Data Wrangling 2

DATA MANIPULATION WITH `dplyr` PACKAGE

- **`dplyr`** package was written by the most popular R programmer Hadley Wickham
- It contains a set of functions (or "verbs") that perform *common data manipulation operations*

Some commonly used `dplyr` functions

Function	Description
`select()`	Selecting variables
`filter()`	Filter (subset) rows
`group-by()`	Group the data
`summarise()`	Summarise data
`arrange()`	Sort the data
`mutate()`	Create new variables
`join()`	Joining data tables

Let's look into a practical approach:

In this tutorial, we are using the following data which contains income generated by states from year 2002 to 2015. This dataset contains 51 observations (rows) and 16 variables (columns)

Note : This data do not contain actual income figures of the states. To download the dataset, click on this link - **Dataset** and then right click and hit Save as option.

Load the Dataset

```
library(readr)
data <- read_csv("C:/Users/User/Downloads/raw.githubusercontent.com_deepanshu88_data_master_sampledata.csv")
```

```
## Rows: 51 Columns: 16
## ── Column specification ────────────────────────────────────────────────────
## Delimiter: ","
## chr  (2): Index, State
## dbl (14): Y2002, Y2003, Y2004, Y2005, Y2006, Y2007, Y2008, Y2009, Y2010, Y20...
##
## ⓘ Use `spec()` to retrieve the full column specification for this data.
## ⓘ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Create as "Tibble"

```
library(tibble)
data=as.tibble(data)
```

```
## Warning: `as.tibble()` was deprecated in tibble 2.0.0.
## ⓘ Please use `as_tibble()` instead.
## ⓘ The signature and semantics have changed, see `?as_tibble`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
head(data)
```

```
## # A tibble: 6 × 16
##   Index State      Y2002  Y2003  Y2004  Y2005  Y2006  Y2007  Y2008  Y2009
Y2010
##   <chr> <chr>      <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
<dbl>
## 1 A     Alabama   1.30e6 1.32e6 1.12e6 1.49e6 1.11e6 1.44e6 1.95e6 1.94e6
1.24e6
## 2 A     Alaska    1.17e6 1.96e6 1.82e6 1.45e6 1.86e6 1.47e6 1.55e6 1.44e6
1.63e6
## 3 A     Arizona   1.74e6 1.97e6 1.38e6 1.78e6 1.10e6 1.11e6 1.75e6 1.55e6
1.30e6
## 4 A     Arkansas  1.49e6 1.99e6 1.12e6 1.95e6 1.67e6 1.80e6 1.19e6 1.63e6
1.67e6
```

```
## 5 C     Californ… 1.69e6 1.68e6 1.89e6 1.48e6 1.74e6 1.81e6 1.49e6 1.66e6
1.62e6
## 6 C     Colorado  1.34e6 1.88e6 1.89e6 1.24e6 1.87e6 1.81e6 1.88e6 1.75e6
1.91e6
## # ⓘ 5 more variables: Y2011 <dbl>, Y2012 <dbl>, Y2013 <dbl>, Y2014 <dbl>
,
## #   Y2015 <dbl>
```

Structure of the Dataset

```
str(data)
```

```
## tibble [51 × 16] (S3: tbl_df/tbl/data.frame)
##  $ Index: chr [1:51] "A" "A" "A" "A" ...
##  $ State: chr [1:51] "Alabama" "Alaska" "Arizona" "Arkansas" ...
##  $ Y2002: num [1:51] 1296530 1170302 1742027 1485531 1685349 ...
##  $ Y2003: num [1:51] 1317711 1960378 1968140 1994927 1675807 ...
##  $ Y2004: num [1:51] 1118631 1818085 1377583 1119299 1889570 ...
##  $ Y2005: num [1:51] 1492583 1447852 1782199 1947979 1480280 ...
##  $ Y2006: num [1:51] 1107408 1861639 1102568 1669191 1735069 ...
##  $ Y2007: num [1:51] 1440134 1465841 1109382 1801213 1812546 ...
##  $ Y2008: num [1:51] 1945229 1551826 1752886 1188104 1487315 ...
##  $ Y2009: num [1:51] 1944173 1436541 1554330 1628980 1663809 ...
##  $ Y2010: num [1:51] 1237582 1629616 1300521 1669295 1624509 ...
##  $ Y2011: num [1:51] 1440756 1230866 1130709 1928238 1639670 ...
##  $ Y2012: num [1:51] 1186741 1512804 1907284 1216675 1921845 ...
##  $ Y2013: num [1:51] 1852841 1985302 1363279 1591896 1156536 ...
##  $ Y2014: num [1:51] 1558906 1580394 1525866 1360959 1388461 ...
##  $ Y2015: num [1:51] 1916661 1979143 1647724 1329341 1644607 ...
```

(a) Selects variables from "Index" to "Y2005".

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
data %>%
  select (Index:Y2005)
```

```
## # A tibble: 51 × 6
##    Index State                Y2002   Y2003   Y2004   Y2005
##    <chr> <chr>                <dbl>   <dbl>   <dbl>   <dbl>
##  1 A     Alabama            1296530 1317711 1118631 1492583
##  2 A     Alaska             1170302 1960378 1818085 1447852
##  3 A     Arizona            1742027 1968140 1377583 1782199
##  4 A     Arkansas           1485531 1994927 1119299 1947979
##  5 C     California         1685349 1675807 1889570 1480280
##  6 C     Colorado           1343824 1878473 1886149 1236697
##  7 C     Connecticut        1610512 1232844 1181949 1518933
##  8 D     Delaware           1330403 1268673 1706751 1403759
##  9 D     District of Columbia 1111437 1993741 1374643 1827949
## 10 F     Florida            1964626 1468852 1419738 1362787
## # ℹ 41 more rows
```

