#< ignore Save the file as `week_one_sol.Rmd`. Run setup chunk in RStudio.

#>

Exercise Overview

Week 1 - Data Analysis & Visualization in R

PBHLTH W142: UC Berkeley School of Public Health

Author: Xander Posner, Graduate Student Instructor

xander@berkeley.edu

Access this week's bCourses site <a href="https://bcourses.berkeley.edu/courses/1492482/pages/week-1", target = _blank>here

Check out the source code for this project on <a href="https://github.com/posnerab/RTutorial-Week1", target = _blank>Github

Tools Used in this Tutorial

In this tutorial, we will:

Get to know the Western Collaborative Group Study (wcgs) dataset from the epitools package

Change some variable types (e.g., numeric to factor)

Create a new BMI variable based on existing height and weight measurements

Generate summary statistics and figures

Interpret your summary statistics and figures

R Packages Used in this Tutorial

"tidyverse"

"epitools"

When practicing on your own in RStudio, you will first need to install each of these packages using the install.packages("") function, putting the name(s) of the package(s) inside the quotation marks like this:

install.packages("tidyverse") OR install.packages(c("tidyverse", "epitools"))

You won't need to install packages more than once on the same computer.

After installing, you must load each package library separately, using the library() function, putting the name of each package inside the parentheses without quotation marks like this:

library(tidyverse)

library(epitools)

You will need to load packages each time you need to use them, using the library() function whenever you initiate a new R session.

You don't need to install any packages for this tutorial. Just press check to run any chunk with a library() function in it.

R Functions Used in this Tutorial

library() to load each package library

data() to read built-in datasets into R

head() to view the first few lines of your dataset

nrow() to view the number of rows in your dataset

str() to view the "structure" of your dataset, including variable names and types

summary() to produce summary statistics

ggplot(), geom_histogram(), geom_boxplot(), and geom_col() to create histograms, boxplots, and bar charts

Click the "Go to next exercise..." button to continue.

Exercise 1 – Get Started

Some code chunks are finished for you; just press check to proceed. Other chunks require you to modify and/or add code before pressing check to move through the tutorial.

If you get stuck, use the hint in each code chunk first. If you're still having trouble, head to Piazza and post your question.

Make sure you run each chunk in order!

Get to Know Your Data

a) The wcgs dataset is built into the epitools package. As long as you load the libraries first using the library() functions, you will be able to load the dataset by calling data(wcgs) as shown below. Click "check" to continue.

```
#< task
library(epitools)
library(tidyverse) # helps us make charts and graphics
```

```
## -- Attaching packages --------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(readr)
library(markdown)
library(DescTools)

data(wcgs)
#>
```

b) Now that the wcgs dataset is loaded into your environment, use the head() function to view the first six rows.

```
#< task
head(wcgs) # view the first six rows of the dataset in table format
```

```
##     id age0 height0 weight0 sbp0 dbp0 chol0 behpat0 ncigs0 dibpat0 chd69
## 1 2001   49      73     150  110   76   225       2     25       1     0
## 2 2002   42      70     160  154   84   177       2     20       1     0
## 3 2003   42      69     160  110   78   181       3      0       0     0
## 4 2004   41      68     152  124   78   132       4     20       0     0
## 5 2005   59      70     150  144   86   255       3     20       0     1
## 6 2006   44      72     204  150   90   182       4      0       0     0
##   typechd time169 arcus0
## 1       0    1664      0
## 2       0    3071      1
## 3       0    3071      0
## 4       0    3064      0
## 5       1    1885      1
## 6       0    3102      0
```

```
#>
```

c) Use the nrow() function to calculate the number of rows in your dataset, which here amounts to our sample size, n.

```
#< task
n <- nrow(wcgs)
#>
```

Notice that when we assigned the nrow() function to the object named n, R doesn't immediately print the result. Instead the resulting value is saved in your Global Environment.

d) Here are the ways you can have the results of a function print to your Console. Press check to observe.

```
#< task
nrow(wcgs) # PRINTS to the console but DOES NOT save the value
```

```
## [1] 3154
```

```
n <- nrow(wcgs) # SAVES the value but DOES NOT print

n # PRINTS out the results that you previously saved above
```

```
## [1] 3154
```

```
n + 1 # adds 1 to the value saved in the n object: example of what you can do with stored objects in yo
```

```
## [1] 3155
```

```
print(n <- nrow(wcgs)) # SAVES the value AND also PRINTS to the console in one step
```

```
## [1] 3154
```

```
#>
```

NOTE. R is cAsE sEnSiTiVe.N would be an entirely separate object from n from R's perspective.

