First, use the functions previously designed to obtain the desired characteristics.

Later, the RMS in the Time domain will be extracted to a different file. To be used in later approaches too.

Imports and definitions

```
# importing packages
import pandas as pd
import glob
import numpy as np
import scipy as sp
from scipy.stats import kurtosis, skew
# Importing matplotlib to plot the graphs.
import matplotlib.pyplot as plt
from datetime import datetime, date, time, timezone
from natsort import index_natsorted
```

Functions from previous experiments


```
def readFiles(folderpath):
  folder_path = folderpath
  file_list = glob.glob(folder_path + "/*")
  res_dataframe = pd.DataFrame(pd.read_table(file_list[0], header = None))

  for i in range(1,len(file_list)):
    data = pd.read_table(file_list[i], header = None)
    df = pd.DataFrame(data)
    res_dataframe = pd.concat([res_dataframe, df] )#, axis = 1)

  names = []
  for path in file_list:
    names.append(path.split("/")[-1])

  return res_dataframe, names


def rms(array):
  return np.sqrt(np.mean(array**2))


def computeFunctions(df, filename = "Archivo 1", functions = []):
  index = []
  for i in range(0,df.shape[1]):
    index.append("{}: CH{}".format(filename,i))
  if len(functions) ==0 :
    functions = [np.mean, np.std, kurtosis, skew, rms, max, min]
  columns = [f.__name__ for f in functions]

  result = pd.DataFrame(columns = columns,
                        index = index)

  for i in range(0,df.shape[1]):
    for f in functions:
      data = f(df.iloc[:,i])
      result.loc["{}: CH{}".format(filename,i),f.__name__] = data

  return result


def computeFiles(path, amount = -1, functions = []):
  folder_path = path
  file_list = glob.glob(folder_path + "/*")

  filenames = []
  for filepath in file_list:
    filenames.append(filepath.split("/")[-1])

  df = pd.DataFrame(pd.read_table(file_list[0], header = None))
  res = computeFunctions(df, filename = filenames[0] , functions = functions)

  if (amount <0 or amount > len(file_list)):
```

```
    amount = len(file_list)

  for i in range(1, amount):
    temp_df = computeFunctions(
        pd.DataFrame(pd.read_table(file_list[i], header = None)),
        filename = filenames[i],
        functions = []
    )
    res = res.append(temp_df)
    print("Progress: {}/{}".format(i+1,amount))

  return res
```

New Functions

```
def computeRMS(df, filename):
  # For the df given, index = filename, column = channel
  index = [filename]
  columns = range(0,df.shape[1])

  result = pd.DataFrame(columns = columns,
                        index = index)

  for i in range(0,df.shape[1]):
    data = rms(df.iloc[:,i])
    result.loc[filename,i] = data



  return result


def computeFilesRMS(path, amount = -1):
  # First get the files
  folder_path = path
  file_list = glob.glob(folder_path + "/*")

  filenames = []
  for filepath in file_list:
    filenames.append(filepath.split("/")[-1])

  print("Progress: Done loading files")

  # Do the task for the first file
  df = pd.DataFrame(pd.read_table(file_list[0], header = None))
  res = computeRMS(df, filename = filenames[0])


  # And then for the rest
  if (amount <0 or amount > len(file_list)):
    amount = len(file_list)
  print("Progress: {}/{}".format(1,amount))

  for i in range(1, amount):
    temp_df = computeRMS(
        pd.DataFrame(pd.read_table(file_list[i], header = None)),
        filename = filenames[i],
    )
    res = res.append(temp_df)
    print("Progress: {}/{}".format(i+1,amount))


  return res


test = pd.DataFrame(columns = range(0, 8),
            index = ["filename"])
test
```

	0	1	2	3	4	5	6	7
filename	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
def parseFilenameToDate(Filename):
  new_str = Filename.replace(".","/")
  dt = datetime.strptime(new_str, "%Y/%m/%d/%H/%M/%S")
  return dt


dt = parseFilenameToDate("2003.11.16.06.58.46")
dt
```

```
     datetime.datetime(2003, 11, 16, 6, 58, 46)
```

Better functions

Developed on future approaches, but useful here

```
def computeFilesVelocityRMS(path, amount = -1):
  # First get the files
  folder_path = path
  file_list = glob.glob(folder_path + "/*")

  filenames = []
  for filepath in file_list:
    filenames.append(filepath.split("/")[-1])

  print("Progress: Done loading files")

