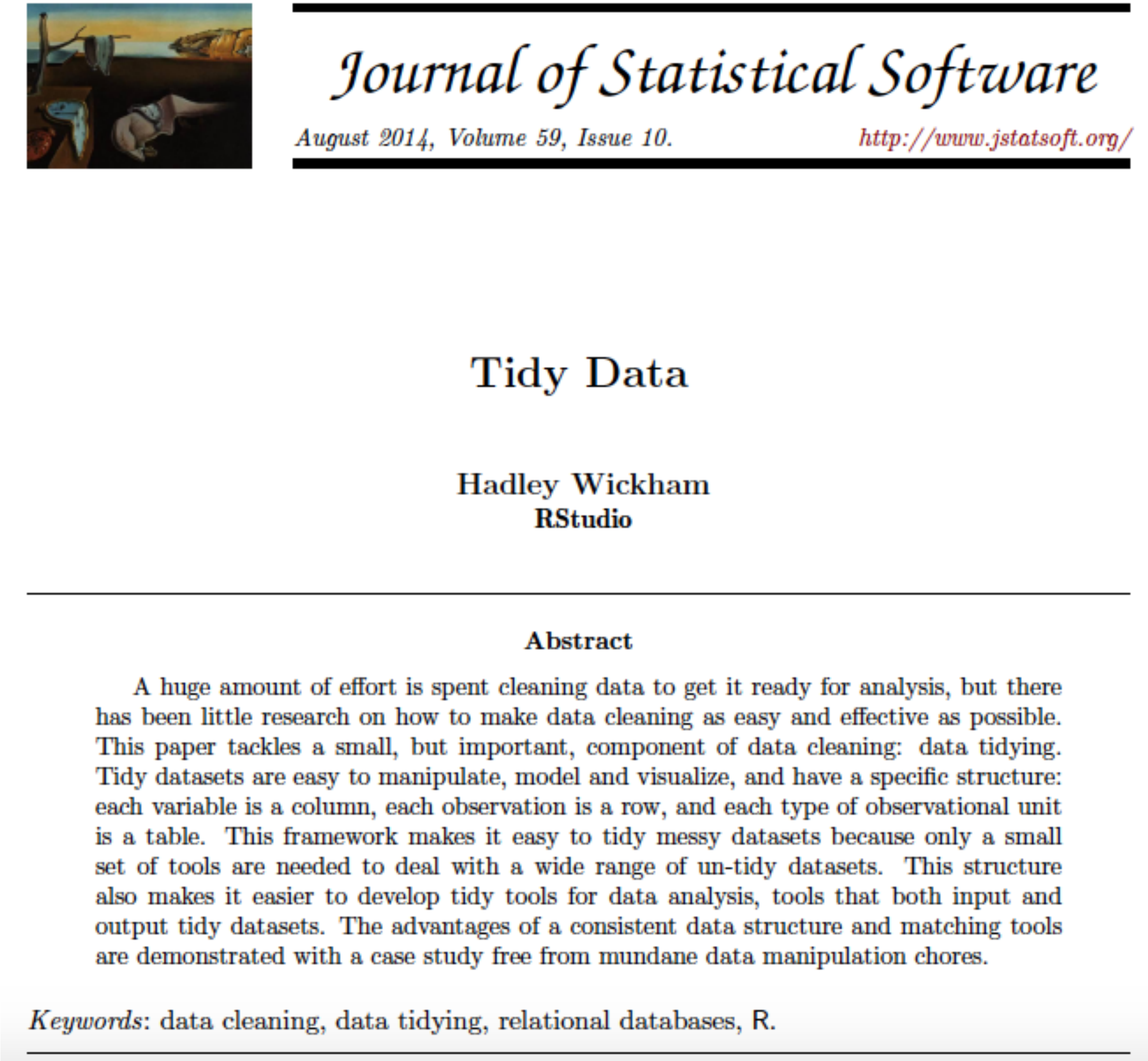
CT5102: Programming for Data Analytics

Week 8:
Tidy Data and dplyr

https://github.com/JimDuggan/CT5102

Dr. Jim Duggan,
Information Technology,
School of Engineering & Informatics

Key Reference



Journal of Statistical Software

August 2014, Volume 59, Issue 10. http://www.jstatsoft.org/

Tidy Data

Hadley Wickham
RStudio

Abstract

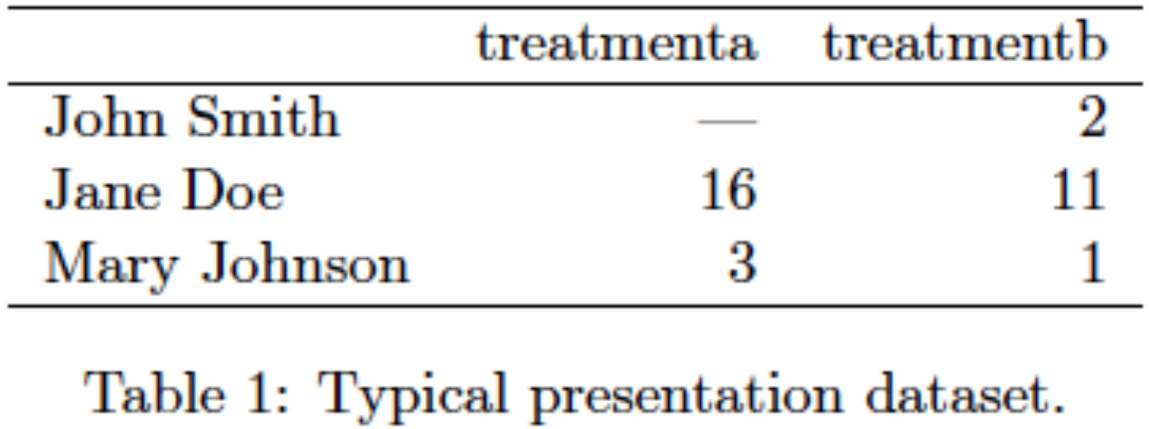
A huge amount of effort is spent cleaning data to get it ready for analysis, but there has been little research on how to make data cleaning as easy and effective as possible. This paper tackles a small, but important, component of data cleaning: data tidying. Tidy datasets are easy to manipulate, model and visualize, and have a specific structure: each variable is a column, each observation is a row, and each type of observational unit is a table. This framework makes it easy to tidy messy datasets because only a small set of tools are needed to deal with a wide range of un-tidy datasets. This structure also makes it easier to develop tidy tools for data analysis, tools that both input and output tidy datasets. The advantages of a consistent data structure and matching tools are demonstrated with a case study free from mundane data manipulation chores.

Keywords: data cleaning, data tidying, relational databases, R.

Overview

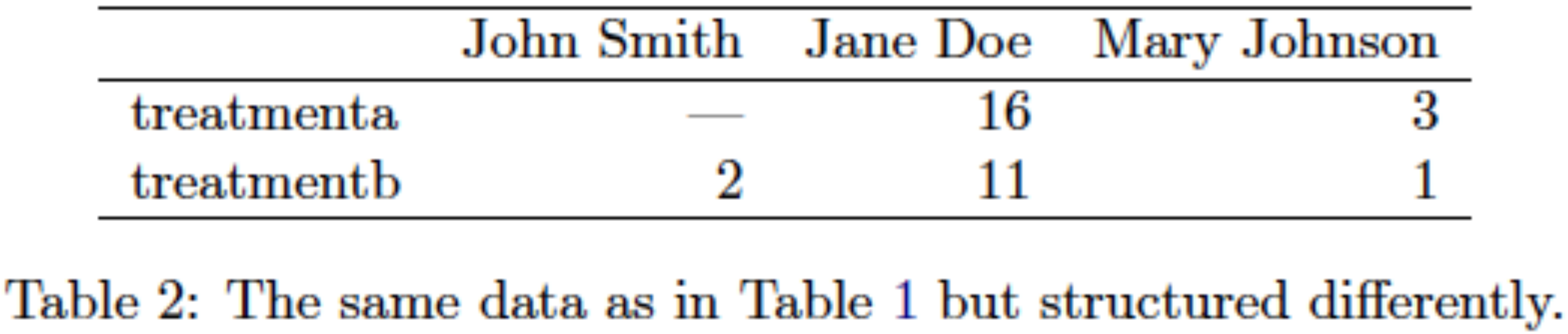
- What is data tidying?
  - Structuring datasets to facilitate analysis
- The tidy data standard is designed to:
  - Facilitate initial exploration and analysis of data
  - Simplify the development of data analysis tools that work well together
- Principles closely related to relational algebra (Codd 1990)
- Related packages: ggplot2, reshape, reshape2, plyr, dplyr

Typical Structure: Rows and Columns (Wickham 2014)



	treatmenta	treatmentb
John Smith	—	2
Jane Doe	16	11
Mary Johnson	3	1

Table 1: Typical presentation dataset.



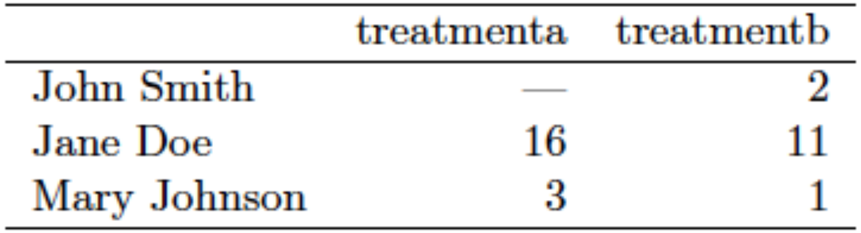
	John Smith	Jane Doe	Mary Johnson
treatmenta	—	16	3
treatmentb	2	11	1

Table 2: The same data as in Table 1 but structured differently.

Numbers refer to the result of the treatments on a given person.

Data Semantics

- A dataset is a collection of values, usually numbers (if quantitative) or strings (if qualitative)
- Every value belongs to a variable and an observation
- An observation contains all values measured on the same unit (e.g. person or day)

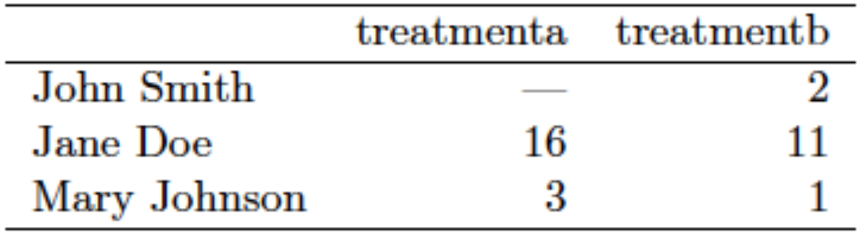


	treatmenta	treatmentb
John Smith	—	2
Jane Doe	16	11
Mary Johnson	3	1

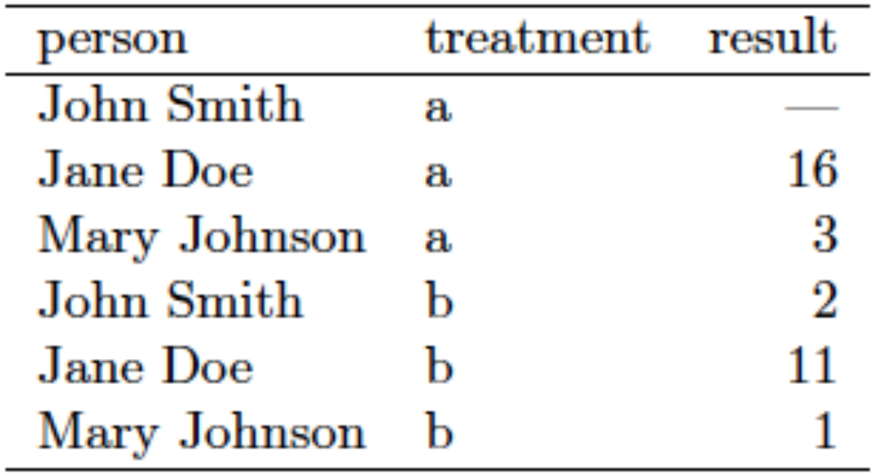
- The variables are:
  - Person (John, Jane, and Mary)
  - Treatment (a or b)
  - Result (6 values including NA)
- Every combination of person and treatment were measured.

Reorganising Tables

- Makes values, variables and observations clearer
- Variables in Columns
- Observations in Rows



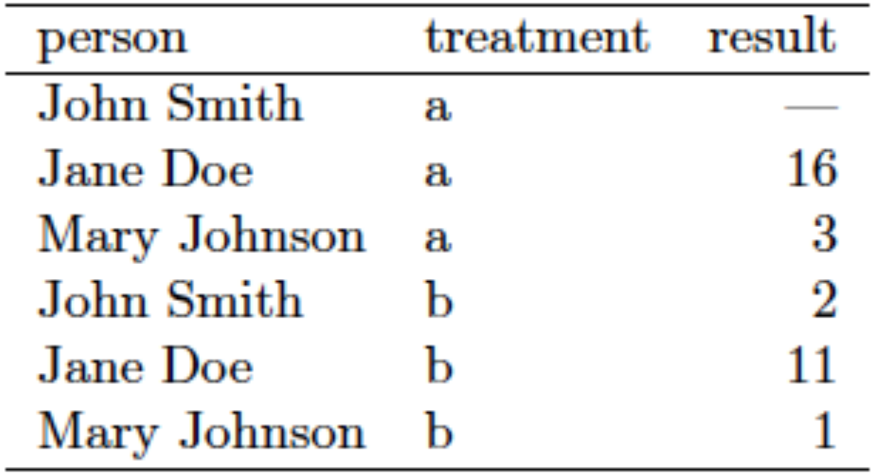
	treatmenta	treatmentb
John Smith	—	2
Jane Doe	16	11
Mary Johnson	3	1



person	treatment	result
John Smith	a	—
Jane Doe	a	16
Mary Johnson	a	3
John Smith	b	2
Jane Doe	b	11
Mary Johnson	b	1

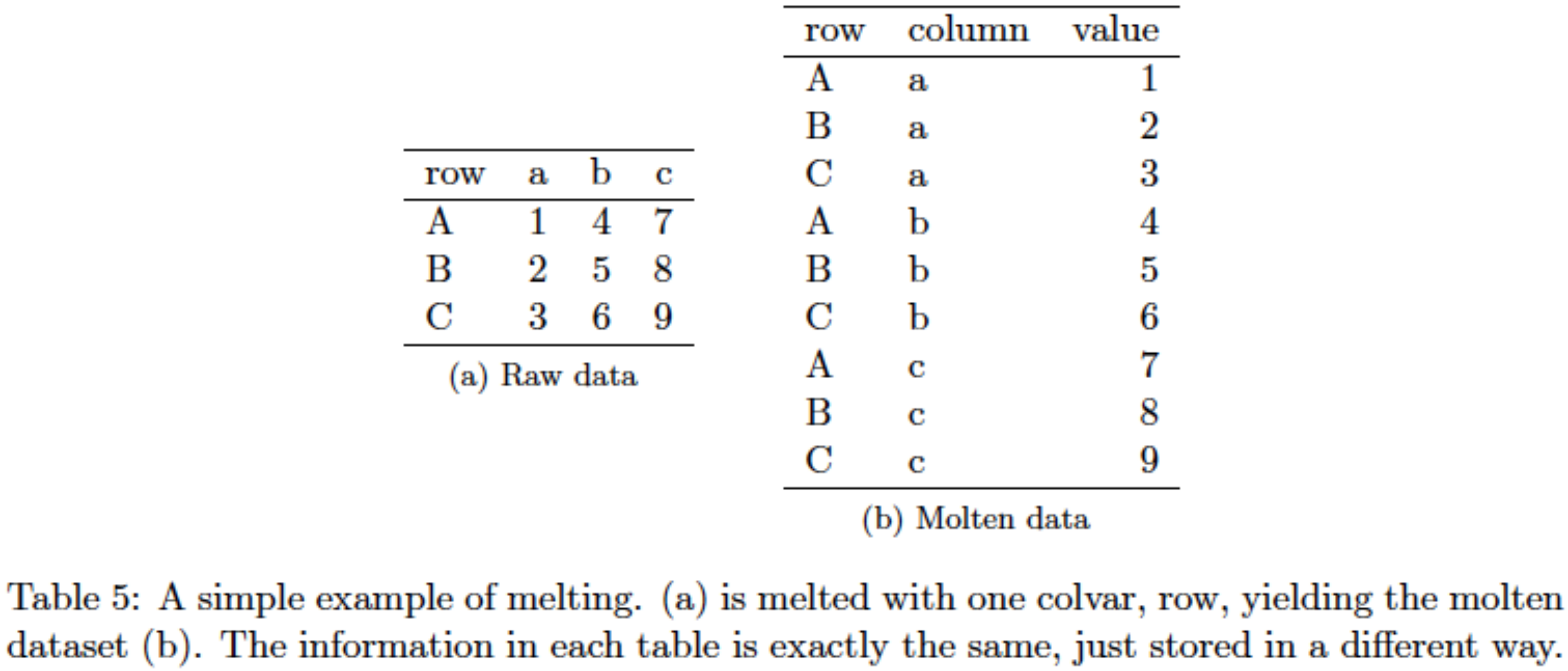
Tidy Data

- In tidy data
  - Each variable forms a column
  - Each observation forms a row
  - Each type of observational unit forms a table



person	treatment	result
John Smith	a	—
Jane Doe	a	16
Mary Johnson	a	3
John Smith	b	2
Jane Doe	b	11
Mary Johnson	b	1

Melting (Wickham 2014)



row	a	b	c
A	1	4	7
B	2	5	8
C	3	6	9

(a) Raw data

row	column	value
A	a	1
B	a	2
C	a	3
A	b	4
B	b	5
C	b	6
A	c	7
B	c	8
C	c	9

(b) Molten data

Table 5: A simple example of melting. (a) is melted with one colvar, row, yielding the molten dataset (b). The information in each table is exactly the same, just stored in a different way.

