



 Início  E-book (Gratuito)  Sobre  Contato

Se junte a nossa comunidade e receba atualizações de artigos, tutoriais e muito mais!



Insira seu email aqui

QUERO RECEBER! 

 Início > 2017 > novembro > 10 > Dominando o Pandas: A Biblioteca para Análise de Dados preferida entre os Cientistas de Dados (Parte 2)



Dominando o Pandas: A Biblioteca para Análise de Dados preferida entre os Cientistas de Dados (Parte 2)

 RODRIGO SANTANA  NOVEMBRO 10, 2017  4 COMMENTS  DATA ANALYSIS



Neste artigo daremos sequência na série **Dominando o Pandas**

Falar do **Pandas** nunca é demais, aliás daria um livro se fôssemos mostrar todo o potencial dessa biblioteca.

Mais interessante que mostrar diversas funcionalidades é mostrar **sus aplicações**.

Assim além do leitor aprender sobre a biblioteca, também aprende sobre como aplicar seus recursos.

Na [artigo anterior](#) exploramos uma base de dados e vimos como o Pandas é útil para análise de dados.

Vamos continuar explorando essa base usando os recursos que o Pandas nos oferece.

Sempre de forma que seja **útil para o leitor**, nada de copiar a documentação e colar aqui 😊

Resumo da ópera:

- [Scripts e Dataset](#)
- [Análise de Dados](#)
- [Estatística Descritiva](#)
- [Extraindo Insights](#)
- [Correlação de Variáveis](#)
- [Tabelas Pivot](#)
- [Crosstab, Wtf?](#)
- [Trabalhando com Excel](#)
- [Gerando Planilhas](#)

Se você já gostou do assunto desse artigo, não deixe de compartilhar com seus amigos para que cada vez mais pessoas aprendam sobre como usar o Pandas 😊

Aproveite e assine nossa lista para ficar sempre atualizado com novas postagens.

Coloque o seu e-mail abaixo para receber gratuitamente as atualizações do blog!

Seu melhor email

QUERO RECEBER! 

Scripts e Dataset



Para este artigo escolhi um dataset bem interessante que pode ser baixado gratuitamente no site do [Kaggle](#).
Dominando o Pandas

Essa base de dados é uma base de dados de preço e diversos outros atributos sobre imóveis nos EUA.

Para ficar fácil, disponibilizo a base aqui juntamente com os Scripts (notebook) utilizados neste artigo.

Base de dados : [hc_house_data](#)

Scripts: [Scripts Artigo \(Parte2\)](#)

É só baixar a base de dados e os scripts para acompanhar tudo que foi feito neste artigo.

Se não conhece essa biblioteca, ou não tem instalada, [aqui](#) ensinamos como instalar.

São 2 minutos, coisa rápida...:)

Análise de Dados



Com a biblioteca instalada, o primeiro a ser feito é a importação:

```
import pandas as pd
```

Agora vamos carregar a base de dados na memória. Para isso use o método **read_csv()** do Pandas, veja:

```
1 dataset = pd.read_csv('/home/rodrigo/scripts/minerandodados/kc_house_data.csv', sep=',')
```

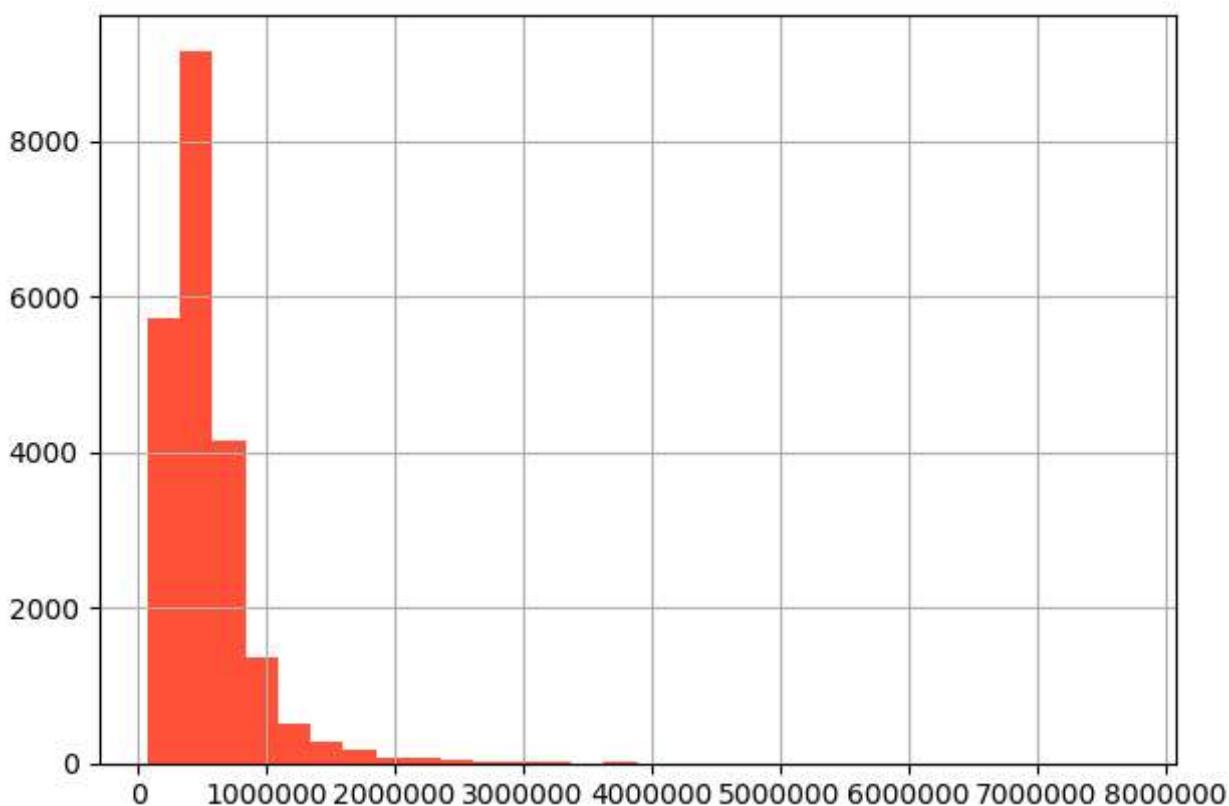
No comando acima criamos uma variável chamada “dataset”, esta recebe o caminho do arquivo e em seguida usamos o parâmetro **sep=”,** para dizer que o separador de colunas da nossa base é a vírgula.

A variável **dataset** agora é um objeto do tipo dataframe.

A base de dados contém dados de imóveis, seria interessante ver a distribuição dos preços.

Vamos plotar um histograma da coluna ‘price’. Veja:

```
1 %matplotlib notebook
2
3 dataset['price'].hist(bins=30)
```



A primeira linha do comando acima, configura o jupyter notebook para exibir o gráfico na célula do notebook.

