Credibility in network regression with σ-hot encoding and weight balancing

Anne van der Scheer

Perunum
ACTUARIEEL ADVIES

Outline

- Credibility in networks
- From 1-hot to σ-hot encoding
- Weight balancing and initialization
- Simulation Friedman function
- Simulation disability insurance
- Implementation
- Conclusions
- References

Credibility in networks

Introduction

Credibility theory fills in the gap between using or not using some categories as covariates in a regression model. GLM needed the extension to GLMM for handling credibility of sparse[1] categories, by adding random effects to the model and thereby preventing overfitting.

In the literature of neural network regression we find the same distinction between fixed and randomized categoricals. For randomized categoricals the network comes with specific extensions under restrictive assumptions. For an overview of developments see Avanzi (2023) and Richman (2023).

[1] this often comes with high-cardinality, but sparse categories can also occur within a small number of categories

Credibility in networks

Alternative for randomization
Neural networks can be seen as a general structure without the specific form of classical models. The learning process is still hardly understood, but it is well known that standardization of continuous variables plays an important role in the learning capacity of the network.

If we extend this equal level playing field of continuous variables to categorical variables, then they can also fairly compete for the available complexity (by influencing weights) and credibility will be naturally achieved. This idea fits a Machine Learning setting.

Therefore we reconsider the encoding and handling of categorical variables.

From 1-hot to σ-hot encoding

The standard transformation procedure for getting from an unordered categorical variable with k levels to numbers is 1-hot encoding:

> *for every* $v \in \{v_1, v_2, \ldots, v_k\}$ *the corresponding 1-hot encoded values* $w_1, w_2, \ldots, w_k \in \{0, 1\}$ *are defined by* $w_i = \mathbb{1}_{(v=v_i)}$.

- Following the electrical metaphor, we will call 0 the cold-value and 1 the hot-value
- We will here assume that we do not use a reference variable and therefore use k instead of $k-1$ variables

From 1-hot to σ-hot encoding

Issues with 1-hot encoding:

- sparse categories have their own "full" weights which can lead to overfitting
- in regularization (like L1- and L2-penalties) all weights are of equal importance

Therefore we conclude:

- hot-values and their weights seem to be interchangeable, but they are not
- hot-values should not be arbitarily chosen (i.e. always equal 1)

From 1-hot to σ-hot encoding

We repeat standardizing continuous variables before they entry the network:

$$v \to \frac{v - \mu}{\sigma}$$

where μ is the mean and σ is the standard deviation[1] of the observations for each variable. Standardized variables become centered around zero with a standard deviation of 1.

We try to adopt these properties for categorical variables, so that they have optimal interaction with the continuous variables during training.

[1] population or sample standard deviation

From 1-hot to σ-hot encoding

In an attempt to equalize categorical and continuous variables, we start naively by standardizing the 1-hot encoding in the same way as a continuous variable:

$$0 \to \frac{-p}{\sqrt{p(1-p)}}$$

$$1 \to \frac{1-p}{\sqrt{p(1-p)}}$$

where p is again the fraction of hot-values in the observations.

Remark: the resulting values are independent of the specific choice of 0 and 1, as long as cold-value $<$ hot-value.

From 1-hot to σ-hot encoding

We notice the following:

- $p \approx 0$: standardized hot-value goes to infinity, standardized cold-value goes to 0.
- $p \approx 1$: standardized hot-value goes to 0, standardized cold-value goes to -infinity.

Recognizing that each hot-encoded variable is only part of a complete category, we multiply the standardized outcome by the relative frequency of the cold- en hot-values ($1-p$ resp. p) to get

$$\text{cold-value} \rightarrow (1-p) \cdot \frac{-p}{\sqrt{p(1-p)}} = -\sqrt{p(1-p)} = -\sigma$$

$$\text{hot-value} \rightarrow p \cdot \frac{1-p}{\sqrt{p(1-p)}} = \sqrt{p(1-p)} = \sigma$$

From 1-hot to σ-hot encoding

Instead of the 1-hot encoding this leads to a signed σ-coding, where σ is the standard deviation of the observed values, independent of the original cold- and hot-values.

Remarks:

- for sparse ($p \approx 0$) and dominant ($p \approx 1$) categories the encoding gets close to zero
- σ gets a maximal value of 0.5 when $p = 0.5$
- for categorical variables with only two categories, both categories get the same encoding
- the last step in the encoding destroyed the balance of the hot-encoded variables. This needs to be repaired

From 1-hot to σ-hot encoding

To make the following easier, we set the cold-value to zero (instead of -σ) and leave the hot-value unchanged:

$$\text{cold-value} \rightarrow 0$$

$$\text{hot-value} \rightarrow \sigma$$

We will call this end result the σ-hot encoding.

Just like the continuous variables we want a balanced input around zero for the whole categorical variable. To be precise, we want to ensure that the sum of the weighted inputs equals zero for every categorical variable and for every receiving neuron. At initialization and during the whole training.

Weight balancing and initialization

Balancing the sum of the weighted inputs for a categorical variable starts with a proper initialization. The standard in literature is a random initialization of all weights. Biases are often initialized as zero. We will do it the other way around:

- all weights are initialized as zero
- all biases in a layer are deterministically initialized around zero, thereby avoiding symmetry

With zero weights we start with a balanced sum of each categorical variable at each entry of neurons in the first hidden layer. After each update of the weights we want to restore existing inbalances.

Weight balancing and initialization

Let $W_{ji}^{(1)}$ for $i = 1, \ldots, k$ be the weights of the σ-hot encoded variables of some categorical variable, let n_i be the corresponding numbers of hot-encodings σ_i and let N be the total number of observations. Then we get

$$\text{inbalance}_j = \sum_{i=1}^{k} W_{ji}^{(1)} \cdot \sigma_i \cdot n_i$$

To restore balance again, we update the bias b_j and the weights of the receiving neuron j:

$$b_j^* = b_j + \frac{\text{inbalance}_j}{N}$$

$$W_{ji}^{(1)^*} = W_{ji}^{(1)} - \frac{\text{inbalance}_j}{\sigma_i \cdot N}$$

Weight balancing and initialization

Remarks:

- the updated biases and weights do not change the response of the network for all observations
- for very small categories we will get a weighted input that becomes almost zero and because of the weight balancing the network will produce an outcome close to average. Just what can be expected from a credibility approach
- σ-hot encoding does not change by scaling the observations, just like the continuous variables. The number of observations can however have effect on hyperparameters (e.g. weight regularization)
- σ-hot encoding is not the one-and-only proper encoding. For example the squared version of variance or 2 times σ could also be used. The network should be capable of transitioning different functions to the "right one"

Simulation Friedman function

We follow the simulation study for GLMMNet by Avanzi (2023)[1]:

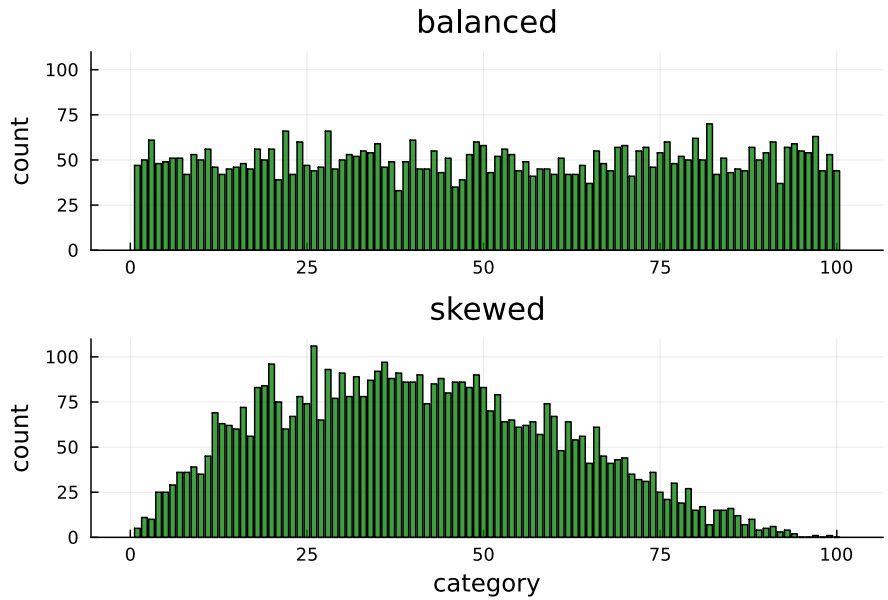
- for each of 6 experiments we set triple $(\mu_f, \sigma_u, \sigma_\epsilon)$
  - μ_f; fixed effects mean
  - σ_u; random effects standard deviation
  - σ_ϵ; noise
- fixed effects Friedman function for $x = (x_1, \cdots x_{10})$:
  $f(x) = 10 \sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5$
  scaled to mean value μ_f across all observations
- additional random effects $u_i \overset{iid}{\sim} \mathcal{N}(0, \sigma_u^2)$ of 100 categories
- response distributions:
  - Gaussian with identity-link, or
  - Gamma with log-link
  - corresponding dispersion parameter $\phi = \sigma_\epsilon^2$

[1] statistics from that study are taken from GitHub and are plotted to compare results

Simulation Friedman function

- distribution of 5,000 training and 2,500 testing observations:
  - $x_1, \cdots, x_{10} \overset{iid}{\sim} \mathcal{U}(0,1)$
  - category balanced ($\sim \mathcal{U}(0,1)$) or skewed ($\sim$ Beta$(2,3)$)

number of training observations by category

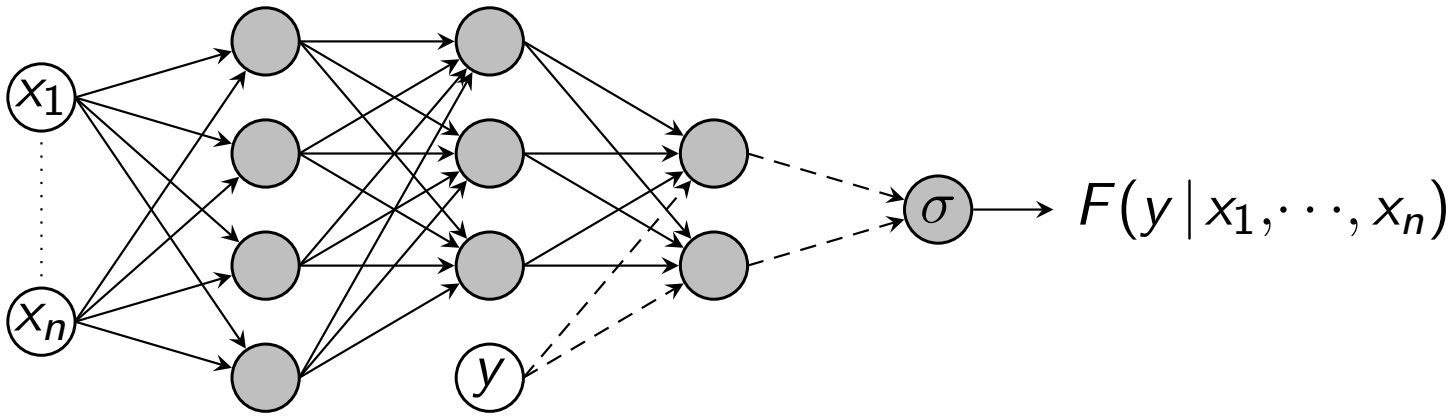


- each experiment will contain 50 simulation runs

Simulation Friedman function

In (Avanzi, 2023) the simulation is performed on many models besides GLMMNet. Among the best performing models were an entity embedded neural network (NN_ee) and GPBoost.

We adopt a general architecture of Chilinski (2020) to model the CDF of response variable y, using only 10 neurons:



The dashed arrows indicate non-negative weights to ensure a non-decreasing CDF for every input. The neuron in the last layer has sigmoid activation, other neurons have tanh activation.

Simulation Friedman function

Settings in simulation runs:

- optimization criterion: maximize log-likelihood
- Adam optimization (Kingma, 2014) with learning rate $\alpha = 0.01$ and 15,000 full iterations
- weights have absolute value ≤ 3, i.e. $\|W\|_\infty \leq 3$

Every simulation run is exercized on 3 models:

- model NN_nocats: ignoring categorical information
- model NN_1hot: 1-hot encoding without weight balancing
- model NN_σhot: σ-hot encoding with weight balancing

Simulation Friedman function

We use as evaluation criterion the average accuracy of point predictions (Root Mean Squared Error) per category

$$\text{RMSE\_avg} = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(\bar{y}_i^* - \bar{\hat{y}}_i^*)^2}$$

$\bar{y}_i^*$ and $\bar{\hat{y}}_i^*$ are the observed and estimated averages per category of the testing observations.

