

Big data analytics with RRE: Modeling





Sample Data

```
sampleDataDir <- rxGetOption("sampleDataDir")
list.files(path = sampleDataDir, pattern = "*.csv")

## [1] "AirlineDemo1kNoMissing.csv" "AirlineDemoSmall.csv"
## [3] "mortDefaultSmall2000.csv"  "mortDefaultSmall2001.csv"
## [5] "mortDefaultSmall2002.csv"  "mortDefaultSmall2003.csv"
## [7] "mortDefaultSmall2004.csv"  "mortDefaultSmall2005.csv"
## [9] "mortDefaultSmall2006.csv"  "mortDefaultSmall2007.csv"
## [11] "mortDefaultSmall2008.csv"  "mortDefaultSmall2009.csv"
```



Import Setup

```
# identify input file
inputFile <- file.path(sampleDataDir, "AirlineDemoSmall.csv")
# identify output file
airDS <- "xdf/ADS.xdf"
# Make a factor variable while reading in the file
colInfo <- list(DayOfWeek = list(type = "factor", levels = c("Monday", "Tuesday", "Wednesday",
  "Thursday", "Friday", "Saturday", "Sunday")))
```



Import the Data

```
# Import the file  
system.time(rxImport(inData = inputFile, outFile = airDS, colInfo = colInfo, missingValueString = "M",  
  overwrite = TRUE))
```

```
##      user  system elapsed  
##      1.63    0.08    1.78
```



Examine the data

```
rxGetInfo(airDS, getVarInfo = TRUE)
```

```
## File name: E:\GitHub\Revolution_Course_Materials\modules\BigData\DataCamp_Analysis\doc\xdf\ADS.xdf
## Number of observations: 600000
## Number of variables: 3
## Number of blocks: 2
## Compression type: zlib
## Variable information:
## Var 1: ArrDelay, Type: integer, Low/High: (-86, 1490)
## Var 2: CRSDepTime, Type: numeric, Storage: float32, Low/High: (0.0167, 23.9833)
## Var 3: DayOfWeek
##          7 factor levels: Monday Tuesday Wednesday Thursday Friday Saturday Sunday
```



Summarize Each Variable

```
print(rxSummary(~., data = airDS), header = FALSE)
```

```
## Name      Mean      StdDev    Min      Max      ValidObs MissingObs
## ArrDelay  11.31794 40.688536 -86.000000 1490.00000 582628    17372
## CRSDepTime 13.48227 4.697566  0.016667  23.98333 600000     0
##
## Category Counts for DayOfWeek
## Number of categories: 7
## Number of valid observations: 600000
## Number of missing observations: 0
##
## DayOfWeek Counts
## Monday      97975
## Tuesday     77725
## Wednesday   78875
## Thursday    81304
## Friday      82987
## Saturday    86159
## Sunday      94975
```



Summarize by Levels

```
print(delaySumm <- rxSummary(ArrDelay ~ DayOfWeek, data = airDS), header = FALSE)

##      Name              Mean      StdDev   Min Max  ValidObs MissingObs
## ArrDelay:DayOfWeek 11.31794 40.68854 -86 1490 582628    17372
##
## Statistics by category (7 categories):
##
##      Category              DayOfWeek Means      StdDev   Min Max  ValidObs
## ArrDelay for DayOfWeek=Monday    Monday    12.025604 40.02463 -76 1017 95298
## ArrDelay for DayOfWeek=Tuesday   Tuesday    11.293808 43.66269 -70 1143 74011
## ArrDelay for DayOfWeek=Wednesday Wednesday 10.156539 39.58803 -81 1166 76786
## ArrDelay for DayOfWeek=Thursday  Thursday     8.658007 36.74724 -58 1053 79145
## ArrDelay for DayOfWeek=Friday    Friday    14.804335 41.79260 -78 1490 80142
## ArrDelay for DayOfWeek=Saturday  Saturday   11.875326 45.24540 -73 1370 83851
## ArrDelay for DayOfWeek=Sunday    Sunday    10.331806 37.33348 -86 1202 93395
```



Compute Means

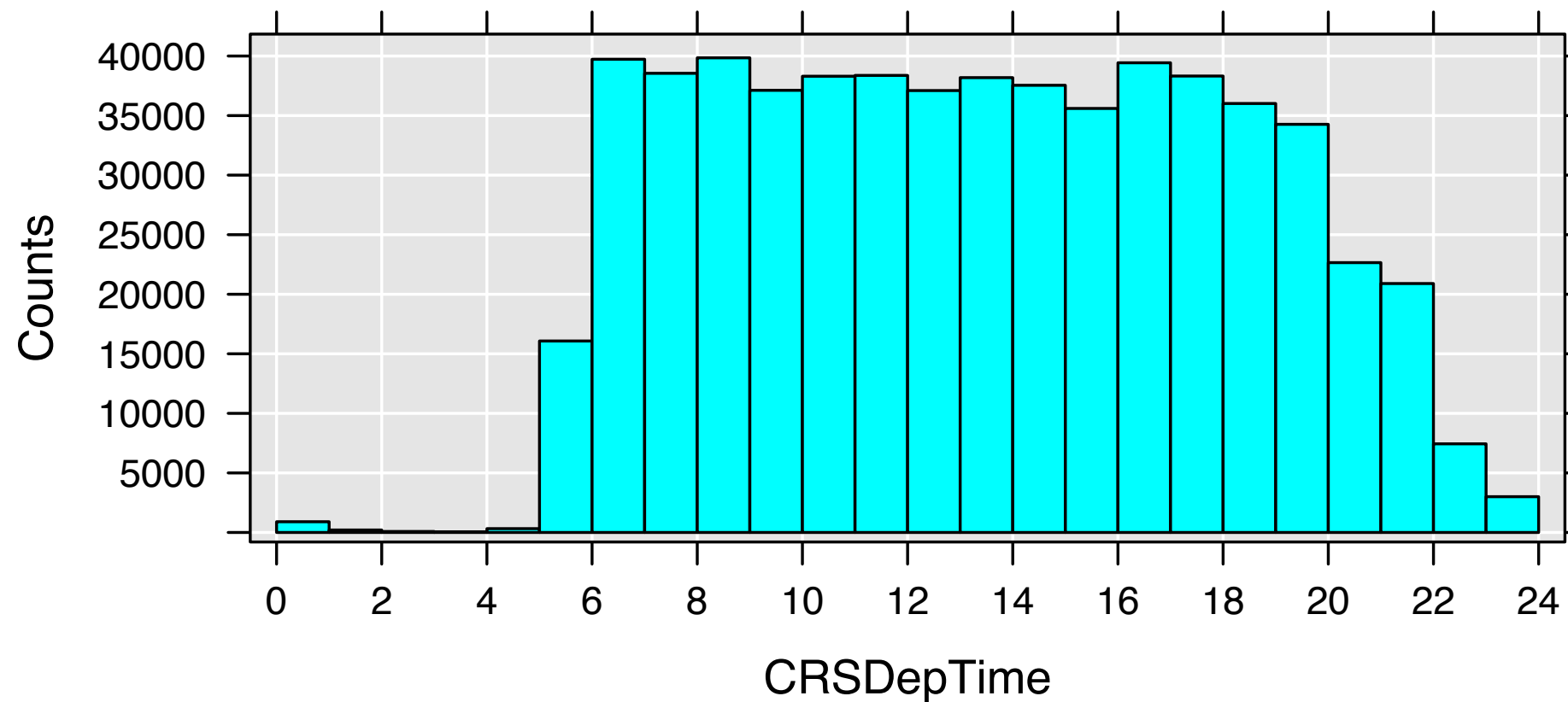
```
delayCube <- rxCube(ArrDelay ~ DayOfWeek, data = airDS, means = TRUE)  
data.frame(delaySumm$categorical[[1]][, c("DayOfWeek", "Means")], delayCube$ArrDelay)
```

##	DayOfWeek	Means	delayCube.ArrDelay
## 1	Monday	12.025604	12.025604
## 2	Tuesday	11.293808	11.293808
## 3	Wednesday	10.156539	10.156539
## 4	Thursday	8.658007	8.658007
## 5	Friday	14.804335	14.804335
## 6	Saturday	11.875326	11.875326
## 7	Sunday	10.331806	10.331806



Examine a Distribution of Departure Time

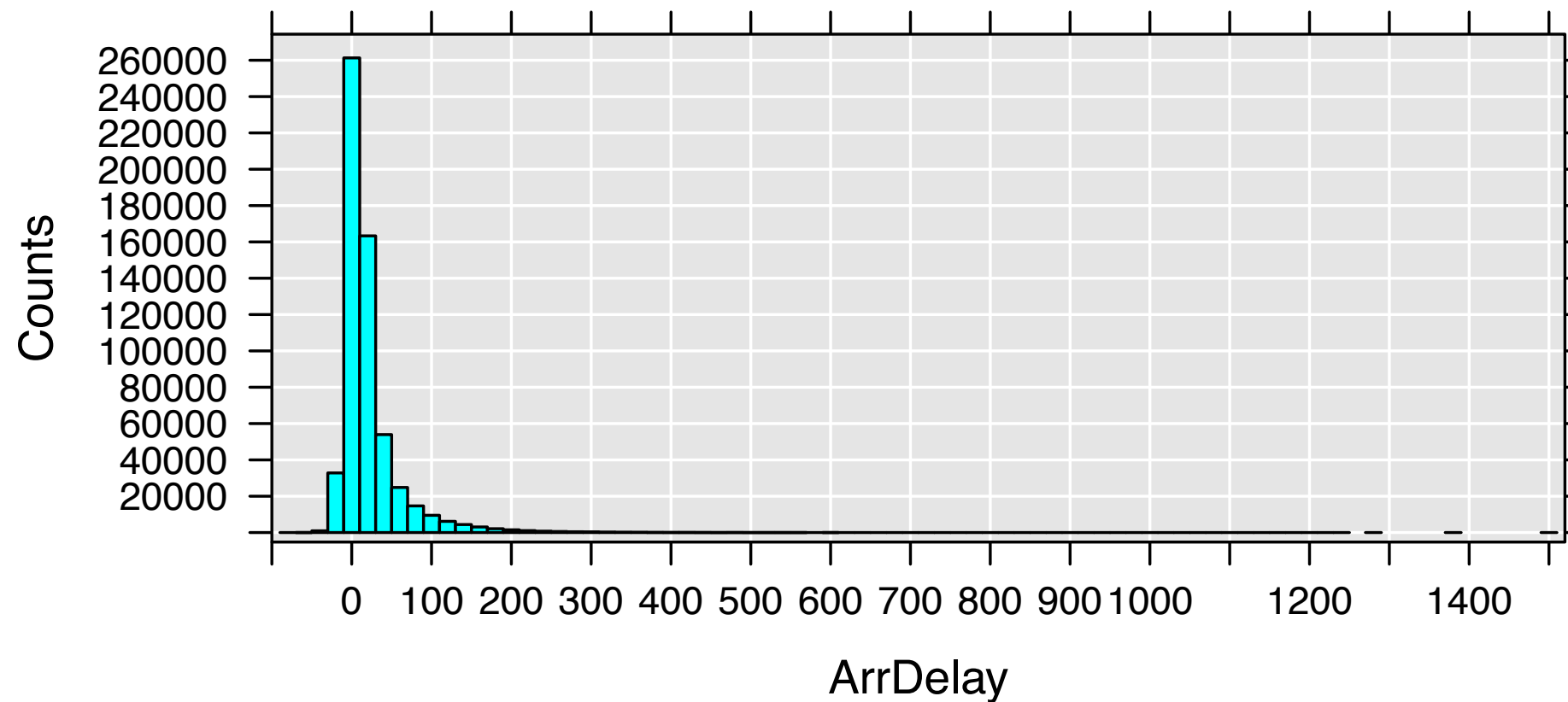
```
rxHistogram(~CRSDepTime, data = airDS, numBreaks = 25, xNumTicks = 13)
```





Examine a Distribution of Arrival Delay

```
rxHistogram(~ArrDelay, data = airDS, xNumTicks = 16)
```





Predict Arrival Delay by Day of Week

```
system.time(arrDelayLm1 <- rxLinMod(ArrDelay ~ DayOfWeek, data = airDS))
```

```
##      user  system elapsed  
##      0.06    0.00    0.11
```



