Joining Tables

Contents

Data is generally stored in multiple smaller tables. These smaller tables share a common variable or field, thus are related (which is why these tables are also known as relations). All the relations together comprise a relational database.

Storing data across relations offers a number of benefits such as reduced redundancy, fewer errors from updates, and greater efficiency in storage and retrieval. The data required for analysis at hand is retrieved by linking relations through joins. The languages most widely used for this is SQL, however, this note will illustrate the joins using R.

Baseball Data

To illustrate joins, we will use baseball data compiled in library(Lahman) based on the Lahman Database

```
library(Lahman)
```

Baseball data is spread across a number of smaller tables.

```
library(printr)
data(package = "Lahman")
```

Table 1: Data sets in Lahman

Item	Title
AllstarFull	AllstarFull table
Appearances	Appearances table
AwardsManagers	AwardsManagers table
AwardsPlayers	AwardsPlayers table
AwardsShareManagers	AwardsShareManagers table
AwardsSharePlayers	AwardsSharePlayers table
Batting	Batting table
BattingPost	BattingPost table
CollegePlaying	CollegePlaying table
Fielding	Fielding table
FieldingOF	FieldingOF table
FieldingOFsplit	FieldingOFsplit table
FieldingPost	FieldingPost data
HallOfFame	Hall of Fame Voting Data
HomeGames	HomeGames table

Item	Title
LahmanData	Lahman Datasets
Managers	Managers table
ManagersHalf	ManagersHalf table
Master	Master table
Parks	Parks table
People	People table
Pitching	Pitching table
PitchingPost	PitchingPost table
Salaries	Salaries table
Schools	Schools table
SeriesPost	SeriesPost table
Teams	Teams table
TeamsFranchises	TeamFranchises table
TeamsHalf	TeamsHalf table
battingLabels	Variable Labels
fieldingLabels	Variable Labels
pitchingLabels	Variable Labels

Teams data

In a well-designed relational database, each table has a variable, called a primary key, that uniquely identifies each row. Let us see if this is the case for the `Teams` table.

```
detach('package:printr', unload = TRUE)
head(Teams)
```

```
##   yearID lgID teamID franchID divID Rank  G Ghome  W  L DivWin WCWin LgWin
## 1   1871   NA    BS1      BNA  <NA>    3 31    NA 20 10   <NA>  <NA>     N
## 2   1871   NA    CH1      CNA  <NA>    2 28    NA 19  9   <NA>  <NA>     N
## 3   1871   NA    CL1      CFC  <NA>    8 29    NA 10 19   <NA>  <NA>     N
## 4   1871   NA    FW1      KEK  <NA>    7 19    NA  7 12   <NA>  <NA>     N
## 5   1871   NA    NY2      NNA  <NA>    5 33    NA 16 17   <NA>  <NA>     N
## 6   1871   NA    PH1      PNA  <NA>    1 28    NA 21  7   <NA>  <NA>     Y
##   WSWin   R   AB   H X2B X3B HR BB SO SB CS HBP SF  RA  ER  ERA CG SHO SV
## 1  <NA> 401 1372 426  70  37  3 60 19 73 16  NA NA 303 109 3.55 22   1  3
## 2  <NA> 302 1196 323  52  21 10 60 22 69 21  NA NA 241  77 2.76 25   0  1
## 3  <NA> 249 1186 328  35  40  7 26 25 18  8  NA NA 341 116 4.11 23   0  0
## 4  <NA> 137  746 178  19   8  2 33  9 16  4  NA NA 243  97 5.17 19   1  0
## 5  <NA> 302 1404 403  43  21  1 33 15 46 15  NA NA 313 121 3.72 32   1  0
## 6  <NA> 376 1281 410  66  27  9 46 23 56 12  NA NA 266 137 4.95 27   0  0
##   IPouts  HA HRA BBA SOA   E DP    FP                    name
## 1    828 367   2  42  23 243 24 0.834    Boston Red Stockings
## 2    753 308   6  28  22 229 16 0.829 Chicago White Stockings
## 3    762 346  13  53  34 234 15 0.818  Cleveland Forest Citys
## 4    507 261   5  21  17 163  8 0.803    Fort Wayne Kekiongas
## 5    879 373   7  42  22 235 14 0.840        New York Mutuals
## 6    747 329   3  53  16 194 13 0.845  Philadelphia Athletics
##                           park attendance BPF PPF teamIDBR teamIDlahman45
## 1          South End Grounds I         NA 103  98      BOS            BS1
## 2      Union Base-Ball Grounds         NA 104 102      CHI            CH1
## 3 National Association Grounds         NA  96 100      CLE            CL1
## 4                Hamilton Field        NA 101 107      KEK            FW1
```

```
## 5     Union Grounds (Brooklyn)         NA  90  88      NYU             NY2
## 6     Jefferson Street Grounds         NA 102  98      ATH             PH1
##   teamIDretro
## 1         BS1
## 2         CH1
## 3         CL1
## 4         FW1
## 5         NY2
## 6         PH1
```

