Data Wrangling I

Meghan Muse

July 11, 2024

Lesson Objectives

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

1. Generate random data
2. Derive new variables and subset data frames based on your existing data
3. Move between long and wide formatted data
4. Produce basic boxplots and scatter plots using ggpubr
5. Perform basic manipulation of character strings

Resources

Overview of ggpubr: https://rpkgs.datanovia.com/ggpubr/

Examples of reshape2 package: https://seananderson.ca/2013/10/19/reshape/

Wide vs long data frames: https://www.youtube.com/watch?v=pHPgMNXyzqc

Generating Random Data

For this lecture, we will start by generating a new data set. To generate random variables, we can use the *rnorm* and *rbinom* functions.

```
# Define a variable of random systolic blood pressure
rnorm(n = 10,mean = 128.4,sd = 19.6)
```

```
##  [1] 153.0027 122.0408 156.0632 100.7644 143.7820 112.4153 106.1646 151.1675
##  [9] 126.9746 102.2304
```

Did you get the same values as me? Why or why not?

```
set.seed(103)
rnorm(n = 10,mean = 128.4,sd = 19.6)
```

```
##  [1] 112.99493 129.47288 105.41782 125.12067  91.84538 126.03937 144.58918
##  [8] 151.72713 107.00754 121.04780
```

```
rnorm(n = 10,mean = 128.4,sd = 19.6)
```

```
##  [1] 110.1285 129.0511 130.7935 178.9627 123.1178 110.9893 137.4007 130.6614
##  [9] 123.8652 143.9297
```

```
set.seed(103)
rnorm(n = 10,mean = 128.4,sd = 19.6)
```

```
##  [1] 112.99493 129.47288 105.41782 125.12067  91.84538 126.03937 144.58918
##  [8] 151.72713 107.00754 121.04780
```

What do you think the set seed function does?

Many functions in R incorporate some level of randomziation such as sampling, some clustering algorithms, and some plotting functions. Always be aware of if you are using randomization, and always ensure that your code and results are reproducible by using the *set.seed()* function.

Now let's generate a random binary variable

```
set.seed(103)
rbinom(n = 10,size = 1,prob = 0.5)
```

```
##  [1] 0 0 1 1 0 0 0 0 0 0
```

```
rbinom(n = 10,size = 10,prob = 0.5)
```

```
##  [1] 5 4 6 6 7 3 3 4 4 4
```

What does the "size" parameter change? If we were creating a binary variable like biological sex, what would you set it as?

```
set.seed(103)
mean(rbinom(n = 100,size = 10,prob = 0.5))
```

```
## [1] 5.01
```

```
mean(rbinom(n = 100,size = 10,prob = 0.7))
```

```
## [1] 6.93
```

What does the "prob" parameter change? When might you use different values here?

Okay now lets build a toy data set

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

# Define a data frame with our randomly generated data
randomData <- data.frame('SubjectID' = seq(1:1000),
                         'systolicBP' = rnorm(n = 1000,mean = 128,sd = 20),
                         'diastolicBP' = 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))

# Take a peak at the top entries for our dataset
head(randomData)
```

```
##   SubjectID systolicBP diastolicBP Age Male
## 1         1  112.28054    75.52894  52    0
## 2         2  129.09478    59.95778  56    0
## 3         3  104.54879    74.51568  25    1
## 4         4  124.65374    52.99577  41    0
## 5         5   90.69937    71.17388  41    0
## 6         6  125.59120    69.50961  31    0
```

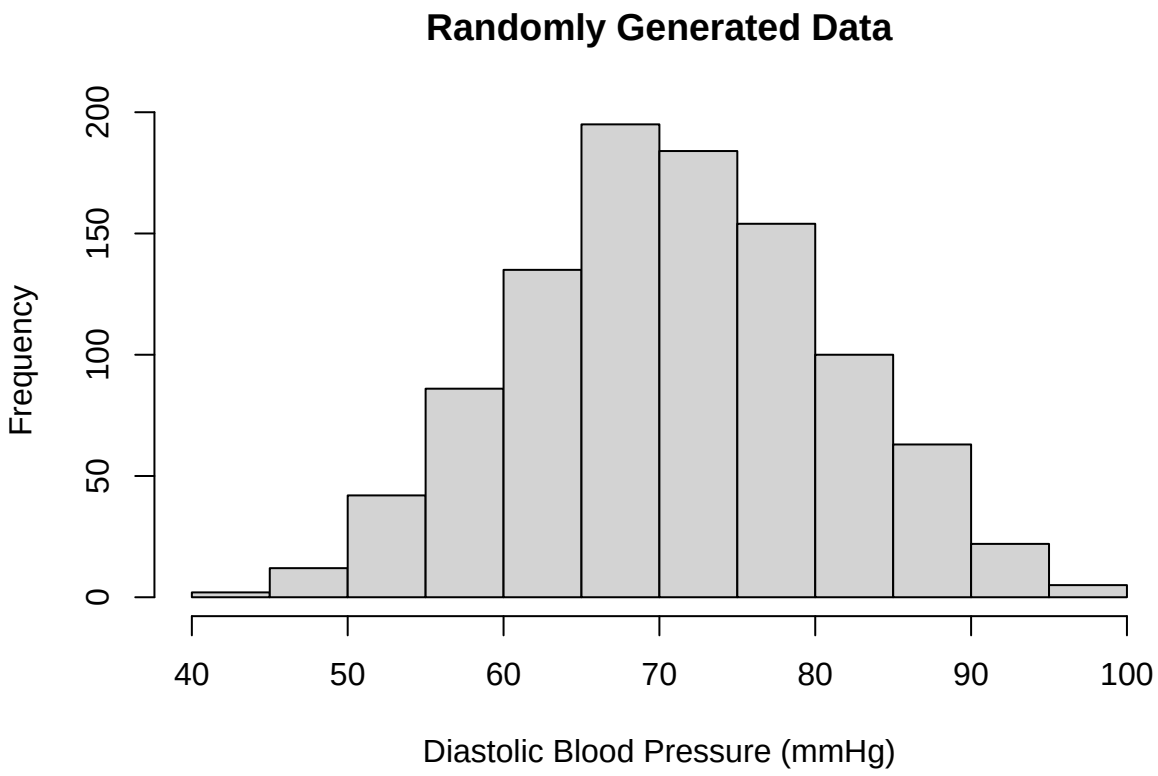
```r
# Assess the mean and standard deviation for systolic blood pressure
mean(randomData$systolicBP)
```

