SOCIAL DATA SCIENCE

Data Manipulation

Sebastian Barfort

August 03, 2016

University of Copenhagen
Department of Economics

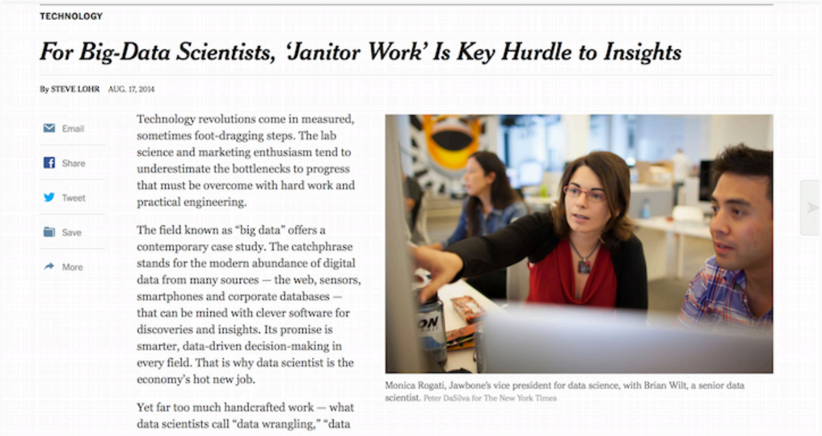
- dplyr
- tidyr
- purrr
- tidytext
- stringr

INTRO

"Herein lies the dirty secret about most data scientists' work – it's more data munging than deep learning. The best minds of my generation are deleting commas from log files, and that makes me sad. A Ph.D. is a terrible thing to waste."

Source

DATA JANITOR



TECHNOLOGY

For Big-Data Scientists, 'Janitor Work' Is Key Hurdle to Insights

By STEVE LOHR AUG. 17, 2014

Technology revolutions come in measured, sometimes foot-dragging steps. The lab science and marketing enthusiasm tend to underestimate the bottlenecks to progress that must be overcome with hard work and practical engineering.

The field known as "big data" offers a contemporary case study. The catchphrase stands for the modern abundance of digital data from many sources — the web, sensors, smartphones and corporate databases — that can be mined with clever software for discoveries and insights. Its promise is smarter, data-driven decision-making in every field. That is why data scientist is the economy's hot new job.

Yet far too much handcrafted work — what data scientists call "data wrangling," "data

Monica Rogati, Jawbone's vice president for data science, with Brian Wilt, a senior data scientist. Peter DaSilva for The New York Times

Source

RAW VERSUS PROCESSED DATA

Raw data

The original source of the data

Often hard to use directly for data analysis

You should *never* process your original data

Processed data

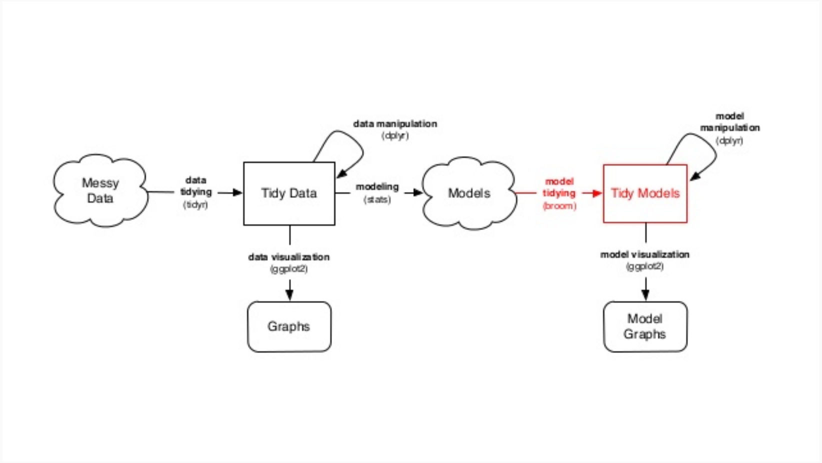
Data that is ready for analysis

Data manipulation involves going from *raw* to *processed* data.

This can include merging, subsetting, transforming, etc.

