% Managing Data with `dplyr` and `tidyr` % Biostatistics 140.776 %

dplyr

The data frame is a key data structure in statistics and in R.

- There is one observation per row
- Each column represents a variable or measure or characteristic
- Primary implementation that you will use is the default R implementation
- Other implementations, particularly relational databases systems

dplyr

- Developed by RStudio
- An optimized and distilled version of `plyr` package (by Hadley Wickham)
- Provides a "grammar" (in particular, verbs) for data manipulation
- Is **very** fast, as many key operations are coded in C++
- Functions (verbs) can be chained together via a "pipe" operator

dplyr Verbs

- `select`: return a subset of the columns of a data frame
- `filter`: extract a subset of rows from a data frame based on logical conditions
- `arrange`: reorder rows of a data frame
- `rename`: rename variables in a data frame
- `mutate`: add new variables/columns or transform existing variables
- `summarize` / `summarise`: generate summary statistics of different variables in the data frame, possibly within strata

There is also a handy `print` method that prevents you from printing a lot of data to the console.

dplyr Properties

- The first argument is a data frame.
- The subsequent arguments describe what to do with it, and you can refer to columns in the data frame directly without using the $ operator (just use the names).
- The result is a new data frame
- Data frames must be properly formatted and annotated for this to all be useful
- There are no "inputs" and "results"; it's all just *data*

Load the dplyr package

This step is important!

```
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

select

```
library(readr)
specdata <- read_csv("SPEC_2014.csv.gz")
dim(specdata)
```

```
[1] 1519790      26
```

```
head(select(specdata, 1:5))
```

```
# A tibble: 6 x 5
  State.Code County.Code Site.Num Parameter.Code
  <chr>      <chr>       <chr>              <int>
1 01         073         0023               88102
2 01         073         0023               88102
3 01         073         0023               88102
4 01         073         0023               88102
5 01         073         0023               88102
6 01         073         0023               88102
# ... with 1 more variable: POC <int>
```

select

```
names(specdata)[1:3]
```

```
[1] "State.Code"  "County.Code" "Site.Num"
```

```
head(select(specdata, State.Code:Site.Num))
```

```
# A tibble: 6 x 3
  State.Code County.Code Site.Num
  <chr>      <chr>       <chr>
1 01         073         0023
2 01         073         0023
3 01         073         0023
4 01         073         0023
5 01         073         0023
6 01         073         0023
```

select

In dplyr you can do

```
head(select(specdata, -(State.Code:Site.Num)))
```

Equivalent base R

```
i <- match("State.Code", names(specdata))
j <- match("Site.Num", names(specdata))
head(specdata[, -(i:j)])
```

filter

```
specdata.f <- filter(specdata, Parameter.Name == "Sulfate P
select(specdata.f, 1:3, Parameter.Name)

# A tibble: 36,012 x 4
   State.Code County.Code Site.Num Parameter.Name
   <chr>      <chr>       <chr>    <chr>
 1 01         073         0023     Sulfate PM2.5...
 2 01         073         0023     Sulfate PM2.5...
 3 01         073         0023     Sulfate PM2.5...
 4 01         073         0023     Sulfate PM2.5...
 5 01         073         0023     Sulfate PM2.5...
 6 01         073         0023     Sulfate PM2.5...
 7 01         073         0023     Sulfate PM2.5...
 8 01         073         0023     Sulfate PM2.5...
 9 01         073         0023     Sulfate PM2.5...
10 01         073         0023     Sulfate PM2.5...
# ... with 3.6e+04 more rows
```

filter

```
specdata.f <- filter(specdata, Parameter.Name == "Sulfate P
                       & State.Name == "Texas")
select(specdata.f, 1:3, Parameter.Name, State.Name)

# A tibble: 988 x 5
   State.Code County.Code Site.Num Parameter.Name
   <chr>      <chr>       <chr>    <chr>
 1 48         043         0101     Sulfate PM2.5...
 2 48         043         0101     Sulfate PM2.5...
 3 48         043         0101     Sulfate PM2.5...
 4 48         043         0101     Sulfate PM2.5...
 5 48         043         0101     Sulfate PM2.5...
 6 48         043         0101     Sulfate PM2.5...
 7 48         043         0101     Sulfate PM2.5...
 8 48         043         0101     Sulfate PM2.5...
 9 48         043         0101     Sulfate PM2.5...
10 48         043         0101     Sulfate PM2.5...
# ... with 978 more rows, and 1 more variable:
```

arrange

Reordering rows of a data frame (while preserving corresponding order of other columns) is normally a pain to do in R.

