Regression_HW3

Warren Geither

10/20/2020

Problem 2

a.)

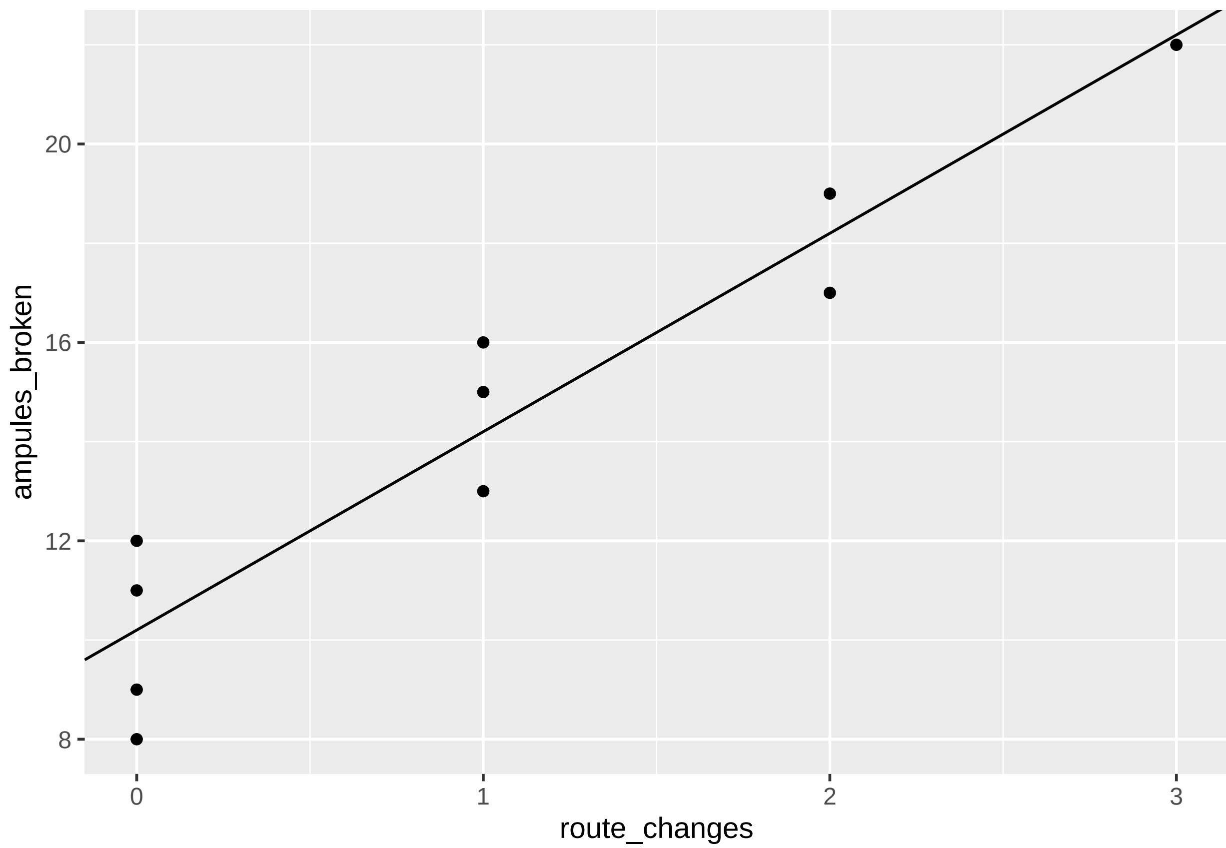
```r
# create dataframe
freight_data_df <- data.frame(route_changes = c(1,0,2,0,3,1,0,1,2,0)
                              , ampules_broken = c(16,9,17,12,22,13,8,15,19,11))

# get values needed to calculate
x <- freight_data_df$route_changes
y <- freight_data_df$ampules_broken
x_bar <- mean(x)
ybar <- mean(y)
n <- length(x)

# get beta hats
beta1hat = sum((x-x_bar)*(y-ybar))/sum((x-x_bar)^2)
beta0hat = ybar-beta1hat*x_bar

# print estimates
print(paste0("betahat0: ", beta0hat))
```

```
## [1] "betahat0: 10.2"
```

```r
print(paste0("betahat1: ", beta1hat))
```

```
## [1] "betahat1: 4"
```

```r
# plot
ggplot(freight_data_df,aes(route_changes, ampules_broken))+
  geom_point()+
  geom_abline(slope=beta1hat, intercept=beta0hat)
```



b.)

```
# sigma hat^2 = Y^T(I-H)Y/n-2
y_T <- t(y)
I <- diag(10)
X <- cbind(c(rep(1,10)), x)
X_T <- t(X)
H <- X%*%solve(X_T%*%X)%*%X_T

# sig hat
sigma_hat_squared <- ((y_T%*%(I - H)%*%y)/(n-2))

# sxx
sxx <- sum((x-x_bar)^2)

# t-value
crit_val <- qt(0.05/2, 8, lower.tail = FALSE)

# beta0hat C.I
beta0_upper_bound <- beta0hat + crit_val*sqrt(sigma_hat_squared*((1/n)+((x_bar^2)/sxx)))
beta0_lower_bound <- beta0hat - crit_val*sqrt(sigma_hat_squared*((1/n)+((x_bar^2)/sxx)))

# format for print
b0_ci1 <- paste0(round(beta0_lower_bound,3), ",")
b0_ci2 <- paste0(b0_ci1, round(beta0_upper_bound, 3))
b0_ci3 <- paste0("(",b0_ci2)
```

```r
b0_ci4 <- paste0(b0_ci3, ")")

# beta1hat C.I
beta1_upper_bound <- beta1hat + crit_val*sqrt(sigma_hat_squared/sxx)
beta1_lower_bound <- beta1hat - crit_val*sqrt(sigma_hat_squared/sxx)

# format for print
b1_ci1 <- paste0(round(beta1_lower_bound,3), ",")
b1_ci2 <- paste0(b1_ci1, round(beta1_upper_bound, 3))
b1_ci3 <- paste0("(",b1_ci2)
b1_ci4 <- paste0(b1_ci3, ")")

# print C.I.
print(paste0("95% C.I. for beta0hat: ", b0_ci4))
```

```
## [1] "95% C.I. for beta0hat: (8.67,11.73)"
```

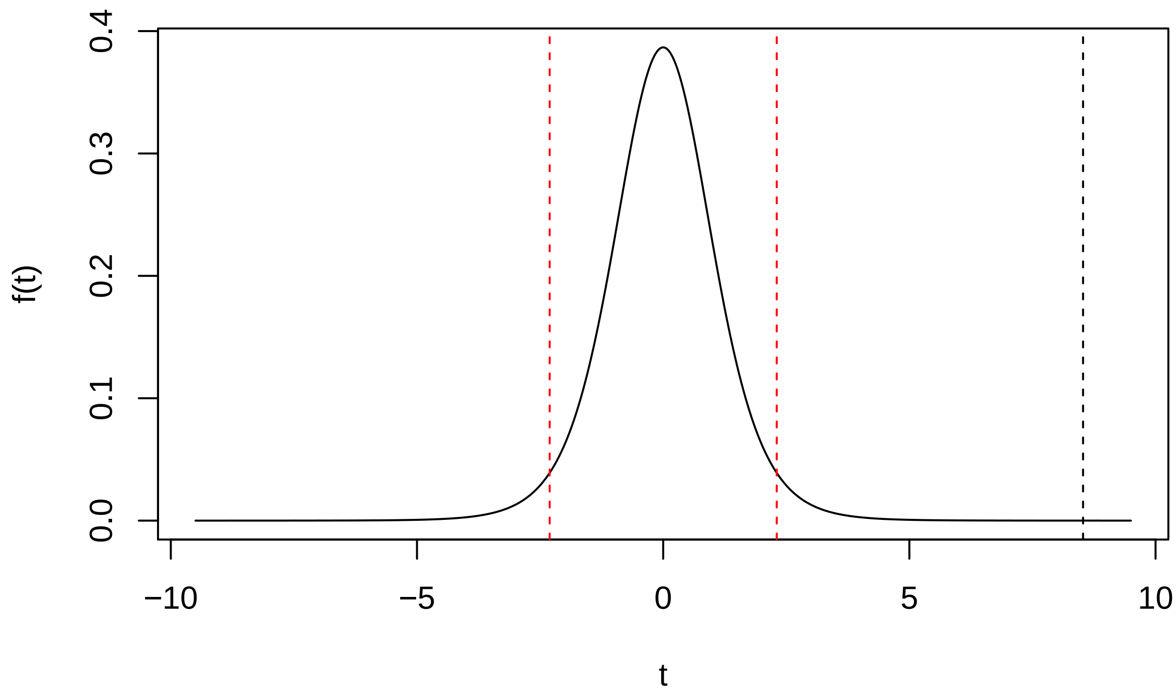
```r
print(paste0("95% C.I. for beta1hat: ", b1_ci4))
```

```
## [1] "95% C.I. for beta1hat: (2.918,5.082)"
```

c.)

