Homework 5 - Loss estimation

Luka Utješinović

Setup

Generate toy dataset

First, we prepare the generator for our toy binary classification data, which has 8 independent variables, 3 of which are unrelated to the target variable. Because we generate the data, we know all the properties of the data generating process, which will allow us to study the quality of our loss estimation. Negative log-loss is the loss function of choice throughout this homework

```
set.seed(0)

toy_data <- function(n) {
  x <- matrix(rnorm(8 * n), ncol = 8)
  z <- 0.4 * x[,1] - 0.5 * x[,2] + 1.75 * x[,3] - 0.2 * x[,4] + x[,5]
  y <- runif(n) > 1 / (1 + exp(-z))
  return (data.frame(x = x, y = y))
}

log_loss <- function(y, p) {
  -(y * log(p) + (1 - y) * log(1 - p))
}
```

A proxy for true risk

We'll be using this huge dataset as a proxy for the DGP and determining the ground-truth true risk of our models. Of course, this is itself just an estimate, but the dataset is large enough so that a model's risk on this dataset differs from its true risk at most on the 3rd decimal digit. Indeed, empirical risk for model h and the loss function of choice l is just a **Monte-Carlo** estimation of the true risk:

$$Risk_{MC}(h) = \frac{1}{n}\sum_{i=1}^{n} l(y_i, h(x_i)) \approx Risk(h)$$

From the **Central Limit Theorem**, it follows that for sufficiently large n, the error of empirical risk as an estimator of the true risk is close to $\frac{SD(l(y,h(x))}{\sqrt{n}}$. For this dataset ($n = 1e5$, and seed set to 0) this quantity - **standard error** is approximately equal to 0.003, which means that the error is at most $1e-3$.

```
n <- 100000
dataset <- toy_data(n)
preds <- runif(n)
losses <- log_loss(dataset$y, preds)

mu <- mean(losses)
s <- sd(losses) / sqrt(n)

print(paste0("Empirical  risk: ", round(mu, digits = 4)))
```

```
## [1] "Empirical  risk: 1.0035"
```

```
print(paste0("Standard error: ", round(s, digits = 4)))
```

```
## [1] "Standard error: 0.0032"
```

