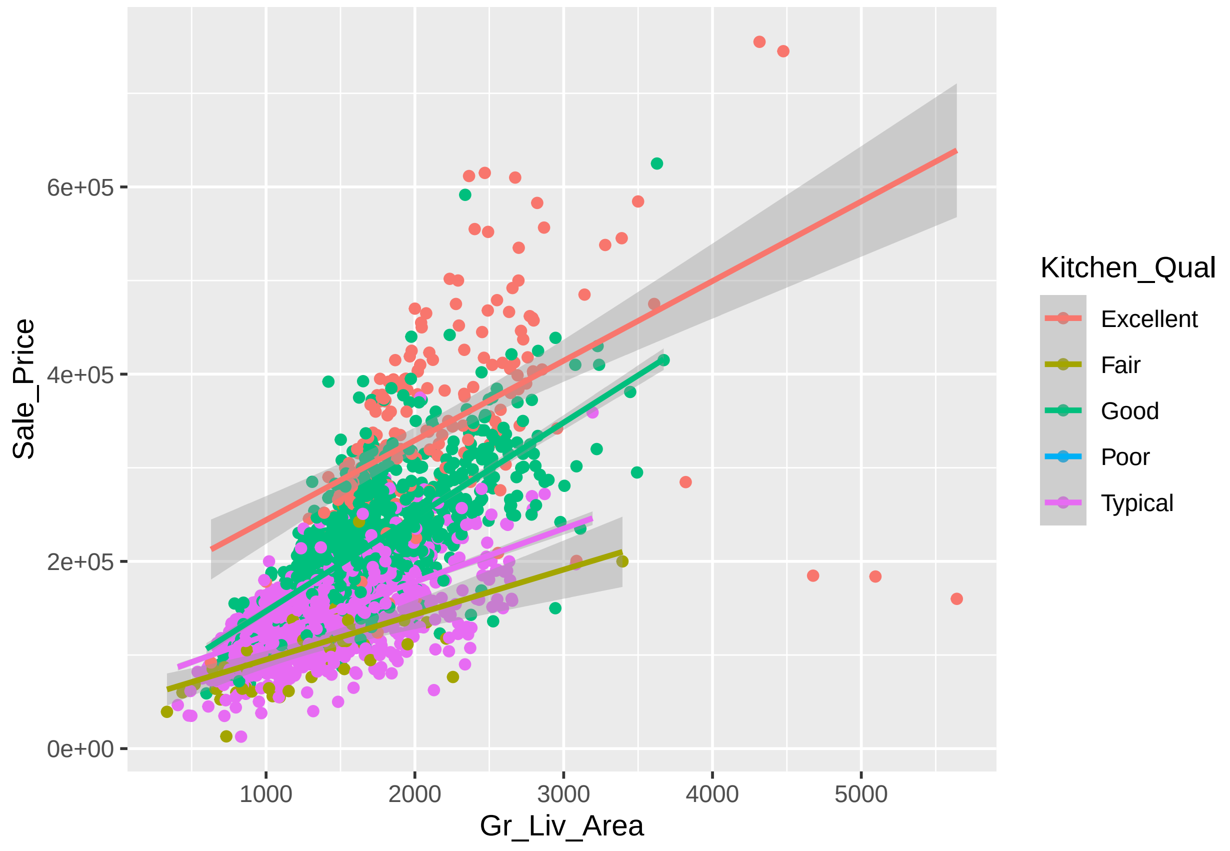
module3

Andrew Estes

2023-01-28

Part 1

Use this script to make a limited version of the AmesHousing Dataset and make a quick scatterplot and linear model.



```
##
## Call:
## lm(formula = Sale_Price ~ ., data = Ames_small)
##
## Coefficients:
##         (Intercept)          Gr_Liv_Area         Central_AirY
##          2265184.67                77.69             40945.69
##    Kitchen_QualFair     Kitchen_QualGood     Kitchen_QualPoor
##          -144273.35           -90505.83          -119301.88
## Kitchen_QualTypical            Year_Sold
##          -134650.92            -1061.43
```

```
##
## Call:
## lm(formula = Sale_Price ~ ., data = Ames_small)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -453124  -23174    -249   20696  324609
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          2.265e+06  1.240e+06   1.827  0.06780 .