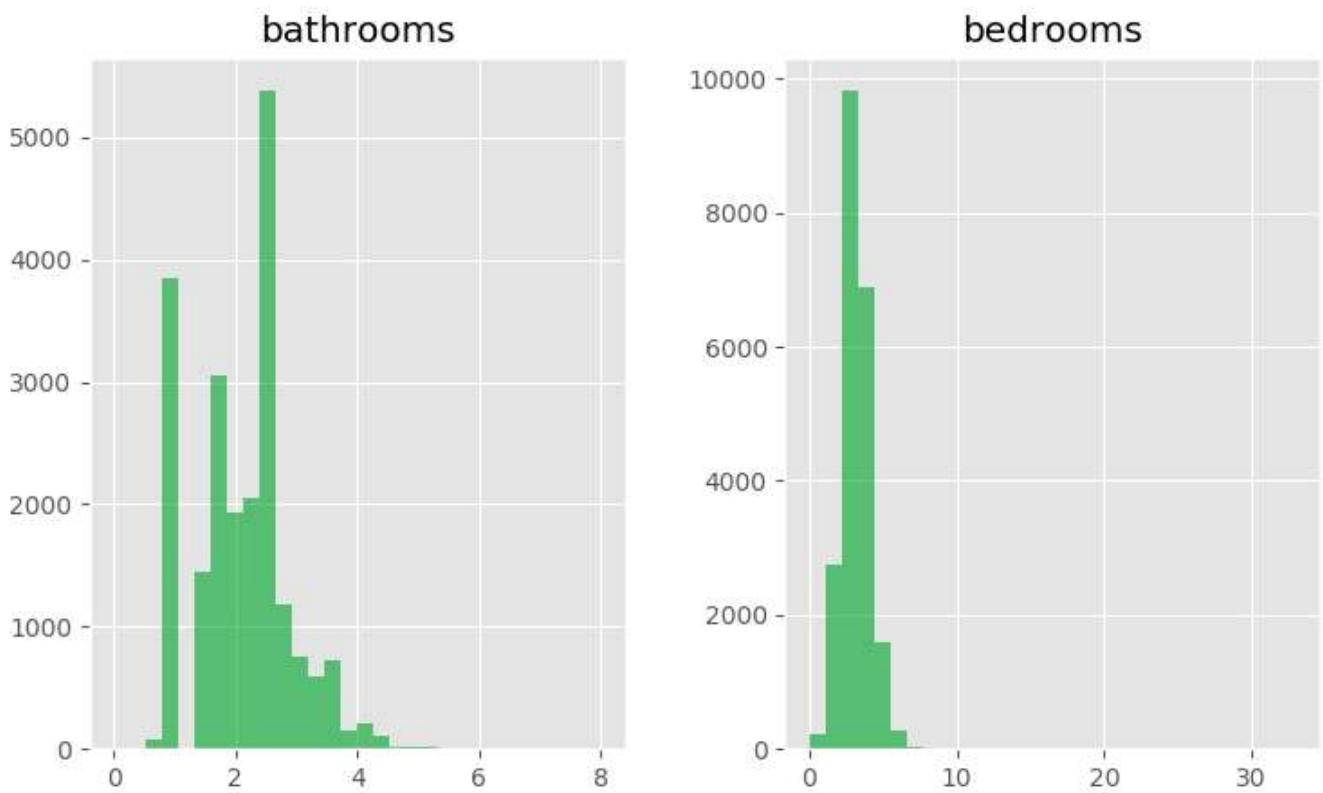
Na segunda linha plotamos um histograma da coluna ‘price’ passando como parâmetro o número de bins e parâmetro **color='red'** para colocar o gráfico em vermelho.

O número de bins é a quantidade de caixas que quero agrupar os dados.

Vimos no gráfico que a grande maioria dos preços são entre 1 e 2 milhões, poucos casos de 3 milhões.

Vamos plotar também um histograma com 2 colunas, e usar alguns parâmetros para customizar o gráfico, veja:

```
1 %matplotlib notebook
2
3 dataset[['bedrooms', 'bathrooms']].hist(bins=30, alpha=0.5, color='Green')
```



Acima plotamos um histograma para a coluna bathrooms(banheiros) e outro para a coluna bedrooms(quartos).

Nesse gráfico podemos ver a distribuição dos números de banheiros dos imóveis (bathrooms).

Fica claro no gráfico que a maioria dos imóveis possuem de 1 a 3 banheiros.

O parâmetro **alpha** permite controlar a opacidade da cor.

Pode ser interessante diminuir esse valor, caso queira que as linhas do fundo do gráfico se sobressaem.

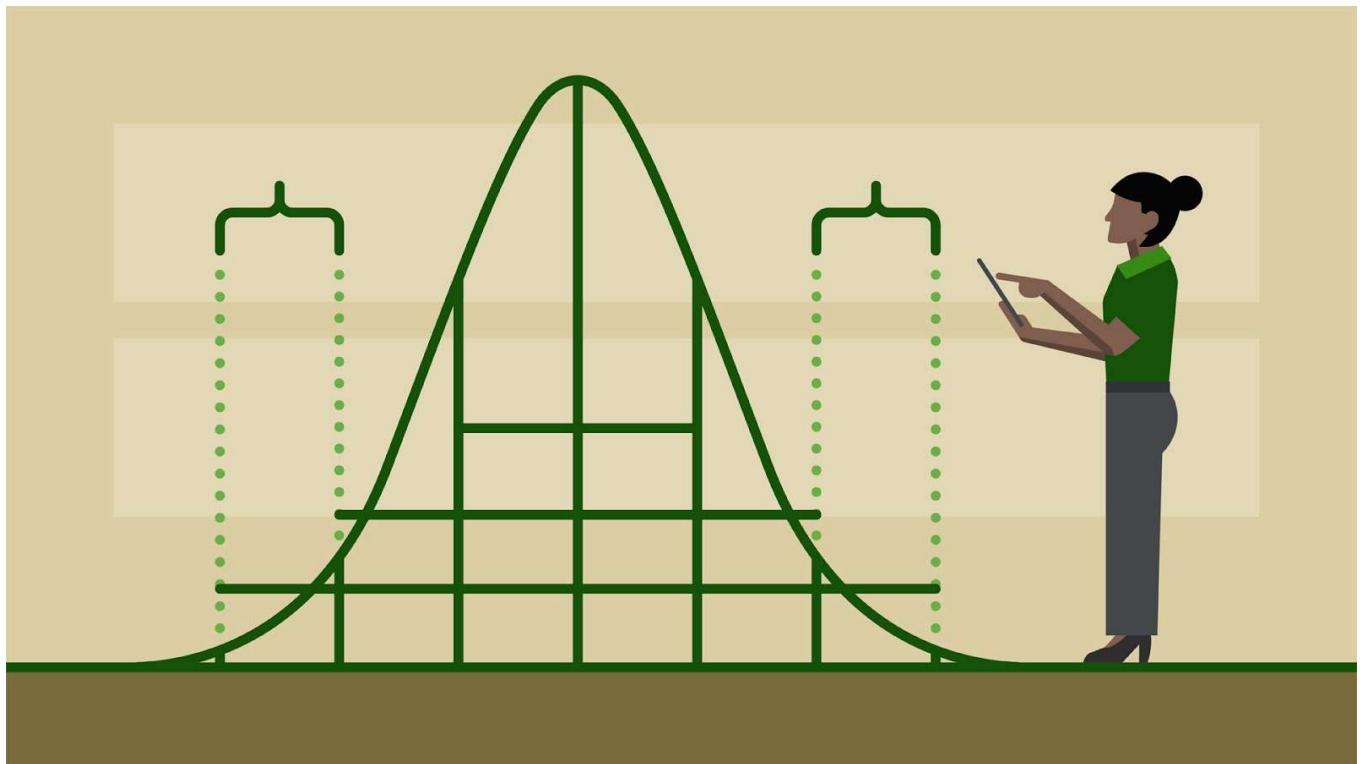
O parâmetro **color** define a cor das barras do gráfico para verde (Green).

Coloque o seu e-mail abaixo para receber gratuitamente as atualizações do blog!

Seu melhor email

QUERO RECEBER! 

Estatística Descritiva



Na tarefa de análise de dados, algo essencial é obter informações estatísticas dos dados.

O pandas facilita muito esse trabalho veja alguns exemplos:

E se quisermos saber qual o número médio de quartos têm os imóveis da base?

```
1 dataset['bedrooms'].mean()
```

In [59]: `dataset['bedrooms'].mean()`

Out[59]: 3.3709102688694523

O método **mean()** (média) nos dá essa informação.

E no caso do número máximo e mínimo de quartos? Ou seja, qual o maior valor da coluna bedrooms?

Use o método **max()** e **min()**

```
1 dataset['bedrooms'].max()
```

```
1 dataset['bedrooms'].min()
```

In [33]: `dataset['bedrooms'].max()`

Out[33]: 33.0

In [34]: `dataset['bedrooms'].min()`

Out[34]: 0.0

O método **std()** calcula o desvio padrão da coluna.

O desvio padrão é uma medida interessante para ver a variabilidade dos dados.

O exemplo abaixo calcula o desvio padrão para a coluna bedrooms:

In [38]: `dataset['bedrooms'].std()`

Out[38]: 0.9300844679399579

Com o desvio padrão de **0.93** podemos dizer que nossos dados estão com uma variação de 0.93 para **cima** e para **abaixo da média**.