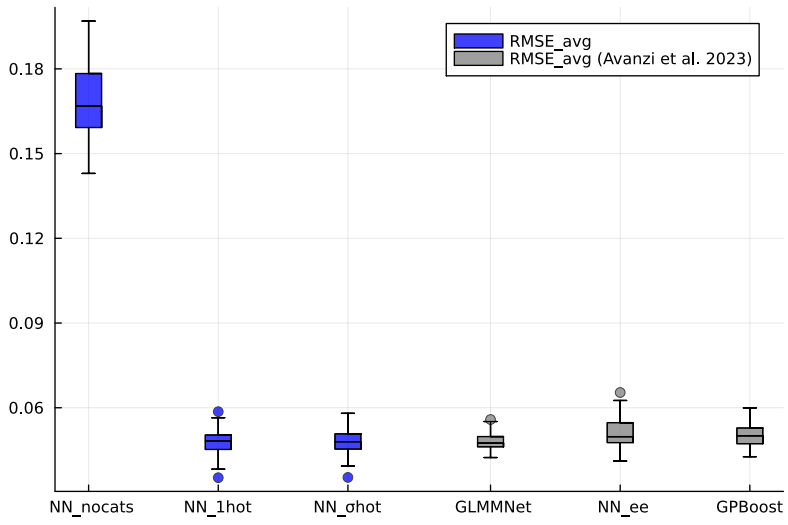
To get point predictions from the estimated CDF:

- calculate 100 quantiles for $q = 0.005, 0.015, 0.025, \cdots, 0.985, 0.995$
- each $F^{-1}(q)$ is outcome of binary search (F is monotone)
- calculate the mean of all quantiles

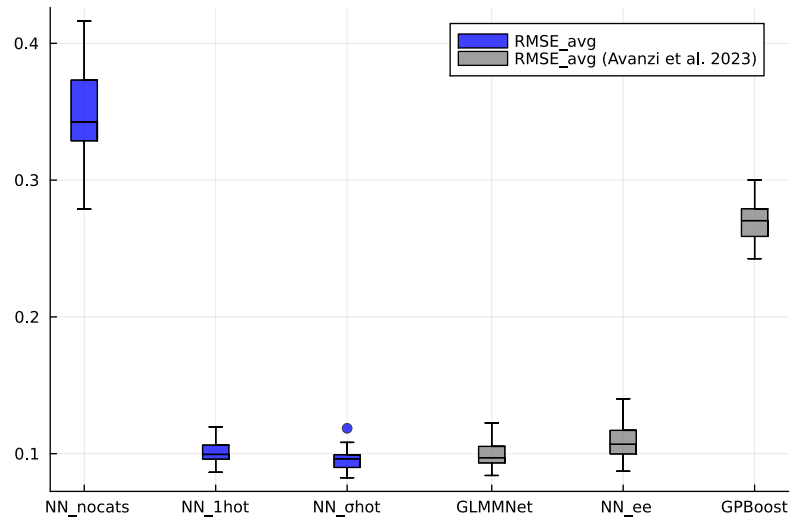
Simulation Friedman function

Experiment 1: high signal-to-noise with $(\mu_f, \sigma_u, \sigma_\epsilon) = \frac{1}{6} \cdot (4, 1, 1)$, Gaussian response and balanced categories.
Experiment 2: same as 1, but Gamma response.

statistics experiment 1



statistics experiment 2

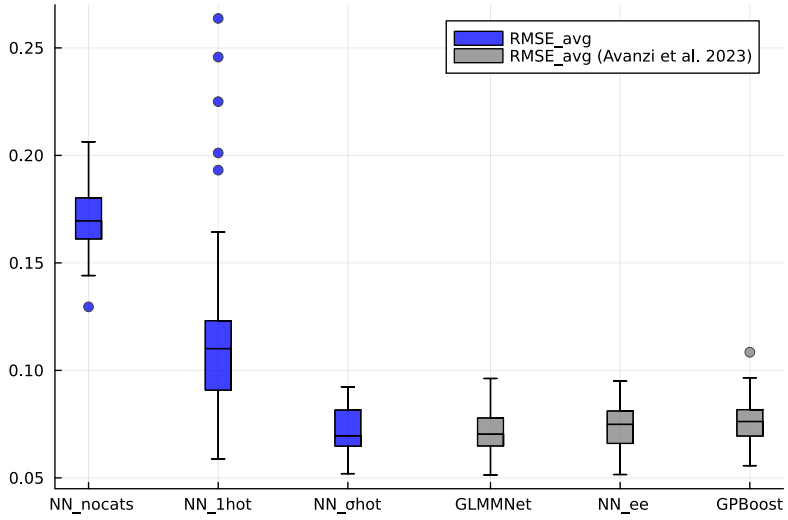


The results of 1-hot and σ-hot are comparable to other models. There are no sparse categories and therefore σ-hot has at most a small advantage above 1-hot.

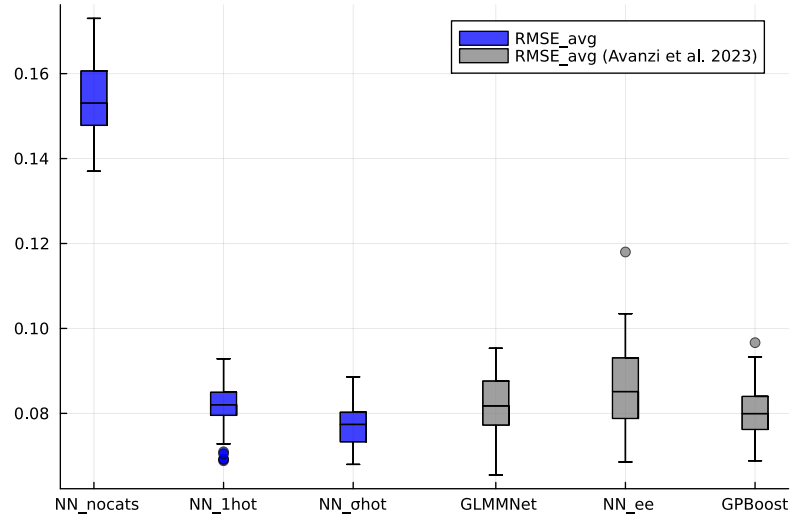
Simulation Friedman function

Experiment 3: same as 1, but skewed categories.
Experiment 4: same as 1, but medium signal-to-noise with $(\mu_f, \sigma_u, \sigma_\epsilon) = \frac{1}{7} \cdot (4, 1, 2)$.

statistics experiment 3



statistics experiment 4

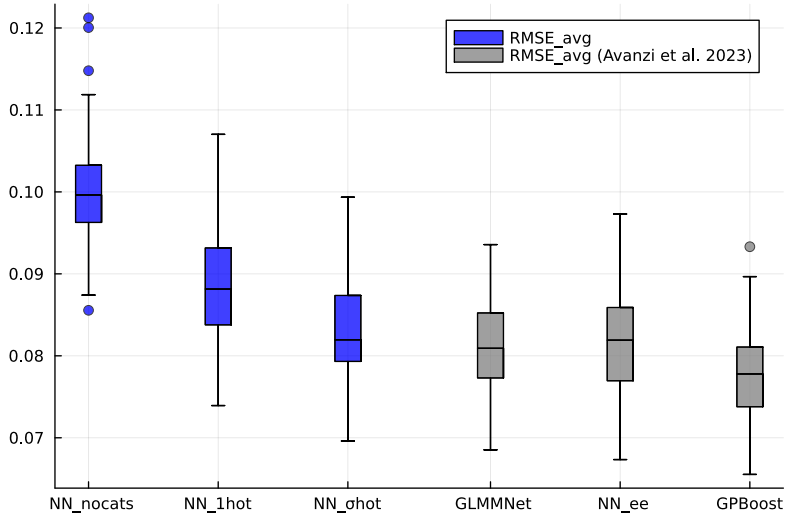


The performance of σ-hot is better than 1-hot in experiment 3 due to the sparse categories.

