waterByCity

Purpose

- The purpose here is to understand water production by city and population served using HB-1051 data. Data was obtained from CWCB Water Efficiency Data Portal (http://cowaterefficiency.com/unauthenticated_home) All years and water providers were retrieved

Read in data

- Files used:
    - water_use_02_production.csv (HB 1051 Data (http://cowaterefficiency.com/unauthenticated_home))
    - overview.csv (HB 1051 Data (http://cowaterefficiency.com/unauthenticated_home))
    - normalizing_03_population.csv (HB 1051 Data (http://cowaterefficiency.com/unauthenticated_home))
    - historical-census-data-counties.csv (Historical Census Data (https://demography.dola.colorado.gov/population/data/historical_census/))
    - historical-census-data-municipalities.csv (Historical Census Data (https://demography.dola.colorado.gov/population/data/historical_census/))

```
production_filepath = "/Users/dsciacca/Documents/Data Analytics Challenge/Water
Efficiency Data - CWCB/water_use/water_use_02_production.csv"
overview_filepath = "/Users/dsciacca/Documents/Data Analytics Challenge/Water E
fficiency Data - CWCB/overview/overview.csv"
pop_filepath = "/Users/dsciacca/Documents/Data Analytics Challenge/Water Effici
ency Data - CWCB/normalizing/normalizing_03_population.csv"
water_production = read.csv(production_filepath, header = TRUE, stringsAsFactor
s = FALSE)
water_overview = read.csv(overview_filepath, header = TRUE, stringsAsFactors =
FALSE)
water_pop = read.csv(pop_filepath, header = TRUE, stringsAsFactors = FALSE)
```

Clean up water_production file

Convert all units to million gallons

```
#first place all dif measurments in separate dfs
water_prod_acre = sqldf("SELECT * FROM water_production WHERE units = 'Acre-fee
t'")
water_prod_ccf = sqldf("SELECT * FROM water_production WHERE units = 'CCF (100
cubic feet)'")
water_prod_milgal = sqldf("SELECT * FROM water_production WHERE units = 'Gallon
s, Millions'")
water_prod_thougal = sqldf("SELECT * FROM water_production WHERE units = 'Gallo
ns, Thousands'")
water_prod_gal = sqldf("SELECT * FROM water_production WHERE units = 'Gallons'"
)

#1 acre-foot = 325851 gal = 0.325851 mil gal
water_prod_acre$produced = water_prod_acre$produced * 0.325851
water_prod_acre$exported = water_prod_acre$exported * 0.325851
water_prod_acre$water_in_system = water_prod_acre$water_in_system * 0.325851

#1 ccf = 748.052 gal = 0.000748052 mil gal
water_prod_ccf$produced = water_prod_ccf$produced * 0.000748052
water_prod_ccf$exported = water_prod_ccf$exported * 0.000748052
water_prod_ccf$water_in_system = water_prod_ccf$water_in_system * 0.000748052

#thousand gal to mil gal
water_prod_thougal$produced = water_prod_thougal$produced * 0.001
water_prod_thougal$exported = water_prod_thougal$exported * 0.001
water_prod_thougal$water_in_system = water_prod_thougal$water_in_system * 0.001

#gal to mil gal
water_prod_gal$produced = water_prod_gal$produced / 1000000
water_prod_gal$exported = water_prod_gal$exported / 1000000
water_prod_gal$water_in_system = water_prod_gal$water_in_system / 1000000

#recombine datasets - effectively drops unspecified and blank unit rows, 8 tota
l
water_prod = rbind(water_prod_gal, water_prod_milgal, water_prod_thougal, water
_prod_acre, water_prod_ccf)

#drop units column
water_prod = water_prod[-c(7)]
```

Merge datasets

First merge water production with overview on ce_annual_ndx to get city, county, and ce_index, etc. columns. Then, merge with population dataset, again on ce_annual_ndx to get population data

```
#merge production and overview
water_prod_city = merge(water_prod, water_overview, all.x = TRUE, by.x = "ce_an
nual_ndx", by.y = "ce_annual_ndx")

#merge with population dataset
water_prod_city_pop = merge(water_prod_city, water_pop, all.x = TRUE, by.x = "c
e_annual_ndx", by.y = "ce_annual_ndx")
```

