Homework 5

Student: Matej Kalc, 63180368

Load libraries

```
set.seed(0)
library(stats)
library(ggplot2)
library(dplyr)
```

Data generating process

```
toy_data <- function(n, seed = NULL) {
  set.seed(seed)
  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))
}

df_huge <- toy_data(100000, 0)
```

A proxy for true risk

We can use the VC theorem. Let n be 100000. The number of VC dimensions for Logistic regression is one more then the number of features. Let vc_dim be 100001. By choosing a small delta, we can be assured that the error will be smaller then 0.001.

Model loss estimator variability due to test data variability

```
n <- 50
df_dgp <- toy_data(n, 0)
glm.fit <- glm(y ~ x.1 + x.2 + x.3+ x.4 + x.5+ x.6 + x.7+ x.8, data=df_dgp, family="binomial")
```

Evaluate the huge data set using the trained model.

```
y <- df_huge$y
X <- df_huge[, seq(1, 8)]
glm.probs <- predict(glm.fit, newdata = X, type = "response")
```

Now let's calculate the true risk proxy.

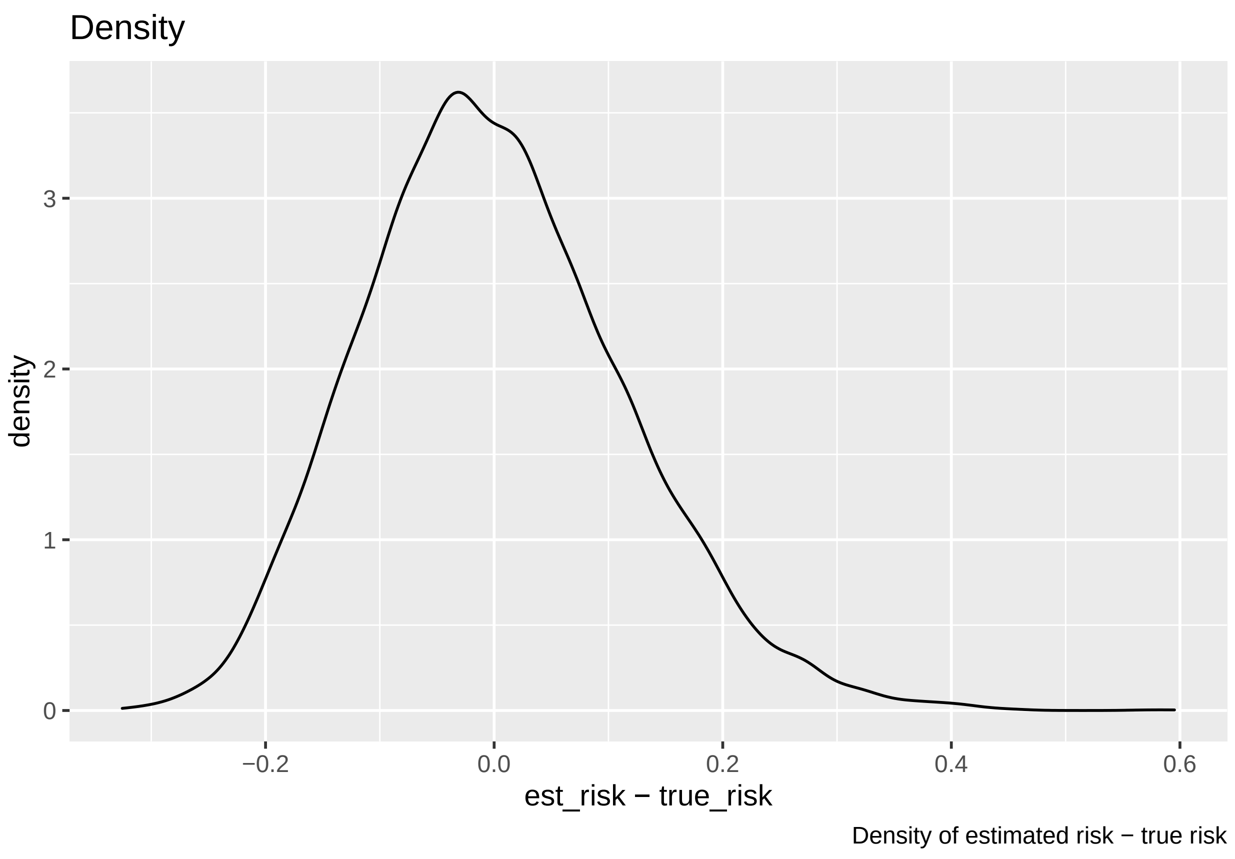
```
true_risk <- mean(log_loss(y, glm.probs))
true_risk_5050 <- mean(log_loss(y, 0.5))
```

