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
```

```
# 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"}))
```

```python
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])

# 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)
```

```python
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
    '''

    # Display main plot
    plt.show()

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

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

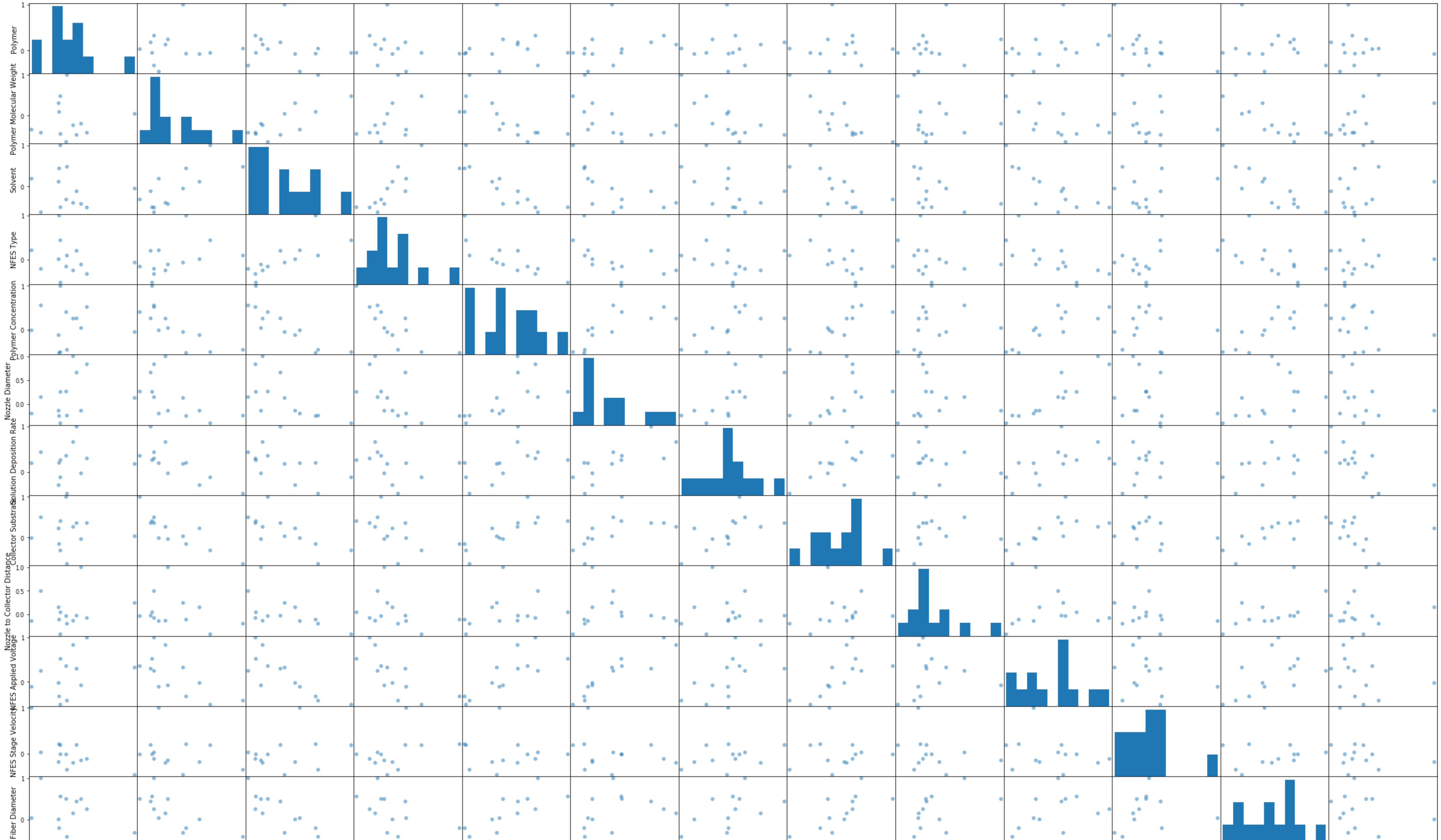
	Polymer	Polymer Molecular Weight	Solvent	Solvent Surface Tension	Solvent Dielectric Constant	Solvent Boiling Point	Solvent Density	Solvent Vapour Pressure	NFES Type	Polymer Concentration	Nozzle Diameter	Solution Deposition Rate	Collector Substrate	Nozzle to Collector Distance	NFES Applied Voltage	NFES Stage Velocity	Fiber Diameter	Distance Between Fibers	Reference
0	Gelatin	NaN	AceticAcid	26.5555	6.1700	117.9710	1.0510	1.520	NFES	11.0	NaN	NaN	PDMS	1.25	1000.0	NaN	2500.0	40.0	Xue 2014
1	PVDF	534000.0	Acetone	22.4998	20.9000	56.2645	0.7845	24.227	3D ES	17.0	100.0	0.84	paper	0.75	1900.0	10.0	NaN	NaN	Kim 2018
2	POSS-PCU	2000.0	Butanol	24.1947	17.4849	117.7000	0.8098	0.580	EHD jetting	20.0	750.0	60.00	NaN	1.25	9000.0	10.0	27500.0	250.0	Gupta 2007
3	POSS-PCL-PCU	2000.0	Butanol	24.1947	17.4849	117.7000	0.8098	0.580	EHD jetting	20.0	750.0	60.00	NaN	1.25	9000.0	10.0	27500.0	250.0	Gupta 2007
4	POSS-PCU	2000.0	Dimethylacetamide DMAC	34.0000	23.0000	165.0000	0.9366	0.330	EHD jetting	20.0	750.0	60.00	NaN	1.25	9000.0	10.0	27500.0	250.0	Gupta 2007

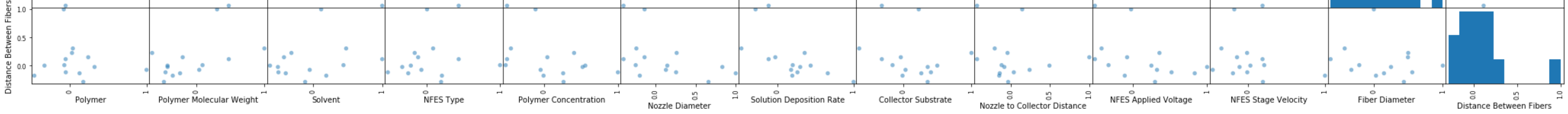
	Polymer	Polymer Molecular Weight	Solvent	Solvent Surface Tension	Solvent Dielectric Constant	Solvent Boiling Point	Solvent Density	Solvent Vapour Pressure	NFES Type	Polymer Concentration	Nozzle Diameter	Solution Deposition Rate	Collector Substrate	Nozzle to Collector Distance	NFES Applied Voltage	NFES Stage Velocity	Fiber Diameter	Distance Between Fibers	Reference
0	0	NaN	1	26.5555	6.1700	117.9710	1.0510	1.520	8	11.0	NaN	NaN	3	1.25	1000.0	NaN	2500.0	40.0	23
1	10	534000.0	2	22.4998	20.9000	56.2645	0.7845	24.227	0	17.0	100.0	0.84	9	0.75	1900.0	10.0	NaN	NaN	11
2	8	2000.0	3	24.1947	17.4849	117.7000	0.8098	0.580	3	20.0	750.0	60.00	8	1.25	9000.0	10.0	27500.0	250.0	8
3	7	2000.0	3	24.1947	17.4849	117.7000	0.8098	0.580	3	20.0	750.0	60.00	8	1.25	9000.0	10.0	27500.0	250.0	8
4	8	2000.0	4	34.0000	23.0000	165.0000	0.9366	0.330	3	20.0	750.0	60.00	8	1.25	9000.0	10.0	27500.0	250.0	8

Correlation Matrix

In [6]:

```
scale = 12;
corrMatrix = df_x.drop(["Solvent Surface Tension", "Solvent Dielectric Constant", "Solvent Boiling Point", "Solvent Density", "Solvent Vapour Pressure", "Reference"],
axis=1).corr()
pd.plotting.scatter_matrix(corrMatrix, alpha=0.5, figsize=(3*scale, 2*scale), s=scale*10)
plt.show()
display(corrMatrix.style.background_gradient(cmap='viridis'))
```



