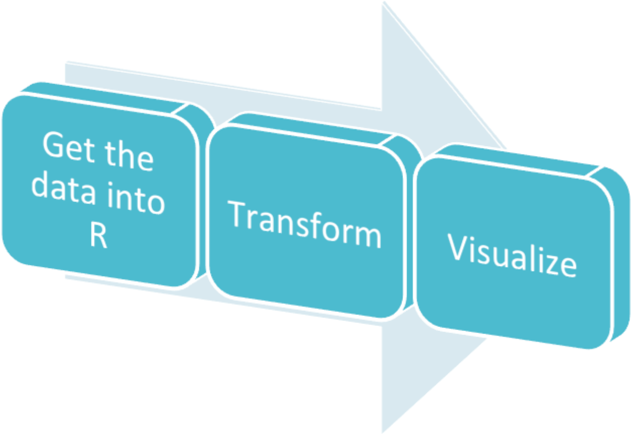
R language and data analysis:data manipulation advanced

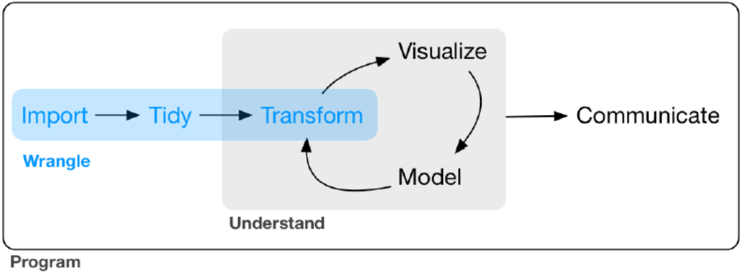
Qiang Shen

Dec. 19, 2017

data analysis procedure



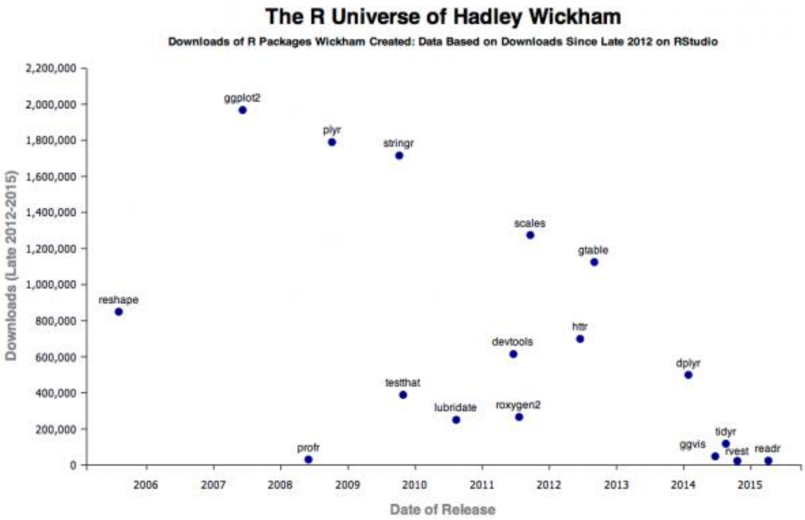
data analysis procedure



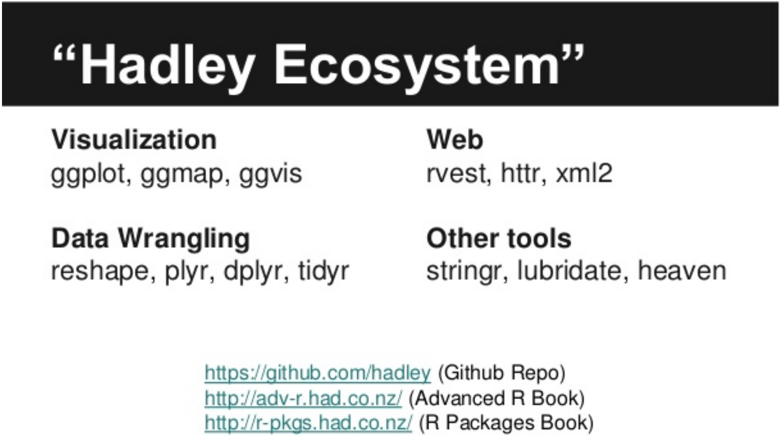
Hadley Wickham

- A statistician, a chief scientist,an assistant professor.
- readr,readxl
- stringr, reshape, reshape2, plyr, dplyr
- ggplot2, ggvis

