Exam 2

Jiaming

11/18/2017

1.

a.

We first define a set of candidate θ values. Then we randomly divide all obsevations into K folds and for each fold i, we hold that fold data out and then for each θ, we fit a model to the remaining $(K-1)$ folds and then use that model to make the predictions $\hat{y}$ for observations in the i^{th} fold. Finally $RMSE$ is calculated for each non-overlap hold-out fold for each θ. Thus each θ is associated with a set of K RMSE values $RMSE_i(\theta),\ i = 1, 2, \dots K$, we can choose the one with the lowest **weighted** mean RMSE ($RMSE(\theta) = \frac{n_i}{N}\sum_{i=1}^{K} RMSE_i(\theta)$) value.

b.

To calculate the standard error for each θ we consider in a K-fold validation, we need to calculate the standard deviation of residuals

$StdDev(\theta) = \sqrt{\frac{1}{K-1}\sum_{i=1}^{K}(RMSE_i(\theta) - RMSE(\theta))^2}$

and then $SE(\theta) = StdDev(\theta)/\sqrt{K}$

c.

We chose the largest value of θ that is within 1 Standard Error of the best fitting $\hat{\theta} = argmin(RMSE(\theta))$. So the oneSE method can be described as $RMSE_{oneSE}(\theta) \leq RMSE(\hat{\theta}) + SE(\hat{\theta})$. It would be a larger value than "best" θ because a larger θ corresponds to a more flexible model. Because the model with "oneSE" θ is simpler and less flexible than "best" model, we can avoid overfitting using "oneSE" model

d.

The difference will decrease. Using repeated (n times) K-fold validation, each θ is associated with a set of $K \times n$ RMSE values $RMSE_i(\theta),\ i = 1, 2, \dots K \times n$. As we're averaging across more folds, the standard deviation $StdDev$ does not change, but SE decrease because $SE(\theta) = StdDev(\theta)/\sqrt{K \times n}$. So the difference will decrease.

2

```
#' Create a data set where y ~ x1 + ... +x5  but there are num_w extra covariates that are
  #' independent of the response y.  The independent variables are labeled wXX.
  #'
  #' n: sample size
#' num_w: number of independent covariates
#' beta: The beta vector for the non-zero linear model terms
```

```r
#' sd: The standard deviation of the error terms
#' beta0: The beta term associated the the intercept
make_data <- function(n, num_w=5, beta=c(4,2,-4,-2,3), sd=1, beta0=2 ){
num_x <- length(beta)
X <- NULL
for(j in 1:num_x){
X <- cbind(X, runif(n, 0,1))
}
Ey <- cbind(rep(1,n),X) %*% c(beta0,beta)
for(j in 1:num_w){
X <- cbind(X, runif(n, 0,1))
}
colnames(X) <- c( paste('x',1:num_x,sep=''), paste('w',1:num_w,sep='') )
y <- Ey + rnorm(n, sd=sd)
return(data.frame(y=y, X))
}
#' Perform stepwise model selection using AIC. This function returns the number of x
#' variables selected, the number of w variables selected as well as the estmated RMSE. #'
#' data: the data to do model selection on. Assumes there are y, x, and w columns.
#' start_f, smallest_f, and biggest_f all determin the range and starting point
#' for the stepwise model selelction.
do_stepwise <- function(data, start_f=y~., smallest_f=y~1, biggest_f=y~.){
model<- lm(start_f, data=data)
model <- step(model, scope=list(lowest=smallest_f, upper=biggest_f), trace=0)
coefs <- names(model$coefficients)
output <- data.frame(
  method='Stepwise',
  num_x = length(dplyr::starts_with('x', vars=coefs)),
  num_w = length(dplyr::starts_with('w', vars=coefs)),
  RMSE = summary(model)$sigma)
return(output)
}
```

```r
#' Perform model shrinkage using ridge regression or LASSO. This function returns
#' the number of x variables selected, the number of w variables selected as well
#' as the estmated RMSE which is obtained via cross-validation.
#'
#' data:  the data to do model selection on.  Assumes there are y, x, and w columns.
#' type: Either 'lasso' or 'ridge'
#' oneSE: Should we use the lambda than minimizes or the largest lambda within in SE?
#' biggest_f: determines the largest model considered.
do_shrinkage <- function(data, type, oneSE=FALSE, biggest_f=y~.){
type = stringr::str_to_lower(type)
if( stringr::str_detect('lasso', type) & !stringr::str_detect('ridge', type) ){
  type='lasso'
}
else if( !stringr::str_detect('lasso', type) & stringr::str_detect('ridge', type) ){
  type='ridge'
}
else{
  stop('Type must either be "lasso" or "ridge"')
}
X <- model.matrix(biggest_f, data=data)[,-1] # remove the intercept column
y <- data$y
```

```
cvfit <- glmnet::cv.glmnet(X, y, nfolds =5, alpha = ifelse(type=='lasso',1,0) )
betas <- coef(cvfit, s = ifelse(oneSE==TRUE, 'lambda.1se','lambda.min'))
foo <- as.vector(betas)
coefs <- rownames( betas )[ which(foo != 0 ) ]
output <- data.frame(
  method=paste(type, ifelse(oneSE,'_oneSE',''), sep=''),
  num_x = length(dplyr::starts_with('x', vars=coefs)),
  num_w = length(dplyr::starts_with('w', vars=coefs)),
  RMSE = sqrt( mean( (y-predict(cvfit, newx=X))^2 )) )
return(output)
}
```

a.

From the command "y <- Ey + rnorm(n, sd=sd)", we know that the error term is a normal distributed term with standard deviation equals to 1. In the decomposition of MSE, we have a term as "Var(noise)". So the minimum value of RMSE that the model can obtain is "std(noise) = 1".

b.

We have 5 true predictors and 5 extraneous predictors in this case. The table of different sample size and the corresponding RMSE. And as the sample size increase, the difference between these two methods decreases. Generally, stepwise model selection method gives a lower RMSE value and thus performs better.

