In [1]:

```
# PYTHON LIBRARIES
%matplotlib inline

import math
import numpy as np
np.seterr(divide='ignore', invalid='ignore')
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
plt.rc('xtick', labelsize=15)
plt.rc('ytick', labelsize=15)
import statsmodels.api as sm
import statsmodels.formula.api as smf

from matplotlib import cm
from matplotlib.axes._axes import _log as matplotlib_axes_logger
matplotlib_axes_logger.setLevel('ERROR')
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
number = LabelEncoder()
from statsmodels.stats.outliers_influence import summary_table
from adjustText import adjust_text
from collections import OrderedDict

# Adjust css for usability
from IPython.core.display import HTML
HTML('''
<style type="text/css">

.jp-RenderedHTMLCommon table {
    table-layout: auto;
    border-collapse: collapse;
    width: 75%;
}

.jp-RenderedHTMLCommon table .absorbing-column {
    width: 75%;
}

</style>
''')
```

Out[1]:

Function to scatter plot

In [2]:

```
# GET THE MAGNITUDE ORDER OF A NUMBER
def magnitude(value):
    if (value == 0): return 0
    return 10**(int(math.floor(math.log10(abs(value)))))
```

In [3]:

```
# SIMPLE SCATTER PLOT OF TWO VARIABLES
def scatterPlot(x_str, x_units, y_str, y_units, df, fig_name):
    # PLOT FIG
    scale = 6;
    fig, ax = plt.subplots(figsize=(3*scale, 2*scale));

    # sort values by the independent variable
    df_x = df.sort_values(by=[x_str])
    # remove NANs from both variables and store them
    df_x = df_x.dropna(subset=[x_str, y_str])
    x = df_x.iloc[:][x_str]
```

```
    y = df_x.iloc[:][y_str]

    # Plot
    plt.scatter(x, y, s=25)

    # Display plots
    plt.yscale('linear');
    plt.xlabel(x_str + '     ' + x_units, fontsize=24);
    plt.ylabel(y_str + '     ' + y_units, fontsize=24);
    plt.title(fig_name, size=24);
    #plt.legend(prop={'size': 18});
    #plt.ticklabel_format(axis='both', style='sci', scilimits=(-2,2))
    plt.show();
```

In [4]:

```
# SCATTER PLOT WITH AXIS-BREAK AND REFERENCE ANNOTATIONS
def scatterPlot_breakAxis(x_str, x_units, y_str, y_units, df, df_x, breakYlim, legloc):

    # GET THE X Y VALUES
    # sort values by the independent variable
    df_xx = df_x.sort_values(by=[y_str])
    # remove NANs from both variables and store them
    df_xx = df_x.dropna(subset=[x_str, y_str])
    x = df_xx.iloc[:][x_str]
    y = df_xx.iloc[:][y_str]
    polymerColour = df_xx.iloc[:]['Polymer']

    # GET THE REFERENCE STRING VALUES FOR PLOT ANNOTATIONS
    ref = df.iloc[:]['Reference']
    polymerName = df.iloc[:]['Polymer']

    # CREATE A NEW DATAFRAME WITH THE INTERESTING DATA ONLY
    # IN ORDER TO EFFECTIVELY REMOVE DUPLICATES
    new_df = pd.DataFrame(x)
    new_df = new_df.join(pd.DataFrame(y))
    new_df = new_df.join(pd.DataFrame(ref))
    new_df = new_df.join(pd.DataFrame(polymerColour))
    new_df = new_df.join(pd.DataFrame(polymerName).rename(columns={"Polymer": "Polymer Name"}))

    # Drop duplicate values
    new_df = new_df.drop_duplicates(subset=new_df.columns.difference(['Polymer', 'Polymer Name']))
    # sort values by the independent variable
    new_df = new_df.sort_values(by=[y_str])

    # Print the interesting data
    print('>>> new_df')
    display(new_df)

    # Extract the interesting data frame into individual
    # panda series
    x   = new_df.iloc[:][x_str]
    y   = new_df.iloc[:][y_str]
    ref = new_df.iloc[:]['Reference']
    polColour = new_df.iloc[:]['Polymer']
    polName = new_df.iloc[:]['Polymer Name']

    # PLOT SETUP
    scale = 6;
    fig = plt.figure(figsize=(3*scale, 2*scale))

    # Inplement a 3rows-1column grid to plot an "axis break"
    # SMALL TOP - BIG BOTTOM
    grid = plt.GridSpec(3, 1, wspace=0.4)
    ax0 = fig.add_subplot(grid[0, 0]);  # TOP part
    ax1 = fig.add_subplot(grid[1:, 0]); # BOTTOM part
    '''
    # EQUAL SIZE TOP AND BOTTOM
    grid = plt.GridSpec(2, 1, wspace=0.4)
    ax0 = fig.add_subplot(grid[0, 0]);  # TOP part
    ax1 = fig.add_subplot(grid[1, 0]); # BOTTOM part
    '''
    '''
    # BIG TOP - SMALL BOTTOM
    grid = plt.GridSpec(3, 1, wspace=0.4)
    ax0 = fig.add_subplot(grid[:2, 0]);  # TOP part
```

```
    ax1 = fig.add_subplot(grid[2, 0]); # BOTTOM part
    '''

    # Use breakYlim to split the data and plot accordingly on each subplot
    # Plot each point individually to give each a defined color according to its related polymer
    color = cm.get_cmap('Paired', len(polName))
    for xi, yi, ci, ni in zip(x[breakYlim:], y[breakYlim:], polColour[breakYlim:], polName[breakYlim:])
:
        ax0.scatter(xi, yi, s=75, label=ni, c=color(ci))
    for xi, yi, ci, ni in zip(x[:breakYlim], y[:breakYlim], polColour[:breakYlim], polName[:breakYlim])
:
        ax1.scatter(xi, yi, s=75, label=ni, c=color(ci))

    # ZOOM-IN AND LIMIT THE VIEW TO DIFFERENT PORTIONS OF THE DATA
    dy_top = magnitude(max(y)-y.values[breakYlim])/10
    dy_bot = magnitude(y.values[breakYlim]-min(y))/10
    dx = magnitude(max(x)-min(x))

    # same x-axis limits for all subplots to be consistent with scaling
    ax0.set_xlim(min(x)-dx, max(x)+dx)
    ax1.set_xlim(min(x)-dx, max(x)+dx)

    # y-limits for the TOP part
    ax0.set_ylim(y.values[breakYlim]-dy_top, max(y)+dy_top)

    # y-limits for the BOTTOM part
    ax1.set_ylim(min(y)-dy_bot, y.values[breakYlim-1]+dy_bot)

    # hide the spines and axis between ax0 and ax1
    ax0.spines['bottom'].set_visible(False) # hide bottom border
    ax0.axes.get_xaxis().set_visible(False) # hide xaxis labels
    ax1.spines['top'].set_visible(False)
    ax1.xaxis.tick_bottom()

    ax0.yaxis.get_major_ticks()[1].label1.set_visible(False)

    # FORMAT THE AXIS BREAK GRAPHICS
    d = .0075; # how big to make the diagonal lines in axes coordinates
    d0 = d*2;  # add some offset to have the same inclination on all diagonals
    # arguments to pass to plot, just so we don't keep repeating them
    kwargs = dict(transform=ax0.transAxes, color='k', clip_on=False)
    # draw top-left diagonal
    ax0.plot((0-d, 0+d), (0-d0, 0+d0), **kwargs)
    # draw top-right diagonal
    ax0.plot((1-d, 1+d), (0-d0, 0+d0), **kwargs)
    kwargs.update(transform=ax1.transAxes)  # switch to the bottom axes
    # draw bottom-left diagonal
    ax1.plot((0-d, 0+d), (1-d, 1+d), **kwargs)
    # draw bottom-right diagonal
    ax1.plot((1-d, 1+d), (1-d, 1+d), **kwargs)

    # Vary the distance between ax0 and ax1
    fig.subplots_adjust(hspace=0.1)

    # GROUP ALL SUBPLOTS TO ADD FURTHER FORMATTING
    # add a big axis to group all, and hide its frame
    main = fig.add_subplot(111, frameon=False)
    # hide tick and tick label of the big axis
    plt.tick_params(labelcolor='none', top=False, bottom=False, left=False, right=False)

    # Display plots
    plt.xlabel(x_str + '    ' + x_units, fontsize=24);
    ax0.set_ylabel(y_str + '    ' + y_units, fontsize=24)
    ax0.yaxis.set_label_coords(-0.06, 0)
    #plt.title(fig_name, size=24);
    # add annotations (references on aech point)
    texts_ax0 = []
    for xs, ys, ss in zip(x[breakYlim:], y[breakYlim:], ref[breakYlim:]):
        texts_ax0.append(ax0.text(xs, ys, str(ss), fontsize=15))
    texts_ax1 = []
    for xs, ys, ss in zip(x[:breakYlim], y[:breakYlim], ref[:breakYlim]):
        texts_ax1.append(ax1.text(xs, ys, str(ss), fontsize=15))
    # avoid overlaps between annotations and add a linking line
    kwargs = dict(transform=ax0.transAxes)
    adjust_text(texts_ax0, ax=ax0, arrowprops=dict(arrowstyle="-", color='k', lw=0.5), save_steps=False
, **kwargs)
```

```
    kwargs = dict(transform=ax1.transAxes)
    adjust_text(texts_ax1, ax=ax1, arrowprops=dict(arrowstyle="-", color='k', lw=0.5), save_steps=False
, **kwargs)

    # Show the plot lengend to link colors and polymer names
    handles0, labels0 = ax0.get_legend_handles_labels()
    handles1, labels1 = ax1.get_legend_handles_labels()
    lgd = dict(zip(labels0+labels1, handles0+handles1))
    main.legend(lgd.values(), lgd.keys(), prop={'size': 15}, loc=legloc)

    ''' legloc CAN BE:
    Location String  Location Code
    'best'           0
    'upper right'    1
    'upper left'     2
    'lower left'     3
    'lower right'    4
    'right'          5
    'center left'    6
    'center right'   7
    'lower center'   8
    'upper center'   9
    'center'         10
    '''

    plt.show()
```