Summarizing your data

R has many ways to view summary statistics. The simplest is the summary() function.

e) Run the chunk below to see what summary statistics the summary() function produces.

```
#< task
summary(wcgs)
```

```
##        id              age0          height0         weight0         sbp0
##  Min.   : 2001   Min.   :39.00   Min.   :60.00   Min.   : 78   Min.   : 98.0
##  1st Qu.: 3741   1st Qu.:42.00   1st Qu.:68.00   1st Qu.:155   1st Qu.:120.0
##  Median :11406   Median :45.00   Median :70.00   Median :170   Median :126.0
##  Mean   :10478   Mean   :46.28   Mean   :69.78   Mean   :170   Mean   :128.6
##  3rd Qu.:13115   3rd Qu.:50.00   3rd Qu.:72.00   3rd Qu.:182   3rd Qu.:136.0
##  Max.   :22101   Max.   :59.00   Max.   :78.00   Max.   :320   Max.   :230.0
##
##       dbp0            chol0          behpat0          ncigs0
##  Min.   : 58.00   Min.   :103.0   Min.   :1.000   Min.   : 0.0
##  1st Qu.: 76.00   1st Qu.:197.2   1st Qu.:2.000   1st Qu.: 0.0
##  Median : 80.00   Median :223.0   Median :2.000   Median : 0.0
##  Mean   : 82.02   Mean   :226.4   Mean   :2.523   Mean   :11.6
##  3rd Qu.: 86.00   3rd Qu.:253.0   3rd Qu.:3.000   3rd Qu.:20.0
##  Max.   :150.00   Max.   :645.0   Max.   :4.000   Max.   :99.0
##                   NA's   :12
##     dibpat0           chd69            typechd          time169
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.0000   Min.   :  18
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:2842
##  Median :1.0000   Median :0.00000   Median :0.0000   Median :2942
##  Mean   :0.5038   Mean   :0.08148   Mean   :0.1363   Mean   :2684
##  3rd Qu.:1.0000   3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:3037
##  Max.   :1.0000   Max.   :1.00000   Max.   :3.0000   Max.   :3430
##
##      arcus0
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.2985
##  3rd Qu.:1.0000
##  Max.   :1.0000
##  NA's   :2
```

```
#>
```

You will notice the summary() function prints out the following values for each variable in the wcgs dataset:

The minimum value

The value at Q1, the first quartile (25th percentile)

The median value or Q2 (50th percentile)

The mean (average) value

The value at Q3, the third quartile (75th percentile)

The maximum value

f) Use the dollar sign $ to access different variables in your dataset. The first one is done for you. Fill in the code below to summarize the height0 variable.

```
#< task
summary_age0 <- summary(wcgs$age0)
summary_age0
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   39.00   42.00   45.00   46.28   50.00   59.00
```

```
#>

#< task_notest
summary_height0 <- "<<<<<<YOUR CODE HERE>>>>>>"
summary_height0
```

```
## [1] "<<<<<<YOUR CODE HERE>>>>>>"
```

```
#>

summary_height0 <- summary(wcgs$height0)
summary_height0
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   60.00   68.00   70.00   69.78   72.00   78.00
```

#< award "Summary" You summarized it! Click 'Go to next exercise...' to continue. #>

Exercise 2 – Modify Your Dataset

Make Factor Variables

a) First, view the structure of your dataset using the str() function. Click Check to observe. Notice the different variable "types."

```
#< task
str(wcgs)
```

```
## 'data.frame':    3154 obs. of  14 variables:
##  $ id     : int  2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 ...
##  $ age0   : int  49 42 42 41 59 44 44 40 43 42 ...
##  $ height0: int  73 70 69 68 70 72 72 71 72 70 ...
##  $ weight0: int  150 160 160 152 150 204 164 150 190 175 ...
##  $ sbp0   : int  110 154 110 124 144 150 130 138 146 132 ...
##  $ dbp0   : int  76 84 78 78 86 90 84 60 76 90 ...
```

```
##  $ chol0  : int  225 177 181 132 255 182 155 140 149 325 ...
##  $ behpat0: int  2 2 3 4 3 4 4 2 3 2 ...
##  $ ncigs0 : int  25 20 0 20 20 0 0 0 25 0 ...
##  $ dibpat0: int  1 1 0 0 0 0 0 1 0 1 ...
##  $ chd69  : int  0 0 0 0 1 0 0 0 0 0 ...
##  $ typechd: int  0 0 0 0 1 0 0 0 0 0 ...
##  $ time169: int  1664 3071 3071 3064 1885 3102 3074 3071 3064 1032 ...
##  $ arcus0 : int  0 1 0 0 1 0 0 0 0 1 ...
```

```
#>
```

