Data Wrangling in R

Data manipulation

dplyr review

`filter()` (and `slice()`)	select rows
`arrange()`	order rows
`select()` (and `rename()`)	order/rename columns
`distinct()`	find distinct rows
`mutate()` (and `transmute()`)	add new variables
`summarise()`	summarize a data set
`sample_n()` and `sample_frac()`	sample from a data set

Gapminder example

```
# install.packages("gapminder")
library(dplyr)
library(gapminder)
str(gapminder)
```

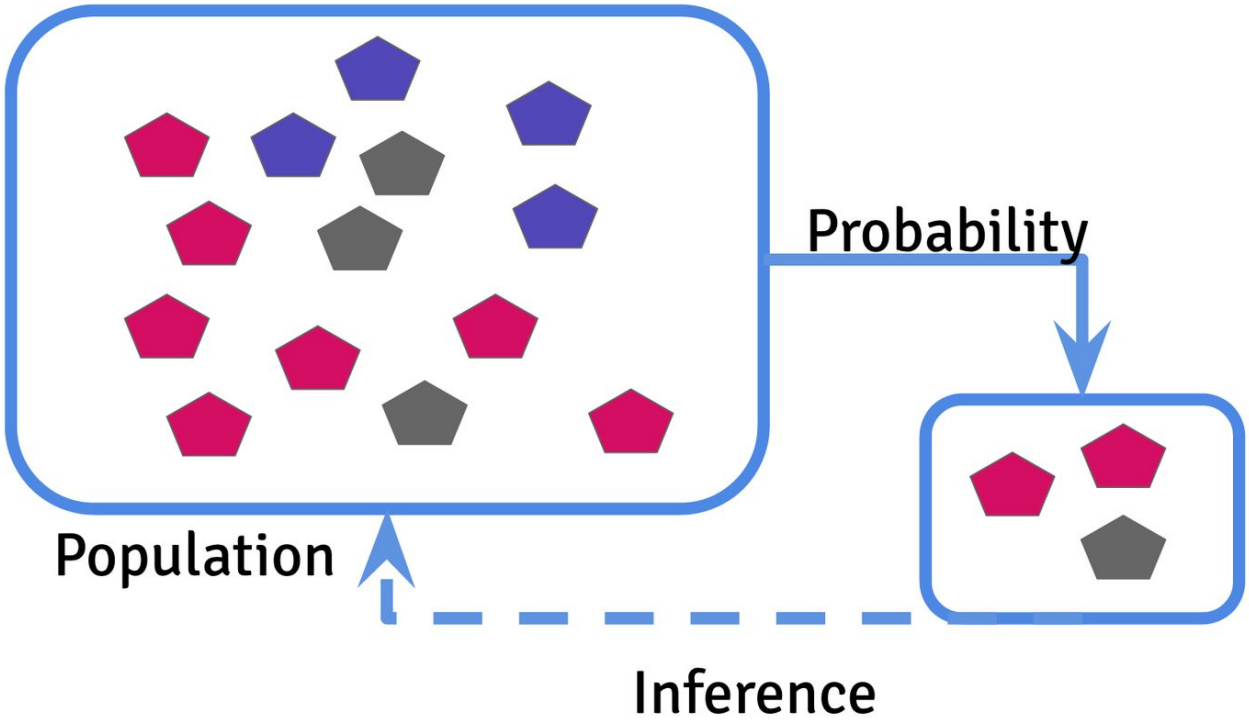
dplyr

```
glimpse(gtbl)
filter(gtbl, lifeExp < 29)
filter(gtbl, country == "Rwanda")
select(gtbl,country,pop,continent)
arrange(gtbl,pop)
arrange(gtbl,desc(pop))
gtbl = mutate(gtbl,
   newVar = (lifeExp / gdpPercap))
select(gtbl,lifeExp,gdpPercap,newVar)
distinct(gtbl)
```

Key principle of big data

Key principle of big data

Central dogma of statistics



Sampling

```
sample_n(gtbl,3)
sample_frac(gtbl,0.5)
```

Pipes

%>%

(Read as “then”)

http://cran.r-project.org/web/packages/magrittr/index.html

Piping stuff

```
gtbl %>% head
gtbl %>% head(3)
```

Uh…big deal?

Example

Show me a random sample of the data for Asian countries with life expectancy < 65.

