Getting Data and Linear Models

Canhong Wen

Agenda

- Getting data into and out of R
- Using data frames for statistical purposes
- Common transformations of numerical data
- Re-ordering data frames
- Merging data frames

Reading Data from R

- You can load and save R objects
  - R has its own format for this, which is shared across operating systems
  - It's an open, documented format if you really want to pry into it
- `save(thing, file="name")` saves `thing` in a file called `name` (conventional extension: `rda` or `Rda`)
- `load("name")` loads the object or objects stored in the file called `name`, *with their old names*

```
gmp <- read.table("data/gmp.dat")
gmp$pop <- round(gmp$gmp/gmp$pcgmp)
save(gmp,file="data/gmp.Rda")
rm(gmp)
exists("gmp")
```

```
## [1] FALSE
```

```
not_gmp <- load(file="data/gmp.Rda")
colnames(gmp)
```

```
## [1] "MSA"   "gmp"   "pcgmp" "pop"
```

```
not_gmp
```

```
## [1] "gmp"
```

- We can load or save more than one object at once; this is how RStudio will load your whole workspace when you're starting, and offer to save it when you're done
- Many packages come with saved data objects; there's the convenience function `data()` to load them

```
data(cats,package="MASS")
summary(cats)
```

```
##  Sex         Bwt             Hwt
##  F:47   Min.   :2.000   Min.   : 6.30
##  M:97   1st Qu.:2.300   1st Qu.: 8.95
##         Median :2.700   Median :10.10
##          Mean   :2.724   Mean   :10.63  
##          3rd Qu.:3.025   3rd Qu.:12.12  
##          Max.   :3.900   Max.   :20.50  
```

Non-R Data Tables

- Tables full of data, just not in the R file format
- Main function: `read.table()`
  - Presumes space-separated fields, one line per row
  - Main argument is the file name or URL
  - Returns a dataframe
  - Lots of options for things like field separator, column names, forcing or guessing column types, skipping lines at the start of the file…
- `read.csv()` is a short-cut to set the options for reading comma-separated value (CSV) files
  - Spreadsheets will usually read and write CSV

Writing Dataframes

- Counterpart functions `write.table()`, `write.csv()` write a dataframe into a file
- Drawback: takes a lot more disk space than what you get from `load` or `save`
- Advantage: can communicate with other programs, or even edit manually

Less Friendly Data Formats

- The `foreign` package on CRAN has tools for reading data files from lots of non-R statistical software
- Spreadsheets are special
- Full of ugly irregularities
- Values or formulas?
- Headers, footers, side-comments, notes
- Columns change meaning half-way down

Spreadsheets, If You Have To

- Save the spreadsheet as a CSV; `read.csv()`
- Save the spreadsheet as a CSV; edit in a text editor; `read.csv()`
- Use `read.xls()` from the `gdata` package
- Tries very hard to work like `read.csv()`, can take a URL or filename
- Can skip down to the first line that matches some pattern, select different sheets, etc.
- You may still need to do a lot of tidying up after

So You've Got A Data Frame

What can we do with it?

- Plot it: examine multiple variables and distributions
- Test it: compare groups of individuals to each other
- Check it: does it conform to what we'd like for our needs

Test Case: Birth weight data

```r
library(MASS)
data(birthwt)
summary(birthwt)
```

```
##       low              age             lwt             race      
##  Min.   :0.0000   Min.   :14.00   Min.   : 80.0   Min.   :1.000  
##  1st Qu.:0.0000   1st Qu.:19.00   1st Qu.:110.0   1st Qu.:1.000  
##  Median :0.0000   Median :23.00   Median :121.0   Median :1.000  
##  Mean   :0.3122   Mean   :23.24   Mean   :129.8   Mean   :1.847  
##  3rd Qu.:1.0000   3rd Qu.:26.00   3rd Qu.:140.0   3rd Qu.:3.000  
##  Max.   :1.0000   Max.   :45.00   Max.   :250.0   Max.   :3.000  
##      smoke             ptl               ht                ui        
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.00000   Min.   :0.0000  
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.0000  
##  Median :0.0000   Median :0.0000   Median :0.00000   Median :0.0000  
##  Mean   :0.3915   Mean   :0.1958   Mean   :0.06349   Mean   :0.1481  
##  3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.0000  
##  Max.   :1.0000   Max.   :3.0000   Max.   :1.00000   Max.   :1.0000  
##       ftv              bwt      
##  Min.   :0.0000   Min.   : 709  
##  1st Qu.:0.0000   1st Qu.:2414  
##  Median :0.0000   Median :2977  
##  Mean   :0.7937   Mean   :2945  
##  3rd Qu.:1.0000   3rd Qu.:3487  
##  Max.   :6.0000   Max.   :4990  
```

From R help

Go to R help for more info, because someone documented this data

`help(birthwt)`

Make it Readable

```r
colnames(birthwt)
```

```
##  [1] "low"   "age"   "lwt"   "race"  "smoke" "ptl"   "ht"    "ui"   
##  [9] "ftv"   "bwt"
```

```r
colnames(birthwt) <- c("birthwt.below.2500", "mother.age",
                       "mother.weight", "race",
                       "mother.smokes", "previous.prem.labor",
                       "hypertension", "uterine.irr",
                       "physician.visits", "birthwt.grams")
```

Make it Readable

Can make all the factors more descriptive.

```r
birthwt$race <- factor(c("white", "black", "other")[birthwt$race])
birthwt$mother.smokes <- factor(c("No", "Yes")[birthwt$mother.smokes + 1])
```

```
birthwt$uterine.irr <- factor(c("No", "Yes")[birthwt$uterine.irr + 1])
birthwt$hypertension <- factor(c("No", "Yes")[birthwt$hypertension + 1])
```

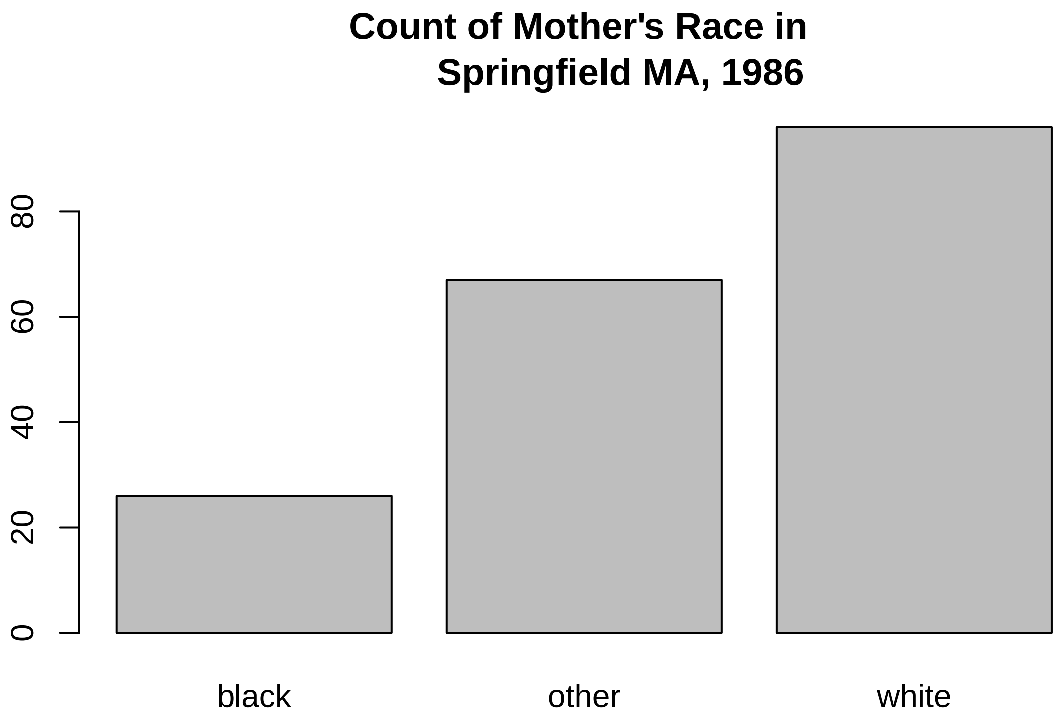
Make it Readable

```
summary(birthwt)
```

```
##  birthwt.below.2500   mother.age    mother.weight      race   
##  Min.   :0.0000     Min.   :14.00   Min.   : 80.0   black:26  
##  1st Qu.:0.0000     1st Qu.:19.00   1st Qu.:110.0   other:67  
##  Median :0.0000     Median :23.00   Median :121.0   white:96  
##  Mean   :0.3122     Mean   :23.24   Mean   :129.8             
##  3rd Qu.:1.0000     3rd Qu.:26.00   3rd Qu.:140.0             
##  Max.   :1.0000     Max.   :45.00   Max.   :250.0             
##  mother.smokes previous.prem.labor hypertension uterine.irr
##  No :115       Min.   :0.0000      No :177      No :161    
##  Yes: 74       1st Qu.:0.0000      Yes: 12      Yes: 28    
##                Median :0.0000                              
##                Mean   :0.1958                              
##                3rd Qu.:0.0000                              
##                Max.   :3.0000                              
##  physician.visits birthwt.grams 
##  Min.   :0.0000   Min.   : 709  
##  1st Qu.:0.0000   1st Qu.:2414  
##  Median :0.0000   Median :2977  
##  Mean   :0.7937   Mean   :2945  
##  3rd Qu.:1.0000   3rd Qu.:3487  
##  Max.   :6.0000   Max.   :4990
```

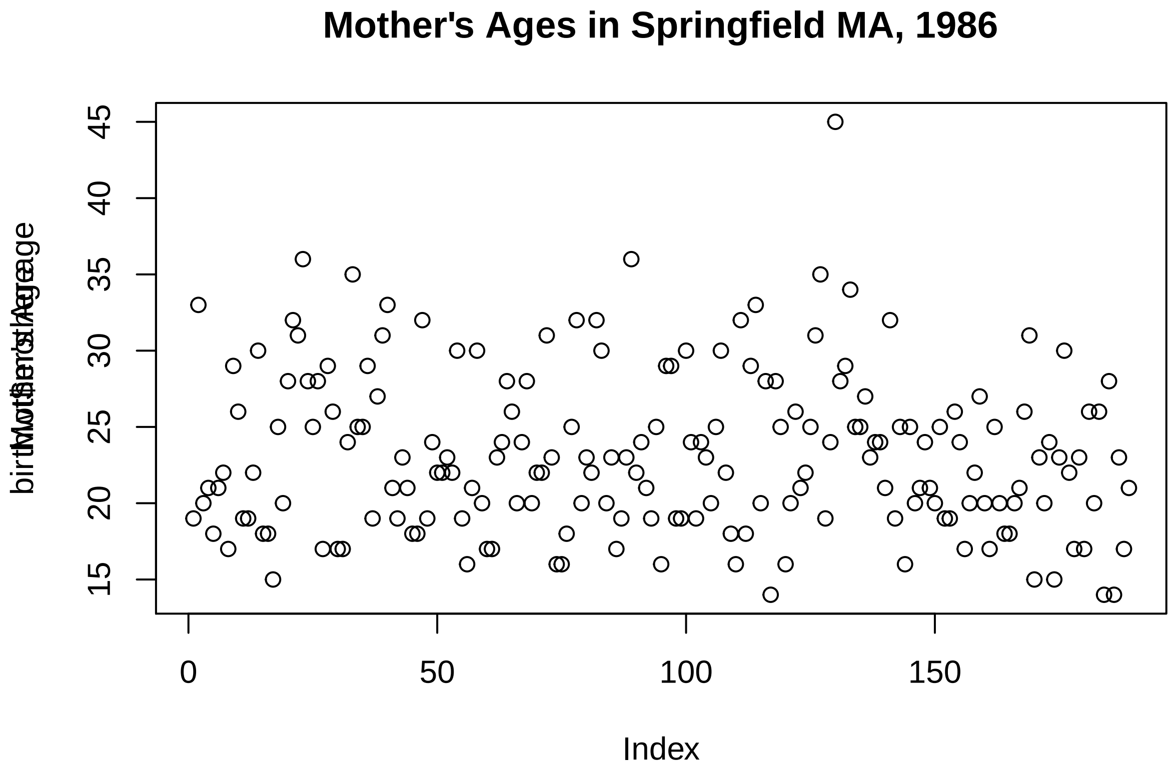
Explore It

```
plot (birthwt$race)
title (main = "Count of Mother's Race in
       Springfield MA, 1986")
```



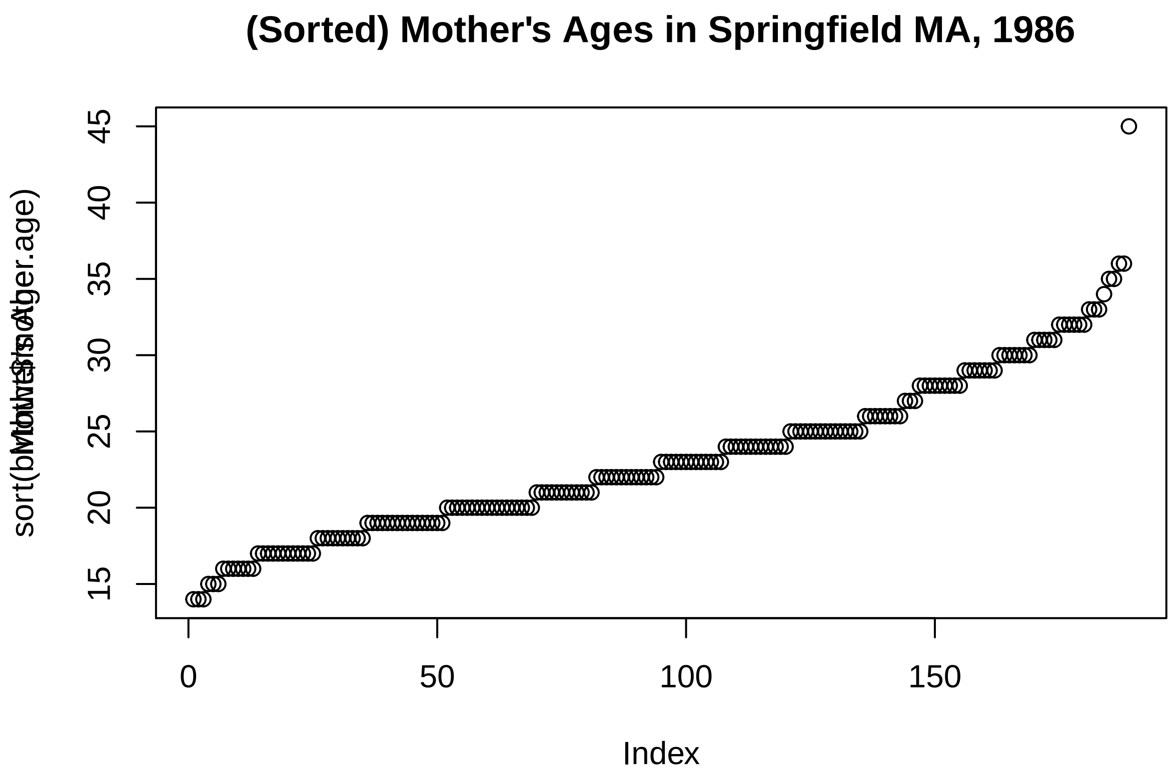
Explore It

```
plot (birthwt$mother.age)
title (main = "Mother's Ages in Springfield MA, 1986", ylab="Mother's Age")
```

