Coloring in R's Blind Spot

by Achim Zeileis and Paul Murrell

Abstract Prior to version 4.0.0 R had a poor default color palette (using highly saturated red, green, blue, etc.) and provided very few alternative palettes, most of which also had poor perceptual properties (like the infamous rainbow palette). Starting with version 4.0.0 R gained a new and much improved default palette and, in addition, a selection of more than 100 well-established palettes are now available via the functions `palette.colors()` and `hcl.colors()`. The former provides a range of popular qualitative palettes for categorical data while the latter closely approximates many popular sequential and diverging palettes by systematically varying the perceptual hue, chroma, and luminance (HCL) properties in the palette. This paper provides a mix of contributions including an overview of the new color functions and the palettes they provide along with advice about which palettes are appropriate for specific tasks, especially with regard to making them accessible to viewers with color vision deficiencies.

1 Introduction

Color can be a very effective way to distinguish between different groups within a data visualization. Color is a "preattentive" visual feature, meaning that groups are identified rapidly and without conscious effort (Ware 2012). For example, it is trivial to identify the two groups of points in the scatterplot in Figure 1.

Employing color to represent values on a continuous numeric scale will be less successful (Cleveland and McGill 1984), but color can still be useful to convey additional variables when more effective visual features, such as location, have already been used. For example, color might be used to fill in different regions on a map, as demonstrated in the right hand plot of Figure 1.
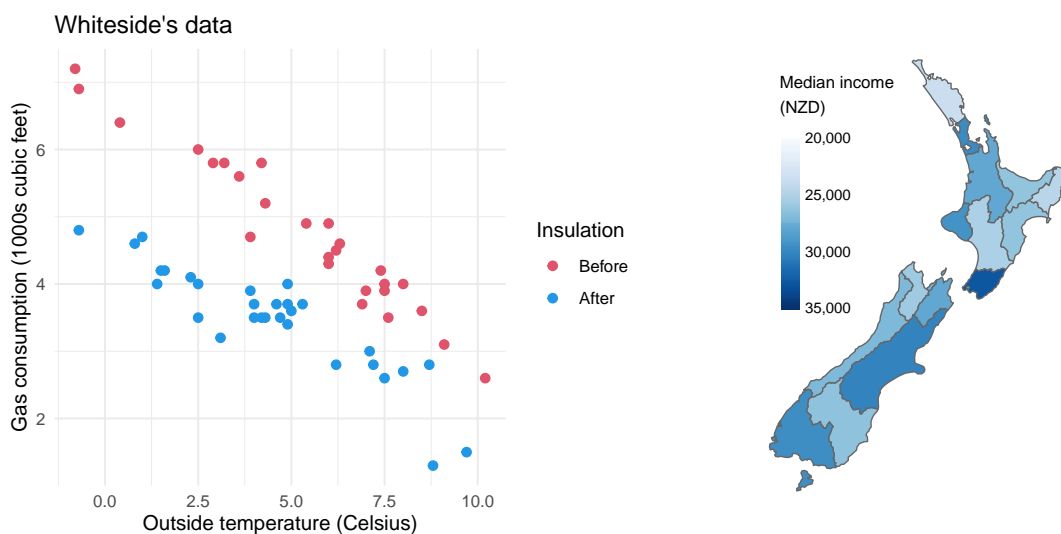
Figure 1: Typical usage of color for coding qualitative/categorical information (left) and quantitative/continuous information (right). Left: Scatter plot of weekly gas consumption by outside temperature before and after installing house insulation. Right: Choropleth map of median income in the 16 regions of New Zealand in 2018.

R provides several ways to specify a color: by name (e.g., `"red"`); by hexadecimal RGB code (e.g., `"#FF0000"`); or by integer (e.g., 2). When we specify an integer, that provides an index into a default set of colors; the color 2 means the second color in the default set of colors.

However, a more important task than specifying one particular color is the task of specifying a set of colors to use in combination with each other. For example, in the left panel of Figure 1, we need two colors that are very easily perceived as different from each other. In the right panel of Figure 1, we require a set of colors that appear to change monotonically, e.g., from darker to lighter.

We call this the problem of selecting a good *palette* of colors. What we need to generate is a vector of R colors, e.g., `c("red", "blue")`, `c("#FF0000", "#0000FF")`, or `c(2, 4)`.

2   A brief history of R palettes

Early versions of R provided very few functions for choosing colors from readily available palettes. The palettes that were provided, although standard at the time they were implemented, have meanwhile been widely recognized as being rather poor.
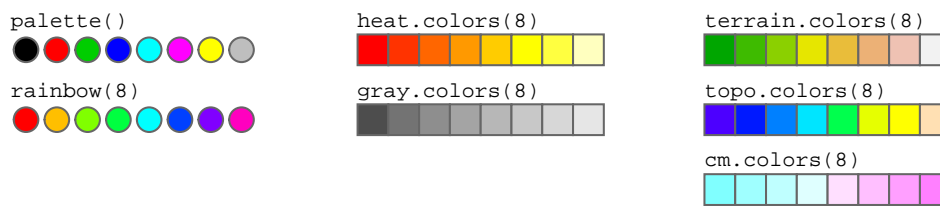
Figure 2: Old base R palettes. At top left is the old default palette (prior to version 4.0.0), consisting largely of highly saturated primary colors or combinations thereof. Below that is the rainbow palette of different highly saturated hues. The middle column shows the old sequential palettes, with heat colors again being highly saturated. The last column shows an old diverging palette plus two palettes motivated by shadings of geographic maps.

The `palette()` function generates a vector of eight colors. These provide the default set of colors that an integer color specification selects from and can be used for coding *categorical* information. The output below shows what R produced prior to version 4.0.0, along with a *swatch* of color circles.

```
palette()

#> [1] "black"   "red"     "green3"  "blue"    "cyan"    "magenta" "yellow"
#> [8] "gray"
```


Figure 2 depicts this old default `palette()` (top-left) along with other old base R palettes using swatches of circles or rectangles that are filled with the corresponding colors. The other palette functions all take an argument `n` to generate that number of colors (possibly along with further arguments that allow for certain customizations):

- `heat.colors()`, `terrain.colors()`, `topo.colors()`, and `gray.colors()` can be used as *sequential* palettes for ordered or numeric information.
- `cm.colors()` can be used as a *diverging* palette for values that are distributed around a "neutral" value, such as zero.
- `rainbow()` implements the infamous rainbow (or "jet") palette that was widely used (possibly with restrictions of the hue range) for all types of variables: *categorical*, *sequential*, and *diverging*.

All of these palettes – except `gray.colors()` – have poor perceptual properties. The colors are highly saturated, which can be distracting and overly stimulating, and the colors are unbalanced with respect to chroma and luminance, which means that they have unequal visual impact (Lonsdale and Lonsdale 2019; Bartram, Patra, and Stone 2017; Etchebehere and Fedorovskaya 2017). In addition, the palettes do not perform well for viewers with some form of colorblindness (about 10% of the male population, Ware 2012). Most of the palettes also use sequences of hues obtained in the RGB (red-green-blue) space or simple derivations thereof like HSV (hue-saturation-value) or HLS (hue-lightness-saturation), which leads to clustering of colors at the red, green, and blue primaries.

Although these limitations have been well known for some time, no changes were made to these palettes provided by the core R graphics system for a number of years. There were various reasons for this including the following:

- In R version 2.1.0, Thomas Lumley added the `colorRampPalette()` function. This made it easier to generate a palette, though the user is still required to select, for example, start and end colors from which a palette of colors can then be interpolated.
- Better palettes became available via packages on CRAN (Comprehensive R Archive Network) starting with **RColorBrewer** (Neuwirth 2022, first published on CRAN in 2002), later **colorspace** (Ihaka 2003; Zeileis, Hornik, and Murrell 2009), and more recently **viridis** (Garnier 2023), **rcartocolor** (Nowosad 2023b), **scico** (Pedersen and Crameri 2023), and **cols4all** (Tennekes 2023),

among many others. In most cases, these make palettes available in R that were developed elsewhere, e.g., ColorBrewer.org (Harrower and Brewer 2003), CARTOColors (CARTO 2023), the scientific color maps of Crameri, Shephard, and Heron (2020), and the Viridis palettes for matplotlib in Python (Smith and Van der Walt 2015).