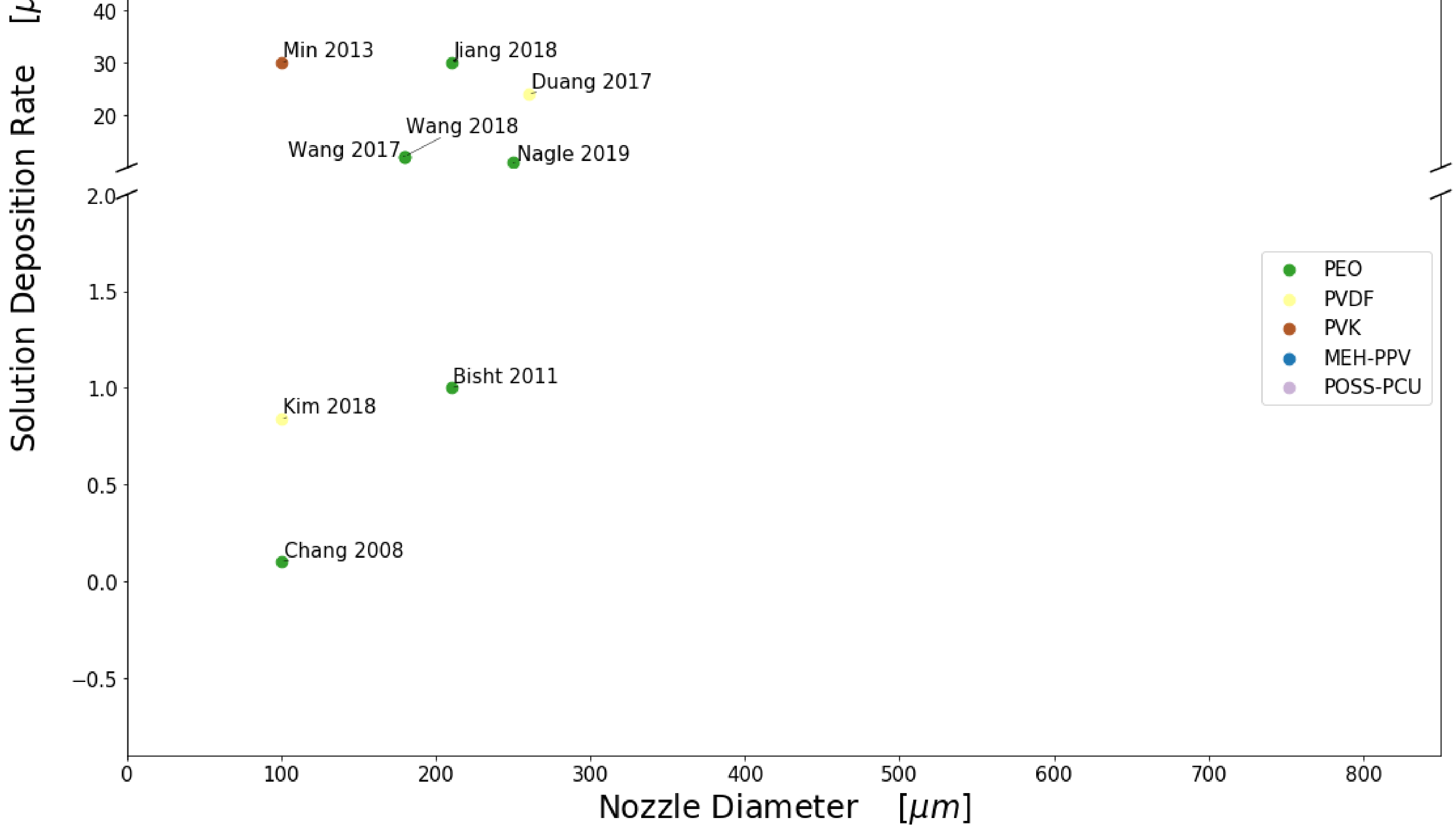


	Polymer	Polymer Molecular Weight	Solvent	NFES Type	Polymer Concentration	Nozzle Diameter	Solution Deposition Rate	Collector Substrate	Nozzle to Collector Distance	NFES Applied Voltage	NFES Stage Velocity	Fiber Diameter	Distance Between Fibers
Polymer	1	0.0428118	-0.0510177	-0.0680483	-0.0491501	0.129933	0.178395	0.0317516	0.242419	0.324557	-0.458095	-0.325883	-0.0751442
Polymer Molecular Weight	0.0428118	1	0.476861	0.0912632	-0.452465	-0.238393	-0.475521	-0.648674	-0.199924	-0.42197	-0.345799	-0.422463	0.305256
Solvent	-0.0510177	0.476861	1	0.438269	-0.504662	-0.398241	-0.117123	-0.316495	-0.42409	-0.511809	0.188832	-0.626835	0.115133
NFES Type	-0.0680483	0.0912632	0.438269	1	-0.523583	-0.24655	0.19984	-0.158982	-0.112428	-0.32799	0.210757	-0.211196	0.00874078
Polymer Concentration	-0.0491501	-0.452465	-0.504662	-0.523583	1	0.256802	0.261842	0.405099	0.0422222	0.520597	-0.00954965	0.557077	-0.118291
Nozzle Diameter	0.129933	-0.238393	-0.398241	-0.24655	0.256802	1	0.660487	0.263588	-0.136423	0.832933	-0.195392	0.147306	-0.13579
Solution Deposition Rate	0.178395	-0.475521	-0.117123	0.19984	0.261842	0.660487	1	0.356602	-0.0292881	0.298177	0.19395	0.435519	-0.289193
Collector Substrate	0.0317516	-0.648674	-0.316495	-0.158982	0.405099	0.263588	0.356602	1	-0.0374787	0.35755	-0.0122234	0.498273	0.225484
Nozzle to Collector Distance	0.242419	-0.199924	-0.42409	-0.112428	0.0422222	-0.136423	-0.0292881	-0.0374787	1	-0.0779359	-0.142671	0.4945	0.149042
NFES Applied Voltage	0.324557	-0.42197	-0.511809	-0.32799	0.520597	0.832933	0.298177	0.35755	-0.0779359	1	-0.109636	0.249073	-0.0239977
NFES Stage Velocity	-0.458095	-0.345799	0.188832	0.210757	-0.00954965	-0.195392	0.19395	-0.0122234	-0.142671	-0.109636	1	0.0311531	-0.178259
Fiber Diameter	-0.325883	-0.422463	-0.626835	-0.211196	0.557077	0.147306	0.435519	0.498273	0.4945	0.249073	0.0311531	1	0.000311073
Distance Between Fibers	-0.0751442	0.305256	0.115133	0.00874078	-0.118291	-0.13579	-0.289193	0.225484	0.149042	-0.0239977	-0.178259	0.000311073	1

In [30]:

```
# PLOT FIG
x_str = 'Nozzle Diameter'; x_units = r'$[\mu m]$';
y_str = 'Solution Deposition Rate'; y_units = r'$[\mu L \cdot h^{-1}]$';
breakYlim = 3;

#scatterPlot(x_str, x_units, y_str, y_units, df, 'original')
scatterPlot_breakAxis(x_str, x_units, y_str, y_units, df, df_x, breakYlim, 'center right');
```




```
>>> new_df
```

	Nozzle Diameter	Solution Deposition Rate	Reference	Polymer	Polymer Name
31	100.0	0.10	Chang 2008	3	PEO
1	100.0	0.84	Kim 2018	10	PVDF
25	210.0	1.00	Bisht 2011	3	PEO
14	250.0	10.80	Nagle 2019	3	PEO
35	180.0	12.00	Wang 2017	3	PEO
36	180.0	12.00	Wang 2018	3	PEO
7	260.0	24.00	Duang 2017	10	PVDF
10	210.0	30.00	Jiang 2018	3	PEO
18	100.0	30.00	Min 2013	11	PVK

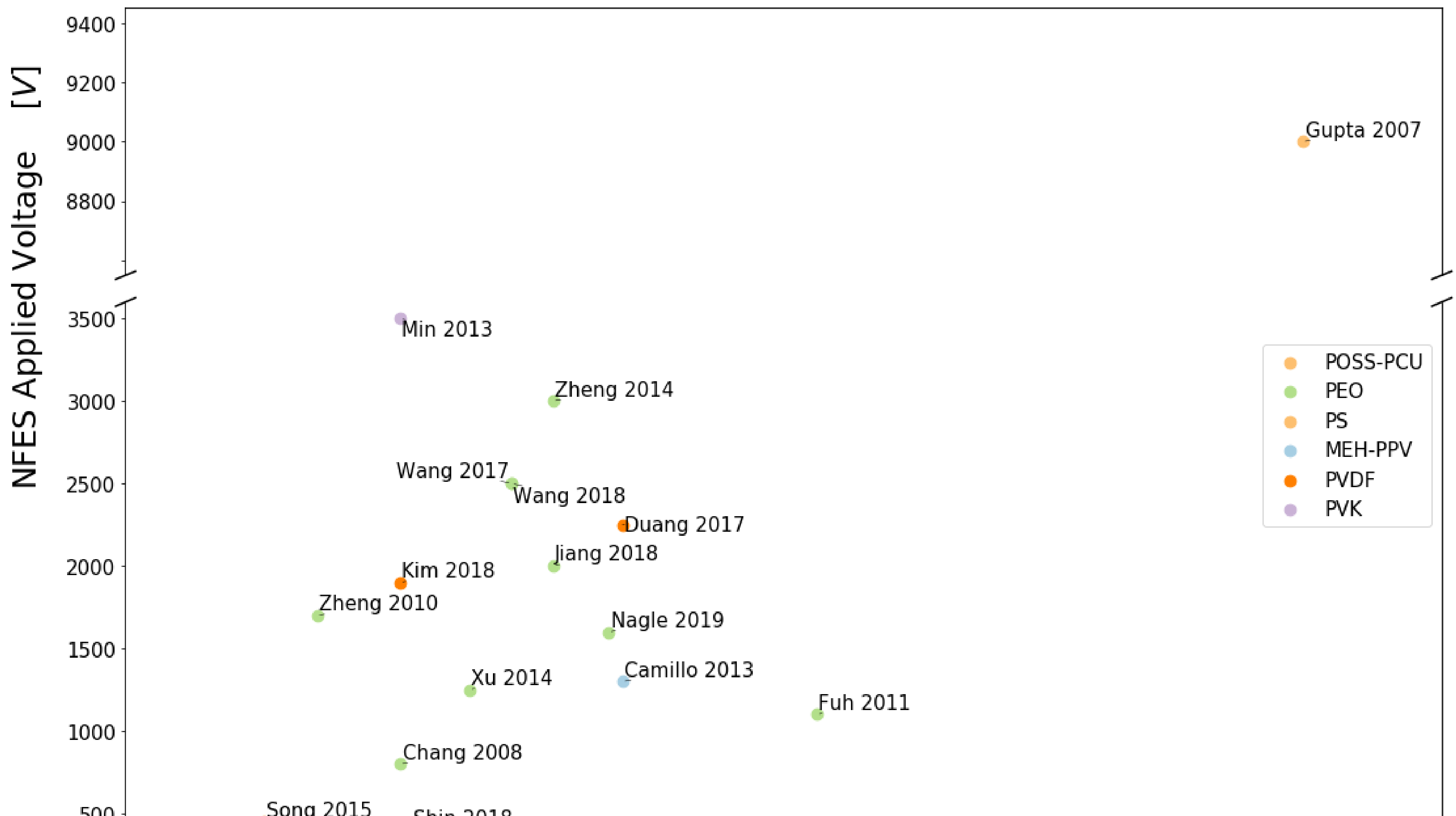
	Nozzle Diameter	Solution Deposition Rate	Reference	Polymer	Polymer Name
13	210.0	50.00	Zheng 2014	3	PEO
21	260.0	50.00	Camillo 2013	1	MEH-PPV
2	750.0	60.00	Gupta 2007	8	POSS-PCU

