Analytics - Predictive Maintenance - Anomaly Detection

Applications

Hierarchical Model Case

Notebook: Infer_HM_1

Author: Celso Axelrud

Revision: 1.0 (4/8/2018)

In [1]:
```
(*Refs:
    RATS (Hierarchical Model)
    =========================
    https://github.com/dotnet/infer/blob/master/src/Tutorials/MixtureOfGaussians.
    C:\Users\inter\OneDrive\Projects(Comp)\Dev_2018\Infer.NET_2018\infer-master\s
    F# interactive:PLL_infer_4.fsx
*)
```

In [2]:
```
open System
```

In [3]:
```
#load "Paket.fsx" //CA
#load "XPlot.Plotly.Paket.fsx"
#load "XPlot.Plotly.fsx"
open XPlot.Plotly
```

In [4]:
```
#I @"C:\Users\inter\OneDrive\Projects(Comp)\Dev_2018\Infer.NET_2018\packages"
#r "MathNet.Numerics.dll"
#r "MathNet.Numerics.FSharp.dll"

open MathNet.Numerics
//open MathNet.Numerics.Distributions
//open MathNet.Numerics.Statistics

let getValues (histogram:Statistics.Histogram) =
    let bucketWidth = Math.Abs(histogram.LowerBound - histogram.UpperBound) / (fl
    [0..(histogram.BucketCount-1)]
    |> Seq.map (fun i -> (histogram.Item(i).LowerBound + histogram.Item(i).UpperB
```

Description:

```
30 rats whose weights were measured at each of five consecutive weeks.
```

Model:

$$Y_{i,j} \sim Normal(\alpha_i + \beta_i * (x_j - x_{bar}), \theta_c)$$

$$\alpha_i \sim Normal(\alpha_c, \theta_\alpha)$$

$$\beta_i \sim Normal(\beta_c, \theta_\beta)$$

$$x_{bar} = 22$$

$$\alpha_0 = \alpha_c - \beta_c * x_{bar}$$

$\alpha_c, \theta_\alpha, \beta_c, \theta_\beta, \theta_c$ are given independent noninformative priors.

In [5]:
```
#I @"C:\Users\inter\OneDrive\Projects(Comp)\Dev_2018\Infer.NET_2018\infer-master\

#r "netstandard.dll"
#r "Microsoft.ML.Probabilistic.Compiler.dll"
#r "Microsoft.ML.Probabilistic.dll"
#r "Microsoft.ML.Probabilistic.FSharp.dll"

#r "FSharp.Core.dll"
```

In [6]:
```
open Microsoft.ML.Probabilistic
open Microsoft.ML.Probabilistic.FSharp
open Microsoft.ML.Probabilistic.Models
open Microsoft.ML.Probabilistic.Distributions
open Microsoft.ML.Probabilistic.Math
```

