# Simulations for Sharpening Wald-type Inference in Robust Regression for Small Samples

Manuel Koller

November 1, 2024

## Contents



## 1 Introduction

In this vignette, we recreate the simulation study of Koller and Stahel (2011). This vignette is supposed to complement the results presented in the above cited reference and render its results reproducible. Another goal is to provide simulation functions, that, with small changes, could also be used for other simulation studies.

Additionally, in Section 5, we calculate the maximum asymptotic bias curves of the $\psi$-functions used in the simulation.

## 2 Setting

The simulation setting used here is similar to the one in Maronna and Yohai (2010). We simulate $N = 1000$ repetitions. To repeat the simulation, we recommend using a small value of $N$ here, since for large $n$ and $p$, computing all the replicates will take days.

### 2.1 Methods

We compare the methods

- MM, SMD, SMDM as described in Koller and Stahel (2011). These methods are available in the package `robustbase` (`lmrob`).

- MM as implemented in the package `robust` (`lmRob`). This method will be denoted as *MMrobust* later on.
- MM using S-scale correction by $q_{\mathrm{T}}$ and $q_{\mathrm{E}}$ as proposed by Maronna and Yohai (2010).

  $q_{\mathrm{T}}$ and $q_{\mathrm{E}}$ are defined as follows.

$$q_{\mathrm{E}} = \frac{1}{1-(1.29-6.02/n)p/n},$$

$$\hat{q}_{\mathrm{T}} = 1 + \frac{p}{2n}\frac{\hat{a}}{\hat{b}\hat{c}},$$

  where

$$\hat{a} = \frac{1}{n}\sum_{i=1}^{n}\psi\left(\frac{r_i}{\hat{\sigma}_{\mathrm{S}}}\right)^2, \hat{b} = \frac{1}{n}\sum_{i=1}^{n}\psi'\left(\frac{r_i}{\hat{\sigma}_{\mathrm{S}}}\right), \hat{c} = \frac{1}{n}\sum_{i=1}^{n}\psi\left(\frac{r_i}{\hat{\sigma}_{\mathrm{S}}}\right)\frac{r_i}{\hat{\sigma}_{\mathrm{S}}},$$

  with $\psi = \rho'$,$n$ the number of observations, $p$ the number of predictor variables, $\hat{\sigma}_{\mathrm{S}}$ is the S-scale estimate and $r_i$ is the residual of the $i$-th observation.

  When using $q_{\mathrm{E}}$ it is necessary to adjust the tuning constants of $\chi$ to account for the dependence of $\kappa$ on $p$. For $q_{\mathrm{T}}$ no change is required.

  This method is implemented as `lmrob.mar()` in the source file `estimating.functions.R`.

### 2.2 $\psi$-Functions

We compare *bisquare*, *optimal*, *lqq* and *Hampel* $\psi$-functions. They are illustrated in Fig. 1. The tuning constants used in the simulation are compiled in Table 1. Note that the *Hampel* $\psi$-function is tuned to have a downward slope of $-1/3$ instead of the originally proposed $-1/2$. This was set to allow for a comparison to an even slower descending $\psi$-function.

| psi | tuning.chi | tuning.psi |
|---|---|---|
| optimal | 0.405 | 1.06 |
| bisquare | 1.548 | 4.685 |
| lqq | −0.5, 1.5, NA, 0.5 | −0.5, 1.5, 0.95, NA |
| hampel | 0.318, 0.742, 1.695 | 1.352, 3.156, 7.213 |

Table 1: Tuning constants of $\psi$-functions used in the simulation.

### 2.3 Designs

Two types of designs are used in the simulation: fixed and random designs. One design with $n = 20$ observations, $p = 1+3$ predictors and strong leverage points. This design also includes an intercept column. It is shown in Fig. 21. The other designs are random, i.e., regenerated for every repetition, and the models are fitted without an intercept. We use the same distribution to generate the designs as for the errors. The number of observations simulated are $n = 25, 50, 100, 400$ and the ratio to the number of parameters are $p/n = 1/20, 1/10, 1/5, 1/3, 1/2$. We round $p$ to the nearest smaller integer if necessary.

The random datasets are generated using the following code.

```
> f.gen <- function(n, p, rep, err) {
+   ## get function name and parameters
+   lerrfun <- f.errname(err$err)
+   lerrpar <- err$args
+   ## generate random predictors
```

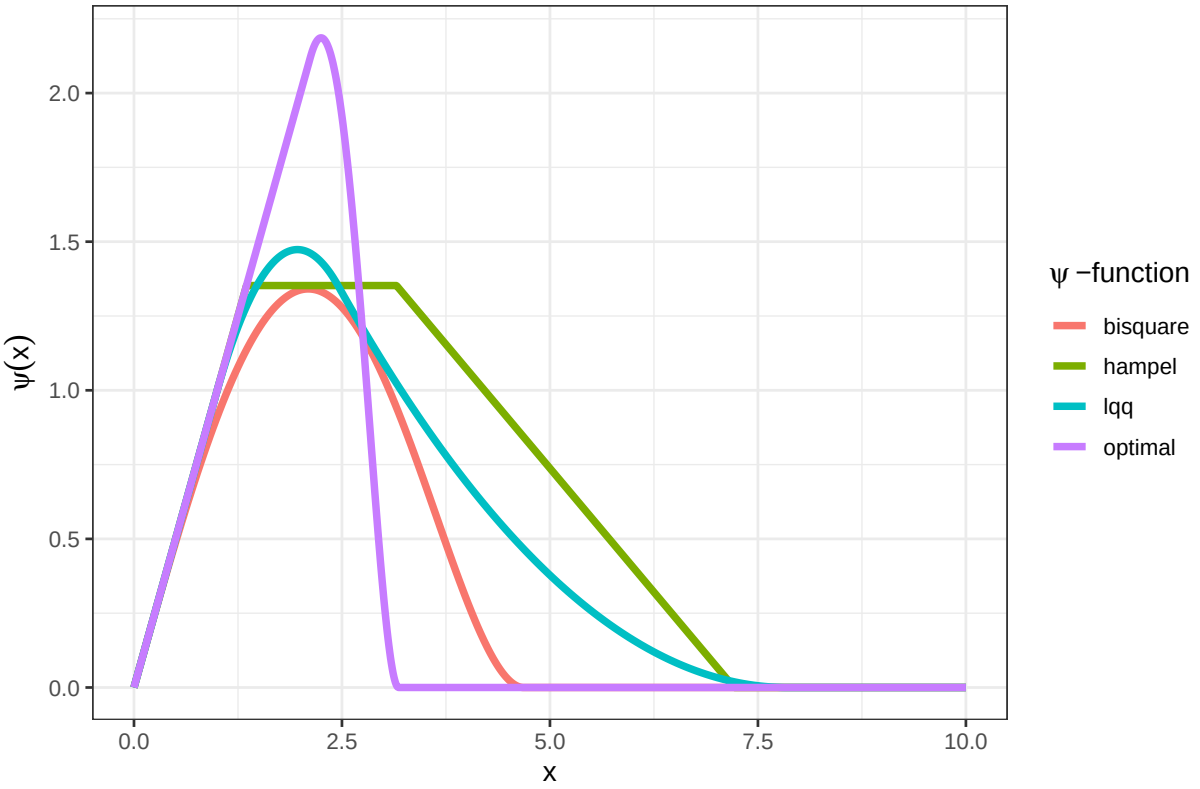


Figure 1: $\psi$-functions used in the simulation.

```
+   ret <- replicate(rep, matrix(do.call(lerrfun, c(n = n*p, lerrpar)),
+                                n, p), simplify=FALSE)
+   attr(ret[[1]], 'gen') <- f.gen
+   ret
+ }
> ratios <- c(1/20, 1/10, 1/5, 1/3, 1/2)## p/n
> lsit <- expand.grid(n = c(25, 50, 100, 400), p = ratios)
> lsit <- within(lsit, p <- as.integer(n*p))
> .errs.normal.1 <- list(err = 'normal',
+                        args = list(mean = 0, sd = 1))
> for (i in 1:NROW(lsit))
+   assign(paste('rand',lsit[i,1],lsit[i,2],sep='_'),
+          f.gen(lsit[i,1], lsit[i,2], rep = 1, err = .errs.normal.1)[[1]])
```

An example design is shown in Fig. 2.

### 2.4 Error Distributions

We simulate the following error distributions

- standard normal distribution,
- $t_5$, $t_3$, $t_1$,
- centered skewed t with $df = \infty, 5$ and $\gamma = 2$ (denoted by $cskt(\infty, 2)$ and $cskt(5, 2)$, respectively); as introduced by Fernández and Steel (1998) using the `R` package `skewt`,
- contaminated normal, $\mathcal{N}(0, 1)$ contaminated with 10% $\mathcal{N}(0, 10)$ (symmetric, $cnorm(0.1, 0, 3.16)$) or $\mathcal{N}(4, 1)$ (asymmetric, $cnorm(0.1, 4, 1)$).

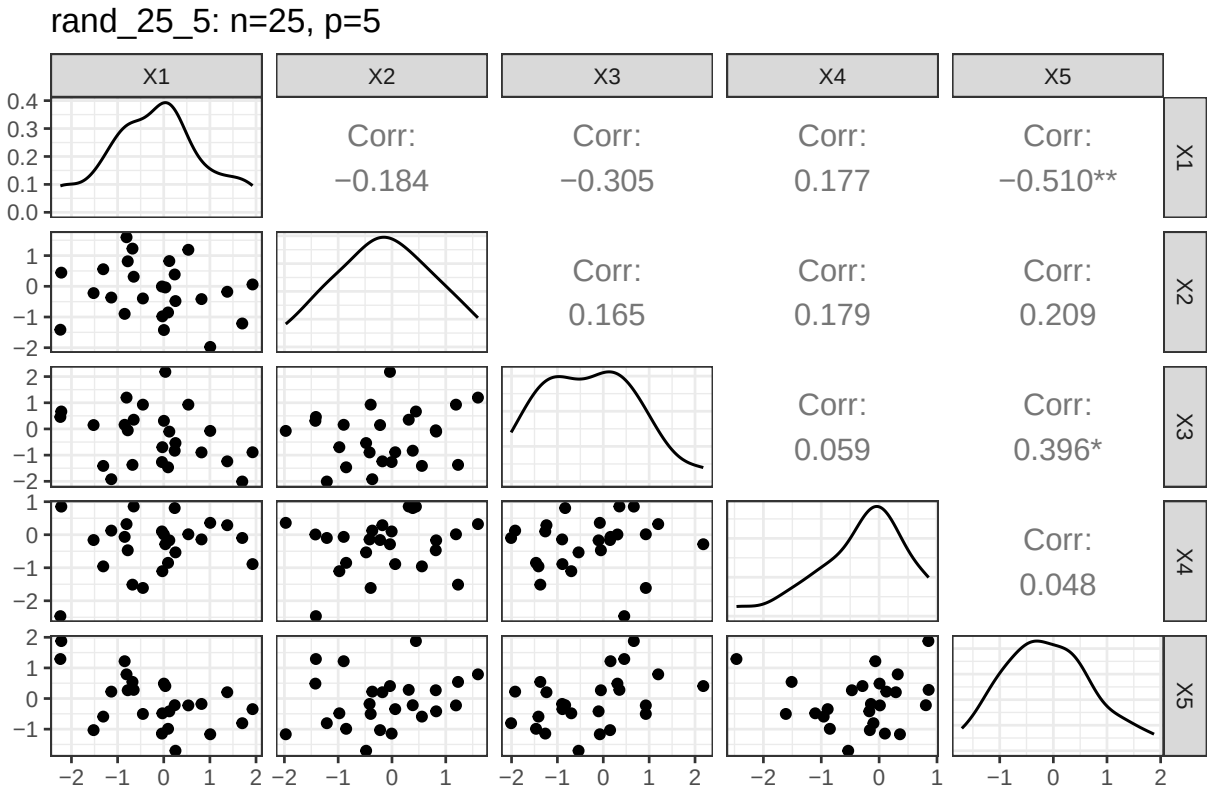


Figure 2: Example random design.

