PSET 07 - CV, Ridge, Lasso, Random Intercepts

S&DS 361

Due 2024-04-26

Landcover

```
load('data/labeled_points.Rdata')

labeled = labeled %>%
  select(ID, landcover)

dl = labeled_train %>%
  left_join(labeled, by = 'ID') %>%
  mutate(veg = ifelse(landcover %in% c('natforest', 'orchard', 'cropland'),
                      1, 0),
         NDVI100 = NDVI*100,
         NDBI100 = NDBI*100)
```

Let's compute mean band values for each location.

```
dm = dl %>%
  group_by(ID) %>%
  summarise(
    B1   = mean(B1, na.rm=T),
    B2   = mean(B2, na.rm=T),
    B3   = mean(B3, na.rm=T),
    B4   = mean(B4, na.rm=T),
    B5   = mean(B5, na.rm=T),
    B6_VCID_1 = mean(B6_VCID_1, na.rm=T),
    B6_VCID_2 = mean(B6_VCID_2, na.rm=T), ##  cor is .998 with B6_VCID_1
    B7        = mean(B7, na.rm=T),
    NDVI100   = mean(NDVI100, na.rm=T),
    #NDBI100   = mean(NDBI100, na.rm=T), ## causes warnings with lasso
    #EVI  = mean(EVI, na.rm=T),
    landcover = unique(landcover),
    veg = unique(veg)) %>%
  select(-ID, -landcover)


ht(dm) ## head and tail, each with 2 rows
```

```
## # A tibble: 2 x 10
##      B1    B2    B3    B4    B5 B6_VCID_1 B6_VCID_2    B7 NDVI100   veg
##   <dbl> <dbl> <dbl> <dbl> <dbl>     <dbl>     <dbl> <dbl>   <dbl> <dbl>
## 1  80.4  69.2  73.4  91.4  79        130.      148.  51.3    15.1     1
## 2  83.5  73.5  78.9  94.6  87.1      131.      150   57.9    13.4     1
##   ...
```

```
## # A tibble: 2 x 10
##      B1    B2    B3    B4    B5 B6_VCID_1 B6_VCID_2    B7 NDVI100   veg
##   <dbl> <dbl> <dbl> <dbl> <dbl>     <dbl>     <dbl> <dbl>   <dbl> <dbl>
## 1  88.8  79.3  89.7  98.3  104.      133.      153.  70.1    8.41     1
## 2  92.2  87   102.  101.   118.      134.      155.  88.5    2.06     1
```

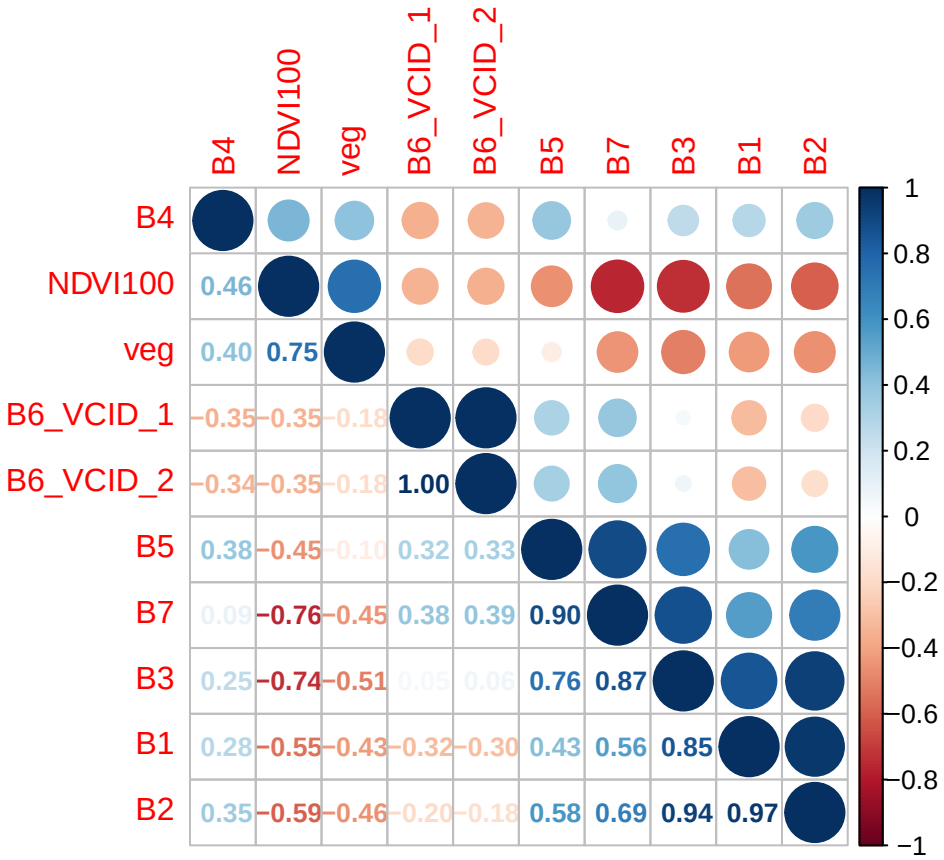
Data Exploration

Recall that there was a lot of collinearity among the mean band values at each location.

```
corr = dm %>%
  cor(use = 'pairwise.complete.obs')
corr %>% round(2)
```

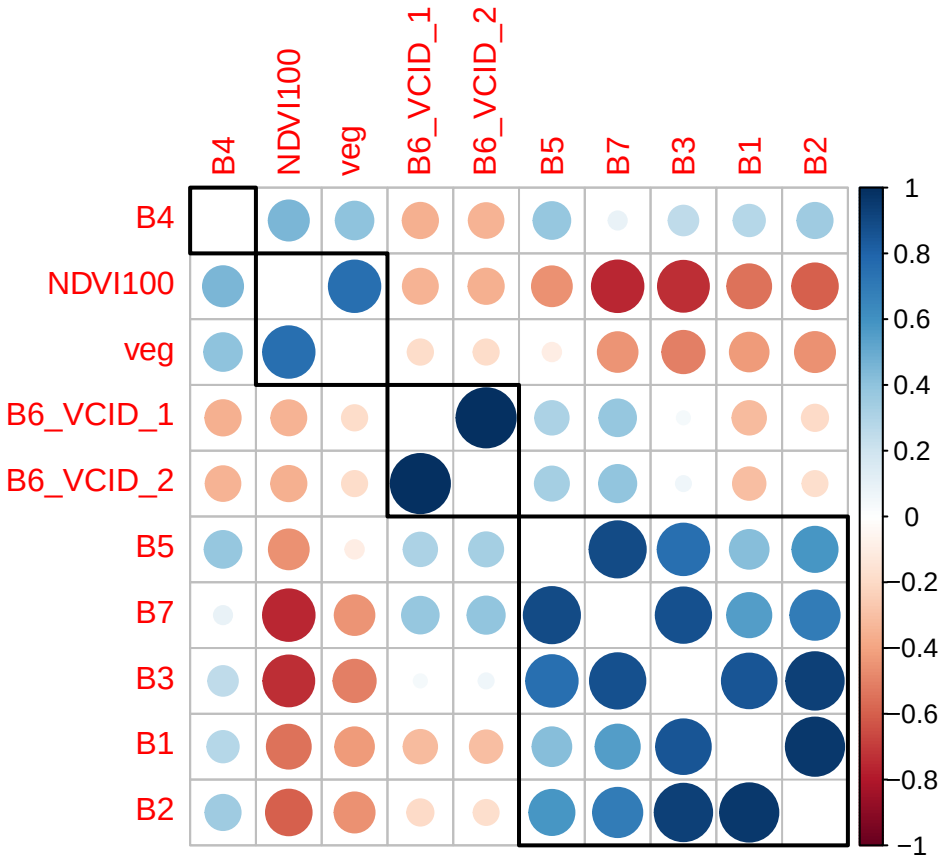
```
##              B1    B2    B3    B4    B5 B6_VCID_1 B6_VCID_2    B7 NDVI100   veg
## B1         1.00  0.97  0.85  0.28  0.43     -0.32     -0.30  0.56   -0.55 -0.43
## B2         0.97  1.00  0.94  0.35  0.58     -0.20     -0.18  0.69   -0.59 -0.46
## B3         0.85  0.94  1.00  0.25  0.76      0.05      0.06  0.87   -0.74 -0.51
## B4         0.28  0.35  0.25  1.00  0.38     -0.35     -0.34  0.09    0.46  0.40
## B5         0.43  0.58  0.76  0.38  1.00      0.32      0.33  0.90   -0.45 -0.10
## B6_VCID_1 -0.32 -0.20  0.05 -0.35  0.32      1.00      1.00  0.38   -0.35 -0.18
## B6_VCID_2 -0.30 -0.18  0.06 -0.34  0.33      1.00      1.00  0.39   -0.35 -0.18
## B7         0.56  0.69  0.87  0.09  0.90      0.38      0.39  1.00   -0.76 -0.45
## NDVI100   -0.55 -0.59 -0.74  0.46 -0.45     -0.35     -0.35 -0.76    1.00  0.75
## veg       -0.43 -0.46 -0.51  0.40 -0.10     -0.18     -0.18 -0.45    0.75  1.00
```

```
corrplot(corr, order = 'hclust', diag=T, type = 'upper', tl.pos = 'tp')
corrplot(corr, order = 'hclust', diag=F, type = 'lower', tl.pos =  'n',
         method='number',
         cl.pos = 'n',
         add = T,
         number.cex = 0.8)
```



