Code for slide deck on penalised regression and cross-validation

Benjamin Skov Kaas-Hansen

11/02/2020

Contents

Setup

```r
packages <- c("plyr", "tidyr", "broom", "boot", "glmnet", "selectiveInference", "MASS", "tidyverse")
for (p in packages)
    library(p, character.only = TRUE)
knitr::opts_chunk$set(fig.align = "center")

theme_set(theme_minimal())

# Little helper to get the glmnet coefficients in nice tidy format
pretty_coefs <- function(coefs) { # coefs: the output from coef(fit_object, s = [value])
    enframe(coefs[, 1], "predictor", "coefficient") %>%
        filter(coefficient != 0) %>%
        arrange(desc(abs(coefficient)))
}
```

Cross-validation Pima

Homegrown

```r
# Very simplistic implementation
data(PimaIndiansDiabetes2, package = "mlbench")
glimpse(PimaIndiansDiabetes2)
```

```
## Rows: 768
## Columns: 9
## $ pregnant <dbl> 6, 1, 8, 1, 0, 5, 3, 10, 2, 8, 4, 10, 10, 1, 5, 7, 0, 7, 1...
## $ glucose  <dbl> 148, 85, 183, 89, 137, 116, 78, 115, 197, 125, 110, 168, 1...
## $ pressure <dbl> 72, 66, 64, 66, 40, 74, 50, NA, 70, 96, 92, 74, 80, 60, 72...
## $ triceps  <dbl> 35, 29, NA, 23, 35, NA, 32, NA, 45, NA, NA, NA, NA, 23, 19...
## $ insulin  <dbl> NA, NA, NA, 94, 168, NA, 88, NA, 543, NA, NA, NA, NA, 846,...
## $ mass     <dbl> 33.6, 26.6, 23.3, 28.1, 43.1, 25.6, 31.0, 35.3, 30.5, NA, ...
## $ pedigree <dbl> 0.627, 0.351, 0.672, 0.167, 2.288, 0.201, 0.248, 0.134, 0....
## $ age      <dbl> 50, 31, 32, 21, 33, 30, 26, 29, 53, 54, 30, 34, 57, 59, 51...
## $ diabetes <fct> pos, neg, pos, neg, pos, neg, pos, neg, pos, pos, neg, pos...
```

```
summary(PimaIndiansDiabetes2)
```

```
##     pregnant         glucose         pressure         triceps
##  Min.   : 0.000   Min.   : 44.0   Min.   : 24.00   Min.   : 7.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 64.00   1st Qu.:22.00
##  Median : 3.000   Median :117.0   Median : 72.00   Median :29.00
##  Mean   : 3.845   Mean   :121.7   Mean   : 72.41   Mean   :29.15
##  3rd Qu.: 6.000   3rd Qu.:141.0   3rd Qu.: 80.00   3rd Qu.:36.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##                   NA's   :5       NA's   :35       NA's   :227
##     insulin            mass          pedigree           age         diabetes
##  Min.   : 14.00   Min.   :18.20   Min.   :0.0780   Min.   :21.00   neg:500
##  1st Qu.: 76.25   1st Qu.:27.50   1st Qu.:0.2437   1st Qu.:24.00   pos:268
##  Median :125.00   Median :32.30   Median :0.3725   Median :29.00
##  Mean   :155.55   Mean   :32.46   Mean   :0.4719   Mean   :33.24
##  3rd Qu.:190.00   3rd Qu.:36.60   3rd Qu.:0.6262   3rd Qu.:41.00
##  Max.   :846.00   Max.   :67.10   Max.   :2.4200   Max.   :81.00
##  NA's   :374      NA's   :11
```

```
set.seed(42)
pima <- na.exclude(PimaIndiansDiabetes2) %>%
    mutate(cv_fold = sample(1:10, n(), replace = TRUE))
table(pima$cv_fold) # fairly equal distribution
```

```
##
##  1  2  3  4  5  6  7  8  9 10
## 44 40 29 39 39 47 29 26 46 53
```

```
err_cv <- c()
for (i in unique(pima$cv_fold)) {
    train <- filter(pima, cv_fold != i)
    mod <- glm(diabetes ~ age + mass + insulin + pregnant, data = train, family = binomial)

    val <- filter(pima, cv_fold == i)
    y_pred <- predict(mod, newdata = val, type = "response")
    y_true <- as.numeric(val$diabetes) - 1 # bring binary factor to 0/1 scale
    err <- mean(abs(y_true - y_pred) > 0.5)
    err_cv <- c(err_cv, err)
}
err_cv
```

```
##  [1] 0.2500000 0.2564103 0.2608696 0.2641509 0.3333333 0.2500000 0.3076923
##  [8] 0.3103448 0.3448276 0.2978723
```

```
mean(err_cv)
```

```
## [1] 0.2875501
```

Using packages

```
binary_pred_cost <- function(y_true, y_pred) {
    mean(abs(y_true - y_pred) > 0.5)
}

library(boot)
pima_glm <- glm(diabetes ~ age + mass + insulin + pregnant, data = pima, family = binomial)

pima_loo <- cv.glm(pima, pima_glm, cost = binary_pred_cost)
pima_loo$delta # 1st is raw estimate, 2nd is bias-corrected
```

```
## [1] 0.2857143 0.2844908
```

```
pima_cv1 <- cv.glm(pima, pima_glm, cost = binary_pred_cost, K = 10)
pima_cv1$delta # 1st is raw estimate, 2nd is bias-corrected
```

```
## [1] 0.2806122 0.2801112
```

```
# This model doesn't really overfit
```

The `modelr`-package has some powerful functionalities for CV, LOOCV and bootstrapping.

