Appendix S11: Model averaging: fitting the full model

Dormann et al.

06 December, 2017

Contents

1 Introduction

When $k > n$, i.e. more predictors than data points, then only few options exist for fitting all predictors.

Let's see how model averaging does in the best-case scenario: only independent linear effects. In principle, averaging bivariate regression should be able to approximate the full model here.

We start by producing a case study:

```
set.seed(1) #5
N <- 20 # number of data points
X <- as.data.frame(matrix(NA, ncol=50, nrow=N))
for (i in 1:50) X[,i] <- runif(N) # 10 uncorrelated predictors
colnames(X) <- paste0("X", 1:50)
X <- as.data.frame(scale(X, scale=F)) # center all predictors
attach(X)
Y <- 1 + as.matrix(X) %*% rep(1, 50) + rnorm(N, sd=1)
dats <- cbind(Y, X)
range(cor(X)[upper.tri(cor(X))])
```

```
[1] -0.6529581  0.6411668
```

```
detach(X)
```

Predictors are (unintentionally, but almost unavoidably, and at least realistically) correlated. Thus, there will be many options to estimate and combine predictors to yield a good fit, making the task mathematically ill-posed. That is one of the points here, however: if we have few data points and many predictors, fitting the full model is a challenge and any other model is likely to be "not the truth", even if fitting will.

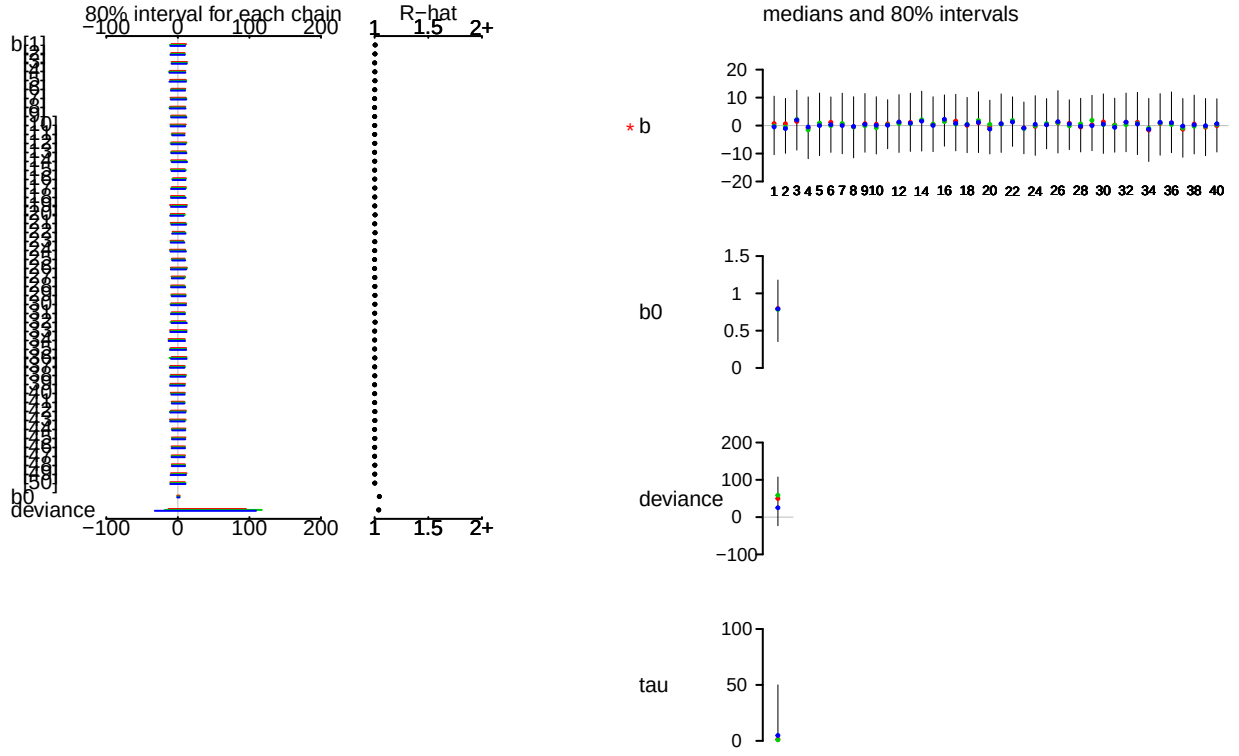
2 Bayesian full model

```
library(R2jags)
fullmodel <- function(){
  for (n in 1:N){ #loop through data points
    y[n] ~ dnorm(mu[n], tau)
    mu[n] <- b0 + inprod(b[], X[n,])
  }
  # priors:
  tau ~ dgamma(0.01, 0.01)
  b0 ~ dnorm(0, 0.01)
  for (k in 1:K){ b[k] ~ dnorm(0, 0.01)}
}
jags.data <- list(y=as.numeric(Y), X=X, N=20, K=50)
parameters <- c("b0", "b", "tau")
fulljags <- jags(jags.data, inits=NULL, parameters.to.save=parameters, model.file=fullmodel, n.chains=3
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 20
   Unobserved stochastic nodes: 52
   Total graph size: 1239

Initializing model
```

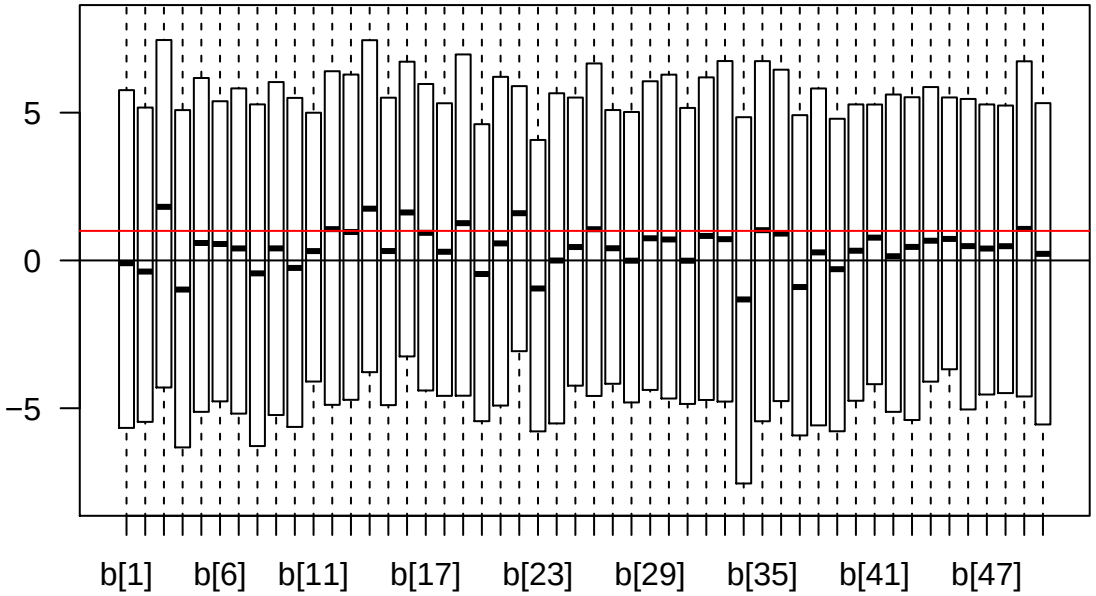
```
plot(fulljags)
```

lders/cc/3jfhfx190rb2ptxnqrqxj94m0000gp/T//RtmpHAX1fT/model59a23b53a39a.txt", fit using jags, 3 chains, each with 1000 iteration




```
#fulljags
boxplot(fulljags$BUGSoutput$sims.matrix[, 1:50], las=1, ylim=c(-8, 8))
```

