12. Quiz

Normal Distribution [1]

- 정규분포표

z	.00	.01	.02	.03	.04	.05	.06	.07	.08	.09
0.0	.0000	.0040	.0080	.0120	.0160	.0199	.0239	.0279	.0319	.0359
0.1	.0398	.0438	.0478	.0517	.0557	.0596	.0636	.0675	.0714	.0753
0.2	.0793	.0832	.0871	.0910	.0948	.0987	.1026	.1064	.1103	.1141
0.3	.1179	.1217	.1255	.1293	.1331	.1368	.1406	.1443	.1480	.1517
0.4	.1554	.1591	.1628	.1664	.1700	.1736	.1772	.1808	.1844	.1879
0.5	.1915	.1950	.1985	.2019	.2054	.2088	.2123	.2157	.2190	.2224
0.6	.2257	.2291	.2324	.2357	.2389	.2422	.2454	.2486	.2517	.2549
0.7	.2580	.2611	.2642	.2673	.2704	.2734	.2764	.2794	.2823	.2852
0.8	.2881	.2910	.2939	.2967	.2995	.3023	.3051	.3078	.3106	.3133
0.9	.3159	.3186	.3212	.3238	.3264	.3289	.3315	.3340	.3365	.3389
1.0	.3413	.3438	.3461	.3485	.3508	.3531	.3554	.3577	.3599	.3621
1.1	.3643	.3665	.3686	.3708	.3729	.3749	.3770	.3790	.3810	.3830
1.2	.3849	.3869	.3888	.3907	.3925	.3944	.3962	.3980	.3997	.4015
1.3	.4032	.4049	.4066	.4082	.4099	.4115	.4131	.4147	.4162	.4177
1.4	.4192	.4207	.4222	.4236	.4251	.4265	.4279	.4292	.4306	.4319
1.5	.4332	.4345	.4357	.4370	.4382	.4394	.4406	.4418	.4429	.4441
1.6	.4452	.4463	.4474	.4484	.4495	.4505	.4515	.4525	.4535	.4545
1.7	.4554	.4564	.4573	.4582	.4591	.4599	.4608	.4616	.4625	.4633
1.8	.4641	.4649	.4656	.4664	.4671	.4678	.4686	.4693	.4699	.4706
1.9	.4713	.4719	.4726	.4732	.4738	.4744	.4750	.4756	.4761	.4767
2.0	.4772	.4778	.4783	.4788	.4793	.4798	.4803	.4808	.4812	.4817
2.1	.4821	.4826	.4830	.4834	.4838	.4842	.4846	.4850	.4854	.4857
2.2	.4861	.4864	.4868	.4871	.4875	.4878	.4881	.4884	.4887	.4890
2.3	.4893	.4896	.4898	.4901	.4904	.4906	.4909	.4911	.4913	.4916
2.4	.4918	.4920	.4922	.4925	.4927	.4929	.4931	.4932	.4934	.4936
2.5	.4938	.4940	.4941	.4943	.4945	.4946	.4948	.4949	.4951	.4952
2.6	.4953	.4955	.4956	.4957	.4959	.4960	.4961	.4962	.4963	.4964
2.7	.4965	.4966	.4967	.4968	.4969	.4970	.4971	.4972	.4973	.4974
2.8	.4974	.4975	.4976	.4977	.4977	.4978	.4979	.4979	.4980	.4981
2.9	.4981	.4982	.4982	.4983	.4984	.4984	.4985	.4985	.4986	.4986
3.0	.4987	.4987	.4987	.4988	.4988	.4989	.4989	.4989	.4990	.4990

Normal Distribution [1]

- Example
  - 특정 용기에 들어있는 식품의 무게는 평균 500kg
    표준편차 5kg의 정규분포를 따른다. 하나의 용기를 임의로 뽑았을 때의 다음 아래의 확률을 구하라
  - 1) 510kg 이상일 확률
  - 2) 498kg 이하의 확률
  - 3) 491kg과 498kg 사이의 확률
  - 4) 492kg과 514kg 사이의 확률

Implementations [1]

- **Normal Distribution Declarations and Prototypes**

```
#ifndef BOOL
typedef int BOOL;
#define FALSE 0
#define TRUE !FALSE
#endif

#ifndef DATATYPE
#define DATATYPE
#define SCANFORMAT "%f"
#define PRNFORMAT "%f"
typedef float EType;
#endif

#ifndef DATASET
#define DATASET
typedef struct
{
        int rows;
        int cols;
        int length;
        EType *elem;

} DataSet;
#endif
```

Implementations [2]

- **Normal Distribution Declarations and Prototypes**

```
#ifndef NDIST
#define NDIST(mat, row, col) (mat->elem[row * (mat->cols) + col])
#endif

static DataSet *CreateNormDist(const int rowsDataSet, const int colsDataSet);
static void DestroyDataSet(DataSet *dataSet);
static void PrintDataSet(DataSet *dataSet);
static float FindNormDist(const DataSet *dataSet, const float value);
static float Standardize(const float mean, const float sd, const float v);
static float ComputeNormDistGt(const DataSet *dataSet, const float mean, const float sd, const
float v);
static float ComputeNormDistGtLt(const DataSet *dataSet, const float mean, const float sd, const
float v1, const float v2);
static float ComputeNormDistLt(const DataSet *dataSet, const float mean, const float sd, const
float v1);
```