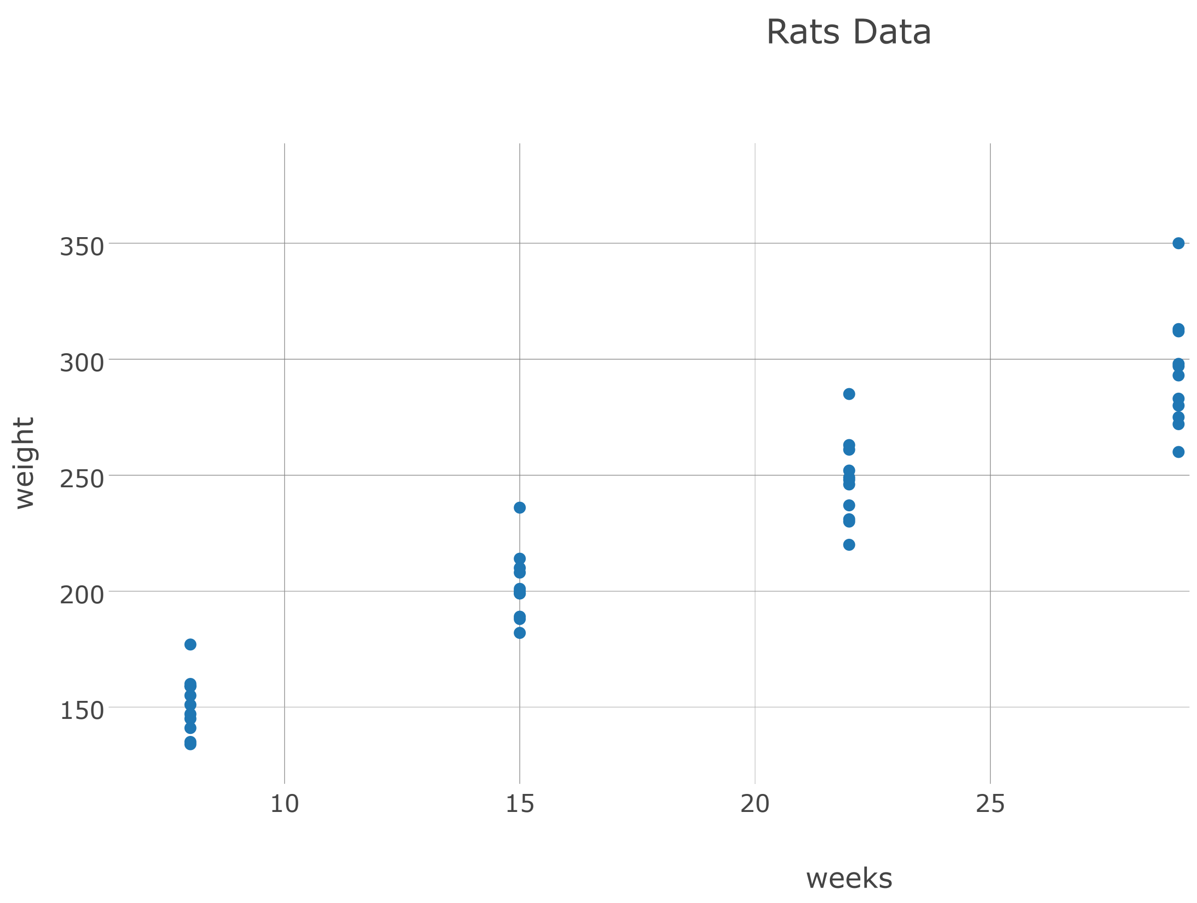
In [7]:
```
//The Data-------------------------------------------
// Height data
let RatsHeightData:double[,] = array2D [
        [ 151.0; 199.0; 246.0; 283.0; 320.0 ];
       [ 145.0; 199.0; 249.0; 293.0; 354.0 ];
       [ 147.0; 214.0; 263.0; 312.0; 328.0 ];
       [ 155.0; 200.0; 237.0; 272.0; 297.0 ];
       [ 135.0; 188.0; 230.0; 280.0; 323.0 ];
       [ 159.0; 210.0; 252.0; 298.0; 331.0 ];
       [ 141.0; 189.0; 231.0; 275.0; 305.0 ];
       [ 159.0; 201.0; 248.0; 297.0; 338.0 ];
       [ 177.0; 236.0; 285.0; 350.0; 376.0 ];
       [ 134.0; 182.0; 220.0; 260.0; 296.0 ];
       [ 160.0; 208.0; 261.0; 313.0; 352.0 ];
       [ 143.0; 188.0; 220.0; 273.0; 314.0 ];
       [ 154.0; 200.0; 244.0; 289.0; 325.0 ];
       [ 171.0; 221.0; 270.0; 326.0; 358.0 ];
       [ 163.0; 216.0; 242.0; 281.0; 312.0 ];
       [ 160.0; 207.0; 248.0; 288.0; 324.0 ];
       [ 142.0; 187.0; 234.0; 280.0; 316.0 ];
       [ 156.0; 203.0; 243.0; 283.0; 317.0 ];
       [ 157.0; 212.0; 259.0; 307.0; 336.0 ];
       [ 152.0; 203.0; 246.0; 286.0; 321.0 ];
       [ 154.0; 205.0; 253.0; 298.0; 334.0 ];
       [ 139.0; 190.0; 225.0; 267.0; 302.0 ];
       [ 146.0; 191.0; 229.0; 272.0; 302.0 ];
       [ 157.0; 211.0; 250.0; 285.0; 323.0 ];
       [ 132.0; 185.0; 237.0; 286.0; 331.0 ];
       [ 160.0; 207.0; 257.0; 303.0; 345.0 ];
       [ 169.0; 216.0; 261.0; 295.0; 333.0 ];
       [ 157.0; 205.0; 248.0; 289.0; 316.0];
       [ 137.0; 180.0; 219.0; 258.0; 291.0 ];
       [ 153.0; 200.0; 244.0; 286.0; 324.0]
    ]
```