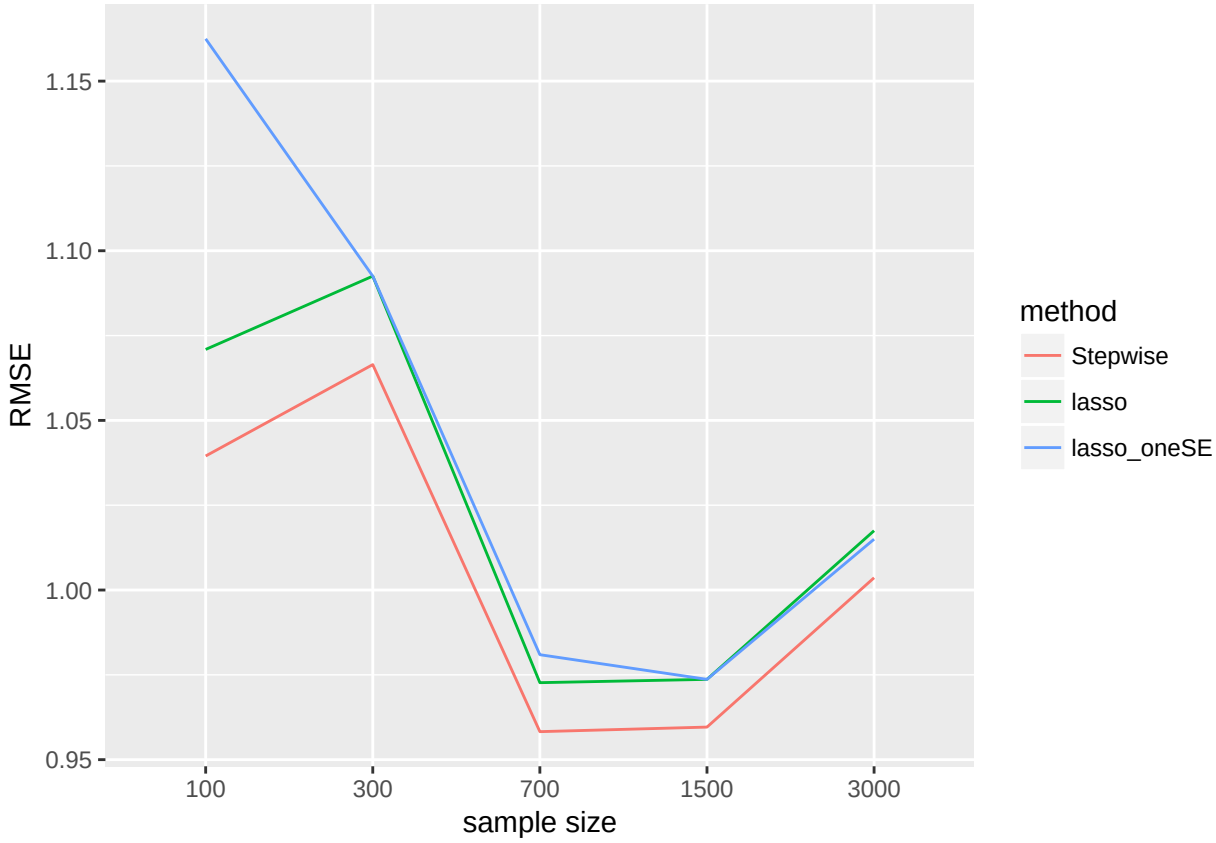
In this simulation, the error term in the data is normally distributed. And this fact matches up with the standard linear model theory, so the stepwise model works very well here. But as we discussed in 2.a, the theoretical minimum RMSE should be 1. So stepwise methods tend to overfit than Lasso.

```
compares <- data.frame()
sample_size <- c(100,300,700,1500,3000)
for(s in 1:length(sample_size)){
  results <- data.frame()
  data <- make_data(n=sample_size[s], num_w=5)
  results <- rbind(results, cbind( do_stepwise(data), rep=1) )
  results <- rbind(results, cbind( do_shrinkage(data, type='lasso', oneSE=FALSE), rep=1))
  results <- rbind(results, cbind( do_shrinkage(data, type='lasso', oneSE=TRUE), rep=1))
  results <- rbind(results, cbind( do_shrinkage(data, type='ridge', oneSE=FALSE), rep=1))
  results <- rbind(results, cbind( do_shrinkage(data, type='ridge', oneSE=TRUE), rep=1))
  #results
  compares <- rbind(compares, cbind(size = sample_size[s],
                                    method = 'Stepwise',
                                    RMSE =  subset(results,method == 'Stepwise')$RMSE ,
                                    num_w = subset(results,method == 'Stepwise')$num_w))
  compares <- rbind(compares, cbind(size = sample_size[s],
                                    method = 'lasso',
                                    RMSE =  subset(results,method == 'lasso')$RMSE ,
                                    num_w = subset(results,method == 'lasso')$num_w))
  compares <- rbind(compares, cbind(size = sample_size[s],
                                    method = 'lasso_oneSE',
                                    RMSE =  subset(results,method == 'lasso_oneSE')$RMSE ,
                                    num_w = subset(results,method == 'lasso_oneSE')$num_w))
```

```
}

compares$RMSE <-  as.numeric(levels(compares$RMSE))[compares$RMSE]
compares
```

```
##    size      method      RMSE num_w
## 1   100    Stepwise 1.0395035     3
## 2   100       lasso 1.0709129     5
## 3   100 lasso_oneSE 1.1624640     0
## 4   300    Stepwise 1.0664420     0
## 5   300       lasso 1.0925216     0
## 6   300 lasso_oneSE 1.0925216     0
## 7   700    Stepwise 0.9582600     0
## 8   700       lasso 0.9726948     3
## 9   700 lasso_oneSE 0.9809432     0
## 10 1500    Stepwise 0.9595969     0
## 11 1500       lasso 0.9736672     4
## 12 1500 lasso_oneSE 0.9736672     0
## 13 3000    Stepwise 1.0036160     0
## 14 3000       lasso 1.0174907     5
## 15 3000 lasso_oneSE 1.0149800     0
```

```
ggplot(data = compares, aes(x = size, y = RMSE) ) +
  geom_line(aes(color = method, group = method)) +
  xlab( 'sample size' ) +
  ylab( 'RMSE' )
```



c.

