Combining Variables into Financial Metrics

One strategy we attempted was to transform the variables in our dataset into the ratios that RapidRadings uses to construct their financial health rating. The goal of this method is to get a set of covariates that explain FHR better than our original data does.

Creating Variables Needed for Ratios:

Some variables not explicitly defined in our dataset are needed to form the financial raitos.

```
df_old = df

# operating profitability
df$operating_profit = df$total_operating_revenue - df$total_operating_expense
df$capital_employed <- df$total_assets - df$total_current_liabilities
df$total_revenue <- df$sales_revenue + df$other_income + df$investment_income + df$extraordinary_income
df$operating_profit_before_depreciation <- df$operating_profit + df$depreciation + df$amortization

# net profitability
df$operating_profit_after_tax <- df$operating_profit - df$company_tax_expense

# cost structure efficiency
df$total_cash_operating_expenditure <- df$cost_of_goods_sold + df$total_operating_expense +
                                       df$total_staff_costs + df$other_operating_expense

# capital structure efficiency

# other ratios
df$quick_assets <- df$total_current_assets - df$total_inventories - df$prepayments
df$working_capital <- df$total_current_assets - df$total_current_liabilities

# Resilience
# leverage
df$total_debt <- df$total_current_liabilities + df$total_term_liabilities

# liquidity
df$cash <- df$bank_cash_balances + df$net_change_in_cash_and_cash_equivalents
df$cash_from_operations <- df$net_profit_after_tax + df$depreciation + df$amortization

# earnings performance
df$current_debt_service <- df$interest_expense + df$total_current_liabilities
```

