EHR Vignette for *Extract-Med* and *Pro-Med-NLP*

Introduction

The `EHR` package provides several modules to perform diverse medication-related studies using data from electronic health record (EHR) databases. Especially, the package includes modules to perform pharmacokinetic/pharmacodynamic (PK/PD) analyses using EHRs, as outlined in Choi, et al.[1], and additional modules will be added in future. This vignette describes two modules in the system, when drug dosing information should be obtained from unstructured clinical notes. The process starts with data extraction (*Extract-Med*), then moves on to data processing (*Pro-Med-NLP*), from which a final data set can be built.

```
library(EHR)
```

Extract-Med

The *Extract-Med* module uses a natural language processing (NLP) system called *medExtractR*.[2] The medExtractR system is a medication extraction system that uses regular expressions and rule-based approaches to identify key dosing information including drug name, strength, dose amount, frequency or intake time, dose change, and last dose time. Function arguments can be specified to allow the user to tailor the medExtractR system to the particular drug or dataset of interest, improving the quality of extracted information.

Setup of `extractMed`

The function `extractMed` will run the Extract-Med module. In order to run the Extract-Med module with medExtractR, certain function arguments must be specified, including:

- `note_fn`: The file name of the note on which to run the system. This can be either a single file name (e.g., `"clinical_note01.txt"`) or a vector or list of file names (e.g., `c("clinical_note01.txt", "clinical_note02.txt")` or `list("clinical_note01.txt", "clinical_note02.txt")`).

- `drugnames`: Names of the drugs for which we want to extract medication dosing information. This can include any way in which the drug name might be represented in the clinical note, such as generic name (e.g., `"lamotrigine"`), brand name (e.g., `"Lamictal"`), or an abbreviation (e.g., `"LTG"`).

- `drgunit`: The unit of the drug(s) listed in `drugnames`, for example `"mg"`.

- `windowlength`: Length of the search window around each found drug name in which to search for dosing information. There is no default for this argument, requiring the user to carefully consider its value through tuning (see tuning section below).

- `max_edit_dist`: The maximum edit distance allowed when identifying `drugnames`. Maximum edit distance determines the difference between two strings, and is defined as the number of insertions, deletions, or substitutions required to change one string into the other. This allows us to capture misspellings in the drug names we are searching for, and its value should be carefully considered through tuning (see tuning section below).

  - The default value is '0', or exact spelling matches to `drugnames`. A value of 0 is always used for drug names with less than 5 characters regardless of the value set by `max_edit_dist`.
  - A value of 1 would capture mistakes such as a single missing or extra letter, e.g., "tacrlimus" or "tacroolimus" instead of "tacrolimus"
  - A value of 2 would capture these mistakes or a single transposition, e.g. "tcarolimus" instead of "tacrolimus"
  - Higher values (3 or above) would capture increasingly more severe mistakes, though setting the value too high can cause similar words to be mistaken as the drug name.

Generally, the function call to `extractMed` is

```
extractMed(note_fn, drugnames, drgunit, windowlength, max_edit_dist, ...)
```

where `...` refers to additional arguments to *medExtractR* (see package documentation for details[3]). Examples of additional arguments include:

- `strength_sep`, where users can specify special characters to separate doses administered at different times of day. For example, consider the drug name *"lamotrigine"* and the phrase *"Patient is on lamotrigine 200-300"*, indicating that the patient takes 200 mg of the drug in the morning and 300 mg in the evening. Setting `strength_sep = c('-')` would allow `extractMed` to identify the expression *200-300* as "Dose" (i.e., dose given intake) since they are separated by the special character "-". The default value is `NULL`.
- `lastdose`, a logical input specifying whether or not the last dose time entity should be extracted. Default value is `FALSE`.

As mentioned above, some arguments to `extractMed` should be specified through a tuning process. In a later section, we briefly describe the process by which a user could tune the medExtractR system (and by extension the `extractMed` function) using a validated gold standard dataset.

Running `extractMed`

Below, we demonstrate how to execute the Extract-Med module using sample notes for two drugs: tacrolimus (simpler prescription patterns, used to prevent rejection after organ transplant) and lamotrigine (more complex prescription patterns, used to treat epilepsy). The arguments specified for each drug here were determined based on training sets of 60 notes for each drug.[2] Note that for tacrolimus we enter the file names as a list, and for lamotrigine we enter the file names as a character vector; either form of input is acceptable to the `extractMed` function and will produce the same output. We also specify `lastdose=TRUE` for tacrolimus to extract information about time of last dose, and `strength_sep="-"` for lamotrigine.

```
# tacrolimus note file names
tac_fn <- list(
  system.file("examples", "tacpid1_2008-06-26_note1_1.txt", package = "EHR"),
  system.file("examples", "tacpid1_2008-06-26_note2_1.txt", package = "EHR"),
  system.file("examples", "tacpid1_2008-12-16_note3_1.txt", package = "EHR")
)

# execute module
tac_mxr <- extractMed(tac_fn,
                      drugnames = c("tacrolimus", "prograf", "tac", "tacro", "fk", "fk506"),
                      drgunit = "mg",
                      windowlength = 60,
                      max_edit_dist = 2,
                      lastdose=TRUE)
```

```
## running notes 1-3 in batch 1 of 1 (100%)
```

```
## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'
```

```
# lamotrigine note file name
lam_fn <- c(
  system.file("examples", "lampid1_2016-02-05_note4_1.txt", package = "EHR"),
  system.file("examples", "lampid1_2016-02-05_note5_1.txt", package = "EHR"),
  system.file("examples", "lampid2_2008-07-20_note6_1.txt", package = "EHR"),
  system.file("examples", "lampid2_2012-04-15_note7_1.txt", package = "EHR")
)

# execute module
lam_mxr <- extractMed(lam_fn,
                      drugnames = c("lamotrigine", "lamotrigine XR",
                                    "lamictal", "lamictal XR",
                                    "LTG", "LTG XR"),
                      drgunit = "mg",
                      windowlength = 130,
                      max_edit_dist = 1,
                      strength_sep="-")
```

```
## running notes 1-4 in batch 1 of 1 (100%)
```

```
## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'
```

The format of output from the Extract-Med module is a `data.frame` with 4 columns:

