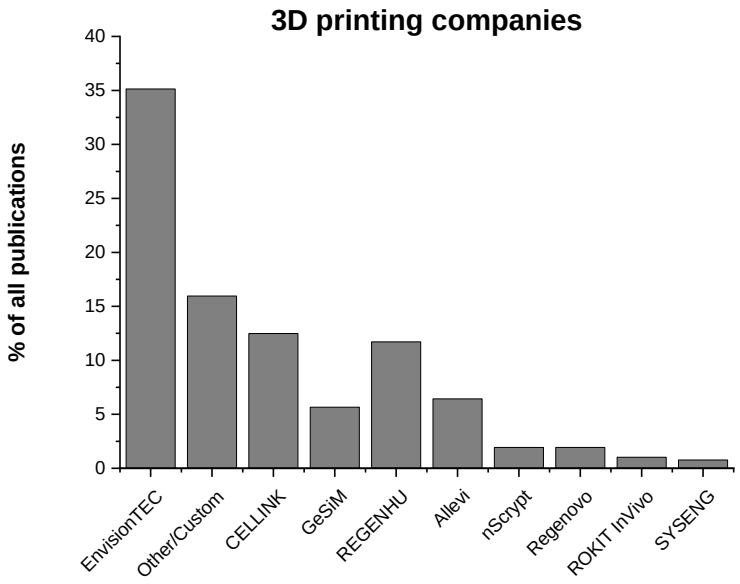
Supplementary information

An Open-Source Bioink Database for Microextrusion 3D Printing



Supplemental Figure 1. List of top 10 3D printing companies whose printers of various models were used in the database publications.

Supplementary Table 1. Pearson correlation coefficients for printing parameters involving 100 wt% PCL at different molecular weights.

	Mol. Wt. (kDa)	**Pressure (bar)**	**Temp (°C)**	**Needle Size (µm)**	**Speed (mm/s)**
Mol. Wt. (kDa)	1	0.034	0.07	0.044	-0.067
Pressure (bar)		1	-0.015	-0.074	0.1
Temp (°C)			1	0.13	-0.186
Needle Size (µm)				1	-0.066
Speed (mm/s)					1

Supplementary Table 2. Pearson correlation coefficients for printing parameters involving all PCL wt%

	Wt %	**Press. (bar)**	**Temp (°C)**	**Speed (mm/s)**	**Needle Size (µm)**
Wt %	1	0.3***	0.28***	-0.155*	-0.098
Press. (bar)		1	0.44***	0.016	-0.668***
Temp (°C)			1	-0.183*	-0.08
Speed (mm/s)				1	-0.109
Needle Size (µm)					1

* $p<0.05$; ** $p<0.01$; *** $p<0.001$

Supplementary Table 3. Pearson correlation coefficients for printing parameters involving GelMA as the only bioink components.

	Wt %	Press. (bar)	Temp (°C)	Speed (mm/s)	Needle Size (µm)
Wt %	1	0.63***	0.028	0.346***	0.002
Press. (bar)		1	-0.126	0.145	-0.015
Temp (°C)			1	-0.265**	-0.53***
Speed (mm/s)				1	0.133
Needle Size (µm)					1

* p<0.05; ** p<0.01; *** p<0.001

Supplementary Table 4. Pearson correlation coefficients for printing parameters involving all GelMA-based bioinks.

	Wt %	Press. (bar)	Temp (°C)	Speed (mm/s)	Needle Size (µm)
Wt %	1	0.262***	0.133	0.015	-0.091
Press. (bar)		1	-0.027	-0.00327	-0.048
Temp (°C)			1	-0.445***	-0.221**
Speed (mm/s)				1	0.191*
Needle Size (µm)					1

* p<0.05; ** p<0.01; *** p<0.001

Supplementary Table 5. Pearson correlation coefficients for printing parameters involving all cell-based bioinks.

	Nozzle diameter (µm)	Press. (bar)	Needle type
Nozzle diameter (µm)	1	0.128*	-0.02
Press. (bar)		1	0.184***
Needle type			1

Needle type: 1 = cone; 2 = cylinder

* $p<0.05$; *** $p<0.001$

Supplementary Table 6. List of cells used in 3D printing for publications on the database.

