# Generating Pseudo-random Numbers in MATLAB

# or,

# How to use `rand` and `randn`

Gerald W. Recktenwald
Department of Mechanical Engineering
Portland State University
`gerry@pdx.edu`

## Why Use Statistics in ME 350?

- It's important for engineers to understand statistics
- Some numerical methods rely on statistical properties and methods
  - ▷ Least squares curve fitting
  - ▷ Monte-Carlo methods
  - ▷ Machine learning
  - ▷ Signal processing
- Using random inputs can be useful to
  - ▷ Demonstrate algorithms, i.e., as a teaching device
  - ▷ To generate test inputs, i.e., don't rely on "simple" or "nice" values for testing

## `rand` and `randn`

`A = rand(m,n)`

- `A` is a matrix with `m` rows and `n` columns.
- Elements of `A` are approximately a uniform random distribution on $[0, 1]$.

`B = randn(m,n)`

- `B` is a matrix with `m` rows and `n` columns.
- Elements of `B` are approximately a normal distribution with mean of 0 and standard deviation of 1.

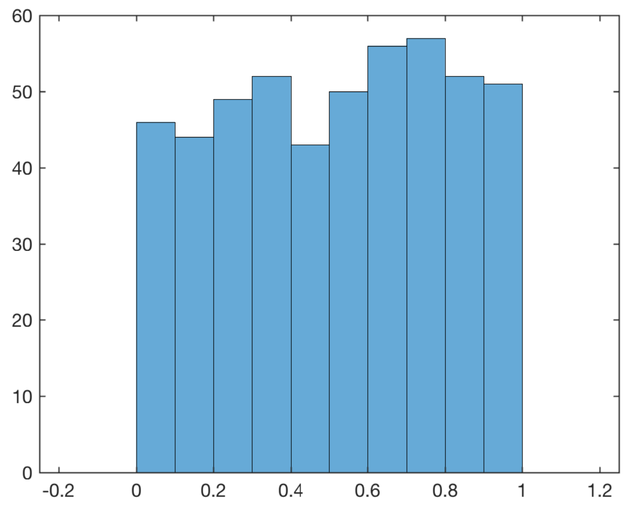
  The probability density for the (ideal) normal distribution is

$$p(x) = \frac{1}{\sqrt{2\sigma^2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

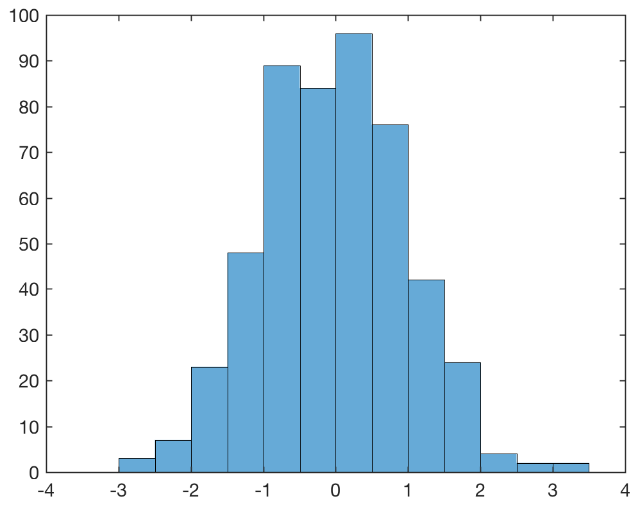
  where $\mu$ is the mean and $\sigma$ is the standard deviation.

