Business Data Science Assignment 1

Submitted By:
Aman Bhardwaj
Manvi Mahajan
Rithu Anand Krishnan

In [ ]:

```
#QUESTION 1 AND QUESTION 2 ARE IN THE SAME JUPYTR NOTEBOOK
```

In [ ]:

```
#1. Create 1000 samples from a Gaussian distribution with mean -10 and standard deviation 5. Create another 1000 samples from another independent Gaussian with mean 10 and standard deviation 5.
#(a) Take the sum of these Gaussians by adding the two sets of 1000 points, point by point, and plot the histogram of the resulting 1000 points. What do you observe? Deliverables: three histograms (two for each Gaussian, one for the sum), written response, code
#(b) Estimate the mean and the variance of the sum.
#Deliverables: written response, code
```

In [ ]:

```
#Declaring the libraries
```

In [6]:

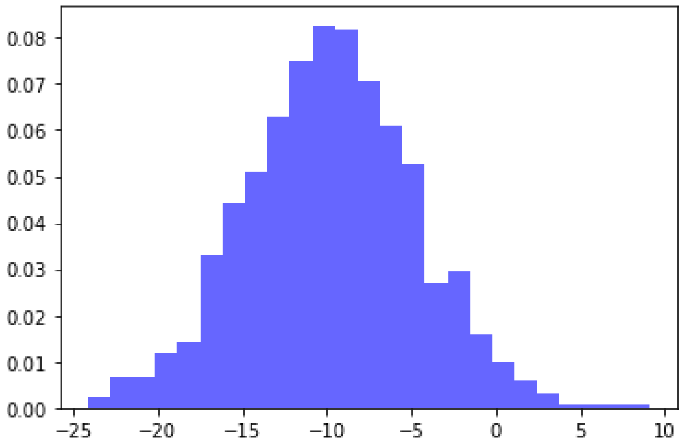
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [ ]:

```
#Generating 1st set of 1000 samples from gaussian distribution
```

In [9]:

```
mu = -10
sigma = 5
s1 = np.random.normal(mu,sigma,1000)
plt.hist(s1, bins=25, density=True, alpha=0.6, color='b')
plt.show()
#Generating histogram for the first set
```

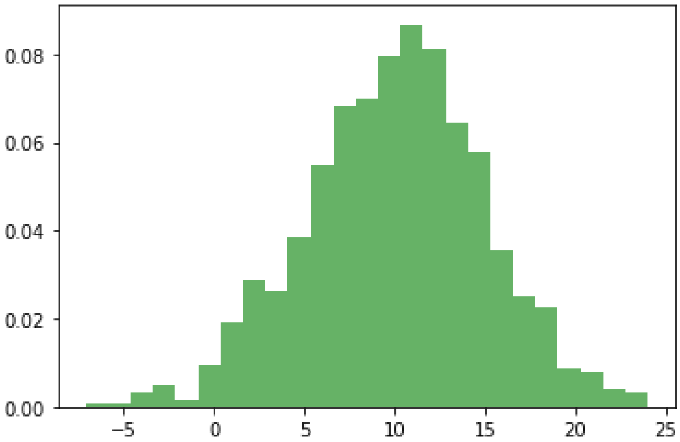


In [ ]:

```
#Generating 2nd set of 1000 samples from gaussian distribution
```

In [14]:

```
mu = 10
sigma = 5
s2 = np.random.normal(mu,sigma,1000)
plt.hist(s2, bins=25, density=True, alpha=0.6, color='g')
plt.show()
#histogram for the 2nd set of 1000 samples
```



In [ ]:

```
#Adding both the samples point by point
```

In [53]:

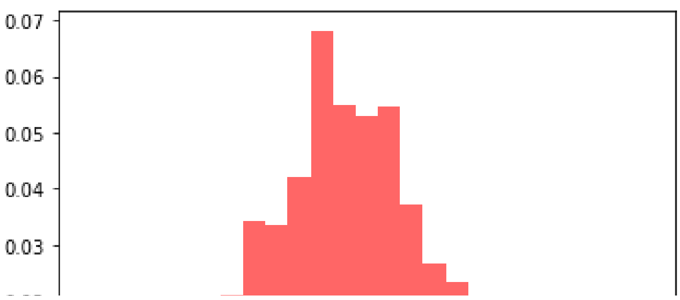
```
sum = np.add(s1,s2)
print(sum)
plt.hist(sum, bins=25, density=True, alpha=0.6, color='r')
plt.show()
#Histogram depicting sum of samples
```

```
[-1.75377299e+00  7.64505289e+00  2.48001658e+00  1.72026346e+01
  4.08929101e-01  7.02001883e+00 -2.72394812e+00  6.98268911e+00
 -9.03793899e+00 -7.44048441e+00 -4.18677571e+00  5.49749809e-01
 -9.77369511e+00  1.85790676e+00  5.65991479e-01  1.76806868e+00
 -3.51727000e+00  6.73641926e+00  6.63901590e+00  5.23756594e+00
  3.99136419e-01  9.89875915e+00  1.76449995e+00  5.01452915e+00
 -7.22102671e-01  5.66899422e+00 -1.06084021e+01  1.45364244e+01
  1.26877193e+00 -4.87586518e+00  8.41828622e+00  1.01393779e+01
  2.22538386e+00 -2.57005313e-01  2.64831398e+01 -2.94998307e+00
 -7.29461914e+00 -2.26581776e+01  1.33546829e+01 -8.16855701e+00
  3.36642826e+00 -1.06082759e+01 -3.40595710e+00 -2.75743524e+00
 -7.02812921e+00  2.90877800e+00  3.18734473e+00 -1.20514549e+01
  4.93839440e+00 -1.94930487e+00  6.93329955e+00 -1.67638419e+00
 -7.24787473e+00 -1.43234027e+01  4.06842777e+00  3.03252274e+00
 -2.54686241e+00 -6.95320340e+00 -3.76998307e+00  3.72295727e+00
 -8.73033708e+00 -6.00697278e+00  1.27315638e+01  3.09672225e+00
 -4.65033416e+00  1.95158172e+00 -7.53540988e+00  3.90177044e-01
 -1.08646906e+00 -3.00436346e-01  1.91835791e+00 -2.15378502e+00
 -1.58800945e+00  2.37712831e+00  9.01181022e+00  3.58448395e+00
 -4.01254224e+00  1.16700137e+01  2.19507111e+00 -8.34582310e+00
  1.56595682e+00 -5.25568049e+00  7.69846231e-01  1.19445521e+01
  1.31590298e+00 -7.73743735e+00 -5.08357456e+00 -2.47613118e-02
  6.59772401e+00  1.16244165e+01  9.24636934e+00 -1.09790016e+01
  9.54290966e+00 -9.83722240e+00  3.26417810e+00  8.31616304e+00
  6.11335152e+00  1.25473309e+00 -2.44453389e+00  6.67170668e+00
  6.52768430e+00 -7.05648214e+00  1.55052089e+01 -6.89212969e+00
 -5.07040322e+00  1.03711455e+00 -3.09570834e+00 -3.18938506e-01
  1.64791953e+00 -5.18878066e+00  1.96872757e-01 -9.54140567e+00
 -2.28542372e+00 -4.58102895e+00 -2.68190005e+00 -1.27573045e+01
 -1.36552368e+00  6.82692293e-01 -7.24372452e+00 -5.16834801e-01
  1.43397773e+00 -1.51700068e+00  6.96173258e+00 -7.06709366e+00
  2.57052298e+00  3.69985865e+00  9.22998309e+00 -9.29924341e+00
  4.94900036e+00 -5.40342185e+00 -1.54209386e+01 -4.66214115e+00
  5.59642484e+00  9.57363227e+00  2.08274394e+00 -8.31727377e+00
 -7.22502747e+00 -4.94157663e+00 -8.41519248e+00 -5.77420537e+00
  5.30089325e+00 -3.52590610e+00  1.16572758e+01 -5.67400129e+00
  2.41181584e+00  1.07481580e+01  1.22435478e+01 -4.57792759e+00
 -3.45647567e+00 -1.45906834e+00 -3.69902006e-01 -6.36567823e+00
  4.48280649e+00  3.62990383e+00  3.92432000e+00 -2.17572483e+00
 -4.00522269e+00 -1.94871673e+00 -5.71249016e+00 -3.64431299e+00
  4.49402016e+00  4.64799361e+00  1.66256391e+01 -2.48173592e+00
  6.08782674e+00  1.07691548e+01 -1.23873920e+01 -7.69586147e+00
 -2.87092749e+00  8.60184191e+00  1.47641665e+01  4.21193573e+00
 -3.00279622e+00  6.62910680e+00 -2.39713878e+00  4.29109955e+00
```

