



JOINING DATA WITH DATA.TABLE IN R

Concatenating data.tables

Scott Ritchie

Postdoctoral Researcher in Systems Genomics

Same columns, different data.tables

Concatenating `data.tables`

sales_2015:

quarter	amount
1	\$3,200,100
2	\$2,950,000
3	\$2,980,700
4	\$3,420,000

sales_2016:

quarter	amount
1	\$3,350,000
2	\$3,000,300
3	\$3,120,200
4	\$3,670,000



sales:

year	quarter	amount
2015	1	\$3,200,100
2015	2	\$2,950,000
2015	3	\$2,980,700
2015	4	\$3,420,000
2016	1	\$3,350,000
2016	2	\$3,000,300
2016	3	\$3,120,200
2016	4	\$3,670,000

Concatenation functions

`rbind()`: concatenate rows from `data.tables` stored in different variables

`rbindlist()`: concatenate rows from a list of `data.tables`

The rbind() function

Concatenate two or more `data.tables` stored as variables

```
# ... takes any number of arguments  
rbind(...)
```

```
rbind(sales_2015, sales_2016)
```

```
  quarter amount  
1:      1 3200100  
2:      2 2950000  
3:      3 2980700  
4:      4 3420000  
5:      1 3350000  
6:      2 3000300  
7:      3 3120200  
8:      4 3670000
```

Adding an identifier column

The `idcol` argument adds a column indicating the `data.table` of origin

```
rbind("2015" = sales_2015, "2016" = sales_2016, idcol = "year")
```

	year	quarter	amount
1:	2015	1	3200100
2:	2015	2	2950000
3:	2015	3	2980700
4:	2015	4	3420000
5:	2016	1	3350000
6:	2016	2	3000300
7:	2016	3	3120200
8:	2016	4	3670000

Adding an identifier column

```
rbind(sales_2015, sales_2016, idcol = "year")
```

	year	quarter	amount
1:	1	1	3200100
2:	1	2	2950000
3:	1	3	2980700
4:	1	4	3420000
5:	2	1	3350000
6:	2	2	3000300
7:	2	3	3120200
8:	2	4	3670000

Adding an identifier column

```
rbind(sales_2015, sales_2016, idcol = TRUE)
```

```
   .id quarter  amount
1:   1       1 3200100
2:   1       2 2950000
3:   1       3 2980700
4:   1       4 3420000
5:   2       1 3350000
6:   2       2 3000300
7:   2       3 3120200
8:   2       4 3670000
```



Handling missing columns

```
rbind("2015" = sales_2015, "2016" = sales_2016, idcol = "year",  
      fill = TRUE)
```

sales_2015:

quarter	profit
1	\$3,200,100
2	\$2,950,000
3	\$2,980,700
4	\$3,420,000

sales_2016:

quarter	profit	revenue
1	\$3,350,000	\$1,860,000
2	\$3,000,300	\$1,500,000
3	\$3,120,200	\$1,307,000
4	\$3,670,000	\$2,400,000

fill = TRUE



sales:

year	quarter	profit	revenue
2015	1	\$3,200,100	NA
2015	2	\$2,950,000	NA
2015	3	\$2,980,700	NA
2015	4	\$3,420,000	NA
2016	1	\$3,350,000	\$1,860,000
2016	2	\$3,000,300	\$1,500,000
2016	3	\$3,120,200	\$1,307,000
2016	4	\$3,670,000	\$2,400,000



Handling missing columns

```
rbind(sales_2015, sales_2016, idcol = "year")
```

```
Error in rbindlist(l, use.names, fill, idcol) :  
  Item 2 has 3 columns, inconsistent with item 1 which has 2 columns.  
  If instead you need to fill missing columns, use set argument 'fill'  
  to TRUE.
```

The rbindlist() function

Concatenate rows from a list of data.tables

```
# Read in a list of data.tables
table_files <- c("sales_2015.csv", "sales_2016.csv")
list_of_tables <- lapply(table_files, fread)
```

```
rbindlist(list_of_tables)
```

```
  quarter amount
1:      1 3200100
2:      2 2950000
3:      3 2980700
4:      4 3420000
5:      1 3350000
6:      2 3000300
7:      3 3120200
8:      4 3670000
```

