# LocalGLMnet and more

Mario V. Wüthrich
RiskLab, ETH Zurich

"Deep Learning with Actuarial Applications in R"
Swiss Association of Actuaries SAA/SAV, Zurich
October 14/15, 2021

# Programme SAV Block Course

- Refresher: Generalized Linear Models (THU 9:00-10:30)
- Feed-Forward Neural Networks (THU 13:00-15:00)
- Discrimination-Free Insurance Pricing (THU 17:15-17:45)

- LocalGLMnet (FRI 9:00-10:30)
- Convolutional Neural Networks (FRI 13:00-14:30)
- Wrap Up (FRI 16:00-16:30)

# Contents: LocalGLMnet and more

- Balance property for neural networks
- Multiplicity of equally good FNN models
- The nagging predictor
- LocalGLMnet: interpretable deep learning

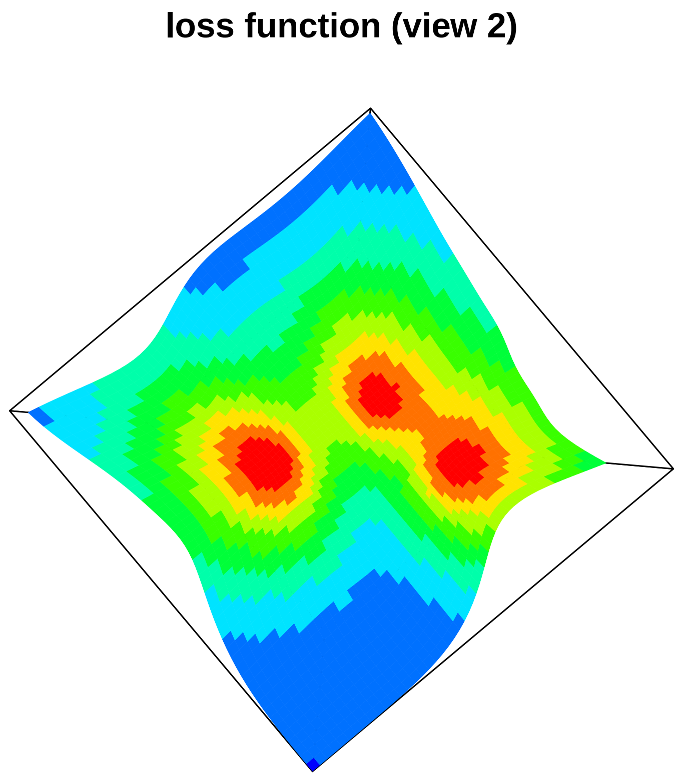
- **Balance Property for Neural Networks**

# Balance Property for FNN Models

| | epochs | run time | # param. | in-sample loss in $10^{-2}$ | out-of-sample loss in $10^{-2}$ | average frequency |
|---|---|---|---|---|---|---|
| homogeneous model | | – | 1 | 32.935 | 33.861 | 10.02% |
| Model GLM1 | | 20s | 49 | 31.267 | 32.171 | 10.02% |
| Deep FNN One-Hot | 250 | 152s | 1'306 | 30.268 | 31.673 | 10.19% |
| Deep FNN Emb($b=1$) | 700 | 419s | 719 | 30.245 | 31.506 | 9.90% |
| Deep FNN Emb($b=2$) | 600 | 365s | 792 | 30.165 | 31.453 | 9.70% |
| Deep FNN Emb($b=2$) seed 1 | | 365s | 792 | 30.411 | 31.503 | 9.90% |
| Deep FNN Emb($b=2$) seed 2 | | 365s | 792 | 30.352 | 31.418 | 10.23% |
| Deep FNN Emb($b=2$) seed 3 | | 365s | 792 | 30.315 | 31.500 | 9.61% |

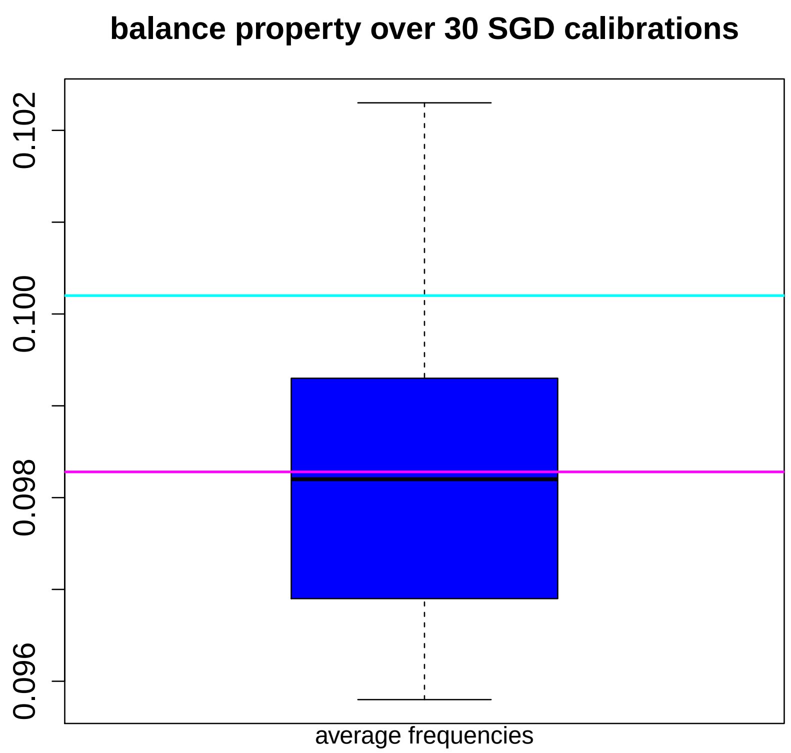
- Balance property fails to hold for FNN regression models.
- The reason is early stopping which prevents from being in a critical point of the deviance loss $D^*(\boldsymbol{Y}, \cdot)$ (under canonical link choice).

# Critical Points of Deviance Loss Function




Under the canonical link choice, the balance property is fulfilled in the critical points.

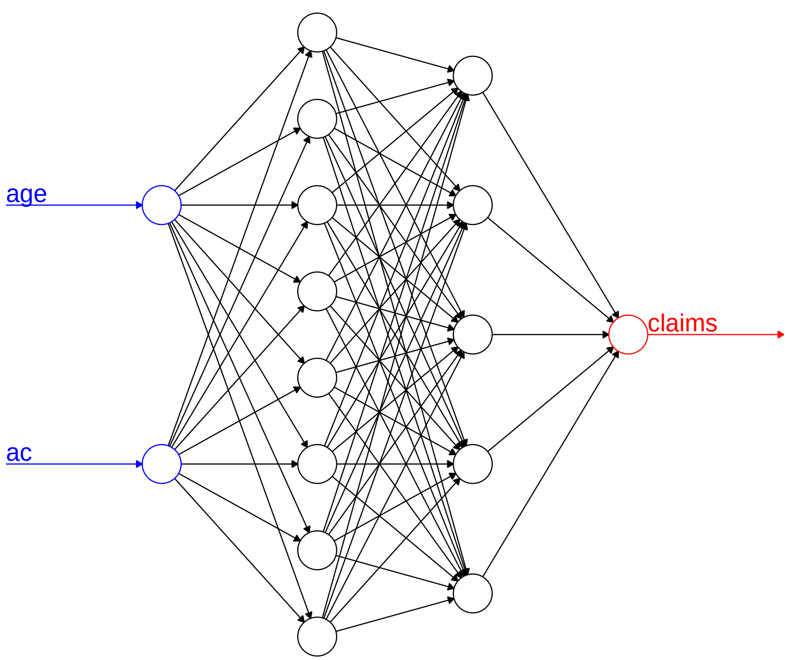
# Failure of Balance Property of FNNs

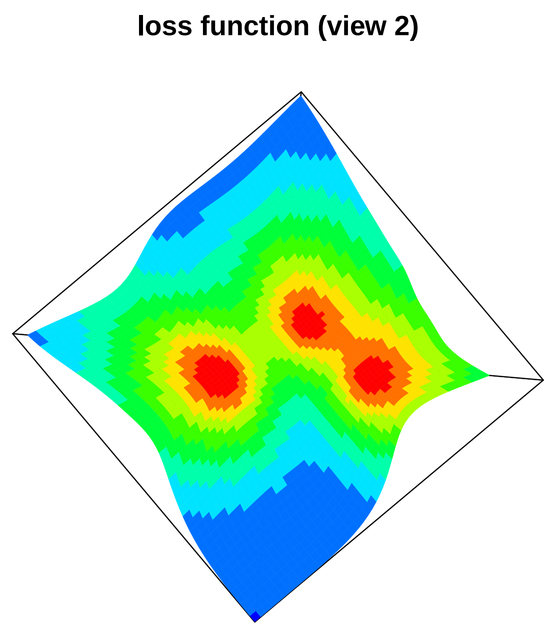



The failure of the balance property is significant.

