000_Overview

November 25, 2024

Oreum Industries Internal Project, 2024Q3

---

1 000_Overview.ipynb

1.1 Oreum Case Studies - ONS Deaths Survival Regression `oreum_cs_ons`

Demonstrate Survival Regression Modelling using Bayesian inference and a Bayesian workflow, specifically using the `pymc` & `arviz` ecosystem.

PDF version

This is very brief overview of a full case study `oreum_cs_ons` in which we demonstrate an E2E workflow for survival regression models of increasing sophistication.

This overview is for verbal presentation and quick discussion purposes only, and ideally should accompany a deeper technical walkthrough of the case study in a long-form style. There we evaluate the behaviour and performance of the models throughout the workflows, including several state-of-the-art methods unavailable to conventional max-likelihood / machine-learning models.

We use a complicated, real-world dataset: the ONS England & Wales Deaths 2022 reference tables:

```
Death registrations by single year of age, (males | females),
England and Wales, registered 1963 to 2022
```

This dataset requires multiple advanced modelling methods to handle and mitigate bad / messy data, and in particular a very unusual **right-truncation** due to the ONS only publishing aggregated death counts. i.e this is not an observational study of cohorts / individuals from a specified start-time, instead we only learn about individuals when we record a death!

This has several implications on the choice of model, which we discuss in detail in §1 **Model Architecture**, to summarise: + We use a parametric distribution to estimate the event density function $\pi(t)$ as a Gompertz PDF, reparameterised using the **modal-age-of-death** + We regress this distribution onto features in the data + The model likelihood is evaluated using a count-based distribution (we try Poisson and NegBinomial) compared to the overall deaths for a naive observation-year cohort.

Our General Model Architecture in this project is a modified Accelerated Failure Time (AFT) model. See comprehensive explanations and deep technical demonstrations of various survival mod-

els and explanations of censoring in our public reference project oreum_survival which includes Accelerated Failure Time and Piecewise Regression Models.

1.2 Contents

---

2 Setup

2.1 Imports

```
from pathlib import Path

from oreum_core import curate, eda
from pyprojroot.here import here
```

Notebook config

```
%matplotlib inline
%config InlineBackend.figure_format = 'retina'
```

2.2 Data Connections and Helper Objects

```
ppqio_cleaned = curate.PandasParquetIO(here(Path('data', 'raw', 'cleaned')).
 ↪resolve(strict=True))
ppqio_prepared = curate.PandasParquetIO(here(Path('data', 'prepared')).
 ↪resolve(strict=True))
figio = eda.FigureIO(here(Path('plots')).resolve(strict=True))
```

---

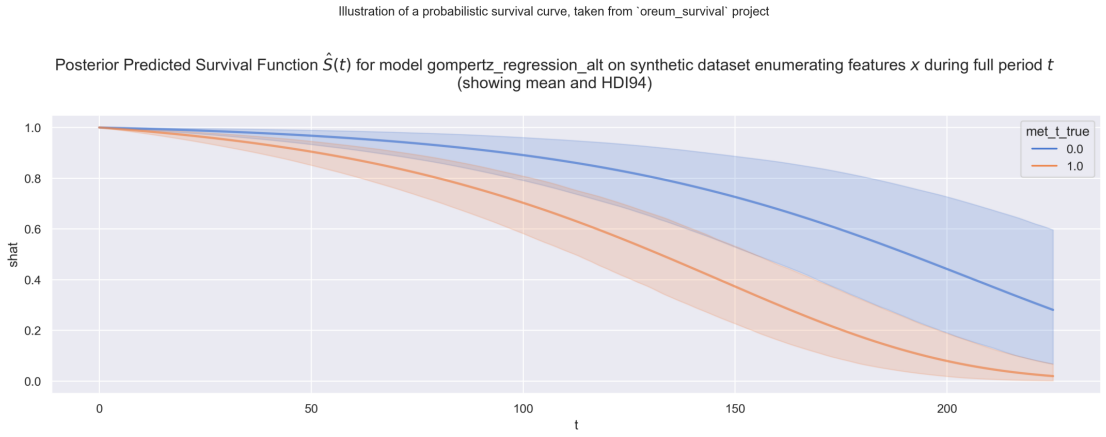
3 Preamble

3.1 What is Survival Regression?

See comprehensive explanations and deep technical demonstrations in our public project oreum_survival

Essentially, we seek to create *principled* models that provide explanatory inference and predictions of the Survival Function $\hat{S}(t)$ and Expected Time-to-Event $\hat{\mathbb{E}}_t$ with quantified uncertainty to support real-world decision-making.

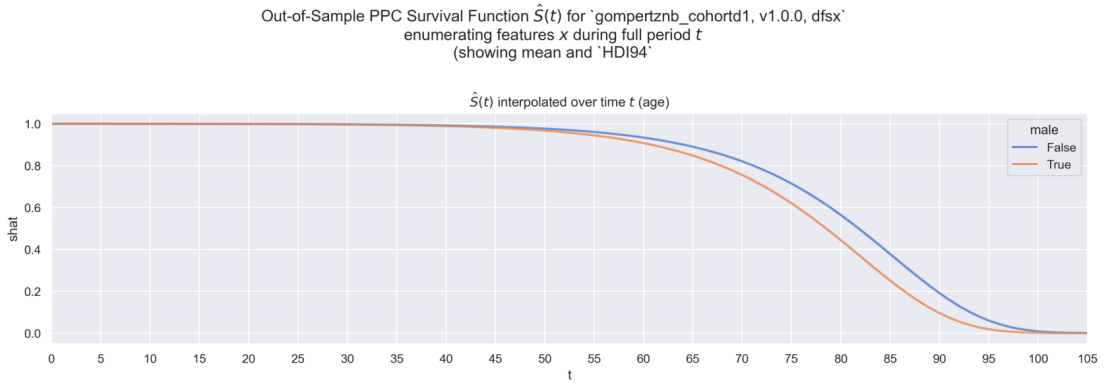
```
f = figio.read(fn='../assets/img/
 ↪001_gompertz_regression_alt_forecast_survival_function.png',
        title=f'Illustration of a probabilistic survival curve, taken from␣
 ↪`oreum_survival` project',
        figsize=(16, 6))
```

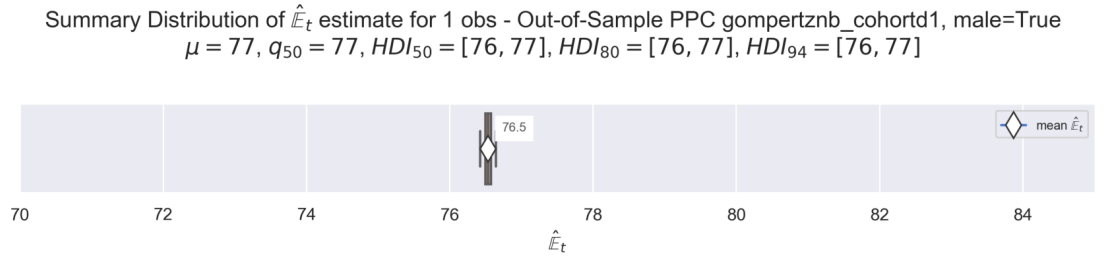


3.1.1 Illustrative Survival Function $S(t)$ and Expected Time-to-Event $\mathbb{E}_t$ from this Project `oreum_cs_ons`

Taken from notebook `oreum_cs_ons` `311_GompertzNB_CohortD1.ipynb` and shown here just to give the reader a feel for the outputs and the problem-space. All estimates are made with quantified uncertainty, which reveals (rather than hides) the inherent noise in real-world data and processes.

```
f = figio.read(fn='311_gompertznb_cohortd1_v1_0_0_dfsx_forecast_shat.png',␣
 ↪figsize=(12, 10))
f = figio.read(fn='311_gompertznb_cohortd1_v1_0_0_dfsx_forecast_et_male_true.
 ↪png')
```