Adding an identifier column

The `idcol` argument takes names from the input list

```
names(list_of_tables) <- c("2015", "2016")  
rbindlist(list_of_tables, idcol = "year")
```

```
   year quarter amount  
1: 2015         1 3200100  
2: 2015         2 2950000  
3: 2015         3 2980700  
4: 2015         4 3420000  
5: 2016         1 3350000  
6: 2016         2 3000300  
7: 2016         3 3120200  
8: 2016         4 3670000
```

Handling different column orders

```
rbind("2015" = sales_2015, "2016" = sales_2016, idcol = "year",  
      use.names = TRUE)
```

sales_2015:

quarter	amount
1	\$3,200,100
2	\$2,950,000
3	\$2,980,700
4	\$3,420,000

sales_2016:

amount	quarter
\$3,350,000	1
\$3,000,300	2
\$3,120,200	3
\$3,670,000	4

use.names = TRUE



sales:

year	quarter	amount
2015	1	\$3,200,100
2015	2	\$2,950,000
2015	3	\$2,980,700
2015	4	\$3,420,000
2016	1	\$3,350,000
2016	2	\$3,000,300
2016	3	\$3,120,200
2016	4	\$3,670,000

data.tables with different column names

```
rbind("2015" = sales_2015, "2016" = sales_2016, idcol = "year",  
      use.names = FALSE)
```

sales_2015:

quarter	amount
1	\$3,200,100
2	\$2,950,000
3	\$2,980,700
4	\$3,420,000

sales_2016:

quarter	profit
1	\$3,350,000
2	\$3,000,300
3	\$3,120,200
4	\$3,670,000

use.names = FALSE



sales:

year	quarter	amount
2015	1	\$3,200,100
2015	2	\$2,950,000
2015	3	\$2,980,700
2015	4	\$3,420,000
2016	1	\$3,350,000
2016	2	\$3,000,300
2016	3	\$3,120,200
2016	4	\$3,670,000

Pitfalls of use.names = FALSE

```
rbind("2015" = sales_2015, "2016" = sales_2016, idcol = "year",  
      use.names = FALSE)
```

sales_2015:

quarter	amount
1	\$3,200,100
2	\$2,950,000
3	\$2,980,700
4	\$3,420,000

sales_2016:

amount	quarter
\$3,350,000	1
\$3,000,300	2
\$3,120,200	3
\$3,670,000	4

use.names = FALSE



sales:

year	quarter	amount
2015	1	\$3,200,100
2015	2	\$2,950,000
2015	3	\$2,980,700
2015	4	\$3,420,000
2016	\$3,350,000	1
2016	\$3,000,300	2
2016	\$3,120,200	3
2016	\$3,670,000	4

Differing defaults

- Default for `rbind()` **is** `use.names = TRUE`
- Default for `rbindlist()` **is** `use.names = FALSE` **unless** `fill = TRUE`.



JOINING DATA WITH DATA.TABLE IN R

Let's practice!



JOINING DATA WITH DATA.TABLE IN R

Set operations

Scott Ritchie

Postdoctoral Researcher in Systems Genomics

Set operation functions

Given two `data.tables` with the same columns:

- `fintersect()`: what rows do these two `data.tables` share in common?
- `funion()`: what is the unique set of rows across these two `data.tables`?
- `fsetdiff()`: what rows are unique to this `data.table`?

Set operations: fintersect()

Extract rows that are present in both `data.tables`

```
fintersect(dt1, dt2)
```

dt1:

id	animal	color
1	giraffe	yellow
2	lion	yellow
3	antelope	brown
4	mouse	grey

dt2:

id	animal	color
2	lion	yellow
4	mouse	grey
5	whale	blue
6	cassowary	black

fintersect()



id	animal	color
2	lion	yellow
4	mouse	grey

fintersect() and duplicate rows

Duplicate rows are ignored by default:

```
fintersect(dt1, dt2)
```

dt1:

id	animal	color
1	giraffe	yellow
2	lion	yellow
3	antelope	brown
4	mouse	grey
2	lion	yellow
2	lion	yellow

dt2:

id	animal	color
2	lion	yellow
4	mouse	grey
5	whale	blue
6	cassowary	black
2	lion	yellow

fintersect()



id	animal	color
2	lion	yellow
4	mouse	grey

fintersect() and duplicate rows

`all = TRUE`: keep the number of copies present in both `data.tables`:

```
fintersect(dt1, dt2, all = TRUE)
```

dt1:

id	animal	color
1	giraffe	yellow
2	lion	yellow
3	antelope	brown
4	mouse	grey
2	lion	yellow
2	lion	yellow

dt2:

id	animal	color
2	lion	yellow
4	mouse	grey
5	whale	blue
6	cassowary	black
2	lion	yellow

fintersect()



id	animal	color
2	lion	yellow
4	mouse	grey
2	lion	yellow

Set operations: fsetdiff()

Extract rows found exclusively in the first `data.table`

```
fsetdiff(dt1, dt2)
```

dt1:

id	animal	color
1	giraffe	yellow
2	lion	yellow
3	antelope	brown
4	mouse	grey

dt2:

id	animal	color
2	lion	yellow
4	mouse	grey
5	whale	blue
6	cassowary	black

fsetdiff()



id	animal	color
1	giraffe	yellow
3	antelope	brown

fsetdiff() and duplicates

Duplicate rows are ignored by default:

```
fsetdiff(dt1, dt2)
```

dt1:

id	animal	color
1	giraffe	yellow
2	lion	yellow
3	antelope	brown
4	mouse	grey
2	lion	yellow
2	lion	yellow
3	antelope	brown

dt2:

id	animal	color
2	lion	yellow
4	mouse	grey
5	whale	blue
6	cassowary	black
2	lion	yellow

fsetdiff()
➔

id	animal	color
1	giraffe	yellow
3	antelope	brown

fsetdiff() and duplicates

`all = TRUE`: return all extra copies:

```
fsetdiff(dt1, dt2, all = TRUE)
```

dt1:

id	animal	color
1	giraffe	yellow
2	lion	yellow
3	antelope	brown
4	mouse	grey
2	lion	yellow
2	lion	yellow
3	antelope	brown

dt2:

id	animal	color
2	lion	yellow
4	mouse	grey
5	whale	blue
6	cassowary	black
2	lion	yellow

fsetdiff()
➔

id	animal	color
1	giraffe	yellow
3	antelope	brown
2	lion	yellow
3	antelope	brown

Set operations: union()

Extract all rows found in either `data.table`:

```
union(dt1, dt2)
```

dt1:

id	animal	color
1	giraffe	yellow
2	lion	yellow
3	antelope	brown
4	mouse	grey

dt2:

id	animal	color
2	lion	yellow
4	mouse	grey
5	whale	blue
6	cassowary	black

union()
➔

id	animal	color
1	giraffe	yellow
3	antelope	brown
2	lion	yellow
4	mouse	grey
5	whale	blue
6	cassowary	black

union() and duplicates

Duplicate rows are ignored by default:

```
union(dt1, dt2)
```

dt1:

id	animal	color
1	giraffe	yellow
2	lion	yellow
2	lion	yellow
2	lion	yellow

dt2:

id	animal	color
2	lion	yellow
4	mouse	grey
5	whale	blue
2	lion	yellow

union()
➔

id	animal	color
1	giraffe	yellow
2	lion	yellow
4	mouse	grey
5	whale	blue

union() and duplicates

`all = TRUE`: return all rows:

```
union(dt1, dt2, all = TRUE) # rbind()
```

dt1:

id	animal	color
1	giraffe	yellow
2	lion	yellow
2	lion	yellow
2	lion	yellow

dt2:

id	animal	color
2	lion	yellow
4	mouse	grey
5	whale	blue
2	lion	yellow

union()
➔

id	animal	color
1	giraffe	yellow
2	lion	yellow
2	lion	yellow
2	lion	yellow
2	lion	yellow
4	mouse	grey
5	whale	blue
2	lion	yellow

Removing duplicates when combining many `data.tables`

Two `data.tables`:

1. Use `union()` to concatenate unique rows

Three or more:

1. Concatenate all `data.tables` using `rbind()` or `rbindlist()`
2. Identify and remove duplicates using `duplicated()` and `unique()`



JOINING DATA WITH DATA.TABLE IN R

Let's practice!



JOINING DATA WITH DATA.TABLE IN R

Melting data.tables

Scott Ritchie

Postdoctoral Researcher in Systems Genomics

Melting a wide data.table

sales_wide:

quarter	2015	2016
1	\$3,200,100	\$3,350,000
2	\$2,950,000	\$3,000,300
3	\$2,980,700	\$3,120,200
4	\$3,420,000	\$3,670,000



sales_long:

quarter	year	amount
1	2015	\$3,200,100
2	2015	\$2,950,000
3	2015	\$2,980,700
4	2015	\$3,420,000
1	2016	\$3,350,000
2	2016	\$3,000,300
3	2016	\$3,120,200
4	2016	\$3,670,000

The melt() function

Use `measure.vars` to specify columns to stack:

```
melt(sales_wide, measure.vars = c("2015", "2016"))
```

	quarter	variable	value
1:	1	2015	3200100
2:	2	2015	2950000
3:	3	2015	2980700
4:	4	2015	3420000
5:	1	2016	3350000
6:	2	2016	3000300
7:	3	2016	3120200
8:	4	2016	3670000



The melt() function

Use `variable.name` and `value.name` to rename these columns in the result:

```
melt(sales_wide, measure.vars = c("2015", "2016"),  
     variable.name = "year", value.name = "amount")
```

```
   quarter year amount  
1:         1 2015 3200100  
2:         2 2015 2950000  
3:         3 2015 2980700  
4:         4 2015 3420000  
5:         1 2016 3350000  
6:         2 2016 3000300  
7:         3 2016 3120200  
8:         4 2016 3670000
```

The melt() function

Use `id.vars` to specify columns to keep aside

```
melt(sales_wide, id.vars = "quarter",  
     variable.name = "year", value.name = "amount")
```

```
   quarter year  amount  
1:      1 2015 3200100  
2:      2 2015 2950000  
3:      3 2015 2980700  
4:      4 2015 3420000  
5:      1 2016 3350000  
6:      2 2016 3000300  
7:      3 2016 3120200  
8:      4 2016 3670000
```

The melt() function

Use both to keep only a subset of columns

```
melt(sales_wide, id.vars = "quarter", measure.vars = "2015",  
     variable.name = "year", value.name = "amount")
```

```
  quarter year  amount  
1:      1 2015 3200100  
2:      2 2015 2950000  
3:      3 2015 2980700  
4:      4 2015 3420000
```



JOINING DATA WITH DATA.TABLE IN R

Let's practice!



JOINING DATA WITH DATA.TABLE IN R

Casting data.tables

Scott Ritchie

Postdoctoral Researcher in Systems Genomics

Casting a long data.table

```
sales_wide <- dcast(sales_long, quarter ~ year, value.var = "amount")
```

sales_long:

quarter	year	amount
1	2015	\$3,200,100
2	2015	\$2,950,000
3	2015	\$2,980,700
4	2015	\$3,420,000
1	2016	\$3,350,000
2	2016	\$3,000,300
3	2016	\$3,120,200
4	2016	\$3,670,000



sales_wide:

quarter	2015	2016
1	\$3,200,100	\$3,350,000
2	\$2,950,000	\$3,000,300
3	\$2,980,700	\$3,120,200
4	\$3,420,000	\$3,670,000

The dcast() function

The general form of `dcast()`:

```
dcast(DT, ids ~ group, value.var = "values")
|      |      |                               |
|      |      |                               --> column to split
|      |      |-----> group labels to split by
|-----> rows to keep behind as identifiers
-----> data.table to reshape
```

The dcast() function

```
sales_wide <- dcast(sales_long, quarter ~ year, value.var = "amount")
```

sales_long:

quarter	year	amount
1	2015	\$3,200,100
2	2015	\$2,950,000
3	2015	\$2,980,700
4	2015	\$3,420,000
1	2016	\$3,350,000
2	2016	\$3,000,300
3	2016	\$3,120,200
4	2016	\$3,670,000



sales_wide:

quarter	2015	2016
1	\$3,200,100	\$3,350,000
2	\$2,950,000	\$3,000,300
3	\$2,980,700	\$3,120,200
4	\$3,420,000	\$3,670,000

Splitting multiple value columns

```
dcast(profit_long, quarter ~ year, value.var = c("revenue", "profit"))
```

profit_long:

quarter	year	revenue	profit
1	2015	\$3,200,100	\$640,020
2	2015	\$2,950,000	\$590,000
3	2015	\$2,980,700	\$596,140
4	2015	\$3,420,000	\$684,000
1	2016	\$3,350,000	\$670,000
2	2016	\$3,000,300	\$600,060
3	2016	\$3,120,200	\$624,040
4	2016	\$3,670,000	\$734,000



quarter	revenue_2015	revenue_2016	profit_2015	profit_2016
1	\$3,200,100	\$3,350,000	\$640,020	\$670,000
2	\$2,950,000	\$3,000,300	\$590,000	\$600,060
3	\$2,980,700	\$3,120,200	\$596,140	\$624,040
4	\$3,420,000	\$3,670,000	\$684,000	\$734,000

Multiple row identifiers

Keep multiple columns as row identifiers:

```
dcast(sales_long, quarter + season ~ year, value.var = "amount")
```

sales_long:

quarter	season	year	amount
1	Winter	2015	\$3,200,100
2	Spring	2015	\$2,950,000
3	Summer	2015	\$2,980,700
4	Autumn	2015	\$3,420,000
1	Winter	2016	\$3,350,000
2	Spring	2016	\$3,000,300
3	Summer	2016	\$3,120,200
4	Autumn	2016	\$3,670,000



quarter	season	2015	2016
1	Winter	\$3,200,100	\$3,350,000
2	Spring	\$2,950,000	\$3,000,300
3	Summer	\$2,980,700	\$3,120,200
4	Autumn	\$3,420,000	\$3,670,000

Dropping columns

Only columns included in the formula or `value.var` will be in the result:

```
sales_wide <- dcast(sales_long, quarter ~ year, value.var = "amount")
```

sales_long:

quarter	season	year	amount
1	Winter	2015	\$3,200,100
2	Spring	2015	\$2,950,000
3	Summer	2015	\$2,980,700
4	Autumn	2015	\$3,420,000
1	Winter	2016	\$3,350,000
2	Spring	2016	\$3,000,300
3	Summer	2016	\$3,120,200
4	Autumn	2016	\$3,670,000



quarter	2015	2016
1	\$3,200,100	\$3,350,000
2	\$2,950,000	\$3,000,300
3	\$2,980,700	\$3,120,200
4	\$3,420,000	\$3,670,000

Multiple groupings

Split on multiple group columns:

```
dcast(sales_long, quarter ~ department + year, value.var = "amount")
```

sales_long:

quarter	department	year	amount
1	retail	2015	\$3,200,100
3	retail	2015	\$2,980,700
1	retail	2016	\$3,350,000
3	retail	2016	\$3,120,200
1	consulting	2015	\$100,400
3	consulting	2015	\$130,200
1	consulting	2016	\$125,000
3	consulting	2016	\$150,400



quarter	retail_2015	retail_2016	consulting_2015	consulting_2016
1	\$3,200,100	\$3,350,000	\$100,400	\$125,000
3	\$2,980,700	\$3,120,200	\$130,200	\$150,400

Converting to a matrix

```
sales_wide <- dcast(sales_long, season ~ year, value.var = "amount")
sales_wide

  season    2015    2016
1: Autumn 3420000 3670000
2: Spring 2950000 3000300
3: Summer 2980700 3120200
4: Winter 3200100 3350000
```

`as.matrix()` can take one of the columns to use as the matrix rownames:

```
mat <- as.matrix(sales_wide, rownames = "season")
mat

      2015    2016
Autumn 3420000 3670000
Spring 2950000 3000300
Summer 2980700 3120200
Winter 3200100 3350000
```



JOINING DATA WITH DATA.TABLE IN R

Let's practice!