Problem Sheet 4

```julia
begin
    using Distributions
    using PlutoUI
    using LinearAlgebra
    using DelimitedFiles
    using DataFrames
    using KernelFunctions
    using Plots
    using StatsPlots
    default(;linewidth=3.0, legendfontsize=15.0)
end
```

1. Gaussian process (GP) regression

For the GP regression problem, we assume that data are generated as

$$y_i = f(x_i) + \nu_i \qquad i = 1, \ldots, n$$

where the ν_i are independent, zero mean Gaussian noise variables within $E[\nu_i^2] = \sigma^2$ and $f(\cdot)$ has a GP prior with kernel $K(x, x')$.

(a) [MATH] Show that the Bayesian evidence is given by

$$p(\mathbf{y}) = \frac{1}{(2\pi)^{n/2}|\det(\mathbf{K} + \sigma^2\mathbf{I})|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}\mathbf{y}^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y}\right]$$

where $\mathbf{y} = (y_1, \ldots, y_n)$ and the kernel matrix is defined by $\mathbf{K}_{ij} = K(x_i, x_j)$.

Tip

Calculate the joint density of $\mathbf{y}$ and use the fact that $f(x_j)$ and ν_i are independent Gaussian random variables. Hence you can add the respective covariance matrices.

Write your answer here or on paper

(b) [CODE] Create a Gaussian prior with zero mean and a RBF Kernel and sample from it on a grid

```
begin
    x_test = range(0, 10, length = 200)
    k = SqExponentialKernel() # This computes k(x, x') = exp(-0.5||x - x'||^2)
    K = kernelmatrix(k, x_test) + 1e-5I
    prior_f = ## !! FILL IN !! Give the prior distribution
    S = 10 # Number of GP samples
end;
```

Expects 200 elements in each col of y, found 10.

```
1. error(::String) @ error.jl:33
2. _compute_xyz(::StepRangeLen{Float64, Base.TwicePrecision{Float64},
   Base.TwicePrecision{Float64}}, ::Vector{Float64}, ::Nothing,
   ::Bool) @ series.jl:97
3. macro expansion @ series.jl:163 [inlined]
4. apply_recipe(::AbstractDict{Symbol, Any}, ::Type{RecipesPipeline.SliceIt}, ::Any,
   ::Any, ::Any) @ RecipesBase.jl:282
5. _process_userrecipes!(::Any, ::Any, ::Any) @ user_recipe.jl:36
6. recipe_pipeline!(::Any, ::Any, ::Any) @ RecipesPipeline.jl:70
7. _plot!(::Plots.Plot, ::Any, ::Any) @ plot.jl:172
8. #plot#148 @ plot.jl:58 [inlined]
9. top-level scope @ Local: 1 [inlined]
```

```
plot(x_test, rand(prior_f, S); xlabel="x", ylabel="f", label="", alpha=0.5)
```

(c) [MATH] Given a set of training data (X, y), compute the predictive distribution of some test data X_{test}

Write your answer here or on paper

(d) [CODE] Implement the predictive distribution and plot the predictive mean along with one standard error

N 20

logσ -1

```
begin
    σ = exp(logσ)
    X = rand(Uniform(0, 10), N)
    y = sin.(X) + randn(N) * σ
end;
```

Cyclic references among m, pred_mean_and_cov C

```
function pred_mean_and_cov(k, x_test, x, y)
    Kx = kernelmatrix(k, x)
```

```
    Kxtest_x = kernelmatrix(k, x_test, x)
    Kxtest = kernelmatrix(k, x_test) + 1e-5I
    ## !! FILL IN !! This should return the mean and the covariance of the
predictions
    return m, C
end;
```

Cyclic references among m, pred_mean_and_cov C

```
m, C = pred_mean_and_cov(k, x_test, X, y);
```

UndefVarError: C not defined

1. top-level scope @ *Local: 2*

```
begin
    plot(x_test, rand(MvNormal(m, Symmetric(C)), S * 10), color=:black, label="",
alpha=0.1)
    plot!(x_test, m, ribbon = sqrt.(diag(C)), color=:blue, fillalpha=0.2, label =
"Prediction")
    plot!(x_test, sin.(x_test), color=:red, label="f")
    scatter!(X, y, color=:green, label = "Data")
end
```