Model Summary

```
summary(arrDelayLm1)

## Call:
## rxLinMod(formula = ArrDelay ~ DayOfWeek, data = airDS)
##
## Linear Regression Results for: ArrDelay ~ DayOfWeek
## Data: airDS (RxxdfData Data Source)
## File name: xdf/ADS.xdf
## Dependent variable(s): ArrDelay
## Total independent variables: 8 (Including number dropped: 1)
## Number of valid observations: 582628
## Number of missing observations: 17372
##
## Coefficients: (1 not defined because of singularities)
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    10.3318    0.1330  77.673 2.22e-16 ***
## DayOfWeek=Monday    1.6938    0.1872   9.049 2.22e-16 ***
## DayOfWeek=Tuesday    0.9620    0.2001   4.809 1.52e-06 ***
## DayOfWeek=Wednesday -0.1753    0.1980  -0.885   0.376
## DayOfWeek=Thursday  -1.6738    0.1964  -8.522 2.22e-16 ***
## DayOfWeek=Friday     4.4725    0.1957  22.850 2.22e-16 ***
## DayOfWeek=Saturday   1.5435    0.1934   7.981 2.22e-16 ***
## DayOfWeek=Sunday    Dropped    Dropped Dropped  Dropped
## ---
```



Model Summary (continued)

```
print(summary(arrDelayLm1), header = FALSE)

## Coefficients: (1 not defined because of singularities)
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    10.3318    0.1330   77.673 2.22e-16 ***
## DayOfWeek=Monday    1.6938    0.1872    9.049 2.22e-16 ***
## DayOfWeek=Tuesday    0.9620    0.2001    4.809 1.52e-06 ***
## DayOfWeek=Wednesday -0.1753    0.1980   -0.885  0.376
## DayOfWeek=Thursday  -1.6738    0.1964   -8.522 2.22e-16 ***
## DayOfWeek=Friday     4.4725    0.1957   22.850 2.22e-16 ***
## DayOfWeek=Saturday   1.5435    0.1934    7.981 2.22e-16 ***
## DayOfWeek=Sunday     Dropped    Dropped Dropped  Dropped
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 40.65 on 582621 degrees of freedom
## Multiple R-squared:  0.001869
## Adjusted R-squared:  0.001858
## F-statistic: 181.8 on 6 and 582621 DF,  p-value: < 2.2e-16
## Condition number: 10.5595
```



Argument: dropFirst=TRUE

```
arrDelayLm1df <- rxLinMod(ArrDelay ~ DayOfWeek, data = airDS, dropFirst = TRUE)
```



Results: dropFirst=TRUE

```
coef(summary(arrDelayLm1df)[[1]])
```

##	Estimate	Std. Error	t value	Pr(> t)
## (Intercept)	12.0256039	0.1316820	91.3230913	2.220446e-16
## DayOfWeek=Monday	NA	NA	NA	NA
## DayOfWeek=Tuesday	-0.7317962	0.1991674	-3.6742778	2.385445e-04
## DayOfWeek=Wednesday	-1.8690649	0.1971313	-9.4813216	2.220446e-16
## DayOfWeek=Thursday	-3.3675964	0.1954976	-17.2257730	2.220446e-16
## DayOfWeek=Friday	2.7787309	0.1948321	14.2621822	2.220446e-16
## DayOfWeek=Saturday	-0.1502774	0.1924772	-0.7807543	4.349473e-01
## DayOfWeek=Sunday	-1.6937981	0.1871726	-9.0493894	2.220446e-16



Argument: cube=TRUE

```
arrDelayLm1cube <- rxLinMod(ArrDelay ~ DayOfWeek, data = airDS, cube = TRUE)
```




Results: cube=TRUE

```
coef(summary(arrDelayLm1cube)[[1]])
```

##		Estimate	Std. Error	t value	Pr(> t)	Counts
##	DayOfWeek=Monday	12.025604	0.1316820	91.32309	2.220446e-16	95298
##	DayOfWeek=Tuesday	11.293808	0.1494239	75.58234	2.220446e-16	74011
##	DayOfWeek=Wednesday	10.156539	0.1466990	69.23386	2.220446e-16	76786
##	DayOfWeek=Thursday	8.658007	0.1444962	59.91858	2.220446e-16	79145
##	DayOfWeek=Friday	14.804335	0.1435946	103.09813	2.220446e-16	80142
##	DayOfWeek=Saturday	11.875326	0.1403829	84.59243	2.220446e-16	83851
##	DayOfWeek=Sunday	10.331806	0.1330168	77.67296	2.220446e-16	93395



Compare Results to delayCube

```
lm1est <- coef(arrDelayLm1)["(Intercept)"] + c(coef(arrDelayLm1)[-c(1, 8)], 0)
lm1dfest <- coef(arrDelayLm1df)["(Intercept)"] + c(0, coef(arrDelayLm1df)[-c(1, 2)])
lm1cubeest <- coef(arrDelayLm1cube)
data.frame(delayCube[, "ArrDelay", drop = FALSE], lm1est, lm1dfest, lm1cubeest, row.names = delayCube[,
  "DayOfWeek"])
```

##	ArrDelay	lm1est	lm1dfest	lm1cubeest
## Monday	12.025604	12.025604	12.025604	12.025604
## Tuesday	11.293808	11.293808	11.293808	11.293808
## Wednesday	10.156539	10.156539	10.156539	10.156539
## Thursday	8.658007	8.658007	8.658007	8.658007
## Friday	14.804335	14.804335	14.804335	14.804335
## Saturday	11.875326	11.875326	11.875326	11.875326
## Sunday	10.331806	10.331806	10.331806	10.331806



Elements of rxLinMod Objects

```
names(arrDelayLm1)
```

## [1] "coefficients"	"residual.squares"	"condition.number"	"rank"
## [5] "aliased"	"coef.std.error"	"coef.t.value"	"coef.p.value"
## [9] "total.squares"	"y.var"	"sigma"	"residual.variance"
## [13] "r.squared"	"f.pvalue"	"df"	"y.names"
## [17] "deviance"	"aic"	"params"	"formula"
## [21] "call"	"fststatistics"	"adj.r.squared"	"nValidObs"
## [25] "nMissingObs"	"coefLabelStyle"		



Additive Models

```
arrDelayLm3 <- rxLinMod(ArrDelay ~ DayOfWeek + CRSDepTime, data = airDS, cube = TRUE)
```



Additive Models: Results

```
coef(summary(arrDelayLm3)[[1]])
```

##		Estimate	Std. Error	t value	Pr(> t)	Counts
##	DayOfWeek=Monday	-1.1135823	0.19988456	-5.571127	2.532079e-08	95298
##	DayOfWeek=Tuesday	-1.8544247	0.21191691	-8.750716	2.220446e-16	74011
##	DayOfWeek=Wednesday	-3.0430310	0.21045458	-14.459325	2.220446e-16	76786
##	DayOfWeek=Thursday	-4.5175933	0.20874452	-21.641733	2.220446e-16	79145
##	DayOfWeek=Friday	1.6058429	0.20832119	7.708495	2.220446e-16	80142
##	DayOfWeek=Saturday	-1.0049295	0.20346881	-4.938985	7.855173e-07	83851
##	DayOfWeek=Sunday	-3.1945789	0.20412829	-15.649859	2.220446e-16	93395
##	CRSDepTime	0.9786211	0.01125522	86.948171	2.220446e-16	NaN



Interactive Models

Only the interaction:

```
arrDelayLm4 <- rxLinMod(ArrDelay ~ DayOfWeek:CRSDepTime, data = airDS)
```

Main Effects and Interactions:

```
arrDelayLm4b <- rxLinMod(ArrDelay ~ DayOfWeek * CRSDepTime, data = airDS)
```



Inline Transformations

```
arrDelayLm5 <- rxLinMod(log(abs(ArrDelay)) ~ DayOfWeek:F(CRSDepTime), data = airDS)
```



More Complex Transforms

```
system.time(  
  arrDelayLm6 <-  
    rxLinMod(ArrDelay ~ DayOfWeek:depTimeCat,  
              data = airDS,  
              transforms=list(depTimeCat =  
                              cut(CRSDepTime,  
                                  breaks=seq(from = 5, to = 23, by = 2)  
                                )  
            )  
)  
  
##      user  system elapsed  
##    0.27    0.02    0.28
```




rxLinMod and Text Files: Setup

```
inputDataSource <- RxTextData(file = inputFile, colInfo = colInfo, missingValueString = "M")
```



rxLinMod and Text Files: Run

```
system.time(  
  arrDelayLm6b <-  
    rxLinMod(ArrDelay ~ DayOfWeek:depTimeCat,  
              data = inputDataSource,  
              transforms=list(depTimeCat =  
                              cut(CRSDepTime,  
                                  breaks=seq(from = 5,to = 23, by = 2)  
                                )  
            )  
  )  
)
```

```
##      user  system elapsed  
##      1.83    0.05     2.16
```



Same procedure afterwards

```
coef(summary(arrDelayLm6b)[[1]])
```

##	Estimate	Std. Error	t value
## (Intercept)	14.3121890547	0.5812290	24.6240114461
## DayOfWeek=Monday, depTimeCat=(5,7]	-10.6937623767	0.7155505	-14.9448052737
## DayOfWeek=Tuesday, depTimeCat=(5,7]	-10.6104011470	0.7484972	-14.1756058396
## DayOfWeek=Wednesday, depTimeCat=(5,7]	-12.8586943580	0.7479515	-17.1918827866
## DayOfWeek=Thursday, depTimeCat=(5,7]	-14.1515975383	0.7410982	-19.0954418535
## DayOfWeek=Friday, depTimeCat=(5,7]	-10.7489603552	0.7393418	-14.5385526380
## DayOfWeek=Saturday, depTimeCat=(5,7]	-8.5746353524	0.7373944	-11.6282885727
## DayOfWeek=Sunday, depTimeCat=(5,7]	-12.2679527435	0.7567371	-16.2116441288
## DayOfWeek=Monday, depTimeCat=(7,9]	-8.4609244864	0.6833984	-12.3806621672
## DayOfWeek=Tuesday, depTimeCat=(7,9]	-7.4557272978	0.7093253	-10.5110124827
## DayOfWeek=Wednesday, depTimeCat=(7,9]	-9.9322050836	0.7078763	-14.0309904391
## DayOfWeek=Thursday, depTimeCat=(7,9]	-12.1323997827	0.7027991	-17.2629702965
## DayOfWeek=Friday, depTimeCat=(7,9]	-7.6800706037	0.7017865	-10.9436003770
## DayOfWeek=Saturday, depTimeCat=(7,9]	-6.6614208629	0.6894919	-9.6613469219
## DayOfWeek=Sunday, depTimeCat=(7,9]	-12.0232803688	0.6949892	-17.2999521053
## DayOfWeek=Monday, depTimeCat=(9,11]	-4.6174917687	0.6884164	-6.7074109563
## DayOfWeek=Tuesday, depTimeCat=(9,11]	-5.4519443727	0.7193369	-7.5791252249
## DayOfWeek=Wednesday, depTimeCat=(9,11]	-8.6766525561	0.7129235	-12.1705243282
## DayOfWeek=Thursday, depTimeCat=(9,11]	-10.3031073191	0.7087136	-14.5377587653
## DayOfWeek=Friday, depTimeCat=(9,11]	-4.9864834096	0.7085966	-7.0371254536

...



rxPredict()

```
args(RevoScaleR:::rxPredict.default)
```

```
## function (modelObject, data = NULL, outData = NULL, computeStdErrors = FALSE,  
##      interval = "none", confLevel = 0.95, computeResiduals = FALSE,  
##      type = c("response", "link"), writeModelVars = FALSE, removeMissings = FALSE,  
##      overwrite = FALSE, checkFactorLevels = TRUE, predVarNames = NULL,  
##      residVarNames = NULL, intervalVarNames = NULL, stdErrorsVarNames = NULL,  
##      predNames = NULL, blocksPerRead = rxGetOption("blocksPerRead"),  
##      reportProgress = rxGetOption("reportProgress"), verbose = 0,  
##      xdfCompressionLevel = rxGetOption("xdfCompressionLevel"),  
##      ...)  
## NULL
```



Review the Model

```
system.time(  
  arrDelayLm <- rxLinMod(  
    ArrDelay ~ DayOfWeek:depTimeCat,  
    data = airDS, cube=TRUE,  
    transforms=list(depTimeCat =  
                     cut(CRSDepTime, breaks=seq(from=5, to=23, by=2))  
                   )  
  )  
)  
  
##      user  system elapsed  
##    0.31    0.00    0.32
```