- Higher-level graphics systems like **ggplot2** (Wickham 2016) and **lattice** (Sarkar 2008) developed their own color themes.

3 A new set of R palettes

On the road to R version 4.0.0 an attempt was made to address the limited and deficient set of palettes in base R and to add a range of modern color palettes. In particular, `palette()` has a new improved default color palette, `palette.colors()` provides further well-established qualitative palettes (Zeileis et al. 2019), and `hcl.colors()` provides a wide range of qualitative, sequential, and diverging palettes obtained by a standardized approach in the so-called HCL (hue-chroma-luminance) space (Wikipedia 2023); see Zeileis and Murrell (2019) and Zeileis et al. (2020).

3.1 A new default color `palette()`

The default color palette in R – the default set of colors that can be specified by integer index – has been replaced. The new palette follows the same basic hues as the old default palette, but the palette is less saturated overall and reduces the size of changes in chroma and luminance across the palette. This produces a calmer and less distracting palette with a more even visual impact. An attempt has also been made to improve the discriminability of the colors in the default palette for colorblind viewers. The output (and swatches) below show what R produces from version 4.0.0 onwards.

```
palette()

#> [1] "black"   "#DF536B" "#61D04F" "#2297E6" "#28E2E5" "#CD0BBC" "#F5C710"
#> [8] "gray62"
```

3.2 The `palette.colors()` function

The `palette.colors()` function, new in R 4.0.0, provides a way to access several other predefined palettes (see also Figure 7). All of these are *qualitative palettes* so they are appropriate for encoding qualitative (categorical) variables. In other words, these palettes are appropriate for differentiating between groups. By default `palette.colors()` returns the "Okabe-Ito" (Okabe and Ito 2008) palette. This palette was designed to be very robust under color vision deficiencies, so the different colors in this palette should be easily distinguishable for all viewers.

```
palette.colors()

#> [1] "#000000" "#E69F00" "#56B4E9" "#009E73" "#F0E442" "#0072B2" "#D55E00"
#> [8] "#CC79A7" "#999999"
```

The first argument to `palette.colors()` is a number of colors. Each palette has a fixed number of colors, but we can ask for fewer or, with `recycle = TRUE`, we can get more colors by recycling. For example, the following code just requests the first four colors from the "Okabe-Ito" palette.

```
palette.colors(4)

#> [1] "#000000" "#E69F00" "#56B4E9" "#009E73"
```

Note that up to R version 4.2.x some `palette.colors()`, including "Okabe-Ito", provide *named* output but starting from 4.3.0 all output is unnamed by default.

The following code requests ten colors from the `"Okabe-Ito"` palette. That palette only contains nine colors, but because `recycle = TRUE`, a tenth color is provided by recycling the first color (black) from the palette.

```
palette.colors(10, recycle = TRUE)

#>  [1] "#000000" "#E69F00" "#56B4E9" "#009E73" "#F0E442" "#0072B2" "#D55E00"
#>  [8] "#CC79A7" "#999999" "#000000"
```

The second argument to `palette.colors()` is the palette to select colors from. For example, the following code requests the first four colors from the "R4" palette (the new default in `palette()`).

```
palette.colors(4, palette = "R4")

#> [1] "#000000" "#DF536B" "#61D04F" "#2297E6"
```

3.3 The `hcl.colors()` function

The `hcl.colors()` function was added in R 3.6.0, with the range of supported palettes slowly expanded over time. This function provides access to another range of palettes, including sequential and diverging palettes for representing continuous variables. As with `palette.colors()`, the first argument is a number of colors to generate and the second specifies a palette to generate colors from. The `hcl.pals()` function provides a full list of the available palette names that we can choose from.

```
hcl.colors(8, palette = "Blues 3")

#> [1] "#00366C" "#005893" "#007BC0" "#5E9BD8" "#91BAEB" "#BAD5FA" "#DDECFF"
#> [8] "#F9F9F9"
```

One difference with `hcl.colors()` is that the palette we are selecting colors from is *not* a fixed set of colors. Instead, the palettes in `hcl.colors()` are a path within HCL colorspace. For each dimension – hue, chroma, and luminance – a palette can have a constant value, a monotonic trajectory, or a triangular trajectory. For example, the trajectories for the "Blues 3" palette are shown in Figure 3. The palette is (almost) constant in the hue dimension yielding different shades of (almost) the same blue. The palette is monotonically increasing in the luminance dimension, so the blues vary from very dark to very light. Finally, the palette has a triangular trajectory in the chroma dimension, so the blues are more colorful towards the middle of the palette. The trajectories do not involve exactly straight lines because in some cases a power curve is employed and in other cases the palette has to be adjusted to remain within the range of representable colours – see Zeileis, Hornik, and Murrell (2009) and Ihaka et al. (2023) for more details.

Because the palettes from `hcl.colors()` are based on a continuous path in HCL space, we can select as many colors as we like. For example, the following code generates five colors from the multi-hue sequential palette "YlGnBu" (see also Figure 6) and nine colors from the diverging palette "Purple-Green" (see also Figure 12).

```
hcl.colors(5, palette = "YlGnBu")

#> [1] "#26185F" "#007EB3" "#18BDB0" "#BCE9C5" "#FCFFDD"
```
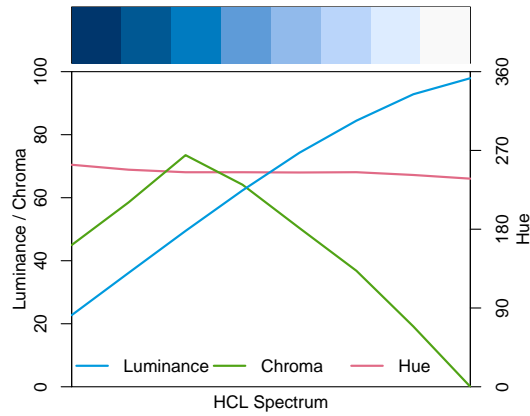
```
hcl.colors(9, palette = "Purple-Green")

#> [1] "#492050" "#90529C" "#C490CF" "#E4CAE9" "#F1F1F1" "#BCDABC" "#72B173"
#> [8] "#2C792D" "#023903"
```

Figure 3: Hue, chroma, and luminance paths for the `"Blues 3"` palette. This plot is created by the `colorspace::specplot()` function. We can see that hue is held constant in this palette, while luminance increases monotonically and chroma peaks towards the middle of the palette.

