

Data Visualization in R with **ggplot2**

The Structure of ggplot2 (Part 2)

Cédric Scherer

Physalia Courses | November 9-13 2020

Photo by Richard Strozyński

Part 2

Scales & Coordinate Systems

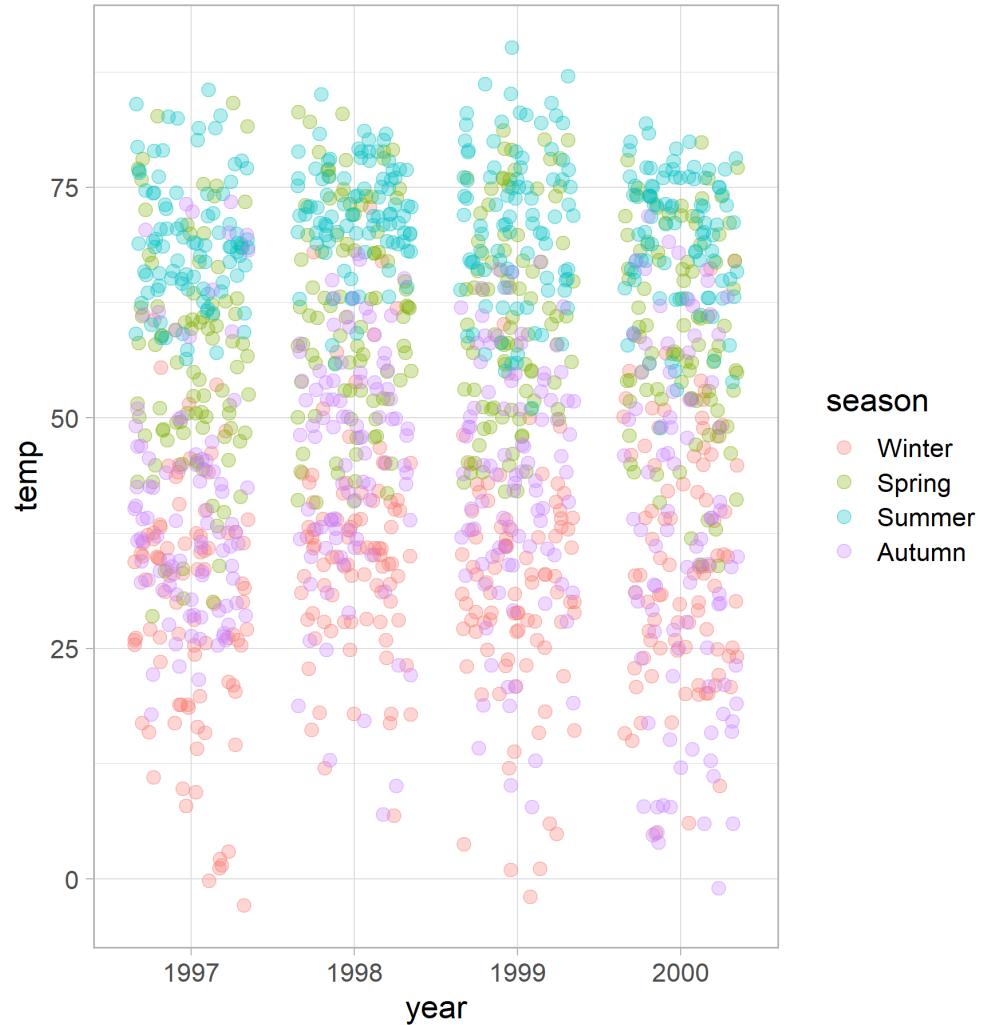
ggplot2: Build a data MASTERpiece



Illustration by Allison Horst (github.com/allisonhorst/stats-illustrations)

A Masterpiece?

```
chic <- readr::read_csv("https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2020/2020-01-07/Chicago.csv")  
chic$year <- as.numeric(gsub("(.*)-(\\d{2})-(\\d{2})", "\\1-\\3-\\2", chic$date))  
  
(g <-  
  ggplot(chic,  
    aes(  
      year,  
      temp,  
      color = season  
    )) +  
  geom_jitter(  
    size = 3,  
    alpha = .3,  
    position = position_jitter(  
      width = .35,  
      seed = 2020  
    )  
  )  
)
```