(b) Drop variables "Index", and "State" variables from data.

```
data %>%
  select (-Index, -State)
```

```
## # A tibble: 51 × 14
##      Y2002  Y2003  Y2004  Y2005  Y2006  Y2007  Y2008  Y2009  Y2010  Y2011
Y2012
##      <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
<dbl>
##  1 1296530 1.32e6 1.12e6 1.49e6 1.11e6 1.44e6 1.95e6 1.94e6 1.24e6 1.44e6
1.19e6
##  2 1170302 1.96e6 1.82e6 1.45e6 1.86e6 1.47e6 1.55e6 1.44e6 1.63e6 1.23e6
1.51e6
##  3 1742027 1.97e6 1.38e6 1.78e6 1.10e6 1.11e6 1.75e6 1.55e6 1.30e6 1.13e6
1.91e6
##  4 1485531 1.99e6 1.12e6 1.95e6 1.67e6 1.80e6 1.19e6 1.63e6 1.67e6 1.93e6
1.22e6
##  5 1685349 1.68e6 1.89e6 1.48e6 1.74e6 1.81e6 1.49e6 1.66e6 1.62e6 1.64e6
1.92e6
##  6 1343824 1.88e6 1.89e6 1.24e6 1.87e6 1.81e6 1.88e6 1.75e6 1.91e6 1.67e6
1.49e6
##  7 1610512 1.23e6 1.18e6 1.52e6 1.84e6 1.98e6 1.76e6 1.97e6 1.97e6 1.95e6
1.23e6
##  8 1330403 1.27e6 1.71e6 1.40e6 1.44e6 1.30e6 1.76e6 1.55e6 1.37e6 1.32e6
1.98e6
##  9 1111437 1.99e6 1.37e6 1.83e6 1.80e6 1.60e6 1.19e6 1.74e6 1.71e6 1.35e6
1.98e6
## 10 1964626 1.47e6 1.42e6 1.36e6 1.34e6 1.28e6 1.76e6 1.82e6 1.20e6 1.50e6
1.13e6
```

```
## # ⓘ 41 more rows
## # ⓘ 3 more variables: Y2013 <dbl>, Y2014 <dbl>, Y2015 <dbl>
```

(c) Filter rows in which Index is equal to C.

```
data %>%
  filter(Index == "C")
```

```
## # A tibble: 3 × 16
##   Index State      Y2002  Y2003  Y2004  Y2005  Y2006  Y2007  Y2008  Y2009
Y2010
##   <chr> <chr>      <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
<dbl>
## 1 C     Californ… 1.69e6 1.68e6 1.89e6 1.48e6 1.74e6 1.81e6 1.49e6 1.66e6
1.62e6
## 2 C     Colorado  1.34e6 1.88e6 1.89e6 1.24e6 1.87e6 1.81e6 1.88e6 1.75e6
1.91e6
## 3 C     Connecti… 1.61e6 1.23e6 1.18e6 1.52e6 1.84e6 1.98e6 1.76e6 1.97e6
1.97e6
## # ⓘ 5 more variables: Y2011 <dbl>, Y2012 <dbl>, Y2013 <dbl>, Y2014 <dbl>
,
## #   Y2015 <dbl>
```

(d) Filter rows having 'D' and 'F' in column 'Index'.

```
data %>%
  filter(Index %in% c("D", "F"))
```

```
## # A tibble: 3 × 16
##   Index State      Y2002  Y2003  Y2004  Y2005  Y2006  Y2007  Y2008  Y2009
Y2010
##   <chr> <chr>      <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
<dbl>
## 1 D     Delaware  1.33e6 1.27e6 1.71e6 1.40e6 1.44e6 1.30e6 1.76e6 1.55e6
1.37e6
## 2 D     District… 1.11e6 1.99e6 1.37e6 1.83e6 1.80e6 1.60e6 1.19e6 1.74e6
1.71e6
## 3 F     Florida   1.96e6 1.47e6 1.42e6 1.36e6 1.34e6 1.28e6 1.76e6 1.82e6
1.20e6
## # ⓘ 5 more variables: Y2011 <dbl>, Y2012 <dbl>, Y2013 <dbl>, Y2014 <dbl>
,
## #   Y2015 <dbl>
```

(e) Filter rows 'D' and 'F' from 'Index' and income greater than 1.2 million in Year 2004.

```
data %>%
  filter(Index %in% c("D", "F")& Y2004 >= 1200000)
```

```
## # A tibble: 3 × 16
##   Index State      Y2002  Y2003  Y2004  Y2005  Y2006  Y2007  Y2008  Y2009
Y2010
##   <chr> <chr>      <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
<dbl>
## 1 D     Delaware  1.33e6 1.27e6 1.71e6 1.40e6 1.44e6 1.30e6 1.76e6 1.55e6
1.37e6
## 2 D     District… 1.11e6 1.99e6 1.37e6 1.83e6 1.80e6 1.60e6 1.19e6 1.74e6
1.71e6
## 3 F     Florida   1.96e6 1.47e6 1.42e6 1.36e6 1.34e6 1.28e6 1.76e6 1.82e6
1.20e6
## # ⓘ 5 more variables: Y2011 <dbl>, Y2012 <dbl>, Y2013 <dbl>, Y2014 <dbl>
,
## #   Y2015 <dbl>
```

