```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 1,
   "metadata": {
    "collapsed": true
   },
   "outputs": [],
   "source": [
    "%matplotlib inline"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 2,
   "metadata": {
    "collapsed": true
   },
   "outputs": [],
   "source": [
    "import pandas as pd\n",
    "import matplotlib.pyplot as plt\n",
    "import matplotlib.colors as mcolors\n",
    "import matplotlib.colorbar as mcbar\n",
    "import matplotlib.cm as cm\n",
    "import matplotlib.style as mstyle\n",
    "import numpy as np\n",
    "from mpl_toolkits.axes_grid1.axes_divider import
make_axes_locatable\n",
    "import seaborn as sns"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 3,
   "metadata": {
    "collapsed": true
   },
   "outputs": [],
   "source": [
    "def parseData(df):\n",
    "    CTevs = df.CTev.unique()\n",
    "    NTevs = df.NTev.unique()\n",
    "    NTevs = list(reversed(NTevs))\n",
    "    dataON = []\n",
    "    dataOFF = []\n",
    "    dataFI = []\n",
    "    dataG = []\n",
    "    for ntev in NTevs:\n",
    "        ONS = []\n",
    "        OFFS = []\n",
    "        FIS = []\n",
    "        GS = []\n",
    "        for ctev in CTevs:\n",
```

```
    "            dfN = df.loc[df['NTev'] == ntev]\n",
    "            dfNC = dfN.loc[df['CTev'] == ctev]\n",
    "            ONS.append(dfNC['ON'].item())\n",
    "            OFFS.append(dfNC['OFF'].item())\n",
    "            FIS.append(dfNC['FI'].item())\n",
    "            GS.append(dfNC['Delta G'].item())\n",
    "        dataON.append(ONS)\n",
    "        dataOFF.append(OFFS)\n",
    "        dataFI.append(FIS)\n",
    "        dataG.append(GS)\n",
    "    return NTevs, CTevs, dataON, dataOFF, dataFI, dataG"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 4,
   "metadata": {
    "collapsed": true
   },
   "outputs": [],
   "source": [
    "def plotDataEdgesBGR(NTevs, CTevs, dataON, dataOFF, dataFI, dataG,
minDot, title, filetype):\n",
    "    cmap = mcolors.LinearSegmentedColormap.from_list(\"\", [(0,(0.2,
0.2, 0.4)), \n",
    "
(0.1666,(0.8980392156862745, 0.8980392156862745, 0.8980392156862745)), \n",
    "                                                    (1.0,
(0.8823529411764706, 0.1607843137254902, 0.09411764705882353))])\n",
    "    normalizeH = mcolors.Normalize(vmin=-5, vmax=25)\n",
    "    mstyle.use('default')\n",
    "    \n",
    "    figBubbleON = plt.figure(figsize=(30,25))\n",
    "    dfSNSON = pd.DataFrame({'CTev' : [], 'NTev' : [], 'Magnitude' :
[], 'Delta G' : []})\n",
    "    EdgeList = []\n",
    "    for i in range(0,len(dataON)):\n",
    "        for j in range(0,len(dataON[i])):\n",
    "            if dataFI[i][j] == dataFI[i][j]:\n",
    "                if dataFI[i][j] >= 1.2:\n",
    "                    edge = 'black'                \n",
    "                else:\n",
    "                    edge = 'none'\n",
    "                EdgeList.append(edge)\n",
    "            dfSNSON = dfSNSON.append({'CTev' : j, 'NTev': i,
'Magnitude': dataON[i][j], 'Delta G' : dataG[i][j]},\n",
    "                                     ignore_index=True)\n",
    "    axON = sns.scatterplot(x='CTev', y='NTev', data=dfSNSON,
