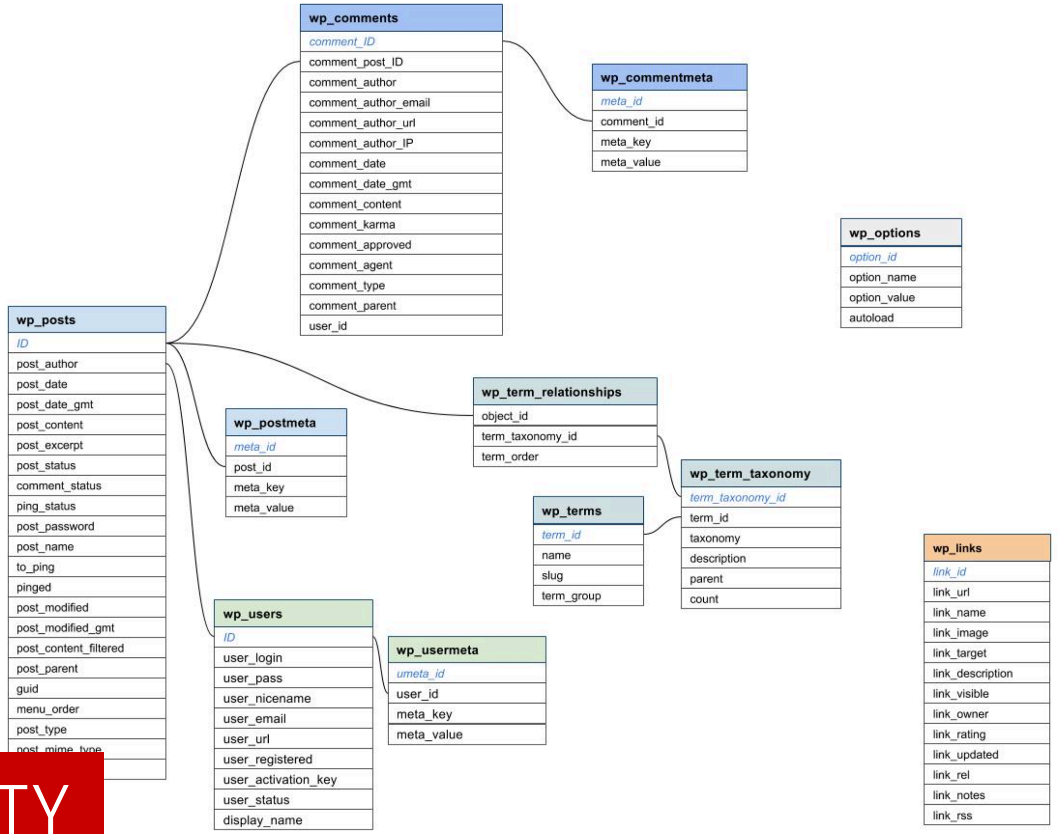
NC STATE UNIVERSITY
DEPARTMENT OF STATISTICS

