Data Wrangling II

QBS Bootcamp 2025

Lesson Objectives

At the end of this lecture you should be able to:

1. Use pipes in dplyr
2. Subset data using dplyr
3. Move between wide and long data frames in tidyr
4. Generate simple summary tables

Resources

Cheat Sheet for Functions in dplyr: https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf

Pipes in tidyverse: https://style.tidyverse.org/pipes.html

```r
# Install all tidyverse associated packages
#install.packages('tidyverse')

# If you only want to install the packages we will use in this lecture:
#install.packages('dplyr')
#install.packages('tidyr')

# load tidyverse libraries
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(dplyr)
library(tidyr)
```

Data Set

We're going to start this lecture by generating the same random data set that we used in the last lecture. As always, don't forget to set a random seed so that our data is comparable across lectures.

```
# Set a random seed
set.seed(103)

# Define a data frame with our randomly generated data
randomData <- data.frame('SubjectID' = seq(1:1000),
                         'Systolic.BP' = rnorm(n = 1000,mean = 128,sd = 20),
                         'Diastolic.BP' = rnorm(n = 1000,mean = 71,sd = 10),
                         'Age' = trunc(runif(n = 1000,min = 18,max = 70)),
                         'Male' = rbinom(n = 1000,size = 1,prob = 0.5))

# Define binary variable for biological sex
randomData$BiologicalSex <- factor(ifelse(randomData$Male == 1,'Male','Female'))

# Define variable specifying age above 65 (medicare eligible)
randomData$MedicareAge <- ifelse(randomData$Age < 65,F,T)

head(randomData, 5)
```

```
##   SubjectID Systolic.BP Diastolic.BP Age Male BiologicalSex MedicareAge
## 1         1   112.28054     75.52894  52    0        Female       FALSE
## 2         2   129.09478     59.95778  56    0        Female       FALSE
## 3         3   104.54879     74.51568  25    1          Male       FALSE
## 4         4   124.65374     52.99577  41    0        Female       FALSE
## 5         5    90.69937     71.17388  41    0        Female       FALSE
```

Pipes

If you've looked at a lot of sample code online before, you've probably run into this syntax: %>%. This is a pipe! Pipes are used in tidyverse to keep code clean and prevent the defining of a lot of unnecessary intermediate variables. One of the main goals of this syntax is to keep a lot of white space in your code to help make it as readable as possible for anyone reading through your code.

Pipes will get more complex as we go through the lecture but first lets start of with something simple to start to see what they do. First, lets define a subset of our data that reflects only individuals eligible for medicare. Last lecture, we used the following syntax:

```
# Subset to only those at medicare age using our binary variable
medicareData <- randomData[which(randomData$MedicareAge == T),]
head(medicareData)
```

```
##    SubjectID Systolic.BP Diastolic.BP Age Male BiologicalSex MedicareAge
## 7          7    144.5196     66.34599  69    0        Female        TRUE
## 8          8    151.8032     51.76280  67    0        Female        TRUE
## 23        23    123.0903     61.61668  67    0        Female        TRUE
## 28        28    151.7242     73.91189  66    0        Female        TRUE
## 33        33    139.8668     84.28383  66    0        Female        TRUE
## 40        40    133.1999     61.29819  67    1          Male        TRUE
```

```r
# Subset to only those at medicare age using a continuous variable
medicareData <- randomData[randomData$Age >= 65,]
head(medicareData)
```

```
##    SubjectID Systolic.BP Diastolic.BP Age Male BiologicalSex MedicareAge
## 7          7    144.5196     66.34599  69    0        Female        TRUE
## 8          8    151.8032     51.76280  67    0        Female        TRUE
## 23        23    123.0903     61.61668  67    0        Female        TRUE
## 28        28    151.7242     73.91189  66    0        Female        TRUE
## 33        33    139.8668     84.28383  66    0        Female        TRUE
## 40        40    133.1999     61.29819  67    1          Male        TRUE
```