All steps that take you from raw to processed data should be scripted

WORKFLOW



Messy Data
data tidying (tidyr)
Tidy Data
data manipulation (dplyr)
modeling (stats)
Models
model tidying (broom)
Tidy Models
model manipulation (dplyr)
data visualization (ggplot2)
Graphs
model visualization (ggplot2)
Model Graphs

Tidy data

`tidyr`

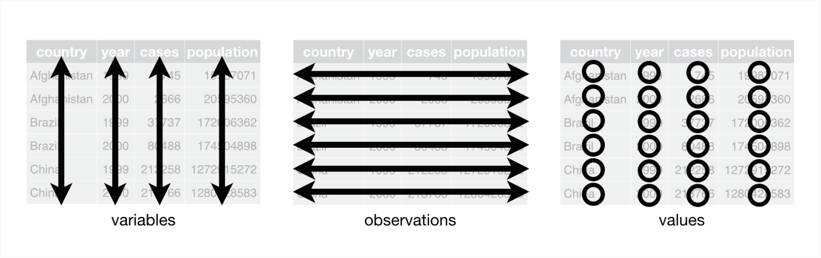
Happy families are all alike; every unhappy family is unhappy in its own way

Leo Tolstoy

Goal of `tidyr`: take your messy data and turn it into a tidy format

tidy data: observations are in the rows, variables are in the columns

TIDY DATA


country
year
cases
population
variables
observations
values

TIDY DATA

```
library("readr")
library("dplyr")
library("tidyr")

df = read_csv("https://raw.githubusercontent.com/hadley/
head(df, 3)
```

```
## Source: local data frame [3 x 11]
##
##   religion <$10k $10-20k $20-30k $30-40k $40-50k $50-
##      <chr> <int>   <int>   <int>   <int>   <int>   <i
## 1 Agnostic    27      34      60      81      76
## 2  Atheist    12      27      37      52      35
## 3 Buddhist    27      21      30      34      33
## Variables not shown: $100-150k <int>, >150k <int>, Do
##   <int>.
```

THE gather FUNCTION

Objective: Reshaping wide format to long format

To tidy this data, we need to **gather** the non-variable columns into a two-column key-value pair

```
args(gather)
```

```
## function (data, key, value, ..., na.rm = FALSE, conve
##     factor_key = FALSE)
## NULL
```

Arguments:

- `data`: data frame
- `key`: column name representing new variable
- `value`: column name representing variable values
- `...`: names of columns to gather (or not gather)

gather AT WORK

```
df %>% gather(income, frequency, -religion)
```

```
## Source: local data frame [180 x 3]
##
##                  religion income frequency
##                     <chr>  <chr>     <int>
## 1                Agnostic  <$10k        27
## 2                 Atheist  <$10k        12
## 3                Buddhist  <$10k        27
## 4                Catholic  <$10k       418
## 5      Don’t know/refused  <$10k        15
## 6        Evangelical Prot  <$10k       575
## 7                   Hindu  <$10k         1
## 8  Historically Black Prot  <$10k       228
## 9        Jehovah’s Witness  <$10k        20
## 10                 Jewish  <$10k        19
## ..                    ...    ...       ...
```

ALTERNATIVES

This

```
df %>% gather(income, frequency, 2:11)
```

returns the same as

```
df %>% gather(income, frequency, -religion)
```

MORE COMPLICATED EXAMPLE

Billboard data

```r
df = read_csv("https://raw.githubusercontent.com/hadley/
head(df, 3)
```

```
## Source: local data frame [3 x 81]
##
##    year       artist                     track  time da
##   <int>        <chr>                     <chr> <chr>
## 1  2000        2 Pac Baby Don't Cry (Keep...  4:22
## 2  2000      2Ge+her The Hardest Part Of ...  3:15
## 3  2000 3 Doors Down               Kryptonite  3:53
## Variables not shown: wk2 <int>, wk3 <int>, wk4 <int>,
##   <int>, wk7 <int>, wk8 <int>, wk9 <int>, wk10 <int>,
##   <int>, wk13 <int>, wk14 <int>, wk15 <int>, wk16 <in
##   <int>, wk19 <int>, wk20 <int>, wk21 <int>, wk22 <in
##   <int>, wk25 <int>, wk26 <int>, wk27 <int>, wk28 <in
##   <int>, wk31 <int>, wk32 <int>, wk33 <int>, wk34 <in
```

TIDYING THE BILLBOARD DATA

To tidy this dataset, we first gather together all the `wk` columns. The column names give the week and the values are the ranks:

```
billboard2 = df %>%
  gather(week, rank, wk1:wk76,na.rm = TRUE)
head(billboard2, 3)
```

```
## Source: local data frame [3 x 7]
##
##    year       artist                  track  time da
##   <int>        <chr>                  <chr> <chr>
## 1  2000        2 Pac Baby Don't Cry (Keep...  4:22
## 2  2000      2Ge+her The Hardest Part Of ...  3:15
## 3  2000 3 Doors Down              Kryptonite  3:53
## Variables not shown: rank <chr>.
```

What more would we want to do to the data?