NIH 3T3 Fibroblasts
Human Mesenchymal Stem Cells
Human Umbilical Vein Endothelial Cells
Primary Rat Schwann Cells
C2C12 Myoblasts
Human Dermal Fibroblasts
EGFR T790M Non-Small Cell Lung Carcinoma PDX
AA0022 Lung Cancer Associated Fibroblasts
Primary Bovine Chondrocytes
RSC96 Neuronal Schwann Cells
L929 Fibroblasts
Bone Marrow Derived Mesenchymal Stem Cells
Saos-2 Human Bone Osteosarcoma cells
Placenta BeWo Cells
Human Chondrocytes
Primary Embryonic Chick Chondrocytes
HEPG2 Liver Cells
Primary Chick Chondrocytes
A549 Human Epithelial Cells
Primary Porcine Bone Marrow Stromal Cells
Primary Porcine Cartilage Chondrocytes
BEL-7402 Adenocarcinoma Cells

Primary Rat Bone Marrow Mesenchymal Stem Cells
PCS-201-010 Primary Dermal Fibroblasts
hiPSC-derived Ventral Spinal Neuronal Progenitor Cells
miPSC-derived Oligodendrocyte Progenitor Cells
Human Induced Pluripotent Stem Cells
Primary Rabbit Adipose-Derived Mesenchymal Stem Cells
Human Glial Cells
Primary Human Nasal Septal Cartilage Chondrocyte
Primary Human Periodontal Ligament Cells
ATDC5 Chondrocytes
Primary Sheep Chondrocytes
HEK293 Kidney Cells
Human Fibroblasts
Cell Line SH-SY5Y
Primary Human Adipose-derived Mesenchymal Stem Cells
S16Y Schwann Cells
Human Coronary Artery Endothelial Cells
Human Adipose Stem Cells
Human Bone Marrow-derived Mesenchymal Stem Cells
NIH 3T3 Fibroblasts GFP
Immortalized Human Mesenchymal Stem Cells
Human Esophageal Epithelial Cells
Human Esophageal Smooth Muscle Cells
Cell Line HepaRG
Primary Human Umbilical Vein Endothelial Cells
Primary Human Amniotic Fluid-derived Stem Cells
Primary Rabbit Ear Chondrocytes
Primary Neonatal Rat Ventricular Cardiomyocytes
Primary Human Muscle Progenitor Cells
Human Epidermal Keratinocytes
Human Epidermal Fibroblasts
Primary Adult Human Dermal Fibroblasts
Primary Human Nasoseptal Cartilage
Primary Rat Bone Marrow-Derived Mesenchymal Stem Cells
Primary Rat Bone Mesenchymal Stem Cells
H9C2 Myoblasts
SK-N-BE(2) Neuroblasts
Primary Mouse Epidermal Stem Cells
MDA-MB-231 Epithelial Cells
Primary Rabbit Fibrochondrocytes
Primary Rat Cardiac Myocytes

Human Dermal Fibroblasts/ Primary Human Epidermal Keratinocytes Mix
MG63 Osteoblast-like Cells
Human Adipose-derived Stem Cells
Plant cells - Basil
Primary Human Nasal Inferior Turbinate Tissue-derived
Primary Rabbit Bladder Urothelial Cells
Primary Rabbit Smooth Muscle Cells
Balb 3T3 Mouse Fibroblasts
Cell Line MC3T3-E1 Osteoblasts
Murine Bone Marrow Stromal Cells
PC-12 Cells
C57BL/6 Mouse Embryonic Stem Cells
Neural Stem Cells iPSC Differentiated
Bone Marrow-derived Mesenchymal Stem Cells
Primary Porcine Stromal Vascular Fraction Cells
Primary Human Adipose Derived Stem Cells
Human Neural Stem Cells
Human Osteogenic Sarcoma Cells
21 PT Breast Cancer Cells
Adipose-Derived Mesenchymal Stem Cells
Primary Human Dermal Fibroblasts
Hela Pancreatic Cancer Cells
Primary Juvenile Mice Interstitial Cells
Primary Human Hepatocytes
Primary Myoblasts
Human Articular Chondrocytes
Primary Bone Marrow Stem Cells
Human amniotic epithelial cells
Wharton’s jelly derived Mesenchymal Stem Cells
Primary Human Articular Chondrocytes
Primary Human Chondrocytes
Human Dermal Fibroblasts with FoxD3 Plasmid
Human-adipose tissue derived Mesenchymal Stem Cells
GSC23 Glioblastoma Cells
Equine-derived Mesenchymal Stromal Cells
Human Adipose Derived Stem Cells
Skeletal Stem Cells
Human mesenchymal stem cell
Human Bone Marrow Mesenchymal Stem Cells
Primary Porcine Chondrocytes
Wharton's Jelly Derived Mesenchymal Stem Cells