### 2.5 Covariance Matrix Estimators

For the standard MM estimator, we compare $\text{Avar}_1$ of Croux et al. (2003) and the empirical weighted covariance matrix estimate corrected by Huber's small sample correction as described in Huber and Ronchetti (2009) (denoted by *Wssc*). The latter is also used for the variation of the MM estimate proposed by Maronna and Yohai (2010). For the SMD and SMDM variants we use the covariance matrix estimate as described in Koller and Stahel (2011) ($W\tau$).

The covariance matrix estimate consists of three parts:

$$\text{cov}(\hat{\beta}) = \sigma^2 \gamma \boldsymbol{V}_{\boldsymbol{X}}^{-1}.$$

The SMD and SMDM methods of `lmrob` use the following defaults.

$$\hat{\gamma} = \frac{\frac{1}{n}\sum_{i=1}^{n} \tau_i^2 \psi\left(\frac{r_i}{\tau_i \hat{\sigma}}\right)^2}{\frac{1}{n}\sum_{i=1}^{n} \psi'\left(\frac{r_i}{\tau_i \hat{\sigma}}\right)} \tag{1}$$

where $\tau_i$ is the rescaling factor used for the D-scale estimate (see Koller and Stahel (2011)).
**Remark:** Equation (1) is a corrected version of $\gamma$. It was changed in `robustbase` version `0.91` (April 2014) to ensure that the equation reduces to 1 in the classical case ($\psi(x) = x$). If the former (incorrect) version is needed for compatibility reasons, it can be obtained by adding the argument `cov.corrfact = "tauold"`.

$$\widehat{\boldsymbol{V}}_{\boldsymbol{X}} = \frac{1}{\frac{1}{n}\sum_{i=1}^{n} \text{w}_{ii}} \boldsymbol{X}^T \boldsymbol{W} \boldsymbol{X}$$

where $\boldsymbol{W} = \text{diag}\left(\text{w}\left(\frac{r_1}{\hat{\sigma}}\right), \ldots, \text{w}\left(\frac{r_n}{\hat{\sigma}}\right)\right)$. The function $\text{w}(r) = \psi(r)/r$ produces the robustness weights.

# 3 Simulation

The main loop of the simulation is fairly simple. (This code is only run if there are no aggregate results available.)

```
> aggrResultsFile <- file.path(robustDta, "aggr_results.Rdata")
```

```
> if (!file.exists(aggrResultsFile)) {
+   ## load packages required only for simulation
+   stopifnot(require(robust),
+             require(skewt),
+             require(foreach))
+   if (!is.null(getOption("cores"))) {
+       if (getOption("cores") == 1)
+           registerDoSEQ() ## no not use parallel processing
+       else {
+           stopifnot(require(doParallel))
+           if (.Platform$OS.type == "windows") {
+               cl <- makeCluster(getOption("cores"))
+               clusterExport(cl, c("N", "robustDoc"))
+               clusterEvalQ(cl, slave <- TRUE)
+               clusterEvalQ(cl, source(file.path(robustDoc, 'simulation.init.R')))
+               registerDoParallel(cl)
+           } else registerDoParallel()
+       }
+   } else registerDoSEQ() ## no not use parallel processing
+   for (design in c("dd", ls(pattern = 'rand_\\d+_\\d+'))) {
+     print(design)
+     ## set design
+     estlist$design <- get(design)
+     estlist$use.intercept <- !grepl('^rand', design)
+     ## add design.predict: pc
+     estlist$design.predict <-
+       if (is.null(attr(estlist$design, 'gen')))
+         f.prediction.points(estlist$design) else
+       f.prediction.points(estlist$design, max.pc = 2)
+
+     filename <- file.path(robustDta,
+                           sprintf('r.test.final.%s.Rdata',design))
+     if (!file.exists(filename)) {
+       ## run
+       print(system.time(r.test <- f.sim(estlist, silent = TRUE)))
+       ## save
+       save(r.test, file=filename)
+       ## delete output
+       rm(r.test)
+       ## run garbage collection
+       gc()
+     }
+   }
+ }
```

The variable `estlist` is a list containing all the necessary settings required to run the simulation as outlined above. Most of its elements are self-explanatory.

```
> str(estlist, 1)

List of 8
 $ design        :'data.frame':        20 obs. of  3 variables:
 $ nrep          : num 1000
 $ errs          :List of 8
 $ seed          : num 13082010
 $ procedures    :List of 21
 $ design.predict:'data.frame':        10 obs. of  3 variables:
  ..- attr(*, "npcs")= int 3
 $ output        :List of 6
 $ use.intercept : logi TRUE
```

`errs` is a list containing all the error distributions to be simulated. The entry for the standard normal looks as follows.

```
> estlist$errs[[1]]

$err
[1] "normal"

$args
$args$mean
[1] 0

$args$sd
[1] 1
```

`err` is translated internally to the corresponding random generation or quantile function, e.g., in this case `rnorm` or `qnorm`. `args` is a list containing all the required arguments to call the function. The errors are then generated internally with the following call.

```
> set.seed(estlist$seed)
> errs <- c(sapply(1:nrep, function(x) do.call(fun, c(n = nobs, args))))
```

All required random numbers are generated at once instead of during the simulation. Like this, it is certain, that all the compared methods run on exactly the same data.

The entry `procedures` follows a similar convention. `design.predict` contains the design used for the prediction of observations and calculation of confidence or prediction intervals. The objects returned by the procedures are processed by the functions contained in the `estlist$output` list.

```
> str(estlist$output[1:3], 2)

List of 3
 $ sigma:List of 2
  ..$ names: chr "sigma"
  ..$ fun  : language sigma(lrr)
 $ beta :List of 2
  ..$ names: language paste("beta", 1:npar, sep = "_")
  ..$ fun  : language coef(lrr)
 $ se   :List of 2
  ..$ names: language paste("se", 1:npar, sep = "_")
  ..$ fun  : language sqrt(diag(covariance.matrix(lrr)))
```

The results are stored in a 4-dimensional array. The dimensions are: repetition number, type of value, procedure id, error id. Using `apply` it is very easy and fast to generate summary statistics. The raw results are stored on the hard disk, because typically it takes much longer to execute all the procedures than to calculate the summary statistics. The variables saved take up a lot of space quite quickly, so only the necessary data is stored. These are $\sigma$, $\boldsymbol{\beta}$ as well as the corresponding standard errors.

To speed up the simulation routine `f.sim`, the simulations are carried out in parallel, as long as this is possible. This is accomplished with the help of the `R`-package `foreach`. This is most easily done on a machine with multiple processors or cores. The `multicore` package provides the methods to do so easily. The worker processes are just forked from the main `R` process.

After all the methods have been simulated, the simulation output is processed. The code is quite lengthy and thus not displayed here (check the Sweave source file `lmrob_simulation.Rnw`). The residuals, robustness weights, leverages and $\tau$ values have to be recalculated. Using vectorized operations and some specialized `C` code, this is quite cheap. The summary statistics generated are discussed in the next section.

# 4 Simulation Results

## 4.1 Criteria

The simulated methods are compared using the following criteria.

**Scale estimates.** The criteria for scale estimates are all calculated on the log-scale. The bias of the estimators is measured by the 10% trimmed mean. To recover a meaningful scale, the results are exponentiated before plotting. It is easy to see that this is equivalent to calculating geometric means. Since the methods are all tuned at the central model, $\mathcal{N}(0,1)$, a meaningful comparison of biases can only be made for $\mathcal{N}(0,1)$ distributed errors.

The variability of the estimators, on the other hand, can be compared over all simulated error distributions. It is measured by the 10% trimmed standard deviation, rescaled by the square root of the number of observations.

For completeness, the statistics used to compare scale estimates in Maronna and Yohai (2010) are also calculated. They are defined as

$$q = \text{median}\left(\frac{S(\boldsymbol{e})}{\hat{\sigma}_S}\right), \quad M = \text{mad}\left(\frac{S(\boldsymbol{e})}{\hat{\sigma}_S}\right), \tag{2}$$

where $S(e)$ stands for the S-scale estimate evaluated for the actual errors $\boldsymbol{e}$. For the D-scale estimate, the definition is analogue. Since there is no design to correct for, we set $\tau_i = 1\ \forall i$.

**Coefficients.** The efficiency of estimated regression coefficients $\hat{\boldsymbol{\beta}}$ is characterized by their mean squared error ($MSE$). Since we simulate under $H_0 : \boldsymbol{\beta} = 0$, this is determined by the covariance matrix of $\hat{\boldsymbol{\beta}}$. We use $\mathbf{E}\left[\|\hat{\boldsymbol{\beta}}\|_2^2\right] = \sum_{j=1}^{p} \text{var}(\hat{\beta}_j)$ as a summary. When comparing to the MSE of the ordinary least squares estimate ($OLS$), this gives the efficiency, which, by the choice of tuning constants of $\psi$, should yield

$$\frac{\text{MSE}(\hat{\boldsymbol{\beta}}_{\text{OLS}})}{\text{MSE}(\hat{\boldsymbol{\beta}})} \approx 0.95$$

for standard normally distributed errors. The simulation mean of $\sum_{j=1}^{p} \text{var}(\hat{\beta}_j)$ is calculated with 10% trimming. For other error distributions, this ratio should be larger than 1, since by using robust procedures we expect to gain efficiency at other error distributions (relative to the least squares estimate).

**$\boldsymbol{\gamma}$.** We compare the behavior of the various estimators of $\gamma$ by calculating the trimmed mean and the trimmed standard deviation for standard normal distributed errors.