DATA CLEANING

Let’s turn the week into a numeric variable and create a proper date column

```
billboard3 = billboard2 %>%
  mutate(
    week = extract_numeric(week),
    date = as.Date(date.entered) + 7 * (week - 1)) %>%
  select(-date.entered) %>%
  arrange(artist, track, week)
head(billboard3, 3)
```

```
## Source: local data frame [3 x 7]
##
##    year artist                          track  time  week  r
##   <int>  <chr>                          <chr> <chr> <dbl> <c
## 1  2000  2 Pac Baby Don't Cry (Keep...  4:22     1
## 2  2000  2 Pac Baby Don't Cry (Keep...  4:22     2
## 3  2000  2 Pac Baby Don't Cry (Keep...  4:22     3
```

EVEN MORE COMPLICATED EXAMPLE

After gathering columns, the key column is sometimes a combination of multiple underlying variable names.

```
df = read_csv("https://raw.githubusercontent.com/hadley/
head(df, 3)
```

```
## Source: local data frame [3 x 22]
##
##    iso2  year   m04  m514  m014 m1524 m2534 m3544 m45
##   <chr> <int> <int> <int> <int> <int> <int> <int> <in
## 1    AD  1989    NA    NA    NA    NA    NA    NA
## 2    AD  1990    NA    NA    NA    NA    NA    NA
## 3    AD  1991    NA    NA    NA    NA    NA    NA
## Variables not shown: f04 <int>, f514 <int>, f014 <int
##   <int>, f3544 <int>, f4554 <int>, f5564 <int>, f65 <
```

Question: what are the variables here?

ANSWER

The dataset comes from the World Health Organisation, and records the counts of confirmed tuberculosis cases by country, year, and demographic group. The demographic groups are broken down by sex (m, f) and age (0-14, 15-25, 25-34, 35-44, 45-54, 55-64, unknown).

GATHERING THE NON-VARIABLE COLUMNS

```
tb2 = df %>%
  gather(demo, n, -iso2, -year, na.rm = TRUE)
head(tb2, 3)
```

```
## Source: local data frame [3 x 4]
##
##    iso2  year  demo     n
##   <chr> <int> <chr> <int>
## 1    AD  2005   m04     0
## 2    AD  2006   m04     0
## 3    AD  2008   m04     0
```

Is this dataset tidy?

SEPARATING THE demo VARIABLE

`separate` makes it easy to split a compound variables into individual variables. You can either pass it a regular expression to split on or a vector of character positions. In this case we want to split after the first character.

```
tb3 = tb2 %>%
  separate(demo, c("sex", "age"), 1)
head(tb3, 3)
```

```
## Source: local data frame [3 x 5]
##
##    iso2  year   sex   age     n
##   <chr> <int> <chr> <chr> <int>
## 1    AD  2005     m    04     0
## 2    AD  2006     m    04     0
## 3    AD  2008     m    04     0
```

Question: Compare `tb3` to the original data frame (`df`). What are

RESHAPING FROM LONG TO WIDE FORMAT

There are times when we are required to turn long formatted data into wide formatted data. The `spread` function spreads a key-value pair across multiple columns.

```
args(spread)
```

```
## function (data, key, value, fill = NA, convert = FALS
##     sep = NULL)
## NULL
```

- `data`: data frame
- `key`: column values to convert to multiple columns
- `value`: single column values to convert to multiple columns' values
- `fill`: If there isn't a value for every combination of the other variables and the key column, this value will be substituted

spread IN ACTION

```
tb3.wide = tb3 %>% spread(age, n)
tb3.wide
```

```
## Source: local data frame [4,885 x 13]
##
##      iso2  year   sex   014    04  1524  2534  3544  4
##     <chr> <int> <chr> <int> <int> <int> <int> <int> <i
## 1      AD  1996     f     0    NA     1     1     0
## 2      AD  1996     m     0    NA     0     0     4
## 3      AD  1997     f     0    NA     1     2     3
## 4      AD  1997     m     0    NA     0     1     2
## 5      AD  1998     m     0    NA     0     0     1
## 6      AD  1999     f     0    NA     0     0     1
## 7      AD  1999     m     0    NA     0     0     1
## 8      AD  2000     m     0    NA     0     1     0
## 9      AD  2001     m     0    NA    NA    NA     2
## 10     AD  2002     f     0    NA     1     0     0
```

