

# Principles for modelling packages

*TBD*

*2018-07-06*



# Contents

|           |  |           |
|-----------|--|-----------|
| <b>1</b>  | <b>Intro</b>                             | <b>5</b>  |
| <b>2</b>  | <b>Conceptual overview of modelling</b>  | <b>7</b>  |
| <b>3</b>  | <b>Danger Zone</b>                       | <b>9</b>  |
| <b>4</b>  | <b>Data Specification</b>                | <b>11</b> |
| <b>5</b>  | <b>Documentation</b>                     | <b>13</b> |
| <b>6</b>  | <b>Functional programming principles</b> | <b>15</b> |
| <b>7</b>  | <b>Implementation</b>                    | <b>17</b> |
| <b>8</b>  | <b>Interactive modelling</b>             | <b>19</b> |
| <b>9</b>  | <b>Interface</b>                         | <b>21</b> |
| <b>10</b> | <b>Low and high level interfaces</b>     | <b>23</b> |
| <b>11</b> | <b>Model objects</b>                     | <b>25</b> |
| <b>12</b> | <b>Programmatic modelling</b>            | <b>27</b> |
| <b>13</b> | <b>References</b>                        | <b>29</b> |
| <b>14</b> | <b>Testing</b>                           | <b>31</b> |
| <b>15</b> | <b>Workflow</b>                          | <b>33</b> |
| 15.1      | Prediction . . . . .                     | 33        |
| 15.2      | Inference . . . . .                      | 33        |



# Chapter 1

## Intro

**Rule 1:** Always spell it *modelling*, never *modeling*.



## Chapter 2

# Conceptual overview of modelling

- what is a model: models, estimands, estimators and model specifications
- what do we do with models
- how do fit models
- once we have a fit model, how do we predict or do inference
- the difference between working with a single fit vs a set of fits. LASSO example: wanting to use the coefficients for prediction vs wanting to see the order in which features enter the model





## Chapter 3

# Danger Zone

little things to include somewhere: - the danger of misspecified arguments disappearing into . . .



## Chapter 4

# Data Specification

- formulas, model.frame, term objects, etc
- data / design matrix specification - **recipes**

habit: get the df right, then  $y \sim .$  in the formula. would be nice to still see the features in the call?

- ask users to use data.frames and tibbles, not matrices.



## Chapter 5

# Documentation

- vignette should include not only the coefficients as output in an example, but also those coefficients written up as a general latex model and as a latex model with those specific coefficients substituted in
- **show** your example data in the README so users immediately see the structure

function to write out model form and fitted model in latex for sanity checking: some sort of `model_report` / `model_form` generic

it's a bad idea to expect users to learn the *math* for your model from function level documentation, or math presented in ascii or unicode or poorly rendered latex.

show write out the math in a nicely formatted vignette, and then clearly describe the connection between code objects and math objects there as well



## Chapter 6

# Functional programming principles

calls to fit should be pure: i.e. no side effects like plotting, and especially no plotting with invisible object  
return - side effects: useful in interactive mode, irritating in programmatic mode

- type safety, particularly of returned objects
- type safety with respect to single fits vs sets of fits





## Chapter 7

# Implementation

- argument matching for categorical choices



## Chapter 8

# Interactive modelling

what most people do different because there's a person looking at stuff as opposed to programmatic model when it's just code interacting with the model with no human involved

this is a chapter mostly to remind us to think of differences between the two and how they might be important in terms of interface



## Chapter 9

# Interface

- user friendly interfaces

good and bad existing idioms

- methods to implement
- examples of tried and true workflows

methods to implement - note on plotting: Should be easy to get the values plotted so others can make their own plots

TWO DISTINCT ISSUES THAT GET RESOLVED IN FORMULAE:

design matrix specification

model specification. (a la `fGarch::garchFit(~arma(1, 1) + garch(1, 0))`)



## Chapter 10

# Low and high level interfaces

- high level versus low level interface
- programmatic versus interactive use

when you should use which

examples: - high level: keras, brms - low level: tensorflow, stan





## Chapter 11

# Model objects

- some explanation of why and how to save the function call
- generally what kinds of things should go into a model object, giving model objects a class so other people can extend them
- S3 object creation and validation for model building a la Advanced R
- Model classes beyond lists. when is S4 worth it? when is R6?
- Every modeling function should include its package version in its data object I will now save my models as a list of three objects: model, data, and `sessioninfo::session_info()`



## Chapter 12

# Programmatic modelling

i.e. interacting with models programmatically

examples: - packages that export a model from someone to use a la `botornot` - models sitting behind a Plumber API - etc



## Chapter 13

# References

- bdr's model fitting functions in r



## Chapter 14

# Testing

- testing against existing software - say a Matlab implementation
- saving long running models in `R/sysdata.rda` with `usethis::use_data(model_obj, internal = TRUE)`





# Chapter 15

## Workflow

### 15.1 Prediction

1. feature engineering
2. ML wizardry
3. more feature engineering
4. ???
5. predictions

### 15.2 Inference

1. Clean data
2. Specify model
3. Fit model
4. Check that model fitting process converged / worked
5. Check statistical assumptions of model

KEY part that always gets left out: working with multiple modellings