Human Amniotic Epithelial Cells
Bone Marrow Stromal Stem Cells
Bovine Colon Organoids
GSC23 Human Glioma Stem Cells
Human Corneal Epithelial Cells
Human Corneal Keratocytes

Supplementary Python codes

Trends In Bioink.py

```
# This script returns the Top Biomaterials Used, Top Biomaterial Combinations Used, and Top Printer Companies Used

import os, csv, pprint, xlsxwriter

#User Input
dataFile = open("MasterDatabase.csv", encoding="utf8")
dataReader = csv.reader(dataFile)
dataList = list(dataReader)
workbook=xlsxwriter.Workbook("PercentOfBioinksNew.xlsx")
bioInks = workbook.add_worksheet('PercentofBionks')
bioPrinters = workbook.add_worksheet('PercentBioprinters')
topBioInks = workbook.add_worksheet('Top Bioinks')


#Determines the Top Biomaterials Used
def topBioinks():
    bioInkDict = {}
    counter =1
    #For Loop Runs through the Entire Spreadsheet
    for i in range(3,len(dataList)):
        temp = []
        #For Loop Runs through each Biomaterial
        for materialInt in range(3, 12, 3):
            #Converts the Biomaterial Combinations into a Temp Variable.
            #This Temp Variable is Sorted Alphabetically and then Converted into Tuple which is Added to Dictionary
            if dataList[i][materialInt] != "":
                temp.append(dataList[i][materialInt].strip())
        materialTuple =tuple(sorted(temp))
        if materialTuple in bioInkDict:
            bioInkDict[materialTuple] = bioInkDict[materialTuple]+1
        else:
            bioInkDict[materialTuple]=1
    #Outputs and Writes the Dictionary into a Spreadsheet
    topBioInks.write(0,0,"BIOINK NAME")
    topBioInks.write(0,1,"AMOUNT OF TIMES")
    for key, value in bioInkDict.items():
        temp = ""
        for i in list(key):
            temp += i + ", "
        temp = temp[0:len(temp)-2]
        topBioInks.write(counter,0,temp)
        topBioInks.write(counter, 1, value)
        counter +=1
```

```
def percentOfBioinks():
    materialDict = {}
    #For Loop Iterates through Each Material Columnn
    for materialInt in range(3, 12, 3):
        counter=1
        #For Loop Iterates through the Length of the Spreadsheet
        for i in range(3, len(dataList)):
            material = dataList[i][materialInt].strip()
            #Checks for Blanks and Other Non-Materials
            if(material!="Bioink " and material!= "Component 1" and material!= ""):
                if material in materialDict:
                    materialDict[material] = materialDict[material]+1
                else:
                    materialDict[material] = 1
    #Writing to a Spreadsheet
        bioInks.write(0,0, "MATERIAL NAME")
        bioInks.write(0,1, "AMOUNT OF TIMES")
        counter = 1
        for key, value, in materialDict.items():
            bioInks.write(counter,0, key)
            bioInks.write(counter, 1, value)
            counter +=1

def tableOfPrinters():
    printerDict = {}
    #For Loop Iterates down the Spreadsheet
    for i in range(3, len(dataList)):
        printer = dataList[i][1].strip()
        #Checks for Blanks and Other Non-Printers
        if printer != "Printer " and printer != "":
            if printer in printerDict:
                printerDict[printer] = printerDict[printer]+1
            else:
                printerDict[printer] = 1
    #Writes to a Spreadsheet
    bioPrinters.write(0,0, "PRINTER COMPANY")
    bioPrinters.write(0,1, "AMOUNT OF TIMES")
    counter = 1
    for key, value, in printerDict.items():
        bioPrinters.write(counter,0, key)
        bioPrinters.write(counter, 1, value)
        counter +=1

topBioinks()
tableOfPrinters()
percentOfBioinks()
workbook.close()
```