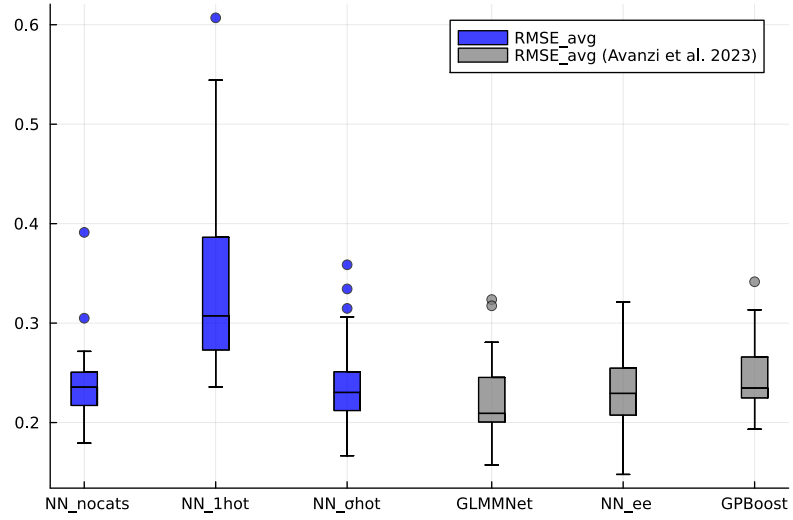
Simulation Friedman function

Experiment 5: same as 1, but high signal-to-noise with $(\mu_f, \sigma_u, \sigma_\epsilon) = \frac{1}{13} \cdot (8, 1, 4)$.
Experiment 6: same as 5, but Gamma response and skewed categories.

statistics experiment 5



statistics experiment 6



For the noisy and skew data of experiment 6 σ-hot has a clear advantage above 1-hot.

Simulation disability insurance

Our second simulation example is insurance of disability after some waiting period. We want to model the claim probability.

- categories can be thought of as industries, sectors and/or occupation classes
- this application does not meet the assumptions of GLMMNet and is therefore a good test case of the general setting of σ-hot encoding and weight balancing
- this example makes credibility visible in detailed graphs and we can therefore verify the meant properties of σ-hot encoding and weight balancing

Simulation disability insurance

The individual claim probability is given by

$$p(\text{claim}) = \alpha_2 \cdot e^{\alpha_1 x} + \beta_2 \cdot e^{\beta_1 (x-\beta_0)^2} + \gamma_2 \cdot e^{\gamma_1 (x-\gamma_0)^2}$$

where x is the (whole) age of the insured and the other parameters are random effects of 18 categories:

- $\alpha_1 \overset{iid}{\sim} \mathcal{U}(0.04, 0.05)$ and $\alpha_2 \overset{iid}{\sim} \mathcal{U}(0.0025, 0.004)$
- $\beta_0 \overset{iid}{\sim} \mathcal{U}(30, 40)$, $\beta_1 \overset{iid}{\sim} \mathcal{U}(-0.02, 0)$ and $\beta_2 \overset{iid}{\sim} \mathcal{U}(0, 0.0025)$
- $\gamma_0 \overset{iid}{\sim} \mathcal{U}(65, 70)$, $\gamma_1 \overset{iid}{\sim} \mathcal{U}(-0.02, 0)$ and $\gamma_2 \overset{iid}{\sim} \mathcal{U}(-0.01, 0)$

The second and third term add "local" age disturbances to the first term, thereby creating variations of a standard exponential curve.

Simulation disability insurance

The distibution of 500,000 training and 500,000 testing observations are:

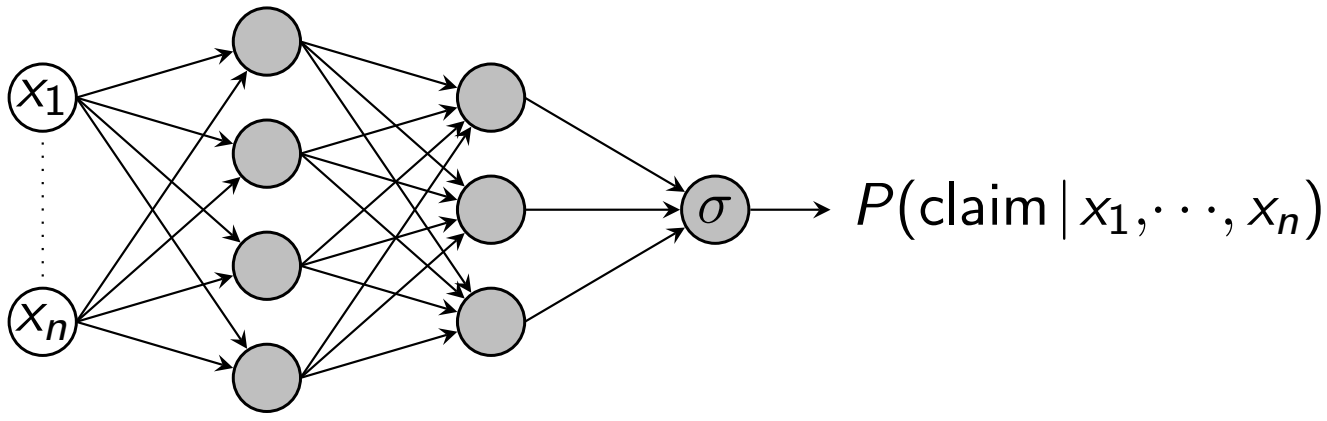
$$x \sim \text{round}(\text{age}_{\text{min}} + (\text{age}_{\text{max}} - \text{age}_{\text{min}}) * x_0)$$
$$x_0 \sim \text{Beta}(\max(2 + \delta, 2), \max(2 - \delta, 2))$$

- $\delta \overset{iid}{\sim} \mathcal{U}(-2, 2)$ is another random category-effect, shifting the population age density for each category
- age_{min}=18 and age_{max}=67 have fixed values
- category balanced ($\sim \mathcal{U}(0, 1)$) or skewed ($\sim \text{Beta}(2, 3)$)

We will perform two experiments. Experiment 1 has balanced category sizes, experiment 2 has skewed category sizes. Each experiment will contain 50 simulation runs.

