Lecture 11 Notes

2024-07-21

```
require(tidyverse)
```

```
## Loading required package: tidyverse
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ ggplot2   3.5.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.3     ✓ tidyr     1.3.1
## ✓ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts t
o become errors
```

```
require(tidymodels)
```

```
## Loading required package: tidymodels
## ── Attaching packages ────────────────────────────────────── tidymodels 1.2.0 ──
## ✓ broom        1.0.6     ✓ rsample      1.2.1
## ✓ dials        1.2.1     ✓ tune         1.2.1
## ✓ infer        1.0.7     ✓ workflows    1.1.4
## ✓ modeldata    1.4.0     ✓ workflowsets 1.1.0
## ✓ parsnip      1.2.1     ✓ yardstick    1.3.1
## ✓ recipes      1.1.0
## ── Conflicts ───────────────────────────────────────── tidymodels_conflicts() ──
## ✕ scales::discard() masks purrr::discard()
## ✕ dplyr::filter()   masks stats::filter()
## ✕ recipes::fixed()  masks stringr::fixed()
## ✕ dplyr::lag()      masks stats::lag()
## ✕ yardstick::spec() masks readr::spec()
## ✕ recipes::step()   masks stats::step()
## • Use tidymodels_prefer() to resolve common conflicts.
```

```
fn <- read_rds("https://github.com/jbisbee1/ISP_Data_Science_2024/raw/main/data/fn_clean
ed_final.rds")
```

RQ: Relationship between damage_to_players and won

```
summary(fn %>%
  select(won,damage_to_players))
```

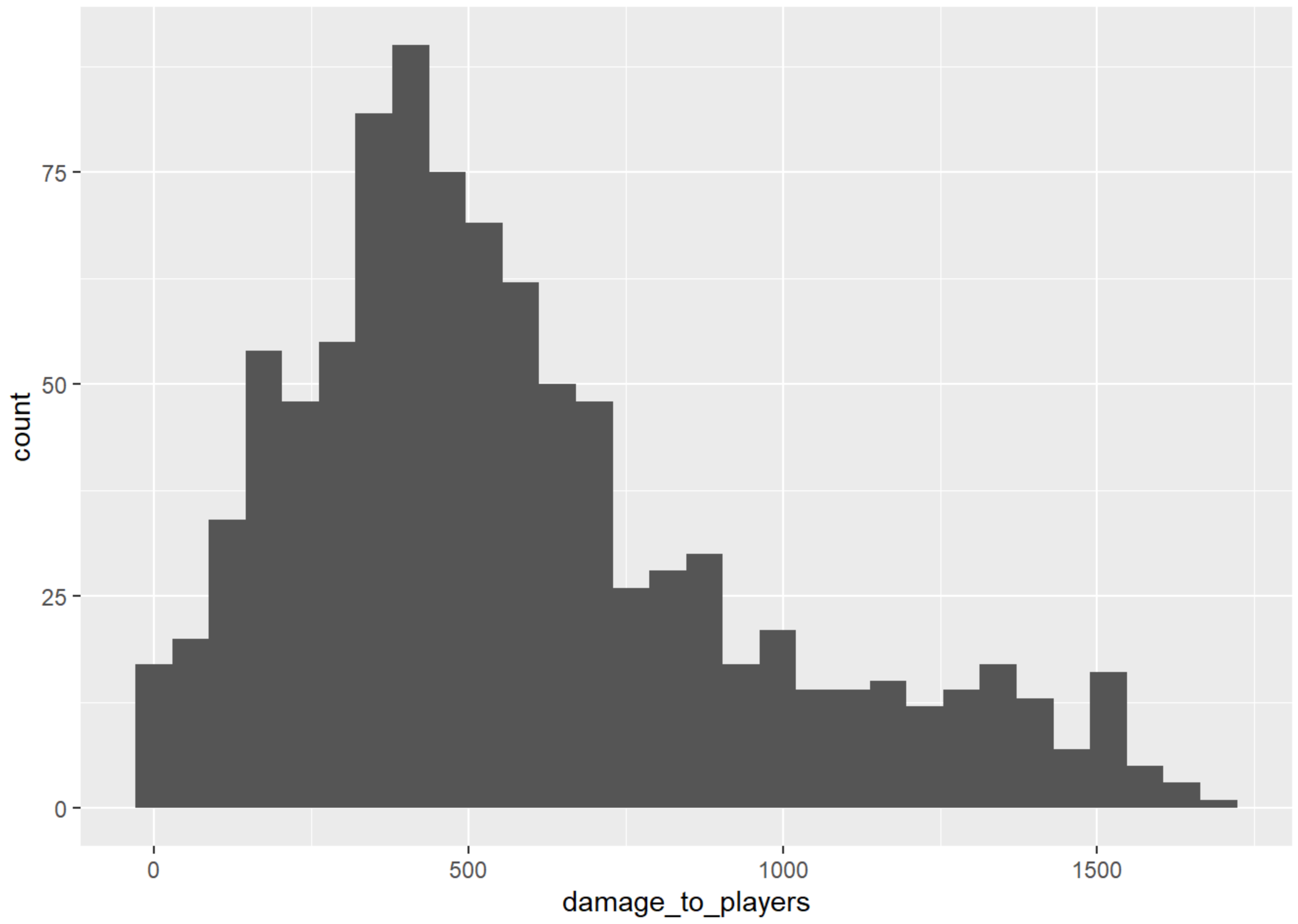
```
##       won         damage_to_players
##  Min.   :0.0000   Min.   :   0.0
##  1st Qu.:0.0000   1st Qu.: 334.0
##  Median :0.0000   Median : 499.0
##  Mean   :0.3041   Mean   : 581.7
##  3rd Qu.:1.0000   3rd Qu.: 764.0
##  Max.   :1.0000   Max.   :1693.0
```

```
fn %>%
  select(damage_to_players)
```

```
## # A tibble: 957 × 1
##    damage_to_players
##                <dbl>
##  1               372
##  2               354
##  3               206
##  4               286
##  5               823
##  6               122
##  7               663
##  8               395
##  9              1031
## 10               338
## # ℹ 947 more rows
```

```
# Univariate visualization
fn %>%
  ggplot(aes(x = damage_to_players)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Linear regression

