Classification

Part 3

Prof. Bisbee

Vanderbilt University

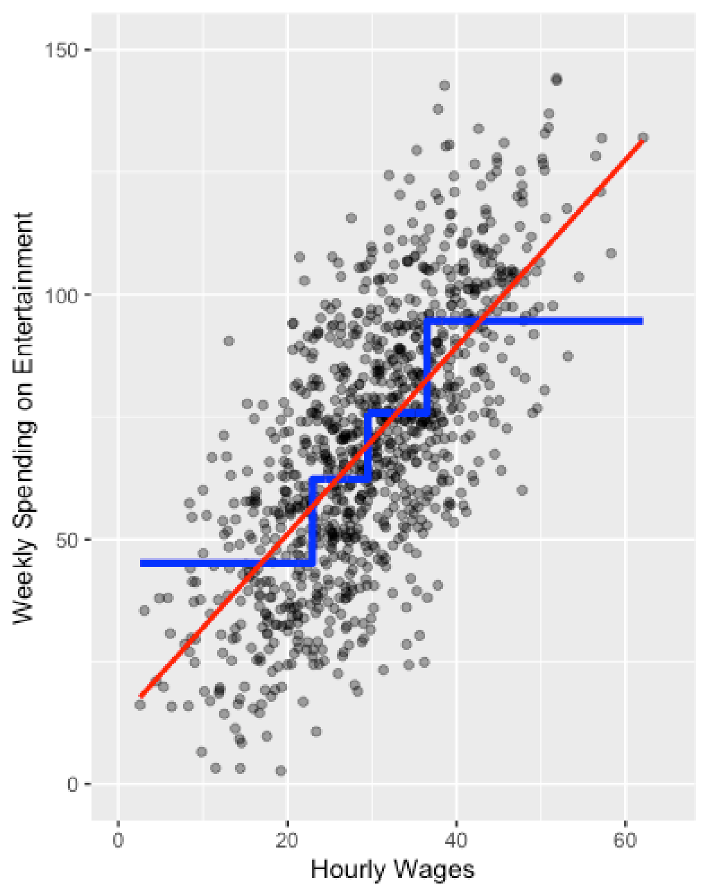
Slides Updated: 2024-07-11

Agenda

1. Recap of regression and classification
2. Introducing (some) machine learning algorithms

What is regression?