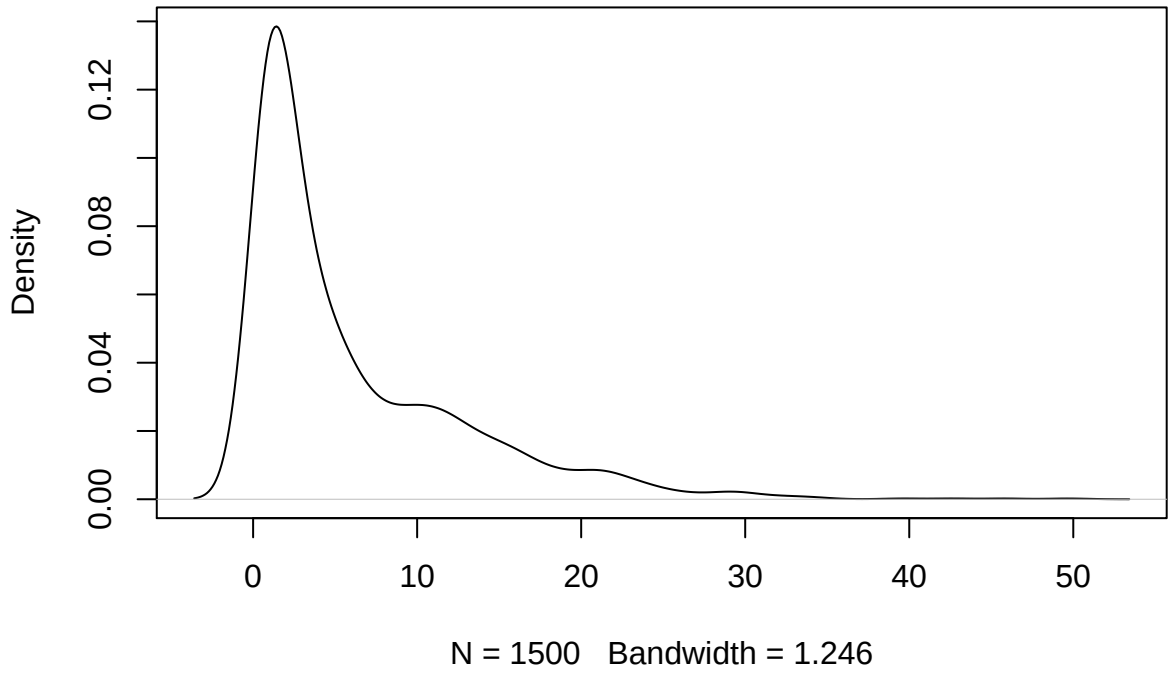
```
abline(h=c(0,1), col=c("black","red"))
```



So the estimates are around the correct values of 1, but the uncertainty in estimation is dramatic. We actually zoomed in to get an idea of where the estimates are relative to 1!

To compute RMSE is a little bit more involved. It requires to simulate the posterior predictions and identify their mode (= region of highest probability density; of course we could/should have done that within the JAGS code itself):

```
RMSEfull <- NULL
for (i in 1:nrow(fulljags$BUGSoutput$sims.list$b)){
  RMSEfull[i] <- sqrt(sum((Y - as.matrix(X) %*% fulljags$BUGSoutput$sims.list$b[i,1:50] - fulljags$BUGS
}
plot(density(RMSEfull))
```

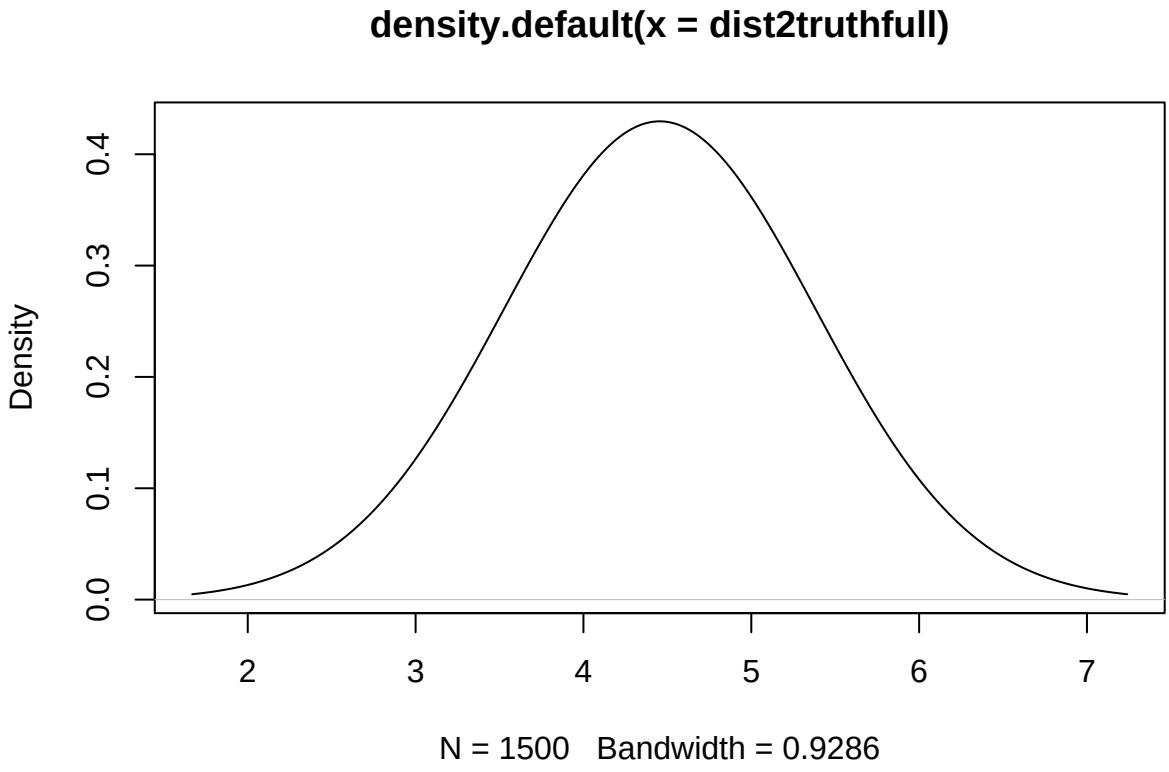



```
density(RMSEfull)$x[which.max(density(RMSEfull)$y)] # mode estimate
```

```
[1] 1.427057
```

