Homework09

Philip Anderson; panders2@tamu.edu

3/21/2018

Question 1

Chapter 3 Exercise 1A

```
library("tidyverse")
library("AER")
data("HousePrices")
library("mgcv")
# take a look to see variable listing and format
str(HousePrices)
```

```
## 'data.frame':    546 obs. of  12 variables:
##  $ price     : num  42000 38500 49500 60500 61000 66000 66000 69000 83800 88500 ...
##  $ lotsize   : num  5850 4000 3060 6650 6360 4160 3880 4160 4800 5500 ...
##  $ bedrooms  : num  3 2 3 3 2 3 3 3 3 3 ...
##  $ bathrooms : num  1 1 1 1 1 1 2 1 1 2 ...
##  $ stories   : num  2 1 1 2 1 1 2 3 1 4 ...
##  $ driveway  : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ recreation: Factor w/ 2 levels "no","yes": 1 1 1 2 1 2 1 1 2 2 ...
##  $ fullbase  : Factor w/ 2 levels "no","yes": 2 1 1 1 1 2 2 1 2 1 ...
##  $ gasheat   : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ aircon    : Factor w/ 2 levels "no","yes": 1 1 1 1 1 2 1 1 1 2 ...
##  $ garage    : num  1 0 0 0 0 0 2 0 0 1 ...
##  $ prefer    : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
```

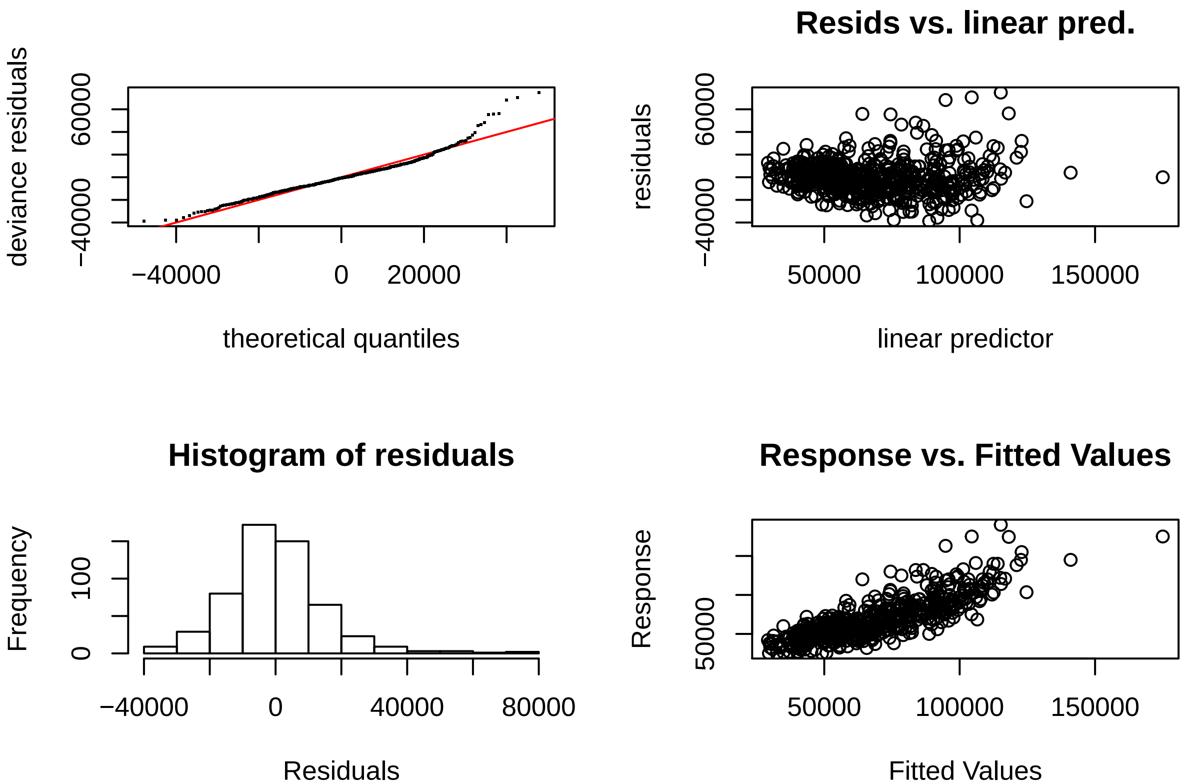
```
fitGaussAM <- mgcv::gam(price ~ s(lotsize, k=27) + bedrooms + factor(bathrooms) +
                          factor(stories) + factor(driveway) + factor(recreation)
                        + factor(fullbase) +
                          factor(gasheat) + factor(aircon) + garage + factor(prefer)
                        , data=HousePrices
                        , family=gaussian
                        )
```

Question 2

Chapter 3 Exercise 1B

Evaluate whether or not the residuals are consistent with the model assumptions.

```
mgcv::gam.check(fitGaussAM)
```




```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 5 iterations.
## The RMS GCV score gradient at convergence was 84.05184 .
## The Hessian was positive definite.
## Model rank =  41 / 41
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##                 k'   edf k-index p-value
## s(lotsize) 26.00  3.22    0.95    0.12
```

From the above output, it appears that we have evidence of skewness in our residual plots (qq plot shows skewness, as does the residual histogram). It also appears that we have some evidence of heteroscedasticity (our fitted vs. residual value plot shows an increase in residual variance as we increase the value of our predictor). Additionally, the p-value from our console output suggests that we should be using a larger number of basis functions for smoothed predictor, *lotsize*.

Question 3

Chapter 3 Exercise 1F

```
# generate data for the hypothetical house described by problem 1F
X <- data.frame(lotsize=5000
                , bedrooms=3
                , bathrooms=2
```

```r
                 , stories=2
                 , driveway="yes"
                 , recreation="no"
                 , fullbase="yes"
                 , gasheat="yes"
                 , aircon="no"
                 , garage=2
                 , prefer="no"
                 )

X_predict <- predict(fitGaussAM, newdata=X, se.fit=T)
print("Home price given by: ")
```

```
## [1] "Home price given by: "
```

```r
paste0("Ca$ ", round(X_predict$fit, 2))
```

```
## [1] "Ca$ 94511.07"
```

The predicted home price is given by $94,511.07.