5.02199550e+00	6.69386454e+00	6.35709487e+00	-4.06995232e+00
2.16439938e-01	6.71552813e+00	-3.16791196e+00	-7.06696822e-01
1.15369928e+01	1.61838461e+00	-8.90971449e+00	-1.59725472e+01
-3.38696116e-01	4.16506300e+00	-3.85723772e+00	9.04476112e+00
-2.10966280e+01	1.27676201e+00	2.64741938e+00	-3.60125075e+00
3.09517506e+00	-1.91214791e+00	4.25064573e+00	-8.69085853e+00
-2.24384711e+00	-7.88820309e+00	-3.91830445e-01	-1.02819318e+01
6.78184910e+00	-1.35191251e+01	-4.91007221e+00	-9.23084552e+00
5.30341486e+00	3.55913817e+00	1.47077054e+00	1.61109388e+00
-2.16927378e+00	1.10946433e+01	-1.97634855e+00	1.06907356e+01
-1.07884791e+01	1.85978940e+01	1.22294245e+01	-1.03944700e+01
8.74141986e+00	3.79949977e+00	8.91528025e+00	8.89892073e-01
-7.94851315e+00	6.49792474e-01	-7.49150044e+00	-9.83776441e+00
-5.38044056e+00	4.21648298e+00	4.10071532e+00	4.47253282e+00
1.00207402e+01	-4.29191662e+00	-5.55047853e+00	-1.81579185e+01
6.41902333e+00	-4.40949817e+00	-7.08742808e-01	-2.56497989e-01
-1.47133224e+01	6.19005471e+00	-5.22700658e-01	-2.83674441e+00
7.20243590e+00	-1.00447285e+00	3.87757519e+00	9.27136947e-01
2.29724806e-02	9.29892014e+00	-5.19705180e+00	-2.08181227e+00
-9.17504937e-01	7.38311105e+00	-7.60442407e-02	2.84645284e+00
-7.53951131e+00	-1.31393209e+01	-6.94089535e+00	2.67914296e+00
-8.38641342e-01	4.25469486e+00	1.46143591e+01	5.61809535e+00
8.77238270e+00	-5.15923782e+00	-8.68913684e-02	9.25613831e-01
1.04230676e+01	-6.57245368e+00	2.95637406e+00	-6.65798271e+00
-3.22184297e+00	3.91760265e+00	-1.25858580e+00	-6.04120914e+00
8.51621835e+00	-6.42699352e-01	-2.32199128e+00	9.80454225e+00
2.48291334e+00	1.28881570e+01	-8.18529533e+00	5.13070316e+00
3.25002014e+00	9.45009663e+00	-4.53892471e-01	7.36180163e+00
6.38294478e+00	-7.83260902e+00	2.57514704e+00	1.03107275e+01
3.86736883e+00	-8.19588391e+00	1.99328178e+00	-7.70077694e+00
-8.95462268e-01	-7.67506467e+00	-1.00367960e+01	4.65968453e+00
-1.48075387e+00	-2.44764494e+00	-8.06890803e+00	1.37807574e+01
1.27564673e+01	-2.47301482e+00	1.55434963e-01	-3.56298160e+00
-1.63872931e+00	2.93560367e+00	1.97544098e+00	2.14635044e+00
-8.40257660e+00	-6.66269456e+00	-2.95999804e+00	5.80541406e-01
4.03572243e+00	4.84468817e-01	6.46951892e-01	-3.36131640e+00
3.26394286e+00	-1.55009648e+00	-4.70439979e+00	1.84841099e+01
-8.72689227e+00	-2.97510864e-01	-6.77960331e+00	4.68538883e+00
8.31201039e+00	4.13212191e-01	-3.08581366e+00	3.25999311e+00
8.64115632e+00	1.10867328e+01	7.90528765e+00	8.99350263e+00
-5.80159277e+00	2.33482725e+00	-9.70936466e+00	-1.46489007e+01
-2.41231669e+00	1.63256309e+01	-4.18240470e+00	7.11250636e+00
-1.84198692e+00	-5.69536179e+00	-1.18190107e+01	1.09448451e+00
1.83829200e+00	5.73959946e+00	6.20475038e+00	-4.39186812e+00
2.30166415e-01	-1.71450272e+01	-1.11943372e+01	7.03513494e+00
-1.10010042e+01	6.47373867e+00	-1.72784774e+01	2.56369198e+00
3.86065833e+00	1.37263200e+00	6.65500395e+00	-7.09388777e+00
5.61760564e+00	-2.75203258e+00	1.02483357e+01	7.73155899e-01
8.40927063e+00	5.42790641e+00	-5.59974161e+00	1.63985679e+00
-5.97193166e+00	5.77081765e+00	-1.63827099e+00	5.72135637e+00
2.49601288e-01	-3.69083402e+00	-7.32731175e+00	1.85609374e+00
-4.23685327e-01	-6.46431384e+00	4.86848302e+00	2.61554187e+00
-7.07381934e+00	-1.12426459e+01	-7.37050135e+00	-5.06686601e+00
5.11451327e-01	-9.21108817e+00	1.17989606e+00	2.47099268e+00
4.48662050e+00	1.06049684e+01	2.42438641e+00	-1.24907808e+01
-3.82482428e+00	9.64528051e+00	3.54648417e+00	-5.85553931e+00
-2.26371201e+00	3.54457888e+00	-1.11202256e+00	-5.82594780e-01
-8.03231392e+00	1.54548909e+01	1.38393884e+00	1.46069153e+00
8.75989696e-02	5.87110596e+00	-1.04023019e+01	7.56678753e+00
-2.78461492e+00	1.21591853e+01	1.33065168e+00	-4.08505870e-01
6.83254166e+00	-4.40706738e-01	2.46545318e+00	5.02438423e+00
-3.74990295e+00	7.61965082e+00	-4.04124768e+00	1.10747631e+01
5.70126405e+00	-1.38397540e+00	-2.84610619e+00	1.95634694e+00
-3.71735962e+00	-1.01210988e+01	8.26252071e+00	4.47260053e+00
3.10005742e+00	-1.25473178e+00	1.75032804e-01	-8.53144952e+00
-1.42113983e+00	1.25974246e+01	-7.03636266e+00	1.29047480e+01
2.85920539e+00	-5.80645912e+00	-6.94044932e+00	-2.44446697e+00
7.33030414e+00	2.56721342e-01	1.32323411e+01	1.12300464e+01
1.45694247e+01	-5.26828983e+00	-2.16714321e+00	7.06655029e+00
-1.36726949e+00	7.56326246e+00	1.45247900e+00	1.42203590e+00
-1.52584464e+00	-6.65527794e+00	1.73319435e+00	3.91927256e+00
1.07402211e+01	7.78272707e+00	-5.50599994e+00	-7.11051432e+00
[illegible]	[illegible]	[illegible]	[illegible]