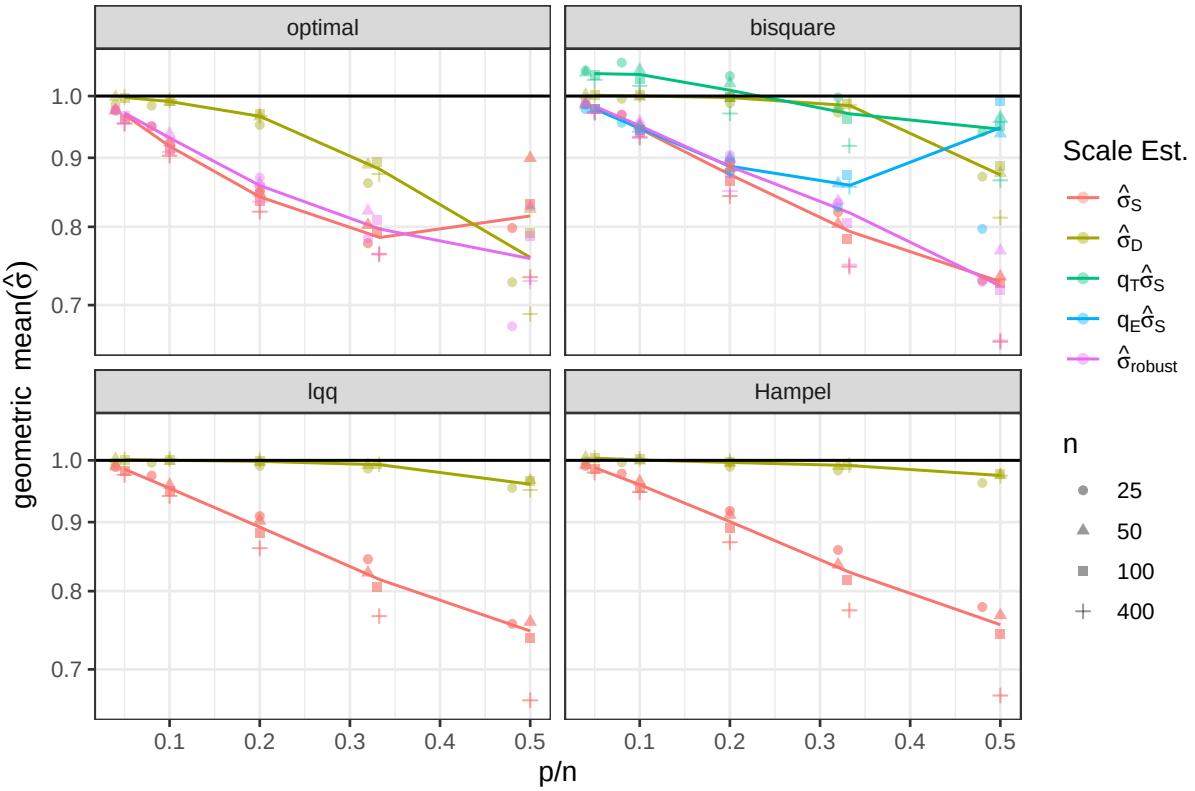


Figure 3: Mean of scale estimates for normal errors. The mean is calculated with 10% trimming. The lines connect the median values for each simulated ratio $p/n$. Results for random designs only.

**Covariance matrix estimate.** The covariance matrix estimates are compared indirectly over the performance of the resulting test statistics. We compare the empirical level of the hypothesis tests $H_0 : \beta_j = 0$ for some $j \in \{1, \ldots, p\}$. The power of the tests is compared by testing for $H_0 : \beta_j = b$ for several values of $b > 0$. The formal power of a more liberal test is generally higher. Therefore, in order for this comparison to be meaningful, the critical value for each test statistic was corrected such that all tests have the same simulated level of 5%.

The simple hypothesis tests give only limited insights. To investigate the effects of other error distributions, e.g., asymmetric error distributions, we compare the confidence intervals for the prediction of some fixed points. Since it was not clear how to assess the quality prediction intervals, either at the central or the simulated model, we do not calculate them here.

A small number of prediction points is already enough, if they are chosen properly. We chose to use seven points lying on the first two principal components, spaced evenly from the center of the design used to the extended range of the design. The principal components were calculated robustly (using `covMcd` of the `robustbase` package) and the range was extended by a fraction of 0.5. An example is shown in Figure 21.

## 4.2 Results

The results are given here as plots (Fig. 3 to Fig. 22). For a complete discussion of the results, we refer to Koller and Stahel (2011).

The different $\psi$-functions are each plotted in a different facet, except for Fig. 8, Fig. 9 and Fig. 15, where the facets show the results for various error distributions. The plots are augmented with auxiliary lines to ease the comparison of the methods. The lines connect the median values over the values of $n$ for each simulated ratio $p/n$. In many plots the y-axis has been truncated. Points in the grey shaded area represent truncated values using a different scale.

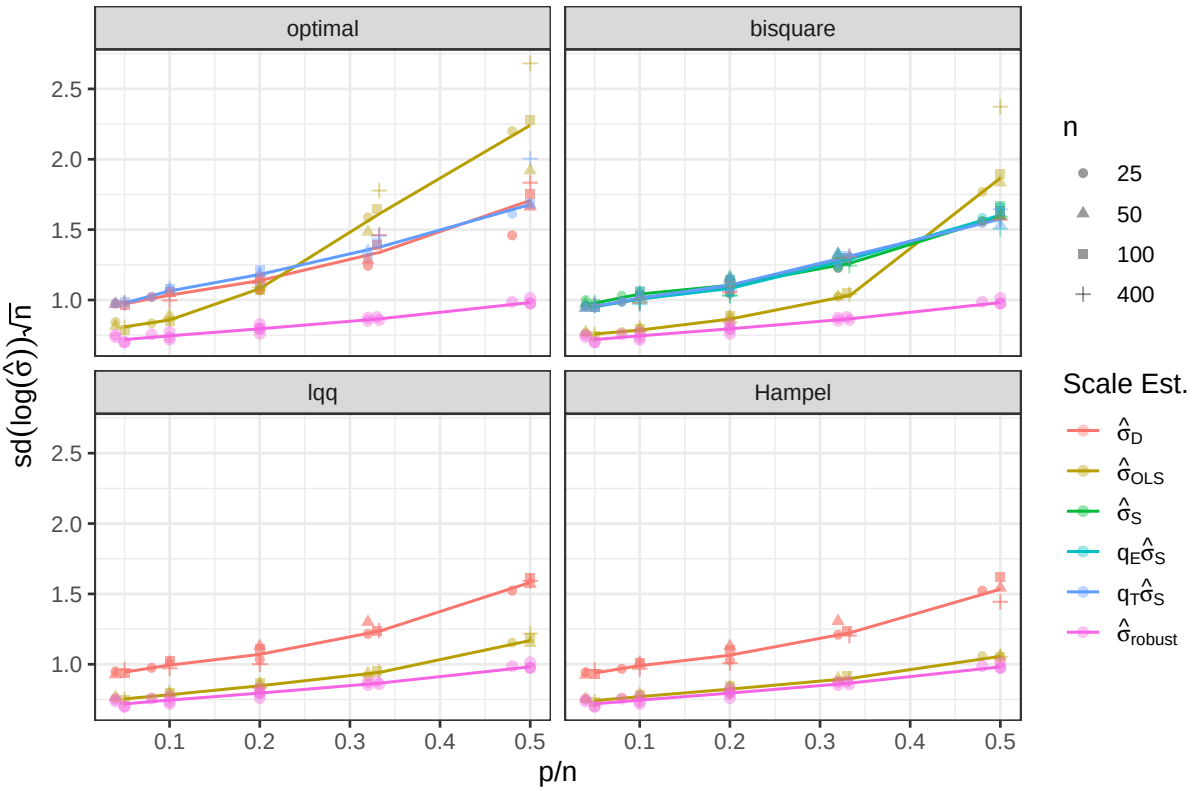


Figure 4: Variability of the scale estimates for normal errors. The standard deviation is calculated with 10% trimming.

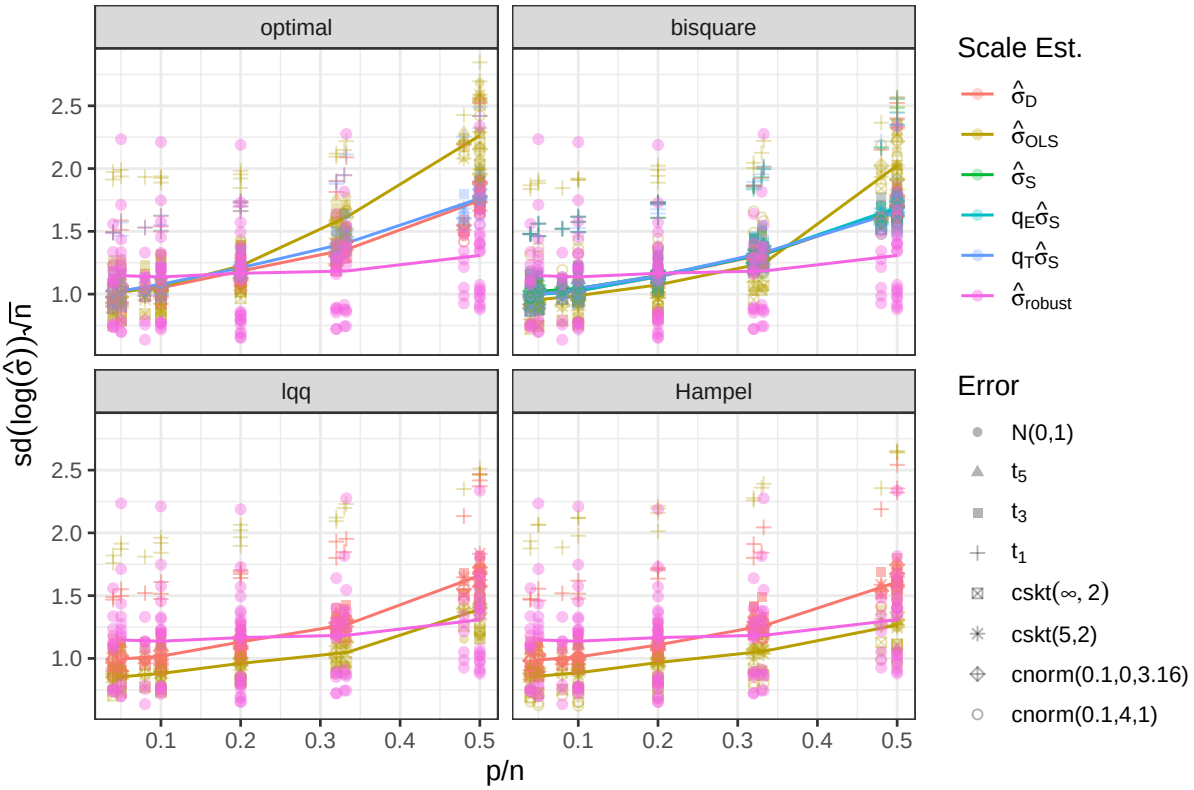


Figure 5: Variability of the scale estimates for all simulated error distributions.

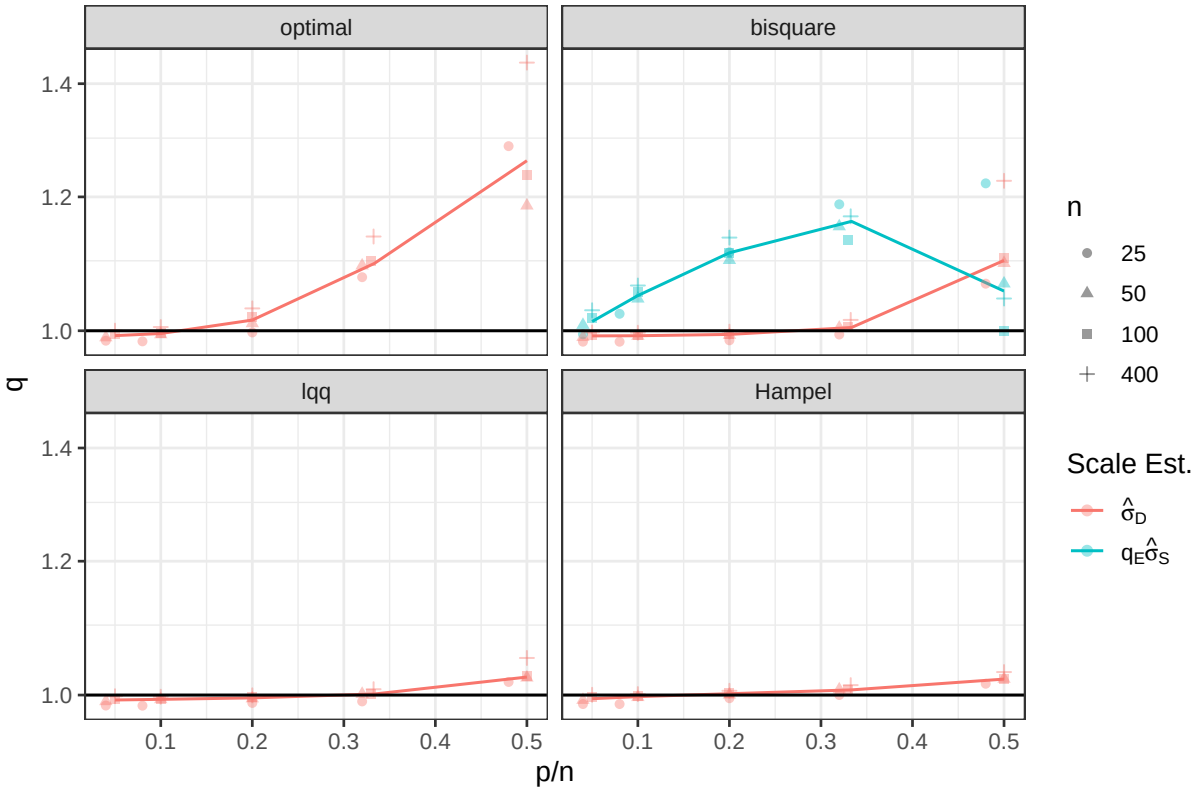


Figure 6: $q$ statistic for normal errors. $q$ is defined in (2).