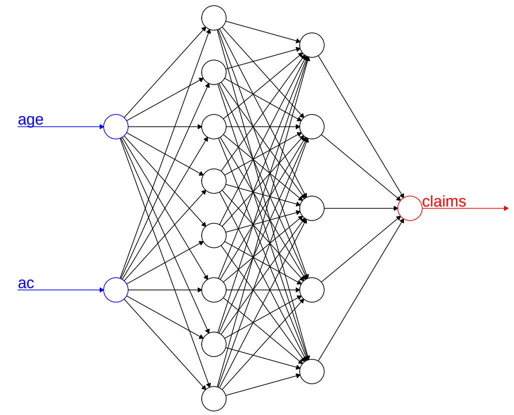
# Seeds and Randomness Involved in SGD

```
layer_dense(units=q1, activation = "tanh",
                      kernel_initializer = initializer_glorot_uniform(),
                      bias_initializer='zeros') %>%
layer_dropout(rate=0.05)

model %>% fit(X, Y, validation_split = 0.2, batch_size = 10000, epochs = 500)
```



# Representation Learning: Additional GLM Step




- Network mapping with link $g$

$$\boldsymbol{x}_i \;\mapsto\; g(\widehat{\mu}_i) = g(\widehat{\mathbb{E}}[Y_i]) = \left\langle \widehat{\boldsymbol{\beta}}, \widehat{\boldsymbol{z}}^{(d:1)}(\boldsymbol{x}_i) \right\rangle,$$

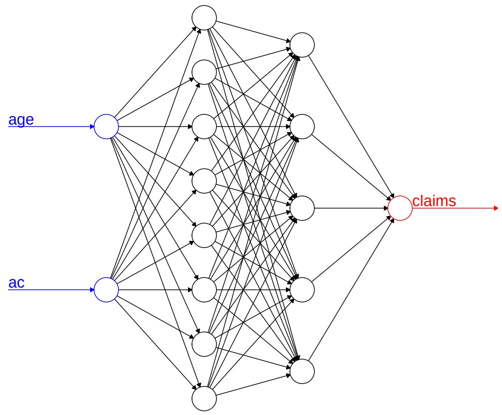
with GDM fitted network parameter $\widehat{\vartheta} = (\widehat{\boldsymbol{w}}, \widehat{\boldsymbol{\beta}}) \in \mathbb{R}^r$.

- Mapping $\boldsymbol{x}_i \mapsto \widehat{\boldsymbol{z}}_i = \widehat{\boldsymbol{z}}^{(d:1)}(\boldsymbol{x}_i)$ should be understood as learned representation.

- **Idea** (with canonical link choice $g = h$): consider a GLM with new covariates $\widehat{\boldsymbol{z}}_i$.

- If design matrix $\widehat{\mathfrak{Z}} \in \mathbb{R}^{n\times(q_d+1)}$ has full rank $q_d + 1 \leq n$, we have a unique MLE $\widehat{\boldsymbol{\beta}}^{\mathrm{MLE}}$, and the balance property will be fulfilled under canonical link choice.

# Rectifying the Balance Property




- Network mapping with link $g$

$$\boldsymbol{x}_i \;\mapsto\; g(\widehat{\mu}_i) = g(\widehat{\mathbb{E}}[Y_i]) = \left\langle \widehat{\boldsymbol{\beta}}, \widehat{\boldsymbol{z}}^{(d:1)}(\boldsymbol{x}_i) \right\rangle,$$

with GDM fitted network parameter $\widehat{\vartheta} = (\widehat{\boldsymbol{w}}, \widehat{\boldsymbol{\beta}}) \in \mathbb{R}^r$.

- Choose $\widehat{\boldsymbol{\beta}}^{\text{MLE}}$ for design matrix $\widehat{\mathfrak{Z}} \in \mathbb{R}^{n \times (q_d+1)}$

$$\boldsymbol{x}_i \;\mapsto\; h(\widehat{\mu}_i) = h(\widehat{\mathbb{E}}[Y_i]) = \left\langle \widehat{\boldsymbol{\beta}}^{\text{MLE}}, \widehat{\boldsymbol{z}}^{(d:1)}(\boldsymbol{x}_i) \right\rangle,$$

for canonical link $h$.

# R Code for Implementing the Balance Property

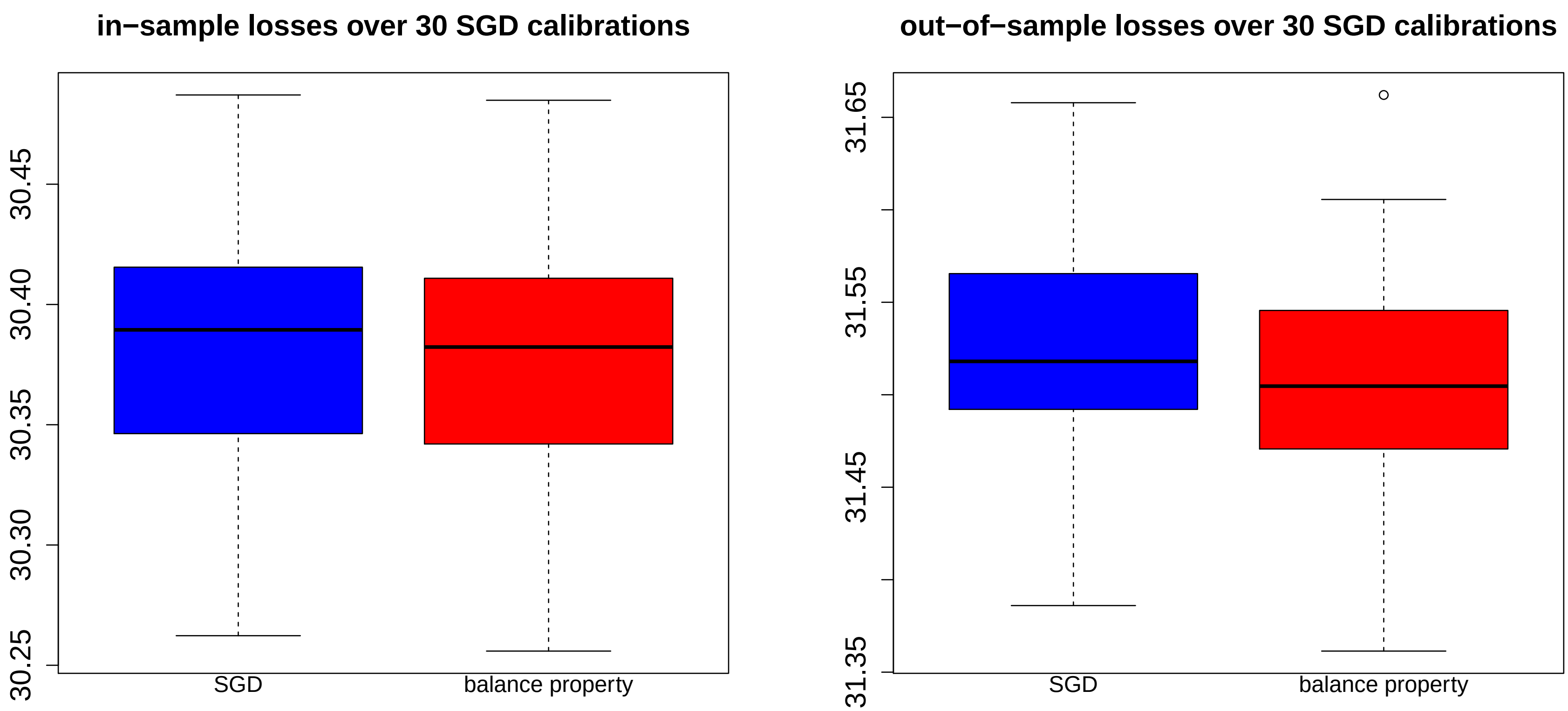
```
z.layer  <- keras_model(inputs=model$input,
                             outputs=get_layer(model,'hidden3')$output)

learn[,c("z1","z2","z3","z4","z5","z6","z7","z8","z9","z10")] <-
            data.frame(z.layer %>% predict(list(Design, VehBrand, Region, LogVol)))

glm(ClaimNb ~ z1 + z2 + z3 + z4 + z5 + z6 + z7 + z8 + z9 + z10,
            data=learn, offset=log(Exposure), family=poisson())
```

- The R code considers the Poisson model with canonical link $g = h = \log$.
- If we do not have canonical link we still need to adjust the intercept $\widehat{\beta}_0^{\mathrm{MLE}}$.
- The additional GLM step may lead to over-fitting, if the size $q_d$ of the last hidden FNN layer is not too large, there won't be over-fitting.

