

# Setting category variables

WORKING WITH CATEGORICAL DATA IN PYTHON



**Kasey Jones**

Research Data Scientist

# New dataset: adoptable dogs

```
dogs.info()
```

```
RangeIndex: 2937 entries, 0 to 2936, Data columns (total 19 columns):
```

#	Column	Non-Null Count	Dtype
--	-----	-----	-----
0	ID	2937 non-null	int64
...			
8	color	2937 non-null	object
9	coat	2937 non-null	object
...			
17	get_along_cats	431 non-null	object
18	keep_in	1916 non-null	object

```
dtypes: float64(1), int64(1), object(17)
```

```
memory usage: 436.1+ KB
```

<sup>1</sup> <https://www.kaggle.com/jmolitoris/adoptable-dogs>

# A dog's coat

```
dogs["coat"] = dogs["coat"].astype("category")  
dogs["coat"].value_counts(dropna=False)
```

```
short      1972  
medium     565  
wirehaired 220  
long       180  
Name: coat, dtype: int64
```

# The .cat accessor object

```
Series.cat.method_name
```

Common parameters:

- `new_categories` : a list of categories
- `inplace` : Boolean - whether or not the update should overwrite the Series
- `ordered` : Boolean - whether or not the categorical is treated as an ordered categorical

# Setting Series categories

Set categories:

```
dogs["coat"] = dogs["coat"].cat.set_categories(  
    new_categories=["short", "medium", "long"]  
)
```

Check value counts:

```
dogs["coat"].value_counts(dropna=False)
```

```
short    1972  
medium   565  
NaN      220  
long     180
```

# Setting order

```
dogs["coat"] = dogs["coat"].cat.set_categories(  
    new_categories=["short", "medium", "long"],  
    ordered=True  
)  
dogs["coat"].head(3)
```

```
0    short  
1    short  
2    short  
Name: coat, dtype: category  
Categories (3, object): ['short' < 'medium' < 'long']
```

# Missing categories

```
dogs["likes_people"].value_counts(dropna=False)
```

yes	1991
NaN	938
no	8

A NaN could mean:

1. Truly unknown (we didn't check)
2. Not sure (dog likes "some" people)

# Adding categories

Add categories

```
dogs["likes_people"] = dogs["likes_people"].astype("category")
dogs["likes_people"] = dogs["likes_people"].cat.add_categories(
    new_categories=["did not check", "could not tell"]
)
```

Check categories:

```
dogs["likes_people"].cat.categories
```

```
Index(['no', 'yes', 'did not check', 'could not tell'], dtype='object')
```



# New categories

```
dogs["likes_people"].value_counts(dropna=False)
```

```
yes          1991
NaN          938
no             8
could not tell  0
did not check  0
```

# Removing categories

```
dogs["coat"] = dogs["coat"].astype("category")  
dogs["coat"] = dogs["coat"].cat.remove_categories(removals=["wirehaired"])
```

Check the categories:

```
dogs["coat"].cat.categories
```

```
Index(['long', 'medium', 'short'], dtype='object')
```

# Methods recap

- Setting: `cat.set_categories()`
  - Can be used to set the order of categories
  - All values not specified in this method are dropped
- Adding: `cat.add_categories()`
  - Does not change the value of any data in the DataFrame
  - Categories not listed in this method are left alone
- Removing: `cat.remove_categories()`
  - Values matching categories listed are set to `NaN`

# Practice updating categories

WORKING WITH CATEGORICAL DATA IN PYTHON

# Updating categories

WORKING WITH CATEGORICAL DATA IN PYTHON



**Kasey Jones**

Research Data Scientist

# The breed variable

Breed value counts:

```
dogs["breed"] = dogs["breed"].astype("category")  
dogs["breed"].value_counts()
```

```
Unknown Mix          1524  
German Shepherd Dog Mix  190  
Dachshund Mix        147  
Labrador Retriever Mix  83  
Staffordshire Terrier Mix 62  
...
```

# Renaming categories

The `rename_categories` method:

```
Series.cat.rename_categories(new_categories=dict)
```

Make a dictionary:

```
my_changes = {"Unknown Mix": "Unknown"}
```

Rename the category:

```
dogs["breed"] = dogs["breed"].cat.rename_categories(my_changes)
```

# The updated breed variable

Breed value counts:

```
dogs["breed"].value_counts()
```

```
Unknown          1524
German Shepherd Dog Mix  190
Dachshund Mix     147
Labrador Retriever Mix  83
Staffordshire Terrier Mix  62
...
```

Multiple changes at once:

```
my_changes = {
    old_name1: new_name1,
    old_name2: new_name2,
    ...
}
Series.cat.rename_categories(
    my_changes
)
```



# Renaming categories with a function

Update multiple categories:

```
dogs['sex'] = dogs['sex'].cat.rename_categories(lambda c: c.title())  
dogs['sex'].cat.categories
```

```
Index(['Female', 'Male'], dtype='object')
```

# Common replacement issues

- Must use new category names

```
# Does not work! "Unknown" already exists
use_new_categories = {"Unknown Mix": "Unknown"}
```

- Cannot collapse two categories into one

```
# Does not work! New names must be unique
cannot_repeat_categories = {
    "Unknown Mix": "Unknown",
    "Mixed Breed": "Unknown"
}
```

