Oreum Industries **Internal Project, 2024Q3**

000_Overview.ipynb

Oreum Case Studies - Lung Cancer Survival Regression `oreum_cs_lung`

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

Here we report a brief overview of a full case study `oreum_cs_lung` in which we demonstrate an E2E workflow for survival regression models of increasing sophistication.

This overview is for 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.

PDF version

Oreum Industries: Technical Resources

We use a complicated, real-world dataset: the NCCTG Lung Cancer Dataset `lung` from the `survival` R package via `statsmodels`. This complicated dataset requires multiple advanced modelling methods to handle and mitigate bad / messy data.

This case study also inspired us to create two original novel contributions to the `pymc` project:

1. A detailed worked example on handling ordinals in `pymc-examples` in GLM-ordinal-features
2. A detailed worked example on handling missing data in `pymc-examples` in GLM-missing-values-in-covariates

See comprehensive explanations and deep technical demonstrations of various survival models in our public project **Oreum Survival** including Accelerated Failure Time and Piecewise Regression Models

Contents

- Setup
- Preamble
- 1. The Dataset and Problem-Space
- 2. The Modelling Work
- 3. Using the Model Outputs

Setup

Imports

```
import sys
from pathlib import Path

from pyprojroot.here import here

# prepend local project src files
module_path = here('src').resolve(strict=True)
if str(module_path) not in sys.path:
    sys.path.insert(0, str(module_path))

from oreum_core import curate, eda
```

Notebook config

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

Data Connections and Helper Objects

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

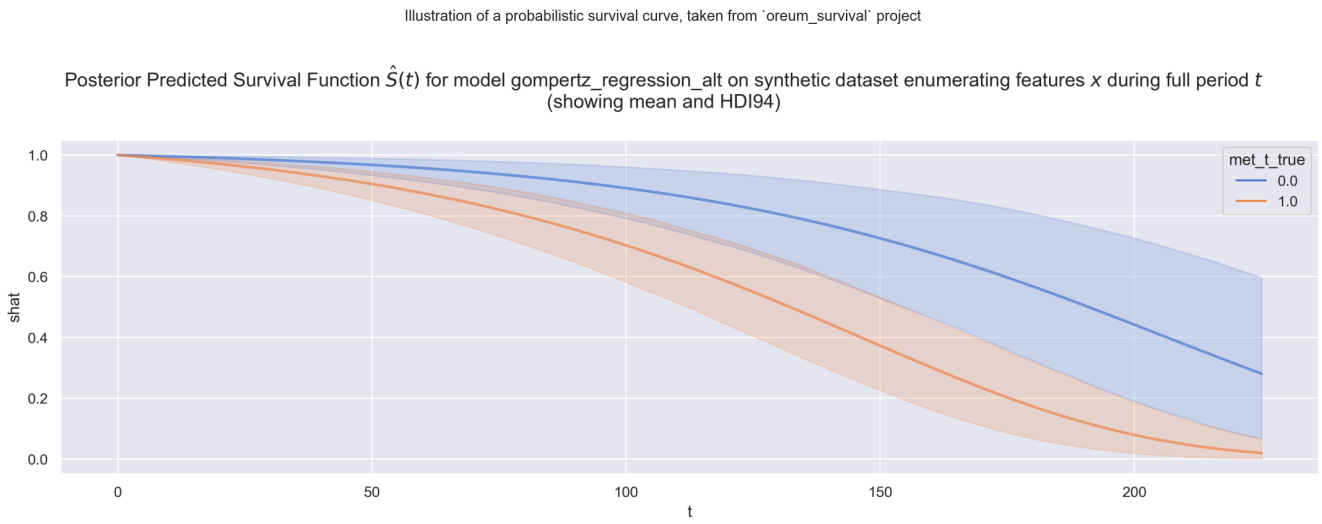
Preamble

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.

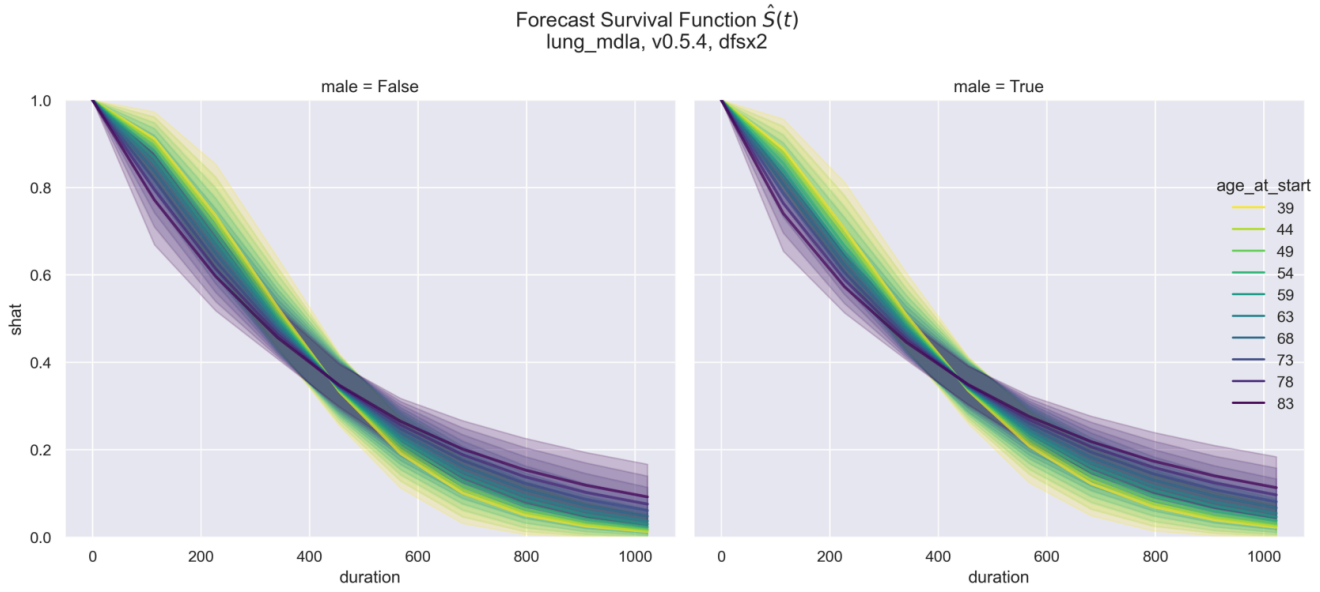
```
f = figio.read(fn='../assets/img/001_gompertz_regression_alt_forecast_survival_functi
        title='Illustration of a probabilistic survival curve, taken from `oreum_surv
        figsize=(16, 6))
```