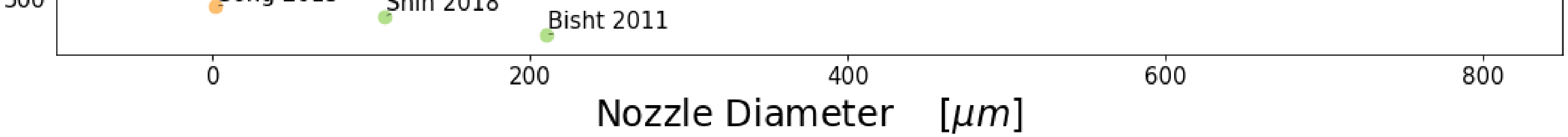
In [21]:

```
# PLOT FIG
x_str = 'Nozzle Diameter'; x_units = r'$[\mu m]$';
y_str = 'NFES Applied Voltage'; y_units = r'$[V]$';
breakYlim = 16;

#scatterPlot(x_str, x_units, y_str, y_units, df, 'original')
scatterPlot_breakAxis(x_str, x_units, y_str, y_units, df, df_x, breakYlim, 'center right');
```

```
C:\Users\oskat\Anaconda3\lib\site-packages\ipykernel_launcher.py:78: UserWarning: Attempting to set identical bottom == top == 9000.0 results in singular transformati
ons; automatically expanding.
```






```
>>> new_df
```

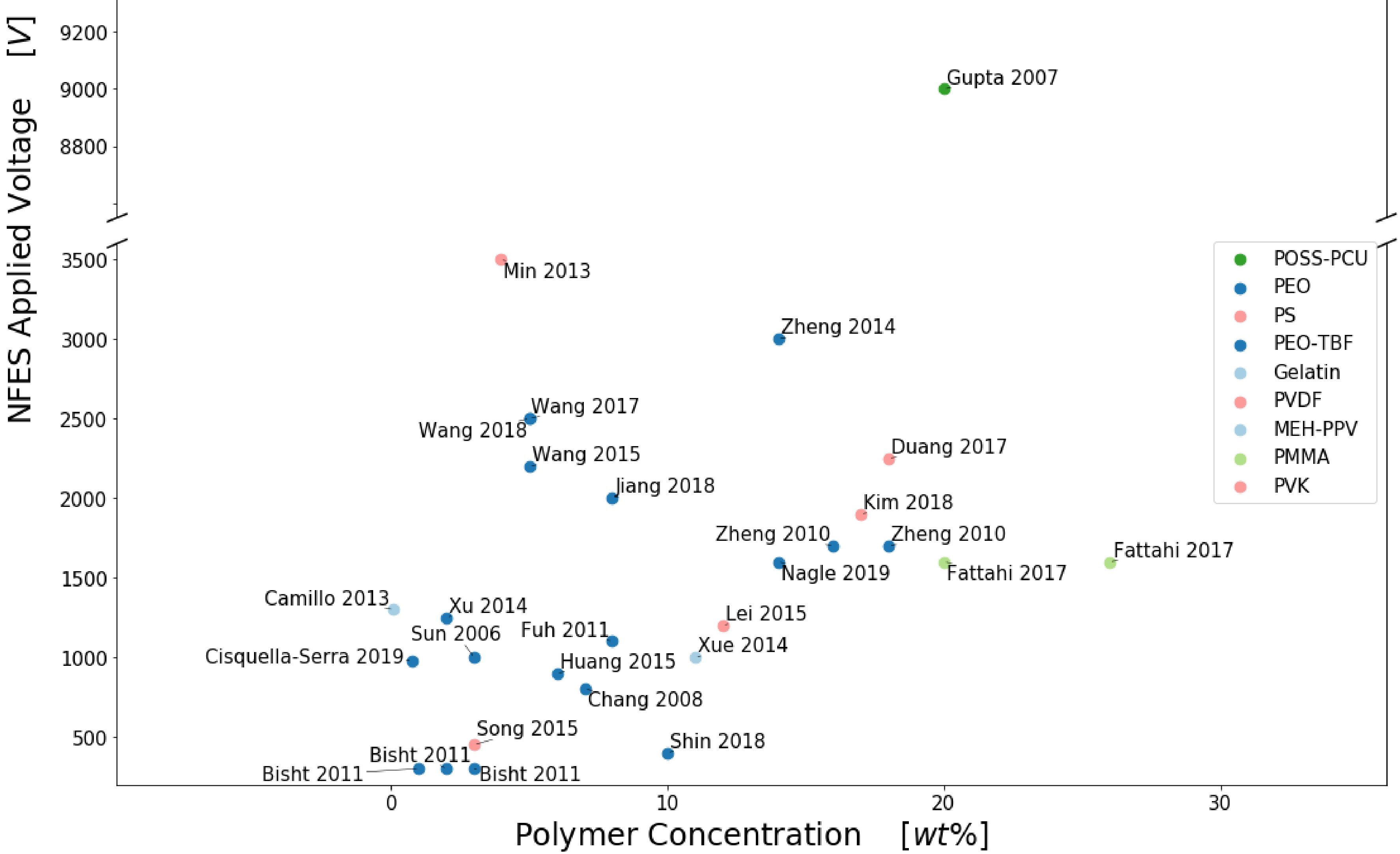
	Nozzle Diameter	NFES Applied Voltage	Reference	Polymer	Polymer Name
25	210.0	300.0	Bisht 2011	3	PEO
41	108.0	400.0	Shin 2018	3	PEO
20	2.0	450.0	Song 2015	9	PS
31	100.0	800.0	Chang 2008	3	PEO
42	400.0	1100.0	Fuh 2011	3	PEO
17	150.0	1250.0	Xu 2014	3	PEO
21	260.0	1300.0	Camillo 2013	1	MEH-PPV
14	250.0	1600.0	Nagle 2019	3	PEO
11	40.0	1700.0	Zheng 2010	3	PEO
1	100.0	1900.0	Kim 2018	10	PVDF
10	210.0	2000.0	Jiang 2018	3	PEO
7	260.0	2250.0	Duang 2017	10	PVDF
35	180.0	2500.0	Wang 2017	3	PEO
36	180.0	2500.0	Wang 2018	3	PEO
13	210.0	3000.0	Zheng 2014	3	PEO
18	100.0	3500.0	Min 2013	11	PVK
2	750.0	9000.0	Gupta 2007	8	POSS-PCU

In [18]:

```
# PLOT FIG
x_str = 'Polymer Concentration'; x_units = r'$[wt\%]$';
y_str = 'NFES Applied Voltage'; y_units = r'$[V]$';
breakYlim = 27;

#scatterPlot(x_str, x_units, y_str, y_units, df, 'original')
scatterPlot_breakAxis(x_str, x_units, y_str, y_units, df, df_x, breakYlim, 'center right');
```

```
C:\Users\oskat\Anaconda3\lib\site-packages\ipykernel_launcher.py:78: UserWarning: Attempting to set identical bottom == top == 9000.0 results in singular transformati
ons; automatically expanding.
```




```
>>> new_df
```

