

11-PandaDataframes

August 10, 2020

1 Pandas Dataframes

```
[1]: %matplotlib inline
%config InlineBackend.figure_format = 'retina'
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

pd.set_option("display.max_rows", 8)
plt.rcParams['figure.figsize'] = (9, 6)
```

1.1 Create a DataFrame

```
[2]: dates = pd.date_range('20130101', periods=6)
pd.DataFrame(np.random.randn(6,4), index=dates, columns=list('ABCD'))
```

```
[2]:
```

	A	B	C	D
2013-01-01	1.268803	-1.133772	-0.087526	-0.999730
2013-01-02	0.828125	-0.295122	-0.048230	-0.853039
2013-01-03	0.294834	-1.651778	-0.554721	0.472772
2013-01-04	0.307750	-0.289791	0.855815	1.075326
2013-01-05	-0.228748	1.710366	0.625279	-0.550218
2013-01-06	-1.989388	0.366438	0.342000	1.188267

```
[3]: pd.DataFrame({'A' : 1.,
                    'B' : pd.Timestamp('20130102'),
                    'C' : pd.Series(1,index=list(range(4)),dtype='float32'),
                    'D' : np.arange(4,dtype='int32'),
                    'E' : pd.Categorical(["test","train","test","train"]),
                    'F' : 'foo' })
```

```
[3]:
```

	A	B	C	D	E	F
0	1.0	2013-01-02	1.0	0	test	foo
1	1.0	2013-01-02	1.0	1	train	foo
2	1.0	2013-01-02	1.0	2	test	foo
3	1.0	2013-01-02	1.0	3	train	foo

1.2 Load Data from CSV File

```
[4]: url = "https://www.fun-mooc.fr/c4x/agrocampusouest/40001S03/asset/
↳AnaDo_JeuDonnees_TemperatFrance.csv"
french_cities = pd.read_csv(url, delimiter=";", encoding="latin1", index_col=0)
french_cities
```

```
[4]:
```

	Janv	Févr	Mars	Avri	Mai	Juin	juil	Août	Sept	Octo	Nove	\
Bordeaux	5.6	6.6	10.3	12.8	15.8	19.3	20.9	21.0	18.6	13.8	9.1	
Brest	6.1	5.8	7.8	9.2	11.6	14.4	15.6	16.0	14.7	12.0	9.0	
Clermont	2.6	3.7	7.5	10.3	13.8	17.3	19.4	19.1	16.2	11.2	6.6	
Grenoble	1.5	3.2	7.7	10.6	14.5	17.8	20.1	19.5	16.7	11.4	6.5	
...	
Rennes	4.8	5.3	7.9	10.1	13.1	16.2	17.9	17.8	15.7	11.6	7.8	
Strasbourg	0.4	1.5	5.6	9.8	14.0	17.2	19.0	18.3	15.1	9.5	4.9	
Toulouse	4.7	5.6	9.2	11.6	14.9	18.7	20.9	20.9	18.3	13.3	8.6	
Vichy	2.4	3.4	7.1	9.9	13.6	17.1	19.3	18.8	16.0	11.0	6.6	

	Déce	Lati	Long	Moye	Ampl	Région
Bordeaux	6.2	44.50	-0.34	13.33	15.4	SO
Brest	7.0	48.24	-4.29	10.77	10.2	NO
Clermont	3.6	45.47	3.05	10.94	16.8	SE
Grenoble	2.3	45.10	5.43	10.98	18.6	SE
...
Rennes	5.4	48.05	-1.41	11.13	13.1	NO
Strasbourg	1.3	48.35	7.45	9.72	18.6	NE
Toulouse	5.5	43.36	1.26	12.68	16.2	SO
Vichy	3.4	46.08	3.26	10.72	16.9	SE

[15 rows x 17 columns]

1.3 Viewing Data

```
[5]: french_cities.head()
```

```
[5]:
```

	Janv	Févr	Mars	Avri	Mai	Juin	juil	Août	Sept	Octo	Nove	\
Bordeaux	5.6	6.6	10.3	12.8	15.8	19.3	20.9	21.0	18.6	13.8	9.1	
Brest	6.1	5.8	7.8	9.2	11.6	14.4	15.6	16.0	14.7	12.0	9.0	
Clermont	2.6	3.7	7.5	10.3	13.8	17.3	19.4	19.1	16.2	11.2	6.6	
Grenoble	1.5	3.2	7.7	10.6	14.5	17.8	20.1	19.5	16.7	11.4	6.5	
Lille	2.4	2.9	6.0	8.9	12.4	15.3	17.1	17.1	14.7	10.4	6.1	

	Déce	Lati	Long	Moye	Ampl	Région
Bordeaux	6.2	44.50	-0.34	13.33	15.4	SO
Brest	7.0	48.24	-4.29	10.77	10.2	NO
Clermont	3.6	45.47	3.05	10.94	16.8	SE

Grenoble	2.3	45.10	5.43	10.98	18.6	SE
Lille	3.5	50.38	3.04	9.73	14.7	NE

```
[6]: french_cities.tail()
```

```
[6]:
```

	Janv	Févr	Mars	Avri	Mai	Juin	juil	Août	Sept	Octo	Nove	\
Paris	3.4	4.1	7.6	10.7	14.3	17.5	19.1	18.7	16.0	11.4	7.1	
Rennes	4.8	5.3	7.9	10.1	13.1	16.2	17.9	17.8	15.7	11.6	7.8	
Strasbourg	0.4	1.5	5.6	9.8	14.0	17.2	19.0	18.3	15.1	9.5	4.9	
Toulouse	4.7	5.6	9.2	11.6	14.9	18.7	20.9	20.9	18.3	13.3	8.6	
Vichy	2.4	3.4	7.1	9.9	13.6	17.1	19.3	18.8	16.0	11.0	6.6	

