Data preparation: filling or removing the NAs

Andreas Hadjiprocopis

February, 2018

```
#!/usr/bin/env Rscript
```

```
source('lib/IO.R');
source('lib/NA.R');

infile='original_data/dat1.csv'
outdir='cleaned_data';
outcleanfile=file.path(outdir, 'dat1.clean.csv')
outeliminateNAfile=file.path(outdir, 'dat1.eliminateNA.csv')
```

Read the original csv data file (dat1.csv)

```
dat1 <- data.frame(read_data(
    filename=infile
))
```

```
## read_data(): data read from file 'original_data/dat1.csv'.
```

```
if( is.null(dat1) ){
    cat("call to read_data() has failed for file '",infile,"'.\n", sep='')
    quit(status=1)
}
```

Remove all rows with at least one NA

```
results <- clean_dataset(
    inp=dat1,
    methods=c('remove_entire_row')
)
if( is.null(results) ){
    cat("call to clean_dataset() has failed for file '",infile,"'.\n", sep='')
    quit(status=1)
}
```

this is the clean data

```
clean_dat1 <- results$imputed
```

And save it

```
if( ! save_data(clean_dat1, outeliminateNAfile) ){
    cat("call to save_data() has failed for file '",outeliminateNAfile,"'.\n", sep='')
    quit(status=1)
}
```

```
## save_data(): data saved to file 'cleaned_data/dat1.eliminateNA.csv'.
```

Alternatively one can replace (fill) NAs with guesses. See notes at the end of this document. But we will not do it now because it takes quite some time. Uncomment this to try it out.

```
if( FALSE ){
results <- clean_dataset(
```

```
    inp=dat1,
    methods=c("mice", "kNN"),
    rng.seed=1234,
    # how many artificial NAs to introduce?
    random_NAs_to_introduce_in_each_column_percent=5/100,
    # number of repeats for the assessment
    repeats=16,
    # parallelise the procedure over so many CPU cores
    ncores=8
)
if( is.null(results) ){
    #cat("call to clean_dataset() has failed for file '",infile,"'.\n", sep='')
    quit(status=1)
}
}
```

this is the cleaned data as a list

```
clean_dat1 <- results$imputed
```

Statistics of the filling NAs procedure

```
stats_of_imputation <- results$stats
print(stats_of_imputation)
```

```
## NULL
```

finally, save the data

```
if( ! save_data(clean_dat1, outcleanfile) ){
    cat("call to save_data() has failed for file '",outcleanfile,"'.\n", sep='')
    quit(status=1)
}
```

```
## save_data(): data saved to file 'cleaned_data/dat1.clean.csv'.
```

