Practical Machine Learning

Table of Contents

[Caret package 3](#_Toc411344353)

[**Caret functionality** 3](#_Toc411344354)

[**Machine learning algorithms in R** 3](#_Toc411344355)

[Why caret? 3](#_Toc411344356)

[SPAM Example: Data splitting 4](#_Toc411344357)

[SPAM Example: Fit a model 4](#_Toc411344358)

[SPAM Example: Final model 4](#_Toc411344359)

[SPAM Example: Prediction 4](#_Toc411344360)

[**Further information** 4](#_Toc411344361)

[Data Slicing 5](#_Toc411344362)

[**SPAM Example: Data splitting** 5](#_Toc411344363)

[**SPAM Example: K-fold** 5](#_Toc411344364)

[**SPAM Example: Return test** 5](#_Toc411344365)

[**SPAM Example: Resampling** 5](#_Toc411344366)

[**SPAM Example: Time Slices** 5](#_Toc411344367)

[Further information 5](#_Toc411344368)

[Training Options 6](#_Toc411344369)

[**SPAM Example** 6](#_Toc411344370)

[**Train options** 6](#_Toc411344371)

[Metric options 6](#_Toc411344372)

[**trainControl** 6](#_Toc411344373)

[**trainControl resampling** 7](#_Toc411344374)

[Further resources 7](#_Toc411344375)

[Plotting predictors 8](#_Toc411344376)

[**Example: predicting wages** 8](#_Toc411344377)

[**Get training/test sets** 8](#_Toc411344378)

[**Feature plot (*caret* package)** 8](#_Toc411344379)

[**Qplot (*ggplot2* package)** 9](#_Toc411344380)

[**Qplot with color (*ggplot2* package)** 9](#_Toc411344381)

[**Add regression smoothers (*ggplot2* package)** 10](#_Toc411344382)

[**cut2, making factors (*Hmisc* package)** 11](#_Toc411344383)

[**Boxplots with cut2** 11](#_Toc411344384)

[**Boxplots with points overlayed** 11](#_Toc411344385)

[**Tables** 12](#_Toc411344386)

[**Density plots** 12](#_Toc411344387)

[Notes and further reading 12](#_Toc411344388)

[Preprocessing 13](#_Toc411344389)

[**Why preprocess?** 13](#_Toc411344390)

[**Standardizing** 13](#_Toc411344391)

[**Standardizing - test set** 13](#_Toc411344392)

[**Standardizing - *preProcess* function** 14](#_Toc411344393)

[**Standardizing - *preProcess* argument** 14](#_Toc411344394)

[**Standardizing - Box-Cox transforms** 14](#_Toc411344395)

[**Standardizing - Imputing data** 15](#_Toc411344396)

[Notes and further reading 15](#_Toc411344397)

[Covariate creation 16](#_Toc411344398)

[Two levels of covariate creation 16](#_Toc411344399)

[Level 1, Raw data -> covariates 16](#_Toc411344400)

[**Level 2, Tidy covariates -> new covariates** 16](#_Toc411344401)

[**Load example data** 17](#_Toc411344402)

[Common covariates to add, dummy variables 17](#_Toc411344403)

[**Removing zero covariates** 17](#_Toc411344404)

[**Spline basis** 18](#_Toc411344405)

[**Fitting curves with splines** 18](#_Toc411344406)

[**Splines on the test set** 19](#_Toc411344407)

[Notes and further reading 19](#_Toc411344408)