$$\text{Hypothesis for linear relationship}$$
$$Ho : \beta_1 = 0$$
$$Ha : \beta_1 \neq 0$$

```r
test_stat <- (beta1hat - 0)/sqrt(sigma_hat_squared/sxx)

# For the plot for t test
dum=seq(-9.5, 9.5, length=10^4)

plot(dum, dt(dum, df=(n-2)), type='l', xlab='t', ylab='f(t)')
abline(v=test_stat, lty=2)
abline(v=crit_val, col='red', lty=2)
abline(v=-crit_val, col='red', lty=2)
```



Because our test statistic of 8.5 is greater than our postive critical value of 2.3, we can reject the null hypothesis at the alpha=0.05 level and say that there is a linear relationship between broken amplues and ship route changes

Problem 3

```
# set seed
set.seed(201547)

# intialize list of estimates
betahat1_vector <- c(rep(NA,100))

# create a loop to bootstrap betahat1
for (i in 1:100){
  # Sample 10 rows from data
  sample_row_nums <- sample(nrow(freight_data_df), n, replace = T)

  # Store sample in dataframe
  sample_rows <- freight_data_df[sample_row_nums, ]

  # get values for betahat1 calculation
  x <- sample_rows$route_changes
  y <- sample_rows$ampules_broken
  x_bar <- mean(x)
  ybar <- mean(y)
```

```r
  n <- length(x)

  # calc betahat1
  beta1hat = sum((x-x_bar)*(y-ybar))/sum((x-x_bar)^2)

  # store in list
  betahat1_vector[i] <- beta1hat
}

# remove na values
betahat1_vector <- betahat1_vector[!is.na(betahat1_vector)]

# calculate confidence interval
quantile(x = betahat1_vector, probs = c(0.025,0.975))
```

```
##     2.5%    97.5%
## 3.256755 5.866964
```

The intervals are similar since both account for the variance in beta1hat. However the first confidence interval multiples by the critical value for the t-distribution at the alpha level we are using. The bootstrap only picks up the natural variance of the mean of beta1hat from the bootstrapping.

Problem 1 & 4

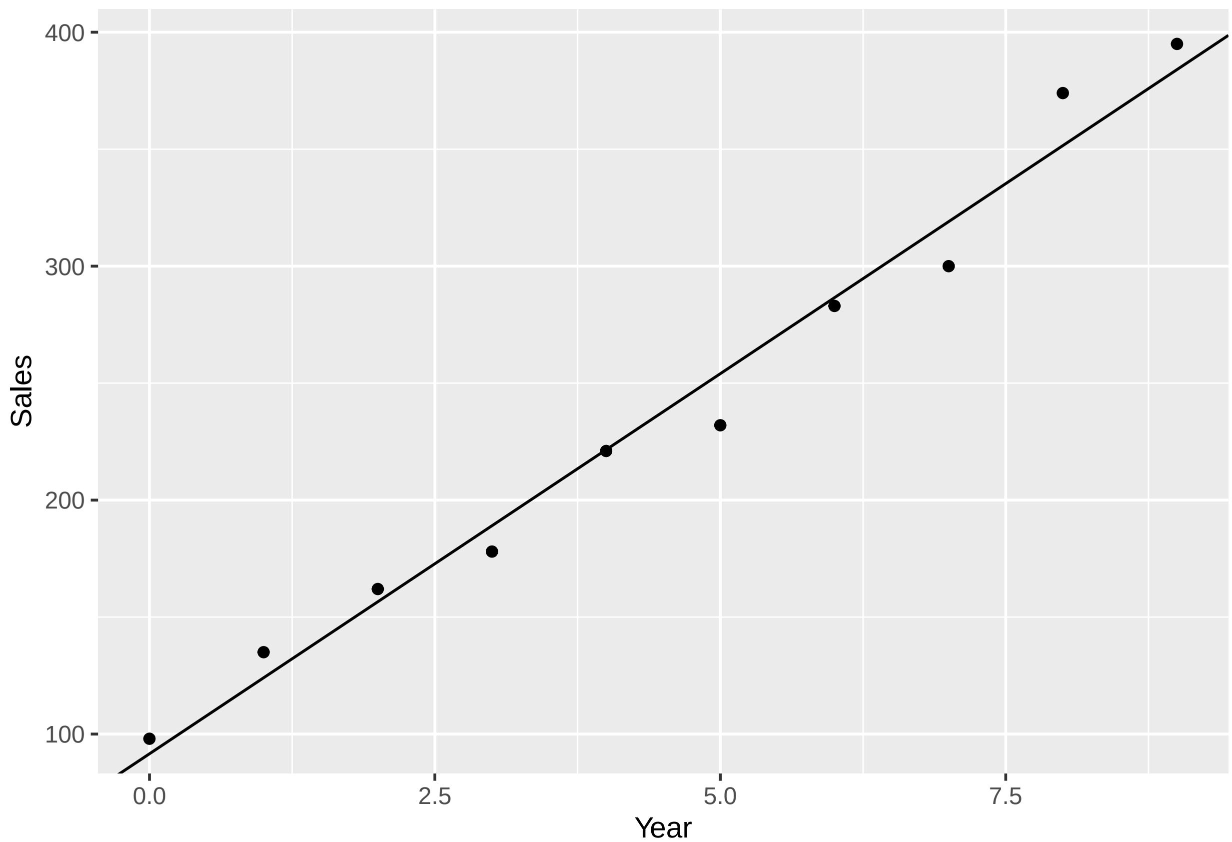
OLS Model

```r
# create dataframe
sales_df <- data.frame(year = c(seq(0,9,by=1))
                       , sales = c(98,135,162,178,221,232,283,300,374,395))

# fit model
lmfit <- lm(sales ~ year, sales_df)

# summary of model
summary(lmfit)
```

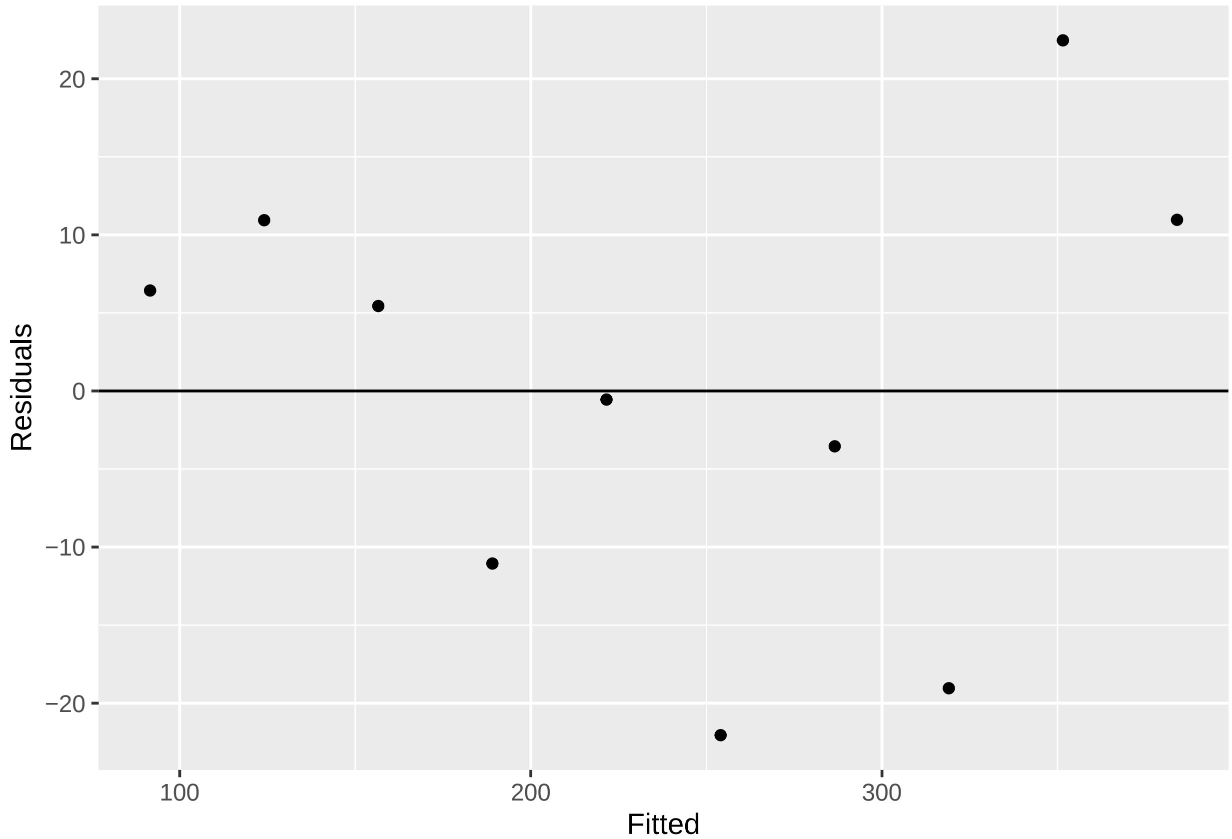
```
##
## Call:
## lm(formula = sales ~ year, data = sales_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.049  -9.177   2.446   9.814  22.461
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   91.564      8.814   10.39 6.38e-06 ***
## year          32.497      1.651   19.68 4.62e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 15 on 8 degrees of freedom
## Multiple R-squared:  0.9798, Adjusted R-squared:  0.9772
## F-statistic: 387.4 on 1 and 8 DF,  p-value: 4.62e-08
```

