Math 158
Jo Hardin
R code

1 Simple Linear Regression

Consider a dataset from ISLR on credit scores. Because we don't know the sampling mechanism used to collect the data, we are unable to generalize the model results to a larger population. However, we can look at the relationship between the variables and build a linear model.[1]

```
library(ISLR)
# Using the Credit data, select only a few of the variables
Credit <- Credit %>%
  select(Balance, Limit, Income, Rating, Age, Education)

glimpse(Credit)
```

```
## Observations: 400
## Variables: 6
## $ Balance   <int> 333, 903, 580, 964, 331, 1151, 203, 872, 279, 1350, ...
## $ Limit     <int> 3606, 6645, 7075, 9504, 4897, 8047, 3388, 7114, 3300...
## $ Income    <dbl> 14.9, 106.0, 104.6, 148.9, 55.9, 80.2, 21.0, 71.4, 1...
## $ Rating    <int> 283, 483, 514, 681, 357, 569, 259, 512, 266, 491, 58...
## $ Age       <int> 34, 82, 71, 36, 68, 77, 37, 87, 66, 41, 30, 64, 57, ...
## $ Education <int> 11, 15, 11, 11, 16, 10, 12, 9, 13, 19, 14, 16, 7, 9,...
```

```
skimr::skim(Credit)
```

```
## Skim summary statistics
##  n obs: 400
##  n variables: 6
##
## Variable type: integer
##    variable missing complete   n    mean      sd min     p25 median
## 1       Age       0      400 400   55.67   17.25  23   41.75   56
## 2   Balance       0      400 400  520.01  459.76   0   68.75  459.5
## 3 Education       0      400 400   13.45    3.13   5   11      14
## 4     Limit       0      400 400 4735.6  2308.2  855 3088    4622.5
## 5    Rating       0      400 400  354.94  154.72  93  247.25  344
##       p75   max     hist
## 1   70       98
## 2  863     1999
## 3   16       20
## 4 5872.75 13913
## 5  437.25   982
##
## Variable type: numeric
##   variable missing complete   n  mean    sd   min   p25 median   p75
## 1   Income       0      400 400 45.22 35.24 10.35 21.01  33.12 57.47
##      max     hist
## 1 186.63
```

[1] Much of this handout is taken from https://ismayc.github.io/moderndiver-book/6-regression.html#model3.

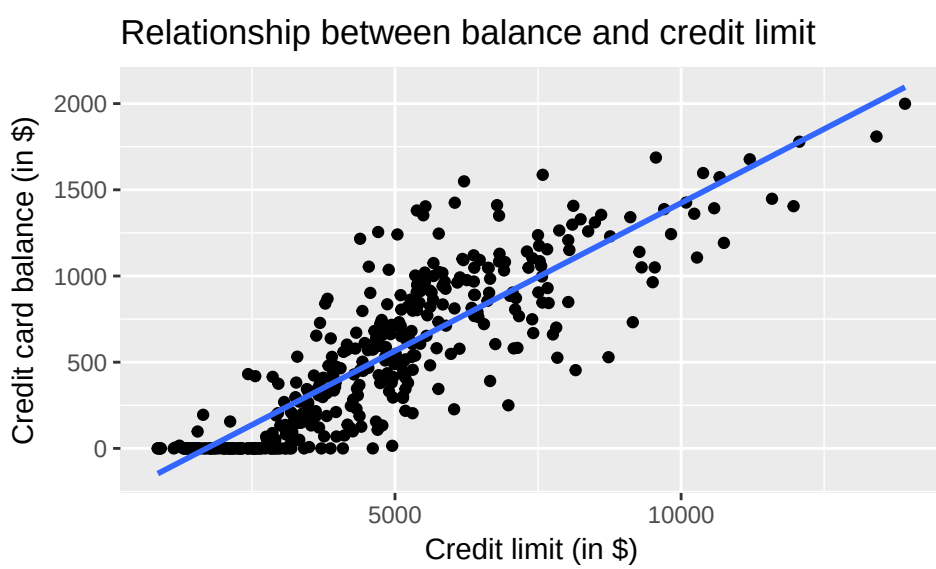
```r
cor(Credit)

##            Balance   Limit  Income  Rating     Age Education
## Balance    1.00000  0.8617  0.4637  0.8636 0.00184  -0.00806
## Limit      0.86170  1.0000  0.7921  0.9969 0.10089  -0.02355
## Income     0.46366  0.7921  1.0000  0.7914 0.17534  -0.02769
## Rating     0.86363  0.9969  0.7914  1.0000 0.10316  -0.03014
## Age        0.00184  0.1009  0.1753  0.1032 1.00000   0.00362
## Education -0.00806 -0.0235 -0.0277 -0.0301 0.00362   1.00000
```

1.1 Looking at the Data

It is *always* a good idea to look at the data. Graphical representations are typically the key to a good analysis. Here we use the `ggplot2` package for creating boxplots and scatterplots.