But we can do better, since we know the truth:

```
dist2truthfull <- NULL
truth <- as.vector(as.matrix(X) %*% rep(1, 50) + 1)
for (i in 1:nrow(fulljags$BUGSoutput$sims.list$b)){
  dist2truthfull[i] <- sqrt(sum((Y - truth)^2))
}
plot(density(dist2truthfull))
```




```
density(dist2truthfull)$x[which.max(density(dist2truthfull)$y)] # mode estimate
```

```
[1] 4.449096
```

Let's compute the distance of the estimates from the truth (of 1):

```
sqrt(1/50*sum((fulljags$BUGSoutput$mean$b - 1)^2)) # mean SE around truth
```

```
[1] 0.870204
```

So, on average, the RMSE of the estimate in the Bayesian full model is as large as the estimate itself.

3 Bayesian lasso

We can try a Bayesian lasso. This is achieved by putting a Laplacian (double-exponential) prior on the model parameters (except the intercept). We would normally set the value for the exponential to a reasonable value, but here we let it be estimated from the data (making it in fact a mixed model, with random slopes). The value for this slope, lambda, should not be too large, as we expect the estimates to be around 1 (so even with bad bias not 20 or 50). A lambda value of around 5 would be reasonable.

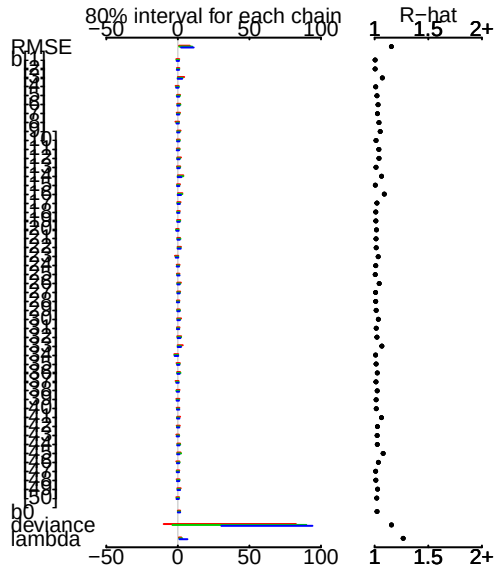
```
lassomodel <- function(){
  for (n in 1:N){ #loop through data points
    y[n] ~ dnorm(mu[n], tau)
    mu[n] <- b0 + inprod(b[], X[n,])
    res[n] <- y[n] - mu[n] # compute residuals for RMSE later
  }
  # priors:
  tau ~ dgamma(0.01, 0.01)
  b0 ~ dnorm(0, 0.01)
  for (k in 1:K){ b[k] ~ ddexp(0, lambda)}
```

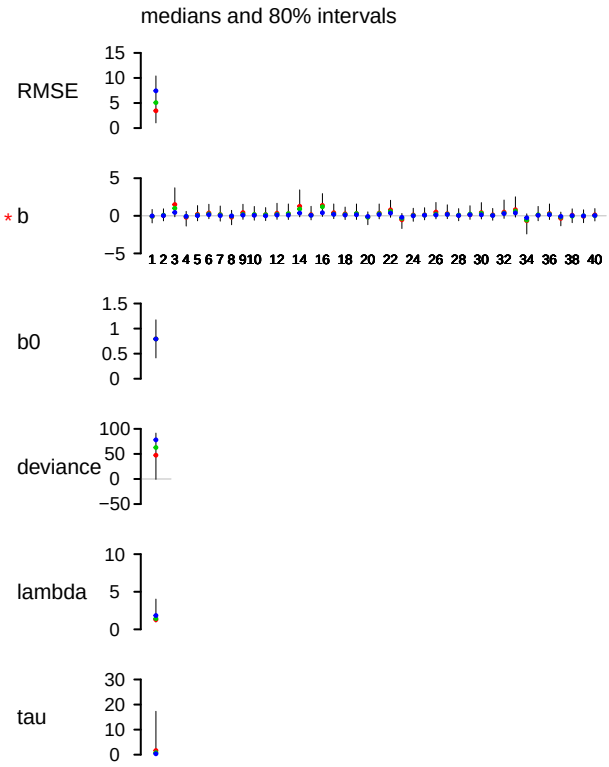
```
  lambda ~ dunif(0.001, 10)
  #http://stats.stackexchange.com/questions/28609/regularized-bayesian-logistic-regression-in-jags

  # derived variables:
  RMSE <- sqrt(sum(res[]^2))
}
jags.data <- list(y=as.numeric(Y), X=X, N=20, K=50)
parameters <- c("b0", "b", "lambda", "tau", "RMSE")
lassojags <- jags(jags.data, inits=NULL, parameters.to.save=parameters, model.file=lassomodel, n.chains=
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 20
   Unobserved stochastic nodes: 53
   Total graph size: 1219

Initializing model
```