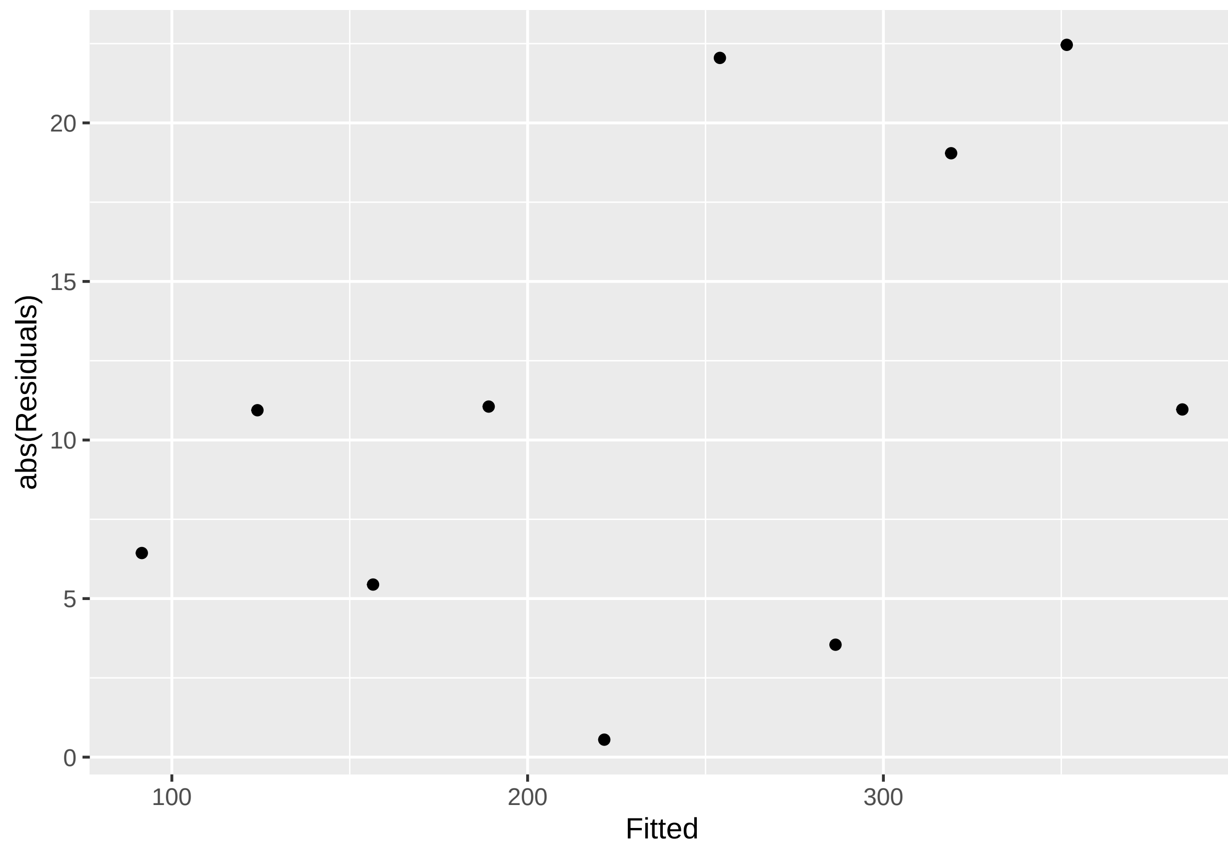
```r
# plot line
ggplot(sales_df, aes(x= year, y=sales)) +
  geom_point() +
  geom_abline(slope= lmfit$coef[2], intercept=lmfit$coef[1], color = "black") +
  xlab("Year") +
  ylab("Sales")
```




```r
# residuals
ggplot(sales_df, aes(x= fitted(lmfit), y=residuals(lmfit))) +
  geom_point() +
  geom_hline(yintercept=0) +
  xlab("Fitted") +
  ylab("Residuals")
```

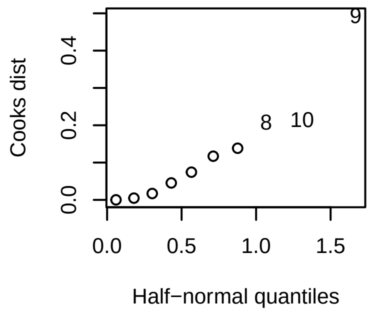


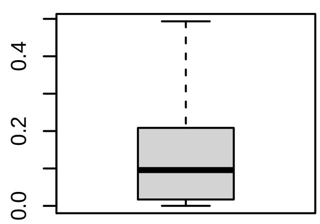

```r
# abs residuals
ggplot(sales_df, aes(x= fitted(lmfit), y=abs(residuals(lmfit)))) +
  geom_point() +
  xlab("Fitted") +
  ylab("abs(Residuals)")
```




```
# cooks distance
par(mfcol=c(2,3))
cook<-cooks.distance(lmfit)
halfnorm(cook,3,ylab="Cooks dist")
boxplot(cook)

# summary stats
knitr::kable(summary(sales_df))
```

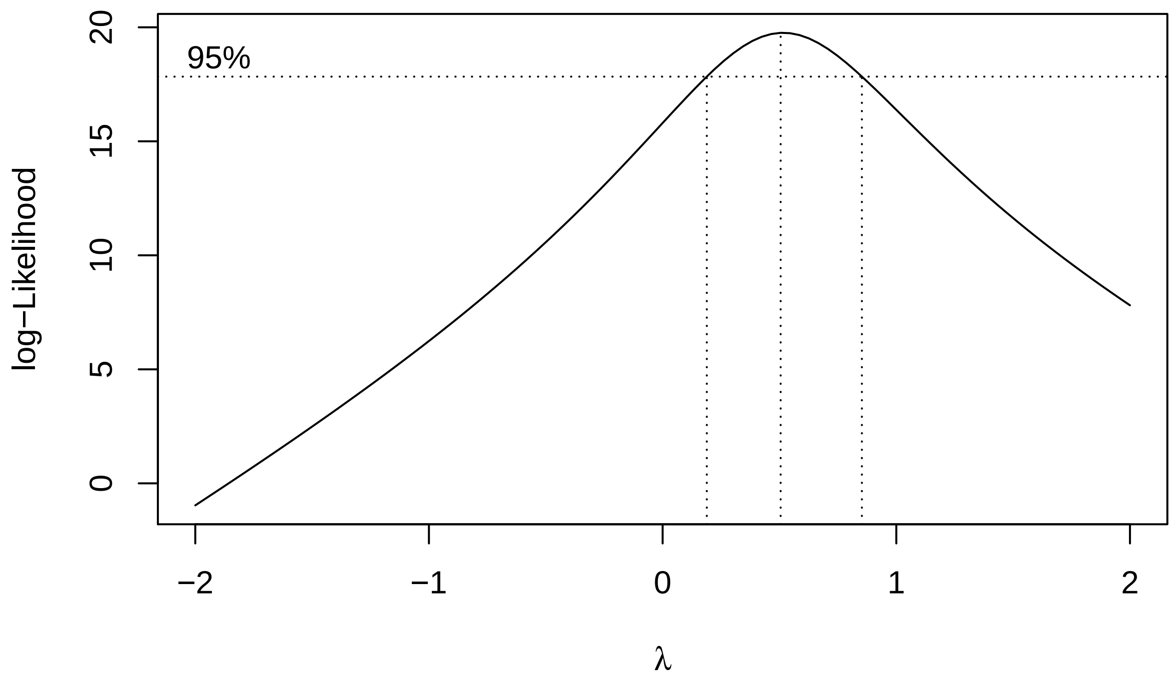
year	sales
Min. :0.00	Min. : 98.0
1st Qu.:2.25	1st Qu.:166.0
Median :4.50	Median :226.5
Mean :4.50	Mean :237.8
3rd Qu.:6.75	3rd Qu.:295.8
Max. :9.00	Max. :395.0