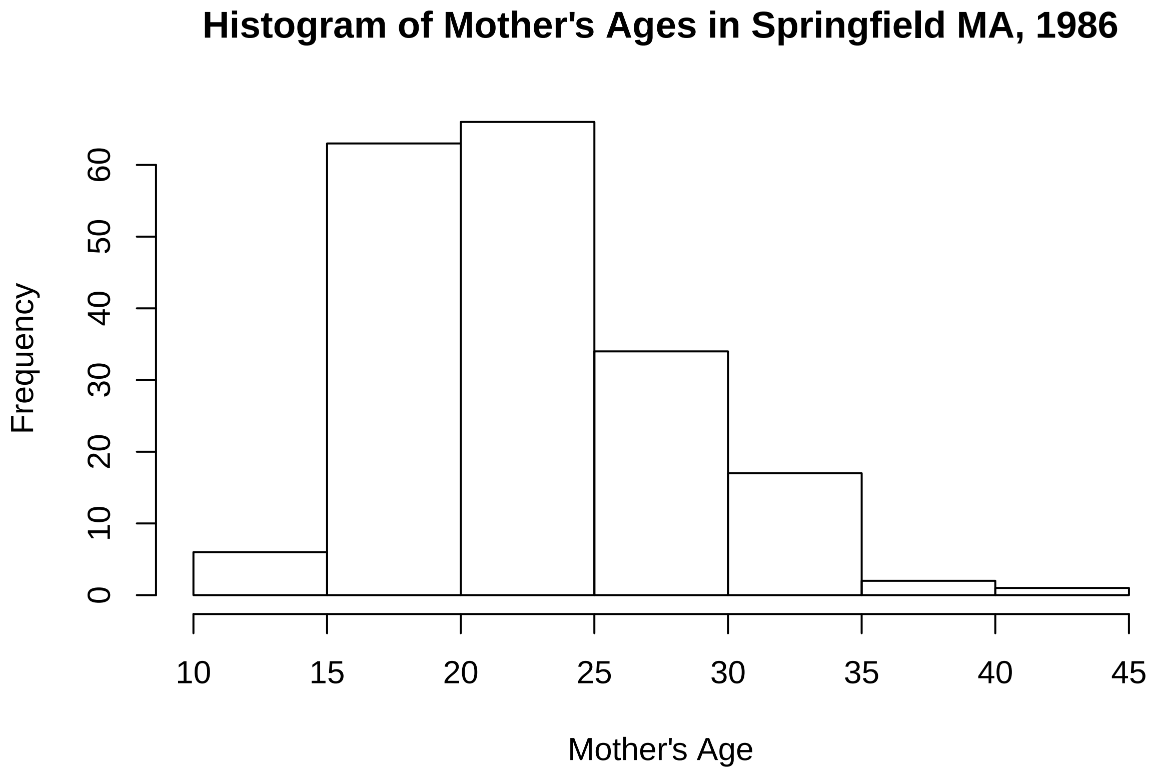


Explore It

```r
plot (sort(birthwt$mother.age))
title (main = "(Sorted) Mother's Ages in Springfield MA, 1986", ylab="Mother's Age")
```

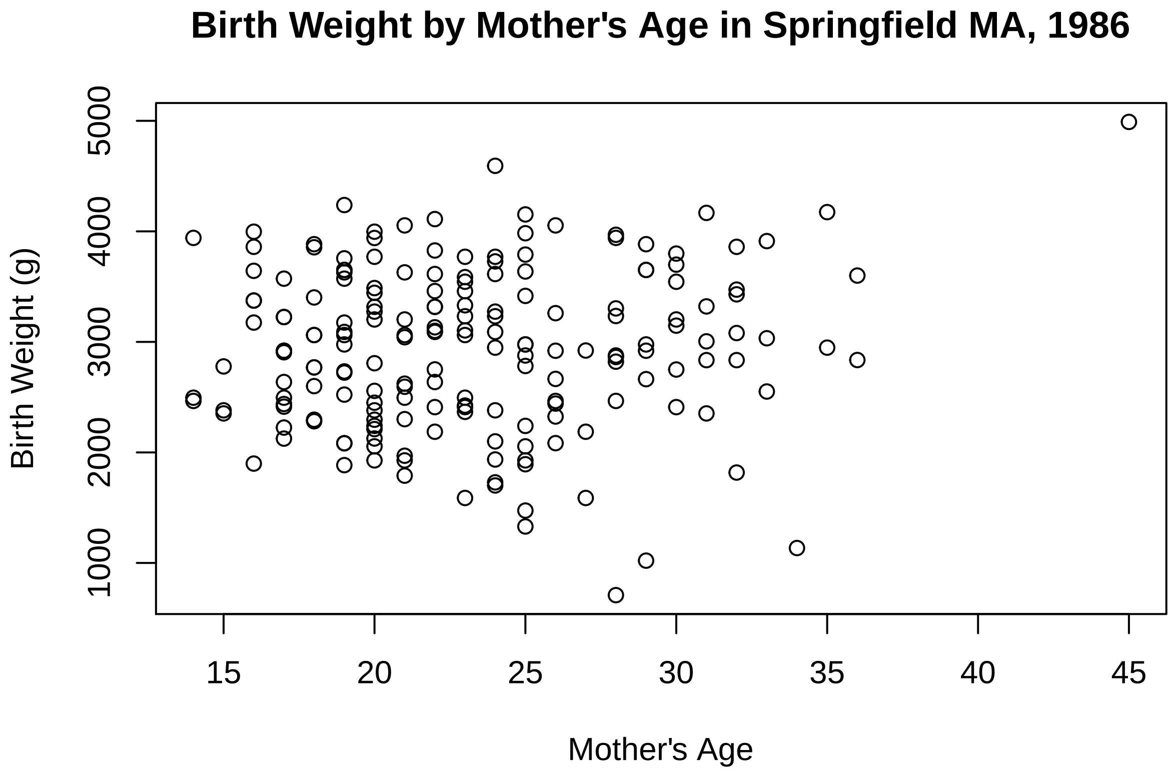


```
hist(birthwt$mother.age, main = "Histogram of Mother's Ages in Springfield MA, 1986", xlab="Mother's Ag
```



Explore It

```
plot (birthwt$mother.age, birthwt$birthwt.grams, xlab = "", ylab = "")
title (main = "Birth Weight by Mother's Age in Springfield MA, 1986",
       xlab="Mother's Age", ylab="Birth Weight (g)")
```



Before testing

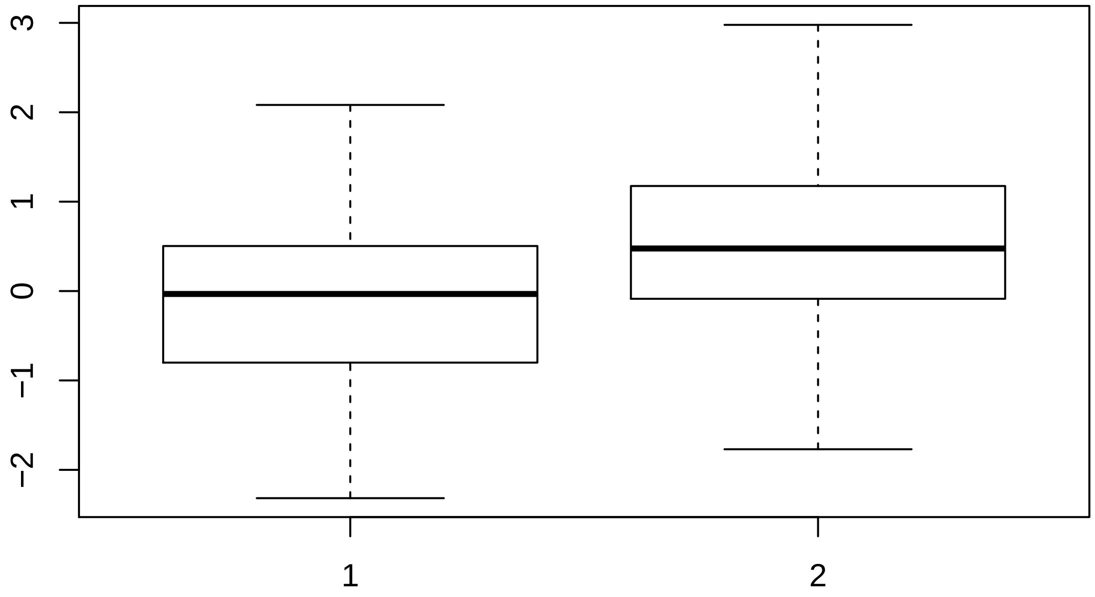
We first introduce some basic statistical model:

1. t test
    - A t-test is a statistical test that is used to compare the means of two groups. It is often used in hypothesis testing to determine whether a process or treatment actually has an effect on the population of interest, or whether two groups are different from one another.
2. linear model
    - A linear model relating the response y to several predictors has the form

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon$$

```
n <- 100
x <- rnorm(n)
y <- 0.5+rnorm(n)

boxplot(x, y)
```

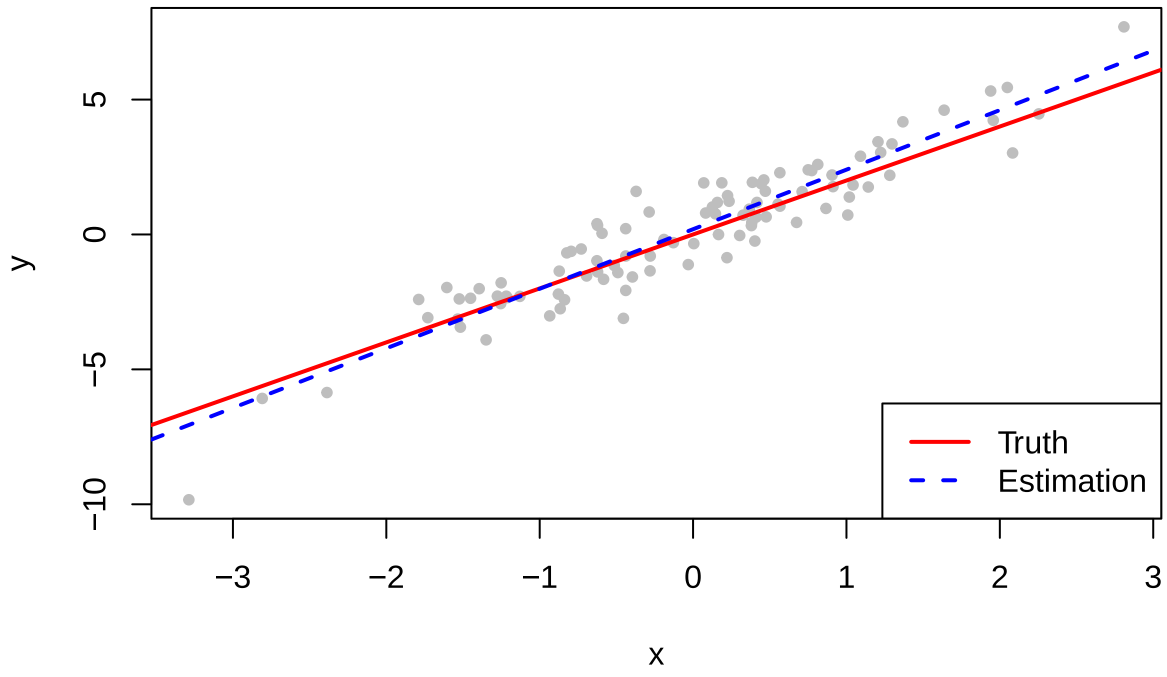


```
t.test(x, y)
```

```
##
##  Welch Two Sample t-test
##
## data:  x and y
## t = -4.7552, df = 197.57, p-value = 3.816e-06
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.8765742 -0.3626491
## sample estimates:
##   mean of x   mean of y
## -0.07809521  0.54151644
```

```
n <- 100
x <- rnorm(n)
y <- 2*x + rnorm(n)
plot(x,y, pch=20, col="grey")
abline(a=0, b=2, col="red", lwd=2)
abline(lm(y~x),  col="blue", lwd=2, lty=2)
```

