The CatFish Handbook

CatFish: A Bayesian Intelligence Model for
Multi-Dimensional Industrial Forecasting and Decision Making

Ed Egan

February, 2024. Version 1.5.

PETABYTE ECONOMICS CORP.
www@petabyteeconomics.com

Contents

1 Summaries

1.1 Executive Summary

CatFish is a state-of-the-art Bayesian Intelligence (BI) model for multi-dimensional demand forecasting, business decision support and simulation, and supply-chain optimization in big data industrial contexts. CatFish is based on a **cat**egorical variable multiplicative **Poisson** economic model, which offers functionality that Artificial Intelligence (AI) models cannot:

- **Bayesian Intelligence**: Leverage expert priors and incorporate information from outside AI or econometric models to conduct what-if scenarios or assert beliefs about the future. Catfish can provide inferences without the panel data required to train an AI and generate forecasts that exhibit mean-reversion, include not-seen-before trend breaks, or follow other models' trends.

- **Millions of interpretable, multi-dimensional categorical variables**: Unlike black box AI-based models, this white box economic model lets you understand the determinants of your demand; and every meaningful factor adds to the model's predictive power! CatFish can use any categorical variable and any number of dimensions. So, you are no longer limited to just time-varying characteristics like seasonality, availability and promotions as factors. Unleash the power of contextual information inherent in regionalized socio-economic data and demand shocks, like zip5-grain income or weather, to forecast in the locations and characteristics that matter for your supply-chain.

- **Full joint predictive distributions**: Full distributions, not point forecasts, are needed for almost every sophisticated business decision. You need full joint distributions to do dynamic optimization, recover demand fluctuation probabilities for risk analysis, or allocate products to locations with different or varying critical ratios.

- **Aggregably consistent petabyte-scale forecasts**: Coherent strategic, operations, and financial decision-making require consistent forecasts at every level of aggregation enterprise-wide. Whether you know your aggregate demand's time trend or want to forecast it, CatFish can properly allocate it across every dimension and grain.

- **Superior accuracy and calibration**: CatFish gets better the more meaningful variables you add, and you can include outside forecasts, so it can quickly achieve better accuracy than most other models. It natively produces count distributions calibrated to your demand's characteristics.

More information on CatFish is available from these resources:

- Webpage: https://www.petabyteeconomics.com/catfish.html
- Brochure: https://www.petabyteeconomics.com/files/CatFish-Brochure.pdf
- GitHub: https://github.com/petabyteeconomics/CatFish

1.2 Technical Summary

CatFish currently comprises a base model, an extension, and two samplers. Both samplers have "autotailor" methods for automatic warm-up and thinning in production environments where manual supervision is inappropriate.

The base model is primarily cross-sectional and produces full-distribution mixed-Poisson ensembles. For inference we use a Markov chain Monte Carlo (MCMC) Gibbs sampler, which guarantees ergodicity and places no constraints on the model's topology.[1].

The model defines a space as the cartesian product of each dimension. A location, E, is a point in that space, such as a date-product-zipcode triplet. The Poisson rate, λ, that applies to location E is then the product of the effect, θ_c, of all categorical variables $c \in C_E$ that pertain to E.

$$p(u_E|\lambda_E) \sim \text{Pois}(\lambda_E) \quad \text{where } \lambda_E = \prod_{c \in C_E} \theta_c \tag{1}$$

The categorical variables can also be percentiles (e.g., deciles) of continuous variables and binary indicators. One dimension of the model is usually time, and then there are typically categories for days, certain holidays[2], quarters, years, or other time-based groupings. A mix

[1] Using such an MCMC sampler is appropriate in experimental and research and development contexts. However, samplers that restrict topologies and other inference approaches, including machine-learning solvers, can be much more efficient once the optimum topology has been determined.

[2] See, for example, https://docs.aws.amazon.com/forecast/latest/dg/holidays.html.

of Poisson rates then provides distributions of levels at each time grain. CatFish's cross-sectional approach to time is competitive with most time-series models.[3] However, users can include pre-determined global growth trends, product life cycle effects, and other time-series constructs in the model.

An extension adds hierarchical priors to CatFish so that it can "learn about the urn" and produce mixed negative binomial ensembles.[4] The extended model uses a Metropolis-within-Gibbs sampler and adds random walks with drift, as well as other functionality.

2 Economic Theory

2.1 Dimensions and Space

The model supports any number of dimensions with labels of any datatype. Commonly used dimensions include product, geography, and time, but clients can include anything on which they have data for demand.

The space of the model, $\mathbf{E}$, is the cartesian product of $d \in \{1, \ldots, D\}$ orthogonal dimension vectors. These dimensions have label vectors, $\mathbf{l_d}$, and vectors of indices, $\mathbf{e_d}$. For each dimension, the two vectors are columns of a table that maps $l \leftrightarrow e$, named $l2e$.

The space contains the model's data, $\mathbf{u}$. This data can include observations (i.e., training data) and outcomes (forecast values), which may cover partially disjoint areas, particularly when using historical data to forecast the future. So, it is convenient to store vectors of 'training' flags, $\mathbf{t_d}$, to indicate the space's nature for each dimension as a third column in the $l2e$ table.

$\mathbf{E}$ can be an D-dimension matrix indexed by coordinates $(\mathbf{e_1}, \ldots, \mathbf{e_D})$ or a unidimensional 'linear' vector. In linear vector notation, u_E is the data at location E. The training flags demark the training space, $\mathbf{E}^T$, and forecast space, $\mathbf{E}^F$, of the model.

[3] In testing, model averaging CatFish with time-series models often greatly enhances accuracy, and CatFish can also use other models for time trends.

[4] Poisson-based models are prone to under-dispersion, as the mean of the distribution is equal to its variance. Using mixed negative binomials addresses this issue.

2.2 Bayesian Model

From equation 1, the Poisson rate λ for some outcome u is the product of categorical variables, $\boldsymbol{\theta_C}$, that bear upon that outcome. Let θ_c be the parameter value an indicator variable takes if category c applies, and let the vector of parameter values of the other orthogonal indicator variables $\tilde{c} \neq c \in C$ be denoted $\boldsymbol{\theta_{\tilde{C}}}$. Then, the vector of rates, $\boldsymbol{\lambda}$ where θ_c applies is given by:

$$\boldsymbol{\lambda} = \prod \boldsymbol{\theta_C} = \theta_c \cdot \prod_{\tilde{c} \neq c} \boldsymbol{\theta_{\tilde{C}}} = \theta_c \cdot \frac{\boldsymbol{\lambda}}{\theta_c} \tag{2}$$

Putting equation 2 into equation 1, and using $\mathbf{u}$ to denote the vector of outcomes where θ_c applies, the joint sampling distribution for θ_c is then:

$$p(\mathbf{u}|\theta_c) = \theta_c^{\sum \mathbf{u}} e^{-\theta_c \cdot \sum \frac{\boldsymbol{\lambda}}{\theta_c}} \cdot c(\mathbf{u}) \quad \text{for } \mathbf{u} \in \mathbb{Z}_{\geq 0} \tag{3}$$

When the sampling distribution is Poisson, the conjugate prior distribution for the parameter is gamma. Accordingly, the prior for each θ_c is given by:

$$p(\theta_c) \sim \text{gamma}(\theta_c, \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \theta_c^{\alpha-1} e^{-\beta\theta_c} \quad \text{for } \alpha, \beta > 0 \tag{4}$$

where α is the shape paramater and β is the rate parameter.[5,6]

The (conditional) posterior distribution (of θ_c given $\mathbf{u}$ and $\boldsymbol{\theta_{\tilde{C}}}$) is given by Bayes' rule from the sampling distribution in equation 3 and the prior in equation 4:

$$p(\theta_c|\mathbf{u}, \boldsymbol{\theta_{\tilde{C}}}) = \left(\theta_c^{(\alpha+\sum u)-1} e^{-\theta_c(\beta+\frac{\lambda}{\theta_c})}\right) \cdot c(\mathbf{u}, \alpha, \beta) \tag{5}$$

$$\sim \text{gamma}(\alpha + \sum \mathbf{u}, \beta + \sum \frac{\boldsymbol{\lambda}}{\theta_c}) \tag{6}$$

Note that $p(\mathbf{u})$, the probability of observing the sample that pertains to θ_c, does not depend on the value of θ_c and so $c(\mathbf{u}, \alpha, \beta)$ is a normalizing constant.

[5]Most software, including MATLAB, use an inverse rate parameter, referred to as a scale parameter.
[6]The gamma distribution has mean $\frac{\alpha}{\beta}$ and variance $\frac{\alpha}{\beta^2}$.

2.3 Markov Chain Monte Carlo Sampler

The model's parameters can be inferred using a Markov chain Monte Carlo approach. We compute M Markov chains in parallel. The inference procedure for each Markov chain is as follows:

1. Initialize the Markov chain: Draw each θ_i from a Gamma distribution using its priors, $\theta_c = \text{gamma}(\alpha, \beta)$
2. Gibbs sample from the conditional posterior: For each θ_c in turn, compute $\sum \mathbf{u}$ from the data and $\sum \boldsymbol{\theta}_{\tilde{\boldsymbol{c}}}$ using the most recently inferred values, then draw $\theta_c = \text{gamma}(\alpha + \sum \mathbf{u}, \beta + \sum \boldsymbol{\theta}_{\tilde{\boldsymbol{c}}})$.
3. Iterate: repeat (2) for R steps.[7]

Then, once every chain is complete:

4. Post-process: Remove warm-up iterations and, if necessary, thin out the remaining steps to eliminate any remaining serial correlation. Add new iterations so that there are $R \times M$ samples for every parameter.

The total of observed data about θ_c, $\sum \mathbf{u}$, does not change across iterations and can be pre-computed. However, the sum of the current inferred values of conditioning factors, $\sum \boldsymbol{\theta}_{\tilde{\boldsymbol{c}}}$, (and so the inferred Poisson parameters and predicted mean values, $\boldsymbol{\lambda}$) do change from iteration to iteration. Calculating this "sum of the lambdas" is the primary computational expense in computing the model.

2.4 Categorical Variables and Sampling Blocks

The fundamental factors, or 'features' in machine learning parlance, inferred by the model are coefficients for indicator variables. For sampling efficiency, we organize factors into mutually exclusive categories with values $c \in C$ so that together the categories constitute a categorical variable, C, and a sampling block, $b \in 1, \ldots, B$. Because the categories are mutually exclusive within each sampling block, they are not conditioned on each other. Put

[7] The software provides an option to iterate each of the R steps until a convergence criteria is achieved, or a maximum of S times, to eliminate serial correlation.

another way, two factors c and c' can belong to the same block b if the parts of $\boldsymbol{\lambda}$ affected by c are disjoint from those affected by c'. For example, a sampling block might have a categorical variable to describe calendar months $c \in \{1, \ldots, 12\}$, and each observation would belong to only one month. Equation (2) is then:

$$\boldsymbol{\lambda} = \prod_{b \in B, c \in C} \boldsymbol{\theta}_{b,c} \tag{7}$$

We can then perform stage 2 in the inference procedure – computing the sum of the lambdas and drawing from the gamma distribution – for an entire block simultaneously, as a "block update", using vectors of values. To do this requires three large vectors, each with a maximum length $|E|$, as well various other (comparatively immaterially small) vectors. The three large vectors are $\boldsymbol{\lambda}$, and two vectors that together map E to the elements of $\boldsymbol{C}$: $E_{C \to E}$, which contains values of E, and $C_{C \to E}$, which contains values of C.

Denoting $\boldsymbol{\theta}_b$ as the vector of thetas that belong to block b, $\boldsymbol{\lambda}(E_{C \to E})$ as the subvector of lambda that is affected by C, and $\boldsymbol{\theta}_b(C_{C \to E})$ as the projection of b's thetas into E-space, the first two steps of the inference procedure are then:

1. Initialize each chain. Draw $\boldsymbol{\theta}$ and compute $\boldsymbol{\lambda}$:
   (a) Draw each $\boldsymbol{\theta}_{b,c}$ from the gamma distribution using its priors, putting the results in $\boldsymbol{\theta}$.
   (b) Set $\boldsymbol{\lambda}$ to a vector of ones of length $|E|$.
   (c) For each b, update the appropriate part of lambda: $\boldsymbol{\lambda}(E_{C \to E}) = \boldsymbol{\lambda}(E_{C \to E}) \odot \boldsymbol{\theta}_b(C_{C \to E})$.
2. Perform a step for a chain. For each block, b, in turn:
   (a) Calculate $\sum \boldsymbol{\theta}_{-b}$: sum $\boldsymbol{\lambda}(E_{C \to E})$ over the distinct c found in $C_{C \to E}$, and Hadamard divide the resulting vector by $\boldsymbol{\theta}_b$.
   (b) Store the old thetas for b, denoted $\boldsymbol{\theta}_{\tilde{b}}$.
   (c) Compute vectors of α and β using the priors, the sum of observed data influenced by the block (i.e., $\sum \mathbf{u}(E_{C \to E})$), and $\sum \boldsymbol{\theta}_{-b}$.

(d) Draw a vector of new thetas for block b from the Gamma distribution using α and β, and update $\boldsymbol{\theta}$.

(e) Update the lambda vector: $\boldsymbol{\lambda}(E_{C\to E}) = \boldsymbol{\lambda}(E_{C\to E}) \odot \boldsymbol{\theta}_b(C_{C\to E}) \oslash \boldsymbol{\theta}_{\tilde{b}}(C_{C\to E})$

For a Lambda-in-Memory (LIM) sampler design, the three large block-update vectors must be held concurrently in memory. Single precision and TensorFloat-32 encoded numbers use 4 bytes, so the memory requirement to perform a LIM block update is at least $12 \cdot |\boldsymbol{E}|$ bytes. (The memory footprint of other vectors is immaterial when the data is much bigger than the number of inferred factors.) Available GPUs and EC2 instances currently have 40Gb and 1Tb of memory, respectively. Accordingly, the maximum size of the model computed with a LIM block-update sampler is around 1×10^{10} using GPUs and 2.5×10^{12} using CPUs.[8] Eight Markov chains can be run in parallel as 8-GPU EC2 instances and clusters of nodes are readily available.

2.5 Partitions, Topology, and Grain

A partition divides up a space made up of one or more dimensions, $\{d\}$, into vectors of category values $\boldsymbol{c}_{\{d\}}$, using vectors of dimensional labels $\{\boldsymbol{l}_d\}$. (The dimensional labels are mapped into indices, $\boldsymbol{e}_d$, using the dimensional vectors.) For example, a single-dimension partition that divides up the first dimension would have a vector of category values $\boldsymbol{c}_1$ and a vector of labels $\boldsymbol{l}_1$; and a two-dimension partition that divides up the first and third dimensions would have a vector of category values $\boldsymbol{c}_{1,3}$ and two vectors of labels $\{\boldsymbol{l}_1, \boldsymbol{l}_3\}$.

There are two special types of one-dimensional partitions:

- A degenerate partition has the same category value (typically 1) for every index of a dimension: $c_d = 1 \; \forall e_d \in \boldsymbol{e}_d$
- A complete partition has a unique category value (typically the same as the index value) for every index of a dimension: $c_d = e_d \; \forall e_d \in \boldsymbol{e}_d$

A block's categorical variable C is the cartesian product of category values from partitions covering all of the model's dimensions: $C = \bigtimes_{d=1}^{D} \{\boldsymbol{c}_d\}$. Usually, modelers specify partitions

[8] For models inferring daily demand into America's 41,690 zip codes based on five years of data, other dimensions can have a length-product up to about 130 and 3,300, respectively.

to cover some dimension(s) and then complete the block's specification using degenerate partitions for the other dimension(s).

A block is 'normalized' if its categories do not cover the entire training space $\mathbf{E}^T$ of the model. Inferred parameters are identified for normalized blocks, with the 'omitted factor' corresponding to the uncoded space having a parameter value equal to one. Inferred parameters for unnormalized blocks provide relative effects (i.e., the magnitude of $\frac{c}{c'}$ is identified).

For example, consider a block for a quarter-of-the-year factor in a two-dimensional product-time model. This block would be encoded using a degenerate partition in the product dimension, $c_1 = \{1\}$, and have values $c_2 = \{1, 2, 3, 4\}$ in sequence across the entire time dimension. The categorical variable would then be $C = c_1 \times c_2 = \{1, 2, 3, 4\}$, and the corresponding vector of category values would be, say, $\mathbf{C} = (1, 2, 3, 4, \ldots)$. As $|\mathbf{C}| = |\mathbf{E}^T|$, the block would not be normalized. It could be normalized by omitting the encoding for, say, Q1.

Users may wish to construct variables based on overlapping factors. In such cases, the variable must use multiple blocks. Recall that categories must be mutually exclusive within blocks; each category must pertain to disjoint parts of the space and different lambdas.[9] A leading example would be to create growth variables, where one factor applies to all time, the second to all but the first period, and so on, with many factors multiplied to produce the effect of the last period. Such a variable and its interpretation must then span as many blocks as there are periods to ensure disjointness.

The collection of disjoint subspaces created by a model's partitions defines its topology. The computation time and memory needed to compute the model is a function of the topology's overall size and 'complexity.'

The size of the finest disjoint subspaces in each dimension defines the model's grain. The outcomes of the model, $\boldsymbol{\theta}$ and so $\boldsymbol{\lambda}$, are at the level of the grain. So, tallying thetas and lambdas to any higher level of aggregation involves aggregating disjoint spaces and produces consistent inference. For instance, if the grain in the time dimension is days and the region dimension is zip5, then the lambdas can be aggregated to create forecasts for weeks and zip2, year and nationwide, et cetera.

[9] This is guaranteed by construction if each partition's categories pertain to disjoint parts of its dimension(s).

2.6 Extending the Model: Families

An extension to the base model adds hierarchical priors. In this extension, priors have a family, f, which belongs to one of three types.

The base model, described in the Bayesian model section, uses type 1 families. It has two levels: Poisson-distributed lambdas on top given by equations **??** and 2, with Gibbs-sampled conditional gamma distributed thetas given by equation 6 in the level below. The priors for the conditional gamma distribution are provided by the user as hyperparameters.

With type 2 and 3 families, the model adds a third level to the hierarchy. Priors for the conditional gamma are learned from the data, and the user provides hyperparameters for one or two normal distributions, from which these priors are sampled in a new bottom level. The hyperparameters are means and variances for w and z, which are transformations of α and β. w is the log mean of the Gamma distribution, $\log(\frac{\alpha}{\beta})$, and z is the log coefficient of variation, the standard deviation divided by the mean, $-\frac{1}{2}\log(\alpha)$.

We use a hybrid Gaussian random walk proposal distribution in a Metropolis-Hastings algorithm to sample for α and β (or equivalently for w and z). Specifically, we draw new parameters from a (multivariate) normal distribution, where the mean is the parameter value from the previous step in the Markov chain. When the log conditional posterior kernel is concave, the Gaussian variance is computed from the (second derivitive of the) kernel. When the log conditional posterior kernel is not concave, we instead use the prior(s) (i.e., σ for type 2, and σ, τ for type 3) for the Gaussian variance.

2.6.1 Type 2 Families

For a type 2 family, the user provides a mean, μ, and variance, σ, for just z:

$$p(z|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{z--\mu}{\sigma}\right)^2} \tag{8}$$

The model then forces the mean of the gamma distribution equal to one (i.e., $\frac{\alpha}{\beta} = 1 \implies \alpha = \beta$). Provided that the thetas for type 2 families are then excluded from the computation of the lambdas, or equivalently if each $\theta = 1$, the posterior distribution is a negative binomial with mean λ and variance $\lambda \cdot (\frac{\lambda+\alpha}{\alpha})$:

$$p(u|\lambda) \sim NB\left(\alpha, \frac{\lambda+\alpha}{\lambda}\right) \tag{9}$$

The log conditional posterior kernel for a type 2 family is then:

$$\begin{aligned} k = &\sum_n \left(\sum u \cdot \log\theta - (\sum \frac{\lambda}{\theta}) \cdot \theta\right) + \\ &\sum (\alpha \cdot \log\alpha - \log\Gamma(\alpha) + (\alpha-1)\cdot\log\theta - \alpha\theta) + \\ &- \frac{(z-\mu)^2}{2\sigma^2} \end{aligned} \tag{10}$$

2.6.2 Type 3 Families

For type 3 families, the user provides means, ν and μ, and variances, τ and σ, for both w and z. $p(z|\mu,\sigma)$ is as in equation 8 and $p(w|\nu,\tau)$ is:

$$p(w|\nu,\tau) = \frac{1}{\tau\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{w-\nu}{\tau}\right)^2} \tag{11}$$

The log conditional posterior kernel for a type 3 family is then:

$$\begin{aligned} k = &\sum_n \left(\sum u \cdot \log\theta - (\sum \frac{\lambda}{\theta}) \cdot \theta\right) + \\ &\sum (\alpha \cdot \log\beta - \log\Gamma(\alpha) + (\alpha-1)\cdot\log\theta - \beta\theta) + \\ &- \left(\frac{(w-\nu)^2}{2\tau^2} + \frac{(z-\mu)^2}{2\sigma^2}\right) \end{aligned} \tag{12}$$

Type 3 families allow the model to include random walks with drift and other advanced functionality.

3 Implementating the Model

The CatFish model is available in software from:

- The CatFish webpage: https://www.petabyteeconomics.com/catfish.html
- GitHub: https://github.com/petabyteeconomics/CatFish

This software has three parts: i) building the topology, ii) parameter inference, and iii) using the inferred parameters to produce forecasts and conduct analysis. The software has been written in MATLAB, as this is the most commonly used language by Bayesian econometricians. However, the software design is object-oriented, language agnostic, and tailored towards big data.

Parts (i) and (iii) of the software are essentially sequences of dataframe transformations, which can be done in SQL databases or Spark-based environments. The MATLAB code stores this data as tables, which can be imported and exported to parquet files, tab or comma delimited text files, or through ODBC/JDBC interfaces.

Parameter inference – part (ii) of the software – currently uses vector-based in-memory storage and transformations for efficiency.[10] These vectors can use single, double, or tensorfloat-32 precision values, suitable for GPU processing. The vectors and methods used in the software are widely supported, including Python's numpy library.

3.1 Building the Topology

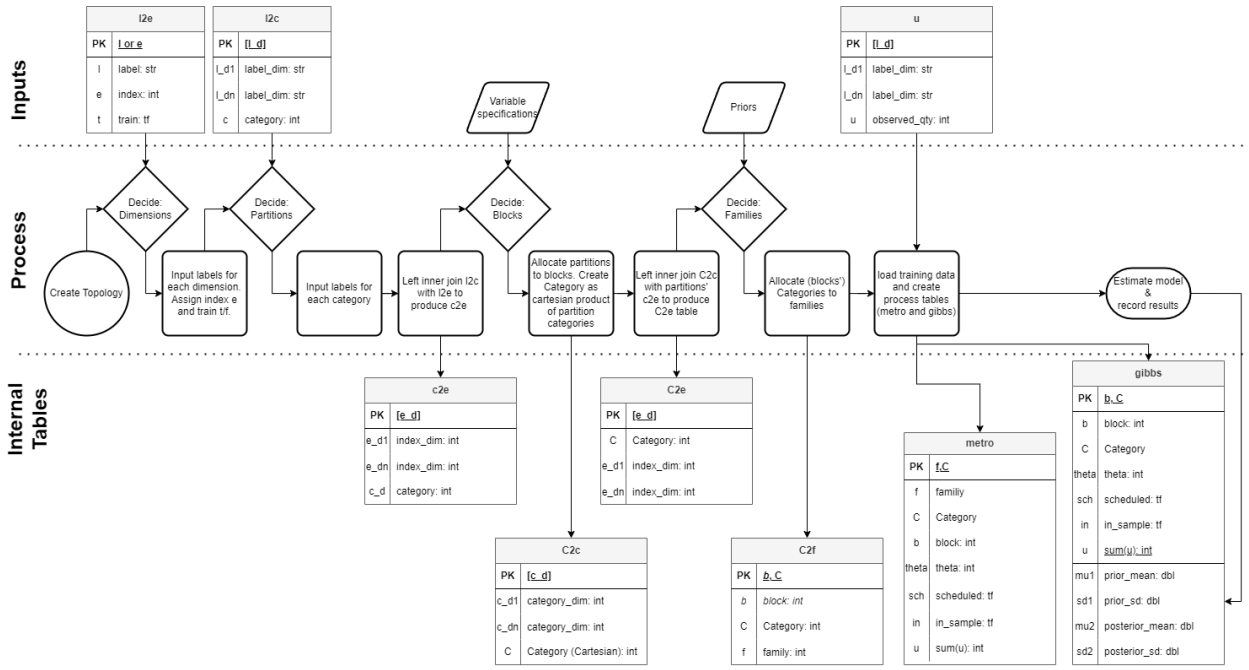
Figure 1 shows the process for building a topology consisting of dimensions, partitions, blocks, and families. The data is stored in tables (shown in grey). The input stream in the top third of the diagram shows data imported from either files or functions. The lower third of the diagram shows tables generated from transformations, which can be exported as files.

3.1.1 Startup Cities Example

The script Example1.m builds a three-dimensional model inferring the demand for new venture capital financing in 198 U.S. "Startup Cities" for three industries (IT, life sciences, and non-high-tech) for 45 years from 1981 to 2025. It uses categorical variables (i.e., fixed effects) for cities, industries, state-decade interactions, quantity and quality of anchor funds

[10] This choice inherently limits the model's maximum size but makes it fast and flexible for research and development. However, parameter inference does not need to be vector-based and can use Spark Scala or other big data languages and environments for production systems.

Figure 1: Topology build process with associated data tables (grey)



(if any) present in a city-year, and a five-year growth trend. It has a forecast date of 2020, so there are 40 years of training data and a five-year forecast period.

In this section, we describe how to build the topology using the Startup Cities model as an example. Table 1 shows three dimensions' $l \leftrightarrow e$ tables (referred to as $l2e$ tables in diagrams and code), which map labels, l_d to indices, e_d, and vice versa.[11] The indices and training indicators, t_d, in these tables can be generated in software if not provided in the data – the model and software support demarking any subsets of any dimensions as training or forecast. However, for simplicity, the software provides a method for marking indices ahead of some cutoff point (in one or more dimensions) as forecast data. This method is often used to set a forecast date (FCD in code) in the time dimension. The dimensions are numbered $d = \{1, \ldots, D\}$ in the order that they are loaded.

Table 2 shows three example partitions' $l \rightarrow c$ tables. Like dimensions, partitions can be loaded from data or generated by functions and numbered in the order they are loaded. Table

[11] The arrows in the table names show the typical direction of lookups. The $l \leftrightarrow e$ table is used to look up labels from indices and vice versa. Some tables, including the $l \leftrightarrow e$ table, also include other variables, and subscripts are suppressed in table names.