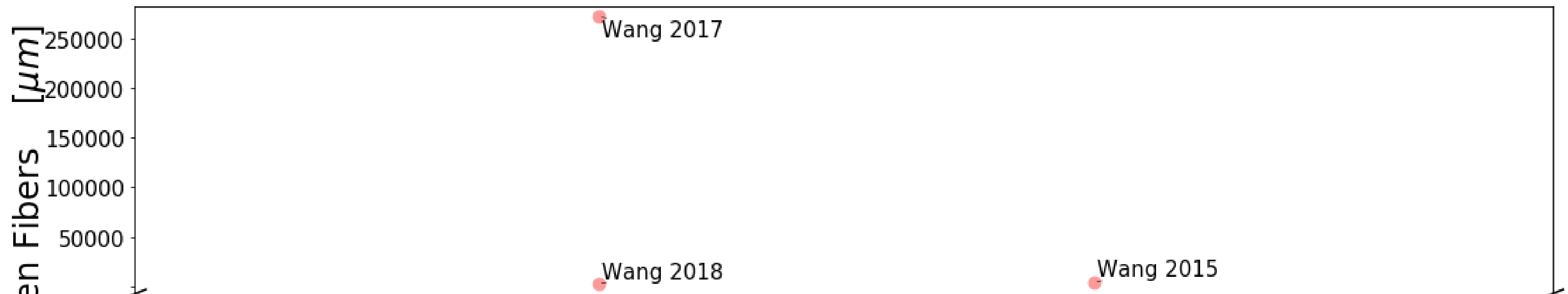
	Polymer Concentration	NFES Applied Voltage	Reference	Polymer	Polymer Name
27	3.00	300.0	Bisht 2011	3	PEO
26	2.00	300.0	Bisht 2011	3	PEO
25	1.00	300.0	Bisht 2011	3	PEO
41	10.00	400.0	Shin 2018	3	PEO
20	3.00	450.0	Song 2015	9	PS
31	7.00	800.0	Chang 2008	3	PEO
32	6.00	900.0	Huang 2015	3	PEO

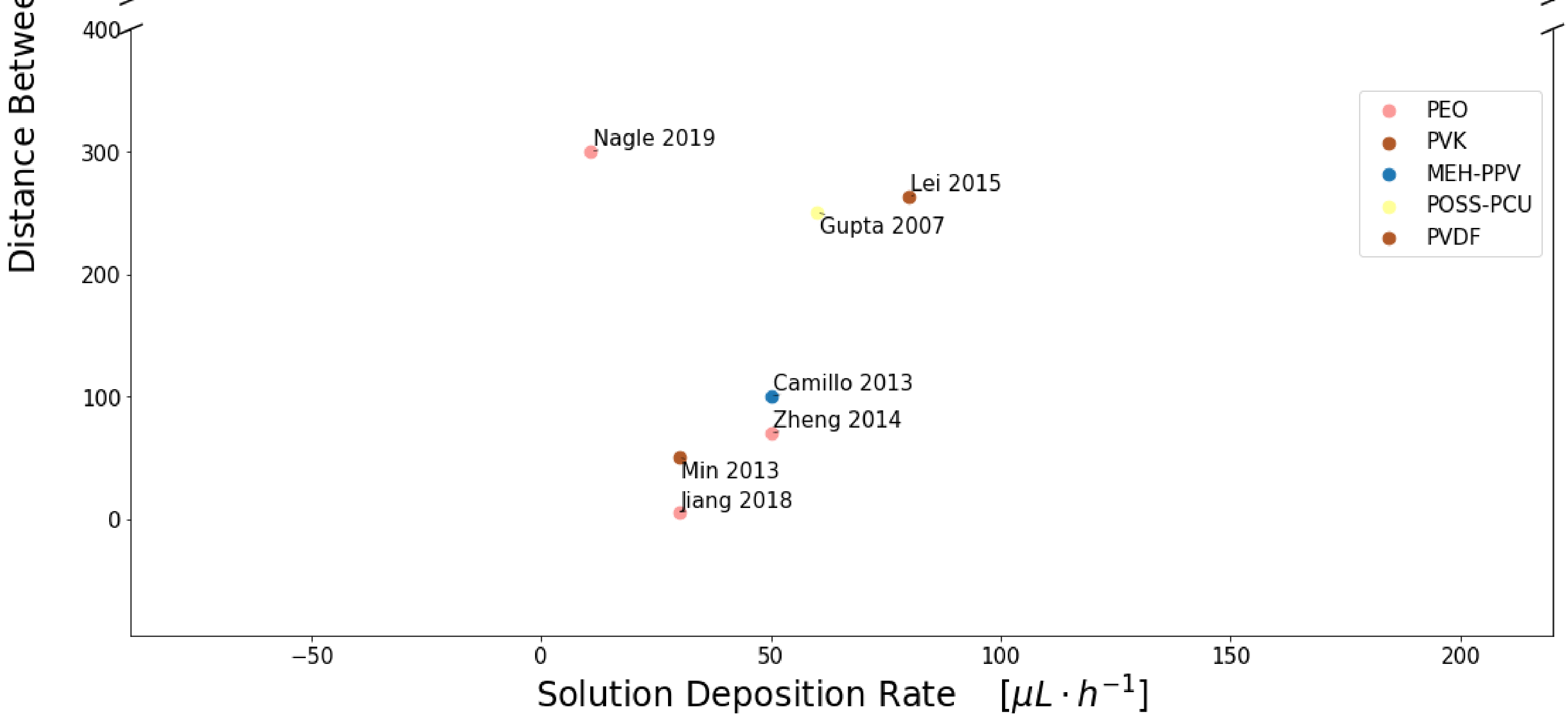
	Polymer Concentration	NFES Applied Voltage	Reference	Polymer	Polymer Name
8	0.75	980.0	Cisquella-Serra 2019	4	PEO-TBF
16	3.00	1000.0	Sun 2006	3	PEO
0	11.00	1000.0	Xue 2014	0	Gelatin
42	8.00	1100.0	Fuh 2011	3	PEO
43	12.00	1200.0	Lei 2015	10	PVDF
17	2.00	1250.0	Xu 2014	3	PEO
21	0.08	1300.0	Camillo 2013	1	MEH-PPV
74	26.00	1600.0	Fattahi 2017	6	PMMA
73	20.00	1600.0	Fattahi 2017	6	PMMA
14	14.00	1600.0	Nagle 2019	3	PEO
12	18.00	1700.0	Zheng 2010	3	PEO
11	16.00	1700.0	Zheng 2010	3	PEO
1	17.00	1900.0	Kim 2018	10	PVDF
10	8.00	2000.0	Jiang 2018	3	PEO
34	5.00	2200.0	Wang 2015	3	PEO
7	18.00	2250.0	Duang 2017	10	PVDF
35	5.00	2500.0	Wang 2017	3	PEO
36	5.00	2500.0	Wang 2018	3	PEO
13	14.00	3000.0	Zheng 2014	3	PEO
18	3.96	3500.0	Min 2013	11	PVK
2	20.00	9000.0	Gupta 2007	8	POSS-PCU

In [8]:

```
# PLOT FIG
x_str = 'Solution Deposition Rate'; x_units = r'$[\mu L \cdot h^{-1}]$';
y_str = 'Distance Between Fibers'; y_units = r'$[\mu m]$';
breakYlim = 7;

#scatterPlot(x_str, x_units, y_str, y_units, df, 'original')
scatterPlot_breakAxis(x_str, x_units, y_str, y_units, df, df_x, breakYlim, 'center right');
```






```
>>> new_df
```