Box-Cox Transformation

```
# plot of picking best lambda
boxcox(lmfit,plotit=T)

# storing transform
bc_transform <- boxcox(lmfit)
```

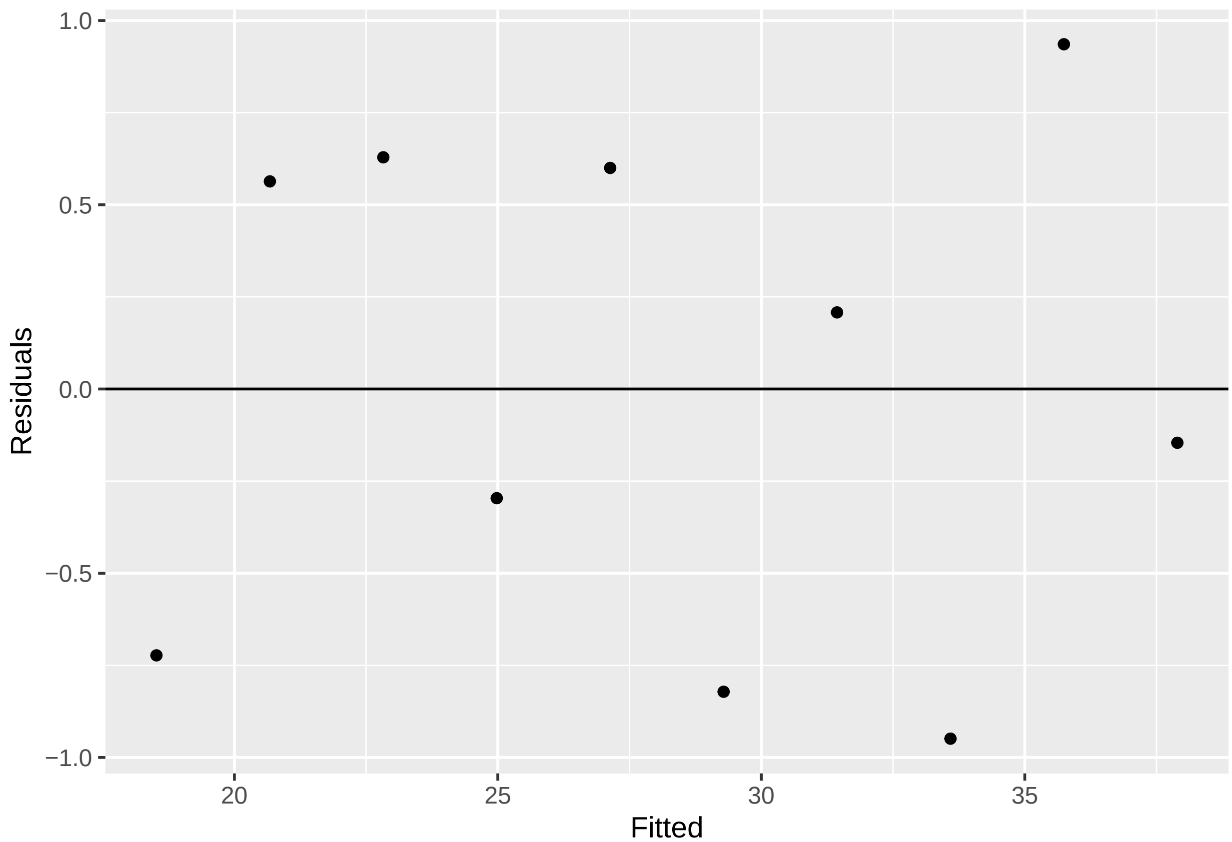



```r
# picking lambda that maximimizs the log liklihood
best_lambda <- bc_transform$x[which(bc_transform$y==max(bc_transform$y))]

# fitting new model
bcfit <- lm(((sales^0.5) - 1)/0.5~year, data = sales_df)

# summary of model
summary(bcfit)
```

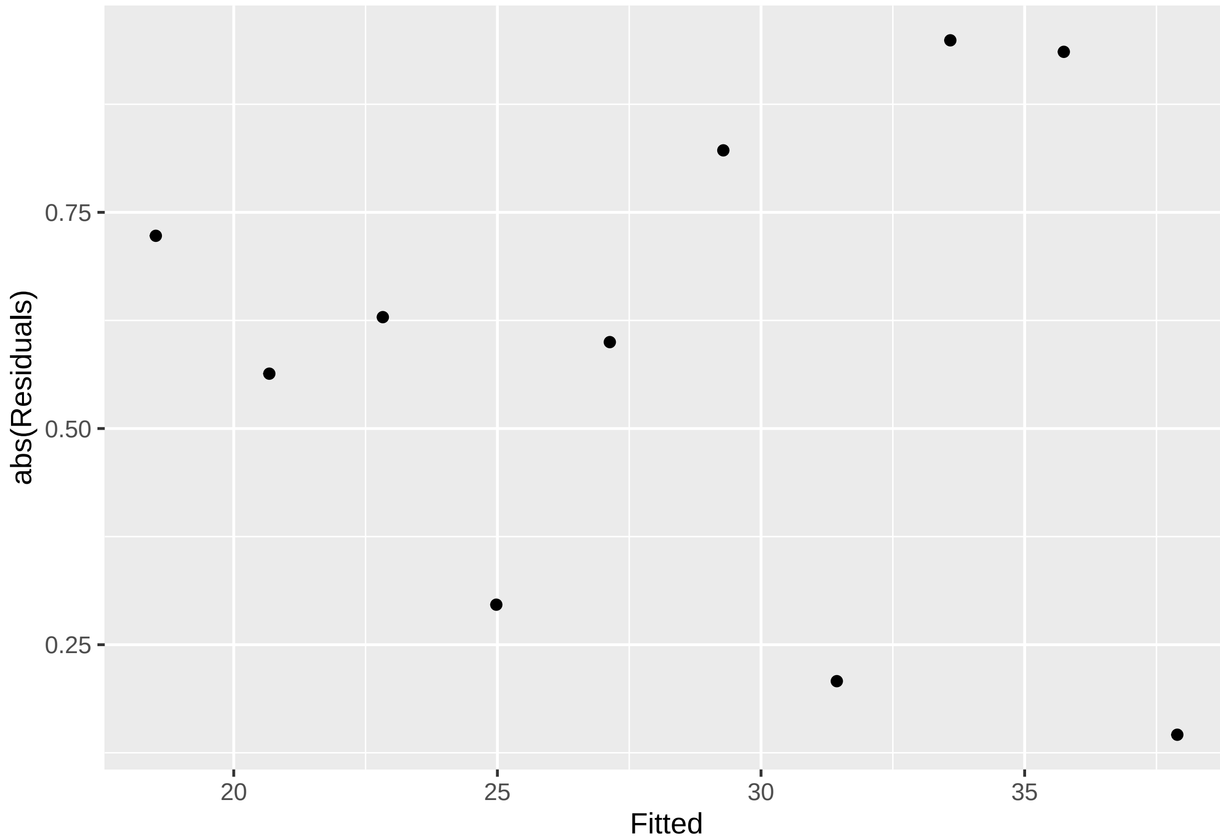
```
##
## Call:
## lm(formula = ((sales^0.5) - 1)/0.5 ~ year, data = sales_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.94893 -0.61623  0.03097  0.59082  0.93563
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 18.52186    0.42580   43.50 8.61e-11 ***
## year         2.15258    0.07976   26.99 3.83e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7245 on 8 degrees of freedom
## Multiple R-squared:  0.9891, Adjusted R-squared:  0.9878
```

```
## F-statistic: 728.4 on 1 and 8 DF,  p-value: 3.826e-09
```