É interessante avaliar como está a distribuição dos nossos dados.

Para isso, existem métodos que medem se os dados estão simétricos.

O Pandas possui métodos para calcular a simetria dos dados veja o método **skew()**:

In [61]: `dataset.skew()`

Out[61]:	id	0.243329
	price	4.024069
	bedrooms	1.974439
	bathrooms	0.511108
	sqft_living	1.471555
	sqft_lot	13.060019
	floors	0.616107
	waterfront	11.385108
	view	3.395750
	condition	1.032805
	grade	0.771103
	sqft_above	1.446664
	sqft_basement	1.577965
	yr_builtin	-0.469805
	yr_renovated	4.549493
	zipcode	0.405661
	lat	-0.485270
	long	0.885053
	sqft_living15	1.108181
	sqft_lot15	9.506743
	dtype:	float64

Esse método retorna o valor de **simetria** de cada coluna do dataset.

Um valor zero indica uma distribuição simétrica, um valor maior que zero ou menor indica uma distribuição assimétrica.

Podemos ver algumas informações interessantes, vejam que algumas colunas tem um valor muito acima de 0 e outras um pouco abaixo.

Valores acima de zero podemos dizer que existe uma **assimetria positiva** e valores abaixo de zero uma **assimetria negativa**.

Isso quer dizer que valores muito acima de zero indicam que existem mais valores acima da média e valores abaixo de zero significa que contém mais valores abaixo da média.

Se quiser várias medidas juntas use o comando **describe()**.

Já falei desse método antes, mas nunca é demais, ele é extremamente útil.

Nesse exemplo usamos o describe em uma coluna:

```
In [47]: dataset['bedrooms'].describe()  
Out[47]: count    21609.000000  
          mean      3.370910  
          std       0.930084  
          min       0.000000  
          25%      3.000000  
          50%      3.000000  
          75%      4.000000  
          max      33.000000  
          Name: bedrooms, dtype: float64
```

Extraindo Insights



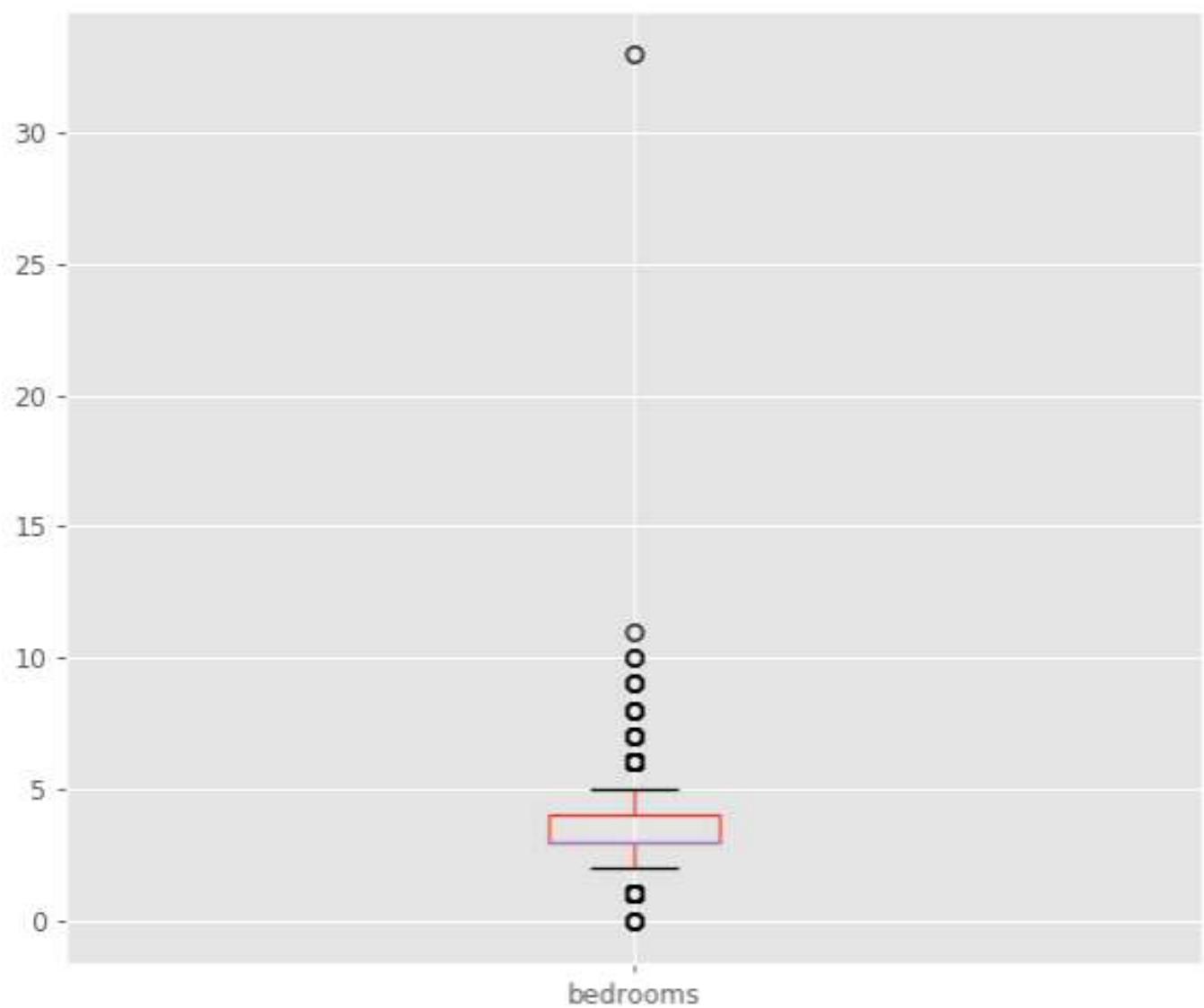
Uma forma de explorar os dados em busca de obter conhecimento é investir na visualização de dados. Dominando o Pandas

Nesse artigo mostramos como plotar os gráficos certos visando obter insights.

O Pandas nos permite de forma rápida e fácil plotar um gráfico do tipo BoxPlot.

Esse gráfico nos mostra diversas informações relevantes, veja:

```
1 %matplotlib notebook
2
3 matplotlib.style.use('ggplot')
4 dataset.boxplot(column='bedrooms')
```



No código acima plotamos o gráfico de Box para a coluna ‘bedrooms’.