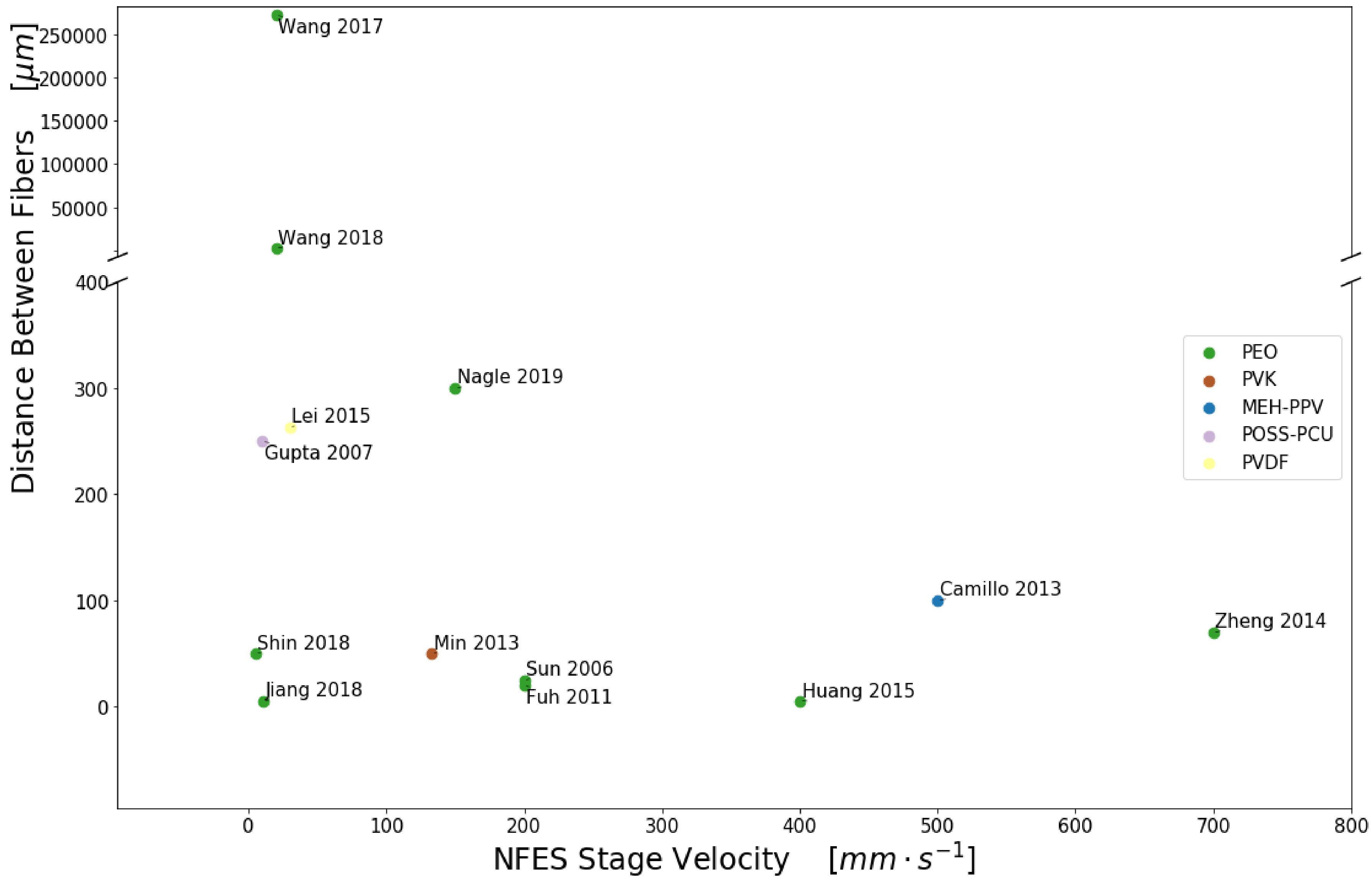
	Solution Deposition Rate	Distance Between Fibers	Reference	Polymer	Polymer Name
10	30.0	5.13	Jiang 2018	3	PEO
18	30.0	50.00	Min 2013	11	PVK
13	50.0	70.00	Zheng 2014	3	PEO
21	50.0	100.00	Camillo 2013	1	MEH-PPV
2	60.0	250.00	Gupta 2007	8	POSS-PCU
43	80.0	263.14	Lei 2015	10	PVDF
14	10.8	300.00	Nagle 2019	3	PEO
36	12.0	2511.30	Wang 2018	3	PEO
34	120.0	4000.00	Wang 2015	3	PEO
35	12.0	272000.00	Wang 2017	3	PEO

In [9]:

```
# PLOT FIG
x_str = 'NFES Stage Velocity'; x_units = r'$[mm \cdot s^{-1}]$';
```

```
x_str = 'NFES Stage Velocity'; x_units = r'$[mm \cdot s^{-1}]$';
y_str = 'Distance Between Fibers'; y_units = r'$[\mu m]$';
breakYlim = 11;

#scatterPlot(x_str, x_units, y_str, y_units, df, 'original')
scatterPlot_breakAxis(x_str, x_units, y_str, y_units, df, df_x, breakYlim, 'center right');
```




```
>>> new_df
```

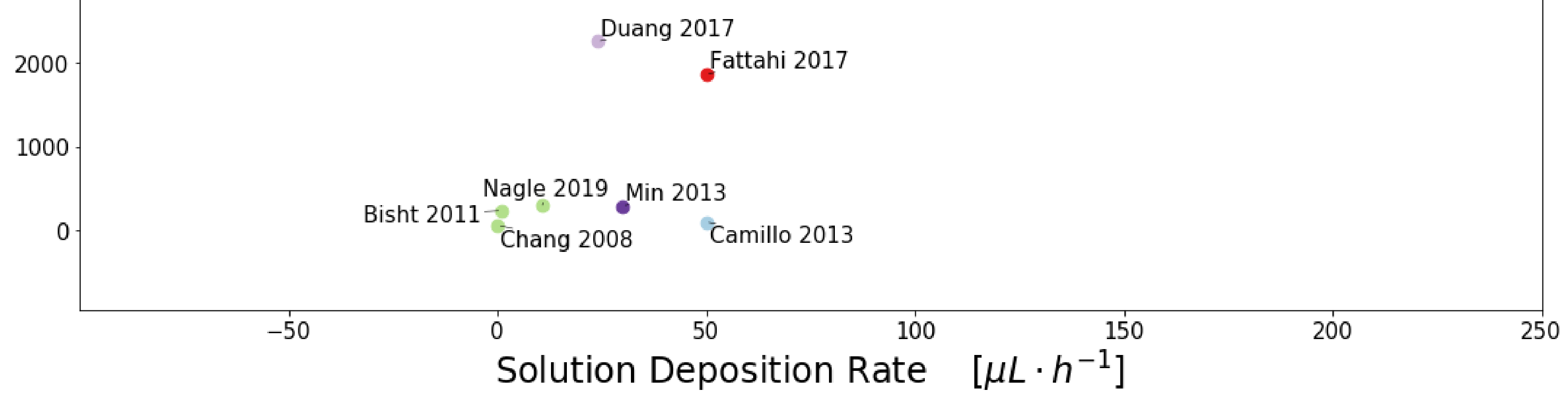
	NFES Stage Velocity	Distance Between Fibers	Reference	Polymer	Polymer Name
32	400.0	5.00	Huang 2015	3	PEO

	NFES Stage Velocity	Distance Between Fibers	Reference	Polymer	Polymer Name
10	10.5	15.13	Jiang 2018	3	PEO
42	200.0	20.00	Fuh 2011	3	PEO
16	200.0	25.00	Sun 2006	3	PEO
18	133.0	50.00	Min 2013	11	PVK
41	5.0	50.00	Shin 2018	3	PEO
13	700.0	70.00	Zheng 2014	3	PEO
21	500.0	100.00	Camillo 2013	1	MEH-PPV
2	10.0	250.00	Gupta 2007	8	POSS-PCU
43	30.0	263.14	Lei 2015	10	PVDF
14	150.0	300.00	Nagle 2019	3	PEO
36	20.0	2511.30	Wang 2018	3	PEO
35	20.0	272000.00	Wang 2017	3	PEO

In [16]:

```
# PLOT FIG
x_str = 'Solution Deposition Rate'; x_units = r'$[\mu L \cdot h^{-1}]$';
y_str = 'Fiber Diameter'; y_units = r'$[nm]$';
breakYlim = 11;

#scatterPlot(x_str, x_units, y_str, y_units, df, 'original')
scatterPlot_breakAxis(x_str, x_units, y_str, y_units, df, df_x, breakYlim, 'center right');
```