# Caret package

<http://caret.r-forge.r-project.org/>

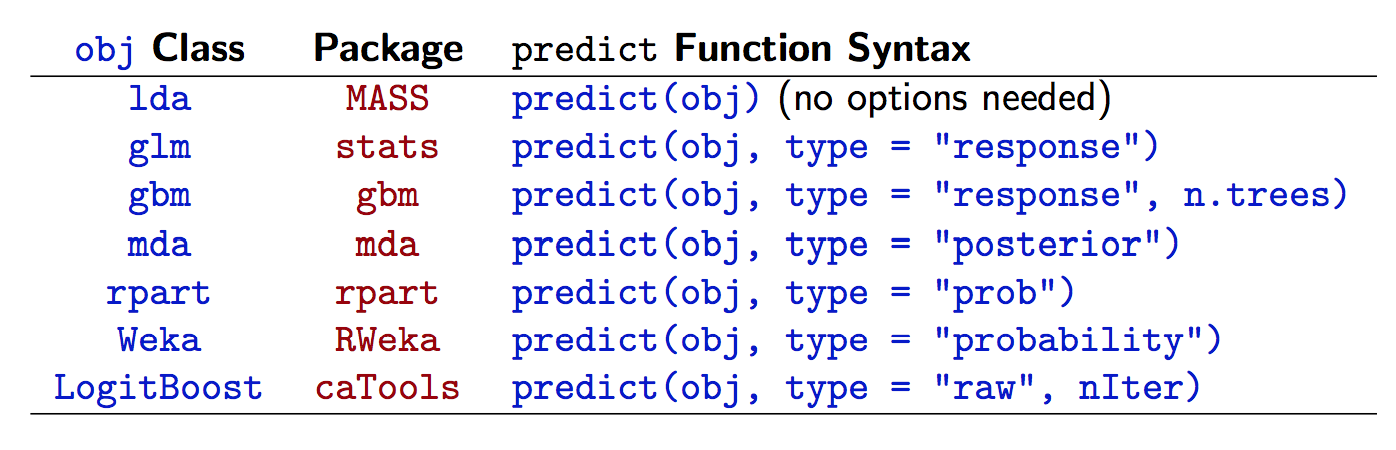
**Caret functionality**

* Some preprocessing (cleaning)
  + preProcess
* Data splitting
  + createDataPartition
  + createResample
  + createTimeSlices
* Training/testing functions
  + train
  + predict
* Model comparison
  + confusionMatrix

**Machine learning algorithms in R**

* Linear discriminant analysis
* Regression
* Naive Bayes
* Support vector machines
* Classification and regression trees
* Random forests
* Boosting
* etc.

## Why caret?



[http://www.edii.uclm.es/~useR-2013/Tutorials/kuhn/user\_caret\_2up.pdf](http://www.edii.uclm.es/%7EuseR-2013/Tutorials/kuhn/user_caret_2up.pdf)

## SPAM Example: Data splitting

|  |
| --- |
| library(caret); library(kernlab); data(spam)  inTrain <- createDataPartition(y=spam$type,  p=0.75, list=FALSE)  training <- spam[inTrain,]  testing <- spam[-inTrain,]  dim(training) |

## SPAM Example: Fit a model

|  |
| --- |
| set.seed(32343)  modelFit <- train(type ~.,data=training, method="glm")  modelFit |

## SPAM Example: Final model

|  |
| --- |
| modelFit <- train(type ~.,data=training, method="glm")  modelFit$finalModel |

## SPAM Example: Prediction

|  |
| --- |
| predictions <- predict(modelFit,newdata=testing)  predictions |

**Further information**

* Caret tutorials:
  + [http://www.edii.uclm.es/~useR-2013/Tutorials/kuhn/user\_caret\_2up.pdf](http://www.edii.uclm.es/%7EuseR-2013/Tutorials/kuhn/user_caret_2up.pdf)
  + <http://cran.r-project.org/web/packages/caret/vignettes/caret.pdf>
* A paper introducing the caret package
  + <http://www.jstatsoft.org/v28/i05/paper>

# Data Slicing

**SPAM Example: Data splitting**

|  |
| --- |
| library(caret); library(kernlab); data(spam)  inTrain <- createDataPartition(y=spam$type,  p=0.75, list=FALSE)  training <- spam[inTrain,]  testing <- spam[-inTrain,]  dim(training) |

**SPAM Example: K-fold**

|  |
| --- |
| set.seed(32323)  folds <- createFolds(y=spam$type,k=10,  list=TRUE,returnTrain=TRUE)  sapply(folds,length) |

**SPAM Example: Return test**

|  |
| --- |
| set.seed(32323)  folds <- createFolds(y=spam$type,k=10,  list=TRUE,returnTrain=FALSE)  sapply(folds,length) |

**SPAM Example: Resampling**

|  |
| --- |
| set.seed(32323)  folds <- createResample(y=spam$type,times=10,  list=TRUE)  sapply(folds,length) |