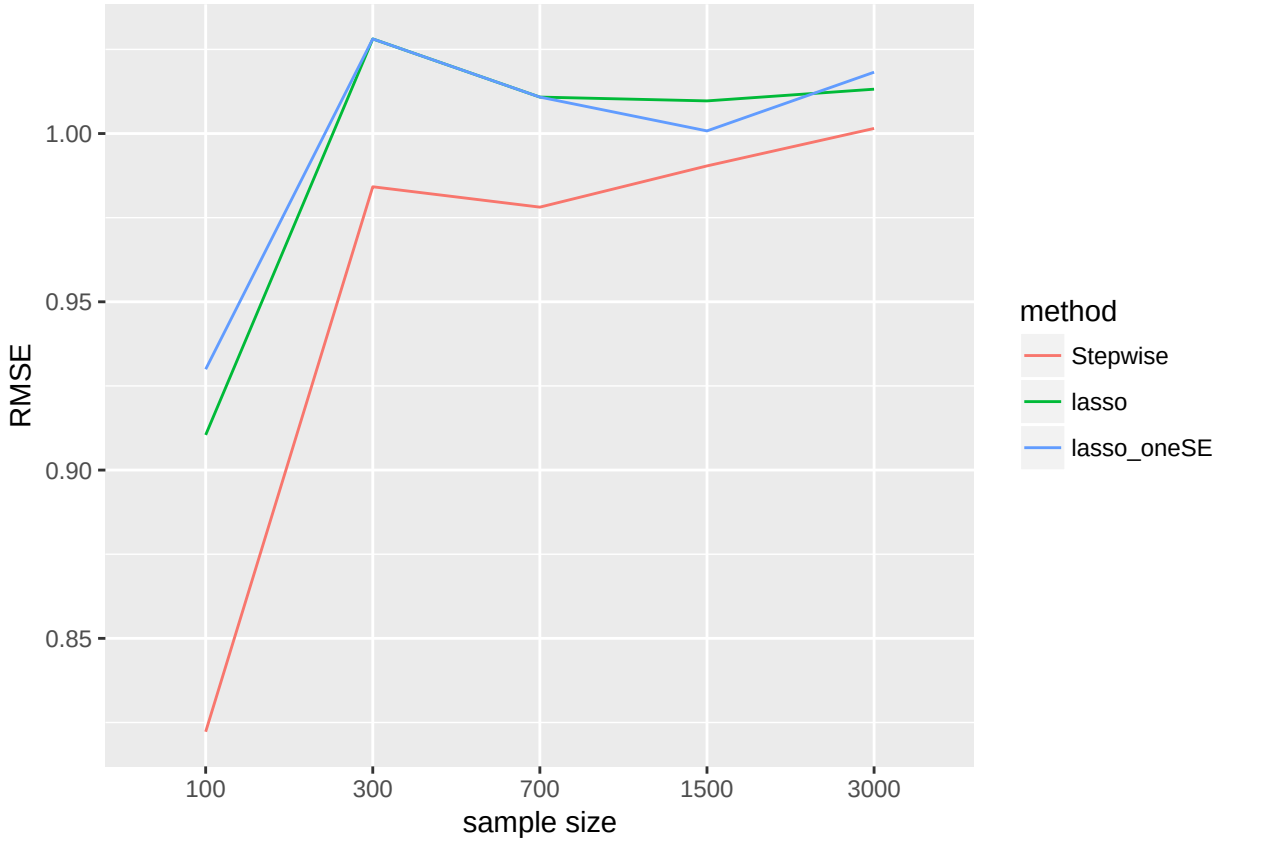
With a medium number of extraneous covariates (num_w=15), I ended up with the following result (see the output of r code).

The RMSE for all models is higher compared to the case with less extraneous covariates. The stepwise method generally produces models with lower RMSE. This means that stepwise model fits the data better. However, if we take "num_w" into consideration, lasso_oneSE method tends to produce models closer to the true model (with a lower num_w value).

```
compares <- data.frame()
sample_size <- c(100,300,700,1500,3000)
for(s in 1:length(sample_size)){
  results <- data.frame()
  data <- make_data(n=sample_size[s], num_w=15)
  results <- rbind(results, cbind( do_stepwise(data), rep=1) )
  results <- rbind(results, cbind( do_shrinkage(data, type='lasso', oneSE=FALSE), rep=1))
  results <- rbind(results, cbind( do_shrinkage(data, type='lasso', oneSE=TRUE), rep=1))
  results <- rbind(results, cbind( do_shrinkage(data, type='ridge', oneSE=FALSE), rep=1))
  results <- rbind(results, cbind( do_shrinkage(data, type='ridge', oneSE=TRUE), rep=1))
  #results
  compares <- rbind(compares, cbind(size = sample_size[s],
                                    method = 'Stepwise',
                                    RMSE = subset(results,method == 'Stepwise')$RMSE,
                                    num_w = subset(results,method == 'Stepwise')$num_w))
  compares <- rbind(compares, cbind(size = sample_size[s],
                                    method = 'lasso',
                                    RMSE = subset(results,method == 'lasso')$RMSE,
                                    num_w = subset(results,method == 'lasso')$num_w))
  compares <- rbind(compares, cbind(size = sample_size[s],
                                    method = 'lasso_oneSE',
                                    RMSE = subset(results,method == 'lasso_oneSE')$RMSE,
                                    num_w = subset(results,method == 'lasso_oneSE')$num_w))

}
compares$RMSE <-  as.numeric(levels(compares$RMSE))[compares$RMSE]
compares
```

```
##    size      method      RMSE num_w
## 1   100    Stepwise 0.8222538     2
## 2   100       lasso 0.9104484     2
## 3   100 lasso_oneSE 0.9299631     0
## 4   300    Stepwise 0.9841865     3
## 5   300       lasso 1.0281111     9
## 6   300 lasso_oneSE 1.0281111     1
## 7   700    Stepwise 0.9781331     3
## 8   700       lasso 1.0108305     5
## 9   700 lasso_oneSE 1.0108305     0
## 10 1500    Stepwise 0.9903775     2
## 11 1500       lasso 1.0096945     4
## 12 1500 lasso_oneSE 1.0007791     1
## 13 3000    Stepwise 1.0015104     3
## 14 3000       lasso 1.0131780     9
## 15 3000 lasso_oneSE 1.0182252     0
```

```r
ggplot(data = compares, aes(x = size, y = RMSE) ) +
  geom_line(aes(color = method, group = method)) +
  xlab( 'sample size' ) +
  ylab( 'RMSE' )
```