The reported true risk is 0.5755. Now we will evaluate the standard error of model h.

```
est_risk <- vector(mode="list", length = 10000)
est_se <- vector(mode="list", length = 10000)
contains95CI <- vector(mode="list", length = 10000)
for(j in 1:10000) {
```

```r
  n <- 50
  df_dgp <- toy_data(n)
  y <- df_dgp$y
  X <- df_dgp[, seq(1, 8)]
  glm.probs <- predict(glm.fit, newdata = X, type = "response")
  losses <- log_loss(y, glm.probs)
  est_risk[j] <- mean(as.numeric(losses))
  est_se[j] <- sd(as.numeric(unlist(losses)))/sqrt(length(losses))

  upper_bound_CI <- as.numeric(est_risk[j]) + 1.96 * as.numeric(est_se[j])
  lower_bound_CI <- as.numeric(est_risk[j]) - 1.96 * as.numeric(est_se[j])
  contains95CI[j] <- true_risk < upper_bound_CI && true_risk > lower_bound_CI
}
```

```r
differences_between_true_and_est_risk <- as.numeric(est_risk) - true_risk
df_dens <- data.frame(differences_between_true_and_est_risk)
colnames(df_dens) <- c("x")
ggplot(data = df_dens, aes(x = x)) +
  geom_density() + labs(title = "Density",
            caption = "Density of estimated risk - true risk") +
   xlab("est_risk - true_risk")
```



```r
sprintf("True risk proxy: %f", as.numeric(true_risk))
```

```
## [1] "True risk proxy: 0.575545"
```

```
sprintf("Mean difference: %f", mean(differences_between_true_and_est_risk))
```

```
## [1] "Mean difference: 0.000534"
```

```
sprintf("0.5-0.5 baseline true risk: %f", true_risk_5050)
```

```
## [1] "0.5-0.5 baseline true risk: 0.693147"
```

```
sprintf("Median standard error: %f", median(as.numeric(est_se)))
```

```
## [1] "Median standard error: 0.109998"
```

```
sprintf("Percentage of 95CI that contain the true risk proxy: %f", mean(as.numeric(contains95CI)))
```

```
## [1] "Percentage of 95CI that contain the true risk proxy: 0.921800"
```

We see that both the mean difference and median standard deviation are small. We managed to estimated the true risk by resampling from the data generating process. Usually in practice we are not capable of resampling again and again. To achieve a better risk estimate we could use cross validation. If we compare our estimate to the baseline, we see that our estimate is much better then the baseline. The 95CI of our estimates contained 93% of the time the true risk. If the training set would increase in size the mean difference and median standard error would get smaller and the percentage of 95CI would increase. If we increase the test set, then the variance will reduce.

Overestimation of the deployed model's risk

```
n <- 50

y <- df_huge$y
X <- df_huge[, seq(1, 8)]

diff_risk1_risk2 <- vector(mode="list", length = 50)

for(j in 1:50) {
  df_dgp1 <- toy_data(n)
  glm.fit1 <- glm(y ~ x.1 + x.2 + x.3+ x.4 + x.5+ x.6 + x.7+ x.8, data=df_dgp1, family="binomial")
  df_dgp2 <- rbind(df_dgp1, toy_data(n))
  glm.fit2 <- glm(y ~ x.1 + x.2 + x.3+ x.4 + x.5+ x.6 + x.7+ x.8, data=df_dgp2, family="binomial")

  glm.probs1 <- predict(glm.fit1, newdata = X, type = "response")
  glm.probs2 <- predict(glm.fit2, newdata = X, type = "response")

  true_risk_1 <- mean(as.numeric(log_loss(y, glm.probs1)))
  true_risk_2 <- mean(as.numeric(log_loss(y, glm.probs2)))

  diff_risk1_risk2[j] <- true_risk_1 - true_risk_2
}

summary(as.numeric(diff_risk1_risk2))
```

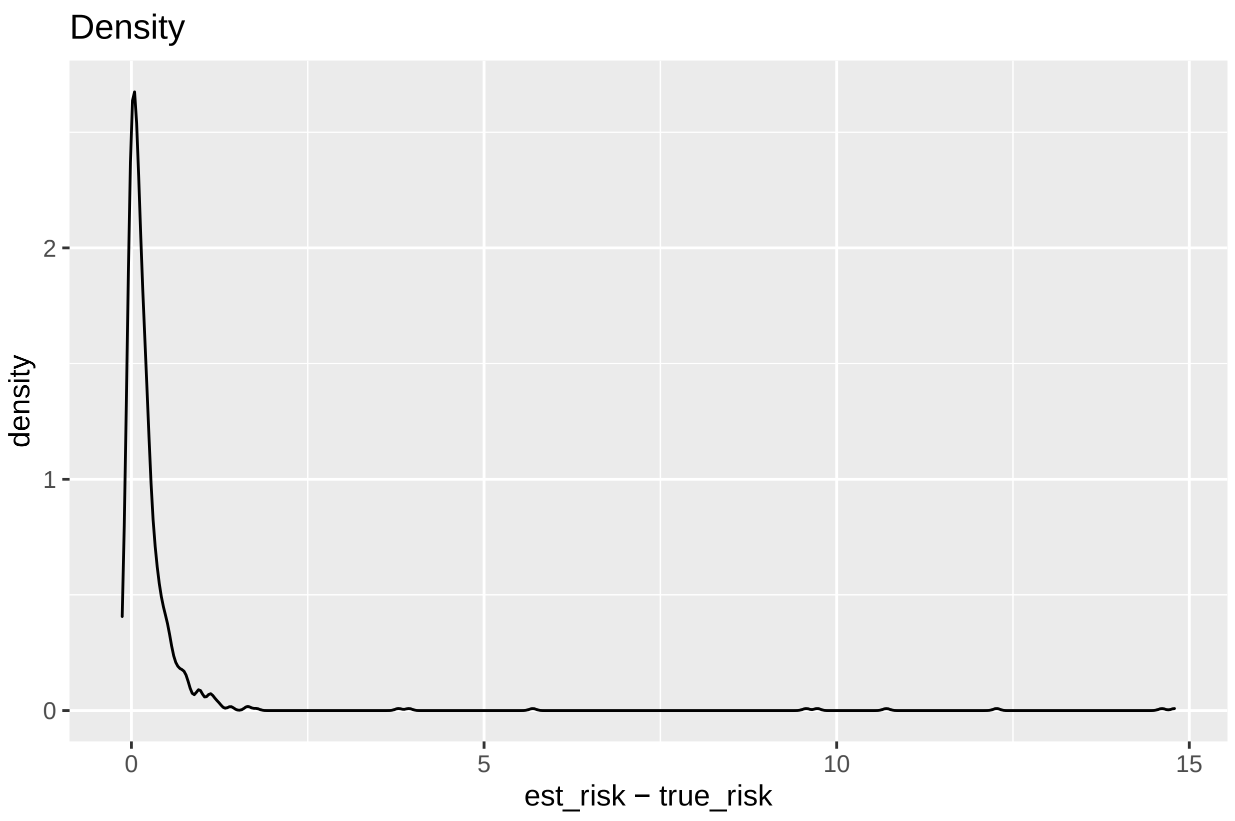
```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.04518  0.02992  0.07885  0.13214  0.17372  0.65435
```