Lasso regression example: biopsies from breast cancer patients

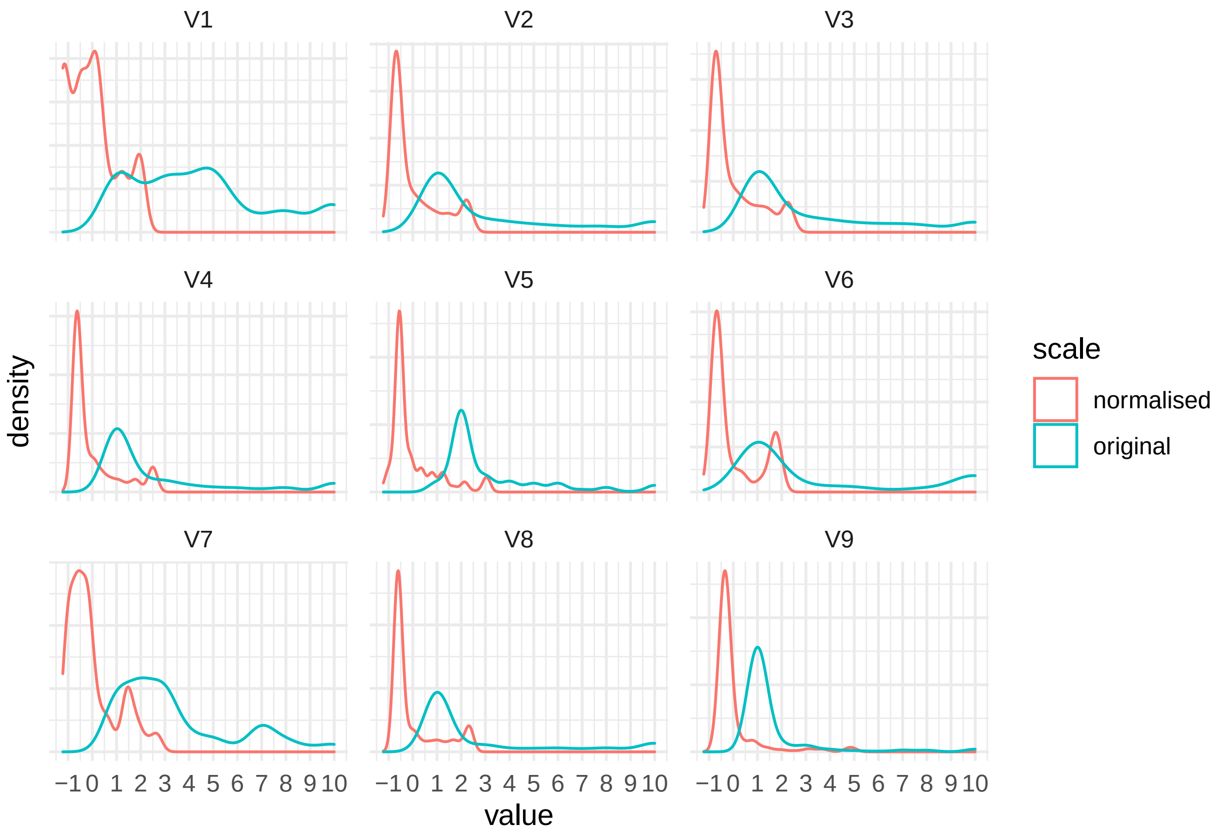
Lasso regression

Let's look at the data

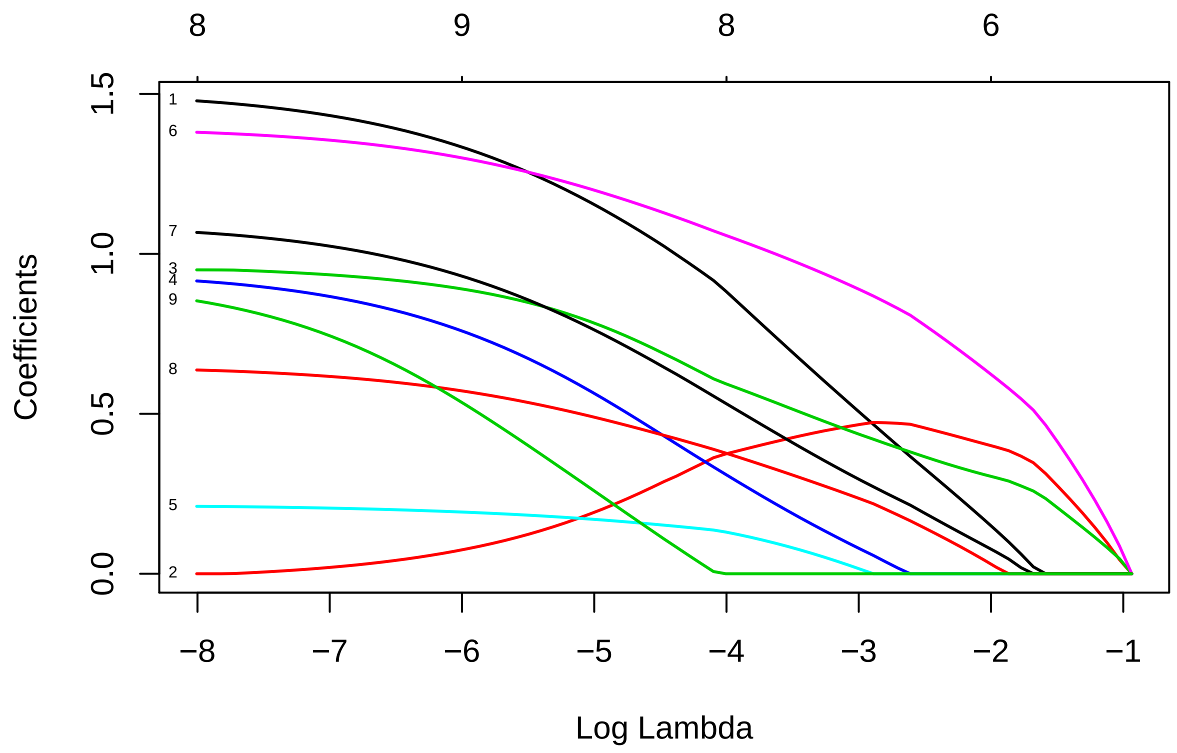
```
data(biopsy)
summary(biopsy) # NA's in V6; mean varies across variables but anyway somewhere around 2 and 4
```

```
##       ID                  V1               V2               V3        
##  Length:699         Min.   : 1.000   Min.   : 1.000   Min.   : 1.000  
##  Class :character   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000  
##  Mode  :character   Median : 4.000   Median : 1.000   Median : 1.000  
##                     Mean   : 4.418   Mean   : 3.134   Mean   : 3.207  
##                     3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000  
##                     Max.   :10.000   Max.   :10.000   Max.   :10.000  
##                                                                       
##        V4               V5               V6               V7        
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000  
##  1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 2.000  
##  Median : 1.000   Median : 2.000   Median : 1.000   Median : 3.000  
##  Mean   : 2.807   Mean   : 3.216   Mean   : 3.545   Mean   : 3.438  
##  3rd Qu.: 4.000   3rd Qu.: 4.000   3rd Qu.: 6.000   3rd Qu.: 5.000  
##  Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000  
##                                    NA's   :16                       
##        V8               V9               class    
##  Min.   : 1.000   Min.   : 1.000   benign   :458  
##  1st Qu.: 1.000   1st Qu.: 1.000   malignant:241  
##  Median : 1.000   Median : 1.000                  
##  Mean   : 2.867   Mean   : 1.589                  
##  3rd Qu.: 4.000   3rd Qu.: 1.000                  
##  Max.   :10.000   Max.   :10.000                  
## 
```

```
biopsy_complete <- na.exclude(biopsy) # remove rows with any missing value
biopsy_predictors <- select(biopsy_complete, -ID, -class) %>%
    scale() # note attributes "remember" normlisation factors; useful for transforming test set

bind_rows(gather(as_tibble(biopsy_predictors), var, value) %>%
            mutate(scale = "normalised"),
          gather(select(biopsy_complete, -ID, -class), var, value) %>%
            mutate(scale = "original")) %>%
    ggplot(aes(x = value, colour = scale)) +
        geom_density(position = "identity") +
        scale_x_continuous(breaks = -2:10) +
        facet_wrap(~ var, scales = "free_y") +
        theme(axis.text.y = element_blank())
```



```
lasso_logreg <- glmnet(biopsy_predictors, biopsy_complete$class, family = "binomial")

# Coefficient profile plot (built-in: ugly but easy)
plot(lasso_logreg, xvar = "lambda", label = TRUE, lwd = 1.5)
```



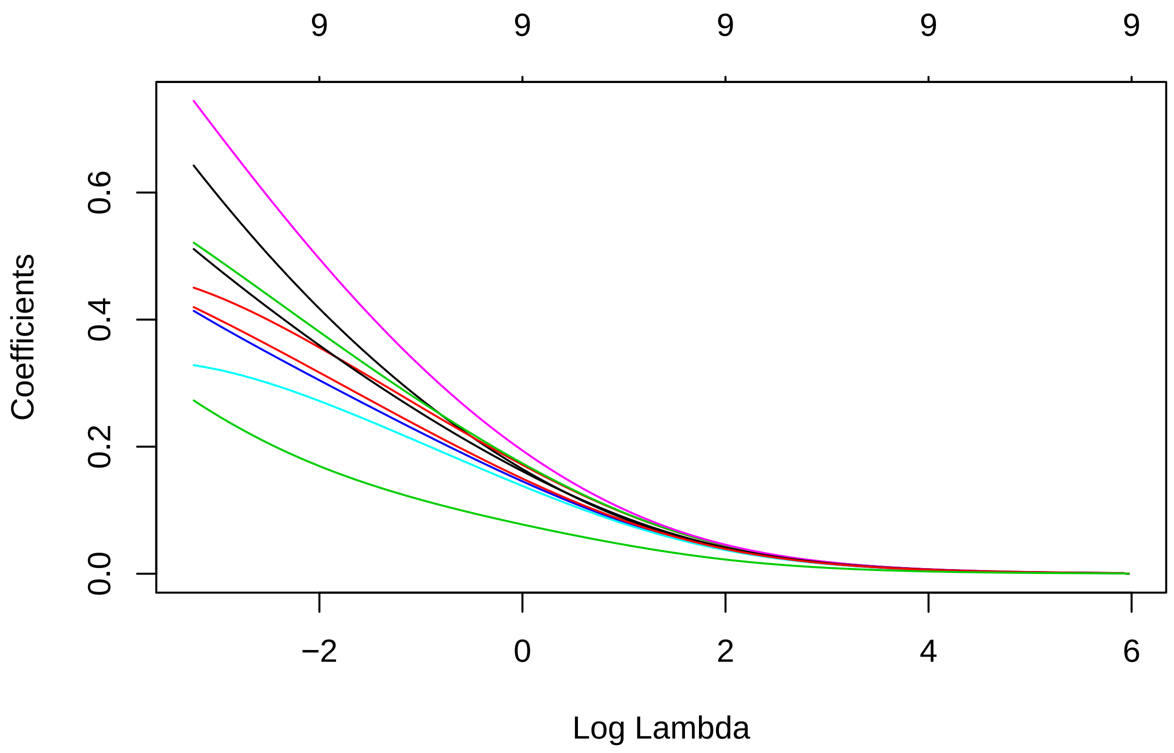
```
pretty_coefs(coef(lasso_logreg, s = exp(-1.2)))
```

```
## # A tibble: 4 x 2
##   predictor   coefficient
##   <chr>             <dbl>
## 1 (Intercept)      -0.642
## 2 V6                0.219
## 3 V2                0.138
## 4 V3                0.109
```

