Table of Contents

ECE537: Lab 1 Report

*It is recommended to access this report by opening the `html` file on the browser or by clicking **here**.*

In this lab, we are supposed to create custom univariate and joint distributions, sample from these distributions to test for convergence cirtieria, and also test for independence of random variables. In the first part, we will be creating a continuous random variable, and in the second part we will be creating discrete independent and joint random variables. Throughout this lab, the **Distributions.jl** package in Julia has been utilized to be able to use the probabllity constructs in code.

```
using Random , Distributions , StatsBase , StatsPlots , LinearAlgebra
, DataFrames , LaTeXStrings , PlutoUI
```

```
import Random : AbstractRNG
```

```
import Distributions : ContinuousUnivariateDistribution, @check_args,
@distr_support
```

1. Simulating Univariate Random Variables

We define a custom continuous probability distribution `ZDist`(a, b, c) with the following cumulative density function (cdf):

$$F_Z(z) = \begin{cases} 0, & z < a \\ \frac{z-a}{2(b-a)}, & a \leq z < b \\ \frac{1}{2} + \frac{z-b}{2(c-b)}, & b \leq z < c \\ 1, & z \geq c \end{cases}$$

which has mean, $\mathrm{E}[Z] = \frac{a+2b+c}{4}$, and variance, $\mathrm{VAR}[Z] = \frac{4b(b-a-c)+5a^2+5c^2-6ac}{48}$.

We will proceed with defining this distibution in code as a sampleable object (distribution `struct`).

```
struct ZDist{T<:Real} <: ContinuousUnivariateDistribution
    a::T
    b::T
    c::T

    ZDist{T}(a::T, b::T, c::T) where {T <: Real} = new{T}(a, b, c)

    function ZDist(a::T, b::T, c::T; check_args=true) where {T <: Real}
        check_args && @check_args(ZDist, a <= b <= c)
        return new{T}(a, b, c)
    end

    ZDist(a::Real, b::Real, c::Real) = ZDist(promote(a, b, c)...)
    ZDist(a::Integer, b::Integer, c::Integer) = ZDist(float(a), float(b),
float(c))
end
```

```
var (generic function with 1 method)
```

```
begin
    # helper functions
    @distr_support ZDist d.a d.c
    params(d::ZDist) = (d.a, d.b, d.c)
    mean(d::ZDist) = (d.a + 2d.b + d.c)/4
    var(d::ZDist) = (4d.b*(d.b - d.a - d.c) + 5d.a^2 + 5d.c^2 - 6d.a*d.c)/48
end
```

Now that we have defined such a distribution in code, we need to make sure that the sampling algorithm matches the probability distribution function (pdf), which is:

$$f_Z(z) = \begin{cases} 0, & z < a \\ \frac{1}{2(b-a)}, & a \leq z < b \\ \frac{1}{2(c-b)}, & b \leq z < c \\ 0, & z > c \end{cases}$$

Notice that this pdf is made up of two uniform pdfs with equal cumulative probability (of half). Therefore, given $U \sim \mathcal{U}(0, 1)$ and $\widetilde{U} \sim \mathcal{U}(\{0, 1\})$ our sampling algorithm is the following transformation:

$$Z = (1 - \widetilde{U}) \cdot (a + (b - a) \cdot U) + \widetilde{U} \cdot (b + (c - b) \cdot U),$$

where $\widetilde{U}$ acts as a uniform "selector" across the two "pieces" in the distribution function.

```
function Base.rand(rng::AbstractRNG, d::ZDist)
    (a, b, c) = params(d)
    u = rand(rng)
    ũ = rand(0:1)
    u₁ = a + (b - a) * u
    u₂ = b + (c - b) * u
    return (1-ũ) * u₁ + ũ * u₂
end
```

In this lab, we will consider the random variable $Z \sim \texttt{ZDist}(0, 1, 3)$.

```
Z = ZDist(0, 1, 3);
```