3.4 Illustrations

To illustrate the benefits of the new color palettes, Figure 4 shows several versions of a time series plot, depicting four different European stock indexes during most of the 1990s (`EuStockMarkets` data). The plots compare the old `"R3"` default palette with the new `"R4"` default and the new qualitative palette `"Okabe-Ito"`. These can all be selected using `palette.colors()`. The first row shows the `"R3"` default using a typical color legend in the top left corner; the second column shows an emulation of a kind of red-green color blindness known as deuteranopia using the **colorspace** package (based on Machado, Oliveira, and Fernandes 2009). The second row uses the `"R4"` palette and the third row uses `"Okabe-Ito"`; both with direct labels for the different time series instead of a color legend.
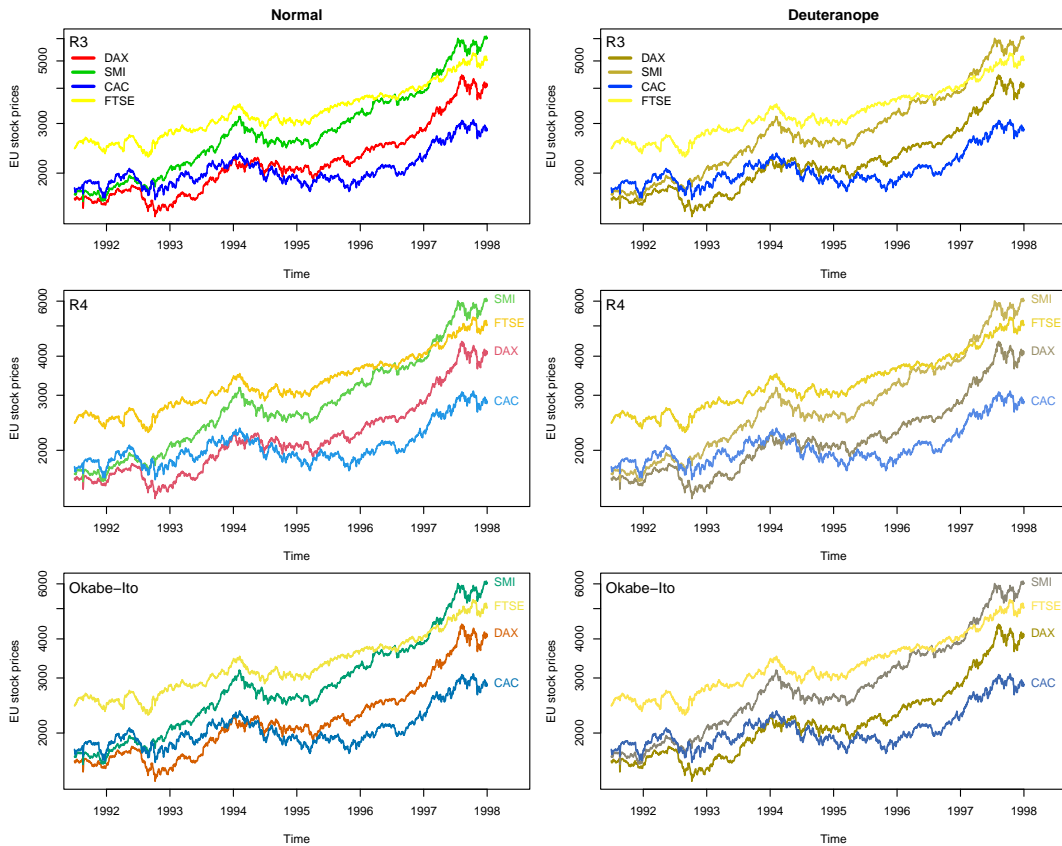
Figure 4: Time series line plot of `EuStockMarkets`. Rows: Old `"R3"` default palette (top), new `"R4"` default palette (middle), `"OkabeIto"` palette (bottom), designed to be robust under color vision deficiencies. Columns: Normal vision (left) and emulated deuteranope vision (right). A color legend is used in the first row and direct labels in the other rows.
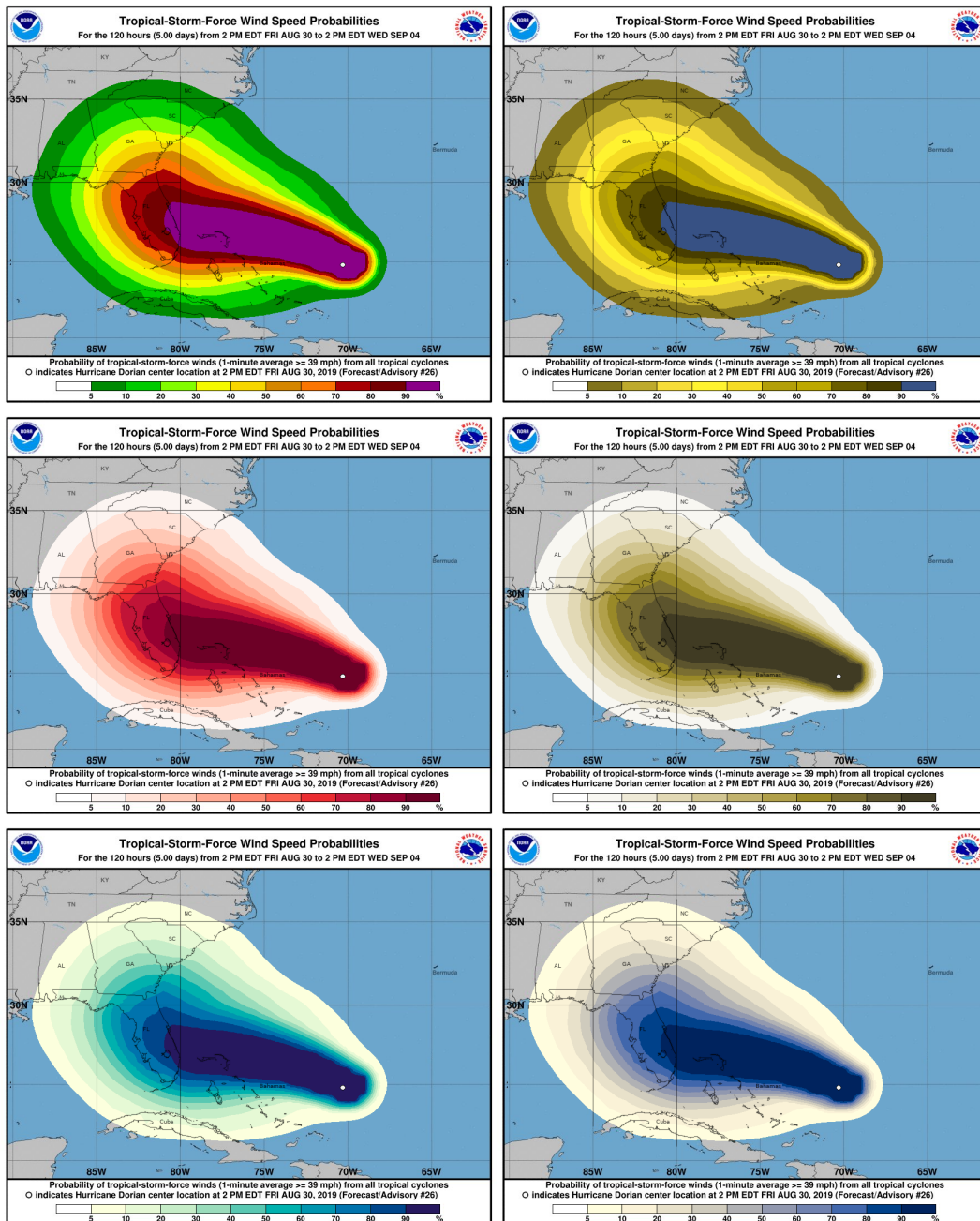
Figure 5: Probability of wind speeds > 39 mph $(63 \, \text{km} \, \text{h}^{-1})$ during hurricane Dorian in 2019. On the left is the the original image (top row) and two reproductions using the "Reds" (middle) and "YlGnBu" (bottom) multi-hue sequential palettes. On the right are emulations of how the images on the left might appear to a colorblind viewer.

We can see that the "R3" colors are highly saturated and they vary in luminance. For example, the yellow line is noticeably lighter than the others. Futhermore, for deuteranope viewers, the DAX and the SMI lines are difficult to distinguish from each other (exacerbated by the use of a color legend that makes matching the lines to labels almost impossible). Moreover, the FTSE line is more difficult to distinguish from the white background, compared to the other lines.

The "R4" palette is an improvement: the luminance is more even and the colors are less saturated, plus the colors are more distinguishable for deuteranope viewers (aided by the use of direct color labels instead of a legend). The "Okabe-Ito" palette works even better, particularly for deuteranope viewers.

To illustrate an application of the new sequential color palettes for use with continuous data, Figure 5 shows several versions of a weather map that was produced by the National Oceanic and Atmospheric Administration (and infamously misinterpreted by a former President of The United
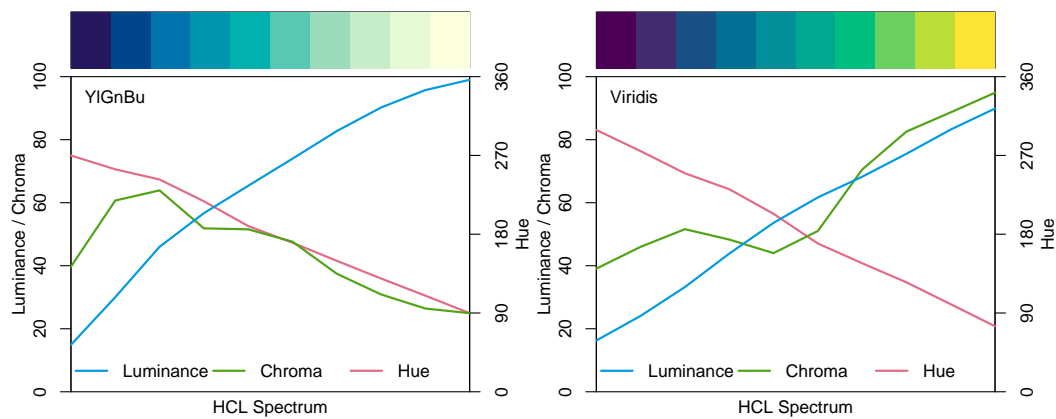
Figure 6: Hue, chroma, and luminance paths for the `"YlGnBu"` (left) and `"Viridis"` (right) palettes. These plots are created by the `colorspace::specplot()` function. For `"YlGnBu"` we can see that hue changes from blue to yellow, luminance increases monotonically, and chroma has a small peak in the blue range and then decreases with luminance. `"Viridis"`, on the other hand, has almost the same trajectory for both hue and luminance, but chroma increases for the light colors.

