

# Data manipulation in R

Hun Yong Cho  
Department of Biostatistics

# Data manipulation and summary tools in R

magrittr

pipes

dplyr

filter / select / mutate / arrange / group\_by / join ...

tidyr

gather / spread

reshape2

melt / dcast

base / stats

summary / table / aggregate

psych

describe

Preliminary tool - pipes

# pipes make your code very simple

`%>%` (Called “forward pipe”)

package: magrittr

(Often exported to other packages: dplyr, tidyr, ... )

Variants:

`%<>%` compound assignment operator

`%T>%` tree operator

`%%$%` exposition operator

`%>%` and `%<>%` are frequently used.

Same concept will be also used later in bash scripts.

# pipes make your code very simple

## Usual coding

```
names(mtcars)  
  
sort(names(mtcars))
```

```
a <- c(1, 3, 2)  
  
a <- sort(a)
```

## pipes

```
mtcars %>% names  
  
mtcars %>% names %>% sort
```

```
a <- a %>% sort      or  
a %<>% sort
```

# How pipes work?

## *Cognitive process:*

1. Take the **ydat** dataset, *then*
2. **filter()** for genes in the leucine biosynthesis pathway, *then*
3. **group\_by()** the limiting nutrient, *then*
4. **summarize()** to correlate rate and expression, *then*
5. **mutate()** to round *r* to two digits, *then*
6. **arrange()** by rounded correlation coefficients

## *The old way:*

```
arrange(  
  mutate(  
    summarize(  
      group_by(  
        filter(ydat, bp=="leucine biosynthesis"),  
        nutrient),  
      r=cor(rate, expression)),  
    r=round(r, 2)),  
  r)
```

## *The dplyr way:*

```
ydat %>%  
  filter(bp=="leucine biosynthesis") %>%  
  group_by(nutrient) %>%  
  summarize(r=cor(rate, expression)) %>%  
  mutate(r=round(r,2)) %>%  
  arrange(r)
```

# pipes make your code very simple

## Frequently used examples

### Usual coding

```
abc <- letters[1:5]
```

```
abc <- as.factor(abc)
```

```
abc <- matrix(1:9, nrow = 3)
```

```
abc <- as.data.frame(abc)
```

```
ggplot(abc, aes(V1, V2)) + geom_point()
```

### pipes

```
abc %<>% as.factor
```

```
abc %<>% as.data.frame
```

```
abc %>% ggplot(aes(V1, V2)) + geom_point()
```

or

```
abc %>% as.data.frame %>% ggplot ...
```

BE CAUTIOUS: You **CANNOT RECOVER** the object after compound assignment operator (**%<>%**)

# pipes make your code very simple

More skills: use “dots”

Usual coding

```
a <- rnorm(100)
a1 <- min(a0)
a2 <- mean(a0)
a3 <- max(a0)
c(a1, a2, a3)
```

pipes

```
rnorm(100) %>% {c(min(.), mean(.), max(.))}
```



**dplyr**

# Review of main functions

most of dplyr were covered in r4ds.

- filter / select / mutate / arrange / group\_by / ...
- “join” will be covered later

filter:	when you want to include only some rows
select:	include only some columns
mutate / transmute:	add some transformed variables
group_by:	summarize data by groups
arrange:	sorting the dataset
join:	merge two datasets

# Review of main functions - Exercise

## Longitudinal data example

a subset of FEV dataset from BIOS 767  
(Reference is given in fev.txt).

Download [fev.csv](#) data into your working directory.

```
fev = read.csv("fev.csv", header=T)
```

```
> fev %>% head
  id  ht   age logfev1
1  1 1.20  9.3415 0.21511
2  1 1.28 10.3929 0.37156
3  1 1.33 11.4524 0.48858
4  1 1.42 12.4600 0.75142
5  1 1.48 13.4182 0.83291
6  1 1.50 15.4743 0.89200
```

We are interested in making a spaghetti plot (i.e. one id makes one line)  
where x-axis as [time from baseline](#) and y-axis is [log\(FEV1\)](#)  
To make such a plot, how can we manipulate the data?

hint: make “time” variable using `group_by`, `mutate`, `min`.

# Review of main functions - Exercise

## Longitudinal data example, (cont'd)

We want to fit a linear mixed effects model  
for those whose baseline age is less than 7 or  
whose baseline height is less than 1.2,  
where the model includes  
time from baseline (time), baseline age, and baseline height.

Manipulate the dataset so that it only includes  
variables “id”, “baseht”, “baseage”, “time”, “logfev1”, and (no “age”) and  
subjects *whose baseage < 7*,  
or *whose baseht < 1.2*

hint: group\_by, mutate, filter, select

```
> fev %>% head
  id ht      age logfev1
1  1 1.20  9.3415 0.21511
2  1 1.28 10.3929 0.37156
3  1 1.33 11.4524 0.48858
4  1 1.42 12.4600 0.75142
5  1 1.48 13.4182 0.83291
6  1 1.50 15.4743 0.89200
```



	id	logfev1	baseage	baseht	time
	<int>	<dbl>	<dbl>	<dbl>	<dbl>
1	2	0.307	6.59	1.13	0
2	2	0.351	6.59	1.13	1.06
3	2	0.756	6.59	1.13	6.15
4	2	0.867	6.59	1.13	7.19
5	2	1.05	6.59	1.13	8.11
6	2	1.15	6.59	1.13	9.23

# Reshaping by (tidyr)

# Reshaping

You want to run  
mixed effect regression, or  
time-varying covariate survival analysis.