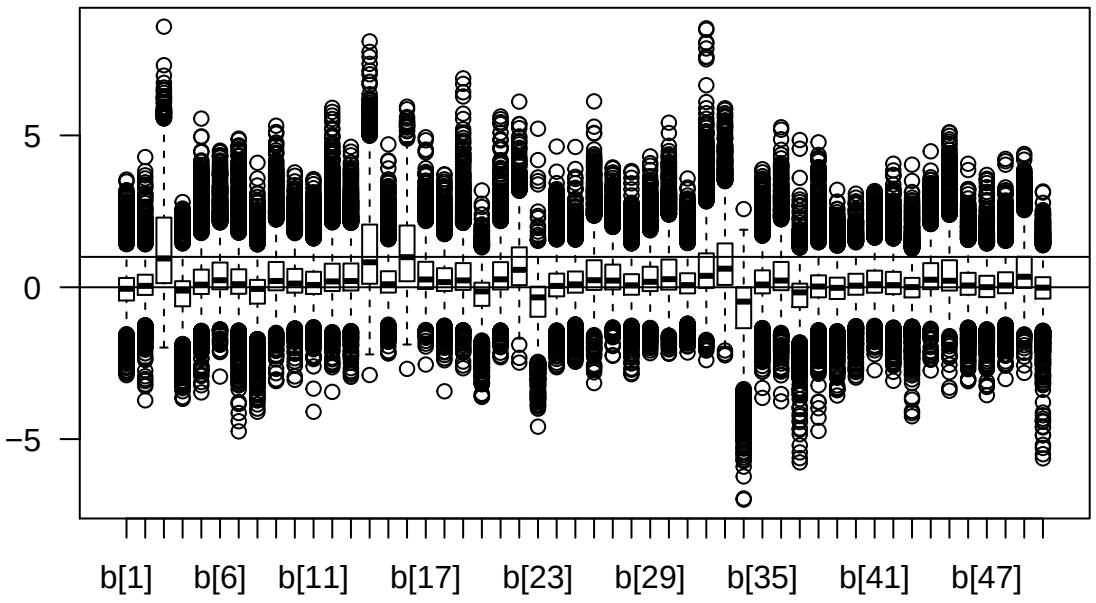
```
plot(lassojags)
```

ders/cc/3jfhfx190rb2ptxnqrqxj94m0000gp/T//RtmpHAX1fT/model59a24281a430.txt", fit using jags, 3 chains, each with 5000 iterations






```
#lassojags
boxplot(lassojags$BUGSoutput$sims.matrix[, 2:51], las=1)
abline(h=c(0,1))
```




```
density(lassojags$BUGSoutput$sims.list$RMSE)$x[which.max(density(lassojags$BUGSoutput$sims.list$RMSE)$y
```

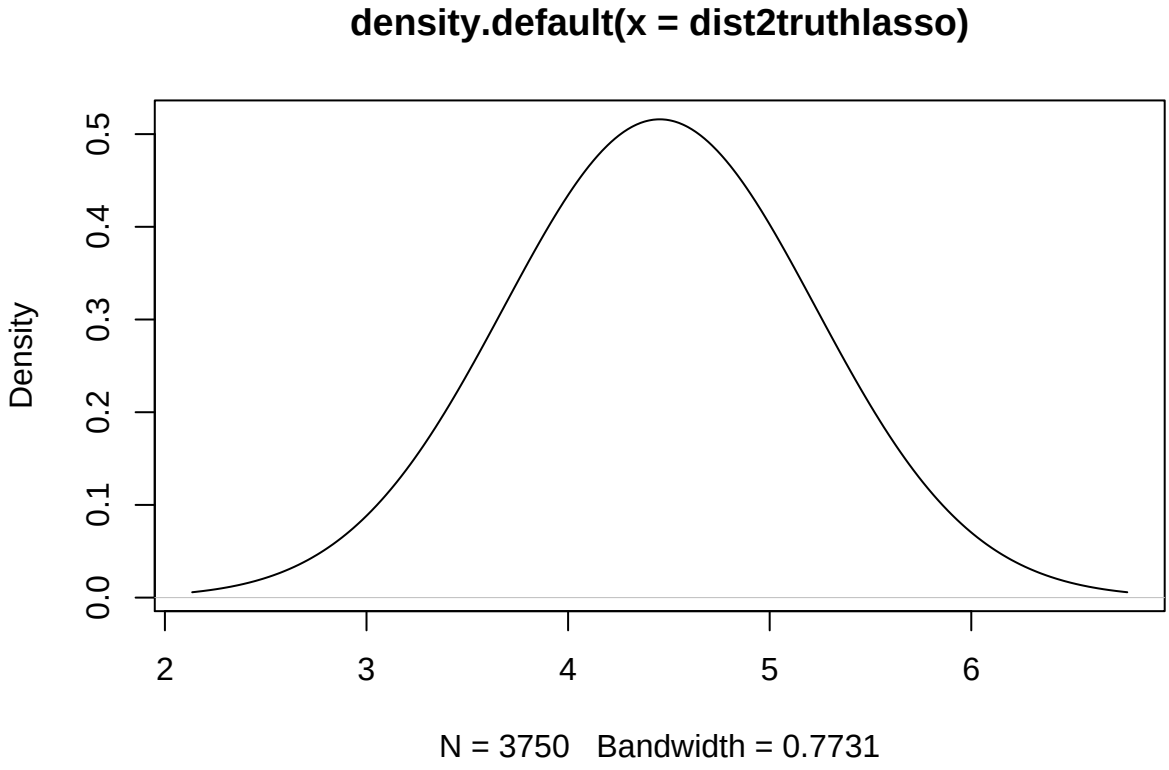
```
[1] 1.233061
```

The RMSE is substantially larger than for the full model. That is as expected: lasso accepts a larger bias in favour of reducing the variance of the estimates (see abstract of: http://www.d.umn.edu/math/Technical%20Reports/Technical%20Reports%202007-/TR%202010/TR_2010_6.pdf).

Of all the actually equally-relevant predictors the lasso shrunk most to around 0 and only picked a handful as larger than 0. More noticeably, the variance of the estimates is substantially reduced!

So while we have nicely reduced variance in the estimates, we have also gracefully removed any proximity of the estimates to the actual true values of 1!

```
dist2truthlasso <- NULL
truth <- as.vector(as.matrix(X) %*% rep(1, 50) + 1)
for (i in 1:nrow(lassojags$BUGSoutput$sims.list$b)){
  dist2truthlasso[i] <- sqrt(sum((Y - truth)^2))
}
plot(density(dist2truthlasso))
```




```
density(dist2truthlasso)$x[which.max(density(dist2truthlasso)$y)] # mode estimate
```

```
[1] 4.450009
```

As we can see, the lasso-distance to truth is about the same as for the full model.

Let's compute the distance of the estimates from the truth (of 1):

```
sqrt(1/50*sum((lassojags$BUGSoutput$mean$b - 1)^2)) # mean SE around truth
```

```
[1] 0.8359501
```

So, on average, the RMSE of the estimate in the lasso is as large as the estimate itself, and only a tiny bit lower than in the above full model.

4 Model averaging through bagging: randomForest

```
set.seed(1)
library(randomForest)
frf <- randomForest(y=as.vector(Y), x=X)
sqrt(mean((predict(frf) - Y)^2))
```

```
[1] 2.19797
```

```
sqrt(mean((predict(frf) - truth)^2))
```

```
[1] 1.915363
```

RandomForest is substantially better than the Bayesian full model and Bayesian lasso.