In base R

```
 tbl1 = gtbl[gtbl$continent=="Asia",]
gtbl2 = gtbl1[gtbl1$lifeExp < 65,]
gtbl3 =
gtbl2[sample(1:dim(gtbl2)[1],size=10),]
gtbl3
```

With pipes + dplyr

```
gtbl %>% filter(continent == "Asia") %>%
         filter(lifeExp < 65) %>%
         sample_n(10)
```

Example 2

What is the average life expectancy by continent?

group_by() and summarize()

```
gtbl %>% group_by(continent) %>%
      summarize(aveLife = mean(lifeExp))
```

Example 3

What is the average life expectancy by continent and what was the sample size for each continent?

Gapminder example

```
gtbl %>% group_by(continent) %>%
    summarize(aveLife = mean(lifeExp),n=n())
```

Common summarization options

`mean`	`mean within groups`
`sum`	`sum within groups`
`sd`	`standard deviation within groups`
`max`	`max within groups`
`...`	`...`
`n()`	`number in each group`
`first()`	`first in group`
`last()`	`last in group`
`nth(n=3)`	`nth in group (3rd here)`

Example 4

Create a variable that multiplies life expectancy by 1.2 for the Asian countries, and 1.3 for the African countries

case_when()

```
gtbl2 = gtbl %>% mutate(newlifeExp =
case_when(
  continent == "Africa" ~ lifeExp*1.3,
  continent == "Asia" ~ lifeExp*1.2,
  TRUE ~ lifeExp
))
```

case_when() - results

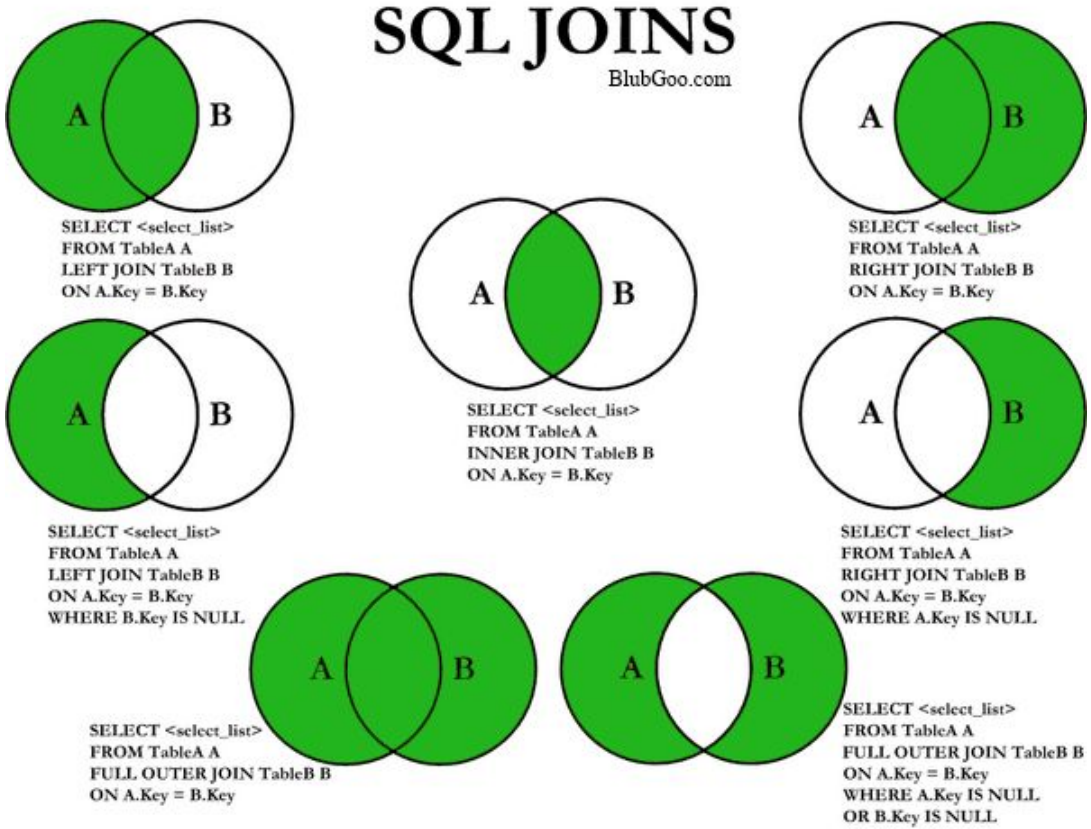
```
library(ggplot2)

ggplot(gtbl2,aes(x=lifeExp,y=newlifeExp,color=continent)) + geom_point()
```