But the data is in a wide form!

Then what would you do?

`tidyr::gather` can be used to change **wide form data to long**.

```
gather(<your data>,  
      key = <new name for time indicator>,  
      value = <new name for time-varying variable>,  
      <existing variable names to be reorganized>)
```

# Reshaping

```
> data.wide  
  name a  b  
1 Wilbur 67 56  
2 Petunia 80 90  
3 Gregory 64 50
```



```
> data.long  
  name drug heartrate  
1 Wilbur  a         67  
2 Petunia a         80  
3 Gregory a         64  
4 Wilbur  b         56  
5 Petunia b         90  
6 Gregory b         50
```

```
data.wide <- data.frame(name = c( "Wilbur", "Petunia", "Gregory"),  
                        a = c(67, 80, 64), b = c(56, 90, 50))
```

```
data.long <-  
  data.wide %>%  
    gather(key = drug, value = heartrate, a:b)
```

# Reshaping

```
> data.wide
  id    trt   work.T1  home.T1  work.T2  home.T2
1  1 treatment 0.08513597 0.6158293 0.1135090 0.05190332
2  2  control 0.22543662 0.4296715 0.5959253 0.26417767
3  3 treatment 0.27453052 0.6516557 0.3580500 0.39879073
4  4  control 0.27230507 0.5677378 0.4288094 0.83613414
```



```
> data.long
  id    trt   info      time
1  1 treatment work.T1 0.08513597
2  2  control work.T1 0.22543662
3  3 treatment work.T1 0.27453052
4  4  control work.T1 0.27230507
5  1 treatment home.T1 0.61582931
6  2  control home.T1 0.42967153
7  3 treatment home.T1 0.65165567
8  4  control home.T1 0.56773775
9  1 treatment work.T2 0.11350898
10 2  control work.T2 0.59592531
11 3 treatment work.T2 0.35804998
12 4  control work.T2 0.42880942
13 1 treatment home.T2 0.05190332
```

```
set.seed(10)
data.wide <- data.frame(
  id = 1:4,
  trt = sample(rep(c('control', 'treatment'), each = 2)),
  work.T1 = runif(4),
  home.T1 = runif(4),
  work.T2 = runif(4),
  home.T2 = runif(4)
)
```

```
set.seed(10)
data.long <- data.wide %>%
  gather (key = info, value = time, -id, -trt)
```

This example is from BSA workshop by Phoebe.



# Reshaping

Back to wide form?

`tidyr::spread` can be used  
to reshape from long to wide.

```
data.long %>%  
  spread (key = info, value = time)
```

```
> data.long  
   id   trt   info   time  
1  1 treatment work.T1 0.08513597  
2  2  control work.T1 0.22543662  
3  3 treatment work.T1 0.27453052  
4  4  control work.T1 0.27230507  
5  1 treatment home.T1 0.61582931  
6  2  control home.T1 0.42967153  
7  3 treatment home.T1 0.65165567  
8  4  control home.T1 0.56773775  
9  1 treatment work.T2 0.11350898  
10 2  control work.T2 0.59592531  
11 3 treatment work.T2 0.35804998  
12 4  control work.T2 0.42880942  
13 1 treatment home.T2 0.05190332
```



```
   id   trt  home.T1  home.T2  work.T1  work.T2  
1  1 treatment 0.6158293 0.05190332 0.08513597 0.1135090  
2  2  control 0.4296715 0.26417767 0.22543662 0.5959253  
3  3 treatment 0.6516557 0.39879073 0.27453052 0.3580500  
4  4  control 0.5677378 0.83613414 0.27230507 0.4288094
```

# Reshaping

Other frequently used package: [reshape2](#)

## wide to long

```
data.wide %>% melt(id.vars = c("id", "trt"), variable.name = "info", value.name = "time")
```

## long to wide

```
data.long %>% dcast(data.long, id + trt ~ info, value.var="time")
```

More on [Cookbook for R](#)

# Summarizing data

summary  
psych::describe  
table  
aggregate

# Data summary

`dim(iris)`

`head(iris)`

`tail(iris)`

```
> summary(iris)
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
Min. :4.300	Min. :2.000	Min. :1.000	Min. :0.100	setosa :50
1st Qu.:5.100	1st Qu.:2.800	1st Qu.:1.600	1st Qu.:0.300	versicolor:50
Median :5.800	Median :3.000	Median :4.350	Median :1.300	virginica :50
Mean :5.843	Mean :3.057	Mean :3.758	Mean :1.199	
3rd Qu.:6.400	3rd Qu.:3.300	3rd Qu.:5.100	3rd Qu.:1.800	
Max. :7.900	Max. :4.400	Max. :6.900	Max. :2.500	

# Data summary

```
> psych::describe(iris)
```

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
Sepal.Length	1	150	5.84	0.83	5.80	5.81	1.04	4.3	7.9	3.6	0.31	-0.61	0.07
Sepal.Width	2	150	3.06	0.44	3.00	3.04	0.44	2.0	4.4	2.4	0.31	0.14	0.04
Petal.Length	3	150	3.76	1.77	4.35	3.76	1.85	1.0	6.9	5.9	-0.27	-1.42	0.14
Petal.Width	4	150	1.20	0.76	1.30	1.18	1.04	0.1	2.5	2.4	-0.10	-1.36	0.06
Species*	5	150	2.00	0.82	2.00	2.00	1.48	1.0	3.0	2.0	0.00	-1.52	0.07

```
iris %>% describe %>% select(n, mean, sd, min, max)
```

```
iris %>% describe %>% select(n, mean, sd, min, max) %>% (knitr::kable)
```

```
iris %>% describe %>% select(n, mean, sd, min, max) %>% (xtable::xtable)
```

# Data summary - subgroup

```
> psych::describeBy(iris, group = "Species")
```

Descriptive statistics by group

group: setosa

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
Sepal.Length	1	50	5.01	0.35	5.0	5.00	0.30	4.3	5.8	1.5	0.11	-0.45	0.05
Sepal.Width	2	50	3.43	0.38	3.4	3.42	0.37	2.3	4.4	2.1	0.04	0.60	0.05
Petal.Length	3	50	1.46	0.17	1.5	1.46	0.15	1.0	1.9	0.9	0.10	0.65	0.02
Petal.Width	4	50	0.25	0.11	0.2	0.24	0.00	0.1	0.6	0.5	1.18	1.26	0.01
Species*	5	50	1.00	0.00	1.0	1.00	0.00	1.0	1.0	0.0	NaN	NaN	0.00

-----  
group: versicolor

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
Sepal.Length	1	50	5.94	0.52	5.90	5.94	0.52	4.9	7.0	2.1	0.10	-0.69	0.07
Sepal.Width	2	50	2.77	0.31	2.80	2.78	0.30	2.0	3.4	1.4	-0.34	-0.55	0.04
Petal.Length	3	50	4.26	0.47	4.35	4.29	0.52	3.0	5.1	2.1	-0.57	-0.19	0.07
Petal.Width	4	50	1.33	0.20	1.30	1.32	0.22	1.0	1.8	0.8	-0.03	-0.59	0.03
Species*	5	50	2.00	0.00	2.00	2.00	0.00	2.0	2.0	0.0	NaN	NaN	0.00

-----  
group: virginica

# Data summary - contingency table

```
> mtcars
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3

For each combination of cyl and gear, want to know how many cars there are.

How to make such a contingency table?

```
table(mtcars$cyl, mtcars$gear)
```

Further exercise: 1) how to simplify this code using pipes?

2) Can you make a contingency table for three variables (cyl, gear, vs)?

# Data summary - aggregate (subgroup summary statistics)

How can we get the mean value of Sepal.Length by Species?

```
> aggregate(Sepal.Length ~ Species, data=iris, FUN = mean)
```

	Species	Sepal.Length
1	setosa	5.006
2	versicolor	5.936
3	virginica	6.588

How can we get the mean value of every variable by Species?

```
> aggregate(. ~ Species, data=iris, FUN = mean)
```

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

How can we get the geometric mean of every variable by Species?

```
> aggregate(. ~ Species, data=iris, FUN = geometric.mean)
```

	Species	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
1	setosa	4.993841	3.407080	1.451734	0.2265819
2	versicolor	5.913979	2.751874	4.233081	1.3111874
3	virginica	6.557795	2.957014	5.525789	2.0072141



sort / order / unique / arrange / distinct

# sort / order / unique

```
a = c(1, 10, 4, 3, 2, 1)
```

```
sort(a)          # sorted values
```

```
order(a)         # order of current vector
```

```
unique(a)        # unique values
```

```
a = c(1, 10, 4, 3, 2, 1)
```

```
sort(a)          # 1 1 2 3 4 10
```

```
order(a)         # 1 6 5 4 3 2
```

```
unique(a)        # 1 10 4 3 2
```

These are for vectors

# sort / order / unique / arrange / distinct

For vectors

```
a = c(1, 10, 4, 3, 2)

sort(a)          # sorted values

sort(a, decreasing = TRUE)

order(a)         # order of current vector

a = c(1, 10, 4, 4, 1)

unique(a)        # unique values
```

For data.frames.

```
iris %>% arrange(Sepal.Length)

iris %>% arrange(desc(Sepal.Length))

iris %>% mutate(order(<var.name>))

iris %>% unique      or

iris %>% distinct
```