The periodicity of male nocturnal emission (Draft 2)

Richard Baxter^{*,✉}

✉2915/350 William St, Melbourne VIC 3000, Australia

* richardbrucebaxter@gmail.com

Abstract

Mental health diagnosis has become separated from theoretical models [1]. Masturbation is a response to (imaginary) stimuli indicative of the environment of evolutionary adaptedness for copulation. Given that culture (including clothing) exists to limit experience of this environment to private, proprietary relations, it is possible that masturbation has become unwittingly normalised in sexually saturated (overstimulated) societies. If so, we would expect nature to exhibit alternate, unconscious systems of sexual regulation. It was first hypothesised (H1) that conscious sexual activity was not an inevitable behaviour for males, in that it could be entirely replaced by nocturnal emission (wet dreams). It was further hypothesised (H2) that periodicity in nocturnal emission would be exhibited, and based on preliminary data (H2B) that the periodicity observed would match the female menstrual cycle. The preliminary data set comprised a qualitative 1390 day observation and 210 day record of nocturnal emission. The primary data set comprised a 2641 day record featuring a total 471 instances of nocturnal emission. All hypotheses were confirmed. It is concluded that care must be taken to partial out societal changes (constraints) before reframing psychological health.

Author Summary

Introduction

Psychological health is presently ill-defined [2] and many clinical classification schemes are atheoretical (e.g. DSM-III+ [1]). While medical health/diagnosis is grounded in biology (societal independent construction of conditions), mental health diagnosis has become separated from theoretical models. Attempts are therefore being made to ground mental health classification in biology (e.g. NIMH). Moreover, psychological health is considered to interact with society. For example, DSM classifications are dependent on an individual operating correctly (flourishing) in their society. This leaves open the possibility however of the society itself being non-conducive to human flourishing, and so mental health construction may become entangled in the failings of an individual society or societies. This is especially the case if research does not adequately span spatio-temporal horizons across culture. Mental health is rarely investigated cross-culturally so as to abstract psychological truths at a species level.

First world societies have evolved rapidly over the last century (shift in social norms). Traditional large scale societies (civilisations) which maintain interactions beyond kin or tribe typically exhibit features that mitigate adaptive tendencies in males/females to respond procreatively to stimuli (e.g. clothing, proprietary). WEIRD (western educated industrialised rich democratic) nations have seen a shift however towards the elimination

of such norms. Such has been facilitated by the availability of artificial birth control and abortion (the elimination of the natural consequences of sexual dysregulation).

Masturbation is a response to (imaginary) stimuli indicative of the environment of evolutionary adaptedness for copulation. Given that culture (including clothing) exists to limit experience of this environment to private, proprietary relations, it is perhaps not coincidental that masturbation has become normalised in sexually saturated (overstimulated) societies. See S1 Appendix for an example of cultural shifts in attitudes towards mental health and sexual morality (compare USA boy scout handbook 1911 to that of 1967 on male maturation), or likewise, 19th and 20th century accounts in which masturbation was reviled [3, 4]. The construal of mental health has evolved to accommodate an increasingly large variety of sexual practices. It remains to be determined whether such changes in classification are a product of an increasingly adept theoretical understanding of human psychology, or are a byproduct of changes in societal expectations with regards to sexual propriety.

Likewise, it remains to be determined whether conscious *sexual activity* (intercourse or otherwise) is a necessary behaviour for a maturing/mature human or is a response to the (perceived) environment. Modern psychology regularly posits that it is a normal phenomenon for *sexually inactive* (null intercourse) males in a civilised society, as opposed to it being a response to (imaginary) stimuli indicative of the onset of natural sexual relations (as encoded within the environment of evolutionary adaptedness). The framing of masturbation as a cognitively ordered or biologically adaptive phenomenon (e.g. [5]) may therefore be incorrect. Such could be a culturally biased deduction based on an expectation for *Homo sapiens* to function normally in phylogenetically recent and therefore evolutionarily foreign large scale societies—without restrictions on propriety. If so, we would expect nature to exhibit alternate, unconscious systems of sexual regulation.

Nocturnal emission (or wet dreams) are commonly exhibited during male maturation. Historic estimates of their frequency have been measured to be once every two weeks on average [6]. Furthermore, there is circumstantial evidence to suggest that the experience of nocturnal emission is decreasing [6] for maturing males, which aligns with the inverse correlation observed between frequency of pornographic masturbation and nocturnal emission [7] and the increasing prevalence of pornographic consumption amongst youth (approaching 85% consumption) [8].

