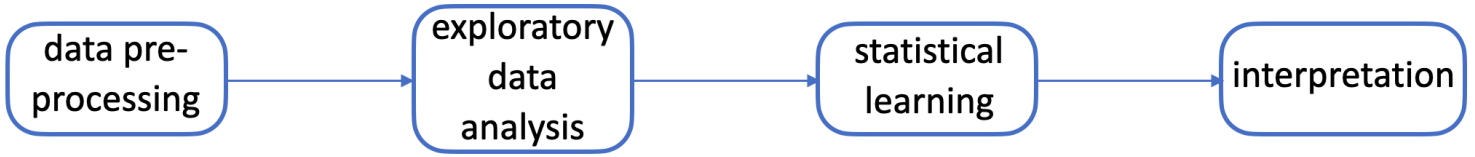
Data Analysis Workflow Review

36-600

The Data Analysis Workflow



1. *Data Pre-Processing*: wrangling data into tabular form; determination of informative variables; determination of variable types (factor v. quantitative); coding data input; determination of amount of missing data (and what to do about them: remove rows? remove columns? impute?)

2. *Exploratory Data Analysis*: for instance, histogram quantitative predictors, tabulate factor predictors, visualize the response variable versus each predictor to look for apparent associations; determine of transformations (e.g., log or sqrt); determination of outliers (by eye, and not by IQR-based rules)

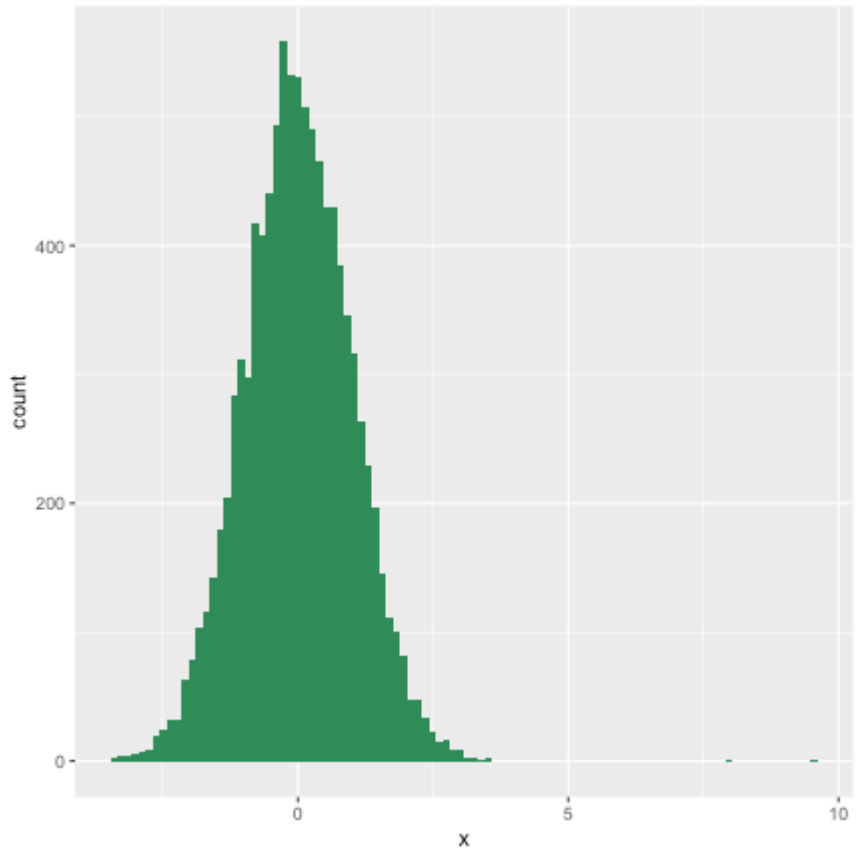
3. *Statistical Learning*: determination of goal: inference or prediction; determination of response variable type and thus the relevant statistical models

