Lab 4 BIS 505b

Maria Ciarleglio

3/15/2021

Goal of Lab 4

In **Lab 4**, we will **(1)** introduce plotting using the `ggplot2` package, **(2)** use **simple linear regression** to examine the linear relationship between two quantitative variables and **(3)** examine **residual plots** to assess model assumption violations.

Analysis Data Set

In this lab, we will analyze data from the Framingham Heart Study `fhs_exam1.csv` (full data set imported as `fhs` in code chunk 3 above) to determine if systolic blood pressure `SYSBP` (y) is associated with `BMI` (x). In your exercises, you will determine if `SYSBP` (y) is associated with `DIABP` (x). We will examine these associations in a subset of `fhs` that includes the first 100 rows of data `fhs100`, created below:

```
# Select first 100 rows from fhs1 data frame for analysis
fhs100 <- fhs[1:100,]
dim(fhs100)    # should only include 100 rows
```

```
## [1] 100  36
```

Research Question

Our Research Question:

- Determine if systolic blood pressure is associated with BMI and if BMI is associated with age.

Variables of Interest:

- `SYSBP` = systolic blood pressure (mmHg)
- `DIABP` = diastolic blood pressure (mmHg)
- `BMI` = body mass index (kg/m^2)

Introduction to Plotting with `ggplot`

In this section, we will introduce a powerful package for creating graphics in **R** called `ggplot2`. We can use the `ggplot()` function in the `ggplot2` package to create a wide variety of plot types. We will use `ggplot()` to create **histograms**, **scatter plots** and **Normal Q-Q plots**. You will find that there are **many, many** different ways of doing something in `ggplot()`.

- **Components of a Plot**

Graphics are customized by adding additional *layers* to the basic plot area created using the initial call of the `ggplot()` function. One difference between `ggplot()` and base **R** plotting is that `ggplot()` works with **data frames** and cannot take individual vectors as arguments. The data used to make the plot is contained in the `dataframe` specified in the first layer of our plot, `ggplot(dataframe, aes(...))`. There are three components to each plot created in `ggplot()`:

1. **Data** (`data=`): The data frame used to create the plot
2. **Aesthetics** (`aes()`): Where the x- and y-axis variables are specified, as well as other plot aesthetics (e.g., color, size, and shape of point, grouping variable, height of bars)
3. **Geometry** (`geoms`): Where the graph type is specified (e.g., histogram, scatter plot, Q-Q plot)

The plot is generally **initialized** using components **(1)** and **(2)** (i.e., specifying the data frame and setting some plot aesthetic such as specifying the x-axis variable). Once the base setup is complete, we can add the desired plot as a layer on top of the base using different `geoms`. The `geom` functions can also specify their own aesthetics. We can also layer the geoms one on top of the other (e.g., adding a smoothed line to a scatter plot of points) by specifying more than one `geom` layer. Layers are added to your plot after the plus sign `+`. Note that the `na.rm=TRUE` option is available in most geoms to remove missing values. For example, `geom_histogram(na.rm=TRUE)` will remove missing values from the requested histogram. If you do not specify `na.rm=TRUE`, `ggplot()` *will* exclude missing values, but it will print a warning message to inform you that missing values were dropped. Using the `na.rm=TRUE` option will silently remove missing values. You can also use the `message=FALSE` code chunk option to suppress warning messages from printing your final **R** Markdown report.

Graph type	**geoms**
Histogram	`+ geom_histogram()`
Scatter plot	`+ geom_point()`
Q-Q plot	`+ geom_qq()`
Reference lines	`+ geom_abline()` (diagonal), `+ geom_hline()` (horizontal), `+ geom_vline()` (vertical), `+ geom_qq_line()` (Q-Q plot reference line)
Regression line or smoothed line	`+ geom_smooth()`

- **Title and Axes**

There are several options for controlling the **plot title** (`labs()`, `ggtitle()`), **axis labels** (`labs()`, `xlab()`, `ylab()`), and **axis limits** (`xlim()`, `ylim()`). To insert a line break in a title, use `\n`. For continuous axes, we also have the option to use the `scale_x_continuous()` and `scale_y_continuous()` functions to modify the axis label (`name=`), range (`limits=`), tick marks (`breaks=`), and tick labels (`labels=`).

Title/Axis Option	**Syntax**
Title, axis labels	`+ labs(title="plot title", x="xaxis label", y="yaxis label")` (include legend title option shown above, if applicable)
Title	`+ ggtitle("plot title")`
Axis labels	`+ xlab("xaxis label")`, `+ ylab("yaxis label")`
Axis limits	`+ xlim(c(min, max))`, `+ ylim(c(min, max))`
Continuous axis formatting	`+ scale_x_continuous(name="xaxis label", breaks=c(), nbreaks=, labels=c(), limits=c(min, max))`, `+ scale_y_continuous(name="yaxis label", breaks=c(), nbreaks=, labels=c(), limits=c(min, max))`

Adding a layer using the `theme()` function allows us to control the appearance (i.e., font type/family, font face, font color, font size, and justification) of the plot title and axis labels that we specified using the options above:

Title/Axis Formatting	**Syntax**
Title	`+ theme(plot.title = element_text(family, face, color, size),`
x-axis label	`axis.title.x = element_text(family, face, color, size),`

Title/Axis Formatting	Syntax
y-axis label	`axis.title.y = element_text(family, face, color, size))`

`element_text()` formatting options for `plot.title=`, `axis.title.x=`, and `axis.title.y=` include:

`theme()` `element_text()` Options	Syntax
Font family	`family=` (`mono`, `sans`, `serif`, among others)
Font face	`face=` (`plain` (default), `italic`, `bold` and `bold.italic`)
Font color	`color=`
Font size	`size=` in pts (default `=5`)
Horizontal justification	`hjust=` (between [0,1])
Vertical justification	`vjust=` (between [0,1])

For example,
`+ ggtitle("my plot title") + theme(plot.title = element_text(family="serif", face="bold", color="blue", size=10, hjust=0.5))`
will apply the plot title, “my plot title” and format the title with a Times New Roman style font that is bold, blue, and center-justified.

- **Themes**

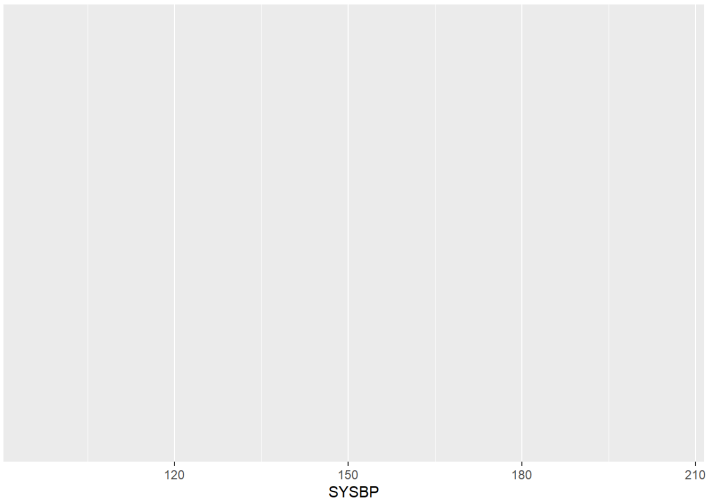
There are many themes that can be applied to change the appearance of the plots. For example, add the layer `+ theme_classic()` to a plot to display a white background with black lines for the x- and y-axis.

