Notas de `tidymodels`

Parte 2: `parsnip`

Emmanuel Alcalá

07 April, 2022

Capítulo 6 de `tmwr`

Flujo de trabajo básico

`recipe()` -> `workflow()` -> `fit()` -> `predict()`

Previo al uso de alguna función de modelamiento, el modelo ya se debió haber escogido. Los datos se preprocesan con un modelo, o varios, en mente. Las funciones de `parsnip` ejecutan dichos modelos, en datos preprocesados.

Funciones principales

`fit()`

`predict()`

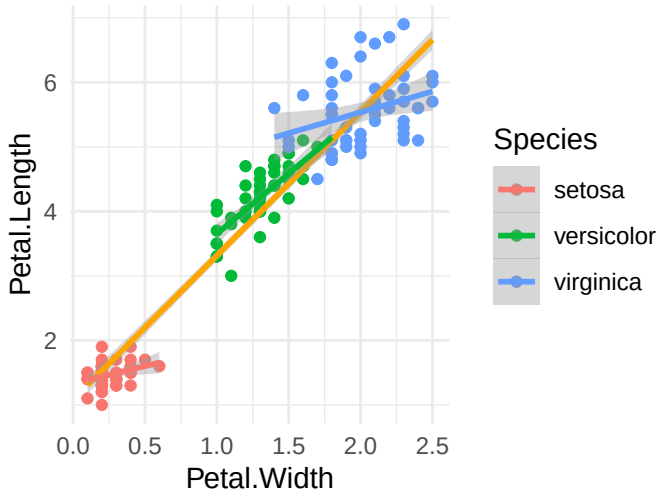
Ejemplo: ANOVA

Los pasos normales de una ANOVA son

1. Hacer un modelo con múltiples predicciones.
2. Hacer un segundo modelo con interacciones.
3. Evaluar qué modelo es mejor, por ejemplo `anova(mod1, mod2)`, y escoger el modelo más complejo si el valor p es significativo.

Usaremos un enfoque tidy para esto. Primero, visualizar

```
iris %>%
  ggplot(aes(Petal.Width, Petal.Length)) +
  geom_point(aes(color = Species)) +
  geom_smooth(
    method = lm, color = "orange",
    formula = "y~x"
  ) +
  geom_smooth(
    method = lm, aes(color = Species),
    formula = "y~x"
  ) +
  theme_minimal()
```



Preprocesar:

```
set.seed(123)
iris_split <- initial_split(iris, strata = Species, prop = 0.8)
iris_train <- training(iris_split)
iris_test <- testing(iris_split)
# receta de modelo sin interacciones
rec_normal <-
  recipe(Petal.Length ~ Petal.Width + Species,
    data = iris_train
  ) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_center(all_numeric_predictors())

# receta de modelo con interacciones
rec_interaction <-
  rec_normal %>%
  step_interact(~ Petal.Width:starts_with("Species"))
```

Ahora especificamos un modelo lineal

```
iris_model_lm <-
  linear_reg() %>%
  set_engine("lm") %>% # aquí se especifica qué engine usar; usaremos stats::lm
  set_mode("regression")
```

Ahora añadimos el `workflow`

```
# modelo sin interaccion
iris_wf <-
  workflow() %>%
  add_model(iris_model_lm) %>%
  add_recipe(rec_normal)

# modelo con interaccion
iris_wf_interaction <-
  iris_wf %>%
  update_recipe(rec_interaction)
```

Ahora haremos el ajuste. Normalmente se usaría `fit()` sobre el conjunto de entrenamiento o de prueba, pero

usaremos `last_fit()` para simplificar el proceso. Según la documentación, esto asegura que el modelo se ejecuta en el conjunto entero de datos y se evalúa y retorna el mejor modelo.

```
iris_normal_lf <-
  last_fit(iris_wf,
    split = iris_split
  )

iris_inter_lf <-
  last_fit(iris_wf_interaction,
    split = iris_split
  )
```

Ahora ejecutamos el ANOVA, pero debmos extraer el modelo lineal primero

```
normalmodel <- iris_normal_lf %>% extract_fit_engine()
intermodel <- iris_inter_lf %>% extract_fit_engine()

anova(normalmodel, intermodel) %>% tidy()
```

```
## # A tibble: 2 x 6
##   res.df   rss    df sumsq statistic  p.value
##    <dbl> <dbl> <dbl> <dbl>     <dbl>    <dbl>
## 1    116  17.7    NA NA        NA    NA
## 2    114  15.9     2  1.73      6.17  0.00285
```

Si lo comparamos con la forma tradicional, nos retorna lo mismo

```
mod1 <- lm(Petal.Length ~ Petal.Width * Species, data = iris)
# summary(mod1)

mod2 <- lm(Petal.Length ~ Petal.Width + Species, data = iris)
# summary(mod2)

anova(mod1, mod2) %>% tidy()
```

```
## # A tibble: 2 x 6
##   res.df   rss    df sumsq statistic   p.value
##    <dbl> <dbl> <dbl> <dbl>     <dbl>     <dbl>
## 1    144  18.8    NA NA        NA    NA
## 2    146  20.8    -2 -2.02      7.72  0.000653
```