hue='Delta G', hue_norm=normalizeH, size='Magnitude', \n",
    "                           palette=cmap, edgecolor=EdgeList,
linewidth=2, sizes=(minDot,5000))\n",
    "    hON,lON = axON.get_legend_handles_labels()\n",
    "    axON.legend(hON[-5:],lON[-5:], bbox_to_anchor=(1.22, 1.0),
frameon=False, labelspacing=5, handletextpad=5)\n",
    "    plt.setp(axON.get_legend().get_texts(), fontsize=30)\n",
    "    plt.setp(axON.get_legend().get_title(), fontsize=50)\n",
```

```
    "    axON.set_xticks(np.arange(len(CTevs)))\n",
    "    axON.set_yticks(np.arange(len(NTevs)))\n",
    "    axON.set_xticklabels(CTevs, size=30, fontname='Arial',
rotation=45)\n",
    "    axON.set_yticklabels(NTevs, size=30, fontname='Arial')\n",
    "    axON.yaxis.set_ticks_position('left')\n",
    "    axON.xaxis.set_ticks_position('top')\n",
    "    axON.tick_params(top=False, left=False, labeltop=True,
labelleft=True)\n",
    "    axON.set_xlabel('CTEV', fontsize=40, fontname='Arial',
fontweight='bold', labelpad=20)\n",
    "    axON.set_ylabel('NTEV', fontsize=40, fontname='Arial',
fontweight='bold', labelpad=20)\n",
    "    axON.xaxis.set_label_position('top')\n",
    "    axON.set_title('ON', fontsize=50, fontname='Arial',
fontweight='bold', pad=40)\n",
    "    dividerON = make_axes_locatable(axON)\n",
    "    caxON, kwON = mcbar.make_axes(axON, shrink=0.5, pad=0.02,
anchor=(0.0,0.0))\n",
    "    cbarON = mcbar.ColorbarBase(caxON, cmap=cmap, norm=normalizeH)\n",
    "    cbarON.set_label('Delta G', fontsize=30, fontname='Arial',
rotation=360, labelpad=90)\n",
    "    cbarON.ax.tick_params(labelsize=30)\n",
    "    cbarON.ax.tick_params(pad=15)\n",
    "    plt.savefig(title + 'ON.' + filetype)\n",
    "    \n",
    "    figBubbleOFF = plt.figure(figsize=(30,25))\n",
    "    dfSNSOFF = pd.DataFrame({'CTev' : [], 'NTev' : [], 'Magnitude' :
[], 'Delta G' : []})\n",
    "    for i in range(0,len(dataOFF)):\n",
    "        for j in range(0,len(dataOFF[i])):\n",
    "            dfSNSOFF = dfSNSOFF.append({'CTev' : j, 'NTev': i,
'Magnitude': dataOFF[i][j], 'Delta G' : dataG[i][j]},\n",
    "                                       ignore_index=True)\n",
    "    axOFF = sns.scatterplot(x='CTev', y='NTev', data=dfSNSOFF,
hue='Delta G', hue_norm=normalizeH, size='Magnitude', \n",
    "                            palette=cmap, edgecolor=EdgeList,
linewidth=2, sizes=(minDot,5000))\n",
    "    hOFF,lOFF = axOFF.get_legend_handles_labels()\n",
    "    axOFF.legend(hOFF[-5:],lOFF[-5:], bbox_to_anchor=(1.22, 1.0),
frameon=False, labelspacing=5, handletextpad=5)\n",
    "    plt.setp(axOFF.get_legend().get_texts(), fontsize=30)\n",
    "    plt.setp(axOFF.get_legend().get_title(), fontsize=50)\n",
    "    axOFF.set_xticks(np.arange(len(CTevs)))\n",
    "    axOFF.set_yticks(np.arange(len(NTevs)))\n",
    "    axOFF.set_xticklabels(CTevs, size=30, fontname='Arial',
rotation=45)\n",
    "    axOFF.set_yticklabels(NTevs, size=30, fontname='Arial')\n",
    "    axOFF.yaxis.set_ticks_position('left')\n",
    "    axOFF.xaxis.set_ticks_position('top')\n",
    "    axOFF.tick_params(top=False, left=False, labeltop=True,
labelleft=True)\n",