Chapter 3 Exercise 1G

```r
# generate the bounds for a 95% confidence interval for the mean house price
lwr <- X_predict$fit - (1.96*X_predict$se.fit)
upr <- X_predict$fit + (1.96*X_predict$se.fit)
paste0("95% Confidence Interval given by: (", round(lwr, 2), " , ", round(upr, 2), ")")
```

```
## [1] "95% Confidence Interval given by: (87232.86 , 101789.29)"
```

Question 4

Run a cosso on the model in Exercise 1A, along with a stepwise regression. Compare.

```r
library("cosso")
```

I am first going to prep the data to get it into a form that the COSSO package appears to be looking for (I suspect that it may struggle with factor encodings).

```r
# select out non-factor variables
X1 <- as.matrix(HousePrices[, c(2, 3, 4, 5, 11)])
# one-hot encode the factors
X2 <- mod_mat <- stats::model.matrix(~ -1 + driveway + recreation + fullbase + gasheat +
                                 aircon + prefer
                               , data=HousePrices)
# complete design matrix
X <- data.frame(cbind(X1, X2))
# complete response vector
y <- HousePrices[, 1]
y_alt <- log(y)
```

```r
# attempt a cosso
cosso_mod <- cosso::cosso(x=X, y=y, family=c("Gaussian"))
```

```
## Error in solve.default(A + 1e-07 * diag(nrow(A)), b): Lapack routine dgesv: system is exactly singula
```

```r
# follow-up on suspicion that lack of normality in response variable distribution may be causing some i
# conduct test of normality for response and log-transformed response.
print(shapiro.test(y))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  y
## W = 0.92206, p-value = 3.383e-16
```

```r
print(shapiro.test(y_alt))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  y_alt
## W = 0.99626, p-value = 0.2274
```

```r
# y_alt follows normal distribution - retry COSSO model
```

```r
cosso_mod2 <- cosso::cosso(x=X, y=y_alt, family=c("Gaussian"))
```

```
## Error in solve.default(A + 1e-07 * diag(nrow(A)), b): system is computationally singular: reciprocal
```

```r
# using the hint from the problem set, try converting the factors to integers using as.integer
X <- HousePrices[, -1]
X$driveway <- as.integer(X$driveway)
X$recreation <- as.integer(X$recreation)
X$fullbase <- as.integer(X$fullbase)
X$gasheat <- as.integer(X$gasheat)
X$aircon <- as.integer(X$aircon)
X$prefer <- as.integer(X$prefer)
str(X)
```

```
## 'data.frame':    546 obs. of  11 variables:
##  $ lotsize   : num  5850 4000 3060 6650 6360 4160 3880 4160 4800 5500 ...
##  $ bedrooms  : num  3 2 3 3 2 3 3 3 3 3 ...
##  $ bathrooms : num  1 1 1 1 1 1 2 1 1 2 ...
##  $ stories   : num  2 1 1 2 1 1 2 3 1 4 ...
##  $ driveway  : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ recreation: int  1 1 1 2 1 2 1 1 2 2 ...
##  $ fullbase  : int  2 1 1 1 1 2 2 1 2 1 ...
##  $ gasheat   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ aircon    : int  1 1 1 1 1 2 1 1 1 2 ...
##  $ garage    : num  1 0 0 0 0 0 2 0 0 1 ...
##  $ prefer    : int  1 1 1 1 1 1 1 1 1 1 ...
```

```r
# now we have our new design matrix - let's transform the response and proceed
cosso_mod3 <- cosso::cosso(x=X
                           , y=log(HousePrices[, 1])
                           , family=c("Gaussian")
                           )
```

```
## Error in solve.default(A + 1e-07 * diag(nrow(A)), b): system is computationally singular: reciprocal
```

The COSSO package does not seem to be able to solve the matrices we are giving it. Move on to the stepwise regression.

```r
detach("package:mgcv", unload=TRUE)
```

```
## Warning: 'mgcv' namespace cannot be unloaded:
##   namespace 'mgcv' is imported by 'car' so cannot be unloaded
```

```r
library("gam")
```

```r
# create the initial model object
house_init <- gam::gam(price ~ s(lotsize, 27) + bedrooms + factor(bathrooms)
                         + factor(stories) + factor(driveway) + factor(recreation)
                       + factor(fullbase)
                          + factor(gasheat) + factor(aircon) + garage + factor(prefer)
                        , data=HousePrices
                       , family=gaussian
                       )

# note that I am only allowing the continuous terms the possibility of
# entering the model as smooths
house_step <- gam::step.gam(house_init, scope=list(
                             "lotsize" = ~1 + lotsize + s(lotsize, 23)
                             , "bedrooms" = ~1 + bedrooms
                             , "bathrooms" = ~ 1 + bathrooms
                             , "stories" = ~ 1 + stories
                             , "driveway" =  ~ 1 + driveway
                             , "recreation" =  ~ 1 + recreation
                             , "fullbase" = ~ 1 + fullbase
                             , "gasheat" = ~ 1 + gasheat
                             , "aircon" = ~ 1 + aircon
                             , "garage" = ~ 1 + garage + s(garage, 23)
                             , "prefer" = ~ 1 + prefer
                            )
                            , trace=T
                            , direction="forward"
                              )
```

```
## Start:  price ~ s(lotsize, 27) + bedrooms + factor(bathrooms) + factor(stories) +      factor(drivewa
```

```r
print("Final model Variables are given by: ")
```

```
## [1] "Final model Variables are given by: "
```

```r
print(names(house_step$model)[-1])
```

```
##  [1] "s(lotsize, 27)"     "bedrooms"           "factor(bathrooms)"
##  [4] "factor(stories)"    "factor(driveway)"   "factor(recreation)"
##  [7] "factor(fullbase)"   "factor(gasheat)"    "factor(aircon)"
## [10] "garage"             "factor(prefer)"
```

From the stepwise output, it appears that we could not have improved the model by taking away any terms - we appear to have selected the correct model from the beginning. This is perhaps unsurprising - if we examine summary output for the initial model fit, we can see that all of the included terms are highly significant (see below).

```r
gam::summary.gam(house_init)
```

```
##
## Call: gam::gam(formula = price ~ s(lotsize, 27) + bedrooms + factor(bathrooms) +
```

```
##     factor(stories) + factor(driveway) + factor(recreation) +
##     factor(fullbase) + factor(gasheat) + factor(aircon) + garage +
##     factor(prefer), family = gaussian, data = HousePrices)
## Deviance Residuals:
##    Min     1Q Median     3Q    Max
## -39048  -8214  -1556   7715  67467
##
## (Dispersion Parameter for gaussian family taken to be 222856358)
##
##     Null Deviance: 388602785841 on 545 degrees of freedom
## Residual Deviance: 112319290958 on 503.9986 degrees of freedom
## AIC: 12087.01
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##                     Df     Sum Sq    Mean Sq  F value    Pr(>F)
## s(lotsize, 27)       1 1.1156e+11 1.1156e+11 500.5867 < 2.2e-16 ***
## bedrooms             1 3.1687e+10 3.1687e+10 142.1836 < 2.2e-16 ***
## factor(bathrooms)    3 4.1081e+10 1.3694e+10  61.4469 < 2.2e-16 ***
## factor(stories)      3 1.7319e+10 5.7729e+09  25.9043 1.343e-15 ***
## factor(driveway)     1 5.4481e+09 5.4481e+09  24.4469 1.042e-06 ***
## factor(recreation)   1 5.7514e+09 5.7514e+09  25.8076 5.321e-07 ***
## factor(fullbase)     1 7.5499e+09 7.5499e+09  33.8778 1.045e-08 ***
## factor(gasheat)      1 1.6098e+09 1.6098e+09   7.2237  0.007433 **
## factor(aircon)       1 1.5496e+10 1.5496e+10  69.5314 7.224e-16 ***
## garage               1 5.9405e+09 5.9405e+09  26.6562 3.505e-07 ***
## factor(prefer)       1 9.2782e+09 9.2782e+09  41.6331 2.588e-10 ***
## Residuals          504 1.1232e+11 2.2286e+08
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                    Npar Df Npar F     Pr(F)
## (Intercept)
## s(lotsize, 27)          26 2.3903 0.0001687 ***
## bedrooms
## factor(bathrooms)
## factor(stories)
## factor(driveway)
## factor(recreation)
## factor(fullbase)
## factor(gasheat)
## factor(aircon)
## garage
## factor(prefer)
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Question 5

Chapter 3 Exercise 4

4A