```
specdata <- arrange(specdata, Date.Local)
head(select(specdata, Date.Local, Parameter.Name, Sample.Va

# A tibble: 3 x 3
  Date.Local Parameter.Name           Sample.Value
  <date>     <chr>                           <dbl>
1 2014-01-01 EC CSN PM2.5 LC TOT             0.866
2 2014-01-01 Total Carbon PM2.5 LC ...         3.66
3 2014-01-01 Optical EC PM2.5 LC TOT         0.457

tail(select(specdata, Date.Local, Parameter.Name, Sample.Va

# A tibble: 3 x 3
  Date.Local Parameter.Name   Sample.Value
  <date>     <chr>                   <dbl>
1 2014-12-31 EC2 PM2.5 LC                0.002
2 2014-12-31 EC3 PM2.5 LC                0
```

arrange

Columns can be arranged in descending order too.

```
specdata <- arrange(specdata, desc(Date.Local))
head(select(specdata, Date.Local, Parameter.Name, Sample.Va

# A tibble: 3 x 3
  Date.Local Parameter.Name    Sample.Value
  <date>     <chr>                    <dbl>
1 2014-12-31 Antimony PM2.5 LC        0
2 2014-12-31 Arsenic PM2.5 LC         0.002
3 2014-12-31 Arsenic PM2.5 LC         0.002

tail(select(specdata, Date.Local, Parameter.Name, Sample.Va

# A tibble: 3 x 3
  Date.Local Parameter.Name          Sample.Value
  <date>     <chr>                          <dbl>
1 2014-01-01 Total Carbon PM2.5 LC ...        4.23
2 2014-01-01 Black Carbon PM2.5 at ...        0.180
3 2014-01-01 Optical EC PM2.5 LC TOT          0.707
```

rename

Renaming a variable in a data frame in R is surprising hard to do!

```
head(specdata[, 1:7], 3)
```

```
# A tibble: 3 x 7
  State.Code County.Code Site.Num Parameter.Code
  <chr>      <chr>       <chr>              <int>
1 01         073         0023               88102
2 01         073         0023               88103
3 01         073         0023               88103
# ... with 3 more variables: POC <int>,
#   Latitude <dbl>, Longitude <dbl>
```

```
specdata <- rename(specdata, lat = Latitude,
                   lon = Longitude)
head(specdata[, 1:7], 3)
```

```
# A tibble: 3 x 7
  State.Code County.Code Site.Num Parameter.Code
  <chr>      <chr>       <chr>              <int>
```

mutate

```
specdata <- mutate(specdata,
                   city_state = paste(City.Name, State.Name
                   sample_mg = Sample.Value / 1000)
head(select(specdata, City.Name, State.Name, city_state, sa
```

```
# A tibble: 6 x 4
  City.Name  State.Name city_state       sample_mg
  <chr>      <chr>      <chr>                <dbl>
1 Birmingham Alabama    Birmingham, Al... 0
2 Birmingham Alabama    Birmingham, Al... 0.000002
3 Birmingham Alabama    Birmingham, Al... 0.000002
4 Birmingham Alabama    Birmingham, Al... 0.000022
5 Birmingham Alabama    Birmingham, Al... 0.0000180
6 Birmingham Alabama    Birmingham, Al... 0.000013
```

group_by

Generating summary statistics by stratum