NFESdata.csv description:

Parameter_Name	Parameter_Units	Data_Type	Description
Polymer	N/A	string	polymer used in the NFES solution
Polymer Molecular Weight	$g \cdot {mol}^{-1}$	float	polymer molecular weight
Solvent	N/A	string	solvent used in the NFES solution
Solvent Surface Tension	$mN \cdot m^{-1}$	float	solvent surface tension at $298.2 K$ and $101325 Pa$
Solvent Dielectric Constant	N/A	float	solvent dielectric constant at $298.2 K$
Solvent Boiling Point	$^{\circ} C$	float	solvent boiling point
Solvent Density	$g \cdot ml^{-1}$	float	solvent relative density (water = 1) at $293.15 K$
Solvent Vapour Pressure	kPa	float	solvent vapour pressure at $293.15 K$
NFES Type	N/A	string	NFES process type/variant implemented in [reference]
Polymer Concentration	$wt\%$	float	polymer concentration used in the NFES solution
Nozzle Diameter	μm	float	inner diameter of the dispensing nozzle
Solution Deposition Rate	$\mu L \cdot h^{-1}$	float	rate at which the solution is dispensed from the reservoir
Collector Substrate	N/A	string	composition of the collector
Nozzle to Collector Distance	mm	float	distance between the dispensing nozzle and the collector
NFES Applied Voltage	V	float	applied voltage between the dispensing nozzle and the collector during NFES
NFES Stage Velocity	$mm \cdot s^{-1}$	float	velocity at which the stage/collector moves with respect to the dispensing nozzle
Fiber Diameter	nm	float	diameter of the produced fibers
Distance Between Fibers	μm	float	minimum distance achieved between two parallel fibers
Reference	N/A	string	reference author name and publication year

Give strings a numeric value

In [5]:

```
df = pd.read_csv("./NFESdata.csv", delimiter=",");

# df.loc[<ROWS RANGE> , <COLUMNS RANGE>] to get elements by index

# Assign a numeric value to string data type values
df_x = df.copy();
for col in range(len(df.columns)):
```

```
    if str(type(df.iloc[0 , col])) == "<class 'str'>":
        df_x.iloc[: , col] = number.fit_transform(df.iloc[: , col].astype('str'))

## Print column name and its data type
#print()
#for col in range(len(df.columns)):
#    print(str(df.columns[col]) + ' ' + str(type(df.iloc[0 , col])))

display(df.head(df.size));
display(df_x.head());
```