States, see Zeileis and Stauffer 2019). The top row shows the original image along with an emulation of deuteranopia in the second column. The middle row uses the sequential palette `"Reds"` that can be selected using `hcl.colors()` and the bottom row uses the sequential palette `"YlGnBu"`, which is also available via `hcl.colors()`.

The weather map is intended to convey the probability of wind speeds > 39 mph during hurricane Dorian, 2019-08-30–2019-09-04. The probabilities are highest in the central magenta region and lowest in the outer green regions. The original image does not convey the information very well because there is a non-monotonic change in luminance (from dark to light and back to dark); the high saturation across all of the colors is also distracting. These issues persist for deuteranope viewers, plus any benefit of a red (danger!) to green (safe) change in hue is lost.

The `"Reds"` version of the image conveys the information more clearly by relating the monotonic changes in probability to monotonic changes in luminance. Hue is fairly constant in this palette and the saturation peaks towards the middle, which is similar to the `"Blues 3"` palette shown in Figure 3, just with a different narrow range of hues. The deuteranope version retains this advantage.

The `"YlGnBu"` version of the image is also more effective than the original. This palette employs a much broader range of hues and varies chroma along with luminances so that the dark colors have higher chroma and the light colors lower chroma (see Figure 6). This still clearly conveys the order from light to dark but additionally yields more distinguishable colors, making it easier to associate contour bands with the legend. Note that the `"YlGnBu"` palette is similar to the very popular `"Viridis"` palette (also shown in Figure 6 on the right), with almost the same hue and luminance trajectories. However, an important advantage of the `"YlGnBu"` palette in this visualization is that the light colors have low chroma and thus signal low risk better than the light colors in the `"Viridis"` palette which have very high chroma. Finally, we remark that the `"YlGnBu"` version does lose the benefit of red (danger!) at high probabilities; an alternative would be to use the `"Purple-Yellow"` multi-hue palette instead, a variation of which was used by Zeileis and Stauffer (2019).

The following sections describe the full range of new color palettes in more detail. A much more condensed overview of the new functions and palettes that are available and some suggestions for robust default palettes are given in Section 6.

4 A gallery of palettes

This section goes through all of the color palettes that are now available in base R (without using any additional packages). There is some discussion of the background for the palettes, strengths and weaknesses of different palettes, and appropriate uses of the palettes.

4.1 The `palette.colors()` function

The `palette.colors()` function provides a range of qualitative palettes (see Figure 7 for an overview). The first argument to the `palette.colors()` function specifies the number of colors to return and the

`palette` argument allows us to select the palette of colors to choose from. As previously mentioned, the default palette is `"Okabe-Ito"`, which has very good perceptual properties. The `"R4"` palette specifies the new R default palette which is also returned by `palette()` by default. As previously mentioned, this was constructed to have reasonable perceptual properties, including accommodation for color vision deficiencies (see Zeileis et al. 2019 for more details). The accompanying `palette.pals()` function returns a character vector of the available palette names.
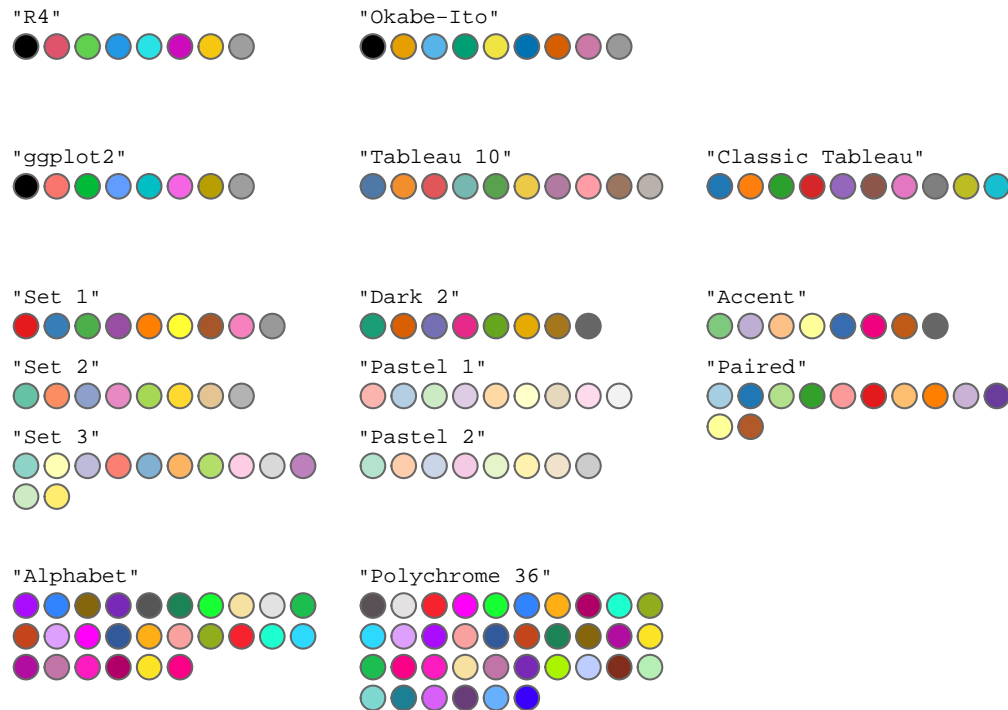
Figure 7: New qualitative palettes in base R available from the `palette.colors()` function. The label above each swatch shows the argument to provide to `palette.colors()` to produce the set of colors. The palette at top-left is the new default that is also produced by `palette()`. The `"Okabe-Ito"` palette is the default that is produced by `palette.colors()` (with no arguments).

```
palette.pals()
```

```
#>  [1] "R3"            "R4"              "ggplot2"         "Okabe-Ito"
#>  [5] "Accent"        "Dark 2"          "Paired"          "Pastel 1"
#>  [9] "Pastel 2"      "Set 1"           "Set 2"           "Set 3"
#> [13] "Tableau 10"    "Classic Tableau" "Polychrome 36"   "Alphabet"
```

Each of the predefined palettes can be set as the default palette by passing the palette name to the `palette()` function. For example, the following code sets the Okabe-Ito palette as the default palette.

```
palette("Okabe-Ito")
```

There are several palettes that have been taken from the color schemes of the **ggplot2** package and Tableau (Tableau Software, LLC 2021), both well-established graphics systems.
The `"Tableau 10"` palette represents the redesign of the default palette in Tableau (Stone 2016). These palettes may be useful to emulate the look and feel of plots from those other systems.

Several palettes come from ColorBrewer.org (Harrower and Brewer 2003) and were originally designed for filling regions on maps. However, they are also useful for filling regions within data visualizations like bar plots, density plots, and heatmaps, among others. Two of these palettes are a bit different because they deliberately contain darker and lighter colors: the `"Accent"` palette may be useful to emphasize one or more categories over the others; the `"Paired"` palette may be useful to represent more than one categorical variable via color, e.g., different types of treatment as well as high vs. low levels of each treatment.

Finally, there are two palettes from the **Polychrome** package (Coombes et al. 2019). These are much larger palettes, with colors chosen to be evenly spread throughout HCL colorspace. The `"Polychrome 36"` palette represents the largest set of colors that could be generated while still being visually distinguishable. The `"Alphabet"` palette is a smaller, but still large, set (one for each letter of the alphabet). These palettes may be useful if we are attempting to represent a very large number of categories at once. The result is unlikely to be easy to interpret, but these palettes will provide the best chance.