# Balance Property for FNN Models

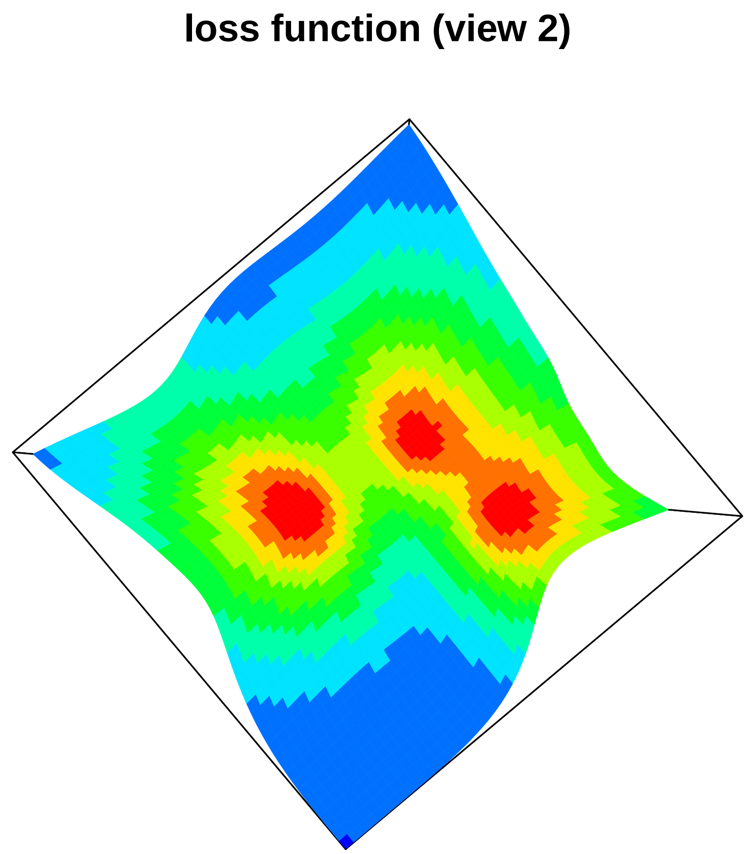
| | epochs | run time | # param. | in-sample loss in $10^{-2}$ | out-of-sample loss in $10^{-2}$ | average frequency |
|---|---|---|---|---|---|---|
| homogeneous model | | – | 1 | 32.935 | 33.861 | 10.02% |
| Model GLM1 | | 20s | 49 | 31.267 | 32.171 | 10.02% |
| Deep FNN One-Hot | 250 | 152s | 1'306 | 30.268 | 31.673 | 10.19% |
| Deep FNN Emb($b = 1$) | 700 | 419s | 719 | 30.245 | 31.506 | 9.90% |
| Deep FNN Emb($b = 2$) | 600 | 365s | 792 | 30.165 | 31.453 | 9.70% |
| Deep FNN Emb($b = 2$) seed 1 | | 365s | 792 | 30.411 | 31.503 | 9.90% |
| Deep FNN Emb($b = 2$) seed 2 | | 365s | 792 | 30.352 | 31.418 | 10.23% |
| Deep FNN Emb($b = 2$) seed 3 | | 365s | 792 | 30.315 | 31.500 | 9.61% |
| Reg. FNN Emb($b = 2$) seed 1 | | $+$7s | 792 | 30.408 | 31.488 | 10.02% |
| Reg. FNN Emb($b = 2$) seed 2 | | $+$7s | 792 | 30.346 | 31.418 | 10.02% |
| Reg. FNN Emb($b = 2$) seed 3 | | $+$7s | 792 | 30.303 | 31.462 | 10.02% |

# Balance Property for FNN Models



- **The "Best" FNN Regression Model**

# Multiplicity of Equally Good FNN Models

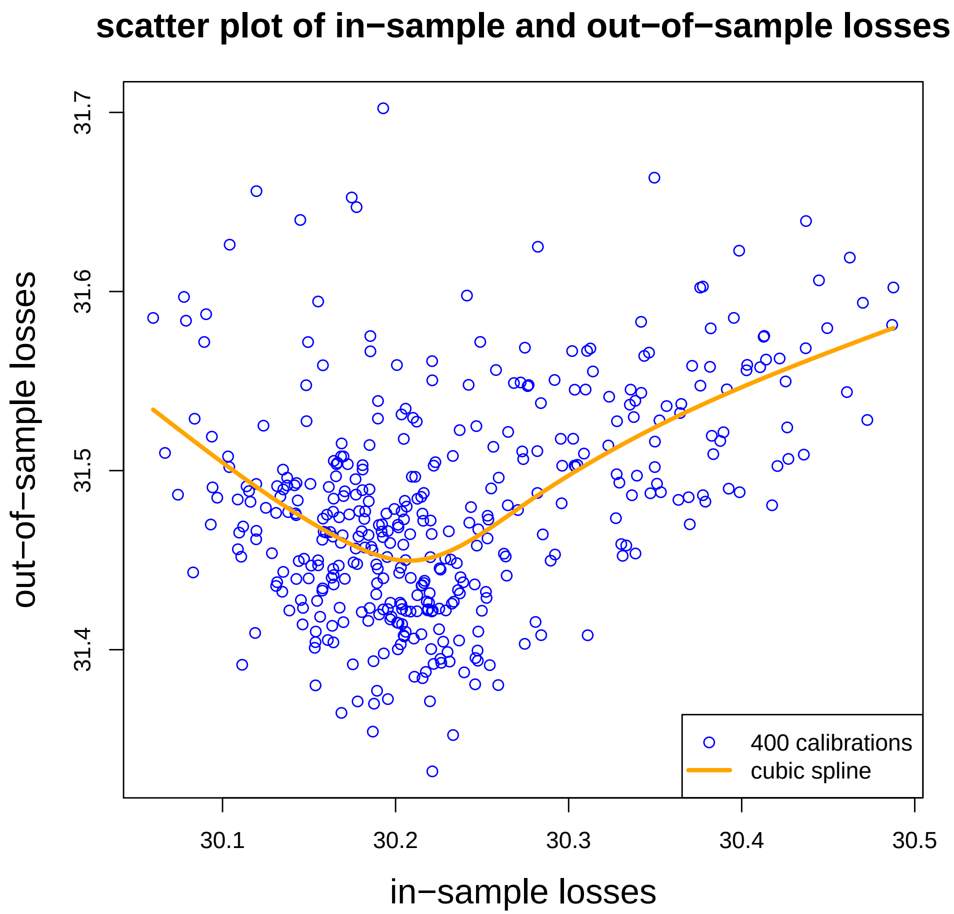



- Many network parameters $\vartheta$ produce the same loss figure for a given objective function $\vartheta \mapsto D^*(\boldsymbol{Y}, \vartheta)$, i.e. they are “equally good” (on portfolio level).

- The chosen network solution will depend in the initial seed of the algorithm.

- This is very troublesome for insurance pricing!

# Scatter Plot: In-Sample vs. Out-of-Sample Losses




This example is taken from Richman–Wüthrich (2020) and the in-sample losses are smaller than in the table above because we used different data cleaning.

- **The Nagging Predictor**

# The Nagging Predictor

- Breiman (1996) uses bootstrap aggregating = bagging to reduce noise in predictors.