```
## [1] 128.4313
```

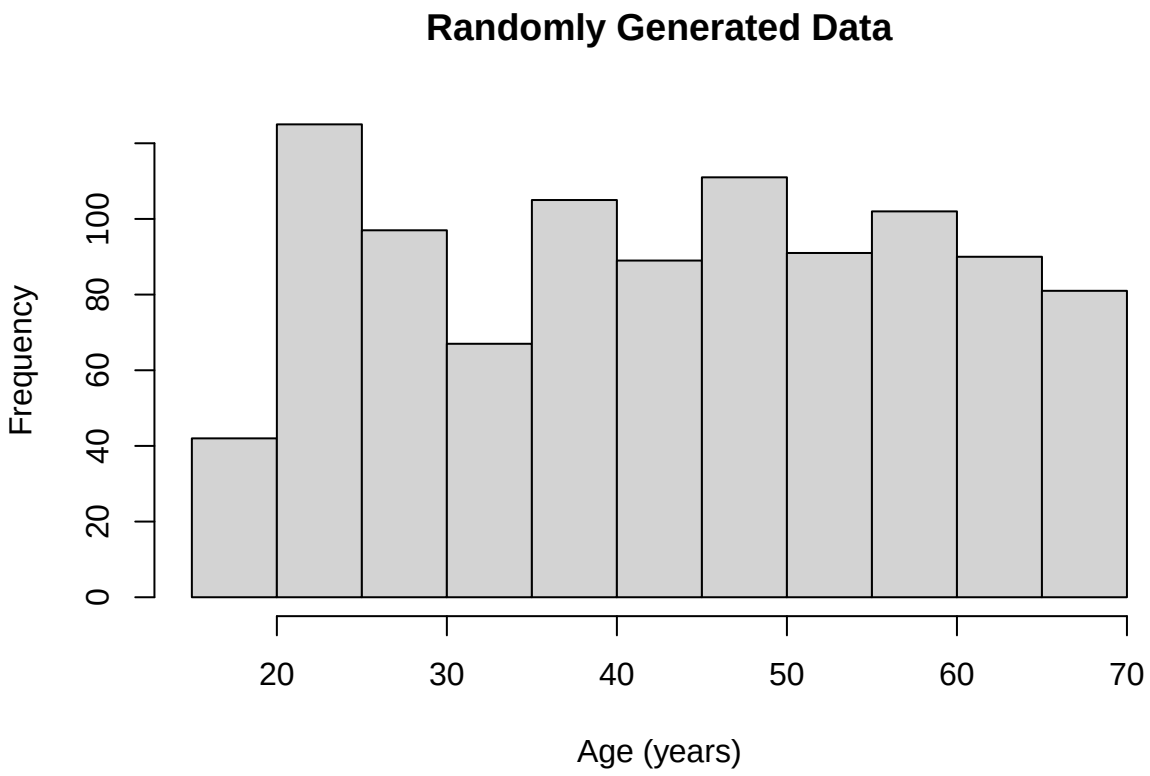
```r
sd(randomData$systolicBP)
```

```
## [1] 19.37998
```

```r
# Visualize the distribution of diastolic blood pressure
hist(randomData$diastolicBP,main = 'Randomly Generated Data',xlab = 'Diastolic Blood Pressure (mmHg)')
```




```
# Visualize the distribution of age
hist(randomData$Age,main = 'Randomly Generated Data',xlab = 'Age (years)')
```



What differences do you notice in the distribution of these two variables?

```
# Check the distribution of our variables
summary(randomData)
```

```
##    SubjectID        systolicBP      diastolicBP         Age
##  Min.   :   1.0   Min.   : 67.48   Min.   :43.07   Min.   :18.00
##  1st Qu.: 250.8   1st Qu.:115.12   1st Qu.:64.15   1st Qu.:29.75
##  Median : 500.5   Median :127.77   Median :70.67   Median :44.00
##  Mean   : 500.5   Mean   :128.43   Mean   :70.99   Mean   :43.54
##  3rd Qu.: 750.2   3rd Qu.:141.24   3rd Qu.:78.02   3rd Qu.:56.00
##  Max.   :1000.0   Max.   :207.49   Max.   :98.99   Max.   :69.00
##       Male
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :1.000
##  Mean   :0.525
##  3rd Qu.:1.000
##  Max.   :1.000
```

Is it summarizing the factor variable the way we want? Why or why not?

Deriving New Variables

Lets try converting the binary variable to a more logical format

```
# Define a new factor variable from an old binary
randomData$BiologicalSex <- factor(ifelse(randomData$Male == 1,'Male','Female'))

# Re-check the distribution of our variables
summary(randomData)
```

```
##    SubjectID        systolicBP      diastolicBP         Age
##  Min.   :   1.0   Min.   : 67.48   Min.   :43.07   Min.   :18.00
##  1st Qu.: 250.8   1st Qu.:115.12   1st Qu.:64.15   1st Qu.:29.75
##  Median : 500.5   Median :127.77   Median :70.67   Median :44.00
##  Mean   : 500.5   Mean   :128.43   Mean   :70.99   Mean   :43.54
##  3rd Qu.: 750.2   3rd Qu.:141.24   3rd Qu.:78.02   3rd Qu.:56.00
##  Max.   :1000.0   Max.   :207.49   Max.   :98.99   Max.   :69.00
##       Male       BiologicalSex
##  Min.   :0.000   Female:475
##  1st Qu.:0.000   Male  :525
##  Median :1.000
##  Mean   :0.525
##  3rd Qu.:1.000
##  Max.   :1.000
```

We can also generate a new binary variable from a continuous variable

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

# Re-check the distribution of our variables
summary(randomData)
```

```
##    SubjectID        systolicBP      diastolicBP         Age
##  Min.   :   1.0   Min.   : 67.48   Min.   :43.07   Min.   :18.00
##  1st Qu.: 250.8   1st Qu.:115.12   1st Qu.:64.15   1st Qu.:29.75
##  Median : 500.5   Median :127.77   Median :70.67   Median :44.00
##  Mean   : 500.5   Mean   :128.43   Mean   :70.99   Mean   :43.54
##  3rd Qu.: 750.2   3rd Qu.:141.24   3rd Qu.:78.02   3rd Qu.:56.00
##  Max.   :1000.0   Max.   :207.49   Max.   :98.99   Max.   :69.00
##       Male       BiologicalSex MedicareAge
##  Min.   :0.000   Female:475    Mode :logical
##  1st Qu.:0.000   Male  :525    FALSE:903
##  Median :1.000                 TRUE :97
##  Mean   :0.525
##  3rd Qu.:1.000
##  Max.   :1.000
```

Subsetting Data

As we learned in a previous lecture, we can subset a data frame in a few ways

```r
# Subset to only those at medicare age
medicareData1 <- randomData[which(randomData$MedicareAge == T),]
medicareData2 <- randomData[which(randomData$Age >= 65),]

# We can use the following statement to ensure both methods produced the same result
all(medicareData1 == medicareData2)
```