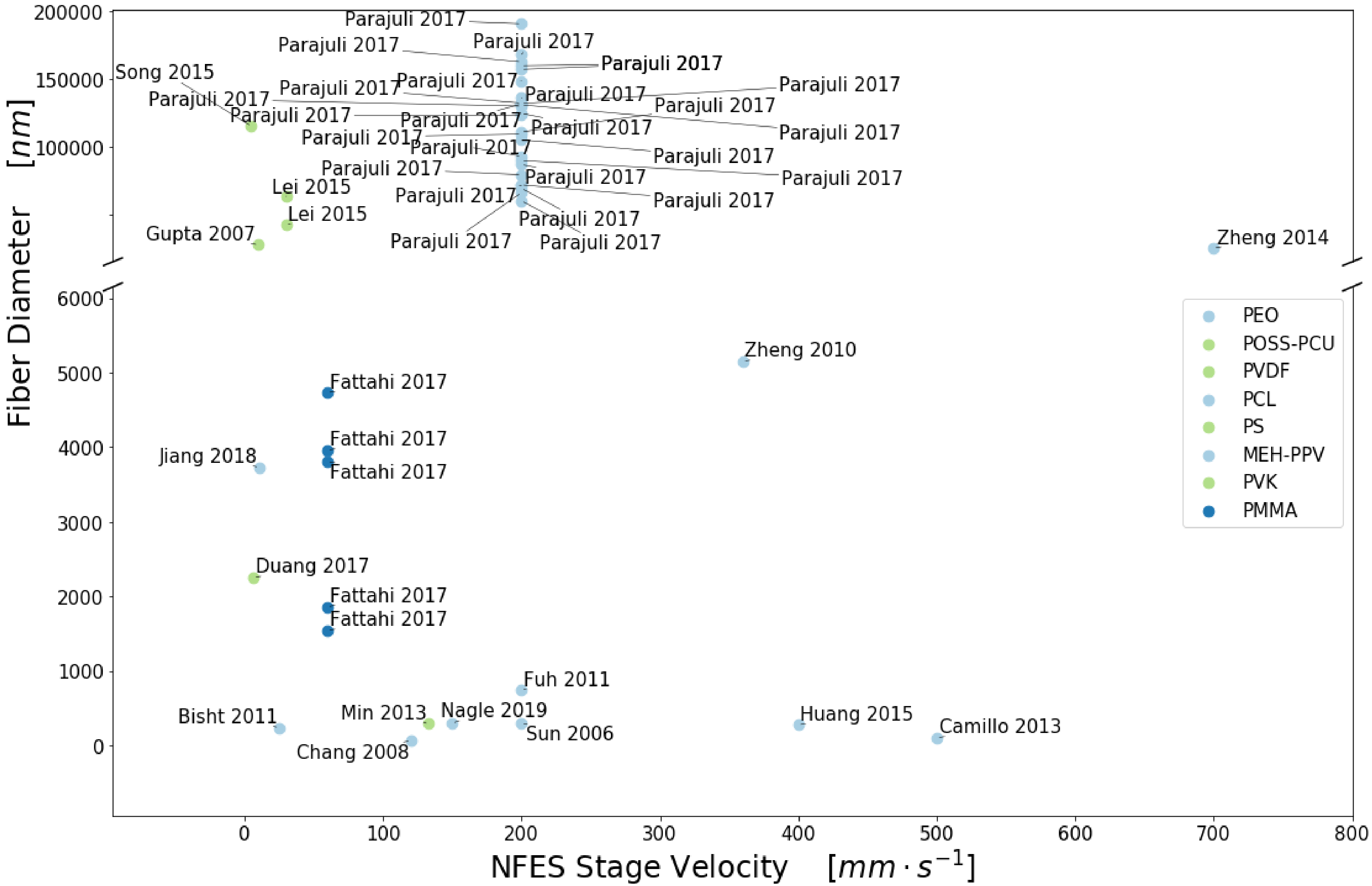

```
>>> new_df
```

	Solution Deposition Rate	Fiber Diameter	Reference	Polymer	Polymer Name
31	0.1	61.00	Chang 2008	3	PEO
21	50.0	100.00	Camillo 2013	1	MEH-PPV
25	1.0	237.00	Bisht 2011	3	PEO
18	30.0	289.26	Min 2013	11	PVK
14	10.8	300.00	Nagle 2019	3	PEO
73	50.0	1860.00	Fattahi 2017	6	PMMA
7	24.0	2250.00	Duang 2017	10	PVDF
10	30.0	3730.00	Jiang 2018	3	PEO
76	150.0	3960.00	Fattahi 2017	6	PMMA
74	50.0	4730.00	Fattahi 2017	6	PMMA
34	120.0	5470.00	Wang 2015	3	PEO
13	50.0	25000.00	Zheng 2014	3	PEO
2	60.0	27500.00	Gupta 2007	8	POSS-PCU
43	80.0	42258.81	Lei 2015	10	PVDF
44	80.0	63262.15	Lei 2015	10	PVDF

In [10]:

```
# PLOT FIG
x_str = 'NFES Stage Velocity'; x_units = r'$[mm \cdot s^{-1}]$';
y_str = 'Fiber Diameter'; y_units = r'$[nm]$';
breakYlim = 16;

#scatterPlot(x_str, x_units, y_str, y_units, df, 'original')
scatterPlot_breakAxis(x_str, x_units, y_str, y_units, df, df_x, breakYlim, 'center right');
```




```
>>> new_df
```

	NFES Stage Velocity	Fiber Diameter	Reference	Polymer	Polymer Name
31	120.000	61.00	Chang 2008	3	PEO
21	500.000	100.00	Camillo 2013	1	MEH-PPV
25	25.000	237.00	Bisht 2011	3	PEO
32	400.000	275.00	Huang 2015	3	PEO
18	133.000	289.26	Min 2013	11	PVK

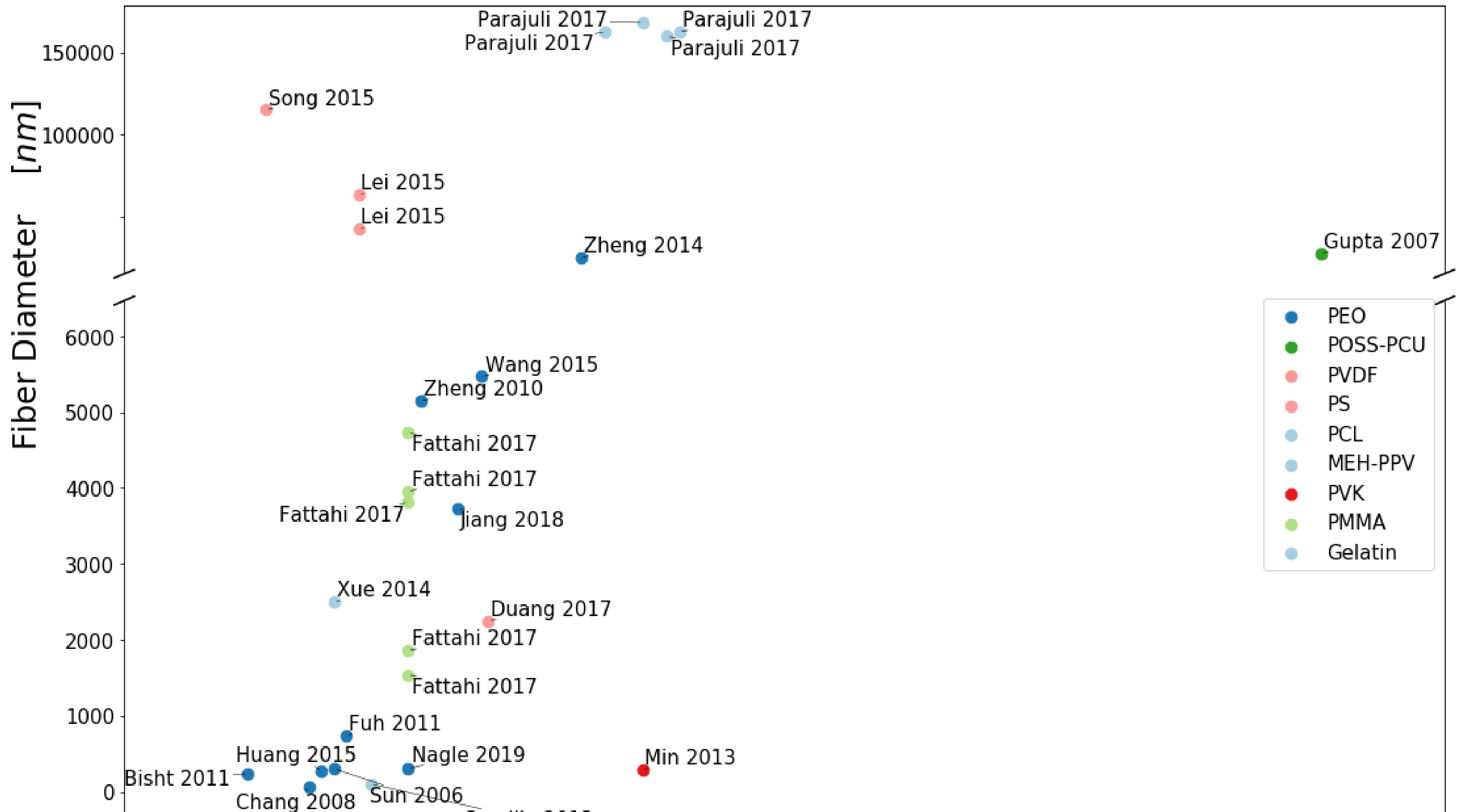
	NFES Stage Velocity	Fiber Diameter	Reference	Polymer	Polymer Name
14	150.000	300.00	Nagle 2019	3	PEO
16	200.000	300.00	Sun 2006	3	PEO
42	200.000	740.00	Fuh 2011	3	PEO
78	60.000	1540.00	Fattahi 2017	6	PMMA
73	60.000	1860.00	Fattahi 2017	6	PMMA
7	6.667	2250.00	Duang 2017	10	PVDF
10	10.500	3730.00	Jiang 2018	3	PEO
77	60.000	3810.00	Fattahi 2017	6	PMMA
76	60.000	3960.00	Fattahi 2017	6	PMMA
74	60.000	4730.00	Fattahi 2017	6	PMMA
11	360.000	5150.00	Zheng 2010	3	PEO
13	700.000	25000.00	Zheng 2014	3	PEO
2	10.000	27500.00	Gupta 2007	8	POSS-PCU
43	30.000	42258.81	Lei 2015	10	PVDF
72	200.000	59960.00	Parajuli 2017	2	PCL
44	30.000	63262.15	Lei 2015	10	PVDF
71	200.000	65890.00	Parajuli 2017	2	PCL
70	200.000	69250.00	Parajuli 2017	2	PCL
69	200.000	71220.00	Parajuli 2017	2	PCL
67	200.000	71680.00	Parajuli 2017	2	PCL
66	200.000	79270.00	Parajuli 2017	2	PCL
65	200.000	86760.00	Parajuli 2017	2	PCL
58	200.000	89680.00	Parajuli 2017	2	PCL
64	200.000	92820.00	Parajuli 2017	2	PCL
55	200.000	104920.00	Parajuli 2017	2	PCL
57	200.000	109300.00	Parajuli 2017	2	PCL
68	200.000	110800.00	Parajuli 2017	2	PCL
20	5.000	115000.00	Song 2015	9	PS
56	200.000	123080.00	Parajuli 2017	2	PCL
62	200.000	124510.00	Parajuli 2017	2	PCL
53	200.000	129850.00	Parajuli 2017	2	PCL
52	200.000	130780.00	Parajuli 2017	2	PCL
61	200.000	131370.00	Parajuli 2017	2	PCL
63	200.000	131840.00	Parajuli 2017	2	PCL
60	200.000	132160.00	Parajuli 2017	2	PCL
59	200.000	136040.00	Parajuli 2017	2	PCL
51	200.000	148530.00	Parajuli 2017	2	PCL