- Aggregate over different network predictors of different calibrations $j \geq 1$

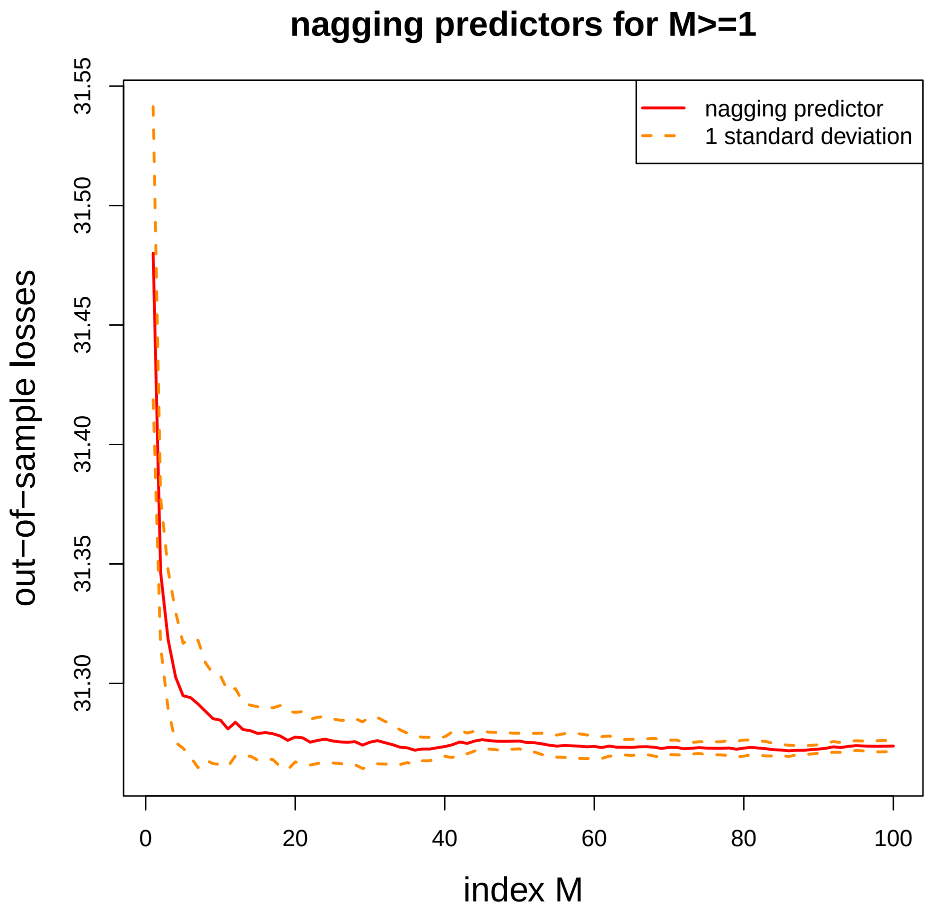
$$\bar{\mu}_i^{(M)} = \frac{1}{M}\sum_{j=1}^{M} \widehat{\mu}_i^{(j)} = \frac{1}{M}\sum_{j=1}^{M} \mu_{\widehat{\vartheta}^{(j)}}(\boldsymbol{x}_i).$$

- **Theorem (generalization loss).**
  Under suitable assumptions we have for deviance loss $D^*(Y_i, \cdot)$

$$\mathbb{E}\left[D^*\left(Y_i, \bar{\mu}_i^{(M)}\right)\right] \geq \mathbb{E}\left[D^*\left(Y_i, \bar{\mu}_i^{(M+1)}\right)\right] \geq \mathbb{E}\left[D^*(Y_i, \mu_i)\right],$$

  and there is also a corresponding asymptotic normality result (CLT).

# Nagging Predictor: Car Insurance Example



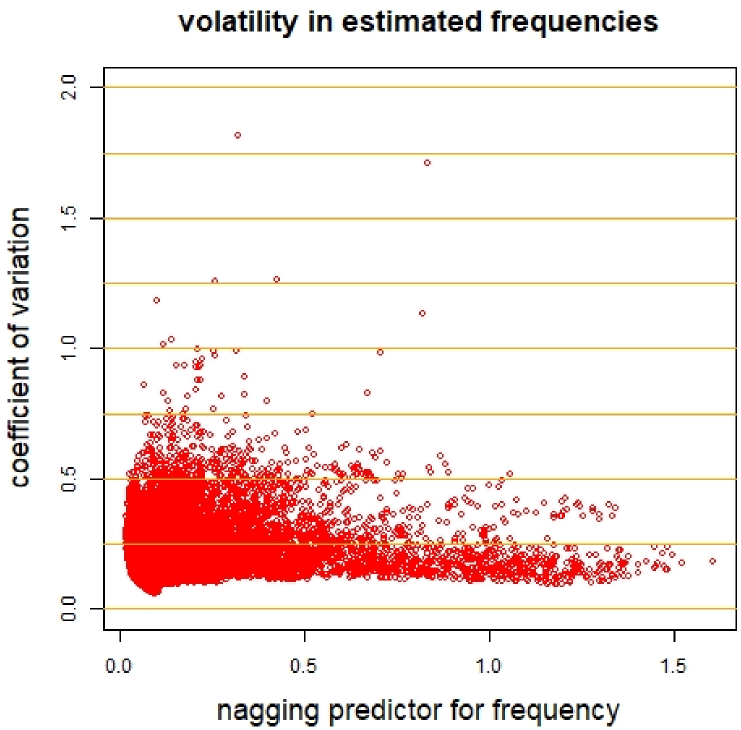

- After $M = 20$ iterations: the out-of-sample loss on portfolio has converged.
- After $M = 40$ iterations: confidence bounds are narrow.

# Car Insurance Frequency Poisson Example

| | epochs | run time | # param. | in-sample loss in $10^{-2}$ | out-of-sample loss in $10^{-2}$ | average frequency |
|---|---|---|---|---|---|---|
| homogeneous model | | – | 1 | 32.935 | 33.861 | 10.02% |
| Model GLM1 | | 20s | 49 | 31.267 | 32.171 | 10.02% |
| Deep FNN One-Hot | 250 | 152s | 1'306 | 30.268 | 31.673 | 10.19% |
| Deep FNN Emb($b = 1$) | 700 | 419s | 719 | 30.245 | 31.506 | 9.90% |
| Deep FNN Emb($b = 2$) | 600 | 365s | 792 | 30.165 | 31.453 | 9.70% |
| Deep FNN Emb($b = 2$) seed 1 | | 365s | 792 | 30.411 | 31.503 | 9.90% |
| Deep FNN Emb($b = 2$) seed 2 | | 365s | 792 | 30.352 | 31.418 | 10.23% |
| Deep FNN Emb($b = 2$) seed 3 | | 365s | 792 | 30.315 | 31.500 | 9.61% |
| Reg. FNN Emb($b = 2$) seed 1 | | +7s | 792 | 30.408 | 31.488 | 10.02% |
| Reg. FNN Emb($b = 2$) seed 2 | | +7s | 792 | 30.346 | 31.418 | 10.02% |
| Reg. FNN Emb($b = 2$) seed 3 | | +7s | 792 | 30.303 | 31.462 | 10.02% |
| Nagging predictor for $M = 400$ | | | | 30.060 | 31.272 | 10.02% |

The nagging predictor leads to a clear model improvement.