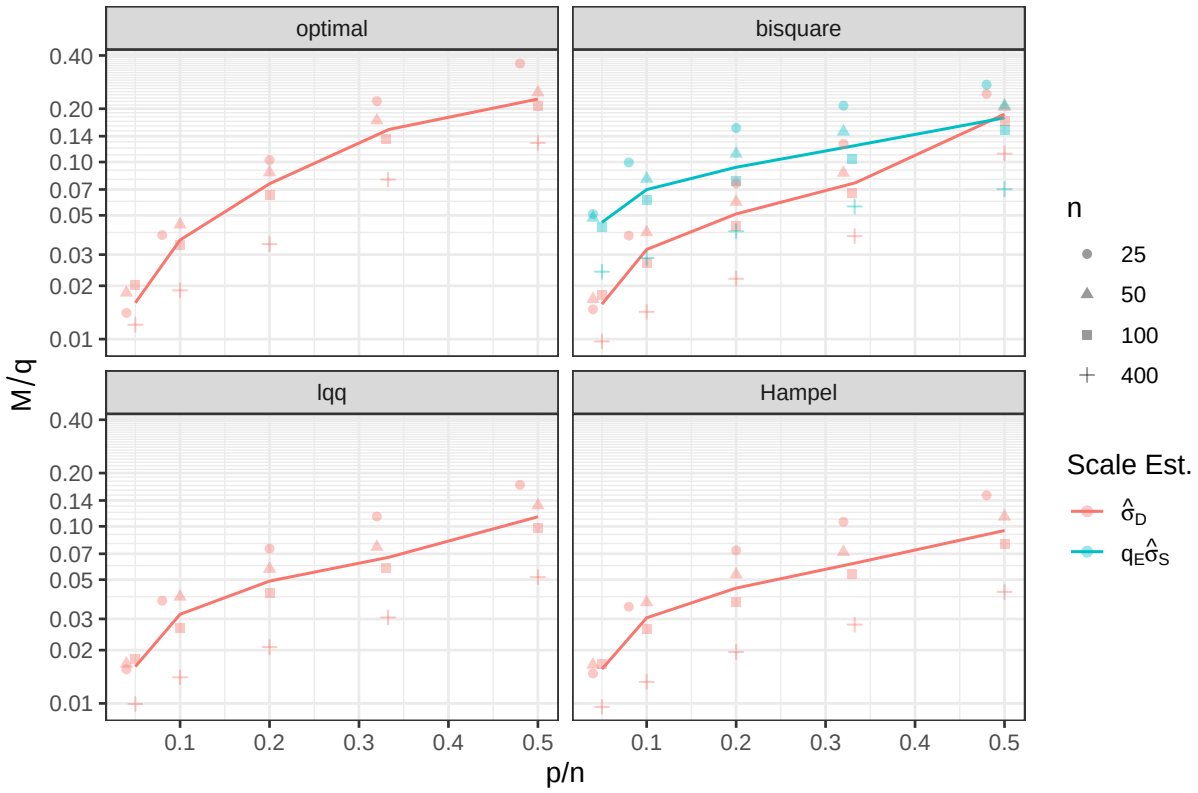


Figure 7: $M/q$ statistic for normal errors. $M$ and $q$ are defined in (2).

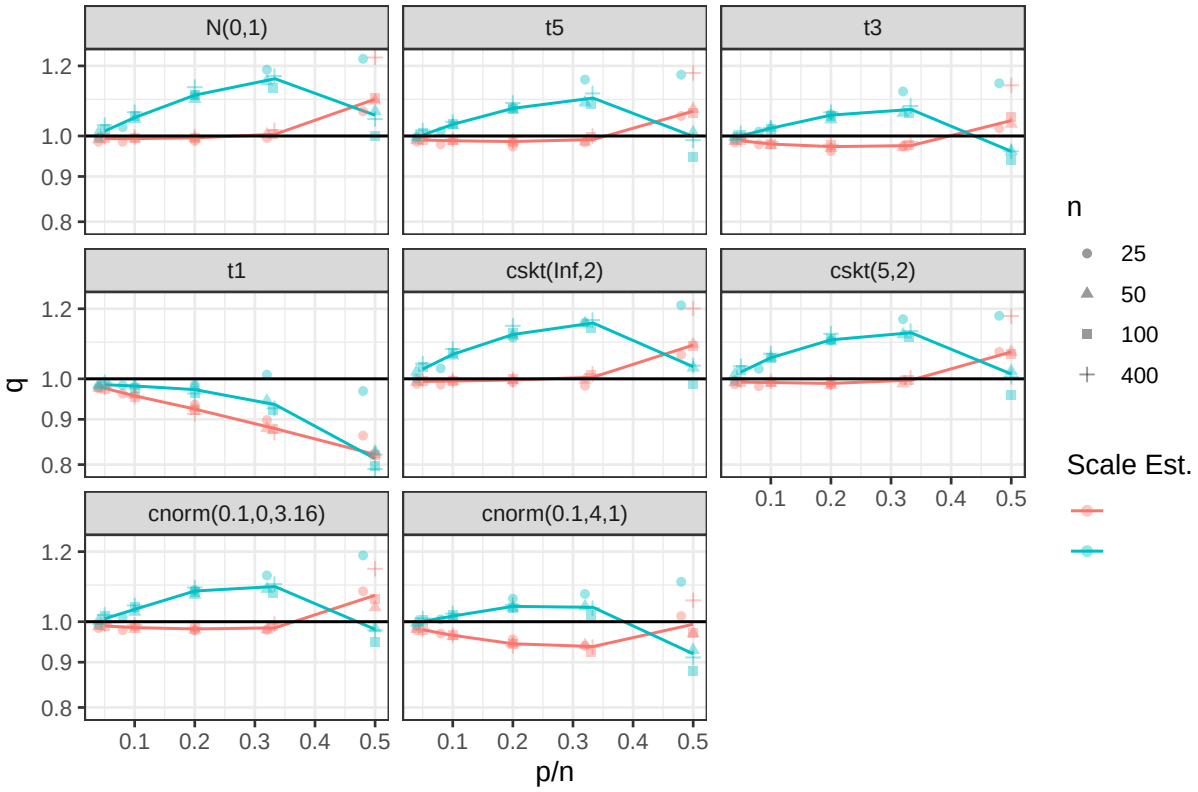


Figure 8: $q$ statistic for *bisquare* $\psi$.

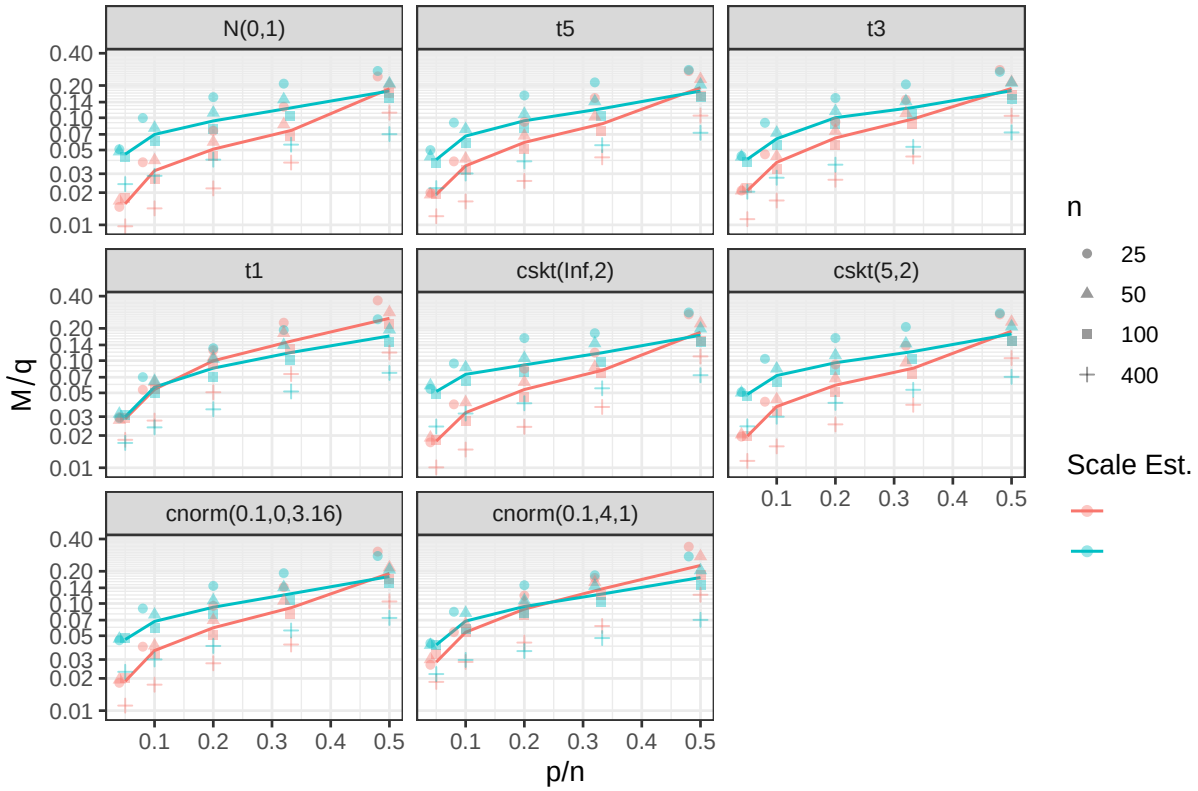


Figure 9: $M/q$ statistic for *bisquare* $\psi$.