  # Do the task for the first file
  df = pd.DataFrame(pd.read_table(file_list[0], header = None))
  res = computeVelocityRMS(df, filename = filenames[0])


  # And then for the rest
  if (amount <0 or amount > len(file_list)):
    amount = len(file_list)
  print("Progress: {}/{}".format(1,amount))

  for i in range(1, amount):
    temp_df = computeVelocityRMS(
        pd.DataFrame(pd.read_table(file_list[i], header = None)),
        filename = filenames[i],
    )
    res = res.append(temp_df)
    print("Progress: {}/{}".format(i+1,amount))


  return res


def computeVelocityRMS(df, filename):
  # For the df given, index = filename, column = channel
  index = [filename]
  columns = range(0,df.shape[1])

  result = pd.DataFrame(columns = columns,
                        index = index)

  for i in range(0,df.shape[1]):
    data = speed_rms_from_acceleration(df.iloc[:,i], 20480)
    result.loc[filename,i] = data



  return result


def speed_rms_from_acceleration(data, fs):
  """
  Data is an array like, fs the sampling frequency

  Data is assumed that is measured in g
  """
  # First center the data using the mean to cancel out gravity
  mean_values = data.mean()
  data = data - mean_values

  # Perform the operations
```

```
  G = 9.80665
  accel = G * data * 1000

  velocity = sp.integrate.cumulative_trapezoid(accel, dx = 1/fs, initial = 0)

  return rms(velocity)
```

Old tests

For the previously designed functions. Can be left out of execution

```
result = computeFiles('/content/drive/MyDrive/Colab Notebooks/TFG/Trasteo/test1', amount = 3)
```

```
Progress: 2/3
Progress: 3/3
```

```
result
```

	mean	std	kurtosis	skew	rm
2003.11.16.06.58.46: CH0	-0.118469	0.097826	0.879857	-0.039519	0.15363
2003.11.16.06.58.46: CH1	-0.119506	0.089263	1.140088	0.090826	0.14916
2003.11.16.06.58.46: CH2	-0.11776	0.092779	0.315984	-0.061028	0.14991
2003.11.16.06.58.46: CH3	-0.118254	0.07583	0.362793	-0.01728	0.14047
2003.11.16.06.58.46: CH4	-0.115443	0.113252	0.568761	0.024861	0.16171
2003.11.16.06.58.46: CH5	-0.115118	0.119898	0.555058	-0.016134	0.16621
2003.11.16.06.58.46: CH6	-0.114694	0.076305	0.698221	-0.004425	0.13775
2003.11.16.06.58.46: CH7	-0.113825	0.079905	0.62012	0.001561	0.13907
2003.11.16.07.38.46: CH0	-0.116214	0.096984	1.008371	0.008472	0.15136
2003.11.16.07.38.46: CH1	-0.116402	0.088075	0.754524	0.046039	0.14596
2003.11.16.07.38.46: CH2	-0.115832	0.091484	0.176354	-0.041141	0.14760
2003.11.16.07.38.46: CH3	-0.116716	0.074438	0.411683	-0.027022	0.13843

```
result = computeFiles('/content/drive/MyDrive/Colab Notebooks/TFG/Trasteo/test1', amount = 3, functions = [rms])
```

```
Progress: 2/3
Progress: 3/3
```

```
result
```

	rms
2003.11.16.06.58.46: CH0	0.153639
2003.11.16.06.58.46: CH1	0.149163
2003.11.16.06.58.46: CH2	0.149918
2003.11.16.06.58.46: CH3	0.140479
2003.11.16.06.58.46: CH4	0.161719
2003.11.16.06.58.46: CH5	0.166215
2003.11.16.06.58.46: CH6	0.137758
2003.11.16.06.58.46: CH7	0.139072
2003.11.16.07.38.46: CH0	0.151366
2003.11.16.07.38.46: CH1	0.145968
2003.11.16.07.38.46: CH2	0.147602
2003.11.16.07.38.46: CH3	0.138433
2003.11.16.07.38.46: CH4	0.16126
2003.11.16.07.38.46: CH5	0.162983
2003.11.16.07.38.46: CH6	0.135839
2003.11.16.07.38.46: CH7	0.138224
2003.11.16.06.48.46: CH0	0.152085
2003.11.16.06.48.46: CH1	0.1455
2003.11.16.06.48.46: CH2	0.149149