b) Notice that the dichotomous behavioral pattern variable, dibpat0, is currently coded as an integer (1 = Type A, 0 = Type B). We want R to know that behpat0 is a factor variable with 2 discrete levels: A and B. To do this, we will use the factor() function and create a new factor variable called dibpat0_fact.

```
#< task
wcgs$dibpat0_fact <- factor(wcgs$dibpat0, ordered = TRUE, labels = c("A","B"))
#>
```

c) Now create a new variable called smoker0 that is equal to 1 if someone currently smokes any cigarettes and 0 if they don't

```
#< task
wcgs$smoker0[wcgs$ncigs0 > 0] <- 1 # creates a new variable called smoker0 and sets the value to 1 for

wcgs$smoker0[wcgs$ncigs0 == 0] <- 0 # sets the value of smoker0 to 0 for all individuals with ncigs = 0
#>
```

The square brackets indicate an R operation called "subsetting." This operation tells R to set the value of smoker0 to 0 if the ncigs0 value for that individual is 0.

Note that we had to use two equal signs (==) when we specified the second condition for subsetting.

Options for Subsetting Conditions

dataset$variable == X < /code >< /li > (equaltosomenumberX) < li >< code > dataset$variable < Y

(less than some number Y)

dataset$variable > Z < /code >< /li > (greaterthansomenumberZ) < li >< code > dataset$variable <= A

(less than or equal to some number A)

dataset$variable >= B

(greater than or equal to some number B)

d) Now try making a "yes/no" factor variable for systolic blood pressure, sbp0, just like we did for smoking above. Call the new variable highsbp0 and set it equal to 1 if sbp0 is greater than or equal to 140, and set it equal to 0 if sbp0 is less than 140.

```
#< task_notest
"<<<<<<YOUR CODE HERE>>>>>>" <- 1

"<<<<<<YOUR CODE HERE>>>>>>" <- 0
```

```
#>

wcgs$highsbp0[wcgs$sbp0 >= 140] <- 1

wcgs$highsbp0[wcgs$sbp0 < 140] <- 0

# now make highsbp0 a factor variable with 2 levels

wcgs$highsbp0 <- factor(wcgs$highsbp0, labels = c("normal sbp", "high sbp"))
```

Create a new BMI variable

The wcgs dataset has the height variable, height0, in inches (in) and the weight variable, weight0, in pounds (lbs). We want to analyze BMI, which is equal to weight (mass) in kilograms divided by height in meters squared, so we'll have to convert height to centimeters (cm) and weight to kilograms (kg).

e) Create new variables for height, weight, and BMI in the wcgs dataset by using the assignment operator (<-) like so:

```
#< task
wcgs$heightcm0 <- round(wcgs$height0 * 2.54, digits = 2)
# this line of code tells R to take the Height0 value for each individual (row) in the wcgs dataset, mu

wcgs$weightkg0 <- round(wcgs$weight0/2.2, digits = 2)

wcgs$BMI0 <- round(wcgs$weightkg0/((wcgs$heightcm0/100)^2), digits = 1)
#>
```

f) This chunk writes the wcgsdata frame, including the new variables we created, to a comma separated value (.csv) file named "wcgs_data_new.csv". The csv file is then loaded into R and assigned to the wcgs_new object so it can be called later on.

```
#< task
wcgs_new <- write_csv(wcgs, "wcgs_data_new.csv")
#>
```

g) Now, use the head() function again to view the first six lines of the wcgs_new dataset. You will notice that there are now five new columns with the variable names we assigned and the values we calculated.

```
#< task_notest
head(wcgs_new)
```

```
##     id age0 height0 weight0 sbp0 dbp0 chol0 behpat0 ncigs0 dibpat0 chd69
## 1 2001   49      73     150  110   76   225       2     25       1     0
## 2 2002   42      70     160  154   84   177       2     20       1     0
## 3 2003   42      69     160  110   78   181       3      0       0     0
## 4 2004   41      68     152  124   78   132       4     20       0     0
## 5 2005   59      70     150  144   86   255       3     20       0     1
## 6 2006   44      72     204  150   90   182       4      0       0     0
##   typechd time169 arcus0 dibpat0_fact smoker0   highsbp0 heightcm0 weightkg0
## 1       0    1664      0            B       1 normal sbp    185.42     68.18
## 2       0    3071      1            B       1   high sbp    177.80     72.73
```

```
## 3       0    3071      0            A       0 normal sbp    175.26     72.73
## 4       0    3064      0            A       1 normal sbp    172.72     69.09
## 5       1    1885      1            A       1   high sbp    177.80     68.18
## 6       0    3102      0            A       0   high sbp    182.88     92.73
##   BMI0
## 1 19.8
## 2 23.0
## 3 23.7
## 4 23.2
## 5 21.6
## 6 27.7
```

