

2 Vamos Começar

O foco aqui é o que você precisa para instalar os softwares (*R* e *RStudio*) e tentar alguns exercícios básicos que vai facilitar nosso trabalho.

2.1 Historia de R e Porque Precisamos Ele

2.1.1 Historia

O antigo Bell Labs de ATT em Nova Jersey foi uma fonte rica das linguagens de computação. C e seus derivativos vêm de lá. S, uma linguagem que permitiu que cientistas do laboratório podem pôr estatística no contexto de computação sem precisar programar do início todas as funções necessárias para executar análises estatísticas como tinha sido necessário em Fortran ou Cobol ou Pascal. Quando o Bell Labs foi desmembrado ao final da década dos 1980s, S tornou um produto comercial (que ainda existe).

Nos anos 1990s, Ross Ihaka e Robert Gentleman de Nova Zelândia criou um derivativo *open-source* da S, que eles chamaram “R”, continuando a tradição de Bell Labs de dar às linguagens nomes de uma letra só. Ihaka e Gentleman disponibilizaram a R em 1999. Agora, estamos usando versão 3.6.1. R tem uma comunidade extremamente ativa. Desenvolvedores e usuários contribuíram com mais de 14.500 pacotes com funções e conjuntos de dados adicionais. Esses pacotes abrangem todas as áreas acadêmicas e profissionais em que possamos pensar, de análise de finanças quantitativas até astrofísica até microbiologia. R realmente tornou na última década uma das mais importantes linguagens de computação.

Nas faculdades em quase todo o mundo, R é a ferramenta de pesquisa mais popular hoje. E a presença dela nas empresas está crescendo diariamente. Especialmente em medicina e microbiologia, R tem recursos muitos importantes. Além dos pacotes das funções em R, existe pacotes do sistema Bioconductor que tratam de quase todas as

análises importantes possíveis genômicas que precisamos usar para projetos em biologia e medicina.

Uma das razões que explica este crescimento forte da R é que ela fica *open-source*. Portanto, pode adquirir **de graça**, sim—livre de custo. **Custo zero** (para aqueles que não acreditaram a primeira vez).

2.1.2 Porque Professor Jim Usa R

Eu tenho uma longa história com programas como SPSS, Stata, SAS e mesmo Graphpad Prism. Comecei de usar SPSS quando ainda estava nos cartões IBM e os computadores eram os IBM 360s (aliás, os anos 60). Tudo bem. Usei várias versões de todos esses outros programas, mas sempre eu estava insatisfeito. Em 2012, descobri R e percebi que minhas insatisfações com SPSS, etc. tiveram solução. A vantagem grande de R sobre os outros programas é que você pode controlar absolutamente a sequência das operações (passos) em sua análise. Você escreve um programa num *script* (i.e., pequeno programa) e o R vai executar exatamente o que você o mandou fazer. E amanhã vai executar uma nova análise com exatamente o mesmo resultado que ontem.

2.1.3 Problemas com os Tradicionais Programas da Análise de Dados

Com todas as caixas de diálogo e menus, os programas da análise de dados distanciam você dos detalhes e passos da análise e da limpeza dos dados. E mais, se você não clicar nas opções exatas na terça-feira da análise que você fez inicialmente na segunda-feira, a análise produzirá um resultado diferente. Pior ainda, essas caixas oferecem muitas opções que você não entende (porque você é um administrador, biólogo, arquiteto, etc., mas não um estatístico) e como resultado, a análise não faz sentido e você não pode realmente explicar as conclusões.

Como disse Prof. Sacha Eskamp da Universidade de Amsterdã:

You can only do what the buttons say you can do. ¹ [Você pode fazer somente o que os botões dizem que você pode fazer.]

2.2 Software: Open-Source vs. Comercial

Além disso, o custo desses programas é absurdamente caro. Mesmo nas edições para estudantes, o preço frequentemente fica acima do que os alunos podem pagar. Como efeito colateral, essa encoraja a proliferação das cópias não autorizadas. Onde fica a ética em ciência ou negócios quando alunos precisam começar a carreira roubando software? Faz muito tempo, softwares *open-source* eram de baixa qualidade. Mas, não mais. Existem equivalentes para os softwares pagos em quase todas as áreas em que precisamos programas. O mundo dos softwares *open-source* realmente mudou. Fora do Microsoft Word®, eu raramente uso softwares pagos.

