Data Pre-processing (Feature Engineering)

Applied Machine Learning for Educational Data Science

true

08/23/2021

Contents

[Updated: Fri, Aug 27, 2021 - 22:06:32]

1. Scales of Measurement and Types of Variables

It is important to understand the nature of variables and how they were measured and represented in a dataset. In social sciences, in particular psychology, there is a methodological consensus about the framework provided by Stevens (1946), also see Michell (2002) for an in-depth discussion. According to Stevens' definition, there are four levels of measurement: nominal, ordinal, interval, and ratio. Whether a variable is considered having a nominal, ordinal, interval, or ratio scale depends on the character of the empirical operations performed while constructing the variable.

- Nominal scale: Variables with a nominal scale cannot be be meaningfully added, subtracted, divided, or multiplied. Also, there is no hierarchical order among the assigned values. Most variables that contains labels for individual observations can be considered as nominal, e.g., hair color, city, state, ethnicity.
- Ordinal scale: Variables with an ordinal scale also represent labels; however, there is a meaningful hierarchy among the assigned values. For instance, if a variable is coded as Low, Medium, and High, they are simply labels but we know that High represents something more than Medium, and Medium represents something higher than Low (High > Medium > Low). On the other side, the distance between the assigned values do not necessarily represent the same amount of difference. Other examples of variables that can be considered as ordinal are letter grades (A-F), scores from likert type items (Strongly agree, agree, disagree, strongly disagree), education status(high school, college, master's, PhD), cancer stage (stage1, stage2, stage3), order of finish in a competition (1st, 2nd, 3rd).
- Interval scale: Variables with an ordinal scale represents quantities with equal measurement units but they don't have an absolute zero point. For instance, a typical example of an interval scale is temperature measured on the Fahrenheit scale. The difference between 20F and 30F is the same difference as the difference between 60F and 70F. However, 0F does not indicate no heat.
- Ratio scale: Variables with a ratio scale represents quantities with equal measurement units and have an absolute zero. Due to the nature of the existence of absolute zero point that represent 'nothing', ratio of measurements are also meaningful. Typical examples are height, mass, distance, length.

Below table provides a summary of properties for each scale.

	Indicating Difference	Indicating Direction of Difference	Indicating Amount of Difference	Has absolute zero
Nominal	X			
Ordinal	X	X		
Interval	X	X	X	
Ratio	X	X	X	X

In this class, we classify the variables in two types: **Categorical** and **Continuous**. The variables with a nominal or ordinal scale are considered as **Categorical** and the variables with an interval or ratio scale are considered as **Continuous**.

NOTE

In the following sections, you will see some R code to process different types of variables using the base R functions to introduce certain types of coding schemes. At the end, I will introduce the `recipes` package to provide a more complete picture of how to do all these different types of encoding in a more efficient manner for this specific dataset.

---

2. Processing Categorical Variables

When there are categorical predictors in a dataset, it is important to translate them into numerical codes. When encoding categorical predictors, we try to preserve as much information as possible from its labels. Therefore, different strategies may be used for categorical variables with different ordinal scales.

2.1 One-hot encoding (Dummy Variables)

A dummy variable is a synthetic variable with two outcomes (0 and 1) to represent a group membership. When there is a nominal variable with N levels, it is typical to create N dummy variables to represent the information in the nominal variable. Each dummy variable represents a membership to one of the levels in the nominal variable. These dummy variables can be used as features in predictive models.

In its simplest case, consider variable `Race` in the Recidivism dataset with two levels: Black and White. We can create two dummy variables such that the first dummy variable represents whether or not an individual is Black and the second dummy variable represents whether or not the individual is White.

	Dummy Variable 1	Dummy Variable 2
Black	1	0
White	0	1

```
recidivism <- read.csv('https://raw.githubusercontent.com/uo-datasci-specialization/c4-ml-fall-2021/mai

table(recidivism$Race)
```

```
BLACK WHITE
10222  7889
```

```
recidivism$black <- ifelse(recidivism$Race=='BLACK',1,0)
recidivism$white <- ifelse(recidivism$Race=='WHITE',1,0)

head(recidivism[,c('Race','black','white')])
```

```
   Race black white
1 BLACK     1     0
2 BLACK     1     0
3 BLACK     1     0
4 WHITE     0     1
5 WHITE     0     1
6 BLACK     1     0
```

```
table(recidivism$black)
```

```
    0     1
 7889 10222
```

```r
table(recidivism$white)
```

```

    0     1 
10222  7889 
```

Let's consider another example from the Recidivism dataset. Variable `Prison_Offense` has five categories: Violent/Sex, Violent/Non-Sex, Property, Drug, Other. We can create five dummy variables using the following coding scheme.

	Dummy Variable 1	Dummy Variable 2	Dummy Variable 3	Dummy Variable 4	Dummy Variable 5
Violent/Sex	1	0	0	0	0
Violent/Non-Sex	0	1	0	0	0
Property	0	0	1	0	0
Drug	0	0	0	1	0
Other	0	0	0	0	1

Note that `Prison_Offence` is missing for a number of observations. You can fill-in the missing values prior to creating dummy variables using one of the methods we will discuss later. Alternatively, we can define Missing as the sixth category to preserve that information.