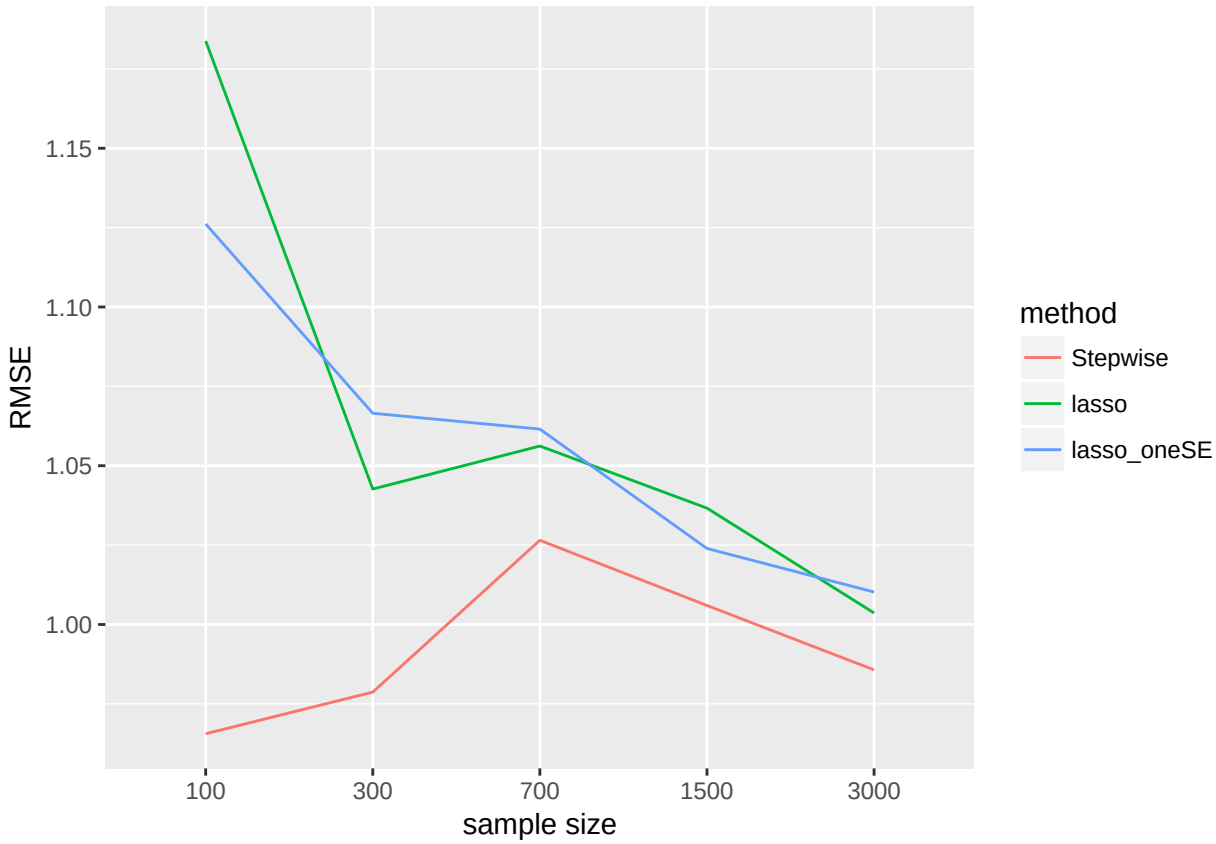
We repeat the same procedure for a larger "num_w", let $num\_w = 50$. We ended up with even higher general RMSE. The results are similar to the previous test case, stepwise model produces a lower RMSE and fit the data better than other models. Lasso_oneSE performs better than Lasso, it gives lower RMSE than lasso and results in better(lower) "num_w" than stepwise model.

```r
compares <- data.frame()
sample_size <- c(100,300,700,1500,3000)
for(s in 1:length(sample_size)){
  results <- data.frame()
  data <- make_data(n=sample_size[s], num_w=50)
  results <- rbind(results, cbind( do_stepwise(data), rep=1) )
  results <- rbind(results, cbind( do_shrinkage(data, type='lasso', oneSE=FALSE), rep=1))
  results <- rbind(results, cbind( do_shrinkage(data, type='lasso', oneSE=TRUE), rep=1))
  results <- rbind(results, cbind( do_shrinkage(data, type='ridge', oneSE=FALSE), rep=1))
  results <- rbind(results, cbind( do_shrinkage(data, type='ridge', oneSE=TRUE), rep=1))
  #results
  compares <- rbind(compares, cbind(size = sample_size[s],
                                    method = 'Stepwise',
                                    RMSE = subset(results,method == 'Stepwise')$RMSE,
                                    num_w = subset(results,method == 'Stepwise')$num_w))
  compares <- rbind(compares, cbind(size = sample_size[s],
                                    method = 'lasso',
```

```
                                    RMSE = subset(results,method == 'lasso')$RMSE,
                                    num_w = subset(results,method == 'lasso')$num_w))
  compares <- rbind(compares, cbind(size = sample_size[s],
                                    method = 'lasso_oneSE',
                                    RMSE = subset(results,method == 'lasso_oneSE')$RMSE,
                                    num_w = subset(results,method == 'lasso_oneSE')$num_w))

}
compares$RMSE <-  as.numeric(levels(compares$RMSE))[compares$RMSE]
compares
```

```
##    size      method      RMSE num_w
## 1   100    Stepwise 0.9655904    13
## 2   100       lasso 1.1838117     2
## 3   100 lasso_oneSE 1.1261581     1
## 4   300    Stepwise 0.9786979     4
## 5   300       lasso 1.0426563     6
## 6   300 lasso_oneSE 1.0665345     0
## 7   700    Stepwise 1.0265217     5
## 8   700       lasso 1.0562094     3
## 9   700 lasso_oneSE 1.0615432     0
## 10 1500    Stepwise 1.0059786     5
## 11 1500       lasso 1.0366456     3
## 12 1500 lasso_oneSE 1.0239336     0
## 13 3000    Stepwise 0.9857036    12
## 14 3000       lasso 1.0036361     3
## 15 3000 lasso_oneSE 1.0102226     0
```

```
ggplot(data = compares, aes(x = size, y = RMSE) ) +
  geom_line(aes(color = method, group = method)) +
  xlab( 'sample size' ) +
  ylab( 'RMSE' )
```



d.

