capitulo R e R Studio

# R e RStudio

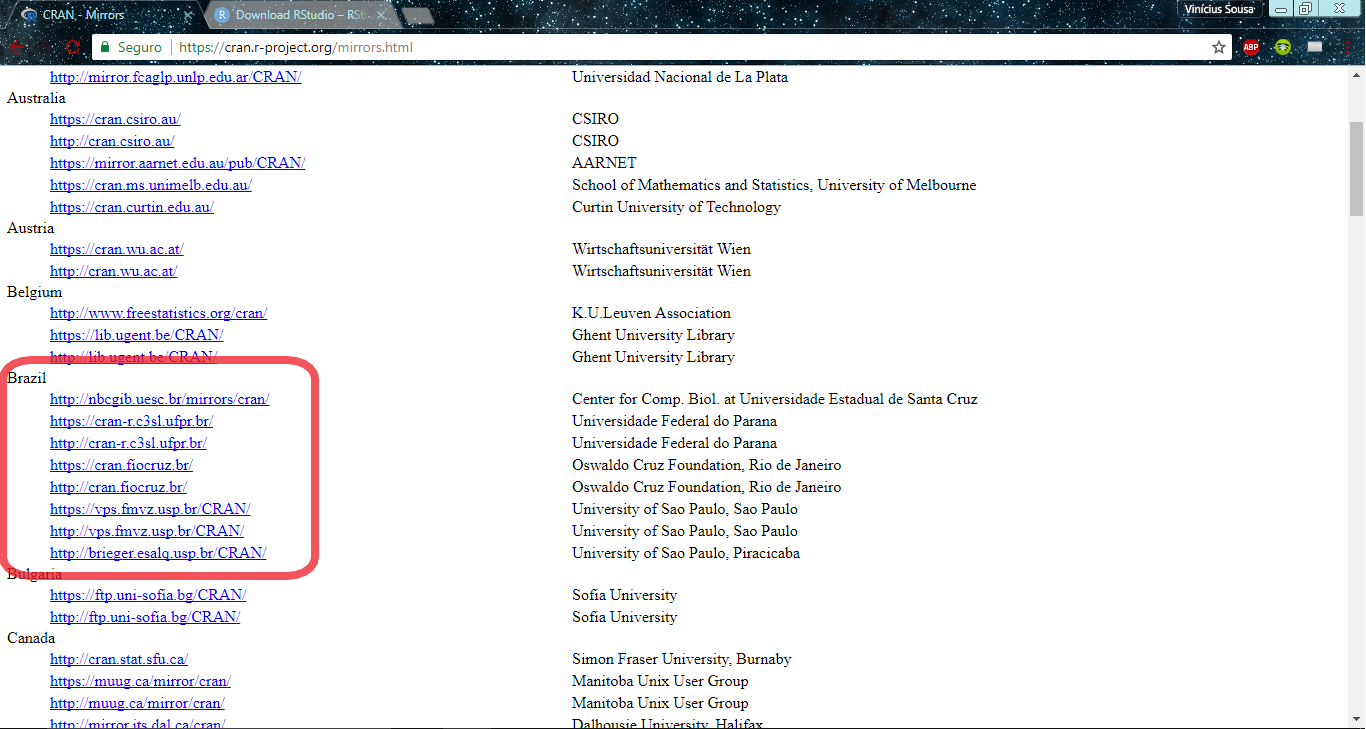
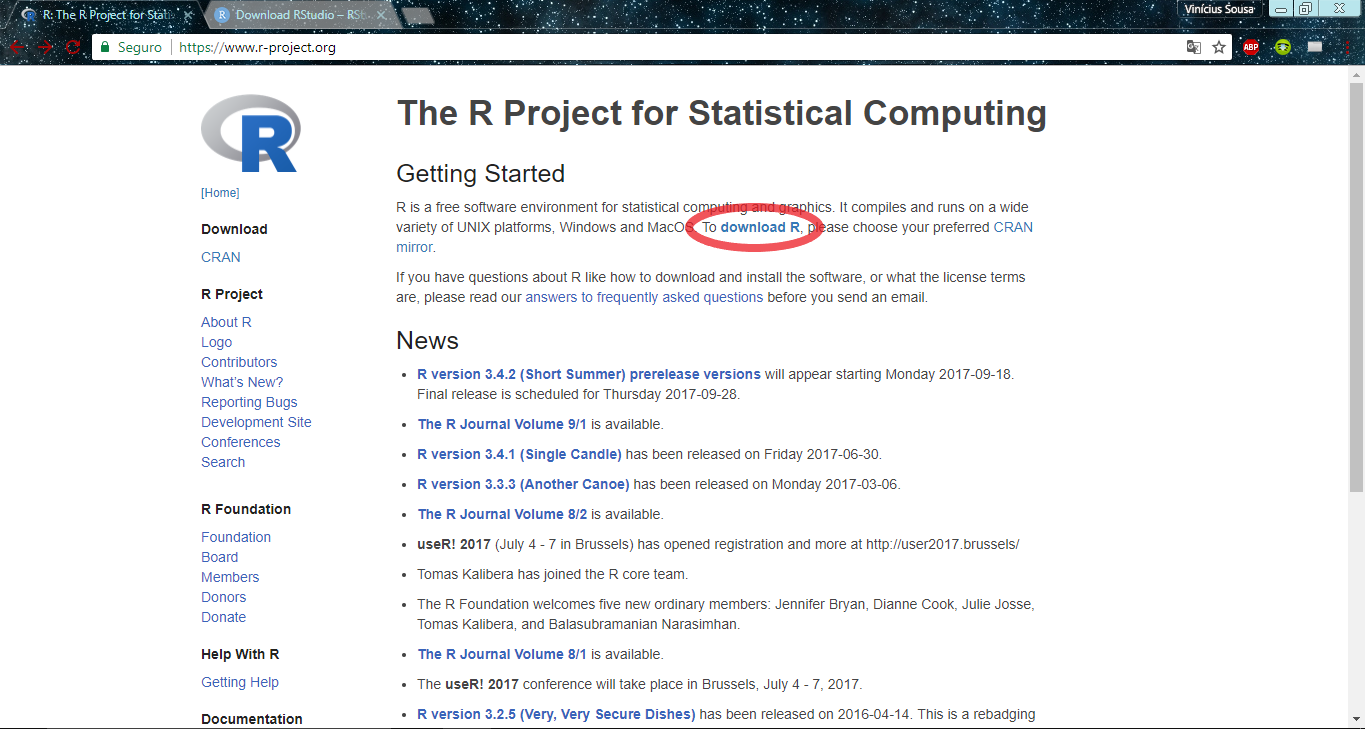
Antes de introduzir o leitor ao mundo do R gostaria de parabenizá-lo por aceitar o desafio de aprender um linguagem de programação que o ajudará a realizar seus trabalhos. Então, parabéns!

## O que são

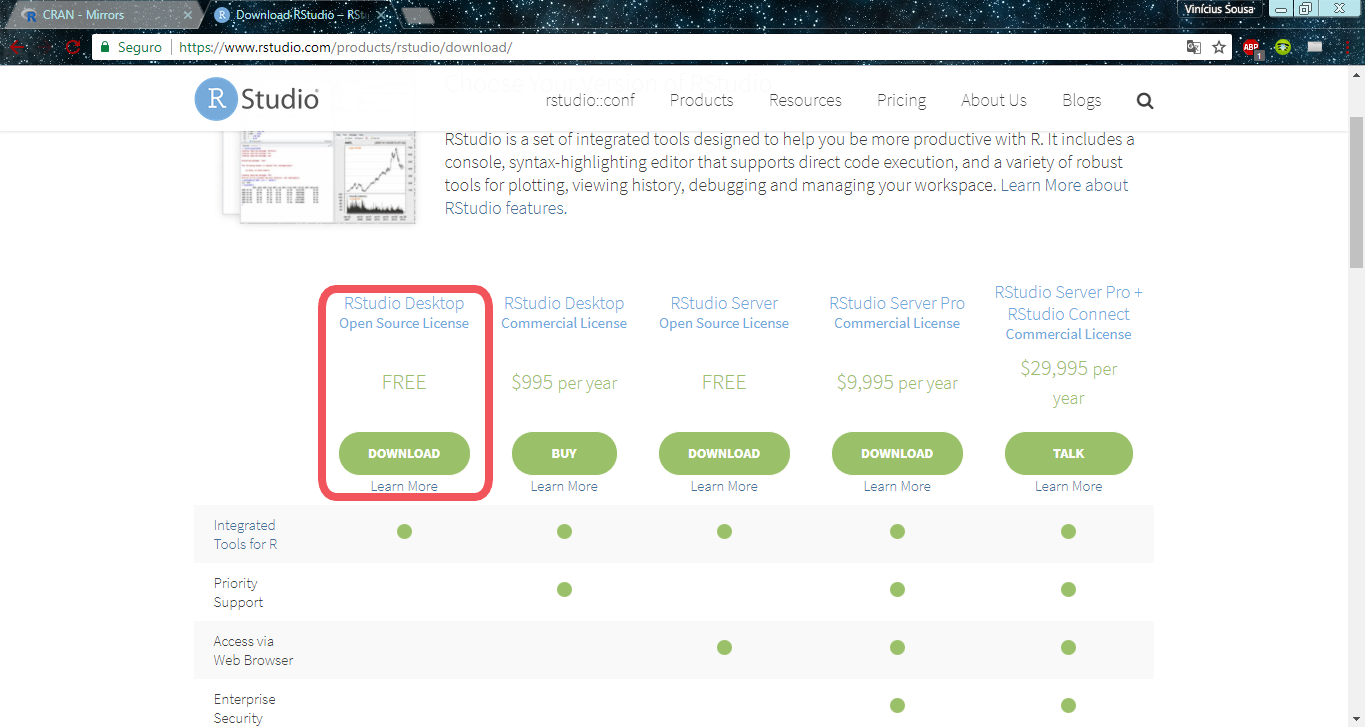
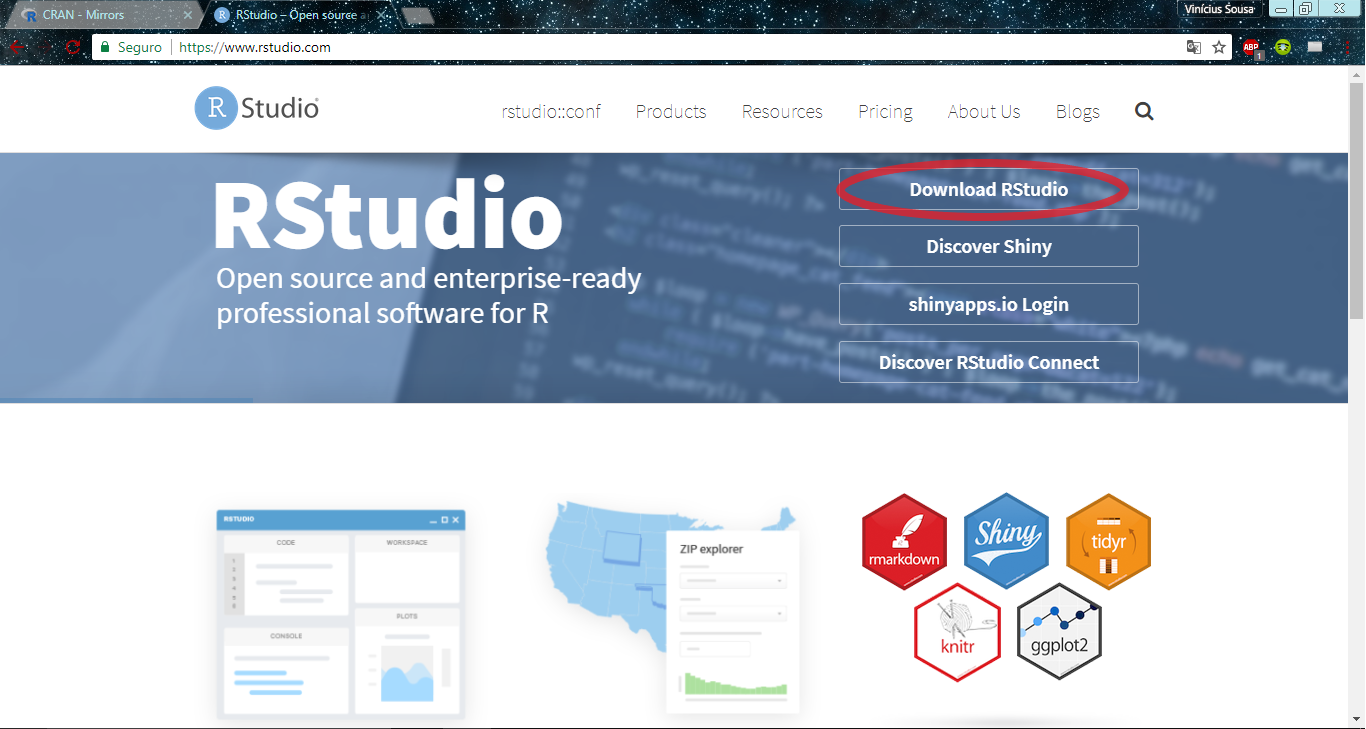
R é uma linguagem de programação estatística baseada em comandos de linha, de modo que é dado ao usuário uma liberdade e flexilibidade na hora de fazer as análises. O RStudio é um ambiente de programação que torna mais fácil e eficiente a utilização do R. @matloff2011 apresenta a analogia de Greg Snow para exemplificar a lógica do R. Podemos pensar no mundo de análise dados como sendo uma cidade e as possíveis análises são diferentes locais nesta nossa cidade hipotética. Os principais meios de locomoção que pensamos são transporte público (ônibus, metrô, bondinho etc.) e transporte privado (em especial carros). Analisando as possilidades sabemos que andar de ônibus é mais fácil, basta pagar a passagem saber em qual ponto subir e em qual ponto descer. Já para poder utilizar o carro há um custo de aprender a dirigir o carro, ter um conhecimento de leis, e significados de placas, ter uma espécie de mapa mental para saber onde está indo (além do financeiro hehe) e etc. Mas e afinal de contas qual é a vantagem de se utilizar o transporte privado no lugar do público? É o simples fato de ser mais confortável e possibilitar você conseguir chegar em locais que o transporte público não te levaria sem uma boa caminhada. Destaco que a analogia foi feita antes de surgir aplicativos como UBER, e na minha interpretação andar de UBER seria como pagar alguém para fazer a análise por você.

O ponto é que utilizar pacotes estatíscos de botões (como SPSS, Gretl, Stata e afins) é como se você estivesse utilizando o transporte público, ao passo que ao introduzir ferramentas de programação (como o R e Python fazem, por exemplo) abre inúmeras possibilidades que são limitadas apenas pelo conhecimento do usuário. Deste modo você consegue chegar em lugares novos e chegar mais rápido em lugares que de outra maneira seria mais demorado.

Dito isso, o primeiro passo para entrarmos no mundo é baixar e instalar p O eo RStudio. Para baixar o R acesse o site do R Project (<https://www.r-project.org/>), clique em *download R* e em seguida escolha um dos *mirrors* brasileiros, a figura @ref(fig:baixar-r) sintetizaos passos. O próximo passo é apenas executar o arquivo baixado.

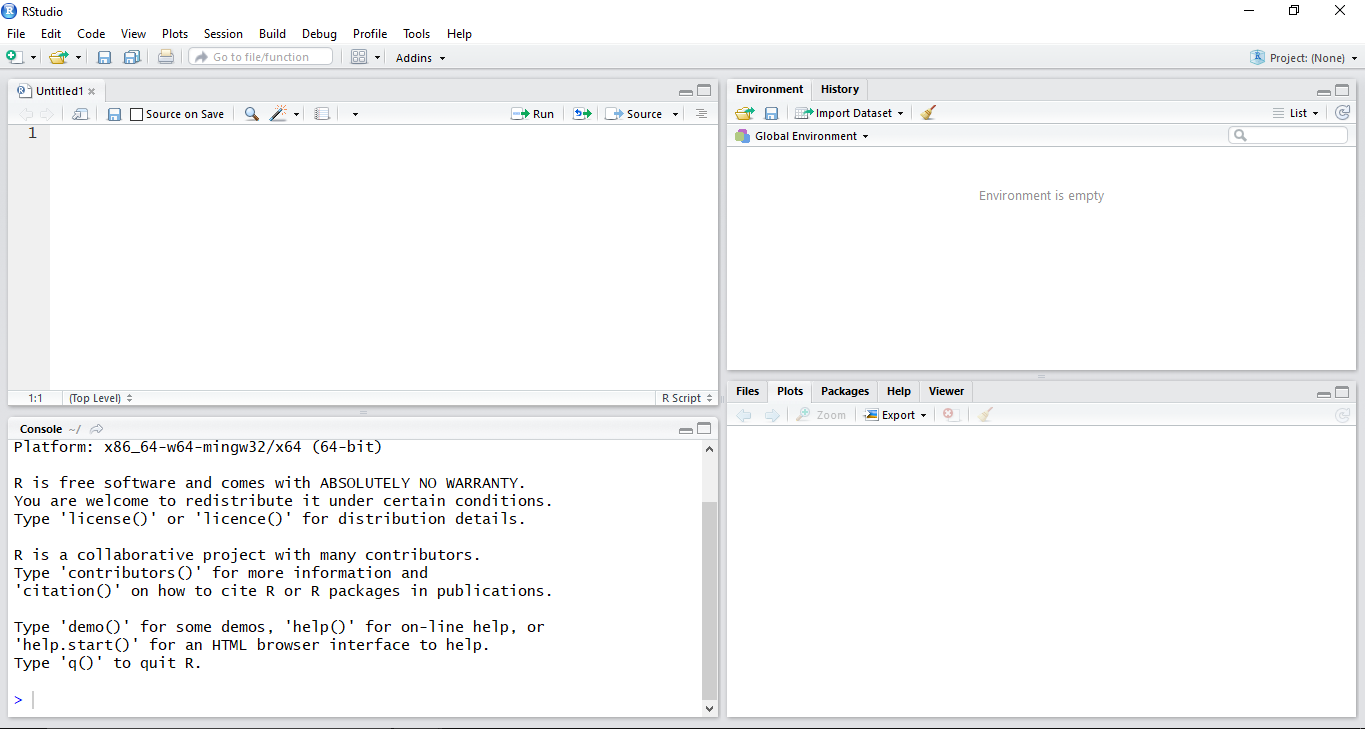


Já o RStudio, como dito anteriormente, é um ambiente que torna a utilização da linguagem R mais agradavél e possibilita uma organização muito boa da análise que está sendo feita. Para baixar o RStudio acesse o site do RStudio (<https://www.rstudio.com/>), clique no icone *Download RSstudio*, em seguida clique em *Desktop Open Source License* e por fim basta executar o arquivo baixado, a figura @ref(fig:baixar-rstudio) sintetiza os passos.



É importante destacar que **a ordem de instalação deve ser respeitada,isto é, primeiro instalr o R e depois o RStudio**.

## *Layout* do RStudio

Depois de instalar o R e RStudio abra o RStudio, pois como comentado anteriorment o RStudio é um ambiente que torna mais eficiente a programação em R, e desse modo utilizaremos ele como interface para usar o R. Ao abrir o RStudio você deve se deparar com o *layout* apresentado na figura @ref(fig:layout-rstudio). 

Esse *layout* apresenta 4 janelas principais para se usar o R no RStudio. De maneira sucinta:

1. A janela superior esquerda é o *script* que, como veremos em seguida, é onde o código é escrito para ser executado;
2. A janela superior direita, chamada de *environment*, e é onde encontram-se as variáveis, dados, e objetos;
3. A janela inferior esquerda é o *console* onde são exibidos os resultados que não são figuras, como tabelas, regressões, contas etc;
4. E por fim, a janela do canto inferior direito é onde são exibidos os plots produzidos.

## *Scripts*

Outra grande vantagem de se utilizar linguagens baseadas em comandos de linha são os *scripts* que são onde ficam armazenados os comandos que devem ser executados, no caso do R os *scripts* são salvos na extensão **.R**. Os *scripts*, junto com variáveis e funções, possibilitam evitar a repetição mecânica dos comandos, um exemplo trivial para ilustrar o ponto é o cálculo da índice de massa corporal (IMC), podemos calcular comuma calculadora através da fórmula

. Para calcular o IMC dos membros de sua família você pode anotar o peso e altura de cada um e calcular individualmente, ou então você pode escrever o seguinte código:

> peso <- 65  
> altura <- 1.75  
>   
> IMC <- peso/altura^2  
> IMC  
# [1] 21.22449

De modo que o que você precisa fazer é apenas mudar os valores do peso altura e executar o código novamente. Dois comandos são importantes você ter em mente sobre *scripts*: (i) Para abrir um novo *script* tem se usar o atalho ctrl+shift+n; e (ii) para salvar um *script* baste clicar no icone de disquete que aparece na aba embaixo de *untitled1* na figura @ref(fig:layout-rstudio).

Podemos dizer que os scripts substituiem os mouse e os botões por uma tela onde o usuário através dos comandos realiza suas análises da maneira mais livre possível.

## Funcionamento do Código

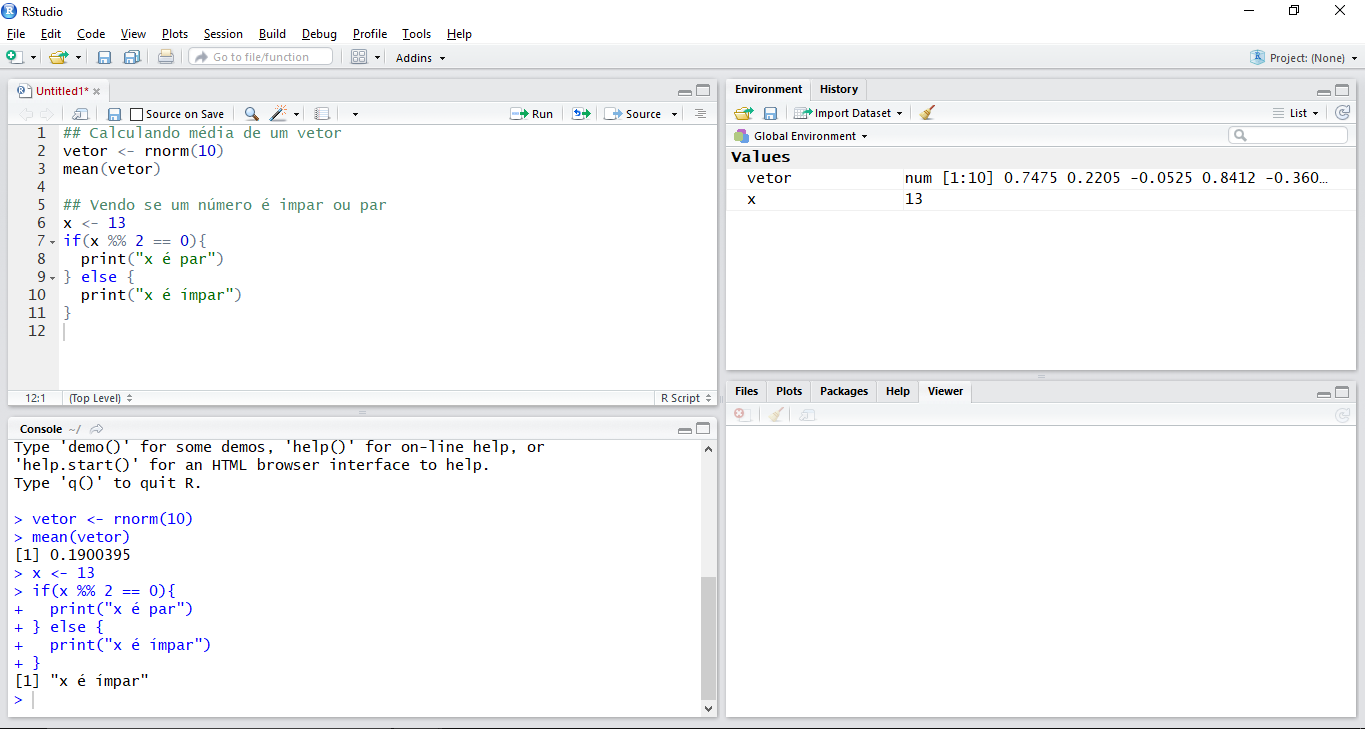
É importante entender o funcionamento da execução do código do R. Chamamos de código o conjunto de comandos que estão salvos em um script, onde cada comando representa uma ação que pedimos para a linguagem executar. Comandos podem ser simples como calcular a média de uma vetor ou podem ser mais complexos como retornar uma mensagem caso no número seja par e outra caso o número seja impar.

O *chunk* abaixo mostra como seriam escritos os exemplos citados anteriror mente:

> ## Calculando média de um vetor  
> vetor <- rnorm(10)  
> mean(vetor)   
>   
> ## Vendo se um número é impar ou par  
> x <- 13  
> if(x %% 2 == 0){  
+ print("x é par")  
+ } else {  
+ print("x é ímpar")  
+ }

O *chunk* possui ao todo 4 comandos, pois as linhas que começam com # são chamadas de comentários elas são executadas apenas como texto. O primeiro comando, vetor <- rnorm(10), cria um vetor com 10 números aleatório, o segundo, mean(vetor), calcula a média dos elementos do vetor, o terceiro, x <- 3, atribui à variável xo valor 3 e, por fim, o quarto comando, que começa no if(x %% 2 == 0), testa de x é par e retorna uma mensagem caso ele seja par e outra caso x seja ímpar.

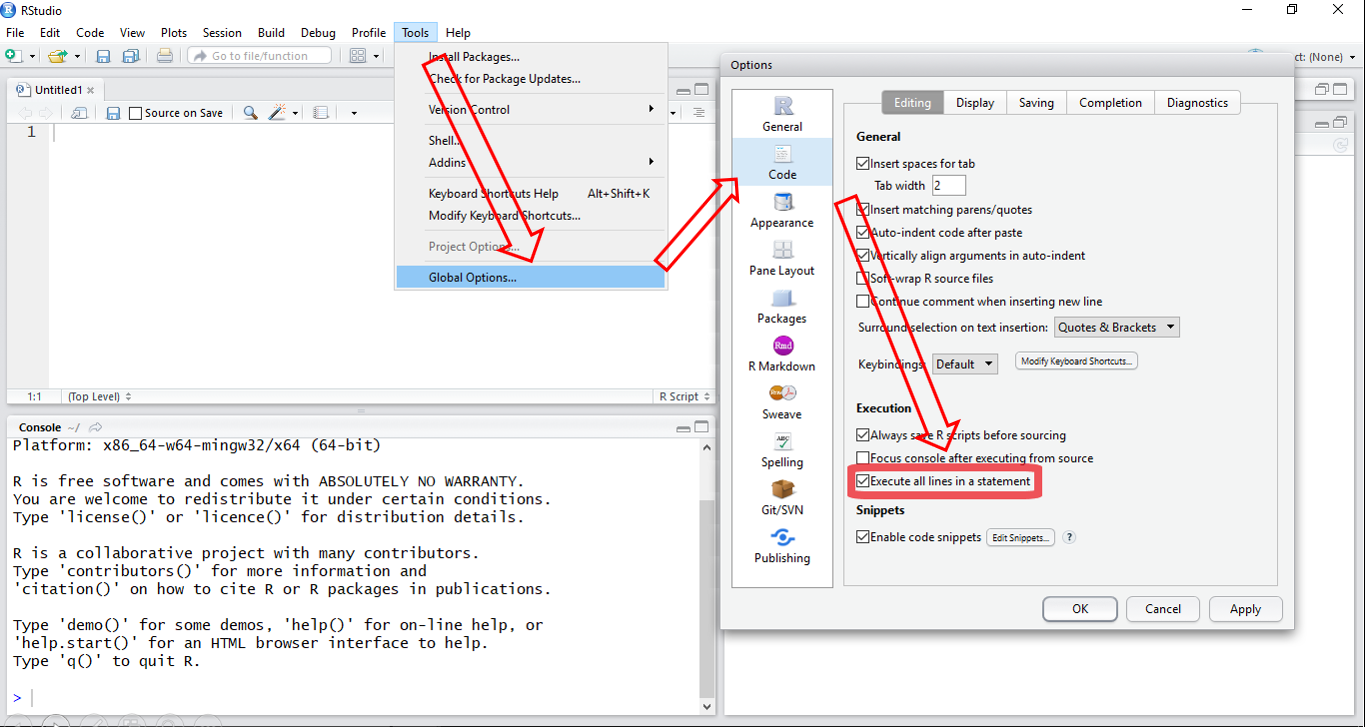
Para executar um comando utiliza-se o atalho no teclado ctrl+Enter e a linha em que você se encontra será executada, no caso dos três primeiros comandos é direto a execução com o atalho. Porém, se prestarmos atenção o quarto comando encontra-se "espalhado" em 5 linhas e aí reside um detalhe, note que sabemos que o comando tem mais de uma linha através do + que aparece nas três linhas subsequentes à primeira. O RStudio permite que você escolha entre duas opções de execução de comandos com mais de uma linha. A primeira opção é executar uma linha de cada vez, no nosso exemplo seria necessário utilizar o ctrl+Enter quatro vezes para executar o quarto comando. A segundo opção é fazer com que comandos com mais de uma linha sejam executados de uma só vez, neste caso para executar o quarto comando seria necessário utilizar p ctrl+Enter apenas na linha if(x %% 2 ==0) para executar o quarto comando. Independente de qual das opções foi utilizada para executar o código o resultado apóes executar os quatro comandos acima é apresentado na figura @ref(fig:resultados-4-comandos)