BACK TO PRESENTATION

Ridge and elastic net models

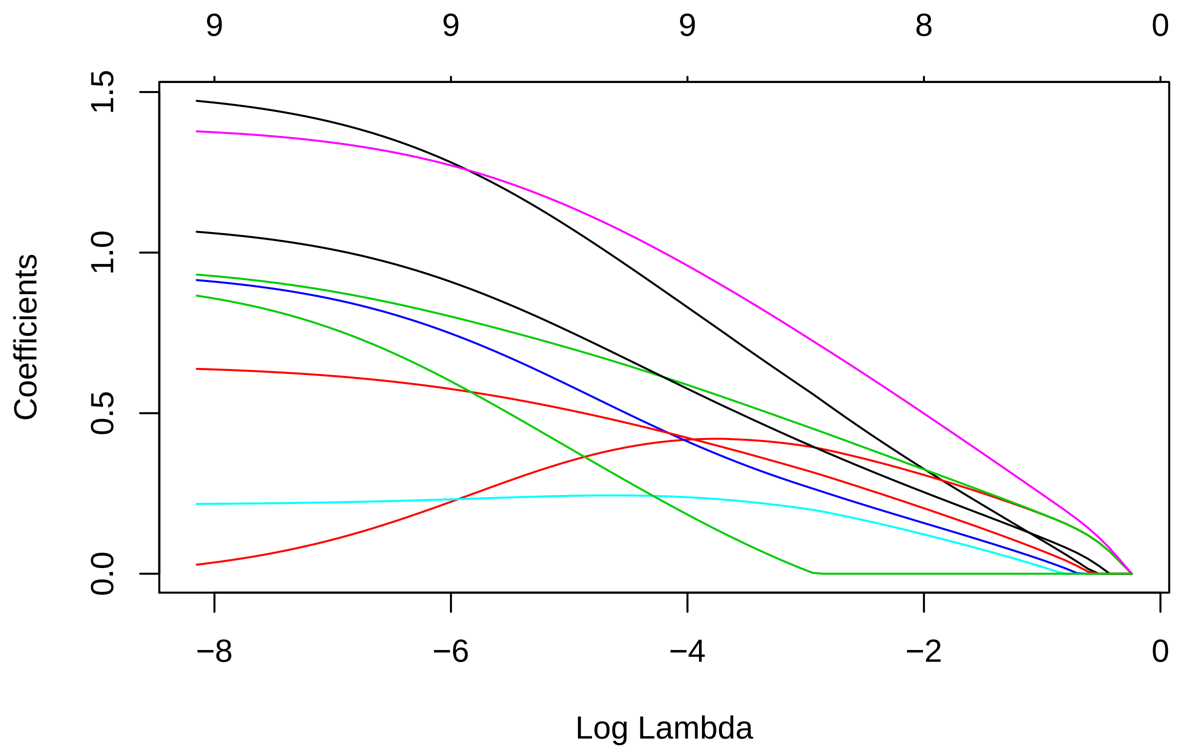
```
ridge_logreg <- update(lasso_logreg, alpha = 0)
plot(ridge_logreg, xvar = "lambda")
```



```
pretty_coefs(coef(ridge_logreg, s = exp(2))) # all shrunk but no real ranking or anything
```

```
## # A tibble: 10 x 2
##    predictor   coefficient
##    <chr>             <dbl>
##  1 (Intercept)     -0.630
##  2 V6               0.0457
##  3 V3               0.0448
##  4 V2               0.0447
##  5 V7               0.0414
##  6 V1               0.0395
##  7 V8               0.0391
##  8 V4               0.0383
##  9 V5               0.0373
## 10 V9               0.0224
```

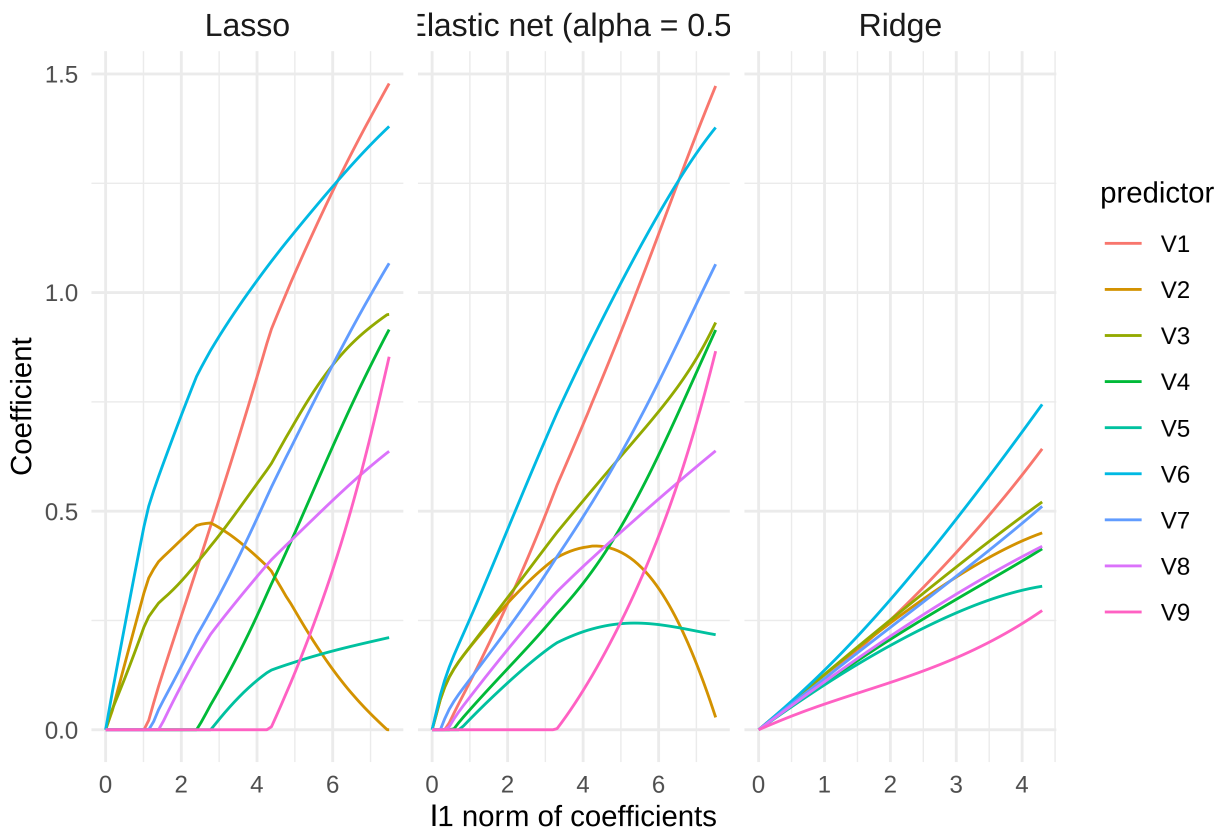
```
elastic_logreg <- update(lasso_logreg, alpha = 0.5)
plot(elastic_logreg, xvar = "lambda")
```



```
pretty_coefs(coef(elastic_logreg, s = exp(-1.2))) # all shrunk AND "ranking"
```

```
## # A tibble: 9 x 2
##   predictor   coefficient
##   <chr>             <dbl>
## 1 (Intercept)     -0.719
## 2 V6               0.298
## 3 V3               0.215
## 4 V2               0.213
## 5 V1               0.148
## 6 V7               0.141
## 7 V8               0.0996
## 8 V4               0.0675
## 9 V5               0.0435
```

```
ldply(list(Lasso = lasso_logreg, Ridge = ridge_logreg, Elastic_net = elastic_logreg),
      function(.) with(., data.frame(as.matrix(t(beta)), x = apply(abs(beta), 2, sum))), .id = "mod") %
    gather(predictor, value, -x, -mod) %>%
    mutate(mod = factor(mod, levels = c("Lasso", "Elastic_net", "Ridge"),
                        labels = c("Lasso", "Elastic net (alpha = 0.5)", "Ridge"))) %>%
    ggplot(aes(x, value, colour = predictor)) +
        geom_line() +
        labs(x = "l1 norm of coefficients", y = "Coefficient") +
        facet_wrap(~ mod, scales = "free_x") +
        theme(strip.text = element_text(size = 12))
```



Over-fitting biopsy

It's a little data set but let's try and split into training and test sets. There is quite some over-fitting on the relative scale, but the absolute prediction error difference is minor.

```
set.seed(42) # reproducible stochastic code
train_idx <- runif(nrow(biopsy_complete)) <= 0.8 # not exactly indices
mean(train_idx) # fraction in training set
```

```
## [1] 0.806735
```

```
biopsy_train <- filter(biopsy_complete, train_idx)
biopsy_test <- filter(biopsy_complete, !train_idx)
all_equal(biopsy_complete, bind_rows(biopsy_train, biopsy_test)) # sanity check
```

```
## [1] TRUE
```

```
# Alternative with actual indices (mostly a matter of taste)
set.seed(42)
train_idx <- which(runif(nrow(biopsy_complete)) <= 0.8)
length(train_idx) / nrow(biopsy_complete)
```

```
## [1] 0.806735
```

