# Pandas 的数据读取与输出

Guangyao Zhao

2022-12-01

## Contents



Pandas 提供了丰富的数据类型读取接口，比如 CSV，Excel，JSON 等。在此只介绍前两者。

## 数据读取

`read_csv()` 的参数如下：

```
pandas.read_csv(filepath_or_buffer,  # 文件路径
                sep=_NoDefault.no_default,  # 分隔符，和 delimiter 一样，两者选其一即可
                delimiter=None,
                header='infer',  # 表头，默认为第一行
                names=_NoDefault.no_default, # 如果表头设置为 None，则可自定义表头名称，即 columns 名称
                index_col=None,  # 索引列，默认无
                usecols=None,  # 选择要使用的列
                squeeze=None,
                prefix=_NoDefault.no_default,  # 给表头添加前缀，比如原表头名称为 '1,2,3'；添加前缀 'ORP_'
                mangle_dupe_cols=True,  # 表头名称如果重复，比如有两列 'ORP'，使用该选项后可将其转化为 ' OR
                dtype=None,  # 数据类型
                engine=None,
                converters=None,
                true_values=None,
```

```
                false_values=None,
                skipinitialspace=False,
                skiprows=None,  # 跳过指定汗
                skipfooter=0,
                nrows=None,
                na_values=None,  # 空值替换，若想将 '5, 5.0' 设置为空值，则可使用 na_values=[5, 5.0]
                keep_default_na=True,  # 是否保留 NaN 为空值，此选项和 na_values 共同决定空值有哪些
                na_filter=True,  # 是否检查空值，如果确定文件中无空值，则可将其设置为 False，提高运行速度
                verbose=False,
                skip_blank_lines=True,  # 是否跳过空行
                parse_dates=None,  # 是否对时间日期进行解析
                infer_datetime_format=False,  # 如果开启了 parse_dates 选项，而且 datetime 字符串的格式都
                keep_date_col=False,
                date_parser=None,
                dayfirst=False,
                cache_dates=True,
                iterator=False,
                chunksize=None,
                compression='infer',
                thousands=None,  # 是否设置千分位分隔符
                decimal='.',  # 小数点标识符
                lineterminator=None,  # 行结束标识符，比如 'a,b,c~1,2,3~4,5,6'，将 lineterminator='~'，则读
                quotechar='"',
                quoting=0,
                doublequote=True,
                escapechar=None,
                comment=None,  # 注释标识符，比如 comment='#'，则会被忽视
                encoding=None,
                encoding_errors='strict',
                dialect=None,
                error_bad_lines=None,
                warn_bad_lines=None,
                on_bad_lines=None,
                delim_whitespace=False,  # 空格分隔符，和 delimiter, sep 冲突，相当于 sep='\s+'
                low_memory=True,
                memory_map=False,
```

```
                  float_precision=None,
                  storage_options=None)
```

和 read_csv() 相比，read_excel() 仅仅多了一个 sheet_name() 选项：

```
pandas.read_excel(io,
                  sheet_name=0, # 要读取的 sheet 名称，可以是一个也可以是多个，如果不指定则默认第一个。int 为
                  header=0,
                  names=None,
                  index_col=None,
                  usecols=None,
                  squeeze=None,
                  dtype=None,
                  engine=None,
                  converters=None,
                  true_values=None,
                  false_values=None,
                  skiprows=None,
                  nrows=None,
                  na_values=None,
                  keep_default_na=True,
                  na_filter=True,
                  verbose=False,
                  parse_dates=False,
                  date_parser=None,
                  thousands=None,
                  decimal='.',
                  comment=None,
                  skipfooter=0,
                  convert_float=None,
                  mangle_dupe_cols=True,
                  storage_options=None)
```

可以看到大部分命令和 read_csv 基本一致，但是多了 sheet_name 选项。如果 csv 文件格式能满足需求，则尽量避免使用 excel，因为前者更通用，读取数据快且处理方法更丰富。

## 数据输出

处理后的表格也可以进行相应的输出。to_csv() 的参数如下：

```
DataFrame.to_csv(path_or_buf=None,  # 路径
                 sep=',',  # 分隔符
                 na_rep='', # 空值代替符号
                 float_format=None,
                 columns=None, # 表头名称
                 header=True,  # 是否含有表头，和 columns 结合使用，也可直接在此传入表头名称
                 index=True,
                 index_label=None,
                 mode='w',
                 encoding=None,
                 compression='infer', # 是否压缩，不常用
                 quoting=None,
                 quotechar='"',
                 lineterminator=None,
                 chunksize=None,
                 date_format=None,
                 doublequote=True,
                 escapechar=None,
                 decimal='.',
                 errors='strict',
                 storage_options=None)
```

to_excel() 和 to_csv() 相比，特殊之处在于前者支持多 sheet：

```
with pandas.ExcelWriter('xxx.xlsx') as writer:
    df1.to_excel(writer, sheet_name='name1')
    df2.to_excel(writer, sheet_name='name2')
```