Now, we can generate the same data set using a pipe and the filter function in dplyr.

```r
# Subset without pipe
medicareData <- filter(randomData, Age >= 65)

head(medicareData)
```

```
##   SubjectID Systolic.BP Diastolic.BP Age Male BiologicalSex MedicareAge
## 1         7    144.5196     66.34599  69    0        Female        TRUE
## 2         8    151.8032     51.76280  67    0        Female        TRUE
## 3        23    123.0903     61.61668  67    0        Female        TRUE
## 4        28    151.7242     73.91189  66    0        Female        TRUE
## 5        33    139.8668     84.28383  66    0        Female        TRUE
## 6        40    133.1999     61.29819  67    1          Male        TRUE
```

```r
# Subset with a pipe
medicareData <- randomData %>%
  filter(Age >= 65)

head(medicareData)
```

```
##   SubjectID Systolic.BP Diastolic.BP Age Male BiologicalSex MedicareAge
## 1         7    144.5196     66.34599  69    0        Female        TRUE
## 2         8    151.8032     51.76280  67    0        Female        TRUE
## 3        23    123.0903     61.61668  67    0        Female        TRUE
## 4        28    151.7242     73.91189  66    0        Female        TRUE
## 5        33    139.8668     84.28383  66    0        Female        TRUE
## 6        40    133.1999     61.29819  67    1          Male        TRUE
```

Based on the the use of the filter function above, can you describe the syntax of how a pipe works?

Pipes might not seem too useful when we are only providing it a single function, but what if we want it to work through multiple steps?

```r
medicareData <- randomData %>%
  dplyr::filter(Age >= 65) %>%
  dplyr::select(SubjectID,Systolic.BP,Diastolic.BP,BiologicalSex,Age)

head(medicareData)
```

```
##   SubjectID Systolic.BP Diastolic.BP BiologicalSex Age
## 1         7    144.5196     66.34599        Female  69
## 2         8    151.8032     51.76280        Female  67
## 3        23    123.0903     61.61668        Female  67
## 4        28    151.7242     73.91189        Female  66
## 5        33    139.8668     84.28383        Female  66
## 6        40    133.1999     61.29819          Male  67
```

What is the select function doing?

```r
medicareData <- randomData %>%
  filter(Age >= 65) %>%
  select(SubjectID,Systolic.BP,Diastolic.BP,BiologicalSex,Age) %>%
  mutate(MedicareID = row_number()) %>%
  mutate(BP.Diff = Systolic.BP - Diastolic.BP)

head(medicareData)
```

```
##   SubjectID Systolic.BP Diastolic.BP BiologicalSex Age MedicareID   BP.Diff
## 1         7    144.5196     66.34599        Female  69          1  78.17358
## 2         8    151.8032     51.76280        Female  67          2 100.04039
## 3        23    123.0903     61.61668        Female  67          3  61.47363
## 4        28    151.7242     73.91189        Female  66          4  77.81236
## 5        33    139.8668     84.28383        Female  66          5  55.58301
## 6        40    133.1999     61.29819          Male  67          6  71.90172
```

Based on these examples, what is the mutate function doing?

Wide -> Long Data in *tidyverse*

Last class, we moved from a wide to a long data frame using the *melt* function in *reshape2*.

```r
# Melt the data frame into a long form
longData <- reshape2::melt(randomData[,c('SubjectID','Systolic.BP','Diastolic.BP','Age','BiologicalSex'
                           id.vars = c('SubjectID','Age','BiologicalSex'),value.name = 'BP',
                           variable.name = 'BP.Type')

head(longData)
```

```
##   SubjectID Age BiologicalSex     BP.Type        BP
## 1         1  52        Female Systolic.BP 112.28054
## 2         2  56        Female Systolic.BP 129.09478
## 3         3  25          Male Systolic.BP 104.54879
## 4         4  41        Female Systolic.BP 124.65374
## 5         5  41        Female Systolic.BP  90.69937
## 6         6  31        Female Systolic.BP 125.59120
```

In *dplyr*, we will use the the *gather* function or the *pivot_longer*.

```r
# use gather()
longData2 <- randomData %>%
  tidyr::gather(Systolic.BP,Diastolic.BP,key = BP.Type, value = BP)

head(randomData)
```

```
##   SubjectID Systolic.BP Diastolic.BP Age Male BiologicalSex MedicareAge
## 1         1   112.28054     75.52894  52    0        Female       FALSE
## 2         2   129.09478     59.95778  56    0        Female       FALSE
## 3         3   104.54879     74.51568  25    1          Male       FALSE
## 4         4   124.65374     52.99577  41    0        Female       FALSE
## 5         5    90.69937     71.17388  41    0        Female       FALSE
## 6         6   125.59120     69.50961  31    0        Female       FALSE
```

```r
head(longData2)
```

```
##   SubjectID Age Male BiologicalSex MedicareAge     BP.Type        BP
## 1         1  52    0        Female       FALSE Systolic.BP 112.28054
## 2         2  56    0        Female       FALSE Systolic.BP 129.09478
## 3         3  25    1          Male       FALSE Systolic.BP 104.54879
## 4         4  41    0        Female       FALSE Systolic.BP 124.65374
## 5         5  41    0        Female       FALSE Systolic.BP  90.69937
## 6         6  31    0        Female       FALSE Systolic.BP 125.59120
```

```r
# use pivot_longer()
longData3 <- randomData %>%
  pivot_longer(cols = c(Systolic.BP,Diastolic.BP),
                    names_to = 'BP.Type',
                    values_to = 'BP')

head(longData3)
```

```
## # A tibble: 6 x 7
##   SubjectID   Age  Male BiologicalSex MedicareAge BP.Type         BP
##       <int> <dbl> <int> <fct>         <lgl>       <chr>        <dbl>
## 1         1    52     0 Female        FALSE       Systolic.BP  112.
## 2         1    52     0 Female        FALSE       Diastolic.BP  75.5
## 3         2    56     0 Female        FALSE       Systolic.BP  129.
## 4         2    56     0 Female        FALSE       Diastolic.BP  60.0
## 5         3    25     1 Male          FALSE       Systolic.BP  105.
## 6         3    25     1 Male          FALSE       Diastolic.BP  74.5
```