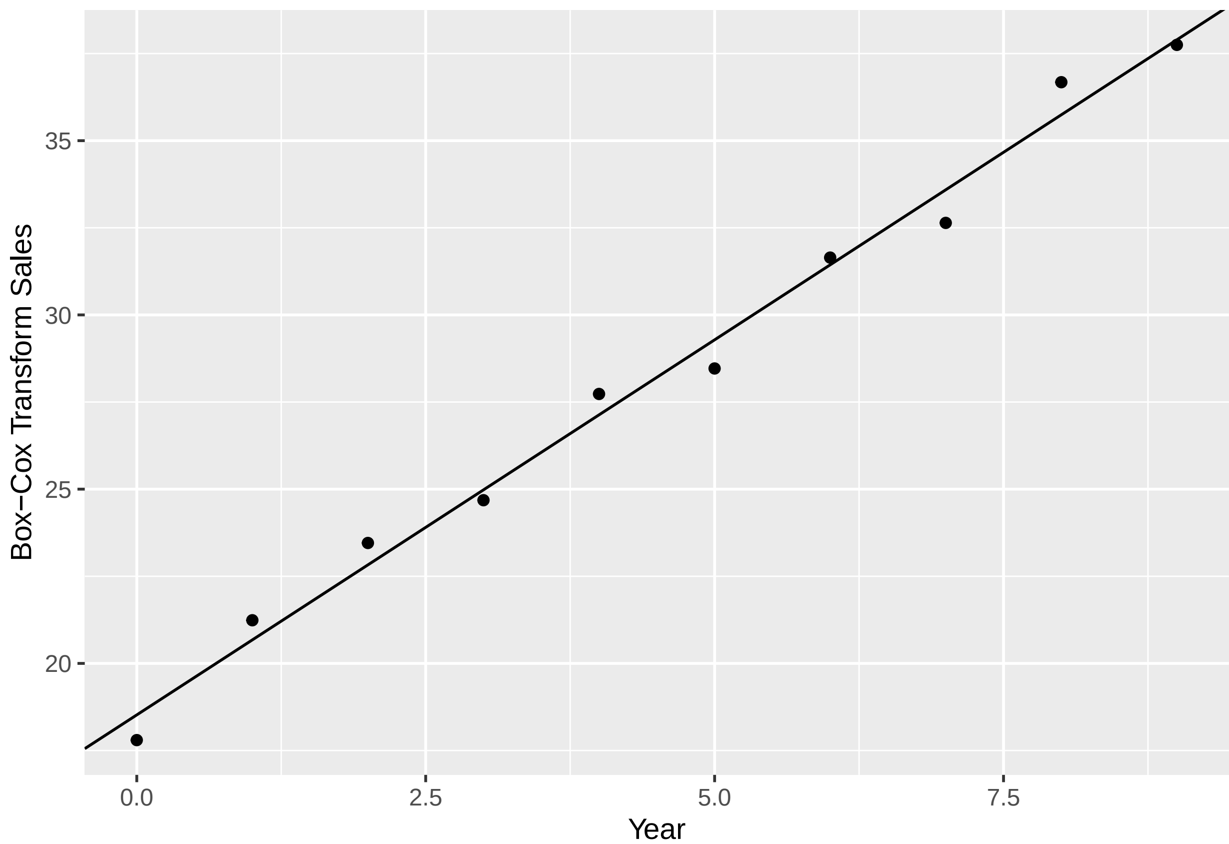
```r
# residuals
ggplot(sales_df, aes(x= fitted(bcfit), y=residuals(bcfit))) +
  geom_point() +
  geom_hline(yintercept=0) +
  xlab("Fitted") +
  ylab("Residuals")
```




```r
# abs residuals
ggplot(sales_df, aes(x= fitted(bcfit), y=abs(residuals(bcfit)))) +
  geom_point() +
  xlab("Fitted") +
  ylab("abs(Residuals)")
```




```
# regression line
ggplot(sales_df, aes(x= year, y=((sales^0.5) - 1)/0.5)) +
  geom_point() +
  geom_abline(slope= bcfit$coef[2], intercept=bcfit$coef[1], color = "black") +
  xlab("Year") +
  ylab("Box-Cox Transform Sales")
```



WLS Transformation

```
# store residuals
resid<-residuals(lmfit)

# absolute value of residuals
absresid<-abs(resid)

# bring together in dataframe
res_data <- as.data.frame(cbind(absresid, year = sales_df$year))

# find linear relationshp between abs(res) vs X
lmfitw0 <- lm(absresid~., data = res_data)

# weight is proportion to inverse of variance
w <- 1/(fitted(lmfitw0))^2

# fit WLS
wfit <- lm(sales~year, data = sales_df, weights = w)

# summary of model
summary(wfit)
```

```
##
```

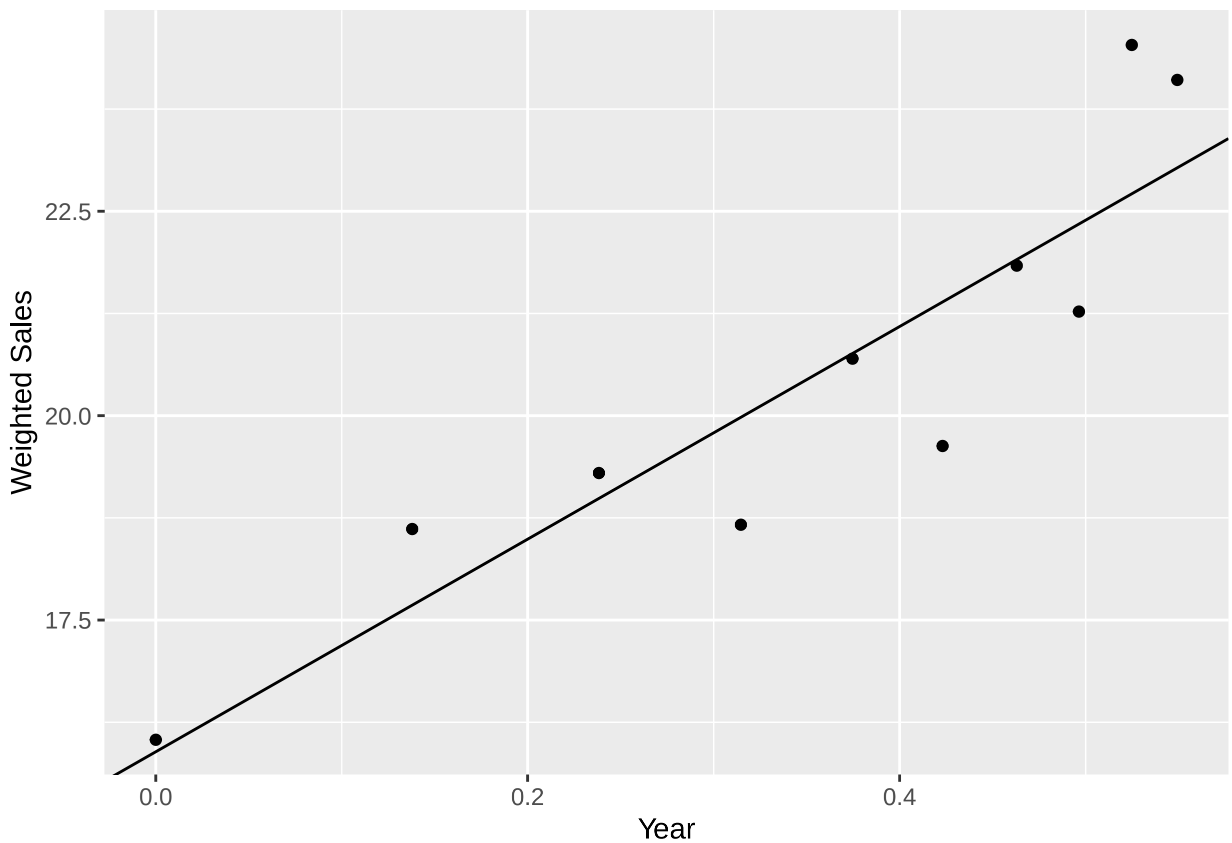
```
## Call:
## lm(formula = sales ~ year, data = sales_df, weights = w)
##
## Weighted Residuals:
##      Min       1Q   Median       3Q      Max
## -1.76138 -0.82180  0.04168  0.77607  1.82059
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   97.104      5.297   18.33 8.07e-08 ***
## year          31.143      1.393   22.36 1.69e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.196 on 8 degrees of freedom
## Multiple R-squared:  0.9842, Adjusted R-squared:  0.9823
## F-statistic: 499.9 on 1 and 8 DF,  p-value: 1.694e-08
```

```r
# x and y values
y <- sales_df$sales
x <- sales_df$year

# create weighted values
yw <- w^0.5*y
xw <- w^0.5*x

# fit the model a different way
wlm <-lm(yw~xw)

# sumary of wlm
summary(wlm)
```