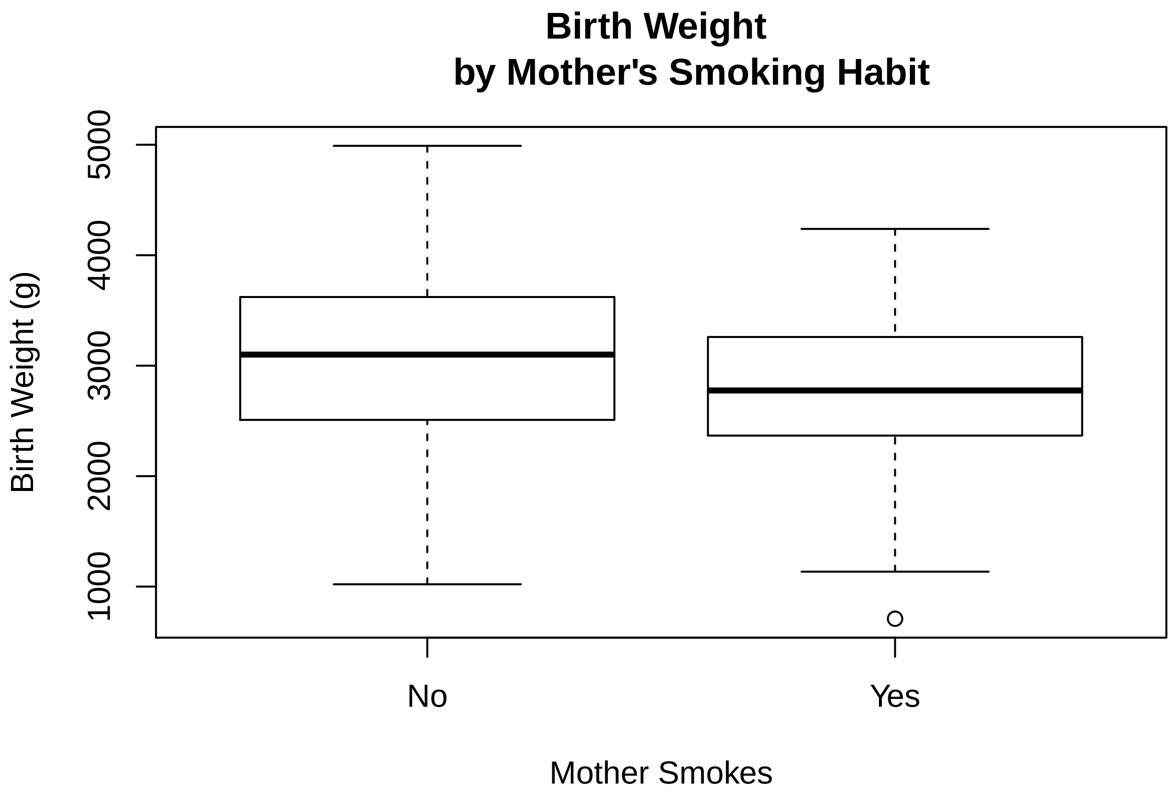
```
legend("bottomright", legend=c("Truth", "Estimation"), col=c("red", "blue"), lty=1:2, lwd=2)
```



Basic statistical testing

Let’s fit some models to the data pertaining to our outcome(s) of interest.

```
plot (birthwt$mother.smokes, birthwt$birthwt.grams, main="Birth Weight
      by Mother's Smoking Habit", ylab = "Birth Weight (g)", xlab="Mother Smokes")
```



Basic statistical testing

Tough to tell! Simple two-sample t-test:

```
t.test (birthwt$birthwt.grams[birthwt$mother.smokes == "Yes"],
        birthwt$birthwt.grams[birthwt$mother.smokes == "No"])
```

```
##
##  Welch Two Sample t-test
##
## data:  birthwt$birthwt.grams[birthwt$mother.smokes == "Yes"] and birthwt$birthwt.grams[birthwt$mothe
## t = -2.7299, df = 170.1, p-value = 0.007003
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -488.97860  -78.57486
## sample estimates:
## mean of x mean of y
##  2771.919  3055.696
```

Basic statistical testing

Does this difference match the linear model?

```
linear.model.1 <- lm (birthwt.grams ~ mother.smokes, data=birthwt)
linear.model.1
```

```
##
## Call:
## lm(formula = birthwt.grams ~ mother.smokes, data = birthwt)
##
## Coefficients:
##      (Intercept)  mother.smokesYes
##           3055.7            -283.8
```

Basic statistical testing

Does this difference match the linear model?

```
summary(linear.model.1)
```

```
##
## Call:
## lm(formula = birthwt.grams ~ mother.smokes, data = birthwt)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2062.9  -475.9    34.3   545.1  1934.3
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        3055.70      66.93  45.653  < 2e-16 ***
## mother.smokesYes   -283.78     106.97  -2.653  0.00867 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 717.8 on 187 degrees of freedom
## Multiple R-squared:  0.03627,    Adjusted R-squared:  0.03112
## F-statistic: 7.038 on 1 and 187 DF,  p-value: 0.008667
```

Basic statistical testing

Does this difference match the linear model?

```
linear.model.2 <- lm (birthwt.grams ~ mother.age, data=birthwt)
linear.model.2
```

```
##
## Call:
## lm(formula = birthwt.grams ~ mother.age, data = birthwt)
##
## Coefficients:
## (Intercept)   mother.age
##     2655.74        12.43
```

Basic statistical testing

```
summary(linear.model.2)
```

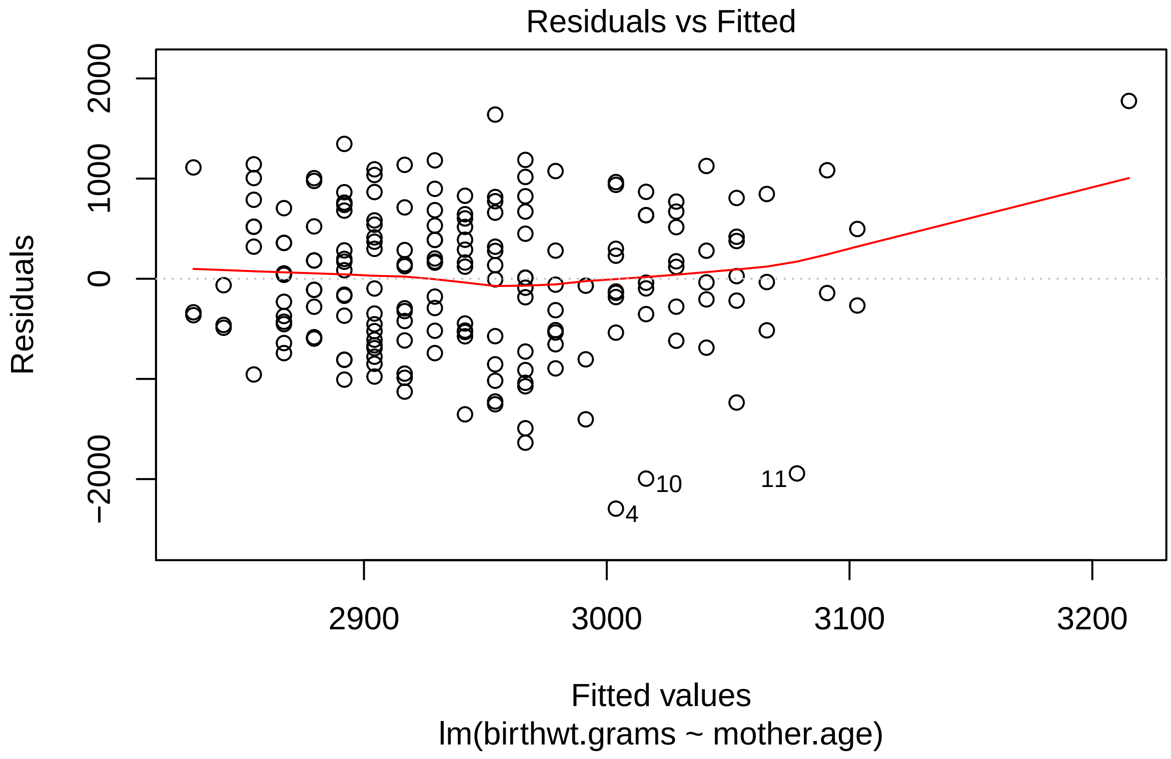
```
##
```

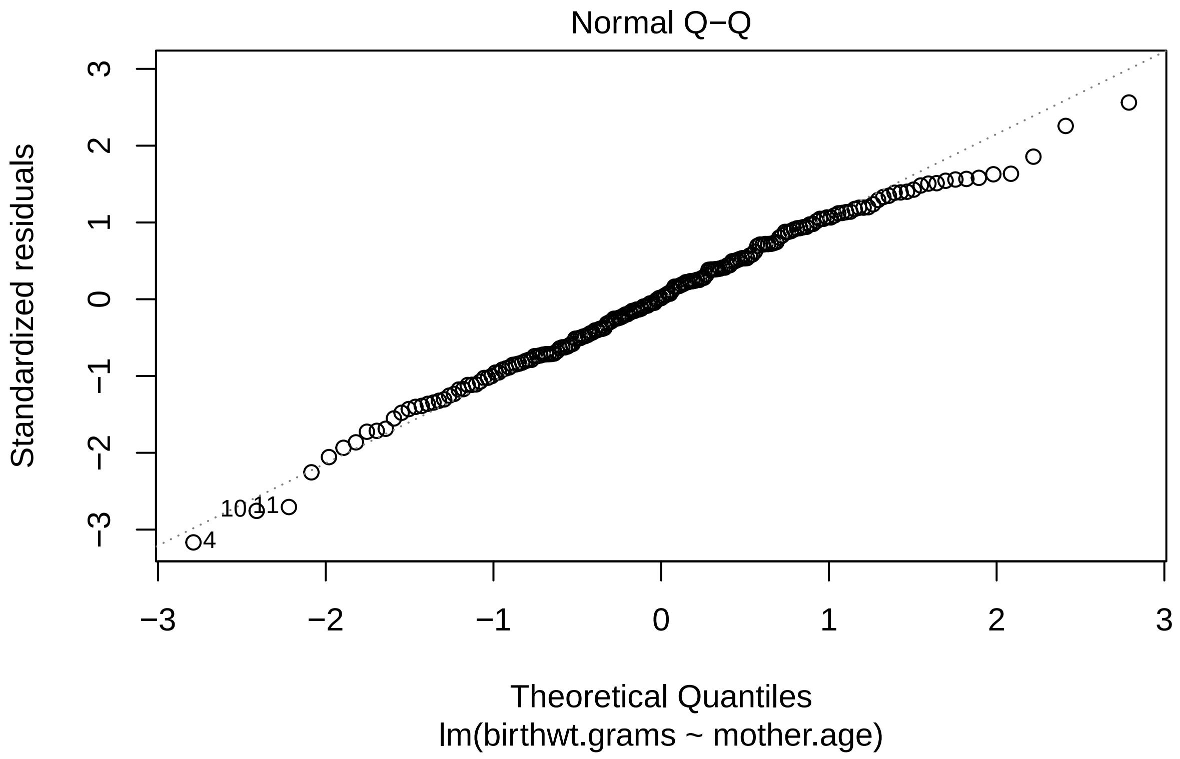
```
## Call:
## lm(formula = birthwt.grams ~ mother.age, data = birthwt)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2294.78  -517.63    10.51   530.80  1774.92
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2655.74     238.86   11.12   <2e-16 ***
## mother.age     12.43      10.02    1.24    0.216
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 728.2 on 187 degrees of freedom
## Multiple R-squared:  0.008157,   Adjusted R-squared:  0.002853
## F-statistic: 1.538 on 1 and 187 DF,  p-value: 0.2165
```