4.40056663e+00	-1.69481188e+01	2.98369638e+00	3.75803454e+00
4.94607496e+00	-8.21051022e+00	-1.09572756e+01	-4.44599763e+00
-7.98526193e+00	-5.66538964e-01	1.18842693e+01	9.40720043e+00
1.21983583e+00	5.44880252e+00	1.17715829e+01	-2.80022186e+00
-6.11511435e+00	5.22197985e+00	-8.62045467e+00	-1.45892737e+01
-2.22149890e+00	6.26319390e+00	1.13361337e+01	-9.83390792e-01
-1.04739020e+01	-2.29625386e+00	6.72956426e+00	-2.19755183e+00
1.28349782e+00	-1.41674822e+00	-2.84584300e+00	7.51935559e+00
-4.23945059e+00	3.74683374e+00	2.32079834e+00	1.89272575e+00
1.09261707e+01	-3.52363290e+00	-2.23241682e+00	-1.11342850e+00
-4.03404080e+00	3.59334836e+00	5.87022812e+00	-2.65103689e-01
1.61579534e+01	-1.99598292e+00	1.19471041e+00	2.57164827e-02
4.43841128e+00	4.62467781e+00	-1.29471033e+01	-6.93152777e+00
-5.60401796e+00	-6.76348169e+00	-1.82462137e+00	1.77809422e+01
3.80969869e+00	-6.29913620e+00	-2.63525646e+00	-1.78206280e+00
1.65454310e-01	1.90715601e+01	-8.24234828e-01	-4.04547654e+00
-5.92443710e+00	1.60977253e+00	1.26463161e+00	-4.81904915e+00
7.84367207e+00	-6.67970792e+00	9.09678024e+00	-2.35931554e+00
4.14761266e+00	-3.84355280e+00	-8.65786862e+00	7.65905372e+00
6.32267871e+00	2.11043190e+00	-3.11777482e+00	-2.75558776e+00
2.28750622e+00	1.76993625e+00	2.00944268e+00	-7.20027168e+00
3.44961773e+00	-1.53717553e+00	2.33481789e+00	-2.81511597e+00
-5.02127878e+00	4.52243881e+00	4.35446871e+00	-2.35170666e+00
-1.46628019e+00	-6.90669348e+00	-8.54586991e+00	-4.40256688e-01
-1.25783712e+01	-6.21125051e-01	-6.28815494e+00	-2.46249060e+00
-4.03894179e+00	-4.57108511e-01	2.32340342e+00	-4.83157669e+00
-1.09069014e+01	-3.53083834e+00	-1.56105857e+00	3.88679492e+00
1.25520215e+00	-3.90101662e-01	7.22799428e-01	3.15955622e+00
-2.00494002e+00	7.19677022e+00	1.26230849e+01	3.11300921e+00
9.00438118e-01	2.31629603e+00	-7.74747344e+00	6.07603905e+00
-2.05244214e-01	6.86435026e+00	-4.33434010e+00	4.00689766e+00
-2.89995903e+00	2.19832935e+00	-1.17822548e+01	1.81945822e+01
2.67012132e+00	-3.23086113e-02	-2.76891629e+00	6.04077882e+00
6.69216983e+00	-4.89080540e+00	-1.04491032e+00	1.75822443e+00
2.20883761e-01	-2.97869710e+00	-5.13019464e+00	-9.69388870e+00
-1.96137464e+00	2.86793787e+00	3.83865309e+00	-7.66350546e+00
3.96134409e-01	1.07740997e+01	1.51497292e+01	7.90948492e+00
-3.93564168e+00	9.37216485e+00	5.33323978e+00	-3.08585949e+00
-1.58238557e-01	-3.13743678e-01	-2.71498359e+00	4.59684169e-01
1.33973751e+00	-1.18383466e+00	5.64139781e+00	4.67979626e+00
-2.39375963e+00	1.09894717e+01	3.53702423e+00	1.60203806e+00
9.57998412e+00	1.15742045e+01	-2.77606860e+00	-1.10159406e+00
-7.69069147e+00	-1.87258891e+01	-4.99046998e+00	4.89040304e+00
3.85554691e+00	1.14412233e+01	6.72836333e+00	-2.84444604e+00
-1.11371335e+00	-9.84428186e-01	-8.03183347e+00	-9.96689430e+00
-1.21039510e+00	-6.05438173e+00	-9.03838442e+00	9.96835192e+00
-1.07806375e+01	4.68596381e+00	-3.55402825e+00	-6.36488943e+00
-1.48997061e+00	4.47142984e+00	1.00264246e+01	-3.22010463e+00
4.07324778e+00	-1.47511897e+01	-3.08868321e+00	1.04485716e+01
9.18646487e+00	4.64299732e-02	6.97291892e+00	-4.06456238e-01
1.23299361e+01	1.49548842e+01	6.17542325e+00	-3.33500139e+00
-5.14503018e+00	-7.35740854e+00	4.39192280e+00	1.72812691e+00
1.96466274e+01	1.12119264e+01	3.19894070e+00	-1.66116774e+00
1.90393259e+00	3.55221398e+00	-9.15293143e+00	4.57057999e+00
1.00366471e+01	-2.66820379e+00	4.77332594e+00	-1.75373406e+00
-4.26189852e+00	1.57794077e+00	1.89715858e+01	-4.56372188e+00
-5.39212136e+00	9.17403069e+00	-1.19582124e+01	-8.66333024e+00
3.70376001e+00	-1.13843741e+01	-3.23145973e-01	1.41742942e+01
1.10996449e+00	-3.00845691e+00	8.33145115e+00	4.92322028e+00
1.79938718e+01	7.01499490e-01	-9.33364332e+00	3.36492340e+00
2.86802719e+00	6.21348816e+00	-1.52893321e+01	-1.65591028e+00
-2.18776524e+00	-1.44092262e+00	1.54463671e-02	4.09075860e+00
-1.54529721e+00	4.02406565e+00	-4.89968162e+00	-9.49775852e+00
1.51938011e+01	3.94102511e+00	1.01566083e+01	-1.96956481e+00
-8.87938261e+00	-7.18818238e+00	-7.30311198e-01	-2.91925795e+00
7.82952602e+00	8.14851114e+00	3.24672685e+00	1.15833243e+00
-3.54062789e+00	-4.43089524e+00	-1.59876403e+01	3.58745647e+00
-7.93798195e-01	9.89950065e+00	-4.90343483e+00	3.61017411e-02
9.29063865e+00	-1.75555785e+01	-1.02379528e+01	9.88467366e-02