Some general principles regarding the prevalence of nocturnal emission are evident. Although the consistency of historic accounts is disputed [4], it has been suggested that nocturnal emission is strongly affected by sexual activity, in that its occurrence can be eliminated by the conscious acceptance of a subconscious reaction to real or imagined stimuli (i.e. intercourse or masturbation respectively) [7]. It may also be weakly affected by the environment irrespective of sexual activity (e.g. saturated experience of sexual stimuli), although such a possibility has not been investigated here. Therefore, in order to test the periodicity of nocturnal emission its occurrence was observed across a *complete sexually inactive* time period (null copulation/masturbation).

An experiment was conducted to confirm (H1) that conscious sexual activity was not an inevitable or necessary phenomenon in an adult male. It was further hypothesised (H2) that if nocturnal emission is a natural and psychologically healthy behaviour for regulating sexual drive in human males, then rhythmic periodicity may be exhibited.

Method

Preliminary qualitative observations suggested that (in accordance with H1) conscious sexual activity was not inevitable (1390 day complete sexually inactive observation). Likewise, (in accordance with H2) periodicity was observed in a preliminary data set that imprecisely (+/- 5 day accuracy) recorded nocturnal emission events (210 days).

Furthermore, it was observed that nocturnal emission appeared to occur on an approximately 30 day cycle, coincidental with the average 28 day (SD:4) female menstrual cycle [9]. The ability of a male cycle (if existent) to coordinate with that of its female group may be adaptive. It was therefore further hypothesised (H2B) that the periodicity observed would match the female menstrual cycle.

These findings were retested in a controlled setting across a large time period. The existing primary data set comprises a 2641 day record featuring a total 471 instances of nocturnal emission (wet dream) with zero instances of sexual activity/masturbation. In order to maintain this requirement (in a sexually suggestive society), both external and internal stimuli were analysed for their influence on prospective behaviours/desire, and a high degree of exercise (physical discipline) was engaged in.

Instances of nocturnal emission were recorded upon wake (or the following day).

Results

An average of 5.4 nocturnal emission events were observed per month.

According to the first hypothesis (H1), it was found that conscious sexual activity was entirely replaced by subconscious sexual activity (nocturnal emission) for the time period sampled. See Fig. 1 and Fig. 2 for graphs of the recorded instances of wet dreams (WD), either stacked (where repeat events occurred within a 1-3 day time period), or used to create a histogram (with preliminary smoothing applied, enabling capture of repeat events across an approx 1-3 day time period). See S2 Appendix for a record of the primary raw dataset.

Figure 1. Record of all instances of wet dreams (WD) in the preliminary and primary data gathering stages, in stacked format (where repeat events occurred within a 1-3 day time period).

Figure 2. Record of all instances of wet dreams (WD) in the preliminary and primary data gathering stages, in histogram format (with preliminary smoothing applied, enabling capture of repeat events across an approx 1-3 day time period).

When fitting a model of periodicity to the data, the optimum models indicated approximate 60 and 30 day periodicity respectively (frequency of the wet dream calculated optimum period across all data segmentation sizes and indices). That is, of the periodic models being overlayed with the data (periodicity spikes modelled as gaussian distributions), the models most frequently converged at an optimum periodicity of 74 (67-78) and 37 (30-43) days. See Fig. 3 below. See S3 Appendix for a record of the software used to calculate the optimum periodicity (and offset) for different data segmentation sizes and indices.

Figure 3. Frequency of calculated male WD optimum period (days) across all data segmentation sizes and indices.

Discussion

Both hypotheses were confirmed. It was found (H1) that conscious sexual activity was not an inevitable or necessary phenomenon for an adult male. This concurs with 19th and 20th century accounts of masturbation/nocturnal emission [3, 4].

Likewise, it was found (H2) that nocturnal emission exhibited periodicity, similar ($\lesssim 2\sigma$) in frequency to that of the female (average 28 day) menstrual cycle (H2B). This concurs with various 19-20th century anecdotal accounts of nocturnal emission periodicity [10]; "four weeks", "at regular intervals of exactly a month", "every fortnight". Likewise, various 19-20th century anecdotal accounts cite similar rates of nocturnal emission to this sample; "the average frequency is about 3 per month of twenty-eight days" [10], 3-4 per month [3], 3-4 per month [4].

The results must be replicated for a large sample of individuals.

It is concluded that in the effort to better construe mental health, theoretical psychology needs to focus harder on partialling out societal changes before reframing psychological health (or presenting traditionally ostracised behaviours as normal).