	Dummy Variable 1	Dummy Variable 2	Dummy Variable 3	Dummy Variable 4	Dummy Variable 5	Dummy Variable 6
Violent/Sex	1	0	0	0	0	0
Violent/Non-Sex	0	1	0	0	0	0
Property	0	0	1	0	0	0
Drug	0	0	0	1	0	0
Other	0	0	0	0	1	0
Missing	0	0	0	0	0	1

```r
table(recidivism$Prison_Offense)
```

```

                           Drug           Other        Property Violent/Non-Sex 
           2291            3852            1895            5330            4000 
    Violent/Sex 
            743 
```

```r
names(table(recidivism$Prison_Offense))
```

```
[1] ""                "Drug"            "Other"           "Property"       
[5] "Violent/Non-Sex" "Violent/Sex"    
```

```r
recidivism$off_viosex   <- ifelse(recidivism$Prison_Offense=='Violent/Sex',1,0)
recidivism$off_vionosex <- ifelse(recidivism$Prison_Offense=='Violent/Non-Sex',1,0)
recidivism$off_property <- ifelse(recidivism$Prison_Offense=='Property',1,0)
recidivism$off_drug     <- ifelse(recidivism$Prison_Offense=='Drug',1,0)
recidivism$off_other    <- ifelse(recidivism$Prison_Offense=='Other',1,0)
recidivism$off_missing  <- ifelse(recidivism$Prison_Offense=='',1,0)

head(recidivism[,c('Prison_Offense','off_viosex','off_vionosex','off_property','off_drug','off_other','
```

```
    Prison_Offense off_viosex off_vionosex off_property off_drug off_other
1             Drug          0            0            0        1         0
2  Violent/Non-Sex          0            1            0        0         0
3             Drug          0            0            0        1         0
4         Property          0            0            1        0         0
5         Property          0            0            1        0         0
6                           0            0            0        0         0
7             Drug          0            0            0        1         0
8  Violent/Non-Sex          0            1            0        0         0
9         Property          0            0            1        0         0
10 Violent/Non-Sex          0            1            0        0         0
   off_missing
1            0
2            0
3            0
4            0
5            0
6            1
7            0
8            0
9            0
10           0
```

In some cases, when you have geographical location with a reasonable number of categories (e.g., counties or cities in a state, schools in a district), you can also create dummy variables to represent this information. In our case, the Recidivism dataset has a variable called `Residence_PUMA` indicating Public Use Microdata Area (PUMA) for the residence address at the time individual was released. There is a total of 25 unique codes (1-25) for this variable; however, these numbers are just labels. So, one can create 25 different dummy variables to represent 25 different PUMAs.

NOTE

When you fit a typical regression model without regularizaton using ordinary least-squares (OLS), a typical practice is to drop a dummy variable for one of the levels. So, for instance, if there are N levels for a nominal variable, you only have to create *(N-1)* dummy variables, as the Nth one has redundant information. The information regarding to the excluded category is represented in the intercept term. It creates a problem when you put all N dummy variables into the model, because the OLS procedure tries to invert a singular matrix and you will likely get an error message.

On the other hand, this is not an issue when you fit a regularized regression model, which will be the case in this class. Therefore, you do not need to drop one of the dummy variables and can include all of them in the analysis. In fact, it may be beneficial to keep the dummy variables for all categories in the model when regularization is used in regression. Otherwise, the model may produce different predictions depending on which category is excluded.

2.2. Label encoding

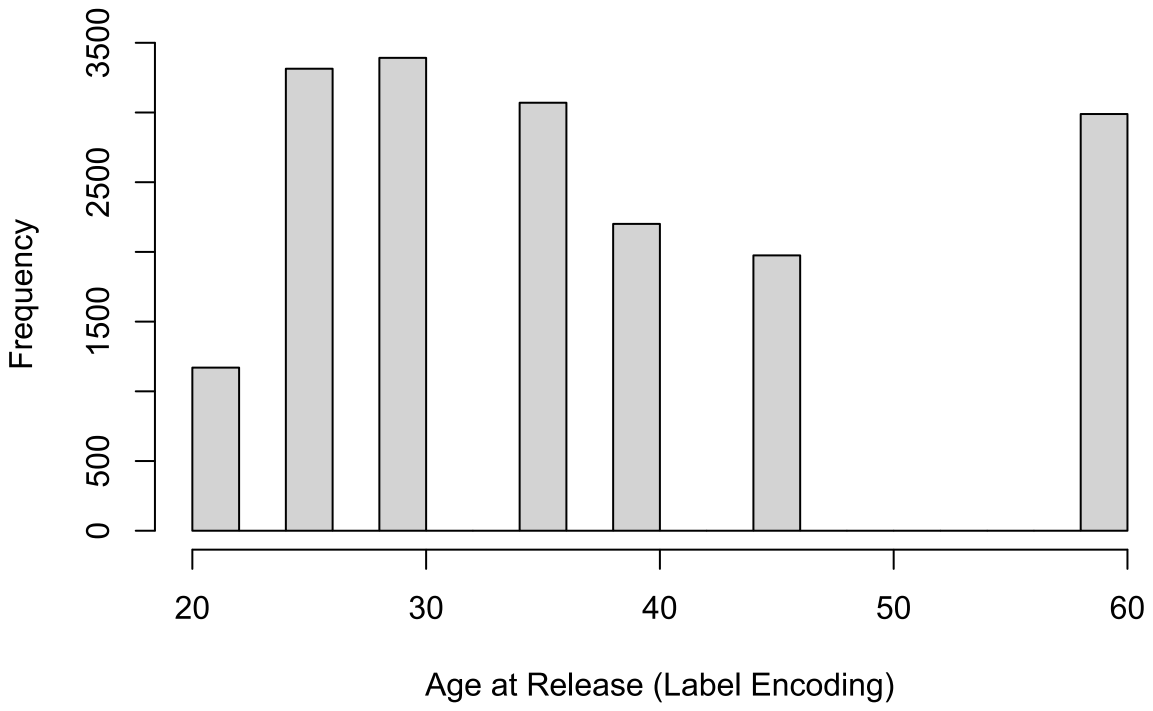
When the variable of interest is ordinal and there is a hierarchy among the levels, we can still use one-hot encoding to create a set of dummy variables to represent the information in the ordinal variable. However, dummy variables will not provide any information regarding the hierarchy among categories.