When $num\_w > n$, we have more predictors than observations. Stepwise model does not work in this case, but other models are still useful. This implicates that Lasso and Ridge regression have a better generalization ability. And also the stepwise model is derived from standard linear model theory, which is not true in lots of real life database. These two facts indicate that Lasso model is often preferred than stepwise model.

```
results <- data.frame()
data <- make_data(n=200, num_w=200)
#results <- rbind(results, cbind( do_stepwise(data), rep=1) )
results <- rbind(results, cbind( do_shrinkage(data, type='lasso', oneSE=FALSE), rep=1))
results <- rbind(results, cbind( do_shrinkage(data, type='lasso', oneSE=TRUE), rep=1))
results <- rbind(results, cbind( do_shrinkage(data, type='ridge', oneSE=FALSE), rep=1))
results <- rbind(results, cbind( do_shrinkage(data, type='ridge', oneSE=TRUE), rep=1))
results
```

```
##        method num_x num_w     RMSE rep
## 1       lasso     5    12 1.085671   1
## 2 lasso_oneSE     5     0 1.096934   1
## 3       ridge     5   200 1.886794   1
## 4 ridge_oneSE     5   200 2.097649   1
```

3

```
# web address is too long, so I split it across multiple lines...
file <- paste('http://scrippsco2.ucsd.edu', '/assets/data/atmospheric/stations/',
              'in_situ_co2/monthly/monthly_in_situ_co2_mlo.csv',
sep='')
columns <- c('Year','Month','Date.Excel','Date','CO2','CO2.Season',
             'fit','fit.adj','CO2.filled','CO2.adj.filled')
co2 <- read_csv(file, skip=57, col_names=columns) %>% dplyr::select(Year, Month, CO2) %>%
mutate(Month = as.integer(Month)) %>%
mutate( Time = Year + (Month-1)/12 )
```

```
## Parsed with column specification:
## cols(
##   Year = col_integer(),
##   Month = col_character(),
##   Date.Excel = col_integer(),
##   Date = col_double(),
##   CO2 = col_double(),
##   CO2.Season = col_double(),
##   fit = col_double(),
##   fit.adj = col_double(),
##   CO2.filled = col_double(),
##   CO2.adj.filled = col_double()
## )
```

```
attr(co2, 'spec') <- NULL # just cleaning up the column names
```

a.

It looks like the general trend is ascending but there's some periodic pattern as well. It's possible that the period for this pattern is exactly one year.

```
library(mgcv)   # for some reason the gam library fails when I build the book.
```

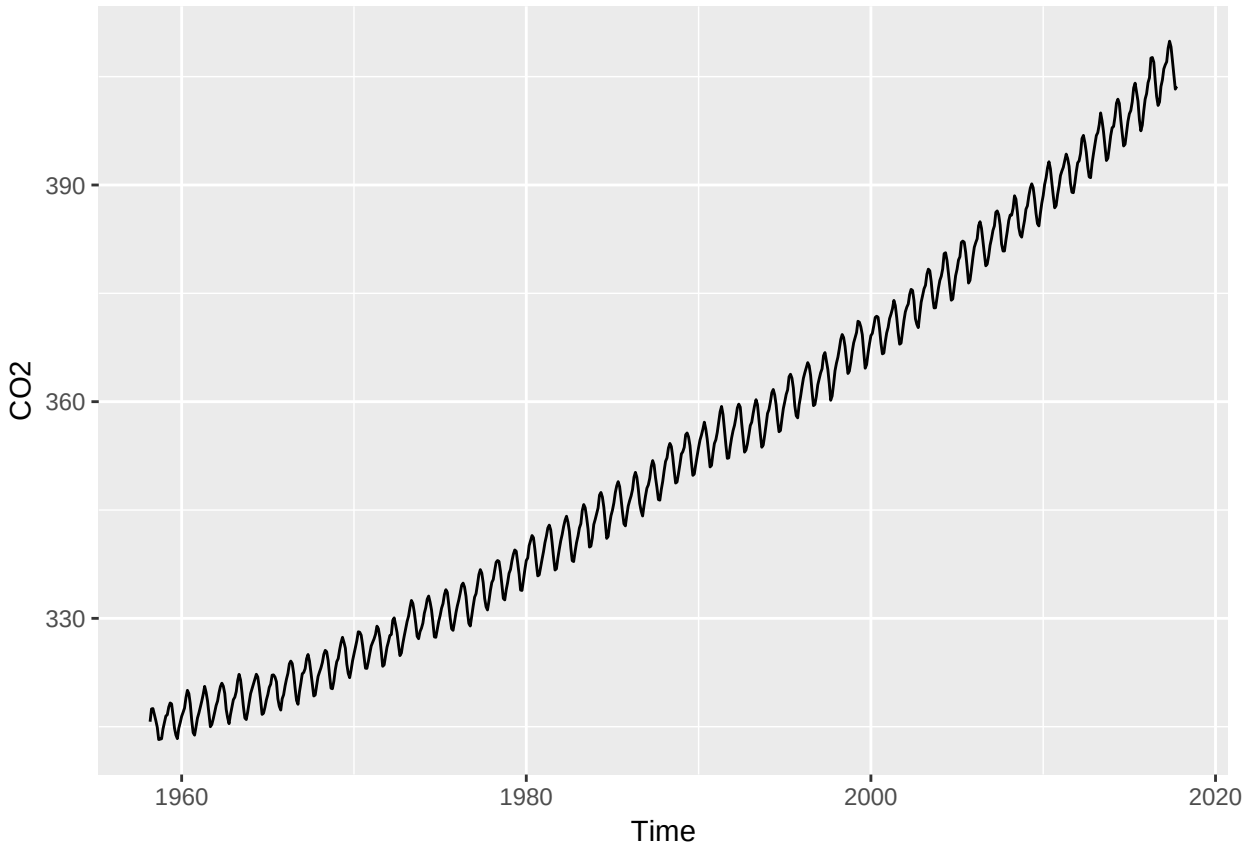
```
## Loading required package: nlme
```

```
##
## Attaching package: 'nlme'
```

```
## The following object is masked from 'package:dplyr':
##
##     collapse
```

```
## This is mgcv 1.8-15. For overview type 'help("mgcv-package")'.
```

```
# from the documentation we know that the NaN value is set as -99.99, so we drop these value first
co2 <- subset(co2,CO2 != -99.99)

ggplot(data = co2, aes(x = Time, y = CO2)) +
  geom_line()
```