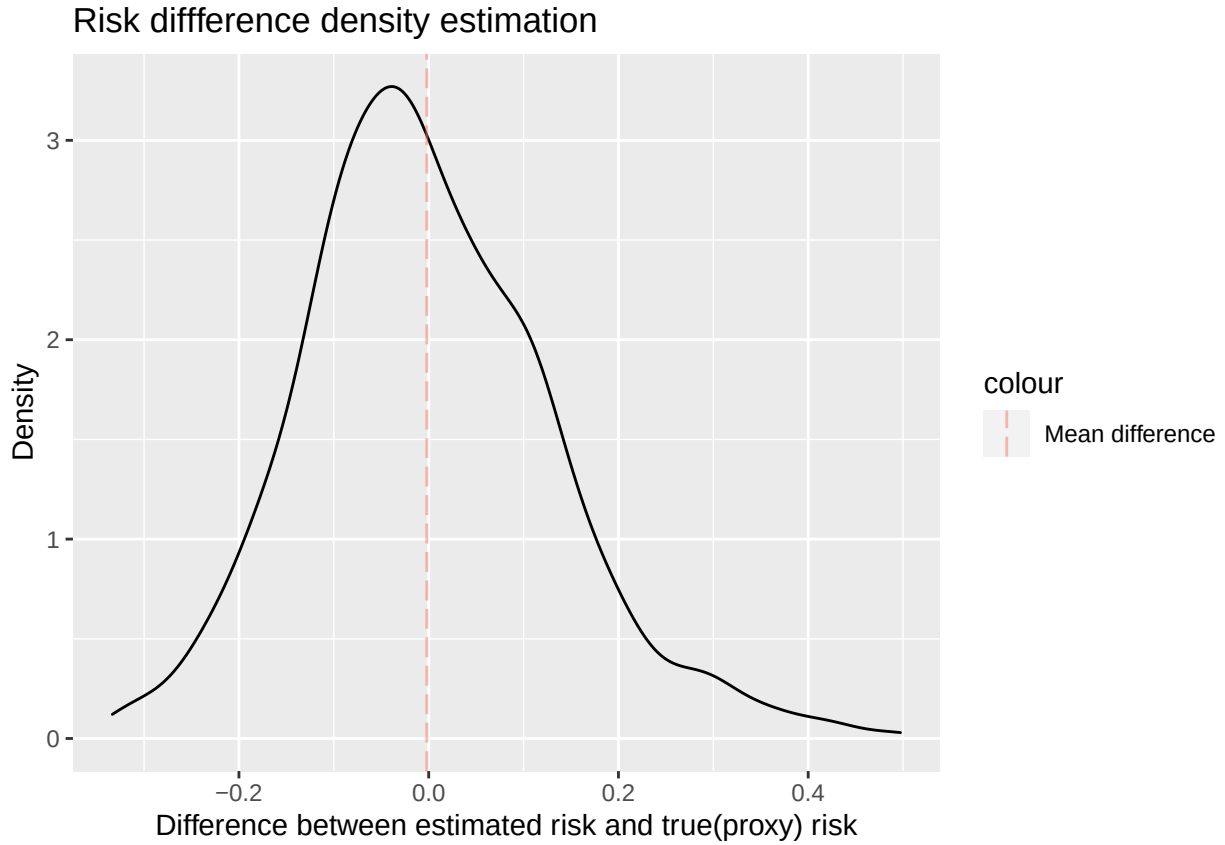
Holdout estimation

Holdout estimation or, as it is commonly referred to, **train-test splitting**, is the most common approach to estimating a model's risk. The first and most important thing to understand is that the model's risk on the test data is just an estimate of the model's true risk. As such, it should always contain some sort of quantification of uncertainty. The simplest method for quantifying the uncertainty constructs **confidence intervals** with the underlying **asymptotic normality argument** - the mean loss is normally distributed with n times lower than the variance of the loss.

For this homework, **logistic regression** will be the model of interest, and we will compare different risk estimation methods, starting with holdout estimation. For all risk estimation methods, DGP will be the same as in the previous section. First we generate a dataset of size 50, and using a logistic regression learner we obtain a model h. Then we generate a dataset of size 100000 to obtain a proxy for the true risk of h. Afterwards the following procedure is repeated 1000 times:

1. Dataset of size 50 is generated
2. Loss estimation is computed using this dataset and h
3. 95% confidence interval for the true loss is computed, and the coverage is saved
4. Standard error of losses and the difference between true(proxy) risk - **bias** is saved

Estimated density of loss difference, along with other quantities of interest is shown in the figure below:




```
## [1] "True risk proxy: 0.5772"
```

```
## [1] "Mean difference: -0.002"
```

```
## [1] "Coin flip baseline true risk proxy: 0.6931"
```

```
## [1] "Standard errors median: 0.1242"
```

```
## [1] "Coverage of 95% CIs: 90.2%"
```

The first thing to note is that the mean difference is distributed around 0, while the whole distribution is skewed to the left. From here we conclude that holdout estimation is an **unbiased** estimate of the true risk. Coverage of confidence intervals is slightly lower than expected. This means that asymptotic normality argument slightly underestimates the uncertainty of holdout estimation, and it may be appropriate to use more sophisticated techniques. We also notice that the right tail of the estimated density is heavier. This might be attributed to the nature of log-loss - function $-ln(x)$ around 0 has very high values.

Effects of changing training/test size parameters - Because holdout estimation is also consistent (follows from the law of large numbers), we would expect that as we increase the size of the test set (or the number of iterations), the whole distribution would be squished to 0 (differences would be smaller), while the differences would be bigger if we took a smaller test set (or decrease the number of iterations). As for the size of the training set, an increase should lead to decrease in the true (proxy) risk. This also depends on the choice of the model. We expect that logistic regression is a **smart learner** - true risk does not increase as we increase the size of the training set.