- `filename`: The file name of the corresponding clinical note, to label results.
- `entity`: The label of the entity for the extracted expression.
- `expr`: Expression extracted from the clinical note.
- `pos`: Position of the extracted expression in the note, in the format `startPosition:stopPosition`

```
## tacrolimus medExtractR output:
```

```
##                         filename    entity        expr     pos
## 1 tacpid1_2008-06-26_note1_1.txt  DrugName     Prograf 930:937
## 2 tacpid1_2008-06-26_note1_1.txt     Route        Oral 938:942
```

```
## 3 tacpid1_2008-06-26_note1_1.txt  Strength        1 mg 951:955
## 4 tacpid1_2008-06-26_note1_1.txt   DoseAmt           3 956:957
## 5 tacpid1_2008-06-26_note1_1.txt     Route    by mouth 967:975
## 6 tacpid1_2008-06-26_note1_1.txt Frequency twice a day 976:987
```

```
## lamotrigine medExtractR output:
```

```
##                          filename       entity        expr     pos
## 1 lampid1_2016-02-05_note4_1.txt     DrugName    Lamictal 810:818
## 2 lampid1_2016-02-05_note4_1.txt DoseStrength      300 mg 819:825
## 3 lampid1_2016-02-05_note4_1.txt    Frequency         BID 826:829
## 4 lampid1_2016-02-05_note4_1.txt     DrugName Lamotrigine 847:858
## 5 lampid1_2016-02-05_note4_1.txt     Strength       200mg 859:864
## 6 lampid1_2016-02-05_note4_1.txt      DoseAmt         1.5 865:868
```

The lamotrigine results can be immediately put into the `parse` step of the Pro-Med-NLP module. For the tacrolimus output, we chose to also extract the last dose time entity by specifying `lastdose=TRUE`. The last dose time entity is extracted as raw character expressions from the clinical note, and must first be converted to a standardized datetime format, which is addressed in a later section (see "Handling `lastdose`" section). The output of `extractMed` must be saved as a CSV file, the filename of which will serve as the first input to the Pro-Med-NLP module.

```
# save as csv files
write.csv(tac_mxr, file='tac_mxr.csv', row.names=FALSE)
write.csv(lam_mxr, file='lam_mxr.csv', row.names=FALSE)
```

Tuning the medExtractR system

In a previous section, we mentioned that parameters within the Extract-Med module should be tuned in order to ensure higher quality of extracted drug information. This section describes how we go about tuning medExtractR, the NLP system underlying the Extract-Med module.

In order to tune medExtractR, we recommend selecting a small set of tuning notes, from which the parameter values can be selected. Below, we describe this process with a set of three notes (note that these notes were chosen for the purpose of demonstration, and we recommend using tuning sets of at least 10 notes).

Once a set of tuning notes has been curated, they must be manually annotated by reviewers to identify the information that should be extracted. This process produces a gold standard set of annotations, which identify the correct drug information of interest. This includes entities like the drug name, strength, and frequency. For example, in the phrase

Patient is taking **lamotrigine** *300 mg* in the morning and *200 mg* in the evening

bolded, italicized, and underlined phrases represent annotated drug names, dose (i.e., dose given intake), and intake times, respectively. These annotations are stored as a dataset.

First, we read in the annotation files for three example tuning notes, which can be generated using an annotation tool, such as the Brat Rapid Annotation Tool (BRAT) software.[4] By default, the output file from BRAT is tab delimited with 3 columns: an annotation identifier, a column with labelling information in the format "label startPosition stopPosition", and the annotation itself, as shown in the example below:

```
##   id           entity  annotation
## 1 T1   DrugName 19 30 lamotrigine
## 2 T2       Dose 31 37      300 mg
## 3 T3 IntakeTime 45 52     morning
## 4 T4       Dose 57 63      200 mg
## 5 T5 IntakeTime 71 78     evening
```

In order to compare with the medExtractR output, the format of the annotation dataset should be four columns with:

1. The file name of the corresponding clinical note
2. The entity label of the annotated expression
3. The annotated expression
4. The start and stop position of the annotated expression in the format "start:stop"

The exact formatting performed below is specific to the format of the annotation files, and may vary if an annotation software other than BRAT is used.

```
# Read in the annotations - might be specific to annotation method/software
ann_filenames <- list(system.file("mxr_tune", "tune_note1.ann", package = "EHR"),
                      system.file("mxr_tune", "tune_note2.ann", package = "EHR"),
                      system.file("mxr_tune", "tune_note3.ann", package = "EHR"))

tune_ann <- do.call(rbind, lapply(ann_filenames, function(fn){
  annotations <- read.delim(fn,
                            header = FALSE, sep = "\t", stringsAsFactors = FALSE,
                            col.names = c("id", "entity", "annotation"))

  # Label with file name
  annotations$filename <- sub(".ann", ".txt", sub(".+/", "", fn), fixed=TRUE)

  # Separate entity information into entity label and start:stop position
  # Format is "entity start stop"
  ent_info <- strsplit(as.character(annotations$entity), split="\\s")
  annotations$entity <- unlist(lapply(ent_info, '[[', 1))
  annotations$pos <- paste(lapply(ent_info, '[[', 2),
                           lapply(ent_info, '[[', 3), sep=":")

  annotations <- annotations[,c("filename", "entity", "annotation", "pos")]

  return(annotations)
}))

head(tune_ann)
```

```
##         filename    entity  annotation       pos
## 1 tune_note1.txt  DrugName     Prograf 1219:1226
## 2 tune_note1.txt  Strength        1 mg 1227:1231
## 3 tune_note1.txt   DoseAmt           3 1236:1237
## 4 tune_note1.txt     Route    by mouth 1247:1255
## 5 tune_note1.txt Frequency twice a day 1256:1267
## 6 tune_note1.txt  DrugName     porgraf 3873:3880
```