	Polymer	Polymer Molecular Weight	Solvent	Solvent Surface Tension	Solvent Dielectric Constant	Solvent Boiling Point	Solvent Density	Solvent Vapour Pressure	NFES Type	Polymer Concentration	Nozzle Diameter	D
0	Gelatin	NaN	AceticAcid	26.5555	6.1700	117.9710	1.0510	1.5200	NFES	11.00	NaN	
1	PVDF	534000.0	Acetone	22.4998	20.9000	56.2645	0.7845	24.2270	3D ES	17.00	100.0	
2	POSS-PCU	2000.0	Butanol	24.1947	17.4849	117.7000	0.8098	0.5800	EHD jetting	20.00	750.0	
3	POSS-PCL-PCU	2000.0	Butanol	24.1947	17.4849	117.7000	0.8098	0.5800	EHD jetting	20.00	750.0	
4	POSS-PCU	2000.0	Dimethylacetamide DMAC	34.0000	23.0000	165.0000	0.9366	0.3300	EHD jetting	20.00	750.0	
5	POSS-PCL-PCU	2000.0	Dimethylacetamide DMAC	34.0000	23.0000	165.0000	0.9366	0.3300	EHD jetting	20.00	750.0	
6	PLGA	NaN	Dimethylcarbonate DMC	28.3000	3.0870	90.5000	1.0636	7.4000	TPES	NaN	NaN	
7	PVDF	440000.0	Dimethylformamide DMF	36.4200	36.7000	152.8000	0.9445	0.4900	Helix EHD	18.00	260.0	
8	PEO-TBF	4000000.0	Dimethylformamide DMF	36.4200	36.7000	152.8000	0.9445	0.4900	NFES	0.75	NaN	
9	PVDF	534000.0	Dimethylsulfoxide DMSO	42.8600	46.7000	189.0000	1.1010	0.0493	3D ES	17.00	100.0	
10	PEO	30000.0	Ethanol	21.8433	24.5000	78.2400	0.7893	5.8000	Blow EDW	8.00	210.0	
11	PEO	300000.0	Ethanol	21.8433	24.5000	78.2400	0.7893	5.8000	NFES	16.00	40.0	
12	PEO	300000.0	Ethanol	21.8433	24.5000	78.2400	0.7893	5.8000	NFES	18.00	40.0	
13	PEO	300000.0	Ethanol	21.8433	24.5000	78.2400	0.7893	5.8000	EDW	14.00	210.0	
14	PEO	200000.0	Ethanol	21.8433	24.5000	78.2400	0.7893	5.8000	Suspension	14.00	250.0	
15	Gelatin	NaN	Ethylacetate	23.1700	6.0200	77.1000	0.9020	10.0000	NFES	11.00	NaN	
16	PEO	300000.0	NaN	NaN	NaN	NaN	NaN	NaN	NFES	3.00	NaN	
17	PEO	4000000.0	NaN	NaN	NaN	NaN	NaN	NaN	NFES	2.00	150.0	
18	PVK	1100000.0	Styrene	30.7800	2.4700	145.3000	0.9060	0.6700	NFES	3.96	100.0	
19	PVK	1100000.0	Styrene	30.7800	2.4700	145.3000	0.9060	0.6700	NFES	3.96	100.0	
20	PS	NaN	1,2,4-trichlorobenzene TCB	38.5400	6.7500	213.5000	1.4590	40.0000	EHD jetting	3.00	2.0	
21	MEH-PPV	380000.0	toluene	28.5300	2.3800	110.6000	0.8623	3.8000	typical	0.08	260.0	
22	PEO	300000.0	toluene	28.5300	2.3800	110.6000	0.8623	3.8000	typical	0.08	260.0	
23	MEH-PPV	380000.0	toluene	28.5300	2.3800	110.6000	0.8623	3.8000	typical	0.08	260.0	
24	PEO	300000.0	toluene	28.5300	2.3800	110.6000	0.8623	3.8000	typical	0.08	260.0	
25	PEO	4000000.0	Water	72.8000	80.1000	99.9740	1.0000	2.3393	LV NFES	1.00	210.0	

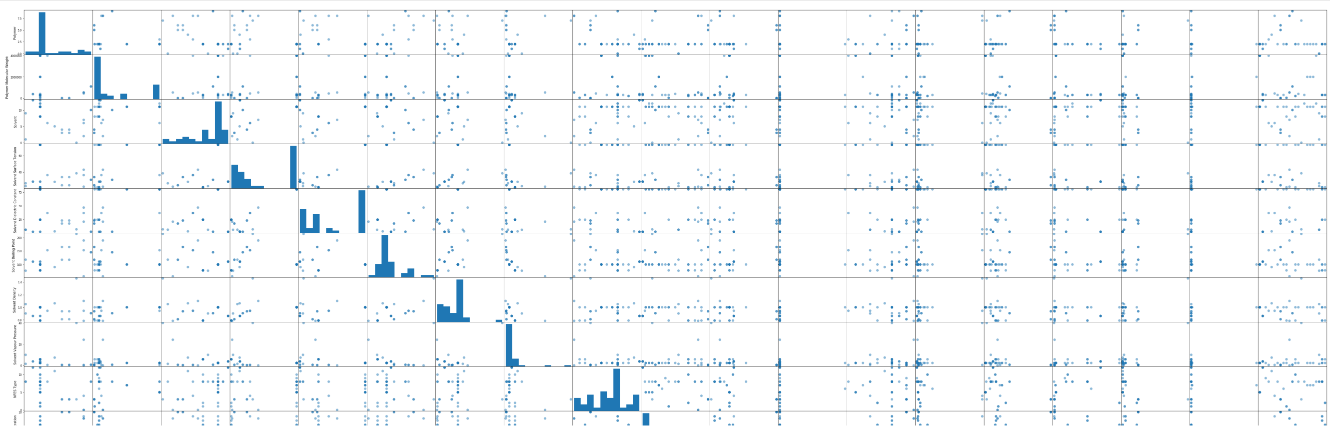
	Polymer	Polymer Molecular Weight	Solvent	Solvent Surface Tension	Solvent Dielectric Constant	Solvent Boiling Point	Solvent Density	Solvent Vapour Pressure	NFES Type	Polymer Concentration	Nozzle Diameter	D
26	PEO	4000000.0	Water	72.8000	80.1000	99.9740	1.0000	2.3393	LVNFES	2.00	210.0	
27	PEO	4000000.0	Water	72.8000	80.1000	99.9740	1.0000	2.3393	LVNFES	2.00	210.0	
28	PEO	4000000.0	Water	72.8000	80.1000	99.9740	1.0000	2.3393	LVNFES	1.00	210.0	
29	PEO	4000000.0	Water	72.8000	80.1000	99.9740	1.0000	2.3393	LVNFES	2.00	210.0	
30	PEO	4000000.0	Water	72.8000	80.1000	99.9740	1.0000	2.3393	LVNFES	3.00	210.0	
31	PEO	300000.0	Water	72.8000	80.1000	99.9740	1.0000	2.3393	Scanning Tip	7.00	100.0	
32	PEO	300000.0	Water	72.8000	80.1000	99.9740	1.0000	2.3393	MES	6.00	NaN	
33	PEO	30000.0	Water	72.8000	80.1000	99.9740	1.0000	2.3393	Blow EDW	8.00	210.0	
34	PEO	2000000.0	Water	72.8000	80.1000	99.9740	1.0000	2.3393	MultiNozz	5.00	NaN	
35	PEO	2000000.0	Water	72.8000	80.1000	99.9740	1.0000	2.3393	MultiNozz	5.00	180.0	
36	PEO	2000000.0	Water	72.8000	80.1000	99.9740	1.0000	2.3393	MultiNozz	5.00	180.0	
37	PEO	300000.0	Water	72.8000	80.1000	99.9740	1.0000	2.3393	NFES	16.00	40.0	
38	PEO	300000.0	Water	72.8000	80.1000	99.9740	1.0000	2.3393	NFES	18.00	40.0	
39	PEO	300000.0	Water	72.8000	80.1000	99.9740	1.0000	2.3393	EDW	14.00	210.0	
40	PEO	200000.0	Water	72.8000	80.1000	99.9740	1.0000	2.3393	Suspension	14.00	250.0	
41	PEO	400000.0	Water	72.8000	80.1000	99.9740	1.0000	2.3393	NFES	10.00	108.0	
42	PEO	300000.0	Water	72.8000	80.1000	99.9740	1.0000	2.3393	NFES	8.00	400.0	