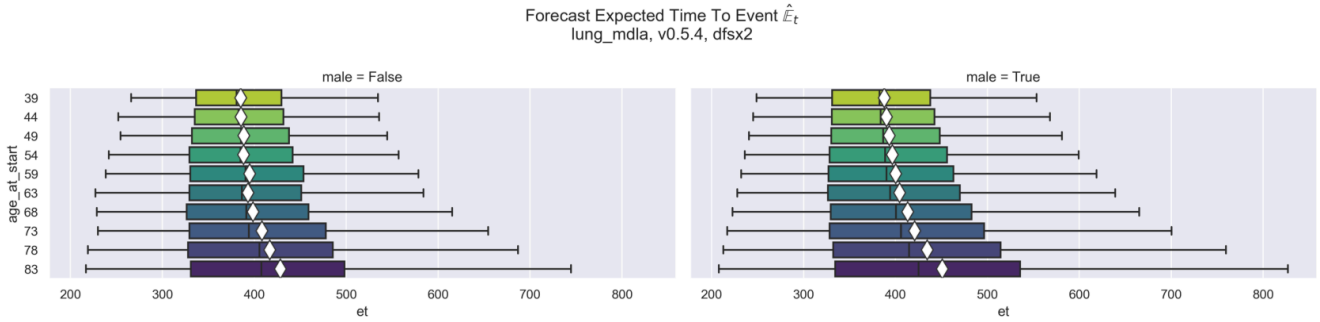


Illustrative Survival Function $S(t)$ and Expected Time-to-Event $\mathbb{E}_t$

Taken from notebook `oreum_cs_lung` `300_ModelA_E2E.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='300_lung_mdla_v054_dfsx2_forecast_sf_p0.png', figsize=(12, 10))
f = figio.read(fn='300_lung_mdla_v054_dfsx2_forecast_et_p1.png')
```





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	**Very strong** Marginal prior distributions represent	**Very weak** No concept of priors. Lack of joint

	Bayesian Inference	Frequentist Max-Likelihood
their initial values represent domain expert knowledge	real-world probability of parameter values before any data is seen.	probability distribution can lead to discontinuities in parameter values.
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.
Handle small datasets	**Very strong** Bayesian concept assumes that there is a probable range of values for each parameter, and that we evidence our	**Very weak** Frequentist concept assumes that there is a single true value for each parameter and that we only discover

Desirable Trait	Bayesian Inference	Frequentist Max-Likelihood
	prior on any amount of data (even very small counts).	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.

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.

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 detailed 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))
```

---

1. The Dataset and Problem-Space

1.1 Dataset

As noted above, We use a complicated, real-world dataset that requires multiple advanced modelling methods to handle and mitigate bad / messy data.

We use the **NCCTG Lung Cancer Dataset** `lung` from the `survival` R package via `statsmodels`.

For illustration: table of the dataset (post-cleaning)

See `oreum_cs_lung` `100_CurateExtractClean.ipynb` for details

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

pid	duration	death	age_at_start	calories_consumed_amt	ecog_physician_cat	inst_id	karno
p000	306	True	74	1175.0	c1	i3	
p001	455	True	68	1225.0	c0	i3	
p002	1010	False	56	NaN	c0	i3	
p225	105	False	75	1025.0	c2	i32	
p226	174	False	66	1075.0	c1	i6	
p227	177	False	58	1060.0	c1	i22	

```
'Shape: (228, 10), Memsize 0.0 MB'
```

Explanation of the features for each of the 228 observations, post-cleaning (which includes cleaning dirty values forcing correct datatypes, renaming to be more clear, etc)

Endogenous (target) features captured at the end of the study period

```
duration: Time-to-event observed during study period (days) (can be
censored)
death:  Death observed during study period
```

Exogenous (predictor) features captured at the start of the study period

```
age_at_start:               Age in years
calories_consumed_count:    Calories consumed at meals
ecog_physician_cat:         ECOG  score as rated by the physician. c0
to c4
inst_id:                    Institution identifier code
karno_patient_amt:          Karnofsky score as rated by patient {0,
..., 100}
karno_physician_amt:        Karnofsky score as rated by physician {0,
..., 100}
male:                       Sex of patient is male (True/False)
weight_loss_prior_amt:      Weight loss in last six months (pounds)
```

The medical details behind this data are explained and linked further in the dataset page

1.2. Time to Event t and Event Density π

The 228 individuals have an observed "lifetime" aka time-to-event during the study, measured in days.

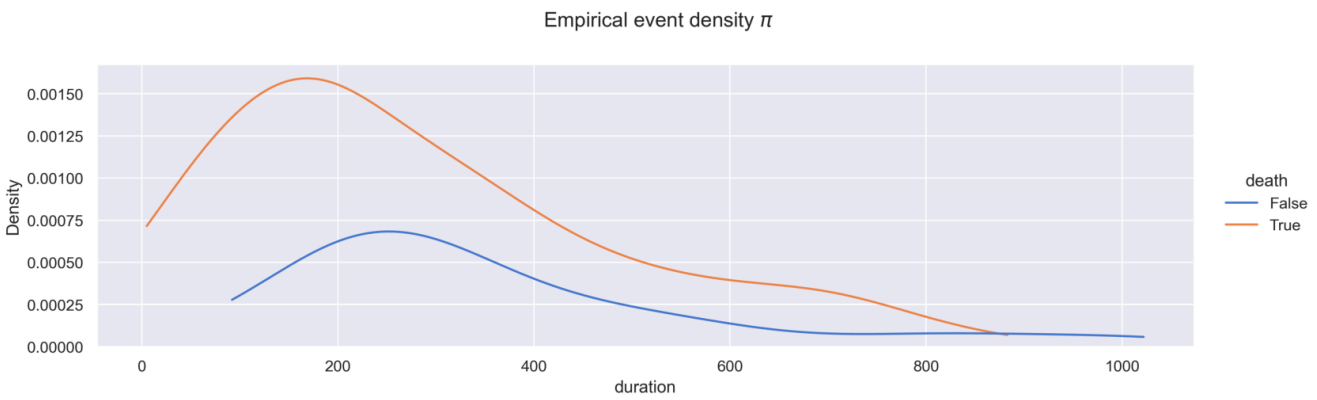
For each individual this starts when they enter the study (not necessarily all on the same day), and ends either at death (`death=True`) or when the study ends.

Lifetimes (Censored Time-to-Event)

```
f = figio.read(fn='100_individual_observed_t.png', figsize=(14, 10))
```

Empirical event density π

```
f = figio.read(fn='100_empirical_event_density.png', figsize=(12, 12))
```



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

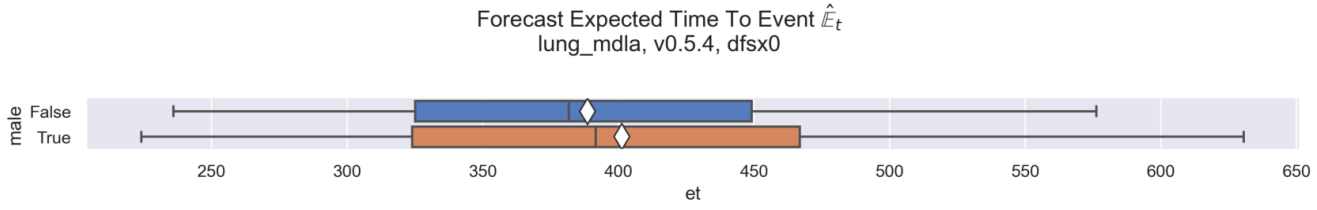
We seek to estimate π and thus the **Expected Time-to-Event** $\hat{\mathbb{E}}_t$

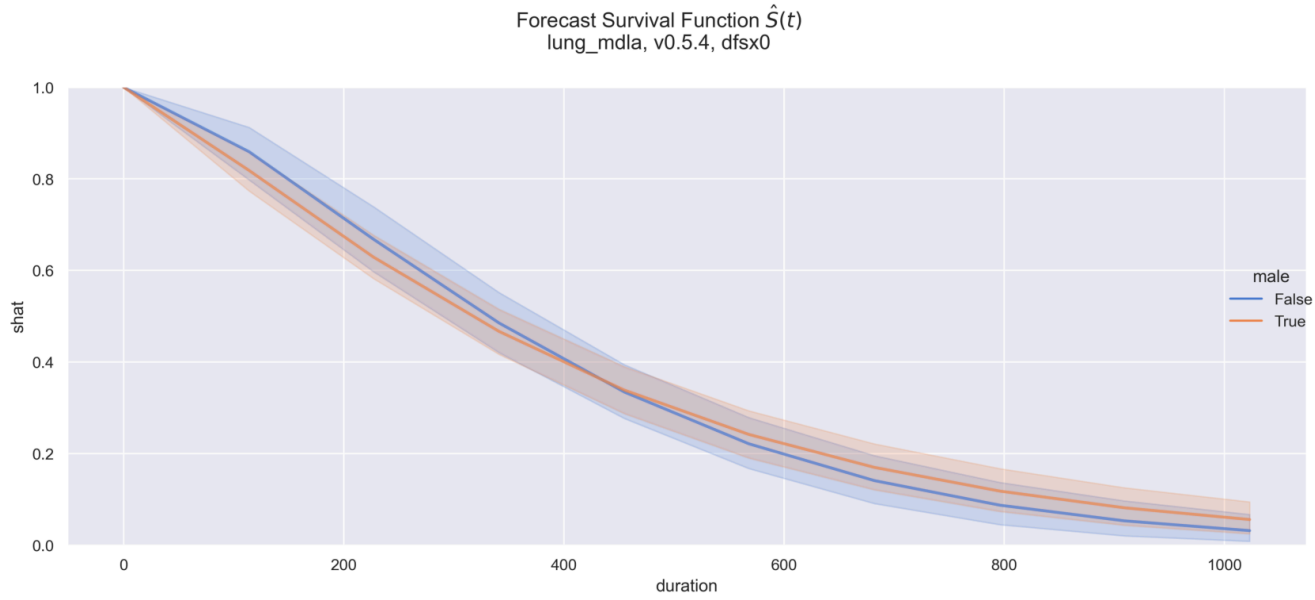
As a side-effect of the mathematical relationships, we also get a **Survival Function** $\hat{S}(t)$ which can be a more familiar representation for newcomers to understand.

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.

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='300_lung_mdla_v054_dfsx0_forecast_et_p0.png', figsize=(12, 10))
f = figio.read(fn='300_lung_mdla_v054_dfsx0_forecast_sf_p0.png', figsize=(12, 10))
```





---

2. The Modelling Work

2.1 Architectures

We use an **Accelerated Failure Time (AFT) model**, specifically a **censored Weibull-distribution on the Event Density** π.

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 purpose of this case study is to demonstrate an E2E workflow for models of increasing sophistication. 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.

For illustration: math and plate notation diagram of the most basic model (`Model A0`)

See `oreum_cs_lung` `300_ModelA_E2E.ipynb` for details and the full workflow. See `oreum_cs_lung` `src/model/lung.py` for the code

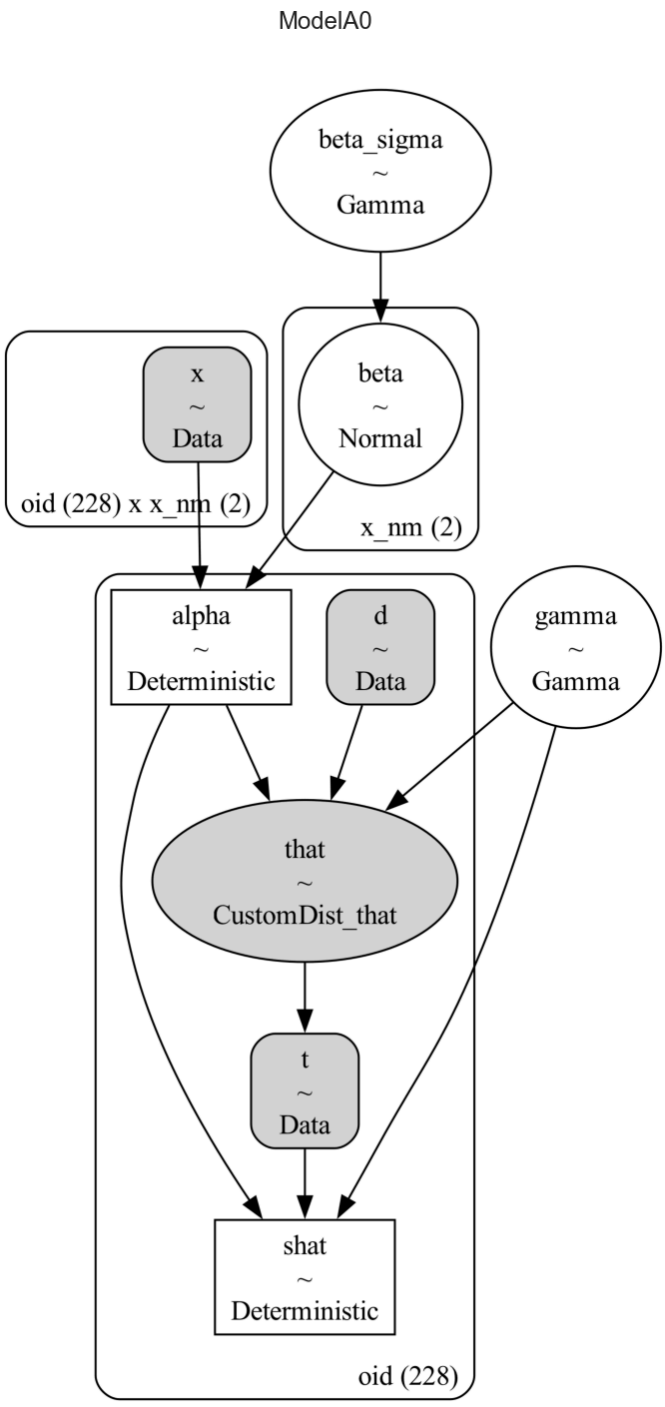
Snippet of the general math for the **censored Weibull likelihood** and the linear-submodel for regression onto features:

$$
\begin{aligned}
\sigma_\beta &\sim \text{Gamma}(\alpha = 1, \beta = 2) \\
\beta &\sim \text{Normal}(\mu = 0, \sigma = \sigma_\beta) \\
\alpha &\sim \exp(\beta^T \mathbf{x}) \\
\gamma &\sim \text{Gamma}(\alpha = 1, \beta = 200)
\end{aligned}
$$

$$
\begin{aligned}
\hat{\pi}(t) &\sim \alpha\gamma(\gamma t_i)^{\alpha-1,\, d_i} \cdot \exp(-(\gamma t)^\alpha) \\
&\sim \text{Weibull}(t \mid \alpha, \gamma)
\end{aligned}
$$

Plate notation of simplest model `ModelA0`

```
f = figio.read(fn='../data/models/graph_lung_mdla_v054_dfx0.png', title='ModelA0', fi
```

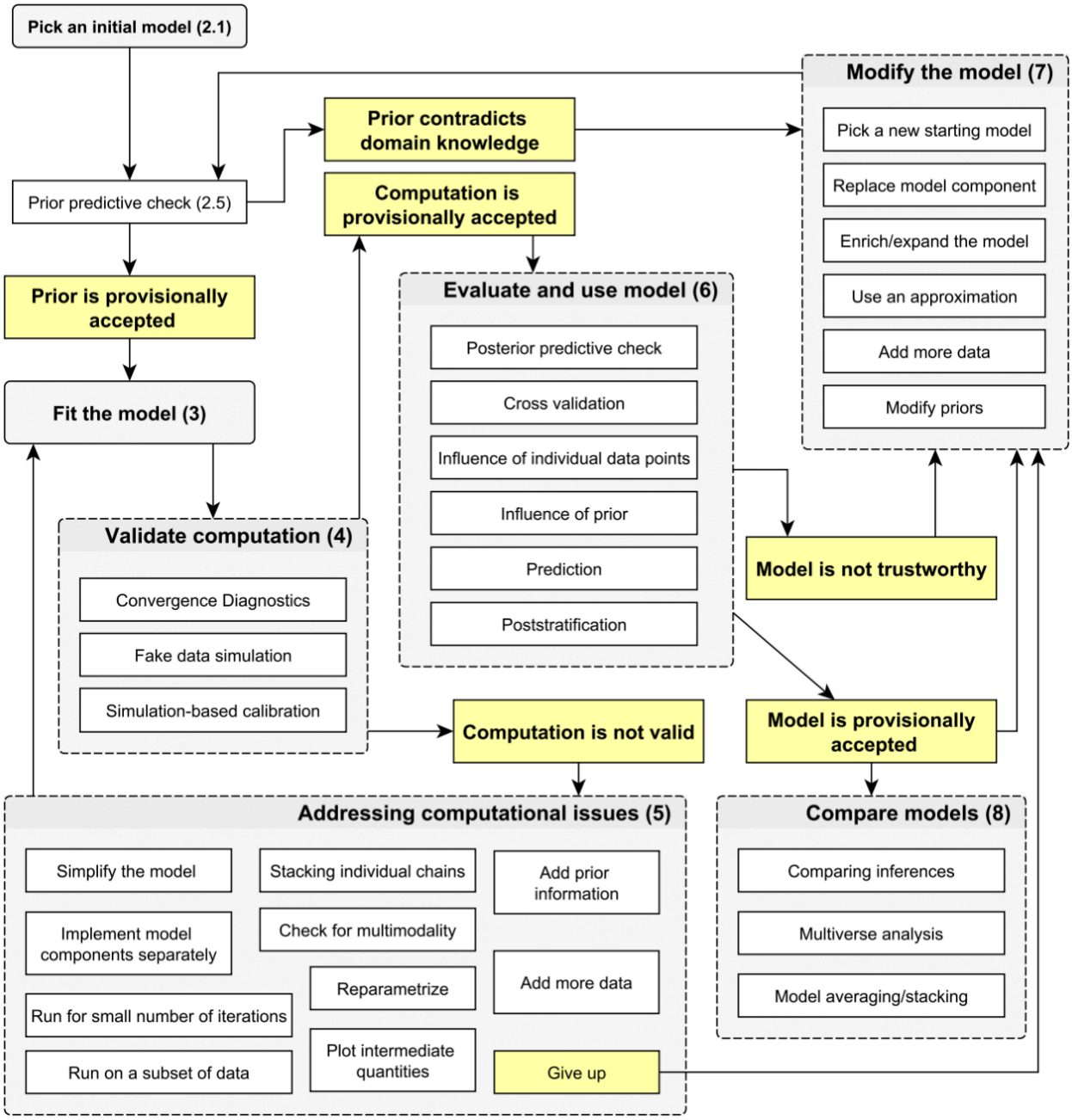


2.2 Bayesian Workflow

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='Gelman et al., 2000 - I
figsize=(12, 10))
```

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



2.3 The Model Variants and Technical Evaluation

Generally, we want to use as many observations and features from the dataset as possible, although obviously we have to mitigate overfitting (we use L2 regularization) and resource constraints.

In general we aim to create models with increasing sophistication and ability to handle the nature of the dataset. In particular in this dataset we encounter numerics, booleans, categoricals, ordinals and missing values!

In this particular case study we ended up with 5 different model variants:

`Model A`

- Establish the core model architecture for Accelerated-Failure-Time (AFT) including:
  - Weibull likelihood on event density
  - Event censoring via CustomDist
  - Linear submodel on α param
- In particular create 3 sub-variants using the (complete) numeric and boolean values available:
  - `male_t_true`
  - `age_at_start`
- Linear component: `1 + male_t_true + age_at_start + age_at_start^2`

`Model B`

- Extend `Model A` to include (missing-value / incomplete) numeric values
- We use a **sophisticated technique of hierarchical auto-imputation** to fill-in missing values of `calories_consumed_amt`, `weight_loss_prior_amt` within the model itself (not a pre-processing step!)
- Linear component: `1 + male_t_true + age_at_start + age_at_start^2 + calories_consumed_amt + weight_loss_prior_amt`

`Model C`

- Extend `Model B` to include (complete) categorical values
- We use a **sophisticated technique with a hierarchical prior** (aka mixed random effects) to include the categorical feature `inst_id`
- Interestingly, we see the model performance decrease, likely because the cardinality is too high given the small dataset, and in the real-world this would encourage us to seek more information from the data source to learn latent groupings within the `inst_id` and apply further hierarchies into the model
- Linear component: `0 + inst_id + male_t_true + age_at_start + age_at_start^2 + calories_consumed_amt + weight_loss_prior_amt`

`Model D`

- Extend `Model C` to include (complete) ordinal values
- We use a **sophisticated technique with Dirichlet allocator to include ordinal categorical features** `ecog_physician_cat`, `karno_physician_cat`, `karno_patient_cat`
- These are present due to misuse of a subjective value that's been mapped to a metric scale, also see `pymc-examples` `GLM-ordinal-features.ipynb` for detailed discussion and worked examples

- Linear component: `0 + inst_id + ecog_physician_cat + karno_physician_cat + karno_patient_cat + 'male + age_at_start + np.power(age_at_start,2) + calories_consumed_amt + weight_loss_prior_amt + death`

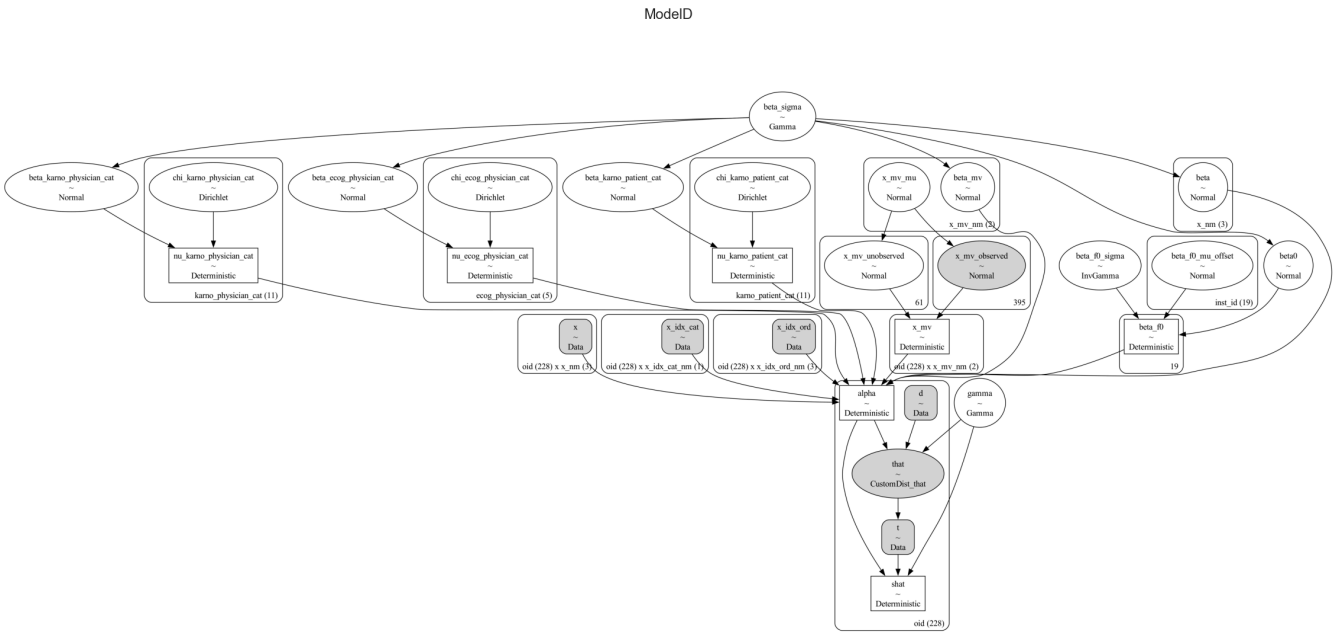
`Model E`

- Simplify `Model C` to remove `inst_id`, to make use of the innovations in `Model D` and remove the performance drop of `Model C`
- We compare the models `Model A0` vs `Model E` on an in-sample dataset to eyeball the results