To select appropriate tuning parameters, we identify a range of possible values for each of the `windowlength` and `max_edit_dist` parameters. Here, we allow `windowlength` to vary from 30 to 120 characters in increments of 30, and `max_edit_dist` to take a value of 0, 1, or 2. We then obtain the `extractMed` results for each combination. We set `progress=FALSE` to avoid printing the progress message for each combination of arguments.

```r
wind_len <- seq(30, 120, 30)
max_edit <- seq(0, 2, 1)
tune_pick <- expand.grid("window_length" = wind_len,
                         "max_edit_distance" = max_edit)

# Run the Extract-Med module on the tuning notes
note_filenames <- list(system.file("mxr_tune", "tune_note1.txt", package = "EHR"),
                       system.file("mxr_tune", "tune_note2.txt", package = "EHR"),
                       system.file("mxr_tune", "tune_note3.txt", package = "EHR"))

# List to store output for each parameter combination
mxr_tune <- vector(mode="list", length=nrow(tune_pick))

for(i in 1:nrow(tune_pick)){
  mxr_tune[[i]] <- extractMed(note_filenames,
               drugnames = c("tacrolimus", "prograf", "tac", "tacro", "fk", "fk506"),
               drgunit = "mg",
               windowlength = tune_pick$window_length[i],
               max_edit_dist = tune_pick$max_edit_distance[i],
               progress = FALSE)
}
```

```
## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'
```

```
## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'
```

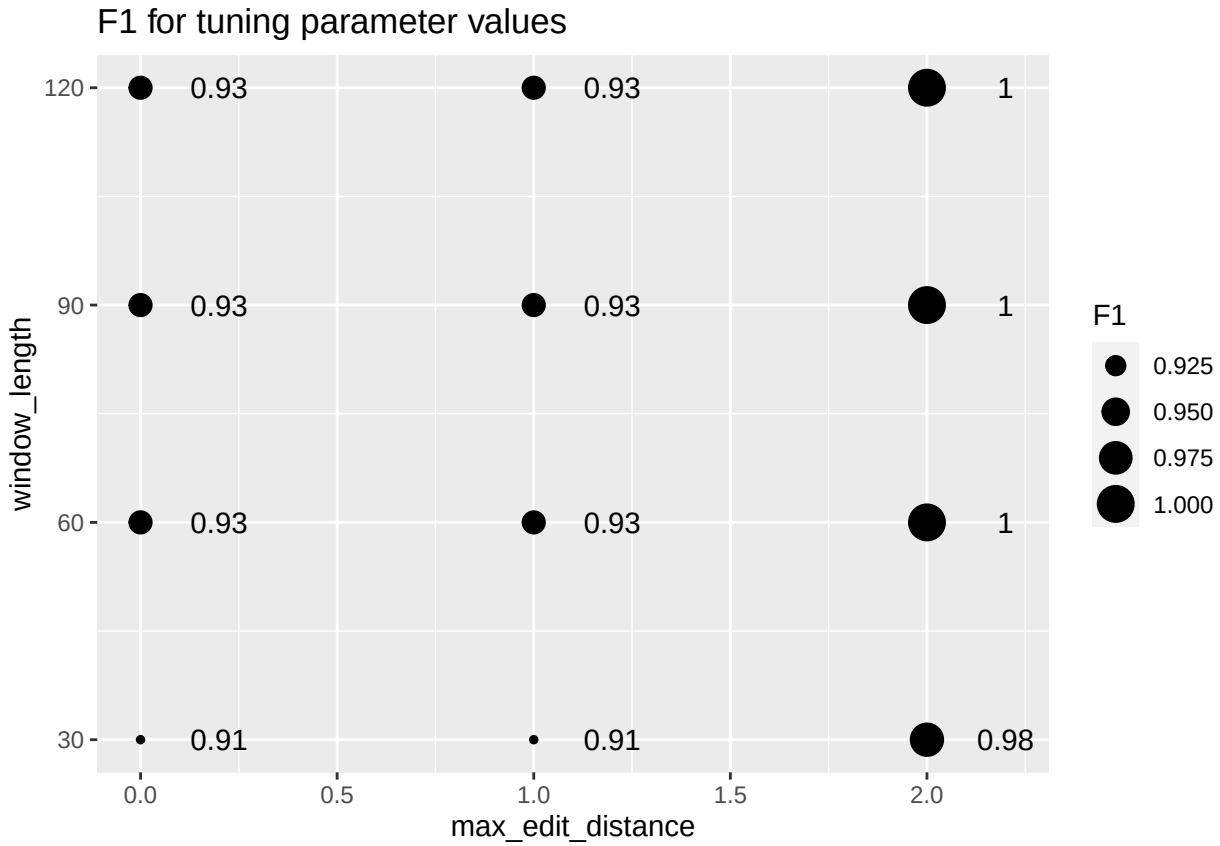
```
## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'

## Warning in length(dosechange_dict) == 1L && dosechange_dict[1] == "default":
## 'length(x) = 24 > 1' in coercion to 'logical(1)'
```

Finally, we determine which parameter combination yielded the highest performance, quantified by some metric. For our purpose, we used the F1-measure (F1), the harmonic mean of precision $\left(\frac{\text{true positives}}{\text{true positives + false positives}}\right)$ and recall $\left(\frac{\text{true positives}}{\text{true positives + false negatives}}\right)$. Tuning parameters were selected based on which combination maximized F1 performance within the tuning set. The code below determines true positives as well as false positives and negatives, used to compute precision, recall, and F1.

```
# Functions to compute true positive, false positive, and false negatives
# number of true positives - how many annotations were correctly identified by extractMed
Tpos <- function(df){
  sum(df$annotation == df$expr, na.rm=TRUE)
}
# number of false positive (identified by extractMed but not annotated)
Fpos <- function(df){
  sum(is.na(df$annotation))
}
# number of false negatives (annotated but not identified by extractMed)
Fneg <- function(df){
  # keep only rows with annotation
  df_ann <- subset(df, !is.na(annotation))
  sum(is.na(df$expr))
}

prf <- function(df){
  tp <- Tpos(df)
  fp <- Fpos(df)
  fn <- Fneg(df)

  precision <- tp/(tp + fp)
  recall <- tp/(tp + fn)
  f1 <- (2*precision*recall)/(precision + recall)

  return(f1)
```

```
}
```

```
tune_pick$F1 <- sapply(mxr_tune, function(x){
  compare <- merge(x, tune_ann,
                   by = c("filename", "entity", "pos"), all = TRUE)
  prf(compare)
})


ggplot(tune_pick) + geom_point(aes(max_edit_distance, window_length, size = F1)) +
  scale_y_continuous(breaks=seq(30,120,30)) +
  annotate("text", x = tune_pick$max_edit_distance+.2, y = tune_pick$window_length,
           label = round(tune_pick$F1, 2)) +
  ggtitle("F1 for tuning parameter values")
```



The plot shows that the highest F1 achieved was 1, and occurred for three different combinations of parameter values: a maximum edit distance of 2 and a window length of 60, 90, or 120 characters. The relatively small number of unique F1 values is likely the result of only using 3 tuning notes. In this case, we would typically err on the side of allowing a larger search window and decide to use a maximum edit distance of 2 and a window length of 120 characters. In a real-world tuning scenario and with a larger tuning set, we would also want to test longer window lengths since the best case scenario occurred at the longest window length we used.

