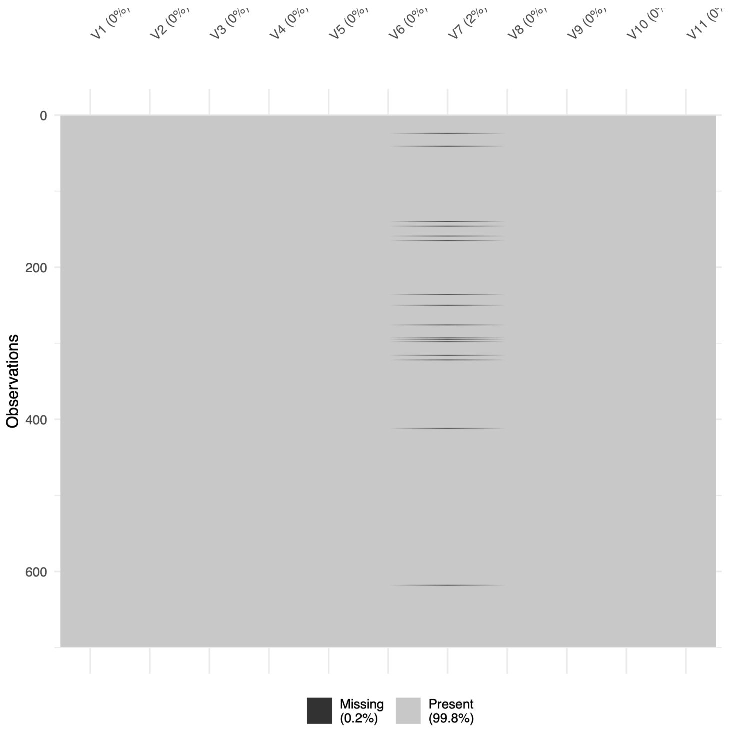
Question 14.1

The data was visualized and the extent of the missing data was evaluated.

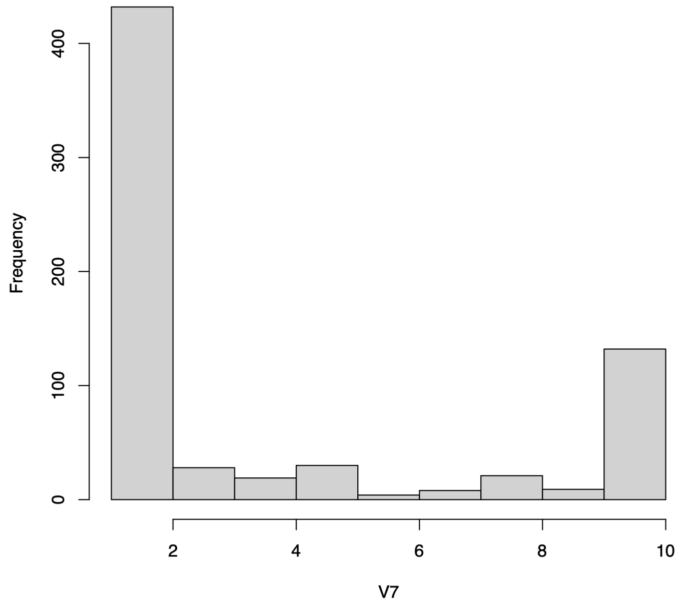
- The data was loaded
- vis_mis() (from visdat library) quickly showed where and how much data is missing



  - Results showed rows from column 7 (V7) were missing
- The indices corresponding to missing columns were collected. This result confirmed column 7 is missing data, which aligns with what was visualized.
  - sapply() applied anyNA (a test for whether there are any “NA” values) to each column, returning a logical vector of TRUE/FALSE values corresponding to the columns in the df
  - which() returned the indices of TRUE values
- The number of empty rows within column 7 was evaluated. This result confirmed 16 missing rows in column 7, which also aligns with what was visualized.
  - is.na() tested whether the value of the row was NA, sum() was used to get the number of times is.na() returned true
- The dataset was evaluated for suitability for imputation. Approximately 2.3% of the factor given by column 7 is missing, which is within the 5%-per-factor rule of thumb provided in lecture.
  - Calculation - number of empty rows calculated in previous step/ nrows(data)

Mean/mode imputation was conducted (Q 14.1 part 1)