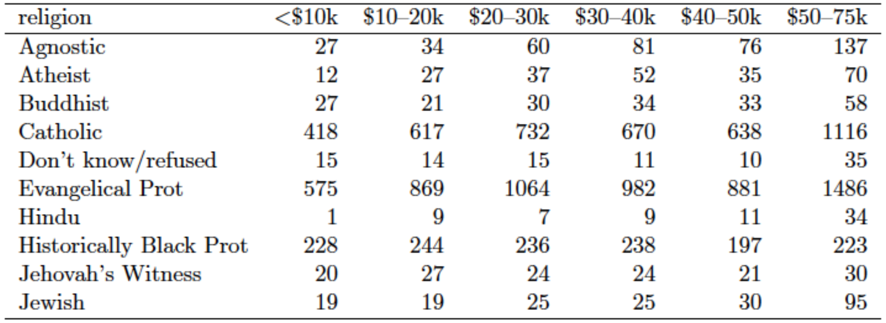
melt() function – reshape library

melt(data, id.vars, measure.vars)

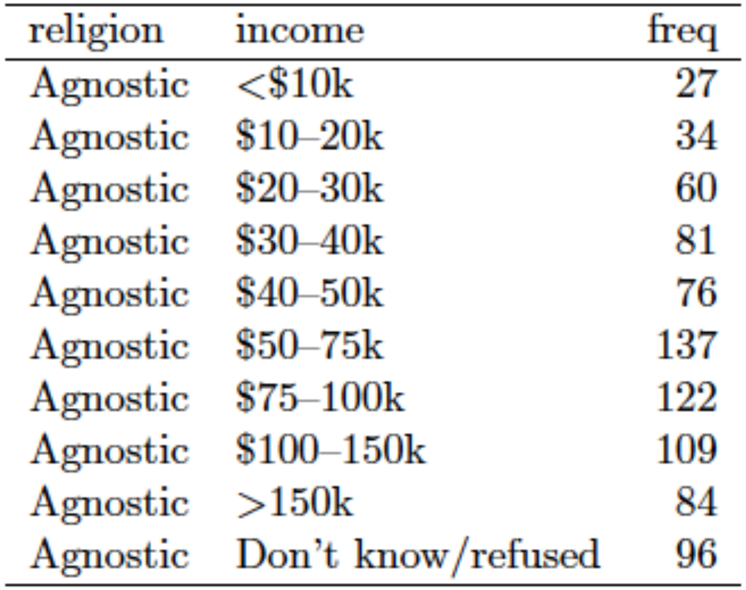
- data Data set to melt
- id.vars Id variables. If blank, will use all non measure.vars variables. Can be integer (variable position) or string (variable name)
- measure.vars Measured variables. If blank, will use all non id.vars variables. Can be integer (variable position) or string (variable name)

Ordering Variables

- Fixed Variables
  - Describe the experimental design
  - Known in advance
- Measured Variables
  - What is measured in the study
- Fixed variables come first



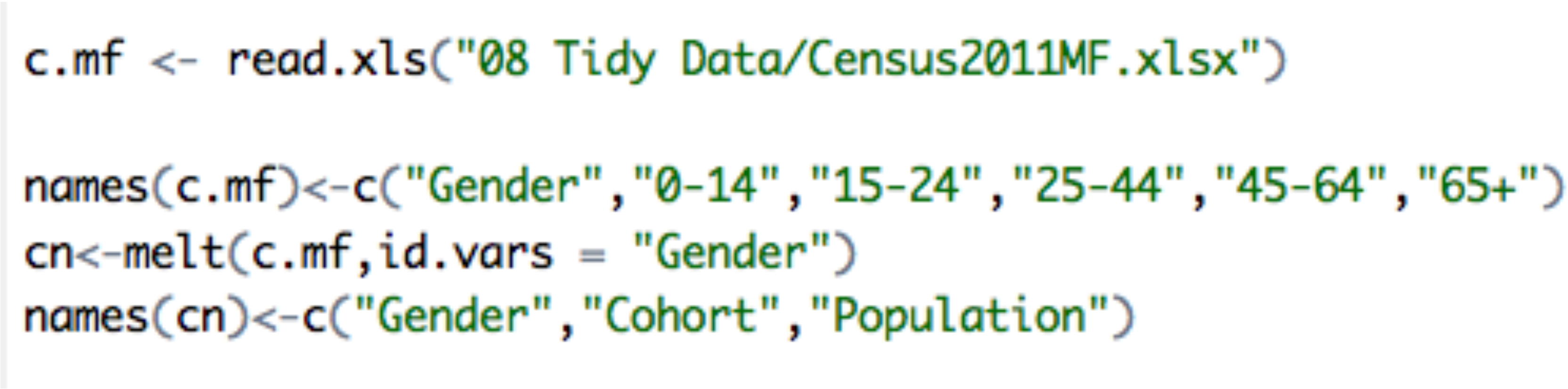
religion	<$10k	$10–20k	$20–30k	$30–40k	$40–50k	$50–75k
Agnostic	27	34	60	81	76	137
Atheist	12	27	37	52	35	70
Buddhist	27	21	30	34	33	58
Catholic	418	617	732	670	638	1116
Don't know/refused	15	14	15	11	10	35
Evangelical Prot	575	869	1064	982	881	1486
Hindu	1	9	7	9	11	34
Historically Black Prot	228	244	236	238	197	223
Jehovah's Witness	20	27	24	24	21	30
Jewish	19	19	25	25	30	95



religion	income	freq
Agnostic	<$10k	27
Agnostic	$10–20k	34
Agnostic	$20–30k	60
Agnostic	$30–40k	81
Agnostic	$40–50k	76
Agnostic	$50–75k	137
Agnostic	$75–100k	122
Agnostic	$100–150k	109
Agnostic	>150k	84
Agnostic	Don't know/refused	96

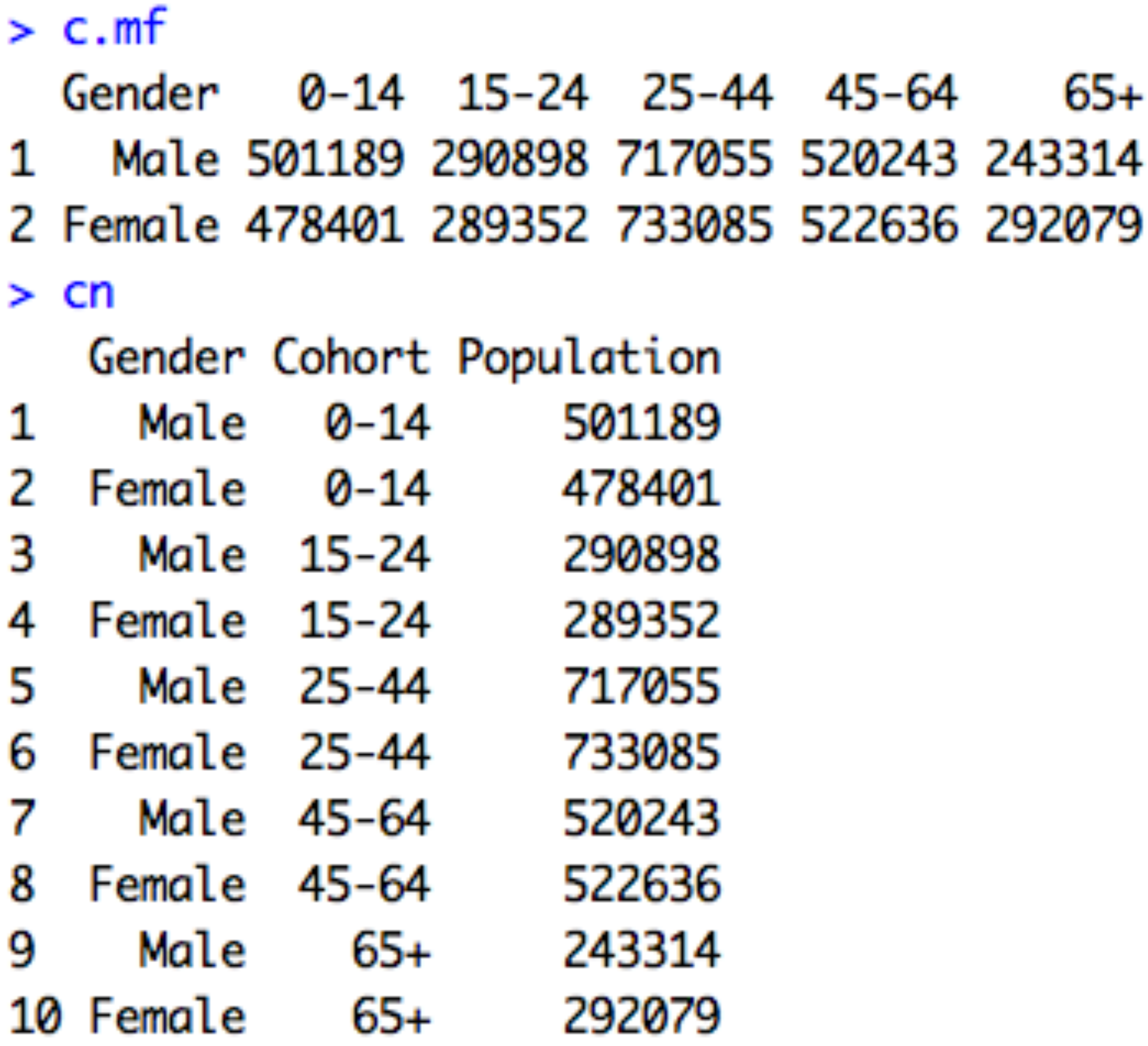
Example

Gender	0 - 14 years	15 - 24 years	25 - 44 years	45 - 64 years	65 years and over
Male	501,189	290,898	717,055	520,243	243,314
Female	478,401	289,352	733,085	522,636	292,079



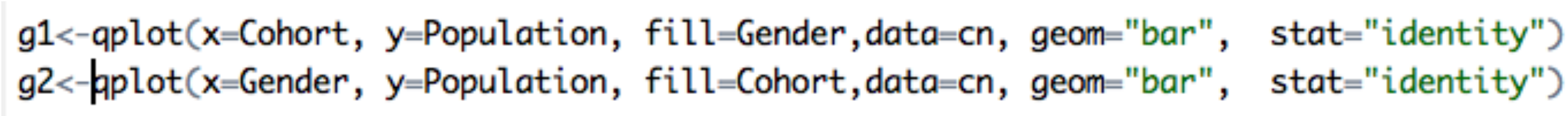
```
c.mf <- read.xls("08 Tidy Data/Census2011MF.xlsx")

names(c.mf)<-c("Gender","0-14","15-24","25-44","45-64","65+")
cn<-melt(c.mf,id.vars = "Gender")
names(cn)<-c("Gender","Cohort","Population")
```

Result

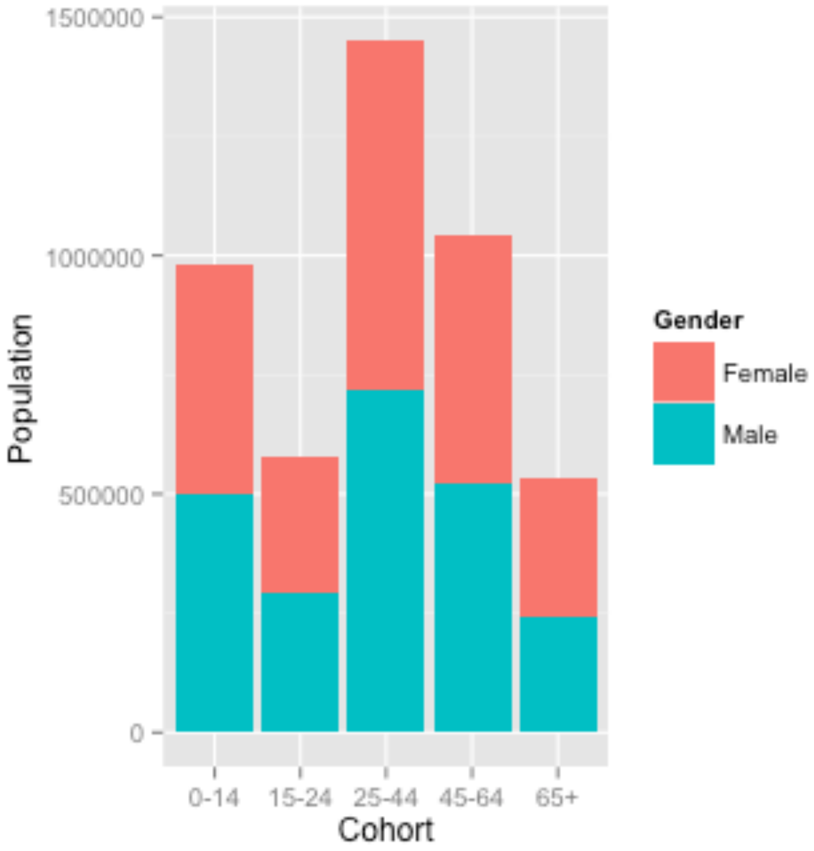


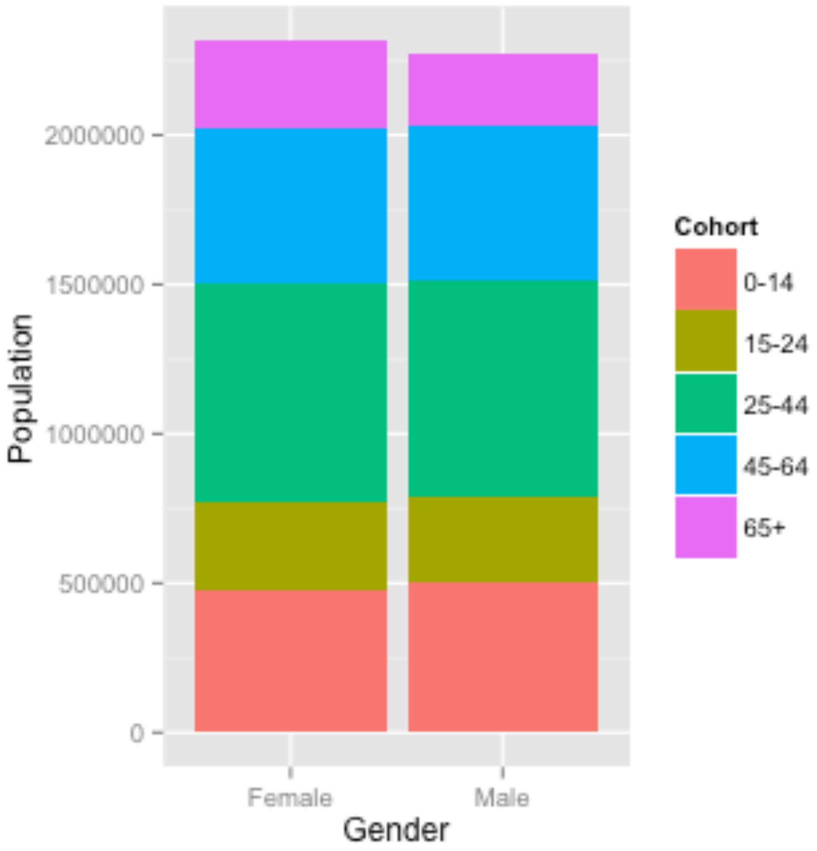
```
> c.mf
  Gender  0-14  15-24  25-44  45-64    65+
1   Male 501189 290898 717055 520243 243314
2 Female 478401 289352 733085 522636 292079
> cn
   Gender Cohort Population
1    Male   0-14     501189
2  Female   0-14     478401
3    Male  15-24     290898
4  Female  15-24     289352
5    Male  25-44     717055
6  Female  25-44     733085
7    Male  45-64     520243
8  Female  45-64     522636
9    Male    65+     243314
10 Female    65+     292079
```

Process Tidy Data with qplot



```
g1<-qplot(x=Cohort, y=Population, fill=Gender,data=cn, geom="bar",  stat="identity")
g2<-qplot(x=Gender, y=Population, fill=Cohort,data=cn, geom="bar",  stat="identity")
```

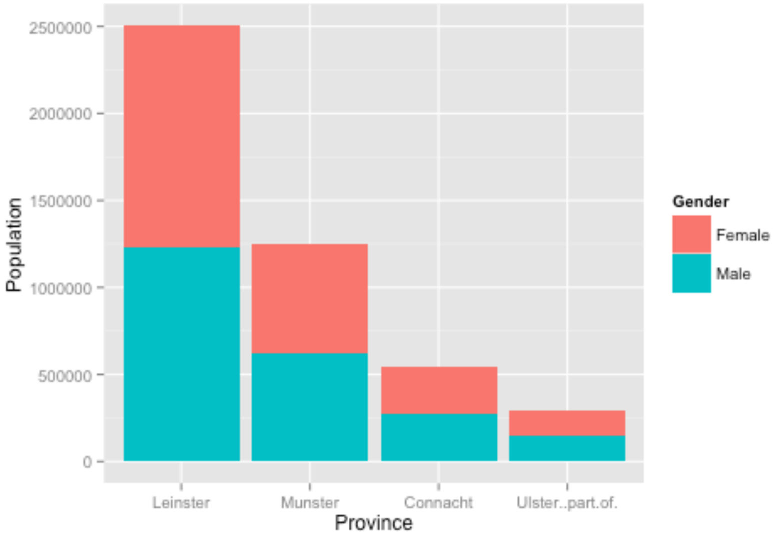




Challenge 8.1

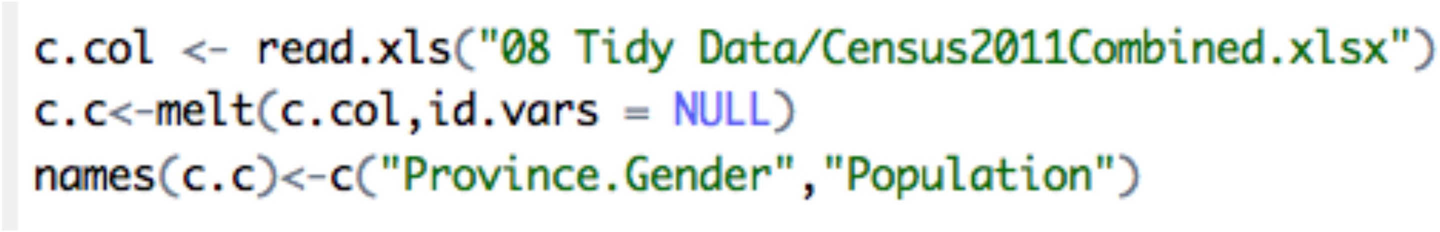
Gender	Leinster	Munster	Connacht	Ulster (part of)
Male	1,233,352	620,260	271,110	147,977
Female	1,271,462	625,828	271,437	146,826

- Show the above table in its tidy form
- Write the melt code to create a tidy data set
- Visualise the data using qplot

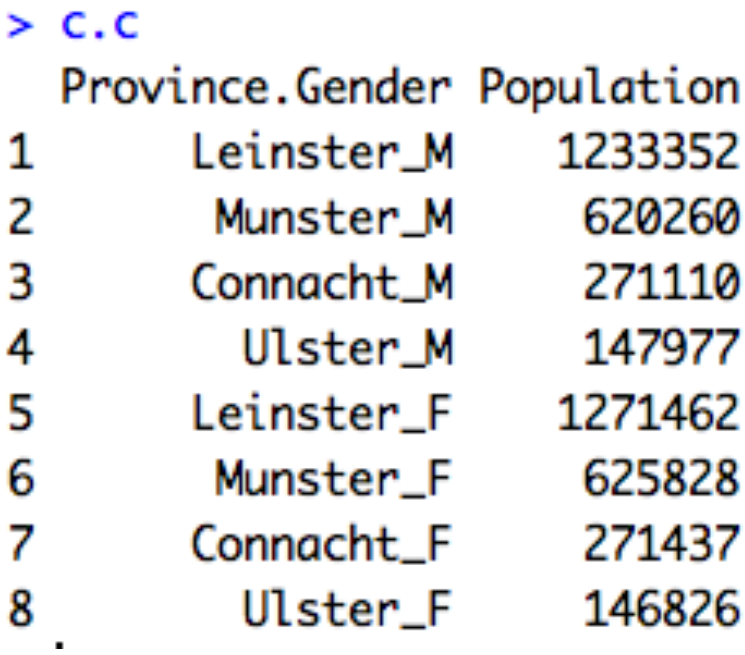


Multiple variables stored in one column (after melt)

Leinster_M	Munster_M	Connacht_M	Ulster_M	Leinster_F	Munster_F	Connacht_F	Ulster_F
1,233,352	620,260	271,110	147,977	1,271,462	625,828	271,437	146,826

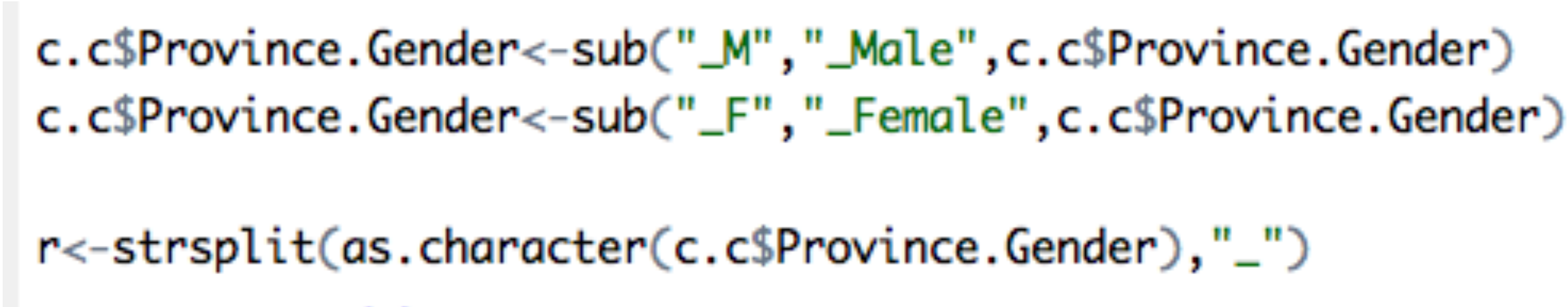


```
c.col <- read.xls("08 Tidy Data/Census2011Combined.xlsx")
c.c<-melt(c.col,id.vars = NULL)
names(c.c)<-c("Province.Gender","Population")
```

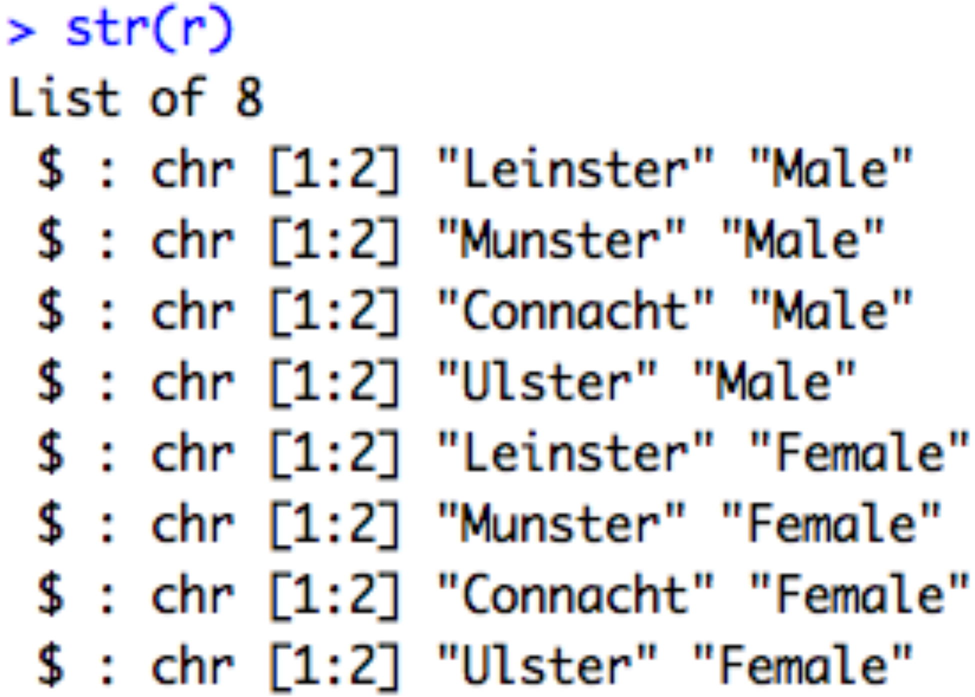


```
> c.c
  Province.Gender Population
1      Leinster_M    1233352
2       Munster_M     620260
3      Connacht_M     271110
4        Ulster_M     147977
5      Leinster_F    1271462
6       Munster_F     625828
7      Connacht_F     271437
8        Ulster_F     146826
```

Split a column into two

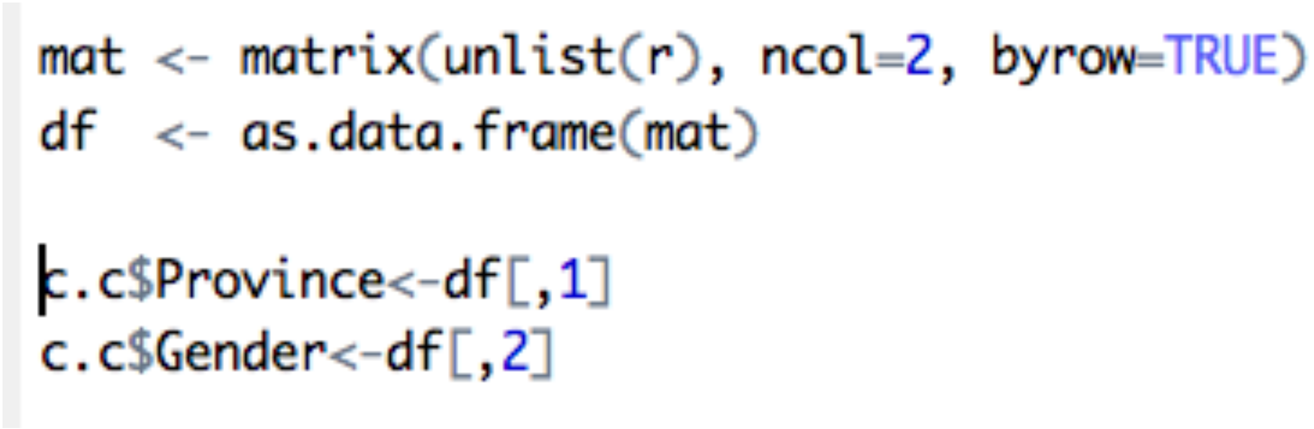


```
c.c$Province.Gender<-sub("_M","_Male",c.c$Province.Gender)
c.c$Province.Gender<-sub("_F","_Female",c.c$Province.Gender)

r<-strsplit(as.character(c.c$Province.Gender),"_")
```

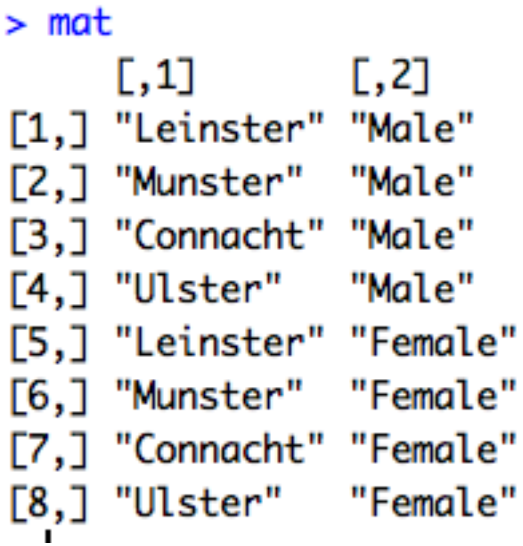


```
> str(r)
List of 8
 $ : chr [1:2] "Leinster" "Male"
 $ : chr [1:2] "Munster" "Male"
 $ : chr [1:2] "Connacht" "Male"
 $ : chr [1:2] "Ulster" "Male"
 $ : chr [1:2] "Leinster" "Female"
 $ : chr [1:2] "Munster" "Female"
 $ : chr [1:2] "Connacht" "Female"
 $ : chr [1:2] "Ulster" "Female"
```

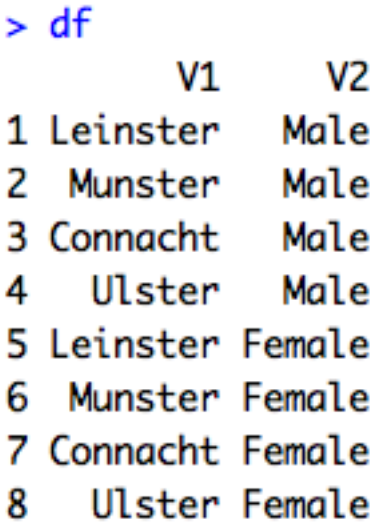
Split a column into two



```
mat <- matrix(unlist(r), ncol=2, byrow=TRUE)
df  <- as.data.frame(mat)

c.c$Province<-df[,1]
c.c$Gender<-df[,2]
```

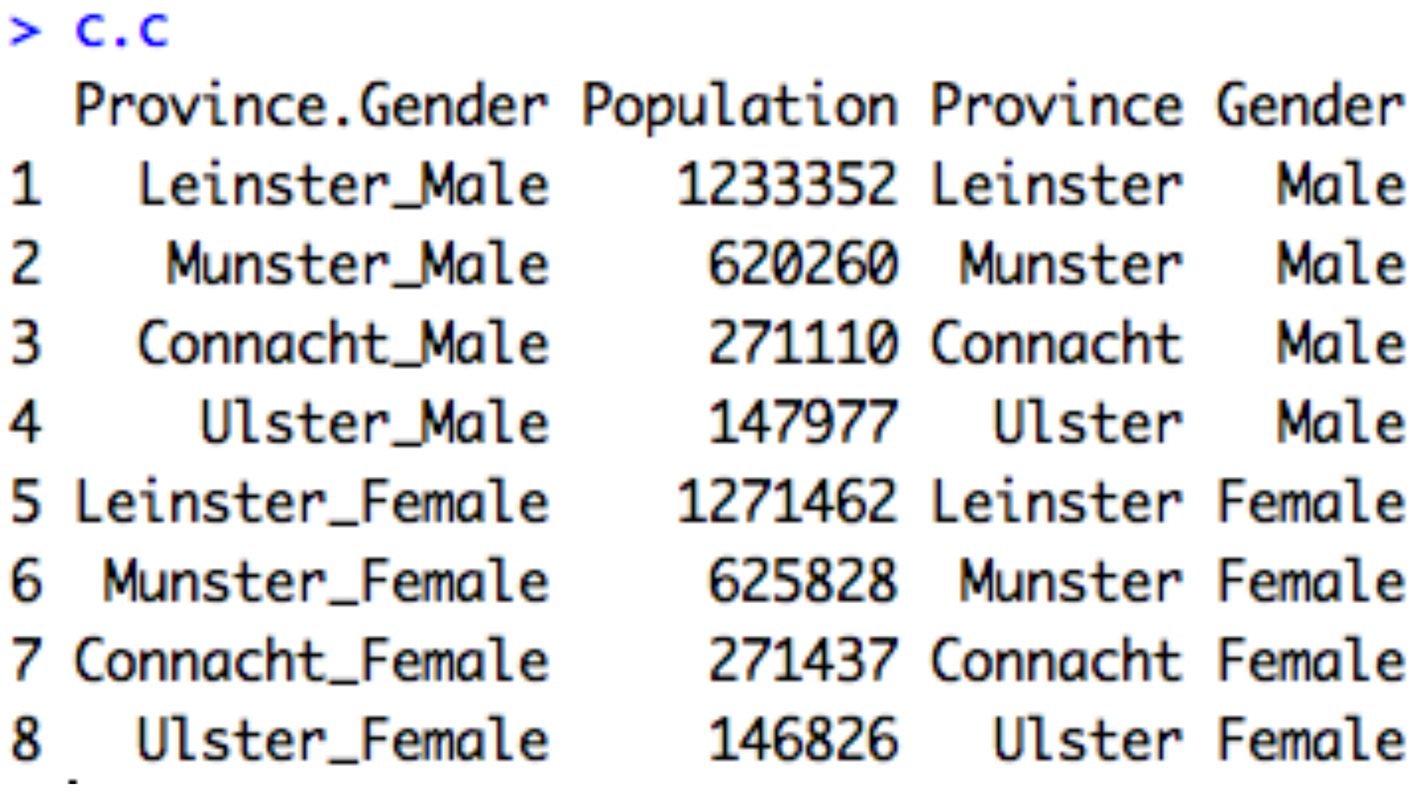


```
> mat
     [,1]       [,2]    
[1,] "Leinster" "Male"  
[2,] "Munster"  "Male"  
[3,] "Connacht" "Male"  
[4,] "Ulster"   "Male"  
[5,] "Leinster" "Female"
[6,] "Munster"  "Female"
[7,] "Connacht" "Female"
[8,] "Ulster"   "Female"
```



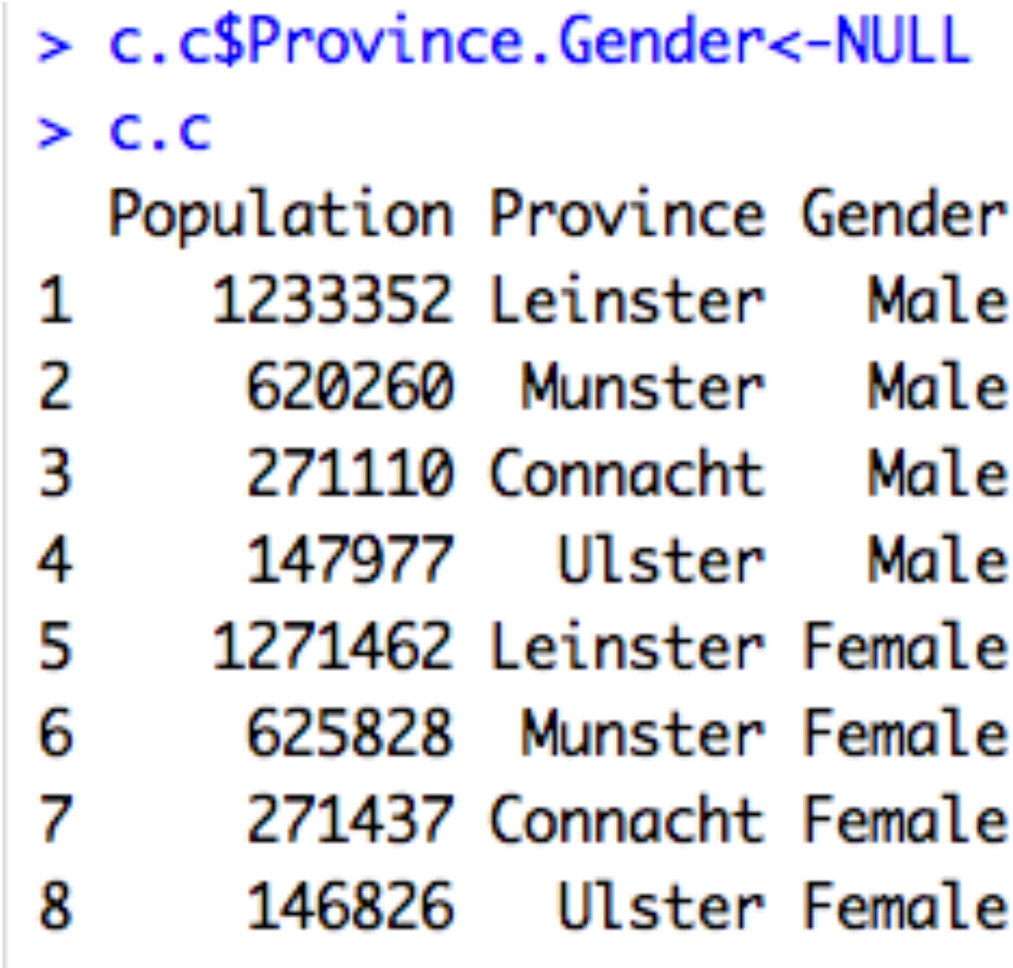
```
> df
        V1     V2
1 Leinster   Male
2  Munster   Male
3 Connacht   Male
4   Ulster   Male
5 Leinster Female
6  Munster Female
7 Connacht Female
8   Ulster Female
```

Result



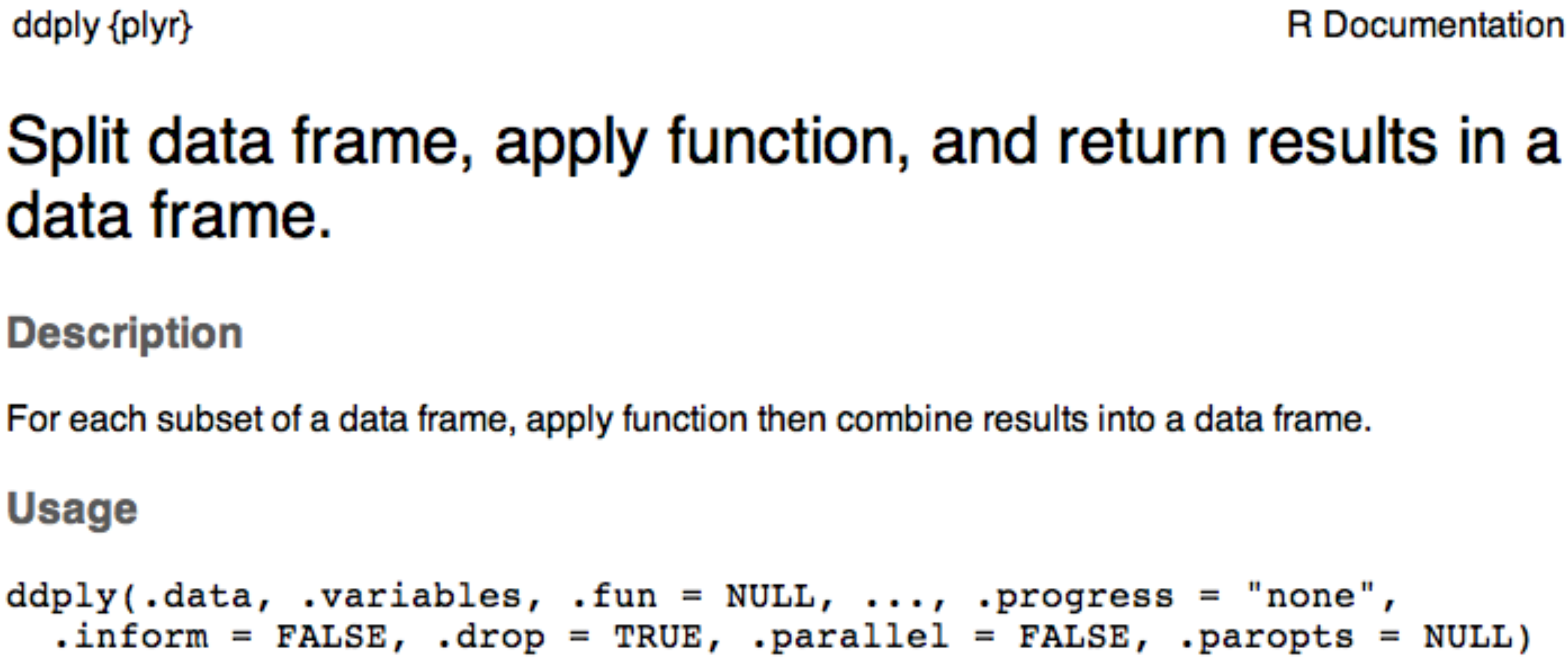
```
> c.c
  Province.Gender Population Province Gender
1   Leinster_Male    1233352 Leinster   Male
2    Munster_Male     620260  Munster   Male
3   Connacht_Male     271110 Connacht   Male
4     Ulster_Male     147977   Ulster   Male
5 Leinster_Female    1271462 Leinster Female
6  Munster_Female     625828  Munster Female
7 Connacht_Female     271437 Connacht Female
8   Ulster_Female     146826   Ulster Female
```

Remove old column



```
> c.c$Province.Gender<-NULL
> c.c
  Population Province Gender
1    1233352 Leinster   Male
2     620260  Munster   Male
3     271110 Connacht   Male
4     147977   Ulster   Male
5    1271462 Leinster Female
6     625828  Munster Female
7     271437 Connacht Female
8     146826   Ulster Female
```

ddply()



ddply {plyr} R Documentation

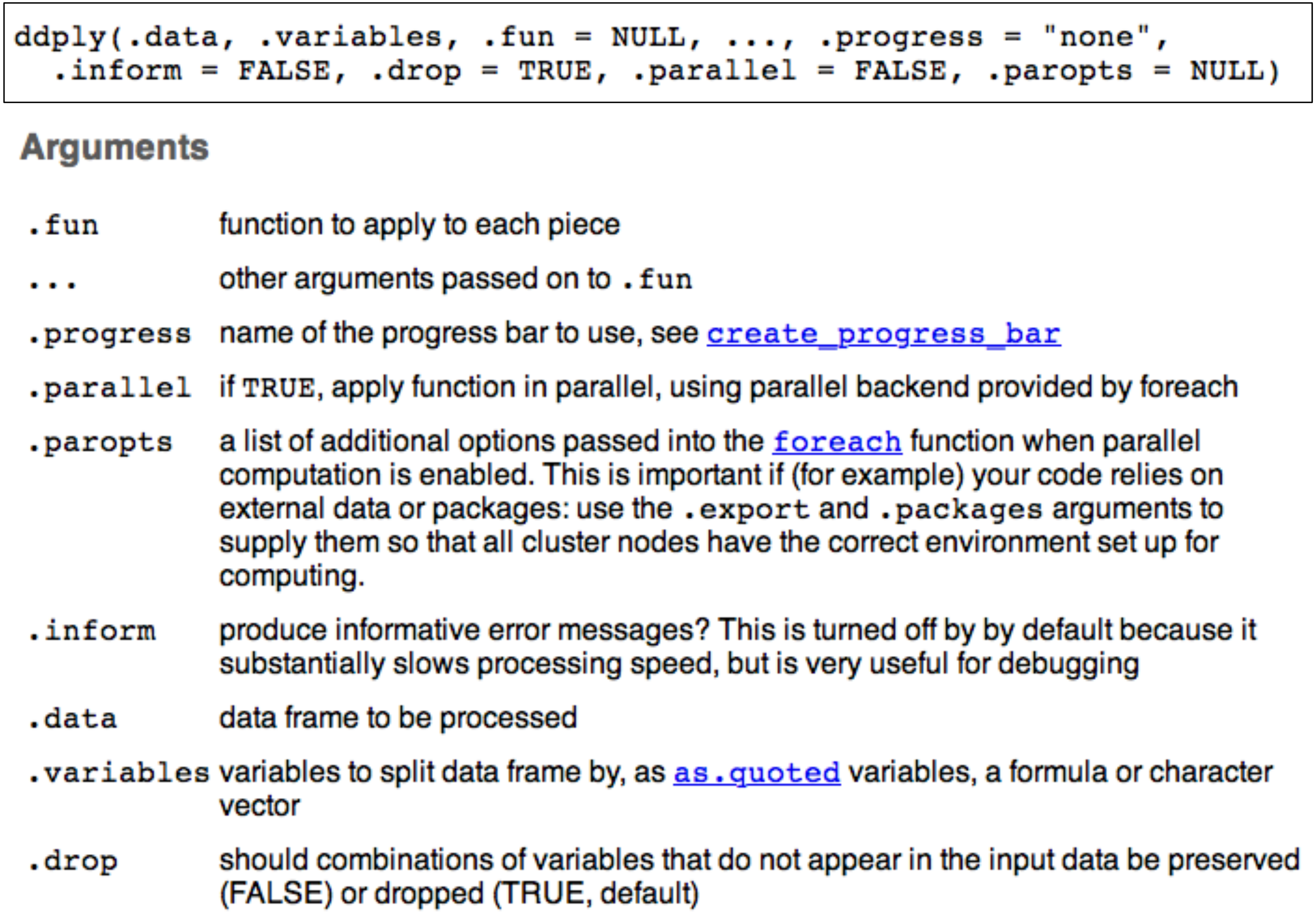
Split data frame, apply function, and return results in a data frame.

Description

For each subset of a data frame, apply function then combine results into a data frame.

Usage

```
ddply(.data, .variables, .fun = NULL, ..., .progress = "none",
  .inform = FALSE, .drop = TRUE, .parallel = FALSE, .paropts = NULL)
```



```
ddply(.data, .variables, .fun = NULL, ..., .progress = "none",
  .inform = FALSE, .drop = TRUE, .parallel = FALSE, .paropts = NULL)
```

Arguments

`.fun` function to apply to each piece

`...` other arguments passed on to `.fun`

`.progress` name of the progress bar to use, see `create_progress_bar`

`.parallel` if `TRUE`, apply function in parallel, using parallel backend provided by foreach

`.paropts` a list of additional options passed into the `foreach` function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the `.export` and `.packages` arguments to supply them so that all cluster nodes have the correct environment set up for computing.

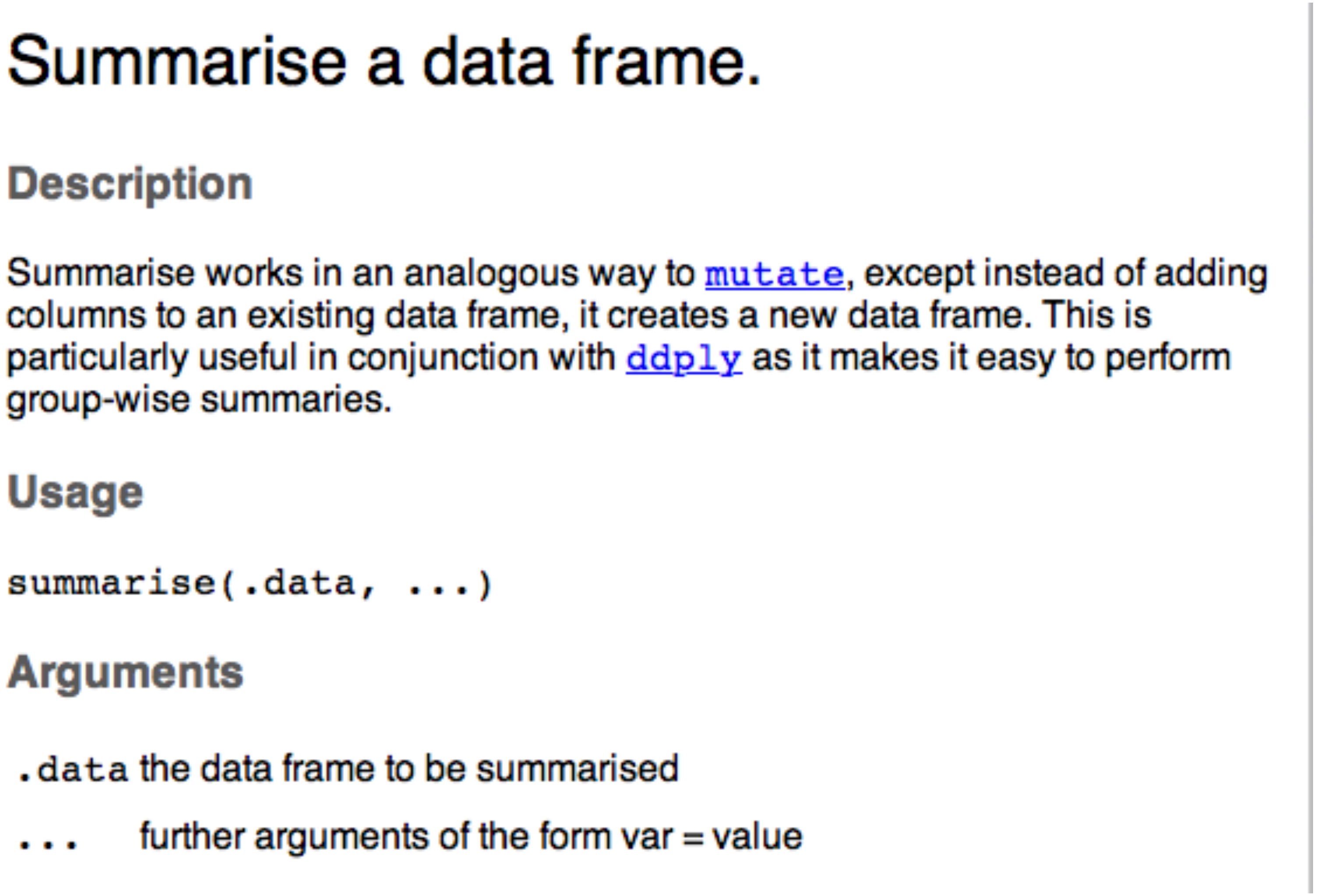
`.inform` produce informative error messages? This is turned off by by default because it substantially slows processing speed, but is very useful for debugging

`.data` data frame to be processed

`.variables` variables to split data frame by, as `as.quoted` variables, a formula or character vector

`.drop` should combinations of variables that do not appear in the input data be preserved (FALSE) or dropped (TRUE, default)

Summarise function



Summarise a data frame.

Description

Summarise works in an analogous way to mutate, except instead of adding columns to an existing data frame, it creates a new data frame. This is particularly useful in conjunction with ddply as it makes it easy to perform group-wise summaries.

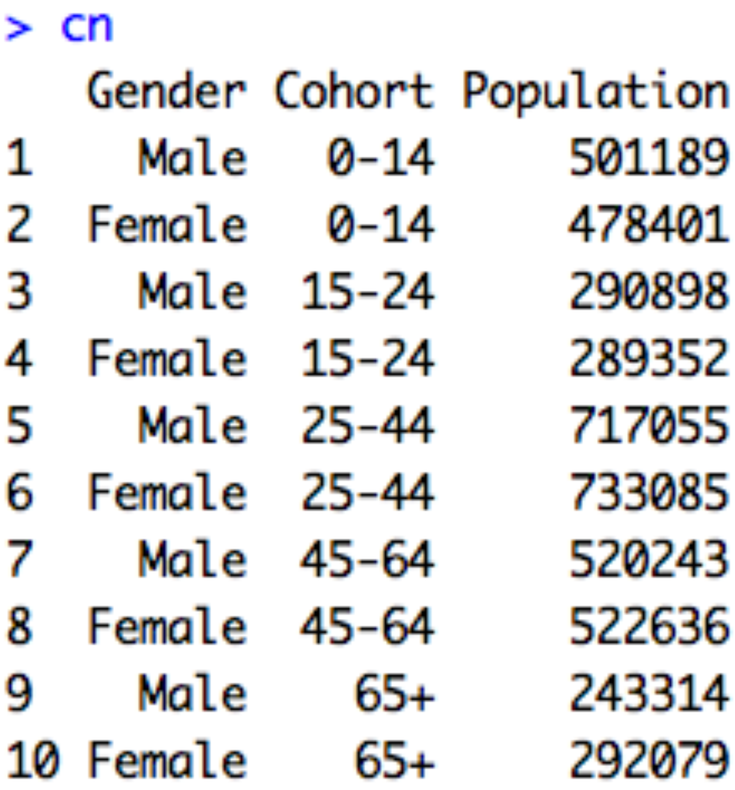
Usage

```
summarise(.data, ...)
```

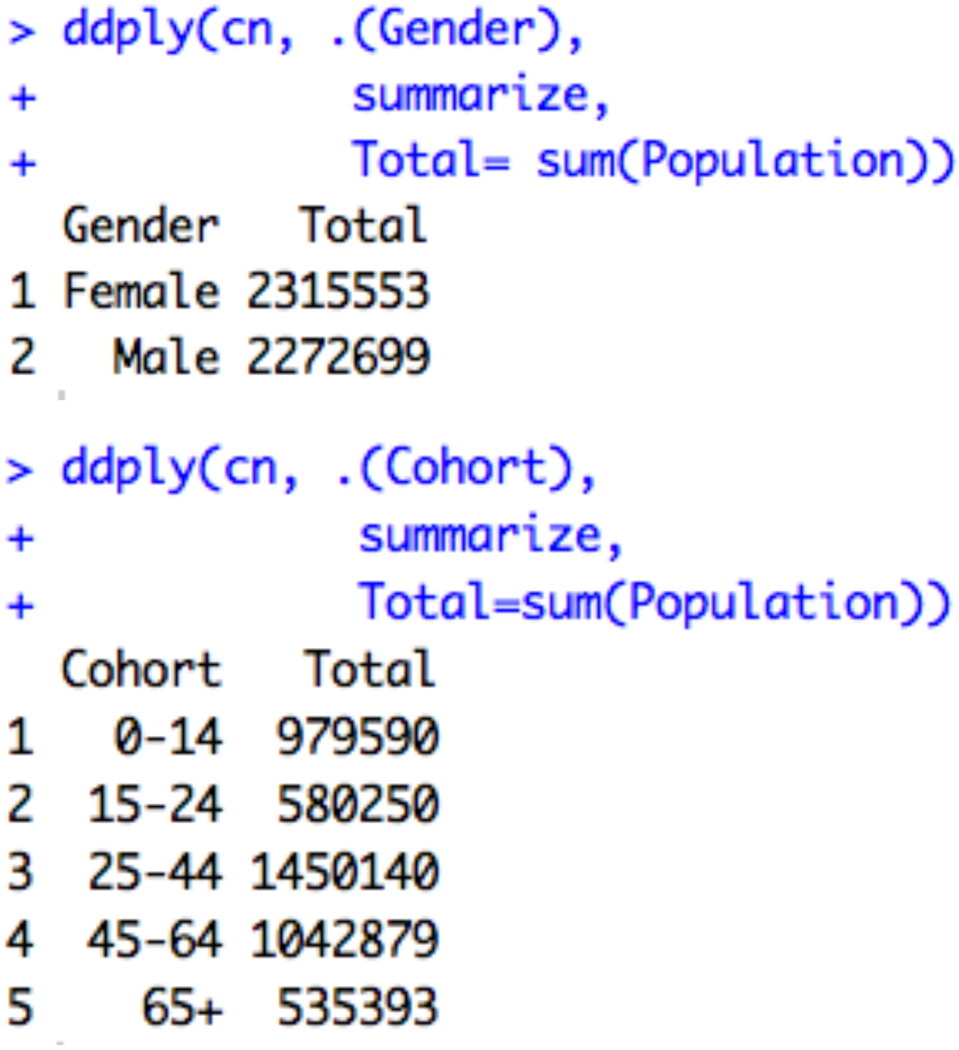
Arguments

`.data`	the data frame to be summarised
`...`	further arguments of the form var = value

Example



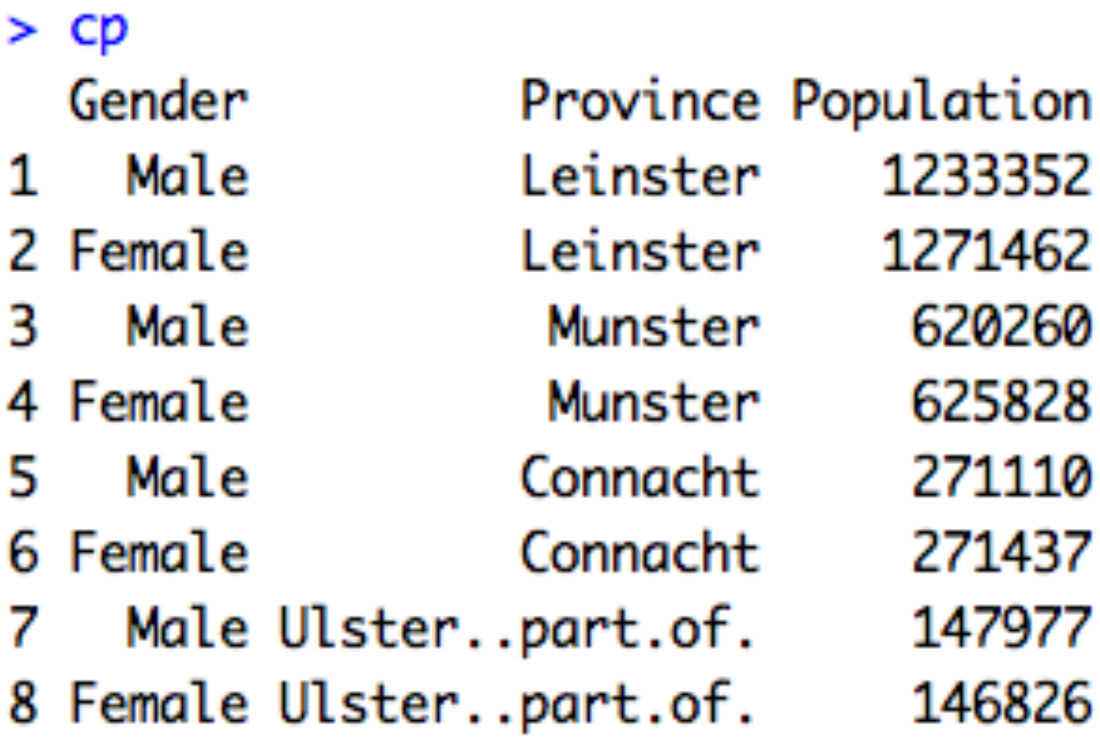
```
> cn
   Gender Cohort Population
1    Male   0-14     501189
2  Female   0-14     478401
3    Male  15-24     290898
4  Female  15-24     289352
5    Male  25-44     717055
6  Female  25-44     733085
7    Male  45-64     520243
8  Female  45-64     522636
9    Male    65+     243314
10 Female    65+     292079
```



```
> ddply(cn, .(Gender),
+           summarize,
+           Total= sum(Population))
  Gender   Total
1 Female 2315553
2   Male 2272699

> ddply(cn, .(Cohort),
+           summarize,
+           Total=sum(Population))
  Cohort   Total
1   0-14  979590
2  15-24  580250
3  25-44 1450140
4  45-64 1042879
5    65+  535393
```

Challenge 8.2

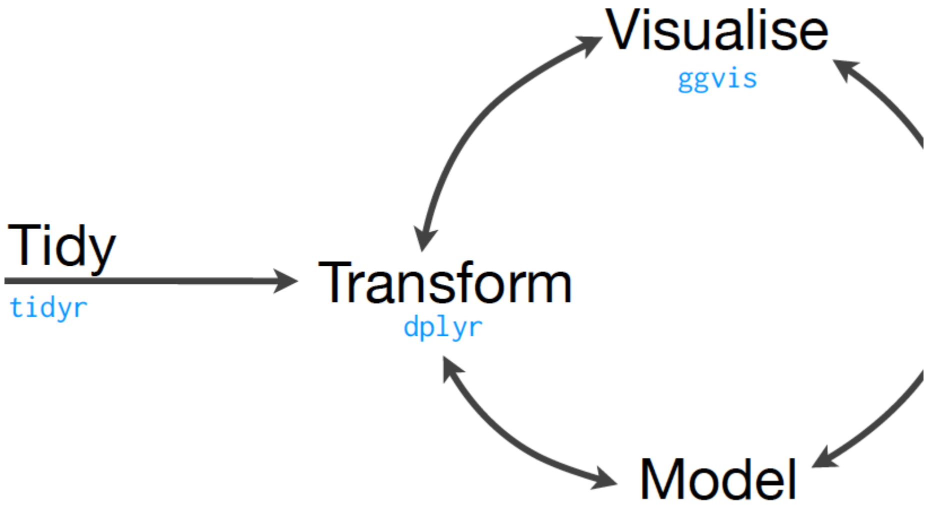
Use ddply() to get the total population by province.



```
> cp
  Gender         Province Population
1   Male         Leinster    1233352
2 Female         Leinster    1271462
3   Male          Munster     620260
4 Female          Munster     625828
5   Male         Connacht     271110
6 Female         Connacht     271437
7   Male Ulster..part.of.     147977
8 Female Ulster..part.of.     146826
```

Analysing Tidy Data

- Tidy data is only worthwhile if it makes analysis easier
- Tidy tools
  - Take tidy data sets as inputs and return tidy data sets as outputs
- Tools cover:
  - Data manipulation
  - Visualisation
  - Modelling



https://www.dropbox.com/sh/i8qnluwmuieicxc/AACsepZJvULCKkbIxK9KP-6Ea/dplyr-tutorial.pdf?dl=0

dplyr package

- https://rpubs.com/justmarkham/dplyr-tutorial
- https://www.youtube.com/watch?v=8SGif63VW6E
- https://www.youtube.com/watch?v=Ue08LVuk790
- http://renkun.me/pipeR-tutorial/Examples/dplyr.html

Data manipulation with **dplyr**

Hadley Wickham

@hadleywickham

Chief Scientist, RStudio

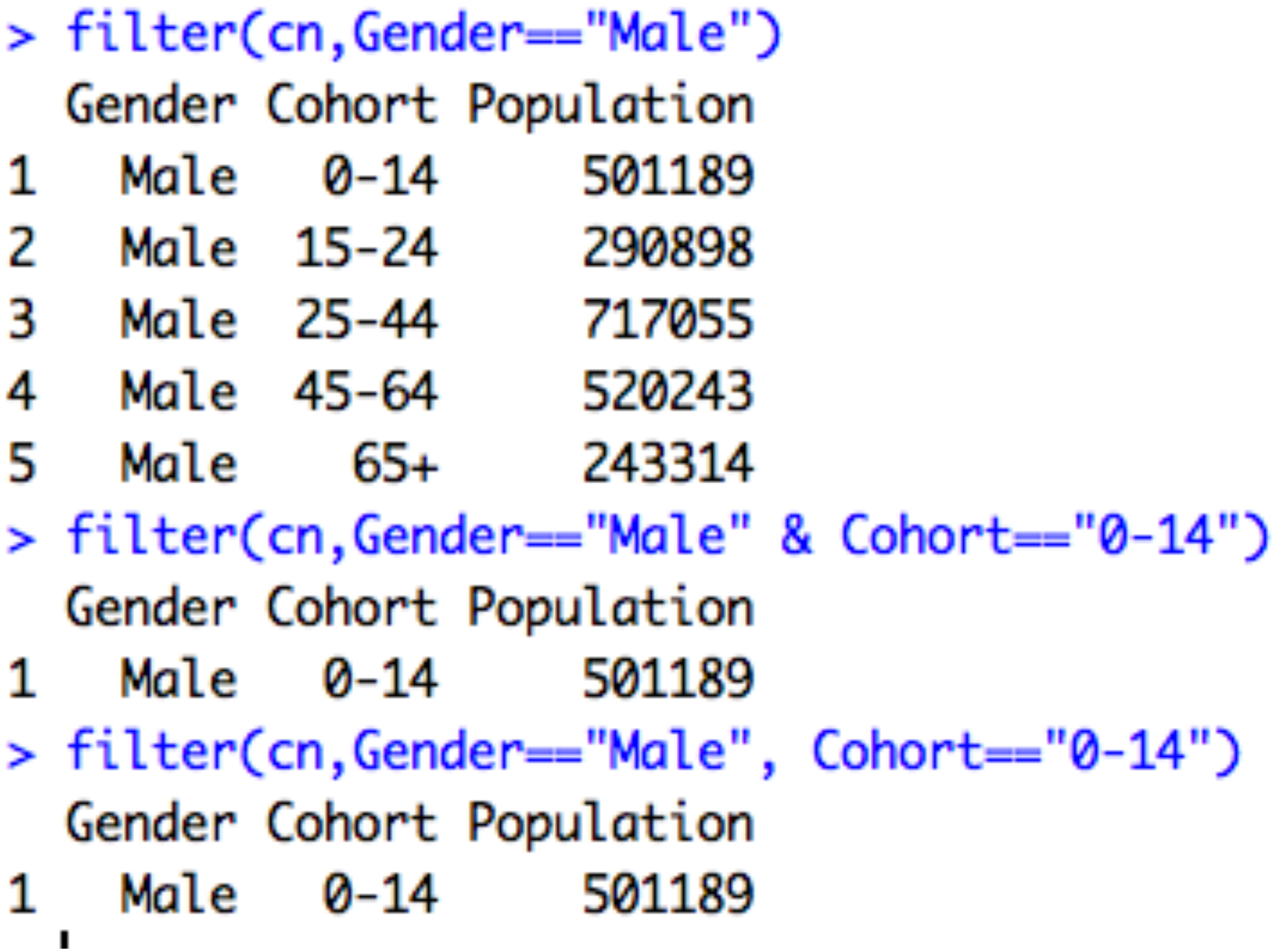
June 2014

Using dplyr

- Functions
  - **filter**: keep rows matching criteria
  - **select**: pick columns by name
  - **arrange**: reorder rows
  - **mutate**: add new variables
  - **summarise**: reduce variables to values
- Approach
  - First argument is a data frame
  - Subsequent arguments say what to do with data frame
  - Always return a data frame (Tidy Data approach)

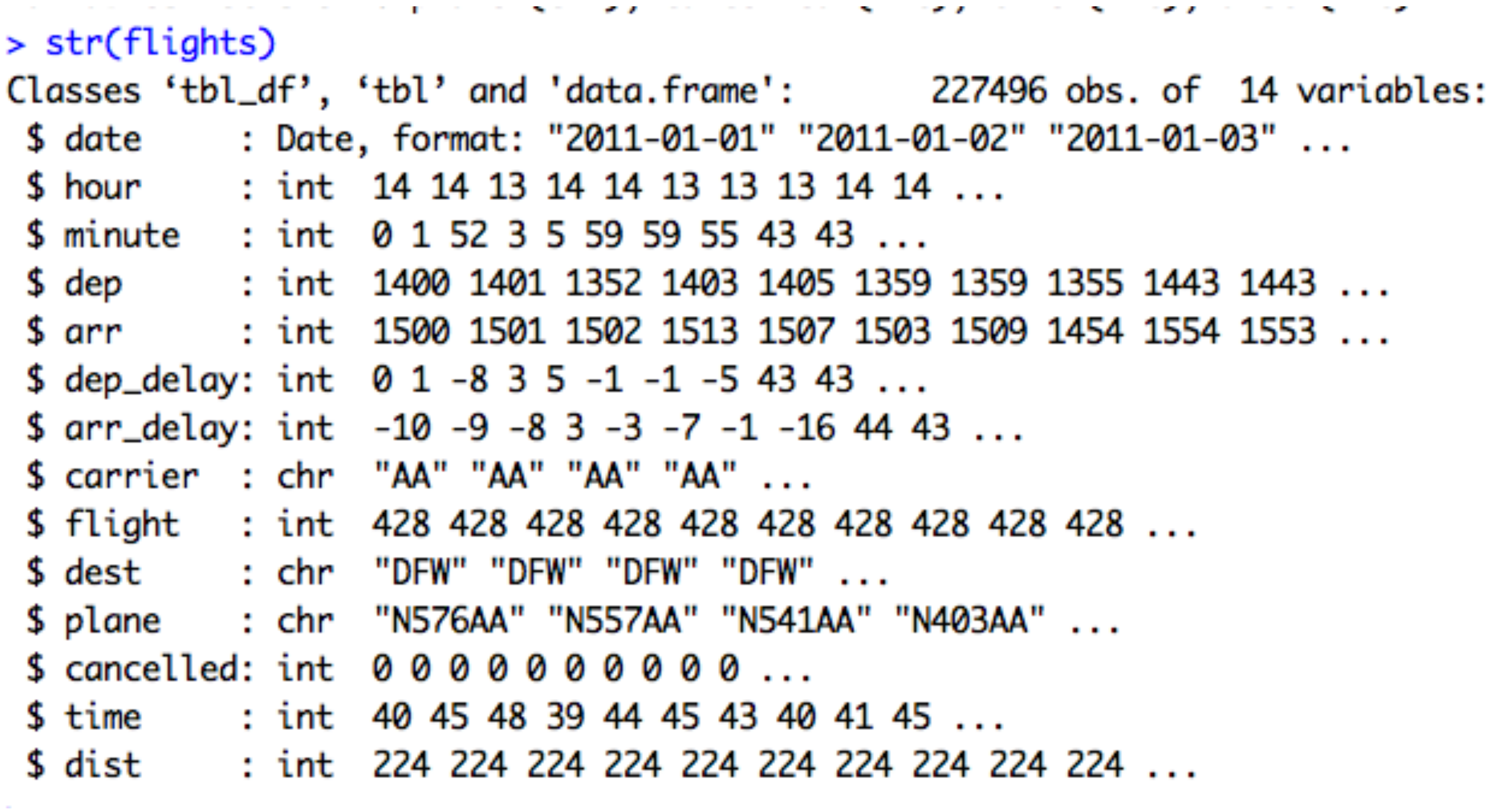
1. filter()

Keep rows by matching criteria

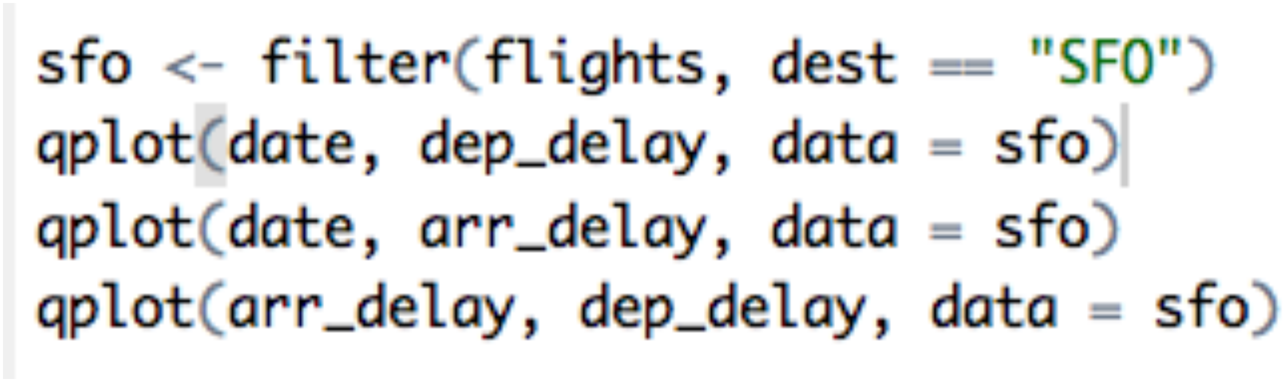


```
> filter(cn,Gender=="Male")
  Gender Cohort Population
1   Male   0-14     501189
2   Male  15-24     290898
3   Male  25-44     717055
4   Male  45-64     520243
5   Male    65+     243314
> filter(cn,Gender=="Male" & Cohort=="0-14")
  Gender Cohort Population
1   Male   0-14     501189
> filter(cn,Gender=="Male", Cohort=="0-14")
  Gender Cohort Population
1   Male   0-14     501189
```

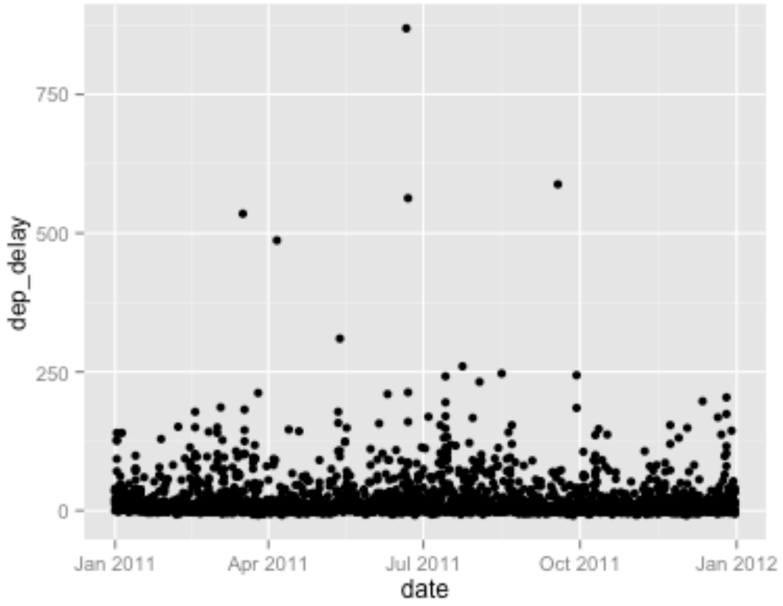
flights data

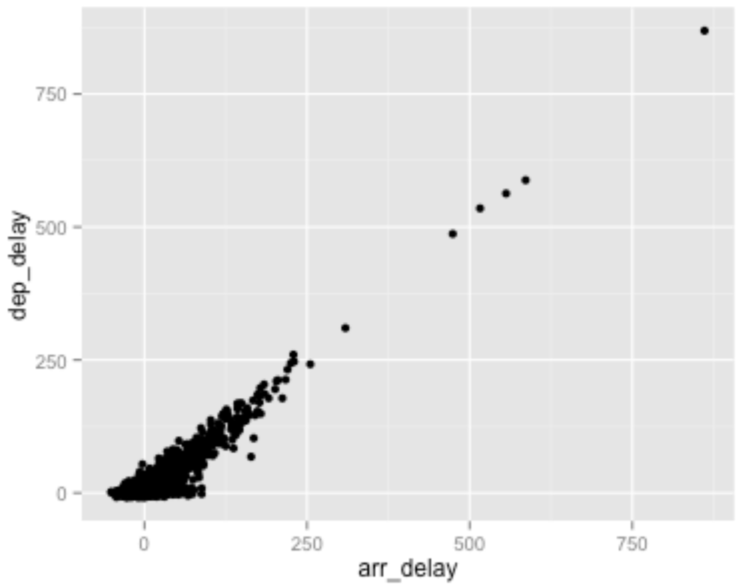


```
> str(flights)
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':      227496 obs. of  14 variables:
 $ date     : Date, format: "2011-01-01" "2011-01-02" "2011-01-03" ...
 $ hour     : int  14 14 13 14 14 13 13 13 14 14 ...
 $ minute   : int  0 1 52 3 5 59 59 55 43 43 ...
 $ dep      : int  1400 1401 1352 1403 1405 1359 1359 1355 1443 1443 ...
 $ arr      : int  1500 1501 1502 1513 1507 1503 1509 1454 1554 1553 ...
 $ dep_delay: int  0 1 -8 3 5 -1 -1 -5 43 43 ...
 $ arr_delay: int  -10 -9 -8 3 -3 -7 -1 -16 44 43 ...
 $ carrier  : chr  "AA" "AA" "AA" "AA" ...
 $ flight   : int  428 428 428 428 428 428 428 428 428 428 ...
 $ dest     : chr  "DFW" "DFW" "DFW" "DFW" ...
 $ plane    : chr  "N576AA" "N557AA" "N541AA" "N403AA" ...
 $ cancelled: int  0 0 0 0 0 0 0 0 0 0 ...
 $ time     : int  40 45 48 39 44 45 43 40 41 45 ...
 $ dist     : int  224 224 224 224 224 224 224 224 224 224 ...
```

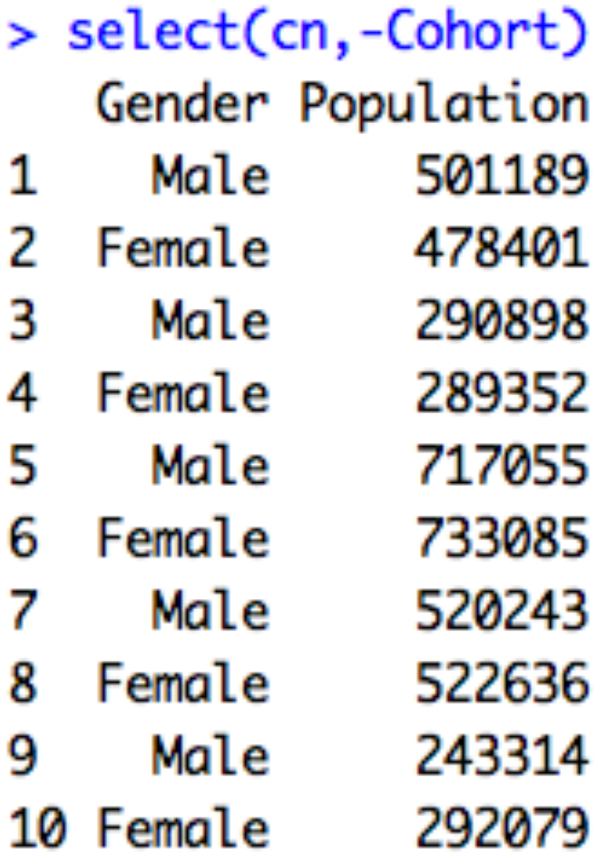


```
sfo <- filter(flights, dest == "SFO")
qplot(date, dep_delay, data = sfo)
qplot(date, arr_delay, data = sfo)
qplot(arr_delay, dep_delay, data = sfo)
```

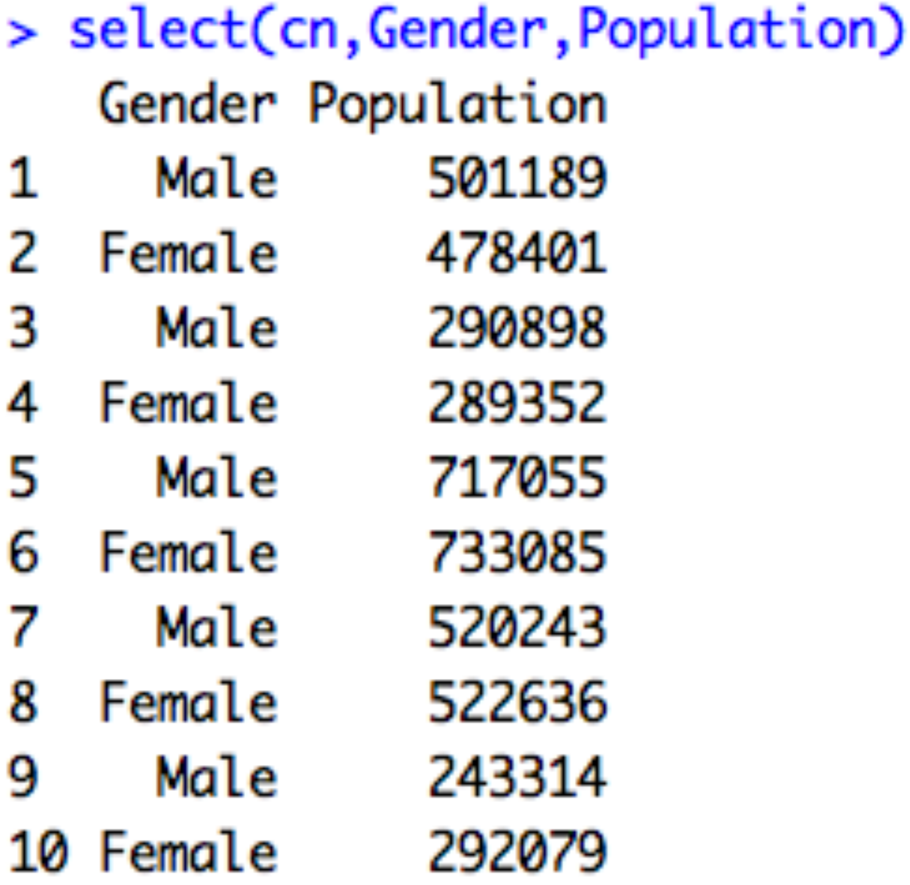




2. select()
pick columns by name

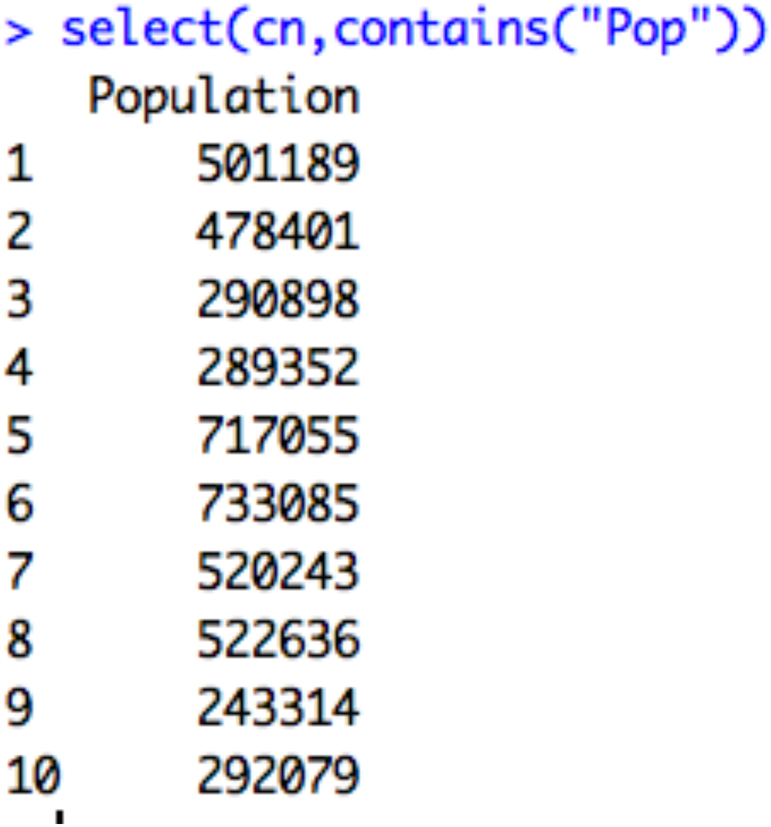


```
> select(cn,-Cohort)
   Gender Population
1    Male     501189
2  Female     478401
3    Male     290898
4  Female     289352
5    Male     717055
6  Female     733085
7    Male     520243
8  Female     522636
9    Male     243314
10 Female     292079
```

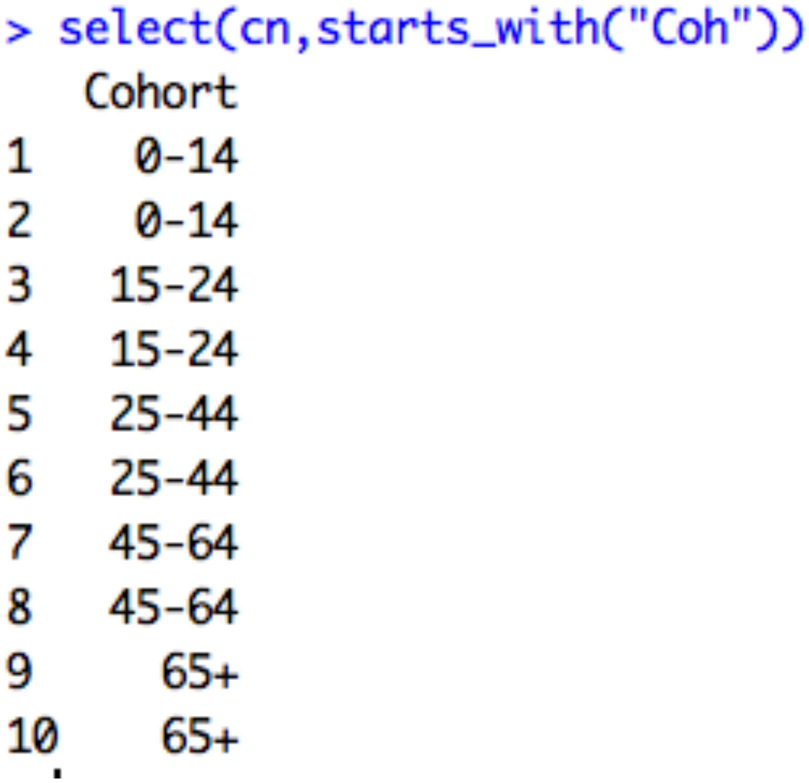


```
> select(cn,Gender,Population)
   Gender Population
1    Male     501189
2  Female     478401
3    Male     290898
4  Female     289352
5    Male     717055
6  Female     733085
7    Male     520243
8  Female     522636
9    Male     243314
10 Female     292079
```

select()
pick columns by name

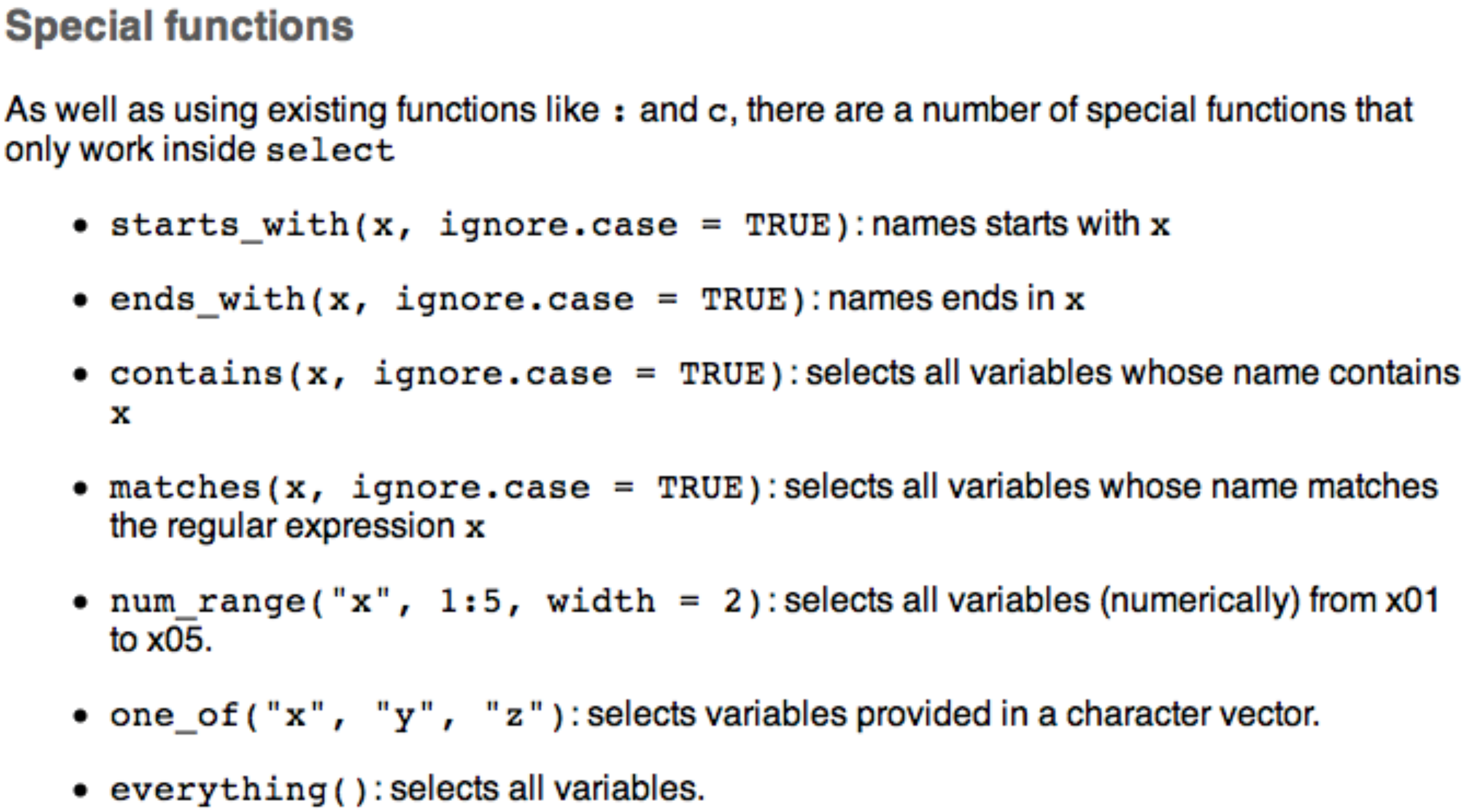


```
> select(cn,contains("Pop"))
   Population
1      501189
2      478401
3      290898
4      289352
5      717055
6      733085
7      520243
8      522636
9      243314
10     292079
```



```
> select(cn,starts_with("Coh"))
   Cohort
1    0-14
2    0-14
3   15-24
4   15-24
5   25-44
6   25-44
7   45-64
8   45-64
9     65+
10    65+
```

Special functions with select()



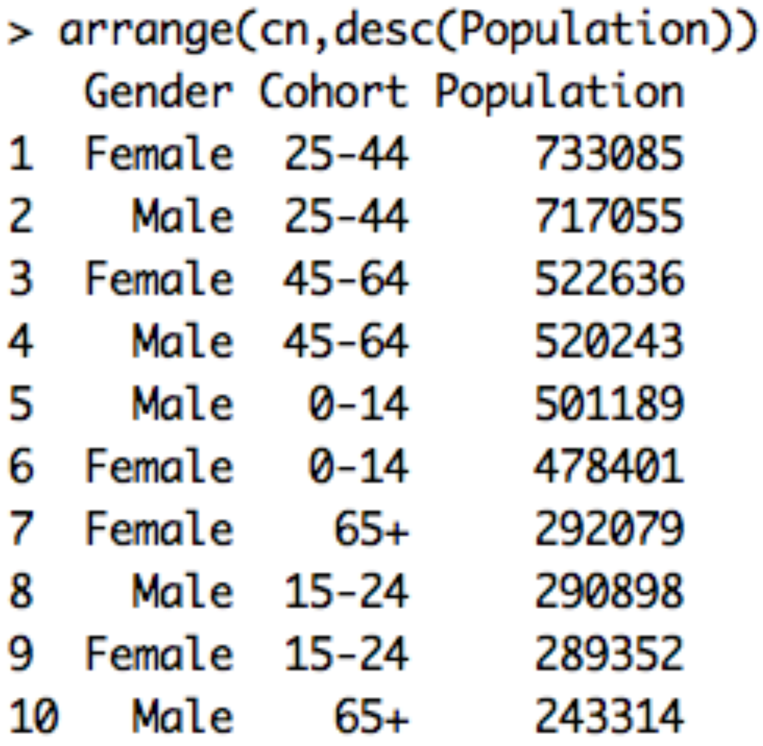
Special functions

As well as using existing functions like `:` and `c`, there are a number of special functions that only work inside `select`

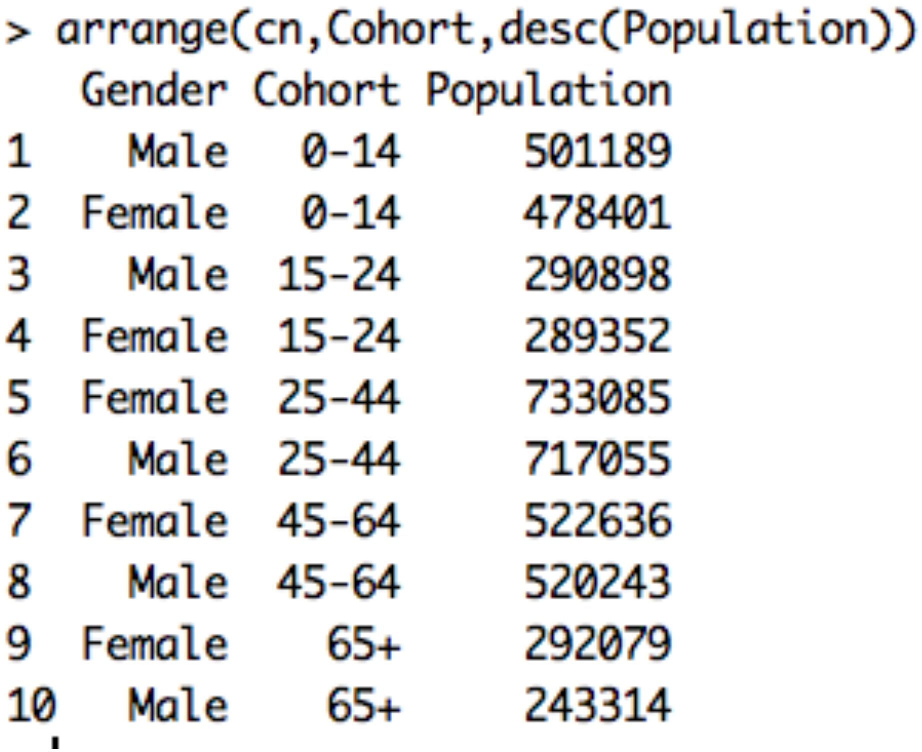
- `starts_with(x, ignore.case = TRUE)`: names starts with `x`
- `ends_with(x, ignore.case = TRUE)`: names ends in `x`
- `contains(x, ignore.case = TRUE)`: selects all variables whose name contains `x`
- `matches(x, ignore.case = TRUE)`: selects all variables whose name matches the regular expression `x`
- `num_range("x", 1:5, width = 2)`: selects all variables (numerically) from x01 to x05.
- `one_of("x", "y", "z")`: selects variables provided in a character vector.
- `everything()`: selects all variables.

3. arrange()

reorder rows

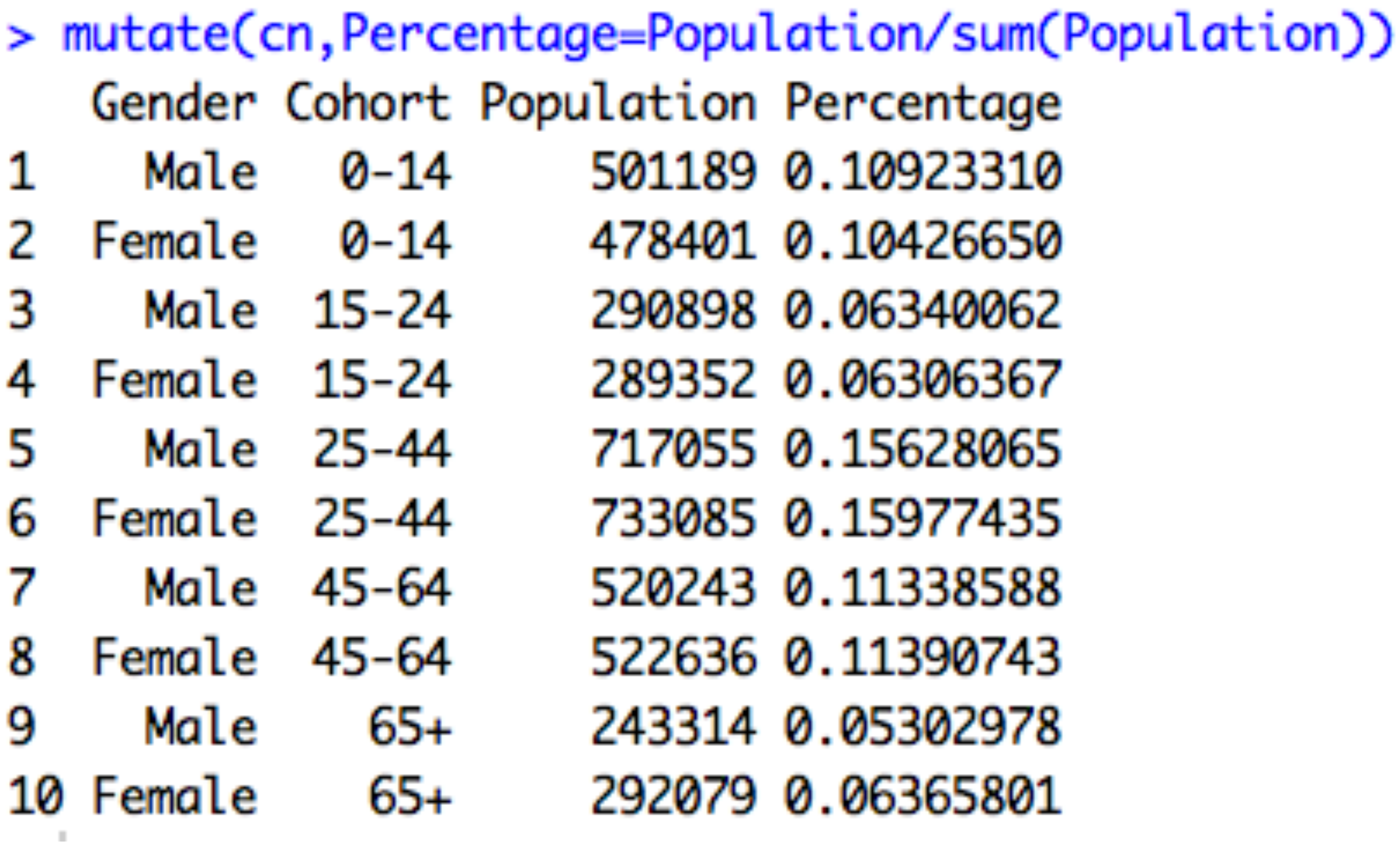


```
> arrange(cn,desc(Population))
   Gender Cohort Population
1  Female  25-44     733085
2    Male  25-44     717055
3  Female  45-64     522636
4    Male  45-64     520243
5    Male   0-14     501189
6  Female   0-14     478401
7  Female    65+     292079
8    Male  15-24     290898
9  Female  15-24     289352
10   Male    65+     243314
```



```
> arrange(cn,Cohort,desc(Population))
   Gender Cohort Population
1    Male   0-14     501189
2  Female   0-14     478401
3    Male  15-24     290898
4  Female  15-24     289352
5  Female  25-44     733085
6    Male  25-44     717055
7  Female  45-64     522636
8    Male  45-64     520243
9  Female    65+     292079
10   Male    65+     243314
```

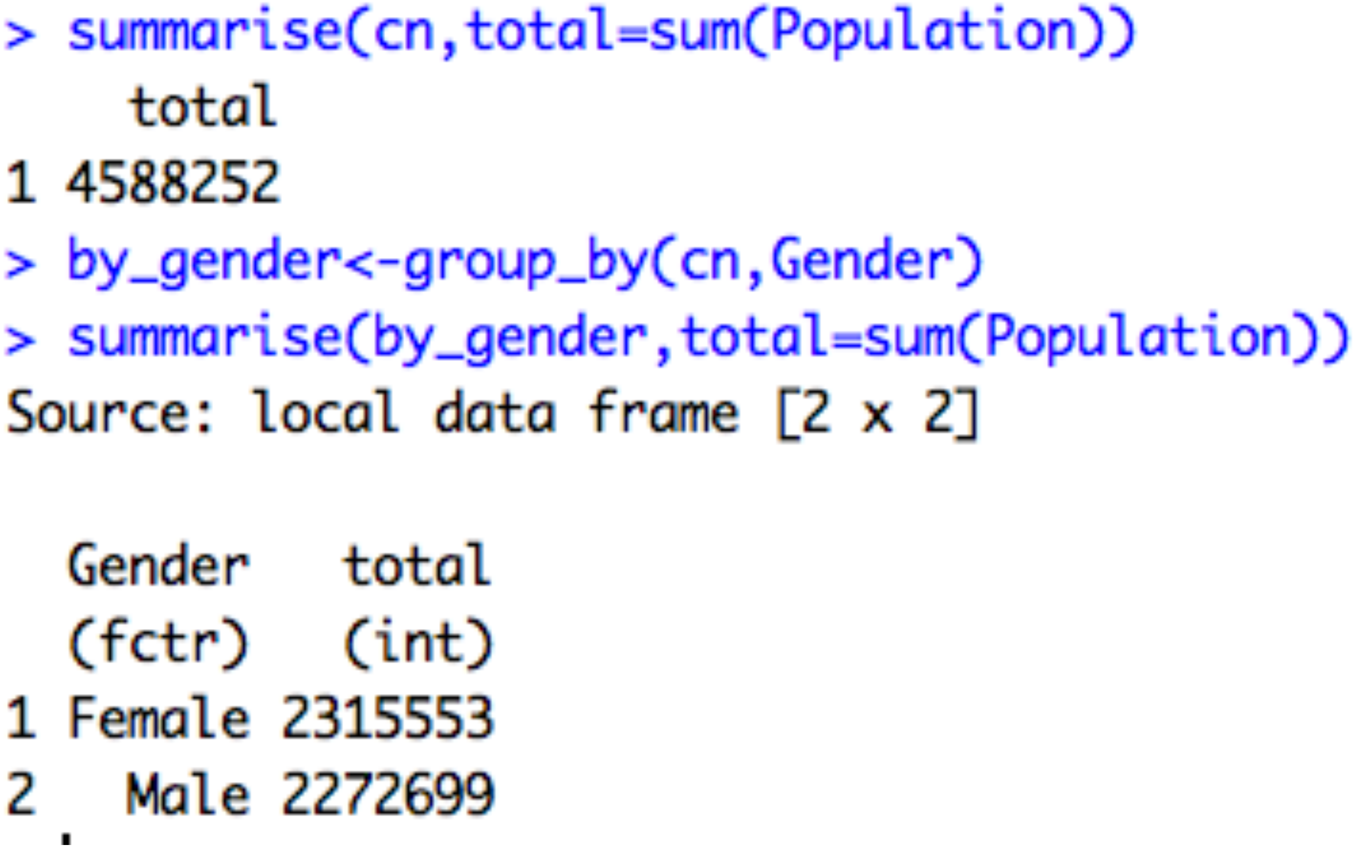
4. mutate()
Add new variables



```
> mutate(cn,Percentage=Population/sum(Population))
   Gender Cohort Population Percentage
1    Male   0-14     501189 0.10923310
2  Female   0-14     478401 0.10426650
3    Male  15-24     290898 0.06340062
4  Female  15-24     289352 0.06306367
5    Male  25-44     717055 0.15628065
6  Female  25-44     733085 0.15977435
7    Male  45-64     520243 0.11338588
8  Female  45-64     522636 0.11390743
9    Male    65+     243314 0.05302978
10 Female    65+     292079 0.06365801
```

5. summarise()

Reduce variables to values



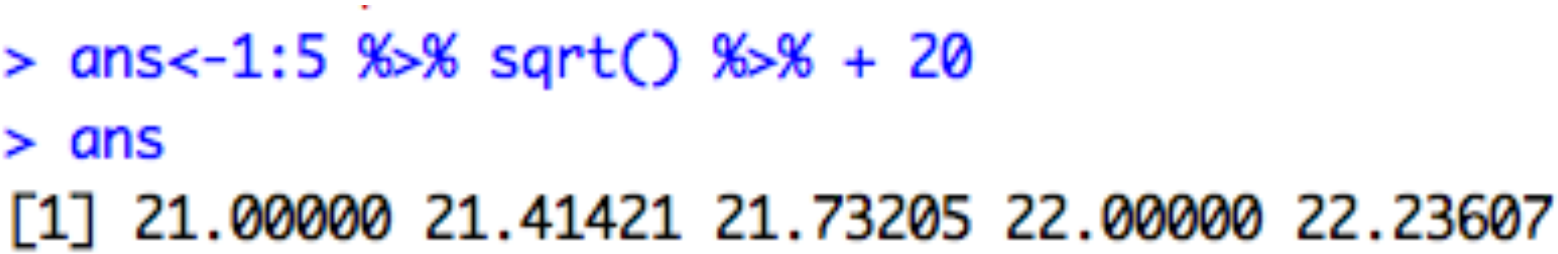
```
> summarise(cn,total=sum(Population))
    total
1 4588252
> by_gender<-group_by(cn,Gender)
> summarise(by_gender,total=sum(Population))
Source: local data frame [2 x 2]

  Gender   total
  (fctr)   (int)
1 Female 2315553
2   Male 2272699
```

Summary Functions

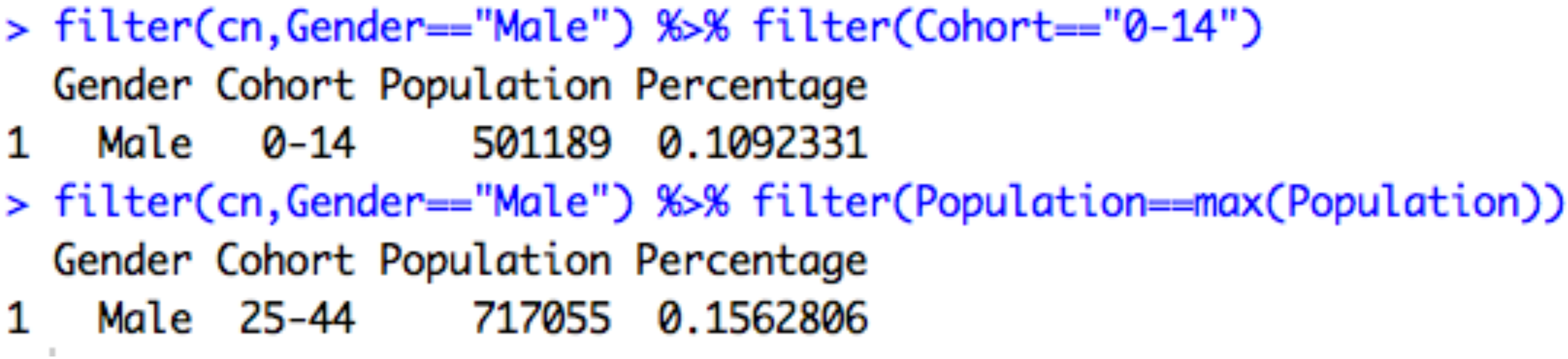
- min(x), median(x), max(x),
- quantile(x, p)
- n(), n_distinct(), sum(x), mean(x)
- sum(x > 10), mean(x > 10)
- sd(x), var(x), iqr(x), mad(x)

Pipe Operator in R %>%

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



```
> ans<-1:5 %>% sqrt() %>% + 20
> ans
[1] 21.00000 21.41421 21.73205 22.00000 22.23607
```

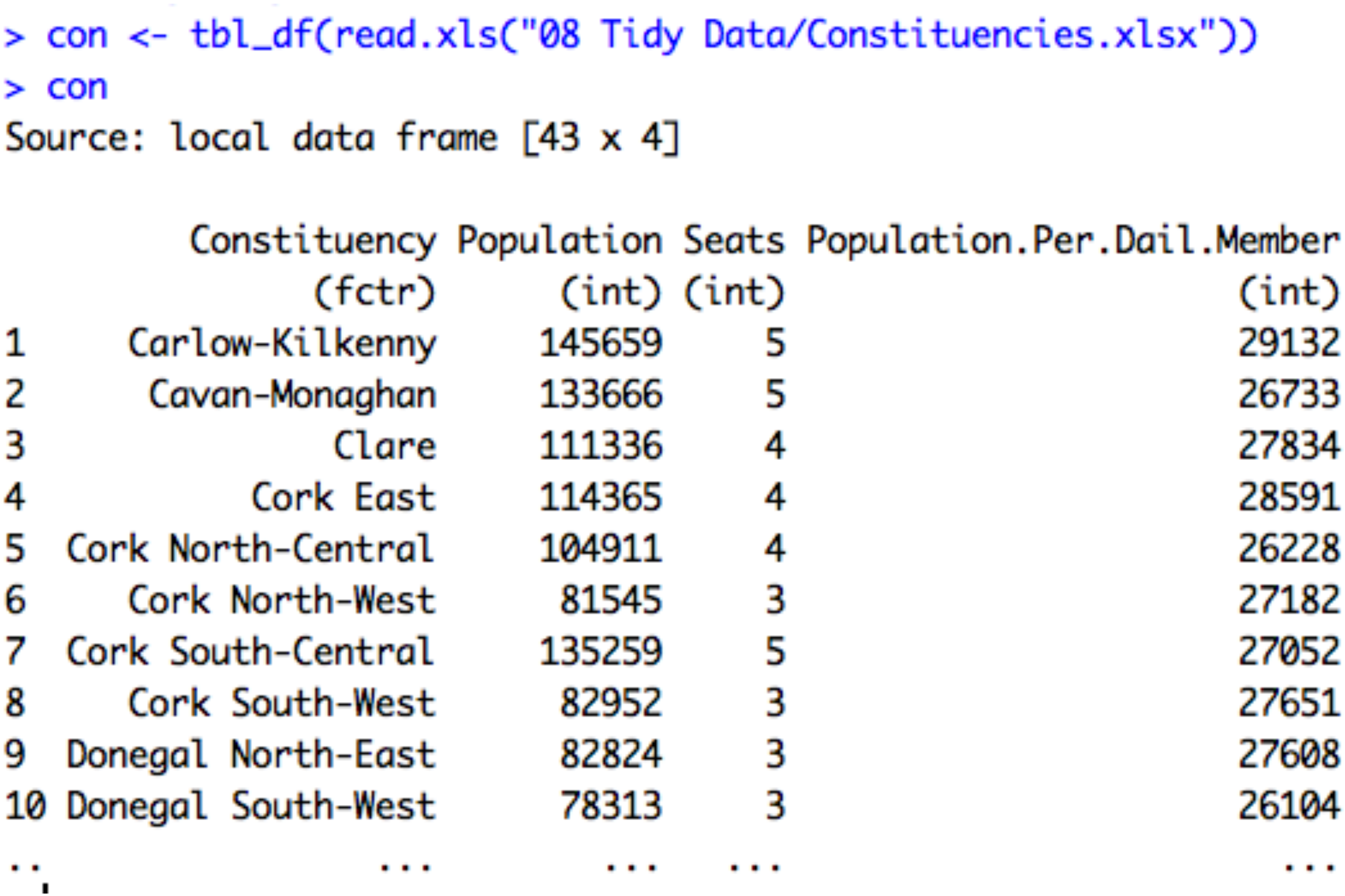


```
> filter(cn,Gender=="Male") %>% filter(Cohort=="0-14")
  Gender Cohort Population Percentage
1   Male   0-14     501189  0.1092331
> filter(cn,Gender=="Male") %>% filter(Population==max(Population))
  Gender Cohort Population Percentage
1   Male  25-44     717055  0.1562806
```

Challenge 8.3

Constituency	Population	Seats	Population Per Dail Member
Carlow-Kilkenny	145,659	5	29,132
Cavan-Monaghan	133,666	5	26,733
Clare	111,336	4	27,834
Cork East	114,365	4	28,591
Cork North-Central	104,911	4	26,228
Cork North-West	81,545	3	27,182
Cork South-Central	135,259	5	27,052
Cork South-West	82,952	3	27,651
Donegal North-East	82,824	3	27,608
Donegal South-West	78,313	3	26,104
Dublin Central	113,792	4	28,448
Dublin Mid-West	110,427	4	27,607
Dublin North	114,322	4	28,580
Dublin North-Central	74,501	3	24,834
Dublin North-East	81,560	3	27,187
Dublin North-West	79,028	3	26,343
Dublin South	140,543	5	28,109
Dublin South-Central	127,223	5	25,445
Dublin South-East	103,833	4	25,958
Dublin South-West	105,597	4	26,399
Dublin West	117,214	4	29,304
Dún Laoghaire	105,029	4	26,257
Galway East	110,085	4	27,521
Galway West	140,568	5	28,114
Kerry North-West Limerick	80,883	3	26,961
Kerry South	77,971	3	25,990
Kildare North	120,048	4	30,012
Kildare South	90,264	3	30,088
Laois-Offaly	152,825	5	30,565
Limerick City	102,638	4	25,660
Limerick	81,679	3	27,226
Longford-Westmeath	116,802	4	29,200
Louth	143,272	5	28,654
Mayo	130,638	5	26,128
Meath East	86,572	3	28,857
Meath West	85,550	3	28,517
Roscommon - South Leitrim	80,973	3	26,991
Sligo-North Leitrim	80,283	3	26,761
Tipperary North	85,024	3	28,341
Tipperary South	79,748	3	26,583
Waterford	112,198	4	28,050
Wexford	145,320	5	29,064
Wicklow	141,012	5	28,202

Using tbl_df()

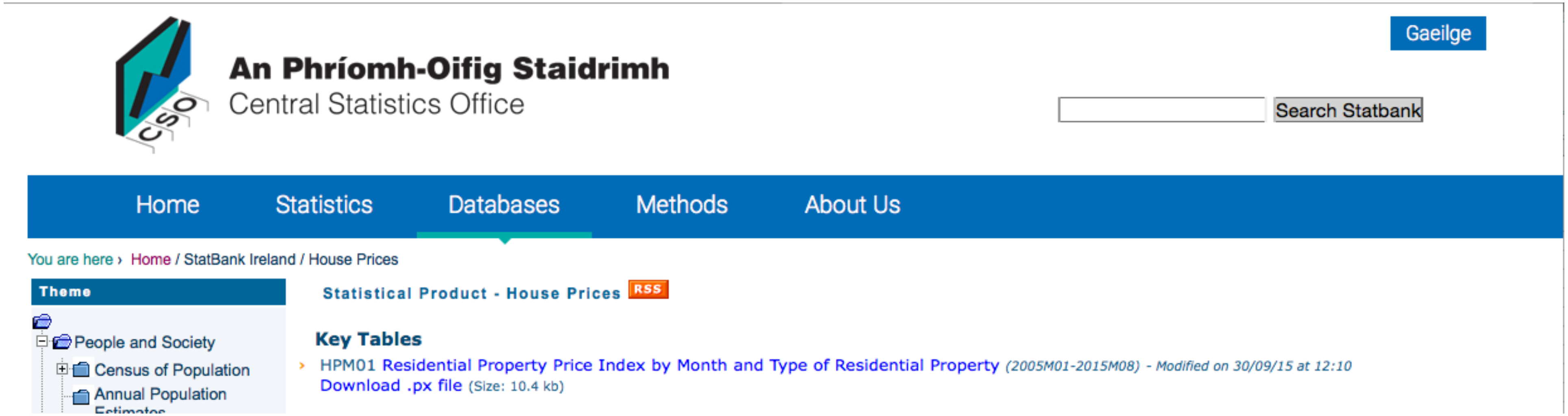


```
> con <- tbl_df(read.xls("08 Tidy Data/Constituencies.xlsx"))
> con
Source: local data frame [43 x 4]

         Constituency Population Seats Population.Per.Dail.Member
               (fctr)      (int) (int)                      (int)
1     Carlow-Kilkenny     145659     5                      29132
2      Cavan-Monaghan     133666     5                      26733
3               Clare     111336     4                      27834
4          Cork East      114365     4                      28591
5  Cork North-Central     104911     4                      26228
6     Cork North-West      81545     3                      27182
7  Cork South-Central     135259     5                      27052
8     Cork South-West      82952     3                      27651
9  Donegal North-East      82824     3                      27608
10 Donegal South-West      78313     3                      26104
..                ...        ...   ...                        ...
```

Queries

1. List all the three seat constituencies
2. List all constituencies in Dublin
3. Show table without Population Per Dail Member
4. Add in the proportion of population for each constituency
5. Arrange data set by population and number of seats (desc)
6. Show total population covered by the different constituency types
7. Show the percentages for query (6).

dplyr assignment



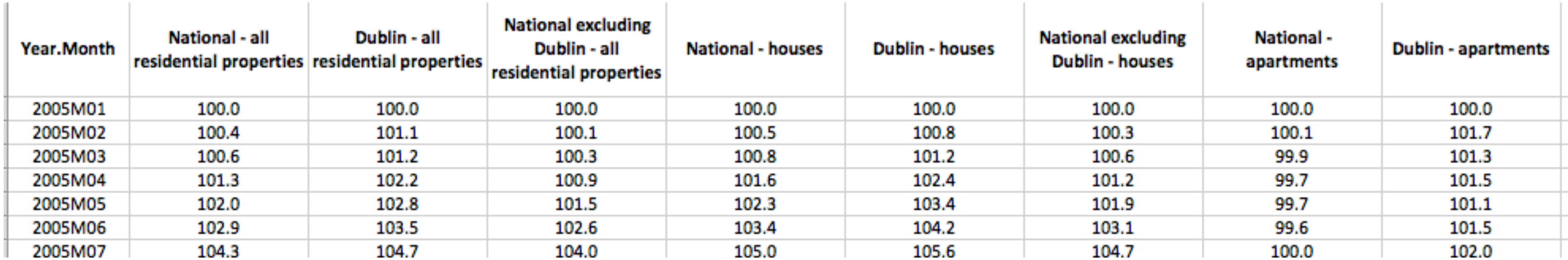
An Phríomh-Oifig Staidrimh
Central Statistics Office

You are here › Home / StatBank Ireland / House Prices

Statistical Product - House Prices RSS

Key Tables

› HPM01 Residential Property Price Index by Month and Type of Residential Property *(2005M01-2015M08) - Modified on 30/09/15 at 12:10*
Download .px file (Size: 10.4 kb)



Year.Month	National - all residential properties	Dublin - all residential properties	National excluding Dublin - all residential properties	National - houses	Dublin - houses	National excluding Dublin - houses	National - apartments	Dublin - apartments
2005M01	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
2005M02	100.4	101.1	100.1	100.5	100.8	100.3	100.1	101.7
2005M03	100.6	101.2	100.3	100.8	101.2	100.6	99.9	101.3
2005M04	101.3	102.2	100.9	101.6	102.4	101.2	99.7	101.5
2005M05	102.0	102.8	101.5	102.3	103.4	101.9	99.7	101.1
2005M06	102.9	103.5	102.6	103.4	104.2	103.1	99.6	101.5
2005M07	104.3	104.7	104.0	105.0	105.6	104.7	100.0	102.0