Model Results

```
summary(arrDelayLm)

## Call:
## rxLinMod(formula = ArrDelay ~ DayOfWeek:depTimeCat, data = airDS,
##          cube = TRUE, transforms = list(depTimeCat = cut(CRSDepTime,
##                  breaks = seq(from = 5, to = 23, by = 2))))
##
## Cube Linear Regression Results for: ArrDelay ~ DayOfWeek:depTimeCat
## Data: airDS (RxXdfData Data Source)
## File name: xdf/ADS.xdf
## Dependent variable(s): ArrDelay
## Total independent variables: 63
## Number of valid observations: 578128
## Number of missing observations: 21872
##
## Coefficients:
##                                     Estimate Std. Error t value Pr(>|t|)      | Counts
## DayOfWeek=Monday, depTimeCat=(5,7]      3.6184      0.4174   8.670 2.22e-16 *** |   9356
## DayOfWeek=Tuesday, depTimeCat=(5,7]     3.7018      0.4716   7.849 2.22e-16 *** |   7327
## DayOfWeek=Wednesday, depTimeCat=(5,7]    1.4535      0.4707   3.088 0.00202 **  |   7354
## DayOfWeek=Thursday, depTimeCat=(5,7]     0.1606      0.4598   0.349 0.72688      |   7709
## DayOfWeek=Friday, depTimeCat=(5,7]       3.5632      0.4569   7.798 2.22e-16 *** |   7805
## DayOfWeek=Saturday, depTimeCat=(5,7]     5.7376      0.4538  12.644 2.22e-16 *** |   7914
```



Predicting Values

```
rxPredict(modelObject = arrDelayLm, data = airDS, outData = airDS, overwrite = TRUE)
```



View Predictions

```
rxGetVarInfo(airDS)
```

```
## Var 1: ArrDelay, Type: integer, Low/High: (-86, 1490)
## Var 2: CRSDepTime, Type: numeric, Storage: float32, Low/High: (0.0167, 23.9833)
## Var 3: DayOfWeek
##           7 factor levels: Monday Tuesday Wednesday Thursday Friday Saturday Sunday
## Var 4: urv, Type: numeric, Low/High: (0.0000, 1.0000)
## Var 5: TrainTest
##           2 factor levels: Train Test
## Var 6: ArrDelay_Pred, Type: numeric, Low/High: (0.1606, 25.1872)
```




Compute Residuals

```
rxPredict(modelObject = arrDelayLm, data = airDS, outData = airDS, computeResiduals = TRUE,  
          overwrite = TRUE)
```



View Residuals

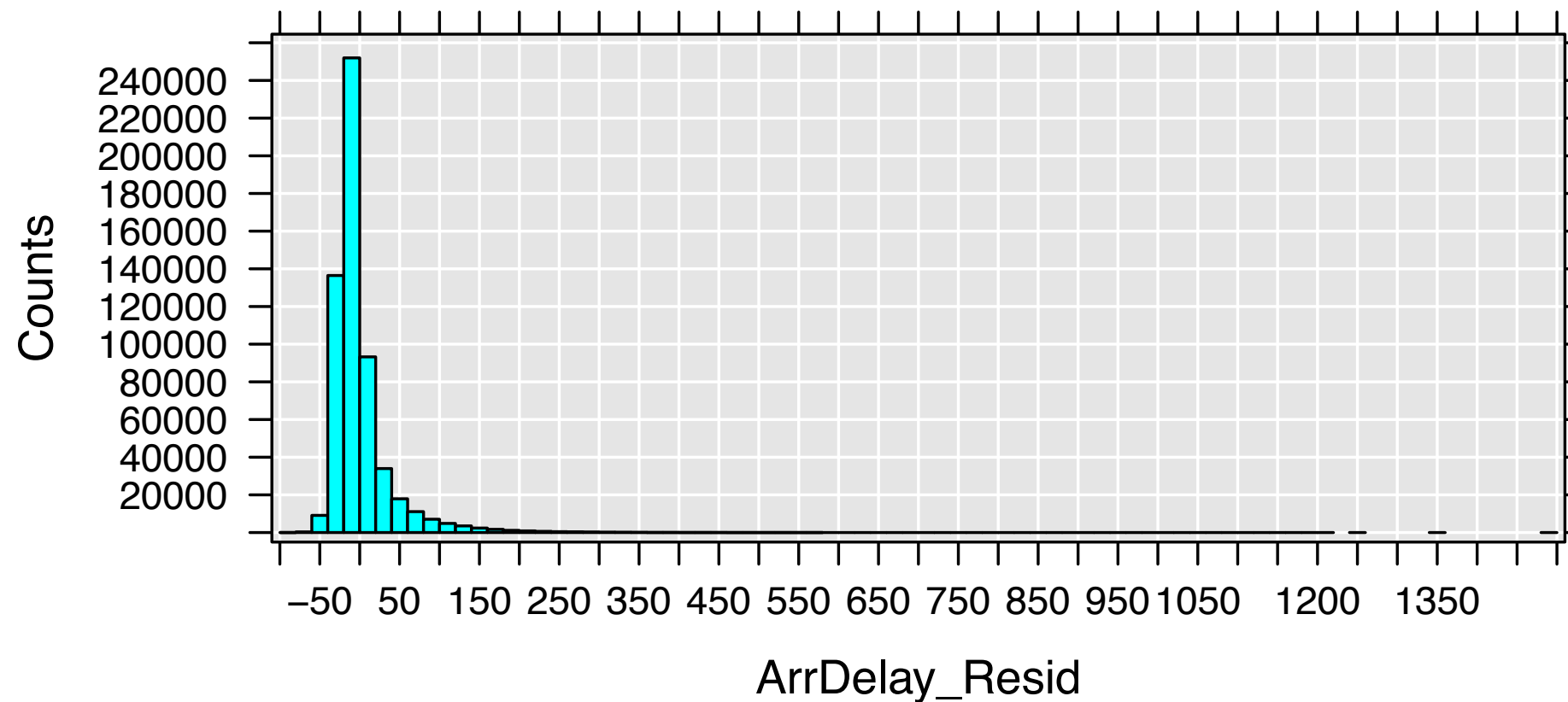
```
rxGetVarInfo(airDS)
```

```
## Var 1: ArrDelay, Type: integer, Low/High: (-86, 1490)
## Var 2: CRSDepTime, Type: numeric, Storage: float32, Low/High: (0.0167, 23.9833)
## Var 3: DayOfWeek
##          7 factor levels: Monday Tuesday Wednesday Thursday Friday Saturday Sunday
## Var 4: urv, Type: numeric, Low/High: (0.0000, 1.0000)
## Var 5: TrainTest
##          2 factor levels: Train Test
## Var 6: ArrDelay_Pred, Type: numeric, Low/High: (0.1606, 25.1872)
## Var 7: ArrDelay_Resid, Type: numeric, Low/High: (-97.6677, 1486.4368)
```



Examine the Distribution of Residuals

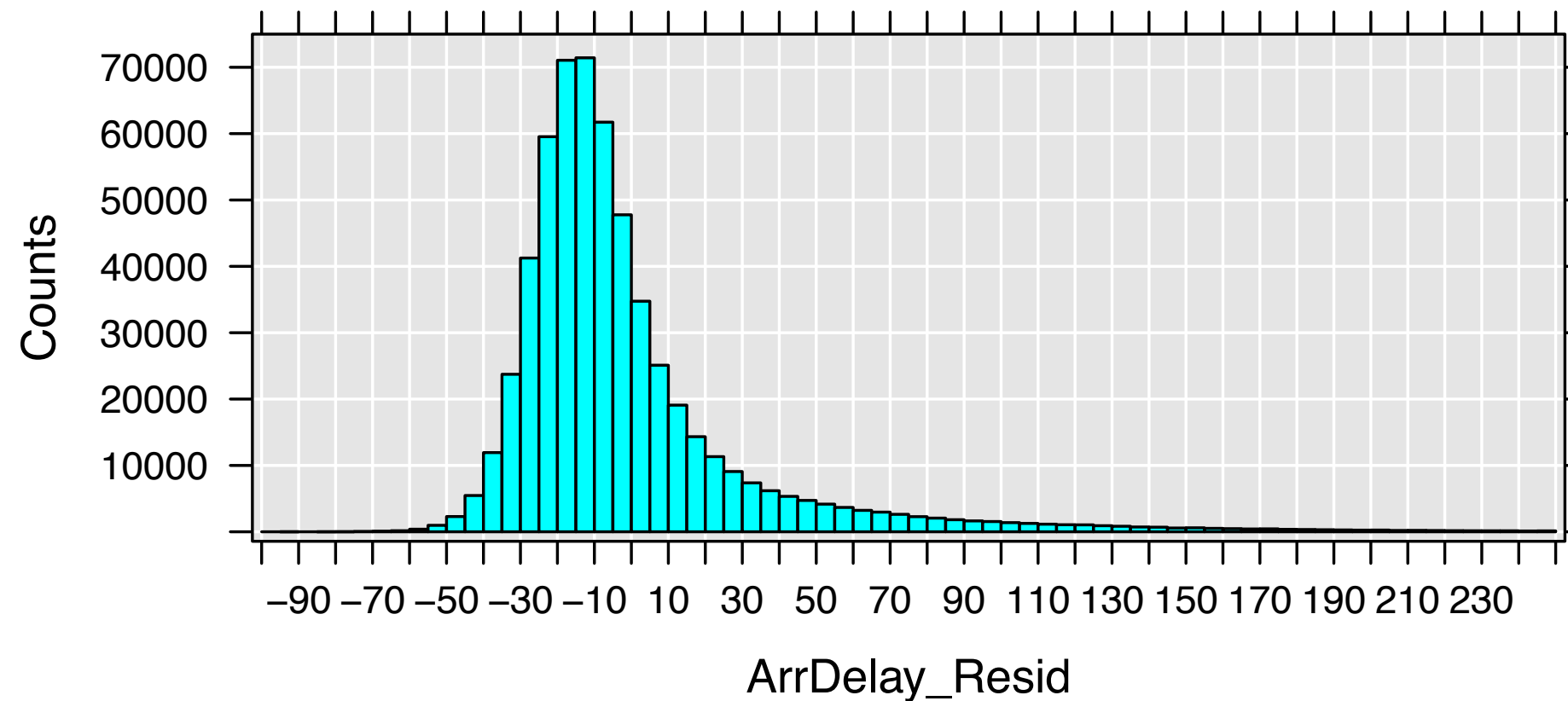
```
rxHistogram(~ArrDelay_Resid, data = airDS)
```





Examine the Distribution in More Detail

```
rxHistogram(~ArrDelay_Resid, data = airDS, numBreaks = 100, startVal = -100, endVal = 250)
```





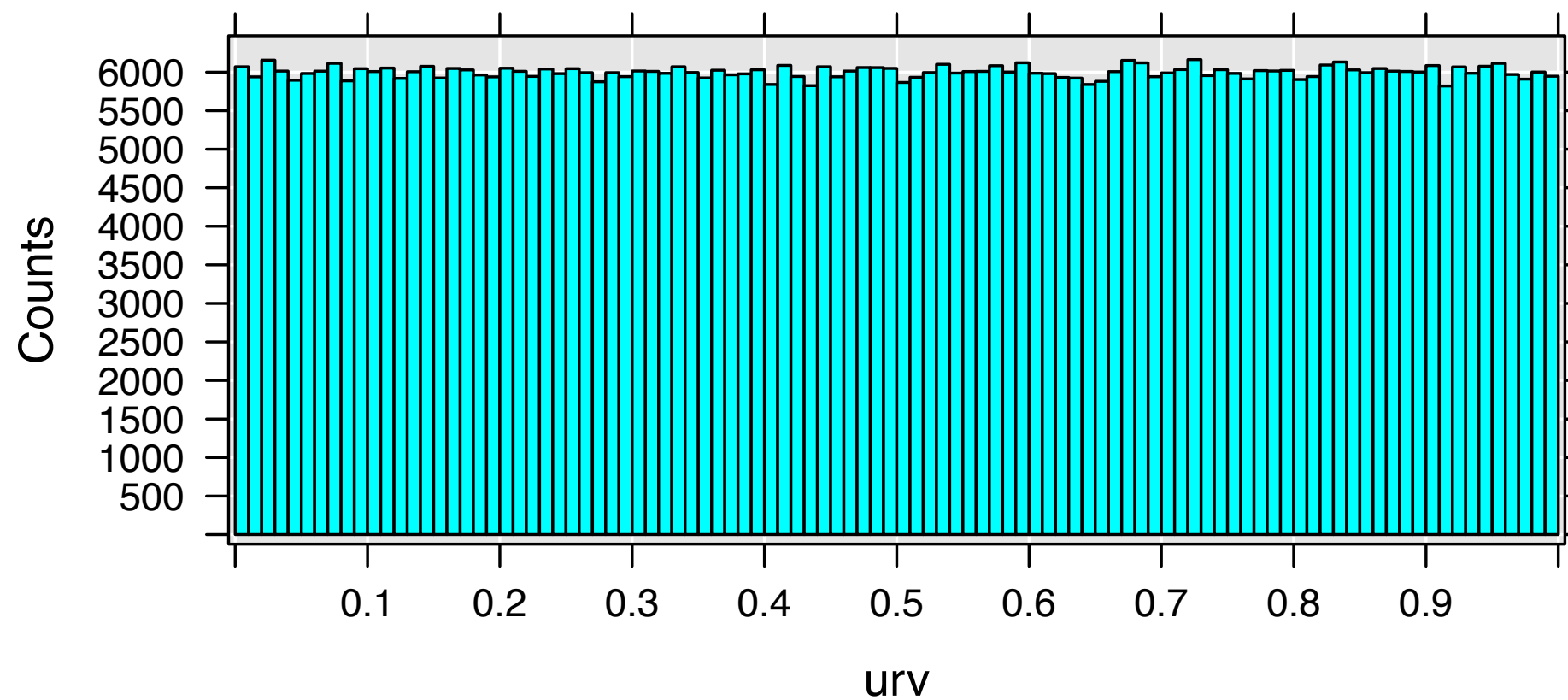
Training and Validation Setup

```
rxDataStep(inData = airDS, outFile = airDS, transforms = list(urv = runif(length(DayOfWeek))),  
  append = "cols", overwrite = TRUE)
```