DATA FOR TODAY

In this part of the lecture we will work with the Danish federal budget proposal for 2016

```
library("readr")
df = read_csv("https://raw.githubusercontent.com/sebasti
```

Some nice guy has already cleaned this data for you

OVERVIEW OF YOUR DATA

Useful functions:

- `str`: displays the structure of your data frame
- `head`: displays the first rows
- `summary`: gives summary statistics
- `glimpse` (from the `dplyr` package): modern alternative to `str`

str

```r
str(df)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    8430 obs
##  $ paragraf   : chr  "Dronningen" "Medlemmer af det k
##  $ hovedomrode: chr  "Statsydelse" "Årpenge mv." "Udg
##  $ aktivitet  : chr  "Statsydelse" "Årpenge mv." "Fol
##  $ hovedkonto : chr  "Statsydelse" "Årpenge mv." "Fol
##  $ aar        : int  2014 2014 2014 2014 2014 2014 20
##  $ udgift     : num  77.7 26.4 387.8 264.2 6.5 ...
```

head

```r
head(df)
```

```
## Source: local data frame [6 x 6]
##
##                                    paragraf              h
##                                       <chr>
## 1                                 Dronningen              S
## 2 Medlemmer af det kongelige hus m.fl.                    Å
## 3                               Folketinget Udgifter ved F
## 4                               Folketinget Udgifter ved F
## 5                               Folketinget Udgifter ved F
## 6                               Folketinget Udgifter ved F
## Variables not shown: aktivitet <chr>, hovedkonto <chr
##   <dbl>.
```

summary

```r
summary(df)
```

```
##    paragraf           hovedomrode         aktivitet
##  Length:8430        Length:8430        Length:8430
##  Class :character   Class :character   Class :charact
##  Mode  :character   Mode  :character   Mode  :charact
##
##
##
##   hovedkonto             aar           udgift
##  Length:8430        Min.   :2014   Min.   :-306424.10
##  Class :character   1st Qu.:2015   1st Qu.:       0.00
##  Mode  :character   Median :2016   Median :       7.00
##                     Mean   :2016   Mean   :     223.18
##                     3rd Qu.:2018   3rd Qu.:      89.95
##                     Max.   :2019   Max.   : 132541.40
```

glimpse

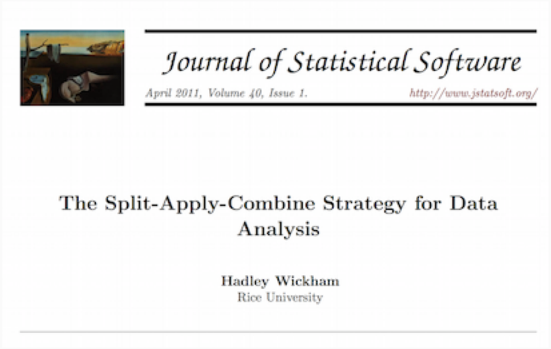
```r
library("dplyr")
glimpse(df)
```

```
## Observations: 8,430
## Variables: 6
## $ paragraf    (chr) "Dronningen", "Medlemmer af det k
## $ hovedomrode (chr) "Statsydelse", "Årpenge mv.", "Ud
## $ aktivitet   (chr) "Statsydelse", "Årpenge mv.", "Fo
## $ hovedkonto  (chr) "Statsydelse", "Årpenge mv.", "Fo
## $ aar         (int) 2014, 2014, 2014, 2014, 2014, 201
## $ udgift      (dbl) 77.7, 26.4, 387.8, 264.2, 6.5, 2.
```

Data Manipulation (`dplyr`)

THE SPLIT-APPLY-COMBINE APPROACH

Many data analysis problems involve the application of a *split-apply-combine strategy*, where you break up a big problem into manageable pieces, opereate on each piece independently and then put the pieces back together