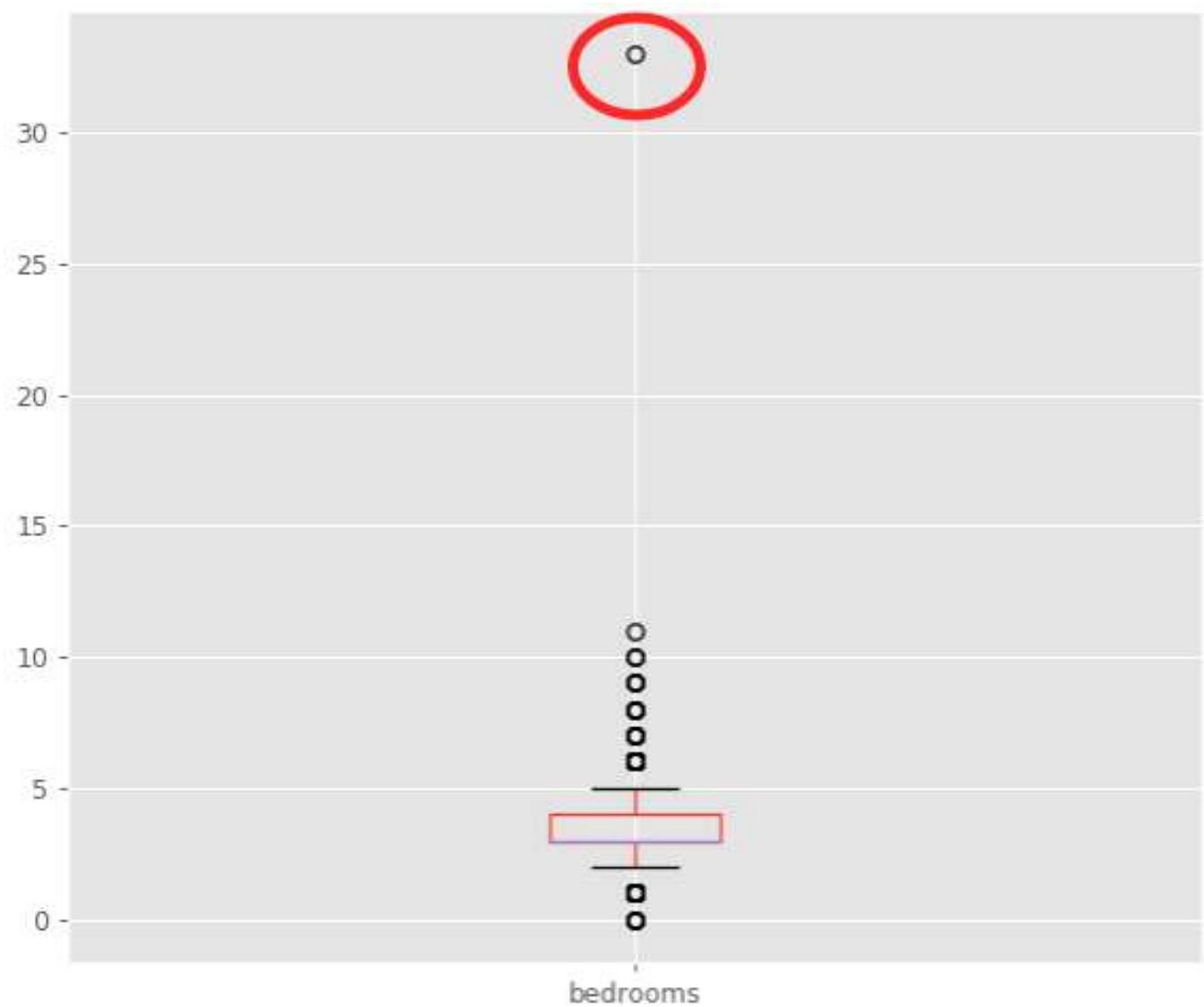
O que esse gráfico nos mostra? Dominando o Pandas

Esse gráfico mostra várias informações relevantes, como: o valor médio dos dados, o valor máximo e mínimo do conjunto de dados e os **outliers**.

As linhas superior e inferior extra ao retângulo representam o valor máximo e mínimo dos dados.

Os pontos fora das linhas superior e inferior são chamados de outliers, pois, se distanciam da média dos valores.

Podemos ver um ponto que se distancia bastante, este seria um outlier, veja:



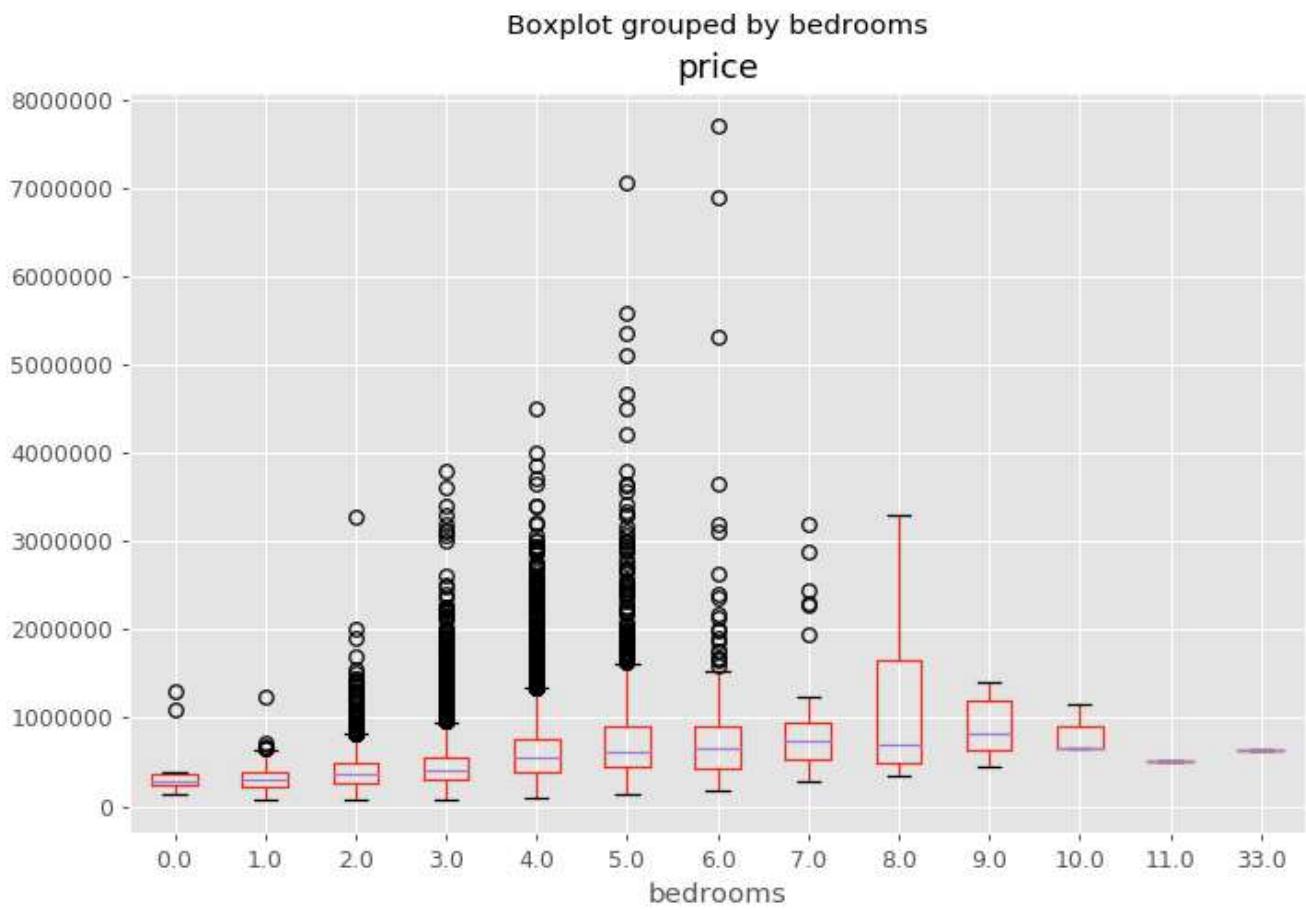
Com o BoxPlot conseguimos visualizar de forma rápida a distribuição dos dados através da dimensão do retângulo e visualizar os outliers representadas pelos valores pontos distantes.

Obs: A linha `matplotlib.style.use('ggplot')` usei para mudar o estilo do gráfico. Isso apenas muda o layout do gráfico, mudando as cores e o fundo.

Se quiser conhecer os estilos disponíveis use o comando: `matplotlib.style.available`.

Podemos também plotar um gráfico do tipo Box de uma coluna agrupado por outra coluna, veja:

```
1 %matplotlib notebook
2
3 dataset.boxplot(column='price', by='bedrooms')
```



No comando acima plotamos a coluna ‘bedrooms’ agrupada pela coluna ‘price’.

Dessa forma podemos ver os imóveis pela quantidade de quartos e seus preços.

Bem interessante né?

Coloque o seu e-mail abaixo para receber gratuitamente as atualizações do blog!

Seu melhor email

QUERO RECEBER!

Correlação de Variáveis

Outro recurso interessante para fazer análise de dados é visualizar a correlação entre as variáveis. Dominando o Pandas

Será que existe uma correlação entre o número de banheiros do imóvel e o seu preço?

Ou, o número de quartos e o preço? Dominando o Pandas

Veja que com o método **corr()** do pandas podemos calcular a correlação entre todas as colunas do dataset.

