Becoming a data ninja with dplyr

major data manipulation verbs

Verb	Usage
filter	Keep matching row criteria
summarize	reduces summary values calculated
mutate	add new variables to existing data frame
select	select columns by name
arrange	reorder rows

```
df <- data.frame(ID = 1:5,
GENDER = c("MALE", "MALE", "FEMALE", "MALE", "FEMALE"),
WT = c(70, 76, 60, 64, 68))
```

ID	GENDER	WT
1	MALE	70
2	MALE	76
3	FEMALE	60
4	MALE	64
5	FEMALE	68

Verb	Usage
filter	Keep matching row criteria
summarize	reduces summary values calculated
mutate	add new variables to existing data frame
select	select columns by name
arrange	reorder rows

```
filter(df, GENDER == "FEMALE")
```

ID	GENDER	WT
1	MALE	70
2	MALE	76
3	FEMALE	60
4	MALE	64
5	FEMALE	68

ID	GENDER	WT
3	FEMALE	60
5	FEMALE	68

common dplyr filter (subset) operators

operator	meaning
==, !=	equal, not equal
>, >=	greater than, greater than or equal to
<, <=	less than, less than or equal to
is.na, !is.na	is NA, not NA
!duplicated	only first value
%in%	in specified values

filter seperator	base equivalent	meaning
,	&	and
|	|	or

```
filter(df, ID %in% c(1, 3, 5))
```

ID	GENDER	WT
1	MALE	70
2	MALE	76
3	FEMALE	60
4	MALE	64
5	FEMALE	68

ID	GENDER	WT
1	MALE	70
3	FEMALE	60
5	FEMALE	68

```
filter(df, GENDER == "MALE", WT > 70)
```

ID	GENDER	WT
1	MALE	70
2	MALE	76
3	FEMALE	60
4	MALE	64
5	FEMALE	68

ID	GENDER	WT
2	MALE	76

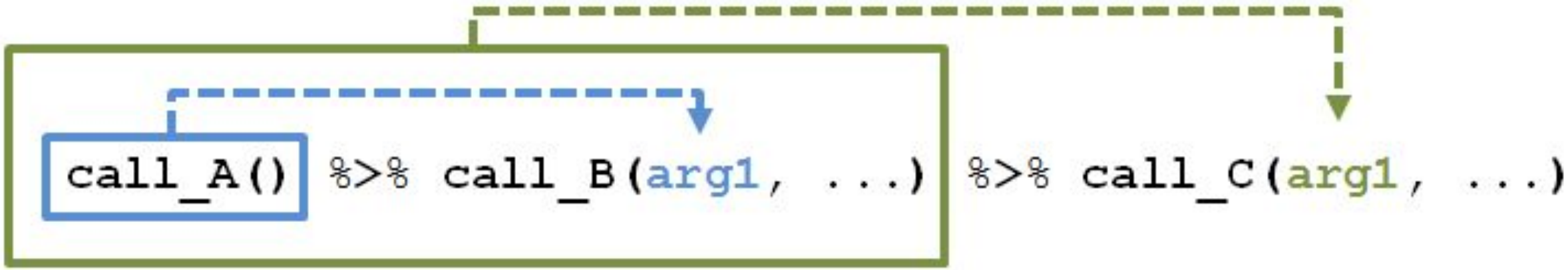
```
filter(df, GENDER == "FEMALE" | WT < 70)
```