5 Model averaging based on many bivariate regression

The first choice would be MuMIn, but it "only" accepts 31 predictors.

```
library(MuMIn)
  options(na.action = "na.fail")
fm <- lm(Y~., data=X[,1:15])
fd <- dredge(fm, m.lim=c(0,1))
fMA <- get.models(fd, subset=TRUE)
fdMA <- model.avg(fMA)
sqrt(sum((Y - predict(fdMA))^2))
```

```
[1] 8.250682
```

```
sqrt(sum((truth - predict(fdMA))^2))
```

```
[1] 8.325972
```

```
#library(leaps)
#coef(regsubsets(x=X, y=Y, nvmax=1, really.big=T))+

#library(glmnet) # doesn't seem to go through all predictors
#fglmnet <- cv.glmnet(x=as.matrix(X), y=Y)
#plot(fglmnet)
#predict(fglmnet, type="coefficients") # only the intercept is left??
```

I guess this requires some handwork, a little for-loop going through the 50 single-variable models.

```
coef.mat <- matrix(NA, 50, 2)
for (i in 1:50){
  f <- as.formula(paste0("Y ~ X[,", i, "]"))
  fm <- lm(f)
  coef.mat[i,] <- coef(fm)
}
mean(coef.mat[,1]) # the intercept
```

```
[1] 0.791819
```

```
# conditional averaging:
predsC <- as.matrix(X) %*% coef.mat[,2]
# lm(Y ~ predsC) # just to check whether the intercept is really too far off (it is)
#abline(20.9, 0.29)
sqrt(mean((Y - predsC - mean(coef.mat[,1]))^2))
```

```
[1] 10.01779
```

```
sqrt(mean((truth - predsC - mean(coef.mat[,1]))^2))
```

```
[1] 10.13198
```

```
# unconditional averaging:
predsU <- rowMeans(as.matrix(X) %*% diag(coef.mat[,2])) # average of single model predictions
# alternatively (since for the linear model only the averaging of coefficients is fine):
predsU <- as.matrix(X) %*% colMeans(diag(coef.mat[,2])) # same values!
sqrt(mean((Y - predsU - mean(coef.mat[,1]))^2))
```

```
[1] 2.240833
```

```
sqrt(mean((truth - predsU - mean(coef.mat[,1]))^2))
```

```
[1] 2.090847
```

So the unconditional averaging actually makes quite reasonable predictions, even better than randomForest, albeit worse than the Bayesian models. The conditional is clearly unacceptable.

Let's compute the distance of the estimates from the truth (of 1):

```
sqrt(1/50*sum((mean(coef.mat[,2]) -1)^2)) # mean SE around truth
```

```
[1] 0.04818903
```

So, on average, the RMSE of the estimate in the unconditional MA is substantially (!) lower than in the Bayesian approaches.

We need to repeat the above analyses a few times (with different seeds, obviously) and possibly for different-size data sets to see the general picture. When we do that, we find that

- full and lasso models yield very similar prediction RMSE and estimator biases;
- randomForest and MA of linear models are substantially worse in prediction;
- MA of linear models is *very* good in terms of getting the estimators right. (Note that this may be because our truth is also a simple linear prediction problem.)

6 A more complex model - and linear model averaging

```
set.seed(4)
N <- 50 # number of data points
X <- as.data.frame(matrix(NA, ncol=10, nrow=N))
for (i in 1:10) X[,i] <- runif(N) # 10 uncorrelated predictors
colnames(X) <- paste0("X", 1:10)
attach(X)
truth <- X1-X1^2 +1.2*X2-1.2*X2^2 + 1.3*X3-1.3*X3^2 + 1.4*X4-1.4*X4^2 + 1.5*X5-1.5*X5^2 + X6-X6^2 + X7-
Y <- truth + rnorm(N, sd=1)
dats <- cbind(Y, X)
#pairs(dats, pch=".")
detach(X)

#summary(lm(Y~poly(X1, 2) + poly(X2,2) + poly(X3, 2) + poly(X4, 2) + poly(X5, 2), data=dats))

library(MuMIn)
options(na.action = "na.fail") # needs to be set, somehow
f <- as.formula(paste("Y~ ", paste0("X", 1:10, collapse="+")))      # only linear terms
#f <- as.formula(paste("Y~", paste0("poly(X", 1:10, ", 2)", collapse=" + "))) # always polynoms
#f <- as.formula(paste("Y~", paste0("X", 1:10, collapse=" + "), " + ", paste0("I(X", 1:10, "^2)", colla
summary(fm <- lm(f, data=dats))
```

```

Call:
lm(formula = f, data = dats)

Residuals:
     Min       1Q   Median       3Q      Max
-2.14113 -0.78823  0.09564  0.64597  2.74252
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.14478    1.05494   2.033   0.0489 *
X1           0.77065    0.58323   1.321   0.1941
X2          -0.82179    0.62219  -1.321   0.1943
X3           0.39340    0.62123   0.633   0.5303
X4           0.34719    0.60672   0.572   0.5704
X5           0.40716    0.57226   0.712   0.4810
X6          -0.09949    0.52117  -0.191   0.8496
X7           0.33624    0.49729   0.676   0.5029
X8          -0.17102    0.57146  -0.299   0.7663
X9          -1.53920    0.64173  -2.399   0.0213 *
X10         -0.64509    0.62268  -1.036   0.3066
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.048 on 39 degrees of freedom
Multiple R-squared:  0.179, Adjusted R-squared:  -0.03156
F-statistic: 0.8501 on 10 and 39 DF,  p-value: 0.5852
```

```
dres <- dredge(fm, m.lim=c(0, 3))  ###### here is the decision on model complexity ########
coef.mat <- print(dres)[, 1:11]
```

```
Global model call: lm(formula = f, data = dats)
---
Model selection table
    (Intrc)     X1     X10        X2     X3       X4     X5        X6     X7        X8      X9 df
513   2.081                                                                          -0.8382  3
517   2.741               -0.819600                                                  -1.2860  4
518   2.463 0.6471        -0.873000                                                  -1.4170  5
1     1.639                                                                                   2
514   1.789 0.5883                                                                   -0.9307  4
641   1.925                                                          0.3520          -0.8613  4
521   1.898                          0.3969                                          -0.8661  4
545   1.897                                          0.3648                          -0.8324  4
519   2.998       -0.4116 -0.884300                                                  -1.3320  5
645   2.584               -0.797400                                                  -1.2950  5
515   2.220       -0.2804                                                            -0.8456  4
529   2.137                                  -0.16100                                -0.8031  4
525   2.588               -0.766500 0.2387                                           -1.2740  5
577   2.060                                                  0.053570                -0.8465  4
769   2.108                                                                -0.054690 -0.8415  4
549   2.594               -0.770000                  0.2127                          -1.2550  5
2     1.391 0.4271                                                                            3
533   2.755               -0.809900        -0.06311                                  -1.2670  5
581   2.726               -0.818400                          0.036500                -1.2910  5
773   2.740               -0.819800                                        0.001726 -1.2860  5
33    1.449                                          0.3804                                   3
17    1.797                                  -0.34180                                        3
129   1.495                                                          0.3008                3
9     1.479                          0.3190                                                  3
546   1.600 0.5916                                   0.3703                          -0.9253  5
522   1.618 0.5792                   0.3812                                          -0.9561  5
3     1.760       -0.2510                                                                     3
642   1.687 0.5428                                                   0.2819          -0.9420  5
```

