R for Health Data Science

Week 06: Tidyverse and Pipes

Sam Stewart

2021-03-26

```
library(Hmisc)
library(kableExtra)
library(tidyverse)
library(ggplot2)

dat = read.csv("data/framinghamFirst.csv",header=TRUE,
               na.strings=".",stringsAsFactors=FALSE)
dat$BMIGroups = cut(dat$BMI,breaks=c(0,18.5,25,30,Inf),
                    labels=c("Underweight","Normal","Overweight","Obese"))
dat$SEX = factor(dat$SEX,levels=1:2,labels=c("Male","Female"))
dat$DIABETES = factor(dat$DIABETES,levels=0:1,labels=c("No Diabetes","Diabetes"))
dat$HYPERTEN = factor(dat$HYPERTEN,levels=0:1,labels=c("Normotensive","Hypertensive"))
```

Today we're going to be learning about the *tidyverse*, a set of libraries developed by Wickham and colleagues at RStudio. There are two major components that we'll focus on. This week we'll look at data manipulation with pipes (`%>%`), and next week we'll look at data visualizations with the `ggplot2` library.

We'll be borrowing heavily from the R4DS textbook - if you like learning from print sources there are few R-related textbooks more useful that this one. We'll be focusing on Chapters 5 and 18 this week, while chapter 3 will be the focus next week.

1 `dplyr` and data manipulation

The `dplyr` library (pronounced either *dip-ler* or *d-plier*) is the basis for data manipulation within the tidyverse, and is dependent on the pipe operator, `%>%`. Pipes work by passing the results of a function to the first argument in another function. The following two lines produce the same results

```
table(dat$SEX,dat$HYPERTEN)
```

```
##
##          Normotensive Hypertensive
##   Male            540         1404
##   Female          642         1848
```

```
dat$SEX %>% table(dat$HYPERTEN)
```

```
##
## .        Normotensive Hypertensive
##   Male            540         1404
##   Female          642         1848
```

In the back end R takes a command like `x %>% table(y)` and turns it into `table(x,y)`. Multiple pipes just continue the pattern, so that `x %>% f(y) %>% g(z,arg1,arg2,...)` becomes `f(x,y) %>% g(z,arg1,arg2)`

and then finally `g(f(x,y),z,arg1,arg2)`.

While pipes seem like a neat trick, their true power is unlocked by the other functions in the `dplyr` library that are designed to work with them. The functions in this library all take a dataset as the first argument, then use the following arguments to change the dataset. This changed dataset is then passed to the next function. This allows you to use functions sequentially and read them in the order they execute: take `x`, apply function `f(y)`, then apply function `g(z,arg1,arg2,...)`. See Chapter 18 for a good justification of why pipes make code easier to read.

The result is a vertical arrangement of functions that are much easier to read than traditional R, and require *way* fewer `$`. The functions we'll start with are

1. `filter()` to select specific rows
2. `select()` to pick specific columns (i.e variables)
3. `mutate()` to create new variables
4. `summarise()` and `group_by()` to create numeric summaries

There are plenty of others, but once you get the hang of it you'll learn how to Google the answer to the others: "dyplr how do I . . . ".

All of the functions (the tidyverse calls them *"verbs"*) work the same way

- The first argument is a data frame
- the subsequent arguments are what to do with the data frame
- the function returns a new data frame

The only other difference is that the data isn't a data frame, it's a **tibble**.

1.1 tibbles

Tibbles are a new kind of data frame specific to the tidyverse. They are almost identical to data frames, with a couple of notable differences (see the command `vignette("tibble")` for a complete list).