dplyr lab

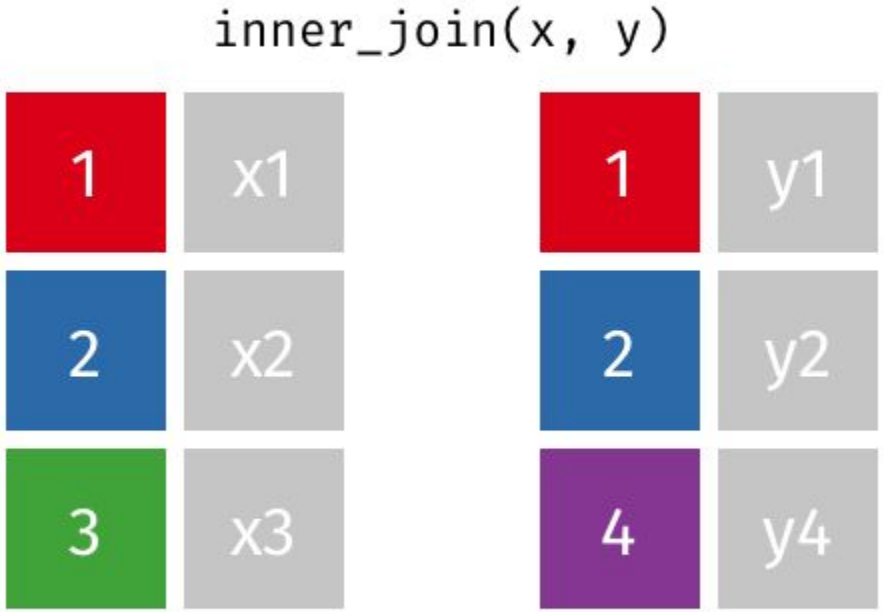
https://bit.ly/1TiSxqp

Merging data sets

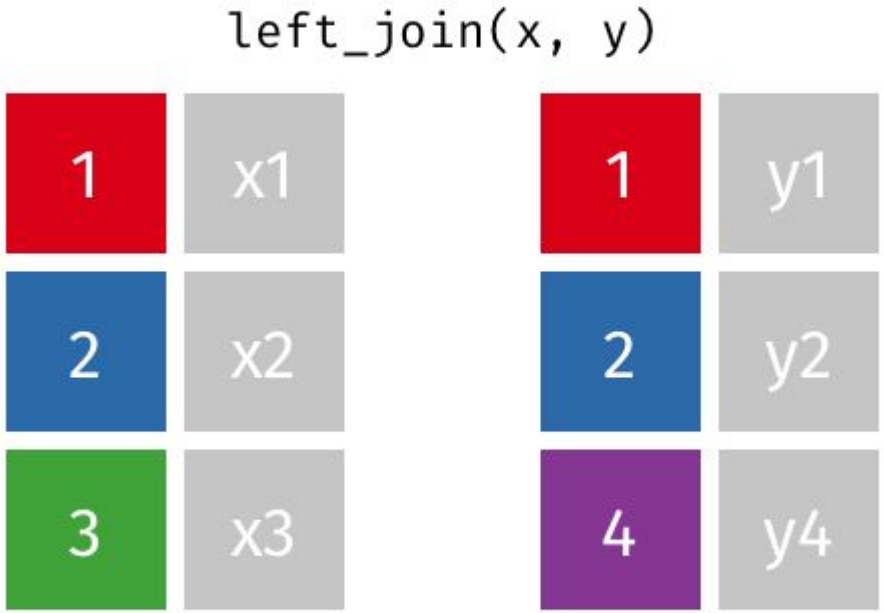
The worst and most common task



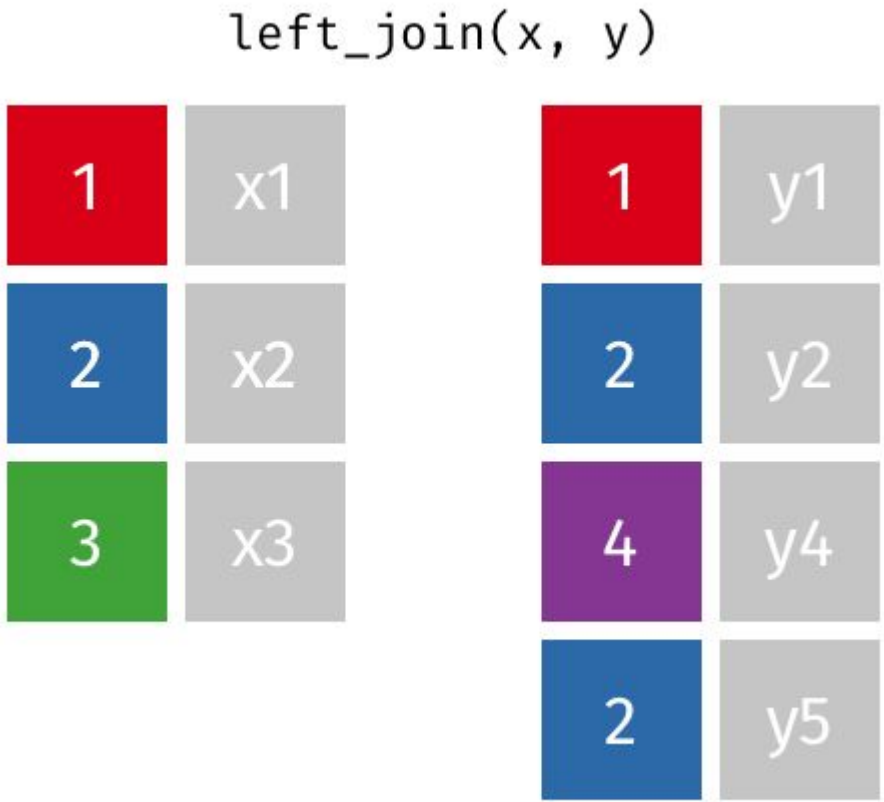
https://www.blubgoo.com/sql-join-overview/



