PROJECT

BLAL BLA BLA

anonymous

1 Introduction

- Background
- Problem formulation/scope
- Main modeling idea
- Some picture of the data?

2 Data Desctription

The "cath" dataset used in this report is obtained from Duke University Cardiovascular Disease Databank. It encapsulates a collection of 6 variables (@Data-table) that are closely related to cardiovascular health.

Table 1: TBD

sex	age	cad_dur	choleste	sigdz	tvdlm
0	73	132	268	1	1
0	68	85	120	1	1
0	54	45	NA	1	0
1	58	86	245	0	0
1	56	7	269	0	0
0	64	0	NA	1	0

The dataset consists of four explanatory variables (*sex*, *age*, *cad_dur*, *choleste*) and two response variables (*sigdz*, *tvdlm*) that provide an overview on patient demographics, clinical indicators, and critical outcomes related to coronary artery disease:

- **Sex** (*sex*): Categorized as 0 for male and 1 for female, this variable represents the gender distribution within our dataset.

- **Age** (*age*): Representing the age of patients in years, this variable serves as a demographic feature.
- **Chest Pain Duration** (*cad_dur*): The duration of chest pain symptoms in days.
- **Serum Cholesterol Level** (*choleste*): Measured in milligrams per deciliter, serum cholesterol levels are indicative of lipid metabolism and play a crucial role in cardiovascular health.
- **Significant Coronary Disease** (*sigdz*): A binary variable that captures the presence (1) or absence (0) of at least 75% blockage in one of the major coronary arteries.
- **Three Vessel Disease or Left Main Disease** (*tvdlm*): Denoting the presence (1) or absence (0) of blockage in either all three coronary vessels or in the left main coronary artery.

The univariate distributions of these variables are visualized in @Univariate-analysis.

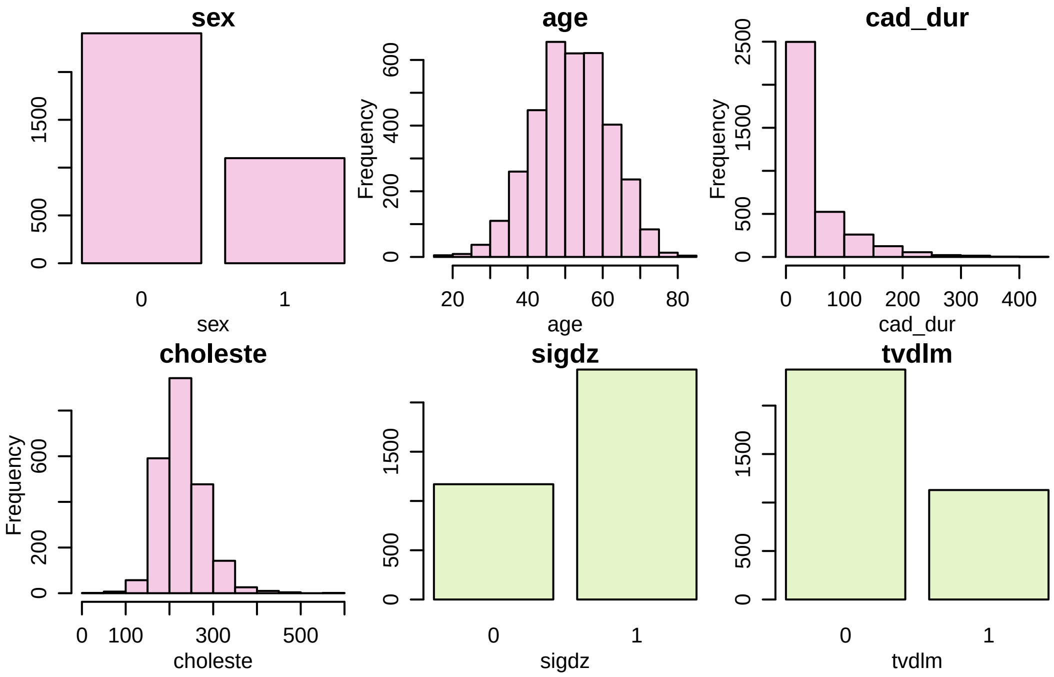


Figure 1: TBD 1

While constructing the Bayesian models to predict the probaility of significant coronary disease, the report strives to utilize the correlation between the explanatory variables (*sex*, *age*, *cad_dur*, *choleste*) and the desired response variable (*sigdz*). The *tvdlm* variable is not relevant in this report as the main focus is to predict the probability of significant coronary disease, independent of the type of the blockade.