For instance, consider the variable `Age_At_Release` in the Recidivism dataset. It is coded as 7 different age intervals in the dataset: 18-22, 23-27, 28-32, 33-37, 38-42, 43-47, 48 or older. One can create 7 dummy variables to represent each category in this variable. Alternatively, one can assign a numeric variable to each category that may represent the information in these categories. For instance, one can assign numbers from 1 to 7, respectively. Or, one can choose the midpoint of each interval to represent each category (e.g., 20,25,31,35,40,45,60).

```
require(dplyr)
table(recidivism$Age_at_Release)
```

```


      18-22       23-27       28-32       33-37       38-42       43-47 
       1170        3314        3392        3070        2201        1975 
48 or older 
       2989 
```

```
#?dplyr::recode
recidivism$age <- recode(recidivism$Age_at_Release,
                        '18-22' = 20,
                        '23-27' = 25,
                        '28-32' = 30,
                        '33-37' = 35,
                        '38-42' = 40,
                        '43-47' = 45,
                        '48 or older' = 60)

hist(recidivism$age,main='',xlab='Age at Release (Label Encoding)')
```



Another example would be the variable `Education Level`. It has three different levels: At least some college, High School Diploma, Less than HS diploma. One can create three dummy variables to represent each level. Alternatively, one can assign 1, 2, and 3, respectively. Or, one can assign a number for the approximate years of schooling for each level such as 9, 12, and 15.

```
table(recidivism$Education_Level)

recidivism$edu <- recode(recidivism$Education_Level,
                         'At least some college' = 14,
                         'High School Diploma'   = 12,
                         'Less than HS diploma'  = 8)

hist(recidivism$edu,main='',xlab='Education Level (Label Encoding)')
```

2.3. Polynomial Contrasts

Another way of encoding an ordinal variable is to use polynomial contrasts. The polynomial contrasts may be helpful if one wants to explore whether or not there is a linear, quadratic, cubic, etc. relationship between the predictor variable and outcome variable. You can use `stat::poly()` function in R to obtain the set of polynomial contrasts. If there are N levels in an ordinal variable, then you can get polynomials up to degree *N-1*.

For instance, suppose you have an ordinal variable with three levels: Low, Medium, and High. Then, `stat::poly(x=1:3,degree=2)` will return the polynomial contrasts for the linear and quadratic terms. Notice that the input values for the `poly` function is a vector of numeric values that correspond to the levels in the ordinal variable, and the degree of polynomial terms requested. For this examle, it creates two sets of vectors to represent this ordinal variable. Note that the sum of the squares within each column is equal to 1, and the dot product of the contrast vectors is equal to 0. In other words, the polynomial terms represent a set of **orthonormal vectors**.

	Linear	Quadratic
Low	-0.707	0.408
Medium	0	-0.816
High	0.707	0.408

```
ctr <- poly(1:3,2)
ctr
```

```
                                1          2
[1,] -0.70710678118654757273731  0.4082483
[2,] -0.00000000000000007850462 -0.8164966
[3,]  0.70710678118654746171501  0.4082483
attr(,"coefs")
attr(,"coefs")$alpha
[1] 2 2

attr(,"coefs")$norm2
[1] 1.0000000 3.0000000 2.0000000 0.6666667

attr(,"degree")
[1] 1 2
attr(,"class")
[1] "poly"   "matrix"
```

```
sum(ctr[,1]^2)
```

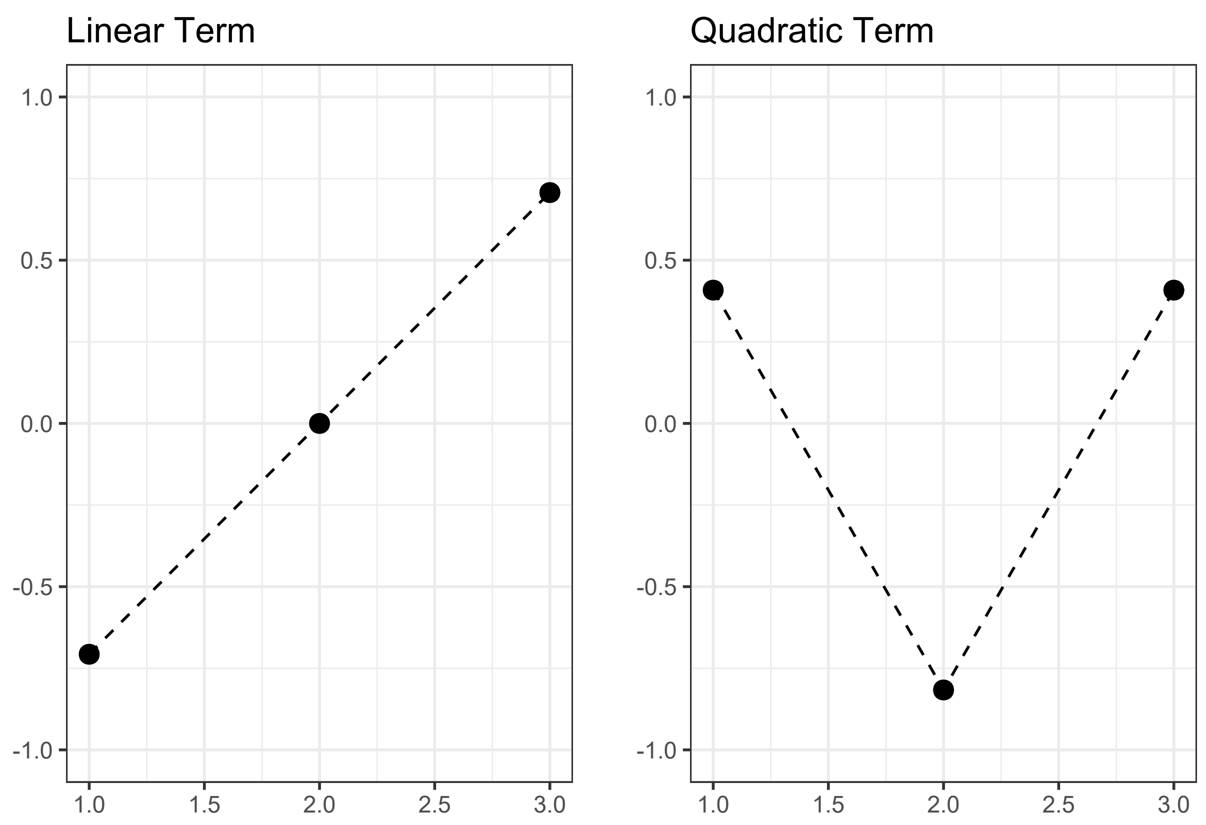
```
[1] 1
```

```
sum(ctr[,2]^2)
```

```
[1] 1
```

```
sum(ctr[,1]*ctr[,2])
```

```
[1] 0.00000000000000006410345
```