https://github.com/gadenbuie/tidyexplain

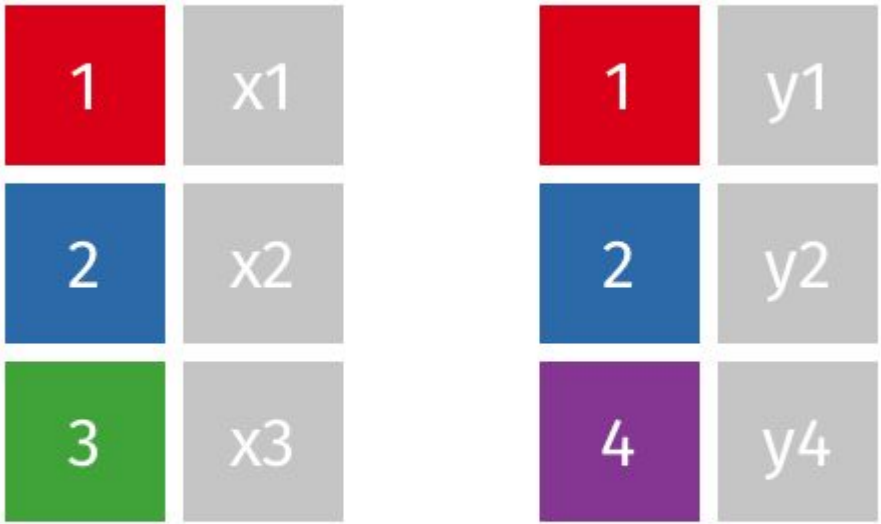


https://github.com/gadenbuie/tidyexplain

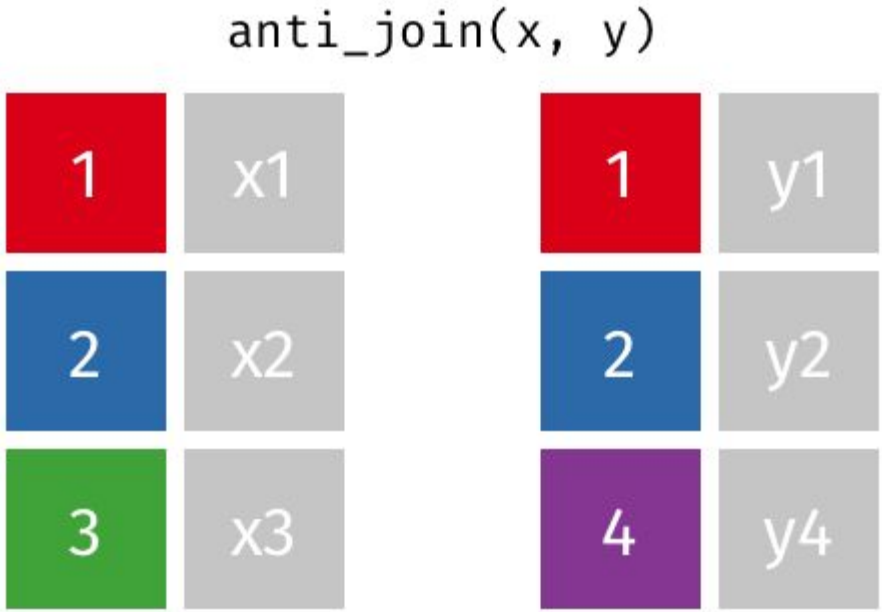


https://github.com/gadenbuie/tidyexplain

full_join(x, y)