Themes Syntax	Description
`theme_gray()`	Default
`theme_bw()`	White background, gridlines, black box
`theme_light()`	White background, gridlines, gray box
`theme_dark()`	Dark gray background, gridlines
`theme_minimal()`	White background, gridlines
`theme_classic()`	White background, no gridlines, black axes
`theme_void()`	White background, no axes

Histogram

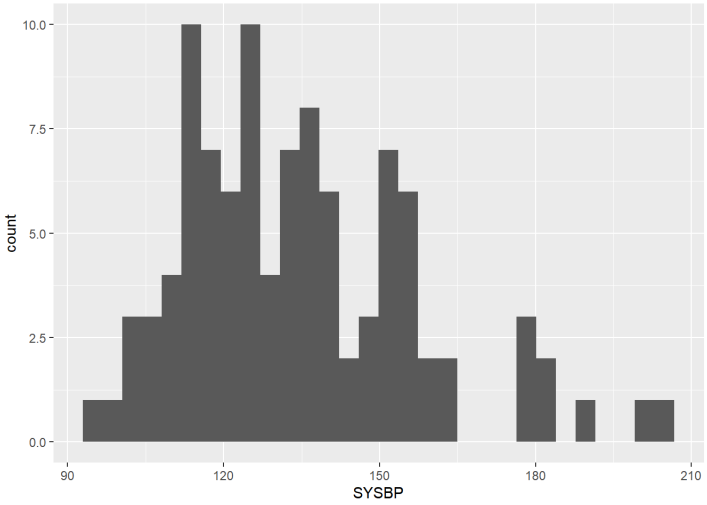
We begin by discussing syntax for creating **histograms**. In this section, we will create a histogram of the variable `SYSBP` contained in the `fhs100` data frame. All `ggplot()` plots are **initialized** in the same way: by specifying the data frame name (`data=`) and specifying plot aesthetics (`aes()`) that will carry over in all layers. Notice how the line of code below initializes the plot and sets up the plot area; it does not display a histogram since we did not yet specify the `geom_histogram()` layer:

```
# Initialize the plot by specifying the data source and the plot aesthetic (x-variable, here, for the histogram)
ggplot(data = fhs100, aes(x = SYSBP))
```



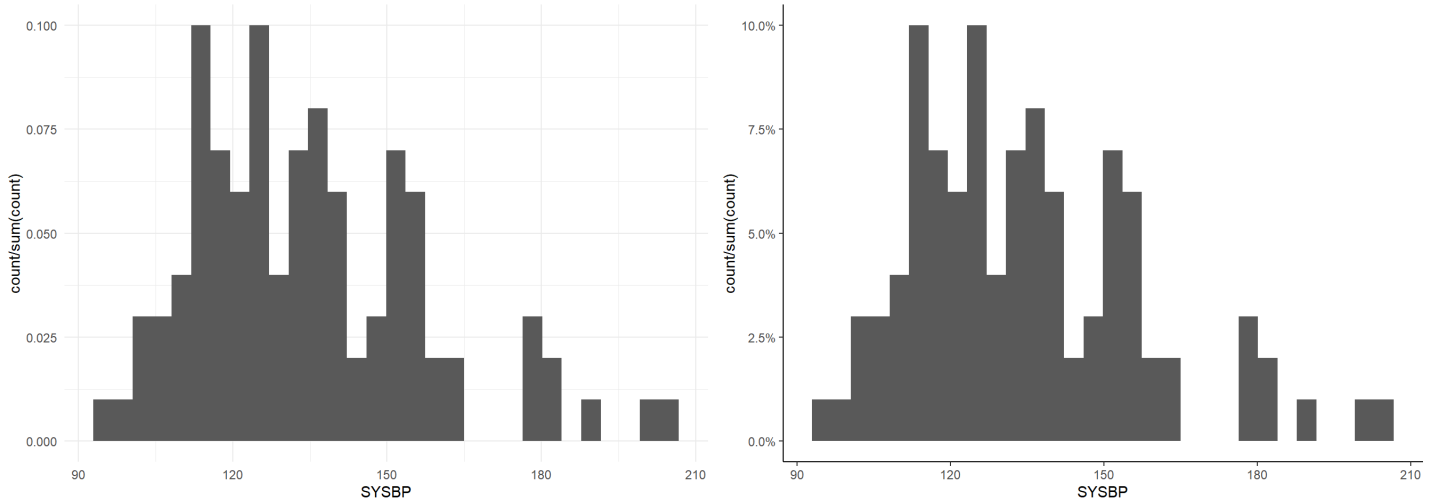
Next, we add the histogram layer to the plot after a `+`. By default, a **frequency histogram** is produced (i.e., `y=stat(count)`). Therefore, the `y=` variable does not need to be specified when producing a frequency histogram.

```
# Basic (uncustomized) frequency histogram (R chooses default number of bins = 30)
ggplot(data = fhs100, aes(x = SYSBP)) +
  geom_histogram(na.rm = TRUE)
```



We can produce a **relative frequency histogram** by specifying the y-axis aesthetic `y = stat(count)/sum(stat(count))` in the `ggplot(aes())`. To format the y-axis to display a percentage instead of a proportion, use the `scales` package and add the following y-axis format layer to the plot, `+ scale_y_continuous(labels=scales::percent_format())`. We also apply two different "themes" below to change the appearance of each plot (`+ theme_minimal()` and `+ theme_classic()`).

```
# Relative frequency histogram
ggplot(data = fhs100, aes(x = SYSBP, y = stat(count)/sum(stat(count)))) + # relative frequency on y-axis
  geom_histogram(na.rm = TRUE) +
  theme_minimal()

# Formatting y-axis to display percentage instead of proportion
ggplot(data = fhs100, aes(x = SYSBP)) +
  geom_histogram(aes(y = stat(count)/sum(stat(count))), na.rm = TRUE) +
  scale_y_continuous(labels = scales::percent_format()) + # format y-axis as percentage
  theme_classic()
```

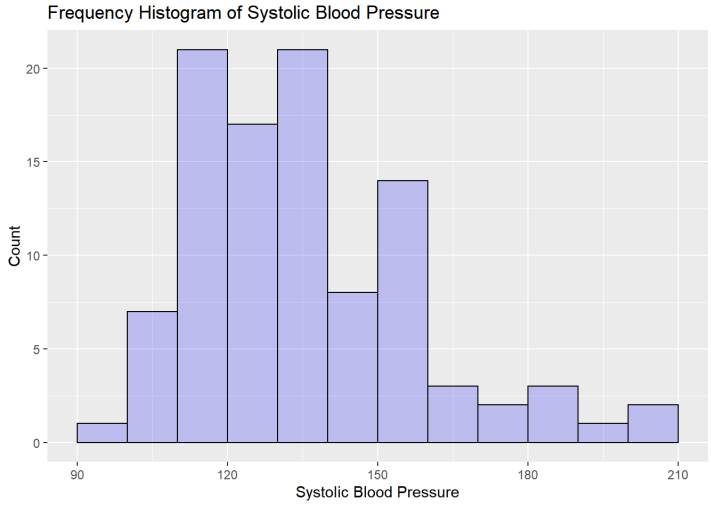


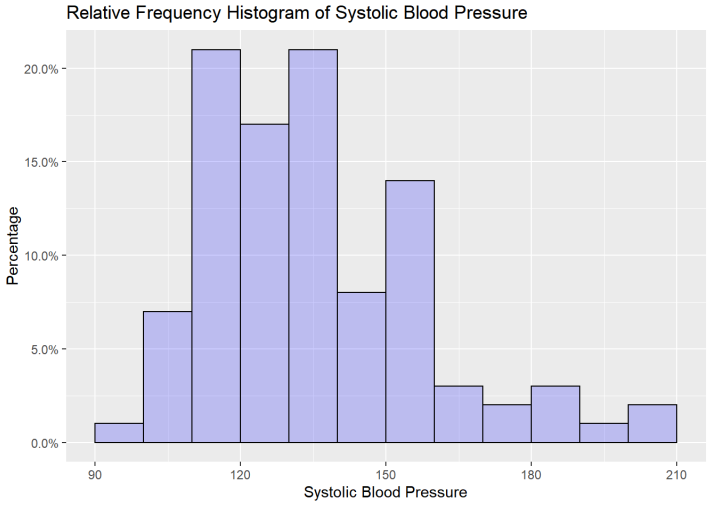
We can further customize the appearance of histogram using the following options:

Option	**Syntax in `geom_histogram()`**
Histogram bin boundaries	`breaks=c()`
Histogram bin width	`binwidth=`
Bar outline color	`col=`
Bar fill color	`fill=`
Bar transparency	`alpha=` (`0` = transparent - `1` = solid)
Left or right-closed bins	`closed=left` or `closed=right`

Below, the histograms of `SYSBP` are customized by specifying the histogram bin breaks (`breaks=`), requesting a black bar outline (`col="black"`), a blue bar fill color (`col="blue"`), semi-transparent bar coloring (`alpha=0.2`), and left-closed bins (`closed="left"`). In the frequency histogram on the **left**, the `labs()` function is used to change the x- and y-axis labels and add a plot title. In the relative-frequency histogram on the **right**, the `ggtitle()` function is used to add a plot title, the `xlab()` function is used to change the x-axis label, and the `scale_y_continuous()` function is used to change the y-axis label (`name=`), format the y-axis labels as percentages (`labels=`), and specify the tick mark placement every 5% (`breaks=`). The same modifications can be applied to the x-axis using the `scale_x_continuous()` function.

```r
# Customized histogram (breaks option)
ggplot(data = fhs100, aes(x = SYSBP)) +
  geom_histogram(breaks = seq(90, 210, by = 10),  # bin breaks (boundaries)
                 col = "black",                   # bar outline color
                 fill = "blue",                   # bar fill color
                 alpha = 0.2,                     # bar transparency
                 closed = "left",                 # left-closed (my preferred method)
                 na.rm = TRUE) +
  labs(title = "Frequency Histogram of Systolic Blood Pressure",
       x = "Systolic Blood Pressure", y = "Count")   # labeling

# Customized relative frequency histogram
ggplot(data = fhs100, aes(x = SYSBP)) +
  geom_histogram(aes(y = stat(count)/sum(stat(count))),  # relative frequency on y-axis
                 breaks = seq(90, 210, by = 10),
                 col = "black",
                 fill = "blue",
                 alpha = 0.2,
                 closed = "left",
                 na.rm = TRUE) +
  ggtitle("Relative Frequency Histogram of Systolic Blood Pressure") +       # equivalent way of applying title
  xlab("Systolic Blood Pressure") +                                          #   and axis labels
  scale_y_continuous(name = "Percentage", labels = scales::percent_format(), # format y-axis as percentage
                     breaks = seq(0, 0.25, by = 0.05))                       #   and specify axis tick placement
```





Scatter Plot

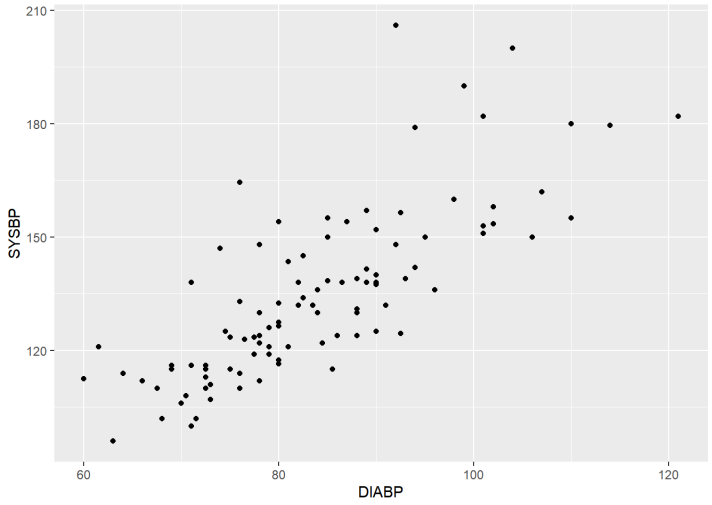
A **scatter plot** is produced using the `geom_point()` function. We can customize the appearance of the scatter plot points using the following options:

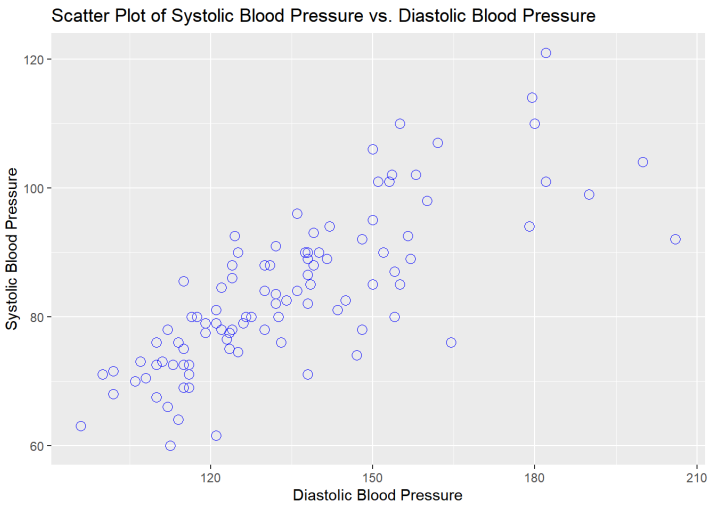
Option	**Syntax in `geom_point()`**
Point size	`size=`
Point shape (pch)	`shape=` (default `=19`, closed circle)
Point color	`col=` (solid shapes: full color, outlined shapes: outline color)
Point fill	`fill=` (outlined shapes: fill color)
Point transparency	`alpha=` (`0` = transparent - `1` = solid)

For example, change the point size and shape using the `size=` and `shape=` options in the `geom_point()` function. The plotting symbol (`shape=`) takes standard `pch` point shapes used in base **R** `plot()` function. `shape=1` gives an open circle plotting character while `shape=21` gives an outlined circle with outline color `col=` and fill color `fill=`.

Below, we create a scatter plot of systolic blood pressure (y) vs. diastolic blood pressure (x).

```
# Basic scatter plot SYSBP vs. DIABP
ggplot(data = fhs100, aes(x = DIABP, y = SYSBP)) +
  geom_point()

# Add title, axis labels, change point size and shape
ggplot(data = fhs100, aes(x = SYSBP, y = DIABP)) +
  geom_point(size = 3, shape = 1, col = "blue") +
  labs(title = "Scatter Plot of Systolic Blood Pressure vs. Diastolic Blood Pressure",
       x = "Diastolic Blood Pressure", y = "Systolic Blood Pressure")
```

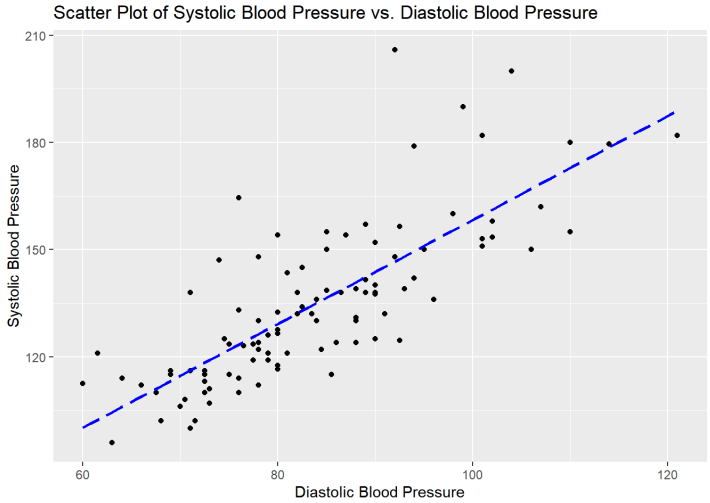




Add a regression line to the scatter plot by adding a `geom_smooth()` layer. `method=lm` adds the fitted simple linear regression to the plot. The `method=` option also accepts other values such as `loess` for a smooth local regression. The option `se=FALSE` is used to prevent the default confidence interval for from being displayed in the plot.