SQL Style Joins

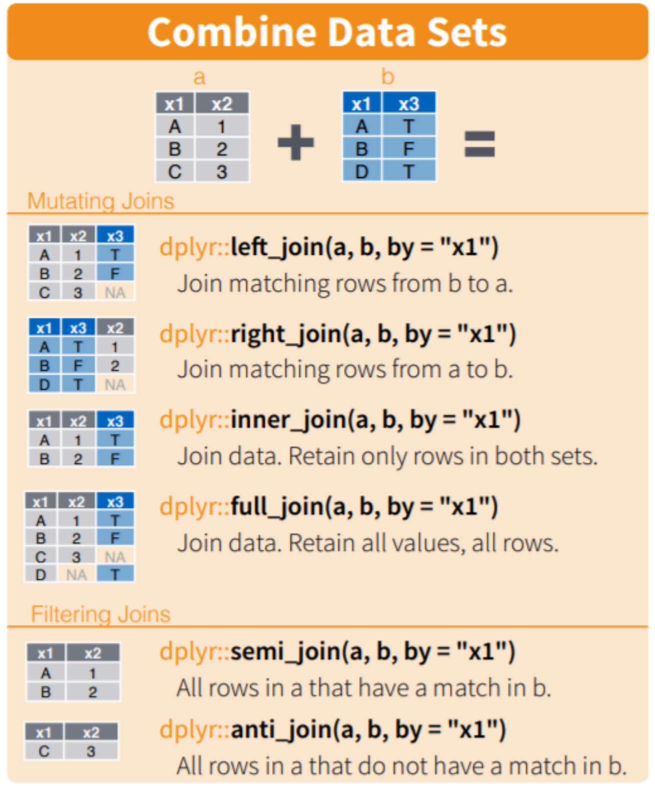
Justin Post

Relational Databases

- Often want to combine data from multiple tables to summarize/model



- The common types of joins we do are given below! (Using `dplyr` not the particular SQL language.)



Combine Data Sets

a

x1	x2
A	1
B	2
C	3

+

b

x1	x3
A	T
B	F
D	T

=

Mutating Joins

x1	x2	x3
A	1	T
B	2	F
C	3	NA

dplyr::**left_join(a, b, by = "x1")**
Join matching rows from b to a.

x1	x3	x2
A	T	1
B	F	2
D	T	NA

dplyr::**right_join(a, b, by = "x1")**
Join matching rows from a to b.

x1	x2	x3
A	1	T
B	2	F

dplyr::**inner_join(a, b, by = "x1")**
Join data. Retain only rows in both sets.

x1	x2	x3
A	1	T
B	2	F
C	3	NA
D	NA	T

dplyr::**full_join(a, b, by = "x1")**
Join data. Retain all values, all rows.

Filtering Joins

x1	x2
A	1
B	2

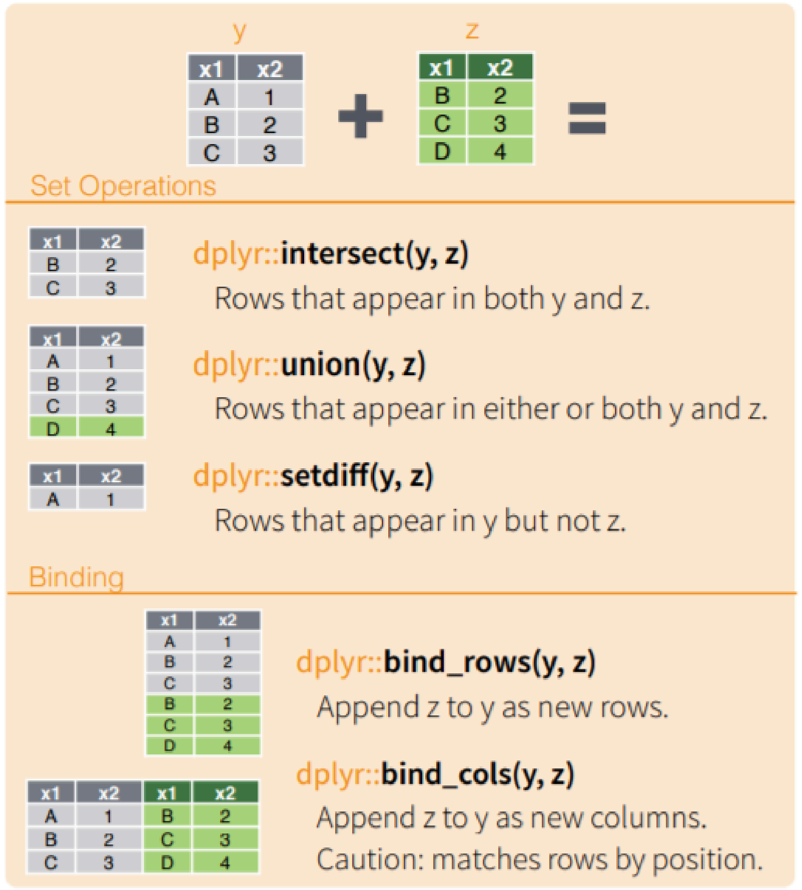
dplyr::**semi_join(a, b, by = "x1")**
All rows in a that have a match in b.

x1	x2
C	3

dplyr::**anti_join(a, b, by = "x1")**
All rows in a that do not have a match in b.