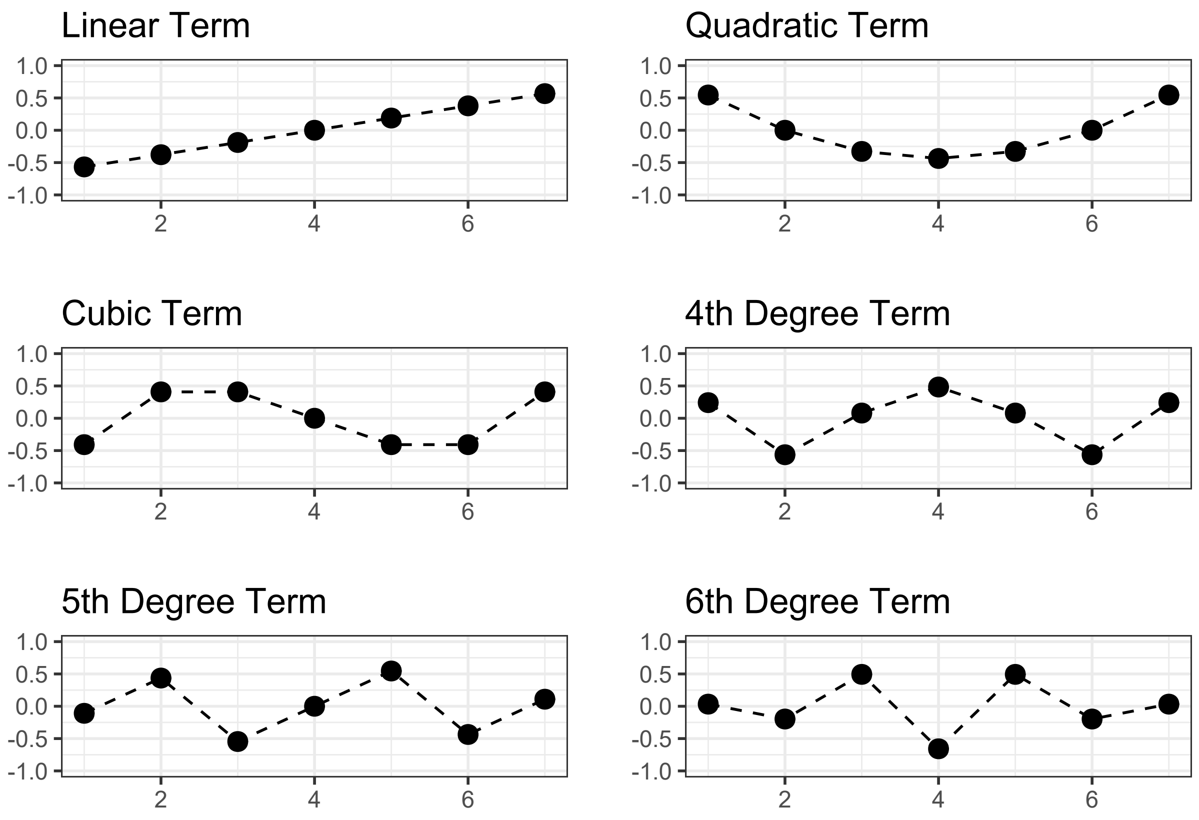
If we consider the variable `Age_At_Release` with 7 different levels, then we can have polynomial terms up to the 6th degree.

```
ctr <- poly(1:7,6)
ctr
```

```
                               1                          2
[1,] -0.5669467095138408518906 2  0.5455447255899807945667 1
```

```
                                1                         2
[1,] -0.56694670951384085189062  0.54554472558998079456671
[2,] -0.37796447300922730860862  0.00000000000000009690821
[3,] -0.18898223650461357103758 -0.32732683535398837681996
[4,]  0.00000000000000002098124 -0.43643578047198489100467
[5,]  0.18898223650461357103758 -0.32732683535398865437571
[6,]  0.37796447300922719758631  0.00000000000000000000000
[7,]  0.56694670951384085189062  0.54554472558998101661132
                                3          4                          5
[1,] -0.40824829046386312825234  0.2417469 -0.10910894511799962505067
[2,]  0.40824829046386318376349 -0.5640761  0.43643578047198505753 81
[3,]  0.40824829046386312825234  0.0805823 -0.54554472558998079456 67
[4,]  0.00000000000000003021644  0.4834938 -0.00000000000000009751389
[5,] -0.40824829046386268416313  0.0805823  0.54554472558998123865559
[6,] -0.40824829046386301723004 -0.5640761 -0.43643578047198411384 85
[7,]  0.40824829046386279518543  0.2417469  0.10910894511799959313176
               6
[1,]  0.03289758
[2,] -0.19738551
```

```
[3,]  0.49346377
[4,] -0.65795169
[5,]  0.49346377
[6,] -0.19738551
[7,]  0.03289758
attr(,"coefs")
attr(,"coefs")$alpha
[1] 4 4 4 4 4 4

attr(,"coefs")$norm2
[1]   1.0000   7.0000  28.0000  84.0000 216.0000 452.5714 685.7143 561.0390

attr(,"degree")
[1] 1 2 3 4 5 6
attr(,"class")
[1] "poly"   "matrix"
```