Option	**Syntax in `geom_smooth()`**
Line thickness	`size=` (default `=1`)
Line color	`col=`
Line type	`linetype=` (`solid` (default), `twodash`, `longdash`, `dotted`, `dotdash`, `dashed`)
Fitted line	`method=` (`lm` = linear regression, `loess` = loess smoothed function)
Formula for fitted line	`formula=` (`y ~ x` for linear model)
Display confidence interval	`se=` (default `=TRUE`)

```
# Add regression line to scatter plot
ggplot(data = fhs100, aes(x = DIABP, y = SYSBP)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x,
              se = FALSE, col = "blue", linetype = "longdash") +    # add regression line (no CI)
  labs(title = "Scatter Plot of Systolic Blood Pressure vs. Diastolic Blood Pressure",
       x = "Diastolic Blood Pressure", y = "Systolic Blood Pressure")
```



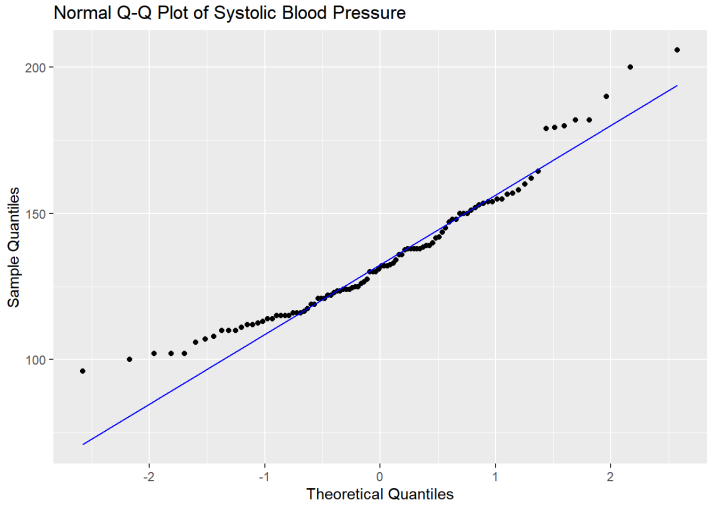
Normal Q-Q Plot

A **Normal Q–Q plot** is a graphical method for assessing if a sample comes from a Normal distribution. This plot is created in `ggplot()` using the `geom_qq()` function. This function plots the sample quantiles (of the variable being evaluated) against theoretical quantiles of a standard Normal random variable. The same point options discussed above for the scatter plot (`size` , `shape` , `col` , `fill` , `alpha`) can be applied to the Q-Q plot.

A Normal Q-Q plot traditionally includes a 45-degree reference line, which is added to the plot using a `geom_qq_line()` layer. If the sample in question follows a Normal distribution, the points should fall along this reference line. The greater the departure from this reference line, the greater the evidence for the conclusion that the sample does not follow a Normal distribution.

Below, we create a Normal Q-Q plot of the `SYSBP` variable to determine if it follows a Normal distribution.

```
# Normal Q-Q plot of SYSBP and reference line
ggplot(data = fhs100, aes(sample = SYSBP)) +
  geom_qq() +
  geom_qq_line(col = "blue") +
  labs(title = "Normal Q-Q Plot of Systolic Blood Pressure",
       x = "Theoretical Quantiles", y = "Sample Quantiles")
```



The plot of the quantiles of `SYSBP` against the quantiles of a standard Normal distribution gives a Normal Q-Q plot that is curved upward. This shape is typical in right-skewed distributions. The histogram of `SYSBP` also shows

Linear Regression

Linear regression is used to describe the linear relationship between a quantitative predictor variable y and one or more explanatory variables x or $x_1, \ldots, x_p$. In the case of **simple linear regression**, one predictor variable is of interest and the equation of the population regression line is given by $y = \alpha + \beta x + \epsilon$, where α is the intercept and β is the slope of the line. The equation of this line is estimated using the method of **least squares**, giving a fitted line, $\hat{y} = a + bx$.

- The fitted line is used to **predict** or **estimate** the expected value of y for a given value of x by plugging values for x into the fitted equation, $\hat{y} = a + bx$.
- The estimated slope b is used to **describe the association** between x and y. For example, a one-unit increase in x is associated with an expected change in y that is equal to b.

A test of the slope, β (i.e., $H_0 : \beta = 0$ vs. $H_1 : \beta \neq 0$) is used to determine if a linear association exists between x and y. A test of β is performed using the t-statistic, $t = \frac{b}{s_b}$, which is compared to a t-distribution with $n - 2$ degrees of freedom in simple linear regression.

The `lm()` function in **R** is used to estimate the regression coefficients (i.e., the intercept and slope parameter(s)) of the linear model. The result of the `lm()` function is usually saved as an object (e.g., `regobject`) and the `summary()` function is applied to that object (`summary(regobject)`) to output detailed results.

`lm()` Function Arguments	Option Definition
`formula=`	`analysis_variable ~ group_variable`
`data=`	Data frame containing sample data

A $100(1 - \alpha)$% confidence interval for the parameter β can be estimated using a confidence interval of the form, $b \pm t_{1-\alpha/2;n-2}\ s_b$.

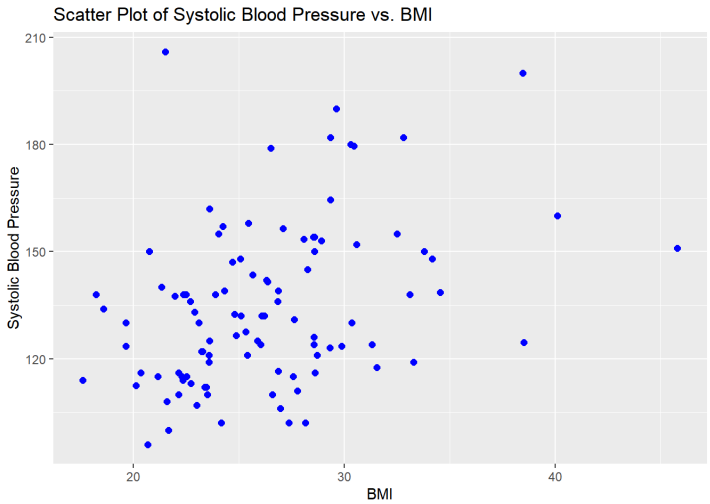
The `confint(regobject)` function will be used below to print 95% confidence intervals for the model parameters. Adjust the confidence level through the `level=` argument in the `confint()` function. By default, 95% confidence intervals (`level=0.95`) are produced.

Finally, the fitted model can be used to estimate or predict a value of y for a given value of x using the `predict()` function. A new data frame must be created and specified in the `newdata=` argument of the `predict()` function that contains the values of x for which we want to predict values of y. The `predict()` function will be applied in the example below.

Estimating the Simple Linear Regression Model

Suppose we are interested in estimating the effect of BMI (x) on systolic blood pressure (y). We begin by looking at a scatter plot of the association.

```
# Scatter plot: SYSBP vs. BMI
ggplot(data = fhs100, aes(x = BMI, y = SYSBP)) +
  geom_point(size = 2, shape = 19, col = "blue") +
  labs(title = "Scatter Plot of Systolic Blood Pressure vs. BMI",
       x = "BMI", y = "Systolic Blood Pressure")
```



The scatter plot shows a modest positive linear relationship between BMI and systolic blood pressure.

Exercise: In the previous section, we created a scatter plot of the relationship between `DIABP` (x) and `SYSBP` (y). Modify the appearance of this scatter plot using your theme of choice. Does there appear to be a linear relationship between these two variables?