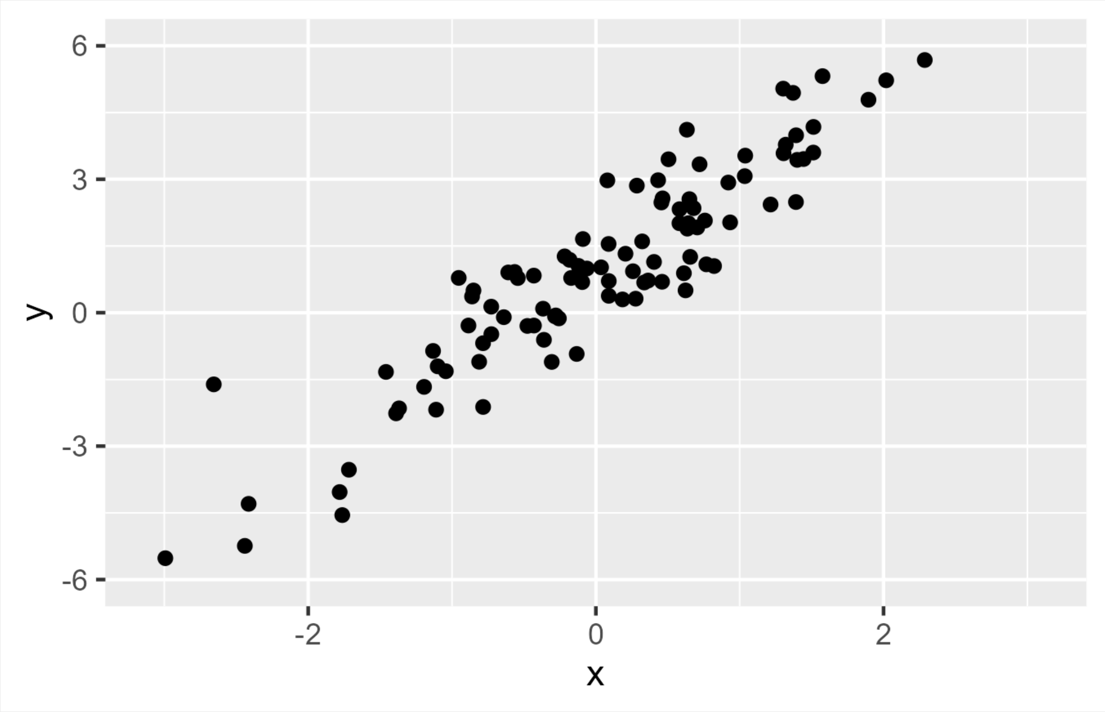
- Conditional means for continuous data



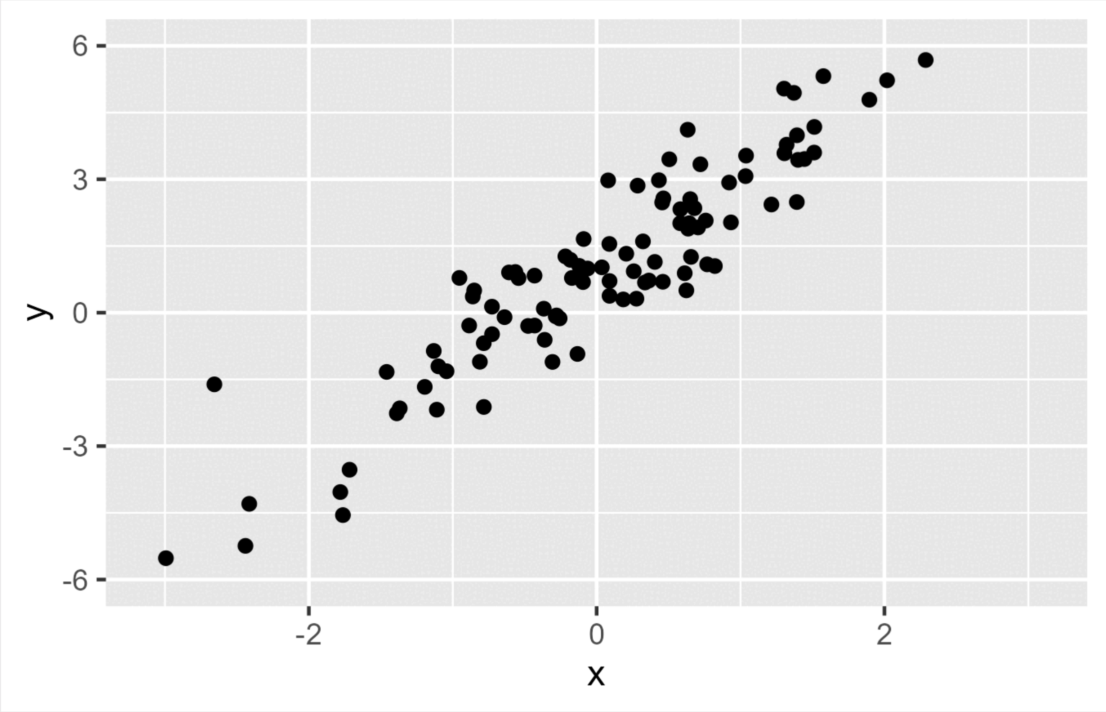
Regression

- Calculating a **line** that minimizes mistakes *for every observation*
  - NB: could be a curvey line! For now, just assume straight
- Recall from geometry how to graph a straight line
- $Y = a + bX$
  - a: the "intercept" (where the line intercepts the y-axis)
  - b: the "slope" (how much Y changes for each increase in X)
- (Data scientists use α and β instead of a and b b/c nerds)
- Regression analysis simply chooses the best line
  - "Best"?
  - The line that minimizes the mistakes (the **line of best fit**)

