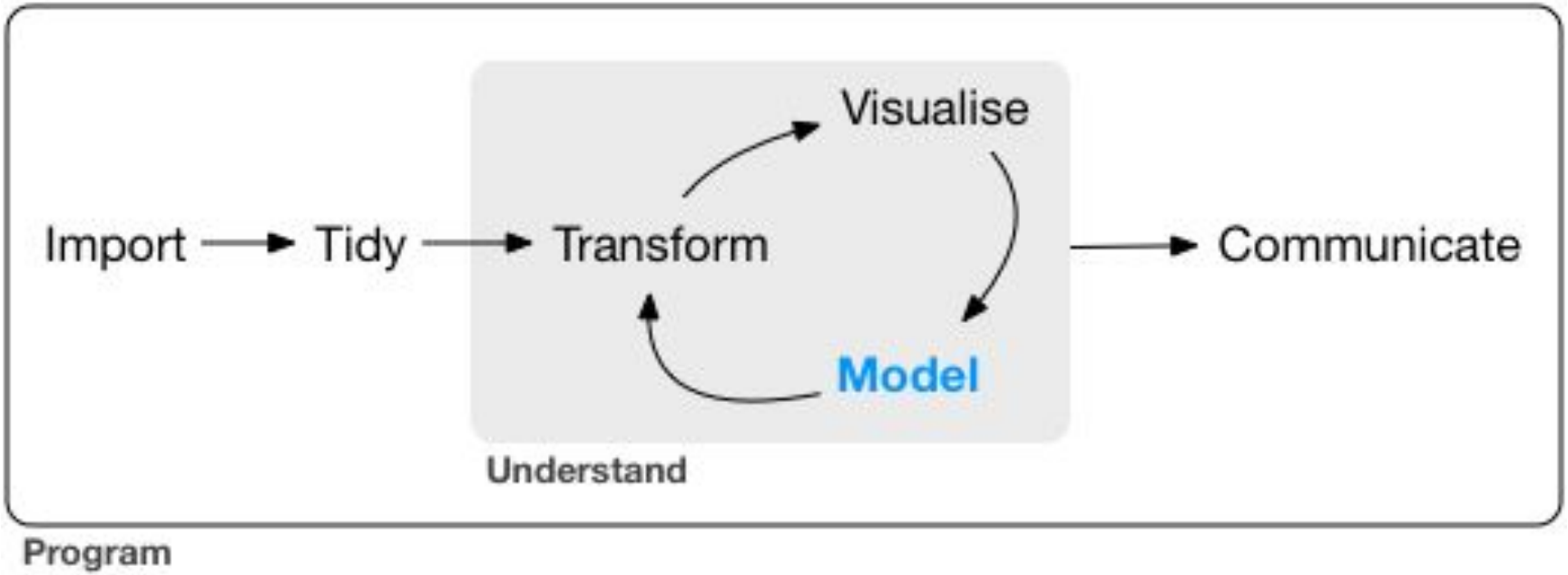
R for Data Science

Chapters 22 - 25

IV MODEL

Chapter 22 Introduction


Import
Tidy
Transform
Visualise
Model
Communicate
Understand
Program

- 60% of your data goes into a **training** (or exploration) set.

- 20% goes into a **query** set.

- 20% is held back for a **test** set.

Chapter 23 Model basics

2 parts to model:

- Define a **family of models**
- Generate a **fitted model**

Root-mean-squared deviation

- Difference between actual and predicted
- Square the number
- Average those numbers
- Take the square root of the number

`optim()` - general-purpose optimization based on Nelder–Mead, quasi-Newton and conjugate-gradient algorithms

`lm()` - used to fit linear models

modelr

`data_grid()` - generate an evenly spaced grid of points from the data

`add_predictions()` - add predictions to a data frame

`add_residuals()` - add residuals to a data frame

modelr

```
grid <- sim1 %>%
  data_grid(x)

grid <- grid %>%
  add_predictions(sim1_mod)

sim1 <- sim1 %>%
  add_residuals(sim1_mod)
```

Other Tips

Wrap `+`, `*`, `^`, `-` in `I()` to avoid being misinterpreted

`poly()` - returns or evaluates orthogonal polynomials of degree 1 to degree over the specified set of points x

`splines::ns()` - generate the B-spline basis matrix for a natural cubic spline

Other Models

`stats::glm()` - generalized linear models

`mgcv::gam()` - generalized additive models

`glmnet::glmnet()` - penalized linear models

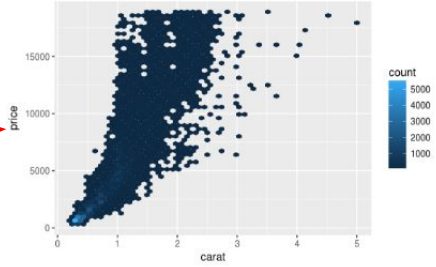
`MASS::rlm()` - robust linear models

`rpart::rpart()` - trees

Chapter 24 Model building

(1) Plot variables of interest

(2) Plot confounding variable



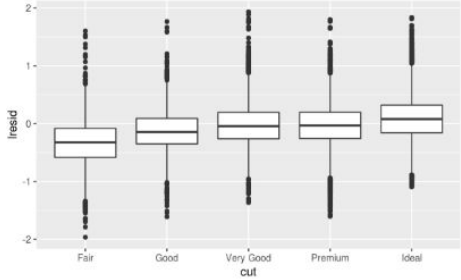
(3) Model confounding variable

```
mod_diamond <-
    lm(lprice ~
       lcarat,
       data = diamonds2)
```

(4) Pull out model residuals

```
diamonds2 <-diamonds2 %>%
   add_residuals(mod_diamond,
                 "lresid")
```

(5) Plot residuals for original variables of interest



Chapter 25 Many models

`tidyr::nest()` - creates a list of data frames containing all the nested variables

`tidyr::unnest()` - if you have a list-column, this makes each element of the list its own row

`purrr::map()` - transform their input by applying a function to each element and returning a vector the same length as the input

`tibble::enframe()` - converts named atomic vectors or lists to two-column data frames

```
by_country %>%
  mutate(glance = map(model,
                      broom::glance)) %>%
  unnest(glance)
```

new column to create

list column to use

function to apply to column

expand new column

broom

`glance()` - a row for each model

`tidy()` - a row for each coefficient in the model

`augment()` - a row for each row in the dataframe