Let us check to see if teamID uniquely identifies each row.

```
library(dplyr)
Teams %>%
  group_by(teamID, name)%>%
  summarize(n = n())%>%
  filter(n>1)%>%
  arrange(desc(n))
```

```
## # A tibble: 116 x 3
## # Groups:   teamID [90]
##    teamID name                      n
##    <fct>  <chr>                 <int>
##  1 PIT    Pittsburgh Pirates      130
##  2 PHI    Philadelphia Phillies   129
##  3 CIN    Cincinnati Reds         125
##  4 SLN    St. Louis Cardinals     121
##  5 CHA    Chicago White Sox       120
##  6 DET    Detroit Tigers          120
##  7 CHN    Chicago Cubs            118
##  8 BOS    Boston Red Sox          113
##  9 NYA    New York Yankees        108
## 10 CLE    Cleveland Indians       106
## # ... with 106 more rows
```

```
head(Teams)
```

```
##   yearID lgID teamID franchID divID Rank  G Ghome  W  L DivWin WCWin LgWin
## 1   1871   NA    BS1      BNA  <NA>    3 31    NA 20 10   <NA>  <NA>     N
## 2   1871   NA    CH1      CNA  <NA>    2 28    NA 19  9   <NA>  <NA>     N
## 3   1871   NA    CL1      CFC  <NA>    8 29    NA 10 19   <NA>  <NA>     N
## 4   1871   NA    FW1      KEK  <NA>    7 19    NA  7 12   <NA>  <NA>     N
## 5   1871   NA    NY2      NNA  <NA>    5 33    NA 16 17   <NA>  <NA>     N
## 6   1871   NA    PH1      PNA  <NA>    1 28    NA 21  7   <NA>  <NA>     Y
##   WSWin   R   AB   H X2B X3B HR BB SO SB CS HBP SF  RA  ER  ERA CG SHO SV
## 1  <NA> 401 1372 426  70  37  3 60 19 73 16  NA NA 303 109 3.55 22   1  3
## 2  <NA> 302 1196 323  52  21 10 60 22 69 21  NA NA 241  77 2.76 25   0  1
## 3  <NA> 249 1186 328  35  40  7 26 25 18  8  NA NA 341 116 4.11 23   0  0
## 4  <NA> 137  746 178  19   8  2 33  9 16  4  NA NA 243  97 5.17 19   1  0
## 5  <NA> 302 1404 403  43  21  1 33 15 46 15  NA NA 313 121 3.72 32   1  0
## 6  <NA> 376 1281 410  66  27  9 46 23 56 12  NA NA 266 137 4.95 27   0  0
##   IPouts  HA HRA BBA SOA   E DP    FP                    name
## 1    828 367   2  42  23 243 24 0.834    Boston Red Stockings
## 2    753 308   6  28  22 229 16 0.829 Chicago White Stockings
## 3    762 346  13  53  34 234 15 0.818  Cleveland Forest Citys
## 4    507 261   5  21  17 163  8 0.803    Fort Wayne Kekiongas
```

```
## 5    879 373   7  42   22 235 14 0.840        New York Mutuals
## 6    747 329   3  53   16 194 13 0.845  Philadelphia Athletics
##                             park attendance BPF PPF teamIDBR teamIDlahman45
## 1          South End Grounds I         NA 103  98      BOS            BS1
## 2      Union Base-Ball Grounds         NA 104 102      CHI            CH1
## 3 National Association Grounds         NA  96 100      CLE            CL1
## 4               Hamilton Field         NA 101 107      KEK            FW1
## 5     Union Grounds (Brooklyn)         NA  90  88      NYU            NY2
## 6     Jefferson Street Grounds         NA 102  98      ATH            PH1
##   teamIDretro
## 1         BS1
## 2         CH1
## 3         CL1
## 4         FW1
## 5         NY2
## 6         PH1
```