Using `order = "hclust"` with rectangles added

```
corrplot(corr,
         diag = F,
         order = 'hclust',
         addrect = 4)
```



We are interested once again in modeling the probability of `veg` as a function of the mean band values. We will compare the following four models:

- logistic regression (with no regularization) using only `NDVI100`
- logistic regression (with no regularization) using all band values
- logistic regression with ridge regularization using all band values
- logistic regression with lasso regularization using all band values

Note that for logistic regression with regularization (3rd and 4th models), you can use the `cv.glmnet` function in the `glmnet` package as we did with linear regression, but with the argument `family = binomial` added.

1. Models with all of the data

Fit models with all of the data and find (in-sample) predicted probabilities for each observation. Explore the trace curves and discuss any notable observations.

```
## logistic regression with no regularization using only NDVI100
m1 = glm(veg ~ NDVI100, data = dm, family = binomial)
dm$predm1 <- predict(m1, type = 'response', newdata = dm)
summary(m1)
```

```
##
## Call:
## glm(formula = veg ~ NDVI100, family = binomial, data = dm)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.1318     0.2599  -0.507    0.612
## NDVI100       0.7953     0.1029   7.729 1.09e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 449.87  on 399  degrees of freedom
## Residual deviance: 105.95  on 398  degrees of freedom
## AIC: 109.95
##
## Number of Fisher Scoring iterations: 8
```

```
## logistic regression with no regularization using all band values
m2 = glm(veg ~ ., data = dm, family = binomial)
dm$predm2 <- predict(m2, type = 'response', newdata = dm)
summary(m2)
```

```
##
## Call:
## glm(formula = veg ~ ., family = binomial, data = dm)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 178.4566   116.6565   1.530  0.12608
## B1            0.9808     0.4390   2.234  0.02549 *
## B2           -1.6828     0.6943  -2.424  0.01537 *
## B3            1.0323     0.5807   1.778  0.07547 .
## B4           -0.5766     0.5156  -1.118  0.26340
## B5            0.6270     0.2004   3.129  0.00175 **
## B6_VCID_1    -4.1567     2.5178  -1.651  0.09876 .
## B6_VCID_2     2.3382     1.4277   1.638  0.10149
## B7           -0.5541     0.2218  -2.499  0.01246 *
## NDVI100       1.7048     0.9351   1.823  0.06830 .
## predm1       -2.7813     4.1185  -0.675  0.49946
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 449.868  on 399  degrees of freedom
## Residual deviance:  54.646  on 389  degrees of freedom
## AIC: 76.646
##
## Number of Fisher Scoring iterations: 10
```

```
## logistic regression with ridge regularization using all band values
x = model.matrix(veg ~ B1 + B2 + B3 + B4 + B5 +
                   B6_VCID_1 + B6_VCID_2 + B7 + NDVI100, data = dm)[,-1]
y <- dm$veg
m3 <- cv.glmnet(x, y, family = 'binomial', alpha = 0)
dm$predm3 <- predict(m3, newx = x, s = 'lambda.1se', type = 'response')
summary(m3)
```

```
##            Length Class  Mode
## lambda     100    -none- numeric
## cvm        100    -none- numeric
## cvsd       100    -none- numeric
## cvup       100    -none- numeric
## cvlo       100    -none- numeric
## nzero      100    -none- numeric
## call         5    -none- call
## name         1    -none- character
## glmnet.fit  13    lognet list
## lambda.min   1    -none- numeric
## lambda.1se   1    -none- numeric
## index        2    -none- numeric
```

```
## logistic regression with lasso regularization using all band values
m4 <- cv.glmnet(x, y, family = 'binomial', alpha = 1)
dm$predm4 <- predict(m4, newx = x, s = 'lambda.1se', type = 'response')
summary(m4)
```

```
##            Length Class  Mode
## lambda     100    -none- numeric
## cvm        100    -none- numeric
## cvsd       100    -none- numeric
## cvup       100    -none- numeric
## cvlo       100    -none- numeric
## nzero      100    -none- numeric
## call         5    -none- call
## name         1    -none- character
## glmnet.fit  13    lognet list
## lambda.min   1    -none- numeric
## lambda.1se   1    -none- numeric
## index        2    -none- numeric
```

```
head(dm)
```

```
## # A tibble: 6 x 14
##      B1    B2    B3    B4    B5 B6_VCID_1 B6_VCID_2    B7 NDVI100   veg   predm1
##   <dbl> <dbl> <dbl> <dbl> <dbl>     <dbl>     <dbl> <dbl>   <dbl> <dbl>    <dbl>
## 1  80.4  69.2  73.4  91.4  79        130.      148.  51.3   15.1      1 1.00
## 2  83.5  73.5  78.9  94.6  87.1      131.      150   57.9   13.4      1 1.00
## 3  91.0  81.7  95.0  97.1 111.       131.      149.  79.2    4.71     1 0.974
## 4  87.9  78.3  87.8  95.3 102.       131.      150.  71.1    7.67     1 0.997
```

```
## 5  85.9  77.4  91.9  75.1  93.0      139.      164.  77.0   -9.53     0 0.000448
## 6  85.9  77.4  91.9  75.1  93.0      139.      164.  77.0   -9.53     0 0.000448
## # i 3 more variables: predm2 <dbl>, predm3 <dbl[,1]>, predm4 <dbl[,1]>
```

To find the trace curves, we need to plot the coefficients as functions of lambda:

```
coefs.m3 = coef(m3, s = m3$lambda)
coefs.m4 = coef(m4, s = m4$lambda)
colnames(coefs.m3) = paste0('lambda', round(m3$lambda,6))
colnames(coefs.m4) = paste0('lambda', round(m4$lambda,6))

coefs.m3   <- coefs.m3   %>%
  as.matrix() %>%
  as.data.frame() %>%
  rownames_to_column() %>%
  pivot_longer(cols=-rowname) %>%
  mutate(model='Ridge')

coefs.m4   <- coefs.m4   %>%
  as.matrix() %>%
  as.data.frame() %>%
  rownames_to_column() %>%
  pivot_longer(cols=-rowname) %>%
  mutate(model='Lasso')

# bind rows
coefs1 = bind_rows(coefs.m3, coefs.m4) %>%
  mutate(name = as.numeric(gsub('lambda', '', name))) %>%
  filter(rowname!='(Intercept)') %>%
  rename(lambda=name,
         var = rowname)

head(coefs1)
```