```
Index(['2003.11.16.06.58.46: CH0', '2003.11.16.06.58.46: CH1',
       '2003.11.16.06.58.46: CH2', '2003.11.16.06.58.46: CH3',
       '2003.11.16.06.58.46: CH4', '2003.11.16.06.58.46: CH5',
       '2003.11.16.06.58.46: CH6', '2003.11.16.06.58.46: CH7',
       '2003.11.16.07.38.46: CH0', '2003.11.16.07.38.46: CH1',
       '2003.11.16.07.38.46: CH2', '2003.11.16.07.38.46: CH3',
       '2003.11.16.07.38.46: CH4', '2003.11.16.07.38.46: CH5',
       '2003.11.16.07.38.46: CH6', '2003.11.16.07.38.46: CH7',
       '2003.11.16.06.48.46: CH0', '2003.11.16.06.48.46: CH1',
       '2003.11.16.06.48.46: CH2', '2003.11.16.06.48.46: CH3',
       '2003.11.16.06.48.46: CH4', '2003.11.16.06.48.46: CH5',
       '2003.11.16.06.48.46: CH6', '2003.11.16.06.48.46: CH7'],
      dtype='object')
```

Characteristics Overview

Compare the first and last file's characteristics, for all channels

For this, those files were selected and saved in a different folder

```
folder_name = "Test1-FirstLast"
```

Computations

```
path_to_folder = f"/content/drive/MyDrive/Colab Notebooks/TFG/Trasteo/{folder_name}"
```

```
result = computeFiles(path_to_folder, amount=2)
```

```
Progress: 2/2
<ipython-input-13-e05b1a3dde58>:21: FutureWarning: The frame.append method is deprecated and will be removed from pandas ir
  res = res.append(temp_df)
```

Result

```
result
```

	mean	std	kurtosis	skew	rm
2003.10.22.12.06.24: CH0	-0.094593	0.081122	1.069163	-0.029993	0.1246
2003.10.22.12.06.24: CH1	-0.09388	0.070648	3.065884	0.220116	0.1174
2003.10.22.12.06.24: CH2	-0.093817	0.090648	0.209486	-0.092073	0.1304
2003.10.22.12.06.24: CH3	-0.093752	0.077508	0.292221	-0.053183	0.1216
2003.10.22.12.06.24: CH4	-0.090812	0.091461	0.405439	0.034372	0.1288
2003.10.22.12.06.24: CH5	-0.090881	0.095486	0.777063	-0.022223	0.1318
2003.10.22.12.06.24: CH6	-0.090969	0.060085	0.79025	0.046609	0.109
2003.10.22.12.06.24: CH7	-0.094235	0.06638	1.80799	0.008295	0.1152

New Tests and compute RMS dataset

For new functions. Compute new dataset containing RMS for each Channel.

WARNING: Takes its time!

```
test_to_use = "2nd_test"
```

Acceleration RMS

```
computeFilesRMS(f'/content/drive/MyDrive/Colab Notebooks/TFG/Trasteo/{test_to_use}', amount= 10)
```

```
Progress: Done loading files
Progress: 1/10
Progress: 2/10
Progress: 3/10
```

```
result = computeFilesRMS(f'/content/drive/MyDrive/Colab Notebooks/TFG/Trasteo/{test_to_use}')
```

```
Progress: Done loading files
Progress: 1/2156
Progress: 2/2156
Progress: 3/2156
Progress: 4/2156
Progress: 5/2156
Progress: 6/2156
Progress: 7/2156
Progress: 8/2156
Progress: 9/2156
Progress: 10/2156
Progress: 11/2156
Progress: 12/2156
Progress: 13/2156
Progress: 14/2156
Progress: 15/2156
Progress: 16/2156
Progress: 17/2156
Progress: 18/2156
Progress: 19/2156
Progress: 20/2156
Progress: 21/2156
Progress: 22/2156
Progress: 23/2156
Progress: 24/2156
Progress: 25/2156
Progress: 26/2156
Progress: 27/2156
Progress: 28/2156
Progress: 29/2156
Progress: 30/2156
Progress: 31/2156
Progress: 32/2156
Progress: 33/2156
Progress: 34/2156
Progress: 35/2156
Progress: 36/2156
Progress: 37/2156
Progress: 38/2156
Progress: 39/2156
Progress: 40/2156
Progress: 41/2156
Progress: 42/2156
Progress: 43/2156
Progress: 44/2156
Progress: 45/2156
Progress: 46/2156
Progress: 47/2156
Progress: 48/2156
Progress: 49/2156
Progress: 50/2156
Progress: 51/2156
Progress: 52/2156
Progress: 53/2156
Progress: 54/2156
Progress: 55/2156
Progress: 56/2156
Progress: 57/2156
```

