Notes on Ch3: A Review of R Modeling Fundamentals

N_Lim

2025-06-24

Fundamentals

R is based on S Language. Chambers and Hastie (of the ISLR fame?) published the "White Book" on data analysis on S.

Example

Exeprimental data on the relationship between the ambient temp and cricket chirp rate (per minute) on two species: *O. exclamationis* and *O. niveus*. Data frame `crickets` has 31 data points.

```
knitr::opts_chunk$set(eval = FALSE)

library(tidyverse)
```

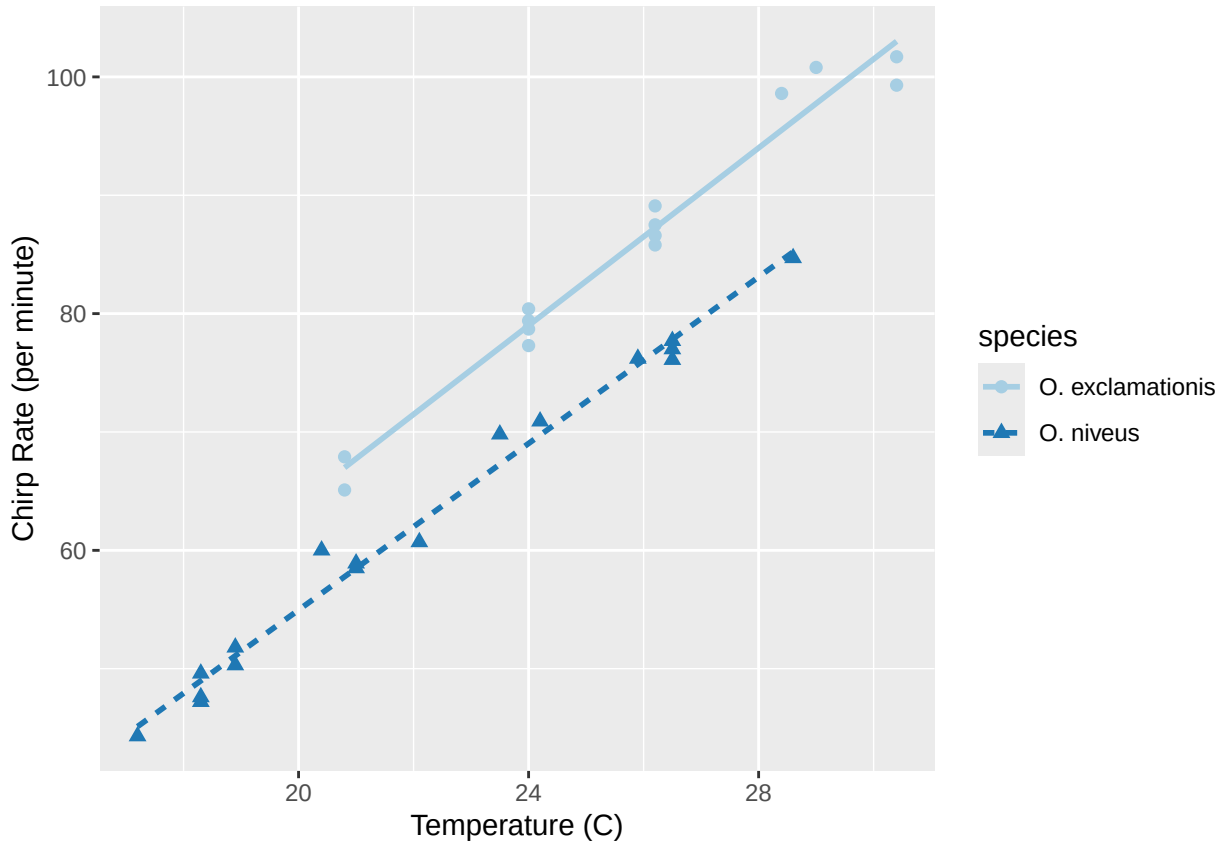
```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.2     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
data(crickets, package = "modeldata")
names(crickets)
```

```
## [1] "species" "temp"    "rate"
```

```
ggplot(
  data = crickets,
  aes(x = temp, y = rate, color = species, pch = species, lty = species)
) +
  geom_point(size = 2) +
  geom_smooth(method = lm, se = FALSE, alpha = 0.5) +
  scale_color_brewer(palette = "Paired") +
  labs(x = "Temperature (C)", y = "Chirp Rate (per minute)")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Data exhibit fairly linear trends for each species. O. exclamationis seems to chirp more.

Possible null hypotheses: - Temperature has no effect on the chirp rate. - There are no differences between the chirp rates of speices

To fit an ordinary linear model in R, we use the `lm()` function. In this function, we need a model formula in the form of `rate ~ temp` which means that the chirp rate is the outcome (the one on the left of the tilde sign), and the temp is the predictor.

Suppose the time of the day is important in this data. The formula becomes `rate ~ temp + time` which is a symbolic representation of the linear model. This does not imply that we add temp and time together. This simply means that each of the "effects" adds or contributes to the outcome.

The second formula creates a model with different y-intercepts for each species.

We can specify how the "effects" interact with each other: - `rate ~ temp + species + temp:species` - `rate ~ (temp + species)^2` - `rate ~ temp * species`

If you need to add a mathematical(i.e., non-symbolic) funciton in the formula, the identity function`I` can be used. for example, if we need the temp to be in degrees C instead of degrees F in order for the formula to work, we could type `rate ~ I((temp * 9/5) + 32)`. There are other configurations...

For the chirping crickets model with a two-way interaction model, the following model can be used:

```
interaction_fit <- lm(rate ~ (temp + species)^2, data = crickets)

# Print a short summary of the model:
interaction_fit
```

plotting:

```r
# place 2 plots next to each other
par(mfrow = c(1, 2))

# show the residuals vs predicted values
plot(interaction_fit, which = 1)

# quantile plot on the residuals
plot(interaction_fit, which = 2)
```

To assess if the inclusion of the interaction term is necessary, we can use the `anova()` method:

```r
# fit a reduced model
main_effect_fit <- lm(rate ~ temp + species, data = crickets)

# Compare the two
anova(main_effect_fit, interaction_fit)
```

- This test's p-value is 0.25.
- This implies a lack of evidence against the null hypo (that the interaction term is not needed)
- Residual plots should be reassess to make sure that our theoretical assumptions are valid enough to trust the p-values produced by the model.
- The `summary()` method can be used t inspect the "parts" i.e. the coeffecients, standard errors, and p-values.

```r
summary(main_effect_fit)
```

according to the result: - every 1 deg increase in temp increases the chirp rate by 3.6. - o. niveus has 10 chirps *fewer* per minute than o. exclamationis. - the species effect is associated with a very small p-val (the observed effect could not have happened due to chance alone). - the observed relationship is only sensible above 17.2 deg – extrapolating beyond the "applicable range" of the temperature values – is a bad idea.

Making predictions from the model: - use values that are within the applicable range i.e. from 15 to 20 degC.

```r
new_values = data.frame(species = "O. exclamationis", temp = 15:20)
predict(main_effect_fit, new_values)
```

What does the formula do?

- defines the columns the the model uses.
- the formula is used to encode the columns into an appropriate format
- the roles of the columns are defined by the formula.

For example, when we consider the formula `(temp + species)^2`, we can draw the following ideas: - there are two predictors - the model should contain their main effects and interactions. - species is a "factor" or category so there should be indicator variable columns (more of this on chap 8)

Why tidiness is important for modeling?

- for consistency
- to avoid errors and confusion coming from mismatched options

Wickham's Design-for-Humans rubric specifies the following goals of tidymodels design: - OOP-cabpability - has sensible defaults - capable of deriving the defaults from the data - functions should take the data structures that users have as opposed to the data structures that the developers had in mind. (Example: many models in Python require the data to be in matrix form, but most human data are not.)

Example showing the power of tidy functions:

```r
library(purrr)
# traditional method
corr_res <- map(mtcars %>% select(-mpg), cor.test, y = mtcars$mpg)
corr_res[[1]]
```

Using `tidy()` method:

```r
tidy(corr_res[[1]])
```

Although the results can be stacked and added to a ggplot function like so,

```r
corr_res %>%
  # convert each to a tidy format and stack the data frames
  map_dfr(tidy, .id = "predictor") %>%
  ggplot(aes(x = fct_reorder(predictor, estimate))) +
  geom_point(aes(y = estimate)) +
  geom_errorbar(aes(ymin = conf.low, ymax = conf.high), width = 0.1) +
  labs(x = NULL, y = "Correlation with mpg")
```

the "acrobatics" to pull this off is more complicated and more prone to error.

Combining base R models and the tidyverse

- other r packages can be used in conjunction with the tidyverse, especially with dplyr, purrr, and tidyr.

Example: fitting separate models for each cricket species

```r
split_by_species <- crickets %>%
  group_nest(species)

split_by_species
```

The `data` column contains the rate and temp columns. We can use the `purrr:map()` function to create individual models for each speices:

```r
model_by_species <- split_by_species %>%
  mutate(model = map(data, ~lm(rate ~ temp, data = .x)))

model_by_species
```

To collect the coefficients of these models, we can use `broom::tidy()`:

```r
model_by_species %>%
  mutate(coef = map(model, tidy)) %>%
  select(species, coef) %>%
  unnest(cols = c(coef))
```

Very tidy indeed!

The tidymodels metapackage

- `tidymodels` package loads a core set of tidyverse and tidymodels packages

```r
library(tidymodels)
```

How to handle naming conflicts in R packages: - explicitly speciry the package you need. Ex. `stats::filter()` - use the `conflicted` package (see code below) - use `tidyemodels_prefer()` (see code below) - this captures most of the common naming conflicts that we might encounter.

```r
library(conflicted)
conflict_prefer("filter", winner = "dplyr")
```

```r
tidymodels_prefer(quiet = FALSE)
```