The caret package

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

The caret R package

the caret package

The **caret** package (short for *C*lassification *A*nd *RE*gression *T*raining) is a set of functions that attempt to streamline the process for creating predictive models. The package contains tools for:

Links

`train` Model List

Topics

- Main Page
- Data Sets
- Visualizations
- Pre-Processing

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?

`obj` Class	Package	`predict` Function Syntax
`lda`	`MASS`	`predict(obj)` (no options needed)
`glm`	`stats`	`predict(obj, type = "response")`
`gbm`	`gbm`	`predict(obj, type = "response", n.trees)`
`mda`	`mda`	`predict(obj, type = "posterior")`
`rpart`	`rpart`	`predict(obj, type = "prob")`
`Weka`	`RWeka`	`predict(obj, type = "probability")`
`LogitBoost`	`caTools`	`predict(obj, type = "raw", nIter)`

http://www.edii.uclm.es/~useR-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)
```

```
[1] 3451   58
```

SPAM Example: Fit a model

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

```
Generalized Linear Model

3451 samples
  57 predictors
   2 classes: 'nonspam', 'spam'

No pre-processing
Resampling: Bootstrapped (25 reps)

Summary of sample sizes: 3451, 3451, 3451, 3451, 3451, 3451, ...

Resampling results

  Accuracy  Kappa  Accuracy SD  Kappa SD
  0.9       0.8    0.02         0.04
```

SPAM Example: Final model

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

```
Call:  NULL

Coefficients:
      (Intercept)              make           address               all             num3d
        -1.78e+00        -7.76e-01        -1.39e-01         3.68e-02         1.94e+00
              our             over            remove          internet            order
         7.61e-01         6.66e-01         2.34e+00         5.94e-01         4.10e-01
              mail          receive             will           people           report
         4.08e-02         2.71e-01        -1.08e-01        -2.28e-01        -1.14e-01
         addresses             free         business            email              you
         2.16e+00         8.78e-01         6.49e-01         1.38e-01         6.91e-02
            credit             your             font           num000            money
         8.00e-01         2.17e-01         2.17e-01         2.04e+00         1.95e+00
                hp              hpl           george           num650              lab
        -1.82e+00        -9.17e-01        -7.50e+00         3.33e-01        -1.89e+00
              labs           telnet           num857             data           num415
```

SPAM Example: Prediction

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

```
  [1] spam    spam    spam    nonspam nonspam nonspam spam    spam    spam    spam    spam
 [12] spam    spam    spam    spam    spam    spam    spam    nonspam spam    spam    spam
 [23] nonspam spam    nonspam nonspam spam    spam    spam    spam    spam    spam    spam
 [34] spam    spam    spam    spam    spam    spam    spam    spam    spam    spam    spam
 [45] spam    spam    spam    spam    nonspam spam    nonspam spam    spam    spam    spam
 [56] spam    nonspam nonspam spam    spam    spam    spam    spam    nonspam spam    spam
 [67] spam    spam    spam    spam    spam    spam    spam    spam    spam    spam    spam
 [78] nonspam nonspam nonspam spam    spam    nonspam spam    nonspam nonspam spam    spam
 [89] spam    spam    spam    spam    spam    spam    nonspam spam    spam    spam    spam
[100] spam    spam    spam    nonspam spam    nonspam spam    spam    spam    spam    spam
[111] spam    spam    spam    spam    nonspam spam    spam    spam    spam    spam    spam
[122] spam    spam    spam    spam    spam    spam    spam    nonspam spam    spam    nonspam
[133] spam    spam    spam    spam    spam    spam    spam    spam    spam    spam    spam
[144] spam    spam    spam    nonspam spam    spam    spam    spam    spam    spam    spam
[155] nonspam spam    nonspam spam    nonspam spam    spam    spam    spam    spam    spam
[166] spam    spam    spam    spam    spam    spam    spam    spam    spam    spam    spam
[177] spam    spam    spam    spam    spam    spam    spam    spam    spam    spam    spam
```

SPAM Example: Confusion Matrix

```
confusionMatrix(predictions,testing$type)
```

```
Confusion Matrix and Statistics

          Reference
Prediction nonspam spam
   nonspam     665   54
   spam         32  399

               Accuracy : 0.925
                 95% CI : (0.908, 0.94)
    No Information Rate : 0.606
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.842
 Mcnemar's Test P-Value : 0.0235

            Sensitivity : 0.954
            Specificity : 0.881
         Pos Pred Value : 0.925
```

Further information

- Caret tutorials:
  - http://www.edii.uclm.es/~useR-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

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

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)
```

```
[1] 3451   58
```

SPAM Example: K-fold

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

```
Fold01 Fold02 Fold03 Fold04 Fold05 Fold06 Fold07 Fold08 Fold09 Fold10
  4141   4140   4141   4142   4140   4142   4141   4141   4140   4141
```

```
folds[[1]][1:10]
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

SPAM Example: Return test

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

```
Fold01 Fold02 Fold03 Fold04 Fold05 Fold06 Fold07 Fold08 Fold09 Fold10
   460    461    460    459    461    459    460    460    461    460
```

```
folds[[1]][1:10]
```

```
 [1] 24 27 32 40 41 43 55 58 63 68
```

SPAM Example: Resampling

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

```
Resample01 Resample02 Resample03 Resample04 Resample05 Resample06 Resample07 Resample08 Resample09
      4601       4601       4601       4601       4601       4601       4601       4601       4601
Resample10
      4601
```

```
folds[[1]][1:10]
```

```
 [1]  1  2  3  3  3  5  5  7  8 12
```

SPAM Example: Time Slices

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

```
[1] "train" "test"
```

```
folds$train[[1]]
```

```
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
folds$test[[1]]
```

```
 [1] 21 22 23 24 25 26 27 28 29 30
```

Further information

- Caret tutorials:
  - http://www.edii.uclm.es/~useR-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

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

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

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*

Setting the seed

- It is often useful to set an overall seed
- You can also set a seed for each resample
- Seeding each resample is useful for parallel fits

seed example

```
set.seed(1235)
modelFit2 <- train(type ~.,data=training, method="glm")
modelFit2
```

```
Generalized Linear Model

3451 samples
  57 predictors
   2 classes: 'nonspam', 'spam'

No pre-processing
Resampling: Bootstrapped (25 reps)

Summary of sample sizes: 3451, 3451, 3451, 3451, 3451, 3451, ...

Resampling results

  Accuracy  Kappa  Accuracy SD  Kappa SD
  0.9       0.8    0.007        0.01
```

seed example

```
set.seed(1235)
modelFit3 <- train(type ~.,data=training, method="glm")
modelFit3
```

```
Generalized Linear Model

3451 samples
  57 predictors
   2 classes: 'nonspam', 'spam'

No pre-processing
Resampling: Bootstrapped (25 reps)

Summary of sample sizes: 3451, 3451, 3451, 3451, 3451, 3451, ...