R Universe of Hadley



R ecosystem of Hadley



“Hadley Ecosystem”

Visualization
ggplot, ggmap, ggvis

Web
rvest, httr, xml2

Data Wrangling
reshape, plyr, dplyr, tidyr

Other tools
stringr, lubridate, heaven

https://github.com/hadley (Github Repo)
http://adv-r.had.co.nz/ (Advanced R Book)
http://r-pkgs.had.co.nz/ (R Packages Book)

tidyverse

dplyr
forcats
ggplot2
haven
purrr
readr
readxl
stringr
tidyr

tidyverse

The tidy tools manifesto

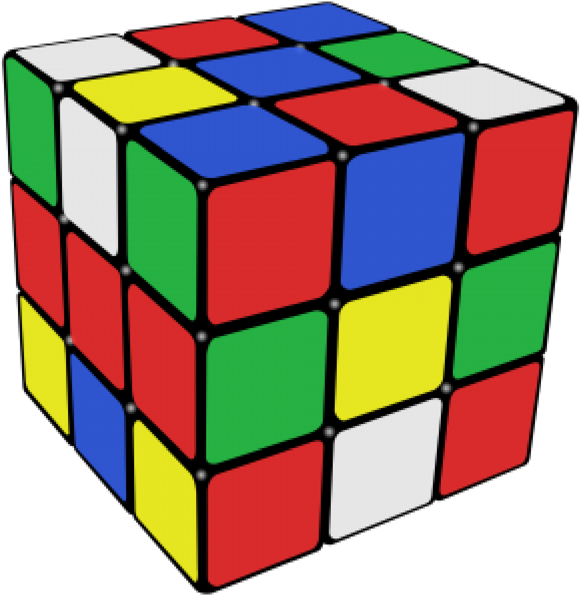
There are four basic principles to a tidy API:

- Reuse existing data structures.
- Compose simple functions with the pipe.
- Embrace functional programming.
- Design for humans.

Design for humans

> Programs must be written for people to read, and only incidentally for machines to execute.
>
> — Hal Abelson

data manipulation: Rubik's cube



data manipulation

- long and wide formatted data
- pivot table
- merge data

data manipulation

- **long and wide formatted data**
- pivot table
- merge data

data format

- long formatted data:R
- wide formatted data:SPSS

long data

```
  subject sex condition measurement
1       1   M   control         7.9
2       1   M     cond1        12.3
3       1   M     cond2        10.7
4       2   F   control         6.3
5       2   F     cond1        10.6
6       2   F     cond2        11.1
```

wide data

```
  subject sex control cond1 cond2
1       1   M     7.9  12.3  10.7
2       2   F     6.3  10.6  11.1
3       3   F     9.5  13.1  13.8
4       4   M    11.5  13.4  12.9
```

iris data: wide part

```r
head(iris[, c(1:4)])
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1          5.1         3.5          1.4         0.2
2          4.9         3.0          1.4         0.2
3          4.7         3.2          1.3         0.2
4          4.6         3.1          1.5         0.2
5          5.0         3.6          1.4         0.2
6          5.4         3.9          1.7         0.4
```

iris data: long part

```r
head(iris[, c(4:5)])
```

```
  Petal.Width Species
1         0.2  setosa
2         0.2  setosa
3         0.2  setosa
4         0.2  setosa
5         0.2  setosa
6         0.4  setosa
```

stack and unstack

```
head(iris, 1)
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
```

```
iris_w <- iris[, c(1:4)]  ## wide data
iris_l <- stack(iris_w)  ##long data = stacked data
str(iris_l)
```

```
'data.frame':   600 obs. of  2 variables:
 $ values: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ ind   : Factor w/ 4 levels "Sepal.Length",..: 1 1 1 1 1
```

```
iris_w <- unstack(iris_l)  ##wide data = unstacked data
head(iris_w, 1)
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width
```

```
subdata <- iris[, 4:5]  ## long data
str(subdata)
```

```
'data.frame':   150 obs. of  2 variables:
 $ Petal.Width: num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.
 $ Species    : Factor w/ 3 levels "setosa","versicolor",..
```

```
data_w <- unstack(subdata)
colMeans(data_w)
```

```
    setosa versicolor  virginica
     0.246      1.326      2.026
```