**SPAM Example: Time Slices**

|  |
| --- |
| set.seed(32323)  tme <- 1:1000  folds <- createTimeSlices(y=tme,initialWindow=20,  horizon=10)  names(folds) |

## Further information

* Caret tutorials:
  + [http://www.edii.uclm.es/~useR-2013/Tutorials/kuhn/user\_caret\_2up.pdf](http://www.edii.uclm.es/%7EuseR-2013/Tutorials/kuhn/user_caret_2up.pdf)
  + <http://cran.r-project.org/web/packages/caret/vignettes/caret.pdf>
* A paper introducing the caret package
  + <http://www.jstatsoft.org/v28/i05/paper>

# Training Options

**SPAM Example**

|  |
| --- |
| library(caret); library(kernlab); data(spam)  inTrain <- createDataPartition(y=spam$type,  p=0.75, list=FALSE)  training <- spam[inTrain,]  testing <- spam[-inTrain,]  modelFit <- train(type ~.,data=training, method="glm") |

**Train options**

|  |
| --- |
| args(train.default)  function (x, y, method = "rf", preProcess = NULL, ..., weights = NULL,  metric = ifelse(is.factor(y), "Accuracy", "RMSE"), maximize = ifelse(metric ==  "RMSE", FALSE, TRUE), trControl = trainControl(), tuneGrid = NULL,  tuneLength = 3)  NULL |

## Metric options

**Continous outcomes**:

* RMSE = Root mean squared error
* RSquared = *R*2 from regression models

**Categorical outcomes**:

* Accuracy = Fraction correct
* Kappa = A measure of [concordance](http://en.wikipedia.org/wiki/Cohen%27s_kappa)

**trainControl**

|  |
| --- |
| args(trainControl)  function (method = "boot", number = ifelse(method %in% c("cv",  "repeatedcv"), 10, 25), repeats = ifelse(method %in% c("cv",  "repeatedcv"), 1, number), p = 0.75, initialWindow = NULL,  horizon = 1, fixedWindow = TRUE, verboseIter = FALSE, returnData = TRUE,  returnResamp = "final", savePredictions = FALSE, classProbs = FALSE,  summaryFunction = defaultSummary, selectionFunction = "best",  custom = NULL, preProcOptions = list(thresh = 0.95, ICAcomp = 3,  k = 5), index = NULL, indexOut = NULL, timingSamps = 0,  predictionBounds = rep(FALSE, 2), seeds = NA, allowParallel = TRUE)  NULL |

**trainControl resampling**

* *method*
  + *boot* = bootstrapping
  + *boot632* = bootstrapping with adjustment
  + *cv* = cross validation
  + *repeatedcv* = repeated cross validation
  + *LOOCV* = leave one out cross validation
* *number*
  + For boot/cross validation
  + Number of subsamples to take
* *repeats*
  + Number of times to repeate subsampling
  + If big this can *slow things down*

## Further resources

* [Caret tutorial](http://www.edii.uclm.es/%7EuseR-2013/Tutorials/kuhn/user_caret_2up.pdf)
* [Model training and tuning](http://caret.r-forge.r-project.org/training.html)

# Plotting predictors

**Example: predicting wages**

Image Credit <http://www.cahs-media.org/the-high-cost-of-low-wages>

Data from: [ISLR package](http://cran.r-project.org/web/packages/ISLR) from the book: [Introduction to statistical learning](http://www-bcf.usc.edu/%7Egareth/ISL/)

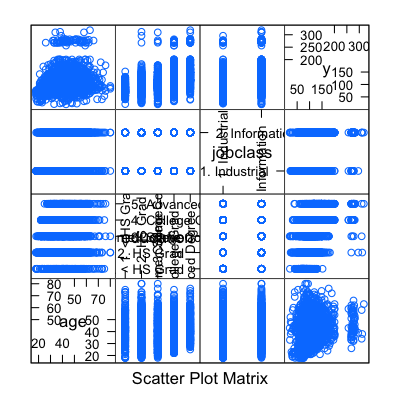
|  |
| --- |
| library(ISLR); library(ggplot2); library(caret);  data(Wage)  summary(Wage) |