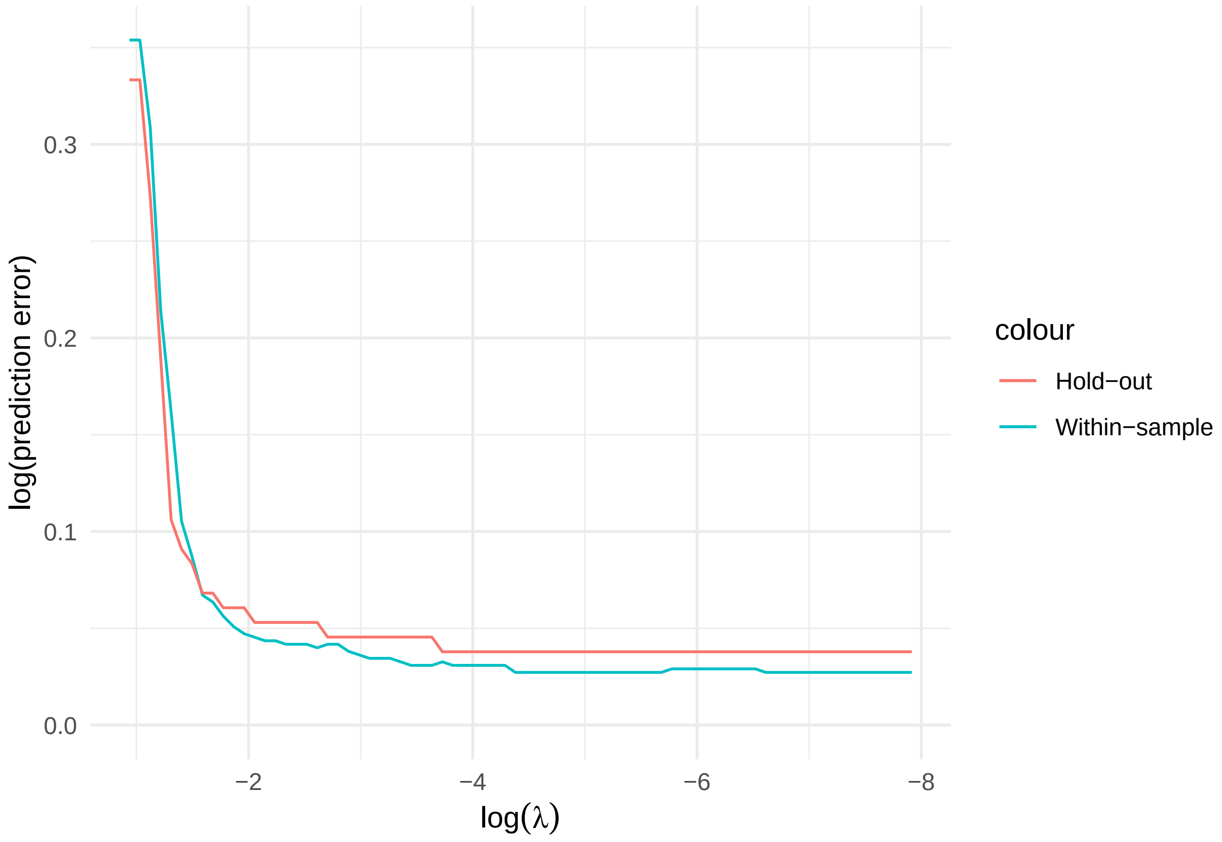
```
biopsy_train <- slice(biopsy_complete, train_idx)
biopsy_test <- slice(biopsy_complete, -train_idx)
all_equal(biopsy_complete, bind_rows(biopsy_train, biopsy_test)) # sanity check
```

```
## [1] TRUE
```

```
# Normalise predictors and put them in matrix format
predictors_train <- select(biopsy_train, -ID, -class) %>%
    scale()
predictors_test <- select(biopsy_test, -ID, -class) %>%
    scale()

# Train model
lasso_biopsy_train <- glmnet(predictors_train, biopsy_train$class, family = "binomial")
D_train <- predict(lasso_biopsy_train, predictors_train, type = "class") %>%
    apply(2, function(.) mean(. != biopsy_train$class)) # s39 is the best-performing

D_test <- predict(lasso_biopsy_train, predictors_test, type = "class") %>%
    apply(2, function(.) mean(. != biopsy_test$class))

tibble(D_test = D_test[which.min(D_train)],
       D_train = D_train[which.min(D_train)],
       D_diff_abs = scales::percent(D_test - D_train),
       D_diff_rel = scales::percent((D_test - D_train) / D_train, big.mark = ","))
```

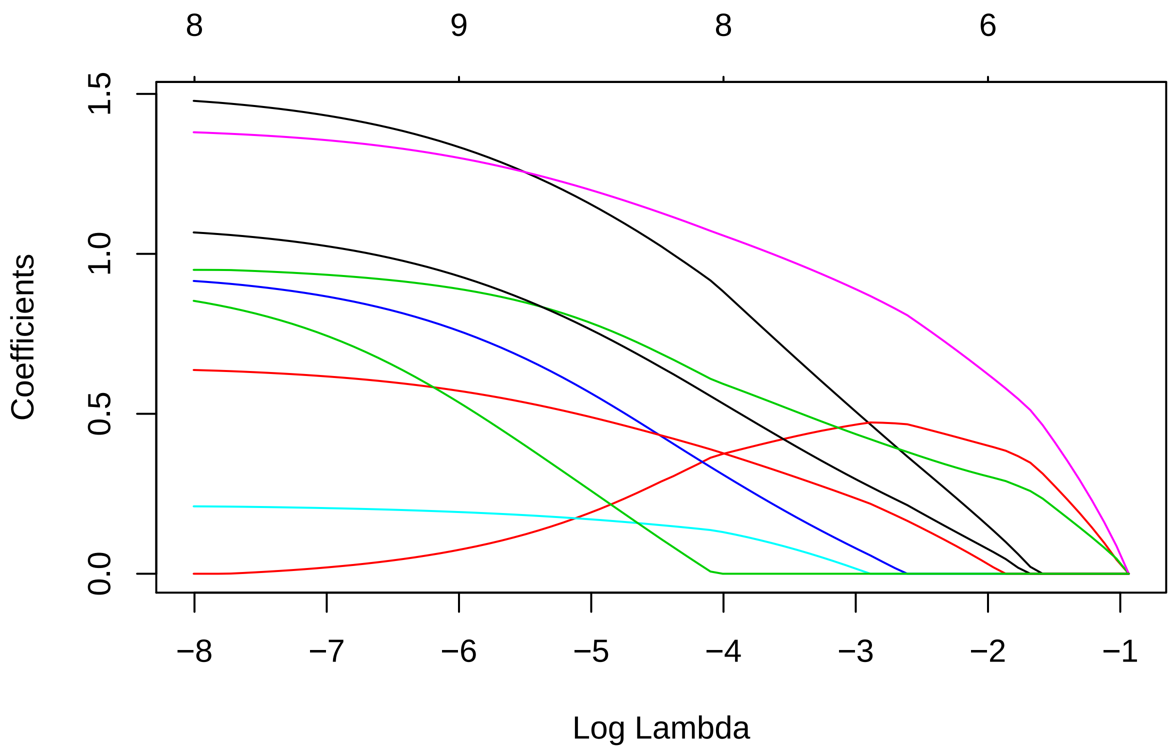
```
## # A tibble: 1 x 4
##   D_test D_train D_diff_abs D_diff_rel
##    <dbl>   <dbl> <chr>      <chr>
## 1 0.0379  0.0272 1%         39%
```

```
ggplot(mapping = aes(x = log(lasso_biopsy_train$lambda))) +
    coord_cartesian(ylim = c(0, NA)) +
    geom_line(aes(y = D_train, colour = "Within-sample")) +
    geom_line(aes(y = D_test, colour = "Hold-out")) +
    labs(y = "log(prediction error)", x = expression(log(lambda))) +
    scale_x_reverse()
```