We observe that using additional 50 data points we reduced our log loss on average. If we increase the data sets the differences and variance will start to get smaller.

Loss estimator variability due to split variability

```r
n <- 100
y <- df_huge$y
X <- df_huge[, seq(1, 8)]
df_dgp <- toy_data(n, 0)
glm.fit <- glm(y ~ x.1 + x.2 + x.3+ x.4 + x.5+ x.6 + x.7+ x.8, data=df_dgp, family="binomial")

glm.probs <- predict(glm.fit, newdata = X, type = "response")

true_risk <- mean(as.numeric(log_loss(y, glm.probs)))

est_risk <- vector(mode="list", length = 1000)
est_se <- vector(mode="list", length = 1000)
contains95CI <- vector(mode="list", length = 1000)
for(j in 1:1000) {
  split_dummy <- sample(c(rep(0, 0.5 * nrow(df_dgp)), rep(1, 0.5 * nrow(df_dgp))))
  data_train <- df_dgp[split_dummy == 0, ]
  data_test <- df_dgp[split_dummy == 1, ]

  glm.fit_h <- glm(y ~ x.1 + x.2 + x.3+ x.4 + x.5+ x.6 + x.7+ x.8, data=data_train, family="binomial")

  y <- data_test$y
  X <- data_test[, seq(1, 8)]
  glm.probs <- predict(glm.fit_h, newdata = X, type = "response")
  losses <- log_loss(y, glm.probs)
  est_risk[j] <- mean(as.numeric(losses))
  est_se[j] <- sd(as.numeric(unlist(losses)))/sqrt(length(losses))

  upper_bound_CI <- as.numeric(est_risk[j]) + 1.96 * as.numeric(est_se[j])
  lower_bound_CI <- as.numeric(est_risk[j]) - 1.96 * as.numeric(est_se[j])
  contains95CI[j] <- true_risk < upper_bound_CI && true_risk > lower_bound_CI
}
```

```r
differences_between_true_and_est_risk <- as.numeric(est_risk) - as.numeric(true_risk)
df_dens <- data.frame(differences_between_true_and_est_risk)
colnames(df_dens) <- c("x")
plot(ggplot(data = df_dens, aes(x = x)) +
  geom_density() + labs(title = "Density",
             caption = "Density of estimated risk - true risk") +
   xlab("est_risk - true_risk"))
```



Density of estimated risk − true risk

```r
sprintf("True risk proxy: %f", as.numeric(true_risk))
```

```
## [1] "True risk proxy: 0.525473"
```

```r
sprintf("Mean difference: %f", mean(differences_between_true_and_est_risk))
```

```
## [1] "Mean difference: 0.271040"
```

```r
sprintf("Median standard error: %f", median(as.numeric(est_se)))
```

```
## [1] "Median standard error: 0.128184"
```

```r
sprintf("Percentage of 95CI that contain the true risk proxy: %f", mean(as.numeric(contains95CI)))
```

```
## [1] "Percentage of 95CI that contain the true risk proxy: 0.832000"
```

The mean difference is quite learger than the mean difference in the first exercise. The estimated risks are negatively biased, but in this case we did not resample from the data generating process. Also the median standard error is quite large, which could be reduced using a bigger test set. This approach is not reliable for estimating the risk of a model. If the data set was larger the mean difference would be smaller. With a smaller training proportion the mean difference would increase (higher bias) and the 95CI would contain less times the true risk. If we would increase the training proportion the standard deviation would increase.