4.2 The `hcl.colors()` function

The `hcl.colors()` function provides qualitative, sequential, and diverging palettes that are derived from certain trajectories of the perceptual properties – hue, chroma, and luminance (HCL). Most of the resulting palettes have one or more desirable perceptual properties:

- **Colorblind-safe:** This means that the palette retains its perceptual properties for colorblind users.
- **Perceptual order:** This means that there is a perceived ordering of the colors, typically arising from a monotonic change from light to dark or vice versa.
- **Perceptual uniformity:** This means that if we take a small step along the path of the palette in HCL space, the perceived difference between the two colors will be the same anywhere along the path.
- **Perceptual balance:** This means that, for example, while there are changes in hue and chroma, luminance remains pretty much the same, so no color stands out from the others.

These properties are very difficult to achieve in a single palette, which is one reason why there are multiple palettes available. Furthermore, different properties will be more or less important depending on the data being displayed and the point that a data visualization is attempting to make. For example, perceptual balance is not desirable when we want to highlight a particular point or category of interest; in that scenario we explicitly want some colors to have a greater visual impact than others. The choice of palette may also depend on how many colors are needed. For example, a palette with a light gray, a medium color, and a full color may still work effectively on a white background if the light gray group is less important and is just provided in the background for reference.

Perceptual order and colorblind-safety are closely linked because the easiest approach to obtaining a colorblind-safe palette is by using a monotonic change in luminance. All of the sequential palettes in `hcl.colors()` in fact have this property and are colorblind-safe to a certain degree, though the effectiveness depends on the range of luminance within the palette. A quick way to check a palette for colorblind-safety is via `colorspace::swatchplot(pal, cvd = TRUE)`, where `pal` is a palette of colors. More elaborate tools are provided by the package **colorblindcheck** (Nowosad 2023a).

The **colorspace** package also provides functions like `sequential_hcl()` and `diverging_hcl()` to generate palettes by defining a custom set of hue, chroma, and luminance trajectories, e.g., based on specific hues that have inherent meanings for a particular data set.

Qualitative palettes

The qualitative palettes available from `hcl.colors()` are shown in Figure 8. The common feature of these palettes is that they only vary hue while using the same chroma and luminance for all of their colors. One drawback to this approach is that fewer easily distinguishable colors can be generated from these palettes.

The first five palettes are inspired by the ColorBrewer.org palettes of the same name. They employ different fixed levels of chroma and luminance and span the full hue range. Most of these palettes are also available as a fixed set of colors via `palette.colors()`. There are two key differences: First, chroma and luminance are fixed in `hcl.colors()` but typically vary somewhat in `palette.colors()`. The former has the advantage that the colors are more balanced. The latter has the advantage that more sufficiently different colors can be obtained. Second, `hcl.colors()` will return `n` colors interpolated from the full range of hues, whereas `palette.colors()` will return the first `n` colors from a fixed set.

The ColorBrewer.org palettes were designed with good perceptual properties in mind, but also relied on expert opinion and trial and error. This means that a little more care should be taken when selecting one of the ColorBrewer-inspired HCL-based palettes because, for example, they are often not colorblind-safe. The ColorBrewer.org palettes are also available in R via the **RColorBrewer** package, which includes a facility for generating ColorBrewer palettes that are colorblind-safe.

The remaining four palettes are taken from Ihaka (2003). These palettes keep chroma and luminance fixed and restrict the range of hues (blues and greens for `"Cold"` and reds and oranges for `"Warm"`).

Figure 8: The qualitative palettes that are available with the `hcl.colors()` function.

Holding chroma and luminance fixed means that the visual impact is even across the palette. This makes these palettes appropriate if all categories in a variable have equal importance, but, as with the ColorBrewer.org emulations, they are not colorblind-safe and they will not be appropriate for grayscale printing.

When palettes are employed for shading areas in statistical displays (e.g., in bar plots, pie charts, or regions in maps), lighter colors (with moderate chroma and high luminance) such as "Pastel 1" or "Set 3" are typically less distracting. By contrast, when coloring points or lines, colors with a higher chroma are often required: On a white background a moderate luminance as in "Dark 2" or "Dark 3" usually works better while on a black/dark background the luminance should be higher as in "Set 3" for example.

Single-hue sequential palettes

We divide sequential palettes into single-hue (this section) and multi-hue palettes (the next section).

Single-hue sequential palettes vary only from dark/colorful to light/gray, with a constant underlying hue. Figure 3 provides a good example of the hue, chroma, and luminance trajectories for these palettes. Certain hues will be more appropriate for representing data on specific concepts, such as green for "vegetation" and red for "temperature".

Figure 9 shows the sequential palettes that hold hue largely constant. All of these palettes have a large monotonic variation in luminance, typically from dark to light. This is also typically accompanied by a change in chroma from more colorful to less. The result is a palette that makes it very easy to distinguish extreme values. Some palettes also have a pronounced peak of chroma somewhere in the middle, which makes it easier to distinguish moderate values from extreme values (e.g., "Reds 3", "Blues 3", etc.).
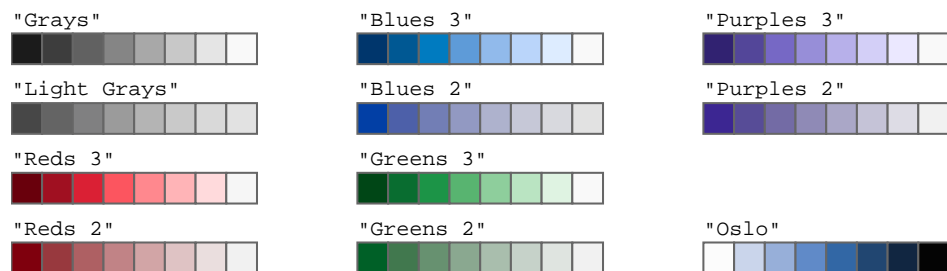
Figure 9: The single-hue sequential palettes that are available with the `hcl.colors()` function.

All palettes in this group, except the last one, are inspired by the ColorBrewer.org palettes with the same base name, but are restricted to a single hue only. They are intended for a white/light background. The last palette, "Oslo", is taken from Crameri's scientific color maps and is intended for a black/dark background and hence the order is reversed starting from a very light blue (almost white).

When only a few colors are needed (e.g., for coding an ordinal categorical variable with few levels) then a lower luminance contrast may suffice (e.g., "Light Grays", "Reds 2", "Blues 2", etc.).

Multi-hue sequential palettes

Multi-hue sequential palettes not only vary luminance, from light to dark (typically along with chroma), but also vary hue. In order to not only bring out extreme colors in a sequential palette but also better distinguish middle colors, it is a common strategy to employ a sequence of hues. This leads to a large range of possible palettes. Figure 6 shows examples of the hue, chroma, and luminance trajectories from multi-hue palettes.

Note that the palettes in this section differ substantially in the amount of chroma and luminance contrasts. For example, many palettes go from a dark high-chroma color to a neutral low-chroma color (e.g., "Reds", "Purples", "Greens", "Blues") or even light gray (e.g., "Purple-Blue"). But some palettes also employ relatively high chroma throughout the palette (e.g., emulations of Viridis and CARTOColor palettes). The former strategy is suitable to emphasize extreme values, while the latter works better if all values along the sequence should receive the same perceptual weight.

