Conceitos de DS

Bem vindo ao mundo da ciência de dados

raccoon MARKETING DIGITAL

S4 CAPITAL

media.monks

1 Data Science/Data scientist

O que é isso?

2 Ferramentas de DS

RStudio;
Spider/Python;
Github/Gitbash;
Credenciais
Drive;
Planilha de controle de análises;
Doc Marketing Objectives.

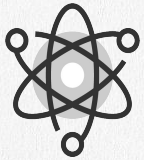
3 Boas práticas de código

Visão macro do código;
Visão micro do código;
Ferramentas úteis;
Materiais para consulta.

Nos importamos
com **Ma**

Data science/Data scientist

Data Science: O que é?



O que faz um Data scientist?

Trilha de DS

Cronograma - Trilha de Data Science

Semana 1:

- Conceitos de Data Science
  - **03/01/21 - Cristiano**
- Não seja enganado pelos números/Storytelling com dados
  - **03/01/21 - Gustavo**
- Dicas de Planejamento e execução de análises
  - **04/01/21 - Treinamento gravado**
- Como meus dados se comportam?
  - **04/01/21 - Treinamento gravado**
- Manipulação em R
  - **05/01/21 - Vittória**
- Autenticação e extração
  - **05/01/21 - Wesley**
- Visualize melhor os seu dados
  - **06/01/21 - Treinamento gravado**
- GGPLOT
  - **06/01/21 - Treinamento gravado**
- Plantão de dúvidas
  - **07/01/21 - Cristiano, Gabriel, Vittória, Gustavo e Wesley**

Cronograma - Trilha de Data Science

Semana 2:

- Análise de Elasticidade
  - **10/01/21 - Gabriel**
- Análise N-Gram
  - **12/01/21 - Wesley**
- Plantão de Dúvidas
  - **14/01/21 - Cristiano, Gabriel, Vittória, Gustavo e Wesley**

Cronograma - Trilha de Data Science

Semana 3:

- Análise de Markov
  - **17/01/21 - Cristiano**
- Análise de Sazonalidade
  - **19/01/21 - Treinamento gravado**
- Plantão de Dúvidas e entrega das PBL's
  - **21/01/21 - Cristiano, Gabriel, Vittória, Gustavo e Wesley**

Ferramentas de DS

Ferramentas

- RStudio:
  - Como instalar.
- Spider - Python:
  - Como instalar.
- Github/Gitbash;
  - Manual de Uso e Comandos.
- Credenciais
  - Criar pasta “Credenciais” no caminho “*C:\Users\Usuario\Documents\Credenciais*” e salvar os arquivos enviados por e-mail.
- Materiais para consulta:
  - Drive DS;
  - Planilha de controle de análises;
  - Doc Marketing Objectives;

Boas práticas de código

Objetivo: Definir padrões de produção de código para torná-los mais concisos.

Sumário:

- Visão macro do código;
- Visão micro do código;
- Ferramentas úteis;
- Materiais para consulta.

Visão Macro

Por que seguir um padrão com seções?

- Facilidade de leitura. O code review se torna mais rápido;
- Agilidade nas alterações de código, pois cada tipo de informação tem o seu “espaço”;
- Códigos com trechos espalhados dificulta a identificação de bugs;
- Replicabilidade do mesmo código alterando pouca informação (Ex.: Dados de cliente, métricas, dimensões e etc.).

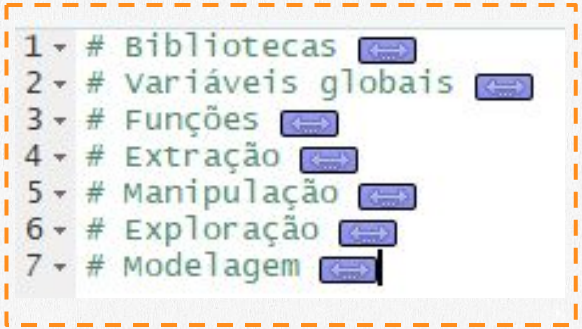
Reflexão: Imagine que você tenha um código com mais de 500 linhas semelhantes a um spaghetti (Tudo espalhado no código). Será trabalhoso adivinhar onde estão as informações, não?