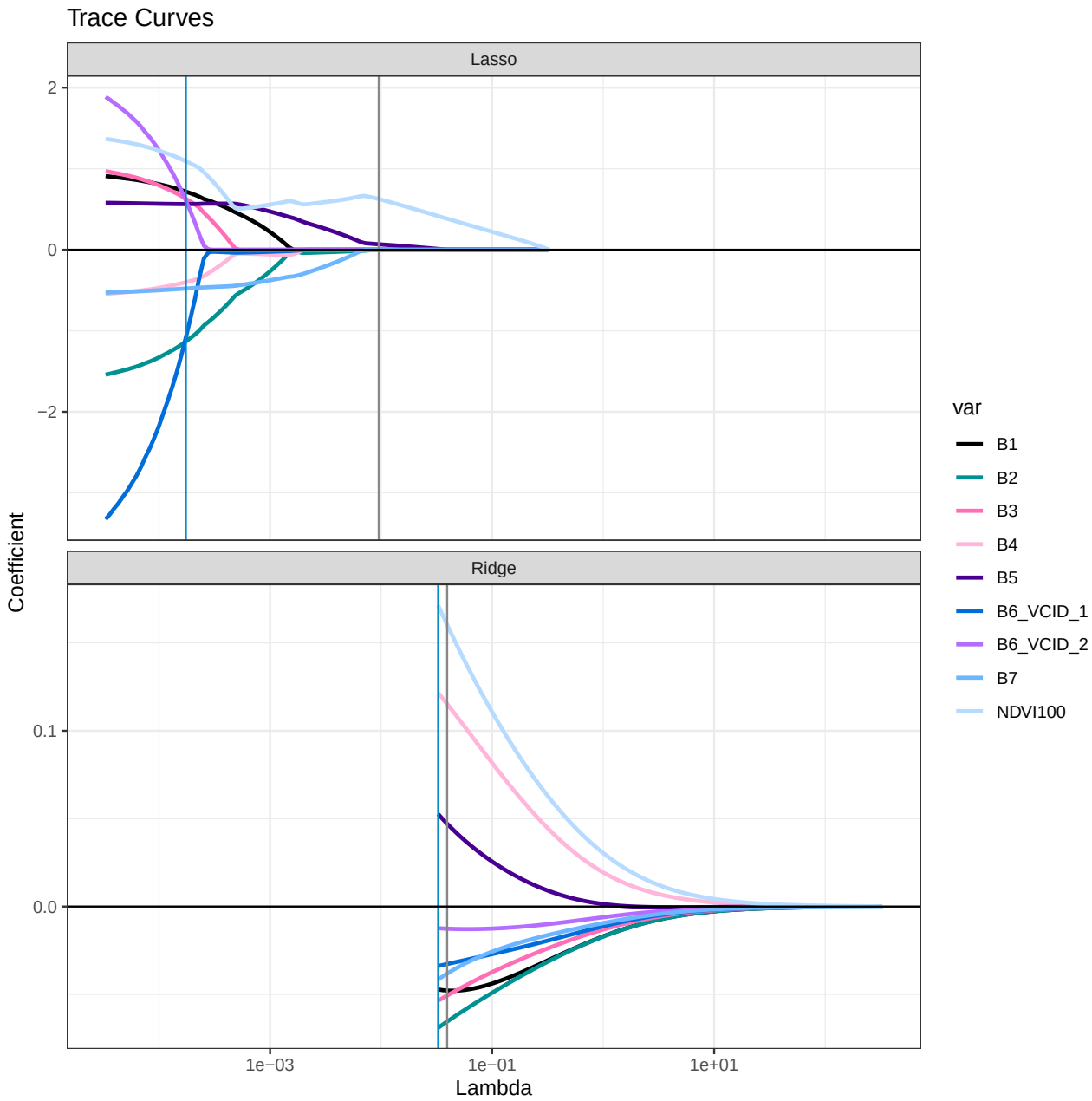
```
## # A tibble: 6 x 4
##   var   lambda     value model
##   <chr>  <dbl>     <dbl> <chr>
## 1 B1      327. -2.94e-38 Ridge
## 2 B1      298. -9.76e- 5 Ridge
## 3 B1      271. -1.07e- 4 Ridge
## 4 B1      247. -1.18e- 4 Ridge
## 5 B1      225. -1.29e- 4 Ridge
## 6 B1      205. -1.42e- 4 Ridge
```

Now we can plot the trace curves.

```
lambda.lines = data.frame(model=c('Ridge', 'Lasso'),
                          lambda.min = c(m3$lambda.min, m4$lambda.min),
                          lambda.1se = c(m3$lambda.1se, m4$lambda.1se))
dg = coefs1

g = ggplot(data=dg, aes(x=lambda, y=value, group=var, color=var))+
    geom_line(alpha=1, linewidth=1)+
    facet_wrap(~model, ncol=1, scales='free_y')+
    geom_vline(data=lambda.lines, aes(xintercept=lambda.min),
               color = pubblue)+
```

```
    geom_vline(data=lambda.lines, aes(xintercept=lambda.1se),
               color = pubmediumgray)+
    scale_x_log10()+
    geom_hline(yintercept = 0)+
    scale_color_manual(values=cb.pal) +
    theme_bw() +
    labs(title='Trace Curves', x='Lambda', y='Coefficient')
```

```
g
```



It seems the coefficients for the Ridge model are much more stable than the ones for the Lasso model, as most of the coefficients in the Ridge model are close between -0.1 and 0.1 for every lambda value, but in the Lasso model, the coefficients are more spread out, from around -3 to 2. Also, for the Ridge model, we see that only 3 band values are positive for all values of Lambda, namely B5, B4, and NDVI100. For the Lasso

model, we see that B3, B6_VCID_2, B1, and NDVI100 are positive, but B4 becomes negative.

Also, in the Lasso model, we see that most bands collapse to 0 as the values for Lambda increase. However, for NDVI100, it seems to decrease, then start following a linear pattern, and then decrease again. Additionally, in the Ridge model, it's interesting how most bands follow the same pattern, approaching 0 exponentially as Lambda increases, but B1 seems to first go farther away from 0 and then approach 0 exponentially like the other bands.

2. Cross-validation

Use cross-validation to make out of sample predictions for each observation and show the first 6 rows of the resulting data frame.

```
# for reproducibility
set.seed(123)

# set number of folds (usually 5 or 10)
k=5

# create a vector of fold numbers, repeating 1:k until reaching the number of rows in the data
folds = rep(1:k,
            length.out = nrow(dm))

# create a new column in the data frame that randomly assigns a fold to each row without replacement
# (so that we don't have a row assigned to two or more folds)
dm$fold = sample(folds,
                 nrow(dm),
                 replace = F)

# view the first few rows of the data frame to see the fold assignments
head(dm)
```

```
## # A tibble: 6 x 15
##      B1    B2    B3    B4    B5 B6_VCID_1 B6_VCID_2    B7 NDVI100   veg   predm1
##   <dbl> <dbl> <dbl> <dbl> <dbl>     <dbl>     <dbl> <dbl>   <dbl> <dbl>    <dbl>
## 1  80.4  69.2  73.4  91.4  79        130.      148.  51.3   15.1      1 1.00
## 2  83.5  73.5  78.9  94.6  87.1      131.      150   57.9   13.4      1 1.00
## 3  91.0  81.7  95.0  97.1 111.       131.      149.  79.2    4.71     1 0.974
## 4  87.9  78.3  87.8  95.3 102.       131.      150.  71.1    7.67     1 0.997
## 5  85.9  77.4  91.9  75.1  93.0      139.      164.  77.0   -9.53     0 0.000448
## 6  85.9  77.4  91.9  75.1  93.0      139.      164.  77.0   -9.53     0 0.000448
## # i 4 more variables: predm2 <dbl>, predm3 <dbl[,1]>, predm4 <dbl[,1]>,
## #   fold <int>
```

```
# create a data frame to keep track of the metrics
metrics = data.frame(fold = 1:k,
                     llr1se = NA,
                     llrmin = NA,
                     lllas1se = NA,
                     lllasmin = NA,
                     llm1 = NA,
                     llm2 = NA)

# initialize the prediction columns
dm[, c('r.1se', 'las.1se', 'r.min', 'las.min', 'm1pred', 'm2pred')] = NA
```

```r
# loop through the folds
for (j in 1:k){
  cat(j,'')

  ## train rows are the ones not in the j-th fold
  train.rows <- dm$fold != j

  # test rows are the observations in the j-th fold
  test.rows  <- dm$fold == j

  ## ======
  ## fit models to training data
  ## ======

  ## logistic regression with no regularization using only NDVI100
  m1 = glm(veg ~ NDVI100, data = dm[train.rows, 1:10], family = "binomial")

  ## logistic regression with no regularization using all band values
  m2 = glm(veg ~ ., data = dm[train.rows,1:10], family = "binomial")

  ## logistic regression with ridge regularization using all band values
  r.train <- cv.glmnet(x = x[train.rows,], y = dm$veg[train.rows], family = 'binomial', alpha = 0)

  ## logistic regression with lasso regularization using all band values
  las.train <- cv.glmnet(x = x[train.rows,], y = dm$veg[train.rows], family = 'binomial', alpha = 1)

  ## ======
  ## make predictions
  ## ======

  dm$m1pred[test.rows] = predict(m1, newdata=dm[test.rows,], type='response')
  dm$m2pred[test.rows] = predict(m2, newdata=dm[test.rows,], type='response')
  dm$r.1se[test.rows] = predict(r.train, newx=x[test.rows,], s='lambda.1se', type='response')
  dm$r.min[test.rows] = predict(r.train, newx=x[test.rows,], s='lambda.min', type='response')
  dm$las.1se[test.rows] = predict(las.train, newx=x[test.rows,], s='lambda.1se', type='response')
  dm$las.min[test.rows] = predict(las.train, newx=x[test.rows,], s='lambda.min', type='response')

  ## Test logloss for each fold
  metrics[j,'llr1se'] = -mean(dm$veg[test.rows]*
                              log(dm$r.1se[test.rows]) +
                              (1-dm$veg[test.rows])*
                              log(1-dm$r.1se[test.rows]))
  metrics[j,'llrmin'] = -mean(dm$veg[test.rows]*
                              log(dm$r.min[test.rows]) +
                              (1-dm$veg[test.rows])*
                              log(1-dm$r.min[test.rows]))
  metrics[j,'lllas1se'] = -mean(dm$veg[test.rows]*
                              log(dm$las.1se[test.rows]) +
                              (1-dm$veg[test.rows])*
                              log(1-dm$las.1se[test.rows]))
  metrics[j,'lllasmin'] = -mean(dm$veg[test.rows]*
                              log(dm$las.min[test.rows]) +
                              (1-dm$veg[test.rows])*
```

```
                                  log(1-dm$las.min[test.rows]))
  metrics[j,'llm1'] = -mean(dm$veg[test.rows]*
                                  log(dm$m1pred[test.rows]) +
                                  (1-dm$veg[test.rows])*
                                  log(1-dm$m1pred[test.rows]))
  metrics[j,'llm2'] = -mean(dm$veg[test.rows]*
                                  log(dm$m2pred[test.rows]) +
                                  (1-dm$veg[test.rows])*
                                  log(1-dm$m2pred[test.rows]))

}
```

```
## 1 2 3 4
```