URV Visualization

```
rxHistogram(~urv, airDS, xNumTicks = 10)
```





Converting urv to a Factor Variable

```
rxDataStep(inData = airDS, outFile = airDS,  
           transforms=list(TrainTest = cut(urv, breaks = c(0,0.8,1),  
                                           labels = c("Train", "Test"))),  
           append = "cols", overwrite = TRUE)
```



View TrainTest

```
rxSummary(~TrainTest, data = airDS)

## Call:
## rxSummary(formula = ~TrainTest, data = airDS)
##
## Summary Statistics Results for: ~TrainTest
## Data: airDS (RxXdfData Data Source)
## File name: xdf/ADS.xdf
## Number of valid observations: 600000
##
##
## Category Counts for TrainTest
## Number of categories: 2
## Number of valid observations: 600000
## Number of missing observations: 0
##
## TrainTest Counts
## Train      479854
## Test       120146
```




Splitting the dataset

```
mySplit <- rxSplit(inData = airDS, splitByFactor = "TrainTest",  
                  rowSelection = ArrDelay < 450,  
                  overwrite = TRUE)
```



View the New File Names

```
names(mySplit)

## [1] "E:\\BigData\\DataCamp_Analysis\\doc\\ADS.TrainTest.Train.xdf"
## [2] "E:\\BigData\\DataCamp_Analysis\\doc\\ADS.TrainTest.Test.xdf"

trainDS <- mySplit[[1]]
testDS <- mySplit[[2]]
```



Create Training Model

```
system.time(  
  arrDelayLm.train <- rxLinMod(  
    ArrDelay ~ DayOfWeek:depTimeCat,  
    data = trainDS,  
    cube=TRUE,  
    transforms=list(  
      depTimeCat = cut(CRSDepTime, breaks=seq(from=5, to=23, by=2)))  
    )  
  )  
  
##      user  system elapsed  
##      0.24    0.02     0.25
```



Cross-Validate Model

```
rxPredict(modelObject = arrDelayLm.train, data = testDS, outData = testDS, computeResiduals = TRUE,  
  overwrite = TRUE)  
rxGetVarInfo(testDS)
```

```
## Var 1: ArrDelay, Type: integer, Low/High: (-86, 1490)  
## Var 2: CRSDepTime, Type: numeric, Low/High: (0.0167, 23.9833)  
## Var 3: DayOfWeek  
##      7 factor levels: Monday Tuesday Wednesday Thursday Friday Saturday Sunday  
## Var 4: urv, Type: numeric, Low/High: (0.0000, 1.0000)  
## Var 5: TrainTest  
##      2 factor levels: Train Test  
## Var 6: ArrDelay_Pred, Type: numeric, Low/High: (0.0812, 25.1872)  
## Var 7: ArrDelay_Resid, Type: numeric, Low/High: (-97.6677, 1486.4368)
```



Correlate Predicted and Actual Values

```
(pred.act.cors <- rxCor(formula = ~ArrDelay_Pred + ArrDelay, data = testDS))
```

```
##           ArrDelay_Pred  ArrDelay
## ArrDelay_Pred      1.0000000 0.1358719
## ArrDelay           0.1358719 1.0000000
```

```
arrDelayLm.train$r.squared
```

```
## [1] 0.02078747
```

```
pred.act.cors[1, 2]^2
```

```
## [1] 0.01846116
```



Computing Std. Errors

```
try(rxPredict(modelObject = arrDelayLm.train, data = testDS, outData = testDS, computeResiduals = TRUE,  
  computeStdErrors = TRUE, overwrite = TRUE))
```

```
##
```

```
## The estimated variance-covariance matrix of the coefficients must be available in order to compute prediction standard errors
```



Argument: covCoef

```
system.time(arrDelayLm.train <- rxLinMod(ArrDelay ~ DayOfWeek:depTimeCat, data = trainDS,  
  transforms = list(depTimeCat = cut(CRSDepTime, breaks = seq(from = 5, to = 23, by = 2))),  
  covCoef = TRUE))
```

```
##      user  system elapsed  
##      0.23    0.03    0.26
```



Generating Std. Errors

```
rxPredict(modelObject = arrDelayLm.train, data = testDS, outData = testDS, computeResiduals = TRUE,  
  computeStdErrors = TRUE, interval = "prediction", overwrite = TRUE)  
rxGetVarInfo(testDS)
```

```
## Var 1: ArrDelay, Type: integer, Low/High: (-86, 1490)  
## Var 2: CRSDepTime, Type: numeric, Low/High: (0.0167, 23.9833)  
## Var 3: DayOfWeek  
##      7 factor levels: Monday Tuesday Wednesday Thursday Friday Saturday Sunday  
## Var 4: urv, Type: numeric, Low/High: (0.0000, 1.0000)  
## Var 5: TrainTest  
##      2 factor levels: Train Test  
## Var 6: ArrDelay_Pred, Type: numeric, Low/High: (0.0812, 25.1872)  
## Var 7: ArrDelay_Resid, Type: numeric, Low/High: (-97.6677, 1486.4368)  
## Var 8: ArrDelay_StdErr, Type: numeric, Low/High: (0.3784, 0.8077)  
## Var 9: ArrDelay_Lower, Type: numeric, Low/High: (-74.5331, -49.5667)  
## Var 10: ArrDelay_Upper, Type: numeric, Low/High: (74.6955, 99.6713)
```




Mortgage Data

Set the path:

```
mortXdf <- file.path(rxGetOption("sampleDataDir"), "mortDefaultSmall.xdf")
```

Review the Data

```
rxGetInfo(data = mortXdf, getVarInfo = TRUE)

## File name: C:\Revolution\R-Enterprise-7.2\R-3.0.3\library\RevoScaleR\SampleData\mortDefaultSmall.xdf
## Number of observations: 100000
## Number of variables: 6
## Number of blocks: 10
## Compression type: zlib
## Variable information:
## Var 1: creditScore, Type: integer, Low/High: (470, 925)
## Var 2: houseAge, Type: integer, Low/High: (0, 40)
## Var 3: yearsEmploy, Type: integer, Low/High: (0, 14)
## Var 4: ccDebt, Type: integer, Low/High: (0, 14094)
## Var 5: year, Type: integer, Low/High: (2000, 2009)
## Var 6: default, Type: integer, Low/High: (0, 1)
```



rxLogit() Arguments

`args(rxLogit)`

```
## function (formula, data, pweights = NULL, fweights = NULL, cube = FALSE,  
##   cubePredictions = FALSE, variableSelection = list(), rowSelection = NULL,  
##   transforms = NULL, transformObjects = NULL, transformFunc = NULL,  
##   transformVars = NULL, transformPackages = NULL, transformEnvir = NULL,  
##   dropFirst = FALSE, dropMain = rxGetOption("dropMain"), covCoef = FALSE,  
##   covData = FALSE, covariance = FALSE, initialValues = NULL,  
##   coefLabelStyle = rxGetOption("coefLabelStyle"), blocksPerRead = rxGetOption("blocksPerRead"),  
##   maxIterations = 25, coeffTolerance = 0.000001, gradientTolerance = 0.000001,  
##   objectiveFunctionTolerance = 0.00000001, reportProgress = rxGetOption("reportProgress"),  
##   verbose = 0, computeContext = rxGetOption("computeContext"),  
##   ...)  
## NULL
```



Logistic Regression: Estimation

```
logitModel1 <-  
  rxLogit(  
    default ~ F(year) + ccDebt + creditScore + houseAge + yearsEmploy,  
    data = mortXdf,  
    dropFirst = TRUE  
  )
```



Logistic Regression: Summary

```
summary(logitModel1)

## Call:
## rxLogit(formula = default ~ F(year) + ccDebt + creditScore +
##         houseAge + yearsEmploy, data = mortXdf, dropFirst = TRUE)
##
## Logistic Regression Results for: default ~ F(year) + ccDebt + creditScore +
##         houseAge + yearsEmploy
## Data: mortXdf (RxxdfData Data Source)
## File name:
##         C:/Revolution/R-Enterprise-7.2/R-3.0.3/library/RevoScaleR/SampleData/mortDefaultSmall.xdf
## Dependent variable(s): default
## Total independent variables: 15 (Including number dropped: 1)
## Number of valid observations: 100000
## Number of missing observations: 0
## -2*LogLikelihood: 2946.1416 (Residual deviance on 99986 degrees of freedom)
##
...

```



Summary (continued)

```
##               Estimate      Std. Error  z value Pr(>|z|)
## (Intercept) -10.676300691    0.868163310 -12.2976 2.22e-16 ***
## F_year=2000           NA           NA           NA           NA
## F_year=2001     1.036735208    0.377298214   2.7478  0.0060 **
## F_year=2002    -0.059887745    0.451414000  -0.1327  0.8945
## F_year=2003    -0.906620754    0.595801034  -1.5217  0.1281
## F_year=2004    -0.754256100    0.530412342  -1.4220  0.1550
## F_year=2005    -0.795193515    0.553608298  -1.4364  0.1509
## F_year=2006    -0.805887276    0.563593927  -1.4299  0.1527
## F_year=2007     0.607389800    0.398804738   1.5230  0.1278
## F_year=2008     3.168104457    0.330552944   9.5843 2.22e-16 ***
## F_year=2009     3.737629855    0.327482184  11.4132 2.22e-16 ***
## ccDebt          0.001319981    0.000039315  33.5745 2.22e-16 ***
## creditScore    -0.007823851    0.001080576  -7.2404 2.22e-16 ***
## houseAge        0.028769237    0.007035559   4.0891 4.33e-05 ***
## yearsEmploy    -0.267533265    0.027405533  -9.7620 2.22e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```



Prediction Data

```
(newData <- data.frame(  
  year = rep(c(2006, 2009), each=4),  
  ccDebt = rep(c(1000, 10000), 4),  
  creditScore = rep(c(700, 800), 4),  
  houseAge = rep(c(1, 5, 10, 20), 2),  
  yearsEmploy = 7  
))
```

##	year	ccDebt	creditScore	houseAge	yearsEmploy
## 1	2006	1000	700	1	7
## 2	2006	10000	800	5	7
## 3	2006	1000	700	10	7
## 4	2006	10000	800	20	7
## 5	2009	1000	700	1	7
## 6	2009	10000	800	5	7
## 7	2009	1000	700	10	7
## 8	2009	10000	800	20	7



Predicted Values

```
(dataWithPredictions <- rxPredict(  
  modelObject = logitModel1,  
  data = newData,  
  outData = newData,  
  type = "response"))
```

##	year	ccDebt	creditScore	houseAge	yearsEmploy	default_Pred
## 1	2006	1000	700	1	7	0.00000002554393
## 2	2006	10000	800	5	7	0.00188800972037
## 3	2006	1000	700	10	7	0.00000003309304
## 4	2006	10000	800	20	7	0.00290386751027
## 5	2009	1000	700	1	7	0.00000240165789
## 6	2009	10000	800	5	7	0.15099413007025
## 7	2009	1000	700	10	7	0.00000311142776
## 8	2009	10000	800	20	7	0.21495934590121



Census Data