It is clear that teamID is repeated multiple times, so it does not uniquely identify each row. From reviewing the first few rows of Teams, you have probably realized that the table lists teams for each Year. So, let us see if each row is uniquely defined by teamID and yearID.

```
Teams %>%
  group_by(teamID, yearID)%>%
  summarize(n = n())%>%
  filter(n>1)
```

```
## # A tibble: 0 x 3
## # Groups:   teamID [0]
## # ... with 3 variables: teamID <fct>, yearID <int>, n <int>
```

Salaries data

From the above, it is clear that teamID and yearID uniquely identify each row in Teams. While this table is a rich source of information on team performance, it doesn't contain any information on player Salary. Salary data is contained in a table called Salaries. However, salary data is at a more granular level, providing compensation for each player.

```
head(Salaries)
```

```
##   yearID teamID lgID  playerID salary
## 1   1985    ATL   NL barkele01 870000
## 2   1985    ATL   NL bedrost01 550000
## 3   1985    ATL   NL benedbr01 545000
## 4   1985    ATL   NL  campri01 633333
## 5   1985    ATL   NL ceronri01 625000
## 6   1985    ATL   NL chambch01 800000
```

We will roll up the data to the team level by computing average player salary.

```
team_salary =
  Salaries %>%
  group_by(yearID, teamID)%>%
  summarize(avgSalary = mean(salary,na.rm=T))
team_salary
```

```
## # A tibble: 918 x 3
## # Groups:   yearID [32]
```

```
##    yearID teamID avgSalary
##     <int> <fct>      <dbl>
##  1   1985 ATL      673045.
##  2   1985 BAL      525487.
##  3   1985 BOS      435902.
##  4   1985 CAL      515282.
##  5   1985 CHA      468866.
##  6   1985 CHN      577405.
##  7   1985 CIN      379996.
##  8   1985 CLE      327583.
##  9   1985 DET      517407.
## 10   1985 HOU      499653.
## # ... with 908 more rows
```

Join Teams and Salaries

You will note that the `team_salary` data by `teamID` and `yearID` just like the `Teams` table. So, we can join the `Teams` data with `Salaries` using `teamID` and `yearID`. There are four kinds of joins: inner join, left outer, right outer and full outer join. An inner join will keep rows for which the keys match across the two tables. A left outer join will keep all rows in the table in the left and matching rows from the right. A right outer join will keep all rows in the table in the right and matching rows from the left. Finally, a full outer join will keep rows that match across the two tables and the ones that don't match.