(f) Filter rows 'D' and 'F' from 'Index' or income greater than 1.2 million in Year 2004.

```
data %>%
  filter(Index %in% c("D", "F") | Y2004 >= 1200000)
```

```
## # A tibble: 42 × 16
##    Index State     Y2002  Y2003  Y2004  Y2005  Y2006  Y2007  Y2008  Y2009
Y2010
##    <chr> <chr>     <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
<dbl>
##  1 A     Alaska   1.17e6 1.96e6 1.82e6 1.45e6 1.86e6 1.47e6 1.55e6 1.44e6
1.63e6
##  2 A     Arizona  1.74e6 1.97e6 1.38e6 1.78e6 1.10e6 1.11e6 1.75e6 1.55e6
1.30e6
##  3 C     Califor… 1.69e6 1.68e6 1.89e6 1.48e6 1.74e6 1.81e6 1.49e6 1.66e6
1.62e6
##  4 C     Colorado 1.34e6 1.88e6 1.89e6 1.24e6 1.87e6 1.81e6 1.88e6 1.75e6
1.91e6
##  5 D     Delaware 1.33e6 1.27e6 1.71e6 1.40e6 1.44e6 1.30e6 1.76e6 1.55e6
1.37e6
##  6 D     Distric… 1.11e6 1.99e6 1.37e6 1.83e6 1.80e6 1.60e6 1.19e6 1.74e6
1.71e6
##  7 F     Florida  1.96e6 1.47e6 1.42e6 1.36e6 1.34e6 1.28e6 1.76e6 1.82e6
1.20e6
##  8 G     Georgia  1.93e6 1.54e6 1.81e6 1.78e6 1.33e6 1.22e6 1.77e6 1.63e6
1.15e6
##  9 H     Hawaii   1.46e6 1.20e6 1.21e6 1.25e6 1.46e6 1.43e6 1.92e6 1.93e6
```

```
1.33e6
## 10 I     Idaho    1.35e6 1.44e6 1.74e6 1.54e6 1.12e6 1.77e6 1.34e6 1.75e6
1.44e6
## # ⓘ 32 more rows
## # ⓘ 5 more variables: Y2011 <dbl>, Y2012 <dbl>, Y2013 <dbl>, Y2014 <dbl>
,
## #   Y2015 <dbl>
```

(g) Find the mean and median for the variable Y2012.

```
data %>%
  summarise(mean2012 = mean(Y2012), median2012=median(Y2012))
```

```
## # A tibble: 1 × 2
##   mean2012 median2012
##      <dbl>      <dbl>
## 1 1591135.    1643855
```

(h) Find mean and median of Y2010 and Y2015.

```
library(dplyr)
data %>%
  summarise(mean2010=mean(Y2010), mean2015=mean(Y2015),median2010=median(Y201
0), median2015=median(Y2015), na.rm = TRUE)
```

```
## # A tibble: 1 × 5
##   mean2010 mean2015 median2010 median2015 na.rm
##      <dbl>    <dbl>      <dbl>      <dbl> <lgl>
## 1 1504108. 1588297.    1498662    1627508 TRUE
```

(i) Arrange variable Y2014 by variable Index in ascending order.

```
data %>%
  arrange(Index, Y2014)
```

```
## # A tibble: 51 × 16
##    Index State     Y2002  Y2003  Y2004  Y2005  Y2006  Y2007  Y2008  Y2009
Y2010
##    <chr> <chr>     <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
<dbl>
##  1 A     Arkansas 1.49e6 1.99e6 1.12e6 1.95e6 1.67e6 1.80e6 1.19e6 1.63e6
1.67e6
##  2 A     Arizona  1.74e6 1.97e6 1.38e6 1.78e6 1.10e6 1.11e6 1.75e6 1.55e6
1.30e6
##  3 A     Alabama  1.30e6 1.32e6 1.12e6 1.49e6 1.11e6 1.44e6 1.95e6 1.94e6
1.24e6
```

```
##  4 A     Alaska   1.17e6 1.96e6 1.82e6 1.45e6 1.86e6 1.47e6 1.55e6 1.44e6
1.63e6
##  5 C     Colorado 1.34e6 1.88e6 1.89e6 1.24e6 1.87e6 1.81e6 1.88e6 1.75e6
1.91e6
##  6 C     Califor… 1.69e6 1.68e6 1.89e6 1.48e6 1.74e6 1.81e6 1.49e6 1.66e6
1.62e6
##  7 C     Connect… 1.61e6 1.23e6 1.18e6 1.52e6 1.84e6 1.98e6 1.76e6 1.97e6
1.97e6
##  8 D     Distric… 1.11e6 1.99e6 1.37e6 1.83e6 1.80e6 1.60e6 1.19e6 1.74e6
1.71e6
##  9 D     Delaware 1.33e6 1.27e6 1.71e6 1.40e6 1.44e6 1.30e6 1.76e6 1.55e6
1.37e6
## 10 F     Florida  1.96e6 1.47e6 1.42e6 1.36e6 1.34e6 1.28e6 1.76e6 1.82e6
1.20e6
## # ⓘ 41 more rows
## # ⓘ 5 more variables: Y2011 <dbl>, Y2012 <dbl>, Y2013 <dbl>, Y2014 <dbl>
,
## #   Y2015 <dbl>
```

(j) Calculate count and mean of variables Y2005 and Y2008 by variable Index.

```
data %>%
  group_by(Index) %>%
  summarise(n = n(),mean2005=mean(Y2005), mean2008=mean(Y2008)
  )
```

```
## # A tibble: 19 × 4
##    Index     n mean2005 mean2008
##    <chr> <int>    <dbl>    <dbl>
##  1 A         4 1667653. 1609511.
##  2 C         3 1411970  1708973.
##  3 D         2 1615854  1477670.
##  4 F         1 1362787  1756185
##  5 G         1 1779091  1773090
##  6 H         1 1245931  1919423
##  7 I         4 1348852. 1416165
##  8 K         2 1391309  1625663
##  9 L         1 1751920  1185085
## 10 M         8 1597524. 1515155.
## 11 N         8 1464692. 1541591.
## 12 O         3 1534168  1418718.
## 13 P         1 1122030  1274168
## 14 R         1 1961923  1151409
## 15 S         2 1437666  1211184.
## 16 T         2 1532192. 1806196
## 17 U         1 1241662  1939284
## 18 V         2 1716560  1463448.
## 19 W         4 1592654  1587891
```

