Using MSPrep

***Matt Mulvahill*[*1], *Grant Hughes*[†2], *Sean Jacobson*[‡2], *Katerina Kechris*[§1], *and Harrison Pielke-Lombardo*[¶1]**

[1]UC Anschutz
[2]National Jewish Hospital

*matthew.mulvahill@ucdenver.edu †fill@me.in ‡jacobsons@njhealth.org §katerina.kechris@ucdenver.edu
¶harrison.pielke-lombardo@ucdenver.edu

21 March 2019

Package

MSPrep 0.0.2

Contents

1 Introduction

MSPrep provides the five key functions for preparing metabolomics data for analysis. This vignette will provide - an explanation what each of the functions does - suggestions for how to select parameters - an explanation of how to use the final outputted data - code to perform all of the steps in a pipeline

2 Loading libraries

```
library(MSPrep)
library(tidyverse)
```

3 Getting our columns in order

In order to allow MSPrep to do the heavy lifting, we first need to get our data into the correct format. Most mass spectrometry data will have a similar format, so not much should be required to get it ready.

Two columns are for the mass-to-charge ratio and the retention-time. They can have any name you'd like (usually "mz" and "rt").

The other columns are for the other sample variables. These columns should have names that contain information about their contents which will be used later in the pipeline. There are three pieces of information which need to be present at the end of each column name, separated by a consistent separator. These are the spike, batch, and replicate ID.

As an example take a look at the provided dataset, msquant.

```
data(msquant)
colnames(msquant)[1: 6]
## [1] "mz"
## [2] "rt"
## [3] "Neutral_Operator_Dif_Pos_1x_01_A_01"
## [4] "Neutral_Operator_Dif_Pos_1x_01_B_01"
## [5] "Neutral_Operator_Dif_Pos_1x_01_C_01"
## [6] "Neutral_Operator_Dif_Pos_1x_02_A_01"
```

The third column name is `colnames(msquant)[3]`. The first part "Neutral_Operator_Dif_Pos_" won't be used, so we will assign it to col_extra_txt. The next value, "1x", is the spike. The following value, "O1", is the batch. The remaining values, "A" and "01", are the replicate and subject IDs.

Identifiers in the datasets - LCMS_Run_ID = operator/replicate (A-C), subject (01-03), concentration (1x,2x,4x) - SubjectID = subject (01-03), concentration (1x,2x,4x)

With our data in this format, we can start the pipeline.

4 Tidying the data

The first step is to extract the information from the column names discussed above, and get it into a tidy data frame. We can do this with the ms_tidy function.

```
tidied_data = ms_tidy(msquant, mz = "mz", rt = "rt", col_extra_txt =
                        "Neutral_Operator_Dif_Pos_", separator = "_",
                      col_names = c("spike","batch", "replicate", "subject_id"))
```

Note, the names chosen for col_name are arbitratry, but we will use them later on.

5 Preparing the data

This step summarizes the technical replicates. It does this using the following procedure for each compound in each batch.

1. If there are less than a minimum proportion of the values found among the replicates (usually one or zero), leave value empty. Otherwise proceed.
2. Calculate the coefficient of variation between the replicates using $c_v = \frac{\sigma}{\mu}$.
3. For three replicates, if the coefficient of variation is above a specified level, use the median value for the compound, to correct for the large dispersion.
4. Otherwise, use the mean value of the replicates for the compound.

```
summarized_data = ms_prepare(tidied_data, mz = "mz", rt = "rt", replicate = "replicate", batch = "batch", gr
```

6 Filtering missing compounds

This step is pretty straight forward but very important. Simply supply a percentage of the number of samples for which a compound needs to have data present for in order to be kept.

```
filtered_data = ms_filter(summarized_data, filter_percent=0.8)
```

7 Imputing missing values

Next, you'll need to make a decision about how you'd like to handle the remaining missing data.There are three methods provided. 1. half-min (half the minimum value) 2. bpca (Bayesian PCA), 3. knn (k-nearest neighbors)

Half-min is the fastest and is often sufficient. KNN typically takes the longest. If you choose to use KNN, you can provide a value for k.

```
imputed_data = ms_impute(filtered_data, method ="knn", k = 5)
```

8 Normalizing the data

In order to make comparisons between compounds, the data need to be normalized. This step performs one of six normalization strategies. 1. ComBat () 1. quantile + ComBat 2. median + ComBat 3. CRMN () 4. RUV () 5. SVA ()

For experiments which have control compounds, a list of the column numbers containing them should be provided in the controls parameter. Otherwise, simply leave the controls parameter blank or NULL.

```
normalized_data = ms_normalize(imputed_data, method ="CRMN", controls = NULL,  n_comp = 2, n_control = 10)
## Number of significant surrogate variables is:  4
## Iteration (out of 5 ):1  2  3  4  5
```

9 Final output

10 Performing all steps as a pipline

Often, you will want to just perform the whole pipeline. This can easily be done in a single statement using the %>% operator from the magrittr package. This threads the result of each function into the first position of the next.

```
dat <-
  msquant %>%
  ms_tidy %>%
  ms_prepare(replicate = "replicate", batch = "batch", groupingvars = "spike") %>%
  ms_filter(0.8) %>%
  ms_impute(method = "halfmin") %>%
  ms_normalize(method = "quantile + ComBat")
## Found3batches
## Adjusting for2covariate(s) or covariate level(s)
## Standardizing Data across genes
## Fitting L/S model and finding priors
## Finding parametric adjustments
## Adjusting the Data
```