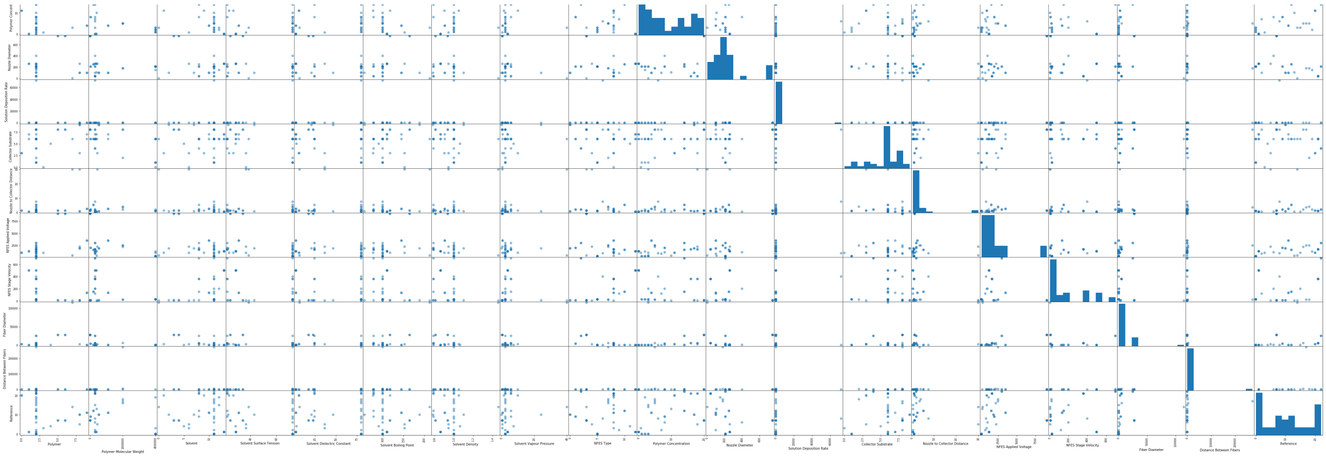
	Polymer	Polymer Molecular Weight	Solvent	Solvent Surface Tension	Solvent Dielectric Constant	Solvent Boiling Point	Solvent Density	Solvent Vapour Pressure	NFES Type	Polymer Concentration	Nozzle Diameter	Solution Deposition Rate	Collecto Substrate
0	0	NaN	1	26.5555	6.1700	117.9710	1.0510	1.520	8	11.0	NaN	NaN	[illegible]
1	8	534000.0	2	22.4998	20.9000	56.2645	0.7845	24.227	0	17.0	100.0	0.84	[illegible]
2	6	2000.0	3	24.1947	17.4849	117.7000	0.8098	0.580	3	20.0	750.0	60.00	[illegible]
3	5	2000.0	3	24.1947	17.4849	117.7000	0.8098	0.580	3	20.0	750.0	60.00	[illegible]
4	6	2000.0	4	34.0000	23.0000	165.0000	0.9366	0.330	3	20.0	750.0	60.00	[illegible]

Correlation Matrix

In [6]:

```
scale = 24;
pd.plotting.scatter_matrix(df_x, alpha=0.5, figsize=(3*scale, 2*scale), s=scale*10)
plt.show()
```





In [7]:

```
# all '-1's are to remove 'Reference' from the Correlation Matrix
corrMatrix = df_x.iloc[:, :-1].corr()
display(corrMatrix.style.background_gradient(cmap='viridis'))
```