Palettes that involve a significant variation in hue, e.g., "YlGnBu", can be more effective when we need to match specific colors to a legend (e.g., the bottom row of Figure 5) or across several small-multiples, as in facetted plots.
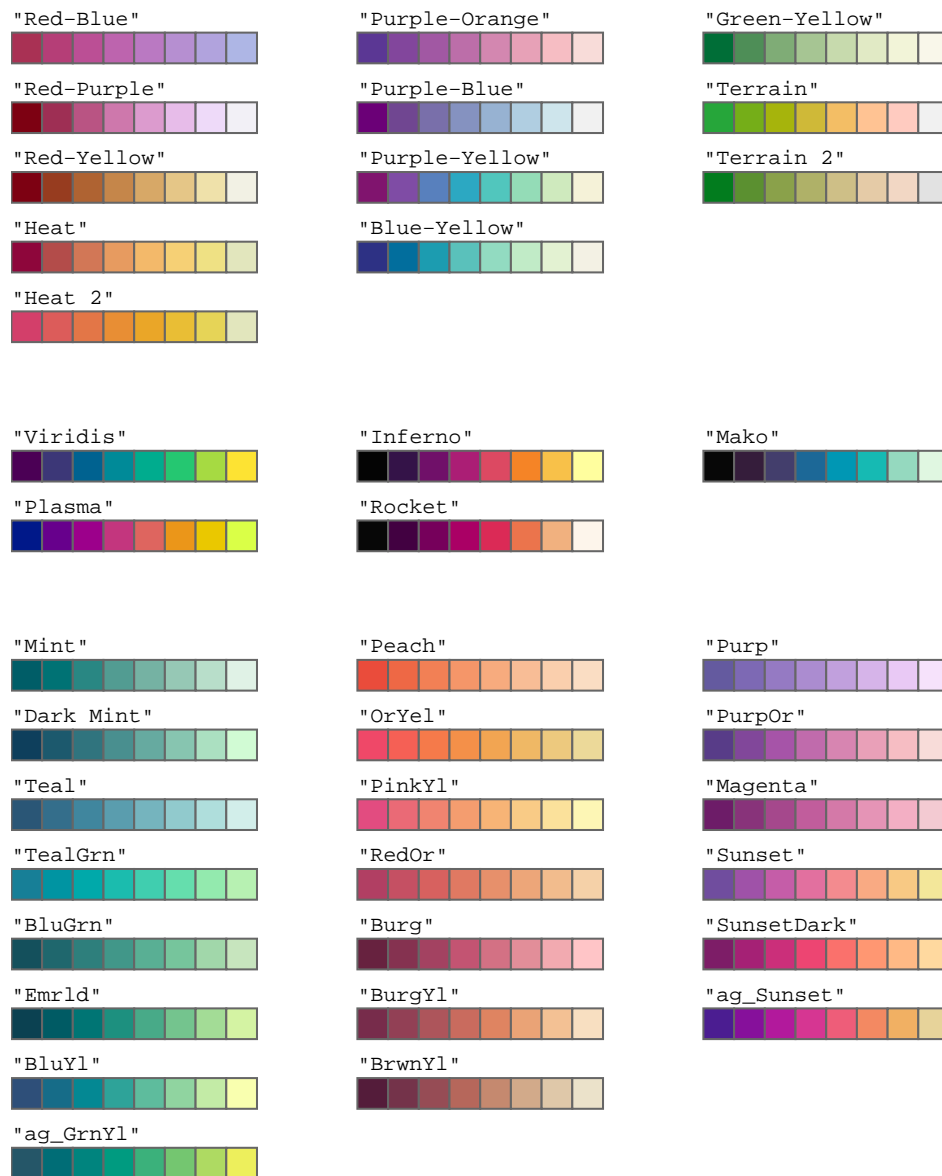
Figure 10: Some of the multi-hue sequential palettes that are available with the `hcl.colors()` function.

Of the palettes shown in Figure 10, "Red-Blue" to "Terrain 2" are palettes created during the development of the **colorspace** package.

The next collection of palettes, "Viridis" to "Mako", emulate popular palettes within the Python

community. The "Viridis", "Plasma", and "Inferno" palettes come from the matplotlib Python library and work well for identifying features of interest in false-color images. This means that they should also work well for heatmaps. The large range of hues means that these palettes can also serve as qualitative palettes, which makes them robust default palettes. However, this versatility means that a palette that is purely sequential or purely qualitative may serve better for a specific purpose.

The "Mako" and "Rocket" palettes are from the seaborn Python library with an emphasis on high chroma and a wide range of luminance. This makes these palettes a good choice for heatmaps.

The remaining palettes in Figure 10, from "Mint" to "Sunset" closely match the corresponding CARTOColors palettes. These palettes tend to span a much narrower range of hues, chroma, and luminance, so can be useful if we just need to represent a small number of ordered values. The resulting colors from these palettes will have, for example, more similar hues than a palette generated from "Viridis", with its wide range of hues.
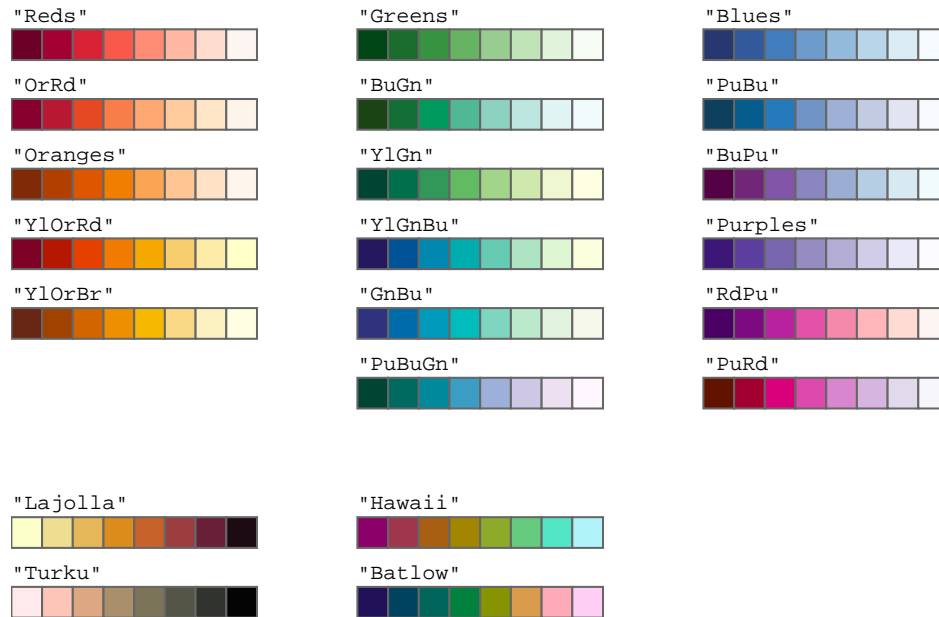
Figure 11: Some of the multi-hue sequential palettes that are available with the `hcl.colors()` function.

Figure 11 shows the remaining multi-hue sequential palettes that are available in `hcl.colors()`. Most of the top group of palettes, starting with "Reds", "Greens", and "Blues", closely match Color-Brewer.org palettes of the same name. The "YlGnBu" palette is of particular note as it uses essentially the same hues as the "Viridis" palette (Figure 10), but it is more useful as a sequential palette because chroma decreases for the high-luminance colors (see also Figure 6).

The next group of palettes, "Lajolla" to "Batlow" closely match the palettes of the same name from Crameri's scientific color maps. These palettes are constructed with a luminance scale so that there is a clear visual ordering of the palette. They are also designed to be readable by colorblind users, to work for grayscale printing, and to provide perceptual balance, so that no color has a greater visual emphasis than any other. While "Lajolla" and "Turku" are intended for use with a black/dark background, "Hawaii" and "Batlow" are for use with a white/light background. Moreover, the latter two span a particularly large range of hues, thus yielding a kind of "scientific rainbow".

Diverging palettes

The diverging palettes offer a range of underlying hues for either extreme, with either light gray or yellow as the central "neutral" value. The palettes with yellow at the centre provide less of a change in colorfulness, so the "neutral" value is more of a turning point rather than a local minimum. Figure 13 shows the selection of diverging palettes for use with `hcl.colors()`.

All of these palettes are "balanced" in the sense that chroma and luminance vary in the same way as we move from the central neutral color towards either end of the palette. Figure 12 (left) shows this idea of balance for the "Purple-Green" palette.