    "    axOFF.set_xlabel('CTEV', fontsize=40, fontname='Arial',
fontweight='bold', labelpad=20)\n",
    "    axOFF.set_ylabel('NTEV', fontsize=40, fontname='Arial',
fontweight='bold', labelpad=20)\n",
```

```
    "    axOFF.xaxis.set_label_position('top')\n",
    "    axOFF.set_title('OFF', fontsize=50, fontname='Arial',
fontweight='bold', pad=40)\n",
    "    dividerOFF = make_axes_locatable(axOFF)\n",
    "    caxOFF, kwON = mcbar.make_axes(axOFF, shrink=0.5, pad=0.02,
anchor=(0.0,0.0))\n",
    "    cbarOFF = mcbar.ColorbarBase(caxOFF, cmap=cmap,
norm=normalizeH)\n",
    "    cbarOFF.set_label('Delta G', fontsize=30, fontname='Arial',
rotation=360, labelpad=90)\n",
    "    cbarOFF.ax.tick_params(labelsize=30)\n",
    "    cbarOFF.ax.tick_params(pad=15)\n",
    "    plt.savefig(title + 'OFF.' + filetype)\n",
    "    \n",
    "    figBubbleFI = plt.figure(figsize=(30,25))\n",
    "    dfSNSFI = pd.DataFrame({'CTev' : [], 'NTev' : [], 'FI' : [],
'Delta G' : [], })\n",
    "    for i in range(0,len(dataFI)):\n",
    "        for j in range(0,len(dataFI[i])):\n",
    "            dfSNSFI = dfSNSFI.append({'CTev' : j, 'NTev': i, 'FI':
dataFI[i][j], 'Delta G' : dataG[i][j]}, ignore_index=True)\n",
    "    axFI = sns.scatterplot(x='CTev', y='NTev', data=dfSNSFI,
hue='Delta G', hue_norm=normalizeH, size='FI', \n",
    "                           palette=cmap, edgecolor=EdgeList,
linewidth=2, sizes=(minDot,5000))\n",
    "    hFI,lFI = axFI.get_legend_handles_labels()\n",
    "    axFI.legend(hFI[-5:],lFI[-5:], bbox_to_anchor=(1.15, 1.0),
frameon=False, labelspacing=5, handletextpad=5)\n",
    "    plt.setp(axFI.get_legend().get_texts(), fontsize=30)\n",
    "    plt.setp(axFI.get_legend().get_title(), fontsize=50)\n",
    "    axFI.set_xticks(np.arange(len(CTevs)))\n",
    "    axFI.set_yticks(np.arange(len(NTevs)))\n",
    "    axFI.set_xticklabels(CTevs, size=30, fontname='Arial',
rotation=45)\n",
    "    axFI.set_yticklabels(NTevs, size=30, fontname='Arial')\n",
    "    axFI.yaxis.set_ticks_position('left')\n",
    "    axFI.xaxis.set_ticks_position('top')\n",
    "    axFI.tick_params(top=False, left=False, labeltop=True,
labelleft=True)\n",
    "    axFI.set_xlabel('CTEV', fontsize=40, fontname='Arial',
fontweight='bold', labelpad=20)\n",
    "    axFI.set_ylabel('NTEV', fontsize=40, fontname='Arial',
fontweight='bold', labelpad=20)\n",
    "    axFI.xaxis.set_label_position('top')\n",
    "    axFI.set_title('FI', fontsize=50, fontname='Arial',
fontweight='bold', pad=40)\n",
    "    dividerFI = make_axes_locatable(axFI)\n",
    "    caxFI, kwFI = mcbar.make_axes(axFI, shrink=0.5, pad=0.02,
anchor=(0.0,0.0))\n",
    "    cbarFI = mcbar.ColorbarBase(caxFI, cmap=cmap, norm=normalizeH)\n",
    "    cbarFI.set_label('Delta G', fontsize=30, fontname='Arial',
rotation=360, labelpad=90)\n",
    "    cbarFI.ax.tick_params(labelsize=30)\n",
    "    cbarFI.ax.tick_params(pad=15)\n",
    "    plt.savefig(title + 'FI.' + filetype)"
   ]
```

```
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "collapsed": true
   },
   "outputs": [],
   "source": []
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.6.8"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 2
}
```