Before the analysis, the data is preprocessed by removing *tvdlm* column and all rows that contain missing values, as well as by scaling the continuous variables to zero mean and unit variance. After this, we are left with $n = 2258$ observations. The pairwise correlations of variables are visualized in @Bivariate-analysis. We can see that variables *sex* and *age* have the most significant bivariate correlation to the responsive variable *sigdz*.

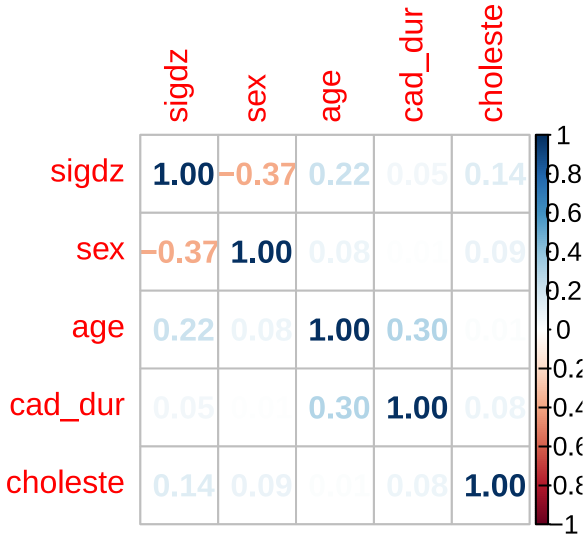


Figure 2: TBD 3

3 Mathematical Model

In this analysis, we will construct two models for inferring the binary response variable, *sigdz* based on input explanatory variables. The first model is a generalized linear model (GLM), namely Bayesian logistic regression. The other model is a generalized additive mixed model (GAMM), which implements Bayesian logistic regression with nonlinear transformations on the input variables. These models will be referred as linear and nonlinear model, respectively.

(((To-be-done still: - Check likelihood notation for nonlinear model - Prior justification - Check prior notations - Include priors with own values ? - **Posteriors** ?)))

3.1 The generalized linear model and priors

Let y be the number of times the variable *sigdz* is realized to be 1 for one individual in the dataset, and let x be the explanatory variables for this outcome. Then, this number of successes for one individual follows a Binomial distribution

$$y \sim \binom{n}{y} \theta^y (1-\theta)^{n-y},$$

where n is the number of observations for that specific individual and $\theta = g^{-1}(\eta)$ $(\eta = \alpha + x^T \beta)$ is the probability of success (patient presenting with significant coronary disease). The inverse link function g^{-1} maps the output of the linear predictor η to a probability interval between 0 and 1.

For the binomial GLM, this project utilizes logit $g(x) = \ln(\frac{x}{1-x})$ as a link function, which makes it a logistic regression model. As each individual occurs only once in the data, y can be directly presented as the binary response variable. Therefore, the likelihood of the response variable of one individual is reduced to Bernoulli distribution

$$y \sim \text{ logit}^{-1}(\eta)^y (1 - \text{ logit}^{-1}(\eta))^{1-y}.$$

The complete data likelihood is then a product of $n = 2258$ likelihoods, with unshared probability of success.

As in Bayesian logistic regression the scope is to infer the distribution of the regression weights, namely the intercept α and coefficients $\beta = [\beta_1, \beta_2, \beta_3, \beta_4]^T$, we define the prior to be Student's t-distribution

$$\begin{aligned} \alpha &\sim t_v(\mu, \sigma) \\ \beta_k &\sim t_v(\mu, \sigma), \;\; k = 1, .., 4 \end{aligned}$$

where v is the degrees of freedom, μ is the location and σ is the scale.

The selection of prior distribution was done based on the nature of the data. Due to correlations, there is reason to believe that the parameters are not very close to zero, but most are still rather small than large. Therefore, as Student's t-distribution has heavy tails and larger scale compared to, for example, Gaussian distribution, t-distribution is a suitable choice of prior for this purpose. The parameters of the prior were defined to be

$$v = 3, \quad \mu = 0, \quad \sigma = 2.5,$$

as the coefficients can be positive or negative.

