Notes on Ch 11: Comparing Models with Resampling

Norman Lim

2025-08-26

This chapter talks about how to compare two or more models to understand which one is best.

Creating multiple models with workflow sets

Splitting the dataset into training and testing sets:

```
library(tidymodels)
```

```
## -- Attaching packages --------------------------------------- tidymodels 1.3.0 --

## v broom        1.0.8     v recipes      1.3.1
## v dials        1.4.0     v rsample      1.3.0
## v dplyr        1.1.4     v tibble       3.3.0
## v ggplot2      3.5.2     v tidyr        1.3.1
## v infer        1.0.8     v tune         1.3.0
## v modeldata    1.4.0     v workflows    1.2.0
## v parsnip      1.3.2     v workflowsets 1.1.1
## v purrr        1.0.4     v yardstick    1.3.2

## -- Conflicts ------------------------------------------ tidymodels_conflicts() --
## x purrr::discard() masks scales::discard()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x recipes::step()  masks stats::step()
```

```
tidymodels_prefer()
data(ames)

ames <- mutate(ames, Sale_Price = log10(Sale_Price))

set.seed(502)

ames_split <- initial_split(ames, prop = 0.80, strata = Sale_Price)
ames_train <- training(ames_split)
ames_test <- testing(ames_split)
```

Creating three different model recipes:

```
basic_rec <-
  recipe(Sale_Price ~ Neighborhood + Gr_Liv_Area + Year_Built + Bldg_Type + Latitude + Longitude, data =
  step_log(Gr_Liv_Area, base = 10) |>
  step_other(Neighborhood, threshold = 0.01) |>
  step_dummy(all_nominal_predictors())

interaction_rec <-
  basic_rec |>
  step_interact( ~ Gr_Liv_Area:starts_with("Bldg_Type_"))

spline_rec <-
  interaction_rec |>
  step_ns(Latitude, Longitude, deg_free = 50)
```

Creating a workflow set:

```
preproc <-
  list(
    basic = basic_rec,
    interact = interaction_rec,
    splines = spline_rec
  )

lm_models <- workflow_set(preproc, list(lm = linear_reg()), cross = FALSE)

lm_models
```

```
## # A workflow set/tibble: 3 x 4
##   wflow_id    info             option    result
##   <chr>       <list>           <list>    <list>
## 1 basic_lm    <tibble [1 x 4]> <opts[0]> <list [0]>
## 2 interact_lm <tibble [1 x 4]> <opts[0]> <list [0]>
## 3 splines_lm  <tibble [1 x 4]> <opts[0]> <list [0]>
```

Splitting the training data into 10 folds in preparation for cross-validation:

```
set.seed(1001)
ames_folds <- vfold_cv(ames_train, v = 10)
ames_folds
```

```
## #  10-fold cross-validation
## # A tibble: 10 x 2
##    splits             id
##    <list>             <chr>
##  1 <split [2107/235]> Fold01
##  2 <split [2107/235]> Fold02
##  3 <split [2108/234]> Fold03
##  4 <split [2108/234]> Fold04
##  5 <split [2108/234]> Fold05
##  6 <split [2108/234]> Fold06
##  7 <split [2108/234]> Fold07
##  8 <split [2108/234]> Fold08
```

```
##  9 <split [2108/234]> Fold09
## 10 <split [2108/234]> Fold10
```

Setting the `control` arguments for resampling:

```
keep_pred <- control_resamples(save_pred = TRUE, save_workflow = TRUE)
```

Resampling each model using the **purrr*-like function called `workflow_map()`:

```
lm_models <-
  lm_models |> workflow_map(
    "fit_resamples",
    # Options to workflow_map()
    seed = 1101, verbose = TRUE,
    # Options to fit_resamples()
    resamples = ames_folds, control = keep_pred
  )
```

```
## i 1 of 3 resampling: basic_lm

## v 1 of 3 resampling: basic_lm (821ms)

## i 2 of 3 resampling: interact_lm

## v 2 of 3 resampling: interact_lm (887ms)

## i 3 of 3 resampling: splines_lm

## v 3 of 3 resampling: splines_lm (1.7s)
```

```
lm_models
```

```
## # A workflow set/tibble: 3 x 4
##   wflow_id    info             option    result
##   <chr>       <list>           <list>    <list>
## 1 basic_lm    <tibble [1 x 4]> <opts[2]> <rsmp[+]>
## 2 interact_lm <tibble [1 x 4]> <opts[2]> <rsmp[+]>
## 3 splines_lm  <tibble [1 x 4]> <opts[2]> <rsmp[+]>
```

Collating the performance statistics of the models:

```
collect_metrics(lm_models) |>
  filter(.metric == "rmse")
```

```
## # A tibble: 3 x 9
##   wflow_id    .config      preproc model .metric .estimator   mean     n std_err
##   <chr>       <chr>        <chr>   <chr> <chr>   <chr>       <dbl> <int>   <dbl>
## 1 basic_lm    Preprocesso~ recipe  line~ rmse    standard   0.0803    10 0.00264
## 2 interact_lm Preprocesso~ recipe  line~ rmse    standard   0.0799    10 0.00272
## 3 splines_lm  Preprocesso~ recipe  line~ rmse    standard   0.0785    10 0.00282
```

Since the result is a tibble, it is possible to add other results to it by binding rows.