When choosing a particular palette for a display similar considerations apply as for the sequential
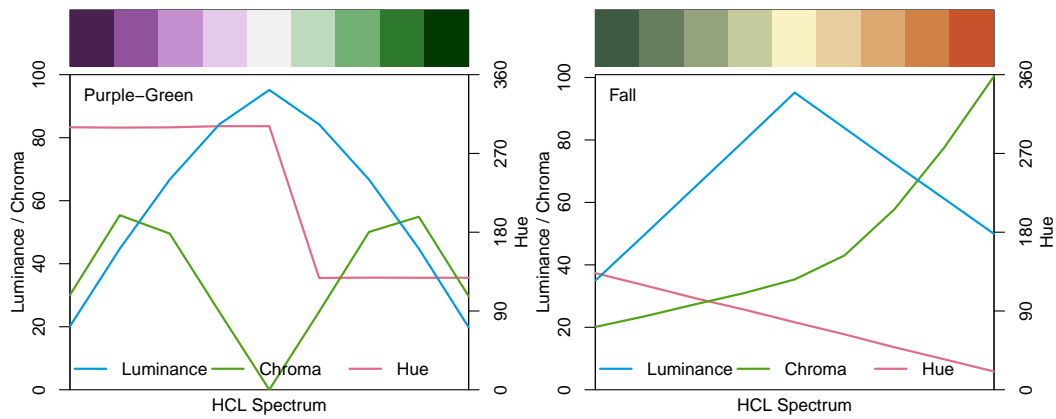
Figure 12: Hue, chroma, and luminance paths for the "Purple-Green" (left) and "Fall" (right) palettes. The plots are created by the colorspace::specplot() function. We can see that the "Purple-Green" palette is "balanced" with luminance and chroma varying symmetrically about the central neutral color for both hues. In contrast, the "Fall" palette is "unbalanced" with the left arm of the palette having somewhat darker colors with far less chroma than the right arm. Hue changes gradually from green through yellow to red, yielding a warmer palette compared to "Purple-Green".

palettes. For example, large luminance differences are important when many colors are used while smaller luminance contrasts may suffice for palettes with fewer colors.

**Figure 13:** The balanced diverging palettes that are available with the hcl.colors() function.

Almost all of the palettes in the first two groups, those involving simple color pairs like "Blue-Red" or "Cyan-Magenta", were developed as part of the **colorspace** package, taking inspiration from various other palettes, including more balanced and simplified versions of several ColorBrewer.org palettes. The exception is the "Tropic" palette, which closely matches the palette of the same name from CARTOColors.

The palettes "Broc" to "Vik" and "Berlin" to "Tofino" closely match Crameri's scientific color maps of the same name, where the first three are intended for a white/light background and the other three for a black/dark background.

Flexible diverging palettes

Figure 14 shows a set of more flexible diverging palettes. These do not impose any restrictions that the two "arms" of the palette need to be balanced and also may go through a non-gray neutral color (typically light yellow). Consequently, these palettes may be used to provide a larger set of distinguishable colors compared to the diverging palettes from the previous section. The price of this

flexibility is that the chroma/luminance within these palettes can be rather unbalanced. For example, Figure 12 (right) demonstrates this feature of the "Fall" palette.
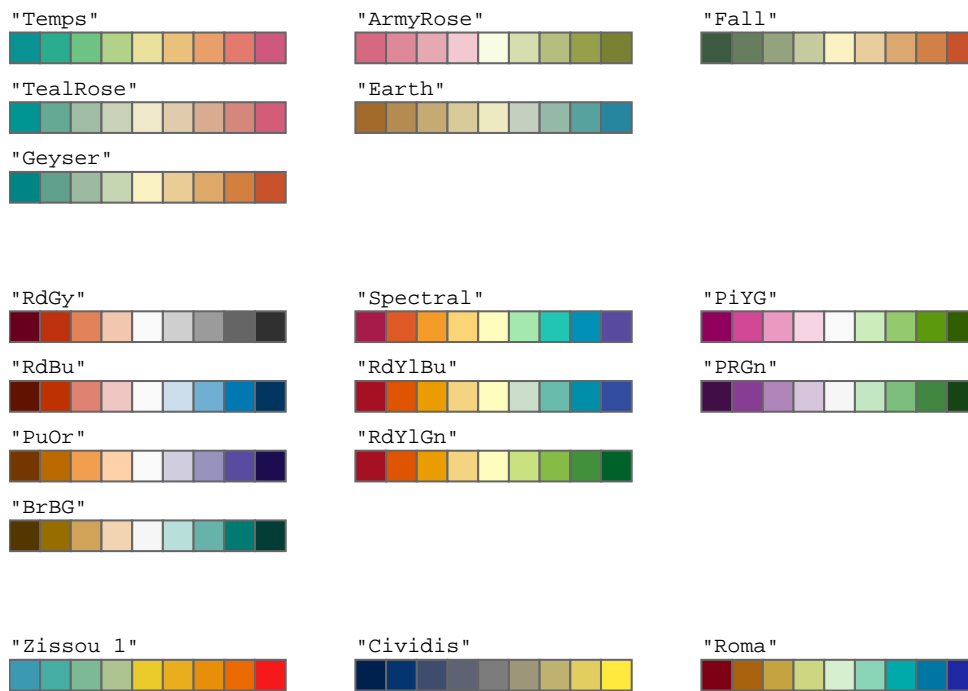
Figure 14: The flexible diverging palettes that are available with the `hcl.colors()` function.

The first group of palettes, including "ArmyRose" and "Temps" closely match the palettes of the same name from CARTOColors.

The next group, based on two or three hues, like "PuOr" and "RdYlGn" closely match the palettes of the same name from ColorBrewer.org.

The final group contains "Zissou 1", which closely matches the palette of the same name from the **wesanderson** package (Ram and Wickham 2023), "Cividis", which is an even more colorblind-safe version of "Viridis" (Nuñez, Anderton, and Renslow 2018) and "Roma", which closely matches the palette of the same name from Crameri's scientific color maps.

## 5 New defaults in graphical functions

The new default color palette will be most visible in the output from functions in the **grDevices** and **graphics** packages. Several functions from these packages now have slightly different default output, namely when they are using integer color specifications such as 2 or 3. The resulting colors will still be similar to the old output, e.g., still a red or a green, but just a different shade.

Moreover, a couple of functions explicitly have new defaults: `image()` and `filled.contour()`, now use the sequential "YlOrRd" palette (from ColorBrewer) which uses similar hues as the old `heat.colors()`. See the left panel in Figure 15.

Finally, the `hist()` and `boxplot()` functions (and therefore formula-based calls of the form `plot(num ~ factor, ...)`, also have a new default color: light gray which makes it easier to compare the shaded areas (see the middle and right panels in Figure 15).

```
image(volcano)
boxplot(weight ~ feed, data = chickwts)
hist(chickwts$weight)
```

Package authors may also benefit from the new palettes available in R; the new functions `palette.colors()` and `hcl.colors()` allow good default palettes to be set without requiring additional package dependencies. For example, the **lattice** package has already changed its default colors to use the "Okabe-Ito" and "YlGnBu" palettes (for categorical and numerical data, respectively).
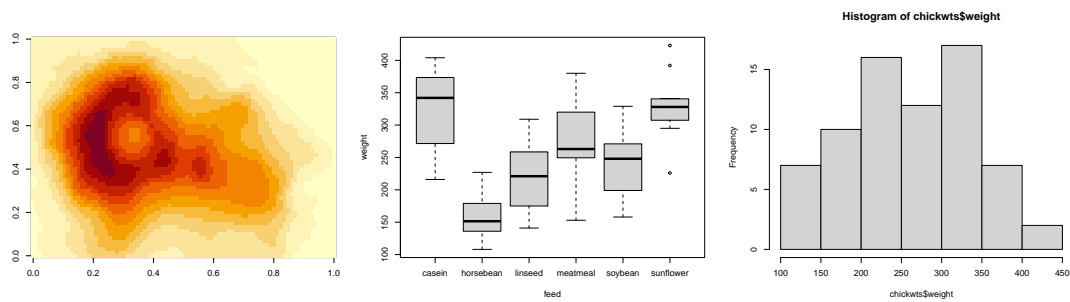
Figure 15: Examples of the new default color palettes that are used in the base graphics functions
`image()`, `boxplot()`, and `hist()`.

6   Summary

The default color palette in R has been improved since R version 4.0.0. The functions `palette.colors()`
and `hcl.colors()`, from the **grDevices** package, also provide a wide range of predefined palettes
based on a number of widely used graphics systems. There are qualitative palettes for use with
categorical data and sequential and diverging palettes for use with ordinal or continuous data. The
table below summarizes the main types of palettes and provides suggestions for good default palettes
for each type. We encourage package authors to make use of these palettes when providing default
colors for functions that produce plots.