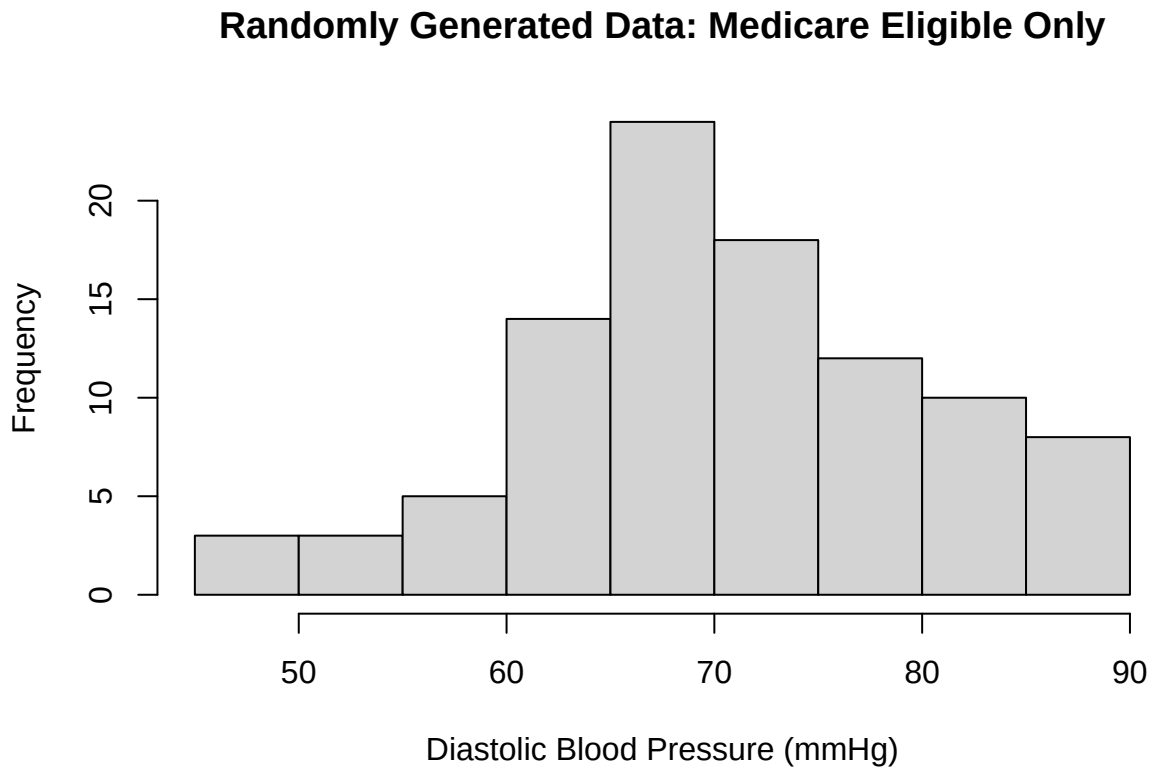
```
## [1] TRUE
```

```r
table(medicareData1 == medicareData2)
```

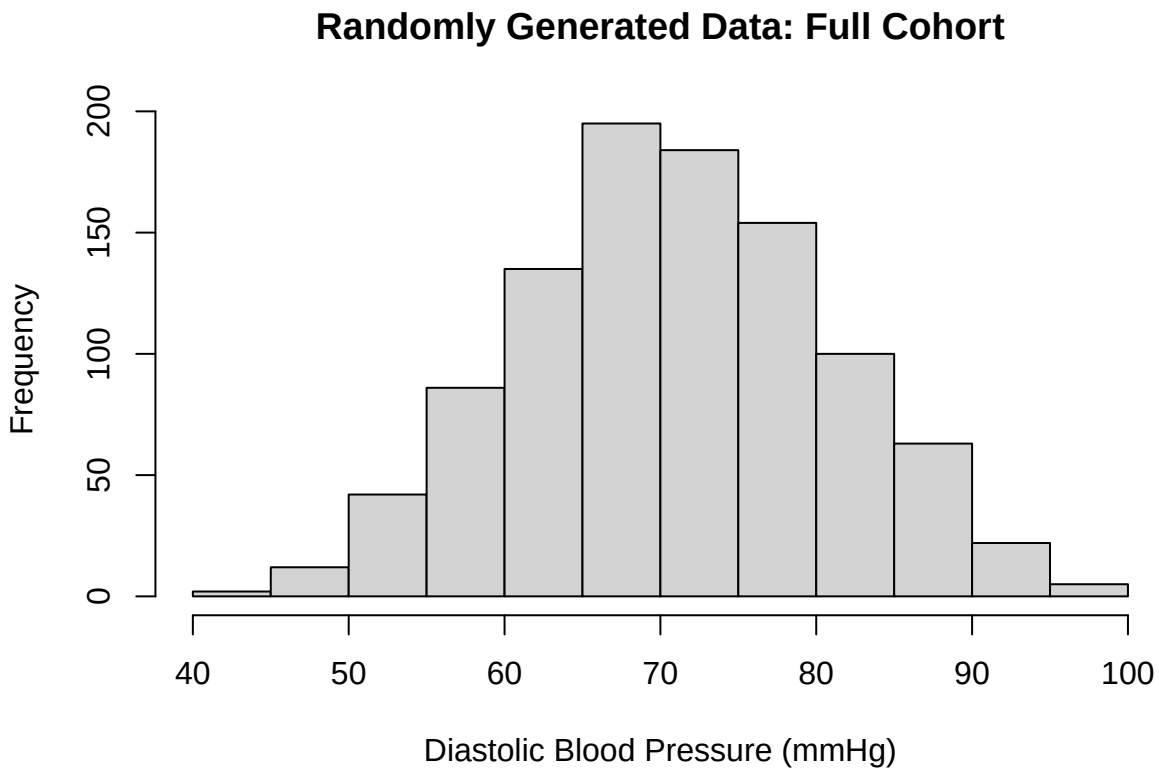
```
##
## TRUE
##  679
```

Now we can look at the distribution in diastolic blood pressure just among individuals meeting a given age cutoff

```r
# Visualize the distribution of diastolic blood pressure in Medicare Only
hist(medicareData1$diastolicBP,main = 'Randomly Generated Data: Medicare Eligible Only',xlab = 'Diastoli
```