1.1 Numerical Simulation

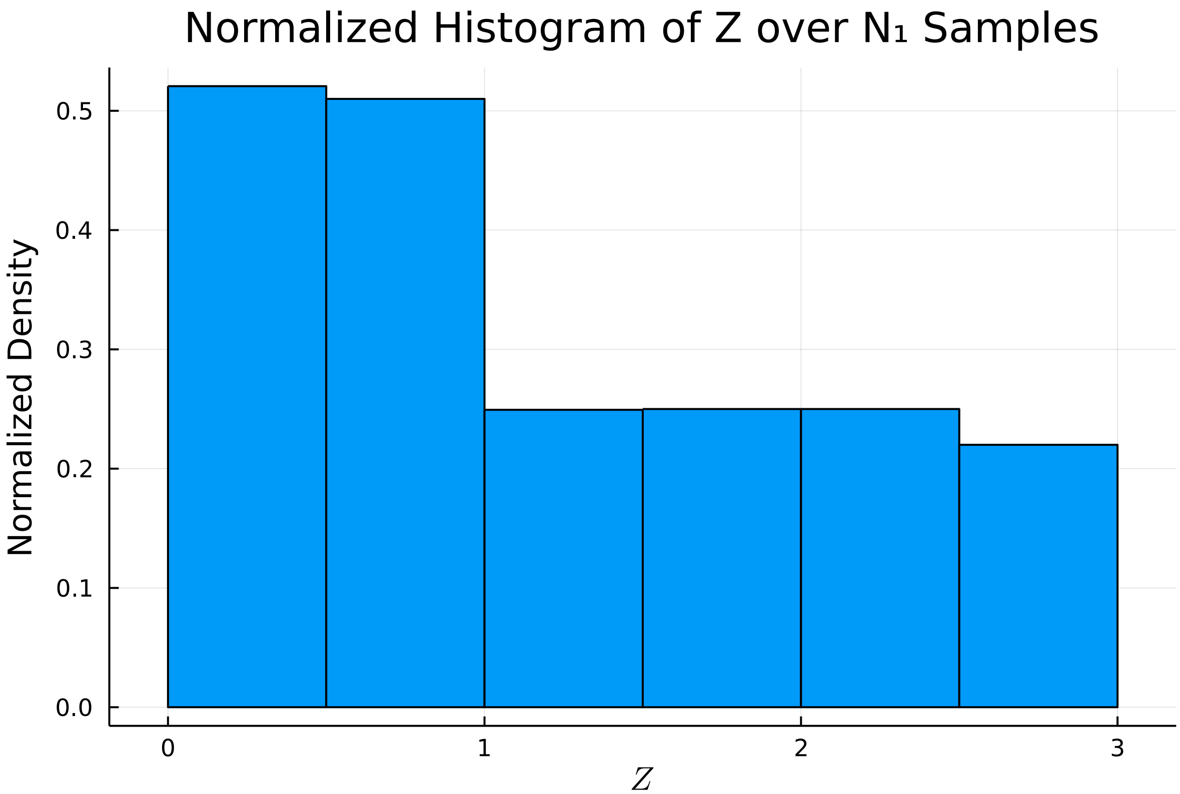
We can now sample the distribution many times and check for convergence of key statistics, like the mean and variance, empirically.

N_1 = 3000

```
Zsamples =
  [0.569545, 2.1984, 0.593724, 2.65274, 2.22261, 1.18231, 0.31918, 1.08235, 0.105953, 1.0
```

