Data Pre-processing (Feature Engineering)

Applied Machine Learning for Educational Data Science

true

08/23/2021

Contents

[Updated: Thu, Aug 26, 2021 - 17:51:42]

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**.

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
```

```
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            0        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).

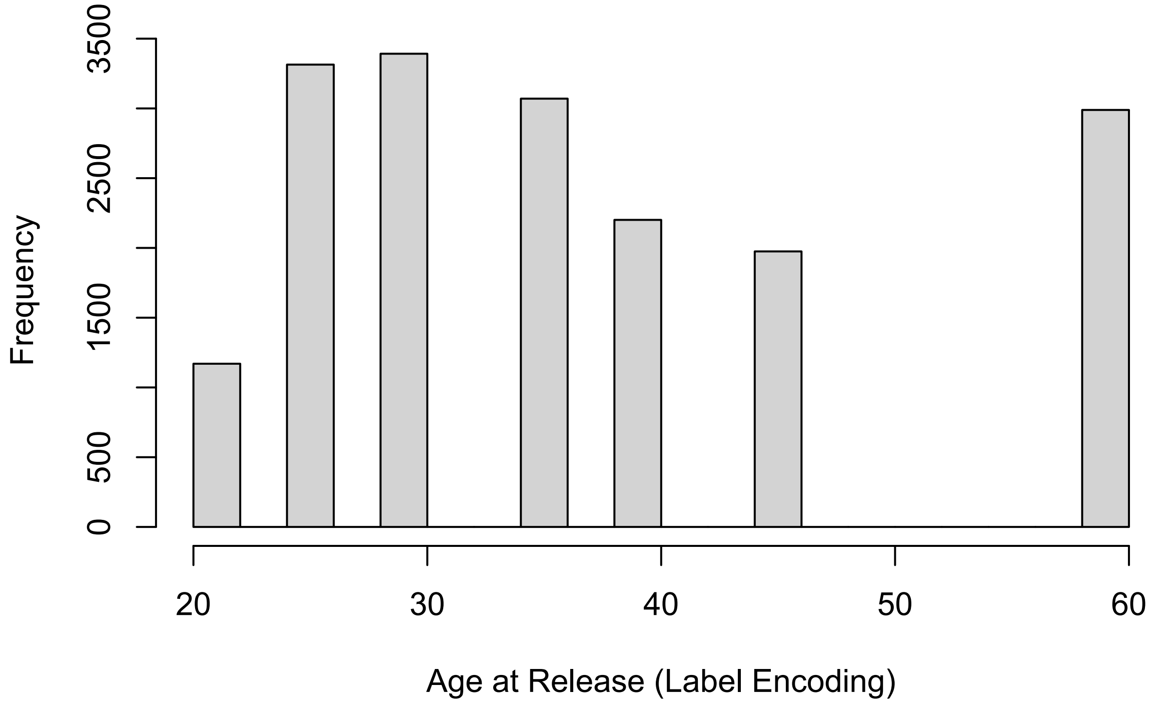
```
require(car)
table(recidivism$Age_at_Release)
```

```

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

```
#?car::recode
recidivism$age <- recode(recidivism$Age_at_Release,
                         recodes = "'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,
                         recodes = "’At least some college’ = 14;
                                    ’High School Diploma’   = 12;
                                    ’Less than HS diploma’  = 8")

hist(recidivism$edu,main=’’,xlab=’Age at Release (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 there is a linear, quadratic, cubic, etc. relationship between the predictor variable and outcome variable. You can use `contr.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, `contr.poly(3)` will return the polynomial contrasts for the linear and quadratic terms. Notice that the input value for the `contr.poly` function is the number of levels in the ordinal variable, and 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. So, the polynomial contrasts represent a set of orthonormal vectors.

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

```
ctr <- contr.poly(3)
ctr
```

```
                             .L         .Q
[1,] -0.70710678118654757273731  0.4082483
[2,] -0.00000000000000007850462 -0.8164966
[3,]  0.70710678118654746171501  0.4082483
```