Finally, the joint posterior distribution that is simulated using Hamiltonian Monte Carlo (HMC) is proportional to the product of likelihood and prior distributions:

$$p(\alpha, \beta | \mathbf{x}, \mathbf{y}) \propto t_v(\alpha | \mu, \sigma) \times \prod_{k=1}^{4} t_v(\beta_k | \mu, \sigma) \times \prod_{i=1}^{n} \text{ logit}^{-1}(\eta_i)^{y_i} (1 - \text{ logit}^{-1}(\eta_i))^{1-y_i}$$

3.2 The generalized additive nonlinear model

The additive nonlinear model on the other hand, is a generalized additive mixed model which combines multiple functions in a way that is not strictly linear. This allows for a more flexible relationship between the explanatory variables and response variable y. In this report, the nonlinear model uses the same link function as the linear model, and so the nonlinear predictor for the logistic regression model can be written as:

$$\text{logit}(\theta) = \eta = \alpha + \sum_{k=1}^{4} \beta_k f_k(x_k),$$

where f_k are nonlinear functions that transform the explanatory features individually. In this report, all functions of the continuous variables are smoothing functions. The f_{age} is simply the variable itself, due to *sex* being binary variable. The smoothing functions utilize penalized splines, allowing the model to create a curved relationship between the features. The shape of the smoothing functions is estimated from the data and the penalty helps avoid overfitting. The smoothing function works by minimizing the sum of the model fit with the smoothness / penalty (here, thin plate regression splines (default)).

The likelihood of observation is the same as with the linear model, with the difference that a nonlinear transformation is applied to all input explanatory variables $\mathbf{x}$. Additionally, we utilize the same prior in both models to make them as comparable as possible. The posterior is again similar as with the linear model, but with the difference of nonlinear transformations on the input data.

4 Model Definitions and Implementation

The linear and nonlinear models are implemented as Stan code with the rstanarm package as described below. Both models were implemented with identical number of chains, draws and warm-up. The default values (chains = 4, draws = 4000, and warmup = 2000) were used in both cases.

4.1 Linear model

The linear model was implemented with the help of the `stan_glm` function from the rstanarm package. `stan_glm` is used to fit generalized linear models and performs a full Bayesian estimation with Markov Chain Monte Carlo (MCMC) estimation instead of maximum likelihood estimation and by adding priors to the GLM coefficients and intercept. By defining the model parameter `family = binomial(link = 'logit')` the model performs logistic regression.

The linear relationship between the response variable `sigdz` and the explanatory variables are defined with the help of the `formula` function and the prior for the regression coefficients and intercept with the help of the `student_t` function.

```r
# Make response variable a factor
cath$sigdz <- as.factor(cath$sigdz)
y <- cath$sigdz

# Formula
formula_linear <- formula(sigdz ~ sex + age + cad_dur + choleste)

# Prior
prior_linear <- student_t(df = 3, location = 0, scale = 2.5)

# The model
model_linear <- stan_glm(formula_linear, data = cath,
                 family = binomial(link = "logit"),
                 prior = prior_linear, prior_intercept = prior_linear,
                 QR=TRUE, refresh=0)

# saveRDS(model_linear, file = "./additional_files/model_linear.rds")
# model_linear <- readRDS("./additional_files/model_linear.rds")
```

4.2 Nonlinear model

The nonlinear model is similarly implemented utilizing `stan_gamm4` function from the rstanarm package and defined to be a logistic regression model with the help of the model parameter `family = binomial(link = 'logit')`. `stan_gamm4` fits a generalized additive mixed model, and performs Bayesian MCMC estimation instead of maximum likelihood estimation. In the same way as `stan_glm`, the model adds independent priors to the regression coefficients and intercept.

The nonlinear relationship between the response variable `sigdz` and the explanatory variables is again defined with the help of the `formula` function. This time, passing the smoothing function `s()` for all continuous explanatory variables, to allow for more complexity in the model, while simultaneously penalizing over-fitting of the model with the the thin plate regression splines smoothness.

The same priors are used as for the linear model for both the intercept and the regression coefficients.

```
# Formula
formula_nonlinear <- formula(sigdz ~ sex + s(age) + s(cad_dur) + s(choleste))

# Model definition
model_nonlinear <- stan_gamm4(
  formula_nonlinear, data = cath,
  family = binomial(link = "logit"),
  prior = prior_linear, prior_intercept = prior_linear,
  refresh = 0
)

# saveRDS(model_nonlinear, file = "./additional_files/model_nonlinear.rds")
# model_nonlinear <- readRDS("./additional_files/model_nonlinear.rds")
```

5 Model Evaluation

After fitting the models, multiple evaluation metrics such as split-$\hat{R}$, effective sample size (ESS) and number of divergent transitions were used to assess the convergence of MCMC chains separately for each model. Additionally, we perform posterior predictive checks, assess the model performances as well as compare the models utilizing leave-one-out cross validation (LOO-CV). Finally, we perform prior sensitivity analysis for both models.