	Déce	Lati	Long	Moye	Ampl	Région
Paris	4.3	48.52	2.20	11.18	15.7	NE
Rennes	5.4	48.05	-1.41	11.13	13.1	NO
Strasbourg	1.3	48.35	7.45	9.72	18.6	NE
Toulouse	5.5	43.36	1.26	12.68	16.2	SO
Vichy	3.4	46.08	3.26	10.72	16.9	SE

1.4 Index

```
[7]: french_cities.index
```

```
[7]: Index(['Bordeaux', 'Brest', 'Clermont', 'Grenoble', 'Lille', 'Lyon',
          'Marseille', 'Montpellier', 'Nantes', 'Nice', 'Paris', 'Rennes',
          'Strasbourg', 'Toulouse', 'Vichy'],
          dtype='object')
```

We can rename an index by setting its name.

```
[8]: french_cities.index.name = "City"
      french_cities.head()
```

```
[8]:
```

	Janv	Févr	Mars	Avri	Mai	Juin	juil	Août	Sept	Octo	Nove	\
City												
Bordeaux	5.6	6.6	10.3	12.8	15.8	19.3	20.9	21.0	18.6	13.8	9.1	
Brest	6.1	5.8	7.8	9.2	11.6	14.4	15.6	16.0	14.7	12.0	9.0	
Clermont	2.6	3.7	7.5	10.3	13.8	17.3	19.4	19.1	16.2	11.2	6.6	
Grenoble	1.5	3.2	7.7	10.6	14.5	17.8	20.1	19.5	16.7	11.4	6.5	
Lille	2.4	2.9	6.0	8.9	12.4	15.3	17.1	17.1	14.7	10.4	6.1	

	Déce	Lati	Long	Moye	Ampl	Région
City						
Bordeaux	6.2	44.50	-0.34	13.33	15.4	SO
Brest	7.0	48.24	-4.29	10.77	10.2	NO
Clermont	3.6	45.47	3.05	10.94	16.8	SE

Grenoble	2.3	45.10	5.43	10.98	18.6	SE
Lille	3.5	50.38	3.04	9.73	14.7	NE

1.4.1 Exercise: Rename DataFrame Months in English

```
[9]: import locale
import calendar

locale.setlocale(locale.LC_ALL, 'en_US')

months = calendar.month_abbr
print(*months)
```

```

      □
↳ -----

Error                                Traceback (most recent call↳
↳ last)

<ipython-input-9-e38d847f0b53> in <module>
      2 import calendar
      3
----> 4 locale.setlocale(locale.LC_ALL, 'en_US')
      5
      6 months = calendar.month_abbr

/usr/share/miniconda3/envs/big-data/lib/python3.8/locale.py in↳
↳ setlocale(category, locale)
    606         # convert to string
    607         locale = normalize(_build_localename(locale))
--> 608     return _setlocale(category, locale)
    609
    610 def resetlocale(category=LC_ALL):

Error: unsupported locale setting
```

```
[10]: french_cities.rename(
        columns={ old : new
                  for old, new in zip(french_cities.columns[:12], months[1:])
                  if old != new },
        inplace=True)
french_cities.columns
```

```

↳ -----
↳

```

```

NameError                                Traceback (most recent call↳
↳last)

```

```

<ipython-input-10-39c4d54b672b> in <module>
    1 french_cities.rename(
    2     columns={ old : new
----> 3         for old, new in zip(french_cities.columns[:12], months[1:
↳)])
    4         if old != new },
    5     inplace=True)

```

NameError: name 'months' is not defined

```
[11]: french_cities.rename(columns={'Moye': 'Mean'}, inplace=True)
```

```
[12]: french_cities
```

```
[12]:
```

	Janv	Févr	Mars	Avri	Mai	Juin	juil	Août	Sept	Octo	Nove	\
City												
Bordeaux	5.6	6.6	10.3	12.8	15.8	19.3	20.9	21.0	18.6	13.8	9.1	
Brest	6.1	5.8	7.8	9.2	11.6	14.4	15.6	16.0	14.7	12.0	9.0	
Clermont	2.6	3.7	7.5	10.3	13.8	17.3	19.4	19.1	16.2	11.2	6.6	
Grenoble	1.5	3.2	7.7	10.6	14.5	17.8	20.1	19.5	16.7	11.4	6.5	
...	
Rennes	4.8	5.3	7.9	10.1	13.1	16.2	17.9	17.8	15.7	11.6	7.8	
Strasbourg	0.4	1.5	5.6	9.8	14.0	17.2	19.0	18.3	15.1	9.5	4.9	
Toulouse	4.7	5.6	9.2	11.6	14.9	18.7	20.9	20.9	18.3	13.3	8.6	
Vichy	2.4	3.4	7.1	9.9	13.6	17.1	19.3	18.8	16.0	11.0	6.6	

	Déce	Lati	Long	Mean	Ampl	Région
City						
Bordeaux	6.2	44.50	-0.34	13.33	15.4	SO
Brest	7.0	48.24	-4.29	10.77	10.2	NO
Clermont	3.6	45.47	3.05	10.94	16.8	SE
Grenoble	2.3	45.10	5.43	10.98	18.6	SE
...	
Rennes	5.4	48.05	-1.41	11.13	13.1	NO
Strasbourg	1.3	48.35	7.45	9.72	18.6	NE
Toulouse	5.5	43.36	1.26	12.68	16.2	SO
Vichy	3.4	46.08	3.26	10.72	16.9	SE

[15 rows x 17 columns]

1.5 From a local or remote HTML file

We can download and extract data about mean sea level stations around the world from the [PSMSL website](http://www.psmsl.org).

```
[13]: # Needs `lxml`, `beautifulSoup4` and `html5lib` python packages
table_list = pd.read_html("http://www.psmsl.org/data/obtaining/")

[14]: # there is 1 table on that page which contains metadata about the stations
      ↳ where
      # sea levels are recorded
local_sea_level_stations = table_list[0]
local_sea_level_stations
```

```
[14]:
```

	Station Name	ID	Lat.	Lon.	GLOSS ID	Country	\
0	BREST	1	48.383	-4.495	242.0	FRA	
1	SWINOUJSCIE	2	53.917	14.233	NaN	POL	
2	SHEERNESS	3	51.446	0.743	NaN	GBR	
3	HOLYHEAD	5	53.314	-4.620	NaN	GBR	
...	
1544	SUVA-B	2356	-18.133	178.428	NaN	FJI	
1545	SYDNEY PORT JACKSON	2358	-33.826	151.259	NaN	AUS	
1546	ARKO	2359	58.484	16.961	NaN	SWE	
1547	UDDEVALLA	2360	58.348	11.895	NaN	SWE	

	Date	Coastline	Station
0	07/08/2019	190	91
1	19/10/2001	110	92
2	06/06/2019	170	101
3	06/06/2019	170	191
...
1544	28/01/2020	742	14
1545	13/06/2019	680	138
1546	12/09/2019	50	112
1547	12/09/2019	50	22

[1548 rows x 9 columns]

1.6 Indexing on DataFrames

```
[15]: french_cities['Lati'] # DF [] accesses columns (Series)
```

```
[15]: City
      Bordeaux      44.50
      Brest         48.24
      Clermont     45.47
      Grenoble     45.10
      ...
      Rennes       48.05
      Strasbourg   48.35
      Toulouse     43.36
      Vichy        46.08
      Name: Lati, Length: 15, dtype: float64
```

.loc and .iloc allow to access individual values, slices or masked selections:

```
[16]: french_cities.loc['Rennes', "Sep"]
```

```

      □
↳ -----

      KeyError                                Traceback (most recent call↳
↳ last)

      /usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
↳ core/indexes/base.py in get_loc(self, key, method, tolerance)
      2888         try:
      -> 2889             return self._engine.get_loc(casted_key)
      2890         except KeyError as err:

      pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

      pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

      pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.
↳ PyObjectHashTable.get_item()

      pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.
↳ PyObjectHashTable.get_item()

      KeyError: 'Sep'
```

The above exception was the direct cause of the following exception:

```
KeyError                                Traceback (most recent call
↳last)

<ipython-input-16-766b5d3b5de4> in <module>
----> 1 french_cities.loc['Rennes', "Sep"]

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
↳core/indexing.py in __getitem__(self, key)
    871             # AttributeError for IntervalTree get_value
    872             pass
--> 873         return self._getitem_tuple(key)
    874     else:
    875         # we by definition only have the 0th axis

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
↳core/indexing.py in _getitem_tuple(self, tup)
   1042     def _getitem_tuple(self, tup: Tuple):
   1043         try:
-> 1044             return self._getitem_lowerdim(tup)
   1045         except IndexingError:
   1046             pass

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
↳core/indexing.py in _getitem_lowerdim(self, tup)
    808             return section
    809             # This is an elided recursive call to iloc/loc
--> 810             return getattr(section, self.name)[new_key]
    811
    812         raise IndexingError("not applicable")

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
↳core/indexing.py in __getitem__(self, key)
    877
    878         maybe_callable = com.apply_if_callable(key, self.obj)
--> 879         return self._getitem_axis(maybe_callable, axis=axis)
    880
    881     def _is_scalar_access(self, key: Tuple):
```



```

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
core/indexing.py in _getitem_axis(self, key, axis)
    1108         # fall thru to straight lookup
    1109         self._validate_key(key, axis)
-> 1110         return self._get_label(key, axis=axis)
    1111
    1112     def _get_slice_axis(self, slice_obj: slice, axis: int):

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
core/indexing.py in _get_label(self, label, axis)
    1057     def _get_label(self, label, axis: int):
    1058         # GH#5667 this will fail if the label is not present in the
-> axis.
-> 1059         return self.obj.xs(label, axis=axis)
    1060
    1061     def _handle_lowerdim_multi_index_axis0(self, tup: Tuple):

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
core/generic.py in xs(self, key, axis, level, drop_level)
    3480         loc, new_index = self.index.get_loc_level(key,
-> drop_level=drop_level)
    3481     else:
-> 3482         loc = self.index.get_loc(key)
    3483
    3484         if isinstance(loc, np.ndarray):

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
core/indexes/base.py in get_loc(self, key, method, tolerance)
    2889         return self._engine.get_loc(casted_key)
    2890     except KeyError as err:
-> 2891         raise KeyError(key) from err
    2892
    2893     if tolerance is not None:

KeyError: 'Sep'

```

```
[17]: french_cities.loc['Rennes', ["Sep", "Dec"]]
```

```

    □
-> -----

```

```

KeyError                                Traceback (most recent call_
↳last)

<ipython-input-17-988685453654> in <module>
----> 1 french_cities.loc['Rennes', ["Sep", "Dec"]]

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
↳core/indexing.py in __getitem__(self, key)
    871             # AttributeError for IntervalTree get_value
    872             pass
--> 873         return self._getitem_tuple(key)
    874     else:
    875         # we by definition only have the 0th axis

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
↳core/indexing.py in _getitem_tuple(self, tup)
    1042     def _getitem_tuple(self, tup: Tuple):
    1043         try:
-> 1044             return self._getitem_lowerdim(tup)
    1045         except IndexingError:
    1046             pass

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
↳core/indexing.py in _getitem_lowerdim(self, tup)
    808             return section
    809             # This is an elided recursive call to iloc/loc
--> 810             return getattr(section, self.name)[new_key]
    811
    812         raise IndexingError("not applicable")

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
↳core/indexing.py in __getitem__(self, key)
    877
    878         maybe_callable = com.apply_if_callable(key, self.obj)
--> 879         return self._getitem_axis(maybe_callable, axis=axis)
    880
    881     def _is_scalar_access(self, key: Tuple):

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
↳core/indexing.py in _getitem_axis(self, key, axis)
    1097             raise ValueError("Cannot index with_
↳multidimensional key")

```

```

1098
-> 1099             return self._getitem_iterable(key, axis=axis)
1100
1101             # nested tuple slicing

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
core/indexing.py in _getitem_iterable(self, key, axis)
1035
1036         # A collection of keys
-> 1037         keyarr, indexer = self._get_listlike_indexer(key, axis,
raise_missing=False)
1038         return self.obj._reindex_with_indexers(
1039             {axis: [keyarr, indexer]}, copy=True, allow_dups=True

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
core/indexing.py in _get_listlike_indexer(self, key, axis, raise_missing)
1252         keyarr, indexer, new_indexer = ax.
_reindex_non_unique(keyarr)
1253
-> 1254         self._validate_read_indexer(keyarr, indexer, axis,
raise_missing=raise_missing)
1255         return keyarr, indexer
1256

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
core/indexing.py in _validate_read_indexer(self, key, indexer, axis,
raise_missing)
1296         if missing == len(indexer):
1297             axis_name = self.obj._get_axis_name(axis)
-> 1298             raise KeyError(f"None of [{key}] are in the
[{axis_name}]")
1299
1300         # We (temporarily) allow for some missing keys with .
loc, except in

KeyError: "None of [Index(['Sep', 'Dec'], dtype='object')] are in the
[index]"

```

```
[18]: french_cities.loc['Rennes', "Sep":"Dec"]
```



```

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
core/indexing.py in _getitem_tuple(self, tup)
1042     def _getitem_tuple(self, tup: Tuple):
1043         try:
-> 1044             return self._getitem_lowerdim(tup)
1045         except IndexingError:
1046             pass

```

```

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
core/indexing.py in _getitem_lowerdim(self, tup)
808         return section
809         # This is an elided recursive call to iloc/loc
--> 810         return getattr(section, self.name)[new_key]
811
812         raise IndexingError("not applicable")

```

```

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
core/indexing.py in __getitem__(self, key)
877
878         maybe_callable = com.apply_if_callable(key, self.obj)
--> 879         return self._getitem_axis(maybe_callable, axis=axis)
880
881     def _is_scalar_access(self, key: Tuple):

```

```

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
core/indexing.py in _getitem_axis(self, key, axis)
1086     if isinstance(key, slice):
1087         self._validate_key(key, axis)
-> 1088         return self._get_slice_axis(key, axis=axis)
1089     elif com.is_bool_indexer(key):
1090         return self._getbool_axis(key, axis=axis)

```

```

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
core/indexing.py in _get_slice_axis(self, slice_obj, axis)
1120
1121         labels = obj._get_axis(axis)
-> 1122         indexer = labels.slice_indexer(
1123             slice_obj.start, slice_obj.stop, slice_obj.step,
-> kind="loc"
1124         )

```

```

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
core/indexes/base.py in slice_indexer(self, start, end, step, kind)
4958         slice(1, 3, None)
4959         """
-> 4960         start_slice, end_slice = self.slice_locs(start, end,
core/step=step, kind=kind)
4961
4962         # return a slice

```

```

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
core/indexes/base.py in slice_locs(self, start, end, step, kind)
5159         start_slice = None
5160         if start is not None:
-> 5161             start_slice = self.get_slice_bound(start, "left", kind)
5162         if start_slice is None:
5163             start_slice = 0

```

```

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
core/indexes/base.py in get_slice_bound(self, label, side, kind)
5081         except ValueError:
5082             # raise the original KeyError
-> 5083             raise err
5084
5085         if isinstance(slc, np.ndarray):

```

```

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
core/indexes/base.py in get_slice_bound(self, label, side, kind)
5075         # we need to look up the label
5076         try:
-> 5077             slc = self.get_loc(label)
5078         except KeyError as err:
5079             try:

```

```

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
core/indexes/base.py in get_loc(self, key, method, tolerance)
2889         return self._engine.get_loc(casted_key)
2890         except KeyError as err:
-> 2891             raise KeyError(key) from err
2892
2893         if tolerance is not None:

```

```
KeyError: 'Sep'
```

1.7 Masking

```
[19]: mask = [True, False] * 6 + 5 * [False]
      print(french_cities.iloc[:, mask])
```

	Janv	Mars	Mai	juil	Sept	Nove
City						
Bordeaux	5.6	10.3	15.8	20.9	18.6	9.1
Brest	6.1	7.8	11.6	15.6	14.7	9.0
Clermont	2.6	7.5	13.8	19.4	16.2	6.6
Grenoble	1.5	7.7	14.5	20.1	16.7	6.5
...
Rennes	4.8	7.9	13.1	17.9	15.7	7.8
Strasbourg	0.4	5.6	14.0	19.0	15.1	4.9
Toulouse	4.7	9.2	14.9	20.9	18.3	8.6
Vichy	2.4	7.1	13.6	19.3	16.0	6.6

[15 rows x 6 columns]

```
[20]: print(french_cities.loc["Rennes", mask])
```

```
Janv    4.8
Mars     7.9
Mai     13.1
juil     17.9
Sept     15.7
Nove     7.8
Name: Rennes, dtype: object
```

1.8 New column

```
[21]: french_cities["std"] = french_cities.iloc[:, :12].std(axis=1)
      french_cities
```

```
[21]:
```

	Janv	Févr	Mars	Avri	Mai	Juin	juil	Août	Sept	Octo	Nove	\
City												
Bordeaux	5.6	6.6	10.3	12.8	15.8	19.3	20.9	21.0	18.6	13.8	9.1	
Brest	6.1	5.8	7.8	9.2	11.6	14.4	15.6	16.0	14.7	12.0	9.0	
Clermont	2.6	3.7	7.5	10.3	13.8	17.3	19.4	19.1	16.2	11.2	6.6	
Grenoble	1.5	3.2	7.7	10.6	14.5	17.8	20.1	19.5	16.7	11.4	6.5	
...	
Rennes	4.8	5.3	7.9	10.1	13.1	16.2	17.9	17.8	15.7	11.6	7.8	
Strasbourg	0.4	1.5	5.6	9.8	14.0	17.2	19.0	18.3	15.1	9.5	4.9	