In [8]:
```
// x data
let RatsXData:double[] = [| 8.0; 15.0; 22.0; 29.0; 36.0 |]
```

In [9]:
```
//CHart 1
let Points = [ for c in 0 .. 4 do
                  for r in 0..10 ->
                      RatsXData.[c], RatsHeightData.[r,c] ]
```

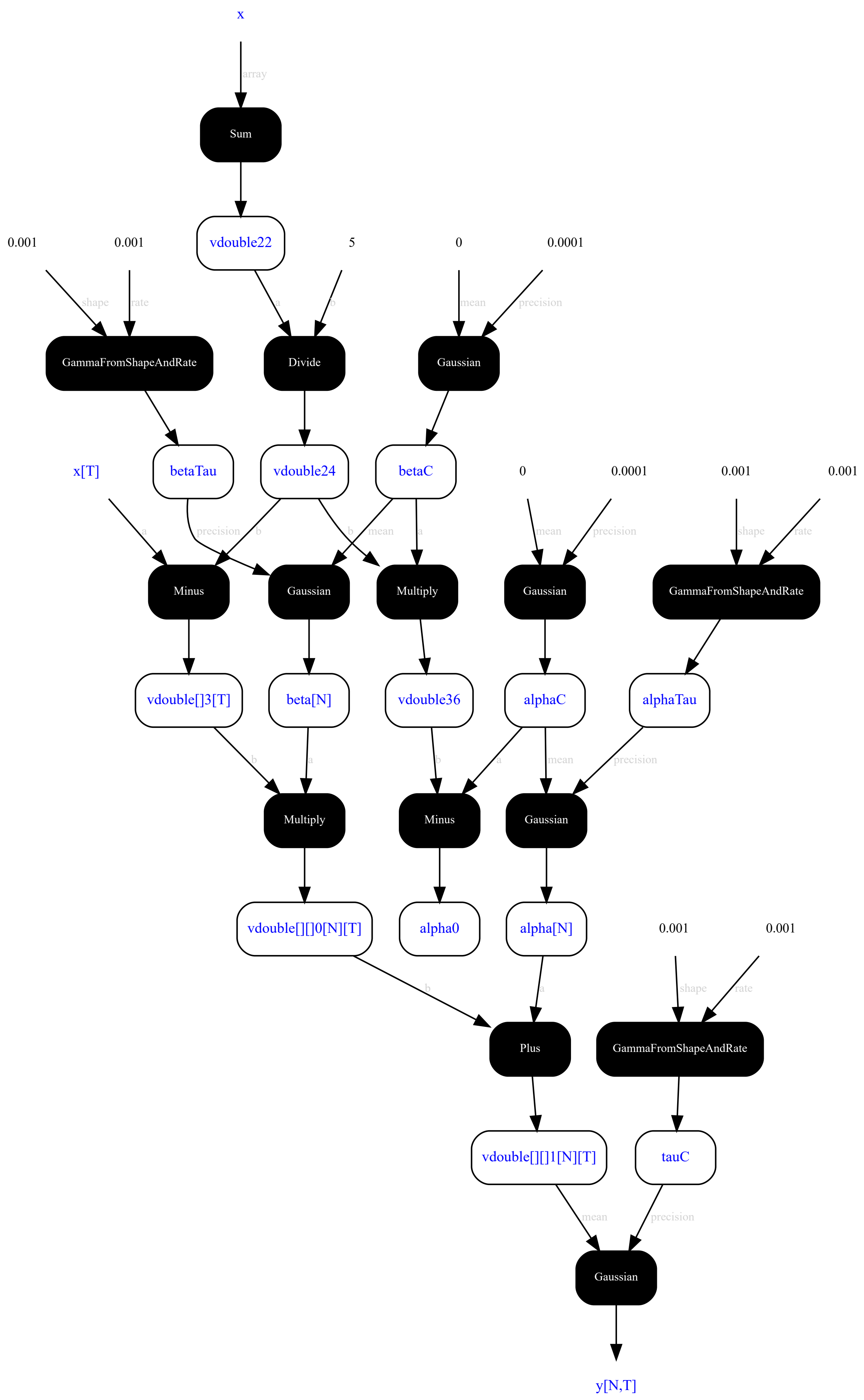
In [10]:
```
Points   |> Chart.Scatter
         |> Chart.WithTitle "Rats Data"
         |> Chart.WithXTitle "weeks"
         |> Chart.WithYTitle "weight"
```

Out[10]:



In [11]:
```
//The model---------------------------------------------------
Rand.Restart(12347)
let N = Range(RatsHeightData.GetLength(0)).Named("N")
let T = Range(RatsHeightData.GetLength(1)).Named("T")
let alphaC =   Variable.GaussianFromMeanAndPrecision(0.0, 1e-4).Named("alphaC")
let alphaTau = Variable.GammaFromShapeAndRate(1e-3, 1e-3).Named("alphaTau")

let alpha = (Variable.ArrayInit N (fun _ -> Variable.GaussianFromMeanAndPrecision
let betaC = Variable.GaussianFromMeanAndPrecision(0.0, 1e-4).Named("betaC")
let betaTau = Variable.GammaFromShapeAndRate(1e-3, 1e-3).Named("betaTau")

let beta = (Variable.ArrayInit N (fun _ -> Variable.GaussianFromMeanAndPrecision(
let tauC = Variable.GammaFromShapeAndRate(1e-3, 1e-3).Named("tauC")
let x = Variable.Observed<double>(RatsXData, T).Named("x")
let xbar = Variable.Sum(x) / (float T.SizeAsInt) //CA
let y = Variable.Observed<double>(RatsHeightData, N, T).Named("y")

Variable.AssignVariableArray2D y N T (fun n t  -> Variable.GaussianFromMeanAndPre

let alpha0 = (alphaC - betaC * xbar).Named("alpha0")
```

Factor Graph


x
array
Sum
vdouble22
0.001
0.001
5
0
0.0001
shape
rate
a
b
mean
precision
GammaFromShapeAndRate
Divide
Gaussian
betaTau
vdouble24
betaC
x[T]
0
0.0001
0.001
0.001
Minus
Gaussian
Multiply
Gaussian
GammaFromShapeAndRate
vdouble[]3[T]
beta[N]
vdouble36
alphaC
alphaTau
Multiply
Minus
Gaussian
vdouble[][]0[N][T]
alpha0
alpha[N]
0.001
0.001
Plus
GammaFromShapeAndRate
vdouble[][]1[N][T]
tauC
Gaussian
y[N,T]

In [12]:
```
// Initialise with the mean of the prior (needed for Gibbs to converge quickly)
alphaC.InitialiseTo(Gaussian.PointMass(0.0))
tauC.InitialiseTo(Gamma.PointMass(1.0))
alphaTau.InitialiseTo(Gamma.PointMass(1.0))
betaTau.InitialiseTo(Gamma.PointMass(1.0))
()
```

In [13]:
```
 // Inference engine
let ie = InferenceEngine()
ie.ShowFactorGraph<-false
```

In [14]:
```
let betaCMarg = ie.Infer<Gaussian>(betaC)
betaCMarg
```

Out[14]:
```
Gaussian(6.186, 0.01315)

Compiling model...done.
Iterating:
.........|.........|.........|.........|.........| 50
```

In [29]:
```
let alphaCMarg = ie.Infer<Gaussian>(alphaC)
let alphaTauMarg = ie.Infer<Gamma>(alphaTau)
printfn "alphaC: %A, alphaTau: %A" alphaCMarg alphaTauMarg
```

```
alphaC: Gaussian(242.5, 7.623), alphaTau: Gamma(12.84, 0.0003832)[mean=0.00492
1]
```

In [30]:
```
let alphaMarg = ie.Infer<Gaussian[]>(alpha)
printfn "alpha: %A" alphaMarg
```

```
alpha: [|Gaussian(239.9, 8.436); Gaussian(247.8, 8.559); Gaussian(252.5, 9.20
5);
  Gaussian(232.6, 8.63); Gaussian(231.6, 8.422); Gaussian(249.8, 8.44);
  Gaussian(228.7, 8.469); Gaussian(248.4, 8.387); Gaussian(283.4, 9.168);
  Gaussian(219.2, 8.481); Gaussian(258.3, 8.53); Gaussian(228.1, 8.48);
  Gaussian(242.4, 8.382); Gaussian(268.3, 8.642); Gaussian(242.8, 8.64);
  Gaussian(245.3, 8.403); Gaussian(232.2, 8.405); Gaussian(240.5, 8.421);
  Gaussian(253.8, 8.582); Gaussian(241.6, 8.438); Gaussian(248.6, 8.446);
  Gaussian(225.2, 8.478); Gaussian(228.5, 8.466); Gaussian(245.1, 8.497);
  Gaussian(234.5, 8.491); Gaussian(254, 8.418); Gaussian(254.4, 8.468);
  Gaussian(243, 8.514); Gaussian(217.9, 8.506); Gaussian(241.4, 8.379)|]
```

In [31]:
```
let alpha0Marg = ie.Infer<Gaussian>(alpha0)
printfn "alpha0:%A [sd=%f]" alpha0Marg (Math.Sqrt(alpha0Marg.GetVariance()))
```

```
alpha0:Gaussian(106.4, 13.99) [sd=3.740131]
```

In [32]:
```
let betaCMarg = ie.Infer<Gaussian>(betaC)
let betaTauCMarg = ie.Infer<Gamma>(tauC)
printfn "betaC: %A, betaTau: %A" betaCMarg betaTauCMarg
```

```
betaC: Gaussian(6.186, 0.01315), betaTau: Gamma(42.26, 0.0006696)[mean=0.0283]
```

In [34]:
```
let betaMarg = ie.Infer<Gaussian[]>(beta)
printfn "beta: %A" betaMarg
```

```
beta: [|Gaussian(6.061, 0.07115); Gaussian(7.078, 0.0771); Gaussian(6.491, 0.07
867);
  Gaussian(5.316, 0.07754); Gaussian(6.581, 0.07187); Gaussian(6.174, 0.07122);
  Gaussian(5.971, 0.07181); Gaussian(6.423, 0.07096); Gaussian(7.079, 0.07953);
  Gaussian(5.835, 0.07194); Gaussian(6.819, 0.07366); Gaussian(6.118, 0.07076);
  Gaussian(6.163, 0.07076); Gaussian(6.706, 0.07338); Gaussian(5.394, 0.0762);
  Gaussian(5.915, 0.07133); Gaussian(6.276, 0.0709); Gaussian(5.835, 0.07181);
  Gaussian(6.412, 0.07259); Gaussian(6.05, 0.07137); Gaussian(6.412, 0.07149);
  Gaussian(5.847, 0.07202); Gaussian(5.734, 0.07262); Gaussian(5.88, 0.07205);
  Gaussian(6.932, 0.07476); Gaussian(6.559, 0.07159); Gaussian(5.892, 0.07152);
  Gaussian(5.836, 0.0727); Gaussian(5.655, 0.07303); Gaussian(6.129, 0.07075)|]
```

In [35]:
```
let tauCMarg = ie.Infer<Gamma>(tauC)
printfn "tauC:%A [sd=%f]" tauCMarg (Math.Sqrt(tauCMarg.GetVariance()))
```

```
tauC:Gamma(42.26, 0.0006696)[mean=0.0283] [sd=0.004353]
```

Charts

In [24]:
```
let Mean1b=ie.Infer<Gaussian>(alphaC).GetMean()
let Std1b=Math.Sqrt(ie.Infer<Gaussian>(alphaC).GetVariance())
Mean1b,Std1b
```

Out[24]:
```
(242.489319, 2.761011433)
```

In [25]:
```
//let P_alphaC = Distributions.Normal.WithMeanPrecision(0.0, 1e-4)
let Mean1a=0.0
let Std1a=1.0/(1e-4)
let P_alphaC = Distributions.Normal.WithMeanStdDev(Mean1a,Std1a)
Mean1a,Std1a
```

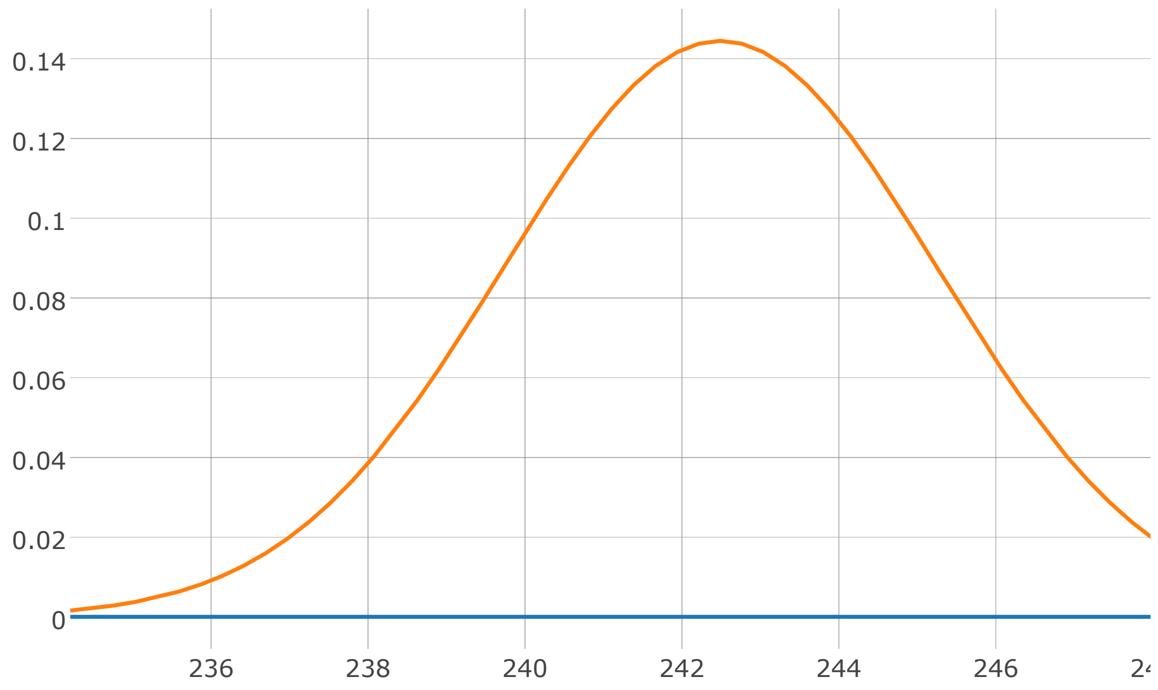
Out[25]:
```
(0.0, 10000.0)
```

In [26]:
```
let X1b=[|for i in (Mean1b-3.0*Std1b)..(Std1b/10.0)..(Mean1b+3.0*Std1b)->i|]
let Y1b=X1b |> Array.map(fun x-> Distributions.Normal.PDF(Mean1b,Std1b,x))
```

In [27]:
```
//let X1a=[|for i in (Mean1a-3.0*Std1a)..(Std1a/10.0)..(Mean1a+3.0*Std1a)->i|]
let X1a=X1b
let Y1a=X1a |> Array.map(fun x-> Distributions.Normal.PDF(Mean1a,Std1a,x))
```

In [28]:
```
let Trace1 =
    Scatter(x = X1a,y = Y1a,name="alphaC prior")

let Trace2 =
    Scatter(x = X1b,y = Y1b,name="alphaC posterior")

//Trace2 |> Chart.Plot
[Trace1;Trace2] |> Chart.Plot
```

Out[28]:



In [ ]: