

A APPENDIX

A.1 DATA

A.1.1 HONEY BEE DATASET

Two honey bee colonies with individual markers attached to each individual were recorded. Timestamps, positions, and unique identifiers of all individuals from these colonies were obtained (details redacted during double-blind peer review). See Table 1 for dates and number of individuals.

Temporal affinity matrices were derived from this data as follows: For each day, counts of proximity contact events were extracted. Two individuals were defined to be in proximity if their markers' positions had an euclidean distance of less than 2 cm for at least 0.9 seconds. The daily affinity between two individuals i and j based on their counts of proximity events $p_{t,i,j}$ at day t was then computed as: $\mathbf{A}_{t,i,j} = \log(1 + p_{t,i,j})$, $\mathbf{A} \in \mathbb{R}^{N_t \times N_i \times N_i}$, where N_t is the number of days and N_i the number of individuals in the dataset.

The datasets are open access and will be deanonymized with more details after the double-blind peer review: Anonymous (2020)

A.1.2 SYNTHETIC DATASET

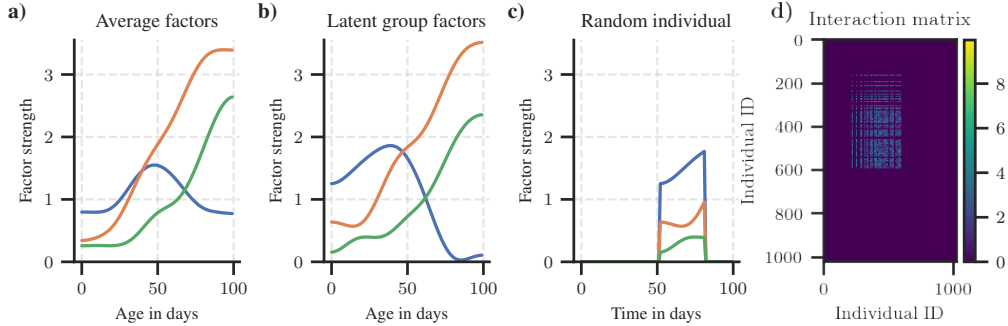


Figure 7: Example of one generated synthetic dataset. **a)** Common lifetime trajectory of all entities. **b)** The lifetime trajectory of one random latent group. **c)** The factors of one individual in the dataset. **d)** Generated interaction matrix for one day.

We generated synthetic datasets to evaluate whether the model is able to identify groups of individuals with common latent factors that determine their interaction frequencies. We model a common lifetime trajectory for all entities using a smoothed Gaussian random walk in \mathbb{R}^+ with $\sigma_{\text{walk}} = 1$ for the steps of the random walk and $\sigma_{\text{smoothing}} = 10$ for the Gaussian smoothing kernel. See Figure 7 a) for one example of a generated lifetime trajectory with three factors. We then randomly create latent groups by creating smoothed Gaussian random walks that define how these groups differ from the common lifetime trajectory. See Figure 7 b) for the lifetime trajectory of one latent group. For each group, we also define different expected mean lifetimes. We set the average lifetime of an entity to 30 days with a standard deviation of 10 days. We then randomly assign 1024 individuals to those latent groups and also assign random dates of emergence and disappearance of these individuals in the dataset. We then compute the individual factor trajectories for each individual, as can be seen in Figure 7 c). Finally, for 100 days of simulated data, we generate interaction matrices by computing the dot products of the factors of all individuals (Figure 7 d).

A.2 MODEL DETAILS AND HYPERPARAMETERS

A.2.1 REGULARIZATION TERMS

$$R_{\text{embeddings}} = \lambda_{\text{embeddings}} N_i^{-1} \sum_{i=0}^{N_i} \sum_{k=0}^K |\phi_{i,k}| \quad (8)$$

$$R_f = \lambda_f N_i^{-1} \sum_{i=0}^{N_i} \sum_{t=0}^{N_t} f^+(t, i) \quad (9)$$

$$R_{\text{basis}} = \lambda_{\text{basis}} N_a^{-1} \sum_{a=0}^{N_a} \sum_{k=0}^K |b_k(a)| \quad N_a = 60 \quad (10)$$

where N_a can be any number higher than the oldest individual in the dataset at any time.

$$R_{\text{adv}} = \lambda_{\text{adv}} N_i^{-1} \sum_{i=0}^{N_i} \log \left(\frac{\exp(\mathbf{d}(\phi_i)_{c[i]})}{\sum_d^{N_d} \exp(\mathbf{d}(\phi_i)_d)} \right) \quad (11)$$

where $\mathbf{d}(\phi_i)$ is the probability distribution returned by the discriminative network, $c[i]$ the day the entity i emerged in the dataset, and N_d the number of days in the dataset.

See appendix A.4 for an ablation study of the effect of these regularization term on the results.

A.2.2 NETWORK ARCHITECTURE

We use the following neural network architecture for the functions $\mathbf{m}(c(t, i))$, $\mathbf{b}(c(t, i))$, and $\mathbf{d}(\phi_i)$:

$$\text{Linear}(N_{in}, N_h) \rightarrow \text{LReLU} \rightarrow \underbrace{\text{Linear}(N_h, N_h) \rightarrow \text{LReLU}}_{N_l\text{-times}} \rightarrow \text{Linear}(N_h, N_{out})$$

where *Linear* is an affine transformation $f(x) = Ax + b$ and $\alpha = 0.3$ for the Leaky ReLU activation function. For $\mathbf{m}(c(t, i))$ and $\mathbf{b}(c(t, i))$: $N_{in} = 1$ (the individuals' ages). For $\mathbf{m}(c(t, i))$: $N_{out} = M$ and for $\mathbf{b}(c(t, i))$: $N_{out} = MK$. For $\mathbf{d}(\phi_i)$: $N_{in} = K$ and $N_{out} = N_{\text{labels}}$.

A.2.3 HYPERPARAMETERS

Table 4: Hyperparameters used in the evaluated models (if not stated otherwise)

Parameter	Value	Description
N_l	3	Number of hidden layers
N_h	64	Hidden layer size
M	8	Number of factors
K	16	Number of individuality basis function
N_{labels}	100	Number of cohorts
N_{batch}	128	Minibatch size
N_{steps}	100 000	Number of training iterations
λ_f	0.1	Factor L_1 regularization
λ_{adv}	0.1	Factor L_1 regularization
λ_{basis}	0.01	Basis function L_1 regularization
$\lambda_{\text{embeddings}}$	0.1	Embedding L_1 regularization

The scaling factors for the regularization losses (see Table 4) were manually selected by increasing each factor until it prevented the model from converging (i.e. the reconstruction loss of the full model $\mathbf{f}^+(t, i)$ did not improve on the age model \mathbf{m}). This initial set of hyperparameters was then manually refined such that each regularization loss was still effective (e.g. the factor regularization loss L_f reduced the total number of factors effectively used by the model). Overfitting was not a concern because the model is fitted unsupervised and the goal of the hyperparameter selection was to find a set of parameters that is sparse and interpretable, and not to increase the predictive capabilities of the learned factors.

A.2.4 MODEL FITTING

The model was fitted using a single GPU (GeForce RTX 2080 Ti). A training run consisting of 200 000 minibatches finished in about six hours. Due to overhead in data loading and preprocessing, up to three training runs could be executed in parallel without negatively affecting the runtime.

Algorithm 1: Training loop

```

for  $b = 0$  to  $N_{steps}$  do
  Draw minibatch of  $N_{batch}$  random individuals
  Compute  $\hat{\mathbf{A}}_{t,i,j} \forall t$  for individuals  $i, j$  in minibatch
  Compute model training loss: equation 5 + regularization
  Update parameters for  $\mathbf{m}(c(t, i))$ ,  $\mathbf{b}(c(t, i))$ , and  $\mathbf{d}(\phi_i)$ 
  Compute  $\hat{l}_i = \mathbf{d}(\phi_i)$  for individuals  $i$  in minibatch
  Compute discriminator training loss
  Update parameters for  $\mathbf{d}(\phi_i)$ 
end for

```

A.3 TRADEOFF BETWEEN TEMPORAL CONSISTENCY AND SEMANTIC MEANINGFULNESS

We performed a grid search over the hyperparameters λ_f , λ_{adv} , λ_{basis} , and $\lambda_{embeddings}$ to evaluate whether models can only be either semantically meaningful or temporally consistent. For this analysis, we define *Semantic meaningfulness* as the sum of the *Rhythmicity* and *Mortality* metrics introduced in section 2.5. We find that models that are very temporally consistent fail to learn semantically meaningful information. Interestingly, the models with the best tradeoff between the two metrics are almost as semantically meaningful as those models with low temporal consistency and the highest semantic meaningfulness. This analysis suggests that regularization encourages the model to only represent different individuals differently if this is strictly necessary to factorize the data.

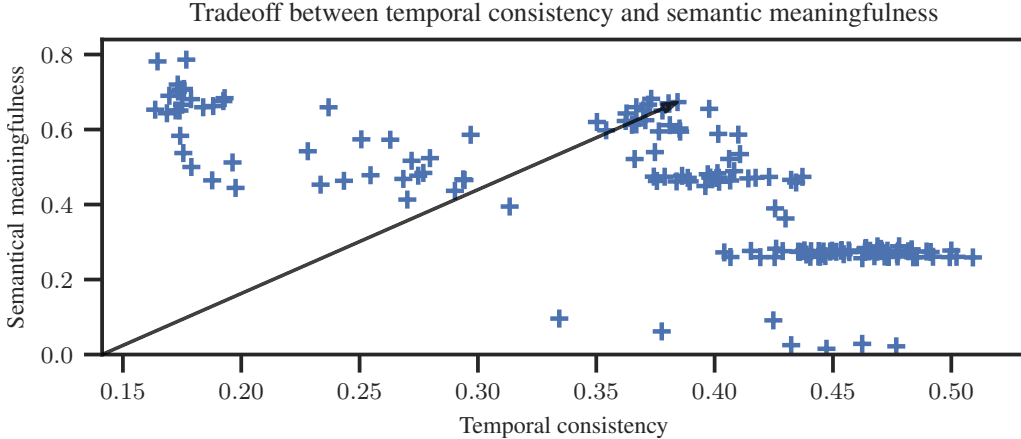


Figure 8: A grid search over the hyperparameters reveals that models can be temporally consistent, semantically meaningful, or both. Semantic meaningfulness is the sum of the *Rhythmicity* and *Mortality* metrics introduced in section 2.5. The arrow points to the model with the best tradeoff.

A.4 ABLATION STUDY

As highlighted in section 3, the regularization terms R_f and R_{adv} improve the semantic meaningfulness and temporal consistency of the model. Here we present an ablation study that shows the effects of the regularization terms on the sparseness and interpretability of the learned factors and embeddings, and how the adversarial term influences the distribution of the learned embeddings. While the regularization terms increase the methodological complexity of the model, we argue that they improve the interpretability of the results.

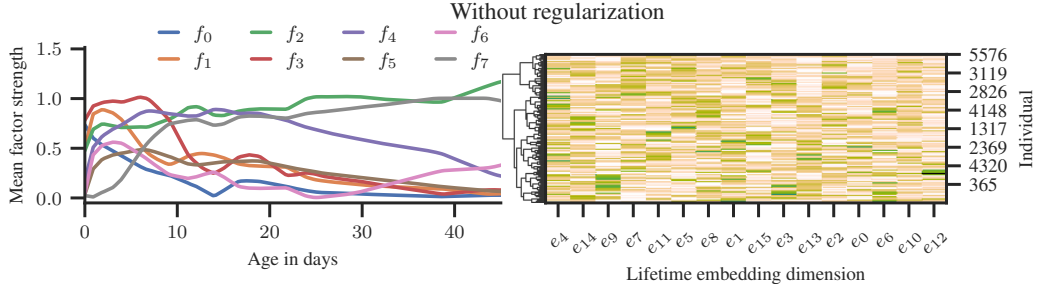


Figure 9: A model trained without the regularization terms $R_{\text{embeddings}}$, R_f , R_{basis} , and R_{adv} . The model uses all eight factors, and many of them are strongly correlated (left). Most individuals personality offsets are a function of multiple embeddings (right).

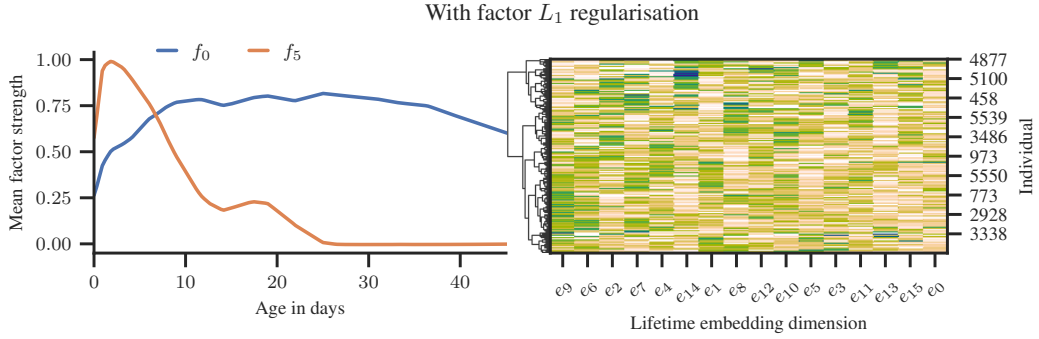


Figure 10: A model trained with only the regularization term R_f . The factor trajectories are now sparse and uncorrelated (left), most individuals personality offsets are still a function of multiple embeddings (right).

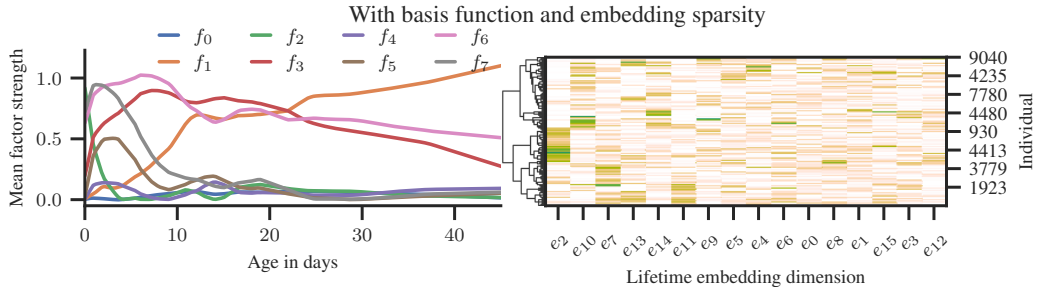


Figure 11: The regularization terms $R_{\text{embeddings}}$ and R_{basis} introduce sparseness in the embeddings (right), and also slightly decorrelate the factor trajectories (left).

A.5 BASELINE MODELS

We implemented SymNMF, and models proposed in Jiao et al. (2017) and Yu et al. (2017) in PyTorch and compare them to our method. Following the notation given in Yu et al. (2017), we list the degree of the fitted polynomial as d , and the regularization parameter as β . For the model proposed by Jiao et al. (2017), we list the regularization as γ . For SymNMF, the *Aligned* variant refers to the *Aligned symmetric NMF* described in section 2.4.

We only evaluate the non-negative and symmetric variants of the models for consistency with our method. For all baselines, we only list the hyperparameters with the best results that we were able to obtain.

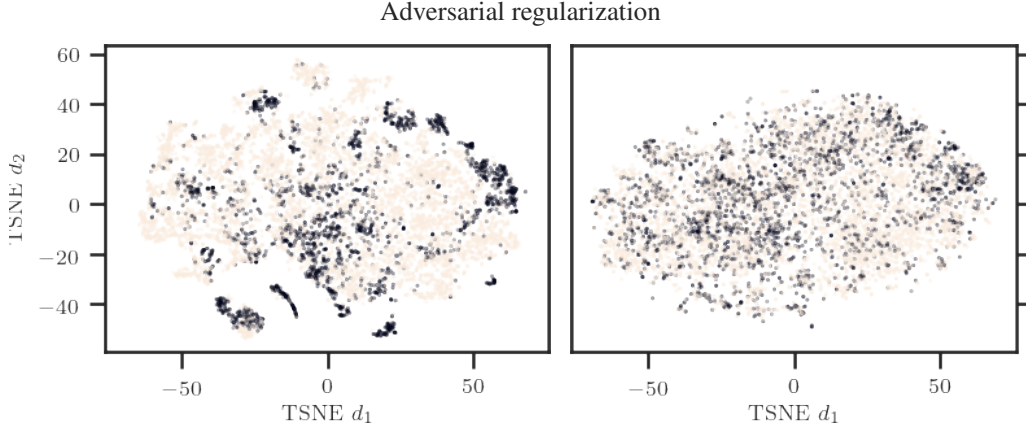


Figure 12: Scatter plots of the individuality embeddings ϕ (reduced to two dimensions using TSNE) for two models trained without (left) and with adversarial (right) regularization. The color encodes the dataset of the individuals (Dark = BN16, Bright = BN19). The model with adversarial regularization learns to embed the individuals from two different colonies that never interacted with each other in a joint individuality embedding space.

A.5.1 TEMPORAL REORDERING OF FACTORS FOR BASELINE MODELS

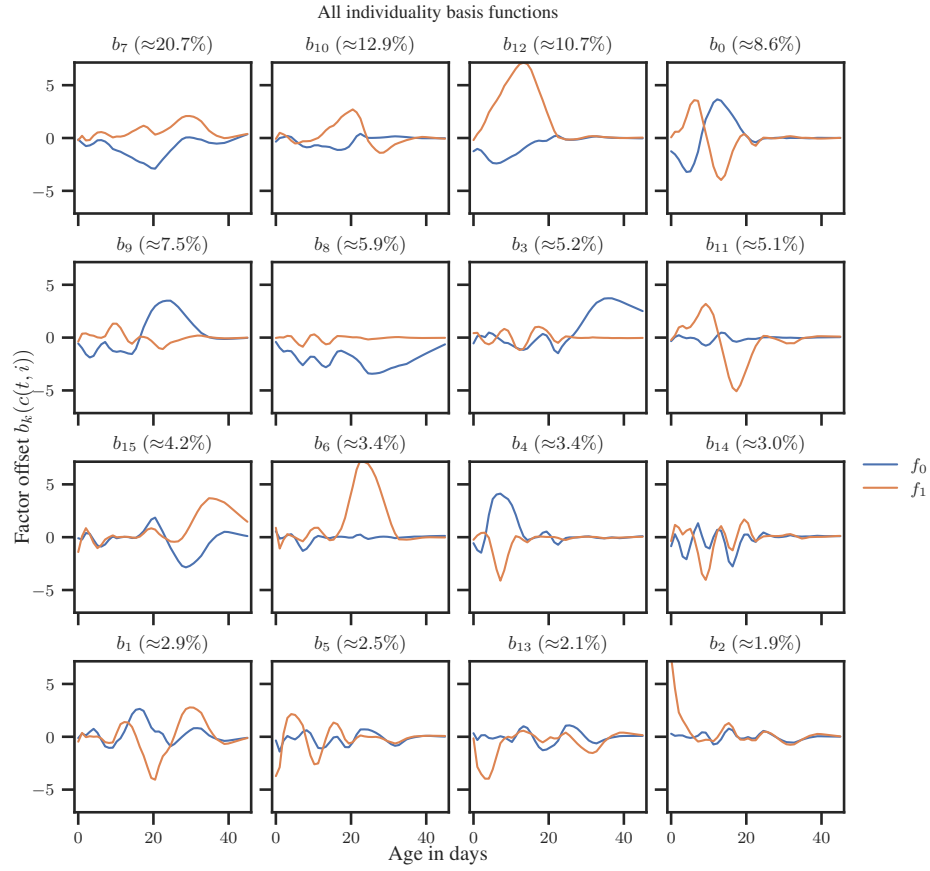
Algorithm 2: Temporal reordering of factors for baseline models

```

for  $d = 0$  to  $N_{days}$  do
   $F_{result} \leftarrow -$ 
  if  $d == 0$  then
     $F_{previous\_reordered} \leftarrow$  Precomputed SymNMF factors of day 0
  else
     $F_{current} \leftarrow$  Precomputed SymNMF factors of day  $d$ 
     $min\_loss \leftarrow 0$ 
     $F_{best} \leftarrow F_{current}$ 
    for all permutations  $p$  of orderings of factors  $[0..M]$  do
       $F_{current\_reordered} \leftarrow F_{current}$  reordered by permutation  $p$ 
      if  $(F_{current\_reordered} - F_{previous\_reordered})^2 < min\_loss$  then
         $min\_loss \leftarrow (F_{current\_reordered} - F_{previous\_reordered})^2$ 
         $F_{best} \leftarrow F_{current\_reordered}$ 
      end if
    end for
     $F_{previous\_reordered} \leftarrow F_{best}$ 
  end if
   $F_{result}.insert(F_{previous\_reordered})$ 
end for
return  $F_{result}$ 