3.2 We gain massive advantage by using a Bayesian Framework

We specifically use **Bayesian Inference** rather than Frequentist Max-Likelihood methods for many reasons, including:

	Bayesian Inference	Frequentist Max-Likelihood
General formulation → Desirable Trait ↓	*Bayes' Rule* $\underbrace{P(\hat{\mathcal{H}}\|D)}_{\text{posterior}} = \frac{\overbrace{P(D\|\mathcal{H})}^{\text{likelihood}} \cdot \overbrace{P(\mathcal{H})}^{\text{prior}}}{\underbrace{P(D)}_{\text{evidence}}}$	*MLE* $\hat{\mathcal{H}}^{\text{MLE}} \propto \arg\max_{\mathcal{H}} P(D\|\mathcal{H})$
***Principled* model structure represents hypothesis about the data-generating process**	**Very strong** Can build bespoke arbitrary and hierarchical structures of parameters to map to the real-world data-generating process.	**Weak** Can only state structure under strict limited assumptions of model statistical validity.
Model parameters and their initial values represent domain expert knowledge	**Very strong** Marginal prior distributions represent real-world probability of parameter values before any data is seen.	**Very weak** No concept of priors. Lack of joint probability distribution can lead to discontinuities in parameter values.

	Bayesian Inference	Frequentist Max-Likelihood
Robust parameter fitting process	**Strong** Estimate full joint posterior probability mass distribution for parameters - more stable and representative of the expectation for the parameter values. Sampling can be a computationally expensive process.	**Weak** Estimate single-point max-aposterioi-likelihood (density) of parameters - this can be far outside the probability mass and so is prone to overfitting and only correct in the limit of infinite data. But optimization method can be computationally cheap.
Fitted parameters have meaningful summary statistics for inference	**Very strong** Full marginal probability distributions can be interpreted exactly as probabilities.	**Weak** Point estimates only have meaningful summary statistics under strict limited assumptions of model statistical validity.

continues ...

... continued

Desirable Trait	Bayesian Inference	Frequentist Max-Likelihood
Robust model evaluation process	**Strong** Use entire dataset, evaluate via Leave-One-Out Cross Validation (best theoretically possible).	**Weak** Cross-validation rarely seen in practice, even if used, rarely better than 5-fold CV. Simplistic method can be computationally cheap.
Predictions made with quantified variance	**Very strong** Predictions made using full posterior probability distributions, so predictions have full empirical probability distributions.	**Weak** Predictions using point estimates can be bootstrapped, but predictions only have interpretation under strict limited assumptions of model validity.
Handle imbalanced, high cardinality & hierarchical factor features	**Very strong** Can introduce partial-pooling to automatically balance factors through hierarchical priors.	**Weak** Difficult to introduce partial-pooling (aka mixed random effects) without affecting strict limited assumptions of model validity.
Handle skewed / multimodal / extreme value target variable	**Very strong** Represent the model likelihood as any arbitrary probability distribution, including mixture (compound) functions e.g. a zero-inflated Weibull.	**Weak** Represent model likelihood with a usually very limited set of distributions. Very difficult to create mixture compound functions.

Desirable Trait	Bayesian Inference	Frequentist Max-Likelihood
Handle small datasets	**Very strong** Bayesian concept assumes that there is a probable range of values for each parameter, and that we evidence our prior on any amount of data (even very small counts).	**Very weak** Frequentist concept assumes that there is a single true value for each parameter and that we only discover that value in the limit (of infinite observations).
Automatically impute missing data	**Very strong** Establish a prior for each datapoint, evidence on the available data within the context of the model, to automatically impute missing values.	**Very weak** No inherent method. Usually impute as a pre-processing step with weak non-modelled methods.

3.3 Practical Implementations of Bayesian Inference

We briefly referenced *Bayes Rule* above, which is a useful mnemonic when discussing Bayesian Inference, but in practice the crux of putting these advanced statistical techniques into practice is estimating the evidence $P(D)$ i.e. the probability of observing the data that we use to evidence the model

$$\underbrace{P(\hat{\mathcal{H}}|D)}_{\text{posterior}} = \frac{\overbrace{P(D|\mathcal{H})}^{\text{likelihood}} \cdot \overbrace{P(\mathcal{H})}^{\text{prior}}}{\underbrace{P(D)}_{\text{evidence}}}$$

...where:

$$P(D) \sim \int_{\Theta} P(D, \theta)\ d\theta$$

This joint probability $P(D, \theta)$ of data D and parameters θ requires an almost impossible-to-solve integral over parameter-space Θ. Rather than attempt to calculate that integral, we do something that sounds far more difficult, but given modern computing capabilities is actually practical.

3.3.1 We use a Bleeding-edge MCMC Toolkit for Bayesian Inference: `pymc` & `arviz`

We use **Markov Chain Monte-Carlo (MCMC)** sampling to take a series of *ergodic*, *partly-reversible*, *partly-randomised* samples of model parameters θ, and at each step compute the ratio of log-likelihoods $\log P(D|\mathcal{H})$ between a starting position (current values) θ_{p0} and proposed "sampled" position θ_p in parameter space, so as to reduce that log-likelihood (whilst exploring the parameter space).

This results in a posterior estimate $P(\hat{\theta}|D)$:

$$P(\hat{\theta}|D) \sim \frac{\overbrace{P(D|\theta_p)}^{\text{likelihood @ proposal}} \cdot \overbrace{P(\theta_p)}^{\text{prior @ proposal}}}{\underbrace{P(D|\theta_{p0})}_{\text{likelihood @ current}} \cdot \underbrace{P(\theta_{p0})}_{\text{prior @ current}}}$$

This is the heart of MCMC sampling: for detailled practical explanations see Betancourt, 2021 and Tweicki, 2015

We use the bleeding-edge `pymc` and `arviz` Python packages to provide the full Bayesian toolkit that we require, including advanced sampling, probabilistic programming, statistical inferences, model evaluation and comparison, and more.

```
f = figio.read(fn='../assets/img/logos',figsize=(12, 2))
```

---

4 1. The Dataset and Problem-Space

4.1 1.1 Dataset

As noted above, we use a complicated, real-world dataset: the ONS England & Wales Deaths 2022 reference tables:

```
Death registrations by single year of age, (males | females),
England and Wales, registered 1963 to 2022
```

4.1.1 For illustration: table of the dataset, verbatim as supplied

```
df = ppqio_cleaned.read('deaths_2022')
eda.display_ht(df)
```

		death_m_ct	death_f_ct
yr	age_at_death		
1963-01-01	0	10401	7641
	1	665	537
	2	378	288
2022-01-01	103	90	493

```
           104                   52         268
           105                   39         385

'Shape: (6360, 2), Memsize 0.1 MB'
```

Post-cleaning (pre-preparation)

Explanation of the features for each of the 6360 raw observations:

```
yr:             year of recorded death
age_at_death:   age at death (rounded down to birthday passed)
death_m_ct:     summed count of deaths for `sex=male`
death_f_ct:     summed count of deaths for `sex=female`
```

4.1.2 For illustration: table of the dataset (post-preparation)

See `oreum_cs_ons` `100_Curate.ipynb` for more detail

```
df = ppqio_prepared.read('deaths_2022')
eda.display_ht(df)
```

```
                           age_at_death     cohort_b5    cohort_b10  \
birth_yr   death_yr   male
1858-01-01 1963-01-01 False           105  1855<=b<1860  1850<=b<1860
                      True            105  1855<=b<1860  1850<=b<1860
1859-01-01 1963-01-01 False           104  1855<=b<1860  1850<=b<1860
2021-01-01 2022-01-01 True              1  2020<=b<2025  2020<=b<2030
2022-01-01 2022-01-01 False             0  2020<=b<2025  2020<=b<2030
                      True              0  2020<=b<2025  2020<=b<2030

                           death_ct
birth_yr   death_yr   male
1858-01-01 1963-01-01 False        16
                      True          1
1859-01-01 1963-01-01 False        14
2021-01-01 2022-01-01 True         75
2022-01-01 2022-01-01 False      1027
                      True       1384

'Shape: (12720, 4), Memsize 0.4 MB'
```

