COMPSCIX 415.2 Homework 7

Vishnu Vardhan

3/17/2018

Contents

Data sources from Kaggle: https://www.kaggle.com/c/house-prices-advanced-regression-techniques/leaderboard

Data has not been uploaded into git

Exercise 1

Load the train.csv dataset into R. How many observations and columns are there?

Answer

There are 81 variables, and 1460 observations

```
train <- read_csv('train.csv')
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   Id = col_integer(),
##   MSSubClass = col_integer(),
##   LotFrontage = col_integer(),
##   LotArea = col_integer(),
##   OverallQual = col_integer(),
##   OverallCond = col_integer(),
##   YearBuilt = col_integer(),
##   YearRemodAdd = col_integer(),
##   MasVnrArea = col_integer(),
##   BsmtFinSF1 = col_integer(),
##   BsmtFinSF2 = col_integer(),
##   BsmtUnfSF = col_integer(),
##   TotalBsmtSF = col_integer(),
##   `1stFlrSF` = col_integer(),
##   `2ndFlrSF` = col_integer(),
##   LowQualFinSF = col_integer(),
##   GrLivArea = col_integer(),
##   BsmtFullBath = col_integer(),
##   BsmtHalfBath = col_integer(),
##   FullBath = col_integer()
##   # ... with 18 more columns
## )
```

```
## See spec(...) for full column specifications.
```

```r
glimpse(train)
```

```
## Observations: 1,460
## Variables: 81
## $ Id            <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1...
## $ MSSubClass    <int> 60, 20, 60, 70, 60, 50, 20, 60, 50, 190, 20, 60,...
## $ MSZoning      <chr> "RL", "RL", "RL", "RL", "RL", "RL", "RL", "RL", ...
## $ LotFrontage   <int> 65, 80, 68, 60, 84, 85, 75, NA, 51, 50, 70, 85, ...
## $ LotArea       <int> 8450, 9600, 11250, 9550, 14260, 14115, 10084, 10...
## $ Street        <chr> "Pave", "Pave", "Pave", "Pave", "Pave", "Pave", ...
## $ Alley         <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ LotShape      <chr> "Reg", "Reg", "IR1", "IR1", "IR1", "IR1", "Reg",...
## $ LandContour   <chr> "Lvl", "Lvl", "Lvl", "Lvl", "Lvl", "Lvl", "Lvl",...
## $ Utilities     <chr> "AllPub", "AllPub", "AllPub", "AllPub", "AllPub"...
## $ LotConfig     <chr> "Inside", "FR2", "Inside", "Corner", "FR2", "Ins...
## $ LandSlope     <chr> "Gtl", "Gtl", "Gtl", "Gtl", "Gtl", "Gtl", "Gtl",...
## $ Neighborhood  <chr> "CollgCr", "Veenker", "CollgCr", "Crawfor", "NoR...
## $ Condition1    <chr> "Norm", "Feedr", "Norm", "Norm", "Norm", "Norm",...
## $ Condition2    <chr> "Norm", "Norm", "Norm", "Norm", "Norm", "Norm", ...
## $ BldgType      <chr> "1Fam", "1Fam", "1Fam", "1Fam", "1Fam", "1Fam", ...