The input data contains quite a few NAs I have created a few functions in lib/NA.R which either remove all rows with at least one NA or try to fill the NA with an appropriate value based on the other values in the row. Remove the NAs is straightforward but filling them out risks distorting the dataset. I have used two methods for filling: 1. kNN 2. mice I have created a function to assess which method works best. This is done as follows. 1. Remove all NAs from the dataset. 2. Create some random NAs in the clean dataset, say 5% of the total items. 3. Apply each method to filling the artificial NAs. 4. Calculate a metric of badness of fill by, say, simple root mean square of the difference between filled and actual values. 5. Report the method yielding the lowest value. 6. Repeat this process N times. Method 'kNN' has done consistently better. And so this is the method used. The main function is clean_dataset() which takes parameters to either eliminate rows with NAs or fill NAs. Additionally, input parameters control the number of repeats should a filling-NAs route was taken, the percentage of artificial NAs to introduce and the methods to assess, and the column names to process. A very important feature of the clean_dataset() function is that assessment, which is quite time-consuming, can be *parallelised* over as many CPU cores as specified. A *parallelisation* option is important to exist in such methods and I always spend a bit more effort in implementing it, and considerably more in … debugging it. But it is worthy. here is a dump of the NA.R library:

```
clean_dataset <- function(
    inp=NULL,
    methods=c("mice", "kNN"),
    columns_to_do=c("A","B","C","D","E","F"),
    rng.seed=as.numeric(Sys.time()),
    random_NAs_to_introduce_in_each_column_percent=5/100,
```

```
    repeats=5,
    ncores=1
){
    whoami=paste0(match.call()[[1]],'()',collapse='')
    if( methods %in% 'remove_entire_row' ){
        # asked to remove all rows with at least 1 NA
        # no guessing of what an NA could be is required
        dummy <- remove_all_rows_with_NA(inp=inp)
        if( is.null(dummy) ){
            cat(whoami, " : call to remove_all_rows_with_NA() has failed.\n");
            return(NULL)
        }
        ret=list()
        ret[['stats']] = NULL
        ret[['imputed']] = dummy
        return(ret)
    }

    # we are asked to guess NA values
    best <- assess_na_methods_repeatedly(
        inp=inp,
        methods=methods,
        columns_to_do=columns_to_do,
        rng.seed=rng.seed,
        random_NAs_to_introduce_in_each_column_percent=random_NAs_to_introduce_in_each_column_percent,
        repeats=repeats,
        ncores=ncores
    )
    if( is.null(best) ){
        cat("call to assess_na_methods_repeatedly() has failed.\n", sep='')
        return(NULL)
    }
    recom_method = best$recommended_method
    if( recom_method == 'none' ){ recom_method = best$best_method_wrt_mean }

    clean_data = NULL;
    if( recom_method == 'kNN' ){
        clean_data <- deal_with_na_using_kNN(
            inp=inp,
            columns_to_do=columns_to_do,
            rng.seed=rng.seed
        )
        if( is.null(clean_data) ){
            cat(whoami, " : call to deal_with_na_using_kNN() has failed.\n", sep='')
            return(NULL)
        }
    } else {
        clean_data <- deal_with_na_using_mice(
            inp=inp,
            columns_to_do=columns_to_do,
            rng.seed=rng.seed
        )
        if( is.null(clean_data) ){
```

```r
            cat(whoami, " : call to deal_with_na_using_mice() has failed.\n", sep='')
            return(NULL)
        }
    }
    ret=list()
    ret[['stats']] = best
    ret[['imputed']] = clean_data
    return(ret)
}
remove_all_rows_with_NA <- function(
    inp=NULL
){
    # asked to remove all rows with at least 1 NA
    # no guessing of what an NA could be is required
    return(na.omit(inp))
}

assess_na_methods_repeatedly <- function(
    inp=NULL,
    methods=c("mice", "kNN"),
    columns_to_do=c("A","B","C","D","E","F"),
    rng.seed=as.numeric(Sys.time()),
    random_NAs_to_introduce_in_each_column_percent=5/100,
    repeats=10,
    ncores=1
){
    library(parallel)
    whoami=paste0(match.call()[[1]],'()',collapse='')

    cat(whoami, " : spawning ", repeats, " processes over ", ncores, " cores.\n", sep='')
    set.seed(rng.seed)
    seeds = sample.int(n=1000, size=repeats)
    num_methods = length(methods)
    # mclapply will return a list of results, one for each process spawn,
    # some processes might be failures, in this case the result will be of class 'try-error'
    # how does that work:
    # 1. all is done within system.time() in order to measure time taken
    # 2.
    stime <- system.time({
      res <-mclapply(
        X=seeds,
        function(aseed){
            assess_na_methods_once(
                inp,
                methods,
                columns_to_do,
                aseed,
                random_NAs_to_introduce_in_each_column_percent
            )
        },
        mc.cores=ncores
      ) # mclapply
    }) # system.time
```

```
    print(res) # debug
    # remove all test results which are NULL (crash, failed etc.)
    clean_res=list()
    num_failed = 0; num_clean_res = 0
    for(i in 1:repeats){
        if( is.null(res[[i]]) || (class(res[[i]]) == "try-error") ){
            cat(whoami, " : repeat #",i," has failed.\n", sep='');
            print(is.null(res[[i]]))
            num_failed = num_failed + 1
        } else {
            num_clean_res = num_clean_res+1
            clean_res[[num_clean_res]] = res[[i]]
        }
    }
    if( num_failed > 0 ){
        cat(whoami, " : ", num_failed, " repeats (of ", repeats, ") have failed.\n", sep='')
    } else {
        cat(whoami, " : all ", repeats, " repeats have succeeded.\n", sep='')
    }

    stats=matrix(0, nrow=num_methods, ncol=2)
    rownames(stats) <- methods
    colnames(stats) <- c('wins', 'mean')
    mean_assessment=0
    print(clean_res)
    for(i in 1:num_clean_res){
        print(clean_res[[i]])
        winning_method = clean_res[[i]][['best_method']]
        stats[winning_method,'wins'] = stats[winning_method,'wins']+1
        for(amethod in methods){
            mean_got = clean_res[[i]][['assessment']][[amethod]][['mean']]
            stats[amethod,'mean'] = stats[amethod,'mean']+mean_got/num_clean_res
        }
    }
    best_method_wrt_wins = names(which(stats[,'wins'] == max(stats[,'wins'])))
    best_method_wrt_mean = names(which(stats[,'mean'] == min(stats[,'mean'])))
    best_wins = stats[best_method_wrt_wins,'wins']
    best_mean = stats[best_method_wrt_mean,'mean']
    ret = list()
    ret[['individual_results']] = ret
    ret[['stats']] = stats
    ret[['best_method_wrt_wins']] = best_method_wrt_wins
    ret[['best_method_wrt_mean']] = best_method_wrt_mean
    ret[['best_wins']] = best_wins
    ret[['best_mean']] = best_mean
    if( best_method_wrt_wins == best_method_wrt_mean ){
        touse = best_method_wrt_mean
    } else {
        #no consensus
        touse = 'none'
    }
    ret[['recommended_method']] = touse
    cat(whoami, " : after ", repeats, " repeats here is the overall assessment:\n", sep='')
```

```
    print(stats)
    cat("best method wrt to mean: ", best_method_wrt_mean, "\n", sep='')
    cat("best method wrt to counting best performance (wins): ", best_method_wrt_wins, "\n", sep='')
    cat("recommended method for NA imputation: ", touse, "\n", sep='')
    cat(whoami, " : finished ", repeats, " repeats in ", stime[3], " seconds.\n", sep='')
    return(ret)
}
assess_na_methods_once <- function(
    inp=NULL,
    methods=c("mice", "kNN"),
    columns_to_do=c("A","B","C","D","E","F"),
    rng.seed=as.numeric(Sys.time()),
    random_NAs_to_introduce_in_each_column_percent=5/100
){
    whoami=paste0(match.call()[[1]],'()',collapse='')

    # First remove all NAs from input
    cleaned_inp = inp[complete.cases(inp), ]

    nrows = nrow(cleaned_inp)
    ncols_to_do = length(columns_to_do)
    nNAs_per_column = round(nrows * random_NAs_to_introduce_in_each_column_percent)

    idx_of_NAs_in_columns = list();
    # then introduce our own NAs for each column
    # first get a random set of indices for each column:
    total_NAs = 0
    for(acol in columns_to_do){
        idx_of_NAs_in_columns[[acol]] = sample(1:nrows, nNAs_per_column)
    }
    cat(whoami, " : ", (nNAs_per_column*ncols_to_do), " NAs were added in total over all columns (conta

    # then set them to NA
    cleaned_inp_with_NAs = cleaned_inp
    for(acol in columns_to_do){
        idx_for_NA_for_this_col = idx_of_NAs_in_columns[[acol]]
        cleaned_inp_with_NAs[[acol]][idx_for_NA_for_this_col] <- NA
    }

    # now call each method for imputing our random but controlled NAs
    imputed = NULL
    best_mean = -1
    best_method = NULL
    assessment = list()
    mean_assessment = list()
    for(amethod in methods){
        # inputed will be a list with NAs completed by the methods
        if( amethod == "kNN" ){
            imputed = deal_with_na_using_kNN(
                inp=cleaned_inp_with_NAs,
                columns_to_do=columns_to_do
            )
            if( is.null(imputed) ){
```

```
                cat(whoami, " : call to deal_with_na_using_kNN() has failed.\n", sep='')
                return(NULL)
            }
        } else if( amethod == "mice" ){
            imputed = deal_with_na_using_mice(
                inp=cleaned_inp_with_NAs,
                columns_to_do=columns_to_do
            )
            if( is.null(imputed) ){
                cat(whoami, " : call to deal_with_na_using_mice() has failed.\n", sep='')
                return(NULL)
            }
        } else {
            cat(whoami, " : method '",amethod,"' is not known.\n", sep='')
            return(NULL)
        }
        # and do the assessment of this specific imputation
        assessment[[amethod]] = c()
        for(acolname in columns_to_do){
            assessment[[amethod]][acolname] = calculate_discrepancy(
                imputed[[acolname]],
                cleaned_inp[[acolname]],
                idx_of_NAs_in_columns[[acolname]]
            )
        }
        assessment[[amethod]]['mean'] = mean(assessment[[amethod]])
        if( is.null(best_method) || (assessment[[amethod]]['mean'] < best_mean) ){
            best_mean = assessment[[amethod]]['mean']
            best_method = amethod
        }
    } # for methods
    # we are looking for the lowest mean discrepancy over all columns
    cat(whoami, " : results of assessment:\n", sep='')
    for(amethod in methods){
        print(assessment[[amethod]])
        cat("mean for method '", amethod, "' is ", assessment[[amethod]]['mean'], "\n\n", sep='')
    }
    cat("--------------------------------\n", sep='')
    cat(whoami, " : best method wrt mean assessment over all columns is \"", best_method, "\" with mean
    ret = list()
    ret[['best_method']] = best_method
    ret[['best_mean']] = best_mean
    ret[['assessment']] = assessment
    ret[['seed']] = rng.seed
    return(ret)
}
calculate_discrepancy <- function(
    vector1=NULL,
    vector2=NULL,
    indices=NULL
){
    whoami=paste0(match.call()[[1]],'()',collapse='')
    if( is.null(indices) ){ indices=c(1:nrow(vector1)) }
```

```r
    sum = 0
    for(i in indices){
        sum = sum + (vector1[i]-vector2[i])^2
    }
    return( sqrt(sum/length(indices)) )
}
deal_with_na_using_kNN <- function(
    inp=NULL,
    columns_to_do=c("A","B","C","D","E","F"),
    rng.seed=as.numeric(Sys.time())
){
    library(DMwR)
    whoami=paste0(match.call()[[1]],'()',collapse='')
    columns_to_ignore=setdiff(colnames(inp), columns_to_do)
    # remove those 'ignore' columns from input data before processing
    ignored_columns=list()
    for(acol in columns_to_ignore){
        # move the ignored column temporarily here
        ignored_columns[[acol]] <- inp[[acol]]
        # and erase it from the data to be processed
        inp[[acol]] <- NULL
        cat(whoami, " : column '",acol,"' will not have its NAs imputed, it is temporarily removed.\n",
    }

    cat(whoami, " : calling knnImputation, columns to consider: ", paste0(columns_to_do,collapse=','),
    # use k >= 3 for eliminating the "Error in rep(1, ncol(dist)) : invalid 'times' argument"
    # see https://stackoverflow.com/questions/36239695/error-with-knnimputer-from-the-dmwr-package-inva
    knnOutput <- knnImputation(
        data=inp[,columns_to_do],
        k=5
    )
    if( is.null(knnOutput) ){
        cat(whoami, " : call to knnImputation() has failed.\n", sep='')
        return(NULL)
    }
    # now add the ignored columns back to output
    for(acol in columns_to_ignore){
        knnOutput[[acol]] <- ignored_columns[[acol]]
    }
    cat(whoami, " : done.\n", sep='')
    return(knnOutput)
}
deal_with_na_using_mice <- function(
    inp=NULL,
    columns_to_do=c("A","B","C","D","E","F"),
    rng.seed=as.numeric(Sys.time())
){
    library(mice)
    whoami=paste0(match.call()[[1]],'()',collapse='')

    columns_to_ignore=setdiff(colnames(inp), columns_to_do)
    # remove those 'ignore' columns from input data before processing
    ignored_columns=list()
```

```
    for(acol in columns_to_ignore){
        # move the ignored column temporarily here
        ignored_columns[[acol]] <- inp[[acol]]
        # and erase it from the data to be processed
        inp[[acol]] <- NULL

        # we can leave the ignored column in and use
        # predM[, c(acol)]=0
        # to ignore it, but we remove it.
        cat(whoami, " : column '",acol,"' will not have its NAs imputed, it is temporarily removed.\n",
    }

    init = mice(inp, maxit=0)
    meth = init$method
    predM = init$predictorMatrix

    cat(whoami, " : calling mice ...\n", sep='')
    miceOutput <- mice(
        inp,
        method='pmm',
        predictorMatrix=predM,
        m=6,
        seed=rng.seed
    )
    if( is.null(miceOutput) ){
        cat(whoami, " : call to mice() has failed.\n", sep='')
        return(NULL)
    }
    cat(whoami, " : information from mice:\n", sep='')
    print(miceOutput)
    cat(whoami, " : doing the imputation ...\n", sep='')
    compl <- complete(miceOutput)
    if( is.null(compl) ){
        cat(whoami, " : call to mice() has failed.\n", sep='')
        return(NULL)
    }
    cat(whoami, " : done.\n", sep='')
    return(compl)
}
```

‘‘‘