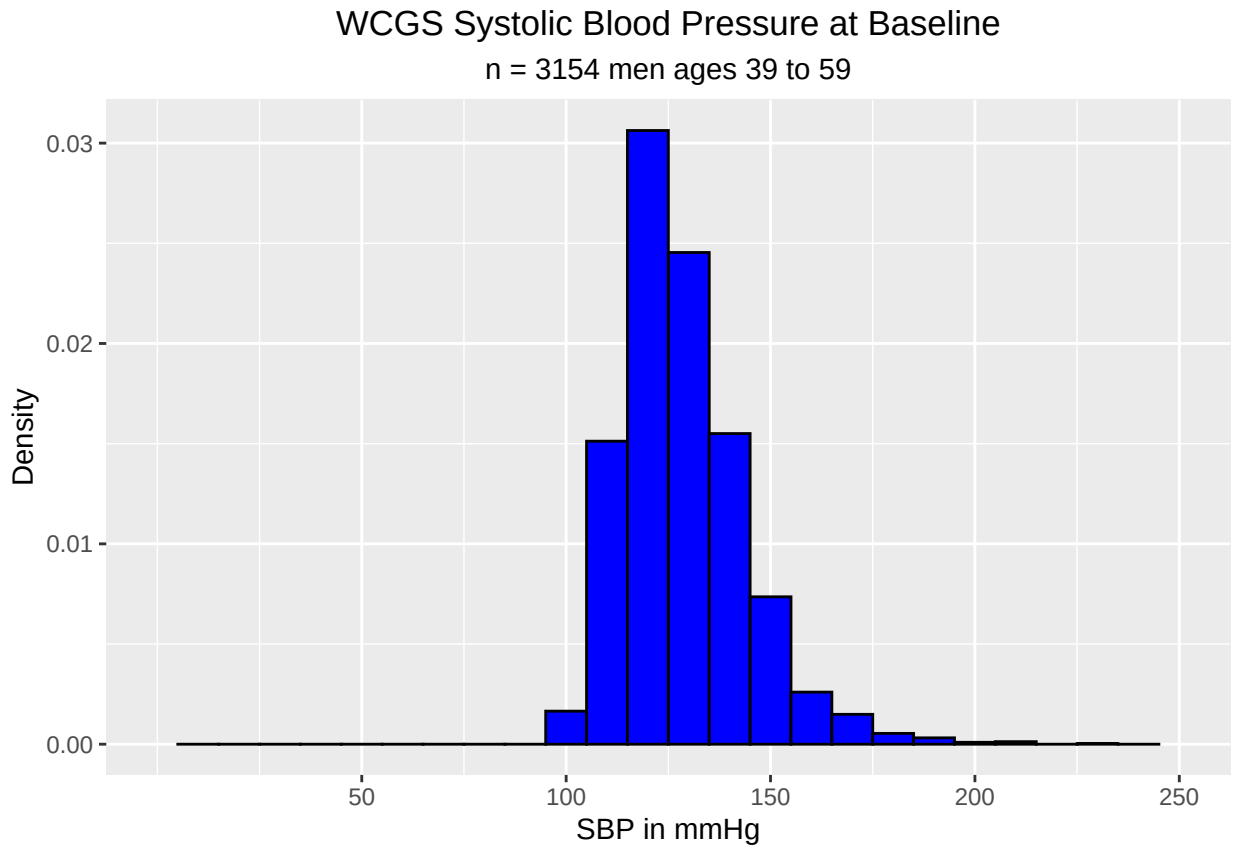
```
#>
```

Exercise 3 – Create Histograms & Box Plots

a) Create a histogram for the sbp0 variable.

```
#< task
ggplot(wcgs_new, aes(x = sbp0, y = ..density..)) +
  geom_histogram(binwidth = 10,
                 color = "black",
                 fill = "blue") +
  labs(title = "WCGS Systolic Blood Pressure at Baseline",
       subtitle = "n = 3154 men ages 39 to 59",
       x = "SBP in mmHg", y = "Density" ) +
  scale_x_continuous(limits = c(0, 250),
                     breaks = c(50, 100, 150, 200, 250),
                     labels = c(50, 100, 150, 200, 250)) +
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5))
```