2.3 R vs. Python

Uma alternativa para R com um conceito paralelo de programação das análises é a linguagem *Python*. Lançado em 1991 pelo holandês Guido van Rossum, Python tirou o nome do grupo comédico inglês “Monty Python’s Flying Circus”, não da espécie da serpente. Ela é uma linguagem de alto nível interpretada, exatamente como R. Mas, Python foi desenhada como uma linguagem geral em contraste a R, que tem sua origem em estatística. Para executar até estatística básica em Python, precisa também aprender a programação de vários módulos como Pandas, necessários para representação dos conjuntos de dados em Python.

R e Python estão tornando mais compatíveis. Nesta matéria, vocês aprenderão as duas idiomas juntas.

2.4 Aprendizagem de R, É Difícil?

É bastante fácil aprender os elementos básicos de R, inclusive, especificar vetores e conjuntos de dados (que chamamos em R, *dataframes* ou *tibbles*), executar funções básicas de estatística e matemática. No primeiro fim de semana, nós vamos construir e executar várias simulações Monte Carlo,² uma tarefa que precisa habilidade de construir e especificar dados e fazer uma análise.

Então, nas fases iniciais de aprender R, você não vai sofrer muito. Mais tarde, quando você quer utilizar pacotes e funções mais avançados (como os de Bioconductor), esses

precisarão bastante foco e concentração para programar corretamente. Mas, nós deixaremos essa fase para a última aula do curso e os módulos seguintes.

Como as outras linguagens de programação, você escrever comandos numa forma que o programa entende, um depois do outro com uma sintaxe abreviada que segue as regras da função que você está executando. Por exemplo, se queremos achar a média de 100 números aleatórios entre 1 e 1000, podemos dizer o seguinte:

```
set.seed(1)
dados <- runif(100, min = 0, max = 1000)
media <- mean(dados)
```

Este exemplo funciona. (O resultado é 517.847.) Quando você instala R, pode executar os comandos. Mas, o ponto aqui é mostrar para vocês que o código precisa ser escrito exatamente como o programa demanda. Como vocês provavelmente já sabem, o computador é uma máquina de adição grande e estúpida (*“a computer is a big, dumb adding machine”*). Ele faz exatamente o que você comanda ele fazer.

2.5 Advertência Importante

Vai ser imprescindível que vocês investem um pouco de tempo no curso antes das aulas presenciais. Primeiro, precisam instalar os softwares nos seus laptops para poder fazer as lições da casa. Segundo, precisa ler o material que sugerirei aqui. E, terceiro, talvez podem experimentar um curso de R básico no Internet, pelo menos tentar algumas lições para não ficar totalmente perdido na primeira aula.

2.6 Instalação dos Softwares

2.6.1 Instalação do R

Você pode achar R no site seguinte:

<https://cran.r-project.org/>

CRAN é o grupo oficial que mantem R e garanta que todos os componentes funcionam

sozinhos e juntos. Esta é uma razão que R está tornando muito popular – a qualidade do produto é muito alta. Figura 1 mostra a tela principal de R.

