Data Cleaning in R

John Brandt, Yale F&ES

March 15, 2018

Common Tasks

- Data class
- Creating dataframes
- Updating column names
- Combining columns
- Transposing
- Rownames/column names
- Concatenating
- Creating lists
- Subsetting (indices, logic, which)
- Merging
- Reclassification
- Dealing with NA values w/ logic
- Strings
- Functions
- lapply

Basic background

- dataframe$column
- read.csv(“data.csv”)
- write.csv(object, “object.csv”)
- head()
- str()
- colnames()
- summary()
- unique()
- class()
- levels()
- nrow(), length()
- dataframe[row, column]

Packages

```r
install.packages(c("lubridate",
                   "dplyr",
                   "tidyr",
                   "reshape2",
                   "tibble"))
```

```r
library(lubridate)
library(dplyr)
library(tidyr)
library(reshape2)
library(tibble)
```

Structure

`as.Date`, `as.String`, `as.Numeric`, `as.Factor` use `class()` to find type

```r
class(sales$event_dt)
```

```
[1] "factor"
```

```r
sales$event_dt <- as.Date(sales$event_dt)
class(sales$event_dt)
```

```
[1] "Date"
```

Splitting columns

```
sales$event_dt[1:3]
```

```
[1] "2015-09-12" "2009-09-04" "2006-04-21"
```

```
strsplit(as.character(sales$event_dt[1:3]), "-")
```

```
[[1]]
[1] "2015" "09"   "12"

[[2]]
[1] "2009" "09"   "04"

[[3]]
[1] "2006" "04"   "21"
```

Creating new dataframes

```r
new.df <- data.frame(matrix(nrow=5, ncol=3))
```

Updating column names

```
colnames(new.df) <- c("column.1", "column.2", "column.3")
new.df$column.1 <- c(1,2,3,4,5)
new.df$column.2 <- c("a", "b", "c", "d", "e")
new.df
```

```
  column.1 column.2 column.3
1        1        a       NA
2        2        b       NA
3        3        c       NA
4        4        d       NA
5        5        e       NA
```

Renaming singular column

```
colnames(new.df[colnames(new.df) == "column.2"])
          <- "whatever"
```

Combining columns

```r
new.df$column.3 <- paste(new.df$column.1,
                new.df$column.2, sep="")
new.df$column.3
```

```
[1] "1a" "2b" "3c" "4d" "5e"
```

Transposing

t() returns a nested list. A dataframe must be specified if you want a dataframe.

```
new.df.t <- as.data.frame(t(new.df))
new.df.t
```

```
         V1 V2 V3 V4 V5
column.1  1  2  3  4  5
column.2  a  b  c  d  e
column.3 1a 2b 3c 4d 5e
```

Rownames/colnames

```r
require(tibble)
rownames_to_column(new.df.t)
```

```
   rowname V1 V2 V3 V4 V5
1 column.1  1  2  3  4  5
2 column.2  a  b  c  d  e
3 column.3 1a 2b 3c 4d 5e
```

Concatenating dataframes

```r
new.df.2 <- new.df
rbind(new.df, new.df.2)
```

```
   column.1 column.2 column.3
1         1        a       1a
2         2        b       2b
3         3        c       3c
4         4        d       4d
5         5        e       5e
6         1        a       1a
7         2        b       2b
8         3        c       3c
9         4        d       4d
10        5        e       5e
```

Creating new lists

```
mylist <- rep(NA, nrow(sales)/500)

mylist
```

```
 [1] NA NA NA NA NA NA NA NA NA NA
```

Subsetting by indices

R makes use of the [row,column] notation

```
sales[5:10, c(3,9)]
```

```
         primary_act_id major_cat_name
5  91c03a34b562436efa3c           MISC
6  ac4b847b3fde66f2117e           MISC
7  a14232befff04be1e2f3           MISC
8  91c03a34b562436efa3c           MISC
9  0efaba7ce3f0d7466b42           MISC
10 f6425a3223e73ea6de5a       CONCERTS
```

