Robust regression with compositional covariates

Aditya Mishra

July 27, 2019

Simulation Model

Features dimension

```
## Define parameters to simulate example
##
p <- 80                                 # number of predictors
n <- 300                                # number of sample
O <- 0.15*n                             # number of outlier, e.g. 15% of observation
L <- 1                                  # indicator variable for outlier type,
                                        # L = {0,1} => leveraged {no, yes}

# generate outlier by shifting "O"observation by amount equals to shFac times
# true error variance sigma.
# shFac = {6,8} corresponds to {moderate, high} outlier
shFac <- 6
ngrp <- 4                               # number of sub-composition
snr <- 3                                # Signal to noise ratio
example_seed <- 2*p+1                   # example seed
set.seed(example_seed)
```

Simulate model

```
## Simulate true model variables, i.e., y, X, C, beta
## Follow examples from [Pixu Shi 2016]

# Simulate subcomposition matrix
C1 <- matrix(0,ngrp,23)
tind <- c(0,10,16,20,23)
for(ii in 1:ngrp)
  C1[ii,(tind[ii]+1):tind[ii+1]] <- 1
C <- matrix(0,ngrp,p)
C[,1:ncol(C1)] <- C1


# model parameter beta
beta <- c(1, - 0.8, 0.4, 0, 0, - 0.6, 0, 0, 0, 0, -1.5,
          0, 1.2, 0, 0, 0.3)
beta <- c(beta,rep(0,p-length(beta)))

# Simulate response and predictor, i.e., X, y
Sigma  <- 1:p %>% outer(.,.,'-') %>% abs(); Sigma  <- 0.5^Sigma
data.case <- vector("list",1)
data.case <- robregcc_sim(n,beta,O = O,Sigma,levg = L, snr,shft = shFac,0,
                          C,out=data.case)
```

Data preprocessing

```r
X <- data.case$X                          # predictor matrix
y <- data.case$y                          # model response
#

# Predictor transformation due to compositional constraint:
# Equivalent to performing centered log-ratio transform
Xt <- svd(t(C))$u %>% tcrossprod() %>% subtract(diag(p),.) %>%
  crossprod(t(X),.)
Xm <- colMeans(Xt)
Xt <- scale(Xt,Xm,FALSE)                  # centering of predictors
#
mean.y <- mean(y)
y <- y - mean.y                           # centering of response
#
# Account for intercept in the model
Xt <- cbind(1,Xt)                         # accounting for intercept in predictor
C <- cbind(0,C)                           # accounting for intercept in constraint
bw <- c(0,rep(1,p))                       # weight matrix to not penalize intercept
```

Robust regression with compositional covariates

Initialization

Model fitting

```r
# control parameters
control <- robregcc_option()
beta.wt <- fit.init$betaR           # Set weight for model parameter beta
beta.wt[1] <- 0
control$gamma = 2                   # gamma for constructing  weighted penalty
control$spb = 40/p                  # fraction of maximum non-zero model parameter beta
control$outMiter = 1000             # Outer loop iteration
control$inMiter = 3000              # Inner loop iteration
control$nlam = 50                   # Number of tuning parameter lambda to be explored
control$lmaxfac = 1                 # Parameter for constructing sequence of lambda
control$lminfac = 1e-8              # Parameter for constructing sequence of lambda
control$tol = 1e-20;                # tolrence parameter for converging [inner  loop]
control$out.tol = 1e-16             # tolerence parameter for convergence [outer loop]
control$kfold = 5                   # number of fold of crossvalidation


# Robust regression using adaptive lasso penalty
fit.ada <- robregcc_sp(Xt,y,C, beta.init=fit.init$betaR,
                       gamma.init = fit.init$residualR,
                       beta.wt=abs(beta.wt),
                       gamma.wt = abs(fit.init$residualR),
                       control = control,
                       penalty.index = 1, alpha = 0.95)
```

```
## [1] 1
```

```
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

```r
# Robust regression using lasso penalty [Huber equivalent]
fit.soft <- robregcc_sp(Xt,y,C, beta.init=NULL, gamma.init = NULL,
                        beta.wt=bw, gamma.wt = NULL,
                        control = control, penalty.index = 2,
                        alpha = 0.95)
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

```r
# Robust regression using hard thresholding penalty
control$lmaxfac = 1e2              # Parameter for constructing sequence of lambda
control$lminfac = 1e-3             # Parameter for constructing sequence of lambda
fit.hard <- robregcc_sp(Xt,y,C, beta.init=fit.init$betaf,
                        gamma.init = fit.init$residuals,
                        beta.wt=bw, gamma.wt = NULL,
                        control = control, penalty.index = 3,
                        alpha = 0.95)
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

Extract model parameter estimate

```r
## Extract fitted model parameters

# coefficient estimate: [adaptive]
coef_cc(fit.ada, type = 0, s = 1)
```

```
##  [1]  0.143896075  0.972849932 -0.528317343  0.000000000  0.000000000
##  [6]  0.000000000 -0.444532590  0.000000000  0.000000000  0.000000000
## [11]  0.000000000 -1.465779576  0.000000000  1.188646647  0.000000000
## [16]  0.112160931  0.164971997  0.000000000  0.000000000  0.000000000
## [21]  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
## [26]  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
## [31]  0.000000000  0.000000000  0.000000000  0.000000000 -0.010898183
## [36]  0.000000000  0.000000000  0.000000000 -0.047603035  0.000000000
## [41]  0.015747351  0.000000000  0.000000000  0.000000000  0.000000000
## [46]  0.000000000 -0.077243287  0.000000000  0.000000000  0.000000000
## [51]  0.000000000  0.000000000  0.000000000  0.000000000 -0.059318072
## [56]  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
## [61]  0.000000000  0.000000000  0.000000000 -0.035620761  0.000000000
## [66]  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
## [71]  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
## [76]  0.000000000  0.000000000  0.021902559  0.004703247  0.000000000
```