Resampling results

  Accuracy  Kappa  Accuracy SD  Kappa SD
  0.9       0.8    0.007        0.01
```

Further resources

- Caret tutorial
- Model training and tuning

Plotting predictors

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

Example: predicting wages

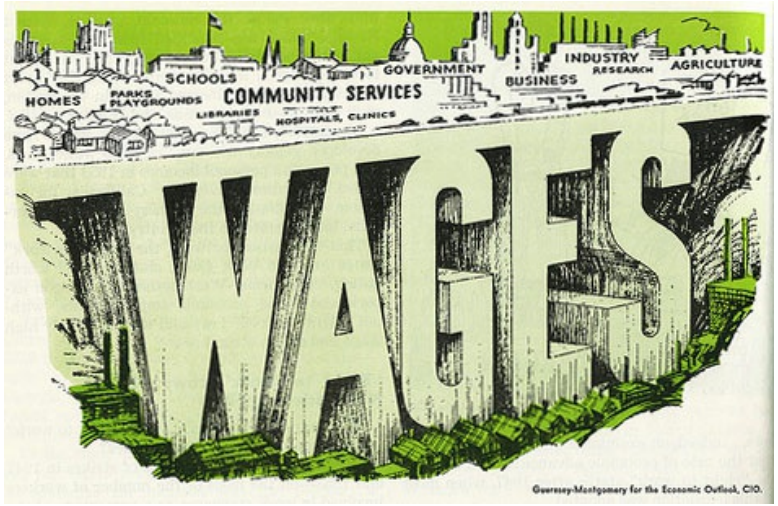


Image Credit http://www.cahs-media.org/the-high-cost-of-low-wages

Data from: ISLR package from the book: Introduction to statistical learning

Example: Wage data

```
library(ISLR); library(ggplot2); library(caret);
data(Wage)
summary(Wage)
```

```
      year            age               sex                    maritl           race
 Min.   :2003   Min.   :18.0   1. Male  :3000   1. Never Married: 648   1. White:2480
 1st Qu.:2004   1st Qu.:33.8   2. Female:   0   2. Married      :2074   2. Black: 293
 Median :2006   Median :42.0                    3. Widowed      :  19   3. Asian: 190
 Mean   :2006   Mean   :42.4                    4. Divorced     : 204   4. Other:  37
 3rd Qu.:2008   3rd Qu.:51.0                    5. Separated    :  55
 Max.   :2009   Max.   :80.0

             education                     region                jobclass               health
 1. < HS Grad      :268   2. Middle Atlantic   :3000   1. Industrial :1544   1. <=Good      : 858
 2. HS Grad        :971   1. New England       :   0   2. Information:1456   2. >=Very Good:2142
 3. Some College   :650   3. East North Central:   0
 4. College Grad   :685   4. West North Central:   0
 5. Advanced Degree:426   5. South Atlantic    :   0
                          6. East South Central:   0
                          (Other)              :   0
```

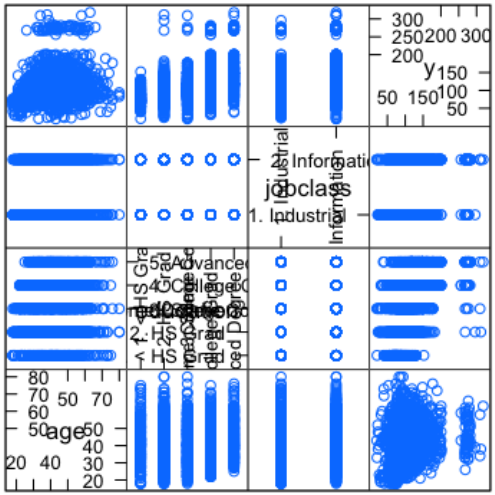
Get training/test sets

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

```
[1] 898  12
```

Feature plot (*caret* package)

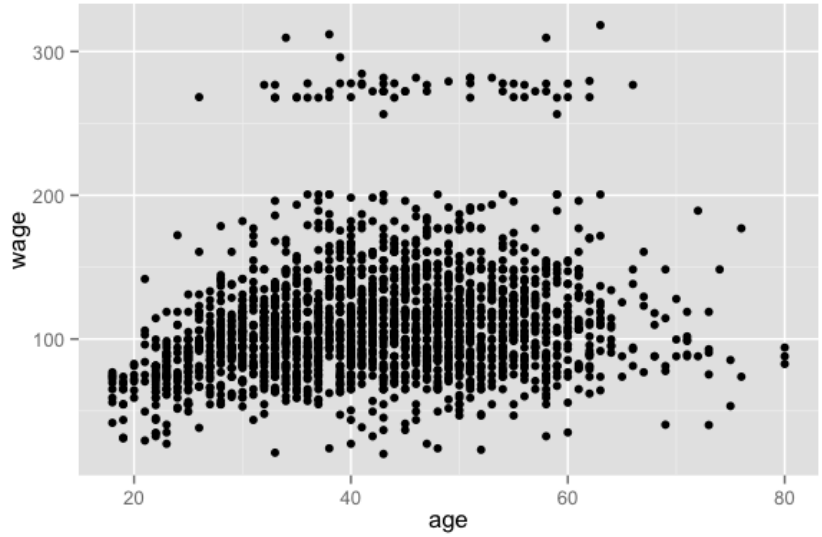
```
featurePlot(x=training[,c("age","education","jobclass")],
            y = training$wage,
            plot="pairs")
```



Scatter Plot Matrix

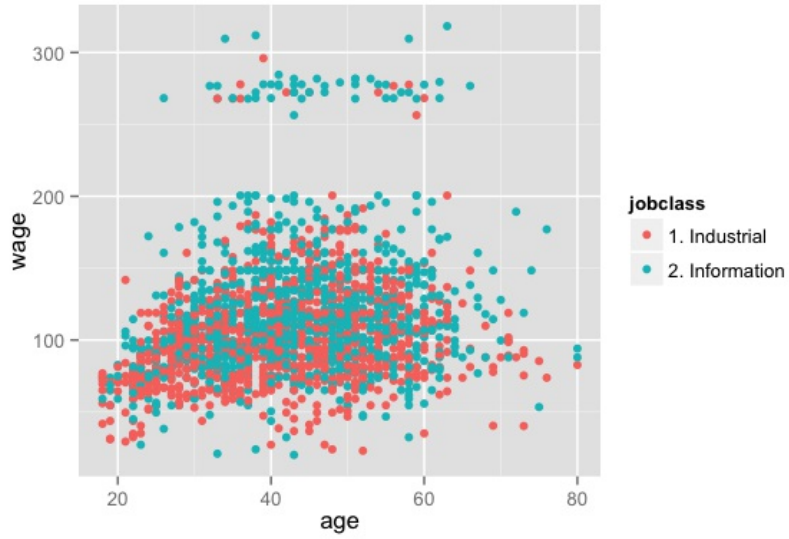
Qplot (*ggplot2* package)

```
qplot(age,wage,data=training)
```



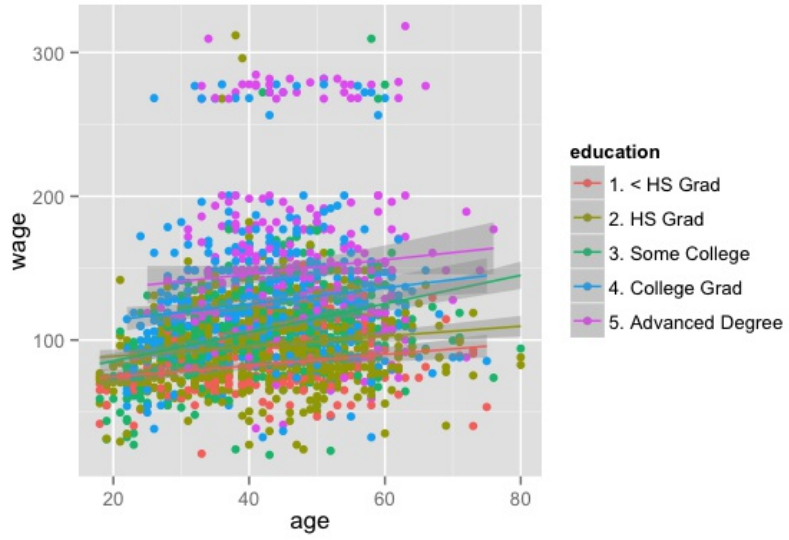
Qplot with color (*ggplot2* package)

```
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)
```



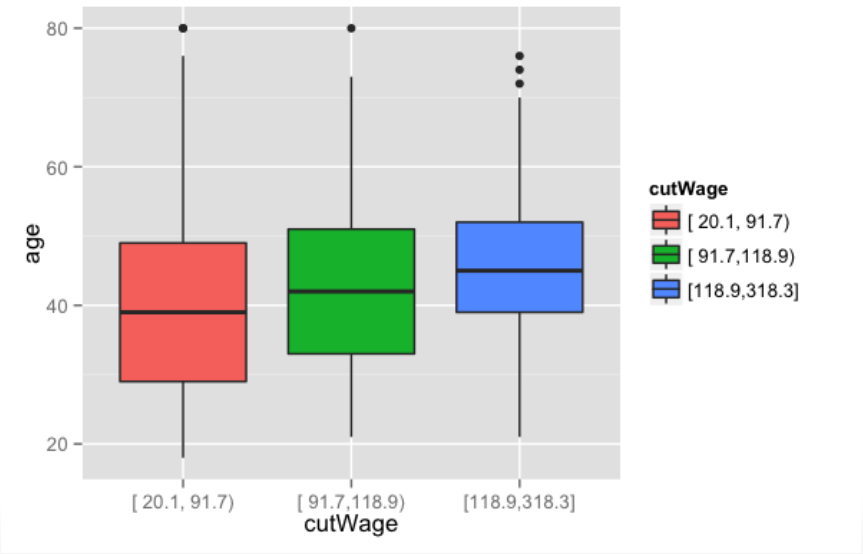
cut2, making factors (*Hmisc* package)

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

```
cutWage
[ 20.1, 91.7) [ 91.7,118.9) [118.9,318.3]
          704           725           673
```

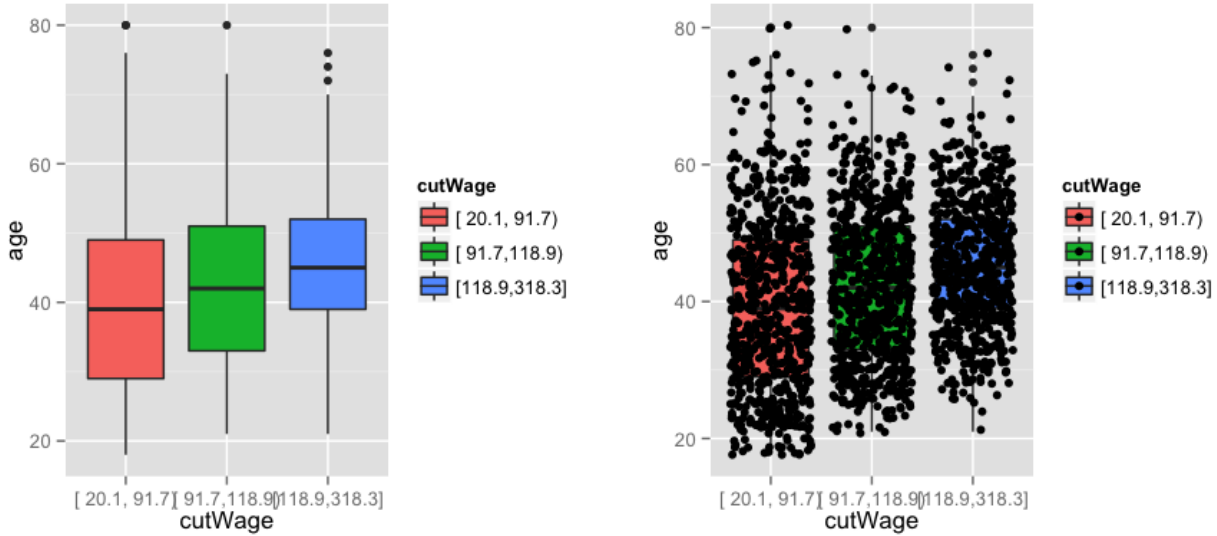
Boxplots with cut2

```
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
```

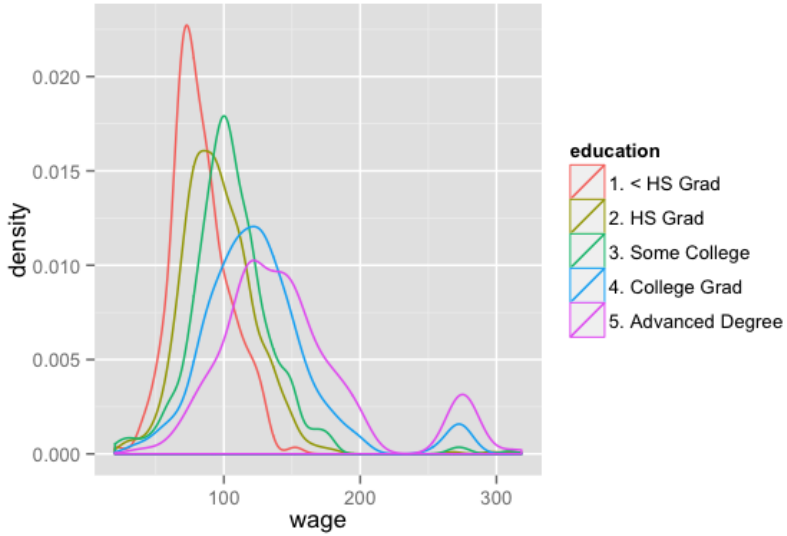
```
cutWage         1. Industrial 2. Information
  [ 20.1, 91.7)           437            267
  [ 91.7,118.9)           365            360
  [118.9,318.3]           263            410
```