## Gr_Liv_Area          7.769e+01  1.791e+00  43.383  < 2e-16 ***
## Central_AirY         4.095e+04  3.426e+03  11.953  < 2e-16 ***
## Kitchen_QualFair    -1.443e+05  6.470e+03 -22.298  < 2e-16 ***
## Kitchen_QualGood    -9.051e+04  3.430e+03 -26.386  < 2e-16 ***
## Kitchen_QualPoor    -1.193e+05  4.423e+04  -2.697  0.00703 **
## Kitchen_QualTypical -1.347e+05  3.557e+03 -37.851  < 2e-16 ***
## Year_Sold           -1.061e+03  6.174e+02  -1.719  0.08570 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 43950 on 2922 degrees of freedom
## Multiple R-squared:  0.698,  Adjusted R-squared:  0.6973
## F-statistic: 964.7 on 7 and 2922 DF,  p-value: < 2.2e-16
```

Part 2

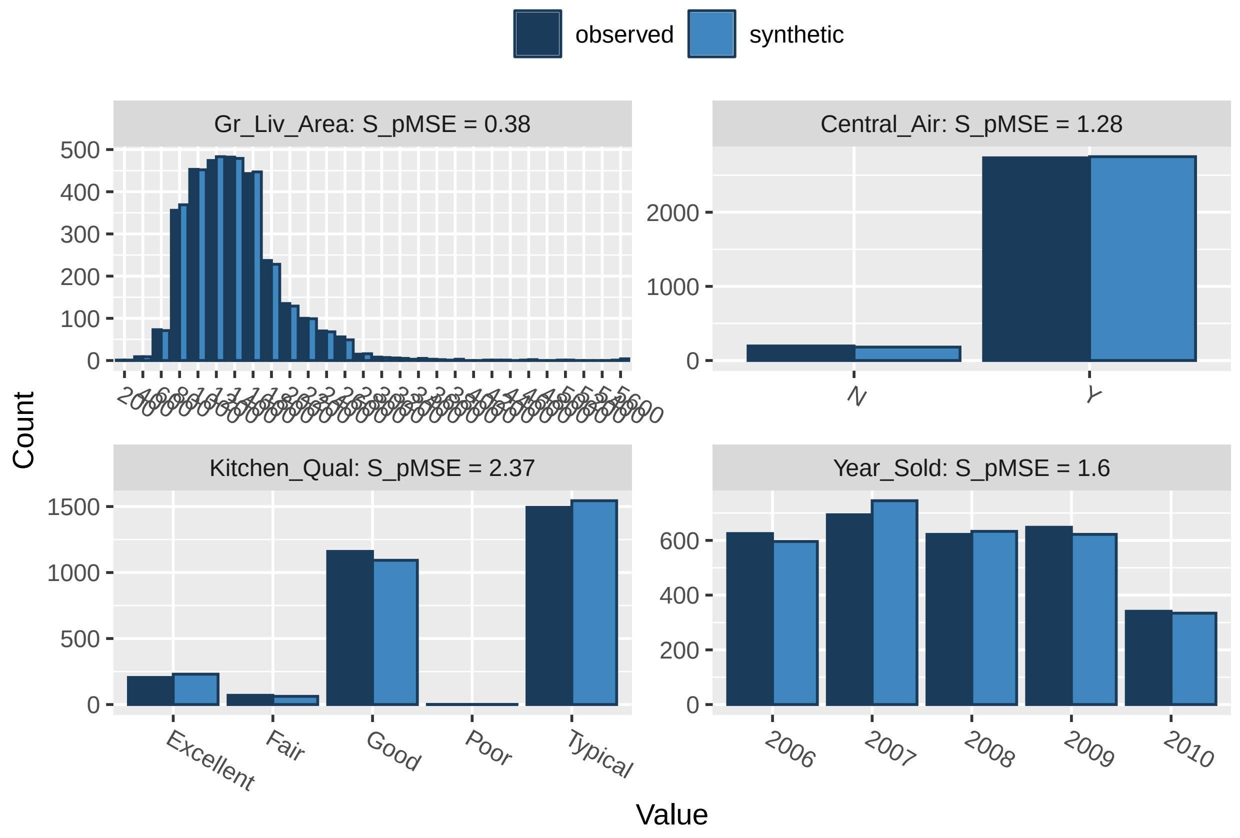
Now, use the synthpop package, perhaps including the "syn" and "lm.synds" commands to make a scatter plot and a synthetic linear model of Ames_small.