We want to only keep data for which there is a match across the `Teams` and `Salaries` table, so we implement an inner join using the `dplyr` function `inner_join`. An alternative is to use `merge()` from Base R.

```
Teams %>%
  inner_join(team_salary, by = c('yearID' = 'yearID', 'teamID' = 'teamID'))%>%
  head()
```

```
##   yearID lgID teamID franchID divID Rank   G Ghome  W  L DivWin WCWin LgWin
## 1   1985   NL    ATL      ATL     W    5 162    81 66 96      N  <NA>     N
## 2   1985   AL    BAL      BAL     E    4 161    81 83 78      N  <NA>     N
## 3   1985   AL    BOS      BOS     E    5 163    81 81 81      N  <NA>     N
## 4   1985   AL    CAL      ANA     W    2 162    79 90 72      N  <NA>     N
## 5   1985   AL    CHA      CHW     W    3 163    81 85 77      N  <NA>     N
## 6   1985   NL    CHN      CHC     E    4 162    81 77 84      N  <NA>     N
##   WSWin   R   AB    H X2B X3B  HR  BB  SO  SB CS HBP SF  RA  ER  ERA CG SHO SV
## 1     N 632 5526 1359 213  28 126 553 849  72 52  22 41 781 679 4.19  9   9 29
## 2     N 818 5517 1451 234  22 214 604 908  69 43  19 40 764 694 4.38 32   6 33
## 3     N 800 5720 1615 292  31 162 562 816  66 27  30 57 720 659 4.06 35   8 29
## 4     N 732 5442 1364 215  31 153 648 902 106 51  39 35 703 633 3.91 22   8 41
## 5     N 736 5470 1386 247  37 146 471 843 108 56  43 45 720 656 4.07 20   8 39
## 6     N 686 5492 1397 239  28 150 562 937 182 49  18 39 729 666 4.16 20   8 42
##   IPouts   HA HRA BBA  SOA   E  DP    FP              name
## 1   4372 1512 134 642  776 159 197 0.976    Atlanta Braves
## 2   4282 1480 160 568  793 129 168 0.979 Baltimore Orioles
## 3   4384 1487 130 540  913 145 161 0.977    Boston Red Sox
## 4   4372 1453 171 514  767 112 202 0.982 California Angels
## 5   4355 1411 161 569 1023 111 152 0.982 Chicago White Sox
## 6   4327 1492 156 519  820 134 150 0.979      Chicago Cubs
##                           park attendance BPF PPF teamIDBR teamIDlahman45
## 1 Atlanta-Fulton County Stadium    1350137 105 106      ATL            ATL
## 2             Memorial Stadium    2132387  97  97      BAL            BAL
## 3                Fenway Park II    1786633 104 104      BOS            BOS
```

```
## 4              Anaheim Stadium    2567427 100 100      CAL             CAL
## 5                Comiskey Park    1669888 104 104      CHW             CHA
## 6                Wrigley Field    2161534 110 110      CHC             CHN
##   teamIDretro avgSalary
## 1         ATL  673045.5
## 2         BAL  525486.9
## 3         BOS  435902.4
## 4         CAL  515281.9
## 5         CHA  468865.6
## 6         CHN  577405.3
```