To conclude, holdout estimation has nice theoretical properties (unbiasedness and consistency), and as a consequence of that it should be the preferred method of risk estimation as long as we have enough data (which is sadly not often the case in practice).

Overestimation of the deployed model's risk

In practice we rarely deploy the model trained only on the training data. Similarly, we never deploy any of the k models that are learned during k-fold cross-validation. Instead, we use the estimation as a means to select the best learner and then deploy the model trained on more data, typically all the available data. More data typically means that the learner will produce a model with lower true risk. To illustrate this we execute the following procedure:

1. We generate two datasets, each with 50 observations
2. We train model h_1 using a learner on the first dataset only
3. We train model h_2 using the same learning algorithm using both datasets
4. We compute the true(proxy) risk of models h_1, h_2 using the huge dataset
5. We repeat steps (1-4) 50 times

We report the R summary of previous results:

```
## [1] "Summary of risk_h1 - risk_h2"
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.009582  0.064972  0.143439  0.816870  0.257212 11.163528
```

As expected, the difference in estimated risk is mostly positive - h_2 mostly performs better than h_1. We can see that this is not always the case - the minimum difference is negative. However, it is still very small, especially relative to the maximum difference. This might also be caused by training convergence issues - training sets for h_1 and h_2 are not that big, especially compared to the proxy dataset of size 100000. Since h_1 is trained using only 50 data points, it is reasonable to assume that risk estimation for h_1 will be much more variable. Indeed we can confirm this from previous results.

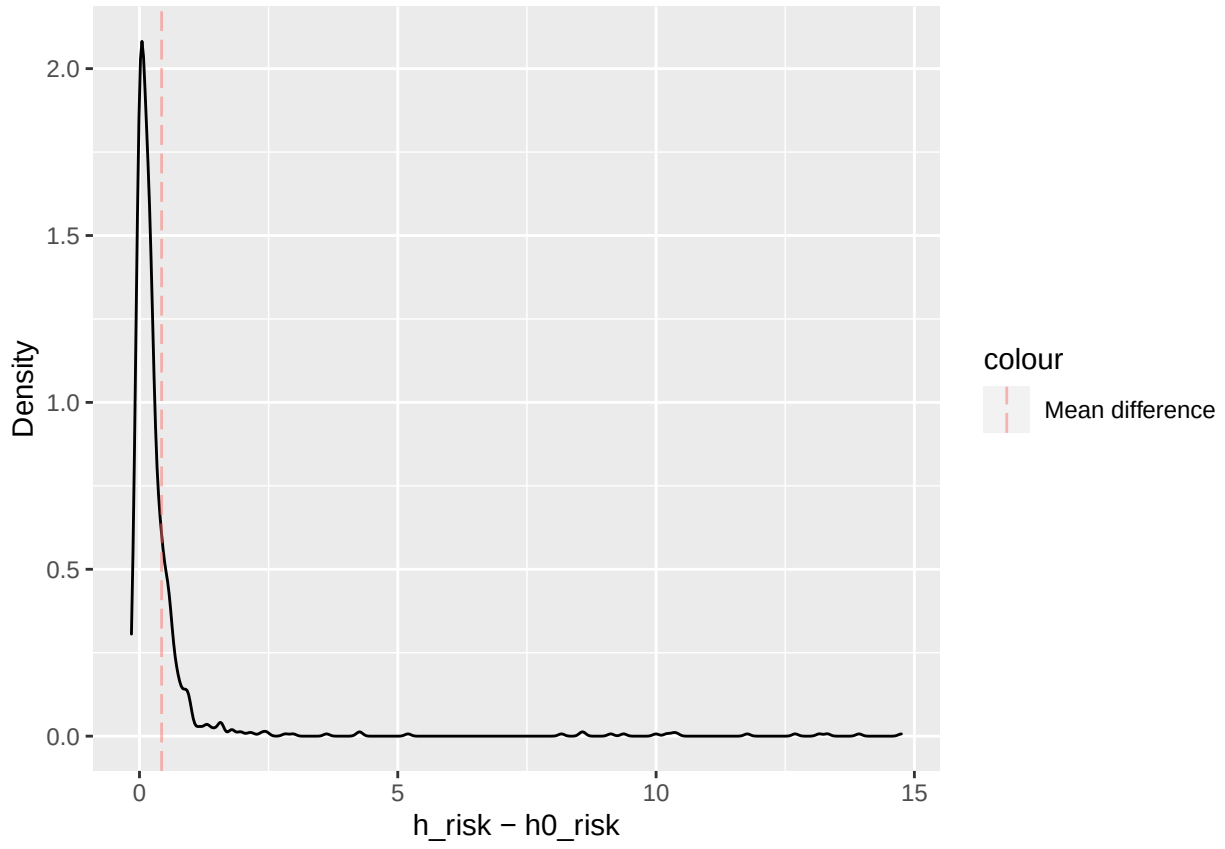
Effects of changing training size paramemters - If we increase sizes of training sets while still preserving the relation $|D_{train}(h_2)| = 2|D_{train}(h_1)|$, we expect that the fraction of cases where h_2 performs better than h_1 increases since risk estimation is less variable for both models in this case. Conversely by reducing training sets, variance of risk estimation increases and the fraction of cases where h_2 performs better than h_1 becomes much more unstable.