Discuss how it's different than the original linear model, if at all.

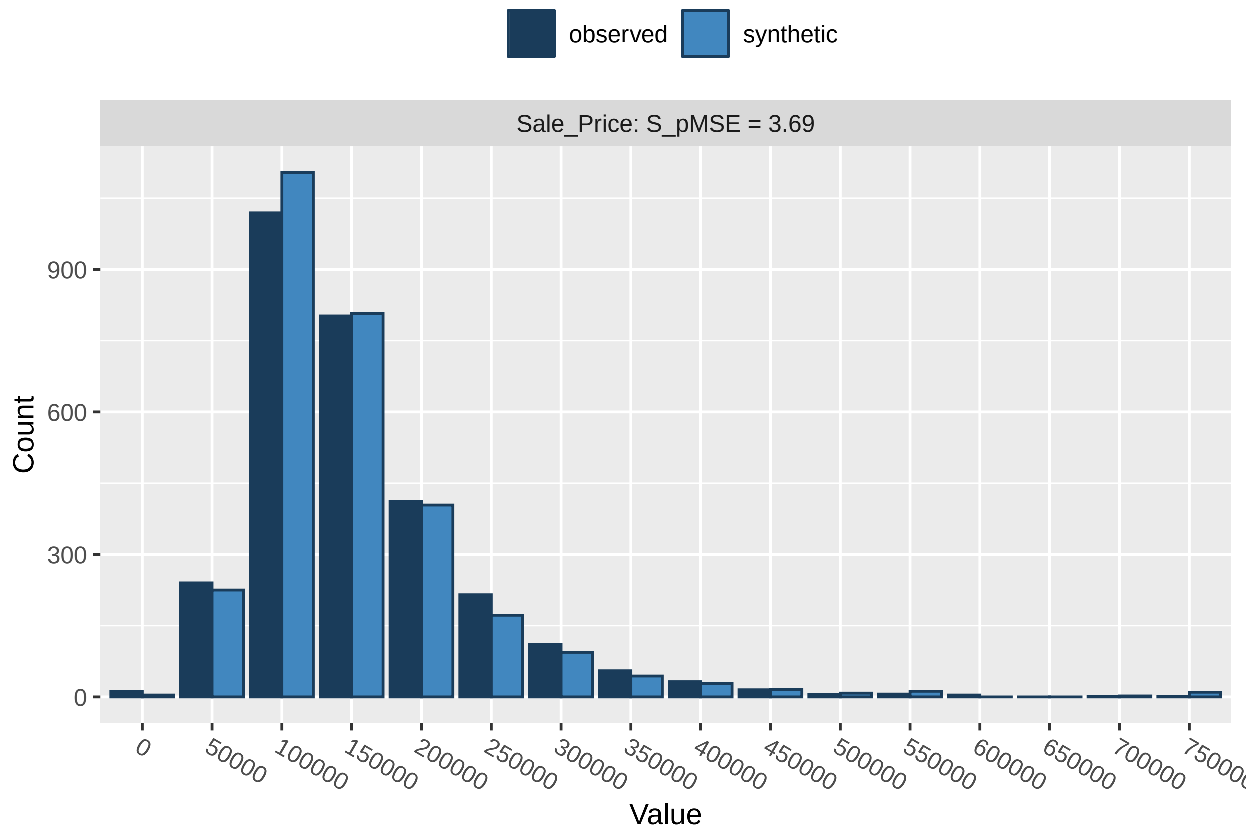
```
##
## Variable(s): Year_Sold numeric but with only 5 or fewer distinct values turned into factor(s) for syn
##
##
## Synthesis
## -----------
##  Gr_Liv_Area Central_Air Kitchen_Qual Year_Sold Sale_Price
```

```
## Synthetic object with one synthesis using methods:
##  Gr_Liv_Area  Central_Air Kitchen_Qual    Year_Sold   Sale_Price
##     "sample"     "logreg"    "polyreg"    "polyreg"   "normrank"
##
##   Gr_Liv_Area   Central_Air    Kitchen_Qual    Year_Sold      Sale_Price
##  Min.   : 334   N: 181      Excellent: 230   Min.   :2006   Min.   : 13100
##  1st Qu.:1123   Y:2749      Fair     :  62   1st Qu.:2007   1st Qu.:129500
##  Median :1436               Good     :1093   Median :2008   Median :157500
##  Mean   :1496               Poor     :   1   Mean   :2008   Mean   :179725
##  3rd Qu.:1729               Typical  :1544   3rd Qu.:2009   3rd Qu.:207000
##  Max.   :5642                                Max.   :2010   Max.   :755000
```

