Assignment 1 - Q4: Simulating Gaussian Distributions

Welcome to Pluto Notebooks! In this course you will be spending quite a bit of time working within these notebooks. These notebooks are an HTML/CSS/Javascript built interface for interacting with and working with Julia. They were inspiried by Jupyter, and you can learn more about the notebooks through **their github**. The notebooks are lightweight (with files that can be used by the normal julia interpretor), and track how notebook cells depend on eachother to re-run cells when their dependencies change.

The only code in this notebook which is not found in julia's base is the plotting code provided by **StatsPlots** and **Plots**. Please go read more about these packages, as they will be useful for you throught the course. Another package we use is **PlutoUI** which contains a multitude of utilities for building interfaces in pluto notebooks (see below). For all the cells, some are hidden by default. To see hidden cells click on the eye to the top-left of the cell of interest.

To complete this assignment you shouldn't need to import any other packages.

In this assignment you will:

1. Learn about using Pluto Notebooks and Stats Plots for visualizing and analysing data.
2. Get experience with calculating the mean, variance, and other metrics of sample data.
3. Build intuition about how a gaussian distribution acts with various parameters.

```
# Import the packages and export their exported functions to the main namespace.
using StatsPlots    , PlutoUI    , Random
```

!!!IMPORTANT!!!

Insert your details below. You should see a green checkmark.

```
student =
  (name = "Joshua George", email = "jjgeorge@ualberta.ca", ccid = "jjgeorge", idnumber = 16
```

```
student = (name="Joshua George", email="jjgeorge@ualberta.ca", ccid="jjgeorge",
idnumber=1665548)
```

Welcome Joshua George! ✅

Gaussian Distribution

A gaussian distribution with mean μ, standard deviation σ. Sample data from this distribution using sample(gd, n). You can also get the mean (mean), standard deviation (stddev), and variance (var) through their corresponding functions.

Main.workspace2.GaussianDistribution

```
# The block of text below add documentation to julia struct or function.
# Check out the live docs to the right when your cursor
# is in GaussianDistribution.
"""
    GaussianDistribution(μ::Float64, σ::Float64)

    A gaussian distribution with mean μ, standard deviation σ. Sample data from this distribution using `sample(gd, n)`. You can also get the mean (`mean(gd)`), standard deviation (`stddev(gd)`), and variance (`var(gd)`) through their corresponding functions.
    """
struct GaussianDistribution
    μ::Float64 # mean
    σ::Float64 # standard deviation
end
```

mean (generic function with 2 methods)

```
mean(gd::GaussianDistribution) = gd.μ
```

stddev (generic function with 1 method)

```
stddev(gd::GaussianDistribution) = gd.σ
```

var (generic function with 1 method)

```
var(gd::GaussianDistribution) = gd.σ^2
```

sample (generic function with 2 methods)

```
function sample(gd::GaussianDistribution, n = 1)
    gd.σ*randn(n) .+ gd.μ
end
```

Implementing basic statistcs from data

Below you will be implementing the sample mean, variance, and standard deviation of a dataset X. This dataset is guaranteed to be a vector of floating point numbers. We want the estimates of the variance and standard deviation to be unbiased.

Great job! ✅ 🎉

```
let
    # Testing cell
    X = [1.0, 2.0, 3.0, 4.0, 5.0]
    μ̂ = mean(X)
    σ̂ = stddev(X)
    v = var(X)
    μ̂, σ̂, v
    if μ̂ == 3.0 && σ̂ == sqrt(2.0) && v == 2.0
        md"**Ok, but something is slightly wrong!** 🤔"
    elseif μ̂ == 3.0 && σ̂ == sqrt(2.5) && v == 2.5
        md"**Great job!** ✅🎉"
    else
        md"**Keep at it** ❌"
    end
end
```

```
mean (generic function with 2 methods)
```

```
function mean(X::Vector{Float64})
    m=0
    for element in X
        m+=element
    end
    sze=length(X)
    result=m/sze

end

```

```
var (generic function with 2 methods)
```

```
function var(X::Vector{Float64})
    s=0
    result2=0
    for i in X
        s+=(i-mean(X))^2
        result2=s/(length(X)-1)
    end
    result2
end

```

```
stddev (generic function with 2 methods)
```

```
function stddev(X::Vector{Float64})
    t=var(X)^(0.5)
end
```

Simulating sample variance

Use the below let blocks to complete question 4(bcde). You will be graded on your written work, not on the code in these cells.

```
0.4051645617153231
```

```julia
let # 4b
    X=rand(10)
    stddev=1.0
    var(X)
    mean(X)
end
```

```
0.0936831266005151
```

```julia
let # 4c
    X=rand(100)
    stddev=1.0
    var(X)
end
```

```
0.5095627542899137
```

```julia
let # 4d, 4e
    X=rand(30)
    stddev=10.0
    mean(X)

end
```

Plotting

Below you will see some example plotting code `plot_density` and `plot_box_and_violin`.