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

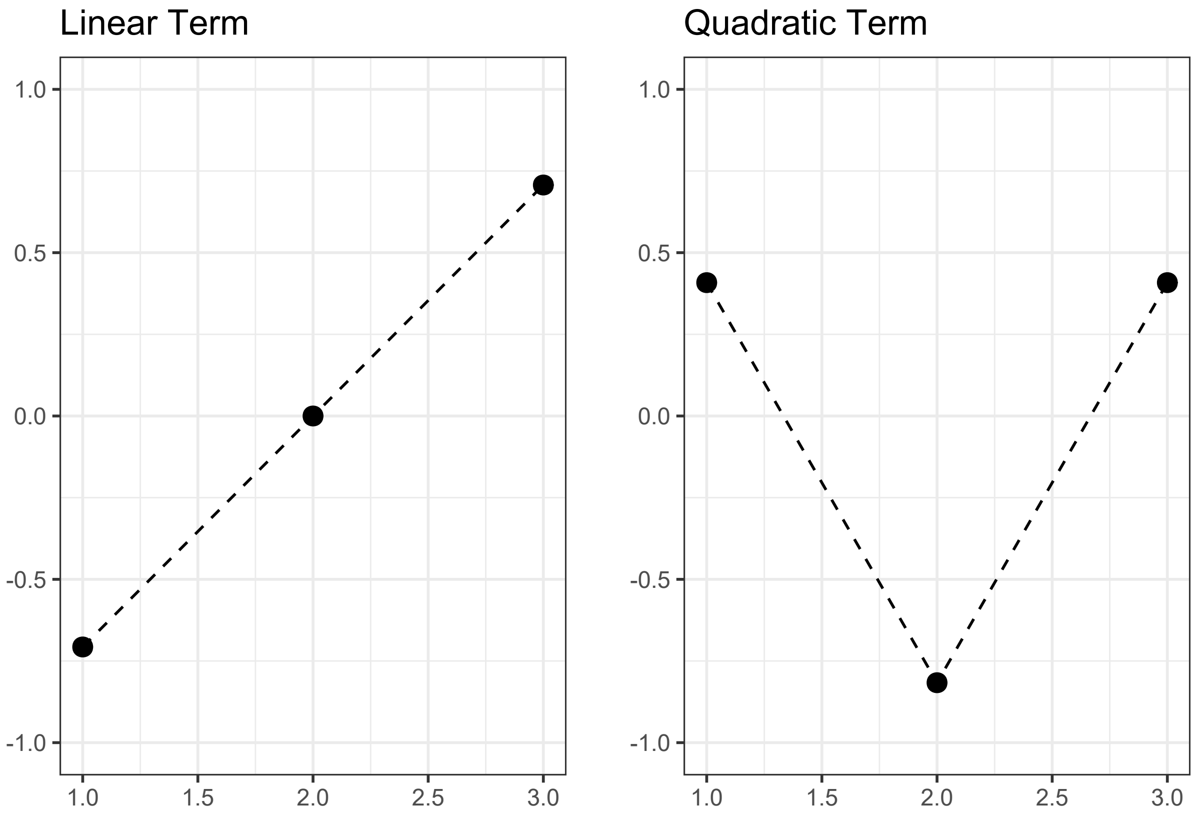
```
[1] 1
```

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

```
[1] 1
```

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

```
[1] 0.0000000000000000006410345
```

