Supervised Learning

Intro to variable selection

June 25th, 2021

The setting

We wish to learn a linear model. Our estimate (denoted by hats) is

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \cdots + \hat{\beta}_p X_p$$

Why would we attempt to select a **subset** of the p variables?

- *To improve prediction accuracy*
    - Eliminating uninformative predictors can lead to lower variance in the test-set MSE, at the expense of a slight increase in bias
- *To improve model interpretability*
    - Eliminating uninformative predictors is obviously a good thing when your goal is to tell the story of how your predictors are associated with your response.

Best subset selection

- Start with the **null model** $\mathcal{M}_0$ (intercept-only) that has no predictors
  - just predicts the sample mean for each observation
- For $k = 1, 2, \ldots, p$ (each possible number of predictors)
  - Fit **all** $\binom{p}{k} = \frac{p!}{k!(p-k)!}$ with exactly k predictors
  - Pick the best (some criteria) among these $\binom{p}{k}$ models, call it $\mathcal{M}_k$
  - Best can be up to the user: cross-validation error, highest adjusted R^2, etc.
- Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$

This is not typically used in research!

- only practical for a smaller number of variables
- arbitrary way of defining **best** and ignores **prior knowledge** about potential predictors

Use the shoe leather approach

Prof. David Freeman:

- algorithms can be tempting but they are NOT substitutes!
- you should NOT avoid the hard work of EDA in your modeling efforts

Variable selection is a difficult problem!

- Like much of a statistics & data science research there is not one unique, correct answer

You should justify which predictors / variables used in modeling based on:

- **context**,
- **extensive EDA**, and
- **model assessment based on holdout predictions**

Covariance and correlation

- **Covariance** is a measure of the **linear** dependence between two variables
  - To be *"uncorrelated"* is not the same as to be *"independent"*...
  - Independence means **there is no dependence**, linear or otherwise
- **Correlation** is a *normalized* form of covariance, ranges from -1 through 0 to 1
  - -1 means one variable linearly decreases absolutely in value while the other increases in value
  - 0 means no linear dependence
  - 1 means one variable linear increases absolutely while the other increases
- We can use the `cov()` / `cor()` functions in `R` to generate the **covariance** / **correlation** matrices

Example data: NFL teams summary

Created dataset using `nflfastR` summarizing NFL team performances from 1999 to 2020

