Applied Measure Theory for Composable Statistical Modeling

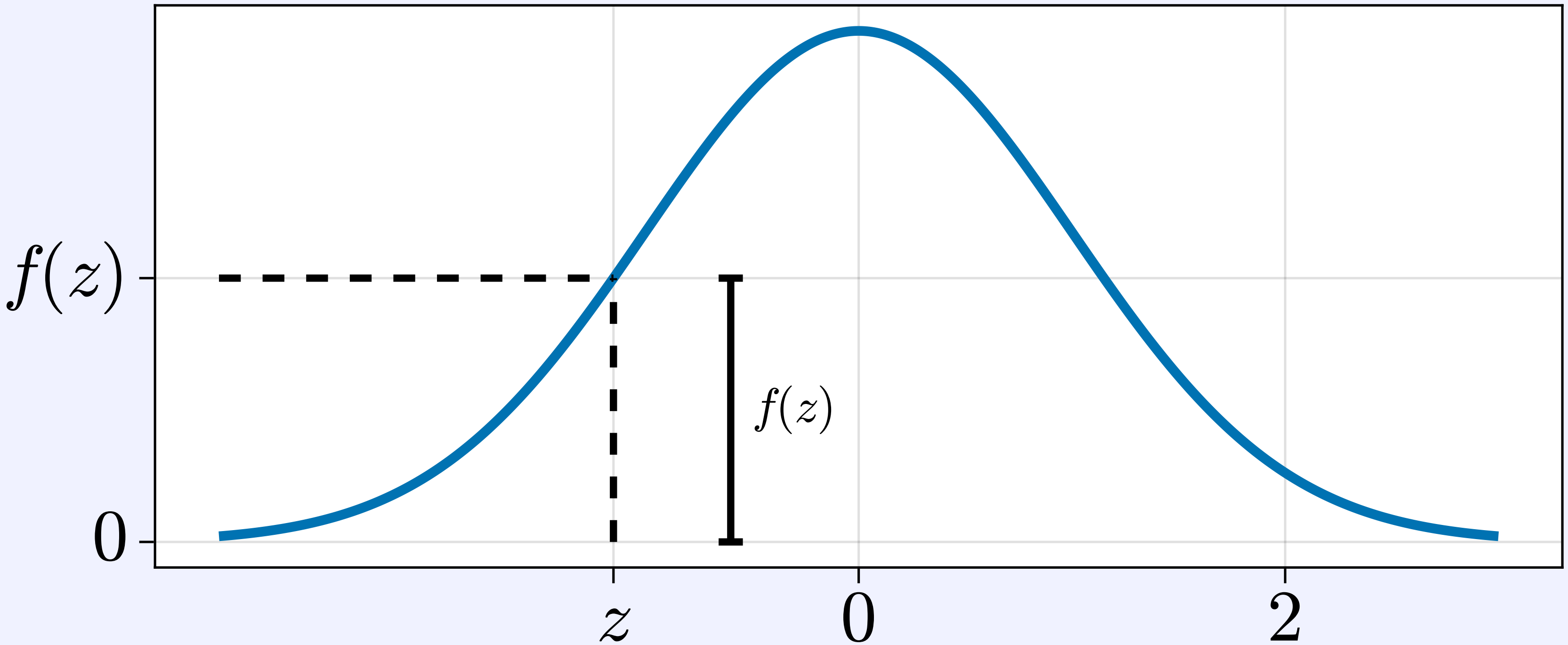
Chad Scherrer

Goals

- Introduction measure theory at a high level
- Focus on structural intuition and composability
- Introduce some basic concepts in measure theory
- See these implemented in MeasureTheory.jl and Tilde.jl

Motivation: Probability Density Functions

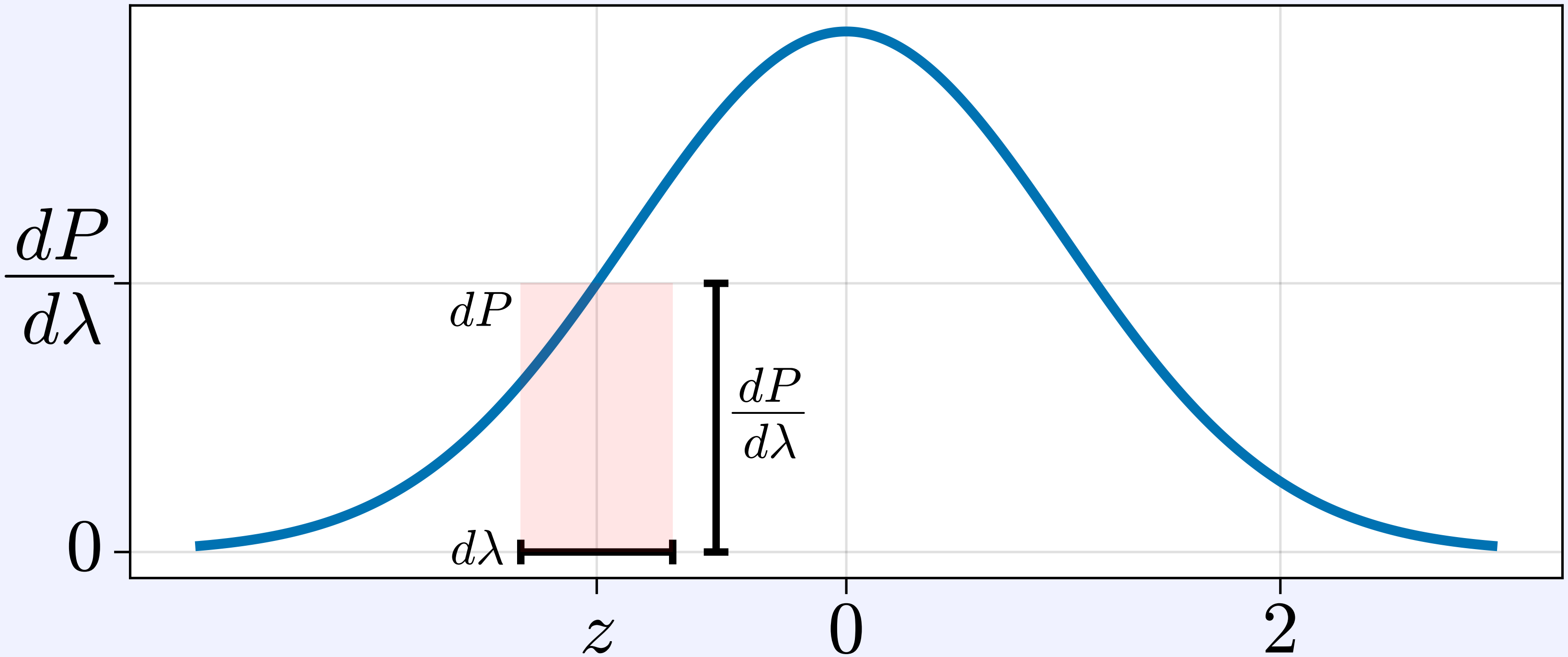
A normal distribution has a *density* of $f(z) = \frac{e^{-z^2/2}}{\sqrt{2\pi}}$



But… "Density" with respect to what?

Motivation: Probability Density Functions

A normal distribution has a *density* of $\frac{dP}{d\lambda} = \frac{e^{-z^2/2}}{\sqrt{2\pi}}$



Distributions are usually defined in terms of measures

Our Approach

For construction

- *Primitive* measures like `LebesgueMeasure()` and `CountingMeasure()`
- *Combinators* for building new measures from existing ones

For computation

- Each measure has a `logdensity_def` and `basemeasure`

σ-Additivity

A function

$$f : (\Sigma \subseteq 2^X) \to \mathbb{R}_{0+}$$

is *σ-additive* if

$$f\left(\bigcup_{k=1}^{\infty} E_k\right) = \sum_{k=1}^{\infty} f(E_k)$$

for all pairwise-disjoint sequences $(E_k \in \Sigma \mid k \in \mathbb{Z}_+)$.

Measures

A *measure* is a σ-additive function

$$\mu : (\Sigma \subseteq 2^X) \to \mathbb{R}_{0+}$$

- Σ is the *set of measurable sets*
- μ is a *measure on* X, written $\mu \in \mathcal{M}(X)$

Tilde.jl

A Tilde model has a *latent space* and (optionally) a *return value*.

```
m = @model s begin
    x ~ Normal(σ=s)
    return x^2
end
```

A model yields a measure over `NamedTuples`

```
rand(m(3))
```

```
(x = 1.321067350106375,)
```

```
logdensityof(m(3), (x = 1.0,))
```

```
-2.073106377428338
```

We can get the return value with `predict`

```
predict(m(3))
```

```
13.974225587905508
```

```
predict(m(3), (x = 2.0,))
```

```
4.0
```

Measurable Functions $f : X \to Y$

- Just as a **continuous** function is one whose inverse preserves **openness**, ...

- a **measurable** function is one whose inverse preserves **measurability**.

$f : X \to Y$ is *measurable* if

$$B \in \Sigma_Y \Rightarrow f^{-1}(B) \in \Sigma_X$$

Kernels

- A *kernel* is a function $X \times \Sigma_Y \to \mathbb{R}_{0+}$
- Currying turns this into $\kappa : X \to \mathcal{M}(Y)$
- We'll write $\kappa \in \mathcal{K}(X, Y)$
- Note that a *distribution family* or *parameterized measure* is a kernel

Abstract Transition Kernels

Normal approximation to a `Poisson(λ)`

```
k = kernel() do λ
    Normal(μ=λ, σ²=λ)
end

k(2.0)
```

```
Normal(μ = 2.0, σ² = 2.0)
```

Compared to a generic function, a kernel "knows"

- The return value is a measure
- `TypedTransitionKernels` know t
  type of that measure
- `ParameterizedTransitionKern`
  know the specific parameterization

Parameterized Measures

```
m1 = Normal(2,3)
logdensityof(m1, 4)
```

-2.2397730440950046

```
m2 = Normal(μ=2, σ²=9)
logdensityof(m2, 4)
```

-2.239773044095005

```
m3 = Normal(μ=2, λ=1/3)
logdensityof(m3, 4)
```

-2.2397730440950046

```
m4 = Normal(μ=2, τ=1/9)
logdensityof(m4, 4)
```

-2.239773044095005

Integration and Densities

Given $\mu \in \mathcal{M}(X)$ and measurable $f : X \to \mathbb{R}_{0+}$, we can define $d\nu = f\ d\mu$. Then

Lebesgue integration gives

$$\nu(A) = \int_A f\, d\mu$$

The *density* or *Radon-Nikodym derivative* is

$$f = \frac{d\nu}{d\mu}$$

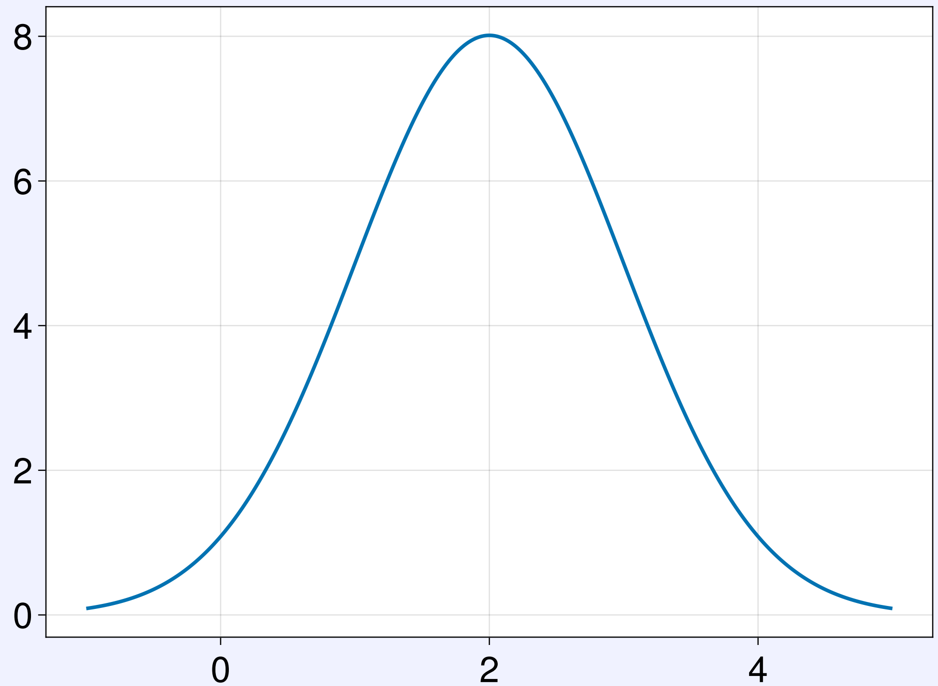
Integration and Densities

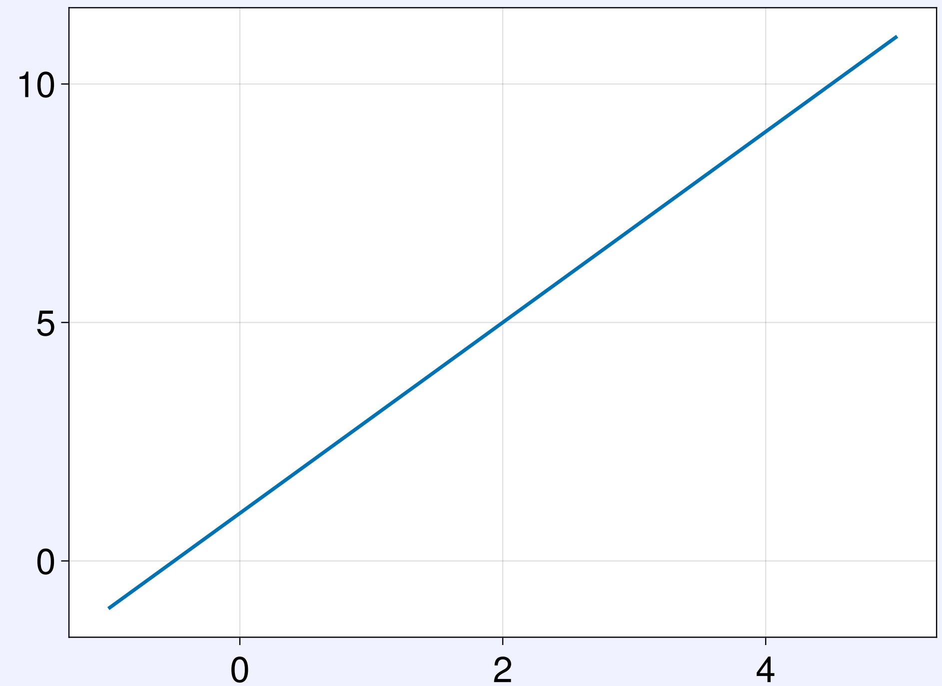
```
ν = ∫exp(Normal()) do z
    2z + 1
end
```

```
DensityMeasure
∫(LogFuncDensity(var"#13#14"()),
Normal())
```

```
d = 𝒹(ν, Normal())
```

```
MeasureBase.Density{MeasureBase.Densit
 Normal{(), Tuple{}}}, Normal{(),
Tuple{}}}(DensityMeasure
∫(LogFuncDensity(var"#13#14"()),
Normal()), Normal())
```





Likelihoods

A *likelihood* is a function $\ell : \Theta \to \mathbb{R}$ of the form

$$\ell(\theta) = \frac{d\kappa(\theta)}{d\beta}(y)$$

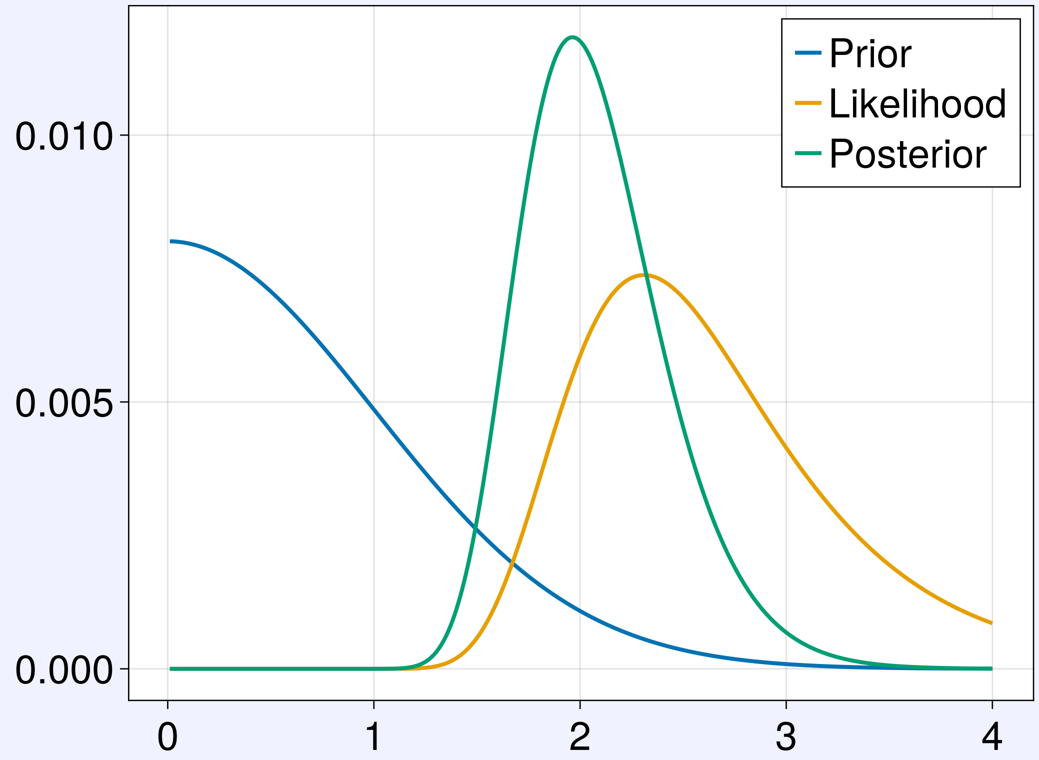
where

Likelihoods

```
prior = HalfNormal()

lik = likelihoodof(obs) do σ
    Normal(σ=σ) ^ 10
end

post = prior ⊙ lik
```



Pushforward Measure

Let $\mu \in \mathcal{M}(X)$ and $f : X \to Y$.

For $B \subseteq Y$, define

$$\nu(B) = \mu(f^{-1}(B))$$

Then ν is the *pushforward of* μ *along* f, written

$$\nu = f_*(\mu) \in \mathcal{M}(Y)$$

Example: Sampling from a Pushforward

```
Random.seed!(rng, 1)
f(x) = 3 * x + 4
μ = Normal()
```

$$\nu = f_*(\mu)$$

$$y \sim \nu$$

$$z \sim \mu$$

$$y = f(z)$$

```
m = @model begin
    ν = pushfwd(f, μ)
    y ~ ν
    return y
end

predict(rng, m())
```

3.788250583138306

```
m = @model begin
    z ~ μ
    y = f(z)
    return y
end

predict(rng, m())
```

3.788250583138306

$\mathcal{M}$ is a Functor

- $\mathcal{M}$ takes a **point** to a Dirac measure on that point

$$\mathcal{M}(x) = \delta_x$$

- $\mathcal{M}$ takes a **function** to its pushforward

$$\mathcal{M}(f) = f_*$$

$\mathcal{M}$ is a Monad

- The *unit* or *return* is $\mathcal{M}(x) = \delta_x$
- Given $\mu \in \mathcal{M}(X)$ and $\kappa : X \to \mathcal{M}(Y)$, define

$$\mu \gg\!= \kappa \in \mathcal{M}(Y)$$

$$(\mu \gg\!= \kappa)(B) = \int_X \kappa(x)(B)\; d\mu(x)$$

This is the *monadic bind* operator

Implementation

Log-density evaluation

Evaluating `logdensityof(Normal(3,4), 1.0)`

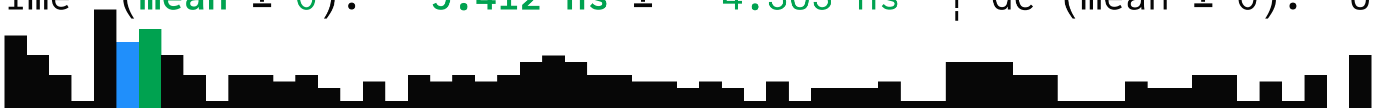
Measure	`logdensity_def`
`AffinePushfwd((μ = 3, σ = 4), Normal())`	`-0.125`
`0.3989 * AffinePushfwd((μ = 3, σ = 4), LebesgueMeasure())`	`-0.92`
`AffinePushfwd((μ = 3, σ = 4), LebesgueMeasure())`	`0.0`
`0.25 * LebesgueMeasure()`	`-1.39`
`LebesgueMeasure()`	`0.0`

Performance

MeasureTheory.jl

```
@benchmark logdensityof(Normal(μ,σ),x) setup=(μ=rand(); σ=rand(); x=rand())
```

```
BenchmarkTools.Trial: 10000 samples with 999 evaluations.
 Range (min … max):  7.348 ns … 246.257 ns  ┊ GC (min … max): 0.00% … 0.00%
 Time  (median):     8.894 ns               ┊ GC (median):    0.00%
 Time  (mean ± σ):   9.412 ns ±   4.303 ns  ┊ GC (mean ± σ):  0.00% ± 0.00%
```

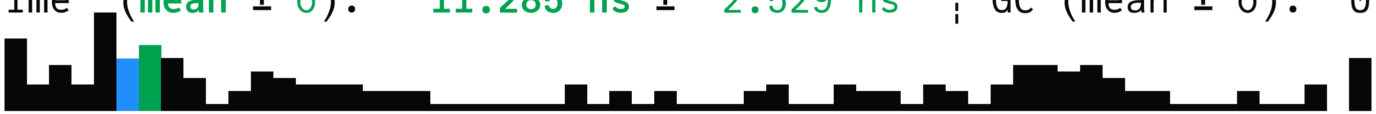


```
  7.35 ns      Histogram: log(frequency) by time      27.2 ns <
 Memory estimate: 0 bytes, allocs estimate: 0.
```

Distributions.jl

```
@benchmark logdensityof(Dists.Normal(μ,σ),x) setup=(μ=rand(); σ=rand(); x=rand())
```

```
BenchmarkTools.Trial: 10000 samples with 999 evaluations.
 Range (min … max):   9.523 ns … 63.201 ns  ┊ GC (min … max): 0.00% … 0.00%
 Time  (median):     10.986 ns              ┊ GC (median):    0.00%
 Time  (mean ± σ):   11.285 ns ±  2.529 ns  ┊ GC (mean ± σ):  0.00% ± 0.00%
```



```
  9.52 ns      Histogram: log(frequency) by time      27.9 ns <
 Memory estimate: 0 bytes, allocs estimate: 0.
```

Thank you!