NC STATE UNIVERSITY

- We often need some different logic to make our joins work. That exists in `dplyr` as well!



y

x1	x2
A	1
B	2
C	3

z

x1	x2
B	2
C	3
D	4

Set Operations

x1	x2
B	2
C	3

dplyr::**intersect(y, z)**
Rows that appear in both y and z.

x1	x2
A	1
B	2
C	3
D	4

dplyr::**union(y, z)**
Rows that appear in either or both y and z.

x1	x2
A	1

dplyr::**setdiff(y, z)**
Rows that appear in y but not z.

Binding

x1	x2
A	1
B	2
C	3
B	2
C	3
D	4

dplyr::**bind_rows(y, z)**
Append z to y as new rows.

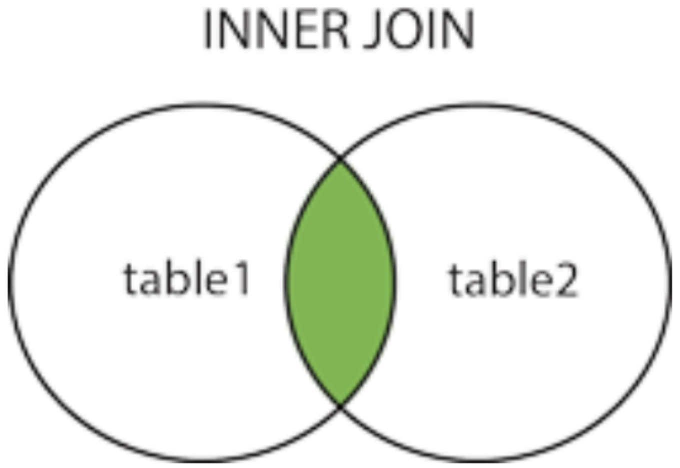
x1	x2	x1	x2
A	1	B	2
B	2	C	3
C	3	D	4

dplyr::**bind_cols(y, z)**
Append z to y as new columns.
Caution: matches rows by position.

NC STATE UNIVERSITY

Joins

- Let's go through our common joins!
- Inner Join: Returns records with matching keys in both tables



Inner Join

Make our connection and look at the tables

```
library(DBI)
library(dplyr)
con <- dbConnect(RSQLite::SQLite(), "lahman.db")
dbListTables(con)
```

```
##  [1] "AllstarFull"         "Appearances"         "AwardsManagers"
##  [4] "AwardsPlayers"       "AwardsShareManagers" "AwardsSharePlayers"
##  [7] "Batting"             "BattingPost"         "CollegePlaying"
## [10] "Fielding"            "FieldingOF"          "FieldingOFsplit"
## [13] "FieldingPost"        "HallOfFame"          "HomeGames"
## [16] "LahmanData"          "Managers"            "ManagersHalf"
## [19] "Parks"               "People"              "Pitching"
## [22] "PitchingPost"        "Salaries"            "Schools"
## [25] "SeriesPost"          "Teams"               "TeamsFranchises"
## [28] "TeamsHalf"           "battingLabels"       "fieldingLabels"
## [31] "pitchingLabels"
```

