



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Data Manipulation

Statistical Programming – ADDB5021

Lucy Boyt - 2308345

Table of Contents

Introduction	2
Code Explanation.....	2
Initial Setup	2
Question One – Generate index, Fibonacci & Pell Sequences	3
Question Two and Three - Calculate Pell Approximation to $\sqrt{2}$ and difference.	5
Question Four – Accuracy Calculation	7
Reflection.....	10
Conclusion.....	11
Bibliography	11

Introduction

This report provides an explanation of the R code submission

lb_data_manipulation_assignment.R supported by flowcharts and testing examples.

The program creates and stores 200 values of the Fibonacci and Pell sequences; it also calculates an approximation of the true $\sqrt{2}$ and evaluates the accuracy based on decimal precision.

Code Explanation

Initial Setup

Before executing the task solutions, the R environment was prepared and initial data structures were created to ensure outputs were legible, logic was reproducible and later steps ran efficiently.

```
#-----  
# 1. Initial Setup  
#-----  
  
# Turn Off Scientific notation to verify output.  
options(scipen = 999)  
  
# Set number of iterations dynamically  
values <- 200  
  
# Define initial values for the sequences  
index_n <- c(1, 2)  
fib      <- c(0, 1)  
pell     <- c(0, 1)  
  
# Pre-define supporting vectors  
sqrt2      <- sqrt(2)  
sqrt_diff  <- NULL  
pellsq_approx <- NULL  
accuracy    <- NULL
```

Figure 1 - Initial Setup of script

The use of `scipen [1]` was applied to disable scientific notation, ensuring the larger values of Fibonacci and Pell were displayed fully. A vector named “values” was created

to control the number of iterations the loops were to be run through. This was to make testing simple and allow easy changes later if more or fewer rows were required.

Vectors “index_n”, “fib” and “pell” were initialised with the first two terms for their sequence. “sqrt2” was computed in preparation for question three. Further supporting vectors were created with null values which would be updated later within the script.

Question One – Generate index, Fibonacci & Pell Sequences

The question requested the first two hundred Pell numbers be created and placed into a dataframe with three columns: index n, Fibonacci and pell. The Fibonacci and Pell Sequences were generated using a for loop based on their mathematical formulas:

$$\text{Fibonacci} = Fn = Fn - 1 + Fn - 2$$

$$\text{Pell} = Pn = 2Pn - 1 + Pn - 2$$

The code started its iterations from the third position up to the value stored within the “values” vector. Each iteration calculated the next number in both sequences and appended it to the respective vector. The loop also inserted the index value or iteration value into “index_n”. Once the loop was ended the vectors were inserted into a dataframe named “pelldata”.

```
#-----  
# 2. Question 1 - Generate Index, Fibonacci & Pell sequences  
#-----  
  
# For loop to calculate the next values in index_n, fib, pell up to the dynamic value vector.  
for (i in 3:values) {  
  index_n[i] <- i  
  fib[i]      <- fib [i - 1] + fib [i - 2]  
  pell[i]     <- (2 * pell [i - 1]) + pell [i - 2]  
}  
  
# insert the above vectors into a data frame called pelldata.  
pelldata <- data.frame(  
  index      = index_n,  
  fibonacci  = fib,  
  pell       = pell  
)
```

Figure 2 - Creation of Dataframe “pelldata” and vectors held within

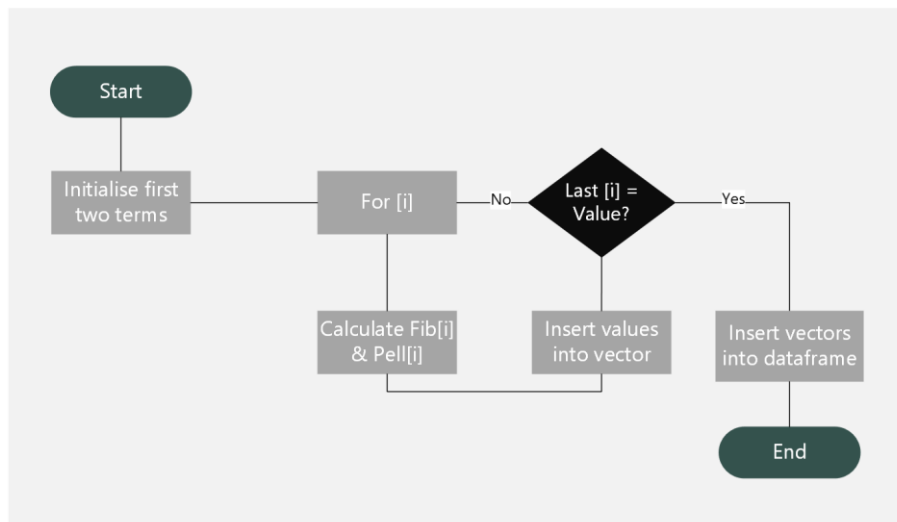


Figure 3 - Flow chart of Question 1 for loop

To verify the loop is correctly producing the expected values, the initial ten iterations validated against known Fibonacci and Pell sequences manually. The values returned matched the expected results, confirming that both sequences were functioning correctly.