## Examples of `rand` and `randn`

```
x = rand(500,1);
histogram(x);
```



```
x = randn(500,1);
histogram(x);
```



## How to Make a Histogram

Given a vector $x$

1. Find minimum and maximum: $x_{\min}$ and $x_{\max}$
2. Divide the range of $x$ into $n$ bins

$$\Delta x = \frac{x_{\max} - x_{\min}}{n}$$

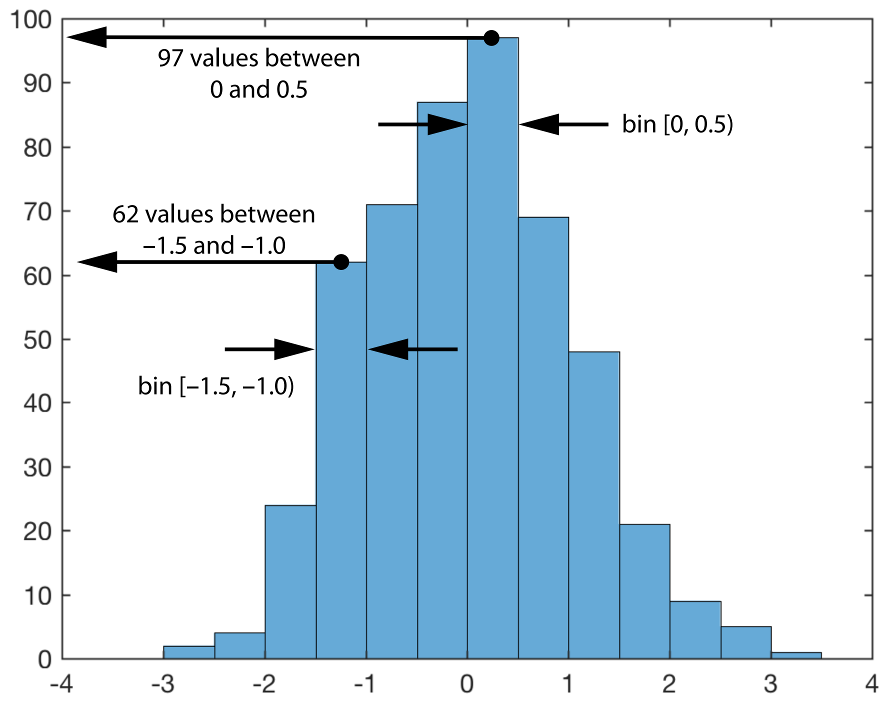
3. Count the number of $x$ values in each bin
4. Make a bar chart showing the number of counts for each bin

Histograms tell us how the values in $x$ are distributed over the range of $x$.

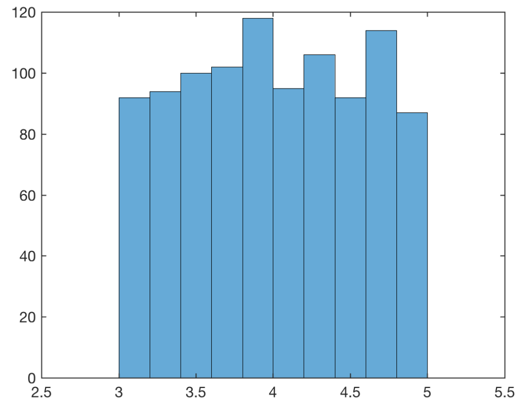
Histograms show us *frequency distributions*, not the shape of $y = f(x)$.

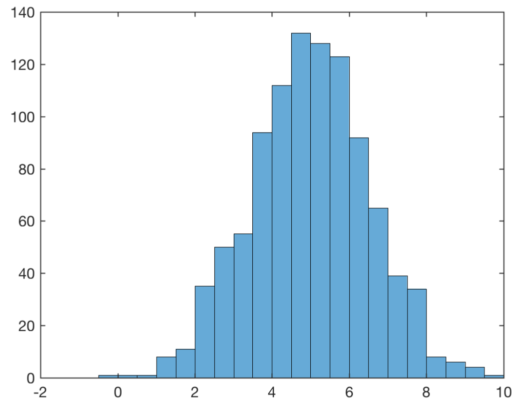
## Sample Histogram

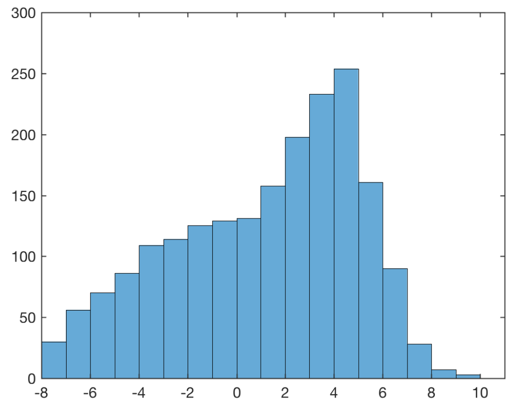
| Left edge | Right edge | Counts |
|---|---|---|
| -3.0 | -2.5 | 2 |
| -2.5 | -2.0 | 4 |
| -2.0 | -1.5 | 24 |
| -1.5 | -1.0 | 62 |
| -1.0 | -0.5 | 71 |
| -0.5 | 0.0 | 87 |
| 0.0 | 0.5 | 97 |
| 0.5 | 1.0 | 69 |
| 1.0 | 1.5 | 48 |
| 1.5 | 2.0 | 21 |
| 2.0 | 2.5 | 9 |
| 2.5 | 3.0 | 5 |
| 3.0 | 3.5 | 1 |