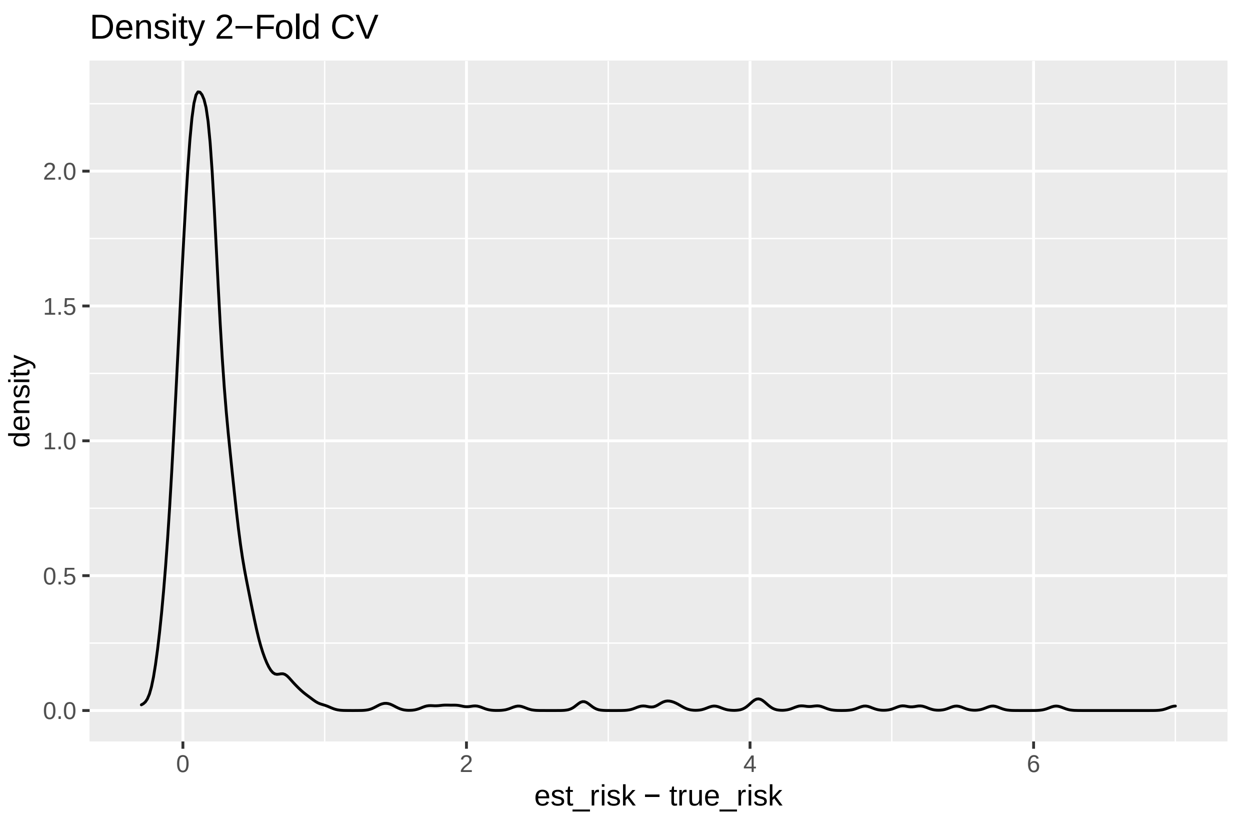
Cross-validation

```r
k_fold_cv <- function(df, k) {
  losses <- vector(mode="list", length = nrow(df))

  indexes <- c()
  for(j in 1:k){
    indexes <- c(indexes, rep(j, nrow(df)/k))
  }
  split_dummy <- sample(indexes)

  for(i in 1:k) {
    test_fold <- df[split_dummy == i, ]
    train_folds <- df[split_dummy != i, ]

    glm.fit <- glm(y ~ x.1 + x.2 + x.3+ x.4 + x.5+ x.6 + x.7+ x.8, data=train_folds, family="binomial")
    y <- test_fold$y
    X <- test_fold[, seq(1, 8)]
    glm.probs <- predict(glm.fit, newdata = X, type = "response")


    losses[split_dummy == i] <- as.numeric(log_loss(y, glm.probs))
  }
  as.numeric(losses)
}

k_fold_cv_precalc <- function(df, k) {
  losses <- k_fold_cv(df, k)
  results <- c(risk = mean(as.numeric(losses)), se = sd(as.numeric(unlist(losses)))/sqrt(length(losses)
  return(results)
}
```

```
n <- 100
number_of_iterations <- 500
est_2_fold_risk <- vector(mode="list", length = number_of_iterations)
se_2_fold_risk <- vector(mode="list", length = number_of_iterations)
contains95CI_2_fold <- vector(mode="list", length = number_of_iterations)
est_4_fold_risk <- vector(mode="list", length = number_of_iterations)
se_4_fold_risk <- vector(mode="list", length = number_of_iterations)
contains95CI_4_fold <- vector(mode="list", length = number_of_iterations)
est_10_fold_risk <- vector(mode="list", length = number_of_iterations)
se_10_fold_risk <- vector(mode="list", length = number_of_iterations)
contains95CI_10_fold <- vector(mode="list", length = number_of_iterations)
est_LOOCV_risk <- vector(mode="list", length = number_of_iterations)
se_LOOCV_fold_risk <- vector(mode="list", length = number_of_iterations)
contains95CI_LOOCV_fold <- vector(mode="list", length = number_of_iterations)
est_rep_CV_risk <- vector(mode="list", length = number_of_iterations)
se_rep_CV_fold_risk <- vector(mode="list", length = number_of_iterations)
contains95CI_rep_CV_fold <- vector(mode="list", length = number_of_iterations)

for(iter in 1:number_of_iterations) {
  df_dgp <- toy_data(n)
  y <- df_huge$y
  X <- df_huge[, seq(1, 8)]
  glm.fit <- glm(y ~ x.1 + x.2 + x.3+ x.4 + x.5+ x.6 + x.7+ x.8, data=df_dgp, family="binomial")
  glm.probs <- predict(glm.fit, newdata = X, type = "response")

  true_risk <- mean(as.numeric(log_loss(y, glm.probs)))

  rez <- k_fold_cv_precalc(df_dgp, 2)
```