Visual Intuition



Visual Intuition

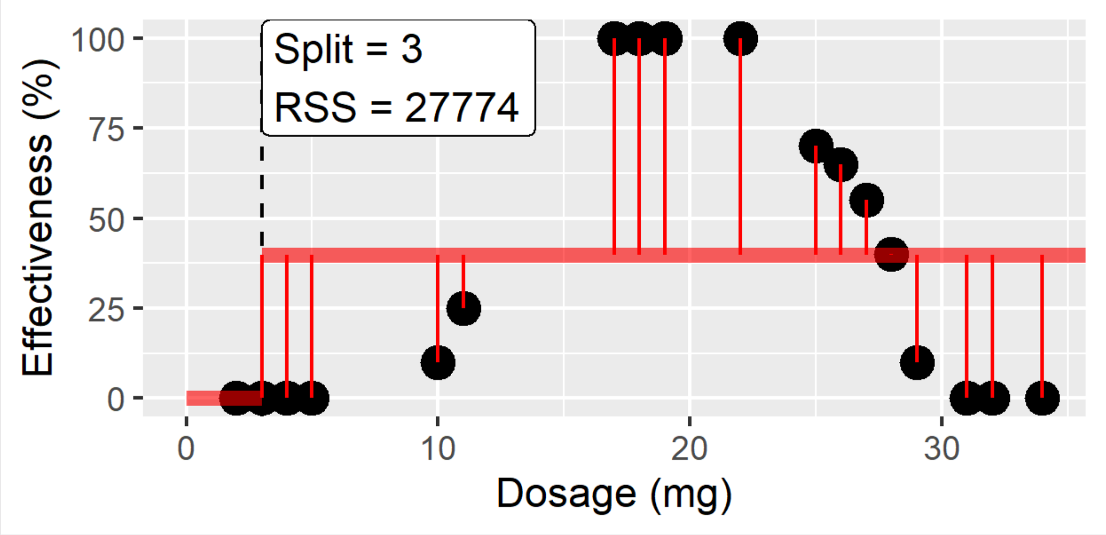


Two Camps Revisited

- Regression is great for **theory testing**
  - Results tell us something **meaningful** about our theory
- But if all we care about is **prediction**...?
  - Want to test every possible predictor (and combinations)
  - Don't care about **relationships**
  - Just care about **accuracy**
- Algorithms can save us time!
  - Random Forests
  - LASSO

Random Forests

- Identify the best "partition" (split) that divides the data



- In `R`: `ranger`
  - `formula = Y ~ .`

Random Forests

```
require(tidyverse)
require(scales)
require(tidymodels)
fn <-
read_rds('https://github.com/jbisbee1/ISP_Data_Science_2024/raw/main/da
```

Research Question

- What predicts whether you win at Fortnite?

```
form.perf <- 'won ~ hits + assists + accuracy + head_shots +
damage_to_players'

form.games <- 'won ~ eliminations + revives + distance_traveled +
materials_gathered'

form.context <- 'won ~ mental_state + startTime + gameIdSession'

form.full <- 'won ~ hits + assists + accuracy + head_shots +
damage_to_players + eliminations + revives + distance_traveled +
materials_gathered + mental_state + startTime + gameIdSession'
```

Comparing models

```
require(broom)
m.perf <- lm(as.formula(form.perf),fn)
tidy(m.perf)
```

```
## # A tibble: 6 × 5
##   term               estimate std.error statistic  p.value
##   <chr>                 <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)        0.0879   0.0377        2.33  1.99e- 2
## 2 hits               0.000696 0.00100       0.695 4.87e- 1
## 3 assists            0.0345   0.0102        3.38  7.64e- 4
## 4 accuracy          -0.416    0.108        -3.85  1.26e- 4
## 5 head_shots        -0.00481  0.00315      -1.53  1.27e- 1
## 6 damage_to_players  0.000473 0.0000571     8.27  4.31e-16
```

Comparing models

```
m.games <- lm(as.formula(form.games),fn)
tidy(m.games)
```

```
## # A tibble: 5 × 5
##   term                  estimate std.error statistic  p.value
##   <chr>                    <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)            4.17e-2 0.0221       1.89   5.94e- 2
## 2 eliminations           1.15e-2 0.00976      1.18   2.39e- 1
## 3 revives                6.99e-2 0.0181       3.86   1.19e- 4
## 4 distance_traveled      1.81e-4 0.0000175   10.3    1.31e-23
## 5 materials_gathered    -2.55e-6 0.0000351   -0.0725 9.42e- 1
```

Comparing models

```
m.context <- lm(as.formula(form.context),fn)
tidy(m.context)
```