MaterialParser.py

```
# This script does two things. First, the script reads and translates a CSV file into a 2D matrix then
dictionary of dictionaries(described in the comments below). After doing this, we then write every
possible combination of printing parameters into a XLSX file.

import os, csv, pprint, xlsxwriter
#User Input
material = input("Enter the material")
materialLower = material.lower()
soloOrNot = input("Solo or Combination:").lower()
variableX = input("Enter Variable X").lower()
variableY = input("Enter Y Variable to be Graphed between Temperature, Pressure, Speed, Needle Size,
and Product of Temperature/Pressure (POTP)").lower()
variableZ = input("Enter Z Variable to be Graphed between Temperature, Pressure, Speed, Needle Size,
and Product of Temperature/Pressure (POTP)").lower()
x = ""
y = ""
z = ""
xIndex = 0
yIndex = 0
zIndex = 0


#Determines the Variables Used for Contour Plot
def XYZVariableConverter(variableXYorZ):
    printingParameters = {
        "temperature": 3,
        "pressure": 2,
        "concentration": 1,
        "speed":4,
        "needle size": 5,
        "molecular weight": 6,
        "potp": 7
    }
    return printingParameters[variableXYorZ]

xIndex = XYZVariableConverter(variableX)
yIndex = XYZVariableConverter(variableY)
zIndex = XYZVariableConverter(variableZ)


#Reads Spreadsheet/Puts Spreadsheet in 2D Matrix
dataFile = open("MasterDatabase.csv", encoding="utf8")
dataReader = csv.reader(dataFile)
dataList = list(dataReader)
#Converts Gauge to Micrometer
def needleSizeConverter(needleSize):
    needleSizes = {
        "10": 2692,
        "11": 2388,
        "12": 2159,
        "13": 1803,
```

```
        "14": 1600,
        "15": 1372,
        "16": 1194,
        "17": 1067,
        "18": 838,
        "19": 686,
        "20": 603,
        "21": 514,
        "22": 413,
        "23": 337,
        "24": 311,
        "25": 260,
        "26": 260,
        "27": 210,
        "28": 184,
        "29": 184,
        "30": 159,
        "31": 133,
        "32": 108,
        "33": 108,
        "34": 51,
    }
    return needleSizes[str(needleSize)]

#Function Returns the Type of Parameter.
#For example, Molecular Weight would return {Molecular Weight High, Molecular Weight Low}.
#While POTP would return {HighHigh, HighLow, LowHigh, and LowLow}.
def amountOfParameters(variableXYorZ):
    parameters = {
        "concentration": ["Concentration"],
        "molecular weight": ["Molecular Weight High", "Molecular Weight Low"],
        "pressure": ["Pressure High", "Pressure Low"],
        "temperature": ["Temperature High", "Temperature Low"],
        "speed": ["Speed High", "Speed Low"],
        "potp": ["HighHigh", "HighLow", "LowHigh", "LowLow"],
        "needle size": ["Needle Size"]
    }
    return parameters[variableXYorZ]

#Creates all Possible Permutations of Temperature*Pressure(POTP) and Adds to a Dictionary
def POTP(v):
    POTPDict = {}
    #If High and High Exist
    if(v["Temperature High"] != "" and v["Pressure High"] != ""):
        POTPDict["HighHigh"] = float(v["Temperature High"])*float(v["Pressure High"])
    #If High and Low Exist
    if(v["Temperature High"] != "" and v["Pressure Low"] != ""):
        POTPDict["HighLow"] = float(v["Temperature High"])*float(v["Pressure Low"])
    #If Low and High Exist
    if(v["Temperature Low"] != "" and v["Pressure High"] != ""):
        POTPDict["LowHigh"] = float(v["Temperature Low"])*float(v["Pressure High"])
```

```
    #If Low and Low Exist
    if(v["Temperature Low"] != "" and v["Pressure Low"] != ""):
        POTPDict["LowLow"] = float(v["Temperature Low"])*float(v["Pressure Low"])
    return POTPDict

 #Function Creates a Data Structure that is a Dictionary of Dictionaries with an example structure as
follows:
#   {6: {'Concentration': 5.0, 'Needle Size': 400.0, 'Pressure High': 0.5, 'Pressure Low': '',
#       'Row Number': 5, 'Speed High': 10.0, 'Speed Low': '', 'Temperature High': 23.0, 'Temperature Low':
''},
#    7: {'Concentration': 10.0, 'Needle Size': 400.0, 'Pressure High': 0.7, 'Pressure Low': '',
#       'Row Number': 6, 'Speed High': 10.0, 'Speed Low': '', 'Temperature High': 28.0, 'Temperature Low':
''}
#The outer dictionary key holds the column number for that material and the inner dictionary holds the
printing parameter values (we grab all parameters)
def matParser(dataList, soloOrNot):
    matDict = {}
    counter = 0
    #If Bioink is Solo
    if(soloOrNot.lower() == "solo"):
        #Iterates down the Spreadsheet
        for i in range (0, len(dataList) ,1):