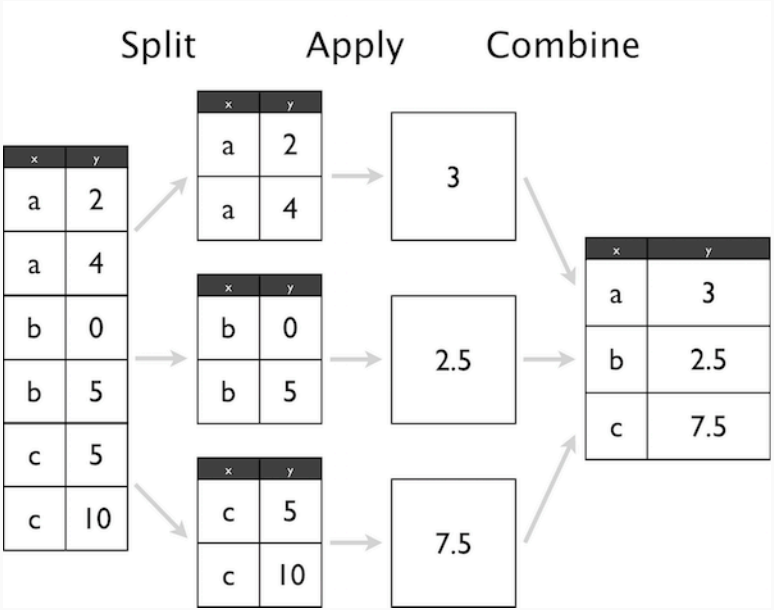
Journal of Statistical Software

April 2011, Volume 40, Issue 1. http://www.jstatsoft.org/

The Split-Apply-Combine Strategy for Data Analysis

Hadley Wickham
Rice University

SPLIT-APPLY-COMBINE



The `dplyr` package

`dplyr`: (efficiently) split-apply-combine for data frames

Verbs a verb is a function that takes a data frame as it's first argument

- `filter`: select rows
- `arrange`: order rows
- `select`: select columns
- `rename`: rename columns
- `distinct`: find distinct rows
- `mutate`: add new variables
- `summarise`: summarize across a data set
- `sample_n`: sample from a data set

THE filter FUNCTION I

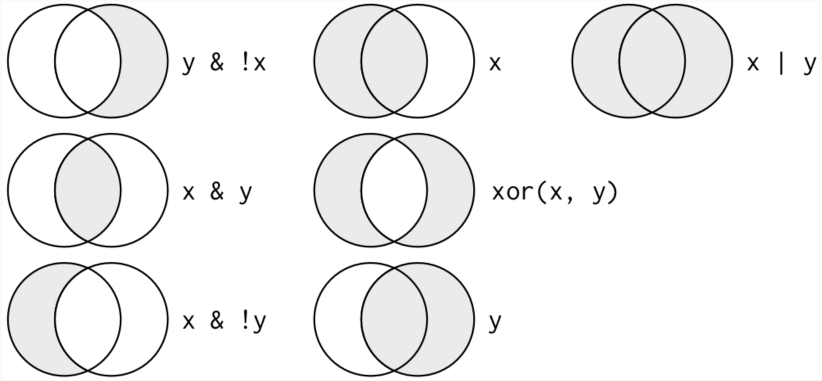
```r
filter(df, udgift == min(udgift))
```

```
## Source: local data frame [1 x 6]
##
##              paragraf                         hovedomrode
##                 <chr>                               <chr>
## 1 Skatter og afgifter Skatter på indkomst og formue P
## Variables not shown: hovedkonto <chr>, aar <int>, udg
```

```r
filter(df, paragraf == "Skatter og afgifter ")
```

```
## Source: local data frame [0 x 6]
##
## Variables not shown: paragraf <chr>, hovedomrode <chr
##   hovedkonto <chr>, aar <int>, udgift <dbl>.
```

LOGICAL OPERATORS



COMBINING CONDITIONS: and

You can easily combine conditions

```
filter(df, paragraf == ”Skatter og afgifter ” & aktivite
```

```
## Source: local data frame [0 x 6]
##
## Variables not shown: paragraf <chr>, hovedomrode <chr
##   hovedkonto <chr>, aar <int>, udgift <dbl>.
```

COMBINING CONDITIONS: or

```
filter(df, paragraf == ”Skatter og afgifter ” | aktivite
```

```
## Source: local data frame [0 x 6]
##
## Variables not shown: paragraf <chr>, hovedomrode <chr
##   hovedkonto <chr>, aar <int>, udgift <dbl>.
```

THE select FUNCTION

```r
select(df, aar, udgift)
```

```
## Source: local data frame [8,430 x 2]
##
##       aar udgift
##     <int>  <dbl>
## 1    2014   77.7
## 2    2014   26.4
## 3    2014  387.8
## 4    2014  264.2
## 5    2014    6.5
## 6    2014    2.2
## 7    2014   63.3
## 8    2014    5.0
## 9    2014  202.5
## 10   2014   76.7
## ..    ...    ...
```