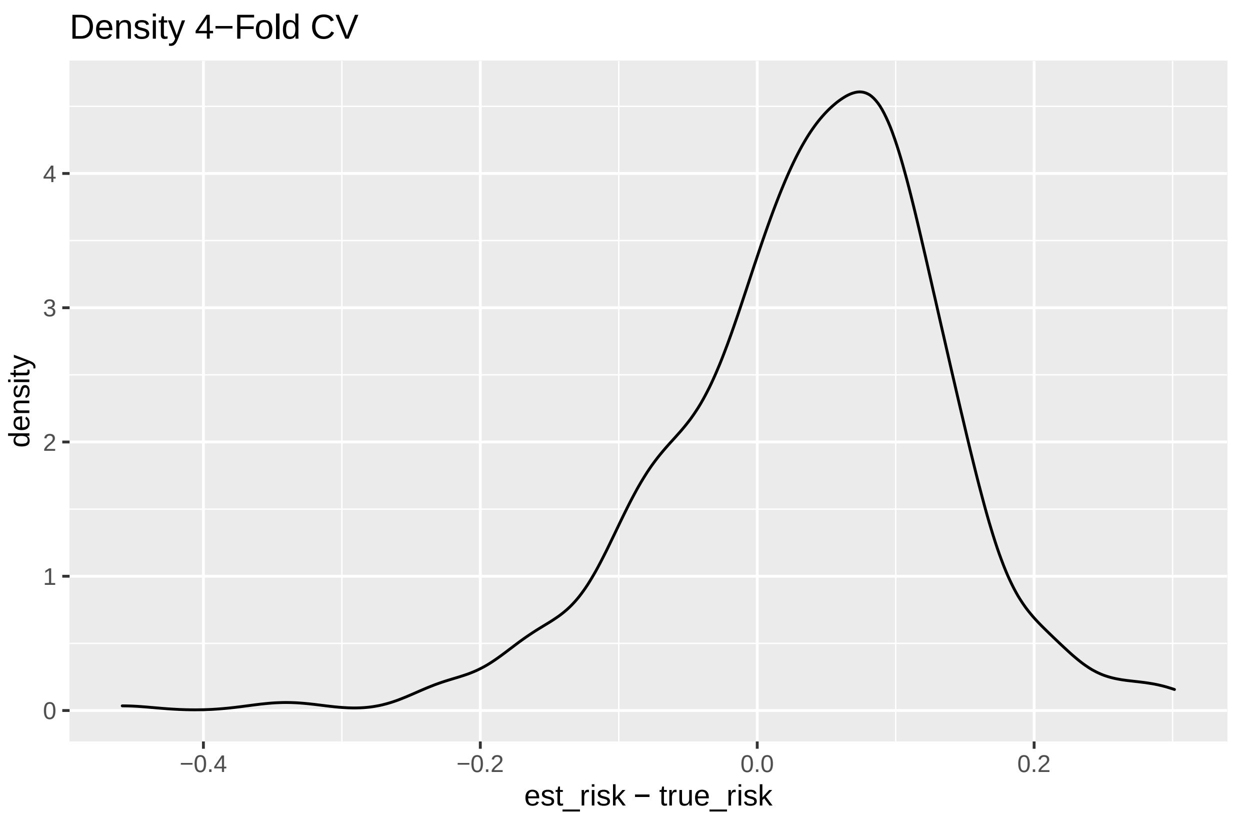
```r
  est_2_fold_risk[iter] <- rez["risk"] - true_risk
  upper_bound_CI <- as.numeric(rez["risk"]) + 1.96 * as.numeric(rez["se"])
  lower_bound_CI <- as.numeric(rez["risk"]) - 1.96 * as.numeric(rez["se"])
  contains95CI_2_fold[iter] <- true_risk < upper_bound_CI && true_risk > lower_bound_CI
  se_2_fold_risk[iter] <- rez["se"]


  rez <- k_fold_cv_precalc(df_dgp, nrow(df_dgp))
  est_LOOCV_risk[iter] <- rez["risk"] - true_risk
  upper_bound_CI <- as.numeric(rez["risk"]) + 1.96 * as.numeric(rez["se"])
  lower_bound_CI <- as.numeric(rez["risk"]) - 1.96 * as.numeric(rez["se"])
  contains95CI_LOOCV_fold[iter] <- true_risk < upper_bound_CI && true_risk > lower_bound_CI
  se_LOOCV_fold_risk[iter] <- rez["se"]

  rez <- k_fold_cv_precalc(df_dgp, 10)
  est_10_fold_risk[iter] <- rez["risk"] - true_risk
  upper_bound_CI <- as.numeric(rez["risk"]) + 1.96 * as.numeric(rez["se"])
  lower_bound_CI <- as.numeric(rez["risk"]) - 1.96 * as.numeric(rez["se"])
  contains95CI_10_fold[iter] <- true_risk < upper_bound_CI && true_risk > lower_bound_CI
  se_10_fold_risk[iter] <- rez["se"]

  rez <- k_fold_cv_precalc(df_dgp, 4)
  est_4_fold_risk[iter] <- rez["risk"] - true_risk
  upper_bound_CI <- as.numeric(rez["risk"]) + 1.96 * as.numeric(rez["se"])
  lower_bound_CI <- as.numeric(rez["risk"]) - 1.96 * as.numeric(rez["se"])
  contains95CI_4_fold[iter] <- true_risk < upper_bound_CI && true_risk > lower_bound_CI
  se_4_fold_risk[iter] <- rez["se"]

  rep_losses <- matrix(nrow = 20, ncol = nrow(df_dgp))
  for(l in 1:20) {
    rep_losses[l,] <- k_fold_cv(df_dgp, 10)
  }
  rep_losses_avg <- colMeans(rep_losses)

  est_rep_CV_risk[iter] <- mean(rep_losses_avg) - true_risk
  se <- sd(as.numeric(unlist(rep_losses_avg)))/sqrt(length(rep_losses_avg))
  upper_bound_CI <- as.numeric(mean(rep_losses_avg)) + 1.96 * as.numeric(se)
  lower_bound_CI <- as.numeric(mean(rep_losses_avg)) - 1.96 * as.numeric(se)
  contains95CI_rep_CV_fold[iter] <- true_risk < upper_bound_CI && true_risk > lower_bound_CI
  se_rep_CV_fold_risk[iter] <- se
}
```

```r
par(mfrow=c(1,5))

df_dens <- data.frame(as.numeric(est_2_fold_risk))
colnames(df_dens) <- c("x")
plot(ggplot(data = df_dens, aes(x = x)) +
  geom_density() + labs(title = "Density 2-Fold CV",
             caption = "Density of estimated risk - true risk") +
   xlab("est_risk - true_risk"))
```