(k) Calculate the variable rate=Y2006/Y2010.

```
data %>%
  mutate(rate=Y2006/Y2010)
```

```
## # A tibble: 51 × 17
##    Index State     Y2002  Y2003  Y2004  Y2005  Y2006  Y2007  Y2008  Y2009
Y2010
##    <chr> <chr>     <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
<dbl>
##  1 A     Alabama  1.30e6 1.32e6 1.12e6 1.49e6 1.11e6 1.44e6 1.95e6 1.94e6
1.24e6
##  2 A     Alaska   1.17e6 1.96e6 1.82e6 1.45e6 1.86e6 1.47e6 1.55e6 1.44e6
1.63e6
##  3 A     Arizona  1.74e6 1.97e6 1.38e6 1.78e6 1.10e6 1.11e6 1.75e6 1.55e6
1.30e6
##  4 A     Arkansas 1.49e6 1.99e6 1.12e6 1.95e6 1.67e6 1.80e6 1.19e6 1.63e6
1.67e6
##  5 C     Califor… 1.69e6 1.68e6 1.89e6 1.48e6 1.74e6 1.81e6 1.49e6 1.66e6
1.62e6
##  6 C     Colorado 1.34e6 1.88e6 1.89e6 1.24e6 1.87e6 1.81e6 1.88e6 1.75e6
1.91e6
##  7 C     Connect… 1.61e6 1.23e6 1.18e6 1.52e6 1.84e6 1.98e6 1.76e6 1.97e6
1.97e6
##  8 D     Delaware 1.33e6 1.27e6 1.71e6 1.40e6 1.44e6 1.30e6 1.76e6 1.55e6
1.37e6
##  9 D     Distric… 1.11e6 1.99e6 1.37e6 1.83e6 1.80e6 1.60e6 1.19e6 1.74e6
1.71e6
## 10 F     Florida  1.96e6 1.47e6 1.42e6 1.36e6 1.34e6 1.28e6 1.76e6 1.82e6
1.20e6
## # ⓘ 41 more rows
## # ⓘ 6 more variables: Y2011 <dbl>, Y2012 <dbl>, Y2013 <dbl>, Y2014 <dbl>
,
##   Y2015 <dbl>, rate <dbl>
```

(l) Calculate the cumulative sum of Y2014 and assign it to total, and select variables index, Y2014 and total.

```
data %>%
  mutate(total=cumsum(Y2014))%>%
  select(Index,Y2014, total)
```

```
## # A tibble: 51 × 3
##    Index   Y2014    total
##    <chr>   <dbl>    <dbl>
##  1 A     1558906  1558906
##  2 A     1580394  3139300
```

```
##  3 A     1525866  4665166
##  4 A     1360959  6026125
##  5 C     1388461  7414586
##  6 C     1383978  8798564
##  7 C     1503156 10301720
##  8 D     1803169 12104889
##  9 D     1782169 13887058
## 10 F     1407784 15294842
## # ⓘ 41 more rows
```

RELATIONAL DATA

- Data tables which are related to each other called ***relational data***
- This concept is similar to the *relational* database management systems (RDBMS)
- When working with relational data we may want to combine information from different data tables

Primary Key and Foreign Key

To connect each pair of tables we use unique identifiers called **keys**

- **primary key** - uniquely identifies an observation in its own table
- **foreign key** - uniquely identifies an observation in another table

NOTE:

A primary key and the corresponding foreign key in another table form a ***relation***

Work with Relational Data using R

- **dplyr** package is a powerful tool to work with relational data using R
- To get the details about relational data set we can use **dm** package

Let's look into a practical approach:

Here we use the data tables in the *nycflights13* package for this example

```
# Check variables in each tibble
library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.
0.0 ──
## ✔ forcats   1.0.0     ✔ purrr     1.0.1
## ✔ ggplot2   3.4.2     ✔ stringr   1.5.0
## ✔ lubridate 1.9.2     ✔ tidyr     1.3.0
## ── Conflicts ────────────────────────────────────────── tidyverse_conflict
s() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force a
ll conflicts to become errors

library(nycflights13)
flights

## # A tibble: 336,776 × 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_
time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <
int>
##  1  2013     1     1      517            515         2      830
819
##  2  2013     1     1      533            529         4      850
830
##  3  2013     1     1      542            540         2      923
850
##  4  2013     1     1      544            545        -1     1004
1022
##  5  2013     1     1      554            600        -6      812
837
```

```
##  6  2013     1     1      554            558        -4      740
728
##  7  2013     1     1      555            600        -5      913
854
##  8  2013     1     1      557            600        -3      709
723
##  9  2013     1     1      557            600        -3      838
846
## 10  2013     1     1      558            600        -2      753
745
## # ⓘ 336,766 more rows
## # ⓘ 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl
>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

airlines

```
## # A tibble: 16 × 2
##    carrier name
##    <chr>   <chr>
##  1 9E      Endeavor Air Inc.
##  2 AA      American Airlines Inc.
##  3 AS      Alaska Airlines Inc.
##  4 B6      JetBlue Airways
##  5 DL      Delta Air Lines Inc.
##  6 EV      ExpressJet Airlines Inc.
##  7 F9      Frontier Airlines Inc.
##  8 FL      AirTran Airways Corporation
##  9 HA      Hawaiian Airlines Inc.
## 10 MQ      Envoy Air
## 11 OO      SkyWest Airlines Inc.
## 12 UA      United Air Lines Inc.
## 13 US      US Airways Inc.
## 14 VX      Virgin America
## 15 WN      Southwest Airlines Co.
## 16 YV      Mesa Airlines Inc.
```

airports

```
## # A tibble: 1,458 × 8
##    faa   name                            lat    lon   alt    tz dst   tzo
ne
##    <chr> <chr>                         <dbl>  <dbl> <dbl> <dbl> <chr> <ch
r>
##  1 04G   Lansdowne Airport              41.1  -80.6  1044    -5 A     Ame
rica/…
##  2 06A   Moton Field Municipal Airport  32.5  -85.7   264    -6 A     Ame
rica/…
##  3 06C   Schaumburg Regional            42.0  -88.1   801    -6 A     Ame
```