```
dataPath = "../../../Data/"
bigCensusData <- file.path(dataPath, "Census5PCT2000.xdf")
rxGetInfo(bigCensusData)

## File name: E:\GitHub\Revolution_Course_Materials\Data\Census5PCT2000.xdf
## Number of observations: 14058983
## Number of variables: 265
## Number of blocks: 98
## Compression type: none
```




Subset Data

```
propinFile <- file.path("xdf/CensusPropertyIns.xdf")
rxDataStep(inData = bigCensusData,
  rowSelection = (related == "Head/Householder") & (age > 20) & (age < 90),
  varsToKeep = c("propinsr", "age", "sex", "region", "perwt"),
  outFile = propinFile,
  blocksPerRead = 10,
  overwrite = TRUE
)
```



View Subset Data

```
rxGetInfoXdf(propinFile, getVarInfo = TRUE)
```

```
## File name: E:\GitHub\Revolution_Course_Materials\modules\BigData\DataCamp_Analysis\doc\xdf\CensusPropertyIns.xdf
```

```
## Number of observations: 5175270
```

```
## Number of variables: 5
```

```
## Number of blocks: 10
```

```
## Compression type: zlib
```

```
## Variable information:
```

```
## Var 1: propinsr, Annual property insurance cost
```

```
##           Type: integer, Low/High: (0, 3700)
```

```
## Var 2: age, Age
```

```
##           Type: integer, Low/High: (21, 89)
```

```
## Var 3: sex, Sex
```

```
##           2 factor levels: Male Female
```

```
## Var 4: region, Census region and division
```

```
##           17 factor levels: New England Division Middle Atlantic Division Mixed Northeast Divisions (1970 Metro) East North Cent
```

```
## Var 5: perwt, Type: integer, Low/High: (2, 193)
```



Summarize Factor

```
print(rxSummary(~region, data = propinFile), header = FALSE)
```

```
##
## Category Counts for region
## Number of categories: 17
## Number of valid observations: 5175270
## Number of missing observations: 0
##
## region                      Counts
## New England Division       265372
## Middle Atlantic Division   734585
## Mixed Northeast Divisions (1970 Metro) 0
## East North Central Div.    847367
## West North Central Div.    366417
## Mixed Midwest Divisions (1970 Metro) 0
## South Atlantic Division    981614
## East South Central Div.    324003
## West South Central Div.    553425
## Mixed Southern Divisions (1970 Metro) 0
## Mountain Division          328940
## Pacific Division           773547
## Mixed Western Divisions (1970 Metro) 0
## Military/Military reservations 0
## PUMA boundaries cross state lines-1% sample 0
```



Clean Factor Variables

```
regionLevels <- list(`New England` = "New England Division", `Middle Atlantic` = "Middle Atlantic Division",  
  `East North Central` = "East North Central Div.", `West North Central` = "West North Central Div.",  
  `South Atlantic` = "South Atlantic Division", `East South Central` = "East South Central Div.",  
  `West South Central` = "West South Central Div.", Mountain = "Mountain Division",  
  Pacific = "Pacific Division")  
  
rxFactors(inData = propinFile, outFile = propinFile, factorInfo = list(region = list(newLevels = regionLevels,  
  otherLevel = "Other")), overwrite = TRUE)
```



Re-Summarize Factor

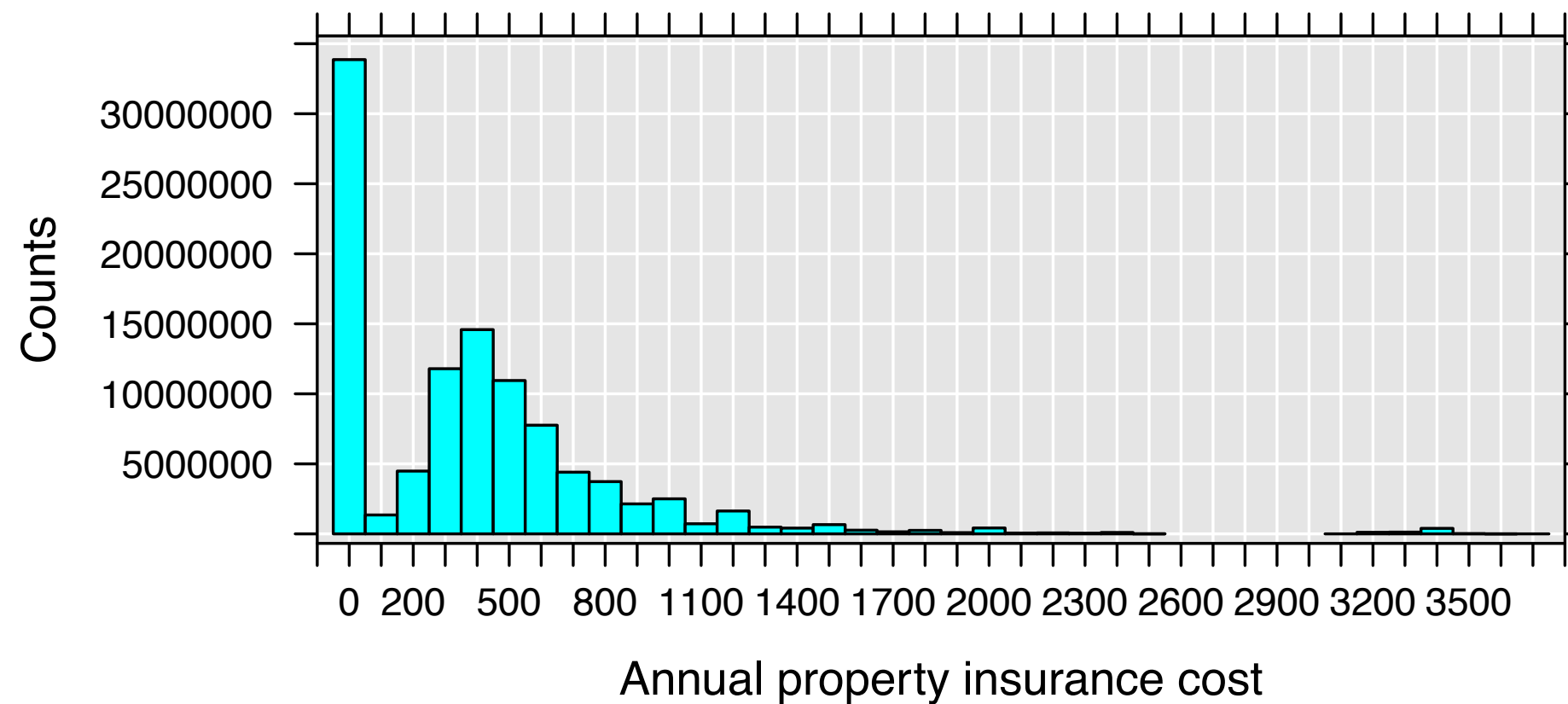
```
print(rxSummary(~region, data = propinFile), header = FALSE)
```

```
##  
## Category Counts for region  
## Number of categories: 10  
## Number of valid observations: 5175270  
## Number of missing observations: 0  
##  
##   region          Counts  
## New England      265372  
## Middle Atlantic   734585  
## East North Central 847367  
## West North Central 366417  
## South Atlantic    981614  
## East South Central 324003  
## West South Central 553425  
## Mountain          328940  
## Pacific           773547  
## Other              0
```



Property Insurance: Histogram

```
rxHistogram(~propinsr, data = propinFile, pweights = "perwt", numBreaks = 50)
```





rxGlm() Arguments

`args(rxGlm)`

```
## function (formula, data, family = gaussian(), pweights = NULL,  
##      fweights = NULL, offset = NULL, cube = FALSE, variableSelection = list(),  
##      rowSelection = NULL, transforms = NULL, transformObjects = NULL,  
##      transformFunc = NULL, transformVars = NULL, transformPackages = NULL,  
##      transformEnvir = NULL, dropFirst = FALSE, dropMain = rxGetOption("dropMain"),  
##      covCoef = FALSE, computeAIC = FALSE, initialValues = NA,  
##      coefLabelStyle = rxGetOption("coefLabelStyle"), blocksPerRead = rxGetOption("blocksPerRead"),  
##      maxIterations = 25, coeffTolerance = 0.000001, objectiveFunctionTolerance = 0.00000001,  
##      reportProgress = rxGetOption("reportProgress"), verbose = 0,  
##      computeContext = rxGetOption("computeContext"), ...)  
## NULL
```



GLM with Tweedie Family

```
system.time(  
  propinGlm <-  
    rxGlm(propinsr~sex + F(age) + region,  
          pweights = "perwt",  
          data = propinFile,  
          family = rxTweedie(var.power = 1.5),  
          dropFirst = TRUE  
    )  
)
```

```
##      user  system elapsed  
##    32.40     6.40    39.58
```




GLM with Tweedie Family: Output

```
printCoefmat(coef(summary(propinGlm)))
```

##	Estimate	Std. Error	t value	Pr(> t)	
## (Intercept)	0.123131629	0.000589255	208.9614	2.220e-16	***
## sex=Male	NA	NA	NA	NA	
## sex=Female	0.009026344	0.000031637	285.3054	2.220e-16	***
## F_age=21	NA	NA	NA	NA	
## F_age=22	-0.009208145	0.000752297	-12.2400	2.220e-16	***
## F_age=23	-0.019803733	0.000696578	-28.4300	2.220e-16	***
## F_age=24	-0.028556094	0.000664796	-42.9546	2.220e-16	***
## F_age=25	-0.036518683	0.000643205	-56.7761	2.220e-16	***
## F_age=26	-0.043705574	0.000628859	-69.4998	2.220e-16	***
## F_age=27	-0.048937559	0.000618199	-79.1616	2.220e-16	***
## F_age=28	-0.053979330	0.000609898	-88.5055	2.220e-16	***
## F_age=29	-0.057865092	0.000604339	-95.7494	2.220e-16	***
## F_age=30	-0.060635022	0.000602040	-100.7160	2.220e-16	***
## F_age=31	-0.063358444	0.000600426	-105.5224	2.220e-16	***
## F_age=32	-0.065257338	0.000599059	-108.9332	2.220e-16	***
## F_age=33	-0.067210867	0.000597490	-112.4887	2.220e-16	***
## F_age=34	-0.068536775	0.000596242	-114.9478	2.220e-16	***
## F_age=35	-0.069415252	0.000594879	-116.6879	2.220e-16	***
## F_age=36	-0.070896305	0.000594058	-119.3425	2.220e-16	***
## F_age=37	-0.071837352	0.000593582	-121.0234	2.220e-16	***

...



Data To Predict

```
Ages <- 21:89
predData <-
  data.frame(age = Ages,
    sex = gl(2, length(Ages), labels = c("Male", "Female")),
    region = factor(rep(c(5, 2), each =
      length(Ages)*2),
    levels = 1:10, labels = rxGetVarInfo(propinFile)$region$levels)
  )
nrow(predData)