	Polymer	Polymer Molecular Weight	Solvent	Solvent Surface Tension	Solvent Dielectric Constant	Solvent Boiling Point	Solvent Density	Solvent Vapour Pressure	NFES Type	Polymer Concentration	D
Polymer	1	-0.195529	-0.508548	-0.325552	-0.349413	0.60224	0.0866076	0.213883	-0.412241	0.31068	0.
Polymer Molecular Weight	-0.195529	1	0.254333	0.454131	0.436777	-0.0294683	0.412449	-0.132867	-0.0235542	-0.589623	0.
Solvent	-0.508548	0.254333	1	0.531	0.464396	-0.382652	-0.119187	-0.438474	0.470837	-0.566223	0.
Solvent Surface Tension	-0.325552	0.454131	0.531	1	0.951059	-0.119883	0.494901	-0.237112	-0.0605943	-0.282838	0.
Solvent Dielectric Constant	-0.349413	0.436777	0.464396	0.951059	1	-0.246079	0.329352	-0.285717	-0.193681	-0.106421	-(
Solvent Boiling Point	0.60224	-0.0294683	-0.382652	-0.119883	-0.246079	1	0.538262	0.119483	-0.193306	0.0123328	0.
Solvent Density	0.0866076	0.412449	-0.119187	0.494901	0.329352	0.538262	1	0.379391	-0.110497	-0.260983	0.
Solvent Vapour Pressure	0.213883	-0.132867	-0.438474	-0.237112	-0.285717	0.119483	0.379391	1	-0.176309	-0.0501446	0.
NFES Type	-0.412241	-0.0235542	0.470837	-0.0605943	-0.193681	-0.193306	-0.110497	-0.176309	1	-0.421728	0.
Polymer Concentration	0.31068	-0.589623	-0.566223	-0.282838	-0.106421	0.0123328	-0.260983	-0.0501446	-0.421728	1	(
Nozzle Diameter	0.140809	-0.217249	-0.388678	-0.243367	-0.22198	0.217331	-0.273021	-0.323479	-0.213949	0.35787	
Solution Deposition Rate	-0.107235	-0.115792	0.0985082	0.194133	0.188893	-0.0721396	0.145042	-0.0342806	0.0138823	-0.0625564	0.
Collector Substrate	0.316376	-0.546839	-0.0969222	-0.14717	-0.151791	0.0943243	-0.128737	0.0870418	-0.0851616	0.302978	0.
Nozzle to Collector Distance	0.39669	-0.0998979	-0.377631	-0.0762387	-0.120195	0.506841	0.4858	0.506007	-0.175647	0.057618	0.
NFES Applied Voltage	0.449574	-0.415238	-0.497165	-0.369287	-0.331356	0.307226	-0.307351	-0.217749	-0.345986	0.606318	0.
NFES Stage Velocity	-0.398348	-0.365493	0.407842	-0.190986	-0.219783	-0.280908	-0.29183	-0.068592	0.424013	-0.0672371	0.
Fiber Diameter	0.392098	-0.359426	-0.635928	-0.177031	-0.232586	0.661911	0.53869	0.793935	-0.435602	0.188283	0.

	Polymer	Polymer Molecular Weight	Solvent	Solvent Surface Tension	Solvent Dielectric Constant	Solvent Boiling Point	Solvent Density	Solvent Vapour Pressure	NFES Type	Polymer Concentration	D
Distance Between Fibers	-0.07[illegible]	0.304476	0.0979352	0.272454	0.268365	0.08[illegible]	0.2[illegible]	-0.04947	0.0113388	-0.1[illegible]	-[illegible]

In [8]:

```
'''
# Multivariate Linear Regression
df_x = df.sort_values(by=['Fiber Diameter'])
df_x = df_x.dropna(subset=['Polymer Concentration','Nozzle Diameter','NFES Applied Voltage','Fiber Diam
eter'])

# X is the independent variable (bivariate in this case)
X = np.array([df_x.iloc[:]['Polymer Concentration'], df_x.iloc[:]['Nozzle Diameter'], df_x.iloc[:]['NFE
S Applied Voltage']])

# Y is the dependent data
Y = df_x.iloc[:]['Fiber Diameter']

# predict is an independent variable for which we'd like to predict the value
predict= [[20.0, 750.0, 9000.0]]

# generate a model of polynomial features
poly = PolynomialFeatures(degree=2)

# transform the x data for proper fitting (for single variable type it returns,[1,x,x**2])
X_ = poly.fit_transform(X)

# transform the prediction to fit the model type
predict_ = poly.fit_transform(predict)

# generate the regression object
clf = LinearRegression()

# perform the actual regression
clf = clf.fit(np.transpose(X), Y)

print('\n')
print('>>> INTERCEPT & COEFFICIENTS')
print(clf.intercept_)
print(clf.coef_)
print('\n')
print('>>> PREDICTION')
print("Prediction = " + str(clf.predict(predict)))
print('\n')
'''
```

Out[8]:

```
'\n# Multivariate Linear Regression\ndf_x = df.sort_values(by=[\'Fiber Diameter\'])\ndf_x = df_x.dropna
(subset=[\'Polymer Concentration\',\'Nozzle Diameter\',\'NFES Applied Voltage\',\'Fiber Diameter\'])\n\
n# X is the independent variable (bivariate in this case)\nX = np.array([df_x.iloc[:][\'Polymer Concent
ration\'], df_x.iloc[:][\'Nozzle Diameter\'], df_x.iloc[:][\'NFES Applied Voltage\']])\n\n# Y is the de
pendent data\nY = df_x.iloc[:][\'Fiber Diameter\']\n\n# predict is an independent variable for which we
\'d like to predict the value\npredict= [[20.0, 750.0, 9000.0]]\n\n# generate a model of polynomial fea
tures\npoly = PolynomialFeatures(degree=2)\n\n# transform the x data for proper fitting (for single var
iable type it returns,[1,x,x**2])\nX_ = poly.fit_transform(X)\n\n# transform the prediction to fit the 
model type\npredict_ = poly.fit_transform(predict)\n\n# generate the regression object\nclf = LinearReg
ression()\n\n# perform the actual regression\nclf = clf.fit(np.transpose(X), Y)\n\nprint(\'\\n\')\nprint
(\'>>> INTERCEPT & COEFFICIENTS\')\nprint(clf.intercept_)\nprint(clf.coef_)\nprint(\'\\n\')\nprint(\'>>>
PREDICTION\')\nprint("Prediction = " + str(clf.predict(predict)))\nprint(\'\\n\')\n'
```