516	1.938	0.6035	-0.3142								-0.9414	5
5	1.706			-0.130400								3
65	1.666							-0.058560				3
257	1.635									0.007264		3
770	1.834	0.5950								-0.098430	-0.9376	5
530	1.744	0.6219				0.08229					-0.9538	5
578	1.773	0.5872						0.042020			-0.9370	5
673	1.735						0.3719		0.3577		-0.8558	5
649	1.755				0.3798				0.3385		-0.8872	5
553	1.703				0.4078		0.3756				-0.8609	5
523	2.052		-0.3996		0.4919						-0.8834	5
643	2.073		-0.3187						0.3746		-0.8713	5
547	2.042		-0.3303				0.4016				-0.8406	5
585	1.808				0.4544			0.162300			-0.8955	5
657	1.969					-0.10920			0.3380		-0.8367	5
897	1.972								0.3634	-0.106900	-0.8685	5
705	1.931							-0.018900	0.3560		-0.8587	5
537	1.941				0.3778	-0.09969					-0.8431	5
34	1.197	0.4315					0.3856					4
777	1.931				0.4000					-0.071210	-0.8706	5
801	1.937						0.3731			-0.090550	-0.8377	5
561	1.937					-0.08490	0.3445				-0.8142	5
609	1.888						0.3621	0.026530			-0.8365	5
531	2.296		-0.3008			-0.18800					-0.8053	5
10	1.247	0.4163			0.3019							4
130	1.297	0.3854							0.2476			4
161	1.299						0.3868		0.3070			4
771	2.270		-0.2924							-0.090760	-0.8514	5
4	1.516	0.4387	-0.2731									4
579	2.227		-0.2845					-0.011640			-0.8439	5
41	1.280				0.3308		0.3896					4
18	1.542	0.3462				-0.22520						4
49	1.610					-0.27470	0.3137					4
35	1.579		-0.3026				0.4142					4
145	1.654					-0.30690			0.2656			4
785	2.185					-0.17610				-0.088450	-0.8052	5
593	2.120					-0.15660		0.039460			-0.8102	5
6	1.455	0.4236		-0.119400								4
19	1.951		-0.2936			-0.36860						4
66	1.425	0.4310						-0.075830				4
137	1.349				0.3028				0.2888			4
25	1.646				0.2668	-0.30480						4
11	1.606		-0.3469		0.4001							4
258	1.399	0.4281								-0.019110		4
833	2.087							0.053740		-0.054930	-0.8499	5
131	1.622		-0.2830						0.3203			4
97	1.486						0.3888	-0.086170				4
21	1.866			-0.133000		-0.34280						4
37	1.494			-0.072990			0.3667					4
289	1.462						0.3831			-0.029840		4
81	1.837					-0.34750		-0.079210				4
193	1.541							-0.127000	0.3289			4
273	1.832					-0.35330				-0.065840		4
133	1.553			-0.105500					0.2926			4