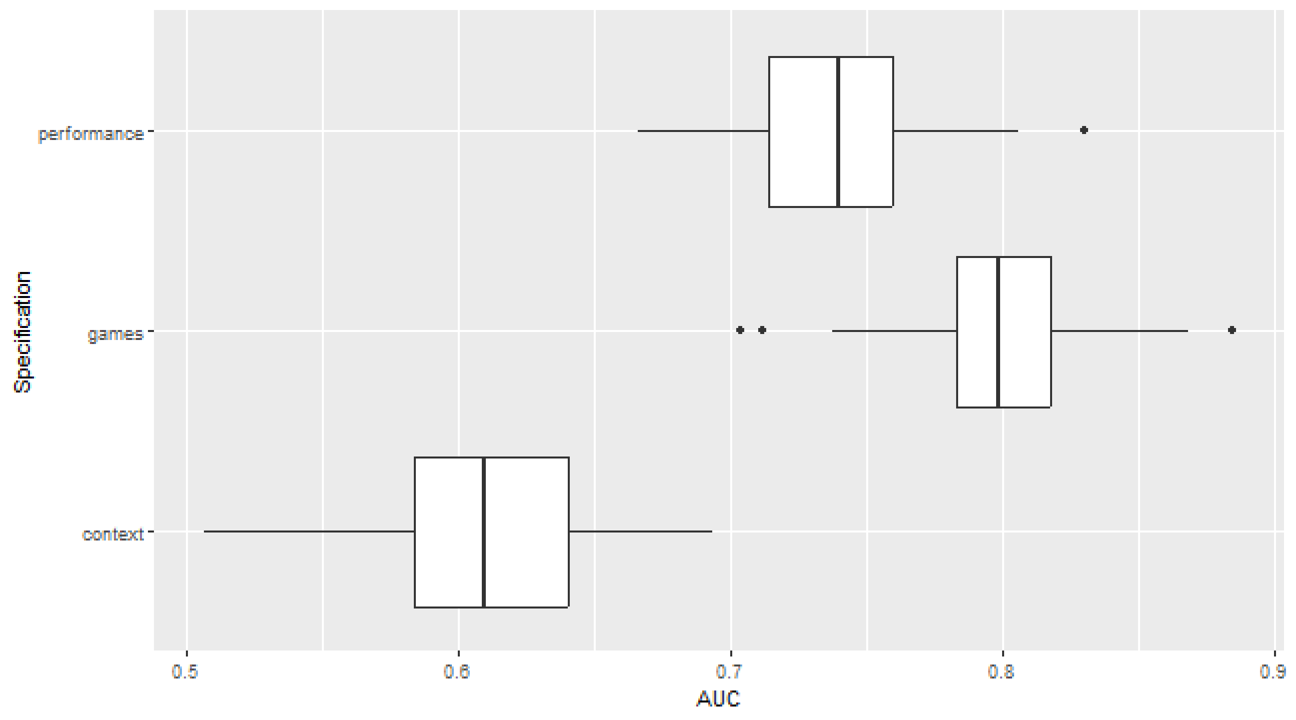
```
## # A tibble: 4 × 5
##   term                     estimate std.error statistic p.value
##   <chr>                       <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)               9.03e+1   2.99e+1     3.02  2.58e-3
## 2 mental_statesober         1.37e-1   2.93e-2     4.66  3.56e-6
## 3 startTime                -5.67e-8   1.88e-8    -3.02  2.64e-3
## 4 gameIdSession             1.46e-3   1.46e-3     0.998 3.19e-1
```

Evaluate Model Fit

```
cvRes <- NULL
for(i in 1:100) {
  inds <- sample(1:nrow(fn),size = round(nrow(fn)*.8),replace = F)
  train <- fn %>% slice(inds)
  test <- fn %>% slice(-inds)

  # Train
  mTmp.perf <- lm(as.formula(form.perf),train)
  mTmp.games <- lm(as.formula(form.games),train)
  mTmp.context <- lm(as.formula(form.context),train)

  # Test
  toEval <- test %>%
    mutate(prob.p = predict(mTmp.perf,newdata = test),
           prob.g = predict(mTmp.games,newdata = test),
           prob.c = predict(mTmp.context,newdata = test),
           truth = factor(won,levels = c('1','0')))

  auc.p <- roc_auc(toEval,truth,prob.p) %>%
    mutate(model = 'performance')

  auc.g <- roc_auc(toEval,truth,prob.g) %>%
    mutate(model = 'games')
```

Evaluate Model Fit

```
cvRes %>%
  ggplot(aes(x = .estimate,y = model)) +
  geom_boxplot() + labs(x = 'AUC',y = 'Specification')
```



Random Forests

```r
require(ranger) # Fast random forests package
rf.f <- ranger(formula = as.formula(form.full),data = fn)

toEval <- fn %>%
  mutate(prob_won = rf.f$predictions) %>%
  mutate(truth = factor(won,levels = c('1','0')))

roc_auc(toEval,truth,prob_won)
```