Ratios:

```
# operating profitability
df$opProf_capEmp = df$operating_profit / df$capital_employed
df$opProf_shareEq = df$operating_profit / df$total_shareholder_equity
df$opProf_totRev = df$operating_profit / df$total_revenue
df$opProf_totAssets = df$operating_profit / df$total_assets
df$opProfB4Dep_capEmp = df$operating_profit_before_depreciation / df$capital_employed
df$opProfB4Dep_totRev = df$operating_profit_before_depreciation / df$total_revenue
```

```
df$opProfB4Dep_totAssets = df$operating_profit_before_depreciation / df$total_assets
df$opProfB4Dep_shareEq = df$operating_profit_before_depreciation / df$total_shareholder_equity
df$grProf_capEmp = df$gross_profit / df$capital_employed
df$grProf_totRev = df$gross_profit / df$total_revenue
df$grProf_totAssets = df$gross_profit / df$total_assets

# net profitability
df$opProfAftTax_capEmp = df$operating_profit_after_tax / df$capital_employed
df$opProfAftTax_shareEq = df$operating_profit_after_tax / df$total_shareholder_equity
df$opProfAftTax_totRev = df$operating_profit_after_tax / df$total_revenue
df$opProfAftTax_totAssets = df$operating_profit_after_tax / df$total_assets
df$netProfAftTax_capEmp = df$net_profit_after_tax / df$capital_employed
df$netProfAftTax_shareEq = df$net_profit_after_tax / df$total_shareholder_equity
df$netProfAftTax_totRev = df$net_profit_after_tax / df$total_revenue
df$netProfAftTax_totAssets = df$net_profit_after_tax / df$total_assets
df$netProfB4Tax_capEmp = df$net_profit_before_tax / df$capital_employed
df$netProfB4Tax_shareEq = df$net_profit_before_tax / df$total_shareholder_equity
df$netProfB4Tax_totRev = df$net_profit_before_tax / df$total_revenue
df$netProfB4Tax_totAssets = df$net_profit_before_tax / df$total_assets

# cost structure efficiency
df$cogs_totRev = df$cost_of_goods_sold / df$total_revenue
df$cogs_totCashOpExp = df$cost_of_goods_sold / df$total_cash_operating_expenditure
df$dep_totRev = df$depreciation / df$total_revenue
df$dep_totCashOpExp = df$depreciation / df$total_cash_operating_expenditure
df$intExp_totRev = df$interest_expense / df$total_revenue
df$intExp_totCashOpExp = df$interest_expense / df$total_cash_operating_expenditure
df$intExp_staffCost = df$interest_expense / df$total_staff_costs
df$othOpExp_totRev = df$other_operating_expense / df$total_revenue
df$othOpExp_totCashOpExp = df$other_operating_expense / df$total_cash_operating_expenditure
df$staffCost_totRev = df$total_staff_costs / df$total_revenue
df$staffCost_totCashOpExp = df$total_staff_costs / df$total_cash_operating_expenditure
df$taxExp_totRev = df$company_tax_expense / df$total_revenue
df$taxExp_totCashOpExp = df$company_tax_expense / df$total_cash_operating_expenditure
df$opProf_intExp = df$operating_profit / df$interest_expense
df$intExp_totLiab = df$interest_expense / df$total_liabilities
df$totRev_inv = df$total_revenue / df$total_inventories
df$totRev_staffCost = df$total_revenue / df$total_staff_costs
df$opProfAftTax_intExp = df$operating_profit_after_tax / df$interest_expense

# capital structure efficiency
df$termLiab_capEmp = df$total_term_liabilities / df$capital_employed
df$termLiab_totRev = df$total_term_liabilities / df$total_revenue
df$shareEq_totAssets = df$total_shareholder_equity / df$total_assets
df$totLiab_totAssets = df$total_liabilities / df$total_assets
df$currAssets_totAssets = df$total_current_assets / df$total_assets
df$currAssets_totLiab = df$total_current_assets / df$total_liabilities
df$currLiab_totAssests = df$total_current_liabilities / df$total_assets
df$currLiab_totLiab = df$total_current_liabilities / df$total_liabilities
df$totRev_capEmp = df$total_revenue / df$capital_employed
df$totRev_shareEq = df$total_revenue / df$total_shareholder_equity
df$totRev_totAssets = df$total_revenue / df$total_assets
```

```
# other ratios
df$currAssets_currLiab = df$total_current_assets / df$total_current_liabilities
df$qAssets_capEmp = df$quick_assets / df$capital_employed
df$qAssets_currAssets = df$quick_assets / df$total_current_assets
df$qAssets_currLiab = df$quick_assets / df$total_current_liabilities
df$qAssets_totAssets = df$quick_assets / df$total_assets
df$qAssets_totRev = df$quick_assets / df$total_revenue
df$workCap_capEmp = df$working_capital / df$capital_employed
df$workCap_totRev = df$working_capital / df$total_revenue
df$workCap_totAssets = df$working_capital / df$total_assets
df$totRev_workCap = df$total_revenue / df$working_capital

# leverage
df$totDebt_totAssets = df$total_debt / df$total_assets

# liquidity
df$cash_currLiab = df$cash / df$total_current_liabilities
df$totCashOpExp_currLiab = df$total_cash_operating_expenditure / df$total_current_liabilities
df$workCap_totAssets = df$working_capital / df$total_assets

# earnings performance
df$totCashOpExp_currDebtServ = df$total_cash_operating_expenditure / df$current_debt_service
df$opProf_currDebtServ = df$operating_profit / df$current_debt_service
df$opProf_intExp = df$operating_profit / df$interest_expense
df$opProf_totAssets = df$operating_profit / df$total_assets
df$netProf_totAssets = df$net_profit_after_tax / df$total_assets
df$cogs_sales = df$cost_of_goods_sold / df$sales_revenue
df$retEarn_totAssets = df$retained_earnings / df$total_assets
```

```
ratios = df[, c(1,2,3,65,66,79:147)]

ratios[] <- lapply(ratios, function(x) {
  x[is.infinite(x) | is.nan(x)] = 0
  return(x)
})
```

Evaluating Data with Random Forests

We tested the effectiveness of using the transformed ratio variables with random forests that model a supplier's current FHR. This was before we began modeling future FHR with lag models.

```
rf = randomForest(fhr~.-vlookup_supplier_number - period - eqy_year - chs,
                  data = df_old, ntree = 500, mtry = 17)
rf
```

```
##
## Call:
##  randomForest(formula = fhr ~ . - vlookup_supplier_number - period -      eqy_year - chs, data = df_
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 17
##
```

```
##           Mean of squared residuals: 38.16555
##                     % Var explained: 91.59
```

```
rf_ratios = randomForest(fhr~.-vlookup_supplier_number - period - eqy_year - chs,
                         data = ratios, ntree = 500, mtry = 25, importance = T)
rf_ratios
```

```
##
## Call:
##  randomForest(formula = fhr ~ . - vlookup_supplier_number - period -      eqy_year - chs, data = rati
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 25
##
##           Mean of squared residuals: 44.16957
##                     % Var explained: 90.27
```