Clean data

Drop extra columns, then remove rows with NA in produced column

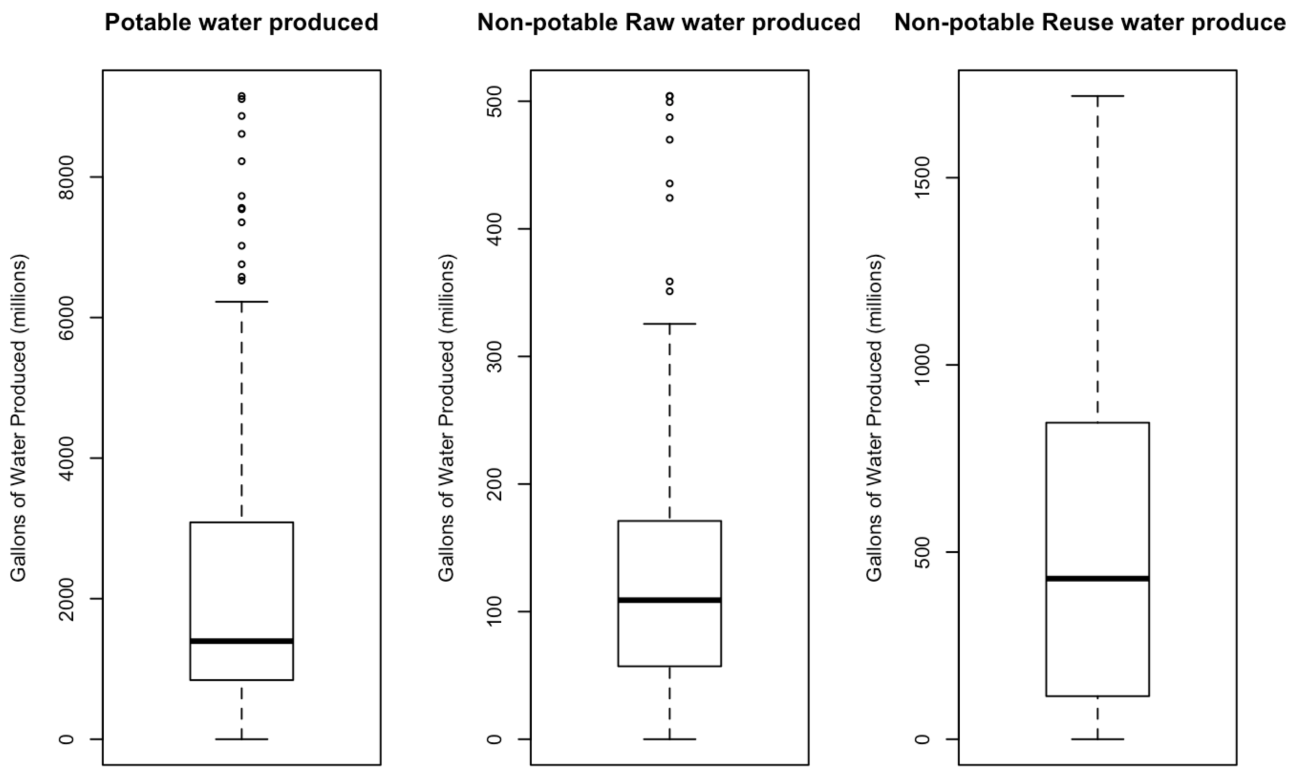
```
# Drop columns (remarks, pop sources, provider comments, transient pop descript
ion)
water_prod_city_pop = water_prod_city_pop[, -c(7, 15, 20, 22, 23, 24)]

water_prod_city_pop = as.data.table(water_prod_city_pop)
water_prod_city_pop = na.omit(water_prod_city_pop, cols = "produced")
water_prod_city_pop = as.data.frame(water_prod_city_pop)
```

Separate datasets into different water types and remove duplicate rows

```
water_prod_city_pop_potable = sqldf("SELECT * FROM water_prod_city_pop WHERE wa
ter_type_index = 1")
water_prod_city_pop_npraw = sqldf("SELECT * FROM water_prod_city_pop WHERE wate
r_type_index = 2")
water_prod_city_pop_npreuse = sqldf("SELECT * FROM water_prod_city_pop WHERE wa
ter_type_index = 3")

#remove duplicate rows - all but npreuse had extra rows, including just to be s
ure
water_prod_city_pop_potable = unique(water_prod_city_pop_potable)
water_prod_city_pop_npraw = unique(water_prod_city_pop_npraw)
water_prod_city_pop_npreuse = unique(water_prod_city_pop_npreuse)
```