Simulation disability insurance

We use a standard architecture to model the probability of a claim response with only 8 neurons:



The neuron in the last layer has sigmoid activation, other neurons have tanh activation. Changed settings in simulation runs:

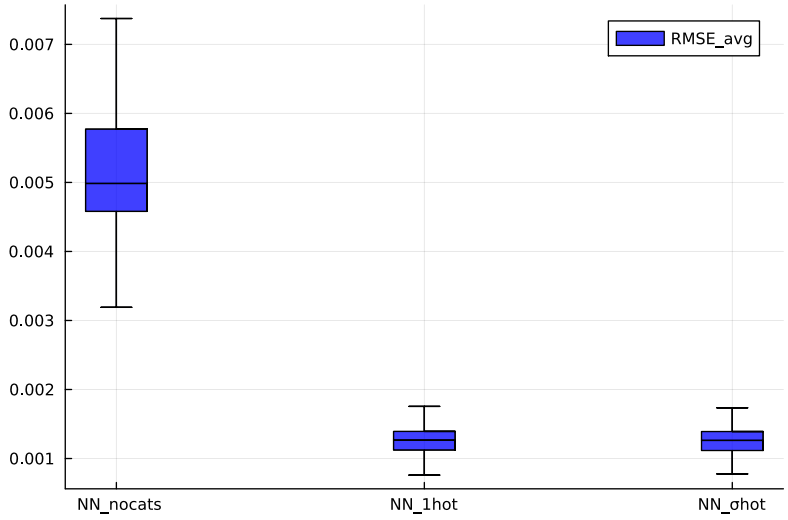
- Adam learning rate $\alpha = 0.001$
- weights have absolute value ≤ 1, i.e. $\|W\|_\infty \leq 1$

Every simulation run is again exercised on the same 3 models.

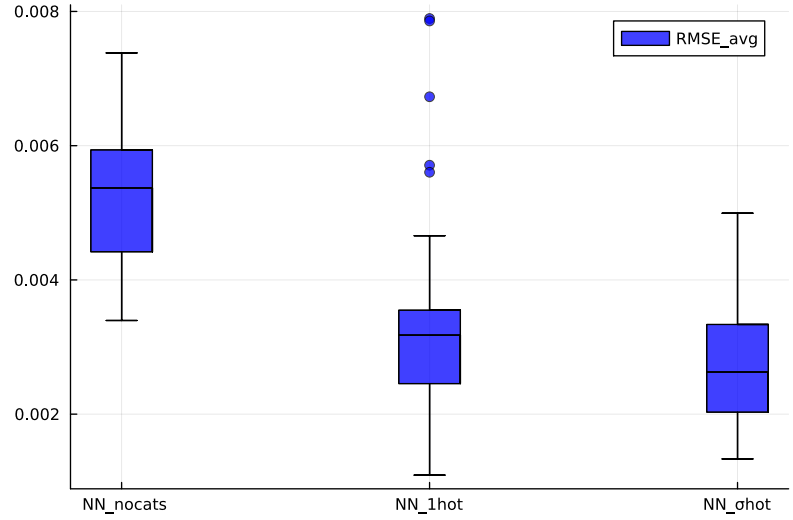
Simulation disability insurance

The outcomes of the 2 experiments are:

statistics experiment 1



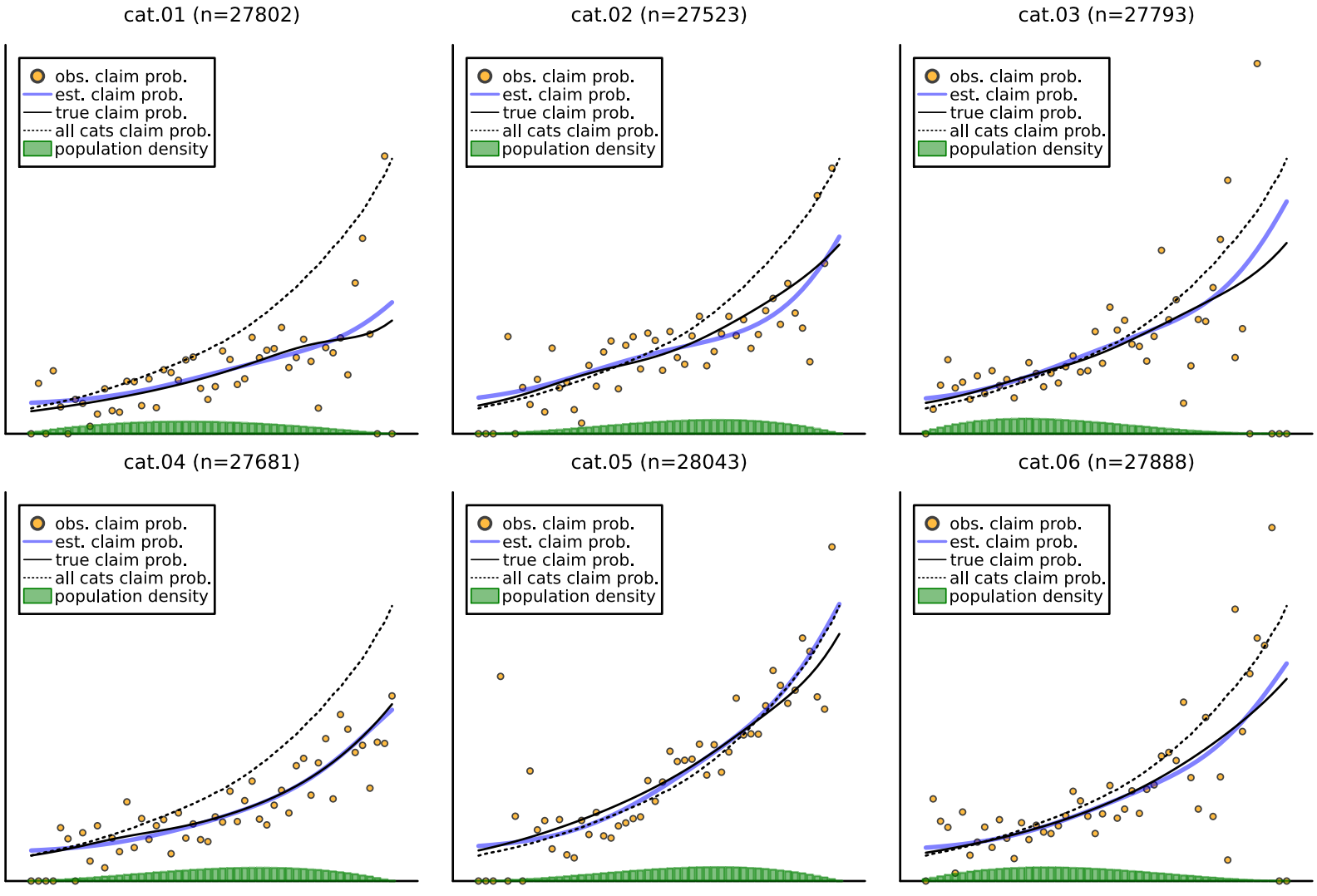
statistics experiment 2



The results of 1-hot and σ-hot are comparable in experiment 1, due to balanced category sizes. In experiment 2 there is only a slight advantage in using σ-hot. The next slides contain detailed graphs of the fit of the training observations.