## [1] 276
```



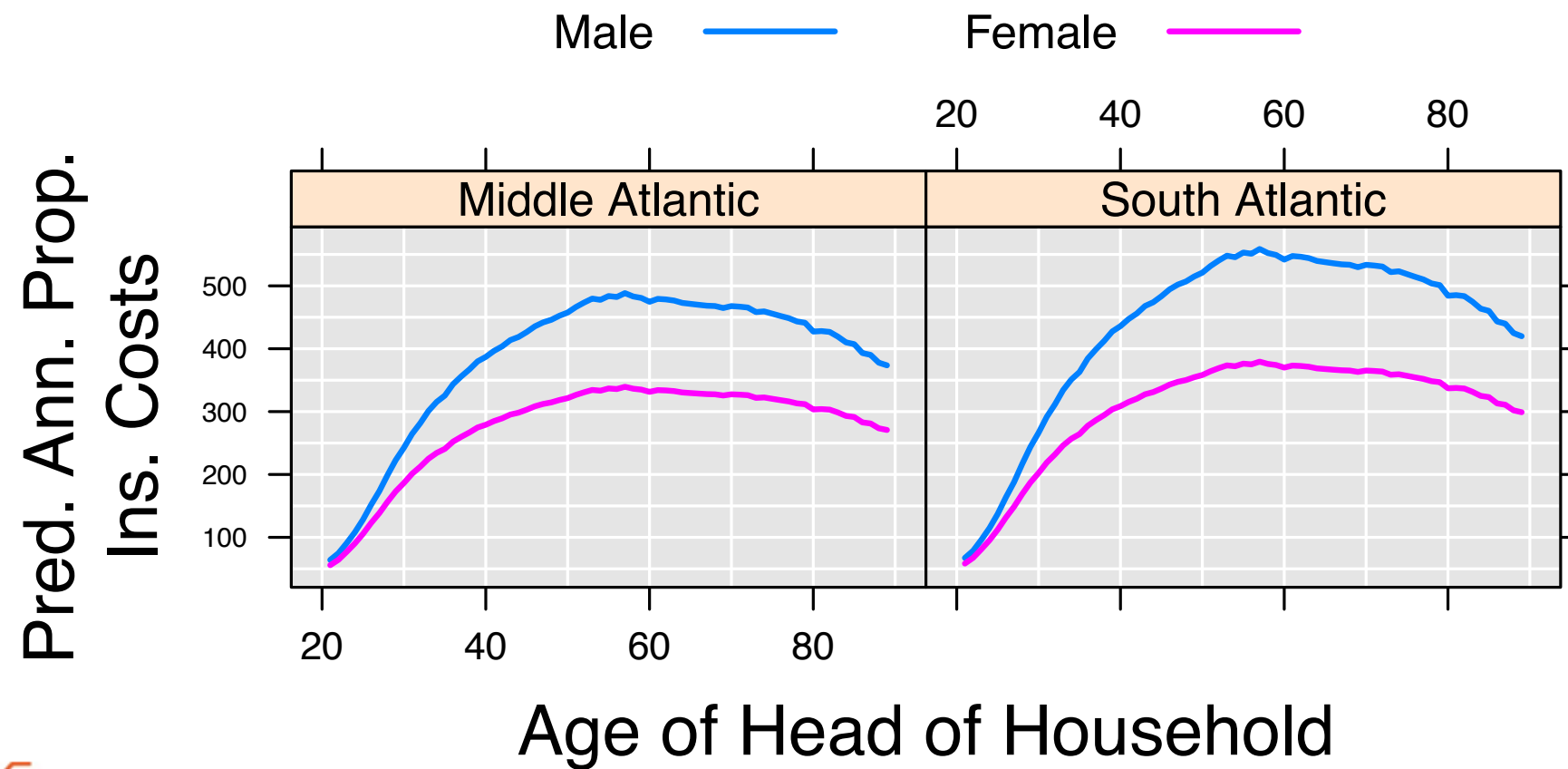
Predicted Data for GLM

```
predData <- rxPredict(propinGlm,  
  data = predData,  
  outData=predData,  
  predVarNames=c("predicted"),  
  overwrite=TRUE)
```



Plot Predictions

```
rxLinePlot(predicted ~age|region, group = sex, data = predData,  
  auto.key=list(columns=2, points=FALSE, lines=TRUE),  
  xTitle = list("Age of Head of Household", cex=1.5),  
  yTitle = list("Pred. Ann. Prop.\nIns. Costs",cex=1.5),  
  scales=list(y=list(cex=0.6))  
)
```





Data

```
sampleDataDir <- rxGetOption("sampleDataDir")  
mdata <- file.path(sampleDataDir, "mortDefaultSmall.xdf")  
mortDefault2 <- file.path("xdf/mortDefault2.xdf")
```



Create New Variables

```
rxDataStep(inData = mdata, outFile = mortDefault2,  
           transforms = list(RandomSample = sample(10, size = .rxNumRows, replace = TRUE)),  
           overwrite = TRUE)  
rxGetVarInfo(mortDefault2)
```

```
## Var 1: creditScore, Type: integer, Low/High: (470, 925)  
## Var 2: houseAge, Type: integer, Low/High: (0, 40)  
## Var 3: yearsEmploy, Type: integer, Low/High: (0, 14)  
## Var 4: ccDebt, Type: integer, Low/High: (0, 14094)  
## Var 5: year, Type: integer, Low/High: (2000, 2009)  
## Var 6: default, Type: integer, Low/High: (0, 1)  
## Var 7: RandomSample, Type: integer, Low/High: (1, 10)
```



rxKmeans()

`args(rxKmeans)`

```
## function (formula, data, outFile = NULL, outColName = ".rxCluster",  
##     writeModelVars = FALSE, overwrite = FALSE, numClusters = NULL,  
##     centers = NULL, algorithm = "Lloyd", numStartRows = 0, maxIterations = 1000,  
##     numStarts = 1, rowSelection = NULL, transforms = NULL, transformObjects = NULL,  
##     transformFunc = NULL, transformVars = NULL, transformPackages = NULL,  
##     transformEnvir = NULL, blocksPerRead = rxGetOption("blocksPerRead"),  
##     reportProgress = rxGetOption("reportProgress"), verbose = 0,  
##     computeContext = rxGetOption("computeContext"), xdfCompressionLevel = rxGetOption("xdfCompressionLevel"),  
##     ...)  
## NULL
```



Estimate the Clusters

```
md.km <- rxKmeans(formula = ~ creditScore + houseAge + yearsEmploy + ccDebt + year,  
                  data = mortDefault2,  
                  numClusters = 3,  
                  outFile = mortDefault2,  
                  algorithm = "lloyd",  
                  overwrite = TRUE)
```




View the Output

```
print(md.km)

## Call:
## rxKmeans(formula = ~creditScore + houseAge + yearsEmploy + ccDebt +
##   year, data = mortDefault2, outFile = mortDefault2, overwrite = TRUE,
##   numClusters = 3, algorithm = "lloyd")
##
## Data: mortDefault2
## Number of valid observations: 100000
## Number of missing observations: 0
## Clustering algorithm:
##
## K-means clustering with 3 clusters of sizes 26816, 27364, 45820
##
## Cluster means:
##   creditScore houseAge yearsEmploy   ccDebt     year
## 1    699.8421 19.95693    5.013723 2555.351 2004.494
## 2    699.5454 19.96609    4.996199 7433.085 2004.520
## 3    700.1138 19.95853    4.999956 4987.269 2004.492
##
## Within cluster sum of squares by cluster:
##           1           2           3
## 24302543771 27215486100 21700772048
##
```



Clusters

```
rxGetVarInfo(mortDefault2)
```

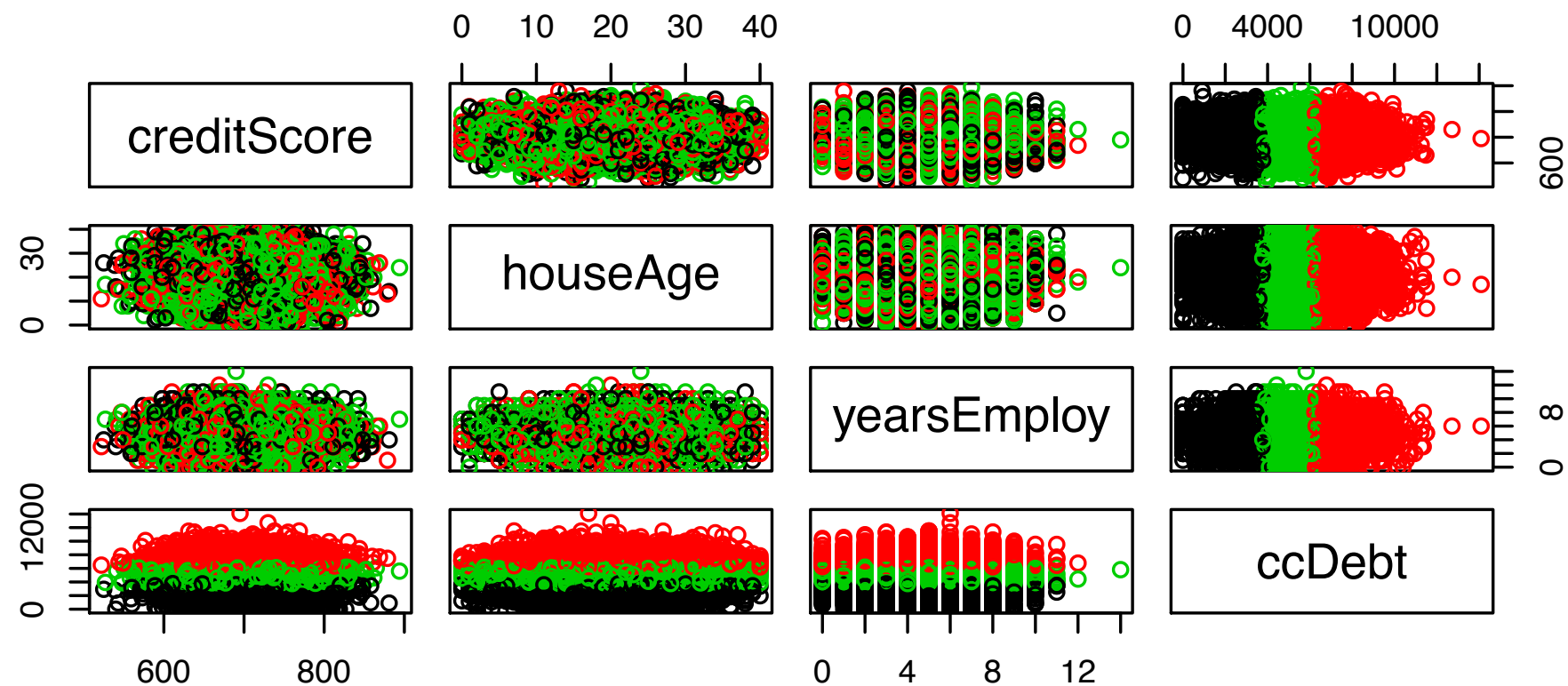
```
## Var 1: creditScore, Type: integer, Low/High: (470, 925)
## Var 2: houseAge, Type: integer, Low/High: (0, 40)
## Var 3: yearsEmploy, Type: integer, Low/High: (0, 14)
## Var 4: ccDebt, Type: integer, Low/High: (0, 14094)
## Var 5: year, Type: integer, Low/High: (2000, 2009)
## Var 6: default, Type: integer, Low/High: (0, 1)
## Var 7: RandomSample, Type: integer, Low/High: (1, 10)
## Var 8: .rxCluster, Type: integer, Low/High: (1, 3)
```



Plot Clusters

```
mdDf <- rxXdfToDataFrame(file=mortDefault2,  
                          rowSelection = RandomSample == 5)  
plot(mdDf[,1:4], col=mdDf$.rxCluster)
```

Clusters in Mortgage Default Data





Decision Trees: Data

mortDefault2 was created in the last session

```
mortDefault2 <- file.path("xdf/mortDefault2.xdf")
rxGetInfo(mortDefault2, getVarInfo = TRUE)

## File name: E:\GitHub\Revolution_Course_Materials\modules\BigData\DataCamp_Analysis\doc\xdf\mortDefault2.xdf
## Number of observations: 100000
## Number of variables: 8
## Number of blocks: 10
## Compression type: zlib
## Variable information:
## Var 1: creditScore, Type: integer, Low/High: (470, 925)
## Var 2: houseAge, Type: integer, Low/High: (0, 40)
## Var 3: yearsEmploy, Type: integer, Low/High: (0, 14)
## Var 4: ccDebt, Type: integer, Low/High: (0, 14094)
## Var 5: year, Type: integer, Low/High: (2000, 2009)
## Var 6: default, Type: integer, Low/High: (0, 1)
## Var 7: RandomSample, Type: integer, Low/High: (1, 10)
## Var 8: .rxCluster, Type: integer, Low/High: (1, 3)
```



Split Files into Training and Testing

```
trainTestFiles <- rxSplit(  
  inData = mortDefault2,  
  transforms = list(TrainTest=factor(ifelse(RandomSample < 9,"Train","Test"))),  
  splitByFactor="TrainTest", overwrite=TRUE  
)
```



rxDTree Arguments

`args(rxDTree)`

```
## function (formula, data, outFile = NULL, outColName = ".rxNode",  
##      writeModelVars = FALSE, overwrite = FALSE, pweights = NULL,  
##      fweights = NULL, method = NULL, parms = NULL, cost = NULL,  
##      minSplit = max(20, sqrt(numObs)), minBucket = round(minSplit/3),  
##      maxDepth = 10, cp = 0, maxCompete = 0, maxSurrogate = 0,  
##      useSurrogate = 2, surrogateStyle = 0, xVal = 2, maxNumBins = NULL,  
##      maxUnorderedLevels = 32, removeMissings = FALSE, pruneCp = 0,  
##      rowSelection = NULL, transforms = NULL, transformObjects = NULL,  
##      transformFunc = NULL, transformVars = NULL, transformPackages = NULL,  
##      transformEnvir = NULL, blocksPerRead = rxGetOption("blocksPerRead"),  
##      reportProgress = rxGetOption("reportProgress"), verbose = 0,  
##      computeContext = rxGetOption("computeContext"), xdfCompressionLevel = rxGetOption("xdfCompressionLevel"),  
##      ...)  
## NULL
```



Estimate a Regression Tree