**Get training/test sets**

|  |
| --- |
| inTrain <- createDataPartition(y=Wage$wage,  p=0.7, list=FALSE)  training <- Wage[inTrain,]  testing <- Wage[-inTrain,]  dim(training); dim(testing) |

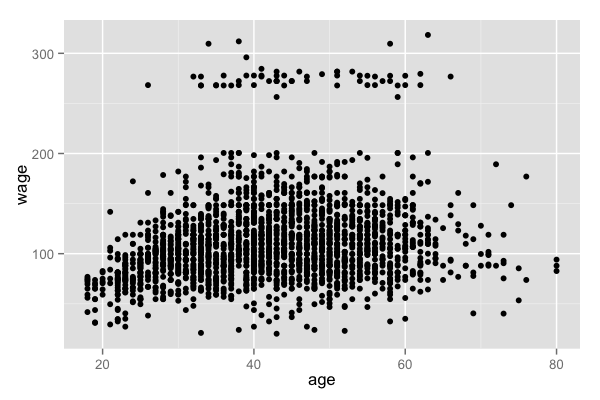
**Feature plot (*caret* package)**

|  |
| --- |
| featurePlot(x=training[,c("age","education","jobclass")],  y = training$wage,  plot="pairs") |



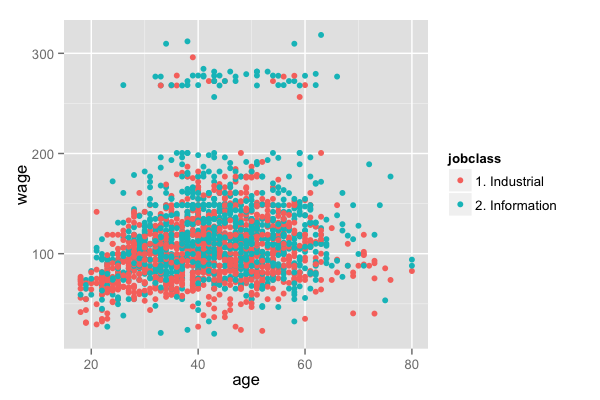
**Qplot (*ggplot2* package)**

|  |
| --- |
| qplot(age,wage,data=training) |



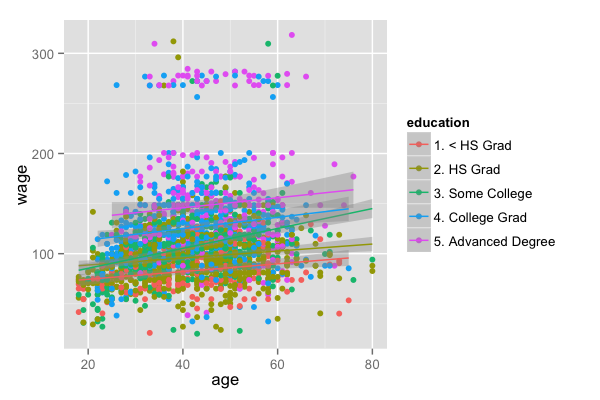
**Qplot with color (*ggplot2* package)**

|  |
| --- |
| qplot(age,wage,colour=jobclass,data=training) |



**Add regression smoothers (*ggplot2* package)**

|  |
| --- |
| qq <- qplot(age,wage,colour=education,data=training)  qq + geom\_smooth(method='lm',formula=y~x) |