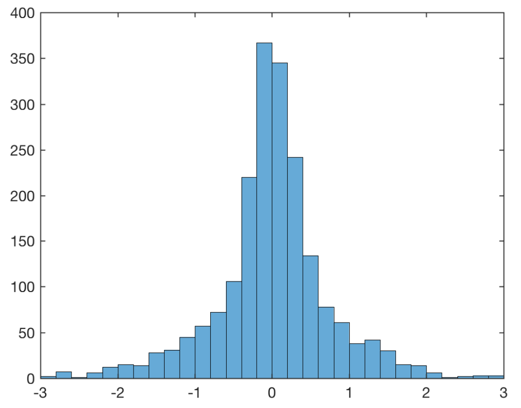


## More Histograms: Shape tells us something





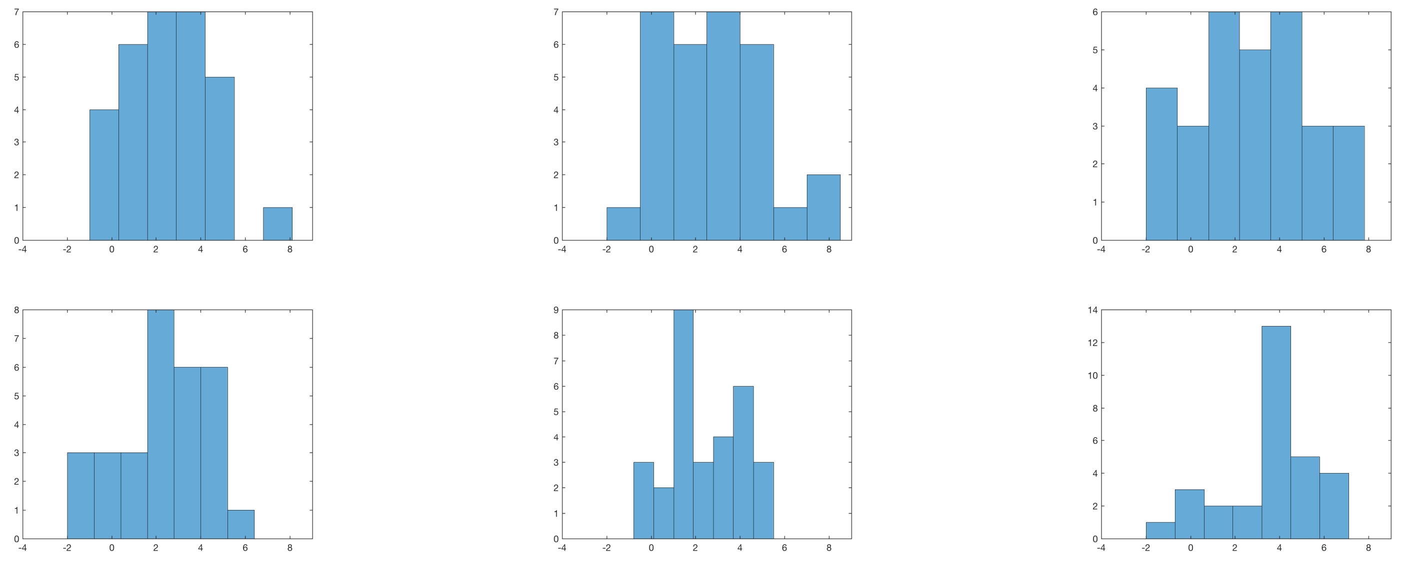




## `randn` **Samples are not Perfect**

Especially for small sample sizes, the distributions from `randn` will not look like a classical normal distribution. That's OK. The output should be random, not uniformly normal.