Loss estimator variability due to split variability

In practical applications of train-test splitting, we would choose a train-test split proportion, train on the training data, and test on the test data. We would then use this result on the test data as an estimate of the true risk of the model trained on all data. From the experiments so far, we can gather that this estimate will be biased, because the tested model is trained on less data than the model we are interested in. It will also have variance due to which observations ended up in the training set and which in the test set. To that we can add the most basic source of variability - the variability of the losses across observations. To illustrate this we execute the following procedure:

1. We generate a toy dataset with 100 observations
2. We train model h_0 using logistic regression on all 100 observations and compute its true risk proxy using the huge dataset.
3. We split the dataset into 50 training and 50 test observations at random
4. We train model h on the training set and compute the its risk on the test set as an estimator of h_0's true risk. We also record if the 95% CI of this estimate contains the true risk of h_0
5. We repeat steps (3-4) 1000 times

On the following figure, we show the density estimation of risk differences, along with other quantities of interest:



```
## [1] "True risk proxy: 0.4904"

## [1] "Mean difference: 0.4287"

## [1] "Standard errors median: 0.1622"

## [1] "Coverage of 95% CIs: 84%"
```

We can confirm our suspicions from the beginning of this section. First, this risk estimation technique is biased - it more often than not overestimates the true risk (**positive bias**). Second, this method is highly variable and depends largely on data points that end up in training/test sets. Furthermore, we can see that extreme difference values can occur, which can lead to wrong model selection in practice. 95% CI coverage is significantly lower, which means that the asymptotic normality argument vastly underestimates the uncertainty of this method.

Effects of changing dataset size and train/test proportion size - With a fixed proportion size for train/test splits, increasing the size of the dataset would lower the true risk of h_0 (because we have more data), which would in turn lower the proxy risk. For h_1 we are also using more data, and since our proportions remain fixed, this leads to increase in both train and test sets, which in turn gives a better estimate of the true risk. Even though this estimate becomes less biased, it is still positively biased. The variance problem is still present - the estimate depends on which data points end up in training/test sets, although it could be lower if we dramatically increase the size of the dataset. Opposite holds true if we decrease the size of the dataset.

Now let us investigate what happens if the train/test proportions are changed. If we increase the size of the train set, more data will be available during training which would again lead to **lowering the bias** of our risk estimate. However, this means that less data points are present in the test set, which means that it is much more affected by the data points that do end up in the test set - **high variance**. The opposite holds true if we decrease the size of the training set. Less data will be used for training - **higher bias**, and more data will be present for testing - **lower variance**.