Table 1: Three example dimension tables

l_1 (City)	e_1	t_1
Addison, TX	1	1
Alameda, CA	2	1
Albuquerque, NM	3	1
...	...	...
Wilmington, DE	196	1
Wilmington, MA	197	1
Woburn, MA	198	1

l_2 (Industry)	e_2	t_2
Information Technology	1	1
Medical/Health/Life Science	2	1
Non-High Technology	3	1

l_3 (Year)	e_3	t_3
1981	1	1
1982	2	1
1983	3	1
...	...	...
2018	38	0
2019	39	0
2020	40	0

2 shows the 5^{th}, 6^{th}, and 8^{th} partitions. The 5^{th} and 8^{th} partitions use a single dimension (industry: $d = 2$, and time: $d = 3$, respectively) and so have only one label column. The 6^{th} partition uses two dimensions ($d = 1$ and $d = 3$) and so has two label columns.

The 5^{th} partition divides the time dimension, which is in years, into decades. The 6^{th} partition assigns city-years that have "anchor funds" to one of 4 categories, depending on how many they have and their quality. City-years without an anchor fund are not assigned to a category. The 8^{th} partition is degenerate; it assigns all dimension labels to a single category value (i.e., 1).

In general, there is no need for partitions to cover dimensions fully, and the software provides methods to 'normalize' partitions by removing a category. Degenerate and complete partitions are exceptions; they must fully cover the dimension. Complete partitions assign each label to a unique value. Each of the dimension tables in Table 1 could be used as a complete partition by declaring the index as the category number.

Partitions' categories must bear upon disjoint spaces. For instance, one cannot have a partition that simultaneously divides the time dimension into days of the week and holiday days, as holiday days fall on days of the week. In some cases, users may wish to create complicated variables that overlap, requiring separate partitions and so blocks. A leading example concerns "scale" (also called "cascade") factors. Scale factors can be used to create growth trends and (with type 3 families) random walks with drift.

The Startup Cities example includes a scale factor that changes the level of the forecast every five years. It has one category (in partition 9 and block 6) covering every five-year period from 1986 to 2025, another (in partition 10 and block 7) covering 1991 to 2025, and so

Table 2: Three example partitions

$p = 5$ (Decade)

l_3	c_3
1981	1
1982	1
1983	1
...	...
2018	4
2019	4
2020	5

$p = 6$ (Fund Quartile)

l_1	l_3	$c_{1\times3}$
Alexandria, VA	1987	1
Alexandria, VA	1988	1
Alexandria, VA	1989	1
...	...	...
Wellesley, MA	2016	3
Wellesley, MA	2017	2
Wellesley, MA	2018	2

$p = 8$ (Degenerate)

l_2 (Industry)	c_2
Information Technology	1
Medical/Health/Life Science	1
Non-High Technology	1

on, with the last one (in partition 15 and block 13) covering 2021 to 2025. Table 3 provides a stylized depiction of this factor. The effect of the n $^{\text{th}}$ factor of this scale variable then is the sum of the effects of the first n categories. However, to ensure disjointness, each of the n categories is in a different partition and block.

Table 3: Example of a "scale" factor using multiple partitions and blocks

	period 1	period 2	period 3	period 4
p and b	1	1	1	1
$p+1$ and $b+1$		1	1	1
$p+2$ and $b+2$			1	1
$p+3$ and $b+3$				1

Raw partition data can contain labels not appearing in the dimension tables. Partitions are inner-joined with their dimension tables using the labels to create $c \leftrightarrow e$ tables (not shown to save space).[12]

A block's Categories (uppercase), C, is a sequential numeric labeling of the cartesian product of partitions' categories (lowercase), c_d, across all dimensions: $\mathbf{C} = \{\mathbf{c_1} \times \mathbf{c_2} \times \ldots \mathbf{c_D}\}$. The $C \rightarrow c$ table for block 5 is shown in table 4. Degenerate blocks are used to cover dimensions that do not interact with other partitions. For example, to make the "Anchor Fund" block ($b = 5$), interact the anchor fund partition ($p = 6$), which is defined in terms of dimensions 1 and 3, with a degenerate partition that covers dimension 2 (i.e., $p = 8$,

[12]This allows greater reuse of partition tables: Dimensions restrict partitions to the topology's space.

shown in Table 2). Note that most $C \to c$ tables are made from the cartesian product of D partitions, as most partitions divide up a single dimension.

Table 4: An example block's ($b = 5$) $C \to c$ table

$b = 5$ (Anchor Fund)

C	$c_{1\times3}$	c_2
1	1	1
2	2	1
3	3	1
4	4	1

Inner joining block Category definitions inherent in $C \to c$ tables to partitions' $c \leftrightarrow e$ tables creates $C \leftrightarrow e$ tables. $C \leftrightarrow e$ tables are then used to create $C \leftrightarrow E$ vectors and their refinements, C2E Gibbs In-sample and C2E Metro In-sample, which are used in the inference process described in the next section. Examples are shown in Table 5.

