Comments on the prep_data Function

0.1 Introduction

0.1.1 The overall code

This is the overall code for the 'prep_data' function.

```
prep_data <- function(dataframe) {
  # Ensure the data has the required columns
  if (!("antigen_iso" %in% names(dataframe)) || !("visit_num" %in%
                                                    names(dataframe))) {
    stop("Dataframe must contain 'antigen_iso' and 'visit_num' columns")
  }



  # Extract unique visits and antigens
  visits <- unique(dataframe$visit_num)
  antigens <- unique(dataframe$antigen_iso)
  subjects <- unique(dataframe$index_id)

  # Initialize arrays to store the formatted data
  max_visits <- length(visits)
  max_antigens <- length(antigens)
  num_subjects <- length(subjects)

  # Define arrays with dimensions to accommodate extra dummy subject

  dimnames1 = list(
    subjects = c(subjects, "newperson"),
    visit_number = paste0("V", visits)
  )
```

```r
dims1 = sapply(F = length, X = dimnames1)

visit_times <- array(
  NA,
  dim = dims1,
  dimnames = dimnames1)

dimnames2 = list(
  subjects = c(subjects, "newperson"),
  visit_number = paste0("V", visits),
  antigens = antigens
)

antibody_levels <- array(
  NA,
  dim = c(num_subjects + 1, max_visits, max_antigens),
  dimnames = dimnames2)

nsmpl <- integer(num_subjects + 1)  # Array to store the maximum number
#of samples per participant

for (i in seq_len(num_subjects)) {
  subject_data <- dataframe[dataframe$index_id == subjects[i], ]
  subject_visits <- unique(subject_data$visit_num)
  nsmpl[i] <- length(subject_visits)  # Number of non-missing visits for this
  # participant

  for (j in seq_along(subject_visits)) {
    for (k in seq_len(max_antigens)) {
      subset <- subject_data[subject_data$visit_num == subject_visits[j] &
                               subject_data$antigen_iso == antigens[k], ]
      if (nrow(subset) > 0) {
        visit_times[i, j] <- subset$timeindays
        antibody_levels[i, j, k] <- log(max(0.01, subset$result))
        # Log-transform and handle zeroes
      }
    }
  }
}

# Add missing observation for Bayesian inference
visit_times[num_subjects + 1, 1:3] <- c(5, 30, 90)
```

```
  # Ensure corresponding antibody levels are set to NA (explicitly missing)
  antibody_levels[num_subjects + 1, 1:3, ] <- NA
  nsmpl[num_subjects + 1] <- 3  # Since we manually add three timepoints
  # for the dummy subject




  # Return results as a list
  to_return =
    list(
      "smpl.t" = visit_times,
      "logy" = antibody_levels,
      "n_antigen_isos" = max_antigens,
      "nsmpl" = nsmpl ,
      "nsubj" = num_subjects + 1


      # "index_id_names" = subjects,
      # "antigen_names" = antigens
    ) |>
    structure(
      antigens = antigens,
      n_antigens = max_antigens,
      ids = c(subjects, "newperson")
    )

  # Return results as a list
  return(to_return)
}
```

0.2 Body

0.2.1 Comments on each part

0.2.1.1 Function Definition and Column Check:

```
prep_data <- function(dataframe) {
  if (!("antigen_iso" %in% names(dataframe)) || !("visit_num" %in%
                                                  names(dataframe))) {
```

```
    stop("Dataframe must contain 'antigen_iso' and 'visit_num' columns")
  }}
```

- The function prep_data takes a dataframe as an input.
- It checks if the dataframe contains the required columns antigen_iso and visit_num.

0.2.1.2 Extract Unique Values:

```
visits <- unique(dataframe$visit_num)
antigens <- unique(dataframe$antigen_iso)
subjects <- unique(dataframe$index_id)
```

- Extracts unique values of visit_num, antigen_iso, and index_id.

0.2.1.3 Initialize Arrays:

```
max_visits <- length(visits)
max_antigens <- length(antigens)
num_subjects <- length(subjects)
```

- Initializes variables to store the number of unique visits, antigens, and subjects.

0.2.1.4 Define Array Dimensions:

```
dimnames1 = list(
    subjects = c(subjects, "newperson"),
    visit_number = paste0("V", visits)
  )

  dims1 = sapply(F = length, X = dimnames1)

  visit_times <- array(
    NA,
    dim = dims1,
    dimnames = dimnames1)

  dimnames2 = list(
    subjects = c(subjects, "newperson"),
    visit_number = paste0("V", visits),
    antigens = antigens
```

```
)

antibody_levels <- array(
  NA,
  dim = c(num_subjects + 1, max_visits, max_antigens),
  dimnames = dimnames2)

nsmpl <- integer(num_subjects + 1)  # Array to store the maximum number of
# samples per participant
```

- Creates arrays to store visit times and antibody levels, initializing with NA values.
- The arrays have dimensions based on the number of subjects, visits, and antigens, with an additional "newperson" for Bayesian inference.

0.2.1.5 Populate Arrays:

```
for (i in seq_len(num_subjects)) {
  subject_data <- dataframe[dataframe$index_id == subjects[i], ]
  subject_visits <- unique(subject_data$visit_num)
  nsmpl[i] <- length(subject_visits)  # Number of non-missing visits for this
  # participant

  for (j in seq_along(subject_visits)) {
    for (k in seq_len(max_antigens)) {
      subset <- subject_data[subject_data$visit_num == subject_visits[j] &
                               subject_data$antigen_iso == antigens[k], ]
      if (nrow(subset) > 0) {
        visit_times[i, j] <- subset$timeindays
        antibody_levels[i, j, k] <- log(max(0.01, subset$result))
        # Log-transform and handle zeroes
      }
    }
  }
}
```

- This loop fills the arrays with actual data from the dataframe:
  - For each subject, it gets their data and unique visits.
  - Fills visit_times with timeindays and antibody_levels with log-transformed result.

0.2.1.6 Add Dummy Subject for Bayesian Inference:

```
visit_times[num_subjects + 1, 1:3] <- c(5, 30, 90)
antibody_levels[num_subjects + 1, 1:3, ] <- NA
nsmpl[num_subjects + 1] <- 3  # Since we manually add three timepoints for the
# dummy subject
```

- Adds a dummy subject with three time points (5, 30, 90 days) and corresponding NA antibody levels.

0.2.1.7 Return Results:

```
  to_return = list(
    "smpl.t" = visit_times,
    "logy" = antibody_levels,
    "n_antigen_isos" = max_antigens,
    "nsmpl" = nsmpl,
    "nsubj" = num_subjects + 1
  ) |> structure(
    antigens = antigens,
    n_antigens = max_antigens,
    ids = c(subjects, "newperson")
  )

  return(to_return)
```

- Returns a list containing the visit times, antibody levels, number of antigens, sample counts, and number of subjects.

0.3 Conclusion

The 'timeindays' variable may used to fill the visit_times array with the actual days when visits occurred. This might be crucial for accurate longitudinal analysis as it provides the exact time points for each measurement.