

# Data Manipulation

## Contents

<b>Data Manipulation</b>	<b>1</b>
Five basic functions in data handling . . . . .	1
Grouping and Combining . . . . .	4
Base subsetting . . . . .	5
Base Reference . . . . .	6

## Data Manipulation

Report exemplifying the use of dplyr in data handling on the example of **dsL**.

### Five basic functions in data handling

For a more detailed discussion of basic verbs and operations consult the [R-Studio guide](#) or internal [vignette](#)

```
vignette("introduction",package="dplyr")
```

The following is a brief demonstration of dplyr syntax using **dsL** dataset as an example. I attach prefix dplyr:: to avoid possible conflicts with plyr package on which ggplot2 package relies. I recommend such practice in all dplyr expressions in sharable publications.

#### select()

selects variables into a smaller data set

```
ds<-dsL
dim(ds)
```

```
[1] 134745      60
```

```
ds<- dplyr::select(ds,id,year, byear, attend, attendF)
head(ds,13)
```

	id	year	byear	attend	attendF
1	1	1997	1981	NA	<NA>
2	1	1998	1981	NA	<NA>
3	1	1999	1981	NA	<NA>
4	1	2000	1981	1	Never
5	1	2001	1981	6	About once/week
6	1	2002	1981	2	Once or Twice
7	1	2003	1981	1	Never
8	1	2004	1981	1	Never
9	1	2005	1981	1	Never

10	1	2006	1981	1	Never
11	1	2007	1981	1	Never
12	1	2008	1981	1	Never
13	1	2009	1981	1	Never

```
dim(ds)
```

```
[1] 134745      5
```

### filter()

Removes observations that do not meet criteria. The following code selects observation based on the type of sample

	sample	sampleF
1	1	Cross-Sectional
2	0	Oversample

and only between years 2000 and 2011, as only during those years the outcome of interest attend was recorded.

```
ds<- dplyr::filter(dsL,sample==1, year %in% c(2000:2011))
ds<- dplyr::select(ds,id, year, attend, attendF)
head(ds,13)
```

	id	year	attend	attendF
1	1	2000	1	Never
2	1	2001	6	About once/week
3	1	2002	2	Once or Twice
4	1	2003	1	Never
5	1	2004	1	Never
6	1	2005	1	Never
7	1	2006	1	Never
8	1	2007	1	Never
9	1	2008	1	Never
10	1	2009	1	Never
11	1	2010	1	Never
12	1	2011	1	Never
13	2	2000	2	Once or Twice

### arrange()

Sorts observations

```
ds<- dplyr::filter(dsL,sample==1, year %in% c(2000:2011))
ds<- dplyr::select(ds,id, year, attend)
ds<- dplyr::arrange(ds, year, desc(id))
head(ds,13)
```

	id	year	attend
1	9022	2000	1
2	9021	2000	2

3	9020	2000	2
4	9018	2000	4
5	9017	2000	6
6	9012	2000	5
7	9011	2000	6
8	9010	2000	1
9	9009	2000	2
10	9008	2000	6
11	8992	2000	NA
12	8991	2000	3
13	8987	2000	6

```
ds<- arrange(ds, id, year)
head(ds, 13)
```

	id	year	attend
1	1	2000	1
2	1	2001	6
3	1	2002	2
4	1	2003	1
5	1	2004	1
6	1	2005	1
7	1	2006	1
8	1	2007	1
9	1	2008	1
10	1	2009	1
11	1	2010	1
12	1	2011	1
13	2	2000	2

### mutate()

Creates additional variables from the values of existing.

```
ds<- dplyr::filter(dsL,sample==1, year %in% c(2000:2011))
ds<- dplyr::select(ds,id, byear, year, attend)
ds<- dplyr::mutate(ds,
  age = year-byear,
  timec = year-2000,
  linear= timec,
  quadratic= linear^2,
  cubic= linear^3)
head(ds,13)
```

	id	byear	year	attend	age	timec	linear	quadratic	cubic
1	1	1981	2000	1	19	0	0	0	0
2	1	1981	2001	6	20	1	1	1	1
3	1	1981	2002	2	21	2	2	4	8
4	1	1981	2003	1	22	3	3	9	27
5	1	1981	2004	1	23	4	4	16	64
6	1	1981	2005	1	24	5	5	25	125
7	1	1981	2006	1	25	6	6	36	216

8	1	1981	2007	1	26	7	7	49	343
9	1	1981	2008	1	27	8	8	64	512
10	1	1981	2009	1	28	9	9	81	729
11	1	1981	2010	1	29	10	10	100	1000
12	1	1981	2011	1	30	11	11	121	1331
13	2	1982	2000	2	18	0	0	0	0

## summarize()

collapses data into a single value computed according to the aggregate functions.

```
require(dplyr)
ds<- dplyr::filter(dsL,sample==1)
ds<- dplyr::summarize(ds, N= n_distinct(id))
ds
```

```
      N
1 6747
```

Other functions one could use with summarize() include:

From base + min()  
 + max()  
 + mean()  
 + sum()  
 + sd()  
 + median()  
 + IQR()

Native to dplyr + n() - number of observations in the current group + n\_distinct(x) - count the number of unique values in x. + first(x) - similar to x[1] + control over NA + last(x) - similar to x[length(x)] + control over NA + nth(x, n) - similar to x[n] + control over NA

## Grouping and Combining

The function group\_by() is used to identify groups in split-apply-combine (SAC) procedure: it splits the initial data into smaller datasets (according to all possible interactions between the levels of supplied variables). It is these smaller datasets that summarize() will individually collapse into a single computed value according to its formula.

```
ds<- dplyr::filter(dsL,sample==1, year %in% c(2000:2011))
ds<- dplyr::select(ds,id, year, attendF)

s <- dplyr::group_by(ds, year,attendF)
s <- dplyr::summarise(s, count = n())
s <- dplyr::mutate(s, total = sum(count),
                  percent= count/total)
head(s,10)
```

Source: local data frame [10 x 5]

Groups: year

year	attendF	count	total	percent
------	---------	-------	-------	---------

1	2000	Never	1580	6747	0.234178
2	2000	Once or Twice	1304	6747	0.193271
3	2000	Less than once/month	775	6747	0.114866
4	2000	About once/month	362	6747	0.053653
5	2000	About twice/month	393	6747	0.058248
6	2000	About once/week	1101	6747	0.163184
7	2000	Several times/week	463	6747	0.068623
8	2000	Everyday	36	6747	0.005336
9	2000	NA	733	6747	0.108641
10	2001	Never	1626	6747	0.240996

The same result can be achieved with a more elegant syntax that relies on `%>%` operator, in which `x %>% f(y)` turns into `f(x, y)`. Alternatively, one can use `%.%` for identical results.

```
ds<-dsL %>%
  dplyr::filter(sample==1, year %in% c(2000:2011)) %>%
  dplyr::select(id, year, attendF) %>%
  dplyr::group_by(year,attendF) %>%
  dplyr::summarise(count = n()) %>%
  dplyr::mutate(total = sum(count),
                percent= count/total)
head(ds,10)
```

Source: local data frame [10 x 5]

Groups: year

	year	attendF	count	total	percent
1	2000	Never	1580	6747	0.234178
2	2000	Once or Twice	1304	6747	0.193271
3	2000	Less than once/month	775	6747	0.114866
4	2000	About once/month	362	6747	0.053653
5	2000	About twice/month	393	6747	0.058248
6	2000	About once/week	1101	6747	0.163184
7	2000	Several times/week	463	6747	0.068623
8	2000	Everyday	36	6747	0.005336
9	2000	NA	733	6747	0.108641
10	2001	Never	1626	6747	0.240996

To verify that this is what we wanted to achieve:

```
dplyr::summarize( filter(s, year==2000), should.be.one=sum(percent))
```

Source: local data frame [1 x 2]

	year	should.be.one
1	2000	1

## Base subsetting

Generally, we can compose any desired dataset by using matrix calls. The general formula is of the form: `ds[ rowCond , colCond ]`, where `ds` is a dataframe, and `rowCond` and `colCond` are conditions for including rows and columns of the new dataset, respectively. One can also call a variable by attaching `$` followed variable name to the name of the dataset: `ds$variableName`.

```
ds<-dsL[dsL$year %in% c(2000:2011),c('id',"byear","year","attendF","ageyearF","agemon")]
print(ds[ds$id==1,])
```

	id	byear	year	attendF	ageyearF	agemon
4	1	1981	2000	Never	19	231
5	1	1981	2001	About once/week	20	243
6	1	1981	2002	Once or Twice	21	256
7	1	1981	2003	Never	22	266
8	1	1981	2004	Never	23	279
9	1	1981	2005	Never	24	290
10	1	1981	2006	Never	25	302
11	1	1981	2007	Never	26	313
12	1	1981	2008	Never	27	325
13	1	1981	2009	Never	28	337
14	1	1981	2010	Never	29	350
15	1	1981	2011	Never	29	360

The following is a list of operations that can be used in these calls.

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

dplyr can translate all of these into SQL. For more of on dplyr and SQL compatibility consult another built-in [vignette](#)

```
vignette("database",package="dplyr")
```

## Base Reference

The following unary and binary operators are defined for base. They are listed in precedence groups, from highest to lowest.

- :: ::: - access variables in a namespace
- \$ @ - component / slot extraction
- [ [[ - indexing
- ^ - exponentiation (right to left)
- - + - unary minus and plus
- : - sequence operator
- %any% - special operators (including %% and %/%)
- \* / - multiply, divide

- + - - (binary) add, subtract
- < > <= >= == != - ordering and comparison
- ! - negation
- & && - and
- | || - or
- ~ - as in formulae
- -> ->> - rightwards assignment
- <- <<- - assignment (right to left)
- = - assignment (right to left)
- ? - help (unary and binary)