https://github.com/gadenbuie/tidyexplain



https://github.com/gadenbuie/tidyexplain

Superhero example

```
superheroes <-
  c("    name, alignment, gender,         publisher",
    " Magneto,       bad,   male,            Marvel",
    "   Storm,      good, female,            Marvel",
    "Mystique,       bad, female,            Marvel",
    "  Batman,      good,   male,                DC",
    "   Joker,       bad,   male,                DC",
    "Catwoman,       bad, female,                DC",
    " Hellboy,      good,   male, Dark Horse Comics")

superheroes <- read.csv(text = superheroes, strip.white = TRUE)
```

http://stat545.com/bit001_dplyr-cheatsheet.html

Superhero example

```
publishers <-
  c("publisher, yr_founded",
    "       DC,       1934",
    "   Marvel,       1939",
    "    Image,       1992")
publishers <- read.csv(text = publishers,

   strip.white = TRUE)
```

http://stat545.com/bit001_dplyr-cheatsheet.html

superheroes

name	alignment	gender	publisher
Magneto	bad	male	Marvel
Storm	good	female	Marvel
Mystique	bad	female	Marvel
Batman	good	male	DC
Joker	bad	male	DC
Catwoman	bad	female	DC
Hellboy	good	male	Dark Horse Comics

publishers

publisher	yr_founded
DC	1934
Marvel	1939
Image	1992

inner_join(x = superheroes, y = publishers)

name	alignment	gender	publisher	yr_founded
Magneto	bad	male	Marvel	1939
Storm	good	female	Marvel	1939
Mystique	bad	female	Marvel	1939
Batman	good	male	DC	1934
Joker	bad	male	DC	1934
Catwoman	bad	female	DC	1934

superheroes

name	alignment	gender	publisher
Magneto	bad	male	Marvel
Storm	good	female	Marvel
Mystique	bad	female	Marvel
Batman	good	male	DC
Joker	bad	male	DC
Catwoman	bad	female	DC
Hellboy	good	male	Dark Horse Comics

publishers

publisher	yr_founded
DC	1934
Marvel	1939
Image	1992

left_join(x = superheroes, y = publishers)

name	alignment	gender	publisher	yr_founded
Magneto	bad	male	Marvel	1939
Storm	good	female	Marvel	1939
Mystique	bad	female	Marvel	1939
Batman	good	male	DC	1934
Joker	bad	male	DC	1934
Catwoman	bad	female	DC	1934
Hellboy	good	male	Dark Horse Comics	NA

superheroes

name	alignment	gender	publisher
Magneto	bad	male	Marvel
Storm	good	female	Marvel
Mystique	bad	female	Marvel
Batman	good	male	DC
Joker	bad	male	DC
Catwoman	bad	female	DC
Hellboy	good	male	Dark Horse Comics

publishers

publisher	yr_founded
DC	1934
Marvel	1939
Image	1992

full_join(x = superheroes, y = publishers)

name	alignment	gender	publisher	yr_founded
Magneto	bad	male	Marvel	1939
Storm	good	female	Marvel	1939
Mystique	bad	female	Marvel	1939
Batman	good	male	DC	1934
Joker	bad	male	DC	1934
Catwoman	bad	female	DC	1934
Hellboy	good	male	Dark Horse Comics	NA
NA	NA	NA	Image	1992

Merging lab

https://bit.ly/1S5WQ5T

“flat files” vs. databases

Flat files
- .csv, .xlsx, .txt,...