Delassoing

```
plot(lasso_logreg, xvar = "lambda")
```



```
lambda <- exp(-4.5)
beta <- coef(lasso_logreg, x = biopsy_predictors, y = biopsy_complete$class,
             s = lambda/nrow(biopsy_complete), exact = TRUE)
delasso_fit <- fixedLassoInf(biopsy_predictors, as.numeric(biopsy_complete$class)-1, beta, lambda, "bin
                             alpha = 0.05)
```

```
## Warning in fixedLogitLassoInf(x, y, beta, lambda, alpha = alpha, type =
## type, : Solution beta does not satisfy the KKT conditions (to within specified
## tolerances)
```

```
delasso_fit
```

```
##
## Call:
## fixedLassoInf(x = biopsy_predictors, y = as.numeric(biopsy_complete$class) -
##     1, beta = beta, lambda = lambda, family = "binomial", alpha = 0.05)
##
## Testing results at lambda = 0.011, with alpha = 0.050
##
##  Var   Coef Z-score P-value LowConfPt UpConfPt LowTailArea UpTailArea
##    1  1.509   3.770   0.596    -9.749    2.005       0.025      0.024
##    2 -0.019  -0.030   0.989     0.857      Inf       0.025      0.000
##    3  0.964   1.399   0.962      -Inf    0.570       0.000      0.025
##    4  0.947   2.680   0.808   -17.165    1.237       0.025      0.024
##    5  0.215   0.617   0.863   -10.079    0.635       0.025      0.024
##    6  1.396   4.084   0.000     0.743    3.114       0.025      0.025
##    7  1.095   2.611   0.749   -14.329    1.537       0.025      0.025
##    8  0.650   1.889   0.805   -12.461    1.008       0.025      0.025
##    9  0.927   1.629   0.705   -10.952    1.724       0.025      0.025
```

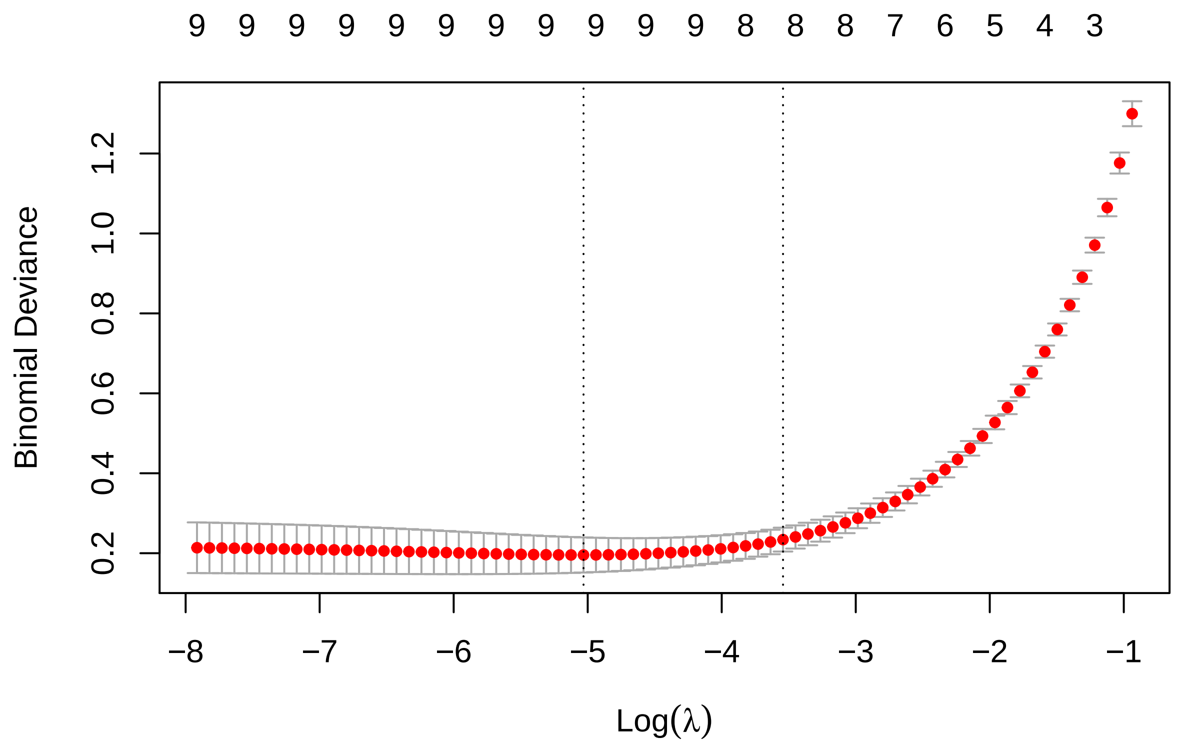
```
##
## Note: coefficients shown are full regression coefficients
```

Cross-validation

Use cross-validation to find best λ value. We found quite clear over-fitting above. Let's try to remedy this with cross-validation.

```
lasso_logreg_cv <- cv.glmnet(predictors_train, biopsy_train$class, family = "binomial")
```

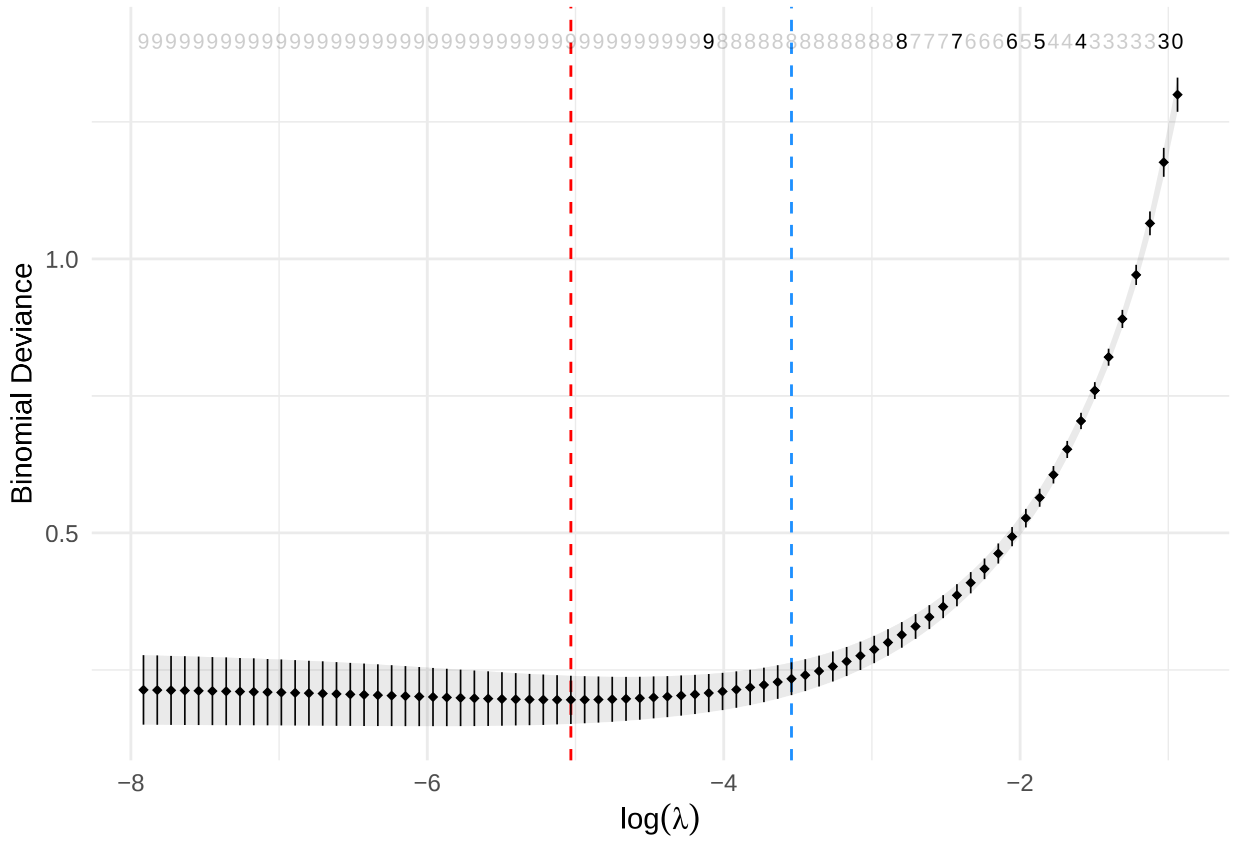
```
plot(lasso_logreg_cv)
```



```
with(lasso_logreg_cv, data.frame(lambda.min, lambda.1se))
```

```
##    lambda.min lambda.1se
## 1 0.006531203 0.02893729
```

```
# With ggplot2 (more control and prettier)
cv_glmnet_coef_profile <- function(fit) {
    fade_text <- function(x, alpha = 0.2) {
        ifelse(paste(x) == lag(paste(x), default = ""), alpha, 1)
    }
    every_n <- function(x, n = 5) {
        seq_along(x) %% n == 0
    }
    with(fit, tibble(log_lambda = log(lambda), cvm, cvup, cvlo, nzero)) %>%
        ggplot(aes(x = log_lambda)) +
            geom_vline(xintercept = log(fit$lambda.min), linetype = 2, size = 0.5, colour = "red") +
            geom_vline(xintercept = log(fit$lambda.1se), linetype = 2, size = 0.5, colour = "dodgerblue
            geom_linerange(aes(ymin = cvlo, ymax = cvup), size = 0.3) +
            geom_ribbon(aes(ymin = cvlo, ymax = cvup), alpha = 0.1) +
            geom_point(aes(y = cvm), shape = 18, size = 1.5) +
            geom_text(aes(y = max(cvup) * 1.05, label = nzero, alpha = fade_text(nzero)), size = 8 / ggp
            scale_alpha_identity() +
            labs(x = expression(log(lambda)), y = fit$name) +
            theme_minimal()
}
cv_glmnet_coef_profile(lasso_logreg_cv)
```