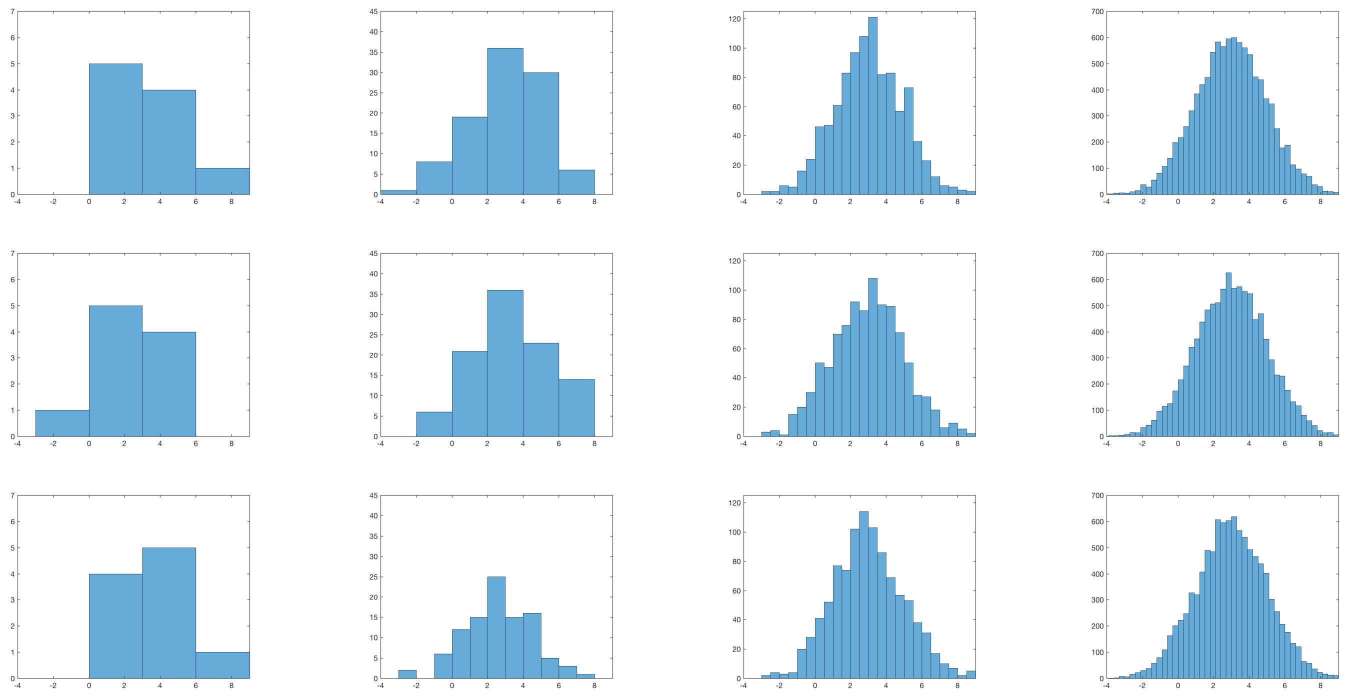
Here are six consecutive distributions created with 30 samples from `randn`



## `randn` distributions look more normal as $n$ increases

Results for three runs of `randn(n,1)` at each `n`.

$n = 10$ $n = 100$ $n = 1000$ $n = 10000$



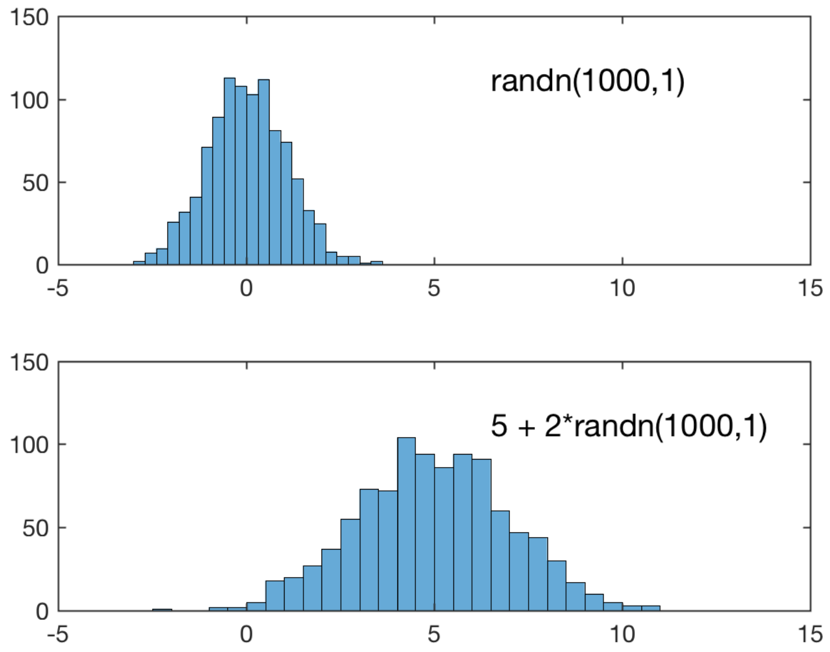
## Easily change center ($\bar{x}$) and spread ($\sigma$) of `randn` distributions

`randn` produces pseudo-random distributions with a mean of 0 and a standard deviation of 1.

We can modify the output from `randn` to create distributions with any $\bar{x}$ and $\sigma$ we want.

Use this:

```
x = xbar + sig*randn(n,1)
```



`xbar` is the *desired* mean and `sig` is the *desired* standard deviation.

Since the output from `randn` is not a perfect normal distribution, `mean(x)` $\neq$ `xbar` and `std(x)` $\neq$ `sig`.

## `rand` and `randn` Repeat the "Random" Numbers

The seed values use to initialize the pseudo-random number generators are reused at the start of each MATLAB session, unless you explicitly tell MATLAB not to start at the same values.

The sequence of random values generated by `rand` and `randn` repeats *unless* you issue the

```
rng('shuffle')
```

command after startup.