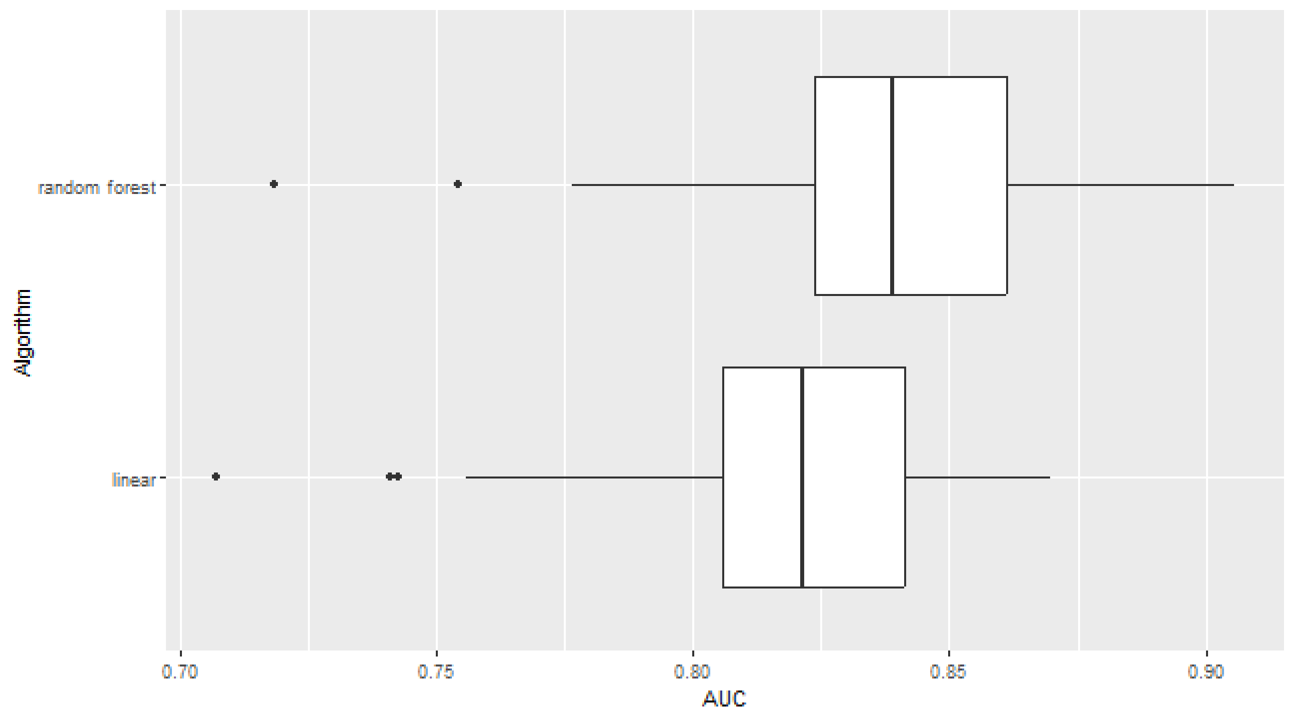
```
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.837
```

Random Forest Comparison

```
cvRes <- NULL
for(i in 1:100) {
  inds <- sample(1:nrow(fn),size = round(nrow(fn)*.8),replace = F)
  train <- fn %>% slice(inds)
  test <- fn %>% slice(-inds)

  # Train
  mLM.f <- lm(as.formula(form.full),train)
  mRF.f <- ranger(as.formula(form.full),train)

  # Test
  # NEED TO RUN PREDICTION ON RF FIRST
  tmpPred <- predict(mRF.f,test)

  toEval <- test %>%
    mutate(prob.lm = predict(mLM.f,newdata = test),
           prob.rf = tmpPred$predictions,
           truth = factor(won,levels = c('1','0')))

  auc.lm <- roc_auc(toEval,truth,prob.lm) %>%
    mutate(model = 'linear')

  auc.rf <- roc_auc(toEval,truth,prob.rf) %>%
```

Random Forest Comparison

```
cvRes %>%
  ggplot(aes(x = .estimate,y = model)) +
  geom_boxplot() + labs(x = 'AUC',y = 'Algorithm')
```



What matters most?

- Random Forests are particularly suitable for investigating **variable importance**
    - I.e., which X predictors are most helpful?
- A few options, but we rely on **permutation tests**
    - Idea: run the best model you have, then re-run it after "permuting" one of the variables
    - "Permute" means randomly reshuffle...breaks relationship
    - How much **worse** is the model when you break a variable?