```
prop.table(t1,1)
```

```
cutWage         1. Industrial 2. Information
  [ 20.1, 91.7)        0.6207         0.3793
  [ 91.7,118.9)        0.5034         0.4966
  [118.9,318.3]        0.3908         0.6092
```

Density plots

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



Notes and further reading

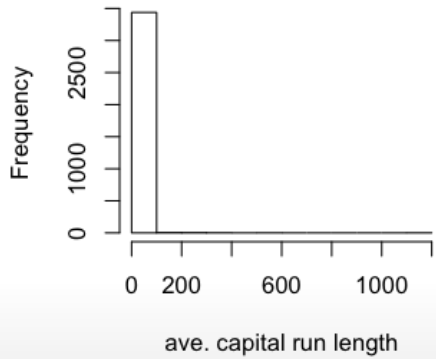
- Make your plots only in the training set
  - Don't use the test set for exploration!
- Things you should be looking for
  - Imbalance in outcomes/predictors
  - Outliers
  - Groups of points not explained by a predictor
  - Skewed variables
- ggplot2 tutorial
- caret visualizations

Preprocessing

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

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")
```



Why preprocess?

```
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
```

Standardizing - *preProcess* function

```
testCapAveS <- predict(preObj,testing[,-58])$capitalAve
mean(testCapAveS)
```

```
[1] 0.07579
```

```
sd(testCapAveS)
```

```
[1] 1.79
```

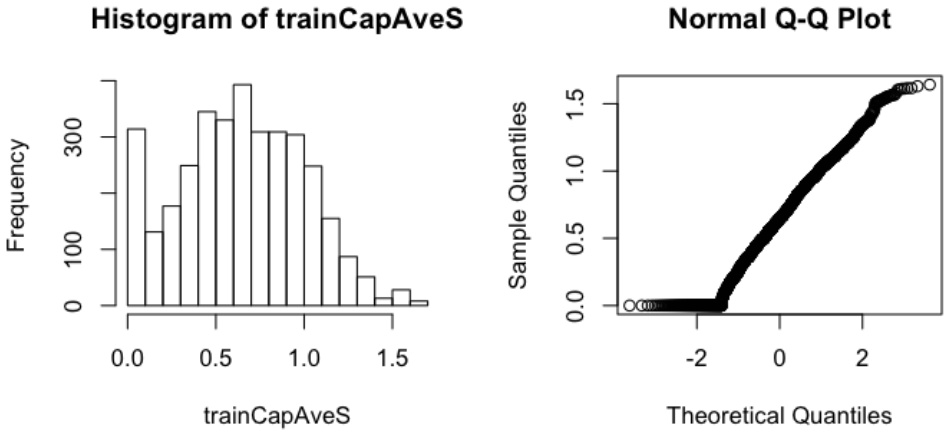
Standardizing - *preProcess* argument

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

```
3451 samples
  57 predictors
   2 classes: 'nonspam', 'spam'

Pre-processing: centered, scaled
Resampling: Bootstrap (25 reps)

Summary of sample sizes: 3451, 3451, 3451, 3451, 3451, 3451, ...

Resampling results

  Accuracy  Kappa  Accuracy SD  Kappa SD
  0.9       0.8    0.007        0.01
```

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)
```

Standardizing - Imputing data

```
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

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
```

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
```

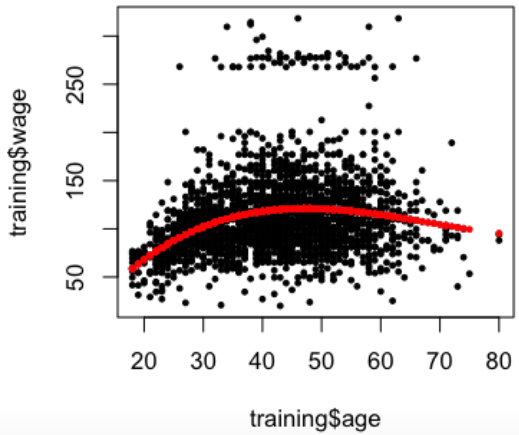
Spline basis

```
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
```

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
```

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

Preprocessing with Principal Components Analysis (PCA)

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

Correlated predictors

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

M <- abs(cor(training[,-58]))
diag(M) <- 0
which(M > 0.8,arr.ind=T)
```

```
       row col
num415  34  32
num857  32  34
```

Correlated predictors

```
names(spam)[c(34,32)]
```

```
[1] "num415" "num857"
```

```
plot(spam[,34],spam[,32])
```

Basic PCA idea

- We might not need every predictor
- A weighted combination of predictors might be better
- We should pick this combination to capture the "most information" possible
- Benefits
  - Reduced number of predictors
  - Reduced noise (due to averaging)

We could rotate the plot

$$X = 0.71 \times num415 + 0.71 \times num857$$

$$Y = 0.71 \times num415 - 0.71 \times num857$$

```
X <- 0.71*training$num415 + 0.71*training$num857
Y <- 0.71*training$num415 - 0.71*training$num857
plot(X,Y)
```

Related problems

You have multivariate variables $X_1, \dots, X_n$ so $X_1 = (X_{11}, \dots, X_{1m})$

- Find a new set of multivariate variables that are uncorrelated and explain as much variance as possible.
- If you put all the variables together in one matrix, find the best matrix created with fewer variables (lower rank) that explains the original data.

The first goal is statistical and the second goal is data compression.

Related solutions - PCA/SVD

SVD

If X is a matrix with each variable in a column and each observation in a row then the SVD is a "matrix decomposition"

$$X = UDV^T$$

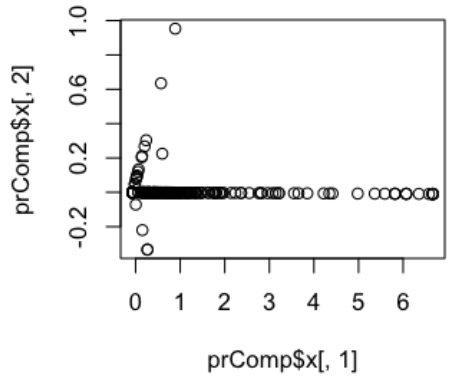
where the columns of U are orthogonal (left singular vectors), the columns of V are orthogonal (right singluar vectors) and D is a diagonal matrix (singular values).

PCA

The principal components are equal to the right singular values if you first scale (subtract the mean, divide by the standard deviation) the variables.