Detect outliers to make visualization more useful

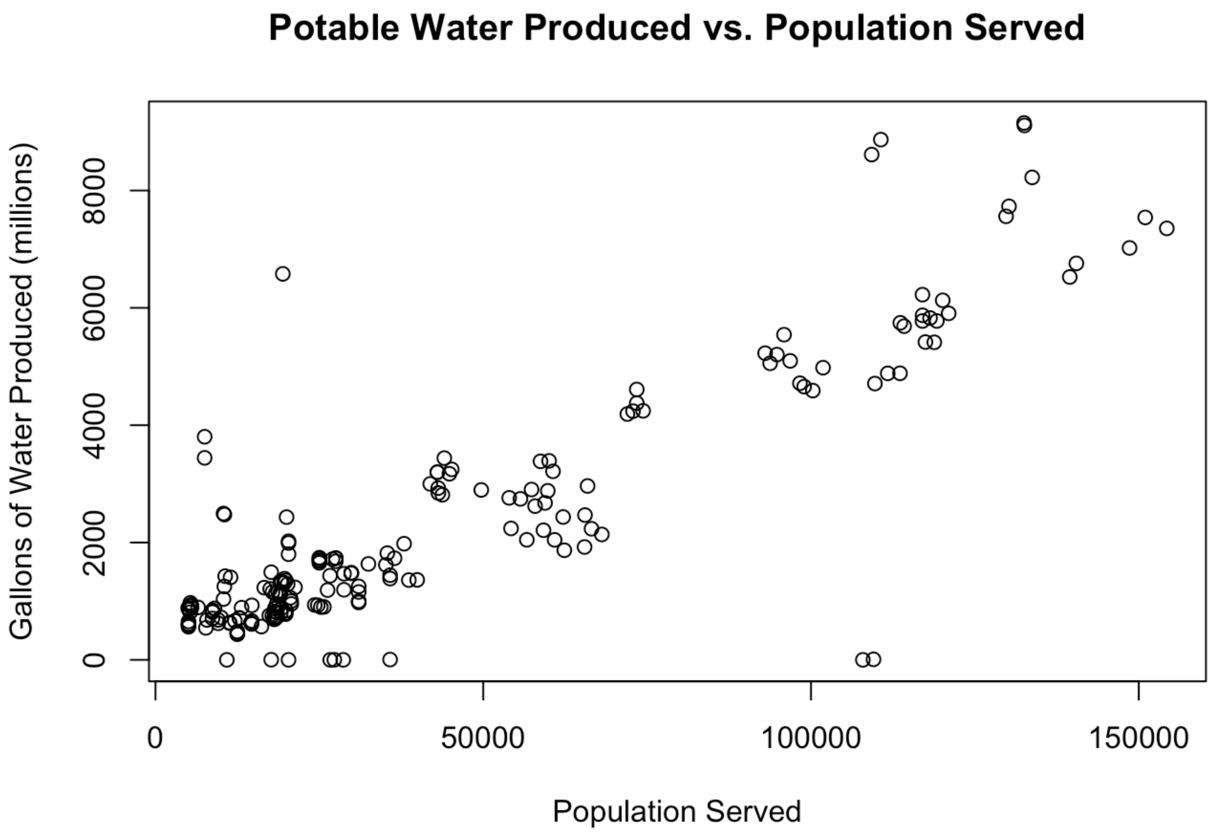
```
#first find outlier values
potable_outliers = boxplot.stats(water_prod_city_pop_potable$produced)$out
npraw_outliers = boxplot.stats(water_prod_city_pop_npraw$produced)$out
npreuse_outliers = boxplot.stats(water_prod_city_pop_npreuse$produced)$out

#obtain indeces of outliers in each dataset
potable_outlier_ndx = match(potable_outliers, water_prod_city_pop_potable$produ
ced)
npraw_outlier_ndx = match(npraw_outliers, water_prod_city_pop_npraw$produced)
npreuse_outlier_ndx = match(npreuse_outliers, water_prod_city_pop_npreuse$produ
ced)

#create boxplots with outliers removed
par(mfrow=c(1,3))
boxplot(water_prod_city_pop_potable$produced[-potable_outlier_ndx], main = "Pot
able water produced", ylab = "Gallons of Water Produced (millions)")
boxplot(water_prod_city_pop_npraw$produced[-npraw_outlier_ndx], main = "Non-pot
able Raw water produced", ylab = "Gallons of Water Produced (millions)")
boxplot(water_prod_city_pop_npreuse$produced[-npreuse_outlier_ndx], main = "Non
-potable Reuse water produced", ylab = "Gallons of Water Produced (millions)")
```



