Team Control # 2016021

Setting a new world record is an exceptional feat that often results in a monetary reward for the athlete who sets it. Thus, race organizers must ensure that they have the ability to pay out the bonus if the time comes to do so. Typically, organizers will have two choices: purchase prize indemnity insurance that protects them against the need to pay, or self-insure, paying out the entire prize when someone breaks the world record.

The goal of our paper is to produce a model that determines whether or not an event organizer should buy insurance for a given event. We consider multiple criteria, including the organizers' budget, the average cost of the bonus (as defined in the problem), the probability a world record is broken, and the number of races to be held, among other factors, to make recommendations about insurance by mathematically modeling the organization's financial exposure under each scenario (purchasing insurance or not).

We considered what we quantitatively defined as the financial risk to the organizing committee, rather than long-term cash flow, to account for the fact that while buying insurance ensures a financial loss in the long run, it is better to consistently endure a small financial loss than risk facing the prospect of an athlete winning a prize the race organizers cannot pay. After developing equations that quantify financial exposure, we plugged in known values for the criteria listed in the above paragraph, and used three-dimensional graphs to quantify the benefit (or harm) of buying insurance in terms of the financial exposure to the organizers. We then created a computer program based on our model that would allow the event organizers to enter the relevant information and obtain a recommendation on whether or not to purchase insurance. Since financial risk is a function of a number of different variables, we can partially differentiate the risk function with respect to each of the variables to determine the effect of a change in each variable on the financial risk to the organization, and therefore, the weight each variable ought to be given in the organizing committee's decision-making process.

In the real world, there are generally more options than simply insuring against the full cost of the bonus or not insuring at all. Many insurance contracts include provisions for a deductible -- a portion of the payout that the organizers must pay before the insurance covers the rest -- which reduces the cost of the insurance premium but means that the insurance provides less protection to the organizing committee. We thus created new financial exposure expressions that describe financial exposure as a function of the deductible amount. We may use these expressions to suggest to the organizers whether they should purchase full insurance, buy insurance associated with a certain deductible value, or self-insure.

We created user-friendly software that returns the results of our model based on the user's inputs, and we generated an overall decision-making scheme that suggests precisely which events to insure based on the events' individual needs for insurance and on the organization's ability to afford the premiums. We wrap all of our models together into a final piece of software available to the race organizers that returns an insurance recommendation for each of the 40 events by automating the decision-making process.

Another important consideration of our model is its sensitivity to the input variables. A detailed sensitivity analysis is necessary for us to predict changes in the outputs of our model based on changes in the inputs. Our approach to sensitivity analysis includes both local and global methods to give us the most accurate prediction of the effects of all sizes of variability on the input variables. The results of our sensitivity analysis are consistent with the results returned by our software and research we find in the literature, and they also assure us that our model is not very sensitive to these input changes, functioning well over a wide array of conditions.

Our model can quickly, easily, and reliably determine the most cost-effective way to insure a competition, with specific recommendations for each event, based on input variables easily accessible to the user. When an insurance policy with a deductible seems appropriate, we can recommend that as well, even

offering a recommended optimal deductible level. We are confident that our model could be immediately deployed in the real world to produce actionable, valuable results for the organizers of any competition.

From Track to Plaque: Prize Indemnification Insurance for Track and Field Events *Team 2016021*

Introduction

Breaking a world record is an extraordinary feat. Thus, monetary bonuses are often awarded for pushing the limits of human capability. In this problem, we analyze the prospect of purchasing prize indemnification insurance to protect race organizers against the need to pay large bonuses. We examine two scenarios:

1. Should the organizers of the Zevenheuvelenloop, an annual 15k race, purchase prize indemnity insurance to protect them against the need to pay out the €25000 bonus prize to record-breakers?
2. Which events, if any, should the organizers of a 40-event track and field meet consider insuring?

The question asks us to develop models for a number of different factors that fall into one of these two scenarios. Our model approaches this problem from two perspectives: from that of the insurance companies and from that of the event organizers. We first create expressions for the premium charged by the insurance company by considering its operating costs and profit (as well as the average cost of the bonus, which it must recover in the premium). We then quantify the financial exposure taken on by the organization when buying insurance or self-insuring and compare the financial exposures to suggest a course of action to the organizers.

While the problem asked us to consider two options, to buy insurance or to self-insure, those options are often not the only ones available in the real world. Many insurance plans offer a deductible policy that may offer lower financial risk as compared to both a full insurance policy and self-insurance. Thus, we create a separate expression for the financial risk when selecting such an insurance policy. We standardize the financial exposure of an insurance policy with a deductible against that of one without and then suggest a course of action for organizers: buy insurance, buy insurance with a deductible policy, or self-insure.

Ultimately, the strength of our model is its ability to take real-world constraints into account. While purchasing insurance will always result in long-run financial loss, most people would prefer a definite, small financial loss to the risk of losing a large sum of money. By quantifying financial "risk" (financial exposure) to determine whether or not to purchase insurance, we can compare the financial exposure entailed in different insurance options and make optimal recommendations to the event organizers. Also, our user-friendly software programs allow for instant, easy use, which greatly augment the real-world applicability of our model.

Variables

a = average cost of the bonus, as defined in the problem statement
b = amount of bonus paid to record-breakers (the value of the prize)
c = total number of times a race is run on average before someone wins (inclusive)
d = operating costs of the insurance company as a fraction of income
e = insurance premium paid by the race organizers to the insurance company
m = additional cost the insurance company adds to the average cost (a) to cover their operating costs and realize a profit ($m + a = e$, equation 3)
o = operating costs for the insurance company per insurance contract
p_{PV} = profit margin (present value)
p_{FV} = profit margin (future value)
q = the probability of breaking the record ($q = \frac{1}{c}$); ($q = \hat{p}$ in Part 1 to use standard confidence interval notation)
r_d = financial exposure for the race organizers if they purchase insurance with a deductible provision
r_i = financial exposure for the race organizers with deductible-free insurance
r_n = financial exposure for the race organizers without insurance
t = number of times the race is held (and thus the number of times the premium is paid; see Assumption 3)
s = total expenditures by the race organizer excluding the actual payment of the bonus or the insurance premium
u = amount of deductible in insurance policies with deductibles
β = organizing committee's budget for the race event
α = insurance company's addition to the premium as a fraction of the average cost of the bonus ($\alpha = \frac{m}{a}$)
γ = constant for the case where $r_i = r_n$ ($\gamma = \frac{m(s+a)}{ab}$), derivation on page 7

δ = constant for the case where $r_i = r_d$ ($\delta = \frac{(a+m)*(s+q*u)}{a*(b-u)}$), derivation on page 10

Assumptions

1. Taken over all events, athletes will continue to break world records at a rate similar to that in the recent past. ***Justification***: Although this may seem counterintuitive since people may expect athletes to approach the limit of human potential, data reveal that the 21st century has pushed many limits further. Better training and technology, as well as more opportunities for traditionally underrepresented athletes, have kept the record-breaking rate relatively constant.[13]
2. Insurance companies' operating costs to income ratio (*d*) is fairly constant across each continent. ***Justification***: If we treat the top 60 multinational insurance companies as a sample of the insurance companies we are considering, the standard deviation of the sample of combined ratios (the quotient of incurred losses and expenses over earned premium) is 11.4049. The standard deviation of the sampling distribution of the mean combined ratio (also known as the standard error, SE) is defined as $SE = \frac{s}{\sqrt{n}}$, where s = standard deviation of the sample, and n = size of the sample. Therefore, the standard error is $\frac{11.049}{\sqrt{60}}$ = 1.43%, meaning that the ratio of expenses to earnings is quite constant across the board. Although the combined ratio has a very low variability for multinational insurance companies, we also need to take into account the local insurance companies. To generate a more realistic approximation of the ratio of expenses to earnings, we use representative insurances companies for each continent.
3. Race organizers pay the insurer a set premium per *race* rather than per time interval. ***Justification:*** Many insurance policies do involve premiums paid over time because the risk to the insurer is proportional to time (e.g. health insurers bear more risk of illness over longer periods of time). But for prize indemnification insurance, the insurer bears zero risk except in the case of a race, where a prize might need to be paid out.
4. Both the race organizers and insurance company will do what is best for them financially (for the organizers, this may mean buying insurance to reduce financial risk). ***Justification:*** The framing of this problem does not suggest any motivation for the two entities beyond financial considerations.
5. The profit margin for insurance brokerages averages 10.2%. ***Justification:*** According to Yahoo! Finance, the average profit margin in the insurance industry is 10.2%.[20]
6. For scenarios in "the near future," the time value of money is irrelevant. ***Justification:*** The time value of money is only consequential over long periods of time. If the "near future" describes a period of fewer than five years, for example, a 0.2% interest rate means €1 today is worth €1.01 five years from now. Such gains are trivial, and we thus consider the time value of money only over longer periods of time.

Part 1: Calculating Average Cost

We approach part one in two ways: reporting *a* through direct computation and with confidence intervals. First, we compute average cost using the formula provided:

$$a = q * b = \frac{b}{c} \tag{1}$$

We can compute this for the Zevenheuvelenloop in three ways: for men, for women, and combined.

For men: $a_{men} = \frac{€25000/record}{32\ races/2\ records} = \frac{€25000/record}{16\ races/record} = €1562.50/race$

For women: $a_{women} = \frac{€25000\ /record}{32\ races/1\ record} = \frac{€25000\ /record}{32\ races/record} = €781.25/race$

Combined: $a_{combined} = \frac{€25000\ /record}{64\ races/3\ records} = \frac{€25000\ /record}{21.3\ races/record} = €1171.875/race$

Since the issue of payment is important to the organizers of the Zevenheuvelenloop regardless of the gender of the winner, the combined value is the most important for them to consider.

Assumption 1 allows us to perform inference to predict future costs. We can treat past races as a sample of all of the instances of the Zevenheuvelenloop and then create an approximate confidence interval for the proportion $\hat{p}$ of races in which records are broken using the one-proportion z-statistic. (The interval is only approximate because we cannot be assured of the Normality of the sampling distribution of $\hat{p}$ when our data only includes three successes. We simply proceed with caution.) We use the confidence interval formula:

$$\hat{p} \pm z^* \sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \text{ with } \hat{p} = \frac{3}{64} \approx .0469,\ z^* = 1.645,\ n = 64 \qquad (2)$$

(Note: The z^* value is the x-value in the standard Normal distribution, where the area under the distribution to the right of x is .05. Because there is a corresponding value to the left of zero, we have a 90% confidence interval, with 95% confidence on the upper and lower limits.)
We obtain:

$$.0469 \pm 1.645\sqrt{\frac{.0469 \times (1-.0469)}{64}} \Rightarrow (.34\%,\ 9.03\%)$$

We now plug these values of $\hat{p}$ (note that $\hat{p} = \frac{1}{c}$) into our equation for a. Then we are 95% confident the lower limit of cost lies above .0034 x €25000 = €85 per race and that the upper limit of cost lies below .0903 x €25000 = €2257.50 per race.
Therefore, our complete answer to question 1 is:

- The average cost of the bonus for men, *in the past*, was €1562.50.
- The average cost of the bonus for women, *in the past*, was €781.25.
- The average cost of all bonuses using the combined probability, *in the past*, was €1171.88. <u>We will use this value going forward, since it is the most relevant to the contest organizers.</u>
- We estimate with 90% confidence that the average combined bonus cost, *in the future*, will lie between €85 and €2257.50.

Part 2: Criteria for Insurance Companies in Pricing Determinations

Much of the remainder of this paper concerns the construction of mathematical models surrounding various types and amounts of insurance. For such models, it is important to have an end goal or objective function: the criterion by which we measure our outcomes. Assumption 4 established that all actors in this problem will make decisions on a purely financial basis, so we will judge decisions by *the extent to which they result in a) positive financial outcomes, or b) reductions in the count or severity of negative financial outcomes*, where risk (quantified as financial exposure) counts as a financial outcome and its reduction is financially beneficial.

Revenue for the insurance company comes from a premium paid by the race organizers, with one premium paid per race (Assumption 3). We now describe the components of the insurance premium from the insurer's perspective:

- Average cost: The company must recoup the average cost with each premium so that, over the long run, it does not lose money paying out the prizes.
- Time value of money: This helps the insurer, because insurance premiums received today will be worth more tomorrow due to the accumulation of interest (at a rate of 0.2% in the Netherlands)[19].
- Operating expenditures: The insurance company requires basic necessities (i.e., utilities) and paid employees, which cost money that must be recovered through the insurance premium.
- Profit: The insurance company needs to make a profit to justify its existence.

In future calculations, we will need to separate out the average cost (a) component of the insurance premium from the other components -- essentially, profit and overhead. We therefore define the insurance premium e to be the sum of the average cost a and the other associated costs (profit and operating costs) m:

$$e = a + m \qquad (3)$$

We start by examining six different criteria for insurance companies in pricing determinations:

1. Operating costs
2. Profit margin
3. Time value of money
4. Ability of participants in the race (which affects the likelihood q of a payout, therefore affecting a)
5. Number of participants in the race (which also affects q, and therefore, a)
6. Value of the prize money (which affects b, and therefore a)

Since the average cost of the bonus accounts for criteria 4-6 and is already included in the insurance premium, we primarily analyze the effects of operating costs, profit margin, and time value of money in determining the additional cost m the insurance company should include in the premium above and beyond the average cost a.

Operating Costs

Based on Assumption 2, we used data from an insurance agency with operations in the Netherlands, Ageas, to calculate European insurance companies' average operating costs per euro earned as income. According to its 2013 financial report[1], Ageas spent €13.017 billion in that year. Out of those expenses, we subtracted expenses used for and related to the payment of insurance claims and benefits (€10.8324 billion). When each organization/race pays its own average cost in the insurance premium, that money covers the insurer's cost of paying for insurance claims. Therefore, we find that the other expenses (which must be paid for with income from *m*) total €2.1846 billion. The report also stated that Ageas earned €14.01 billion in revenue. To calculate the average operating cost of the insurance company per euro it earns as income from premiums, we divided the total non-insurance/claim-based operating costs by the total income:

$$d = \frac{€2.1846\ billion}{€14.01\ billion} = 0.1559$$

From this value, we can determine the amount of money needed to cover the operating costs per contract. In this part, we were asked to consider the criteria for the Zevenheuvelenloop. Recall that we found in equation 3 that

$$e = a + m = €1171.88 + m$$

We calculated in Part 1 that $a = €1171.88$. If we multiply the value of the insurance premium *e* by the operating costs per euro of insurance premiums paid to the insurance company, we can calculate the share of its operating costs the insurance company should cover from this event's premium:

$$o = d \times e = d\,(1171.88 + m) = (0.1559)\,(1171.88 + m) = 182.73 + 0.1559\,m \tag{4}$$

Profit

The profit margin from the premium can be found by subtracting the operational costs (*o*) we found in equation (4) and the average cost of the bonus (*a*) from the total insurance premium (*e*), leaving the component of the premium that the insurer takes home as profit:

$$p_{PV} = e - a - o = m - o = m - (182.73 + 0.1559\,m) = 0.8441\,m - 182.73 \tag{5}$$

However, this equation only describes the present value of the profit. Considerations of the time value of money are financially important to the insurance company and will be discussed in the following section.

Time Value of Money (TVM)

The time value of money is an important criterion to consider because of money's potential earning capacity. If the insurance company holds one euro today, that euro, if kept in the bank and allowed to accumulate interest, will become more valuable in a few years. This can be modeled through the TVM equation:

$$FV = PV(1+r)^n \tag{6}$$

where PV = present value FV = future value r = interest rate
n = number of times the interest rate is compounded

Because of the time value of money, the insurance company would rather earn money now and pay money later.

To find the future value of the profit, we apply the TVM (equation 6) to the p_{PV} from equation 5. We use the Netherlands' real interest rate (meaning that the rate is adjusted for inflation) of 0.2%.[19] We compute the future value in ten years, a reasonable period of time that we choose since the problem did not specify a specific time period. Our future value of profit is:

$$p_{FV} = (0.8441\,m - 182.73)(1.002)^{10} = 0.8611\,m - 186.42 \tag{7}$$

Weighing the Criteria

We can express *m* in terms of the operating costs and profit. To do so, we rearrange equation (4):

$$m = 6.4144\,o - 1171.88 \tag{8}$$

We then rearrange the equation for the future value of profit (equation 7) to obtain:

$$m = 1.1613\,p_{FV} + 216.491 \tag{9}$$

We now graph these two functions:

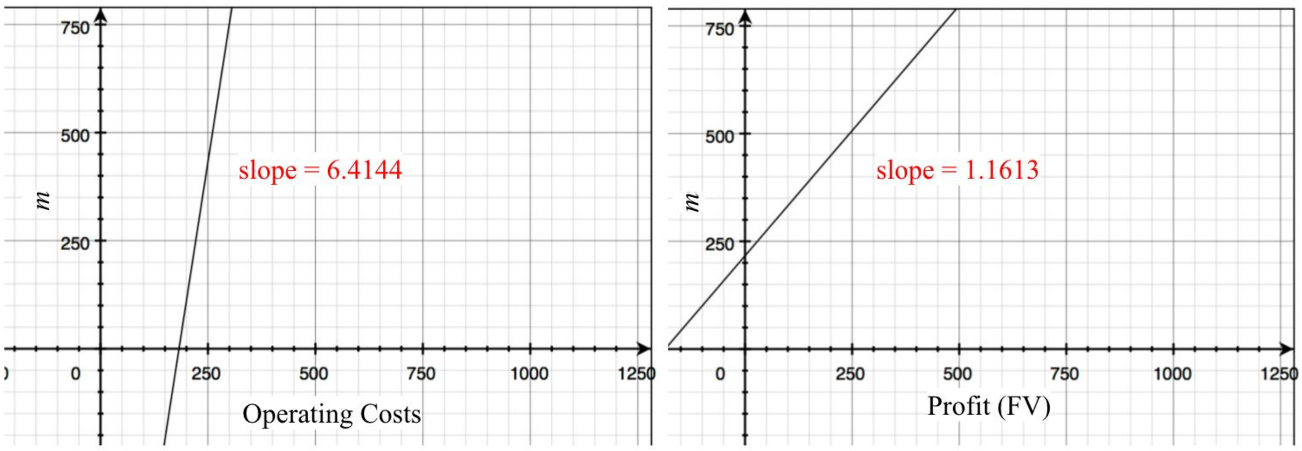


Figure 1. Graphs of additional cost m versus operating costs (o) and future value profit (p_{FV}) imposed by insurer on top of average cost of bonus

These graphs provide information about how changes in operating cost and desired profit affect the amount of the insurance premium the company charges. Specifically, the slope of operating costs graph indicates that if operating costs increase by €1, the insurance company must increase m by €6.4144, accounting for the operating cost to premium income ratio ($\frac{1}{.1559} = 6.4144$). Similarly, the slope of the profit graph shows that if the insurance company desires to increase the future value of its profit by €1, m must increase by €1.1613. This shows that the greater the value of the slope, the larger the effect the variable in question will have on the cost of the premium.

To quantify the weight of operating costs versus profit, we examine the relative weights of the slopes:

$$w_o = \frac{6.4144}{6.4144+1.1613} = 84.6654\% \qquad w_p = \frac{1.1613}{6.4144+1.1613} = 15.3346\%$$

Put more simply, changes in operating costs contribute to 84.7% of changes in premium costs, while changes in the profit desired contribute 15.3%. Therefore, it follows that we give operating costs a weight of 84.7% in the decision making process while we give profit a weight of 15.3%. We will apply this type of analysis later on when we discuss weights in greater detail in our response to question four. This conclusion is reasonable since operating costs are relatively fixed, while profit varies based on revenue. As variability increases, the effect on premium costs decreases, since it is not as critical to recover the cost in question from the premium; it is more important to the insurer to cover operating costs and avoid potential losses than to add additional profit.

Example Case

We were asked to consider the case where the insurer adds 20% to the average cost of bonus so that the premium covers his operating costs and provides a profit over a period of time as affected by the TVM.

First, we find 20% of the average cost of the bonus, or, by our definition, m, the amount added to the premium beyond the average cost of the bonus:

$$m = €1171.88 * .2 = €234.38$$

Then, we find the operating costs using equation 4:

$$o = €182.73 + .1559 * m = €182.73 + .1559 * €234.38 = €219.27$$

Finally, we find both the present value and the future value of the profit using equations 5 and 7:

$$p_{PV} = .8441 * €234.38 - €182.73 = €15.11$$
$$p_{FV} = .8611 * €234.38 - €186.42 = €15.40$$

We find that the insurer would use a 20% increase in premium costs above the a-value to earn €219.27 to cover operating costs and to earn €15.40 in future value of profit as computed 10 years from today.

Part 3: Criteria for the Organizing Committee to Purchase Insurance

Insurance, in nearly all cases, is a for-profit business. Prize indemnification insurance firms in particular therefore exist only for the prospect of profits. Over the long run, and over a large pool of insured entities, the insurance company nets a profit, and those who are insured lose money. Then why would anyone buy insurance? The answer is that insurance limits risk by distributing it among all the people and organizations insured. People who see enormous, but unlikely, negative financial outcomes without insurance (such as needing to pay out a prize to a runner) benefit the most from insurance, and insurers suffer a loss from this small subset of customers. For the majority of customers who never experience this type of loss, the insurance is a drain on their financial assets. In other words, people buy insurance because they would rather be guaranteed a small negative financial outcome than take the risk of sustaining a large negative one.

Therefore, if we were to consider this problem solely by net expenditures by the organizing committee, our model would always recommend that the organizers self-insure because it is the only way to reduce long-term expected cost. The organizers, however, face the risk of being unable to pay out the full prize value. For each event, the prize payment may lie somewhere between "impossible to pay out without going bankrupt" and "not a significant financial burden on the race organizers." We need to know where to draw the line between an affordable payout from an impossible one, and quantify the risk on a payment in between. Thus, we create a way of quantifying the financial exposure of purchasing insurance and self-insurance: we assign a financial value to the risk the organizers would take on by not purchasing insurance.

We define r_i and r_n as the financial exposure if the organizer buys insurance and if the organizer self-insures, respectively. The financial exposure describes the amount of money the organizer expects to pay scaled with the prospect of a high payout and the ability of the organizer to afford such a payout. It serves as a metric that can compare the risks of buying insurance and self-insuring on the same scale. Thus, when the financial exposure of purchasing insurance is less than the financial exposure of self-insurance ($r_i < r_n$), the organizers should purchase insurance. If the financial exposure of purchasing insurance is greater than self-insurance ($r_i > r_n$), then the organizer should self-insure.

We base our calculation of financial exposure on four criteria:

1. Prize money
2. Probability of breaking the world record
3. Budget and spending for the race (in this specific example/part, the Zevenheuvelenloop)
4. Number of races held (and thus, opportunities for the record to be broken)

The first two criteria, prize money and the probability of breaking the world record, are known for the Zevenheuvelenloop and absorbed into the calculation for the average cost, which we incorporate into the expression for financial exposure. Ultimately, our values describing financial exposure change depending on the budget and the number of races held (average cost has no effect because it is a constant and cancels for relative risk calculations), which will determine whether or not to buy insurance.

Financial Exposure When Purchasing Insurance

If the organizers purchase insurance, they will never be liable for a prize payout. Therefore, their financial exposure is limited to the price of the insurance premium. Calculating the financial exposure over a number of races t, we obtain:

$$r_i = t * e = t * (a + m) \tag{10}$$

If we let $\alpha = \frac{m}{a}$, then we can rewrite financial exposure as:

$$r_i = a * t * (1 + \alpha) \tag{11}$$

The value of α can be calculated based on the desired profit margin set by the insurance company (see equation 5 to determine the appropriate m-value and then divide it by a since $\alpha = \frac{m}{a}$). It is the percentage of the average cost that the insurance company adds on to cover operating cost and realize a profit. We will use α in the calculations below.

Financial Exposure When Self-insuring

To calculate the financial exposure associated with self insurance, we created a scaled numerical value that represents how much money the organizers would need to save per race to ensure that they would have enough money to pay out the bonus prize when the record is broken. We introduce the variable k as a scaling factor that takes into account the fact that people would rather be guaranteed a small financial loss than risk losing a large amount of money. Thus, the financial exposure of self-insurance is:

$$r_n = a * t * (1 + k) \tag{12}$$

We know that in order for the organizers to accumulate enough money to pay the bonus over the long run, they must save at least a euros per race (of course, this does not hold true in the short run, where a world record, however unlikely, might be broken on the second race). We artificially increase the amount of financial exposure by adding a factor of *k* * *a* * *t*. Mathematically, we define *k* as some constant γ multiplied by the proportion of the race budget the prize money occupied by the prize money:

$$k = \gamma * \frac{b}{\beta} \tag{13}$$

To standardize *k*, we find a scenario in which the two financial exposure values would equal: when it would not be more risky to choose one course of action over the other:

$$r_i = r_n$$
$$a * t * (1 + \alpha) = a * t * (1 + k)$$
$$\alpha = k$$
$$k = \gamma * \frac{b}{\beta}$$
$$\frac{m}{a} = \gamma * \frac{b}{\beta}$$

Because we set the two risks equal, the organization is saving just enough to pay the prize itself, and so the budget β should equal *s* + *a*. This is because we assume that if the race organizer saves the average cost of bonus for each race as part of their budget, they will, over the long run, have the capacity to pay the prize money in full when a world record is broken. Therefore, there would not be much of a difference in financial exposure for the organizer whether or not they buy insurance because the risk of the potential inability of them to pay the full prize money in the short run (from not saving enough in a small number of races) is balanced by the drawback of paying the additional cost *m* charged by the insurance company in the insurance premium when no one breaks a world record. We can now find γ in terms of the constants *m*, *a*, *s*, and *b*:

$$\frac{m}{a} = \gamma * \frac{b}{s+a}$$
$$\gamma = \frac{m(s+a)}{ab} \tag{14}$$

Thus, we can use equation 12 to write the financial exposure of self-insurance as:

$$r_n = a * t * (1 + \gamma * \tfrac{b}{\beta}), \text{ where } \gamma \text{ is the constant we found above for the case } r_i = r_n \tag{15}$$

Given these two expressions for financial exposure, we can use them to suggest that the race organizer purchase insurance when $r_i < r_n$ and not purchase insurance when $r_i > r_n$. (If the organizers cannot afford the insurance premium, r_i is always greater than r_n.)

Zevenheuvelenloop

In the case of the Zevenheuvelenloop, we set numerical values for some of the variables defined above.

From Assumption 5, the profit margin the average insurance company hopes to earn is 10.2% of their total premium income (since the premium is the insurance company's only source of revenue). Using the present value profit formula (equation 5), we can calculate the amount the company must add to the average cost of the bonus to determine the total cost of the premium:

$$p_{PV} = 0.102(e) = 0.102(a + m) = 0.102(1171.88 + m)$$
$$0.102(1171.88 + m) = 0.8441m - 182.70 \text{ (right side from equation 5)}$$
$$m = €407.27$$

We can calculate the value of α by dividing the value of *m* by the average cost:

$$\alpha = \frac{m}{a} = \frac{€407.27}{€1171.88} = 0.35 \tag{16}$$

The value of α can be used to calculate the financial exposure entailed by purchasing insurance (equation 11):

$$r_i = 1171.88 * t\,(1 + 0.35) = 1582.038 * t$$

To find the financial exposure of self-insurance, we need to calculate the value of γ using equation 14. From the problem, we are given that the prize money, *b*, for the Zevenheuvelenloop is equal to €25000, and we found *a* to be €1171.81 in Part 1. From the Zevenheuvelenloop records[11], we know that the amount the organizing committee spent on the 2008 race, *s*, was €338000. We use all these values to find γ:

$$\gamma = \frac{m(s+a)}{ab} = \frac{€407.27(€338000+€1171.81)}{(€1171.81)(€25000)}$$
$$\gamma = 4.7484$$

We can write the financial exposure of self-insuring (using equation 12) as:

$$r_n = 1171.88 * t * (1 + 4.7484 * \tfrac{25000}{\beta}) = 1171.88 * t * (\tfrac{\beta+118710.158}{\beta}) \tag{17}$$

We now analyze the results graphically. Specifically, we graph the difference between the financial exposure when self-insuring and the financial exposure when purchasing insurance (r_n - r_i) to examine how budget and the number of times a race is hosted affect whether or not our model suggests that the Zevenheuvelenloop organizers should purchase insurance. The individual graphs for r_n and r_i on their own can be found in the Appendix.

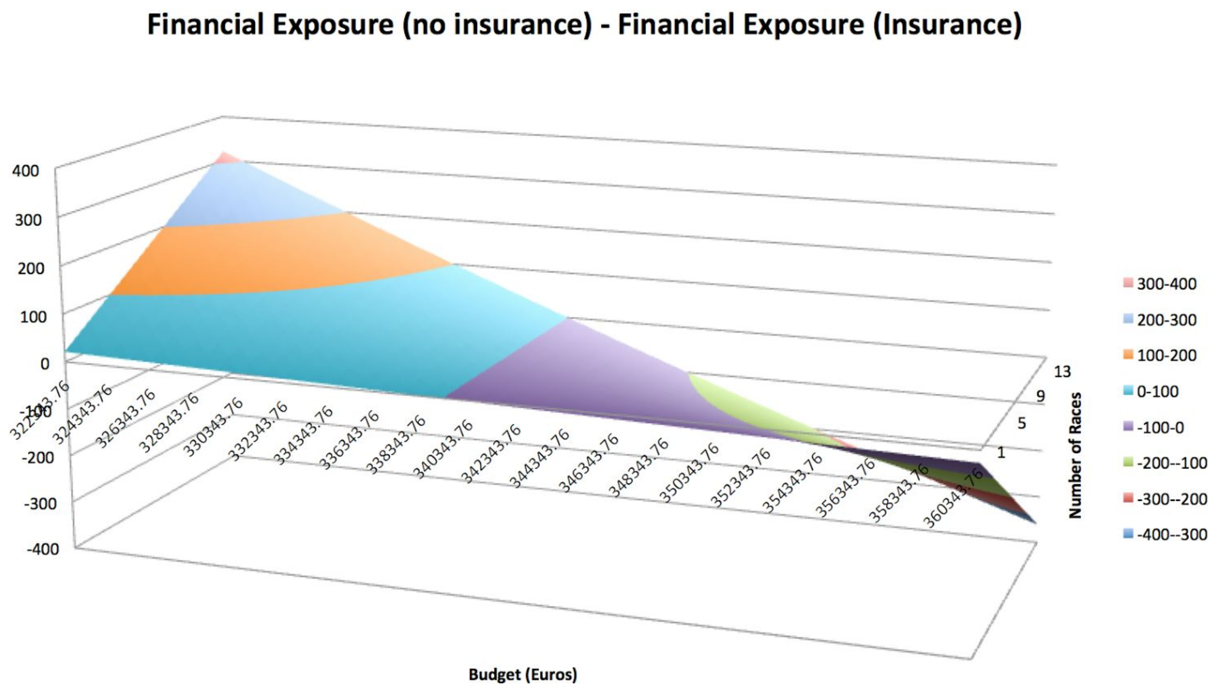


Figure 2. 3-D surface graph of r_n - r_i versus budget and the number of Zevenheuvelenloop races held

This graph shows that as the budget and the number of times the race is hosted increases, it becomes less worthwhile to purchase insurance. This is reasonable, since as the budget for the race increases, the organizers have more money and are more likely to be able to pay the prize money, meaning paying for insurance they do not need becomes unnecessary. Furthermore, as the number of times the race is hosted increases, the organizers accumulate more money from the participation fee, giving them an excess reserve from which to pay out the bonus. However, the budget has a far more significant effect on determining whether or not to purchase insurance.

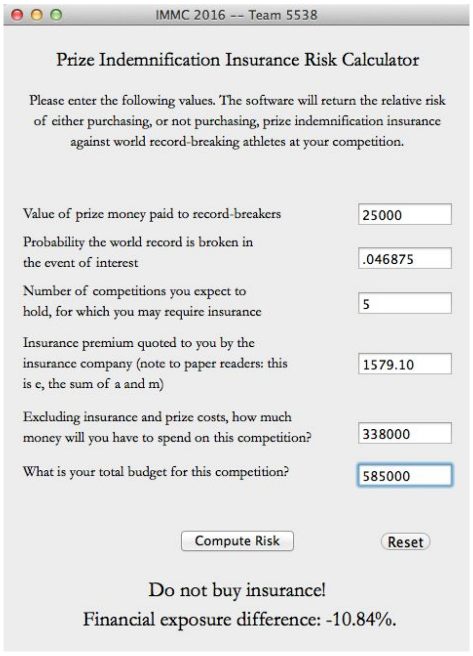
Although the graph provides an overall visual representation of the results of our model, we wish to express these results in a more precise and quantitative fashion. The next section concerns the development of a computer program to do so.

Objective-C User-Facing Program #1: Relative Risks Calculator

Although we devote much of this paper to the development of equations and formulas that can mathematically model the scenario we have been given, we must also keep in mind that our model is designed for a real-world race-organizing committee for practical application. We would also like a way to rapidly verify the results of some of our models in a precise and quantitative fashion. To these ends, we develop software that automates computation using our models.

We develop three programs in this paper, and address the first one here. We title this program "Prize Indemnification Insurance Risk Calculator." A screenshot of sample output is included below, and the full source code (written in Objective-C, as with all the user-facing programs included in this paper) may be found in the Appendix starting on page 32. The program takes six inputs, which, in variable form, are b, q, t, e, s, and β. Using the methods described above, the code captures the user's input, converts it into numerical values, calculates $a = qb$, calculates $m = e - a$, calculates $\gamma = \frac{m(s+a)}{ab}$, and then calculates the two risk values: $r_n = a * t\,(1 + \gamma\,(\frac{b}{\beta}))$; $r_i = a * t\,(1 + \frac{m}{a})$. It then computes the difference between the two risks, divides by the risk with insurance, and reports a percentage risk difference equal to $\frac{r_n - r_i}{r_i} * 100\%$. This difference is positive when self-insurance is more risky, and it is negative when buying insurance is more risky.

In our example screenshot, below, we run the numbers for the Zevenheuvelenloop through the software. The prize is €25,000; the odds of breaking the world record (from Part 1) are $\frac{3}{64} = .046875$; we assume they will hold five competitions in the future; we use equation 16 to write that $e = 1.3475 * a$ and obtain an insurance premium of €1579.10; we use the €338,000 spending figure[11]; and finally, we set the total budget as 30,000*€19.50 = €585,000, an approximation of the budget using €19.50 entry prices for 30,000 runners.[21] Our software affirms the decision of the organizers of the Zevenheuvelenloop to self-insure: as a result of the wide gap between their budget and their spending, they can easily afford to pay any record-breakers without insurance. Since insurance is more expensive in the long run, and the organizers can easily pay for the prize using existing funds, the software suggests not buying insurance, resulting in a predicted net financial exposure difference of -10.84%.


IMMC 2016 -- Team 5538

Prize Indemnification Insurance Risk Calculator

Please enter the following values. The software will return the relative risk of either purchasing, or not purchasing, prize indemnification insurance against world record-breaking athletes at your competition.

Value of prize money paid to record-breakers	25000
Probability the world record is broken in the event of interest	.046875
Number of competitions you expect to hold, for which you may require insurance	5
Insurance premium quoted to you by the insurance company (note to paper readers: this is e, the sum of a and m)	1579.10
Excluding insurance and prize costs, how much money will you have to spend on this competition?	338000
What is your total budget for this competition?	585000

Do not buy insurance!
Financial exposure difference: -10.84%.

Figure 3. The risk calculator we have built uses our model to make recommendations for the Zevenheuvelenloop.

Deductibles

Up to this point, we have been assuming that the race organizers pay a certain premium to the insurance company, and the insurance company indemnifies them against the entire cost of the payout. But this is often not the case in the real world. Many insurance plans lower premiums by including so-called "deductibles" as part of the insurance contract. An example of an insurance plan with a deductible is shown below:

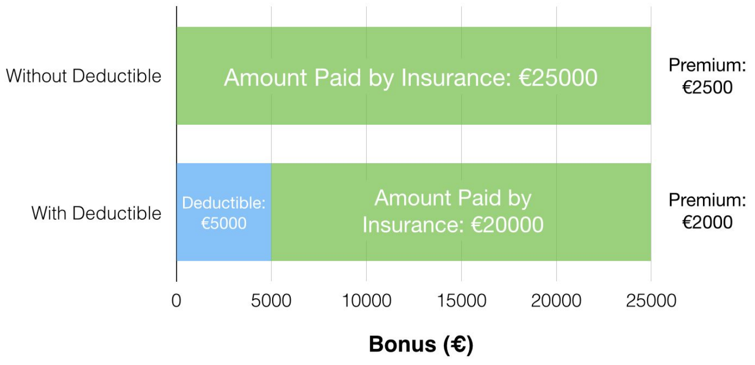


Figure 4. A hypothetical scenario in which the insurance company charges a premium equal to 10% of the amount it covers, showing that higher deductibles reduce premium costs.

With an insurance contract that includes deductibles, the organizing committee is liable for the payout costs up to a certain extent, and the insurance company is liable for all costs beyond the value of the deductible.

The purchaser of the insurance would therefore opt to insure him/herself against part of the risk and sign an insurance contract with a lower premium, but agree to pay a deductible before the insurance steps in.

In order to quantify the financial exposure involved when buying insurance with deductibles, we need to generate an equation that quantifies financial exposure based on the amount of the deductible. We first calculate the expected insurance premium for an insurance policy with deductibles (using equation 11):

$$e_d = (b-u)*q*t*(1+\tfrac{m}{a}) \tag{18}$$

where *b* - *u* replaces *b* in equation 11/15 since a deductible effectively means that the insurance covers a lower amount. Then, we need to also factor in the additional financial exposure caused by the possibility of paying the full amount of the deductible when a world record is broken. Essentially, we add the premium component of equation 11 to the payout component of equation 15, modified to reflect deductibles (see Figure 5 below).

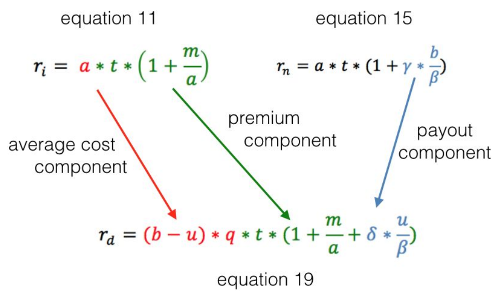

equation 11
equation 15
$r_i = a*t*\left(1+\frac{m}{a}\right)$
$r_n = a*t*(1+\gamma*\frac{b}{\beta})$
average cost component
premium component
payout component
$r_d = (b-u)*q*t*(1+\frac{m}{a}+\delta*\frac{u}{\beta})$
equation 19

Figure 5. Deriving equation 19 from equations 11 and 15 but using deductibles rather than total payouts

We define a constant δ in the same way we previously defined γ in our equation for r_n, which quantified the payout amount without insurance. Under an insurance plan with deductibles, that payout amount is now *u* rather than *b*, so $\gamma\frac{b}{\beta}$ becomes $\delta\frac{u}{\beta}$. Our financial exposure expression becomes:

$$r_d = (b-u)*q*t*(1+\tfrac{m}{a}+\delta*\tfrac{u}{\beta}) \tag{19}$$

with the new factor $\delta\frac{u}{\beta}$ to include the possibility of financial exposure from paying the deductible in the case when the world record is broken. To standardize our r_d equation, we equate it with r_i in the situation where $\beta = s+u*q$ because $u*q$ is the new average value the race organizer expects to pay if a world record is broken (previously, when we derived an expression for γ, we used $\beta = s+a = s+bq$, but the average cost of the payout is now *uq* rather than *bq*). Solving for δ (we omit *t* because it cancels on both sides):

$b*q*(1+\frac{m}{a}) = (b-u)*q*(1+\frac{m}{a}+\frac{\delta*u}{s+u*q})$ (left side from equation 11; right from equation 19)

$$\delta = \frac{(a+m)*(s+q*u)}{a*(b-u)} \tag{20}$$

After plugging our expression for δ back into equation 19, we have:

$$r_d = (b-u)*q*t*(1+\tfrac{m}{a}+\tfrac{(a+m)*(s+q*u)}{a*(b-u)}*\tfrac{u}{\beta}) \tag{21}$$

Because all of the variables other than *u* (*b, q, t, m, a, s,* and β) are known constants for each event/insurance policy, we are able to plot financial exposure in terms of the amount of the deductible. Note that *a* is no longer *b* * *q* but rather (*b* - *u*) * *q* because the average cost of the bonus that the insurer has to pay is no longer the entirety of the prize money but the prize money minus the amount of the deductible, which will be paid for by the organizers if the world record is broken.

To make this model more user-friendly (the race organizer likely does not have information about the cost of the premium), we found *m* in terms of all of the other known variables, assuming a 10.2% profit margin for the insurance company (Assumption 5) and an operating cost to income ratio *d* depending on the continent of the insurer (Assumption 2):

$$m = \frac{(.102+d)*(b*q-q*u)}{.898-d} \quad \text{*See derivation on Appendix pg. 80} \tag{22}$$

Using constants from the Zevenheuvelenloop as a test case, we tested equation 21 with three different β values to reveal the three possible scenarios:

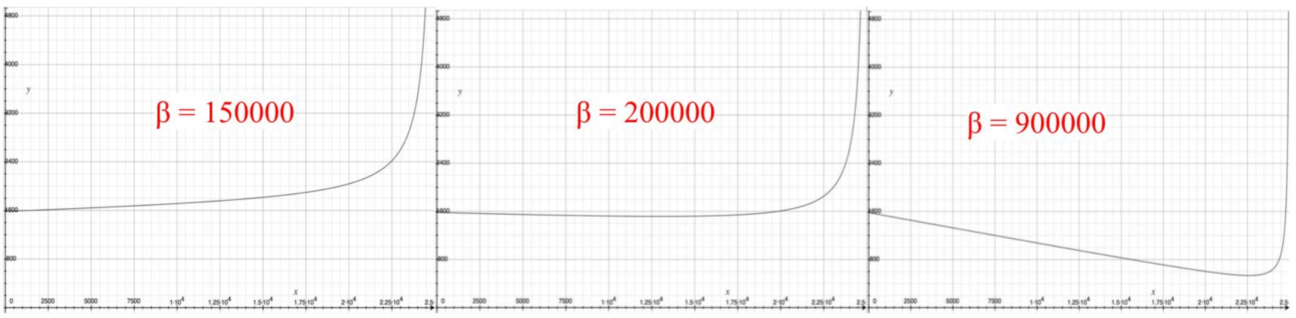


Figure 6. Test cases for financial exposure using the deductible model (x = deductible; y = financial exposure)

The graphs illustrate the the relationship between financial exposure and the deductible amount. The risk function asymptotically approaches b, the total cost of the payout. If the race organizers are paying for all of b, they are essentially self-insuring. As the deductible amount approaches b, however, the relative financial exposure exponentially increases because purchasing insurance, even with a large deductible, will still entail an additional cost added to the average cost of bonus so the insurer can make a profit. Thus, when approaching b, it becomes more advantageous to self-insure.

In the first graph, the organizers have a relatively small budget of €150000. We examine the behavior of the financial exposure function in the interval from $u = 0$ to $u = b$. Analysis is only necessary across this interval because if the deductible is equal to the payout, the organizers are self-insuring and if the deductible is zero, the organizers are purchasing the full insurance. The financial exposure for the deductible increases across the interval in question if the organizers have a budget of €150000, meaning that the organizers should consider purchasing full insurance (the minimum risk occurs at around $u = 0$).

In the second graph, the organizers have a budget of €200000. The financial exposure dips and reaches a relative minimum when the value of the deductible equals €13061.67. In this scenario, the risk associated with purchasing a deductible is lower than the risk either self-insuring (u = €25,000) or paying for the entire insurance (u = €0). Thus, the organizers should negotiate an insurance policy with a deductible of around €13060.

In the final graph, the organizers have a budget of €900000. As the value of the deductible b increases, the financial exposure decreases until the value of the deductible is very close to the value of b. Similar to the second graph, there is an absolute minimum between 0 and b; however, in this case, the value of u at the minimum is much closer to b. In this case, it is impractical for the organizers to purchase insurance because the minuscule amount they save does not financially justify paying staff to select and purchase insurance for the race (for a quantitative justification, see the Deductibles section in Part 5).

Monte Carlo Simulation

For our sensitivity analysis, we utilize Monte Carlo simulation by using pseudo-random numbers drawn from a uniform (0,1) distribution in MATLAB. For the simulation of one event (the Zevenheuvelenloop), we take a finite number of repeated samples, each of which simulates a finite number of times of running the event. The simulation returns to us the cost of the insurance premium (or the payout, under self-insurance) over a sample of races. We use this to create a probability distribution of the cost of running the event a certain number of times. We bring in the known variables q=.0469 and p=€25000 for the Zevenheuvelenloop.

Our simulation considers the use of three policies: full insurance, no insurance, and insurance with a deductible. Below are the equations for calculating the per-event cost of each method of insurance, which we repeat 100 times and then use as one sample in our probability distribution on page 4 of the Appendix.

Uninsured: $\textit{if random number from uniform } (0,1) \textit{ distribution} < q,\ \textit{annual cost} = b;$
$\textit{else},\ \textit{annual cost} = 0$

Full Insurance: $\textit{annual cost} = \textit{insurance flat rate} = (1+\frac{m}{a}) * b * q$

Scaled Deductible: $\textit{if random number } < q,\ \textit{annual cost} = (1+\frac{m}{a})(1-\frac{u}{b}) * b * q + \frac{u}{b};$
$\textit{else},\ \textit{annual cost} = (1+m)(1-\frac{u}{b}) * b * q$

As the problem states, the organizing committee loses money over a long period of time if it purchases insurance, as insurance companies relies on a large pool of customers to generate a profit. The 95% confidence intervals and means for the costs of each of these policies over a sample of 1000 races are:

Uninsured:	(720000, 1160000)	Mean: €938422
Fully Insured:	(1031250, 1031250)	Mean: €1031250
Deductible (20%):	(969000, 1057000)	Mean: €1012775

As one would expect, the consequence of the lower mean cost over many races without insurance has a wider confidence interval for the cost in each of the samples, since a few record-breakings can create a large amount of variability. In other words, the organizing committee can choose to obtain a lower mean cost over 1000 races of the Zevenheuvelenloop at the expense of having less predictability of the cost from race to race. This is why calculating confidence intervals for the financial exposure is important, because while an average lower cost is preferable for the organizers of the race, they have to take into account the chance that a world record will be broken without sufficient funding or insurance to prevent the committee from going into bankruptcy (this represents a scenario near the upper extreme of the confidence interval).

Part 4: Criteria for the Track and Field Meet Organizer to Purchase Insurance

There are several factors that will impact whether or not purchasing insurance will create greater financial exposure for the organizers relative to purchasing insurance. First, the amount of prize money (b) increases financial exposure if self-insuring because the organizer has a lower chance of being able to pay the prize money if b is high. Second, the percentage likelihood that the world record will be broken at this event (q) affects whether or not insurance should be purchased: the greater likelihood the record will be broken, the more advantageous it is for the organizers to buy insurance due to the increased probability of paying the prize money if they self-insure. Third, the amount of the money in the organizer's budget available to be spent on either paying the prize money directly or paying the insurance premium ($\beta - s$) affects the decision to buy or not buy insurance because if the organizers have enough money in their budget to pay for the prize money, they would not need to buy insurance. Conversely, if the organizer does not have nearly enough money to pay the prize without going bankrupt, they have much more reason to buy insurance. Last, the amount of the insurance company's expenditures as a fraction of income (d) affects the insurance purchase decision because the insurance company's operating costs (which vary roughly by continent, see Assumption 2) affect the price of the premium, which in turn affects the desirability of an insurance purchase.

When the organization chooses events for which to purchase insurance, it is logical that the organization chooses events for which the record has a higher probability of being broken. Since the record-breaking frequency of different events has serious financial implications, research has already been done into the determinants of that frequency. Perhaps unsurprisingly, researchers have uncovered a connection between the "accessibility" of a sport and the frequency with which its records are broken.

For example, eastern Africa is home to some of the world's best athletes, as well as some of its worst poverty. East Africans with bodies well-suited to swimming might never encounter a swimming pool, as researchers point out[12]. Much of the human potential remains untapped, and the world record reflects those with bodies less-suited to swimming or the small proportion of ideal athletes who do have more access to athletic resources. By contrast, strong athletes in East Africa have much more opportunity to develop their running ability, and running is one of the sports with the highest accessibility. This disparity in accessibility explains the observation that *world records are much less likely to be broken in high-accessibility sports*, where many more people have had the opportunity to try and beat previous standards, and where more of human potential has been tapped.

Although this is arguably intuitive, we now quantitatively examine the most salient determinants of the decision to insure; that is, which factors ought the race organization consider most heavily in their determination to purchase, or not purchase, insurance? To answer this question, we find the derivatives (slopes) of the difference between non-insured and insured financial exposure, which determines whether or not the organizer should purchase insurance and how much more risk they take on acting opposite that determination, against each criterion variable. To derive $r_n - r_i$, we first find m in terms of a and d by assuming that the profit margin for the insurance company is around 10.2% (see Assumption 5) and equating 10.2% of the total premium ($e = a + m$) and the profit calculated from m - o:

$$.102 * (a + m) = (1 - d) * \text{m} - a * d \text{ (from equation 5)}$$

$$m = \frac{.102a+a*d}{.898-d}$$

$$\frac{m}{a} = \frac{.102+d}{.898-d} \tag{23}$$

We find r_n - r_i to be $b * q * \frac{m}{a} * (1 - \frac{s+b*q}{\beta})$ and derive equation 24 by substituting in $\frac{.102+d}{.898-d}$ for $\frac{m}{a}$. We obtain:

$$r_n - r_i = b * q * (\frac{d+.102}{.898-d})*(1 - \frac{s+b*q}{\beta}) \tag{24}$$

We then partially differentiate equation 23 by each variable (b, q, β, d, s) assuming all other variables remain constant to find the how much the financial exposure difference is changed by one unit change in each factor (see equations 25-29). The partial differentiation provides us with five weights, one weight per factor, since the amount of influence each factor has on the risk should directly relate to the weight we assign to it. To compute the weight of each factor relative to the others, we divide each raw weight by the sum of the raw weights to obtain a relative weight for each (see equations 30-34)

$$\frac{\partial(r_n-r_i)}{\partial b} = X_b = \left|\frac{2*q*(b*q+.5*(s-\beta))*(d+.102)}{\beta*(d-.898)}\right)\right| \tag{25}$$

$$\frac{\partial(r_n-r_i)}{\partial q} = X_q = \left|\frac{2*b*(b*q+.5*(s-\beta))*(d+.102)}{\beta*(d-.898)}\right)\right| \tag{26}$$

$$\frac{\partial(r_n-r_i)}{\partial \beta} = X_\beta = \left|\frac{-b*q*(d+.102)(s+bq)}{\beta^2*(d-.898)}\right| \tag{27}$$

$$\frac{\partial(r_n-r_i)}{\partial d} = X_d = \left|\frac{-b*q*(s+bq-\beta)}{\beta*(d-.898)^2}\right| \tag{28}$$

$$\frac{\partial(r_n-r_i)}{\partial s} = X_s = \left|\frac{b*q*(d+.102)}{\beta*(d-.898)}\right| \tag{29}$$

$$X_{b.std} = \frac{X_b}{X_b+X_q+X_\beta+X_d+X_s} \tag{30}$$

$$X_{q.std} = \frac{X_q}{X_b+X_q+X_\beta+X_d+X_s} \tag{31}$$

$$X_{\beta.std} = \frac{X_\beta}{X_b+X_q+X_\beta+X_d+X_s} \tag{32}$$

$$X_{d.std} = \frac{X_d}{X_b+X_q+X_\beta+X_d+X_s} \tag{33}$$

$$X_{s.std} = \frac{X_s}{X_b+X_q+X_\beta+X_d+X_s} \tag{34}$$

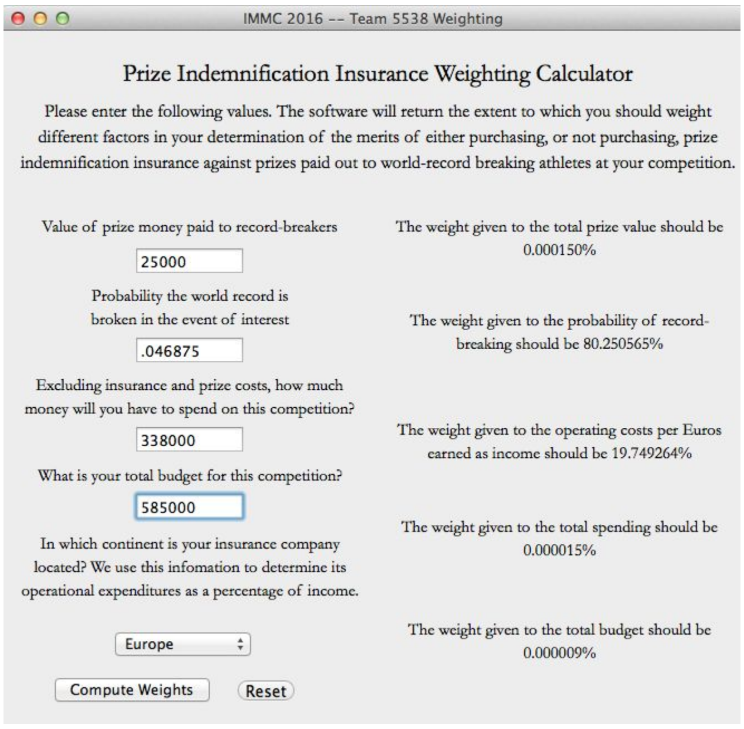
All of the weights computed above require a d value, so we calculated a corresponding value for each of the six continents. We first select a large insurance company based in Europe, North America, Asia, South America, Australia, and Africa. Based on Assumption 2, the operating cost per unit of income should remain relatively constant per continent, so we analyze the financial reports of one insurance company based in each continent. From their financial report, we found the company's operating costs excluding insurance related expenditures and divided it by the total revenue. Our results are summarized in the table below:

Continent	Europe	North Amer.	Asia	South Amer.	Australia	Africa
Company	Ageas	Allstate	AIA Group	Bradesco	IAG	MMI Holdings
d	0.1559	0.127	0.1564	0.1613	0.0783	0.1

Table 1. Operating costs to income ratio for the six main continents

Objective-C User-Facing Program #2: Relative Weights Calculator

Now that the model used to address question 4 is complete, we again develop a software program using Objective-C to automate computations with this model, as we did with question 3. The source code begins on page 35 of the Appendix. Again, this will allow us to rapidly verify our results and provides the race organizers a simple interface from which to enter inputs and obtain results. Since the issue in question is now specifically how to weight each factor in the insurance-purchasing decision, the software we develop here will address that issue directly. This program takes five inputs; in variable form, they are b, q, s, β, and d. Although the race organizers can easily supply the first four values themselves, they have no easy way of accessing d. We do the work for them by requesting the continent in which they wish to purchase insurance, then using a standard value of d for that continent from the table above, consistent with Assumption 2. Using equations 25-34, the program computes the five standardized weights. The results the program provides explain how much weight ought to be given to each factor in the organizing committee's decision-making analysis. We now provide an example of the question 4 weighting software, again using the values from the Zevenheuvelenloop. We obtain the following output:



IMMC 2016 -- Team 5538 Weighting

Prize Indemnification Insurance Weighting Calculator

Please enter the following values. The software will return the extent to which you should weight different factors in your determination of the merits of either purchasing, or not purchasing, prize indemnification insurance against prizes paid out to world-record breaking athletes at your competition.

Value of prize money paid to record-breakers: 25000

Probability the world record is broken in the event of interest: .046875

Excluding insurance and prize costs, how much money will you have to spend on this competition? 338000

What is your total budget for this competition? 585000

In which continent is your insurance company located? We use this infomation to determine its operational expenditures as a percentage of income. Europe

The weight given to the total prize value should be 0.000150%

The weight given to the probability of record-breaking should be 80.250565%

The weight given to the operating costs per Euros earned as income should be 19.749264%

The weight given to the total spending should be 0.000015%

The weight given to the total budget should be 0.000009%

Figure 7. The weighting calculator we have built uses our model to determine how much weight should be given to each factor when deciding whether or not to purchase insurance for a given event.

Notice that *q*, the probability that the record is broken, is given an overwhelming majority of the weight. This makes sense, since *q* determines how likely it is that a payout needs to happen in the first place. If *q* is always this high, we should largely base our decisions on *q*. In the next section, an extensive sensitivity analysis demonstrates that *q* does in fact carry the greatest weight in most circumstances. We can apply this result directly to the insurance question. If the organizers of a major athletics competition must choose between insuring the 50-meter freestyle and the long jump, for example, there is perhaps little question that they ought to insure the long jump, where the world's best talent has had less opportunity to try to break the record. This point is illustrated in the difference in the world record progression for the men's 50m freestyle, which is broken on average roughly every other year, and the world record progression for the men's long jump, which is only broken roughly every five years.[18] The record progression, in conjunction with the high weight allocated to *q*, in conjunction with the research on accessibility mentioned above, all suggest the organizers ought to give *q* the vast majority of the weight.

Sensitivity Analysis

Local Method:

Our sensitivity analysis is based on combining the one-factor-at-a-time approach, where we observe the effects of the variables of a multivariate function, in our case the weight functions, with partial derivatives to find the sensitivity of our weights around a local value which we fix as constant using real-world data. Using MATLAB, we take the partial derivatives at the real-world values, x^0, for all combinations of the five weight functions and their five constituent variables. Essentially, this method tests the sensitivity of the weighting expressions developed in equations 25-29 to a wide range of values.

$\left\|\frac{\partial xb}{\partial b}\right\|_{x0} = 4.5 * 10^{-9}$	$\left\|\frac{\partial xb}{\partial q}\right\|_{x0} = .0027$	$\left\|\frac{\partial xb}{\partial \beta}\right\|_{x0} = 4.8 * 10^{-8}$	$\left\|\frac{\partial xb}{\partial s}\right\|_{x0} = 4.8 * 10^{-8}$	$\left\|\frac{\partial xb}{\partial d}\right\|_{x0} = 8.6 * 10^{-5}$
$\left\|\frac{\partial xq}{\partial b}\right\|_{x0} = .0027$	$\left\|\frac{\partial xq}{\partial q}\right\|_{x0} = 1277.7$	$\left\|\frac{\partial xq}{\partial \beta}\right\|_{x0} = .0256$	$\left\|\frac{\partial xq}{\partial s}\right\|_{x0} = .0256$	$\left\|\frac{\partial xq}{\partial d}\right\|_{x0} = 46.06$
$\left\|\frac{\partial x\beta}{\partial b}\right\|_{x0} = 4.8 * 10^{-8}$	$\left\|\frac{\partial x\beta}{\partial q}\right\|_{x0} = .0256$	$\left\|\frac{\partial x\beta}{\partial \beta}\right\|_{x0} = 7.03 * 10^{-9}$	$\left\|\frac{\partial x\beta}{\partial s}\right\|_{x0} = 3.52 * 10^{-9}$	$\left\|\frac{\partial x\beta}{\partial d}\right\|_{x0} = .0062$
$\left\|\frac{\partial xs}{\partial b}\right\|_{x0} = 4.8 * 10^{-8}$	$\left\|\frac{\partial xs}{\partial q}\right\|_{x0} = .0256$	$\left\|\frac{\partial xs}{\partial \beta}\right\|_{x0} = 3.53 * 10^{-9}$	$\left\|\frac{\partial xs}{\partial s}\right\|_{x0} = 0$	$\left\|\frac{\partial xs}{\partial d}\right\|_{x0} = .0063$

$\left|\frac{\partial xd}{\partial b}\right|_{x^0} = 8.6 * 10^{-5}$ $\left|\frac{\partial xd}{\partial q}\right|_{x^0} = 46.06$ $\left|\frac{\partial xd}{\partial \beta}\right|_{x^0} = .0062$ $\left|\frac{\partial xd}{\partial s}\right|_{x^0} = .0063$ $\left|\frac{\partial xd}{\partial d}\right|_{x^0} = 13.97$

The partial derivative of one of the weight functions with respect to any relevant variable gives us the rate of change in financial risk if we were to alter that variable near the local value specified (x^0). As a result, this method gives us a good idea of how much the weight of a factor changes with a change in one of the variables.

Most of these values are relatively small, but some notable relationships are $\left|\frac{\partial xq}{\partial q}\right|_{x^0} = 1277.7$ and $\left|\frac{\partial xq}{\partial d}\right|_{x^0} = 46.06$. This means that the weight of the probability of a record being broken is very sensitive to changes in that probability itself and also to the ratio *d* of the insurance company's operating expenses to its income.

Scatter Plot Approach:

The downside of using the partial derivative in sensitivity analysis is that it limits the scope to which the sensitivity of the outputs may be affected by the uncertainty in the inputs, since it is only accurate close to x^0, the only place where we have real-world values. To account for a larger scope of values for each of the variables, we use the scatter plot approach, which can incorporate limitless variability in the input variables. Below are the 25 scatter plots, pairing each one of the five weight functions with each of the five variables.

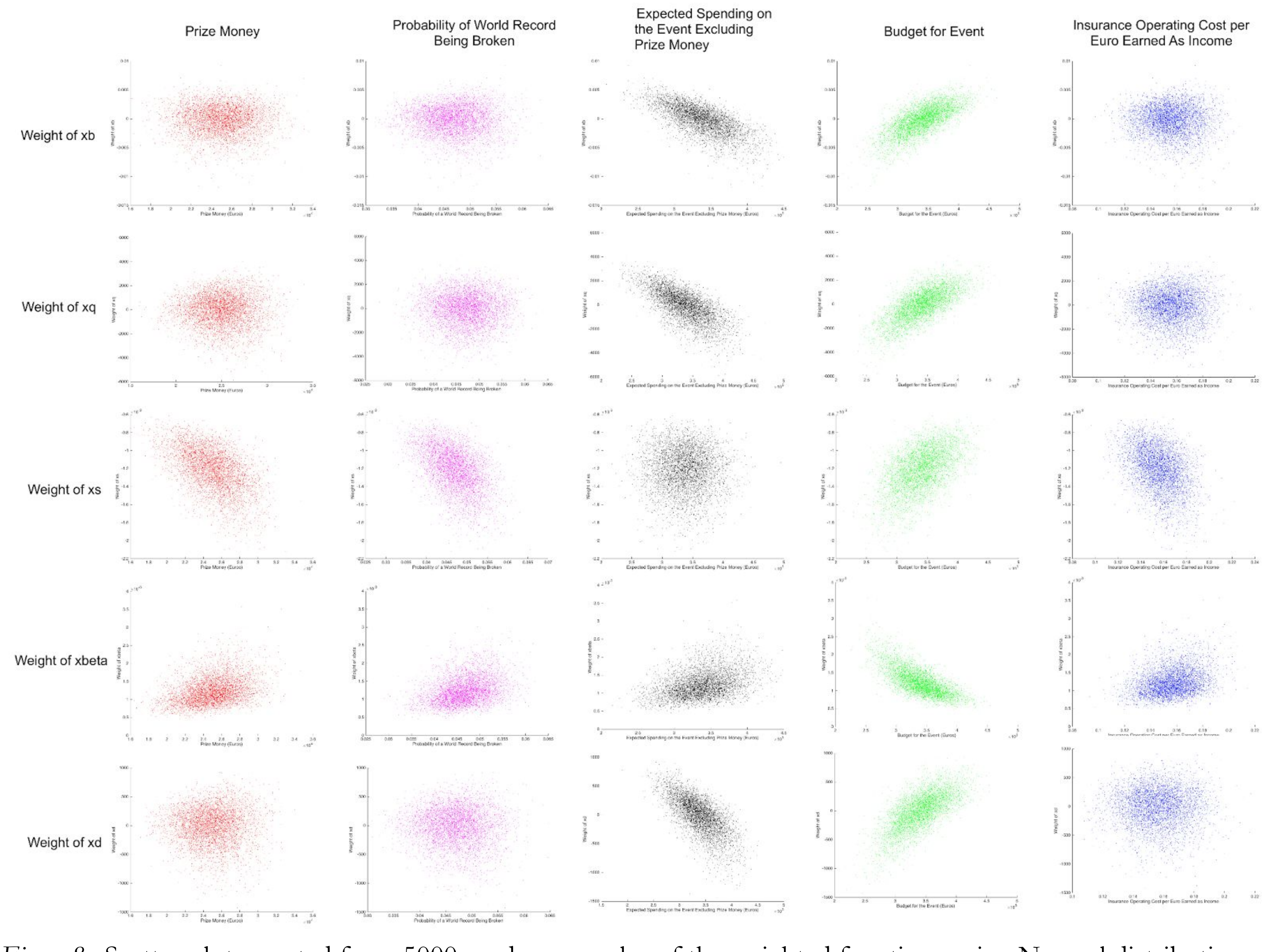


Figure 8. Scatter plots created from 5000 random samples of the weighted functions using Normal distributions for each of the variables; the labels down the left denotes the weight function that is being plotted and the labels across the top denote the variable that the function is plotted against; full graphs on page 5 of the Appendix

The scatter plots are produced by taking samples of sets of the five variables from five separate Normal distributions centered around their real-world values with arbitrary standard deviations (we choose $\mu/10$, but the standard deviation we choose is inconsequential because we are concerned with the effect on the value of the function, which will show the same effect regardless of the spread of the independent variables). The y-axis of the graph is the label of the weight functions down the left and the x-axis of the graph is the label of the variables across the top. If the function is not sensitive to the variable, one can expect to see an upright elliptical cluster, as changing the variable as an independent value does not significantly alter the distribution. However, if the shape of the cluster strays off from this shape to look more like an ellipse on an oblique axis, the function is significantly sensitive to the variable.

For example, the effect of the amount of prize money on the weight allocated to x_b is very small, as the scatter plot (in red) shows an upright ellipse. However, the weight of x_b is greatly affected by the budget for the event, whose scatter plot (in green) shows an ellipse on a tilted axis. Full graphs begin on page 5 of the Appendix and the MATLAB code used to generate them may be found on page 26 of the Appendix.

Further analysis on these scatter plots yields correlation coefficients that quantify the sensitivity of a function to each of its constituent variables. Below is a table for the correlation coefficients as they correspond to the scatter plots above. The higher the absolute value of the correlation coefficient, the more influential the variable is on the function.

Correlation between	**b**	**q**	**s**	β	**d**
x_b	-.0033	-.0245	-.6889	.6901	.0155
x_q	-.0087	.0095	-.6869	.7108	-.0169
x_s	-.5189	-.5207	.0033	.5274	-.3851
x_β	.3391	.3353	.3441	-.7116	.2842
x_d	-.0119	-.0314	-.6945	.6975	.0092

Table 2. Correlation coefficients quantifying the sensitivity of the weight functions in the above scatter plots from uncertainty in the input variables

Note that a given variable is not necessarily the biggest influencer of the weight given to itself: a change in *b*, for example, does not change the weight given to *b* nearly as much as a change in *s* or β.

Results of the local sensitivity analysis show us which variables have the greatest direct change on the value of each of our multivariate weight functions. Results of the scatter plot approach show us how much a change in one variable can affect the spread of samples of the weight function. Our thorough sensitivity analysis shows us which variables our model is more sensitive towards and which ones do not affect our outcome as much. The scatter plot method shows us that our model is most sensitive to the spending on an event, the budget of an event, and the probability that a world record will be broken, but the overall sensitivity values are extremely low, meaning that our model can function accurately even under extreme conditions.

<u>Part 5: Determination of Which Events to Insure</u>

We now consider the example case of a track and field competition holding 40 events, 20 for men and 20 of the same events for women. The organizing committee planning the competition has a fixed budget and fixed amount of spending (exclusive of insurance premiums and bonus payment costs), but it can choose how to allocate that budget toward insurance purchases. The decision to buy insurance varies significantly by event, on the basis of factors such as the probability the record will be broken in any given event, and the magnitude of the payout that would be paid to the record-breaker. The organizing committee must determine whether or not it seems to be in the financial interest of the event to purchase insurance for any particular event, taking into account the factors mentioned above.

We already have a computer program (User-Facing Program #1; see Part 3) that uses our expressions for financial exposure to find the percent difference between the financial exposure associated with self insuring and that associated with purchasing insurance for each event. We prioritize which events to insure based on the calculated percent difference of financial exposure. This is summarized in the following decision scheme:

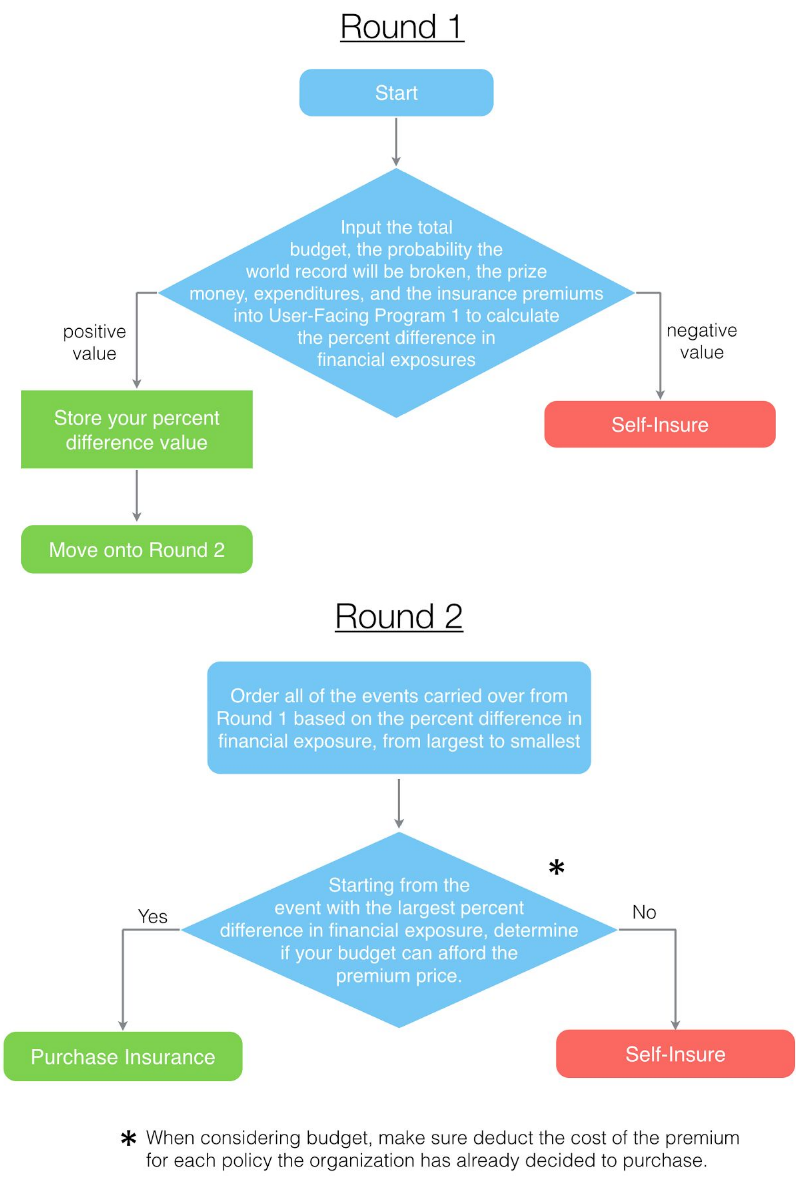

Round 1
Start
Input the total budget, the probability the world record will be broken, the prize money, expenditures, and the insurance premiums into User-Facing Program 1 to calculate the percent difference in financial exposures
positive value
negative value
Store your percent difference value
Self-Insure
Move onto Round 2
Round 2
Order all of the events carried over from Round 1 based on the percent difference in financial exposure, from largest to smallest
Starting from the event with the largest percent difference in financial exposure, determine if your budget can afford the premium price.
*
Yes
No
Purchase Insurance
Self-Insure
* When considering budget, make sure deduct the cost of the premium for each policy the organization has already decided to purchase.

Figure 9. Decision scheme for 40 track and field events with binary outcomes (insurance vs. self-insurance)

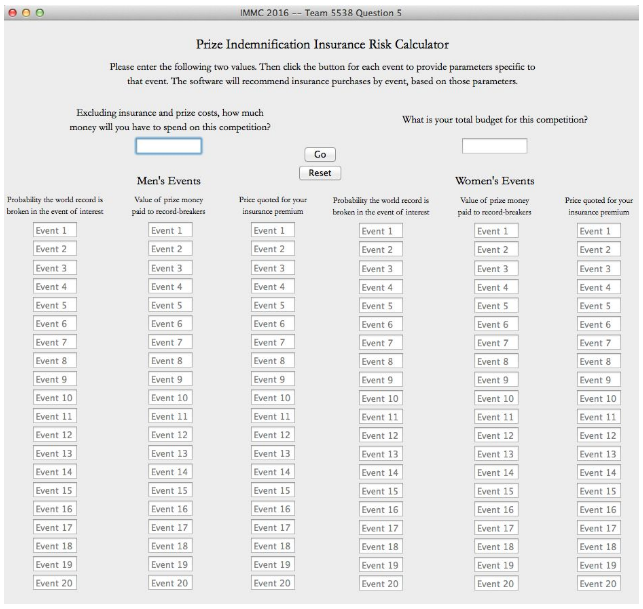
Objective-C User-Facing Program #3: Decision-Scheme Automation for 40 Events

To automate the Round 1 and 2 decision schemes, we create our third and final program, which is our most comprehensive computational treatment of the insurance problem. As always, the Objective-C code may be found in the Appendix, starting on page 42. The code is very heavily commented; we have included extensive "comments" that are ignored by the compiler but show the viewer what the code is doing. The program starts out with a very long list of variable definitions and conversions, but the logic of the program (where the bulk of the commenting appears) may be found on pages 68-73 of the Appendix. Using the decision scheme described above, the program builds upon all the models we have built so far, analyzing risk, calculating excess budget amounts, and eventually making recommendations. The program automates the entire procedure, including the risk calculations previously processed by User-Facing Program #1, so that the user may simply enter the inputs and instantly (<600 ms) obtain results. Inputs required include the total budget for the meet, the total expenditures excluding spending related to insurance or paying the prize money, and the probability the world record will be broken, the prize money, and the cost of the insurance premium for each event. The program essentially follows the following four steps:

1. Calculate the financial exposure entailed with each of the two scenarios (purchasing insurance or not purchasing insurance). Compute the relative risk and determine which of the two scenarios is more financially risky. All events for which it is riskier to purchase insurance are rendered nonviable and taken out of consideration, so the program recommends not purchasing insurance for these events.

2. Viable events continue into the next round. The program orders the events from greatest percent difference between the financial exposure of self-insurance and that of purchasing insurance to least (those events where it is most urgent to purchase insurance to those events where it is least urgent to purchase insurance). Starting with the first event, the program determines whether the organizing committee can afford the insurance premium for that event (we assume they know the value of the premium from their insurance quote, and we also have equations 3-7 to estimate that premium, which we use to generate premium values for the test case).
3. If the organizing committee cannot afford the premium, they will be forced to take the risk of self-insurance (any event that is still viable by round 2 has a greater non-insurance risk than insurance risk, but if the premium is not financially accessible, the committee does not have a choice). The program will recommend not purchasing insurance for that event. If the organizing committee can afford the premium, the program will recommend that insurance be purchased. The amount of the premium for this plan is deducted from the amount of capital the organization has at its disposal. This process is repeated until there is no disposable capital left. Even if later insurance premiums are cheaper, they are for events where the committee faces less risk with self-insurance, so they should not insure.
4. The program informs the user of its recommendations; specifically, where (for which events) insurance should be purchased and where it should not.

Using real-world data from the same source as that provided to us for the Zevenheuvelenloop[18], we generate a test case of 40 track and field events, with 20 for men and 20 for women.



IMMC 2016 -- Team 5538 Question 5

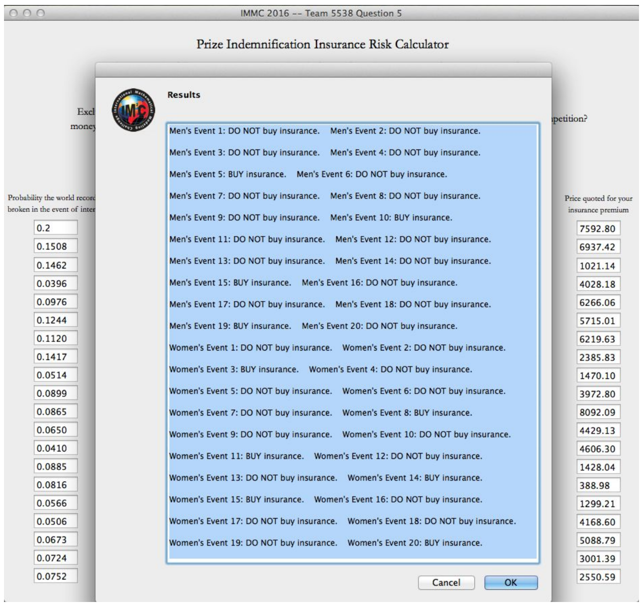
Prize Indemnification Insurance Risk Calculator

Please enter the following two values. Then click the button for each event to provide parameters specific to that event. The software will recommend insurance purchases by event, based on those parameters.

Excluding insurance and prize costs, how much money will you have to spend on this competition?

What is your total budget for this competition?

Men's Events			Women's Events		
Probability the world record is broken in the event of interest	Value of prize money paid to record-breakers	Price quoted for your insurance premium	Probability the world record is broken in the event of interest	Value of prize money paid to record-breakers	Price quoted for your insurance premium
Event 1	Event 1	Event 1	Event 1	Event 1	Event 1
Event 2	Event 2	Event 2	Event 2	Event 2	Event 2
Event 3	Event 3	Event 3	Event 3	Event 3	Event 3
Event 4	Event 4	Event 4	Event 4	Event 4	Event 4
Event 5	Event 5	Event 5	Event 5	Event 5	Event 5
Event 6	Event 6	Event 6	Event 6	Event 6	Event 6
Event 7	Event 7	Event 7	Event 7	Event 7	Event 7
Event 8	Event 8	Event 8	Event 8	Event 8	Event 8
Event 9	Event 9	Event 9	Event 9	Event 9	Event 9
Event 10	Event 10	Event 10	Event 10	Event 10	Event 10
Event 11	Event 11	Event 11	Event 11	Event 11	Event 11
Event 12	Event 12	Event 12	Event 12	Event 12	Event 12
Event 13	Event 13	Event 13	Event 13	Event 13	Event 13
Event 14	Event 14	Event 14	Event 14	Event 14	Event 14
Event 15	Event 15	Event 15	Event 15	Event 15	Event 15
Event 16	Event 16	Event 16	Event 16	Event 16	Event 16
Event 17	Event 17	Event 17	Event 17	Event 17	Event 17
Event 18	Event 18	Event 18	Event 18	Event 18	Event 18
Event 19	Event 19	Event 19	Event 19	Event 19	Event 19
Event 20	Event 20	Event 20	Event 20	Event 20	Event 20



IMMC 2016 -- Team 5538 Question 5

Prize Indemnification Insurance Risk Calculator

Results

```
Men's Event 1: DO NOT buy insurance.    Men's Event 2: DO NOT buy insurance.
Men's Event 3: DO NOT buy insurance.    Men's Event 4: DO NOT buy insurance.
Men's Event 5: BUY insurance.    Men's Event 6: DO NOT buy insurance.
Men's Event 7: DO NOT buy insurance.    Men's Event 8: DO NOT buy insurance.
Men's Event 9: DO NOT buy insurance.    Men's Event 10: BUY insurance.
Men's Event 11: DO NOT buy insurance.    Men's Event 12: DO NOT buy insurance.
Men's Event 13: DO NOT buy insurance.    Men's Event 14: DO NOT buy insurance.
Men's Event 15: BUY insurance.    Men's Event 16: DO NOT buy insurance.
Men's Event 17: DO NOT buy insurance.    Men's Event 18: DO NOT buy insurance.
Men's Event 19: BUY insurance.    Men's Event 20: DO NOT buy insurance.
Women's Event 1: DO NOT buy insurance.    Women's Event 2: DO NOT buy insurance.
Women's Event 3: BUY insurance.    Women's Event 4: DO NOT buy insurance.
Women's Event 5: DO NOT buy insurance.    Women's Event 6: DO NOT buy insurance.
Women's Event 7: DO NOT buy insurance.    Women's Event 8: BUY insurance.
Women's Event 9: DO NOT buy insurance.    Women's Event 10: DO NOT buy insurance.
Women's Event 11: BUY insurance.    Women's Event 12: DO NOT buy insurance.
Women's Event 13: DO NOT buy insurance.    Women's Event 14: BUY insurance.
Women's Event 15: BUY insurance.    Women's Event 16: DO NOT buy insurance.
Women's Event 17: DO NOT buy insurance.    Women's Event 18: DO NOT buy insurance.
Women's Event 19: DO NOT buy insurance.    Women's Event 20: BUY insurance.
```

Probability the world record broken in the event of inter	Price quoted for your insurance premium
0.2	7592.80
0.1508	6937.42
0.1462	1021.14
0.0396	4028.18
0.0976	6266.06
0.1244	5715.01
0.1120	6219.63
0.1417	2385.83
0.0514	1470.10
0.0899	3972.80
0.0865	8092.09
0.0650	4429.13
0.0410	4606.30
0.0885	1428.04
0.0816	388.98
0.0566	1299.21
0.0506	4168.60
0.0673	5088.79
0.0724	3001.39
0.0752	2550.59

Figure 10. Our final program uses our risk and weight models, along with our decision scheme, to choose events for which insurance should be purchased based on inputs provided by the user (here, our test case).

Note: To find the probability of breaking the world record in each event, we obtained data describing the number of world records broken per event. We calculated the average probability of breaking the world record per year by dividing the number of times the world record was broken by the number of years event times were counted. Ideally, we would have calculated the probability of the world record being broken based on the number races run rather than by year, however, we are only interested in the *relative* world record breaking probabilities of each event: we only want to know if, for example, it is more likely for a steeplechase world record to be broken than a 100m sprint world record. This would mean that the prize money for steeplechase would be more likely to be paid out than that of the 100m sprint and would possibly make it more worthwhile to purchase insurance for the steeplechase than the 100m sprint.

Deductibles

Our alternative model takes deductibles into account and uses the equation 19 from Part 3 that gives us the financial exposure for the organizing committee in terms of the amount of the deductible.

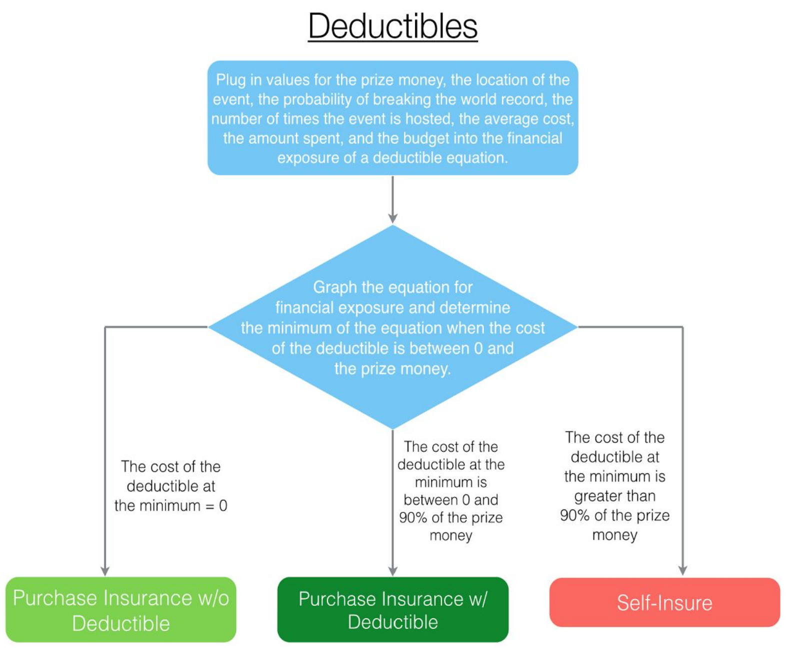

Deductibles
Plug in values for the prize money, the location of the event, the probability of breaking the world record, the number of times the event is hosted, the average cost, the amount spent, and the budget into the financial exposure of a deductible equation.
Graph the equation for financial exposure and determine the minimum of the equation when the cost of the deductible is between 0 and the prize money.
The cost of the deductible at the minimum = 0
The cost of the deductible at the minimum is between 0 and 90% of the prize money
The cost of the deductible at the minimum is greater than 90% of the prize money
Purchase Insurance w/o Deductible
Purchase Insurance w/ Deductible
Self-Insure

Figure 11. Decision scheme for 40 track and field events with three outcomes

The organizers will first input the necessary known values about each event into MATLAB Program 6 (see page 31 of the Appendix for the software code). The software will automatically find the minimum of the equation for each event with the appropriate constants and return a suggested course of action for the organizers, whether to purchase full insurance, purchase insurance with a suggested deductible amount, or self-insure. Because of the shape of the equation with an asymptote at the prize value (b), it will be impossible for the risk to be lowest when the deductible amount is the same as the prize value (see the graphs in Figure 6). While the graphs indicate that it may never be more advantageous to self-insure, the x-value of the minimum of the graph could be so close to the prize value that buying an insurance policy with such a high deductible amount will be practically the same as self-insuring. We decided for the cut-off point to be 90% of the prize value. A small example affirms the validity of our cut-off point. If the prize value is $50,000 and the suggested deductible is 90%, or $45,000, then the insurance premium will be around 1.35 * (average cost of the bonus minus the deductible) -- see equations 16 and 18 -- and the amount the insurance company will need to pay $5000. If we set the hypothetical probability of breaking the world record to be .3, the premium is approximately:

$$1.35 * (b - u) * q = 1.35 * 5{,}000 * .3 = \$2025$$

Therefore, the total amount the organizer needs to pay if the world record is broken is $47,025, which is sufficiently close to the amount they need to pay had they not bought insurance ($50,000, a difference of 5.95%) that it is generally not worth the effort (and cost, which could easily exceed $3000) of researching and buying insurance.

Examples of how the deductible scheme will work for a single event can be found in Part 3.

Strengths and Weaknesses

We are confident in our model's ability to make the best possible recommendation as to whether or not an event organization committee should purchase insurance for any given event. A challenge presented in this problem is the fact that, in the long run, purchasing insurance will always net a negative financial outcome for the committee. Models focused solely on long-term cash flow will thus always recommend that the committee self-insure in all cases. Our model overcomes this challenge by quantifying the risk of financial exposure, the reason for which people are more willing to sustain a small financial loss than risk losing a large loss of money. Our model takes into account the financial exposure of buying insurance (potential financial loss if no world record is broken) and self-insurance (potential financial loss if world record is broken).

Another important strength of our solution is the deductible model that allows the race organizer the additional option of purchasing insurance with deductible provisions that allow them to lower their premiums. Although the problem only asked for us to distinguish between buying insurance and self-insuring, adding on the option of deductibles greatly contributes to the real world applicability of our paper since most insurers provide insurance with deductibles as an option, and this can be the best option for the organizers, as shown in Part 3. Our model not only suggests to the organizer whether they should purchase full insurance, insurance with deductibles, or self-insure, but also suggests a deductible amount that yields the lowest financial exposure for the organizer if deductible-based insurance is indeed the most financially viable option.

An important strength to our model is the thoroughness of our sensitivity analysis. We have utilized both local and global methods to find and display how sensitive the weight of each factor is to each of the variables in our model. The 25 scatter plots included in the sensitivity analysis show exactly what impact a variable has on the random samples, and this effect is quantified with the correlation coefficient. This shows us to place the most emphasis on *q*, an assertion reaffirmed both by 1) our software in a quantitative way, and 2) the literature cited above, in turn allowing us to provide a concrete and simple piece of advice to the organizing committee: when deciding whether or not to insure an event, if there is one factor you can consider, consider *q*, the probability a record is set.

Finally, our suite of user-friendly software programs follows a reasonable decision-making scheme that assists the committee in buying the most insurance it can while prioritizing those events in most urgent need of insurance. Our software allows the user to easily compute the results from our models, and the models are fully developed and can be easily and immediately implemented by a real committee with no further revision; it accurately, clearly, and reliably explains exactly how they should allocate the limited funds they have with which to purchase insurance.

A potential weakness in our paper is our second Assumption that states that insurance companies have similar operating costs per currency (dollar/euro) earned as income across a continent. While we calculated distinct rates per continent to have better approximations of the real operating costs to income ratio, our estimates are still not 100% accurate since these numbers are at least somewhat unique to each individual insurance broker. We still believe, however, that the variation will not be so great as to disaffirm the outcome of our model because in order for an insurance company to offer reasonable premiums and still make a profit, the operating cost to income ratio must be set within a small range (the ratios for all of the continents, with the exception of Australia, lay within 4% of each other). A second weakness of our paper and model is that we are significantly hampered in terms of our ability to accurately quantify *q*, the probability that a world record is broken. Although our sensitivity analysis confirms that *q* is extremely important to our calculations, determination of that value is extremely difficult. The best we can do is use record progressions similar to that presented in the final page of the problem statement (since we always compare events relative to each other and insurance premiums are computed based on *q*, systematic inflation or deflation of those rates due to better or worse competition is not a problem), but it is very difficult to know whether these values will hold for any given race.

<u>Conclusion</u>

We approach this problem by developing a relative financial exposure model that compares the relative financial risks of purchasing insurance versus self-insuring. Taking into account the prize money to budget ratio for each event, the model accounts for the possibility of a payout the organizer cannot afford. Multiple criteria (prize money, the probability of breaking the world record, the budget of the race, and the number of races held) are taken into account. This model is of great use to any event organizer with potential need for prize indemnity insurance because its quantification of financial exposure allows for direct comparison and points to a clear suggested course of action. The programs we develop, which process inputs through our models and then output the suggested insurance policy also make our model extremely easy to use and thus applicable on a large scale in the real world. Our additional model offering the possibility of an insurance policy with deductibles provides more realistic choices for the race organizer that could lower financial exposure even more than the binary option of buying insurance or self-insuring. We designed a program to do this that reduces operational difficulties associated with its implementation and allows for widespread use. We are excited to offer robust models whose results can be integrated and automated to produce the best possible advice for race organizing committees.

Works Cited

1. Ageas, S.A. "Ageas Annual Report 2013." *Ageas.com*. Ageas, S.A., 19 Mar. 2014. Web. 26 Mar. 2016. <http://ageas.com/sites/default/files/AFS_2013_UK_Final.pdf>.
2. Allstate Corporation. "The Allstate Corporation Notice of 2014 Annual Meeting, Proxy Statement and 2013 Annual Report." *Allstate.com*. The Allstate Corporation, 7 Apr. 2014. Web. 30 Mar. 2016. <https://www.allstate.com/resources/allstate/attachments/annual-report/allstate-2014-am-materials.pdf>.
3. Apple, Inc. *Xcode*. Vers. 5. Cupertino: Apple, Inc., 2013. Computer software.
4. Banco Bradesco SA. "Bradesco 2014 Annual Report." *Bradescori.com.br*. Bradesco SA, 2014. Web. 30 Mar. 2016. <https://www.bradescori.com.br/site/conteudo/informacoes-financeiras/relatorios-anuais.aspx?secaoId=811&idiomaId=2>.
5. Forbes. "The World's Biggest Public Companies: AIA Group." *Forbes.com*. Forbes Magazine, May 2015. Web. 30 Mar. 2016. <http://www.forbes.com/companies/aia-group/>.
6. Insurance Australia Group. "The Numbers: Insurance Australia Group Annual Report 2014." *Iag.com.au*. Insurance Australia Group Limited, 19 Aug. 2014. Web. 30 Mar. 2016. <http://www.iag.com.au/sites/default/files/Documents/Results%20%26%20reports/FY14_RR_IAG_annual_report_2014_20140819.pdf>.
7. Mathworks. *MATLAB*. Vers. 2015b. Natick, MA: Mathworks, 2015. Computer software.

8. MMI Holdings. "MMI Holdings Integrated Report 2014." *Mmiholdingsintegratedreport2014.com*. MMI Holdings Ltd Group, 2014. Web. 30 Mar. 2016. <http://mmiholdingsintegratedreport2014.com/wp-content/uploads/pdf/MMI_IR_2014_Full.pdf>.

9. Moynihan, Shawn. "Top 100 P&C Insurance Companies: Q1 Comparisons, Ranked by Net Premiums Written." *Propertycasualty360.com*. ALM Media, LLC, 12 June 2014. Web. 27 Mar. 2016. <http://www.propertycasualty360.com/2014/06/12/top-100-pc-insurance-companies-q1-comparisons-rank?slreturn=1459124584>.

10. Mulkeen, John. "BOLT AGAIN, AND AGAIN! 19.19 WORLD RECORD IN BERLIN." *International Association of Athletics Federations*. IAAF, 20 Aug. 2009. Web. 30 Mar. 2016.

11. Pruijsen, Klaas. "Zevenheuvelenloop: Effectmeting Nijmeegse Evenementenmonitor." *Nijmegen.nl*. Gemeente Nijmegen Afdeling Onderzoek En Statistiek, Feb. 2009. Web. 27 Mar. 2016. <http://www.nijmegen.nl/rapportenzoeker/Docs/Effectmeting_7heuvelenloop_2008_feb2009.pdf>.

12. Silver, Nate. "Which Records Get Shattered?" *The New York Times*. The New York Times, 28 July 2012. Web. 26 Mar. 2016. <http://www.nytimes.com/2012/07/29/sunday-review/why-olympic-records-are-broken-or-not.html?_r=0>.

13. Sohn, Emily. "How Do Olympians Keep Getting Better?" *News.discovery.com*. Discovery Communications, LLC, 9 Aug. 2012. Web. 26 Mar. 2016. <http://news.discovery.com/adventure/extreme-sports/athletes-olympic-records-120809.htm>.

14. Statista. "Leading Global Insurance Companies by Revenue 2014." *Statista*. Statista, Inc., n.d. Web. 26 Mar. 2016. <http://www.statista.com/statistics/185746/revenue-of-the-leading-global-insurance-companies/>.

15. USA Track & Field. "USA Track & Field - Prize Money (Indoor Events)." *Usatf.org*. USA Track & Field, Inc., 2012. Web. 30 Mar. 2016. <http://www.usatf.org/Events---Calendar/2012/IAAF-World-Indoor-Championships/Athlete-Info/Prize-Money.aspx>.

16. USA Track & Field. "USA Track & Field - Prize Money (Outdoor Events)." *Usatf.org*. USA Track & Field, Inc., 2013. Web. 30 Mar. 2016. <http://www.usatf.org/Events---Calendar/2013/IAAF-World-Championships-in-Athletics/Athlete-Info/Prize-Money.aspx>.

17. Wall Street Journal. "AIA Group Ltd. 1299 (Hong Kong)." *Quotes.wsj.com*. Dow Jones & Company, Inc., 30 Mar. 2016. Web. 30 Mar. 2016. <http://quotes.wsj.com/HK/XHKG/1299/financials/annual/income-statement>.

18. Wikimedia Foundation. "List of World Records in Athletics." *Wikipedia*. Wikimedia Foundation, 20 Mar. 2016. Web. 30 Mar. 2016.

19. World Bank. "Real Interest Rates." *Data.worldbank.org*. World Bank Group, n.d. Web. 26 Mar. 2016. <http://data.worldbank.org/indicator/FR.INR.RINR>.

20. Yahoo! Finance. "Industry Summary." *Biz.yahoo.com*. Yahoo! Inc., 2016. Web. 26 Mar. 2016. <https://biz.yahoo.com/p/sum_qpmd.html>.

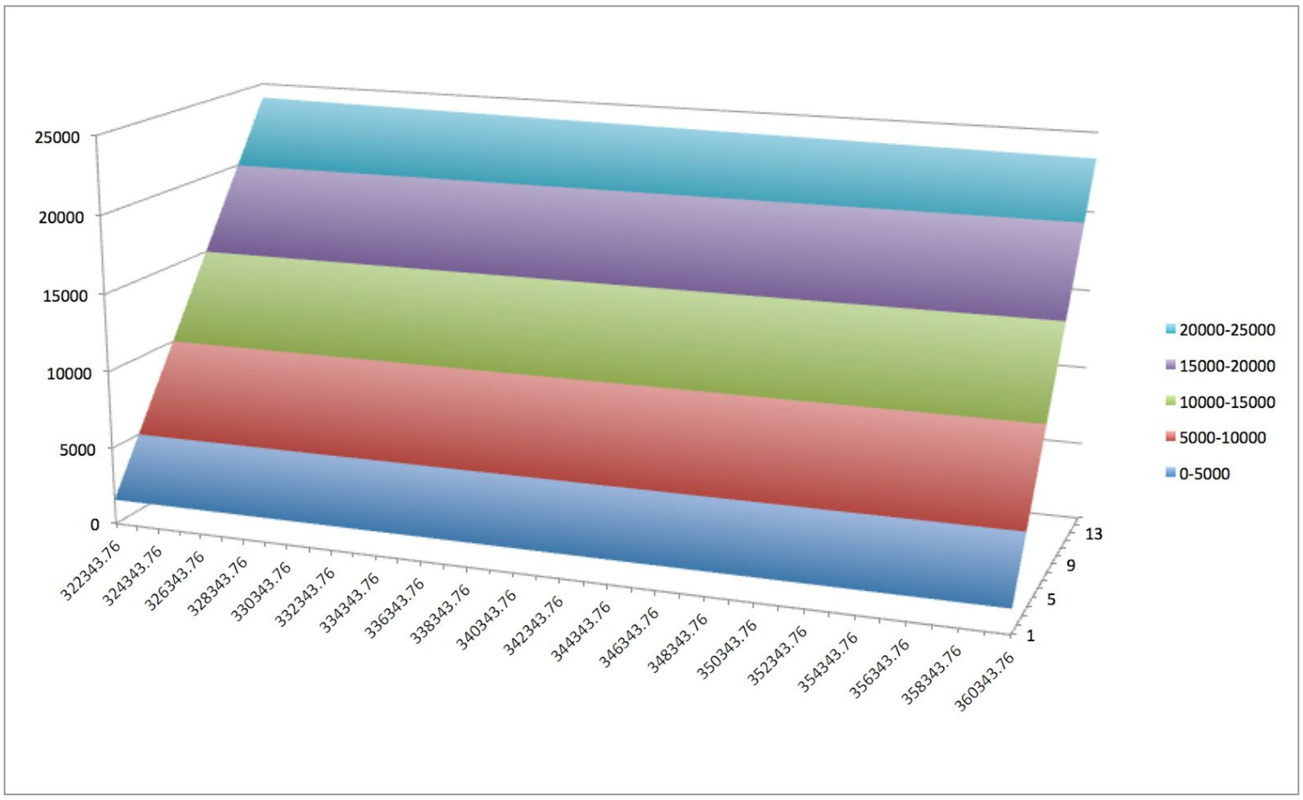
21. Zevenheuvelenloop. "Inschrijving 2016 Vanaf 1 Juni." *Scholten Awater Zevenheuvelenloop*. De Scholten Awater Zevenheuvelenloop Wordt Georganiseerd Door Stichting Zevenheuvelenloop, 2015. Web. 30 Mar. 2016. <http://www.zevenheuvelenloop.nl/deelnemers/zevenheuvelenloop/inschrijven-15-km>.

Appendix

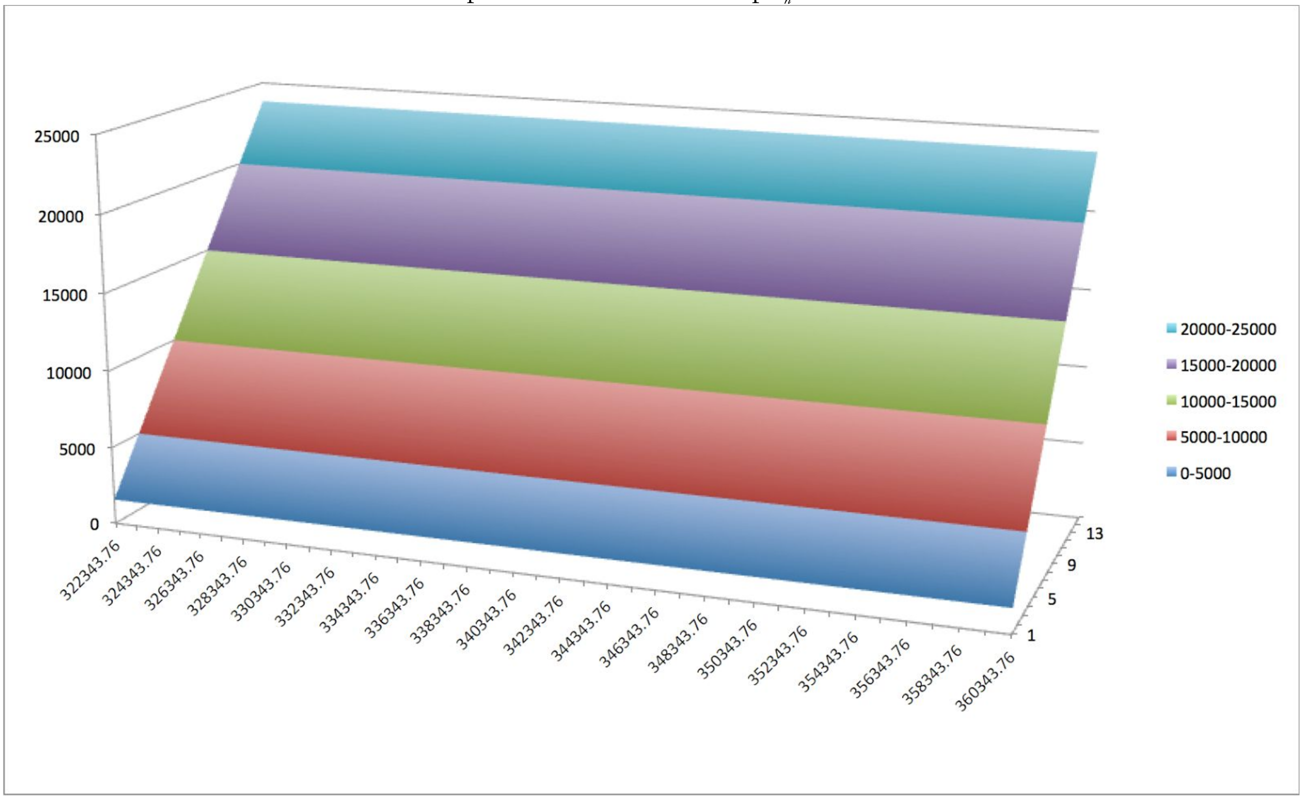
Table of Contents

Graphs and Displays

Question 3:

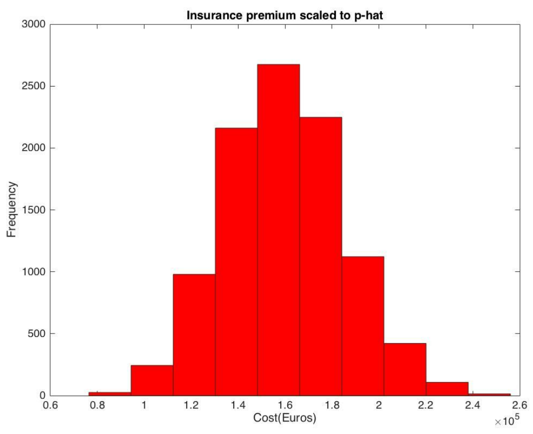


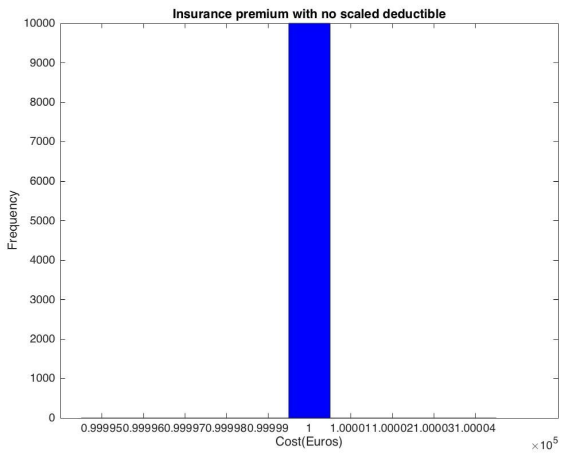
Graph of Zevenheuvelenloop r_n values



Graph of Zevenheuvelenloop r_i values

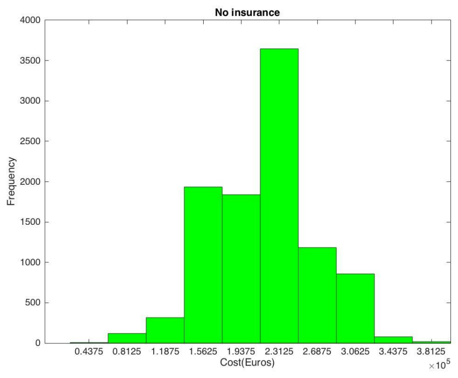
Graphs from MATLAB Program 2:

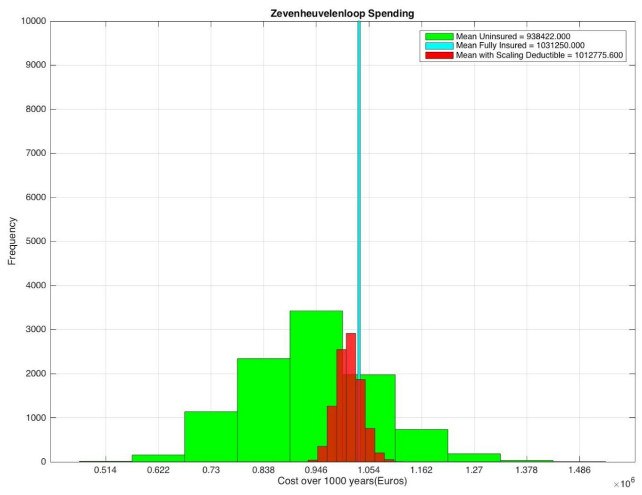




95% Confidence Interval of the Cost of 100 Zevenheuvelenloops with Deductible: (83750, 102780)

95% Confidence Interval of the Cost of 100 Zevenheuvelenloops with Full Insurance: (100000, 100000)

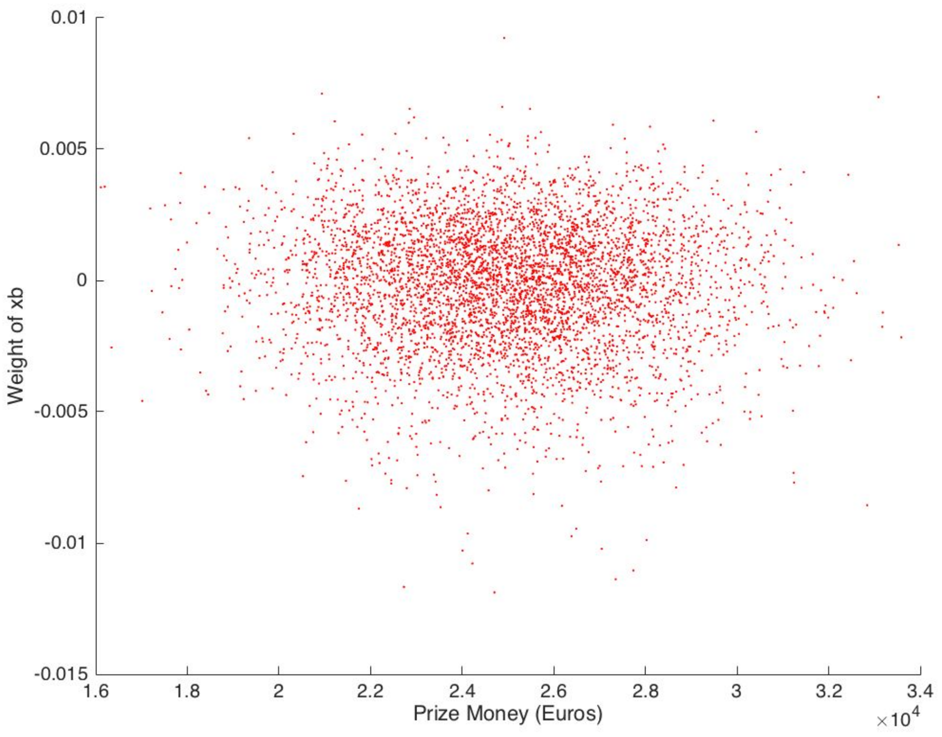




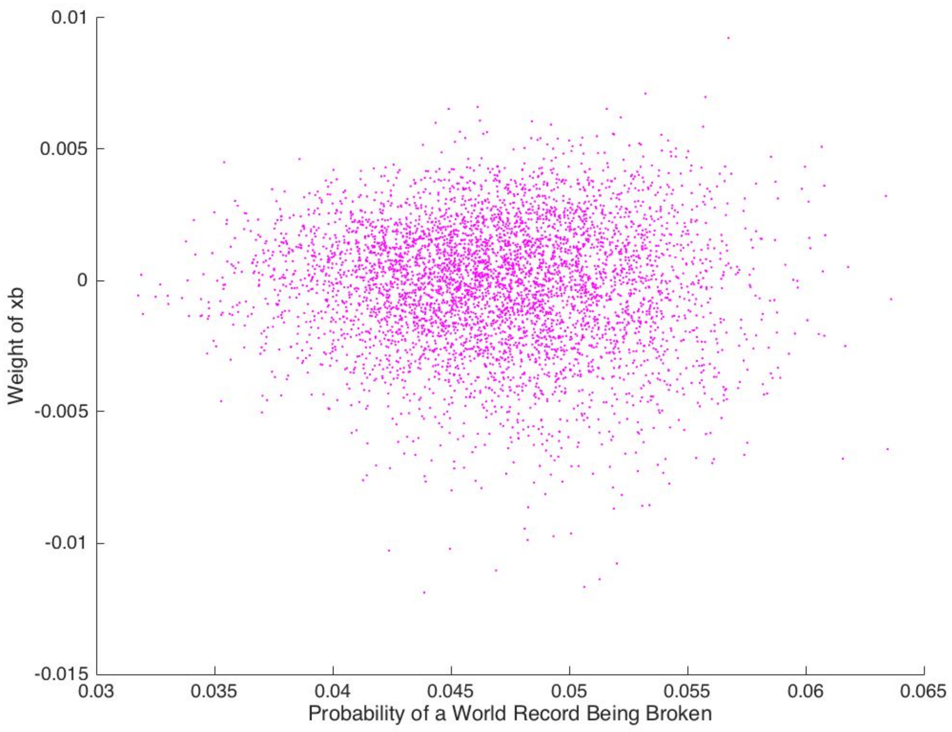
95% Confidence Interval of the Cost of 100 Zevenheuvelenloops with No Insurance: (50000, 100000)

A histogram of the cost of running the Zevenheuvelenloop for 1000 years with different insurance plans

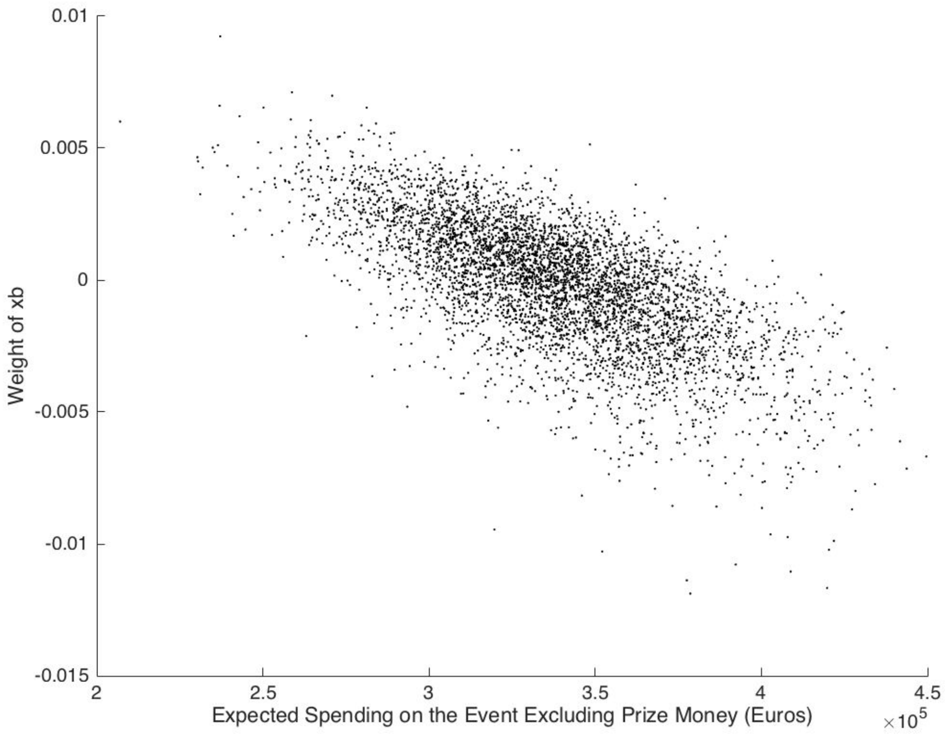
Question 4:
Graphs from MATLAB Program 4:



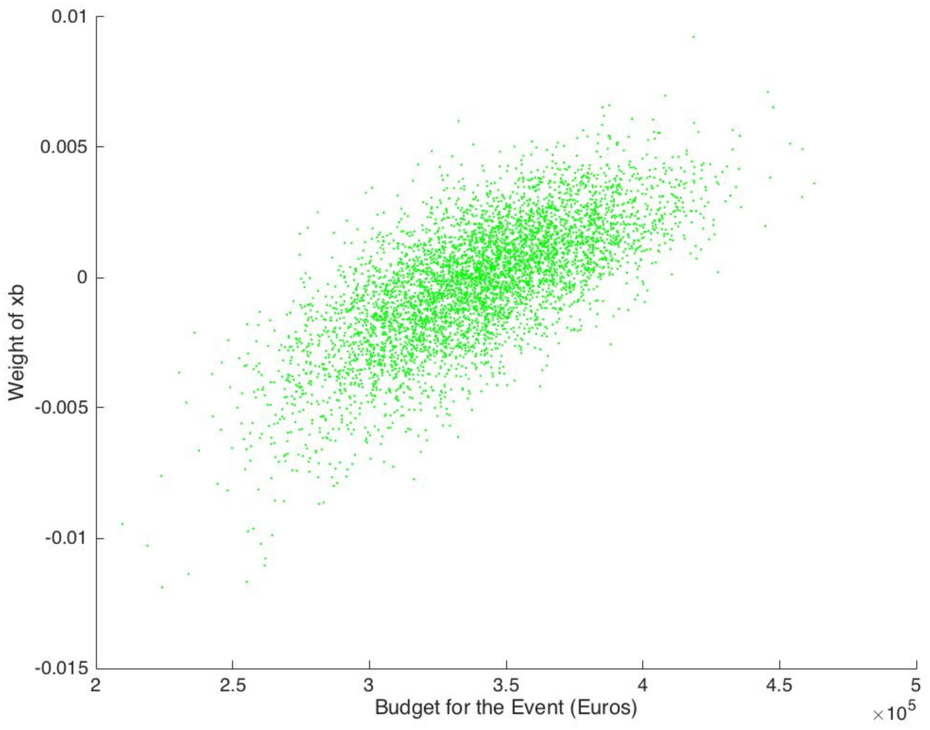
Correlation Coefficient: r=-.0033



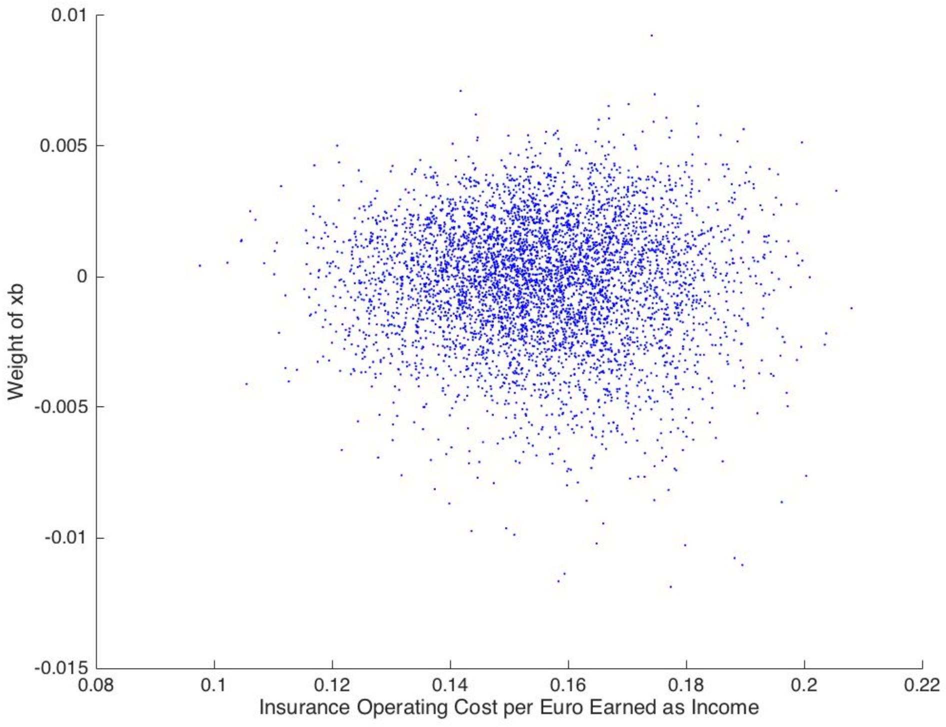
r=-.0245


0.01
0.005
0
-0.005
-0.01
-0.015
Weight of xb
2
2.5
3
3.5
4
4.5
Expected Spending on the Event Excluding Prize Money (Euros)
$\times 10^5$

r=-.6889

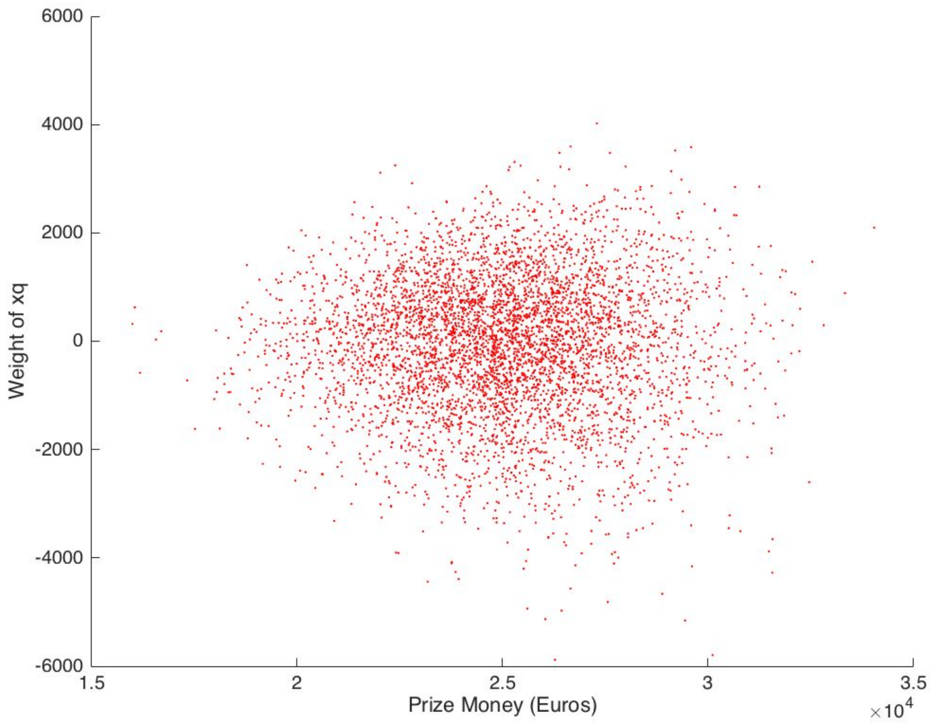

0.01
0.005
0
-0.005
-0.01
-0.015
Weight of xb
2
2.5
3
3.5
4
4.5
5
Budget for the Event (Euros)
$\times 10^5$

r=.6901



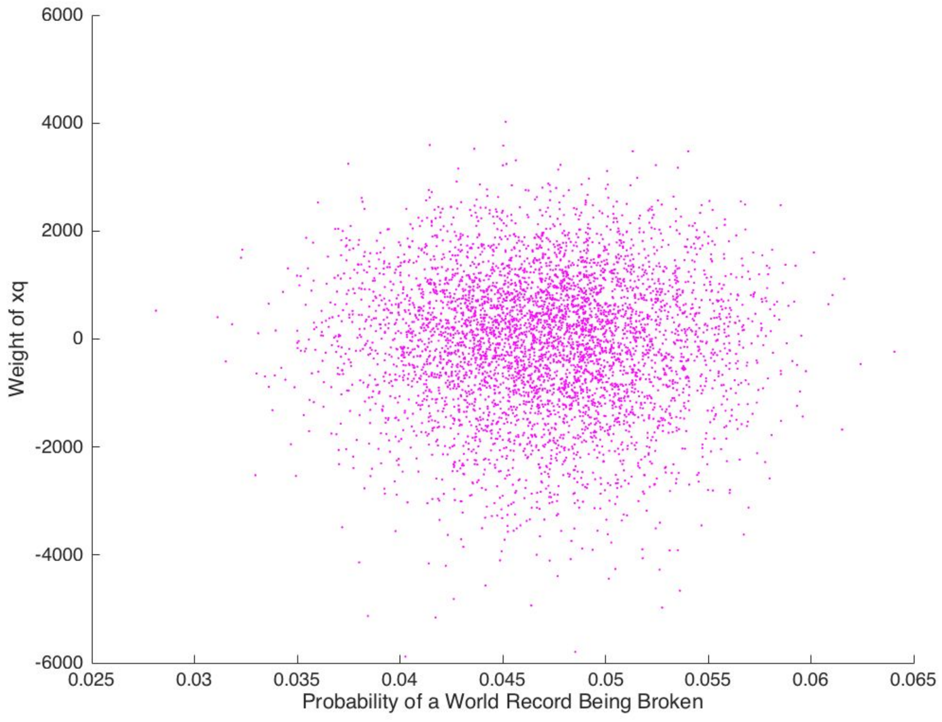
0.01
0.005
0
-0.005
-0.01
-0.015
Weight of xb
0.08
0.1
0.12
0.14
0.16
0.18
0.2
0.22
Insurance Operating Cost per Euro Earned as Income

r=.0155

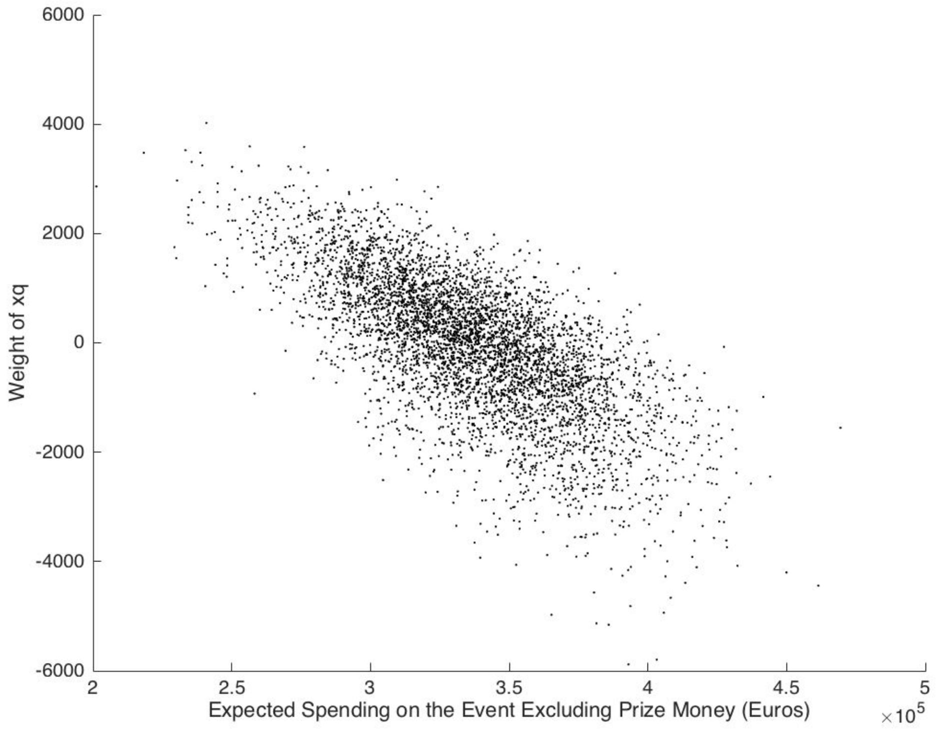


6000
4000
2000
0
-2000
-4000
-6000
Weight of xq
1.5
2
2.5
3
3.5
Prize Money (Euros)
×10^4

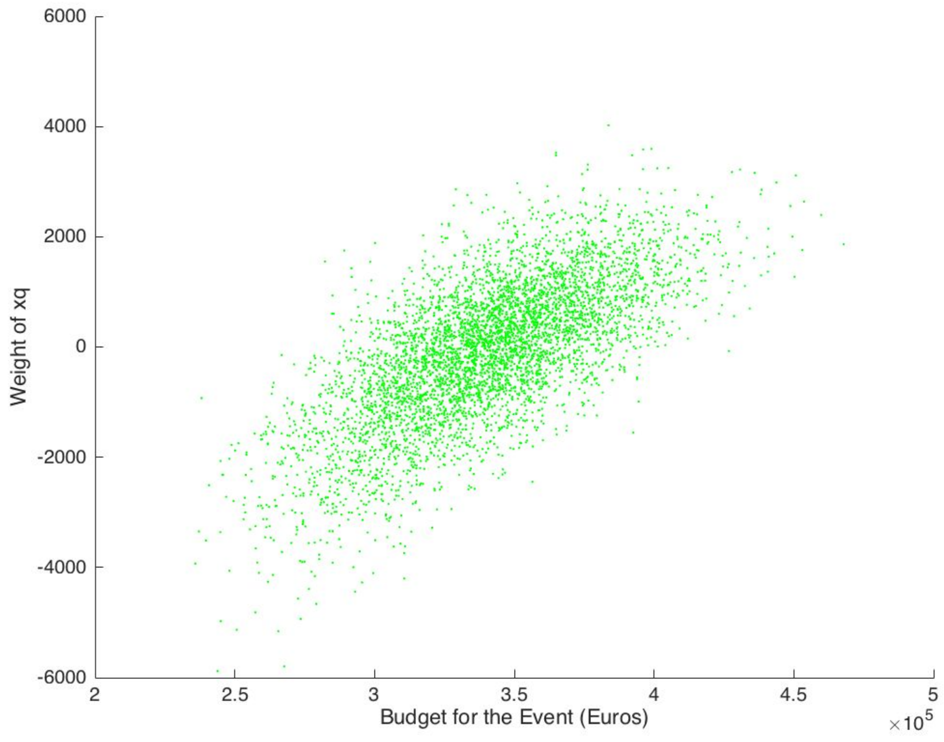
r=-.0087


Weight of xq
Probability of a World Record Being Broken

r=.0095

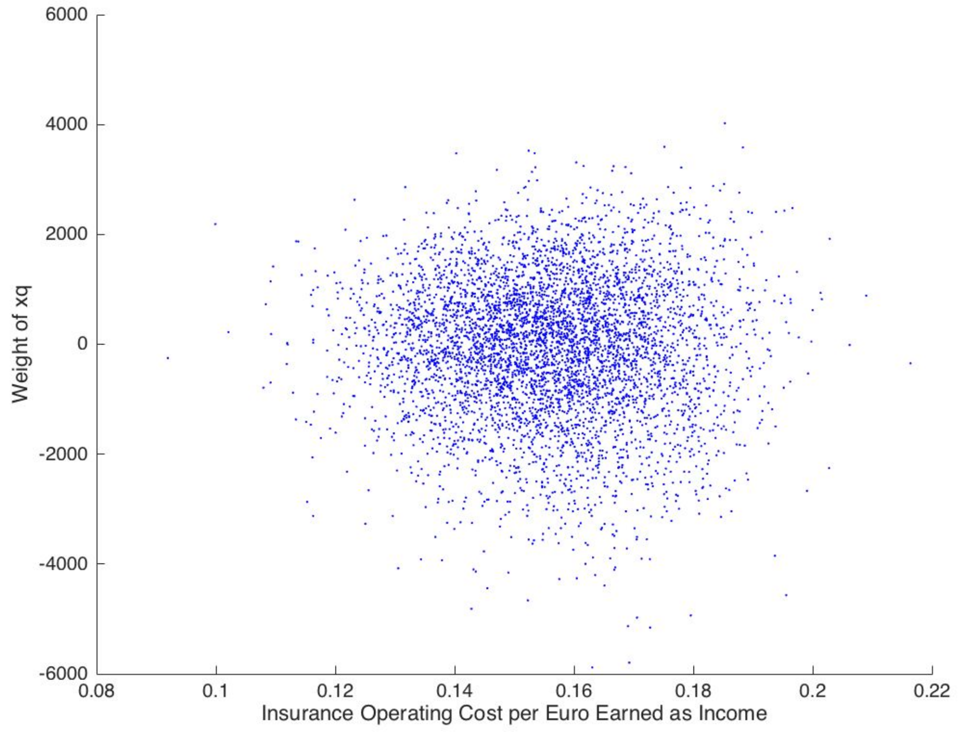

Weight of xq
Expected Spending on the Event Excluding Prize Money (Euros)
×10^5

r=-.6869



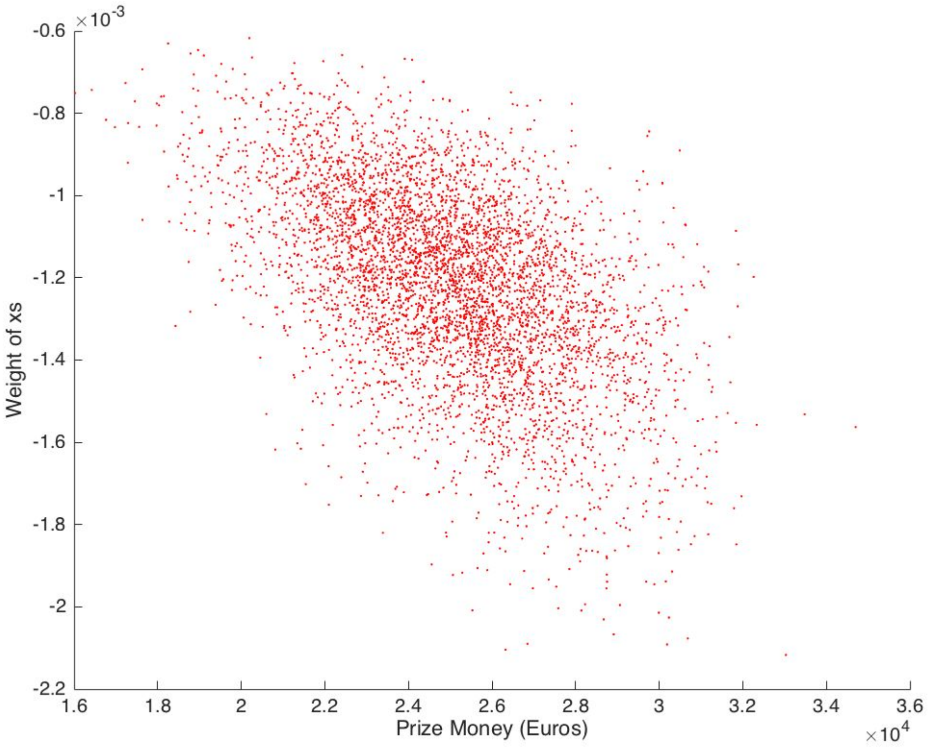
Weight of xq
6000
4000
2000
0
-2000
-4000
-6000
2
2.5
3
3.5
4
4.5
5
Budget for the Event (Euros)
$\times 10^5$

r=.7108



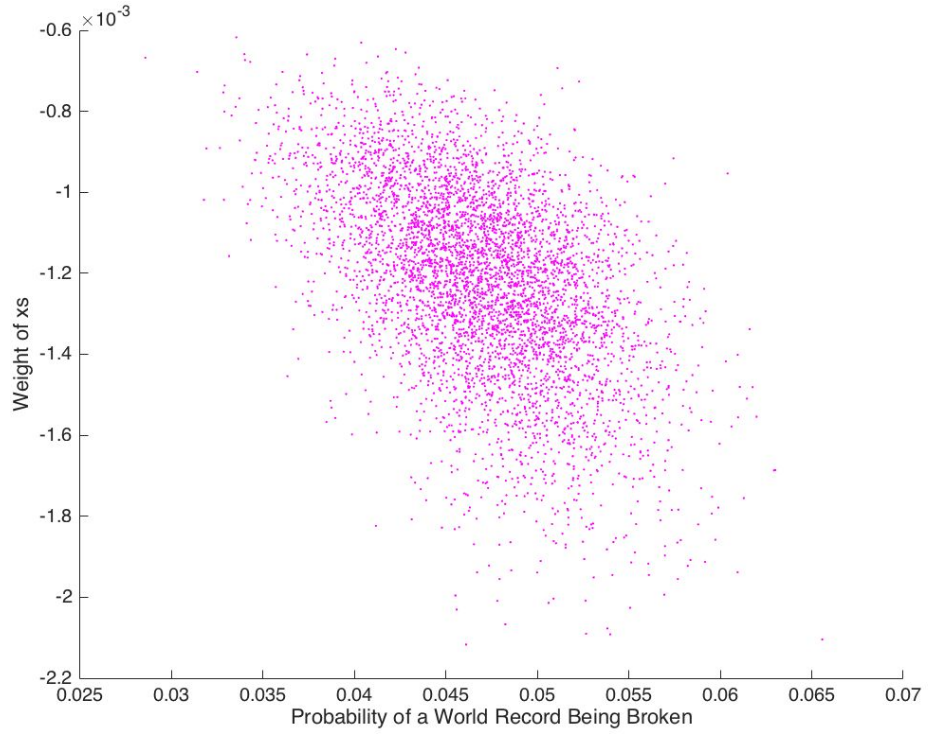
Weight of xq
6000
4000
2000
0
-2000
-4000
-6000
0.08
0.1
0.12
0.14
0.16
0.18
0.2
0.22
Insurance Operating Cost per Euro Earned as Income

r=-.0169



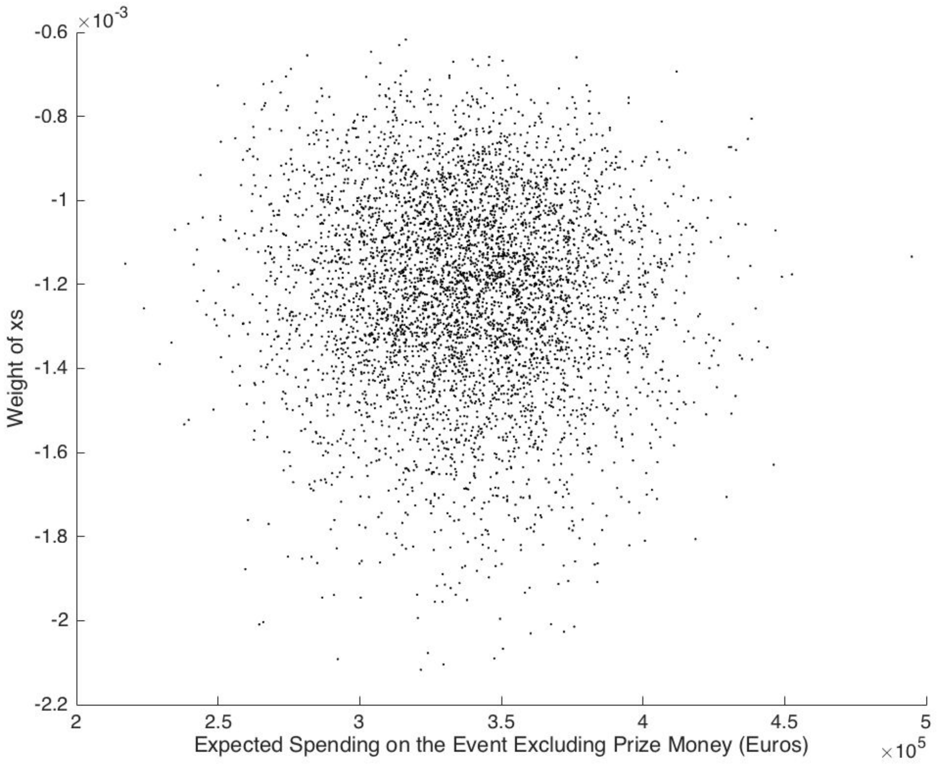
×10^-3
-0.6
-0.8
-1
-1.2
-1.4
-1.6
-1.8
-2
-2.2
Weight of xs
1.6
1.8
2
2.2
2.4
2.6
2.8
3
3.2
3.4
3.6
Prize Money (Euros)
×10^4

r=-.5189

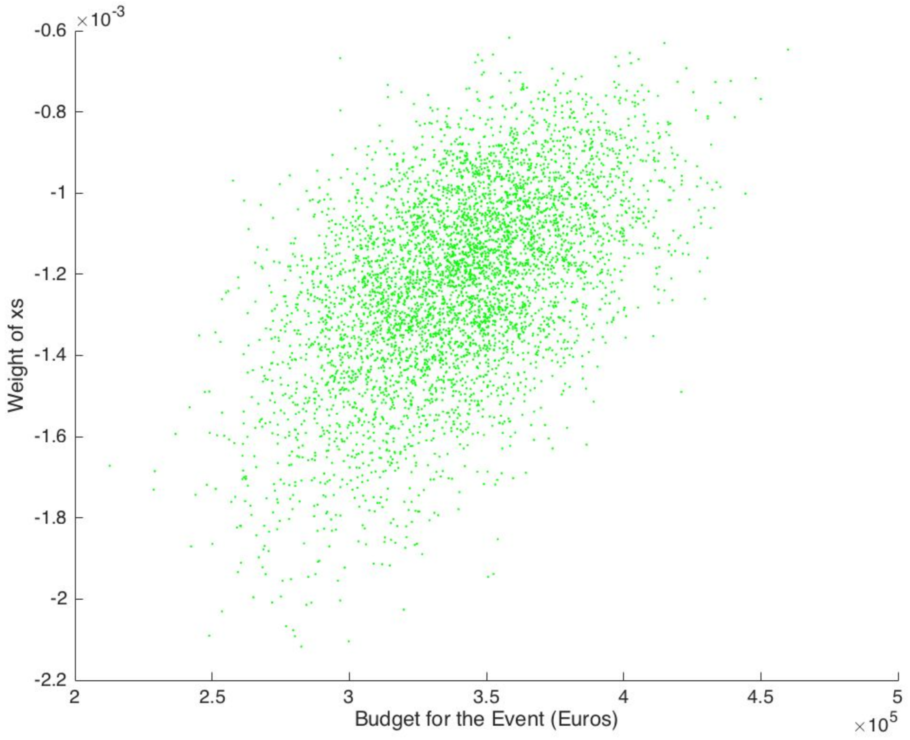


×10^-3
-0.6
-0.8
-1
-1.2
-1.4
-1.6
-1.8
-2
-2.2
Weight of xs
0.025
0.03
0.035
0.04
0.045
0.05
0.055
0.06
0.065
0.07
Probability of a World Record Being Broken

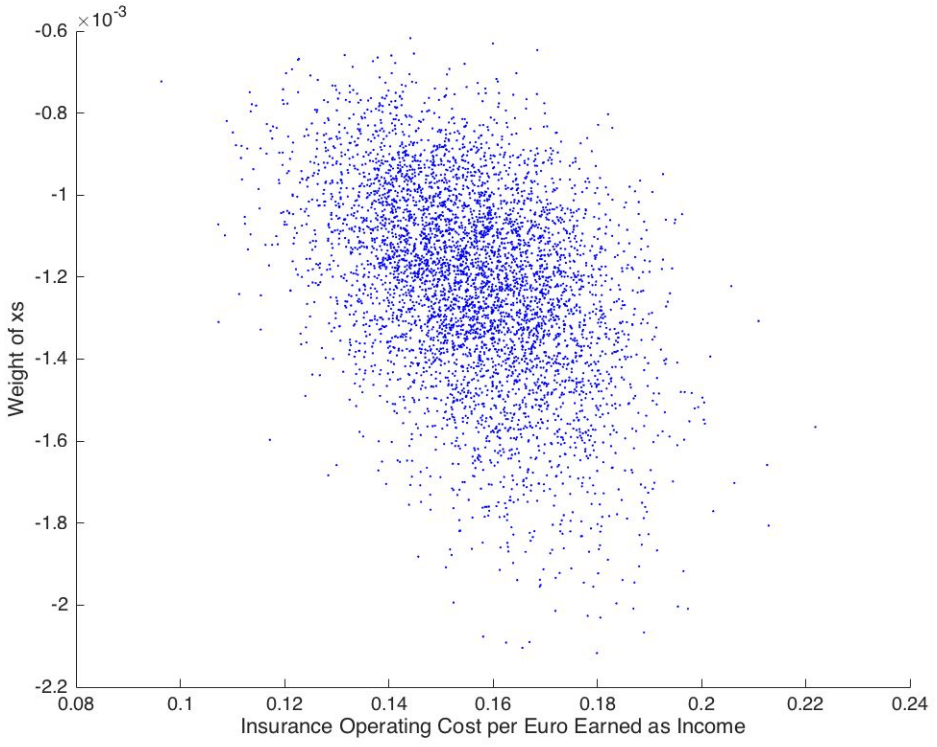
r=-.5207


×10^-3
-0.6
-0.8
-1
-1.2
-1.4
-1.6
-1.8
-2
-2.2
Weight of xs
2
2.5
3
3.5
4
4.5
5
×10^5
Expected Spending on the Event Excluding Prize Money (Euros)

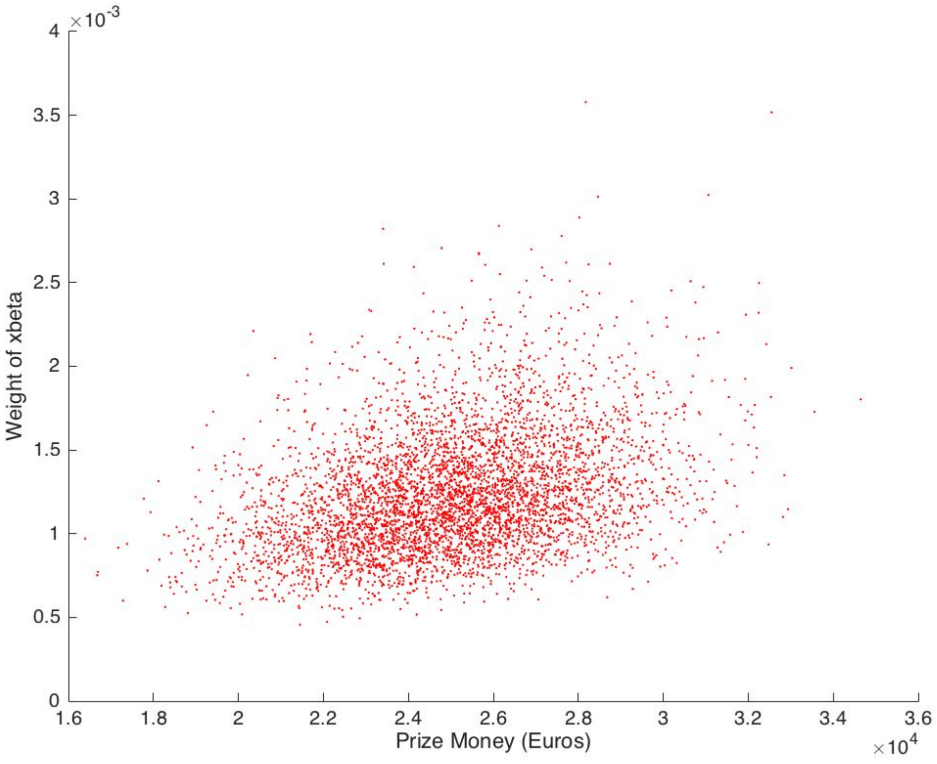
r=.0033


×10^-3
-0.6
-0.8
-1
-1.2
-1.4
-1.6
-1.8
-2
-2.2
Weight of xs
2
2.5
3
3.5
4
4.5
5
×10^5
Budget for the Event (Euros)

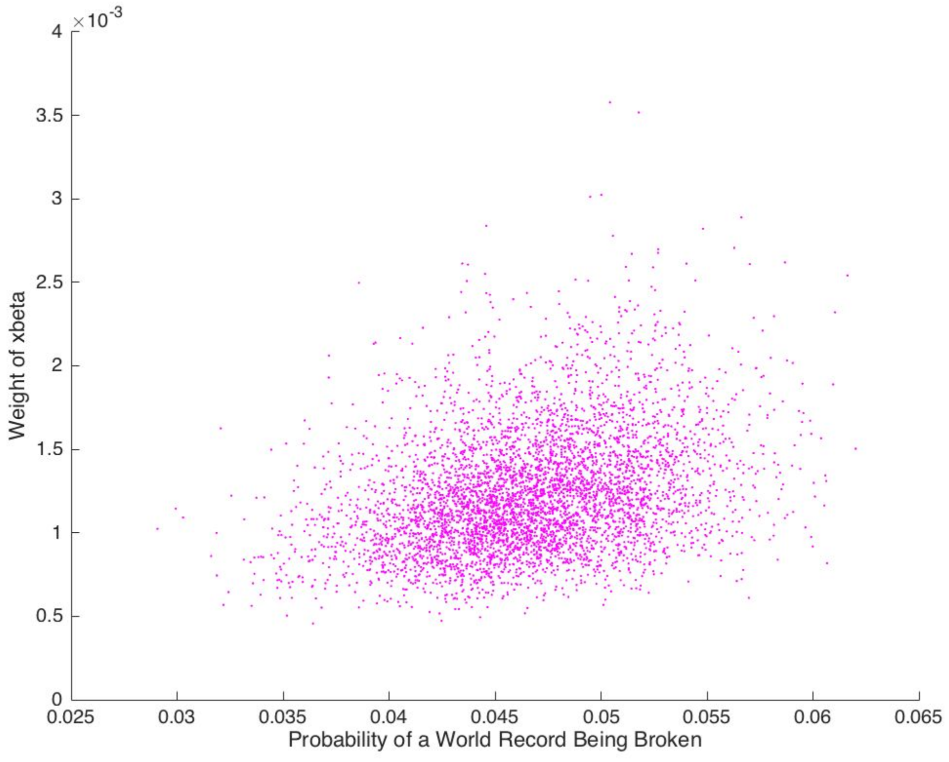
r=.5274


×10⁻³
-0.6
-0.8
-1
-1.2
-1.4
-1.6
-1.8
-2
-2.2
Weight of xs
0.08
0.1
0.12
0.14
0.16
0.18
0.2
0.22
0.24
Insurance Operating Cost per Euro Earned as Income

r=-.3851

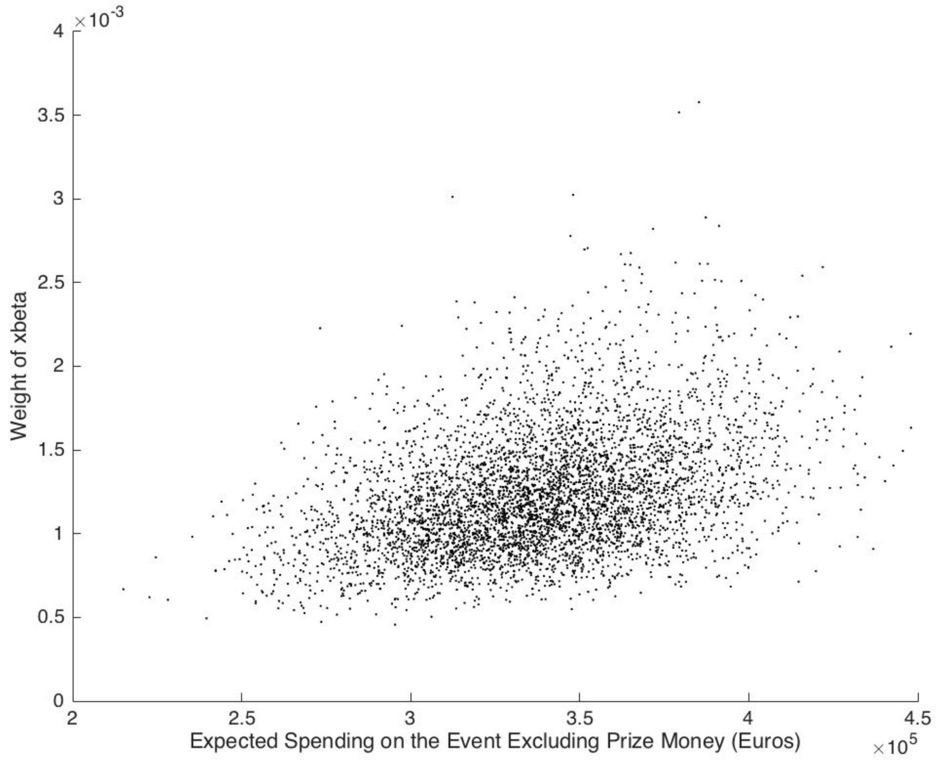

×10⁻³
4
3.5
3
2.5
2
1.5
1
0.5
0
Weight of xbeta
1.6
1.8
2
2.2
2.4
2.6
2.8
3
3.2
3.4
3.6
Prize Money (Euros)
×10⁴

r=.3391



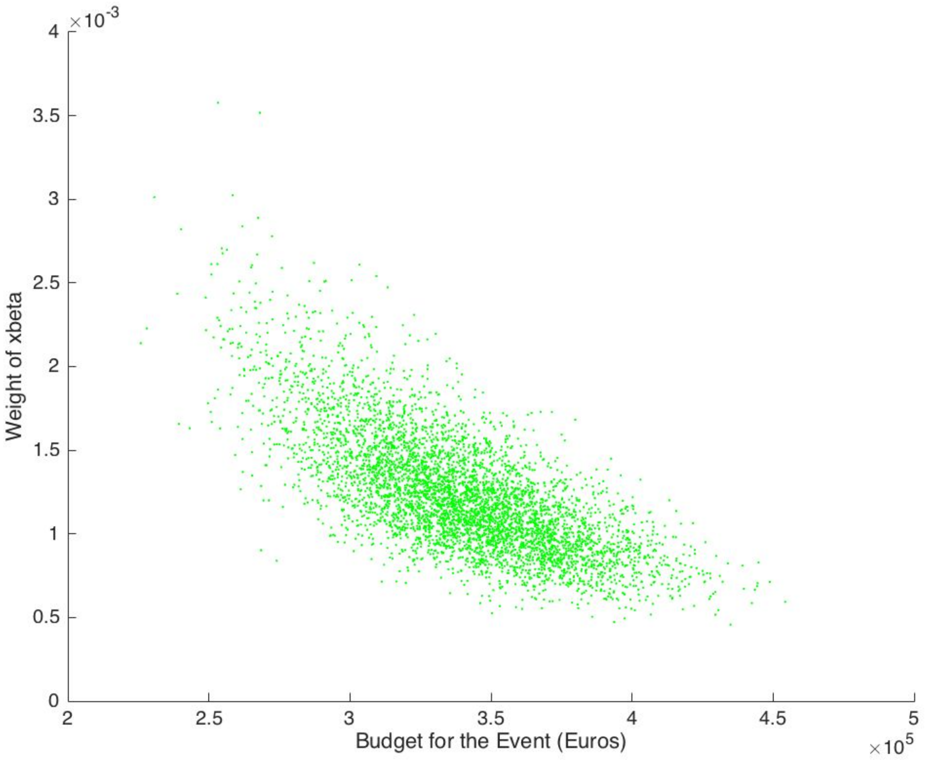
×10^-3
Weight of xbeta
Probability of a World Record Being Broken
0
0.5
1
1.5
2
2.5
3
3.5
4
0.025
0.03
0.035
0.04
0.045
0.05
0.055
0.06
0.065

r=.3353



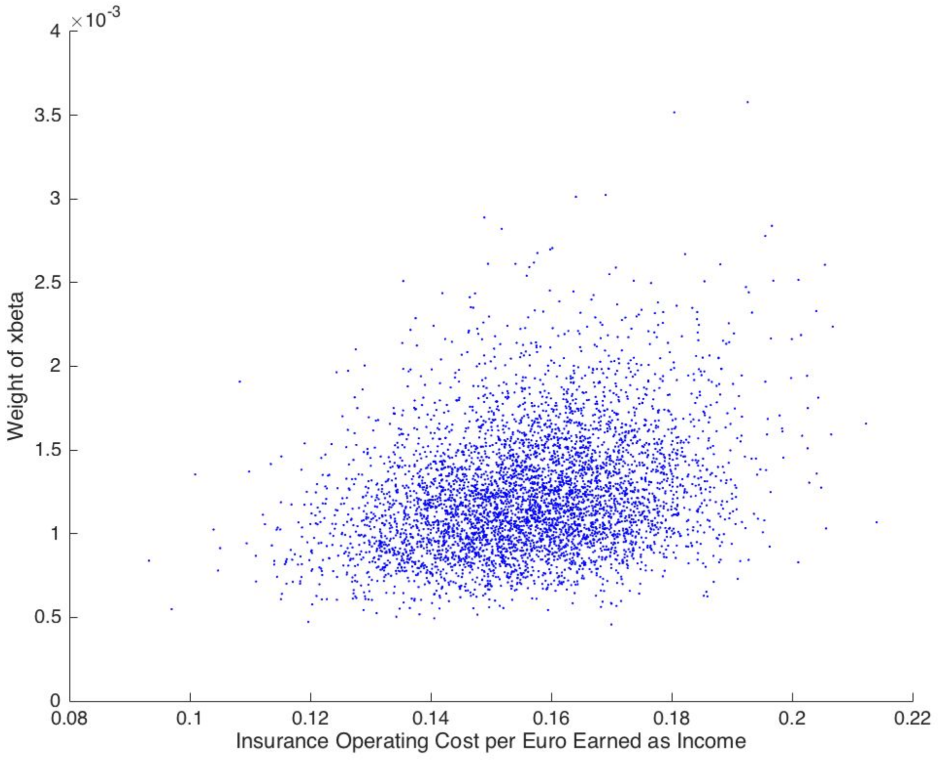
×10^-3
Weight of xbeta
Expected Spending on the Event Excluding Prize Money (Euros)
×10^5
0
0.5
1
1.5
2
2.5
3
3.5
4
2
2.5
3
3.5
4
4.5

r=.3441



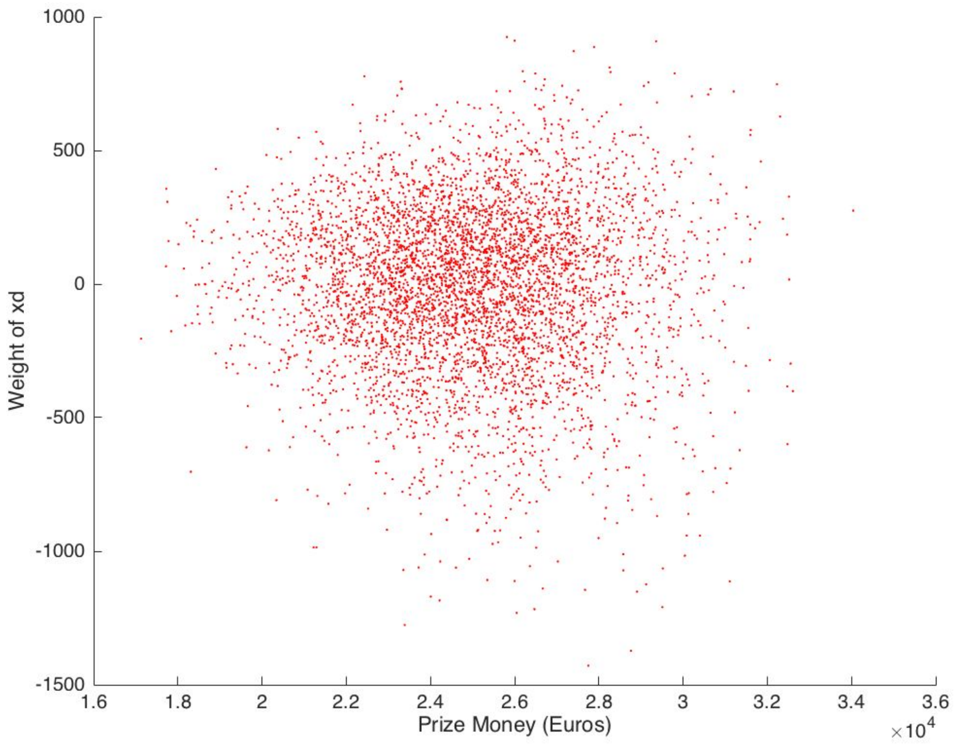
×10^-3
Weight of xbeta
Budget for the Event (Euros)
×10^5

r=-.7116

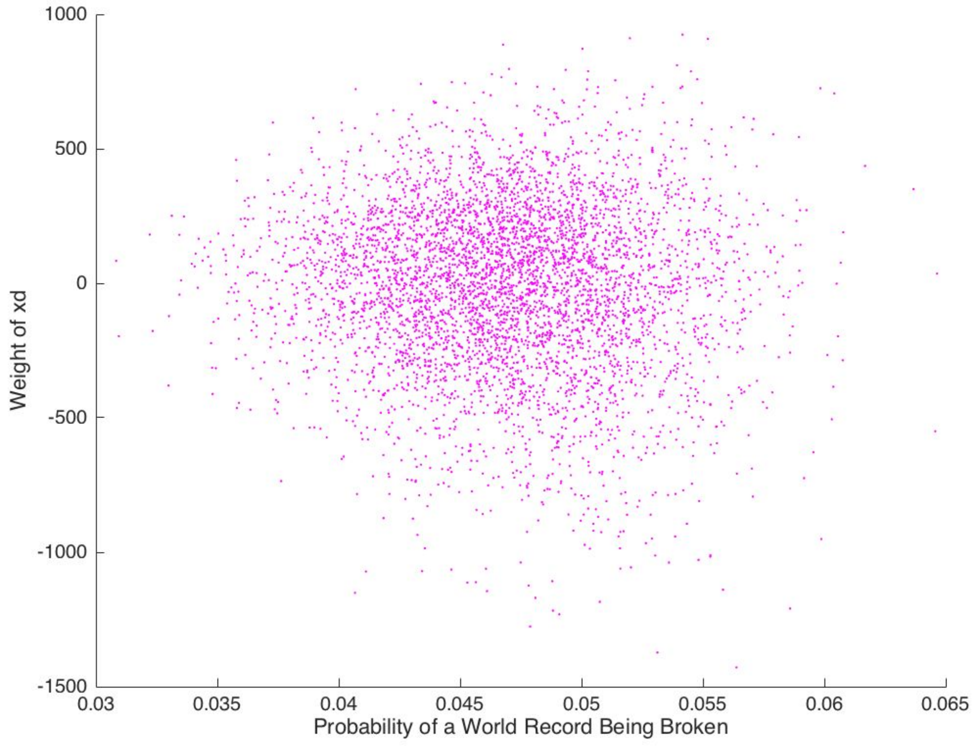


×10^-3
Weight of xbeta
Insurance Operating Cost per Euro Earned as Income

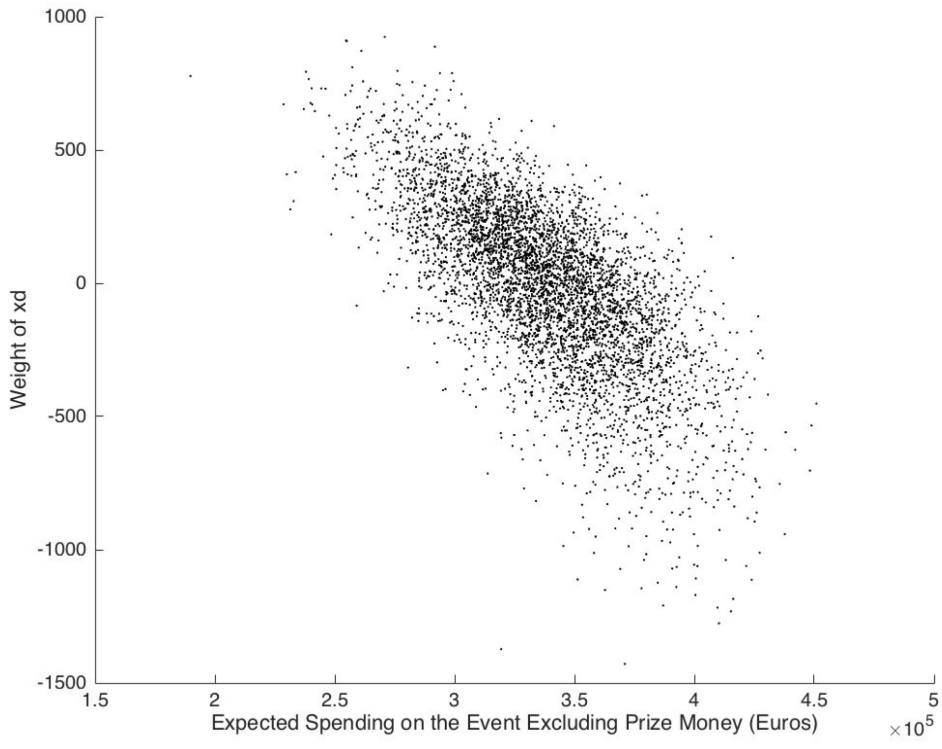
r=.2842


1000
500
0
-500
-1000
-1500
Weight of xd
1.6
1.8
2
2.2
2.4
2.6
2.8
3
3.2
3.4
3.6
Prize Money (Euros)
×10^4

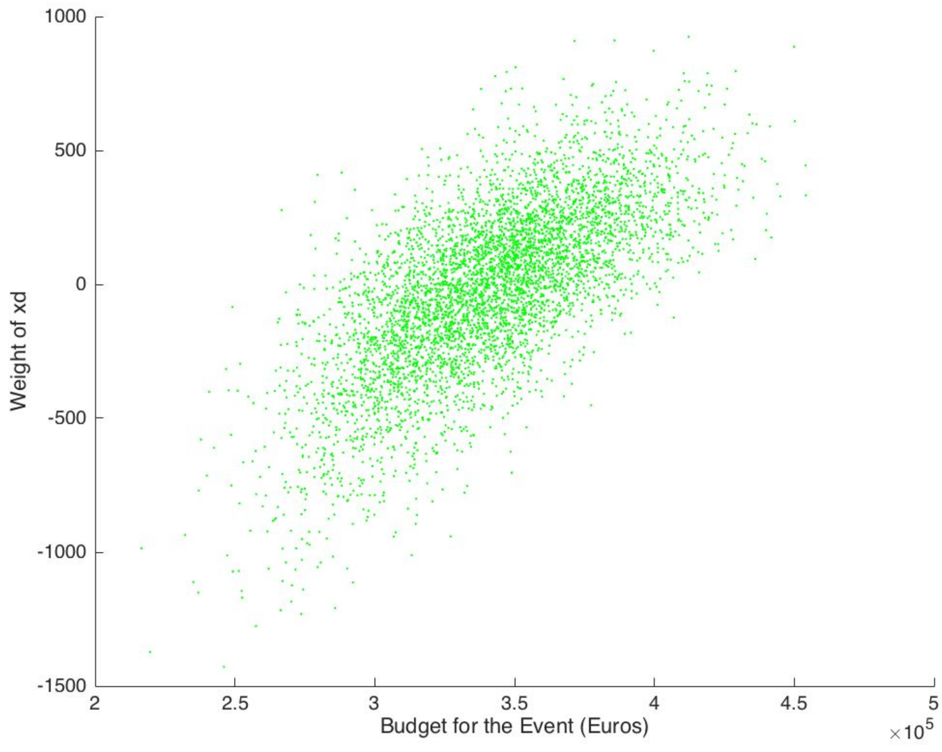
r=-.0119


1000
500
0
-500
-1000
-1500
Weight of xd
0.03
0.035
0.04
0.045
0.05
0.055
0.06
0.065
Probability of a World Record Being Broken

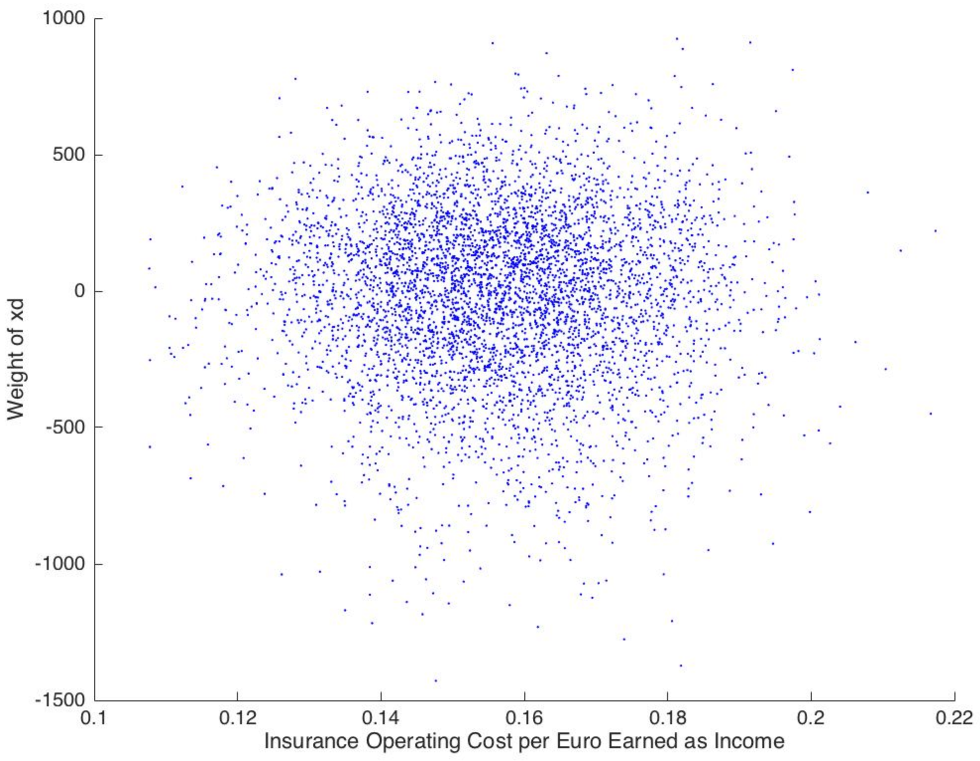
r=-.0314


1000
500
0
-500
-1000
-1500
Weight of xd
1.5
2
2.5
3
3.5
4
4.5
5
Expected Spending on the Event Excluding Prize Money (Euros)
×10^5

r=-.6945


1000
500
0
-500
-1000
-1500
Weight of xd
2
2.5
3
3.5
4
4.5
5
Budget for the Event (Euros)
×10^5

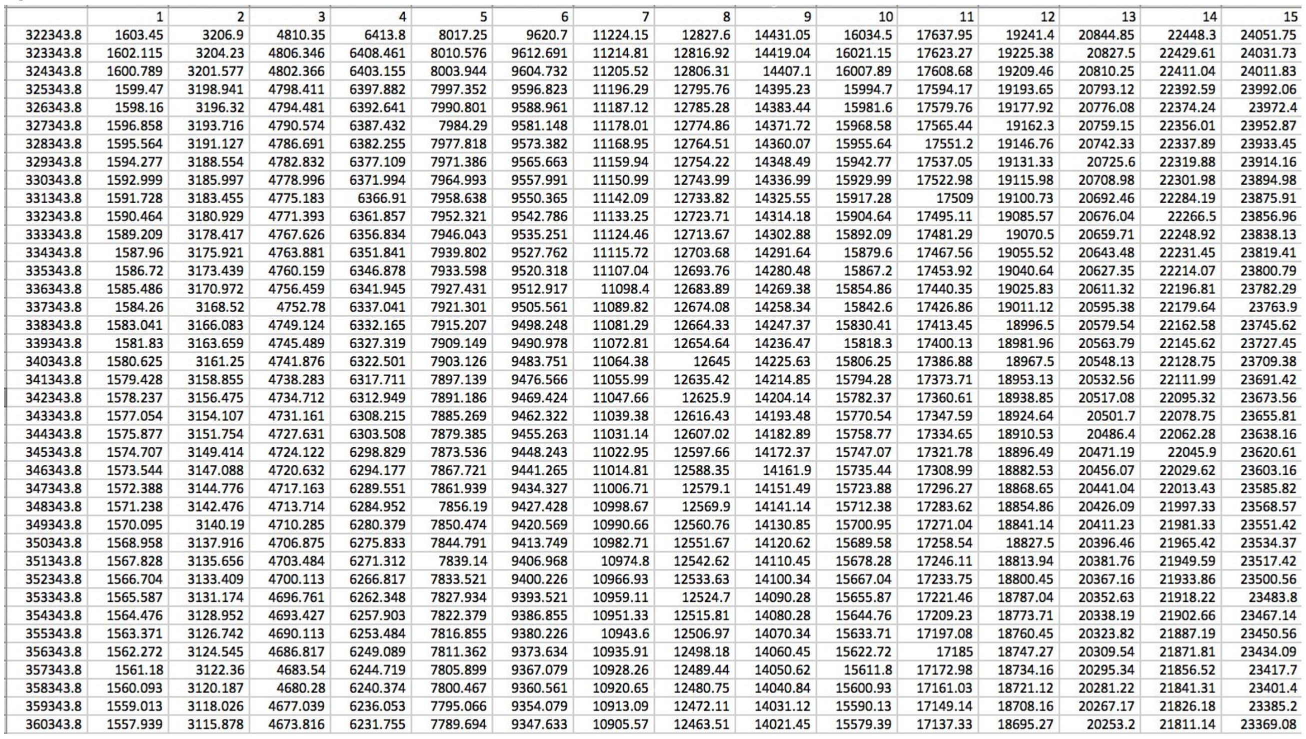
r=.6975



r=.0092

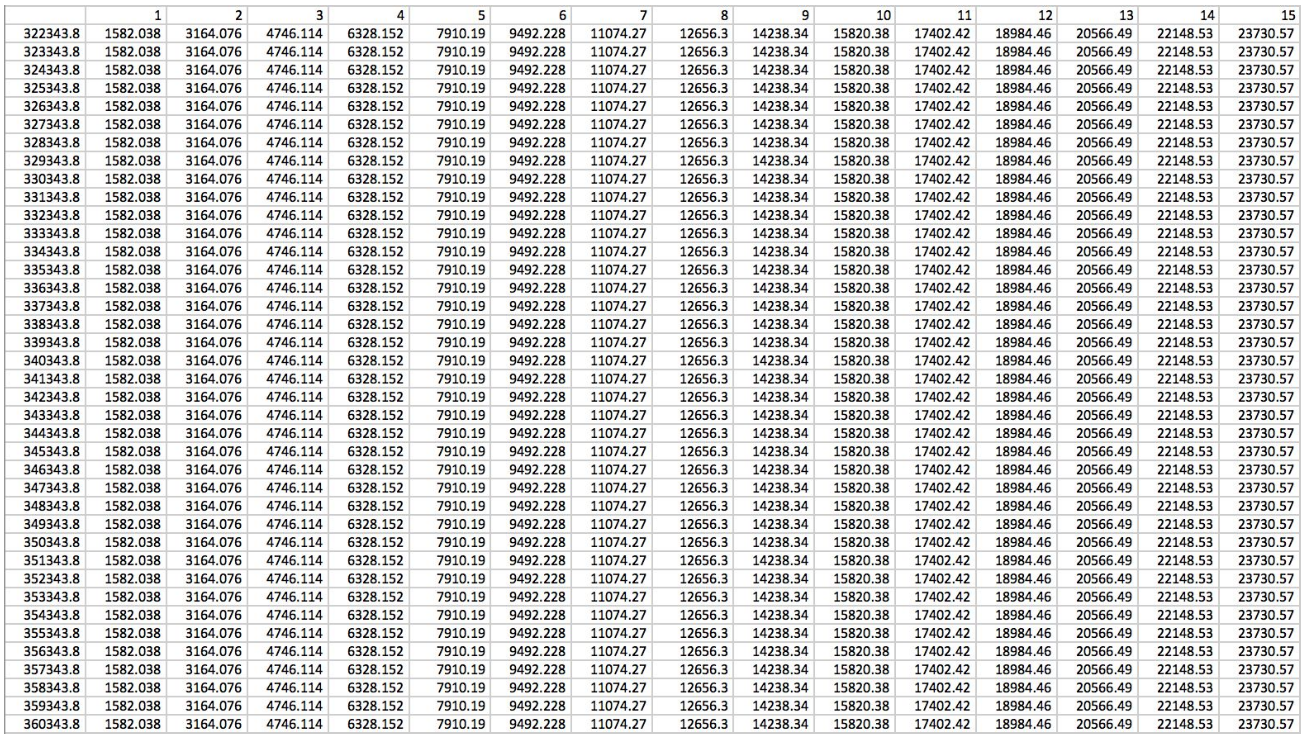
Tables

Question 3:



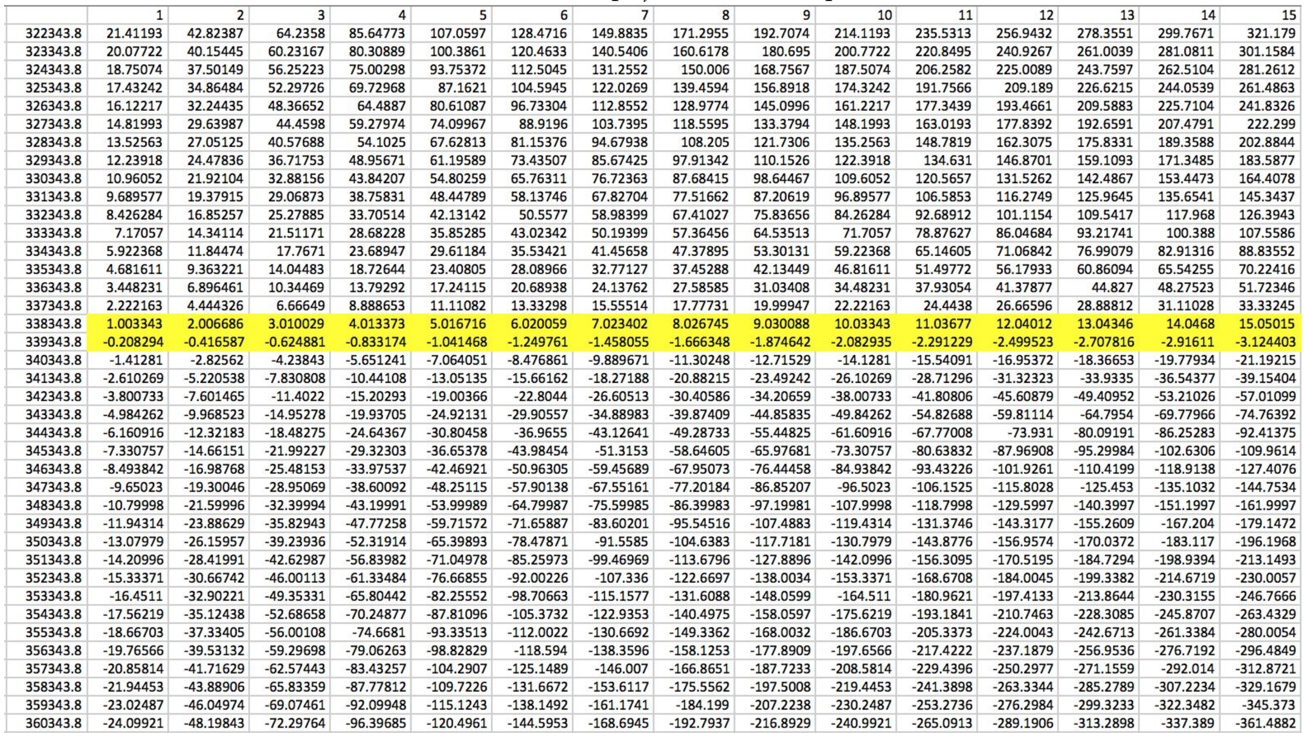
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
322343.8	1603.45	3206.9	4810.35	6413.8	8017.25	9620.7	11224.15	12827.6	14431.05	16034.5	17637.95	19241.4	20844.85	22448.3	24051.75
323343.8	1602.115	3204.23	4806.346	6408.461	8010.576	9612.691	11214.81	12816.92	14419.04	16021.15	17623.27	19225.38	20827.5	22429.61	24031.73
324343.8	1600.789	3201.577	4802.366	6403.155	8003.944	9604.732	11205.52	12806.31	14407.1	16007.89	17608.68	19209.46	20810.25	22411.04	24011.83
325343.8	1599.47	3198.941	4798.411	6397.882	7997.352	9596.823	11196.29	12795.76	14395.23	15994.7	17594.17	19193.65	20793.12	22392.59	23992.06
326343.8	1598.16	3196.32	4794.481	6392.641	7990.801	9588.961	11187.12	12785.28	14383.44	15981.6	17579.76	19177.92	20776.08	22374.24	23972.4
327343.8	1596.858	3193.716	4790.574	6387.432	7984.29	9581.148	11178.01	12774.86	14371.72	15968.58	17565.44	19162.3	20759.15	22356.01	23952.87
328343.8	1595.564	3191.127	4786.691	6382.255	7977.818	9573.382	11168.95	12764.51	14360.07	15955.64	17551.2	19146.76	20742.33	22337.89	23933.45
329343.8	1594.277	3188.554	4782.832	6377.109	7971.386	9565.663	11159.94	12754.22	14348.49	15942.77	17537.05	19131.33	20725.6	22319.88	23914.16
330343.8	1592.999	3185.997	4778.996	6371.994	7964.993	9557.991	11150.99	12743.99	14336.99	15929.99	17522.98	19115.98	20708.98	22301.98	23894.98
331343.8	1591.728	3183.455	4775.183	6366.91	7958.638	9550.365	11142.09	12733.82	14325.55	15917.28	17509	19100.73	20692.46	22284.19	23875.91
332343.8	1590.464	3180.929	4771.393	6361.857	7952.321	9542.786	11133.25	12723.71	14314.18	15904.64	17495.11	19085.57	20676.04	22266.5	23856.96
333343.8	1589.209	3178.417	4767.626	6356.834	7946.043	9535.251	11124.46	12713.67	14302.88	15892.09	17481.29	19070.5	20659.71	22248.92	23838.13
334343.8	1587.96	3175.921	4763.881	6351.841	7939.802	9527.762	11115.72	12703.68	14291.64	15879.6	17467.56	19055.52	20643.48	22231.45	23819.41
335343.8	1586.72	3173.439	4760.159	6346.878	7933.598	9520.318	11107.04	12693.76	14280.48	15867.2	17453.92	19040.64	20627.35	22214.07	23800.79
336343.8	1585.486	3170.972	4756.459	6341.945	7927.431	9512.917	11098.4	12683.89	14269.38	15854.86	17440.35	19025.83	20611.32	22196.81	23782.29
337343.8	1584.26	3168.52	4752.78	6337.041	7921.301	9505.561	11089.82	12674.08	14258.34	15842.6	17426.86	19011.12	20595.38	22179.64	23763.9
338343.8	1583.041	3166.083	4749.124	6332.165	7915.207	9498.248	11081.29	12664.33	14247.37	15830.41	17413.45	18996.5	20579.54	22162.58	23745.62
339343.8	1581.83	3163.659	4745.489	6327.319	7909.149	9490.978	11072.81	12654.64	14236.47	15818.3	17400.13	18981.96	20563.79	22145.62	23727.45
340343.8	1580.625	3161.25	4741.876	6322.501	7903.126	9483.751	11064.38	12645	14225.63	15806.25	17386.88	18967.5	20548.13	22128.75	23709.38
341343.8	1579.428	3158.855	4738.283	6317.711	7897.139	9476.566	11055.99	12635.42	14214.85	15794.28	17373.71	18953.13	20532.56	22111.99	23691.42
342343.8	1578.237	3156.475	4734.712	6312.949	7891.186	9469.424	11047.66	12625.9	14204.14	15782.37	17360.61	18938.85	20517.08	22095.32	23673.56
343343.8	1577.054	3154.107	4731.161	6308.215	7885.269	9462.322	11039.38	12616.43	14193.48	15770.54	17347.59	18924.64	20501.7	22078.75	23655.81
344343.8	1575.877	3151.754	4727.631	6303.508	7879.385	9455.263	11031.14	12607.02	14182.89	15758.77	17334.65	18910.53	20486.4	22062.28	23638.16
345343.8	1574.707	3149.414	4724.122	6298.829	7873.536	9448.243	11022.95	12597.66	14172.37	15747.07	17321.78	18896.49	20471.19	22045.9	23620.61
346343.8	1573.544	3147.088	4720.632	6294.177	7867.721	9441.265	11014.81	12588.35	14161.9	15735.44	17308.99	18882.53	20456.07	22029.62	23603.16
347343.8	1572.388	3144.776	4717.163	6289.551	7861.939	9434.327	11006.71	12579.1	14151.49	15723.88	17296.27	18868.65	20441.04	22013.43	23585.82
348343.8	1571.238	3142.476	4713.714	6284.952	7856.19	9427.428	10998.67	12569.9	14141.14	15712.38	17283.62	18854.86	20426.09	21997.33	23568.57
349343.8	1570.095	3140.19	4710.285	6280.379	7850.474	9420.569	10990.66	12560.76	14130.85	15700.95	17271.04	18841.14	20411.23	21981.33	23551.42
350343.8	1568.958	3137.916	4706.875	6275.833	7844.791	9413.749	10982.71	12551.67	14120.62	15689.58	17258.54	18827.5	20396.46	21965.42	23534.37
351343.8	1567.828	3135.656	4703.484	6271.312	7839.14	9406.968	10974.8	12542.62	14110.45	15678.28	17246.11	18813.94	20381.76	21949.59	23517.42
352343.8	1566.704	3133.409	4700.113	6266.817	7833.521	9400.226	10966.93	12533.63	14100.34	15667.04	17233.75	18800.45	20367.16	21933.86	23500.56
353343.8	1565.587	3131.174	4696.761	6262.348	7827.934	9393.521	10959.11	12524.7	14090.28	15655.87	17221.46	18787.04	20352.63	21918.22	23483.8
354343.8	1564.476	3128.952	4693.427	6257.903	7822.379	9386.855	10951.33	12515.81	14080.28	15644.76	17209.23	18773.71	20338.19	21902.66	23467.14
355343.8	1563.371	3126.742	4690.113	6253.484	7816.855	9380.226	10943.6	12506.97	14070.34	15633.71	17197.08	18760.45	20323.82	21887.19	23450.56
356343.8	1562.272	3124.545	4686.817	6249.089	7811.362	9373.634	10935.91	12498.18	14060.45	15622.72	17185	18747.27	20309.54	21871.81	23434.09
357343.8	1561.18	3122.36	4683.54	6244.719	7805.899	9367.079	10928.26	12489.44	14050.62	15611.8	17172.98	18734.16	20295.34	21856.52	23417.7
358343.8	1560.093	3120.187	4680.28	6240.374	7800.467	9360.561	10920.65	12480.75	14040.84	15600.93	17161.03	18721.12	20281.22	21841.31	23401.4
359343.8	1559.013	3118.026	4677.039	6236.053	7795.066	9354.079	10913.09	12472.11	14031.12	15590.13	17149.14	18708.16	20267.17	21826.18	23385.2
360343.8	1557.939	3115.878	4673.816	6231.755	7789.694	9347.633	10905.57	12463.51	14021.45	15579.39	17137.33	18695.27	20253.2	21811.14	23369.08

Table of Zevenheuvelenloop r_n values with respect to t and β



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
322343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
323343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
324343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
325343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
326343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
327343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
328343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
329343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
330343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
331343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
332343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
333343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
334343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
335343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
336343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
337343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
338343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
339343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
340343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
341343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
342343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
343343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
344343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
345343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
346343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
347343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
348343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
349343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
350343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
351343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
352343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
353343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
354343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
355343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
356343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
357343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
358343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
359343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57
360343.8	1582.038	3164.076	4746.114	6328.152	7910.19	9492.228	11074.27	12656.3	14238.34	15820.38	17402.42	18984.46	20566.49	22148.53	23730.57

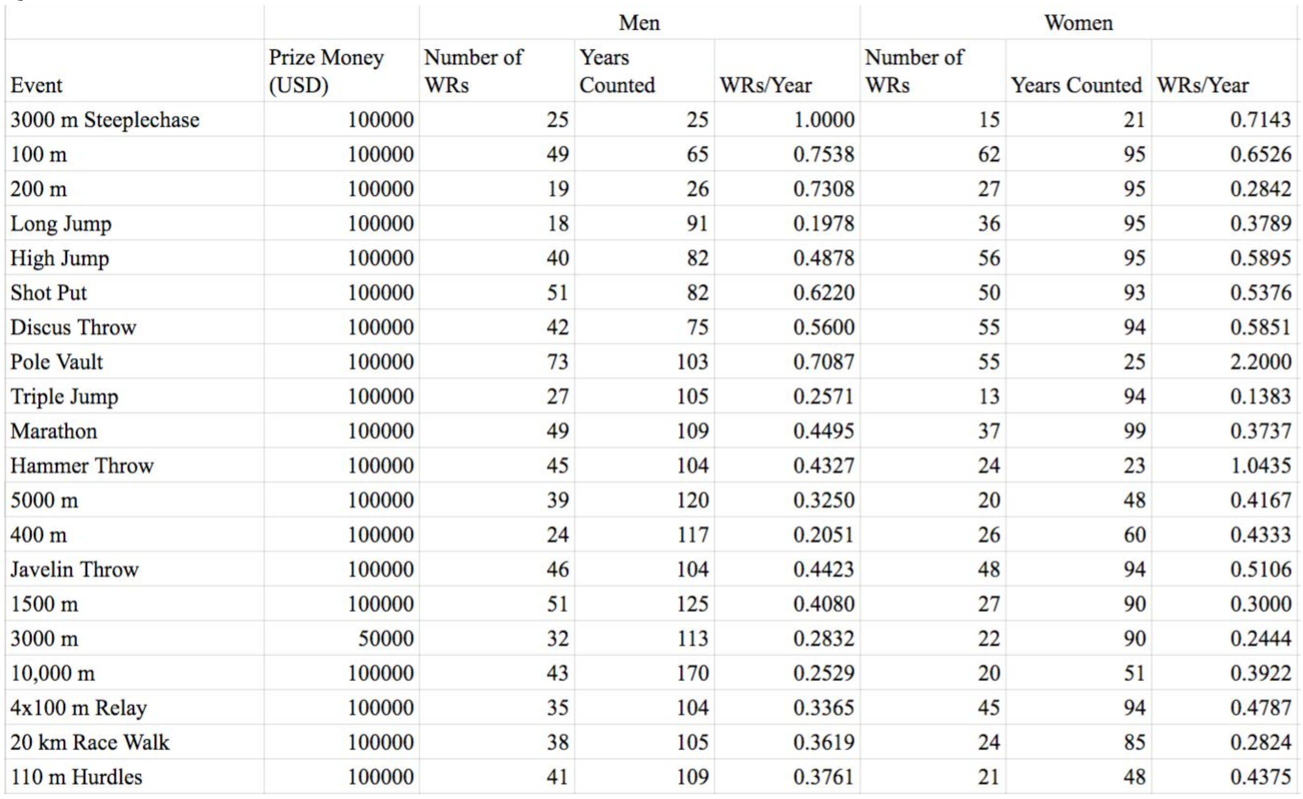
Table of Zevenheuvelenloop r_i values with respect to t and β



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
322343.8	21.41193	42.82387	64.2358	85.64773	107.0597	128.4716	149.8835	171.2955	192.7074	214.1193	235.5313	256.9432	278.3551	299.7671	321.179
323343.8	20.07722	40.15445	60.23167	80.30889	100.3861	120.4633	140.5406	160.6178	180.695	200.7722	220.8495	240.9267	261.0039	281.0811	301.1584
324343.8	18.75074	37.50149	56.25223	75.00298	93.75372	112.5045	131.2552	150.006	168.7567	187.5074	206.2582	225.0089	243.7597	262.5104	281.2612
325343.8	17.43242	34.86484	52.29726	69.72968	87.1621	104.5945	122.0269	139.4594	156.8918	174.3242	191.7566	209.189	226.6215	244.0539	261.4863
326343.8	16.12217	32.24435	48.36652	64.4887	80.61087	96.73304	112.8552	128.9774	145.0996	161.2217	177.3439	193.4661	209.5883	225.7104	241.8326
327343.8	14.81993	29.63987	44.4598	59.27974	74.09967	88.9196	103.7395	118.5595	133.3794	148.1993	163.0193	177.8392	192.6591	207.4791	222.299
328343.8	13.52563	27.05125	40.57688	54.1025	67.62813	81.15376	94.67938	108.205	121.7306	135.2563	148.7819	162.3075	175.8331	189.3588	202.8844
329343.8	12.23918	24.47836	36.71753	48.95671	61.19589	73.43507	85.67425	97.91342	110.1526	122.3918	134.631	146.8701	159.1093	171.3485	183.5877
330343.8	10.96052	21.92104	32.88156	43.84207	54.80259	65.76311	76.72363	87.68415	98.64467	109.6052	120.5657	131.5262	142.4867	153.4473	164.4078
331343.8	9.689577	19.37915	29.06873	38.75831	48.44789	58.13746	67.82704	77.51662	87.20619	96.89577	106.5853	116.2749	125.9645	135.6541	145.3437
332343.8	8.426284	16.85257	25.27885	33.70514	42.13142	50.5577	58.98399	67.41027	75.83656	84.26284	92.68912	101.1154	109.5417	117.968	126.3943
333343.8	7.17057	14.34114	21.51171	28.68228	35.85285	43.02342	50.19399	57.36456	64.53513	71.7057	78.87627	86.04684	93.21741	100.388	107.5586
334343.8	5.922368	11.84474	17.7671	23.68947	29.61184	35.53421	41.45658	47.37895	53.30131	59.22368	65.14605	71.06842	76.99079	82.91316	88.83552
335343.8	4.681611	9.363221	14.04483	18.72644	23.40805	28.08966	32.77127	37.45288	42.13449	46.81611	51.49772	56.17933	60.86094	65.54255	70.22416
336343.8	3.448231	6.896461	10.34469	13.79292	17.24115	20.68938	24.13762	27.58585	31.03408	34.48231	37.93054	41.37877	44.827	48.27523	51.72346
337343.8	2.222163	4.444326	6.66649	8.888653	11.11082	13.33298	15.55514	17.77731	19.99947	22.22163	24.4438	26.66596	28.88812	31.11028	33.33245
338343.8	1.003343	2.006686	3.010029	4.013373	5.016716	6.020059	7.023402	8.026745	9.030088	10.03343	11.03677	12.04012	13.04346	14.0468	15.05015
339343.8	-0.208294	-0.416587	-0.624881	-0.833174	-1.041468	-1.249761	-1.458055	-1.666348	-1.874642	-2.082935	-2.291229	-2.499523	-2.707816	-2.91611	-3.124403
340343.8	-1.41281	-2.82562	-4.23843	-5.651241	-7.064051	-8.476861	-9.889671	-11.30248	-12.71529	-14.1281	-15.54091	-16.95372	-18.36653	-19.77934	-21.19215
341343.8	-2.610269	-5.220538	-7.830808	-10.44108	-13.05135	-15.66162	-18.27188	-20.88215	-23.49242	-26.10269	-28.71296	-31.32323	-33.9335	-36.54377	-39.15404
342343.8	-3.800733	-7.601465	-11.4022	-15.20293	-19.00366	-22.8044	-26.60513	-30.40586	-34.20659	-38.00733	-41.80806	-45.60879	-49.40952	-53.21026	-57.01099
343343.8	-4.984262	-9.968523	-14.95278	-19.93705	-24.92131	-29.90557	-34.88983	-39.87409	-44.85835	-49.84262	-54.82688	-59.81114	-64.7954	-69.77966	-74.76392
344343.8	-6.160916	-12.32183	-18.48275	-24.64367	-30.80458	-36.9655	-43.12641	-49.28733	-55.44825	-61.60916	-67.77008	-73.931	-80.09191	-86.25283	-92.41375
345343.8	-7.330757	-14.66151	-21.99227	-29.32303	-36.65378	-43.98454	-51.3153	-58.64605	-65.97681	-73.30757	-80.63832	-87.96908	-95.29984	-102.6306	-109.9614
346343.8	-8.493842	-16.98768	-25.48153	-33.97537	-42.46921	-50.96305	-59.45689	-67.95073	-76.44458	-84.93842	-93.43226	-101.9261	-110.4199	-118.9138	-127.4076
347343.8	-9.65023	-19.30046	-28.95069	-38.60092	-48.25115	-57.90138	-67.55161	-77.20184	-86.85207	-96.5023	-106.1525	-115.8028	-125.453	-135.1032	-144.7534
348343.8	-10.79998	-21.59996	-32.39994	-43.19991	-53.99989	-64.79987	-75.59985	-86.39983	-97.19981	-107.9998	-118.7998	-129.5997	-140.3997	-151.1997	-161.9997
349343.8	-11.94314	-23.88629	-35.82943	-47.77258	-59.71572	-71.65887	-83.60201	-95.54516	-107.4883	-119.4314	-131.3746	-143.3177	-155.2609	-167.204	-179.1472
350343.8	-13.07979	-26.15957	-39.23936	-52.31914	-65.39893	-78.47871	-91.5585	-104.6383	-117.7181	-130.7979	-143.8776	-156.9574	-170.0372	-183.117	-196.1968
351343.8	-14.20996	-28.41991	-42.62987	-56.83982	-71.04978	-85.25973	-99.46969	-113.6796	-127.8896	-142.0996	-156.3095	-170.5195	-184.7294	-198.9394	-213.1493
352343.8	-15.33371	-30.66742	-46.00113	-61.33484	-76.66855	-92.00226	-107.336	-122.6697	-138.0034	-153.3371	-168.6708	-184.0045	-199.3382	-214.6719	-230.0057
353343.8	-16.4511	-32.90221	-49.35331	-65.80442	-82.25552	-98.70663	-115.1577	-131.6088	-148.0599	-164.511	-180.9621	-197.4133	-213.8644	-230.3155	-246.7666
354343.8	-17.56219	-35.12438	-52.68658	-70.24877	-87.81096	-105.3732	-122.9353	-140.4975	-158.0597	-175.6219	-193.1841	-210.7463	-228.3085	-245.8707	-263.4329
355343.8	-18.66703	-37.33405	-56.00108	-74.6681	-93.33513	-112.0022	-130.6692	-149.3362	-168.0032	-186.6703	-205.3373	-224.0043	-242.6713	-261.3384	-280.0054
356343.8	-19.76566	-39.53132	-59.29698	-79.06263	-98.82829	-118.594	-138.3596	-158.1253	-177.8909	-197.6566	-217.4222	-237.1879	-256.9536	-276.7192	-296.4849
357343.8	-20.85814	-41.71629	-62.57443	-83.43257	-104.2907	-125.1489	-146.007	-166.8651	-187.7233	-208.5814	-229.4396	-250.2977	-271.1559	-292.014	-312.8721
358343.8	-21.94453	-43.88906	-65.83359	-87.77812	-109.7226	-131.6672	-153.6117	-175.5562	-197.5008	-219.4453	-241.3898	-263.3344	-285.2789	-307.2234	-329.1679
359343.8	-23.02487	-46.04974	-69.07461	-92.09948	-115.1243	-138.1492	-161.1741	-184.199	-207.2238	-230.2487	-253.2736	-276.2984	-299.3233	-322.3482	-345.373
360343.8	-24.09921	-48.19843	-72.29764	-96.39685	-120.4961	-144.5953	-168.6945	-192.7937	-216.8929	-240.9921	-265.0913	-289.1906	-313.2898	-337.389	-361.4882

Table of Zevenheuvelenloop r_n - r_i values with respect to t and β, with the transition from positive to negative financial exposure difference highlighted

Question 4:



		Men			Women		
Event	Prize Money (USD)	Number of WRs	Years Counted	WRs/Year	Number of WRs	Years Counted	WRs/Year
3000 m Steeplechase	100000	25	25	1.0000	15	21	0.7143
100 m	100000	49	65	0.7538	62	95	0.6526
200 m	100000	19	26	0.7308	27	95	0.2842
Long Jump	100000	18	91	0.1978	36	95	0.3789
High Jump	100000	40	82	0.4878	56	95	0.5895
Shot Put	100000	51	82	0.6220	50	93	0.5376
Discus Throw	100000	42	75	0.5600	55	94	0.5851
Pole Vault	100000	73	103	0.7087	55	25	2.2000
Triple Jump	100000	27	105	0.2571	13	94	0.1383
Marathon	100000	49	109	0.4495	37	99	0.3737
Hammer Throw	100000	45	104	0.4327	24	23	1.0435
5000 m	100000	39	120	0.3250	20	48	0.4167
400 m	100000	24	117	0.2051	26	60	0.4333
Javelin Throw	100000	46	104	0.4423	48	94	0.5106
1500 m	100000	51	125	0.4080	27	90	0.3000
3000 m	50000	32	113	0.2832	22	90	0.2444
10,000 m	100000	43	170	0.2529	20	51	0.3922
4x100 m Relay	100000	35	104	0.3365	45	94	0.4787
20 km Race Walk	100000	38	105	0.3619	24	85	0.2824
110 m Hurdles	100000	41	109	0.3761	21	48	0.4375

Table of 20 track-and-field events with prize money and number of world records per year for men and women

MATLAB Programs

Question 3:
MATLAB Program 1: Simulation of the Zevenheuvelenloop

```
clear all;

phat=3/64;

NUMBER_OF_TRIALS=10000;
NUMBER_OF_YEARS=1000;
COST_PER_YEAR=0;
INSURANCE_PROFIT_MARGIN=1.1;
PRIZE_MONEY=20000;
DEDUCTIBLE=.2;
INSURANCE_FLAT_RATE=INSURANCE_PROFIT_MARGIN*phat*PRIZE_MONEY;

%% creates distribution using deductible scaled to p-hat
for j=1:NUMBER_OF_TRIALS
    for i=1:NUMBER_OF_YEARS

COST_PER_YEAR=COST_PER_YEAR+INSURANCE_PROFIT_MARGIN*(1-DEDUCTIBLE)*PRIZE_MONEY*phat;
        if rand<phat
            COST_PER_YEAR=COST_PER_YEAR+(DEDUCTIBLE)*PRIZE_MONEY;
        end
        COSTS_SCALED_SAMPLE(i)=COST_PER_YEAR;
        COST_PER_YEAR=0;
    end
    COSTS_SCALED(j)=sum(COSTS_SCALED_SAMPLE);
end

CI_SCALED=[prctile(COSTS_SCALED,5), prctile(COSTS_SCALED,95)]


%% insurance with no deductible
for j=1:NUMBER_OF_TRIALS
    for i=1:NUMBER_OF_YEARS
        COST_PER_YEAR=COST_PER_YEAR+INSURANCE_FLAT_RATE;
        COSTS_FLAT_SAMPLE(i)=COST_PER_YEAR;
        COST_PER_YEAR=0;
    end
    COSTS_FLAT(j)=sum(COSTS_FLAT_SAMPLE);
end

CI_FLAT=[prctile(COSTS_FLAT,5), prctile(COSTS_FLAT,95)]



%% creates distribution with no insurance at all
for j=1:NUMBER_OF_TRIALS
    for i=1:NUMBER_OF_YEARS
        if rand<phat
            COST_PER_YEAR=COST_PER_YEAR+PRIZE_MONEY;
        end
        COSTS_UNINSURED_SAMPLE(i)=COST_PER_YEAR;
        COST_PER_YEAR=0;
    end
    COSTS_UNINSURED(j)=sum(COSTS_UNINSURED_SAMPLE);
end

CI_UNINSURED=[prctile(COSTS_UNINSURED,5), prctile(COSTS_UNINSURED,95)]

% [counts1, binCenters1] = hist(COSTS_UNINSURED);
% [counts2, binCenters2] = hist(COSTS_FLAT);
% [counts3, binCenters3] = hist(COSTS_SCALED);
% plot(binCenters1, counts1, 'r-','LineWidth',5);
% hold on;
% plot(binCenters2, counts2, 'g-','LineWidth',5);
% plot(binCenters3, counts3, 'b-','LineWidth',5);

[n,x] = hist(COSTS_UNINSURED);
[n2,x2] = hist(COSTS_FLAT,[mean(COSTS_FLAT)+2500,mean(COSTS_FLAT)-2500]);
[n3,x3] = hist(COSTS_SCALED);
bar(x,n,'hist','facealpha',.3)
h=bar(x,n,'hist','facealpha',.8);
set(h,'facecolor','g')
hold on;
h=bar(x2,n2,'hist','facealpha',.8);
hold off
set(h,'facecolor','c')
hold on;
h=bar(x3,n3,'hist');
```

```
hold off
set(h,'facecolor','r','facealpha',.8)

grid on;
legend1 = sprintf('Mean Uninsured = %.3f', mean(COSTS_UNINSURED));
legend2 = sprintf('Mean Fully Insured = %.3f', mean(COSTS_FLAT));
legend3 = sprintf('Mean with Scaling Deductible = %.3f', mean(COSTS_SCALED));
legend({legend1, legend2, legend3});

title('Zevenheuvelenloop Spending');
xlabel(['Cost over ' int2str(NUMBER_OF_YEARS) ' years(Euros)']);
ylabel('Frequency');
```

Question 4:
MATLAB Program 2: Simulation of an Event with Multiple Races

```
clear all

SAMPLE_WR_PROPS=[1.000, 0.653, 0.754,0.731, 0.198, 0.488, 0.622, 0.560, 0.709, 0.608,
0.308, 0.260, 0.450, 0.374, 0.433, 1.043, 0.714, 0.284, 0.538, 0.417];

NUMBER_OF_YEARS=10000;
COST_PER_YEAR=0;
INSURANCE_PROFIT_MARGIN=1.1;
INSURANCE_FLAT_RATE=5000;
PRIZE_MONEY=25000;
PREMIUM_FRACTION=.2;


%% creates distribution using deductible scaled to p-hat
for i=1:NUMBER_OF_YEARS
    for j=1:length(SAMPLE_WR_PROPS)

COST_PER_YEAR=COST_PER_YEAR+INSURANCE_PROFIT_MARGIN*(SAMPLE_WR_PROPS(j))*PRIZE_MONEY*P
REMIUM_FRACTION;
        if rand<SAMPLE_WR_PROPS(j)
            COST_PER_YEAR=COST_PER_YEAR+(1-SAMPLE_WR_PROPS(j))*PRIZE_MONEY;
        end
    end
    COSTS_SCALED(i)=COST_PER_YEAR;
    COST_PER_YEAR=0;
end

figure();
hist(COSTS_SCALED);
set(get(gca,'child'),'FaceColor','r');
title('Insurance premium scaled to p-hat')
xlabel('Cost(Euros)');
ylabel('Frequency');
CI_SCALED=[prctile(COSTS_SCALED,.05), prctile(COSTS_SCALED,.95)]


%% insurance with no deductible
```

```
for i=1:NUMBER_OF_YEARS
    for j=1:length(SAMPLE_WR_PROPS)
        COST_PER_YEAR=COST_PER_YEAR+INSURANCE_FLAT_RATE;
    end
    COSTS_FLAT(i)=COST_PER_YEAR;
    COST_PER_YEAR=0;
end

figure();
hist(COSTS_FLAT);
set(get(gca,'child'),'FaceColor','b');
title('Insurance premium with no scaled deductible');
xlabel('Cost(Euros)');
ylabel('Frequency');
CI_FLAT=[prctile(COSTS_FLAT,.05), prctile(COSTS_FLAT,.95)]



%% creates distribution with no insurance at all
for i=1:NUMBER_OF_YEARS
    for j=1:length(SAMPLE_WR_PROPS)
        if SAMPLE_WR_PROPS(j)<rand
            COST_PER_YEAR=COST_PER_YEAR+PRIZE_MONEY;
        end
    end
    COSTS_UNINSURED(i)=COST_PER_YEAR;
    COST_PER_YEAR=0;
end

figure();
hist(COSTS_UNINSURED);
set(get(gca,'child'),'FaceColor','g');
title('No insurance');
xlabel('Cost(Euros)');
ylabel('Frequency');
CI_UNINSURED=[prctile(COSTS_UNINSURED,.05), prctile(COSTS_UNINSURED,.95)]
```

MATLAB Program 3: Analysis of the Difference Between going Insured and Uninsured

```
clear all
x=(0:.01:15);
y=(300000:100:400000);


[xx,yy]=meshgrid(x,y);
for i=1:size(xx,1)
    for j=1:size(yy,2)
        zz(i,j)=2*1171.88*xx(i,j)*(1+2.3826*50000/yy(i,j));
    end
end


figure();
```

```
surf(xx,yy,zz);
xlabel('time (years)');
ylabel('budget (euros)');
zlabel('risk (uninsured)');
shading interp
colorbar
%%


for i=1:size(xx,1)
    for j=1:size(yy,2)
        zz2(i,j)=2*1171.88*xx(i,j)*1.35;
    end
end


hold on;
surf(xx,yy,zz2);
xlabel('time (years)');
ylabel('budget (euros)');
zlabel('risk (uninsured)');

diff=(zz-zz2)>0;
for i=1:size(diff,1)-1
    if diff(i,2)<1
        x_val=i
        approx=300000+100*x_val;
        break
    end
end
```

MATLAB Program 4: Local Sensitivity Analysis using Partial Derivatives

```
clear all

syms b q s beta d marg
xb=abs((2*(b*q+.5*(s-beta))*(d+marg)*q)/(beta*(d-(1-marg))));
xq=abs((2*(b*q+.5*(s-beta))*b*(d+marg))/(beta*(d-(1-marg))));
xbeta=abs((-b*(d+marg)*q*(s+b*q))/((beta^2)*(d-(1-marg))));
xd=abs((-b*q*(s+b*q-beta))/((d-(1-marg))^2*beta));
xs=abs((b*(d+marg)*q)/(beta*(d-(1-marg))));

%% sensitivity analysis for xb

dxb_db=diff(xb,b);
b=25000;
q=.0469;
s=338000;
beta=340000;
d=.1559;
marg=.102;
FINALdxb_db=double(subs(dxb_db));
```

```
syms q
dxb_dq=diff(xb,q);
q=.0469;
FINALdxb_dq=double(subs(dxb_dq));

syms s
dxb_ds=diff(xb,s);
s=338000;
FINALdxb_ds=double(subs(dxb_ds));

syms beta
dxb_dbeta=diff(xb,beta);
beta=340000;
FINALdxb_dbeta=double(subs(dxb_dbeta));

syms d
dxb_dd=diff(xb,d);
d=.1559;
FINALdxb_dd=double(subs(dxb_dd));

%% sensitivity analysis for xq

syms b q s beta d marg
dxq_db=diff(xq,b);
b=25000;
q=.0469;
s=338000;
beta=340000;
d=.1559;
marg=.102;
FINALdxq_db=double(subs(dxq_db));

syms q
dxq_dq=diff(xq,q);
q=.0469;
FINALdxq_dq=double(subs(dxq_dq));

syms s
dxq_ds=diff(xq,s);
s=338000;
FINALdxq_ds=double(subs(dxq_ds));

syms beta
dxq_dbeta=diff(xq,beta);
beta=340000;
FINALdxq_dbeta=double(subs(dxq_dbeta));

syms d
dxq_dd=diff(xq,d);
d=.1559;
FINALdxq_dd=double(subs(dxq_dd));

%% sensitivity analysis for xbeta
```

```
syms b q s beta d marg
dxbeta_db=diff(xbeta,b);
b=25000;
q=.0469;
s=338000;
beta=340000;
d=.1559;
marg=.102;
FINALdxbeta_db=double(subs(dxbeta_db));

syms q
dxbeta_dq=diff(xbeta,q);
q=.0469;
FINALdxbeta_dq=double(subs(dxbeta_dq));

syms s
dxbeta_ds=diff(xbeta,s);
s=338000;
FINALdxbeta_ds=double(subs(dxbeta_ds));

syms beta
dxbeta_dbeta=diff(xbeta,beta);
beta=340000;
FINALdxbeta_dbeta=double(subs(dxbeta_dbeta));

syms d
dxbeta_dd=diff(xbeta,d);
d=.1559;
FINALdxbeta_dd=double(subs(dxbeta_dd));

%% sensitivity analysis for xd

syms b q s beta d marg
dxd_db=diff(xd,b);
b=25000;
q=.0469;
s=338000;
beta=340000;
d=.1559;
marg=.102;
FINALdxd_db=double(subs(dxd_db));

syms q
dxd_dq=diff(xd,q);
q=.0469;
FINALdxd_dq=double(subs(dxd_dq));

syms s
dxd_ds=diff(xd,s);
s=338000;
FINALdxd_ds=double(subs(dxd_ds));

syms beta
dxd_dbeta=diff(xd,beta);
```

```
beta=340000;
FINALdxd_dbeta=double(subs(dxd_dbeta));

syms d
dxd_dd=diff(xd,d);
d=.1559;
FINALdxd_dd=double(subs(dxd_dd));

%% sensitivity analysis for xs

syms b q s beta d marg
dxs_db=diff(xs,b);
b=25000;
q=.0469;
s=338000;
beta=340000;
d=.1559;
marg=.102;
FINALdxs_db=double(subs(dxs_db));

syms q
dxs_dq=diff(xs,q);
q=.0469;
FINALdxs_dq=double(subs(dxs_dq));

syms s
dxs_ds=diff(xs,s);
s=338000;
FINALdxs_ds=double(subs(dxs_ds));

syms beta
dxs_dbeta=diff(xs,beta);
beta=340000;
FINALdxs_dbeta=double(subs(dxs_dbeta));

syms d
dxs_dd=diff(xs,d);
d=.1559;
FINALdxs_dd=double(subs(dxs_dd));
```

MATLAB Program 5: Global Sensitivity Analysis using Sampling Based Approach

```
clear all

reps=5000;
marg=.102;
i=1;

syms b q s beta d
xb=((2*(b*q+.5*(s-beta))*(d+marg)*q)/(beta*(d-(1-marg))));
xq=((2*(b*q+.5*(s-beta))*b*(d+marg))/(beta*(d-(1-marg))));
xbeta=((-b*(d+marg)*q*(s+b*q))/((beta^2)*(d-(1-marg))));
xd=((-b*q*(s+b*q-beta))/((d-(1-marg))^2*beta));
```

```
xs=((b*(d+marg)*q)/(beta*(d-(1-marg))));

%% analysis for xb

for i=1:reps
    b=normrnd(25000,2500);
    q=normrnd(.0469,.00469);
    s=normrnd(338000,33800);
    beta=normrnd(340000,34000);
    d=normrnd(.1559,.01559);

    vars(1,i)=b;
    vars(2,i)=q;
    vars(3,i)=s;
    vars(4,i)=beta;
    vars(5,i)=d;
    vars(6,i)=subs(xb);
end

xbb=corrcoef(vars(1,:),vars(6,:));
xbq=corrcoef(vars(2,:),vars(6,:));
xbs=corrcoef(vars(3,:),vars(6,:));
xbbeta=corrcoef(vars(4,:),vars(6,:));
xbd=corrcoef(vars(5,:),vars(6,:));

figure();
scatter(vars(1,:),vars(6,:),'r.');
xlabel('Prize Money (Euros)');
ylabel('Weight of xb');

figure();
scatter(vars(2,:),vars(6,:),'m.');
xlabel('Probability of a World Record Being Broken');
ylabel('Weight of xb');

figure();
scatter(vars(3,:),vars(6,:),'k.');
xlabel('Expected Spending on the Event Excluding Prize Money (Euros)');
ylabel('Weight of xb');

figure();
scatter(vars(4,:),vars(6,:),'g.');
xlabel('Budget for the Event (Euros)');
ylabel('Weight of xb');

figure();
scatter(vars(5,:),vars(6,:),'b.');
xlabel('Insurance Operating Cost per Euro Earned as Income');
ylabel('Weight of xb');


%% analysis for xq

for i=1:reps
```

```
    b=normrnd(25000,2500);
    q=normrnd(.0469,.00469);
    s=normrnd(338000,33800);
    beta=normrnd(340000,34000);
    d=normrnd(.1559,.01559);

    vars(1,i)=b;
    vars(2,i)=q;
    vars(3,i)=s;
    vars(4,i)=beta;
    vars(5,i)=d;
    vars(6,i)=subs(xq);

end

xqb=corrcoef(vars(1,:),vars(6,:));
xqq=corrcoef(vars(2,:),vars(6,:));
xqs=corrcoef(vars(3,:),vars(6,:));
xqbeta=corrcoef(vars(4,:),vars(6,:));
xqd=corrcoef(vars(5,:),vars(6,:));

figure();
scatter(vars(1,:),vars(6,:),'r.');
xlabel('Prize Money (Euros)');
ylabel('Weight of xq');

figure();
scatter(vars(2,:),vars(6,:),'m.');
xlabel('Probability of a World Record Being Broken');
ylabel('Weight of xq');

figure();
scatter(vars(3,:),vars(6,:),'k.');
xlabel('Expected Spending on the Event Excluding Prize Money (Euros)');
ylabel('Weight of xq');

figure();
scatter(vars(4,:),vars(6,:),'g.');
xlabel('Budget for the Event (Euros)');
ylabel('Weight of xq');

figure();
scatter(vars(5,:),vars(6,:),'b.');
xlabel('Insurance Operating Cost per Euro Earned as Income');
ylabel('Weight of xq');

%% analysis for xs

for i=1:reps
    b=normrnd(25000,2500);
    q=normrnd(.0469,.00469);
    s=normrnd(338000,33800);
    beta=normrnd(340000,34000);
    d=normrnd(.1559,.01559);
    vars(1,i)=b;
    vars(2,i)=q;
    vars(3,i)=s;
    vars(4,i)=beta;
    vars(5,i)=d;
    vars(6,i)=subs(xs);

end

xsb=corrcoef(vars(1,:),vars(6,:));
xsq=corrcoef(vars(2,:),vars(6,:));
xss=corrcoef(vars(3,:),vars(6,:));
xsbeta=corrcoef(vars(4,:),vars(6,:));
xsd=corrcoef(vars(5,:),vars(6,:));

figure();
scatter(vars(1,:),vars(6,:),'r.');
xlabel('Prize Money (Euros)');
ylabel('Weight of xs');

figure();
scatter(vars(2,:),vars(6,:),'m.');
xlabel('Probability of a World Record Being Broken');
ylabel('Weight of xs');

figure();
scatter(vars(3,:),vars(6,:),'k.');
xlabel('Expected Spending on the Event Excluding Prize Money (Euros)');
ylabel('Weight of xs');

figure();
scatter(vars(4,:),vars(6,:),'g.');
xlabel('Budget for the Event (Euros)');
ylabel('Weight of xs');

figure();
scatter(vars(5,:),vars(6,:),'b.');
xlabel('Insurance Operating Cost per Euro Earned as Income');
ylabel('Weight of xs');

%% analysis for xbeta

for i=1:reps
    b=normrnd(25000,2500);
    q=normrnd(.0469,.00469);
    s=normrnd(338000,33800);
    beta=normrnd(340000,34000);
    d=normrnd(.1559,.01559);

    vars(1,i)=b;
    vars(2,i)=q;
    vars(3,i)=s;
    vars(4,i)=beta;
    vars(5,i)=d;
    vars(6,i)=subs(xbeta);

end

xbetab=corrcoef(vars(1,:),vars(6,:));
xbetaq=corrcoef(vars(2,:),vars(6,:));
xbetas=corrcoef(vars(3,:),vars(6,:));
xbetabeta=corrcoef(vars(4,:),vars(6,:));
xbetad=corrcoef(vars(5,:),vars(6,:));

figure();
scatter(vars(1,:),vars(6,:),'r.');
xlabel('Prize Money (Euros)');
ylabel('Weight of xbeta');

figure();
scatter(vars(2,:),vars(6,:),'m.');
xlabel('Probability of a World Record Being Broken');
ylabel('Weight of xbeta');

figure();
scatter(vars(3,:),vars(6,:),'k.');
xlabel('Expected Spending on the Event Excluding Prize Money (Euros)');
ylabel('Weight of xbeta');

figure();
scatter(vars(4,:),vars(6,:),'g.');
xlabel('Budget for the Event (Euros)');
ylabel('Weight of xbeta');

figure();
scatter(vars(5,:),vars(6,:),'b.');
xlabel('Insurance Operating Cost per Euro Earned as Income');
ylabel('Weight of xbeta');

%% analysis for xd

for i=1:reps
    b=normrnd(25000,2500);
    q=normrnd(.0469,.00469);
    s=normrnd(338000,33800);
    beta=normrnd(340000,34000);
    d=normrnd(.1559,.01559);

    vars(1,i)=b;
    vars(2,i)=q;
    vars(3,i)=s;
    vars(4,i)=beta;
    vars(5,i)=d;
    vars(6,i)=subs(xd);

end
```

```
xdb=corrcoef(vars(1,:),vars(6,:));
xdq=corrcoef(vars(2,:),vars(6,:));
xds=corrcoef(vars(3,:),vars(6,:));
xdbeta=corrcoef(vars(4,:),vars(6,:));
xdd=corrcoef(vars(5,:),vars(6,:));

figure();
scatter(vars(1,:),vars(6,:),'r.');
xlabel('Prize Money (Euros)');
ylabel('Weight of xd');

figure();
scatter(vars(2,:),vars(6,:),'m.');
xlabel('Probability of a World Record Being Broken');
ylabel('Weight of xd');

figure();
scatter(vars(3,:),vars(6,:),'k.');
xlabel('Expected Spending on the Event Excluding Prize Money (Euros)');
ylabel('Weight of xd');

figure();
scatter(vars(4,:),vars(6,:),'g.');
xlabel('Budget for the Event (Euros)');
ylabel('Weight of xd');

figure();
scatter(vars(5,:),vars(6,:),'b.');
xlabel('Insurance Operating Cost per Euro Earned as Income');
ylabel('Weight of xd');
```

MATLAB Program 6: Using Deductibles

```
clear all

b=25000;
t=1;
beta=200000;
s=190000;
d=.127;

qs=[0.05,0.1508,0.1462,0.0396,0.0976,0.1244,0.1120,0.1417,0.0514,0.0899,0.0865,0.0650,
0.0410,0.0885,0.0816,0.0566,0.0506,0.0673,0.0724,0.0752,0.1428571429,0.1305263158,0.05
684210526,0.07578947368,0.1178947368,0.1075268817,0.1170212766,0.44,0.02765957447,0.07
474747475,0.2086956522,0.08333333333,0.08666666667,0.1021276596,0.06,0.04888888889,0.0
7843137255,0.09574468085,0.05647058824,0.0875];
syms x q
rd=(b-x)*q*t*(1+(.102+d)/(.898-d)+((((b*q-q*x)/(.898-d)))*(s+q*x))/((b-x)*q*(b-x))*x/b
eta);

disp('got to loop');
for i=1:length(qs)
    i
    q=qs(i);
    x=linspace(0,b,b*1000);
    f= @(x) subs(rd);
    [minx(i),~]=fminbnd(f,0,b);

end
```

User-Facing Programs

Question 3:

Objective-C Code for User-Facing Program #1: Prize Indemnification Insurance Risk Calculator

immc2016021AppDelegate.h

```
//  immc5538AppDelegate.h
//  IMMC 2016 -- Team 2016021
//
//  Created by [name redacted] on 3/27/16.
//  Copyright (c) 2016 IMMC Team 2016021. All rights
reserved. //

#import <Cocoa/Cocoa.h>

@interface immc5538AppDelegate : NSObject <NSApplicationDelegate> {

    IBOutlet NSTextField *b;
    IBOutlet NSTextField *q;
    IBOutlet NSTextField *t;
    IBOutlet NSTextField *e;
    IBOutlet NSTextField *s;
    IBOutlet NSTextField *beta;

    IBOutlet NSTextField *result;

}

@property (assign) IBOutlet NSWindow *window;

- (IBAction)computerisk:(id)sender;
- (IBAction)reset:(id)sender;

@end
```

immc2016021AppDelegate.m

```
//  immc2016021AppDelegate.m
//  IMMC 2016 -- Team 2016021
//
//  Created by [name redacted] on 3/27/16.
//  Copyright (c) 2016 IMMC Team 2016021. All rights
reserved. //

#import "immc5538AppDelegate.h"

@implementation immc5538AppDelegate
```

```
- (void)applicationDidFinishLaunching:(NSNotification *)aNotification
{
    [result setStringValue:@""];
    [result display];

    [b setStringValue:@""];
    [b display];

    [q setStringValue:@""];
    [q display];

    [t setStringValue:@""];
    [t display];

    [e setStringValue:@""];
    [e display];

    [s setStringValue:@""];
    [s display];

    [beta setStringValue:@""];
    [beta display];

}

- (IBAction)computerisk:(id)sender {

    //Note to IMMC judges: all variables used in this code are consistent with the
variables we use in our paper. Therefore, you may refer to the variable definition
statement at the top of our paper for the definition of variables used in this code.

    //Capture text input into NSStrings

    NSString *bstring = [NSString stringWithFormat:@"%@", [b stringValue]];
    NSString *qstring = [NSString stringWithFormat:@"%@", [q stringValue]];
    NSString *tstring = [NSString stringWithFormat:@"%@", [t stringValue]];
    NSString *estring = [NSString stringWithFormat:@"%@", [e stringValue]];
    NSString *sstring = [NSString stringWithFormat:@"%@", [s stringValue]];
    NSString *betastring = [NSString stringWithFormat:@"%@", [beta stringValue]];

    //Turn the newly created NSStrings into floats

    float bfloat = [bstring floatValue];
    float qfloat = [qstring floatValue];
    float tfloat = [tstring floatValue];
    float efloat = [estring floatValue];
    float sfloat = [sstring floatValue];
    float betafloat = [betastring floatValue];

    //Compute risk
    float a;
    float m;
    float riskwithinsurance;
    float riskwithoutinsurance;
    float gamma;

    a = qfloat*bfloat; //average cost of the bonus

    m = efloat - (qfloat * bfloat); //cost of the premium minus the average cost of 
the bonus equals excess cost added to the premium by the insurance company

    gamma = m * (sfloat + a)/(a*bfloat);


    riskwithoutinsurance = a*tfloat*(1+gamma*(bfloat / betafloat));
    riskwithinsurance = a*tfloat*(1+m/a);


    //Compute relative financial exposure

    float relativefinancialexposure;

    if (riskwithinsurance < riskwithoutinsurance) {

        relativefinancialexposure = 
(riskwithoutinsurance-riskwithinsurance)/riskwithinsurance*100;

        NSString *resultString = [NSString stringWithFormat:@"Buy 
insurance!\nFinancial exposure difference: %.2f%%.",relativefinancialexposure];
        [result setStringValue:resultString];
        [result display];

    }

    else

        if (riskwithoutinsurance < riskwithinsurance) {

            relativefinancialexposure = 
(riskwithoutinsurance-riskwithinsurance)/riskwithinsurance*100;

            NSString *resultString = [NSString stringWithFormat:@"Do not buy 
insurance!\n Financial exposure difference: %.2f%%.",relativefinancialexposure];
            [result setStringValue:resultString];
            [result display];

    }

    else

        if (riskwithoutinsurance == riskwithinsurance) {

            [result setStringValue:@"The risks are equal. Use your judgement to 
determine the best option."];
            [result display];

        }
    else


        [result setStringValue:@"Error"];
        [result display];




}

- (IBAction)reset:(id)sender {

    [result setStringValue:@""];
    [result display];

    [b setStringValue:@""];
    [b display];

    [q setStringValue:@""];
    [q display];

    [t setStringValue:@""];
    [t display];

    [e setStringValue:@""];
    [e display];

    [s setStringValue:@""];
    [s display];

    [beta setStringValue:@""];
    [beta display];

}
@end
```

Question 4:
Objective-C Code for User-Facing Program #2: Weight Calculations

<u>immc2016021AppDelegate.h</u>

```
//  immc2016021AppDelegate.h
//  IMMC 2016 -- Team 2016021 Weighting
//
//  Created by [name redacted] on 3/27/16.
//  Copyright (c) 2016 IMMC Team 2016021. All rights
reserved. //

#import <Cocoa/Cocoa.h>

@interface immc5538AppDelegate : NSObject <NSApplicationDelegate> {

    IBOutlet NSTextField *b;
    IBOutlet NSTextField *q;
    IBOutlet NSTextField *s;
    IBOutlet NSTextField *beta;

    IBOutlet NSTextField *resultb;
    IBOutlet NSTextField *resultq;
    IBOutlet NSTextField *resultd;
    IBOutlet NSTextField *results;
    IBOutlet NSTextField *resultbeta;

    IBOutlet NSPopUpButton *continentselector;



}

@property (assign) IBOutlet NSWindow *window;

- (IBAction)computeweights:(id)sender;
- (IBAction)reset:(id)sender;

@end
```

immc2016021AppDelegate.m

```
//  immc2016021AppDelegate.m
//  IMMC 2016 -- Team 2016021 Weighting
//
//  Created by [name redacted] on 3/27/16.
//  Copyright (c) 2016 IMMC Team 2016021. All rights
reserved. //

#import "immc5538AppDelegate.h"


@implementation immc5538AppDelegate

- (void)applicationDidFinishLaunching:(NSNotification *)aNotification
{



    [resultb setStringValue:@""];
    [resultb display];

    [resultq setStringValue:@""];
    [resultq display];

    [results setStringValue:@""];
    [results display];

    [resultd setStringValue:@""];
    [resultd display];

    [resultbeta setStringValue:@""];
    [resultbeta display];
    [b setStringValue:@""];
    [b display];

    [q setStringValue:@""];
    [q display];

    [s setStringValue:@""];
    [s display];

    [beta setStringValue:@""];
    [beta display];

    NSString *africa = @"Africa";

    NSString *asia = @"Asia";

    NSString *australia = @"Australia";

    NSString *europe = @"Europe";

    NSString *northamerica = @"North America";

    NSString *southamerica = @"South America";

    NSMutableArray *continentArray;

    continentArray= [NSMutableArray arrayWithObjects: africa, asia, australia, europe,
northamerica, southamerica, nil];

    [continentselector removeAllItems];

    [continentselector addItemsWithTitles: continentArray];

}


- (IBAction)computeweights:(id)sender {


    //Note to IMMC judges: all variables used in this code are consistent with the
variables we use in our paper. Therefore, you may refer to the variable definition
statement at the top of our paper for the definition of variables used in this code.


    //Capture the user's continent selection to determine d
    [[continentselector itemAtIndex:0]setTag:0];
    [[continentselector itemAtIndex:1]setTag:1];
    [[continentselector itemAtIndex:2]setTag:2];
    [[continentselector itemAtIndex:3]setTag:3];
    [[continentselector itemAtIndex:4]setTag:4];
    [[continentselector itemAtIndex:5]setTag:5];
    NSInteger continentSelected = [[continentselector selectedItem]tag]; //Store that
value in a variable continentSelected


    //Capture text input into NSStrings

    NSString *bstring = [NSString stringWithFormat:@"%@", [b stringValue]];
    NSString *qstring = [NSString stringWithFormat:@"%@", [q stringValue]];
    NSString *sstring = [NSString stringWithFormat:@"%@", [s stringValue]];
    NSString *betastring = [NSString stringWithFormat:@"%@", [beta stringValue]];

    //Turn the newly created NSStrings into floats

    float bfloat = [bstring floatValue];
    float qfloat = [qstring floatValue];
    float sfloat = [sstring floatValue];
    float betafloat = [betastring floatValue];

    //Compute weights

    float a;

    a = qfloat*bfloat; //average cost of the bonus

    float dfloat;

    if (continentSelected == 0) {

        //Use the d ratio for African insurance companies

        dfloat = .1236;
    }

    if (continentSelected == 1) {

        //Use the d ratio for Asian insurance companies

        dfloat = .1564;
    }

    if (continentSelected == 2) {

        //Use the d ratio for Australian insurance companies

        dfloat = .0783;
    }

    if (continentSelected == 3) {

        //Use the d ratio for European insurance companies

        dfloat = .1559;
    }
    if (continentSelected == 4) {

        //Use the d ratio for North American insurance companies

        dfloat = .127;
    }

    if (continentSelected == 5) {

        //Use the d ratio for South American insurance companies

        dfloat = .1613;
    }

    //Calculate unstandardized weights

    float xb;
    xb =
fabsf((2*(bfloat*qfloat+((sfloat-betafloat)/2))*(dfloat+.102)*qfloat)/(betafloat*(dflo
at-.898)));

    float xq;
    xq =
fabsf((2*(bfloat*qfloat+((sfloat-betafloat)/2))*(dfloat+.102)*bfloat)/(betafloat*(dflo
at-.898)));

    float xbeta;
    xbeta =
fabsf(((-1)*bfloat*(dfloat+.102)*qfloat*(sfloat+bfloat*qfloat))/(betafloat*betafloat*(
dfloat-.898)));

    float xd;
    xd =
fabsf((-1*bfloat*qfloat*(sfloat+bfloat*qfloat-betafloat))/((dfloat-.898)*(dfloat-.898)
*betafloat));

    float xs;
    xs = fabsf((bfloat*(dfloat+.102)*qfloat)/(betafloat*(dfloat-.898)));

    //Calculate sum of unstandardized weights

    float sumofunstandardizedweights;
    sumofunstandardizedweights = xb+xq+xd+xs+xbeta;

    //Standardize the weights

    float xbstd;
    xbstd = xb/sumofunstandardizedweights;
    xbstd = xbstd*100;

    float xqstd;
    xqstd = xq/sumofunstandardizedweights;
    xqstd = xqstd*100;
    float xdstd;
    xdstd = xd/sumofunstandardizedweights;
    xdstd = xdstd*100;

    float xsstd;
    xsstd = xs/sumofunstandardizedweights;
    xsstd = xsstd*100;

    float xbetastd;
    xbetastd = xbeta/sumofunstandardizedweights;
    xbetastd = xbetastd*100;

    //Create text output for the user

    NSString *resultbstring = [NSString stringWithFormat:@"The weight given to the
total prize value should be %f%%",xbstd];
    [resultb setStringValue:resultbstring];
    [resultb display];

    NSString *resultqstring = [NSString stringWithFormat:@"The weight given to the
probability of record-breaking should be %f%%",xqstd];
    [resultq setStringValue:resultqstring];
    [resultq display];

    NSString *resultdstring = [NSString stringWithFormat:@"The weight given to the
operating costs per Euros earned as income should be %f%%",xdstd];
    [resultd setStringValue:resultdstring];
    [resultd display];

    NSString *resultsstring = [NSString stringWithFormat:@"The weight given to the
total spending should be %f%%",xsstd];
    [results setStringValue:resultsstring];
    [results display];

    NSString *resultbetastring = [NSString stringWithFormat:@"The weight given to the
total budget should be %f%%",xbetastd];
    [resultbeta setStringValue:resultbetastring];
    [resultbeta display];


}

- (IBAction)reset:(id)sender {

    [resultb setStringValue:@""];
    [resultb display];

    [resultq setStringValue:@""];
    [resultq display];

    [results setStringValue:@""];
    [results display];

    [resultd setStringValue:@""];
    [resultd display];

    [resultbeta setStringValue:@""];
    [resultbeta display];

    [b setStringValue:@""];
    [b display];

    [q setStringValue:@""];
    [q display];

    [s setStringValue:@""];
    [s display];

    [beta setStringValue:@""];
    [beta display];
}
@end
```

Question 5:
Objective-C Code for User-Facing Program #3: Decision-Scheme Automation for 40 Events

immc2016021AppDelegate.h

```
//  immc2016021AppDelegate.h
//  IMMC 2016 -- Team 2016021 Question 5
//
//  Created by [name redacted] on 3/28/16.
//  Copyright (c) 2016 IMMC Team 2016021. All rights
reserved. //

#import <Cocoa/Cocoa.h>

@interface immc5538AppDelegate : NSObject <NSApplicationDelegate> {


    IBOutlet NSTextField *s;
    IBOutlet NSTextField *beta;

    //Declare IBOutlets for men's events

    IBOutlet NSTextField *bm1;
    IBOutlet NSTextField *qm1;
    IBOutlet NSTextField *em1;

    IBOutlet NSTextField *bm2;
    IBOutlet NSTextField *qm2;
    IBOutlet NSTextField *em2;

    IBOutlet NSTextField *bm3;
    IBOutlet NSTextField *qm3;
    IBOutlet NSTextField *em3;

    IBOutlet NSTextField *bm4;
    IBOutlet NSTextField *qm4;
    IBOutlet NSTextField *em4;

    IBOutlet NSTextField *bm5;
    IBOutlet NSTextField *qm5;
    IBOutlet NSTextField *em5;

    IBOutlet NSTextField *bm6;
    IBOutlet NSTextField *qm6;
    IBOutlet NSTextField *em6;

    IBOutlet NSTextField *bm7;
    IBOutlet NSTextField *qm7;
    IBOutlet NSTextField *em7;

    IBOutlet NSTextField *bm8;
    IBOutlet NSTextField *qm8;
    IBOutlet NSTextField *em8;

    IBOutlet NSTextField *bm9;
    IBOutlet NSTextField *qm9;
    IBOutlet NSTextField *em9;

    IBOutlet NSTextField *bm10;
    IBOutlet NSTextField *qm10;
    IBOutlet NSTextField *em10;

    IBOutlet NSTextField *bm11;
    IBOutlet NSTextField *qm11;
    IBOutlet NSTextField *em11;

    IBOutlet NSTextField *bm12;
    IBOutlet NSTextField *qm12;
    IBOutlet NSTextField *em12;

    IBOutlet NSTextField *bm13;
    IBOutlet NSTextField *qm13;
    IBOutlet NSTextField *em13;

    IBOutlet NSTextField *bm14;
    IBOutlet NSTextField *qm14;
    IBOutlet NSTextField *em14;

    IBOutlet NSTextField *bm15;
    IBOutlet NSTextField *qm15;
    IBOutlet NSTextField *em15;

    IBOutlet NSTextField *bm16;
    IBOutlet NSTextField *qm16;
    IBOutlet NSTextField *em16;

    IBOutlet NSTextField *bm17;
    IBOutlet NSTextField *qm17;
    IBOutlet NSTextField *em17;
    IBOutlet NSTextField *bm18;
    IBOutlet NSTextField *qm18;
    IBOutlet NSTextField *em18;

    IBOutlet NSTextField *bm19;
    IBOutlet NSTextField *qm19;
    IBOutlet NSTextField *em19;

    IBOutlet NSTextField *bm20;
    IBOutlet NSTextField *qm20;
    IBOutlet NSTextField *em20;

    //Declare IBOutlets for women's events

    IBOutlet NSTextField *bw1;
    IBOutlet NSTextField *qw1;
    IBOutlet NSTextField *ew1;

    IBOutlet NSTextField *bw2;
    IBOutlet NSTextField *qw2;
    IBOutlet NSTextField *ew2;

    IBOutlet NSTextField *bw3;
    IBOutlet NSTextField *qw3;
    IBOutlet NSTextField *ew3;

    IBOutlet NSTextField *bw4;
    IBOutlet NSTextField *qw4;
    IBOutlet NSTextField *ew4;

    IBOutlet NSTextField *bw5;
    IBOutlet NSTextField *qw5;
    IBOutlet NSTextField *ew5;

    IBOutlet NSTextField *bw6;
    IBOutlet NSTextField *qw6;
    IBOutlet NSTextField *ew6;

    IBOutlet NSTextField *bw7;
    IBOutlet NSTextField *qw7;
    IBOutlet NSTextField *ew7;

    IBOutlet NSTextField *bw8;
    IBOutlet NSTextField *qw8;
    IBOutlet NSTextField *ew8;

    IBOutlet NSTextField *bw9;
    IBOutlet NSTextField *qw9;
    IBOutlet NSTextField *ew9;

    IBOutlet NSTextField *bw10;
    IBOutlet NSTextField *qw10;
    IBOutlet NSTextField *ew10;
```

```
    IBOutlet NSTextField *bw11;
    IBOutlet NSTextField *qw11;
    IBOutlet NSTextField *ew11;

    IBOutlet NSTextField *bw12;
    IBOutlet NSTextField *qw12;
    IBOutlet NSTextField *ew12;

    IBOutlet NSTextField *bw13;
    IBOutlet NSTextField *qw13;
    IBOutlet NSTextField *ew13;

    IBOutlet NSTextField *bw14;
    IBOutlet NSTextField *qw14;
    IBOutlet NSTextField *ew14;

    IBOutlet NSTextField *bw15;
    IBOutlet NSTextField *qw15;
    IBOutlet NSTextField *ew15;

    IBOutlet NSTextField *bw16;
    IBOutlet NSTextField *qw16;
    IBOutlet NSTextField *ew16;

    IBOutlet NSTextField *bw17;
    IBOutlet NSTextField *qw17;
    IBOutlet NSTextField *ew17;

    IBOutlet NSTextField *bw18;
    IBOutlet NSTextField *qw18;
    IBOutlet NSTextField *ew18;

    IBOutlet NSTextField *bw19;
    IBOutlet NSTextField *qw19;
    IBOutlet NSTextField *ew19;

    IBOutlet NSTextField *bw20;
    IBOutlet NSTextField *qw20;
    IBOutlet NSTextField *ew20;

}

@property (assign) IBOutlet NSWindow *window;

- (IBAction)reset:(id)sender;
- (IBAction)go:(id)sender;


@end
```

<u>immc2016021AppDelegate.m</u>

```
//  immc2016021AppDelegate.m
//  IMMC 2016 -- Team 2016021
Question 5 //
```

```
//  Created by [name redacted] on 3/28/16.
//  Copyright (c) 2016 IMMC Team 2016021. All rights
reserved. //

#import "immc2016021AppDelegate.h"

@implementation immc2016021AppDelegate

- (void)applicationDidFinishLaunching:(NSNotification *)aNotification
{


}

- (IBAction) go:(id)sender {

    NSString *sstring = [NSString stringWithFormat:@"%@", [s stringValue]];
    NSString *betastring = [NSString stringWithFormat:@"%@", [beta stringValue]];

    float sfloat = [sstring floatValue];
    float betafloat = [betastring floatValue];

    //Convert the men's text inputs into NSStrings

    NSString *bm1string = [NSString stringWithFormat:@"%@", [bm1 stringValue]];
    NSString *qm1string = [NSString stringWithFormat:@"%@", [qm1 stringValue]];
    NSString *em1string = [NSString stringWithFormat:@"%@", [em1 stringValue]];

    NSString *bm2string = [NSString stringWithFormat:@"%@", [bm2 stringValue]];
    NSString *qm2string = [NSString stringWithFormat:@"%@", [qm2 stringValue]];
    NSString *em2string = [NSString stringWithFormat:@"%@", [em2 stringValue]];

    NSString *bm3string = [NSString stringWithFormat:@"%@", [bm3 stringValue]];
    NSString *qm3string = [NSString stringWithFormat:@"%@", [qm3 stringValue]];
    NSString *em3string = [NSString stringWithFormat:@"%@", [em3 stringValue]];

    NSString *bm4string = [NSString stringWithFormat:@"%@", [bm4 stringValue]];
    NSString *qm4string = [NSString stringWithFormat:@"%@", [qm4 stringValue]];
    NSString *em4string = [NSString stringWithFormat:@"%@", [em4 stringValue]];

    NSString *bm5string = [NSString stringWithFormat:@"%@", [bm5 stringValue]];
    NSString *qm5string = [NSString stringWithFormat:@"%@", [qm5 stringValue]];
    NSString *em5string = [NSString stringWithFormat:@"%@", [em5 stringValue]];

    NSString *bm6string = [NSString stringWithFormat:@"%@", [bm6 stringValue]];
    NSString *qm6string = [NSString stringWithFormat:@"%@", [qm6 stringValue]];
    NSString *em6string = [NSString stringWithFormat:@"%@", [em6 stringValue]];

    NSString *bm7string = [NSString stringWithFormat:@"%@", [bm7 stringValue]];
    NSString *qm7string = [NSString stringWithFormat:@"%@", [qm7 stringValue]];
    NSString *em7string = [NSString stringWithFormat:@"%@", [em7 stringValue]];

    NSString *bm8string = [NSString stringWithFormat:@"%@", [bm8 stringValue]];
    NSString *qm8string = [NSString stringWithFormat:@"%@", [qm8 stringValue]];
    NSString *em8string = [NSString stringWithFormat:@"%@", [em8 stringValue]];

    NSString *bm9string = [NSString stringWithFormat:@"%@", [bm9 stringValue]];
    NSString *qm9string = [NSString stringWithFormat:@"%@", [qm9 stringValue]];
    NSString *em9string = [NSString stringWithFormat:@"%@", [em9 stringValue]];

    NSString *bm10string = [NSString stringWithFormat:@"%@", [bm10 stringValue]];
    NSString *qm10string = [NSString stringWithFormat:@"%@", [qm10 stringValue]];
    NSString *em10string = [NSString stringWithFormat:@"%@", [em10 stringValue]];

    NSString *bm11string = [NSString stringWithFormat:@"%@", [bm11 stringValue]];
    NSString *qm11string = [NSString stringWithFormat:@"%@", [qm11 stringValue]];
    NSString *em11string = [NSString stringWithFormat:@"%@", [em11 stringValue]];

    NSString *bm12string = [NSString stringWithFormat:@"%@", [bm12 stringValue]];
    NSString *qm12string = [NSString stringWithFormat:@"%@", [qm12 stringValue]];
    NSString *em12string = [NSString stringWithFormat:@"%@", [em12 stringValue]];

    NSString *bm13string = [NSString stringWithFormat:@"%@", [bm13 stringValue]];
    NSString *qm13string = [NSString stringWithFormat:@"%@", [qm13 stringValue]];
    NSString *em13string = [NSString stringWithFormat:@"%@", [em13 stringValue]];

    NSString *bm14string = [NSString stringWithFormat:@"%@", [bm14 stringValue]];
    NSString *qm14string = [NSString stringWithFormat:@"%@", [qm14 stringValue]];
    NSString *em14string = [NSString stringWithFormat:@"%@", [em14 stringValue]];

    NSString *bm15string = [NSString stringWithFormat:@"%@", [bm15 stringValue]];
    NSString *qm15string = [NSString stringWithFormat:@"%@", [qm15 stringValue]];
    NSString *em15string = [NSString stringWithFormat:@"%@", [em15 stringValue]];

    NSString *bm16string = [NSString stringWithFormat:@"%@", [bm16 stringValue]];
    NSString *qm16string = [NSString stringWithFormat:@"%@", [qm16 stringValue]];
    NSString *em16string = [NSString stringWithFormat:@"%@", [em16 stringValue]];

    NSString *bm17string = [NSString stringWithFormat:@"%@", [bm17 stringValue]];
    NSString *qm17string = [NSString stringWithFormat:@"%@", [qm17 stringValue]];
    NSString *em17string = [NSString stringWithFormat:@"%@", [em17 stringValue]];

    NSString *bm18string = [NSString stringWithFormat:@"%@", [bm18 stringValue]];
    NSString *qm18string = [NSString stringWithFormat:@"%@", [qm18 stringValue]];
    NSString *em18string = [NSString stringWithFormat:@"%@", [em18 stringValue]];

    NSString *bm19string = [NSString stringWithFormat:@"%@", [bm19 stringValue]];
    NSString *qm19string = [NSString stringWithFormat:@"%@", [qm19 stringValue]];
    NSString *em19string = [NSString stringWithFormat:@"%@", [em19 stringValue]];

    NSString *bm20string = [NSString stringWithFormat:@"%@", [bm20 stringValue]];
    NSString *qm20string = [NSString stringWithFormat:@"%@", [qm20 stringValue]];
    NSString *em20string = [NSString stringWithFormat:@"%@", [em20 stringValue]];


    //Conver the women's text inputs into NSStrings

    NSString *bw1string = [NSString stringWithFormat:@"%@", [bw1 stringValue]];
    NSString *qw1string = [NSString stringWithFormat:@"%@", [qw1 stringValue]];
    NSString *ew1string = [NSString stringWithFormat:@"%@", [ew1 stringValue]];

    NSString *bw2string = [NSString stringWithFormat:@"%@", [bw2 stringValue]];
    NSString *qw2string = [NSString stringWithFormat:@"%@", [qw2 stringValue]];
    NSString *ew2string = [NSString stringWithFormat:@"%@", [ew2 stringValue]];

    NSString *bw3string = [NSString stringWithFormat:@"%@", [bw3 stringValue]];
    NSString *qw3string = [NSString stringWithFormat:@"%@", [qw3 stringValue]];
    NSString *ew3string = [NSString stringWithFormat:@"%@", [ew3 stringValue]];

    NSString *bw4string = [NSString stringWithFormat:@"%@", [bw4 stringValue]];
    NSString *qw4string = [NSString stringWithFormat:@"%@", [qw4 stringValue]];
    NSString *ew4string = [NSString stringWithFormat:@"%@", [ew4 stringValue]];

    NSString *bw5string = [NSString stringWithFormat:@"%@", [bw5 stringValue]];
    NSString *qw5string = [NSString stringWithFormat:@"%@", [qw5 stringValue]];
    NSString *ew5string = [NSString stringWithFormat:@"%@", [ew5 stringValue]];

    NSString *bw6string = [NSString stringWithFormat:@"%@", [bw6 stringValue]];
    NSString *qw6string = [NSString stringWithFormat:@"%@", [qw6 stringValue]];
    NSString *ew6string = [NSString stringWithFormat:@"%@", [ew6 stringValue]];

    NSString *bw7string = [NSString stringWithFormat:@"%@", [bw7 stringValue]];
    NSString *qw7string = [NSString stringWithFormat:@"%@", [qw7 stringValue]];
    NSString *ew7string = [NSString stringWithFormat:@"%@", [ew7 stringValue]];

    NSString *bw8string = [NSString stringWithFormat:@"%@", [bw8 stringValue]];
    NSString *qw8string = [NSString stringWithFormat:@"%@", [qw8 stringValue]];
    NSString *ew8string = [NSString stringWithFormat:@"%@", [ew8 stringValue]];

    NSString *bw9string = [NSString stringWithFormat:@"%@", [bw9 stringValue]];
    NSString *qw9string = [NSString stringWithFormat:@"%@", [qw9 stringValue]];
    NSString *ew9string = [NSString stringWithFormat:@"%@", [ew9 stringValue]];

    NSString *bw10string = [NSString stringWithFormat:@"%@", [bw10 stringValue]];
    NSString *qw10string = [NSString stringWithFormat:@"%@", [qw10 stringValue]];
    NSString *ew10string = [NSString stringWithFormat:@"%@", [ew10 stringValue]];

    NSString *bw11string = [NSString stringWithFormat:@"%@", [bw11 stringValue]];
    NSString *qw11string = [NSString stringWithFormat:@"%@", [qw11 stringValue]];
    NSString *ew11string = [NSString stringWithFormat:@"%@", [ew11 stringValue]];

    NSString *bw12string = [NSString stringWithFormat:@"%@", [bw12 stringValue]];
    NSString *qw12string = [NSString stringWithFormat:@"%@", [qw12 stringValue]];
    NSString *ew12string = [NSString stringWithFormat:@"%@", [ew12 stringValue]];

    NSString *bw13string = [NSString stringWithFormat:@"%@", [bw13 stringValue]];
    NSString *qw13string = [NSString stringWithFormat:@"%@", [qw13 stringValue]];
    NSString *ew13string = [NSString stringWithFormat:@"%@", [ew13 stringValue]];

    NSString *bw14string = [NSString stringWithFormat:@"%@", [bw14 stringValue]];
    NSString *qw14string = [NSString stringWithFormat:@"%@", [qw14 stringValue]];
    NSString *ew14string = [NSString stringWithFormat:@"%@", [ew14 stringValue]];
    NSString *bw15string = [NSString stringWithFormat:@"%@", [bw15 stringValue]];
    NSString *qw15string = [NSString stringWithFormat:@"%@", [qw15 stringValue]];
    NSString *ew15string = [NSString stringWithFormat:@"%@", [ew15 stringValue]];

    NSString *bw16string = [NSString stringWithFormat:@"%@", [bw16 stringValue]];
    NSString *qw16string = [NSString stringWithFormat:@"%@", [qw16 stringValue]];
    NSString *ew16string = [NSString stringWithFormat:@"%@", [ew16 stringValue]];

    NSString *bw17string = [NSString stringWithFormat:@"%@", [bw17 stringValue]];
    NSString *qw17string = [NSString stringWithFormat:@"%@", [qw17 stringValue]];
    NSString *ew17string = [NSString stringWithFormat:@"%@", [ew17 stringValue]];

    NSString *bw18string = [NSString stringWithFormat:@"%@", [bw18 stringValue]];
    NSString *qw18string = [NSString stringWithFormat:@"%@", [qw18 stringValue]];
    NSString *ew18string = [NSString stringWithFormat:@"%@", [ew18 stringValue]];

    NSString *bw19string = [NSString stringWithFormat:@"%@", [bw19 stringValue]];
    NSString *qw19string = [NSString stringWithFormat:@"%@", [qw19 stringValue]];
    NSString *ew19string = [NSString stringWithFormat:@"%@", [ew19 stringValue]];

    NSString *bw20string = [NSString stringWithFormat:@"%@", [bw20 stringValue]];
    NSString *qw20string = [NSString stringWithFormat:@"%@", [qw20 stringValue]];
    NSString *ew20string = [NSString stringWithFormat:@"%@", [ew20 stringValue]];


    //Convert the men's strings into floats

    float bm1float = [bm1string floatValue];
    float qm1float = [qm1string floatValue];
    float em1float = [em1string floatValue];

    float bm2float = [bm2string floatValue];
    float qm2float = [qm2string floatValue];
    float em2float = [em2string floatValue];

    float bm3float = [bm3string floatValue];
    float qm3float = [qm3string floatValue];
    float em3float = [em3string floatValue];

    float bm4float = [bm4string floatValue];
    float qm4float = [qm4string floatValue];
    float em4float = [em4string floatValue];

    float bm5float = [bm5string floatValue];
    float qm5float = [qm5string floatValue];
    float em5float = [em5string floatValue];

    float bm6float = [bm6string floatValue];
    float qm6float = [qm6string floatValue];
    float em6float = [em6string floatValue];

    float bm7float = [bm7string floatValue];
    float qm7float = [qm7string floatValue];
    float em7float = [em7string floatValue];

    float bm8float = [bm8string floatValue];
    float qm8float = [qm8string floatValue];
    float em8float = [em8string floatValue];

    float bm9float = [bm9string floatValue];
    float qm9float = [qm9string floatValue];
    float em9float = [em9string floatValue];

    float bm10float = [bm10string floatValue];
    float qm10float = [qm10string floatValue];
    float em10float = [em10string floatValue];

    float bm11float = [bm11string floatValue];
    float qm11float = [qm11string floatValue];
    float em11float = [em11string floatValue];

    float bm12float = [bm12string floatValue];
    float qm12float = [qm12string floatValue];
    float em12float = [em12string floatValue];

    float bm13float = [bm13string floatValue];
    float qm13float = [qm13string floatValue];
    float em13float = [em13string floatValue];

    float bm14float = [bm14string floatValue];
    float qm14float = [qm14string floatValue];
    float em14float = [em14string floatValue];

    float bm15float = [bm15string floatValue];
    float qm15float = [qm15string floatValue];
    float em15float = [em15string floatValue];

    float bm16float = [bm16string floatValue];
    float qm16float = [qm16string floatValue];
    float em16float = [em16string floatValue];

    float bm17float = [bm17string floatValue];
    float qm17float = [qm17string floatValue];
    float em17float = [em17string floatValue];

    float bm18float = [bm18string floatValue];
    float qm18float = [qm18string floatValue];
    float em18float = [em18string floatValue];

    float bm19float = [bm19string floatValue];
    float qm19float = [qm19string floatValue];
    float em19float = [em19string floatValue];

    float bm20float = [bm20string floatValue];
    float qm20float = [qm20string floatValue];
    float em20float = [em20string floatValue];
```

```
//Convert the women's strings into floats


float bw1float = [bw1string floatValue];
float qw1float = [qw1string floatValue];
float ew1float = [ew1string floatValue];

float bw2float = [bw2string floatValue];
float qw2float = [qw2string floatValue];
float ew2float = [ew2string floatValue];

float bw3float = [bw3string floatValue];
float qw3float = [qw3string floatValue];
float ew3float = [ew3string floatValue];

float bw4float = [bw4string floatValue];
float qw4float = [qw4string floatValue];
float ew4float = [ew4string floatValue];

float bw5float = [bw5string floatValue];
float qw5float = [qw5string floatValue];
float ew5float = [ew5string floatValue];

float bw6float = [bw6string floatValue];
float qw6float = [qw6string floatValue];
float ew6float = [ew6string floatValue];

float bw7float = [bw7string floatValue];
float qw7float = [qw7string floatValue];
float ew7float = [ew7string floatValue];

float bw8float = [bw8string floatValue];
float qw8float = [qw8string floatValue];
float ew8float = [ew8string floatValue];

float bw9float = [bw9string floatValue];
float qw9float = [qw9string floatValue];
float ew9float = [ew9string floatValue];

float bw10float = [bw10string floatValue];
float qw10float = [qw10string floatValue];
float ew10float = [ew10string floatValue];

float bw11float = [bw11string floatValue];
float qw11float = [qw11string floatValue];
float ew11float = [ew11string floatValue];

float bw12float = [bw12string floatValue];
float qw12float = [qw12string floatValue];
float ew12float = [ew12string floatValue];

float bw13float = [bw13string floatValue];
float qw13float = [qw13string floatValue];
float ew13float = [ew13string floatValue];

float bw14float = [bw14string floatValue];
float qw14float = [qw14string floatValue];
float ew14float = [ew14string floatValue];

float bw15float = [bw15string floatValue];
float qw15float = [qw15string floatValue];
float ew15float = [ew15string floatValue];

float bw16float = [bw16string floatValue];
float qw16float = [qw16string floatValue];
float ew16float = [ew16string floatValue];

float bw17float = [bw17string floatValue];
float qw17float = [qw17string floatValue];
float ew17float = [ew17string floatValue];

float bw18float = [bw18string floatValue];
float qw18float = [qw18string floatValue];
float ew18float = [ew18string floatValue];

float bw19float = [bw19string floatValue];
float qw19float = [qw19string floatValue];
float ew19float = [ew19string floatValue];

float bw20float = [bw20string floatValue];
float qw20float = [qw20string floatValue];
float ew20float = [ew20string floatValue];



//Declare the variables we will be using in our calculations

float am1;
float am2;
float am3;
float am4;
float am5;
float am6;
float am7;
float am8;
float am9;
float am10;
float am11;
float am12;
float am13;
float am14;
float am15;
float am16;
float am17;
float am18;
float am19;
float am20;

float aw1;
float aw2;
float aw3;
float aw4;
float aw5;
float aw6;
float aw7;
float aw8;
float aw9;
float aw10;
float aw11;
float aw12;
float aw13;
float aw14;
float aw15;
float aw16;
float aw17;
float aw18;
float aw19;
float aw20;

float mm1;
float mm2;
float mm3;
float mm4;
float mm5;
float mm6;
float mm7;
float mm8;
float mm9;
float mm10;
float mm11;
float mm12;
float mm13;
float mm14;
float mm15;
float mm16;
float mm17;
float mm18;
float mm19;
float mm20;

float mw1;
float mw2;
float mw3;
float mw4;
float mw5;
float mw6;
float mw7;
float mw8;
float mw9;
float mw10;
float mw11;
float mw12;
float mw13;
float mw14;
float mw15;
float mw16;
float mw17;
float mw18;
float mw19;
float mw20;

float gammam1;
float gammam2;
float gammam3;
float gammam4;
float gammam5;
float gammam6;
float gammam7;
float gammam8;
float gammam9;
float gammam10;
float gammam11;
float gammam12;
float gammam13;
float gammam14;
float gammam15;
float gammam16;
float gammam17;
float gammam18;
float gammam19;
float gammam20;

float gammaw1;
float gammaw2;
float gammaw3;
float gammaw4;
float gammaw5;
float gammaw6;
float gammaw7;
float gammaw8;
float gammaw9;
float gammaw10;
float gammaw11;
float gammaw12;
float gammaw13;
float gammaw14;
float gammaw15;
float gammaw16;
float gammaw17;
float gammaw18;
float gammaw19;
float gammaw20;

float riskwithinsurancem1;
float riskwithinsurancem2;
float riskwithinsurancem3;
float riskwithinsurancem4;
float riskwithinsurancem5;
float riskwithinsurancem6;
float riskwithinsurancem7;
float riskwithinsurancem8;
float riskwithinsurancem9;
float riskwithinsurancem10;
float riskwithinsurancem11;
float riskwithinsurancem12;
float riskwithinsurancem13;
float riskwithinsurancem14;
float riskwithinsurancem15;
float riskwithinsurancem16;
float riskwithinsurancem17;
float riskwithinsurancem18;
float riskwithinsurancem19;
float riskwithinsurancem20;

float riskwithinsurancew1;
float riskwithinsurancew2;
float riskwithinsurancew3;
float riskwithinsurancew4;
float riskwithinsurancew5;
float riskwithinsurancew6;
float riskwithinsurancew7;
float riskwithinsurancew8;
float riskwithinsurancew9;
float riskwithinsurancew10;
float riskwithinsurancew11;
float riskwithinsurancew12;
float riskwithinsurancew13;
float riskwithinsurancew14;
float riskwithinsurancew15;
float riskwithinsurancew16;
float riskwithinsurancew17;
float riskwithinsurancew18;
float riskwithinsurancew19;
float riskwithinsurancew20;

float riskwithoutinsurancem1;
float riskwithoutinsurancem2;
float riskwithoutinsurancem3;
float riskwithoutinsurancem4;
float riskwithoutinsurancem5;
float riskwithoutinsurancem6;
float riskwithoutinsurancem7;
float riskwithoutinsurancem8;
float riskwithoutinsurancem9;
float riskwithoutinsurancem10;
float riskwithoutinsurancem11;
float riskwithoutinsurancem12;
float riskwithoutinsurancem13;
float riskwithoutinsurancem14;
float riskwithoutinsurancem15;
float riskwithoutinsurancem16;
float riskwithoutinsurancem17;
float riskwithoutinsurancem18;
float riskwithoutinsurancem19;
float riskwithoutinsurancem20;

float riskwithoutinsurancew1;
float riskwithoutinsurancew2;
float riskwithoutinsurancew3;
float riskwithoutinsurancew4;
float riskwithoutinsurancew5;
float riskwithoutinsurancew6;
float riskwithoutinsurancew7;
float riskwithoutinsurancew8;
float riskwithoutinsurancew9;
float riskwithoutinsurancew10;
float riskwithoutinsurancew11;
float riskwithoutinsurancew12;
float riskwithoutinsurancew13;
float riskwithoutinsurancew14;
float riskwithoutinsurancew15;
float riskwithoutinsurancew16;
float riskwithoutinsurancew17;
float riskwithoutinsurancew18;
float riskwithoutinsurancew19;
float riskwithoutinsurancew20;

float riskdifferencem1;
float riskdifferencem2;
float riskdifferencem3;
float riskdifferencem4;
float riskdifferencem5;
float riskdifferencem6;
float riskdifferencem7;
float riskdifferencem8;
float riskdifferencem9;
float riskdifferencem10;
float riskdifferencem11;
float riskdifferencem12;
float riskdifferencem13;
float riskdifferencem14;
float riskdifferencem15;
float riskdifferencem16;
float riskdifferencem17;
float riskdifferencem18;
float riskdifferencem19;
float riskdifferencem20;

float riskdifferencew1;
float riskdifferencew2;
float riskdifferencew3;
float riskdifferencew4;
```

```
    float riskdifferencew5;
    float riskdifferencew6;
    float riskdifferencew7;
    float riskdifferencew8;
    float riskdifferencew9;
    float riskdifferencew10;
    float riskdifferencew11;
    float riskdifferencew12;
    float riskdifferencew13;
    float riskdifferencew14;
    float riskdifferencew15;
    float riskdifferencew16;
    float riskdifferencew17;
    float riskdifferencew18;
    float riskdifferencew19;
    float riskdifferencew20;


    //Compute risk for all men's events

    am1 = qm1float*bm1float; //average cost of the bonus

    mm1 = em1float - am1; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammam1 = mm1 * (sfloat +  (am1))/ (am1*bm1float);

    riskwithoutinsurancem1 = am1*(1+gammam1* (bm1float / betafloat));
    riskwithinsurancem1 = am1*(1+mm1/am1);


    am2 = qm2float*bm2float; //average cost of the bonus

    mm2 = em2float - am2; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammam2 = mm2 * (sfloat +  (am2))/ (am2*bm2float);

    riskwithoutinsurancem2 = am2*(1+gammam2* (bm2float / betafloat));
    riskwithinsurancem2 = am2*(1+mm2/am2);


    am3 = qm3float*bm3float; //average cost of the bonus

    mm3 = em3float - am3; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammam3 = mm3 * (sfloat +  (am3))/ (am3*bm3float);

    riskwithoutinsurancem3 = am3*(1+gammam3* (bm3float / betafloat));
    riskwithinsurancem3 = am3*(1+mm3/am3);


    am4 = qm4float*bm4float; //average cost of the bonus
    mm4 = em4float - am4; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammam4 = mm4 * (sfloat +  (am4))/ (am4*bm4float);

    riskwithoutinsurancem4 = am4*(1+gammam4* (bm4float / betafloat));
    riskwithinsurancem4 = am4*(1+mm4/am4);


    am5 = qm5float*bm5float; //average cost of the bonus

    mm5 = em5float - am5; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammam5 = mm5 * (sfloat +  (am5))/ (am5*bm5float);

    riskwithoutinsurancem5 = am5*(1+gammam5* (bm5float / betafloat));
    riskwithinsurancem5 = am5*(1+mm5/am5);


    am6 = qm6float*bm6float; //average cost of the bonus

    mm6 = em6float - am6; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammam6 = mm6 * (sfloat +  (am6))/ (am6*bm6float);

    riskwithoutinsurancem6 = am6*(1+gammam6* (bm6float / betafloat));
    riskwithinsurancem6 = am6*(1+mm6/am6);


    am7 = qm7float*bm7float; //average cost of the bonus

    mm7 = em7float - am7; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammam7 = mm7 * (sfloat +  (am7))/ (am7*bm7float);

    riskwithoutinsurancem7 = am7*(1+gammam7* (bm7float / betafloat));
    riskwithinsurancem7 = am7*(1+mm7/am7);


    am8 = qm8float*bm8float; //average cost of the bonus

    mm8 = em8float - am8; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammam8 = mm8 * (sfloat +  (am8))/ (am8*bm8float);

    riskwithoutinsurancem8 = am8*(1+gammam8* (bm8float / betafloat));
    riskwithinsurancem8 = am8*(1+mm8/am8);
    am9 = qm9float*bm9float; //average cost of the bonus

    mm9 = em9float - am9; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammam9 = mm9 * (sfloat +  (am9))/ (am9*bm9float);

    riskwithoutinsurancem9 = am9*(1+gammam9* (bm9float / betafloat));
    riskwithinsurancem9 = am9*(1+mm9/am9);


    am10 = qm10float*bm10float; //average cost of the bonus

    mm10 = em10float - am10; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammam10 = mm10 * (sfloat +  (am10))/ (am10*bm10float);

    riskwithoutinsurancem10 = am10*(1+gammam10* (bm10float / betafloat));
    riskwithinsurancem10 = am10*(1+mm10/am10);


    am11 = qm11float*bm11float; //average cost of the bonus

    mm11 = em11float - am11; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammam11 = mm11 * (sfloat +  (am11))/ (am11*bm11float);

    riskwithoutinsurancem11 = am11*(1+gammam11* (bm11float / betafloat));
    riskwithinsurancem11 = am11*(1+mm11/am11);


    am12 = qm12float*bm12float; //average cost of the bonus

    mm12 = em12float - am12; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammam12 = mm12 * (sfloat +  (am12))/ (am12*bm12float);

    riskwithoutinsurancem12 = am12*(1+gammam12* (bm12float / betafloat));
    riskwithinsurancem12 = am12*(1+mm12/am12);


    am13 = qm13float*bm13float; //average cost of the bonus

    mm13 = em13float - am13; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammam13 = mm13 * (sfloat +  (am13))/ (am13*bm13float);

    riskwithoutinsurancem13 = am13*(1+gammam13* (bm13float / betafloat));
    riskwithinsurancem13 = am13*(1+mm13/am13);
    am14 = qm14float*bm14float; //average cost of the bonus

    mm14 = em14float - am14; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammam14 = mm14 * (sfloat +  (am14))/ (am14*bm14float);

    riskwithoutinsurancem14 = am14*(1+gammam14* (bm14float / betafloat));
    riskwithinsurancem14 = am14*(1+mm14/am14);


    am15 = qm15float*bm15float; //average cost of the bonus

    mm15 = em15float - am15; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammam15 = mm15 * (sfloat +  (am15))/ (am15*bm15float);

    riskwithoutinsurancem15 = am15*(1+gammam15* (bm15float / betafloat));
    riskwithinsurancem15 = am15*(1+mm15/am15);


    am16 = qm16float*bm16float; //average cost of the bonus

    mm16 = em16float - am16; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammam16 = mm16 * (sfloat +  (am16))/ (am16*bm16float);

    riskwithoutinsurancem16 = am16*(1+gammam16* (bm16float / betafloat));
    riskwithinsurancem16 = am16*(1+mm16/am16);


    am17 = qm17float*bm17float; //average cost of the bonus

    mm17 = em17float - am17; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammam17 = mm17 * (sfloat +  (am17))/ (am17*bm17float);

    riskwithoutinsurancem17 = am17*(1+gammam17* (bm17float / betafloat));
    riskwithinsurancem17 = am17*(1+mm17/am17);


    am18 = qm18float*bm18float; //average cost of the bonus

    mm18 = em18float - am18; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammam18 = mm18 * (sfloat +  (am18))/ (am18*bm18float);

    riskwithoutinsurancem18 = am18*(1+gammam18* (bm18float / betafloat));
    riskwithinsurancem18 = am18*(1+mm18/am18);
    am19 = qm19float*bm19float; //average cost of the bonus

    mm19 = em19float - am19; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammam19 = mm19 * (sfloat +  (am19))/ (am19*bm19float);

    riskwithoutinsurancem19 = am19*(1+gammam19* (bm19float / betafloat));
    riskwithinsurancem19 = am19*(1+mm19/am19);


    am20 = qm20float*bm20float; //average cost of the bonus

    mm20 = em20float - am20; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammam20 = mm20 * (sfloat +  (am20))/ (am20*bm20float);

    riskwithoutinsurancem20 = am20*(1+gammam20* (bm20float / betafloat));
    riskwithinsurancem20 = am20*(1+mm20/am20);





    //Compute risk for all women's events



    aw1 = qw1float*bw1float; //average cost of the bonus

    mw1 = ew1float - aw1; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammaw1 = mw1 * (sfloat +  (aw1))/ (aw1*bw1float);

    riskwithoutinsurancew1 = aw1*(1+gammaw1* (bw1float / betafloat));
    riskwithinsurancew1 = aw1*(1+mw1/aw1);


    aw2 = qw2float*bw2float; //average cost of the bonus

    mw2 = ew2float - aw2; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammaw2 = mw2 * (sfloat +  (aw2))/ (aw2*bw2float);

    riskwithoutinsurancew2 = aw2*(1+gammaw2* (bw2float / betafloat));
    riskwithinsurancew2 = aw2*(1+mw2/aw2);
    aw3 = qw3float*bw3float; //average cost of the bonus

    mw3 = ew3float - aw3; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammaw3 = mw3 * (sfloat +  (aw3))/ (aw3*bw3float);

    riskwithoutinsurancew3 = aw3*(1+gammaw3* (bw3float / betafloat));
    riskwithinsurancew3 = aw3*(1+mw3/aw3);


    aw4 = qw4float*bw4float; //average cost of the bonus

    mw4 = ew4float - aw4; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammaw4 = mw4 * (sfloat +  (aw4))/ (aw4*bw4float);

    riskwithoutinsurancew4 = aw4*(1+gammaw4* (bw4float / betafloat));
    riskwithinsurancew4 = aw4*(1+mw4/aw4);


    aw5 = qw5float*bw5float; //average cost of the bonus

    mw5 = ew5float - aw5; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammaw5 = mw5 * (sfloat +  (aw5))/ (aw5*bw5float);

    riskwithoutinsurancew5 = aw5*(1+gammaw5* (bw5float / betafloat));
    riskwithinsurancew5 = aw5*(1+mw5/aw5);


    aw6 = qw6float*bw6float; //average cost of the bonus

    mw6 = ew6float - aw6; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammaw6 = mw6 * (sfloat +  (aw6))/ (aw6*bw6float);

    riskwithoutinsurancew6 = aw6*(1+gammaw6* (bw6float / betafloat));
    riskwithinsurancew6 = aw6*(1+mw6/aw6);


    aw7 = qw7float*bw7float; //average cost of the bonus

    mw7 = ew7float - aw7; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammaw7 = mw7 * (sfloat +  (aw7))/ (aw7*bw7float);

    riskwithoutinsurancew7 = aw7*(1+gammaw7* (bw7float / betafloat));
    riskwithinsurancew7 = aw7*(1+mw7/aw7);
    aw8 = qw8float*bw8float; //average cost of the bonus

    mw8 = ew8float - aw8; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammaw8 = mw8 * (sfloat +  (aw8))/ (aw8*bw8float);

    riskwithoutinsurancew8 = aw8*(1+gammaw8* (bw8float / betafloat));
    riskwithinsurancew8 = aw8*(1+mw8/aw8);


    aw9 = qw9float*bw9float; //average cost of the bonus

    mw9 = ew9float - aw9; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammaw9 = mw9 * (sfloat +  (aw9))/ (aw9*bw9float);

    riskwithoutinsurancew9 = aw9*(1+gammaw9* (bw9float / betafloat));
    riskwithinsurancew9 = aw9*(1+mw9/aw9);


    aw10 = qw10float*bw10float; //average cost of the bonus

    mw10 = ew10float - aw10; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammaw10 = mw10 * (sfloat +  (aw10))/ (aw10*bw10float);

    riskwithoutinsurancew10 = aw10*(1+gammaw10* (bw10float / betafloat));
    riskwithinsurancew10 = aw10*(1+mw10/aw10);


    aw11 = qw11float*bw11float; //average cost of the bonus

    mw11 = ew11float - aw11; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammaw11 = mw11 * (sfloat +  (aw11))/ (aw11*bw11float);

    riskwithoutinsurancew11 = aw11*(1+gammaw11* (bw11float / betafloat));
    riskwithinsurancew11 = aw11*(1+mw11/aw11);


    aw12 = qw12float*bw12float; //average cost of the bonus

    mw12 = ew12float - aw12; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammaw12 = mw12 * (sfloat +  (aw12))/ (aw12*bw12float);

    riskwithoutinsurancew12 = aw12*(1+gammaw12* (bw12float / betafloat));
    riskwithinsurancew12 = aw12*(1+mw12/aw12);
    aw13 = qw13float*bw13float; //average cost of the bonus

    mw13 = ew13float - aw13; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammaw13 = mw13 * (sfloat +  (aw13))/ (aw13*bw13float);

    riskwithoutinsurancew13 = aw13*(1+gammaw13* (bw13float / betafloat));
    riskwithinsurancew13 = aw13*(1+mw13/aw13);


    aw14 = qw14float*bw14float; //average cost of the bonus

    mw14 = ew14float - aw14; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammaw14 = mw14 * (sfloat +  (aw14))/ (aw14*bw14float);

    riskwithoutinsurancew14 = aw14*(1+gammaw14* (bw14float / betafloat));
    riskwithinsurancew14 = aw14*(1+mw14/aw14);


    aw15 = qw15float*bw15float; //average cost of the bonus

    mw15 = ew15float - aw15; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammaw15 = mw15 * (sfloat +  (aw15))/ (aw15*bw15float);

    riskwithoutinsurancew15 = aw15*(1+gammaw15* (bw15float / betafloat));
    riskwithinsurancew15 = aw15*(1+mw15/aw15);


    aw16 = qw16float*bw16float; //average cost of the bonus

    mw16 = ew16float - aw16; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammaw16 = mw16 * (sfloat +  (aw16))/ (aw16*bw16float);

    riskwithoutinsurancew16 = aw16*(1+gammaw16* (bw16float / betafloat));
    riskwithinsurancew16 = aw16*(1+mw16/aw16);


    aw17 = qw17float*bw17float; //average cost of the bonus

    mw17 = ew17float - aw17; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammaw17 = mw17 * (sfloat +  (aw17))/ (aw17*bw17float);

    riskwithoutinsurancew17 = aw17*(1+gammaw17* (bw17float / betafloat));
    riskwithinsurancew17 = aw17*(1+mw17/aw17);


    aw18 = qw18float*bw18float; //average cost of the bonus

    mw18 = ew18float - aw18; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammaw18 = mw18 * (sfloat +  (aw18))/ (aw18*bw18float);

    riskwithoutinsurancew18 = aw18*(1+gammaw18* (bw18float / betafloat));
    riskwithinsurancew18 = aw18*(1+mw18/aw18);


    aw19 = qw19float*bw19float; //average cost of the bonus

    mw19 = ew19float - aw19; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammaw19 = mw19 * (sfloat +  (aw19))/ (aw19*bw19float);

    riskwithoutinsurancew19 = aw19*(1+gammaw19* (bw19float / betafloat));
    riskwithinsurancew19 = aw19*(1+mw19/aw19);


    aw20 = qw20float*bw20float; //average cost of the bonus

    mw20 = ew20float - aw20; //cost of the premium minus the average cost of the bonus
equals excess cost added to the premium by the insurance company

    gammaw20 = mw20 * (sfloat +  (aw20))/ (aw20*bw20float);

    riskwithoutinsurancew20 = aw20*(1+gammaw20* (bw20float / betafloat));
    riskwithinsurancew20 = aw20*(1+mw20/aw20);


    //Calculate risk differences for men

    riskdifferencem1 =
(riskwithoutinsurancem1-riskwithinsurancem1)/riskwithoutinsurancem1*100;
    riskdifferencem2 =
(riskwithoutinsurancem2-riskwithinsurancem2)/riskwithoutinsurancem2*100;
    riskdifferencem3 =
(riskwithoutinsurancem3-riskwithinsurancem3)/riskwithoutinsurancem3*100;
    riskdifferencem4 =
(riskwithoutinsurancem4-riskwithinsurancem4)/riskwithoutinsurancem4*100;
    riskdifferencem5 =
(riskwithoutinsurancem5-riskwithinsurancem5)/riskwithoutinsurancem5*100;
    riskdifferencem6 =
(riskwithoutinsurancem6-riskwithinsurancem6)/riskwithoutinsurancem6*100;
    riskdifferencem7 =
(riskwithoutinsurancem7-riskwithinsurancem7)/riskwithoutinsurancem7*100;
    riskdifferencem8 =
(riskwithoutinsurancem8-riskwithinsurancem8)/riskwithoutinsurancem8*100;
```

```
    riskdifferencem9 =
(riskwithoutinsurancem9-riskwithinsurancem9)/riskwithoutinsurancem9*100;
    riskdifferencem10 =
(riskwithoutinsurancem10-riskwithinsurancem10)/riskwithoutinsurancem10*100;
    riskdifferencem11 =
(riskwithoutinsurancem11-riskwithinsurancem11)/riskwithoutinsurancem11*100;
    riskdifferencem12 =
(riskwithoutinsurancem12-riskwithinsurancem12)/riskwithoutinsurancem12*100;
    riskdifferencem13 =
(riskwithoutinsurancem13-riskwithinsurancem13)/riskwithoutinsurancem13*100;
    riskdifferencem14 =
(riskwithoutinsurancem14-riskwithinsurancem14)/riskwithoutinsurancem14*100;
    riskdifferencem15 =
(riskwithoutinsurancem15-riskwithinsurancem15)/riskwithoutinsurancem15*100;
    riskdifferencem16 =
(riskwithoutinsurancem16-riskwithinsurancem16)/riskwithoutinsurancem16*100;
    riskdifferencem17 =
(riskwithoutinsurancem17-riskwithinsurancem17)/riskwithoutinsurancem17*100;
    riskdifferencem18 =
(riskwithoutinsurancem18-riskwithinsurancem18)/riskwithoutinsurancem18*100;
    riskdifferencem19 =
(riskwithoutinsurancem19-riskwithinsurancem19)/riskwithoutinsurancem19*100;
    riskdifferencem20 =
(riskwithoutinsurancem20-riskwithinsurancem20)/riskwithoutinsurancem20*100;


    //Calculate risk differences for women

    riskdifferencew1 =
(riskwithoutinsurancew1-riskwithinsurancew1)/riskwithoutinsurancew1*100;
    riskdifferencew2 =
(riskwithoutinsurancew2-riskwithinsurancew2)/riskwithoutinsurancew2*100;
    riskdifferencew3 =
(riskwithoutinsurancew3-riskwithinsurancew3)/riskwithoutinsurancew3*100;
    riskdifferencew4 =
(riskwithoutinsurancew4-riskwithinsurancew4)/riskwithoutinsurancew4*100;
    riskdifferencew5 =
(riskwithoutinsurancew5-riskwithinsurancew5)/riskwithoutinsurancew5*100;
    riskdifferencew6 =
(riskwithoutinsurancew6-riskwithinsurancew6)/riskwithoutinsurancew6*100;
    riskdifferencew7 =
(riskwithoutinsurancew7-riskwithinsurancew7)/riskwithoutinsurancew7*100;
    riskdifferencew8 =
(riskwithoutinsurancew8-riskwithinsurancew8)/riskwithoutinsurancew8*100;
    riskdifferencew9 =
(riskwithoutinsurancew9-riskwithinsurancew9)/riskwithoutinsurancew9*100;
    riskdifferencew10 =
(riskwithoutinsurancew10-riskwithinsurancew10)/riskwithoutinsurancew10*100;
    riskdifferencew11 =
(riskwithoutinsurancew11-riskwithinsurancew11)/riskwithoutinsurancew11*100;
    riskdifferencew12 =
(riskwithoutinsurancew12-riskwithinsurancew12)/riskwithoutinsurancew12*100;
    riskdifferencew13 =
(riskwithoutinsurancew13-riskwithinsurancew13)/riskwithoutinsurancew13*100;
```

```
    riskdifferencew14 = 
(riskwithoutinsurancew14-riskwithinsurancew14)/riskwithoutinsurancew14*100;
    riskdifferencew15 = 
(riskwithoutinsurancew15-riskwithinsurancew15)/riskwithoutinsurancew15*100;
    riskdifferencew16 = 
(riskwithoutinsurancew16-riskwithinsurancew16)/riskwithoutinsurancew16*100;
    riskdifferencew17 = 
(riskwithoutinsurancew17-riskwithinsurancew17)/riskwithoutinsurancew17*100;
    riskdifferencew18 = 
(riskwithoutinsurancew18-riskwithinsurancew18)/riskwithoutinsurancew18*100;
    riskdifferencew19 = 
(riskwithoutinsurancew19-riskwithinsurancew19)/riskwithoutinsurancew19*100;
    riskdifferencew20 = 
(riskwithoutinsurancew20-riskwithinsurancew20)/riskwithoutinsurancew20*100;



    //Create an array of the relative risks.

    NSArray *unsortedRiskDifferences = [NSArray arrayWithObjects: [NSNumber 
numberWithFloat:riskdifferencem1],[NSNumber 
numberWithFloat:riskdifferencem2],[NSNumber 
numberWithFloat:riskdifferencem3],[NSNumber 
numberWithFloat:riskdifferencem4],[NSNumber 
numberWithFloat:riskdifferencem5],[NSNumber 
numberWithFloat:riskdifferencem6],[NSNumber 
numberWithFloat:riskdifferencem7],[NSNumber 
numberWithFloat:riskdifferencem8],[NSNumber 
numberWithFloat:riskdifferencem9],[NSNumber 
numberWithFloat:riskdifferencem10],[NSNumber 
numberWithFloat:riskdifferencem11],[NSNumber 
numberWithFloat:riskdifferencem12],[NSNumber 
numberWithFloat:riskdifferencem13],[NSNumber 
numberWithFloat:riskdifferencem14],[NSNumber 
numberWithFloat:riskdifferencem15],[NSNumber 
numberWithFloat:riskdifferencem16],[NSNumber 
numberWithFloat:riskdifferencem17],[NSNumber 
numberWithFloat:riskdifferencem18],[NSNumber 
numberWithFloat:riskdifferencem19],[NSNumber 
numberWithFloat:riskdifferencem20],[NSNumber 
numberWithFloat:riskdifferencew1],[NSNumber 
numberWithFloat:riskdifferencew2],[NSNumber 
numberWithFloat:riskdifferencew3],[NSNumber 
numberWithFloat:riskdifferencew4],[NSNumber 
numberWithFloat:riskdifferencew5],[NSNumber 
numberWithFloat:riskdifferencew6],[NSNumber 
numberWithFloat:riskdifferencew7],[NSNumber 
numberWithFloat:riskdifferencew8],[NSNumber 
numberWithFloat:riskdifferencew9],[NSNumber 
numberWithFloat:riskdifferencew10],[NSNumber 
numberWithFloat:riskdifferencew11],[NSNumber 
numberWithFloat:riskdifferencew12],[NSNumber 
numberWithFloat:riskdifferencew13],[NSNumber 
numberWithFloat:riskdifferencew14],[NSNumber 
numberWithFloat:riskdifferencew15],[NSNumber
numberWithFloat:riskdifferencew16],[NSNumber
numberWithFloat:riskdifferencew17],[NSNumber
numberWithFloat:riskdifferencew18],[NSNumber
numberWithFloat:riskdifferencew19],[NSNumber numberWithFloat:riskdifferencew20], nil];

    NSArray *sortedRiskDifferencesInverted = [NSArray arrayWithObjects: [NSNumber
numberWithFloat:riskdifferencem1],[NSNumber
numberWithFloat:riskdifferencem2],[NSNumber
numberWithFloat:riskdifferencem3],[NSNumber
numberWithFloat:riskdifferencem4],[NSNumber
numberWithFloat:riskdifferencem5],[NSNumber
numberWithFloat:riskdifferencem6],[NSNumber
numberWithFloat:riskdifferencem7],[NSNumber
numberWithFloat:riskdifferencem8],[NSNumber
numberWithFloat:riskdifferencem9],[NSNumber
numberWithFloat:riskdifferencem10],[NSNumber
numberWithFloat:riskdifferencem11],[NSNumber
numberWithFloat:riskdifferencem12],[NSNumber
numberWithFloat:riskdifferencem13],[NSNumber
numberWithFloat:riskdifferencem14],[NSNumber
numberWithFloat:riskdifferencem15],[NSNumber
numberWithFloat:riskdifferencem16],[NSNumber
numberWithFloat:riskdifferencem17],[NSNumber
numberWithFloat:riskdifferencem18],[NSNumber
numberWithFloat:riskdifferencem19],[NSNumber
numberWithFloat:riskdifferencem20],[NSNumber
numberWithFloat:riskdifferencew1],[NSNumber
numberWithFloat:riskdifferencew2],[NSNumber
numberWithFloat:riskdifferencew3],[NSNumber
numberWithFloat:riskdifferencew4],[NSNumber
numberWithFloat:riskdifferencew5],[NSNumber
numberWithFloat:riskdifferencew6],[NSNumber
numberWithFloat:riskdifferencew7],[NSNumber
numberWithFloat:riskdifferencew8],[NSNumber
numberWithFloat:riskdifferencew9],[NSNumber
numberWithFloat:riskdifferencew10],[NSNumber
numberWithFloat:riskdifferencew11],[NSNumber
numberWithFloat:riskdifferencew12],[NSNumber
numberWithFloat:riskdifferencew13],[NSNumber
numberWithFloat:riskdifferencew14],[NSNumber
numberWithFloat:riskdifferencew15],[NSNumber
numberWithFloat:riskdifferencew16],[NSNumber
numberWithFloat:riskdifferencew17],[NSNumber
numberWithFloat:riskdifferencew18],[NSNumber
numberWithFloat:riskdifferencew19],[NSNumber numberWithFloat:riskdifferencew20], nil];

    sortedRiskDifferencesInverted = [sortedRiskDifferencesInverted
sortedArrayUsingSelector:@selector(compare:)]; //Sort the "sorted" array so that it is
actually sorted. However, the @selector(compare:) function sorts in ascending order,
while we want the array sorted in descending order. We therefore reverse the sort
order below:
    NSArray *sortedRiskDifferences = [[sortedRiskDifferencesInverted
reverseObjectEnumerator]allObjects]; //Reverse the order of the objects in the array
so it is sorted from highest to lowest

    NSArray *unsortedInsuranceRisks = [NSArray arrayWithObjects: [NSNumber
numberWithFloat:riskwithinsurancem1],[NSNumber
numberWithFloat:riskwithinsurancem2],[NSNumber
numberWithFloat:riskwithinsurancem3],[NSNumber
numberWithFloat:riskwithinsurancem4],[NSNumber
numberWithFloat:riskwithinsurancem5],[NSNumber
numberWithFloat:riskwithinsurancem6],[NSNumber
numberWithFloat:riskwithinsurancem7],[NSNumber
numberWithFloat:riskwithinsurancem8],[NSNumber
numberWithFloat:riskwithinsurancem9],[NSNumber
numberWithFloat:riskwithinsurancem10],[NSNumber
numberWithFloat:riskwithinsurancem11],[NSNumber
numberWithFloat:riskwithinsurancem12],[NSNumber
numberWithFloat:riskwithinsurancem13],[NSNumber
numberWithFloat:riskwithinsurancem14],[NSNumber
numberWithFloat:riskwithinsurancem15],[NSNumber
numberWithFloat:riskwithinsurancem16],[NSNumber
numberWithFloat:riskwithinsurancem17],[NSNumber
numberWithFloat:riskwithinsurancem18],[NSNumber
numberWithFloat:riskwithinsurancem19],[NSNumber
numberWithFloat:riskwithinsurancem20],[NSNumber
numberWithFloat:riskwithinsurancew1],[NSNumber
numberWithFloat:riskwithinsurancew2],[NSNumber
numberWithFloat:riskwithinsurancew3],[NSNumber
numberWithFloat:riskwithinsurancew4],[NSNumber
numberWithFloat:riskwithinsurancew5],[NSNumber
numberWithFloat:riskwithinsurancew6],[NSNumber
numberWithFloat:riskwithinsurancew7],[NSNumber
numberWithFloat:riskwithinsurancew8],[NSNumber
numberWithFloat:riskwithinsurancew9],[NSNumber
numberWithFloat:riskwithinsurancew10],[NSNumber
numberWithFloat:riskwithinsurancew11],[NSNumber
numberWithFloat:riskwithinsurancew12],[NSNumber
numberWithFloat:riskwithinsurancew13],[NSNumber
numberWithFloat:riskwithinsurancew14],[NSNumber
numberWithFloat:riskwithinsurancew15],[NSNumber
numberWithFloat:riskwithinsurancew16],[NSNumber
numberWithFloat:riskwithinsurancew17],[NSNumber
numberWithFloat:riskwithinsurancew18],[NSNumber
numberWithFloat:riskwithinsurancew19],[NSNumber
numberWithFloat:riskwithinsurancew20],nil];

    NSMutableArray *sortedInsuranceRisks = [[NSMutableArray alloc]
initWithCapacity:0];

    NSMutableArray *finalDecisions = [[NSMutableArray alloc] initWithCapacity:0];

    for(int q=0; q<40; q++) {
        //Assume insurance should not be purchased for all 40 events unless otherwise
determined in the future (especially because some events simply may not be held, so
insurance should only be recommended if the event definitely will be held in the first
place and definitely makes sense economically to insure)

        [finalDecisions addObject:[NSNumber numberWithBool:NO]];
    }


    float currentSortedDifferenceValue;
    float currentUnsortedDifferenceValue;

    for(int i=0; i<=39; i++) { //Determine whether the risk with insurance is greater
than that without. After this for loop, the next for loop will determine, for those
events where the insurance risk is lower, whether it makes sense to buy the insurance
given budget constraints.

        currentSortedDifferenceValue = [sortedRiskDifferences[i] floatValue];

        if (currentSortedDifferenceValue > 0) { //It might make sense to purchase
insurance; this event remains viable


            for (int j=0; j<40; j++) {//We have sorted the events by risk difference,
essentially putting them in random order. We look for a match against the original
risk difference array (ordered by men's 1, men's 2, men's 3...women's 20 rather than
by risk value) to determine where the event we are examining originally stood (this is
the j-value).

                currentUnsortedDifferenceValue = [unsortedRiskDifferences[j]
floatValue];

                if (currentUnsortedDifferenceValue != currentSortedDifferenceValue) {

                    //The program did not find a match; try again

                }

                else {
                    [sortedInsuranceRisks addObject:unsortedInsuranceRisks[j]]; //We
found the j-value, which is the standing in the original array for the event we are
looking at from the sorted array. We use this j-value to search the original insurance
risks array for the relevant risk, and then we add this risk to the new array we are
building, sortedInsuranceRisks.
                    break;
                }

            }

        }



    }
    int arraySize = [sortedInsuranceRisks count]; //determine how many races are
currently viable for insurance (this returns a whole number, and we are assigning that
whole number to an integer, which causes some compilers to warn the programmer of a
loss of precision, but the loss of precision only applies in the other direction; that
is, putting an integer value into a whole number variable -- not putting a whole
number value into an integer variable).
    float disposableBudget;
    disposableBudget = betafloat - sfloat;

    float currentSortedRiskValue;
    float currentUnsortedRiskValue;

    for(int k=0; k<arraySize; k++) {

        if (disposableBudget > 0) { //if we start with extra disposable cash, check if
we can add another insurance policy

        disposableBudget = disposableBudget - [sortedInsuranceRisks[k] floatValue];

            if (disposableBudget > 0) { //if after adding the insurance policy, we
still have extra disposable cash, insure that race

                for (int n=0; n<40; n++) {

                    currentSortedRiskValue = [sortedInsuranceRisks[k] floatValue];
                    currentUnsortedRiskValue = [unsortedInsuranceRisks[n] floatValue];

                    if (currentUnsortedRiskValue != currentSortedRiskValue) {//We
again try and find the appropriate value in the unsorted array; see explanation
further above

                        //You did not find a match; try again

                    }

                    else {
                        //Buy insurance for this event
                        finalDecisions[n] = @YES;
                        break;
                    }

                }


            }


            else {

                //Although we started with disposable cash, adding the insurance
policy would give us negative capital levels. We will not insure that event because we
cannot afford to, but we do get to add the cost of that insurance premium back to the
disposable cash value, since we are not purchasing that insurance
```

```
                disposableBudget = disposableBudget + [sortedInsuranceRisks[k]
floatValue];


            }




        }




    }


    //The following code assembles and formats the results

    NSMutableString *results = [[NSMutableString alloc] initWithString:@""];

    int z = 0;

    while (z<20) {
        int eventNumber = z+1;

        float zCheckOne = (z+1)%2; //Checks whether there is a remainder or not when
the z-value is divided by 2 (the "%" operator gives the remainder of the first
expression divided by the second expression). This has the effect of checking whether
z is even or odd. We want two results per line, so we type a return (/n) after results
with even z, while just adding a few spaces after results with odd z. This gives us
two results per line.


        if (zCheckOne == 0) {

        if([finalDecisions[z] boolValue] == NO) {

            NSString *temporaryString = [NSString stringWithFormat:@"Men's Event %i:
DO NOT buy insurance.\n\n",eventNumber];
            [results appendString:temporaryString];
        }

        if([finalDecisions[z] boolValue] == YES) {

            NSString *temporaryString = [NSString stringWithFormat:@"Men's Event %i:
BUY insurance.\n\n",eventNumber];
            [results appendString:temporaryString];

        }

        }

        else {
            if([finalDecisions[z] boolValue] == NO) {

                NSString *temporaryString = [NSString stringWithFormat:@"Men's Event
%i: DO NOT buy insurance.    ",eventNumber];
                [results appendString:temporaryString];
            }

            if([finalDecisions[z] boolValue] == YES) {

                NSString *temporaryString = [NSString stringWithFormat:@"Men's Event
%i: BUY insurance.    ",eventNumber];
                [results appendString:temporaryString];

            }
        }

        z=z+1;
    }

    z=1;

    while (z<21) {

        float zCheckOne = (z+1)%2;

        if(zCheckOne == 0) {


            if([finalDecisions[z+19] boolValue] == NO) { //Add 19 to all the indices
of the array because all women's values are 20 spaces further along in the array than
the corresponding men's values. For example, men's event 1 has index [0], while
women's event 1 has index [20]

                NSString *temporaryString = [NSString stringWithFormat:@"Women's Event
%i: DO NOT buy insurance.    ",z];
                [results appendString:temporaryString];
            }

            if([finalDecisions[z+19] boolValue] == YES) {
                NSString *temporaryString = [NSString stringWithFormat:@"Women's Event
%i: BUY insurance.    ",z];
                [results appendString:temporaryString];

            }


        }

        else {

        if([finalDecisions[z+19] boolValue] == NO) {
            NSString *temporaryString = [NSString stringWithFormat:@"Women's Event %i:
DO NOT buy insurance.\n\n",z];
            [results appendString:temporaryString];
        }

        if([finalDecisions[z+19] boolValue] == YES) {
            NSString *temporaryString = [NSString stringWithFormat:@"Women's Event %i:
BUY insurance.\n\n",z];
            [results appendString:temporaryString];

        }

        }

        z = z+1;

    }

    //Display the results

    NSAlert *alert = [NSAlert alertWithMessageText: @"Results"
                                     defaultButton:@"OK"
                                   alternateButton:@"Cancel"
                                       otherButton:nil
                         informativeTextWithFormat:@""];

    NSTextField *input = [[NSTextField alloc] initWithFrame:NSMakeRect(0, 0, 550,
650)];
    [input setStringValue:results];
    [alert setAccessoryView:input];
    [alert runModal];




}

- (IBAction)reset:(id) sender {

    [s setStringValue:@""];
    [s display];

    [beta setStringValue:@""];
    [beta display];

    [qm1 setStringValue:@""];
    [qm1 display];

    [qm2 setStringValue:@""];
    [qm2 display];

    [qm3 setStringValue:@""];
    [qm3 display];
```

```
[qm4 setStringValue:@""];
[qm4 display];

[qm5 setStringValue:@""];
[qm5 display];

[qm6 setStringValue:@""];
[qm6 display];

[qm7 setStringValue:@""];
[qm7 display];

[qm8 setStringValue:@""];
[qm8 display];

[qm9 setStringValue:@""];
[qm9 display];

[qm10 setStringValue:@""];
[qm10 display];

[qm11 setStringValue:@""];
[qm11 display];

[qm12 setStringValue:@""];
[qm12 display];

[qm13 setStringValue:@""];
[qm13 display];

[qm14 setStringValue:@""];
[qm14 display];

[qm15 setStringValue:@""];
[qm15 display];

[qm16 setStringValue:@""];
[qm16 display];

[qm17 setStringValue:@""];
[qm17 display];

[qm18 setStringValue:@""];
[qm18 display];

[qm19 setStringValue:@""];
[qm19 display];

[qm20 setStringValue:@""];
[qm20 display];


[bm1 setStringValue:@""];
[bm1 display];
```

```
[bm2 setStringValue:@""];
[bm2 display];

[bm3 setStringValue:@""];
[bm3 display];

[bm4 setStringValue:@""];
[bm4 display];

[bm5 setStringValue:@""];
[bm5 display];

[bm6 setStringValue:@""];
[bm6 display];

[bm7 setStringValue:@""];
[bm7 display];

[bm8 setStringValue:@""];
[bm8 display];

[bm9 setStringValue:@""];
[bm9 display];

[bm10 setStringValue:@""];
[bm10 display];

[bm11 setStringValue:@""];
[bm11 display];

[bm12 setStringValue:@""];
[bm12 display];

[bm13 setStringValue:@""];
[bm13 display];

[bm14 setStringValue:@""];
[bm14 display];

[bm15 setStringValue:@""];
[bm15 display];

[bm16 setStringValue:@""];
[bm16 display];

[bm17 setStringValue:@""];
[bm17 display];

[bm18 setStringValue:@""];
[bm18 display];

[bm19 setStringValue:@""];
[bm19 display];
```

```
[bm20 setStringValue:@""];
[bm20 display];




[em1 setStringValue:@""];
[em1 display];

[em2 setStringValue:@""];
[em2 display];

[em3 setStringValue:@""];
[em3 display];

[em4 setStringValue:@""];
[em4 display];

[em5 setStringValue:@""];
[em5 display];

[em6 setStringValue:@""];
[em6 display];

[em7 setStringValue:@""];
[em7 display];

[em8 setStringValue:@""];
[em8 display];

[em9 setStringValue:@""];
[em9 display];

[em10 setStringValue:@""];
[em10 display];

[em11 setStringValue:@""];
[em11 display];

[em12 setStringValue:@""];
[em12 display];

[em13 setStringValue:@""];
[em13 display];

[em14 setStringValue:@""];
[em14 display];

[em15 setStringValue:@""];
[em15 display];
```

```
[em16 setStringValue:@""];
[em16 display];

[em17 setStringValue:@""];
[em17 display];

[em18 setStringValue:@""];
[em18 display];

[em19 setStringValue:@""];
[em19 display];

[em20 setStringValue:@""];
[em20 display];


[qw1 setStringValue:@""];
[qw1 display];

[qw2 setStringValue:@""];
[qw2 display];

[qw3 setStringValue:@""];
[qw3 display];

[qw4 setStringValue:@""];
[qw4 display];

[qw5 setStringValue:@""];
[qw5 display];

[qw6 setStringValue:@""];
[qw6 display];

[qw7 setStringValue:@""];
[qw7 display];

[qw8 setStringValue:@""];
[qw8 display];

[qw9 setStringValue:@""];
[qw9 display];

[qw10 setStringValue:@""];
[qw10 display];

[qw11 setStringValue:@""];
[qw11 display];

[qw12 setStringValue:@""];
[qw12 display];

[qw13 setStringValue:@""];
[qw13 display];
```

```
[qw14 setStringValue:@""];
[qw14 display];

[qw15 setStringValue:@""];
[qw15 display];

[qw16 setStringValue:@""];
[qw16 display];

[qw17 setStringValue:@""];
[qw17 display];

[qw18 setStringValue:@""];
[qw18 display];

[qw19 setStringValue:@""];
[qw19 display];

[qw20 setStringValue:@""];
[qw20 display];


[bw1 setStringValue:@""];
[bw1 display];

[bw2 setStringValue:@""];
[bw2 display];

[bw3 setStringValue:@""];
[bw3 display];

[bw4 setStringValue:@""];
[bw4 display];

[bw5 setStringValue:@""];
[bw5 display];

[bw6 setStringValue:@""];
[bw6 display];

[bw7 setStringValue:@""];
[bw7 display];

[bw8 setStringValue:@""];
[bw8 display];

[bw9 setStringValue:@""];
[bw9 display];

[bw10 setStringValue:@""];
[bw10 display];

[bw11 setStringValue:@""];
```

```
[bw11 display];

[bw12 setStringValue:@""];
[bw12 display];

[bw13 setStringValue:@""];
[bw13 display];

[bw14 setStringValue:@""];
[bw14 display];

[bw15 setStringValue:@""];
[bw15 display];

[bw16 setStringValue:@""];
[bw16 display];

[bw17 setStringValue:@""];
[bw17 display];

[bw18 setStringValue:@""];
[bw18 display];

[bw19 setStringValue:@""];
[bw19 display];

[bw20 setStringValue:@""];
[bw20 display];




[ew1 setStringValue:@""];
[ew1 display];

[ew2 setStringValue:@""];
[ew2 display];

[ew3 setStringValue:@""];
[ew3 display];

[ew4 setStringValue:@""];
[ew4 display];

[ew5 setStringValue:@""];
[ew5 display];

[ew6 setStringValue:@""];
[ew6 display];

[ew7 setStringValue:@""];
[ew7 display];
```

```
    [ew8 setStringValue:@””];
    [ew8 display];

    [ew9 setStringValue:@””];
    [ew9 display];

    [ew10 setStringValue:@””];
    [ew10 display];

    [ew11 setStringValue:@””];
    [ew11 display];

    [ew12 setStringValue:@””];
    [ew12 display];

    [ew13 setStringValue:@””];
    [ew13 display];

    [ew14 setStringValue:@””];
    [ew14 display];

    [ew15 setStringValue:@””];
    [ew15 display];

    [ew16 setStringValue:@””];
    [ew16 display];

    [ew17 setStringValue:@””];
    [ew17 display];

    [ew18 setStringValue:@””];
    [ew18 display];

    [ew19 setStringValue:@””];
    [ew19 display];

    [ew20 setStringValue:@””];
    [ew20 display];
}
@end
```

Derivations

Question 3:

Derivation of *m*:

$$o = [(b - u) * q + m] * d = (b * q - u * q + m) * d = b * d * q - d * q * u - d * m$$
$$p = m - d * m + d * q * u - b * d * q$$
$$.102 * (b * q - q * u + m) = m - d * m + d * q * u - b * d * q$$
$$.102 * b * q - .102 * q * u + .102 * m = m - d * m + d * q * u - b * d * q$$
$$(.898 - d) * m = .102 * (b * q - q * u) + b * d * q - d * q * u$$
$$m = \frac{(.102 + d) * (b * q - q * u)}{.898 - d}$$