

Untitled3

November 7, 2019

```
In [ ]: {
  "cells":
  {
    "cell_type": "markdown",
    "metadata": {},
    "source": [
      "# Statistics 3: Peak Integration and Internal Standards\n",
      "\n",
      "This Python exercise will cover how to use Python to do peak integration and quanti
    ],
  },
  {
    "cell_type": "code",
    "execution_count": 1,
    "metadata": {},
    "outputs": [],
    "source": [
      "# Import Everything you Need\n",
      "import numpy as np\n",
      "import matplotlib.pyplot as plt\n",
      "import matplotlib.patches as patches"
    ]
  },
  {
    "cell_type": "markdown",
    "metadata": {},
    "source": [
      "What is the area under the peak? The simplest explanation is that it is sum of the
    ]
  },
  {
    "cell_type": "code",
    "execution_count": 68,
    "metadata": {},
    "outputs": [
      {
        "name": "stdout",
        "output_type": "stream",
```

```

    "text": [
      "The integral is: 13.0\n"
    ],
  },
  {
    "data": {
      "image/png": "iVBORwOKGgoAAAAANSUhEUgAAAXwAAAAEKAYAAAAARnO4WAAAAABHNCSVQICAgIfAhkiAA
      "text/plain": [
        "<matplotlib.figure.Figure at 0x7f2b41557fd0>"
      ]
    },
    "metadata": {},
    "output_type": "display_data"
  }
],
"source": [
  "# Define the data points\n",
  "x1 = 1\n",
  "x2 = 3\n",
  "y1 = 5\n",
  "y2 = 8\n",
  "\n",
  "# Calculate the average y and change in x\n",
  "average_y = (y1 + y2)/2\n",
  "delta_x = (x2-x1)\n",
  "\n",
  "# Calculate the integral\n",
  "integral = average_y * delta_x\n",
  "print(\"The integral is:\", integral)\n",
  "\n",
  "# Plot the results\n",
  "plt.plot(x1, y1, marker='o', color='blue')\n",
  "plt.plot(x2, y2, marker='o', color='blue')\n",
  "plt.plot(2, average_y, marker='o', color='red')\n",
  "plt.gca().add_patch(patches.Rectangle((x1, 0),delta_x,average_y, color='red'))\n",
  "x = [x1, x2, x2, x1]\n",
  "y = [0, 0, y2, y1]\n",
  "plt.gca().add_patch(patches.Polygon(xy=list(zip(x,y)), color = 'blue', alpha = 0.5))\n",
  "plt.ylim(0,8.5)\n",
  "plt.xlabel('X')\n",
  "plt.ylabel('Y')\n",
  "plt.show()"
],
},
{
  "cell_type": "markdown",
  "metadata": {
    "slideshow": {

```



```

"source": [
    "Before we write our function, we need to introduce the idea of a \"for\" loop. For
]
},
{
    "cell_type": "code",
    "execution_count": 6,
    "metadata": {},
    "outputs": [
        {
            "name": "stdout",
            "output_type": "stream",
            "text": [
                "Starting to loop through elements in indexes: [0 1 2 3 4]\n",
                "The value of k in this loop is: 0\n",
                "Running Some Code...k squared is: 0\n",
                "The value of k in this loop is: 1\n",
                "Running Some Code...k squared is: 1\n",
                "The value of k in this loop is: 2\n",
                "Running Some Code...k squared is: 4\n",
                "The value of k in this loop is: 3\n",
                "Running Some Code...k squared is: 9\n",
                "The value of k in this loop is: 4\n",
                "Running Some Code...k squared is: 16\n",
                "For loop done\n"
            ]
        }
    ],
    "source": [
        "indexes = np.arange(0, 5)\n",
        "print(\"Starting to loop through elements in indexes:\", indexes)\n",
        "for k in indexes:\n",
        "    print(\"The value of k in this loop is:\", k)\n",
        "    s = k**2\n",
        "    print(\"Running Some Code...k squared is:\", s)\n",
        "print(\"For loop done\")"
    ]
},
{
    "cell_type": "markdown",
    "metadata": {},
    "source": [
        "Now, let's write a function to calculate the integral. Our data point area function
    ]
},
{
    "cell_type": "code",
    "execution_count": 7,

```

```

"metadata": {},
"outputs": [
  {
    "name": "stdout",
    "output_type": "stream",
    "text": [
      "0.999999997112\n"
    ]
  }
],
"source": [
  "# A function to calculate the trapezoid area for two points. This comes directly from\n",
  "def data_point_area(x1, x2, y1, y2):\n",
  "    average_y = (y1 + y2)/2\n",
  "    delta_x = (x2-x1)\n",
  "    area = average_y * delta_x\n",
  "    return area\n",
  "\n",
  "# A function to calculate the integral for an array\n",
  "def integral(x, y):\n",
  "    length = len(x) # Calculate the length of the x array\n",
  "    indexes = np.arange(1,length) # Create an array of indexes. Don't start at 0 because\n",
  "    sum_value = 0 # Set the sum value to 0 initially\n",
  "    for k in indexes:\n",
  "        # Get the correct x and y values from the list based on their indexes\n",
  "        x1 = x[k-1]\n",
  "        y1 = y[k-1]\n",
  "        x2 = x[k]\n",
  "        y2 = y[k]\n",
  "        # Calculate the area for those set of two points from the function above\n",
  "        area = data_point_area(x1, x2, y1, y2)\n",
  "        # Add the area from the two points to the total sum\n",
  "        sum_value = sum_value + area\n",
  "    return sum_value\n",
  "\n",
  "print(integral(x, y))
]
},
{
  "cell_type": "markdown",
  "metadata": {},
  "source": [
    "The integral should be almost exactly one. That is because the area of a Gaussian d
  ]
},
{
  "cell_type": "code",
  "execution_count": 8,