NC STATE UNIVERSITY

Inner Join

Combine the `Batting` table and the `Pitching` table on common variables

```
inner_join(tbl(con, "Batting") |> filter(yearID == 2000),
           tbl(con, "Pitching") |> filter(yearID == 2000),
           by = join_by(playerID == playerID, stint == stint, teamID == teamID, lgID == lgID)) |>
  collect()
```

```
## # A tibble: 677 × 48
##   playerID yearID.x stint teamID lgID    G.x    AB   R.x   H.x   X2B   X3B  HR.x
##   <chr>       <int> <int> <chr>  <chr> <int> <int> <int> <int> <int> <int> <int>
## 1 abbotpa…     2000     1 SEA    AL       35     5     1     2     1     0     0
## 2 aceveju…     2000     1 MIL    NL       62     1     1     0     0     0     0
## 3 adamste…     2000     1 LAN    NL       66     2     0     0     0     0     0
## 4 aguilri…     2000     1 CHN    NL       54     0     0     0     0     0     0
## 5 aldresc…     2000     1 PHI    NL       23     0     0     0     0     0     0
## # ℹ 672 more rows
## # ℹ 36 more variables: RBI <int>, SB <int>, CS <int>, BB.x <int>, SO.x <int>,
## #   IBB.x <int>, HBP.x <int>, SH.x <int>, SF.x <int>, GIDP.x <int>,
## #   yearID.y <int>, W <int>, L <int>, G.y <int>, GS <int>, CG <int>, SHO <int>,
## #   SV <int>, IPouts <int>, H.y <int>, ER <int>, HR.y <int>, BB.y <int>,
## #   SO.y <int>, BAOpp <dbl>, ERA <dbl>, IBB.y <int>, WP <int>, HBP.y <int>,
## #   BK <int>, BFP <int>, GF <int>, R.y <int>, SH.y <int>, SF.y <int>, …
```

NC STATE UNIVERSITY

Can Write SQL code instead

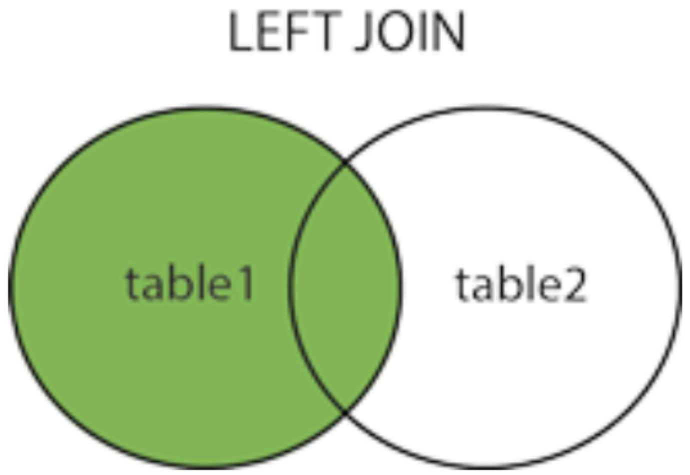
- (I'm not a great SQL programmer)

```
tbl(con, sql(
"SELECT p.playerID as pplayerID,
        p.stint as pstint,
        p.teamID as pteamID,
        p.lgID as plgID,
        p.G as pG,
        p.HR as pHR,
        p.BB as pBB,
        p.SO as pSO,
        p.HBP as pHBP,
        p.R as pR,
        p.SF as pSF,
        p.GIDP as pGIDP,
        p.IBB as pIBB,
        p.SH as pSH,
        p.W, p.L, p.GS, p.CG, p.SHO, p.SV, p.IPouts, p.ER, p.BAopp,
        p.ERA, p.WP, p.BK, p.BFP, p.GF,
        b.*
FROM Pitching as p
INNER JOIN Batting as b on ((p.playerID = b.playerID) AND (pstint = b.stint) AND (pteamID = b.teamID) AND (plgID
WHERE b.yearID = 2000 AND p.yearID = 2000"
))
```

NC STATE UNIVERSITY

Joins

- Left Join: Returns all records from the 'left' table and any matching records from the 'right' table



