“Detalhes estruturais e funcionais do SADEPrev” [1]

Microssimulação usando Sadeprev – Simulador atuarial demográfico de RPPS

Resumo

A área de tecnologia da informação está constantemente inovando e criando tecnologia para automatizar processos de todas as áreas do conhecimento. As ciências atuariais não ficam de fora, softwares que se baseiam em micro simulações e conceitos da

área surgem com o fim de simular situações da sociedade e projetar possíveis resultados futuros. Este trabalho apresenta o software SADEPrev (Simulador Atuarial-Demográfico de Regimes Próprios de Previdência Social), software que utiliza dados e premissas populacionais para estimar a solvência entre recursos disponíveis e compromissos assumidos por um RPPS.

GLOSSÁRIO DE TERMOS

- A ***reserva matemática*** é o montante calculado em uma determinada data, correspondente aos encargos acumulados, destinado a pagamento futuro de benefícios, considerando o regulamento do plano e o plano de custeio em vigor, que corresponde à diferença entre o valor atual das obrigações com os benefícios do plano e valor atual dos direitos de contribuições futuras destinadas à cobertura destes mesmos benefícios.
- O ***fundo financeiro*** é o valor do capital do plano para cobertura dos compromissos futuros, ou seja, o valor que o RPPS tem.
- O ***Superávit*** acontece caso, ao final dos pagamentos das obrigações devidas, ainda esteja sobrando recursos provenientes das contribuições.
- O ***Déficit*** acontece quando o fundo não tem recurso suficiente para pagar todos os benefícios.
- O ***Benefício*** toda e qualquer prestação assegurada pelo plano de benefícios aos seus participantes e respectivos beneficiários, na forma e condições estabelecidas no regulamento.
- ***Dependente***: toda e qualquer pessoa física, assim considerada com relação a outra pessoa, conforme legislação.
- ***Invalidez***: (v. Incapacidade).

(abesprev, [s.d.])

INTRODUÇÃO

No Brasil, o desenvolvimento da previdência social resultou em regimes diferenciados para servidores públicos e para os demais trabalhadores, os Regimes Próprios de Previdência Social (RPPS) e o Regime Geral de Previdência Social (RGPS), respectivamente. Cada ente federativo pode ter um RPPS para seus servidores. Assim, a

União tem um RPPS, cada estado tem um RPPS, e cada município pode ter um RPPS (Corrêa, 2014).

Atualmente, muito se fala na reestruturação e reforma do regime previdenciário no Brasil. Com o objetivo de desafogar o sistema geral de previdência, existem grandes incentivos para a criação e manutenção dos RPPS, que gerenciam um número menor de servidores, fazendo qualquer mudança de modelo previdenciário ter consequências menos impactantes para a administração (Myrrha & Ojima, 2016).

Por outro lado, implantar RPPS em municípios pequenos, que não possuem capacidade de gestão e orçamentária para uma boa administração. Portanto, acreditar que os RPPS podem sem a solução para todos os problemas da previdência brasileira pode ser um grande erro (Myrrha & Ojima, 2016).

A maior parte dos RPPS são pequenos: cerca de 60% tem menos de 500 servidores ativos (Corrêa, 2014). Quanto menor a população, maior a variabilidade aleatória dos eventos demográficos e maior a possibilidade de observação de valores de eventos mais distantes dos valores esperados (Corrêa, 2014).

Outro fator que pode levar a resultados distantes dos esperados no estudo analítico de solvência de um RPPS é a metodologia adotada de calculo empregada. Além de escolhas indevidas das premissas demográficas, como idade de aposentadoria, idades médias de morte e invalidez, que, se modificadas podem alterar significativamente as projeções para uma mesma população.

O SADEPrev surge com o objetivo de analisar a solvências de RPPS, considerando o tamanho da população e a variabilidade dos eventos demográficos. Utilizando a metodologia de microssimulação, onde cada indivduo é submetido aos possíveis eventos demográficos ao longo de 75 anos. Ao final da simulação, o SADEPrev gera resultados importantes que seram mostrados em sessões posteriores deste artigo, além de propor uma alíquota de risco demográfico capaz de amortizar o déficit esperado.

Socsim

O **SocSim** é um software de código aberto, escrito na década de 1970, que utiliza microssimulação para submeter indivíduos de uma população em eventos demograficos como morte, mudanças de estado civil e nascimentos. Tendo como foco, a projeção e análise da mudança de composição e caracteristicas da redes familiares (Mason, 2010).

Como é dada uma grande importancia para os eventos de modificação da estrutura familiar, fica interessante observar como o SocSim forma um casamento por exemplo. Não basta que dois individuos de sexo oposto estejam agendados para o evento “casamento”, os individuos precisam estar em uma fila de casamento, com isso, a aleatoriedade do casamento fica apenas por conta do agendamento do evento, pois o parceiro escolhido não será qualquer um da população.

O software pode ser extendido ou modificado, tendo como pre-requisito para o conhecimento na linguagem C, na qual o software foi desenvolvido (Mason, 2010).