Subsetting by >=, ==, <=, !=

R accepts logical statements within a [row, column] subsetting argument

```
sales[sales$age_yr>70 & !is.na(sales$age_yr),
      c(34,14,15)]
```

```
     age_yr tickets_purchased_qty trans_face_val_amt
11       80                     1                 20
428      82                     1                 20
1220     80                     1                 20
1314     76                     1                 30
1701     72                     2                 30
3830     72                     1                 22
4188     90                     1                 15
4328     94                     1                  9
4975     78                     1                 20
```

Subsetting by %in%

```
sales[sales$venue_state %in% c("RHODE ISLAND",
                               "MANITOBA"), c(18,26)]
```

```
       event_dt  venue_state
732  2015-08-08 RHODE ISLAND
1860 2015-09-17     MANITOBA
2046 2016-02-19     MANITOBA
2177 2015-12-16 RHODE ISLAND
2832 2015-09-17     MANITOBA
3304 2015-09-17     MANITOBA
4064 2016-01-27     MANITOBA
4119 2015-09-17     MANITOBA
4308 2015-11-16     MANITOBA
4935 2016-01-15     MANITOBA
```

Subsetting by which

If you have 500 columns, you may not know which column index to subset by in the previous example. Here the age column is extracted using `which`.

```
sales[sales$age_yr>70 & !is.na(sales$age_yr),
      c(which(colnames(sales) %in%
                c("age_yr","tickets_purchased_qty")))]
```

```
     tickets_purchased_qty age_yr
11                       1     80
428                      1     82
1220                     1     80
1314                     1     76
1701                     2     72
3830                     1     72
4188                     1     90
4328                     1     94
4975                     1     78
```

Which.max, which.min

`which.max` and `which.min` are useful for removing known outliers.

```r
sales[-c(which.max(sales$income_amt)),]
```

Merging

```r
merged <- merge(x1, x2, by.x="column.x", by.y="column.y")
```

Reclassifying

```r
levels(cut(sales$trans_face_val_amt, 5))
```

```
[1] "(-0.52,305]"         "(305,609]"           "(609,913]"
[4] "(913,1.22e+03]"      "(1.22e+03,1.52e+03]"
```

Removing NA values

```
new.df[2,3] <- NA
new.df[1,2] <- NA
new.df[1,3] <- NA
new.df
```

```
  column.1 column.2 column.3
1        1     <NA>     <NA>
2        2        b     <NA>
3        3        c       3c
4        4        d       4d
5        5        e       5e
```

Na.omit

```r
na.omit(new.df)
```

```
  column.1 column.2 column.3
3        3        c       3c
4        4        d       4d
5        5        e       5e
```

Column NAs

```
new.df[, colSums(is.na(new.df)) <= 1]
```

```
  column.1 column.2
1        1     <NA>
2        2        b
3        3        c
4        4        d
5        5        e
```

Row NAs

```
new.df[rowSums(is.na(new.df)) <= 1,]
```

```
  column.1 column.2 column.3
2        2        b     <NA>
3        3        c       3c
4        4        d       4d
5        5        e       5e
```

Converting date/time

lubridate package

```
year()
month()
day()
week()
as_date()
as_datetime()
time_length()
```

Lubridate examples

```r
library(lubridate)
dates <- as.Date(sales$event_dt[1:3])
dates
```

```
[1] "2015-09-12" "2009-09-04" "2006-04-21"
```

```r
year(dates)
```

```
[1] 2015 2009 2006
```

```r
month(dates)
```

```
[1] 9 9 4
```

```r
week(dates)
```

```
[1] 37 36 16
```

```r
day(dates)
```

More examples

```
floor_date(dates, "month")
```

```
[1] "2015-09-01" "2009-09-01" "2006-04-01"
```

```
ceiling_date(dates, "season")
```

```
[1] "2015-12-01" "2009-12-01" "2006-06-01"
```

```
date1 <- "2009-08-03 12:01:59"
as.Date(date1) # uh-oh
```

```
[1] "2009-08-03"
```

```
as_datetime(date1)
```

```
[1] "2009-08-03 12:01:59 UTC"
```

Strings

grepl returns a logical TRUE/FALSE

```
files <- files[grepl('name', files) == TRUE]
```

```
gsub("hello","goodbye",files)
```

Creating functions

```r
myfunction <- function(x) {
  z <- x + 1
  return(z)
}

myfunction(3)
```

```
[1] 4
```

Function to calculate percentage of NA in columns

```r
calc.na <- function(x, data) {
    calc <- sum(is.na(data[[x]])/nrow(data))
    return(unlist(calc))
}

calc.na("onsale_dt", sales)
```

```
[1] 0.0202
```

What about the amount of every column?

Option 1: For loop

Problem 1: results duplicated Problem 2: returns list indices as column names, which cannot be iterated over or used in the future.

```r
results <- rep(NA, length(colnames(sales)))

for (i in colnames(sales)) {
  results[i] <- calc.na(i, sales)
}

results[60]
```

```
tickets_purchased_qty
                    0
```

Option 2: lapply

```
unlist(lapply(colnames(sales), calc.na, sales))
```

```
 [1] 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.682
[11] 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.000
[21] 0.0008 0.0000 0.0848 0.0000 0.0000 0.0000 0.0000 0.484
[31] 0.0000 0.0000 0.9646 0.9664 0.9692 0.9722 0.9776 0.982
[41] 0.9740 0.9724 0.9842 0.9846 0.9874 0.9354
```

Function - example 2

```r
toMatch <- c("x", "y", "z", "...")
sentences <- c("sentence 1", "sentence 2", "...")

subset_sentences <- function(Match, sentences){
    sentences[grep(Match,sentences)]
}

subsetted <- lapply(toMatch, subset_sentences, sentences)
```

Examples

Example 1

Example of creating list of dates to loop over

```
dates <- seq(ymd_hms('2018-03-08 00:00:00'),
             ymd_hms('2018-03-12 23:00:00'),
             by="1 hour")

dates <- as.character(dates)

for (i in seq_along(dates)) {
  dates[i] <- gsub(" ", "T", dates[i])
}

dates[1:5]
```

```
[1] "2018-03-08T00:00:00" "2018-03-08T01:00:00" "2018-03-0
[4] "2018-03-08T03:00:00" "2018-03-08T04:00:00"
```

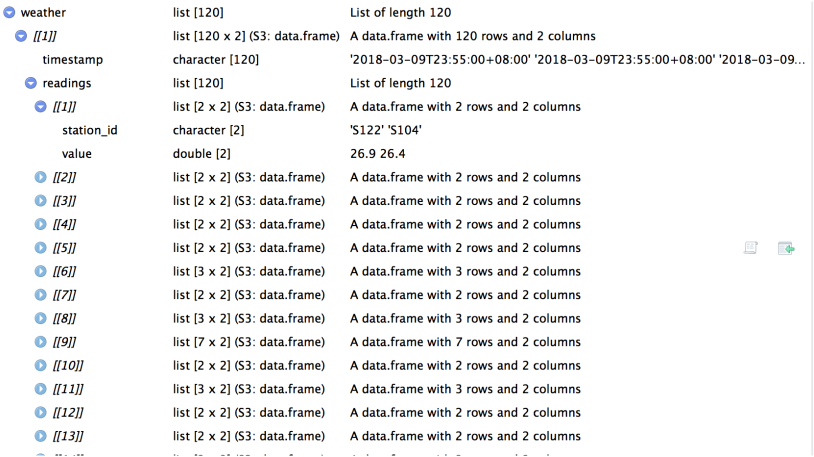
Read in RDS

Data is saved as an RDS because it is a recursively nested list

```
require(dplyr)
require(lubridate)
require(tidyr)
weather <- readRDS("data/scraped_data.rds")
class(weather)
```

```
[1] "matrix"
```

Data



```
weather              list [120]                          List of length 120
  [[1]]              list [120 x 2] (S3: data.frame)     A data.frame with 120 rows and 2 columns
    timestamp        character [120]                     '2018-03-09T23:55:00+08:00' '2018-03-09T23:55:00+08:00' '2018-03-09...
    readings         list [120]                          List of length 120
      [[1]]          list [2 x 2] (S3: data.frame)       A data.frame with 2 rows and 2 columns
        station_id   character [2]                       'S122' 'S104'
        value        double [2]                          26.9 26.4
      [[2]]          list [2 x 2] (S3: data.frame)       A data.frame with 2 rows and 2 columns
      [[3]]          list [2 x 2] (S3: data.frame)       A data.frame with 2 rows and 2 columns
      [[4]]          list [2 x 2] (S3: data.frame)       A data.frame with 2 rows and 2 columns
      [[5]]          list [2 x 2] (S3: data.frame)       A data.frame with 2 rows and 2 columns
      [[6]]          list [3 x 2] (S3: data.frame)       A data.frame with 3 rows and 2 columns
      [[7]]          list [2 x 2] (S3: data.frame)       A data.frame with 2 rows and 2 columns
      [[8]]          list [3 x 2] (S3: data.frame)       A data.frame with 3 rows and 2 columns
      [[9]]          list [7 x 2] (S3: data.frame)       A data.frame with 7 rows and 2 columns
      [[10]]         list [2 x 2] (S3: data.frame)       A data.frame with 2 rows and 2 columns
      [[11]]         list [3 x 2] (S3: data.frame)       A data.frame with 3 rows and 2 columns
      [[12]]         list [2 x 2] (S3: data.frame)       A data.frame with 2 rows and 2 columns
      [[13]]         list [2 x 2] (S3: data.frame)       A data.frame with 2 rows and 2 columns
```

Figure 1:

Cleaning

```r
weather <- unlist(rbind(weather[sapply(weather,
                  length)>0]),
                  recursive=FALSE)
```

```r
weather.times <- weather[seq_along(weather) %% 2 > 0]
weather.times <- unlist(weather.times)

weather.readings <- weather[seq_along(weather) %% 2 == 0]
for (i in c(1:length(weather.readings))) {
  weather.readings[[i]][[1]][3] <- i
}

weather.readings <- do.call("rbind", weather.readings)
weather.readings <- do.call("rbind", weather.readings)
```

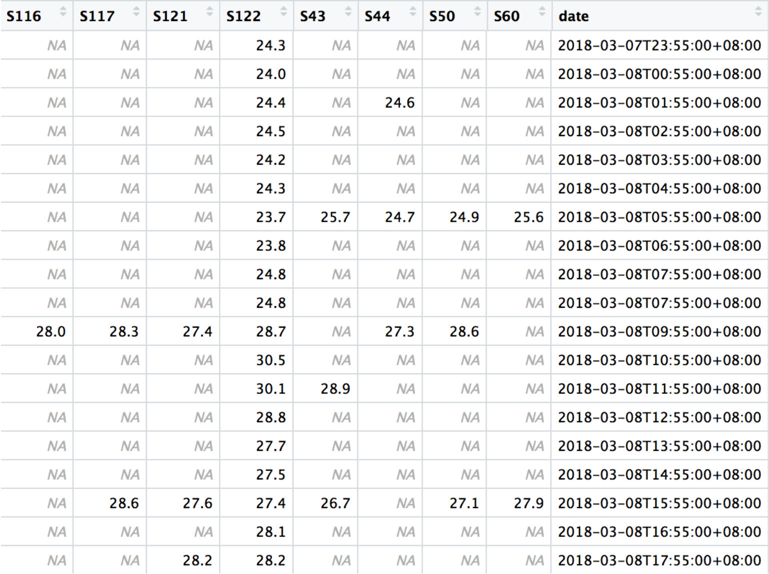
Continued

```
weather.readings <- spread(weather.readings,
                           station_id, value)
```

```
weather.times.df <- as.data.frame(seq
                    (1, length(weather.times)))
weather.times.df$date <- weather.times
colnames(weather.times.df)[1] <- "V3"
```

```
weather.readings <- merge(weather.readings,
                          weather.times.df, by="V3")
colnames(weather.readings)[19] <- "date"
```

Finished



S116	S117	S121	S122	S43	S44	S50	S60	date
NA	NA	NA	24.3	NA	NA	NA	NA	2018-03-07T23:55:00+08:00
NA	NA	NA	24.0	NA	NA	NA	NA	2018-03-08T00:55:00+08:00
NA	NA	NA	24.4	NA	24.6	NA	NA	2018-03-08T01:55:00+08:00
NA	NA	NA	24.5	NA	NA	NA	NA	2018-03-08T02:55:00+08:00
NA	NA	NA	24.2	NA	NA	NA	NA	2018-03-08T03:55:00+08:00
NA	NA	NA	24.3	NA	NA	NA	NA	2018-03-08T04:55:00+08:00
NA	NA	NA	23.7	25.7	24.7	24.9	25.6	2018-03-08T05:55:00+08:00
NA	NA	NA	23.8	NA	NA	NA	NA	2018-03-08T06:55:00+08:00
NA	NA	NA	24.8	NA	NA	NA	NA	2018-03-08T07:55:00+08:00
NA	NA	NA	24.8	NA	NA	NA	NA	2018-03-08T07:55:00+08:00
28.0	28.3	27.4	28.7	NA	27.3	28.6	NA	2018-03-08T09:55:00+08:00
NA	NA	NA	30.5	NA	NA	NA	NA	2018-03-08T10:55:00+08:00
NA	NA	NA	30.1	28.9	NA	NA	NA	2018-03-08T11:55:00+08:00
NA	NA	NA	28.8	NA	NA	NA	NA	2018-03-08T12:55:00+08:00
NA	NA	NA	27.7	NA	NA	NA	NA	2018-03-08T13:55:00+08:00
NA	NA	NA	27.5	NA	NA	NA	NA	2018-03-08T14:55:00+08:00
NA	28.6	27.6	27.4	26.7	NA	27.1	27.9	2018-03-08T15:55:00+08:00
NA	NA	NA	28.1	NA	NA	NA	NA	2018-03-08T16:55:00+08:00
NA	NA	28.2	28.2	NA	NA	NA	NA	2018-03-08T17:55:00+08:00

Figure 2:

Join metadata

```r
require(reshape2)
weather.metadata <- read.csv("data/weather.metadata.csv")
weather.melted <- melt(weather.readings, id.vars="date")
weather.joined <- merge(weather.melted, weather.metadata,
                        by.x="variable", by.y="id")
weather.joined$date <- unlist(weather.joined$date)
weather.joined$day <- as_date(weather.joined$date)
```

Done!

```r
weather.joined[1:5,c(9,3,7,8)]
```

```
         day value location.latitude location.longitude
1 2018-01-04  29.3            1.4172           103.7485
2 2018-01-02    NA            1.4172           103.7485
3 2018-01-02    NA            1.4172           103.7485
4 2018-01-03  25.7            1.4172           103.7485
5 2018-01-04  29.3            1.4172           103.7485
```

Example 2

```
sales <- read.csv("data/sales.csv",
                  na.strings=c("NULL", "NA"))
```

Function

```r
perc.na <- function(data) {
  sing.na <- function(x) {
    sum(is.na(data[[x]])/nrow(data))
  }
  return(unlist(lapply(colnames(data), sing.na)))
}

sales.na <- perc.na(sales)
```

Remove outliers, select columns

```
sales <- sales[,-which(sales.na >= 0.9)]
```

```
sales <- sales %>%
  select(-c(X, primary_act_id, secondary_act_id,
            primary_act_id, secondary_act_id,
            purch_party_lkup_id,
            web_session_cookie_val))
```

Reclassify

```r
sales$secondary <- 0
sales$secondary[!is.na(sales$secondary_act_name)] <- 1
sales$secondary <- as.factor(sales$secondary)
```

Calc new columns

```r
sales$onsale_dt <- as.Date(sales$onsale_dt)
sales$event_dt <- as.Date(sales$event_dt)
sales$sales_ord_tran_dt <-
  as.Date(sales$sales_ord_tran_dt)

sales$sale_length <- sales$event_dt - sales$onsale_dt
sales$purchased_date <- sales$sales_ord_tran_dt -
  sales$onsale_dt

sales$event_hour <- hour(sales$event_date_time)
```

Remove unnecessary columns before aggregating

```
sales <- sales[,-c(1,2,3,4,7,8,9,13,
                   15,16,17,18,19,20,
                   23,24,25,26,27)]
```

Datetime conversions

```
sales$month <- month(sales$event_dt)
sales$year <- year(sales$event_dt)
sales$month_year <- paste(sales$month, sales$year)
```

dplyr for final data

Go to Matt's data exploration workshop to learn more about this!

```r
sales.y <- sales %>%
  select(-c(minor_cat_name, venue_city, event_dt,
            month, month_year, secondary,
            major_cat_name)) %>%
  group_by(year, delivery_type_cd,
           venue_state) %>%
  summarise_all(c("mean", "sum"))
```