2.52394381e+00	1.13217667e+01	-2.77131504e+00	-1.09059992e+00
-1.41027491e+01	-2.03223179e+00	-3.00381314e+00	8.72831834e+00
-6.78201211e+00	4.21926935e+00	-1.21387577e+01	-1.76357675e+00
[illegible]	[illegible]	[illegible]	[illegible]

```
 -1.43918071e+01  1.12625109e+01  1.01753573e+01  1.05431889e+01
 -2.64013940e+00 -2.96575238e+00  2.94456187e+00 -1.89351137e+01
 -9.42091240e+00 -2.88300163e+00 -1.07643733e+01 -8.06663890e+00
 -7.96511671e+00 -9.42116217e+00 -3.02785078e+00  6.21840831e+00
 -2.52335645e-01 -1.22189279e+01 -4.31908742e+00  7.42110958e+00
 -7.07211450e+00  2.42099923e+00  1.24686088e+01  6.31633881e+00
 -2.40579087e+00  5.07205148e+00 -1.62953251e+00  7.32491563e+00
 -3.46718934e+00  4.65804984e+00 -2.54902130e+00  1.13653853e+01
  3.71945552e+00  1.75376450e+00  4.71662739e+00  9.08411623e+00
  3.89170156e+00 -4.55349961e+00 -7.64613146e-01 -3.37408413e+00
  1.22234346e+00 -1.71643277e+00  6.82627910e+00  3.69255458e+00
 -2.52001564e+00 -4.28393202e+00  6.16127925e+00 -1.94441954e+00
 -2.25651579e+00 -1.36214764e+01  1.25674386e+00 -5.17265962e+00
 -1.30718429e+00  1.00873695e+00 -5.73281387e+00 -1.95805544e+00
 -1.06243073e+01 -5.72596328e+00 -4.28908877e+00  5.55516826e+00
  8.13827299e-01  2.73779183e+00 -2.01554903e+00 -9.34059774e+00
  1.43781738e+01  8.10798658e-01 -1.57120158e+01 -2.80165867e+00
 -7.03231707e+00 -1.51315965e+00 -5.17440559e+00 -5.00370666e+00
 -2.28843398e+00  5.25910709e-01 -6.32354095e+00  1.18220955e+00
  2.46833547e+00 -9.94269995e+00  2.38755806e+00 -2.13507720e-01
 -8.47799537e+00  1.18953268e+01 -1.76389220e+00 -9.16530472e-01
  4.10762872e+00  9.93373800e+00  1.51658455e+00 -1.88587414e+00
 -2.84976947e+00 -1.39624700e+00 -3.62139379e+00  6.66726076e+00
  3.83316667e+00  2.71949642e+00  1.09269550e+01 -5.38289228e-01
  8.01119007e+00 -4.95767583e-02  6.55860785e-01  3.81521041e+00
  1.69466921e+00  5.82028587e+00  4.10777041e+00 -1.01214032e+01
  1.98065445e+01 -8.48744796e+00 -1.49983889e+01 -5.42023765e+00
 -5.80332019e+00  2.64863658e+00 -2.09482412e+00 -3.64255459e+00
  8.06267903e+00  1.21530997e+01 -1.08233349e+01  4.10506063e+00
  1.16857338e+00 -2.27561933e+00 -1.73654867e+00  4.25433564e+00
 -5.72031701e+00 -6.95979403e+00 -4.47927831e+00  7.05672101e+00
 -9.29972186e+00  1.10324403e+00 -1.08720206e+01  5.76105298e-01
 -1.62038993e+00  4.02565405e+00 -2.04093545e+00  9.27465392e+00
 -3.97818064e+00  6.72777298e+00 -8.01078030e+00  2.12408445e+00
  1.08771016e+00  7.50800286e+00  1.74647349e-01 -6.07078516e+00
 -3.95082479e+00 -2.53602651e+00  6.35623543e+00 -6.36549495e-01
 -3.17357813e+00 -2.99891498e+00 -1.28254454e+00 -1.25246490e+00
  2.12375631e+00 -4.44474435e+00 -2.02904185e+00 -6.81123618e+00
 -4.80820321e+00 -1.88681662e+00 -4.16797338e+00  2.69645510e-01
  5.28667384e+00  1.04530212e+00  5.24717412e+00  1.52867670e-01
 -1.20340815e+01 -3.61250750e+00 -5.29433904e-02  8.62611716e+00
 -5.73334027e+00  8.98312302e+00  5.93117852e+00 -9.96401337e+00
  1.14311242e+01  9.53178327e+00  4.20726442e+00 -1.48324722e-01
  7.89508770e-01  9.36437770e+00  3.22381747e+00 -5.42207291e+00
  1.53741249e+01 -1.04515007e+01 -1.51247841e+00 -2.72217558e-01
 -1.26698026e+01  1.02656225e+01 -1.07282266e+01  3.99482926e+00
  6.07620666e+00 -2.97177676e-01 -6.70784245e+00  3.63410583e+00
 -2.28758060e+00  3.07894923e-01  5.98473016e+00 -8.17712727e+00
  4.09064185e+00 -4.88044621e+00  1.06005561e+00  4.60029258e+00
  7.05038315e+00  1.80166989e+00  7.14102939e+00 -9.43538158e+00
  7.63558111e+00 -5.48583861e+00 -9.88432553e-01  1.61820284e+01
 -1.14921641e+01  3.05277963e+00  1.56443795e+00 -4.63306123e-01
  4.59583685e+00 -1.01088973e+01  4.90953848e+00 -3.08754560e+00
 -7.08837279e+00  1.90364546e+00  1.75583058e+01  7.35948708e+00
  4.26892951e+00  8.83900779e+00 -9.63468783e-01  7.11242838e+00
 -6.39898750e+00  4.27949846e-01 -8.34592337e+00  1.53111902e+00
  1.18923781e+01 -1.01916862e+00  4.72096385e+00 -1.82903248e+00
 -1.33122417e+01  4.90773902e+00 -2.68784976e+00  8.66533341e+00
  1.13002218e+01 -2.89860599e+00  4.22958666e+00 -6.26082723e+00
 -3.91619396e+00  1.84178348e+00  4.42659535e+00 -7.05753767e+00
  4.29693724e+00  4.30749942e+00  9.66902580e+00  5.20114742e+00
 -7.05473366e+00 -1.09882620e+01 -5.78497652e+00  3.10269094e-01]
```



