Managing Data

January 25, 2016

BTW

Examples of bad visualizations: `http://viz.wtf/`

Examples of spurious correlations:
`http://www.tylervigen.com/spurious-correlations`

Package `plyr`

- Simplifies common split-apply-combine tasks.
- Also provides useful utility functions for working with data frames.
- Main function is `??ply` where the `?` are replaced by one of `d, a, m, l, _` according to the type of input and output, data frame, array, matrix, list, or nothing respectively.
- Helper functions include: `join`, `mutate`, `summarize`, `arrange`, `count`.

Missing Values

- Why are data missing?
  - Missing completely at random (MCAR)
  - Missing at random (MAR)
  - Missing not at random (MNAR)
- How to deal with missing values?
  - Delete records - okay in a large dataset with few missing values
  - Impute values
- Also check for incorrect values

Missing Values

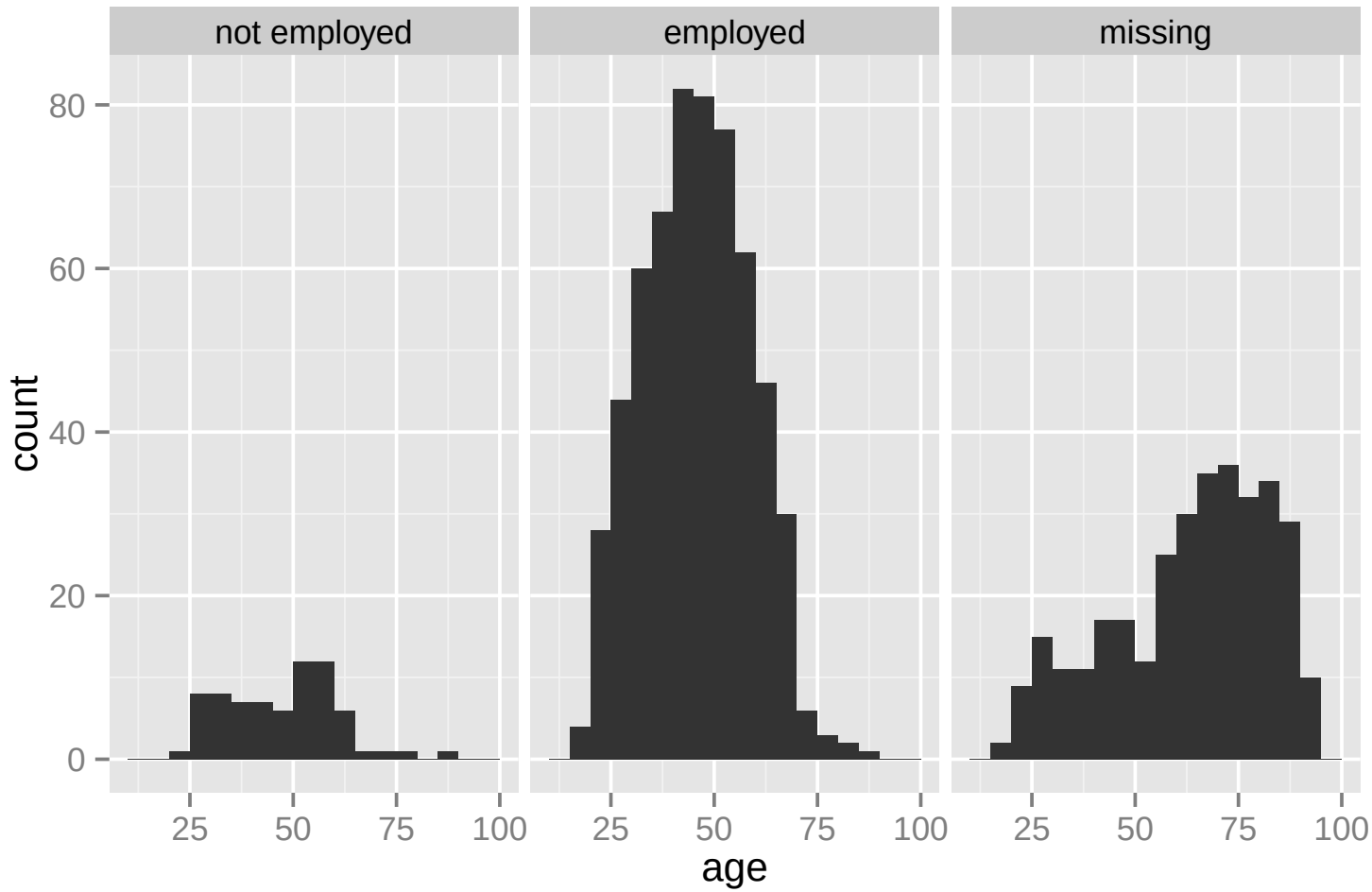
- In a categorical variable we may convert the missing values into additional level.

```
custdata <- mutate(custdata,
    is.employed.fix = factor(
        ifelse( is.na(is.employed), 2, is.employed),
        labels=c("not employed", "employed", "missing")
    )
)
```