To conclude, even though train-test splitting is commonly encountered in practice, it is a flawed method, and for small/medium sized datasets we should definitely use more sophisticated techniques like **cross-validation** which we will discuss during the next section.

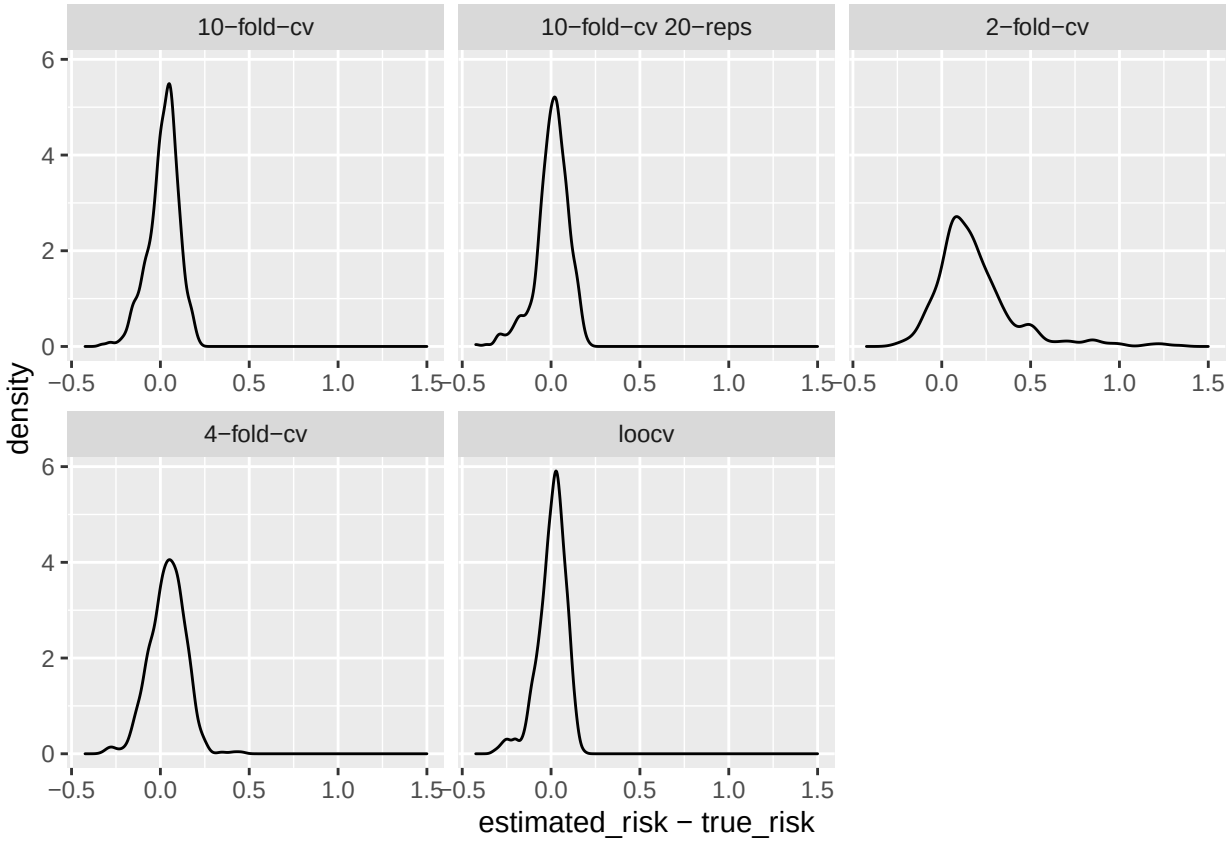
Cross-validation

If we extrapolate the results so far to cross-validation, we can conclude that cross-validation estimates of true risk will also be biased and will contain a lot of variability if the dataset is relatively small. This variability will be both due to training set and due to test set variability on top of the inherent variability of the losses.