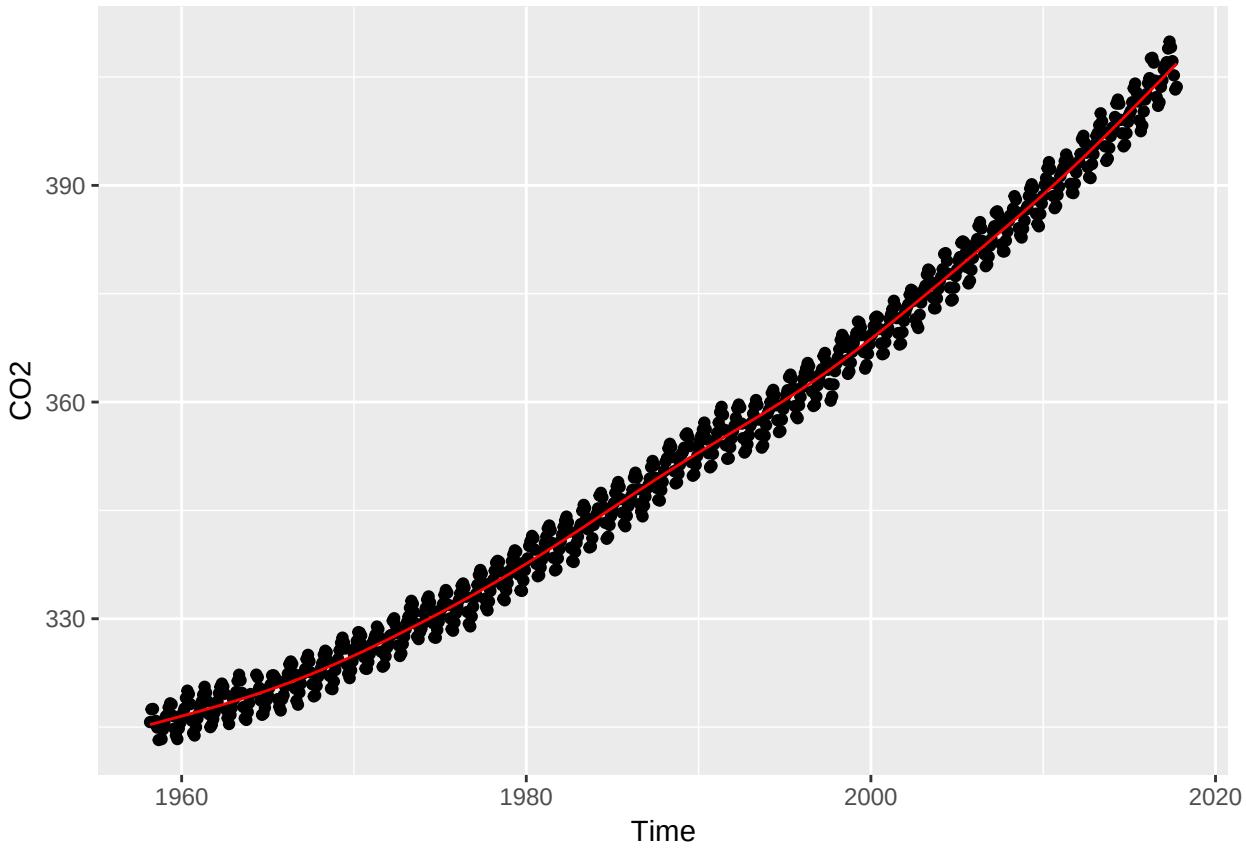


b.