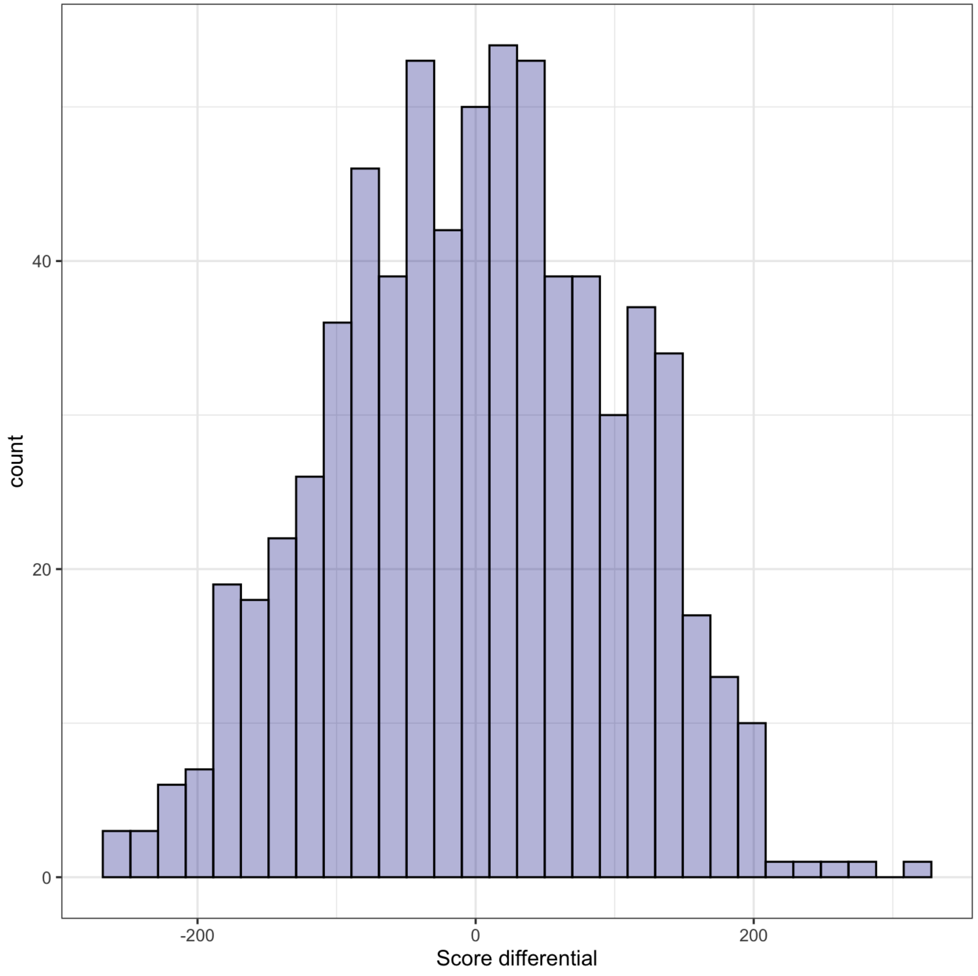
```
library(tidyverse)
nfl_teams_data <- read_csv("http://www.stat.cmu.edu/cmsac/sure/2021/materials/data/regression_pro
nfl_teams_data
```

```
## # A tibble: 701 × 55
##    season team  offens…¹ offen…² offen…³ offen…⁴ offen…⁵ offen…⁶ offen…⁷ offen…⁸
##     <dbl> <chr>    <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
##  1   1999 ARI      0.477    2796    1209    4.67    3.15       0     NaN       0
##  2   1999 ATL      0.504    3317    1176    6.08    3.20       0     NaN      11
##  3   1999 BAL      0.452    2805    1663    5.07    4.13       0     NaN       0
##  4   1999 BUF      0.540    3275    2038    6.17    4.13       0     NaN     161
##  5   1999 CAR      0.552    4144    1484    6.68    4.29       0     NaN      89
##  6   1999 CHI      0.561    4090    1359    5.75    3.55       0     NaN     508
##  7   1999 CIN      0.498    3178    1971    5.37    4.63       0     NaN       0
##  8   1999 CLE      0.489    2574    1140    4.71    3.67       0     NaN      35
##  9   1999 DAL      0.560    3083    2054    5.95    4.29       0     NaN       0
## 10   1999 DEN      0.546    3378    1852    5.85    4.05       0     NaN       9
## # … with 691 more rows, 45 more variables: offense_ave_yac <dbl>,
## #   offense_n_plays_pass <dbl>, offense_n_plays_run <dbl>,
## #   offense_n_interceptions <dbl>, offense_n_fumbles_lost_pass <dbl>,
## #   offense_n_fumbles_lost_run <dbl>, offense_total_epa_pass <dbl>,
```

Modeling NFL score differential

Interested in modeling a team's **score differential**

```
nfl_teams_data <- nfl_teams_data %>%
  mutate(score_diff =
           points_scored - points_allowed)
nfl_teams_data %>%
  ggplot(aes(x = score_diff)) +
  geom_histogram(color = "black",
                 fill = "darkblue",
                 alpha = 0.3) +
  theme_bw() +
  labs(x = "Score differential")
```