Visão Macro

Normalmente quase todos os códigos podem seguir as seguintes seções:

- Bibliotecas;
- Variáveis globais (Ids, datas, parâmetros de modelos, ...)
- Funções personalizadas;
- Extração;
- Manipulação;
- Exploração;
- Modelagem.

O RStudio permite criar seções através do comentário
 # Seção A ----



Em Python
 # %%

Reflexão: Nem todo código terá exatamente estas seções, mas o importante é seguir o raciocínio semelhante.

Visão Micro

Por que seguir um padrão de escrita?

- Enquanto o Macro poderia ser o idioma falado, a visão micro seriam os dialetos;
- Se você ler um texto com pontuações esquisitas e gírias você não entenderá o contexto, mas se a escrita seguir um padrão, será mais fácil de entender;
- Também é importante manter seu código em um idioma. Não declare objetos em português e outros em inglês. **Escolha um padrão**.

Visão Micro

Nome de Arquivos

- Importante que os nomes tenham significados. Há casos que um mesmo projeto possui mais de um arquivo .R, se o nome não traduzir o que está acontecendo, teríamos que adivinhar;

- Também é importantíssimo não utilizar espaço no nome de arquivos. Utilizamos o caractere "_" para separar nomes compostos;

Padrão	Ruim
fit_models.R	fred.R
utility_functions.R	gambiarra.R
graphics.R	fit models

Visão Micro

Nome de Objetos

- Importante que os nomes tenham significados. Normalmente variáveis colocamos substantivos e funções utilizamos verbos, desde que faça sentido;

- Não ocupe muito espaço com preposições no nome. Uma combinação de substantivo(s) e verbo(s) já é suficiente.

Padrão	Ruim
start_date	primeiro_dia_do_mes
calculating_rmse()	calculoDoRMSE()
graph_mean	graficoDasMean

Visão Micro

Sintaxe

- Tentar padronizar a maneira de escrever o código, para facilitar a visualização. Entre caracteres como “<-”, “=”, “+”, “,”, ... sempre dar um espaço.
  Exceto para o “:”, como 1:10 e package::function();

- Sempre que for utilizar “{“ quebre linhas.

Padrão	Ruim

x <- 2
if (x == 2) {
  y <- x + 2
  return(y)
}
	x<-2
	if(x==2){y<-y+2
	return(y)}

Visão Micro

Sintaxe

- Caso queira dar espaçamentos extras para manter o padrão, está ok também.

  Exemplo:

  ```
  name      = “Fred”
  city      = “San Charles”
  born_city = “Olimpia”
  ```

Visão Micro

Indentação

- Importante manter um padrão de quebra de linhas, nunca comece funções ou loops e continue operando como se nada tivesse acontecido;

- Quando for identar, utilize **4 espaços** ou **1 tab**, nunca misture **tab + espaços.**

- **A única exceção é quando os argumentos de uma função ocupam mais de uma linha, assim pule espaços até que chegue na "altura" de onde começou a definição.**

Padrão

```
long_function <- function(a = "a long term",
                          b = "a medium term",
                          c = "a very small term"){
    # body of function
}
```

Visão Micro

Tamanho das linhas

- É como se estivesse deixando um texto **justificado.** Pense em um relatório no word, a boa prática é deixar cada linha com uma margem independente, ou ter uma margem máxima?
  O mesmo acontece na programação. O padrão normalmente é **80 caracteres.**

  E não, você não precisa contar os caracteres.