Table 5: Example $C \leftrightarrow e$ tables and $C \leftrightarrow E$ vectors for block 5

C2e (Table)

C	e_1	e_2	e_3
1	4	1	7
1	4	2	7
1	4	3	7
...	...	...	...
2	190	1	38
2	190	2	38
2	190	3	38

C2E (Vector)

C
0
...
1
0
2
...
0

C2E Gibbs In-sample

C	E
1	3568
1	3766
1	3964
...	...
2	22168
2	22366
2	22564

$\mathbf{E}$ is the linear vector equivalent of $\{\mathbf{e_1} \times \mathbf{e_2} \times \ldots \mathbf{e_D}\}$, so each combination of dimensional indices maps to a location in a vector that is 26,730 (i.e., $198 \times 3 \times 45$) long. Put another way, each $C \leftrightarrow e$ table is also a $C \leftrightarrow E$ vector of length 26,730 with $C = 0$ where no Category applies. Moreover, not all C are necessarily Gibbs or metropolis sampled, and not all E (and so C) are in-sample. Accordingly, we can create two vectors (or equivalently a two-column table) for "C2E Gibbs In-sample": one for the non-zero C and another for the E where they

apply in-sample. Likewise, for metropolis sampled C. Such vectors are the same length and generally much shorter than the length of E.[13]

The topology is completed by specifying the priors for each family and assigning the families to blocks' Categories in $C \to f$ tables. Note that for type 1 families the priors are, α and β, whereas for type 2 families they are, μ and σ, and for type 3 families they are ν, μ, τ, and σ. Table 6 provides an example.

Table 6: Example priors and $C \to f$ table

Priors (Input)

f	type	p1	p2	p3	p4
1	1	0.6	0.6		
2	1	1.25	1.25		
3	1	5	5		
4	2	-1.7	0.5		
5	3	0.15	0.1	-0.7	0.5

C2f (Table)

b	C	f
5	1	1
5	2	1
5	3	1
5	4	1

Type 1 families can only be Gibbs sampled, as they have no hierarchical priors. However, for type 2 and 3 families, Gibbs sampling is optional. By default, type 2 families are only metropolis sampled – so they can be used to transform mixed Poisson distributions to mixed negative binomials – and type 3 families are both metropolis and Gibbs sampled. These defaults can be overridden in the code.

The topology is then recorded in two tables stored at the model level: "gibbs" and "metro".[14] These tables' component parts (broken up by b and f, respectively) are also stored in the blocks and families. Table 7 shows example gibbs and metro tables for blocks 5 (for Gibbs) and 6-13 (for metropolis). The theta column provides a global variable number for each b, C combination and is zero in the metropolis table when the Category isn't Gibbs sampled. The 'in' column denotes whether a block's Category is ever in-sample, and the 'sch.' column denotes whether metropolis sampling will be run when the sampler reaches the indicated block.

[13] These vectors are at most 23,760 long (i.e., $198 \times 3 \times 40$), as the training period is 40 years in the example.

[14] These tables are also used to store aggregate statistics once the inference is complete.

Table 7: Example Gibbs and Metropolis Topology Summaries

Gibbs Table

theta	b	C	in	f
345	5	1	1	1
346	5	2	1	1
347	5	3	1	1
348	5	4	1	1
349	5	5	1	1

Metropolis Table

theta	b	C	in	f	sch.
350	6	1	1	5	1
351	7	1	1	5	0
...	...	...	...	...	...
356	12	1	1	5	0
357	13	1	0	5	0

Finally, the model needs training data. This data is loaded from a file that specifies demand (or any count variable) for each observation, where observations are identified using dimension labels, $\{l_1, \ldots, l_D\}$. An example is provided in Table 8. The training data does not need to include zeros – these will be added automatically within the space **E**. Likewise, rows pertaining to observations outside of the space **E** are ignored, allowing the same input file to be used in multiple contexts. The variable 'u', reflecting the sum of demand for each block's category, is computed by the software and stored in the blocks' gibbs tables and the families' metro tables.

Table 8: Example Observed Demand Data

City	Industry	Year	Demand
Wellesley, MA	Medical/Health/Life Science	1984	1
Santa Monica, CA	Information Technology	2004	2
Broomfield, CO	Non-High Technology	1991	1
...	...	...	...
Chicago, IL	Information Technology	2015	23
Burlingame, CA	Medical/Health/Life Science	2014	1
Burlingame, CA	Information Technology	1995	2

3.2 Parameter Inference

Once the topology has been built, the inference is straightforward: For each chain $m \in M$, for each iteration $r \in R$, metropolis sample each applicable family, and then Gibbs sample

each applicable block. The inference procedure is shown in a swimlane diagram in Figure 2.

The first two processing steps in the model swimlane – "build the sampling reference tables" and "build linear C2E, restrict to Gibbs & in-sample" – were described in the previous section. The initialization step draws α and β from priors for each family, f, draws each θ from a Gamma distribution using the appropriate priors, and computes $\boldsymbol{\lambda}$ using the blocks' C2E Gibbs In-Sample vectors and appropriate thetas.[15] The sampling then follows the procedure described in section 2.4. It is convenient to cache the inferred parameters ($\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\theta}$) and $\boldsymbol{\lambda}$ for each of the M chains and write the parameters to storage after each iteration.

This procedure might not infer the parameter values for some blocks' Categories from the training data. The omissions happen when a Category is Gibbs-sampled but not in-sample, which is particularly common in conjunction with type 3 factors. For example, suppose that four Categories apply to different time regimes, so that the first three are in-sample, but the fourth regime doesn't begin until after the forecast date. One could infer the correct α and β from which to draw $\theta_1, \ldots, \theta_4$ using a type 3 factor. However, there is only data to override the priors for the first three thetas. Once the in-sample inference is complete, θ_4 can be drawn from its (endogenously determined type 3) prior. The software provides a procedure to do $R \times M$ out-of-sample draws and record the corresponding thetas where applicable.