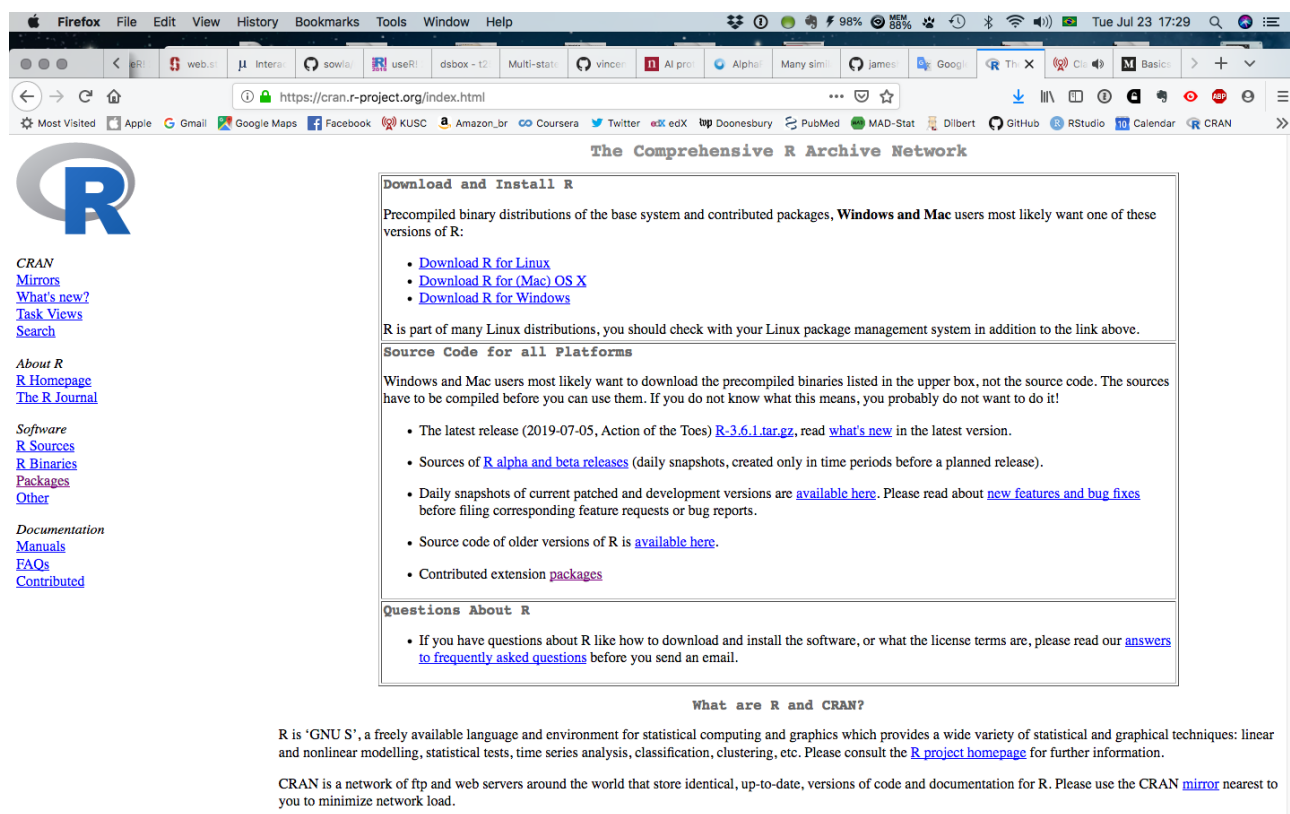


Figure 2.1: R Tela Inicial

Verifique que a versão é 3.6.1, com o codinome de “Action of the Toes”.

Existem versões para Windows, MacOS (o novo nome de OSX) e para vários sabores de Linux. Na parte superior da tela, clique no link que é apropriado para seu sistema operacional e seguir as instruções para download do software. Vou detalhar aqui alguns dos passos para Windows e para MacOS.

