Appendix

Load data. Group variables. Perform transformations.

```
#setwd("/Users/njosephs/Desktop/PhD/575/final project/")
setwd("~/Desktop/github/LM_Project/data/")
fires <- read.csv("forestfires.csv")

# create the indicators for weekend and summer
wkd <- rep(0, nrow(fires))
wkd[fires$day %in% c("fri", "sat", "sun")] <- 1
wkdM <- rep(0,nrow(fires))
wkdM[fires$day %in% c("fri", "sat", "sun", "mon")] <- 1
summer <- rep(0, nrow(fires))
summer[fires$month %in% c("jun", "jul", "aug", "sep")] <- 1

fires$wkd <- wkd
fires$wkdM <- wkdM
fires$summer <- summer

# transform the area
areaTrans <- log(fires$area + 1)
fires$areaTrans <- areaTrans

# rain indicator
rainvnorain <- rep(0, nrow(fires))
rainvnorain[fires$rain != 0] <- 1
fires$rainvnorain <- rainvnorain

# wetness metric
rel_humid100_temp <- c(0, 5, 10, 15, 17, 19, 20, 22, 24, 26, 29, 32, 35)
rel_humid100_water <- c(4.2, 5.74, 7.84, 10.7, 12.12, 13.73, 14.62, 16.56, 18.76, 21.25, 25.62, 30.89, 3

water_at_full <- approxfun(x = rel_humid100_temp, y = rel_humid100_water )
est_wetness_metric <- function(Temp,rh){
  return(water_at_full(Temp)*rh)
}
wetness <- est_wetness_metric(fires$temp, fires$RH)

fires$wetness <- wetness

# simplified FFMC
fires$tFFMC <- ifelse(fires$FFMC < 80, 0, 1)

# indicator for forest
forest_coords <- c(1, 1, 1, 1, 1, 1, 1, 1, 1
                   , 0, 0, 1, 1, 1, 0, 1, 1, 1
                   , 0, 0, 1, 0, 0, 0, 1, 1, 0
                   , 0, 0, 1, 0, 0, 0, 1, 1, 0
                   , 0, 1, 0, 0, 1, 1, 1, 1, 1
                   , 0, 0, 0, 1, 0, 0, 0, 0, 1
                   , 1, 1, 1, 1, 0, 0, 0, 0, 1
                   , 1, 1, 1, 1, 1, 0, 0, 0, 0
```

```
                  , 1, 1, 1, 1, 1, 0, 0, 0, 0)
forest_coords <- matrix(forest_coords, nrow = 9, ncol = 9)
for(i in 1:nrow(fires)){
  fires[i, "forest_ind"] <- forest_coords[fires[i, "X"], fires[i, "Y"]]
}

# geo-spatial grid
fires[, "grid_group"] <- "other"                       # default (other)
fires[fires$X %in% c(1, 2, 3) &
        fires$Y %in% c(2, 3, 4), "grid_group"] <- "tl" # top left mountain
fires[fires$X %in% c(3, 4, 5) &
        fires$Y %in% c(3, 4, 5), "grid_group"] <- "ml" # middle left mountain
fires[fires$X %in% c(5, 6, 7) &
        fires$Y %in% c(3, 4, 5), "grid_group"] <- "mr" # middle right mountain
fires[fires$X %in% c(7, 8) &
        fires$Y %in% c(6, 7), "grid_group"] <- "br"    # bottom right mountain

# transform response variable (ISI)
fires$sqISI <- sqrt(fires$ISI)

#  create train/test split
set.seed(575)
train.ind <- sample.int(n = nrow(fires), size = floor(nrow(fires) * 0.7), replace = FALSE)
train<- fires[train.ind, ]
test <- fires[-train.ind, ]
```

Create data-driven variables.

```
# calculate the groupings of FFMC by training quantile
ffmcQuant <- quantile(train$FFMC, probs = seq(0, 1, .1))
FFMCQuantile_train <- rep(0, nrow(train))
FFMCQuantile_test <- rep(0, nrow(test))
for (i in 10) {
  FFMCQuantile_train[ffmcQuant[i] < train$FFMC &
                train$FFMC <= ffmcQuant[i + 1]] <- i
  FFMCQuantile_test[ffmcQuant[i] < test$FFMC &
                test$FFMC <= ffmcQuant[i + 1]] <- i
}

train$FFMCQuantile <- FFMCQuantile_train
test$FFMCQuantile <- FFMCQuantile_test

# condense X-Y grid
train$X2 <- train$X
train$Y2 <- train$Y
i <- 0
while (i < 1) {
  m <- as.matrix(table(train$Y2, train$X2))
  top <- sum(m[rownames(m) == min(rownames(m)), ])
  bottom <- sum(m[rownames(m) == max(rownames(m)), ])
  left <- sum(m[, colnames(m) == min(colnames(m))])
  right <- sum(m[, colnames(m) == max(colnames(m))])

  if (top == min(top, bottom, left, right)) {
```

```
    train[train$Y2 == min(rownames(m)), "Y2"] <- as.integer(min(rownames(m))) + 1
    if (min(prop.table(table(train$Y2, train$X2))) < .01) {
      i = 0
    } else{
      i = 1
    }
  } else if (bottom == min(top, bottom, left, right)) {
    train[train$Y2 == max(rownames(m)), "Y2"] <- as.integer(max(rownames(m))) - 1
    if (min(prop.table(table(train$Y2, train$X2))) < .01) {
      i = 0
    } else{
      i = 1
    }
  } else if (left == min(top, bottom, left, right)) {
    train[train$X2 == min(colnames(m)), "X2"] <- as.integer(min(colnames(m))) + 1
    if (min(prop.table(table(train$Y2, train$X2))) < .01) {
      i = 0
    } else{
      i = 1
    }
  } else {
    train[train$X2 == max(colnames(m)), "X2"] <- as.integer(max(colnames(m))) - 1
    if (min(prop.table(table(train$Y2, train$X2))) < .01) {
      i = 0
    } else{
      i = 1
    }
  }
}
```

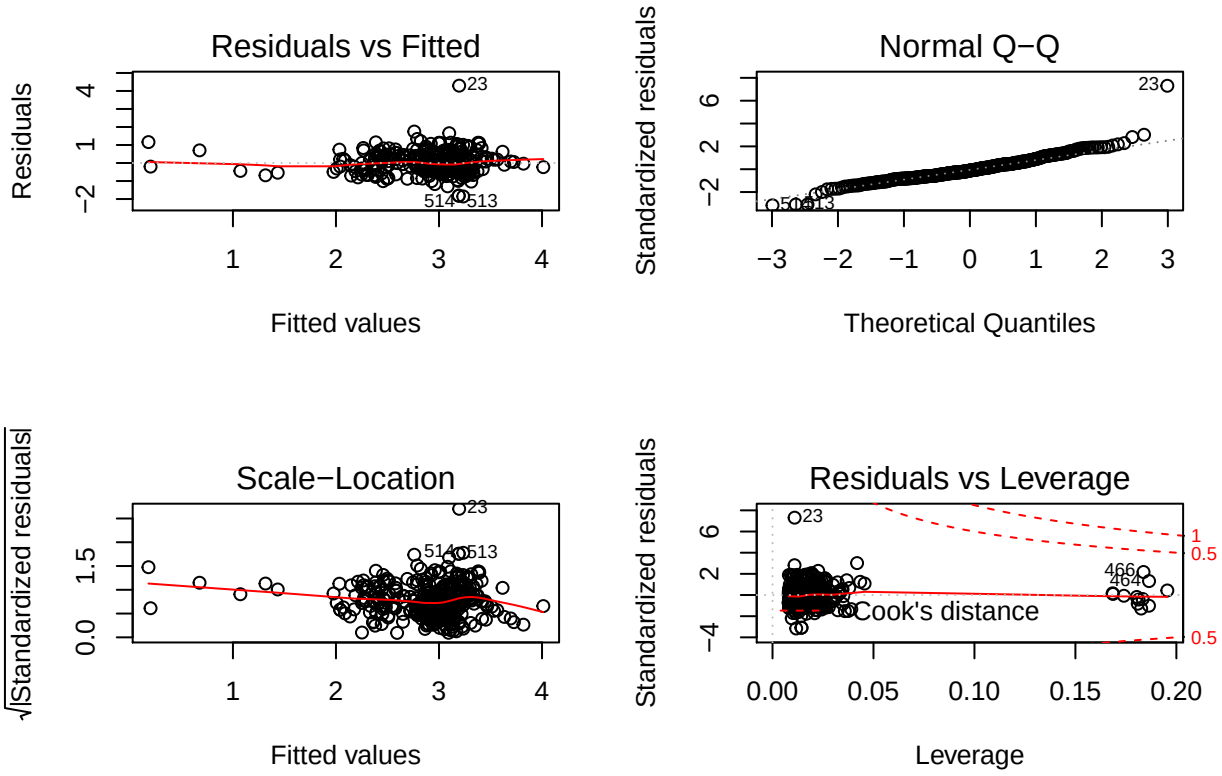
Perform variable selection using LASSO.

```
# cast as factors
vars_factors <- c("wkd", "wkdM", "summer", "FFMCQuantile", "rainvnorain", "grid_group", "month", "day",
for(var in vars_factors) {
  train[, var] <- as.factor(train[, var])
}

# construct regression equation
f <- formula(sqISI ~
               tFFMC
             + X2
             + Y2
             + temp
             + RH
             + wind
             + wkd
             + summer
             + rainvnorain
             + forest_ind
             + DMC
             + DC
             + areaTrans
             + wetness
```

```
)
# build model matrix
X <- model.matrix(f, train)
Y <- as.matrix(train$sqISI)
a <- 1

# run lasso
cv = cv.glmnet(X, Y, alpha = a)
lambda_opt = cv$lambda.min

lasso <- glmnet(X, Y, alpha = a, lambda = lambda_opt)
tmp <- sort(abs(coef(lasso)[, 1]), decreasing = TRUE)
varImp <- data.frame(VarNames = names(tmp), Beta = round(as.vector(tmp), 3))
varImp
```

```
##        VarNames  Beta
## 1         tFFMC 1.511
## 2  rainvnorain1 0.662
## 3       summer1 0.630
## 4   (Intercept) 0.273
## 5           X25 0.264
## 6    forest_ind 0.144
## 7           X23 0.110
## 8           X24 0.101
## 9           X27 0.076
## 10         wind 0.072
## 11          Y25 0.069
## 12         temp 0.035
## 13         wkd1 0.020
## 14    areaTrans 0.010
## 15          DMC 0.001
## 16      wetness 0.000
## 17           DC 0.000
## 18  (Intercept) 0.000
## 19          X26 0.000
## 20           RH 0.000
```

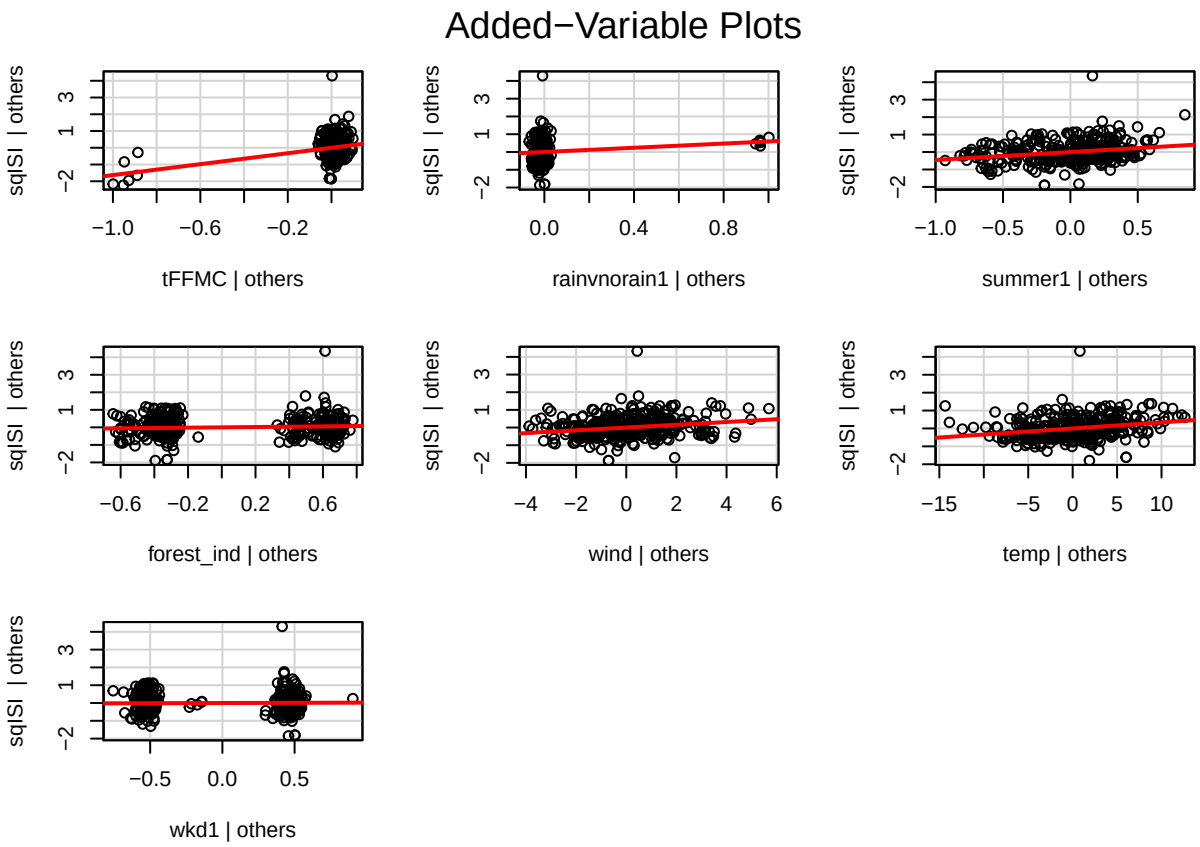
```
# fit lm with most important variables
m <- lm(sqISI ~
          tFFMC
        + rainvnorain
        + summer
        + forest_ind
        + wind
        + temp
        + wkd
        , data = train)

# diagnostics and added variable plots
par(mfrow = c(2,2))
plot(m)
```




```
summary(m)
```

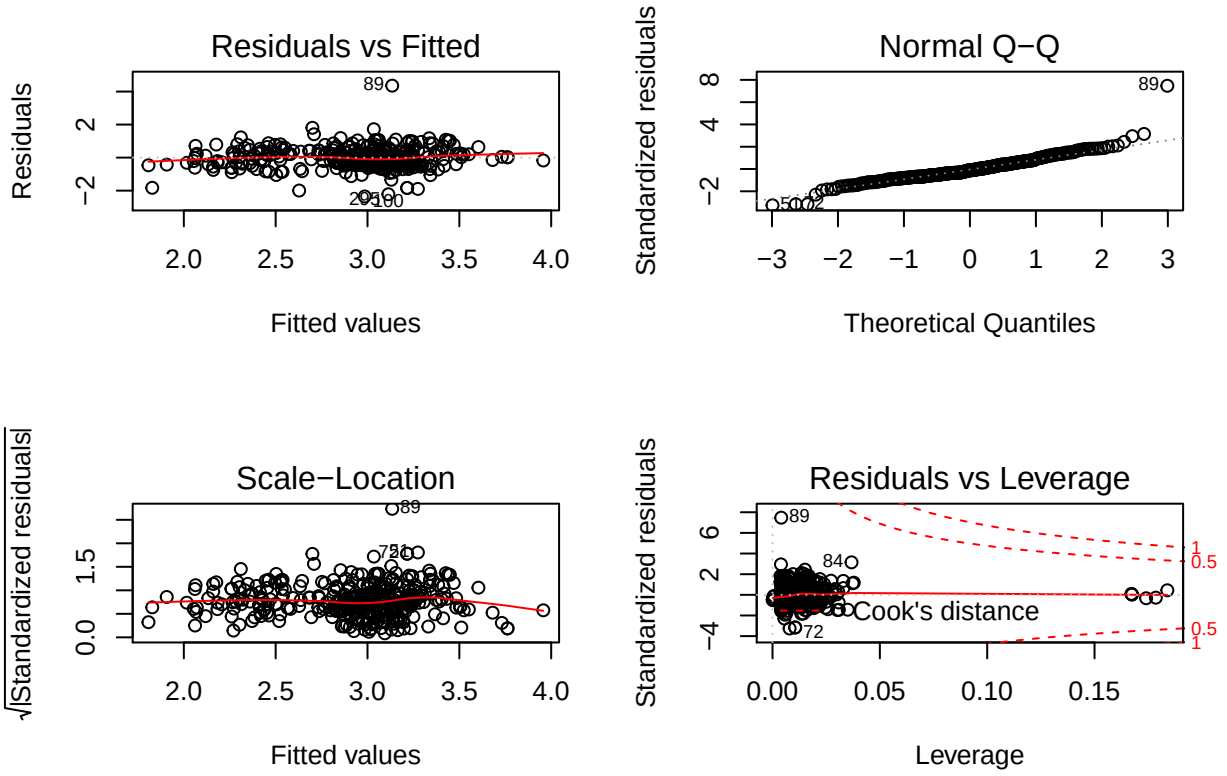
```
##
## Call:
## lm(formula = sqISI ~ tFFMC + rainvnorain + summer + forest_ind +
##     wind + temp + wkd, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.8571 -0.3710 -0.0623  0.3038  4.2941
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.067354   0.263243  -0.256   0.7982
## tFFMC         1.630545   0.249684   6.530 2.29e-10 ***
## rainvnorain1  0.593138   0.245501   2.416   0.0162 *
## summer1       0.454792   0.096851   4.696 3.81e-06 ***
## forest_ind    0.092883   0.065020   1.429   0.1540
## wind          0.078533   0.018721   4.195 3.46e-05 ***
## temp          0.033832   0.006947   4.870 1.69e-06 ***
## wkd1          0.021140   0.062893   0.336   0.7370
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5909 on 353 degrees of freedom
## Multiple R-squared:  0.3641, Adjusted R-squared:  0.3515
## F-statistic: 28.87 on 7 and 353 DF,  p-value: < 2.2e-16
```

```
avPlots(m)
```



After throwing out the noise (forest_ind and wkd), we have 5 strong covariates. The non-linear relationship between ISI and FFMC suggests two working models: with and without tFFMC. Here, we consider the simplier model without tFFMC, where instead we weight by tFFMC.

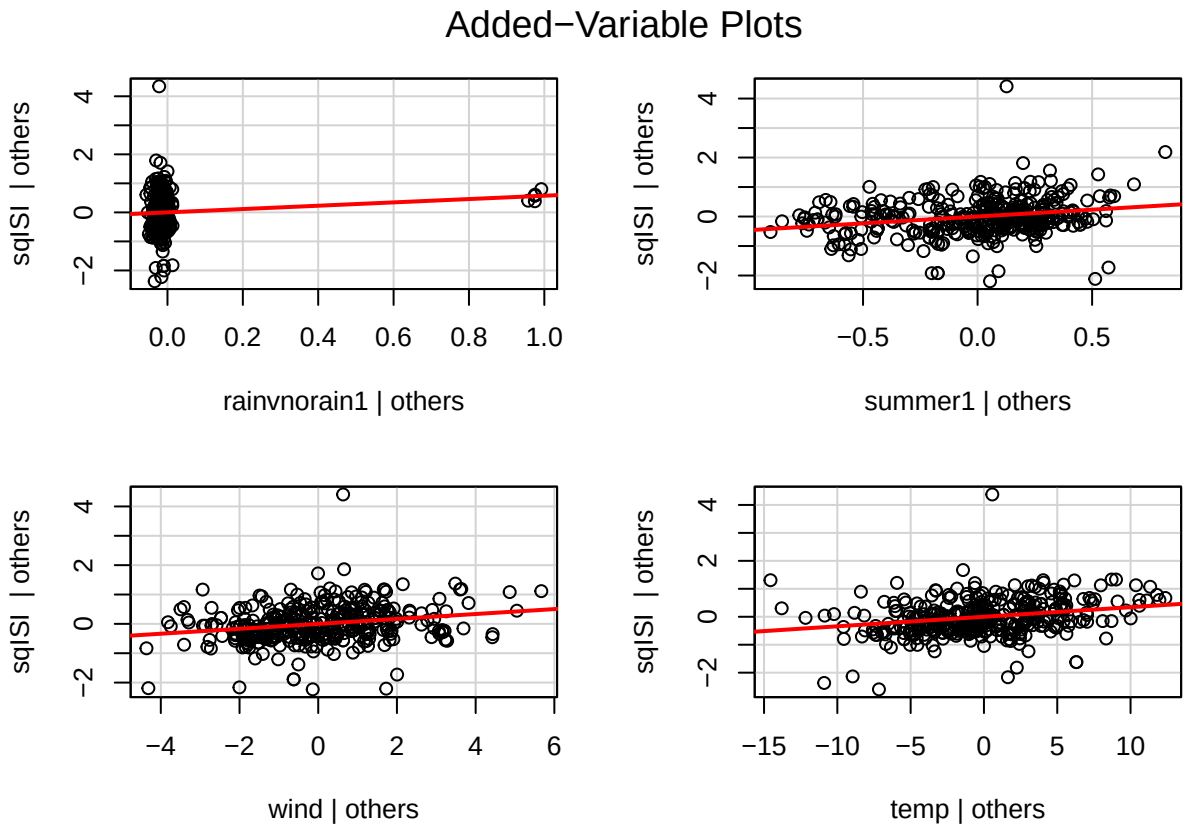
```
# Weight based on tFFMC
train <- data.frame(train %>% group_by(tFFMC) %>% mutate(weight = n()))
m_weight <- lm(sqISI ~
                 rainvnorain
               + summer
               + wind
               + temp
               , weights = weight
               , data = train)
par(mfrow = c(2,2))
plot(m_weight)
```




```
summary(m_weight)
```

```
##
## Call:
## lm(formula = sqISI ~ rainvnorain + summer + wind + temp, data = train,
##     weights = weight)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -35.722  -6.969  -1.309   5.985  82.064
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.574888   0.148493  10.606  < 2e-16 ***
## rainvnorain1 0.573597   0.241519   2.375   0.0181 *
## summer1      0.464246   0.096105   4.831 2.03e-06 ***
## wind         0.083909   0.018773   4.470 1.05e-05 ***
## temp         0.034180   0.006898   4.955 1.12e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.01 on 356 degrees of freedom
## Multiple R-squared:  0.2701, Adjusted R-squared:  0.2619
## F-statistic: 32.93 on 4 and 356 DF,  p-value: < 2.2e-16
```

```
avPlots(m_weight)
```

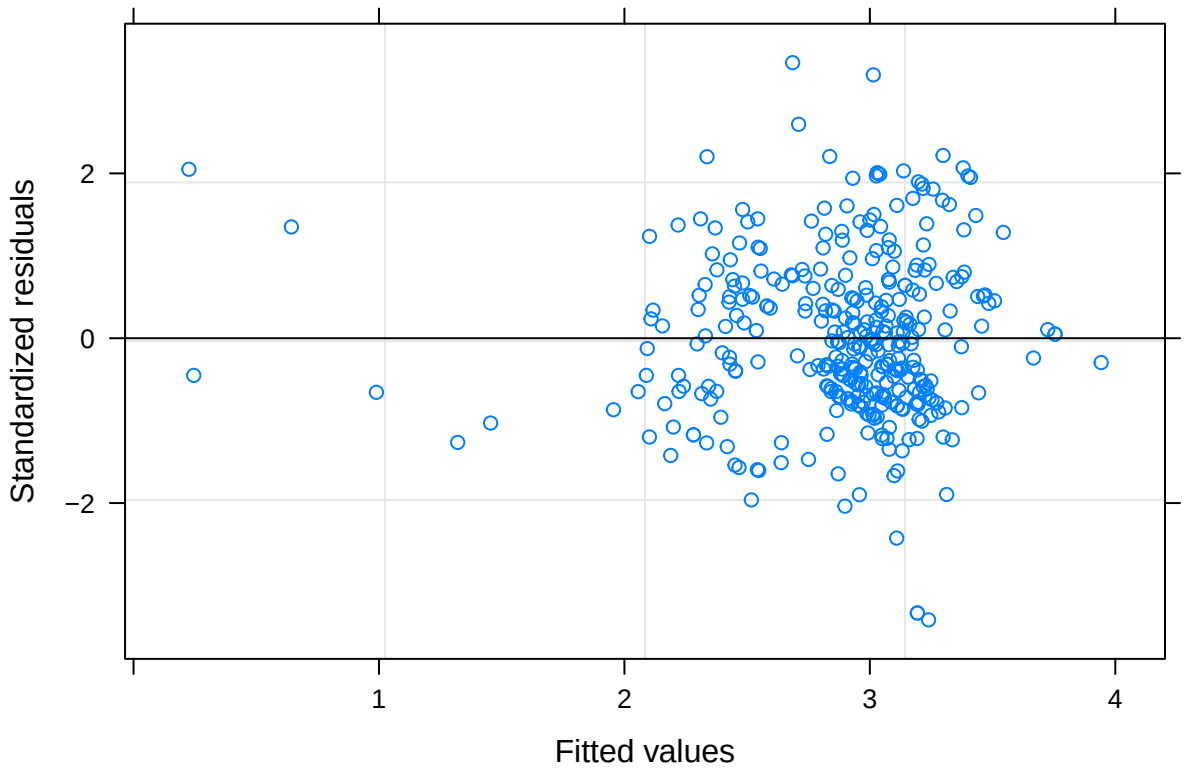



```
# Random intercepts with tFFMC
train[train$X2 == 2 & train$Y2 == 4, "region"] = 1
train[train$X2 == 3 & train$Y2 == 4, "region"] = 2
train[train$X2 == 4 & train$Y2 == 4, "region"] = 3
train[train$X2 == 5 & train$Y2 == 4, "region"] = 4
train[train$X2 == 6 & train$Y2 == 4, "region"] = 5
train[train$X2 == 7 & train$Y2 == 4, "region"] = 6

train[train$X2 == 2 & train$Y2 == 5, "region"] = 7
train[train$X2 == 3 & train$Y2 == 5, "region"] = 8
train[train$X2 == 4 & train$Y2 == 5, "region"] = 9
train[train$X2 == 5 & train$Y2 == 5, "region"] = 10
train[train$X2 == 6 & train$Y2 == 5, "region"] = 11
train[train$X2 == 7 & train$Y2 == 5, "region"] = 12

train$region = as.factor(train$region)


m_rand <- lme(sqISI ~ summer + wind + temp + rainvnorain + tFFMC
              , random = ~1|region
              , data =  train[-89, ]
              , method = "REML")
summary(m_rand)
```

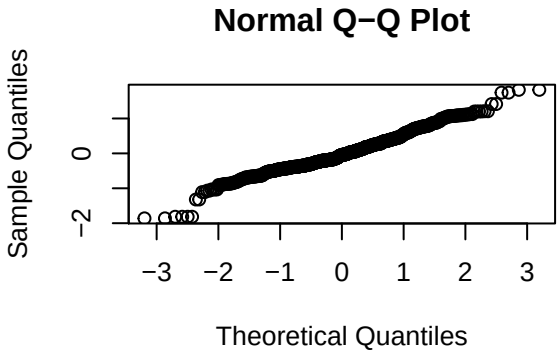
```
## Linear mixed-effects model fit by REML
##  Data: train[-89, ]
##        AIC      BIC    logLik
##   619.5402 650.4946 -301.7701
```

```
##
## Random effects:
##  Formula: ~1 | region
##          (Intercept) Residual
## StdDev: 1.793574e-05 0.544431
##
## Fixed effects: sqISI ~ summer + wind + temp + rainvnorain + tFFMC
##                  Value  Std.Error  DF  t-value p-value
## (Intercept)  0.0044579 0.23905632 343 0.018648  0.9851
## summer1      0.4271146 0.08886132 343 4.806530  0.0000
## wind         0.0773203 0.01720119 343 4.495053  0.0000
## temp         0.0329559 0.00638748 343 5.159451  0.0000
## rainvnorain1 0.5948855 0.22507826 343 2.643016  0.0086
## tFFMC        1.6372146 0.22964015 343 7.129479  0.0000
##  Correlation:
##              (Intr) summr1 wind   temp   rnvnr1
## summer1      -0.025
## wind         -0.306  0.007
## temp         -0.206 -0.610  0.214
## rainvnorain1  0.026 -0.024 -0.087 -0.001
## tFFMC        -0.832  0.042 -0.099 -0.190 -0.010
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -3.41746546 -0.66518568 -0.08520754  0.61794330  3.34336658
##
## Number of Observations: 360
## Number of Groups: 12
```

```
plot(m_rand)
```

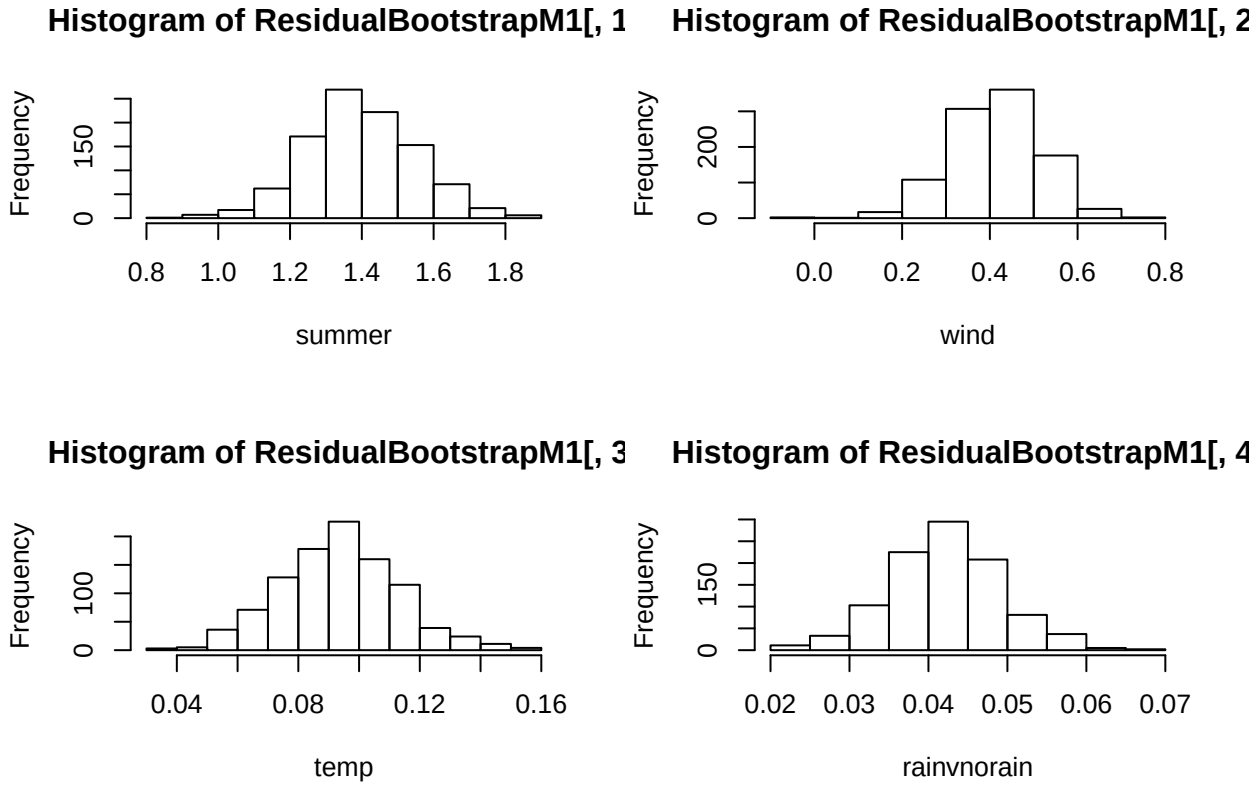



```
qqnorm(m_rand$residuals)
```




```
#build final model
m1 = lm(sqISI ~ summer + wind + temp + rainvnorain
        #,data = train[-89,])
       ,data = train)

#1000 boostrap samples for each \beta
B <- 1000
ResidualBootstrapM1 <-t( replicate(B, {
  #yb <- fitted(m1) + resid(m1)[sample.int(nrow(train[-89,]), replace = TRUE)]
  yb <- fitted(m1) + resid(m1)[sample.int(nrow(train), replace = TRUE)]
  boot <- model.matrix(m1)
  coef(lm(yb ~ boot - 1))
}))

#look at beta distributions
par(mfrow=c(2,2))
```

```r
hist(ResidualBootstrapM1[,1], xlab = "summer")
hist(ResidualBootstrapM1[,2], xlab ="wind")
hist(ResidualBootstrapM1[,3], xlab ="temp")
hist(ResidualBootstrapM1[,4], xlab ="rainvnorain")
```




```r
#empirical CI
t(apply(ResidualBootstrapM1, 2, quantile, c(.025, .975)))
```

```
##                         2.5%      97.5%
## boot(Intercept)  1.10171852 1.70125520
## bootsummer1      0.21177705 0.60609326
## bootwind         0.05535082 0.13311909
## boottemp         0.02781247 0.05668135
## bootrainvnorain1 0.10702560 1.14043899
```