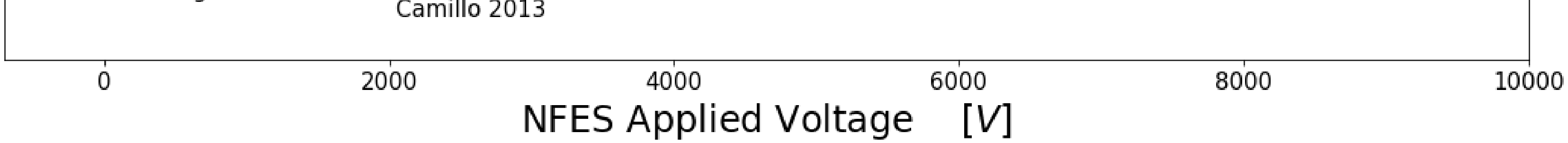
	NFES Stage Velocity	Fiber Diameter	Reference	Polymer	Polymer Name
51	200.000	148530.00	Parajuli 2017	2	PCL
50	200.000	156940.00	Parajuli 2017	2	PCL
47	200.000	159740.00	Parajuli 2017	2	PCL
45	200.000	162540.00	Parajuli 2017	2	PCL
46	200.000	168140.00	Parajuli 2017	2	PCL
54	200.000	190660.00	Parajuli 2017	2	PCL

In [11]:

```
# PLOT FIG
x_str = 'NFES Applied Voltage'; x_units = r'$[V]$';
y_str = 'Fiber Diameter'; y_units = r'$[nm]$';
breakYlim = 18;

#scatterPlot(x_str, x_units, y_str, y_units, df, 'original')
scatterPlot_breakAxis(x_str, x_units, y_str, y_units, df, df_x, breakYlim, 'center right');
```





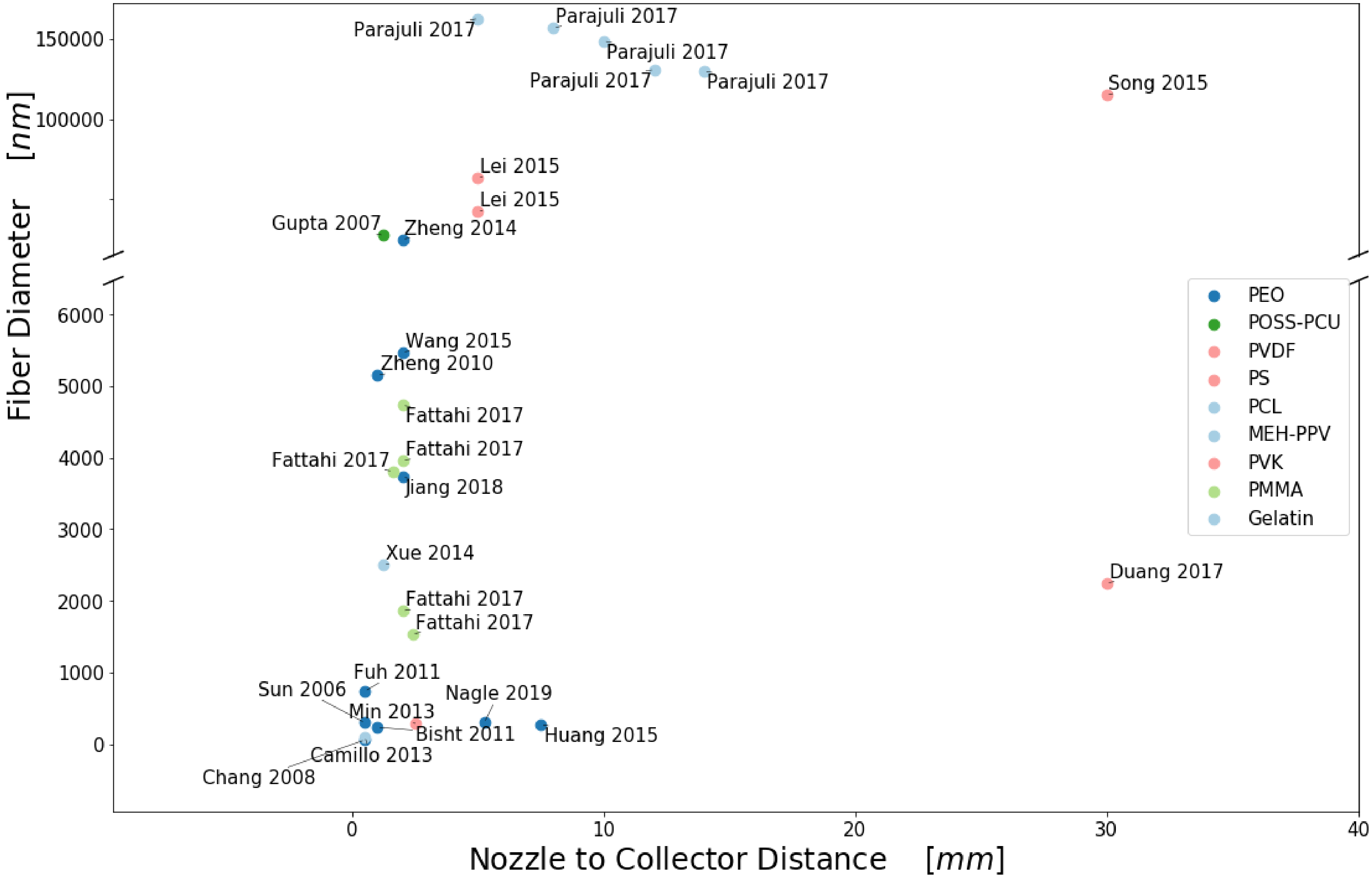

```
>>> new_df
```

	NFES Applied Voltage	Fiber Diameter	Reference	Polymer	Polymer Name
31	800.0	61.00	Chang 2008	3	PEO
21	1300.0	100.00	Camillo 2013	1	MEH-PPV
25	300.0	237.00	Bisht 2011	3	PEO
32	900.0	275.00	Huang 2015	3	PEO
18	3500.0	289.26	Min 2013	11	PVK
14	1600.0	300.00	Nagle 2019	3	PEO
16	1000.0	300.00	Sun 2006	3	PEO
42	1100.0	740.00	Fuh 2011	3	PEO
78	1600.0	1540.00	Fattahi 2017	6	PMMA
73	1600.0	1860.00	Fattahi 2017	6	PMMA
7	2250.0	2250.00	Duang 2017	10	PVDF
0	1000.0	2500.00	Xue 2014	0	Gelatin
10	2000.0	3730.00	Jiang 2018	3	PEO
77	1600.0	3810.00	Fattahi 2017	6	PMMA
76	1600.0	3960.00	Fattahi 2017	6	PMMA
74	1600.0	4730.00	Fattahi 2017	6	PMMA
11	1700.0	5150.00	Zheng 2010	3	PEO
34	2200.0	5470.00	Wang 2015	3	PEO
13	3000.0	25000.00	Zheng 2014	3	PEO
2	9000.0	27500.00	Gupta 2007	8	POSS-PCU
43	1200.0	42258.81	Lei 2015	10	PVDF
44	1200.0	63262.15	Lei 2015	10	PVDF
20	450.0	115000.00	Song 2015	9	PS
47	3700.0	159740.00	Parajuli 2017	2	PCL
45	3200.0	162540.00	Parajuli 2017	2	PCL
48	3800.0	162540.00	Parajuli 2017	2	PCL
46	3500.0	168140.00	Parajuli 2017	2	PCL

In [12]:

```
# PLOT FIG
x_str = 'Nozzle to Collector Distance'; x_units = r'$[mm]$';
y_str = 'Fiber Diameter'; y_units = r'$[nm]$';
breakYlim = 18;

#scatterPlot(x_str, x_units, y_str, y_units, df, 'original')
scatterPlot_breakAxis(x_str, x_units, y_str, y_units, df, df_x, breakYlim, 'center right');
```




```
>>> new_df
```