2.6.2 Instalação de R para os Windows

1. Clique no link “Download R for Windows”
2. Na próxima tela, clique no link “base” como na figura seguinte.

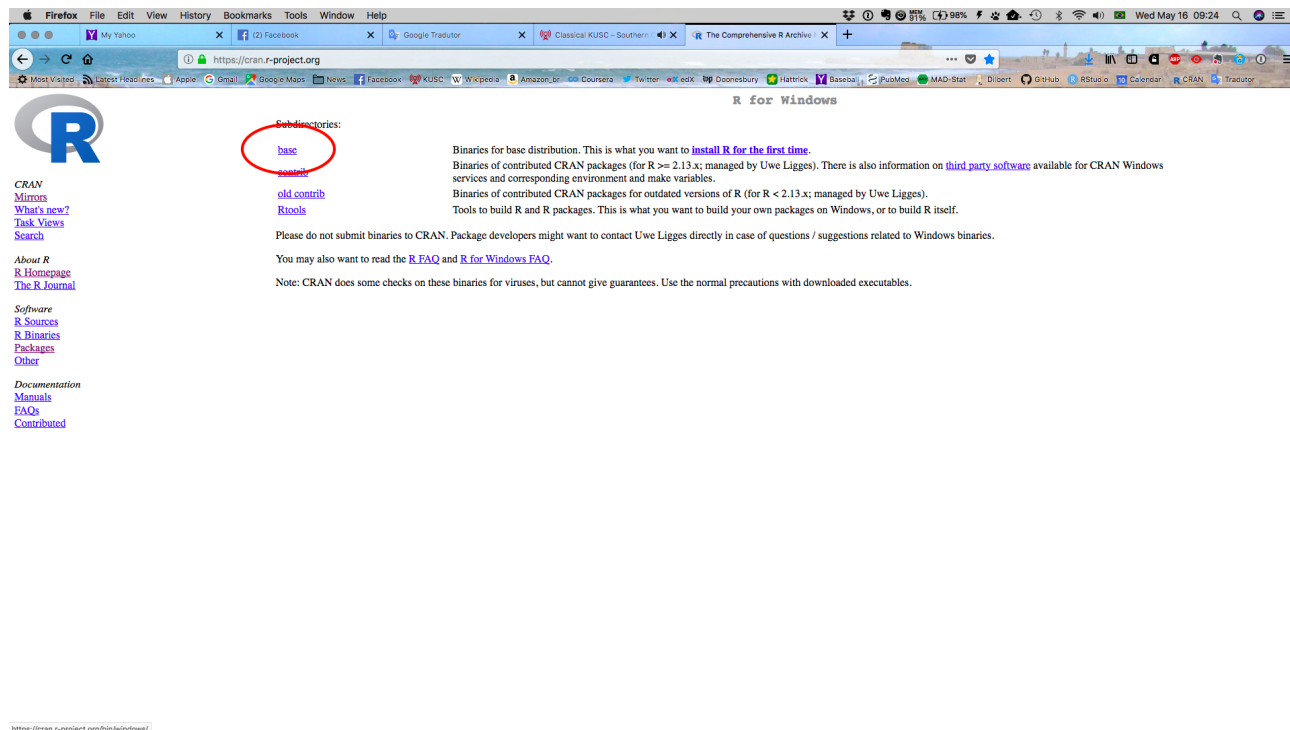


Figure 2.2: R Tela Windows

3. Clique no link: “Download R 3.6.1 for Windows”

O site vai fazer o download do programa como é normal para os downloads. Você pode instalar o programa seguindo seu procedimento normal e R vai depositar no Desktop um ícone.

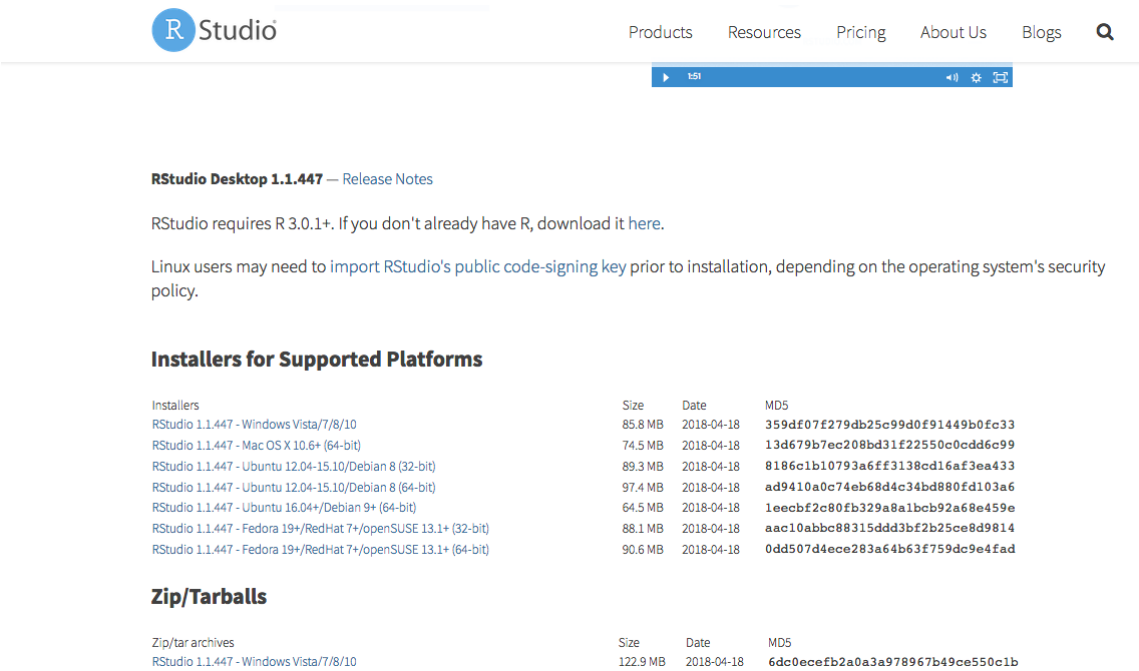
2.6.3 Instalação do RStudio

Primeiro, porque precisamos este programa? RStudio dá asas para R. Ao nível mais básico, RStudio é nossa interface para R. É um programa baseado num editor de texto onde você escreve os scripts que vai desenvolver para R. Mas, é realmente o canivete suíço de R. Lá pode gerenciar os arquivos, estabelecer projetos que colocam todos os arquivos de um projeto junto, fazer controle de versões, e ver seus resultados, relatórios, etc. Em todas nossas aulas, vou usar RStudio para controlar todas as demonstrações de programação, e RStudio é o software em que fiz toda a documentação da matéria. RStudio também é de graça. Existe uma versão comercial, mas vocês não precisam ela.

Para instalar, navegue diretamente á página <https://www.rstudio.com/products/rstudio/download/>, que permite que você evita muita informação desnecessária.