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

```
## 5
```

```
head(dm %>% select('r.1se', 'r.min', 'las.1se', 'las.min', 'm1pred', 'm2pred'))
```

```
## # A tibble: 6 x 6
##    r.1se  r.min  las.1se   las.min   m1pred        m2pred
##    <dbl>  <dbl>    <dbl>     <dbl>    <dbl>         <dbl>
## 1 0.994  0.996  1.00     1.00      1.00     1
## 2 0.989  0.994  1.00     1.00      1.00     1.00
## 3 0.901  0.917  0.959    1.00      0.980    1.00
## 4 0.947  0.960  0.993    0.999     0.996    0.999
## 5 0.0447 0.0313 0.000544 0.000104  0.000384 0.000113
## 6 0.0619 0.0349 0.00173  0.0000470 0.000357 0.00000000166
```

```
metrics
```

```
##   fold    llr1se    llrmin   lllas1se   lllasmin       llm1       llm2
## 1    1 0.1669045 0.1498231 0.09445221 0.09269673 0.09517429 0.11472190
## 2    2 0.1395895 0.1235677 0.06310080 0.04475571 0.06630667 0.06393426
## 3    3 0.1757241 0.1656531 0.10224240 0.06953002 0.14931390 0.05261686
## 4    4 0.2606668 0.2392228 0.19783758 0.35470053 0.21519616 0.93120602
## 5    5 0.1665322 0.1576225 0.13905100 0.09379078 0.17305322 0.09587438
```

3. Log Loss

Compute log loss for all models. Compare across models, compare in-sample vs CV log loss, and discuss any notable observations.

```
logloss <- data.frame(m1 = NA,
                      m2 = NA,
                      m3 = NA,
                      m4 = NA,
                      m1out = NA,
                      m2out = NA,
                      r1se = NA,
                      rmin = NA,
                      las1se = NA,
                      lasmin = NA)

# Log loss of every model (in sample)
logloss$m1 <- (-mean(dm$veg*log(dm$predm1) + (1-dm$veg)*log(1-dm$predm1)))
```

```
logloss$m2 <- (-mean(dm$veg*log(dm$predm2) + (1-dm$veg)*log(1-dm$predm2)))
logloss$m3 <- (-mean(dm$veg*log(dm$predm3) + (1-dm$veg)*log(1-dm$predm3)))
logloss$m4 <- (-mean(dm$veg*log(dm$predm4) + (1-dm$veg)*log(1-dm$predm4)))

# Log Loss of every model (out of sample)
logloss$r1se <- (-mean(dm$veg*log(dm$r.1se) + (1-dm$veg)*log(1-dm$r.1se)))
logloss$rmin <- (-mean(dm$veg*log(dm$r.min) + (1-dm$veg)*log(1-dm$r.min)))
logloss$las1se <- (-mean(dm$veg*log(dm$las.1se) + (1-dm$veg)*log(1-dm$las.1se)))
logloss$lasmin <- (-mean(dm$veg*log(dm$las.min) + (1-dm$veg)*log(1-dm$las.min)))
logloss$m1out <- (-mean(dm$veg*log(dm$m1pred) + (1-dm$veg)*log(1-dm$m1pred)))
logloss$m2out <- (-mean(dm$veg*log(dm$m2pred) + (1-dm$veg)*log(1-dm$m2pred)))

logloss <- logloss %>%
  pivot_longer(cols = everything(), names_to = 'model', values_to = 'logloss') %>%
  mutate(sample = ifelse(model %in% c('m1', 'm2', 'm3', 'm4'), 'in', 'out'))

logloss
```

```
## # A tibble: 10 x 3
##    model  logloss sample
##    <chr>    <dbl> <chr>
##  1 m1      0.132  in
##  2 m2      0.0683 in
##  3 m3      0.171  in
##  4 m4      0.119  in
##  5 m1out   0.140  out
##  6 m2out   0.252  out
##  7 r1se    0.182  out
##  8 rmin    0.167  out
##  9 las1se  0.119  out
## 10 lasmin  0.131  out
```

Across in-sample vs. out-sample, the log loss for Ridge is consistently higher than the log loss for Lasso (see m3 vs. m4, r1se vs. las1se, and rmin vs. lasmin).

It seems in-sample models have similar log loss to out-sample models (see m3 vs. r1se, m4 vs. las1se)

Also, it appears that the log loss for the in-sample logistic regression with no regularization has the lowest log loss.

It seems choosing lambda.min is better for both Ridge and Lasso models, as the log loss is lower for lambda.min in both models.

Partitioned matrices

Note that if we have partitioned matrices, or matrices written in block form, we can add and multiply matrices as usual, as if the submatrices were scalars. Consider the following simple example. Suppose the 3×3 matrix A given in block form as

$$\underset{3\times3}{A} = \begin{pmatrix} \underset{2\times2}{B} & \underset{2\times1}{C} \\ \underset{1\times2}{D} & \underset{1\times1}{E} \end{pmatrix}$$

where B is a 2×2 matrix, C is 2×1, D is 1×2, and E is 1×1. Suppose E is a 3×1 vector in block form

$$\underset{3\times1}{F} = \begin{pmatrix} \underset{2\times1}{G} \\ \underset{1\times1}{H} \end{pmatrix}$$

where G is 2×1 and H is 1×1.

For the following questions, you can write out your work for each question by hand and scan it, or write it up in LaTex in this document.

4. Block Transpose

Show that the transpose of A, denoted A^T, is given by

$$\underset{3\times3}{A}^T = \begin{pmatrix} \underset{2\times2}{B}^T & \underset{2\times1}{D}^T \\ \underset{1\times2}{C}^T & \underset{1\times1}{E}^T \end{pmatrix}$$

(Write out $B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$, $C = \begin{pmatrix} c_{11} \\ c_{21} \end{pmatrix}$, etc., and take the transpose.)

$$B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \rightarrow B^T = \begin{pmatrix} b_{11} & b_{21} \\ b_{12} & b_{22} \end{pmatrix}$$

$$C = \begin{pmatrix} c_{11} \\ c_{21} \end{pmatrix} \rightarrow C^T = \begin{pmatrix} c_{11} & c_{21} \end{pmatrix}$$

$$D = \begin{pmatrix} d_{11} & d_{12} \end{pmatrix} \rightarrow D^T = \begin{pmatrix} d_{11} \\ d_{12} \end{pmatrix}$$

$$E = \begin{pmatrix} e_{11} \end{pmatrix} \rightarrow E^T = \begin{pmatrix} e_{11} \end{pmatrix}$$

Then, the matrix A is given by:

$$\underset{3\times3}{A} = \begin{pmatrix} \underset{2\times2}{B} & \underset{2\times1}{C} \\ \underset{1\times2}{D} & \underset{1\times1}{E} \end{pmatrix}$$

So, the matrix A^T is given by taking the transpose of the matrices, and the transpose of A, which will turn columns into rows, effectively flipping the position of D and C.

5. Block Multiplication

Show that

$$\underset{3\times 1}{AF} = \begin{pmatrix} \underset{2\times 1}{BG+CH} \\ \underset{1\times 1}{DG+EH} \end{pmatrix}$$

(Again, write out the matrices and multiply.)