When we apply this to a variable in the dataset, these columns are rescaled such that they are orthonormal vectors. See below for an example:

```
# First, recode the Age_at_Release so that its levels are represented by numbers
# poly() doesn't work with character vectors

recidivism$age <- dplyr::recode(recidivism$Age_at_Release,
                                '18-22' = 1,
                                '23-27' = 2,
                                '28-32' = 3,
                                '33-37' = 4,
                                '38-42' = 5,
                                '43-47' = 6,
                                '48 or older' = 7)

# Create polynomial terms

poly.mat <- poly(x = recidivism$age, degree=6)
head(poly.mat)
```

```
                 1            2             3              4            5
[1,]  0.0075566965 -0.001134569 -0.0094993975 -0.01283641494 -0.010472332
[2,] -0.0003464813 -0.008158592 -0.0005626538  0.00844393307  0.001808606
[3,]  0.0115082853  0.009369599  0.0064071011  0.00381687318  0.001776355
[4,]  0.0036051076 -0.006977299 -0.0089263309 -0.00006546544  0.010800278
[5,]  0.0036051076 -0.006977299 -0.0089263309 -0.00006546544  0.010800278
[6,]  0.0115082853  0.009369599  0.0064071011  0.00381687318  0.001776355
                 6
[1,] -0.0052954033
[2,] -0.0113555065
[3,]  0.0005831617
[4,]  0.0118791703
[5,]  0.0118791703
[6,]  0.0005831617
```

```
# Append them to the original data

recidivism$age_poly1 <- poly.mat[,1]
recidivism$age_poly2 <- poly.mat[,2]
recidivism$age_poly3 <- poly.mat[,3]
recidivism$age_poly4 <- poly.mat[,4]
recidivism$age_poly5 <- poly.mat[,5]
recidivism$age_poly6 <- poly.mat[,6]

head(recidivism[,c('Age_at_Release','age','age_poly1','age_poly2',
                   'age_poly3','age_poly4','age_poly5','age_poly6')],10)
```

```
   Age_at_Release age     age_poly1    age_poly2     age_poly3      age_poly4
1           43-47   6  0.0075566965 -0.001134569 -0.0094993975 -0.01283641494
2           33-37   4 -0.0003464813 -0.008158592 -0.0005626538  0.00844393307
3     48 or older   7  0.0115082853  0.009369599  0.0064071011  0.00381687318
4           38-42   5  0.0036051076 -0.006977299 -0.0089263309 -0.00006546544
5           38-42   5  0.0036051076 -0.006977299 -0.0089263309 -0.00006546544
6     48 or older   7  0.0115082853  0.009369599  0.0064071011  0.00381687318
7           38-42   5  0.0036051076 -0.006977299 -0.0089263309 -0.00006546544
8           43-47   6  0.0075566965 -0.001134569 -0.0094993975 -0.01283641494
9     48 or older   7  0.0115082853  0.009369599  0.0064071011  0.00381687318
10          33-37   4 -0.0003464813 -0.008158592 -0.0005626538  0.00844393307
      age_poly5     age_poly6
1  -0.010472332 -0.0052954033
2   0.001808606 -0.0113555065
```

```
3   0.001776355  0.0005831617
4   0.010800278  0.0118791703
5   0.010800278  0.0118791703
6   0.001776355  0.0005831617
7   0.010800278  0.0118791703
8  -0.010472332 -0.0052954033
9   0.001776355  0.0005831617
10  0.001808606 -0.0113555065
```

Notice that the values assigned to each category are different than the ones you see above, because they are rescaled such that polynomial terms have a unit length (sum of squared values equal to 1) and they are independent of each other (dot product is zero).

```r
# Length of each polynomial term (sum of square values)

colSums(recidivism[,c('age_poly1','age_poly2','age_poly3','age_poly4','age_poly5','age_poly6')]^2)
```

```
age_poly1 age_poly2 age_poly3 age_poly4 age_poly5 age_poly6 
        1         1         1         1         1         1 
```

```r
# The dot product is zero for each pair of 6 terms, see for a few examples

sum(recidivism[,c('age_poly1')]*recidivism[,c('age_poly2')])
```

```
[1] -0.00000000000004808991
```

```r
sum(recidivism[,c('age_poly1')]*recidivism[,c('age_poly3')])
```

```
[1] -0.000000000000007869722
```

```r
sum(recidivism[,c('age_poly4')]*recidivism[,c('age_poly6')])
```

```
[1] 0.0000000000000119
```

NOTE

There are other ways of encoding nominal and ordinal variables (e.g., Helmert contrasts), or one can come up with their own set of contrast values. When the goal of analysis is inference and you run analysis to respond to a specific research question, your research question typically dictates the type of encoding to use. You choose a coding scheme that provides you the most interpretable coefficients to respond to your research question.

On the other hand, when the goal of analysis is prediction, how you encode your categorical variable does not make much difference. In fact, they provide very similar predictions. Below, I provide an example using Age_at_Release variable to predict the outcome using different coding schemes and report the average squared error of predictions from a logistic regression model.

Encoding	Average Squared Error
Intercept-Only (NULL)	0.1885789
Dummy Variables	0.1861276
Label Encoding	0.1861888
Polynomial Contrasts	0.1861276
Helmert Contrasts	0.1861276

Notice that one-hot encoding, polynomial contrasts, and helmert contrasts have identical performance. In fact, they yield the exact same predicted value for observations. Moreover, a simple label encoding with a single constructed variable does (almost) as well as other encoding types with multiple constructed variables.

---

3. Processing Cyclic Variables

There are sometimes variables that are cyclic by nature (e.g., months, days, hour), and a type of encoding that represents their cyclic nature may be the most meaningful way to represent them instead of numerical or categorical encoding. One way to achieve this is to create two new variables using a sine and cosine transformation as the following:

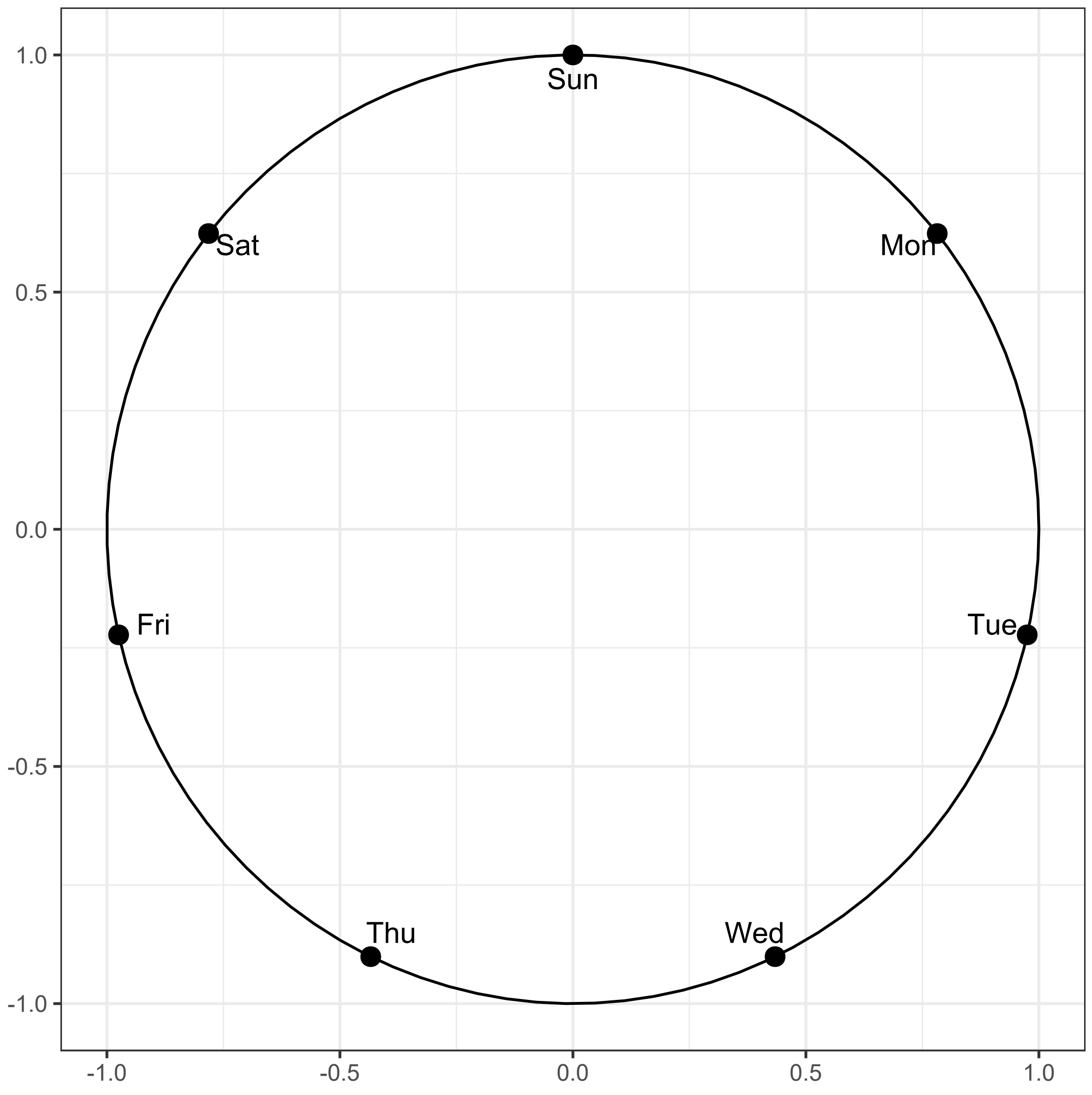
$$x_1 = sin(\frac{2\pi x}{max(x)}),$$

$$x_2 = cos(\frac{2\pi x}{max(x)}).$$

For instance, suppose one of the variables in a dataset is the day of the week. We can represent its cyclic nature using the two variables as defined the following. Once the corresponding coordinates are calculated for each day of the week, the single variable that represents days in the data can be replaced with these two variables representing their coordinates in a unit circle.

```
d <- data.frame(days = c('Mon','Tue','Wed','Thu','Fri','Sat','Sun'),
                x = 1:7)

d$x1 <- sin((2*pi*d$x)/7)
d$x2 <- cos((2*pi*d$x)/7)

d
```

```
  days x                         x1         x2
1  Mon 1  0.78183148246802980363 41  0.6234898
2  Tue 2  0.97492791218182361934 20 -0.2225209
3  Wed 3  0.43388373911755823142 40 -0.9009689
4  Thu 4 -0.43388373911755800937 94 -0.9009689
5  Fri 5 -0.97492791218182361934 20 -0.2225209
6  Sat 6 -0.78183148246802991465 64  0.6234898
7  Sun 7 -0.00000000000000024492 13  1.0000000
```

