Walkthrough of Predicting IgG Titers Using Least Squares Regression

Jason Hsiao

Introduction

This document is a step-by-step walkthrough that demonstrates how to build a prediction model using data from Computational Models of Immunity - Pertussis Boost (CMI-PB) database: https://www.cmi-pb.org/. For this walkthrough, we will use batch-corrected data from the CMI-PB predictional challenge database to predict IgG titers at day 14 post-vaccination. More information about the data can be found on the CMI-PB website.

We will be predicting IgG titers using a lasso regression model. As with any machine learning model, we will need to have a prediction dataset and a training dataset. The prediction dataset will contain the data we want to predict, and the training dataset will contain the data we will use to train the model.

We will start by pre-processing the data for modeling, learn how to specify and train the model, then perform a prediction of subject IgG titers at day 14 post-vaccination.

Loading Packages

```r
library(tidymodels)
library(readr)       # for importing data
library(ggplot2)     # for visualizing data
library(dplyr)       # for data manipulation
```

Load Data

Let's import our antibody titer (ab_titer) data from the CMI-PB database. You will have to download the raw datasets for from the CMI-PB website, under 'Data and Resources' tab. You will need the following files:

1) Prediction data: 2022BD_plasma_ab_titer.tsv, 2022BD_subject.tsv, 2022BD_specimen.tsv
2) Training data: 2021LD_plasma_ab_titer.tsv, 2021LD_subject.tsv, 2021LD_specimen.tsv

The subject and specimen tables contain metadata about the subjects and specimens, which will be helpful in our predictions. The plasma_ab_titer table contains the antibody titer data.

Note: Since CMI-PB releases new data every year, the file names and data used in this walkthrough may not be the same as the data you have downloaded. If you have questions about this, there is a helpful 'Solutions Center' where you may post questions and get help from the CMI-PB community.

```r
# Import the prediction datasets
prediction_ab <-
  read_tsv("2022BD_plasma_ab_titer.tsv")

prediction_subject <-
  read_tsv("2022BD_subject.tsv")

prediction_specimen <-
  read_tsv("2022BD_specimen.tsv")

# Import the training datasets
training_ab <-
  read_tsv("2021LD_plasma_ab_titer.tsv")

training_subject <-
  read_tsv("2021LD_subject.tsv")

training_specimen <-
  read_tsv("2021LD_specimen.tsv")
```

Data Pre-Processing

Join tables

We want to have a master table for each of the training and prediction datasets such that each one contains antibody titer and associated metadata.

Firstly, `subject_id` corresponds to the unique identifier for each volunteer, from which specimens (samples) are collected at different time points, designated by `specimen_id`. To obtain a master metadata table for each of the training and prediction datasets (`subject_specimen`), we will join the subject and specimen tables by `subject_id`.

Then, to attach metadata to the training and prediction data, we will join the `subject_specimen` and abtiter tables by `specimen_id`.

```
# Join the subject and specimen tables by subject_id (common denominator for both tables)
training_meta <- inner_join(training_subject, training_specimen, by = "subject_id")

# Do the same for prediction data
prediction_meta <- inner_join(prediction_subject, prediction_specimen, by = "subject_id")


# Join the training antibody titer table to its metadata by specimen_id (common denominato
training <- inner_join(training_ab, training_meta, by = "specimen_id")

# Do the same for prediction
prediction <- inner_join(prediction_ab, prediction_meta, by = "specimen_id")
```

Inspect the Data

Now that we have our master tables for the training and prediction datasets, we can inspect the data to see what we are working with.

```
# Inspect each dataset
View(training)
View(prediction)
```

Woah! That's a lot of data... Since we are only looking at IgG PT, we can filter the data to only include IgG PT.