THE arrange FUNCTION

```
arrange(df, hovedomrode, udgift)
```

```
## Source: local data frame [8,430 x 6]
##
##                                               paragraf
##                                                  <chr>
## 1  Min. for Børn, Undervisning og Ligestilling Admini
## 2  Min. for Børn, Undervisning og Ligestilling Admini
## 3  Min. for Børn, Undervisning og Ligestilling Admini
## 4  Min. for Børn, Undervisning og Ligestilling Admini
## 5  Min. for Børn, Undervisning og Ligestilling Admini
## 6  Min. for Børn, Undervisning og Ligestilling Admini
## 7  Min. for Børn, Undervisning og Ligestilling Admini
## 8  Min. for Børn, Undervisning og Ligestilling Admini
## 9  Min. for Børn, Undervisning og Ligestilling Admini
## 10 Min. for Børn, Undervisning og Ligestilling Admini
## ..                                                ...
```

ARRANGE BY NUMERIC VARIABLE

```r
arrange(df, -aar)
```

```
## Source: local data frame [8,430 x 6]
##
##                                  paragraf
##                                     <chr>
## 1                              Dronningen
## 2  Medlemmer af det kongelige hus m.fl.
## 3                             Folketinget           Udg
## 4                             Folketinget           Udg
## 5                             Folketinget           Udg
## 6                             Folketinget           Udg
## 7                             Folketinget            Fo
## 8                             Folketinget Statsrevisore
## 9                             Folketinget Statsrevisore
## 10                            Folketinget             F
## ..                                    ...
```

THE mutate FUNCTION

`mutate` let's you add new variables to your data frame

```r
df.mutated = mutate(df, newVar = udgift/2)
select(df.mutated, newVar, udgift)
```

```
## Source: local data frame [8,430 x 2]
##
##    newVar udgift
##     <dbl>  <dbl>
## 1   38.85   77.7
## 2   13.20   26.4
## 3  193.90  387.8
## 4  132.10  264.2
## 5    3.25    6.5
## 6    1.10    2.2
## 7   31.65   63.3
## 8    2.50    5.0
## 9  101.25  202.5
```

THE sample_n FUNCTION

```r
sample_n(df, 3)
```

```
## Source: local data frame [3 x 6]
##
##                                    paragraf          hov
##                                       <chr>
## 1         Sundheds- og Ældreministeriet             Syge
## 2     Transport- og Bygningsministeriet Trafik og byg
## 3 Uddannelses- og Forskningsministeriet         Fælles
## Variables not shown: aktivitet <chr>, hovedkonto <chr
##   <dbl>.
```

DIGRESSION: MEET THE PIPE

```
magrittr::
%>%
```

THE PIPE OPERATOR

The pipe operator `%>%` (RStudio has keyboard shortcuts, learn to use them!) let's you write sequences instead of nested functions

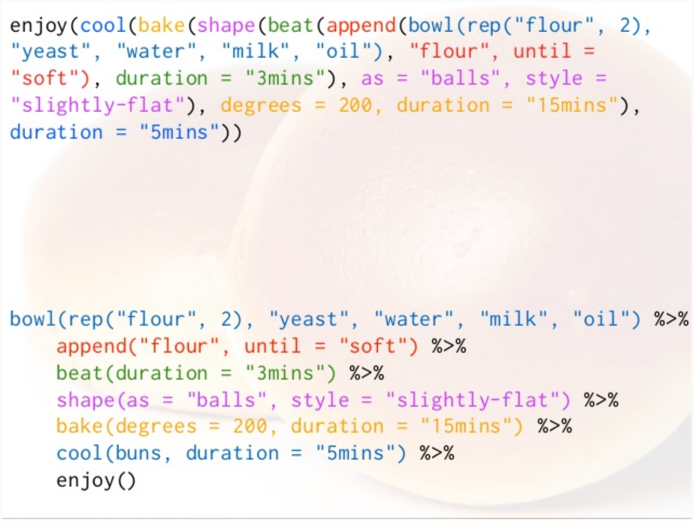
`x %>% f(y)` -> `f(x,y)`

`x %>% f(z, .)` -> `f(z, x)`

Read `%>%` as "then". First do this, *then* do this, etc...

It's implemented in R by a Danish econometrician

INTUITION



```
enjoy(cool(bake(shape(beat(append(bowl(rep("flour", 2),
"yeast", "water", "milk", "oil"), "flour", until =
"soft"), duration = "3mins"), as = "balls", style =
"slightly-flat"), degrees = 200, duration = "15mins"),
duration = "5mins"))
```

```
bowl(rep("flour", 2), "yeast", "water", "milk", "oil") %>%
    append("flour", until = "soft") %>%
    beat(duration = "3mins") %>%
    shape(as = "balls", style = "slightly-flat") %>%
    bake(degrees = 200, duration = "15mins") %>%
    cool(buns, duration = "5mins") %>%
    enjoy()
```

source

COMBINING dplyr AND THE PIPE

dplyr is designed to work with the pipe.