In [ ]:

```
#calculating mean and variance of the new dataset as a result of the addition above
```

In [20]:

```
m = np.mean(sum)
print(m)
v = np.var(sum)
print(v)
```

```
0.2464763481519463
48.49678381905323
```

Question 2

Estimate the mean and standard deviation from 1 dimensional data: generate 25,000 samples from a Gaussian distribution with mean 0 and standard deviation 5. Then estimate the mean and standard deviation of this gaussian using elementary numpy commands, i.e., addition, multiplication, division (do not use a command that takes data and returns the mean or standard deviation).

Deliverables: mean, standard deviation, code

In [25]:

```
s3 = np.random.normal(0,5,25000)
#mean
m1 = np.sum(s3)/25000
print(m1)
```

```
-0.004951883883411094
```

In [43]:

```
x = abs(s3 - s3.mean())**2
sum1 = np.sum(x)/(25000-1)
print(sum1)
```

```
25.14410606732171
```

In [44]:

```
sd = np.sqrt(sum1)
```

In [52]:

```
print("mean", m1)
print("standard deviation", sd)
```

```
mean -0.004951883883411094
standard deviation 5.014389899810515
```

In [ ]:

Question 3

For generating samples we can use multivariate_normal method from numpy

From
https://numpy.org/doc/stable/reference/random/generated/numpy.random.multivariate_normal.h

We can pick the two dimensional mean as given to us : (-5,5)

and the diagonal covariance as given to us : (20,0.8) and (0.8,30)

In [1]:
```
import numpy as np
import random

#defining mean and cov as stated above
mean = [-5,5]
cov = [[20,0.8], [0.8,30]]


#generating two arrays using multivariate_normal()
x, y = np.random.multivariate_normal(mean, cov, 10000).T
```

For estimating Mean and covariance matrix we can use sum function on x and y arrays and divide by length of x and y respectively to calculate meanOfX and meanOfY

In [2]:
```
lenX=len(x)
meanOfX=sum(x)/lenX
print(meanOfX)
```

```
-5.0012317483383555
```

In [3]:
```
lenY=len(y)
meanOfY=sum(y)/lenY
print(meanOfY)
```

```
5.075280078772711
```

Following the formula for sample covariance: Σ(Xi – μ)(Yj – ν) / (n-1)

We can loop through x and y and calculate their sum of difference from mean by keeping a counter and then dividing by number of samples (10000) subtracted by 1

In [4]:
```
cov = 0
for i,j in zip(x,y):
    cov += (i-meanOfX)*(j-meanOfY)
cov = cov/(10000-1)
print(cov)
```

```
0.5498363198717441
```

Calculating variance of X and Y:

In [5]:
```
samplesOfX=x
counter = 0
for i in samplesOfX:
    counter = counter + ((i - meanOfX)**2)
varX = (counter / (10000-1))
print(varX)
```

```
19.761022379267047
```

In [6]:
```
samplesOfY=y
counter = 0
for i in samplesOfY:
    counter = counter + ((i - meanOfY)**2)
varY = (counter / (10000-1))
print(varY)
```

```
29.87131886868815
```

Creating the Covariance Matrix:

In [7]:
```
cov=[[cov,varY],[varX,cov]]
print(cov)
```

```
[[0.5498363198717441, 29.87131886868815], [19.761022379267047, 0.5498363198717
441]]
```

Printing the Mean and Covariance Matrix

In [8]:
```
mean2d=[meanOfX,meanOfY]

print("Here is the Mean of bivariate data:",mean2d)
print("Here is the Covariance Matrix:")
cov
```

```
Here is the Mean of bivariate data: [-5.0012317483383555, 5.075280078772711]
Here is the Covariance Matrix:
```

Out[8]:
```
[[0.5498363198717441, 29.87131886868815],
 [19.761022379267047, 0.5498363198717441]]
```

In [ ]:
```
```

Question 4

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df=pd.read_csv('PatientData.csv', sep = ',')
x = 1
for col in df.columns:
  name = 'Column '+str(x)
  df = df.rename(columns={col:name})
  x = x+1

df[df.columns[13:]]
df['Column 1'].dtype.name
```

Out[3]:
```
'int64'
```

Answers to Question 4 part a and c

In [4]:
```
#(a) How many patients and how many features are there?