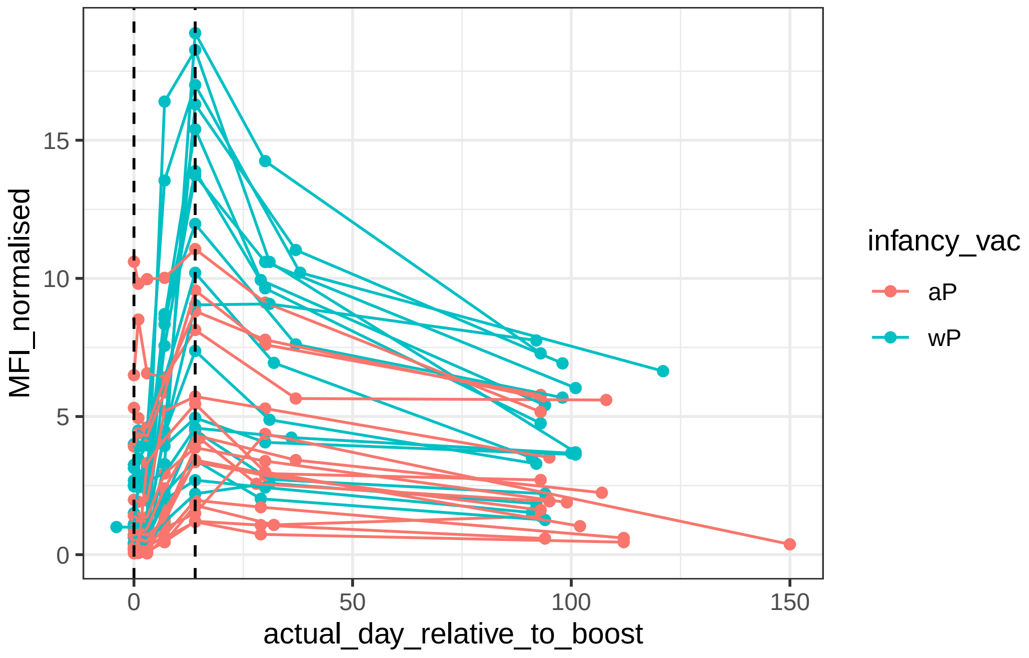
Filtering for IgG PT Data

We want only IgG PT, so we will reassign the `training` and `prediction` objects to only include specimens that are IgG PT.

```
# Filter the data to only include IgG PT
training <- training %>% filter(antigen == "PT", isotype == "IgG")
prediction <- prediction %>% filter(antigen == "PT", isotype == "IgG")
```

Now, let's plot the data and see what we have:

```r
ggplot(training,
       aes(actual_day_relative_to_boost, MFI_normalised,
           col = infancy_vac,
           group = subject_id)) +
  geom_point() +
  geom_line() +
  geom_vline(xintercept=0, linetype="dashed") +
  geom_vline(xintercept=14, linetype="dashed") +
  theme_bw()
```



While the overall trend is more like a curve, we can see that from day 0 to day 14, the data is more or less linear. Given that we are predicting the antibody titers at day 14, a linear model is a good starting point.

We can go one-step further and make our model more complex. If we look at the plot above, we can see that `infancy_vac` status has an influence on baseline IgG_PT values (y-intercept), and also the slope from day 0 to day 14 for each subject (each line). We can account for this later when we specify our model by adding `infancy_vac` as a predictor in the model.

Firstly, let's split the training data into a training set and a validation set.

Split the Training Data

In machine learning, it is important to split the training data into a training set and a validation set. The training set will be used to train the model, and the validation set will be used to evaluate the model's performance. Once we have the model, we can use it to predict the antibody titers in the prediction dataset by applying the model to the prediction dataset.