It seems as though there is a linear relationship between `Limit` and `Balance`, but that the prediction is certainly not perfect (and what about all the zeros???).

```r
ggplot(Credit, aes(x = Limit, y = Balance)) +
  geom_point() +
  labs(x = "Credit limit (in $)", y = "Credit card balance (in $)",
       title = "Relationship between balance and credit limit") +
  geom_smooth(method = "lm", se = FALSE)
```



1.2 Fitting a Linear Model

Note that the formula for running a linear model in R is `response ~ explanatory`. The `broom` package has three important functions:

- `glance` reports the information which is based on the overall model
- `tidy` reports the information which is based on each explanatory variable
- `augment` reports the information which is based on each observation

```
credit_lm <- lm(Balance ~ Limit, data=Credit)
broom::glance(credit_lm)

##   r.squared adj.r.squared sigma statistic   p.value df logLik  AIC  BIC
## 1     0.743         0.742   234      1148 2.53e-119  2  -2748 5502 5514
##   deviance df.residual
## 1 21715657         398

broom::tidy(credit_lm)

##          term estimate std.error statistic   p.value
## 1 (Intercept) -292.790  26.68341     -11.0  1.18e-24
## 2       Limit    0.172   0.00507      33.9 2.53e-119

broom::augment(credit_lm) %>% head()

##   Balance Limit .fitted .se.fit  .resid    .hat .sigma  .cooksd .std.resid
## 1     333  3606     326    13.0    6.87 0.00310    234 1.35e-06     0.0294
## 2     903  6645     848    15.2   55.26 0.00422    234 1.19e-04     0.2371
## 3     580  7075     922    16.6 -341.54 0.00507    233 5.48e-03    -1.4659
## 4     964  9504    1338    26.8 -374.45 0.01320    233 1.74e-02    -1.6137
## 5     331  4897     548    11.7 -216.72 0.00251    234 1.09e-03    -0.9290
## 6    1151  8047    1088    20.4   62.63 0.00766    234 2.80e-04     0.2691
```

Also available through `broom` are the CIs for the coefficients.

```
broom::tidy(credit_lm, conf.int = TRUE, conf.level = 0.90)

##          term estimate std.error statistic   p.value conf.low conf.high
## 1 (Intercept) -292.790  26.68341     -11.0  1.18e-24 -336.783   -248.80
## 2       Limit    0.172   0.00507      33.9 2.53e-119    0.163      0.18
```

1.3 Predicting Future Responses

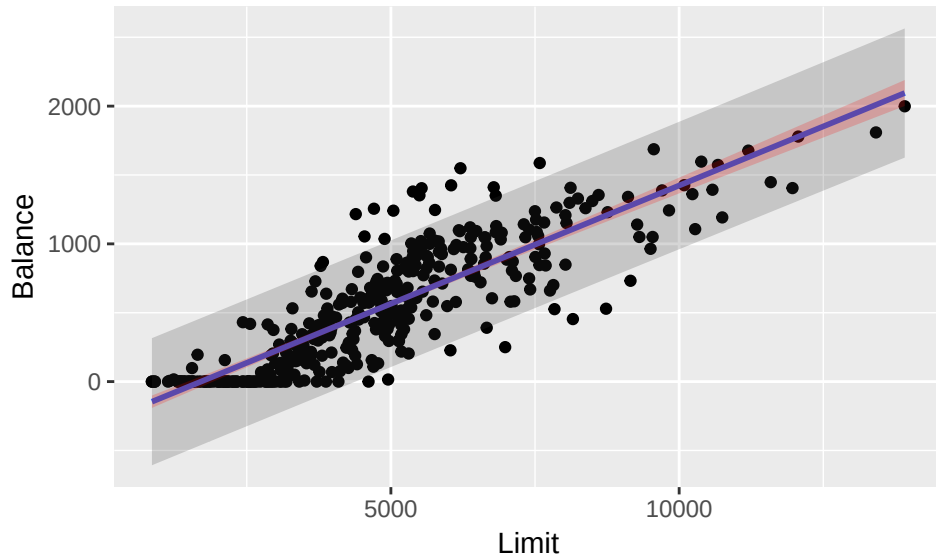
A typical use of the regression model is to predict average or future values for the response variable (here `Balance`, the credit card balance of each individual). We give the predictions for three individuals with `Limit` of $2,000, $5,000, and $7,0000.

```
newcredit <- data.frame(Limit=c(2000, 5000, 7000))
crit_val <- qt(.975, glance(credit_lm)$df.resid)

credit_pred <- augment(credit_lm, newdata=newcredit, type.predict = "response")

credit_pred <- credit_pred %>%
  mutate(lower_CI = .fitted -  crit_val * .se.fit,
           upper_CI = .fitted + crit_val * .se.fit)

# the SE of the predictions also include the overall variability of the model

.se.pred <- sqrt(glance(credit_lm)$sigma^2 + credit_pred$.se.fit)
credit_pred <- credit_pred %>%
  mutate(lower_PI = .fitted - crit_val * .se.pred,
```

```
         upper_PI = .fitted + crit_val * .se.pred,
         lower_CI = .fitted - crit_val * .se.fit,
         upper_CI = .fitted + crit_val * .se.fit)

credit_pred

##   Limit .fitted .se.fit lower_CI upper_CI lower_PI upper_PI
## 1  2000    50.5    18.1     14.9     86.1     -409      510
## 2  5000   565.4    11.8    542.3    588.5      106     1025
## 3  7000   908.7    16.4    876.5    940.9      449     1368
```