For AF to be a 3×1 matrix, A must be a 3×3 matrix, and F must be a 3×1 matrix. \

Now we notice that $BG + CH$ and $DG + EH$ both share G and H. So, consider the following the product of matrices:

$$\underset{3\times 3}{A} = \begin{pmatrix} \underset{2\times 2}{B} & \underset{2\times 1}{C} \\ \underset{1\times 2}{D} & \underset{1\times 1}{E} \end{pmatrix} \quad , \quad \underset{3\times 1}{F} = \begin{pmatrix} \underset{2\times 1}{G} \\ \underset{1\times 1}{H} \end{pmatrix} \rightarrow \underset{3\times 1}{AF} = \begin{pmatrix} \underset{2\times 1}{BG+CH} \\ \underset{1\times 1}{DG+EH} \end{pmatrix}$$

For BG to be 2×1 matrix, then B must be 2×2 and G must be 2×1. Also, C must be 2×1 since we know that H is 1×1 (otherwise F can't be 3×1).

Similarly, for DG to be 1×1, then D must be 1×2, since we know that G is 2×1. Also, E must be 1×1 since H is 1×1. Note that adding a 1×1 matrix with a 1×1 matrix results in a 1×1 matrix

6. Ridge and OLS with Augmented Data

Using the previous two results, show ridge regression is like OLS but with augmented data. Specifically, consider

$$\underset{(n+p)\times p}{\tilde{X}} = \begin{pmatrix} \underset{n\times p}{X} \\ \sqrt{\lambda}\underset{p\times p}{I} \end{pmatrix}$$

where $\underset{p\times p}{I}$ is a $p \times p$ identity matrix, and consider

$$\underset{(n+p)\times 1}{\tilde{y}} = \begin{pmatrix} \underset{n\times 1}{y} \\ \underset{p\times 1}{0} \end{pmatrix}$$

where 0 is a $p \times 1$ vector of all zeros. So $\tilde{X}$ is like X that has been augmented with some extra rows, and $\tilde{y}$ is like y that has been augmented with some extra zeros. Show that, for a given λ, the OLS estimates of $\tilde{y} = \tilde{X}\beta$ are the same as the ridge estimates of $y = X\beta$. (Start by writing $\tilde{X}^T\tilde{X}$ and $\tilde{X}^T\tilde{y}$.)

The OLS solution to $\tilde{y} = \tilde{X}\beta$ is

$$(\tilde{X}^T\tilde{X})\beta = \tilde{X}^T\tilde{y}$$

where

$$\underset{n\times(n+p)}{\tilde{X}^T} = \begin{pmatrix} \underset{n\times p}{X} & \sqrt{\lambda}\underset{p\times p}{I} \end{pmatrix}$$

so

$$\underset{n\times p}{\tilde{X}^T}\tilde{X} = \begin{pmatrix} \underset{n\times p}{X} & \sqrt{\lambda}\,\underset{p\times p}{I} \end{pmatrix} \times \begin{pmatrix} \underset{n\times p}{X} \\ \sqrt{\lambda}\,\underset{p\times p}{I} \end{pmatrix} = \left(X^T X + \lambda I\right)$$

Also,

$$\underset{n\times 1}{\tilde{X}^T\tilde{y}} = \begin{pmatrix} \underset{n\times p}{X} & \sqrt{\lambda}\,\underset{p\times p}{I} \end{pmatrix} \times \begin{pmatrix} \underset{n\times 1}{y} \\ \underset{p\times 1}{0} \end{pmatrix} = X^T y$$

Thus, the OLS solution to $\tilde{y} = \tilde{X}\beta$ can be re-written as $(X^T X + \lambda I)\beta = X^T y$, which is exactly the same as the ridge estimates of $y = X\beta$

NBA Games

Let's load and prep the data.

```
d = readRDS('data/games.rds')
tms = read.csv('data/nba.teams.csv')

d = d %>%
  filter(lg=='nba', season %in% 2022, season.type=='reg') %>%
  dplyr::select(date, away, home, ascore, hscore, season, gid)


da = d %>% dplyr::select(date, away, ascore, home, hscore, season, gid) %>% mutate(ha = 'away')
dh = d %>% dplyr::select(date, home, hscore, away, ascore, season, gid) %>% mutate(ha = 'home')
colnames(da) = c('date', 'team', 'score',  'opp', 'opp.score', 'season', 'gid', 'ha')
colnames(dh) = c('date', 'team', 'score',  'opp', 'opp.score', 'season', 'gid', 'ha')
d = bind_rows(da, dh) %>%
  arrange(date, gid) %>%
  left_join(tms %>% dplyr::select(team, div),
            by = c('team'='team')) %>%
  left_join(tms %>% dplyr::select(team, div),
            by = c('opp'='team'), suffix=c('.team', '.opp'))
head(d)
```

```
##         date team score opp opp.score season      gid   ha  div.team   div.opp
## 1 2021-10-19  BKN   104 MIL       127   2022 22100001 away  atlantic   central
## 2 2021-10-19  MIL   127 BKN       104   2022 22100001 home   central  atlantic
## 3 2021-10-19  GSW   121 LAL       114   2022 22100002 away   pacific   pacific
## 4 2021-10-19  LAL   114 GSW       121   2022 22100002 home   pacific   pacific
## 5 2021-10-20  IND   122 CHA       123   2022 22100003 away   central southeast
## 6 2021-10-20  CHA   123 IND       122   2022 22100003 home southeast   central
```

Here is a function for extracting coefficients that we used in class.

```
extract.ranef = function(lmer.model=NULL, lm.model=NULL){

  vars = names(ranef(lmer.model))
  lmer.coefs = NULL
  lm.coefs = NULL

  if(!is.null(lm.model)){
    lm.coefs = summary(lm.model)$coefficients %>%
      as.data.frame() %>%
      rownames_to_column(var='var') %>%
      filter(grepl(paste0(vars, collapse="|"), var)) %>%
      rename(est='Estimate',
             se ='Std. Error') %>%
      dplyr::select(var, est, se) %>%
      mutate(model='glm')
  }

  for (j in vars){

    ## lmer
    est = ranef(lmer.model)[[j]]
    se  = se.ranef(lmer.model)[[j]]
```

```
    colnames(est) = 'est'
    colnames(se) = 'se'
    temp = data.frame(var = rownames(est),
                      est=est,
                      se=se,
                      model='glmer',
                      ranef = j)
    lmer.coefs = rbind(lmer.coefs, temp)

    ## lm
    if(!is.null(lm.model)){
      rows=grepl(j, lm.coefs$var)
      lm.coefs$var[rows] = lm.model$xlevels[[j]][-length(lm.model$xlevels[[j]])]
      lm.coefs$ranef[rows] = j

    } ## if lm.model

  } ## end j loop

  coefs = rbind(lmer.coefs, lm.coefs)
  return(coefs)
}
```

7. Random effects for team and opp

Previously we used `ha`, `team` and `opp` to predict `score`. Fit a linear regression model with outcome `score` and predictors `ha`, `team`, and `opp`, as well as a similar mixed effects linear regression model with both `team` and `opp` as random effects terms. Create visualizations that help you compare the coefficients obtained from the two models. Discuss any observations.

```
## linear regression model
lm1 <- lm(score ~ ha + team + opp, data = d, contrasts = list(team = 'contr.sum',
                                                              opp = 'contr.sum'))

## mixed effects linear regression model
lmer1 <- lmer(score ~ ha + (1|team) + (1|opp), data = d)
```

```
# extract coefficients
coefs = extract.ranef(lmer1, lm1)
head(coefs)
```

```
##     var       est       se model ranef
## ATL ATL  3.083996 1.263538 glmer  team
## BKN BKN  2.276941 1.263538 glmer  team
## BOS BOS  0.808452 1.263538 glmer  team
## CHA CHA  4.373273 1.263538 glmer  team
## CHI CHI  1.050838 1.263537 glmer  team
## CLE CLE -2.640037 1.263538 glmer  team
```

To see how many observations we have per team:

```
n.obs.team = d %>%
  group_by(team) %>%
  summarise(n = n(),
            value = mean(score)) %>%
  rename(var = team) %>%
```

```
  mutate(ranef = 'team')
n.obs.team %>% head()
```

```
## # A tibble: 6 x 4
##   var       n value ranef
##   <chr> <int> <dbl> <chr>
## 1 ATL      82  114. team
## 2 BKN      82  113. team
## 3 BOS      82  112. team
## 4 CHA      82  115. team
## 5 CHI      82  112. team
## 6 CLE      82  108. team
```

```
n.obs.opp = d %>%
  group_by(opp) %>%
  summarise(n = n(),
            value = mean(score)) %>%
  rename(var = opp) %>%
  mutate(ranef = 'opp')
n.obs.opp %>% head()
```

```
## # A tibble: 6 x 4
##   var       n value ranef
##   <chr> <int> <dbl> <chr>
## 1 ATL      82  112. opp
## 2 BKN      82  112. opp
## 3 BOS      82  104. opp
## 4 CHA      82  115. opp
## 5 CHI      82  112  opp
## 6 CLE      82  106. opp
```