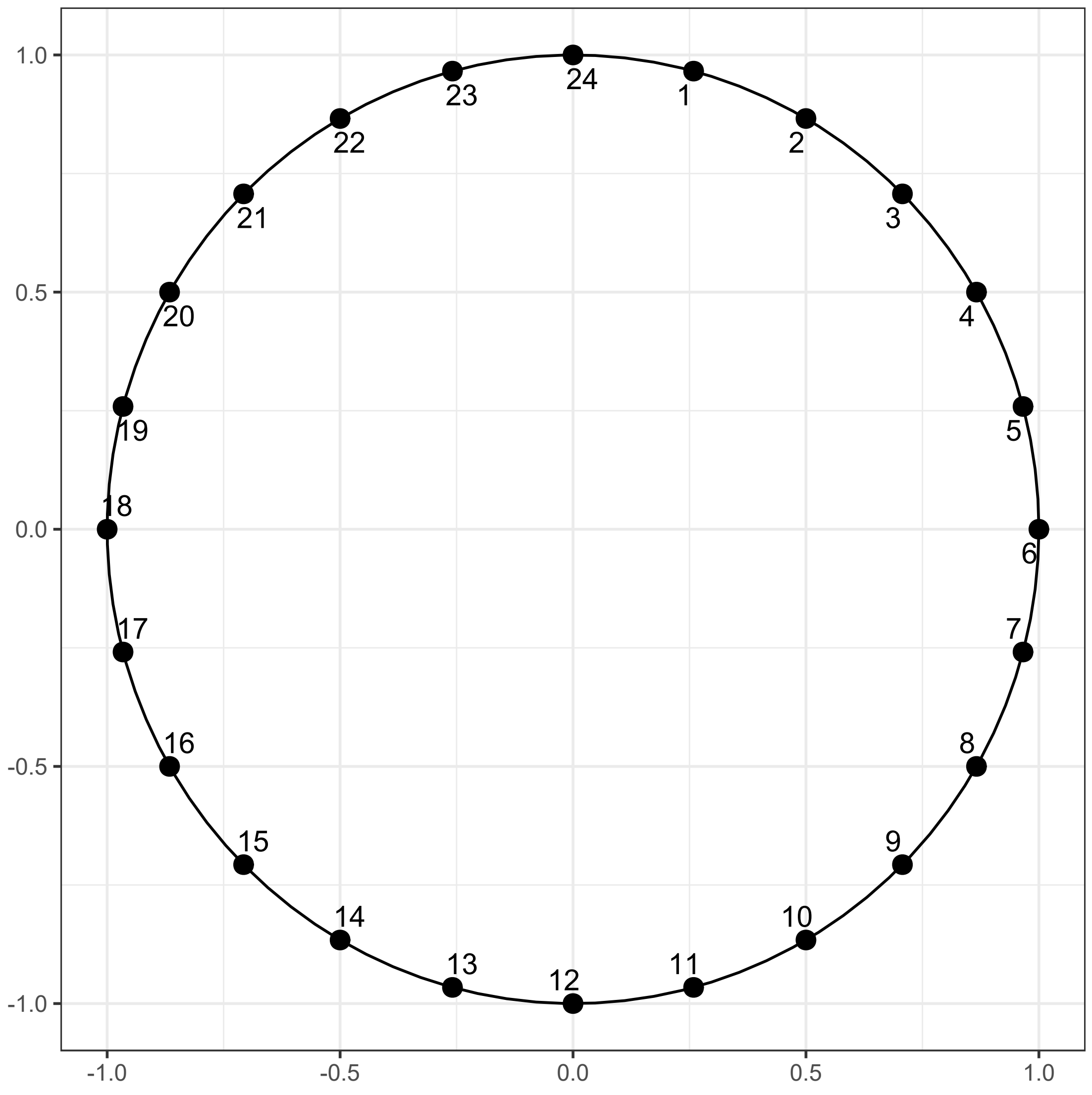


We can apply the same concept to any cyclic variable. Here is another example for the time of day.

```
d <- data.frame(hour = 1:24)

d$x1 <- sin((2*pi*d$hour)/24)
d$x2 <- cos((2*pi*d$hour)/24)

d
```

```
   hour                         x1                         x2
1     1  0.25881904510252073947640  0.96592582628906831221371
2     2  0.49999999999999994448884  0.86602540378443870761060
3     3  0.70710678118654746171500  0.70710678118654757273731
4     4  0.86602540378443859658830  0.50000000000000011102230
5     5  0.96592582628906831221371  0.25881904510252073947640
6     6  1.00000000000000000000000  0.00000000000000006123032
```

```
7     7  0.9659258262890683122137 -0.25881904510252062845410
8     8  0.8660254037844387076106 -0.49999999999999977795540
9     9  0.7071067811865475727373 -0.70710678118654746171501
10   10  0.4999999999999999444888 -0.86602540378443870761060
11   11  0.2588190451025210170322 -0.96592582628906820119141
12   12  0.0000000000000001224606 -1.0000000000000000000000
13   13 -0.2588190451025207949876 -0.96592582628906831221371
14   14 -0.4999999999999997224442 -0.86602540378443881863291
15   15 -0.7071067811865471286481 -0.70710678118654790580422
16   16 -0.8660254037844383745437 -0.50000000000000044408921
17   17 -0.9659258262890683122137 -0.25881904510252062845410
18   18 -1.0000000000000000000000 -0.00000000000000018369095
19   19 -0.9659258262890684232360  0.25881904510252029538719
20   20 -0.8660254037844385965883  0.50000000000000011102230
21   21 -0.7071067811865476837596  0.70710678118654735069271
22   22 -0.5000000000000004440892  0.86602540378443837454370
23   23 -0.2588190451025215721437  0.96592582628906809016911
24   24 -0.0000000000000002449213  1.0000000000000000000000
```



4. Processing Continuous Variables

4.1 Centering and Scaling (Standardization)

Centering a variable is done by substracting the mean of the variable from every value of the variable. This ensures that the mean of the centered variable is equal to zero. Scaling a variable is by dividing every value of variable by its standard deviation. When centering and scaling are both applied, this is called standardizing a variable. When we standardize a variable, we make sure that its mean is equal to zero and variance is equal to 1. Standardization may be important for certain types of models (e.g., K-nearest neighbor, support vector machines, penalized regression).

Below is an example from the Recidivism dataset using the variable `Avg_Days_per_DrugTest`.

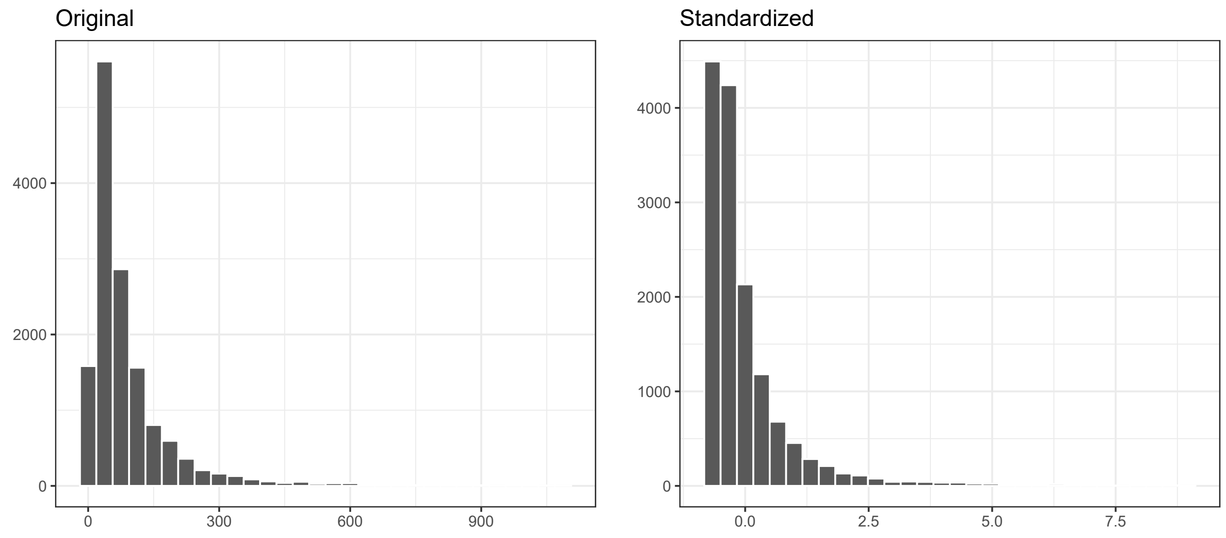
```
require(psych)
```

```r
# Descriptive stats for the original variable

describe(recidivism$Avg_Days_per_DrugTest)
```

```
   vars     n  mean     sd median trimmed   mad  min    max   range skew
X1    1 14343 92.41 113.11  56.03   69.29 47.28 1.27 1088.5 1087.23 3.64
   kurtosis   se
X1    18.24 0.94
```

```r
  # Center

  recidivism$Avg_Days_per_DrugTest2 <- recidivism$Avg_Days_per_DrugTest - mean(recidivism$Avg_Days_per_I

  # Scale

  recidivism$Avg_Days_per_DrugTest2 <- recidivism$Avg_Days_per_DrugTest2/sd(recidivism$Avg_Days_per_Dru

# Descriptive stats for the standardized variable

describe(recidivism$Avg_Days_per_DrugTest2)
```

```
   vars     n mean sd median trimmed  mad   min  max range skew kurtosis   se
X1    1 14343    0  1  -0.32    -0.2 0.42 -0.81 8.81  9.61 3.64    18.24 0.01
```