```
result
```

	0	1	2	3	4
2003.11.16.06.58.46	0.153639	0.149163	0.149918	0.140479	0.161719
2003.11.16.07.38.46	0.151366	0.145968	0.147602	0.138433	0.16126
2003.11.16.06.48.46	0.152085	0.1455	0.149149	0.13931	0.160837
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

```
result.to_csv(f"/content/drive/MyDrive/Colab Notebooks/TFG/Trasteo/timeRMS-{test_to_use}.csv")
```

Velocity RMS

```
computeFilesVelocityRMS(f'/content/drive/MyDrive/{test_to_use}', amount= 10)
```

```
Progress: Done loading files
Progress: 1/10
Progress: 2/10
Progress: 3/10
Progress: 4/10
Progress: 5/10
Progress: 6/10
Progress: 7/10
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.appen
  res = res.append(temp_df)
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.appen
  res = res.append(temp_df)
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.appen
  res = res.append(temp_df)
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.appen
  res = res.append(temp_df)
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.appen
  res = res.append(temp_df)
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.appen
  res = res.append(temp_df)
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.appen
  res = res.append(temp_df)
Progress: 8/10
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.appen
  res = res.append(temp_df)
Progress: 9/10
Progress: 10/10
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.appen
  res = res.append(temp_df)
```

	0	1	2	3
2004.02.12.10.42.39	1.579103	1.413506	2.276786	1.082238
2004.02.12.11.22.39	0.827692	0.782831	1.256357	1.761797
2004.02.12.12.22.39	0.940373	1.202879	2.317534	2.194585
2004.02.12.13.22.39	1.173243	1.429123	1.252211	2.667121
2004.02.12.12.02.39	0.864917	0.976204	2.604701	2.360595

```
result = computeFilesVelocityRMS(f'/content/drive/MyDrive/{test_to_use}')
```

```
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.append method is deprecated and will be removed from pandas
  res = res.append(temp_df)
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.append method is deprecated and will be removed from pandas
  res = res.append(temp_df)
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.append method is deprecated and will be removed from pandas
  res = res.append(temp_df)
Progress: Done loading files
Progress: 1/984
Progress: 2/984
Progress: 3/984
Progress: 4/984
Progress: 5/984
```

```
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.append method is deprecated and will be removed from pandas
  res = res.append(temp_df)
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.append method is deprecated and will be removed from pandas
  res = res.append(temp_df)
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.append method is deprecated and will be removed from pandas
  res = res.append(temp_df)
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.append method is deprecated and will be removed from pandas
  res = res.append(temp_df)
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.append method is deprecated and will be removed from pandas
  res = res.append(temp_df)
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.append method is deprecated and will be removed from pandas
  res = res.append(temp_df)
Progress: 6/984
Progress: 7/984
Progress: 8/984
Progress: 9/984
Progress: 10/984
Progress: 11/984
Progress: 12/984
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.append method is deprecated and will be removed from pandas
  res = res.append(temp_df)
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.append method is deprecated and will be removed from pandas
  res = res.append(temp_df)
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.append method is deprecated and will be removed from pandas
  res = res.append(temp_df)
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.append method is deprecated and will be removed from pandas
  res = res.append(temp_df)
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.append method is deprecated and will be removed from pandas
  res = res.append(temp_df)
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.append method is deprecated and will be removed from pandas
  res = res.append(temp_df)
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.append method is deprecated and will be removed from pandas
  res = res.append(temp_df)
Progress: 13/984
Progress: 14/984
Progress: 15/984
Progress: 16/984
Progress: 17/984
Progress: 18/984
Progress: 19/984
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.append method is deprecated and will be removed from pandas
  res = res.append(temp_df)
<ipython-input-11-3543710fc30e>:27: FutureWarning: The frame.append method is deprecated and will be removed from pandas
  res = res.append(temp_df)
```

```
result
```

	0	1	2	3
2004.02.12.10.42.39	1.579103	1.413506	2.276786	1.082238
2004.02.12.11.22.39	0.827692	0.782831	1.256357	1.761797
2004.02.12.12.22.39	0.940373	1.202879	2.317534	2.194585
2004.02.12.13.22.39	1.173243	1.429123	1.252211	2.667121
2004.02.12.12.02.39	0.864917	0.976204	2.604701	2.360595
...	...	...	...	...
2004.02.19.04.12.39	5.762988	2.289214	1.45226	1.098697
2004.02.19.05.32.39	5.626528	0.981583	0.964049	1.387236
2004.02.19.05.22.39	5.646649	0.975078	1.555583	1.783656
2004.02.19.05.02.39	12.529171	2.491304	1.411974	1.565856

```
result.to_csv(f"/content/drive/MyDrive/timeRMS-{test_to_use}-velocity.csv")
```

Computing information from RMS dataset

Using the RMS dataset previously computed.