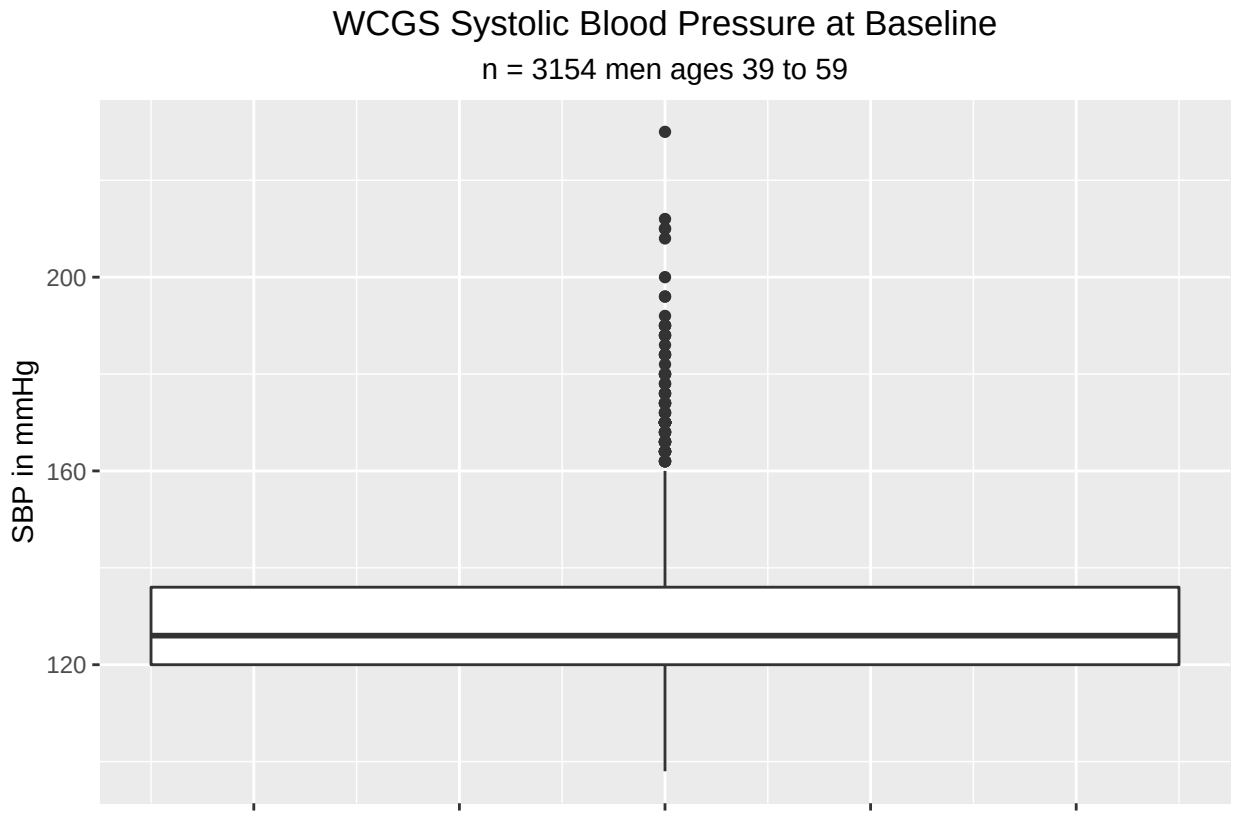
```
## Warning: Removed 2 rows containing missing values (geom_bar).
```




```
#>
```

b) Create a box plot for the sbp0 variable.