What is different about the syntax we used here vs. what we used in the last class?

We can also use some more pipes to clean this up even more:

```r
longData <- randomData %>%
  # Convert to long format
  tidyr::gather(key = BP.Type, value = BP,c('Systolic.BP','Diastolic.BP')) %>%
  # Split into two separate variables
  tidyr::separate(col = BP.Type, into = c('BP.Type','Bad.ID')) %>%
  # Remove the bad ID variable
```

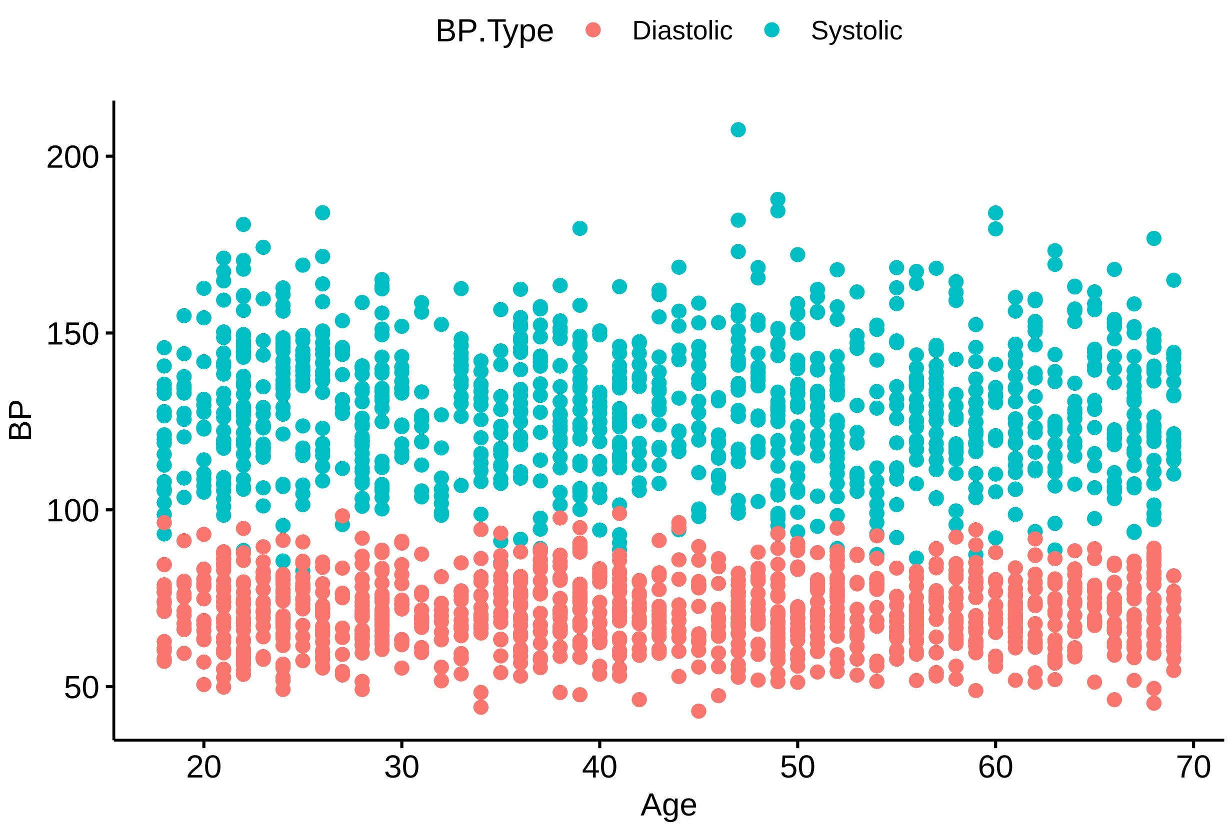
```r
  select(-Bad.ID)

head(longData)
```

```
##   SubjectID Age Male BiologicalSex MedicareAge  BP.Type        BP
## 1         1  52    0        Female       FALSE Systolic 112.28054
## 2         2  56    0        Female       FALSE Systolic 129.09478
## 3         3  25    1          Male       FALSE Systolic 104.54879
## 4         4  41    0        Female       FALSE Systolic 124.65374
## 5         5  41    0        Female       FALSE Systolic  90.69937
## 6         6  31    0        Female       FALSE Systolic 125.59120
```