4. *Interpretation*: this one's on you

Missing Values

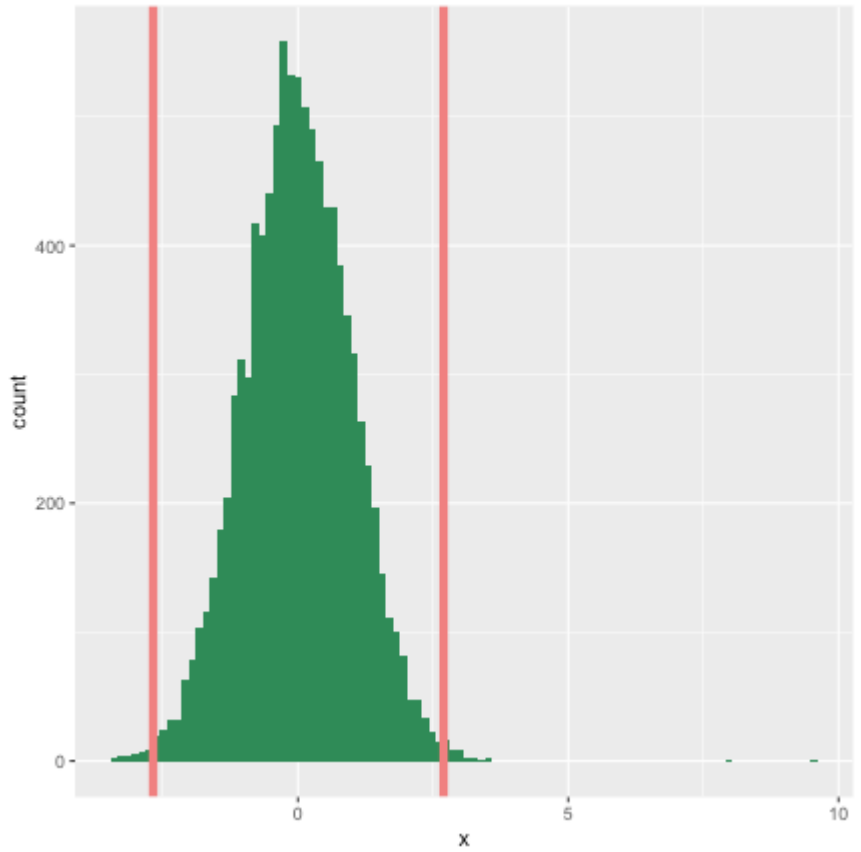
- In a data frame, values may sometimes be missing (labeled as `NA` in `R`)
- There are no heuristics, per se, for dealing with missing data prior to learning models
- The three options are:
  - *remove all rows with missing data*: this has the effect of increasing prediction uncertainty (because the sample size is reduced)
  - *remove all columns with missing data*: the effect of doing this is *a priori* unknown; if a removed column contains informative data, the effect could be severe, and if a removed column contains non-informative data, the effect could be minimal
  - (not covered in this class) *explore data imputation*: replace the `NA`s with estimated data...for instance, see the *MICE algorithm*

Anomalous Values



- What is obvious in the plot is that the histogram is "shifted to the left," meaning there are one or more anomalous values to the right (including one at the right boundary)

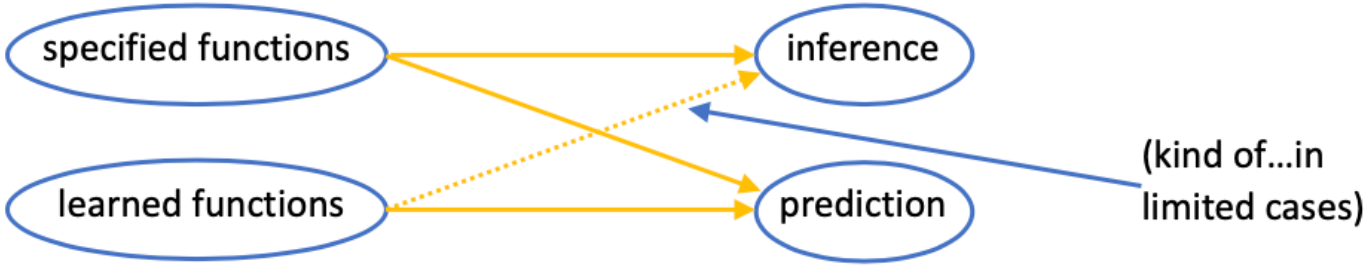
Anomalous Values



- I put outliers at 8 and 9.5; all other data are generated by the normal distribution
- The IQR method would chop off 71 data here: it is too conservative, and works less and less well the more and more data we have
- Here, I would just filter out any data > 4
- Methodologies to explore (that work in multiple dimensions):
  - *Mahalanobis distance* (anomalous data have excessively high distance values)
  - *isolation forest* (anomalous data have excessively high forest metrics)