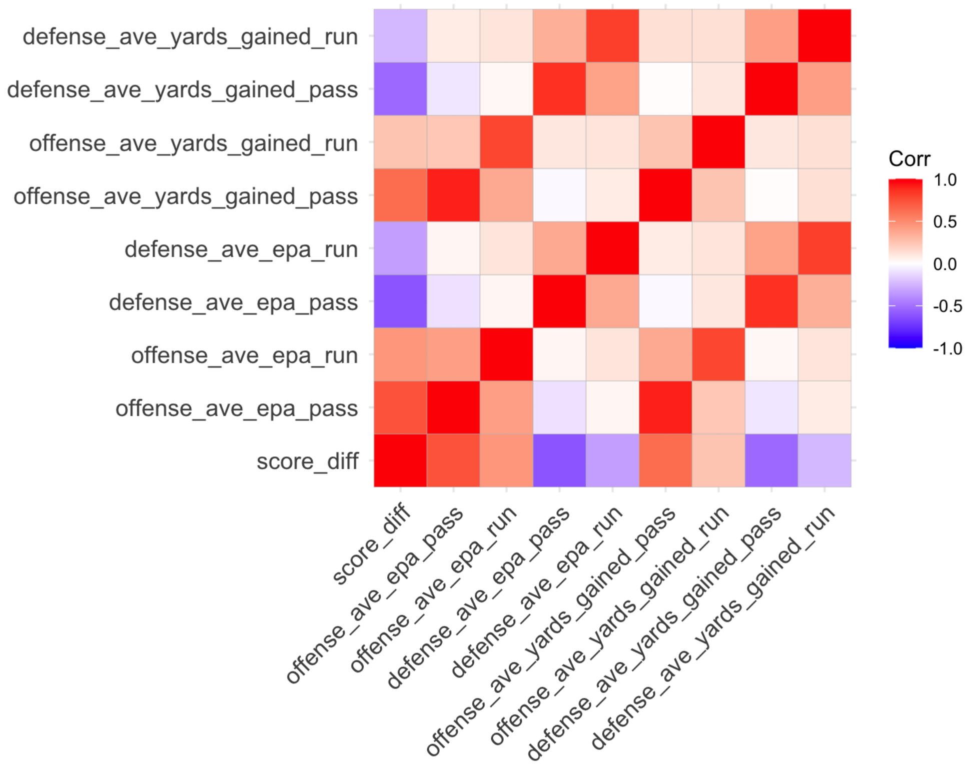


Correlation matrix of score differential and candidate predictors

- Interested in `score_diff` relationships with team passing and rush statistics

- View the correlation matrix with `ggcorrplot`

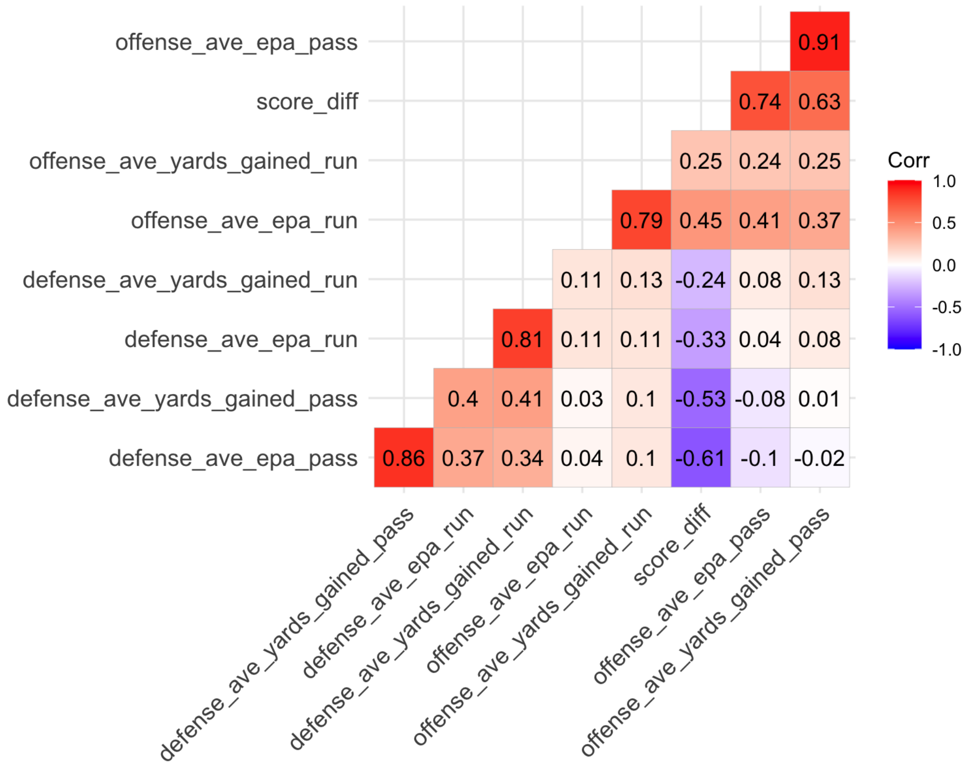
```
library(ggcorrplot)
nfl_model_data <- nfl_teams_data %>%
  dplyr::select(score_diff, offense_ave_epa_p
                offense_ave_epa_run,
                defense_ave_epa_pass,
                defense_ave_epa_run,
                offense_ave_yards_gained_pass
                offense_ave_yards_gained_run,
                defense_ave_yards_gained_pass
                defense_ave_yards_gained_run)
nfl_cor_matrix <- cor(nfl_model_data)
ggcorrplot(nfl_cor_matrix)
```