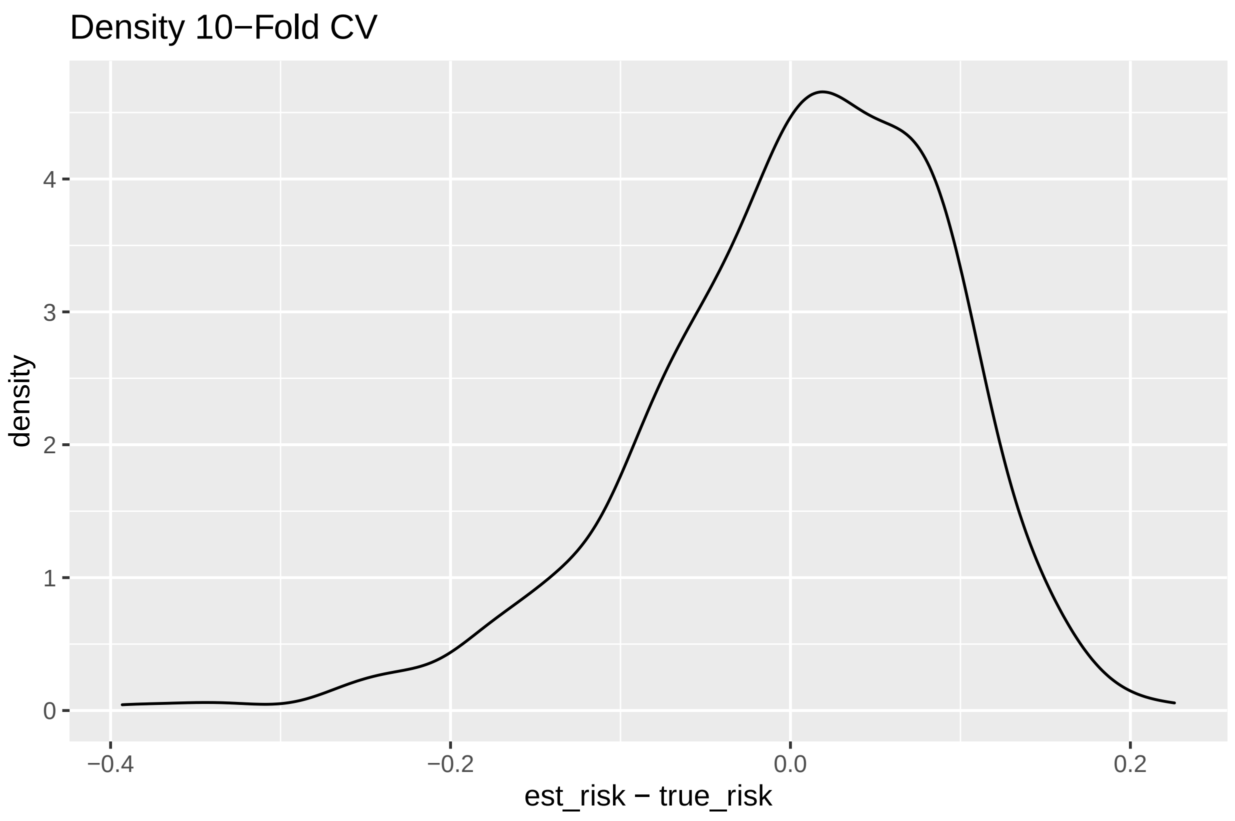
Density of estimated risk − true risk

```r
df_dens <- data.frame(as.numeric(est_4_fold_risk))
colnames(df_dens) <- c("x")
plot(ggplot(data = df_dens, aes(x = x)) +
  geom_density() + labs(title = "Density 4-Fold CV",
              caption = "Density of estimated risk - true risk") +
   xlab("est_risk - true_risk"))
```