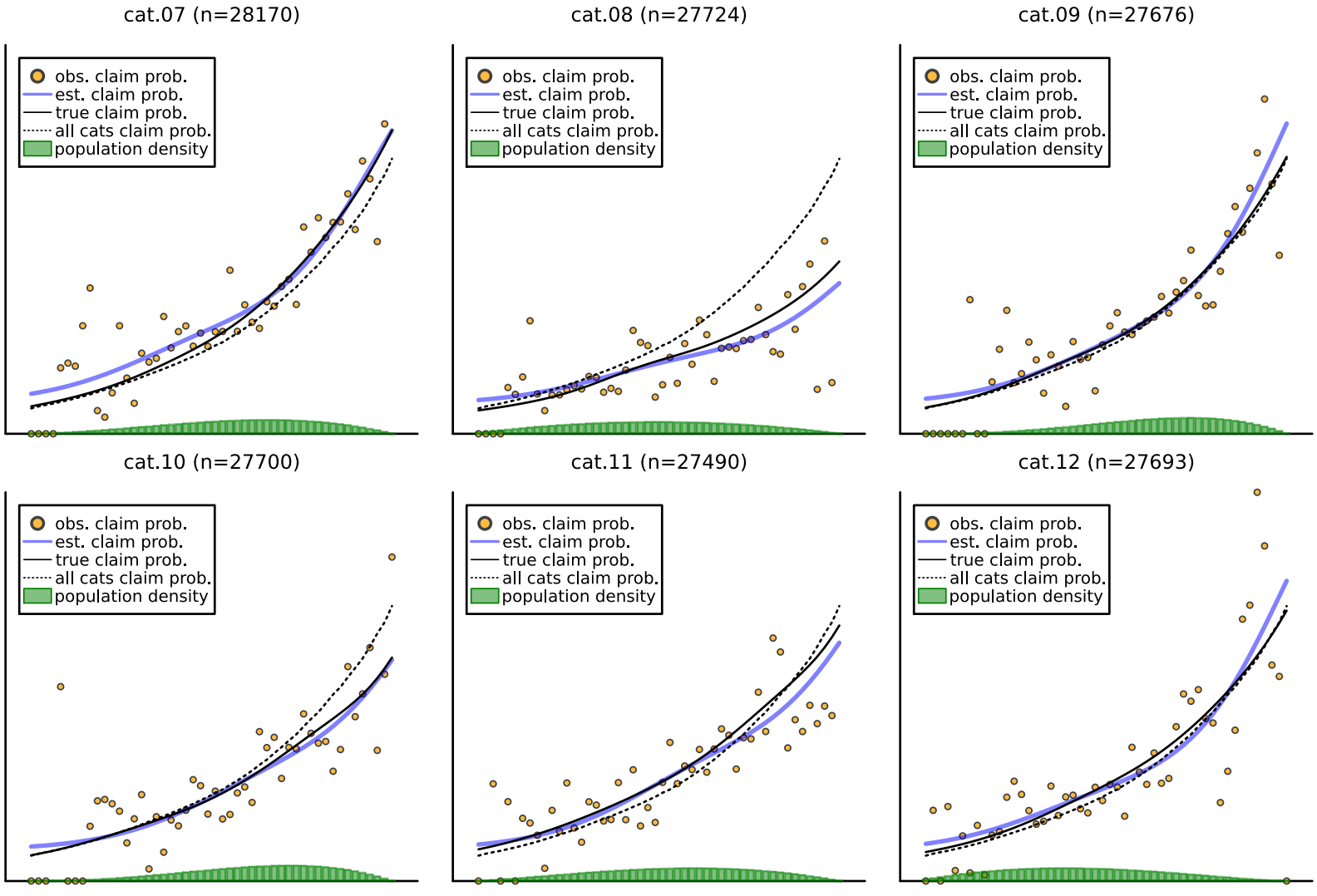
Simulation disability insurance

results model NN_σhot experiment 1, cats 01 t/m 06



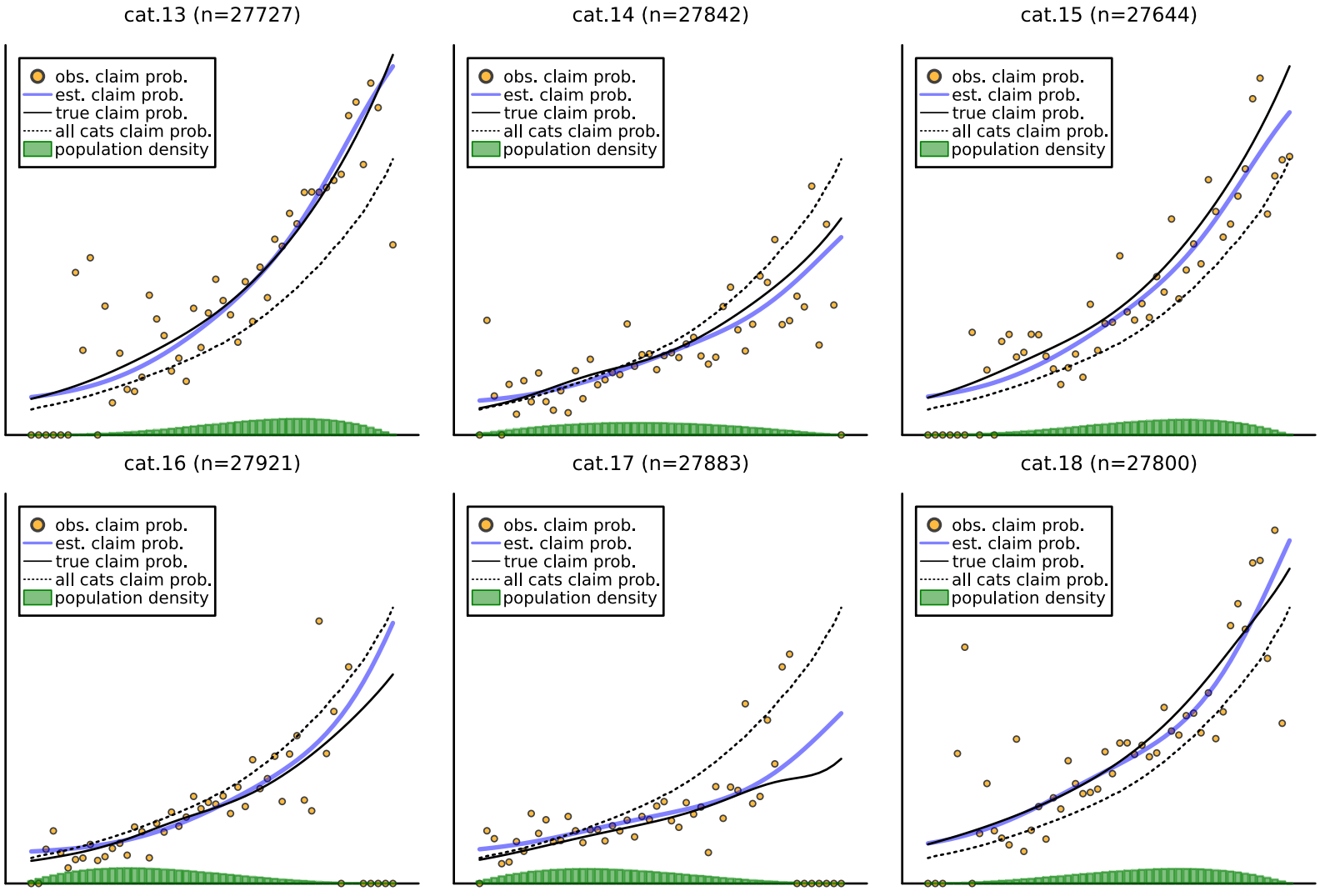
Simulation disability insurance

results model NN_σhot experiment 1, cats 07 t/m 12



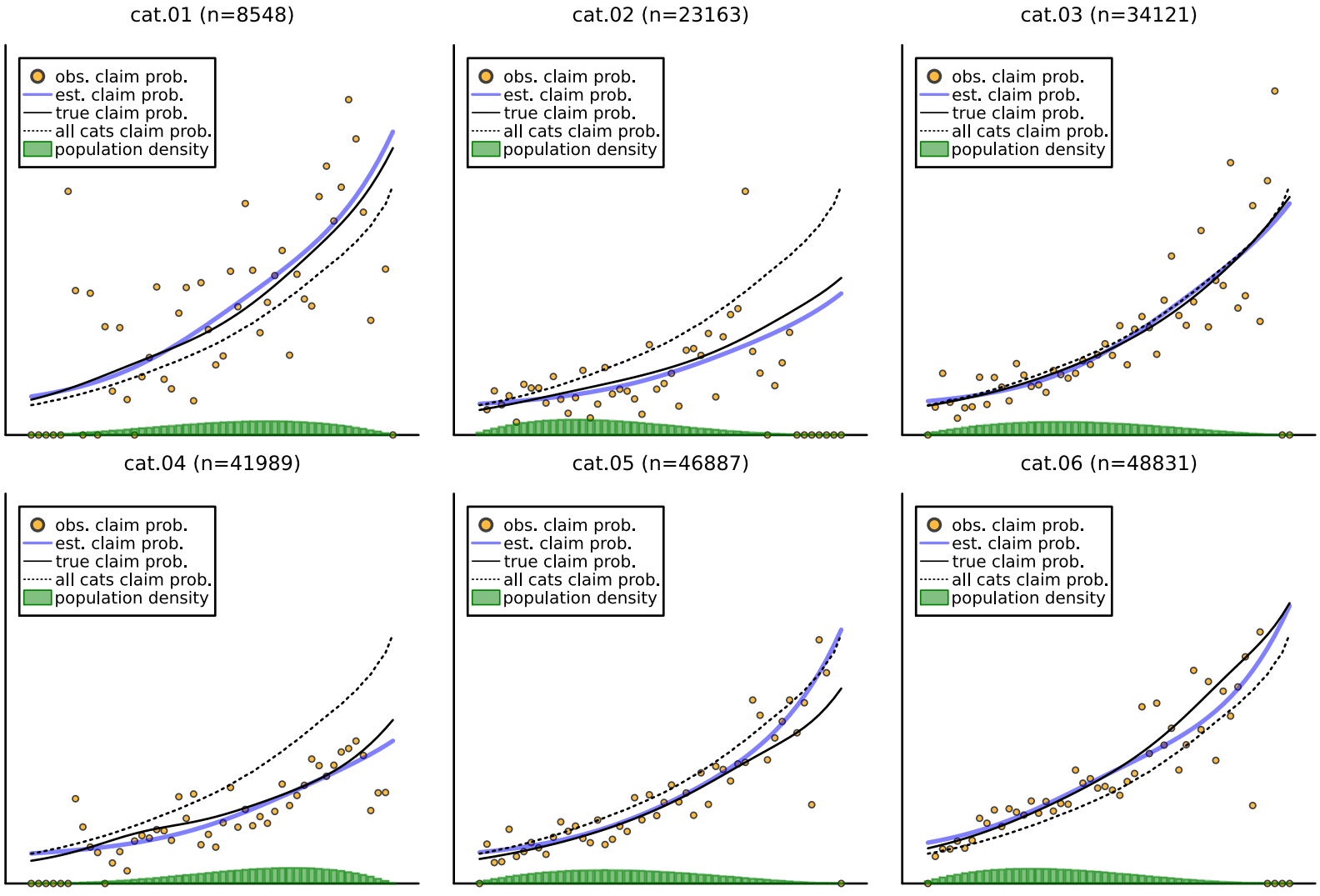
Simulation disability insurance

results model NN_σhot experiment 1, cats 13 t/m 18



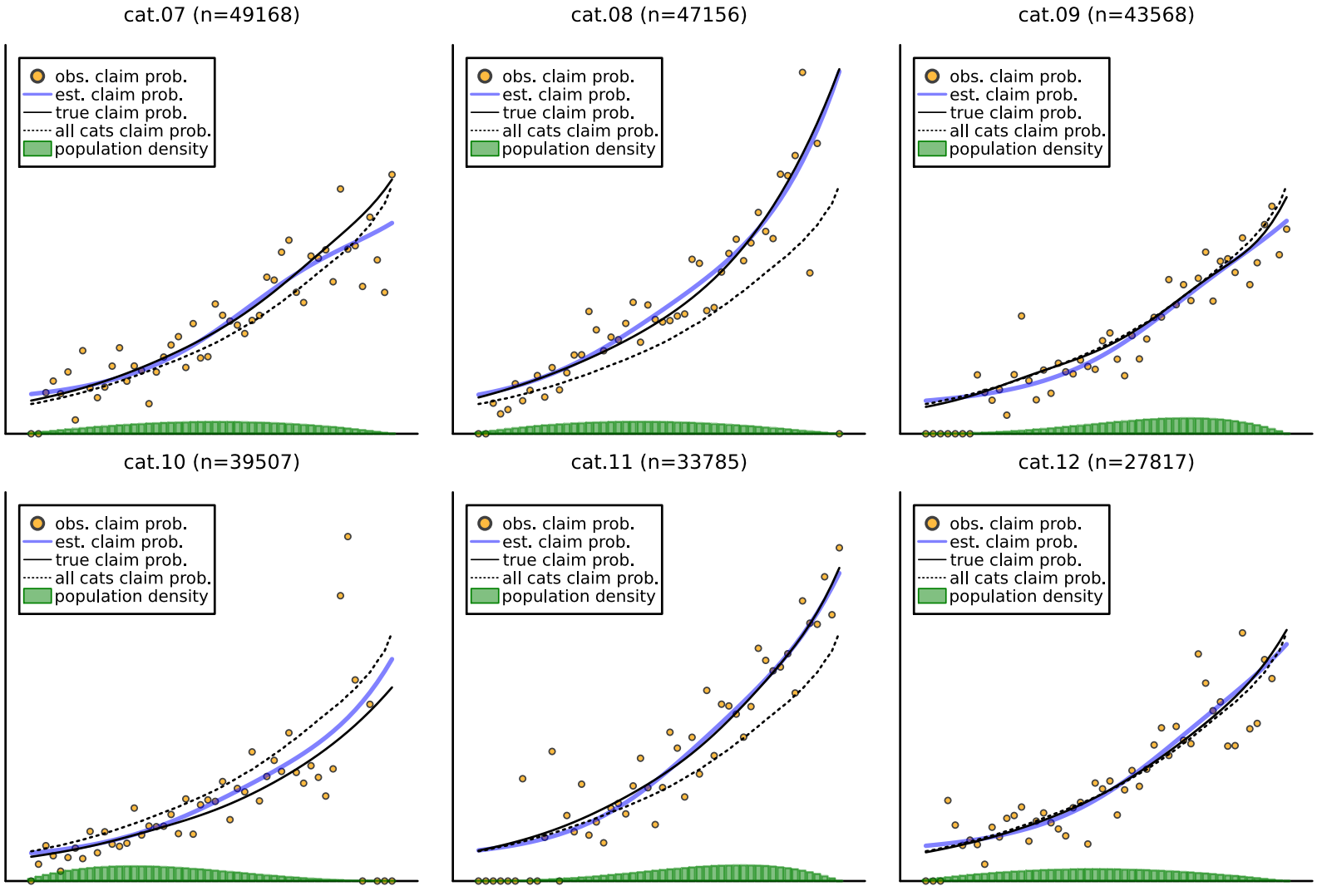
Simulation disability insurance

results model NN_σhot experiment 2, cats 01 t/m 06



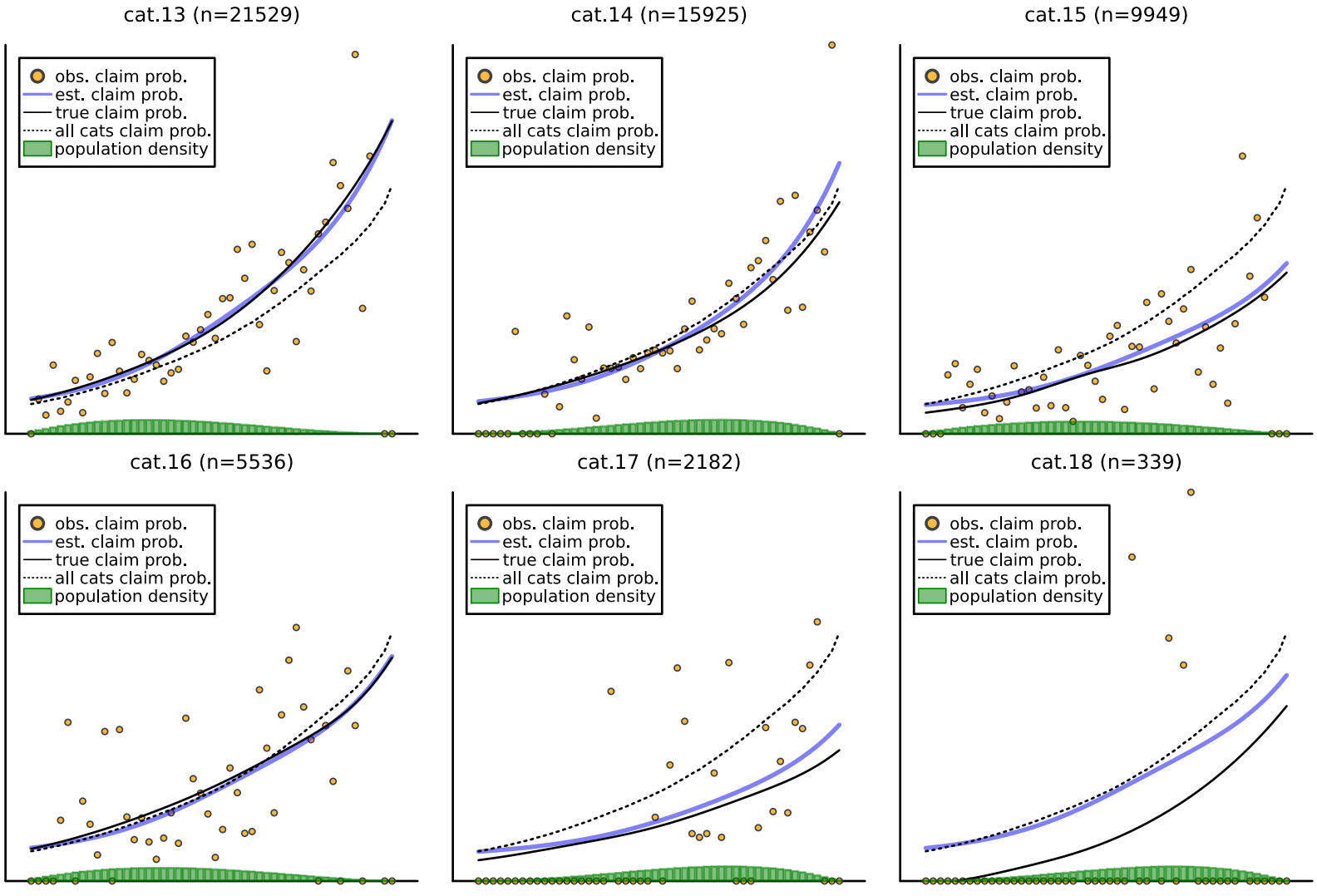
Simulation disability insurance

results model NN_σhot experiment 2, cats 07 t/m 12



Simulation disability insurance

results model NN_σhot experiment 2, cats 13 t/m 18



Implementation

- Julia is a relatively new language (2012) for scientific programming
- Julia is fast and therefore "solves two-language-problem"
- the neural network functions and simulations were all built in Julia code

https://github.com/perunum/sigma-hot-encoding

Conclusions

σ-hot encoding and weight balancing provide a general solution for credibility in network regression:

- they are applicable to all categorical variables
- they need no assumptions for distributions
- they need no extensions of network structure
- they work well in combination with max-norm on weights
- combined with density estimation networks they complete a general framework for actuaries
- they stimulate modelling with small and simple networks and therefore can stimulate applying networks in insurance

References

Avanzi, B., Taylor, G., Wang, M., & Wong, B. (2023). Machine Learning with High-Cardinality Categorical Features in Actuarial Applications. *arXiv preprint arXiv:2301.12710*.

Chilinski, P., & Silva, R. (2020, August). Neural likelihoods via cumulative distribution functions. In *Conference on Uncertainty in Artificial Intelligence* (pp. 420-429). PMLR.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Richman, R., & Wüthrich, M. V. (2023). High-Cardinality Categorical Covariates in Network Regressions. *Available at SSRN 4549049*.