## $ HouseStyle    <chr> "2Story", "1Story", "2Story", "2Story", "2Story"...
## $ OverallQual   <int> 7, 6, 7, 7, 8, 5, 8, 7, 7, 5, 5, 9, 5, 7, 6, 7, ...
## $ OverallCond   <int> 5, 8, 5, 5, 5, 5, 5, 6, 5, 6, 5, 5, 6, 5, 5, 8, ...
## $ YearBuilt     <int> 2003, 1976, 2001, 1915, 2000, 1993, 2004, 1973, ...
## $ YearRemodAdd  <int> 2003, 1976, 2002, 1970, 2000, 1995, 2005, 1973, ...
## $ RoofStyle     <chr> "Gable", "Gable", "Gable", "Gable", "Gable", "Ga...
## $ RoofMatl      <chr> "CompShg", "CompShg", "CompShg", "CompShg", "Com...
## $ Exterior1st   <chr> "VinylSd", "MetalSd", "VinylSd", "Wd Sdng", "Vin...
## $ Exterior2nd   <chr> "VinylSd", "MetalSd", "VinylSd", "Wd Shng", "Vin...
## $ MasVnrType    <chr> "BrkFace", "None", "BrkFace", "None", "BrkFace",...
## $ MasVnrArea    <int> 196, 0, 162, 0, 350, 0, 186, 240, 0, 0, 0, 286, ...
## $ ExterQual     <chr> "Gd", "TA", "Gd", "TA", "Gd", "TA", "Gd", "TA", ...
## $ ExterCond     <chr> "TA", "TA", "TA", "TA", "TA", "TA", "TA", "TA", ...
## $ Foundation    <chr> "PConc", "CBlock", "PConc", "BrkTil", "PConc", "...
## $ BsmtQual      <chr> "Gd", "Gd", "Gd", "TA", "Gd", "Gd", "Ex", "Gd", ...
## $ BsmtCond      <chr> "TA", "TA", "TA", "Gd", "TA", "TA", "TA", "TA", ...
## $ BsmtExposure  <chr> "No", "Gd", "Mn", "No", "Av", "No", "Av", "Mn", ...
## $ BsmtFinType1  <chr> "GLQ", "ALQ", "GLQ", "ALQ", "GLQ", "GLQ", "GLQ",...
## $ BsmtFinSF1    <int> 706, 978, 486, 216, 655, 732, 1369, 859, 0, 851,...
## $ BsmtFinType2  <chr> "Unf", "Unf", "Unf", "Unf", "Unf", "Unf", "Unf",...
## $ BsmtFinSF2    <int> 0, 0, 0, 0, 0, 0, 0, 32, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ BsmtUnfSF     <int> 150, 284, 434, 540, 490, 64, 317, 216, 952, 140,...
## $ TotalBsmtSF   <int> 856, 1262, 920, 756, 1145, 796, 1686, 1107, 952,...
## $ Heating       <chr> "GasA", "GasA", "GasA", "GasA", "GasA", "GasA", ...
## $ HeatingQC     <chr> "Ex", "Ex", "Ex", "Gd", "Ex", "Ex", "Ex", "Ex", ...
## $ CentralAir    <chr> "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y"...
## $ Electrical    <chr> "SBrkr", "SBrkr", "SBrkr", "SBrkr", "SBrkr", "SB...
## $ `1stFlrSF`    <int> 856, 1262, 920, 961, 1145, 796, 1694, 1107, 1022...
## $ `2ndFlrSF`    <int> 854, 0, 866, 756, 1053, 566, 0, 983, 752, 0, 0, ...
## $ LowQualFinSF  <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ GrLivArea     <int> 1710, 1262, 1786, 1717, 2198, 1362, 1694, 2090, ...
## $ BsmtFullBath  <int> 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, ...