Table 1: An overview of the new palette functionality: For each main type of palette, the *Purpose* row
describes what sort of data the type of palette is appropriate for, the *Generate* row gives the functions
that can be used to generate palettes of that type, the *List* row names the functions that can be used to
list available palettes, and the *Robust* row identifies two or three good default palettes of that type.

	Qualitative	Sequential	Diverging
Purpose	Categorical data	Ordered or numeric data (high → low)	Ordered or numeric data with a central value (high ← neutral → low)
Generate	`palette.colors()`, `hcl.colors()`	`hcl.colors()`	`hcl.colors()`
List	`palette.pals()`, `hcl.pals("qualitative")`	`hcl.pals("sequential")`	`hcl.pals("diverging")`, `hcl.pals("divergingx")`
Robust	`"Okabe-Ito"`,`"R4"`	`"Blues 3"`,`"YlGnBu"`, `"Viridis"`	`"Purple-Green"`,`"Blue-Red 3"`

References

Bartram, Lyn, Abhisekh Patra, and Maureen Stone. 2017. "Affective Color in Visualization." In
    Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, 1364–74. New York:
    Association for Computing Machinery. https://doi.org/10.1145/3025453.3026041.

CARTO. 2023. "CARTOColors: Data-Driven Color Schemes." https://carto.com/carto-colors/.

Cleveland, William S., and Robert McGill. 1984. "Graphical Perception: Theory, Experimentation,
    and Application to the Development of Graphical Methods." *Journal of the American Statistical
    Association* 79 (387): 531–54. https://doi.org/10.1080/01621459.1984.10478080.

Coombes, Kevin R., Guy Brock, Zachary B. Abrams, and Lynne V. Abruzzo. 2019. "Polychrome:
    Creating and Assessing Qualitative Palettes with Many Colors." *Journal of Statistical Software, Code
    Snippets* 90 (1): 1–23. https://doi.org/10.18637/jss.v090.c01.

Crameri, Fabio, Grace E. Shephard, and Philip J. Heron. 2020. "The Misuse of Colour in Science
    Communication." *Nature Communications* 11 (1): 5444. https://doi.org/10.1038/s41467-020-
    19160-7.

Etchebehere, Sergio, and Elena Fedorovskaya. 2017. "On the Role of Color in Visual Saliency."
    Electronic Imaging 2017 (14): 58–63. https://doi.org/10.2352/ISSN.2470-1173.2017.14.HVEI-
    119.

Garnier, Simon. 2023. *viridis: Default Color Maps from matplotlib*. https://CRAN.R-project.org/
    package=viridis.

Harrower, Mark A., and Cynthia A. Brewer. 2003. "ColorBrewer.org: An Online Tool for Selecting Color Schemes for Maps." *The Cartographic Journal* 40: 27–37. https://ColorBrewer.org/.

Ihaka, Ross. 2003. "Colour for Presentation Graphics." In *Proceedings of the 3rd International Workshop on Distributed Statistical Computing, Vienna, Austria*, edited by Kurt Hornik, Friedrich Leisch, and Achim Zeileis. https://www.R-project.org/conferences/DSC-2003/Proceedings/.

Ihaka, Ross, Paul Murrell, Kurt Hornik, Jason C. Fisher, Reto Stauffer, Claus O. Wilke, Claire D. McWhite, and Achim Zeileis. 2023. *colorspace: A Toolbox for Manipulating and Assessing Colors and Palettes*. https://CRAN.R-project.org/package=colorspace.

Lonsdale, Maria Dos Santos, and David Lonsdale. 2019. "Design2Inform: Information Visualisation." Report. The Office of the Chief Scientific Advisor, Gov UK. https://eprints.whiterose.ac.uk/141931/.

Machado, Gustavo M., Manuel M. Oliveira, and Leandro A. F. Fernandes. 2009. "A Physiologically-Based Model for Simulation of Color Vision Deficiency." *IEEE Transactions on Visualization and Computer Graphics* 15 (6): 1291–98. https://doi.org/10.1109/TVCG.2009.113.

Neuwirth, Erich. 2022. *RColorBrewer: ColorBrewer Palettes*. https://CRAN.R-project.org/package=RColorBrewer.

Nowosad, Jakub. 2023a. *colorblindcheck: Check Color Palettes for Problems with Color Vision Deficiency*. https://CRAN.R-project.org/package=colorblindcheck.

———. 2023b. *rcartocolor: CARTOColors Palettes*. https://CRAN.R-project.org/package=rcartocolor.

Nuñez, Jamie R., Christopher R. Anderton, and Ryan S. Renslow. 2018. "Optimizing Colormaps with Consideration for Color Cision Deficiency to Enable Accurate Interpretation of Scientific Data." *PLOS ONE* 13 (7): 1–14. https://doi.org/10.1371/journal.pone.0199239.

Okabe, Masataka, and Kei Ito. 2008. "Color Universal Design (CUD): How to Make Figures and Presentations That Are Friendly to Colorblind People." https://jfly.uni-koeln.de/color/.

Pedersen, Thomas Lin, and Fabio Crameri. 2023. *scico: Colour Palettes Based on the Scientific Colour-Maps*. https://CRAN.R-project.org/package=scico.

Ram, Karthik, and Hadley Wickham. 2023. *wesanderson: A Wes Anderson Palette Generator*. https://CRAN.R-project.org/package=wesanderson.

Sarkar, Deepayan. 2008. *lattice: Multivariate Data Visualization with R*. New York: Springer-Verlag. https://lmdvr.R-Forge.R-project.org/.

Smith, Nathaniel, and Stéfan Van der Walt. 2015. "A Better Default Colormap for matplotlib." In *SciPy 2015 – Scientific Computing with Python*. Austin. https://www.youtube.com/watch?v=xAoljeRJ3lU.

Stone, Maureen. 2016. "How We Designed the New Color Palettes in Tableau 10." https://www.tableau.com/about/blog/2016/7/colors-upgrade-tableau-10-56782.

Tableau Software, LLC. 2021. "Tableau: Business Intelligence and Analytics Software." https://www.tableau.com/.

Tennekes, Martijn. 2023. *cols4all: Colors for All*. https://CRAN.R-project.org/package=cols4all.

Ware, Colin. 2012. *Information Visualization: Perception for Design*. 3rd ed. Morgan Kaufmann. https://doi.org/10.1016/C2009-0-62432-6.

Wickham, Hadley. 2016. *ggplot2: Elegant Graphics for Data Analysis*. 2nd ed. Springer-Verlag. https://ggplot2.tidyverse.org/.

Wikipedia. 2023. "HCL Color Space — Wikipedia, the Free Encyclopedia." URL https://en.wikipedia.org/wiki/HCL_color_space, accessed 2023-11-20.

Zeileis, Achim, Jason C. Fisher, Kurt Hornik, Ross Ihaka, Claire D. McWhite, Paul Murrell, Reto Stauffer, and Claus O. Wilke. 2020. "colorspace: A Toolbox for Manipulating and Assessing Colors and Palettes." *Journal of Statistical Software* 96 (1): 1–49. https://doi.org/10.18637/jss.v096.i01.

Zeileis, Achim, Kurt Hornik, and Paul Murrell. 2009. "Escaping RGBland: Selecting Colors for Statistical Graphics." *Computational Statistics & Data Analysis* 53: 3259–70. https://doi.org/10.1016/j.csda.2008.11.033.

Zeileis, Achim, and Paul Murrell. 2019. "HCL-Based Color Palettes in grDevices." https://blog.R-project.org/2019/04/01/hcl-based-color-palettes-in-grdevices/.

Zeileis, Achim, Paul Murrell, Martin Maechler, and Deepayan Sarkar. 2019. "A New palette() for R." https://blog.R-project.org/2019/11/21/a-new-palette-for-r/.

Zeileis, Achim, and Reto Stauffer. 2019. "Was Trump Confused by Rainbow Color Map?" https://www.zeileis.org/news/dorian_rainbow/.

Achim Zeileis
Universität Innsbruck
Department of Statistics
https://www.zeileis.org/
ORCiD: 0000-0003-0918-3766
Achim.Zeileis@R-project.org

Paul Murrell
University of Auckland
Department of Statistics
https://www.stat.auckland.ac.nz/~paul/
ORCiD: 0000-0002-3224-8858
paul@stat.auckland.ac.nz