Pro-Med-NLP

The *Pro-Med-NLP* module uses a dose data building algorithm to generate longitudinal medication dose data from the raw output of an NLP system.[5] The algorithm is divided into two parts. Part I parses the raw output and pairs entities together, and Part II calculates dose intake and daily dose and removes redundant information.

Part I

Two main functions are used in this part of the algorithm, a parse function (`parseMedExtractR`, `parseMedXN`, `parseCLAMP`, or `parseMedEx`) and `buildDose`.

Parse functions (`parseMedExtractR`, `parseMedXN`, `parseCLAMP`, `parseMedEx`) Parse functions are available for the medExtractR, MedXN, CLAMP, and MedEx systems (`parseMedExtractR`, `parseMedXN`, `parseCLAMP`, `parseMedEx`). If the user has output from another NLP system, the user can write code to standardize the output before calling the `buildDose` function.

The parse functions require the following argument:

- `filename`: file name for a file containing the raw output from the NLP system

The following are the function calls for the parse functions.

```
parseMedExtractR(filename)
parseMedXN(filename, begText = "^[ID0-9]+_[0-9-]+_[Note0-9]")
parseCLAMP(filename)
parseMedEx(filename)
```

The `parseMedXN` function also has an argument specifying a regular expression pattern which indicates the start of each row (i.e., drug mention) in the raw MedXN output. In the example above, the `begText` argument identifies how the file names are structured for a set of clinical notes in a Vanderbilt University dataset (e.g., "ID111_2000-01-01_Note001" where ID111 is the patient ID, 2000-01-01 is the date of the note, and Note001 is the note ID). Thus, each row beginning with a file name in this format indicates a new drug mention.

The parse functions output a standardized form of the data that includes a row for each drug mention and columns for all entities anchored to that drug mention.

Running `parseMedExtractR` Below we demonstrate how to run `parseMedExtractR` using the example medExtractR output from above. The file names here correspond to the files generated from the final output of the Extract-Med module.

```
tac_mxr_fn <- system.file("examples", "tac_mxr.csv", package = "EHR")
lam_mxr_fn <- system.file("examples", "lam_mxr.csv", package = "EHR")
lam_mxn_fn <- system.file("examples", "lam_medxn.csv", package = "EHR")
tac_mxr_parsed <- parseMedExtractR(tac_mxr_fn)
lam_mxr_parsed <- parseMedExtractR(lam_mxr_fn)
```

Running `parseMedXN` Below is an example using the `parseMedXN` function.

```
## MedXN output:

## ID1_2012-11-22_Note1.txt|lamotrigine::255::266|28439::196502|100 mg::278::284|
##      2::288::289`1.5::310::313|tabs::290::294`tabs::314::318`tabs::350::354||
##      morning::298::305`evening::322::329|2 weeks::334::341
## ID1_2012-11-22_Note1.txt|Vimpat::1086::1092|809979|200mg::1093::1098||tab::1099::1102|
```

```
##      mouth::1106::1111|twice daily::1112::1123
## ID1_2012-11-22_Note1.txt|lamotrigine::1172::1183|28439|100 mg::1184::1190||
##      tablet::1191::1197
## ID1_2012-11-22_Note1.txt|Lamictal::1213::1221|196502||1.5::1223::1226|
##      tablets::1227::1234|mouth::1238::1243|twice a day::1244::1255
```

The output from all systems, once parsed, has the same structure as the example parsed MedXN output below.

```
lam_mxn_parsed <- parseMedXN(lam_mxn_fn, begText = "^[ID0-9]+_[0-9-]+_[Note0-9]")
```

```
##                    filename                drugname           strength
## 1 ID1_2012-11-22_Note1.txt   lamotrigine::255::266   100 mg::278::284
## 2 ID1_2012-11-22_Note1.txt      Vimpat::1086::1092  200mg::1093::1098
## 3 ID1_2012-11-22_Note1.txt lamotrigine::1172::1183 100 mg::1184::1190
## 4 ID1_2012-11-22_Note1.txt    Lamictal::1213::1221
##
##                              dose             route
##      1 2::288::289`1.5::310::313
##      2                            mouth::1106::1111
##      3                                         <NA>
##      4           1.5::1223::1226 mouth::1238::1243
##
##                                      freq          duration
##      1 morning::298::305`evening::322::329 2 weeks::334::341
##      2             twice daily::1112::1123              <NA>
##      3                                <NA>              <NA>
##      4             twice a day::1244::1255              <NA>
```

buildDose After the NLP output is parsed, the buildDose function is run to pair the parsed entities. The main buildDose function arguments are as follows:

- dat: standardized form of the data (i.e., output from one of the parse functions)
- dn: argument to specify drug names of interest (generally not used with medExtractR since medExtractR is a targeted system and only includes the drug of interest in the raw output)

The general function call is:

```
buildDose(dat, dn = NULL)
```

Other notable arguments that can be specified by buildDose include:

- preserve: columns which should be preserved in the final output, but not considered for pairing with other entities. The default is NULL, though for medExtractR the dose change entity is always preserved.
- dist_method: which distance method calculation to use in cost calculation for pairing entities. For more details see the supplementary material in McNeer, et al. (2020). The default from getOption('ehr.dist_method') is *minEntEnd*, or using the minimum distance between the end of one entity and beginning of the next).
- checkForRare: a logical argument, indicates whether or not the output should return rare occurrences of entities (default is FALSE)

The output of the buildDose function is a dataset with a column for each entity and a row for each pairing.