            if(dataList[i][3].lower().strip() == materialLower and dataList[i][6]=="" and dataList[i][9]==""
and dataList[i][12] == ""):
                temp={}
                #Grabs Molecular Weight Data
                if(dataList[i][41] == "" or dataList[i][41]=="N/A"):
                    temp["Molecular Weight High"] = ""
                else:
                    temp["Molecular Weight High"] = float(dataList[i][41])
                if(dataList[i][42] == "" or dataList[i][42]== "N/A"):
                    temp["Molecular Weight Low"] = ""
                else:
                    temp["Molecular Weight Low"] = float(dataList[i][42])
                #Grabs Pressure Data
                if(dataList[i][20] == "" or dataList[i][20]=="N/A"):
                    temp["Pressure High"] = ""
                else:
                    if(dataList[i][21].lower()=="kpa"):
                        temp["Pressure High"] = float(dataList[i][20])/100
                    elif(dataList[i][21].lower()=="psi"):
                        temp["Pressure High"] = float(dataList[i][20])/14.504
                    else:
                        temp["Pressure High"] = float(dataList[i][20])
                if(dataList[i][22] == "" or dataList[i][22]=="N/A"):
                    temp["Pressure Low"] = ""
                else:
                    if(dataList[i][23].lower()=="kpa"):
                        temp["Pressure Low"] = float(dataList[i][22])/100
                    elif(dataList[i][23].lower()=="psi"):
                        temp["Pressure Low"] = float(dataList[i][22])/14.504
```

```
            else:
                temp["Pressure Low"] = float(dataList[i][22])
            #Grabs Temperature Data
            if(dataList[i][24] == "" or dataList[i][24]=="N/A"):
                temp["Temperature High"] = ""
            else:
                temp["Temperature High"] = float(dataList[i][24])
            if(dataList[i][25] == "" or dataList[i][25]== "N/A"):
                temp["Temperature Low"] = ""
            else:
                temp["Temperature Low"] = float(dataList[i][25])
            #Grab Needle Size Data
            if(dataList[i][18] == "" or dataList[i][18] =="N/A"):
                temp["Needle Size"] = ""
            else:
                if dataList[i][19].lower() == "gauge":
                    temp["Needle Size"] = float(needleSizeConverter(dataList[i][18]))
                else:
                    temp["Needle Size"] = float(dataList[i][18])
            #Needle Type
            if(dataList[i][17] == "" or dataList[i][17] == "N/A"):
                temp["Needle Type"] = ""
            else:
                temp["Needle Type"] = dataList[i][17]
            #Grabs Speed Data
            if(dataList[i][28] == "" or dataList[i][28]== "N/A"):
                temp["Speed High"] = ""
            else:
                temp["Speed High"] = float(dataList[i][28])
            if(dataList[i][29] == "" or dataList[i][29]== "N/A"):
                temp["Speed Low"] = ""
            else:
                temp["Speed Low"] = float(dataList[i][29])
            #Grabs Concentration Data
            if(dataList[i][4] == "" or dataList[i][4]=="N/A"):
                temp["Concentration"] = ""
            else:
                if(dataList[i][5].lower() != "mg/ml"):
                    temp["Concentration"] = float(dataList[i][4])
                else:
                    temp["Concentration"] = float(dataList[i][5])/10
            #Which Row the Bioink is Located on the Spreadsheet
            temp["Row Number"] = i+1
            temp.update(POTP(temp))
            matDict[counter] = temp
            counter+=1
    #If Bioink is Combination
    else:
        #For Loop Iterates through the Biomaterial Columns
        for materialInt in range(3, 13,3):
            #For Loop Iterates down the Spreadsheet
```

```
for i in range (0, len(dataList) ,1):
    if (dataList[i][materialInt].lower().strip() == materialLower):
        temp = {}
        #Grabs Molecular Weight Data
        if(dataList[i][41] == "" or dataList[i][41]=="N/A"):
            temp["Molecular Weight High"] = ""
        else:
            temp["Molecular Weight High"] = float(dataList[i][41])
        if(dataList[i][42] == "" or dataList[i][42]== "N/A"):
            temp["Molecular Weight Low"] = ""
        else:
            temp["Molecular Weight Low"] = float(dataList[i][42])
        #Grabs Pressure Data
        if(dataList[i][20] == "" or dataList[i][20]=="N/A"):
            temp["Pressure High"] = ""
        else:
            if(dataList[i][21].lower()=="kpa"):