The testing mean squared error of the random forest is higher for the ratios dataset than it is for the original dataset. We will now take the metrics used in computing ratios as our predictors. These predictors are numerators and denominators of the RapidRatings Ratios, but they are not divided by each other.

```
metrics = df |> select(vlookup_supplier_number, eqy_year, period, chs, fhr,
operating_profit, operating_profit_before_depreciation, gross_profit, capital_employed,
total_shareholder_equity, total_assets, total_revenue, operating_profit_after_tax,
net_profit_after_tax, net_profit_before_tax, cost_of_goods_sold, depreciation,
interest_expense, other_operating_expense, total_staff_costs, company_tax_expense,
total_cash_operating_expenditure, total_liabilities, total_inventories,
total_term_liabilities, total_current_assets, total_current_liabilities, quick_assets,
working_capital, total_debt, cash, retained_earnings, current_debt_service,
cash_from_operations)
```

```
rf_metrics = randomForest(fhr~.-vlookup_supplier_number - period - eqy_year - chs,
                          data = metrics, ntree = 500, mtry = 10)
rf_metrics
```

```
##
## Call:
##  randomForest(formula = fhr ~ . - vlookup_supplier_number - period -      eqy_year - chs, data = meti
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 10
##
##           Mean of squared residuals: 47.47413
##                     % Var explained: 89.54
```

```
numeric_metrics = metrics[,6:34]

pca = prcomp(numeric_metrics, center = TRUE, scale. = TRUE)

metrics_pca = cbind(metrics[,1:5], pca$x)
```

```
rf_pc = randomForest(fhr ~ PC1+PC2+PC3+PC4+PC5, data = metrics_pca, mtry = 5)
rf_pc
```

```
##
## Call:
##  randomForest(formula = fhr ~ PC1 + PC2 + PC3 + PC4 + PC5, data = metrics_pca,      mtry = 5)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##           Mean of squared residuals: 73.15561
##                     % Var explained: 83.88
```

The random forest run on the "undivided" metrics performs worse than the random forest run on the "divided" ratios. The bagging tree model of the first 5 principal components of the "undivided" metrics performs even worse.

Using Cumulative Data

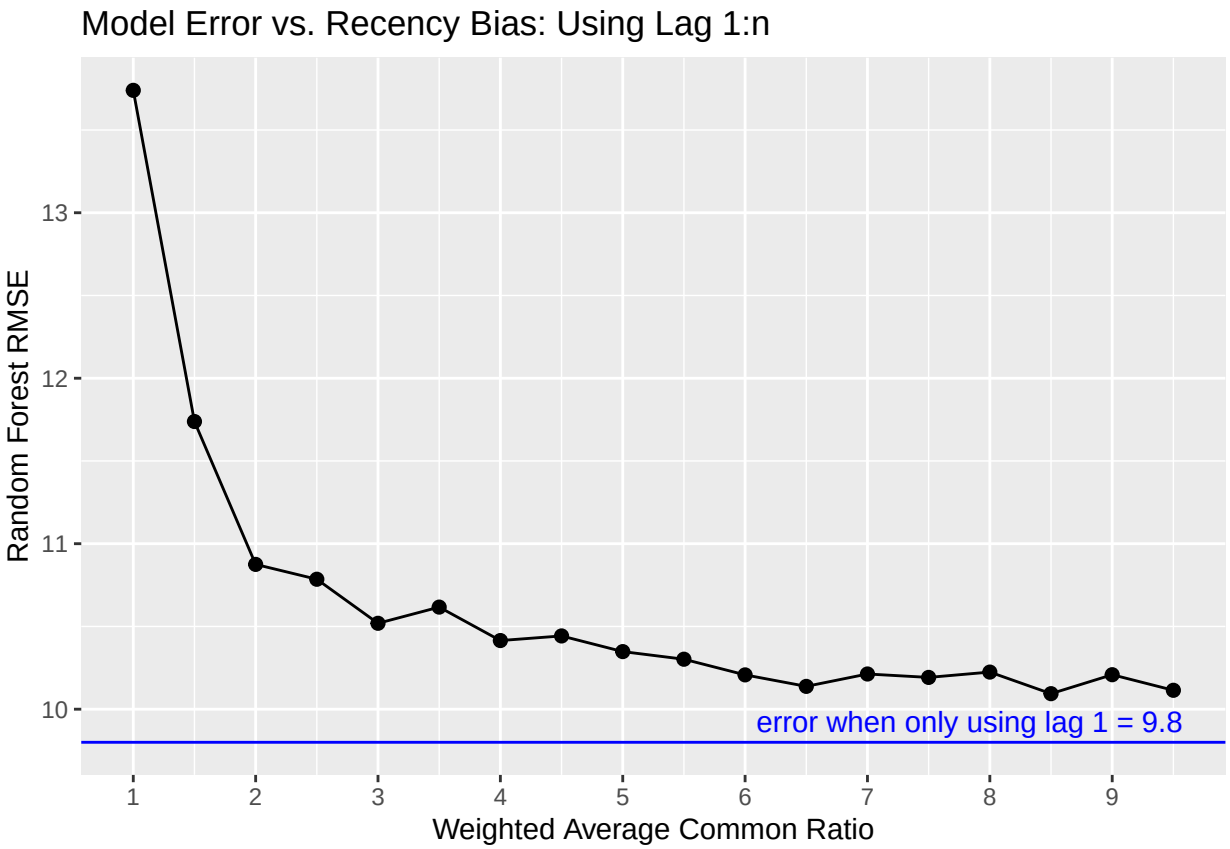
One strategy for improving the predictive power of our models for FHR is incorporating cumulative data. The idea is to all data outside of a given lag period, not just the most recent data. Cumulative data is incorporated by weighted averages. The method for creating weighted averages is a geometric series. The weighted average uses a common ratio, r. r corresponds to the ratio of an observation relative to the one before it. For example, an r of 1.5 means that data from 2024 Q3 has 1.5 times as much weight as data from 2024 Q2. Thus, an r of 1 computes the mean of the data. Several random forests are run on different values of r in order to test the impact of weighing recent data more and more relative to older data. The geometric weighting was chosen because it is easy to implement, especially given that not all suppliers have the same number of data entries. Weights can be divided by r until the oldest data entry is reached and then normalized.

