Task 4

Task 4: Use 5-fold cross-validation to select the best λ. Compare the prediction performance between the "optimal" model and "full" model.

5-fold CV

We write an **R** function `cv.logit.lasso` to conduct 5-fold cross-validation to select the best λ.

```
cv.logit.lasso <- function(x, y, nfolds = 5, lambda) {
  auc <- data.frame(matrix(ncol = 3, nrow = 0))
  colnames(auc) <- c("lambda", "fold", "auc")
  folds <- createFolds(y, k = nfolds)
  for (i in 1:nfolds) {
    valid_index <- folds[[i]]
    x_training <- x[-valid_index, ]
    y_training <- y[-valid_index]
    training_dat <- data.frame(cbind(y_training, x_training))
    x_valid <- cbind(rep(1, length(valid_index)), x[valid_index, ])
    y_valid <- y[valid_index]
    res <- LogisticLASSO(dat = training_dat, start = rep(0, ncol(training_dat)), lambda = lambda)
    for (k in 1:nrow(res)) {
      betavec <- res[k, 2:ncol(res)]
      u_valid <- x_valid %*% betavec
      phat_valid <- sigmoid(u_valid)[, 1]
      roc <- roc(response = y_valid, predictor = phat_valid)
      auc <- rbind(auc, c(lambda[k], i, roc$auc[1]))
    }
  }
  colnames(auc) <- c("lambda", "fold", "auc")
  cv_res <- auc %>%
    group_by(lambda) %>%
    summarize(auc_mean = mean(auc)) %>%
    mutate(auc_ranking = min_rank(desc(auc_mean)))
  bestlambda <- min(cv_res$lambda[cv_res$auc_ranking == 1])
  return(cv_res)
}
```

Compare the results of cross-validation using `glmnet` and using our algorithm.