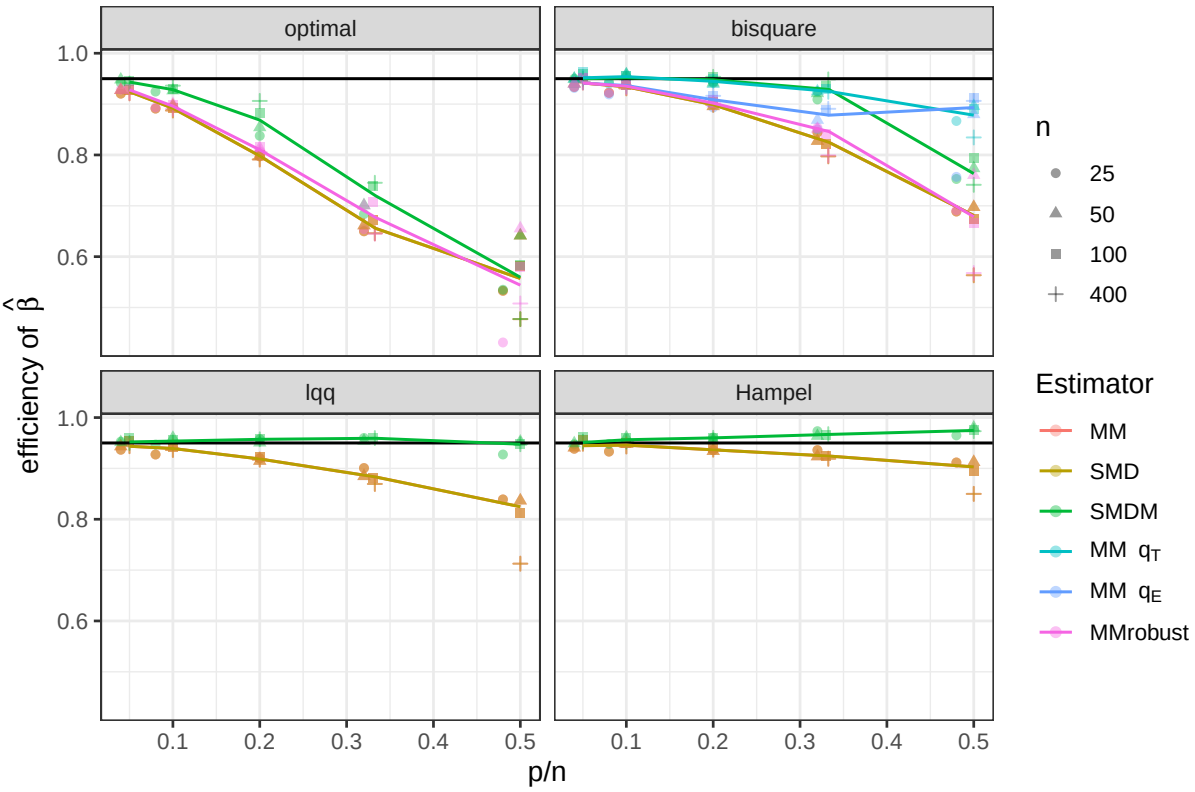


Figure 10: Efficiency for normal errors. The efficiency is calculated by comparing to an OLS estimate and averaging with 10% trimming.

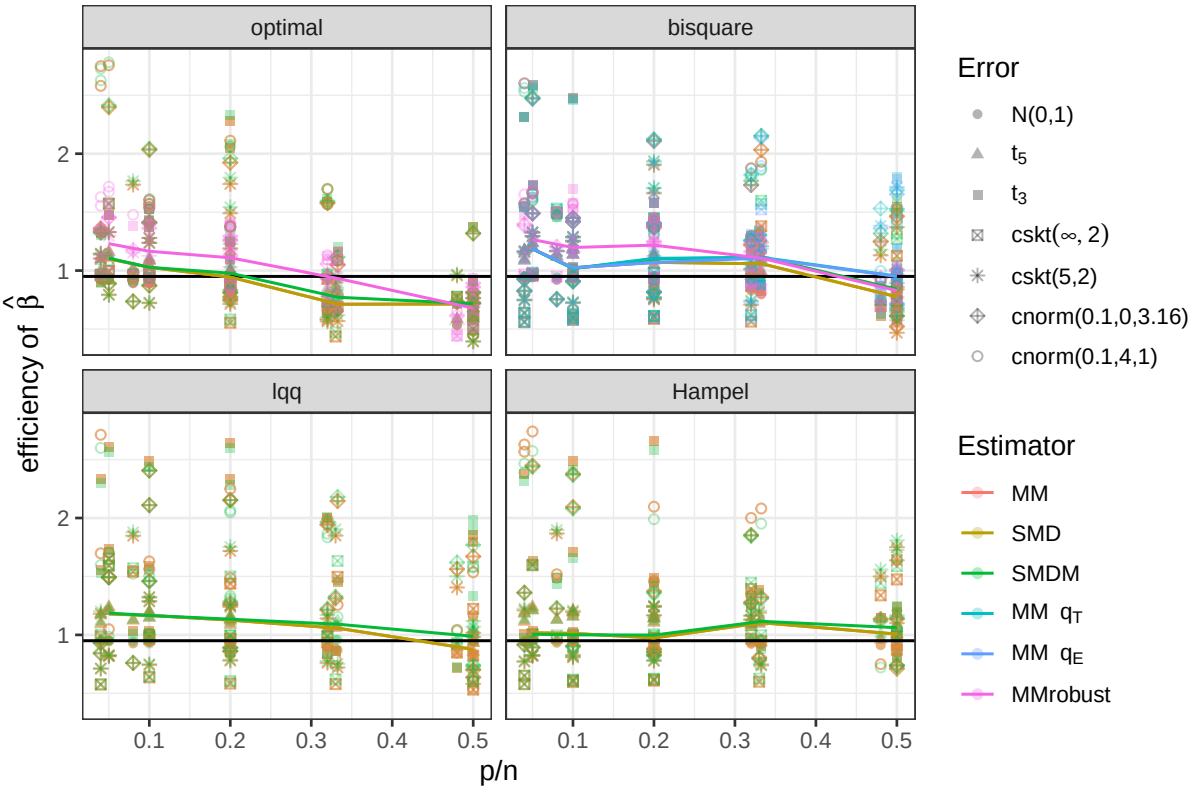


Figure 11: Efficiency for all simulated error distributions except $t_1$.

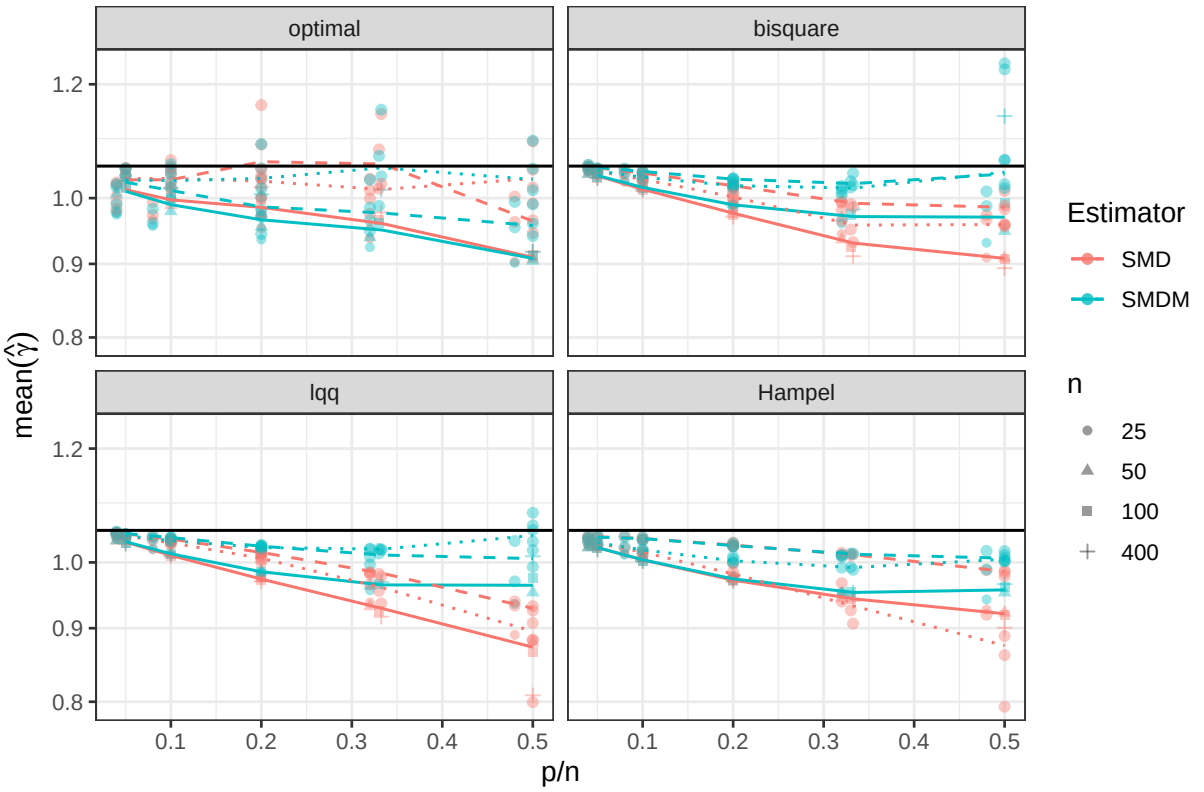


Figure 12: Comparing the estimates of $\gamma$. The solid line connects the uncorrected estimate, dotted the $\tau$ corrected estimate and dashed Huber's small sample correction.

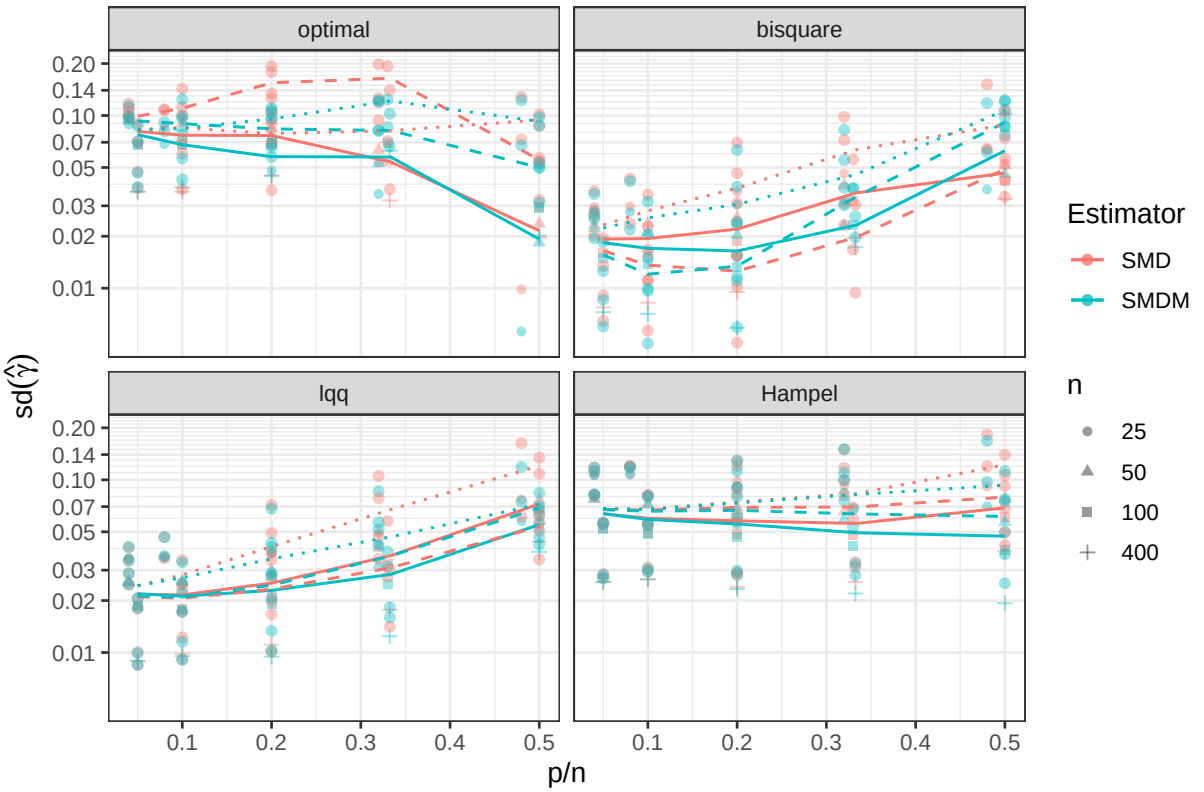


Figure 13: Comparing the estimates of $\gamma$. The solid line connects the uncorrected estimate, dotted the $\tau$ corrected estimate and dashed Huber's small sample correction.