Toulouse	4.7	5.6	9.2	11.6	14.9	18.7	20.9	20.9	18.3	13.3	8.6
Vichy	2.4	3.4	7.1	9.9	13.6	17.1	19.3	18.8	16.0	11.0	6.6

	Déce	Lati	Long	Mean	Ampl	Région	std
City							
Bordeaux	6.2	44.50	-0.34	13.33	15.4	SO	5.792681
Brest	7.0	48.24	-4.29	10.77	10.2	NO	3.773673
Clermont	3.6	45.47	3.05	10.94	16.8	SE	6.189795
Grenoble	2.3	45.10	5.43	10.98	18.6	SE	6.770771
...
Rennes	5.4	48.05	-1.41	11.13	13.1	NO	4.958800
Strasbourg	1.3	48.35	7.45	9.72	18.6	NE	6.931723
Toulouse	5.5	43.36	1.26	12.68	16.2	SO	6.056977
Vichy	3.4	46.08	3.26	10.72	16.9	SE	6.201148

[15 rows x 18 columns]

```
[22]: french_cities = french_cities.drop("std", axis=1) # remove this new column
```

```
[23]: french_cities
```

```
[23]:
```

	Janv	Févr	Mars	Avri	Mai	Juin	juil	Août	Sept	Octo	Nove	\
City												
Bordeaux	5.6	6.6	10.3	12.8	15.8	19.3	20.9	21.0	18.6	13.8	9.1	
Brest	6.1	5.8	7.8	9.2	11.6	14.4	15.6	16.0	14.7	12.0	9.0	
Clermont	2.6	3.7	7.5	10.3	13.8	17.3	19.4	19.1	16.2	11.2	6.6	
Grenoble	1.5	3.2	7.7	10.6	14.5	17.8	20.1	19.5	16.7	11.4	6.5	
...	
Rennes	4.8	5.3	7.9	10.1	13.1	16.2	17.9	17.8	15.7	11.6	7.8	
Strasbourg	0.4	1.5	5.6	9.8	14.0	17.2	19.0	18.3	15.1	9.5	4.9	
Toulouse	4.7	5.6	9.2	11.6	14.9	18.7	20.9	20.9	18.3	13.3	8.6	
Vichy	2.4	3.4	7.1	9.9	13.6	17.1	19.3	18.8	16.0	11.0	6.6	

	Déce	Lati	Long	Mean	Ampl	Région
City						
Bordeaux	6.2	44.50	-0.34	13.33	15.4	SO
Brest	7.0	48.24	-4.29	10.77	10.2	NO
Clermont	3.6	45.47	3.05	10.94	16.8	SE
Grenoble	2.3	45.10	5.43	10.98	18.6	SE
...
Rennes	5.4	48.05	-1.41	11.13	13.1	NO
Strasbourg	1.3	48.35	7.45	9.72	18.6	NE
Toulouse	5.5	43.36	1.26	12.68	16.2	SO
Vichy	3.4	46.08	3.26	10.72	16.9	SE

[15 rows x 17 columns]

1.9 Modifying a dataframe with multiple indexing

```
[24]: # french_cities['Rennes']['Sep'] = 25 # It does not works and breaks the
      ↪ DataFrame
      french_cities.loc['Rennes']['Sep'] # = 25 is the right way to do it
```

```

      ↪ -----

      KeyError                                Traceback (most recent call
      ↪ last)

      /usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
      ↪ core/indexes/base.py in get_loc(self, key, method, tolerance)
      2888             try:
      -> 2889                 return self._engine.get_loc(casted_key)
      2890             except KeyError as err:

      pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

      pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

      pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.
      ↪ PyObjectHashTable.get_item()

      pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.
      ↪ PyObjectHashTable.get_item()

      KeyError: 'Sep'
```

The above exception was the direct cause of the following exception:

```

      KeyError                                Traceback (most recent call
      ↪ last)

      <ipython-input-24-c68575f89e60> in <module>
          1 # french_cities['Rennes']['Sep'] = 25 # It does not works and breaks
      ↪ the DataFrame
      ----> 2 french_cities.loc['Rennes']['Sep'] # = 25 is the right way to do it
```

```

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
core/series.py in __getitem__(self, key)
    880
    881         elif key_is_scalar:
--> 882             return self._get_value(key)
    883
    884         if (

```

```

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
core/series.py in _get_value(self, label, takeable)
    989
    990         # Similar to Index.get_value, but we do not fall back to_
core/positional
--> 991         loc = self.index.get_loc(label)
    992         return self.index._get_values_for_loc(self, loc, label)
    993

```

```

/usr/share/miniconda3/envs/big-data/lib/python3.8/site-packages/pandas/
core/indexes/base.py in get_loc(self, key, method, tolerance)
    2889         return self._engine.get_loc(casted_key)
    2890         except KeyError as err:
-> 2891             raise KeyError(key) from err
    2892
    2893         if tolerance is not None:

```

KeyError: 'Sep'

[25]: french_cities

```

[25]:
      City  Janv  Févr  Mars  Avri  Mai  Juin  juil  Août  Sept  Octo  Nove  \
Bordeaux   5.6   6.6  10.3  12.8  15.8  19.3  20.9  21.0  18.6  13.8   9.1
Brest      6.1   5.8   7.8   9.2  11.6  14.4  15.6  16.0  14.7  12.0   9.0
Clermont   2.6   3.7   7.5  10.3  13.8  17.3  19.4  19.1  16.2  11.2   6.6
Grenoble   1.5   3.2   7.7  10.6  14.5  17.8  20.1  19.5  16.7  11.4   6.5
...
Rennes     4.8   5.3   7.9  10.1  13.1  16.2  17.9  17.8  15.7  11.6   7.8
Strasbourg 0.4   1.5   5.6   9.8  14.0  17.2  19.0  18.3  15.1   9.5   4.9
Toulouse   4.7   5.6   9.2  11.6  14.9  18.7  20.9  20.9  18.3  13.3   8.6
Vichy      2.4   3.4   7.1   9.9  13.6  17.1  19.3  18.8  16.0  11.0   6.6