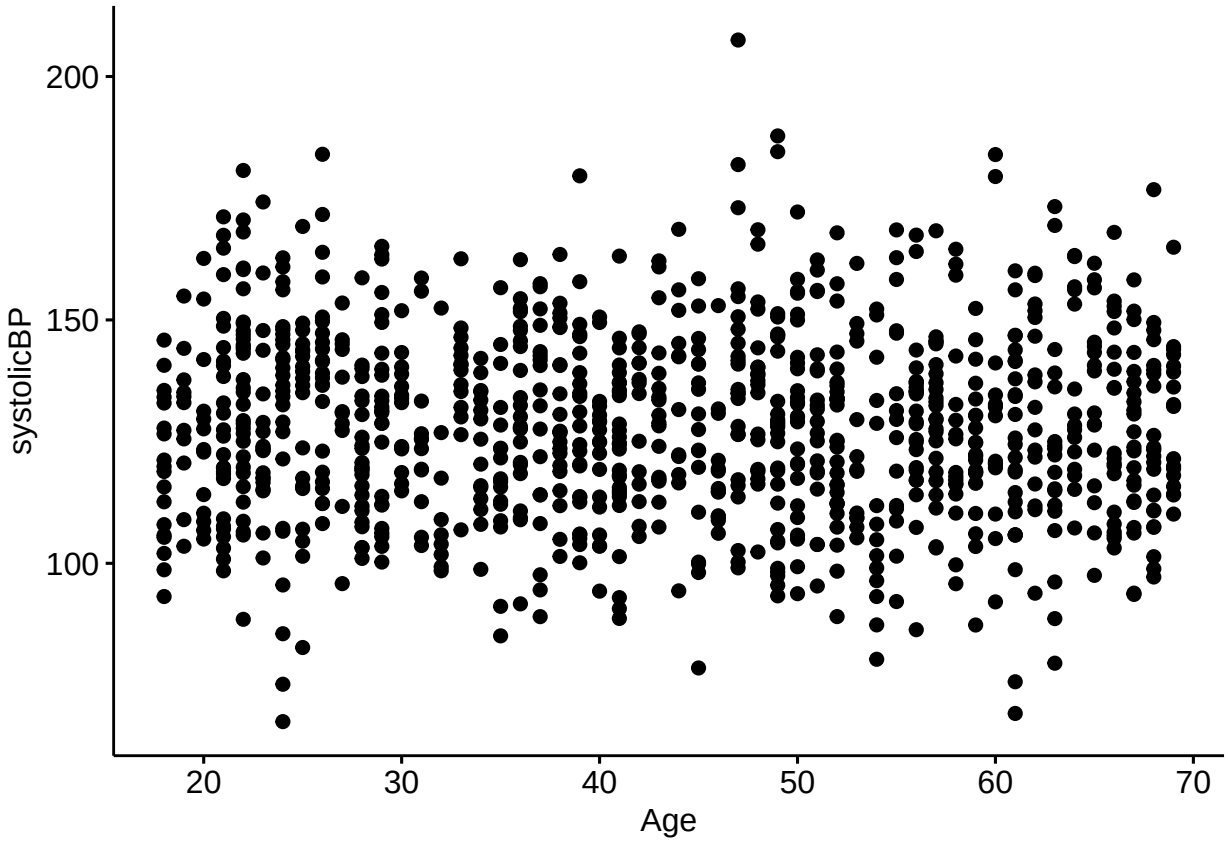
```r
# Visualize the distribution of diastolic blood pressure in full dataset
hist(randomData$diastolicBP,main = 'Randomly Generated Data: Full Cohort',xlab = 'Diastolic Blood Pressu
```



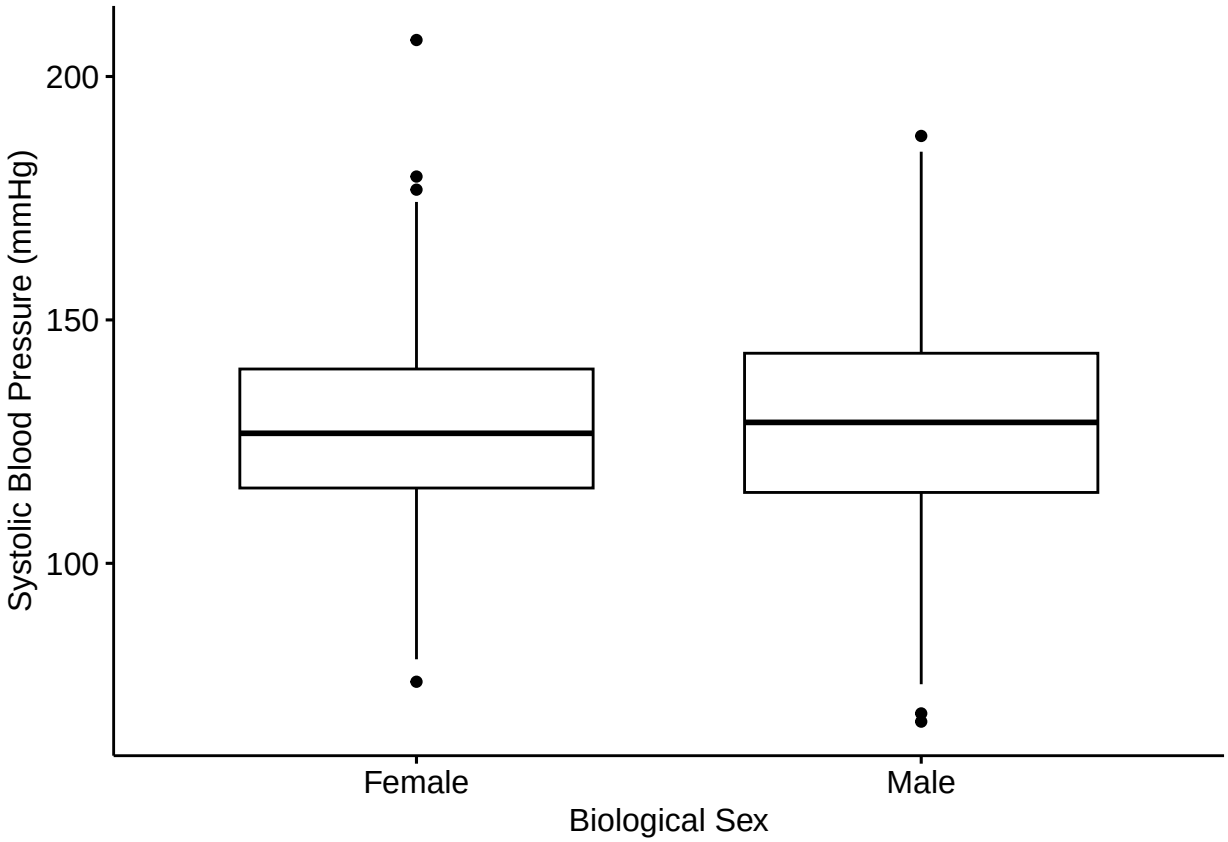
Does the distribution in the subsetted data look any different than in the full data set? Should it?