▶ Answer:

We fit the model using the `lm()` function and save the fitted model to the object `reg`. We output a summary of the results using `summary(reg)` and the 95% CIs using `confint(reg)`.

```
# Simple linear regression model: modeling SYSBP using BMI
reg <- lm(SYSBP ~ BMI, data = fhs100)

# Output results of fitted model
summary(reg)
```

```
##
## Call:
## lm(formula = SYSBP ~ BMI, data = fhs100)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -35.621 -15.616  -3.539  10.829  79.836
##
## Coefficients:
##             Estimate Std. Error t value       Pr(>|t|)
## (Intercept)  89.0482    11.7673   7.567 0.000000000021
## BMI           1.7255     0.4398   3.923       0.000162
##
## Residual standard error: 21.11 on 98 degrees of freedom
## Multiple R-squared:  0.1357, Adjusted R-squared:  0.1269
## F-statistic: 15.39 on 1 and 98 DF,  p-value: 0.0001622
```

```
# Confidence intervals for model parameters (intercept and slope(s))
confint(reg, level = 0.95)
```

```
##                  2.5 %     97.5 %
## (Intercept) 65.6963449 112.399956
## BMI          0.8527064   2.598317
```

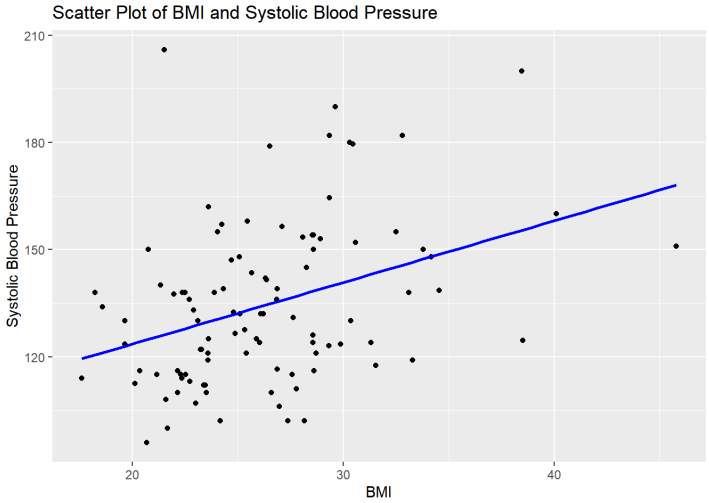
The **fitted model** is given by the equation, $\hat{y} = 89.05 + 1.73$ `BMI`. The **estimated intercept** a is equal to 89.05 [95% CI (65.7, 112.4)], but cannot be interpreted in this example since a BMI of 0 is not meaningful. The **estimated slope** b is equal to 1.73 [95% CI (0.85, 2.6)] indicates that a 1-unit increase in BMI is associated with a 1.73-unit average increase in systolic blood pressure.

The significance test of the slope ($H_0 : \beta = 0$ vs. $\beta \neq 0$) reports a t-statistic t=3.92, which is compared to a t-distribution with 98 degrees of freedom. This yields a highly significant p-value = 0.0002.

The **R-squared** or **Coefficient of Determination** is rather low at 0.136, indicating that `BMI` only explains 13.6% of the total variability in systolic blood pressure. The variability about the regression line $\sigma_{y|x}$ is estimated by the "**residual standard error**" in the output above and is equal to $s_{y|x}$ = 21.11. This estimated variability in Y about the regression line is expected to hold for all values of X (i.e., the constant variance assumption of linear regression).

The **fitted line** is added to the scatter plot below, drawing attention to the positive relationship between BMI and systolic blood pressure

```
# Add regression line to scatter plot
ggplot(data = fhs100, aes(x = BMI, y = SYSBP)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x,
              se = FALSE, col = "blue") +
  labs(title = "Scatter Plot of BMI and Systolic Blood Pressure",
       x = "BMI", y = "Systolic Blood Pressure")
```



Exercise: Fit a linear model to the relationship between `DIABP` (x) and `SYSBP` (y). Is there a statistically significant association between these two variables? If so, interpret the estimated slope. What is the critical value for statistical significance in the test of the slope parameter at the $\alpha = 0.05$-level?

▶ Answer:

ANOVA Table

The **ANOVA table** allows us to see how the value of R^2 is computed since $R^2 = \frac{SS_{model}}{SS_{total}}$, which gives 6859.7/(6859.7 + 43675.93) = 0.14.

```
# ANOVA table from fitted model
anova(reg)
```

```
## Analysis of Variance Table
##
## Response: SYSBP
##           Df Sum Sq Mean Sq F value    Pr(>F)
## BMI        1   6860  6859.7  15.392 0.0001622
## Residuals 98  43676   445.7
```

The F-test presented in the ANOVA table supports the conclusion that the model explains a significant amount of variability in the response, `SYSBP` (p = 0.0002). The F-statistic F=15.39 is compared to an F-distribution with 1 and 98 degrees of freedom. This F-test is equivalent to the t-test of the slope β performed above in the case of simple linear regression with a single predictor x.

The residual standard error that is reported in the `lm()` output above, $s_{y|x}$ = 21.11, is equal to $\sqrt{MSE}$, or the square-root of 445.67.

Exercise: Based on your output from `lm()` in the previous exercise, report the Coefficient of Determination of the linear model that uses `DIABP` (x) to model `SYSBP` (y). Using the `cor()` function, compute the Pearson correlation r between `DIABP` and `SYSBP`; square this value, r^2. Is this value equal to the Coefficient of Determination from your linear regression? (Hint: it should be!)

▶ Answer:

Prediction

To **predict** the value of systolic blood pressure for a particular value of BMI, create a new data frame that contains the values of x for which we would like to predict y. This data frame is then input into the `newdata=` argument of the `predict()` function along with the regression model object that contains the fitted model used in the estimation (i.e., `reg`). A **confidence interval** for the mean response at given values of $x = x^*$ (i.e., $\mu_{y|x^*}$) is requested by specifying the argument `interval = "confidence"` in the `predict()` function. A **prediction interval** for a single individual's response value at given values of $x = x^*$ can be requested by specifying the argument `interval = "prediction"` in the `predict()` function.

The code below will estimate the average systolic blood pressure in those with BMI equal to 30 and 35 along with 95% confidence intervals for the mean systolic blood pressure and prediction intervals at these values of BMI.

```
# Values of x (BMI) used to estimate y (SYSBP)
x.star <- data.frame(BMI = c(30, 35))
x.star
```

```
##   BMI
## 1  30
## 2  35
```

```
# Fitted value and lower and upper CI for mean at values of x in x.star (default level = .95)
predict(reg, newdata = x.star, interval = "confidence", level = 0.95)
```

```
##        fit      lwr      upr
## 1 140.8135 135.5350 146.0920
## 2 149.4411 140.7845 158.0976
```

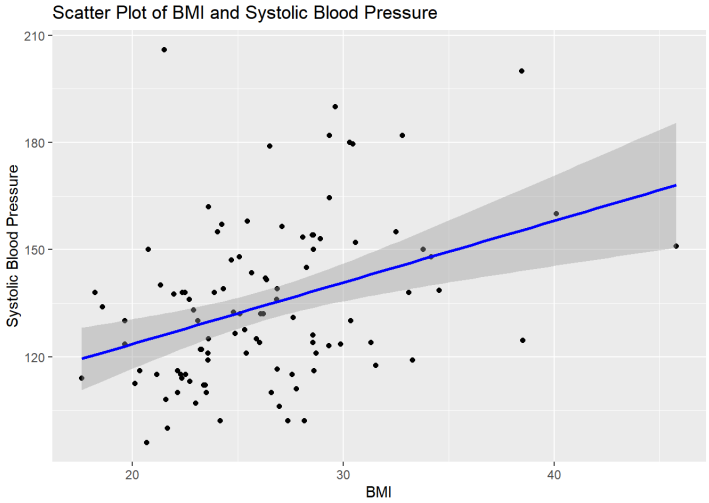
```
# Fitted value and lower and upper prediction interval at values of x in x.star
predict(reg, newdata = x.star, interval = "prediction", level = 0.95)
```

```
##        fit       lwr      upr
## 1 140.8135  98.58825 183.0388
## 2 149.4411 106.66204 192.2201
```

The fitted line estimates a mean systolic blood pressure of 140.81 in those with BMI = 30 [95% confidence interval (135.53, 146.09)]. The estimated mean systolic blood pressure in those with BMI = 35 is equal to 149.44 [95% confidence interval (140.78, 158.1)]. As expected, the prediction intervals at each of these input values of BMI are wider than the confidence intervals for the mean.