- The distribution of the data in column 7 was checked to determine suitability of imputation by a measure of central tendency
  - hist() was used to generate a histogram of V7 values (excluding NA rows)

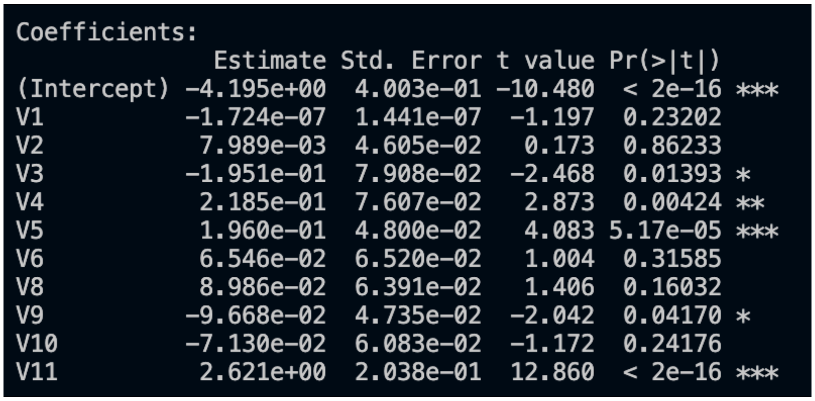


  - The distribution appears right-skewed and bimodal. A central tendency measure is a poor method of imputation here because neither the median nor the mean represents a common value.
    - Imputation by mode is not applicable (this method is for categorical columns; V7 is numeric, so only imputation by median or mode are applicable)
- The mean value of V7 was calculated using colMeans(); na.rm was set to TRUE to ignore NA values; result: ~3.54
- Each NA value in column 7 was overwritten with the mean value calculated in the previous step. This gave the table whose missing values were imputed by mean/mode imputation.
  - It was verified that after this process took place no NAs remained
    - sum() was used to get total count is.na() returned true in the version of the data frame where mean imputation was done

Regression-based imputation was conducted (Q 14.1 part 2)

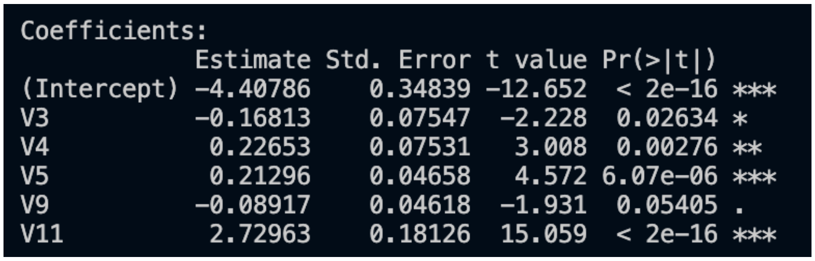
- 75% of the data was allocated for training. The remaining 25% was allocated for testing.
  - A data subset was created by removing all rows with NAs from the data
  - sample() was used to randomly select indices from this data subset so that rows with NA would not be allocated for training or testing.
  - Selected indices were assigned as training data and the remainder were set as test data.
- A model was built based on which factors returned significant in a p-value test
  - A linear model was created to determine significant factors. V7 was modeled as the response, a function of each other attribute.
    - model <- lm(V7 ~ ., data = train_data, na.action = na.omit)
    - Na.action indicates how NAs should be handled. There should be none since the training and testing data subsets were set up from a version of the data with the NAs removed

        - Only a handful of coefficients returned significant ($p<0.05$): V3-V5, V9, and V11



