

Compute CDF for Normal and Bivariate Normal Distributions

back to [Fan's Intro Math for Econ](#), [Matlab Examples](#), or [Dynamic Asset Repositories](#)

CDF for normal random variable through simulation and with NORMCDF function. CDF for bivariate normal random variables through simulation and with NORMCDF function.

- [fs_cholesky_decomposition](#)
- [fs_cholesky_decomposition_d5](#)
- [fs_bivariate_normal](#)

Simulate Normal Distribution Probability with Uniform Draws

Mean score is 0, standard deviation is 1, we want to know what is the chance that children score less than -2, -1, 0, 1, and 2 respectively. We have a solution to the normal CDF cumulative distribution problem, it is:

```
mu = 0;
sigma = 1;
ar_x = [-2,-1,0,1,2];
for x=ar_x
    cdf_x = normcdf(x, mu, sigma);
    disp([strjoin(...
        ["CDF with normcdf", ...
        ['x=' num2str(x)] ...
        ['cdf_x=' num2str(cdf_x)] ...
        ], ";")]);
end
```

```
CDF with normcdf;x=-2;cdf_x=0.02275
CDF with normcdf;x=-1;cdf_x=0.15866
CDF with normcdf;x=0;cdf_x=0.5
CDF with normcdf;x=1;cdf_x=0.84134
CDF with normcdf;x=2;cdf_x=0.97725
```

We can also approximate the probabilities above, by drawing many points from a uniform:

1. Draw from uniform distribution 0 to 1, N times.
2. Invert these using `invnorm`. This means our uniform draws are now effectively drawn from the normal distribution.
3. Check if each draw inverted is below the x threshold or above, count fractions.

We should get very similar results as in the example above (especially if N is large)

```
% set seed
rng(123);
% generate random numbers
N = 10000;
ar_unif_draws = rand(1,N);
% invert
ar_normal_draws = norminv(ar_unif_draws);
% loop over different x values
```

```

for x=ar_x
    % index if draws below x
    ar_it_idx_below_x = (ar_normal_draws < x);
    fl_frac_below_x = (sum(ar_it_idx_below_x))/N;
    disp([strjoin(...
        ["CDF with normcdf", ...
        ['x=' num2str(x)] ...
        ['fl_frac_below_x=' num2str(fl_frac_below_x)] ...
        ], ";"))]);
end

```

```

CDF with normcdf;x=-2;fl_frac_below_x=0.023
CDF with normcdf;x=-1;fl_frac_below_x=0.1612
CDF with normcdf;x=0;fl_frac_below_x=0.4965
CDF with normcdf;x=1;fl_frac_below_x=0.847
CDF with normcdf;x=2;fl_frac_below_x=0.9789

```

Simulate Bivariate-Normal Distribution Probability with Uniform Draws

There are two tests now, a math test and an English test. Student test scores are correlated with correlation 0.5 from the two tests, mean and standard deviations are 0 and 1 for both tests. What is the chance that a student scores below -2 and -2 for both, below -2 and 0 for math and English, below 2 and 1 for math and English, etc?

```

% timer
tm_start_mvncdf = tic;
% mean, and varcov
ar_mu = [0,0];
mt_varcov = [1,0.5;0.5,1];
ar_x = linspace(-3,3,101);
% initialize storage
mt_prob_math_eng = zeros([length(ar_x), length(ar_x)]);
% loop over math and english score thresholds
it_math = 0;
for math=ar_x
    it_math = it_math + 1;
    it_eng = 0;
    for eng=ar_x
        it_eng = it_eng + 1;
        % points below which to compute probability
        ar_scores = [math, eng];
        % volumn of a mountain to the southwest of north-south and east-west cuts
        cdf_x = mvncdf(ar_scores, ar_mu, mt_varcov);
        mt_prob_math_eng(it_math, it_eng) = cdf_x;
    end
end
% end timer
tm_end_mvncdf = toc(tm_start_mvncdf);
st_complete = strjoin(...
    ["MVNCDF Completed CDF computes", ...
    ['number of points=' num2str(numel(mt_prob_math_eng))] ...
    ['time=' num2str(tm_end_mvncdf)] ...
    ], ";");
disp(st_complete);