3.3 Producing Forecasts

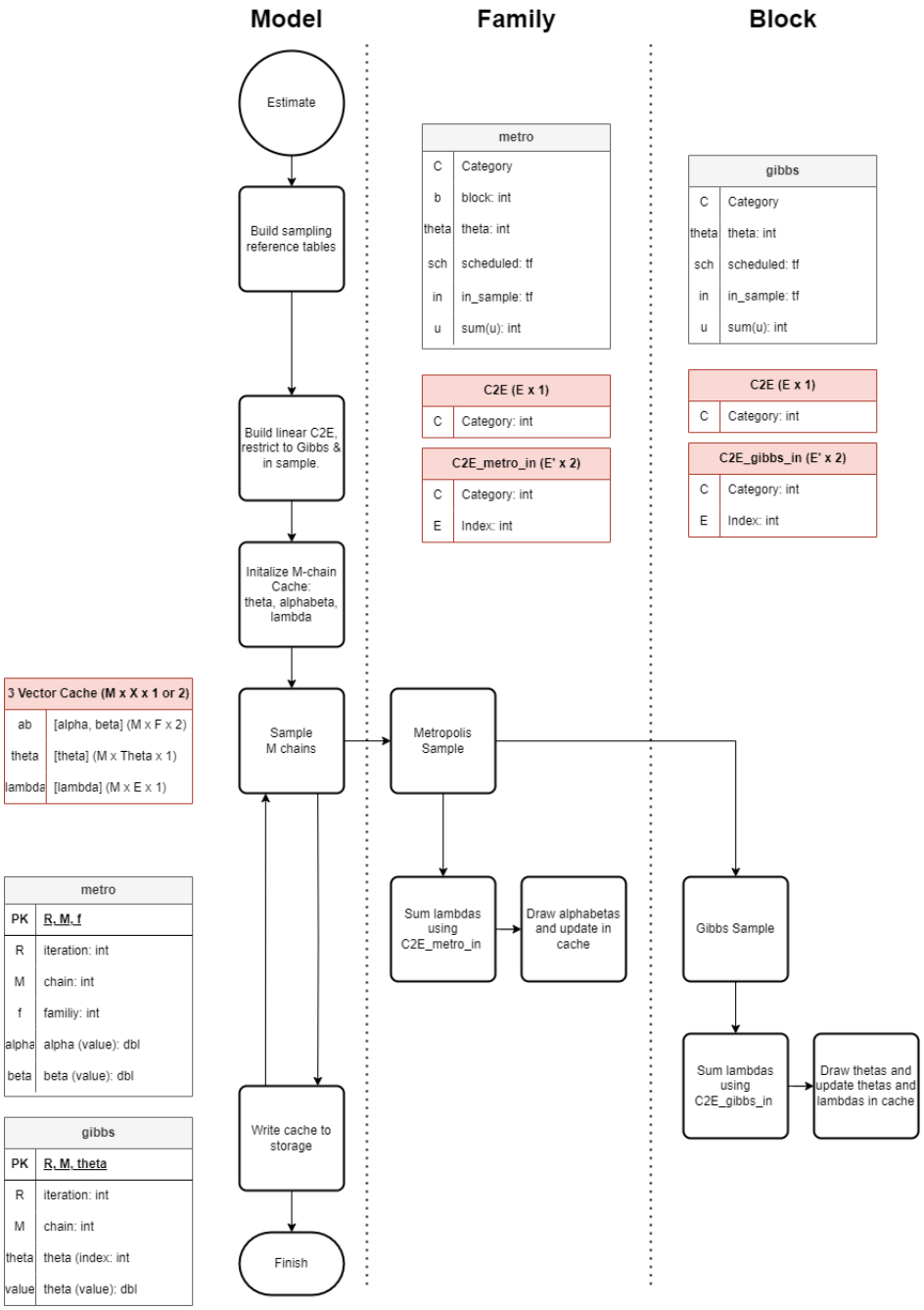
The R iterations of M chains provide $R \times M$ independent samples of the infered parameters (i.e., $\boldsymbol{\alpha}, \boldsymbol{\beta}$ for metropolis sampled families, and $\boldsymbol{\theta}$ for Gibbs in-sample Categories). Each sample of the inferred parameters can be projected back into a subset of the space $\mathbf{E}$, using the $C \leftrightarrow E$ vectors, to create $R \times M$ lambda vectors, which we refer to as the lambda ensemble. Depending on the subset of $\mathbf{E}$ chosen, a lambda ensemble can pertain to in-sample, out-of-sample, or both.

There are then several methods to produce forecasts. The most common of these methods are:

- Making one draw from each lambda in the ensemble creates an $R \times M$ outcome en-

[15] We refer to both the index of each b,C and the inferred value of the corresponding parameter as theta, depending on the context.

Figure 2: Inference process with data tables (grey) and vectors (red)



semble, which provides a full distribution of outcomes.

- The lambda ensemble is a mixed Poisson distribution. Users can compute quantiles or other statistics as appropriate for their needs.
- The mean of Poisson distribution is λ, so the mean of the lambda ensemble is the mean forecast.
- The lambda ensemble can be combined with the $M \times R$ draws of α for type 2 families to create mixed negative binomials.

CatFish's forecasts can then be aggregated to any grain and used as needed, and its inferred parameters provide meaningful insight into the determinants of your demand!

4 Appendix

Figure 3 shows a class (in blue) diagram for the CatFish v1.0 software, along with the primary data table (in grey) definitions.

Figure 3: CatFish v1.0 class (blue) diagram with data tables (grey)