Acknowledgments

Jerry. is acknowledged for providing preliminary review of this manuscript.

Supporting Information

S1 Appendix

Cultural shifts in mental health and sexual morality

S2 Appendix

Primary dataset of wet dream events (raw)

S3 Appendix

WD experiment data analysis software program

References

1. Surís A, Holliday R, North CS. The evolution of the classification of psychiatric disorders. *Behavioral Sciences*. 2016;6(1):5.
2. Hickie IB, Scott J, Hermens DF, Scott EM, Naismith SL, Guastella AJ, et al. Clinical classification in mental health at the cross-roads: which direction next? *BMC medicine*. 2013;11(1):125.
3. Ellis H. Nocturnal emissions. *Am J Psychol*. 1904;15:104–107.
4. Burg B. Nocturnal emission and masturbatory frequency relationships: A 19th-century account. *Journal of sex research*. 1988;24(1):216–220.
5. Robbins CL, Schick V, Reece M, Herbenick D, Sanders SA, Dodge B, et al. Prevalence, frequency, and associations of masturbation with partnered sexual behaviors among US adolescents. *Archives of pediatrics & adolescent medicine*. 2011;165(12):1087–1093.
6. Matthews BS, Wells JW. A Comparative Study of Nocturnal Emissions. *Journal of Sex Education and Therapy*. 1983;9(2):26–31.
7. Yu CKC. Pornography consumption and sexual behaviors as correlates of erotic dreams and nocturnal emissions. *Dreaming*. 2012;22(4):230.
8. Lim MS, Agius PA, Carrotte ER, Vella AM, Hellard ME. Young Australians’ use of pornography and associations with sexual risk behaviours. *Australian and New Zealand journal of public health*. 2017;41(4):438–443.
9. Chiazze L, Brayer FT, Macisco JJ, Parker MP, Duffy BJ. The length and variability of the human menstrual cycle. *Jama*. 1968;203(6):377–380.
10. Ellis H. *Studies in the Psychology of Sex, Volume 1 The Evolution of Modesty, The Phenomena of Sexual Periodicity, Auto-Erotism*. tredition; 2012.

S1 Appendix - Cultural shifts in attitudes towards mental health and sexual morality

Extract from US Boy Scouts Handboook (1911) - "Conservation"

In this chapter much has been said of the active measures which a boy should take in order to become strong and well. We should be equally concerned in saving and storing up natural forces we already have. In the body of every boy, who has reached his teens, the Creator of the universe has sown a very important fluid. This fluid is the most wonderful material in all the physical world. Some parts of it find their way into the blood, and through the blood give tone to the muscles, power to the brain, and strength to the nerves. This fluid is the sex fluid. When this fluid appears in a boy's body, it works a wonderful change in him. His chest deepens, his shoulders broaden, his voice changes, his ideals are changed and enlarged. It gives him the capacity for deep feeling, for rich emotion. Pity the boy, therefore, who has wrong ideas of this important function, because they will lower his ideals of life. These organs actually secrete into the blood material that makes a boy manly, strong, and noble. Any habit which a boy has that causes this fluid to be discharged from the body tends to weaken his strength, to make him less able to resist disease, and often unfortunately fastens upon him habits which later in life he cannot break. Even several years before this fluid appears in the body such habits are harmful to a growing boy.

To become strong, therefore, one must be pure in thought and clean in habit. This power which I have spoken of must be conserved, because this sex function is so deep and strong that there will come times when temptation to wrong habits will be very powerful. But remember that to yield means to sacrifice strength and power and manliness. For boys who desire to know more of this subject we would suggest a splendid book by Dr. Winfield S. Hall, entitled, "From Youth into Manhood." Every boy in his teens who wants to know the secret of strength, power, and endurance should read this book.

Extract from US Boy Scouts Handboook (1965) - "From Boy to Man"

From Boy to Man. At the age of 13, 14, or 15 (sometimes earlier, sometimes later) you not only grow, but numerous changes take place in your body. Your voice changes. Hair appears on your face, in the armpits, around the sex organ. Your sex organ increases in size.

All these changes are caused by the function of the sex glands or testicles. They produce fluids or secretions that have a great effect on your development.

At times the glands discharge part of their secretions through the sex organ during sleep. This process is called a nocturnal emission or a "wet dream." It is perfectly natural and healthy and a sign that nature has taken care of the situation in its own manner. There are boys who do not let nature have its own way with them but cause emissions themselves. This may do no physical harm, but may cause them to worry.