You are welcome to look for other resources to help you understand this concept, since it is a fundamental concept in machine learning. Here is a helpful [article] (https://medium.com/@nahmed3536/the-motivation-for-train-test-split-2b1837f596c3#:~:text=In%20Machine
to get you started.

```
# First, we want to set the seed. The seed is a number that is used to initialize the rand
set.seed(123)

# Here, we are using 80% of the data for training and 20% for testing. We use a special fu
training_split <- group_initial_split(training, prop = 0.8, group = "subject_id")

# Now we can extract the training and testing sets from the split
training_train <- training_split %>% training()
training_validation <- training_split %>% testing()

# We can inspect the training and testing sets to see what they look like
training_train
```

```
# A tibble: 182 x 20
   specimen_id isotype is_antigen_specific antigen   MFI MFI_normalised unit
         <dbl> <chr>   <lgl>               <chr>   <dbl>          <dbl> <chr>
 1         468 IgG     FALSE               PT       113.          1     MFI
 2         469 IgG     FALSE               PT       111.          0.987 MFI
 3         470 IgG     FALSE               PT       126.          1.11  MFI
 4         471 IgG     FALSE               PT       224.          1.99  MFI
 5         472 IgG     FALSE               PT       304           2.70  MFI
 6         473 IgG     FALSE               PT       274           2.43  MFI
 7         474 IgG     FALSE               PT       172.          1.52  MFI
 8         483 IgG     FALSE               PT       279.          2.47  MFI
 9         484 IgG     FALSE               PT       441.          3.92  MFI
10         485 IgG     FALSE               PT       498.          4.42  MFI
# i 172 more rows
# i 13 more variables: lower_limit_of_detection <dbl>, subject_id <dbl>,
#   infancy_vac <chr>, biological_sex <chr>, ethnicity <chr>, race <chr>,
#   year_of_birth <date>, date_of_boost <date>, dataset <chr>,
#   actual_day_relative_to_boost <dbl>, planned_day_relative_to_boost <dbl>,
#   specimen_type <chr>, visit <dbl>
```

```
training_validation
```

```
# A tibble: 49 x 20
   specimen_id isotype is_antigen_specific antigen   MFI MFI_normalised unit
         <dbl> <chr>   <lgl>               <chr>   <dbl>          <dbl> <chr>
 1         475 IgG     FALSE               PT       125.           1.11 MFI
 2         476 IgG     FALSE               PT       275.           2.44 MFI
 3         477 IgG     FALSE               PT       263.           2.33 MFI
 4         478 IgG     FALSE               PT       852.           7.56 MFI
 5         479 IgG     FALSE               PT      1548.          13.7  MFI
 6         480 IgG     FALSE               PT      1194.          10.6  MFI
 7         481 IgG     FALSE               PT       680.           6.03 MFI
 8         506 IgG     FALSE               PT       170.           1.51 MFI
 9         507 IgG     FALSE               PT       148.           1.31 MFI
10         508 IgG     FALSE               PT       150.           1.33 MFI
# i 39 more rows
# i 13 more variables: lower_limit_of_detection <dbl>, subject_id <dbl>,
#   infancy_vac <chr>, biological_sex <chr>, ethnicity <chr>, race <chr>,
#   year_of_birth <date>, date_of_boost <date>, dataset <chr>,
#   actual_day_relative_to_boost <dbl>, planned_day_relative_to_boost <dbl>,
#   specimen_type <chr>, visit <dbl>
```

Now that we have our training and validation sets, we can begin setting up the model. We will use a basic linear regression model (least squares regression) to predict the antibody titers in the prediction dataset.

The following utilizes the `tidymodels` framework to specify the model.

The tidymodels framework is a collection of packages for modeling and machine learning using tidyverse principles. More information on this framework can be found in the following link: https://www.tidymodels.org/start/models/.

Preparing Baseline Value Dataframes

Our ultimate goal is to have a model that predicts antibody titer as a function of `actual_day_relative_to_boost`, `infancy_vac`, and `MFI_normalised_baseline`. The reason why we want `MFI_normalised_baseline` is because we want to account for the baseline antibody titer values for each subject. In other words, any given `subject_id` will have a different baseline antibody titer value, and if we also know their `infancy_vac` status, we can use this information to predict their antibody titer at day 14.

`subject_id` is therefore not a predictor in the model, but rather a grouping variable.

To get MFI_normalised_baseline, we need to add a new column (variable) to our training and validation datasets that contains the baseline antibody titer value for each subject. We can then use this new column as a predictor in our model.

```r
# Create a dataframe by filtering the baseline MFI_normalised values for each subject
baseline_values <- training_train %>%
  filter(planned_day_relative_to_boost == 0)

# Create a dataframe that only has baseline values excluded for each subject
non_baseline_values <- training_train %>%
  filter(planned_day_relative_to_boost != 0)

# Append baseline_values to non_baseline_values as a separate column and call it MFI_norma
training_df <- non_baseline_values %>%
  left_join(baseline_values[,c('subject_id', 'MFI_normalised')], by = "subject_id", suffix


# Do the same for training_test
baseline_values <- training_validation %>%
  filter(planned_day_relative_to_boost == 0)

non_baseline_values <- training_validation %>%
  filter(planned_day_relative_to_boost != 0)

validation_df <- non_baseline_values %>%
  left_join(baseline_values[,c('subject_id', 'MFI_normalised')], by = "subject_id", suffix

# Let's also make infancy_vac column as factors, since we will be using them as factors in
training_validation$infancy_vac <- as.factor(training_validation$infancy_vac)

# Since our end goal is to predict day 14 antibody titers, let's also remove any data in o
training_df <- training_df %>% filter(actual_day_relative_to_boost <= 14)
validation_df <- validation_df %>% filter(actual_day_relative_to_boost <= 14)
```

Setting Up the Model

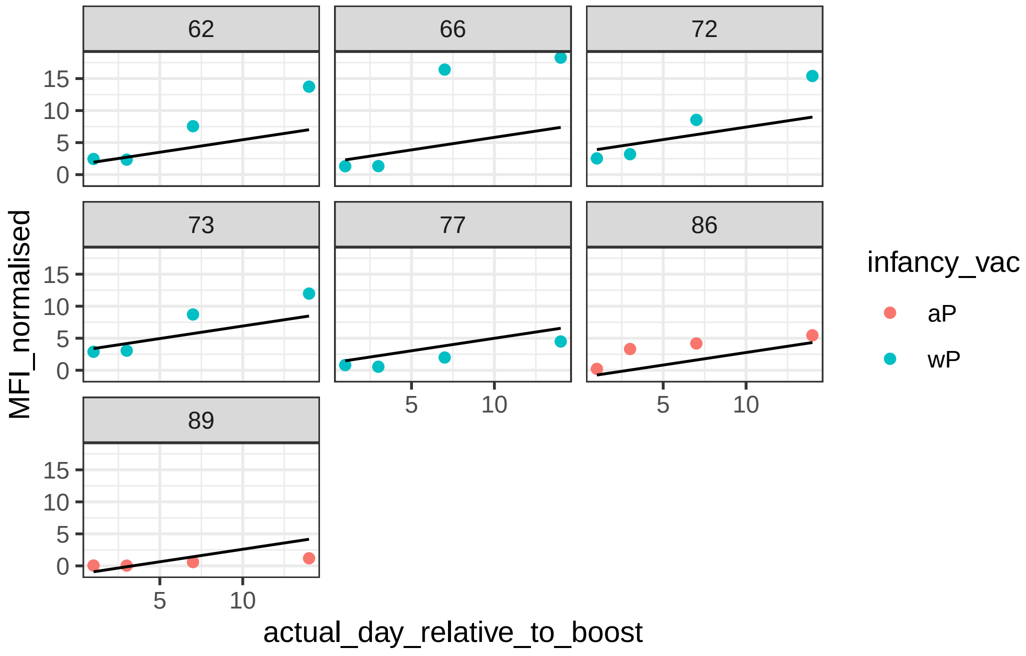
```r
# Specify the model
lm_train <- lm(MFI_normalised ~ actual_day_relative_to_boost + MFI_normalised_baseline + i

# Apply lm_train to the testing data and score for accuracy
ab_validation <- validation_df %>%
  mutate(predicted = predict(lm_train, newdata = validation_df))
```

Evaluating the Model

Let's plot out the data to see how well our model is doing. Each graph represents a different subject, and the black line represents the predicted antibody titers. The colored points represent the actual antibody titers, and the color of the points represents whether the subject received the infancy vaccine or not.

```r
# Evaluate Model Using ggplot
ggplot(ab_validation,
  aes(x = actual_day_relative_to_boost, y = MFI_normalised,
      col = infancy_vac)) +
  geom_point() +
  geom_line(aes(y = predicted), color = "black") +
  facet_wrap(~subject_id) +
  theme_bw()
```



It appears our model is doing a decent job of predicting the antibody titers from d0 to d14. We can also evaluate the model more quantitatively using the Spearman correlation between predicted and actual antibody titers columns in our ab_validation dataframe.

```r
# Calculate the Spearman correlation
cor.test(ab_validation$MFI_normalised, ab_validation$predicted, method = "spearman")
```

```
	Spearman's rank correlation rho

data:  ab_validation$MFI_normalised and ab_validation$predicted
S = 584, p-value = 1.348e-06
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho 
0.8401752 
```

As given by the relatively high Spearman correlation, the model is doing a good job of predicting the antibody titers.

Now that we have evaluated that our model is doing a good job of predicting antibody titers at d14, let's now predict the antibody titers in the 2022 prediction dataset using the model we just fit to the training data.

Predicting Antibody Titers in the Validation Dataset

We need to first repeat the preprocessing steps we did for the training and validation datasets for the prediction dataset. This includes adding a new column for the baseline antibody titer values for each subject, and then using the model to predict the antibody titers at d14.

```r
# Create a dataframe that only has the baseline values for each subject
baseline_values_predict <- prediction %>%
  filter(planned_day_relative_to_boost == 0)

# Rename baseline values to 'MFI_normalised_baseline'
baseline_values_predict <- baseline_values_predict %>%
  rename(MFI_normalised_baseline = MFI_normalised)
```

Now, we will use a function called `expand.grid()` to create a dataframe that specifies the d14 antibody titer predictions for each subject. We will then apply the model to this expanded dataframe to predict the antibody titers at d14 for each subject in the prediction dataset. Remember, this new dataframe must have all the predictor variables that the model was trained on!

```r
# Expand grid to create d14 output for each subject
new_points <- expand.grid(actual_day_relative_to_boost = 14,
                          subject_id = unique(baseline_values_predict$subject_id))

# Append MFI_normalised_baseline and infancy_vac values from baseline_values_predict as se
new_points <- new_points %>%
  left_join(baseline_values_predict[,c('subject_id', 'MFI_normalised_baseline', 'infancy_v

# We now have a dataframe that contains all the information needed for the model to predic
# Apply the model to the prediction dataset
ab_prediction <- new_points %>%
  mutate(predicted = predict(lm_train, newdata = new_points))

# Let's inspect ab_prediction to see the predicted antibody titers at d14 for each subject
ab_prediction
```

```
  actual_day_relative_to_boost subject_id MFI_normalised_baseline infancy_vac
1                           14        114               0.3576442          wP
2                           14        103               0.7043292          wP
3                           14        117               1.0390496          aP
4                           14         98               1.3099382          wP
5                           14        116               1.6782432          aP
```

```
6                          14        105              0.9049383        wP
7                          14         97              1.0606177        wP
8                          14        110              0.4578141        aP
9                          14        100              0.9677517        aP
10                         14        102              1.7359256        aP
11                         14        118              0.9614045        aP
12                         14        106              0.5452703        aP
13                         14        104              0.5212859        wP
14                         14        107              0.1188872        aP
15                         14        108              1.0429091        wP
16                         14        112              1.6592697        aP
17                         14         99              1.1962274        aP
18                         14        109              0.6713381        wP
19                         14        111              1.2502870        wP
20                         14        115              1.7355202        aP
21                         14        101              1.6515831        aP
   predicted
1   6.319894
2   6.638975
3   5.071408
4   7.196364
5   5.659707
6   6.823611
7   6.966895
8   4.536453
9   5.005787
10  5.712797
11  4.999945
12  4.616945
13  6.470506
14  4.224512
15  6.950596
16  5.642245
17  5.216071
18  6.608611
19  7.141462
20  5.712424
21  5.635170
```

Importing Actual d14 Antibody Titers to Compare

While we have been working with a 2022 dataset (prediction dataset) that only has d0 baseline values, we have access to the actual d14 antibody titers from the 2022 dataset. We can use these to compare our predicted values to the actual values. Let's get this from the CMI-PB website (2nd challenge results).