Principal components in R - prcomp

```
smallSpam <- spam[,c(34,32)]
prComp <- prcomp(smallSpam)
plot(prComp$x[,1],prComp$x[,2])
```



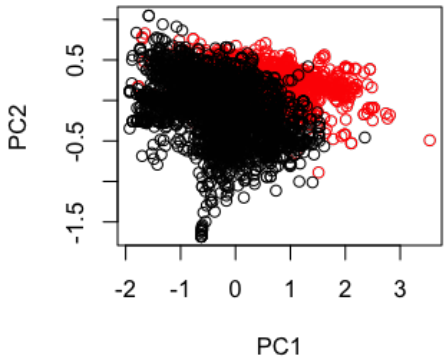
Principal components in R - prcomp

```
prComp$rotation
```

```
          PC1     PC2
num415 0.7081  0.7061
num857 0.7061 -0.7081
```

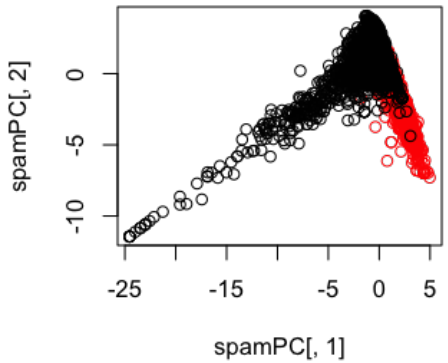
PCA on SPAM data

```
typeColor <- ((spam$type=="spam")*1 + 1)
prComp <- prcomp(log10(spam[,-58]+1))
plot(prComp$x[,1],prComp$x[,2],col=typeColor,xlab="PC1",ylab="PC2")
```



PCA with caret

```
preProc <- preProcess(log10(spam[,-58]+1),method="pca",pcaComp=2)
spamPC <- predict(preProc,log10(spam[,-58]+1))
plot(spamPC[,1],spamPC[,2],col=typeColor)
```



Preprocessing with PCA

```
preProc <- preProcess(log10(training[,-58]+1),method="pca",pcaComp=2)
trainPC <- predict(preProc,log10(training[,-58]+1))
modelFit <- train(training$type ~ .,method="glm",data=trainPC)
```

Preprocessing with PCA

```
testPC <- predict(preProc,log10(testing[,-58]+1))
confusionMatrix(testing$type,predict(modelFit,testPC))
```

```
Confusion Matrix and Statistics

          Reference
Prediction nonspam spam
   nonspam     646   51
   spam         64  389

               Accuracy : 0.9
                 95% CI : (0.881, 0.917)
    No Information Rate : 0.617
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.79
 Mcnemar's Test P-Value : 0.263

            Sensitivity : 0.910
            Specificity : 0.884
```

Alternative (sets # of PCs)

```
modelFit <- train(training$type ~ .,method="glm",preProcess="pca",data=training)
confusionMatrix(testing$type,predict(modelFit,testing))
```

```
Confusion Matrix and Statistics

          Reference
Prediction nonspam spam
   nonspam     660   37
   spam         54  399

               Accuracy : 0.921
                 95% CI : (0.904, 0.936)
    No Information Rate : 0.621
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.833
 Mcnemar's Test P-Value : 0.0935

            Sensitivity : 0.924
            Specificity : 0.915
```

Final thoughts on PCs

- Most useful for linear-type models
- Can make it harder to interpret predictors
- Watch out for outliers!
  - Transform first (with logs/Box Cox)
  - Plot predictors to identify problems
- For more info see
  - Exploratory Data Analysis
  - Elements of Statistical Learning

Predicting with regression

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

Key ideas

- Fit a simple regression model
- Plug in new covariates and multiply by the coefficients
- Useful when the linear model is (nearly) correct

Pros:

- Easy to implement
- Easy to interpret

Cons:

- Often poor performance in nonlinear settings

Example: Old faithful eruptions

Image Credit/Copyright Wally Pacholka http://www.astropics.com/

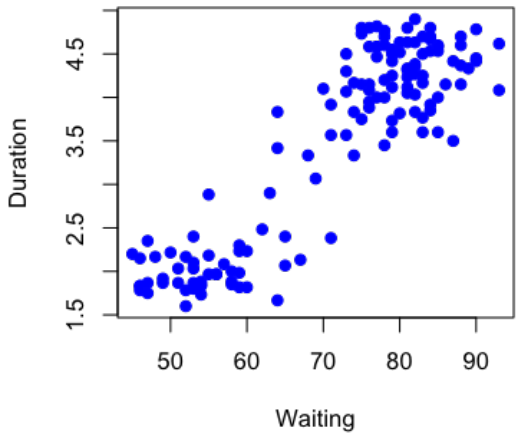
Example: Old faithful eruptions

```
library(caret);data(faithful); set.seed(333)
inTrain <- createDataPartition(y=faithful$waiting,
                               p=0.5, list=FALSE)
trainFaith <- faithful[inTrain,]; testFaith <- faithful[-inTrain,]
head(trainFaith)
```

```
   eruptions waiting
6      2.883      55
11     1.833      54
16     2.167      52
19     1.600      52
22     1.750      47
27     1.967      55
```

Eruption duration versus waiting time

```
plot(trainFaith$waiting,trainFaith$eruptions,pch=19,col="blue",xlab="Waiting",ylab="Duration")
```



Fit a linear model

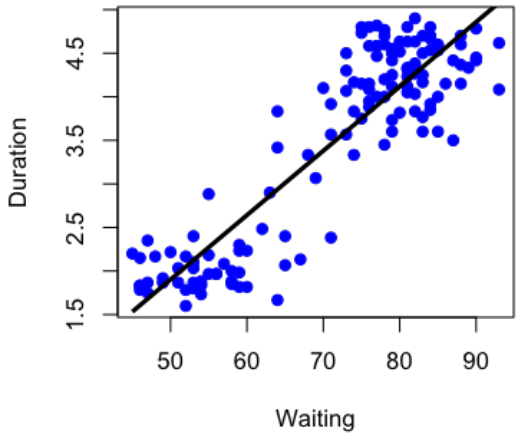
$$ED_i = b_0 + b_1 WT_i + e_i$$

```
lm1 <- lm(eruptions ~ waiting,data=trainFaith)
summary(lm1)
```

```
Call:
lm(formula = eruptions ~ waiting, data = trainFaith)

Residuals:
    Min      1Q  Median      3Q     Max
-1.2699 -0.3479  0.0398  0.3659  1.0502

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.79274    0.22787   -7.87    1e-12 ***
waiting      0.07390    0.00315   23.47   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Model fit

```
plot(trainFaith$waiting,trainFaith$eruptions,pch=19,col="blue",xlab="Waiting",ylab="Duration")
lines(trainFaith$waiting,lm1$fitted,lwd=3)
```



Predict a new value

$$\hat{ED} = \hat{b}_0 + \hat{b}_1 WT$$