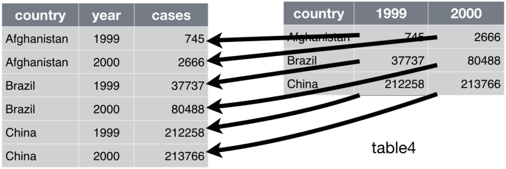
```
with(iris, tapply(iris[, 4], Species, mean))
```

```
    setosa versicolor  virginica
     0.246      1.326      2.026
```

package tidyr

- gather: wide to long
- spread: long to wide

wide to long: gather

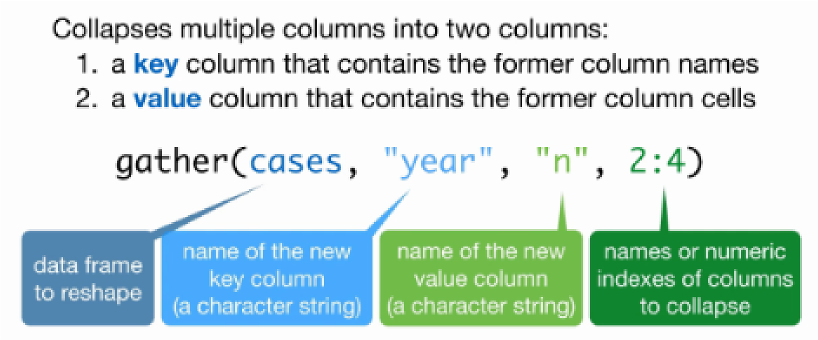


country	year	cases
Afghanistan	1999	745
Afghanistan	2000	2666
Brazil	1999	37737
Brazil	2000	80488
China	1999	212258
China	2000	213766

country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766

table4

wide to long: gather



Collapses multiple columns into two columns:

1. a **key** column that contains the former column names
2. a **value** column that contains the former column cells

```
gather(cases, "year", "n", 2:4)
```

data frame to reshape

name of the new key column (a character string)

name of the new value column (a character string)

names or numeric indexes of columns to collapse

wide data

```r
rawdata_wide <- read.csv("data/rawdata_wide.csv")
rawdata_wide
```

```
  subject sex control cond1 cond2
1       1   M     7.9  12.3  10.7
2       2   F     6.3  10.6  11.1
3       3   F     9.5  13.1  13.8
4       4   M    11.5  13.4  12.9
```

wide to long: gather

The arguments to gather():

- data: Data object
- key: Name of new key column (made from names of data columns)
- value: Name of new value column

```
library(tidyr)
rawdata_wide <- read.csv("data/rawdata_wide.csv")
data_long <- gather(rawdata_wide, condition, measurement, c
# data_long
str(data_long)
```

```
'data.frame':   12 obs. of  4 variables:
 $ subject    : int  1 2 3 4 1 2 3 4 1 2 ...
 $ sex        : Factor w/ 2 levels "F","M": 2 1 1 2 2 1 1 2
 $ condition  : chr  "control" "control" "control" "contro
 $ measurement: num  7.9 6.3 9.5 11.5 12.3 10.6 13.1 13.4 1
```

arrange the data

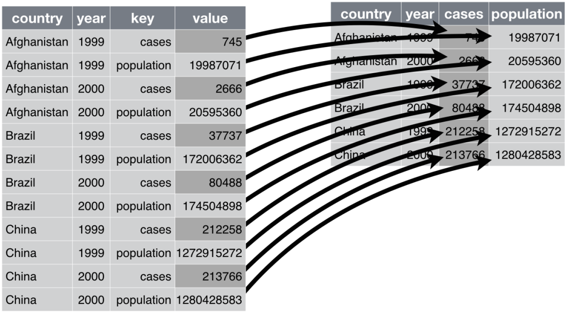
```
# Rename factor names from 'cond1' and 'cond2' to 'first'
# 'second'
levels(data_long$condition)[levels(data_long$condition) ==
levels(data_long$condition)[levels(data_long$condition) ==
# Sort by subject first, then by condition
data_long <- data_long[order(data_long$subject, data_long$
    ]
# arrange(data_long,subject,condition)
```