Inference v. Prediction

- A decision that you must make prior to analyzing data is whether you want to pursue inferential or predictive analyses



- If your goal is inference, then
  - you will want to limit yourself to more inferential models (no SVM, for instance); and
  - you will want to determine if multicollinearity is an issue (and if it is, at least consider utilizing PC regression)
- If your goal is prediction, then you just want to determine which statistical model is the best representation of the data-generating process, full stop

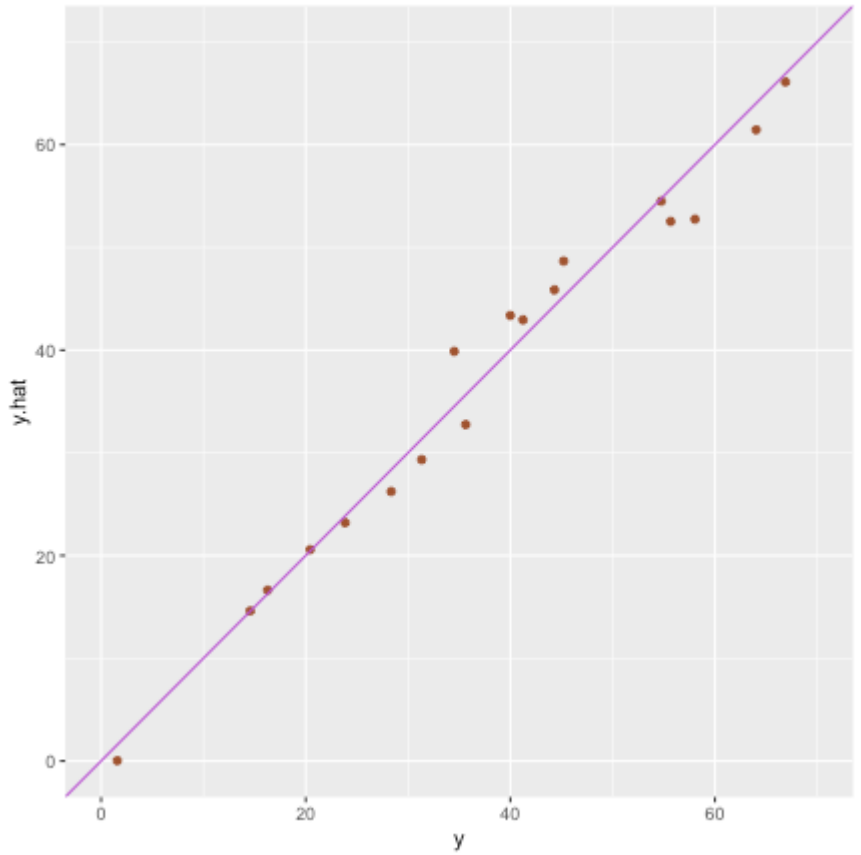
Multicollinearity

```
set.seed(101) ; x1 <- runif(20,min=0,max=20) ; x2 <-
y  <- x1 + x2 + rnorm(20,sd=3) ; summary(lm(y~x1+x2)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.3591 -1.8561  0.1124  2.0286  5.3134
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.9983     1.3899  -2.157   0.0448 *
## x1            3.9273     0.1219  32.204   <2e-16 ***
## x2                NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.762 on 18 degrees of freedom
## Multiple R-squared:  0.9829,    Adjusted R-squared:  0.982
## F-statistic:  1037 on 1 and 18 DF,  p-value: < 2.2e-16
```

- Definition: one or more of your predictor variables are linearly related to other predictor variables
- By construction, $\beta_0 = 0$, $\beta_1 = 1$, and $\beta_2 = 1$, but the estimates do not reflect this reality

Multicollinearity



- However, the presence of multicollinearity does *not* affect the model's predictive ability

