

Manual de utilização do servidor EcoEvol

INCT Ecologia, Evolução e Conservação da Biodiversidade

Contato do administrador: csdambros@gmail.com

Maio 05, 2017

Índice

1	Informações gerais	1
2	Acesso ao servidor	2
2.1	Interface web	2
2.2	SSH	6
2.3	RDP: Programas para Windows	6
2.4	FTP/sFTP	7
3	Primeiro acesso	9
4	Escolha da máquina	10
5	Paralelização: A grande vantagem de utilizar o servidor	11
5.1	Quando meu script pode ser paralelizado?	11
5.2	Exemplo das funções <code>for</code> e <code>lapply</code>	11
5.3	Paralelizando em um ou múltiplos computadores	14
5.4	Começando e encerrando a paralelização automaticamente (altamente recomendado)	15
5.5	Quando é vantagem paralelizar e como otimizar o processo	16

1 Informações gerais

Este arquivo descreve boas práticas para o uso do servidor EcoEvol e fornece orientações básicas para a sua utilização. O servidor está designado especialmente para a execução do programa R e outros programas disponíveis para a plataforma Unix. Também é possível a execução de programas em Windows e a configuração do servidor para a execução de diferentes demandas dos usuários.

Ao utilizar o servidor, é importante que o usuário siga algumas recomendações para que todos possam usufruir do serviço sem interrupções:

- **O uso dos computadores é estritamente profissional, para fins acadêmicos** Qualquer outra aplicação (comercial, pessoal) não é permitida. Como regra geral, não utilize o servidor para fazer nada que você não faria na frente do seu orientador/supervisor. É terminantemente proibido o uso do servidores para baixar da internet conteúdo de entretenimento pessoal (filmes, videos, fotos, música), ou material de distribuição ilegal (software pirata).
- **Nunca forneça sua senha para outros usuários** Todos os membros do grupo de Ecologia da UFG podem ter um login, e portanto não há nenhuma necessidade de compartilhamento de logins. Além disso, todo acesso gera um registro, e qualquer uso indevido pode ser rastreado a um login. Portanto, não empreste sua senha, pois você poderá ser responsabilizado por tudo que fizerem usando-a.

- **Não deixe o servidor processando trabalhos que não são mais necessários** Apesar da grande capacidade de processamento do servidor, múltiplos usuários estarão simultaneamente utilizando as máquinas, podendo eventualmente sobrecarregar algum computador. Isto poderá fazer com que a capacidade de processamento seja reduzida caso processos desnecessários continuem executando. Além disso, não deixe nos servidores dados que você não utilizará mais, pois o espaço que os seus dados/resultados ocupam poderia ser usado por outro usuário.
- **Não assumo, nem por um segundo, que os servidores servem de backup de dados** Coloque nos servidores o mínimo de arquivos que você precisa para sua análise, retire seus resultados assim que possível, e remova os dados originais ao terminar. Apesar do esforço para administrar e manter os computadores, não há qualquer garantia que os seus dados estão seguros. Caso necessário, os computadores poderão ser desligados, re-iniciados, formatados a qualquer momento, sem aviso prévio. Não conte com a sorte! Se a sua pesquisa é importante para você, seja precavido e mantenha backup fora dos servidores.
- **Contate os administradores caso detecte qualquer falha/problema** Sua dificuldade em acessar um recurso pode ajudar a sanar o problema e evitar problemas similares para outros usuários no futuro. Isso inclui: dificuldade para acessar um endereço fornecido (mesmo que outros ainda funcionem), travamento na execução de um comando, entre outros.
- **Mantenha seu cadastro atualizado** O e-mail é o meio de comunicação dos administradores com os usuários. Sempre que uma atualização importante no servidor for realizada ou algum computador for entrar em manutenção, os usuários receberão um aviso. Caso altere seu e-mail, envie o novo endereço para a administração.
- **Respeite os demais usuários** Não tente copiar ou interferir no processo de outros usuários sem autorização. O servidor possui alguns elementos de segurança que irão prevenir abusos ou acesso de pessoas mal intencionadas. Além da prevenção, toda operação nos servidores é registrada, e poderá ser rastreada e o login do usuário será identificado. É importante que todos os usuários sigam condutas éticas durante a utilização do servidor.
- **Utilize os recursos de forma consciente** Se você necessita utilizar múltiplos processadores, não envie todo o seu trabalho para um único computador. Visite a seção sobre paralelização para saber como espalhar seu trabalho na cluster. Isso evitará sobrecarregar um único computador, dificultando o uso pelos demais usuários. Além disso, não faça múltiplos acessos para enviar seu trabalho. Por exemplo, não entre uma vez na máquina 1, 2 3, ... ao mesmo tempo para enviar múltiplos processos. Dê preferência a processos paralelos que irão automaticamente distribuir o seu trabalho.

2 Acesso ao servidor

O servidor pode ser acessado utilizando pelo menos quatro métodos: Interface Web, Secure Shell (ssh), RDP e sFTP. Se você pretende utilizar somente o programa R, provavelmente você precisará utilizar somente a interface web. Para outros programas que executam no terminal, como python, lagrange, MrBayes, etc, será necessário acessar o servidor através de um terminal utilizando ssh. O protocolo sFTP pode ser utilizado para abrir um diretório do servidor em seu computador e transferir arquivos. O protocolo RDP funcionará somente para a execução de programas executáveis exclusivamente em Windows. Com um login e senha, você possui acesso a todos os computadores do servidor e a todos os protocolos.

2.1 Interface web

O Rstudio está configurado para a interface web e pode ser acessado pelo navegador de internet, como google Chrome, Firefox, Opera, etc. Para acessar esta funcionalidade não é necessário ter nenhum programa específico instalado em seu computador pessoal além do navegador (ex. você não precisa do R ou Rstudio em seu computador). Além disso, o acesso pode ser feito a partir de qualquer dispositivo com acesso à internet

(computadores, celulares, tablets, etc.). Você poderá acessar o Rstudio e todas as suas funcionalidades logando em qualquer uma das máquinas disponíveis através dos seguintes endereços (veja a seção **Escolha da Máquina**):

Slave1: <http://cluster.eco.br:10001>

Slave2: <http://cluster.eco.br:10002>

Slave3: <http://cluster.eco.br:10003>

Slave10: <http://cluster.eco.br:10010>

Slave11: <http://cluster.eco.br:10011>

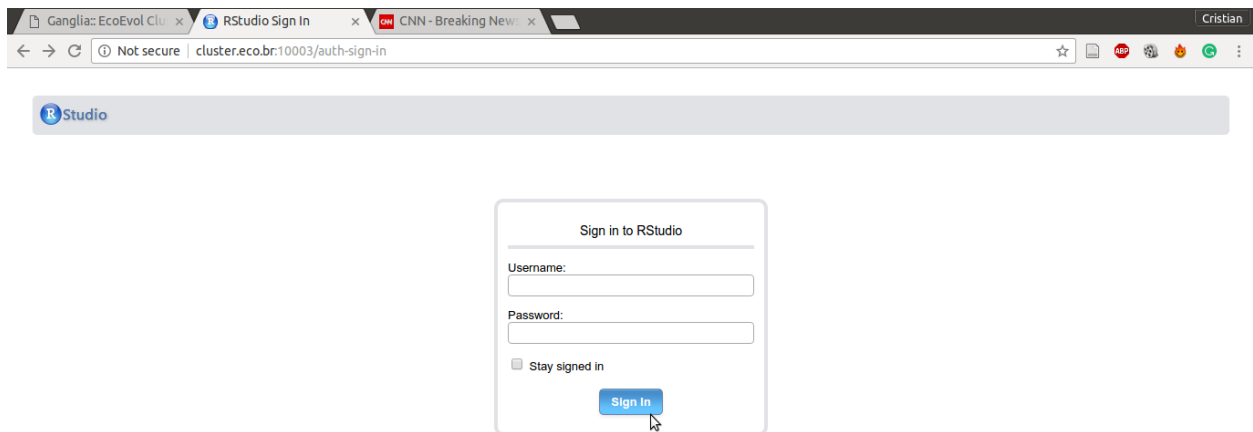


Figure 1: Página de login do RStudio.

Uma vez logado, o Rstudio irá mostrar seus arquivos pessoais no painel inferior direito da tela. Esta é a sua pasta de usuário (similar ao `C://`) e seus arquivos não poderão ser modificados por outros usuários. Se você está acostumado a trabalhar no Rstudio, você pode configurar o Rstudio como preferir e utilizar todos os recursos disponíveis, nenhuma modificação irá interferir na utilização pelos demais usuários (ex. você pode modificar o padrão de cores e o tamanho da fonte em *Tools -> Global Options -> Appearance*).

Uma configuração não obrigatória, mas que pode deixar seus arquivos melhor organizados, é a criação de uma pasta “Documents”. Você pode colocar seus arquivos e organizar sua área de trabalho como preferir.

Para que o Rstudio entre em uma pasta específica toda a vez que você logar, basta ir em *Tools -> Global Options*. Na janela que abrir, você terá a opção de definir uma nova pasta default. Originalmente a pasta é `~`. Isto significa a base do seu diretório de usuário. Você pode clicar em Browse para definir um novo diretório.

Assim como na versão desktop, o Rstudio permite a criação de projetos. Esta opção está acessível no canto superior direito da tela. Caso você crie um projeto, a pasta específica será carregada toda a vez que o usuário carregar o projeto. Caso tenha dúvidas no funcionamento do Rstudio, existem diversos tutoriais disponíveis na internet e na página do programa.

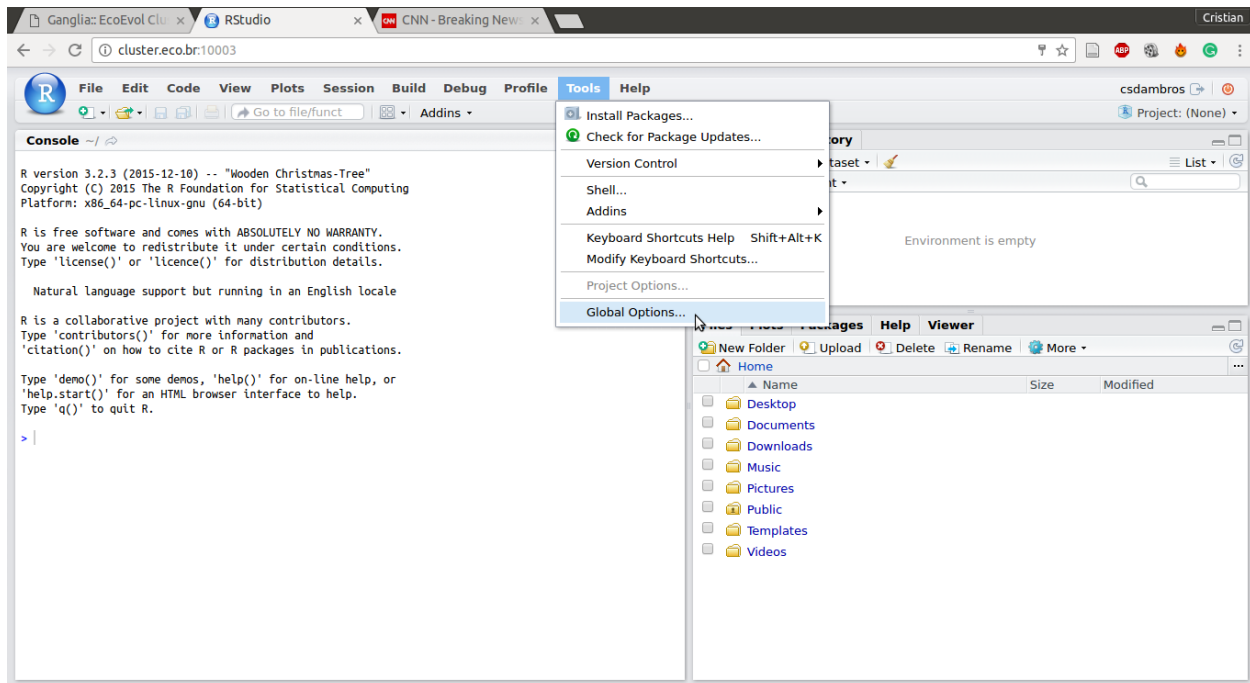


Figure 2: Configurando o RStudio.

2.1.1 Importando e exportando arquivos e pastas

Se você possui um projeto ou uma pasta do seu computador com planilhas, scripts e outros documentos relevantes, todos os arquivos podem ser facilmente importados para o servidor e posteriormente exportados para o seu computador. No painel inferior direito (seu diretório pessoal), há o botão *Upload*. Você pode realizar o upload de um único arquivo ou de uma pasta do seu computador. Para importar a pasta inteira, primeiro vá em seu computador e crie um arquivo .zip desta pasta. Então utilize o botão *Upload* para importar o .zip. O Rstudio irá automaticamente descompactar o arquivo .zip e você poderá trabalhar com os arquivos normalmente, como em seu computador pessoal. Quando estiver terminado de trabalhar no servidor, basta baixar novamente a pasta no formato .zip. Para baixar um arquivo ou uma pasta, clique na caixinha ao lado do nome do arquivo ou da pasta. Após, vá até a engrenagem na barra acima do seu diretório pessoal. Clique no botão *More*, e então *Export*. Para baixar todos os seus arquivos (diretório completo), selecione a caixa ao lado do botão *home*.

Caso você tenha criado ou importado um novo arquivo que não está aparecendo em seu diretório, tente atualizar a pasta clicando na seta acima do seu diretório pessoal (ao lado do botão *More*).

2.1.2 Enviando comandos para o servidor

Assim como na versão Desktop do RStudio, os comandos podem ser diretamente inseridos no console do R (painel inferior esquerdo) ou enviados de um script com o comando CTRL + ENTER.

Não é necessário que o usuário fique conectado ao servidor durante toda a execução de um comando. Se o usuário fechar a aba do navegador ou realizar o logout, o R continuará executando no servidor até que o processo seja terminado. Também é possível restaurar a sessão atual fazendo o login normalmente, desde que o login seja realizado no mesmo endereço/computador.

A sessão do RStudio somente será encerrada se o usuário solicitar o fechamento ou ficar sem utilizar o servidor após o término do processamento. Em qualquer situação, a área de trabalho e todos os arquivos ficarão

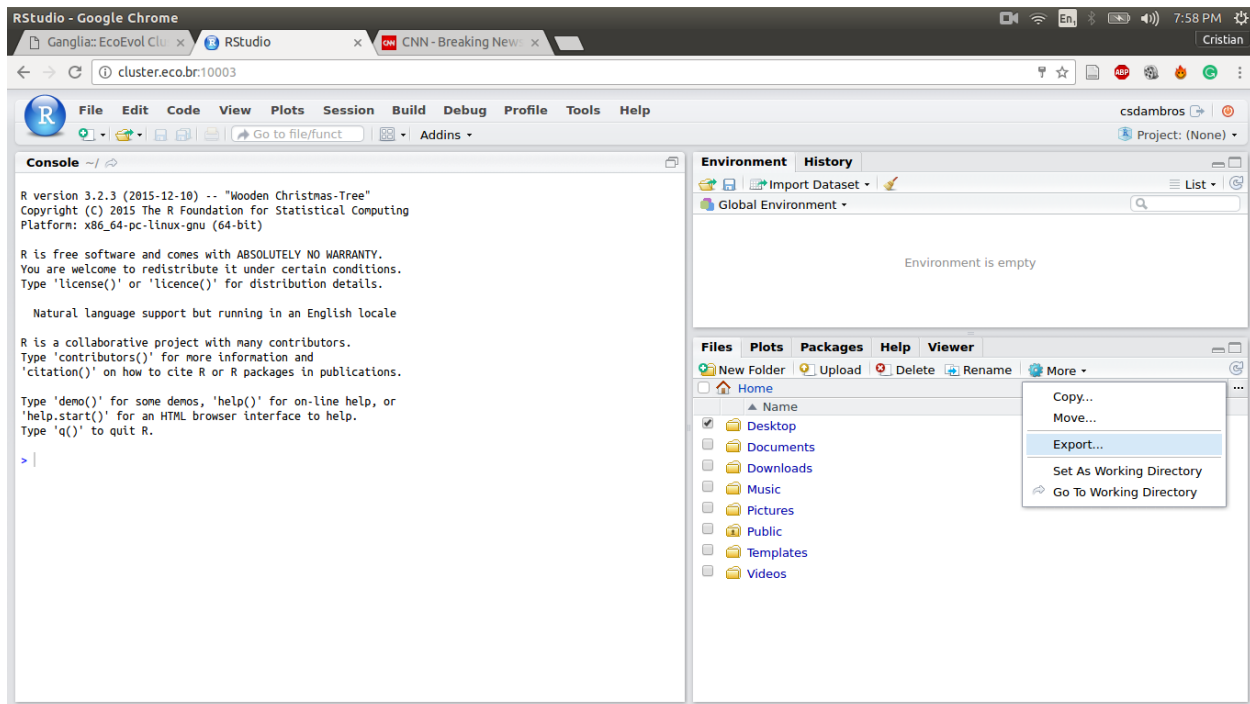


Figure 3: Exportando dados do RStudio.

disponíveis para o usuário. Ou seja, você não perderá seus dados ou a execução dos comandos a menos que solicite explicitamente o encerramento do processo.

2.1.3 Migrando para outro computador dentro do servidor

Quando um arquivo ou uma pasta é importada para o servidor, esta pasta é automaticamente compartilhada entre todos os computadores. Ou seja, você pode terminar um trabalho em um endereço (ex. Slave1), fazer o logout e continuar trabalhando com seus arquivos em outro endereço (ex. Slave2). Se você finalizar a sessão salvando a área de trabalho, o workspace do R com todos os objetos criados também estará disponível nos demais endereços. Este workspace será automaticamente carregado durante o próximo login. Para finalizar uma sessão, clique no botão de desligar no canto superior direito da tela ou vá em File -> Quit session. Se você realizar mudanças na área de trabalho, você será solicitado a salvar ou não a área de trabalho quando finalizar. Atenção, se você somente fechar a aba do seu navegador, o Rstudio não irá finalizar a sessão e sua área de trabalho (objetos dentro do R; painel superior direito) não estará automaticamente disponível em outros endereços, somente seus arquivos pessoais (incluindo scripts, planilhas, etc).

2.1.4 Instalação de pacotes

Você pode instalar normalmente pacotes do R no Rstudio. Todos os pacotes instalados em uma máquina (ex. Slave1) estarão automaticamente disponíveis nas demais (ex. Slave2) e também para todos os usuários do servidor. Alguns pacotes comumente utilizados, como rgdal, raster, rJava, rcpp, vegan e lme4 já estão instalados. Pacotes que permitem a paralelização (veja seção [Paralelização](#)) e modelos bayesianos (rjags e runjags) também já estão configurados.

2.2 SSH

Outra forma de acessar o servidor é através de um terminal (usuários windows podem utilizar o programa PuTTY). O terminal utiliza a linguagem Unix, nativa para linux e mac. Para logar no servidor, digite:

```
ssh -p porta NomeDeUsuário@cluster.eco.br
```

A porta define qual computador da cluster será acessado. As portas disponíveis são:

Slave1: 20001

Slave2: 20002

Slave3: 20003

Slave10: 20010

Slave11: 20011

Para instalar um programa no terminal contate o administrador. O programa screen está disponível. Este programa permite o logoff do usuário mantendo a execução do terminal e dos comandos enviados com posterior acesso. O programa cron/crontab para o agendamento de execuções também está disponível aos usuários.

2.3 RDP: Programas para Windows

Os computadores do servidor estão utilizando o sistema operacional linux (Ubuntu 16.04 LTS, *server edition*). O sistema instalado na maioria dos computadores do servidor não permite a utilização de programas desenvolvidos para Windows e também não possui interface gráfica (X11 ou DirectX; ex. tela com menu iniciar, etc.). Se você necessitar a instalação de programas Windows ou a utilização de uma interface gráfica, somente um dos computadores possui este suporte. Para este acesso, você precisará de um cliente para uma área de trabalho remota, ou seja, um programa que faça a conexão entre o seu computador e o servidor e lhe apresente a área de trabalho remota na tela de seu computador. Assim como os demais serviços, todos os usuários já possuem acesso por RDP.

Programas comumente utilizados:

Windows : Microsoft Remote Desktop Client (Conexão com Área de Trabalho Remota)

Linux: Remmina Remote Desktop Client

Mac: Remote Desktop Client for Mac 2

Android: Microsoft Remote Desktop Client

Você deve configurar a conexão em seu computador utilizando RDP (Remote Desktop Protocol). O endereço do servidor é

Slave11 <http://cluster.eco.br:11100>

Coloque seu nome de usuário e senha. Você pode definir a resolução da tela do servidor remoto como preferir. Utilize uma resolução baixa para conexões de internet lentas. A performance da conexão também pode ser melhorada com uma qualidade de cores baixa. As definições High color (16 bpp e 15bpp) são suficientes para a maioria dos programas.

O sistema operacional instalado neste servidor é linux Ubuntu 16.04 LTS 64 bits. Este sistema é baseado na mesma plataforma dos demais computadores (ubuntu), sendo que o terminal (ssh) e o Rstudio funcionam da mesma forma. Entretanto, o sistema utiliza um ambiente desktop chamado XFCE4, um dos ambientes mais leves existentes. Isto quer dizer que, apesar da aparência não ser tão refinada quanto em outras distribuições com aceleração gráfica e 3D (ex. Ubuntu Desktop), os recursos do computador estarão quase que totalmente direcionados a executar o seu programa e não a interface gráfica ou outros processos no background.

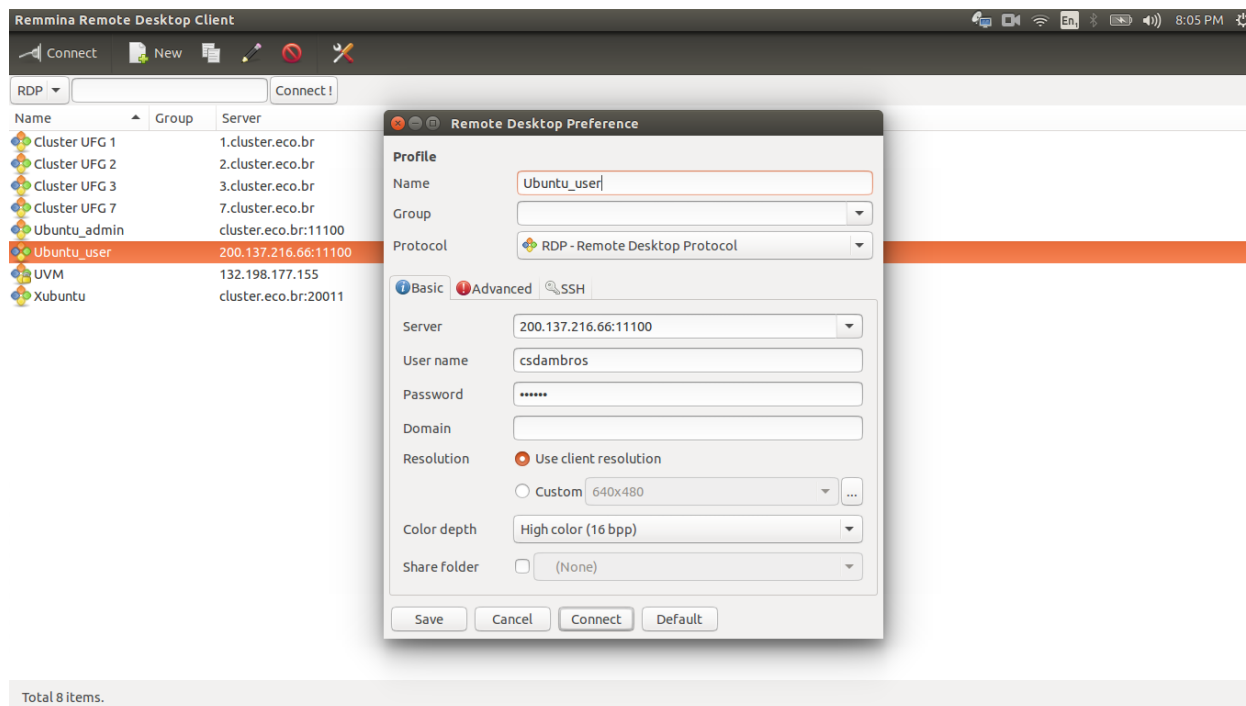


Figure 4: Configuração da conexão por RDP.

2.3.1 Wine

O programa Wine é um emulador do sistema Windows dentro do sistema operacional linux e permite a instalação e execução de programas .exe. Para instalar um programa, transfira o arquivo de instalação para seu diretório utilizando o próprio Rstudio ou uma conexão sFTP (veja seção sFTP). Se você colocar o executável dentro de uma pasta chamada Desktop, ele irá aparecer na área de trabalho do servidor, de outra forma, você terá que encontrar o arquivo no seu diretório base (~ /home/).

Após a transferência, acesse o Slave11 utilizando a conexão RDP. O programa pode ser executado normalmente com um duplo clique. O programa Wine irá automaticamente abrir o executável e emular o ambiente Windows para que você possa executar/instalar o programa.

Atenção, confira se o programa não possui distribuição para linux antes de instalar utilizando o Wine. Programas criados especificamente para linux irão ter um desempenho melhor sem a utilização do emulador. Se precisar de auxílio para a instalação de programas, contate o administrador do servidor.

2.4 FTP/sFTP

Contribuição Elisa Barreto

Uma forma conveniente de transferir arquivos grandes ou pastas inteiras para o servidor é através do protocolo sFTP. Este sistema permite que você abra o seu diretório do servidor como uma pasta no seu computador, permitindo copiar e colar, arrastar e deletar arquivos do servidor diretamente no seu computador. Você pode transferir seus arquivos diretamente para qualquer um dos Slaves do servidor e os arquivos já aparecerão automaticamente nos demais. O endereço para sFTP é o mesmo do endereço para SSH:

Slave1: <http://cluster.eco.br:20001>

Slave2: <http://cluster.eco.br:20002>

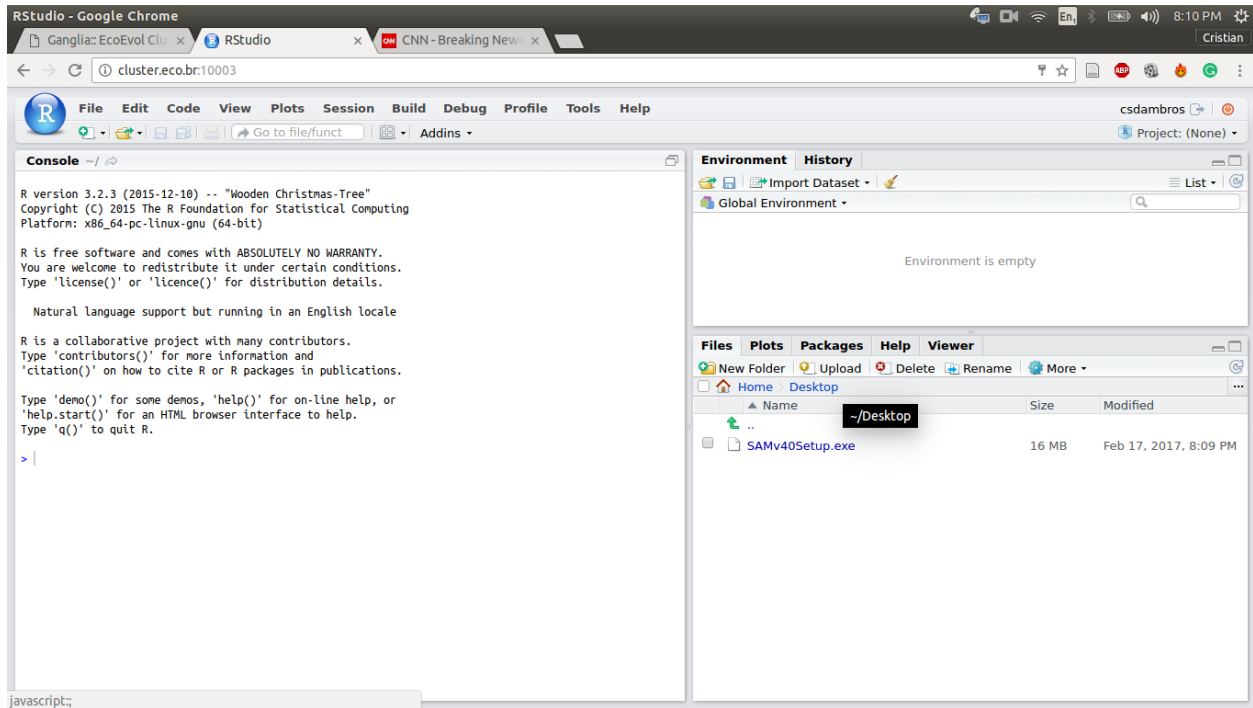


Figure 5: Arquivo importado pelo RStudio.

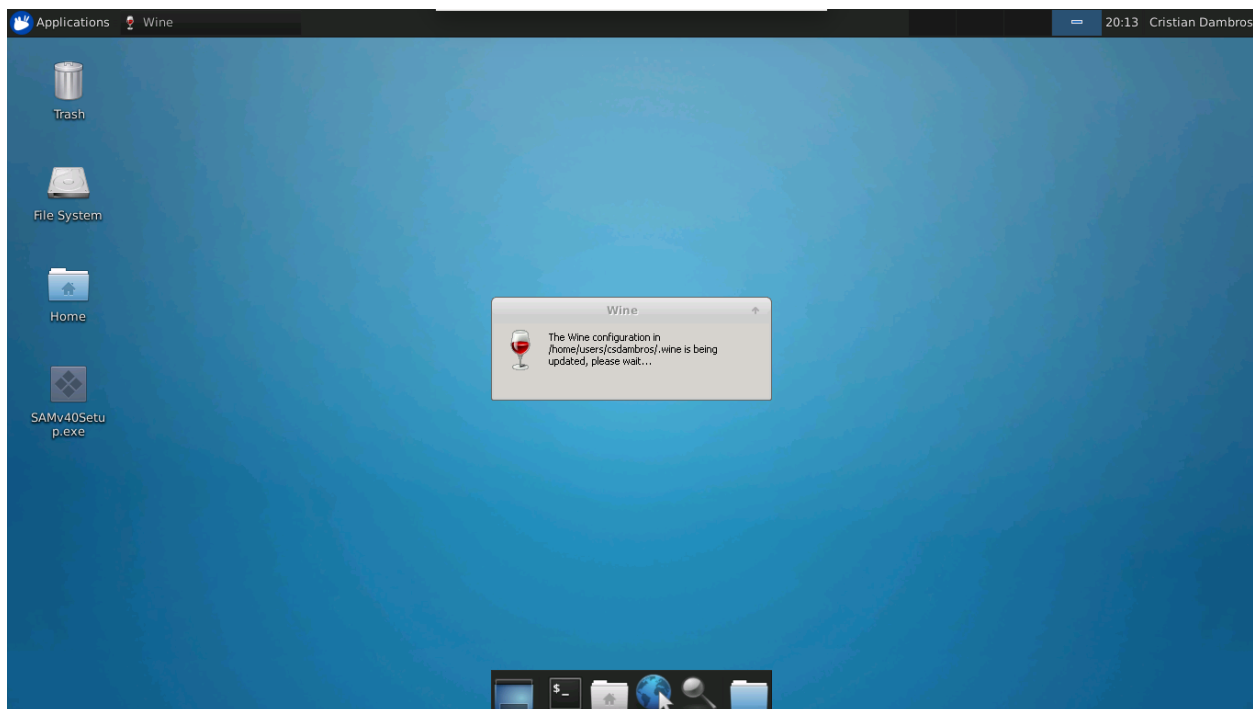


Figure 6: Mesmo arquivo aparece como executável no Slave 11. Após um clique duplo, o programa Wine começa a configuração inicial para prosseguir com a instalação.

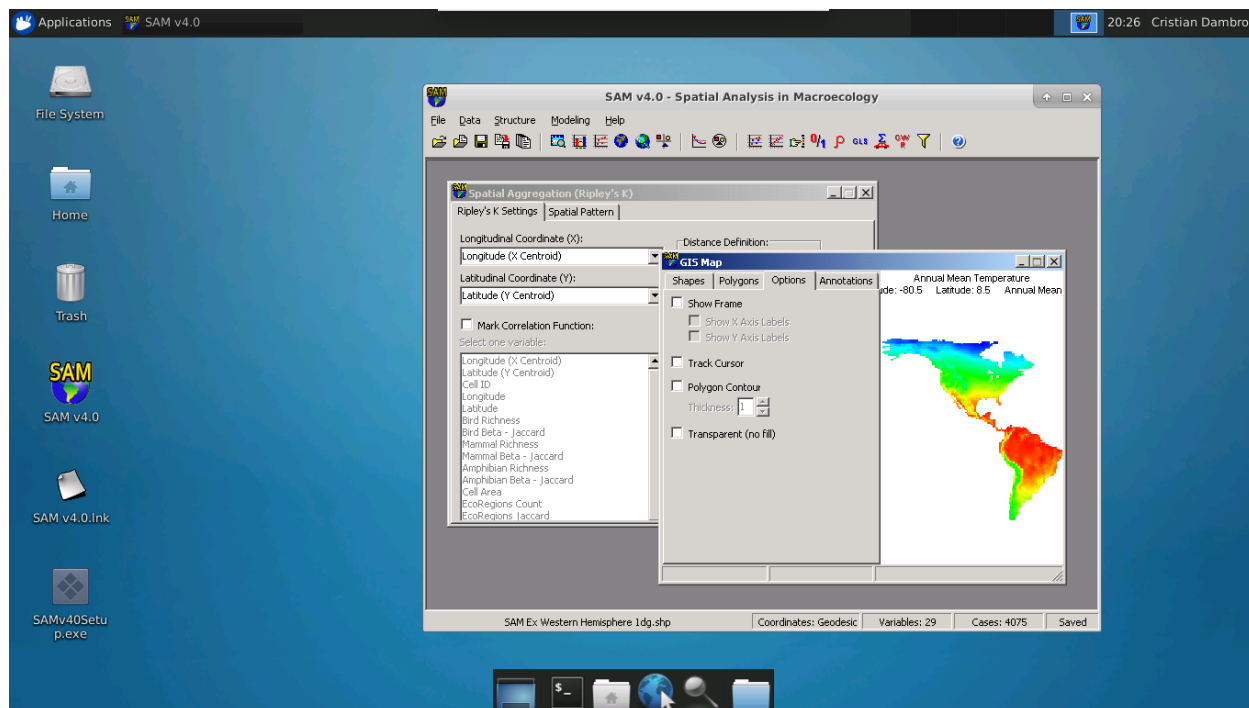


Figure 7: Programa de Windows (SAM) executando no servidor.

Slave3: <http://cluster.eco.br:20003>

Slave10: <http://cluster.eco.br:20010>

Slave11: <http://cluster.eco.br:20011>

A maioria dos sistemas operacionais já vem com programas instalados que permitem este tipo de conexão. Você também pode utilizar softwares gratuitos, como o programa FileZilla <https://filezilla-project.org/>.

2.4.1 Configurando o programa Filezilla

No primeiro uso é preciso configurar a conexão com a cluster. Para isso, instale e abra o FileZilla, selecione a aba *arquivo* e *gerenciador de sites*. Na página de gerenciar sites, selecione *novo site* e dê um nome para ele. Em seguida, preencha o campo *host* com o endereço da cluster (cluster.eco.br) e indique qualquer uma das portas (20001, 20002, 20003, 20010 ou 20011). Selecione o protocolo *SFTP – SSH File Transfer Protocol*. Preencha as informações de usuário e senha conforme o utilizado para acessar a cluster e clique em *OK*. A partir de agora, sempre que você quiser conectar seu computador com a cluster para transferir arquivos, basta abrir o FileZilla, ir em *gerenciar sites*, selecionar a cluster e clicar em conectar. No lado esquerdo da tela estarão indicados os diretórios e arquivos no seu computador, e no lado direito estão os da cluster.

3 Primeiro acesso

Geralmente o administrador já realiza este procedimento quando você cria sua conta. Se você já consegue acessar qualquer computador do servidor com sua credencial, ignore esta seção.

O nome de usuário e senha criados para cada usuário ficam em uma central. Esta central permite acesso através das credenciais a todos os computadores do servidor. Entretanto, quando uma credencial é criada, a central não autoriza o usuário a acessar os computadores automaticamente. Para isso, o usuário necessita

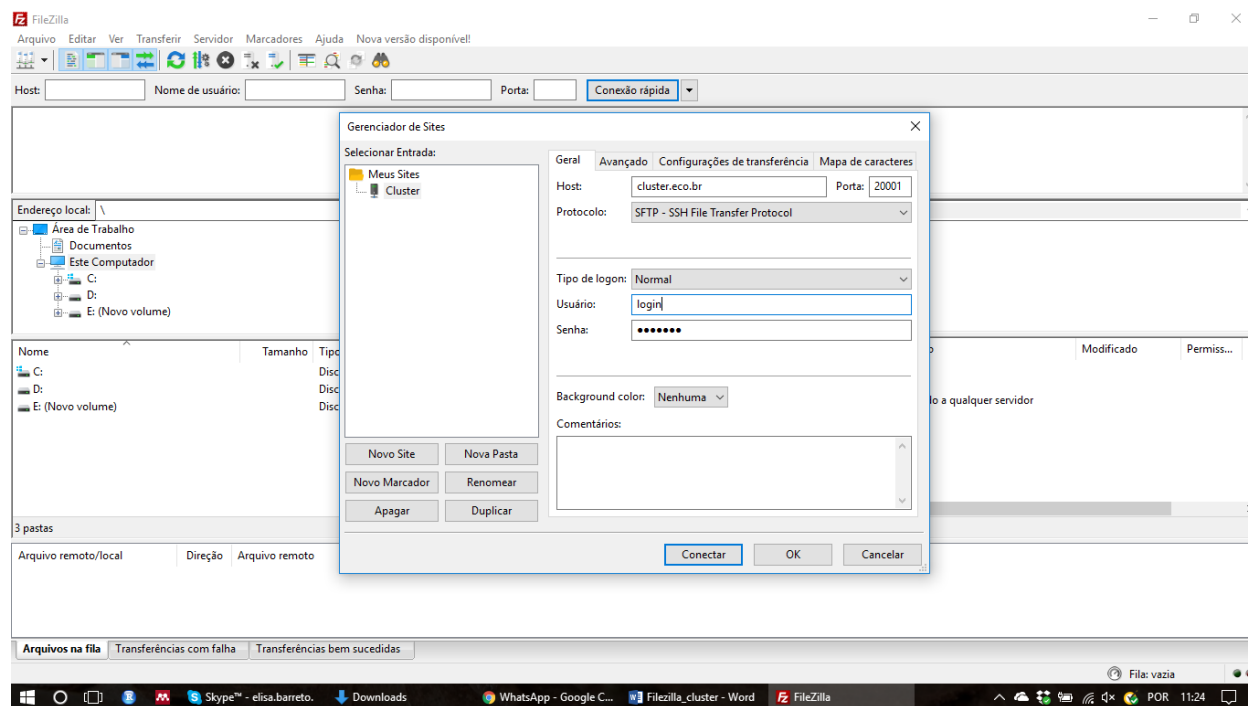


Figure 8: Tela de gerenciamento de sites do FileZilla.

logar em uma das máquinas utilizando SSH. Uma vez que o primeiro acesso é feito em qualquer uma das máquinas, o diretório do usuário é criado em todos os demais computadores, assim como diretórios no servidor (ex. o diretório equivalente ao “C://”).

Se o administrador ainda não fez o seu primeiro acesso, entre em um terminal de qualquer computador (usuários Windows podem utilizar o PuTTY ou pedir rapidamente um computador MAC ou linux emprestado). No terminal digite:

```
ssh -p 20001 NomeDeUsuário@cluster.eco.br
```

Sua senha será solicitada. Enquanto digita a senha, nada será mostrado na tela (assim ninguém saberá nem o tamanho de sua senha). Quando terminar de digitar a senha, pressione Enter. Aceite qualquer pergunta (ex. certificado) se for questionado.

Se seu login funcionou, você verá uma mensagem de entrada do Ubuntu. Se não funcionou, o servidor irá pedir sua senha novamente três vezes até falhar (você pode então tentar novamente). Se você foi capaz de logar, você já tem acesso a este e a todos os demais computadores do servidor utilizando sua credencial. Esta credencial funciona para todos os tipos de conexão (web, RDP, etc.). Após o primeiro acesso, você pode fechar o terminal e encerrar o processo (force a interrupção do processo se quiser).

4 Escolha da máquina

Todos os computadores possuem mais de 8 CPUs, podendo chegar a 16. Isto quer dizer que de uma maneira geral, múltiplos usuários poderão utilizar o R simultaneamente sem muita diferença na performance, a menos que algum usuário esteja realizando processos em paralelo.

Você pode conferir a carga e a capacidade de cada computador através deste link:

http://cluster.eco.br:30009/ganglia/?m=load_one&c=EcoEvol+Cluster&h=&p=2&tab=m&vn=&hide-hf=false&hc=4&p=2

A cor verde representando load significa que a máquina está com uso baixo (abaixo de 5%). Se algum dos nós estiver com a cor vermelha, significa que este nó está em manutenção ou desligado por algum outro motivo e não estará acessível por nenhum protocolo (mesmo que você consiga acessar, escolha outro computador, pois a máquina não está se comportando como esperado).

Através do link <http://cluster.eco.br:30009/ganglia/> você também pode checar diversas outras informações sobre a utilização de cada computador da cluster. Dependendo do objetivo do trabalho, você poderá necessitar de maior capacidade de processamento ou memória.

Aleatorizações e simulações simples geralmente necessitam maior capacidade de processamento (CPUs livres). Processos que necessitam a importação de grandes arquivos, como mapas e bancos de dados com mais de 1GB, geralmente utilizam mais memória RAM.

Escolha o computador que melhor atenda suas necessidades. Lembre-se sempre de salvar seus arquivos novamente em seu computador após o término do trabalho.

5 Paralelização: A grande vantagem de utilizar o servidor

ATENÇÃO: AO EXECUTAR ESTES COMANDOS, NÃO FORCE A FINALIZAÇÃO DO R. SE PARECER TRAVADO EM ALGUM MOMENTO OU ALGO INESPERADO ACONTECER DURANTE A PARELIZAÇÃO, AVISE O ADMINISTRADOR. TESTE SEU SCRIPT ANTES UTILIZANDO A FUNÇÃO `lapply` (VEJA **ABAIXO**)

Apesar da velocidade de processamento das CPUs do servidor ser maior do que a velocidade em computadores pessoais, a diferença neste quesito é pequena. Isto é, o tempo de processamento no servidor e em seu computador pessoal não será a princípio muito diferente (apesar de haver outras vantagens de utilizar um servidor remoto). Tanto no servidor quanto em um computador pessoal, o R iniciará utilizando somente um processador. Entretanto, os processos podem ser paralelizados, fazendo com que o R seja multiplicado em múltiplas CPUs simultaneamente, reduzindo significativamente o tempo para que um output seja obtido. Por exemplo, se um comando do R for iniciado em duas CPUs, a capacidade de processamento será dobrada e o tempo para obter um resultado pode ser reduzido em até 50%. Como existem mais de 10 CPUs em cada computador do servidor, um comando que demoraria mais de uma semana para ser executado pode demorar menos de um dia.

5.1 Quando meu script pode ser paralelizado?

De uma maneira geral, quando você utiliza um loop repetido. Por exemplo, o comando `for (i in 1:100) { resu[i] <- sample (10) }` gera 100 combinações aleatórias dos valores de 1 a 10. Como cada combinação é independente da outra, este processo pode ser dividido em múltiplos processadores. Por exemplo, podemos solicitar que 4 processadores gerem 25 combinações cada. Se você utiliza a função `lapply` em algum local do seu script, esta função pode ser facilmente paralelizada, pois os pacotes que fazem a paralelização possuem um análogo que funciona com a mesma sintaxe. Ou seja, haverá uma diferença muito pequena em como você escreve o seu script normalmente ou paralelizando.

5.2 Exemplo das funções `for` e `lapply`

A função `for` serve para executar um mesmo comando múltiplas vezes. Esta função pode ser executada sequencialmente (o resultado de uma execução interfere na próxima execução) ou de forma independente (uma execução não interfere na próxima). Vamos ver aqui somente este último exemplo.

O comando a seguir aleatoriza os números de 1 a 10 cinco vezes

```
resu <- {}

for(i in 1:5){

resu[[i]] <- sample (10)

}
```

Os dados estão sendo armazenados em uma lista:

```
print (resu)

## [[1]]
## [1] 6 4 3 8 5 1 9 7 10 2
##
## [[2]]
## [1] 9 4 8 5 3 2 7 6 1 10
##
## [[3]]
## [1] 5 6 1 10 7 8 9 4 3 2
##
## [[4]]
## [1] 6 4 2 7 9 1 10 3 8 5
##
## [[5]]
## [1] 8 7 3 9 4 2 10 6 5 1
```

Também podemos realizar o mesmo procedimento utilizando a função *lapply*:

```
# Cria um vetor repetindo o número 10 cinco vezes
input <- rep(10,5)
input

## [1] 10 10 10 10 10

# Para cada valor no input, aplica a função sample
lapply(input,sample)

## [[1]]
## [1] 3 1 4 7 10 8 6 2 9 5
##
## [[2]]
## [1] 1 6 2 3 7 9 10 5 4 8
##
## [[3]]
## [1] 3 9 6 10 2 5 1 8 7 4
##
## [[4]]
## [1] 6 7 5 8 9 2 1 10 4 3
##
## [[5]]
## [1] 4 2 5 10 3 7 1 6 9 8

# funciona da mesma forma se o input for uma lista
input <- as.list(rep(10,5))
input
```

5.2 Exemplo das funções *for* e *lapply*

```
## [[1]]
## [1] 10
##
## [[2]]
## [1] 10
##
## [[3]]
## [1] 10
##
## [[4]]
## [1] 10
##
## [[5]]
## [1] 10

# Para cada elemento da lista, retorna sample (elemento)
lapply(input,sample)
```

```
## [[1]]
## [1] 10  3  5  7  8  1  2  4  6  9
##
## [[2]]
## [1]  9  8  1  7  3  5 10  2  6  4
##
## [[3]]
## [1]  9  1  3  2  8  4  7  5 10  6
##
## [[4]]
## [1]  2  7  5  3  1  8  4  6 10  9
##
## [[5]]
## [1]  5  9 10  8  3  6  2  4  7  1
```

Apesar de servirem para o mesmo propósito, existem algumas vantagens em utilizar a função `lapply` ao invés do `for`. Por exemplo, `lapply` é mais rápido do que `for` (as vezes muito mais), mesmo em aplicações simples que não requerem paralelização. A função `lapply` possui análogos em diferentes pacotes bastante utilizados do R especialmente desenhados para trabalhar com grandes bancos de dados ou quando é preciso alta capacidade de processamento, como os pacotes `plyr` e `snow`. Por fim, utilizando esta função seu código fica mais limpo, pois você não precisa especificar a forma do input e do output (o output será sempre uma lista).

O exemplo a seguir mostra uma aplicação um pouco mais útil da função `lapply`. Este exemplo também será utilizado para paralelizar um comando (veja a [próxima seção](#))

```
# Carrega o pacote ape
library(ape)

# Cria uma árvore filogenética aleatória com 50 espécies
phy <- rtree(n=50)

# Cria 100 árvores filogenéticas aleatórias
rphy <- lapply(rep(50,100),rtree)

# Calcula a distância filogenética entre todos os pares de espécies de cada árvore
dphy <- lapply(rphy,cophenetic)

# Calcula a distância média pareada entre todas as espécies em cada árvore
```

```
mpdphy <- lapply (dphy,mean)
```

Note que o argumento principal que função está utilizando em cada loop é fornecido como um elemento no input da função `lapply`. Por exemplo, para calcular a distância filogenética, 100 árvores filogenéticas foram fornecidas em uma lista. A função `cophenetic` foi então aplicada para cada elemento da lista. O output foi então uma nova lista, onde cada elemento é uma matriz triangular de distâncias filogenéticas. Novamente a função `lapply` foi utilizada para calcular a média de cada uma das matrizes. Árvores filogenéticas já são geralmente organizadas em listas, assim como diversos outros objetos no R, como Rasteres, Mapas, etc., o que facilita bastante a utilização desta função em outras aplicações.

5.3 Paralelizando em um ou múltiplos computadores

Os computadores do servidor já possuem programas e pacotes do R que permitem a paralelização dos serviços tanto dentro de um computador ou através de todos os computadores do servidor. Por default, o servidor já irá paralelizar o seu comando utilizando todos os computadores (são 68 CPUs disponíveis). Entretanto para evitar inutilizar completamente o sistema para os demais usuários, nenhum usuário deve utilizar mais de 15 processadores sem a expressa autorização de um dos administradores.

Como a paralelização irá envolver múltiplos computadores, você deverá carregar a função `passSSHid`:

```
source("https://raw.githubusercontent.com/csdambros/R-functions/master/passSSHid.R")
```

Esta função irá lhe autorizar a compartilhar os processos entre todos os computadores do servidor. A função não necessita de nenhum argumento. Sua senha pessoal será solicitada (a mesma senha que você utilizou para fazer o login).

```
passSSHid()
```

Agora ao que interessa:

Para paralelizar o código na cluster, será utilizado o pacote `snow`

```
library (snow)
```

Após carregar o pacote, é preciso informar quantos processadores serão utilizados criando uma cluster virtual. Essa cluster será chamada `c1` (escolha o nome que preferir). O exemplo a seguir utiliza 10 CPUs

```
c1 <- makeMPIcluster(count = 10)
```

ATENÇÃO: LEMBRE-SE DE ENCERRAR A CLUSTER VIRTUAL QUANDO TERMINAR SEU TRABALHO (VEJA ABAIXO COMO)!!!

Pronto! Se a função funcionou e você recebeu a mensagem `10 slaves are spawned successfully`, as CPUs já estão reservadas aguardando seu próximo comando. Ignore qualquer aviso ou mensagem do sistema, se você recebeu a mensagem, está funcionando.

Para paralelizar de fato, precisamos agora utilizar a função `clusterApply`. Esta função funciona exatamente como a função `lapply`, só precisamos especificar o nome da cluster virtual que foi criada (`c1` neste exemplo). Para o exemplo das árvores filogenéticas:

```
library(ape)
```

```
# Cria 100 árvores filogenéticas aleatórias
rphy <- clusterApply(c1,rep(50,100),rtree)
```

```
# Calcula a distância filogenética entre todos os pares de espécies de cada árvore
dphy <- clusterApply(c1,rphy,cophenetic)
```

```
# Calcula a distância média pareada entre todas as espécies em cada árvore
mpdphy <- clusterApply (cl,dphy,mean)
```

O output da função é o mesmo obtido na função lapply. Entretanto, o processamento foi dividido em 10 processadores. Este procedimento pode ser realizado para aumentar a capacidade de processamento em até ~10x dentro da cluster ou ser realizado em seu próprio computador pessoal (você pode utilizar a função `makeSOCKcluster` em seu computador ou instalar o programa MPI e o pacote Rmpi. Não utilize o modo socket no servidor!).

Por fim, podemos encerrar a cluster virtual.

```
# Terminar com a cluster virtual

stopCluster(cl)
```

ATENÇÃO: LEMBRE-SE DE ENCERRAR A CLUSTER VIRTUAL QUANDO TERMINAR SEU TRABALHO!!!

IMPORTANTE: Não esqueça de encerrar a cluster criada quando terminar de utilizar. Mesmo que você termine o seu trabalho, o servidor continuará utilizando as CPUs solicitadas até que a cluster virtual seja encerrada utilizando o comando acima.