We see that every team had 82 observations since they all played in the season.

```
n.obs = rbind(n.obs.team, n.obs.opp)
df = coefs %>%
  pivot_wider(names_from  = model,
              values_from = c(est, se),
              names_sep   = '.') %>%
  left_join(n.obs,
            by = c('var', 'ranef'))
head(df)
```

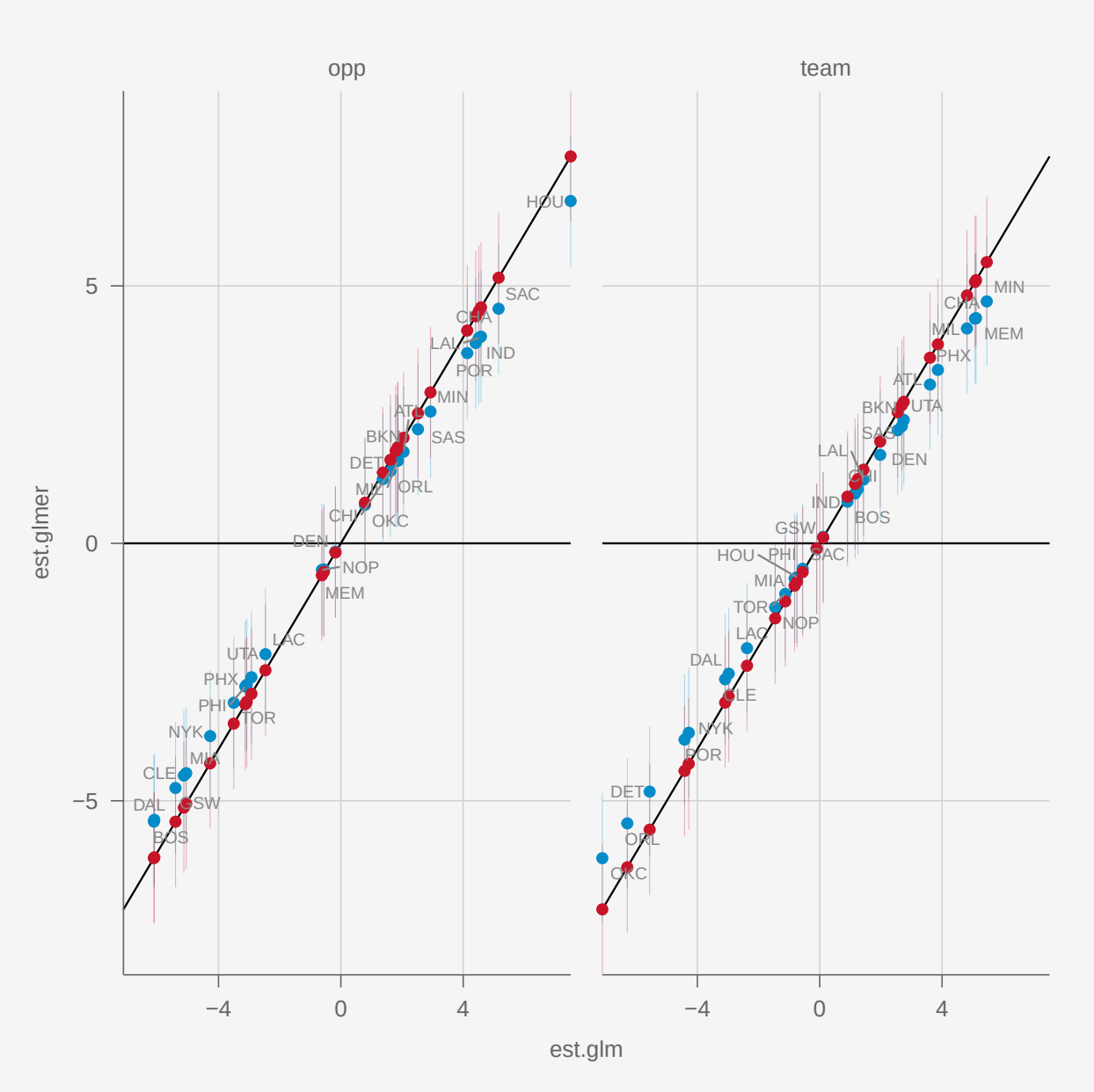
```
## # A tibble: 6 x 8
##   var   ranef est.glmer est.glm se.glmer se.glm     n value
##   <chr> <chr>     <dbl>   <dbl>    <dbl>  <dbl> <int> <dbl>
## 1 ATL   team      3.08    3.60      1.26   1.27    82  114.
## 2 BKN   team      2.28    2.68      1.26   1.27    82  113.
## 3 BOS   team      0.808   0.906     1.26   1.27    82  112.
## 4 CHA   team      4.37    5.11      1.26   1.27    82  115.
## 5 CHI   team      1.05    1.25      1.26   1.27    82  112.
## 6 CLE   team     -2.64   -3.09      1.26   1.27    82  108.
```

```
tail(df)
```

```
## # A tibble: 6 x 8
##   var   ranef est.glmer est.glm se.glmer se.glm     n value
##   <chr> <chr>     <dbl>   <dbl>    <dbl>  <dbl> <int> <dbl>
```

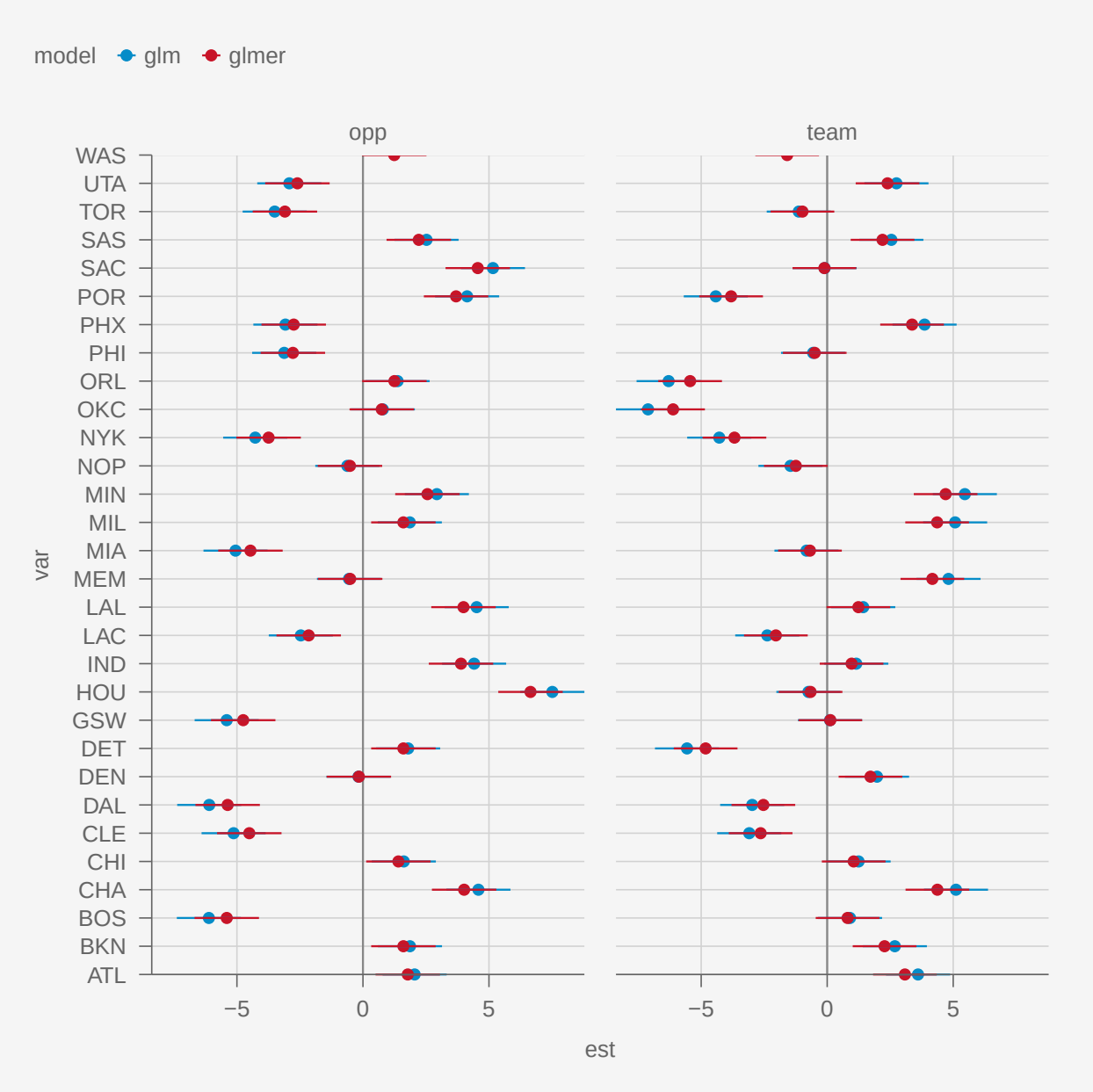
```
## 1 POR   opp        3.69    4.13     1.28   1.27    82  115.
## 2 SAC   opp        4.56    5.16     1.28   1.27    82  116.
## 3 SAS   opp        2.21    2.52     1.28   1.27    82  113.
## 4 TOR   opp       -3.10   -3.50     1.28   1.27    82  107.
## 5 UTA   opp       -2.60   -2.92     1.28   1.27    82  108.
## 6 WAS   opp        1.24   NA        1.28  NA       82  112
```

```r
dg2 = coefs %>%
  arrange(model, est) %>%
  mutate(var = factor(var, levels = unique(var))) %>%
  left_join(n.obs,
            by = c('var', 'ranef'))

g <- ggplot(df,
       aes(x = est.glm,
           y = est.glmer,
           color = ranef,
           label = var)) +
  geom_abline(slope = 1,
              intercept = 0) +
  geom_hline(yintercept = 0) +
  geom_segment(aes(xend = est.glm,
                   y = est.glmer + se.glmer,
                   yend = est.glmer - se.glmer),
               color = pubblue,
               linewidth = 0.25,
               alpha = 0.3)+
  geom_segment(aes(xend = est.glm,
                   y = est.glm + se.glm,
                   yend = est.glm - se.glm),
               color = pubred,
               linewidth = 0.25,
               alpha = 0.3) +
  geom_point(color = pubblue) +
  geom_point(aes(y = est.glm),
             color = pubred) +
  geom_text_repel(hjust = -0.2,
                  size = 3,
                  color = pubmediumgray) +
  facet_wrap(~ranef, nrow = 1) +
  coord_cartesian(clip='off', expand=F)

g %>% pub()
```




```
gg = ggplot(dg2,
       aes(y = var,
           color = model))+
  geom_vline(xintercept = 0,
             color = pubmediumgray)+
  geom_point(aes(x = est)) +
  facet_wrap(~ranef) +
  geom_segment(aes(x     = est + se,
                   xend  = est - se,
                   y     = var,
                   yend  = var),
               linewidth = 0.5)
```

```
gg |> pub()
```