Because the option `save_workflow` was set to `TRUE` in the control arguments when the random forest (previous chapter) model was resampled, it is possible to add it to the results above.

Code for the random forest model from Chapter 10:

```
rf_model <-
  rand_forest(trees = 1000) |>
  set_engine("ranger") |>
  set_mode("regression")

rf_wflow <-
  workflow() |>
  add_formula(
    Sale_Price ~ Neighborhood + Gr_Liv_Area + Year_Built + Bldg_Type + Latitude + Longitude
  ) |>
  add_model(rf_model)

set.seed(1003)
rf_res <- rf_wflow |> fit_resamples(resamples = ames_folds, control = keep_pred)

rf_res
```

```
## # Resampling results
## # 10-fold cross-validation
## # A tibble: 10 x 5
##    splits             id     .metrics         .notes           .predictions
##    <list>             <chr>  <list>           <list>           <list>
##  1 <split [2107/235]> Fold01 <tibble [2 x 4]> <tibble [0 x 3]> <tibble>
##  2 <split [2107/235]> Fold02 <tibble [2 x 4]> <tibble [0 x 3]> <tibble>
##  3 <split [2108/234]> Fold03 <tibble [2 x 4]> <tibble [0 x 3]> <tibble>
##  4 <split [2108/234]> Fold04 <tibble [2 x 4]> <tibble [0 x 3]> <tibble>
##  5 <split [2108/234]> Fold05 <tibble [2 x 4]> <tibble [0 x 3]> <tibble>
##  6 <split [2108/234]> Fold06 <tibble [2 x 4]> <tibble [0 x 3]> <tibble>
##  7 <split [2108/234]> Fold07 <tibble [2 x 4]> <tibble [0 x 3]> <tibble>
##  8 <split [2108/234]> Fold08 <tibble [2 x 4]> <tibble [0 x 3]> <tibble>
##  9 <split [2108/234]> Fold09 <tibble [2 x 4]> <tibble [0 x 3]> <tibble>
## 10 <split [2108/234]> Fold10 <tibble [2 x 4]> <tibble [0 x 3]> <tibble>
```

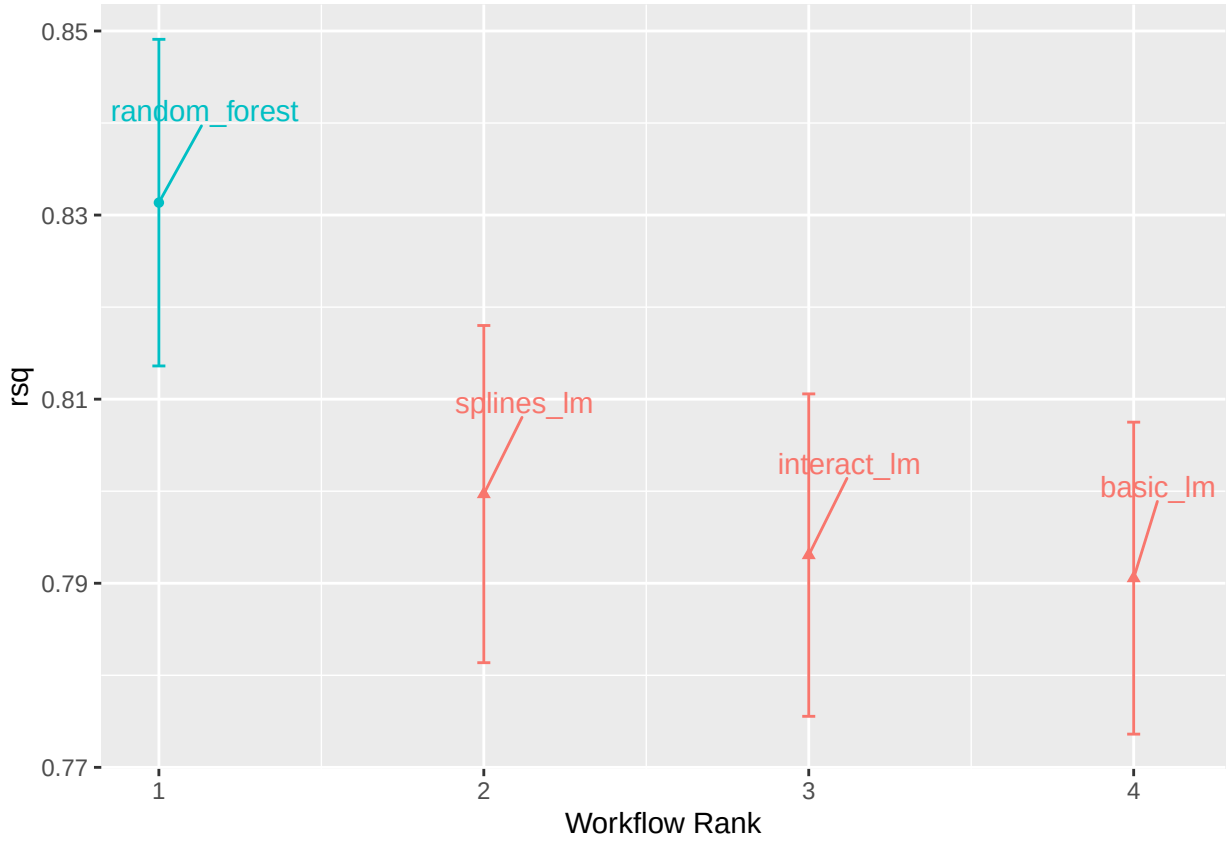
Adding the random forest model to to the workflow set:

```
four_models <- as_workflow_set(random_forest = rf_res) |>
  bind_rows(lm_models)

four_models
```

```
## # A workflow set/tibble: 4 x 4
##   wflow_id      info             option    result
##   <chr>         <list>           <list>    <list>
## 1 random_forest <tibble [1 x 4]> <opts[0]> <rsmp[+]>
## 2 basic_lm      <tibble [1 x 4]> <opts[2]> <rsmp[+]>
## 3 interact_lm   <tibble [1 x 4]> <opts[2]> <rsmp[+]>
## 4 splines_lm    <tibble [1 x 4]> <opts[2]> <rsmp[+]>
```