Test out-of-sample performance of the CV model

```
pred_min <- predict(lasso_logreg_cv, predictors_test, s = "lambda.min", type = "class")
pred_1se <- predict(lasso_logreg_cv, predictors_test, s = "lambda.1se", type = "class")
```

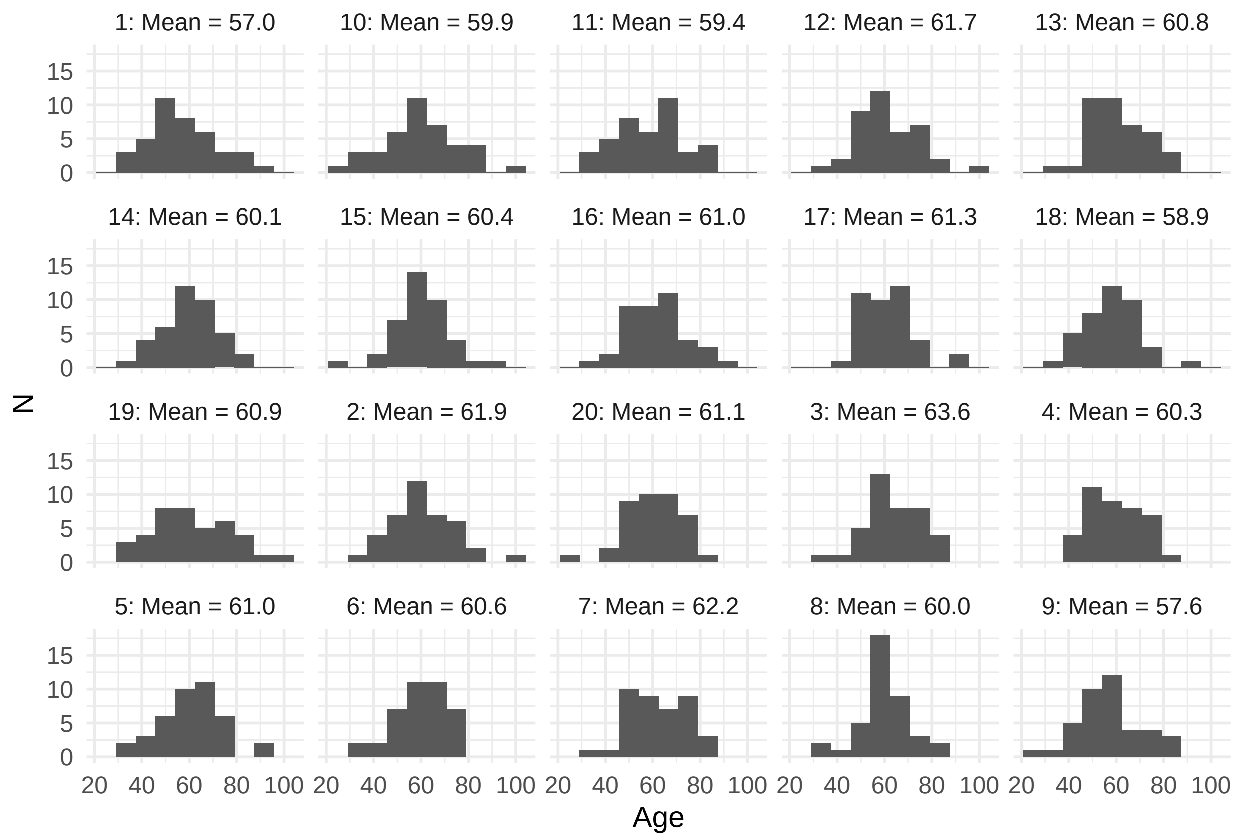
Cross-validation to pick best combination of λ and α

- grid search over alpha and lambda

Resampling methods: jackknife and bootstrap

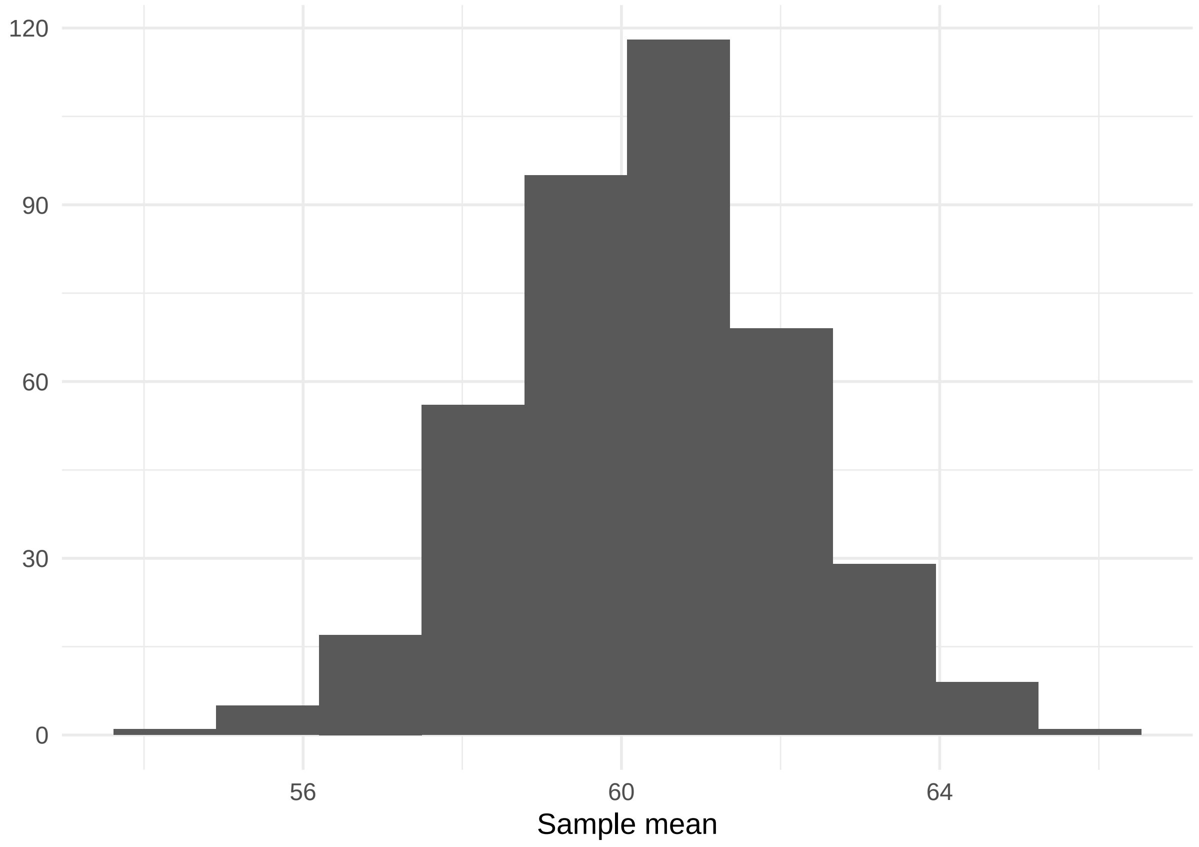
Sampling distribution

```
age <- rnorm(1000, 60, 12)
age <- age + 2 * (age < 30)

age_samples <- ldply(setNames(1:400, 1:400), function(i) {
        set.seed(42 + i)
        enframe(sample(age, 40, replace = FALSE))
    }, .id = "sample") %>%
    group_by(sample) %>%
    mutate(facet_title = sprintf("%s: Mean = %.1f", sample, mean(value)))

ggplot(filter(age_samples, sample %in% 1:20), aes(value)) +
    geom_histogram(bins = 10) +
    facet_wrap(~ facet_title, ncol = 5) +
    labs(x = "Age", y = "N")
```