                temp["Pressure High"] = float(dataList[i][20])/100
            elif(dataList[i][21].lower()=="psi"):
                temp["Pressure High"] = float(dataList[i][20])/14.504
            else:
                temp["Pressure High"] = float(dataList[i][20])
        if(dataList[i][22] == "" or dataList[i][22]=="N/A"):
            temp["Pressure Low"] = ""
        else:
            if(dataList[i][23].lower()=="kpa"):
                temp["Pressure Low"] = float(dataList[i][22])/100
            elif(dataList[i][23].lower()=="psi"):
                temp["Pressure Low"] = float(dataList[i][22])/14.504
            else:
                temp["Pressure Low"] = float(dataList[i][22])
        #Grabs Temperature Data
        if(dataList[i][24] == "" or dataList[i][24]=="N/A"):
            temp["Temperature High"] = ""
        else:
            temp["Temperature High"] = float(dataList[i][24])
        if(dataList[i][25] == "" or dataList[i][25]== "N/A"):
            temp["Temperature Low"] = ""
        else:
            temp["Temperature Low"] = float(dataList[i][25])
        #Grabs Speed Data
        if(dataList[i][28] == "" or dataList[i][28]== "N/A"):
            temp["Speed High"] = ""
        else:
            temp["Speed High"] = float(dataList[i][28])
        if(dataList[i][29] == "" or dataList[i][29]== "N/A"):
            temp["Speed Low"] = ""
        else:
            temp["Speed Low"] = float(dataList[i][29])
        #Grabs Needle Size
        if(dataList[i][18] == "" or dataList[i][18] =="N/A"):
```

```
                temp["Needle Size"] = ""
            else:
                if dataList[i][19].lower() == "gauge":
                    temp["Needle Size"] = float(needleSizeConverter(dataList[i][18]))
                else:
                    temp["Needle Size"] = float(dataList[i][18])
             #Needle Type
            if(dataList[i][17] == "" or dataList[i][17] == "N/A"):
                temp["Needle Type"] = ""
            else:
                temp["Needle Type"] = dataList[i][17]
            #Grabs Concentration Data
            if(dataList[i][materialInt+1] == "" or dataList[i][materialInt+1]=="N/A"):
                temp["Concentration"] = ""
            else:
                if(dataList[i][materialInt+2].lower() != "mg/ml"):
                    temp["Concentration"] = float(dataList[i][materialInt+1])
                else:
                    temp["Concentration"] = float(dataList[i][materialInt+1])/10
            temp["Row Number"] = i+1
            temp.update(POTP(temp))
            matDict[counter] = temp
        counter += 1
    return matDict

matDict = matParser(dataList, soloOrNot)
#Creates a Spreadsheet
workbook=xlsxwriter.Workbook(soloOrNot + material.replace(" ", "") + "X" + variableX + "Y" +
variableY + "Z" + variableZ + ".xlsx")
worksheet = workbook.add_worksheet()
#Creates Columnns Containing Desired Parameters
worksheet.write(0,0,"Material")
worksheet.write(0,1,"Concentration: wt%")
worksheet.write(0,2,"Pressure: bar")
worksheet.write(0,3,"Temperature: C")
worksheet.write(0,4,"Speed: mm/s")
worksheet.write(0,5,"Needle Size")
worksheet.write(0,6,"Molecular Weight")
worksheet.write(0,7,"Product of Temperature and Pressure")
worksheet.write(0,8,"Needle Type")
counter = 1


#Creates All Possible Permutations of the Three Variables
#For example: X=Molecular Weight, Y=POTP, and Z=Needle Size
#Function will Run Through Each Bioink Combination and Create All Possible Combinations of
Molecular Weight High, Molecular Weight Low, HighHigh, HighLow, LowHigh, LowLow, and Needle
Size
def permutation():
    listX = amountOfParameters(variableX)
    listY = amountOfParameters(variableY)
```

```
    listZ = amountOfParameters(variableZ)
    counter = 1
    for k in matDict.keys():
        v = matDict[k]
        for elemX in listX:
            for elemY in listY:
                for elemZ in listZ:
                    if(elemX in v and elemY in v and elemZ in v):
                        if(v[elemX] != "" and v[elemY] != "" and v[elemZ]!= ""):
                            worksheet.write(counter,xIndex, v[elemX])
                            worksheet.write(counter, yIndex, v[elemY])
                            worksheet.write(counter, zIndex, v[elemZ])
                            worksheet.write(counter,0,material)
                            worksheet.write(counter, 5, v["Needle Size"])
                            worksheet.write(counter,8, v["Needle Type"])
                            worksheet.write(counter, 1, v["Concentration"])

                            counter+=1
permutation()
workbook.close()
```