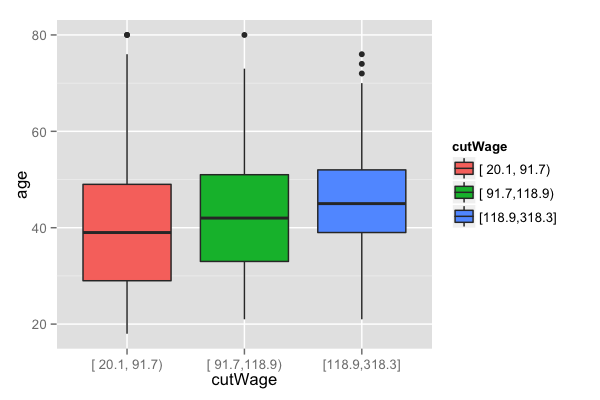


**cut2, making factors (*Hmisc* package)**

|  |
| --- |
| cutWage <- cut2(training$wage,g=3)  table(cutWage) |

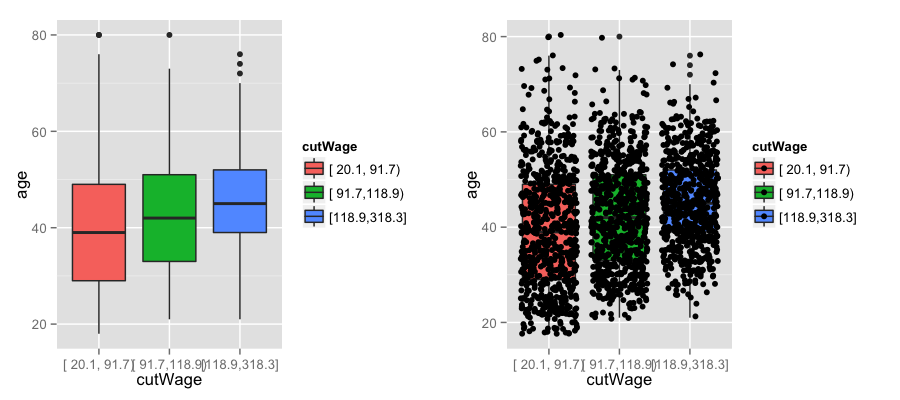
**Boxplots with cut2**

|  |
| --- |
| p1 <- qplot(cutWage,age, data=training,fill=cutWage,  geom=c("boxplot"))  p1 |



**Boxplots with points overlayed**

|  |
| --- |
| p2 <- qplot(cutWage,age, data=training,fill=cutWage,  geom=c("boxplot","jitter"))  grid.arrange(p1,p2,ncol=2) |

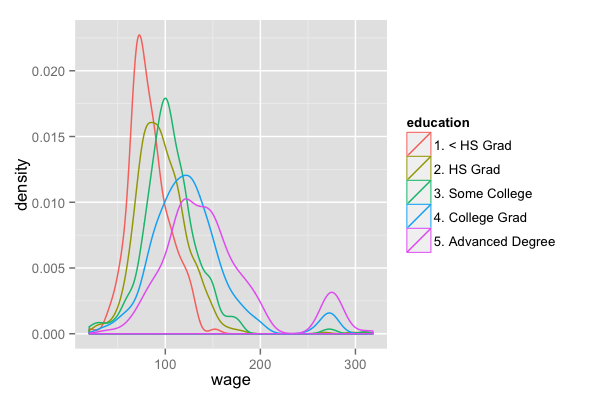


**Tables**

|  |
| --- |
| t1 <- table(cutWage,training$jobclass)  t1  prop.table(t1,1) |

**Density plots**

|  |
| --- |
| qplot(wage,colour=education,data=training,geom="density") |



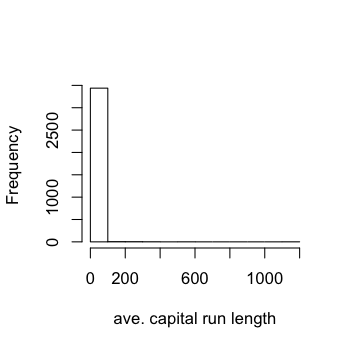
## Notes and further reading

* Make your plots only in the training set
  + Don't use the test set for exploration!
* Things you should be looking for
  + Imbalance in outcomes/predictors
  + Outliers
  + Groups of points not explained by a predictor
  + Skewed variables
* [ggplot2 tutorial](http://rstudio-pubs-static.s3.amazonaws.com/2176_75884214fc524dc0bc2a140573da38bb.html)
* [caret visualizations](http://caret.r-forge.r-project.org/visualizations.html)

# Preprocessing

**Why preprocess?**

|  |
| --- |
| library(caret); library(kernlab); data(spam)  inTrain <- createDataPartition(y=spam$type,  p=0.75, list=FALSE)  training <- spam[inTrain,]  testing <- spam[-inTrain,]  hist(training$capitalAve,main="",xlab="ave. capital run length") |



|  |
| --- |
| mean(training$capitalAve)  [1] 4.709  sd(training$capitalAve)  [1] 25.48 |

**Standardizing**

|  |
| --- |
| trainCapAve <- training$capitalAve  trainCapAveS <- (trainCapAve - mean(trainCapAve))/sd(trainCapAve)  mean(trainCapAveS)  [1] 5.862e-18  sd(trainCapAveS)  [1] 1 |

**Standardizing - test set**

|  |
| --- |
| testCapAve <- testing$capitalAve  testCapAveS <- (testCapAve - mean(trainCapAve))/sd(trainCapAve)  mean(testCapAveS)  [1] 0.07579  sd(testCapAveS)  [1] 1.79 |

**Standardizing - *preProcess* function**

|  |
| --- |
| preObj <- preProcess(training[,-58],method=c("center","scale"))  trainCapAveS <- predict(preObj,training[,-58])$capitalAve  mean(trainCapAveS)  [1] 5.862e-18  sd(trainCapAveS)  [1] 1 |

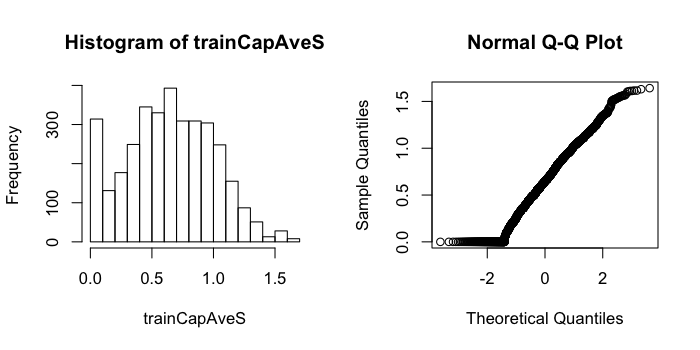
|  |
| --- |
| testCapAveS <- predict(preObj,testing[,-58])$capitalAve  mean(testCapAveS)  [1] 0.07579  sd(testCapAveS)  [1] 1.79 |

**Standardizing - *preProcess* argument**

|  |
| --- |
| set.seed(32343)  modelFit <- train(type ~.,data=training,  preProcess=c("center","scale"),method="glm")  modelFit |

**Standardizing - Box-Cox transforms**

|  |
| --- |
| preObj <- preProcess(training[,-58],method=c("BoxCox"))  trainCapAveS <- predict(preObj,training[,-58])$capitalAve  par(mfrow=c(1,2)); hist(trainCapAveS); qqnorm(trainCapAveS) |



**Standardizing - Imputing data**

|  |
| --- |
| set.seed(13343)  # Make some values NA  training$capAve <- training$capitalAve  selectNA <- rbinom(dim(training)[1],size=1,prob=0.05)==1  training$capAve[selectNA] <- NA  # Impute and standardize  preObj <- preProcess(training[,-58],method="knnImpute")  capAve <- predict(preObj,training[,-58])$capAve  # Standardize true values  capAveTruth <- training$capitalAve  capAveTruth <- (capAveTruth-mean(capAveTruth))/sd(capAveTruth) |

|  |
| --- |
| quantile(capAve - capAveTruth)  0% 25% 50% 75% 100%  -1.1324388 -0.0030842 -0.0015074 -0.0007467 0.2155789  quantile((capAve - capAveTruth)[selectNA])  0% 25% 50% 75% 100%  -0.9243043 -0.0125489 -0.0001968 0.0194524 0.2155789  quantile((capAve - capAveTruth)[!selectNA])  0% 25% 50% 75% 100%  -1.1324388 -0.0030033 -0.0015115 -0.0007938 -0.0001968 |

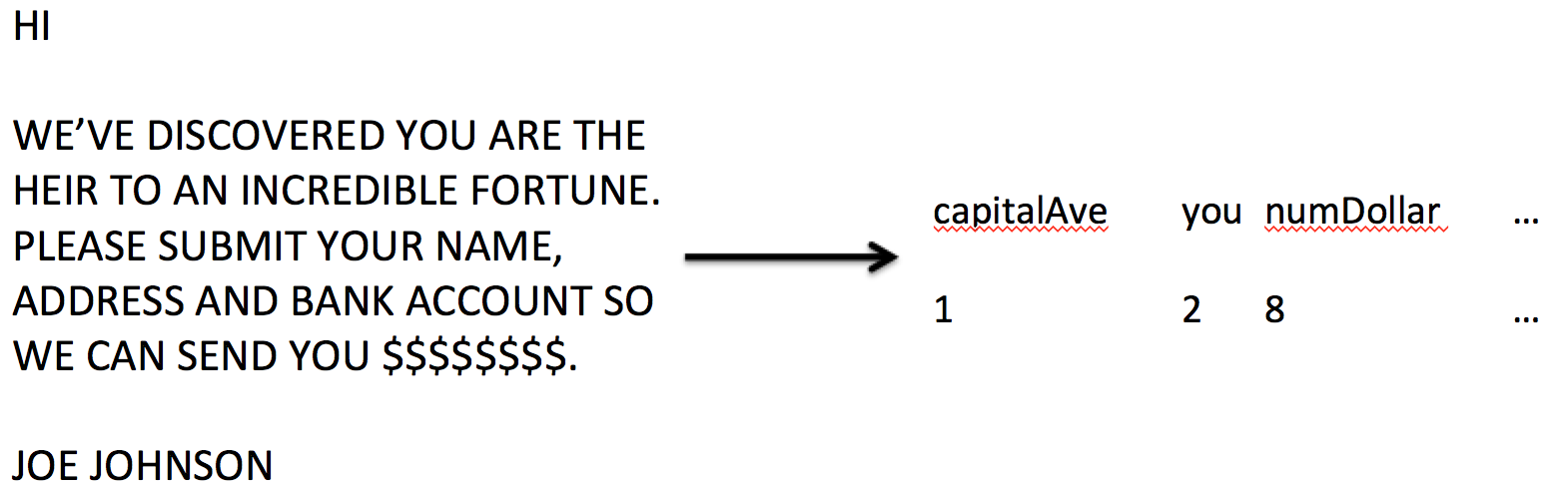
## Notes and further reading

* Training and test must be processed in the same way
* Test transformations will likely be imperfect
  + Especially if the test/training sets collected at different times
* Careful when transforming factor variables!
* [preprocessing with caret](http://caret.r-forge.r-project.org/preprocess.html)

# Covariate creation

## Two levels of covariate creation

**Level 1: From raw data to covariate**



**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](https://books.google.com/ngrams)), frequency of capital letters.
  + Images: Edges, corners, blobs, ridges ([computer vision feature detection](http://en.wikipedia.org/wiki/Feature_detection_%28computer_vision)))
  + Webpages: Number and type of images, position of elements, colors, videos ([A/B Testing](http://en.wikipedia.org/wiki/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**](http://bit.ly/19ZhWB6)

|  |
| --- |
| 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 |

**Removing zero covariates**

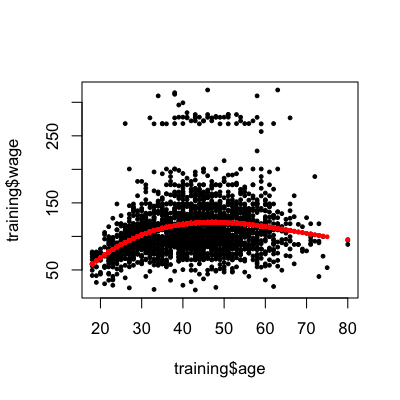
|  |
| --- |
| nsv <- nearZeroVar(training,saveMetrics=TRUE)  nsv  freqRatio percentUnique zeroVar nzv  year 1.029 0.33302 FALSE FALSE  age 1.122 2.80685 FALSE FALSE  sex 0.000 0.04757 TRUE TRUE  maritl 3.159 0.23787 FALSE FALSE  race 8.529 0.19029 FALSE FALSE  education 1.492 0.23787 FALSE FALSE  region 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  [19,] 0.26654 0.0339238 1.439e-03 |

**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 |

## 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](http://www.cs.nyu.edu/%7Eyann/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](http://caret.r-forge.r-project.org/preprocess.html)
* 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