## $ BsmtHalfBath  <int> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ FullBath      <int> 2, 2, 2, 1, 2, 1, 2, 2, 2, 1, 1, 3, 1, 2, 1, 1, ...
## $ HalfBath      <int> 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, ...
## $ BedroomAbvGr  <int> 3, 3, 3, 3, 4, 1, 3, 3, 2, 2, 3, 4, 2, 3, 2, 2, ...
## $ KitchenAbvGr  <int> 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, ...
## $ KitchenQual   <chr> "Gd", "TA", "Gd", "Gd", "Gd", "TA", "Gd", "TA", ...
## $ TotRmsAbvGrd  <int> 8, 6, 6, 7, 9, 5, 7, 7, 8, 5, 5, 11, 4, 7, 5, 5,...
## $ Functional    <chr> "Typ", "Typ", "Typ", "Typ", "Typ", "Typ", "Typ",...
## $ Fireplaces    <int> 0, 1, 1, 1, 1, 0, 1, 2, 2, 2, 0, 2, 0, 1, 1, 0, ...
## $ FireplaceQu   <chr> NA, "TA", "TA", "Gd", "TA", NA, "Gd", "TA", "TA"...
## $ GarageType    <chr> "Attchd", "Attchd", "Attchd", "Detchd", "Attchd"...
## $ GarageYrBlt   <int> 2003, 1976, 2001, 1998, 2000, 1993, 2004, 1973, ...
## $ GarageFinish  <chr> "RFn", "RFn", "RFn", "Unf", "RFn", "Unf", "RFn",...
## $ GarageCars    <int> 2, 2, 2, 3, 3, 2, 2, 2, 2, 1, 1, 3, 1, 3, 1, 2, ...
## $ GarageArea    <int> 548, 460, 608, 642, 836, 480, 636, 484, 468, 205...
## $ GarageQual    <chr> "TA", "TA", "TA", "TA", "TA", "TA", "TA", "TA", ...
## $ GarageCond    <chr> "TA", "TA", "TA", "TA", "TA", "TA", "TA", "TA", ...
## $ PavedDrive    <chr> "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y"...
## $ WoodDeckSF    <int> 0, 298, 0, 0, 192, 40, 255, 235, 90, 0, 0, 147, ...
## $ OpenPorchSF   <int> 61, 0, 42, 35, 84, 30, 57, 204, 0, 4, 0, 21, 0, ...
## $ EnclosedPorch <int> 0, 0, 0, 272, 0, 0, 0, 228, 205, 0, 0, 0, 0, 0, ...
## $ `3SsnPorch`   <int> 0, 0, 0, 0, 0, 320, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ ScreenPorch   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 176, 0, 0, 0...
## $ PoolArea      <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ PoolQC        <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ Fence         <chr> NA, NA, NA, NA, NA, "MnPrv", NA, NA, NA, NA, NA,...
## $ MiscFeature   <chr> NA, NA, NA, NA, NA, "Shed", NA, "Shed", NA, NA, ...
## $ MiscVal       <int> 0, 0, 0, 0, 0, 700, 0, 350, 0, 0, 0, 0, 0, 0, 0,...
## $ MoSold        <int> 2, 5, 9, 2, 12, 10, 8, 11, 4, 1, 2, 7, 9, 8, 5, ...
## $ YrSold        <int> 2008, 2007, 2008, 2006, 2008, 2009, 2007, 2009, ...
## $ SaleType      <chr> "WD", "WD", "WD", "WD", "WD", "WD", "WD", "WD", ...
## $ SaleCondition <chr> "Normal", "Normal", "Normal", "Abnorml", "Normal...
## $ SalePrice     <int> 208500, 181500, 223500, 140000, 250000, 143000, ...
```