`plot_density` plots a histogram of the data passed through X and a density estimated through a kernel density algorithm (see implementation on **github** for more details).

`plot_box_and_violin` plots a box plot over a violin plot. A violin plot shows the density of the sampled data (same as the `density` function), while the overlayed box plot shows the first quartile, median, and third quartile. More information can be found on **github** about these plotting utilites.

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

```julia
function plot_density(X)
    histogram(
        # data/transform paramters
        X, norm=true,
        # make plot pretty parameters
        grid=false, # removes background grid
        tickdir=:out, # changes tick direction to be out
        lw=1, # makes line width thicker
        color=RGB(87/255, 123/255, 181/255), # Changes fill color of histogram
        legend=nothing, # removes legend
        fillalpha=0.6) # makes the histogram transparent

    density!(X, color=:black, lw=2)
end
```

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

```
function plot_box_and_violin(X)
    plt = violin(
        ["data"], #The label for the data on the x-axis
        X, # the data
        grid=false, # remove the background grid
        tickdir=:out, # set the ticks to be out
        lw=0, # set the line width to be zero
        color=RGB(87/255, 123/255, 181/255), # set color
        legend=nothing)
    boxplot!(
        plt, # explicitly pass in plt object
        ["data"], #The label for the data on the x-axis
        X, # the data
        fillalpha=0.5, # make transparent
        lw=3) # emphasize the lines

end
```

Visualizing the distribution from samples

Below are some sliders you can use to visualize different normal distribtuions interactively. The data is then plotted using the above plotting functions.

Mean: -13.5

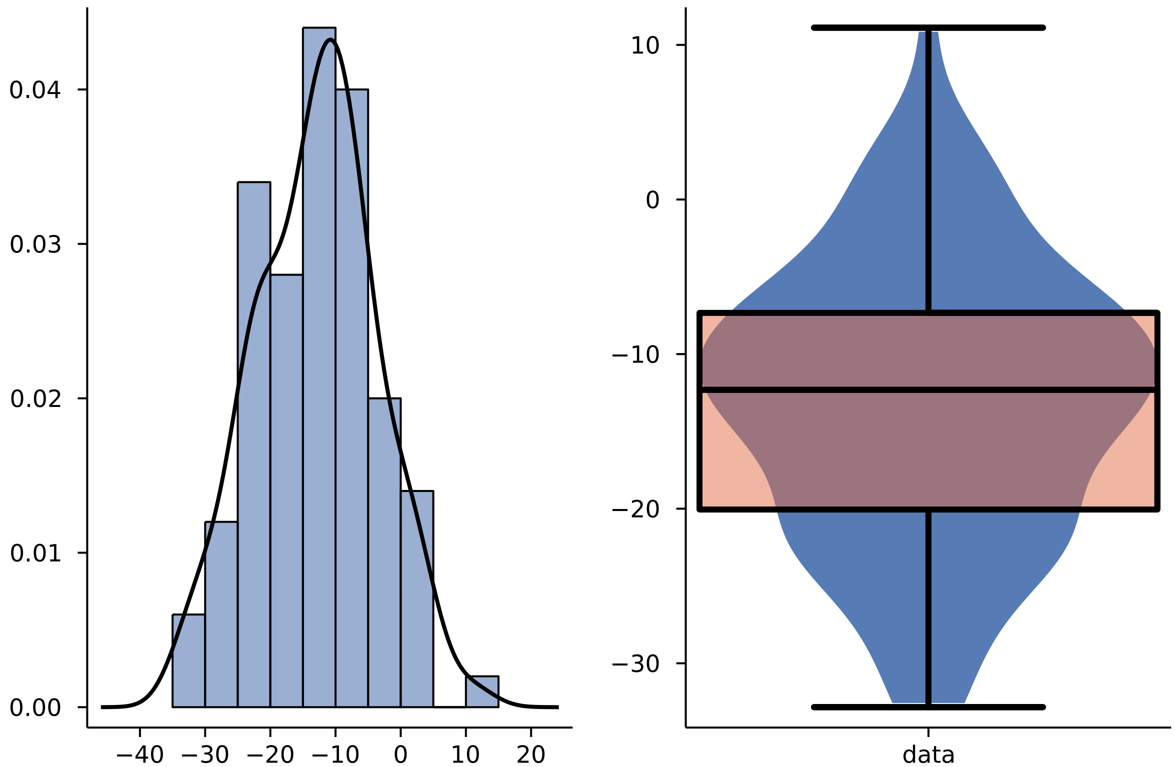
Std Dev: 9.9

number of samples: 100

Resample

```
gd = GaussianDistribution(-13.5, 9.9)
```

```
"Sample mean: -12.973795561004353, Sample Standard Deviation: 9.173276655810415"
```


0.04
0.03
0.02
0.01
0.00
−40
−30
−20
−10
0
10
20
10
0
−10
−20
−30
data