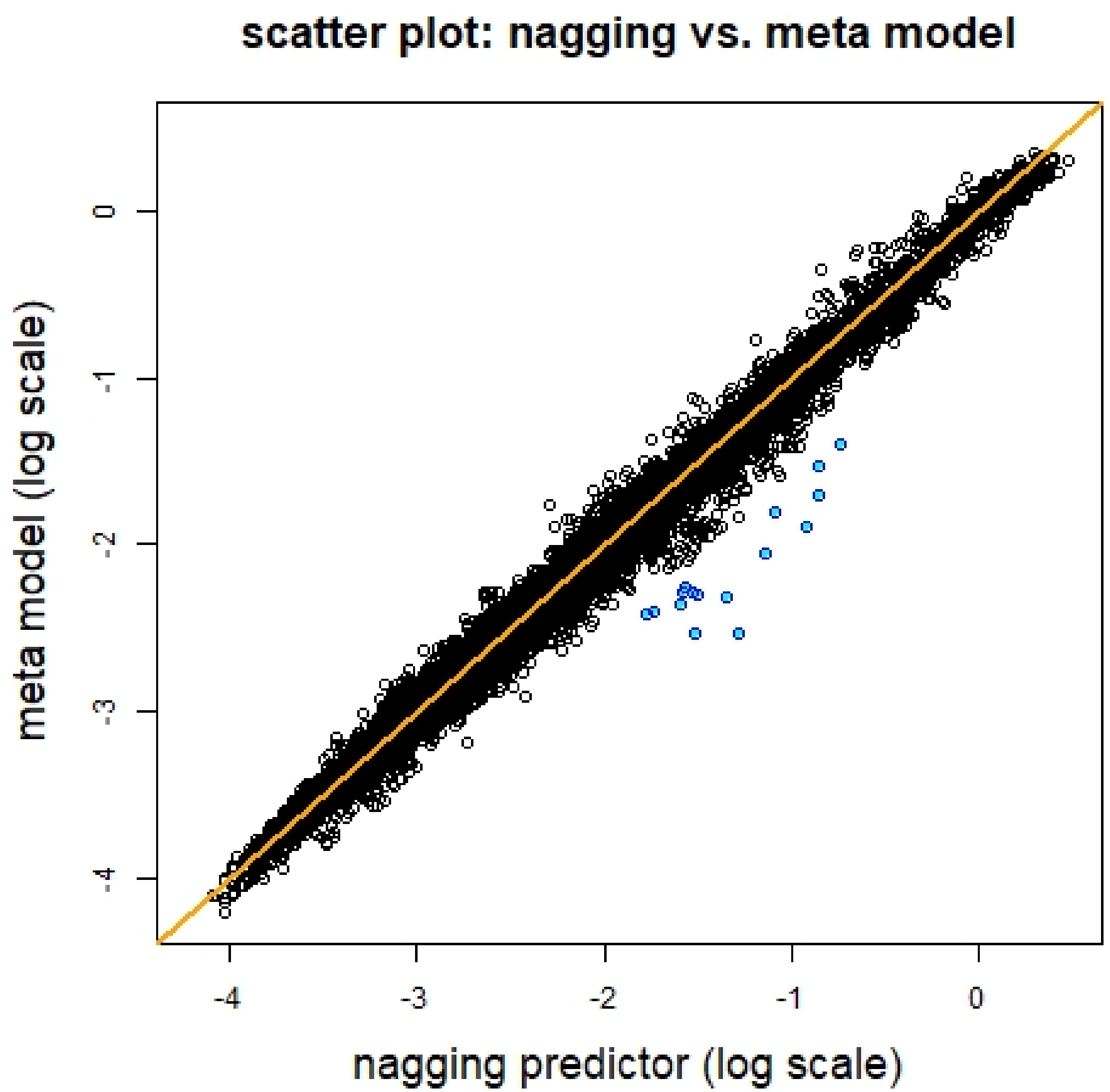
# Stability on Individual Policy Level




$$\widehat{\mathrm{CoV}}_i = \frac{\widehat{\sigma}_i}{\bar{\mu}_i^{(M)}} = \sqrt{\frac{1}{M-1}\sum_{j=1}^{M}\left(\widehat{\mu}_i^{(j)} - \bar{\mu}_i^{(M)}\right)^2} \Bigg/ \bar{\mu}_i^{(M)}.$$

- Individual policy level: average over 400 networks to get coefficient of variation (CoV) of $1/\sqrt{400} = 5\%$.

# Fit Meta Model to Nagging Predictor



- **LocalGLMnet: interpretable deep learning**

# Explainability of Deep FNN Predictors

- Network predictors are criticized for not being explainable (black box).

- There are many tools to make network predictors explainable a posteriori:
  - ⋆ partial dependence plots (PDPs)
  - ⋆ accumulated local effects (ALEs)
  - ⋆ locally interpretable model-agnostic explanation (LIME)
  - ⋆ SHapley Additive exPlanation (SHAP)
  - ⋆ marginal attribution by conditioning on quantiles (MACQ)

- Network predictors do not allow for variable selection.

- LocalGLMnet is an architecture that is explainable and allows for variable selection.

# LocalGLMnet Architecture

- A GLM has the following regression structure for parameter $\boldsymbol{\beta} \in \mathbb{R}^q$

$$g(\mu(\boldsymbol{x})) = \beta_0 + \langle \boldsymbol{\beta}, \boldsymbol{x} \rangle = \beta_0 + \sum_{j=1}^{q} \beta_j x_j.$$

- **Idea:** Estimate regression parameter $\boldsymbol{\beta} = \boldsymbol{\beta}(\boldsymbol{x})$ with a FNN.

- Define regression attentions through a deep FNN

$$\begin{aligned} \boldsymbol{\beta} : \mathbb{R}^q &\to \mathbb{R}^q \\ \boldsymbol{x} &\mapsto \boldsymbol{\beta}(\boldsymbol{x}) = \boldsymbol{z}^{(d:1)}(\boldsymbol{x}) = \left(\boldsymbol{z}^{(d)} \circ \cdots \circ \boldsymbol{z}^{(1)}\right)(\boldsymbol{x}). \end{aligned}$$

- The LocalGLMnet is defined by the additive decomposition

$$g(\mu(\boldsymbol{x})) = \beta_0 + \langle \boldsymbol{\beta}(\boldsymbol{x}), \boldsymbol{x} \rangle = \beta_0 + \sum_{j=1}^{q} \beta_j(\boldsymbol{x}) x_j.$$

# Interpretation of LocalGLMnet Architecture

- The LocalGLMnet is defined by the additive decomposition

$$g(\mu(\boldsymbol{x})) = \beta_0 + \langle \boldsymbol{\beta}(\boldsymbol{x}), \boldsymbol{x} \rangle = \beta_0 + \sum_{j=1}^{q} \beta_j(\boldsymbol{x}) x_j.$$

- We consider the different cases of regression attentions $\beta_j(\boldsymbol{x})$:

  ⋆ If $\beta_j(\boldsymbol{x}) \equiv \beta_j$: GLM term $\beta_j x_j$.
  ⋆ If $\beta_j(\boldsymbol{x}) \equiv 0$: drop term $x_j$.
  ⋆ If $\beta_j(\boldsymbol{x}) = \beta_j(x_j)$: no interactions with covariates $x_{j'}$ for $j' \neq j$.
  ⋆ Test for interactions: Calculate and analyze gradients

$$\nabla \beta_j(\boldsymbol{x}) = \left( \frac{\partial}{\partial x_1} \beta_j(\boldsymbol{x}), \ldots, \frac{\partial}{\partial x_q} \beta_j(\boldsymbol{x}) \right)^\top \in \mathbb{R}^q.$$

  ⋆ Careful, there is no identifiability: $\beta_j(\boldsymbol{x}) x_j = x_{j'}$.

# Implementation of LocalGLMnet

```
1  Design = layer_input(shape = c(q0),  dtype = 'float32', name = 'Design')
2  #
3  Attention = Design %>%
4            layer_dense(units=q1, activation='tanh', name='hidden1') %>%
5            layer_dense(units=q2, activation='tanh', name='hidden2') %>%
6            layer_dense(units=q3, activation='tanh', name='hidden3') %>%
7            layer_dense(units=q0, activation='linear', name='Attention')
8
9  Response  = list(Design, Attention) %>% layer_dot(axes=1) %>%
10           layer_dense(units=1, activation='linear', name='Response')
11 #
12 model <- keras_model(inputs = c(Design), outputs = c(Response))
```

Line 10 is needed to bring in the intercept $\beta_0$; but there are other ways to do so, see also next slide for explanation.

# Remarks and Preparation of Example

- There are other ways in implementing the intercept. Current solution:

$$g(\mu(\boldsymbol{x})) = \alpha_0 + \alpha_1 \sum_{j=1}^{q} \beta_j(\boldsymbol{x}) x_j.$$