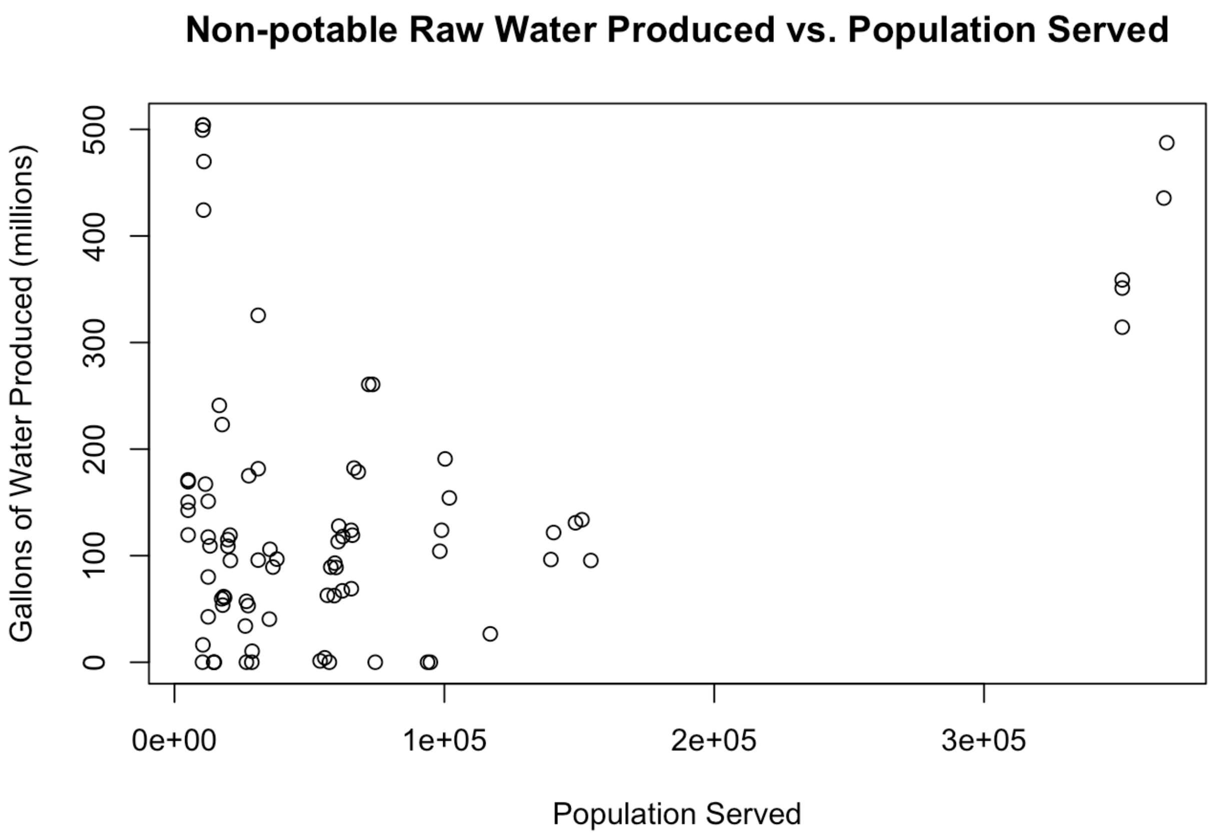
Plot water produced against population

served

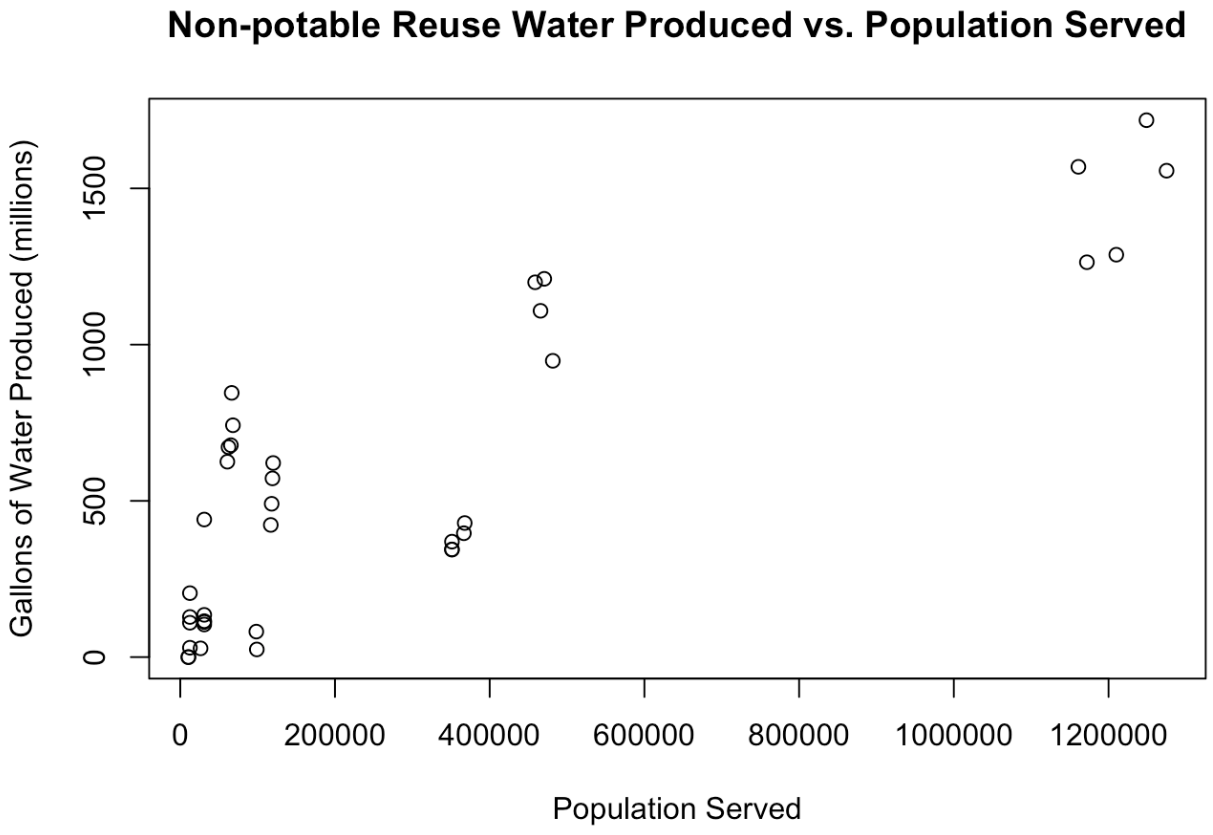
```
plot(water_prod_city_pop_potable$pop_served[-potable_outlier_ndx], water_prod_c
ity_pop_potable$produced[-potable_outlier_ndx], main = "Potable Water Produced
vs. Population Served", xlab = "Population Served", ylab = "Gallons of Water Pr
oduced (millions)")
```



```
plot(water_prod_city_pop_npraw$pop_served[-npraw_outlier_ndx], water_prod_city_
pop_npraw$produced[-npraw_outlier_ndx], main = "Non-potable Raw Water Produced
vs. Population Served", xlab = "Population Served", ylab = "Gallons of Water Pr
oduced (millions)")
```



```
plot(water_prod_city_pop_npreuse$pop_served[-npreuse_outlier_ndx], water_prod_c
ity_pop_npreuse$produced[-npreuse_outlier_ndx], main = "Non-potable Reuse Water
Produced vs. Population Served", xlab = "Population Served", ylab = "Gallons of
Water Produced (millions)")
```