Post-preparation

Explanation of the features for each of the 12720 prepared observations:

Identifier features (from which we derive exogenous features for modelling use)

```
birth_year:     Reverse-engineered birth year based on death_year - age-at-death
death_year:     death_year as supplied
```

Endogenous (target) features

```
death_ct:       Summed count of deaths
```

Exogenous (predictor) features (some are derived / prepared)

```
male:           Sex of group is male (True/False) (this has been melted from raw)
age_at_death:   age_at_death as supplied
cohort_b5:      Prepared aggregated birth-year cohort with bins of 5-year span
cohort_b10:     Prepared aggregated birth-year cohort with bins of 5-year span
```

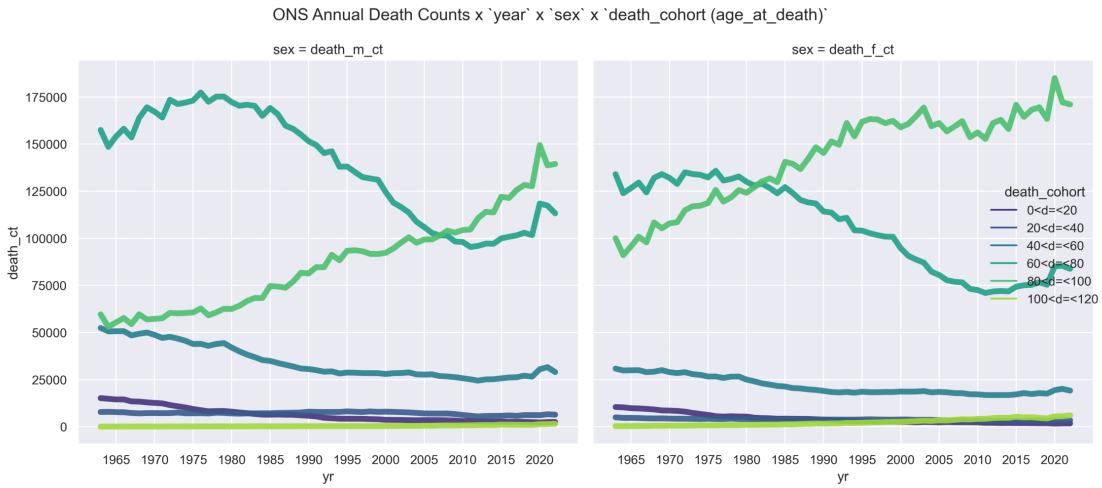
4.2 1.2. Aggregated Death Counts, Right-Truncation, and Event Density π

In the full case study `oreum_cs_ons` `300_ModelArchitecture` §1 we discuss in detail the issues of this dataset and the impacts it has on the model architecture(s) available to us.

4.2.1 Aggregated death counts over time

The data is only supplied in aggregate form, so we don't have lifetime time-to-event t

```
f = figio.read(fn='100_ons_deaths_2022.png', figsize=(16, 10))
```



Observe

- These `death_cohorts` are not used in the model, they're just to condense the data little so we can see general patterns
- The most noticeable trend is a gentle raise and then decrease in the count of deaths in the `60<d<=80` group, matched by a negatively-correlating rise in count of deaths in the `80<d<=100` group: over time, people are dying later
- We also see males are dying sooner than females
- Note we observe deaths from 1963 onwards, due to ONS data availability

See `oreum_cs_ons` `100_Curate.ipynb` §3 for more detail

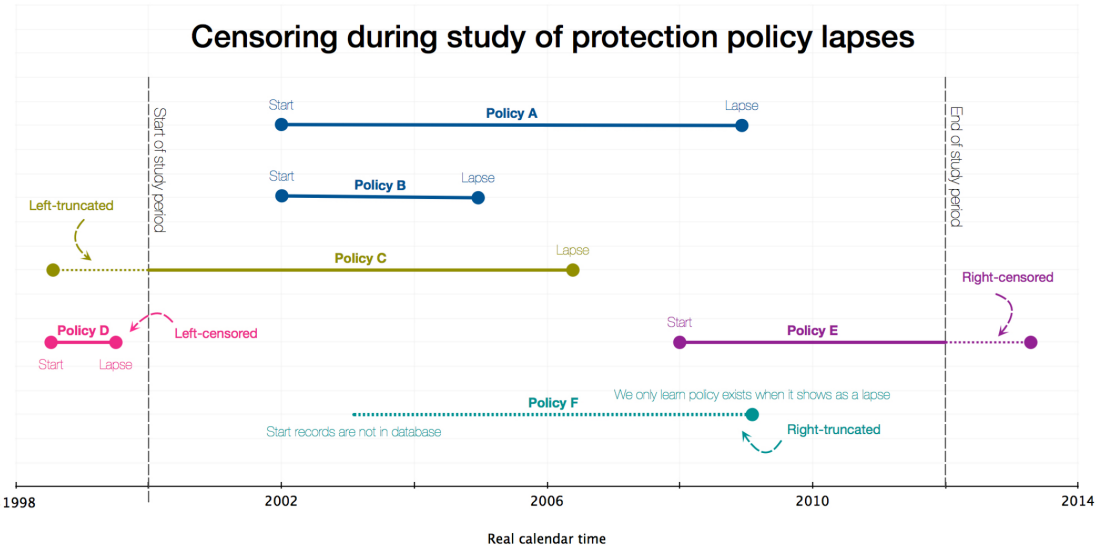
4.2.2 Right-Truncation

This dataset involves a very unusual **right-truncation** due to the ONS only publishing aggregated death counts. i.e this is not an observational study of cohorts / individuals from a specified start-time, instead we only learn about individuals when we record a death!

See reference project `oreum_survival` `000_Intro.ipynb` for detail and helpful illustrations of censoring

```
f = figio.read(fn='../assets/img/censoring', extension = '.jpeg',
        title=f'Recap illustration of censoring and trunction types. In particular, note our dataset suffers right-truncation',
        figsize=(16, 8)) #, fontsize=16)
```

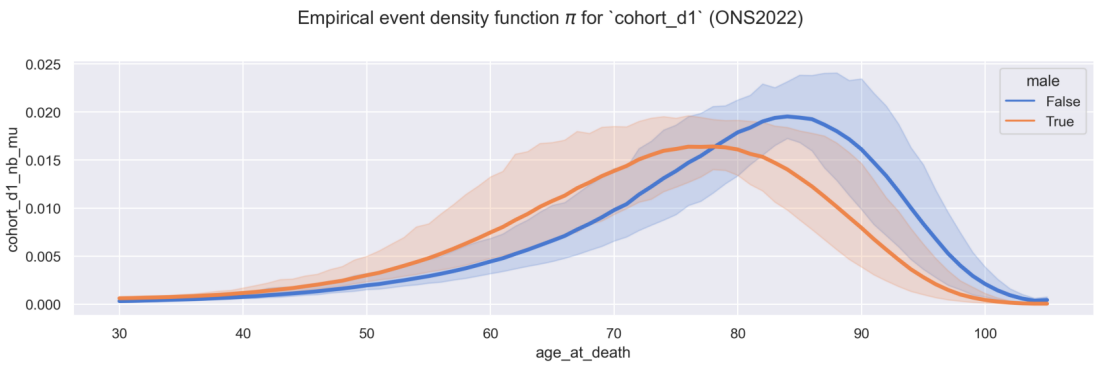
Recap illustration of censoring and trunction types. In particular, note our dataset suffers right-truncation