Weighted avg- lag 1:n

```
div_vals = c(1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 8.5, 9, 9.5)
rmse_vals = c()

for (div in div_vals){

  df_weighted_avg = data.frame()

  for (s in suppliers){
    section = df[df$vlookup_supplier_number == s,]

    nrow = nrow(section)

    if (nrow > 1) {

      new_fhr = section[1, 5]

      new_fhr = new_fhr |> rename(new_fhr = fhr)

      w = 10
      w_sum = 0
```

```
      row = as.data.frame(matrix(0, nrow = 1, ncol = 63))
      for (i in 2:nrow) {

        row = (section[i, 4:66] * w) + row

        w_sum = w_sum + w

        w = w / div

      }

      row = row / w_sum

      new_row = cbind(new_fhr, row)

      df_weighted_avg = rbind(df_weighted_avg, new_row)
    }
  }

      rf_weighted_avg = randomForest(new_fhr~., data = df_weighted_avg)

      rmse = sqrt(rf_weighted_avg$mse[length(rf_weighted_avg$mse)])

      rmse_vals = c(rmse_vals, rmse)
}

df_div_rmse = data.frame(div_vals, rmse_vals)
```

```
df_div_rmse |> ggplot(aes(x=div_vals, y = rmse_vals)) + geom_point(size=2) + scale_x_continuous(breaks =
```