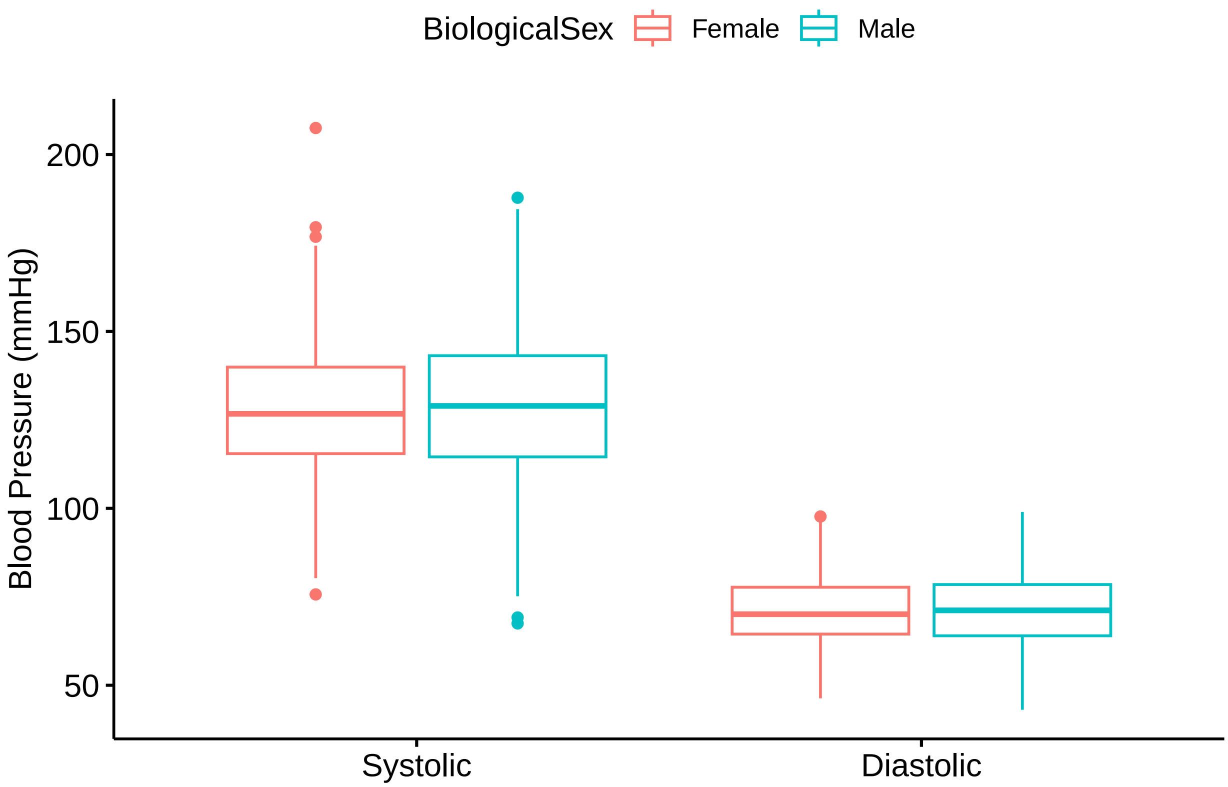
Now that our data is in a long format, we can generate plots with both measures of blood pressure in one plot.

```r
# Generate a scatter plot of age by systolic blood pressure
ggpubr::ggscatter(longData,
                  x = 'Age',
                  y = 'BP',
                  color = 'BP.Type')
```




```r
# Generate a boxplot for diastolic blood pressure distribution by biological sex in our original datase
ggpubr::ggboxplot(longData,
                  x = 'BP.Type',
                  y = 'BP',
                  color = 'BiologicalSex',
```

```r
ylab = 'Blood Pressure (mmHg)',
                  xlab = '')
```



Take a minute and comment what each step of this pipe is doing.