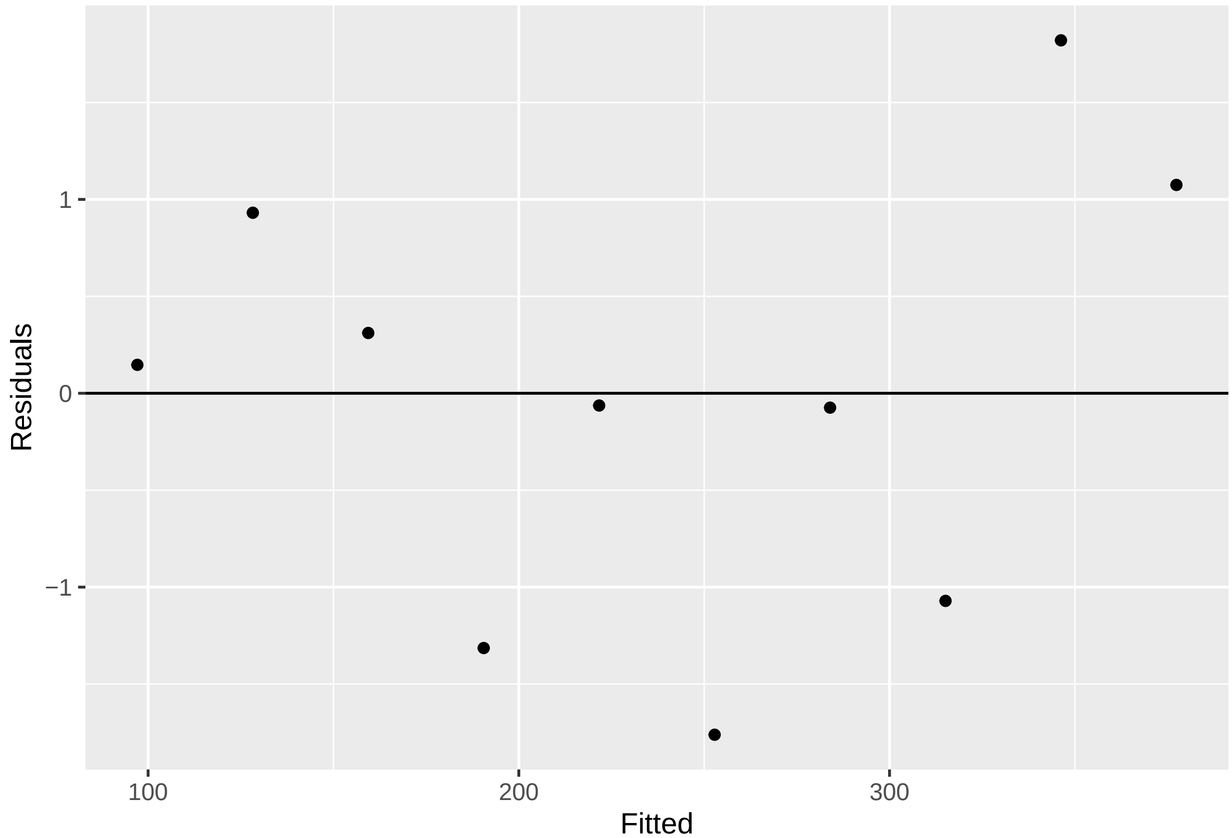
```
##
## Call:
## lm(formula = yw ~ xw)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.76138 -0.82180  0.04168  0.77607  1.82059
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  15.8893     0.8668  18.331 8.07e-08 ***
## xw           13.0037     2.2147   5.871 0.000374 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.196 on 8 degrees of freedom
## Multiple R-squared:  0.8117, Adjusted R-squared:  0.7881
## F-statistic: 34.47 on 1 and 8 DF,  p-value: 0.0003736
```

```r
# create a weighted dataframe
weighted_df <- as.data.frame(cbind(xw,yw))
```

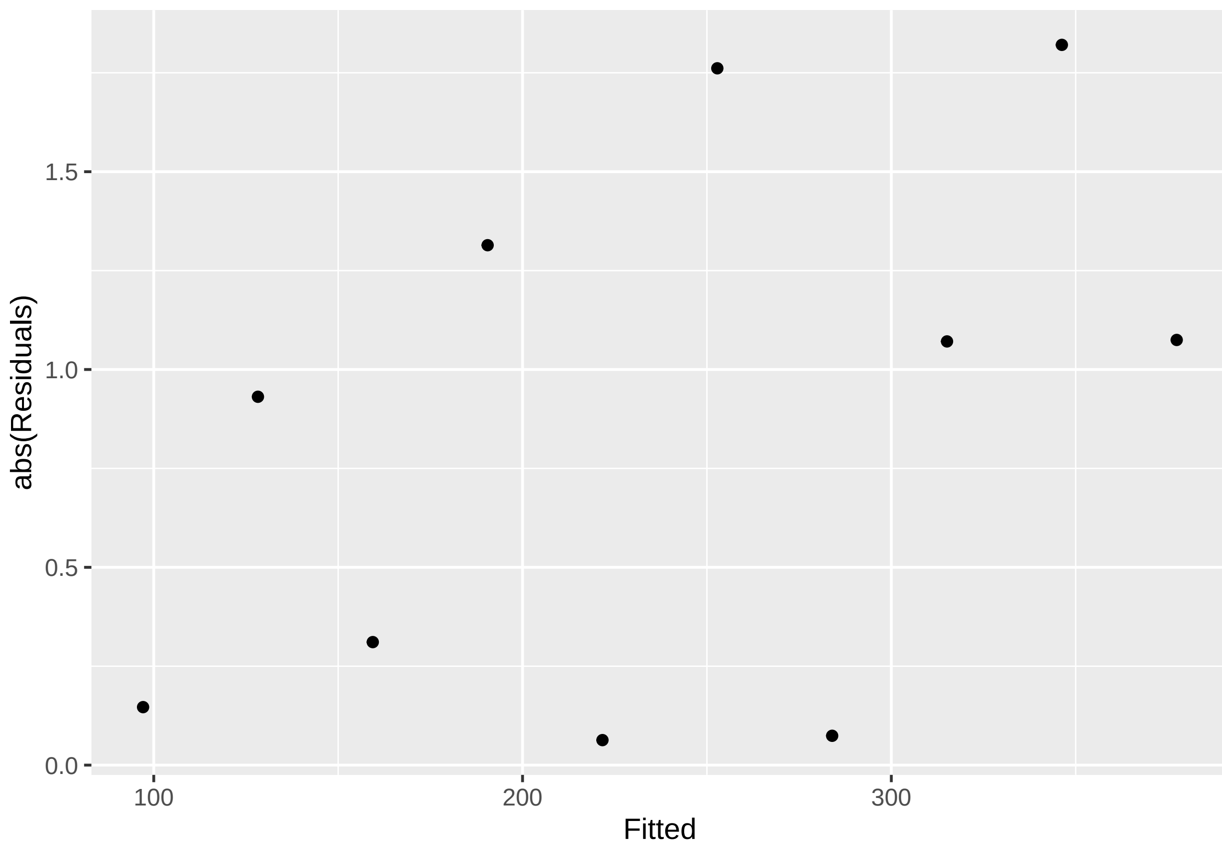
```r
# created a weighted plot
ggplot(weighted_df, aes(x= xw, y=yw)) +
  geom_point() +
  geom_abline(slope= wlm$coef[2], intercept=wlm$coef[1], color = "black") +
  xlab("Year") +
  ylab("Weighted Sales")
```




```r
# weighted residuals
resid <- residuals(wfit)
wresid <- w^0.5*resid

# store weighted resdduals in data frame
new_data <- as.data.frame(cbind(fitted = fitted(wfit), res = wresid))

# plot residuals
ggplot(new_data, aes(x = fitted, y = res)) +
  geom_point() +
  geom_hline(yintercept=0) +
  xlab("Fitted") +
  ylab("Residuals")
```




```r
# abs residuals
ggplot(new_data, aes(x = fitted, y = abs(res))) +
  geom_point() +
  xlab("Fitted") +
  ylab("abs(Residuals)")
```