So

```
df %>%
  select(aar, udgift) %>%
  filter(aar == 2014)
```

returns the same as

```
filter(select(df, aar, udgift), aar == 2014)
```

EXAMPLE

Show me a random sample of the data from 2014, where `paragraf == Folketinget` and `udgift` is above the mean.

```
df.1 = filter(df, aar == 2014 & paragraf == "Folketinget
df.2 = filter(df.1, udgift > mean(udgift, na.rm = TRUE))
df.3 = sample_n(df.2, 3)
df.3
```

```
## Source: local data frame [3 x 6]
##
##      paragraf                         hovedomrode
##         <chr>                               <chr>
## 1 Folketinget           Udgifter ved Folketinget
## 2 Folketinget Statsrevisorerne og Rigsrevisionen
## 3 Folketinget           Udgifter ved Folketinget
## Variables not shown: aktivitet <chr>, hovedkonto <chr
##   <dbl>.
```

WITH THE PIPE

```
df %>%
  filter(aar == 2014 & paragraf == "Folketinget") %>%
  filter(udgift > mean(udgift, na.rm = TRUE)) %>%
  sample_n(3)
```

```
## Source: local data frame [3 x 6]
##
##      paragraf                               hovedomrode
##         <chr>                                     <chr>
## 1 Folketinget                Udgifter ved Folketinget
## 2 Folketinget Statsrevisorerne og Rigsrevisionen
## 3 Folketinget                Udgifter ved Folketinget
## Variables not shown: aktivitet <chr>, hovedkonto <chr
##   <dbl>.
```

Note how readable the code is. Almost like a grammer of data manipulation?

GROUPED OPERATIONS

So far, we have primarily learned how to manipulate data frames.

The `dplyr` package becomes really powerful when we introduce the `group_by` function

`group_by` breaks down a dataset into specified groups of rows. When you then apply the verbs above on the resulting object they'll be automatically applied "by group".

Use in conjunction with `mutate` (to add existing rows to your data frame) or `summarise` (to create a new data frame)

COMMON `mutate`/`summarise` 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)

group_by IN ACTION I

Which ministry has the largest expenses?

```
df %>% filter(udgift >= 0) %>% group_by(paragraf) %>%
  summarise(totale.udgifter = sum(udgift, na.rm = TRUE))
  arrange(-totale.udgifter)
```

```
## Source: local data frame [28 x 2]
##
##                                         paragraf totale
##                                            <chr>
## 1                  Beskæftigelsesministeriet
## 2            Social- og Indenrigsministeriet
## 3      Uddannelses- og Forskningsministeriet
## 4  Min. for Børn, Undervisning og Ligestilling
## 5                          Finansministeriet
## 6                            Pensionsvæsenet
## 7                        Forsvarsministeriet
```

group_by IN ACTION II

Add `totale.udgifter` to the existing data frame

```
df %>% filter(udgift >= 0) %>% group_by(paragraf) %>%
  mutate(totale.udgifter = sum(udgift, na.rm = TRUE)) %>
  select(aar, udgift, totale.udgifter)
```

```
## Source: local data frame [7,609 x 4]
## Groups: paragraf [28]
##
##                                  paragraf   aar udgift
##                                     (chr) (int)  (dbl)
## 1                              Dronningen  2014   77.7
## 2  Medlemmer af det kongelige hus m.fl.  2014   26.4
## 3                             Folketinget  2014  387.8
## 4                             Folketinget  2014  264.2
## 5                             Folketinget  2014    6.5
## 6                             Folketinget  2014    2.2
```

group_by IN ACTION III

You can group by several variables

```
df %>% filter(udgift >= 0) %>% group_by(aar, paragraf) %
  summarise(totale.udgifter = sum(udgift, na.rm = TRUE))
  arrange(-totale.udgifter)
```

```
## Source: local data frame [168 x 3]
## Groups: aar [6]
##
##      aar                                  paragraf
##    (int)                                     (chr)
## 1   2014                   Beskæftigelsesministeriet
## 2   2014             Social- og Indenrigsministeriet
## 3   2014               Afdrag på statsgælden (netto)
## 4   2014       Uddannelses- og Forskningsministeriet
## 5   2014 Min. for Børn, Undervisning og Ligestilling
## 6   2014                                      Renter
```