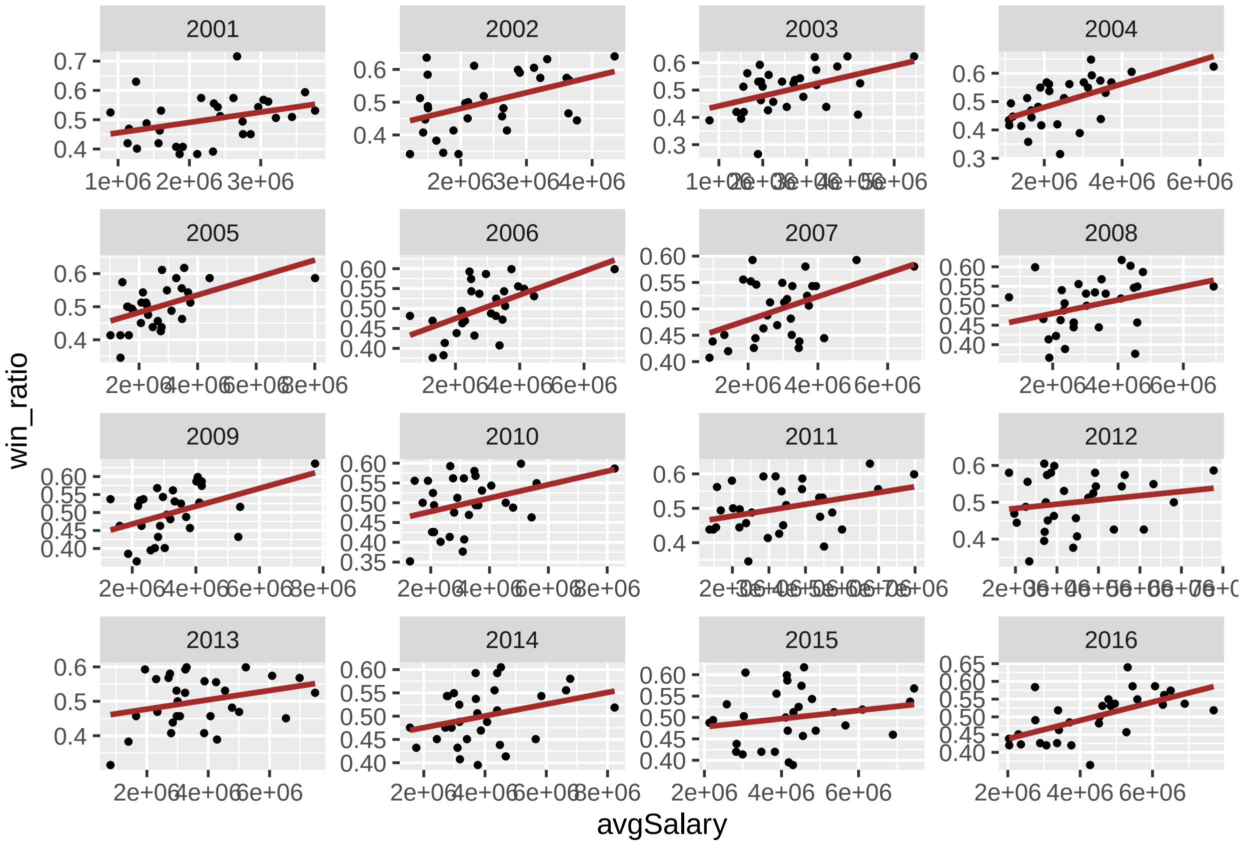
The joined table contains data on both team performance and average team salary. Now, one can filter the data to see if average salary paid is related to good performance or if it draws more fans. We measure performance as proportion of wins.

```
Teams %>%
  inner_join(team_salary, by = c('yearID' = 'yearID', 'teamID' = 'teamID'))%>%
  mutate(win_ratio = W/(W+L))%>%
  select(yearID, teamID, name, win_ratio, avgSalary, attendance)%>%
  head()
```

```
##   yearID teamID              name win_ratio avgSalary attendance
## 1   1985    ATL    Atlanta Braves 0.4074074  673045.5    1350137
## 2   1985    BAL Baltimore Orioles 0.5155280  525486.9    2132387
## 3   1985    BOS    Boston Red Sox 0.5000000  435902.4    1786633
## 4   1985    CAL California Angels 0.5555556  515281.9    2567427
## 5   1985    CHA Chicago White Sox 0.5246914  468865.6    1669888
## 6   1985    CHN      Chicago Cubs 0.4782609  577405.3    2161534
```

Let us visualize the relationship between avgSalary and win_ratio for the the last 16 years. One of the consequences of an inner join is that it will only contain data for years that are present in both tables. Salary data is only available until 2016, so even though Teams data goes past 2016, it is not present in the joined table.

```
library(ggplot2)
Teams %>%
  inner_join(team_salary, by = c('yearID' = 'yearID', 'teamID' = 'teamID'))%>%
  mutate(win_ratio = W/(W+L))%>%
  select(yearID, teamID, name, win_ratio, avgSalary, attendance)%>%
  filter(yearID%in%2001:2016)%>%
  ggplot(aes(x=avgSalary,y=win_ratio))+
  geom_point(size=0.8)+
  geom_smooth(method='lm', se=F,color='brown')+
  facet_wrap(~yearID,scales = 'free')
```



By joining two different tables, we were able to examine the relationship between avgSalary paid to each player and the team win-ratio. What do you think? (Caveat: If you suspect relationship of salary on performance is delayed, then you can examine the relationship between win_ratio and a lagged avgSalary)