ID	GENDER	WT
1	MALE	70
2	MALE	76
3	FEMALE	60
4	MALE	64
5	FEMALE	68

ID	GENDER	WT
3	FEMALE	60
4	MALE	64
5	FEMALE	68

chaining and grouped operations

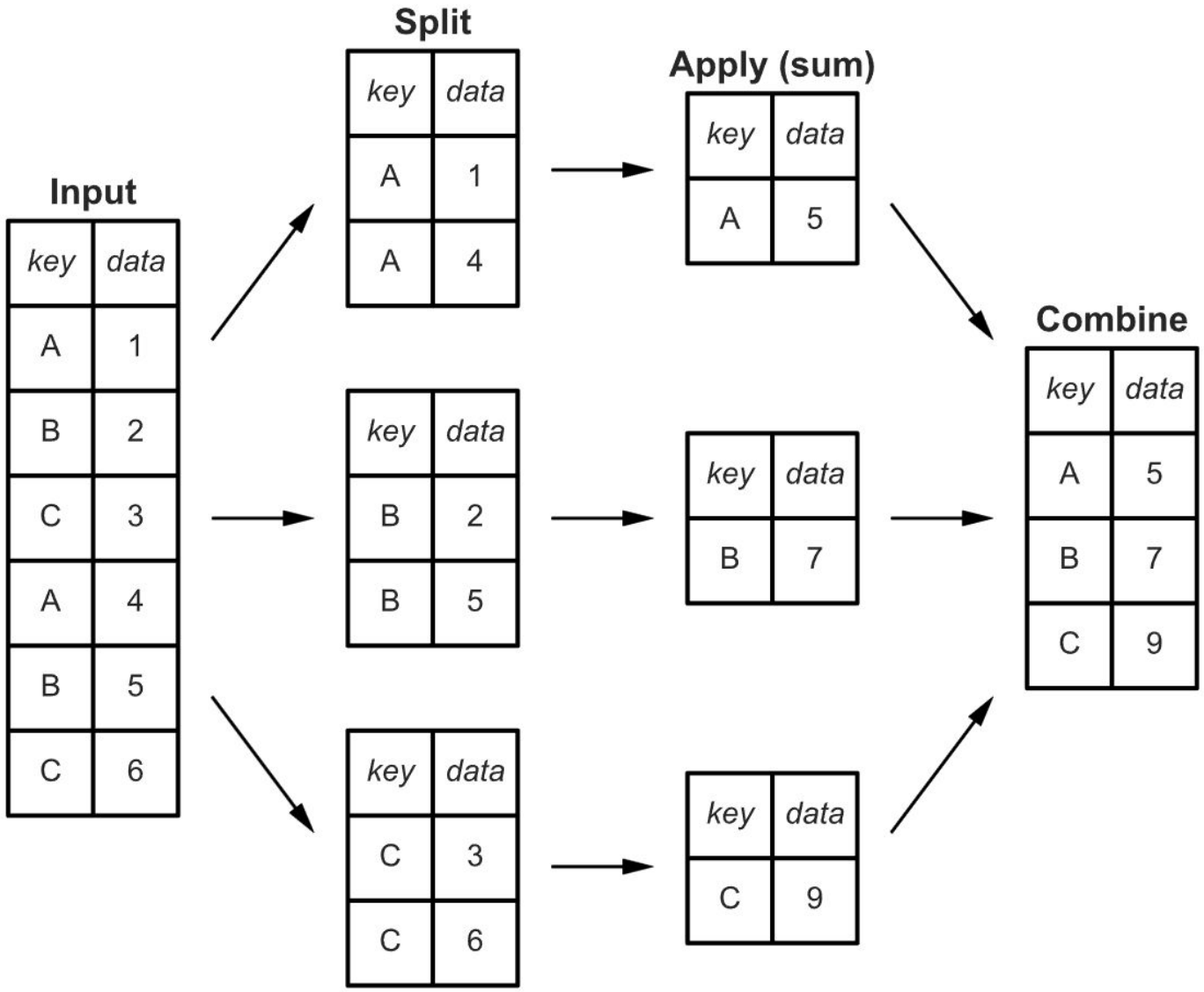
chaining with %>%



%>% is pronounced ‘then’

```
> Theoph %>% filter(!duplicated(Subject)) %>% head()
   Subject   Wt Dose Time conc
1        1 79.6 4.02    0 0.74
2        2 72.4 4.40    0 0.00
3        3 70.5 4.53    0 0.00
4        4 72.7 4.40    0 0.00
5        5 54.6 5.86    0 0.00
6        6 80.0 4.00    0 0.00
```

split-apply-combine -> **group_by**



https://jakevdp.github.io/figures/split-apply-combine.svg

Verb	Usage
filter	Keep matching row criteria
summarize	reduces summary values calculated
mutate	add new variables to existing data frame
select	select columns by name
arrange	reorder rows

```
df %>% summarize(meanWT = mean(WT))
```

ID	GENDER	WT
1	MALE	70
2	MALE	76
3	FEMALE	60
4	MALE	64
5	FEMALE	68

meanWT
67.6

```
summarize(df, meanWT = mean(WT))
```

```
df %>% group_by(GENDER) %>%
  summarize(meanWT = mean(WT))
```

ID	GENDER	WT
1	MALE	70
2	MALE	76
3	FEMALE	60
4	MALE	64
5	FEMALE	68

GENDER	meanWT
MALE	70
FEMALE	64

```
df %>% group_by(GENDER) %>%
  summarize(meanWT = mean(WT), n = n())
```

ID	GENDER	WT
1	MALE	70
2	MALE	76
3	FEMALE	60
4	MALE	64
5	FEMALE	68

GENDER	meanWT	n
MALE	70	3
FEMALE	64	2

Verb	Usage
filter	Keep matching row criteria
summarize	reduces summary values calculated
mutate	add new variables to existing data frame
select	select columns by name
arrange	reorder rows

```
df %>% mutate(meanWT = mean(WT))
```

ID	GENDER	WT
1	MALE	70
2	MALE	76
3	FEMALE	60
4	MALE	64
5	FEMALE	68

ID	GENDER	WT	meanWT
1	MALE	70	67.6
2	MALE	76	67.6
3	FEMALE	60	67.6
4	MALE	64	67.6
5	FEMALE	68	67.6

```
df %>% group_by(GENDER) %>%
 mutate(meanWT = mean(WT))
```

ID	GENDER	WT
1	MALE	70
2	MALE	76
3	FEMALE	60
4	MALE	64
5	FEMALE	68

ID	GENDER	WT	meanWT
1	MALE	70	70
2	MALE	76	70
3	FEMALE	60	64
4	MALE	64	70
5	FEMALE	68	64

```
df %>% group_by(GENDER) %>%
  mutate(meanWT = mean(WT),
         mWT_LB = meanWT*2.2)
```

ID	GENDER	WT
1	MALE	70
2	MALE	76
3	FEMALE	60
4	MALE	64
5	FEMALE	68

ID	GENDER	WT	meanWT	mWT_LB
1	MALE	70	70	154
2	MALE	76	70	154
3	FEMALE	60	64	140.8
4	MALE	64	70	154
5	FEMALE	68	64	140.8

```
df %>%
 mutate(ISM = ifelse(GENDER == "MALE", 1, 0))
```

ID	GENDER	WT
1	MALE	70
2	MALE	76
3	FEMALE	60
4	MALE	64
5	FEMALE	68

ID	GENDER	WT	ISM
1	MALE	70	1
2	MALE	76	1
3	FEMALE	60	0
4	MALE	64	1
5	FEMALE	68	0

Verb	Usage
filter	Keep matching row criteria
summarize	reduces summary values calculated
mutate	add new variables to existing data frame
select	select columns by name
arrange	reorder rows

df2 %>% select(ID, WT)

ID	GENDER	WT	meanWT
1	MALE	70	67.6
2	MALE	76	67.6
3	FEMALE	60	67.6
4	MALE	64	67.6
5	FEMALE	68	67.6

ID	WT
1	70
2	76
3	60
4	64
5	68

df2 %>% select(GENDER:meanWT)

ID	GENDER	WT	meanWT
1	MALE	70	67.6
2	MALE	76	67.6
3	FEMALE	60	67.6
4	MALE	64	67.6
5	FEMALE	68	67.6

GENDER	WT	meanWT
MALE	70	67.6
MALE	76	67.6
FEMALE	60	67.6
MALE	64	67.6
FEMALE	68	67.6

df %>% select(<function>(<values>))

df with the following columns:

WEIGHT	WEIGHT_KG	MEAN_WEIGHT	OCC1	OCC2	OCC3	OCC4	HEIGHT

function	meaning	example	columns selected
starts_with	names start with	starts_with("WEIGHT"))	WEIGHT, WEIGHT_KG
ends_with	names ends with	ends_with("GHT")	WEIGHT, MEAN_WEIGHT, HEIGHT
contains	names contains	contains("EI")	WEIGHT, WEIGHT_KG, MEAN_WEIGHT, HEIGHT
matches	regular expression matching	matches("_")	WEIGHT_KG, MEAN_WEIGHT
num_range	specify range of columns with consistent names with numeric suffix	num_range("OCC",1:3)	OCC1, OCC2, OCC3

```
test_select <-
data.frame("WEIGHT" =0, "WEIGHT_KG"=0,
"MEAN_WEIGHT"=0, "OCC1"=0, "OCC2"=0,
    "OCC3"=0, "OCC4"=0, "HEIGHT"=0)
```

```
test_select %>% select(starts_with("WEIGHT"))
test_select %>% select(matches("_"))
test_select %>% select(num_range("OCC", 1:3)
```

```
df2 %>% select(ID, WEIGHT = WT)
```

ID	GENDER	WT	meanWT
1	MALE	70	67.6
2	MALE	76	67.6
3	FEMALE	60	67.6
4	MALE	64	67.6
5	FEMALE	68	67.6

ID	WEIGHT
1	70
2	76
3	60
4	64
5	68

```
df2 %>% rename(WEIGHT = WT)
```

ID	GENDER	WT	meanWT
1	MALE	70	67.6
2	MALE	76	67.6
3	FEMALE	60	67.6
4	MALE	64	67.6
5	FEMALE	68	67.6

ID	GENDER	WEIGHT	meanWT
1	MALE	70	67.6
2	MALE	76	67.6
3	FEMALE	60	67.6
4	MALE	64	67.6
5	FEMALE	68	67.6

Verb	Usage
filter	Keep matching row criteria
summarize	reduces summary values calculated
mutate	add new variables to existing data frame
select	select columns by name
arrange	reorder rows

df %>% arrange(WT)

ID	GENDER	WT
1	MALE	70
2	MALE	76
3	FEMALE	60
4	MALE	64
5	FEMALE	68

ID	GENDER	WT
3	FEMALE	60
4	MALE	64
5	FEMALE	68
1	MALE	70
2	MALE	76

lowest weight

highest weight

df %>% arrange(desc(WT))

ID	GENDER	WT
1	MALE	70
2	MALE	76
3	FEMALE	60
4	MALE	64
5	FEMALE	68

ID	GENDER	WT
2	MALE	76
1	MALE	70
5	FEMALE	68
4	MALE	64
3	FEMALE	60

highest weight

lowest weight

```
dosing_df <- data.frame(ID = 1:2, TIME =0, AMT = 100, MDV = 1)
sample_df <- data.frame(expand.grid(ID = 1:2, TIME = seq(0, 2,
1),
                        AMT = 0, MDV = 0)
df3 <- bind_rows(sample_df, dosing_df)
```

ID	TIME	AMT	MDV
1	0	0	0
2	0	0	0
1	1	0	0
2	1	0	0
1	2	0	0
2	2	0	0
1	0	100	1
2	0	100	1

df3

* expand.grid is a very handy function for generating permutations

* MDV = missing dependent variable - a nonmem-style flag column

df3 %>% arrange(ID, TIME)

ID	TIME	AMT	MDV
1	0	0	0
2	0	0	0
1	1	0	0
2	1	0	0
1	2	0	0
2	2	0	0
1	0	100	1
2	0	100	1

ID	TIME	AMT	MDV
1	0	0	0
1	0	100	1
1	1	0	0
1	2	0	0
2	0	0	0
2	0	100	1
2	1	0	0
2	2	0	0

```
df3 %>% arrange(ID, TIME, desc(MDV))
```

ID	TIME	AMT	MDV
1	0	0	0
2	0	0	0
1	1	0	0
2	1	0	0
1	2	0	0
2	2	0	0
1	0	100	1
2	0	100	1

ID	TIME	AMT	MDV
1	0	100	1
1	0	0	0
1	1	0	0
1	2	0	0
2	0	100	1
2	0	0	0
2	1	0	0
2	2	0	0

minor data manipulation verbs

Verb	Usage
distinct	Keep all distinct elements per key
slice	return certain rows by number, group aware. Must be integer values
rename	rename columns
transmute	similar to mutate, however all columns not involved in calculation (either grouping or explicit calculations) are dropped

```
Theoph %>%
distinct(Subject)
```

Subject
3
6
7
8
11

```
Theoph %>%
distinct(Subject,
         .keep_all = TRUE)
```

Subject	Wt	Dose	Time	conc
3	70.5	4.53	0	0.00
6	80.0	4.00	0	0.00
7	64.6	4.95	0	0.15
8	70.5	4.53	0	0.00
11	65.0	4.92	0	0.00

```
Theoph %>%
group_by(Subject) %>%
slice(1:2)
```

Subject	Wt	Dose	Time	conc
6	80.0	4.00	0.00	0.00
6	80.0	4.00	0.27	1.29
7	64.6	4.95	0.00	0.15
7	64.6	4.95	0.25	0.85
8	70.5	4.53	0.00	0.00
8	70.5	4.53	0.25	3.05
11	65.0	4.92	0.00	0.00
11	65.0	4.92	0.25	4.86
3	70.5	4.53	0.00	0.00
3	70.5	4.53	0.27	4.40

```
Theoph %>%
group_by(Subject) %>%
slice(c(1, n()))
```

Subject	Wt	Dose	Time	conc
6	80.0	4.00	0.00	0.00
6	80.0	4.00	23.85	0.92
7	64.6	4.95	0.00	0.15
7	64.6	4.95	24.22	1.15
8	70.5	4.53	0.00	0.00
8	70.5	4.53	24.12	1.25
11	65.0	4.92	0.00	0.00
11	65.0	4.92	24.08	0.86
3	70.5	4.53	0.00	0.00
3	70.5	4.53	24.17	1.05

```
Theoph %>% rename(ID = Subject)
```

ID	Wt	Dose	Time	conc
3	70.5	4.53	0.00	0.00
3	70.5	4.53	0.27	4.40

Ifelse() vs case_when()

```
race <- data.frame(
    id=1:5,
    racen=c(0:3,6)
)


race %>%
mutate(
racec = ifelse(racen==0,"Caucasian",
ifelse(racen==1,"Black",
ifelse(racen==2,"Asian",
ifelse(racen==3,"Hispanic","missing")))
))
```

id	racen	racec
1	0	Caucasian
2	1	Black
3	2	Asian
4	3	Hispanic
5	NA	missing

Ifelse() vs **case_when()**

```
race %>%
mutate(
   racec = case_when(
      racen == 0 ~ "Caucasian",
      racen == 1 ~ "Black",
      racen == 2 ~ "Asian",
      racen == 3 ~ "Hispanic",
      racen == 4 ~ "Other",
      TRUE ~ "missing"
   )
)
```

id	racen	racec
1	0	Caucasian
2	1	Black
3	2	Asian
4	3	Hispanic
5	NA	missing

```
race %>%
mutate(
racec = ifelse(
        racen==0,"Caucasian",
ifelse(racen==1,"Black",
ifelse(racen==2,"Asian",
ifelse(racen==3,"Hispanic",
"missing")
))))
```

```
race %>%
mutate(
   racec = case_when(
      racen == 0 ~ "Caucasian",
      racen == 1 ~ "Black",
      racen == 2 ~ "Asian",
      racen == 3 ~ "Hispanic",
      TRUE ~ "missing"
   )
)
```

id	racen	raceifelse	racecasewhen
1	0	Caucasian	Caucasian
2	1	Black	Black
3	2	Asian	Asian
4	3	Hispanic	Hispanic
5	NA	NA	missing

```
race %>%
mutate(
racec =
ifelse(is.na(racen), "missing",
ifelse(racen==0,"Caucasian",
ifelse(racen==1,"Black",
ifelse(racen==2,"Asian",
ifelse(racen==3,"Hispanic",
"missing")
)))))
```

dplyr joins

Join	Usage
inner_join	return all rows from x where there are matching values in y, and all columns from x and y.
left_join	return all rows from x, and all columns from x and y.
semi_join	return all rows from x where there are matching values in y, keeping just columns from x.
anti_join	return all rows from x where there are not matching values in y, keeping just columns from x.
full_join	returns all rows and columns from x and y, with NA values for non-matching values from either.

* A semi join differs from an inner join because an inner join will return one row of x for each matching row of y, where a semi join will never duplicate rows of x

```
idtime <- data.frame(expand.grid(ID = as.numeric(1:3),
TIME = c(0,1))) %>% arrange(ID)
idwt <- data.frame(ID = c(1, 2, 4), WT = c(70, 80, 75))
```

ID	TIME
1	0
1	1
2	0
2	1
3	0
3	1

idtime

ID	WT
1	70
2	80
4	75

idwt

<join>(x_df,y_df)

INNER JOIN

idtime/idwt => in both

ID	TIME
1	0
1	1
2	0
2	1
3	0
3	1

idtime

ID	WT
1	70
2	80
4	75

idwt

inner_join(idtime, idwt)

ID	TIME	WT
1	0	70
1	1	70
2	0	80
2	1	80

ID	WT	TIME
1	70	0
1	70	1
2	80	0
2	80	1

inner_join(idwt, idtime)

LEFT JOIN

idtime/idwt => in both

ID	TIME
1	0
1	1
2	0
2	1
3	0
3	1

idtime

ID	WT
1	70
2	80
4	75

idwt

left_join(idtime, idwt)

ID	TIME	WT
1	0	70
1	1	70
2	0	80
2	1	80
3	0	NA
3	1	NA

ID	WT	TIME
1	70	0
1	70	1
2	80	0
2	80	1
4	75	NA

left_join(idwt, idtime)

SEMI JOIN

idtime/idwt => in both

ID	TIME
1	0
1	1
2	0
2	1
3	0
3	1

idtime

ID	WT
1	70
2	80
4	75

idwt

semi_join(idtime, idwt)

ID	TIME
1	0
1	1
2	0
2	1

ID	WT
1	70
2	80

semi_join(idwt, idtime)

ANTI JOIN

idtime/idwt => in both

ID	TIME
1	0
1	1
2	0
2	1
3	0
3	1

idtime

ID	WT
1	70
2	80
4	75

idwt

anti_join(idtime, idwt)

ID	TIME
3	0
3	1

ID	WT
4	75

anti_join(idwt, idtime)

FULL JOIN

idtime/idwt => in both

ID	TIME
1	0
1	1
2	0
2	1
3	0
3	1

idtime

ID	WT
1	70
2	80
4	75

idwt

full_join(idtime, idwt)

ID	TIME	WT
1	0	70
1	1	70
2	0	80
2	1	80
3	0	NA
3	1	NA
4	NA	75

full_join(idwt, idtime)

ID	WT	TIME
1	70	0
1	70	1
2	80	0
2	80	1
3	NA	0
3	NA	1
4	75	NA

Operate on a selection of variables

```
df <- data.frame(ID = 1:5,
GENDER = c("MALE", "MALE", "FEMALE", "MALE", "FEMALE"),
WT = c(70, 76, 60, 64, 68),
AGE = c(55,52,48,37,70))
```

ID	GENDER	WT	AGE
1	MALE	70	55
2	MALE	76	52
3	FEMALE	60	48
4	MALE	64	37
5	FEMALE	68	70

_all – apply an operation on all variables

```
df %>%
group_by(GENDER) %>%
select(-ID) %>%
summarise_all(mean)
```

GENDER	WT	AGE
FEMALE	64	59
MALE	70	48

_at – apply an operation on a subset of variables

```
df %>%
group_by(GENDER) %>%
summarise_at(vars(WT,AGE),mean)
```

GENDER	WT	AGE
FEMALE	64	59
MALE	70	48

_at – apply an operation on a subset of variables

```
df %>% group_by(GENDER) %>%
summarise_at(vars(WT,AGE),funs(mean,median,n())) %>%
select(GENDER,
contains("mean"),
ends_with("median"),
everything())
```

GENDER	WT_mean	AGE_mean	WT_median	AGE_median	WT_n	AGE_n
FEMALE	64	59	64	59	2	2
MALE	70	48	70	52	3	3

_if – apply an operation on the subset of variables for which a predicate function returns TRUE

```
df %>%  select(-ID)  %>%
summarise_if(is.numeric, funs(mean,median))
```

WT_mean	AGE_mean	WT_median	AGE_median
67.6	52.4	68	52

windows functions

“A **window function** is a variation on an **aggregation function**.”

- Where an aggregation function, like sum() and mean(), takes n inputs and return a single value
- a window function returns n values

cumsum()	row_number()	lead()
cummin()	min_rank()	lag()
cummax()	dense_rank()	
cumall()	cume_dist()	
cumany()	percent_rank()	
cummean()	ntile()	

```
df %>%
    mutate(cum_max = cummax(event))
```

visit	event	cum_max
1	1	1
2	3	3
3	5	5
4	4	5
5	4	5
6	2	5
7	7	7
8	6	7
9	6	7
10	4	7