The Structure of `ggplot2`

Layer	Function	Explanation
Data	<code>ggplot(data)</code>	The raw data that you want to visualise.
Aesthetics	<code>aes()</code>	Aesthetic mappings of the geometric and statistical objects.
Layers	<code>geom_*</code> () and <code>stat_*</code> ()	The geometric shapes and statistical summaries representing the data.
Scales	<code>scale_*</code> ()	Maps between the data and the aesthetic dimensions.
Coordinate System	<code>coord_*</code> ()	Maps data into the plane of the data rectangle.
Facets	<code>facet_*</code> ()	The arrangement of the data into a grid of plots.
Visual Themes	<code>theme()</code> and <code>theme_*</code> ()	The overall visual defaults of a plot.
Annotations	<code>annotate()</code>	Add additional labels, geometries or images to a plot.

Scales

`scale_*`

`scale_*`()

One can use `scale_*`() to change properties of all the **aesthetic dimensions mapped to the data**.

Consequently, there are `scale_*`() functions for all aesthetics such as:

- **positions** via `scale_x_*`() and `scale_y_*`()
- **colors** via `scale_color_*`() and `scale_fill_*`()
- **sizes** via `scale_size_*`() and `scale_radius_*`()
- **shapes** via `scale_shape_*`() and `scale_linetype_*`()
- **transparency** via `scale_alpha_*`()

scale_*()

One can use `scale_*()` to change properties of all the **aesthetic dimensions mapped to the data**

The extensions (`*`) can be filled by e.g.:

- `continuous()`, `discrete()`, `reverse()`, `log10()`, `sqrt()`, `date()` for positions
- `continuous()`, `discrete()`, `manual()`, `gradient()`, `gradient2()`, `brewer()` for colors
- `continuous()`, `discrete()`, `manual()`, `ordinal()`, `area()`, `date()` for sizes
- `continuous()`, `discrete()`, `manual()`, `ordinal()` for shapes
- `continuous()`, `discrete()`, `manual()`, `ordinal()`, `date()` for transparency

Scales

`scale_x|y_*`()

CONTINUOUS

measured data, can have ∞ values within possible range.



I AM 3.1" TALL
I WEIGH 34.16 grams

DISCRETE

OBSERVATIONS CAN ONLY EXIST AT LIMITED VALUES, OFTEN COUNTS.