1. Our function `cv.logit.lasso`:

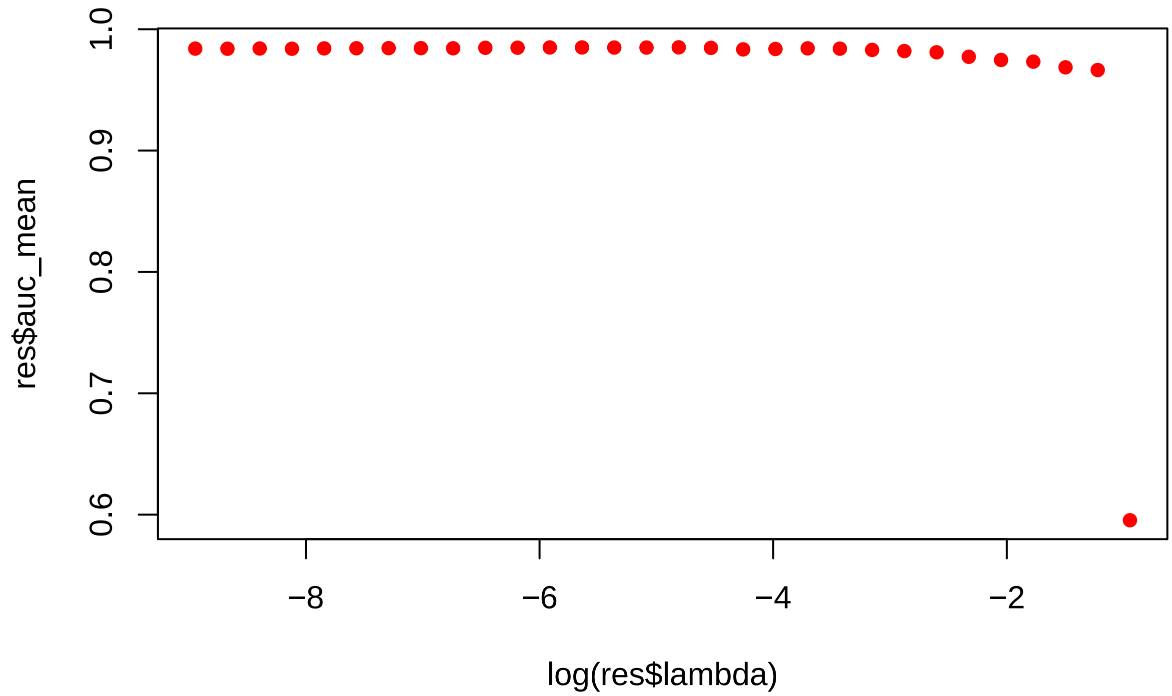
```
lambda_max <- max(t(x) %*% y) / length(y)
lambdas <- exp(seq(log(lambda_max), log(lambda_max) - 8, length = 30))
set.seed(1)
res = cv.logit.lasso(x, y, nfolds = 5, lambda = lambdas)
as.matrix(res %>% arrange(-lambda))
```

```
##              lambda  auc_mean auc_ranking
##  [1,] 0.3880307793 0.5953964          30
##  [2,] 0.2944833885 0.9663792          29
##  [3,] 0.2234886270 0.9686033          28
##  [4,] 0.1696094528 0.9732774          27
##  [5,] 0.1287195992 0.9746754          26
##  [6,] 0.0976875695 0.9772606          25
##  [7,] 0.0741368160 0.9809819          24
##  [8,] 0.0562637346 0.9819983          23
##  [9,] 0.0426995385 0.9829513          22
## [10,] 0.0324054314 0.9839803          17
## [11,] 0.0245930523 0.9842396          13
## [12,] 0.0186641003 0.9837128          20
## [13,] 0.0141645142 0.9833856          21
## [14,] 0.0107496992 0.9846370           8
## [15,] 0.0081581359 0.9850634           1
## [16,] 0.0061913529 0.9848939           4
## [17,] 0.0046987267 0.9848928           5
## [18,] 0.0035659464 0.9849928           3
## [19,] 0.0027062595 0.9849937           2
## [20,] 0.0020538280 0.9847940           6
## [21,] 0.0015586862 0.9846939           7
## [22,] 0.0011829144 0.9842929          12
## [23,] 0.0008977346 0.9844208           9
## [24,] 0.0006813066 0.9844208           9
## [25,] 0.0005170555 0.9843215          11
## [26,] 0.0003924025 0.9842212          14
## [27,] 0.0002978012 0.9839216          18
## [28,] 0.0002260066 0.9841236          15
## [29,] 0.0001715204 0.9839214          19
## [30,] 0.0001301698 0.9840223          16
```

```r
# best lambda
res$lambda[res$auc_ranking == 1]
```

```
## [1] 0.008158136
```

```r
plot(log(res$lambda), res$auc_mean, pch = 16, col = "red")
```