```
library("aplore3")
data(icu)
```

```
# take a look
str(icu)
```

```
## 'data.frame':    200 obs. of  21 variables:
##  $ id    : int  4 8 12 14 27 28 32 38 40 41 ...
##  $ sta   : Factor w/ 2 levels "Lived","Died": 2 1 1 1 2 1 1 1 1 1 ...
##  $ age   : int  87 27 59 77 76 54 87 69 63 30 ...
##  $ gender: Factor w/ 2 levels "Male","Female": 2 2 1 1 2 1 2 1 1 2 ...
##  $ race  : Factor w/ 3 levels "White","Black",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ ser   : Factor w/ 2 levels "Medical","Surgical": 2 1 1 2 2 1 2 1 2 1 ...
##  $ can   : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ crn   : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ inf   : Factor w/ 2 levels "No","Yes": 2 2 1 1 2 2 2 2 1 1 ...
##  $ cpr   : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ sys   : int  80 142 112 100 128 142 110 110 104 144 ...
##  $ hra   : int  96 88 80 70 90 103 154 132 66 110 ...
##  $ pre   : Factor w/ 2 levels "No","Yes": 1 1 2 1 2 1 2 1 1 1 ...
##  $ type  : Factor w/ 2 levels "Elective","Emergency": 2 2 2 1 2 2 2 2 1 2 ...
##  $ fra   : Factor w/ 2 levels "No","Yes": 2 1 1 1 1 2 1 1 1 1 ...
##  $ po2   : Factor w/ 2 levels "> 60","<= 60": 2 1 1 1 1 1 1 2 1 1 ...
##  $ ph    : Factor w/ 2 levels ">= 7.25","< 7.25": 2 1 1 1 1 1 1 1 1 1 ...
##  $ pco   : Factor w/ 2 levels "<= 45","> 45": 2 1 1 1 1 1 1 1 1 1 ...
##  $ bic   : Factor w/ 2 levels ">= 18","< 18": 1 1 1 1 1 1 1 2 1 1 ...
##  $ cre   : Factor w/ 2 levels "<= 2.0","> 2.0": 1 1 1 1 1 1 1 1 1 1 ...
##  $ loc   : Factor w/ 3 levels "Nothing","Stupor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

4B

Select a logistic gam with the response variable being the indicator of a patient dying.

```
# just to make sure interpretation doesn't get messed up, recode the live variable
icu$live <- ifelse(icu$sta=="Lived", 1, 0)
table(icu$live, icu$sta) # quick check
```

```
##
##     Lived Died
##   0     0   40
##   1   160    0
```

```
# initial model object
icu_init <- gam::gam(live ~ age + gender + race + ser
                       + can + crn + inf
                         + cpr + sys + hra + pre
                         + type + fra + po2
                         + pco + bic + cre +
```

```
                    loc
  , data=icu
  , family=binomial(link="logit")
)
# stepwise regression setup
icu_step <- gam::step.gam(icu_init, scope=
                    list("age" = ~ 1 + age + s(age, 2)
                        , "gender" = ~ 1 + gender
                        , "ser" = ~ 1 + ser
                        , "can" = ~ 1 + can
                        , "crn" = ~ 1 + crn
                        , "inf" = ~ 1 + inf
                        , "cpr" = ~ 1 + cpr
                        , "sys" = ~ 1 + sys + s(sys, 2)
                        , "hra" = ~ 1 + hra + s(hra, 2)
                        , "pre" = ~ 1 + pre
                        , "type" = ~ 1 + type
                        , "fra" = ~ 1 + fra
                        , "po2" = ~ 1 + po2
                        , "ph" = ~ 1 + ph
                        , "pco" = ~ 1 + pco
                        , "bic" = ~ 1 + bic
                        , "cre" = ~ 1 + cre
                        , "loc" = ~ 1 + loc
                        )
                    )
```

```
## Start:  live ~ age + gender + race + ser + can + crn + inf + cpr + sys +      hra + pre + type + fra
## Step:1 live ~ race + age + gender + ser + can + crn + cpr + sys + hra +      pre + type + fra + po2 +
## Step:2 live ~ race + age + gender + ser + can + crn + cpr + sys + hra +      pre + type + fra + po2 +
## Step:3 live ~ race + age + gender + ser + can + crn + cpr + sys + pre +      type + fra + po2 + pco +
## Step:4 live ~ race + age + gender + ser + can + crn + cpr + sys + pre +      type + fra + po2 + pco +
## Step:5 live ~ race + age + gender + ser + can + crn + cpr + sys + pre +      type + fra + pco + loc
## Step:6 live ~ race + age + gender + ser + can + cpr + sys + pre + type +      fra + pco + loc ; AIC=
## Step:7 live ~ race + age + gender + can + cpr + sys + pre + type + fra +      pco + loc ; AIC= 144.70
## Step:8 live ~ race + age + gender + can + cpr + sys + pre + type + fra +      ph + pco + loc ; AIC= 1
## Step:9 live ~ race + age + gender + can + cpr + sys + pre + type + ph +      pco + loc ; AIC= 143.25
## Step:10 live ~ race + s(age, 2) + gender + can + cpr + sys + pre + type +      ph + pco + loc ; AIC=
```

The step trace was printed above, but let's take a look to see what the final model is:

```
print("ICU data set stepwise model given by: ")
```

```
## [1] "ICU data set stepwise model given by: "
```

```
(step_vars <- names(icu_step$model)[-1])
```

```
##  [1] "race"      "s(age, 2)" "gender"    "can"       "cpr"
##  [6] "sys"       "pre"       "type"      "ph"        "pco"
## [11] "loc"
```