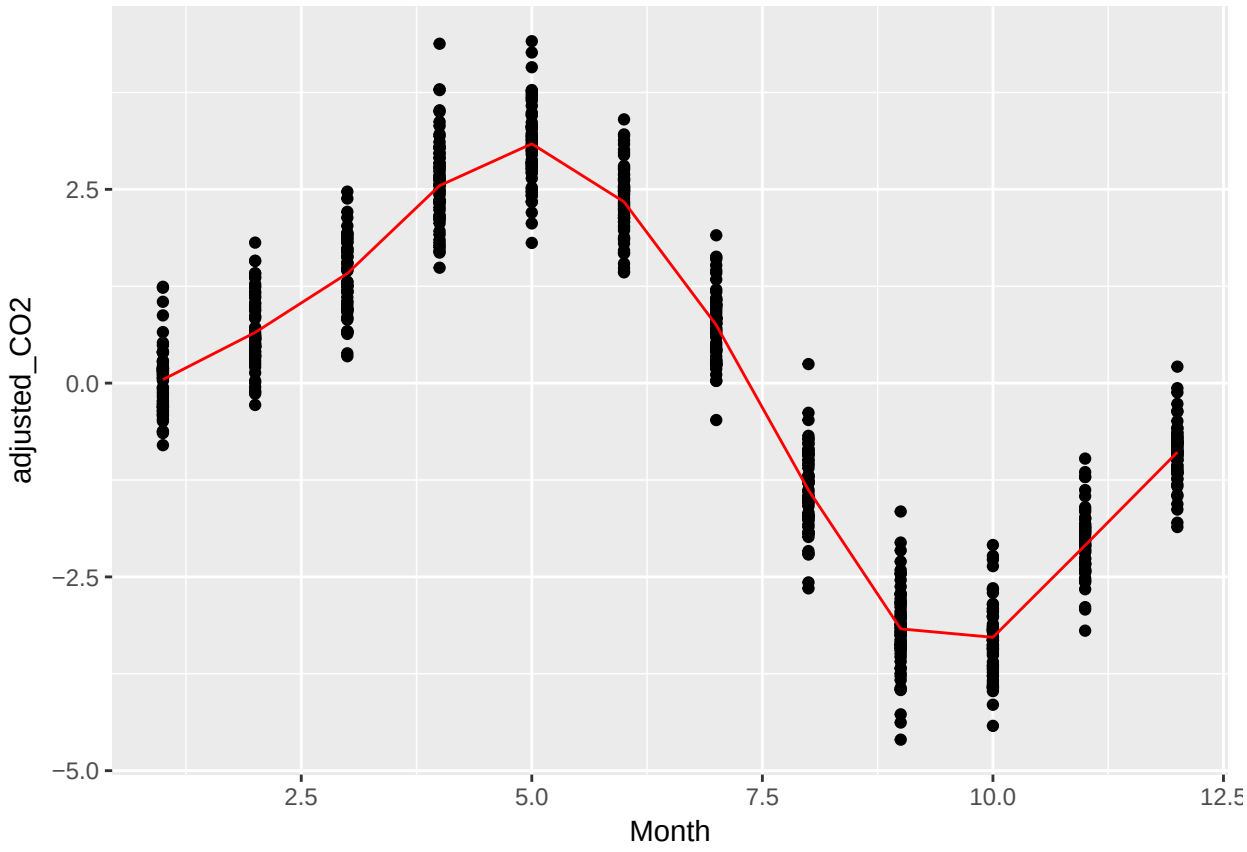
```
# we can first take the year round mean co2 value for each year and model year to year rise
year_co2 <- co2 %>% group_by(Year) %>%
  summarise(year.mean = mean(CO2))
model_yearly <- mgcv::gam(CO2 ~ Year, data=co2)
mgcv::summary.gam(model_yearly)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## CO2 ~ Year
##
## Parametric coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.709e+03  1.695e+01  -159.8   <2e-16 ***
## Year         1.541e+00  8.528e-03   180.6   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## R-sq.(adj) =  0.979   Deviance explained = 97.9%
## GCV = 15.245  Scale est. = 15.202    n = 711
```

```r
# or just fit a smoothing splines to find the general yearly trend
model_year <- gam( CO2 ~ s(Time), data=co2 )
co2$yhat_year <- predict(model_year)
ggplot(co2, aes(x=Time)) +
  geom_point( aes(y=CO2) ) +
  geom_line( aes(y=yhat_year), color='red')
```



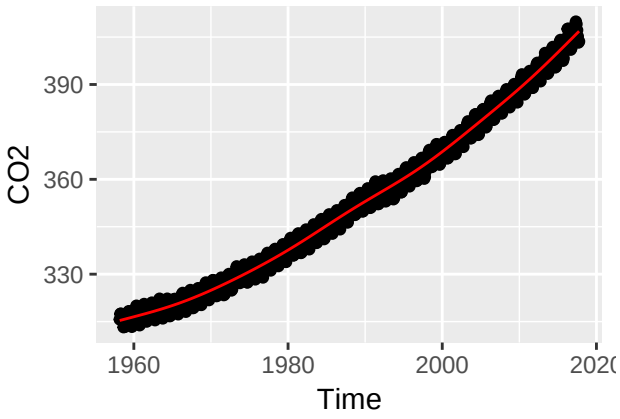

```r
# substract the year to year rise from the data to get the month to month pattern
co2 <- co2 %>% mutate(adjusted_CO2 = CO2 - yhat_year)

model_month <- gam( adjusted_CO2 ~ s(Month), data=co2 )
co2$yhat_month <- predict(model_month)
ggplot(co2, aes(x=Month)) +
  geom_point( aes(y=adjusted_CO2) ) +
  geom_line( aes(y=yhat_month), color='red')
```

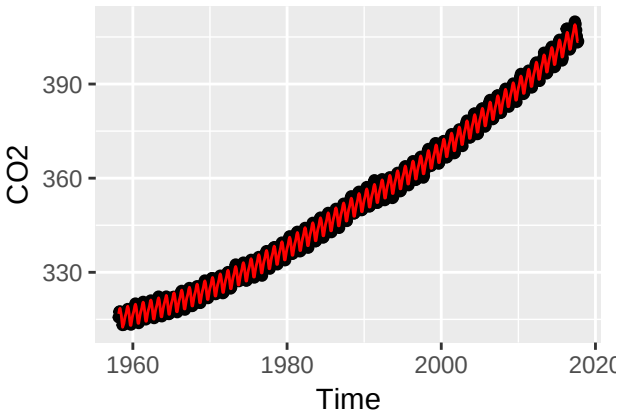


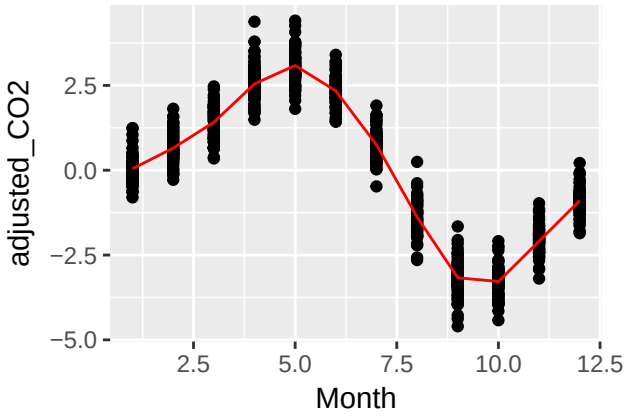
c.