How used
- Usually read into RAM

Good for
- Small/medium sized data

Disadvantage
- Big data = big computer
- Slow reading

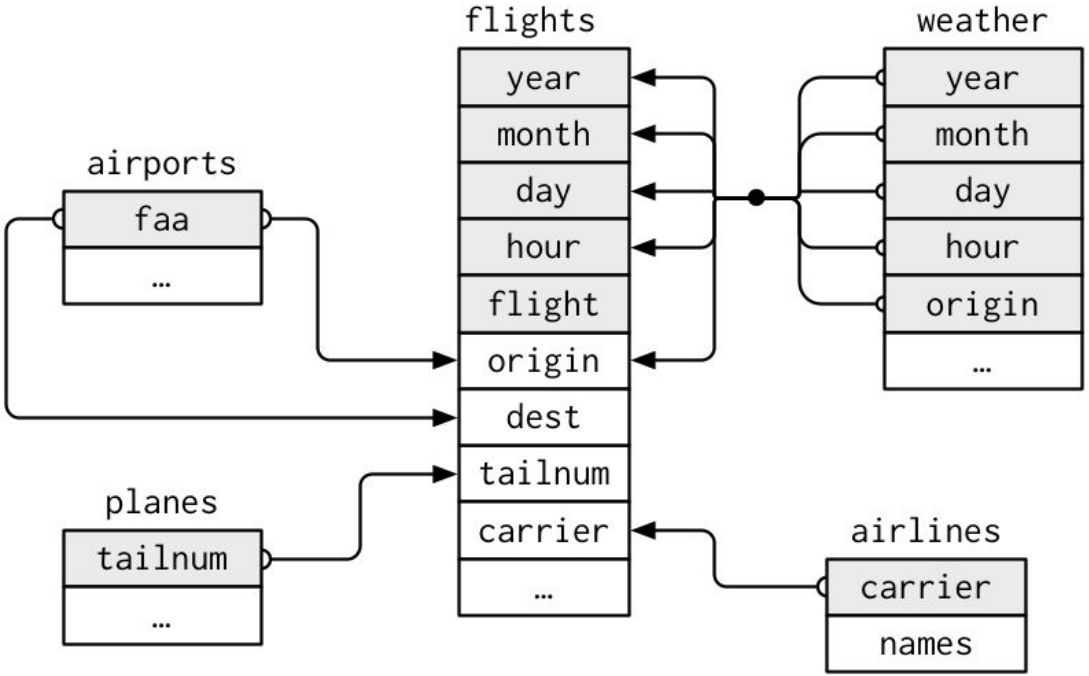
Databases
- sqllite, postgres, mongodb

How used
- Data stays on disk

Good for
- Big data

Disadvantage
- Low interactivity
- Potentially slow I/O

Database



http://r4ds.had.co.nz/relational-data.html

Getting set up

```
install.packages("dplyr")
install.packages("babynames")
install.packages("pryr")
install.packages("RSQLite")
library(dplyr)
library(babynames)
library(pryr)
library(RSQLite)
```

Checking out babynames

```
?babynames
View(babynames)
str(babynames)
object_size(babynames)
```

Getting set up

```
my_db <- src_sqlite("my_db.sqlite3",
  create = TRUE)
babys_sqlite <- copy_to(my_db,
  babynames, temporary = FALSE)
src_tbls(my_db)
tbl(my_db,"babynames")
```

Now you can use dplyr just like before

```
newtbl = my_db %>%
        tbl("babynames")%>%
        filter(name=="Hilary") %>%
        select(year,n,name)
newtbl
```

```
newtbl = my_db %>%
        tbl("babynames")%>%
        filter(name=="Hilary") %>%
        select(year,n,name)
newtbl
```

Dplyr waits to access the database
Here it only gets the first 10 rows

```
newtbl = my_db %>%
        tbl("babynames")%>%
        filter(name=="Hilary") %>%
        select(year,n,name)
output = newtbl %>% collect()
```

collect() asks for the whole result

```
popular = babynames %>%
    group_by(name) %>%
  summarise(N = sum(n)) %>%
arrange(desc(N)) %>% top_n(100)
```

```
popular = my_db %>%
            tbl("babynames")%>%
            group_by(name) %>%
           summarise(N = sum(n)) %>%
           arrange(desc(N)) %>%
           top_n(100)
```

```
popular = my_db %>%
  tbl("babynames")%>%
   group_by(name) %>%
  summarise(N = sum(n)) %>%
arrange(desc(N)) %>% top_n(100)
```

Not supported for this database

```
popular = my_db %>%
          tbl("babynames")%>%
           group_by(name) %>%
         summarise(N = sum(n)) %>%
         arrange(desc(N)) %>%
         collect() %>%
         top_n(100)
```