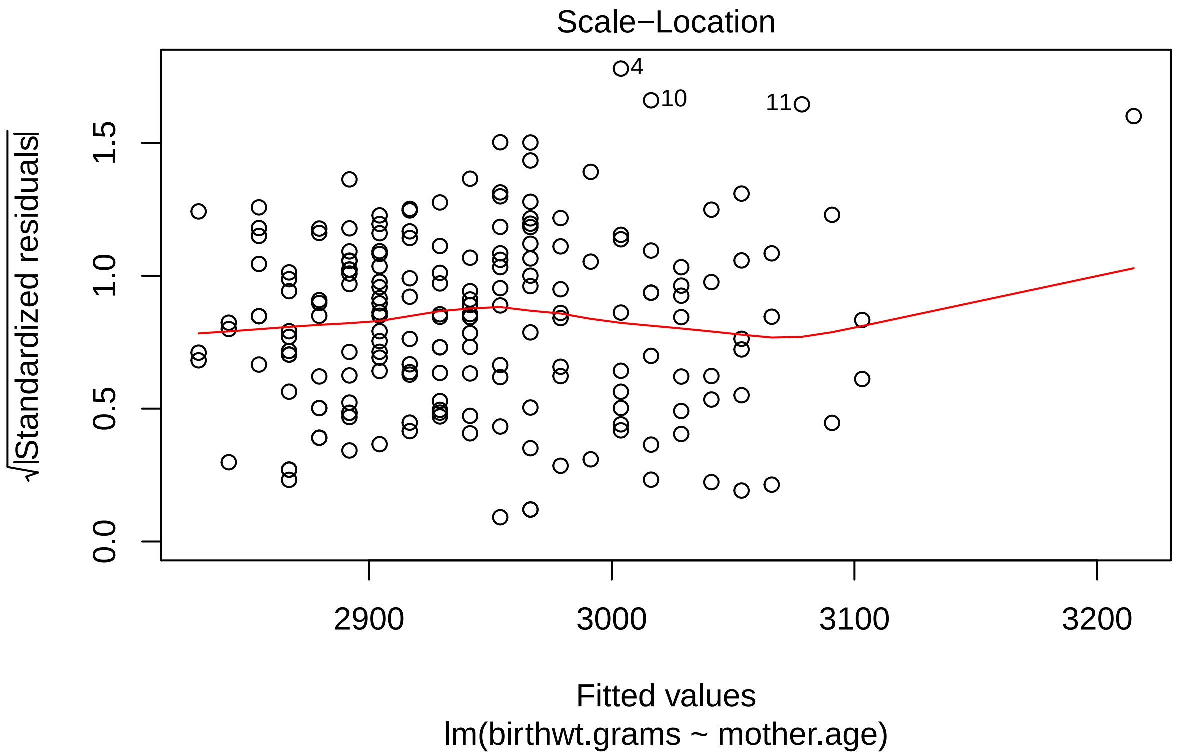
Basic statistical testing

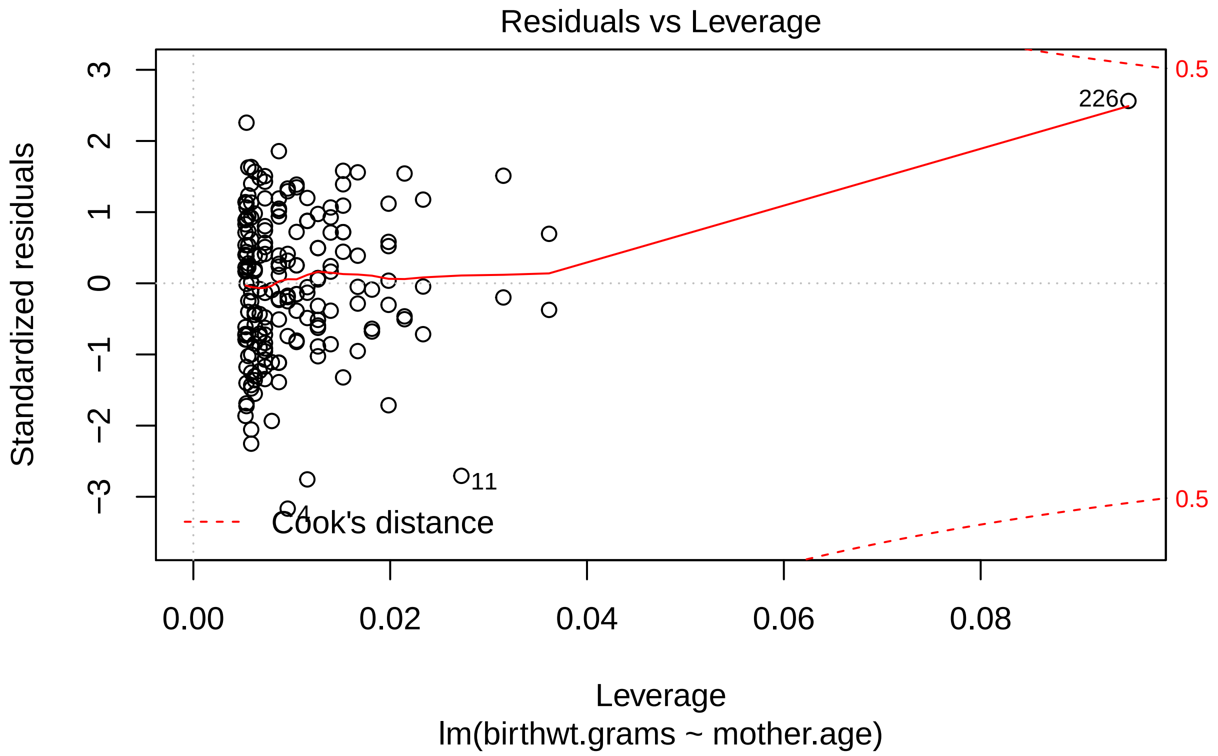
R tries to make diagnostics easy as possible. Try in R console.

```r
plot(linear.model.2)
```


Residuals vs Fitted
Residuals
2000
1000
0
−2000
2900
3000
3100
3200
10
11
4
Fitted values
lm(birthwt.grams ~ mother.age)


Normal Q−Q
Standardized residuals
3
2
1
0
−1
−2
−3
10
11
4
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(birthwt.grams ~ mother.age)


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2900
3000
3100
3200
4
10
11
Fitted values
lm(birthwt.grams ~ mother.age)



Detecting Outliers

Note the oldest mother and her heaviest child are greatly skewing this analysis.

```
birthwt.noout <- birthwt[birthwt$mother.age <= 40,]
linear.model.3 <- lm (birthwt.grams ~ mother.age, data=birthwt.noout)
linear.model.3
```

```
##
## Call:
## lm(formula = birthwt.grams ~ mother.age, data = birthwt.noout)
##
## Coefficients:
## (Intercept)   mother.age
##    2833.273        4.344
```

Detecting Outliers

```
summary(linear.model.3)
```

```
##
## Call:
## lm(formula = birthwt.grams ~ mother.age, data = birthwt.noout)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max 
## -2245.89  -511.24    26.45   540.09  1655.48 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 2833.273    244.954   11.57   <2e-16 ***
## mother.age     4.344     10.349    0.42    0.675    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 717.2 on 186 degrees of freedom
## Multiple R-squared:  0.0009461,  Adjusted R-squared:  -0.004425 
## F-statistic: 0.1761 on 1 and 186 DF,  p-value: 0.6752
```

More complex models

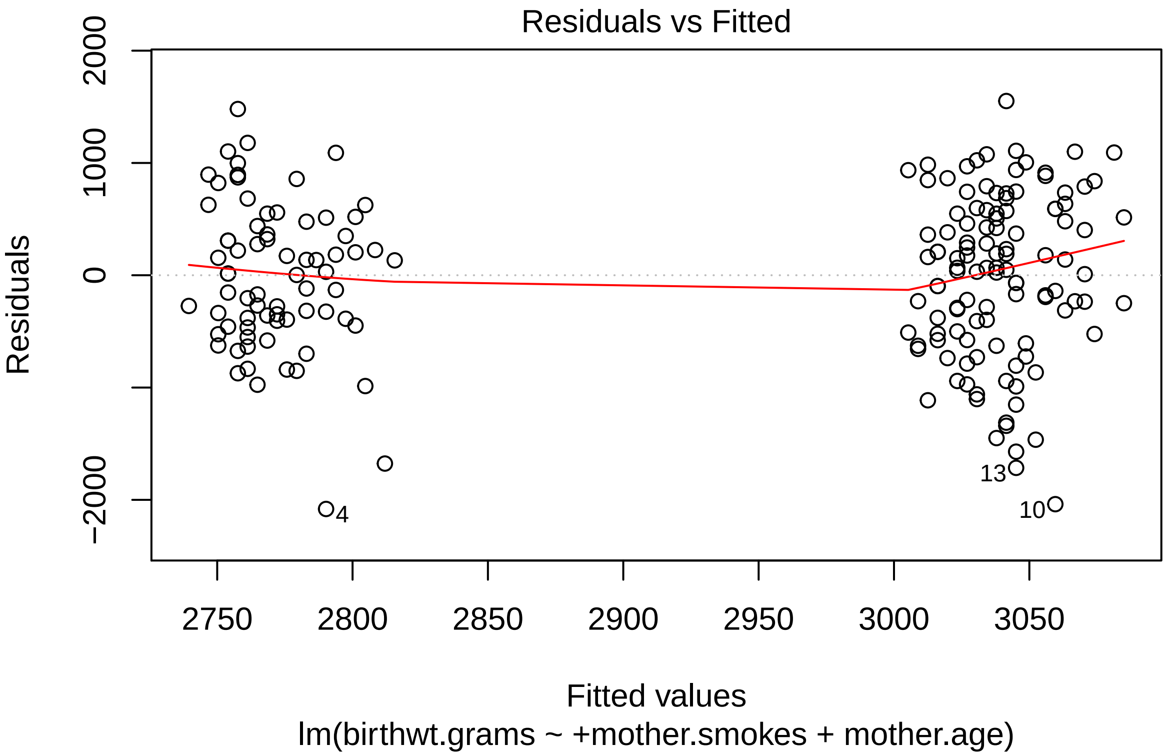
Add in smoking behavior:

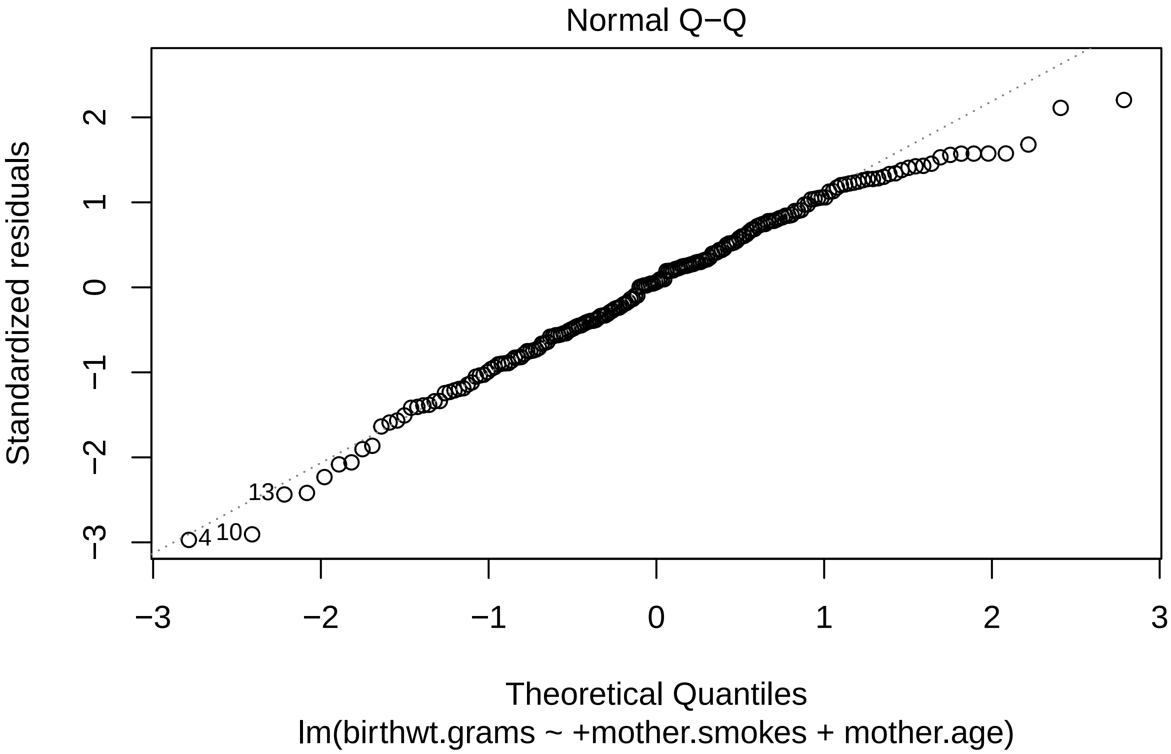
```
linear.model.3a <- lm (birthwt.grams ~ + mother.smokes + mother.age, data=birthwt.noout)
summary(linear.model.3a)
```

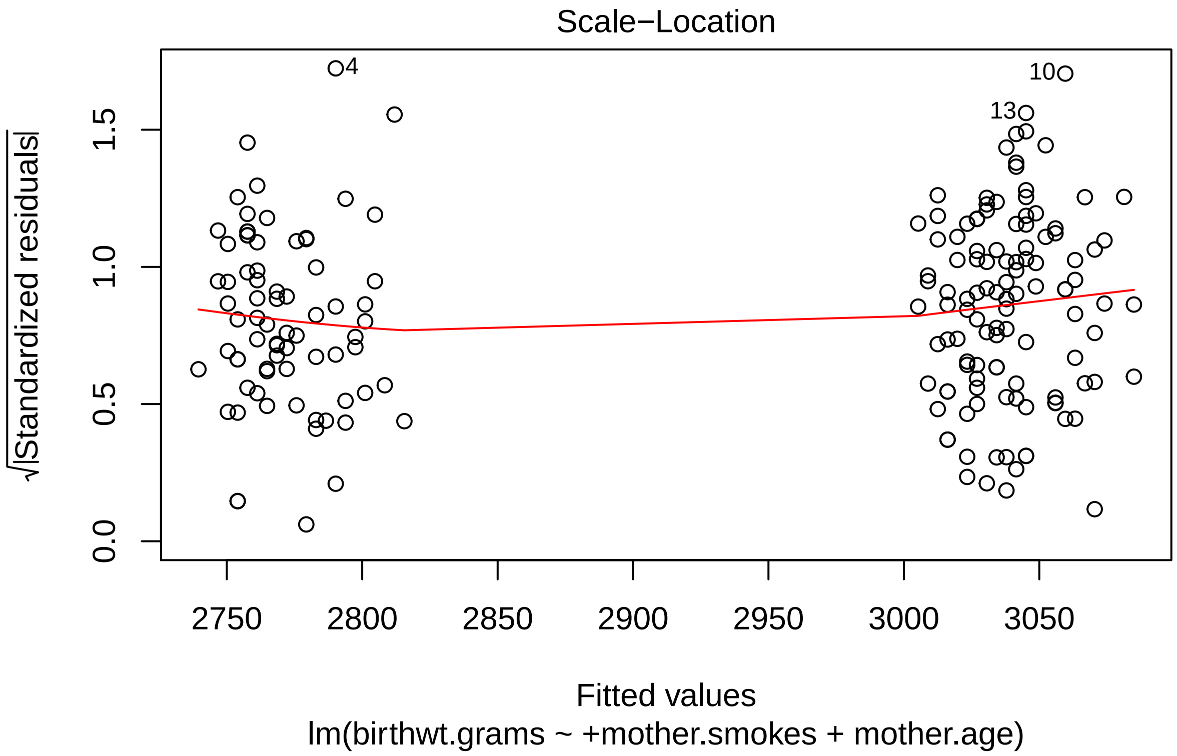
```
## 
## Call:
## lm(formula = birthwt.grams ~ +mother.smokes + mother.age, data = birthwt.noout)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -2081.22  -459.82    43.56   548.22  1551.51 
## 
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)    
## (Intercept)      2954.582    246.280  11.997   <2e-16 ***
## mother.smokesYes -265.756    105.605  -2.517   0.0127 *  
## mother.age          3.621     10.208   0.355   0.7232    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 707.1 on 185 degrees of freedom
## Multiple R-squared:  0.03401,    Adjusted R-squared:  0.02357 
## F-statistic: 3.257 on 2 and 185 DF,  p-value: 0.04072
```

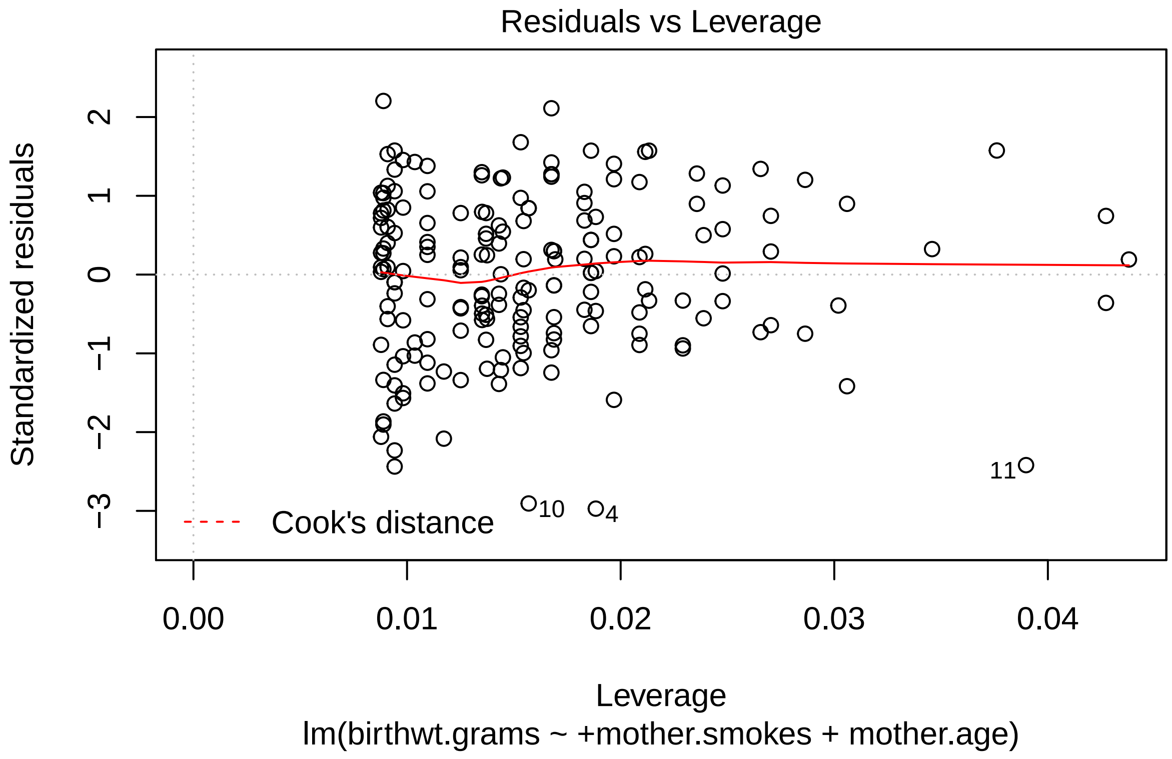
More complex models

```
plot(linear.model.3a)
```


Residuals vs Fitted
Residuals
2000
1000
0
−2000
2750
2800
2850
2900
2950
3000
3050
4
13
10
Fitted values
lm(birthwt.grams ~ +mother.smokes + mother.age)


Normal Q−Q
Standardized residuals
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3
13
4
10
Theoretical Quantiles
lm(birthwt.grams ~ +mother.smokes + mother.age)


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2750
2800
2850
2900
2950
3000
3050
Fitted values
lm(birthwt.grams ~ +mother.smokes + mother.age)
4
10
13



More complex models

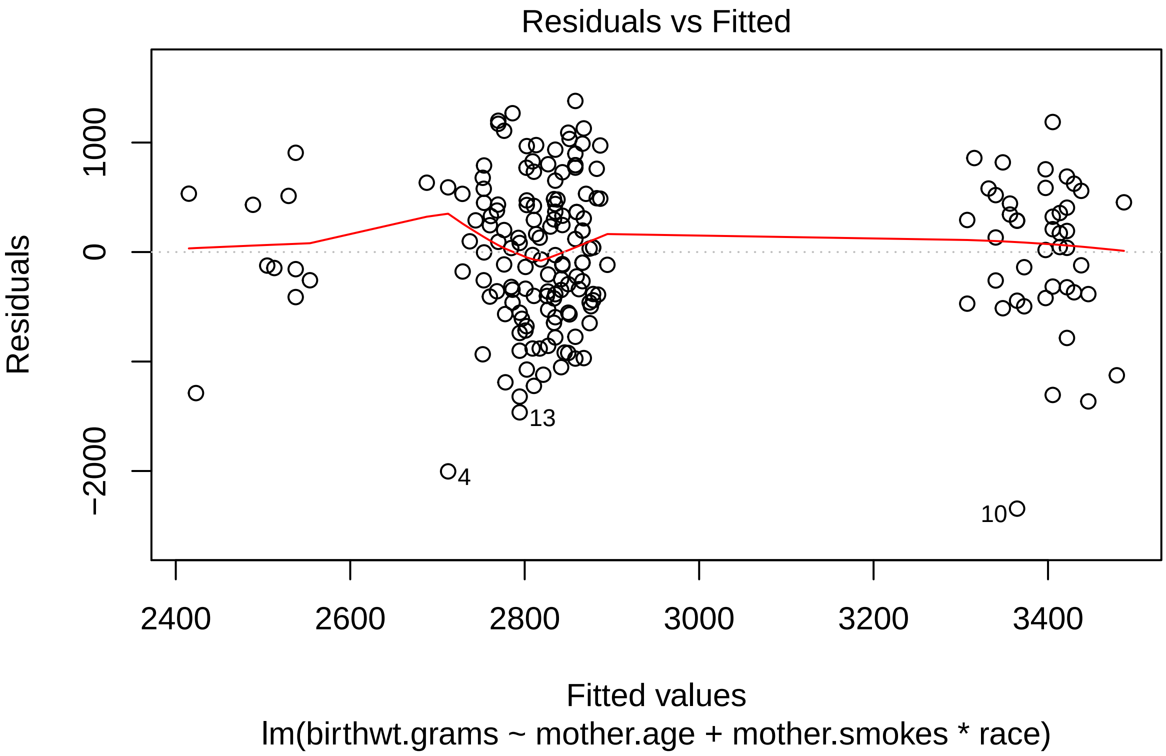
Add in race:

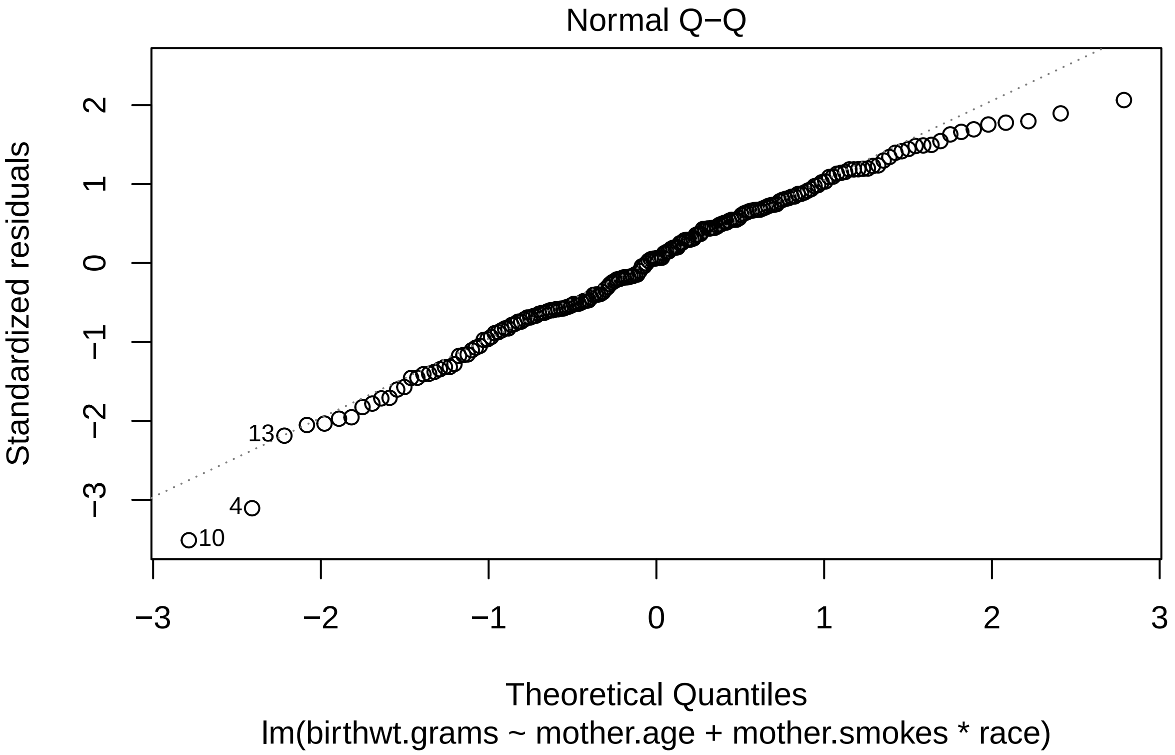
```
linear.model.3b <- lm (birthwt.grams ~ mother.age + mother.smokes*race, data=birthwt.noout)
summary(linear.model.3b)
```

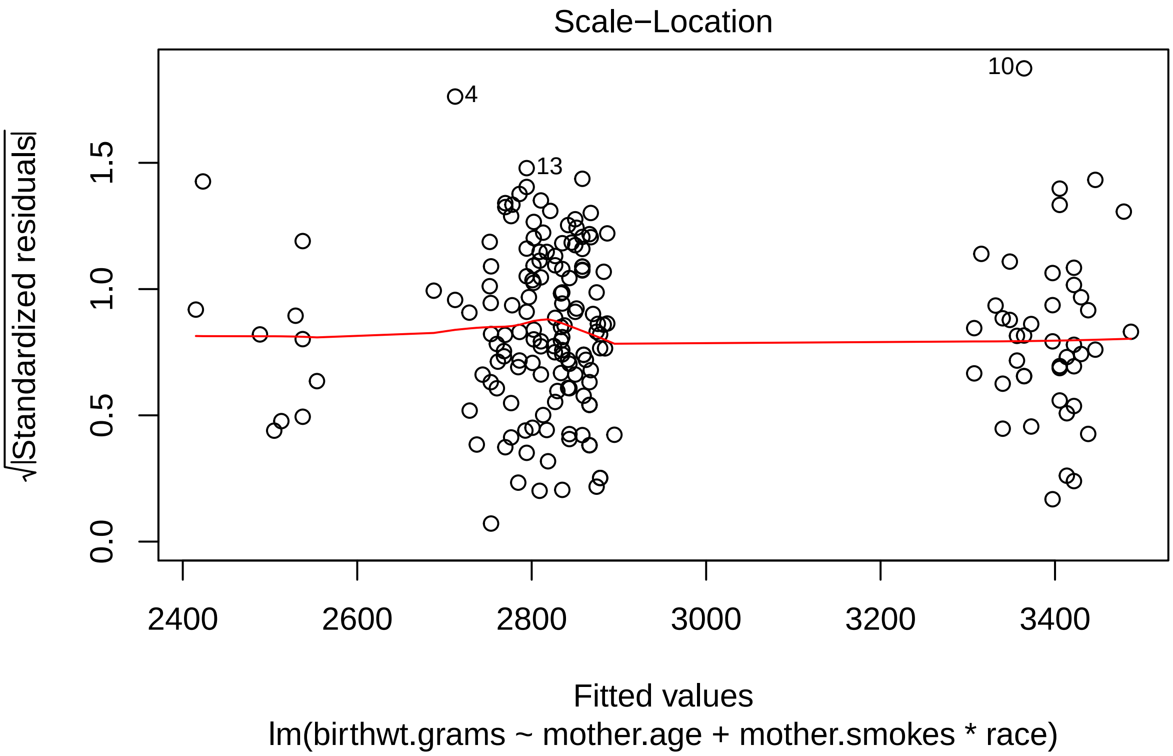
```
##
## Call:
## lm(formula = birthwt.grams ~ mother.age + mother.smokes * race,
##     data = birthwt.noout)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2343.52  -413.66    39.91   480.36  1379.90
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 3017.352    265.606  11.360  < 2e-16 ***
## mother.age                    -8.168     10.276  -0.795  0.42772
## mother.smokesYes            -316.500    275.896  -1.147  0.25282
## raceother                    -18.901    193.665  -0.098  0.92236
## racewhite                    584.042    206.320   2.831  0.00517 **
## mother.smokesYes:raceother   258.999    349.871   0.740  0.46010
## mother.smokesYes:racewhite  -271.594    314.268  -0.864  0.38862
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 676.1 on 181 degrees of freedom
## Multiple R-squared:  0.1359, Adjusted R-squared:  0.1073
## F-statistic: 4.746 on 6 and 181 DF,  p-value: 0.0001625
```

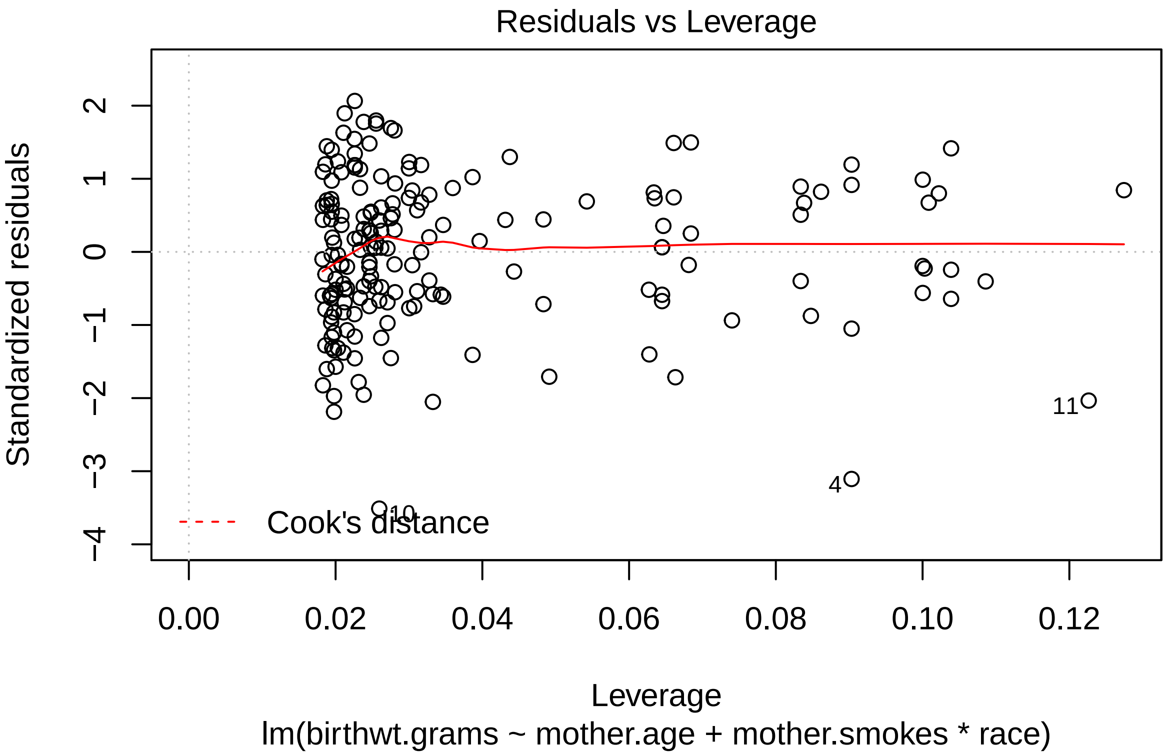
More complex models

```r
plot(linear.model.3b)
```


Residuals vs Fitted
Residuals
1000
0
−2000
2400
2600
2800
3000
3200
3400
13
4
10
Fitted values
lm(birthwt.grams ~ mother.age + mother.smokes * race)


Normal Q−Q
Standardized residuals
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3
13
4
10
Theoretical Quantiles
lm(birthwt.grams ~ mother.age + mother.smokes * race)


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2400
2600
2800
3000
3200
3400
Fitted values
lm(birthwt.grams ~ mother.age + mother.smokes * race)
4
10
13



Including everything

Let's include everything on this new data set:

```
linear.model.4 <- lm (birthwt.grams ~ ., data=birthwt.noout)
linear.model.4
```

```
##
## Call:
## lm(formula = birthwt.grams ~ ., data = birthwt.noout)
##
## Coefficients:
##         (Intercept)   birthwt.below.2500           mother.age
##           3360.5163           -1116.3933             -16.0321
##       mother.weight            raceother            racewhite
##              1.9317              68.8145             247.0241
##    mother.smokesYes  previous.prem.labor      hypertensionYes
##           -157.7041              95.9825            -185.2778
##      uterine.irrYes     physician.visits
##           -340.0918              -0.3519
```

Including everything

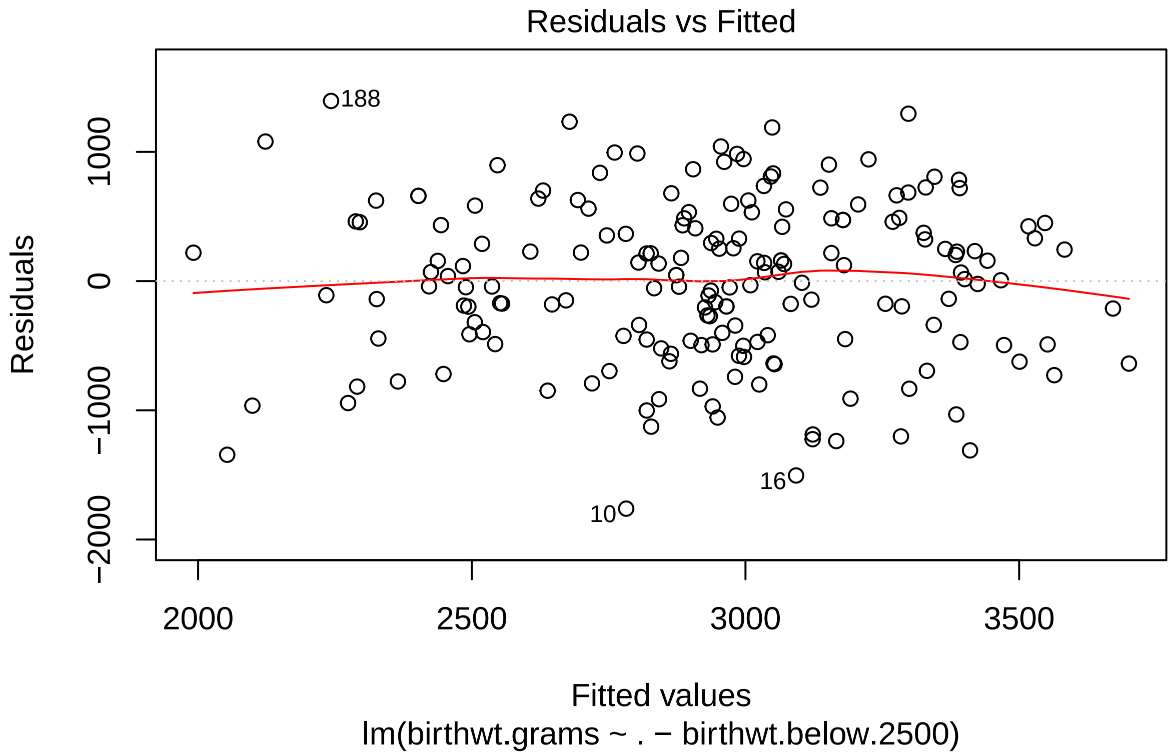
Be careful! One of those variables `birthwt.below.2500` is a function of the outcome.

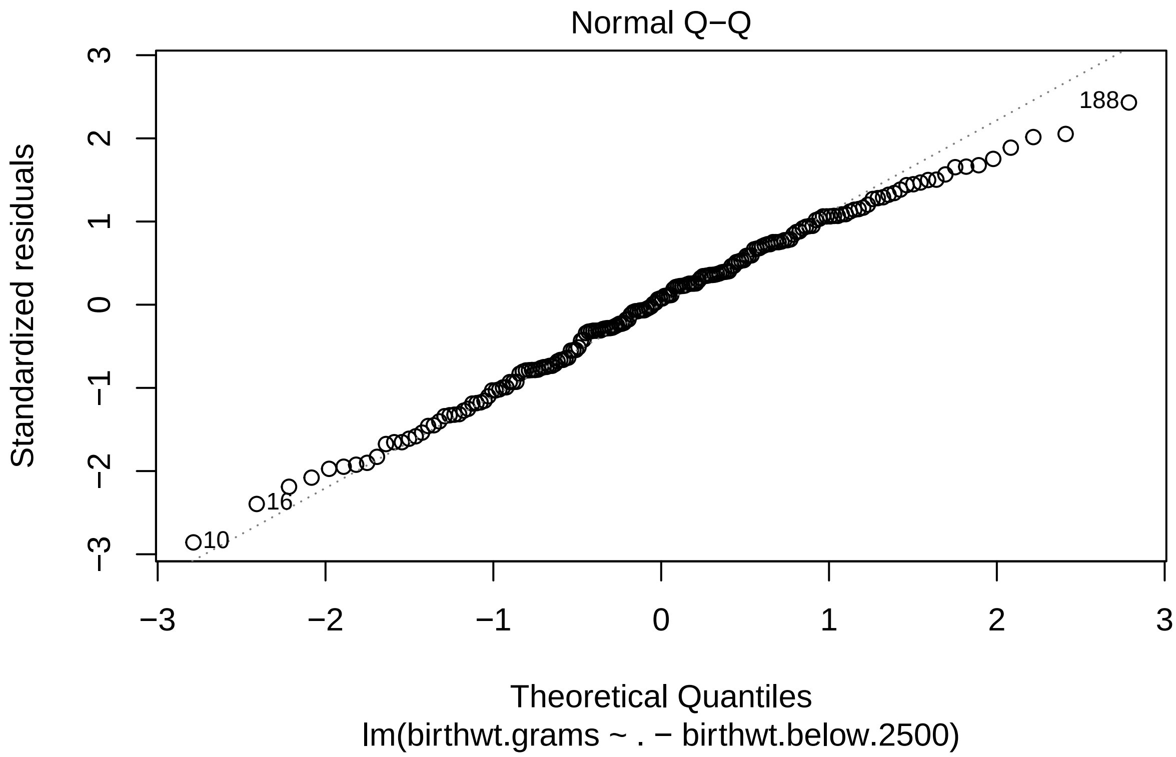
```r
linear.model.4a <- lm (birthwt.grams ~ . - birthwt.below.2500, data=birthwt.noout)
summary(linear.model.4a)
```

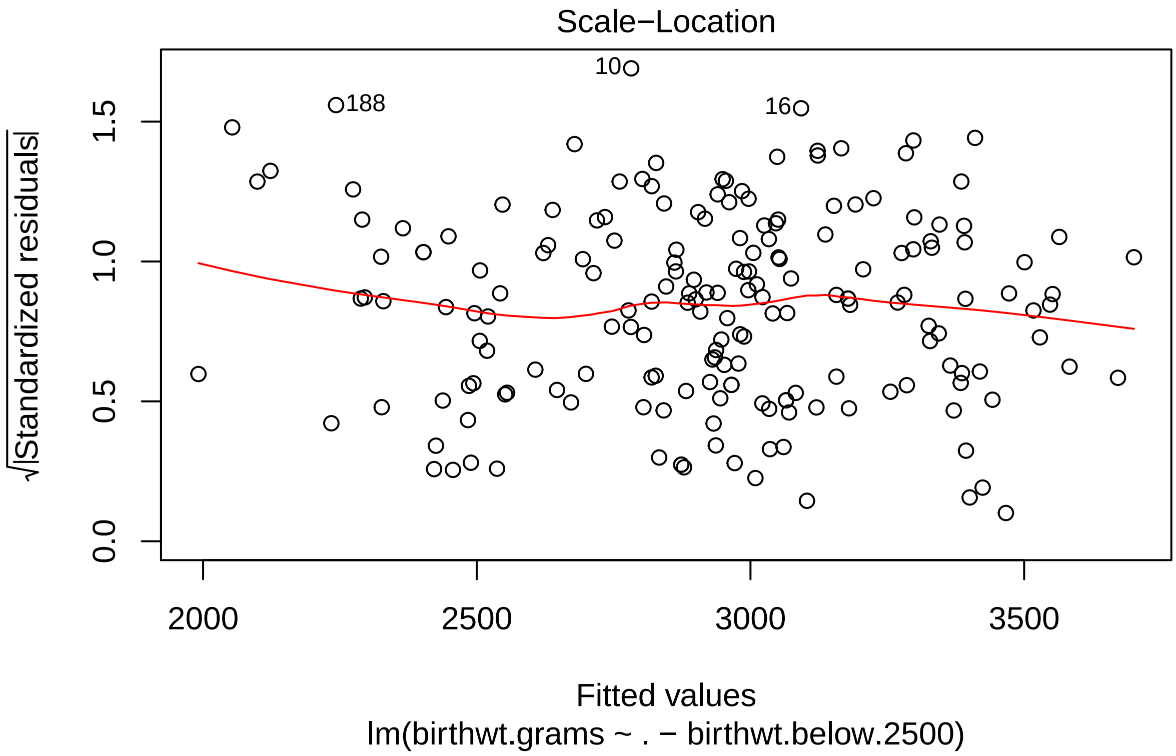
```
##
## Call:
## lm(formula = birthwt.grams ~ . - birthwt.below.2500, data = birthwt.noout)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1761.10  -454.81    46.43   459.78  1394.13
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         2545.584    323.204   7.876 3.21e-13 ***
## mother.age           -12.111      9.909  -1.222 0.223243
## mother.weight          4.789      1.710   2.801 0.005656 **
## raceother            155.605    156.564   0.994 0.321634
## racewhite            494.545    147.153   3.361 0.000951 ***
## mother.smokesYes    -335.793    104.613  -3.210 0.001576 **
## previous.prem.labor  -32.922    100.185  -0.329 0.742838
## hypertensionYes     -594.324    198.480  -2.994 0.003142 **
## uterine.irrYes      -514.842    136.249  -3.779 0.000215 ***
## physician.visits      -7.247     45.649  -0.159 0.874036
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 638 on 178 degrees of freedom
## Multiple R-squared:  0.2435, Adjusted R-squared:  0.2052
## F-statistic: 6.365 on 9 and 178 DF,  p-value: 8.255e-08
```

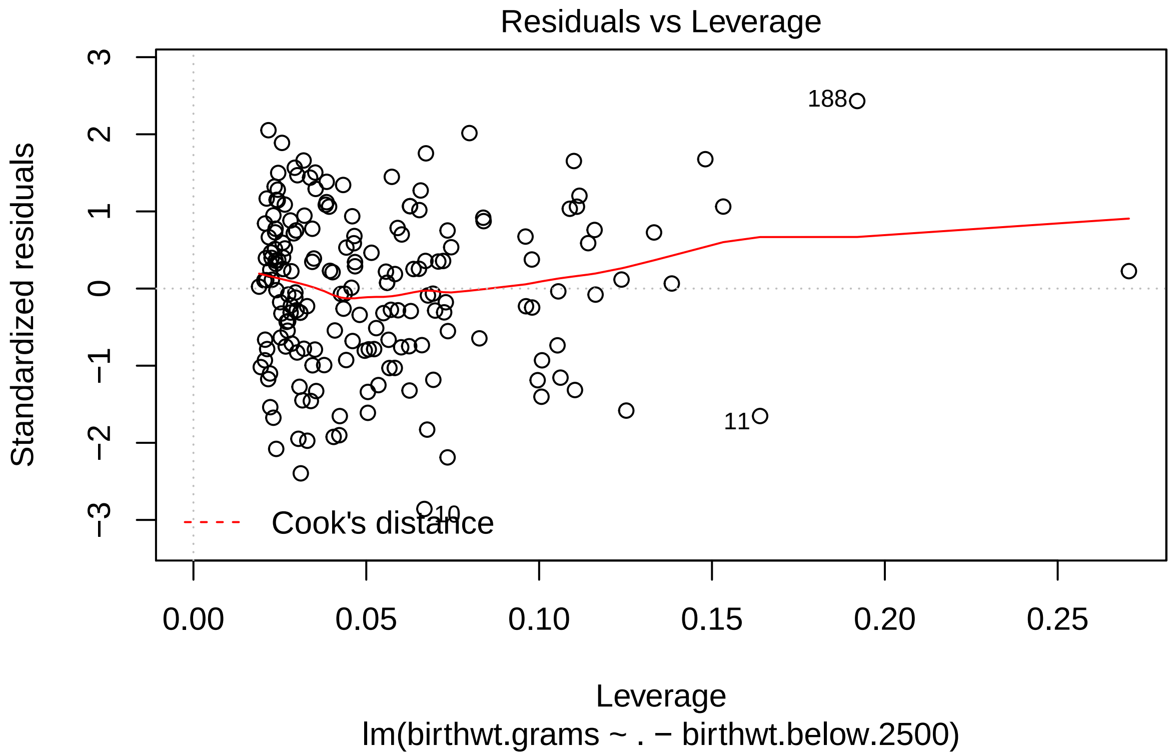
Including everything

```r
plot(linear.model.4a)
```


Residuals vs Fitted
Residuals
1000
0
−1000
−2000
2000
2500
3000
3500
188
16
10
Fitted values
lm(birthwt.grams ~ . − birthwt.below.2500)


Normal Q−Q
Standardized residuals
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3
188
16
10
Theoretical Quantiles
lm(birthwt.grams ~ . − birthwt.below.2500)


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2000
2500
3000
3500
Fitted values
lm(birthwt.grams ~ . − birthwt.below.2500)
10
188
16



Why?

Let's take a subset of this data to do predictions.

```
odds <- seq(1, nrow(birthwt.noout), by=2)
birthwt.in <- birthwt.noout[odds,]
birthwt.out <- birthwt.noout[-odds,]
linear.model.half <-
  lm (birthwt.grams ~
      . - birthwt.below.2500, data=birthwt.in)
```

Why?

```
summary (linear.model.half)
```

```
##
## Call:
## lm(formula = birthwt.grams ~ . - birthwt.below.2500, data = birthwt.in)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1705.17  -303.11    26.48   427.18  1261.57
##
## Coefficients:
```

```
##                     Estimate Std. Error t value Pr(>|t|)    
## (Intercept)         2514.891    450.245   5.586 2.81e-07 ***
## mother.age             7.052     14.935   0.472  0.63801    
## mother.weight          2.683      2.885   0.930  0.35501    
## raceother            113.948    224.519   0.508  0.61312    
## racewhite            466.219    204.967   2.275  0.02548 *  
## mother.smokesYes    -217.218    154.521  -1.406  0.16349    
## previous.prem.labor -206.093    143.726  -1.434  0.15530    
## hypertensionYes     -653.594    281.795  -2.319  0.02280 *  
## uterine.irrYes      -547.884    193.386  -2.833  0.00577 ** 
## physician.visits    -130.202     81.400  -1.600  0.11346    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 643.7 on 84 degrees of freedom
## Multiple R-squared:  0.2585, Adjusted R-squared:  0.1791 
## F-statistic: 3.254 on 9 and 84 DF,  p-value: 0.001942
```

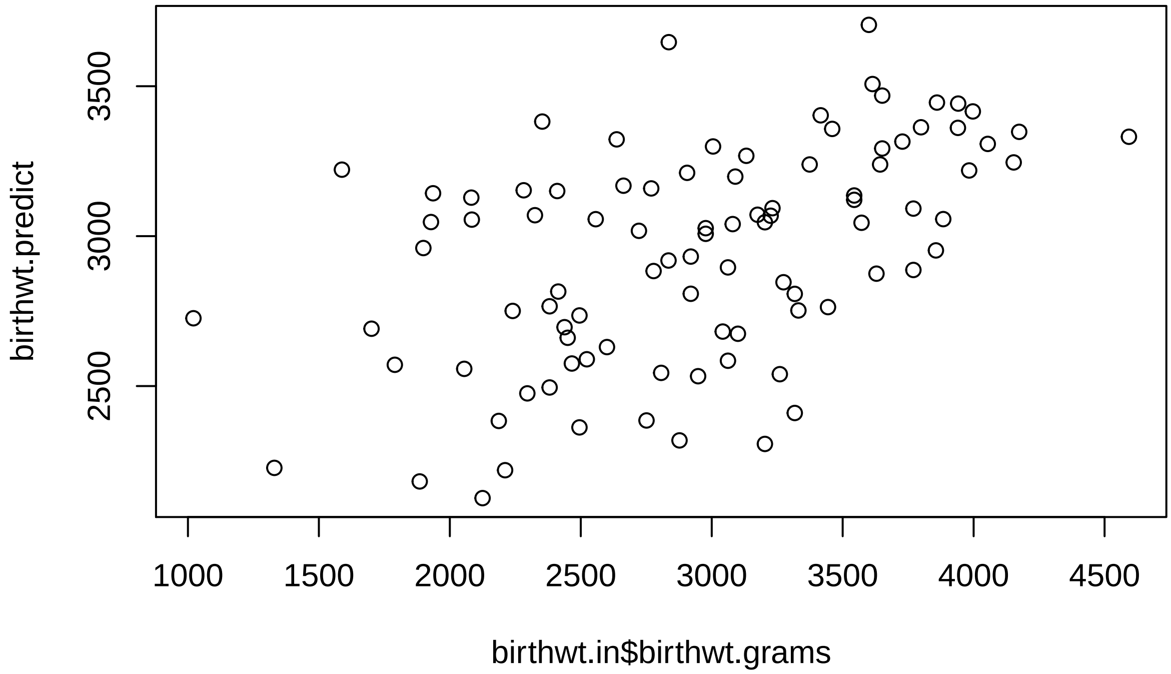
Prediction of Training Data

```
birthwt.predict <- predict (linear.model.half)
cor (birthwt.in$birthwt.grams, birthwt.predict)
```

```
## [1] 0.508442
```

Prediction of Training Data

```
plot (birthwt.in$birthwt.grams, birthwt.predict)
```



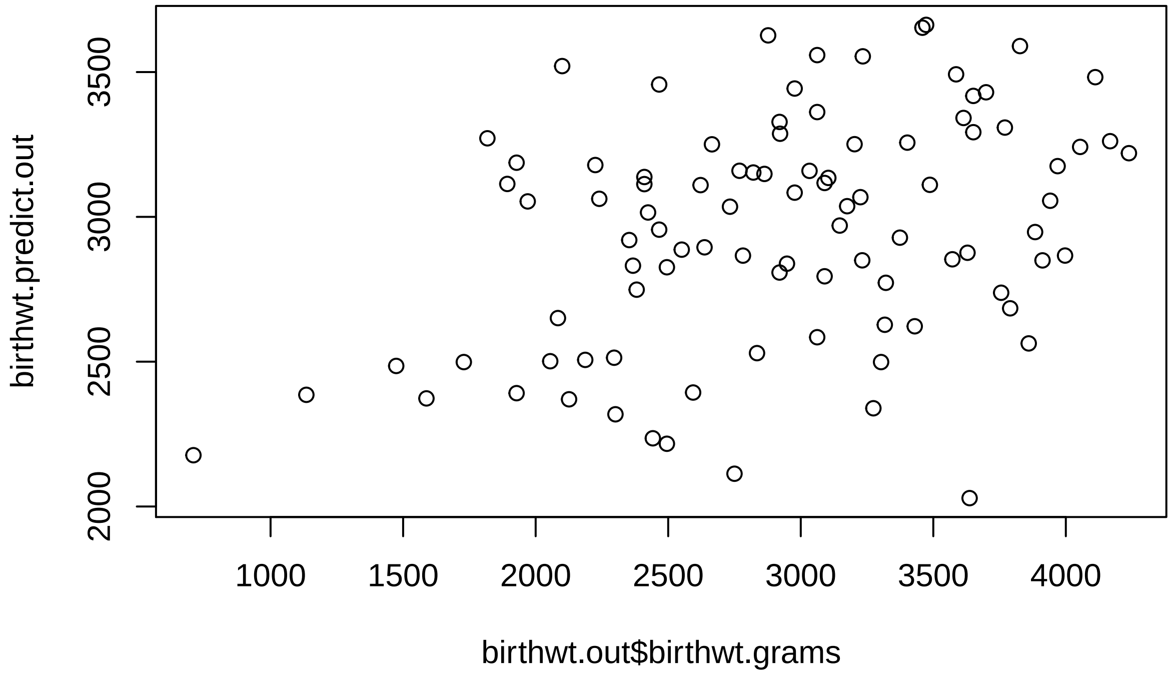
Prediction of Test Data

```
birthwt.predict.out <- predict (linear.model.half, birthwt.out)
cor (birthwt.out$birthwt.grams, birthwt.predict.out)
```

```
## [1] 0.3749431
```

Prediction of Test Data

```
plot (birthwt.out$birthwt.grams, birthwt.predict.out)
```



Transformations

You go to analysis with the data you have, not the data you want.

The variables in the data are often either not what's most relevant to the analysis, or they're not arranged conveniently, or both

Satisfying model assumptions is a big issue here

∴ often want to *transform* the data to make it closer to the data we wish we had to start with

Some Common Transformations of Numerical Data

- `log`:
  - Because $Y = f(X)g(Z) \Leftrightarrow \log Y = \log f(X) + \log g(X)$, taking logs lets us use linear or additive models when the real relationship is multiplicative
  - Taking logs let the data satisfy assumption on distribution, see, `t.test`.
  - How would you take the `log` of a whole column?

Numerical Transformations (cont'd.)

- Z-scores, centering and scaling:

```
head(scale(cats[,-1],center=TRUE,scale=TRUE))
```

```
##         Bwt        Hwt
## 1 -1.491039 -1.4912110
## 2 -1.491039 -1.3269153
## 3 -1.491039 -0.4643633
## 4 -1.284984 -1.4090631
## 5 -1.284984 -1.3679892
## 6 -1.284984 -1.2447675
```

- `center=TRUE` ⇒ subtract the mean; alternately, `FALSE` or a vector
- `scale=TRUE` ⇒ divide by standard deviation, after centering; same options
  - Defaults in `scale` produce "Z-scores"

Numerical Transformations (cont'd.)

- Successive differences: `diff(x)`; differences between `x[t]` and `x[t-k]`, `diff(x,lag=k)`
  - Vectorizes over columns of a matrix
- Cumulative totals etc.: `cumsum`, `cumprod`, `cummax`, `cummin`
  - Exercise: write `cummean`
- Rolling means: `rollmean` from the `zoo` package; s
  - See also `rollapply`

Numerical Transformations (cont'd.)

- Magnitudes to ranks: `rank(x)` outputs the **rank** of each element of `x` within the vector, 1 being the smallest:

```
head(cats$Hwt)
```

```
## [1] 7.0 7.4 9.5 7.2 7.3 7.6
```

```
head(rank(cats$Hwt))
```

```
## [1]  4.0 11.0 50.5  6.5  9.0 12.5
```

Numerical Transformations (cont'd.)

- "Para-normal" values: Based on the percentile, where would this be if it were Gaussian/normal?

```
qnorm(ecdf(x)(x),mean=100,sd=15)
```

- Obviously nothing magic about using `qnorm` there
- This is how IQ tests are scored; raw scores are highly skewed and don't follow bell curves at all
- "Gaussian copula" = run this trick on two or more variables and then measure the correlations

name due to L. Wasserman

Numerical Transformations (cont'd.)

- Extracting deviations from a trend
  - Calculate the predicted value per trend
  - Take the difference

```
gdp_trend <- gdp[1]*exp(growth.rate*(0:length(gdp)-1))
gdp_vs_trend <- gdp/gdp_trend
```

- Use `residuals` when the trend is a regression model:

```
head(residuals(lm(Hwt ~ Bwt, data=cats)))
```

```
##          1          2          3          4          5          6 
## -0.7114630 -0.3114630  1.7885370 -0.9148692 -0.8148692 -0.5148692
```

Summarizing Subsets

- `aggregate` takes a dataframe, a *list* containing the variable(s) to group the rows **by**, and a *scalar* -valued summarizing function:

```
aggregate(cats[,-1],by=cats[1],mean)
```

```
##   Sex      Bwt       Hwt
## 1   F 2.359574  9.202128
## 2   M 2.900000 11.322680
```

Note: No comma in `cats[1]`; treating dataframe as a list of vectors - Each vector in the `by` list must be as long as the number of rows of the data

Summarizing Subsets (cont'd.)