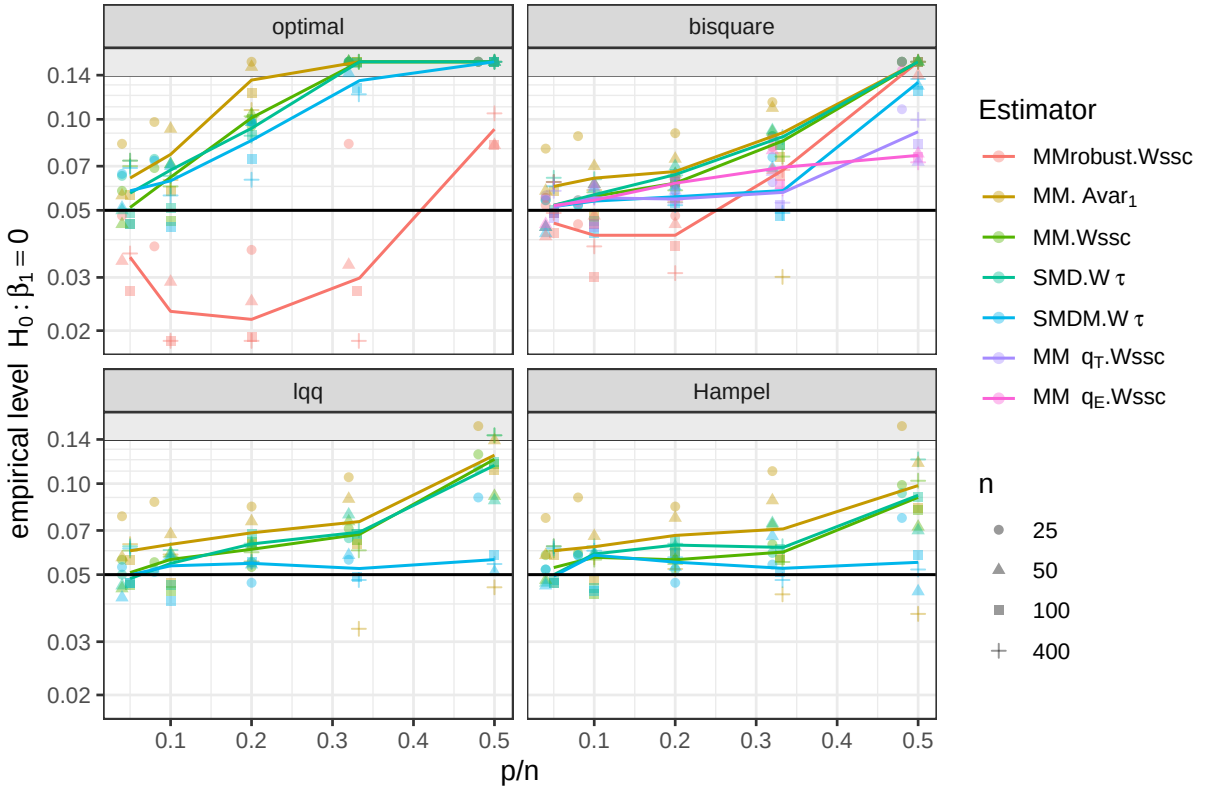


Figure 14: Empirical levels of test $H_0 : \beta_1 = 0$ for normal errors. The y-values are truncated at 0.02 and 0.14.

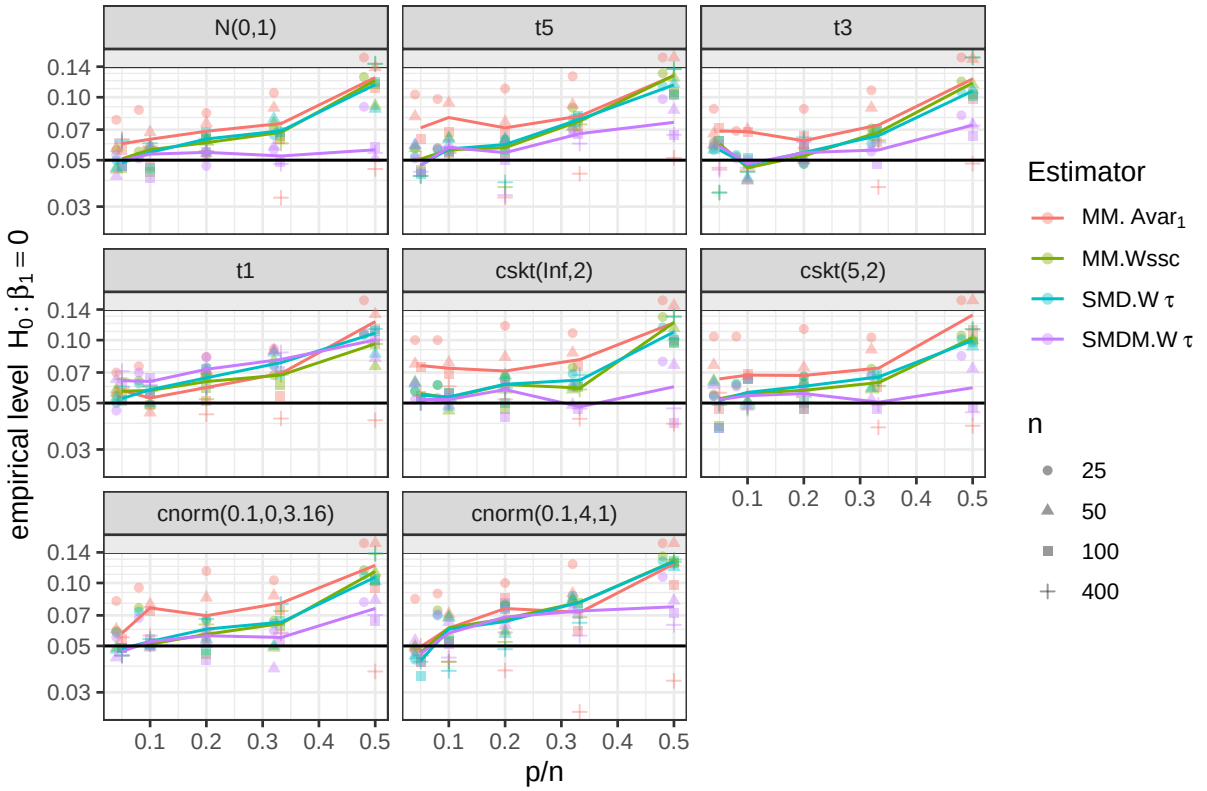


Figure 15: Empirical levels of test $H_0 : \beta_1 = 0$ for *lqq* $\psi$-function and different error distributions.

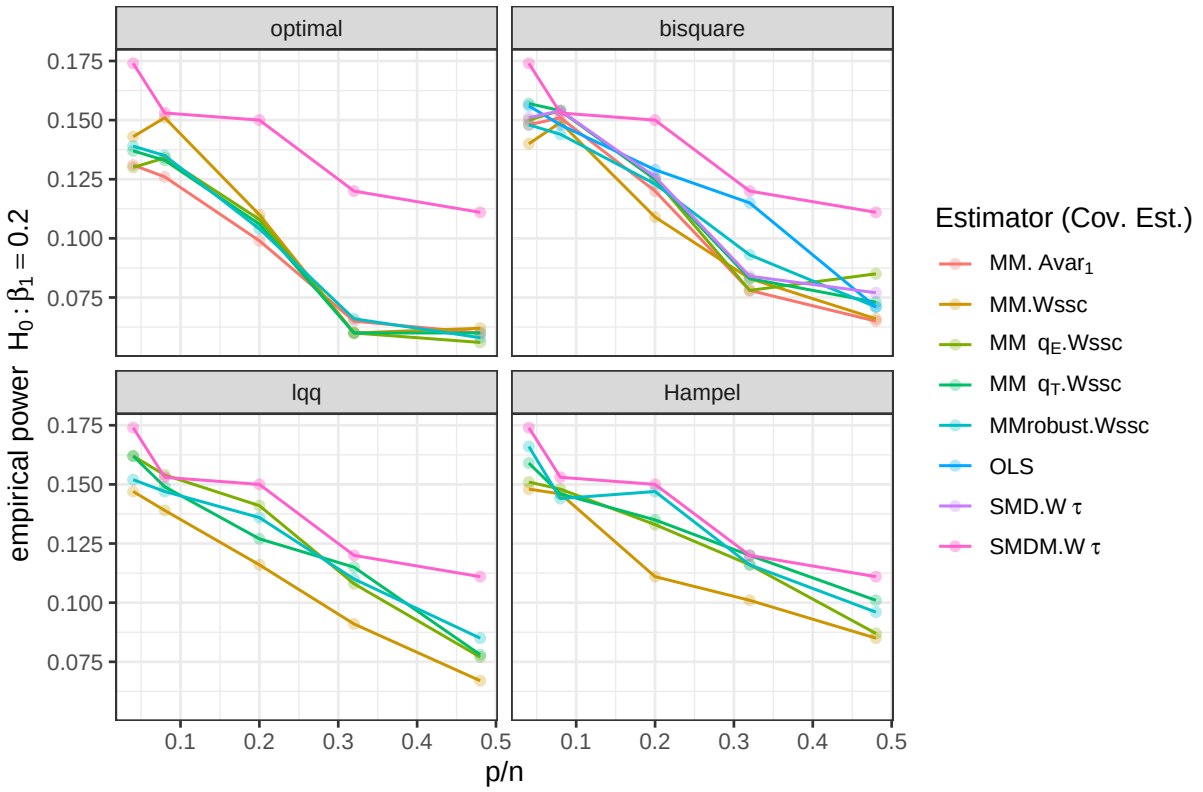


Figure 16: Empirical power of test $H_0 : \beta_1 = 0.2$ for different $\psi$-functions. Results for $n = 25$ and normal errors only.