Customize the appearance of the correlation matrix

- Avoid redundancy by only using one half of matrix with `type`
- Add correlation value labels using `lab` (but round first!)
- Can arrange variables based on clustering...

```
round_cor_matrix <-
  round(cor(nfl_model_data), 2)
ggcorrplot(round_cor_matrix,
           hc.order = TRUE,
           type = "lower",
           lab = TRUE)
```



Clustering variables using the correlation matrix

Apply hierarchical clustering to variables instead of observations

- Select the explanatory variables of interest from our data

```
nfl_ex_vars <- dplyr::select(nfl_model_data, -score_diff)
```

- Compute correlation matrix of these variables:

```
exp_cor_matrix <- cor(nfl_ex_vars)
```

- Correlations measure similarity and can be negative **BUT** distances measure dissimilarity and **CANNOT**
- Convert your correlations to all be ≥ 0: e.g., $1 - |\rho|$ (which drops the sign) or $1 - \rho$

```
cor_dist_matrix <- 1 - abs(exp_cor_matrix)
```

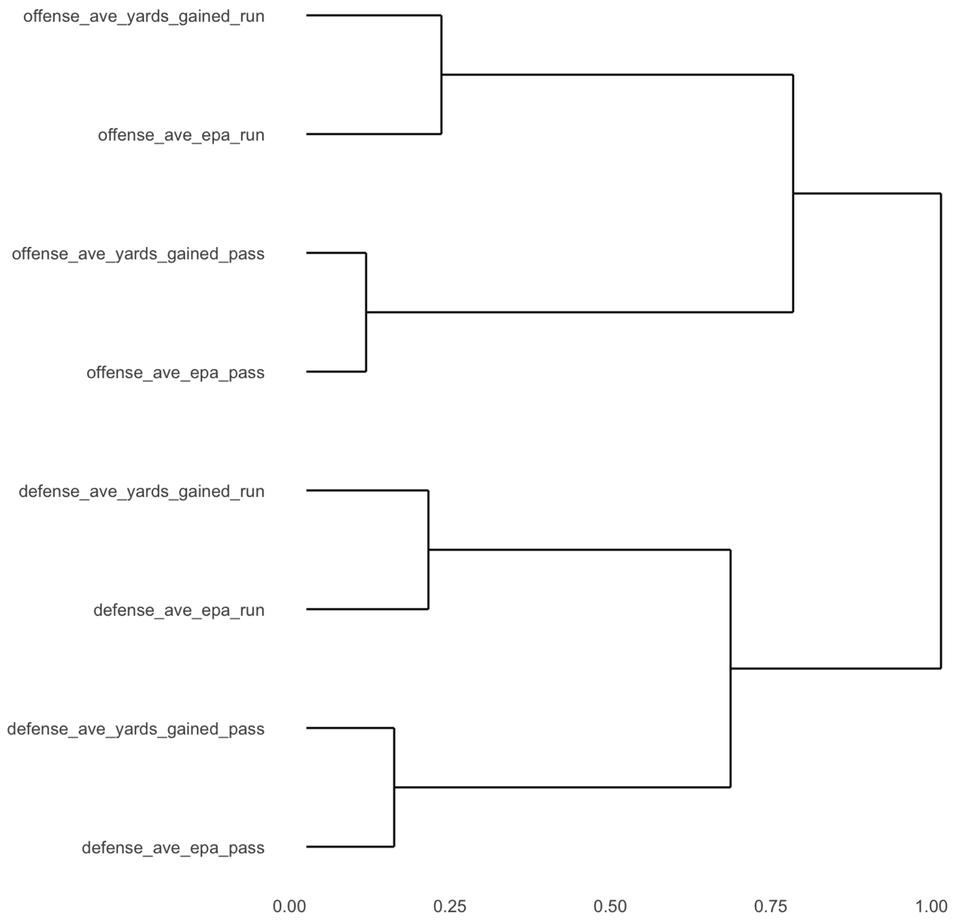
- Convert to distance matrix before using `hclust`

```
cor_dist_matrix <- as.dist(cor_dist_matrix)
```

Clustering variables using the correlation matrix

- Cluster variables using `hclust()` as before!
- Use `ggdendro` to quickly visualize dendrogram

```
library(ggdendro)
nfl_exp_hc <- hclust(cor_dist_matrix,
                     "complete")
ggdendrogram(nfl_exp_hc,
             rotate = TRUE,
             size = 2)
```

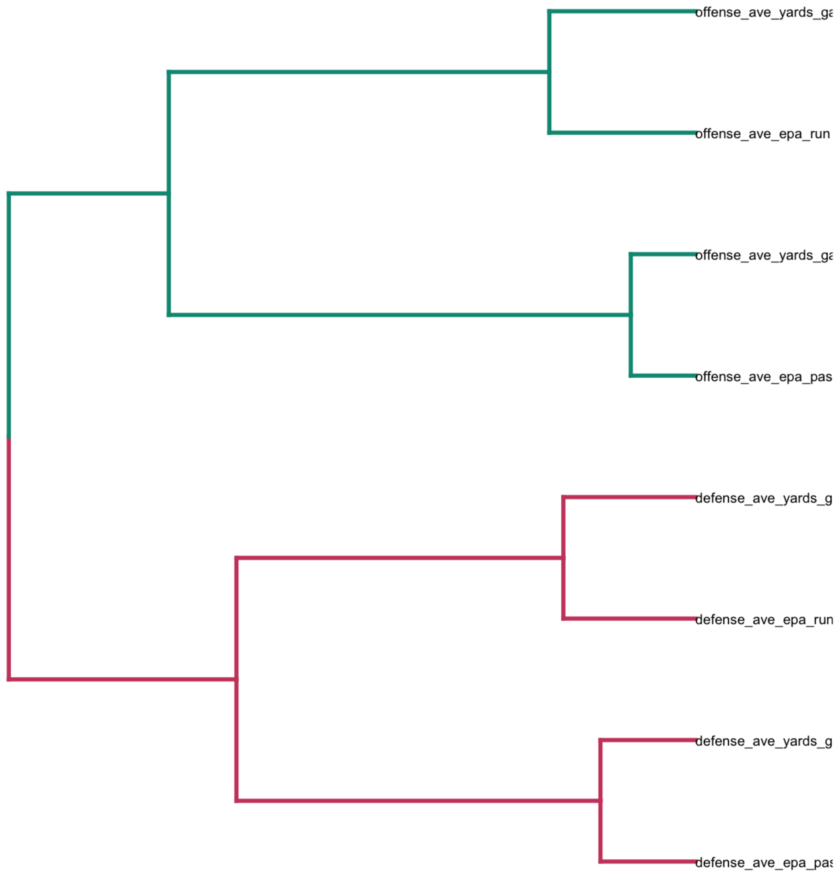


Clustering variables using the correlation matrix

- Another flexible option is dendextend

```
library(dendextend)
cor_dist_matrix %>%
  hclust() %>%
  as.dendrogram() %>%
  set("branches_k_col",
      k = 2) %>%
  set("labels_cex", .5) %>%
  ggplot(horiz = TRUE)
```

- Explore the package documentation for more formatting

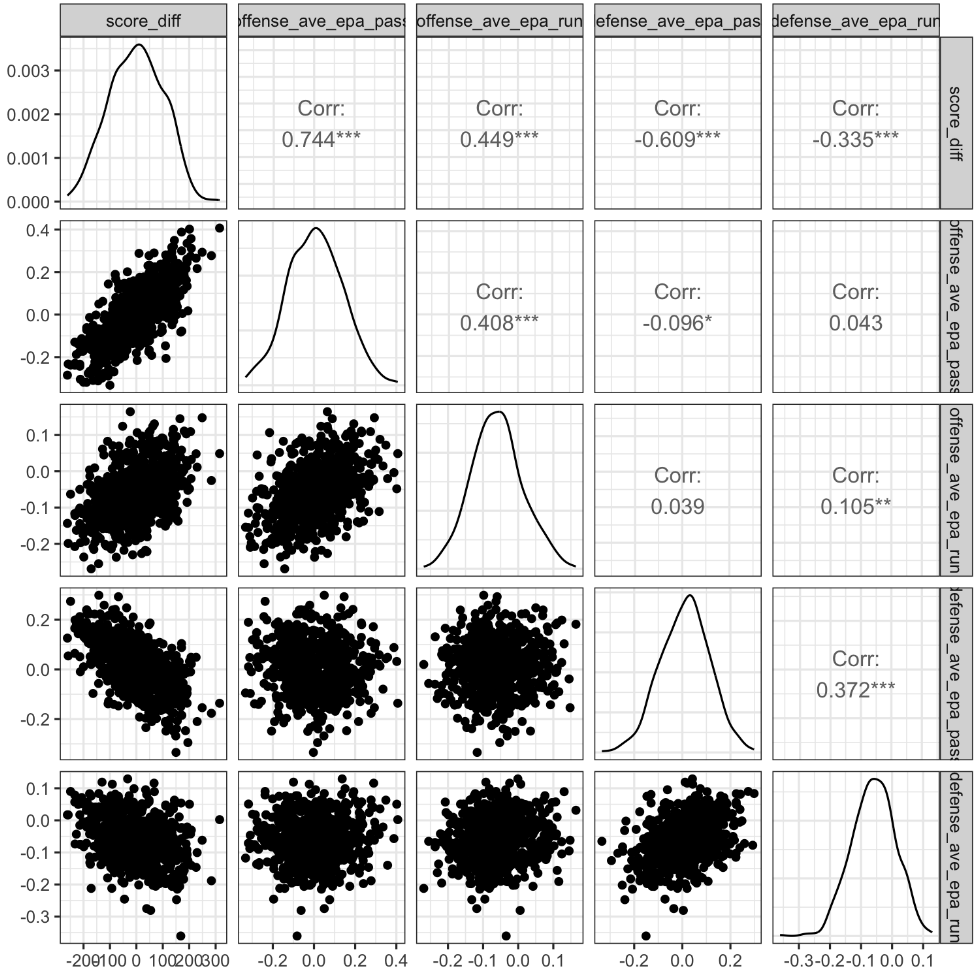


Back to the response variable...

Use the `GGally` package to easily create **pairs** plots of multiple variables

- **always look at your data**
- correlation values alone are not enough!
- what if a variable displayed a quadratic relationship?

```
library(GGally)
ggpairs(nfl_model_data,
        columns =
          c("score_diff", "offense_ave_epa_pa
            "offense_ave_epa_run", "defense_a
            "defense_ave_epa_run")) +
  theme_bw()
```



Do running statistics matter for modeling score differential?

Will use **5-fold cross-validation** to assess how well different sets of variables (combinations of `pass` & `run` variables) perform in predicting `score_diff`?

Can initialize a column of the **test** fold assignments to our dataset with the `sample()` function:

```
set.seed(2020)
nfl_model_data <- nfl_model_data %>%
  mutate(test_fold = sample(rep(1:5, length.out = n())))
```

Always remember to set your seed prior to any k-fold cross-validation!

Writing a function for k-fold cross-validation

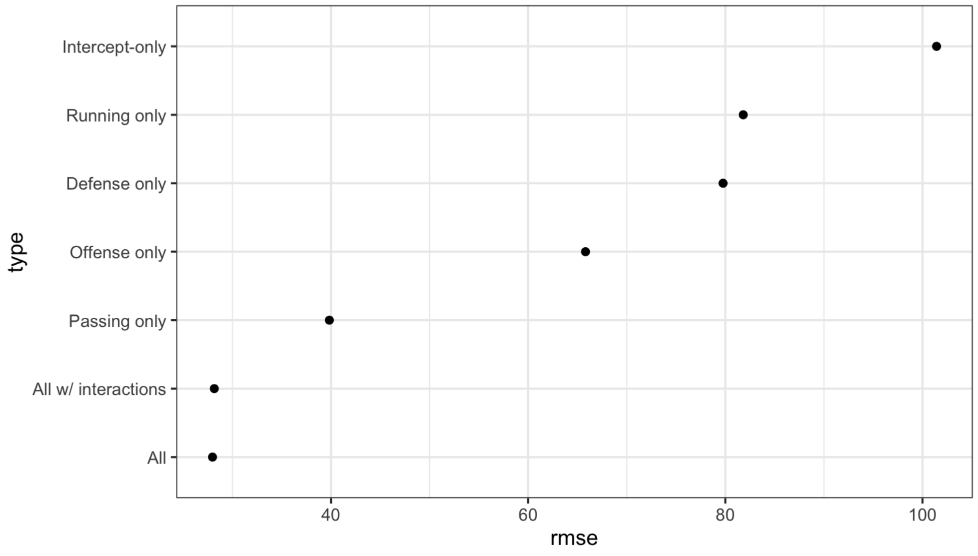
```
get_cv_preds <- function(model_formula, data = nfl_model_data) {
  # generate holdout predictions for every row based season
  map_dfr(unique(data$test_fold),
          function(holdout) {
            # Separate test and training data:
            test_data <- data %>%
              filter(test_fold == holdout)
            train_data <- data %>%
              filter(test_fold != holdout)

            # Train model:
            reg_model <- lm(as.formula(model_formula), data = train_data)

            # Return tibble of holdout results:
            tibble(test_preds = predict(reg_model, newdata = test_data),
                   test_actual = test_data$score_diff,
                   test_fold = holdout)
          })
}
```

Function enables easy generation of holdout analysis

```
all_cv_preds <- get_cv_preds("score_diff ~  offense_ave_epa_pass + offense_ave_epa_run + defense_
all_int_cv_preds <- get_cv_preds("score_diff ~ offense_ave_epa_pass*offense_ave_epa_run + defense
run_only_cv_preds <- get_cv_preds("score_diff ~ offense_ave_epa_run + defense_ave_epa_run")
pass_only_cv_preds <- get_cv_preds("score_diff ~ offense_ave_epa_pass + defense_ave_epa_pass")
off_only_cv_preds <- get_cv_preds("score_diff ~ offense_ave_epa_pass + offense_ave_epa_run")
def_only_cv_preds <- get_cv_preds("score_diff ~ defense_ave_epa_pass + defense_ave_epa_run")
int_only_cv_preds <- get_cv_preds("score_diff ~ 1")
```

Can then summarize together for a single plot:

```
bind_rows(mutate(all_cv_preds, type = "All"),
          mutate(all_int_cv_preds, type = "Al
          mutate(pass_only_cv_preds, type = "
          mutate(run_only_cv_preds, type = "R
          mutate(off_only_cv_preds, type = "O
          mutate(def_only_cv_preds, type = "D
          mutate(int_only_cv_preds, type = "I
  group_by(type) %>%
  summarize(rmse = sqrt(mean((test_actual - t
  mutate(type = fct_reorder(type, rmse)) %>%
  ggplot(aes(x = type, y = rmse)) +
  geom_point() + coord_flip() + theme_bw()
```



Fit selected model on all data and view summary

```
all_lm <- lm(score_diff ~ offense_ave_epa_pass + offense_ave_epa_run + defense_ave_epa_pass + de
summary(all_lm)
```

```
##
## Call:
## lm(formula = score_diff ~ offense_ave_epa_pass + offense_ave_epa_run +
##     defense_ave_epa_pass + defense_ave_epa_run, data = nfl_model_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -74.850 -18.814   0.222  18.964  92.173
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)             3.525      1.732   2.035   0.0422 *
## offense_ave_epa_pass  462.886      8.737  52.979   <2e-16 ***
## offense_ave_epa_run   333.415     15.808  21.092   <2e-16 ***
## defense_ave_epa_pass -480.918     11.226 -42.838   <2e-16 ***
## defense_ave_epa_run  -298.546     16.324 -18.289   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 27.84 on 696 degrees of freedom
```

Do NOT show that summary in a presentation!

- We can instead display a **coefficient plot** with confidence intervals based on the reported standard errors

- Use the `ggcoef()` function from `GGally`

```
ggcoef(all_lm,
       exclude_intercept = TRUE,
       vline = TRUE,
       vline_color = "red") +
  theme_bw()
```

- **A well formatted table** of the summary output is appropriate for a report (not for a presentation)