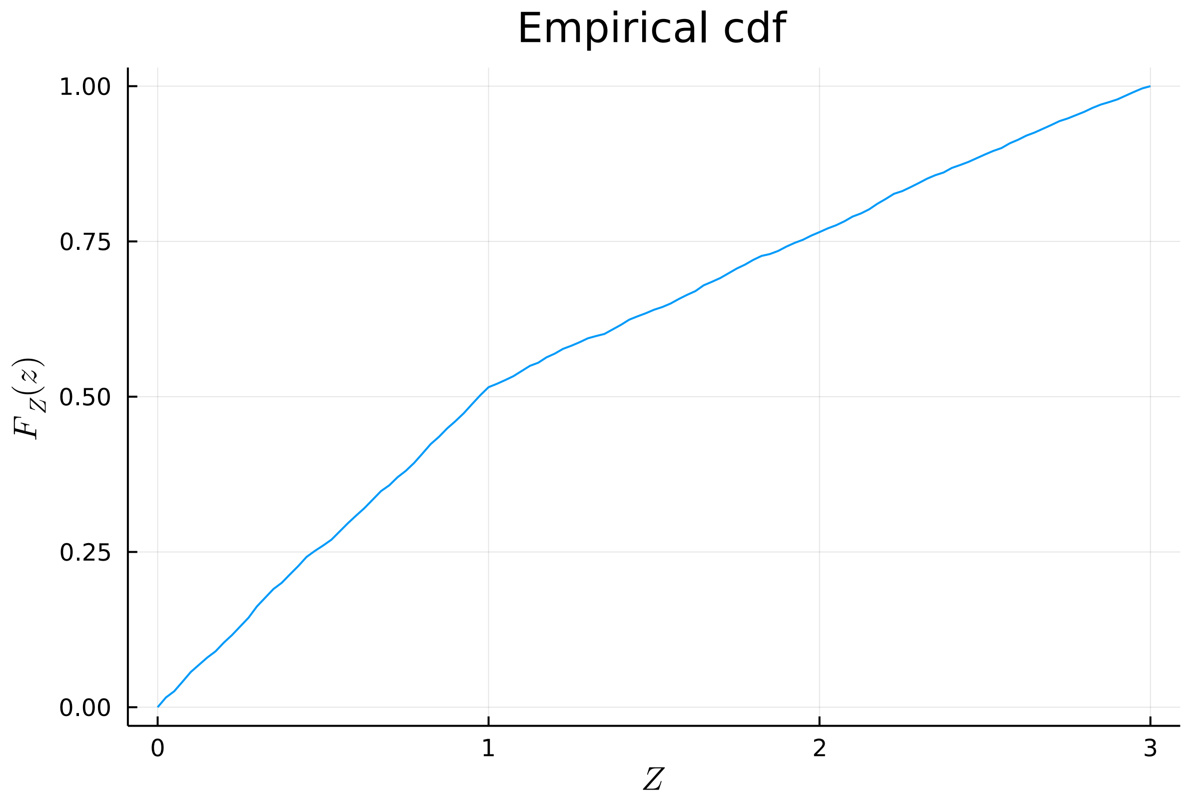
```
Zsamples = rand(Z, N₁)
```

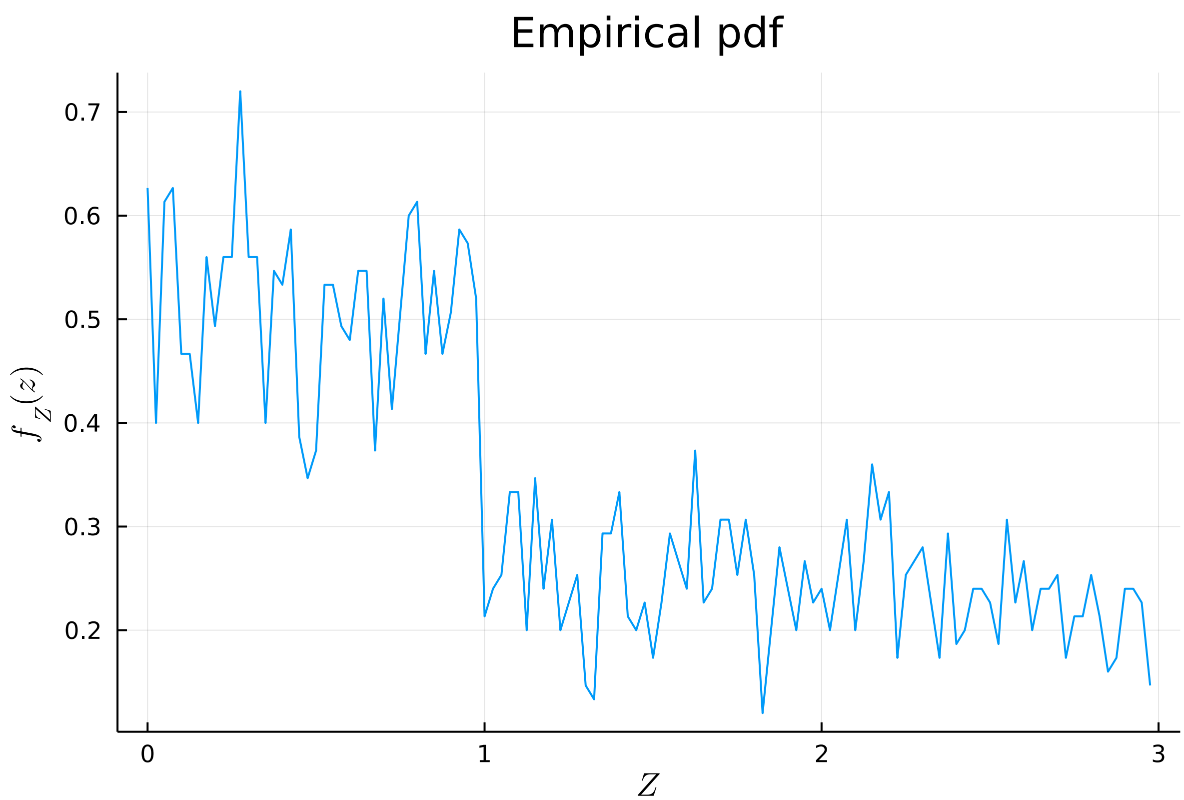
```
h = fit(Histogram, Zsamples);
```




```
[0.626667, 0.4, 0.613333, 0.626667, 0.466667, 0.466667, 0.4, 0.56, 0.493333, 0.56, 0.56
```

```
begin
    cdf(x) = ecdf(ℤsamples)(x);
    z = ℤ.a:0.025:ℤ.c;
    Fz = cdf.(z);
    fz = diff(Fz)./diff(z);
end
```






```
Zmean_error = 0.03863791490235502
```

```
Zmean_error = abs(StatsBase.mean(Zsamples) - mean(Z))
```

Zvar_error = 0.027728786259284766

```
Zvar_error = abs(StatsBase.var(Zsamples) - var(Z))
```

1.2 Summary of Results

Here we will empirically test for convergence of key statistics of the `ZDist` distribution.

```
N = 100; # fixed number of samples
```

```
Zfixedsamples = rand(Z, N);
```

We can compare the empirical mean and variance with the true mean and variance for the fixed number of samples above, and notice that they are indeed very close even for a few samples (in the context of statistical significance).