Running buildDose In our medExtractR example from above, the output of the buildDose function is the following:

```
(tac_part_i_out <- buildDose(tac_mxr_parsed))
```

```
##                         filename   drugname strength dose route        freq
## 1 tacpid1_2008-06-26_note1_1.txt    Prograf     1 mg    3    NA twice a day
## 2 tacpid1_2008-06-26_note1_1.txt    prograf     <NA> <NA>    NA         bid
## 3 tacpid1_2008-06-26_note2_1.txt    Prograf     1 mg    3    NA twice a day
## 4 tacpid1_2008-12-16_note3_1.txt Tacrolimus     <NA> <NA>    NA        <NA>
## 5 tacpid1_2008-12-16_note3_1.txt    Prograf     1 mg    3    NA twice a day
## 6 tacpid1_2008-12-16_note3_1.txt    Prograf     <NA> <NA>    NA         bid
## 7 tacpid1_2008-12-16_note3_1.txt    Prograf     <NA> <NA>    NA        <NA>
##   dosestr dosechange lastdose drugname_start
## 1    <NA>       <NA>     <NA>            930
## 2     3mg       <NA>   8:30pm           2709
## 3    <NA>       <NA>    14 hr            618
## 4    <NA>       <NA>     <NA>            722
## 5    <NA>       <NA>     <NA>            761
## 6     2mg   decrease     <NA>           2179
## 7    <NA>       <NA> 10:30 pm           2205
```

```
(lam_part_i_out <- buildDose(lam_mxr_parsed))
```

```
##                         filename       drugname strength dose route
## 1  lampid1_2016-02-05_note4_1.txt       Lamictal     <NA> <NA>  <NA>
## 2  lampid1_2016-02-05_note4_1.txt    Lamotrigine    200mg  1.5  <NA>
## 3  lampid1_2016-02-05_note4_1.txt Lamotrigine XR   100 mg    3  <NA>
## 4  lampid1_2016-02-05_note4_1.txt Lamotrigine XR   100 mg    2  <NA>
## 5  lampid1_2016-02-05_note5_1.txt            ltg   200 mg  1.5  <NA>
## 6  lampid1_2016-02-05_note5_1.txt         ltg xr   100 mg    3  <NA>
## 7  lampid1_2016-02-05_note5_1.txt         ltg xr   100 mg    2  <NA>
## 8  lampid2_2008-07-20_note6_1.txt    lamotrigine     <NA> <NA>  <NA>
## 9  lampid2_2008-07-20_note6_1.txt       lamictal     <NA> <NA>  <NA>
## 10 lampid2_2008-07-20_note6_1.txt       Lamictal     <NA> <NA>  <NA>
## 11 lampid2_2012-04-15_note7_1.txt    lamotrigine   150 mg <NA>  <NA>
## 12 lampid2_2012-04-15_note7_1.txt       Lamictal     <NA>    1  <NA>
##           freq dosestr dosechange lastdose drugname_start
## 1          BID  300 mg       <NA>     <NA>            810
## 2  twice daily    <NA>       <NA>     <NA>            847
## 3      morning    <NA>       <NA>     <NA>            954
## 4      evening    <NA>       <NA>     <NA>            954
## 5        daily    <NA>       <NA>     <NA>            442
## 6        in am    <NA>       <NA>     <NA>            465
## 7        in pm    <NA>       <NA>     <NA>            465
## 8         <NA>    <NA>       <NA>     <NA>           1267
## 9         q12h  150 mg       <NA>     <NA>           1280
## 10         BID   200mg   Increase     <NA>           2273
## 11        <NA>    <NA>       <NA>     <NA>            103
## 12 twice a day    <NA>       <NA>     <NA>            141
```

If the `checkForRare` argument is set to `TRUE`, any extracted expressions with a proportion of occurrence less than 0.2 are returned as rare values. When rare values are identified, a warning is printed to notify the user. The `var` column indicates the entity (note that `dose` in this output refers to dose amount, while `dosestr` would indicate dose given intake). This can be used as a quick check for potentially inaccurate information and allow the user to remove incorrect extractions before applying the Pro-Med-NLP module as incorrect extractions would reduce accuracy of the dose building data. Note that these values may still be correct extractions even though they are rare, as is the case for our output below.

```
lam_checkForRare <- buildDose(lam_mxr_parsed, checkForRare=TRUE)
```

```
## Warning in buildDose(lam_mxr_parsed, checkForRare = TRUE): rare values found
```

```
##        var   val Freq       Prop
## 1 drugname   ltg    1 0.08333333
## 2 strength 150mg    1 0.14285714
## 3     dose     1    1 0.14285714
```

Part II

In Part II of the algorithm, we form the final analysis datasets containing computed dosing information at the note and date level for each patient. This process requires more detailed meta data associated with each clinical note file, the format of which is described below. In this part, we also discuss how time of last dose should be incorporated when present before producing the final output dataset.

noteMetaData The meta data argument is required by the functions `processLastDose` and `collapseDose`, and requires four columns: `filename`, `pid`, `date`, `note`. In our example data, pid (patient ID), date, and note can all be extracted from the filename. Take the filename "tacpid1_2008-06-26_note1_1.txt" for example. It contains information in the form "[PID]_[date]_[note]", where PID = "tacpid1", date = "2008-06-26" and note = "note1". The function below can build our meta data from each of the filenames.

```
bmd <- function(x) {
  fns <- strsplit(x, '_')
  pid <- sapply(fns, `[`, 1)
  date <- as.Date(sapply(fns, `[`, 2), format = '%Y-%m-%d')
  note <- sapply(fns, `[`, 3)
  data.frame(filename = x, pid, date, note, stringsAsFactors = FALSE)
}
bmd("tacpid1_2008-06-26_note1_1.txt")
```

```
##                          filename     pid       date  note
## 1 tacpid1_2008-06-26_note1_1.txt tacpid1 2008-06-26 note1
```

```
(tac_metadata <- bmd(tac_part_i_out[['filename']]))
```

```
##                          filename     pid       date  note
## 1 tacpid1_2008-06-26_note1_1.txt tacpid1 2008-06-26 note1
## 2 tacpid1_2008-06-26_note1_1.txt tacpid1 2008-06-26 note1
## 3 tacpid1_2008-06-26_note2_1.txt tacpid1 2008-06-26 note2
## 4 tacpid1_2008-12-16_note3_1.txt tacpid1 2008-12-16 note3
## 5 tacpid1_2008-12-16_note3_1.txt tacpid1 2008-12-16 note3
## 6 tacpid1_2008-12-16_note3_1.txt tacpid1 2008-12-16 note3
## 7 tacpid1_2008-12-16_note3_1.txt tacpid1 2008-12-16 note3
```

Handling `lastdose` In this section, we cover how incorporation of the last dose entity should be handled if it was extracted during the Extract-Med module. From the `buildDose` output above, we see the raw last dose time extractions for the tacrolimus dataset. Using the functions `processLastDose` and `addLastDose`, we convert the extracted times into a processed and standardized datetime variable, and add the processed times to the `buildDose` output.

The `processLastDose` function requires the following arguments:

- `mxrData`: raw output from the `extractMed` function

- `noteMetaData`: note meta data for each file name in `mxrData`

- `labData`: a data frame containing lab dates and times associated with the file names within `mxrData`. This must contain at a minimum the columns `pid` and `date` (in the same format as `noteMetaData`), as well as `labtime`, a POSIXct variable indicating the date and time of a laboratory drug measurement

Extracted last dose times can fall into two categories: a time expression (e.g., "10am", "22:00", "7 last night") or a duration expression (e.g. "14 hour" level), where the "time" of last dose indicates the number of hours since the last dose was taken relative to the time of the clinical visit. In the latter case, the lab time (from the `labData` argument) is needed in order to convert the extracted duration expression into a datetime variable. Below is an example lab dataset for our sample tacrolimus data.

```
data(tac_lab, package = 'EHR')
tac_lab
```

```
##       pid       date             labtime
## 1 tacpid1 2008-06-26 2008-06-26 10:42:00
## 2 tacpid1 2008-12-16 2008-12-16 12:11:00
```

Within `processLastDose`, extracted times are converted to time expressions of the format "HH:MM:SS" and assigned a date based on the date of the corresponding note. When the last dose time is after 12pm, it is assumed to have been taken on the previous date.

```
(tac_ld <- processLastDose(mxrData = tac_mxr, noteMetaData = tac_metadata, labData = tac_lab))
```

```
##                          filename            lastdose    ld_pos     pid
## 1 tacpid1_2008-06-26_note1_1.txt 2008-06-25 20:30:00 2740:2746 tacpid1
## 2 tacpid1_2008-06-26_note2_1.txt 2008-06-25 20:42:00   678:683 tacpid1
## 3 tacpid1_2008-12-16_note3_1.txt 2008-12-15 22:30:00 2231:2239 tacpid1
##         date raw_time ld_start time_type             labtime
## 1 2008-06-26   8:30pm     2740      time 2008-06-26 10:42:00
## 2 2008-06-26    14 hr      678  duration 2008-06-26 10:42:00
## 3 2008-12-16 10:30 pm     2231      time 2008-12-16 12:11:00
```

The function output contains the processed and standardized last dose time (`lastdose`), the original extracted expression (`raw_time`), whether the raw expression was a time or duration (`time_type`), as well as position information for the last dose time (`ld_start`) for appropriate pairing with dosing information in `addLastDose`. The `labtime` column in the output above corresponds to the information provided in the `labData` argument.

The `addLastDose` function requires the following arguments:

- `buildData`: output from `buildDose`
- `lastdoseData`: dataset containing last dose time information for the file names in `buildData`. This should include columns for `filename` and `lastdose`, with `lastdose` being a processed POSIXct datetime variable.

In the case where last dose information was extracted from clinical notes using medExtractR, the `lastdoseData` input should be output from the `processLastDose` function containing the last dose start positions, as demonstrated below. It is possible for multiple times to be extracted from a clinical note. For extracted times within a 2 hour window of one another, `addLastDose` treats these as equivalent and extracts the last dose time. Note that this may be context-dependent, and this rule was determined based on drugs administered every 12 hours and assuming a trough drug level. For time differences of more than two hours, the last dose start position is used to pair the extracted time with the closest drug mention. Alternatively, if the user has a separate dataset with validated last dose times, they can provide their own dataset. When providing a validated dataset, there should be only one last dose time per patient ID and date.

```
(tac_part_i_out_lastdose <- addLastDose(buildData = tac_part_i_out, lastdoseData = tac_ld))
```

```
##                          filename   drugname strength dose route        freq
```

```
## 1 tacpid1_2008-06-26_note1_1.txt    Prograf     1 mg    3    NA twice a day
## 2 tacpid1_2008-06-26_note1_1.txt    prograf     <NA> <NA>    NA         bid
## 3 tacpid1_2008-06-26_note2_1.txt    Prograf     1 mg    3    NA twice a day
## 4 tacpid1_2008-12-16_note3_1.txt Tacrolimus     <NA> <NA>    NA        <NA>
## 5 tacpid1_2008-12-16_note3_1.txt    Prograf     1 mg    3    NA twice a day
## 6 tacpid1_2008-12-16_note3_1.txt    Prograf     <NA> <NA>    NA         bid
## 7 tacpid1_2008-12-16_note3_1.txt    Prograf     <NA> <NA>    NA        <NA>
##   dosestr dosechange            lastdose drugname_start
## 1    <NA>       <NA>                <NA>            930
## 2     3mg       <NA> 2008-06-25 20:30:00           2709
## 3    <NA>       <NA> 2008-06-25 20:42:00            618
## 4    <NA>       <NA>                <NA>            722
## 5    <NA>       <NA>                <NA>            761
## 6     2mg   decrease                <NA>           2179
## 7    <NA>       <NA> 2008-12-15 22:30:00           2205
```

Note that in the `lastdose` columns, we now have standardized datetime objects instead of the raw extracted expressions.

`collapseDose` The main function used in Part II of the algorithm is the `collapseDose` function, which relies on the underlying `makeDose` function. `collapseDose` allows the user to split the data using drug names given by regular expressions (`...`) and run `makeDose` separately on each subset of the data. For example, if the data includes multiple drugs, regular expressions can be specified for each drug. Another use of this function is to split the data by different formulations of the drug, such as separating immediate release formulations from extended release formulations, which are often written using "XR" or "ER" in the drug name.

The `collapseDose` function requires the following arguments:

- `x`: output from the `buildDose` function, or from `addLastDose` if last dose information is incorporated
- `noteMetaData`: a `data.frame` with columns for filename, pid (patient id), date, and note
- `naFreq`: method to use when assigning missing frequencies; the default is to assign the most common frequency

The general function call is:

```
collapseDose(x, noteMetaData, naFreq = 'most', ...)
```

The main function underlying `collapseDose` is `makeDose`, which standardizes entities, imputes missing values, calculates dose intake and daily dose, removes redundancies, and generates the final dose data. Two `data.frames` are generated from the `makeDose` function, one with redundancies removed at the note level and one at the date level (see McNeer et al. (2020) for details). The `collapseDose` function serves as a wrapper to `makeDose` to ensure results are collated and formatted properly for analysis.

Running `collapseDose` For our tacrolimus example above, the output of this function is below. Note that we use the output from `addLastDose` rather than directly from `buildDose`.

```
tac_part_ii <- collapseDose(tac_part_i_out_lastdose, tac_metadata, naFreq = 'most')
```

Note level collapsing:

```
tac_part_ii$note
```

```
##                        filename drugname strength dose  route freq dosestr
## 1 tacpid1_2008-06-26_note1_1.txt  Prograf     1 mg    3 orally  bid    <NA>
## 2 tacpid1_2008-06-26_note2_1.txt  Prograf     1 mg    3 orally  bid    <NA>
## 3 tacpid1_2008-12-16_note3_1.txt  Prograf     1 mg    3 orally  bid    <NA>
```

```
## 4 tacpid1_2008-12-16_note3_1.txt  Prograf     <NA> <NA> orally  bid     2mg
##   dosechange            lastdose drugname_start dosestr.num strength.num
## 1       <NA> 2008-06-25 20:30:00            930          NA            1
## 2       <NA> 2008-06-25 20:42:00            618          NA            1
## 3       <NA> 2008-12-15 22:30:00            761          NA            1
## 4   decrease 2008-12-15 22:30:00           2179           2           NA
##   doseamt.num freq.num dose.intake intaketime dose.seq dose.daily
## 1           3        2           3       <NA>       NA          6
## 2           3        2           3       <NA>       NA          6
## 3           3        2           3       <NA>       NA          6
## 4          NA        2           2       <NA>       NA          4
```

Date level collapsing:

```
tac_part_ii$date
```

```
##                         filename drugname strength dose  route freq dosestr
## 1 tacpid1_2008-06-26_note1_1.txt  Prograf     1 mg    3 orally  bid    <NA>
## 2 tacpid1_2008-06-26_note2_1.txt  Prograf     1 mg    3 orally  bid    <NA>
## 3 tacpid1_2008-12-16_note3_1.txt  Prograf     1 mg    3 orally  bid    <NA>
## 4 tacpid1_2008-12-16_note3_1.txt  Prograf     <NA> <NA> orally  bid     2mg
##   dosechange            lastdose drugname_start dosestr.num strength.num
## 1       <NA> 2008-06-25 20:30:00            930          NA            1
## 2       <NA> 2008-06-25 20:42:00            618          NA            1
## 3       <NA> 2008-12-15 22:30:00            761          NA            1
## 4   decrease 2008-12-15 22:30:00           2179           2           NA
##   doseamt.num freq.num dose.intake intaketime dose.seq dose.daily
## 1           3        2           3       <NA>       NA          6
## 2           3        2           3       <NA>       NA          6
## 3           3        2           3       <NA>       NA          6
## 4          NA        2           2       <NA>       NA          4
```

Below, we demonstrate `collapseDose` using our lamotrigine example. In the function call, we supply an additional argument `'xr|er'` to indicate that we want to separately consider extended release formulations of lamotrigine, (usually denoted by "XR" or "ER"). This prevents regular lamotrigine mentions from being collapsed with lamotrigine XR mentions, even if the dosage is identical.

```
data(lam_metadata, package = 'EHR')
lam_part_ii <- collapseDose(lam_part_i_out, lam_metadata, naFreq = 'most', 'xr|er')
```

Note level collapsing:

```
lam_part_ii$note
```

```
##                         filename       drugname strength dose  route  freq
## 1 lampid1_2016-02-05_note4_1.txt       Lamictal     <NA> <NA> orally   bid
## 2 lampid1_2016-02-05_note4_1.txt Lamotrigine XR   100 mg    3 orally    am
## 3 lampid1_2016-02-05_note4_1.txt Lamotrigine XR   100 mg    2 orally    pm
## 4 lampid1_2016-02-05_note5_1.txt            ltg   200 mg  1.5 orally daily
## 5 lampid1_2016-02-05_note5_1.txt         ltg xr   100 mg    3 orally    am
## 6 lampid1_2016-02-05_note5_1.txt         ltg xr   100 mg    2 orally    pm
## 7 lampid2_2008-07-20_note6_1.txt       lamictal     <NA> <NA> orally   bid
## 8 lampid2_2008-07-20_note6_1.txt       Lamictal     <NA> <NA> orally   bid
## 9 lampid2_2012-04-15_note7_1.txt       Lamictal     <NA>    1 orally   bid
##   dosestr dosechange lastdose drugname_start dosestr.num strength.num
## 1  300 mg       <NA>     <NA>            810         300           NA
## 2    <NA>       <NA>     <NA>            954          NA          100
```

```
## 3    <NA>       <NA>     <NA>            954          NA          100
## 4    <NA>       <NA>     <NA>            442          NA          200
## 5    <NA>       <NA>     <NA>            465          NA          100
## 6    <NA>       <NA>     <NA>            465          NA          100
## 7  150 mg       <NA>     <NA>           1280         150           NA
## 8   200mg   Increase     <NA>           2273         200           NA
## 9    <NA>       <NA>     <NA>            141          NA          150
##   doseamt.num freq.num dose.intake intaketime dose.seq dose.daily
## 1          NA        2         300       <NA>       NA        600
## 2         3.0        1         300         am        1        500
## 3         2.0        1         200         pm        2        500
## 4         1.5        1         300       <NA>       NA        300
## 5         3.0        1         300         am        1        500
## 6         2.0        1         200         pm        2        500
## 7          NA        2         150       <NA>       NA        300
## 8          NA        2         200       <NA>       NA        400
## 9         1.0        2         150       <NA>       NA        300
```

Date level collapsing:

```
lam_part_ii$date
```

```
##                         filename       drugname strength dose  route  freq
## 1 lampid1_2016-02-05_note4_1.txt       Lamictal     <NA> <NA> orally   bid
## 2 lampid1_2016-02-05_note4_1.txt Lamotrigine XR   100 mg    3 orally    am
## 3 lampid1_2016-02-05_note4_1.txt Lamotrigine XR   100 mg    2 orally    pm
## 4 lampid1_2016-02-05_note5_1.txt            ltg   200 mg  1.5 orally daily
## 5 lampid2_2008-07-20_note6_1.txt       lamictal     <NA> <NA> orally   bid
## 6 lampid2_2008-07-20_note6_1.txt       Lamictal     <NA> <NA> orally   bid
## 7 lampid2_2012-04-15_note7_1.txt       Lamictal     <NA>    1 orally   bid
##   dosestr dosechange lastdose drugname_start dosestr.num strength.num
## 1  300 mg       <NA>     <NA>            810         300           NA
## 2    <NA>       <NA>     <NA>            954          NA          100
## 3    <NA>       <NA>     <NA>            954          NA          100
## 4    <NA>       <NA>     <NA>            442          NA          200
## 5  150 mg       <NA>     <NA>           1280         150           NA
## 6   200mg   Increase     <NA>           2273         200           NA
## 7    <NA>       <NA>     <NA>            141          NA          150
##   doseamt.num freq.num dose.intake intaketime dose.seq dose.daily
## 1          NA        2         300       <NA>       NA        600
## 2         3.0        1         300         am        1        500
## 3         2.0        1         200         pm        2        500
## 4         1.5        1         300       <NA>       NA        300
## 5          NA        2         150       <NA>       NA        300
## 6          NA        2         200       <NA>       NA        400
## 7         1.0        2         150       <NA>       NA        300
```

Additional collapsing Collapsing by date or note produces observations at the daily intake level. It is possible to further collapse data to the daily level, though you may want to drop the *dose.intake* variable as it would potentially lose meaning.

```
x <- lam_part_ii[['note']]
# retrieve metadata for each filename
k1 <- lam_metadata[match(x[,'filename'], lam_metadata[,'filename']), c('pid','date','note')]
# select additional key data
```

```
k2 <- cbind(k1, x[,c('dose.daily','drugname_start')])
# turn keys into character string
chk <- do.call(paste, c(k2, sep = '|'))
# keep first instance of each chk key
lam_part_iii_note <- x[!duplicated(chk),]
lam_part_iii_note[,c('filename','drugname','drugname_start','dose.daily')]
```

```
##                          filename       drugname drugname_start dose.daily
## 1 lampid1_2016-02-05_note4_1.txt       Lamictal            810        600
## 2 lampid1_2016-02-05_note4_1.txt Lamotrigine XR            954        500
## 4 lampid1_2016-02-05_note5_1.txt            ltg            442        300
## 5 lampid1_2016-02-05_note5_1.txt         ltg xr            465        500
## 7 lampid2_2008-07-20_note6_1.txt       lamictal           1280        300
## 8 lampid2_2008-07-20_note6_1.txt       Lamictal           2273        400
## 9 lampid2_2012-04-15_note7_1.txt       Lamictal            141        300
```

```
x <- lam_part_ii[['date']]
# ignore note for date level collapsing
k1 <- lam_metadata[match(x[,'filename'], lam_metadata[,'filename']), c('pid','date')]
k2 <- cbind(k1, x[,c('dose.daily','drugname_start')])
chk <- do.call(paste, c(k2, sep = '|'))
lam_part_iii_date <- x[!duplicated(chk),]
lam_part_iii_date[,c('filename','drugname','drugname_start','dose.daily')]
```

```
##                          filename       drugname drugname_start dose.daily
## 1 lampid1_2016-02-05_note4_1.txt       Lamictal            810        600
## 2 lampid1_2016-02-05_note4_1.txt Lamotrigine XR            954        500
## 4 lampid1_2016-02-05_note5_1.txt            ltg            442        300
## 5 lampid2_2008-07-20_note6_1.txt       lamictal           1280        300
## 6 lampid2_2008-07-20_note6_1.txt       Lamictal           2273        400
## 7 lampid2_2012-04-15_note7_1.txt       Lamictal            141        300
```

References

1. Choi L, Beck C, McNeer E, Weeks HL, Williams ML, James NT, Niu X, Abou-Khalil BW, Birdwell KA, Roden DM, Stein CM. Development of a System for Post-marketing Population Pharmacokinetic and Pharmacodynamic Studies using Real-World Data from Electronic Health Records. Clinical Pharmacology & Therapeutics. 2020 Jan 20.
2. Weeks HL, Beck C, McNeer E, Williams ML, Bejan CA, Denny JC, Choi L. medExtractR: A targeted, customizable approach to medication extraction from electronic health records. Journal of the American Medical Informatics Association. 2020 Jan 16.
3. Weeks, HL, Beck C, and Choi, L (2019). medExtractR: Extraction of Medication Information from Clinical Text. R package version 0.1. https://CRAN.R-project.org/package=medExtractR
4. Stenetorp P, Pyysalo S, Topić G, Ohta T, Ananiadou S, Tsujii JI. BRAT: a web-based tool for NLP-assisted text annotation. InProceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics 2012 Apr 23 (pp. 102-107). Association for Computational Linguistics.
5. McNeer E, Beck C, Weeks HL, Williams ML, James NT, Choi L. A post-processing algorithm for building longitudinal medication dose data from extracted medication information using natural language processing from electronic health records. bioRxiv https://www.biorxiv.org/content/10.1101/775015v3 (2020).