The confidence interval is shown by default when the fitted line is added to the scatter plot. Notice how the confidence interval is narrowest at the mean of the x-variable, 26.3.

```
# Add regression line and confidence interval for the estimated mean
ggplot(data = fhs100, aes(x = BMI, y = SYSBP)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x,
              col = "blue") +
  labs(title = "Scatter Plot of BMI and Systolic Blood Pressure",
       x = "BMI", y = "Systolic Blood Pressure")
```



Checking Assumptions

The **residuals** $e_i = y_i - \hat{y}_i$ play a major role in checking assumptions of the linear regression model. Based on the assumptions of the linear regression model:

- The histogram of the residuals should be symmetric and bell-shaped
- The residuals should be uncorrelated, have constant variance, and exhibit a random pattern about a horizontal reference line $y = 0$

The residual plots can identify outlying values and suggest if **transformations** of the data are necessary to improve model fit. Transformations can be used to:

1. Stabilize variance (reduce heteroscedasticity)
2. Normalize residuals
3. Linearize regression model
4. Reduce the influence of unusual or extreme values

In general, transforming the y variable corrects problems with error terms (and may help nonlinearity) and transforming the x variable primarily corrects nonlinearity. Common transformations include $\sqrt{()}$, inverse, $\log()$, and $\exp()$.

Summary Statistics of Residuals

The `predict()` function in **R** calculates fitted values $\hat{y}$ for each observation included in the model. We used the `predict()` function above to predict y for specific values of $x = x^*$. Here, when `predict()` is applied to the model object `reg`, the `predict()` function will report fitted values for all observations used to estimate the model. That is, every observation used to fit the model will have a predicted or fitted value computed using the estimated model $\hat{y} = a + bx$.

The `resid()` function extracts model residuals for each observation $e_i = y_i - \hat{y}_i$.

Below, the predicted values `predicted` and the residuals `residuals` are appended to the reduced data frame `dat` that includes the variables analyzed in this regression model, `SYSBP` and `BMI`.

```
# Data frame that includes subset of the variables used in today's analysis
dat <- select(fhs100, c(RANDID, SYSBP, BMI))

# Appending column of predicted values (yhat) to dat
dat$predicted <- predict(reg)

# Appending column of residuals to dat
dat$residuals <- resid(reg)

# Printing the first 3 rows of dat
head(dat, n = 3)
```

```
##   RANDID SYSBP   BMI predicted  residuals
## 1   2448 106.0 26.97  135.5852 -29.585202
## 2   6238 121.0 28.73  138.6221 -17.622103
## 3   9428 127.5 25.34  132.7726  -5.272618
```

In addition to viewing plots, we can examine **summary statistics** of the residuals:

```
# Summary statistics of residuals
summary(dat$residuals)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -35.621 -15.616  -3.539   0.000  10.829  79.836
```

There appears to be a at least one very large positive residual. We can sort the `dat` data frame on the absolute value of the `residuals` variable from largest to smallest and print the first few rows of sorted `dat` to determine if this maximum residual of 79.84 is out of line with the other residuals:

```
# Sort dat data frame by largest |residuals|
dat <- dat[order(-abs(dat$residuals)),]
head(dat)
```

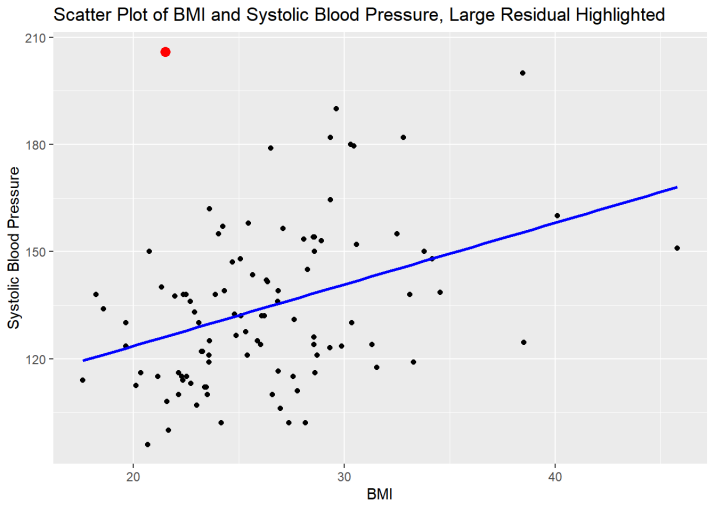
```
##    RANDID SYSBP   BMI predicted residuals
## 47  95541   206 21.51  126.1639  79.83609
## 69 162207   190 29.64  140.1923  49.80768
## 90 202101   200 38.46  155.4113  44.58867
## 83 188334   179 26.52  134.8087  44.19128
## 87 199546   182 29.35  139.6919  42.30808
## 6   11263   180 30.30  141.3312  38.66884
```

Subject 95541 has a residual that is much larger than the other subjects. This individual has a low BMI but a high systolic blood pressure. To highlight this individual in the scatter plot, we first create a data frame subset that contains the data points to be highlighted `highlight_fhs100` and then add another layer to the scatter plot using `geom_point()`. Notice that I did not specify the data frame and plot aesthetics in the plot initialization `ggplot()`. This is because what is specified in the plot initialization will carry through into all layers of the `ggplot()`, including the data frame that can be specified in the `data=` argument of `ggplot()`. Since I am calling on two different data frames (`fhs100` and `highlight_100`) to construct this plot, I specify the data frame that I am using separately in each layer.

```
# filter fhs100 to highlight points according to certain criteria
highlight_fhs100 <- subset(fhs100, RANDID == 95541)
highlight_fhs100
```

```
##    RANDID SEX TOTCHOL AGE SYSBP DIABP CURSMOKE CIGPDAY   BMI DIABETES BPMEDS
## 47  95541   2     311  53   206    92        0       0 21.51        1      1
##    HEARTRTE GLUCOSE PREVCHD PREVAP PREVMI PREVSTRK PREVHYP TIME PERIOD DEATH
## 47       76     215       0      0      0        0       1    0      1     1
##    ANGINA HOSPMI MI_FCHD ANYCHD STROKE CVD HYPERTEN TIMEAP TIMEMI TIMEMIFC
## 47      0      0       0      0      0   0        1   2697   2697     2697
##    TIMECHD TIMESTRK TIMECVD TIMEDTH TIMEHYP
## 47    2697     2697    2697    2697       0
```

```
# Adding new geom_point() layer with highlighted point(s)
ggplot() +
  geom_point(data = fhs100, aes(x = BMI, y = SYSBP)) +     # black points
  geom_smooth(data = fhs100, aes(x = BMI, y = SYSBP),      # blue fitted line
              method = "lm", formula = y ~ x,
              se = FALSE, col = "blue") +
  labs(title = "Scatter Plot of BMI and Systolic Blood Pressure, Large Residual Highlighted",
       x = "BMI", y = "Systolic Blood Pressure") +
  geom_point(data = highlight_fhs100, aes(x = BMI, y = SYSBP),
             col = "red", size = 3)                        # red highlighted point
```



In a case such as this, we may want to check if this data point is valid. For example, if data collection is through a manual chart review, go back to the original data source to verify that these data were recorded correctly. If we are unable to check the validity of this point, then we would not normally discard data from the analysis. However, we can run a **sensitivity analysis** that removes this value from the data set (below), especially if this point is causing issues with the fit of the model.