- basic math operators: `+, -, *, /, %%, ^`
- math functions: `abs, acos, acosh, asin, asinh, atan, atan2, atanh, ceiling, cos, cosh, cot, coth, exp, floor, log,log10, round, sign, sin, sinh, sqrt, tan, tanh`
- logical comparisons: `<, <=, !=, >=, >, ==, %in%`
- boolean operations: `&, &&, |, ||, !, xor`
- basic aggregations: `mean, sum, min, max, sd, var`

https://cran.r-project.org/web/packages/dplyr/vignettes/databases.html

```
translate_sql(filter(name=="James"))
translate_sql(mean(x))
```

https://cran.r-project.org/web/packages/dplyr/vignettes/databases.html

```
how_female = my_db %>%
  tbl("babynames") %>%
  group_by(name) %>%
  summarize(m=mean(sex=="F"))

explain(how_female)
```

Databases lab

https://bit.ly/2JzvJ97

data.table

data.table

- Often the fastest way to process “medium” data
- Basically like using data frames
- Unfortunately has a funky syntax :(
- But basically similar idea - processing on data frames

DATA ANALYSIS THE DATA.TABLE WAY

The official Cheat Sheet for the DataCamp course

General form: `DT[i, j, by]` →  → "Take DT, subset rows using `i`, then calculate `j` grouped by `by`"

CREATE A DATA TABLE

Create a data.table and call it **DT**.

```
library(data.table)
set.seed(45L)
DT <- data.table(V1=c(1L,2L),
                 V2=LETTERS[1:3],
                 V3=round(rnorm(4),4),
                 V4=1:12)
```

```
> DT
    V1 V2      V3 V4
 1:  1  A -1.1727  1
 2:  2  B -0.3825  2
 3:  1  C -1.0604  3
 4:  2  A  0.6651  4
 5:  1  B -1.1727  5
 6:  2  C -0.3825  6
 7:  1  A -1.0604  7
 8:  2  B  0.6651  8
 9:  1  C -1.1727  9
10:  2  A -0.3825 10
11:  1  B -1.0604 11
12:  2  C  0.6651 12
```

https://s3.amazonaws.com/assets.datacamp.com/img/blog/data+table+cheat+sheet.pdf

```
DT[i,j,by]
```

→ Subset by i, apply the function j, in groups defined by by

Set up

```
install.packages("data.table")
install.packages("readr")
library(data.table)
library(readr)
library(babynames)
```

Reads fast

```
write_csv(babynames,
   file='babynames.csv')
# base R
system.time(read.csv('babynames.csv'))
# tidyverse
system.time(read_csv('babynames.csv'))
# data.table
system.time(fread('babynames.csv'))
```

Using DT objects

```
baby_dt = fread('babynames.csv')
class(baby_dt)
female = baby_dt[sex=="F"]
dim(female)
baby_dt[sex=="F",.(n,name,prop)]
```

Using DT objects

```
baby_dt[sex=="F",.(name,mean(prop))]
baby_dt[sex=="F",
   .(name,mean(prop)),name]
baby_dt[sex=="F",
   .(name,aveprop=mean(prop)),name]
```