Let's explore this by also finally incorporating the variability caused by the draw of the dataset from the DGP, which we have up to this point kept fixed. We will conduct the following experiment:

1. We generate a toy dataset with 100 observations
2. We train model h_0 using logistic regression on the generated dataset and compute the true risk proxy using the huge dataset
3. We estimate the risk of h_0 with the following estimators: 2-fold cross-validation, leave one out cross-validation - **LOOCV**, 10-fold cross validation, 4-fold cross validation, 10-fold cross validation repeated 20 times (for each observation the loss is computed as the mean of losses over 20 repetitions)
4. For each of the estimators, we compute the difference between computed risk and true risk proxy, and we record the coverage of 95% CI
5. Steps (1-4) are repeated 500 times

On the following figure, we show the resulting density estimates of loss differences for every mentioned cross-validation method, along with other statistics of interest:




```
## [1] "2 fold cross-validation"
```

```
## [1] "Mean difference: 0.3881"
```

```
## [1] "Standard error median: 0.1101"

## [1] "95% CI coverage: 68.2"

## [1] "4 fold cross-validation"

## [1] "Mean difference: 0.0485"

## [1] "Standard error median: 0.0829"

## [1] "95% CI coverage: 89.4"

## [1] "10 fold cross-validation"

## [1] "Mean difference: 0.0171"

## [1] "Standard error median: 0.0771"

## [1] "95% CI coverage: 91.8"

## [1] "10 fold cross-validation 20 repetitions"

## [1] "Mean difference: 0.0053"

## [1] "Standard error median: 0.0767"

## [1] "95% CI coverage: 89.2"

## [1] "LOOCV"

## [1] "Mean difference: 0.0038"

## [1] "Standard error median: 0.074"

## [1] "95% CI coverage: 93.4"
```

First thing to observe is that as we increase k, the mean difference is lowered, along with standard errors. Lower mean difference (bias) is a consequence of the fact that as we increase k, we also increase the number of training examples for each cross-validation model, and they are more similar to the model trained on the entire dataset. Furthermore, cross-validation models are more similar to each other with increase in k (and for 'smart' learners better trained), which also lowers the variance of the losses, and in turn lowers the standard error. Coverage is still below 95%, which means that we are still underestimating the variability of our estimators. The best coverage is achieved with LOOCV ($\approx 93.4\%$).

For low values of k (2 and 4 in particular), we notice the trend of heavy right density tails. This can be harmful in practice because it could, with higher chance than other cross-validation methods, lead us to selecting the wrong model for our problem. As we increase k we also get sharper peaks around 0, which means that our estimates are getting closer to the true value. Highest mode is achieved by LOOCV again.