Esta tela tem muita informação. Então precisa rolar para baixo até que você chega nos

“Installers for Supported Platforms”. Parece como a figura seguinte:



(#fig:Rstud_install)RStudio Tela Instalação

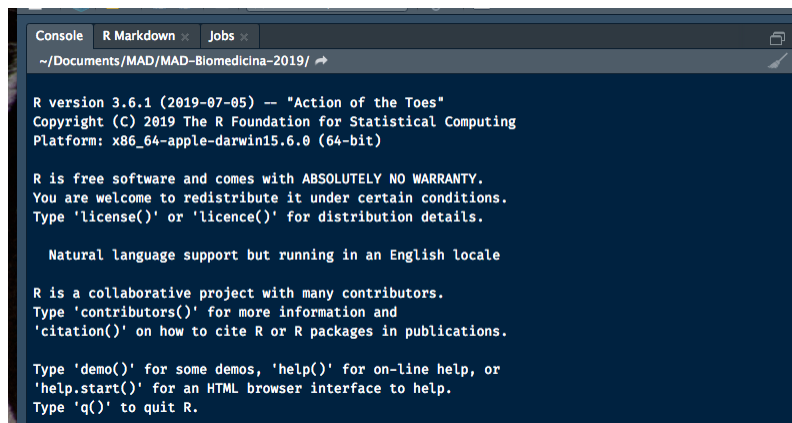
Agora, pode fazer o download para o sistema operacional seu e instalar o programa seguindo as instruções no módulo de instalação. (O número da versão provavelmente seria diferente do aquele na figura.)

Se vocês precisam ajuda adicional com a instalação, coloquei um vídeo no meu canal de YouTube chamado “MAD CD Instalar R”. Para alcançar o canal, vá para: <https://www.youtube.com/channel/UCbvgZ8RYeTtgjhAKE-jub5A>

2.7 Iniciando RStudio

No seu Desktop ou nos menus, clique no ícone de RStudio (não o ícone de R) duas vezes. RStudio vai abrir e R vai abrir automaticamente dentro de RStudio. Num dos painéis (normalmente o painel no lado esquerdo) vai aparecer o *Console*, onde R executa as funções. No lado direito têm dois painéis. Naquele para cima são várias abas relacionadas aos variáveis que R guarda na memória ativa do computador e comandos que você executa. No painel inferior, têm abas que mostram informação sobre os arquivos na pasta “*working directory*”, o lugar onde R acha os arquivos (data, programas) que você quer executar, os pacotes que você tem no computador e as páginas de assistência (“*Help*”) para R. Vou falar sobre todos esse tópicos mais tarde. Utilizando as Preferências (no item de menu de “RStudio”), você pode controlar a aparência da tela).

O painel de Console mostrará que a versão atual (3.6.1) de R está carregado e pronta para operação.

The image shows a screenshot of the RStudio application window, specifically the Console pane. The title bar at the top indicates the active pane is 'Console'. The address bar shows the current working directory as '~/Documents/MAD/MAD-Biomedicina-2019/'. The console output displays the standard R startup message for version 3.6.1 (2019-07-05), including copyright information for The R Foundation for Statistical Computing and the platform 'x86_64-apple-darwin15.6.0 (64-bit)'. It also includes a disclaimer about the software being free and without warranty, and provides instructions on how to use various R functions like 'license()', 'contributors()', 'citation()', 'demo()', 'help()', 'help.start()', and 'q()'.

(#fig:Rstud_main)RStudio Tela Inicial

2.8 Instalar os Pacotes Necessários

Durante o curso, nós vamos usar vários pacotes com funções adicionais que vão além as de R básico (“*Base R*”). Vocês podem instalar os mais importantes através de um script que já preparei para você. Trabalhando com o script vai permitir que vocês comecem de aprender como usar o R. O script tem o nome “instalar_pacotes.R”.

O seguinte é o texto do arquivo. É muito simples. Só 2 comandos. Um que especifica os pacotes que vamos instalar e uma função que faz o trabalho (`install.packages()`).


```
pacotes <- c("tidyverse", "broom", "car",  
            "caret", "corr", "data.table", "DataExplorer",  
            "descr", "devtools", "gapminder",  
            "ggpubr", "ggvis", "glue",  
            "gmodels", "here", "Hmisc",  
            "hms", "jsonlite", "kableExtra",  
            "knitr", "lattice", "librarian",  
            "lubridate", "magrittr", "mice",  
            "nortest", "nycflights13", "outliers",  
            "pROC", "psych", "RColorBrewer",  
            "Rcpp", "readxl", "ROCR",  
            "shiny", "styler", "usethis", "titanic")  
  
install.packages(pacotes)
```

2.8.1 Nosso Primeiro Programa

O primeiro comando, especificando os pacotes para carregar, usei 2 comandos importantes para R. Eu quero usar o nome `pacotes` para referir à lista dos 37 pacotes. Não quero teclar eles cada vez preciso usar. Chato. Então, estou *atribuindo* a palavra `pacotes` para ficar no lugar da lista inteira. Faço isso com o símbolo de uma seta pontado para a esquerda “<-”, composto do caráter “<” e o hífen “-”. Este é chamada o operador *assignment* e não é equivalente ao sinal de “==”, que é uma condição lógica (uma coisa sendo igual a uma outra). Entretanto, R quer ser tão amigável que permite que você usa o sinal “=” para atribuição.