- `aggregate` doesn't work on vectors, but it has a cousin, `tapply`:

```
tapply(cats$Hwt,INDEX=cats$Sex,max)
```

```
##    F    M 
## 13.0 20.5
```

- tapply can return more than just a scalar value:

```
tapply(cats$Hwt,cats$Sex,summary)
```

```
## $F
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   6.300   8.350   9.100   9.202  10.100  13.000 
## 
## $M
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    6.50    9.40   11.40   11.32   12.80   20.50
```

Re-Organizing

- Even if the numbers (or strings, etc.) are fine, they may not be arranged very conveniently
- Lots of data manipulation involves re-arrangement:
  - sorting arrays and dataframes by certain columns
  - merging dataframes
  - Turning short, wide dataframes into long, narrow ones, and vice versa

Re-Ordering

`order` takes in a vector, and returns the vector of indices which would put it in order (increasing by default) - Use the `decreasing=TRUE` option to change that - Output of `order` can be saved to re-order multiple dataframes the same way

order (cont'd.)

```
head(cats,4)
```

```
##   Sex Bwt Hwt
## 1   F 2.0 7.0
## 2   F 2.0 7.4
## 3   F 2.0 9.5
## 4   F 2.1 7.2
```

```
head(order(cats$Hwt))
```

```
## [1] 31 48 49  1 13  4
```

```
head(cats[order(cats$Hwt),],4)
```

```
##    Sex Bwt Hwt
## 31   F 2.4 6.3
## 48   M 2.0 6.5
## 49   M 2.0 6.5
## 1    F 2.0 7.0
```

Related to order

- `rank(x)` does *not* deliver the same thing as `order(x)`!
- `sort` returns the sorted vector, not the ordering

```
head(sort(cats$Hwt))
```

```
## [1] 6.3 6.5 6.5 7.0 7.1 7.2
```

- To just get the index of the smallest or largest element, use `which.min` or `which.max`

```
which.min(cats$Hwt) == order(cats$Hwt)[1]
```

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

What if you want the position of the smallest element in a matrix?

```
set.seed(20190920)
mat <- matrix(rnorm(40), 10, 4)
which(mat == min(mat, na.rm=TRUE), arr.ind = TRUE)
```

```
##      row col
## [1,]   7   3
```

Merging Dataframes

You have two dataframes, say `movies.info` and `movies.biz`, and you want to combine them into one dataframe, say `movies`

- Simplest case: the dataframes have exactly the same number of rows, that the rows represent exactly the same units, and you want all columns from both

```
movies <- data.frame(movies.info, movies.biz)
```

- Next best case: you know that the two dataframes have the same rows, but you only want certain columns from each

```
movies <- data.frame(year = movies.info$year,
                     avg_rating = movies.info$avg_rating,
                     num_rates = movies.info$num_raters,
                     genre = movies.info$genre,
                     gross = movies.biz$gross)
```

Merging Dataframes (cont'd.)

- Next best case: same number of rows but in different order
    - Put one of them in the same order as the other
    - Use `merge`
- Worse cases: different numbers of rows…
    - Cleverer re-ordering tricks
    - Use `merge`

An Example

Claim: People in larger cities travel more

More precise claim: miles driven per person per day increases with the area of the city

Example of Merging (cont'd.)

Distance driven, and city population: table HM-71 in the 2011 "Highway Statistics Series"

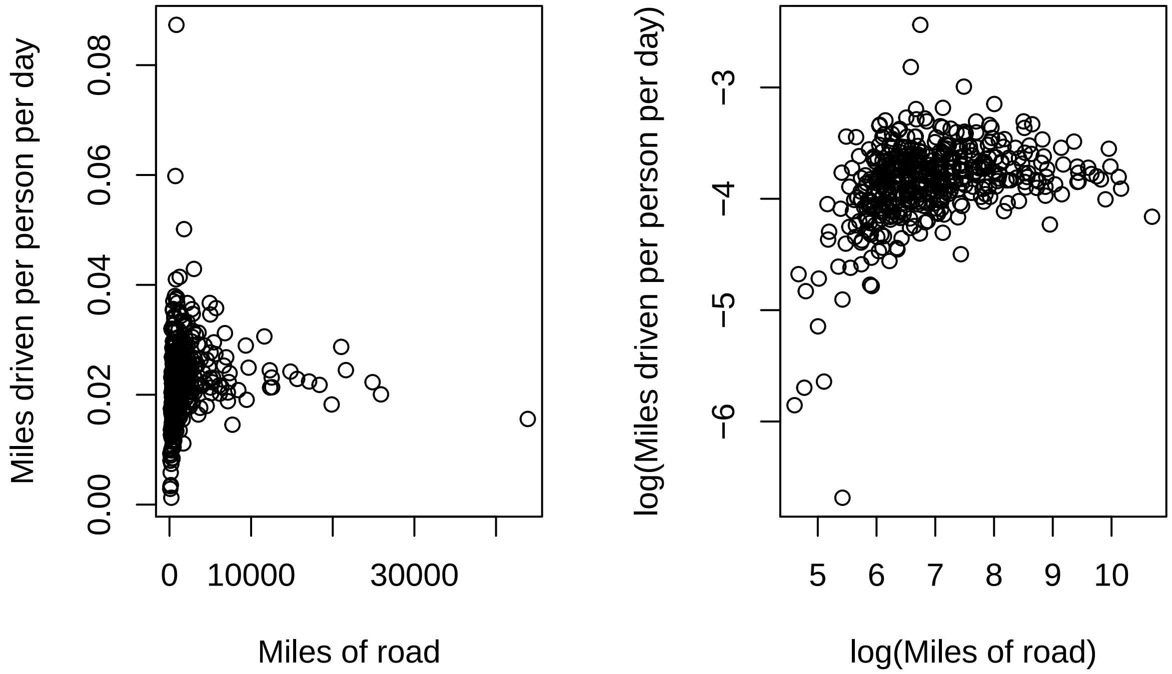
```
fha <- read.csv("data/fha.csv", na.strings = "NA",
                colClasses = c("character","double","double","double"))
nrow(fha)
```

```
## [1] 498
```

```
colnames(fha)
```

```
## [1] "City"                 "Population"           "Miles.of.Road"
## [4] "Daily.Miles.Traveled"
```

```
op <- par(mfrow=c(1,2))
plot(fha$Miles.of.Road, fha$Daily.Miles.Traveled/fha$Population, ylab="Miles driven per person per day"
plot(log(fha$Miles.of.Road), log(fha$Daily.Miles.Traveled/fha$Population), ylab="log(Miles driven per p
```



```r
par(op)
```

Example of Merging (cont'd.)

Area and population of "urbanized areas":

```r
ua <- read.csv("data/ua.txt", sep = ";")
nrow(ua)
```

```
## [1] 3598
```

```r
colnames(ua)
```

```
##  [1] "UACE"          "NAME"          "POP"           "HU"
##  [5] "AREALAND"      "AREALANDSQMI"  "AREAWATER"     "AREAWATERSQMI"
##  [9] "POPDEN"        "LSADC"
```

Example of Merging (cont'd.)

This isn't a simple case, because:

1. ≈ 500 cities vs. ≈ 4000 "urbanized areas"
2. `fha` orders cities by population, `ua` is alphabetical by name
3. Both have place-names, but those don't always agree
4. Not even common names for the shared columns

But both use the same Census figures for population, and it turns out every settlement (in the top 498) has a unique Census population:

```
length(unique(fha$Population)) == nrow(fha)
```

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

```
identical(fha$Population,sort(ua$POP,decreasing=TRUE)[1:nrow(fha)]) # Why?
```

```
## [1] FALSE
```

```
summary(fha$Population - sort(ua$POP,decreasing=TRUE)[1:nrow(fha)])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##       0       0       0       0       0       0
```

```
class(sort(ua$POP,decreasing=TRUE)[1:nrow(fha)])
```

```
## [1] "integer"
```

```
class(fha$Population)
```

```
## [1] "numeric"
```

```
identical(fha$Population,as.numeric(sort(ua$POP,decreasing=TRUE)[1:nrow(fha)]))
```

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

Example of Merging (cont'd.)

Option 1: re-order the 2nd table by population

```
ua <- ua[order(ua$POP,decreasing=TRUE),]
df1 <- data.frame(fha, area=ua$AREALANDSQMI[1:nrow(fha)])
# Neaten up names
colnames(df1) <- c("City","Population","Roads","Mileage","Area")
nrow(df1)
```

```
## [1] 498
```

```
head(df1)
```

```
##                                    City Population Roads Mileage    Area
## 1          New York--Newark, NY--NJ--CT   18351295 43893  286101 3450.20
## 2 Los Angeles--Long Beach--Anaheim, CA   12150996 24877  270807 1736.02
## 3                     Chicago, IL--IN    8608208 25905  172708 2442.75
## 4                           Miami, FL    5502379 15641  125899 1238.61
## 5          Philadelphia, PA--NJ--DE--MD    5441567 19867   99190 1981.37
## 6    Dallas--Fort Worth--Arlington, TX    5121892 21610  125389 1779.13
```

Example of Merging (cont'd.)

Option 2: Use the `merge` function

```
df2 <- merge(x=fha,y=ua,
            by.x="Population",by.y="POP")
nrow(df2)
```

```
## [1] 498
```

```
tail(df2,3)
```

```
##     Population                                 City Miles.of.Road
## 496    8608208                       Chicago, IL--IN         25905
## 497   12150996 Los Angeles--Long Beach--Anaheim, CA         24877
## 498   18351295             New York--Newark, NY--NJ--CT         43893
##     Daily.Miles.Traveled  UACE                                 NAME
## 496               172708 16264                      Chicago, IL--IN
## 497               270807 51445 Los Angeles--Long Beach--Anaheim, CA
## 498               286101 63217         New York--Newark, NY--NJ--CT
##          HU   AREALAND AREALANDSQMI AREAWATER AREAWATERSQMI POPDEN LSADC
## 496 3459257 6326686332      2442.75 105649916         40.79 3524.0    75
## 497 4217448 4496266014      1736.02  61141327         23.61 6999.3    75
## 498 7263095 8935981360      3450.20 533176599        205.86 5318.9    75
```

Example of Merging (cont'd.)

- `by.x` and `by.y` say which columns need to match to do a merge
  - Default: merge on all columns with shared names
- New dataframe has *all* the columns of *both* dataframes
  - Here, should really delete the ones we don't need and tidy `colnames`

Example of Merging (cont'd.)

You'd think merging on names would be easy…

```
df2.1 <- merge(x=fha,y=ua,by.x="City", by.y="NAME")
nrow(df2.1)
```

```
## [1] 492
```

We can force unmatched rows of either dataframe to be included, with NA values as appropriate:

```
df2.2 <- merge(x=fha,y=ua,by.x="City",by.y="NAME",all.x=TRUE)
nrow(df2.2)
```

```
## [1] 498
```

Example of Merging (cont'd.)

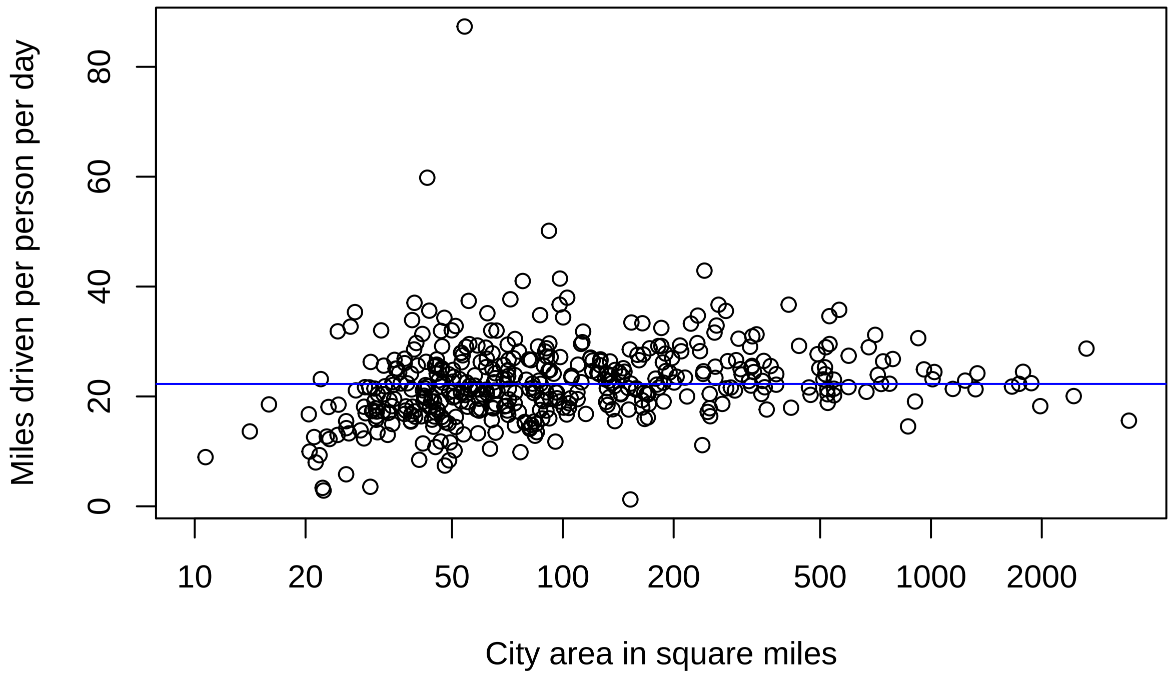
Where are the mis-matches?

```
df2.2$City[is.na(df2.2$POP)]
```

```
## [1] "Aguadilla--Isabela--San Sebastian, PR"
## [2] "Danville, VA -- NC"
## [3] "Florida--Imbery--Barceloneta, PR"
## [4] "Juana Diaz, PR"
## [5] "Mayaguez, PR"
## [6] "San German--Cabo Rojo--Sabana Grande, PR"
```

On investigation, `fha.csv` and `ua.txt` use 2 different encodings for accent characters, and one writes things like `VA -- NC` and the other says `VA--NC`

```
# Convert 1,000s of miles to miles
df1$Mileage <- 1000*df1$Mileage
# Plot daily miles per person vs. area
plot(Mileage/Population ~ Area, data=df1, log="x",
     ylab="Miles driven per person per day",
     xlab="City area in square miles")
# Impressively flat regression line
abline(lm(Mileage/Population~Area,data=df1),col="blue")
```



Using order+data.frame vs. merge

- Re-ordering is easier to grasp; `merge` takes some learning
- Re-ordering is simplest when there's only one column to merge on; `merge` handles many columns
- Re-orderng is simplest when the dataframes are the same size; `merge` handles different sizes

Summary

- Loading and saving R objects is very easy
- Reading and writing dataframes is pretty easy
- Linear models are very easy via `lm()`
- Numerical transformations
- Re-ordering dataframes
- Merging dataframes with `merge`