```
## [81]  0.000000000
```

```r
# coefficient estimate: [lasso/Huber]
coef_cc(fit.soft, type = 0, s = 1)
```

```
##  [1] -0.220233559  0.294750101 -0.028056853  0.001776788  0.089587734
##  [6]  0.000000000 -0.358057770  0.000000000  0.000000000  0.000000000
## [11]  0.000000000 -1.695577345  0.000000000  1.299709431  0.019578010
## [16]  0.031631874  0.344658029 -0.236260190  0.236260190  0.000000000
## [21]  0.000000000 -0.239615630  0.231229074  0.008386555  0.000000000
## [26]  0.000000000  0.000000000  0.010347870  0.000000000  0.000000000
## [31]  0.000000000  0.000000000  0.000000000  0.002244197  0.000000000
## [36]  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
## [41]  0.000000000  0.000000000  0.000000000  0.000000000  0.030438627
## [46]  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
## [51]  0.000000000  0.011495797  0.000000000  0.000000000  0.000000000
## [56]  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
## [61]  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
## [66]  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
## [71]  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
## [76]  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
## [81]  0.000000000
```

```r
# coefficient estimate: [Hard]
coef_cc(fit.hard, type = 0, s = 1)
```

```
##  [1]  0.05663394  0.50175615 -0.16192083  0.00000000  0.00000000
##  [6]  0.00000000 -0.33983533  0.00000000  0.00000000  0.00000000
## [11]  0.00000000 -1.24474549  0.00000000  1.24474549  0.00000000
## [16]  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000
## [21]  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000
## [26]  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000
## [31]  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000
## [36]  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000
## [41]  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000
## [46]  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000
## [51]  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000
## [56]  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000
## [61]  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000
## [66]  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000
## [71]  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000
## [76]  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000
## [81]  0.00000000
```

```r
# residual estimate: [adaptive]
residuals(fit.ada)
```

```
##   [1] -7.127198827 -6.153017783 -7.538430409 -7.251503446 -7.551687996
##   [6] -6.598315396 -6.934506163 -7.041948057 -8.766207104 -5.234250321
##  [11] -7.155413413 -7.473786649 -6.664553006 -6.101692245 -6.425264528
##  [16] -6.515295381 -6.766117149 -7.891281249 -7.531725946 -8.838028669
##  [21] -6.176726768 -6.645999880  5.655202234  8.552173578  6.354627019
##  [26]  5.116707510  6.378111456  6.747890282  6.095536798 -6.110464430
##  [31]  6.099631428  6.269240943  6.878067193  6.101677545  5.315983813
##  [36]  5.768200997  5.544940825  7.210900815  5.457906562 -6.580311395
##  [41]  7.739856335  6.131657954  6.661334504  4.983687182  5.658793871
```

```
##  [46] -0.186790046  0.289799794 -0.495296612 -0.021234634  1.054165436
##  [51]  0.923030562 -0.428613430  0.673669466  1.444122436  0.316349467
##  [56]  0.168798067 -0.543381984  0.879052532 -0.135393074  0.012214881
##  [61] -0.232416117  0.930417701  0.263675486  1.647011103 -1.704706692
##  [66] -0.137672917  0.093087327 -0.264414159  0.901571731  0.368859912
##  [71] -0.504944057  0.641560428 -1.173413354  1.340554815  0.011788315
##  [76] -0.372154353 -0.575294875  0.753533173  0.555587675  1.279166730
##  [81]  1.117994580  0.527941776 -0.525528965 -1.481163032  0.259733834
##  [86] -0.607621567 -0.462422697 -1.803439412  0.749374701  0.998772287
##  [91] -2.109786081 -1.106145600 -0.118958360 -1.365015478  1.927754620
##  [96] -2.009261762  0.181225784 -0.092054219 -1.545575189 -0.838597405
## [101]  0.347888141  1.170224279  1.122363370  0.268680715 -0.561498927
## [106]  0.172720905 -2.123331150  0.281539563  0.330623242  0.253842029
## [111] -0.552206616  0.916670399 -0.138494677 -2.406710731 -0.528923697
## [116]  2.098834934  1.372860459 -0.214536584  1.450718387  0.606826241
## [121] -0.437575043  1.535481013 -0.242510485  1.698438814  1.342947343
## [126] -2.216187285 -0.883503085  0.086109773 -1.079257232 -0.134062799
## [131]  0.702156357 -0.154765253 -0.318302219 -0.173489163  0.303511363
## [136] -0.585230760 -1.291318074 -0.199365780 -0.699494041  0.191469184
## [141] -0.557320111  0.205070843 -0.222538690 -0.422985404  1.453244090
## [146] -0.451600631 -0.456457527  0.170163036  0.256446183 -0.118085355
## [151] -0.593361111  1.108150173  0.258773035 -0.750963530 -0.385301997
## [156] -1.099753045  1.392453946 -2.192821752 -0.604020109 -0.966725933
## [161] -0.957944602 -0.015201814 -1.191154191  1.183343980 -0.268681733
## [166]  0.784755831 -0.208816513  0.710534431  0.173103451  1.272500845
## [171] -1.940452091  0.453100171 -0.332109553  1.054110850  0.178017255
## [176]  1.310516216  1.732893423 -1.594924716 -1.441358436  0.910061364
## [181] -0.304885796  1.067487886 -0.239370852  0.460637898 -0.995082570
## [186] -1.582885233 -0.157086974 -0.355556366  1.609410535 -0.167286751
## [191]  1.471632769 -2.701941563 -0.538279886  0.894824877 -0.368292356
## [196]  1.075723541  0.676353812 -1.122458759 -0.020053607 -0.555245180
## [201]  0.486750537 -0.658018515  0.241986055  0.766059355  0.004069136
## [206]  0.073343136 -0.338329245 -1.151946146 -2.048498396 -0.462038408
## [211]  0.004113488  0.484134693  1.380110366 -2.471275014 -0.583659559
## [216] -0.725205099 -0.724173260 -0.355875499 -0.521072385 -0.278084272
## [221] -0.540856873  0.157157952  1.511154927 -0.339889264  1.626419925
## [226]  1.035731021 -0.237305725 -0.701689075 -1.607178149 -0.212593939
## [231]  0.639274994 -1.785018109 -0.127395315  0.206186388  0.550070689
## [236] -0.749179976  0.796637511 -0.520247480 -0.341263304  1.336536322
## [241]  0.396968904  0.680060807 -0.150568559  0.397569131 -0.388801615
## [246]  1.281634605 -0.898122658 -0.532270605 -1.145978468  0.131020273
## [251]  0.128514045  0.714771720 -0.647492833 -0.498085852 -1.573279863
## [256]  1.707235324 -0.390338967  0.716662305  0.759609167  0.451422029
## [261] -0.274275165 -1.282562836 -0.460219909  0.295985473 -0.653460052
## [266] -1.277529972  0.240143686 -0.660175105 -0.184976269 -0.569421628
## [271]  0.111046168 -1.296249374  1.190543396  1.791859973 -0.322178616
## [276]  0.506024643  0.114554221  0.204309197 -0.153168791 -0.327986999
## [281] -1.260475200  0.500371017  1.606177239 -1.183871446 -0.004628462
## [286] -0.301498239  1.579943910  0.575801427  1.037453005  1.193093450
## [291]  0.379304348 -0.244258538  1.246469119 -0.227134030  0.417834258
## [296]  1.749848755 -2.377025837 -0.111275513 -0.369148373  0.259092364
```