```
m_lm <- lm(won ~ damage_to_players,
           data = fn)

require(broom)
tidy(m_lm)
```

```
## # A tibble: 2 × 5
##   term              estimate std.error statistic  p.value
##   <chr>                <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)       0.0288   0.0258         1.12 2.64e- 1
## 2 damage_to_players 0.000473 0.0000375     12.6  8.39e-34
```

#Getting YHat

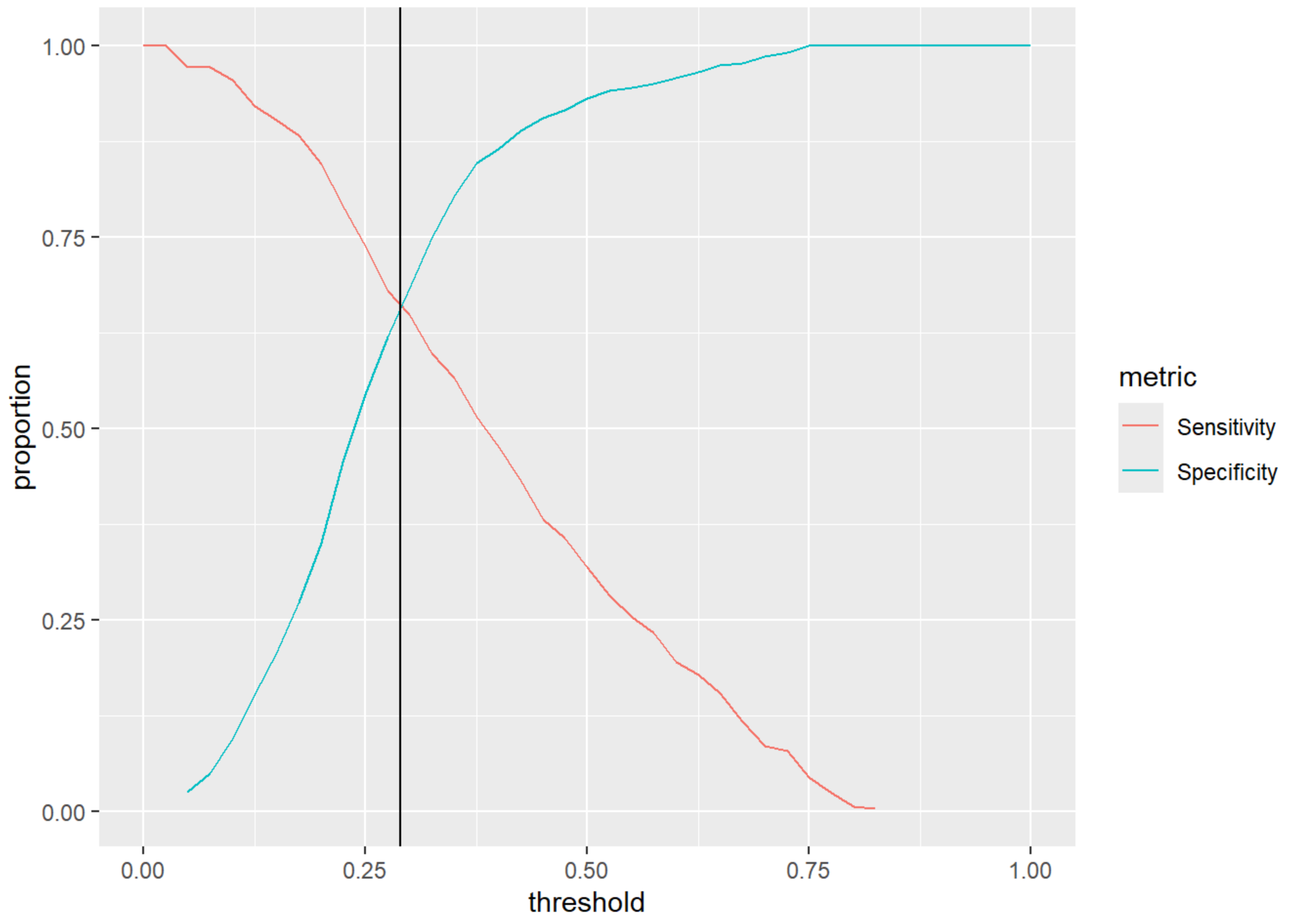
```
fn <- fn %>%
  mutate(prob_win_lm = predict(m_lm))