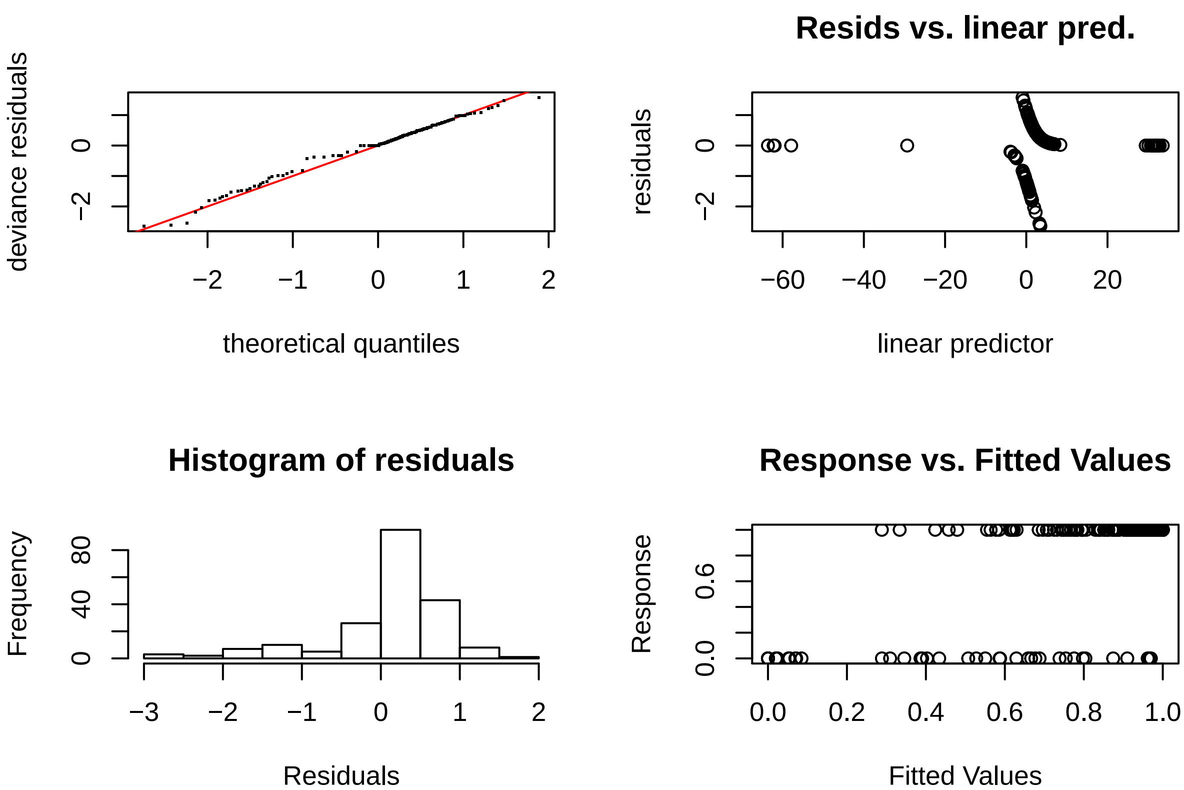
4C - Use mgcv::gam to re-fit the model selected in part B with GCV used for selction of the smoothing paramters. Include numerical and graphical summaries.

```
library("mgcv")
detach("package:gam", unload=TRUE)
refit_gcv <- mgcv::gam(live ~ race + s(age) + gender + can + cpr + sys + pre + type +
                         ph + pco + loc
          , data=icu
          , family=binomial(link="logit")
          , method="GCV.Cp"
          )
```

```
mgcv::summary.gam(refit_gcv)
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## live ~ race + s(age) + gender + can + cpr + sys + pre + type +
##     ph + pco + loc
##
## Parametric coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    3.059e+00  1.559e+00   1.961  0.04984 *
## raceBlack      2.904e+01  1.107e+06   0.000  0.99998
## raceOther     -2.365e-01  1.194e+00  -0.198  0.84306
## genderFemale   7.053e-01  5.260e-01   1.341  0.17993
## canYes        -3.203e+00  1.046e+00  -3.061  0.00221 **
## cprYes        -1.224e+00  8.712e-01  -1.406  0.15987
## sys            1.957e-02  8.370e-03   2.338  0.01940 *
## preYes        -1.177e+00  6.610e-01  -1.781  0.07490 .
## typeEmergency -3.883e+00  1.285e+00  -3.021  0.00252 **
## ph< 7.25      -1.732e+00  9.707e-01  -1.785  0.07433 .
## pco> 45        2.036e+00  9.873e-01   2.062  0.03923 *
## locStupor     -6.376e+01  1.796e+06   0.000  0.99997
## locComa       -3.178e+00  1.130e+00  -2.813  0.00491 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##          edf Ref.df Chi.sq p-value
## s(age) 1.586  1.967  9.997 0.00502 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.419   Deviance explained = 43.3%
## UBRE = -0.28677  Scale est. = 1         n = 200
```

```
mgcv::gam.check(refit_gcv)
```




```
##
## Method: UBRE   Optimizer: outer newton
## full convergence after 4 iterations.
## Gradient range [-9.377449e-08,-9.377449e-08]
## (score -0.2867667 & scale 1).
## Hessian positive definite, eigenvalue range [0.002132358,0.002132358].
## Model rank =  22 / 22
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##          k'  edf k-index p-value
## s(age) 9.00 1.59    1.08    0.86
```

The model does not appear to be a great fit to the data - the residual histogram suffers from dramatic left-skewness. The good news, however, is that we have used an adequate number of basis functions in the fitting of our smoothed parameter (see console p-value above).

4D

Use the refitted GAM model to score a new record with the described characteristics.

```
new_rec <- data.frame(
    age=79
    , gender="Female"
    , race="White"
    , ser="Medical"
    , can="No"
    , crn="No"
```

```r
    , inf="No"
    , cpr="No"
    , sys=228
    , hra=94
    , pre="No"
    , type="Emergency"
    , fra="No"
    , po2="<= 60"
    , ph=">= 7.25"
    , pco="> 45"
    , bic="< 18"
    , cre="> 2.0"
    , loc="Coma"
)
# get prediction on the response scale, rather than logit.
new_rec_pred <- mgcv::predict.gam(refit_gcv, newdata=new_rec, type="response")
print("Case survival probability given by:")
```

```
## [1] "Case survival probability given by:"
```

```r
print(new_rec_pred)
```

```
##         1
## 0.8965365
```

Question 6

Compare the stepwise and cosso models for Exercise 4 in Chapter 3

```r
# first, fit a cosso model
icu_cosso1 <- cosso::cosso(x=icu[3:21]
                            , y=icu$live
                            , family=c("Binomial")
                           )
```

```
## Warning: from glmnet Fortran code (error code -1); Convergence for 1th
## lambda value not reached after maxit=100000 iterations; solutions for
## larger lambdas returned

## Warning in getcoef(fit, nvars, nx, vnames): an empty model has been
## returned; probably a convergence issue

## Error in predmat[which, seq(nlami)] = preds: replacement has length zero
```

```r
# that didn't work - try out an alternative design matrix.
# one-hot encode the factors
X1 <- stats::model.matrix(~ -1 + gender + race + ser + can + crn + inf + cpr + pre + type + fra + po2 +
X2 <- icu %>%
  dplyr::select(age, sys, hra)
y <- icu$live
X <- data.frame(cbind(X1, X2))
str(X)
```

```
## 'data.frame':    200 obs. of  22 variables:
##  $ genderMale   : num  0 0 1 1 0 1 0 1 1 0 ...
##  $ genderFemale : num  1 1 0 0 1 0 1 0 0 1 ...
```

```
##  $ raceBlack    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ raceOther    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ serSurgical  : num  1 0 0 1 1 0 1 0 1 0 ...
##  $ canYes       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ crnYes       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ infYes       : num  1 1 0 0 1 1 1 1 0 0 ...
##  $ cprYes       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ preYes       : num  0 0 1 0 1 0 1 0 0 0 ...
##  $ typeEmergency: num  1 1 1 0 1 1 1 1 0 1 ...
##  $ fraYes       : num  1 0 0 0 0 1 0 0 0 0 ...
##  $ po2...60     : num  1 0 0 0 0 0 0 1 0 0 ...
##  $ ph..7.25     : num  1 0 0 0 0 0 0 0 0 0 ...
##  $ pco..45      : num  1 0 0 0 0 0 0 0 0 0 ...
##  $ bic..18      : num  0 0 0 0 0 0 0 1 0 0 ...
##  $ cre..2.0     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ locStupor    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ locComa      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ age          : int  87 27 59 77 76 54 87 69 63 30 ...
##  $ sys          : int  80 142 112 100 128 142 110 110 104 144 ...
##  $ hra          : int  96 88 80 70 90 103 154 132 66 110 ...
```

```
icu_cosso2 <- cosso::cosso(x=X, y=y, family=c("Binomial"))
```

```
## Warning: from glmnet Fortran code (error code -1); Convergence for 1th
## lambda value not reached after maxit=100000 iterations; solutions for
## larger lambdas returned

## Warning in getcoef(fit, nvars, nx, vnames): an empty model has been
## returned; probably a convergence issue

## Error in predmat[which, seq(nlami)] <- preds: replacement has length zero
```

It does not appear that the cosso regularization implementation is appropriate for the design matrix we are using. I have no model to compare to the stepwise selection model from Exercise 4 in Chapter 3.

Question 7

Rerun the model from Chapter 3, Exercise 4 using mgcv's regularization implementation.

```
icu_mgcv <- mgcv::gam(live ~ s(age) + gender + race + ser
                       + can + crn + inf
                         + cpr + s(sys) + s(hra) + pre
                         + type + fra + po2
                         + pco + bic + cre +
                         loc
  , data=icu
  , family=binomial(link="logit")
  , select=TRUE
)
(icu_mgcv_smry <- mgcv::summary.gam(icu_mgcv))
```

```
##
## Family: binomial
## Link function: logit
##
```

```
## Formula:
## live ~ s(age) + gender + race + ser + can + crn + inf + cpr +
##     s(sys) + s(hra) + pre + type + fra + po2 + pco + bic + cre +
##     loc
##
## Parametric coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    5.109e+00  1.469e+00   3.478 0.000505 ***
## genderFemale   5.299e-01  5.457e-01   0.971 0.331533
## raceBlack      5.114e+01  1.744e+07   0.000 0.999998
## raceOther     -2.570e-02  1.291e+00  -0.020 0.984110
## serSurgical    8.862e-01  6.483e-01   1.367 0.171628
## canYes        -3.533e+00  1.132e+00  -3.122 0.001797 **
## crnYes        -3.864e-01  8.554e-01  -0.452 0.651496
## infYes         5.461e-02  5.571e-01   0.098 0.921906
## cprYes        -1.020e+00  1.041e+00  -0.980 0.327115
## preYes        -1.434e+00  6.995e-01  -2.050 0.040379 *
## typeEmergency -3.520e+00  1.359e+00  -2.590 0.009604 **
## fraYes        -1.416e+00  1.032e+00  -1.372 0.170007
## po2<= 60       4.594e-01  9.506e-01   0.483 0.628895
## pco> 45        1.389e+00  9.736e-01   1.426 0.153752
## bic< 18       -6.120e-02  8.523e-01  -0.072 0.942759
## cre> 2.0      -3.674e-01  1.216e+00  -0.302 0.762555
## locStupor     -9.541e+01  3.021e+07   0.000 0.999997
## locComa       -3.303e+00  1.266e+00  -2.609 0.009071 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##              edf Ref.df Chi.sq p-value
## s(age) 1.661e+00      9  9.791 0.00155 **
## s(sys) 3.800e+00      9  7.383 0.08310 .
## s(hra) 5.162e-05      9  0.000 0.66328
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.416   Deviance explained = 46.9%
## UBRE = -0.23354  Scale est. = 1         n = 200
```

Final model using this selection method is given by the following formula:

```
icu_mgcv_smry$p.coeff
```

```
##   (Intercept)  genderFemale     raceBlack     raceOther   serSurgical
##    5.10934386    0.52987540   51.13740379   -0.02570271    0.88617293
##        canYes        crnYes        infYes        cprYes        preYes
##   -3.53320795   -0.38637136    0.05461166   -1.02002560   -1.43383661
## typeEmergency        fraYes      po2<= 60       pco> 45       bic< 18
##   -3.51965012   -1.41606322    0.45942109    1.38870920   -0.06119857
##      cre> 2.0     locStupor       locComa
##   -0.36739812  -95.40687476   -3.30287387
```

Question 8

7A

```r
library("kernlab")
data(spam)
print(names(spam))
```

```
##  [1] "make"              "address"           "all"
##  [4] "num3d"             "our"               "over"
##  [7] "remove"            "internet"          "order"
## [10] "mail"              "receive"           "will"
## [13] "people"            "report"            "addresses"
## [16] "free"              "business"          "email"
## [19] "you"               "credit"            "your"
## [22] "font"              "num000"            "money"
## [25] "hp"                "hpl"               "george"
## [28] "num650"            "lab"               "labs"
## [31] "telnet"            "num857"            "data"
## [34] "num415"            "num85"             "technology"
## [37] "num1999"           "parts"             "pm"
## [40] "direct"            "cs"                "meeting"
## [43] "original"          "project"           "re"
## [46] "edu"               "table"             "conference"
## [49] "charSemicolon"     "charRoundbracket"  "charSquarebracket"
## [52] "charExclamation"   "charDollar"        "charHash"
## [55] "capitalAve"        "capitalLong"       "capitalTotal"
## [58] "type"
```

7B

Generate our testing and training data sets through randomization

```r
set.seed(1)
nTest <- 1000
indsTest <- base::sample(1:nrow(spam), nTest, replace=FALSE)
indsTrain <- base::setdiff(1:nrow(spam), indsTest)
spamTest <- spam[indsTest, ]
spamTrain <- spam[indsTrain, ]
```

7C

Fit a logistic GLM including all possible predictors.

```r
fitTrainFullGLM <- glm(type ~ .
                       , family=binomial(link="logit")
                       , data=spamTrain
                       )
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
print(summary(fitTrainFullGLM))
```

```
##
## Call:
## glm(formula = type ~ ., family = binomial(link = "logit"), data = spamTrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.1553  -0.2110   0.0000   0.1198   5.3014
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -1.517e+00  1.585e-01  -9.571  < 2e-16 ***
## make              -4.409e-01  2.526e-01  -1.745 0.080919 .
## address           -1.570e-01  8.387e-02  -1.872 0.061152 .
## all                1.181e-01  1.193e-01   0.991 0.321862
## num3d              2.032e+00  1.615e+00   1.259 0.208171
## our                5.592e-01  1.167e-01   4.793 1.64e-06 ***
## over               8.656e-01  2.748e-01   3.149 0.001636 **
## remove             2.362e+00  3.710e-01   6.367 1.92e-10 ***
## internet           5.346e-01  1.934e-01   2.764 0.005703 **
## order              4.622e-01  3.082e-01   1.500 0.133615
## mail               5.913e-02  7.357e-02   0.804 0.421577
## receive           -1.909e-01  3.184e-01  -0.600 0.548687
## will              -1.230e-01  8.344e-02  -1.475 0.140272
## people            -1.943e-01  2.557e-01  -0.760 0.447484
## report             1.382e-01  1.395e-01   0.990 0.322001
## addresses          1.585e+00  9.064e-01   1.748 0.080381 .
## free               9.455e-01  1.548e-01   6.108 1.01e-09 ***
## business           8.731e-01  2.478e-01   3.524 0.000425 ***
## email              1.196e-01  1.344e-01   0.890 0.373724
## you                9.022e-02  4.068e-02   2.218 0.026563 *
## credit             8.568e-01  5.112e-01   1.676 0.093758 .
## your               2.141e-01  5.792e-02   3.697 0.000219 ***
## font               2.467e-01  1.982e-01   1.244 0.213341
## num000             2.036e+00  4.671e-01   4.358 1.31e-05 ***
## money              6.670e-01  2.673e-01   2.496 0.012570 *
## hp                -1.830e+00  3.390e-01  -5.397 6.76e-08 ***
## hpl               -1.100e+00  5.005e-01  -2.199 0.027903 *
## george            -1.118e+01  2.298e+00  -4.866 1.14e-06 ***
## num650             7.530e-01  2.999e-01   2.511 0.012049 *
## lab               -2.529e+00  1.769e+00  -1.430 0.152854
## labs              -2.853e-01  3.261e-01  -0.875 0.381765
## telnet            -1.542e-01  4.297e-01  -0.359 0.719664
## num857             1.315e+00  3.990e+00   0.330 0.741647
## data              -9.200e-01  3.757e-01  -2.449 0.014328 *
## num415            -1.231e+01  4.086e+00  -3.014 0.002582 **
## num85             -2.032e+00  8.319e-01  -2.442 0.014602 *
## technology         7.600e-01  3.567e-01   2.131 0.033100 *
## num1999           -8.878e-02  2.059e-01  -0.431 0.666359
## parts             -6.051e-01  4.881e-01  -1.240 0.215156
## pm                -6.065e-01  4.436e-01  -1.367 0.171540
## direct            -3.423e-01  3.974e-01  -0.861 0.389090
## cs                -4.961e+01  2.481e+01  -1.999 0.045567 *
## meeting           -3.744e+00  1.428e+00  -2.621 0.008755 **
## original          -8.478e-01  7.574e-01  -1.119 0.262971
```

```
## project           -1.353e+00  5.337e-01  -2.535 0.011231 *
## re                -8.749e-01  1.770e-01  -4.944 7.65e-07 ***
## edu               -1.277e+00  2.986e-01  -4.277 1.89e-05 ***
## table             -1.859e+00  1.539e+00  -1.208 0.226970
## conference        -6.055e+00  2.720e+00  -2.226 0.026034 *
## charSemicolon     -1.543e+00  5.511e-01  -2.799 0.005120 **
## charRoundbracket  -5.443e-01  3.756e-01  -1.449 0.147217
## charSquarebracket -3.155e-01  7.198e-01  -0.438 0.661189
## charExclamation    2.670e-01  6.796e-02   3.929 8.52e-05 ***
## charDollar         5.633e+00  8.040e-01   7.005 2.47e-12 ***
## charHash           2.770e+00  1.181e+00   2.345 0.019019 *
## capitalAve         1.797e-02  2.041e-02   0.881 0.378538
## capitalLong        6.638e-03  2.686e-03   2.471 0.013479 *
## capitalTotal       1.356e-03  2.821e-04   4.808 1.52e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4823.9  on 3600  degrees of freedom
## Residual deviance: 1435.1  on 3543  degrees of freedom
## AIC: 1551.1
##
## Number of Fisher Scoring iterations: 13
```