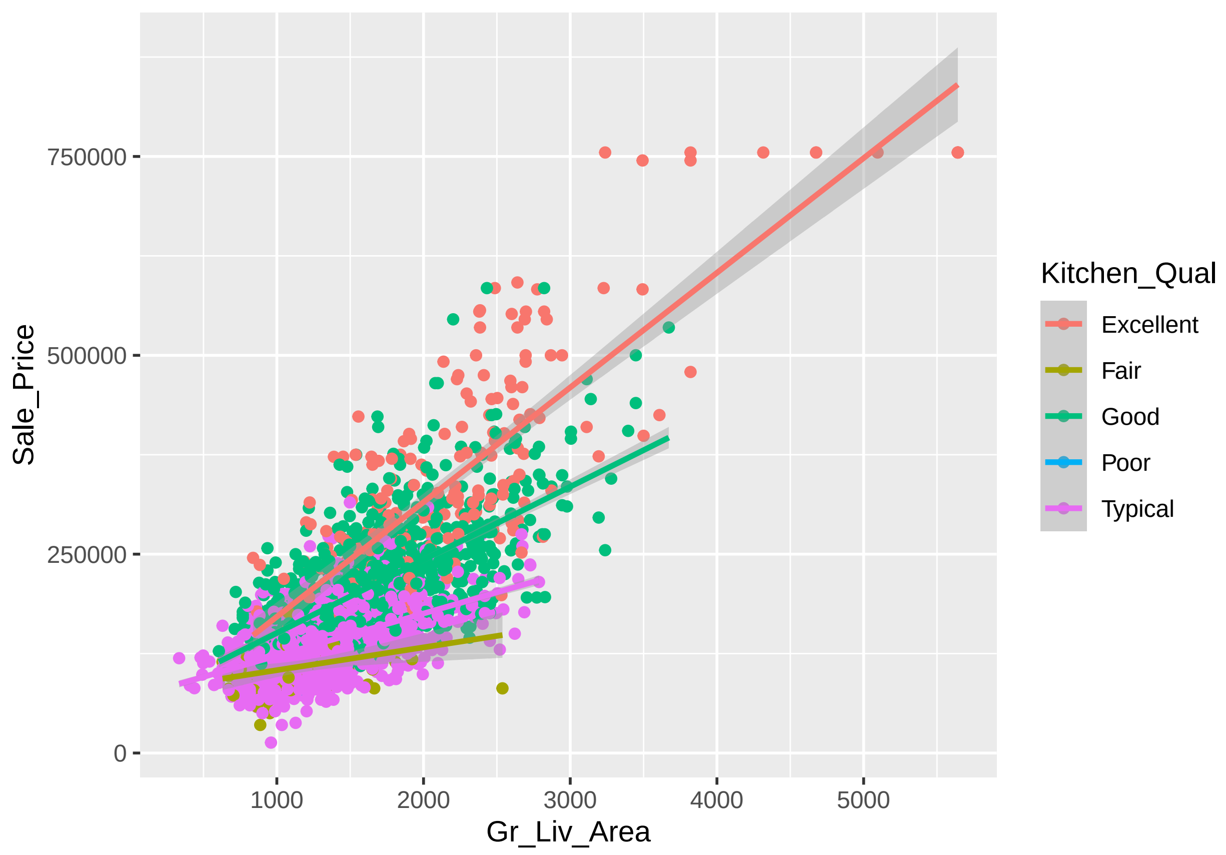
```
##
## Comparing counts observed with synthetic
```



```
## Press return for next variable(s):
```



```
##
## Selected utility measures:
##                   pMSE   S_pMSE df
## Gr_Liv_Area  0.000032 0.380693  4
## Central_Air  0.000027 1.275706  1
## Kitchen_Qual 0.000202 2.368497  4
## Year_Sold    0.000136 1.598788  4
## Sale_Price   0.000315 3.690517  4
```



```
##
## Call:
## lm.synds(formula = Sale_Price ~ ., data = Ames_small_syn)
##
## Coefficient estimates from a single synthesis:
##         (Intercept)         Gr_Liv_Area         Central_AirY    Kitchen_QualFair
##       3349312.87680            89.41704         38039.95928       -127069.72454
##    Kitchen_QualGood    Kitchen_QualPoor Kitchen_QualTypical           Year_Sold
##        -80393.10605       -165129.53058       -119493.42710         -1615.36733

## Fit to synthetic data set with a single synthesis. Inference to population
## coefficients when all variables in the model are synthesised.
##
## Call:
## lm.synds(formula = Sale_Price ~ ., data = Ames_small_syn)
##
## Combined estimates:
##                       Beta.syn se.Beta.syn    z.syn Pr(>|z.syn|)
## (Intercept)          3.3493e+06  1.9053e+06   1.7579      0.07877 .