Problem 4

Randomness Check

$$\text{Hypothesis for Runs Test}$$
$$Ho: \text{Random}$$
$$Ha: \text{Not Random}$$

```
# get residuals
lmfit$residuals
bcfit$residuals
wlm$residuals

# Runs Test
runs.test(residuals(lmfit))
runs.test(residuals(bcfit))
runs.test(residuals(wlm))
```

Runs Test Manual Runs Test:

Ho: residuals are random

From TableA30: rL = 2, rU=10, fail to reject when 2 < r < 10 lmfit: 2 postive run & 1 negative run ==> r = 3 ==> fail to reject null bcfit: 4 positive runs & 5 negative runs ==> r=9 ==> fail to reject null wls: 2 postive run & 1 negative run ==> r = 3 ==> fail to reject null

Runs.test():

lmfit: p-val = 0.04417 ==> Reject Ho bcfit: p-val = 0.04417 ==> Reject Ho wlm: p-val = 0.04417 ==> Reject Ho

Durbin Watson Test

```
# mnaual dw test
lm_res_squared <- sum(lmfit$residuals^2)
bc_res_squared <- sum(bcfit$residuals^2)
wlm_res_squared <- sum(wlm$residuals^2)

# calc numerator for dw test
for (i in 2:10){
  lm_lag_sum <- sum((lmfit$residuals[i]-lmfit$residuals[i-1])^2)
  bc_lag_sum <- sum((bcfit$residuals[i]-bcfit$residuals[i-1])^2)
  wlm_lag_sum <- sum((wlm$residuals[i]-wlm$residuals[i-1])^2)
}

# calculate d
lmfit_D <- lm_lag_sum / lm_res_squared
bcfit_D <- bc_lag_sum / bc_res_squared
wlmfit_D <- wlm_lag_sum / wlm_res_squared

# Durbin-Watson Test
dwtest(lmfit)
dwtest(bcfit)
dwtest(wlm)
```

Manual Durbin Watson Test:

dL = 0.604 dU = 1.001

lmfit: D = 0.0734 bcfit: D = 0.2785 wlm: D = 0.04862

Durbin-watson Test: Ho: autocorrelation = 0 (resdiduals are independent)

lmfit: p-val = 0.2503 ==> Fail to reject bcfit: p-val = 0.9067 ==> Fail to reject wlm: p-val = 0.2381 ==> Fail to reject

Constant Variance Check

$$\text{Hypothesis for BF Test}$$
$$Ho: \text{Constant Variance}$$
$$Ha: \text{Not Constant Variance}$$

BF Test

```r
# set seed
set.seed(346565)

# initialize empty vector for test stats
p_val_matrix <- c(rep(NA, 100))

# loop for different groups (replace lmfit with other model for now)
for(i in 1:100){
  # Store sample in dataframe
  group1 <- sample(lmfit$residuals, 5, replace = FALSE)
  group2 <- lmfit$residuals[!lmfit$residuals %in% group1]

  # calculate n for the 2 groups
  n1 <- length(group1)
  n2 <- length(group2)

  # calculate medians
  median_res1 <- median(group1)
  median_res2 <- median(group2)

  # caluate deviation of residual from median
  d1_vals <- abs(group1 - median_res1)
  d2_vals <- abs(group2 - median_res2)

  s1 <- sd(d1_vals)
  s2 <- sd(d2_vals)

  # mean deviation
  d1_mean <- mean(d1_vals)
  d2_mean <- mean(d2_vals)

  # test statistic
  t_stat <- (d1_mean-d2_mean)/sqrt(((s1^2)/n1)+((s2^2)/n2))

  # simplifies df equation below
  A <- s1^2/n1
  B <- s2^2/n2

  # degrees of freedom for Welch 2-sample t-test
  # https://mse.redwoods.edu/darnold/math15/spring2013/R/Activities/WelchTTest.html
  df <- (A+B)^2/(A^2/(n1-1)+B^2/(n2-1))

  # p-value
  p = 2*pt(t_stat,df)

  # store result
  p_val_matrix[i] <- p
}
```

lmfit: Fail to Reject 100/100 times bcfit: Fail to Reject 100/100 times wlm: Fail to Reject 100/100 times

BP Test

```r
# conduct bp test
bptest(lmfit)
bptest(bcfit)
bptest(wlm)
```

lmfit: p-val = 0.1216 ==> Fail to reject bcfit: p-val = 0.6753 ==> Fail to reject wlm: p-val = 0.1502 ==> Fail to reject

Normality Test

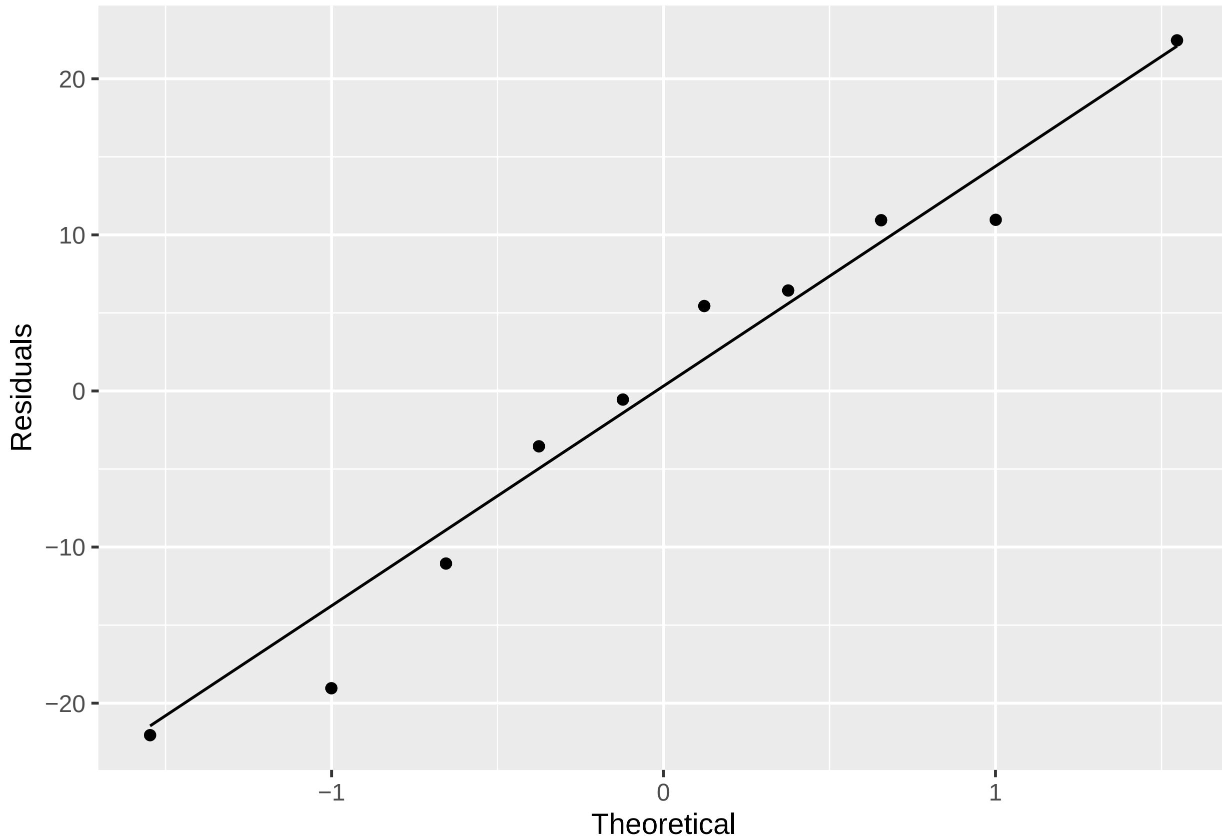
Anderson Darling Test

```r
# Shapiro-Wilks Test
shapiro.test(residuals(lmfit))

# Anderson Darling Test
ad.test(residuals(lmfit))
ad.test(residuals(bcfit))
ad.test(residuals(wlm))
```