- Rhat, ESS, HMC divergences, pp_check(), loo_compare(), classification accuracy
- Prior sensitivity analysis

5.1 Convergence diagnostics & posterior predictive checks

For the linear model, the posterior predictive check and posterior distributions of the parameters with 95 % credible interval are visualized in @linear-model-pp-check. As we can see, the model fits the data relatively well, with a bit of a variation around the probability interval endpoints.

The convergence diagnostics for the linear model are summarized below.

```
Split-Rhat:

  (Intercept)           sex           age       cad_dur      choleste
        1.000         1.000         1.000         0.999         1.000
     mean_PPD log-posterior
        1.000         1.002
```

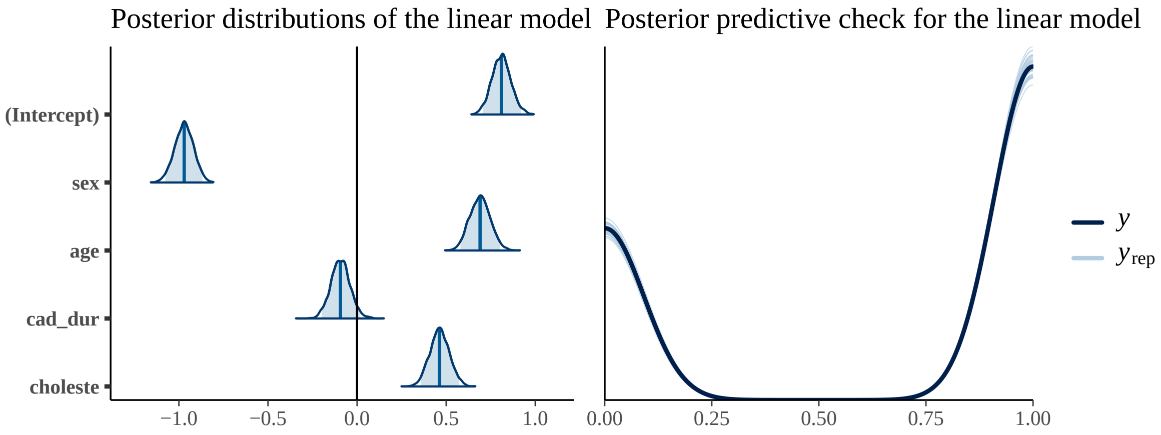


Figure 3: TBD

```
Number of divergent transitions:  0
```

```
ESS ratio:
```

```
(Intercept)         sex         age     cad_dur    choleste 
    1.11550     0.95950     0.99525     1.42925     1.13925 
```

```
Sd of ESS ratio:  0.185
```

The HMC chains have converged, as all split-$\hat{R}$ values are below 0.01. Additionally, there were no divergent transitions during convergence, and thus the HMC simulation is reliable. The ratio of the ESS to the true sample size is over 1 with all explanatory variables as well as the intercept. Although high ESS is generally a good thing, this also implies the MCMC samples may have a negative correlation.

For the nonlinear model, the posterior predictive check is visualized in @nonlinear-model-pp-check. Based on visual inspection, the nonlinear model fits the data as well as the linear model. Like with the linear model, the posterior has a bit of a variation around the probability interval endpoints.

The convergence diagnostics for the nonlinear model are summarized below.

```
Split-Rhat of the first coefficients:
```

```
(Intercept)         sex    s(age).1    s(age).2    s(age).3    s(age).4 
      1.000       1.001       1.000       0.999       1.001       1.000 
```