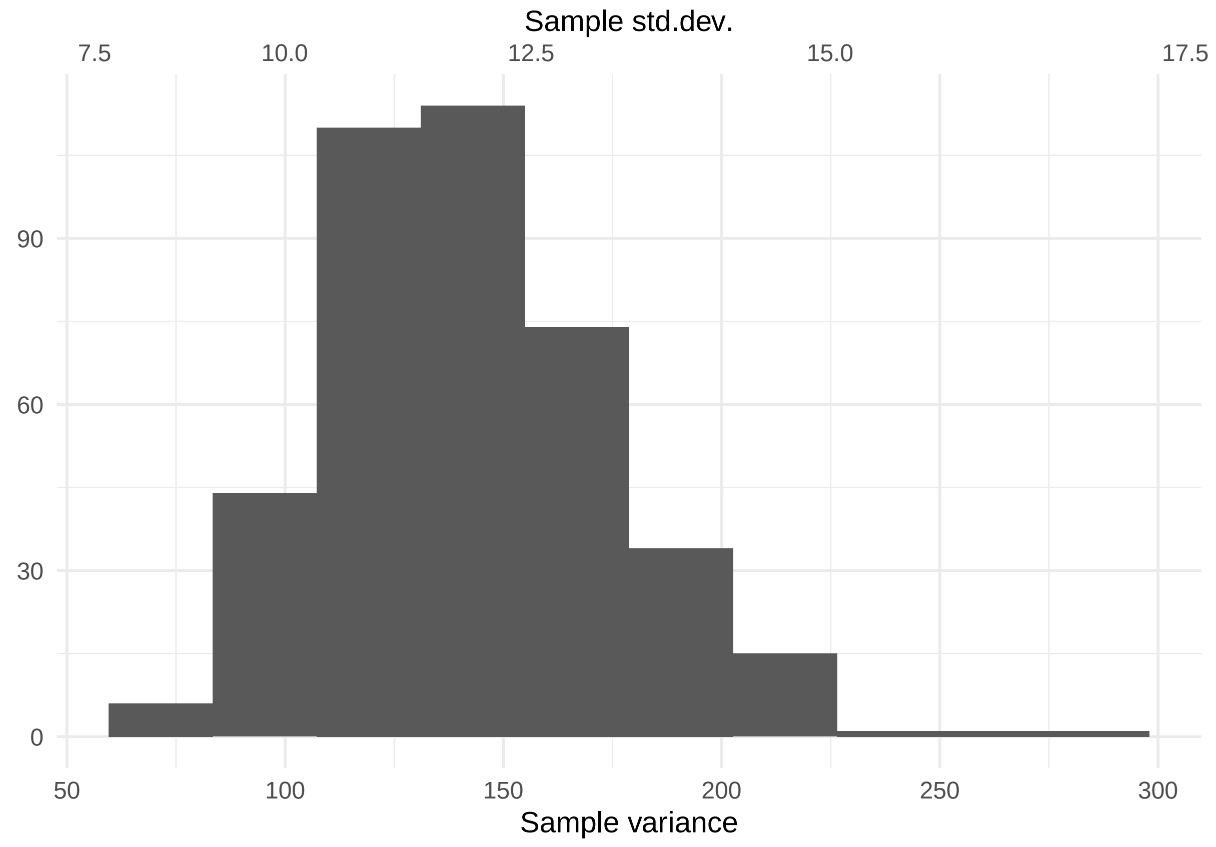


```
aggregate_stats_age <- group_by(age_samples, sample) %>%
    summarise("Sample mean" = mean(value),
              "Sample variance" = var(value),
              "Sample 90-percentile" = quantile(value, 0.9))

## `summarise()` ungrouping output (override with `.groups` argument)
```

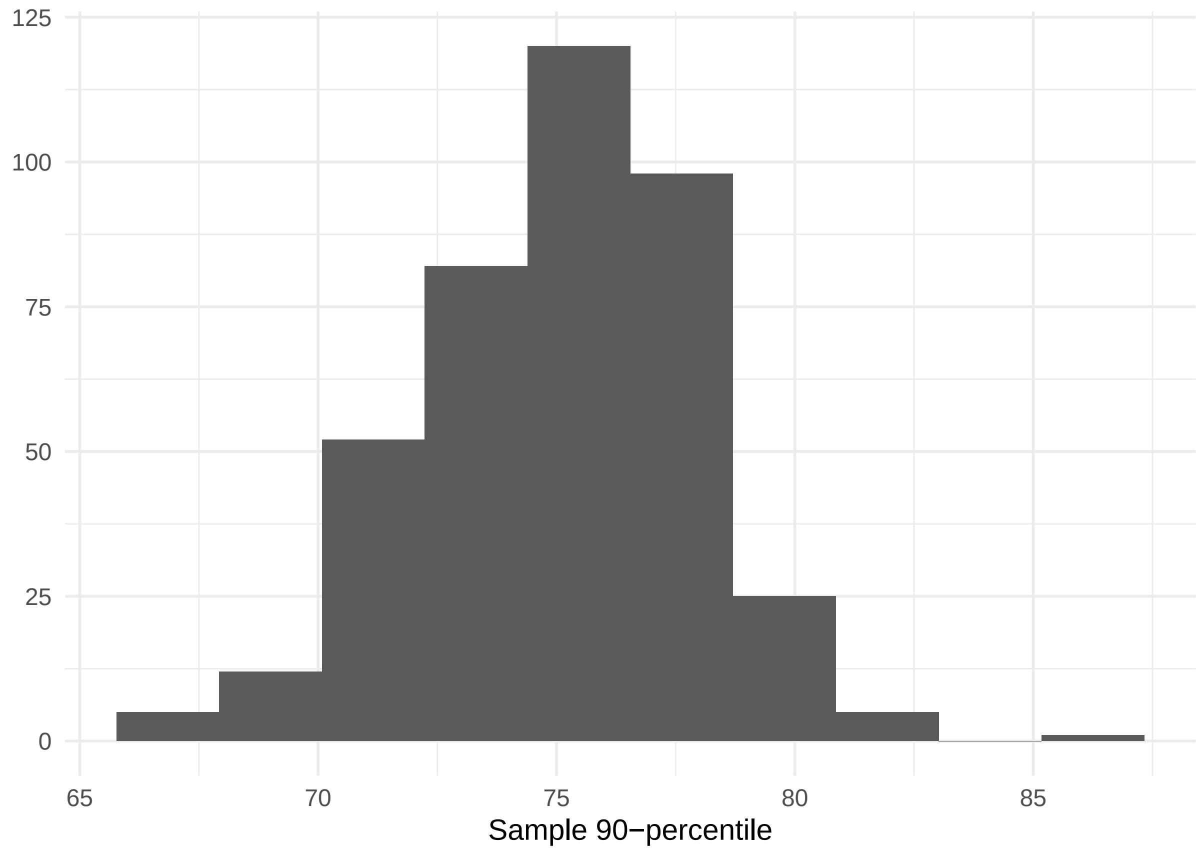
```
sample_plot <- llply(paste("Sample", c("mean", "variance", "90-percentile")),
      function(.) qplot(x = !!sym(.), data = aggregate_stats_age, geom = "histogram", bins = 10))

sample_plot[[1]]
```



```
sample_plot[[2]] + scale_x_continuous(sec.axis = sec_axis(sqrt, "Sample std.dev."))
```



```
sample_plot[[3]]
```



Jacknife

```
x <-c(8.26, 6.33, 10.4, 5.27, 5.35, 5.61, 6.12, 6.19, 5.2, 7.01, 8.74, 7.78, 7.02, 6, 6.5, 5.8, 5.12, 7
coef_var <- function(x) sqrt(var(x))/mean(x) # coefficient of variation
coef_var(x)
```

```
## [1] 0.2524712
```

```
library(bootstrap)
```

```
##
## Attaching package: 'bootstrap'
```

```
## The following object is masked from 'package:broom':
##
##     bootstrap
```

```
jackknife(x, coef_var)
```

```
## $jack.se
## [1] 0.05389943
##
## $jack.bias
## [1] -0.009266436
```

```
##
## $jack.values
##  [1] 0.2563873 0.2565586 0.2384298 0.2507329 0.2513200 0.2530603 0.2557374
##  [8] 0.2560293 0.2501992 0.2580969 0.2541045 0.2577524 0.2581067 0.2551946
## [15] 0.2571038 0.2541711 0.2495662 0.2581975 0.2571609 0.2561093 0.2020978
## [22] 0.2529980 0.2515338 0.2573745 0.2541045
##
## $call
## jackknife(x = x, theta = coef_var)
```