Cell Printing.py

```
# This program returns the Cell Types and a spreadsheet of all cellular printing with needle diameter and
needle type

import os, csv, pprint, xlsxwriter
#User Input
dataFile = open("MasterDatabase.csv", encoding="utf8")
dataReader = csv.reader(dataFile)
dataList = list(dataReader)
workbook=xlsxwriter.Workbook("CellPrinting.xlsx")
cellPrinting = workbook.add_worksheet('AmountofCells')
cellHeatMap = workbook.add_worksheet('CellHeatMap')


#Converts Gauge to um
def needleSizeConverter(needleSize):
    needleSizes = {
        "10": 2692,
        "11": 2388,
        "12": 2159,
        "13": 1803,
        "14": 1600,
        "15": 1372,
        "16": 1194,
        "17": 1067,
        "18": 838,
        "19": 686,
        "20": 603,
```

```
        "21": 514,
        "22": 413,
        "23": 337,
        "24": 311,
        "25": 260,
        "26": 260,
        "27": 210,
        "28": 184,
        "29": 184,
        "30": 159,
        "31": 133,
        "32": 108,
        "33": 108,
        "34": 51,
    }
    return needleSizes[str(needleSize)]

#Returns the Cell Types into A Spreadsheet
def cellTypes():
    cellDict = {}
    for i in range(3, len(dataList)):
        cell = dataList[i][15].strip()
        if cell.lower() != "cell composition" and cell != "" and cell.lower()!= "type":
            if cell in cellDict:
                cellDict[cell] = cellDict[cell]+1
            else:
                cellDict[cell] = 1
    cellPrinting.write(0,0, "CELL TYPE")
    cellPrinting.write(0,1, "AMOUNT OF TIMES")
    counter = 1
    for key, value, in cellDict.items():
        cellPrinting.write(counter,0, key)
        cellPrinting.write(counter, 1, value)
        counter +=1
def matParser(dataList):
    matDict = {}
    counter = 0
    #Iterates down the Spreadsheet
    for i in range (3, len(dataList) ,1):
        if(dataList[i][15]!=""):
            temp={}
            #Grabs Pressure Data
            if(dataList[i][20] == "" or dataList[i][20]=="N/A"):
                temp["Pressure High"] = ""
            else:
                if(dataList[i][21].lower()=="kpa"):
                    temp["Pressure High"] = float(dataList[i][20])/100
                elif(dataList[i][21].lower()=="psi"):
                    temp["Pressure High"] = float(dataList[i][20])/14.504
                else:
                    temp["Pressure High"] = float(dataList[i][20])
```

```
        if(dataList[i][22] == "" or dataList[i][22]=="N/A"):
            temp["Pressure Low"] = ""
        else:
            if(dataList[i][23].lower()=="kpa"):
                temp["Pressure Low"] = float(dataList[i][22])/100
            elif(dataList[i][23].lower()=="psi"):
                temp["Pressure Low"] = float(dataList[i][22])/14.504
            else:
                temp["Pressure Low"] = float(dataList[i][22])
        #Grabs Temperature Data