Implementations [2]

- Main

```
FILE *inFile = NULL;
DataSet *setData = NULL;
EType *begin = NULL;
int rows = 0, cols = 0;
inFile = fopen(PATH_FILE, "r");
if (!inFile) abort();
fscanf(inFile, "%d%d", &rows, &cols);
setData = CreateNormDist(rows, cols);
begin = setData->elem;
while (fscanf(inFile, SCANFORMAT, begin++) != EOF);
fclose(inFile);
```

Implementations [2]

- Main

```
PrintDataSet(setData);
printf("₩n");
float mean = 500.f;
float sd = 5.f;
float prov = ComputeNormDistGt(setData, mean, sd, 510.f);
printf("P(X > 510) = %f₩n", prov);
printf("₩n");
prov = ComputeNormDistLt(setData, mean, sd, 498.f);
printf("P(X < 498) = %f₩n", prov);
printf("₩n");
prov = ComputeNormDistGtLt(setData, mean, sd, 491.f, 498.f);
printf("P(491 < X < 498) = %f₩n", prov);
printf("₩n");
prov = ComputeNormDistGtLt(setData, mean, sd, 492.f, 514.f);
printf("P(492 < X < 514) = %f₩n", prov);
printf("₩n");
DestroyDataSet(setData);
```

Implementations [2]

- Run

```
0.000000 0.004000 0.008000 0.012000 0.016000 0.019900 0.023900 0.027900 0.031900 0.035900
0.039800 0.043800 0.047800 0.051700 0.055700 0.059600 0.063600 0.067500 0.071400 0.075300
0.079300 0.083200 0.087100 0.091000 0.094800 0.098700 0.102600 0.106400 0.110300 0.114100
0.117900 0.121700 0.125500 0.129300 0.133100 0.136800 0.140600 0.144300 0.148000 0.151700
0.155400 0.159100 0.162800 0.166400 0.170000 0.173600 0.177200 0.180800 0.184400 0.187900
0.191500 0.195000 0.198500 0.201900 0.205400 0.208800 0.212300 0.215700 0.219000 0.222400
0.225700 0.229100 0.232400 0.235700 0.238900 0.242200 0.245400 0.248600 0.251700 0.254900
0.258000 0.261100 0.264200 0.267300 0.270400 0.273400 0.276400 0.279400 0.282300 0.285200
0.288100 0.291000 0.293900 0.296700 0.299500 0.302300 0.305100 0.307800 0.310600 0.313300
0.315900 0.318600 0.321200 0.323800 0.326400 0.328900 0.331500 0.334000 0.336500 0.338900
0.341300 0.343800 0.346100 0.348500 0.350800 0.353100 0.355400 0.357700 0.359900 0.362100
0.364300 0.366500 0.368600 0.370800 0.372900 0.374900 0.377000 0.379000 0.381000 0.383000
0.384900 0.386900 0.388800 0.390700 0.392500 0.394400 0.396200 0.398000 0.399700 0.401500
0.403200 0.404900 0.406600 0.408200 0.409900 0.411500 0.413100 0.414700 0.416200 0.417700
0.419200 0.420700 0.422200 0.423600 0.425100 0.426500 0.427900 0.429200 0.430600 0.431900
0.433200 0.434500 0.435700 0.437000 0.438200 0.439400 0.440600 0.441800 0.442900 0.444100
0.445200 0.446300 0.447400 0.448400 0.449500 0.450500 0.451500 0.452500 0.453500 0.454500
0.455400 0.456400 0.457300 0.458200 0.459100 0.459900 0.460800 0.461600 0.462500 0.463300
0.464100 0.464900 0.465600 0.466400 0.467100 0.467800 0.468600 0.469300 0.469900 0.470600
0.471300 0.471900 0.472600 0.473200 0.473800 0.474400 0.475000 0.475600 0.476100 0.476700
0.477200 0.477800 0.478300 0.478800 0.479300 0.479800 0.480300 0.480800 0.481200 0.481700
0.482100 0.482600 0.483000 0.483400 0.483800 0.484200 0.484600 0.485000 0.485400 0.485700
0.486100 0.486400 0.486800 0.487100 0.487500 0.487800 0.488100 0.488400 0.488700 0.489000
0.489300 0.489600 0.489800 0.490100 0.490400 0.490600 0.490900 0.491100 0.491300 0.491600
0.491800 0.492000 0.492200 0.492500 0.492700 0.492900 0.493100 0.493200 0.493400 0.493600
0.493800 0.494000 0.494100 0.494300 0.494500 0.494600 0.494800 0.494900 0.495100 0.495200
0.495300 0.495500 0.495600 0.495700 0.495900 0.496000 0.496100 0.496200 0.496300 0.496400
0.496500 0.496600 0.496700 0.496800 0.496900 0.497000 0.497100 0.497200 0.497300 0.497400
0.497400 0.497500 0.497600 0.497700 0.497700 0.497800 0.497900 0.497900 0.498000 0.498100
0.498100 0.498200 0.498200 0.498300 0.498400 0.498400 0.498500 0.498500 0.498600 0.498600
0.498700 0.498700 0.498700 0.498800 0.498800 0.498900 0.498900 0.498900 0.499000 0.499000

P(X > 510) = 0.022800

P(X < 498) = 0.344600

P(491 < X < 498) = 0.308700

P(492 < X < 514) = 0.942600
```