	index	fibonacci	pell
1	1	0	0
2	2	1	1
3	3	1	2
4	4	2	5
5	5	3	12
6	6	5	29
7	7	8	70
8	8	13	169
9	9	21	408
10	10	34	985
11	11	55	2378
12	12	89	5741

n	fib	Pell
1	0	0
2	1	1
3	1	2
4	2	5
5	3	12
6	5	29
7	8	70
8	13	169
9	21	408
10	34	985
11	55	2378
12	89	5741

Figure 4 - Results from script compared with handwritten notes

Question Two and Three – Calculate Pell Approximation to $\sqrt{2}$ and difference.

The next questions requested the Pell approximation to $\sqrt{2}$ and the difference between the true square root value and the result using the calculated Pell values. The approximation was calculated using the following formula:

$$\frac{P_{n-1} + P_n}{P_n}$$

Both values were calculated using another for loop, relying on the length of the Pell matrices to determine the number of iterations. Each value in “pellsq_approx” was compared with the true value of $\sqrt{2}$. Truncation was chosen over rounding to maintain consistent with the assignments focus on measuring the accuracy of decimal place approximations. By truncating the results, the analysis reflects only the digits that are definitively correct avoiding the potential inflation of accuracy that rounding could introduce. The approximated value and the difference were then appended to the existing data frame as new columns.

```
#-----  
# 3. Questions 2 & 3 - Calculate Pell approximation to  $\sqrt{2}$  and difference  
#-----  
  
# For loop to calculate the Pell Approximation and the difference between the approximation.  
for (i in 2:length(pell)) {  
  # Calculate approximation  
  pellsq_approx[i] <- (pell[i-1] + pell[i]) / pell[i]  
  
  # set decimal place scale (5)  
  dp <- 10^5  
  
  #Calculate the difference between the approximation to  $\sqrt{2}$  (to set decimal places)  
  sqrt_diff[i] <- trunc((pellsq_approx[i] - sqrt2) * dp) / dp  
}  
  
# insert values into data frame (pelldata) as new column  
pelldata$pellsq_approx <-pellsq_approx  
pelldata$sqrt_diff <-sqrt_diff
```

Figure 5 - Calculation of approximate and difference before adding to the dataframe.

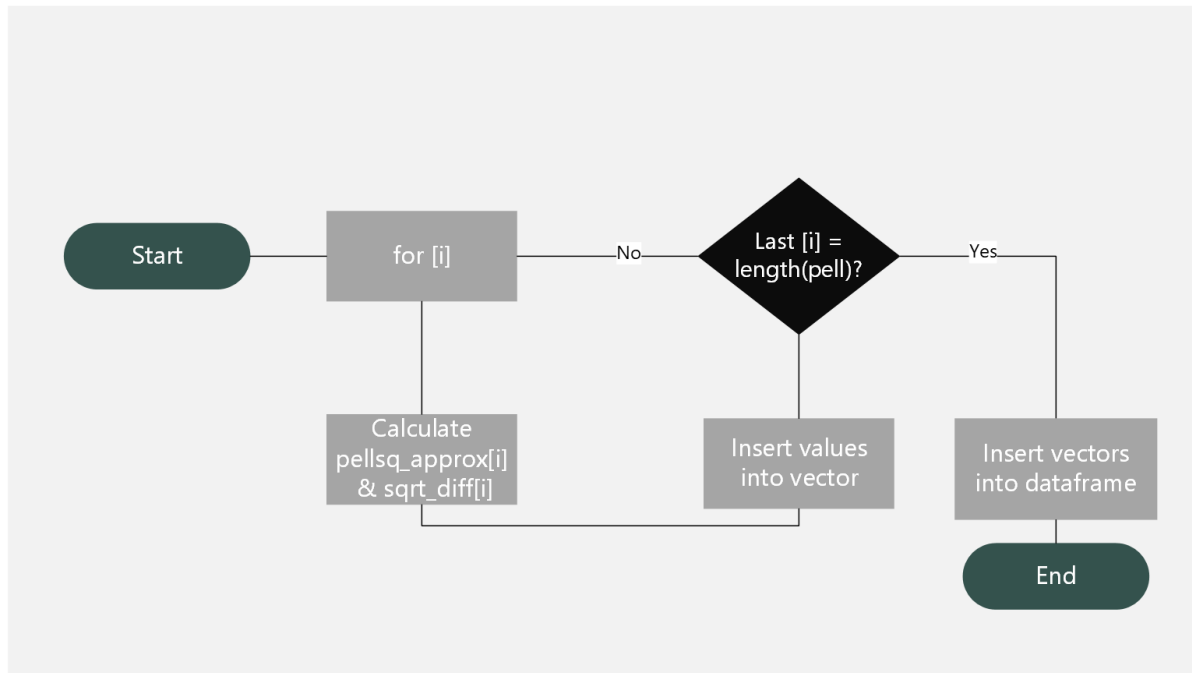


Figure 6 - Flow chart of Questions 2 and 3 - for loop

To verify the calculations were correct the first twelve rows were inspected using `pelldata[1:12,]` and compared with manual calculations. As expected, the “sqrt_diff” decreased each iteration, confirming the approximated value gets closer to the actual value of $\sqrt{2}$.

pell	pellsq_approx	sqrt_diff
0	NA	NA
1	1.000000	-0.41421
2	1.500000	0.08579
5	1.400000	-0.01421
12	1.416667	0.00245
29	1.413793	-0.00042
70	1.414286	0.00007
169	1.414201	-0.00001
408	1.414216	0.00000
985	1.414213	0.00000
2378	1.414214	0.00000
5741	1.414214	0.00000

Pell	Calculations	Pell Sq - approx	difference $\sqrt{2} = 1.4142135623$
0	NA	NA	NA
1	$\frac{0+1}{1} = \frac{1}{1}$	1	-0.4142135623
2	$\frac{1+2}{2} = \frac{3}{2}$	1.5	0.0857864377
5	$\frac{2+5}{5} = \frac{7}{5}$	1.4	-0.0142135623
12	$\frac{5+12}{12} = \frac{17}{12}$	1.4166666666	0.0024531044
29	$\frac{12+29}{29} = \frac{41}{29}$	1.4137931034	-0.0004204589
70	$\frac{29+70}{70} = \frac{99}{70}$	1.41428571428	0.0000721520
169	$\frac{70+169}{169} = \frac{239}{169}$	1.41420118343	-0.0000123789
408	$\frac{169+408}{408} = \frac{577}{408}$	1.41421568627	0.0000021240
985	$\frac{408+985}{985} = \frac{1393}{985}$	1.41421319796	-0.000003643
2378	$\frac{985+2378}{2378} = \frac{3363}{2378}$	1.41421362489	0.000000626
5741	$\frac{2378+5741}{5741} = \frac{8119}{5741}$	1.41421355164	-0.000000107

Figure 7 - Results from script compared with handwritten notes

Question Four – Accuracy Calculation

The final section of the program identifies how accurate each pell approximation is when compared with the true value of $\sqrt{2}$. Accuracy is measured by counting the number of decimal places within each vector in “pellsq_approx” match $\sqrt{2}$ within progressively smaller decimal places.

The logic uses a nested for-loop. The outer loop iterates through each pell approximation, if the value in the vector is “NA” then “NA” is returned to maintain the dataframe consistent length, any other values are passed through the inner loop.

```
#-----  
# 4. Question 4 - Calculate accuracy (number of correct decimal places)  
#-----  
  
for (i in 1:length(pellsq_approx)) {  
  # Handle NA's by inserting an NA in the vector  
  if (is.na(pellsq_approx[i])) {  
    accuracy <- c(accuracy, NA)  
  }  
  else {  
    counter <- 0 # counter for decimal point accuracy  
  
    for (j in 1:20) { # accuracy beyond 15-16 is limited by double-precision representation  
      # Calculate the lowest and highest allowance around sqrt(2) based on current decimal place (j)  
      lower_allow <- sqrt2 - (1/ (10^j))  
      upper_allow <- sqrt2 + (1/ (10^j))  
  
      # inserts decimal place if the approximation is between lower and upper allowance, otherwise stops loop  
      if (pellsq_approx[i] >= lower_allow & pellsq_approx[i] <= upper_allow) {  
        counter <- j  
      } else {  
        break  
      }  
    }  
    # appends the last counter value for this approximation.  
    accuracy <- c(accuracy, counter)  
  }  
}  
  
# insert values into data frame (pelldata) as new column  
pelldata$accuracy <- accuracy  
  
# Testing code:  
# pelldata[1:12,]  
# datacheck <- pelldata[c(6, 15, 32, 45, 70, 130, 200), ]
```

Figure 8 - Nested loop to identify accuracy of pellsq_approx.

The inner loop checks each approximation against progressively narrower constraints:

$$lower_allow = \sqrt{2} - 10_j^{-1}$$

$$upper_allow = \sqrt{2} + 10_j^{-1}$$

If the value in pellsq_approx is between these values the counter increases, once the value is outside the range the loop breaks, and the final counter value is inserted into the vector.

index	F _n	P _n	pellsq_approx	sqrt_diff	accuracy
6	5	29	1.413793	-0.00042	3
15	377	80 782	1.414214	0.00000	10
32	1 346 269	259 717 522 849	1.414214	0.00000	20
45	701 408 733	24 580 185 800 219 268	1.414214	0.00000	20
70	117 669 030 460 994	91 196 316 011 299 234 022 705 885	1.414214	0.00000	20
130	407 305 795 904 080 553 832 073 954	8 443 420 432 013 143 050 795 938 339 643 913 980 856 932 710 785	1.414214	0.00000	20
200	173 402 521 172 797 813 159 685 037 284 371 942 044 301	5 257 949 235 243 715 217 981 999 992 354 509 006 832 052 451 ...	1.414214	0.00000	19

Figure 11 - ChatGPT's values returned

Notably row two hundred returns a different accuracy value compared to ChatGPT's expected nineteen. According to R Documentation R uses IEEE 754 double-precision standard for all numeric values which means it can store roughly 15-16 significant decimal digits accurately. In the case of this script when the Pell numbers exceed fifteen decimal places R cannot represent them exactly resulting in small rounding errors which in turn impacts the accuracy column.

Double-precision values

All R platforms are required to work with values conforming to the IEC 60559 (also known as IEEE 754) standard. This basically works with a precision of 53 bits, and represents to that precision a range of absolute values from about 2×10^{-308} to 2×10^{308} . It also has special values `NaN` (many of them), plus and minus infinity and plus and minus zero (although R acts as if these are the same). There are also *denormal(ized)* (or *subnormal*) numbers with values below the range given above but represented to less precision.

See `.Machine` for precise information on these limits. Note that ultimately how double precision numbers are handled is down to the CPU/FPU and compiler.

In IEEE 754-2008/IEC60559:2011 this is called 'binary64' format.

Figure 12 - Snippet of R Documentation explaining Double-Precision [3]

Reflection

The assignment required the manipulation of data using loops, conditional logic, and structured programming in R. A key challenge arose when calculating the accuracy of the Pell approximation to $\sqrt{2}$. The original code used a fixed threshold range against the “sqrt_diff” rather than the approximation. This approach did not truly measure the number of decimal places within the approximation matched $\sqrt{2}$. Through testing and reviewing the output the issue was identified and the code revised.

```
for (i in 1:length(sqrt_diff)) {  
  if (is.na(pellsq_approx[i])) {  
    accuracy[i] <- NA  
  } else {  
    if (sqrt_diff[i] < 0) sqrt_diff[i] <- -sqrt_diff[i]  
  
    if (sqrt_diff[i] < 0.00001) accuracy[i] <- 5  
    else if (sqrt_diff[i] < 0.0001) accuracy[i] <- 4  
    else if (sqrt_diff[i] < 0.001) accuracy[i] <- 3  
    else if (sqrt_diff[i] < 0.01) accuracy[i] <- 2  
    else if (sqrt_diff[i] < 0.1) accuracy[i] <- 1  
    else accuracy[i] <- 0  
  }  
}  
  
pelledata$accuracy <- accuracy
```

Figure 13 - Original accuracy calculations (incorrect)

```
# Calculate the difference between the approximation to  $\sqrt{2}$  (to 5 decimal places)  
sqrt_diff[i] <- round(pellsq_approx[i] - sqrt2, 5)
```

Figure 14 - Original rounding code used

Initially, the difference values were rounded rather than truncated. Subsequent testing demonstrated that truncation provided a more appropriate measure of accuracy as it ensured only verified digits were counted, while rounding would inflate results.

Another consideration involved R's double-precision numeric format, which limits how accuracy when values approached $\sqrt{2}$. Understanding this constraint supported more reliable interpretation of the results.

Testing against known Fibonacci and Pell values, ChatGPT and manual computations verified that calculations and data structures were functioning correctly. Overall, the task demonstrated effective use of R for iterative programming and data manipulation, highlighting the importance of accuracy and validation within analytical workflows.

Conclusion

The final program demonstrated structured data manipulation and validation in R. The process emphasised the value of systematic testing, precise logic, and clear documentation to ensure accurate analytical outcomes within computational programming tasks.

Bibliography

- [1] GeeksforGeeks, “How to prevent scientific notation in R?,” *GeeksforGeeks*, May 12, 2021. <https://www.geeksforgeeks.org/r-language/how-to-prevent-scientific-notation-in-r/> (accessed Nov. 09, 2025).
- [2] N. S. Matloff, *The art of R programming : a tour of statistical software design*. San Francisco: No Starch Press, 2011.
- [3] Datalab, “double function - RDocumentation,” *Rdocumentation.org*, 2019. <https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/double> (accessed Nov. 09, 2025).