fn <- fn %>%
  mutate(pred_win_lm = ifelse(prob_win_lm > .5,
                              1,
                              0))

# Creating the sensitivity & specificity table
fn %>%
  group_by(won,pred_win_lm) %>%
  summarise(nGames = n()) %>%
  group_by(won) %>%
  mutate(total_games = sum(nGames)) %>%
  ungroup() %>%
  mutate(proportion = nGames / total_games) %>%
  mutate(accuracy = sum((won == pred_win_lm)*nGames) / sum(nGames))
```

```
## # A tibble: 4 × 6
##     won pred_win_lm nGames total_games proportion accuracy
##   <dbl>       <dbl>  <int>       <int>      <dbl>    <dbl>
## 1     0           0    620         666     0.931     0.745
## 2     0           1     46         666     0.0691    0.745
## 3     1           0    198         291     0.680     0.745
## 4     1           1     93         291     0.320     0.745
```

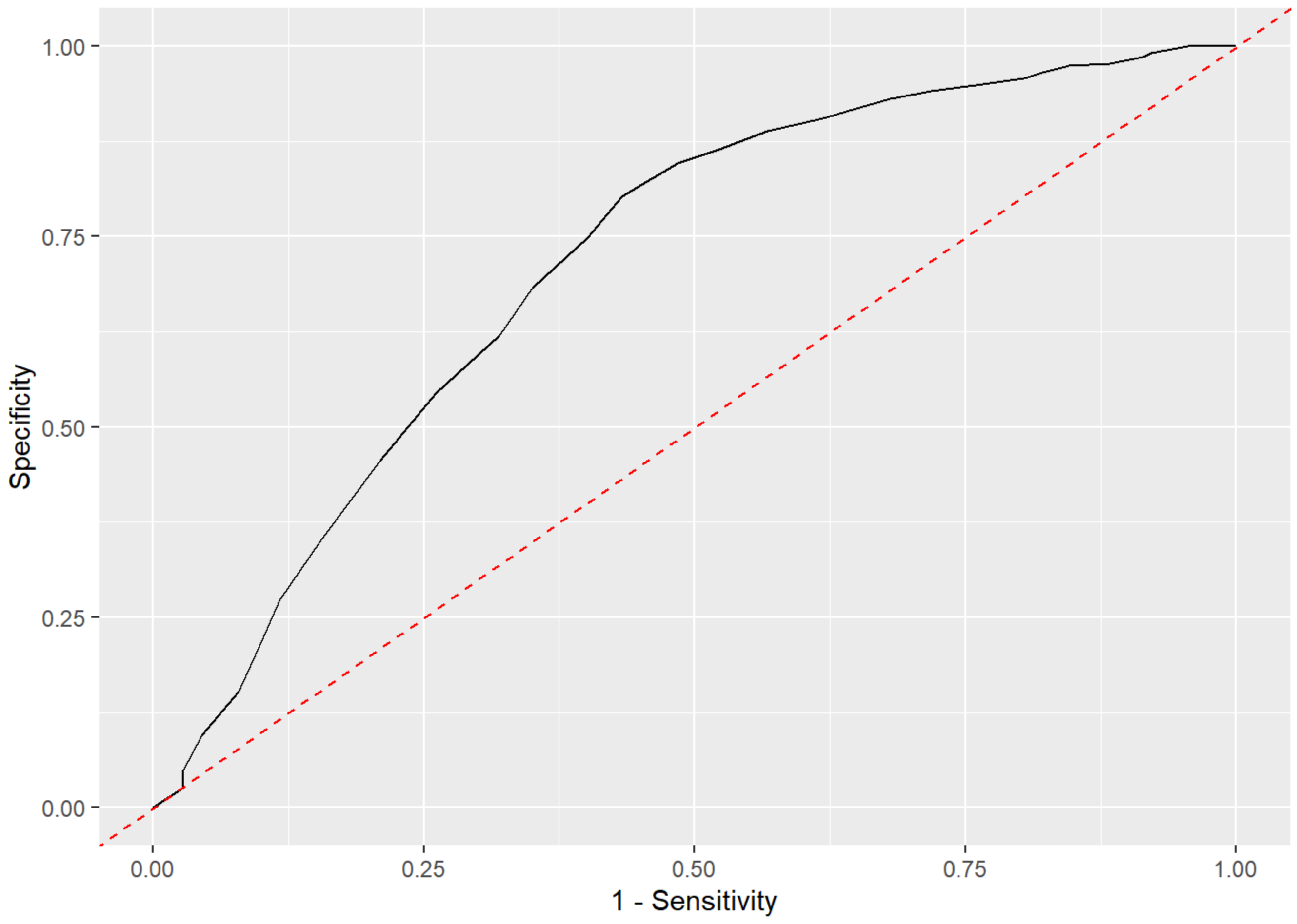
```
# Create sensitivity & specificity plot
toplot <- NULL
for(thresh in seq(0,1,by = 0.025)) {
  fn <- fn %>%
  mutate(pred_win_lm = ifelse(prob_win_lm > thresh,
                              1,
                              0))
# Creating the sensitivity & specificity table
answer <- fn %>%
  group_by(won,pred_win_lm) %>%
  summarise(nGames = n()) %>%
  group_by(won) %>%
  mutate(total_games = sum(nGames)) %>%
  ungroup() %>%
  mutate(proportion = nGames / total_games) %>%
  # mutate(accuracy = sum((won == pred_win_lm)*nGames) / sum(nGames)) %>%
  mutate(threshold = thresh)