```r
# residual estimate: [lasso/Huber]
residuals(fit.soft)
```

```
##   [1]  1.87208424  1.44298945 -0.45483807 -0.33159643 -0.98583251
##   [6] -0.16882227 -0.69913867 -0.98966806 -2.79800369  0.64958928
##  [11] -1.40596179 -1.80050204 -1.22446370 -0.73550120 -1.17925738
##  [16] -1.37217346 -1.72089669 -2.97770887 -2.67136574 -4.09292733
##  [21] -1.52958368 -2.06678712  3.72339111  8.01746093  7.46208724
##  [26]  4.11053673  6.44809743  6.78148722  5.04807966 -7.05291867
##  [31]  6.53450892  5.32377899  6.91552808  5.61252310  4.42613596
##  [36]  6.70300677  4.98122569  7.67261454  4.81239607 -5.61207462
##  [41]  6.14550942  5.84513497  7.07036667  5.27654154  4.79190514
##  [46] -0.19089747  0.35450192 -1.89918168 -0.05828562  1.82526385
##  [51]  0.75762739  0.51495254  2.60200753  1.80071927  0.25277411
##  [56] -0.43104698 -0.06982209  0.62045865  1.10245830  1.06665157
##  [61]  0.36546068 -0.59513726 -1.16232102  0.67393892 -1.79167214
##  [66] -0.13307461  0.57312037 -1.66892117  0.94823822  0.59980461
##  [71]  0.04752422 -0.04985644 -3.08027229  0.87498401  0.17831750
##  [76] -1.18166941 -0.83190218  0.34736315  1.70502064  2.54892421
##  [81]  0.62187643 -0.29478166 -3.26116085 -2.45622893  0.56776494
##  [86] -0.80609131 -0.46784193 -2.51247941  0.46223878  0.28438233
##  [91] -1.74211285 -0.63233366  0.43043719 -1.81334925  0.95247531
##  [96] -1.37986450  0.28427408  0.18526193 -1.54822960 -1.24210655
## [101]  0.40946346  0.67914137  1.11223927  0.25426986 -1.26814452
## [106]  0.61645537 -1.54678847  0.07238729 -0.18804906  0.68037749
## [111]  1.25565688  1.27259797  0.78014932 -2.73673435 -0.27367461
## [116]  2.45945704  1.72317212 -0.93882670  1.29414970  0.49618915
## [121] -0.48698983  1.40856237  0.51121224  2.10362683  2.20966023
## [126] -1.68863989 -0.68637170 -0.49619800 -2.01332559  0.57380058
## [131]  1.06822407  1.13864653 -0.32239204 -0.32667579 -0.18193187
## [136] -0.67176763 -2.66160411 -0.11260313 -0.86171434 -0.46721715
## [141] -0.52138359  0.78834919  0.44756001 -0.79838299  2.37867313
## [146] -2.10528421 -0.48882572  1.04244741 -0.90064256 -0.88598614
## [151] -0.77511143  0.10859662 -0.16546569 -1.00597718 -1.71453931
## [156] -0.44042989  1.49268218 -2.06656215 -0.37774196 -0.63799522
## [161] -1.18609884  0.15722708 -1.56408218  0.85561172 -1.49178028
## [166]  0.40635539 -0.86647442  1.40102927 -0.23472852  0.80780702
## [171] -1.31518506 -0.05107348 -0.43755112 -0.39154416  0.32394683
## [176]  1.85342416  1.85312832 -0.60763319 -1.78436904  1.21386160
## [181] -0.09324453  2.10001903  0.67709514 -0.16856897 -1.57985959
## [186] -2.82308737  0.82471646 -0.94653120  2.74875978  0.57210240
## [191]  1.24650038 -2.47064894 -0.50313786 -0.87604305 -0.94413725
## [196]  1.19712466  0.13464217 -2.11540722  1.01506385 -0.65470605
## [201] -0.34238934  0.18240817 -0.82436236  2.75187916 -0.91930743
## [206]  0.29666157 -0.43142950 -0.01336912 -2.03026264  1.02025225
## [211]  0.90995264 -0.07951866 -0.03967036 -3.13285612  0.75194558
## [216] -1.19435860 -0.36973851 -1.44960997 -0.67174074 -0.23179145
## [221] -0.97551559 -0.24814275  1.19569951 -0.97025605  1.65138364
## [226]  1.38998076  0.12009773 -1.67910296 -1.52086173 -0.02031689
## [231]  0.68775680 -2.13193115  0.13367268  0.82099611  0.80450592
## [236] -0.68931259  0.26272608  0.14891077 -0.20135804  1.81303656
## [241] -0.12028762 -0.06838852  1.18237249  0.66072560  0.43564144
## [246]  1.23900763 -0.64956524 -0.70083356 -1.83299066 -0.68175251
## [251] -0.80982327  0.25156445 -0.88639929  0.90308584 -2.67668052
## [256]  1.08230319 -1.24617637  0.83805765  0.18024025  1.20804478
## [261] -1.31606317 -2.36310718 -0.64805018  0.84103585 -1.78242946
## [266] -0.52217696  0.37340552 -1.00829058  0.51732319 -0.56218178
```