- They print a restricted number of rows and columns
- They don't partially match to variables names
- They don't change column names or change variables types
- They *try* not to support row names

```
tibble(dat)
```

```
## # A tibble: 4,434 x 39
##    SEX   RANDID TOTCHOL   AGE SYSBP DIABP CURSMOKE CIGPDAY   BMI DIABETES BPMEDS
##    <fct>  <int>   <int> <int> <dbl> <dbl>    <int>   <int> <dbl> <fct>     <int>
##  1 Male    2448     195    39  106     70        0       0  27.0 No Diab~      0
##  2 Fema~   6238     250    46  121     81        0       0  28.7 No Diab~      0
##  3 Male    9428     245    48  128.    80        1      20  25.3 No Diab~      0
##  4 Fema~  10552     225    61  150     95        1      30  28.6 No Diab~      0
##  5 Fema~  11252     285    46  130     84        1      23  23.1 No Diab~      0
##  6 Fema~  11263     228    43  180    110        0       0  30.3 No Diab~      0
##  7 Fema~  12629     205    63  138     71        0       0  33.1 No Diab~      0
##  8 Fema~  12806     313    45  100     71        1      20  21.7 No Diab~      0
##  9 Male   14367     260    52  142.    89        0       0  26.4 No Diab~      0
## 10 Male   16365     225    43  162    107        1      30  23.6 No Diab~      0
## # ... with 4,424 more rows, and 28 more variables: HEARTRTE <int>,
## #   GLUCOSE <int>, PREVCHD <int>, PREVAP <int>, PREVMI <int>, PREVSTRK <int>,
## #   PREVHYP <int>, TIME <int>, PERIOD <int>, HDLC <lgl>, LDLC <lgl>,
## #   DEATH <int>, ANGINA <int>, HOSPMI <int>, MI_FCHD <int>, ANYCHD <int>,
## #   STROKE <int>, CVD <int>, HYPERTEN <fct>, TIMEAP <int>, TIMEMI <int>,
## #   TIMEMIFC <int>, TIMECHD <int>, TIMESTRK <int>, TIMECVD <int>,
```

```
## #   TIMEDTH <int>, TIMEHYP <int>, BMIGroups <fct>
```

```
dat = tibble(dat)
```

1.2 filter()

`filter()` will let you subset a dataset based on values of other variables. As with all the functions here the first argument is the dataset, and the subsequent arguments are "expressions", or boolean arguments, that define the new dataset

```
#the full dataset
dat
```

```
## # A tibble: 4,434 x 39
##    SEX   RANDID TOTCHOL   AGE SYSBP DIABP CURSMOKE CIGPDAY   BMI DIABETES BPMEDS
##    <fct>  <int>   <int> <int> <dbl> <dbl>    <int>   <int> <dbl> <fct>     <int>
##  1 Male    2448     195    39  106     70        0       0  27.0 No Diab~      0
##  2 Fema~   6238     250    46  121     81        0       0  28.7 No Diab~      0
##  3 Male    9428     245    48  128.    80        1      20  25.3 No Diab~      0
##  4 Fema~  10552     225    61  150     95        1      30  28.6 No Diab~      0
##  5 Fema~  11252     285    46  130     84        1      23  23.1 No Diab~      0
##  6 Fema~  11263     228    43  180    110        0       0  30.3 No Diab~      0
##  7 Fema~  12629     205    63  138     71        0       0  33.1 No Diab~      0
##  8 Fema~  12806     313    45  100     71        1      20  21.7 No Diab~      0
##  9 Male   14367     260    52  142.    89        0       0  26.4 No Diab~      0
## 10 Male   16365     225    43  162    107        1      30  23.6 No Diab~      0
## # ... with 4,424 more rows, and 28 more variables: HEARTRTE <int>,
## #   GLUCOSE <int>, PREVCHD <int>, PREVAP <int>, PREVMI <int>, PREVSTRK <int>,
## #   PREVHYP <int>, TIME <int>, PERIOD <int>, HDLC <lgl>, LDLC <lgl>,
## #   DEATH <int>, ANGINA <int>, HOSPMI <int>, MI_FCHD <int>, ANYCHD <int>,
## #   STROKE <int>, CVD <int>, HYPERTEN <fct>, TIMEAP <int>, TIMEMI <int>,
## #   TIMEMIFC <int>, TIMECHD <int>, TIMESTRK <int>, TIMECVD <int>,
## #   TIMEDTH <int>, TIMEHYP <int>, BMIGroups <fct>
```

```
#just the men
filter(dat,SEX=='Male')
```

```
## # A tibble: 1,944 x 39
##    SEX   RANDID TOTCHOL   AGE SYSBP DIABP CURSMOKE CIGPDAY   BMI DIABETES BPMEDS
##    <fct>  <int>   <int> <int> <dbl> <dbl>    <int>   <int> <dbl> <fct>     <int>
##  1 Male    2448     195    39  106     70        0       0  27.0 No Diab~      0
##  2 Male    9428     245    48  128.    80        1      20  25.3 No Diab~      0
##  3 Male   14367     260    52  142.    89        0       0  26.4 No Diab~      0
##  4 Male   16365     225    43  162    107        1      30  23.6 No Diab~      0
##  5 Male   20375     294    46  142     94        1      15  26.3 No Diab~      0
##  6 Male   33077     232    48  138     90        1      10  22.4 No Diab~      0
##  7 Male   36459     195    41  139     88        0       0  26.9 No Diab~      0
##  8 Male   47561     270    44  138.    90        1      30  22.0 No Diab~      0
##  9 Male   54224     294    47  102     68        1      20  24.2 No Diab~      0
## 10 Male   63156     225    35  132     91        1      20  26.1 No Diab~      0
## # ... with 1,934 more rows, and 28 more variables: HEARTRTE <int>,
## #   GLUCOSE <int>, PREVCHD <int>, PREVAP <int>, PREVMI <int>, PREVSTRK <int>,
## #   PREVHYP <int>, TIME <int>, PERIOD <int>, HDLC <lgl>, LDLC <lgl>,
## #   DEATH <int>, ANGINA <int>, HOSPMI <int>, MI_FCHD <int>, ANYCHD <int>,
## #   STROKE <int>, CVD <int>, HYPERTEN <fct>, TIMEAP <int>, TIMEMI <int>,
## #   TIMEMIFC <int>, TIMECHD <int>, TIMESTRK <int>, TIMECVD <int>,
```

```
## #   TIMEDTH <int>, TIMEHYP <int>, BMIGroups <fct>
```

```
#just the women, but using a pipe
dat %>% filter(SEX=='Female')
```

```
## # A tibble: 2,490 x 39
##    SEX   RANDID TOTCHOL   AGE SYSBP DIABP CURSMOKE CIGPDAY   BMI DIABETES BPMEDS
##    <fct>  <int>   <int> <int> <dbl> <dbl>    <int>   <int> <dbl> <fct>     <int>
##  1 Fema~   6238     250    46   121    81        0       0  28.7 No Diab~      0
##  2 Fema~  10552     225    61   150    95        1      30  28.6 No Diab~      0
##  3 Fema~  11252     285    46   130    84        1      23  23.1 No Diab~      0
##  4 Fema~  11263     228    43   180   110        0       0  30.3 No Diab~      0
##  5 Fema~  12629     205    63   138    71        0       0  33.1 No Diab~      0
##  6 Fema~  12806     313    45   100    71        1      20  21.7 No Diab~      0
##  7 Fema~  16799     254    50   133    76        0       0  22.9 No Diab~      0
##  8 Fema~  19304     247    43   131    88        0       0  27.6 No Diab~      0
##  9 Fema~  23727     332    41   124    88        0       0  31.3 No Diab~      1
## 10 Fema~  24721     226    39   114    64        1       9  22.4 No Diab~      0
## # ... with 2,480 more rows, and 28 more variables: HEARTRTE <int>,
## #   GLUCOSE <int>, PREVCHD <int>, PREVAP <int>, PREVMI <int>, PREVSTRK <int>,
## #   PREVHYP <int>, TIME <int>, PERIOD <int>, HDLC <lgl>, LDLC <lgl>,
## #   DEATH <int>, ANGINA <int>, HOSPMI <int>, MI_FCHD <int>, ANYCHD <int>,
## #   STROKE <int>, CVD <int>, HYPERTEN <fct>, TIMEAP <int>, TIMEMI <int>,
## #   TIMEMIFC <int>, TIMECHD <int>, TIMESTRK <int>, TIMECVD <int>,
## #   TIMEDTH <int>, TIMEHYP <int>, BMIGroups <fct>
```

```
#Just the obese men
dat %>%
  filter(SEX=='Male',
         BMIGroups=='Obese')
```

```
## # A tibble: 232 x 39
##    SEX   RANDID TOTCHOL   AGE SYSBP DIABP CURSMOKE CIGPDAY   BMI DIABETES BPMEDS
##    <fct>  <int>   <int> <int> <dbl> <dbl>    <int>   <int> <dbl> <fct>     <int>
##  1 Male   82188     225    37  124.  92.5        0       0  38.5 No Diab~      0
##  2 Male   83398     178    52  160   98          0       0  40.1 Diabetes      0
##  3 Male  184857     274    41  152   90          1      43  30.6 No Diab~      0
##  4 Male  192229     285    39  155  110          0       0  32.5 No Diab~      0
##  5 Male  205391     286    62  118.  80          1      20  31.6 Diabetes      0
##  6 Male  231492     219    62  141   82          0       0  31.0 No Diab~      0
##  7 Male  276073     257    50  127   82          0       0  32.2 No Diab~      0
##  8 Male  364589     293    48  149  100          0       0  31.6 No Diab~      0
##  9 Male  411906     244    41  139   86          1      20  30.8 No Diab~      1
## 10 Male  471978     215    56  138   97          0       0  30.8 No Diab~      0
## # ... with 222 more rows, and 28 more variables: HEARTRTE <int>, GLUCOSE <int>,
## #   PREVCHD <int>, PREVAP <int>, PREVMI <int>, PREVSTRK <int>, PREVHYP <int>,
## #   TIME <int>, PERIOD <int>, HDLC <lgl>, LDLC <lgl>, DEATH <int>,
## #   ANGINA <int>, HOSPMI <int>, MI_FCHD <int>, ANYCHD <int>, STROKE <int>,
## #   CVD <int>, HYPERTEN <fct>, TIMEAP <int>, TIMEMI <int>, TIMEMIFC <int>,
## #   TIMECHD <int>, TIMESTRK <int>, TIMECVD <int>, TIMEDTH <int>, TIMEHYP <int>,
## #   BMIGroups <fct>
```

```
#Just the overweight or obese people
dat %>%
  filter(BMIGroups=="Obese" | BMIGroups=="Overweight")
```

```
## # A tibble: 2,422 x 39
##    SEX   RANDID TOTCHOL   AGE SYSBP DIABP CURSMOKE CIGPDAY   BMI DIABETES BPMEDS
##    <fct>  <int>   <int> <int> <dbl> <dbl>    <int>   <int> <dbl> <fct>     <int>
##  1 Male    2448     195    39  106     70        0       0  27.0 No Diab~      0
##  2 Fema~   6238     250    46  121     81        0       0  28.7 No Diab~      0
##  3 Male    9428     245    48  128.    80        1      20  25.3 No Diab~      0
##  4 Fema~  10552     225    61  150     95        1      30  28.6 No Diab~      0
##  5 Fema~  11263     228    43  180    110        0       0  30.3 No Diab~      0
##  6 Fema~  12629     205    63  138     71        0       0  33.1 No Diab~      0
##  7 Male   14367     260    52  142.    89        0       0  26.4 No Diab~      0
##  8 Fema~  19304     247    43  131     88        0       0  27.6 No Diab~      0
##  9 Male   20375     294    46  142     94        1      15  26.3 No Diab~      0
## 10 Fema~  23727     332    41  124     88        0       0  31.3 No Diab~      1
## # ... with 2,412 more rows, and 28 more variables: HEARTRTE <int>,
## #   GLUCOSE <int>, PREVCHD <int>, PREVAP <int>, PREVMI <int>, PREVSTRK <int>,
## #   PREVHYP <int>, TIME <int>, PERIOD <int>, HDLC <lgl>, LDLC <lgl>,
## #   DEATH <int>, ANGINA <int>, HOSPMI <int>, MI_FCHD <int>, ANYCHD <int>,
## #   STROKE <int>, CVD <int>, HYPERTEN <fct>, TIMEAP <int>, TIMEMI <int>,
## #   TIMEMIFC <int>, TIMECHD <int>, TIMESTRK <int>, TIMECVD <int>,
## #   TIMEDTH <int>, TIMEHYP <int>, BMIGroups <fct>
```

Note that filter only shows rows where the condition is TRUE, so any missing values in the condition column will also be dropped.

1.3 select()

`select()` will select specific columns of your dataset (where `filter()` selected rows). The general use is to reduce a dataset to just the needed variables, listed separated by commas after the first argument (which is again the dataset).

```
#pick three variables
select(dat,SEX,BMIGroups,CURSMOKE)
```

```
## # A tibble: 4,434 x 3
##    SEX    BMIGroups  CURSMOKE
##    <fct>  <fct>         <int>
##  1 Male   Overweight        0
##  2 Female Overweight        0
##  3 Male   Overweight        1
##  4 Female Overweight        1
##  5 Female Normal            1
##  6 Female Obese             0
##  7 Female Obese             0
##  8 Female Normal            1
##  9 Male   Overweight        0
## 10 Male   Normal            1
## # ... with 4,424 more rows
```

There are a couple of helper functions for complex datasets (`starts_with()`, `ends_with()`, `contains()`, `matches()`, ...), see `help(select)` for more details.

1.4 mutate()

`mutate()` is used for creating new variables. The equations are the same data-processing equations as before, but the advantage is that there are no more `$`, making it much easier to read.

```
dat %>%
  select(SEX,AGE,BMI,DEATH)%>%#not necessary, for demo purposes
  mutate(DEATH=factor(DEATH,levels=0:1,labels=c("Dead","Alive")),
         BMIGroups = cut(BMI,breaks=c(0,18.5,25,30,Inf)),
         BMINormal = (BMIGroups=='(18.5,25]')
         )
```

```
## # A tibble: 4,434 x 6
##    SEX      AGE   BMI DEATH BMIGroups BMINormal
##    <fct>  <int> <dbl> <fct> <fct>     <lgl>
##  1 Male      39  27.0 Dead  (25,30]   FALSE
##  2 Female    46  28.7 Dead  (25,30]   FALSE
##  3 Male      48  25.3 Dead  (25,30]   FALSE
##  4 Female    61  28.6 Alive (25,30]   FALSE
##  5 Female    46  23.1 Dead  (18.5,25] TRUE
##  6 Female    43  30.3 Dead  (30,Inf]  FALSE
##  7 Female    63  33.1 Dead  (30,Inf]  FALSE
##  8 Female    45  21.7 Dead  (18.5,25] TRUE
##  9 Male      52  26.4 Dead  (25,30]   FALSE
## 10 Male      43  23.6 Dead  (18.5,25] TRUE
## # ... with 4,424 more rows
```

`mutate()` can either create new variables (`BMIGroups`) or overwrite existing ones (`DEATH`). As you can see in the example the function executes sequentially, so we can create a variable and then use it right away to create another new variable (`BMINormal`).

1.5 summarise()

`summarise()` reduces the dataset to a single row, which isn't that useful until you pair it with `group_by()`.

```
dat %>%
  summarise(averageAge = mean(AGE,na.rm=TRUE))
```

```
## # A tibble: 1 x 1
##   averageAge
##        <dbl>
## 1       49.9
```

```
dat %>%
  summarise(
    n = n(),
    mean = mean(BMI,na.rm=TRUE),
    sd = sd(BMI,na.rm=TRUE)
    )
```

```
## # A tibble: 1 x 3
##       n  mean    sd
##   <int> <dbl> <dbl>
## 1  4434  25.8  4.10
```

I know that BMI has some missing values, but the `n()` function can't account for that, so we should address them before the summarise function, which we'll do with `filter()`.

```
dat %>%
  filter(!is.na(BMI))%>%
  summarise(
    n = n(),
```

```
    mean = mean(BMI,na.rm=TRUE),
    sd = sd(BMI,na.rm=TRUE)
    )
```

```
## # A tibble: 1 x 3
##       n  mean    sd
##   <int> <dbl> <dbl>
## 1  4415  25.8  4.10
```

To get some really meaningful results we'll use `group_by()` to stratify the results, first by SEX, then SEX and smoking status

```
dat %>%
  filter(!is.na(BMI))%>%
  group_by(SEX)%>%
  summarise(
    n = n(),
    mean = mean(BMI,na.rm=TRUE),
    sd = sd(BMI,na.rm=TRUE)
    )
```

```
## # A tibble: 2 x 4
##   SEX        n  mean    sd
##   <fct>  <int> <dbl> <dbl>
## 1 Male    1939  26.2  3.41
## 2 Female  2476  25.6  4.56
```

```
dat %>%
  filter(!is.na(BMI))%>%
  group_by(SEX,CURSMOKE)%>%
  summarise(
    n = n(),
    mean = mean(BMI,na.rm=TRUE),
    sd = sd(BMI,na.rm=TRUE)
    )
```

```
## # A tibble: 4 x 5
## # Groups:   SEX [2]
##   SEX    CURSMOKE     n  mean    sd
##   <fct>     <int> <int> <dbl> <dbl>
## 1 Male          0   768  26.9  3.33
## 2 Male          1  1171  25.7  3.37
## 3 Female        0  1473  26.3  4.62
## 4 Female        1  1003  24.5  4.23
```

Note that we haven't yet factored smoking, so we'll do it here with `mutate()`

```
dat %>%
  filter(!is.na(BMI))%>%
  mutate(CURSMOKE=factor(CURSMOKE,levels=0:1,labels=c("Non-smoker","Smoker")))%>%
  group_by(SEX,CURSMOKE)%>%
  summarise(
    n = n(),
    mean = mean(BMI,na.rm=TRUE),
    sd = sd(BMI,na.rm=TRUE)
    )
```

```
## # A tibble: 4 x 5
## # Groups:   SEX [2]
##   SEX    CURSMOKE       n  mean    sd
##   <fct>  <fct>      <int> <dbl> <dbl>
## 1 Male   Non-smoker   768  26.9  3.33
## 2 Male   Smoker      1171  25.7  3.37
## 3 Female Non-smoker  1473  26.3  4.62
## 4 Female Smoker      1003  24.5  4.23
```

Finally, we'll go back to the kable library last week to make it pretty.

```
dat %>%
  filter(!is.na(BMI))%>%
  mutate(CURSMOKE=factor(CURSMOKE,levels=0:1,labels=c("Non-smoker","Smoker")))%>%
  group_by(SEX,CURSMOKE)%>%
  summarise(
    n = n(),
    mean = mean(BMI,na.rm=TRUE),
    sd = sd(BMI,na.rm=TRUE)
    ) %>%
  kable(digits=2) %>% kable_styling()%>%
  kable_paper("striped",full_width=FALSE)%>%
  row_spec(0,bold=T,background='black',color='white')
```

SEX	CURSMOKE	n	mean	sd
Male	Non-smoker	768	26.90	3.33
Male	Smoker	1171	25.69	3.37
Female	Non-smoker	1473	26.35	4.62
Female	Smoker	1003	24.48	4.23

This kind of data-transformation for reporting is where `dplyr` can really shine - I find that I still do my data cleaning as individual commands, since doing it all as a single command with `dplyr` can make debugging more difficult, but I find table prep is much easier with `dplyr`.

2 Breakout Activity

We're going to go back to our week 2 dataset to practice with pipes.

In week 2 we took data from two datasets, `bpSubjects.csv` and `bpMeasures.csv`. The first records the age, sex and enrollment date for 100 subjects in a study to track the systolic blood pressures of LTC residents over 6 months. The second dataset records the BP measurements taken at 4 time intervals, roughly 1, 2, 3 and 6 months after enrollment.

1. Format the variables - numbers, factors and dates
2. Join the two datasets into a single dataset
3. Transform the data into a long dataset
    - start with a single long variable recording the BP values
    - as a second step, make it a long dataset that records both the BP and the date for each measurement

The code below has the non-piped solution.

```
dat01 = read.csv("data/bpSubjects.csv")
dat02 = read.csv("data/bpMeasures.csv",na.strings='.')
```

```r
####fixing dat01
#dates
dat01$enrollDate = as.Date(dat01$enrollDate)
#sex
dat01$sex = factor(toupper(dat01$sex),
             levels=c("M","F"),
             labels=c("Male","Female"))

###fixing dat02
#dates
dat02$sampleDate01 = as.Date(dat02$sampleDate01)
dat02$sampleDate02 = as.Date(dat02$sampleDate02)
dat02$sampleDate03 = as.Date(dat02$sampleDate03)
dat02$sampleDate04 = as.Date(dat02$sampleDate04)
#fixing BP
dat02$sysbp02 = as.numeric(dat02$sysbp02)

#I'll use left_join
dat.all = left_join(dat01, dat02, all.x = TRUE, all.y = FALSE)

library(reshape2)
dat.bp = melt(dat.all,
              id.vars=c("id","sex","age"),
              measure.vars=c("sysbp01","sysbp02","sysbp03","sysbp04"),
              variable.name='index',
              value.name ='BP')

dat.dates = melt(dat.all,
                 id.vars=c("id","sex","age"),
                 measure.vars=sprintf("sampleDate%02d",1:4),
                 variable.name='index',
                 value.name='date')

#once each long dataset is completed we can merge them
#merge would work as well
dat.long = merge(dat.dates,dat.bp,all=TRUE)
```

For the piped solution a couple of hints

- it needs at least two commands - it can't work as a single set of pipes
  - format `dat01` using pipes, then format `dat02`, then join them
- for joining consider `merge()`, `left_join()` and/or `right_join()`
- when formatting, be careful with nested functions (i.e `toupper()`), they don't work well with `mutate()`
- for the making it longer there's a function called `pivot_longer()` that might be helpful