arrange the data

```
data_long <- read.csv("data/rawdata_long.csv")
# Rename factor names from 'cond1' and 'cond2' to 'first' a
# 'second'
levels(data_long$condition)[levels(data_long$condition) ==
levels(data_long$condition)[levels(data_long$condition) ==
# Sort by subject first, then by condition
data_long <- data_long[order(data_long$subject, data_long$c
    ]
# arrange(data_long,subject,condition)
head(data_long)
```

```
  subject sex condition measurement
2       1   M     first        12.3
3       1   M    second        10.7
1       1   M   control         7.9
5       2   F     first        10.6
6       2   F    second        11.1
```

long to wide: spread

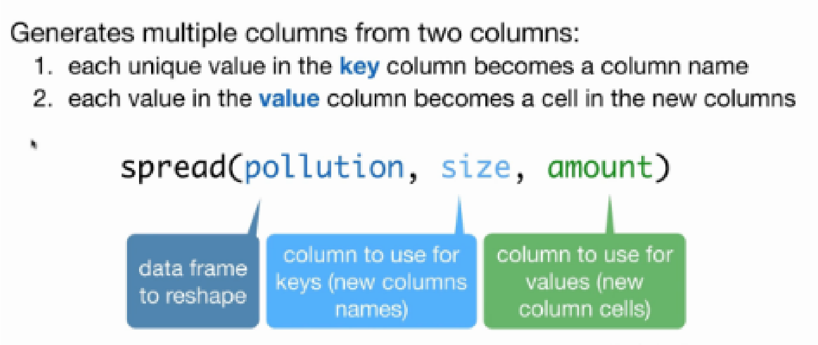


country	year	key	value
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

table2

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

long to wide: spread



Generates multiple columns from two columns:

1. each unique value in the **key** column becomes a column name
2. each value in the **value** column becomes a cell in the new columns

```
spread(pollution, size, amount)
```

data frame to reshape
column to use for keys (new columns names)
column to use for values (new column cells)

long data

```r
rawdata_long <- read.csv("data/rawdata_long.csv")
head(rawdata_long)
```

```
  subject sex condition measurement
1       1   M   control         7.9
2       1   M     cond1        12.3
3       1   M     cond2        10.7
4       2   F   control         6.3
5       2   F     cond1        10.6
6       2   F     cond2        11.1
```

long to wide: spread

The arguments to spread():

- data: Data object
- key: Name of column containing the new column names
- value: Name of column containing values

```
library(tidyr)
rawdata_long <- read.csv("data/rawdata_long.csv")
data_wide <- spread(rawdata_long, condition, measurement)
data_wide
```

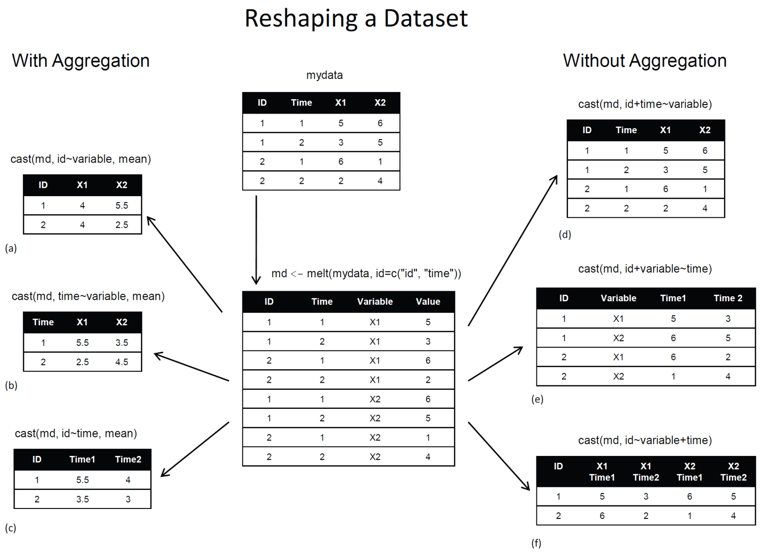
```
  subject sex cond1 cond2 control
1       1   M  12.3  10.7     7.9
2       2   F  10.6  11.1     6.3
3       3   F  13.1  13.8     9.5
4       4   M  13.4  12.9    11.5
```