## Gr_Liv_Area          8.9417e+01  2.6562e+00  33.6633    < 2.2e-16 ***
## Central_AirY         3.8040e+04  5.2965e+03   7.1820    6.867e-13 ***
## Kitchen_QualFair    -1.2707e+05  1.0039e+04 -12.6572    < 2.2e-16 ***
## Kitchen_QualGood    -8.0393e+04  5.0259e+03 -15.9958    < 2.2e-16 ***
## Kitchen_QualPoor    -1.6513e+05  6.6875e+04  -2.4692      0.01354 *
## Kitchen_QualTypical -1.1949e+05  5.1942e+03 -23.0050    < 2.2e-16 ***
## Year_Sold           -1.6154e+03  9.4899e+02  -1.7022      0.08872 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

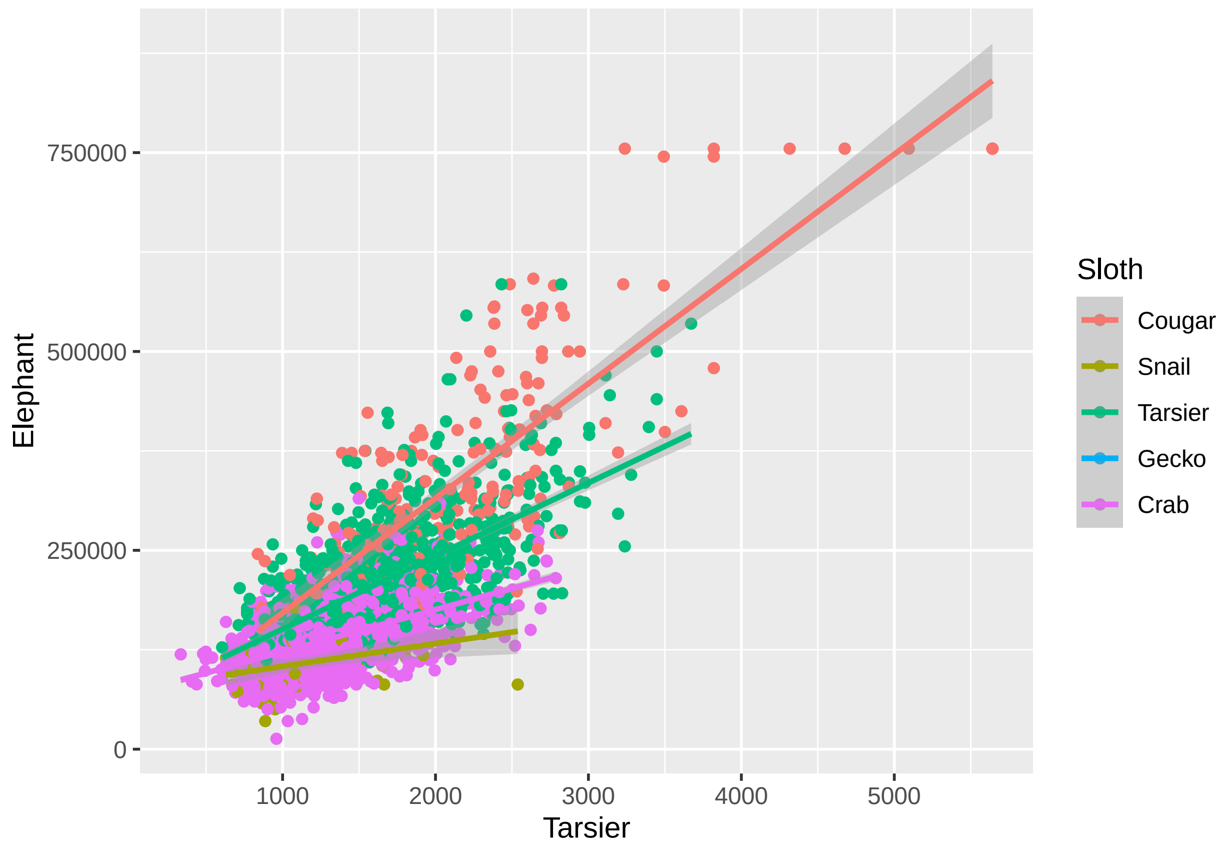
The seemingly significant difference between the original data and the synthesized data is the loss of "poor" kitchen quality. The word seemingly was purposely chosen as there was only one observation of "poor" kitchen quality in the original dataset so the loss of this category is actually a boon to the model.

It is hard to do an actual one-to-one comparison using the summary function due to the difference in the output. That being said, the coefficients are similar to one another as is the synthesized data to the original data so I feel comfortable stating that the two models performed similarly.

Part 3

Then, take a few minutes to check out another feature of synthpop (or another package) and explain what you found.

Implement it in a version of Ames_small and explain briefly what the feature does.



The synthpop package creates a simulated dataframe using parameters from the original data. Now there isn't anything particularly identifying about kitchen quality but I wanted to try the gganonymise function to see a new output.