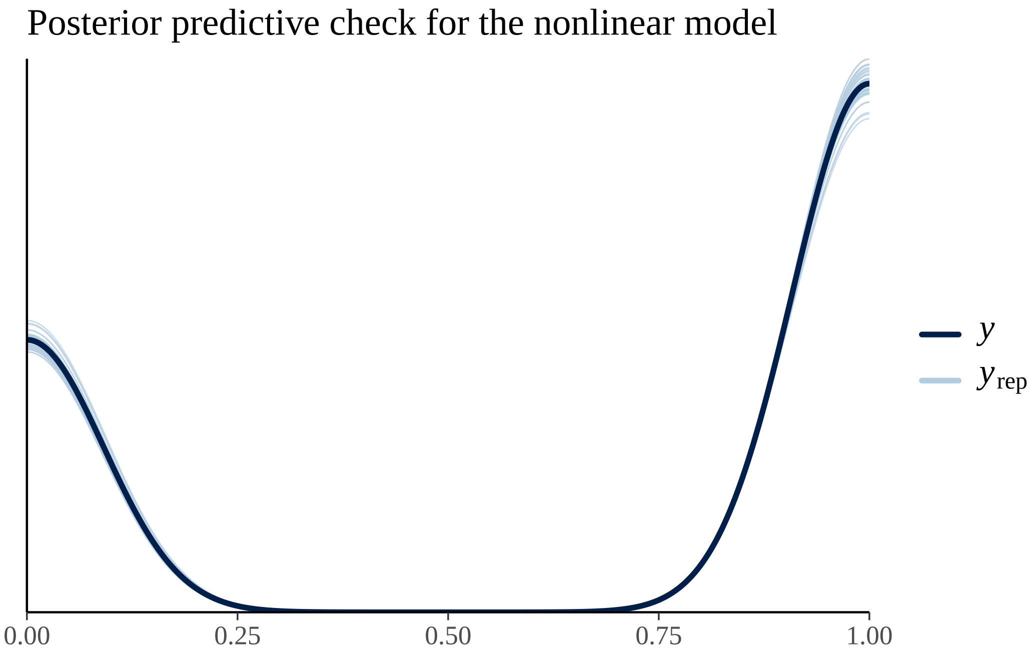


Figure 4: TBD

```
All split-Rhat <= 0.01:  TRUE

Number of divergent transitions:  0

ESS ratio of the first coefficients:

(Intercept)         sex    s(age).1    s(age).2    s(age).3    s(age).4
      1.447       1.301       0.977       0.882       0.467       0.845

Mean of ESS ratio:  0.879

Sd of ESS ratio:  0.368
```

Also the MCMC chains for this model have converged, as the split-$\hat{R}$ is less than 0.01 for all coefficients. As with the linear model, there were no divergent transitions during convergence, and thus the HMC simulation is reliable. On the other hand, the ESS ratio is on average less than 1, so a bit lower than with the linear model. However, the standard deviation of the ESS ratios is over twice larger than with the linear model. Low ESS ratio is not good in the model sense, but nevertheless the nonlinear has multiple coefficients to balance out the overall effect.

5.2 Model comparison using LOO-CV

To compare the performance of the linear model and nonlinear model to each other as well as to a baseline model, we will compute the expected log-densities of the predictive distributions (ELPD) by applying Pareto smoothed LOO-CV (PSIS-LOO) to the models. The baseline model is simply a logistic regression model without any explanatory variables and with a unit coefficient.

```r
# Baseline model
model_baseline <- update(model_linear, formula = sigdz ~ 1, QR = FALSE)
```

The results of the PSIS-LOO for each model are presented below.

```
Linear model:


Computed from 4000 by 2258 log-likelihood matrix

         Estimate   SE
elpd_loo  -1177.4 24.4
p_loo         5.1  0.2
looic      2354.8 48.7
------
Monte Carlo SE of elpd_loo is 0.0.

All Pareto k estimates are good (k < 0.5).
See help('pareto-k-diagnostic') for details.

Nonlinear model:


Computed from 4000 by 2258 log-likelihood matrix

         Estimate   SE
elpd_loo  -1170.5 24.3
p_loo        11.6  0.7
looic      2341.1 48.7
------
Monte Carlo SE of elpd_loo is 0.1.

All Pareto k estimates are good (k < 0.5).
See help('pareto-k-diagnostic') for details.
```

```
Baseline model:


Computed from 4000 by 2258 log-likelihood matrix

         Estimate   SE
elpd_loo  -1448.7 14.9
p_loo         1.0  0.0
looic      2897.3 29.9
------
Monte Carlo SE of elpd_loo is 0.0.

All Pareto k estimates are good (k < 0.5).
See help('pareto-k-diagnostic') for details.
```

For each model, all Pareto k estimates are < 0.5, which implies the importance sampling gives a reliable estimate on the computed ELPDs. Additionally, for linear and nonlinear model the p_loo is less than the number of parameters in the respective model, and thus the model specifications seem to be reasonable.