	SOCSIM	SADEPREV
Nome	Social simulation	Simulador Atuarial-Demográfico de Regimes Próprios de Previdência Social
Objetivo	Sistema de microssimulação demográfica. utilizado para projeção ou retroprojeção de redes familiares.	Utilizar dados e premissas populacionais para estimar a solvência entre recursos disponíveis e compromissos assumidos por um RPPS.
Origem	Desenvolvido por *Eugene A. Hammel* e *Kenneth Wachter* (Berkeley, California, 1970).	Software criado a partir do projeto "*Desenvolvendo a gestão de RPPS: Um programa para auxílio dos gestores de previdência de servidores públicos*". Financiado pelo programa de Apoio à Extensão Universitária MEC/SISU 2016
Forma de simular	A simulação começa gerando uma população de indivíduos não relacionados, então eventos demográficos (casamento, morte, maternidade) são criados para cada indivíduo no decorrer do tempo de simulação e de acordo com suas respectivas probabilidades. Os eventos demográficos continuam acontecendo mês a mês, até o final do período determinado na simulação. Todos os eventos e indivíduos são registrados durante a simulação e qualquer indivíduo pode ser selecionado como uma referência e as informações sobre a estrutura familiar deste podem ser observadas.	A população inicial retrata a população dos contribuintes e beneficiários de um RPPS. Informações como idade, sexo, salário e situação no RPPS (ativo, aposentado, inválido, cônjuge pensionista ou filho pensionista são requeridas. O indivíduo é submetido a um fluxo de mudanças de estados, onde a cada momento de mudança (dependendo da idade e do sexo do indivíduo) é feito um sorteio aleatório baseado no teste de Bernoulli para determinar o próximo estado. Considera-se aleatórias simultaneamente as premissas de morte, invalidez, probabilidade de ter cônjuge e filho, e idade do cônjuge e filho.
População estimada	Simula uma amostra de uma população real de um país em um determinado momento.	Simula a população segurada do RPPS a cada tempo.

Casamento	A partir do instante de tempo que o indivíduo está programado para se casar, o sistema escolhe um indivíduo existente da população.	A partir da mudança de estado de um indivíduo para "Morto", é feito um teste para gerar ou não seu cônjuge, baseado no teste de Bernoulli, com probabilidades de sucesso iguais as probabilidades contidas em tábuas criadas a partir de estudos.
Há em comum	O principal ponto em comum é simulação com metodologia baseada em micro simulações, apesar de serem micro simulações com características distintas.	

Pensim

Camsim

Software de microssimulação

AimSun: Software Brasileiro desenvolvido e mantido pela empresa Fratar, empresa especializada em estudos de Engenharia de Tráfego, Transportes, Pedestres e Logística, que utiliza ferramentas analíticas e de microssimulação em seus estudos. O Aimsun tem como possibilidades principais:

- **Análise de trafégo e transporte**
  - **Estudo de capacidade de rodovias**
  - **Plano de mobilidade**
  - **Otimização semafórica**
- **Pesquisa de trafégo**
- **Consultoria logistica**
  - **Simulação de processos**
  - **Otimização de recursos**
- **Fluxo de multidões.**
  - **Grandes eventos**
  - **Estádios**
  - **Áreas urbanas**

1. Contextualização. Familiariza leitor com grande área e mostra importância da grande área. Falar rapidamente sobre RPPS pequenos e variabilidade das funções demográficas.
2. Gap ou lacuna. Lacunas da área, questões em aberto na área. A dificuldade de incluir o tamanho populacional na análise atuarial é o que faz surgir o Sadeprev. Métodos atuariais comumente usados se focam na distribuição média, e não na variabilidade.
3. Estado da arte. Onde está a fronteira do conhecimento. Artigos mais recentes possível. O que já foi feito. Falar sobre os demais programas de microssimulação em demografia, previdência, trânsito, e outras áreas.
4. Objetivo. O que vou fazer para contribuir com área e como vou fazer. Se possível incluir principais resultados. Resumir o que SADEPREV FAZ.

METODOLOGIA

1 Estrutura do Sadeprev

O SadePrev é um software livre que tem como área de atuação o regime proprio de previdência social (RPPS), que são as previdencias municipais. A licença do software está registrada em nome da Universidade Federal do Rio Grande do Norte e permite que ele seja modificado por colaboradores em parceria com a instituição.

O SadePrev é um software desktop, desenvolvido na linguagem R, utilizando o framework Shiny para construção de uma interface visual com o usuário.

As simulações realizadas pelo software dependem de uma quantidade significativa de dados, refetente aos dados dos individuos de uma população. Atualmente o SadePrev importa estas informações através de arquivos no formato (.xls ou .csv), os resultados, que são gerados na maquina em que o programa foi executado, estão tambem nestes formatos e no formato de imagens.

1.1 Simulação da população e eventos demograficos

Atualmente as simulações podem ser realizadas com dados de uma população real ou uma população fictícia gerada. Visto que em uma simulação com dados reais não é necessário gerar uma população, pois a população é construida com os dados reais de cada indivíduo, a geração da população, descrita nessa seção, diz respeito a populações ficticias.

Com a população construida, os eventos demograficos acontecem de forma igual para os casos de população ficticia ou real.

O Sadeprev simula uma população fechada, o que significa que esta população não admite entrada de novos servidores (Corrêa, 2014), logo, quando um servidor passa para a situação de inatividade, a população não é reposta com um servidor ativo. Este modelo de população reflete uma situação de extinção de um RPPS (Corrêa, 2014).

A população inicial é gerada com 500 servidores (para o caso de uma população fictícia). São conciderados os seguintes dados para a população: Sexo, idade, remuneração, data de nascimento, situação no RPPS (inicial mente ativo), data de ingresso no ente federativo. Os dados para simulação com populações fictícias são gerados através de funções que "sorteiam" as informações considerando estimativas baseadas em estudos prévios e premissas como tábuas de mortalidade e entrada em invalidez.