For illustration: plate notation diagram of the most complicated model in this case study (`Model D`)

See `oreum_cs_lung` `303_ModelD_E2E.ipynb` for details and the full workflow, and note we actually pull this back to a simpler Model E which achieves the best results

```
f = figio.read(fn='../data/models/graph_lung_mdld_v010_dfx0.png', title='ModelD', figs
```



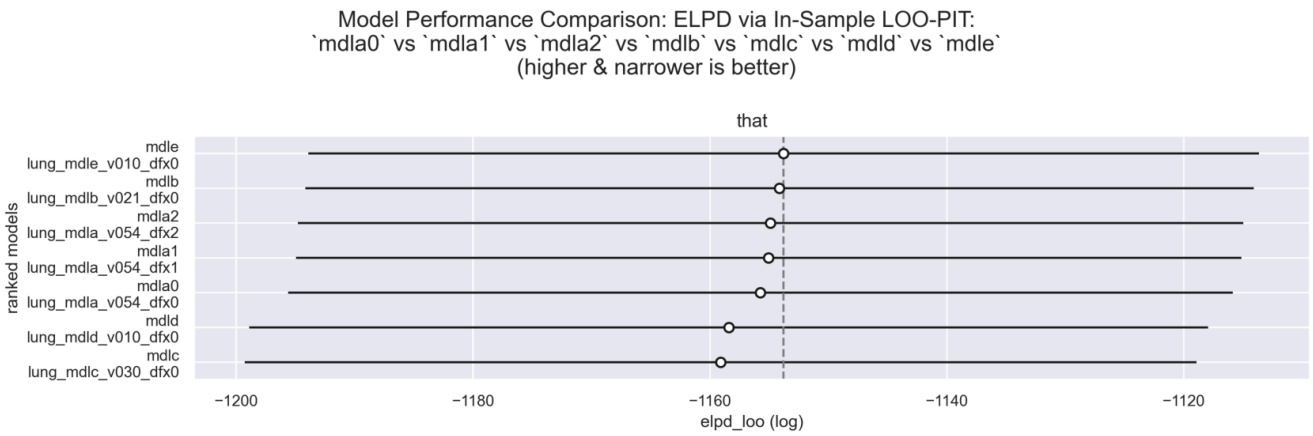
In-sample Model Evaluation

We create a variety of models according to the nature of the data and compare their performance using advanced statistical reasoning.

At the end we can make a quantified evaluation and see that **Model E** performs the best for this dataset.

See `oreum_cs_lung 304_ModelE_E2E.ipynb` for details of this in-sample LOO-PIT evaluation method

```
f = figio.read(fn='304_1_6_compare_model_performance.png', figsize=(12, 10))
```



---

3. Using the Model Outputs

3.1 Inference via the linear coefficients

We used a 2-parameter form of the Weibull with a linear regression onto $\alpha \sim \exp(\beta^T \mathbf{x})$.

When:

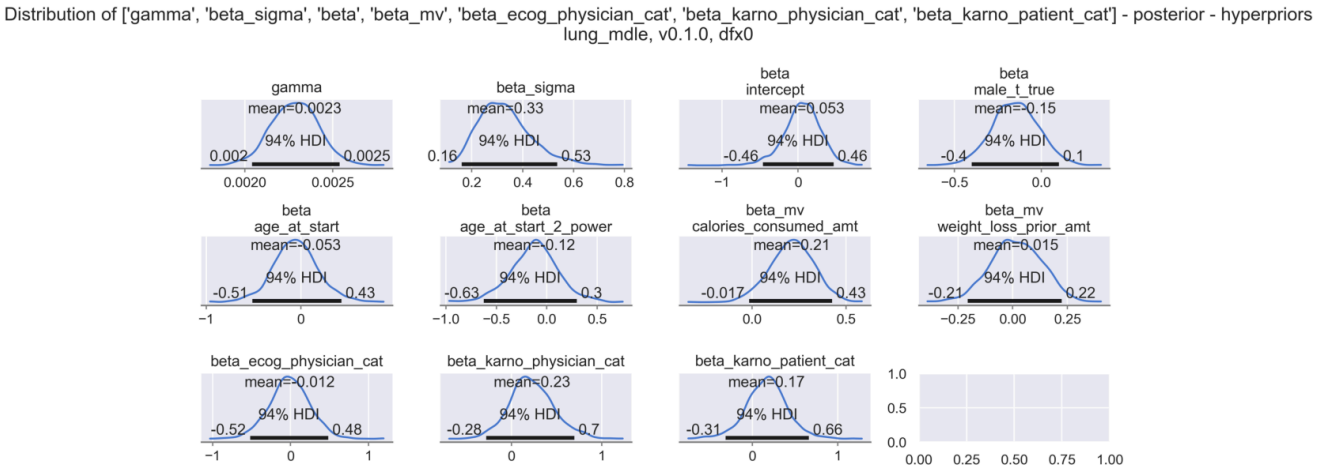
- $\alpha < 1$, hazard decreases over time e.g. early mortality
- $\alpha = 1$, hazard is constant $\lambda \sim \gamma$ and the model reduces to the Exponential model
- $\alpha > 1$, hazard increases over time e.g. later mortality