To assess the model ELPDs with respect to each other, we compute the comparison between the results of PSIS-LOO:

```
               elpd_diff se_diff
model_nonlinear    0.0       0.0
model_linear      -6.9       3.4
model_baseline  -278.1      21.8
```

It indeed seems that the predictive performances of both linear and nonlinear models are better compared to the baseline model. Additionally, the scale of the difference of standard errors (se_diff) of the baseline model and nonlinear model is smaller than the difference of the ELPDs of the nonlinear and baseline model, which implies the difference in predictive log-densities between these models is not simply explained by the variance. Although the difference in ELPDs of the linear and nonlinear models is small, the nonlinear model slightly outperforms the linear model.

5.3 Posterior predictive performance: Classification accuracy

To estimate the generalization error, i.e. the generalization and performance of models on unseen data, we compute LOO-balanced classification accuracies for the linear and nonlinear models. The results are summarized in @Classification-accuracies. The classification accuracy is simply the fraction of correctly classified observations. The balanced classification accuracy on the other hand takes into account true positive rate (sensitivity) and true negative rate (specificity). The latter accounts for the imbalance in our data and is therefore more accurate estimate on the generalization error. Estimating generalization error is highly important for practical usage of the model, as it would be centered around predicting

Table 2: TBD

	classification_accuracy	balanced_classification_accuracy
linear_model	0.75	0.69
nonlinear_model	0.76	0.69

We can see that the performances of both models are nearly equal. The classification accuracy is slightly better for the nonlinear model, but the balanced accuracy is the same for both models. The the balanced classification accuracy is not very high, but nevertheless, both models outperform a random classifier.

To further compare the predictive performances of the models, calibration plots of both linear and nonlinear models are visualized in @Calibration-plots. From these plots we can see that both models are calibrated on approximately the interval of [0.25, 0.5] of predicted probabilities. For the top 10% there are more predictions than observations, and for the bottom 20% there are more observations than predictions.

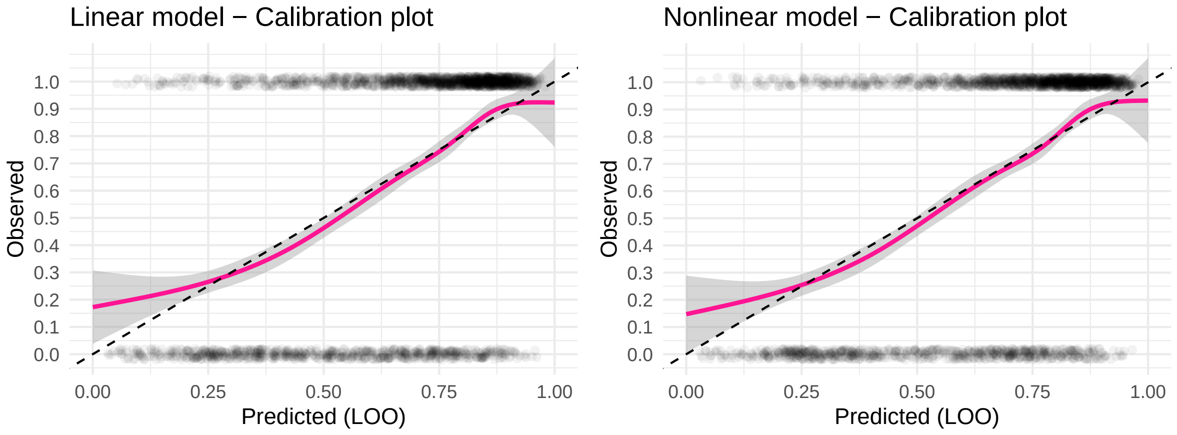


Figure 5: TBD X

5.4 Prior sensitivity analysis

```
#normal prior
normal_prior <- normal(location = 0, scale = NULL, autoscale = FALSE)
linear_model_normal_prior <- stan_glm(formula_linear, data = cath,
                family = binomial(link = "logit"),
                prior = normal_prior, prior_intercept = normal_prior,
                QR=TRUE, refresh=0)

#default prior
linear_model_default_prior <- stan_glm(formula_linear, data = cath,
                family = binomial(link = "logit"),
                QR=TRUE, refresh=0)

#cauchy prior
cauchy_prior <- cauchy(location = 0, scale = NULL, autoscale = FALSE)
linear_model_cauchy_prior <- stan_glm(formula_linear, data = cath,
                family = binomial(link = "logit"),
                prior = cauchy_prior, prior_intercept = cauchy_prior,
                QR=TRUE, refresh=0)

#normal prior, large scale N(0, 100)
normal_prior_sd <- normal(0, scale = 100, autoscale = FALSE)
linear_model_normal_sd <- stan_glm(formula_linear, data = cath,
                family = binomial(link = "logit"),
                prior = normal_prior_sd, prior_intercept = normal_prior_sd,
                QR=TRUE, refresh=0)
```