arrange the data

```r
# Rename cond1 to first, and cond2 to second
names(data_wide)[names(data_wide) == "cond1"] <- "first"
names(data_wide)[names(data_wide) == "cond2"] <- "second"
# Reorder the columns
data_wide <- data_wide[, c(1, 2, 5, 3, 4)]
data_wide
```

```
  subject sex control first second
1       1   M     7.9  12.3   10.7
2       2   F     6.3  10.6   11.1
3       3   F     9.5  13.1   13.8
4       4   M    11.5  13.4   12.9
```

package reshape2



Reshaping a Dataset

With Aggregation

Without Aggregation

mydata

ID	Time	X1	X2
1	1	5	6
1	2	3	5
2	1	6	1
2	2	2	4

md <– melt(mydata, id=c("id", "time"))

ID	Time	Variable	Value
1	1	X1	5
1	2	X1	3
2	1	X1	6
2	2	X1	2
1	1	X2	6
1	2	X2	5
2	1	X2	1
2	2	X2	4

cast(md, id~variable, mean)

ID	X1	X2
1	4	5.5
2	4	2.5

(a)

cast(md, time~variable, mean)

Time	X1	X2
1	5.5	3.5
2	2.5	4.5

(b)

cast(md, id~time, mean)

ID	Time1	Time2
1	5.5	4
2	3.5	3

(c)

cast(md, id+time~variable)

ID	Time	X1	X2
1	1	5	6
1	2	3	5
2	1	6	1
2	2	2	4

(d)

cast(md, id+variable~time)

ID	Variable	Time1	Time 2
1	X1	5	3
1	X2	6	5
2	X1	6	2
2	X2	1	4

(e)

cast(md, id~variable+time)

ID	X1 Time1	X1 Time2	X2 Time1	X2 Time2
1	5	3	6	5
2	6	2	1	4

(f)

package reshape2

- melt takes wide-format data and melts it into long-format data.
- dcast takes long-format data and casts it into wide-format data.

Think of working with metal: if you melt metal, it drips and becomes long. If you cast it into a mould, it becomes wide.

reshape2: wide to long

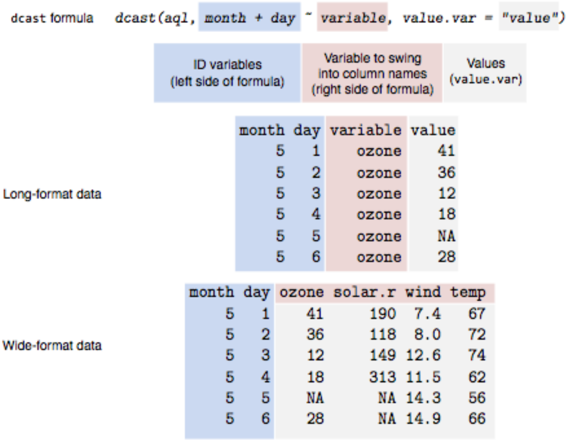
- melt

```
library(reshape2)
rawdata_wide<-read.csv('data/rawdata_wide.csv')
rawdata_wide
```

```
  subject sex control cond1 cond2
1       1   M     7.9  12.3  10.7
2       2   F     6.3  10.6  11.1
3       3   F     9.5  13.1  13.8
4       4   M    11.5  13.4  12.9
```

```
data_long <- melt(rawdata_wide,
                  id.vars=c("subject", "sex"))
data_long <- melt(rawdata_wide,
                    id.vars=c("subject", "sex"), ##not incl
                    measure.vars=c("control",
                                   "cond1", "cond2" ),## or
```

reshape2: long to wide.