NOTE

Standardizing a variable only changes the first and second moments of a distribution (mean and variance); however, it doesn't change the third and fourth moments of a distribution (skewness and kurtosis). Notice that the skewness and kurtosis for both the original and the standardized variable are 3.64 and 18.24, respectively. We only change the mean to zero and variance to one by standardizing a variable.

4.2 Box-Cox transformation

```
require(MASS)

bx <- boxcox(Avg_Days_per_DrugTest ~ 1,data=recidivism,
             plotit=FALSE)

bx
```

```
$x
 [1] -2.0 -1.9 -1.8 -1.7 -1.6 -1.5 -1.4 -1.3 -1.2 -1.1 -1.0 -0.9 -0.8 -0.7 -0.6
[16] -0.5 -0.4 -0.3 -0.2 -0.1  0.0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9
[31]  1.0  1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9  2.0

$y
 [1] -105172.00 -101374.00  -97757.01  -94334.16  -91116.04  -88110.22
 [7]  -85321.10  -82750.12  -80396.26  -78256.69  -76327.48  -74604.24
[13]  -73082.60  -71758.61  -70628.94  -69691.01  -68943.03  -68383.94
[19]  -68013.38  -67831.55  -67838.98  -68036.39  -68424.37  -69003.10
[25]  -69772.06  -70729.69  -71873.23  -73198.50  -74699.87  -76370.38
[31]  -78201.87  -80185.26  -82310.85  -84568.65  -86948.66  -89441.10
[37]  -92036.64  -94726.48  -97502.47 -100357.12 -103283.61
```

```
which.max(bx$y)
```

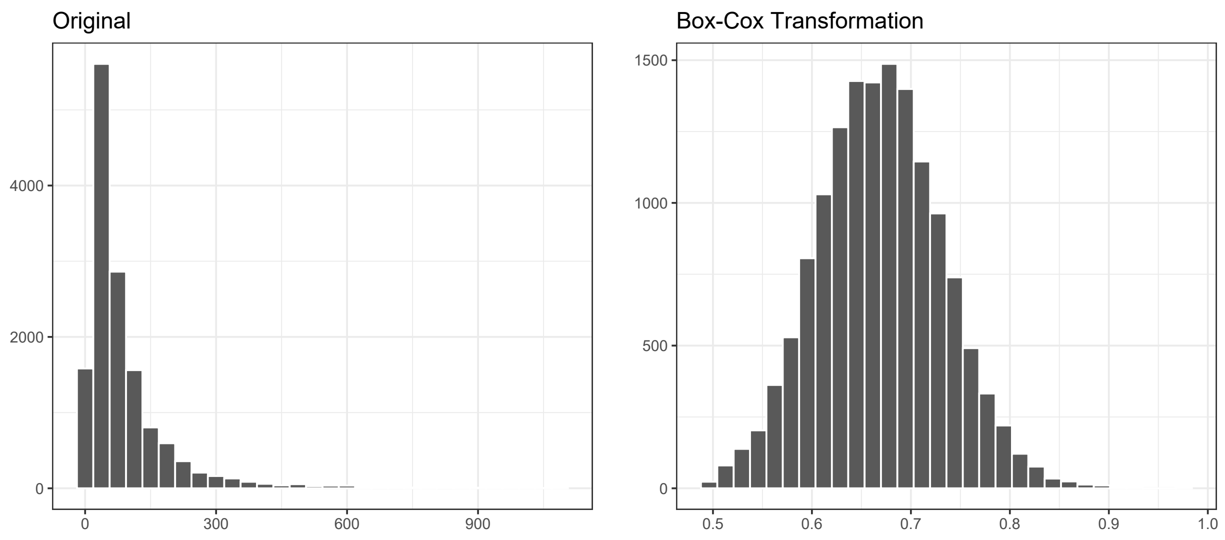
```
[1] 20
```

```
l <- bx$x[which.max(bx$y)]

l
```

```
[1] -0.1
```

```
recidivism$Avg_Days_per_DrugTest_norm <- recidivism$Avg_Days_per_DrugTest^l

describe(recidivism$Avg_Days_per_DrugTest_norm)
```

```
   vars     n mean   sd median trimmed  mad min  max range skew kurtosis se
X1    1 14343 0.67 0.06   0.67    0.67 0.06 0.5 0.98  0.48 0.13     0.03  0
```



NOTE

Apply Box-Cox first, and then standardize.

4.3 Yeo-Johnson transformation

4.4. Logit Transformation

4.5 Basis Expansions and Splines

4.6 Principal Component Analysis

Handling Missing Data

Data Leakage

Processing Text Data

All-in-one using the `recipes` package