We see that the coefficients are similar between models for both opp and team. It seems the coefficients for the GLMER are slightly lower, meaning they are pulled to 0 due to regularization. The number of observations does not play a role here because each team is involved in exactly 82 games in a regular season.

8. Cross-validation

Perform cross-validation, compare the performance of the two models, discuss the results, and state which model you prefer and why.

```
# for reproducibility
set.seed(123)

# set number of folds (usually 5 or 10)
```

```r
k=5

# create a vector of fold numbers, repeating 1:k until reaching the number of rows in the data
folds = rep(1:k,
            length.out = nrow(d))

# create a new column in the data frame that randomly assigns a fold to each row without replacement
# (so that we don't have a row assigned to two or more folds)
d$fold = sample(folds,
                nrow(d),
                replace = F)

# view the first few rows of the data frame to see the fold assignments
head(d)
```

```
##         date team score opp opp.score season      gid   ha  div.team   div.opp
## 1 2021-10-19  BKN   104 MIL       127   2022 22100001 away  atlantic   central
## 2 2021-10-19  MIL   127 BKN       104   2022 22100001 home   central  atlantic
## 3 2021-10-19  GSW   121 LAL       114   2022 22100002 away   pacific   pacific
## 4 2021-10-19  LAL   114 GSW       121   2022 22100002 home   pacific   pacific
## 5 2021-10-20  IND   122 CHA       123   2022 22100003 away   central southeast
## 6 2021-10-20  CHA   123 IND       122   2022 22100003 home southeast   central
##   fold
## 1    2
## 2    1
## 3    5
## 4    2
## 5    2
## 6    3
```

```r
d <- d %>% filter(!is.na(score))

#initialize columns
d[,c('lm', 'lmer')] = NA

# loop through the folds
for (j in 1:k){
  cat(j,'') ## print out the progress

  ## train rows are the ones not in the j-th fold
  train.rows = d$fold != j

  # test rows are the observations in the j-th fold
  test.rows  = d$fold == j

  ## fit model on training data
  ## linear regression model
  lm1 <- lm(score ~ ha + team + opp, data = d[train.rows,])

  ## mixed effects linear regression model
  lmer1 <- lmer(score ~ ha + (1|team) + (1|opp), data = d[train.rows,])

  ## make predictions
  d$lm[test.rows] = predict(lm1, newdata=d[test.rows,], type='response')
```

```
  d$lmer[test.rows] = predict(lmer1, newdata=d[test.rows,], type='response')
}
```

```
## 1 2 3 4 5
```

To calculate the RMSE for the models:

```
sqrt(mean((d$lm    - d$score)^2, na.rm=T))
```

```
## [1] 11.84325
```

```
sqrt(mean((d$lmer  - d$score)^2, na.rm=T))
```

```
## [1] 11.82224
```

The RMSE are almost identical. Also, since the coefficients are similar between models for both opp and team, then I would choose the model with lower RMSE, which in this case is the GLMER model.

EV Charging Stations

Let's load and clean our charging station data.

```
dc = readRDS('data/tracts.and.census.with.EV.stations.rds')
dc = dc@data

## create a indicator for whether there is at least
## one lev2 station in each tract
dc = dc %>%
  mutate(l2 = ifelse(lev2 > 0 , 1,  0),
         l2 = ifelse(is.na(l2), 0, l2))
```

If you have trouble answering these questions with the full data, use this line of code to take a random sample of 5,000 tracts. Doing this will eliminate any memory issues you might have.

```
dc = dc[sample(1:nrow(dc),
               5000,
               replace = F),]
```

9. County as predictor

There are 866 counties in the US, so using county as a predictor adds 865 columns to a logistic regression model, which may not be ideal. Build a logistic regression model with `log(house.value)` and `county` as predictors, and a mixed effects logistic regression model with `log(house.value)` a random intercept for `county`. Create visualizations that help you compare the coefficients. Discuss any observations.

Since we want a logistic regression, then our variable must be binary. Thus, we will be predicting the `l2` variable, which indicates whether a tract has lev2 or lev3 charging stations.

```
lm <- lm(l2 ~ log(house.value) + county, data = dc, family = binomial, contrasts = list(county = 'contr
lmer <- glmer(l2 ~ log(house.value) + (1|county), data = dc, family = binomial)
```

We now extract the coefficients:

```
coeffs = extract.ranef(lmer, lm)
head(coeffs)
```

```
##                          var         est        se model  ranef
## Acadia Parish   Acadia Parish -0.01349410 0.2756148 glmer county
## Ada County         Ada County  0.08484561 0.2637318 glmer county
```

```
## Adams County     Adams County -0.03612196 0.2586735 glmer county
## Addison County Addison County -0.01744521 0.2752938 glmer county
## Alachua County Alachua County -0.03053935 0.2738201 glmer county
## Alameda County Alameda County -0.06425761 0.2261954 glmer county
```

```
n.obs.county = dc %>%
  group_by(county) %>%
  summarise(n = n(),
            value = mean(12)) %>%
  rename(var = county) %>%
  mutate(ranef = 'county')
n.obs.county %>% head()
```

```
## # A tibble: 6 x 4
##   var                n value ranef
##   <chr>          <int> <dbl> <chr>
## 1 Acadia Parish      1 0     county
## 2 Ada County         7 0.429 county
## 3 Adams County      13 0.154 county
## 4 Addison County     1 0     county
## 5 Alachua County     2 0     county
## 6 Alameda County    28 0.357 county
```

```
dff = coeffs %>%
  pivot_wider(names_from  = model,
              values_from = c(est, se),
              names_sep   = '.') %>%
  left_join(n.obs.county,
            by = c('var', 'ranef'))

head(dff)
```