Uma dica sobre o operador de atribuição. Só usa o “<-” e não o “=”. Em algum momento, você vai confundir os 2 usos de “=”. Todos nós fazemos. Desenvolve o hábito agora de usar o “<-” e pode evitar esse erro.

O segundo comando nesta expressão é o `c()`. Este função quer dizer “concatenar” ou colocar os itens que seguem num vetor. Os itens podem ser qualquer tipo de dado que R reconhece. No capítulo X, exploraremos os tipos de dados que pode usar em R. Outra novidade é que agora você viu um **vetor**. Um vetor é um matriz, mas de uma dimensão só. Depois que você executa esta função, vai ver na janela de *Environment* a variável `pacotes` e que ela é do tipo “caráter” (`chr`) e tem elementos 1 até 37.

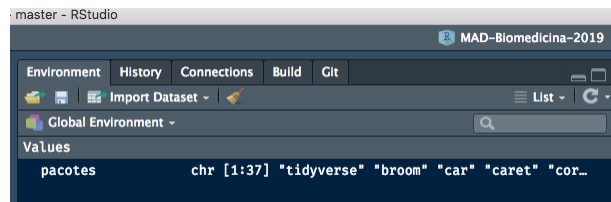


Figure 2.3: Vetor Pacotes

Na função `install.packages(pacotes)`, R vai inserir o conteúdo do vetor `pacotes` no lugar da palavra e assim executar.

2.8.2 Localização do Arquivo `instalar_pacotes.R`

Um dos recursos mais importantes para todos os programadores de computador é GitHub. É a face pública do sistema de controle de versões `git`. Git permite que você sempre tem todas as versões de seus programas ou scripts mantidas no seu computador (numa forma eficiente) para que você possa voltar às versões anteriores quando você está explorando e corrigindo erros que todo programador faz. GitHub tem *repositórios* onde são guardados os arquivos de um projeto. Nós vamos falar dos projetos um pouco mais tarde.

Nosso curso tem um *repo* (como pessoas referem aos repositórios) que você pode achar aqui: <https://github.com/jameshunterbr/MAD-Biomedicina-2019>