4.2.3 Empirical event density π

We can still calculate a simple empirical event density *pi* as used in the AFT models:

```
f = figio.read(fn='310_empirical_pi_cohortd1', figsize=(12, 12))
```



Observe:

- Our empirical π is quite well defined, and behaves as expected, with a mode M well into old age
- Lots of variance around the modal-age-of-death M, which is understandable given we have a lot of observations at this age (with births back to 1880s)

See `oreum_cs_ons` `310_GompertzPoisson_CohortD1.ipynb` §1 for more detail

4.3 1.3 Modelled Survival Function $\hat{S}(t)$ and Expected Time-to-Event $\hat{\mathbb{E}}_t$

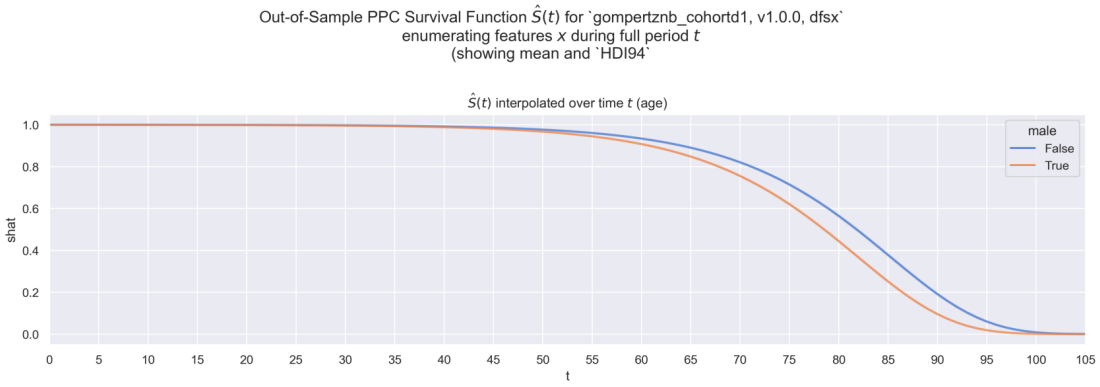
We seek to estimate π and thus the **Expected Time-to-Event** $\hat{\mathbb{E}}_t$

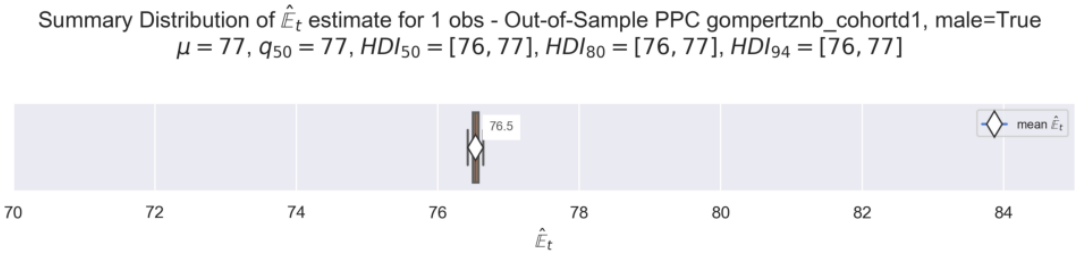
As a side-effect of the mathematical relationships, we also get a **Survival Function** $\hat{S}(t)$ which can be a more familiar representation for newcomers to understand.

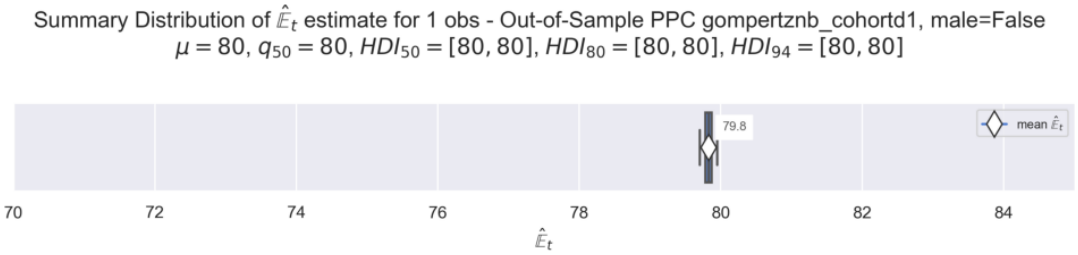
The model produces these estimates with quantified uncertainty **per individual**, but we can substitute a simplified dataset to effectively roll these up to look at the differences between groups according to exogenous feature.

4.3.1 For example, forecasted Expected Time-to-Event $\hat{\mathbb{E}}_t$ and Survival Function $\hat{S}(t)$ for male vs female:

```
f = figio.read(fn='311_gompertznb_cohortd1_v1_0_0_dfsx_forecast_shat.png',␣
 ↪figsize=(12, 10))
f = figio.read(fn='311_gompertznb_cohortd1_v1_0_0_dfsx_forecast_et_male_true.
 ↪png', figsize=(12, 2))
f = figio.read(fn='311_gompertznb_cohortd1_v1_0_0_dfsx_forecast_et_male_false.
 ↪png', figsize=(12, 2))
```







---

5 2. The Modelling Work

5.1 2.1 Architectures

We use an **Accelerated Failure Time (AFT) model**, specifically an **uncensored Gompertz-distribution on the Event Density π, evidenced with a Count distribution κ on aggregated deaths**.

The AFT base model is explained in great detail with examples in our public reference project `oreum_survival` `001_BayesianSurvival_AFT.ipynb`

As noted above, the data here is **right-truncated** which necessitates a novel model architecture to handle it. We describe this in great detail in `oreum_cs_ons` `300_ModelArchitecture.ipynb` and we will happily explain that interactively to interested folk.

5.1.1 For illustration: math snippet and plate notation diagram of the novel `Gompertz-NegBinomial` model

The key insight noted by Monica Alexander in her proposed model is that we *can* still evaluate the actual observed death count κ_c compared to a proposed parametric event density function $\pi(t)$ and the total count of deaths $\mathcal{D}_c$ for the relevant cohort (see §1.4 for more discussion on $\mathcal{D}_c$). This doesn't require us to have d_i available for the likelihood.

See `oreum_survival` `202_AFT_GompertzAlt.ipynb` for detail of the `GompertzAlt` AFT architecture, and illustrations of the Gompertz Alt parameterization in use for π, and the associated λ, Λ, and S.

Here again substituting $\eta = \exp(-\gamma M)$, and also omitting the machinery to handle right-censoring (because we can't use it here), we see:, and using a Poisson count likelihood, we can continue the AFT model spec as:

$$\pi(t) = \lambda(t) \cdot S(t)$$
$$\theta_c(t) = \mathcal{D}_c \cdot \pi(t)$$

$$\hat{\kappa}_c(t) \sim \text{Poisson}(\theta_c(t))$$

where: + $\mathcal{D}_c$ is the total count of events (death) per cohort

5.1.2 Very brief jumping-off point for discussion on Cohorts

To create $\mathcal{D}_c$ we *could* simply sum each observation year $\mathcal{D}_y = \sum_y \kappa_y(t)$

However, within an observation year y, each value $\kappa_y(t = 0, 1, 2, ..., n)$ is sourced from people with different birth years, and if we sum over them, we will smooth out cohort effects due to birth year and their lived experiences in real time. i.e. real-world events will "flow through" the t's. This seems like a mistake, and also a missed opportunity to include birth cohorts.