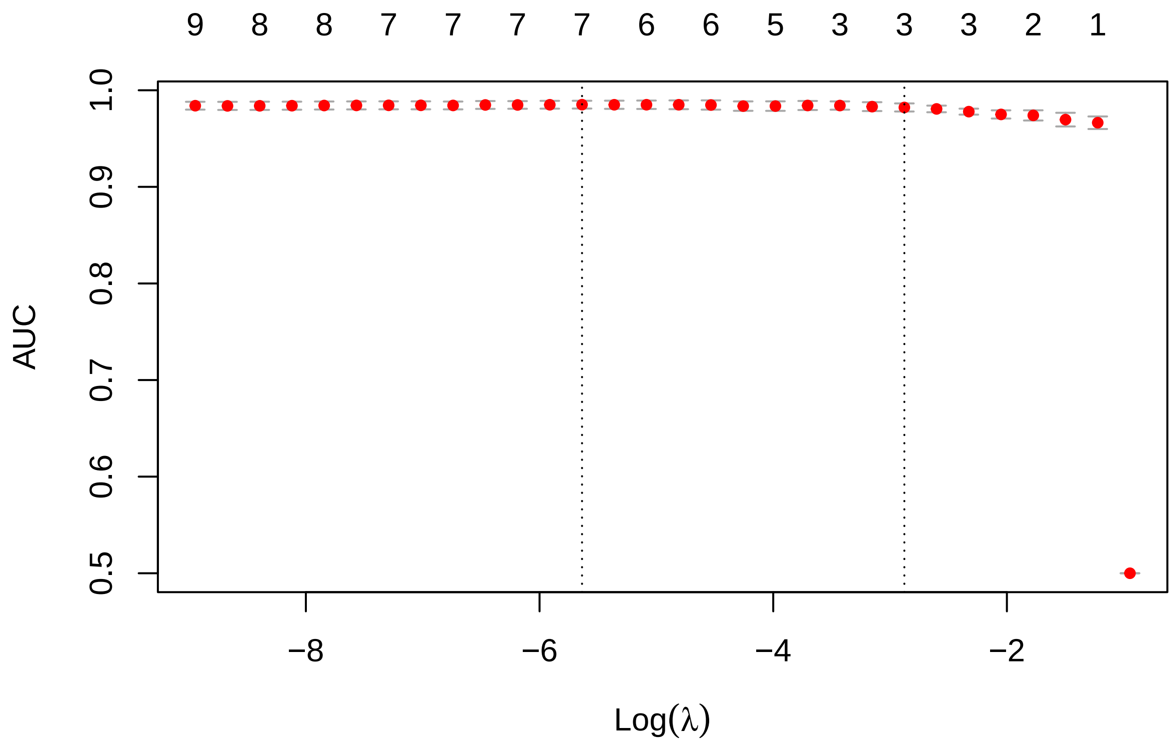


2. `glmnet` from **R** package `caret`

```
set.seed(1)
fit.logit.lasso = cv.glmnet(x, y,
                            nfolds = 5, alpha = 1,
                            lambda = lambdas,
                            family = "binomial", type.measure = "auc")
# best lambda
fit.logit.lasso$lambda.min
```

```
## [1] 0.003565946
```

```
plot(fit.logit.lasso)
```



The results are slightly different (mean AUC values).

```
tibble(
  lambda = lambdas,
  ours_AUC = res %>% arrange(-lambda) %>% .$auc_mean,
  cv.glmnet_AUC = fit.logit.lasso$cvm
) %>%
  knitr::kable()
```

lambda	ours_AUC	cv.glmnet_AUC
0.3880308	0.5953964	0.5000000
0.2944834	0.9663792	0.9664002
0.2234886	0.9686033	0.9696155
0.1696095	0.9732774	0.9739870
0.1287196	0.9746754	0.9749820
0.0976876	0.9772606	0.9778663
0.0741368	0.9809819	0.9806886
0.0562637	0.9819983	0.9822062
0.0426995	0.9829513	0.9830639
0.0324054	0.9839803	0.9841889
0.0245931	0.9842396	0.9842478
0.0186641	0.9837128	0.9836220
0.0141645	0.9833856	0.9835959
0.0107497	0.9846370	0.9847465
0.0081581	0.9850634	0.9849718

lambda	ours_AUC	cv.glmnet_AUC
0.0061914	0.9848939	0.9850007
0.0046987	0.9848928	0.9849974
0.0035659	0.9849928	0.9850940
0.0027063	0.9849937	0.9849969
0.0020538	0.9847940	0.9847955
0.0015587	0.9846939	0.9847967
0.0011829	0.9842929	0.9842911
0.0008977	0.9844208	0.9844187
0.0006813	0.9844208	0.9844187
0.0005171	0.9843215	0.9843196
0.0003924	0.9842212	0.9842184
0.0002978	0.9839216	0.9840182
0.0002260	0.9841236	0.9839183
0.0001715	0.9839214	0.9838160
0.0001302	0.9840223	0.9840178

Prediction performance comparison

NOT FINISHED!