Exercise 2

Normally at this point you would spend a few days on EDA, but for this homework we will get right to fitting some linear regression models.

Our first step is to randomly split the data into train and test datasets. We will use a 70/30 split. There is an R package that will do the split for you, but let's get some more practice with R and do it ourselves by filling in the blanks in the code below.

Answer

```
# load packages
library(tidyverse)
library(broom)
# When taking a random sample, it is often useful to set a seed so that
# your work is reproducible. Setting a seed will guarantee that the same
# random sample will be generated every time, so long as you always set the
# same seed beforehand
set.seed(29283)
# This data already has an Id column which we can make use of.
# Let's create our training set using sample_frac. Fill in the blank.
train_set <- train %>% sample_frac(size=0.7, replace = FALSE)
# let's create our testing set using the Id column. Fill in the blanks.
test_set <- train %>% filter(!(Id %in% train_set$Id))
```

Exercise 3

Our target is called SalePrice. First, we can fit a simple regression model consisting of only the intercept (the average of SalePrice). Fit the model and then use the broom package to

• take a look at the coefficient, • compare the coefficient to the average value of SalePrice, and • take a look at the R-squared.

Use the code below and fill in the blanks.

Answer

The coefficient of to the average sale price is almost the same as the mean sales price.

With an R-Squared value =0, none of the variation in the data is explained by the mean.

```
# Fit a model with intercept only
mod_0 <- lm(SalePrice ~ 1, data = train_set)
# Double-check that the average SalePrice is equal to our model's coefficient
sprintf ("Mean = %f",mean(train_set$SalePrice))
```

```
## [1] "Mean = 182175.954990"
```

```
tidy(mod_0)
```

```
##          term estimate std.error statistic p.value
## 1 (Intercept)   182176  2492.072  73.10222       0
```

```
# Check the R-squared
glance(mod_0)
```

```
##   r.squared adj.r.squared    sigma statistic p.value df    logLik      AIC
## 1         0             0 79668.37        NA      NA  1 -12983.57 25971.13
##        BIC     deviance df.residual
## 1 25980.99 6.480338e+12        1021
```

Exercise 4

Now fit a linear regression model using GrLivArea, OverallQual, and Neighborhood as the features. Don't forget to look at data_description.txt to understand what these variables mean. Ask yourself these questions before fitting the model:

• What kind of relationship will these features have with our target? • Can the relationship be estimated linearly? • Are these good features, given the problem we are trying to solve?

After fitting the model, output the coefficients and the R-squared using the broom package.

Answer these questions: • How would you interpret the coefficients on GrLivArea and OverallQual? • How would you interpret the coefficient on NeighborhoodBrkSide? • Are the features significant? • Are the features practically significant? • Is the model a good fit (to the training set)?

Answer

```r
mod_1 <- lm(SalePrice ~ GrLivArea + OverallQual + Neighborhood, data = train_set)
tidy(mod_1)
```

```
##                   term     estimate    std.error  statistic      p.value
## 1          (Intercept) -45017.87483 12933.341808 -3.4807612 5.216927e-04
## 2            GrLivArea     62.77735     3.006033 20.8837885 1.337222e-80
## 3          OverallQual  21692.23178  1353.714104 16.0242342 1.389020e-51
## 4  NeighborhoodBlueste -38288.88063 36531.907177 -1.0480942 2.948497e-01
## 5   NeighborhoodBrDale -43314.05372 14524.693991 -2.9820975 2.932566e-03
## 6  NeighborhoodBrkSide -14064.37052 11318.850018 -1.2425618 2.143221e-01
## 7  NeighborhoodClearCr  27839.00662 13561.346871  2.0528202 4.035110e-02
## 8  NeighborhoodCollgCr   4297.67432 10372.304467  0.4143413 6.787135e-01
## 9  NeighborhoodCrawfor   7423.05573 11371.511784  0.6527765 5.140512e-01
## 10 NeighborhoodEdwards -15284.11495 10994.287187 -1.3901870 1.647830e-01
## 11 NeighborhoodGilbert  -8357.55930 10894.173472 -0.7671586 4.431692e-01
## 12  NeighborhoodIDOTRR -32689.43085 12603.712743 -2.5936350 9.636216e-03
## 13 NeighborhoodMeadowV -14446.06504 14190.148622 -1.0180348 3.089089e-01
## 14 NeighborhoodMitchel   1922.31487 11788.608170  0.1630655 8.705000e-01
## 15   NeighborhoodNAmes  -7719.67883 10375.956174 -0.7439969 4.570540e-01
## 16 NeighborhoodNoRidge  47685.16790 12567.432633  3.7943444 1.569690e-04
## 17 NeighborhoodNPkVill -20240.71145 16548.664867 -1.2231024 2.215806e-01
## 18 NeighborhoodNridgHt  63872.80848 10880.456671  5.8704161 5.917964e-09
## 19  NeighborhoodNWAmes -12279.33299 11047.502893 -1.1115030 2.666204e-01
## 20 NeighborhoodOldTown -36107.07577 10849.170903 -3.3280954 9.064637e-04
## 21  NeighborhoodSawyer  -4121.92502 11252.369778 -0.3663162 7.142070e-01
## 22 NeighborhoodSawyerW  -5391.97074 11230.758221 -0.4801075 6.312565e-01
## 23 NeighborhoodSomerst  18700.96725 10772.212794  1.7360377 8.286672e-02
## 24 NeighborhoodStoneBr  65712.45881 12745.312907  5.1558137 3.045915e-07
## 25   NeighborhoodSWISU -45451.86707 13564.792586 -3.3507233 8.363074e-04
## 26  NeighborhoodTimber  27925.08619 11985.325857  2.3299397 2.000859e-02
## 27 NeighborhoodVeenker  54913.12768 16521.075497  3.3238228 9.203087e-04
```

• How would you interpret the coefficients on GrLivArea and OverallQual?