dcast formula

```
dcast(aql, month + day ~ variable, value.var = "value")
```

ID variables (left side of formula)

Variable to swing into column names (right side of formula)

Values (value.var)

Long-format data

month	day	variable	value
5	1	ozone	41
5	2	ozone	36
5	3	ozone	12
5	4	ozone	18
5	5	ozone	NA
5	6	ozone	28

Wide-format data

month	day	ozone	solar.r	wind	temp
5	1	41	190	7.4	67
5	2	36	118	8.0	72
5	3	12	149	12.6	74
5	4	18	313	11.5	62
5	5	NA	NA	14.3	56
5	6	28	NA	14.9	66

reshape2: long to wide.

The arguments to dcast():

From the source: “subject” and “sex” are columns we want to keep the same “condition” is the column that contains the names of the new column to put things in “measurement” holds the measurements

```
library(reshape2)
rawdata_long <- read.csv("data/rawdata_long.csv")
head(rawdata_long, 1)
```

```
  subject sex condition measurement
1       1   M   control         7.9
```

```
dcast(rawdata_long, subject + sex ~ condition, value.var =
    mean)
```

```
  subject sex cond1 cond2 control
```

data manipulation

- long and wide formatted data
- **pivot table**
- merge data

iris example

```r
library(reshape2)
iris_long <- melt(iris, id.vars = "Species")
head(iris_long, 1)
```

```
  Species     variable value
1  setosa Sepal.Length   5.1
```

```r
iris_long <- melt(iris, id = "Species", variable.name = "me
    value.name = "value")
head(iris_long)
```

```
  Species  measurement value
1  setosa Sepal.Length   5.1
2  setosa Sepal.Length   4.9
3  setosa Sepal.Length   4.7
4  setosa Sepal.Length   4.6
5  setosa Sepal.Length   5.0
```

Alternative methods

```
t(sapply(iris[, 1:4], function(x) tapply(x, iris$Species, m
    na.rm = T)))
```

```
             setosa versicolor virginica
Sepal.Length  5.006      5.936     6.588
Sepal.Width   3.428      2.770     2.974
Petal.Length  1.462      4.260     5.552
Petal.Width   0.246      1.326     2.026
```

```
s <- split(iris, iris$Species)
sapply(s, function(x) colMeans(x[, 1:4], na.rm = T))
```

```
             setosa versicolor virginica
Sepal.Length  5.006      5.936     6.588
Sepal.Width   3.428      2.770     2.974
Petal.Length  1.462      4.260     5.552
Petal.Width   0.246      1.326     2.026
```

tips example

```r
head(tips)[1:3, ]
```

```
  total_bill  tip    sex smoker day   time size
1      16.99 1.01 Female     No Sun Dinner    2
2      10.34 1.66   Male     No Sun Dinner    3
3      21.01 3.50   Male     No Sun Dinner    3
```

```r
dcast(tips, sex ~ ., value.var = "tip", fun = mean)
```

```
     sex        .
1 Female 2.833448
2   Male 3.089618
```

```r
dcast(tips, sex ~ size, value.var = "tip", fun = mean)
```

```
     sex        1        2        3        4    5    6
1 Female 1.276667 2.528448 3.250000 4.021111 5.14 4.60
```

tips example

```
dcast(tips, sex ~ ., value.var = "tip", fun = mean)
```

```
     sex        .
1 Female 2.833448
2   Male 3.089618
```

```
dcast(tips, sex ~ ., value.var = "total_bill", fun = mean)
```

```
     sex        .
1 Female 18.05690
2   Male 20.74408
```

```
tips_melt <- melt(tips, id.vars = c("sex", "smoker", "day",
    "size"))
head(tips_melt)
```

```
     sex smoker day   time size   variable value
```

tips example

```
tips_melt <- melt(tips, id.vars = c("sex", "smoker", "day"
    "size"))
head(tips_melt)
```

```
     sex smoker day   time size   variable value
1 Female     No Sun Dinner    2 total_bill 16.99
2   Male     No Sun Dinner    3 total_bill 10.34
3   Male     No Sun Dinner    3 total_bill 21.01
4   Male     No Sun Dinner    2 total_bill 23.68
5 Female     No Sun Dinner    4 total_bill 24.59
6   Male     No Sun Dinner    4 total_bill 25.29
```