```
rica/…
##  4 06N   Randall Airport                41.4  -74.4   523    -5 A     Ame
rica/…
##  5 09J   Jekyll Island Airport          31.1  -81.4    11    -5 A     Ame
rica/…
##  6 0A9   Elizabethton Municipal Airport  36.4  -82.2  1593    -5 A     Ame
rica/…
##  7 0G6   Williams County Airport         41.5  -84.5   730    -5 A     Ame
rica/…
##  8 0G7   Finger Lakes Regional Airport   42.9  -76.8   492    -5 A     Ame
rica/…
##  9 0P2   Shoestring Aviation Airfield    39.8  -76.6  1000    -5 U     Ame
rica/…
## 10 0S9   Jefferson County Intl           48.1 -123.    108    -8 A     Ame
rica/…
## # ⓘ 1,448 more rows

planes

## # A tibble: 3,322 × 9
##    tailnum  year type              manufacturer model engines seats speed
engine
##    <chr>   <int> <chr>             <chr>        <chr>   <int> <int> <int>
<chr>
##  1 N10156   2004 Fixed wing multi… EMBRAER      EMB-…       2    55    NA
Turbo…
##  2 N102UW   1998 Fixed wing multi… AIRBUS INDU… A320…       2   182    NA
Turbo…
##  3 N103US   1999 Fixed wing multi… AIRBUS INDU… A320…       2   182    NA
Turbo…
##  4 N104UW   1999 Fixed wing multi… AIRBUS INDU… A320…       2   182    NA
Turbo…
##  5 N10575   2002 Fixed wing multi… EMBRAER      EMB-…       2    55    NA
Turbo…
##  6 N105UW   1999 Fixed wing multi… AIRBUS INDU… A320…       2   182    NA
Turbo…
##  7 N107US   1999 Fixed wing multi… AIRBUS INDU… A320…       2   182    NA
Turbo…
##  8 N108UW   1999 Fixed wing multi… AIRBUS INDU… A320…       2   182    NA
Turbo…
##  9 N109UW   1999 Fixed wing multi… AIRBUS INDU… A320…       2   182    NA
Turbo…
## 10 N110UW   1999 Fixed wing multi… AIRBUS INDU… A320…       2   182    NA
Turbo…
## # ⓘ 3,312 more rows

weather

## # A tibble: 26,115 × 15
##    origin  year month   day  hour  temp  dewp humid wind_dir wind_speed
```

```
##    <chr>  <int> <int> <int> <int> <dbl> <dbl> <dbl>    <dbl>      <dbl>
##  1 EWR     2013     1     1     1  39.0  26.1  59.4      270      10.4
##  2 EWR     2013     1     1     2  39.0  27.0  61.6      250       8.06
##  3 EWR     2013     1     1     3  39.0  28.0  64.4      240      11.5
##  4 EWR     2013     1     1     4  39.9  28.0  62.2      250      12.7
##  5 EWR     2013     1     1     5  39.0  28.0  64.4      260      12.7
##  6 EWR     2013     1     1     6  37.9  28.0  67.2      240      11.5
##  7 EWR     2013     1     1     7  39.0  28.0  64.4      240      15.0
##  8 EWR     2013     1     1     8  39.9  28.0  62.2      250      10.4
##  9 EWR     2013     1     1     9  39.9  28.0  62.2      260      15.0
## 10 EWR     2013     1     1    10  41    28.0  59.6      260      13.8
## # ⓘ 26,105 more rows
## # ⓘ 5 more variables: wind_gust <dbl>, precip <dbl>, pressure <dbl>,
## #   visib <dbl>, time_hour <dttm>
```

Use dm package to get some details about the data set.

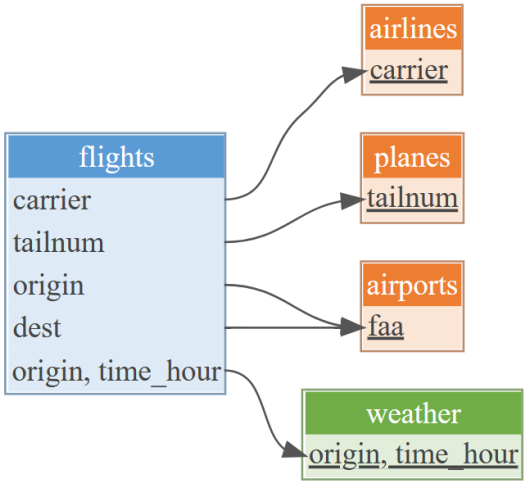
Visual Representation of the Relationships

```
library(dm)

##
## Attaching package: 'dm'

## The following object is masked from 'package:stats':
##
##     filter

dm <- dm_nycflights13(cycle = TRUE)
dm %>%
  dm_draw()
```



Get details all primary keys

```
dm %>%
  dm_get_all_pks()
```

```
## # A tibble: 4 × 3
##   table    pk_col            autoincrement
##   <chr>    <keys>            <lgl>
## 1 airlines carrier           FALSE
## 2 airports faa               FALSE
## 3 planes   tailnum           FALSE
## 4 weather  origin, time_hour FALSE
```

Check the suitability of each variable of a data set to serve as a primary key

```
dm %>%
  dm_enum_pk_candidates(airports)
```

```
## # A tibble: 8 × 3
##   columns candidate why
##   <keys>  <lgl>     <chr>
## 1 faa     TRUE      ""
## 2 name    TRUE      ""
## 3 lat     TRUE      ""
## 4 lon     TRUE      ""
## 5 alt     FALSE     "has duplicate values: 30 (4), 13 (3), 9 (2), 19 (2),
26 (2…
## 6 tz      FALSE     "has duplicate values: -5 (48), -6 (21), -8 (12), -7 (
4)"
## 7 dst     FALSE     "has duplicate values: A (84), N (2)"
## 8 tzone   FALSE     "has duplicate values: America/New_York (48), America/
Chica…
```

Identify foreign keys

```
dm %>%
  dm_enum_fk_candidates(flights, airlines)
```

```
## # A tibble: 19 × 3
##    columns        candidate why
##    <keys>         <lgl>     <chr>
##  1 carrier        TRUE      ""
##  2 year           FALSE     "Can't join `x$value1` with `y$value1` due to
incom…
```