In [6]:	dataset.corr()														
Out[6]:		id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	
	id	1.000000	-0.016762	0.001235	0.005160	-0.012258	-0.132109	0.018595	-0.002721	0.011592	-0.023783	0.008130	-0.010842	-0.005	
	price	-0.016762	1.000000	0.308321	0.525138	0.702035	0.089661	0.256791	0.266369	0.397293	0.036362	0.667434	0.605567	0.323	
	bedrooms	0.001235	0.308321	1.000000	0.515929	0.576679	0.031684	0.175440	-0.006589	0.079515	0.028534	0.356972	0.477618	0.303	
	bathrooms	0.005160	0.525138	0.515929	1.000000	0.754665	0.087740	0.500626	0.063744	0.187737	-0.124982	0.664983	0.685342	0.283	
	sqft_living	-0.012258	0.702035	0.576679	0.754665	1.000000	0.172826	0.353922	0.103818	0.284611	-0.058753	0.762704	0.876597	0.435	
	sqft_lot	-0.132109	0.089661	0.031684	0.087740	0.172826	1.000000	-0.005210	0.021604	0.074710	-0.008958	0.113621	0.183512	0.015	
	floors	0.018595	0.256791	0.175440	0.500626	0.353922	-0.005210	1.000000	0.023695	0.029432	-0.263740	0.458171	0.523863	-0.245	
	waterfront	-0.002721	0.266369	-0.006589	0.063744	0.103818	0.021604	0.023695	1.000000	0.401857	0.016653	0.082775	0.072075	0.080	
	view	0.011592	0.397293	0.079515	0.187737	0.284611	0.074710	0.029432	0.401857	1.000000	0.045990	0.251321	0.167649	0.276	
	condition	-0.023783	0.036362	0.028534	-0.124982	-0.058753	-0.008958	-0.263740	0.016653	0.045990	1.000000	-0.144674	-0.158214	0.174	

A correlação que o pandas implementa por padrão é a correlação de **Pearson**.

Este coeficiente assume valores entre -1 e 1, onde um valor 1 significa uma correlação positiva perfeita entre as variáveis e um valor -1 uma correlação negativa perfeita entre as variáveis.

Os valores 0 significa que não há uma correlação entre as variáveis. Dominando o Pandas

Podemos implementar também a correlação de Spearman. A diferença aqui é que essa calcula a correlação **não linear** entre as variáveis.

Para usar passe o parâmetro para o método **corr()**:

In [7]:	dataset.corr('spearman')					
Out[7]:		id	price	bedrooms	bathrooms	sqft_living
	id	1.000000	0.004178	0.006185	0.015051	0.001656
	price	0.004178	1.000000	0.344615	0.497160	0.644191
	bedrooms	0.006185	0.344615	1.000000	0.521451	0.647395

Vamos investigar melhor a correlação das colunas com o preço.

Veja que algumas colunas tem uma alta correlação com preço.

Isso quer dizer que essas colunas impactam muito no preço dos imóveis, veja:

```
In [18]: dataset[['bedrooms', 'bathrooms', 'sqft_living', 'floors', 'waterfront', 'grade', 'price']].corr()
```

Out[18]:

	bedrooms	bathrooms	sqft_living	floors	waterfront	grade	price
bedrooms	1.000000	0.515929	0.576679	0.175440	-0.006589	0.356972	0.308321
bathrooms	0.515929	1.000000	0.754665	0.500626	0.063744	0.664983	0.525138
sqft_living	0.576679	0.754665	1.000000	0.353922	0.103818	0.762704	0.702035
floors	0.175440	0.500626	0.353922	1.000000	0.023695	0.458171	0.256791
waterfront	-0.006589	0.063744	0.103818	0.023695	1.000000	0.082775	0.266369
grade	0.356972	0.664983	0.762704	0.458171	0.082775	1.000000	0.667434
price	0.308321	0.525138	0.702035	0.256791	0.266369	0.667434	1.000000

Repare que as colunas ‘sqft_living’, e ‘grade’ tiveram uma correlação positiva com a variável ‘price’.

Já a coluna ‘waterfront’ teve uma baixa correlação. Dominando o Pandas

O que essas colunas significam?

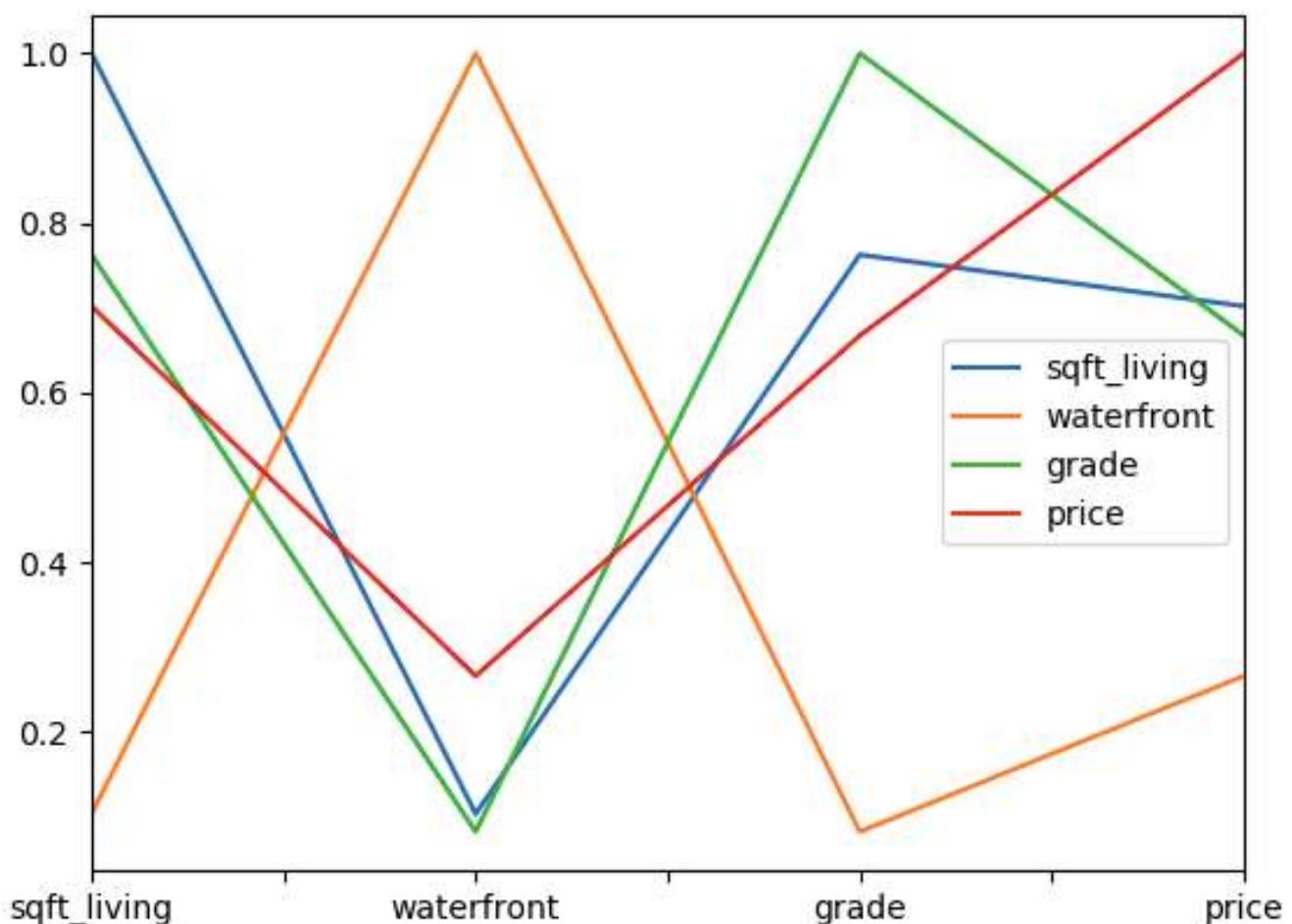
- **sqft_living**: Quantidade de metros quadrados do interior do imóvel.
- **waterfront**: Imóvel tem vista para o mar ou não.
- **grade**: Um índice de qualidade do design e construção do imóvel, onde valores mais altos significam imóveis com qualidade de construção superior.

Interessante aqui é que se o imóvel tem uma vista para o mar, isso não tem tanta correlação com o preço. 😊

Me dê imagens..