Any real boy knows that anything that causes him to worry should be avoided or overcome. If anything like this worries you, this is not unusual just about all boys have the same problem. Seek the correct answer to any question which bothers you about your development from boy to man. But be sure to get your information from reliable sources — your parents, your physician, your spiritual adviser.

frequency.txt (number of events per day);

1

S3 Appendix - WD experiment data analysis software program

WDexperimentDataAnalysis.h

```
/*
*****
* File Name: WDexperimentDataAnalysis.h
* Author: Richard Bruce Baxter - Copyright (c) 2016
* Project: WD Experiment Data Analysis
* Project Version: 1a1d 28-August-2016
* Requirements: requires frequency data generated by WDexperiment-x.ods
*****
*/

#define WD_OPTIMISE_PARAMETERS_PERIOD_AND_OFFSET //1a1b
#define WD_OPTIMISE_PARAMETERS_SEGMENTATION //1a1c
//define WD_DEBUG_USE_PRELIM_DATA

#define WD_WRITE_FREQUENCY_MODEL
#define WD_WRITE_FREQUENCY_AVERAGED
#define WD_MODEL_GAUSSIAN //else use cosine distribution
#define WD_DATA_FREQUENCY_FILE_NAME "frequency.txt"
#define WD_DATA_FREQUENCY_PRELIM_FILE_NAME "frequencyPrelim.txt"
#define WD_DATA_FREQUENCY_AVERAGED_FILE_NAME "frequencyAveraged.txt"
#define WD_DATA_FREQUENCY_THEORETICAL_FILE_NAME "frequencyTheoretical.txt"
#define CHAR_NEWLINE '\n'

#include <fstream>
#include <iostream>
#include <ctime>
#include <cstring>
#include <cstdlib>
#include <stdlib.h>
#include <math>
#include <string>

#include <vector>
#include <list>
#include <random>

using namespace std;

int main();
void findOptimumPeriod();
bool getFilesFromFileList(string inputFileName, vector<string>* inputTextFileNameList, int* numberOfInputFilesInList);
void writeStringToFile(string fileName, string* s);
int convertStringToInt(string number);
string convertDoubleToString(double number, string format);
double absoluteValue(double val);
#define WD_MODEL_GAUSSIAN
double calculateGaussianPeriodCoverage(int period, double gaussianStandardDeviationPerPeriod);
double calculateGaussianDistributionAtPoint(int i, int period, double gaussianStandardDeviationPerPeriod);
double calculateGaussianDistribution(double x, double mean, double standardDeviation);
#define WD_MODEL_GAUSSIAN
double calculateCosinePeriodCoverage(int period);
double calculateCosineDistributionAtPoint(int i, int period);
```

WDexperimentDataAnalysis.cpp

```
/*
*****
* File Name: WDexperimentDataAnalysis.cpp
* Author: Richard Bruce Baxter – Copyright (c) 2016
* Project: WD Experiment Data Analysis
* Project Version: 1aid 28–August–2016
* Requirements: requires frequency data generated by WDexperiment-x.ods
*****
*/
#include "WDexperimentDataAnalysis.h"

int main()
{
    findOptimumPeriod();

    void findOptimumPeriod()
    {
        //http://www.cplusplus.com/reference/random/normal_distribution/
        #ifdef WD_DEBUG USE PRELIM DATA
        string frequencyFileName = WD_DATA_FREQUENCY_PRELIM_FILE_NAME;
        #else
        string frequencyFileName = WD_DATA_FREQUENCY_FILE_NAME;
        #endif
        vector<string> frequencyListString;
        vector<int> frequencyList;
        int numberOfDays = 0;
        if(!getFilesFromFileList(frequencyFileName, &frequencyListString, &numberOfDays))
        {
            cout << "main:{}_error: !getFilesFromFileList:_" << frequencyFileName << endl;
        }
        int numberOfEvents = 0;
        for(vector<string>::iterator iter=frequencyListString.begin(); iter != frequencyListString.end(); iter++)
        {
            string frequencyString = *iter;
            int frequency = convertStringToInt(frequencyString);
            frequencyList.push_back(frequency);
            numberOfEvents = numberOfEvents + frequency;
        }

        double averageNumberOfEventsPerDay = double(numberOfEvents)/double(numberOfDays); //calculated based on sum of frequency.txt
        //convergence optimisation parameters
        int segmentationDefault = 1000; //180: 6 months 365: 12 months //numberOfDays;
        #ifdef WD_OPTIMISE_PARAMETERS SEGMENTATION
        int segmentationMin = numberOfDays/4/4; //approx 3 months //theoretical ideal segmentation = 6 months (360/2)
        int segmentationMax = numberOfDays;
        int segmentationInterval = 10;
        int segmentationNum = (segmentationMax–segmentationMin)/segmentationInterval;
        int segmentationOpt = -1; //48

        int segmentNum = numberOfDays/segmentationMin; //16
        int averageSegmentSize = (segmentationMax – segmentationMin)/2; //this will be an overestimate
        int segmentationNumApprox = numberOfDays/averageSegmentSize;
        cout << "segmentationMin="_" << segmentationMin << endl;
    }
}
```

```

cout << "segmentationMax_=" << segmentationMax << endl;
cout << "segmentationInterval_=" << segmentationInterval << endl;
#endif

int segmentOpt = -1;

int binDefault = 5; //averaging bin size

double seasonalVariationFractionDefault = 0.0; //seasonal variation

int seasonalVariationPeriodDefault = 360; //1 year

int seasonalVariationOffsetDefault = 0; //minimal seasonal variation

int periodDefault = 30; ///theoretical ideal period = 30 days
#ifdef WD_OPTIMISE_PARAMETERS_PERIOD_AND_OFFSET
int periodMin = 10;
int periodMax = 160;
int periodInterval = 1;
int periodNum = (periodMax-periodMin)/periodInterval; //50
int periodOpt = -1;
cout << "periodMin_=" << periodMin << endl;
cout << "periodMax_=" << periodMax << endl;
cout << "periodInterval_=" << periodInterval << endl;
#endif

int offsetDefault = periodDefault/2; //centre first period at first data point
#ifdef WD_OPTIMISE_PARAMETERS_PERIOD_AND_OFFSET
int offsetMin = 0;
int offsetInterval = 1;
int offsetNumApprox = periodMax/2; //e.g. 30
int offsetOpt = -1;
cout << "offsetMin_=" << offsetMin << endl;
cout << "offsetInterval_=" << offsetInterval << endl;
cout << "offsetNumApprox_=" << offsetNumApprox << endl;
#endif

double gaussianStandardDeviationPerPeriodDefault = 3.0; //2.0 - expect 95% of events to occur each period //consider changing to 1.0 - expect 68% of events to occur

double eventsPerDayMinimaFractionDefault = 0.0; //y offset of model distribution (allows for the modelling of a noisy distribution)

int segmentationIndex = 0;
#define SOME_LARGE_NUMBER (9999999999.9)

#ifdef WD_OPTIMISE_PARAMETERS_SEGMENTATION
double frequencyDiffAverageMin = SOME_LARGE_NUMBER;
#endif

vector<double> frequencyListTheoretical(numberOfDays);

#ifdef WD_OPTIMISE_PARAMETERS_SEGMENTATION
for(int segmentation = segmentationMin; segmentation < segmentationMax; segmentation = segmentation+segmentationInterval)
{
    #else
    int segmentation = segmentationDefault;
    #endif
    #ifdef WD_OPTIMISE_PARAMETERS_SEGMENTATION
    vector<double> segmentFrequencyDiffAverageMinList;
    double segmentFrequencyDiffAverageMinAveraged = 0.0;
    #endif
    #ifdef WD_OPTIMISE_PARAMETERS_PERIOD_AND_OFFSET

```

```

#ifdef WD_OPTIMISE_PARAMETERS_VERBOSE_OUTPUT
cout << "segmentation_" << segmentation << " _days" << endl;
#endif
double segmentPeriodOptAveraged = 0.0;
double segmentOffsetOptAveraged = 0.0;
#endif

int segmentIndex = 0;
for (int segmentStartDay = 0; segmentStartDay < numberOfDays; segmentStartDay = segmentStartDay + segmentation)
{
    int segmentEndDay = segmentStartDay + segmentation;
    if (segmentEndDay > numberOfDays)
    {
        segmentEndDay = numberOfDays;
    }
    int segmentNumberOfDays = segmentEndDay - segmentStartDay;
    double segmentFrequencyDiffAverageMin = SOME_LARGE_NUMBER;
    int bin = binDefault;
    vector<double> frequencyListAveraged(numberOfDays);
    list<int> frequencyStacked;
    for (int i = 0; i < numberOfDays; i++)
    {
        frequencyStacked.push_back(frequencyList[i]);
        if (i >= bin)
        {
            int numberOfEventsInBin = 0;
            for (list<int>::iterator iter = frequencyStacked.begin(); iter != frequencyStacked.end(); iter++)
            {
                numberOfEventsInBin = numberOfEventsInBin + *iter;
            }
            double frequencyAveraged = double(numberOfEventsInBin) / double(bin);
            //cout << "n2 = " << i << " , numberOfEventsInBin = " << numberOfEventsInBin << endl;
            //cout << "n2 = " << i << " , frequencyAveraged = " << frequencyAveraged << endl;
            frequencyListAveraged[i - (bin / 2)] = frequencyAveraged;
        }
        frequencyStacked.pop_front();
    }
}

#ifdef WD_WRITE_FREQUENCY_MODEL
//vector<string> frequencyListAveragedString;
string frequencyListAveragedString = "";
for (vector<double>::iterator iter = frequencyListAveraged.begin(); iter != frequencyListAveraged.end(); iter++)
{
    double frequencyAveraged = *iter;
    string frequencyAveragedString = convertDoubleToString(frequencyAveraged, "%0.6f");
    //frequencyListAveragedString.push_back(frequencyAveragedString);
    frequencyListAveragedString = frequencyListAveragedString + frequencyAveragedString + CHAR_NEWLINE;
}
writeStringToFile(WD_DATA_FREQUENCY_AVERAGED_FILE_NAME, &frequencyListAveragedString);
#endif

double seasonalVariationFraction = seasonalVariationFractionDefault;
int seasonalVariationPeriod = seasonalVariationPeriodDefault;
int seasonalVariationOffset = seasonalVariationOffsetDefault;
#ifdef WD_OPTIMISE_PARAMETERS_PERIOD_AND_OFFSET
int periodIndex = 0;
for (int period = periodMin; period < periodMax; period = period + periodInterval)
{
    int offsetIndex = 0;
    int offsetMax = period / 2;

```

```

for(int offset = offsetMin; offset < offsetMax; offset = offset+offsetInterval)
{
    #else
    int period = periodDefault;
    int offset = offsetDefault;
    #endif

    double gaussianStandardDeviationPerPeriod = gaussianStandardDeviationPerPeriodDefault;
    double eventsPerDayMinimaFraction = eventsPerDayMinimaFractionDefault;

    #ifdef WD_MODEL_GAUSSIAN
    double modelPeriodCoverage = calculateGaussianPeriodCoverage(period, gaussianStandardDeviationPerPeriod);
    #else
    double modelPeriodCoverage = calculateCosinePeriodCoverage(period); //will return 0 (as equal negative and positive components of distribution)
    #endif

    double eventsPerPeriod = averageNumberOfEventsPerDay*period;

    #ifdef WD_DEBUG
    cout << "modelPeriodCoverage=" << modelPeriodCoverage << endl;
    #endif

    //int startDay = segmentStartDay+offset;
    //int endDay = segmentEndDay+offset;
    double frequencyDiffAverage = 0.0;
    double frequencySumOfSquares = 0.0;
    int dayIndex = 0;
    for (int day = segmentStartDay; day < segmentEndDay; day++)
    {
        double seasonalVariationForDayModifier = calculateCosineDistributionAtPoint(day+seasonalVariationOffset, seasonalVariationPeriod)*seasonalVariationFraction;
        //cout << "seasonalVariationForDayModifier = " << seasonalVariationForDayModifier << endl;

        int i = (dayIndex+offset)%period; //int i = mod(dayIndex+offset, period);
        #ifdef WD_MODEL_GAUSSIAN
        double modelValue = calculateGaussianDistributionAtPoint(i, period, gaussianStandardDeviationPerPeriod);
        #else
        double modelValue = calculateCosineDistributionAtPoint(i, period) + 1.0; //+1.0 to ensure that distribution coverage is always positive;
        #endif
        double frequencyTheoretical = ((modelValue/modelPeriodCoverage*eventsPerPeriod) + (eventsPerPeriod*eventsPerDayMinimaFraction))/(1.0 + eventsPerDayMinimaFraction);

        #ifdef WD_DEBUG
        cout << "eventsPerDayMinimaFraction=" << eventsPerDayMinimaFraction << endl;
        cout << "modelPeriodCoverage=" << modelPeriodCoverage << endl;
        cout << "eventsPerPeriod=" << eventsPerPeriod << endl;
        cout << "modelValue=" << modelValue << endl;
        cout << "frequencyTheoretical=" << frequencyTheoretical << endl;
        #endif
        frequencyListTheoretical[day] = frequencyTheoretical;
        double frequency = frequencyListAveraged[day];
        double frequencyDiff = frequency - frequencyTheoretical;
        frequencyDiffAverage = frequencyDiffAverage + absoluteValue(frequencyDiff);
        frequencySumOfSquares = frequencySumOfSquares + pow(frequencyDiff, 2);
        dayIndex++;
    }
    double frequencyVariance = frequencySumOfSquares/(dayIndex-1);
    frequencyDiffAverage = frequencyVariance; //use variance rather than average diff
    //frequencyDiffAverage = frequencyDiffAverage/dayIndex;

    if(frequencyDiffAverage < segmentFrequencyDiffAverageMin)
    {
        segmentFrequencyDiffAverageMin = frequencyDiffAverage;
        #ifdef WD_OPTIMISE_PARAMETERS_PERIOD_AND_OFFSET
        periodOpt = period;
        offsetOpt = offset;
        #endif
    }
}

```

```

    }
    #endif
    #ifndef WD_OPTIMISE_PARAMETERS_PERIOD_AND_OFFSET
    }
    }
    }
    periodIndex++;
    #endif
    #endif

    #ifdef WD_OPTIMISE_PARAMETERS_VERBOSE_OUTPUT
    cout << "segmentIndex_" << segmentIndex << endl;
    #ifdef WD_OPTIMISE_PARAMETERS_PERIOD_AND_OFFSET
    cout << "periodOpt_" << periodOpt << endl;
    cout << "offsetOpt_" << offsetOpt << endl;
    #endif
    #else
    cout << segmentation << "\t" << segmentationIndex << "\t" << periodOpt << endl;
    #endif

    #ifdef WD_OPTIMISE_PARAMETERS_PERIOD_AND_OFFSET
    segmentPeriodOptAveraged = segmentPeriodOptAveraged + periodOpt;
    segmentOffsetOptAveraged = segmentOffsetOptAveraged + offsetOpt;
    #endif

    #ifdef WD_OPTIMISE_PARAMETERS_SEGMENTATION
    segmentFrequencyDiffAverageMinList.push_back(segmentFrequencyDiffAverageMin);
    segmentFrequencyDiffAverageMinAveraged = segmentFrequencyDiffAverageMinAveraged + segmentFrequencyDiffAverageMin;
    #endif

    segmentIndex++;
}

#ifdef WD_OPTIMISE_PARAMETERS_PERIOD_AND_OFFSET
segmentPeriodOptAveraged = segmentPeriodOptAveraged / segmentIndex;
segmentOffsetOptAveraged = segmentOffsetOptAveraged / segmentIndex;
#endif
#ifdef WD_OPTIMISE_PARAMETERS_VERBOSE_OUTPUT
cout << "segmentPeriodOptAveraged_" << segmentPeriodOptAveraged << endl;
cout << "segmentOffsetOptAveraged_" << segmentOffsetOptAveraged << endl;
#endif
#endif

#ifdef WD_OPTIMISE_PARAMETERS_SEGMENTATION
segmentFrequencyDiffAverageMinAveraged = segmentFrequencyDiffAverageMinAveraged / segmentIndex;
{
    frequencyDiffAverageMin = segmentFrequencyDiffAverageMinAveraged;
    segmentationOpt = segmentationIndex;
}
#endif

#ifdef WD_OPTIMISE_PARAMETERS_SEGMENTATION
    segmentationIndex++;
}
#endif

#ifdef WD_OPTIMISE_PARAMETERS_SEGMENTATION
    cout << "frequencyDiffAverageMin_" << frequencyDiffAverageMin << endl;
    cout << "segmentationOpt_" << segmentationOpt << endl;
    #endif
#endif

#ifdef WD_WRITE_FREQUENCY_MODEL

```

```

//vector<string> frequencyListTheoreticalString;
string frequencyListTheoreticalString = "";
for(vector<double>::iterator iter=frequencyListTheoretical.begin(); iter != frequencyListTheoretical.end(); iter++)
{
    double frequencyTheoretical = *iter;
    string frequencyTheoreticalString = convertDoubleToString(frequencyTheoretical, "%0.6f");
    //frequencyListTheoreticalString.push_back(frequencyTheoreticalString);
    frequencyListTheoreticalString = frequencyListTheoreticalString + frequencyTheoreticalString + CHAR_NEWLINE;
}
writeStringToFile(WD_DATA_FREQUENCY_THEORETICAL_FILE_NAME, &frequencyListTheoreticalString);
#endif
}

bool getFilesFromFileList(string inputListFileName, vector<string>* inputTextFileNameList, int* numberOfInputFilesInList)
{
    bool result = true;
    *numberOfInputFilesInList = 0;
    ifstream parseFileObject(inputListFileName.c_str());
    if(!parseFileObject.rddbuf()->is_open())
    {
        //txt file does not exist in current directory.
        #ifndef NLC_USE_LIBRARY_DISABLE_FUNCTIONS_LIST_WARNING
        cout << "Error: _input_list_file_does_not_exist_in_current_directory:_" << inputListFileName << endl;
        #endif
        result = false;
    }
    else
    {
        char currentToken;
        int fileNameIndex = 0;
        int charCount = 0;
        string currentFileName = "";
        while(parseFileObject.get(currentToken))
        {
            if(currentToken == CHAR_NEWLINE)
            {
                inputTextFileNameList->push_back(currentFileName);
                #ifdef GIA_MAIN_DEBUG
                //cout << "currentFileName = " << currentFileName << endl;
                #endif
                currentFileName = "";
                fileNameIndex++;
            }
            else
            {
                currentFileName = currentFileName + currentToken;
            }
            charCount++;
        }
        *numberOfInputFilesInList = fileNameIndex;
    }
    #ifdef GIA_MAIN_DEBUG
    //cout << "numberOfInputFilesInList = " << *numberOfInputFilesInList << endl;
    #endif
    return result;
}

void writeStringToFile(string fileName, string s)
{
    ofstream writeFileObject(fileName.c_str());
    //writeFileObject.write(s);
}

```

```

for(int i=0; i < s->size(); i++)
{
    writeToFileObject.put((*s)[i]);
}

writeToFileObject.close();
}

int convertStringToInt(string number)
{
    return atoi(number.c_str());
}

string convertDoubleToString(double number, string format)
{
    char stringCharStar[100];
    sprintf(stringCharStar, format.c_str(), number);
    return string(stringCharStar);
}

double absoluteValue(double val)
{
    return abs(val);
}

#ifdef WD_MODEL_GAUSSIAN
//centre the distribution about the centre of the first period in the segment
double calculateGaussianPeriodCoverage(int period, double gaussianStandardDeviationPerPeriod)
{
    double modelPeriodCoverage = 0.0;
    for(int i=0; i<period; i++)
    {
        double y = calculateGaussianDistributionAtPoint(i, period, gaussianStandardDeviationPerPeriod);
        modelPeriodCoverage = modelPeriodCoverage + y;
    }
    return modelPeriodCoverage;
}

double calculateGaussianDistributionAtPoint(int i, int period, double gaussianStandardDeviationPerPeriod)
{
    int offsetMax = period/2;
    double mean = 0;
    double sigma = 1.0;
    double x = double(i - offsetMax)/double(period)*gaussianStandardDeviationPerPeriod*2.0; // *2 to compensate for either side
    double y = calculateGaussianDistribution(x, mean, sigma);
#ifdef WD_DEBUG
    cout << "x=" << x << endl;
    cout << "y=" << y << endl;
#endif
    return y;
}
//source: http://stackoverflow.com/questions/10847007/using-the-gaussian-probability-density-function-in-c
double calculateGaussianDistribution(double x, double mean, double standardDeviation)
{
    double inv_sqrt_2pi = 0.3989422804014327;
    double a = (x - mean) / standardDeviation;
    double gaussianDistValue = inv_sqrt_2pi / standardDeviation * exp(-0.5f * a * a);
    return gaussianDistValue;
}
#else
//centre the distribution about the centre of the first period in the segment
double calculateCosinePeriodCoverage(int period)
{

```



```

double modelPeriodCoverage = 0.0;
for (int i=0; i<period; i++)
{
    double y = calculateCosineDistributionAtPoint(i, period) + 1.0; //+1.0 to ensure that distribution coverage is always positive
    modelPeriodCoverage = modelPeriodCoverage + y;
}
return modelPeriodCoverage;
}

#ifdef calculateCosineDistributionAtPoint
double calculateCosineDistributionAtPoint(int i, int period)
{
    //cout << "i = " << i << endl;
    //cout << "period = " << period << endl;
    double x = (double(i)/double(period))*(2.0*M_PI) + M_PI;
    double y = cos(x);
    //cout << "x = " << x << endl;
    //cout << "y = " << y << endl;
    return y;
}

```