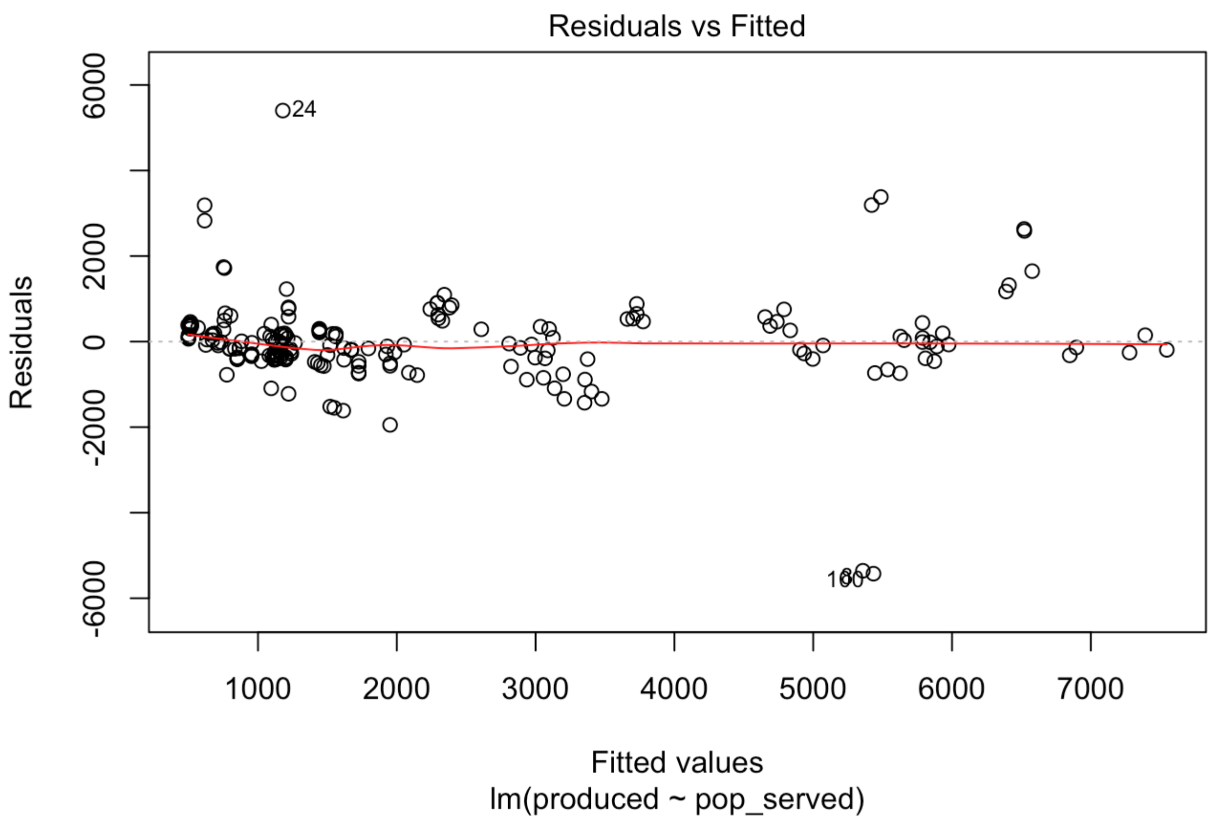


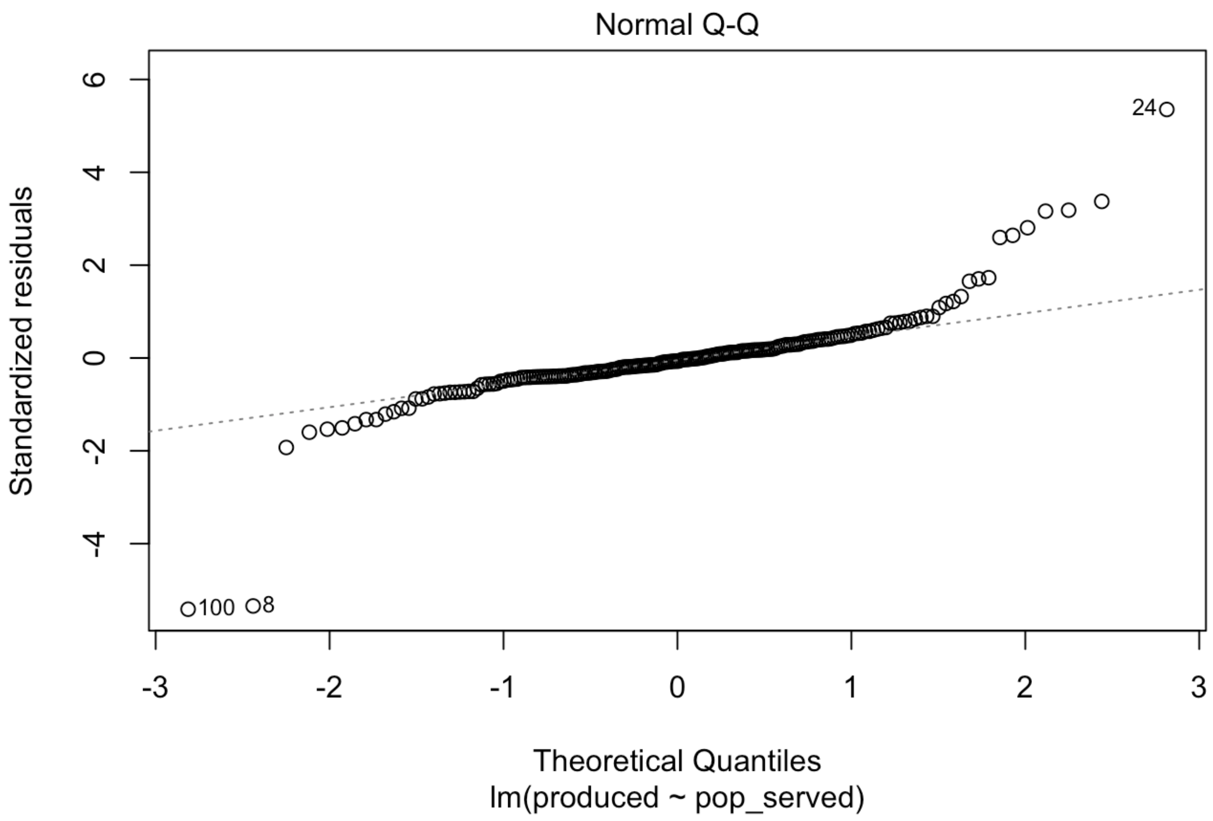
Build Water Production Model

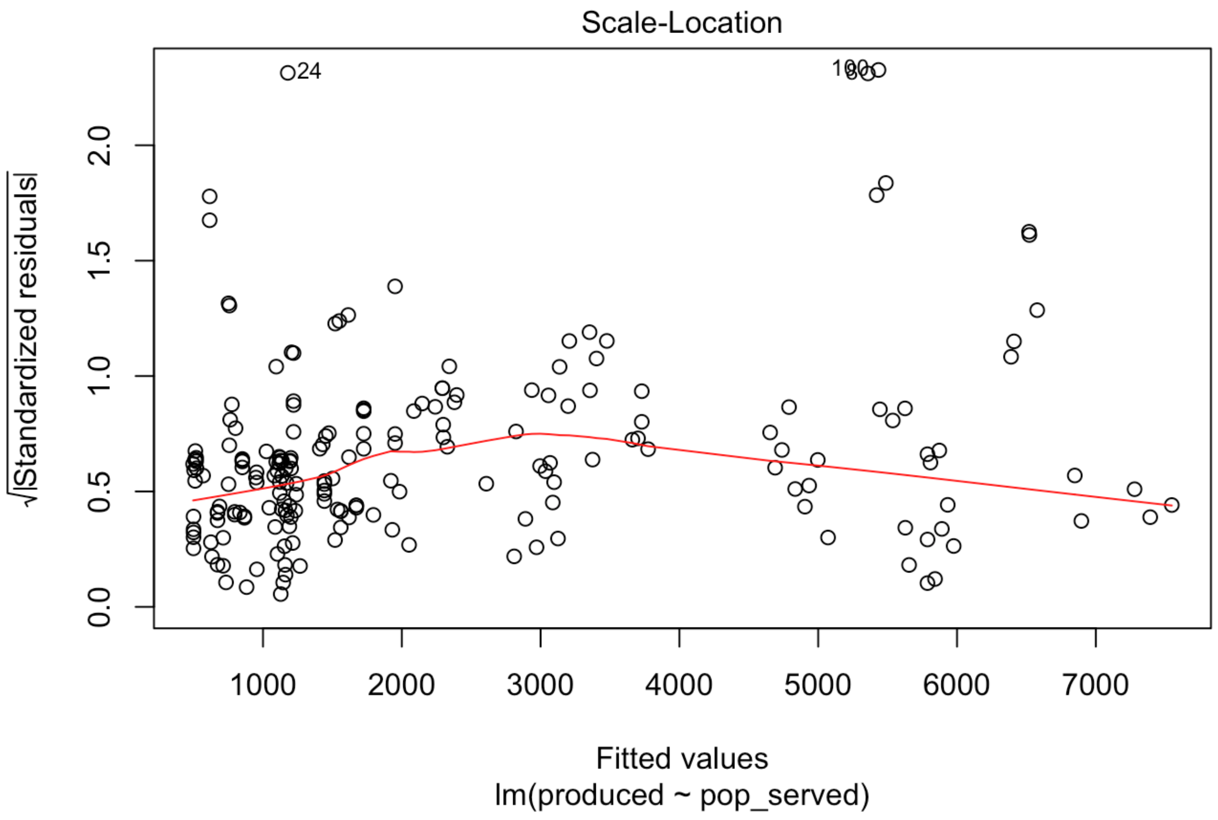
```
#create new df without outliers
water_prod_city_pop_potable_noutlier = water_prod_city_pop_potable[-potable_out
lier_ndx,]

#fit model on new df, use pop to predict water produced
fit = lm(produced ~ pop_served, data = water_prod_city_pop_potable_noutlier)

#check significance of the model
summary(fit)
```