To plot the CI and PI for all the points in the dataset, the SE of the prediction is needed. Note that the SE of the prediction uses both the SE of the fitted value (i.e., the variability of the line at a given value) as well as the overall variability of the points (`sigma` from `glance`). On the plot below, the interior (red) ribbon represent 95% confidence bounds on the mean (expected) values. The exterior (grey) ribbon represent 95% confidence bounds on the future predicted values.

```
credit_gl <- broom::glance(credit_lm)
credit_sig <- dplyr::pull(credit_gl, sigma)

credit_pred <- broom::augment(credit_lm) %>%
  mutate(.se.pred = sqrt(credit_sig^2 + .se.fit^2)) %>%
  mutate(lower_PI = .fitted - crit_val*.se.pred,
         upper_PI = .fitted + crit_val*.se.pred,
         lower_CI = .fitted - crit_val * .se.fit,
         upper_CI = .fitted + crit_val * .se.fit)

credit_pred %>% head()

##   Balance Limit .fitted .se.fit  .resid    .hat .sigma  .cooksd .std.resid
## 1     333  3606     326    13.0    6.87 0.00310    234 1.35e-06     0.0294
## 2     903  6645     848    15.2   55.26 0.00422    234 1.19e-04     0.2371
## 3     580  7075     922    16.6 -341.54 0.00507    233 5.48e-03    -1.4659
## 4     964  9504    1338    26.8 -374.45 0.01320    233 1.74e-02    -1.6137
## 5     331  4897     548    11.7 -216.72 0.00251    234 1.09e-03    -0.9290
## 6    1151  8047    1088    20.4   62.63 0.00766    234 2.80e-04     0.2691
##   .se.pred lower_PI upper_PI lower_CI upper_CI
## 1      234   -133.8      786      301      352
## 2      234    387.6     1308      818      878
## 3      234    461.2     1382      889      954
## 4      235    876.2     1801     1286     1391
## 5      234     87.9     1008      525      571
## 6      234    627.4     1549     1048     1129

ggplot(credit_pred, aes(x = Limit, y = Balance)) + geom_point() +
  stat_smooth(method = "lm", se = FALSE) +
  geom_ribbon(aes(ymin = lower_PI, ymax = upper_PI), alpha = .2) +
  geom_ribbon(data = credit_pred, aes(ymin = lower_CI, ymax = upper_CI), alpha = .2, fill = "red")
```



Using different multipliers (for simultaneous inference)

```
num_int <- 3
crit_Bonf <- qt((1-.975)/num_int, glance(credit_lm)$df.resid)
crit_WH <- sqrt(2*qf(.95, num_int, glance(credit_lm)$df.resid))
```

2 Transforming Data

2.1 Residual Analysis

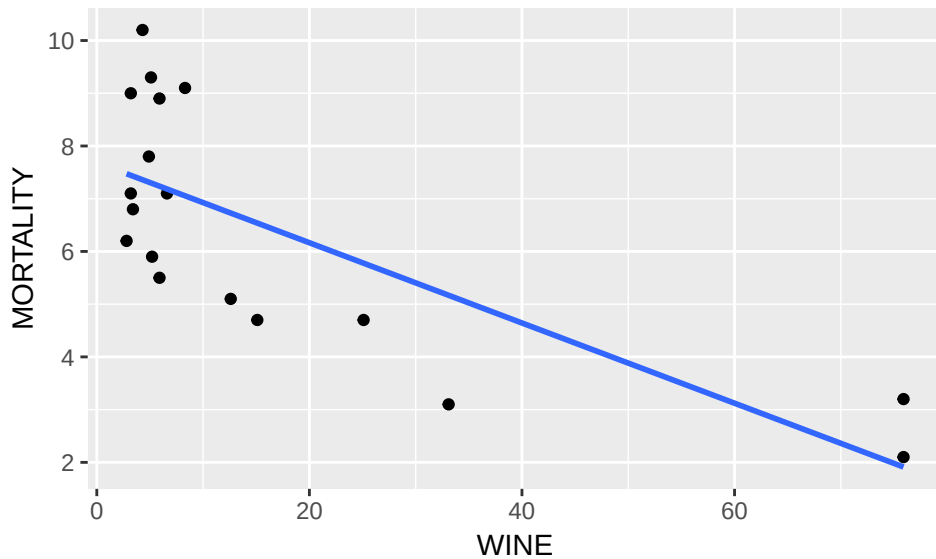
First read the data into R; it is comma delimited (make sure that the data are in the correct place to be read in, or you can read them in directly from the URL). The data consist of wine consumption and heart attack mortality. Columns are the name of the country, the wine consumption (liters per person per year), and the heart disease mortality (% of deaths due to heart disease per year).

```
wineheart <- read.table("wineheart.csv", header=T, sep=",")
skim(wineheart)

## Skim summary statistics
##  n obs: 18
##  n variables: 3
##
## Variable type: factor
##   variable missing complete  n n_unique                     top_counts
## 1  COUNTRY       0       18 18       18 Aus: 1, Aus: 1, Bel: 1, Can: 1
##   ordered
## 1   FALSE
##
## Variable type: numeric
##    variable missing complete  n  mean    sd min  p25 median   p75  max
```

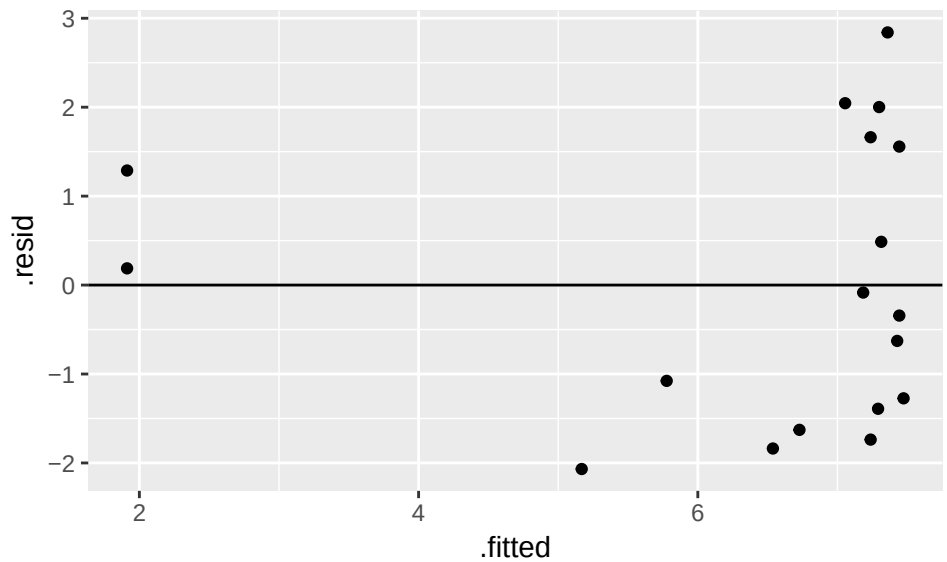
```
## 1 MORTALITY       0       18 18  6.43  2.36 2.1 4.8     6.5  8.62 10.2
## 2      WINE       0       18 18 16.47 23.1  2.8 4.45    5.9 14.48 75.9
##       hist
## 1
## 2

wine_lm <- lm(MORTALITY ~ WINE, data=wineheart)
```

Below is the code for various residual plots. Note that in R the command `log` specifies the natural log. Note that the residual plots tell the same story regardless of the scaling.

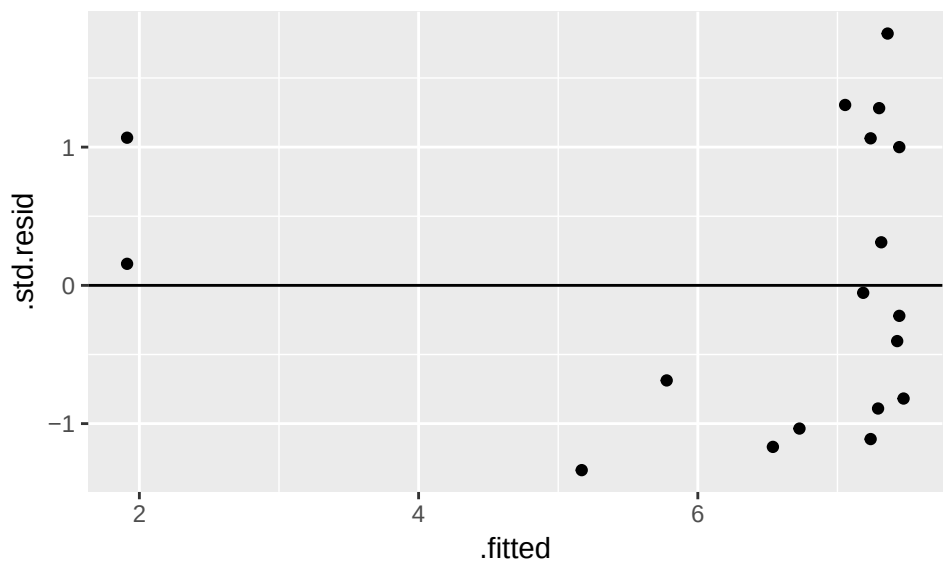
```
wineheart %>%
  ggplot(aes(x=WINE, y=MORTALITY)) +
  geom_point() +
  geom_smooth(method="lm", se=FALSE)
```




```
augment(wine_lm) %>%
  ggplot( aes(x=.fitted, y=.resid)) +
  geom_point() +
  geom_hline(yintercept = 0)
```

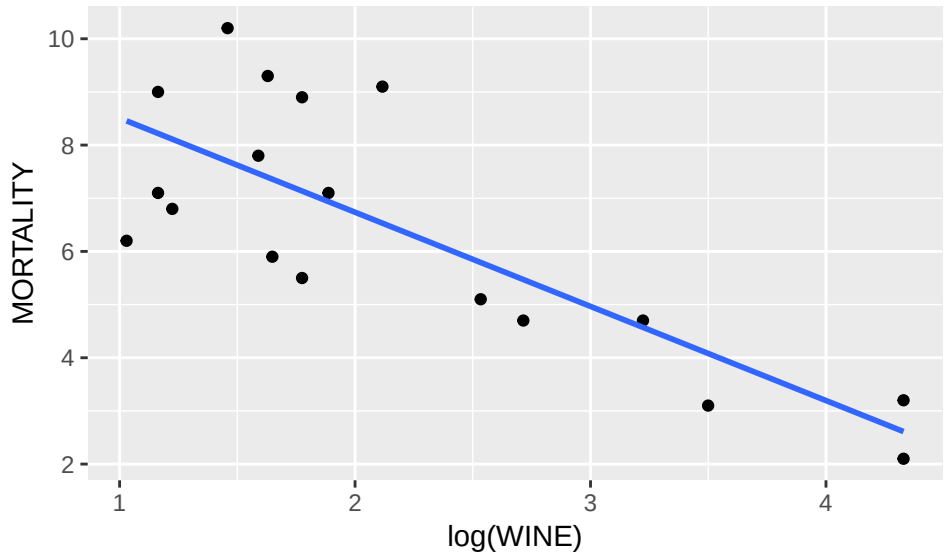



```
augment(wine_lm) %>%
  ggplot( aes(x=.fitted, y=.std.resid)) +
  geom_point() +
  geom_hline(yintercept = 0)
```

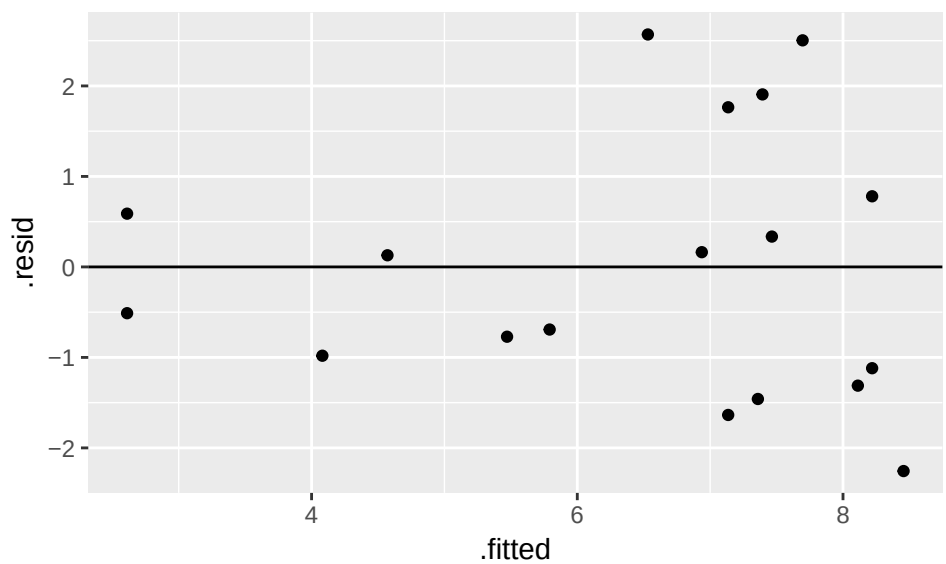


We see a different story by re-running the analysis after transforming WINE (*explanatory variable*) by taking a natural log.

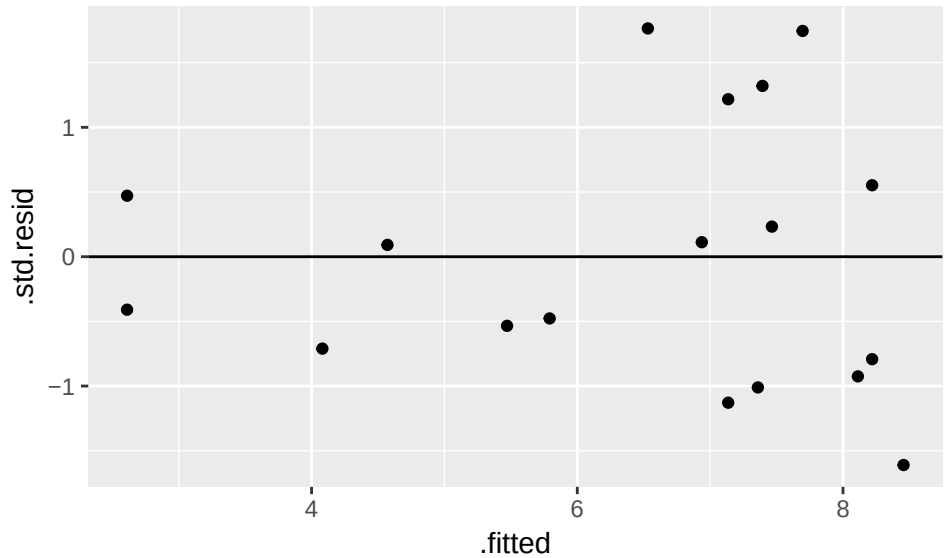
```
wine_lm2 <- lm(MORTALITY ~ log(WINE), data=wineheart)

wineheart %>%
  ggplot(aes(x=log(WINE), y=MORTALITY)) +
  geom_point() +
  geom_smooth(method="lm", se=FALSE)
```




```
augment(wine_lm2) %>%
  ggplot( aes(x=.fitted, y=.resid)) +
  geom_point() +
  geom_hline(yintercept = 0)
```

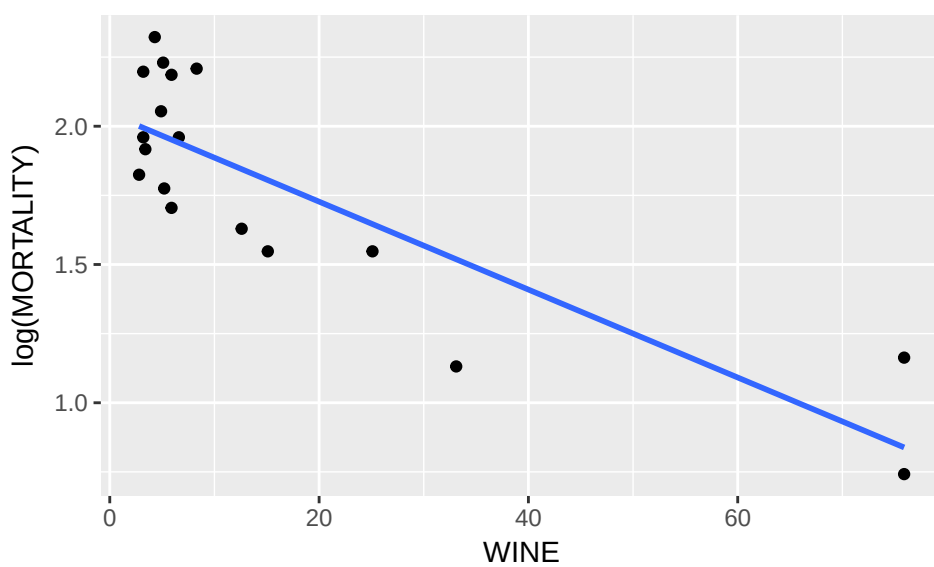



```
augment(wine_lm2) %>%
  ggplot( aes(x=.fitted, y=.std.resid)) +
  geom_point() +
  geom_hline(yintercept = 0)
```