```
filename = "timeRMS-test1-velocity.csv"
channel = 8
bearing = 4
test = 1
```

Load the new dataset

timeRMS-test1 has all the information previously computed, so it can be started from there.

```
all_rms = pd.read_csv(f"/content/drive/MyDrive/{filename}")
all_rms.rename(columns = {all_rms.columns[0] : "Filename"}, inplace = True)
all_rms
```

	Filename	0	1	2	3	
0	2003.11.16.06.38.46	2.627859	4.197441	2.784025	3.019036	2.4
1	2003.11.16.07.28.46	2.262329	6.796080	4.224002	4.363609	5.1
2	2003.11.16.06.28.46	6.518991	8.261459	4.852692	5.535518	4.1
3	2003.11.16.08.18.46	1.707155	1.670507	3.140193	1.903130	2.7
4	2003.11.16.07.38.46	3.929812	4.231810	2.207813	1.833236	4.9
...	...	...	...	...	...	
2151	2003.10.29.14.39.46	5.360851	3.457391	5.620577	4.006830	5.8
2152	2003.10.23.10.04.13	4.529668	3.560215	2.771665	3.915751	3.7
2153	2003.10.23.08.44.13	1.326365	5.320566	2.426306	4.175405	4.0

First graphs

It turns out filenames are not ordered. The graphs of this section are unordered, so they can be left out of execution

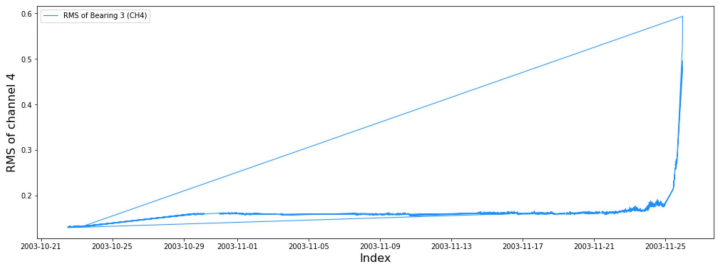
```
X = all_rms.loc[:,"Filename"]
Y = all_rms.loc[:,"0"]


plt.figure(figsize=(17,6))

plt.plot(X, Y, 'dodgerblue', label = 'RMS of Bearing 1 (CH0)', linewidth = 1)

# X-axis label.
plt.xlabel('Index', fontsize = 16)

# Y-axis label.
plt.ylabel('RMS of channel 0', fontsize = 16)

# Grid
# plt.grid(True)
# plt.grid(False)

# Legend for the plot.
plt.legend()

# Displays the plot.
plt.show()
```

```
---------------------------------------------------------------
----------
ValueError                                Traceback (most recent
call last)
<ipython-input-33-16b542c4d5cb> in <module>
      1 plt.figure(figsize=(17,6))
      2
----> 3 plt.scatter(X, Y, 'dodgerblue', label = 'RMS of Bearing 1
(CH0)', linewidth = 1)
      4
      5 # X-axis label.

 2 frames
/usr/local/lib/python3.8/dist-packages/matplotlib/axes/_axes.py
in scatter(self, x, y, s, c, marker, cmap, norm, vmin, vmax,
alpha, linewidths, edgecolors, plotnonfinite, **kwargs)
   4375                 (not np.issubdtype(s.dtype, np.floating)
and
   4376                  not np.issubdtype(s.dtype,
np.integer))):
-> 4377             raise ValueError(
   4378                 "s must be a scalar, "
   4379                 "or float array-like with the same size
as x and y")

ValueError: s must be a scalar, or float array-like with the same
size as x and y
```

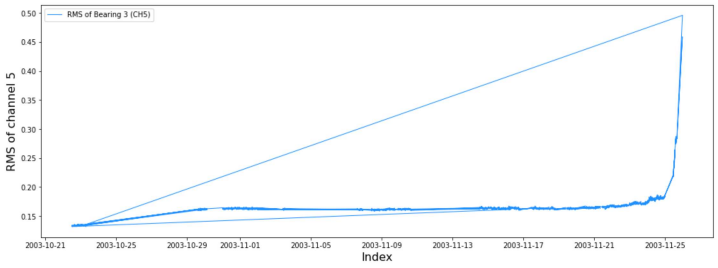
BUSCAR EN STACK OVERFLOW

I should fix the dates...