- Many functions silently drop the entire rows that have missing values. By converting `NA` to another level we preserve the information that the value was missing.
- Why are the data missing? Maybe these are people who are not in the active workforce - students or retired?

Missing Values

```
qplot(age, binwidth=5,
        data=subset(custdata, age>0 & age < 100)) +
    facet_wrap(~is.employed.fix)
```



Missing Values

- In a numerical variable missing values may be imputed.
- It is important to know why the values are missing. This may suggest a default value.
- It is also important if values are missing randomly or systematically.
  - If missing at random, we may replace missing values with the mean.
  - Or we may use relations with other variables to impute.

Missing Values

Suppose some of the income values are missing

```
custdata <- mutate(custdata,
    Income=ifelse(is.na(is.employed) | income < 0,
                  NA, income))
summary(custdata[c('income', 'Income')])
```

Replace missing values with mean

```
custdata <- mutate(custdata,
    Income.fix=ifelse(is.na(Income),
        mean(Income, na.rm=TRUE), Income))
summary(custdata[c("Income", "Income.fix")])
```

Missing Values

Income may be different in different states.

```
custdata <- ddply(custdata, "state.of.res", mutate,
    Income.fix2=ifelse(is.na(Income),
        mean(Income, na.rm=TRUE), Income))
summary(custdata[c('Income', "Income.fix2")])
```

- If data are not missing randomly, we can't use a statistical technique to impute values.
- May replace missing values with some default value
- Or convert continuous variable to categorical and add a special category for missing

Missing Values

Replace missing `Income` with `0`

```
custdata <- mutate(custdata,
    Income.fix3=ifelse(is.na(Income), 0, Income))
```

Convert to categorical and add a level for missing

```
breaks <- c(0, 10000, 50000, 100000, 250000, 1000000)
tmp <- cut(custdata$Income, breaks=breaks,
        include.lowest=TRUE)
levels(tmp) <- c(levels(tmp), "missing")
tmp[ is.na(tmp) ] <- "missing"
custdata <- mutate(custdata, Income.fix4=tmp)
rm(tmp)
```

Data Transformations

Normalization removes location, scale, or both from data. It's useful when relative quantities are more meaningful than absolute ones.

```
custdata <- mutate(custdata,
                   age.normalized=(age-mean(age))/sd(age))

summary(custdata[c("age", "age.normalized")])
      age         age.normalized
 Min.   :  0.0   Min.   :-2.74074
 1st Qu.: 38.0   1st Qu.:-0.72626
 Median : 50.0   Median :-0.09011
 Mean   : 51.7   Mean   : 0.00000
 3rd Qu.: 64.0   3rd Qu.: 0.65207
 Max.   :146.7   Max.   : 5.03516
```