print("Part a: number of patients: ",df.shape[0]+1) #subtracting 1 because last
print("Part a: number of features: ",df.shape[1]-1) #adding 1 because index sta


#(c) Are there missing values? Replace them with the average of the correspondi
df=df.replace("?",np.nan)
print("Part c: Are there any NaN values?(True if ? were replaced with NaN): ",c

print("part c: replace NaN with average of column: ")
df.fillna(df.mean())
```

```
Part a: number of patients:  452
Part a: number of features:  279
Part c: Are there any NaN values?(True if ? were replaced with NaN):  True
part c: replace NaN with average of column:
```

```
/var/folders/xk/szxzvgq138ndv5c88q9xfh1m0000gn/T/ipykernel_23145/2634372972.p
y:12: FutureWarning: Dropping of nuisance columns in DataFrame reductions (wit
h 'numeric_only=None') is deprecated; in a future version this will raise Type
Error.  Select only valid columns before calling the reduction.
  df.fillna(df.mean())
```

Out[4]:

	Column 1	Column 2	Column 3	Column 4	Column 5	Column 6	Column 7	Column 8	Column 9	Column 10
0	56	1	165	64	81	174	401	149	39	25
1	54	0	172	95	138	163	386	185	102	96
2	55	0	175	94	100	202	380	179	143	28
3	75	0	190	80	88	181	360	177	103	-16
4	13	0	169	51	100	167	321	174	91	107
...	...	...	...	...	...	...	...	...	...	...
446	53	1	160	70	80	199	382	154	117	-37
447	37	0	190	85	100	137	361	201	73	86
448	36	0	166	68	108	176	365	194	116	-85
449	32	1	155	55	93	106	386	218	63	54
450	78	1	160	70	79	127	364	138	78	28

451 rows × 280 columns

Answer to Question 4 part b

To comment on what the first 4 features are we can plot them and understand the values given the context of the dataset. We know that Column 280 is the medical condition of a patient. When we plot Column 1 we can see that the Max Values are not more than 100 and do not go below 0; we can assume that this is the age of all patients column in the dataset.

Column 2 can be plotted on a pie chart as we can see there are only two values in the column : 0 and 1. This could mean that this column represents the gender of the patient in a binary format.

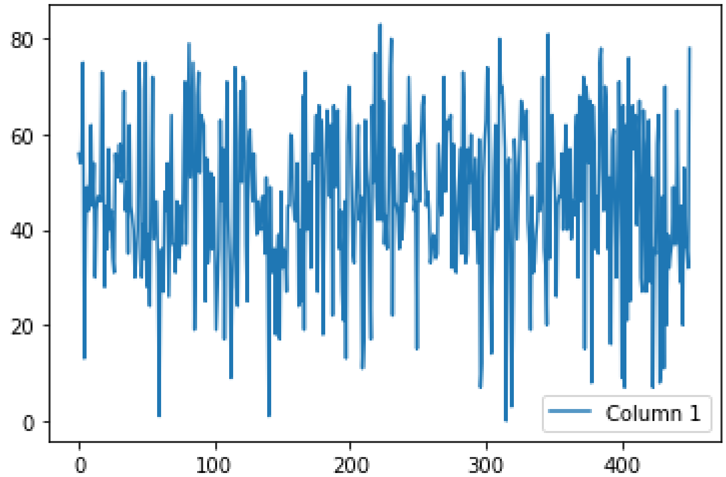
Column 3, 4 are difficult to judge but based on the plots we can assume that Column 3 and Column 4 represent heart and the heart beats per minute for each patient as the values (barring a few outliers) are consistent with the average heart rates.

In [5]:

```
#(b) What is the meaning of the first 4 features? See if you can understand wha

df.plot(y='Column 1')
```

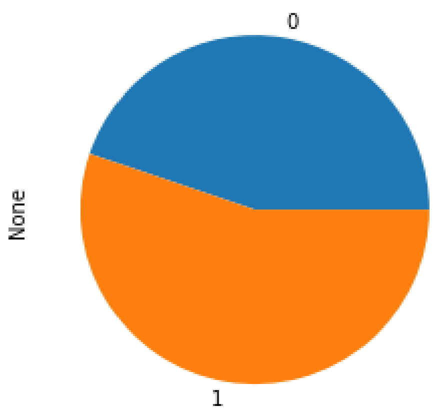
Out[5]: <AxesSubplot:>



In [6]:
```
#(b) What is the meaning of the first 4 features? See if you can understand wha

df.groupby('Column 2').size().plot(kind='pie', subplots=True)
```

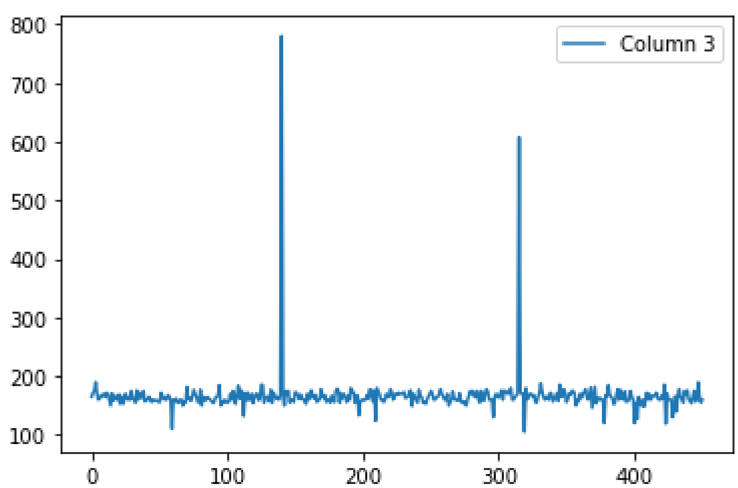
Out[6]: array([<AxesSubplot:ylabel='None'>], dtype=object)

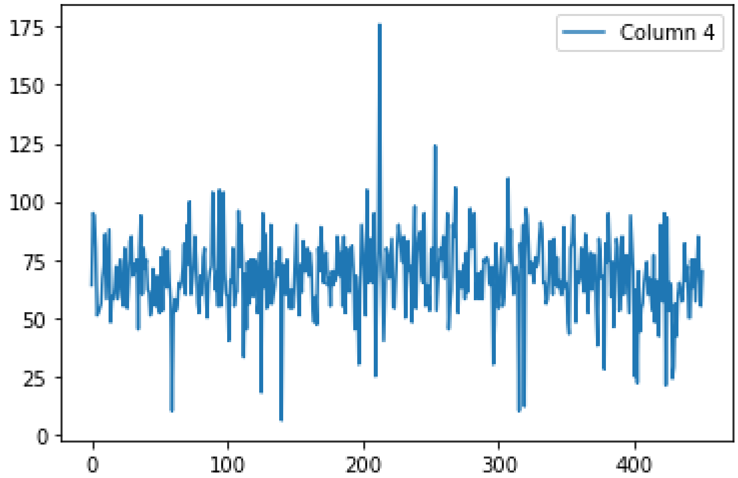


In [7]:
```
#(b) What is the meaning of the first 4 features? See if you can understand wha