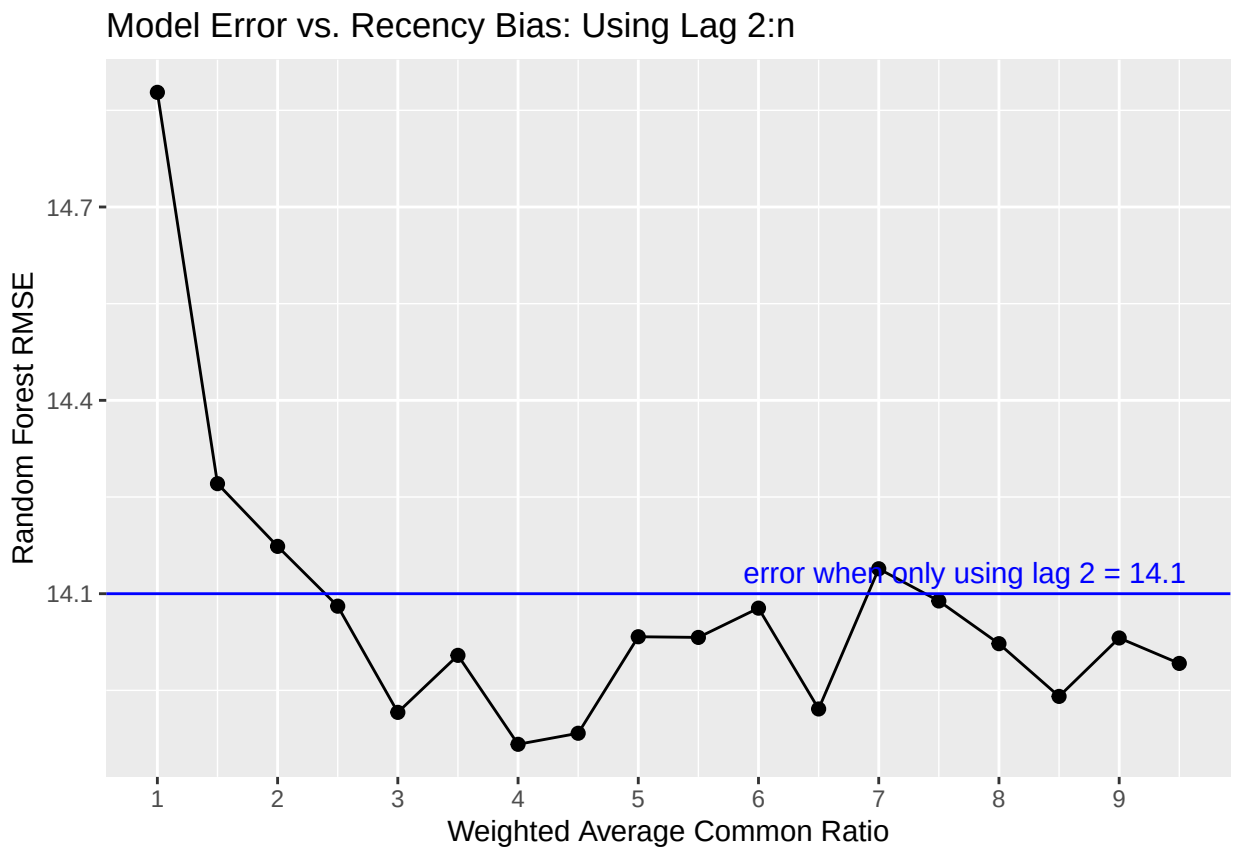
Using cumulative data is not helpful for our lag 1 models. As the weighted average common ratio increases, the random forest testing error decreases. More weight on recent data improves acccuracy.

weighted avg lag 2:n

```
rmse_vals2 = c()

for (div in div_vals){

  df_weighted_avg2 = data.frame()

  for (s in suppliers){
    section = df[df$vlookup_supplier_number == s,]

    nrow = nrow(section)

    if (nrow > 2) {

      new_fhr = section[1, 5]

      new_fhr = new_fhr |> rename(new_fhr = fhr)

      w = 10
      w_sum = 0

      row = as.data.frame(matrix(0, nrow = 1, ncol = 63))
      for (i in 3:nrow) {
```

```
        row = (section[i, 4:66] * w) + row

        w_sum = w_sum + w

        w = w / div

      }

      row = row / w_sum

      new_row = cbind(new_fhr, row)

      df_weighted_avg2 = rbind(df_weighted_avg2, new_row)
    }
  }

      rf_weighted_avg2 = randomForest(new_fhr~., data = df_weighted_avg2)

      rmse = sqrt(rf_weighted_avg2$mse[length(rf_weighted_avg2$mse)])

      rmse_vals2 = c(rmse_vals2, rmse)
}

df_div_rmse2 = data.frame(div_vals, rmse_vals2)
```

```
df_div_rmse2 |> ggplot(aes(x=div_vals, y = rmse_vals2)) + geom_point(size=2) + scale_x_continuous(breaks
```



