ETC3250/5250 Tutorial 2 Instructions

Fitting nonlinear regression models

prepared by Professor Di Cook

Week 2

🎯 Objective

The objectives for this week are to

- practice plotting of data and models to help understand the fit.
- understand the purpose of breaking the model fitting into pre-processing with `recipes`, model specification with `parsnip`, and `yardstick` to compute error.
- develop your ability to support your choice of model from among several.
- understand why to use orthogonal polynomials, for polynomial regression.

🔧 Preparation

Make sure you have these packages installed:

```
install.packages(c("tidyverse", "kableExtra", "tidymodels", "GGally", "ISLR", "here",
"rsample", "mgcv", "gratia", patchwork"))
```

📖 Reading and thinking

- Read and work through https://emilhvitfeldt.github.io/ISLR-tidymodels-labs/moving-beyond-linearity.html (https://emilhvitfeldt.github.io/ISLR-tidymodels-labs/moving-beyond-linearity.html).
- Think about the wages example and the relationship between wage and age. Read about lurking variables here (https://courses.lumenlearning.com/wmopen-concepts-statistics/chapter/causation-and-lurking-variables-1-of-2/).

👋 Getting started

If you are in a zoom tutorial, say hello in the chat. If in person, do say hello to your tutor and to your neighbours.

⚙️ Exercise

```
library(tidyverse)
library(ISLR)
library(tidymodels)
library(GGally)
library(mgcv)
library(patchwork)
```

1. Fitting NRC rankings

In 2010, the National Research Council released rankings for all doctorate programs in the USA (https://en.wikipedia.org/wiki/United_States_National_Research_Council_rankings (https://en.wikipedia.org/wiki/United_States_National_Research_Council_rankings)). The data was initially released and then only available for a fee. I managed to get a copy during the free period, and this is the data that we will use for this exercise. There hasn't been another set of rankings publicly released since then, and I don't know why. Only the rankings and statistics for Statistics programs are included in this data set.

Your job is to answer the question: "How is R Ranking related to rankings on research, student support and diversity?" using the 5th percentile for each of these quantities. Fit your best model, try using splines, and justify your choices.

a.

Read the data. Rename the columns containing the variables of interest to `rank`, `research`, `student` and `diversity`). Using recipes, split the data into 2/3 training and 1/3 test sets.

```
# Read data
nrc <- read_csv(here::here("data/nrc.csv")) %>%
  mutate(rank = R.Rankings.5th.Percentile,
         research = Research.Activity.5th.Percentile,
         student = Student.Support.Outcomes.5th.Percentile,
         diversity = Diversity.5th.Percentile) %>%
  select(rank, research, student, diversity)

# Split the data into training and test
set.seed(22)
train_test_split <- initial_split(nrc, prop = 2/3)

nrc_train <- training(train_test_split)
nrc_test <- testing(train_test_split)
```

b.

Make response vs predictor plots and predictor vs predictor plots of the training data. Explain the relationships, outliers, and how this would affect the modeling, and what you would expect the results of the modeling to be. (Hint: use `GGally::ggduo()`)

c.

Make a scatterplot matrix of the predictors, and discuss any potential problems like multicollinearity. (Hint: use `GGally::ggpairs()`)

d.