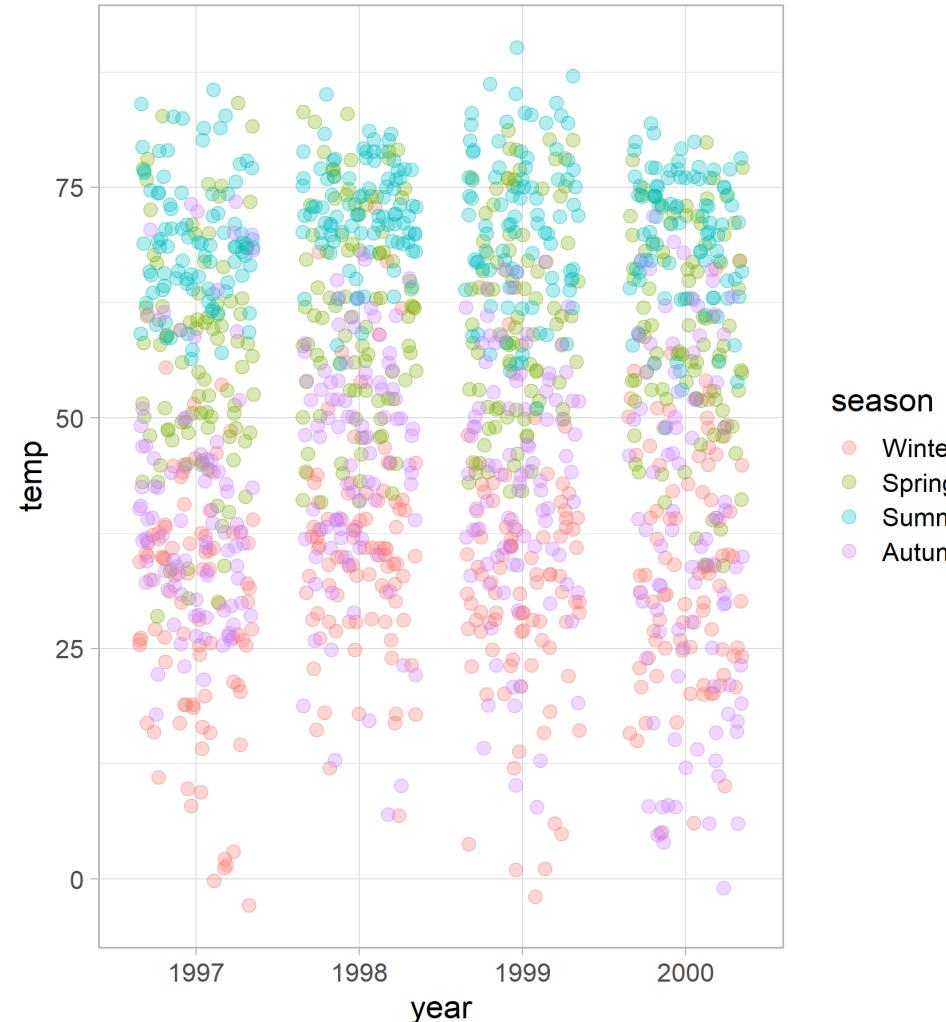


I HAVE 8 LEGS
and
4 SPOTS!