Resultado

Vemos que no console aparece o comando que foi executado, linhas que começam com >, seguido pelo resultado do comando, linhas que começam com [1], o que acontece nos bastidores é que ao executar um comando no *script* esse comando é "jogado" para o console onde ele é executado. Por fim, vemos que no *Environment* estão salvos o vetor e a variável x.

A escolha entre as duas maneiras de executar o código é uma escolha do usuário, pois o resultado final do código será o mesmo. Para optar pela execução de todas as linhas do comando de uma só vez acesse a aba e então marque a opção *Execute all lines in a statement*[[1]](#footnote-34), a figura @ref(fig:all-lines-one-statement) mostra o caminho completo.



Opções de execução do código RStudio

Um último detalhe para ae comentar sobre a execução dos códigos é que em boa parte dos comandos a execução do código é quase que instantanea, porém há situações em que o computador pode demorar para executar o código. Nessas situações é importante que você não execute nenhum outro comando enquanto o R não terminou de executar o comando anterior. Enquanto o R está executando um commando aparece uma "placa" no canto superior direito do console escrito stop, destacado na figura @ref(fig:installing-packages). Ao terminar de executar o código essa "placa" sumirá. Um exemplo de comando tende a demorar (relativamente) a ser executado é a estimação dos modelos de controle sintético apresentados por @abadie2010synthetic.

Passamos agora a discussão para entender onde o R "vive", isto é, qual a maneira como o R interaje com o computador, no sentido de buscar arquivos para serem trabalhados dentro do R ou salvar resultados que foram obtidos no R.

## *Working Directory* (ou pasta de trabalho)

O jeito mais comum de se organizar um trabalho que envolva análise dados (seja projeto de pesquisa, trabalho da universidade etc.) é através de pastas no computador. É comum que os dados que serão analisados tenham de ser importados para dentro do R (**colocar aqui referencia da seção**), ou então que você queira salvar um plot que foi gerado dentro do R, o *working directory* nada mais é do que a pasta no seu computador onde o R vai buscar os arquivos para serem importados e salvar os resultados que serão exportados.

Há duas funções que você precisa conhecer no que tange à pasta de trabalho. A primeira é para definir a pasta de trabalho e chama-se setwd(), onde wd é abreviação de *working directory* o único argumento que essa função leva é o caminho da pasta entre aspas, como o exemplo abaixo:

> setwd("C:/Users/Vinicius/Desktop/working directory")

É importante notar que as barras são inclinadas para a direita e que o endereço começa no disco em que se encontra a página, pode parecer bobeiras, mas são erros comuns nos primeiros contato com o R. Para copiar o endereço de uma pasta basta clicar com o botão direito e selecionar a opção copiar, como se fosse qualquer outro arquivo, e depois colar denteo da função setwd().

A segunda função importante para se saber é a getwd(), você consegue adivinhar o que ela faz? Isso mesmo, a função mostra para você qual é a pasta na qual o R está "vivendo" atualmente e ela não possui nenhum argumento. Por exemplo, este livro foi escrito dentro do RStudio e no momento o *working directory* é a pasta:

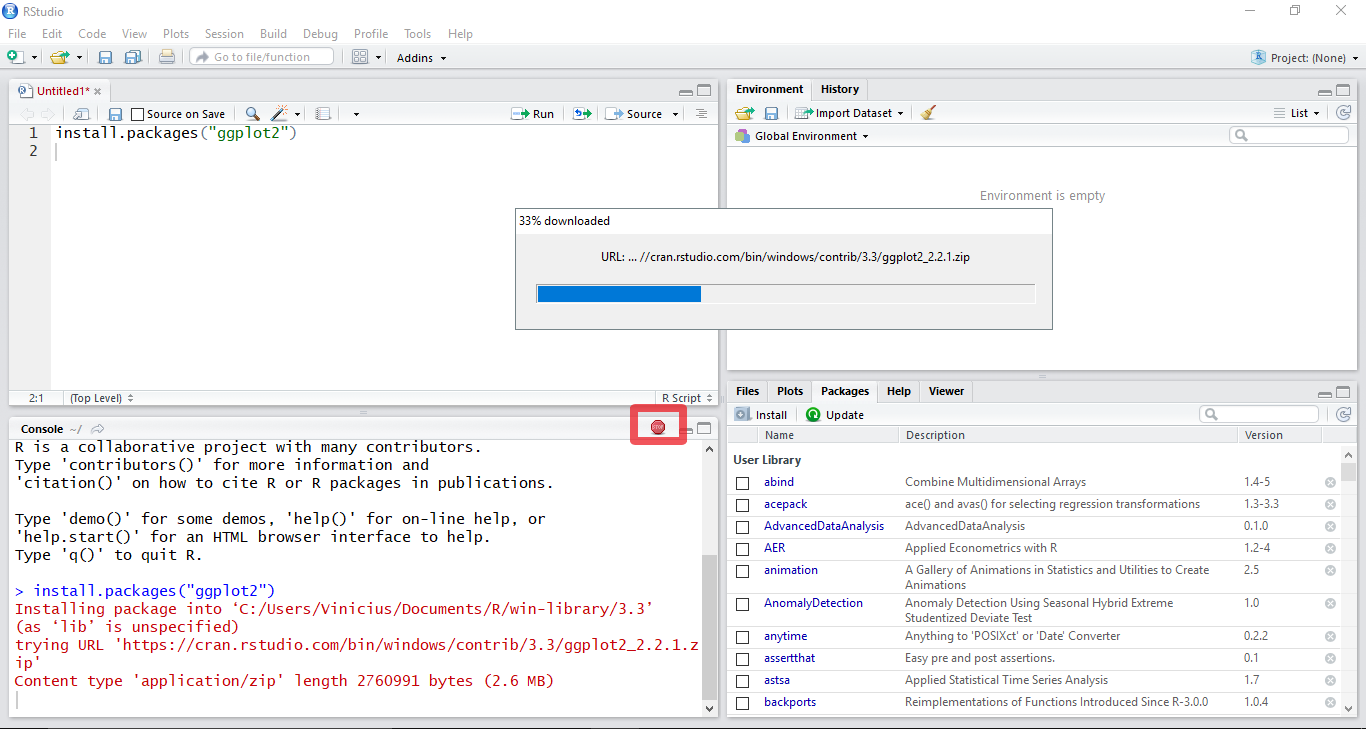
> getwd()  
# [1] "C:/Users/Vinicius/Dropbox/projects/livro\_r/\_capitulos individuais no word"

Na seção (**colocar aqui seção de boas praticas script**) iremos discutir boas práticas em relação ao *script* e *working directory*, mas vale a pena destacar adiantar uma boa prática. Como dito anteriormente é comum organizarmos nossos trabalhos acadêmicas em pastas, então suponhamos que estejamos fazendo um trabalho de econometria sobre o clássico exemplo de renda e consumo, estou assumindo que no nosso computador há uma pasta onde encontram-se os dados que o professor passou (ou você obteve), possíveis arquivos de texto e etc. É uma boa prática você criar dentro dessa pasta uma outra pasta e usá-la como *working directory*.

## Pacotes (ou *libraries*)

Em todas as esferas da análise de dados (importar dados, manipular dados, estimar modelos/testes estatísticos e reportar resultados) no R trabalha-se com funções que nos auxiliam, setwd() e mean() são duas funções já apresentadas. O R possui várias funções *built in*, isto é, que ao instalar o programa as funções já estão disponíveis. Um pacote neste contexto nada mais é do que um conjunto de funções sobre um tema parecido que não são carregadas autimaticamento ao abrir o RStudio. Por exemplo, no capítulo @ref(intrpducao\_ao\_ggplott2) vamos mostrar o básico do pacote ggplot2 que é um pacote desenvolvido para viasualização de dados e contém funções para histogramas, boxplots, gráficos de densidade 2d etc.

Para instalar um novo pacote utiliza-se a função install.packages("nome do pacote"), isso instala no seu computador as funções que estão no pacote, mas não as deixa disponíveis para utilização. Para fazer isso é necessário que você carrege as funções para o R através do comando library(nome do pacote). Ao instalar um pacote uma aba com o progresso abrirá mostrando qual o status da instalação, como mostrado na figura @ref(fig:installing-packages).



Status de instalação de pacote

Duas observações: (i) note que ao instalar o pacote o nome do pacote deve estar entre aspas e ao carregar o nome do pacote está sem aspas; (ii) \*\*só é necessário instalar o pacote uma vez no seu computador (a primeira vez que você for utilizá-lo), mas é necessário que você carrege o pacote sempre que vocÊ abrir o R.

1. Tradução livre: Execute todas a linhas em um comando [↑](#footnote-ref-34)