Data Transformations

- For *age* difference is more meaningful than ratio and the distribution is close to symmetric
- For *income* ratio is more meaningful

```
custdata <- mutate(custdata,
    income.normalized=income/median(income, na.rm=TRUE)*100)

summary(custdata[c("income", "income.normalized")])
     income        income.normalized
 Min.   : -8700   Min.   : -24.86
 1st Qu.: 14600   1st Qu.:  41.71
 Median : 35000   Median : 100.00
 Mean   : 53505   Mean   : 152.87
 3rd Qu.: 67000   3rd Qu.: 191.43
 Max.   :615000   Max.   :1757.14
```

Data Transformations

- Normalize income in each state

```
custdata <- ddply(custdata, .(state.of.res), mutate,
    income.nrml.by.state=
    income/median(income, na.rm=TRUE)*100)

summary(custdata[c("income.normalized","income.nrml.by.state")])
 income.normalized income.nrml.by.state
 Min.   : -24.86   Min.   : -19.33
 1st Qu.:  41.71   1st Qu.:  44.79
 Median : 100.00   Median : 100.00
 Mean   : 152.87   Mean   : 157.37
 3rd Qu.: 191.43   3rd Qu.: 189.87
 Max.   :1757.14   Max.   :1708.33
                   NA's   :1
```

Sampling

- When building a prediction model the data should be split into *training* and *testing* sets. Testing set is usually 10% of the total, but varies.

- If the total dataset is large, we may select a random sample for the initial exploration and prototyping models. When model is built, we train it on the full training set.

Sample 10%

Method one:

```
test.idx <- sample.int(nrow(custdata), 0.1*nrow(custdata))
train.idx <- setdiff(1:nrow(custdata), test.idx)
train <- custdata[train.idx,]
test <- custdata[test.idx,]
```

Method two:

```
custdata <- mutate(custdata, sample=runif(nrow(custdata)))
test <- subset(custdata, sample < 0.1)
train <- subset(custdata, sample >=0.1)
```

- adding a sample column makes sure the sampling is replicable.
- save the dataframe with the sample column to preserve it between sessions.

Choosing Models

- Classification problems: the target variable is categorical.
  - Most common is the case of binary or binomial classification, e.g. "true or false".
  - Multinomial classification can be solved as series of binomial classifications, e.g. "is it this category or not".
  - Typical algorithms: nearest neighbours, naive Bayes, classification trees, support vector machines, neural networks.

Choosing Models

- Scoring problems: the target variable is numerical.
  - Related to classification by way of predicting the probability an item belongs to a class
  - Typical algorithms: various types of regressions, function approximations.
- Scoring and classification are supervised learning because there is a target variable which is known in the training set.

Choosing Models

- Unsupervised learning: there is no target variable.
- Cluster analysis: identify groups of items that are similar within the cluster, but dissimilar across clusters
  - Clusters are not classes: classes are well defined while clusters are to be discovered, if any.
  - Typical algorithms: K-means and its variations, combinatorial algorithms.
- Association rules analysis: identify joint values of all variables that appear most frequently
  - When variables are binary it is also called market basket analysis.
  - Typical algorithm: Apriori algorithm.
- Principal Component Analysis (PCA), Independent Component Analysis (ICA)

Evaluating Models

- Null model: a model that is independent of the predictors, e.g. constant or random predictions.

  - This is the absolute minimum. Any meaningful model should perform better.

- Bayes rate model: the best possible model given the data.

  - If our model is getting close to this one, it's time to stop.

- Single-variable model: the best model where the prediction depends on only one explanatory variable.

  - Reasonable baseline model. There is no point building a complex model if it doesn't outperform the single-variable model.

Evaluating Models

```
> summary(custdata$health.ins)
   Mode   FALSE    TRUE    NA's
logical     159     841        0

> prediction <- rep(TRUE, nrow(custdata))
> prediction <- factor(prediction, levels=c(FALSE, TRUE))
> conf.table <- table(custdata$health.ins, prediction)
> conf.table
       prediction
        FALSE TRUE
  FALSE     0  159
  TRUE      0  841
```

This is the confusion table of a Null model.