```

	Déce	Lati	Long	Mean	Ampl	Région
City						
Bordeaux	6.2	44.50	-0.34	13.33	15.4	SO
Brest	7.0	48.24	-4.29	10.77	10.2	NO
Clermont	3.6	45.47	3.05	10.94	16.8	SE
Grenoble	2.3	45.10	5.43	10.98	18.6	SE
...
Rennes	5.4	48.05	-1.41	11.13	13.1	NO
Strasbourg	1.3	48.35	7.45	9.72	18.6	NE
Toulouse	5.5	43.36	1.26	12.68	16.2	SO
Vichy	3.4	46.08	3.26	10.72	16.9	SE

[15 rows x 7 columns]

1.10 Transforming datasets

```
[26]: french_cities['Mean'].min(), french_cities['Ampl'].max()
```

```
[26]: (9.72, 18.6)
```

1.11 Apply

Let's convert the temperature mean from Celsius to Fahrenheit degree.

```
[27]: fahrenheit = lambda T: T*9/5+32
french_cities['Mean'].apply(fahrenheit)
```

```
[27]: City
Bordeaux      55.994
Brest         51.386
Clermont      51.692
Grenoble      51.764
...
Rennes        52.034
Strasbourg    49.496
Toulouse      54.824
Vichy         51.296
Name: Mean, Length: 15, dtype: float64
```

1.12 Sort

```
[28]: french_cities.sort_values(by='Lati')
```

```
[28]:
```

	Janv	Févr	Mars	Avri	Mai	Juin	juil	Août	Sept	Octo	Nove	\
City												
Marseille	5.5	6.6	10.0	13.0	16.8	20.8	23.3	22.8	19.9	15.0	10.2	
Montpellier	5.6	6.7	9.9	12.8	16.2	20.1	22.7	22.3	19.3	14.6	10.0	
Toulouse	4.7	5.6	9.2	11.6	14.9	18.7	20.9	20.9	18.3	13.3	8.6	
Nice	7.5	8.5	10.8	13.3	16.7	20.1	22.7	22.5	20.3	16.0	11.5	
...	
Brest	6.1	5.8	7.8	9.2	11.6	14.4	15.6	16.0	14.7	12.0	9.0	
Strasbourg	0.4	1.5	5.6	9.8	14.0	17.2	19.0	18.3	15.1	9.5	4.9	
Paris	3.4	4.1	7.6	10.7	14.3	17.5	19.1	18.7	16.0	11.4	7.1	
Lille	2.4	2.9	6.0	8.9	12.4	15.3	17.1	17.1	14.7	10.4	6.1	

	Déce	Lati	Long	Mean	Ampl	Région
City						
Marseille	6.9	43.18	5.24	14.23	17.8	SE
Montpellier	6.5	43.36	3.53	13.89	17.1	SE
Toulouse	5.5	43.36	1.26	12.68	16.2	SO
Nice	8.2	43.42	7.15	14.84	15.2	SE
...
Brest	7.0	48.24	-4.29	10.77	10.2	NO
Strasbourg	1.3	48.35	7.45	9.72	18.6	NE
Paris	4.3	48.52	2.20	11.18	15.7	NE
Lille	3.5	50.38	3.04	9.73	14.7	NE

[15 rows x 17 columns]

```
[29]: french_cities = french_cities.sort_values(by='Lati',ascending=False)
french_cities
```

```
[29]:
```

	Janv	Févr	Mars	Avri	Mai	Juin	juil	Août	Sept	Octo	Nove	\
City												
Lille	2.4	2.9	6.0	8.9	12.4	15.3	17.1	17.1	14.7	10.4	6.1	
Paris	3.4	4.1	7.6	10.7	14.3	17.5	19.1	18.7	16.0	11.4	7.1	
Strasbourg	0.4	1.5	5.6	9.8	14.0	17.2	19.0	18.3	15.1	9.5	4.9	
Brest	6.1	5.8	7.8	9.2	11.6	14.4	15.6	16.0	14.7	12.0	9.0	
...	
Nice	7.5	8.5	10.8	13.3	16.7	20.1	22.7	22.5	20.3	16.0	11.5	
Montpellier	5.6	6.7	9.9	12.8	16.2	20.1	22.7	22.3	19.3	14.6	10.0	
Toulouse	4.7	5.6	9.2	11.6	14.9	18.7	20.9	20.9	18.3	13.3	8.6	
Marseille	5.5	6.6	10.0	13.0	16.8	20.8	23.3	22.8	19.9	15.0	10.2	

	Déce	Lati	Long	Mean	Ampl	Région
City						
Lille	3.5	50.38	3.04	9.73	14.7	NE
Paris	4.3	48.52	2.20	11.18	15.7	NE
Strasbourg	1.3	48.35	7.45	9.72	18.6	NE
Brest	7.0	48.24	-4.29	10.77	10.2	NO

...	
Nice	8.2	43.42	7.15	14.84	15.2		SE
Montpellier	6.5	43.36	3.53	13.89	17.1		SE
Toulouse	5.5	43.36	1.26	12.68	16.2		SO
Marseille	6.9	43.18	5.24	14.23	17.8		SE

[15 rows x 17 columns]