NC STATE UNIVERSITY

Left Join: Return left table and matching right records

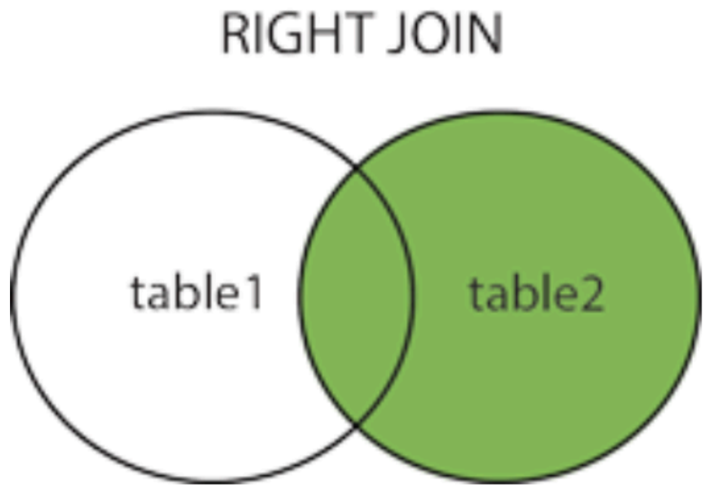
```
left_join(tbl(con, "Batting") |> filter(yearID == 2000),
           tbl(con, "Pitching") |> filter(yearID == 2000),
           by = join_by(playerID == playerID, stint == stint, teamID == teamID, lgID == lgID)) |>
  collect() |>
  select(playerID, ERA, everything())
```

```
## # A tibble: 1,384 × 48
##   playerID   ERA yearID.x stint teamID lgID    G.x    AB   R.x   H.x   X2B   X3B
##   <chr>     <dbl>    <int> <int> <chr>  <chr> <int> <int> <int> <int> <int> <int>
## 1 abbotje… NA        2000     1 CHA    AL       80   215    31    59    15     1
## 2 abbotku… NA        2000     1 NYN    NL       79   157    22    34     7     1
## 3 abbotpa…  4.22     2000     1 SEA    AL       35     5     1     2     1     0
## 4 abreubo… NA        2000     1 PHI    NL      154   576   103   182    42    10
## 5 aceveju…  3.81     2000     1 MIL    NL       62     1     1     0     0     0
## # ℹ 1,379 more rows
## # ℹ 36 more variables: HR.x <int>, RBI <int>, SB <int>, CS <int>, BB.x <int>,
## #   SO.x <int>, IBB.x <int>, HBP.x <int>, SH.x <int>, SF.x <int>, GIDP.x <int>,
## #   yearID.y <int>, W <int>, L <int>, G.y <int>, GS <int>, CG <int>, SHO <int>,
## #   SV <int>, IPouts <int>, H.y <int>, ER <int>, HR.y <int>, BB.y <int>,
## #   SO.y <int>, BAOpp <dbl>, IBB.y <int>, WP <int>, HBP.y <int>, BK <int>,
## #   BFP <int>, GF <int>, R.y <int>, SH.y <int>, SF.y <int>, GIDP.y <int>
```

NC STATE UNIVERSITY

Joins

- Right Join: Returns all records from the 'right' table and any matching records from the 'left' table



NC STATE UNIVERSITY

Right Join

- Just do a left join and switch the table (or use `right_join()`)