df.plot(y='Column 3')
df.plot(y='Column 4')
```

Out[7]: <AxesSubplot:>





Answer to Question 4 part d

Statistically the measure of which value affects the target the most is the Correlation Coefficient. This can be calculated in dataframes using the corr() method. The correlation value is in the range of 1 through -1 and the sign of the value descirbes a positive or negative correlation. For exmaple in the patient dataset Column 163 has a negative value meaning if Column 163 value goes up then the affect on Column 280 (target) will make this value go down. Values above 0.7 and lower than -0.7 are considered to represent a high coreelation and values between 0.3 and -0.3 are considered to be negligible coreelation with values closer (or equal) to 0 representing no correlation.

in this dataset we can assess the highest and lowest correlation values numerical values to judge which feature has the most affect on Column 280. We can say that Column 91, 93 and 5 have the most affect on COlumn 280 with correlation values more than 0.3.

In [8]:
```
#(d) How could you test which features strongly influence the patient conditio

print("Hightest Correlation Values Negative and positive\n",df.corr().sort_valu

corrVal=df.corr()

leastCorr=[]

for item in corrVal['Column 280']:
    if item>=-0.3 and item<=0.3:
        leastCorr.append(item)
print("Lowest Correlation Values Negative and positive\n", corrVal['Column 280'
```

```
Hightest Correlation Values Negative and positive
 Column 280    1.000000
Column 91     0.369935
Column 5      0.323919
Column 93     0.316655
Column 103    0.283321
                ...
Column 271   -0.165585
Column 169   -0.173591
Column 2     -0.176193
Column 243   -0.189687
Column 163   -0.197783
Name: Column 280, Length: 258, dtype: float64
Lowest Correlation Values Negative and positive
 Column 1     -0.096395
Column 2     -0.176193
Column 3      0.005325
Column 4     -0.091773
Column 6     -0.101887
                ...
Column 274   -0.036863
Column 276   -0.088937
Column 277   -0.033325
Column 278    0.002868
Column 279   -0.011539
Name: Column 280, Length: 254, dtype: float64
```

In [ ]:

Consider the vectors v1 = [1, 1, 1] and v2 = [1, 0, 0]. These two vectors define a 2-dimensional subspace of R3. Project the points P 1 = [3, 3, 3], P 2 = [1, 2, 3], P 3 = [0, 0, 1] on this subspace. Write down the coordinates of the three projected points.

In [1]:
```
import numpy as np
from numpy.linalg import inv

v1 = np.array([1,1,1])
v2 = np.array([1,0,0])

x = np.array([[1, 1],[1, 0],[1,0]])
xt = x.transpose()
```

In [2]:
```
xt
```

Out[2]:
```
array([[1, 1, 1],
       [1, 0, 0]])
```

In [3]:
```
xtx = np.dot(xt, x)
```

In [4]:
```
xtx
```

Out[4]:
```
array([[3, 1],
       [1, 1]])
```

In [5]:
```
xtx_inv = inv(xtx)
```

In [6]:
```
xtx_inv
```

Out[6]:
```
array([[ 0.5, -0.5],
       [-0.5,  1.5]])
```

In [7]:
```
innerproduct = np.matmul(xtx_inv, xt)
```

In [8]:
```
p1 = np.array([3,3,3])
bp1 = np.dot(innerproduct, p1)
bp1
```

Out[8]:
```
array([ 3.0000000e+00, -4.4408921e-16])
```

In [9]:
```
p2 = np.array([1,2,3])
bp2 = np.dot(innerproduct, p2)
bp2
```

Out[9]:
```
array([ 2.5, -1.5])
```

In [10]:
```
p3 = np.array([0,0,1])
bp3 = np.dot(innerproduct, p3)
bp3
```

Out[10]:
```
array([ 0.5, -0.5])
```

In [11]:
```
p1hat = np.dot(x, bp1)
p1hat
```

Out[11]:
```
array([3., 3., 3.])
```

In [12]:
```
p2hat = np.dot(x, bp2)
p2hat
```

Out[12]:
```
array([1. , 2.5, 2.5])
```

In [13]:
```
p3hat = np.dot(x, bp3)
p3hat
```

Out[13]:
```
array([5.55111512e-17, 5.00000000e-01, 5.00000000e-01])
```

In [ ]:

In [2]:
```
import numpy as np


def ncr(n):
    return (np.math.factorial(100) / (np.math.factorial(n) * np.math.fact

#We have defined the probability function here as nCr*(p^r)*(q^(n-r)). We
def probability(n):
    return (ncr(n) * ((2 / 3) ** n) * ((1 / 3) ** (100 - n)))