Visualization as follows. We can see from the result that smoothing spline can fit only the year to year rise if applied directly to the time-series. But if we remove the baseline, the month to month pattern can be clearly modeled by GAM as well. Finally, by adding the predicted values by these two models, we can model both detailed periodic pattern and general trend.

```
# this is what we predict combining the two models
co2 <- co2 %>% mutate(yhat = yhat_year + yhat_month)
P <- NULL

P[[1]] <- ggplot(co2, aes(x=Time)) +
  geom_point( aes(y=CO2) ) +
  geom_line( aes(y=yhat_year), color='red')

P[[2]] <- ggplot(co2, aes(x=Month)) +
  geom_point( aes(y=adjusted_CO2) ) +
  geom_line( aes(y=yhat_month), color='red')

P[[3]] <- ggplot(co2, aes(x=Time)) +
  geom_point( aes(y=CO2) ) +
  geom_line( aes(y=yhat), color='red')

Rmisc::multiplot(P[[1]], P[[2]], P[[3]], cols = 2)
```







4

```
# Install the package where the CommunitySurvey data is.
library(devtools)
library(car)
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following object is masked from 'package:purrr':
##
##     some
```

```
install_github('dereksonderegger/dsData')
```

```
## Skipping install of 'dsData' from a github remote, the SHA1 (fad67447) has not changed since last ins
##   Use `force = TRUE` to force installation
```

```
data('CommunitySurvey', package='dsData')
# You might have to restart R to get the documentation to work...
?dsData::CommunitySurvey
```

```
library(vegan); library(tidyverse);
```

```
## Loading required package: permute
```

```
##
## Attaching package: 'permute'
```

```
## The following object is masked from 'package:devtools':
##
##     check
```

```
## Loading required package: lattice
```

```
## This is vegan 2.4-4
```

```
data('CommunitySurvey', package='dsData')
DistMatrix <- CommunitySurvey %>% # create a distance matrix
  select(-Community) %>%          # of just the y variables
  vegdist( ) #
# by default anosim will calculate a p-value using permutations.
# We will suppress that while creating our bootstrap CI to save time
model <- anosim(DistMatrix, CommunitySurvey$Community, permutations=0)
summary(model)
```

```
##
## Call:
## anosim(dat = DistMatrix, grouping = CommunitySurvey$Community,      permutations = 0)
## Dissimilarity: bray
##
## ANOSIM statistic R: 0.8998
##
##
## Dissimilarity ranks between and within classes:
##           0%     25%    50%     75%  100%   N
## Between 69.5 335.125 409.75 488.000 561.0 288
## I        2.0  53.000 117.50 179.000 377.5 153
## II       1.0  95.500 179.00 247.125 493.5 120
```

```
model$statistic
```

```
## [1] 0.8997507
```

a.

```
boot.fn <- function(data, index){
  #first slice the original data
  new_dat <- CommunitySurvey %>% slice(index)
  # create new distance matrix
  new_dist <- new_dat %>%
    select(-Community) %>%
    vegdist()
  # use anosim to calculate R statistic
  model_temp <- anosim(new_dist, new_dat$Community, permutations=0)
  #summary(model_temp)
  return(model_temp$statistic)
}
```

```
index <- c(10,20,30,1,2,3,6,8,9,15,16,17)
boot.fn(CommunitySurvey,index)
```

```
## [1] 0.825
```

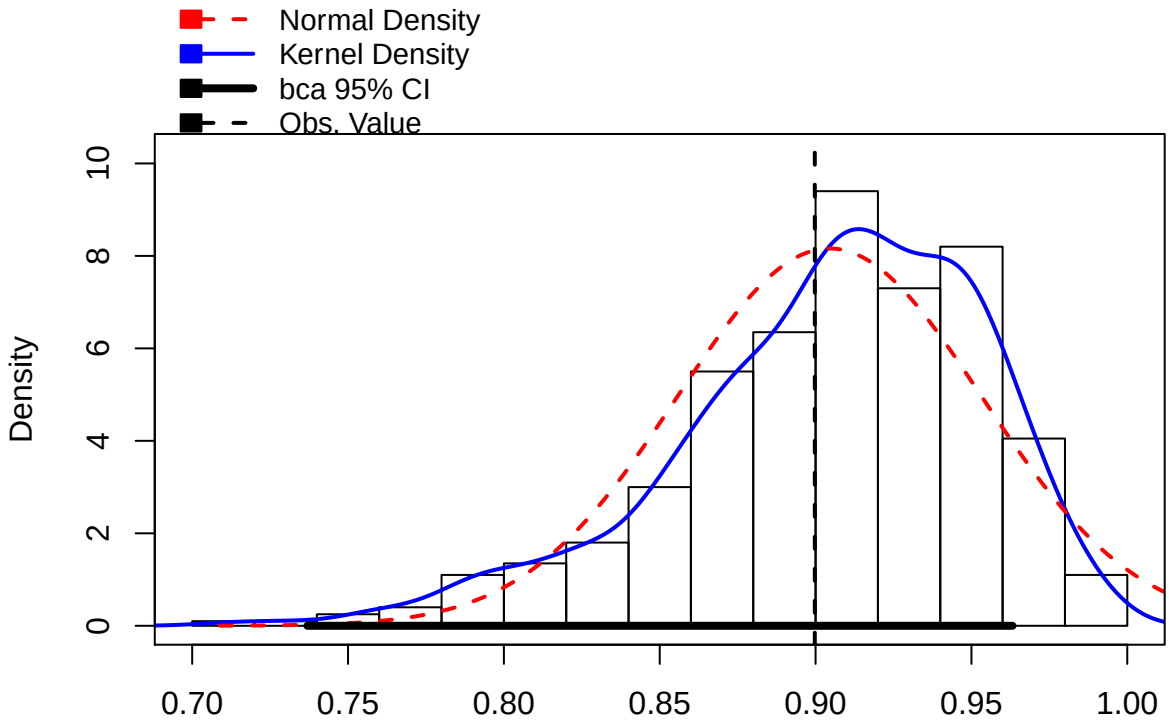
b.

```
boot.model <- boot::boot(CommunitySurvey, boot.fn, R=1000)
```

c.

From the bootstrap histogram of R statistic, we can see that the estimated distribution is skewed. In other words, bootstrap distribution is NOT symmetric. So we cannot use percentile confidence intervals here. It's more appropriate to use basic intervals or BCa.

```
hist(boot.model)
```



d.

The 95% confident interval is $[0.7114, 0.9667]$. So zero is not a reasonable value, test statistic is not centered at zero. We can infer that there is association between species abundance and community type.

```
confint(boot.model)
```

```
## Bootstrap quantiles, type =  bca
##
##       2.5 %    97.5 %
## 1 0.7369611 0.9631926
```