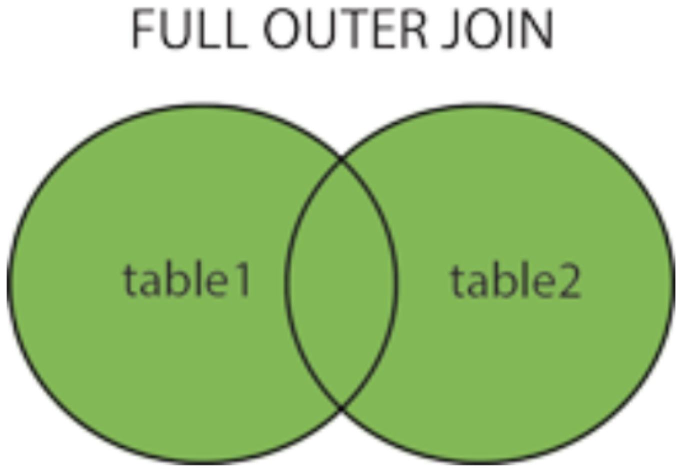
```
right_join(tbl(con, "Batting") |> filter(yearID == 2000),
           tbl(con, "Pitching") |> filter(yearID == 2000),
           by = join_by(playerID == playerID, stint == stint, teamID == teamID, lgID == lgID)) |>
  collect() |>
  select(playerID, ERA, everything())
```

```
## # A tibble: 677 × 48
##   playerID   ERA yearID.x stint teamID lgID    G.x    AB   R.x   H.x   X2B   X3B
##   <chr>    <dbl>    <int> <int> <chr>  <chr> <int> <int> <int> <int> <int> <int>
## 1 abbotpa…  4.22     2000     1 SEA    AL       35     5     1     2     1     0
## 2 aceveju…  3.81     2000     1 MIL    NL       62     1     1     0     0     0
## 3 adamste…  3.52     2000     1 LAN    NL       66     2     0     0     0     0
## 4 aguilri…  4.91     2000     1 CHN    NL       54     0     0     0     0     0
## 5 aldresc…  5.75     2000     1 PHI    NL       23     0     0     0     0     0
## # ℹ 672 more rows
## # ℹ 36 more variables: HR.x <int>, RBI <int>, SB <int>, CS <int>, BB.x <int>,
## #   SO.x <int>, IBB.x <int>, HBP.x <int>, SH.x <int>, SF.x <int>, GIDP.x <int>,
## #   yearID.y <int>, W <int>, L <int>, G.y <int>, GS <int>, CG <int>, SHO <int>,
## #   SV <int>, IPouts <int>, H.y <int>, ER <int>, HR.y <int>, BB.y <int>,
## #   SO.y <int>, BAOpp <dbl>, IBB.y <int>, WP <int>, HBP.y <int>, BK <int>,
## #   BFP <int>, GF <int>, R.y <int>, SH.y <int>, SF.y <int>, GIDP.y <int>
```

NC STATE UNIVERSITY

Joins

- Outer Join: Returns all records when there is a match from the 'left' or 'right' table (also called a **full join**)



NC STATE UNIVERSITY

Outer Join: Return all matches from both tables

(All players are in the Batting table even if they have no at bats!)

```
full_join(tbl(con, "Batting") |> filter(yearID == 2000),
           tbl(con, "Pitching") |> filter(yearID == 2000),
           by = join_by(playerID == playerID, stint == stint, teamID == teamID, lgID == lgID)) |>
  collect()
```

```
## # A tibble: 1,384 × 48
##   playerID yearID.x stint teamID lgID    G.x    AB   R.x   H.x   X2B   X3B  HR.x
##   <chr>       <int> <int> <chr>  <chr> <int> <int> <int> <int> <int> <int> <int>
## 1 abbotje…     2000     1 CHA    AL       80   215    31    59    15     1     3
## 2 abbotku…     2000     1 NYN    NL       79   157    22    34     7     1     6
## 3 abbotpa…     2000     1 SEA    AL       35     5     1     2     1     0     0
## 4 abreubo…     2000     1 PHI    NL      154   576   103   182    42    10    25
## 5 aceveju…     2000     1 MIL    NL       62     1     1     0     0     0     0
## # ℹ 1,379 more rows
## # ℹ 36 more variables: RBI <int>, SB <int>, CS <int>, BB.x <int>, SO.x <int>,
## #   IBB.x <int>, HBP.x <int>, SH.x <int>, SF.x <int>, GIDP.x <int>,
## #   yearID.y <int>, W <int>, L <int>, G.y <int>, GS <int>, CG <int>, SHO <int>,
## #   SV <int>, IPouts <int>, H.y <int>, ER <int>, HR.y <int>, BB.y <int>,
## #   SO.y <int>, BAOpp <dbl>, ERA <dbl>, IBB.y <int>, WP <int>, HBP.y <int>,
## #   BK <int>, BFP <int>, GF <int>, R.y <int>, SH.y <int>, SF.y <int>, …
```

Other Joins

Those are the major joins covered by `dplyr`. Lots of other joins out there!

- See here for examples!
  - The right sidebar has more than the standard joins.
- Also ways to do if then else type logic, intersections, etc. in SQL
- Can do basic summaries using SQL as well (including grouping), but we'll just use `dplyr` for that!

Recap

- Joins are combining two tables
- inner_join - match records that appear in both tables
- left/right join
- full outer join