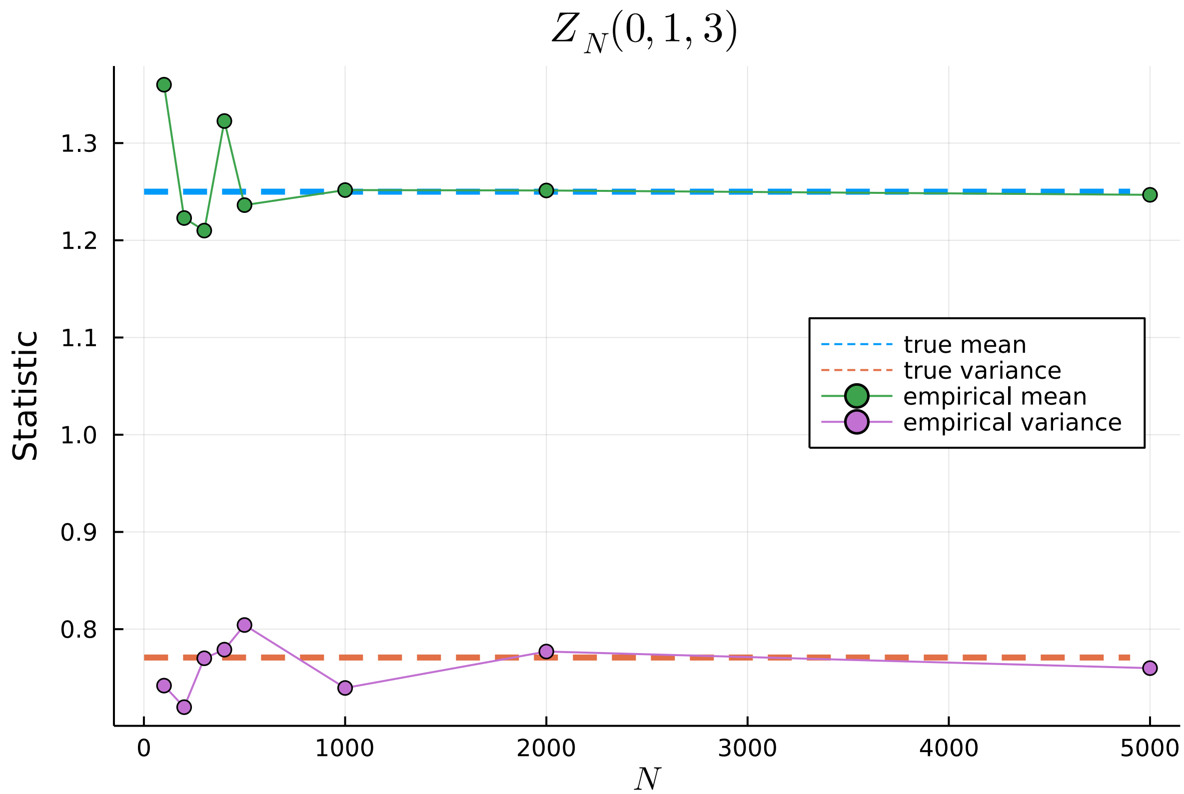
Zfixedmean_error = 0.027980409219780622

```
Zfixedmean_error = abs(StatsBase.mean(Zfixedsamples) - mean(Z))
```

Zfixedvar_error = 0.023735010767492515

```
Zfixedvar_error = abs(StatsBase.var(Zfixedsamples) - var(Z))
```

For testing convergence, we can sample across a wide range of sizes to get an idea of the trend. Below, we sample the $Z(0, 1, 3)$ distribution $N = [100, 200, 300, 400, 500, 1000, 2000, 5000]$ times.



And, indeed, the statistics seem to converge quickly as the number of (independent) samples increase.

2. Independence of Discrete Random Variables

In this section we describe a **categorical distribution** for a biased dice. The probablity mass function (pmf) sampler uses the **alias method** under the hood, which we will be leveraging.

The biased dice has a pmf as follows:

$$P(X=1)=P(X=2)=0.25$$

$$P(X=3)=P(X=4)=P(X=5)=P(X=6)=0.125.$$

We can represent such a categorical pmf as a vector of bin/category probabilities, p, which has size $k=6$ for 6 bins.

```
p = [0.25, 0.25, 0.125, 0.125, 0.125, 0.125];
```

Here we are interested in generating a joint pmf for a sequence of 2 dice throws (X, Y) where both $X, Y \sim \text{Categorical}(p)$. We then define the random variables $Z_1 = X + Y$ and $Z_2 = X - Y$ for which we would also like to analyze the joint pmf. In both joint cases, we want to test for independence of the random variables, which we will do by testing joint pmf factorization, implications from conditional probability, and as well as some analytical proofs.

2.1 Numerical Simulation

We can now sample the distribution to get an idea of the joint pmfs $p_{X,Y}(x, y)$ and $p_{Z_1,Z_2}(z_1, z_2)$.