```
specdata <- mutate(specdata,
                   region = factor(lon > -100,
                                   labels = c("west", "east
eastwest <- group_by(specdata, region)
summarize(eastwest, pollutant = mean(Sample.Value, na.rm =
          obs = mean(Observation.Count, na.rm = TRUE))
```

```
# A tibble: 2 x 3
  region pollutant   obs
  <fct>      <dbl> <dbl>
1 west       0.153  1.07
2 east       0.406  1.26
```

group_by

Generating summary statistics by stratum

```
library(lubridate)
```

```
Attaching package: 'lubridate'

The following object is masked from 'package:base':

    date
```

```
specdata <- mutate(specdata, month = month(Date.Local))
months <- group_by(specdata, month)
months <- filter(months, Parameter.Name == "Sulfate PM2.5 I
summarize(months, sulfate = mean(Sample.Value, na.rm = TRUE
```

```
# A tibble: 12 x 2
   month sulfate
   <dbl>   <dbl>
 1     1   0.840
```

%>% (pipe operator)

```
specdata %>%
        mutate(month = month(Date.Local)) %>%
        filter(Parameter.Name == "Sulfate PM2.5 LC") %>%
        group_by(month) %>%
        summarize(sulfate = mean(Sample.Value, na.rm = TRUE

# A tibble: 12 x 2
   month sulfate
   <dbl>   <dbl>
 1     1   0.840
 2     2   1.31
 3     3   1.25
 4     4   1.18
 5     5   1.19
 6     6   1.30
 7     7   1.37
 8     8   1.37
 9     9   1.20
```

%>% (pipe operator)

```
specdata %>%
        mutate(month = month(Date.Local)) %>%
        filter(Parameter.Name == "Sulfate PM2.5 LC") %>%
        group_by(month, region) %>%
        summarize(sulfate = mean(Sample.Value, na.rm = TRU

# A tibble: 24 x 3
# Groups:   month [?]
   month region sulfate
   <dbl> <fct>    <dbl>
 1     1 west     0.468
 2     1 east     1.13
 3     2 west     0.549
 4     2 east     1.92
 5     3 west     0.537
 6     3 east     1.82
 7     4 west     0.666
 8     4 east     1.60
```

dplyr

Once you learn the dplyr "grammar" there are a few additional benefits

- dplyr can work with other data frame "backends"
- `data.table` for large fast tables
- SQL interface for relational databases via the DBI package

tidyr

The `tidyr` package helps with manipulation of data frames between "wide" and "long" formats, depending on what you're trying to do.

- Sometimes the meaning of a "variable" depends on the application
- Sometimes PM10, O3, NO2 are all different variables with continuous levels
- Sometimes "Pollutant" is the variable with levels "PM10", "O3", and "NO2"

Long Format

Here are the specdata pollution data in long format

```
head(spec)
```

```
# A tibble: 6 x 4
  city_state Date.Local Parameter.Name
  <chr>      <date>     <chr>
1 Birmingha... 2014-12-31 Antimony PM2....
2 Birmingha... 2014-12-31 Arsenic PM2.5...
3 Birmingha... 2014-12-31 Arsenic PM2.5...
4 Birmingha... 2014-12-31 Aluminum PM2....
5 Birmingha... 2014-12-31 Aluminum PM2....
6 Birmingha... 2014-12-31 Barium PM2.5 ...
# ... with 1 more variable: Sample.Value <dbl>
```

Long Format

```
stats <- group_by(spec, Parameter.Name) %>%
         summarize(mean = mean(Sample.Value, na.rm = TRUE),
                   median = median(Sample.Value, na.rm = TRU
                   max = max(Sample.Value, na.rm = TRUE))
stats
```