385	1.509								0.3044	-0.034790	4
13	1.525		-0.071650	0.3008							4
73	1.470			0.3239			0.014330				4
265	1.481			0.3192						-0.004204	4
7	1.850	-0.2713	-0.156600								4
67	1.840	-0.2952					-0.125900				4
259	1.772	-0.2541								-0.023430	4
69	1.743		-0.138000				-0.071360				4
261	1.697		-0.132300							0.021650	4
321	1.663						-0.058520			0.006849	4
36	1.328	0.4458	-0.3261			0.4223					5
42	1.042	0.4204		0.3136		0.3942					5
162	1.098	0.3889				0.3904		0.2534			5
12	1.373	0.4289	-0.3653	0.3868							5
43	1.404		-0.4081	0.4277		0.4379					5
163	1.432		-0.3370			0.4250		0.3309			5
50	1.307	0.3815			-0.13820	0.3515					5
98	1.238	0.4370				0.3959	-0.104200				5
169	1.144			0.3144		0.3953		0.2947			5
132	1.426	0.3948	-0.2975					0.2668			5
138	1.162	0.3768		0.2902				0.2373			5
38	1.235	0.4296	-0.060680			0.3742					5
290	1.218	0.4347				0.3910				-0.057390	5
177	1.450				-0.23480	0.3292		0.2792			5
20	1.696	0.3495	-0.2977		-0.25120						5
51	1.764		-0.3284		-0.29810	0.3448					5
139	1.476		-0.3753	0.3894				0.3112			5
146	1.440	0.3139			-0.20540			0.2339			5
27	1.795		-0.3727	0.3501	-0.32730						5
147	1.810		-0.3179		-0.33350			0.2845			5
26	1.385	0.3506		0.2727	-0.18590						5
57	1.435			0.2896	-0.23060	0.3325					5
225	1.349					0.4030	-0.158400	0.3423			5
194	1.344	0.3873					-0.131600	0.2764			5
68	1.607	0.4488	-0.3266				-0.151000				5
8	1.603	0.4352	-0.2920	-0.147400							5
134	1.353	0.3837		-0.100000				0.2400			5
153	1.512			0.2579	-0.27200			0.2594			5
14	1.288	0.4151		-0.063830	0.2857						5
266	1.258	0.4179		0.3029						-0.029370	5
74	1.251	0.4169		0.2992			-0.007934				5
386	1.317	0.3874							0.2526	-0.051510	5
417	1.326					0.3939		0.3148		-0.074370	5
165	1.329		-0.045390			0.3782		0.3033			5
260	1.542	0.4420	-0.2803							-0.053830	5
99	1.679		-0.3664			0.4383	-0.173200				5
22	1.610	0.3417		-0.123200	-0.22770						5
82	1.584	0.3488			-0.23060		-0.086240				5
274	1.576	0.3457			-0.23640					-0.062950	5
209	1.707				-0.31300		-0.138700	0.2956			5
297	1.296			0.3325		0.3936				-0.042800	5
195	1.739		-0.3655				-0.219900	0.3746			5
105	1.287			0.3265		0.3907	-0.012820				5
45	1.283		-0.005252	0.3294		0.3885					5

```
83    2.063        -0.3523                -0.38560        -0.161800                        5
23    2.047        -0.3151 -0.163700      -0.37180                                         5
113   1.654                               -0.28000 0.3220 -0.098060                        5
149   1.716                -0.110900      -0.30900                 0.2567                  5
39    1.643        -0.3132 -0.098590              0.3969                                   5
53    1.665                -0.086080      -0.27910 0.2965                                  5
305   1.652                               -0.28840 0.3181                 -0.083210        5
401   1.701                               -0.32290                 0.2738 -0.097380        5
291   1.612        -0.3129                        0.4220                  -0.071410        5
70    1.498 0.4279         -0.128700                      -0.087650                        5
275   2.019        -0.3104                -0.38950                        -0.110800        5
262   1.457 0.4239         -0.118900                                      -0.005929        5
135   1.703        -0.2992 -0.132900                                0.3110                 5
201   1.380                        0.2825                -0.057910 0.3024                 5
15    1.667        -0.3534 -0.090680 0.3786                                                5
29    1.703                -0.084960 0.2446 -0.30860                                        5
141   1.382                -0.049390 0.2905                         0.2854                 5
393   1.366                        0.3042                           0.2933 -0.044190        5
322   1.433 0.4321                                        -0.075980        -0.019900        5
281   1.682                        0.2673 -0.31660                 -0.067850               5
75    1.646        -0.3624         0.3854                -0.054480                        5
267   1.630        -0.3542         0.4037                                 -0.050040        5
89    1.659                        0.2603 -0.30700       -0.018230                        5
387   1.656        -0.2933                                          0.3285 -0.073570        5
85    1.918                -0.142900      -0.34960       -0.092590                        5
101   1.538                -0.081750              0.3741 -0.092710                        5
277   1.891                -0.128500      -0.35180                 -0.051560               5
353   1.499                               0.3918 -0.086530        -0.031290               5
197   1.610                -0.117600                     -0.136400 0.3218                 5
293   1.501                -0.070810              0.3690                  -0.020780        5
337   1.874                               -0.35940       -0.080250        -0.067680        5
449   1.558                                              -0.128100 0.3332 -0.039680        5
389   1.561                -0.103300                                0.2950 -0.022260        5
71    1.958        -0.3265 -0.178000                      -0.149500                        5
269   1.524                -0.072080 0.3005                          0.004304               5
77    1.523                -0.071140 0.3019                0.002775                       5
329   1.472                        0.3240                 0.014340        -0.004276       5
263   1.854        -0.2723 -0.156000                                      -0.008674       5
323   1.856        -0.2994                                -0.127000        -0.029820       5
325   1.734                -0.139900                      -0.071430         0.021970       5
     logLik  AICc delta weight
513 -70.452 147.4  0.00  0.049
517 -69.302 147.5  0.07  0.047
518 -68.394 148.2  0.73  0.034
1   -72.030 148.3  0.89  0.031
514 -69.733 148.4  0.93  0.031
641 -70.133 149.2  1.73  0.020
521 -70.166 149.2  1.80  0.020
545 -70.187 149.3  1.84  0.019
519 -68.994 149.4  1.93  0.019
645 -69.036 149.4  2.01  0.018
515 -70.312 149.5  2.09  0.017
529 -70.396 149.7  2.26  0.016
525 -69.200 149.8  2.34  0.015
```

```
577 -70.444 149.8  2.35  0.015
769 -70.446 149.8  2.36  0.015
549 -69.213 149.8  2.36  0.015
2   -71.667 149.9  2.43  0.014
533 -69.294 150.0  2.53  0.014
581 -69.299 150.0  2.54  0.014
773 -69.302 150.0  2.54  0.014
33  -71.760 150.0  2.62  0.013
17  -71.783 150.1  2.66  0.013
129 -71.811 150.1  2.72  0.012
9   -71.856 150.2  2.81  0.012
546 -69.453 150.3  2.84  0.012
522 -69.462 150.3  2.86  0.012
3   -71.925 150.4  2.95  0.011
642 -69.527 150.4  2.99  0.011
516 -69.553 150.5  3.04  0.011
5   -71.992 150.5  3.08  0.010
65  -72.021 150.6  3.14  0.010
257 -72.030 150.6  3.16  0.010
770 -69.714 150.8  3.37  0.009
530 -69.720 150.8  3.38  0.009
578 -69.728 150.8  3.39  0.009
673 -69.855 151.1  3.65  0.008
649 -69.868 151.1  3.68  0.008
553 -69.882 151.1  3.70  0.008
523 -69.895 151.2  3.73  0.008
643 -69.951 151.3  3.84  0.007
547 -69.994 151.4  3.93  0.007
585 -70.101 151.6  4.14  0.006
657 -70.107 151.6  4.15  0.006
897 -70.111 151.6  4.16  0.006
705 -70.132 151.6  4.20  0.006
537 -70.145 151.7  4.23  0.006
34  -71.386 151.7  4.24  0.006
777 -70.156 151.7  4.25  0.006
801 -70.172 151.7  4.28  0.006
561 -70.173 151.7  4.28  0.006
609 -70.185 151.7  4.31  0.006
531 -70.237 151.8  4.41  0.005
10  -71.509 151.9  4.48  0.005
130 -71.520 151.9  4.50  0.005
161 -71.530 151.9  4.52  0.005
771 -70.297 152.0  4.53  0.005
4   -71.541 152.0  4.55  0.005
579 -70.312 152.0  4.56  0.005
41  -71.571 152.0  4.61  0.005
18  -71.572 152.0  4.61  0.005
49  -71.607 152.1  4.68  0.005
35  -71.608 152.1  4.68  0.005
145 -71.613 152.1  4.69  0.005
785 -70.382 152.1  4.70  0.005
593 -70.392 152.1  4.72  0.005
6   -71.635 152.2  4.73  0.005
19  -71.639 152.2  4.74  0.005
```