```

```

"metadata": {},
"outputs": [
  {
    "name": "stdout",
    "output_type": "stream",
    "text": [
      "0.999999997112\n"
    ]
  },
  {
    "source": [
      "print(np.trapz(y, x=x))"
    ]
  },
  {
    "cell_type": "markdown",
    "metadata": {},
    "source": [
      "The final thing we need to learn to put it all together is how to chop out specific\n",
      "After doing that, we can calculate the area under that part of the curve only. "
    ]
  },
  {
    "cell_type": "code",
    "execution_count": 10,
    "metadata": {},
    "outputs": [
      {
        "name": "stdout",
        "output_type": "stream",
        "text": [
          "Integral from -1 to 1 is: 0.656883274351\n"
        ]
      },
      {
        "data": {
          "text/plain": [
            "<matplotlib.figure.Figure at 0x7f4b1873c588>"
          ]
        },
        "metadata": {},
        "output_type": "display_data"
      }
    ],
    "source": [
      "# Chop an array to include only x values between minx and maxx\n",

```

```

def datachop(x, y, minx, maxx):\n",
    "    bool = np.logical_and(x <= maxx, x >= minx)\n",
    "    return x[bool], y[bool]\n",
    "\n",
    "# Set the min and max\n",
    "minx = -1\n",
    "maxx = 1\n",
    "\n",
    "# Get the new cut x and y arrays\n",
    "cutx, cuty = datachop(x, y, minx, maxx)\n",
    "\n",
    "# Calculate the area under the cut arrays\n",
    "area = integral(cutx, cuty)\n",
    "print(\"Integral from\", minx, \"to\", maxx, \"is:\", area)\n",
    "\n",
    "# Plot the cut arrays in red against the original in black\n",
    "plt.plot(x, y, color=\"black\")\n",
    "plt.plot(cutx, cuty, marker=\"o\", color=\"red\")\n",
    "plt.xlabel(\"z\")\n",
    "plt.show()"
]
},
{
    "cell_type": "markdown",
    "metadata": {},
    "source": [
        "Now, we should put everything together to make it easier and in one function."
    ]
},
{
    "cell_type": "code",
    "execution_count": 11,
    "metadata": {},
    "outputs": [
        {
            "name": "stdout",
            "output_type": "stream",
            "text": [
                "0.656883274351\n"
            ]
        }
    ],
    "source": [
        "def integral_range(x, y, minx, maxx):\n",
        "    cutx, cuty = datachop(x, y, minx, maxx)\n",
        "    area = np.trapz(cuty, x = cutx)\n",
        "    return area\n",
        "\n"
    ]
}
],
"source": [
    "def integral_range(x, y, minx, maxx):\n",
    "    cutx, cuty = datachop(x, y, minx, maxx)\n",
    "    area = np.trapz(cuty, x = cutx)\n",
    "    return area\n",
    "\n"
]

```

```

        "# Test it\n",
        "print(integral_range(x, y, minx,maxx))"
    ]
},
{
    "cell_type": "markdown",
    "metadata": {},
    "source": [
        "Because the mean is 0 and the standard deviation is 1, the  $x$  axis is actually  $z$ \n",
        "\n",
        "**Problem 1:** Using the code above, calculate the area under our Gaussian curve fr
    ]
},
{
    "cell_type": "code",
    "execution_count": 75,
    "metadata": {},
    "outputs": [],
    "source": [
        "# Your code here!"
    ]
},
{
    "cell_type": "markdown",
    "metadata": {},
    "source": [
        "Now, let's apply what you've learned to a simulated chromatogram. This code will ge
    ]
},
{
    "cell_type": "code",
    "execution_count": 12,
    "metadata": {},
    "outputs": [
        {
            "data": {
                "text/plain": [
                    "<matplotlib.figure.Figure at 0x7f4b1872c048>"
                ]
            },
            "metadata": {},
            "output_type": "display_data"
        }
    ],
    "source": [
        "# Set the retention times\n",
        "t1 = 10\n",

```

```

    "t2 = 15\n",
    "\n",
    "# Set the peak widths\n",
    "w1 = 1\n",
    "w2 = 1.5\n",
    "\n",
    "# Define the x axis\n",
    "x = np.arange(0,20, 0.1)\n",
    "\n",
    "# Sum together two gaussians and some noise. Note, the second one is set to have ha
    "y = Gaussian(x, t1, w1/4) + 0.5*Gaussian(x, t2, w2/4) + np.random.normal(0,0.01, le
    "\n",
    "# Plot the results\n",
    "plt.plot(x, y)\n",
    "plt.xlabel(\"Retention Time\")\n",
    "plt.ylabel(\"Absorbance\")\n",
    "plt.text(t1+1, 1.4, \"A\")\n",
    "plt.text(t2, 0.7, \"B\")\n",
    "plt.show()"
]
},
{
    "cell_type": "markdown",
    "metadata": {},
    "source": [
        "**Problem 2:** Find the a) the area of peak A, b) the area of peak B, and c)
    ]
},
{
    "cell_type": "code",
    "execution_count": 78,
    "metadata": {},
    "outputs": [],
    "source": [
        "# Your code here\n",
        "# Hint: you should use the integral_range function from above"
    ]
},
{
    "cell_type": "markdown",
    "metadata": {},
    "source": [
        "**Problem 3:** Assume peak A is the internal standard with a known molar absorptivi
    ]
},
{
    "cell_type": "code",
    "execution_count": 79,

```

```

    "metadata": {},
    "outputs": [],
    "source": [
        "# Your code here"
    ]
}
],
"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": "3.6.2"
    }
},
"nbformat": 4,
"nbformat_minor": 2
}

```