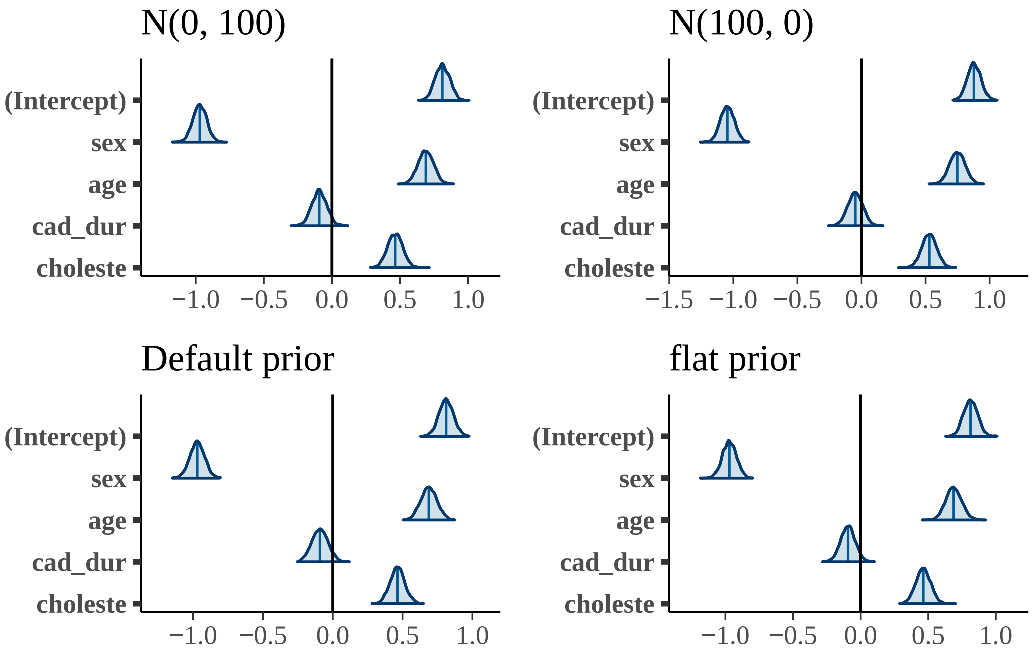
```
Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians
Running the chains for more iterations may help. See
https://mc-stan.org/misc/warnings.html#bulk-ess
```

```
#normal prior, large location
normal_prior_loc <- normal(100, scale = 2.5, autoscale = FALSE)
linear_model_normal_loc <- stan_glm(formula_linear, data = cath,
                family = binomial(link = "logit"),
                prior = normal_prior_loc, prior_intercept = normal_prior_loc,
                QR=TRUE, refresh=0)

#flat prior
flat_prior <- NULL
```

```
linear_model_flat_prior <- stan_glm(formula_linear, data = cath,
                family = binomial(link = "logit"),
                prior = flat_prior, prior_intercept = flat_prior,
                QR=TRUE, refresh=0)
```