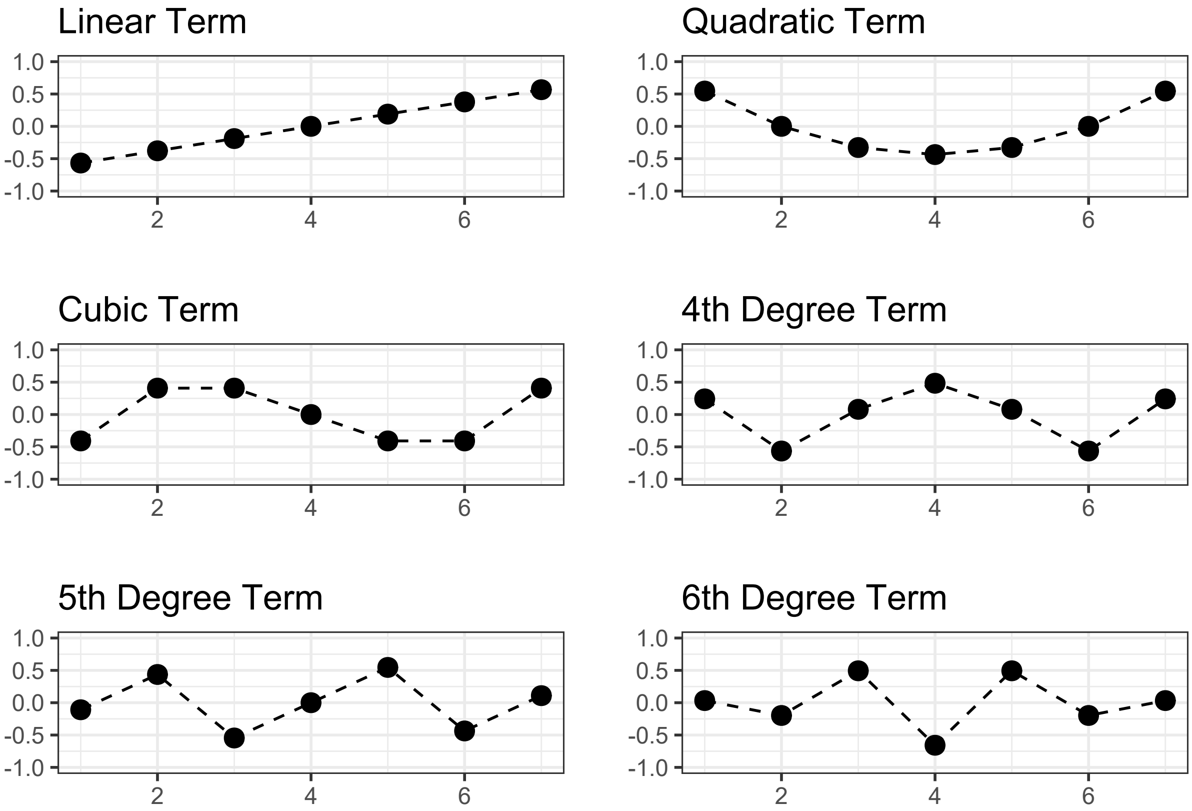


If we consider the variable `Age_At_Release` with 7 different levels, then we can have polynomial ters up to the 6th degree.

```
ctr <- contr.poly(7)
ctr
```

```
                             .L                         .Q
[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
                             .C         ^4                         ^5
[1,] -0.40824829046386312825234  0.2417469 -0.10910894511799962505067
[2,]  0.40824829046386318376349 -0.5640761  0.4364357804719850575381
[3,]  0.40824829046386312825234  0.0805823 -0.54554472558998807945667
[4,]  0.00000000000000003021644  0.4834938 -0.00000000000000009751389
[5,] -0.40824829046386268416313  0.0805823  0.54554472558998812386559
```

```
[6,] -0.40824829046386301723004 -0.5640761 -0.4364357804719841138485
[7,]  0.40824829046386279518543  0.2417469  0.1091089451179959313176
              ^6
[1,]  0.03289758
[2,] -0.19738551
[3,]  0.49346377
[4,] -0.65795169
[5,]  0.49346377
[6,] -0.19738551
[7,]  0.03289758
```



We can create 6 new variables to represent the ordinal `Age_at_Release` variable in the dataset using the following coding scheme.

Age	Linear	Quadratic	Qubic	^4	^5	^6
18-22	-0.567	0.546	-0.408	0.242	-0.109	0.033
23-27	-0.378	0.000	0.408	-0.564	0.436	-0.197
28-32	-0.189	-0.327	0.408	0.081	-0.546	0.493
33-37	0.000	-0.436	0.000	0.483	0.000	-0.068
38-42	0.189	-0.327	-0.408	0.081	0.546	0.493
43-47	0.378	0.000	-0.408	-0.564	-0.436	-0.197
48 or older	0.567	0.546	0.408	0.242	0.109	0.033

```
?dplyr::recode

recidivism$agepoly1 <- dplyr::recode(recidivism$Age_at_Release,
                                     ’18-22’ = ctr[1,1],
                                     ’23-27’ = ctr[2,1],
                                     ’28-32’ = ctr[3,1],
                                     ’33-37’ = ctr[4,1],
                                     ’38-42’ = ctr[5,1],
                                     ’43-47’ = ctr[6,1],
                                     ’48 or older’ = ctr[7,1])

recidivism$agepoly2 <- dplyr::recode(recidivism$Age_at_Release,
                                     ’18-22’ = ctr[1,2],
                                     ’23-27’ = ctr[2,2],
                                     ’28-32’ = ctr[3,2],
                                     ’33-37’ = ctr[4,2],
                                     ’38-42’ = ctr[5,2],
                                     ’43-47’ = ctr[6,2],
                                     ’48 or older’ = ctr[7,2])

recidivism$agepoly3 <- dplyr::recode(recidivism$Age_at_Release,
                                     ’18-22’ = ctr[1,3],
                                     ’23-27’ = ctr[2,3],
                                     ’28-32’ = ctr[3,3],
                                     ’33-37’ = ctr[4,3],
                                     ’38-42’ = ctr[5,3],
                                     ’43-47’ = ctr[6,3],
                                     ’48 or older’ = ctr[7,3])

recidivism$agepoly4 <- dplyr::recode(recidivism$Age_at_Release,
                                     ’18-22’ = ctr[1,4],
                                     ’23-27’ = ctr[2,4],
                                     ’28-32’ = ctr[3,4],
                                     ’33-37’ = ctr[4,4],
                                     ’38-42’ = ctr[5,4],
                                     ’43-47’ = ctr[6,4],
                                     ’48 or older’ = ctr[7,4])

recidivism$agepoly5 <- dplyr::recode(recidivism$Age_at_Release,
                                     ’18-22’ = ctr[1,5],
                                     ’23-27’ = ctr[2,5],
                                     ’28-32’ = ctr[3,5],
                                     ’33-37’ = ctr[4,5],
                                     ’38-42’ = ctr[5,5],
                                     ’43-47’ = ctr[6,5],
                                     ’48 or older’ = ctr[7,5])

recidivism$agepoly6 <- dplyr::recode(recidivism$Age_at_Release,
                                     ’18-22’ = ctr[1,6],
                                     ’23-27’ = ctr[2,6],
                                     ’28-32’ = ctr[3,6],
                                     ’33-37’ = ctr[4,6],
                                     ’38-42’ = ctr[5,6],
                                     '43-47' = ctr[6,6],
                                     '48 or older' = ctr[7,6])
```

3. Processing Cyclic Variables

4. Processing Continuous Variables

4.1 Centering and Scaling

4.2 Transformations

4.3 Basis Expansions and Splines

4.4 Linear Projections

Handling Missing Data

Data Leakage

Processing Text Data

All-in-one using the `recipes` package