Assessing Multicollineary

```
suppressMessages(library(car))
set.seed(101)
x1 <- runif(20,min=0,max=20)
x2 <- 3*x1 - 3 + rnorm(20,sd=0.5)
y  <- x1 + x2 + rnorm(20,sd=3)
vif(lm(y~x1+x2))
```

```
##       x1       x2
## 1201.598 1201.598
```

- We use `vif()` from the `car` package to assess multicollinearity
  - you have issues if any values are greater than 5 or 10
  - note: `vif()` breaks down if the relationships are deterministic
- Mitigation:
  - remove variables with the highest VIF score one at a time until all scores are < 5 or 10
  - rotate the coordinate system using PCA in order to remove multicollinearity
- Issues:
  - removing variables can negatively impact predictive ability
  - rotation via PCA makes the inferential story harder to tell

Variable Selection Methods

- What?
  - utilizing, e.g., `bestglm()` (in either best-subset, forward, or backward selection mode) along with an informative metric (e.g., AIC v. BIC), to select the most informative subset of predictor variables
  - note: another oft-used penalized regression model, the *lasso*, was not covered in this class
- Why?
  - uninformative predictor variables affect inference (clearly)
  - uninformative predictor variables *can* negatively impact predictive ability
- You should always include a variable selection method in your model suite, even if your analysis goal is prediction and not inference, because perhaps it will be the best overall model (in terms of, e.g., mean-squared error)

Statistical Learning

- A good suite of models to explore, in order from least flexible/most inferential to most flexible/least inferential would be
    - regression (quantitative response): linear regression, bestglm, regression tree, random forest, extreme gradient boosting, K-nearest neighbors (large amount of data), deep learning (huge amount of data)...pick the model with the lowest test-set mean-squared error
    - classification (factor response, two outcomes): logistic regression, bestglm, classification tree, random forest, extreme gradient boosting, K-nearest neighbors (large amount of data), support vector machines (small amount of data), deep learning (huge amounts of data)...pick the model with...well...this is more complicated
- You said "test set"
    - to avoid overfitting and thus to make your models more generalizable (i.e., to make them work better with new data), you want to split your data into training and test datasets (or better yet use K-fold cross-validation with 5 or 10 folds)
    - you learn models using the training set, assess their performance using the test set.
- **NOTE**: the *same* training and test sets should be used when learning all models!

Statistical Learning: Output

- Any statistical learning exercise should display the following output:
  - the summary for `lm()` or `glm()`, which shows the estimated coefficients and the results of hypothesis tests (and for linear regression, highlight the adjusted R^2 as a metric of the viability of the linear model)
  - the list of predictor variables retained by `bestglm()`, and the summary of the output for the best model
  - a plotted decision tree, if you learn a tree
  - variable importance plots, if you run `randomForest()` and `xgboost()`
  - for regression: a plot of the predicted test-set response (along the y-axis) versus the observed test-set response (along the x-axis)
  - for classification: the final adopted metric (e.g., misclassification rate) and details on how that was computed (e.g., the classification threshold)
  - a table of results for all models considered

Classification

- In classification settings, statistical models do not output a predicted class, but rather an estimate that a datum belongs to, e.g., Class 1
- If the two classes are balanced (i.e., equally represented in the data), then

  ```
  resp.class <- ifelse(class1.prob > 0.5,"CLASS_1","CLASS_0")
  ```

  is sufficient to map probabilities to classes...otherwise, you might have to change the "0.5" to a smaller value (if Class 0 dominates) or a larger one (if Class 1 dominates)
- As far as how to judge a classifier, there is...
  - the *misclassification rate*: this can be suboptimal for unbalanced classes, as the classifier will work to get the dominant class "right" at the expense of the weaker class (also: is there an equal cost to both kinds of misclassification...one might be "worse" than the other)
  - *Youden's J statistic*: this attempts to find the threshold where the sum of correct classifications for both classes is maximized
  - *area-under-curve (AUC)*: this metric basically says which classifier is *generally* best regardless of the actual threshold applied
- I personally would compute AUCs for all competing models, select the one with the largest AUC value, determine an optimal threshold for that model using Youden's J statistic, use that threshold to assign the test-set data to predicted classes, and output the misclassification rate (and/or other confusion matrix metrics) given those classes
  - as you gain experience, your mileage may begin to vary