@allison_horst

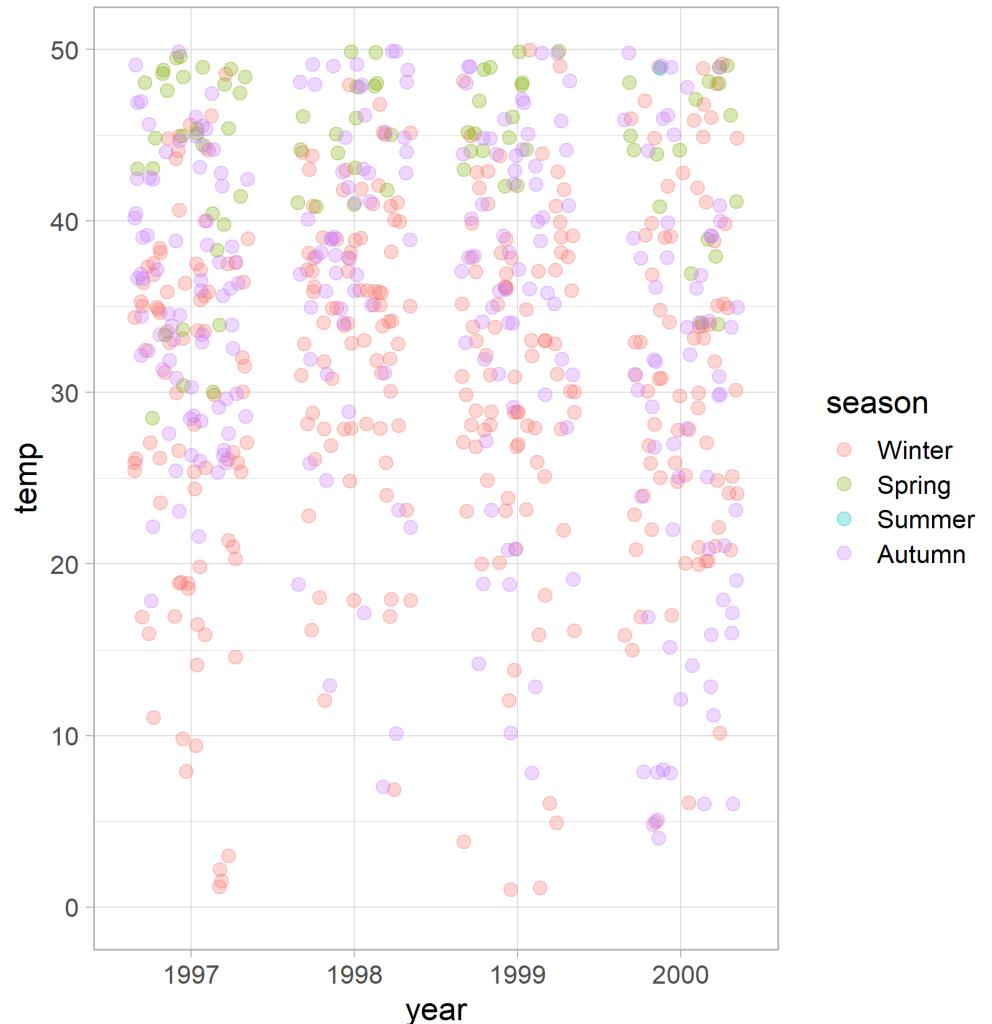
scale_x|y_continuous()

```
gg
```



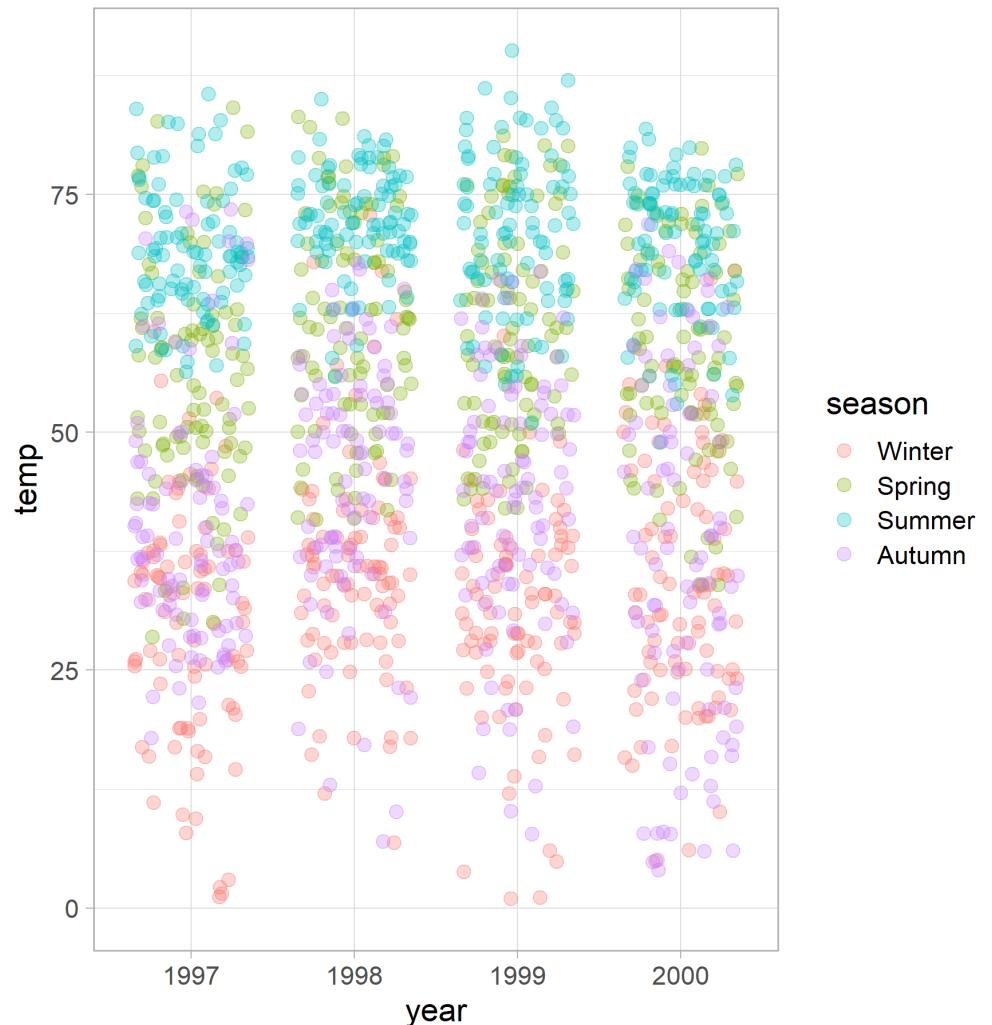
scale_x|y_continuous()

```
g +  
  scale_y_continuous(  
    limits = c(0, 50)  
)
```