toplot <- toplot %>%
  bind_rows(answer)
}
```

# Look at sensitivity & specificity

```
# Intersection plot
toplot <- toplot %>%
  mutate(metric = ifelse(won == 1 & pred_win_lm == 1,'Sensitivity',
                         ifelse(won == 0 & pred_win_lm == 0,'Specificity',
                                NA)))

toplot %>%
  drop_na(metric) %>%
  ggplot(aes(x = threshold,
             y = proportion,
             color = metric)) +
  geom_line() +
  geom_vline(xintercept = .29)
```



```
# ROC plot
toplot %>%
  drop_na(metric) %>%
  select(proportion,threshold,metric) %>%
  pivot_wider(names_from = 'metric',
              values_from = 'proportion',
              values_fill = 0) %>%
  ggplot(aes(x = 1-Sensitivity,
             y = Specificity)) +
  geom_line() +
  geom_abline(intercept = 0,
              slope = 1,color = 'red',linetype = 'dashed')
```



Calculate AUC

```
require(tidymodels)
# roc_auc()

forAUC <- fn %>%
  select(won,prob_win_lm) %>%
  mutate(won = factor(won,levels = c('1','0')))

roc_auc(forAUC,won,prob_win_lm)
```

```
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.717
```

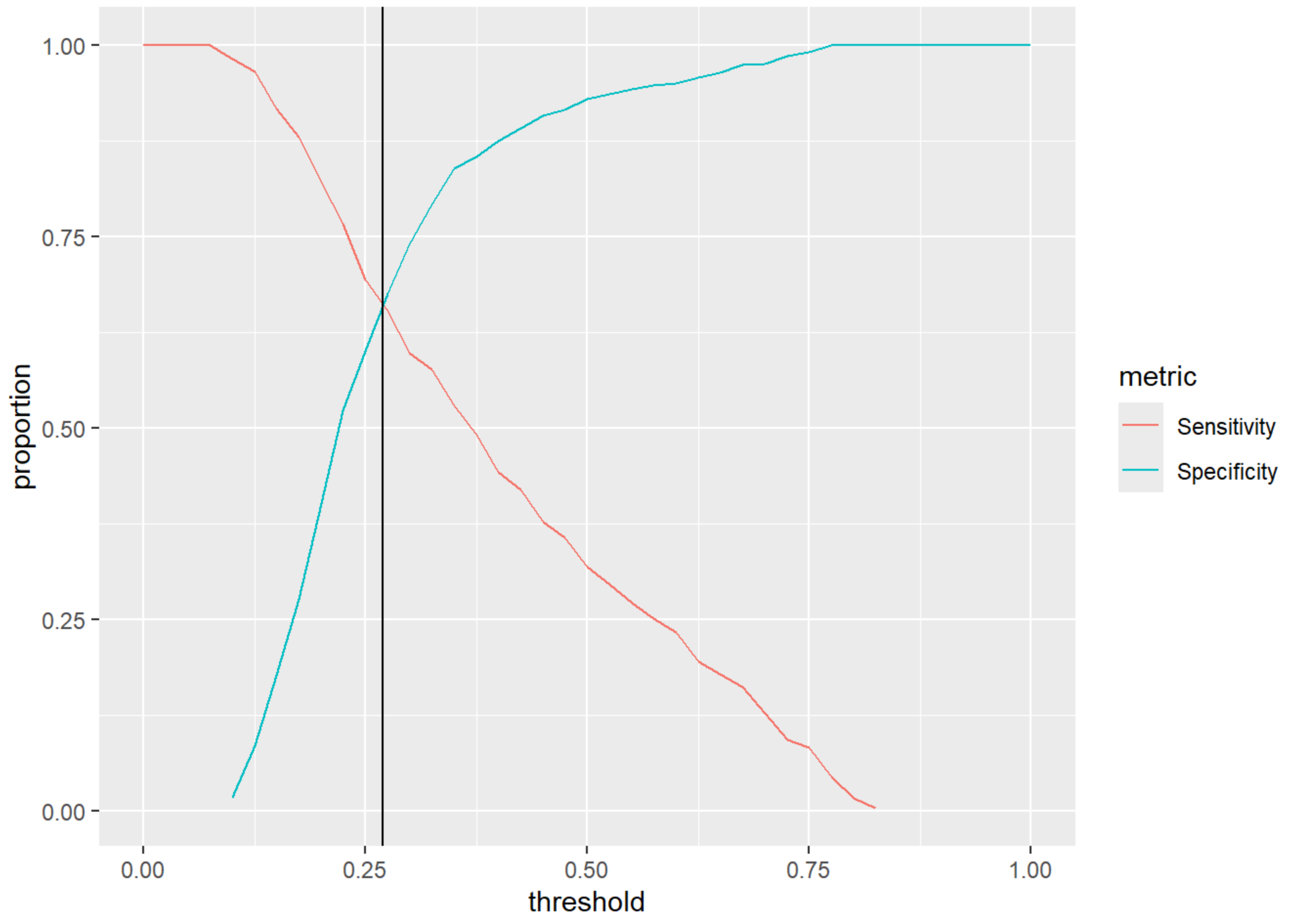
Running a logit in R