Variable Importance

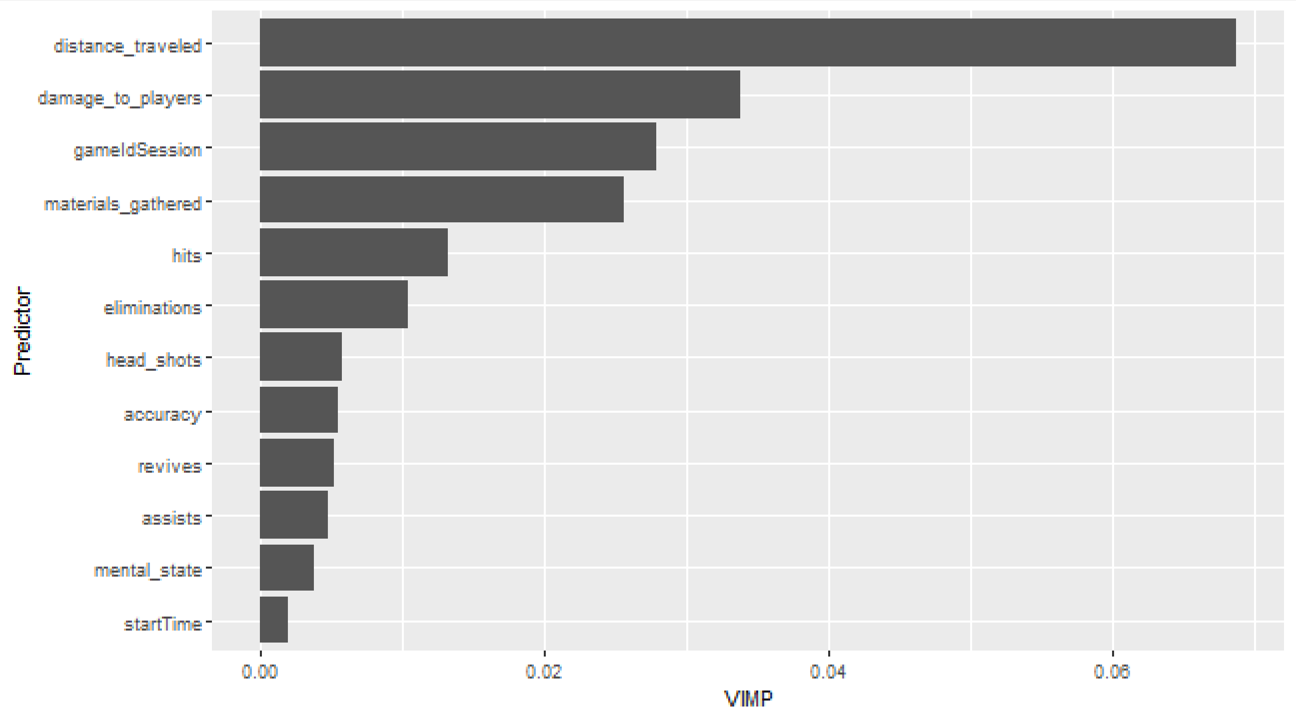
- In `ranger()`, use `importance = "permutation"`

```
rf.full <- ranger(formula = as.formula(form.full),data = fn %>%
                    mutate(mental_state = ifelse(mental_state ==
'sober',1,0)),importance = 'permutation')

rf.full$variable.importance
```

```
##           hits           assists          accuracy
##    0.013803552       0.003892707       0.005061289
##     head_shots damage_to_players      eliminations
##    0.005871168       0.033652154       0.011048596
##        revives distance_traveled materials_gathered
##    0.005001197       0.068280506       0.026299848
##   mental_state         startTime     gameIdSession
##    0.005080621       0.000381819       0.027336278
```

Variable Importance

```
toplot <- data.frame(vimp = rf.full$variable.importance,
                     vars = names(rf.full$variable.importance))

toplot %>%
  ggplot(aes(x = vimp,y = reorder(vars,vimp))) +
  geom_bar(stat = 'identity') + labs(x = 'VIMP',y = 'Predictor')
```



LASSO

- "Least Absolute Shrinkage and Selection Operator"
- Concept: Make it hard for predictors to matter
  - Practice: λ penalizes how many variables you can include
  - $\sum_{i=1}^{n}(y_i - \sum_j x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{p}|\beta_j|$
  - Minimize the errors, but penalize for each additional predictor
  - You *could* kitchen-sink a regression and get super low errors
  - LASSO penalizes you from throwing everything into the kitchen sink
- In `R`, need to install a new package! `install.packages('glmnet')`

```r
require(glmnet)
```

LASSO

- Function doesn't use formulas
- Give it the raw data instead, divided into `Y` (outcome) and `X` (predictors)

```
rhsVars <- str_split(gsub('won ~ ','',form.full),' \\+ ')[[1]]
fn <- fn %>%
  drop_na(all_of(rhsVars))
Y <- fn$won
X <- fn %>% select(all_of(rhsVars)) %>%
  mutate(mental_state = ifelse(mental_state == 'sober',1,0),
         startTime = as.numeric(startTime))
```