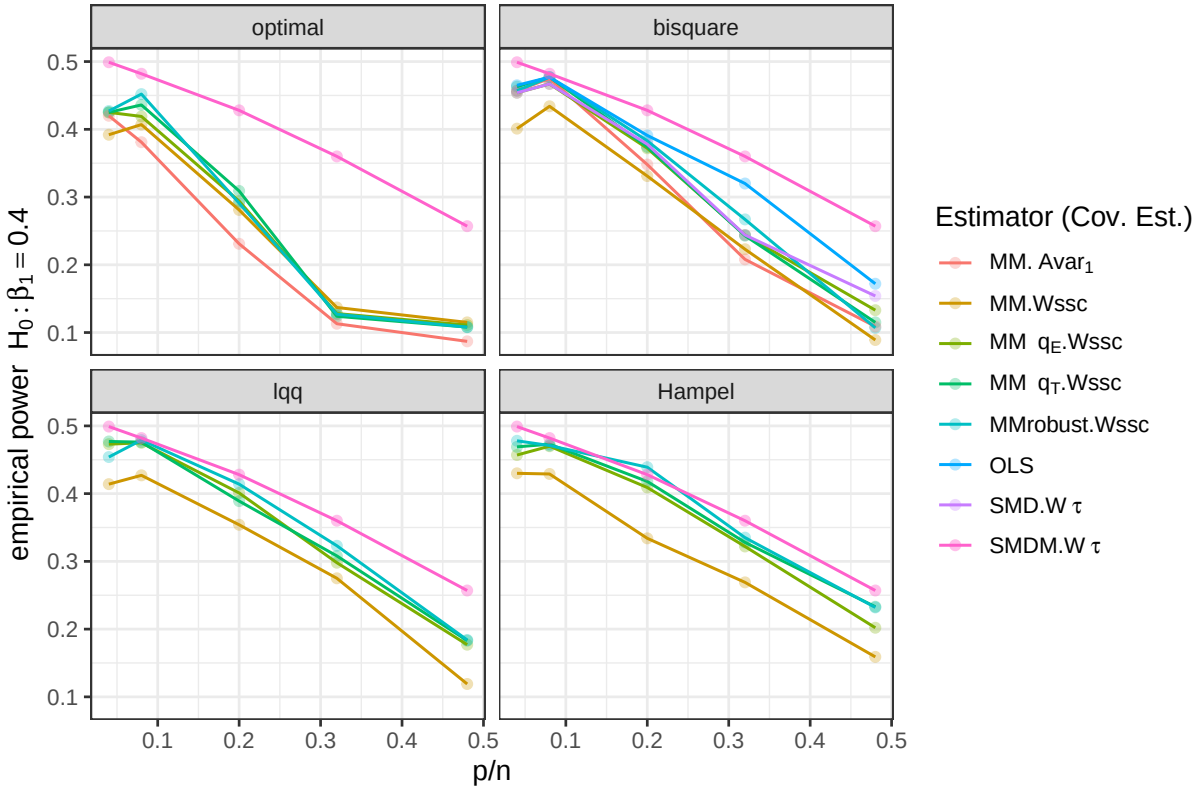


Figure 17: Empirical power of test $H_0 : \beta_1 = 0.4$ for different $\psi$-functions. Results for $n = 25$ and normal errors only.

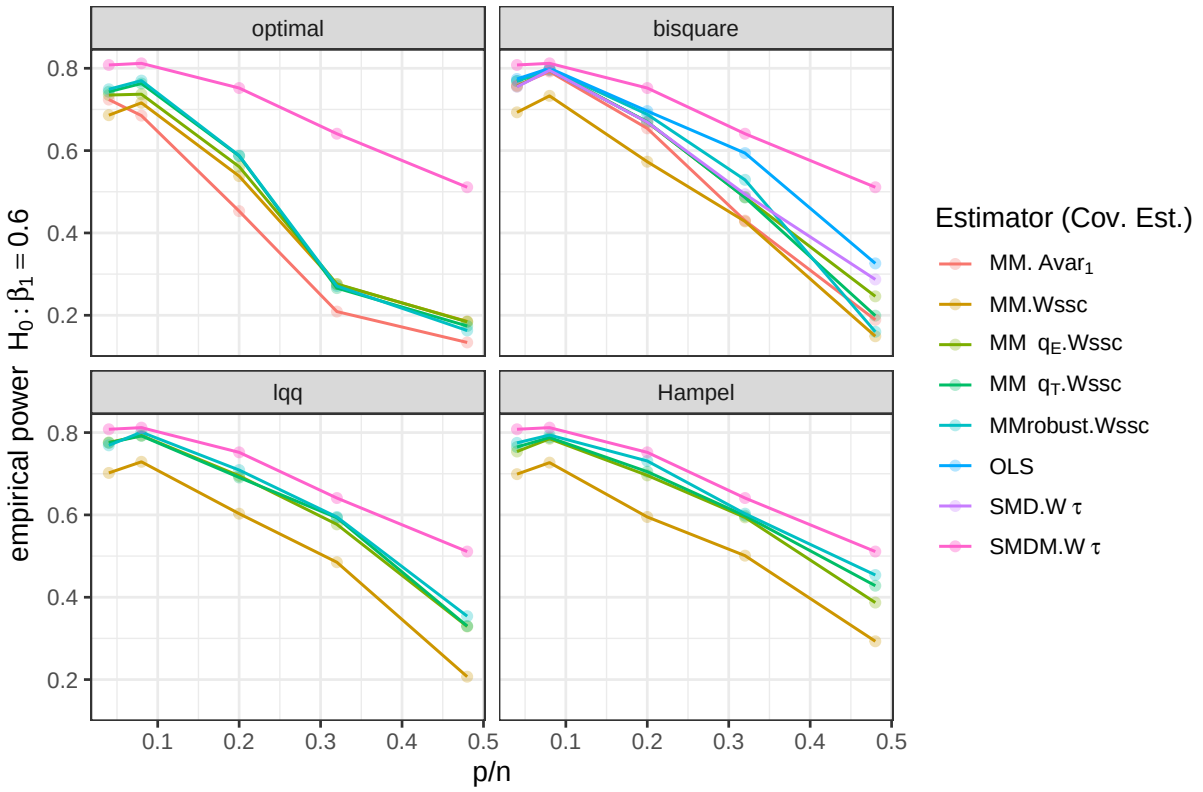


Figure 18: Empirical power of test $H_0 : \beta_1 = 0.6$ for different $\psi$-functions. Results for $n = 25$ and normal errors only.

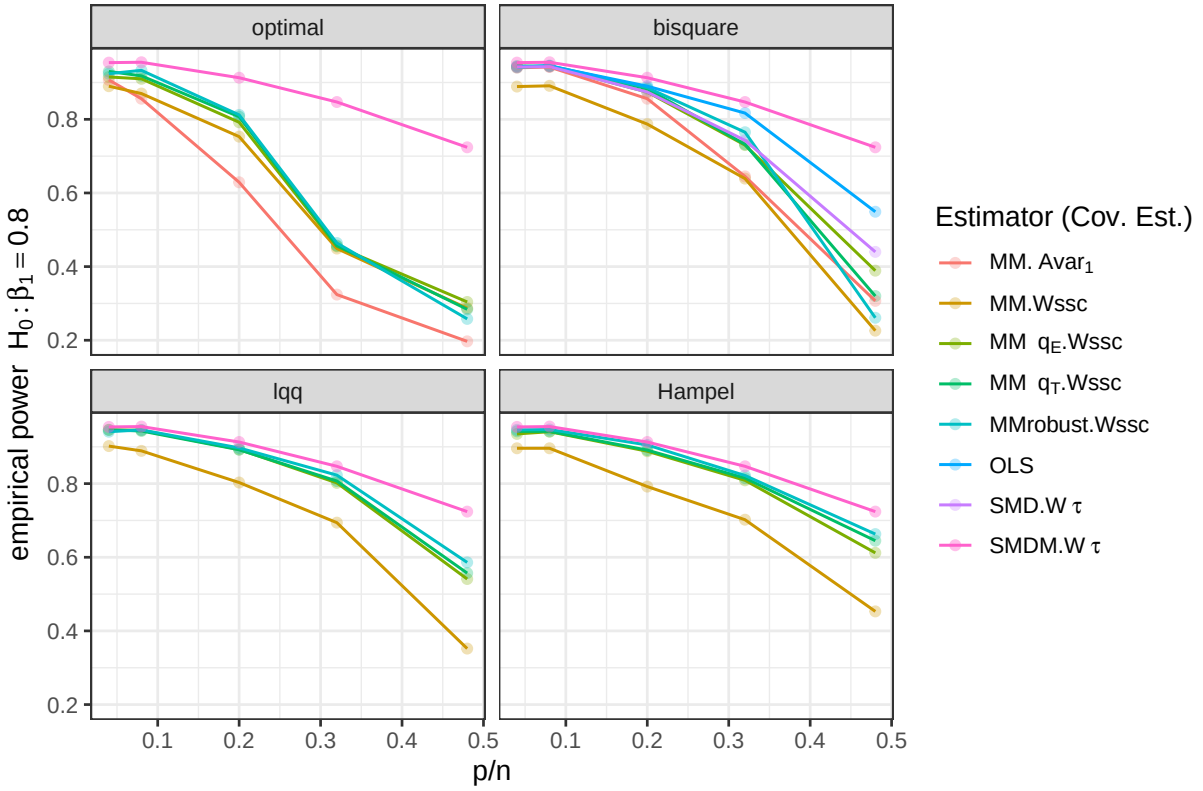


Figure 19: Empirical power of test $H_0 : \beta_1 = 0.8$ for different $\psi$-functions. Results for $n = 25$ and normal errors only.

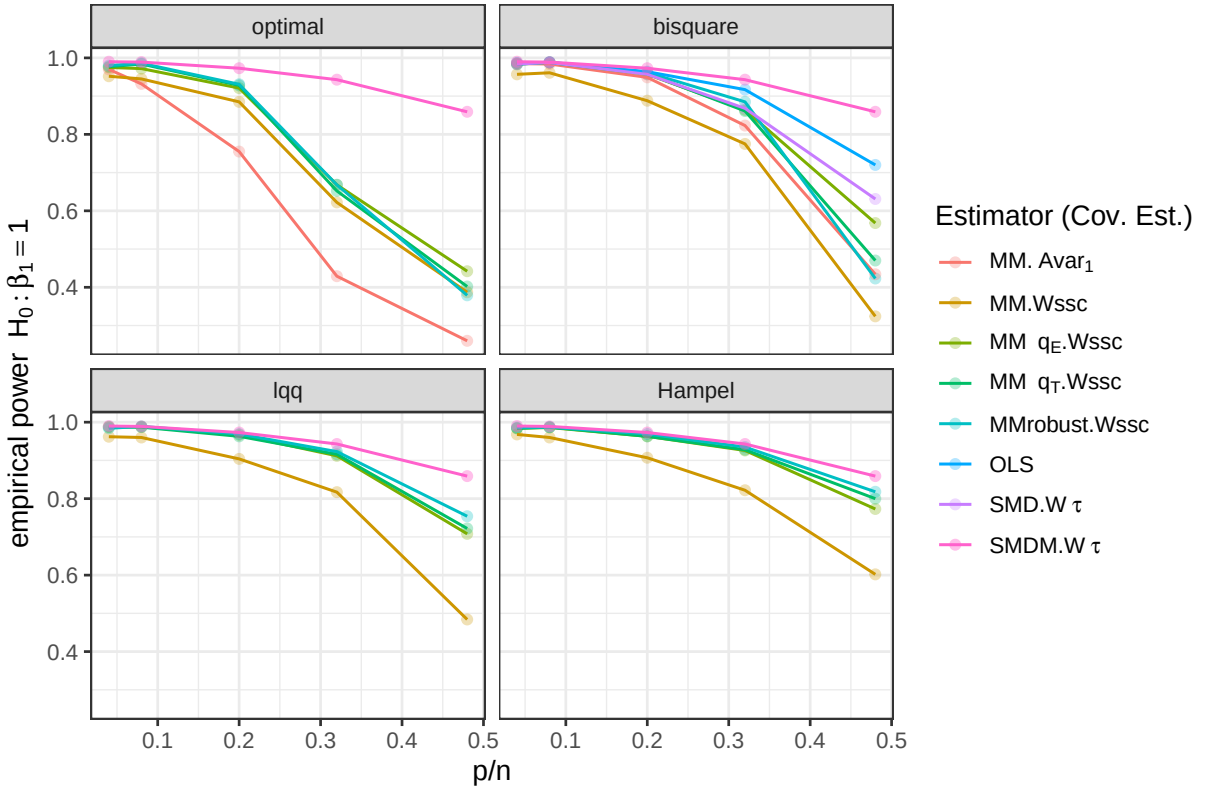


Figure 20: Empirical power of test $H_0 : \beta_1 = 1$ for different $\psi$-functions. Results for $n = 25$ and normal errors only.