Basic Plotting Using *ggpubr*

```r
# Run the following line of code one time to install the package if you haven't already
# install.packages('ggpubr')

# Generate a scatter plot of age by systolic blood pressure
ggpubr::ggscatter(randomData,x = 'Age',y = 'systolicBP')
```




```
# Generate a boxplot for diastolic blood pressure distrubition by biological sex in our original datase
ggpubr::ggboxplot(randomData,x = 'BiologicalSex',y = 'systolicBP',xlab = 'Biological Sex',
                  ylab = 'Systolic Blood Pressure (mmHg)')
```



Was there any visual association between age or sex and blood pressure?

Would we expect there to be based on how we generated the data?

What if we want to look at systolic and diastolic blood pressure in the same plot? To do this, we need to convert out data to a long format.

Wide vs Long Dataframes

Wide format data will have each subject in a single row and then multiple measures related to that subject in a unique column.

```
# Wide format data
data.frame('PatientID' = c('P001','P002','P003'),
                          'Age' = c(24,35,27),
                          'Height.in' = c(67,70,64))
```

```
##   PatientID Age Height.in
## 1      P001  24        67
## 2      P002  35        70
## 3      P003  27        64
```

Long format data will have each measurement in a new row.

```r
# Wide format data
data.frame('PatientID' = c('P001','P001','P002','P002','P003','P003'),
           'Measure' = c('Age','Height.in','Age','Height.in','Age','Height.in'),
           'Value' = c(24,67,35,70,27,64))
```

```
##   PatientID   Measure Value
## 1      P001       Age    24
## 2      P001 Height.in    67
## 3      P002       Age    35
## 4      P002 Height.in    70
## 5      P003       Age    27
## 6      P003 Height.in    64
```

While this might seem odd for this type of use case, it is a very useful tool for datasets with **repeated measures**, i.e. measurements taken over time like this:

```r
# Wide format repeated measures
data.frame('PatientID' = c('P001','P002','P003'),
           'Height.in.5y' = c(40,43,39),
           'Height.in.10y' = c(53,58,51))
```

```
##   PatientID Height.in.5y Height.in.10y
## 1      P001           40            53
## 2      P002           43            58
## 3      P003           39            51
```

```r
# Long format repeated measures
data.frame('PatientID' = c('P001','P001','P002','P002','P003','P003'),
           'Age.yrs' = c(5,10,5,10,5,10),
           'Height.in' = c(40,53,43,58,39,51))
```

```
##   PatientID Age.yrs Height.in
## 1      P001       5        40
## 2      P001      10        53
## 3      P002       5        43
## 4      P002      10        58
## 5      P003       5        39
## 6      P003      10        51
```

Long Format Data

Currently, our data is in a **wide** format because each subject is only in a single row and all measures from that subject are in columns. There are a few different ways to convert between wide an long format data.

To achieve this today, we will use the *melt* function from the package *reshape2*. Next class, we will discuss how to achieve this in *tidyverse*.

```r
# Run the following line of code one time to install the package if you haven't already
# install.packages('reshape2')

# Melt the data frame into a long form
```

```
longData <- reshape2::melt(randomData[,c('SubjectID','systolicBP','diastolicBP','Age','BiologicalSex')]
                           id.vars = c('SubjectID','Age','BiologicalSex'),value.name = 'BP',
                           variable.name = 'BP.Type')
# Check enteries for our first subject
longData[longData$SubjectID == 1,]
```

```
##      SubjectID Age BiologicalSex     BP.Type        BP
## 1            1  52        Female  systolicBP 112.28054
## 1001         1  52        Female diastolicBP  75.52894
```

```
# How it would look if we forgot one of our ID variables
longData2 <- reshape2::melt(randomData[,c('SubjectID','systolicBP','diastolicBP','Age','BiologicalSex')]
                           id.vars = c('SubjectID','Age'),value.name = 'BP',
                           variable.name = 'BP.Type')
```

```
## Warning: attributes are not identical across measure variables; they will be
## dropped
```

```
typeof(longData2$BP)
```

```
## [1] "character"
```

```
longData2[longData2$SubjectID == 1,]
```

```
##      SubjectID Age       BP.Type                BP
## 1            1  52    systolicBP 112.280536472035
## 1001         1  52   diastolicBP 75.5289415224527
## 2001         1  52 BiologicalSex            Female
```

We can also use the *order* function to ensure that all our subjects values are listed in sequence in the table or to look at subjects who have the oldest age easily

```
# Print first 20 values
longData$Age[1:20]
```

```
##  [1] 52 56 25 41 41 31 69 67 46 60 36 56 40 39 38 52 52 60 20 61
```

```
order(longData$Age)[1:20]
```

```
##  [1]   35  121  158  214  237  337  342  350  594  616  653  656  717  736  739
## [16]  846  976  991 1035 1121
```

What does the *order* function return?

```
longData$Age[14]
```

```
## [1] 39
```

```r
longData[order(longData$Age),'Age'][1:20]
```

```
##  [1] 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18
```

```r
# Order by subject ID
head(longData[order(longData$SubjectID,decreasing = F),])
```

```
##      SubjectID Age BiologicalSex     BP.Type        BP
## 1            1  52        Female  systolicBP 112.28054
## 1001         1  52        Female diastolicBP  75.52894
## 2            2  56        Female  systolicBP 129.09478
## 1002         2  56        Female diastolicBP  59.95778
## 3            3  25          Male  systolicBP 104.54879
## 1003         3  25          Male diastolicBP  74.51568
```

```r
# Order by decreasing age
head(longData[order(longData$Age,decreasing = T),])
```

```
##     SubjectID Age BiologicalSex    BP.Type       BP
## 7           7  69        Female systolicBP 144.5196
## 114       114  69          Male systolicBP 119.9431
## 124       124  69        Female systolicBP 140.5145
## 336       336  69          Male systolicBP 115.8081
## 363       363  69        Female systolicBP 139.2851
## 391       391  69          Male systolicBP 119.4529
```

The order function can be particularly helpful when trying to merge two data frames and want to first ensure that subjects or samples are in the same order in each data frame.