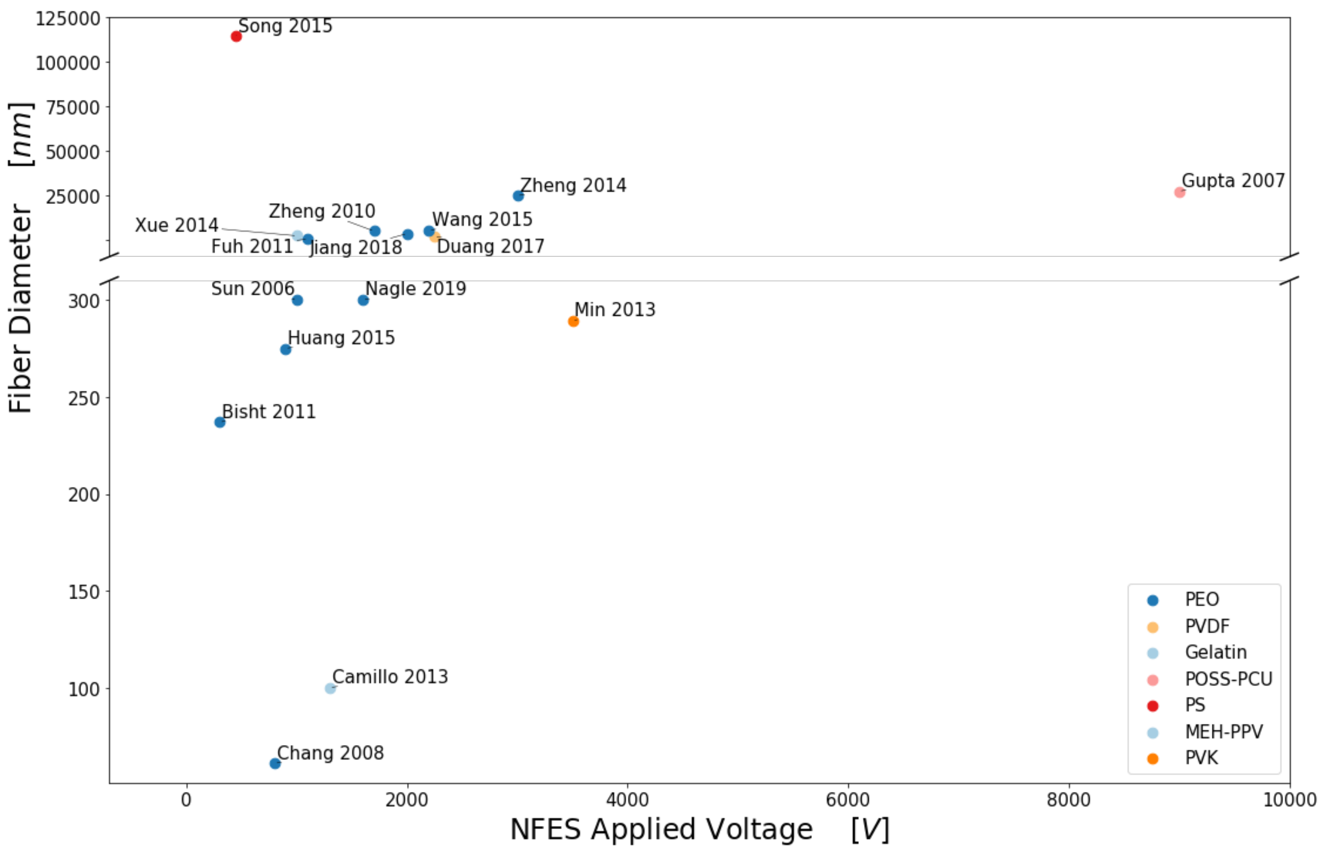
In [9]:

```
# PLOT FIG
fig_name = 'Figure 1';
x_str = 'NFES Applied Voltage'; x_units = r'$[V]$';
y_str = 'Fiber Diameter'; y_units = r'$[nm]$';
breakYlim = 7;

scatterPlot_breakAxis(x_str, x_units, y_str, y_units, df, df_x, breakYlim, 'lower right');
#scatterPlot(x_str, x_units, y_str, y_units, df, 'original: ' + fig_name)
```

```
>>> new df
```

	NFES Applied Voltage	Fiber Diameter	Reference	Polymer	Polymer Name
31	800.0	61.00	Chang 2008	2	PEO
21	1300.0	100.00	Camillo 2013	1	MEH-PPV
25	300.0	237.00	Bisht 2011	2	PEO
32	900.0	275.00	Huang 2015	2	PEO
18	3500.0	289.26	Min 2013	9	PVK
14	1600.0	300.00	Nagle 2019	2	PEO
16	1000.0	300.00	Sun 2006	2	PEO
42	1100.0	740.00	Fuh 2011	2	PEO
7	2250.0	2250.00	Duang 2017	8	PVDF
0	1000.0	2500.00	Xue 2014	0	Gelatin
10	2000.0	3730.00	Jiang 2018	2	PEO
11	1700.0	5150.00	Zheng 2010	2	PEO
34	2200.0	5470.00	Wang 2015	2	PEO
13	3000.0	25000.00	Zheng 2014	2	PEO
2	9000.0	27500.00	Gupta 2007	6	POSS-PCU
20	450.0	115000.00	Song 2015	7	PS



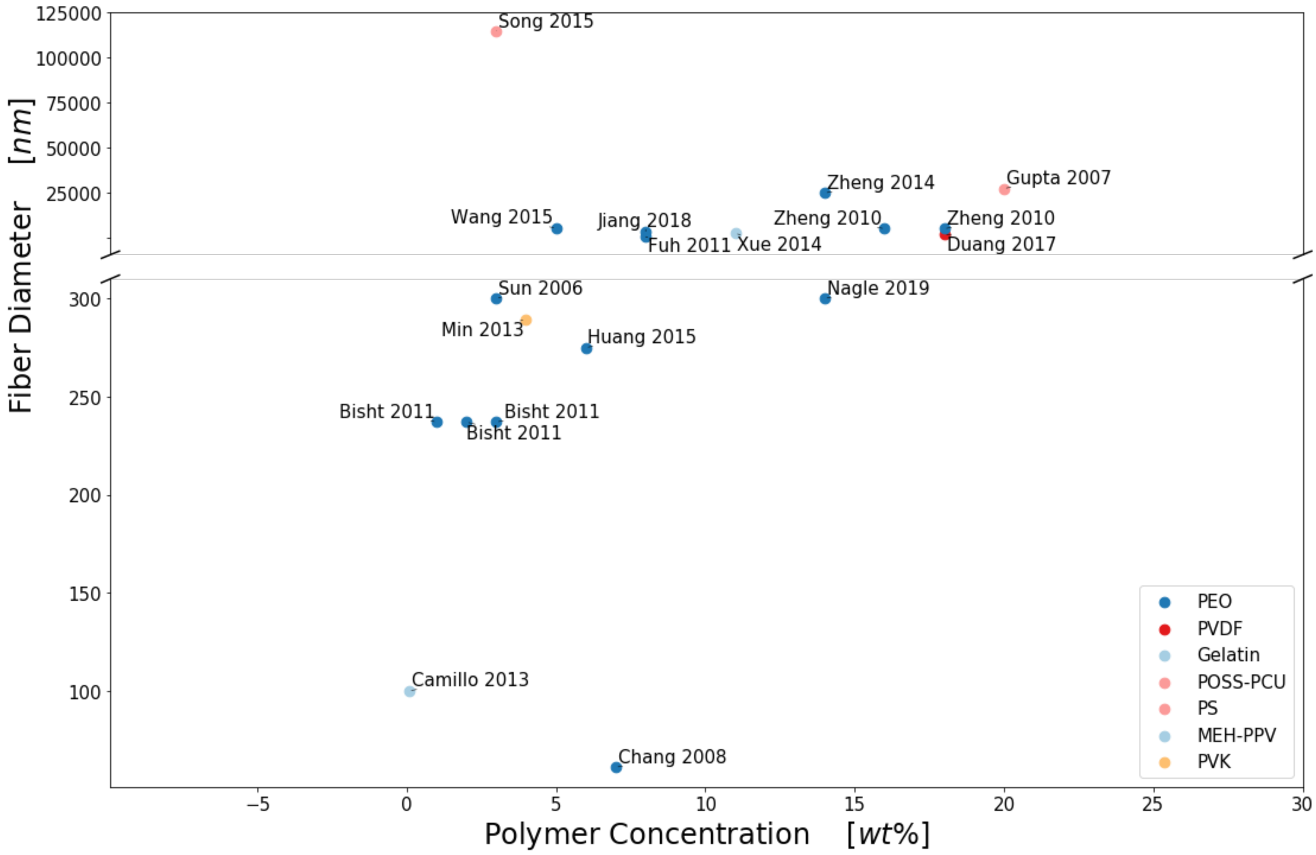
In [10]:

```
# PLOT FIG
fig_name = 'Figure 1';
x_str = 'Polymer Concentration'; x_units = r'$[wt\%]$';
y_str = 'Fiber Diameter'; y_units = r'$[nm]$';
breakYlim = 9;

scatterPlot_breakAxis(x_str, x_units, y_str, y_units, df, df_x, breakYlim, 'lower right');
#scatterPlot(x_str, x_units, y_str, y_units, df, 'original: ' + fig_name)
```