```
treeR <- rxDTree(  
  formula = default ~ houseAge + year + creditScore + yearsEmploy + ccDebt,  
  data=trainTestFiles[[1]],  
  maxdepth=5  
)
```



Print the Tree

```
treeR

## Call:
## rxDTree(formula = default ~ houseAge + year + creditScore + yearsEmploy +
##         ccDebt, data = trainTestFiles[[1]], maxdepth = 5)
## File:   E:\GitHub\Revolution_Course_Materials\modules\BigData\DataCamp_Analysis\doc\mortDefault2.TrainTest.Test.xdf
## Number of valid observations:  20241
## Number of missing observations:  0
##
## Tree representation:
## n= 20241
##
## node), split, n, deviance, yval
##      * denotes terminal node
##
##  1) root 20241 97.5255200 0.0048416580
##    2) ccDebt< 9215 19955 59.8195900 0.0030067650
##      4) ccDebt< 8230 19237 34.9363200 0.0018194110
##        8) ccDebt< 6765 16536  5.9978230 0.0003628447
##          16) ccDebt< 5880 13706  0.0000000 0.00000000000 *
##            17) ccDebt>=5880 2830  5.9872790 0.0021201410
##              34) year< 2007.5 2287  0.0000000 0.00000000000 *
##                35) year>=2007.5 543  5.9337020 0.0110497200 *
```

...



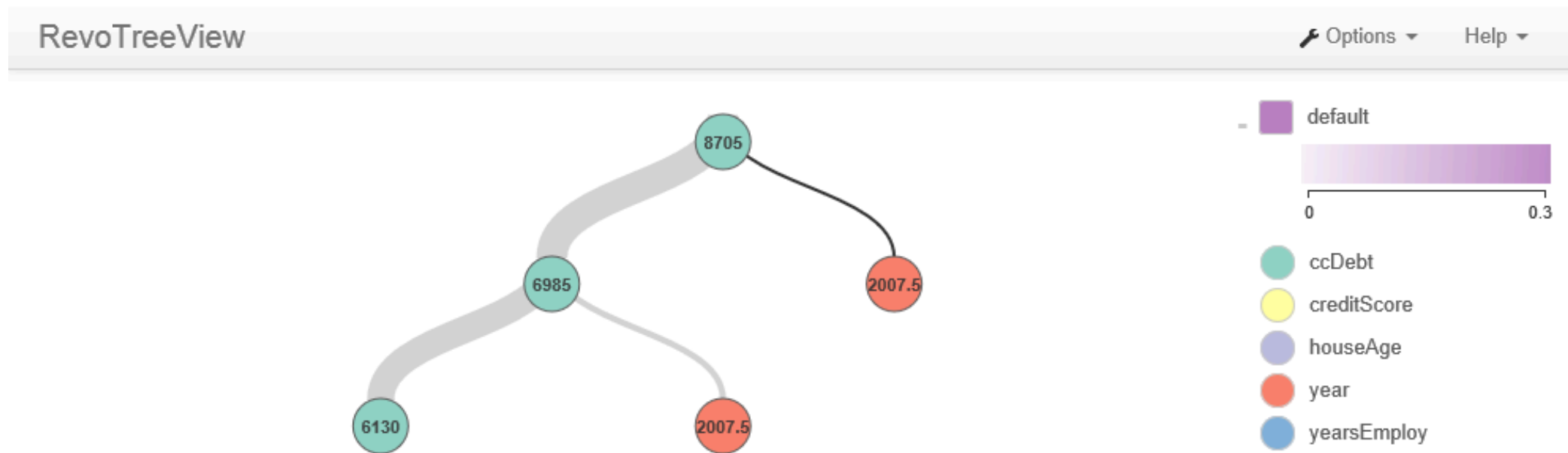
Regression Tree for Mortgage Data





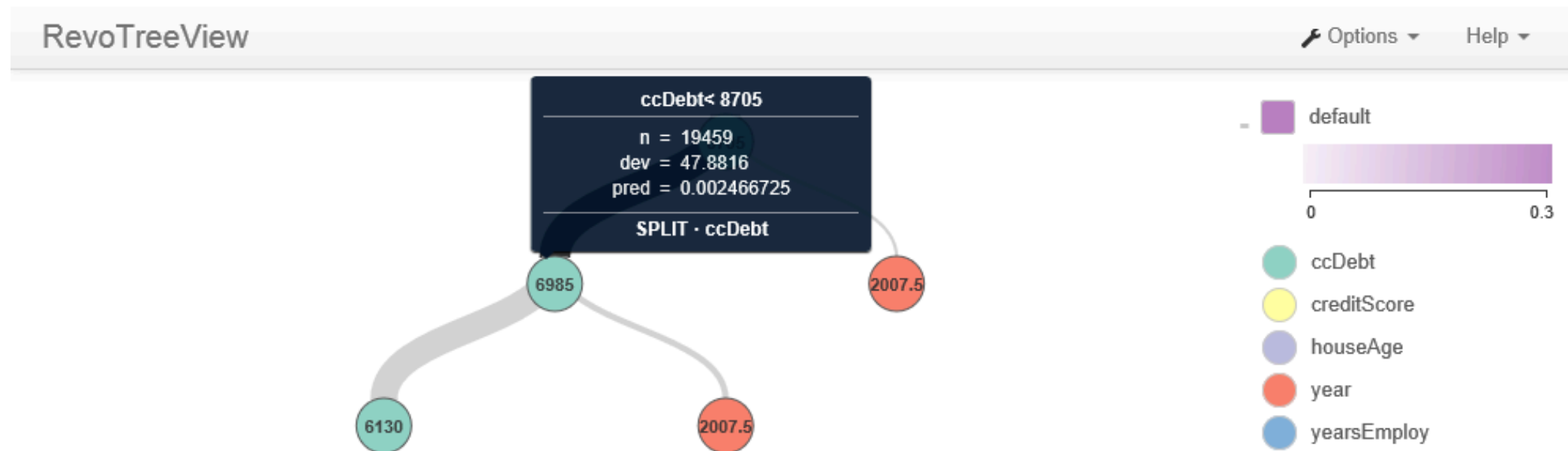
RevoTreeView

```
library(RevoTreeView)
plot(createTreeView(treeR))
```



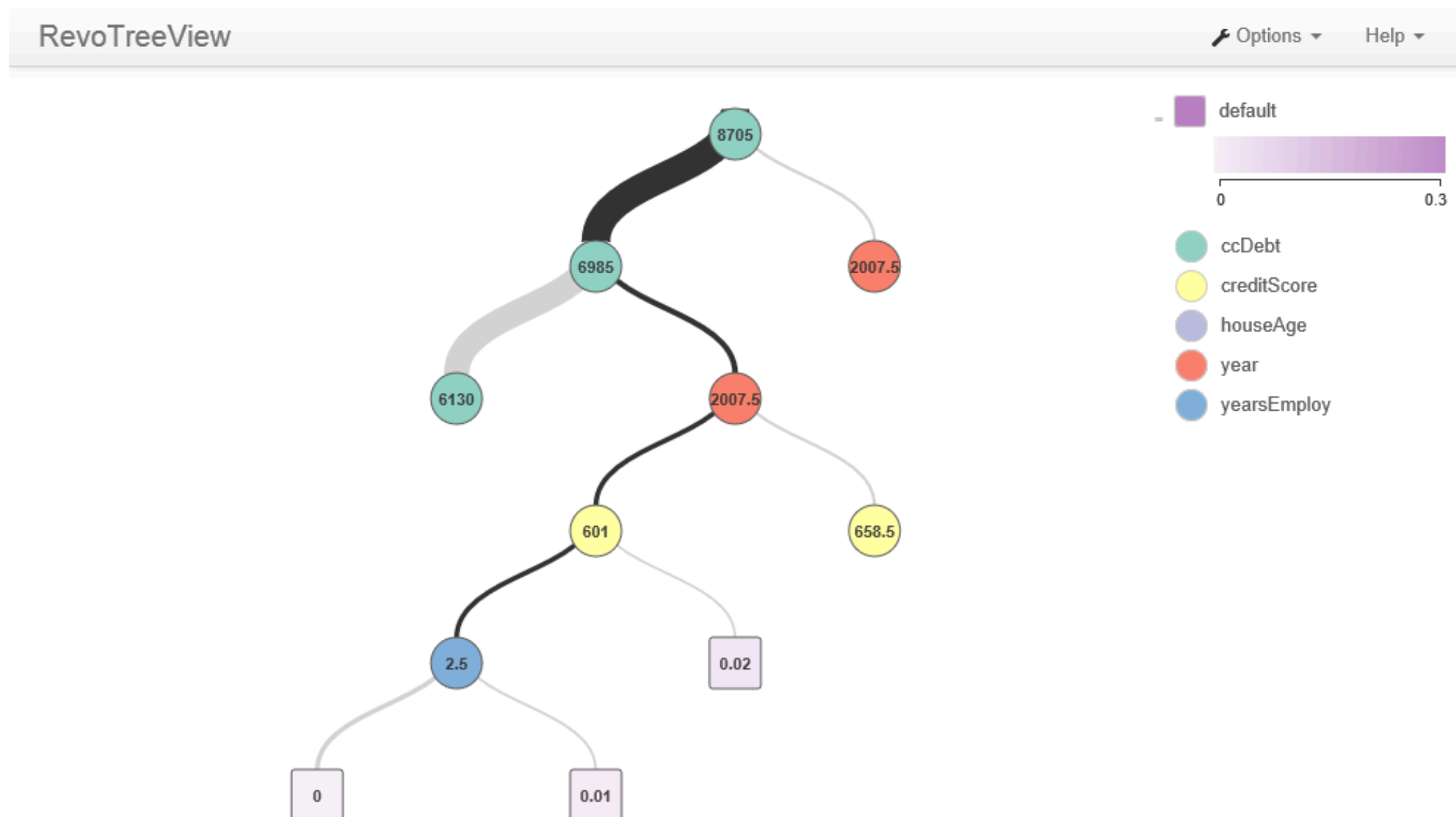


RevoTreeView (Mouseover)





RevoTreeView (Deep)





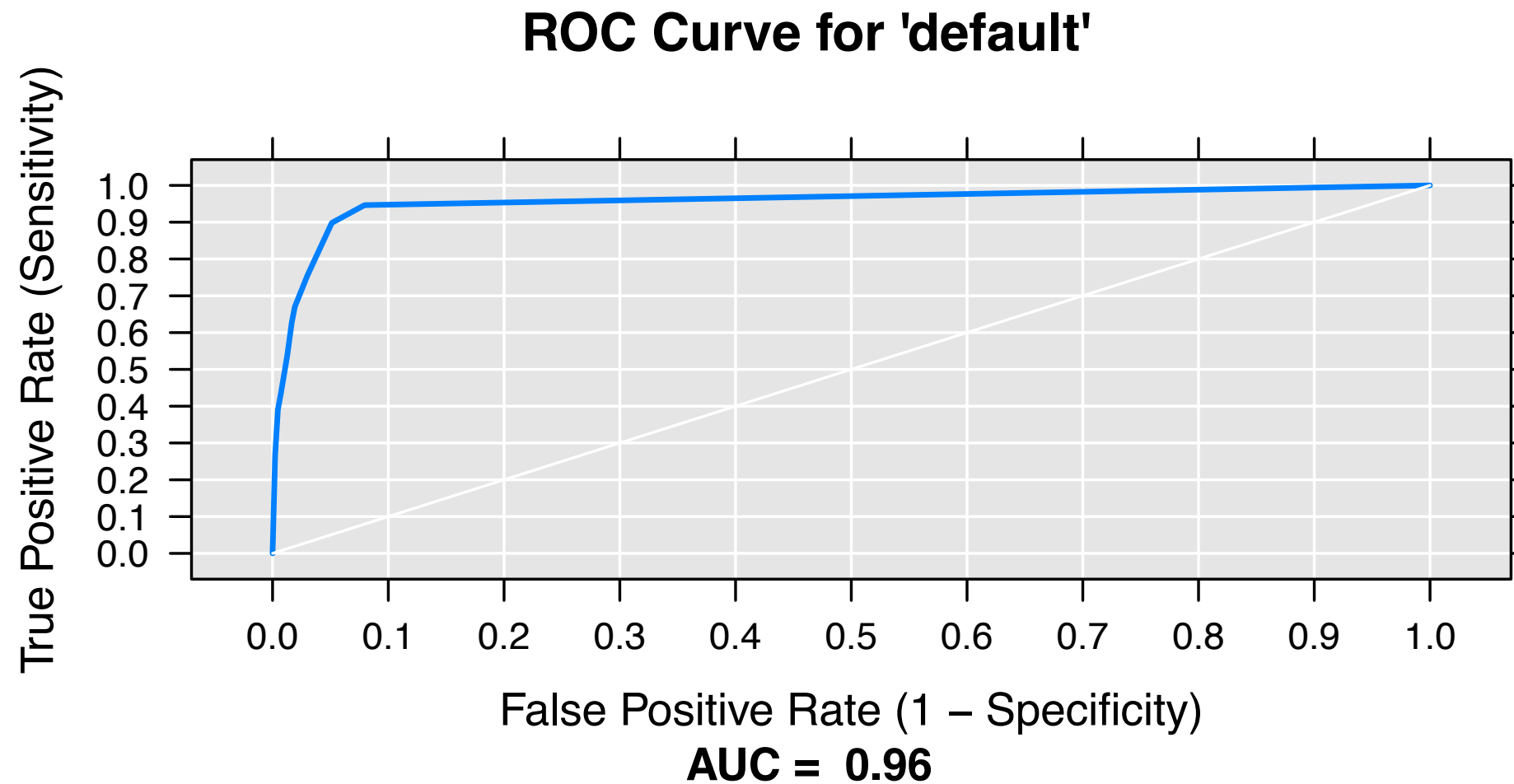
Tree Predictions