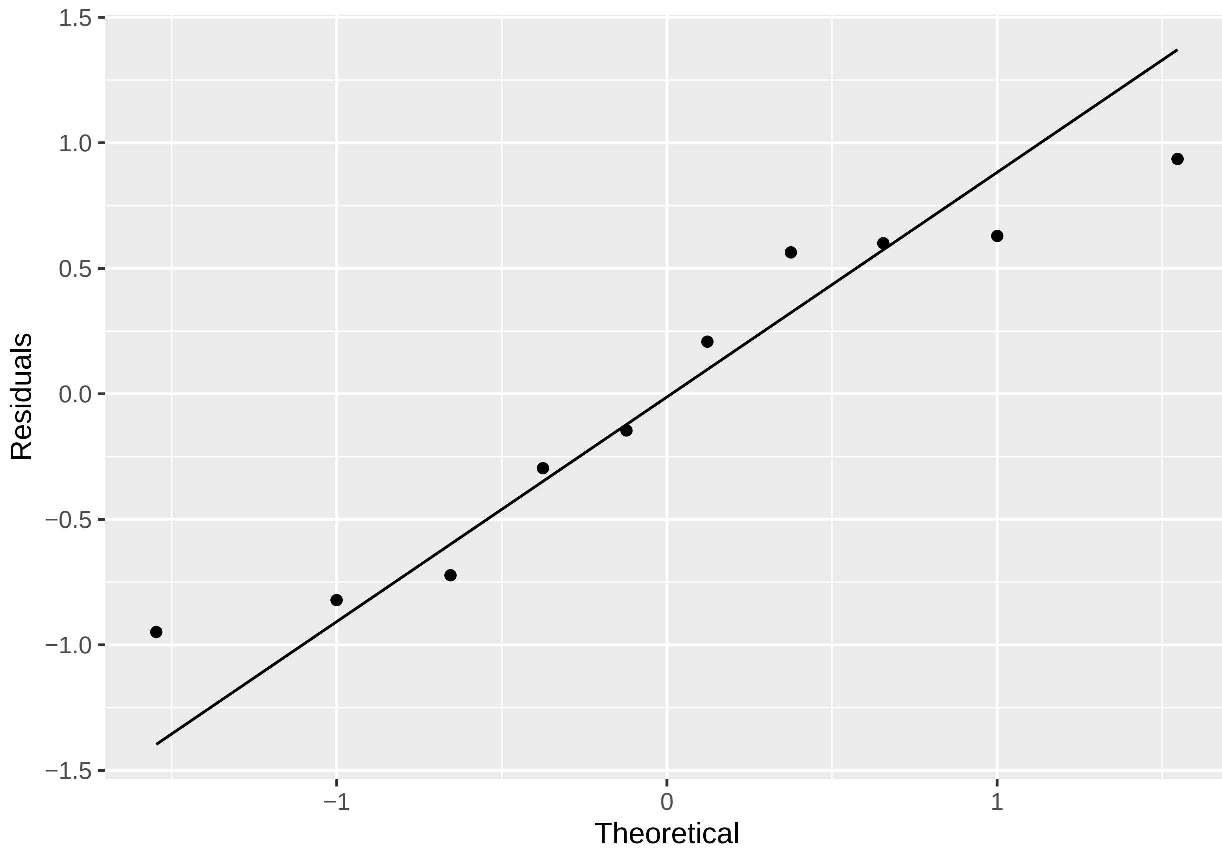
lmfit: p-val = 0.7703 ==> Fail to Reject bcfit: p-val = 0.3809 ==> Fail to Reject wlm: p-val = 0.7958 ==> Fail to Reject

QQ-Plots

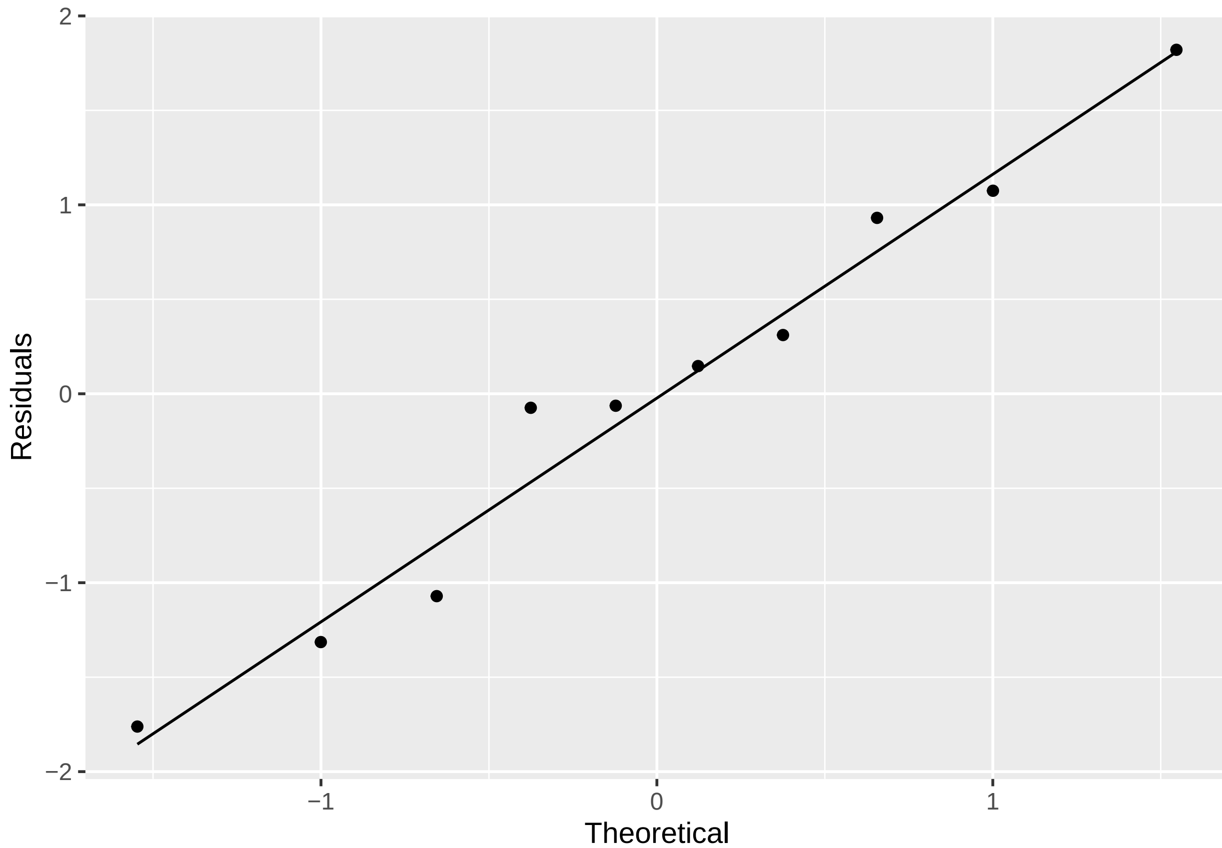
```r
# QQ for lmfit
ggplot(lmfit, aes(sample=residuals(lmfit)))+
  stat_qq() +
  stat_qq_line() +
  xlab("Theoretical") +
  ylab("Residuals")
```




```r
# QQ for bcfit
ggplot(lmfit, aes(sample=residuals(bcfit)))+
  stat_qq() +
  stat_qq_line() +
  xlab("Theoretical") +
  ylab("Residuals")
```




```
# QQ for wls
ggplot(wlm, aes(sample=residuals(wlm)))+
  stat_qq() +
  stat_qq_line() +
  xlab("Theoretical") +
  ylab("Residuals")
```


Residuals
Theoretical
2
1
0
−1
−2
−1
0
1