The practical impact is: + During model development we will experiment with increasing the fidelity of D_c, from $y \rightarrow b$ + We will incorporate these indicator variables as part of the regression submodel on $M \sim \exp(\beta \mathbf{x})$

Cohort options

(A) Unpooled death year cohort To create D_c we could simply sum over each observation year death D_y, which seems a little backwards, but could yield M varying by observed year of death, letting us make statements about years with high mortality (useful?)

$$D_{d1} = \sum_{d1} \kappa_{d1}(t)$$

Cohorts with a single 1-year resolution: models `GompertzPoissonCohortD1` and `GompertzNBCohortD1`

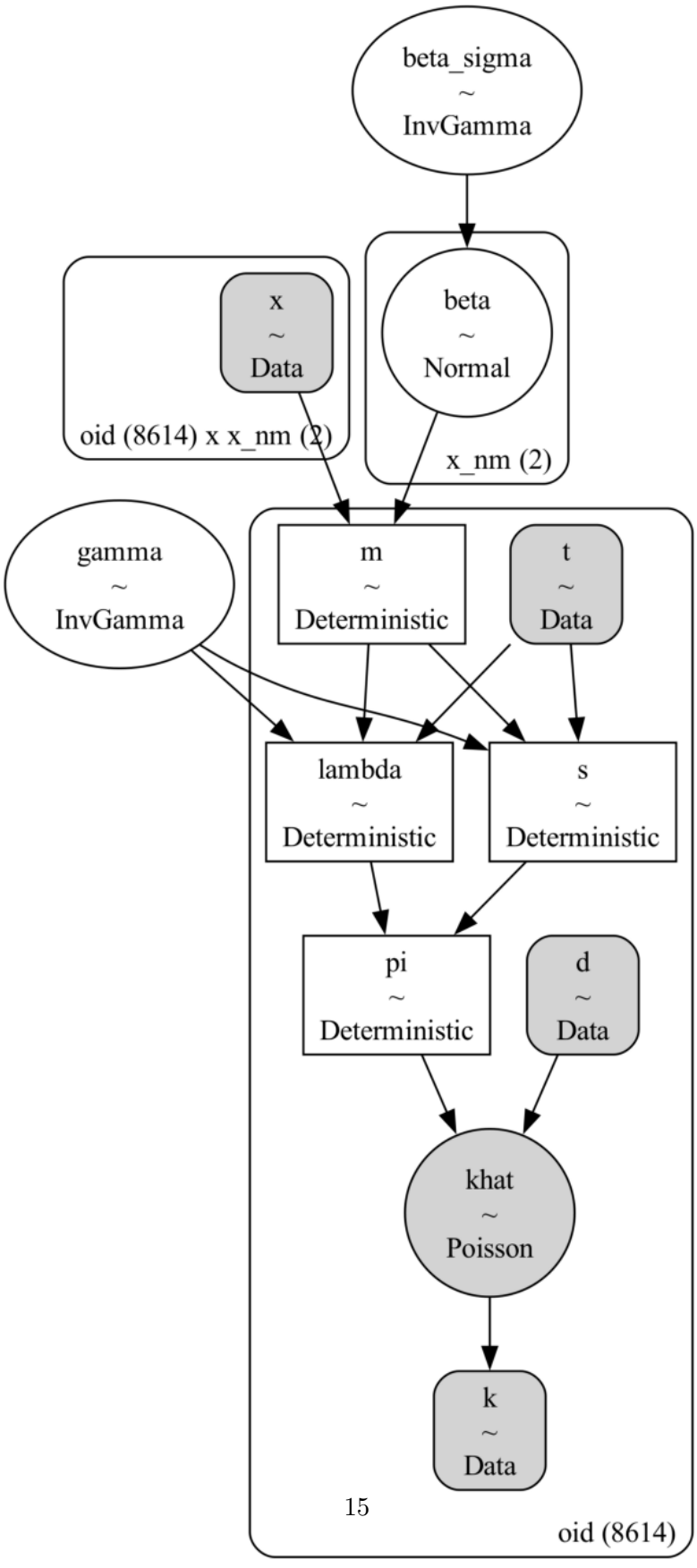
(B) Partial-pooled birth year cohort This is our initially-preferred option for $\sum_c \kappa_c(t)$, to yield M varying by birth cohort. This seems closer to a real observational study where we have birth dates by individual

$$D_{b10} = \sum_{b10} \kappa_{b10}(t)$$

Cohorts with a 10-year resolution: models `GompertzNBCohortB10` etc

5.1.3 Plate notation of simplest model: `GompertzPoisson`

```
f = figio.read(fn='../data/models/graph_gompertzpoisson_cohortd1_v101_dfx.png',
        title=f'GompertzPoisson_CohortD1', figsize=(12, 10))
```

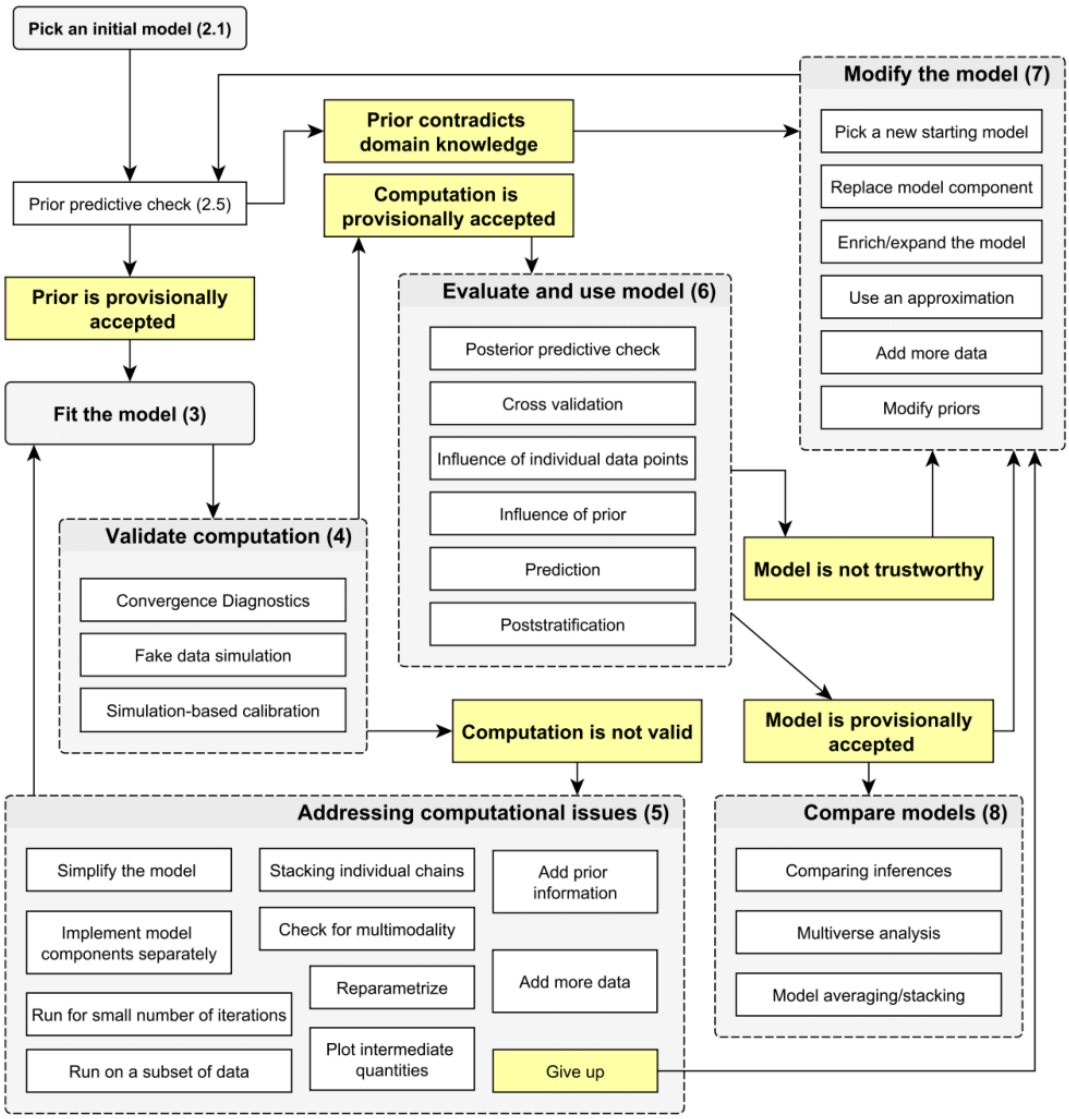

GompertzPoisson_CohortD1
beta_sigma
~
InvGamma
x
~
Data
oid (8614) x x_nm (2)
beta
~
Normal
x_nm (2)
gamma
~
InvGamma
m
~
Deterministic
t
~
Data
lambda
~
Deterministic
s
~
Deterministic
pi
~
Deterministic
d
~
Data
khat
~
Poisson
k
~
Data
oid (8614)