#Function to find the final probability
def probless(n):
    if n >= 0:
        return (probability(n) + probability(n - 1))
```

In [3]:
```
probless(50)
```

Out[3]: 0.0003284517931420376

In [ ]:

Assignment-1

1) written Questions.

2 - 2D vectors $v_1 = [1,1,1]$ & $v_2 = [1,0,0]$

Project the following points
$P_1 = [3,3,3]$ $P_2 = [1,2,3]$ and $P_3 = [0,0,1]$

steps :-
1. normalize vector v_1 & v_2
2. Perform inner product

$$\beta^* = \frac{1}{x^T x} \cdot x^T y \text{ - for vectors}$$

$$\beta^* = (x^T x)^{-1} x^T y \text{ for matrix}$$

Let's consider $X = [v_1 \; v_2]$

$$X = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 0 \end{bmatrix} \quad X^T X = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}$$

$$X^T X = \begin{bmatrix} 1\times1+1\times1+1\times1 & 1\times1+1\times0+1\times0 \\ 1\times1+0\times0+0\times1 & 1\times1+0\times0+0\times0 \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 1 & 1 \end{bmatrix}$$

if, $X = \begin{bmatrix} i & j \\ k & h \end{bmatrix}$

$$X^{-1} = \frac{1}{ih - jk} \begin{bmatrix} h & -j \\ -k & i \end{bmatrix}$$

$$\therefore (X^T X)^{-1} = \begin{bmatrix} 3 & 1 \\ 1 & 1 \end{bmatrix}^{-1} = \frac{1}{(3\times1)-(1\times1)} \begin{bmatrix} 1 & -1 \\ -1 & 3 \end{bmatrix}$$

$$= \frac{1}{2} \begin{bmatrix} 1 & -1 \\ -1 & 3 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{3}{2} \end{bmatrix} = \begin{bmatrix} 0.5 & -0.5 \\ -0.5 & 1.5 \end{bmatrix}$$

$$(X^T X)^{-1} \cdot X^T = \begin{bmatrix} 0.5 & -0.5 \\ -0.5 & 1.5 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0.5\times1+(-0.5)\times1 & 0.5+0 & 0.5+0 \\ -0.5+1.5 & -0.5+0 & -0.5+0 \end{bmatrix} = \begin{bmatrix} 0 & 0.5 & 0.5 \\ 1 & -0.5 & -0.5 \end{bmatrix}$$

$$\beta^*_{P_1} = (X^T X)^{-1} X^T P_1 \qquad P_1 = [3,3,3]$$

$$= \begin{bmatrix} 0 & 0.5 & 0.5 \\ 1 & -0.5 & -0.5 \end{bmatrix} \begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix} = \begin{bmatrix} 0+0.5\times3+0.5\times3 \\ 3-0.5\times3-0.5\times3 \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$$

$$\beta^*_{P_2} = (X^T X)^{-1} X^T P_2 \qquad P_2 = [1,2,3]$$

$$= \begin{bmatrix} 0 & 0.5 & 0.5 \\ 1 & -0.5 & -0.5 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 0+0.5\times2+0.5\times3 \\ 1-0.5\times2-0.5\times3 \end{bmatrix} = \begin{bmatrix} 2.5 \\ -1.5 \end{bmatrix}$$

$$\beta^*_{P_3} = (X^T X)^{-1} X^T P_3$$

$$= \begin{bmatrix} 0 & 0.5 & 0.5 \\ 1 & -0.5 & -0.5 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0+0+0.5\times1 \\ 0+0-0.5\times1 \end{bmatrix} = \begin{bmatrix} 0.5 \\ -0.5 \end{bmatrix}$$

projection of P_1 is $\hat{P}_1$

$$\hat{P}_1 = X\beta^*_{P_1} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 3 \\ 0 \end{bmatrix} = \begin{bmatrix} 3+0 \\ 3+0 \\ 3+0 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix}$$

$$\hat{P}_2 = X\beta^*_{P_2} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 2.5 \\ -1.5 \end{bmatrix} = \begin{bmatrix} 2.5-1.5 \\ 2.5+0 \\ 2.5+0 \end{bmatrix} = \begin{bmatrix} 1 \\ 2.5 \\ 2.5 \end{bmatrix}$$

$$\hat{P}_3 = X\beta^*_{P_3} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 0.5-0.5 \\ 0.5+0 \\ 0.5+0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.5 \\ 0.5 \end{bmatrix}$$

$\therefore$ the projection of the following points on vector v_1 & v_2 are

$P_1 = [3,3,3]$ & $\hat{P}_1 = [3,3,3]$

$P_2 = [1,2,3]$ & $\hat{P}_2 = [1 \; 2.5 \; 2.5]$

$P_3 = [0,0,1]$ $\hat{P}_3 = [0 \; 0.5 \; 0.5]$

extra Credit Question.

toss a coin 100 times.: Find Probability of Heads 50 or less

Probability of heads $= 2/3$

Two possibilities while flipping a coin → Heads or tails

For $P(\text{Heads}) = \frac{2}{3}$

$P(\text{Tails}) = 1 - \frac{2}{3} = \frac{1}{3}$

$P(\text{Heads} \leq 50) = P(\text{heads}=50) + P(\text{Heads}=49) + \ldots + P(H=0)$

Use the binomial distribution

here n no. of trials is 100 & $r = 50$; $p = \frac{2}{3}$ & $q = \frac{1}{3}$

$P(R=r) = {}^{n}C_{r}\, p^{r} q^{n-r}$

$P(R=50) = {}^{100}C_{50} \cdot \left(\frac{2}{3}\right)^{50} \cdot \left(\frac{1}{3}\right)^{100-50}$

$P(H=50) = P(R=50) = {}^{100}C_{50} \cdot \left(\frac{2}{3}\right)^{50} \cdot \left(\frac{1}{3}\right)^{50}$

for $P(H=49) = P(R=49) = {}^{100}C_{49} \cdot \left(\frac{2}{3}\right)^{49} \cdot \left(\frac{1}{3}\right)^{51}$

$\therefore P(H \leq 50) = {}^{100}C_{50} \cdot \left(\frac{2}{3}\right)^{50} \cdot \left(\frac{1}{3}\right)^{50} + \ldots\ldots$

$+ {}^{100}C_{0} \cdot \left(\frac{2}{3}\right)^{0} \cdot \left(\frac{1}{3}\right)^{100}$

which is ~~P(H) =~~

$= 0.00032845$

$= 0.032845\%$

* Let's try with Central limit theorem

$\mu = np = 100 \cdot \frac{2}{3} = \frac{200}{3}$ & $\sigma = \sqrt{npq}$

$= \sqrt{100 \times \frac{2}{3} \times \frac{1}{2}} = \frac{10\sqrt{2}}{3}$

$P(0 < x < 50)$

$= P\left(\dfrac{0 - \frac{200}{3}}{\frac{10\sqrt{2}}{3}} < z < \dfrac{50 - \frac{200}{3}}{\frac{10\sqrt{2}}{3}}\right)$

$= P\left(\dfrac{\frac{-197}{\cancel{3}}}{\frac{10\sqrt{2}}{\cancel{3}}} < z < \dfrac{\frac{-\cancel{50}^{\,5}}{\cancel{3}}}{\frac{\cancel{10}\sqrt{2}}{\cancel{3}}}\right)$

$= P\left(\frac{-197}{10\sqrt{2}} < z < \frac{-5}{\sqrt{2}}\right) = P\left(\frac{5}{\sqrt{2}} < z < \frac{197}{10\sqrt{2}}\right)$

$= P\left(0 < z < \frac{197}{10\sqrt{2}}\right) - P\left(0 < z < \frac{5}{\sqrt{2}}\right)$