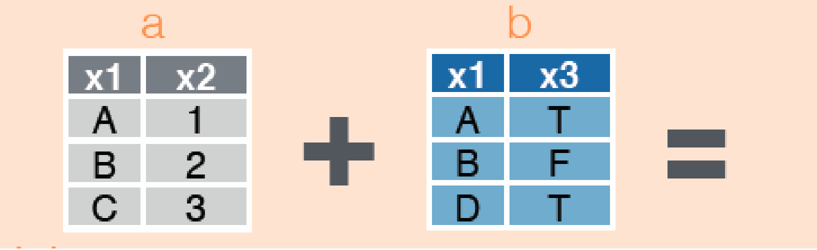
```
tips_mean <- dcast(tips_melt, sex + smoker ~ variable, fun
tips_mean
```

```
     sex smoker total_bill      tip
1 Female     No   18.10519 2.773519
```

data manipulation

- long and wide formatted data
- pivot table
- **merge data**

merge data



a

x1	x2
A	1
B	2
C	3

b

x1	x3
A	T
B	F
D	T

merge data

```
data1 <- read.dta("data/data1.dta")
data2 <- read.dta("data/data2.dta")
data1
```

```
    IDs gender
1 subj1   male
2 subj2 female
3 subj3 female
4 subj4   male
5 subj5   male
```

```
data2
```

```
    IDs age
1 subj2  21
2 subj3  23
3 subj4  33
4 subj5  27
```

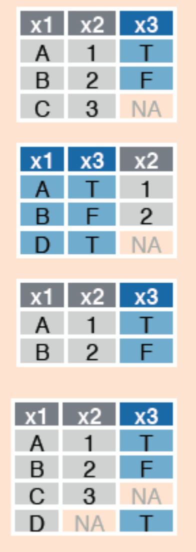
merge data

```r
data1 <- read.dta("data/data1.dta")
data2 <- read.dta("data/data2.dta")
merge(data1, data2, by = "IDs")
```

```
    IDs gender age
1 subj2 female  21
2 subj3 female  23
3 subj4   male  33
4 subj5   male  27
```

```r
# merge(data1,data2,by.x ='IDs',by.y='ID')
```

merge data



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

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

x1	x2	x3
A	1	T
B	2	F

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

merge data

```
library(foreign)
data1 <- read.dta("data/data1.dta")
data2 <- read.dta("data/data2.dta")
merge(data1, data2, by = "IDs", all = T)
```

```
    IDs gender age
1 subj1   male  NA
2 subj2 female  21
3 subj3 female  23
4 subj4   male  33
5 subj5   male  27
6 subj6   <NA>  19
```

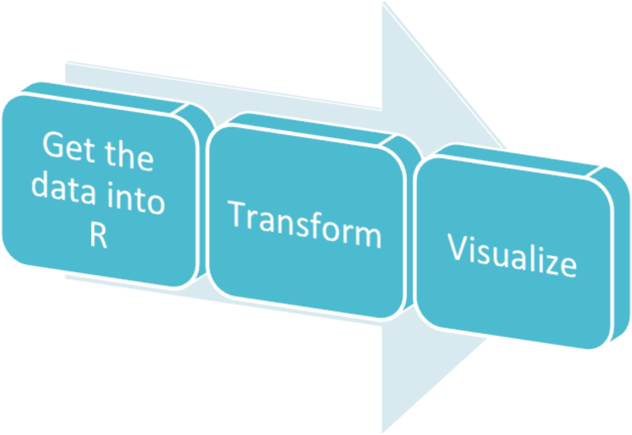
```
merge(data1, data2, by = "IDs", all.x = T)
```

```
    IDs gender age
1 subj1   male  NA
```

data manipulation

- long and wide formatted data
- pivot table
- merge data

data analysis procedure



Get the data into R
Transform
Visualize