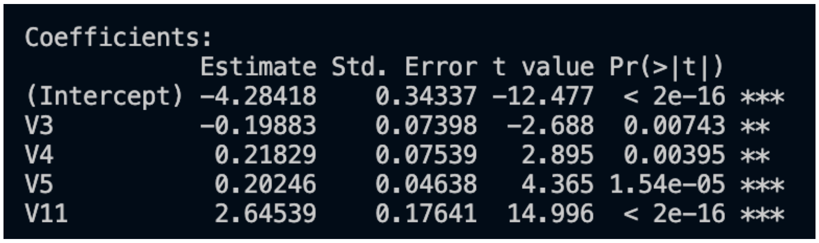
```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -4.195e+00  4.003e-01 -10.480  < 2e-16 ***
V1          -1.724e-07  1.441e-07  -1.197  0.23202    
V2           7.989e-03  4.605e-02   0.173  0.86233    
V3          -1.951e-01  7.908e-02  -2.468  0.01393 *  
V4           2.185e-01  7.607e-02   2.873  0.00424 ** 
V5           1.960e-01  4.800e-02   4.083 5.17e-05 ***
V6           6.546e-02  6.520e-02   1.004  0.31585    
V8           8.986e-02  6.391e-02   1.406  0.16032    
V9          -9.668e-02  4.735e-02  -2.042  0.04170 *  
V10         -7.130e-02  6.083e-02  -1.172  0.24176    
V11          2.621e+00  2.038e-01  12.860  < 2e-16 ***
```

    - A refined linear model was generated based on the coefficients returned as significant from the full model
        - model_2 <- lm (V7 ~ V3+ V4 + V5 + V9 + V11, data = train_data ,na.action = na.omit)



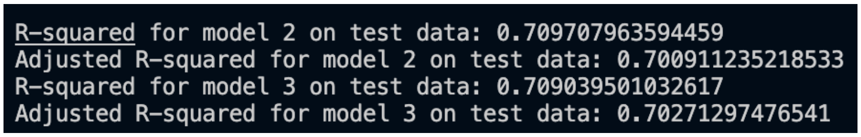
```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -4.40786    0.34839 -12.652  < 2e-16 ***
V3          -0.16813    0.07547  -2.228  0.02634 *  
V4           0.22653    0.07531   3.008  0.00276 ** 
V5           0.21296    0.04658   4.572 6.07e-06 ***
V9          -0.08917    0.04618  -1.931  0.05405 .  
V11          2.72963    0.18126  15.059  < 2e-16 ***
```

        - All return significant except V9, which is borderline
    - A refined linear model was generated without V9
        - model_3 <- lm (V7 ~ V3+ V4 + V5 + V11, data = train_data ,na.action = na.omit)



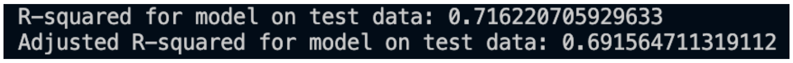
```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -4.28418    0.34337 -12.477  < 2e-16 ***
V3          -0.19883    0.07398  -2.688  0.00743 ** 
V4           0.21829    0.07539   2.895  0.00395 ** 
V5           0.20246    0.04638   4.365 1.54e-05 ***
V11          2.64539    0.17641  14.996  < 2e-16 ***
```

        - All returned significant
    - model_2 (includes borderline predictor V9) and model_3 (excludes borderline predictor V9) were compared. Both models were used to predict values in the training set using predict()
            - For both models, R^2 and adjusted R^2 were calculated and used to evaluate model performance
            - Performance was identical



```
R-squared for model 2 on test data: 0.709707963594459
Adjusted R-squared for model 2 on test data: 0.700911235218533
R-squared for model 3 on test data: 0.709039501032617
Adjusted R-squared for model 3 on test data: 0.70271297476541
```

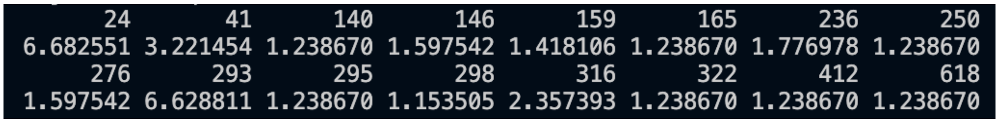
                - Just to see whether the factor selection made a difference at all, the same testing was done on the original model that used all variables. Performance when comparing adjusted R2 was marginally worse



```
R-squared for model on test data: 0.716220705929633
Adjusted R-squared for model on test data: 0.691564711319112
```

    - Model 3 (V7 ~ V3+ V4 + V5 + V11) was selected since performance was best. This marginally superior performance of model 3 may be due to chance. It is not clear whether model 3 is meaningfully better than model 2 or 1.
- Model 3 was selected and retrained on all the data, except for rows containing NA

    - Note that previously, it was trained on just 75% of the data
- Model 3 was used to make predictions about the 16 “NA” values in column 7. Again, predict() was used.