```
##  3 month          FALSE     "Can't join `x$value1` with `y$value1` due to
incom…
##  4 day            FALSE     "Can't join `x$value1` with `y$value1` due to
incom…
##  5 dep_time       FALSE     "Can't join `x$value1` with `y$value1` due to
incom…
##  6 sched_dep_time FALSE     "Can't join `x$value1` with `y$value1` due to
incom…
##  7 dep_delay      FALSE     "Can't join `x$value1` with `y$value1` due to
incom…
##  8 arr_time       FALSE     "Can't join `x$value1` with `y$value1` due to
incom…
##  9 sched_arr_time FALSE     "Can't join `x$value1` with `y$value1` due to
incom…
## 10 arr_delay      FALSE     "Can't join `x$value1` with `y$value1` due to
incom…
## 11 flight         FALSE     "Can't join `x$value1` with `y$value1` due to
incom…
## 12 tailnum        FALSE     "values of `flights$tailnum` not in `airlines$
carri…
## 13 origin         FALSE     "values of `flights$origin` not in `airlines$c
arrie…
## 14 dest           FALSE     "values of `flights$dest` not in `airlines$car
rier`…
## 15 air_time       FALSE     "Can't join `x$value1` with `y$value1` due to
incom…
## 16 distance       FALSE     "Can't join `x$value1` with `y$value1` due to
incom…
## 17 hour           FALSE     "Can't join `x$value1` with `y$value1` due to
incom…
## 18 minute         FALSE     "Can't join `x$value1` with `y$value1` due to
incom…
## 19 time_hour      FALSE     "Can't join `x$value1` with `y$value1` due to
incom…
```

Extract a summary of all foreign key relations

```
dm %>%
  dm_get_all_fks()

## # A tibble: 5 × 5
##   child_table child_fk_cols     parent_table parent_key_cols   on_delete
##   <chr>       <keys>            <chr>        <keys>            <chr>
## 1 flights     carrier           airlines     carrier           no_action
## 2 flights     origin            airports     faa               no_action
## 3 flights     dest              airports     faa               no_action
```

```
## 4 flights     tailnum           planes       tailnum           no_action
## 5 flights     origin, time_hour weather      origin, time_hour no_action
```

Joins

Joins are used to access data from multiple tables based on logical relationships between them

`dplyr` has several types of ***Joins***

Types of *joins* in `dplyr`

Suppose we have two data tables x and y.

- **inner_join(x,y) -** keeps observations common to xand y.
- **left_join(x,y)** - keeps all observations in x.
- **right_join(x,y)** - keeps all observations in y.
- **full_join(x,y) -** keeps all observations in x and y.
- **semi_join(x, y)** - keeps all observations in x that have a match in y.
- **anti_join(x, y)** - drops all observations in x that have a match in y.

Let's look into a practical approach:

For this Tutorial let's create two tables *data1* and *data2*

```
# Create first example data frame
data1 <- data.frame(ID = 1:2,
                    X1 = c("a1", "a2"))
```

```r
 # Create second example data frame
data2 <- data.frame(ID = 2:3,
                    X2 = c("b1", "b2"))

data1

##   ID X1
## 1  1 a1
## 2  2 a2

data2

##   ID X2
## 1  2 b1
## 2  3 b2
```

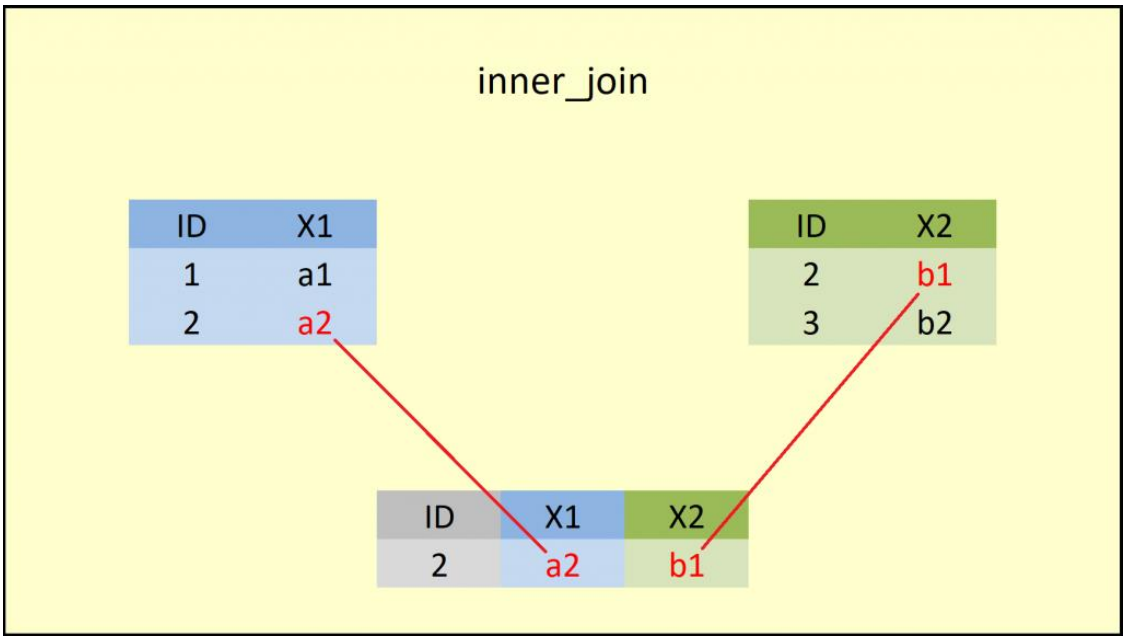
inner_join

- keeps observations common to *data1* and *data2*

```r
inner_join(data1, data2)