5.4 Começando e encerrando a paralelização automaticamente (altamente recomendado)

Para evitar que os processadores fiquem reservados na cluster após o término do processamento, é altamente recomendado e conveniente organizar o seu script da seguinte forma:

```
{ # Coloque chaves para começar o comando

cl <- makeMPIcluster(count = 10)

# SEUS COMANDOS AQUI
# EX.
rphy <- clusterApply(cl,rep(50,100),rtree)

# Terminar com a cluster virtual

stopCluster(cl)

} # Coloque chaves para terminar o comando

# Envie CTRL + Enter na primeira chave para executar todas as linhas em sequência
```

Colocando tudo entre chaves permite que você envie a sequência de comandos incluindo a criação e o fechamento da cluster virtual. Isto quer dizer que quando o processamento acabar, a cluster virtual será automaticamente fechada, poupando recursos da cluster. Não se preocupe, todos os objetos criados no script continuarão salvos e disponíveis após o encerramento da cluster virtual.

Outra forma de fazer a abertura e o fechamento automático é colocando a paralelização dentro de uma função. Entretanto, dentro de funções, é também necessário o envio dos objetos criados para as demais CPUs. Isto inclui argumentos da função e outros objetos criados entre o cabeçalho da função e o processo paralelo. A seguir segue um exemplo de função que poderia ser utilizada:

```
Fun1 <- function (argumento1, argumento2, argumento3, nnodes = 10){
```

```
require(snow)

argumento4 <- sum (argumento1,argumento2) # Cria argumento 4, se for usar, é necessário enviar

cl <- makeMPIcluster (nnodes)

# Envia para todos os nós os argumentos 3 e 4
#snow::clusterExport(cl, list = c("argumento3","argumento4"),envir = environment() )

# Alternativamente, mande todos os objetos
#snow::clusterExport(cl, list = ls(envir = environment() ),envir = environment() )

# Os argumentos 3 e 4 são utilizados pela função dentro do clusterApply
resu <- clusterApply (cl, 1:10, function(x) sum( argumento3, argumento4) )

stopCluster(cl)

return (resu)

}
```

Fun1 (1,2,3)

Este processo também facilita o monitoramento do processo pelo usuário. Por exemplo, se você enviou comandos para 25 processadores, você poderá ver um aumento de uso de 25 CPUs na cluster em <http://cluster.eco.br:30009/ganglia/>. Quando o seu processo terminar, este mesmo link mostrará a redução de 25 CPUs utilizadas.

5.5 Quando é vantagem paralelizar e como otimizar o processo

Utilizando o exemplo acima, talvez você tenha notado que a função `clusterApply` levou mais tempo para executar do que a função `lapply`. Isto ocorre porquê os comandos que estamos executando são extremamente rápidos e um processador único é suficientemente rápido para realizar o procedimento em todas as árvores filogenéticas em menos de um segundo. Já a função `clusterApply` necessita enviar o comando para múltiplas CPUs e recolher o resultado, o que demanda alguns segundos. Estes segundos que a função `clusterApply` utiliza serão multiplicados se a função necessitar reenviar os comandos para as CPUs durante o processamento. É isto que ocorre quanto o número de loops (100 árvores no exemplo) é executado em 10 processadores. São necessários 10 envios. Para reduzir este tempo, o ideal é que o número de envios não seja muito superior ao número de CPUs reservada para o trabalho. Ou seja, a forma mais eficiente de paralelizar seria dividir a lista com as árvores filogenéticas em 10 grupos (uma lista com 10 elementos) e enviar cada grupo para processamento.

Na cluster atual, o tempo médio de envio/processamento extra é de 1.16 segundos. Se você possui uma função que necessita ser executada até 1000 vezes (equivalente a um `for (i in 1:1000){...}`), você irá gastar em torno de 2 minutos para o envio e recebimento de todo o comando dos processadores em adição ao tempo que a função irá demorar para executar. Se a função demora um minuto para executar um único loop, ela demoraria em torno de 1000 minutos (=16.67 horas) para rodar as 10000 repetições utilizando somente uma CPU. Em contraste, se o trabalho for paralelizado em 10 CPUs, a função irá consumir 100 minutos (=1.67 horas), mais 2 minutos para o envio e recebimento do processo.

Quando o tempo de processamento de uma função é alto, paralelizar o comando é especialmente vantajoso para o usuário. Utilizando 15 CPUs, um processo que demoraria um mês para ser executado, pode vir a demorar em torno de 2 dias!

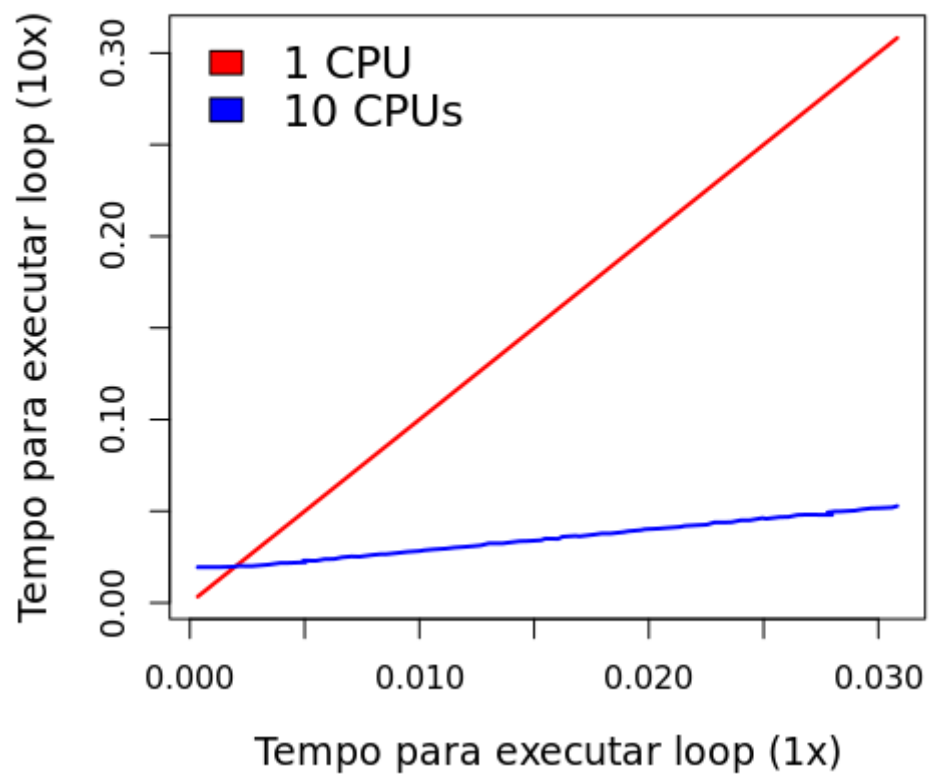


Figure 9: Tempo para execu     de um loop no R em minutos. Se a fun     executa rapidamente em um   nico processador, o tempo de envio far   o processo em paralelo (azul) ser mais demorado. Desta forma, o n  mero de envios deve ser reduzido ao m  ximo para n  o replicar este tempo extra. Quando o tempo de execu     da fun     aumenta, paralelizar se torna especialmente vantajoso. Note que a pr  pria fun     pode conter loops ou outros comandos que demandam tempo para executar.

ATENÇÃO: LEMBRE-SE DE ENCERRAR A CLUSTER VIRTUAL QUANDO TERMINAR SEU TRABALHO!!!

Se você quer mais exemplos de aplicações ecológicas que requerem maior capacidade de processamento, veja o manual **Parallel Computing in R: Computadores de alto desempenho permitem a utilização de novos modelos ecológicos**

Caso tenha alguma dúvida, contate os administradores (contato na primeira página deste manual).