```
## [271]  0.57114448 -1.91961736  0.89291094  1.25269802 -0.61583560
## [276] -0.35703733  1.10567153  0.58265258  0.36461563 -0.95833588
## [281] -0.92415943  0.07090157  1.30411011 -0.42562155  0.01283387
## [286] -0.22316406  1.40029489  0.53679195  0.49668104  2.11014336
## [291]  0.74795006 -0.23739309  1.59113094  0.38308521  0.25357583
## [296]  1.87570689 -2.00355850 -0.79156923 -0.24279914  0.71255278
```

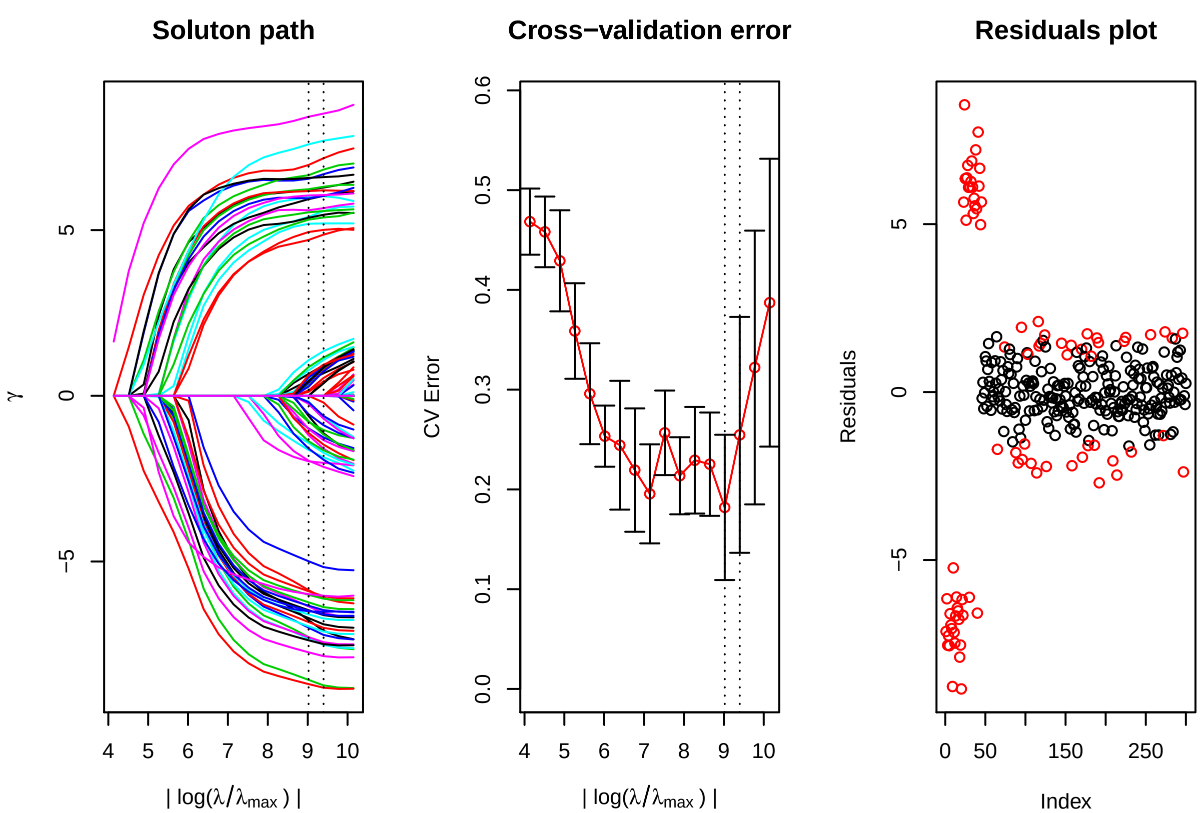
```r
# residual estimate: [Hard]
residuals(fit.hard)
```