```

A.6 BASIS FUNCTIONS OF FITTED MODEL

Figure 13: Magnitude of factor offsets for all learned individuality basis functions over age $b_k(c(t, i))$.

A.7 ADDITIONAL INDIVIDUAL LIFETIME TRAJECTORIES OF FITTED MODEL

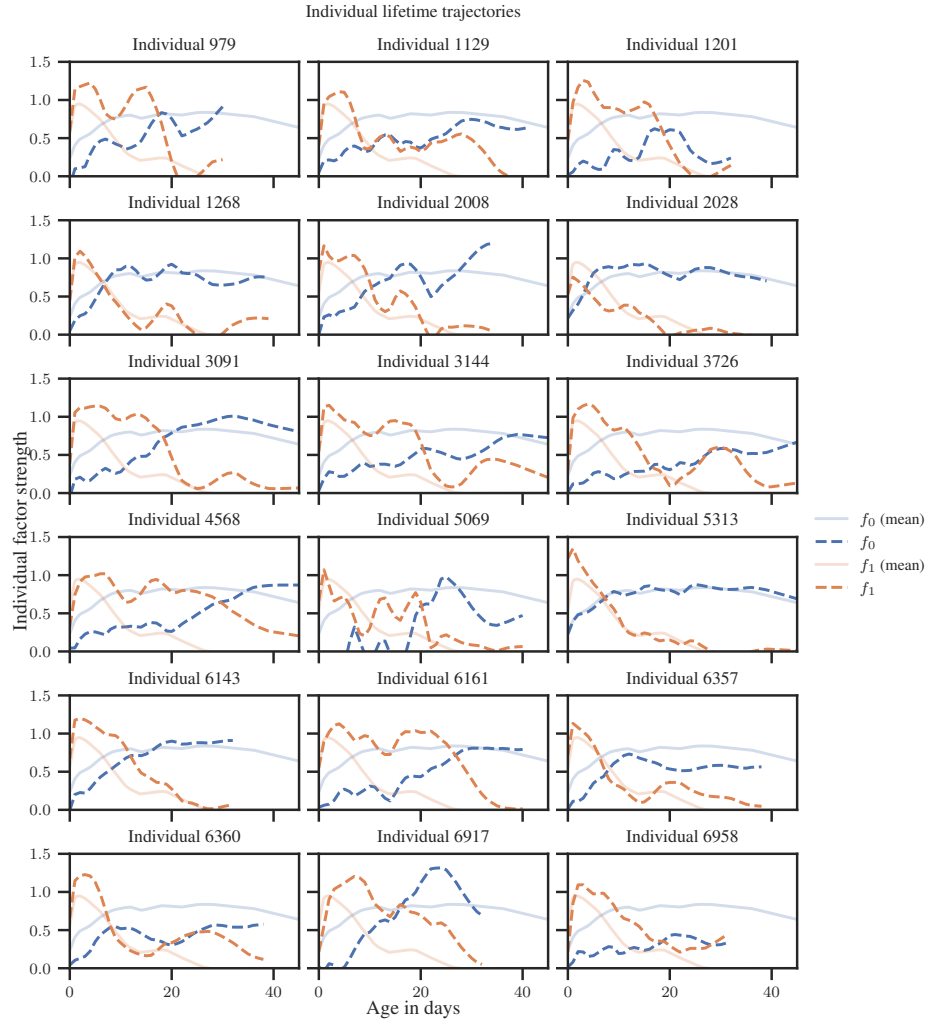


Figure 14: Individual lifetime trajectories: 18 individuals that lived for at least 35 days were randomly sampled and their factors $f(t, i)$ were computed over all t , constituting their lifetime trajectories.