```
m_glm <- glm(formula = won ~ damage_to_players,
             data = fn,
             family = binomial(link = "logit"))

tidy(m_glm)
```

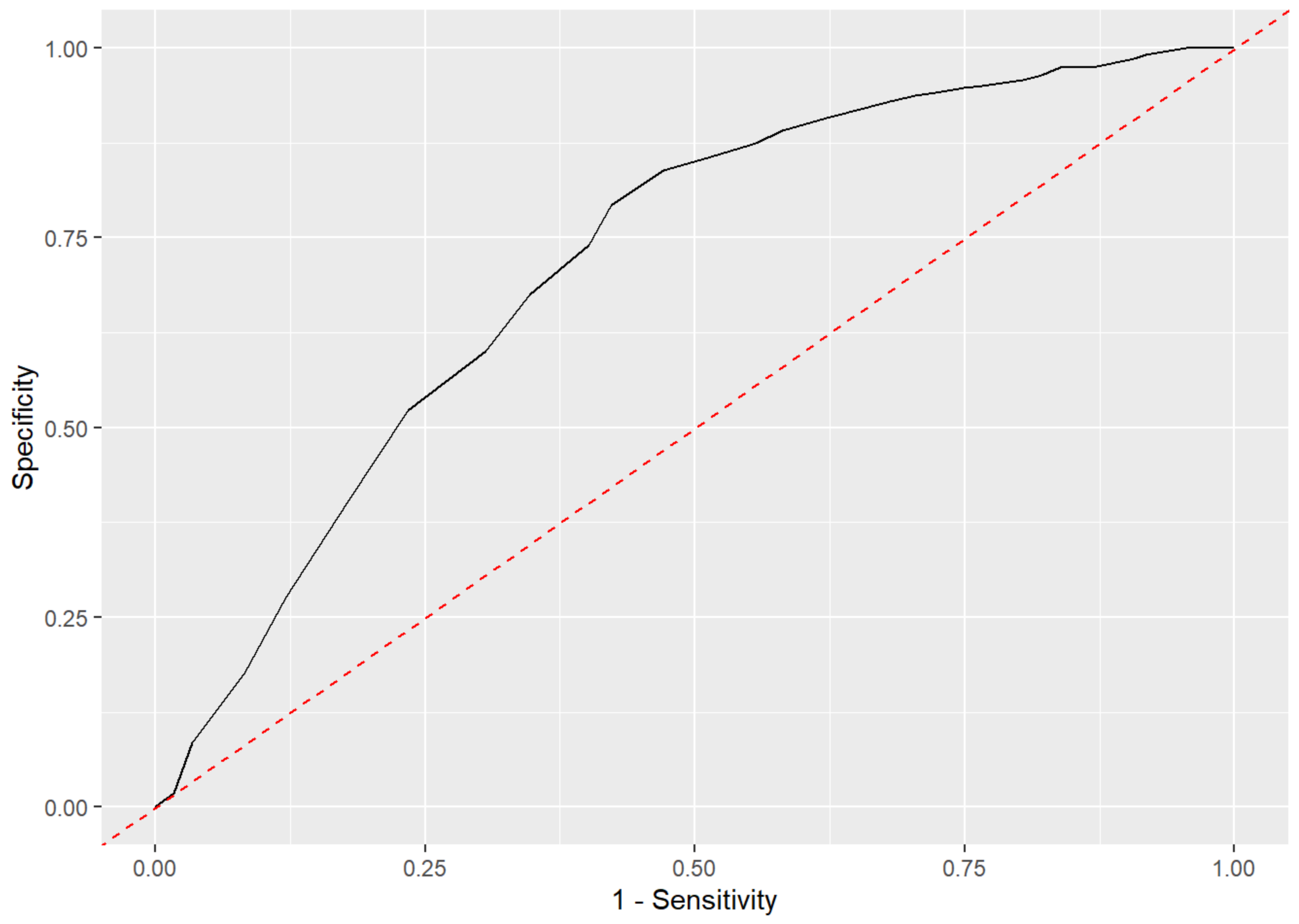
```
## # A tibble: 2 × 5
##   term              estimate std.error statistic  p.value
##   <chr>                <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)       -2.26     0.158        -14.3 1.92e-46
## 2 damage_to_players  0.00230  0.000213      10.8 5.52e-27
```

Evaluate the model