```
# A tibble: 86 x 4
   Parameter.Name                 mean median     max
   <chr>                         <dbl>  <dbl>   <dbl>
 1 Aluminum PM2.5 LC          5.30e-2 0.021    2.78
 2 Ammonium Ion PM2.5 LC      7.00e-1 0.447   15.2
 3 Antimony PM2.5 LC          5.17e-3 0        0.101
 4 Arsenic PM2.5 LC           3.33e-4 0        0.092
 5 Barium PM2.5 LC            2.42e-3 0        0.602
 6 Black Carbon PM2.5 at ... 5.33e-1 0.429    3.10
 7 Bromine PM2.5 LC           2.48e-3 0.0019   0.216
 8 Cadmium PM2.5 LC           1.49e-3 0        0.04
 9 Calcium PM2.5 LC           4.75e-2 0.0233   1.85
```

gather

An alternate representation could have three variables: pollutant, statistic, and value

- `gather` is a function that "gathers" multiple columns and essential sticks them into one column
- The names of multiple columns become levels of a single variable
- In this case mean, median, max –> levels of a "statistic" variable

gather

```
library(tidyr)
gather(stats, statistic, value, -Parameter.Name)
```

```
# A tibble: 258 x 3
   Parameter.Name                 statistic    value
   <chr>                          <chr>          <dbl>
 1 Aluminum PM2.5 LC              mean       0.0530
 2 Ammonium Ion PM2.5 LC          mean       0.700
 3 Antimony PM2.5 LC              mean       0.00517
 4 Arsenic PM2.5 LC               mean       0.000333
 5 Barium PM2.5 LC                mean       0.00242
 6 Black Carbon PM2.5 at 880 nm   mean       0.533
 7 Bromine PM2.5 LC               mean       0.00248
 8 Cadmium PM2.5 LC               mean       0.00149
 9 Calcium PM2.5 LC               mean       0.0475
10 Cerium PM2.5 LC                mean       0.000227
# ... with 248 more rows
```

`spread`

The `spread` function does the inverse of the `gather` function

- `spread` takes a single variable (with multiple levels) and *spreads* them across multiple columns
- Sometimes more intuitive if you want to compute a statistic across multiple levels/variables
- e.g. Compute the maximum of three different pollutants on each day and create a new variable

spread

```
wide <- filter(spec, city_state == "Essex, Maryland") %>%
        spread(Parameter.Name, Sample.Value)
head(wide)

# A tibble: 6 x 53
  city_state Date.Local `Aluminum PM2.5...
  <chr>      <date>                  <dbl>
1 Essex, Ma... 2014-01-05              0
2 Essex, Ma... 2014-01-08              0.016
3 Essex, Ma... 2014-01-11              0
4 Essex, Ma... 2014-01-14              0.013
5 Essex, Ma... 2014-01-17              0.004
6 Essex, Ma... 2014-01-23              0
# ... with 50 more variables: `Ammonium Ion PM2.5
#   LC` <dbl>, `Antimony PM2.5 LC` <dbl>,
#   `Arsenic PM2.5 LC` <dbl>, `Barium PM2.5
#   LC` <dbl>, `Bromine PM2.5 LC` <dbl>, `Cadmium
#   PM2.5 LC` <dbl>, `Calcium PM2.5 LC` <dbl>,
```

separate

Sometimes you need to split one column into two separate columns.

```
m
```

```
             mood    day
1  happy-working Sunday
2 happy-partying Sunday
3  happy-working Sunday
4   sad-partying Sunday
5    sad-working Monday
6   sad-partying Monday
```

separate

```
m <- separate(m, mood, c("state", "activity"), sep = "-")
m
```

```
  state activity    day
1 happy  working Sunday
2 happy partying Sunday
3 happy  working Sunday
4   sad partying Sunday
5   sad  working Monday
6   sad partying Monday
```

unite

The inverse of `separate`

```
unite(m, mood, state, activity, sep = "-")
```

```
             mood    day
1  happy-working Sunday
2 happy-partying Sunday
3  happy-working Sunday
4   sad-partying Sunday
5    sad-working Monday
6   sad-partying Monday
```

Summary

`dplyr`

- Verbs/functions for manipulating data frames in tidy format
- select, filter, arrange, group_by, summarize, rename, mutate

`tidyr`

- Transform data frames from wide to long formats
- spread, gather, separate, unite