```
##   [1] -4.96409001 -4.45683057 -6.00193562 -5.77776484 -6.25079240
##   [6] -5.28214809 -5.68208851 -5.83360779 -7.57491692 -4.07064990
##  [11] -6.01570376 -6.34642339 -5.65439564 -5.10901290 -5.47356750
##  [16] -5.59779180 -5.89093370 -7.06254667 -6.71095184 -8.06724817
##  [21] -5.43133472 -5.92066314  5.22311427  8.58726212  6.25923022
##  [26]  4.62171279  6.57680979  6.90119877  6.15262327 -7.14759273
##  [31]  5.83245322  6.45100952  7.84568140  6.17888737  4.75542441
##  [36]  7.30357866  5.49962532  7.69233732  5.04519113 -6.49368165
##  [41]  7.37410177  5.73702425  7.32143614  5.14070356  5.81332139
##  [46]  0.39627073  0.78052758 -0.84052343 -0.23658915  0.87447741
##  [51]  0.89149460 -0.64164056  1.63350709  1.34676887  0.60683180
##  [56]  0.36453182  0.32674966  0.77643401  0.29872035 -0.26846847
##  [61] -1.05841155  0.68770725 -0.06980770  1.24541151 -2.58418607
##  [66] -0.26700861  0.83294091 -1.18185092  1.17727654  0.12975850
##  [71] -0.19667806  0.17229376 -2.09013065  2.03077569 -0.29470125
##  [76] -1.75346968 -0.66958746  0.39138321  1.35602060  1.46806445
##  [81]  1.42689813 -0.05988210 -0.97670016 -1.52785000 -0.12740980
##  [86] -0.22179673 -0.37280352 -2.65412643 -0.36917380  0.76111057
##  [91] -1.66181102 -0.69843209 -0.24841627 -1.61005130  1.49979707
##  [96] -1.60091175  0.59749966  0.01203849 -2.21987575 -0.93473837
## [101] -0.03023850  1.76628346  2.25831999  0.65267140 -1.53443296
## [106] -0.29001342 -3.09303348  0.88733070 -0.32437234  0.06958821
## [111]  0.41427352  0.93576945 -0.14409899 -3.48047048 -0.46730738
## [116]  1.86088907  1.59247810 -1.24720660  2.25102362  0.67207067
## [121] -0.63469459  1.09550945  0.26788363  1.85558567  1.46383002
## [126] -1.50254369 -0.68651802  0.12069772 -1.60650060  0.30831561
## [131]  0.42456360  0.46689514 -0.74772824 -0.36458703  0.01521195
## [136] -0.65701994 -1.58772311 -1.29357334 -0.76436490  0.01602487
## [141]  0.01046610  0.41933417 -0.08874297 -0.74127678  2.47667426
## [146] -0.72102457  0.28585490  0.17041672 -0.45411292 -0.16783076
## [151] -1.16770794  1.57660881  1.06834805 -0.71057689 -0.77294028
## [156] -0.91570021  1.68035576 -1.71684384 -0.62169794 -1.16768443
## [161] -0.51498834  0.42072134 -1.31012919  1.00285284 -0.87768926
## [166]  0.23117959 -1.07981571  0.87186828 -0.11582194  1.75965317
## [171] -1.45004251  0.22593754 -0.47114205  0.05900993  1.11177916
## [176]  1.44866388  1.44765932 -1.11493457 -1.45261381  1.49772917
## [181] -0.64071239  3.15868586 -0.38112164  0.60694918 -1.61939910
## [186] -2.29220234 -0.02641484 -0.28725113  2.45827435  0.45776759
## [191]  2.59027571 -2.84368276 -0.31566284 -0.30284657 -0.27959665
## [196]  1.47227777  0.69128197 -1.10124356  0.59737186  0.35348923
## [201]  0.73580364 -0.56523331 -0.78291813  1.79633357 -0.27413718
## [206] -0.43975290 -0.76405434 -0.56855614 -2.78557644 -0.12046680
## [211]  0.10500120  0.95167605  1.14336202 -2.16306915  0.19015444
## [216] -0.27132861 -0.41934969 -0.73486050 -1.03787376 -0.75223643
## [221]  0.25939701 -0.09047781  2.11059143 -0.35874951  2.29716735
```

```
## [226]  0.97620746 -0.06443008 -1.30401808 -1.05264132  0.56269879
## [231]  0.60647114 -2.34028656 -0.09827930  0.03114021  0.47464687
## [236] -0.63638466  0.69951796 -0.25027767 -0.72277434  1.75578049
## [241]  0.50572633  0.19035803  0.70661013  0.47828245 -0.25069400
## [246]  1.36055194 -1.45404382 -1.02275164 -2.45321739 -0.72114056
## [251] -0.52511270  0.09291844 -1.12883142 -0.68486304 -1.10699699
## [256]  1.90196815 -0.21127464  0.80626489  1.00394954  1.26989704
## [261] -1.01811627 -2.01016711 -0.09421847  0.69677274 -1.02383221
## [266] -0.54862069  0.27835996 -0.60670296  0.32313116  0.03291781
## [271] -0.70070950 -2.58625223  0.81237528  2.13061122 -0.41469893
## [276] -0.40287708  1.06705691  0.44903584 -0.11354407 -0.10327360
## [281] -1.23489949  0.62095356  1.81678141 -0.43442749  0.07575417
## [286] -0.23963217  1.33535419  1.43434650  0.79993355  2.31460574
## [291]  0.92231131 -0.28891816  0.86278795  0.40711044  0.02234786
## [296]  1.64983253 -2.14444818 -0.91759955 -0.54471226  0.01813460
```

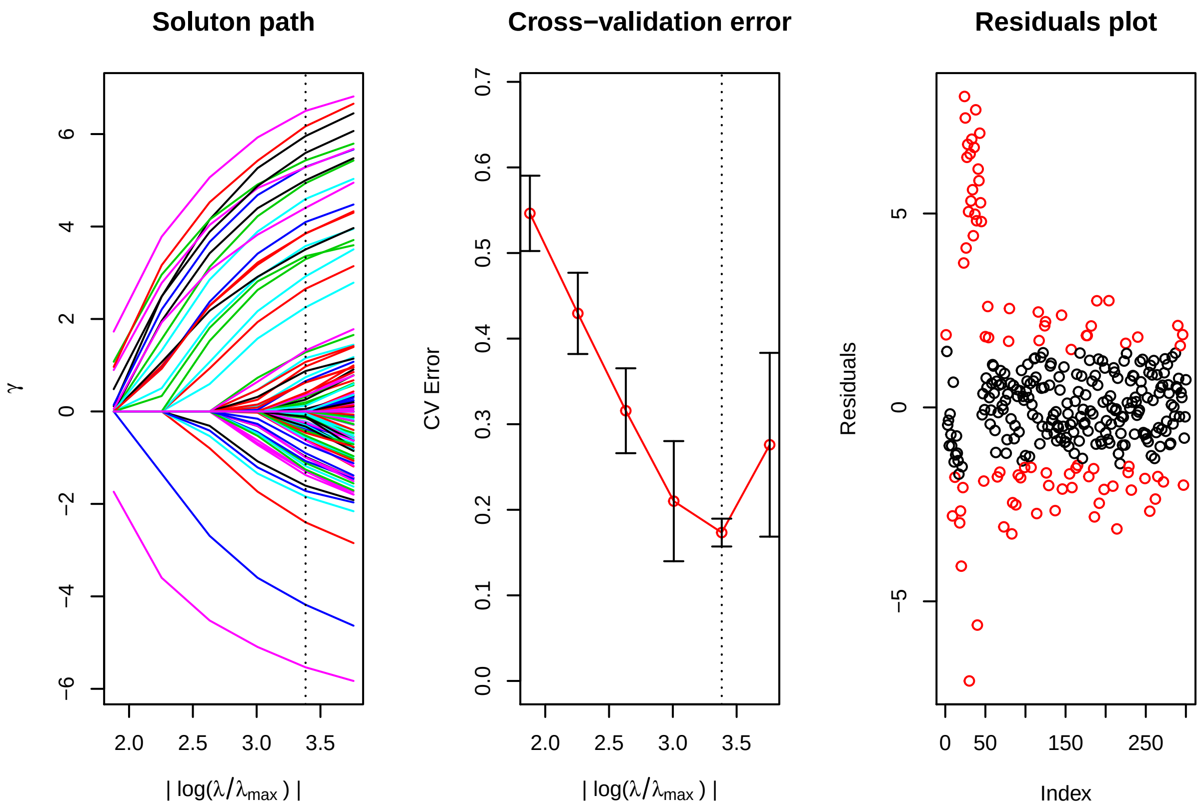
Plot model output