Manipulating Character Strings

If we don't like the end of systolic and diastolic labels having "BP" we can remove it using the *sub* function.

```r
# Look for cases of "BP" and replace with ""
longData$BP.Type <- sub(pattern = 'BP',replacement = '',x = longData$BP.Type)

# Now lets see what values we have for this variable
table(longData$BP.Type)
```

```
##
## diastolic  systolic
##      1000      1000
```

```r
# We could also just replace the values using indexing as follows
longData[which(longData$BP.Type == 'systolic'),'BP.Type'] <- 'Systolic'
longData[which(longData$BP.Type == 'diastolic'),'BP.Type'] <- 'Diastolic'

# Now lets check again what values we have for this variable
table(longData$BP.Type)
```

```
##
## Diastolic  Systolic
##      1000      1000
```

sub and *gsub* can both be used to search and replace in character strings. Can you identify the difference between what each one does?

```
# Define a list of character strings
charStrings <- c('QBS Graduate Program at Dartmouth College','QBS 103','QBS! QBS! QBS!')

# sub function
sub(pattern = 'QBS',replacement = 'Dartmouth',x = charStrings)
```

```
## [1] "Dartmouth Graduate Program at Dartmouth College"
## [2] "Dartmouth 103"
## [3] "Dartmouth! QBS! QBS!"
```

```
# gsub function
gsub(pattern = 'QBS',replacement = 'Dartmouth',x = charStrings)
```

```
## [1] "Dartmouth Graduate Program at Dartmouth College"
## [2] "Dartmouth 103"
## [3] "Dartmouth! Dartmouth! Dartmouth!"
```

This may seem easy enough to do on your own, but sometimes you will have large lists of character strings between two dataframes that have been entered in different formats.

Here, I have 2 lists of sample IDs that I need to make sure are in the same order in both lists but you can see, they are formatted differently across both lists.

```
# Define two lists of sample names
d1 <- c('20210323_PB4_01_09.RCC','20210323_PB4_01_11.RCC','20210323_PB4_01_12.RCC','20210401_ch5-040121
d2 <- c('20210323-PB4_01_11.RCC','20210323-PB4_01_12.RCC','20210323-PB4_01_09.RCC','20210401-ch5-040121

# Check if same samples are in both lists
table(d1 %in% d2)
```

```
##
## FALSE
##     7
```

```
# Compare formatting
head(d1)
```

```
## [1] "20210323_PB4_01_09.RCC"         "20210323_PB4_01_11.RCC"
## [3] "20210323_PB4_01_12.RCC"         "20210401_ch5-040121_01_01.RCC"
## [5] "20210401_ch5-040121_01_02.RCC" "20210401_ch5-040121_01_02.RCC"
```

```r
head(d2)
```

```
## [1] "20210323-PB4_01_11.RCC"         "20210323-PB4_01_12.RCC"
## [3] "20210323-PB4_01_09.RCC"         "20210401-ch5-040121-01-01.RCC"
## [5] "20210401-ch5-040121-01-02.RCC" "20210401-ch5-040121-01-02.RCC"
```

We can change the formatting using *gsub*

```r
# Replace all "_" with "-" in list of sample names
d1 <- gsub(d1,pattern = '_',replacement = '-')
d2 <- gsub(d2,pattern = '_',replacement = '-')

# Verify all names in d1 are now also in d2
table(d1 %in% d2)
```

```
##
## TRUE
##    7
```

```r
# Check if D1 and D2 are in the same order
table(d1 == d2)
```

```
##
## FALSE  TRUE
##     3     4
```

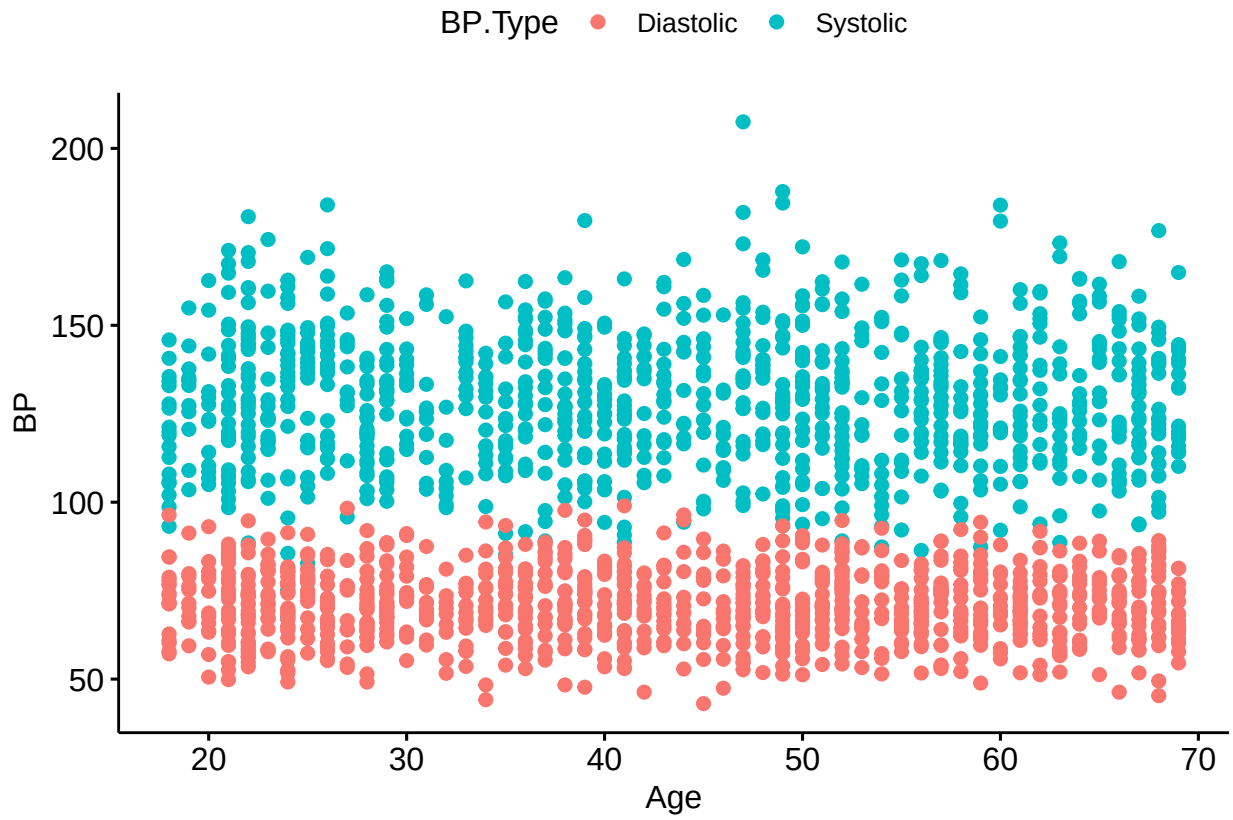
```r
# Reorder both vectors
d1 <- d1[order(d1)]
d2 <- d2[order(d2)]

# Check if D1 and D2 are in the same order
table(d1 == d2)
```

```
##
## TRUE
##    7
```