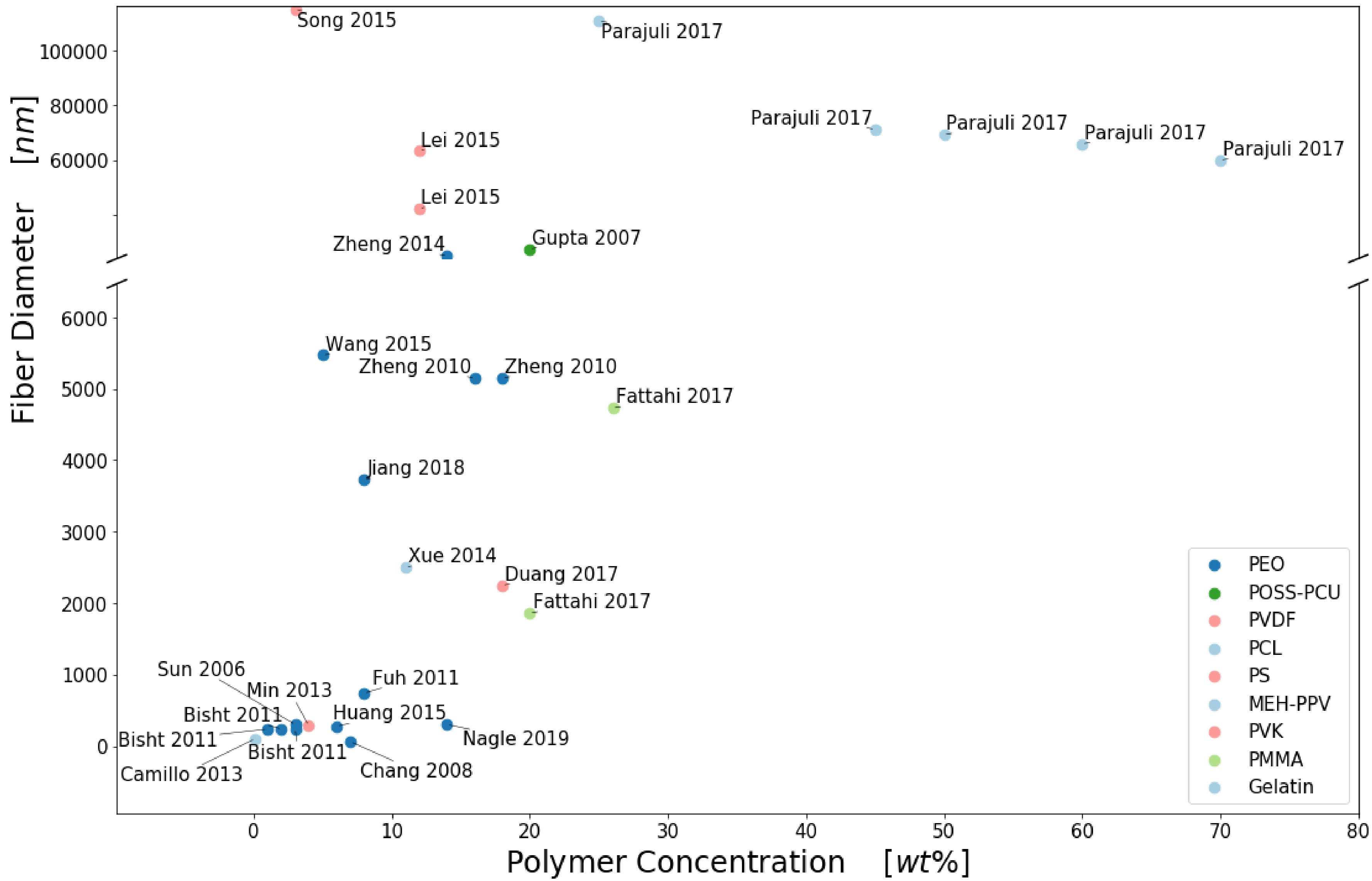
Nozzle to Collector Distance	Fiber Diameter	Reference	Polymer	Polymer Name

	Nozzle to Collector Distance	Fiber Diameter	Reference	Polymer	Polymer Name
31	0.50	61.00	Chang 2008	3	PEO
21	0.50	100.00	Camillo 2013	1	MEH-PPV
25	1.00	237.00	Bisht 2011	3	PEO
32	7.50	275.00	Huang 2015	3	PEO
18	2.50	289.26	Min 2013	11	PVK
14	5.25	300.00	Nagle 2019	3	PEO
16	0.50	300.00	Sun 2006	3	PEO
42	0.50	740.00	Fuh 2011	3	PEO
78	2.40	1540.00	Fattahi 2017	6	PMMA
73	2.00	1860.00	Fattahi 2017	6	PMMA
7	30.00	2250.00	Duang 2017	10	PVDF
0	1.25	2500.00	Xue 2014	0	Gelatin
10	2.00	3730.00	Jiang 2018	3	PEO
77	1.60	3810.00	Fattahi 2017	6	PMMA
76	2.00	3960.00	Fattahi 2017	6	PMMA
74	2.00	4730.00	Fattahi 2017	6	PMMA
11	1.00	5150.00	Zheng 2010	3	PEO
34	2.00	5470.00	Wang 2015	3	PEO
13	2.00	25000.00	Zheng 2014	3	PEO
2	1.25	27500.00	Gupta 2007	8	POSS-PCU
43	5.00	42258.81	Lei 2015	10	PVDF
44	5.00	63262.15	Lei 2015	10	PVDF
20	30.00	115000.00	Song 2015	9	PS
53	14.00	129850.00	Parajuli 2017	2	PCL
52	12.00	130780.00	Parajuli 2017	2	PCL
51	10.00	148530.00	Parajuli 2017	2	PCL
50	8.00	156940.00	Parajuli 2017	2	PCL
49	5.00	162540.00	Parajuli 2017	2	PCL

In [13]:

```
# PLOT FIG
x_str = 'Polymer Concentration'; x_units = r'$[wt\%]$';
y_str = 'Fiber Diameter'; y_units = r'$[nm]$';
breakYlim = 18;

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




```
>>> new_df
```

	Polymer Concentration	Fiber Diameter	Reference	Polymer	Polymer Name
31	7.00	61.00	Chang 2008	3	PEO
21	0.08	100.00	Camillo 2013	1	MEH-PPV
26	2.00	237.00	Bisht 2011	3	PEO
27	3.00	237.00	Bisht 2011	3	PEO
25	1.00	237.00	Bisht 2011	3	PEO
32	6.00	275.00	Huang 2015	3	PEO

	Polymer Concentration	Fiber Diameter	Reference	Polymer	Polymer Name
18	3.90	289.26	Min 2019	11	PVA
14	14.00	300.00	Nagle 2019	3	PEO
16	3.00	300.00	Sun 2006	3	PEO
42	8.00	740.00	Fuh 2011	3	PEO
73	20.00	1860.00	Fattahi 2017	6	PMMA
7	18.00	2250.00	Duang 2017	10	PVDF
0	11.00	2500.00	Xue 2014	0	Gelatin
10	8.00	3730.00	Jiang 2018	3	PEO
74	26.00	4730.00	Fattahi 2017	6	PMMA
12	18.00	5150.00	Zheng 2010	3	PEO
11	16.00	5150.00	Zheng 2010	3	PEO
34	5.00	5470.00	Wang 2015	3	PEO
13	14.00	25000.00	Zheng 2014	3	PEO
2	20.00	27500.00	Gupta 2007	8	POSS-PCU
43	12.00	42258.81	Lei 2015	10	PVDF
72	70.00	59960.00	Parajuli 2017	2	PCL
44	12.00	63262.15	Lei 2015	10	PVDF
71	60.00	65890.00	Parajuli 2017	2	PCL
70	50.00	69250.00	Parajuli 2017	2	PCL
69	45.00	71220.00	Parajuli 2017	2	PCL
68	25.00	110800.00	Parajuli 2017	2	PCL
20	3.00	115000.00	Song 2015	9	PS

In [11]:

```
import pdfkit
path_wkthmltopdf = r'C:\Program Files\wkhtmltopdf\bin\wkhtmltopdf.exe'
config = pdfkit.configuration(wkhtmltopdf=path_wkthmltopdf)

pdfkit.from_file('./NFES_ReviewPaper.html','NFES_ReviewPaper.pdf', configuration=config)
```

```
Loading pages (1/6)
Warning: Failed to load file:///C:/Users/oskat/OneDrive - Instituto Tecnologico y de Estudios Superiores de Monterrey/Documents/MNT_ITESM_Thesis/NFES_Review/_jupyterP
lots/custom.css (ignore)
Counting pages (2/6)
Resolving links (4/6)
Loading headers and footers (5/6)
Printing pages (6/6)
Done
```

Out[11]:

```
True
```

In [12]:

```
'''
# 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 Diameter'])

# 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[:]['NFES 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[12]:

```
'\n# Multivariate Linear Regression\ndf_x = df.sort_values(by=[\'Fiber Diameter\'])\ndf_x = df_x.dropna(subset=[\'Polymer Concentration\',\'Nozzle Diameter\',\'NFES A
pplied Voltage\',\'Fiber Diameter\'])\n\n# X is the independent variable (bivariate in this case)\nX = np.array([df_x.iloc[:][\'Polymer Concentration\'], df_x.iloc[:]
[\'Nozzle Diameter\'], df_x.iloc[:][\'NFES Applied Voltage\']])\n\n# Y is the dependent data\nY = df_x.iloc[:][\'Fiber Diameter\']\n\n# predict is an independent vari
able for which we\'d like to predict the value\npredict= [[20.0, 750.0, 9000.0]]\n\n# generate a model of polynomial features\npoly = PolynomialFeatures(degree=2)\n\n
# transform the x data for proper fitting (for single variable 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 = LinearRegression()\n\n# perform the actual regression\nclf = clf.fit(np.transp
ose(X), Y)\n\nprint(\'\n\')\nprint(\'>>> INTERCEPT & COEFFICIENTS\')\nprint(clf.intercept_)\nprint(clf.coef_)\nprint(\'\n\')\nprint(\'>>> PREDICTION\')\nprint("Predic
tion = " + str(clf.predict(predict)))\nprint(\'\n\')\n'
```

In [ ]: