Diffusion piecewise exponential models for survival extrapolation

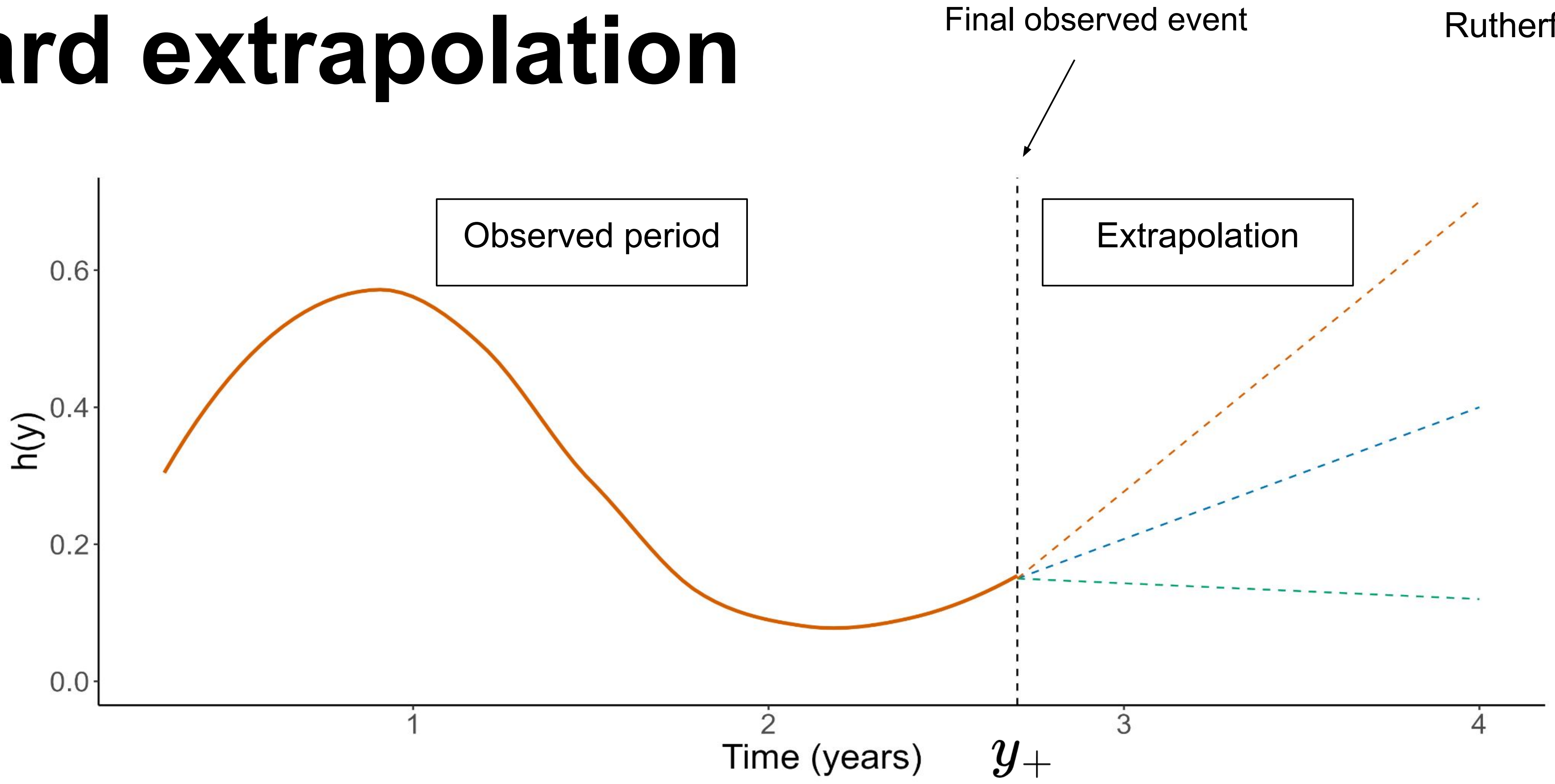
Luke Hardcastle, Samuel Livingstone, Gianluca Baio
Dept. Statistical Science, University College London
luke.hardcastle.20@ucl.ac.uk

9th June 2025
R for Health Technology Assessment Workshop

Extrapolating hazards

Hazard extrapolation

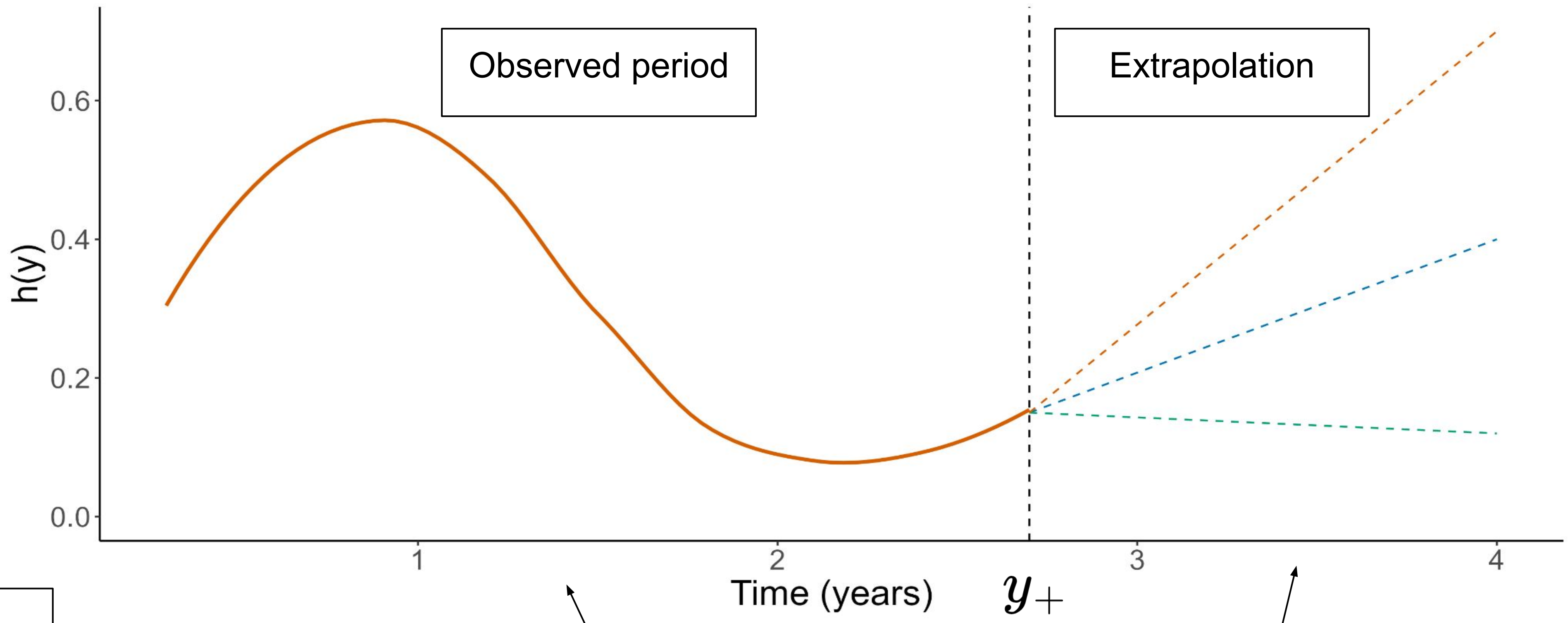
Jackson et al. (2017)
Rutherford et al. (2020)



Hazard extrapolation

Jackson et al. (2017)
Rutherford et al. (2020)

Final observed event



Expected survival - Benefit in economic model

$$\mathbb{E}\left[Y\right] = \int_0^{y_+} \exp\left(-\int_0^y h\left(u\right)du\right)dy + \int_{y_+}^{\infty} \exp\left(-\int_0^y h\left(u\right)du\right)dy$$

Rutherford et al. (2020)

Criteria for principled extrapolation

(See also *Jackson, 2023*)

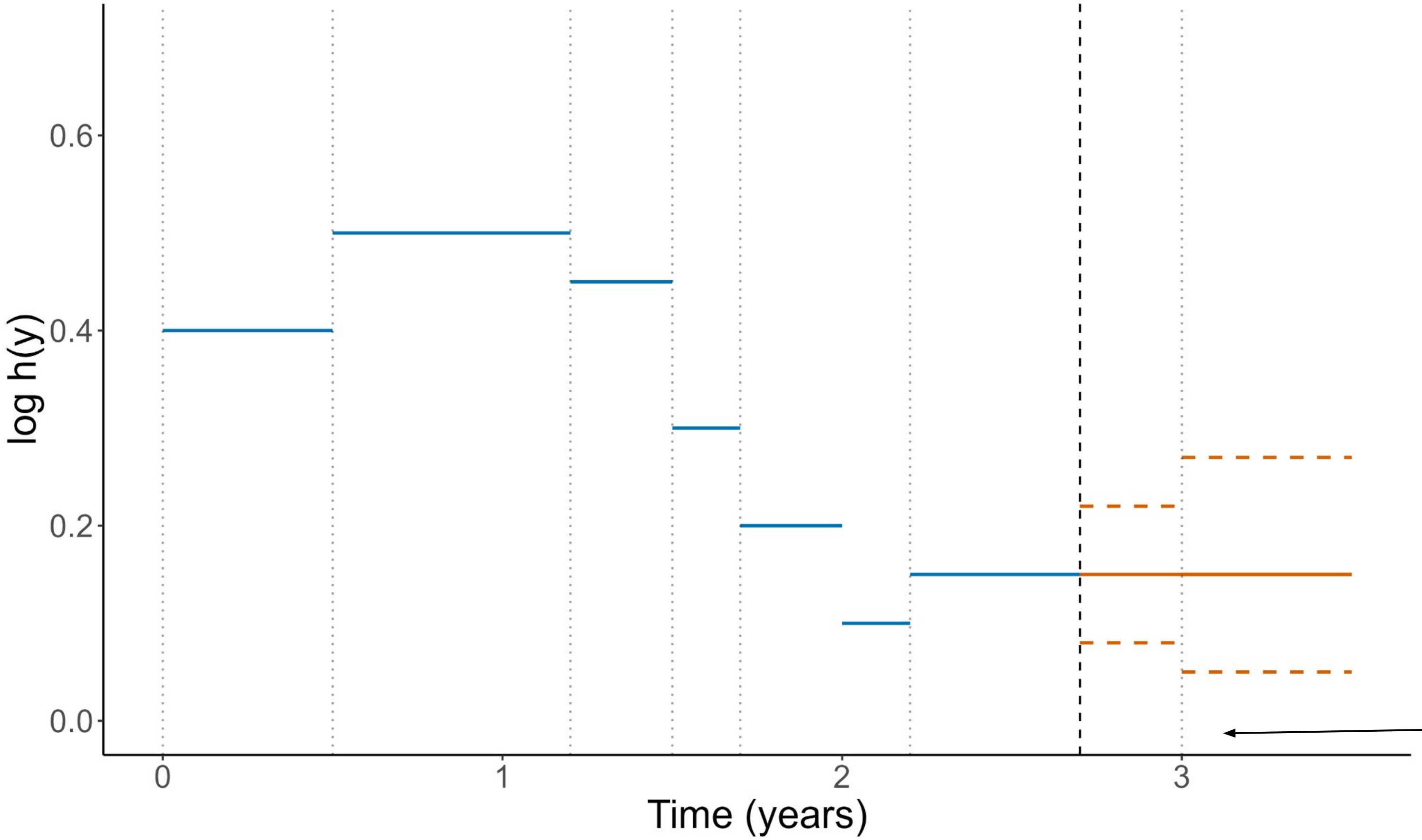
Observation period

- Flexible models - inferences driven by the data
  Recent examples: Dynamics survival models (Kearns et al., (2019)); M-splines (Jackson, 2023); Piecewise Exponential models (Cooney et al., 2023); Polyhazard models (Demiris et al., 2015, Hardcastle et al., 2024).
- Weakly informative prior information

Extrapolation period

- Inherently driven by prior assumptions
- **Note:** This is unavoidable!
- Assumptions should be explicit
- Minimal constraints on the form these assumptions can take

Piecewise exponential models



Local log-hazard

Knots

$$\log h(y) = \sum_{j=1}^{J} \alpha_j \mathbb{1}\left(y \in [s_{j-1}, s_j)\right)$$

Prior dependency for extrapolation, e.g

$$\alpha_j \sim \mathrm{Normal}\left(\alpha_{j-1}, \sigma^2\right)$$

Extrapolations very sensitive to choice of knot location

Diffusion piecewise exponential models

Diffusion piecewise exponential models

$$\log h(y) = \sum_{j=1}^{J} \alpha_j \mathbb{1}\left(y \in [s_{j-1}, s_j)\right)$$

Assume *a priori* that the knots arise from a Poisson Point process:

$$\{s_j\}_{j=1}^{J} \sim PP\left(\gamma, (0, y_+)\right)$$

Idea: Learn volatility of hazard function in extrapolation period from the observation period

Diffusion piecewise exponential models

$$\log h\left(y\right) = \sum_{j=1}^{J} \alpha_j \mathbb{1}\left(y \in \left[s_{j-1}, s_j\right)\right)$$

Assume *a priori* that the knots arise from a Poisson Point process:

$$\left\{s_j\right\}_{j=1}^{J} \sim PP\left(\gamma, \left(0, y_+\right)\right)$$

Idea: Learn volatility of hazard function in extrapolation period from the observation period

Assume *a priori* that local log-hazards $\left\{\alpha_j\right\}_{j=1}^{J}$ are discretisations of a diffusion:

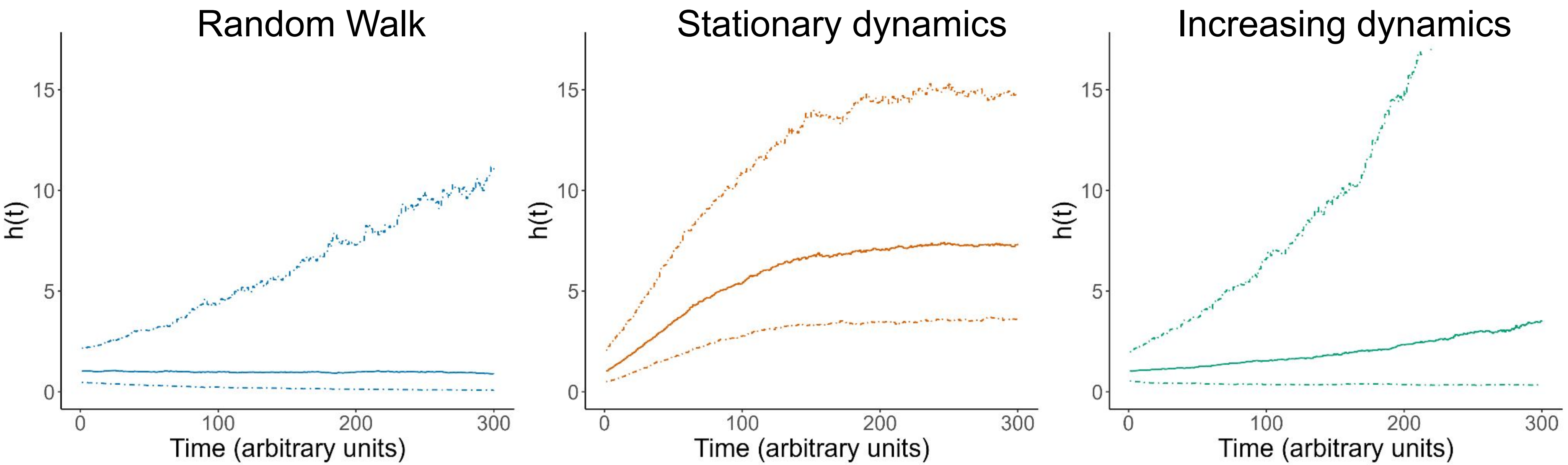
$$d\alpha_{\hat{y}} = \mu\left(\alpha_{\hat{y}}\right) d\hat{y} + \sigma\left(\alpha_{\hat{y}}\right) dW_{\hat{y}}$$

Drift coefficient

Roberts & Sangalli (2010)

Diffusion-based priors

Prior simulations

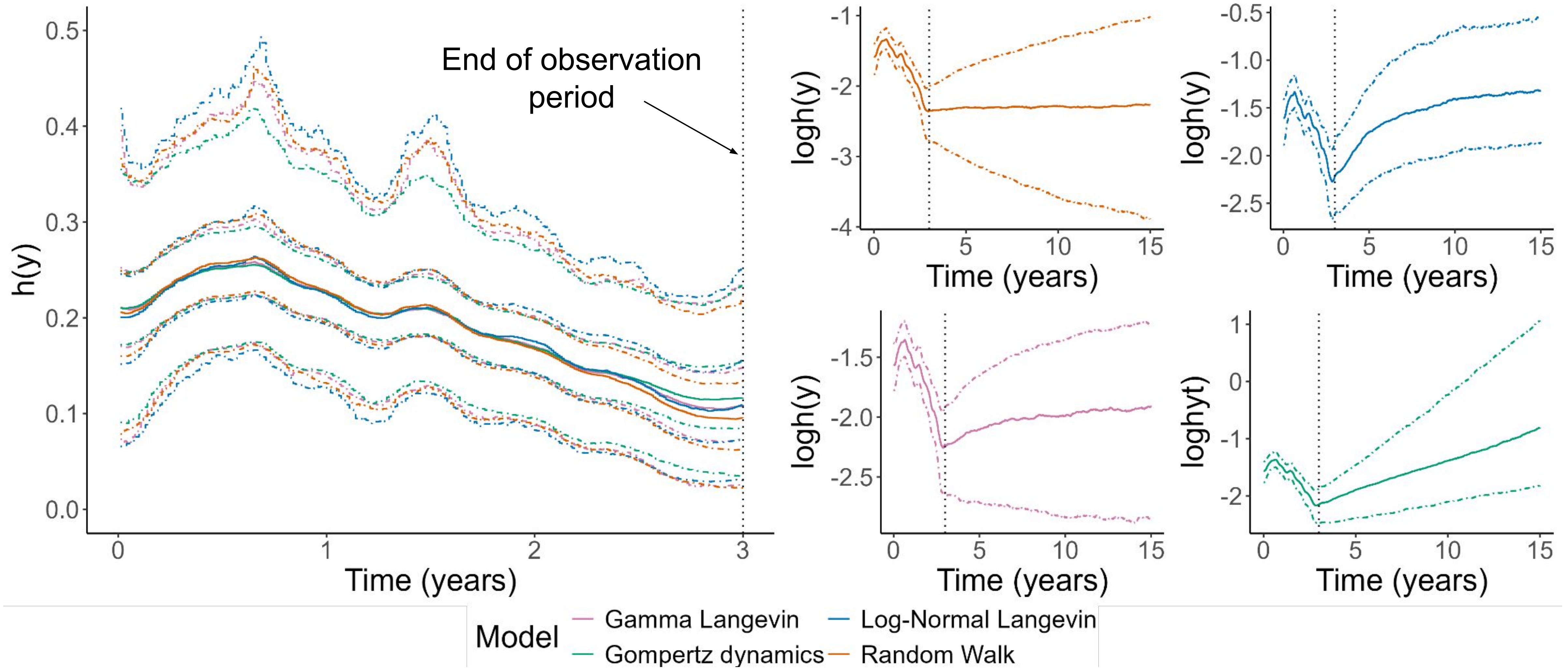


Assume *a priori* that local log-hazards $\{\alpha_j\}_{j=1}^{J}$ are discretisations of a diffusion:

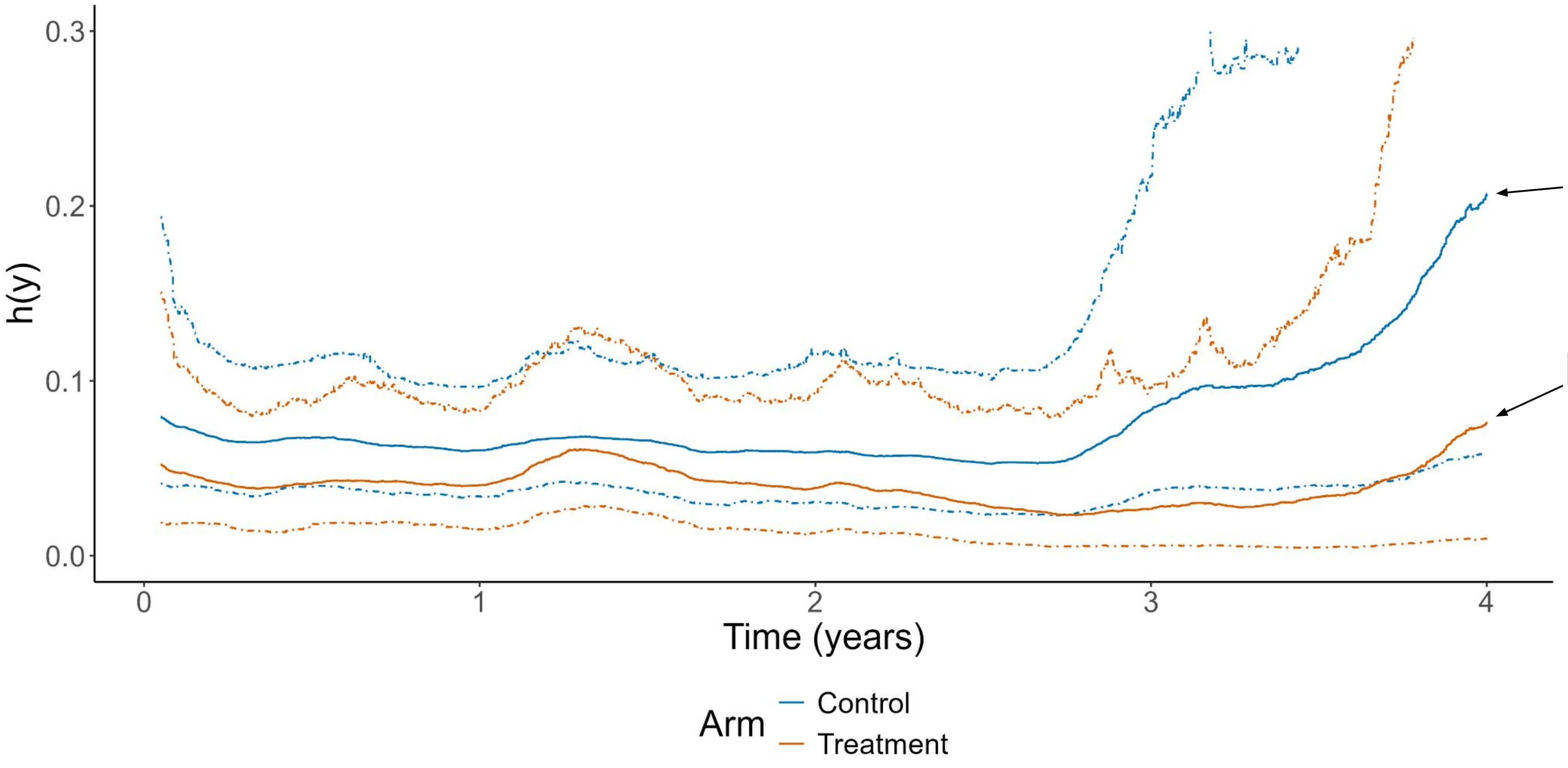
$$d\alpha_{\hat{y}} = \mu\left(\alpha_{\hat{y}}\right)d\hat{y} + \sigma\left(\alpha_{\hat{y}}\right)dW_{\hat{y}}$$

Drift coefficient - contains prior information about long-term hazard

Colon cancer data



Covariates - non-proportional hazards



Flexible baseline hazard

Non-proportional treatment effect

$$d\beta_y = \mu(\beta_y)dy + dW_y$$

Implementation

Fearnhead et al. (2018)

Implementation

- Current state-of-the-art for posterior inference in R is RStan.
- However, Stan cannot handle the prior on the number and location of knots:

$$\{s_j\}_{j=1}^{J} \sim PP\left(\gamma, (0, y_+)\right)$$

- Instead use MCMC scheme based on Piecewise Deterministic Markov Processes
- Computationally efficient with minimal user tuning
- Very new methodology - requires bespoke implementation

Fearnhead et al., (2018)
Bezanson et al., (2017)

Implementation

- Current state-of-the-art for posterior inference in R is RStan.
- However, Stan cannot handle the prior on the number and location of knots:

$$\{s_j\}_{j=1}^{J} \sim PP\left(\gamma, (0, y_+)\right)$$

- Instead use MCMC scheme based on Piecewise Deterministic Markov Processes
- Computationally efficient with minimal user tuning
- Very new methodology - requires bespoke implementation

Julia implementation available via
DiffusionPiecewiseExponential.jl
package

R

- Mature statistical modelling ecosystem - especially for biostatistics
- Fantastic HTA packages

- Slow - very hard/impossible to write high-performance code natively in R
- Speed relies on C++

julia

- Small statistical modelling ecosystem - ML focused
- Almost non-existent HTA ecosystem

- Fast - native code comparable to C++
- Easy to develop in
- Great ecosystem for computational statistics (PPLs, optimisation,...)

- Mature statistical modelling ecosystem - especially for biostatistics
- Fantastic HTA packages

- Slow - very hard/impossible to write high-performance code natively in R
- Speed relies on C++

- Small statistical modelling ecosystem - ML focused
- Almost non-existent HTA ecosystem

- Fast - native code comparable to C++
- Easy to develop in
- Great ecosystem for computational statistics (PPLs, optimisation,...)

Analyse in R

Compute in Julia

Julia implementation

```
# Simulate data
Random.seed!(2352)
n = 100
y = rand(Exponential(1.0), n)
breaks = vcat(0.01,collect(0.26:0.25:1.01))
p = 1
cens = (y .< 1.0)
y[findall(y .> 1.0)] .= 1.0
covar = fill(1.0, 1, n)
dat = init_data(y, cens, covar, breaks)
```

Simulate and format data

```
# Initialise sampler
x0, v0, s0 = init_params(p, dat)
v0 = v0./norm(v0)
t0 = 0.0
state0 = ECMC2(x0, v0, s0, collect(.!s0), breaks, t0, length(breaks), true, findall(s0))
nits = 10_000
nsmp = 10
settings = Splitting(nits, nsmp, 1_000_000, 1.0, 5.0, 0.1, false, true, 0.01, 50.0)
```

MCMC settings

```
# Hazard times for diagnostics and burn in iterations
test_times = collect(0.2:0.2:1.0)
burn_in = 1_000
# Specify the prior
priors = BasicPrior(1.0, PC(1.0, 2, 0.5, Inf),
FixedW([0.5]), 1.0,
CtsPois(7.0, 1.0, 100.0, 1.1), # A Poisson process prior for the knots with intensity 7.0, and maximum knots = 100 on the interval (0.0,1.1)
[GaussLangevin(t -> log(0.29), t-> 0.4)], # A Gaussian stationary distribution for the log-hazard function with mean = log(0.29) and standard deviation = 0.4
[0.1], 2)
```

Specify priors

```
# Run the sampler for two chains
out1 = pem_fit(state0, dat, priors, settings, test_times, burn_in)
```

Run x2 MCMC chains

Julia implementation

```julia
# Simulate data
Random.seed!(2352)
n = 100
y = rand(Exponential(1.0), n)
breaks = vcat(0.01,collect(0.26:0.25:1.01))
p = 1
cens = (y .< 1.0)
y[findall(y .> 1.0)] .= 1.0
covar = fill(1.0, 1, n)
dat = init_data(y, cens, covar, breaks)
```

Simulate and format data

```julia
# Initialise sampler
x0, v0, s0 = init_params(p, dat)
v0 = v0./norm(v0)
t0 = 0.0
state0 = ECMC2(x0, v0, s0, collect(.!s0), breaks, t0, length(breaks), true, findall(s0))
nits = 10_000
nsmp = 10
settings = Splitting(nits, nsmp, 1_000_000, 1.0, 5.0, 0.1, false, true, 0.01, 50.0)
```

MCMC settings

```julia
# Hazard times for diagnostics and burn in iterations
test_times = collect(0.2:0.2:1.0)
burn_in = 1_000
# Specify the prior
priors = BasicPrior(1.0, PC(1.0, 2, 0.5, Inf),
FixedW([0.5]), 1.0,
CtsPois(7.0, 1.0, 100.0, 1.1), # A Poisson process prior for the knots with intensity 7.0, and maximum knots = 100 on the interval (0.0,1.1)
[GaussLangevin(t -> log(0.29), t-> 0.4)], # A Gaussian stationary distribution for the log-hazard function with mean = log(0.29) and standard deviation = 0.4
[0.1], 2)
```

Specify priors

CtsPois(7.0, …., 1.1)

- Poisson process prior with intensity 7 on the interval (0,1.1)

GaussLangevin(t -> log(0.29), t-> 0.4)

- A $\mathrm{Normal}(\log(0.29), 0.4)$ stationary distribution for the log-hazard

```julia
# Run the sampler for two chains
out1 = pem_fit(state0, dat, priors, settings, test_times, burn_in)
```

Run x2 MCMC chains

R implementation

```r
library(JuliaCall) # Package for calling Julia from R
julia_setup() # Package initialisation
julia_library("DiffusionPiecewiseExponential")
julia_library("Distributions")
julia_library("Random")
julia_library("LinearAlgebra")

set.seed(123)
n = 100
y = rexp(n, 1)
cens = as.numeric(y < 1)
for(i in 1:n){
  y[i] = ifelse(cens[i] == 1, y[i], 1)
}
julia_assign("n", as.integer(n))
julia_assign("y", y)
julia_assign("cens", cens)
julia_command("breaks = vcat(0.01,collect(0.26:0.25:1.01))")
julia_command("p = 1")
julia_command("covar = fill(1.0, 1, n)")

julia_source("Setup.jl")

julia_command("priors = BasicPrior(1.0, PC(1.0, 2, 0.5, Inf), FixedW([0.5]), 1.0,
CtsPois(7.0, 1.0, 30.0, 1.1),
[GaussLangevin(t -> log(0.29), t-> 0.4)],
[0.1], 2)")
julia_command("julia_output = pem_fit(state0, dat, priors, settings, test_times, burn_in)")
R_output = julia_eval("julia_output")
```

R implementation uses JuliaCall package to call Julia from R

Load julia packages

- Generate/format data in R
- Assign data in julia using “julia_assign”

- “julia_command” runs julia commands from R
- Return model output to R using julia_eval

References

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM review*, 59(1), 65-98.

Demiris, N., Lunn, D., and Sharples, L. D. (2015). Survival extrapolation using the poly-Weibull model. *Statistical Methods in Medical Research*, 24(2): 287–301.

Fearnhead, P., Bierkens, J., Pollock, M., & Roberts, G. O. (2018). Piecewise Deterministic Markov Processes for Continuous-Time Monte Carlo. *Statistical Science*, *33*(3), 386-412.

Hardcastle, L., Livingstone, S., Baio, G. Averaging polyhazard models using Piecewise deterministic Monte Carlo with applications to data with long-term survivors. 2024, arxiv:2406.14182

Jackson, C. H. (2023). survextrap: a package for flexible and transparent survival extrapolation. *BMC Medical Research Methodology*, 23(1): 282.

Jackson, C., Stevens, J., Ren, S., Latimer, N., Bojke, L., Manca, A., & Sharples, L. (2017). Extrapolating survival from randomized trials using external data: a review of methods. *Medical decision making*, *37*(4), 377-390.

Roberts, G. O., & Sangalli, L. M. (2010). Latent diffusion models for survival analysis. *Bernoulli*, *16*(2), 435-458.

Rutherford, M. J., Lambert, P. C., Sweeting, M. J., Pennington, R., Crowther, M. J., & Abrams, K. R. (2020). NICE DSU technical support document 21: flexible methods for survival analysis. *Decision Support Unit, ScHARR, University of Sheffield*.

Discussion

Summary

- Introduced the **Diffusion Piecewise Exponential model** for survival extrapolation
- Extrapolations are driven through an underlying diffusion and a prior on knot locations
- Provides flexible, data-driven inferences for the observation period
- Extrapolations driven by explicit prior assumptions
- Implementation in julia can be called via R: DiffusionPiecewiseExponential.jl

Additional Points

- Several possibilities for long-term drifts
- Currently improving elicitation procedures
- R package in the works - julia code completely hidden
- Preprint available now:

Diffusion piecewise exponential models for survival extrapolation using Piecewise Deterministic Monte Carlo.
Hardcastle, L., Livingstone, S., & Baio, G. (2025).
arXiv preprint arXiv:2505.05932.

Discussion

Summary

- Introduced the **Diffusion Piecewise Exponential model** for survival extrapolation
- Extrapolations are driven through an underlying diffusion and a prior on knot locations
- Provides flexible, data-driven inferences for the observation period
- Extrapolations driven by explicit prior assumptions
- Implementation in julia can be called via R: DiffusionPiecewiseExponential.jl

Additional Points

- Several possibilities for long-term drifts
- Currently improving elicitation procedures
- R package in the works - julia code completely hidden
- Preprint available now:

Diffusion piecewise exponential models for survival extrapolation using Piecewise Deterministic Monte Carlo.
Hardcastle, L., Livingstone, S., & Baio, G. (2025).
arXiv preprint arXiv:2505.05932.

Thank you!