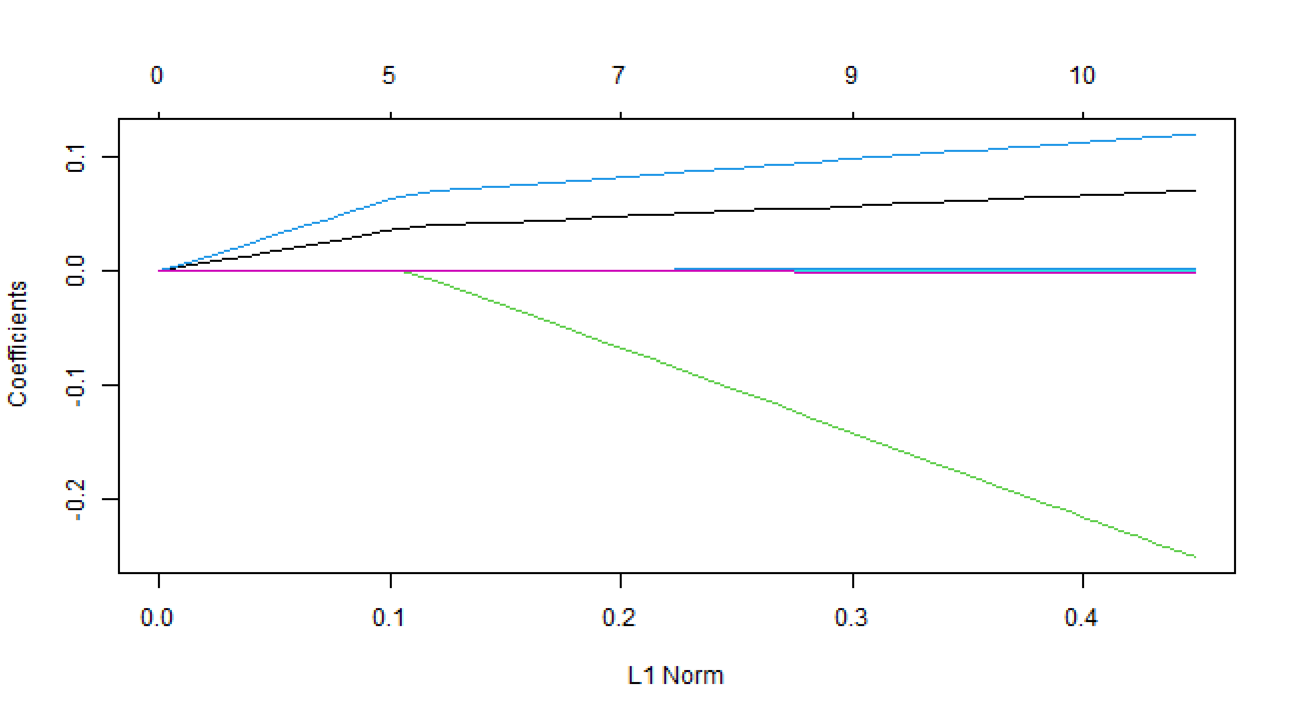
LASSO

- Now estimate!

```
lassFit <- glmnet(x = as.matrix(X),
                  y = as.matrix(Y))
```