```
Y= all_rms.loc[:,"4"]


plt.figure(figsize=(17,6))

plt.plot(X, Y, 'dodgerblue', label = 'RMS of Bearing 3 (CH4)', linewidth = 1)

# X-axis label.
plt.xlabel('Index', fontsize = 16)

# Y-axis label.
plt.ylabel('RMS of channel 4', fontsize = 16)

# Grid
# plt.grid(True)
# plt.grid(False)

# Legend for the plot.
plt.legend()

# Displays the plot.
plt.show()
```



```
Y= all_rms.loc[:,"5"]


plt.figure(figsize=(17,6))

plt.plot(X, Y, 'dodgerblue', label = 'RMS of Bearing 3 (CH5)', linewidth = 1)

# X-axis label.
plt.xlabel('Index', fontsize = 16)

# Y-axis label.
plt.ylabel('RMS of channel 5', fontsize = 16)

# Grid
# plt.grid(True)
# plt.grid(False)

# Legend for the plot.
plt.legend()

# Displays the plot.
plt.show()
```



Graphs with fixed dates

Computation

```
reordered_data = all_rms.sort_values(
   by="Filename",
   key=lambda x: np.argsort(index_natsorted(all_rms["Filename"]))
)


reordered_data.loc[:,"Filename"] = reordered_data.loc[:,"Filename"].apply(parseFilenameToDate)
reordered_data
```

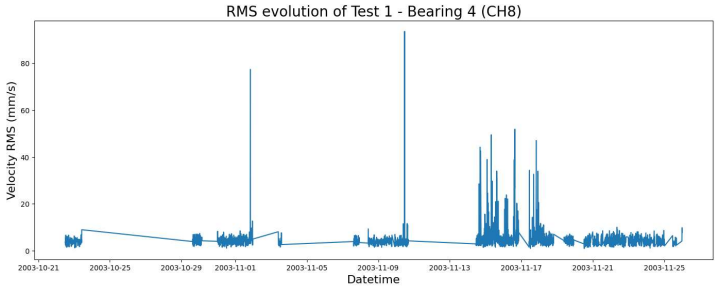
```
<ipython-input-44-0e3220b6247c>:1: DeprecationWarning: In a future
  reordered_data.loc[:,"Filename"] = reordered_data.loc[:,"Filenar
```

	Filename	0	1	2	3	4
2001	2003-10-22 12:06:24	3.068659	5.487772	2.060324	2.112178	3.823492

```
X = reordered_data.loc[:,"Filename"]
Y = reordered_data.loc[:,f"{channel - 1}"]
```

Results

```
plt.figure(figsize=(17,6))

#plt.scatter(X, Y, c='dodgerblue', label = f'RMS of Bearing {bearing} (CH{channel})', linewidth = 0.1)

plt.plot(X, Y)

#plt.plot (X, Y, 'green', linewidth = 0.2)

# X-axis label.
plt.xlabel('Datetime', fontsize = 16)

# Y-axis label.
plt.ylabel(f'Velocity RMS (mm/s)', fontsize = 16)

plt.title(f'RMS evolution of Test {test} - Bearing {bearing} (CH{channel})', fontsize = 20)

# Displays the plot.
plt.show()
```




```
reordered_data
```

	Filename	0	1	2	3
6007	2004-03-04 09:27:46	2.328026	2.133527	1.850454	2.786296
6004	2004-03-04 09:32:46	2.715131	3.353833	1.481665	1.609627
6008	2004-03-04 09:42:46	1.195718	2.054011	1.125469	3.028827
6001	2004-03-04 09:52:46	4.974777	6.160984	2.498490	2.135966
6000	2004-03-04 10:02:46	2.269075	1.561583	1.715397	1.663348
...	...	...	...	...	...
993	2004-04-18 02:02:55	1.174672	1.375939	24.815157	1.611162
983	2004-04-18 02:12:55	1.499966	1.476586	9.229391	2.333371
995	2004-04-18 02:22:55	1.921164	1.650795	98.199024	1.478748
998	2004-04-18 02:32:55	3.169191	1.143316	89.106880	2.414869