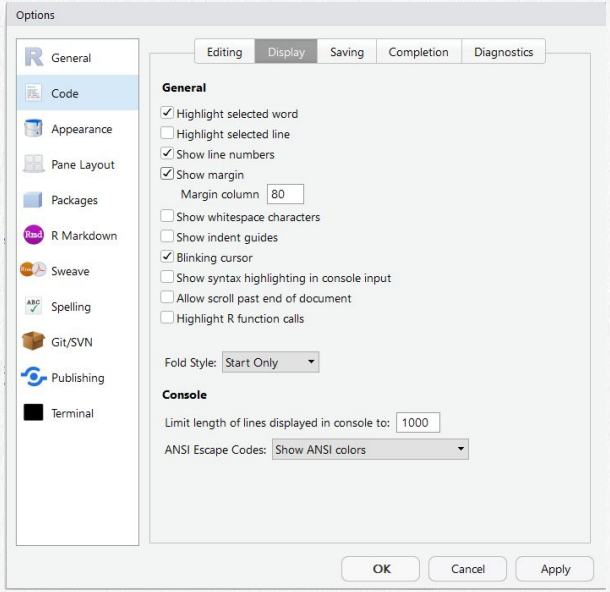
- Algumas IDEs disponibilizam uma ferramenta que te mostra onde encerra o caracter **X** que você especificar;

- Assim, quando você utilizar uma função que tenha muitos argumentos, o padrão é ir quebrando linhas a cada argumento.

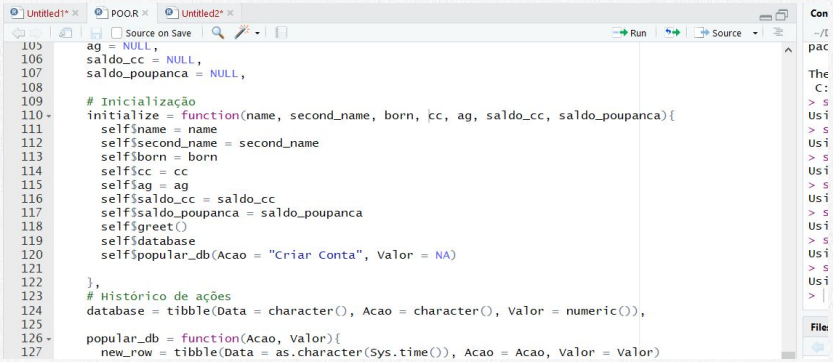
Visão Micro

Tamanho das linhas

Tools (lá em cima) > Global Options > Code > Display > Habilite a opção **Show Margin**



>>>>



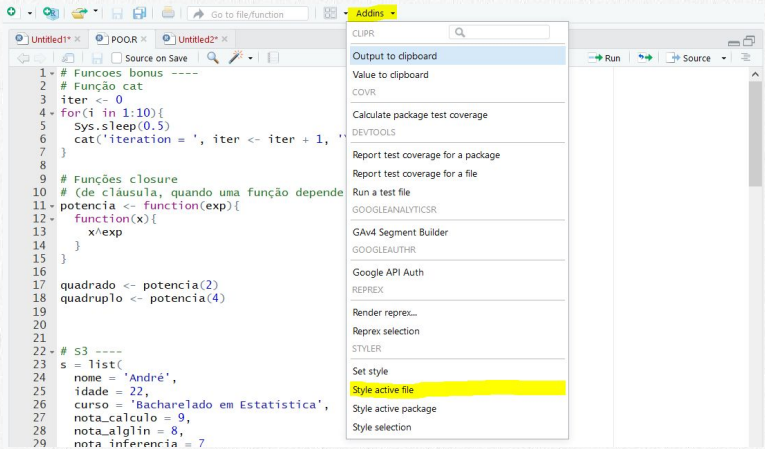
Note que fica uma linha vertical fina indicando onde se encerra o caracter 80. Não irá dar nenhum erro caso ultrapasse, é apenas um auxílio visual.

Styler

- Existe um pacote no R que se chama “styler” ele é um **Addins** que auxilia em algumas boas práticas de código;

- Ao rodar o “Style active” ele tentará verificar algumas inconsistências referentes a espaçamento e organizará todo o código (ou parte selecionada) para você;

- Basta dar um install.packages(“styler”) que ele irá aparecer automaticamente no Addins.



Styler

- Padrão Google no R
  https://google.github.io/styleguide/Rguide.html

- Style do Hadley Wickham (Tidyverse)
  http://adv-r.had.co.nz/Style.html
  https://style.tidyverse.org/

Dúvidas?

Agradecemos!

data-science@raccoon.ag

Referências

https://www.cetax.com.br/blog