# Collapsing categories setup

A dogs color:

```
dogs["color"] = dogs["color"].astype("category")  
print(dogs["color"].cat.categories)
```

```
Index(['apricot', 'black', 'black and brown', 'black and tan',  
      'black and white', 'brown', 'brown and white', 'dotted', 'golden',  
      'gray', 'gray and black', 'gray and white', 'red', 'red and white',  
      'sable', 'saddle back', 'spotty', 'striped', 'tricolor', 'white',  
      'wild boar', 'yellow', 'yellow-brown'],  
      dtype='object')  
...
```

# Collapsing categories example

Create a dictionary and use `.replace` :

```
update_colors = {  
    "black and brown": "black",  
    "black and tan": "black",  
    "black and white": "black",  
}
```

```
dogs["main_color"] = dogs["color"].replace(update_colors)
```

Check the Series data type:

```
dogs["main_color"].dtype
```

```
dtype('O')
```

# Convert back to categorical

```
dogs["main_color"] = dogs["main_color"].astype("category")  
dogs["main_color"].cat.categories
```

```
Index(['apricot', 'black', 'brown', 'brown and white', 'dotted', 'golden',  
      'gray', 'gray and black', 'gray and white', 'red', 'red and white',  
      'sable', 'saddle back', 'spotty', 'striped', 'tricolor', 'white',  
      'wild boar', 'yellow', 'yellow-brown'],  
      dtype='object')
```

# Practice time

WORKING WITH CATEGORICAL DATA IN PYTHON

# Reordering categories

WORKING WITH CATEGORICAL DATA IN PYTHON



**Kasey Jones**

Research data scientist

# Why would you reorder?

1. Creating a ordinal variable
2. To set the order that variables are displayed in analysis
3. Memory savings



# Reordering example

```
dogs['coat'] = dogs["coat"].cat.reorder_categories(  
    new_categories = ['short', 'medium', 'wirehaired', 'long'],  
    ordered=True  
)
```

Using inplace:

```
dogs["coat"].cat.reorder_categories(  
    new_categories = ['short', 'medium', 'wirehaired', 'long'],  
    ordered=True,  
    inplace=True  
)
```

# Grouping when ordered=True

```
dogs['coat'] = dogs["coat"].cat.reorder_categories(  
    new_categories = ['short', 'medium', 'wirehaired', 'long'],  
    ordered=True  
)
```

```
dogs.groupby(by=['coat'])['age'].mean()
```

```
coat  
short      8.364746  
medium     9.027982  
wirehaired 8.424136  
long       9.552056
```

# Grouping when ordered=False

```
dogs['coat'] = dogs["coat"].cat.reorder_categories(  
    new_categories = ['short', 'medium', 'long', 'wirehaired'],  
    ordered=False  
)
```

```
dogs.groupby(by=['coat'])['age'].mean()
```

```
coat  
short      8.364746  
medium     9.027982  
long       9.552056  
wirehaired 8.424136
```

# Reordering practice

WORKING WITH CATEGORICAL DATA IN PYTHON

# Cleaning and accessing data

WORKING WITH CATEGORICAL DATA IN PYTHON



**Kasey Jones**

Research Data Scientist

# Possible issues with categorical data

1) Inconsistent values: "Ham" , "ham" , " Ham"

2) Misspelled values: "Ham" , "Hma"

3) Wrong dtype : df['Our Column'].dtype

```
dtype('O')
```

# Identifying issues

Use either:

- `Series.cat.categories`
- `Series.value_counts()`

```
dogs["get_along_cats"].value_counts()
```

```
No      2503
yes      275
no       156
Noo        2
NO         1
```

# Fixing issues: whitespace

Removing whitespace: `.strip()`

```
dogs["get_along_cats"] = dogs["get_along_cats"].str.strip()
```

Check the frequency counts:

```
dogs["get_along_cats"].value_counts()
```

```
No      2503
yes      275
no       156
Noo        2
N0         1  # < ---- no more whitespace
```



# Fixing issues: capitalization

Capitalization: `.title()` , `.upper()` , `.lower()`

```
dogs["get_along_cats"] = dogs["get_along_cats"].str.title()
```

Check the frequency counts:

```
dogs["get_along_cats"].value_counts()
```

No	2660
Yes	275
Noo	2

# Fixing issues: misspelled words

Fixing a typo with `.replace()`

```
replace_map = {"Noo": "No"}  
dogs["get_along_cats"].replace(replace_map, inplace=True)
```

Check the frequency counts:

```
dogs["get_along_cats"].value_counts()
```

No	2662
Yes	275

# Checking the data type

Checking the `dtype`

```
dogs["get_along_cats"].dtype
```

```
dtype('O')
```

Converting back to a category

```
dogs["get_along_cats"] = dogs["get_along_cats"].astype("category")
```

# Using the str accessor object

Searching for a string

```
dogs["breed"].str.contains("Shepherd", regex=False)
```

```
0      False
1      False
2      False
...
2935   False
2936    True
```

# Accessing data with loc

Access Series values based on category

```
dogs.loc[dogs["get_along_cats"] == "Yes", "size"]
```

Series value counts:

```
dogs.loc[dogs["get_along_cats"] == "Yes", "size"].value_counts(sort=False)
```

```
small    69
medium   169
large    37
```

# Clean and access practice

WORKING WITH CATEGORICAL DATA IN PYTHON