2. Gibbs sampler for outlier detection

The file *outlier.dat* on the web page of the course contains a data set $D = (y_1, \ldots, y_N)$. Most of the observations have been drawn from a Gaussian probability distribution $\mathcal{N}(y_i; \mu, \sigma^2)$ with mean μ and variance σ^2. However, D contains some **outliers**, which occur with probability ϵ and are displaced by a random offset A_i. For the purpose of **outlier detection** the model is augmented with an indicator variable

$$\delta_i = \begin{cases} 1 & \text{if } y_i \text{ is an outlier,} \\ 0 & \text{if } y_i \text{ is a normal data point,} \end{cases}$$

for each observation. Assuming conjugate priors for the parameters yields the full stochastic model

$$\begin{aligned} \mu &\sim \mathcal{N}(\theta, v^2), & \sigma^{-2} &\sim \text{Gamma}(\kappa, \lambda), & \epsilon &\sim \text{Beta}(\alpha, \beta), \\ y_i &\sim \mathcal{N}(\mu + \delta_i\, A_i, \sigma^2), & \delta_i &\sim \text{Bernoulli}(\epsilon), & A_i &\sim \mathcal{N}(0, \tau^2). \end{aligned}$$

We want to use a Gibbs sampler in order to draw samples from the posterior $p(\mu, \sigma^2, \epsilon, \boldsymbol{\delta}, \mathbf{A}|D)$ with $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_N)$ and $\mathbf{A} = (A_1, \ldots, A_N)$. Some conditional posteriors are given by

$$\begin{aligned} \mu \sim & \mathcal{N}\left(\frac{\sigma^2\theta + v^2 \sum_{i=1}^N (y_i - \delta_i\, A_i)}{\sigma^2 + N v^2}, \frac{\sigma^2 v^2}{\sigma^2 + N v^2}\right), \\ \sigma^{-2} \sim & \text{Gamma}\left(\kappa + \frac{N}{2}, \frac{2\lambda}{2 + \lambda \sum_{i=1}^N (y_i - \delta_i\, A_i - \mu)^2}\right). \end{aligned}$$

- **(a) [MATH] Show that the remaining conditional posteriors are given by**

$$\begin{aligned}
\delta_i \sim& \text{Bernoulli}\left(\frac{\epsilon}{\epsilon + (1-\epsilon)\exp(-A_i(y_i - A_i - \mu)/(2\sigma^2))}\right),\\
A_i \sim& \mathcal{N}\left(\frac{\tau^2\delta_i(y_i-\mu)}{\sigma^2+\tau^2}, \frac{\sigma^2\tau^2}{\sigma^2+\tau^2\delta_i}\right),\\
\epsilon \sim& \text{Beta}\left(\alpha + \sum_{i=1}^N \delta_i, \beta + \sum_{i=1}^N (1-\delta_i)\right).
\end{aligned}$$

Write your answer here or on paper

- **(b) [CODE] Write a program that implements the *Gibbs sampler*. Generate 10^3 samples from the posterior using the hyperparameters $\theta = 0$, $v^2 = 100, \kappa = 2, \lambda = 2, \alpha = 2, \beta = 20, \tau^2 = 100$. Plot histograms showing the marginal posteriors $p(\mu|D)$ and $p(\epsilon|D)$.**

Solution

```
function sample_μ(σ², θ, ν², y, δ, A, N)
    ## !! FILL IN !! it should return a sample for μ
    return
end;
```

```
function sample_σ²(κ, N, λ, y, δ, A, μ)
    ## !! FILL IN !! it should return a sample for σ²
    return
end;
```

```
function sample_δ(ϵ, A, y, μ, σ²)
    ## !! FILL IN !! it should return a vector of samples for δ
    return
end;
```