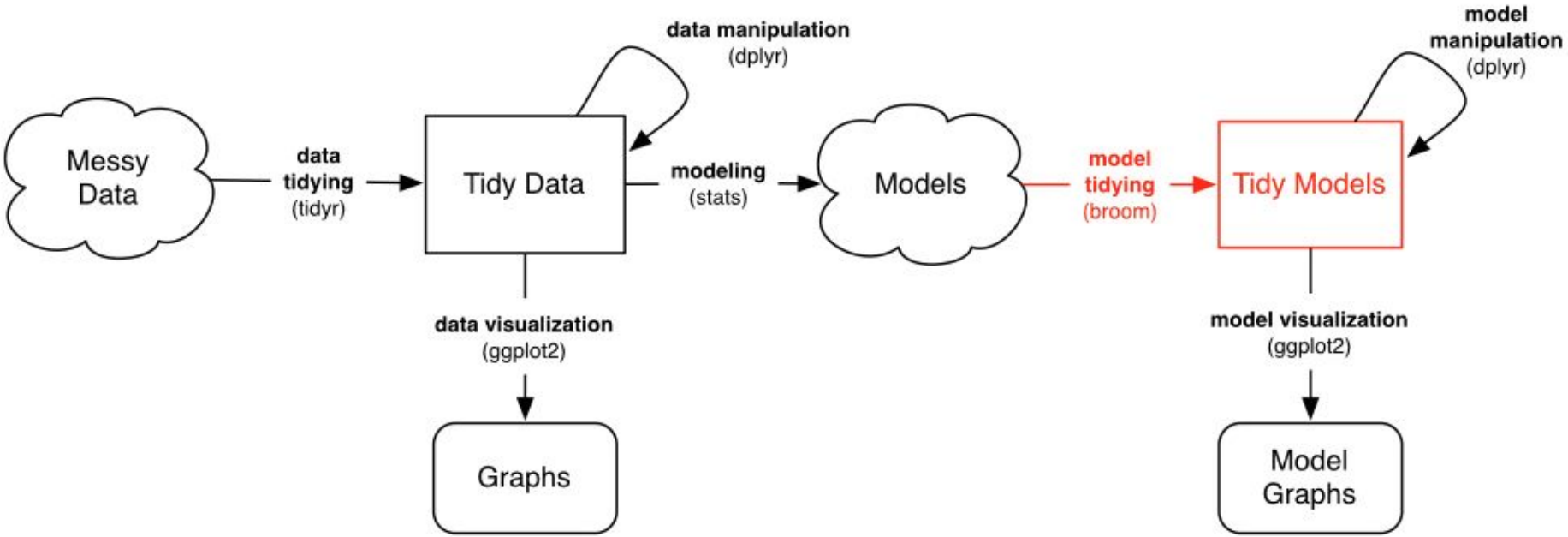
Using DT objects

```
baby_dt[sex=="F",.(name,mean(prop))]
baby_dt[sex=="F",
   .(name,mean(prop)),name]
baby_dt[sex=="F",
   .(name,aveprop=mean(prop)),name]
baby_dt[,aveprop:=mean(prop),name]
```

data.table lab

https://bit.ly/2XJhpE0

broom



http://varianceexplained.org/r/broom-slides/

What's "messy" about a linear regression?

```
> summary(lmfit)

Call:
lm(formula = mpg ~ wt + qsec, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-4.3962 -2.1431 -0.2129  1.4915  5.7486

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  19.7462     5.2521   3.760 0.000765 ***
wt           -5.0480     0.4840 -10.430 2.52e-11 ***
qsec          0.9292     0.2650   3.506 0.001500 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.596 on 29 degrees of freedom
Multiple R-squared:  0.8264,  Adjusted R-squared:  0.8144
F-statistic: 69.03 on 2 and 29 DF,  p-value: 9.395e-12
```

1. Extracting coefficients takes multiple steps:

```
data.frame(coef(summary(lmfit)))
```

2. Information stored in row names (can't combine models)

3. Column names are inconvenient and inconsistent

4. Information is computed in `print` method, not stored

broom's `tidy()` method does the work for you

One function to call

```
> tidy(lmfit)
         term estimate std.error statistic  p.value
1 (Intercept)   19.746     5.252      3.76 7.65e-04
2          wt   -5.048     0.484    -10.43 2.52e-11
3        qsec    0.929     0.265      3.51 1.50e-03
```

Convenient column names

Information stored in columns, never row names

```
install.packages("broom")
library(broom)

data(mtcars)
plot(mtcars$mpg,mtcars$hp)
lm1 = lm(mpg ~ hp, data=mtcars)

tidy(lm1)
augment(lm1)
glance(lm1)
```

```
n <- nls(mpg ~ k * e ^ wt,
   data = mtcars,
   start = list(k = 1, e = 2))
summary(n)
tidy(n)
augment(n)
glance(n)
```

package	class	tidy	augment	glance
base	`data.frame`, `matrix`	✓		✓
	table	✓		
stats	anova, aov, aovlist, density, ftable, manova, pairwise.htest, power.htest, spec, ts, TukeyHSD	✓		
	kmeans, lm, glm, nls	✓	✓	✓
	smooth.spline		✓	✓
	Arima, htest, summaryDefault	✓		✓
biglm	biglm, bigglm	✓		✓
glmnet	cv.glmnet, glmnet	✓		✓
lfe	felm	✓	✓	✓
lmtest	coeftest	✓	✓	✓
lme4	mer, merMod	✓	✓	✓
maps	map	✓		
MASS	ridgelm	✓		✓
multcomp	cld, confint.glht, glht, summary.glht	✓		
plm	plm	✓	✓	✓
sp	Line, Lines, Polygon, Polygons, SpatialLinesDataFrame, SpatialPolygons, SpatialPolygonsDataFrame	✓		
survival	aareg, cch, pyears, survexp, survfit	✓		✓
	coxph, survreg	✓	✓	✓
zoo	zoo	✓		