```

```
% show results
tb_prob_math_eng = array2table(round(mt_prob_math_eng, 4));
cl_col_names_a = strcat('english <=', string(ar_x'));
cl_row_names_a = strcat('math <=', string(ar_x'));
tb_prob_math_eng.Properties.VariableNames = cl_col_names_a;
tb_prob_math_eng.Properties.RowNames = cl_row_names_a;
% subsetting function
% https://fanwangecon.github.io/M4Econ/amto/array/htmlpdfm/fs\_slicing.html#19\_Given\_Array\_of\_si
f_subset = @(it_subset_n, it_ar_n) unique(round(((0:1:(it_subset_n-1))/(it_subset_n-1))*(it_ar_n-1)));
disp(tb_prob_math_eng(f_subset(7, length(ar_x)), f_subset(7, length(ar_x))));
```

	english <=-3	english <=-1.98	english <=-1.02	english <=0	english <=1.02	english <=3
math <=-3	0.0001	0.0005	0.001	0.0013	0.0013	0.0013
math <=-1.98	0.0005	0.0043	0.0136	0.0217	0.0237	0.0237
math <=-1.02	0.001	0.0136	0.0598	0.1239	0.1505	0.1537
math <=0	0.0013	0.0217	0.1239	0.3333	0.4701	0.4978
math <=1.02	0.0013	0.0237	0.1505	0.4701	0.7521	0.8359
math <=1.98	0.0013	0.0238	0.1537	0.4978	0.8359	0.9566
math <=3	0.0013	0.0239	0.1539	0.5	0.8458	0.975

We can also approximate the probabilities above, by drawing many points from two iid uniforms, and translating them to correlated normal using cholesky decomposition:

1. Draw from two random uniform distribution 0 to 1, N times each
2. Invert these using invnorm for both iid vectors from uniform draws to normal draws
3. Cholesky decompose and multiplication

This method below is faster than the method above when the number of points where we have to evaluate probabilities is large.

Generate randomly drawn scores:

```
% timer
tm_start_chol = tic;
% Draws uniform and invert to standard normal draws
N = 10000;
rng(123);
ar_unif_draws = rand(1,N*2);
ar_normal_draws = norminv(ar_unif_draws);
ar_draws_eta_1 = ar_normal_draws(1:N);
ar_draws_eta_2 = ar_normal_draws((N+1):N*2);

% Cholesley decompose the variance covariance matrix
mt_varcov_chol = chol(mt_varcov, 'lower');

% Generate correlated random normals
mt_scores_chol = ar_mu' + mt_varcov_chol*([ar_draws_eta_1; ar_draws_eta_2]);
ar_math_scores = mt_scores_chol(1,:);
ar_eng_scores = mt_scores_chol(2,:);
```

Approximate probabilities from randomly drawn scores:

```
% initialize storage
mt_prob_math_eng_approx = zeros([length(ar_x), length(ar_x)]);
% loop over math and english score thresholds
it_math = 0;
for math=ar_x
    it_math = it_math + 1;
    it_eng = 0;
    for eng=ar_x
        it_eng = it_eng + 1;

        % points below which to compute probability
        % index if draws below x
        ar_it_idx_below_x_math = (ar_math_scores < math);
        ar_it_idx_below_x_eng = (ar_eng_scores < eng);
        ar_it_idx_below_x_joint = ar_it_idx_below_x_math.*ar_it_idx_below_x_eng;
        fl_frac_below_x_approx = (sum(ar_it_idx_below_x_joint))/N;

        % volumn of a mountain to the southwest of north-south and east-west cuts
        mt_prob_math_eng_approx(it_math, it_eng) = fl_frac_below_x_approx;
    end
end
% end timer
tm_end_chol = toc(tm_start_chol);
st_complete = strjoin(...
    ["UNIF+CHOL Completed CDF computes", ...
    ['number of points=' num2str(numel(mt_prob_math_eng_approx))] ...
    ['time=' num2str(tm_end_chol)] ...
    ], ";");
disp(st_complete);
```

UNIF+CHOL Completed CDF computes;number of points=10201;time=0.28661

```
% show results
tb_prob_math_eng_approx = array2table(round(mt_prob_math_eng_approx, 4));
cl_col_names_a = strcat('english <=', string(ar_x));
cl_row_names_a = strcat('math <=', string(ar_x));
tb_prob_math_eng_approx.Properties.VariableNames = cl_col_names_a;
tb_prob_math_eng_approx.Properties.RowNames = cl_row_names_a;
disp(tb_prob_math_eng_approx(f_subset(7, length(ar_x)), f_subset(7, length(ar_x))));
```

	english <=-3	english <=-1.98	english <=-1.02	english <=0	english <=1.02	english <=1.98
math <=-3	0.0001	0.0005	0.001	0.0016	0.0016	0.0016
math <=-1.98	0.0003	0.004	0.0132	0.0218	0.0237	0.0237
math <=-1.02	0.0008	0.0131	0.061	0.1272	0.1529	0.1529
math <=0	0.0009	0.0202	0.1236	0.334	0.4661	0.494
math <=1.02	0.0009	0.0215	0.1493	0.4724	0.754	0.841
math <=1.98	0.0009	0.0217	0.1526	0.4989	0.8344	0.959
math <=3	0.0009	0.0217	0.1526	0.5007	0.8425	0.976