```
coef(lm1)[1] + coef(lm1)[2]*80
```

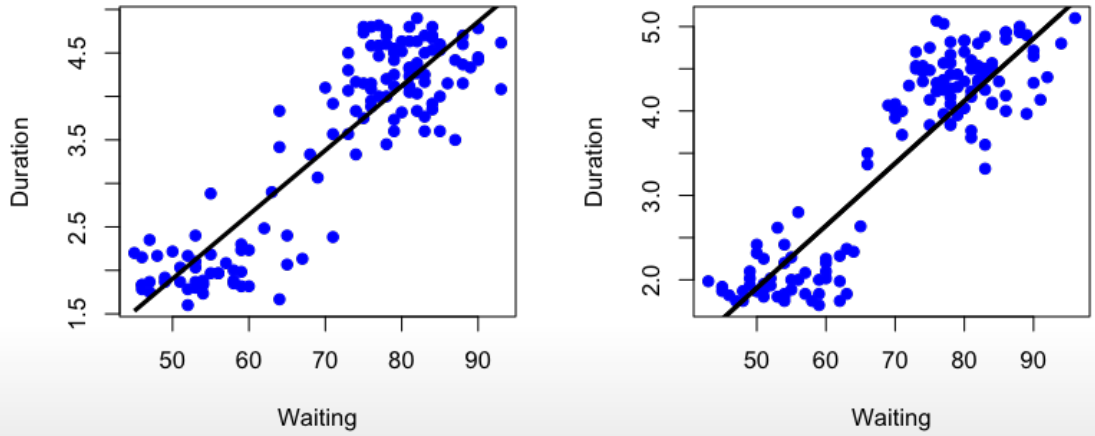
```
(Intercept)
      4.119
```

```
newdata <- data.frame(waiting=80)
predict(lm1,newdata)
```

```
    1
4.119
```

Plot predictions - training and test

```
par(mfrow=c(1,2))
plot(trainFaith$waiting,trainFaith$eruptions,pch=19,col="blue",xlab="Waiting",ylab="Duration")
lines(trainFaith$waiting,predict(lm1),lwd=3)
plot(testFaith$waiting,testFaith$eruptions,pch=19,col="blue",xlab="Waiting",ylab="Duration")
lines(testFaith$waiting,predict(lm1,newdata=testFaith),lwd=3)
```



Get training set/test set errors

```
# Calculate RMSE on training
sqrt(sum((lm1$fitted-trainFaith$eruptions)^2))
```

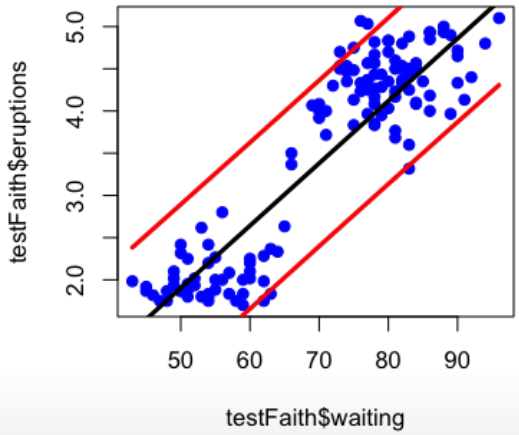
```
[1] 5.752
```

```
# Calculate RMSE on test
sqrt(sum((predict(lm1,newdata=testFaith)-testFaith$eruptions)^2))
```

```
[1] 5.839
```

Prediction intervals

```
pred1 <- predict(lm1,newdata=testFaith,interval="prediction")
ord <- order(testFaith$waiting)
plot(testFaith$waiting,testFaith$eruptions,pch=19,col="blue")
matlines(testFaith$waiting[ord],pred1[ord,],type="l",,col=c(1,2,2),lty = c(1,1,1), lwd=3)
```



Same process with caret

```
modFit <- train(eruptions ~ waiting,data=trainFaith,method="lm")
summary(modFit$finalModel)
```

```
Call:
lm(formula = modFormula, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-1.2699 -0.3479  0.0398  0.3659  1.0502

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.79274    0.22787   -7.87    1e-12 ***
waiting      0.07390    0.00315   23.47   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.495 on 135 degrees of freedom
Multiple R-squared:  0.803, Adjusted R-squared:  0.802
```

Notes and further reading

- Regression models with multiple covariates can be included
- Often useful in combination with other models
- Elements of statistical learning
- Modern applied statistics with S
- Introduction to statistical learning

Predicting with regression, multiple covariates

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

Example: predicting wages

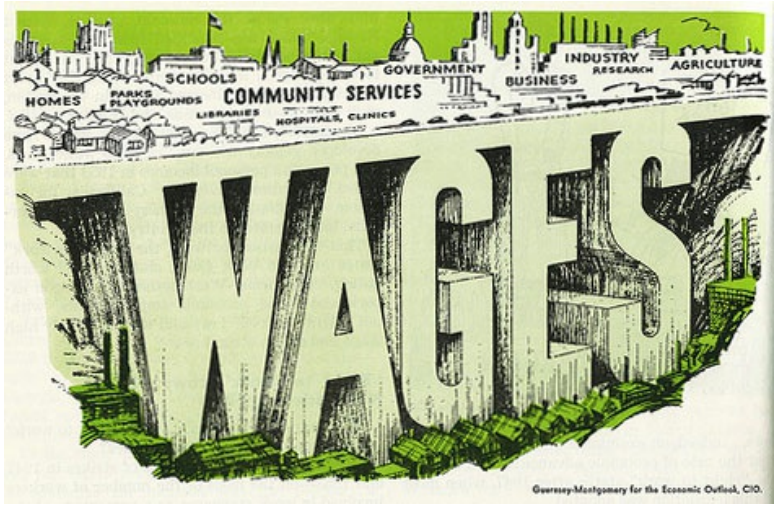


Image Credit http://www.cahs-media.org/the-high-cost-of-low-wages

Data from: ISLR package from the book: Introduction to statistical learning

Example: Wage data

```
library(ISLR); library(ggplot2); library(caret);
data(Wage); Wage <- subset(Wage,select=-c(logwage))
summary(Wage)
```