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

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='304_1_7_krushke_betas.png', figsize=(12, 10))
```



Observe:

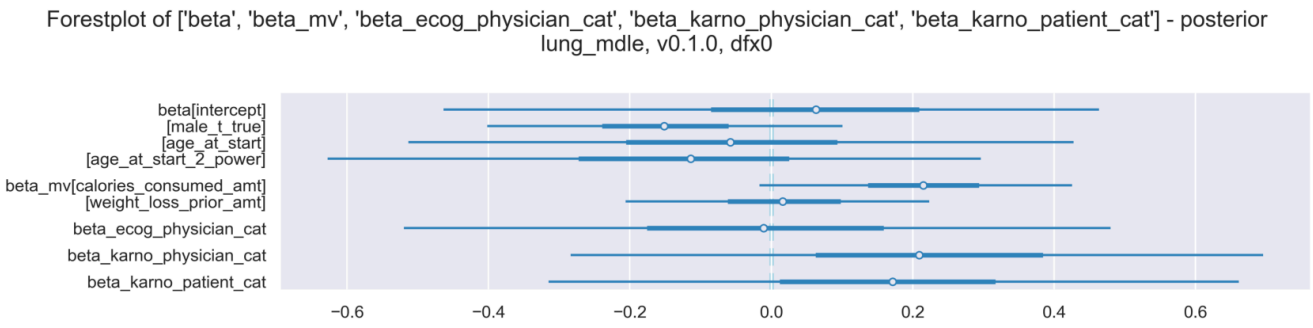
- `beta: intercept` : $\mathbb{E} \sim 0.053,\ 0 \in HDI_{94}$, mild effect $\alpha > 1$ i.e. overall increasing hazard fn λ over duration (later mortality)
- `beta: male_t_true` : $\mathbb{E} \sim -0.15,\ 0 \in HDI_{94},\ ,\ 0 \notin HDI_{80}$, appreciable effect, earlier mortality for males
- `beta: age_at_start` : $\mathbb{E} \sim -0.053,\ 0 \in HDI_{94}$, minimal effect, slightly earlier mortality for higher ages
- `beta: age_at_start_2_power` : $\mathbb{E} \sim -0.12,\ 0 \in HDI_{94}$, mild effect, earlier mortality for higher ages
- `beta_mv: calories_consumed_amt` : $\mathbb{E} \sim 0.21,\ 0 \in HDI_{94},\ 0 \notin HDI_{80}$, stronger effect, later mortality with higher calories
- `beta_mv: weight_loss_prior_amt` : $\mathbb{E} \sim 0.015,\ 0 \in HDI_{94}$ minimal effect, slightly later mortality with higher calories
- `beta_ecog_physician_cat` : $\mathbb{E} \sim -0.012,\ 0 \in HDI_{94}$ minimal effect, slightly earlier mortality with higher ecog physician score
- `beta_karno_physician_cat` : $\mathbb{E} \sim 0.23,\ 0 \in HDI_{94}$, appreciable effect, later mortality with higher karno physician score
- `beta_karno_patient_cat` : $\mathbb{E} \sim 0.17,\ 0 \in HDI_{94}$, appreciable effect, later mortality with higher karno patient score

We won't attempt to interpret the coefficients much further because we're not physicians and this isn't our data / medical study.

However, we can note that `beta: male_t_true` , `beta: age_at_start_2_power` , `beta_mv: calories_consumed_amt` , `beta_karno_physician_cat` , `beta_karno_patient_cat` all have appreciable correlating effects with mortality in this model - and would be good candidate features for further exploration, model finessing, and further medical study

Let's view those `beta` coefficients in a forestplot to gain a better understanding of the relative effects we just noted

```
f = figio.read(fn='304_1_7_forestplot_betas.png', figsize=(12, 8))
```



Observe:

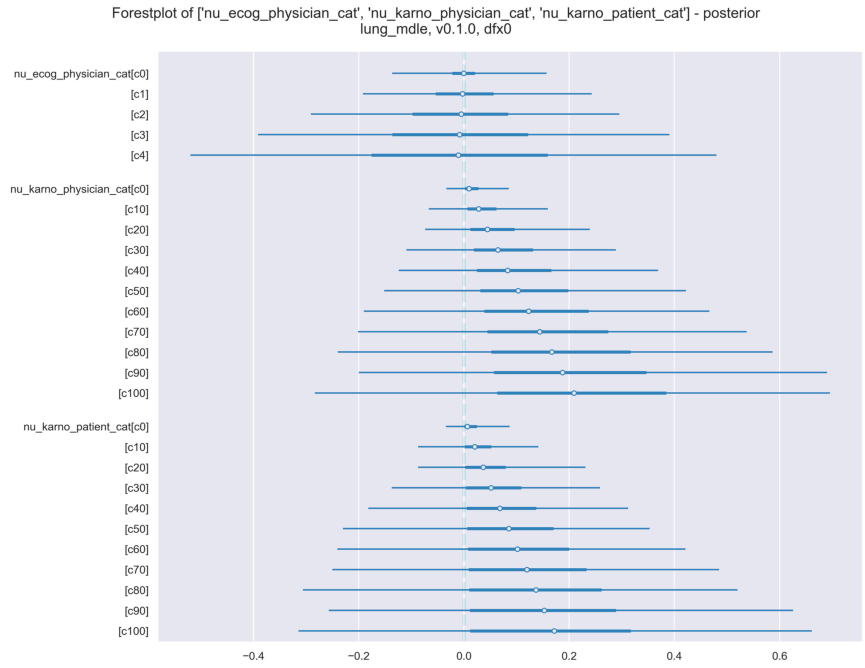
This is the same info we saw above, but let's highlight the biggest movers:

- `beta: male_t_true` : $\mathbb{E} \sim -0.15,\ 0 \in HDI_{94},,\ 0 \notin HDI_{80}$, appreciable effect, earlier mortality for males
- `beta: age_at_start_2_power` : $\mathbb{E} \sim -0.12,\ 0 \in HDI_{94}$, mild effect, earlier mortality for higher ages
- `beta_karno_physician_cat` : $\mathbb{E} \sim 0.23,\ 0 \in HDI_{94}$, appreciable effect, later mortality with higher karno physician score
- `beta_karno_patient_cat` : $\mathbb{E} \sim 0.17,\ 0 \in HDI_{94}$, appreciable effect, later mortality with higher karno patient score

3.1.2 Interpret effects of ordinal values

We can also view the posterior values of `nu_*` to infer how the ordinal categorical values in `beta_ecog_physician_cat`, `beta_karno_physician_cat`, `beta_karno_patient_cat` affect α