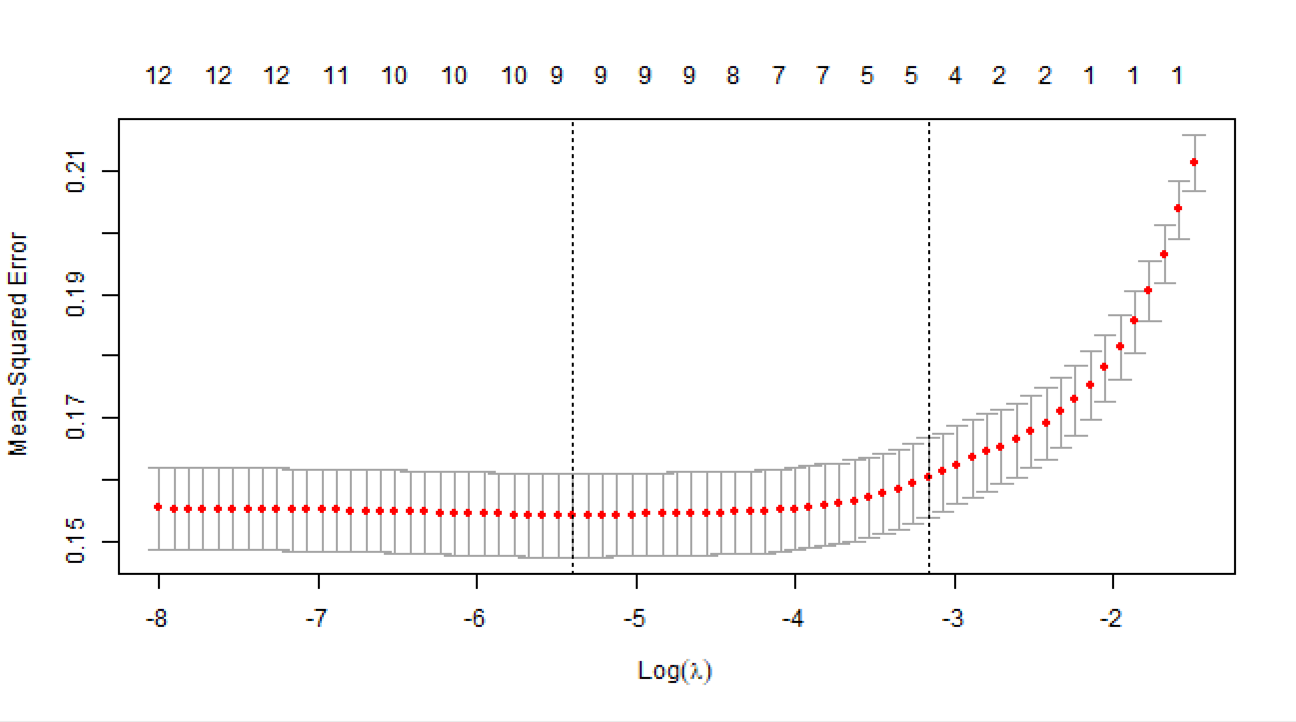
LASSO

```
plot(lassFit)
```



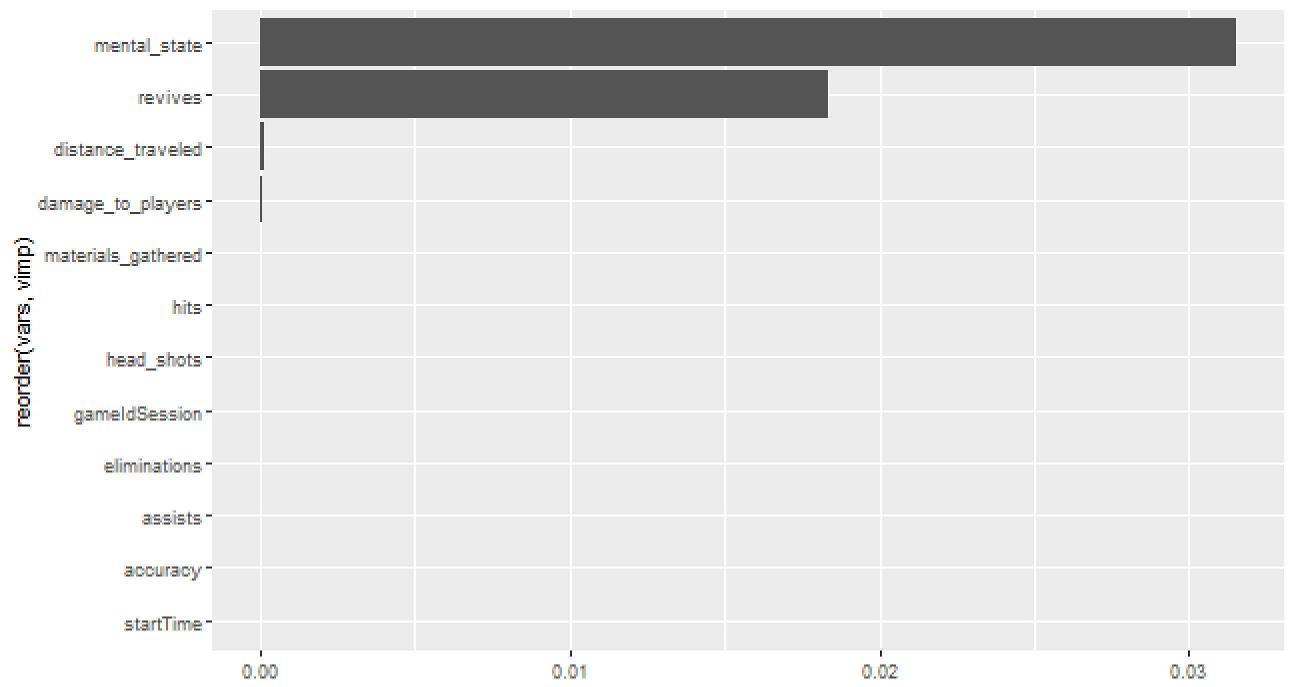
Has its own CV!

```
cv.lassFit <- cv.glmnet(x = as.matrix(X),y = as.matrix(Y))

plot(cv.lassFit)
```



Variable Importance

```
best <- cv.lassFit$glmnet.fit$beta[,cv.lassFit$index[2,]]
vimpLass <- data.frame(vimp = best,
                       vars = names(best))
vimpLass %>%
  ggplot(aes(x = vimp,y = reorder(vars,vimp))) +
  geom_bar(stat = 'identity')
```



Conclusion

- Lots of powerful tools out there!
- Make sure to take **more classes** on these topics!