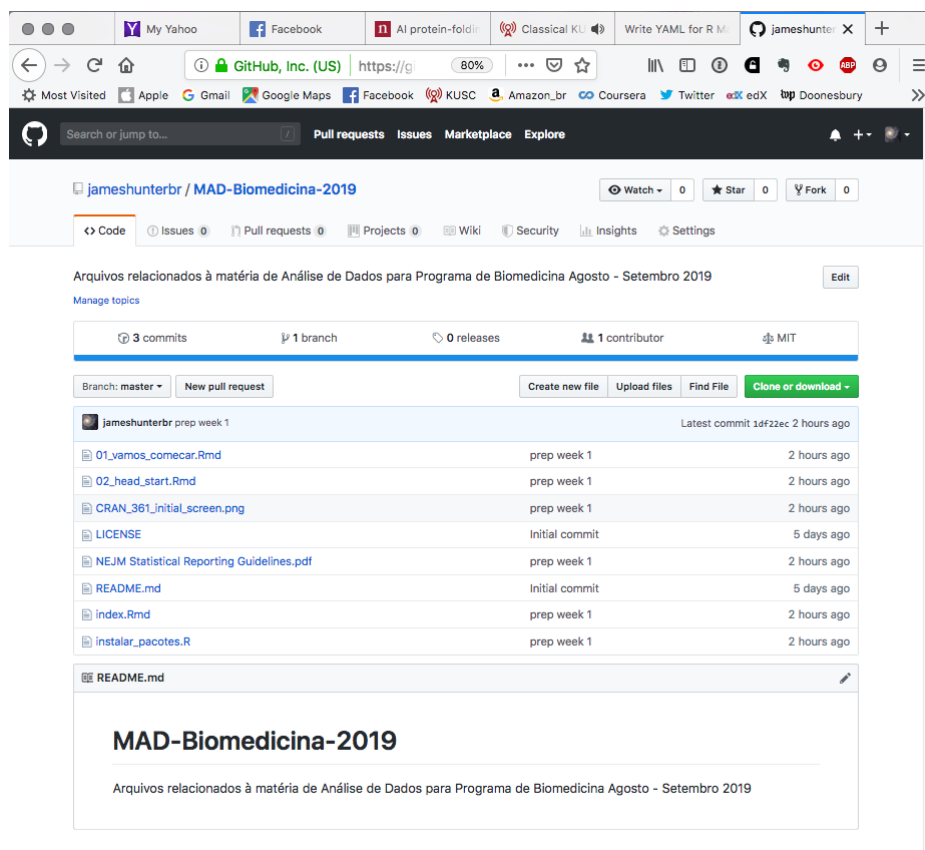


Figure 2.4: GitHub Repo

Se você clica no nome de arquivo, você irá à tela seguinte, onde fica o código dos dois comandos.

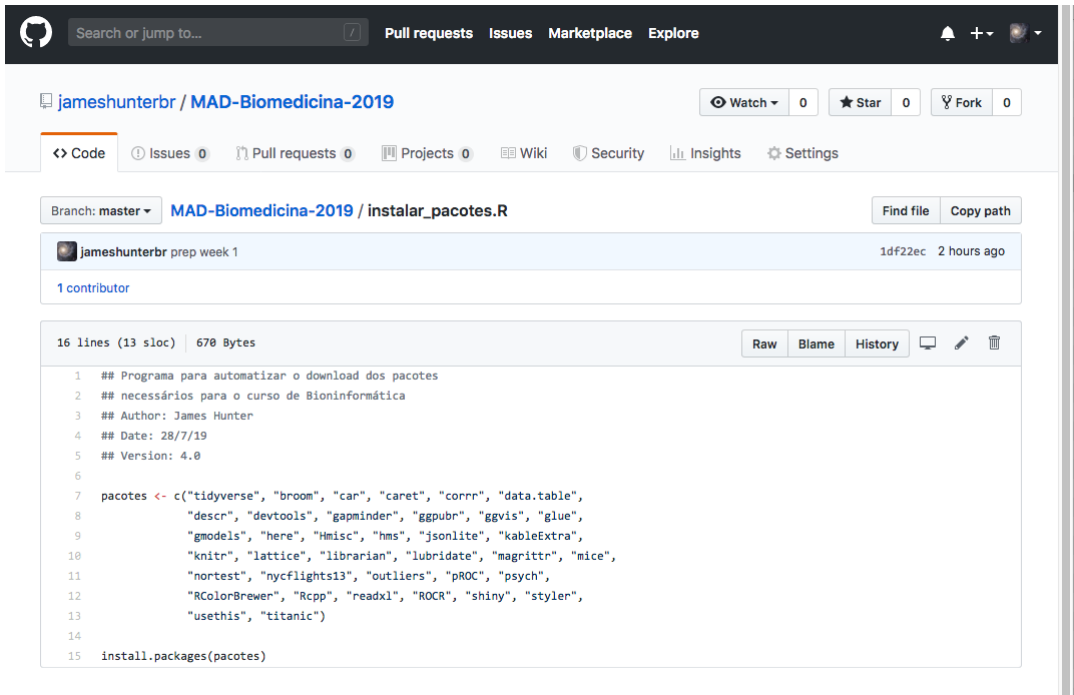


Figure 2.5: Instalar_Pacotes.R no GitHub

Aqui, clicar no botão **Raw** e fazer um “right-click” (botão secundário do mouse) na tela seguinte. Você pode salvar o arquivo para seu *working directory* usando o botão **Save Page As ...**. Esta tela mostra o comando no Firefox e os comandos de seu browser talvez são um pouco diferentes.