Uma imagem vale mais que mil palavras. Vamos plotar essas correlações.

```
1 %matplotlib notebook
2
3 dataset[['sqft_living', 'waterfront', 'grade', 'price']].corr().plot()
```



É possível ver a relação das linhas com a variável ‘price’.

Veja que a linha **verde** (grade) e **azul** (sqft_living) quando próximas a variável ‘**price**’ se mantém acima de **0.6** enquanto a linha **laranja** (waterfront) se mantém bem abaixo.

Para plotar correlação existem outras bibliotecas que nos dão uma visão bem melhor, como a **seaborn**. Porém, para uma rápida visualização o pandas já atende bem.

Coloque o seu e-mail abaixo para receber gratuitamente as atualizações do blog!

Seu melhor email

QUERO RECEBER!

Tabelas Pivot

As tabelas pivot são úteis para fazer agrupamentos nos dados. [Dominando o Pandas](#)

Essa funcionalidade é bastante eficiente e nos dar uma enorme vantagem em termos de tempo e desempenho computacional.

Imagine fazer agrupamento de bases de dados muito grande na mão?

Por exemplo, se quisermos contar a quantidade de imóveis agrupados pelos que têm e não tem visão para o mar e ainda pelo número de andares?

Veja como isso pode ser fácil:

```
1 dataset.pivot_table('id',index=["waterfront","floors"],aggfunc='count',margins=True)
```

id		
waterfront	floors	
0	1.0	10622.0
	1.5	1889.0
	2.0	8166.0
	2.5	159.0
	3.0	605.0
	3.5	8.0
1	1.0	57.0
	1.5	21.0
	2.0	75.0
	2.5	2.0
	3.0	8.0
All		21612.0

Com esse sumário dos dados ficou claro que a grande maioria dos imóveis não tem visão para o mar e são imóveis de **1 e 2 andares**. 😊

O método **pivot_table()** recebe o primeiro parâmetro “id” que a coluna será o resultado da contagem dos dados agrupados.

O parâmetro “**index**” são as colunas que quero agrupar.

O parâmetro “**aggfunc**” uso para definir a função de agregação, no meu caso fiz a contagem dos dados, mas é possível fazer soma, calcular a média entre outras operações.

E por fim, o último parâmetro “**margins**” definido como “true” para imprimir o total de registros na última linha.

Crosstab, Wtf??

Na tradução livre essa funcionalidade seria chamada Tabulação Cruzada. Dominando o Pandas

Esse recurso é bem interessante, principalmente quando temos uma variável resposta ou classe.

Basicamente o que essa funcionalidade faz é cruzar os valores das variáveis.

Veja como o Pandas implementa:

Queremos cruzar as colunas ‘bedrooms’ e ‘condition’ veja:

```
1 pd.crosstab(dataset['bedrooms'], dataset['condition'])
```

Out[80]:

condition	1	2	3	4	5	
bedrooms	0.0	1	1	10	1	0
0.0	1	1	10	1	0	
1.0	4	11	124	48	12	
2.0	12	51	1779	717	200	
3.0	8	69	6306	2711	728	
4.0	4	36	4579	1682	580	
5.0	0	1	1031	418	151	
6.0	1	3	158	87	23	
7.0	0	0	25	9	4	
8.0	0	0	8	3	2	
9.0	0	0	6	0	0	
10.0	0	0	1	2	0	
11.0	0	0	1	0	0	
33.0	0	0	0	0	1	

O código acima é bem simples, apenas passei as colunas ‘bedrooms’ e ‘condition’ como parâmetros para o método **crosstab()**

A coluna ‘condition’ significa um índice entre 1 a 5 da condição do imóvel.

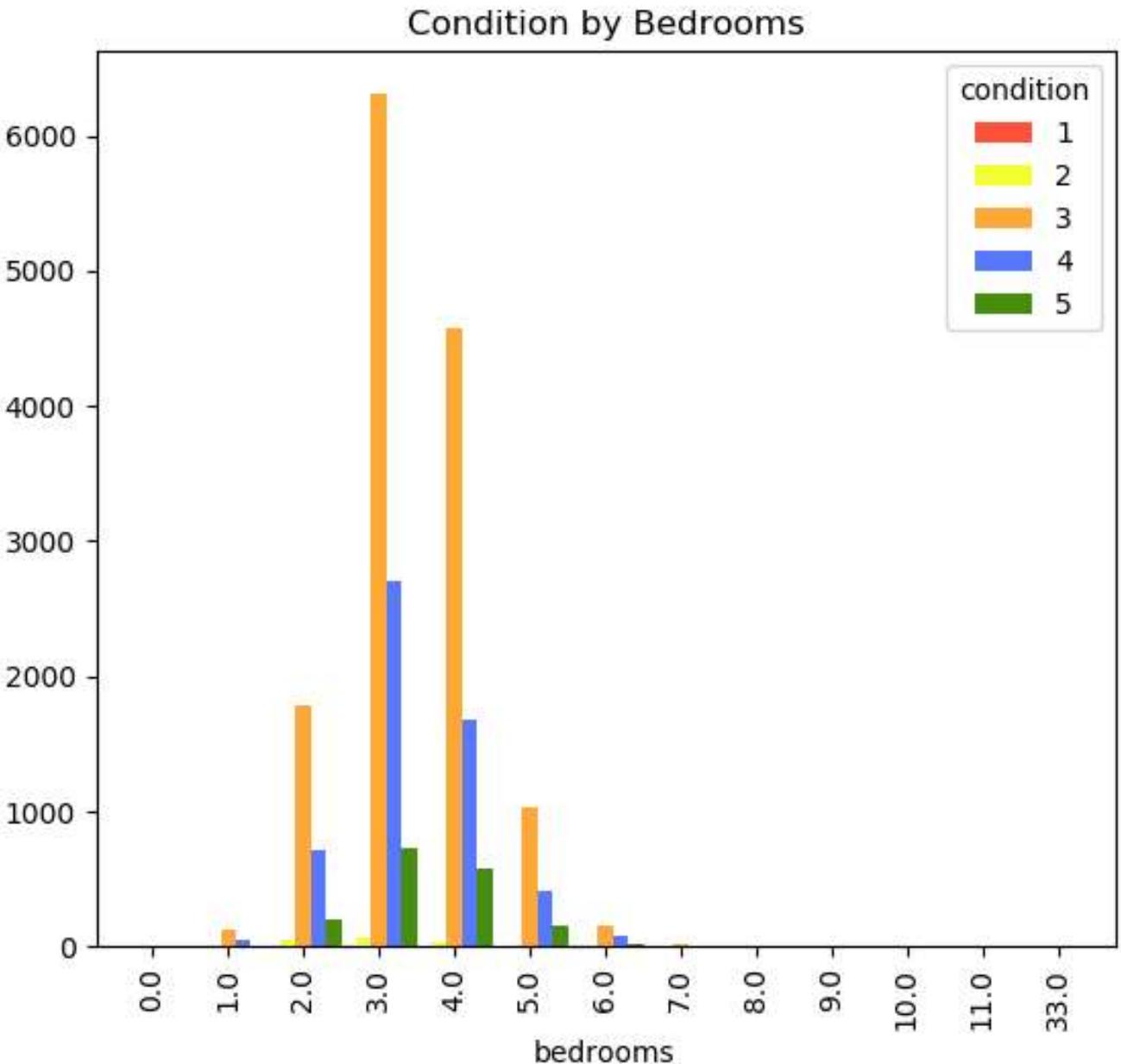
Com o cruzamento dessas colunas, podemos ver por exemplo, qual a distribuição dos imóveis por número de quartos (bedrooms) com relação suas condições (condition).

Analizando a tabela acima vemos que o **maior** número de imóveis na **condição 5** são os imóveis que possuem **3 quartos**.

Simples e útil hein?

Agora vamos visualizar esses dados em um gráfico:

```
1 table = pd.crosstab(dataset['bedrooms'], dataset['condition'])
2 table.plot(kind='bar', width=1.0, color=['red', 'yellow', 'orange', 'blue', 'green'], title='Condition by Bedrooms')
```



Pronto! Agora ficou bem mais elegante 😊

O código acima é bem simples, para simplificar coloquei dentro do objeto ‘table’ a tabulação cruzada.