        if(dataList[i][24] == "" or dataList[i][24]=="N/A"):
            temp["Temperature High"] = ""
        else:
            temp["Temperature High"] = float(dataList[i][24])
        if(dataList[i][25] == "" or dataList[i][25]== "N/A"):
            temp["Temperature Low"] = ""
        else:
            temp["Temperature Low"] = float(dataList[i][25])
        #Grab Needle Size Data
        if(dataList[i][18] == "" or dataList[i][18] =="N/A"):
            temp["Needle Size"] = ""
        else:
            if dataList[i][19].lower() == "gauge":
                temp["Needle Size"] = float(needleSizeConverter(dataList[i][18]))
            else:
                temp["Needle Size"] = float(dataList[i][18])
        #Needle Type
        if(dataList[i][17] == "" or dataList[i][17] == "N/A"):
            temp["Needle Type"] = ""
        else:
            temp["Needle Type"] = dataList[i][17]
        #Grabs Speed Data
        if(dataList[i][28] == "" or dataList[i][28]== "N/A"):
            temp["Speed High"] = ""
        else:
            temp["Speed High"] = float(dataList[i][28])
        if(dataList[i][29] == "" or dataList[i][29]== "N/A"):
            temp["Speed Low"] = ""
        else:
            temp["Speed Low"] = float(dataList[i][29])
        #Which Row the Bioink is Located on the Spreadsheet
        temp["Row Number"] = i+1
        matDict[counter] = temp
        counter+=1
    return matDict
matDict = matParser(dataList)
#Returns a Spreadsheet of the Needle Diameter and Needle Type of Cellular Data
def cellularHeatMap():
    cellDict = {}
    counter = 1
    cellHeatMap.write(0,0, "NOZZLE DIAMETER")
```

```
    cellHeatMap.write(0,1, "NEEDLE TYPE")
    cellHeatMap.write(0,2, "TEMPERATURE")
    cellHeatMap.write(0,3, "PRESSURE")
    cellHeatMap.write(0,4, "SPEED")
    for i in matDict.keys():
        v=matDict[i]
        for elemX in ["Temperature High", "Temperature Low"]:
            for elemY in ["Pressure High", "Pressure Low"]:
                for elemZ in ["Speed High", "Speed Low"]:
                    if(elemX in v and elemY in v and elemZ in v):
                        if(v[elemX] != "" and v[elemY] != "" and v[elemZ]!= "" and v["Needle Type"] != "" and
v["Needle Size"] != ""):
                            cellHeatMap.write(counter,2, v[elemX])
                            cellHeatMap.write(counter, 3, v[elemY])
                            cellHeatMap.write(counter, 4, v[elemZ])
                            cellHeatMap.write(counter, 0, v["Needle Size"])
                            if(v["Needle Type"].lower() == "conical"):
                                cellHeatMap.write(counter, 1, 1)
                            else:
                                cellHeatMap.write(counter, 1,  2)
                            counter+=1


cellTypes()
cellularHeatMap()
workbook.close()
```