Fit a linear model. Report estimates, model fit statistics, and the test RMSE. Make the plots to visually assess the fit, including observed vs fitted, residual vs fitted, histogram of resisuals, normal probability plot of residuals, and the fitted vs each predictor.

```
# Fit a linear model to the original data
lm_mod <-
  linear_reg() %>%
  set_engine("lm")

nrc_lm_fit <-
  lm_mod %>%
  fit(rank ~ ., data = nrc_train)

# Summarise model and check the fit
tidy(nrc_lm_fit)
glance(nrc_lm_fit)

# broom::augment() is an easy way to get predicted
# values and residuals
nrc_lm_train_pred <- augment(nrc_lm_fit, nrc_train)
nrc_lm_test_pred <- augment(nrc_lm_fit, nrc_test)
metrics(nrc_lm_test_pred, truth = rank,
        estimate = .pred)

# Plot fitted and residuals of training data
p_f <- ggplot(nrc_lm_train_pred) +
  geom_point(aes(x = .pred, y = rank))
p_e <- ggplot(nrc_lm_train_pred) +
  geom_point(aes(x = .pred, y = .resid))
p_h <- ggplot(nrc_lm_train_pred, aes(x = .resid)) +
  geom_histogram(binwidth=2.5, colour="white") +
  geom_density(aes(y=..count..), bw = 2, colour="orange")
p_q <- ggplot(nrc_lm_train_pred, aes(sample = .resid)) +
  stat_qq() +
  stat_qq_line() +
  xlab("theoretical") + ylab("sample")
p_f + p_e + p_h + p_q
```

```
  # Plot the fitted model against each predictor
p1 <- ggplot(nrc_lm_train_pred) +
  geom_point(aes(x = research, y = rank)) +
  geom_point(aes(x = research, y = .pred),
             colour="blue")
p2 <- ggplot(nrc_lm_train_pred) +
  geom_point(aes(x = student, y = rank)) +
  geom_point(aes(x = student, y = .pred),
             colour="blue")
p3 <- ggplot(nrc_lm_train_pred) +
  geom_point(aes(x = diversity, y = rank)) +
  geom_point(aes(x = diversity, y = .pred),
             colour="blue")
p1 + p2 + p3
```

e.

Fit a splines model. Report estimates, model fit statistics, and the test RMSE. Make the plots to visually assess the fit, including observed vs fitted, residual vs fitted, histogram of resisuals, normal probability plot of residuals, and the fitted vs each predictor.

```
nrc_ns_rec <-
  recipe(rank ~ research + student + diversity,
         data = nrc_train) %>%
  step_ns(research, student, diversity, deg_free = 3)

# Define workflow
nrc_ns_wf <- workflow() %>%
  add_model(lm_mod) %>%
  add_recipe(nrc_ns_rec)

# Fit a linear model to the transformed data
nrc_ns_fit <- fit(nrc_ns_wf, data=nrc_train)

# Summarise model and check the fit
tidy(nrc_ns_fit)
glance(nrc_ns_fit)

# augment for splines only extracts the fitted
# values and not residuals
nrc_ns_train_pred <- augment(nrc_ns_fit, nrc_train) %>%
  mutate(.resid = rank - .pred)
nrc_ns_test_pred <- augment(nrc_ns_fit, nrc_test) %>%
  mutate(.resid = rank - .pred)
metrics(nrc_ns_test_pred, truth = rank,
        estimate = .pred)

# Plot fitted and residuals of training data
ps_f <- ggplot(nrc_ns_train_pred) +
  geom_point(aes(x = .pred, y = rank))
ps_e <- ggplot(nrc_ns_train_pred) +
  geom_point(aes(x = .pred, y = .resid))
ps_h <- ggplot(nrc_ns_train_pred, aes(x = .resid)) +
  geom_histogram(binwidth=2.5, colour="white") +
  geom_density(aes(y=..count..), bw = 2, colour="orange")
ps_q <- ggplot(nrc_ns_train_pred, aes(sample = .resid)) +
  stat_qq() +
  stat_qq_line() +
  xlab("theoretical") + ylab("sample")
ps_f + ps_e + ps_h + ps_q
```

```
# Plot the fitted model against each predictor
ps1 <- ggplot(nrc_ns_train_pred) +
  geom_point(aes(x = research, y = rank)) +
  geom_point(aes(x = research, y = .pred),
             colour="blue")
ps2 <- ggplot(nrc_ns_train_pred) +
  geom_point(aes(x = student, y = rank)) +
  geom_point(aes(x = student, y = .pred),
             colour="blue")
ps3 <- ggplot(nrc_ns_train_pred) +
  geom_point(aes(x = diversity, y = rank)) +
  geom_point(aes(x = diversity, y = .pred),
             colour="blue")
ps1 + ps2 + ps3
```

f.

Fit a GAM model. Report estimates, model fit statistics, and the test RMSE. Make the plots to visually assess the fit, including observed vs fitted, residual vs fitted, histogram of resisuals, normal probability plot of residuals, and the fitted vs each predictor.

```
# mgcv::gam is not yet integrated with tidymodels
nrc_gam <-
  mgcv::gam(rank ~
        s(research) +
        s(student) +
        s(diversity), data = nrc_train)

# Model fit
tidy(nrc_gam)
glance(nrc_gam)

nrc_gam_train_pred <- augment(nrc_gam, nrc_train) %>%
  rename(.pred = .fitted)
nrc_gam_test_pred <- augment(nrc_gam,
                             newdata=nrc_test) %>%
  rename(.pred = .fitted) %>%
  mutate(.resid = rank - .pred)
metrics(nrc_gam_test_pred, truth = rank,
        estimate = .pred)

draw(nrc_gam, residuals = TRUE)
appraise(nrc_gam)
```

```
# Plot the fitted model against each predictor
pg1 <- ggplot(nrc_gam_train_pred) +
  geom_point(aes(x = research, y = rank)) +
  geom_point(aes(x = research, y = .pred),
             colour="blue")
pg2 <- ggplot(nrc_gam_train_pred) +
  geom_point(aes(x = student, y = rank)) +
  geom_point(aes(x = student, y = .pred),
             colour="blue")
pg3 <- ggplot(nrc_gam_train_pred) +
  geom_point(aes(x = diversity, y = rank)) +
  geom_point(aes(x = diversity, y = .pred),
             colour="blue")
pg1 + pg2 + pg3
```

g.

Based on the model fit statistics, and test mse, and plots, which model is best? How do the predictors relate to the rank? Are all the predictors important for predicting rank? Could the model be simplified?

2. ### 2. (IF TIME ALLOWS) Lurking variables

Lurking variables

a.

The wages data poses an interesting analytical dilemma. There appear to be two wage clusters, one small group with relatively consistent high wage, and the other big group with lower and varied wages. Make a histogram or a density plot of wage to check this more carefully.

b.

What do you think might have caused this? Check the relationship between wage, and other variables such as `jobclass`. Do any of the other variables provided in the data explain the clustering?

c.

The textbook (Section 7.8.1 p.315) includes an analysis where a separate logistic model where a binary response ($\leq 250, > 250$) is used. Why doesn't this make sense?

An alternative approach is to treat this as a "lurking variable", let's call it "manager", create a new predictor and include this in the model. Alternatively, we could treat the high group as outliers, and exclude them from the analysis. The argument for these values are likely due to some unobserved factor, and their presence affects the ability to accurately model the rest of the data.

3. (IF TIME ALLOWS) Explore the polynomial model fitting

a.

This builds from the polynomial model fit for the Wage data, using variables wage and age, in Figure 7.1.

The function `poly` is a convenient way to generate a fourth-degree polynomial. By default it uses "orthogonal polynomials".

```
# Old style way to fit because recipes doesn't seem to all to choose orthogonal polynomials
fit1 <- lm(wage ~ poly(age, 4), data=Wage)
tidy(fit1)
```

b.

We can request that "raw" polynomials are generated instead, with the `raw=TRUE` argument.

```
# Old style way to fit because recipes doesn't seem to all to choose orthogonal polynomials
fit2 <- lm(wage ~ poly(age, 4, raw=TRUE), data=Wage)
tidy(fit2)
```

c.

The coefficients are different, but effectively the fit is the same, which can be seen by plotting the fitted values from the two models.

```
wage_fit <- Wage %>%
  mutate(yhat1 = predict(fit1, Wage),
         yhat2 = predict(fit2, Wage))
ggplot(wage_fit, aes(x=yhat1, y=yhat2)) +
  geom_point() + theme(aspect.ratio = 1)
```

d.

To examine the differences between orthonormal polynomials and "raw" polynomials, we can make scatterplot matrices of the two sets of polynomials.

```
p_orth <- as_tibble(poly(Wage$age, 4))
ggscatmat(p_orth)
p_raw <- as_tibble(poly(Wage$age, 4, raw=TRUE))
ggscatmat(p_raw)
```

e.

Think about: What is the benefit of using orthonomal polynomials?

💬 Class discussion exercises, part of the wrap up

Why do you think polynomials and splines considered to be part of `recipes` rather than the model fitting?

🛑 Wrapping up

Talk to your tutor about what you think you learned today, what was easy, what was fun, what you found hard.