5.2 2.2 Bayesian Workflow

Throughout the case study we employ at reasonably complete **Bayesian Workflow** (see Gelman et al, 2020 and Betancourt, 2020) using a cyclical process of model evaluation and improvement.

```
f = figio.read(fn='../assets/img/gelman_2020_fig1.png', title=f'Gelman et al.,␣
 ↪2000 - Figure 1 - Illustration of Bayesian Workflow',
figsize=(12, 10))
```

Gelman et al., 2000 - Figure 1 - Illustration of Bayesian Workflow



5.3 2.3 The Model Variants and Technical Evaluation

In this particular case study we ended up with 2 model variants based on the count distribution, and further differences based on the death count cohort

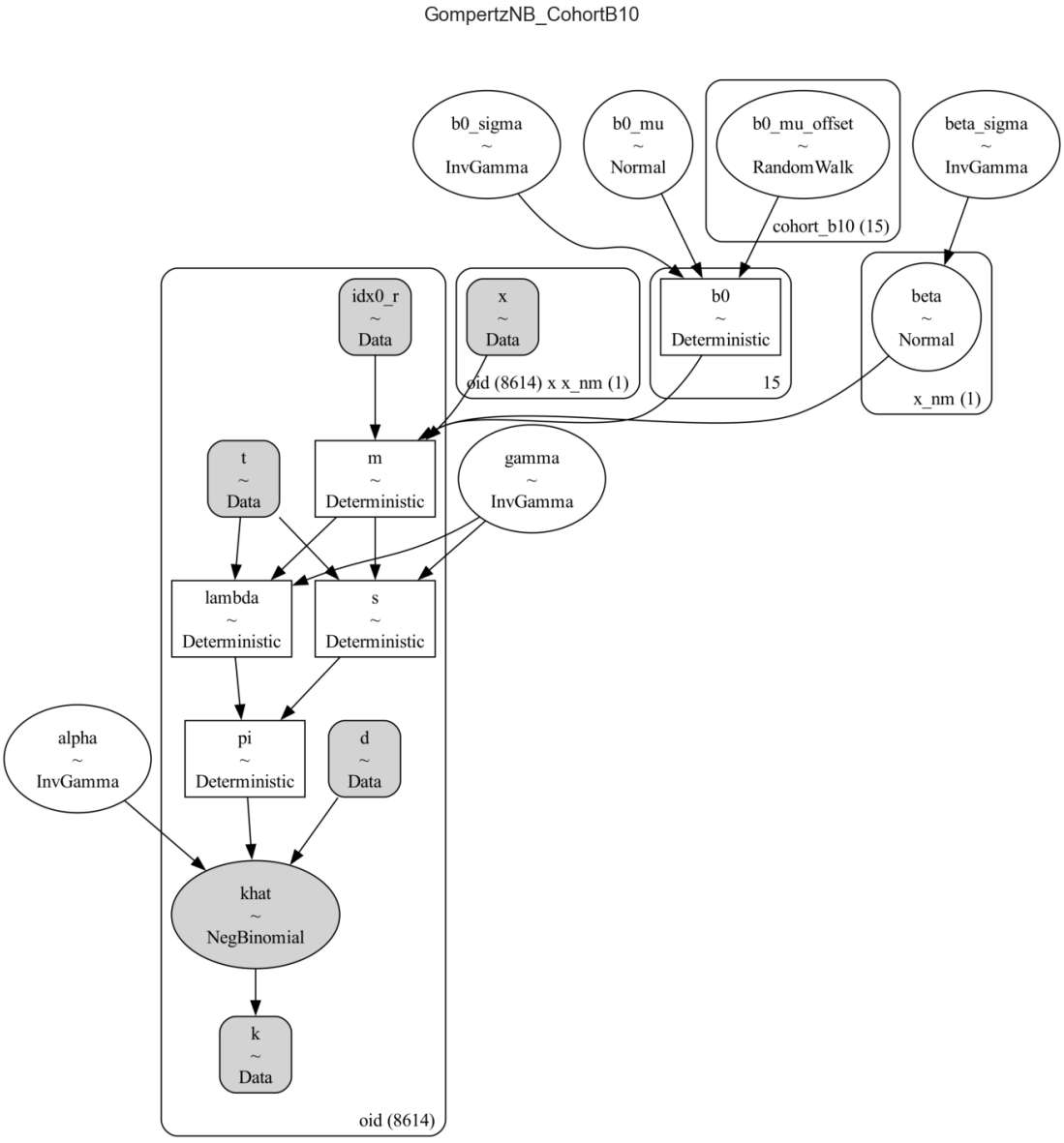
`GompertzPoisson_CohortD1` Establish the core model architecture (based on Accelerated-Failure-Time (AFT)) including: + Gompertz distribution on event density function π with alternative parameterisation (M modal age-of-death, γ) + Poisson distribution on count κ + Linear submodel on M parameter with unpoooled form: `1 + male_t_true` + Use unpooled death year cohort $D_{d1} = \sum_{d1} \kappa_{d1}(t)$, cohorts with a single 1-year resolution

`GompertzNB_CohortD1` Modify `GompertzPoisson_CohortD1` to: + Negative-Binomial distribution on count κ to decouple mean from variance and fit better

`GompertzNB_CohortB10` Modify `GompertzNB_CohortD1` to: + Use partial-pooled **birth year cohort** to yield M varying by birth cohort $D_{b10} = \sum_{b10} \kappa_{b10}(t)$, cohorts with a 10-year resolution

For illustration: plate notation diagram of the most complicated model in this case study (`GompertzNB_CohortB10`) See `oreum_cs_ons` `320_GompertzNB_CohortB10.ipynb` for details and the full workflow

```
f = figio.read(fn='../data/models/graph_gompertznb_cohortb10_v0100_dfx.png',␣
 ↪title=f'GompertzNB_CohortB10', figsize=(12, 10))
```



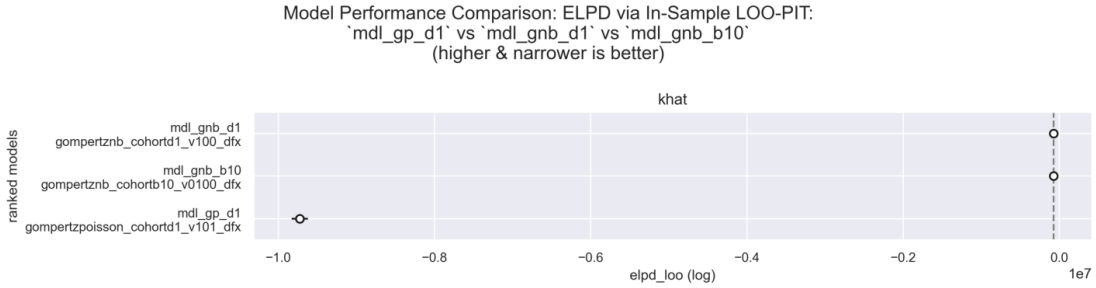
5.3.1 In-sample Model Evaluation

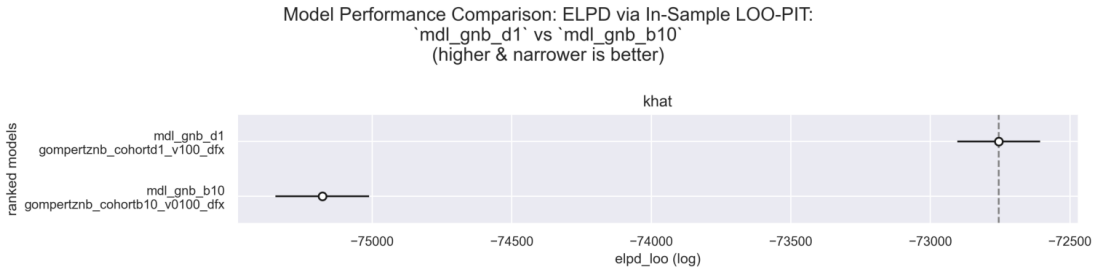
We compare model performance using advanced statistical reasoning.