Em seguida, chamei o método plot passando o parâmetro “**kind**” com o valor ‘**bar**’ para informar que é um gráfico de barras.

O parâmetro ‘**width**’ define a largura das barras, o parâmetro ‘**color**’ recebe uma lista com as legendas e cores dos valores do eixo x (barras).

Em seguida usei o parâmetro ‘**title**’ para definir um título superior para o gráfico e por fim o parâmetro ‘**grid**’ igual a False para remover as linhas do fundo do gráfico.

Coloque o seu e-mail abaixo para receber gratuitamente as atualizações do blog!

Seu melhor email

QUERO RECEBER! 

Trabalhando com Excel



Não poderíamos deixar de falar do nosso queridíssimo Excel.

Mas o que isso tem haver com Pandas ?

O fato é que o Pandas interage muito bem com esse cara e isso para nós é ótimo, concorda?

Veja o quanto isso pode ser interessante, existem muitas organizações que trabalham ativamente com Excel, então não se assuste se um dia ter que importar dados das planilhas mágicas 😊

Chega de conversinha, vamos ao que importa.

Como ler uma carregar uma planilha do Excel ? simples, use o método **read_excel()**.

Esse método permite ler uma planilha do disco, dessa forma criamos um Dataframe do pandas, veja:

```
1 dataframe_excel = pd.read_excel('/home/rodrigo/Downloads/Controle-de-Atividades-2.0.xlsx',
```

Se você quiser fazer download da planilha que usei [aqui](#) está.

O método **read_excel** recebe no primeiro parâmetro o nome do arquivo, o parâmetro “**sheetname**” informo que quero carregar a primeira aba da planilha. Doinando o Pandas

O parâmetro **header** recebe o número da linha na qual será o cabeçalho da planilha, ou seja, aqui informamos em qual linha o pandas deve encontrar nossas colunas.

Como na primeira linha da planilha temos um título, disse ao pandas que o cabeçalho está na segunda linha, linha 1, pois, o número de linhas inicia com 0.

Veja a planilha:

	A	B	C	D	E	F	G
1	PLANILHA DE CONTROLE DE ATIVIDADES						
2	Nº	Atividade	Responsável	Estado Atual	Início	Previsão	Término
4	1	Elaborar relatório de vendas	Antonio	ENCERRADA	1/3/2015 12:20	1/3/2015 18:30	5/3/2015 07:30
5	2	Encerrar Balanço	Antonio	INICIADA	1/3/2015 07:00	8/3/2015 09:00	8/3/2015 09:00
6	3	Preparar novo Plano de Contas	Pedro	ADIADA	1/3/2015 00:00	20/3/2015 00:00	
7	4	Preparar reunião de resultados	Márcia	ENCERRADA	2/3/2015 00:00	10/3/2015 00:00	

Vejamos o dataframe gerado:

```
1 dataframe_excel.head()
```

In [32]: `dataframe_excel.head()`

Out[32]:

	Nº	Atividade	Responsável	Estado Atual	Inicio	Previsão	Término	Duração Prev.	Duração Real	SITUAÇÃO	Unnamed: 10
0	NaN	NaN	NaN	NaN	NaT	NaT	NaT	NaN	NaT	NaN	NaN
1	1.0	Elaborar relatório de vendas	Antonio	ENCERRADA	2015-03-01 12:20:00	2015-03-01 18:30:00	2015-03-05 07:30:00	06:10:00	1900-01-03 19:10:00	ATRASADA	NaN
2	2.0	Encerrar Balanço	Antonio	INICIADA	2015-03-01 07:00:00	2015-03-08 09:00:00	2015-03-08 09:00:00	02:00:00	1900-01-07 02:00:00	EM DIA	NaN
3	3.0	Preparar novo Plano de Contas	Pedro	ADIADA	2015-03-01 00:00:00	2015-03-20 00:00:00	NaT	1900-01-19 00:00:00	NaT	ADIANADA	NaN
4	NaN	NaN	NaN	NaN	NaT	NaT	NaT	NaN	NaT	NaN	NaN

O pandas carregou a planilha corretamente, apenas a linha 0 após o cabeçalho está vazia.

Isso aconteceu pois, as linhas do cabeçalho são duplas (estão mescladas).

Essa é uma planilha simples, mas já dá pra mostrar o poder desse método.

Indexando o Dataset

Outro parâmetro interessante é o `index_col()`.

Com esse parâmetro podemos informar qual a coluna que será usada para indexar o Dataframe, veja um exemplo:

Vou indexar o Dataframe pela coluna “Estado Atual”, assim posso ordenar os resultados pelo estado atual de cada tarefa:

```
1 file = '/home/rodrigo/Downloads/Controle-de-Atividades-2.0.xlsx'
2 dataframe_excel = pd.read_excel(file, sheetname=0, header=1, index_col=3)
```

```
1 dataframe_excel.head()
```

In [40]: `dataframe_excel.head()`

Out[40]:

Nº	Atividade	Responsável	Ínicio	Previsão	Término	Duração Prev.	Duração Real	SITUAÇÃO
Estado Atual								
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
ENCERRADA	1.0	Elaborar relatório de vendas	Antonio	2015-03-01 12:20:00	2015-03-01 18:30:00	2015-03-05 07:30:00	06:10:00	1900-01-03 19:10:00
INICIADA	2.0	Encerrar Balanço	Antonio	2015-03-01 07:00:00	2015-03-08 09:00:00	2015-03-08 09:00:00	02:00:00	1900-01-07 02:00:00
ADIADA	3.0	Preparar novo Plano de Contas	Pedro	2015-03-01 00:00:00	2015-03-20 00:00:00	NaT	1900-01-19 00:00:00	NaT ADIANTADA
ENCERRADA	4.0	Preparar reunião de	Márcia	2015-03-02 00:00:00	2015-03-10 00:00:00	NaT	1900-01-08 00:00:00	NaT ADIANTADA

O parâmetro `index_col` recebe o número da coluna que queremos indexar, nesse caso quero a coluna “**Estado Atual**” na posição 3 (as posições das colunas iniciam com 0).

Com o Dataframe indexado pela coluna “Estado Atual” vou ordená-lo:

```
1 dataframe_excel.sort_index()
```

In [41]: `dataframe_excel.sort_index()`

Out[41]:

Nº	Atividade	Responsável	Inicio	Previsão	Término	Duração Prev.	Duração Real	SITUAÇÃO
Estado Atual								
ADIADA	Preparar novo Plano de Contas	Pedro	2015-03-01 00:00:00	2015-03-20 00:00:00	NaT	1900-01-19 00:00:00	NaT	ADIANTEADA
ENCERRADA	Elaborar relatório de vendas	Antonio	2015-03-01 12:20:00	2015-03-01 18:30:00	2015-03-05 07:30:00	1900-01-03 19:10:00		ATRASADA
ENCERRADA	Preparar reunião de resultados	Márcia	2015-03-02 00:00:00	2015-03-10 00:00:00	NaT	1900-01-08 00:00:00	NaT	ADIANTEADA
INICIADA	Encerrar Balanço	Antonio	2015-03-01 07:00:00	2015-03-08 09:00:00	2015-03-08 09:00:00	1900-01-07 02:00:00	1900-01-07 02:00:00	EM DIA

Gerando planilhas

Além de ler uma planilha do Excel o Pandas tem o método `to_excel()` responsável por gerar um arquivo do tipo xlsx. Dominando o Pandas

Vamos gerar um arquivo a partir do Dataframe que já estávamos trabalhando.