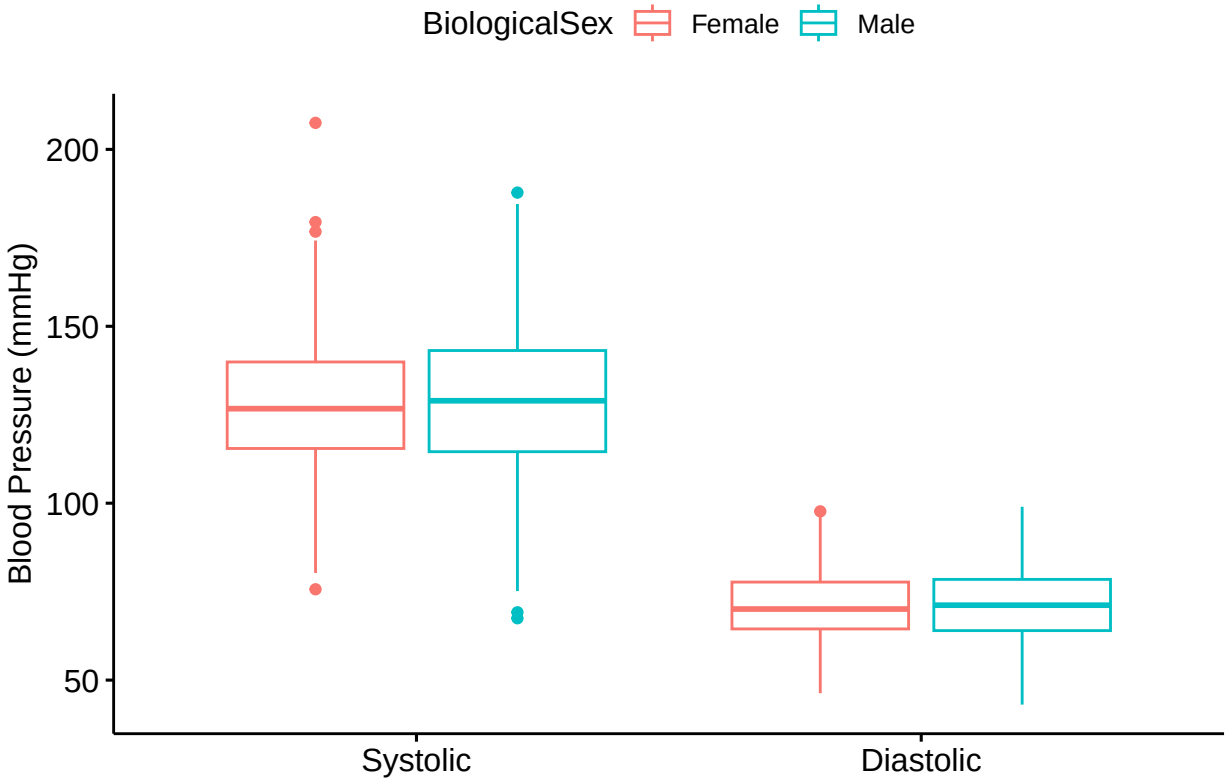
Adding Color to Our *ggpubr* Plots

Now we can go back and generate out 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')
```




```
# Generate a boxplot for diastolic blood pressure distrubition by biological sex in our original datase
ggpubr::ggboxplot(longData,x = 'BP.Type',y = 'BP',color = 'BiologicalSex',
                  ylab = 'Blood Pressure (mmHg)',xlab = '')
```



Converting Back to Wide Format Data

We can also change our data back into a **wide** format using the *cast* function as follows:

```
# Cast into a data frame
wideData <- reshape2::dcast(longData,formula =  SubjectID + Age + BiologicalSex ~ BP.Type,value.var = "
# Note: using dcast because we want a dataframe as output.
# If we want a vector or matrix as output, we use acast

# Lets see if our data look the same as when we started
head(wideData)
```

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

```
# Our original data set
head(randomData)
```

```
##   SubjectID systolicBP diastolicBP 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
```

We can also use the *dcast* function to make summary values. For example, what if we wanted to known the average BP for males and females based on their age group.

```
# Generate table (note here we drop subject ID from the formula)
summaryTable <- reshape2::dcast(longData,formula = Age + BiologicalSex ~ BP.Type,
                                fun.aggregate = mean,value.var = 'BP')

# Check the dimmensions
dim(summaryTable)
```

```
## [1] 104   4
```

```
# Look at the top enteries
head(summaryTable)
```

```
##   Age BiologicalSex Diastolic Systolic
## 1  18        Female  76.03626 119.5091
## 2  18          Male  68.47743 118.7080
## 3  19        Female  74.30795 135.8965
## 4  19          Male  71.88793 124.2671
## 5  20        Female  71.04246 123.8012
## 6  20          Male  71.27133 127.4093
```

We can also use this to tabulate how many observations there are for each category

```
# Generate table (note here we drop subject ID from the formula)
summaryTable <- reshape2::dcast(longData,formula = Age + BiologicalSex ~ BP.Type,
                                fun.aggregate = length,value.var = 'BP')