At the end we can make a quantified evaluation and see that **`GompertzNB_CohortD1`** performs the best for this dataset. Note that `GompertzPoisson_CohortD1` is substantially less performant than any NB model, due to the limitations of the Poisson distribution on the count κ

See `oreum_cs_ons` `311_GompertzNB_CohortD1.ipynb` for details of this in-sample LOO-PIT evaluation method. Note we discover that the

```
f = figio.read(fn='320_2_6_compare_model_performance_all.png', figsize=(12, 3))
f = figio.read(fn='320_2_6_compare_model_performance_subset.png', figsize=(12,␣
↪3))
```





---

6 3. Using the Model Outputs

6.1 3.1 Inference via the linear coefficients

We use the alternative Gompertz parameterisation for event density $\pi(t) \sim \text{Gompertz}(t \mid M, \gamma)$

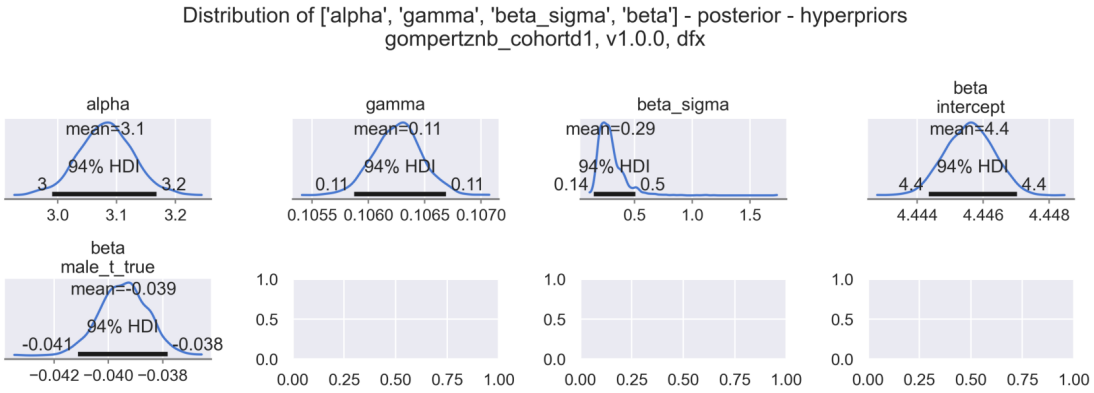
Regression considerations: + We should choose one parameter for the regression term $\beta^T \mathbf{x}$, or carefully manage the covariance between the two. + Because M affects the mode, we will set $M = \exp(\beta^T \mathbf{x})$, and open the potential to check our posterior parameter values to published general literature on "average" lifespan in the UK

So we can view the relative coefficient values to make inferences about the correlation (not causation) of features with changes in M

6.1.1 3.1.1 Interpret effect of Simple Linear Coefficients

First let's view the posterior values of `beta_*` to infer how the numeric values affect α

```
f = figio.read(fn='311_2_7_krushke_priors.png', figsize=(12, 6))
```



Observe:

- `alpha`: $\mathbb{E} \sim 3.1$, HDI_{94} not too narrow, as shown above (§1.2.1) setting $\alpha = 3$ gives a usefully broader peak
- `gamma`: $\mathbb{E} \sim 0.11$, HDI_{94} extremely narrow still
- `beta_s`: $\mathbb{E} \sim 0.29$, HDI_{94} a little higher and broader than specified, same as the `GompertzPoisson` model
- `beta: intercept`: $\mathbb{E} \sim 4.4, 0 \notin \text{HDI}_{94}$ positive, baseline $M \approx 81$, same as the `GompertzPossion` model
- `beta: male_t_true`: $\mathbb{E} \sim -0.039, 0 \notin \text{HDI}_{94}$ driving M lower and π lower (earlier death), slightly more positive and wider than the `GompertzPoisson` model - suggesting this now contains the variance we need to isolate

6.2 3.2 Prediction of Expected Time-to-Event $\hat{\mathbb{E}}_t$ for Individual Observations

We also engineered our model to be able to make predictions on **out-of-sample** (aka previously unseen) data of Expected Time-to-Event $\hat{\mathbb{E}}_t$ (which is formally in the model as $\hat{\pi}(t)$

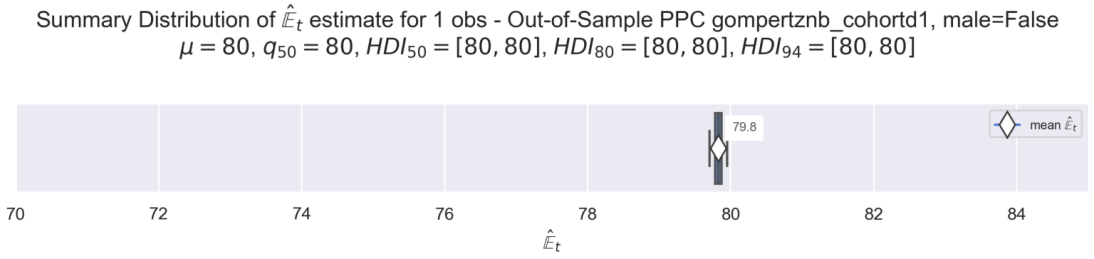
This is not a technical evaluation method, and the proper technical model evaluation is discussed above §2.3, but here we can eyeball the predictions on a subset of data vs the true values, and get a feel for what we can do with the predictive outputs.

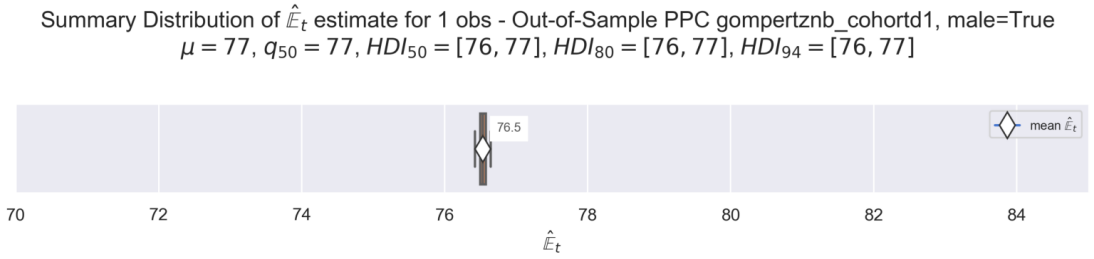
Each prediction is of course a distribution over a range of values, quantifing the uncertainty in the prediction. We can use the mean as the expected value, and the distribution as a measure of the uncertainty.