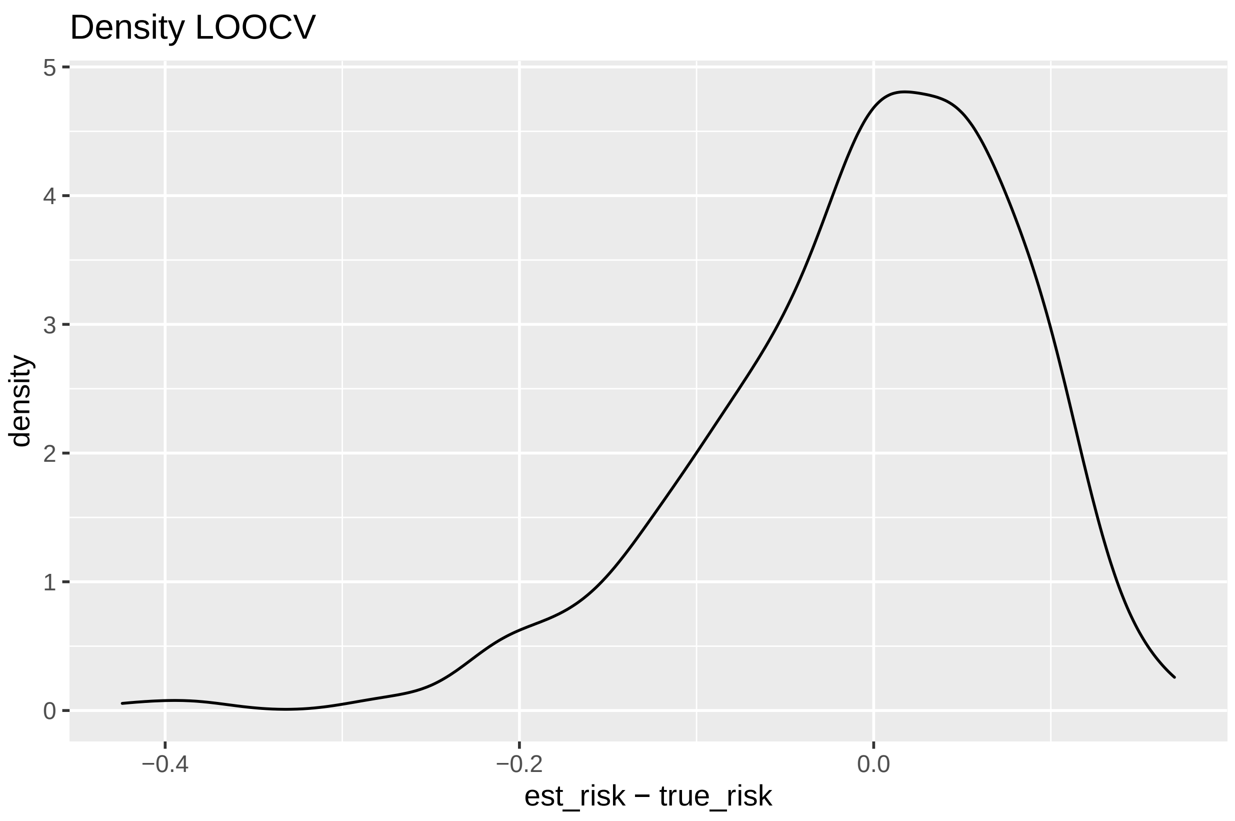
Density of estimated risk − true risk

```
df_dens <- data.frame(as.numeric(est_10_fold_risk))
colnames(df_dens) <- c("x")
plot(ggplot(data = df_dens, aes(x = x)) +
  geom_density() + labs(title = "Density 10-Fold CV",
              caption = "Density of estimated risk - true risk") +
   xlab("est_risk - true_risk"))
```



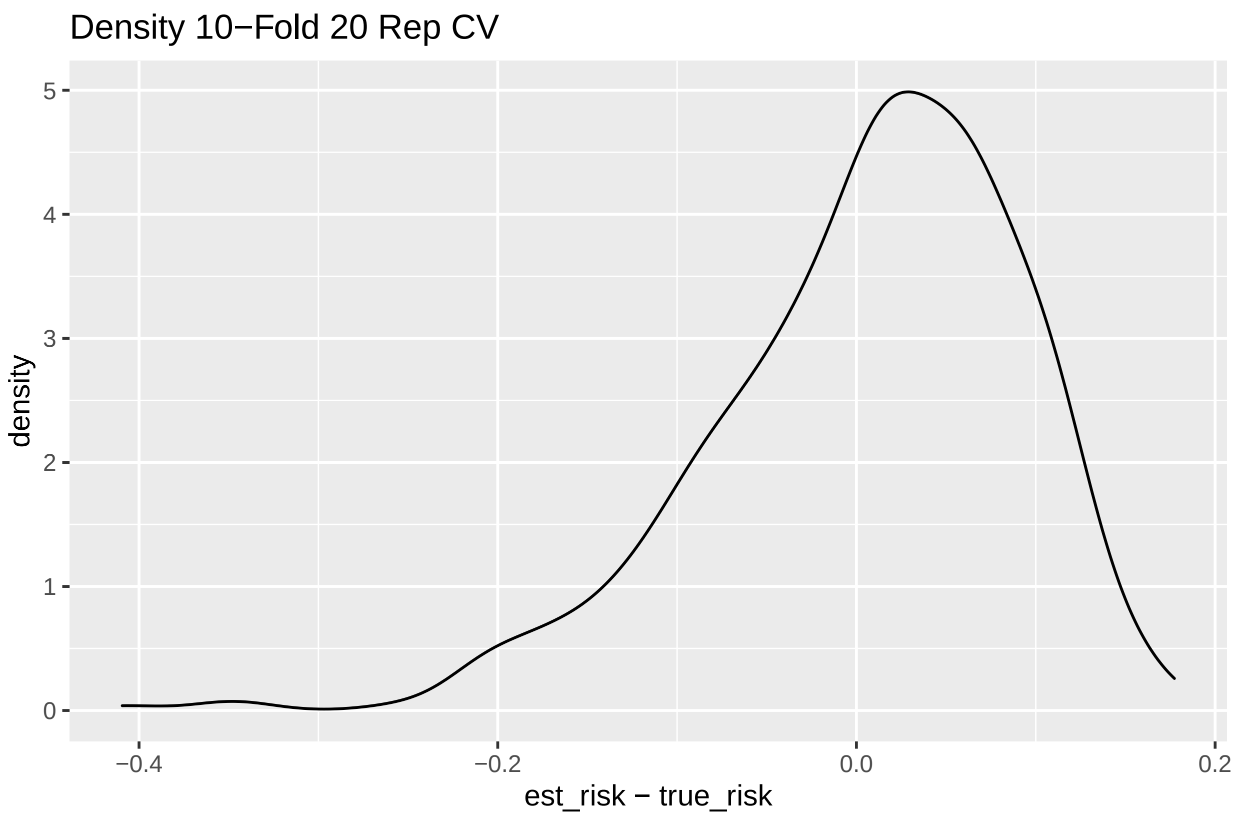
Density of estimated risk − true risk

```
df_dens <- data.frame(as.numeric(est_LOOCV_risk))
colnames(df_dens) <- c("x")
plot(ggplot(data = df_dens, aes(x = x)) +
  geom_density() + labs(title = "Density LOOCV",
              caption = "Density of estimated risk - true risk") +
   xlab("est_risk - true_risk"))
```