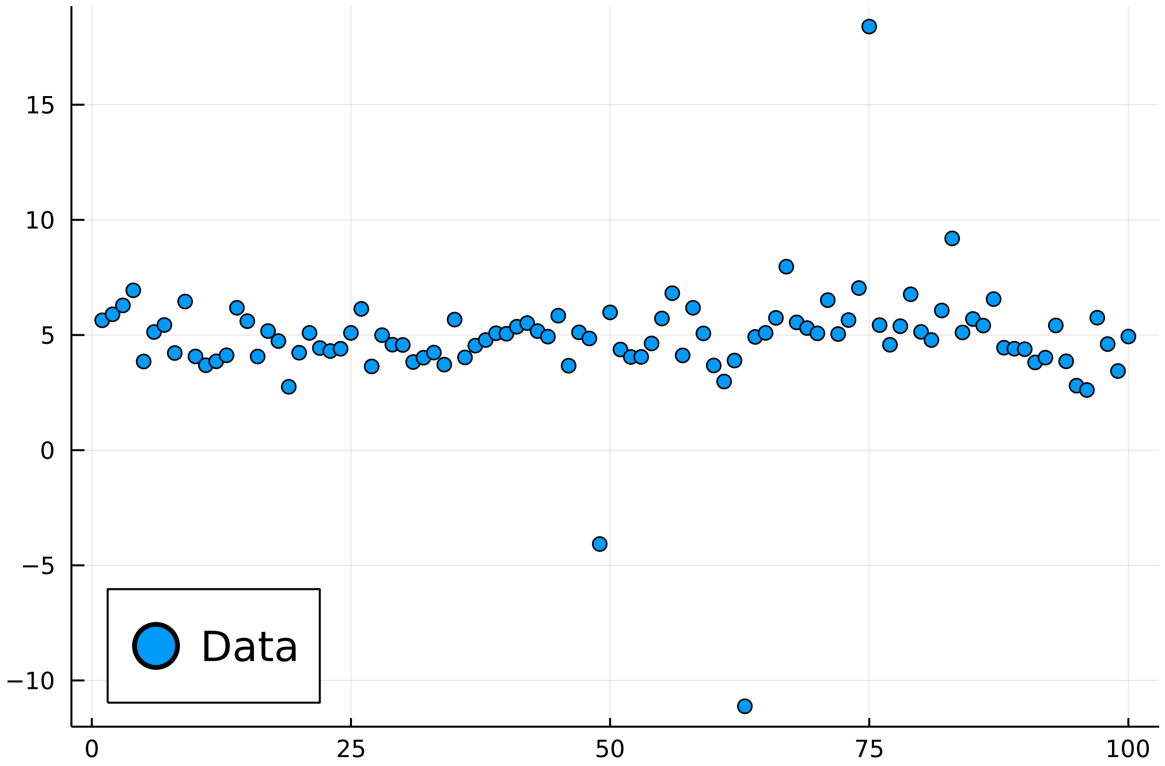
```
function sample_A(τ², δ, y, μ, σ²)
    ## !! FILL IN !! it should return a vector of samples for A
    return
end;
```

```
function sample_ϵ(α, δ, β)
   ## !! FILL IN !! it should return a sample for \epsilon
    return
```

```
end;
```

100

```
begin # We load the data
    y_outlier = vec(readdlm("outlier.dat"))
    Ny = length(y_outlier)
end
```




```
begin # We select multiple hyperparameters
    T = 10000
    θ = 0.0
    ν² = 100
    κ = 2
    λ = 2
    α = 2
    β = 20
    τ² = 100
end;
```

MethodError: Cannot `convert` an object of type Nothing to an object of type Float64

```
Closest candidates are:
convert(::Type{S}, !Matched::CategoricalArrays.CategoricalValue) where
S<:Union{AbstractChar, AbstractString, Number} at
/home/theo/.julia/packages/CategoricalArrays/Fr04b/src/value.jl:79
convert(::Type{T}, !Matched::Base.TwicePrecision) where T<:Number at
twiceprecision.jl:250
convert(::Type{T}, !Matched::AbstractChar) where T<:Number at char.jl:180
...
```

1. **setindex!**(::Vector{Float64}, ::Nothing, ::Int64) @ *array.jl:839*
2. **top-level scope** @ ***Local: 9*** [inlined]

```
begin
    # We initialize the random variables and preallocate storage
    A = randn(Ny); As = zeros(Ny, T)
    δ = rand(0:1, Ny); δs = zeros(Ny, T)
```

```julia
    ϵ = rand(); ϵs = zeros(T)
    σ² = rand(); σ²s = zeros(T)
    μ = randn(); μs = zeros(T)
    for i in 1:T
        μ = sample_μ(σ², θ, ν², y_outlier, δ, A, Ny); μs[i] = μ
        σ² = sample_σ²(κ, Ny, λ, y_outlier, δ, A, μ); σ²s[i] = σ²
        δ = sample_δ(ϵ, A, y_outlier, μ, σ²); δs[:, i] = δ
        A = sample_A(τ², δ, y_outlier, μ, σ²); As[:, i] = A
        ϵ = sample_ϵ(α, δ, β); ϵs[i] = ϵ
    end
end
```

UndefVarError: μs not defined

1. top-level scope @ *Local: 3*

```julia
begin
    p1 = scatter(1:Ny, y_outlier, label = "y")
    p2 = histogram(μs, label = "μ", normalize = true, lw = 0.0)
    p3 = scatter(1:Ny, vec(mean(δs, dims = 2)), label = "δ")
    p4 = histogram(ϵs, label = "ϵ", normalize = true, lw = 0.0)
    plot(p1, p2, p3, p4, legendfontsize=6.0)
end
```

(c) Which data points in the file *outlier.dat* are outliers? Use the samples generated in part (b) and the condition $p(\delta_i|D) \geq 0.02$ in order to identify them.

```julia
begin
    ## !! FILL IN !! Find for which i p(d_i|D) > 0.02
end
```

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