## Joining with `by = join_by(ID)`

##   ID X1 X2
## 1  2 a2 b1
```



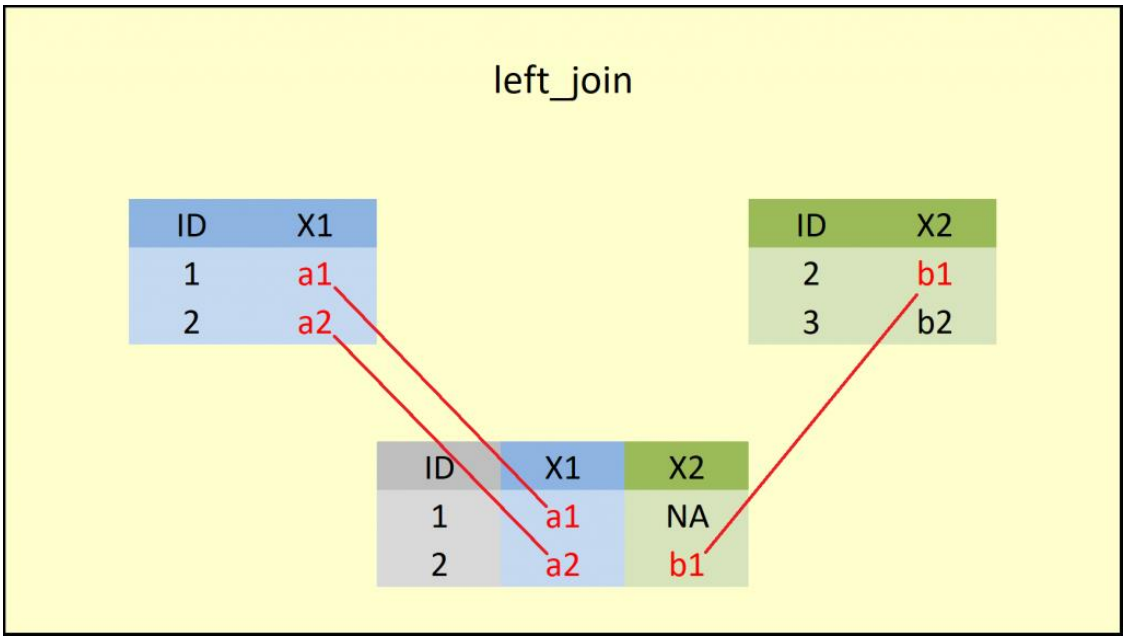
left_join

- keeps all observations in left table (*data1*)

```
left_join(data1, data2)
```

```
## Joining with `by = join_by(ID)`
```

```
##   ID X1   X2
## 1  1 a1 <NA>
## 2  2 a2   b1
```



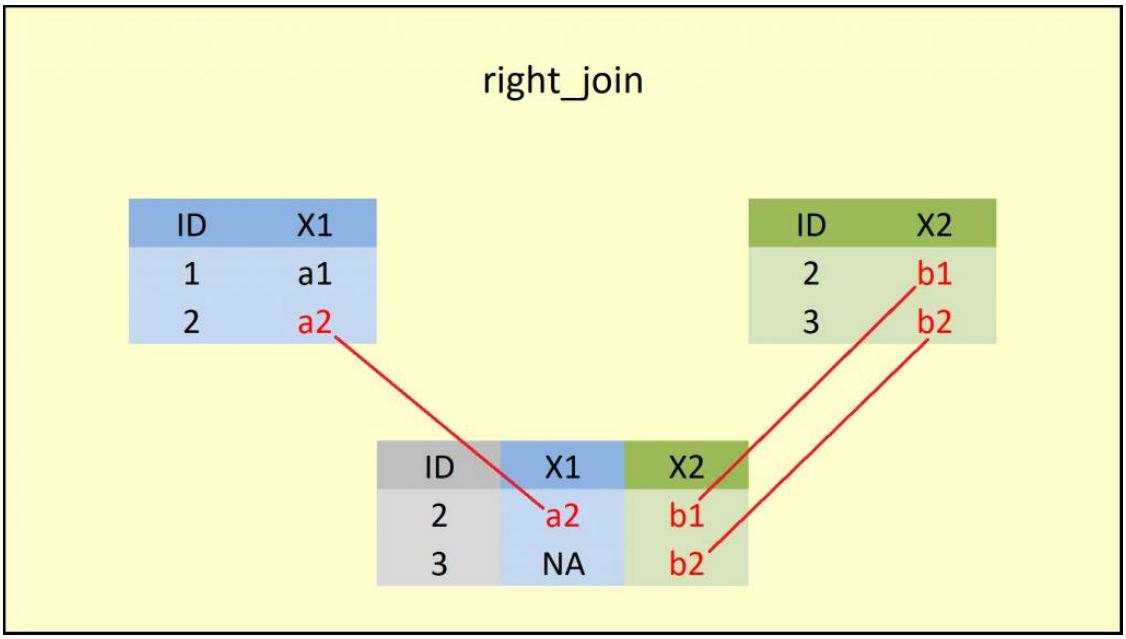
right_join

- keeps all observations in right table (*data2*)

```
right_join(data1, data2)
```

```
## Joining with `by = join_by(ID)`
```

```
##   ID   X1 X2
## 1  2   a2 b1
## 2  3 <NA> b2
```



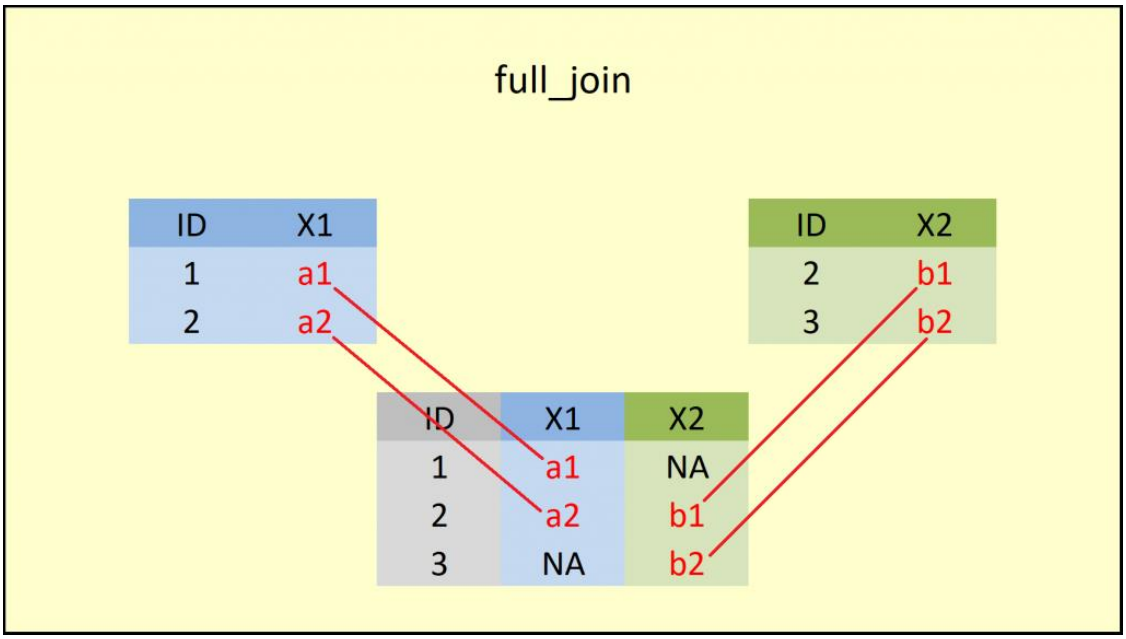
full_join

- keeps all observations in *data1* and data2

