grayC

a perceptually-uniform colour map - www.fabiocrameri.ch/grayC

DOI 10.5281/zenodo. 1243863

Authors & Contributors

- **Fabio Crameri** - *Colour map* - contact
- **Grace Shephard** - *Conversion to .cpt files and more* - contact
- **Clint Conrad** - *Wider compatibility of .cpt files* - contact
- **Casper Pranger** - *Mathematica compatibility* - contact
- **Alexis Plunder** - *Wider compatibility of .xml files* - contact

Acknowledgement

Please acknowledge the free use of the colour map.

> e.g., "The perceptually-uniform colour map ***grayC*** is used in this study to prevent visual distortion of the data (Crameri 2018a,b)."
>
> *Crameri, Fabio. (2018, May 8). Scientific colour maps (Version 3.0.0). Zenodo. http://doi.org/10.5281/zenodo.1243863*
>
> *Crameri, F. (2018b, in open review), Geodynamic diagnostics, scientific visualisation and StagLab 3.0, Geosci. Model Dev. Discuss., doi:10.5194/gmd-2017-328*

Instructions

Using the .mat Format (MatLab)

Load the colour map into MatLab, either by adding the .mat file to the MatLab search path and using the command:

```
load('grayC.mat');
```

or by specifying the full file path to the .mat file:

```
load('~/work/Colormaps/grayC.mat');
```

Then use it, for example, with:

```
figure(1)
colormap(grayC)
colorbar
```

Using the .cpt Format (GMT)

The file grayC.cpt can be resampled for a given z-value range with the Generic Mapping Tools (GMT; http://gmt.soest.hawaii.edu/) command "makecpt".

For example to resample for an array from -2000 to 2000 in 100 increments you could generate a new file with:

```
$makecpt -CgrayC.cpt -T-2000/2000/100 > grayC_resampled.cpt
```

Using the .ct Format (VisIt)

The file grayC.ct can imported to VisIt by placing the .ct file in the .visit directory, which can be found on macOS under e.g.,:

```
/Applications/VisIt.app/Contents/Resources/ ...
... 2.12.3/darwin-x86_64/resources/colortables
```

The colour map should appear in the built-in list after VisIt has been restarted.

Using the .mat Format (Mathematica)

```
ColorMapSuitePath = "/Path/To/ColourMapSuite/";

ColorMapSuite[name_String] := ColorMapSuite[name, -1]
ColorMapSuite[name_String, el_] := With[{
    list =
     Transpose@{Subdivide[0, 1, 255],
       RGBColor @@@
        First@Import[
          ColorMapSuitePath <> "/" <> name <> "/" <> name <> ".mat"]}
    },
   Blend[list, {##}[[el]]] &
   ]
```

The function call `ColorMapSuite["name", i = -1]` returns a lambda function whose ith argument is used to define color (see the Manual for `ColorFunction` for details). `"name"` should be replaced with the name (in quotes) of the color scheme, e.g. `"grayC"`. Be sure to set the variable

`ColorMapSuitePath` to the path where your ColorMapSuite is installed.

General rules are:

- 1D plots of 1D functions/data: no (default) argument *i* suffices
- 2D plots of 2D functions/data: no (default) argument *i* suffices
- 3D plots of 2D functions/data: use *i* = 3
- 3D plots of 3D functions/data: use *i* = 4 (results might be worse than default Mathematica color functions, possibly due to lack of surface normal mapping)

```
ContourPlot[Sin[x] Sin[y], {x, 0, 2 Pi},
{y, 0, 2 Pi}, ColorFunction → ColorMapSuite["grayC"]]
```

License