7D

Use a model selection strategy to select a subset of predictors. We have quite a few predictors - let's use a convenience function to generate potential predictors in our model.

```r
spam_scope <- gam::gam.scope(frame=spamTrain, response=58)
```

```r
head(spamTrain[, 1:57])
```

```
##   make address  all num3d  our over remove internet order mail receive
## 1 0.00    0.64 0.64     0 0.32 0.00   0.00     0.00  0.00 0.00    0.00
## 2 0.21    0.28 0.50     0 0.14 0.28   0.21     0.07  0.00 0.94    0.21
## 3 0.06    0.00 0.71     0 1.23 0.19   0.19     0.12  0.64 0.25    0.38
## 4 0.00    0.00 0.00     0 0.63 0.00   0.31     0.63  0.31 0.63    0.31
## 6 0.00    0.00 0.00     0 1.85 0.00   0.00     1.85  0.00 0.00    0.00
## 7 0.00    0.00 0.00     0 1.92 0.00   0.00     0.00  0.00 0.64    0.96
##   will people report addresses free business email  you credit your font
## 1 0.64   0.00   0.00      0.00 0.32     0.00  1.29 1.93   0.00 0.96    0
## 2 0.79   0.65   0.21      0.14 0.14     0.07  0.28 3.47   0.00 1.59    0
## 3 0.45   0.12   0.00      1.75 0.06     0.06  1.03 1.36   0.32 0.51    0
## 4 0.31   0.31   0.00      0.00 0.31     0.00  0.00 3.18   0.00 0.31    0
## 6 0.00   0.00   0.00      0.00 0.00     0.00  0.00 0.00   0.00 0.00    0
## 7 1.28   0.00   0.00      0.00 0.96     0.00  0.32 3.85   0.00 0.64    0
##   num000 money hp hpl george num650 lab labs telnet num857 data num415
## 1   0.00  0.00  0   0      0      0   0    0      0      0    0      0
## 2   0.43  0.43  0   0      0      0   0    0      0      0    0      0
## 3   1.16  0.06  0   0      0      0   0    0      0      0    0      0
## 4   0.00  0.00  0   0      0      0   0    0      0      0    0      0
## 6   0.00  0.00  0   0      0      0   0    0      0      0    0      0
```

```
## 7   0.00  0.00  0   0      0      0   0    0      0      0    0      0
##   num85 technology num1999 parts pm direct cs meeting original project
## 1     0          0    0.00     0  0   0.00  0       0     0.00       0
## 2     0          0    0.07     0  0   0.00  0       0     0.00       0
## 3     0          0    0.00     0  0   0.06  0       0     0.12       0
## 4     0          0    0.00     0  0   0.00  0       0     0.00       0
## 6     0          0    0.00     0  0   0.00  0       0     0.00       0
## 7     0          0    0.00     0  0   0.00  0       0     0.00       0
##     re  edu table conference charSemicolon charRoundbracket
## 1 0.00 0.00     0          0          0.00            0.000
## 2 0.00 0.00     0          0          0.00            0.132
## 3 0.06 0.06     0          0          0.01            0.143
## 4 0.00 0.00     0          0          0.00            0.137
## 6 0.00 0.00     0          0          0.00            0.223
## 7 0.00 0.00     0          0          0.00            0.054
##   charSquarebracket charExclamation charDollar charHash capitalAve
## 1                 0           0.778      0.000    0.000      3.756
## 2                 0           0.372      0.180    0.048      5.114
## 3                 0           0.276      0.184    0.010      9.821
## 4                 0           0.137      0.000    0.000      3.537
## 6                 0           0.000      0.000    0.000      3.000
## 7                 0           0.164      0.054    0.000      1.671
##   capitalLong capitalTotal
## 1          61          278
## 2         101         1028
## 3         485         2259
## 4          40          191
## 6          15           54
## 7           4          112
```

```r
head(spamTrain[, 58])
```

```
## [1] spam spam spam spam spam spam
## Levels: nonspam spam
```

```r
detach("package:mgcv", unload=TRUE)
```

```
## Warning: 'mgcv' namespace cannot be unloaded:
##   namespace 'mgcv' is imported by 'car' so cannot be unloaded
```

Run the stepwise regression

```r
#spam_init <- gam::gam(type ~ .
#                       , family=binomial(link="logit")
#                       , data=spamTrain
#                       )
#
#spam_step <- gam::step.gam(object=spam_init
#                           , scope=spam_scope
#                           )
#
```

That was taking way too long, and we aren't being asked to include smoothed predictors at this point. Let's fit this with the LASSO penalization.

```r
library("glmnet")
```

```r
X=as.matrix(spamTrain[, 1:57])
y=ifelse(spamTrain[, 58]=="spam", 1, 0)
# select optimal constraint via cross validation
set.seed(1738)
spam_cv <- glmnet::cv.glmnet(x=X
                             , y=y
                             , family="binomial"
                             , alpha=1
                             , nfolds=3 # reduce runtime
                             )
```

```r
# grab our optimal lambda value
(lambda_min <- spam_cv$lambda.min)
```

```
## [1] 0.0003835095
```

```r
# optimal coefficients
(mod_coefs <- coef(spam_cv, s=lambda_min))
```

```
## 58 x 1 sparse Matrix of class "dgCMatrix"
##                             1
## (Intercept)       -1.412852180
## make              -0.400146137
## address           -0.155163456
## all                0.132129269
## num3d              0.402742843
## our                0.549067865
## over               0.771903110
## remove             2.420956962
## internet           0.538726021
## order              0.391826012
## mail               0.052764955
## receive           -0.135399792
## will              -0.116623372
## people            -0.171177788
## report             0.117303655
## addresses          1.216677947
## free               0.936332770
## business           0.767227183
## email              0.132079843
## you                0.088475474
## credit             0.772890313
## your               0.210064405
## font               0.279642205
## num000             2.066734798
## money              0.796224089
## hp                -1.650870873
## hpl               -0.973270171
## george            -3.010975301
## num650             0.577793458
## lab               -1.455605289
## labs              -0.258525089
## telnet            -0.168375380
## num857             .
## data              -0.949914194
```

```
## num415                .
## num85               -1.532587411
## technology           0.690977263
## num1999             -0.127143757
## parts               -0.525389613
## pm                  -0.477385056
## direct              -0.335102920
## cs                  -8.006965011
## meeting             -2.285386490
## original            -0.605485428
## project             -1.214453893
## re                  -0.820218860
## edu                 -1.171519056
## table               -1.724287215
## conference          -4.329302828
## charSemicolon       -1.388743623
## charRoundbracket    -0.463920403
## charSquarebracket   -0.327927801
## charExclamation      0.274358851
## charDollar           5.272469251
## charHash             2.029033613
## capitalAve          -0.002228969
## capitalLong          0.003883233
## capitalTotal         0.001035578
```

It looks like we need most of the variables, with the exception of a few. Let's select those out of our design matrix for the upcoming GLM.

```
reduced_X <- data.frame(X) %>%
  dplyr::select(-c(num857, num415, capitalAve))
```

Fit the logistic GLM based on this subset.

```
design_mat <- cbind(reduced_X, y)
new_glm <- stats::glm(y ~ .
                      , data=design_mat
                      , family=binomial(link="logit")
                      )
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(new_glm)
```

```
##
## Call:
## stats::glm(formula = y ~ ., family = binomial(link = "logit"),
##     data = design_mat)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.1905  -0.2135   0.0000   0.1221   5.0037
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -1.481e+00  1.559e-01  -9.498  < 2e-16 ***
## make               -4.528e-01  2.521e-01  -1.796 0.072464 .
## address            -1.596e-01  8.456e-02  -1.887 0.059116 .
```

```
## all                 1.138e-01  1.189e-01   0.957 0.338653
## num3d               2.046e+00  1.615e+00   1.267 0.205307
## our                 5.594e-01  1.162e-01   4.816 1.47e-06 ***
## over                8.576e-01  2.716e-01   3.158 0.001588 **
## remove              2.382e+00  3.711e-01   6.419 1.38e-10 ***
## internet            5.430e-01  1.934e-01   2.808 0.004988 **
## order               4.437e-01  2.993e-01   1.482 0.138255
## mail                6.223e-02  7.309e-02   0.851 0.394492
## receive            -1.886e-01  3.177e-01  -0.594 0.552706
## will               -1.218e-01  8.313e-02  -1.465 0.142947
## people             -1.991e-01  2.539e-01  -0.784 0.432926
## report              1.331e-01  1.391e-01   0.957 0.338627
## addresses           1.577e+00  9.034e-01   1.745 0.080942 .
## free                9.336e-01  1.534e-01   6.085 1.17e-09 ***
## business            8.392e-01  2.475e-01   3.391 0.000698 ***
## email               1.190e-01  1.333e-01   0.893 0.372040
## you                 8.685e-02  4.057e-02   2.141 0.032277 *
## credit              8.706e-01  5.104e-01   1.706 0.088037 .
## your                2.171e-01  5.766e-02   3.764 0.000167 ***
## font                2.608e-01  1.983e-01   1.315 0.188551
## num000              2.050e+00  4.681e-01   4.381 1.18e-05 ***
## money               6.787e-01  2.693e-01   2.521 0.011712 *
## hp                 -1.833e+00  3.419e-01  -5.361 8.26e-08 ***
## hpl                -1.103e+00  5.022e-01  -2.197 0.028055 *
## george             -9.968e+00  1.963e+00  -5.077 3.83e-07 ***
## num650              7.536e-01  3.000e-01   2.512 0.011988 *
## lab                -2.540e+00  1.773e+00  -1.433 0.151901
## labs               -2.765e-01  3.244e-01  -0.852 0.393987
## telnet             -1.711e-01  4.716e-01  -0.363 0.716814
## data               -9.002e-01  3.698e-01  -2.435 0.014911 *
## num85              -2.030e+00  8.056e-01  -2.520 0.011742 *
## technology          7.764e-01  3.557e-01   2.183 0.029035 *
## num1999            -8.945e-02  2.067e-01  -0.433 0.665229
## parts              -6.028e-01  4.901e-01  -1.230 0.218705
## pm                 -6.081e-01  4.420e-01  -1.376 0.168934
## direct             -3.339e-01  3.946e-01  -0.846 0.397478
## cs                 -4.828e+01  2.534e+01  -1.905 0.056798 .
## meeting            -3.621e+00  1.374e+00  -2.634 0.008427 **
## original           -8.519e-01  7.549e-01  -1.128 0.259111
## project            -1.359e+00  5.344e-01  -2.542 0.011012 *
## re                 -8.583e-01  1.747e-01  -4.912 9.00e-07 ***
## edu                -1.283e+00  2.999e-01  -4.280 1.87e-05 ***
## table              -1.849e+00  1.527e+00  -1.211 0.225878
## conference         -6.114e+00  2.755e+00  -2.220 0.026444 *
## charSemicolon      -1.541e+00  5.483e-01  -2.811 0.004946 **
## charRoundbracket   -5.471e-01  3.752e-01  -1.458 0.144805
## charSquarebracket -3.057e-01  7.073e-01  -0.432 0.665652
## charExclamation     2.638e-01  6.734e-02   3.917 8.95e-05 ***
## charDollar          5.419e+00  7.893e-01   6.866 6.60e-12 ***
## charHash            2.706e+00  1.184e+00   2.287 0.022209 *
## capitalLong         7.992e-03  1.975e-03   4.047 5.20e-05 ***
## capitalTotal        1.271e-03  2.695e-04   4.715 2.42e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4823.9  on 3600  degrees of freedom
## Residual deviance: 1441.7  on 3546  degrees of freedom
## AIC: 1551.7
##
## Number of Fisher Scoring iterations: 13
```

7E

Make predictions for the test data classify as spam if prediction probability exceeds 0.5.

```r
spamTest_preds <- predict.glm(object=new_glm, newdata=spamTest, type="response")
# spam=1 in train, 2 in test
pre_conf <- data.frame(cbind(spam_fl=spamTest$type
                             , spam_pred=spamTest_preds))

pre_conf$pred_decision <- ifelse(pre_conf$spam_pred>0.5, 2, 1)
# table for pre-confusion matrix
conf_table <- table(pre_conf$spam_fl, pre_conf$pred_decision)
# calculate the misclassification rate
(sum(conf_table) - (conf_table[1,1] + conf_table[2,2])) / sum(conf_table)
```

```
## [1] 0.081
```

7F

Fit the same model as in part E, but fit all terms with smoothing splines. Apply the same classification criterion.

```r
library("mgcv")
```

```r
refit_mod <- mgcv::gam(
                      type ~
                        s(make) +
s(address) +
s(all) +
s(num3d) +
s(our) +
s(over)
#s(remove) +
#s(internet) +
#s(order) +
#s(mail) +
#s(receive) +
#s(will) +
#s(people) +
#s(report) +
#s(addresses) +
#s(free) +
#s(business) +
#s(email) +
#s(you) +
```

```
#s(credit) +
#s(your) +
#s(font) +
#s(num000) +
#s(money) +
#s(hp) +
#s(hpl) +
#s(george) +
#s(num650) +
#s(lab) +
#s(labs) +
#s(telnet) +
#s(data) +
#s(num85) +
#s(technology) +
#s(num1999) +
#s(parts) +
#s(pm) +
#s(direct) +
#s(cs) +
#s(meeting) +
#s(original) +
#s(project) +
#s(re) +
#s(edu) +
#s(table) +
#s(conference) +
#s(charSemicolon) +
#s(charRoundbracket) +
#s(charSquarebracket) +
#s(charExclamation) +
#s(charDollar) +
#s(charHash) +
#s(capitalLong) +
#s(capitalTotal)
 , data=spamTrain
, family=binomial
)
```

```
refit_preds <- predict.gam(object=refit_mod, newdata=spamTest, type="response")
refit_pre_conf <- data.frame(cbind(spam_fl=spamTest$type
                        , spam_pred=refit_preds
                        ))
refit_pre_conf$pred_decision <- ifelse(refit_pre_conf$spam_pred > 0.5, 2, 1)
refit_conf_table <- table(refit_pre_conf$spam_fl, refit_pre_conf$pred_decision)
# capture our misclassification rate
(sum(refit_conf_table) - (refit_conf_table[1,1] + refit_conf_table[2,2])) / sum(refit_conf_table)
```

```
## [1] 0.247
```

7G

Based on the above, it appears that the non-smoothed GLM has superior classification accuracy.