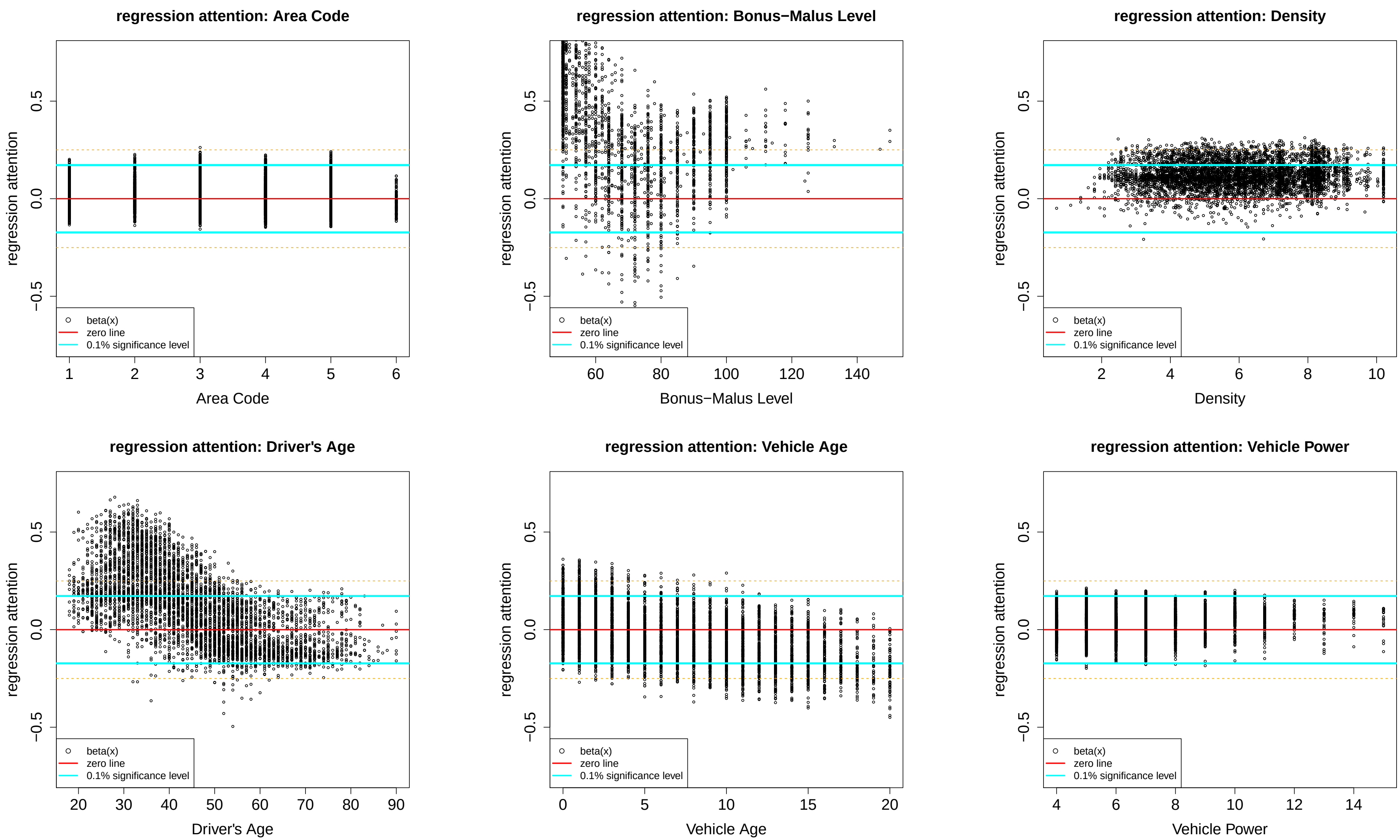
- Categorical variables should either use:
  - ⋆ one-hot encoding
  - ⋆ dummy coding with normalization to centering and unit variance

  ▷ These codings will allow for interaction of all levels.

- We normalize all covariates to centering and unit variance: comparability!

- Fitting is done completely analogously to a FNN with SGD. Resulting networks are competitive with classical FNNs.

# Example: Regression Attentions LocalGLMnet



# Confidence Bounds for Variable Selection

- Add (two) purely random covariates (with different distributions)

$$x_{i,q+1} \overset{\text{i.i.d.}}{\sim} U\left[-\sqrt{3}, \sqrt{3}\right] \qquad \text{and} \qquad x_{i,q+2} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1).$$

  These covariates are standardized.

- Consider extended regression function for $\boldsymbol{x}^+ = (x_1, \ldots, x_q, x_{q+1}, x_{q+2})^\top \in \mathbb{R}^{q+2}$

$$g(\mu(\boldsymbol{x}^+)) = \beta_0^+ + \left\langle \boldsymbol{\beta}^+(\boldsymbol{x}^+), \boldsymbol{x}^+ \right\rangle = \beta_0^+ + \sum_{j=1}^{q+2} \beta_j^+(\boldsymbol{x}^+) x_j.$$

- Magnitudes of $\widehat{\beta}_{q+1}^+(\boldsymbol{x}^+)$ and $\widehat{\beta}_{q+2}^+(\boldsymbol{x}^+)$ determine size of insignificant components.

# Confidence Bounds for Variable Selection

- Magnitudes of $\widehat{\beta}^+_{q+1}(\boldsymbol{x}^+)$ and $\widehat{\beta}^+_{q+2}(\boldsymbol{x}^+)$ determine size of insignificant components.

- Determine empirical mean and standard deviations for $j = q+1, q+2$

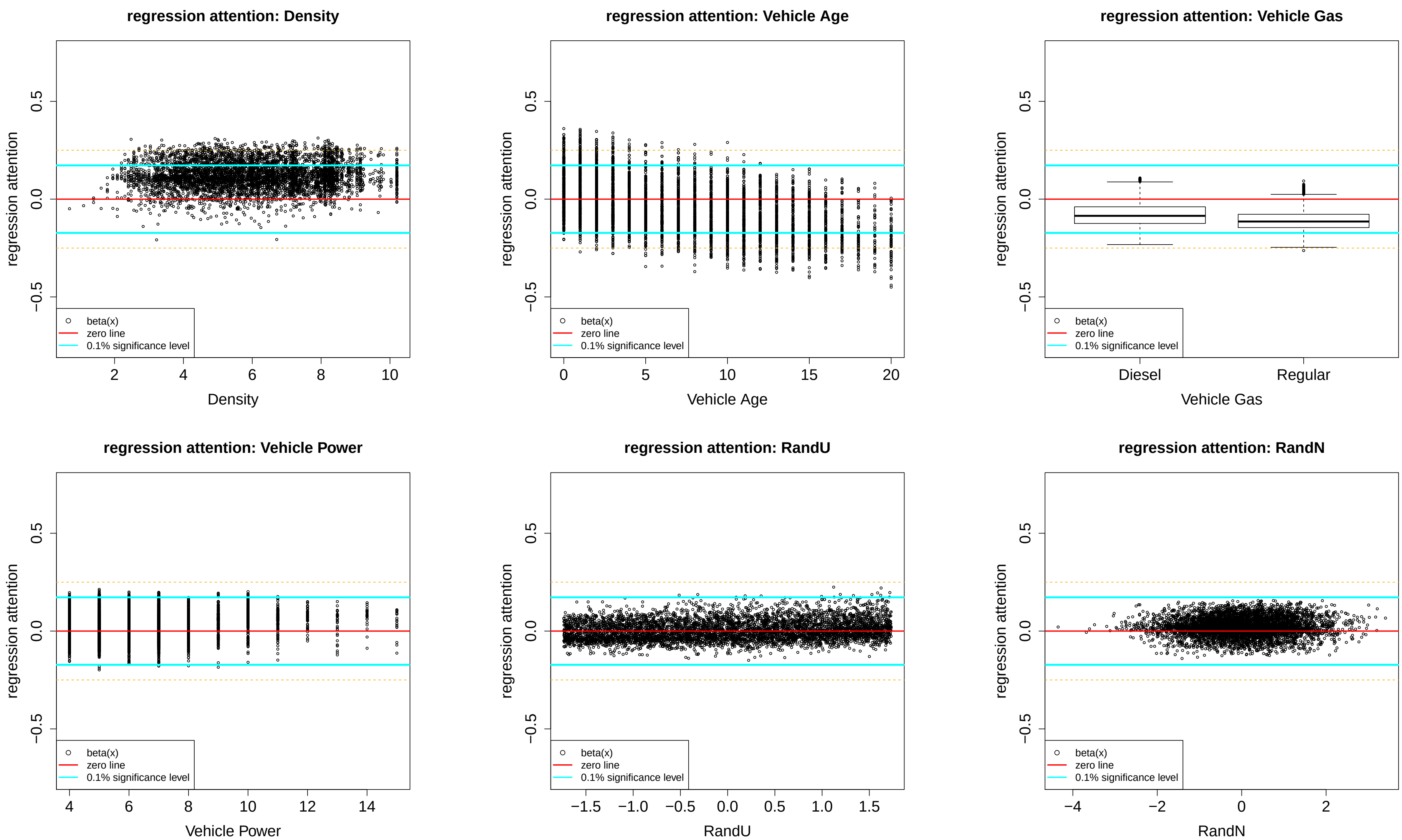
$$\bar{b}_j = \frac{1}{n}\sum_{i=1}^{n} \widehat{\beta}^+_j(\boldsymbol{x}^+_i) \quad \text{and} \quad \widehat{s}_j = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(\widehat{\beta}^+_j(\boldsymbol{x}^+_i) - \bar{b}_j\right)^2}.$$

- Null hypothesis $H_0 : \beta_j(\boldsymbol{x}) = 0$ for component $j$ on significance level $\alpha \in (0, 1/2)$ can be rejected if the coverage ratio of the following interval