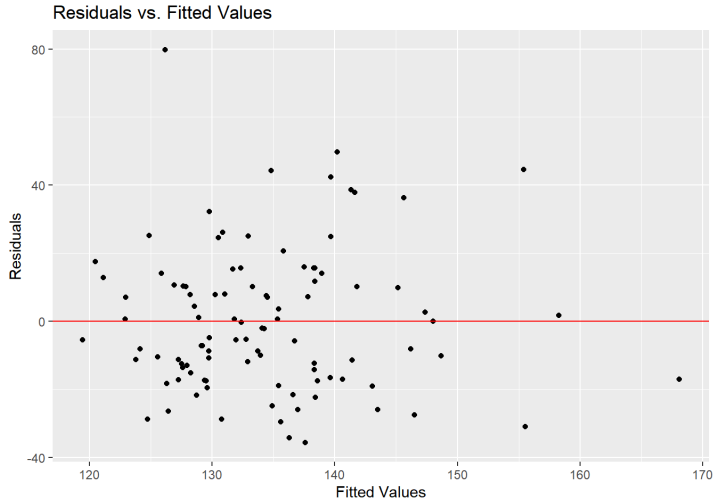
Residual Plots

A **scatter plot** of residuals `dat$residuals` (y-axis) vs. fitted values `dat$predicted` (x-axis) should exhibit a random pattern. This plot allows us to check several model assumptions:

- **Homoscedasticity** or **constant variance**: The variability in the residuals in this plot should also be fairly constant if there is no violation of the constant variance assumption.
- **Outliers**: Outliers will have large residuals that are out of line with the other model residuals.
- **Linearity assumption**: This plot should exhibit a random pattern. Patterns in the data could such as curvature indicate a violation of linearity.

We create a plot of residuals vs. fitted values and add a horizontal reference line at $y = 0$. Residuals should be randomly scattered above and below this reference line. We use the data frame `dat` to create these residual plots since this data frame contains the model residuals.

```
# Plot of residuals vs. fitted values
ggplot(data = dat, aes(x = predicted, y = residuals)) +
  geom_point() +
  geom_hline(yintercept = 0, col = "red") +  # add horizontal line at 0
  labs(title = "Residuals vs. Fitted Values",
       x = "Fitted Values", y = "Residuals")
```

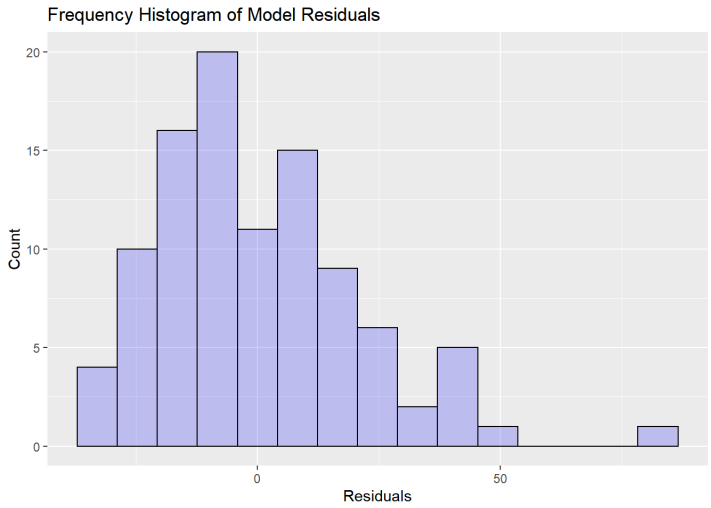


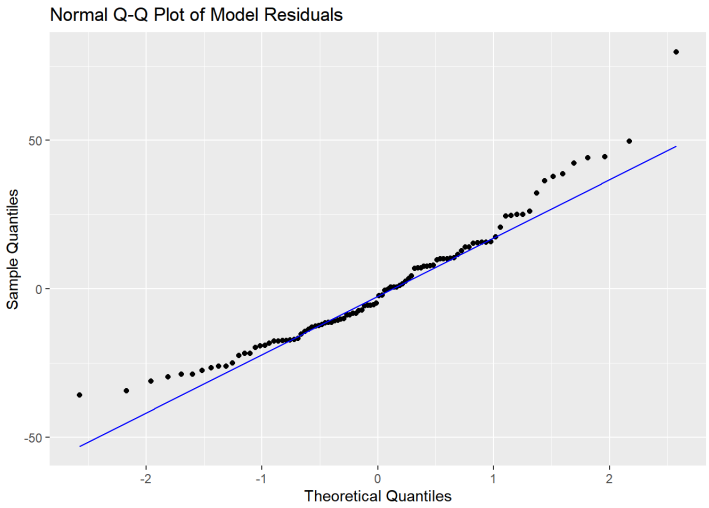
The plot of residuals vs. fitted values above does appear random. Subject 95541's large residual is an apparent outlier since her residual is far from the main cluster of residuals.

A **Normal Q-Q plot** and/or **histogram** of the residuals `dat$residuals` is commonly used to check:

- **Normality** of the residuals.

```
# Histogram of residuals
ggplot(data = dat, aes(x = residuals)) +
  geom_histogram(bins = 15,
                 col = "black",
                 fill = "blue",
                 alpha = 0.2,
                 closed = "left",
                 na.rm = TRUE) +
  labs(title = "Frequency Histogram of Model Residuals",
       x = "Residuals", y = "Count")

# Normal Q-Q plot of residuals
ggplot(dat, aes(sample = residuals)) +
  geom_qq() +
  geom_qq_line(col = "blue") +
  labs(title = "Normal Q-Q Plot of Model Residuals",
       x = "Theoretical Quantiles", y = "Sample Quantiles")
```





The histogram of residuals and the Normal Q-Q plot does indicate right skewed residuals. Subject 95541's large residual is clearly seen in the histogram. The model results are fairly robust to departures from normality. However, we will run a **sensitivity analysis** to determine if removing the outlier will greatly improve the fit of the model.

Exercise: Create a plot of residuals vs. fitted values of the linear model that uses `DIABP` (x) to model `SYSBP` (y). Does the plot suggest any violations of the linear model assumptions?

▶ Answer:

Sensitivity Analysis

In a **sensitivity analysis**, Subject 95541 will be excluded to determine if the model fit improves. Sensitivity analyses are reported along with the analysis on the full data set. Sensitivity analyses are often used to either:

1. show that the model results change or the model fit improves when looking at a subset of observations or
2. show that the model results are unaffected by a subset of observations.

Below, we are creating a data frame `sensitivity_fhs100` that excludes `RANDID == 95541` .

```
# filter fhs100 to remove or exclude points according to certain criteria
sensitivity_fhs100 <- subset(fhs100, RANDID != 95541)
dim(sensitivity_fhs100)
```

```
## [1] 99 36
```

Now, we re-run the model using `data=sensitivity_fhs100` .

```
# Sensitivity analysis removing RANDID == 95541
reg.sens <- lm(SYSBP ~ BMI, data = sensitivity_fhs100)

# Output results of fitted model
summary(reg.sens)
```

```
##
## Call:
## lm(formula = SYSBP ~ BMI, data = sensitivity_fhs100)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -35.118 -14.829  -3.138  11.953  50.058
##
## Coefficients:
##             Estimate Std. Error t value        Pr(>|t|)
## (Intercept)  83.7559    10.9870   7.623 0.0000000000168
## BMI           1.8956     0.4099   4.624 0.0000116029614
##
## Residual standard error: 19.58 on 97 degrees of freedom
## Multiple R-squared:  0.1806, Adjusted R-squared:  0.1722
## F-statistic: 21.38 on 1 and 97 DF,  p-value: 0.0000116
```

The sensitivity analysis **fitted model** is given by the equation, $\hat{y} = 83.76 + 1.9$ `BMI` . The **estimated slope** b is equal to 1.9 [95% CI (1.08, 2.71)] indicates that a 1-unit increase in BMI is associated with a 1.9-unit average increase in systolic blood pressure. The significance test of the slope shows a highly significant p-value <.0001.