N_2 = 500

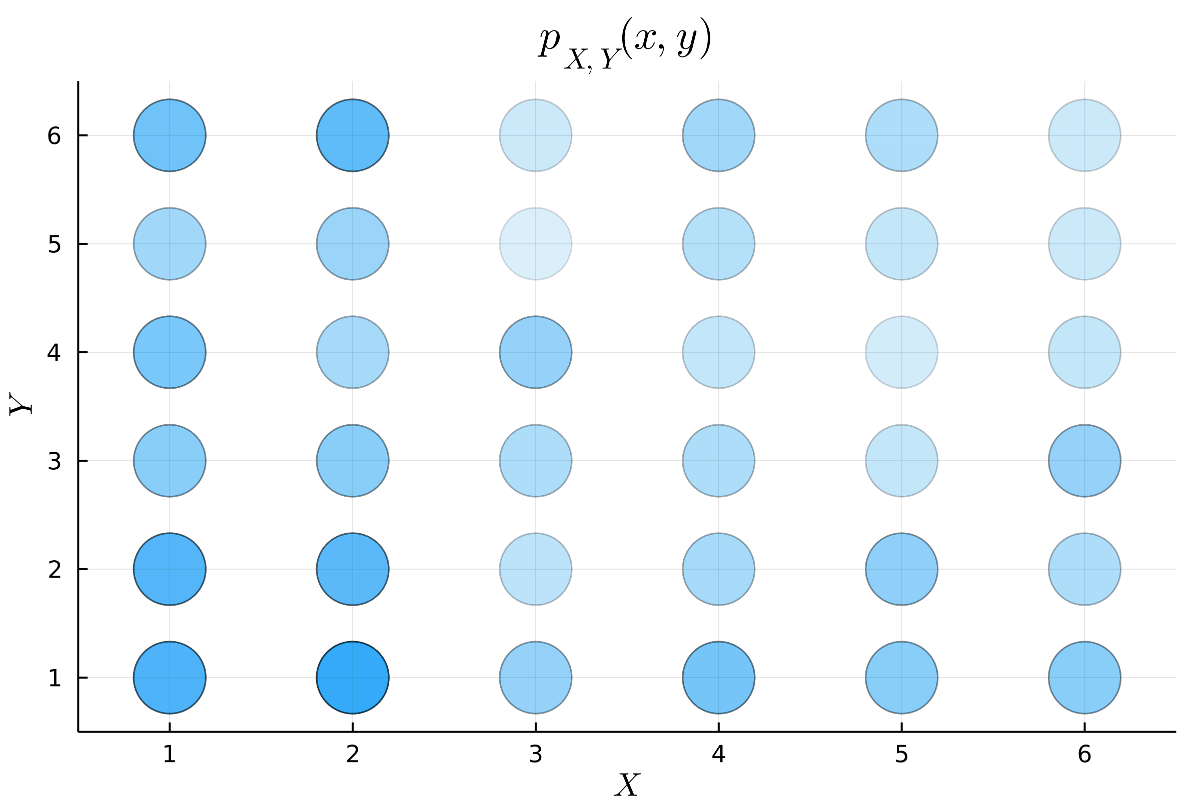
```
matrixtotuple (generic function with 1 method)
```

```
[(2, 1), (2, 2), (5, 5), (5, 1), (2, 6), (1, 2), (2, 6), (6, 5), (4, 6), (2, 1), (4, 5),
```

```
begin
    X  = Categorical(p);  # X with pₓ(k) = p[k]
    Y  = Categorical(p);  # Y with pᵧ(k) = p[k]
    XY = Product([X, Y]); # mixture model with two independent r.v.s X and Y

    XYsamples = rand(XY, N₂) |> matrixtotuple
end
```

The empiricial joint pmf for X and Y is given below:



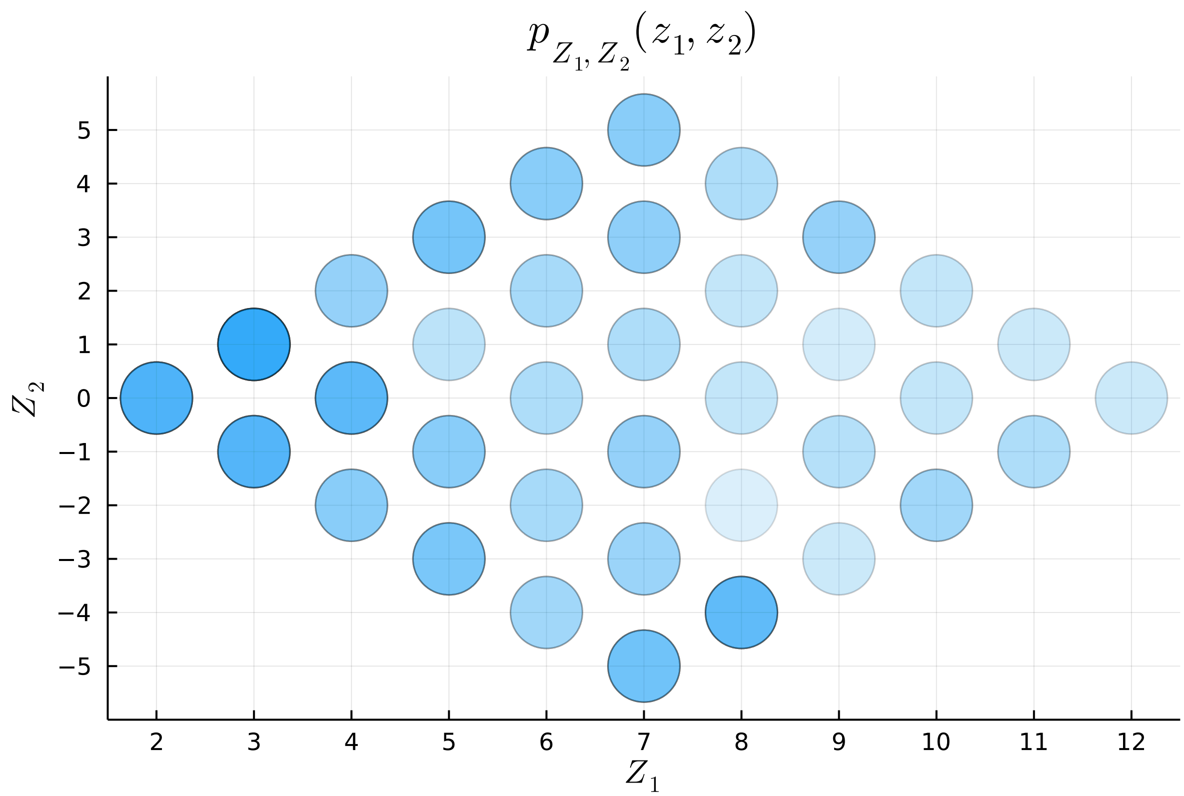
We now define Z_1 and Z_2 in code from the (X, Y) samples already generated. The empiricial joint pmf for Z_1 and Z_2 is also given below.

```
[(3, 1), (4, 0), (10, 0), (6, 4), (8, -4), (3, -1), (8, -4), (11, 1), (10, -2), (3, 1), (
```

```
begin
    Z₁ = [xy[1] + xy[2] for xy ∈ XYsamples] |> transpose
    Z₂ = [xy[1] - xy[2] for xy ∈ XYsamples] |> transpose

    Z₁Z₂samples = vcat(Z₁, Z₂) |> matrixtotuple
end
```

From the simulation, they do not look independent, since low values imply more possibility.



We can already observe here that Z_1 and Z_2 are correlated; knowing Z_1 gives some information about the range of Z_2 and vice-versa.

2.2 Summary of Results

Here we will empirically test for the independence of joint random variables (X, Y) and (Z_1, Z_2). The simulated probabilities is for $N = 100$ samples (defined in section 1.2).

For those fixed N samples, the joint empirical pmf $p_{X,Y}(x, y)$ is shown below, with X along the rows and Y along the columns:

	1	2	3	4	5	6
1	0.06	0.08	0.08	0.03	0.02	0.03
2	0.06	0.05	0.02	0.03	0.03	0.0
3	0.05	0.02	0.02	0.02	0.03	0.01
4	0.04	0.05	0.01	0.01	0.01	0.01
5	0.01	0.06	0.01	0.0	0.0	0.0
6	0.02	0.02	0.03	0.03	0.03	0.02

```
begin
    𝑋𝑌fixedsamples = rand(𝑋𝑌, N) |> matrixtotuple

    𝑋𝑌fixedpmf = zeros(6,6) # initialize
    for (x, y) ∈ 𝑋𝑌fixedsamples
        𝑋𝑌fixedpmf[x, y] += 1
    end
    𝑋𝑌fixedpmf ./= N        # normalize

    DataFrame(𝑋𝑌fixedpmf, ["1", "2", "3", "4", "5", "6"])
end
```

The marginal pmf, $p_X(k)$, is calculated and displayed below:

	$p_X(k)$
1	0.3
2	0.19
3	0.15
4	0.13
5	0.08
6	0.15

```
begin
    #initialize
    𝑋range = size(𝑋𝑌fixedpmf, 1)
    𝑋fixedmarginal = zeros(𝑋range, 1)

    for k ∈ 1:𝑋range
        𝑋fixedmarginal[k] = sum(𝑋𝑌fixedpmf[k, :]) # Σₖp(𝑌=k|𝑋)
    end

    DataFrame(𝑋fixedmarginal, ["pₓ(k)"])
end
```

The marginal pmf, $p_Y(k)$, is calculated and displayed below:

	$P_Y(k)$
1	0.24
2	0.28
3	0.17
4	0.12
5	0.12
6	0.07

```
begin
    #initialize
    Yrange = size(XYfixedpmf, 2)
    Yfixedmarginal = zeros(Yrange, 1)

    # marginalize
    for k ∈ 1:Yrange
        Yfixedmarginal[k] = sum(XYfixedpmf[:, k]) # Σₖp(X=k|Y)
    end

    DataFrame(Yfixedmarginal, ["Pᵧ(k)"])
end
```

Now, to check for independence, we know will use the information that the joint pmf factors into respective marginal pmfs. We will subtract the two matrices, one with the empirical joint pmf $p_{X,Y}(x, y)$ and the other with the factored products of marginal pmfs $p_X(x)p_Y(y)$, to see absolute elementwise error. The factored product pmf was obtained by $\mathbf{p}_X(x)\mathbf{p}_Y(y)^\intercal$.