```r
fn <- fn %>%
  mutate(prob_win_glm = predict(m_glm,type = 'response'))

# Create sensitivity & specificity plot
toplot <- NULL
for(thresh in seq(0,1,by = 0.025)) {
  fn <- fn %>%
  mutate(pred_win_glm = ifelse(prob_win_glm > thresh,
                               1,
                               0))
# Creating the sensitivity & specificity table
answer <- fn %>%
  group_by(won,pred_win_glm) %>%
  summarise(nGames = n()) %>%
  group_by(won) %>%
  mutate(total_games = sum(nGames)) %>%
  ungroup() %>%
  mutate(proportion = nGames / total_games) %>%
  # mutate(accuracy = sum((won == pred_win_lm)*nGames) / sum(nGames)) %>%
  mutate(threshold = thresh)

toplot <- toplot %>%
  bind_rows(answer)
}



# Intersection plot
toplot <- toplot %>%
  mutate(metric = ifelse(won == 1 & pred_win_glm == 1,'Sensitivity',
                         ifelse(won == 0 & pred_win_glm == 0,'Specificity',
                                NA)))

toplot %>%
  drop_na(metric) %>%
  ggplot(aes(x = threshold,
             y = proportion,
             color = metric)) +
  geom_line() +
  geom_vline(xintercept = .27)
```



```
# ROC plot
toplot %>%
  drop_na(metric) %>%
  select(proportion,threshold,metric) %>%
  pivot_wider(names_from = 'metric',
              values_from = 'proportion',
              values_fill = 0) %>%
  ggplot(aes(x = 1-Sensitivity,
             y = Specificity)) +
  geom_line() +
  geom_abline(intercept = 0,
              slope = 1,color = 'red',linetype = 'dashed')
```



```
forAUC <- fn %>%
  select(won,prob_win_glm) %>%
  mutate(won = factor(won,levels = c('1','0')))

roc_auc(forAUC,won,prob_win_glm)
```

```
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.717
```

Random forests with ranger()

```
require(ranger)
```

```
## Loading required package: ranger
```

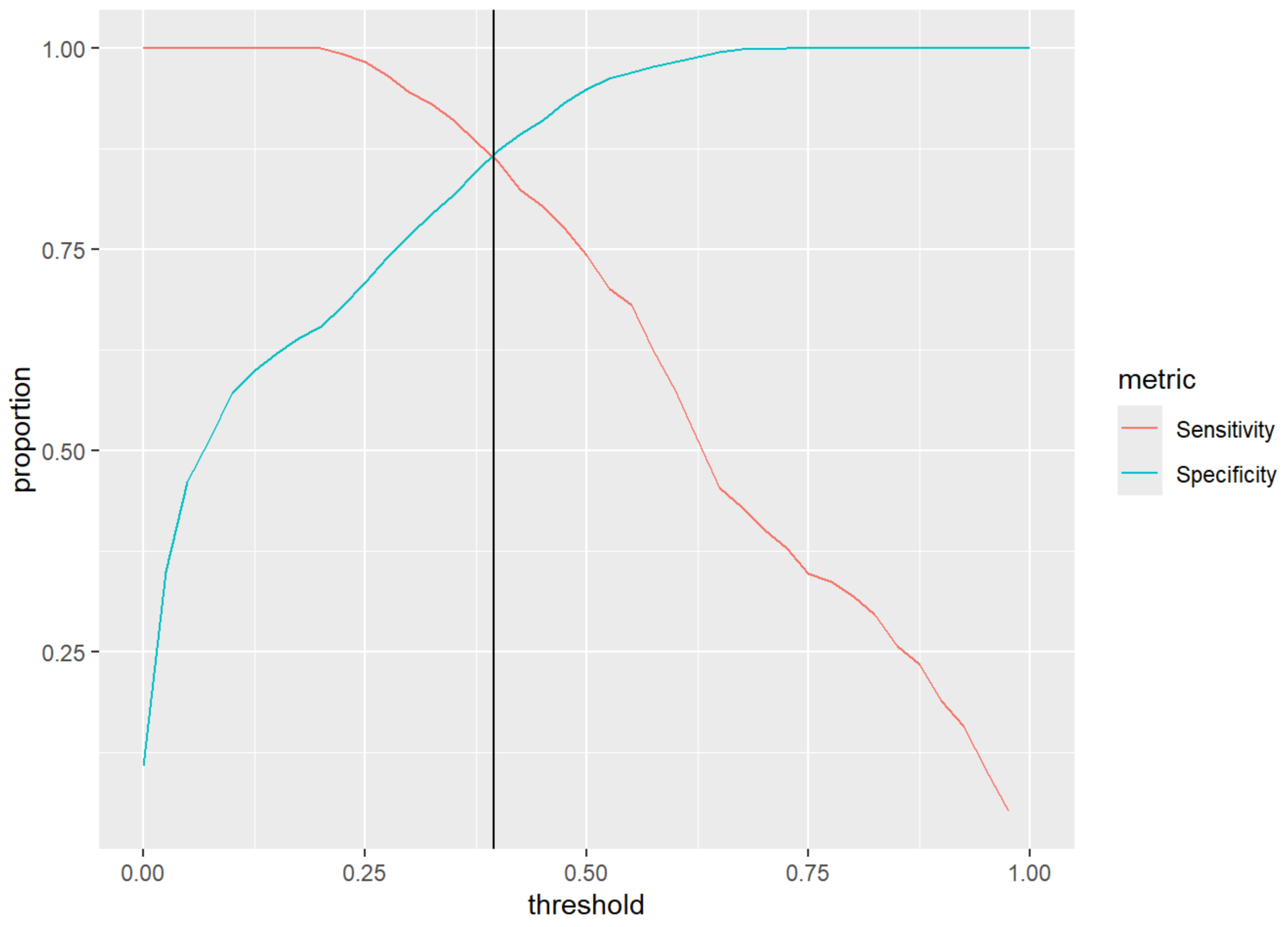
```
m_rf <- ranger(formula = won ~ damage_to_players,
       data = fn)

m_rf
```

```
## Ranger result
##
## Call:
##  ranger(formula = won ~ damage_to_players, data = fn)
##
## Type:                             Regression
## Number of trees:                  500
## Sample size:                      957
## Number of independent variables:  1
## Mtry:                             1
## Target node size:                 5
## Variable importance mode:         none
## Splitrule:                        variance
## OOB prediction error (MSE):       0.2415667
## R squared (OOB):                  -0.1403541
```

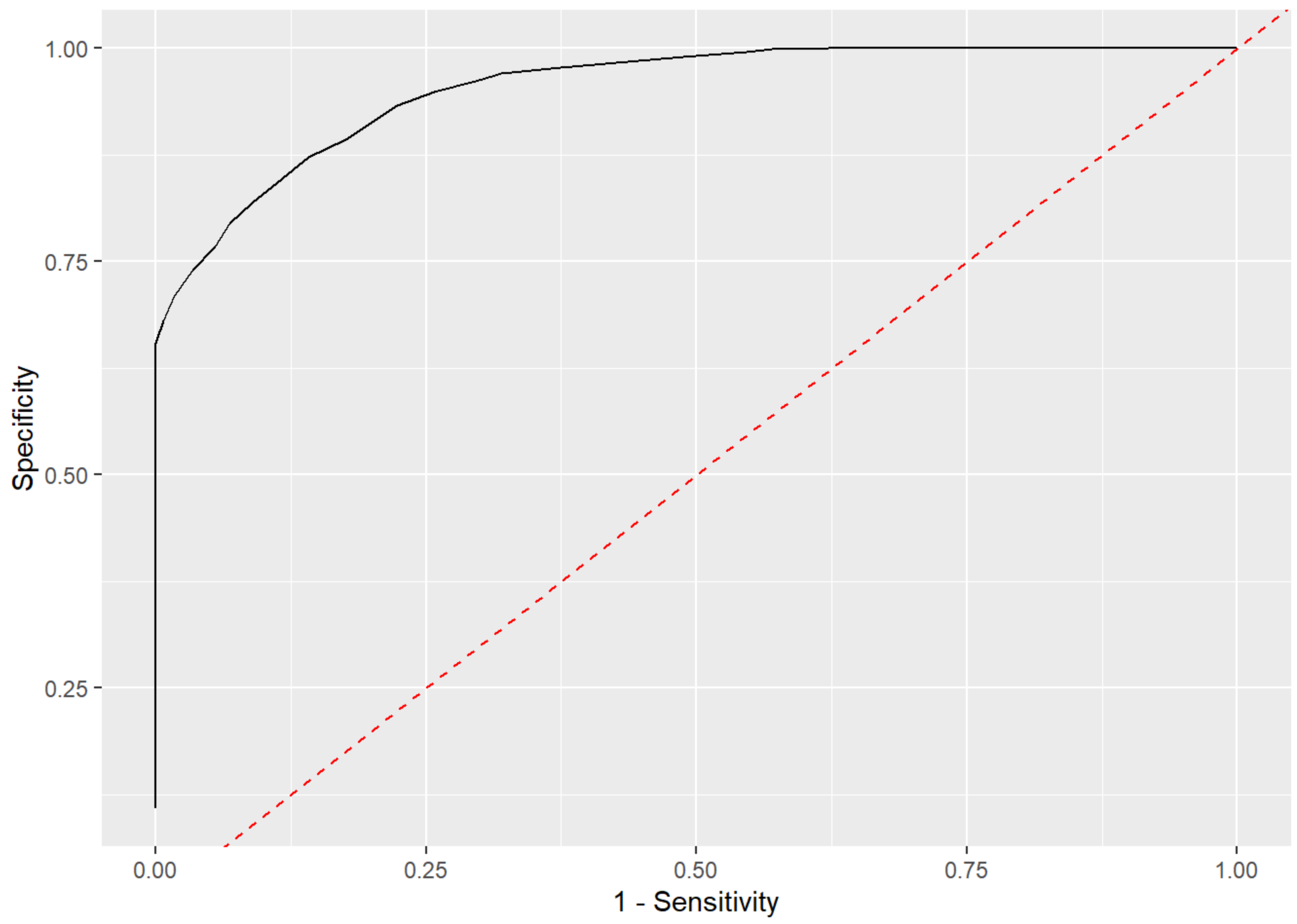
```
# Evaluate
tmp_rf <- predict(m_rf,data = fn)