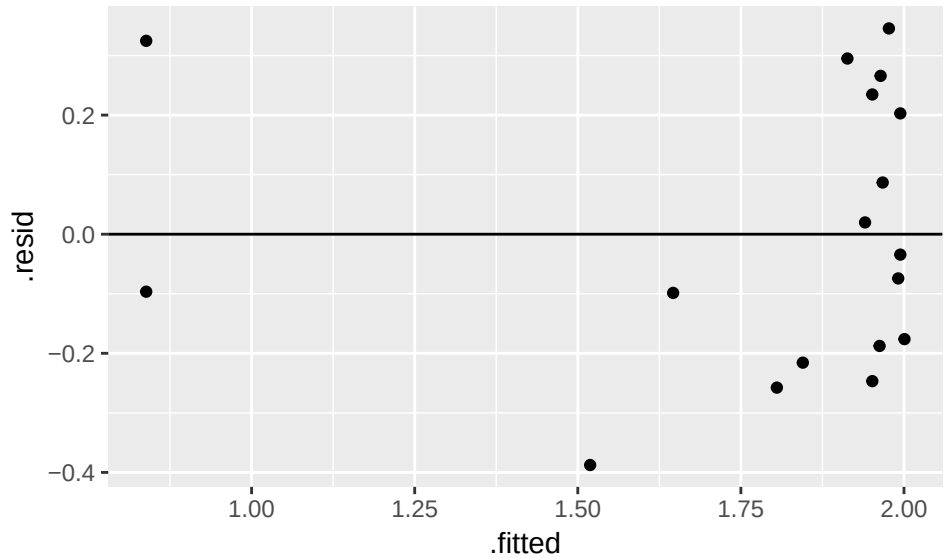


What happens when we transform the MORTALITY (*response*) variable by a natural log instead of the WINE (*explanatory*) variable.

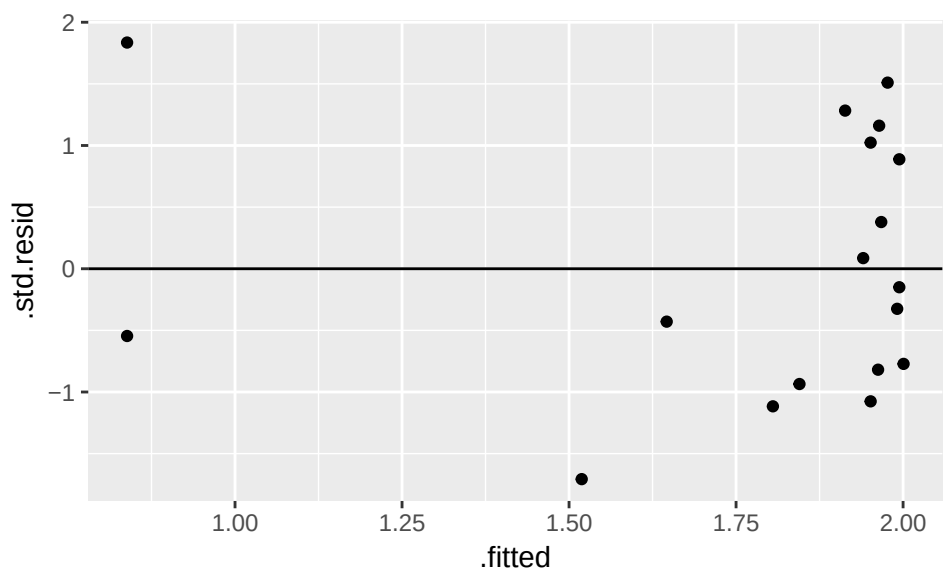
```
wine_lm3 <- lm(log(MORTALITY) ~ WINE, data=wineheart)

wineheart %>%
  ggplot(aes(x=WINE, y=log(MORTALITY))) +
  geom_point() +
  geom_smooth(method="lm", se=FALSE)
```




```
augment(wine_lm3) %>%
  ggplot( aes(x=.fitted, y=.resid)) +
  geom_point() +
  geom_hline(yintercept = 0)
```

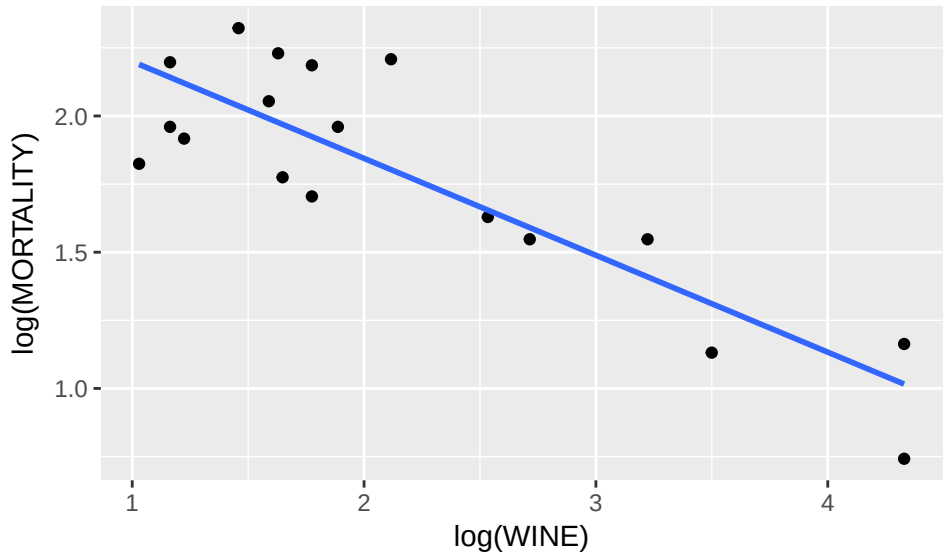



```
augment(wine_lm3) %>%
  ggplot( aes(x=.fitted, y=.std.resid)) +
  geom_point() +
  geom_hline(yintercept = 0)
```

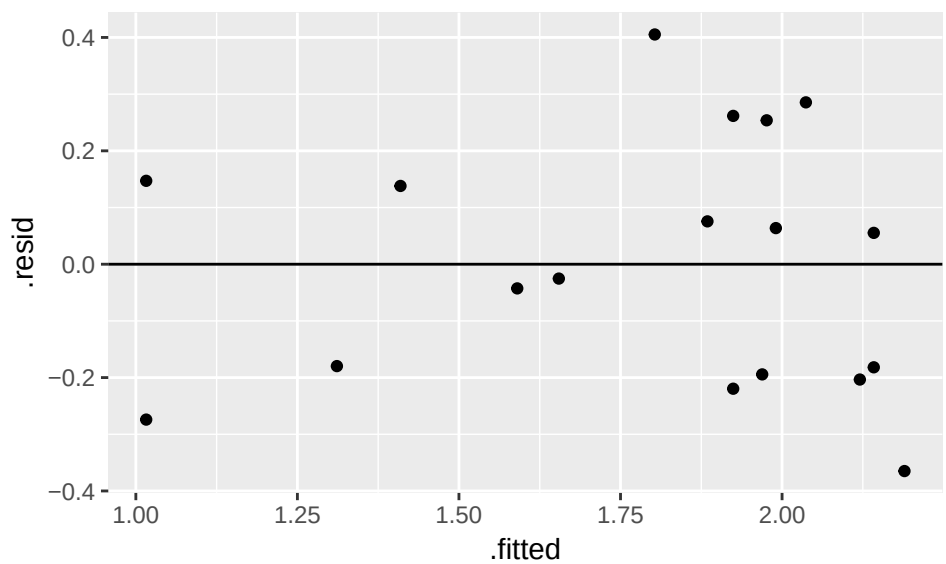


And last, let's look at the residual analysis after having transformed both MORTALITY (*response*) and WINE (*explanatory*).

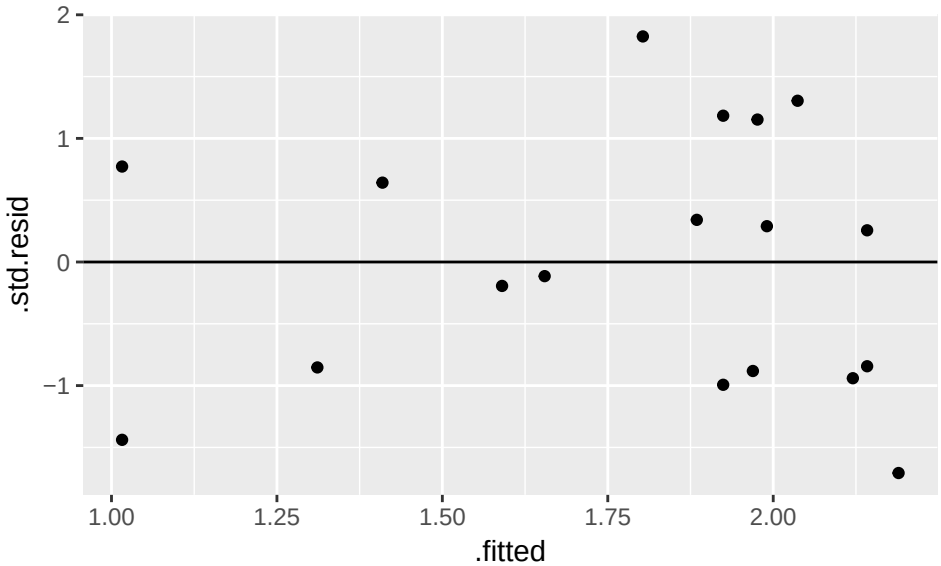
```
wine_lm4 <- lm(log(MORTALITY) ~ log(WINE), data=wineheart)

wineheart %>%
  ggplot(aes(x=log(WINE), y=log(MORTALITY))) +
  geom_point() +
  geom_smooth(method="lm", se=FALSE)
```




```
augment(wine_lm4) %>%
  ggplot( aes(x=.fitted, y=.resid)) +
  geom_point() +
  geom_hline(yintercept = 0)
```




```
augment(wine_lm4) %>%
  ggplot( aes(x=.fitted, y=.std.resid)) +
  geom_point() +
  geom_hline(yintercept = 0)
```



Take home message: Transforming X (WINE) removes the effect of the extreme value(s). Transforming Y (MORTALITY) creates more constant variance. Also, both transformations impacted the shape of the relationship between the two variables. A linear model is one that has linear coefficients describing whatever variables are at hand. A linear model on transformed variables is still considered to be a linear model.