	1	2	3	4	5	6
1	0.012	0.004	0.029	0.006	0.016	0.009
2	0.0144	0.0032	0.0123	0.0072	0.0072	0.0133
3	0.014	0.022	0.0055	0.002	0.012	0.0005
4	0.0088	0.0136	0.0121	0.0056	0.0056	0.0009
5	0.0092	0.0376	0.0036	0.0096	0.0096	0.0056
6	0.016	0.022	0.0045	0.012	0.012	0.0095

```
begin
    XYpmferror = XYfixedpmf .- (Xfixedmarginal * transpose(Yfixedmarginal)) .|> abs
    DataFrame(XYpmferror, ["1", "2", "3", "4", "5", "6"])
end
```

We can observe that the errors are quite smal and despite a small sample size, X and Y do look indpendent. This will again be verified alternatively using conditional probablility later on.

Now, similarly, we will numerically test for the independence of Z_1 and Z_2. The joint empirical pmf $p_{Z_1,Z_2}(z_1, z_2)$ is shown below, with Z_1 along the rows and Z_2 along the columns:

	-5	-4	-3	-2	-1	0	1	2	3
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.06	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.08	0.0	0.06	0.0	0.0
4	0.0	0.0	0.0	0.08	0.0	0.05	0.0	0.05	0.0
5	0.0	0.0	0.03	0.0	0.02	0.0	0.02	0.0	0.04
6	0.0	0.02	0.0	0.03	0.0	0.02	0.0	0.05	0.0
7	0.03	0.0	0.03	0.0	0.02	0.0	0.01	0.0	0.06
8	0.0	0.0	0.0	0.03	0.0	0.01	0.0	0.01	0.0
9	0.0	0.0	0.01	0.0	0.01	0.0	0.0	0.0	0.03
10	0.0	0.0	0.0	0.01	0.0	0.0	0.0	0.03	0.0
11	0.0	0.0	0.0	0.0	0.0	0.0	0.03	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0	0.02	0.0	0.0	0.0

```julia
begin
    Z₁fixed = [xy[1] + xy[2] for xy in XYfixedsamples] |> transpose
    Z₂fixed = [xy[1] - xy[2] for xy in XYfixedsamples] |> transpose

    Z₁Z₂fixedsamples = vcat(Z₁fixed, Z₂fixed) |> matrixtotuple

    Z₁Z₂fixedpmf = zeros(12, 11)       # initialize
    for (z1, z2) ∈ Z₁Z₂fixedsamples
        Z₁Z₂fixedpmf[z1, z2+6] += 1    # shift z2 to match matrix bounds
    end
    Z₁Z₂fixedpmf ./= N                 # normalize

    DataFrame(Z₁Z₂fixedpmf,
        ["-5", "-4", "-3", "-2", "-1", "0", "1", "2", "3", "4", "5"])
end
```

The marginal pmf, $p_{Z_1}(k)$, is calculated and displayed below:

	$pz_1(k)$
1	0.0
2	0.06
3	0.14
4	0.18
5	0.11
6	0.13
7	0.17
8	0.07
9	0.05
10	0.04
11	0.03
12	0.02

```
begin
    #initialize
    Z₁range = size(Z₁Z₂fixedpmf, 1)
    Z₁fixedmarginal = zeros(Z₁range, 1)

    for k ∈ 1:Z₁range
        Z₁fixedmarginal[k] = sum(Z₁Z₂fixedpmf[k, :]) # Σₖp(Z₂=k|Z₁)
    end

    DataFrame(Z₁fixedmarginal, ["pz₁(k)"])
end
```

The marginal pmf, $p_{Z_2}(k)$, is calculated and displayed below (with shifted indices):

	pz₂(k+6)
1	0.03
2	0.02
3	0.07
4	0.15
5	0.13
6	0.16
7	0.12
8	0.14
9	0.13
10	0.03
11	0.02

```
begin
    #initialize
    Z₂range = size(Z₁Z₂fixedpmf, 2)
    Z₂fixedmarginal = zeros(Z₂range, 1)

    for k ∈ 1:Z₂range
        Z₂fixedmarginal[k] = sum(Z₁Z₂fixedpmf[:, k]) # Σₖp(Z₁=k|Z₂)
    end

    DataFrame(Z₂fixedmarginal, ["pz₂(k+6)"])
end
```

Similar to what was done before for X and Y for checking independence, we will subtract the two matrices, one with the empirical joint pmf $p_{Z_1,Z_2}(z_1, z_2)$ and the other with the factored products of marginal pmfs $p_{Z_1}(z_1)p_{Z_2}(z_2)$, to see absolute elementwise error.