Bonus

Not really sure what I'm doing here. Pretty much copy + paste from https://github.com/brubinstein/diffpriv It has something to do with calclating theoretical sensitivity analysis

```
## a target function we'd like to run on private data X, releasing the result
target <- function(Ames_small_syn) mean(Ames_small_syn)

#install.packages("diffpriv")
library(diffpriv)
## target seeks to release a numeric, so we'll use the Laplace mechanism---a
## standard generic mechanism for privatizing numeric responses
mech <- DPMechLaplace(target = target)

## set a dataset sampling distribution, then estimate target sensitivity with
## sufficient samples for subsequent mechanism responses to achieve random
## differential privacy with confidence 1-gamma
distr <- function(n) rnorm(n)
mech <- sensitivitySampler(mech, oracle = distr, n = 5, gamma = 0.1)
#> Sampling sensitivity with m=285 gamma=0.1 k=285
mech@sensitivity    ## DPMech and subclasses are S4: slots accessed via @
```

```
## [1] 0.8101093
```

```
X <- c(0.328,-1.444,-0.511,0.154,-2.062) # length is sensitivitySampler() n
r <- releaseResponse(mech, privacyParams = DPParamsEps(epsilon = 1), X = X)
cat("Private response r$response:   ", r$response,
  "\nNon-private response target(X):", target(X))
```

```
## Private response r$response:    -0.4693441
## Non-private response target(X): -0.707
```