GrLivArea coefficient of 62.77 says that all things being equal, the per-sq-ft charge is about 62 units. The OverallQual parameter specifies that the quality of the house increases the value by 21,692 at each level of quality.

• How would you interpret the coefficient on NeighborhoodBrkSide?

The co-efficient implies that being in that neighborhood penalizes a home (by -14,064), and its value drops. But because the p-value is 0.214, it is not a significant predictor.

- Are the features significant?

Not all p-values are significant. Since p-values have to be less than 0.05, many of the neighborhood p-values are not significant.

- Are the features practically significant?

Yes, the features are practicaly significant, since we know that these factors play critically into a home's value. There are other factors that could improve our prediction.

```
glance(mod_1)
```

```
##   r.squared adj.r.squared   sigma statistic p.value df    logLik      AIC
## 1 0.8099927     0.8050277 35178.1  163.1401       0 27 -12134.95 24325.91
##        BIC     deviance df.residual
## 1 24463.93 1.231311e+12         995
```

- Is the model a good fit (to the training set)?

Yes, the model is a good-fit, because the adjusted R-squared is about 0.8. I.e the model explains 80% of the variance, which is much better than a random guess. It could be better

Exercise 5

Evaluate the model on test_set using the root mean squared error (RMSE). Use the predict function to get the model predictions for the testing set. Recall that RMSE is the square root of the mean of the squared errors:

Hint: use the sqrt() and mean() functions:

test_predictions <- predict(NAME_OF_YOUR_MODEL_HERE, newdata = test_set) rmse <- sqrt(mean((_____ - _____)^2))

Answer

```
test_predictions <- predict(mod_1, newdata = test_set)
rmse <- sqrt(mean((test_predictions - test_set$SalePrice)^2))
rmse
```

```
## [1] 41915.27
```

The root mean square value is 41,915. Given the summary stats below for the sale prie, the value looks reasonable

```
summary(train$SalePrice)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   34900  129975  163000  180921  214000  755000
```

Exercise 6 (OPTIONAL - won't be graded)

Feel free to play around with linear regression. Add some other features and see how the model results change. Test the model on test_set to compare the RMSE's.

Answer

Seems like i was able to improve from the 41K RMS value to about 36K, by including 1. Year built 2. Interactions between BldgType and Neighborhood 3. Interactions between Neighborhood and OverallQuality

```
mod_4 <- lm(SalePrice ~ GrLivArea +
              OverallQual +
              Neighborhood +
              YearBuilt +
              BldgType:Neighborhood  +
              Neighborhood:OverallQual,
            data = train_set)
test_predictions <- predict(mod_4, newdata = test_set)
```

```
## Warning in predict.lm(mod_4, newdata = test_set): prediction from a rank-
## deficient fit may be misleading
```

```
rmse <- sqrt(mean((test_predictions - test_set$SalePrice)^2))
rmse
```

```
## [1] 36646.79
```

Exercise 7

One downside of the linear model is that it is sensitive to unusual values because the distance incorporates a squared term. Fit a linear model to the simulated data below, and visualise the results. Rerun a few times to generate different simulated datasets.