	-5	-4	-3	-2	-1	0	1	2	3
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0018	0.0012	0.0042	0.009	0.0078	0.0504	0.0072	0.0084	0.0078
3	0.0042	0.0028	0.0098	0.021	0.0618	0.0224	0.0432	0.0196	0.0182
4	0.0054	0.0036	0.0126	0.053	0.0234	0.0212	0.0216	0.0248	0.0234
5	0.0033	0.0022	0.0223	0.0165	0.0057	0.0176	0.0068	0.0154	0.0257
6	0.0039	0.0174	0.0091	0.0105	0.0169	0.0008	0.0156	0.0318	0.0169
7	0.0249	0.0034	0.0181	0.0255	0.0021	0.0272	0.0104	0.0238	0.0379
8	0.0021	0.0014	0.0049	0.0195	0.0091	0.0012	0.0084	0.0002	0.0091
9	0.0015	0.001	0.0065	0.0075	0.0035	0.008	0.006	0.007	0.0235
10	0.0012	0.0008	0.0028	0.004	0.0052	0.0064	0.0048	0.0244	0.0052
11	0.0009	0.0006	0.0021	0.0045	0.0039	0.0048	0.0264	0.0042	0.0039
12	0.0006	0.0004	0.0014	0.003	0.0026	0.0168	0.0024	0.0028	0.0026

```julia
begin
    Z₁Z₂pmferror = Z₁Z₂fixedpmf .- (Z₁fixedmarginal*transpose(Z₂fixedmarginal))
.|> abs
    DataFrame(Z₁Z₂pmferror,
        ["-5", "-4", "-3", "-2", "-1", "0", "1", "2", "3", "4", "5"])
end
```

Here we can see that the absolute error is mostly present at a higher magintude and for almost all entries than in the case for (X, Y). But still, it may not directly convey the independence as strongly.

Therefore, an easier test to demonstrate dependence of the random variables (Z_1, Z_2) can be using conditional probablity. Specifically, we can check if the posterior probabilities are much different than the prior probabilities which would imply dependence. More formally, we want to check if,

$$P(Z_2 = z \mid Z_1 = k) = \frac{p_{Z_1,Z_2}(k, z)}{p_{Z_1}(k)} \stackrel{?}{=} p_{Z_2}(z)$$

We can consider the case where we are given $Z_1 = 2$ and we check for the probability of having $Z_2 = 0$. As can be seen from the graph, the probability for this will always be 1 since that is the only possible value for Z_2 because both X and Y will have to be equal to 1. But this would not equal the marginal probability of observing $Z_2 = 0$. We can check the disparity through code as well:

```
pZ₂_given_Z₁ = 1.0
```

```
pZ₂_given_Z₁ = Z₁Z₂fixedpmf[2,0+6]/Z₁fixedmarginal[2]
```

```
pZ₂ = 0.16
```

```
pZ₂ = Z₂fixedmarginal[0+6]
```

Additionally, we can also ***prove*** the dependence of Z_1 and Z_2 by the following argument: if Z_1 is even, then Z_2 has to be even as well, i.e. $P(Z_2 = \text{ odd} \mid Z_1 = \text{ even}) = 0$, and this follows for the flipped case too.

This can be shown by a simple contradiction. If $Z_1 = X + Y = 2k, k \in \mathbb{N}$, and we assume $Z_2 = X - Y = 2m + 1, m \in \mathbb{N}$, then $Z_1 + Z_2 = 2X = 2k + 2m + 1$ which is odd, and does not match the left hand side which will always be even. Hence we arrive at a contradiction and can conslude that if either of Z_1 or Z_2 is even, then the other variable will also be even. The same goes for when either Z_1 or Z_2 are odd, enforcing both variables to be odd. This pattern is also evident in the joint pmf plot and table.

Here, we can notice that even though the probablity for disparity in the eveness or oddness of Z_1 and Z_2 is provably 0, the product of marginals will never reflect this.

On the other hand, we see no such dependence for X and Y, and the posterior and prior probabilities indeed converge in the conditional case as well, espectially for higher number of samples. For $N = 100$, this can be tested below by moving the sliders.

x = 1

y = 3

```
pX_given_Y = 0.2857142857142857
```

```
pX_given_Y = XYfixedpmf[x, y]/Yfixedmarginal[y]
```

```
pX = 0.30000000000000004
```

```
pX = Xfixedmarginal[x]
```

Therefore, we can conclude that (X, Y) are independent while (Z_1, Z_2) are dependent random variables.

3. Code

Note that this lab report can be run on the cloud and viewed as is on the github repository page **here**. All code for the notebook can be accessed **here**.