```
f = figio.read(fn='304_1_7_forestplot_nus.png', figsize=(12, 5))
```



Observe:

- `nu_ecog_physician_cat` : as `c0 -> c4` there's barely any impact, except for widening uncertainty. As we see in `100_Curate_ExtractClean.ipynb` §3.2, the upper values `c3` & `c4` are almost not observed, so this huge uncertainty is expected, however, there's no appreciable difference within `c0` , `c1` , `c2` suggesting it's not a worthwhile metric for our model usage
- `nu_karno_physician_cat` : as `c0 -> c100` there's a fairly linear increase on `\alpha` (later mortality). The near-linear response suggests this ordinal metric is being used quite reliably
- `nu_karno_patient_cat` : as `c0 -> c100` there's a fairly linear increase on `\alpha` (later mortality). The near-linear response suggests this ordinal metric is being used quite reliably, but the effect is weaker than `nu_karno_physician_cat` which makes sense if we assume that the physician is a better judge of health

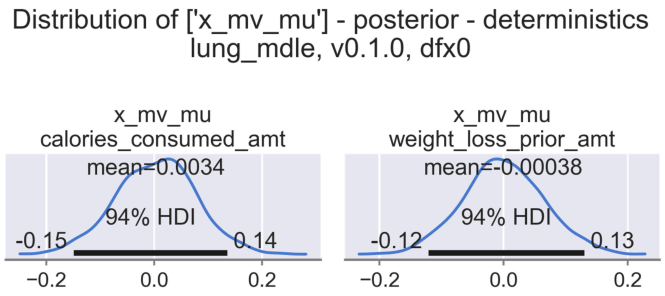
For more detail and discussion on ordinals, see our original novel contribution to `pymc-examples` in GLM-ordinal-features

3.1.3 Interpret auto-imputed missing values

We can also view the posterior values of `x_mv_mu *` to infer how the model has auto-imputed hierarchical expected values for the **missing data** in features `calories_consumed_amt` and `weight_loss_prior_amt` .

For brevity we wont look into the auto-imputed posterior values `x_mv` for individual observations, and refer the reader to `304_ModelE_E2E.ipynb` for full detail

```
f = figio.read(fn='304_1_7_krushke_x_mv.png', figsize=(12, 2.5))
```



Observe:

- `calories_consumed_amt` : $\mathbb{E} \sim 0.0034,\ 0 \in HDI_{94}$
- `weight_loss_prior_amt` : $\mathbb{E} \sim -0.0004,\ 0 \in HDI_{94}$
- In both cases with zscored the data prior to modelling, so these hierarchical values effectively `0` are exactly what we would expect to see if the missing values are Missing at Random (MAR). This supports our choice to include these features without fear that we are introducing systemic issues in the data

For more deatil and discussion on handling missing data, see our original novel contribution to `pymc-examples` in GLM-missing-values-in-covariates

3.2 Prediction of Expected Time-to-Event $\hat{\mathbb{E}}_t$ for individual observations

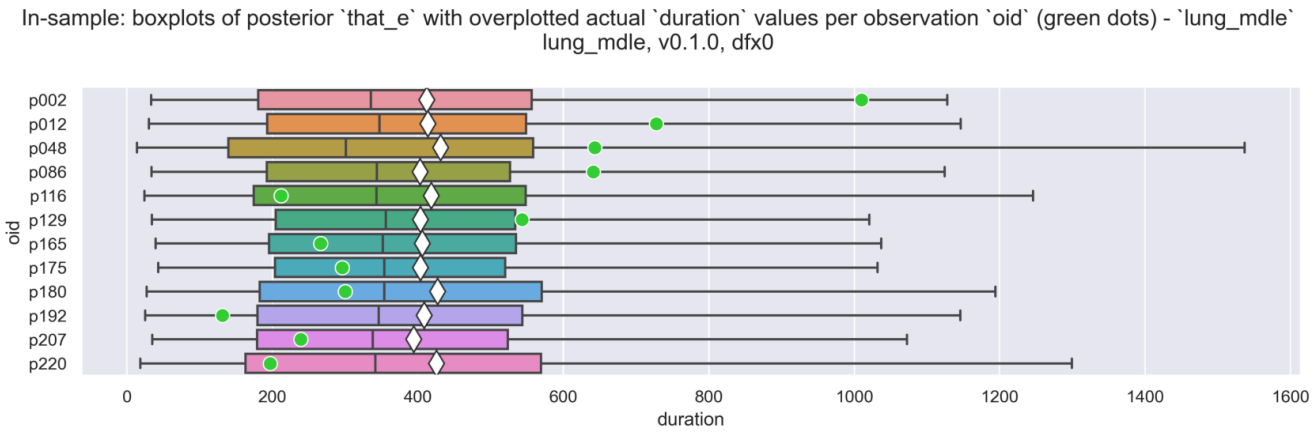
We also engineered our model to be able to make predictions on **out-of-sample** (aka previously unseen) data of Expected Time-to-Event $\hat{\mathbb{E}}_t$ (which is formally in the model as $\hat{\pi}(t)$

This is not a technical evaluation method, and the proper technical model evaluation is discussed above $\S 2.3$, 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, quantifying 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='304_2_1_predicted_that.png', figsize=(12, 6))
```



Observe:

- This is a random subset of the full dataset, which itself is very small and noisy, so we shouldn't expect perfection
- Nonetheless, we see useful predictions where the HDI_{94} has always captured the true value, and in most cases the much smaller HDI_{50} has captured the correct value
- The ability of our principled model to make useful predictions on out-of-sample data is incredibly powerful

---

Next Steps

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

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_ExtractClean.ipynb`
- `200_EDA_Survival.ipynb`
- `300_ModelA_E2E.ipynb`
- `301_ModelB_E2E.ipynb`
- `302_ModelC_E2E.ipynb`
- `303_ModelD_E2E.ipynb`
- `304_ModelE_E2E.ipynb`

oreum

INDUSTRIES