group_by IN ACTION IV

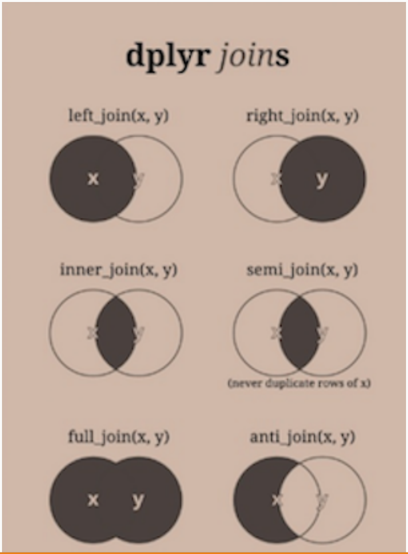
You can group by several variables

```
df %>% filter(udgift >= 0) %>% group_by(paragraf, hovedo
  summarise(totale.udgifter = sum(udgift, na.rm = TRUE))
  arrange(-totale.udgifter)
```

```
## Source: local data frame [120 x 3]
## Groups: paragraf [28]
##
##                               paragraf
##                                  (chr)
## 1  Afdrag på statsgælden (netto)
## 2      Beholdningsbevægelser mv.
## 3      Beholdningsbevægelser mv.
## 4      Beholdningsbevægelser mv.
## 5      Beskæftigelsesministeriet
## 6      Beskæftigelsesministeriet
```

MERGING DATA SETS

Merging two data sets can be tricky and depends on your needs. It’s important to think about what you want before joining.



SUPERHERO EXAMPLE

link

```
superheroes = c("    name, alignment, gender,         pu
    " 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 =
head(superheroes)
```

```
##       name alignment gender publisher
## 1  Magneto       bad   male    Marvel
## 2    Storm      good female    Marvel
## 3 Mystique       bad female    Marvel
```

PUBLISHERS

```
publishers = c("publisher, yr_founded",
    "       DC,        1934",
    "   Marvel,        1939",
    "    Image,        1992")
publishers = read.csv(text = publishers, strip.white = T
head(publishers)
```

```
##   publisher yr_founded
## 1        DC       1934
## 2    Marvel       1939
## 3     Image       1992
```

INNER JOIN

```r
ijsp = inner_join(superheroes, publishers)
ijsp
```

```
##       name alignment gender publisher yr_founded
## 1  Magneto       bad   male    Marvel       1939
## 2    Storm      good female    Marvel       1939
## 3 Mystique       bad female    Marvel       1939
## 4   Batman      good   male        DC       1934
## 5    Joker       bad   male        DC       1934
## 6 Catwoman       bad female        DC       1934
```

LEFT JOIN

```r
ljsp = left_join(superheroes, publishers)
ljsp
```

```
##        name alignment gender         publisher yr_foun
## 1   Magneto       bad   male            Marvel       1
## 2     Storm      good female            Marvel       1
## 3  Mystique       bad female            Marvel       1
## 4    Batman      good   male                DC       1
## 5     Joker       bad   male                DC       1
## 6  Catwoman       bad female                DC       1
## 7   Hellboy      good   male Dark Horse Comics
```

MERGING DIFFERENT NAMES

```
superheroes = mutate(superheroes,
                   seblikes = (publisher=="Marvel"))
publishers = mutate(publishers,
                       seb = (publisher == "Marvel"))
ij2 = inner_join(superheroes,publishers)
ij2
```

```
##        name alignment gender publisher seblikes yr_fou
## 1  Magneto       bad   male    Marvel     TRUE
## 2    Storm      good female    Marvel     TRUE
## 3 Mystique       bad female    Marvel     TRUE
## 4   Batman      good   male        DC    FALSE
## 5    Joker       bad   male        DC    FALSE
## 6 Catwoman       bad female        DC    FALSE
```

MERGING DIFFERENT NAMES

```
ij2 = inner_join(superheroes,publishers,
                    by=c("publisher"="publisher",
                            "seblikes"="seb"))
ij2
```

```
##       name alignment gender publisher seblikes yr_fou
## 1  Magneto       bad   male    Marvel     TRUE
## 2    Storm      good female    Marvel     TRUE
## 3 Mystique       bad female    Marvel     TRUE
## 4   Batman      good   male        DC    FALSE
## 5    Joker       bad   male        DC    FALSE
## 6 Catwoman       bad female        DC    FALSE
```

Iteration