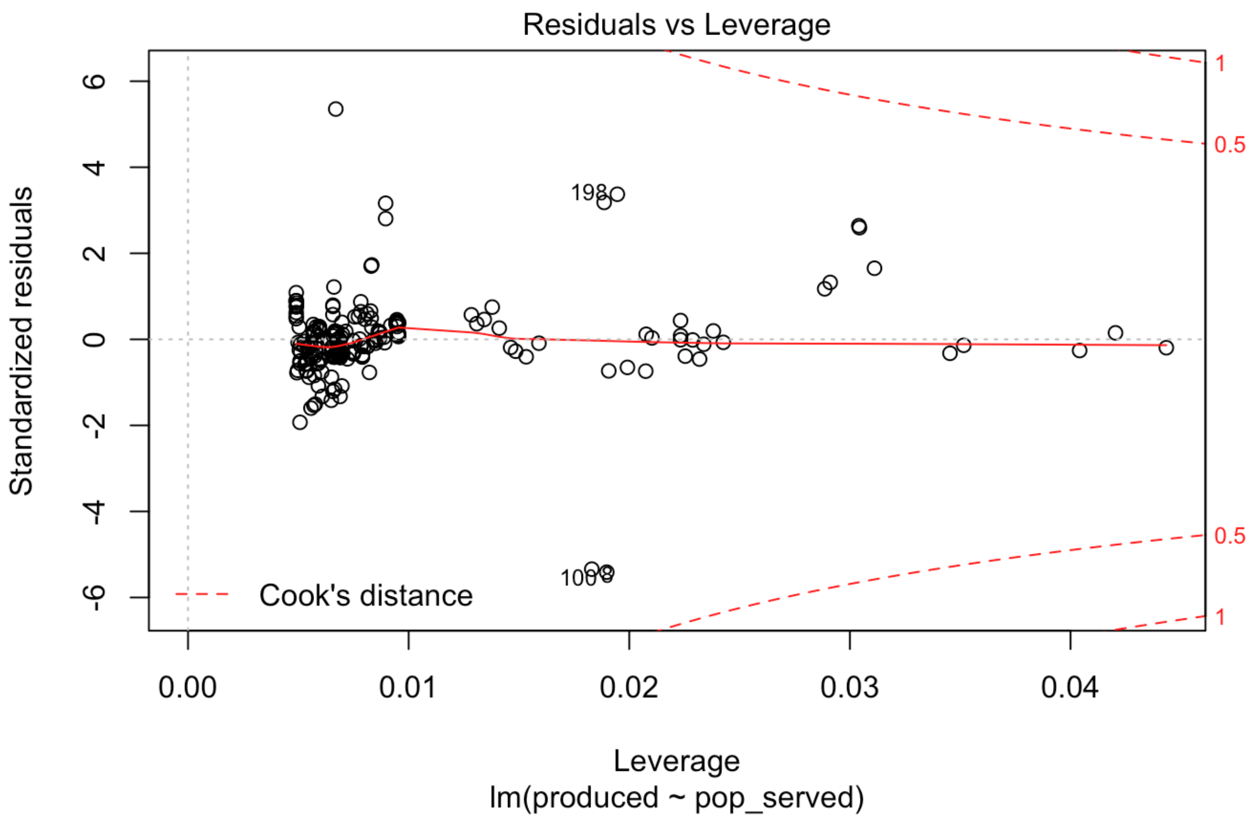
```
##
## Call:
## lm(formula = produced ~ pop_served, data = water_prod_city_pop_potable_noutl
ier)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5426.0  -391.5   -68.4   295.9  5400.7
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.613e+02  1.054e+02   2.479    0.014 *
## pop_served  4.723e-02  1.809e-03  26.109   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1012 on 202 degrees of freedom
## Multiple R-squared:  0.7714, Adjusted R-squared:  0.7703
## F-statistic: 681.7 on 1 and 202 DF,  p-value: < 2.2e-16
```

```
#plot the model to examine adequacy
plot(fit)
```




Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(produced ~ pop_served)
24
100
8
6
4
2
0
-2
-4
-3
-2
-1
0
1
2
3


Scale-Location
√|Standardized residuals|
Fitted values
lm(produced ~ pop_served)
24
100
0.0
0.5
1.0
1.5
2.0
1000
2000
3000
4000
5000
6000
7000



Build Population Model (Code Adapted from Kellen)

```
#Load data
county_dt <- fread("/Users/dsciacca/Documents/Data Analytics Challenge/historic
al-census-data-counties.csv")
municipality_dt <- fread("/Users/dsciacca/Documents/Data Analytics Challenge/hi
storical-census-data-municipalities.csv")

# Remove data with 0 population
county_dt <- county_dt[county_dt$`Total Population` != 0, ]
municipality_dt <- municipality_dt[municipality_dt$`Total Population` != 0, ]

# Only keep citys with 5 or more population values
county_dt<-county_dt[,count:=.N,by=`AREA(S)`][(count>=5)]
municipality_dt<-municipality_dt[,count:=.N,by=`AREA(S)`][(count>=5)]

# Find the unique areas in the data
counties <- unique(county_dt$`AREA(S)`)
municipalities <- unique(municipality_dt$`AREA(S)`)

# Create an empty list
county_mods <- list()
municipality_mods <- list()

# Loop a 3rd order polynomial fit for each area and save results in the empty l
ist
for (i in 1:length(counties)){
  county_mods[[i]] <- lm(`Total Population` ~ poly(YEAR, 3), data = county_dt[c
ounty_dt$`AREA(S)` == counties[i]])
}
for (i in 1:length(municipalities)){
  municipality_mods[[i]] <- lm(`Total Population` ~ poly(YEAR, 3), data = munic
ipality_dt[municipality_dt$`AREA(S)` == municipalities[i]])
}

# Create an empty prediction list
county_preds = NULL
municipality_preds = NULL

# Future years to predict population
fy <- data.frame('YEAR' = c(2020, 2030, 2040, 2050))

# Add the unique areas to the prediction list and empty prediction years
county_preds<- data.table(counties)
county_preds$Y2020_pop = NA
county_preds$Y2030_pop = NA
county_preds$Y2040_pop = NA
county_preds$Y2050_pop = NA

municipality_preds<- data.table(municipalities)
```

```
municipality_preds$Y2020_pop = NA
municipality_preds$Y2030_pop = NA
municipality_preds$Y2040_pop = NA
municipality_preds$Y2050_pop = NA

# Loop predictions through and save results
for (i in 1:length(counties)){
  county_preds$Y2020_pop[i] <- predict(county_mods[[i]], fy)[1]
  county_preds$Y2030_pop[i] <- predict(county_mods[[i]], fy)[2]
  county_preds$Y2040_pop[i] <- predict(county_mods[[i]], fy)[3]
  county_preds$Y2050_pop[i] <- predict(county_mods[[i]], fy)[4]
}

for (i in 1:length(municipalities)){
  municipality_preds$Y2020_pop[i] <- predict(municipality_mods[[i]], fy)[1]
  municipality_preds$Y2030_pop[i] <- predict(municipality_mods[[i]], fy)[2]
  municipality_preds$Y2040_pop[i] <- predict(municipality_mods[[i]], fy)[3]
  municipality_preds$Y2050_pop[i] <- predict(municipality_mods[[i]], fy)[4]
}
```

Predict Future Water Production

```
#add empty water prediction columns
county_preds$Y2020_water = NA
county_preds$Y2030_water = NA
county_preds$Y2040_water = NA
county_preds$Y2050_water = NA

municipality_preds$Y2020_water = NA
municipality_preds$Y2030_water = NA
municipality_preds$Y2040_water = NA
municipality_preds$Y2050_water = NA

#make water production predictions and save results
county_preds$Y2020_water = predict(fit, data.frame("pop_served" = county_preds$
Y2020_pop))
county_preds$Y2030_water = predict(fit, data.frame("pop_served" = county_preds$
Y2030_pop))
county_preds$Y2040_water = predict(fit, data.frame("pop_served" = county_preds$
Y2040_pop))
county_preds$Y2050_water = predict(fit, data.frame("pop_served" = county_preds$
Y2050_pop))

municipality_preds$Y2020_water = predict(fit, data.frame("pop_served" = municip
ality_preds$Y2020_pop))
municipality_preds$Y2030_water = predict(fit, data.frame("pop_served" = municip
ality_preds$Y2030_pop))
municipality_preds$Y2040_water = predict(fit, data.frame("pop_served" = municip
ality_preds$Y2040_pop))
municipality_preds$Y2050_water = predict(fit, data.frame("pop_served" = municip
ality_preds$Y2050_pop))

# Write our data to csv
write.csv(county_preds, "/Users/dsciacca/Documents/Data Analytics Challenge/wat
er_population_predictions_counties.csv", row.names = FALSE)

write.csv(municipality_preds, "/Users/dsciacca/Documents/Data Analytics Challen
ge/water_population_predictions_municipalities.csv", row.names = FALSE)
```