1.13 Stack and unstack

Instead of seeing the months along the axis 1, and the cities along the axis 0, let's try to convert these into an outer and an inner axis along only 1 time dimension.

```
[30]: pd.set_option("display.max_rows", 20)
unstacked = french_cities.iloc[:, :12].unstack()
unstacked
```

```
[30]:      City
Janv  Lille      2.4
      Paris      3.4
      Strasbourg  0.4
      Brest      6.1
      Rennes     4.8
      ...
D  ce  Bordeaux    6.2
      Nice         8.2
      Montpellier  6.5
      Toulouse     5.5
      Marseille   6.9
Length: 180, dtype: float64
```

```
[31]: type(unstacked)
```

```
[31]: pandas.core.series.Series
```

1.14 Transpose

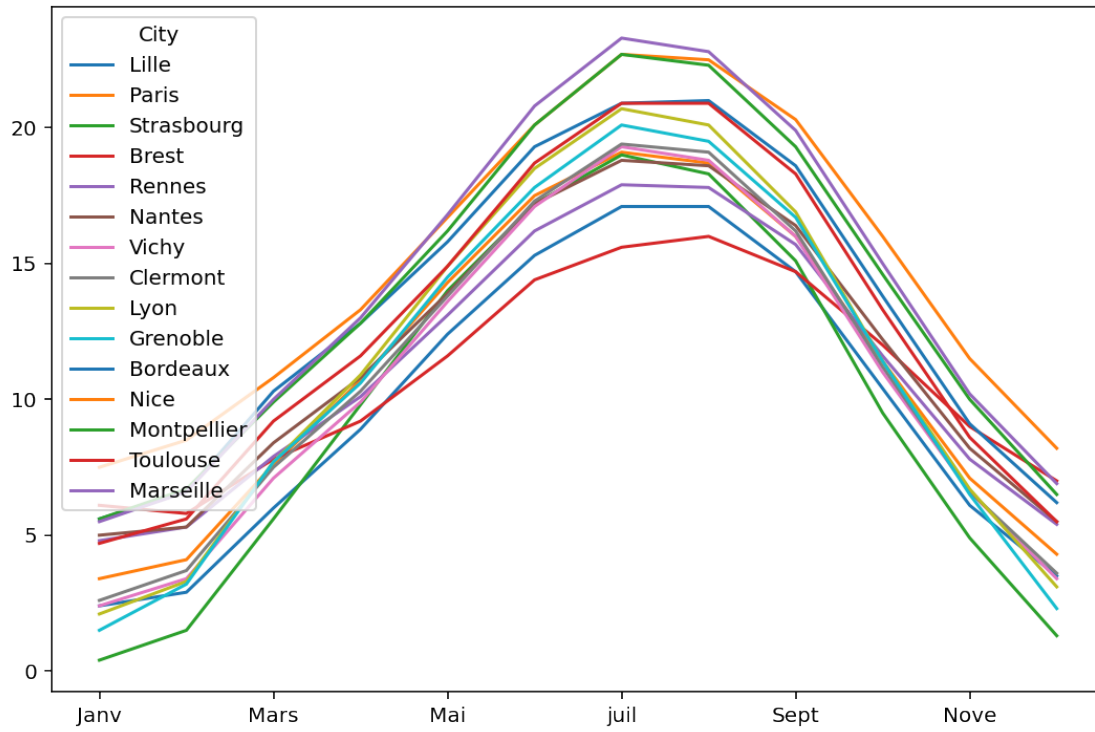
The result is grouped in the wrong order since it sorts first the axis that was unstacked. We need to transpose the dataframe.

```
[32]: city_temp = french_cities.iloc[:, :12].transpose()
city_temp.plot()
```

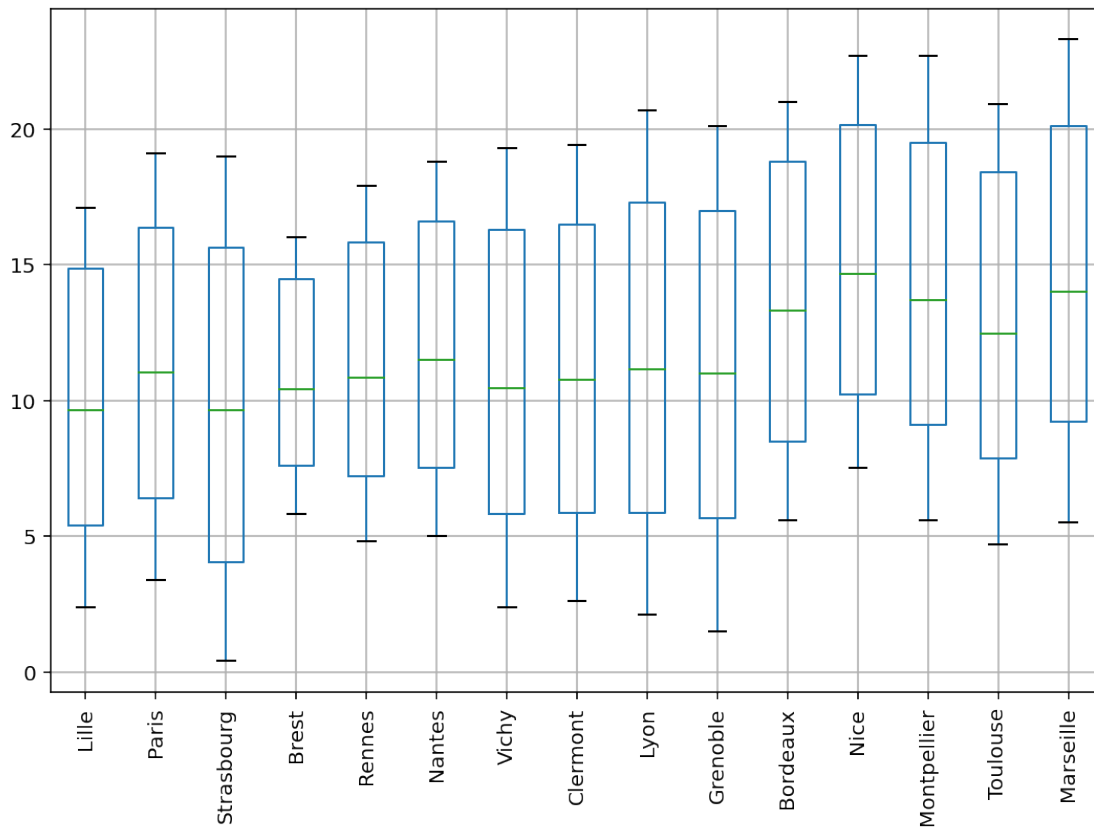
```
/usr/share/miniconda3/envs/big-data/lib/python3.8/site-
packages/pandas/plotting/_matplotlib/core.py:1235: UserWarning: FixedFormatter
```

```
should only be used together with FixedLocator  
ax.set_xticklabels(xticklabels)
```

```
[32]: <AxesSubplot:>
```



```
[33]: city_temp.boxplot(rot=90);
```