We can test this distribution against the true data and comment on the model calibration

Assuming a well-calibrated model, we can use the distribution as an **exceedance curve** wherein we choose to use the predicted value at e.g. the 90th percentile, so that e.g. 90% of the time the true value is below our predicted value. This is critical in risk evaluation etc.

```
f = figio.read(fn='311_gompertznb_cohortd1_v1_0_0_dfsx_forecast_et_male_false.
 ↪png', figsize=(12, 3))
f = figio.read(fn='311_gompertznb_cohortd1_v1_0_0_dfsx_forecast_et_male_true.
 ↪png', figsize=(12, 3))
```





Observe:

- Here's the distribution of estimated Expected time-to-event $\hat{\mathbb{E}}_t$ for the forecast group for our **synthetic out-of-sample dataset** for `death_yr = 2023`
  - For Female `male = False`: $\hat{\mathbb{E}}_t \sim 80 \in [80, 78]_{HDI94}$ days
  - For Male `male = True`: $\hat{\mathbb{E}}_t \sim 77 \in [76, 77]_{HDI94}$ days, $\approx 4\%$ sooner
- The estimates have plausible scale (mean life expectancy in UK is reported to be around these values)
- The estimates have a plausible relationship (males die sooner)
- These estimates are lower than the `GompertzPoisson_CohortD1` model (which may or may not be a good thing), but importantly the `HDI94`s are wider, which is great to see, because it means we're handling uncertainty in the data better and can make more-robust, better-qualified predictions

---

---

7 4. Discussion of `Cohort_B10`

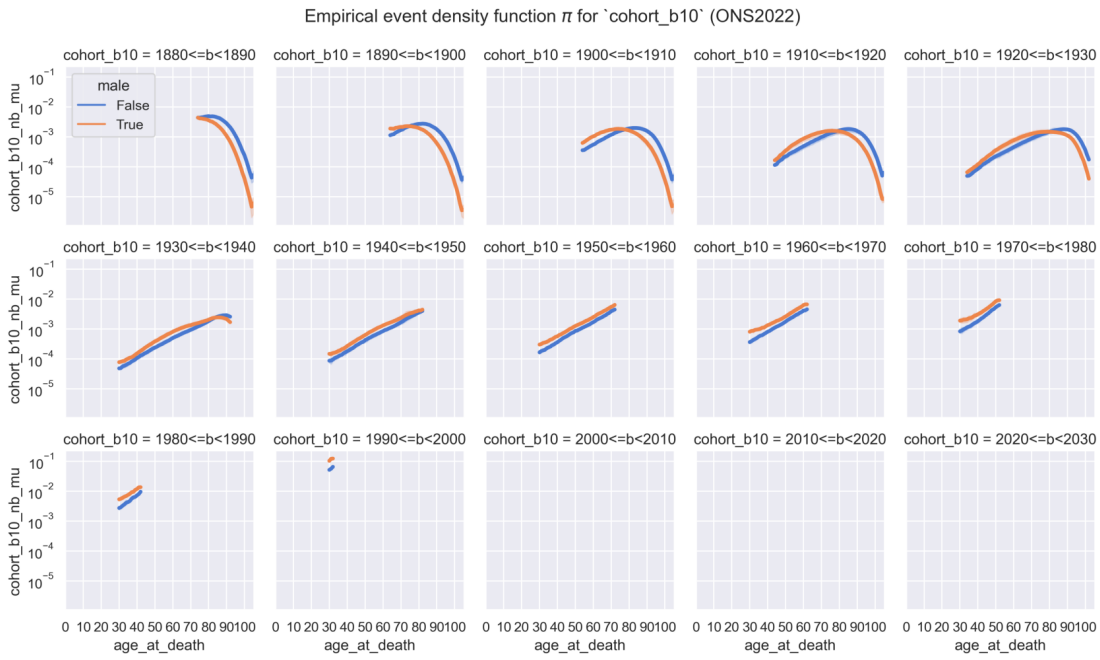
We started this case study with the intention of implementing **partial-pooled birth year cohorts** $\sum_c \kappa_c(t)$, to yield M varying by birth cohort. This seems closer to a real observational study where we have birth dates by individual:

$$D_{b10} = \sum_{b10} \kappa_{b10}(t)$$

We experimented with this at a 10-year resolution in model `GompertzNBCohortB10`, but found poor results with our modified AFT model architecture, mainly due to the nature of the data itself.

7.0.1 The `cohort_b10` groupwise empirical event density functions π have a poorly definied mode M

```
f = figio.read(fn='320_empirical_pi_cohortb10.png', figsize=(12, 8))
```

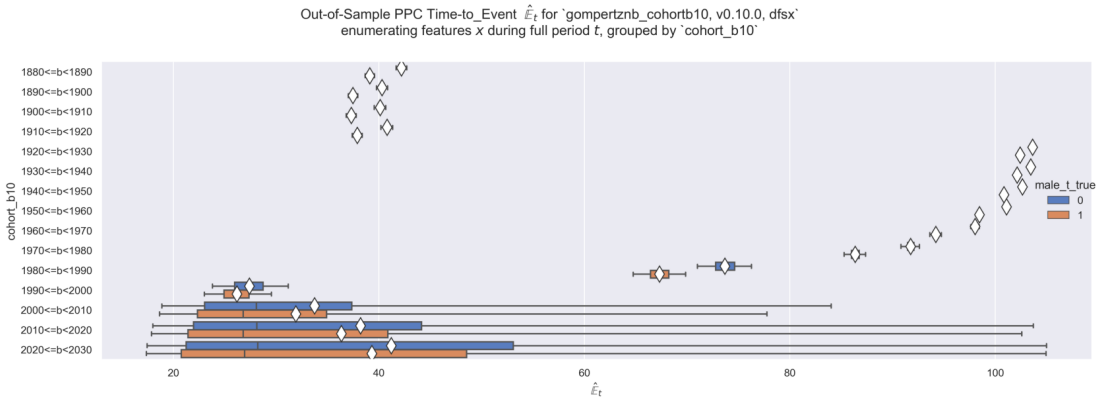


Observe:

- Cohorts `1880 to 1940` have a defined convex modal peak M
- Cohorts `1940 to 2000` do not have a convex modal peak, and could cause trouble - we could try to underweight them in the M component of the log-likelihood
- Cohorts `2000 to 2030` are of course missing, so we will use auto-imputation in the hierarchical structure

7.0.2 The resulting Expected Time-to-Event $\hat{\mathbb{E}}_t$ for Individual Observations were unusable

```
f = figio.read(fn='320_gompertznb_cohortb10_v0_10_0_dfsx_forecast_et.png',␣
↪figsize=(12, 8))
```



Observe:

- So, here's the distribution of estimated Expected time-to-event $\hat{\mathbb{E}}_t$ for our **synthetic out-of-sample dataset** grouped by birth cohort `cohortB10`
- It's not good! Inevitably displaying the same behaviour and relationships discussed above for the plots of $\hat{S}(t)$ i.e
  - Early cohorts `1880 to 1920` still look somewhat plausible, albeit slightly lower than the ONS' own statistics
  - Middle cohorts `1920 to 1980` are terrible - likely because convex M isn't actually observed, and the current top of the event density slope is mistaken for M
  - Middle cohort `1980 to 1990` looks plausible, but there's no reason for the model to get this right, so it's a chance accident
  - Later cohorts `1990 to 2030` are bad because they're underrepresented in the data and largely auto-imputed from the hierarchical mean

7.0.3 We propose using a Gaussian Process to estimate the empirical Event Density Function π

If we were to continue to use this compromised data set from the ONS, the next logical step to improve the general model architecture would be to use a Gaussian Process to directly estimate the empirical Event Density Function π, continue to evidence the count κ

… for future discussion!

7.0.4 Next Steps

Now the interested reader should dig into the full case study Notebooks in project `oreum_cs_ons`

There we demonstrate the full E2E workflow for models of increasing sophistication, including several state-of-the-art methods unavailable to conventional max-likelihood / machine-learning models.

- `100_Curate.ipynb`
- `300_ModelArchitecture.ipynb`
- `310_GompertzPoisson_CohortD1.ipynb`
- `311_GompertzNB_CohortD1.ipynb`
- `320_GompertzNB_CohortB10.ipynb`