```
# mfrow for multiple plots
# [adaptive]
par(mfrow=c(1,3))
plot_path(fit.ada)
plot_cv(fit.ada)
plot_resid(fit.ada)
```

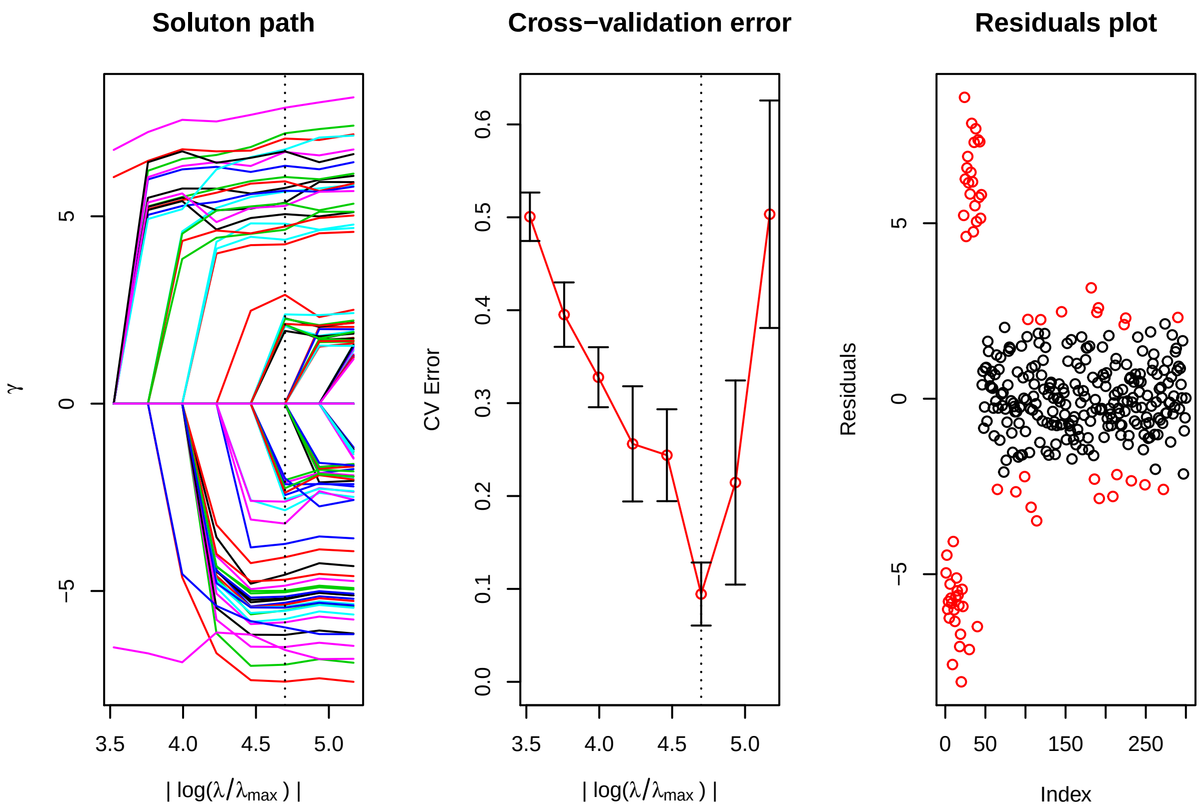