What do you notice about the model?

```
sim1a <- tibble(
x = rep(1:10, each = 3),
y = x * 1.5 + 6 + rt(length(x), df = 2)
)
```

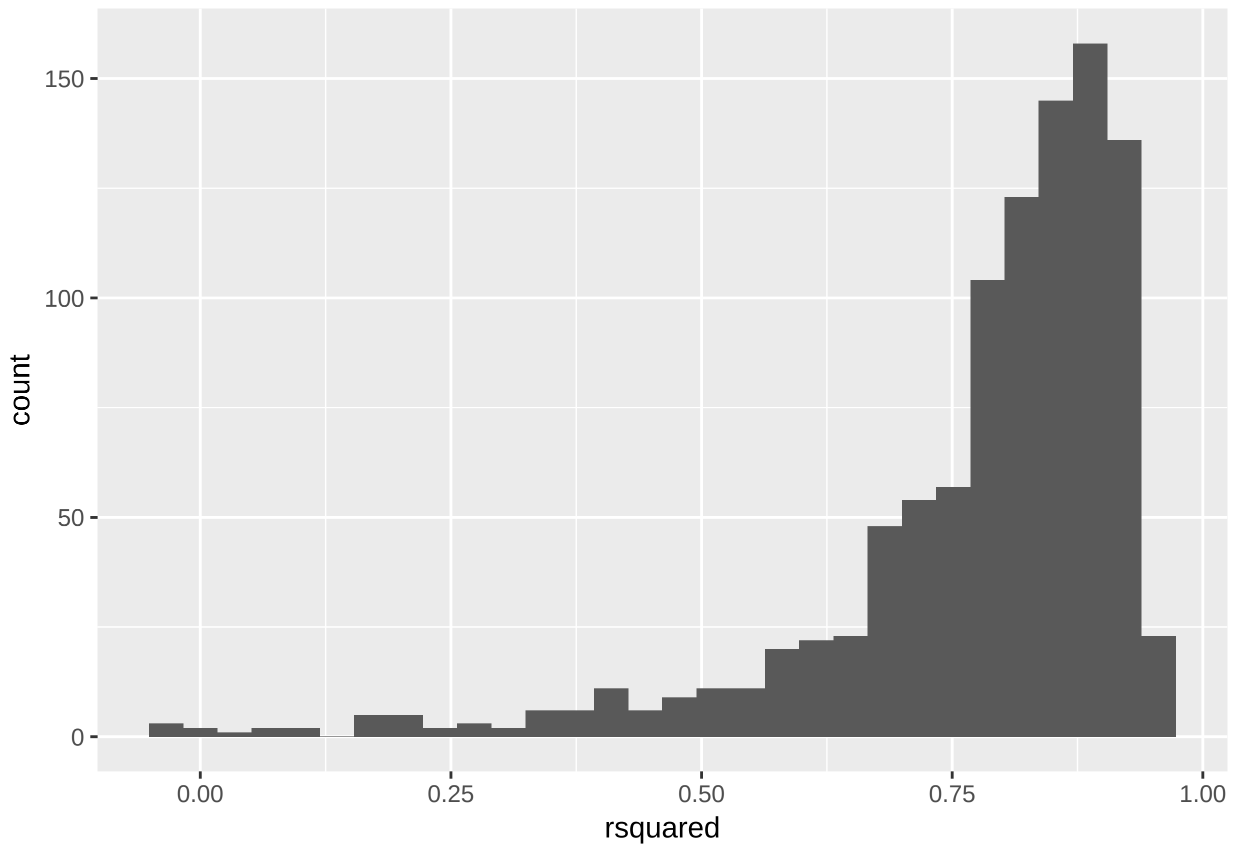
Answer

As can be seen in the code below, the r-squared values can vary pretty dramatically, with most of the values being above 0.75

```
df <- tibble(run = integer(), rsquared = double())
for (i in c(1:1000)) {
  sim1a <- tibble(
  x = rep(1:10, each = 3),
  y = x * 1.5 + 6 + rt(length(x), df = 2)
  )
  mod_2 <- lm(y ~ x, data=sim1a)
  results <- glance(mod_2)$adj.r.squared
  df <- add_row(df,run = i,rsquared = results)
}

ggplot(data = df, mapping=aes(x=rsquared)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



150
100
50
0
count
0.00
0.25
0.50
0.75
1.00
rsquared