Plotting the confidence intervals for each model in the workflow set using `autoplot()`:

```
library(ggrepel)
autoplot(four_models, metric = "rsq") +
  geom_text_repel(aes(label = wflow_id), nudge_x = 1/8, nudge_y = 1/100) +
  theme(legend.position = "none")
```



In the plot of R^2 confidence intervals, we can see that the random forest model was the "best" among the four models.

Comparing resampled performance statistics

There are some resamples where performance across models tends to be low and others where it tends to be high. This is called *resample-to-resample* component of variation.

Demonstration of resample-to-resample variation for the linear models and random forest models used above:

```
rsq_indiv_estimates <- collect_metrics(four_models, summarize = FALSE) |>
  filter(.metric == "rsq")

rsq_indiv_estimates
```

```
## # A tibble: 40 x 8
##    wflow_id      .config        preproc model id    .metric .estimator .estimate
##    <chr>         <chr>          <chr>   <chr> <chr> <chr>   <chr>          <dbl>
```

```
##  1 random_forest Preprocessor1~ formula rand~ Fold~ rsq     standard       0.861
##  2 random_forest Preprocessor1~ formula rand~ Fold~ rsq     standard       0.861
##  3 random_forest Preprocessor1~ formula rand~ Fold~ rsq     standard       0.880
##  4 random_forest Preprocessor1~ formula rand~ Fold~ rsq     standard       0.818
##  5 random_forest Preprocessor1~ formula rand~ Fold~ rsq     standard       0.851
##  6 random_forest Preprocessor1~ formula rand~ Fold~ rsq     standard       0.822
##  7 random_forest Preprocessor1~ formula rand~ Fold~ rsq     standard       0.850
##  8 random_forest Preprocessor1~ formula rand~ Fold~ rsq     standard       0.784
##  9 random_forest Preprocessor1~ formula rand~ Fold~ rsq     standard       0.789
## 10 random_forest Preprocessor1~ formula rand~ Fold~ rsq     standard       0.795
## # i 30 more rows
```

```
rsq_wider <- rsq_indiv_estimates |>
  select(wflow_id, .estimate, id) |>
  pivot_wider(id_cols = "id", names_from = "wflow_id", values_from = .estimate)

rsq_wider
```

```
## # A tibble: 10 x 5
##    id     random_forest basic_lm interact_lm splines_lm
##    <chr>          <dbl>    <dbl>       <dbl>      <dbl>
##  1 Fold01         0.861    0.811       0.813      0.822
##  2 Fold02         0.861    0.804       0.810      0.817
##  3 Fold03         0.880    0.844       0.851      0.858
##  4 Fold04         0.818    0.778       0.782      0.782
##  5 Fold05         0.851    0.825       0.823      0.833
##  6 Fold06         0.822    0.787       0.781      0.801
##  7 Fold07         0.850    0.794       0.799      0.802
##  8 Fold08         0.784    0.778       0.784      0.786
##  9 Fold09         0.789    0.744       0.742      0.750
## 10 Fold10         0.795    0.741       0.744      0.746
```

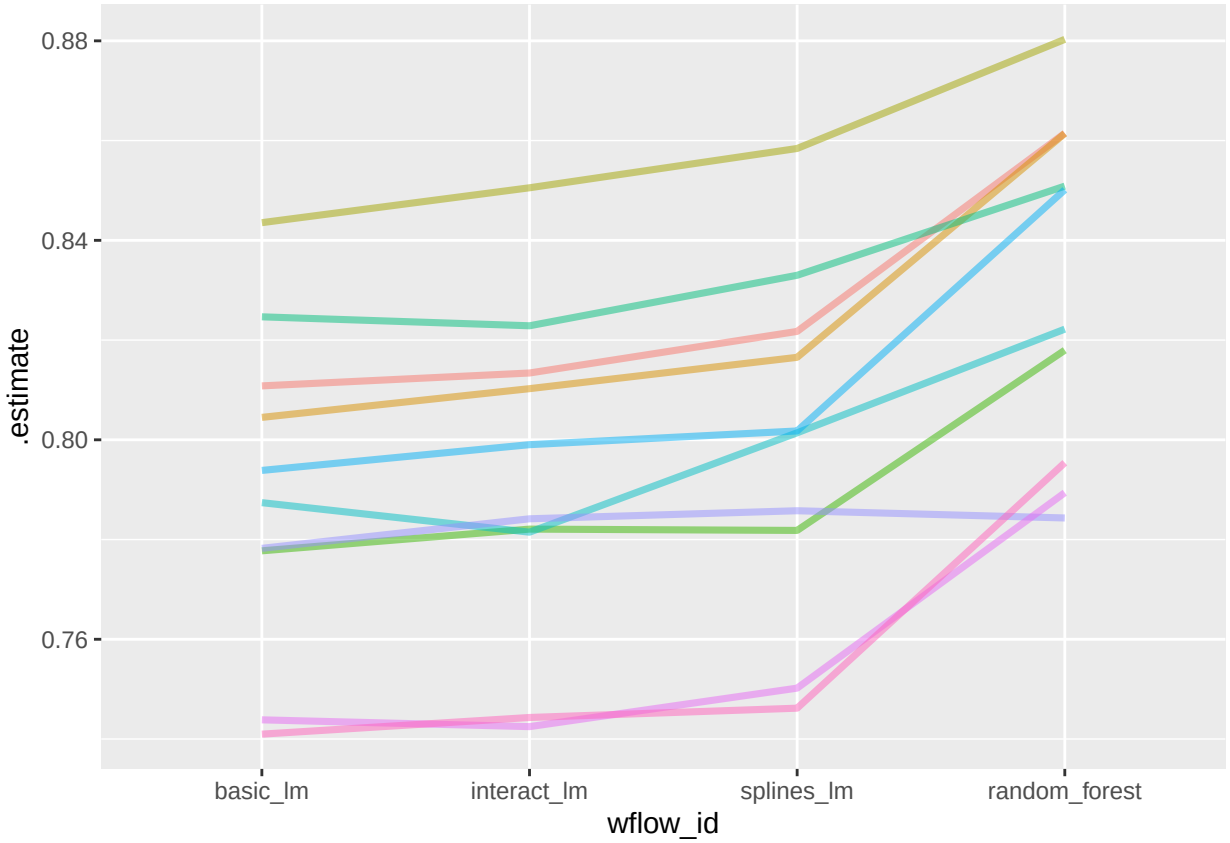
```
corrr::correlate(rsq_wider |> select(-id), quiet = TRUE)
```

```
## # A tibble: 4 x 5
##   term          random_forest basic_lm interact_lm splines_lm
##   <chr>                 <dbl>    <dbl>       <dbl>      <dbl>
## 1 random_forest        NA        0.887       0.888      0.889
## 2 basic_lm              0.887   NA           0.993      0.997
## 3 interact_lm           0.888    0.993      NA          0.987
## 4 splines_lm            0.889    0.997       0.987     NA
```

The correlations are high, and they indicate that across models, there are large within-resample correlations.

Plotting the R^2 statistics for each resample for each model:

```
rsq_indiv_estimates |>
  mutate(wflow_id = reorder(wflow_id, .estimate)) |>
  ggplot(aes(x = wflow_id, y = .estimate, group = id, color = id)) +
  geom_line(alpha = 0.5, linewidth = 1.25) +
  theme(legend.position = "none")
```



If the resample-to-resample effect was not real, there would not be any parallel lines. Running a statistical test for the correlations on `rsq_wider`:

```
rsq_wider |>
  with( cor.test(basic_lm, splines_lm)) |>
  tidy() |>
  select(estimate, starts_with("conf"))
```

```
## # A tibble: 1 x 3
##   estimate conf.low conf.high
##      <dbl>    <dbl>     <dbl>
## 1    0.997    0.987     0.999
```

This test shows the `estimate` of the correlation and the confidence intervals and that the within-resample correlation appears to be real.

Simple hypothesis testing methods

Comparing two models at a time using the differences in R^2 values as the outcome data in an ANOVA model:

```
compare_lm <-
  rsq_wider |>
  mutate(difference = splines_lm - basic_lm)

compare_lm
```

```
## # A tibble: 10 x 6
##    id     random_forest basic_lm interact_lm splines_lm difference
##    <chr>          <dbl>    <dbl>       <dbl>      <dbl>      <dbl>
##  1 Fold01         0.861    0.811       0.813      0.822    0.0109
##  2 Fold02         0.861    0.804       0.810      0.817    0.0120
##  3 Fold03         0.880    0.844       0.851      0.858    0.0149
##  4 Fold04         0.818    0.778       0.782      0.782    0.00409
##  5 Fold05         0.851    0.825       0.823      0.833    0.00834
##  6 Fold06         0.822    0.787       0.781      0.801    0.0140
##  7 Fold07         0.850    0.794       0.799      0.802    0.00792
##  8 Fold08         0.784    0.778       0.784      0.786    0.00754
##  9 Fold09         0.789    0.744       0.742      0.750    0.00636
## 10 Fold10         0.795    0.741       0.744      0.746    0.00521
```

```
lm(difference ~ 1, data = compare_lm) |>
  tidy(conf.int = TRUE) |>
  select(estimate, p.value, starts_with("conf"))
```

```
## # A tibble: 1 x 4
##   estimate   p.value conf.low conf.high
##      <dbl>     <dbl>    <dbl>     <dbl>
## 1  0.00913 0.0000256  0.00650    0.0118
```

Applying a paired t-test to `rsq_wider`:

```
rsq_wider |>
  with(t.test(splines_lm, basic_lm, paired = TRUE)) |>
  tidy() |>
  select(estimate, p.value, starts_with("conf"))
```

```
## # A tibble: 1 x 4
##   estimate   p.value conf.low conf.high
##      <dbl>     <dbl>    <dbl>     <dbl>
## 1  0.00913 0.0000256  0.00650    0.0118
```

Note that the p-value indicates a statistically significant signal. This shows that the collection of spline terms for the longitude and latitude appear to have an effect (the null hypothesis here states the the variations are random). However, the difference in R^2 is estimated at 0.91%. If our practical effect size were 2%, we might not consider these terms worth including in the model.

Bayesian methods

This is a more general approach to making formal comparisons using random effects.

The model used here is more complex than the ANOVA method, but the interpretation is more straightforward than the p-value approach used in ANOVA.

The Bayesian linear model makes the following assumptions:
- The residuals are assumed to be independent and follow a Gaussian distribution with zero mean and constant standard deviation.
- Prior distribution specifications are required.

From the given observed data and prior distribution specifications, the model parameters can be estimated. The combinations of the priors and the likelihood estimates give the final (aka posterior) distributions of the model.

A random intercept model A random intercept model will be used in the demonstration that follows. This is to ensure that the Bayesian ANOVA model adequately models the resamples.

In this model, the prior distribution is assumed to be symmetric, such as a bell-shaped curve (a standard normal or t-distribution).

Using the **tidyposterior** and **rstanarm** for model analysis:

```
library(tidyposterior)
library(rstanarm)
```

```
## Loading required package: Rcpp
```

```
##
## Attaching package: 'Rcpp'
```

```
## The following object is masked from 'package:rsample':
##
##     populate
```

```
## This is rstanarm version 2.32.1
```

```
## - See https://mc-stan.org/rstanarm/articles/priors for changes to default priors!
```

```
## - Default priors may change, so it's safest to specify priors, even if equivalent to the defaults.
```

```
## - For execution on a local, multicore CPU with excess RAM we recommend calling
```

```
##   options(mc.cores = parallel::detectCores())
```

```
# The rstanarm package creates copious amounts of output; those results are worth
# inspecting for potential issues. option `refresh = 0` can be used to eliminate the
# logging.

rsq_anova <-
  perf_mod(
    four_models,
    metric = "rsq",
    prior_intercept = rstanarm::student_t(df = 1),
    chains = 4,
    iter = 5000,
    seed = 1102
  )
```

```
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 3.4e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.34 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
```

```
## Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 1: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 1: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 1: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 2.43 seconds (Warm-up)
## Chain 1:                1.597 seconds (Sampling)
## Chain 1:                4.027 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.9e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.19 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 2: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 2: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 2: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 2: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 2: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 2: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 2: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 2: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 2: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 2: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 2: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 2.516 seconds (Warm-up)
## Chain 2:                1.493 seconds (Sampling)
## Chain 2:                4.009 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 2e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.2 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 3: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 3: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 3: Iteration: 1500 / 5000 [ 30%]  (Warmup)
```

```
## Chain 3: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 3: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 3: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 3: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 3: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 3: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 3: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 3: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 2.699 seconds (Warm-up)
## Chain 3:                1.821 seconds (Sampling)
## Chain 3:                4.52 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 2e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.2 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 4: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 4: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 4: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 4: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 4: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 4: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 4: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 4: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 4: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 4: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 4: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 3.074 seconds (Warm-up)
## Chain 4:                2.203 seconds (Sampling)
## Chain 4:                5.277 seconds (Total)
## Chain 4:
```

The resulting object has information on the resampling process as well as the Stan object embedded within (in an element called `stan`). The **tidyposterior** packaged has a `tidy()` method that extracts these posterior distributions into a tibble:

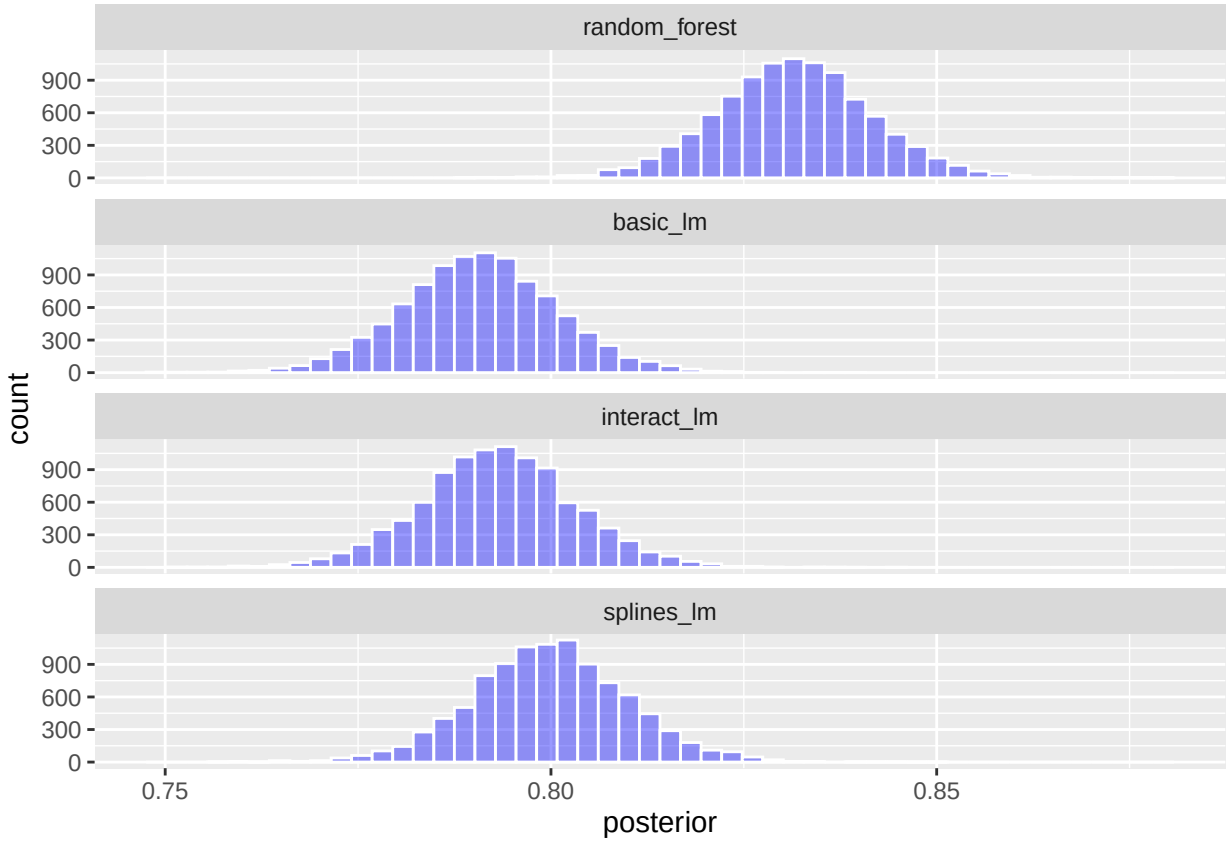
```
model_post <-
  rsq_anova |>
  # Take a random sample from the posterior distribution so set the seed again to be
  # reproducible.
  tidy(seed = 1103)

glimpse(model_post)
```

```
## Rows: 40,000
## Columns: 2
```

```
## $ model     <chr> "random_forest", "basic_lm", "interact_lm", "splines_lm", "r~
## $ posterior <dbl> 0.8196439, 0.7772962, 0.7837052, 0.7862448, 0.8204723, 0.778~
```

Visualizing the four posterior distributions:

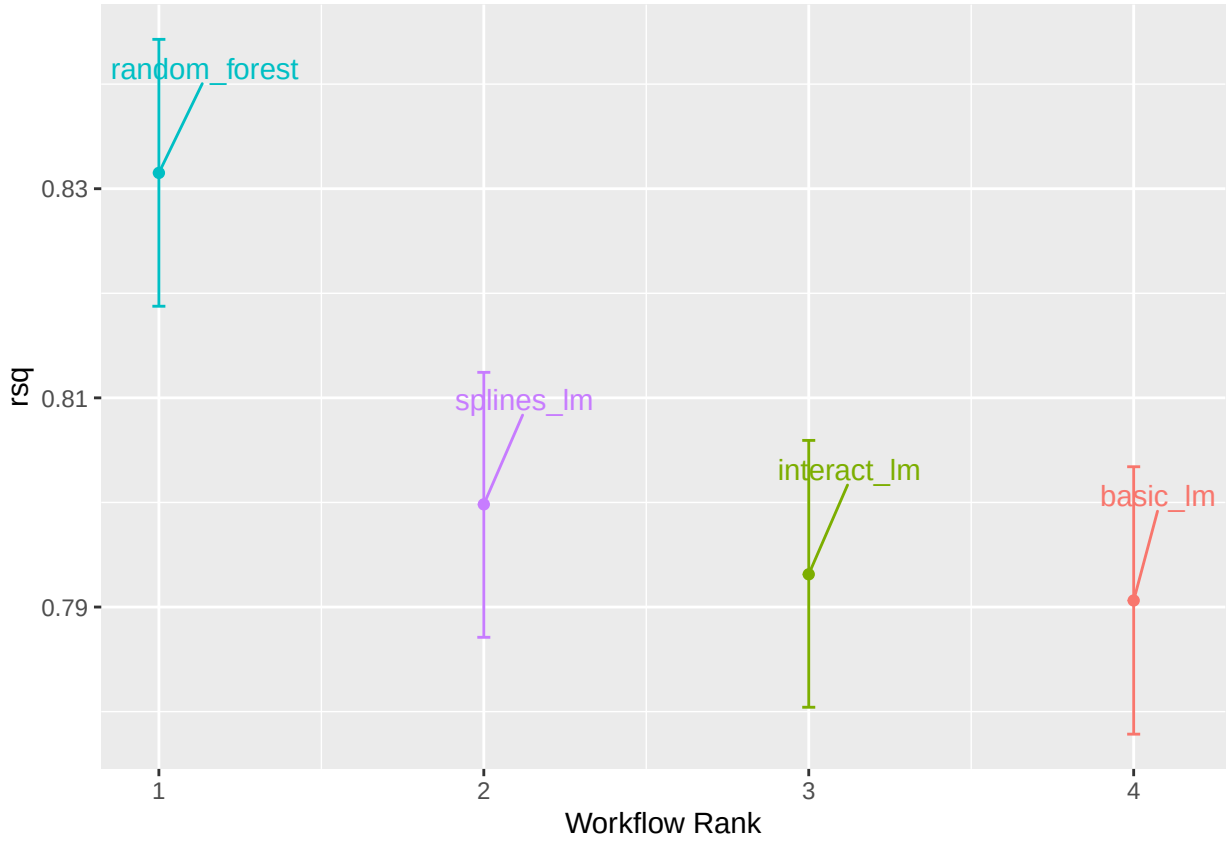
```
model_post |>
  mutate(model = forcats::fct_inorder(model)) |>
  ggplot(aes(x = posterior)) +
  geom_histogram(bins = 50, color = "white", fill = "blue", alpha = 0.4) +
  facet_wrap(~ model, ncol = 1)
```



These histograms describe the estimated probability distributions of the mean R^2 value for each model.

There is also a basic `autoplot()` method for the model results:

```
autoplot(rsq_anova) +
  geom_text_repel(aes(label = workflow), nudge_x = 1/8, nudge_y = 1/100) +
  theme(legend.position = "none")
```



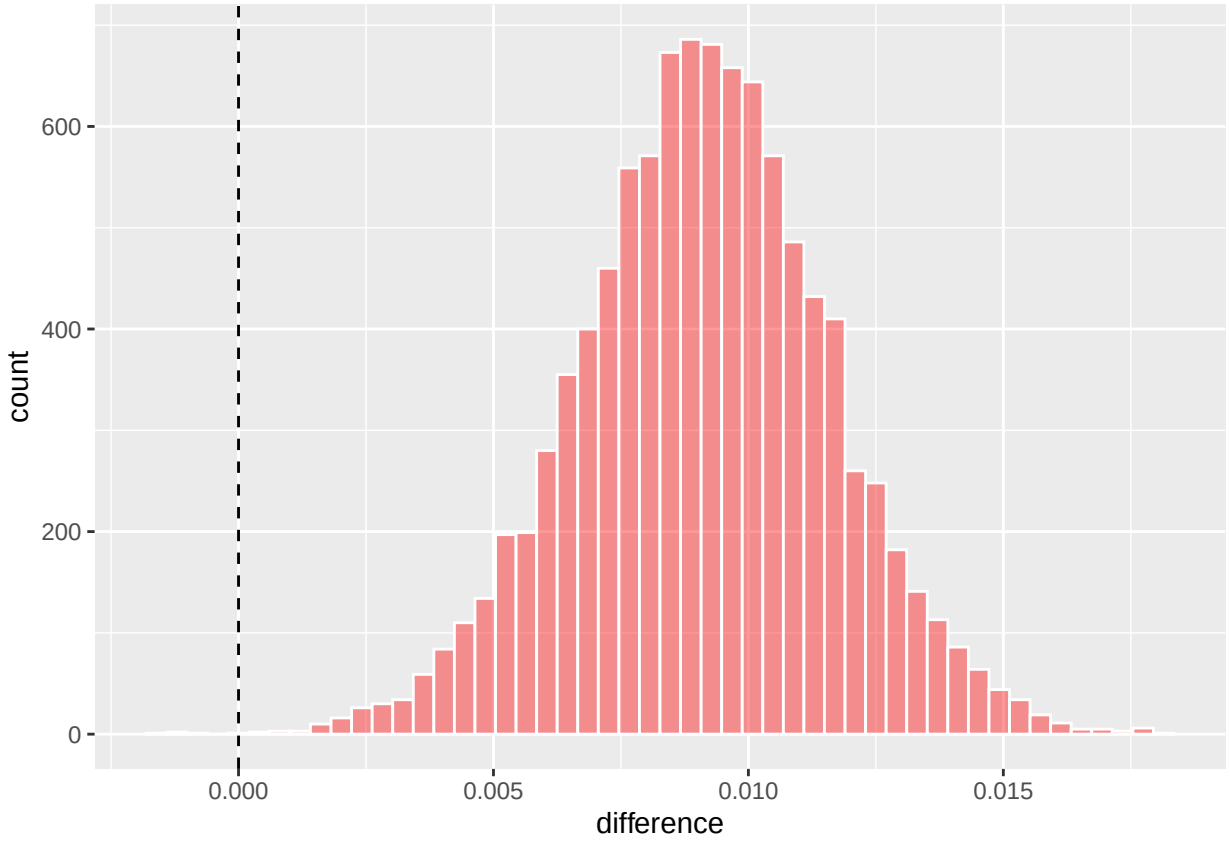
One wonderful aspect of using resampling with Bayesian models is that, once we have the posteriors for parameters, it is trivial to get the posterior distributions for combinations of the parameters.

Comparing the two linear models:

```
rqs_diff <-
  contrast_models(rsq_anova,
                  list_1 = "splines_lm",
                  list_2 = "basic_lm",
                  seed = 1104)
```

Visualizing the comparison results:

```
rqs_diff |>
  as_tibble() |>
  ggplot(aes(x = difference)) +
  geom_vline(xintercept = 0, lty =2) +
  geom_histogram(bins = 50, color = "white", fill = "red", alpha = 0.4)
```



The posterior shows that the center of the distribution is greater than zero – indicating that the model with splines typically had larger values – but does overlap with zero to a degree.

Computing the mean of the distribution as well as the credible intervals using the `summary()` method:

```
summary(rqs_diff) |>
  select(-starts_with("pract"))
```

```
## # A tibble: 1 x 6
##   contrast               probability    mean   lower  upper  size
##   <chr>                        <dbl>   <dbl>   <dbl>  <dbl> <dbl>
## 1 splines_lm vs basic_lm        1.00 0.00912 0.00500 0.0132     0
```

The `probability` column reflects the proportion of the posterior that is greater than zero. This is the probability that the positive difference is real.

The value is not close to zero, providing a strong case for statistical significance, i.e., the idea that statistically the actual difference is not zero.

We can also compute the probability of being practically significant. In Bayesian analysis, this is a ROPE estimate (Region of Practical Equivalence). This can be estimated using the `size` option to the `summary` function:

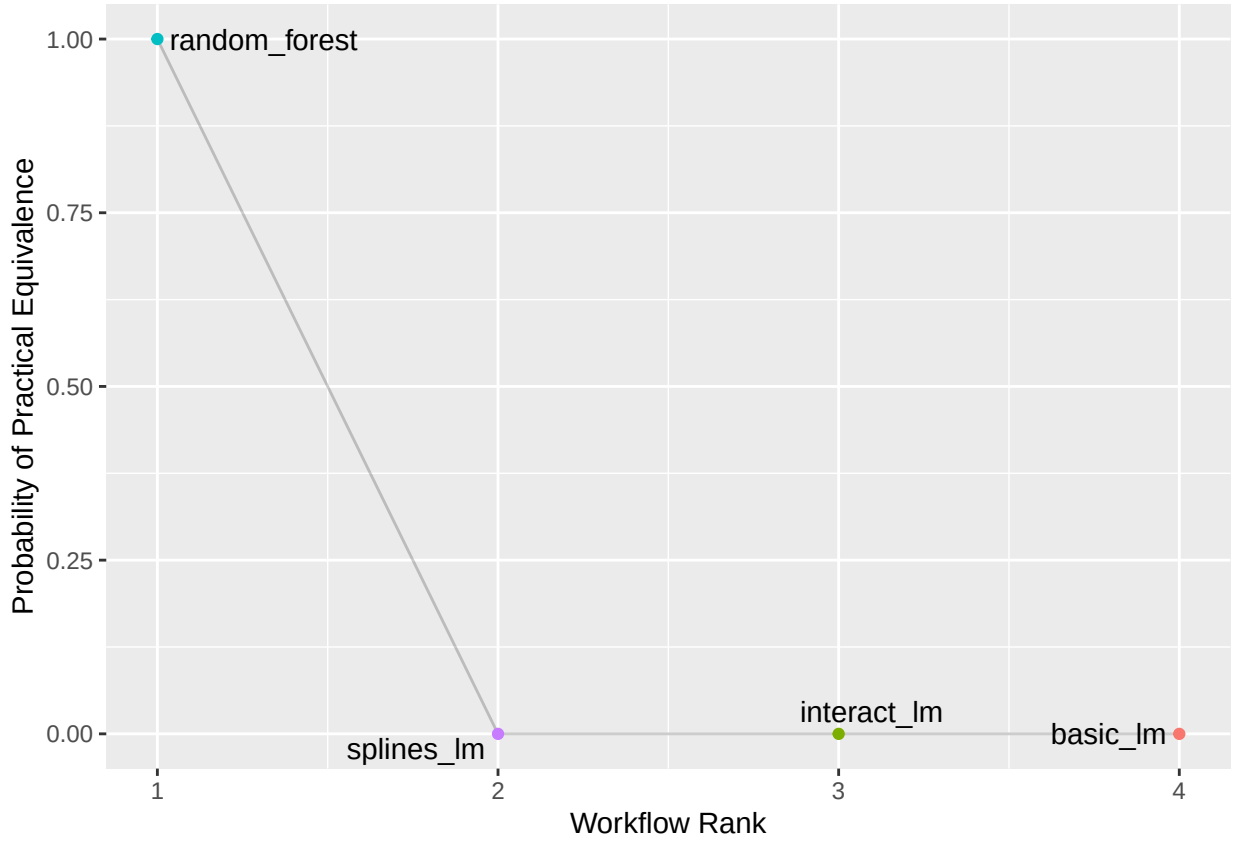
```
summary(rqs_diff, size = 0.02) |>
  select(contrast, starts_with("pract"))
```

```
## # A tibble: 1 x 4
```

```
##   contrast               pract_neg pract_equiv pract_pos
##   <chr>                      <dbl>       <dbl>     <dbl>
## 1 splines_lm vs basic_lm         0           1         0
```

Using `autoplot()` to show the `pract_equiv` results that compare each workflow to the current best (the random forest model, in this case):

```
autoplot(rsq_anova, type = "ROPE", size = 0.02) +
  geom_text_repel(aes(label = workflow)) +
  theme(legend.position = "none")
```



The effect of the amount of resampling Visualizing the effect of the number of resamples on these types of Bayesian comparisons:

```
ggplot(
  intervals,
  aes(x = resamples, y = mean)
) +
  geom_path() +
  geom_robbon(aes(ymin = lower, ymax = upper), fill = "red", alpha = 0.1) +
  labs(x = "Number of Resamples (repeated 10-fold cross-validation)")
```