Construida a população, os individuos estão submetidos aos eventos demograficos, ou seja, os eventos de entrada e saida da população de ativos e inativos/beneficiários (Corrêa, 2014). A figura H mostra as possibilidades de eventos demograficos.

Figura H - Eventos demograficos de entrada e saída da população possiveis para ativos e beneficiários

População	Entrada	Saída
De ativos	– Entrar no serviço público	– Morrer – Ficar inválido – Se aposentar
De beneficiários	– Se aposentar ou invalidar – Ser cônjuge de servidor e servidor morrer – Ser filho de servidor e servidor morrer	– Morrer – Completar 21 anos, se filho de servidor

(Corrêa, 2014)

1.3 **Dimensões de análise e significados das matrizes de dados**

Esta seção vai falar sobre as três matrizes principais da simulção, estas matrizem armazenam os dados da população e estará sujeita a modificações em função dos acontecimentos demograficos ao longo do tempo. Duas destas matrizes armazenam as informações de cada individuo e a ultima é resultante das duas primeira e contém informações sobre o fundo previdenciário. O tempo se refere ao tempo em que o servidor atúa no RPPS, ativo ou inativo, e é fixado em 75 anos.

A primeira matriz tem três dimensões (*tamanho da população* X *tempo* X *rodadas*). Para cada posição da matriz temos o estado de um indivíduo no RPPS em função do tempo. O tempo de contribuição/atividade de cada individuo é sorteado previamente. No momento em que um indivíduo atinge o fim do tempo de atividade, um novo sorteio é realizado e ele morre ou passa para a inatividade, podendo ficar inválido, deixar dependentes (cônjuge ou filhos) ou não.

A segunda matriz, também com três dimensões (*tamanho da população* X *tempo* X *rodadas*), armazena os valores das contribuições ao longo tempo, podendo ser positiva ou negativa, representando uma contribuição ou um benefício respectivamente.

A terceira matriz é bidimencional e é resultado das duas primeiras. De acordo com o estado do servidor (primeira matriz) e do valor que cada individuo contribui para o fundo ou recebe de beneficio em um determinado momento do tempo, podemos somar todos estes valores, obtendo o valor do fundo do RPPS ao final de um ano.

1.4 **Eventos demográficos ao longo do tempo**

Um indivíduo inicia sua "jornada" na simulação como "Ativo" e se mantém neste estado por um tempo sorteado até a sua morte ou invalidez. Neste momento também é estimada a idade de sua aposentadoria.

Após passado o tempo de atividade do servidor, ou ele se aposenta ou um novo sorteio acontece para decidir se ele estará invalido ou morto. Para os casos de aposentadoria e invalidez, é gerado um tempo até a morte. O servidor irá receber benefício como aposentado ou invalido até o dia de sua morte. Nesse momento mais um sorteio é realizado, a fim de saber se o indivíduo deixou dependentes (cônjuge ou filhos) ou não. Para o caso de deixar dependentes, é gerada uma idade aleatória e um tempo até a morte para os dependentes. Estes seguem recebendo beneficio em nome do servidor ao qual dependem, até o momento da morte, e neste momento o beneficio é abortado. Caso contrário, o servidor não deixou dependentes e não terá mais o beneficio.

A figura "K" mostra graficamente como se desenvolve os acontecimentos descritos acima.

FIGURA K: Eventos demográficos incidentes sobre servidores.

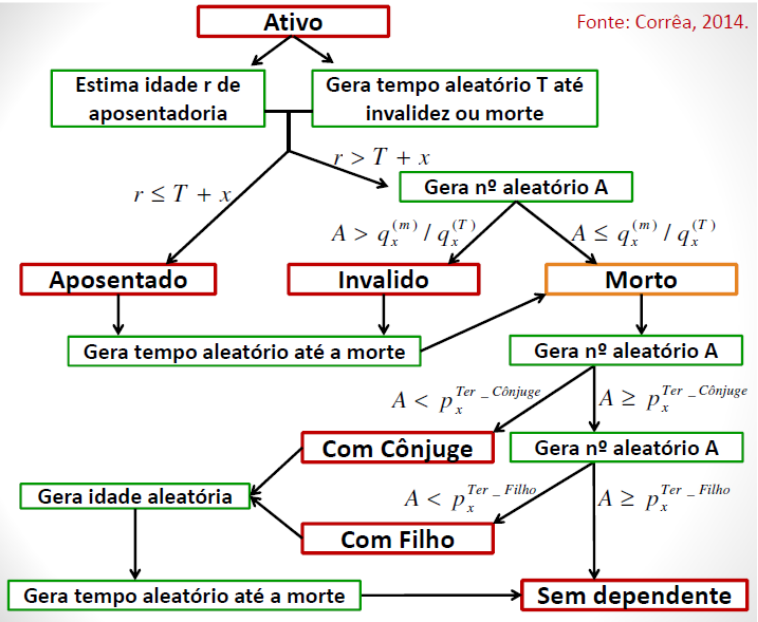
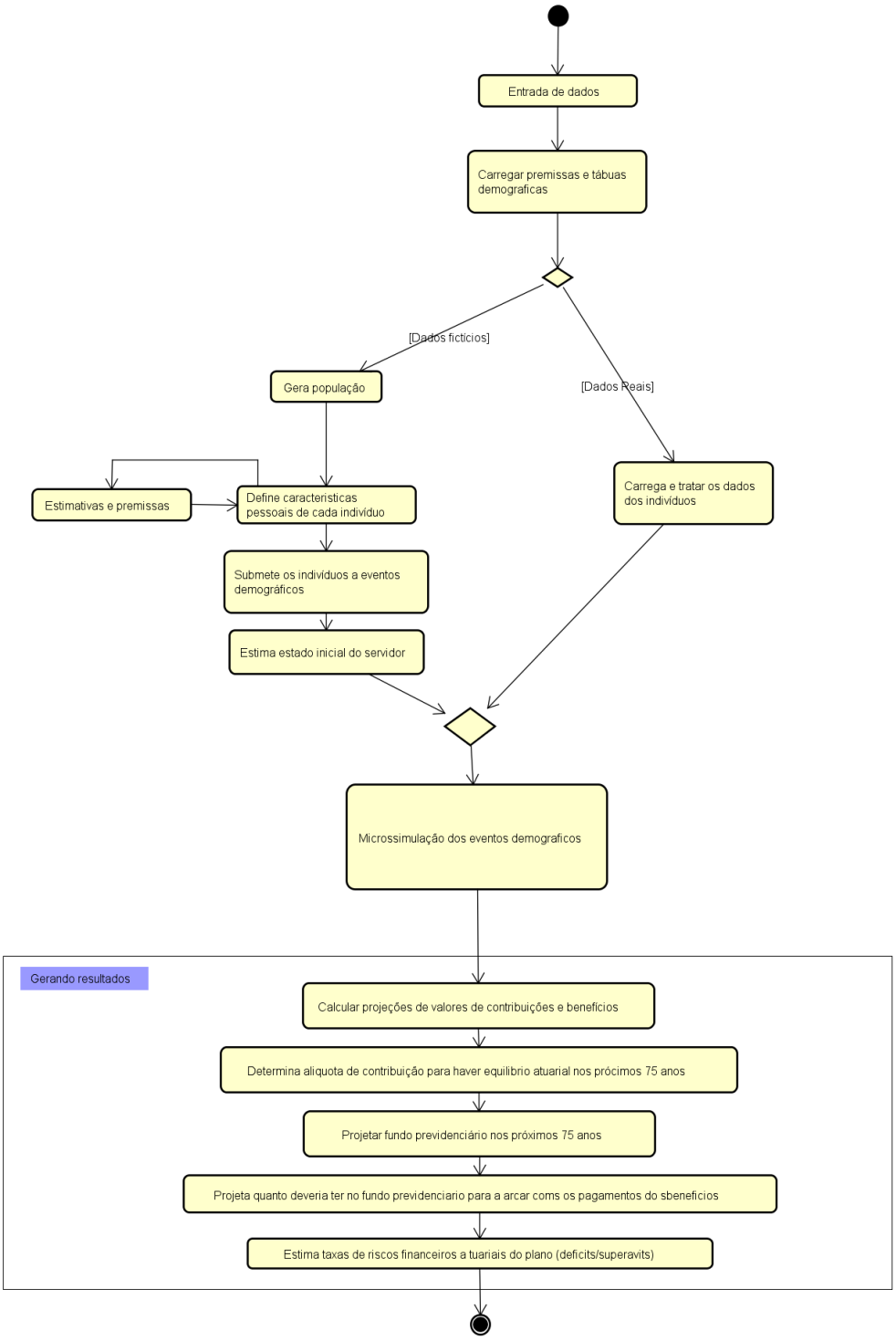


Figura x: Diagrama de atividades de uma execução do SADEPrev



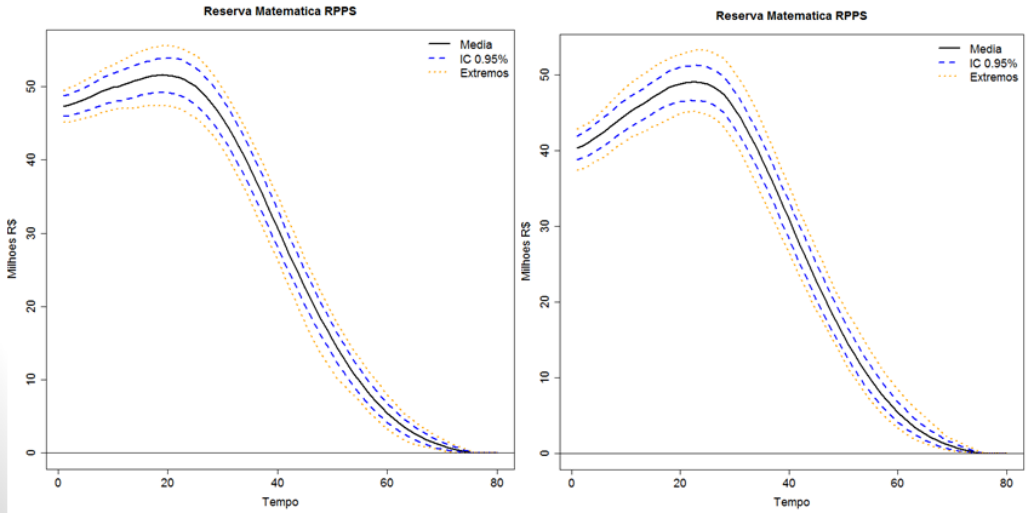
1.2 Resultados

O SADEPrev gera vários resultados relacionados a saúde do RPPS analisado. Nesta sessão são abordados os resultados considerados mais importantes.

Reserva matemática

Interpretasção:

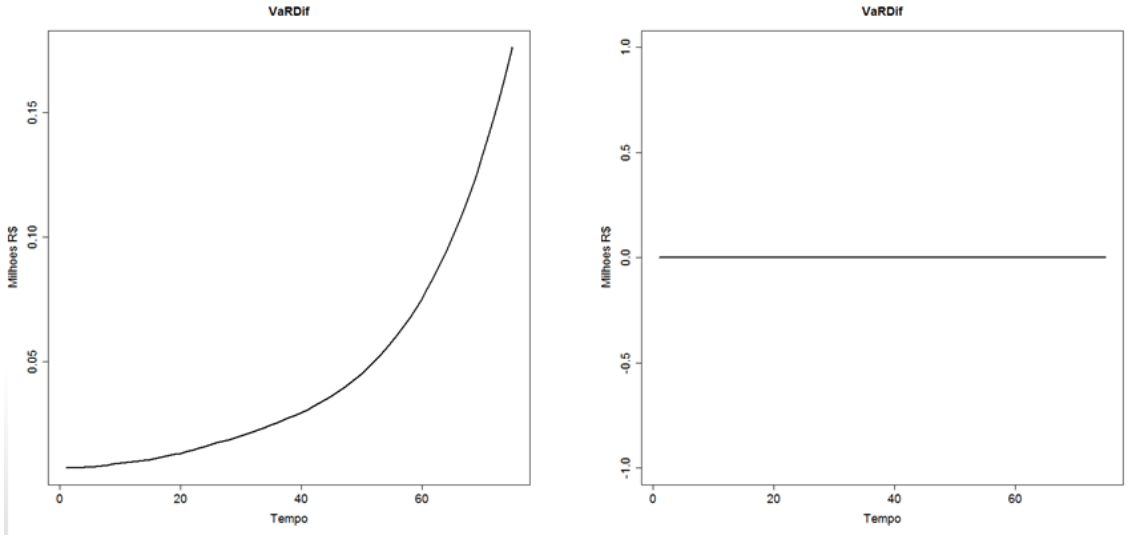
(Déficit) (Superávit)



VaRDif

Mesma medida de solvência do VaR aplicada à diferença, a identificação do déficit atuarial.

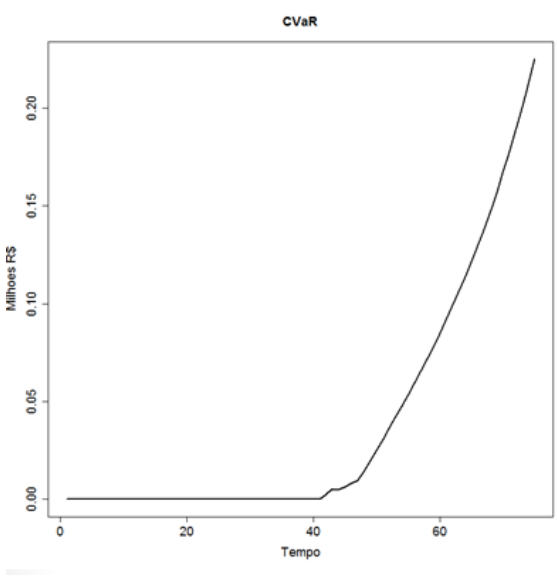
(Déficit) (Superávit)



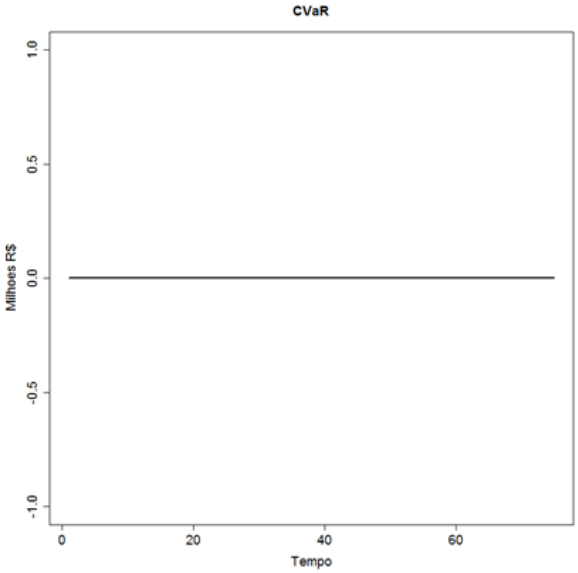
CVar

Interpretação: Espera-se ter um déficit médio de aproximadamente R$ 200.000 após 60 anos de simulação.

(Déficit)



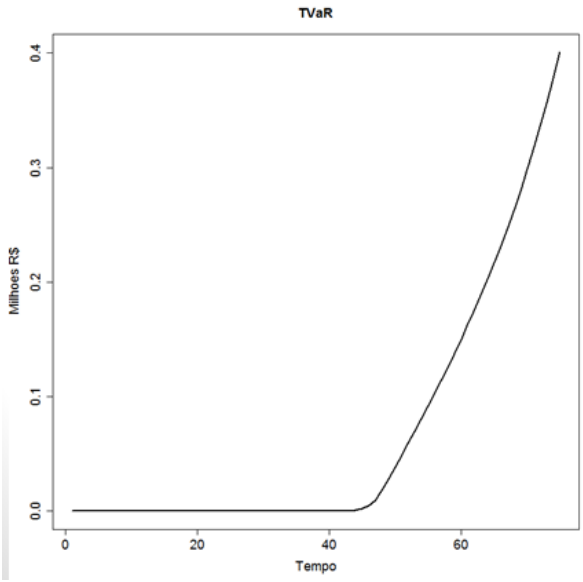
(Superávit)



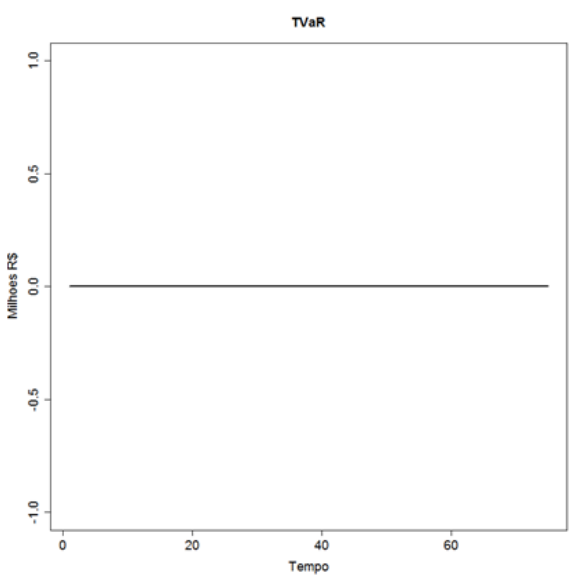
Tvar

Medida de solvência que mede a perda média acima do VaR. Quanto maior for a perda, mais arriscado é o plano.

(Déficit)



(Superávit)



De forma genérica, a simulação computacional consiste em modelar computacionalmente um “mini-mundo” (abstração do mundo real) onde objetos e ambientes do mundo real tem suas caracteristicas e comportamento modelados virtualmente (Maia, 2007). A simulação computacional é uma ferramenta que permite a diversos profissionais, a possibilidade da obsevação prévia de circunstâncias que envolvem seus projetos, através de um ambiente simulado onde observações e conclusões antecipadas podem ser de grande importância para o desenvolvimento dos seus trabalhos. (Maia, 2007).

A aplicabilidade da microssimulação abrange áreas como a biologia, demografia e arquitetura. Um exemplo está no trabalho “*Estudo de Caso: uso de microssimulação de tráfego para revisão do projeto da Av. Manoel Ribas, em Curitiba, com inclusão de ciclofaixa*” (MALUCELLI., 2013). Este trabalho foi motivado pelo crescimento acelerado do tráfego de veiculos nas grandes cidades brasileiras. O trabalho mostra um planejamento feito para a Avenida Manual Ribas, no minucipio de Campo Magro, região metropolitana de Curitiba. O estudo e o planejamento das reformas necessárias foram feitos com base em uma miscrossimulação do tráfego naquela localidade.

2 Problemas durante o desenvolvimento

Durante o desenvolvimento do **SADEPrev**, nos deparamos com alguns problemas críticos. As próximas seções tratam de como os problemas de alocação de memória e tempo de execução podem ser minimizados e qual solução foi escolhida no desenvolvimento do **SADEPrev**.

2.1 problema de alocação de memória

O interpretador R implementa um limite na alocação memória (R Core Team, 2017c), ou seja, qualquer variável instanciada não pode ocupar um espaço maior que o espaço determinado pela função “memory.limit()”. O tamanho de memória alocável dependa da arquitetura do hardware da máquina, do sistema operacional sobre o qual o programa R está sendo executado e da compilação (versão) do R (R Core Team, 2017c).

No geral, podemos observar as limitações para as seguintes características de máquina:

Tabela n: Limite de memória alocavel baseado nas configurações de máquina e compilação de versão.

memória alocável (RAM)	Compilação do R	Arquitetura de SO
2GB	32-bits	32-bits
4GB	64-bits	32-bits
8TB	64-bits	64-bits

(R Core Team, 2017c)

Deve-se utilizar uma versão do R de acordo com a arquitetura do sistema operacional, se utilizamos um sistema operacional 64-bits e executamos um programa com uma versão do R de 32-bits, não teremos o mesmo limite de memória se usássemos uma versão do R de 64-bits (R Core Team, 2017c).

Outra questão importante para evitar este problema, é analisar se a estrutura de dados utilizada causa problema na alocação do objeto em memória. Um vetor pode gerar um problema de alocação, não por indisponibilidade do espaço total que o objeto ocupa, mas por falta de espaço contínuo em memória, já que um vetor precisa de um espaço contínuo livre, ao contrário de uma lista, que pode ser alocada em espaços não contíguos de memória (Celes & Rangel, 2002).

Na implementação do **SADEPrev**, o problema estava na função que continha os algoritmos de simulação e cálculos atuariais. Esta função retornava uma lista de dados que extrapolava o limite máximo de espaço alocável para um objeto. Para solucionar isto, dividimos as simulações em blocos, ou seja, se faziamos 1000 rodadas de simulação e tínhamos como retorno uma lista que extrapolava o limite de memória. Decidimos dividir as 1000 rodadas por 20, executando os blocos de 50 simulações separadamente e tendo como retorno uma lista de dados menor a cada rodada, podendo então trabalhar com elas por partes.

Seguem algumas práticas para minimizar problemas de alocação de memória no R:

- Possibilidade de utilizar pacotes como “ff” (Adler, Gläser, Nenadic, Oehlschlägel, & Zucchini, 2014), “bigmemory” (Kane, Emerson, Haverty, & Jr, 2016) ou “SOAR” (Venables & Brahm, 2013), que economizam memória RAM, arquivando parte dos dados no disco (HD). Este processo é feito através de um algoritmo parecido com a paginação (Tanenbaum, 2009), realizada pelos sistemas operacionas, processo no qual

usa-se o disco (HD) como uma extensão da memória RAM, porém, manipular dados no disco é um processo mais custoso que manipular dados em memória RAM (Tanenbaum, 2009), portanto, é importante verificar se o desempenho do programa também é um fator importante.

- Solicitar que o sistema operacional limpe objetos alocados que não estão mais sendo utilizados, através da função "gc()" (R Core Team, 2017b). Neste caso, o sistema operacional seleciona objetos pouco utilizados ou sem referência para serem desalocados, através de suas políticas (algoritmos) de prioridade (R Core Team, 2017b).
- Apagar explicitamente no código, objetos alocados em memória através da função "remove(objeto)" (R Core Team, 2017e).

2.2 Problema de desempenho

O R é nativamente mono-thread (Lim & Tjhi, 2015), isso significa que todo programa R é executado em apenas uma linha de execução se isso não for modificado explicitamente no código. Ter uma execução em apenas uma linha de execução significa que o código do seu programa será interpretado linha por linha, sequencialmente, sem a possibilidade de execuções de tarefas simultâneas. Algumas linguagens de programação atuais dão suporte ao paralelismo implicitamente, onde o programador não precisa se preocupar com a implementação em muitas partes do código para que seu programa execute operações em paralelo. A linguagem R não é uma delas.

As funções vetorizadas são uma alternativa na melhoria de desempenho, quando precisamos, por exemplo, percorrer vetores ou qualquer estrutura parecida. A função sum() (R Core Team, 2017d) é um exemplo, ela é uma função vetorizada que realiza a soma de todos os elementos de um vetor. A primeira ideia que se tem ao realizar uma operação como esta, é percorrer todos os elementos através de um laço de repetição, porém, funções vetorizadas conseguem aplicar operações nos vários objetos de um vetor em uma mesma operação. As funções da família "apply" (apply, tapply, sapply e rapply), disponíveis na biblioteca/pacote Base (R Core Team, 2017a) também são vetorizadas e podem ser alternativas ao uso de laços de repetição quando precisamos realizar operações sobre estruturas como um vetor ou matriz.

Existe também o pacote RevoScalerR (Revolution Analytics, 2013) que contém funções dedicadas a aplicações em Data Science e Big Data. Portanto, utilizado para

manipulação e armazenamento de grande ou pequena quantidade de dados, aumentando desempenho e poder de armazenamento de dados.

O R dispõe também de algumas bibliotecas para implementação de paralelismo, por exemplo: **Parallel** (Calaway, Analytics, Weston, & Tenenbaum, 2015) e **snow** (Tierney, Rossini, Li, & Sevcikova, 2016).

Para a implementação no **SADEPrev**, escolhemos a biblioteca que tinha a implementação que mais se adequava ao que já tínhamos feito, essa implementação está na função **foreach** da biblioteca **foreach** (Calaway, Analytics, & Weston, 2015) (Contida como dependência da biblioteca **Parallel**). O objetivo desta função é criar uma linha de processamento independente para cada iteração de um laço de repetição, portanto, cada bloco de simulações que tínhamos criado anteriormente, não precisa esperar toda a ordem de execução de um laço de repetição, sendo possível que mais de um bloco seja executado simultaneamente, dependendo da quantidade de núcleos de processamento do computador. É fácil notar que, caso tivéssemos dependência de dados entre os blocos de simulações, isto é, se o segundo bloco dependesse do resultado do primeiro, não poderíamos implementar desta forma, pois teríamos resultados inesperados ou outros erros de execução.

Após a implementação, os testes foram feitos para o laço de repetição principal do programa, que é um laço que basicamente chama três funções e preenche uma matriz a cada iteração. Os parâmetros utilizados nas funções chamadas são: Tamanho da população, número de rodadas, blocos e tempo. Para estes testes, fixamos os parâmetros em: Número de repetições das simulações de Monte Carlo igual a 10.000, 10 blocos e horizonte de projeções de 75 anos.

Foram realizados testes com populações iniciais de 100 e de 1000 indivíduos. Para cada um destes testes verificou-se o tempo de execução do programa sem compilação ou paralelismo, com paralelismo, e com compilação e paralelismo. Os resultados são apresentados na Tabela x.

Tabela x: Tempo de execução, em minutos, e ganho de eficiência em relação ao modelo sequencial para simulações com 100 e 1000 indivíduos iniciais, segundo os modelos sequencial, com paralelismo, e com paralelismo e compilações de funções conjuntamente.

	Tempo de execução (min)		Ganho de eficiência	
População inicial	**100**	**1000**	**100**	**1000**
Sequencial (Comando "for")	3,29	15,67		
Em paralelo (Comando "foreach")	1,53	8,13	53,6%	48,1%
Em paralelo + compilação das funções	1,48	7,28	55,2%	53,5%

Como mostra a Tabela x, o tempo de execução está diretamente relacionado ao tamanho inicial da população. Quando o programa SADEPREV utiliza apenas comandos sequenciais o tempo de execução de 10.000 repetições de Monte Carlo é de 15,67 minutos para uma população inicial de 1.000 indivíduos, mas de apenas 3,29 minutos para uma população inicial de 100 pessoas. Portanto, quanto maior a população inicial, maior o tempo de processamento.

Ademais, percebe-se, também pela tabela, que o paralelismo e a compilação diminuem o tempo de processamento do programa. Considerando a população de 100 indivíduos iniciais, enquanto o modelo sequencial foi executado em 3,29 minutos, o modelo em paralelo foi executado em 1,53 minutos, uma redução de 53,6% do tempo gasto. A compilação das funções aprimorou ainda mais a eficiência do programa, que demorou apenas 1,48 minutos para ser executado nessa situação, um ganho de 55,2% em relação ao modelo sequencial, e de 3,4% em relação ao modelo com paralelismo, apenas.

Ressalta-se, ainda, que o ganho de eficiência foi maior para uma população inicial menor (55,2% para 100 indivíduos) que para a maior (53,5% para 1000 indivíduos) ao se incorporar o paralelismo e a compilação. Apesar disso, foi significativo nas duas situações, já que com o tempo de execução caiu em mais de 50% nas duas situações.

É interessante observar que o uso do paralelismo melhorou o modelo de simulação do **SADEPrev**, pois como antes tínhamos as simulações sequenciais, um indivíduo não poderia morrer ao "mesmo tempo" que outro por exemplo, o que não condiz com a realidade. Portanto, implementando paralelismo nos aproximamos um pouco mais do real.

REFERÊNCIAS

abesprev. ([s.d.]). GLOSSÁRIO DE TERMOS TÉCNICOS ATUARIAIS. Recuperado de http://www.abesprev.com.br/GLOSS%C3%81RIO%20DE%20TERMOS%20T%C3%89CNICOS%20ATUARIAIS.pdf

Adler, D., Gläser, C., Nenadic, O., Oehlschlägel, J., & Zucchini, W. (2014). ff: memory-efficient storage of large data on disk and fast access functions (Versão 2.2-13). Recuperado de https://cran.r-project.org/web/packages/ff/index.html

Calaway, R., Analytics, R., & Weston, S. (2015). foreach: Provides Foreach Looping Construct for R (Versão 1.4.3). Recuperado de https://cran.r-project.org/web/packages/foreach/index.html

Calaway, R., Analytics, R., Weston, S., & Tenenbaum, D. (2015). doParallel: Foreach Parallel Adaptor for the "parallel" Package (Versão 1.0.10). Recuperado de https://cran.r-project.org/web/packages/doParallel/index.html

Celes, W., & Rangel, J. L. (2002). Estruturas de Dados.

Corrêa, C. S. (2014). *Tamanho populacional e aleatoriedade de eventos demográficos na solvência de RPPS municipais capitalizados*. UFMG/Cedeplar, Belo Horizonte, MG.

Kane, M. J., Emerson, J. W., Haverty, P., & Jr, and C. D. (2016). bigmemory: Manage Massive Matrices with Shared Memory and Memory-Mapped Files (Versão 4.5.19). Recuperado de https://cran.r-project.org/web/packages/bigmemory/index.html

Lim, A., & Tjhi, W. (2015, janeiro 30). R High Performance Programming. Recuperado 8 de maio de 2017, de http://www.barnesandnoble.com/w/r-high-performance-programming-aloysius-lim/1120930067

Maia, F. V. B. (2007). *CALIBRAÇÃO E VALIDAÇÃO DE MODELOS DE MESO E MICROSSIMULAÇÃO DO TRÁFEGO PARA A AVALIAÇÃO DE INTERVENÇÕES TÁTICO-OPERACIONAIS NA MALHA VIÁRIA URBANA*. Universidade Federal do Ceará, Fortaleza - CE. Recuperado de http://livros01.livrosgratis.com.br/cp053304.pdf

MALUCELLI., F. C. (2013). Estudo de Caso: uso de micro simulação de tráfego para revisão do projeto da Av. Manoel Ribas, em Curitiba, com inclusão de ciclofaixa. Recuperado de https://www.sinaldetransito.com.br/artigos/estudo_Manoel_Ribas.pdf

Mason, C. (2010). Socsim Oversimplified. Recuperado de http://lab.demog.berkeley.edu/socsim/CurrentDocs/socsimOversimplified.pdf

Myrrha, L., & Ojima, R. (2016, janeiro). DINÂMICA DEMOGRÁFICA, GESTÃO PÚBLICA E REGIMES PRÓPRIOS DE PREVIDÊNCIA SOCIAL: OPORTUNIDADES E DESAFIOS PARA OS SERVIDORES E MUNICIPIOS, *v. 17*, 59–74.

R Core Team, R. C. T. (2017a). R: The R Base Package. R Core Team. Recuperado de https://stat.ethz.ch/R-manual/R-devel/library/base/html/00Index.html

R Core Team, R. C. T. (2017b, junho 5). R: Garbage Collection. R Core Team <R-core@r-project.org. Recuperado de https://stat.ethz.ch/R-manual/R-devel/library/base/html/gc.html

R Core Team, R. C. T. (2017c, junho 5). R: Memory Limits in R. R Core Team <R-core@r-project.org. Recuperado de https://stat.ethz.ch/R-manual/R-devel/library/base/html/Memory-limits.html

R Core Team, R. C. T. (2017d, junho 5). R: Sum of Vector Elements. R Core Team <R-core@r-project.org. Recuperado de https://stat.ethz.ch/R-manual/R-devel/library/base/html/sum.html

R Core Team, R. C. T. (2017e, junho 5). Remove Objects from a Specified Environment. R Core Team <R-core@r-project.org. Recuperado de https://stat.ethz.ch/R-manual/R-devel/library/base/html/rm.html

Tanenbaum, A. S. (2009). *Sistemas Operacionais Modernos* (3ª). Pearson.

Tierney, L., Rossini, A. J., Li, N., & Sevcikova, H. (2016). snow: Simple Network of Workstations (Versão 0.4-2). Recuperado de https://cran.r-project.org/web/packages/snow/index.html

Venables, B., & Brahm, based on original code by D. (2013). SOAR: Memory management in R by delayed assignments (Versão 0.99-11). Recuperado de https://cran.r-project.org/web/packages/SOAR/index.html