```
full_join(data1, data2)

## Joining with `by = join_by(ID)`

##   ID   X1   X2
## 1  1   a1 <NA>
## 2  2   a2   b1
## 3  3 <NA>   b2
```



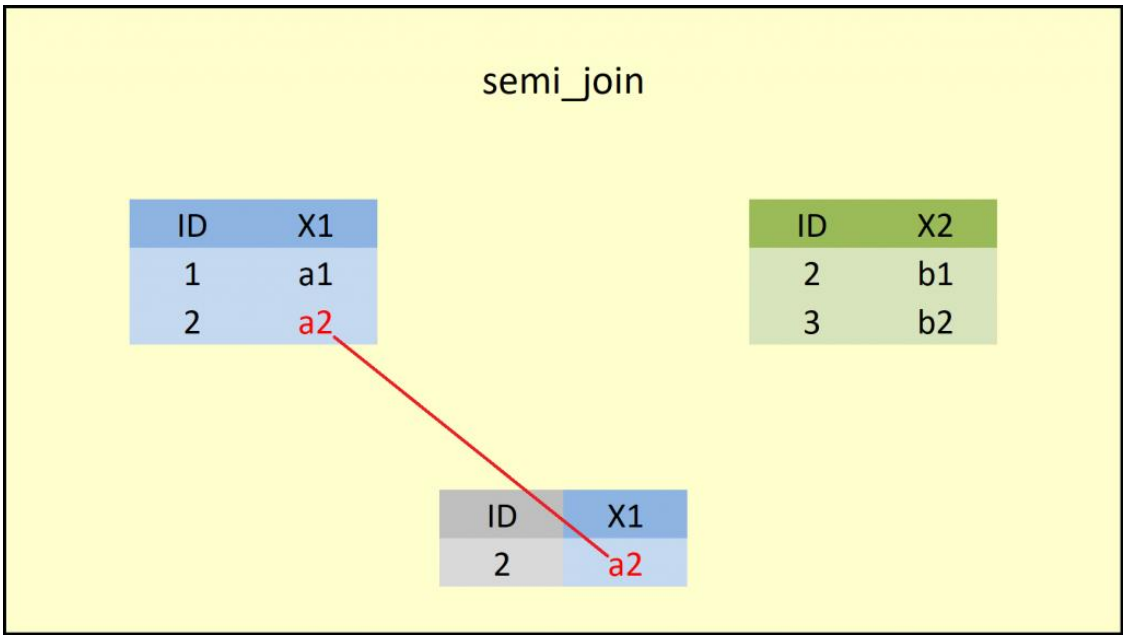
semi_join

- keeps all observations in *data1* that have a match in *data2*

```
semi_join(data1, data2)

## Joining with `by = join_by(ID)`

##   ID X1
## 1  2 a2
```



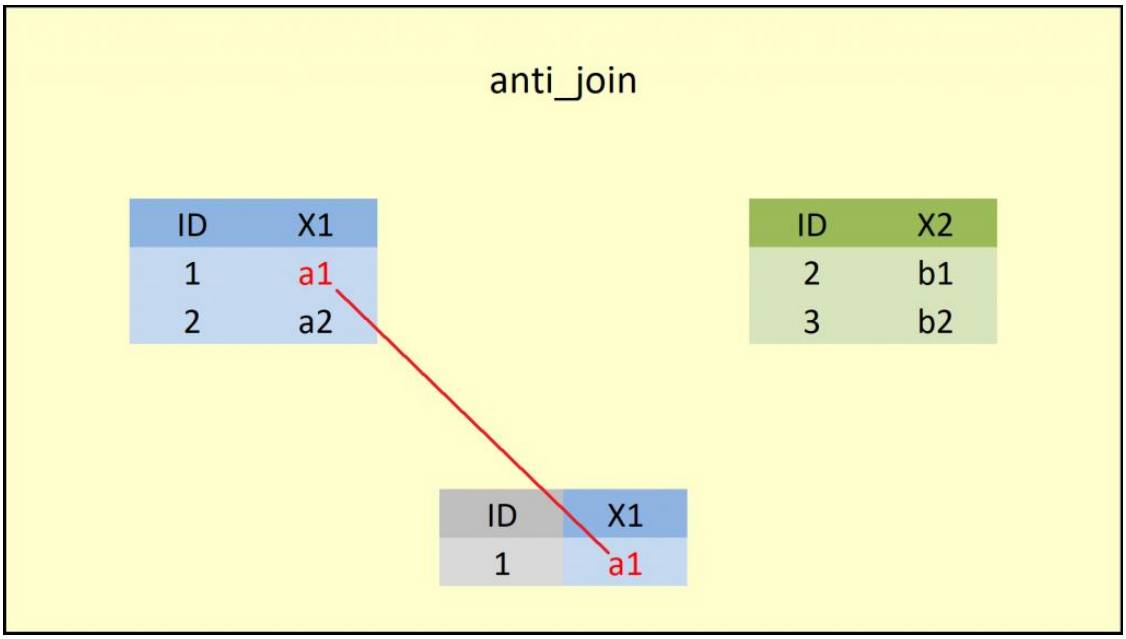
anti_join

- Drops all observations in *data1* that have a match in *data2*

```
anti_join(data1, data2)

## Joining with `by = join_by(ID)`

##   ID X1
## 1  1 a1
```


anti_join
ID
X1
1
a1
2
a2
ID
X2
2
b1
3
b2
ID
X1
1
a1