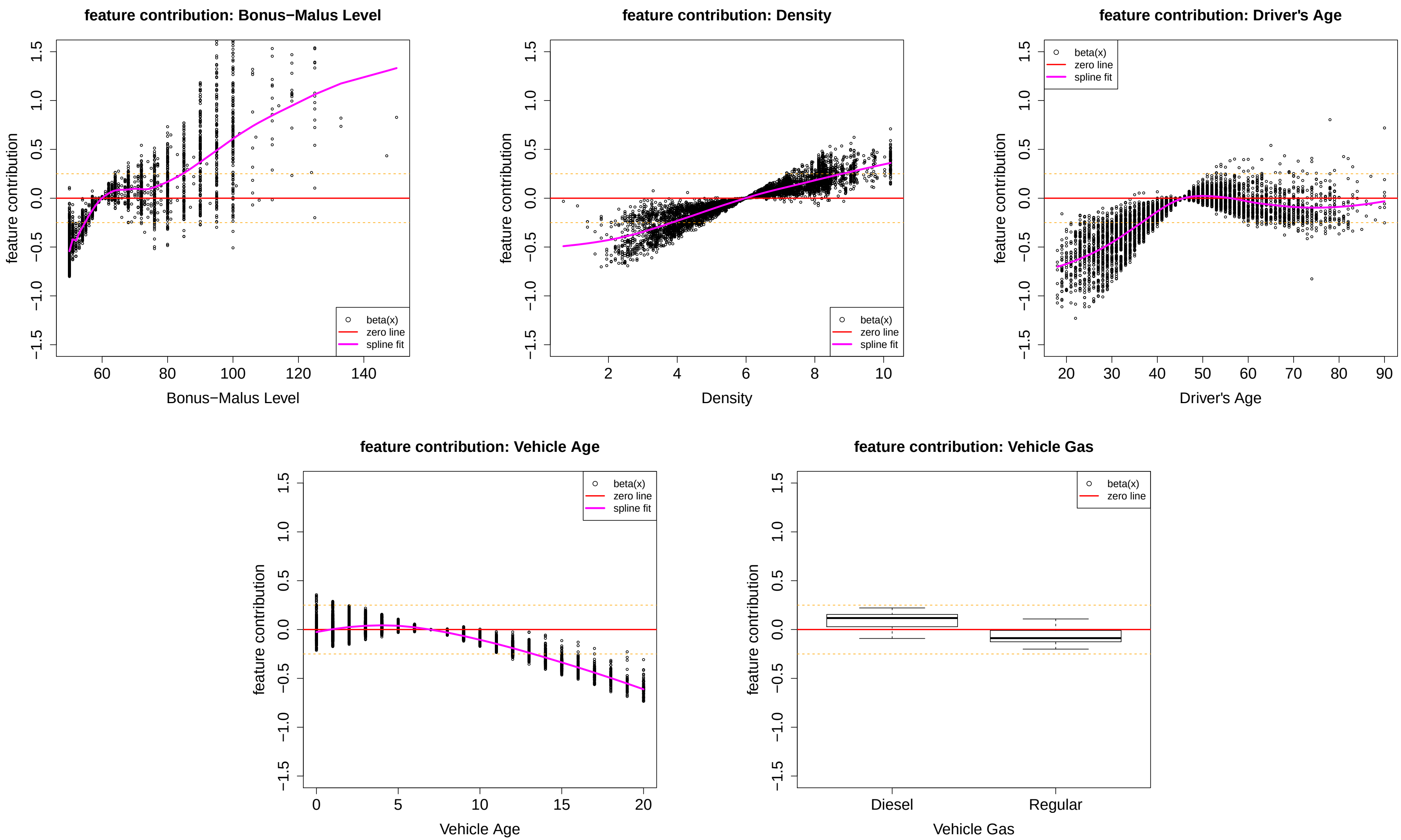
$$I_\alpha = \left[q_{\mathcal{N}}(\alpha/2) \cdot \widehat{s}_{q+1},\ q_{\mathcal{N}}(1-\alpha/2) \cdot \widehat{s}_{q+1}\right]$$

is substantially smaller than $1-\alpha$, where $q_{\mathcal{N}}(p)$ denotes the standard Gaussian quantile on level $p \in (0, 1)$.

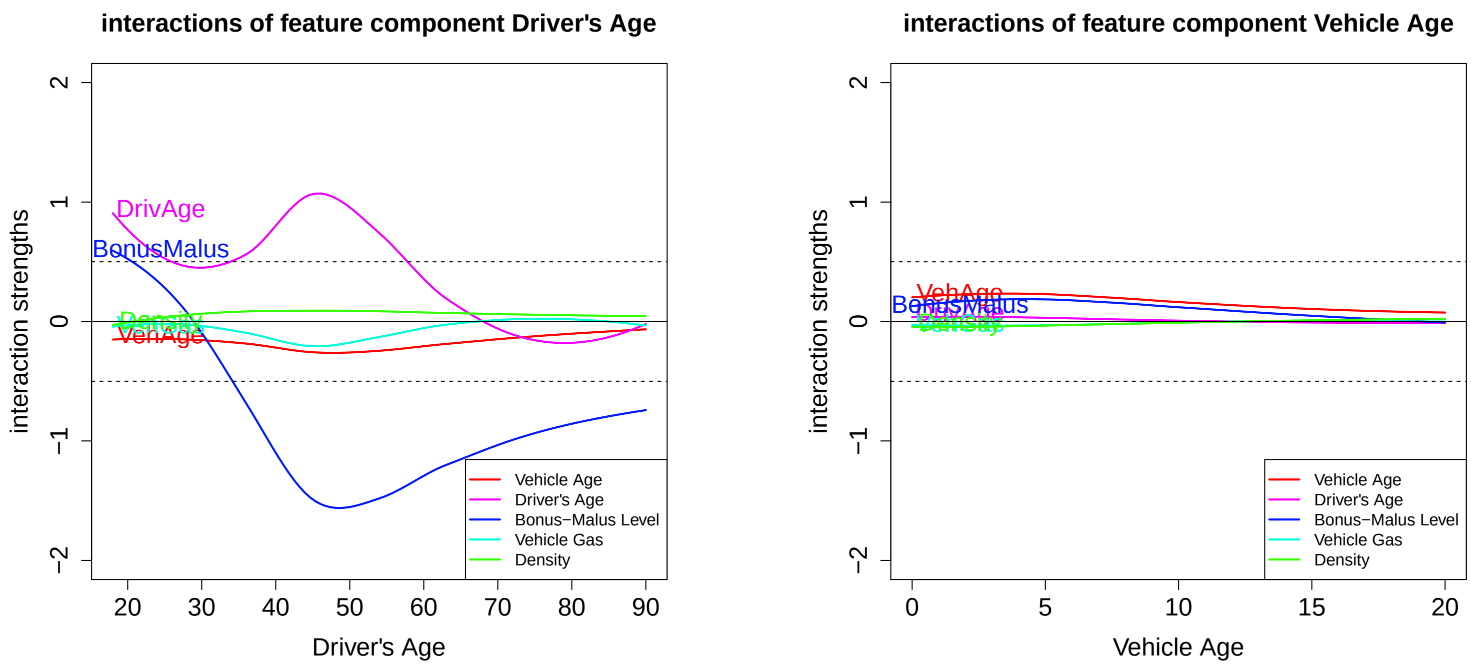
# Variable Selection with Cyan Confidence Bounds