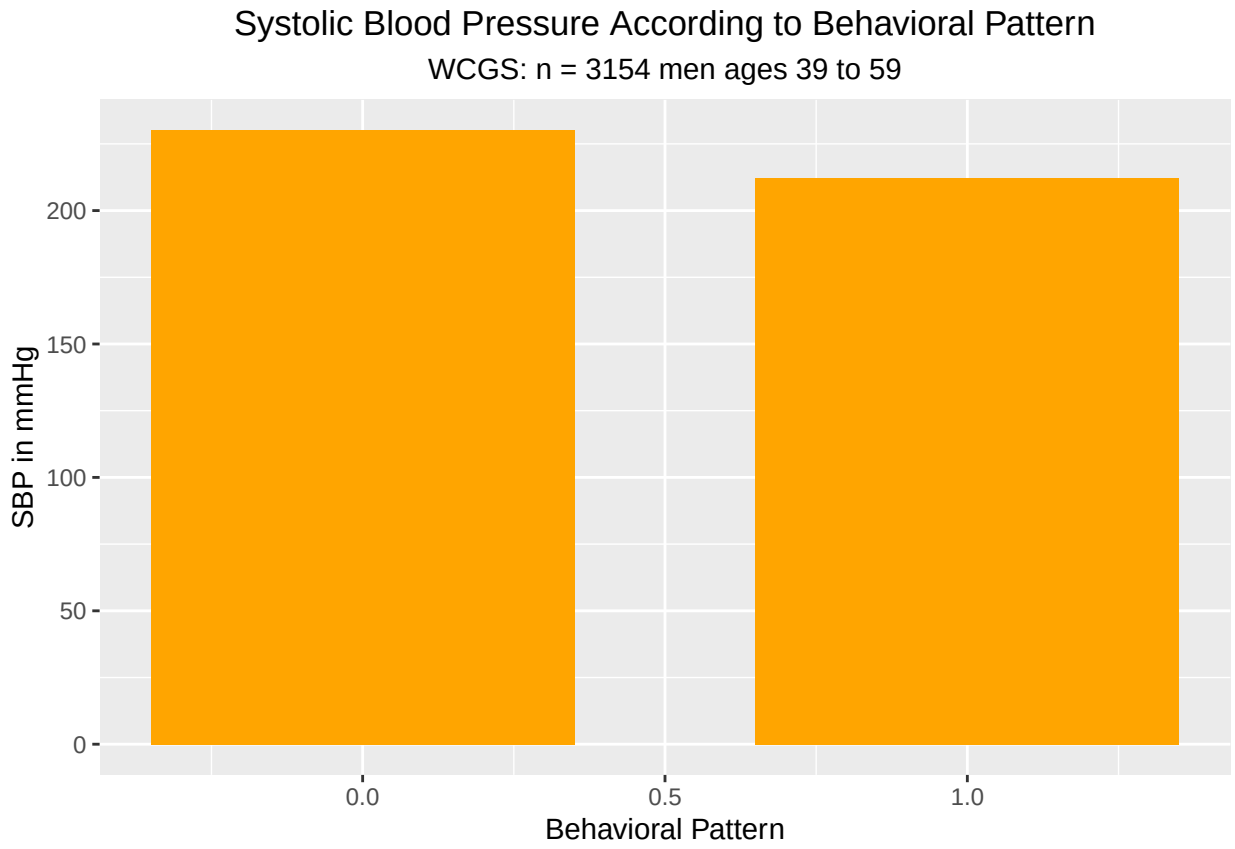
```
#< task
ggplot(wcgs_new, aes(y = sbp0)) +
  geom_boxplot(width = 0.5) +
  scale_x_continuous(labels = NULL) +
  labs(y = "SBP in mmHg",
       x = "",
       title = "WCGS Systolic Blood Pressure at Baseline",
       subtitle = "n = 3154 men ages 39 to 59") +
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5))
```




```
#>
```

c) Create a bar chart for the sbp0 variable for each level of the dibpat0 variable.

```
#< task
ggplot(wcgs_new, aes(y = sbp0)) +
  geom_col(aes(x = dibpat0),
           position = "dodge",
           fill = "orange",
           width = 0.7) +
  labs(y = "SBP in mmHg",
       x = "Behavioral Pattern",
       title = "Systolic Blood Pressure According to Behavioral Pattern",
       subtitle = "WCGS: n = 3154 men ages 39 to 59") +
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5))
```




```
#>
```

d) Create a two-way table of the behavioral pattern factor variable dibpat0_fact and high blood pressure highsbp0.

```
#< task
highsbp_dibpat_table <- table(wcgs_new$dibpat0_fact, wcgs_new$highsbp0)


# You will model your tables in R Lab Workbook 1 on the following two table outputs

addmargins(highsbp_dibpat_table)
```

```
##
##       high sbp normal sbp  Sum
##   A         305        1260 1565
##   B         388        1201 1589
##   Sum       693        2461 3154
```

```
round(prop.table(highsbp_dibpat_table), digits = 3)
```

```
##
##     high sbp normal sbp
##   A    0.097      0.399
##   B    0.123      0.381
```

```
#>
```