```
rxPredict(treeR, data = trainTestFiles[[2]], outData = trainTestFiles[[2]], overwrite = TRUE,  
          predVarNames = "Pred_R")
```



ROC Curves

```
rxRocCurve(actualVarName = "default", predVarNames = c("Pred_R"), data = trainTestFiles[[2]])
```





Build factor for Classification Tree

Training Dataset:

```
rxFactors(inData = trainTestFiles[[1]], factorInfo = list(defaultFactor = list(varName = "default")),  
          outFile = trainTestFiles[[1]], overwrite = TRUE)
```

Testing Dataset:

```
rxFactors(inData = trainTestFiles[[2]], factorInfo = list(defaultFactor = list(varName = "default")),  
          outFile = trainTestFiles[[2]], overwrite = TRUE)
```



View defaultFactor

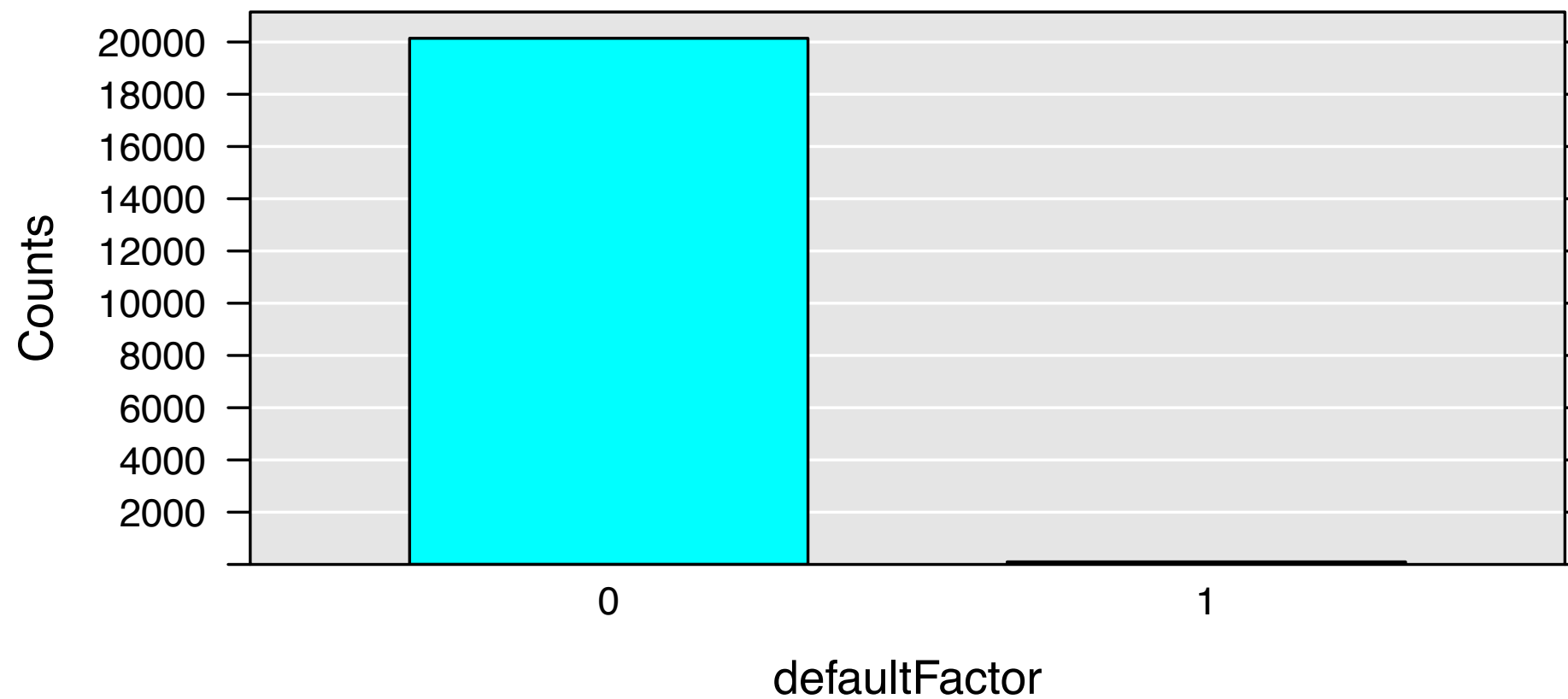
```
rxGetInfo(trainTestFiles[[1]], getVarInfo = TRUE, varsToKeep = c("default", "defaultFactor"))
```

```
## File name: E:\GitHub\Revolution_Course_Materials\modules\BigData\DataCamp_Analysis\doc\mortDefault2.TrainTest.Test.xdf
## Number of observations: 20241
## Number of variables: 15
## Number of blocks: 10
## Compression type: zlib
## Variable information:
## Var 1: default, Type: integer, Low/High: (0, 1)
## Var 2: defaultFactor
##           2 factor levels: 0 1
```




Histogram of Default Factors

```
rxHistogram(~defaultFactor, data = trainTestFiles[[1]])
```





Build a Classification Tree

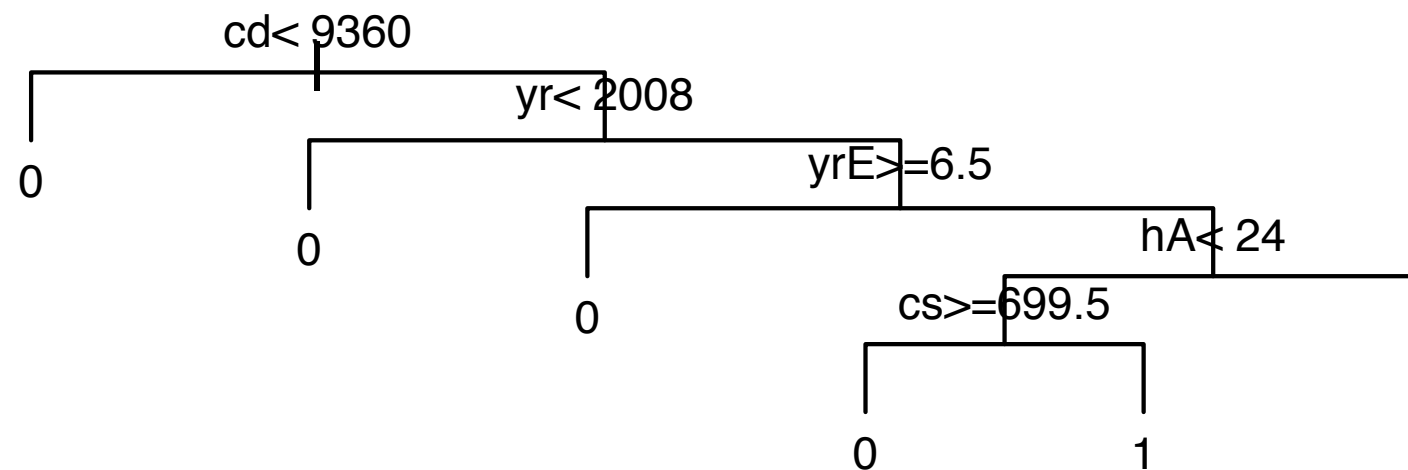
```
control <- list(minsplit = 20, cp = 0.01, xval = 2, maxdepth = 5, maxcompete = 0, maxsurrogate = 0,  
  usesurrogate = 2, surrogatestyle = 0)  
treeC <- rxDTree(formula = defaultFactor ~ houseAge + year + creditScore + yearsEmploy +  
  ccDebt, data = trainTestFiles[[1]], control = control, maxNumBins = 15000)
```



Plot the Classification Tree

```
plot(rxAddInheritance(treeC), uniform = TRUE, margin = 0.15)  
text(rxAddInheritance(treeC), digits = 2, cex = 0.7)  
title(main = "Classification Tree for Mortgage Data")
```

Classification Tree for Mortgage Data





Predict by Classification Tree

```
rxPredict(treeC, data = trainTestFiles[[2]], outData = trainTestFiles[[2]], overwrite = TRUE,  
  predVarNames = "Pred_C", type = "vector")  
rxGetInfo(trainTestFiles[[2]], getVarInfo = TRUE, varsToKeep = c("defaultFactor", "Pred_C"),  
  numRows = 5)
```

```
## File name: E:\GitHub\Revolution_Course_Materials\modules\BigData\DataCamp_Analysis\doc\mortDefault2.TrainTest.Train.xdf  
## Number of observations: 79759  
## Number of variables: 13  
## Number of blocks: 10  
## Compression type: zlib  
## Variable information:  
## Var 1: defaultFactor  
##      2 factor levels: 0 1  
## Var 2: Pred_C, Type: numeric, Low/High: (1.0000, 2.0000)  
## Data (5 rows starting with row 1):  
##   defaultFactor Pred_C  
## 1             0      1  
## 2             0      1  
## 3             0      1  
## 4             0      1  
## 5             0      1
```



Relabel Predictions

```
rxFactors(inData = trainTestFiles[[2]], outFile = trainTestFiles[[2]], overwrite = TRUE,  
  factorInfo = list(Pred_CF = list(newLevels = c("0", "1"), levels = c("1", "2"), varName = "Pred_C")))
```



View the Relabeled Predictions

```
rxGetInfo(trainTestFiles[[2]], getVarInfo = TRUE, varsToKeep = c("defaultFactor", "Pred_C",  
  "Pred_CF"), numRows = 5)
```

```
## File name: E:\GitHub\Revolution_Course_Materials\modules\BigData\DataCamp_Analysis\doc\mortDefault2.TrainTest.Train.xdf  
## Number of observations: 79759  
## Number of variables: 13  
## Number of blocks: 10  
## Compression type: zlib  
## Variable information:  
## Var 1: defaultFactor  
##           2 factor levels: 0 1  
## Var 2: Pred_C, Type: numeric, Low/High: (1.0000, 2.0000)  
## Var 3: Pred_CF  
##           2 factor levels: 0 1  
## Data (5 rows starting with row 1):  
##   defaultFactor Pred_C Pred_CF  
## 1              0      1      0  
## 2              0      1      0  
## 3              0      1      0  
## 4              0      1      0  
## 5              0      1      0
```



Confusion Matrix

```
conf.mat <- rxCrossTabs(~defaultFactor:Pred_CF, data = trainTestFiles[[2]])
conf.mat$counts[[1]]
```

```
##              Pred_CF
## defaultFactor    0    1
##              0 79316 70
##              1   330 43
```

```
prop.table(conf.mat$counts[[1]])
```

```
##              Pred_CF
## defaultFactor          0          1
##              0 0.994445768 0.0008776439
##              1 0.004137464 0.0005391241
```

Thank you

Revolution Analytics is the leading commercial provider of software and support for the popular open source R statistics language.

www.revolutionanalytics.com

1.855.GET.REVO

Twitter: @RevolutionR