After removing the outlying value, the slope does increase and the statistical significance is stronger, as reflected by the smaller p-value in this model. Recall, the model on the full sample was $\hat{y} = 89.05 + 1.73$ `BMI` .

The **R-squared** is larger at R^2=0.181 in this sensitivity analysis (full sample R^2=0.136), and the **residual standard error** has decreased from $s_{y|x}$=21.11 in the full sample to $s_{y|x}$=19.58 in the sensitivity analysis.

Next, we compute model residuals and fitted values:

```
# Data frame that includes subset of the variables used in today's analysis
dat.sens <- select(sensitivity_fhs100, c(RANDID, SYSBP, BMI))

# Appending column of predicted values (yhat) to dat
dat.sens$predicted <- predict(reg.sens)

# Appending column of residuals to dat
dat.sens$residuals <- resid(reg.sens)

# Summary statistics of residuals
summary(dat.sens$residuals)
```

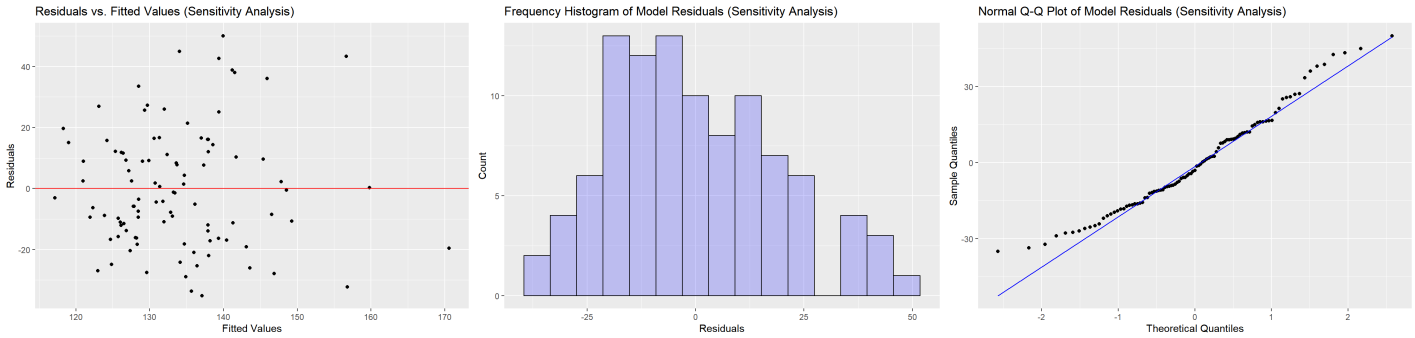
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -35.118 -14.829  -3.138   0.000  11.953  50.058
```

```
# Sort the dat data frame by largest |residuals|
dat.sens <- dat.sens[order(-abs(dat.sens$residuals)),]
head(dat.sens)
```

```
##    RANDID SYSBP   BMI predicted residuals
## 69 162207 190.0 29.64  139.9423  50.05773
## 83 188334 179.0 26.52  134.0279  44.97208
## 90 202101 200.0 38.46  156.6617  43.33831
## 87 199546 182.0 29.35  139.3925  42.60746
## 6   11263 180.0 30.30  141.1934  38.80662
## 49  97895 179.5 30.47  141.5156  37.98436
```

The summary statistics of the residuals above show that the largest residual in this model `reg.sens` is not as extreme as in `reg`.

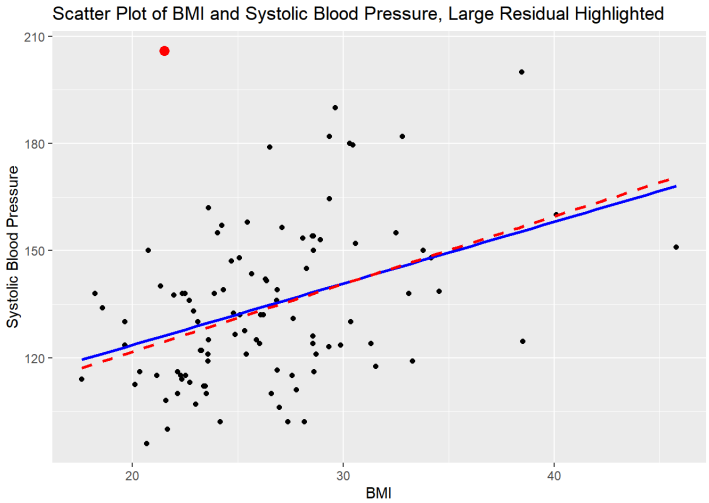
```
# Plot of residuals vs. fitted values
ggplot(data = dat.sens, aes(x = predicted, y = residuals)) +
  geom_point() +
  geom_hline(yintercept = 0, col = "red") +  # add horizontal line at 0
  labs(title = "Residuals vs. Fitted Values (Sensitivity Analysis)",
       x = "Fitted Values", y = "Residuals")

# Histogram of residuals
ggplot(data = dat.sens, aes(x = residuals)) +
  geom_histogram(bins = 15,
                 col = "black",
                 fill = "blue",
                 alpha = 0.2,
                 closed = "left",
                 na.rm = TRUE) +
  labs(title = "Frequency Histogram of Model Residuals (Sensitivity Analysis)",
       x = "Residuals", y = "Count")

# Normal Q-Q plot of residuals
ggplot(dat.sens, aes(sample = residuals)) +
  geom_qq() +
  geom_qq_line(col = "blue") +
  labs(title = "Normal Q-Q Plot of Model Residuals (Sensitivity Analysis)",
       x = "Theoretical Quantiles", y = "Sample Quantiles")
```



The scatter plot of residuals vs. fitted values, the histogram of residuals and the Normal Q-Q plot of residuals are also improved in the sensitivity analysis.

Finally, we can add an additional layer to our scatter plot that includes the sensitivity analysis fitted regression line. The red dashed line below shows a small change in the fitted regression line. The line does appear steeper, as indicated by the larger slope. This provides greater evidence against H_0 that the slope is equal to 0. However, the appearance of the regression line does not change greatly.

```
# Adding new geom_point() layer with highlighted point(s)
ggplot() +
  geom_point(data = fhs100, aes(x = BMI, y = SYSBP)) +         # black points
  geom_smooth(data = fhs100, aes(x = BMI, y = SYSBP),          # blue fitted line
              method = "lm", formula = y ~ x,
              se = FALSE, col = "blue") +
  labs(title = "Scatter Plot of BMI and Systolic Blood Pressure, Large Residual Highlighted",
       x = "BMI", y = "Systolic Blood Pressure") +
  geom_point(data = highlight_fhs100, aes(x = BMI, y = SYSBP),
             col = "red", size = 3) +                           # red highlighted point
  geom_smooth(data = sensitivity_fhs100, aes(x = BMI, y = SYSBP),
              method = "lm", formula = y ~ x,
              se = FALSE, col = "red", linetype = 'dashed')     # sensitivity analysis regression line
```



Ultimately, the final conclusions do not change in the sensitivity analysis: BMI is still an important predictor of systolic blood pressure and there remains a positive association between these two variables. However, you may choose to report the sensitivity analysis in supplementary tables.