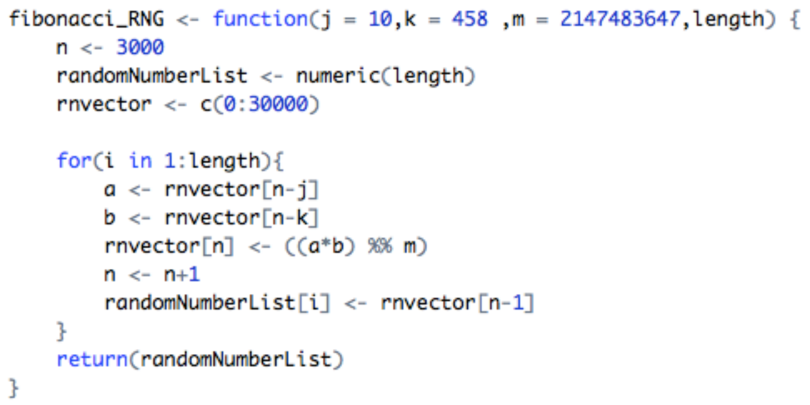
SMDE Assignment 2 – Kymry Burwell

*Please see r file for code. You should be able to simply hit run and see all of the output. I've also included the Excel file I used to run the Yates algorithm.

Step 0 – Create Random Number Generator

I decided to create a function in r that generates pseudo-random numbers using a *Lagged Multiplicative Fibonacci Generator*.



```
fibonacci_RNG <- function(j = 10,k = 458 ,m = 2147483647,length) {
    n <- 3000
    randomNumberList <- numeric(length)
    rnvector <- c(0:30000)

    for(i in 1:length){
        a <- rnvector[n-j]
        b <- rnvector[n-k]
        rnvector[n] <- ((a*b) %% m)
        n <- n+1
        randomNumberList[i] <- rnvector[n-1]
    }
    return(randomNumberList)
}
```

The function takes four inputs and outputs a list of random numbers of specified length. In order to test the generator, we first generate 100,000 random numbers. One of the good things about this type of generator is the large cycle size it provides.

In order to check that the generator does in fact generate random data, we must run a couple randomness tests.

Sign test (*difference.sign.test* in r)

- This test subtracts consecutive values of random numbers and records their signs. If the data is truly random, we would expect both + and – signs to appear roughly equally.
- For example, if we have the following data (5,4,6,2,7,4), the test would execute as follows:
    - 4-5 returns (-)
    - 6-4 returns (+)
    - 2-6 returns (-)
    - etc…
- Running this test on our random numbers returns a statistic of -0.126 and a p-value of 0.9, signifying randomness is present.

Turning point test (*turning.point.test* in r

- This test checks that the numbers are independent and identically distributed.
- It works by recording the length of "turning points" in the data. A turning point is essentially a local maximum or minimum. For example, this set of three data points (1,5,2) would be a turning point because 5 is the local maximum of the series 1,5,2.
- If data is truly random, the length of turning points will follow a normal distribution.
- Running this test on our random numbers returns a statistic of 1.8175 and a p-value of .06. This is a very low p-value but does suggest that our data is random.

Step 1 – Generate Data

The table below summarizes the data I generated in r. I chose to use all normal distributions with varying parameters for creating the factors and noise.

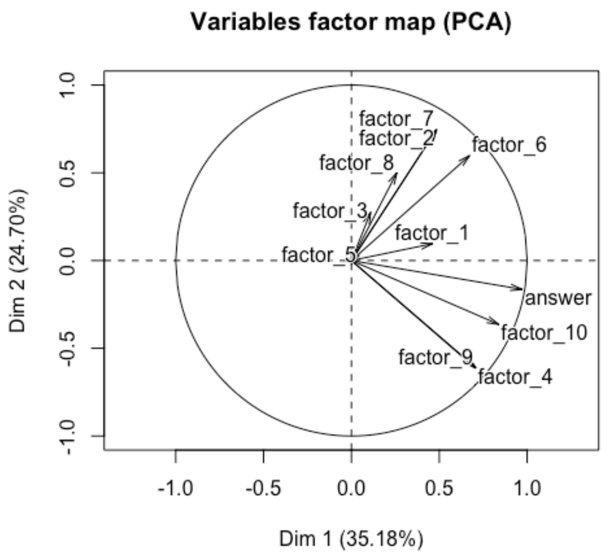
Factor	**(f1-f5) Parameters (f6-answer) Noise**	**Distribution**	**Composition**
f1	mean: 50 stdev: 10	normal	f1
f2	mean: 20 stdev: 5	normal	f2
f3	mean: 35 stdev: 7.5	normal	f3
f4	mean: 40 stdev: 10	normal	f4
f5	mean: 5 stdev: 1	normal	f5
f6	mean: 0 stdev: 1	normal	2*(f1) + 4*(f2)
f7	mean: 2 stdev: 1	normal	3*(f1) + f5
f8	mean: 2 stdev: 0.5	normal	2*(f3) + f2
f9	mean: 1 stdev: 1	normal	5*(f4) + 2*(f5)
10	mean: 10 stdev: 5	normal	2*(f1) + 2*(f4)
answer	mean: 10 stdev: 5	normal	f5 + f9 + f6 + f2