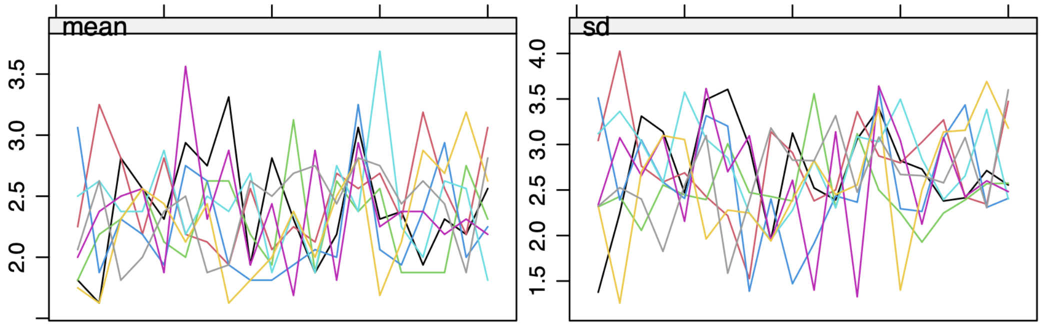
```
      24       41      140      146      159      165      236      250
6.682551 3.221454 1.238670 1.597542 1.418106 1.238670 1.776978 1.238670
     276      293      295      298      316      322      412      618
1.597542 6.628811 1.238670 1.153505 2.357393 1.238670 1.238670 1.238670
```

- These predictions were used to overwrite the NAs in V7 to generate a table whose missing values were imputed by regression.
    - It was verified that after this process took place no NAs remained
        - sum() was used to get total count is.na() returned true in the version of the data frame where mean imputation was done

Imputation by regression with perturbation (Q 14.1 part 3)

- The mice (Multivariate Imputation by Chained Equations) library was used to perform regression with perturbation
- The hyperparameters were selected.
    - Hyperparameter “m” gives the number of multiple imputations (how many dataset versions we get).
        - Since imputation is stochastic, each dataset will be slightly different. As m -> ∞, the results approach those given by regression without imputation. On the other hand, increasing m reduces Monte Carlo error in pooled estimates.
    - Hyperparameter maxit is number of iterations to run the imputation process. The default is 5. To determine whether maxit iterations were sufficient for stable imputations, a trace plot was generated for multiple values of maxit.
        - A trace visualizes convergence of the imputation process. The value of maxit was selected based on the point where the trace plots showed overlapping chains and no overall trend.
            - overlapping chains without drift indicate distribution of imputed means/SDs stops changing with iteration
            - plot() was used to generate the plot

        - maxit value of 10 satisfied this expectation



    - The 'method' argument dictates the statistical technique mice uses to predict missing values.
        - PMM means Predictive Mean Matching; it’s a non-parametric approach particularly suited for continuous data. It operates by finding observed values with similar predictive characteristics to the missing entries. The missing values are then imputed, thus preserving the distribution and variance of the original data more.

- Imputed values with perturbation were given by mice(data, m = 8, maxit = 10, method = 'pmm')
- complete() was used to extract a single completed dataset from the multiple imputations created by mice

Sources:
https://libguides.princeton.edu/R-Missingdata
https://www.rdocumentation.org/packages/mice/versions/3.17.0/topics/mice

Question 15.1
Describe a situation or problem from your job, everyday life, current events, etc., for which optimization would be appropriate. What data would you need?

Optimization could be useful in a manufacturing scenario. Suppose a factory produces both basic and premium water bottles. One machine in the production line is used in the last step for the production of both. Base bottles and premium bottles take different minutes on that machine and yield different revenue per unit. The company must determine how many of each to make to maximize profit without exceeding the machine's minutes.

Variables (units per day)

- $X_b \geq 0$
  - The number of basic bottles produced is 0 or more
- $X_p \geq 0$
  - The number of premium bottles is also at least 0

Parameters needed

- Pb, Pp: profit per unit (for each bottle type)
- Tb, Tp: time (minutes) each unit takes on the machine (for each bottle type)
- T: the total time (minutes) machine operates

Objective is to maximize profit:
Maximize $Pb*Xb + Pp*Xp$

Constraint - there is only one machine, with limited time, to work on both bottle types, handling one bottle at a time
$Tb*Xb + Tp*Xp \leq T$