>>> new df

	Polymer Concentration	Fiber Diameter	Reference	Polymer	Polymer Name
31	7.00	61.00	Chang 2008	2	PEO
21	0.08	100.00	Camillo 2013	1	MEH-PPV
27	3.00	237.00	Bisht 2011	2	PEO
26	2.00	237.00	Bisht 2011	2	PEO
25	1.00	237.00	Bisht 2011	2	PEO
32	6.00	275.00	Huang 2015	2	PEO
18	3.96	289.26	Min 2013	9	PVK
14	14.00	300.00	Nagle 2019	2	PEO
16	3.00	300.00	Sun 2006	2	PEO
42	8.00	740.00	Fuh 2011	2	PEO
7	18.00	2250.00	Duang 2017	8	PVDF
0	11.00	2500.00	Xue 2014	0	Gelatin
10	8.00	3730.00	Jiang 2018	2	PEO
12	18.00	5150.00	Zheng 2010	2	PEO
11	16.00	5150.00	Zheng 2010	2	PEO
34	5.00	5470.00	Wang 2015	2	PEO
13	14.00	25000.00	Zheng 2014	2	PEO
2	20.00	27500.00	Gupta 2007	6	POSS-PCU
20	3.00	115000.00	Song 2015	7	PS



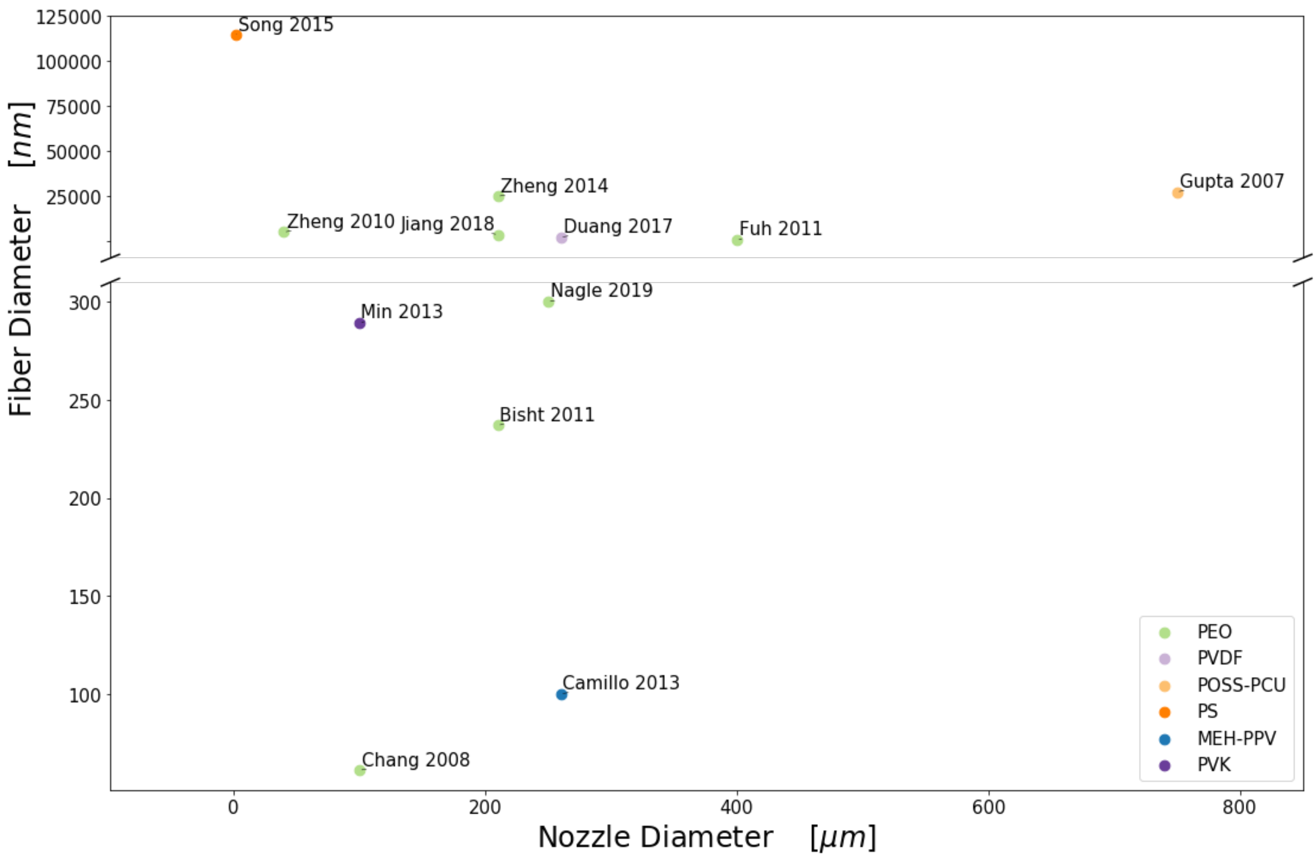
In [11]:

```
# PLOT FIG
fig_name = 'Figure 1';
x_str = 'Nozzle Diameter'; x_units = r'$[\mu m]$';
y_str = 'Fiber Diameter'; y_units = r'$[nm]$';
breakYlim = 5;
```

```
scatterPlot_breakAxis(x_str, x_units, y_str, y_units, df, df_x, breakYlim, 'lower right');
#scatterPlot(x_str, x_units, y_str, y_units, df, 'original: ' + fig_name)
```

```
>>> new_df
```