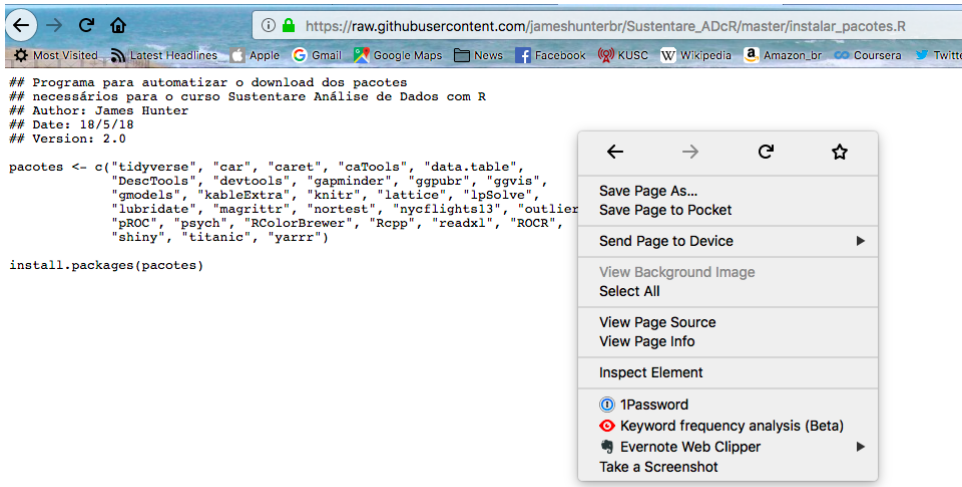


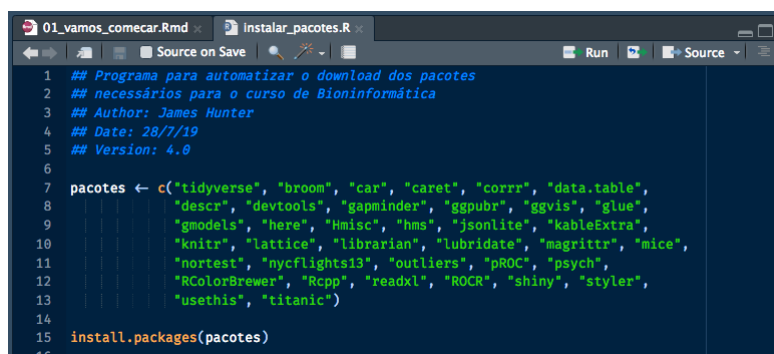
Figure 2.6: Save File As ...

2.8.3 O Que É Um *Working Directory*?

O *Working Directory* (“pasta de trabalho”) é o lugar onde R espera achar todos os arquivos que vai precisar (dados e scripts). Você pode achar ele em RStudio clicando o item do *toolbar* da aba *Files* chamado *More/Go To Working Directory* e RStudio vai levar você lá. Se o *Working Directory* não é a pasta que você esperou, pode navegar na lista dos arquivos para a pasta que quer usar como o *Working Directory*. Nós vamos ver uma alternativa para esta estratégia quando apresento projetos.

2.8.4 Instalar os Pacotes

O próximo passo é carregar o programa com os comandos e executá-lo. Para carregar, só precisa clicar no nome do arquivo `instalar_pacotes.R` e ele vai abrir numa janela acima do Console.



```
1 ## Programa para automatizar o download dos pacotes
2 ## necessários para o curso de Bioninformática
3 ## Author: James Hunter
4 ## Date: 28/7/19
5 ## Version: 4.0
6
7 pacotes <- c("tidyverse", "broom", "car", "caret", "corrr", "data.table",
8             "descr", "devtools", "gapminder", "ggpubr", "ggvis", "glue",
9             "gmodels", "here", "Hmisc", "hms", "jsonlite", "kableExtra",
10            "knitr", "lattice", "librarian", "lubridate", "magrittr", "mice",
11            "nortest", "nycflights13", "outliers", "pROC", "psych",
12            "RColorBrewer", "Rcpp", "readxl", "ROCR", "shiny", "styler",
13            "usethis", "titanic")
14
15 install.packages(pacotes)
```

(#fig:janela_script)RStudio Janela de Programas

Para executar o programa, você pode clicar na palavra *Source* na barra acima do programa. Vai precisar um pouco de tempo, mas vai carregar no seu computador esses pacotes e outros em que eles dependem para funcionar.

2.9 Prontos para Trabalhar

Fizemos a instalação dos softwares e a maioria dos pacotes que vamos usar durante o curso. Finalmente é a hora de usar R, que vamos fazer no próximo capítulo. Mas primeiro vamos relembrar os pontos importantes deste processo inicial:

1. Fizemos o download do web da R e da RStudio e instalamos os programas
2. Aprendemos o que é o *Working Directory*
3. Aprendemos o que é o operador de atribuição (`<-`)
4. Aprendemos o que é um vetor
5. Aprendemos a função `c()`

6. Aprendemos que o comando para instalar pacotes é `install.package()`
7. Temos o repo no GitHub: <https://github.com/jameshunterbr/MAD-Biomedicina-2019>
8. Pode consultar o canal no YouTube: <https://www.youtube.com/channel/UCbvgZ8RYeTtghAKE-jub5A>

1. Baker, Monya, “Code Alert”, **Nature**, Vol 541, 26/1/2017, p. 563 - 565. ↩
2. Irizzary, Rafael, **Introduction to Data Science**, (<https://rafalab.github.io/dsbook>), Ch. 26.3. ↩