# Check the dimmensions
dim(summaryTable)
```

```
## [1] 104   4
```

```
# Look at the top enteries
head(summaryTable)
```

```
##   Age BiologicalSex Diastolic Systolic
## 1  18        Female        10       10
## 2  18          Male         8        8
## 3  19        Female         5        5
## 4  19          Male         6        6
## 5  20        Female         5        5
## 6  20          Male         8        8
```

In Class Activity

Working in groups, define a new variable for hypertension in our original dataset (randomData). Here we will define hypertension as systolic blood pressure over 130 or a diastolic blood pressure over 80. Plot the distrubution in age for individuals with and without hypertension using boxplots.

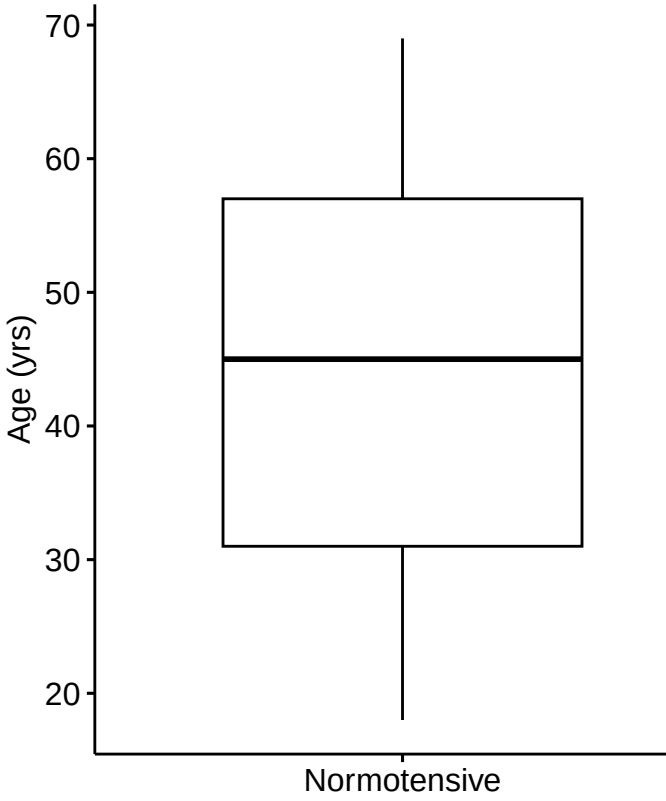
Use the *melt* function to generate boxplots of the distribution of systolic and diastolic blood pressure in hypertensive vs. normotensive individuals (color should be based on hypertension status).

Use the *dcast* function to generate a table summarizing the mean age, systolic, and diastolic BP for males and females, seperately, with and without hypertension. Your table should have 4 rows. Order your table output such that it lists values for normotensive individuals first and hypertensive individuals second.

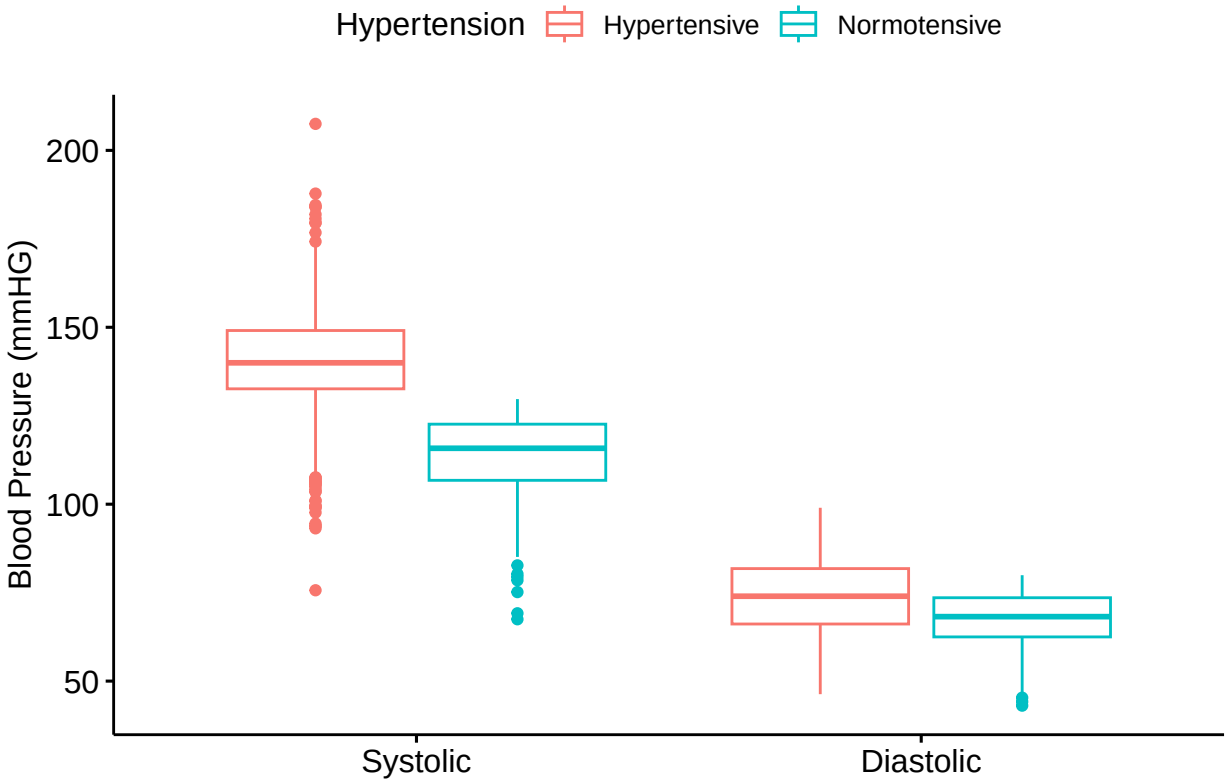
Sample Solution

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

# Define a data frame with our randomly generated data
randomData <- data.frame('SubjectID' = seq(1:1000),
                         'systolicBP' = rnorm(n = 1000,mean = 128,sd = 20),
                         'diastolicBP' = 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 a new factor variable from an old binary
randomData$BiologicalSex <- factor(ifelse(randomData$Male == 1,'Male','Female'))

# Define variable for hypertension
randomData$Hypertension <- ifelse(randomData$systolicBP > 130 | randomData$diastolicBP > 80,
                                  'Hypertensive','Normotensive')
# Plot distribution of age based on hypertension status
ggpubr::ggboxplot(randomData,x = 'Hypertension',y = 'Age',xlab = '',ylab = 'Age (yrs)')
```

Remember: There are many ways to solve any problem in R. As long as you get the same end product, what you did is likely just as valid as what I did.




```
# Generate long format data
longData <- reshape2::melt(randomData,id.vars = c('SubjectID','Age','Male','BiologicalSex','Hypertensio
                           value.name = 'BP',variable.name = 'BP.Type')
# Reformat names for bp type
longData$BP.Measure <- ifelse(longData$BP.Type == 'systolicBP','Systolic','Diastolic')

# Plot BP by hypertensive classification
ggpubr::ggboxplot(longData,x = 'BP.Measure',y = 'BP',color = 'Hypertension',
                  xlab = '',ylab = 'Blood Pressure (mmHG)')
```




```
# Regenerate long data to also include age
longData2 <- reshape2::melt(randomData,id.vars = c('SubjectID','Male','BiologicalSex','Hypertension'),
                            value.name = 'Cont.Value',variable.name = 'Var.Name')
# Generate summary table
sumTab <- reshape2::dcast(longData2,formula = BiologicalSex + Hypertension ~ Var.Name,
                          fun.aggregate = mean,value.var = 'Cont.Value')
sumTab[order(sumTab$Hypertension,decreasing = T),]
```

```
##   BiologicalSex Hypertension systolicBP diastolicBP      Age
## 2        Female Normotensive   114.7668    67.23993 43.64706
## 4          Male Normotensive   112.6688    67.64147 44.62162
## 1        Female Hypertensive   138.8948    73.64332 43.98425
## 3          Male Hypertensive   141.1750    73.96578 42.28713
```