Exercise: cross-validation

```
library(MASS)
data(biopsy)
biopsy_complete <- na.exclude(biopsy)
summary(biopsy_complete)
```

```
##       ID                  V1               V2               V3
##  Length:683         Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  Class :character   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000
##  Mode  :character   Median : 4.000   Median : 1.000   Median : 1.000
##                     Mean   : 4.442   Mean   : 3.151   Mean   : 3.215
##                     3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000
##                     Max.   :10.000   Max.   :10.000   Max.   :10.000
##        V4              V5               V6               V7
##  Min.   : 1.00   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.: 1.00   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 2.000
##  Median : 1.00   Median : 2.000   Median : 1.000   Median : 3.000
##  Mean   : 2.83   Mean   : 3.234   Mean   : 3.545   Mean   : 3.445
##  3rd Qu.: 4.00   3rd Qu.: 4.000   3rd Qu.: 6.000   3rd Qu.: 5.000
##  Max.   :10.00   Max.   :10.000   Max.   :10.000   Max.   :10.000
##        V8              V9               class
##  Min.   : 1.00   Min.   : 1.000   benign   :444
##  1st Qu.: 1.00   1st Qu.: 1.000   malignant:239
##  Median : 1.00   Median : 1.000
##  Mean   : 2.87   Mean   : 1.603
##  3rd Qu.: 4.00   3rd Qu.: 1.000
##  Max.   :10.00   Max.   :10.000
```

```
predictors <- biopsy_complete %>%
    select(-ID, -class)
pca_fit <- prcomp(predictors, scale = TRUE)
df_pca <- data.frame(pca_fit$x[, 1:4], outcome = biopsy_complete$class)
glm_fit <- glm(outcome ~ PC1 + PC2 + PC3 + PC4, data = df_pca, family = binomial)
summary(glm_fit)
```

```
##
## Call:
## glm(formula = outcome ~ PC1 + PC2 + PC3 + PC4, family = binomial,
##     data = df_pca)
##
## Deviance Residuals:
```

```
##     Min       1Q   Median       3Q      Max
## -3.1791  -0.1304  -0.0619   0.0228   2.4799
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.0739     0.3035  -3.539 0.000402 ***
## PC1          -2.4140     0.2556  -9.445  < 2e-16 ***
## PC2          -0.1592     0.5050  -0.315 0.752540
## PC3           0.7191     0.3273   2.197 0.028032 *
## PC4          -0.9151     0.3691  -2.479 0.013159 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 884.35  on 682  degrees of freedom
## Residual deviance: 106.12  on 678  degrees of freedom
## AIC: 116.12
##
## Number of Fisher Scoring iterations: 8
```

```
tidy(glm_fit)
```

```
## # A tibble: 5 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   -1.07      0.303    -3.54  4.02e- 4
## 2 PC1           -2.41      0.256    -9.44  3.56e-21
## 3 PC2           -0.159     0.505    -0.315 7.53e- 1
## 4 PC3            0.719     0.327     2.20  2.80e- 2
## 5 PC4           -0.915     0.369    -2.48  1.32e- 2
```

1. LOO-CV error rate

```
library(boot)
glm_fit_loocv <- cv.glm(df_pca, glm_fit)
glm_fit_loocv$delta
```

```
## [1] 0.02301890 0.02301788
```

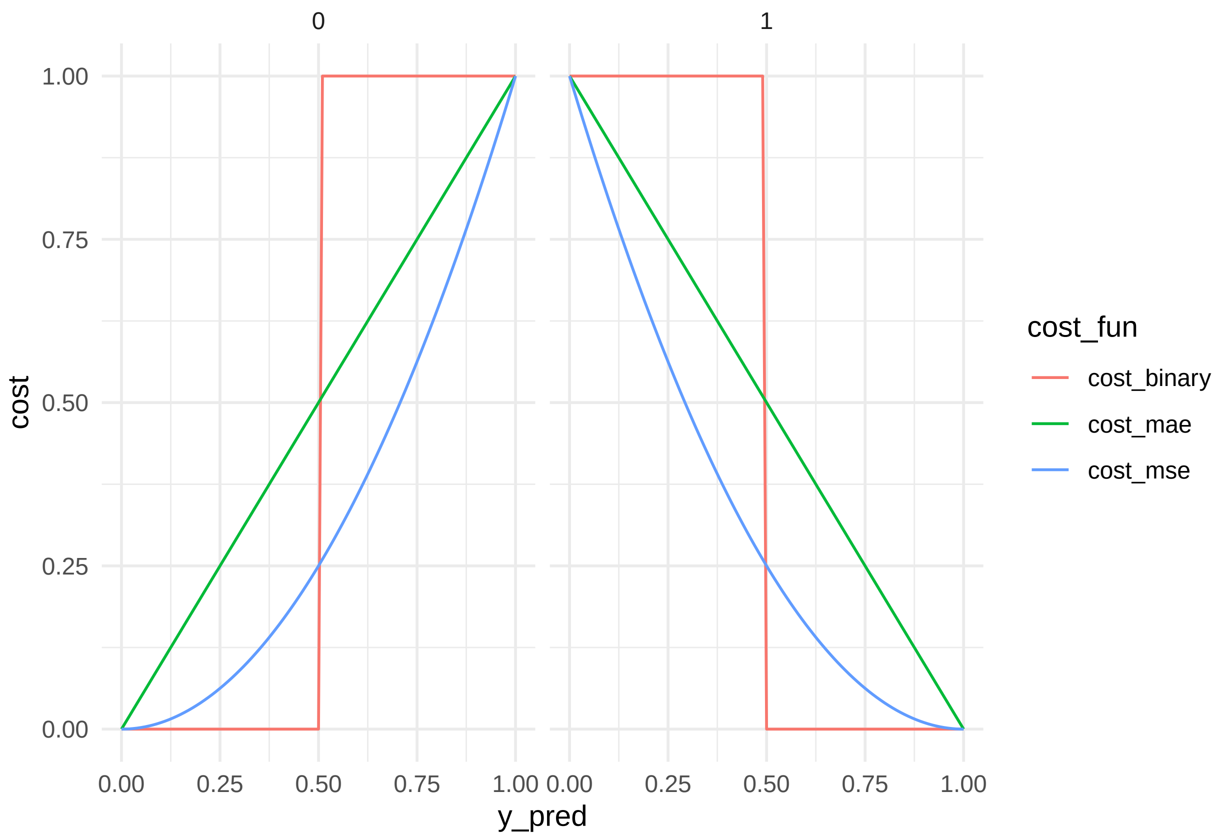
2. Use proper cost function

```
glm_fit_loocv2 <- cv.glm(df_pca, glm_fit, cost = function(r, pi = 0) mean(abs(r-pi) > 0.5))
glm_fit_loocv2$delta
```

```
## [1] 0.02781845 0.02785918
```

3. Difference between error rates, and their interpretation

```
# The binary cost function forces the predictions into a binary class assignment (w.r.t. the 0.5 thresh
expand.grid(y_obs = 0:1,
            y_pred = 0:100 / 100) %>%
    rowwise() %>%
    mutate(cost_mse= mean((y_pred - y_obs)^2),
           cost_binary = mean(abs(y_pred - y_obs) > 0.5),
           cost_mae = mean(abs(y_pred - y_obs))) %>%
    pivot_longer(starts_with("cost_"), names_to = "cost_fun", values_to = "cost") %>%
    ggplot(aes(y_pred, cost, colour = cost_fun)) +
        geom_line() +
        facet_wrap(~ y_obs)
```



4. 10-fold CV

```
# The error rates change quite a bit, which makes sense because 10-fold CV has a lot fewer folds than L
glm_fit_10cv <- update(glm_fit_loocv, K = 10)
glm_fit_10cv$delta
```

```
## [1] 0.02412049 0.02399288
```

```
# glm_fit_10cv2 <- cv.glm(df_pca, glm_fit, cost = function(r, pi = 0) mean(abs(r-pi) > 0.5), K = 10)
glm_fit_10cv2 <- update(glm_fit_loocv2, K = 10)
glm_fit_10cv2$delta
```

```
## [1] 0.03074671 0.03118830
```

```
ldply(list("LOO" = glm_fit_loocv, "LOO 2" = glm_fit_loocv2, "10-fold" = glm_fit_10cv, "10-fold 2" = glm_
      with, delta) %>%
    setNames(c("model", "error_rate", "corrected_error_rate"))
```

```
##       model error_rate corrected_error_rate
## 1       LOO 0.02301890           0.02301788
## 2     LOO 2 0.02781845           0.02785918
## 3   10-fold 0.02412049           0.02399288
## 4 10-fold 2 0.03074671           0.03118830
```

```
knitr::opts_chunk$set(include = FALSE)
```

Exercise: penalised regression

1. + 2. Lasso regression

3. Why does it normally make sense to normalise predictors

4. Using CV to get reasonable estimate of λ

5. Obtain coefficients for "best" λ

6. Re-fit with correct family

Quite some difference between the sets of predictors kept:

7. As ridge regression

8. Get idea about sparse solution using ridge results?

9. Elastic net ($\alpha = 0.5$)

Ideally, one should do CV over α as well.

10. Delasso the results

11. Selective inference