Covariate creation

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

Two levels of covariate creation

Level 1: From raw data to covariate

HI

WE'VE DISCOVERED YOU ARE THE HEIR TO AN INCREDIBLE FORTUNE. PLEASE SUBMIT YOUR NAME, ADDRESS AND BANK ACCOUNT SO WE CAN SEND YOU $$$$$$$$.

JOE JOHNSON

→

capitalAve	you	numDollar	...
1	2	8	...

Level 2: Transforming tidy covariates

```
library(kernlab);data(spam)
spam$capitalAveSq <- spam$capitalAve^2
```

Level 1, Raw data -> covariates

- Depends heavily on application
- The balancing act is summarization vs. information loss
- Examples:
  - Text files: frequency of words, frequency of phrases (Google ngrams), frequency of capital letters.
  - Images: Edges, corners, blobs, ridges (computer vision feature detection))
  - Webpages: Number and type of images, position of elements, colors, videos (A/B Testing)
  - People: Height, weight, hair color, sex, country of origin.
- The more knowledge of the system you have the better the job you will do.
- When in doubt, err on the side of more features
- Can be automated, but use caution!

Level 2, Tidy covariates -> new covariates

- More necessary for some methods (regression, svms) than for others (classification trees).
- Should be done *only on the training set*
- The best approach is through exploratory analysis (plotting/tables)
- New covariates should be added to data frames

Load example data

```
library(ISLR); library(caret); data(Wage);
inTrain <- createDataPartition(y=Wage$wage,
                              p=0.7, list=FALSE)
training <- Wage[inTrain,]; testing <- Wage[-inTrain,]
```

Common covariates to add, dummy variables

Basic idea - convert factor variables to indicator variables

```
table(training$jobclass)
```

```

 1. Industrial 2. Information 
          1090           1012 
```

```
dummies <- dummyVars(wage ~ jobclass,data=training)
head(predict(dummies,newdata=training))
```

```
       jobclass.1. Industrial jobclass.2. Information
231655                      1                       0
86582                       0                       1
11141                       0                       1
229379                      1                       0
86064                       1                       0
377879                      1                       0
```

Removing zero covariates

```
nsv <- nearZeroVar(training,saveMetrics=TRUE)
nsv
```

```
           freqRatio percentUnique zeroVar   nzv
year           1.029       0.33302   FALSE FALSE
age            1.122       2.80685   FALSE FALSE
sex            0.000       0.04757    TRUE  TRUE
maritl         3.159       0.23787   FALSE FALSE
race           8.529       0.19029   FALSE FALSE
education      1.492       0.23787   FALSE FALSE
region         0.000       0.04757    TRUE  TRUE
jobclass       1.077       0.09515   FALSE FALSE
health         2.452       0.09515   FALSE FALSE
health_ins     2.269       0.09515   FALSE FALSE
logwage        1.198      17.26927   FALSE FALSE
wage           1.185      18.07802   FALSE FALSE
```

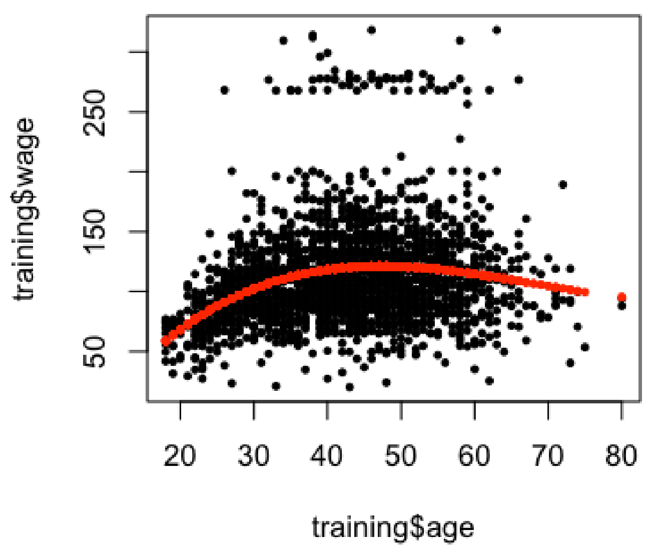
Spline basis

```
library(splines)
bsBasis <- bs(training$age,df=3)
bsBasis
```

```
              1         2         3
 [1,] 0.00000 0.0000000 0.000e+00
 [2,] 0.23685 0.0253768 9.063e-04
 [3,] 0.44309 0.2436978 4.468e-02
 [4,] 0.43081 0.2910904 6.556e-02
 [5,] 0.42617 0.1482327 1.719e-02
 [6,] 0.41709 0.1331149 1.416e-02
 [7,] 0.31823 0.0540390 3.059e-03
 [8,] 0.36253 0.3866940 1.375e-01
 [9,] 0.44436 0.2275981 3.886e-02
[10,] 0.20449 0.0179375 5.245e-04
[11,] 0.07768 0.3601465 5.566e-01
[12,] 0.13145 0.0066841 1.133e-04
[13,] 0.39290 0.1042387 9.218e-03
[14,] 0.26654 0.0339238 1.439e-03
[15,] 0.20449 0.0179375 5.245e-04
[16,] 0.29109 0.4308138 2.125e-01
[17,] 0.23685 0.0253768 9.063e-04
[18,] 0.43624 0.2755195 5.800e-02
```

Fitting curves with splines

```
lm1 <- lm(wage ~ bsBasis,data=training)
plot(training$age,training$wage,pch=19,cex=0.5)
points(training$age,predict(lm1,newdata=training),col="red",pch=19,cex=0.5)
```



Splines on the test set

```
predict(bsBasis,age=testing$age)
```

```
              1         2         3
 [1,] 0.00000 0.0000000 0.000e+00
 [2,] 0.23685 0.0253768 9.063e-04
 [3,] 0.44309 0.2436978 4.468e-02
 [4,] 0.43081 0.2910904 6.556e-02
 [5,] 0.42617 0.1482327 1.719e-02
 [6,] 0.41709 0.1331149 1.416e-02
 [7,] 0.31823 0.0540390 3.059e-03
 [8,] 0.36253 0.3866940 1.375e-01
 [9,] 0.44436 0.2275981 3.886e-02
[10,] 0.20449 0.0179375 5.245e-04
[11,] 0.07768 0.3601465 5.566e-01
[12,] 0.13145 0.0066841 1.133e-04
[13,] 0.39290 0.1042387 9.218e-03
[14,] 0.26654 0.0339238 1.439e-03
[15,] 0.20449 0.0179375 5.245e-04
[16,] 0.29109 0.4308138 2.125e-01
[17,] 0.23685 0.0253768 9.063e-04
[18,] 0.43624 0.2755195 5.800e-02
[19,] 0.26654 0.0339238 1.439e-03
[20,] 0.32118 0.4163380 1.799e-01
```

Notes and further reading

- Level 1 feature creation (raw data to covariates)
  - Science is key. Google "feature extraction for [data type]"
  - Err on overcreation of features
  - In some applications (images, voices) automated feature creation is possible/necessary
  - http://www.cs.nyu.edu/~yann/talks/lecun-ranzato-icml2013.pdf
- Level 2 feature creation (covariates to new covariates)
  - The function *preProcess* in *caret* will handle some preprocessing.
  - Create new covariates if you think they will improve fit
  - Use exploratory analysis on the training set for creating them
  - Be careful about overfitting!
- preprocessing with caret
- If you want to fit spline models, use the *gam* method in the *caret* package which allows smoothing of multiple variables.
- More on feature creation/data tidying in the Obtaining Data course from the Data Science course track.