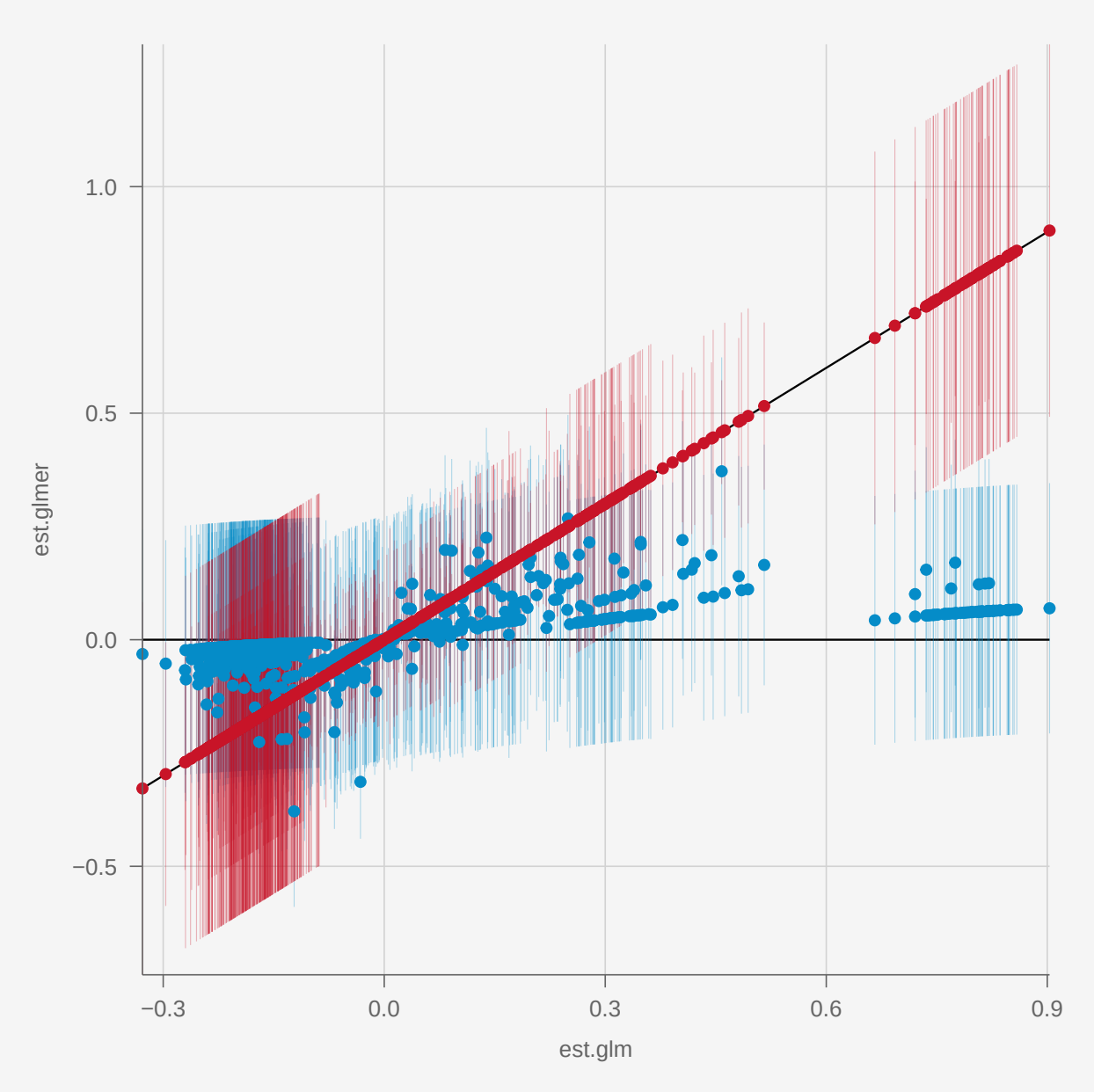
```
## # A tibble: 6 x 8
##   var            ranef  est.glmer est.glm se.glmer se.glm     n value
##   <chr>          <chr>      <dbl>   <dbl>    <dbl>  <dbl> <int> <dbl>
## 1 Acadia Parish  county   -0.0135 -0.187     0.276 0.411      1 0
## 2 Ada County     county    0.0848  0.190     0.264 0.156      7 0.429
## 3 Adams County   county   -0.0361 -0.0394    0.259 0.114     13 0.154
## 4 Addison County county   -0.0174 -0.225     0.275 0.411      1 0
## 5 Alachua County county   -0.0305 -0.207     0.274 0.291      2 0
## 6 Alameda County county   -0.0643  0.0377    0.226 0.0805    28 0.357
```

```
tail(dff)
```

```
## # A tibble: 6 x 8
##   var                    ranef  est.glmer est.glm se.glmer se.glm     n value
##   <chr>                  <chr>      <dbl>   <dbl>    <dbl>  <dbl> <int> <dbl>
## 1 Yell County            county  -0.00959  -0.140    0.276  0.411      1 0
## 2 Yellow Medicine County county  -0.0105   -0.153    0.276  0.411      1 0
## 3 Yellowstone County     county   0.0378    0.265    0.273  0.291      2 0.5
## 4 Yolo County            county   0.101     0.721    0.272  0.291      2 1
## 5 York County            county   0.0951    0.106    0.257  0.114     13 0.308
## 6 Yuba County            county   0.0616   NA        0.275 NA          1 1
```

We see that counties have very different number of observations, which might impact the model.

```r
dgg2 = coeffs %>%
  filter(ranef == 'county') %>%
  arrange(model, est) %>%
  mutate(var = factor(var, levels = unique(var))) %>%
  left_join(n.obs,
            by = c('var', 'ranef'))

g2 = ggplot(dff %>%
              filter(ranef == 'county'),
       aes(x = est.glm,
           y = est.glmer,
           label = var))+
  geom_abline(slope = 1,
              intercept = 0) +
  geom_hline(yintercept = 0) +
  geom_point(color = pubdarkgray) +
  geom_text_repel(hjust = -.2,
                  color = pubmediumgray) +
  labs(title = 'County Coefficients',
       x = 'GLM',
       y = 'GLMER')

g2 <- ggplot(dff,
       aes(x = est.glm,
           y = est.glmer,
           color = ranef,
           label = var)) +
  geom_abline(slope = 1,
              intercept = 0) +
  geom_hline(yintercept = 0) +
  geom_segment(aes(xend = est.glm,
                   y = est.glmer + se.glmer,
                   yend = est.glmer - se.glmer),
               color = pubblue,
               linewidth = 0.25,
               alpha = 0.3)+
  geom_segment(aes(xend = est.glm,
                   y = est.glm + se.glm,
                   yend = est.glm - se.glm),
               color = pubred,
               linewidth = 0.25,
               alpha = 0.3) +
  geom_point(color = pubblue) +
  geom_point(aes(y = est.glm),
             color = pubred) +
  coord_cartesian(clip='off', expand=F)

g2 %>% pub()
```



It seems all the coefficients get pulled to 0. This is most likely because most counties only have a few observations, while others have many more observations.

10. Cross-validation

Perform cross-validation, compare the performance of the two models, discuss the results, and state which model you prefer and why.

```
dcc <- dc |> select(l2, county, house.value)

# for reproducibility
set.seed(123)

# set number of folds (usually 5 or 10)
k=5
```

```r
# create a vector of fold numbers, repeating 1:k until reaching the number of rows in the data
folds = rep(1:k,
            length.out = nrow(dcc))

# create a new column in the data frame that randomly assigns a fold to each row without replacement
# (so that we don't have a row assigned to two or more folds)
dcc$fold = sample(folds,
                  nrow(dcc),
                  replace = F)

# view the first few rows of the data frame to see the fold assignments
head(dcc)
```

```
##       l2                county house.value fold
## 8288   0 San Bernardino County      161500    3
## 44673  0              New York     2000001    1
## 53967  0          Grady County      120900    2
## 52914  0       Franklin County       60300    1
## 22906  0         Grundy County      192500    1
## 69435  1         Kitsap County      218800    1
```

```r
metrics <- data.frame(fold = 1:k,
                      logloss1 = NA,
                      logloss2 = NA)

dcc[,c('lm', 'lmer')] = NA

dcc <- dcc %>% filter(county != "Kodiak Island Borough",
                      county != "Tuolumne County",
                      county != "Albemarle County")

for (j in 1:k){
  cat(j,'')

  train.rows = dcc$fold != j
  test.rows  = dcc$fold == j

  # the models are already defined in question 9, so only need to predict
  dcc$lm[test.rows] = predict(lm, newdata=dcc[test.rows,], type='response')
  dcc$lmer[test.rows] = predict(lmer, newdata=dcc[test.rows,], type='response')
}
```

```
## 1 2 3 4 5
```

```r
sqrt(mean((dcc$lm    - dcc$l2)^2, na.rm=T))
```

```
## [1] 0.3729722
```

```r
sqrt(mean((dcc$lmer  - dcc$l2)^2, na.rm=T))
```

```
## [1] 0.404599
```

```r
roc.glm = roc(dcc$l2, dcc$lm)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
roc.glmer = roc(dcc$l2, dcc$lmer)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
roc.glm$auc
```

```
## Area under the curve: 0.7899
```

```
roc.glmer$auc
```

```
## Area under the curve: 0.6781
```

Since the RMSE for the non-mixed effects model is lower, we prefer that one. Also, the AUC for the non-mixed effects model is higher, which is another reason to prefer it.