Step 2 – PCA Analysis

The first step in exploring our data is to perform a principle component analysis (PCA). This will allow us to understand how different factors are correlated with each other and with our answer, and allow us to see how much variation in our data set is explained by the different dimensions.

First we must ensure our data meets the assumptions of a PCA analysis:

- Confirm all variable are numerical - Yes
- Must have more rows(observations) than columns(factors) – Yes



Examining the PCA map above, we can extrapolate the following:

- 59.88% of the variation in the data is explained by the first two dimensions. Dimension 1 explains 35.18% of the data, while dimension 2 explains 24.70%.
- No single factor is perfectly correlated (or negatively correlated) with our answer, informing us that we will likely need to include multiple factors in our linear model in order to generate an acceptable prediction.
- We can see that factors 1, 3, and 5 are close to the origin (0,0) and therefore do not explain much of the variation in the data. We will likely not include these in our linear model.
- Factors 9, 4, and 10 are semi-correlated to our answer and will be good to include in our initial linear model. It's possible that factor 6 is also slightly correlated (it is not completely orthogonal to our answer).
- Factors 2, 7, and 8 do not appear to be well correlated with our answer, but we can still try including them in our linear model to see if they add value.
- It is not abundantly clear which factors will lead to a predictive linear model, but we've learned enough to start exploring.

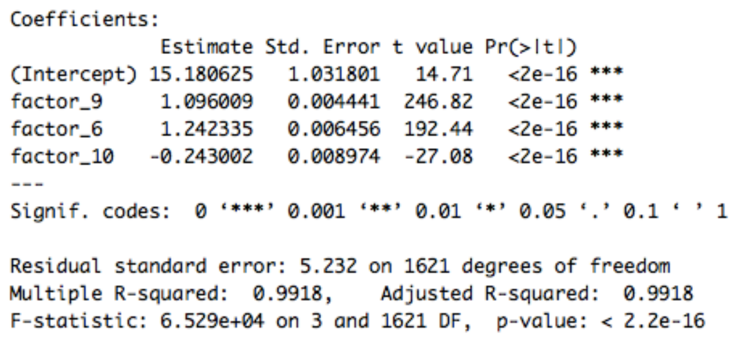
The above analysis seems to be coherent with our data. Our answer is composed of multiples of the factors 5, 9, 6, and 2.

- The likely reason we don't see 5 as a useful variable in the first two components is because the mean and standard deviation for 5 are extremely small.
- Factor 3 doesn't add much value compared to the other factors because it is not used in any of the factors 6-10, thus it is quite weak.
- Factor 9 is composed of factors 4 and 5, which is why we see that it is correlated almost perfectly with factor 4.

- It makes sense that factor 10 is semi-correlated with the answer as it is composed of factors 1 and 4.

Step 3 – Testing multiple linear models

In order to find a linear model that gives accurate predictions, I tested a few different models and picked the one with a high adjusted r-squared value and high p-value for all of the t-tests.



```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 15.180625   1.031801   14.71   <2e-16 ***
factor_9     1.096009   0.004441  246.82   <2e-16 ***
factor_6     1.242335   0.006456  192.44   <2e-16 ***
factor_10   -0.243002   0.008974  -27.08   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.232 on 1621 degrees of freedom
Multiple R-squared:  0.9918,	Adjusted R-squared:  0.9918 
F-statistic: 6.529e+04 on 3 and 1621 DF,  p-value: < 2.2e-16
```

Model ‘lm4’ (shown above) is chosen as the best, most predictive, model for the following reasons:

- Adjusted r-squared is extremely high at .9918, telling us that more than 99% of the variance in our “answer” is explained by the model.
- The F-statistic is <.05, telling us that at least one coefficients is not 0.
- The residual standard error, when compared to our data, is small.

In order to validate the model, we run prediction intervals against our test data set and compare the answers. We need to test that our model gives correct predictions and is not over fitting the data.

- I ran prediction intervals on all observations in the test data set. All answers were well within the prediction interval of our model.
- It looks like the model is a good predictor and does not appear to over-fit the data

We must also test the assumptions of our model.

- First, we look visually at the data to check for any obvious violations. This is done solely in the r code and no obvious violations are present.
- Durbin-Watson Test – Large p-value (.8481) indicates that our observations are independent.
- Shapiro Test – p-value (.1327). This p-value is not large, however, it is greater than .05, so we accept the null hypothesis, stating the population is normal.
- Breusch Pagan Test – Large p-value (.2769) indicating homogeneity of variance

Step 4 – Define and Execute Experiment

Objective

The goal of the experiment is to find the combination of factors and their values that *maximize* the answer.

Process variables

The design will consist of factors 6, 9, and 10.

Experimental design

I will be using a 2k factorial design with 2^3 = 8 experiments (i.e. 3 factors with 2 levels each). Next, I will apply the Yates algorithm to understand the interaction among the different factors used in the experiments (all 8 of them). Applying the yates algorithm helps us isolate the effects of each input variable.

We will use the minimum and maximum value of each factor (out of all of our 2000 observations) as our two levels. Our factorial design (shown below) is built around these two levels.

<u>Key for factorial design table</u>
A = factor_9
B = factor_6
C = factor_10

	factor_9	factor_6	factor_10	Values	values	values	ID
1	–	–	–	27.38453	94.63743	102.8763	mean
2	+	–	–	350.0508	94.63743	102.8763	A
3	–	+	–	27.38453	285.555	102.8763	B
4	+	+	–	350.0508	285.555	102.8763	AB
5	–	–	+	27.38453	94.63743	267.3875	C
6	+	–	+	350.0508	94.63743	267.3875	AC
7	–	+	+	27.38453	285.555	267.3875	BC
8	+	+	+	350.0508	285.555	267.3875	ABC

Execute the design

First, we use our linear model to 'simulate' the experiments. The answers obtained are displayed in the table below.

	factor_9	factor_6	factor_10	Values	values	values	answer	ID
1	–	–	–	27.38453	94.63743	102.8763	137.7666	mean
2	+	–	–	350.0508	94.63743	102.8763	491.4116	A
3	–	+	–	27.38453	285.555	102.8763	374.9502	B
4	+	+	–	350.0508	285.555	102.8763	728.5952	AB
5	–	–	+	27.38453	94.63743	267.3875	97.79005	C
6	+	–	+	350.0508	94.63743	267.3875	451.4351	AC
7	–	+	+	27.38453	285.555	267.3875	334.9737	BC
8	+	+	+	350.0508	285.555	267.3875	688.6187	ABC

Next, we apply the Yates algorithm in order to isolate the effects of each factor and the interactions among the factors. This allows us to see which factors produced the largest effects during our experiments and understand if factor interactions played a roll in altering the answer.

Experiment	factor 9	factor 6	factor 10	answer	Col1	Col2	Col3	Div	Effect	ID
1	27.38453	94.63743	102.8763	137.76658	629.17821	1732.72362	3305.54112	8	413.19264	mean
2	350.05079	94.63743	102.8763	491.41163	1103.54541	1572.8175	1414.5802	4	353.64505	A
3	27.38453	285.55501	102.8763	374.95018	549.22515	707.2901	948.7344	4	237.1836	B
4	350.05079	285.55501	102.8763	728.59523	1023.59235	707.2901	0	4	0	AB
5	27.38453	94.63743	267.3875	97.79005	353.64505	474.3672	-159.90612	4	-39.97653	C
6	350.05079	94.63743	267.3875	451.4351	353.64505	474.3672	0	4	0	AC
7	27.38453	285.55501	267.3875	334.97365	353.64505	0	0	4	0	BC
8	350.05079	285.55501	267.3875	688.6187	353.64505	0	0	4	0	ABC

Analyze and interpret the results, detect effects of main factors and interactions

Analyzing the experiments in combination with the Yates algorithm, we can conclude the following:

- Factor 9 has the largest overall positive effect on the answer.
    - Looking back at the initial experiments, this does seem to be the case. The four largest answers occurred when factor 9 was maximized.
- Factor 6 has a relatively large positive effect, although not as large as factor 9.
- Factor 10 has a slight negative effect.
    - We can also see this effect in the experiments. For example, if we compare experiments 3 and 7, it appears that factor 10 has lowered the answer. Of course, we didn't know this for certain until we applied the Yates algorithm and ruled out any interactions that could have played a role.
- There is no (or so minimal it isn't detected) interactions between the variables. This is understandable, as our factors are linear combinations of themselves.

Finally, just a quick note to state that we may want to check the size of the confidence interval of our answers. This is because we are using a linear model to run our experiments and it may be that our linear model isn't as predictive as we would like. If we decide that our confidence interval for 1 or more of the experiments is too large for us, we would gather more observations, generate a new linear model, and re-run the experiment. However, in this case, the confidence intervals are all very small.