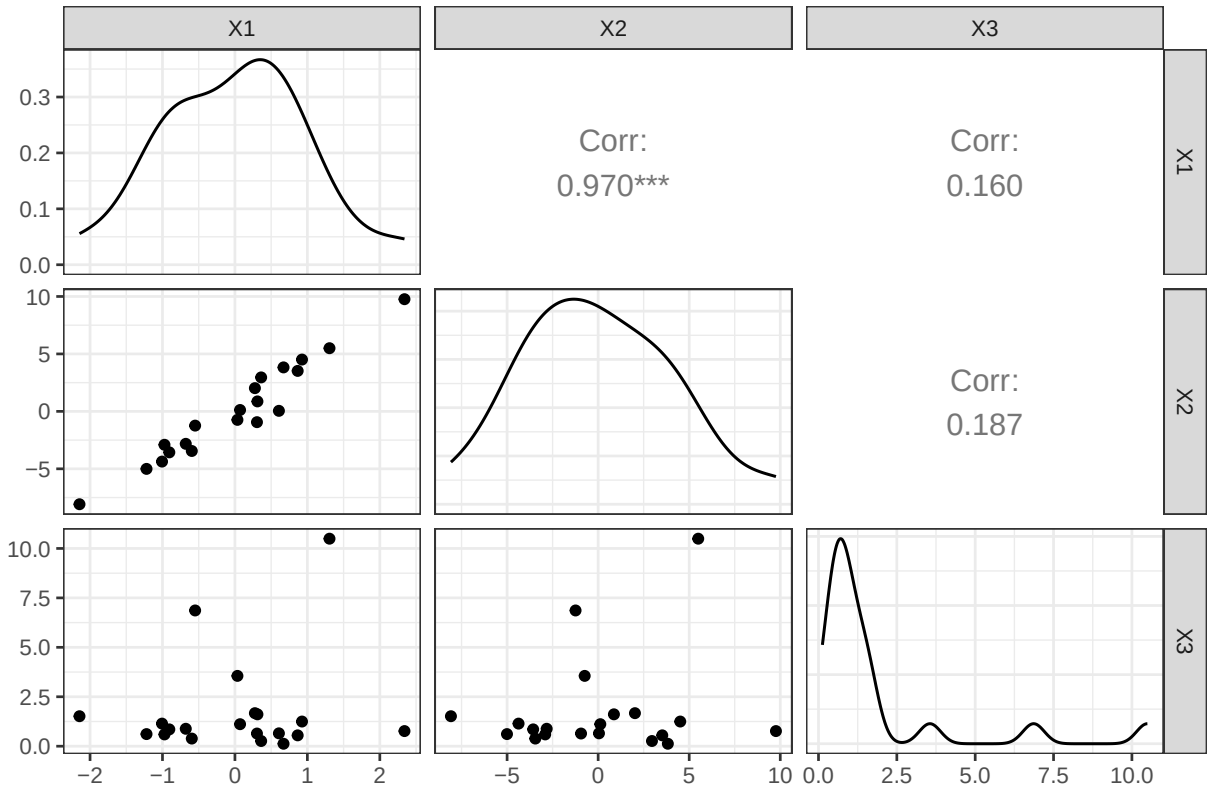


Figure 21: Prediction points for fixed design. The black points are the points of the original design. The red digits indicate the numbers and locations of the points where predictions are taken.

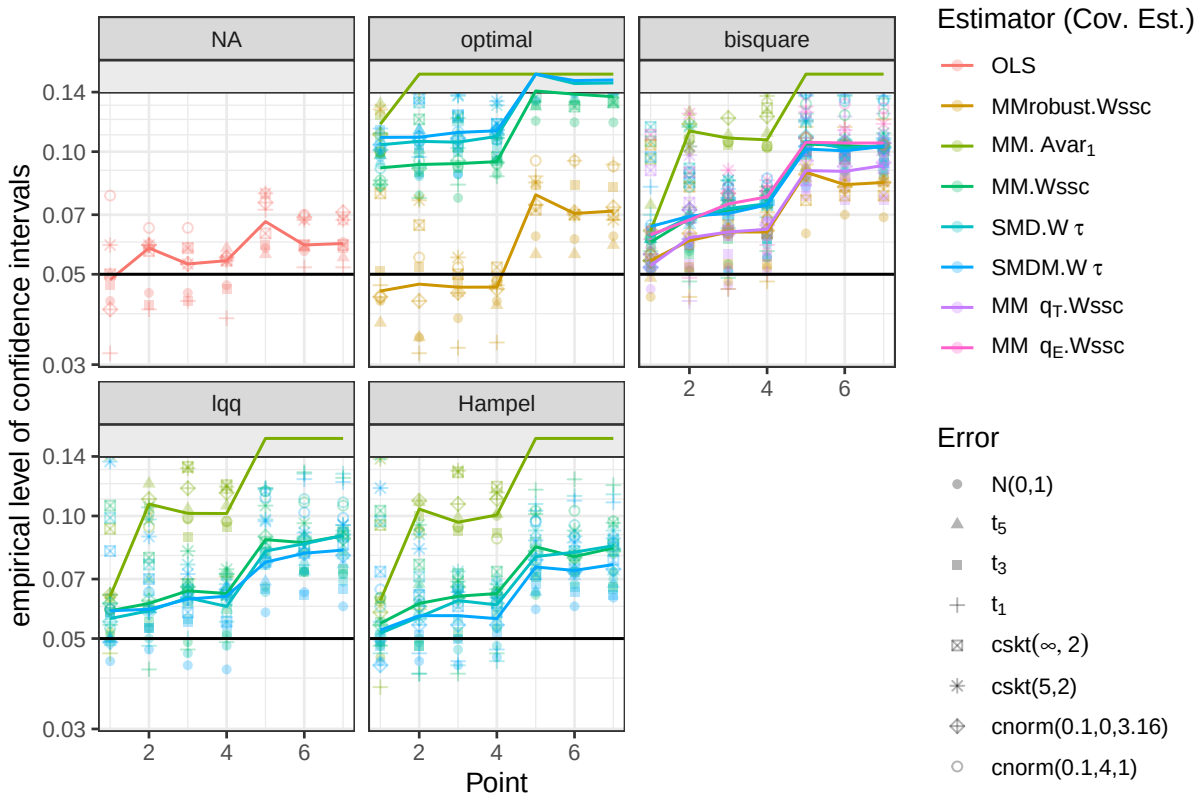


Figure 22: Empirical coverage probabilities. Results for fixed design. The y-values are truncated at 0.14.

## 5 Maximum Asymptotic Bias

The slower redescending $\psi$-functions come with higher asymptotic bias as illustrated in Fig. 23. We calculate the asymptotic bias as in Berrendero et al. (2007).

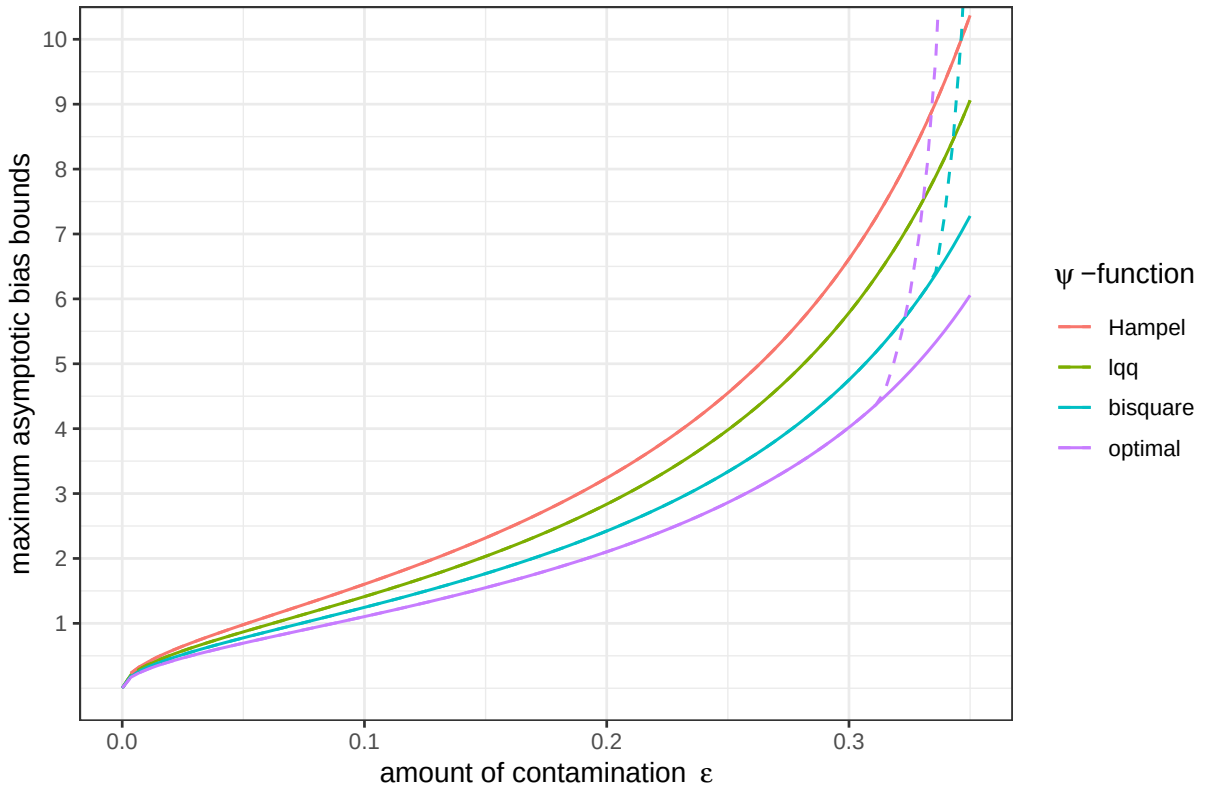


Figure 23: Maximum asymptotic bias bound for the $\psi$-functions used in the simulation. Solid line: lower bound. Dashed line: upper bound.

## References


Berrendero, J., B. Mendes, and D. Tyler (2007). On the maximum bias functions of MM-estimates and constrained M-estimates of regression. *Annals of statistics 35*(1), 13.

Croux, C., G. Dhaene, and D. Hoorelbeke (2003). Robust standard errors for robust estimators. Technical report, Dept. of Applied Economics, K.U. Leuven.

Fernández, C. and M. Steel (1998). On bayesian modeling of fat tails and skewness. *Journal of the American Statistical Association 93*(441), 359–371.

Huber, P. J. and E. M. Ronchetti (2009). *Robust Statistics, Second Edition*. NY: Wiley and Sons Inc.

Koller, M. and W. A. Stahel (2011). Sharpening wald-type inference in robust regression for small samples. *Computational Statistics & Data Analysis 55*(8), 2504–2515.

Maronna, R. A. and V. J. Yohai (2010). Correcting MM estimates for "fat" data sets. *Computational Statistics & Data Analysis 54*(12), 3168–3173.