Long -> Wide Data in *tidyverse*

We can go back to a wide format in *tidyr* using the *spread* or *pivot_wider* function.

```r
# Convert using spread
wideData1 <- longData %>%
  tidyr::spread(key = BP.Type, value = BP)
head(wideData1)
```

```
##   SubjectID Age Male BiologicalSex MedicareAge Diastolic  Systolic
## 1         1  52    0        Female       FALSE  75.52894 112.28054
## 2         2  56    0        Female       FALSE  59.95778 129.09478
## 3         3  25    1          Male       FALSE  74.51568 104.54879
## 4         4  41    0        Female       FALSE  52.99577 124.65374
## 5         5  41    0        Female       FALSE  71.17388  90.69937
## 6         6  31    0        Female       FALSE  69.50961 125.59120
```

```r
# Convert using pivot_wider
wideData2 <- longData %>%
  tidyr::pivot_wider(names_from = BP.Type,
                     values_from = BP)
head(wideData2)
```

```
## # A tibble: 6 x 7
##   SubjectID   Age  Male BiologicalSex MedicareAge Systolic Diastolic
##       <int> <dbl> <int> <fct>         <lgl>          <dbl>     <dbl>
## 1         1    52     0 Female        FALSE          112.       75.5
## 2         2    56     0 Female        FALSE          129.       60.0
## 3         3    25     1 Male          FALSE          105.       74.5
## 4         4    41     0 Female        FALSE          125.       53.0
## 5         5    41     0 Female        FALSE           90.7      71.2
## 6         6    31     0 Female        FALSE          126.       69.5
```

And, just like in *reshape2* we can also create summary tables using the *group_by* and *summarise* functions.

```
summary <- longData %>%
  tidyr::spread(key = BP.Type, value = BP) %>%
  dplyr::mutate(MedicareAge = ifelse(Age >= 65,T,F)) %>%
  dplyr::group_by(BiologicalSex,MedicareAge) %>%
  dplyr::summarise(Mean.Age = mean(Age),Mean.Sys = mean(Systolic),Mean.Dias = mean(Diastolic))
```

```
## `summarise()` has grouped output by 'BiologicalSex'. You can override using the
## `.groups` argument.
```

```
summary
```

```
## # A tibble: 4 x 5
## # Groups:   BiologicalSex [2]
##   BiologicalSex MedicareAge Mean.Age Mean.Sys Mean.Dias
##   <fct>         <lgl>          <dbl>    <dbl>     <dbl>
## 1 Female        FALSE           41.0     128.      70.6
## 2 Female        TRUE            67.1     129.      71.1
## 3 Male          FALSE           41.0     129.      71.4
## 4 Male          TRUE            66.9     127.      70.3
```

In Class Exercises

1. Generate Random Data

Generate a random data set (remember to set a random seed) of 10,000 pregnant women with the following characteristics:

1. Age is uniformly distributed between 18 and 35 years old (Variable Name: Age).
2. There is a probability of 0.5 that each mother is carrying a female infant (Variable Name: InfantSex). This variable should be formatted as a factor variable with levels "Male" and "Female"
3. Define fasting glucose measures (Variable Name: Glucose1) as normally distributed. Mothers carrying a male infant have a mean score of 85 and a standard deviation of 6 mg/dL. Mothers carrying a female infant have a mean score of 80 and a standard deviation of 6 mg/dL.
4. Define 1 hour glucose measures (Variable Name: Glucose2) as normally distributed. Mothers carrying a male infant have a mean score of 165 and a standard deviation of 9 mg/dL. Mothers carrying a female infant have a mean score of 155 and a standard deviation of 9 mg/dL.

5. Define a summary variable for gestational diabetes (Variable Name: Diagnosis) which is "Gestational Diabetes" if either Glucose1 is higher than 95 or Glucose2 is higher than 180 and "Healthy" otherwise.

```
# code here
```

2. Summarize Random Data

Generate a summary table including age, fasting glucose, and one hour glucose of all subjects by both disease status and infant sex.

Your table should have 4 rows in the following order: Healthy & Female, Gestational Diabetes & Female, Healthy & Male, Gestational Diabetes & Male and should summarize mean age, mean and sd fasting glucose, and mean and sd one hour glucose.

```
# code here
```

3. Visualize Data

Generate a boxplot of the distribution of Glucose (y axis) for all subjects where timepoint is on the x-axis and the plot is colored by Diagnosis.

```
# code here
```