```
      year           age               sex                    maritl           race
 Min.   :2003   Min.   :18.0   1. Male  :3000   1. Never Married: 648   1. White:2480
 1st Qu.:2004   1st Qu.:33.8   2. Female:   0   2. Married      :2074   2. Black: 293
 Median :2006   Median :42.0                    3. Widowed      :  19   3. Asian: 190
 Mean   :2006   Mean   :42.4                    4. Divorced     : 204   4. Other:  37
 3rd Qu.:2008   3rd Qu.:51.0                    5. Separated    :  55
 Max.   :2009   Max.   :80.0

             education                     region                jobclass               health
 1. < HS Grad      :268   2. Middle Atlantic   :3000   1. Industrial :1544   1. <=Good      : 858
 2. HS Grad        :971   1. New England       :   0   2. Information:1456   2. >=Very Good:2142
 3. Some College   :650   3. East North Central:   0
 4. College Grad   :685   4. West North Central:   0
 5. Advanced Degree:426   5. South Atlantic    :   0
                          6. East South Central:   0
                          (Other)              :   0
```

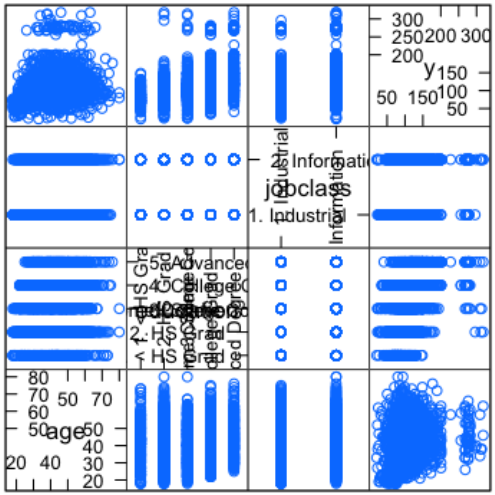
Get training/test sets

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

```
[1] 898  12
```

Feature plot

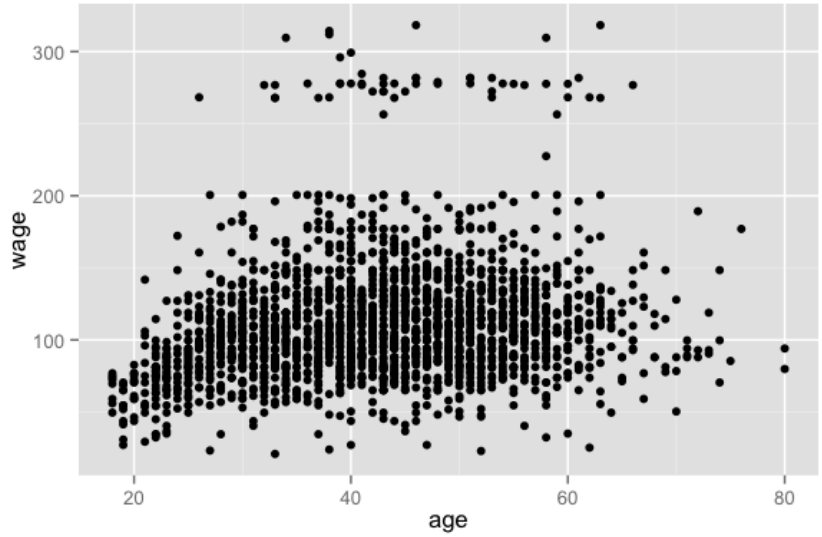
```
featurePlot(x=training[,c("age","education","jobclass")],
            y = training$wage,
            plot="pairs")
```



Scatter Plot Matrix

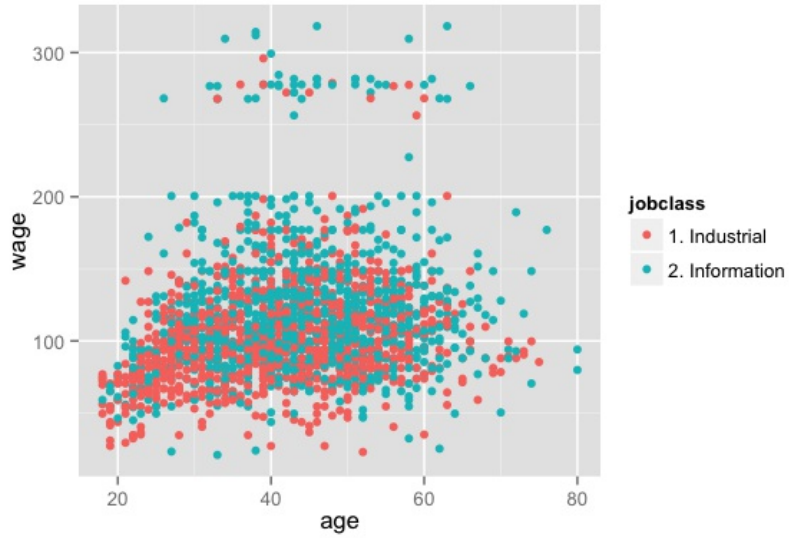
Plot age versus wage

```
qplot(age,wage,data=training)
```



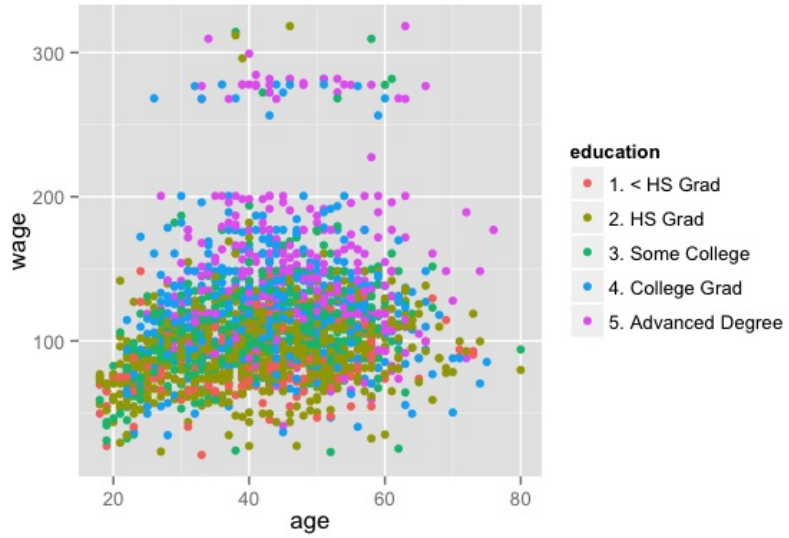
Plot age versus wage colour by jobclass

```
qplot(age,wage,colour=jobclass,data=training)
```



Plot age versus wage colour by education

```
qplot(age,wage,colour=education,data=training)
```



Fit a linear model

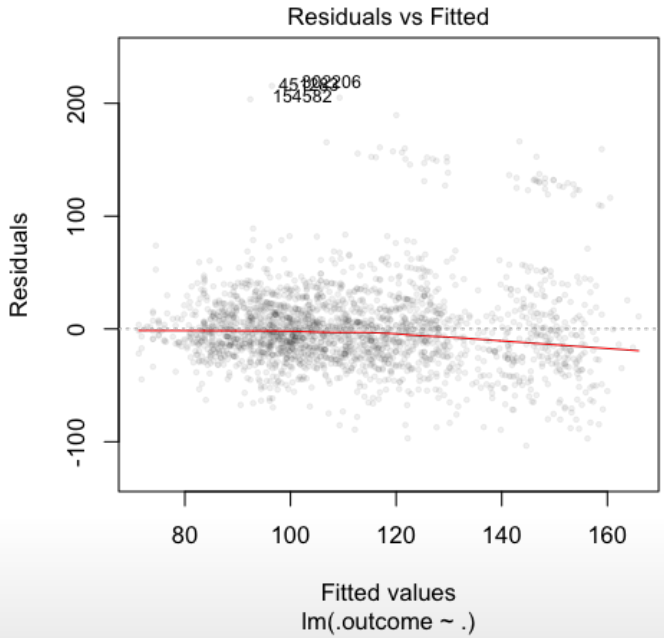
$$ED_i = b_0 + b_1 age + b_2 I(Jobclass_i = "Information") + \sum_{k=1}^{4} \gamma_k I(education_i = levelk)$$

```
modFit<- train(wage ~ age + jobclass + education,
               method = "lm",data=training)
finMod <- modFit$finalModel
print(modFit)
```

```
Linear Regression

2102 samples
  11 predictors

No pre-processing
Resampling: Bootstrapped (25 reps)

Summary of sample sizes: 2102, 2102, 2102, 2102, 2102, 2102, ...