	Nozzle Diameter	Fiber Diameter	Reference	Polymer	Polymer Name
31	100.0	61.00	Chang 2008	2	PEO
21	260.0	100.00	Camillo 2013	1	MEH-PPV
25	210.0	237.00	Bisht 2011	2	PEO
18	100.0	289.26	Min 2013	9	PVK
14	250.0	300.00	Nagle 2019	2	PEO
42	400.0	740.00	Fuh 2011	2	PEO
7	260.0	2250.00	Duang 2017	8	PVDF
10	210.0	3730.00	Jiang 2018	2	PEO
11	40.0	5150.00	Zheng 2010	2	PEO
13	210.0	25000.00	Zheng 2014	2	PEO
2	750.0	27500.00	Gupta 2007	6	POSS-PCU
20	2.0	115000.00	Song 2015	7	PS



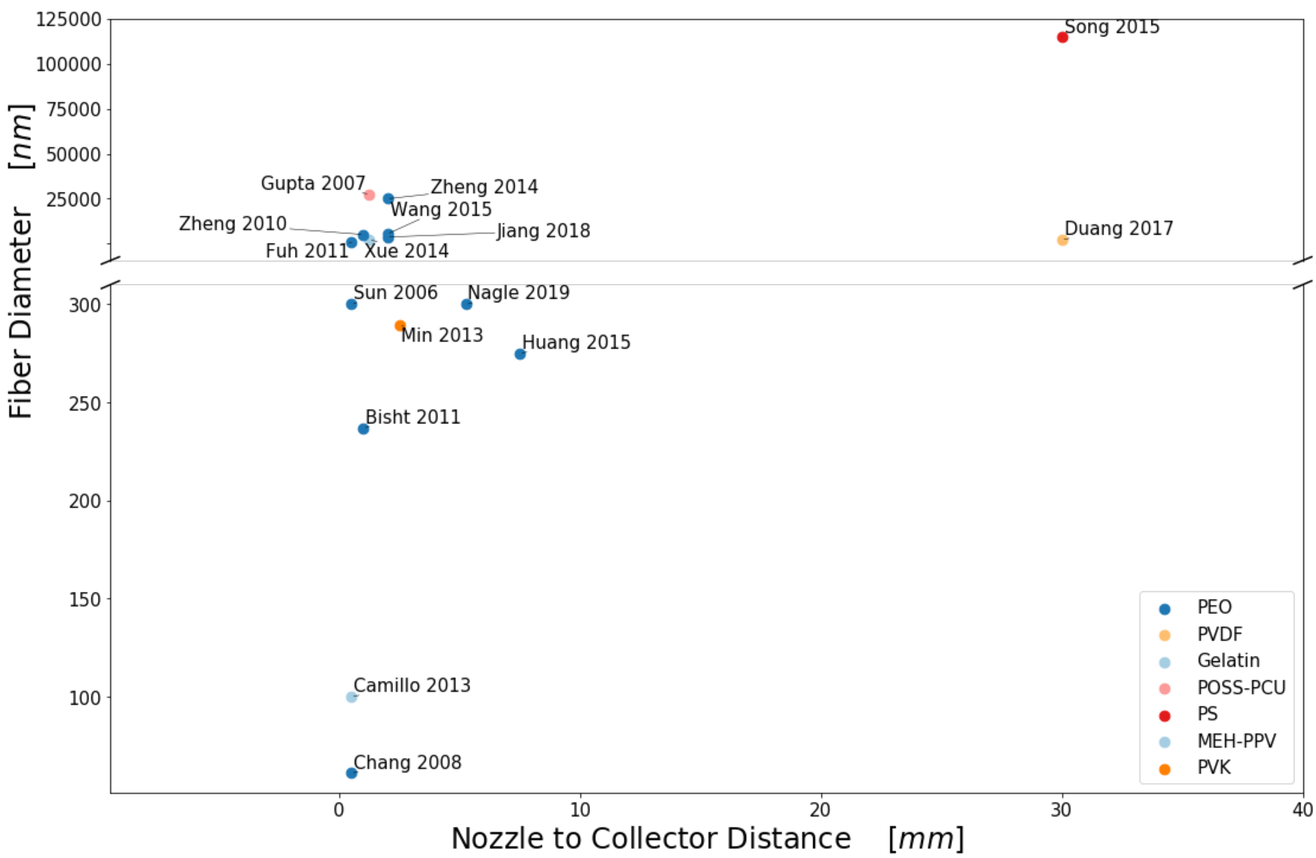
In [12]:

```
# PLOT FIG
fig_name = 'Figure 1';
x_str = 'Nozzle to Collector Distance'; x_units = r'$[mm]$';
y_str = 'Fiber Diameter'; y_units = r'$[nm]$';
breakYlim = 7;

scatterPlot_breakAxis(x_str, x_units, y_str, y_units, df, df_x, breakYlim, 'lower right');
#scatterPlot(x_str, x_units, y_str, y_units, df, 'original: ' + fig_name)
```

```
>>> new_df
```

	Nozzle to Collector Distance	Fiber Diameter	Reference	Polymer	Polymer Name
31	0.50	61.00	Chang 2008	2	PEO
21	0.50	100.00	Camillo 2013	1	MEH-PPV
25	1.00	237.00	Bisht 2011	2	PEO
32	7.50	275.00	Huang 2015	2	PEO
18	2.50	289.26	Min 2013	9	PVK
14	5.25	300.00	Nagle 2019	2	PEO
16	0.50	300.00	Sun 2006	2	PEO
42	0.50	740.00	Fuh 2011	2	PEO
7	30.00	2250.00	Duang 2017	8	PVDF
0	1.25	2500.00	Xue 2014	0	Gelatin
10	2.00	3730.00	Jiang 2018	2	PEO
11	1.00	5150.00	Zheng 2010	2	PEO
34	2.00	5470.00	Wang 2015	2	PEO
13	2.00	25000.00	Zheng 2014	2	PEO
2	1.25	27500.00	Gupta 2007	6	POSS-PCU
20	30.00	115000.00	Song 2015	7	PS



In [ ]: