Learning *R*

Carl James Schwarz

StatMathComp Consulting by Schwarz
cschwarz.stat.sfu.ca @ gmail.com

Merging, Binding, Table Lookup

Using the *merge* function.

Table of Contents I

1. Merging, Binding, Table Lookup

1.1 Stacking data frames - *rbind*

- Simple stacking
- Columns do not match across data frames
- Unspecified number of data.frames *do.call()*
- Accumulating results - no FOR loops!

1.2 *cbind()*

1.3 *merge()*

- Types of merges
- 1-1 Merging
- 1-Many Merging
- Imputing zeros when only positive results recorded

1.4 Summary

Merging, Binding, Table Lookup

Common Tasks

Some common tasks

- Stacking several data frames atop of each other (row binding) - *rbind()*
  - **AVOID** using *rbind()* to accumulate rows in a *for()* loop
  - In general, never a need for a *for()* loop! (use *plyr* and other packages
- Pasting several data frames side by side (column binding)
  - **AVOID** *cbind()*; use *merge()* to avoid assuming a particular row order
- Matching data frames on key values - *merge*
- Table lookup
  - Simple lookup using *car::recode*
  - General lookup using *merge()*

Stacking data frames

- *rbind(df1, df2, df3)*
  - stacks *df1*, *df2*, . . . into a new single data frame
  - all data frames must have the same columns (but could be in a different order in each data frame)
- *plyr::rbind.fill(df1, df2, df3)*
  - stacks *df1*, *df2*, . . . into a new single data frame
  - data frames could have different columns - missing columns filled with NAs
  - *CAUTION:* use *setdiff(names(df1), names(df2))* to find different column names

Caution about stacking data frames with date-times in different time zones.
Caution about stacking data frames with different sets of factor levels for a variable.

Stacking data frames

Simple stacking

```
df1 <- readxl::read_excel(file.path("Rcourse-code-merge-binc
df2 <- readxl::read_excel(file.path("Rcourse-code-merge-binc
df1
df2
```

```
> df1
   Year Species Count
1  2010 ABCD       25
2  2010 EFGH       34
3  2010 IJKL       34
> df2
   Year Species Count
1  2011 ABCD       22
2  2011 CDED       23
3  2011 EFGH       34
4  2011 IJKL       23
5  2011 MNOP       25
```

Stacking data frames

Simple stacking

```
# simple rbind
# species is stored as a character so not a problem in rbin
df.all <- rbind(df1, df2)
df.all
str(df.all)
```

```
> df.all
   Year Species Count
1  2010 ABCD       25
2  2010 EFGH       34
3  2010 IJKL       34
4  2011 ABCD       22
...
> str(df.all)
 $ Year   : num  2010 2010 2010 2011 2011 ...
 $ Species: chr  "ABCD" "EFGH" "IJKL" "ABCD" ...
 $ Count  : num  25 34 34 22 23 34 23 25
```

Stacking data frames

Simple stacking - conversion of some data

```r
# what happens if some data is character and some integer?
df1$count2 <- df1$Count
df2$count2 <- as.character(df2$Count)

df.all <- rbind(df1, df2)
df.all
str(df.all)
```

```
> df.all
   Year Species Count count2
1  2010 ABCD       25 25
2  2010 EFGH       34 34
...
> str(df.all)
...
 $ count2 : chr  "25" "34" "34" "22" ...
```

Stacking data frames I

Simple stacking - factors combined, but levels not reordered

```r
# what happens with factors?
# factor levels are combined but not reordered
df1$speciesF <- factor(df1$Species)
str(df1)
levels(df1$speciesF)

df2$speciesF <- factor(df2$Species)
str(df2)
levels(df2$speciesF)

df.all <- rbind(df1, df2)
df.all
str(df.all)
levels(df.all$speciesF)
```

Stacking data frames II

```
> levels(df1$speciesF)
[1] "ABCD" "EFGH" "IJKL

> levels(df2$speciesF)
[1] "ABCD" "CDED" "EFGH" "IJKL" "MNOP"

> levels(df.all$speciesF)
[1] "ABCD" "EFGH" "IJKL" "CDED" "MNOP"
```

Note file set of levels no longer ordered alphabetically.

Stacking data frames

Simple stacking - names must match across data frames

```
df1$count3 <- df1$Count
df2$Count3 <- df2$Count

df.all <- rbind(df1, df2)
setdiff(names(df1), names(df2))
setdiff(names(df2), names(df1)) # be sure to look both ways
setdiff( union(names(df1), names(df2)), intersect(names(df1)
```

```
> df.all <- rbind(df1, df2)
Error in match.names(clabs, names(xi)) :
  names do not match previous names
> setdiff(names(df1), names(df2))
[1] "count3"
> setdiff(names(df2), names(df1)) # be sure to look both way
[1] "Count3"
> setdiff( union(names(df1), names(df2)), intersect(names(df
[1] "count3" "Count3"
```

Stacking data frames

Simple stacking - *plyr::rbind.fill()*

```
df.all <- plyr::rbind.fill(df1, df2)
df.all
```

```
> df.all
  Year Species Count count2 speciesF count3 Count3
1 2010    ABCD    25     25     ABCD     25     NA
2 2010    EFGH    34     34     EFGH     34     NA
3 2010    IJKL    34     34     IJKL     34     NA
4 2011    ABCD    22     22     ABCD     NA     22
5 2011    CDED    23     23     CDED     NA     23
6 2011    EFGH    34     34     EFGH     NA     34
7 2011    IJKL    23     23     IJKL     NA     23
8 2011    MNOP    25     25     MNOP     NA     25
```

Note missing values inserted as needed.

Stacking data frames - unspecified number of frame

Simple stacking - unspecified number of data frames

```
sheets.to.read <- readxl::excel_sheets(file.path("Rcourse-co
sheets.to.read

data.list <- llply(sheets.to.read, function(x, workbook){
   df <- readxl::read_excel(workbook, sheet=x)
   df
}, workbook=file.path("Rcourse-code-merge-bind-ds","species-

str(data.list)
```

```
List of 2
 $ :Classes 'tbl_df', 'tbl' and 'data.frame': 3 obs. of  3 v
  ..$ Year   : num [1:3] 2010 2010 2010
  ..$ Species: chr [1:3] "ABCD" "EFGH" "IJKL"
  ..$ Count  : num [1:3] 25 34 34
 $ :Classes 'tbl_df', 'tbl' and 'data.frame': 5 obs. of  3 v
  ..$ Year   : num [1:5] 2011 2011 2011 2011 2011
  ..$ Species: chr [1:5] "ABCD" "CDED" "EFGH" "IJKL" ...
```

Stacking data frames - unspecified number of frame

Regular *rbind()* does NOT work

```
# try this?
df.all <- rbind(data.list)
df.all
```

```
> df.all
          [,1]   [,2]
data.list List,3 List,3
```

Stacking data frames - unspecified number of frame

Use the *do.call()* function.

```
df.all <- do.call(rbind, data.list)
df.all
```

```
> df.all
   Year Species Count
1  2010 ABCD       25
2  2010 EFGH       34
3  2010 IJKL       34
4  2011 ABCD       22
5  2011 CDED       23
6  2011 EFGH       34
7  2011 IJKL       23
8  2011 MNOP       25
```

Stacking data frames - accumulating results

Accumulating results - avoid *rbind()*

See `http://www.win-vector.com/blog/2015/07/efficient-accumulation-in-r`

```
results <- NULL
for(i in 1:10){
    sim.result <- data.frame(sim=i, result=rnorm(1))
    results <- rbind(results, sim.result)
}
results
```

```
> results
   sim     result
1    1 -0.3654261
2    2 -0.7185055
3    3  1.2608358
...
```

Must make a new copy of results each time through the loop.
Thinking like a C++ programmer and not a Rexpert.

Stacking data frames - accumulating results

Accumulating results - avoid *rbind()* - II

```
# better to define receiving structure and insert, but stil
results <- data.frame(sim=1:10, sim.result=NA)
for(sim in 1:10){
    sim.result <-rnorm(1)
    results[sim, "sim.result"] <- sim.result
}
results
```

Better because results data structure defined only once and the modified in place.
Thinking like a Reginer.

Stacking data frames - accumulating results

Accumulating results - avoid *rbind()* - III
Use the *plyr* package paradigm of split-apply-combine.

```
# best, use ldply to do the simulation. This allows for par
results <- plyr::ldply(1:10, function(sim){
     sim.result <- data.frame(sim=sim, result=rnorm(1))
})
results
```

Allows for easy parallelization (see else where in notes).
NEVER USE FOR LOOPS (unless you call me first).
Thinking like a Rexpert.

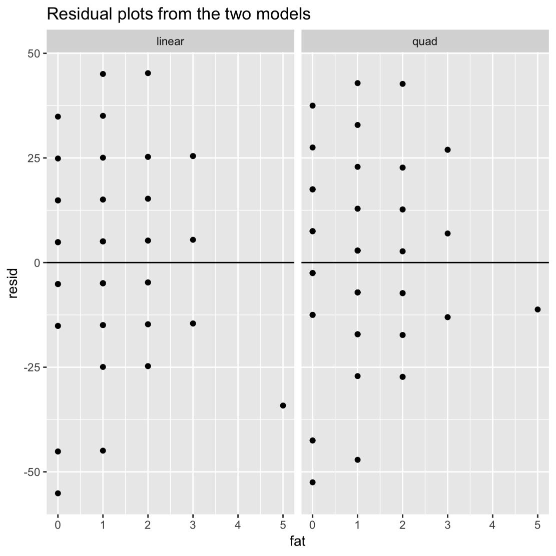
Stacking data frames - Exercise

Return to the cereal data frame.

- Fit a straight line between calories and fat.
- Fit a quadratic line between calories and fat.
- Extract the two fits and residuals; stack them; and create side-by-side fit and residual plots as shown below

Stacking data frames - Exercise

Exercise final plot to be produced



Pasting data frames - *cbind()*

Simple pasting

```
all.df <- cbind(df1. df2)
```

AVOID because it assumes that *df1* and *df2* are sorted in same order.
Do you really want the *data.frame()* function?
Otherwise, you likely want to use *merge()*

Merging data frames- *merge()*

Types of merging

- 1-1 merging (with possible missing matches)
- 1-many merging (table lookup; data at different levels)
- many-many merging - uncommon - are you sure????

Merging data frames- *merge()*

1-1 Merging

- One record from each data frame
- Match on a set (≥ 1) key columns
- CAUTION: Key columns must match on case
- CAUTION: What do with non-matches? *all.x=*, *all.y=*, and *all=* arguments.
- CAUTION: Multiple merges

Merging data frames- *merge()* I

1-1 Merging

```
# 1-1 merging
i2000 <- readxl::read_excel(file.path("Rcourse-code-merge-b
i2001 <- readxl::read_excel(file.path("Rcourse-code-merge-b
i2002 <- readxl::read_excel(file.path("Rcourse-code-merge-b

# notice data in different order. Do not use cbind() here.
i2000
i2001
i2002
```

Merging data frames- *merge()* II

```
> # notice data in different order. Do not use cbind() here
> i2000
  Surname I2000
1 A          50
2 B          60
3 C          70

> i2001
  Surname I2001
1 B          61
2 C          70
3 A          51
```

Merging data frames- *merge()* III

```
> i2002
  Surname i2002
1 C          72
2 A          52
3 D          92
```

Notice different order. Not all families present in all years.

Merging data frames- *merge()*

1-1 Merging

```
# merge the data together.
income <- merge(i2000, i2001)
income
```

```
> income
  Surname I2000 I2001
1       A    50    51
2       B    60    61
3       C    70    70
```

Matching column must match on case.
Careful of beginning/trailing/embedded blanks in character strings.
You can specify variables to match on using the *by* arguments.

Merging data frames- *merge()* I

1-1 Merging - missing values

```r
# what happens with missing data
merge(i2000, i2002)
merge(i2000, i2002, all=TRUE)
merge(i2000, i2002, all.x=TRUE)
merge(i2000, i2002, all.y=TRUE)
```

```
> merge(i2000, i2002)
  Surname I2000 i2002
1       A    50    52
2       C    70    72
```

Merging data frames- *merge()* II

```
> merge(i2000, i2002, all=TRUE)
  Surname I2000 i2002
1       A    50    52
2       B    60    NA
3       C    70    72
4       D    NA    92

> merge(i2000, i2002, all.x=TRUE)
  Surname I2000 i2002
1       A    50    52
2       B    60    NA
3       C    70    72
```

Merging data frames- *merge()* III

```
> merge(i2000, i2002, all.y=TRUE)
  Surname I2000 i2002
1       A    50    52
2       C    70    72
3       D    NA    92
```

Merging data frames- *merge()*

1-1 Merging - multiple merging.
Regular *merge()* only allows two data frames at a time.

```
Reduce(function(...){merge(..., all=TRUE)},
      list(i2000, i2001, i2002))
```

```
  Surname I2000 I2001 i2002
1       A    50    51    52
2       B    60    61    NA
3       C    70    70    72
4       D    NA    NA    92
```

Merging data frames- *merge()*

1-Many Merging.
Data collected at different levels.

```
child<- readxl::read_excel(file.path("Rcourse-code-merge-bi
child
```

```
  Surname Childname   YoB ElemSchool
1 A       ca1        1986 E1
2 A       ca2        1988 E2
3 B       cb1        1972 E1
4 B       cb2        1975 E1
5 D       cd1        1991 E2
6 D       cd2        1993 E2
7 D       cd3        1995 E2
```

Merging data frames- *merge()* I

1-Many Merging.
Dealing with missing values?

```
merge(i2000, child)
merge(i2000, child, all.x=TRUE)
merge(i2000, child, all=TRUE)
```

```
> merge(i2000, child)
  Surname I2000 Childname  YoB
1       A    50       ca1 1986
2       A    50       ca2 1988
3       B    60       cb1 1972
4       B    60       cb2 1975
```

Merging data frames- *merge()* II

```
> merge(i2000, child, all.x=TRUE)
  Surname I2000 Childname  YoB
1       A    50       ca1 1986
2       A    50       ca2 1988
3       B    60       cb1 1972
4       B    60       cb2 1975
5       C    70      <NA>   NA
```

Merging data frames- *merge()* III

```
> merge(i2000, child, all=TRUE)
  Surname I2000 Childname  YoB
1       A    50       ca1 1986
2       A    50       ca2 1988
3       B    60       cb1 1972
4       B    60       cb2 1975
5       C    70      <NA>   NA
6       D    NA       cd1 1991
7       D    NA       cd2 1993
8       D    NA       cd3 1995
```

Merging data frames- *merge()*

1-Many Merging - table lookup.
For small lookups, use *car::recode()* function.

```
eschool <- readxl::read_excel(file.path("Rcourse-code-merge-
eschool
```

```
> eschool
  ElemSchool Built Capacity ClassRooms
1 E1          1972      200         15
2 E2          1973      150         12
3 E3          1980      200         16
4 E4          1982      175         13
```

Merging data frames- *merge()*

1-Many Merging - table lookup.
Use appropriate *all.x* or *all.y* to only match table of interests

```
child <- merge(child, eschool, all.x=TRUE)  # do NOT use al
child
```

```
> child
  ElemSchool Surname Childname  YoB Built Capacity ClassRoo
1         E1       A       ca1 1986  1972      200
2         E1       B       cb1 1972  1972      200
3         E1       B       cb2 1975  1972      200
4         E2       A       ca2 1988  1973      150
5         E2       D       cd1 1991  1973      150
6         E2       D       cd2 1993  1973      150
7         E2       D       cd3 1995  1973      150
```

Merging data frames- *merge()*

Using merges to insert implied zeroes

- Many databases only record POSITIVE species counts
- You need to impute a 0 for a survey with NO species present.
- Need three data frames
  - Detections (positive counts only)
  - Field visit information (which points visited in which years)
  - Species list of interest
- A many-many merge gives the species x points records
- This is merged with detections
- NA's are replaced by 0's.

Merging data frames- *merge()*

Using merges to insert implied zeroes. Refer to the *BirdDetects.xlxs* workbook. We want to compute the average count for each species for each year over the points.

```
Species <- readxl::read_excel(file.path("Rcourse-code-merge-
Species
```

```
> Species
  Species
1 S1
2 S2
3 S3
4 S4
```

This is the list of all species of interest.

Merging data frames- *merge()* I

Using merges to insert implied zeroes. Refer to the *BirdDetects.xlxs* workbook.

```
# Notice that not all points visited in all years
VisitInfo <- readxl::read_excel(file.path("Rcourse-code-merg
VisitInfo
```

```
> VisitInfo
    Year Transect Point Temperature
1   2000        1     1          23
2   2000        1     2          24
3   2000        1     3          23
4   2000        1     4          22
5   2000        2     1          25
6   2000        2     2          24
7   2000        2     3          23
8   2000        2     4          22
```

Merging data frames- *merge()* II

```
9   2000      3    3         47
10  2000      3    4         28
11  2000      4    1         23
12  2000      4    2         25
13  2001      1    1         23
14  2001      1    2         24
15  2001      1    3         23
16  2001      1    4         22
17  2001      3    1         19
18  2001      3    2         18
19  2001      3    3         47
20  2001      3    4         28
21  2001      4    1         23
22  2001      4    2         25
```

Notice that not all points visited in all years

Merging data frames- *merge()* I

Using merges to insert implied zeroes. Refer to the *BirdDetects.xlxs* workbook.

```
# Notice that only positive detections listed here
Detects <- readxl::read_excel(file.path("Rcourse-code-merge-
Detects
```

```
> Detects
   Year Transect Point Species Count
 1 2000        1     1 S1         10
 2 2000        1     1 S3          5
 3 2000        1     2 S1          5
 4 2000        1     2 S2          6
 5 2000        1     2 S3          7
 6 2000        1     2 S4          8
 7 2000        1     3 S2          5
 8 2000        1     4 S1          3
```

Merging data frames- *merge()* II

```
 9   2000          1       4  S2            3
10   2000          1       4  S3            3
...
```

Only positive counts recorded at each year-transect-point.

Merging data frames- *merge()* I

Using merges to insert implied zeroes.
Get master list of species x Visits using a MANY-MANY merge

```
# Get master set of species x VisitInfo, i.e .all visits x
dim(VisitInfo)
dim(Species)

VisitInfoSpecies <- merge(VisitInfo, Species)
dim(VisitInfoSpecies)
head(VisitInfoSpecies)
```

```
> dim(VisitInfo)
[1] 22  4

> dim(Species)
[1] 4 1
```

Merging data frames- *merge()* II

```
> VisitInfoSpecies <- merge(VisitInfo, Species)
> dim(VisitInfoSpecies)
[1] 88  5

> head(VisitInfoSpecies)
  Year Transect Point Temperature Species
1 2000        1     1          23      S1
2 2000        1     2          24      S1
3 2000        1     3          23      S1
4 2000        1     4          22      S1
5 2000        2     1          25      S1
6 2000        2     2          24      S1
```

Merging data frames- *merge()* I

Using merges to insert implied zeroes.
Merge with positive counts and impute missing zeroes.

```
# Now merge with positive counts and impute zeroes
AllCounts <- merge(Detects, VisitInfoSpecies, all.y=TRUE)
dim(Detects)
dim(VisitInfoSpecies)
dim(AllCounts)
head(AllCounts)

# Add the imputed 0’s
AllCounts$Count[ is.na(AllCounts$Count)] <- 0
```

Merging data frames- *merge()* II

```
> AllCounts <- merge(Detects, VisitInfoSpecies, all.y=TRUE)
> dim(Detects)
[1] 43  5
> dim(VisitInfoSpecies)
[1] 88  5
> dim(AllCounts)
[1] 88  6
> head(AllCounts)
  Year Transect Point Species Count Temperature
1 2000        1     1      S1    10          23
2 2000        1     1      S2    NA          23
3 2000        1     1      S3     5          23
4 2000        1     1      S4    NA          23
5 2000        1     2      S1     5          24
6 2000        1     2      S2     6          24
....
```

Merging data frames- *merge()* III

```
> AllCounts$Count[ is.na(AllCounts$Count)] <- 0
> head(AllCounts)
  Year Transect Point Species Count Temperature
1 2000        1     1      S1    10          23
2 2000        1     1      S2     0          23
3 2000        1     1      S3     5          23
4 2000        1     1      S4     0          23
5 2000        1     2      S1     5          24
6 2000        1     2      S2     6          24
....
```

Notice use of *all.y=TRUE* to force all visit x species records to be included.
Now you can compute the proper averages as needed.

Merging data frames- *merge()* - Exercise I I

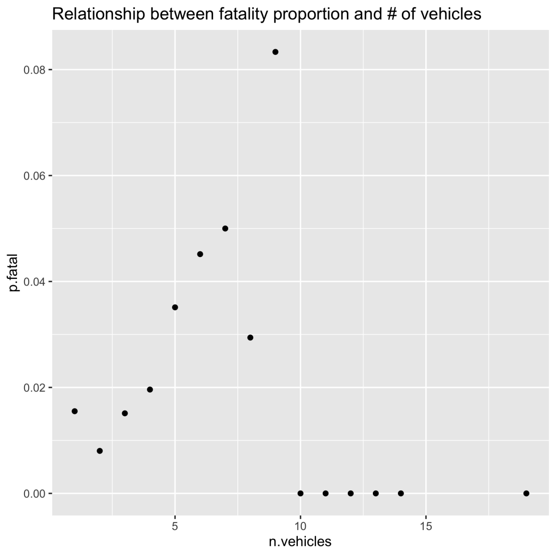
How does the p(fatality) vary with number of vehicles in the accident? Ignore the information on number of vehicles on the accident file.

- Read accident and vehicle information
  - Convert dates to proper format
  - Recode *Accident_Severity* to 1=fatal (code=1) vs 0=non-fatal (codes 2 and 3).
- Summarize vehicle information to get number of vehicles
  - Use *plyr::ddply()* and *plyr::summarize*
  - Are there accidents that are missing information ?
- Merge with accident data. Notice that the key column has a different name in the two files.
- Summarize by number of vehicles. Hint: Mean(fatal as 0/1 variable) = proportion.
- Plot.

.

Stacking data frames - Exercise

Exercise - final plot to be produced.



Merging data frames- *merge()* I

How does the p(fatality) vary with number of vehicles in the accident? Ignore the information on number of vehicles

```
accidents <- read.csv(file.path("..","sampledata","Accidents
                 as.is=TRUE, strip.white=TRUE)
# Convert date to internal date format
accidents$mydate <- as.Date(accidents$Date, format="%d/%m/%
# Create the fatality variable
accidents$Fatality <- accidents$Accident_Severity == 1
```

Merging data frames- *merge()* II

```
vehicles <- read.csv(file.path("..","sampledata","Accidents"
             as.is=TRUE, strip.white=TRUE)
head(vehicles)

n.vehicles <- plyr::ddply(vehicles, "Acc_Index", plyr::summa
                          n.vehicles=length(Acc_Index))

# are there any accidents with missing data?
setdiff(accidents$Accident_Index, n.vehicles$Acc_Index)
setdiff(n.vehicles$Acc_Index, accidents$Accident_Index)
```

Merging data frames- *merge()* III

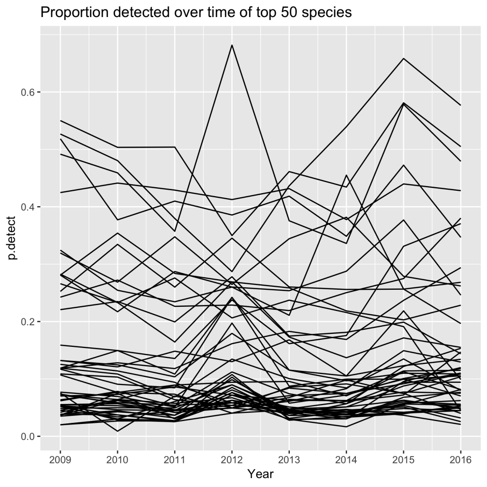
```
accidents2 <- merge(accidents, n.vehicles, by.x="Accident_I
dim(accidents2)

p.fatal <- plyr::ddply(accidents2, "n.vehicles", plyr::summa
                        p.fatal=mean(Fatality))
head(p.fatal)

# a plot
fatal.plot <- ggplot(data=p.fatal, aes(x=n.vehicles, y=p.fat
  ggtitle("Relationship between fatality proportion and # o
  geom_point()
fatal.plot
```

Merging data frames- *merge()* - Exercise- II I

Refer to the *BirdDetects* folder. Plot the proportion of points where the top 50 species of birds (by detections) are detected over time.



Merging data frames- *merge()* - Exercise I

- Read in data files.
- Compute total detections by species and keep the top 50.
- Select the top 50 species from the detection records.
- Check that all detection records correspond to a field visit. Hint: create a key. Look at the reverse. Are you surprised?
- Create field visits x top 50 species.
- Impute 0 detections.
- Find p(detect) by species-year combination.
- Plot.

Merging data frames- *merge()* - Exercise I

```
transect <- read.csv(file.path("..","sampledata","BirdDetect
field    <- read.csv(file.path("..","sampledata","BirdDetect
detect   <- read.csv(file.path("..","sampledata","BirdDetect
species  <- read.csv(file.path("..","sampledata","BirdDetect

head(species)
head(transect)
head(field)
head(detect)
```

Merging data frames- *merge()* - Exercise II

```
# find the total detections by species and get the top 50 s
total.detects <- plyr::ddply(detect, 'AOU_Code', plyr::summa
                             n.detect=length(AOU_Code))
total.detects
sum(total.detects$n.detect > 200)
total.detects <- total.detects[ order(total.detects$n.detect
species.of.interest <- total.detects[1:50,]
species.of.interest
```

```
> species.of.interest
    AOU_Code n.detect
41      CHSP     2406
159     YRWA     2360
105     PISI     2141
8       AMRO     2002
131     SWTH     1949
...
```

Merging data frames- *merge()* - Exercise III

```r
# only select detection records of species of interest
dim(detect)
detect <- detect[ detect$AOU_Code
          %in% species.of.interest$AOU_Code,]
dim(detect)
```

```
> dim(detect)
[1] 39613     6
> detect <- detect[ detect$AOU_Code
       %in% species.of.interest$AOU_Code,]
> dim(detect)
[1] 34544     6
```

Merging data frames- *merge()* - Exercise IV

```
head(field)
field$Year <- lubridate::year(field$Date)
head(detect)
detect$Year <- lubridate::year(detect$Date)

# create a key with Year, transect,point
field$Key <- paste(field$Year, field$ParkTransectID,
          field$PointID, sep=".")
detect$Key<- paste(detect$Year, detect$ParkTransectID,
          detect$PointID, sep=".")
setdiff(detect$Key, field$Key) # this should be empty
setdiff(field$Key, detect$Key) # this may be non-empty
```

Merging data frames- *merge()* - Exercise V

```
> setdiff(detect$Key, field$Key) # this should be empty
character(0)
> setdiff(field$Key, detect$Key) # this may be non-empty
 [1] "2011.2.10"   "2013.2.10"   "2015.2.10"   "2011.2.5"
 [8] "2014.5.1"    "2014.5.2"    "2016.12.3"   "2016.12.4"
[15] "2016.25.14"  "2009.40.7"   "2015.47.40"  "2011.50.5"
[22] "2011.63.7"   "2016.72.2"   "2016.72.6"   "2010.76.10"
[29] "2016.91.10"  "2013.91.6"   "2014.91.7"   "2015.91.7"
[36] "2011.91.9"   "2013.91.9"   "2014.91.9"   "2011.135.10"
[43] "2010.139.3"  "2010.139.4"  "2010.139.5"  "2010.139.6"
[50] "2013.147.19" "2013.147.3"  "2013.147.4"  "2013.147.5"
[57] "2011.149.4"  "2011.149.5"  "2009.149.6"  "2010.149.6"
```

Merging data frames- *merge()* - Exercise VI

```r
# create species x field visit
dim(field)
field <- merge(field, species.of.interest)
dim(field)
```

```
> dim(field)
[1] 236500     10
> dim(detect)
[1] 34544     7
> detect <- merge(detect, field, all.y=TRUE)
> dim(detect)
[1] 236500     11
```

Merging data frames- *merge()* - Exercise VII

```r
# impute zeroes
names(field)
names(detect)
dim(field)
dim(detect)
detect <- merge(detect, field, all.y=TRUE)
dim(detect)
detect$detect[ is.na(detect$detect)] <- 0
```

```
> dim(field)
[1] 4730    8
> field <- merge(field, species.of.interest)

> dim(field)
[1] 236500     10
```

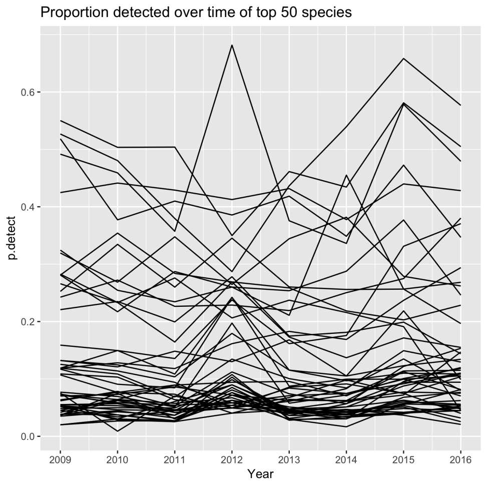
The final dimension should match the field visit x species data frame.

Merging data frames- *merge()* - Exercise VIII

```
p.detect <- plyr::ddply(detect, c("AOU_Code","Year"), plyr:
                        p.detect=mean(detect))
xtabs(~AOU_Code+Year, data=p.detect)

detect.plot <- ggplot(data=p.detect, aes(x=Year, y=p.detect
  ggtitle("Proportion detected over time of top 50 species")
  geom_line()+
  scale_x_continuous(breaks=2000:3000)
detect.plot
```

Merging data frames- *merge()* - Exercise I



Summary I

Stacking data frames

- *rbind()* vs. *plyr::rbind.fill()*
- Caution about combining factor variables with different sets of levels.
- Caution about combining datetime with different time zones.
- *do.call()* to stack indeterminate number of data frames
- Think like an Rexpert when accumulating results - NO FOR LOOPS!

Summary II

Pasting data frames

- Avoid *cbind()*.
- Careful with *merge()* - use *setdiff()* to check keys.
- Use for table lookup with *all.x=* and *all.y=* arguments.
- Use for imputing 0's when only positive counts are recorded.