```
# [soft]
par(mfrow=c(1,3))
```

```
plot_path(fit.soft)
plot_cv(fit.soft)
plot_resid(fit.soft)
```



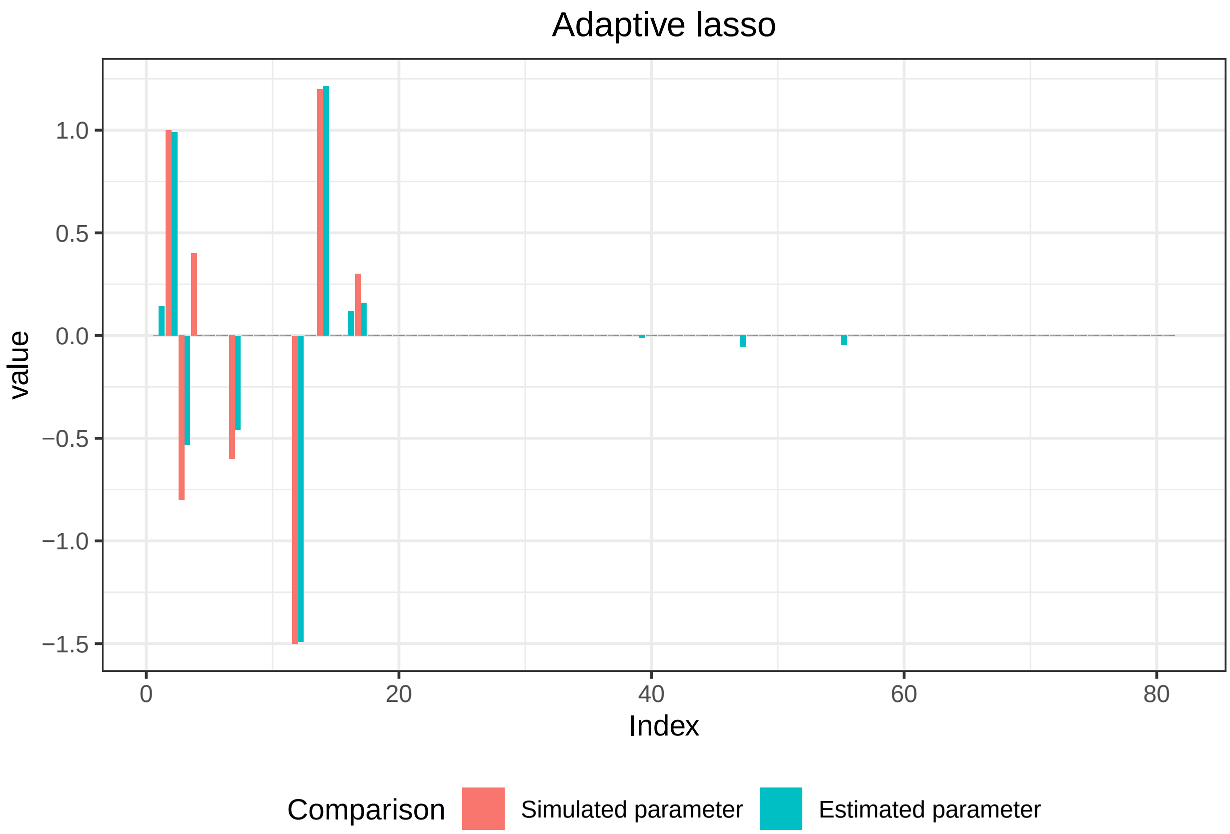

```
#title(sub ='[Soft]: Solution path, Cross-validation error, residual')

# [Hard]
par(mfrow=c(1,3))
plot_path(fit.hard)
plot_cv(fit.hard)
plot_resid(fit.hard)
```



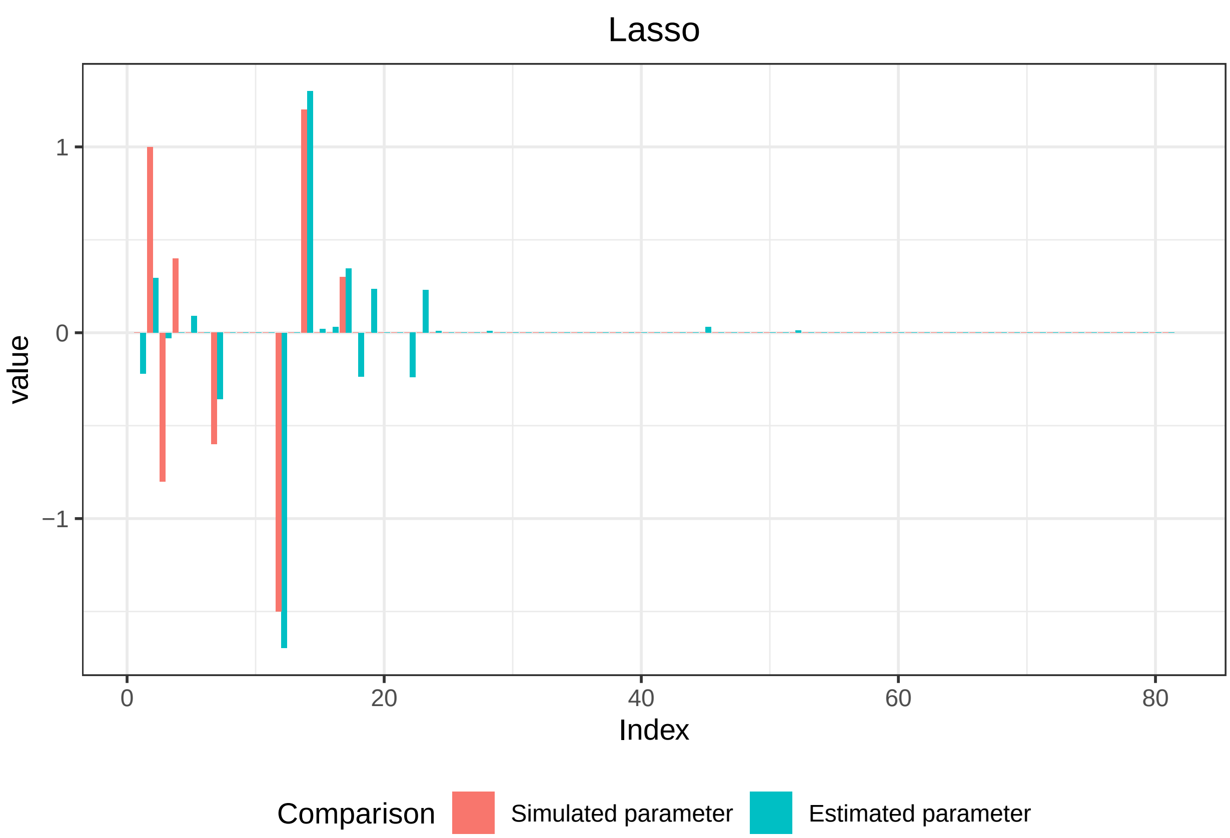