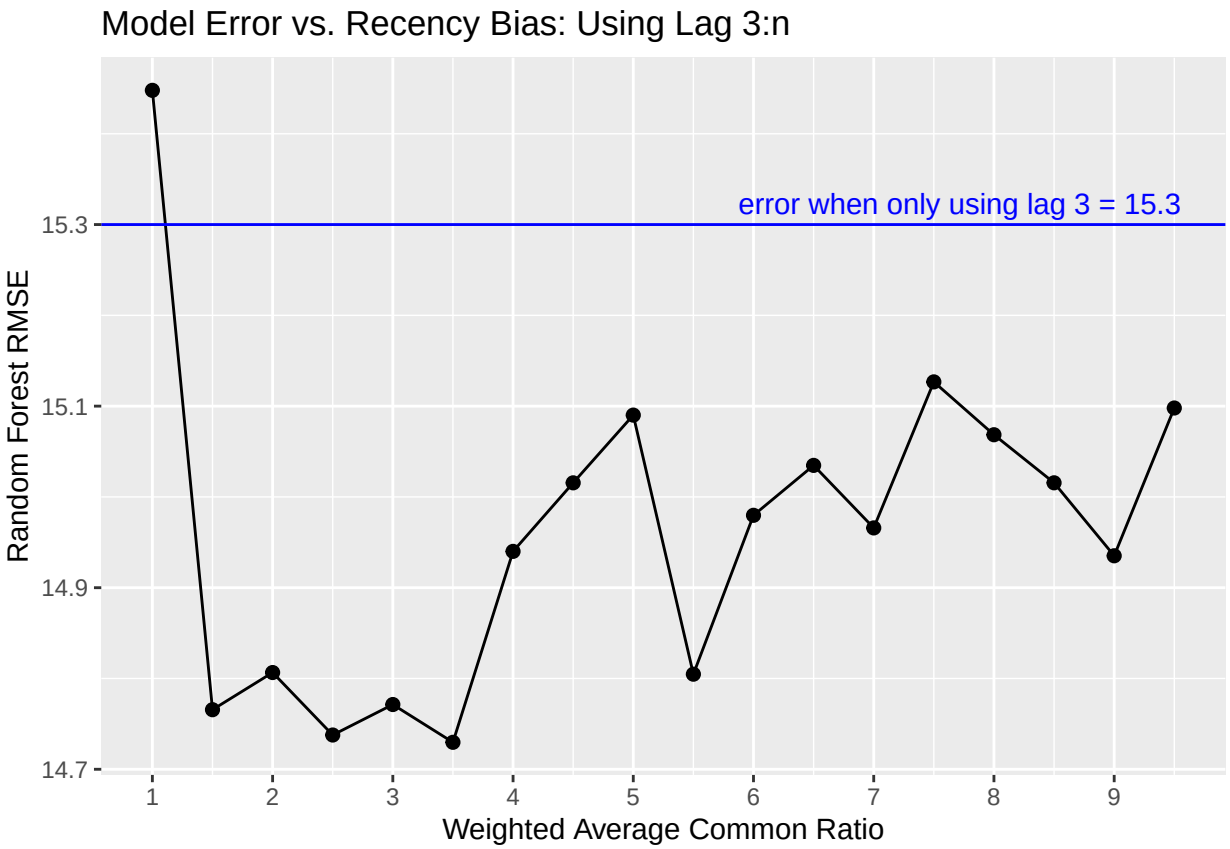
In lag 2:n, using cumulative data leads to slightly lower error than just lag 2. A reasonable way to make predictions about FHR 2 quarters into the future would be a model that weighs recent data 4 times as heavily as the data 1 quarter before it.

weighted avg lag 3:n

```
rmse_vals3 = c()

for (div in div_vals){

  df_weighted_avg3 = data.frame()

  for (s in suppliers){
    section = df[df$vlookup_supplier_number == s,]

    nrow = nrow(section)

    if (nrow > 3) {

      new_fhr = section[1, 5]

      new_fhr = new_fhr |> rename(new_fhr = fhr)

      w = 10
      w_sum = 0

      row = as.data.frame(matrix(0, nrow = 1, ncol = 63))
```

```
      for (i in 4:nrow) {

        row = (section[i, 4:66] * w) + row

        w_sum = w_sum + w

        w = w / div

      }

      row = row / w_sum

      new_row = cbind(new_fhr, row)

      df_weighted_avg3 = rbind(df_weighted_avg3, new_row)
    }
  }

      rf_weighted_avg3 = randomForest(new_fhr~., data = df_weighted_avg3)

      rmse = sqrt(rf_weighted_avg3$mse[length(rf_weighted_avg3$mse)])

      rmse_vals3 = c(rmse_vals3, rmse)
}

df_div_rmse3 = data.frame(div_vals, rmse_vals3)
```

```
df_div_rmse3 |> ggplot(aes(x=div_vals, y = rmse_vals3)) + geom_point(size=2) + scale_x_continuous(breaks
```



There is significant evidence that using cumulative data improves predictive power looking 3 quarters into the future. It would be appropriate to weigh data from a gien quarter 2.5 times as heavily as the data from the previous quarter when making such predictions.