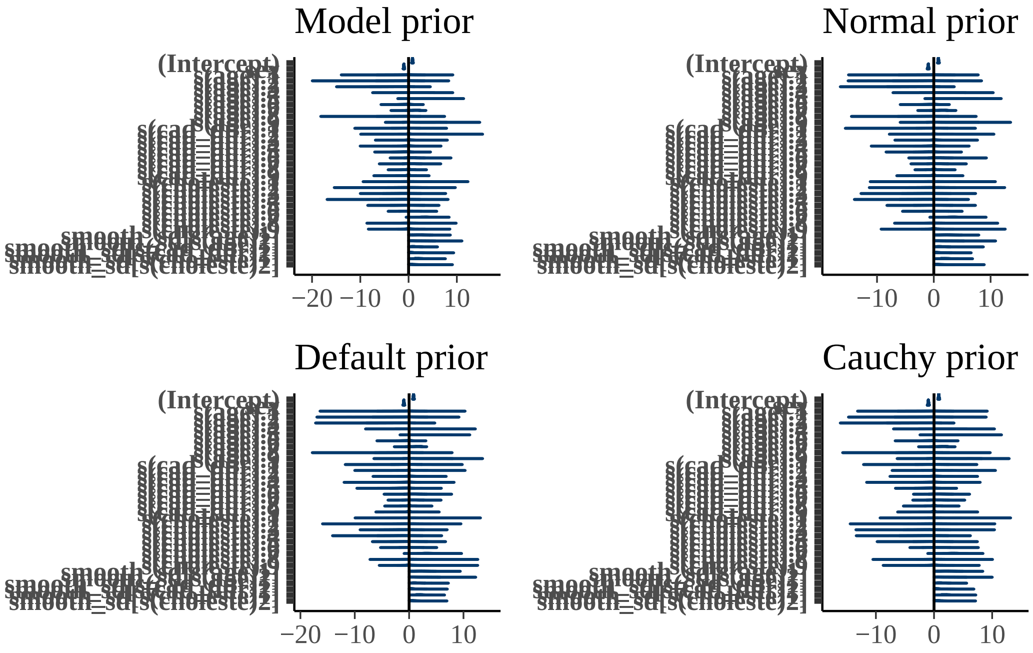
Plotting the posterior distribution for each prior:

```
pplot_normal <- plot(linear_model_normal_prior, "areas", prob = 0.95, prob_outer = 1) +
  geom_vline(xintercept = 0) + labs(title = "Normal prior")

pplot_default <- plot(linear_model_default_prior, "areas", prob = 0.95, prob_outer = 1) +
  geom_vline(xintercept = 0) + labs(title = "Default prior")

pplot_cauchy <- plot(linear_model_cauchy_prior, "areas", prob = 0.95, prob_outer = 1) +
  geom_vline(xintercept = 0) + labs(title = "Cauchy prior")

# pplot_linear <- pplot_linear + geom_vline(xintercept = 0) + labs(title = "Model prior")

pplot_normal_sd <- plot(linear_model_normal_sd, "areas", prob = 0.95, prob_outer = 1) +
  geom_vline(xintercept = 0) + labs(title = "N(0, 100)")

pplot_normal_loc <- plot(linear_model_normal_loc, "areas", prob = 0.95, prob_outer = 1) +
  geom_vline(xintercept = 0) + labs(title = "N(100, 0)")

pplot_flat <- plot(linear_model_flat_prior, "areas", prob = 0.95, prob_outer = 1) +
  geom_vline(xintercept = 0) + labs(title = "flat prior")

#put desired priors into here
grid.arrange(pplot_normal_sd, pplot_normal_loc, pplot_default, pplot_flat, ncol=2, nrow =
```



```
#normal prior
nonlinear_model_normal_prior <- stan_gamm4(
  formula_nonlinear, data = cath,
  family = binomial(link = "logit"),
  prior = normal_prior, prior_intercept = normal_prior,
  refresh = 0
)

#default prior
nonlinear_model_default_prior <- stan_gamm4(
  formula_nonlinear, data = cath,
  family = binomial(link = "logit"),
  refresh = 0
)

#cauchy prior
nonlinear_model_cauchy_prior <- stan_gamm4(
  formula_nonlinear, data = cath,
  family = binomial(link = "logit"),
  prior = cauchy_prior, prior_intercept = cauchy_prior,
  refresh = 0
)
```

```
nonlinear_pplot_normal <- plot(nonlinear_model_normal_prior, "areas", prob = 0.95, prob_ou
  geom_vline(xintercept = 0) + labs(title = "Normal prior")

nonlinear_pplot_default <- plot(nonlinear_model_default_prior, "areas", prob = 0.95, prob_
  geom_vline(xintercept = 0) + labs(title = "Default prior")

nonlinear_pplot_cauchy <- plot(nonlinear_model_cauchy_prior, "areas", prob = 0.95, prob_ou
  geom_vline(xintercept = 0) + labs(title = "Cauchy prior")

pplot_nonlinear <- plot(model_nonlinear, "areas", prob = 0.95, prob_outer = 1) +
geom_vline(xintercept = 0) + labs(title = "Model prior")

grid.arrange(pplot_nonlinear, nonlinear_pplot_normal, nonlinear_pplot_default, nonlinear_p
```



6 Discussion

- Problems, potential improvements?
- Conclusion of analysis

7 Lessons Learned

8 References