# Covariate Contributions $\widehat{\beta}_j(\boldsymbol{x})x_j$



# Interactions between Covariate Components



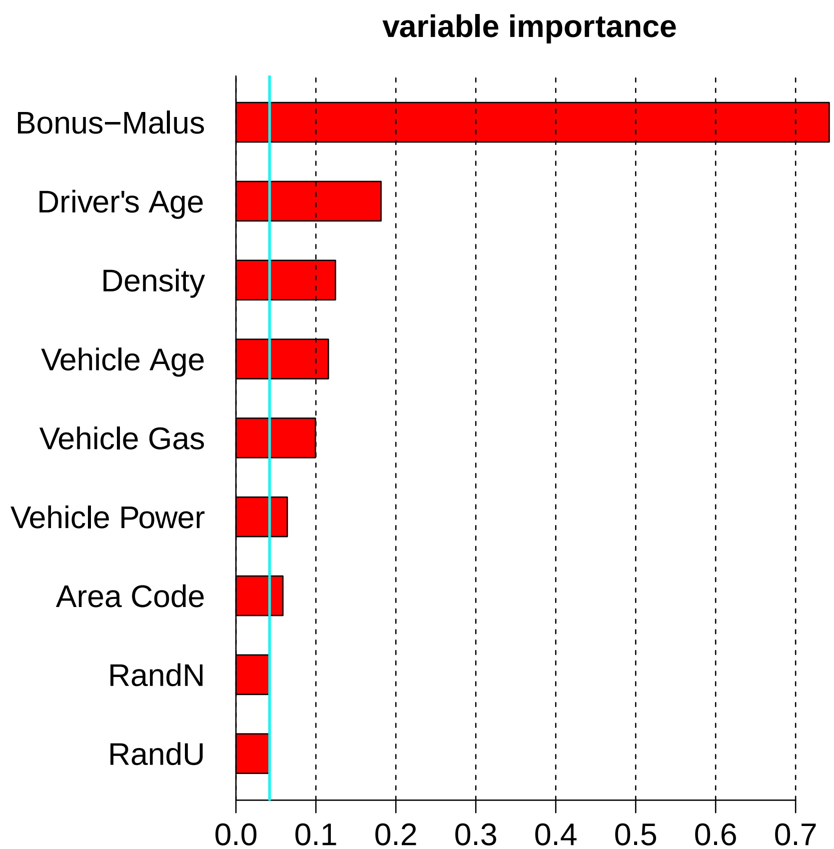

$$\nabla \beta_j(\boldsymbol{x}) = \left( \frac{\partial}{\partial x_1} \beta_j(\boldsymbol{x}), \ldots, \frac{\partial}{\partial x_q} \beta_j(\boldsymbol{x}) \right)^\top \in \mathbb{R}^q.$$

# Calculation of Gradients in keras

```
j <- 1  # select the feature component
#
beta.j  <- Attention %>% layer_lambda(function(x) x[,j])
model.j <- keras_model(inputs = c(Design), outputs = c(beta.j))
#
grad <- beta.j %>% layer_lambda(function(x) k_gradients(model.j$outputs,
                                                            model.j$inputs))
model.grad <- keras_model(inputs = c(Design), outputs = c(grad))
#
grad.beta <- data.frame(model.grad %>% predict(as.matrix(XX)))
```

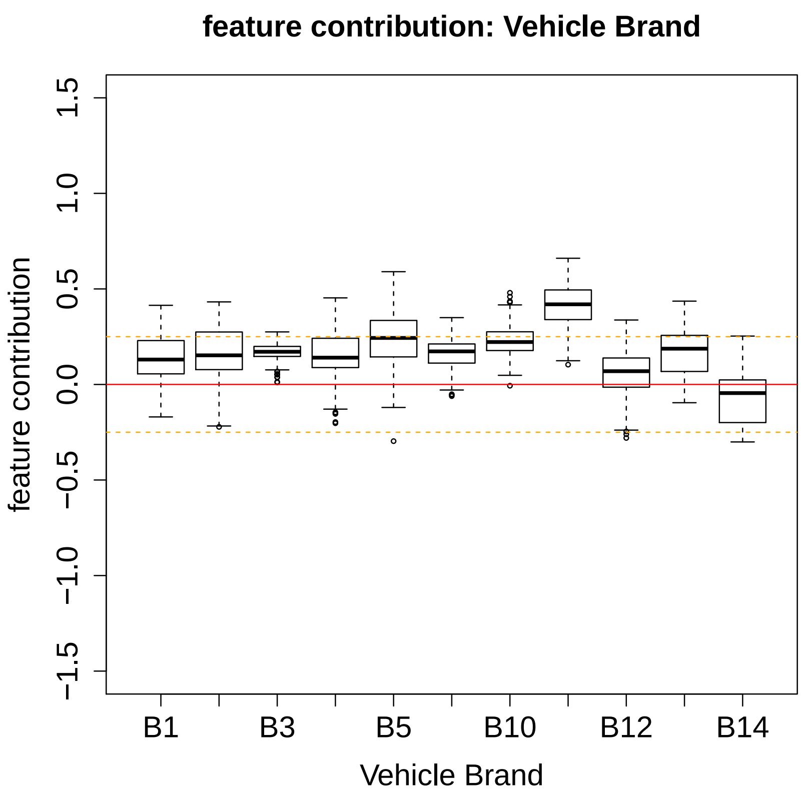
In different TensorFlow/keras versions this may be slightly different.

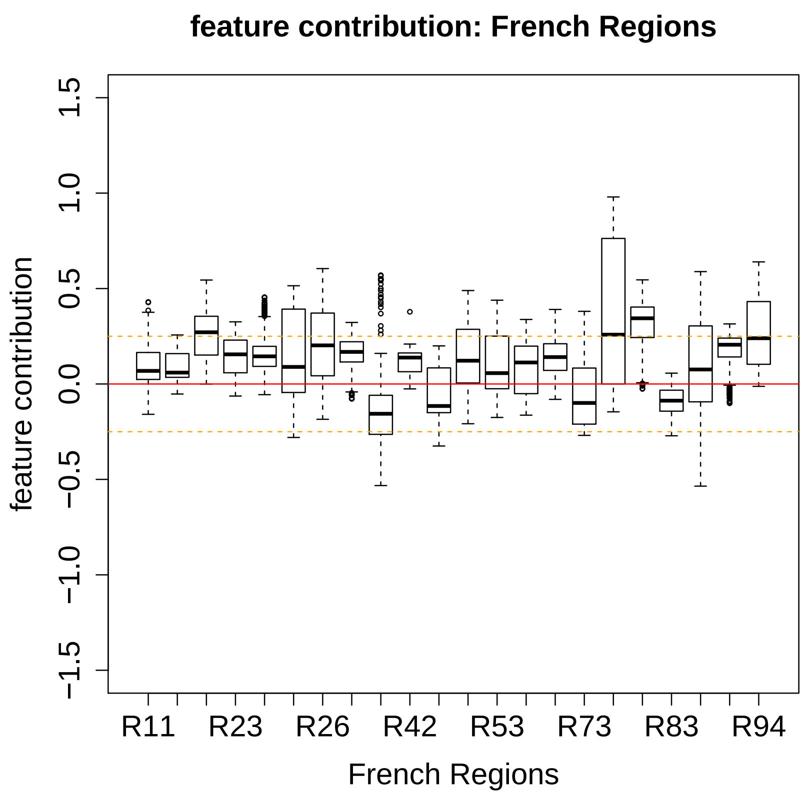
# Variable/Term Importance




$$\mathrm{VI}_j = \frac{1}{n} \sum_{i=1}^{n} \left| \widehat{\beta}_j(\boldsymbol{x}_i) \right|.$$

# Categorical Covariate Components




LASSO regularization within LocalGLMnets: see Richman–Wüthrich (2021b).

# References


- Breiman (1996). Bagging predictors. Machine Learning 24, 123-40.
- Efron, Hastie (2016). Computer Age Statistical Inference: Algorithms, Evidence, and Data Science. Cambridge UP.
- Ferrario, Noll, Wüthrich (2018). Insights from inside neural networks. SSRN 3226852.
- Goodfellow, Bengio, Courville (2016). Deep Learning. MIT Press.
- Hastie, Tibshirani, Friedman (2009). The Elements of Statistical Learning. Springer.
- Lorentzen, Mayer (2020). Peeking into the black box: an actuarial case study for interpretable machine learning. SSRN 3595944.
- Noll, Salzmann, Wüthrich (2018). Case study: French motor third-party liability claims. SSRN 3164764.
- Richman (2020a/b). AI in actuarial science – a review of recent advances – part 1/2. Annals of Actuarial Science.
- Richman, Wüthrich (2020). Nagging predictors. Risks 8/3, 83.
- Richman, Wüthrich (2021a). LocalGLMnet: interpretable deep learning for tabular data. SSRN 3892015.
- Richman, Wüthrich (2021b). LASSO regularization within the LocalGLMnet architecture. SSRN 3927187.
- Schelldorfer, Wüthrich (2019). Nesting classical actuarial models into neural networks. SSRN 3320525.
- Schelldorfer, Wüthrich (2021). LocalGLMnet: a deep learning architecture for actuaries. SSRN 3900350.
- Wüthrich (2020). Bias regularization in neural network models for general insurance pricing. European Actuarial Journal 10/1, 179-202.
- Wüthrich, Buser (2016). Data Analytics for Non-Life Insurance Pricing. SSRN 2870308, Version September 10, 2020.
- Wüthrich, Merz (2019). Editorial: Yes, we CANN! ASTIN Bulletin 49/1, 1-3.
- Wüthrich, Merz (2021). Statistical Foundations of Actuarial Learning and its Applications. SSRN 3822407.