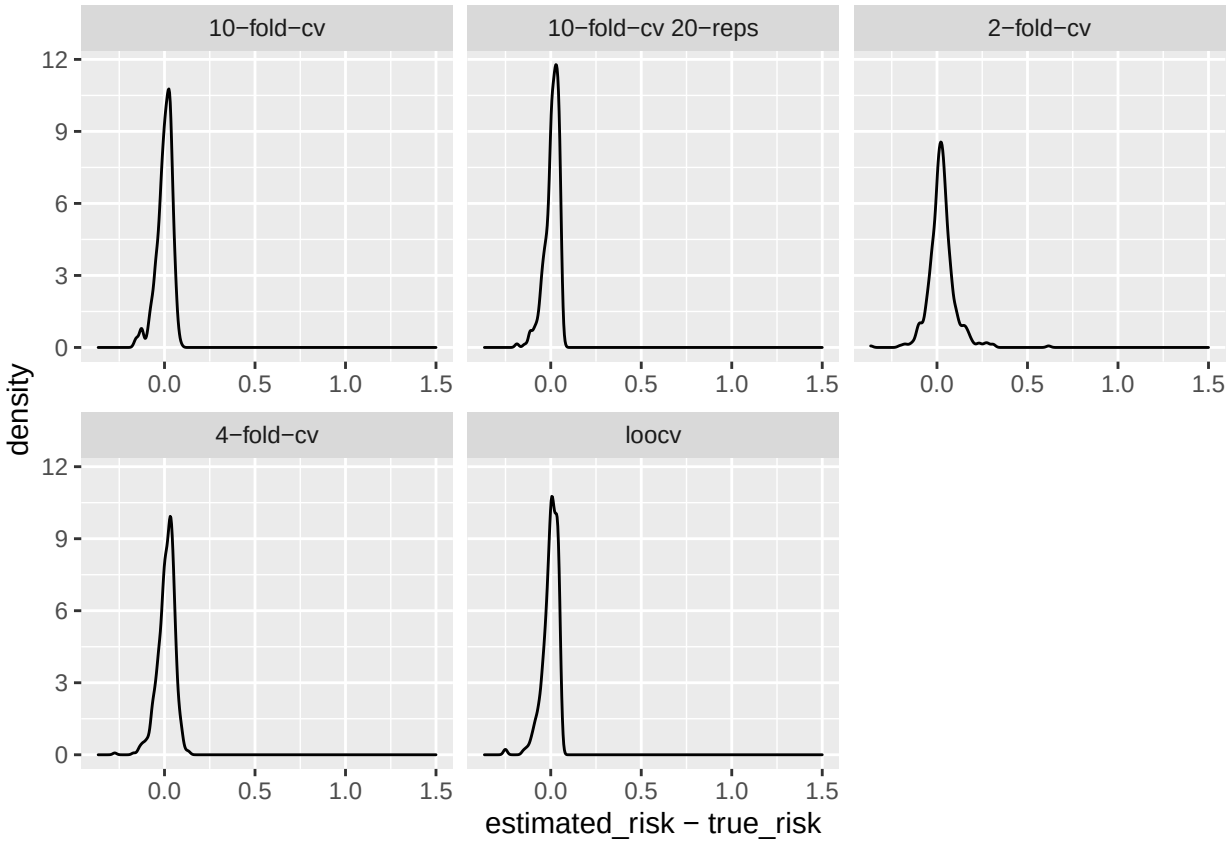
From everything that was said, we conclude that LOOCV is the best cross-validation method for this scenario, and it should be the preferred method of choice as long as the computational overhead is feasible. Also, 10 fold cross-validation and repeated 10-fold cross validation densities are not drastically different, while the computational overhead for repeated 10-fold cross validation is significantly higher.

Different scenario

We tried to construct a different DGP for which the results would disagree with our latest experiment. First we reduced the number of columns to 2, and sampled from Poisson distribution with $\lambda = 1$ instead. We also created a non-linear relationship between the response and the first feature (second feature effectively represents noise) in order to force logistic regression to unerperform.

```
different_dgp <- function(n) {
  x <- matrix(rpois(2 * n, 1), ncol = 2)
  y <- (-1) ^ x[,1] + x[,1] %% 2
  return (data.frame(x = x, y = y))
}
```

Results of the experiment with this DGP are presented on the following figure:




```
## [1] "2 fold cross-validation"

## [1] "Mean difference: 0.0262"

## [1] "Standard error median: 0.0403"

## [1] "95% CI coverage: 81.4"

## [1] "4 fold cross-validation"

## [1] "Mean difference: 0.0093"

## [1] "Standard error median: 0.0366"

## [1] "95% CI coverage: 87"

## [1] "10 fold cross-validation"

## [1] "Mean difference: -9e-04"
```

```
## [1] "Standard error median: 0.036"

## [1] "95% CI coverage: 89.8"

## [1] "10 fold cross-validation 20 repetitions"

## [1] "Mean difference: 0.0059"

## [1] "Standard error median: 0.0349"

## [1] "95% CI coverage: 86.8"

## [1] "LOOCV"

## [1] "Mean difference: -0.0027"

## [1] "Standard error median: 0.0361"

## [1] "95% CI coverage: 89.2"
```

LOOCV seems to be negatively biased now. With 10 fold cross-validation mean is closer to 0 than the mean of LOOCV, with practically the same variance, while the computational overhead of 10 fold cross-validation is drastically lower!