66	-71.652	152.2	4.77	0.004
137	-71.653	152.2	4.77	0.004
25	-71.663	152.2	4.79	0.004
11	-71.665	152.2	4.79	0.004
258	-71.667	152.2	4.80	0.004
833	-70.438	152.2	4.82	0.004
131	-71.677	152.2	4.82	0.004
97	-71.741	152.4	4.95	0.004
21	-71.743	152.4	4.95	0.004
37	-71.749	152.4	4.96	0.004
289	-71.759	152.4	4.98	0.004
81	-71.767	152.4	5.00	0.004
193	-71.771	152.4	5.01	0.004
273	-71.775	152.4	5.01	0.004
133	-71.786	152.5	5.04	0.004
385	-71.809	152.5	5.08	0.004
13	-71.845	152.6	5.15	0.004
73	-71.856	152.6	5.18	0.004
265	-71.856	152.6	5.18	0.004
7	-71.871	152.6	5.21	0.004
67	-71.888	152.7	5.24	0.004
259	-71.924	152.7	5.31	0.003
69	-71.979	152.8	5.42	0.003
261	-71.991	152.9	5.45	0.003
321	-72.021	152.9	5.51	0.003
36	-71.206	153.8	6.35	0.002
42	-71.214	153.8	6.37	0.002
162	-71.230	153.8	6.40	0.002
12	-71.295	154.0	6.53	0.002
43	-71.308	154.0	6.55	0.002
163	-71.340	154.0	6.62	0.002
50	-71.352	154.1	6.64	0.002
98	-71.357	154.1	6.65	0.002
169	-71.358	154.1	6.65	0.002
132	-71.371	154.1	6.68	0.002
138	-71.374	154.1	6.69	0.002
38	-71.378	154.1	6.69	0.002
290	-71.380	154.1	6.70	0.002
177	-71.419	154.2	6.78	0.002
20	-71.423	154.2	6.78	0.002
51	-71.428	154.2	6.79	0.002
139	-71.429	154.2	6.80	0.002
146	-71.441	154.2	6.82	0.002
27	-71.442	154.2	6.82	0.002
147	-71.444	154.3	6.83	0.002
26	-71.445	154.3	6.83	0.002
57	-71.466	154.3	6.87	0.002
225	-71.467	154.3	6.87	0.002
194	-71.477	154.3	6.89	0.002
68	-71.487	154.3	6.91	0.002
8	-71.493	154.3	6.92	0.002
134	-71.498	154.4	6.93	0.002
153	-71.500	154.4	6.94	0.002
14	-71.501	154.4	6.94	0.002

```
266 -71.508 154.4  6.95  0.002
74  -71.509 154.4  6.96  0.002
386 -71.516 154.4  6.97  0.001
417 -71.520 154.4  6.98  0.001
165 -71.525 154.4  6.99  0.001
260 -71.536 154.4  7.01  0.001
99  -71.537 154.4  7.01  0.001
22  -71.537 154.4  7.01  0.001
82  -71.552 154.5  7.04  0.001
274 -71.565 154.5  7.07  0.001
209 -71.565 154.5  7.07  0.001
297 -71.568 154.5  7.07  0.001
195 -71.569 154.5  7.08  0.001
105 -71.571 154.5  7.08  0.001
45  -71.571 154.5  7.08  0.001
83  -71.577 154.5  7.09  0.001
23  -71.579 154.5  7.10  0.001
113 -71.583 154.5  7.10  0.001
149 -71.586 154.5  7.11  0.001
39  -71.587 154.5  7.11  0.001
53  -71.591 154.5  7.12  0.001
305 -71.595 154.6  7.13  0.001
401 -71.597 154.6  7.13  0.001
291 -71.599 154.6  7.14  0.001
70  -71.615 154.6  7.17  0.001
275 -71.618 154.6  7.17  0.001
262 -71.635 154.6  7.21  0.001
135 -71.638 154.6  7.21  0.001
201 -71.646 154.7  7.23  0.001
15  -71.647 154.7  7.23  0.001
29  -71.648 154.7  7.23  0.001
141 -71.648 154.7  7.23  0.001
393 -71.650 154.7  7.24  0.001
322 -71.652 154.7  7.24  0.001
281 -71.655 154.7  7.25  0.001
75  -71.658 154.7  7.26  0.001
267 -71.661 154.7  7.26  0.001
89  -71.662 154.7  7.26  0.001
387 -71.668 154.7  7.27  0.001
85  -71.721 154.8  7.38  0.001
101 -71.727 154.8  7.39  0.001
277 -71.739 154.8  7.42  0.001
353 -71.739 154.8  7.42  0.001
197 -71.741 154.8  7.42  0.001
293 -71.748 154.9  7.43  0.001
337 -71.759 154.9  7.46  0.001
449 -71.768 154.9  7.48  0.001
389 -71.785 154.9  7.51  0.001
71  -71.819 155.0  7.58  0.001
269 -71.845 155.1  7.63  0.001
77  -71.845 155.1  7.63  0.001
329 -71.856 155.1  7.65  0.001
263 -71.871 155.1  7.68  0.001
323 -71.886 155.1  7.71  0.001
```

```
325 -71.979 155.3  7.90  0.001
Models ranked by AICc(x)
```

```
coef.mat[is.na(coef.mat)] <- 0
coef.mat <- coef.mat[, c(1,2,4:11, 3)] # put X10 at the end, rather than in position 3
#get.models(dres, subset=T)[[1]] # there are 56 models, incl. the intercept-only model
predsComplex <- as.vector(cbind(1,as.matrix(X)) %*% colMeans(coef.mat)) # average of single model predi
#RMSE:
sqrt(mean((truth - predsComplex)^2))
```

```
[1] 0.3063338
```

```
# Compare to correct model structure:
fm <- lm(Y ~ poly(X1,2)+poly(X2,2) +poly(X3,2) + poly(X4, 2) +poly(X5,2) + poly(X6, 2) + poly(X7, 2)+ X
summary(fm)
```

```
Call:
lm(formula = Y ~ poly(X1, 2) + poly(X2, 2) + poly(X3, 2) + poly(X4,
    2) + poly(X5, 2) + poly(X6, 2) + poly(X7, 2) + X1:X2 + X3:X4 +
    X5:X6, data = dats)

Residuals:
     Min       1Q   Median       3Q      Max
-1.79298 -0.70339 -0.01707  0.59395  1.92366

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.59704    1.14404   0.522   0.6054
poly(X1, 2)1 -2.40744    2.74228  -0.878   0.3865
poly(X1, 2)2  1.48494    1.30820   1.135   0.2648
poly(X2, 2)1 -1.67677    2.88661  -0.581   0.5654
poly(X2, 2)2 -3.27108    1.28550  -2.545   0.0160 *
poly(X3, 2)1 -0.01537    2.49029  -0.006   0.9951
poly(X3, 2)2  0.08557    1.21992   0.070   0.9445
poly(X4, 2)1  0.73907    2.90619   0.254   0.8009
poly(X4, 2)2 -2.01538    1.27798  -1.577   0.1246
poly(X5, 2)1 -1.81410    2.68642  -0.675   0.5043
poly(X5, 2)2 -1.14888    1.29521  -0.887   0.3817
poly(X6, 2)1 -4.99418    2.91058  -1.716   0.0959 .
poly(X6, 2)2  1.81326    1.21337   1.494   0.1449
poly(X7, 2)1  1.49297    1.22456   1.219   0.2317
poly(X7, 2)2  1.62498    1.41725   1.147   0.2601
X1:X2         1.20785    1.94292   0.622   0.5386
X3:X4        -0.42328    2.64222  -0.160   0.8737
X5:X6         3.21027    2.29329   1.400   0.1712
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.056 on 32 degrees of freedom
Multiple R-squared:  0.3166,    Adjusted R-squared:  -0.0465
F-statistic: 0.8719 on 17 and 32 DF,  p-value: 0.6078
```

```
sqrt(mean((truth - predict(fm))^2))
```

```
[1] 0.578162
```

So the shrinkage due to the MA yields better predictions that the full (and correct) model!