|
|               India|        +|    6|
|               Egypt|        +|    6|
|            Colombia|        +|    6|
|               Chile|        +|    6|
|              Brazil|        +|    6|
|           Argentina|        +|    6|
|         Afghanistan|        +|    6|
|United Arab Emirates|        +|    5|
|        South Africa|        +|    5|
|              Russia|         +|     5|
|               Qatar|         +|     5|
```

```
# this is compatible for Brazil plot ( binned newConfimd vs. date ). Explaine why
for the last bin we observe a decreas?
```


bnc
2020-02-20T00:00:00.0
2020-03-05T00:00:00.0
2020-03-19T00:00:00.0
2020-04-02T00:00:00.0
2020-04-16T00:00:00.0
2020-04-30T00:00:00.0
2020-05-14T00:00:00.0
date