Vou selecionar algumas colunas para ficar mais simples a visualização e gerar uma planilha menor, veja.

```
1 colunas=['id','price','bedrooms','bathrooms','sqft_living','floors','waterfront']
2 dataset[colunas].head()
```

In [43]: `colunas = ['id','price','bedrooms','bathrooms','sqft_living','floors','waterfront']`
`dataset[colunas].head()`

Out[43]:

	id	price	bedrooms	bathrooms	sqft_living	floors	waterfront
0	7129300520	221900.0	3.0	1.00	1180	1.0	0
1	6414100192	538000.0	3.0	2.25	2570	2.0	0
2	5631500400	180000.0	2.0	1.00	770	1.0	0
3	2487200875	604000.0	4.0	3.00	1960	1.0	0
4	1954400510	510000.0	3.0	2.00	1680	1.0	0

Ok, agora é só chamar o método `to_excel` para escrever para o disco o arquivo:

```
1 dataset[colunas].to_excel('planilha_pandas.xls',index=False)
```

Veja uma imagem do arquivo gerado:

	A	B	C	D	E	F	G
1	id	price	bedrooms	bathrooms	sqft_living	floors	waterfront
2	7129300520	221900	3	1	1180	1	0
3	6414100192	538000	3	2,25	2570	2	0
4	5631500400	180000	2	1	770	1	0
5	2487200875	604000	4	3	1960	1	0
6	1954400510	510000	3	2	1680	1	0
7	7237550310	1225000	4	4,5	5420	1	0
8	1321400060	257500	3	2,25	1715	2	0
9	2008000270	291850	3	1,5	1060	1	0
10	2414600126	229500	3	1	1780	1	0
11	3793500160	323000	3	2,5	1890	2	0
12	1736800520	662500	3	2,5	3560	1	0
13	9212900260	468000	2	1	1160		0
14	114101516	310000	3	1	1430	1,5	0
15	6054650070	400000	3	1,75	1370	1	0

Como não passei a localização do arquivo, o pandas gerou no diretório padrão onde se encontra os notebooks.

Pronto a planilha foi criada com sucesso 😊

Consegue perceber o poder dessa funcionalidade?

Sem dúvidas são recursos de extrema importância e que nos da muita produtividade em nossos projetos de Data Science.

Conclusão

Neste artigo vimos como aplicar funcionalidades interessantes do Pandas em uma base de dados real.

Além disso, vimos como extrair insights interessantes dos dados.

Conhecer bem essa biblioteca e saber aplicar seus recursos pode ser um diferencial em projetos de Data Science.

Dominando o Pandas Dominando o Pandas

Se você gostou desse artigo compartilhe com seus amigos e assine nossa lista de e-mail para ficar sempre atualizado sobre novas publicações.

Deixe seu comentário e nos conte o que achou. 😊

Um abraço!

Coloque o seu e-mail abaixo para receber gratuitamente as atualizações do blog!

Seu melhor email

QUERO RECEBER! 

Sobre Rodrigo Santana



Mestrando em Ciência da Computação, interessado em Machine Learning, NLP e Data Science.

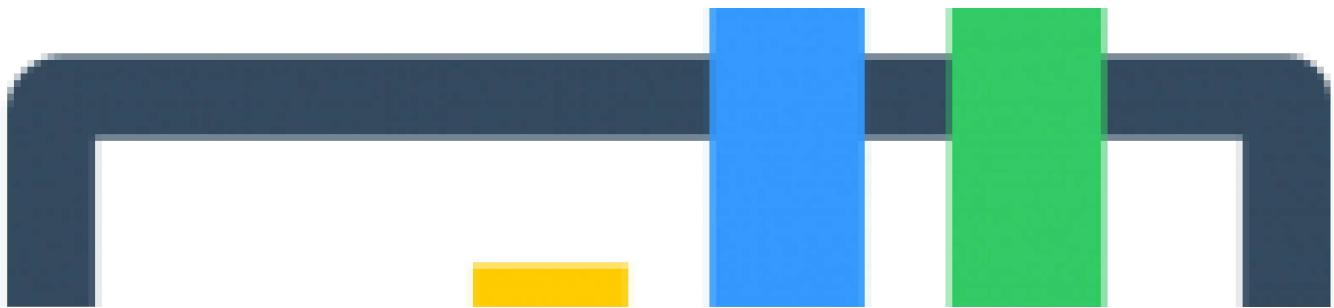
Artigos relacionados



Exploratory Data Analysis (EDA): Aprenda Definitivamente como Extrair Valiosos Insights de Bases de Dados Reais

text mining makes information extraction from huge volume

Mineração de Textos: 7 Técnicas e Aplicações para Você Extrair Valor dos Dados e Alavancar Suas Análises.



7 Tipos de Gráficos que Todo Cientista de Dados Deve Conhecer



One Hot Encoding? Entenda como Funciona com Exemplos em Python



Dominando o Pandas: A Biblioteca para Análise de Dados preferida entre os Cientistas de Dados (Parte 1)



ARTIGO ANTERIOR

[Café Com Código #09: Entendendo Métricas de Avaliação de Modelos](#)



PRÓXIMO ARTIGO

[Café com Código #10: Consultando dados do MongoDB com Python](#)

4 COMENTÁRIOS

www.minerandodados.com.br

 Iniciar sessão ▾

 Recomendar 1

 Partilhar

[Mostrar primeiro os mais votados](#) ▾



Escreva o seu comentário...

INICIE SESSÃO COM O

OU REGISTE-SE NO DISQUS 

Nome



Emerson Carvalho de Souza • há um mês

Rodrigo, excelentes artigos a parte 1 e 2! Foi muito bem escrito, e todos os exemplos me parecem que têm grande aplicação prática. Parabéns!

^ | v • Responder • Partilhar >



Rodrigo Santana Ferreira Moderador → Emerson Carvalho de Souza • há um mês

Opa, obrigado pelo comentário Emerson, fico feliz que gostou dos artigos. Forte Abraco meu caro!

^ | v • Responder • Partilhar >



robson ts • há 3 meses

Só uma dúvida:

Quando você executa a pivot_table de floor e waterfront, como seria possível ter valores "quebrados" de andares(floors) como 1.5, 2.5...?

^ | v • Responder • Partilhar >



Rodrigo Santana Ferreira Moderator → robson ts • há 3 meses

Olá Robson tudo bem?

Esses valores são valores que já contem na base, são valores da própria coluna 'floors'. Pelo que andei pesquisando é algo os valores quebrados também são usados para quantificar o número de andares de um imóvel nos EUA.

Na documentação da base tem apenas a descrição: "Total floors (levels) in house". Você pode consultar aqui: <https://www.kaggle.com/harl...>

Obrigado pelo comentário, um abraço.

^ | v • Responder • Partilhar >

TAMBÉM NO WWW.MINERANDODADOS.COM.BR

Café com Código #17: Manipulando Arrays com Numpy

2 COMENTÁRIOS • há 2 meses



Rodrigo Santana Ferreira — Opa, que bom que gostou Victor.Forte Abraço :)

Algoritmo K-means: Aprenda essa Técnica Essêncial através de

8 COMENTÁRIOS • há um mês



Felipe Santana — Obrigado Antonio, que bom que o artigo te ajudou.Ainda não temos, mas iremos disponibilizar em

Café com Código #13: Visualizando Mapas Interativos com Python

8 COMENTÁRIOS • há 3 meses



Ewerton Silva — Felipe,também tenho que pedir desculpas pela demora...rs...Setor Censitário são micro-regiões

Análise de Sentimentos – Aprenda de uma vez por todas como funciona

26 COMENTÁRIOS • há 7 meses



Leonardo Vilarinho — Muito legal a postagem, só achei o dataset meio difícil por conta do assunto tratado, é difícil

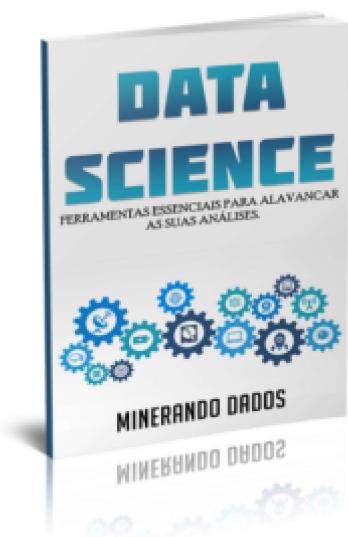


MINERADOR

Eu sou o **Minerador** e vou te ensinar tudo sobre o universo **Data Science** e muito mais...prepare suas ferramentas e mãos ao código!

[SAIBA MAIS](#) +

- ▶ E-book (Gratuito)



- ▶ Café com Código



Machine Learning



Data Analysis



Ferramentas

Buscar por:



Minerando Dados · 2018 © Todos os direitos reservados