```
#title(sub ='[Hard]: Solution path, Cross-validation error, residual')
par(mfrow=c(1,1))
```

Estimated parameter comparison

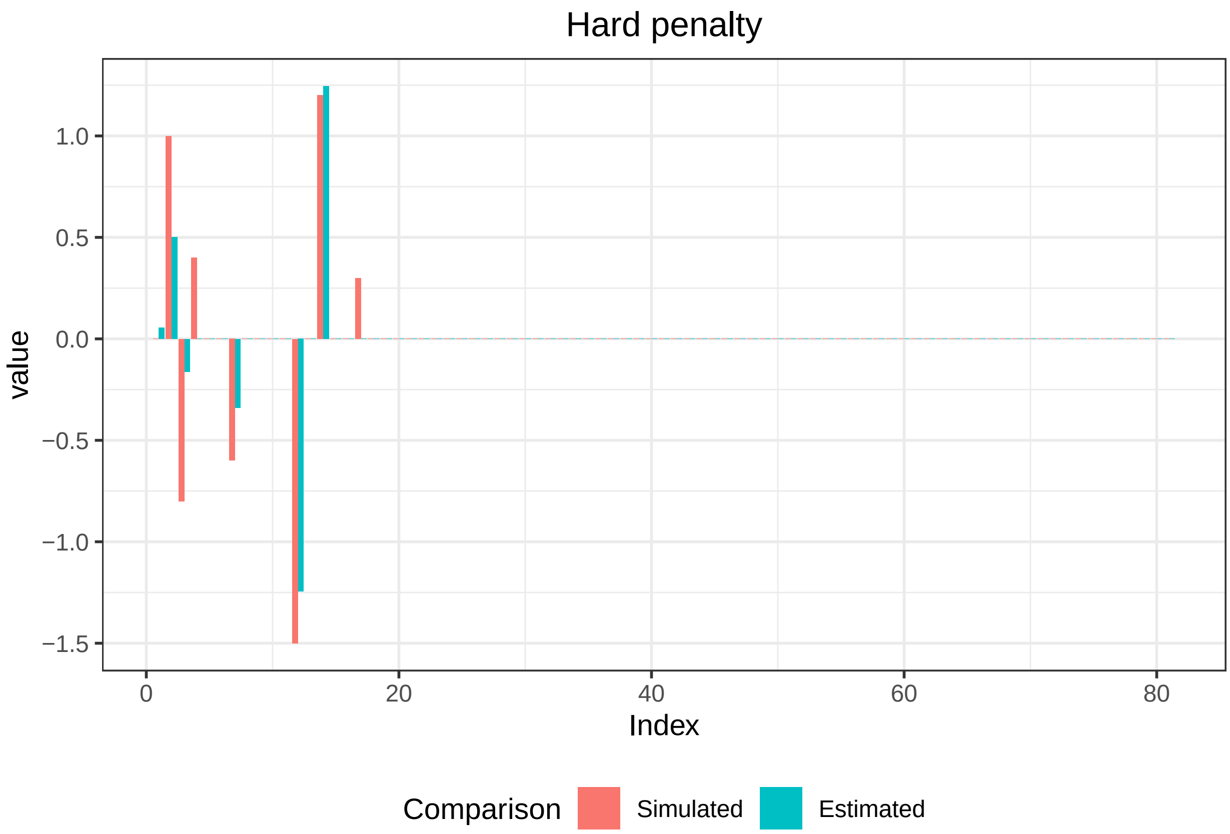
```
library(reshape2)
library(ggplot2)

# [Adaptive]
tmp <- data.frame(c(0,beta),fit.ada$beta0[,1])
names(tmp) <- c('Simulated parameter','Estimated parameter')
tmp$Index <- 1:(p+1)
df <- melt(tmp,3)
names(df)[2] <- "Comparison"
ggplot(data=df, aes(x=Index, y=value, fill=Comparison)) +
  geom_bar(stat="identity", position=position_dodge()) +
  theme_bw() + theme(legend.position="bottom") + ggtitle('Adaptive lasso') +
  theme(plot.title = element_text(hjust = 0.5))
```




```r
# [Lasso/Huber]
tmp <- data.frame(c(0,beta),fit.soft$beta0[,1])
names(tmp) <- c('Simulated parameter','Estimated parameter')
tmp$Index <- 1:(p+1)
df <- melt(tmp,3)
names(df)[2] <- "Comparison"
ggplot(data=df, aes(x=Index, y=value, fill=Comparison)) +
  geom_bar(stat="identity", position=position_dodge()) +
  theme_bw() + theme(legend.position="bottom") + ggtitle('Lasso') +
  theme(plot.title = element_text(hjust = 0.5))
```




```
# [Hard]
tmp <- data.frame(c(0,beta),fit.hard$beta0[,1])
names(tmp) <- c('Simulated','Estimated')
tmp$Index <- 1:(p+1)
df <- melt(tmp,3)
names(df)[2] <- "Comparison"
ggplot(data=df, aes(x=Index, y=value, fill=Comparison)) +
  geom_bar(stat="identity", position=position_dodge()) +
  theme_bw() + theme(legend.position="bottom") + ggtitle('Hard penalty') +
  theme(plot.title = element_text(hjust = 0.5))
```



Hard penalty
value
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
20
40
60
80
Index
Comparison
Simulated
Estimated