Resampling results
```

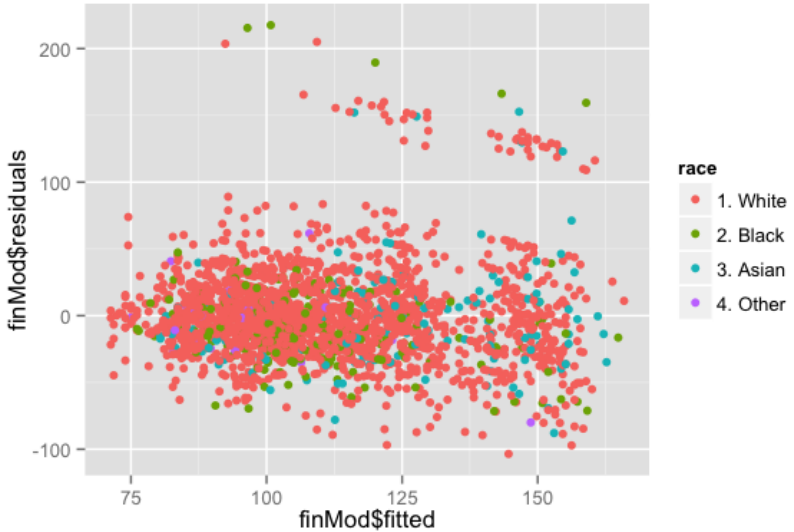
Diagnostics

```
plot(finMod,1,pch=19,cex=0.5,col="#00000010")
```



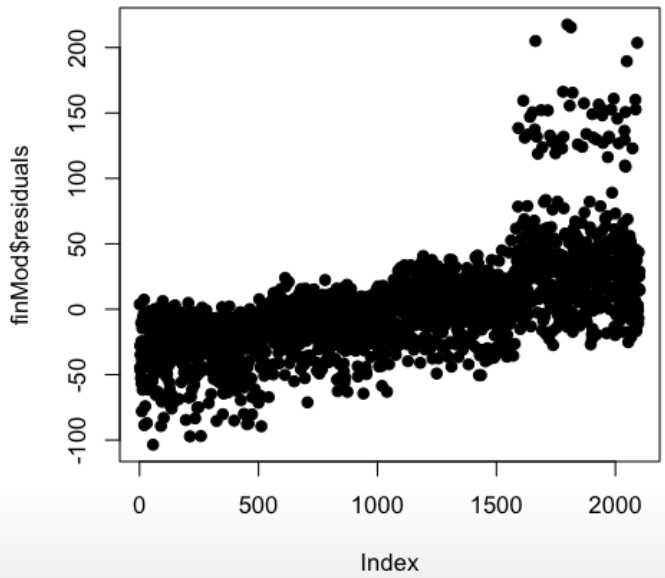
Color by variables not used in the model

```
qplot(finMod$fitted,finMod$residuals,colour=race,data=training)
```



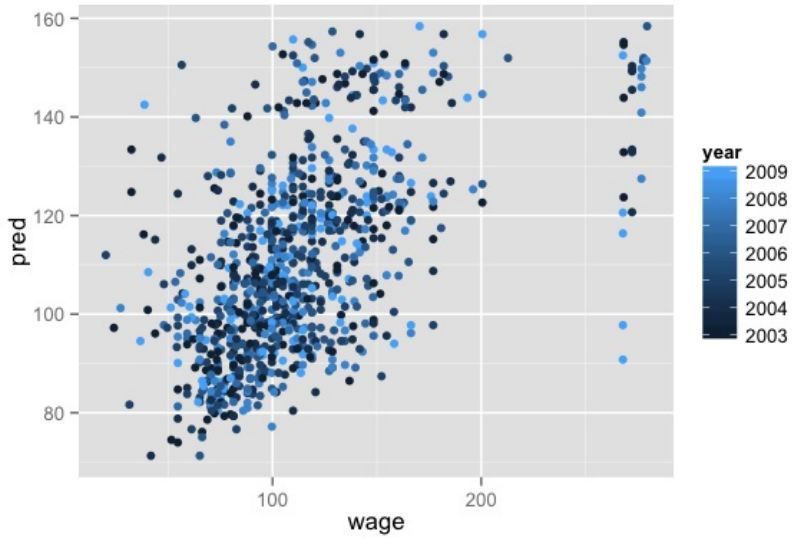
Plot by index

```
plot(finMod$residuals,pch=19)
```



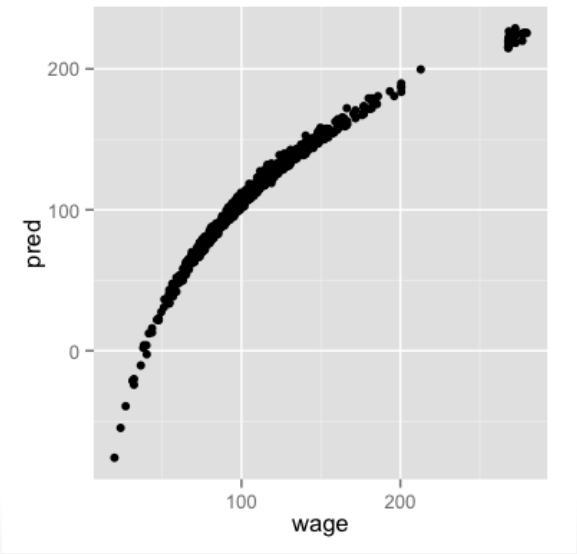
Predicted versus truth in test set

```
pred <- predict(modFit, testing)
qplot(wage,pred,colour=year,data=testing)
```



If you want to use all covariates

```
modFitAll<- train(wage ~ .,data=training,method="lm")
pred <- predict(modFitAll, testing)
qplot(wage,pred,data=testing)
```



Notes and further reading

- Often useful in combination with other models
- Elements of statistical learning
- Modern applied statistics with S
- Introduction to statistical learning