Density of estimated risk − true risk

```r
df_dens <- data.frame(as.numeric(est_rep_CV_risk))
colnames(df_dens) <- c("x")
plot(ggplot(data = df_dens, aes(x = x)) +
  geom_density() + labs(title = "Density 10-Fold 20 Rep CV",
              caption = "Density of estimated risk - true risk") +
   xlab("est_risk - true_risk"))
```



Density of estimated risk − true risk

```
print("2-Fold CV")
```

```
## [1] "2-Fold CV"
```

```
sprintf("Mean difference: %f", mean(as.numeric(est_2_fold_risk)))
```

```
## [1] "Mean difference: 0.355854"
```

```
sprintf("Median standard error: %f", median(as.numeric(se_2_fold_risk)))
```

```
## [1] "Median standard error: 0.109597"
```

```
sprintf("Percentage of 95CI that contain the true risk proxy: %f", mean(as.numeric(contains95CI_2_fold)
```

```
## [1] "Percentage of 95CI that contain the true risk proxy: 0.676000"
```

```
print("4-Fold CV")
```

```
## [1] "4-Fold CV"
```

```
sprintf("Mean difference: %f", mean(as.numeric(est_4_fold_risk)))
```

```
## [1] "Mean difference: 0.036523"
```

```
sprintf("Median standard error: %f", median(as.numeric(se_4_fold_risk)))
```

```
## [1] "Median standard error: 0.083625"
```

```
sprintf("Percentage of 95CI that contain the true risk proxy: %f", mean(as.numeric(contains95CI_4_fold)
```

```
## [1] "Percentage of 95CI that contain the true risk proxy: 0.916000"
```

```
print("10-Fold CV")
```

```
## [1] "10-Fold CV"
```

```
sprintf("Mean difference: %f", mean(as.numeric(est_10_fold_risk)))
```

```
## [1] "Mean difference: 0.003313"
```

```
sprintf("Median standard error: %f", median(as.numeric(se_10_fold_risk)))
```

```
## [1] "Median standard error: 0.077124"
```

```
sprintf("Percentage of 95CI that contain the true risk proxy: %f", mean(as.numeric(contains95CI_10_fold
```

```
## [1] "Percentage of 95CI that contain the true risk proxy: 0.918000"
```

```
print("LOOCV-Fold CV")
```

```
## [1] "LOOCV-Fold CV"
```

```
sprintf("Mean difference: %f", mean(as.numeric(est_LOOCV_risk)))
```

```
## [1] "Mean difference: -0.006358"
```

```
sprintf("Median standard error: %f", median(as.numeric(se_LOOCV_fold_risk)))
```

```
## [1] "Median standard error: 0.075008"
```

```
sprintf("Percentage of 95CI that contain the true risk proxy: %f", mean(as.numeric(contains95CI_LOOCV_f
```

```
## [1] "Percentage of 95CI that contain the true risk proxy: 0.920000"
```

```
print("10-Fold 20 Rep CV")
```

```
## [1] "10-Fold 20 Rep CV"
```

```
sprintf("Mean difference: %f", mean(as.numeric(est_rep_CV_risk)))
```

```
## [1] "Mean difference: 0.004866"
```

```
sprintf("Median standard error: %f", median(as.numeric(se_rep_CV_fold_risk)))
```

```
## [1] "Median standard error: 0.076552"
```

```
sprintf("Percentage of 95CI that contain the true risk proxy: %f", mean(as.numeric(contains95CI_rep_CV_
```

```
## [1] "Percentage of 95CI that contain the true risk proxy: 0.934000"
```

We managed to obtain a better estimate with cross-validation. If we increase k in k-Fold, we obtain evenmore reliable results, since the mean difference (lower bis) and standard error get smaller. LOOCV and repeated k-Fold CV are the most reliable for estimating the risk of the model. In practice using repeated k-Fold CV is a better choice since can be less compuationaly expensive (depending on which parameters we choose).

A different scenario

No. The risk estimate can not get worse (increase in bias) if we increase the number of splits.