1.15 Describing

```
[34]: french_cities['Région'].describe()
```

```
[34]: count      15
      unique       4
      top        SE
      freq        7
      Name: Région, dtype: object
```

```
[35]: french_cities['Région'].unique()
```

```
[35]: array(['NE', 'NO', 'SE', 'SO'], dtype=object)
```

```
[36]: french_cities['Région'].value_counts()
```

```
[36]: SE      7
      NE      3
      NO      3
```

```
SO      2
Name: Région, dtype: int64
```

```
[37]: # To save memory, we can convert it to a categorical column:
french_cities["Région"] = french_cities["Région"].astype("category")
```

```
[38]: french_cities.memory_usage()
```

```
[38]: Index      760
      Janv      120
      Févr      120
      Mars      120
      Avri      120
      Mai       120
      Juin      120
      juil      120
      Août      120
      Sept      120
      Octo      120
      Nove      120
      Déce      120
      Lati      120
      Long      120
      Mean      120
      Ampl      120
      Région    207
      dtype: int64
```

1.16 Data Aggregation/summarization

1.17 groupby

```
[39]: fc_grouped_region = french_cities.groupby("Région")
      type(fc_grouped_region)
```

```
[39]: pandas.core.groupby.generic.DataFrameGroupBy
```

```
[40]: for group_name, subdf in fc_grouped_region:
      print(group_name)
      print(subdf)
      print("")
```

```
NE
      Janv  Févr  Mars  Avri  Mai  Juin  juil  Août  Sept  Octo  Nove  \
City
Lille      2.4   2.9   6.0   8.9  12.4  15.3  17.1  17.1  14.7  10.4   6.1
```


Paris	3.4	4.1	7.6	10.7	14.3	17.5	19.1	18.7	16.0	11.4	7.1
Strasbourg	0.4	1.5	5.6	9.8	14.0	17.2	19.0	18.3	15.1	9.5	4.9

	Déce	Lati	Long	Mean	Ampl	Région
City						
Lille	3.5	50.38	3.04	9.73	14.7	NE
Paris	4.3	48.52	2.20	11.18	15.7	NE
Strasbourg	1.3	48.35	7.45	9.72	18.6	NE

NO

	Janv	Févr	Mars	Avri	Mai	Juin	juil	Août	Sept	Octo	Nove	\
City												
Brest	6.1	5.8	7.8	9.2	11.6	14.4	15.6	16.0	14.7	12.0	9.0	
Rennes	4.8	5.3	7.9	10.1	13.1	16.2	17.9	17.8	15.7	11.6	7.8	
Nantes	5.0	5.3	8.4	10.8	13.9	17.2	18.8	18.6	16.4	12.2	8.2	

	Déce	Lati	Long	Mean	Ampl	Région
City						
Brest	7.0	48.24	-4.29	10.77	10.2	NO
Rennes	5.4	48.05	-1.41	11.13	13.1	NO
Nantes	5.5	47.13	-1.33	11.69	13.8	NO

SE

	Janv	Févr	Mars	Avri	Mai	Juin	juil	Août	Sept	Octo	Nove	\
City												
Vichy	2.4	3.4	7.1	9.9	13.6	17.1	19.3	18.8	16.0	11.0	6.6	
Clermont	2.6	3.7	7.5	10.3	13.8	17.3	19.4	19.1	16.2	11.2	6.6	
Lyon	2.1	3.3	7.7	10.9	14.9	18.5	20.7	20.1	16.9	11.4	6.7	
Grenoble	1.5	3.2	7.7	10.6	14.5	17.8	20.1	19.5	16.7	11.4	6.5	
Nice	7.5	8.5	10.8	13.3	16.7	20.1	22.7	22.5	20.3	16.0	11.5	
Montpellier	5.6	6.7	9.9	12.8	16.2	20.1	22.7	22.3	19.3	14.6	10.0	
Marseille	5.5	6.6	10.0	13.0	16.8	20.8	23.3	22.8	19.9	15.0	10.2	

	Déce	Lati	Long	Mean	Ampl	Région
City						
Vichy	3.4	46.08	3.26	10.72	16.9	SE
Clermont	3.6	45.47	3.05	10.94	16.8	SE
Lyon	3.1	45.45	4.51	11.36	18.6	SE
Grenoble	2.3	45.10	5.43	10.98	18.6	SE
Nice	8.2	43.42	7.15	14.84	15.2	SE
Montpellier	6.5	43.36	3.53	13.89	17.1	SE
Marseille	6.9	43.18	5.24	14.23	17.8	SE

SO

	Janv	Févr	Mars	Avri	Mai	Juin	juil	Août	Sept	Octo	Nove	\
City												
Bordeaux	5.6	6.6	10.3	12.8	15.8	19.3	20.9	21.0	18.6	13.8	9.1	
Toulouse	4.7	5.6	9.2	11.6	14.9	18.7	20.9	20.9	18.3	13.3	8.6	

City	Déce	Lati	Long	Mean	Ampl	Région
Bordeaux	6.2	44.50	-0.34	13.33	15.4	S0
Toulouse	5.5	43.36	1.26	12.68	16.2	S0