fn <- fn %>%
  mutate(prob_win_rf = tmp_rf$predictions)

# Create sensitivity & specificity plot
toplot <- NULL
for(thresh in seq(0,1,by = 0.025)) {
  fn <- fn %>%
  mutate(pred_win_rf = ifelse(prob_win_rf > thresh,
                              1,
                              0))
# Creating the sensitivity & specificity table
answer <- fn %>%
  group_by(won,pred_win_rf) %>%
  summarise(nGames = n()) %>%
  group_by(won) %>%
  mutate(total_games = sum(nGames)) %>%
  ungroup() %>%
  mutate(proportion = nGames / total_games) %>%
  # mutate(accuracy = sum((won == pred_win_lm)*nGames) / sum(nGames)) %>%
  mutate(threshold = thresh)

toplot <- toplot %>%
  bind_rows(answer)
}
```

```
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
## `summarise()` has grouped output by 'won'. You can override using the `.groups`
## argument.
```

```r
# Intersection plot
toplot <- toplot %>%
  mutate(metric = ifelse(won == 1 & pred_win_rf == 1,'Sensitivity',
                         ifelse(won == 0 & pred_win_rf == 0,'Specificity',
                                NA)))

toplot %>%
  drop_na(metric) %>%
  ggplot(aes(x = threshold,
             y = proportion,
             color = metric)) +
  geom_line() +
  geom_vline(xintercept = .395)
```



```
# ROC plot
toplot %>%
  drop_na(metric) %>%
  select(proportion,threshold,metric) %>%
  pivot_wider(names_from = 'metric',
              values_from = 'proportion',
              values_fill = 0) %>%
  ggplot(aes(x = 1-Sensitivity,
             y = Specificity)) +
  geom_line() +
  geom_abline(intercept = 0,
              slope = 1,color = 'red',linetype = 'dashed')
```



```
forAUC <- fn %>%
  select(won,prob_win_rf) %>%
  mutate(won = factor(won,levels = c('1','0')))

roc_auc(forAUC,won,prob_win_rf)
```

```
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.953
```

Cross validation to depress ourselves

```r
set.seed(123)
cvRes <- NULL
for(i in 1:100) {
  train <- fn %>%
    select(won,damage_to_players) %>%
    drop_na() %>%
    sample_n(size = round(nrow(.)*.6))

  test <- fn %>%
    select(won,damage_to_players) %>%
    drop_na() %>%
    anti_join(train)

  # Train the models
  tmpLM <- lm(won ~ damage_to_players,data = train)
  tmpGLM <- glm(won ~ damage_to_players,data = train,
                family = binomial(link = "logit"))
  tmpRF <- ranger(won ~ damage_to_players,data = train)

  # Test the models
  test %>%
    mutate(pred_LM = predict(tmpLM,newdata = test),
           pred_GLM = predict(tmpGLM,newdata = test))
}
```

```
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
## Joining with `by = join_by(won, damage_to_players)`
```