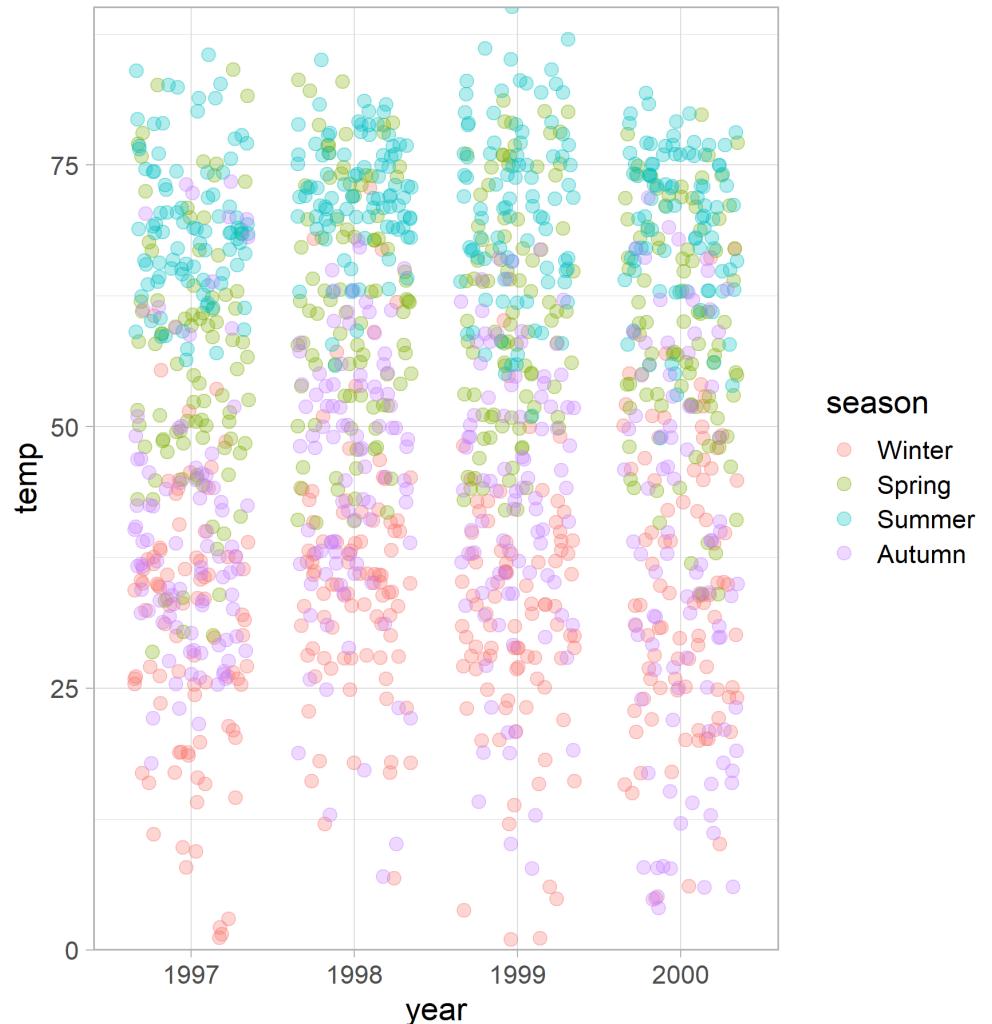
scale_x|y_continuous()

```
g +  
  scale_y_continuous(  
    limits = c(0, NA)  
)
```



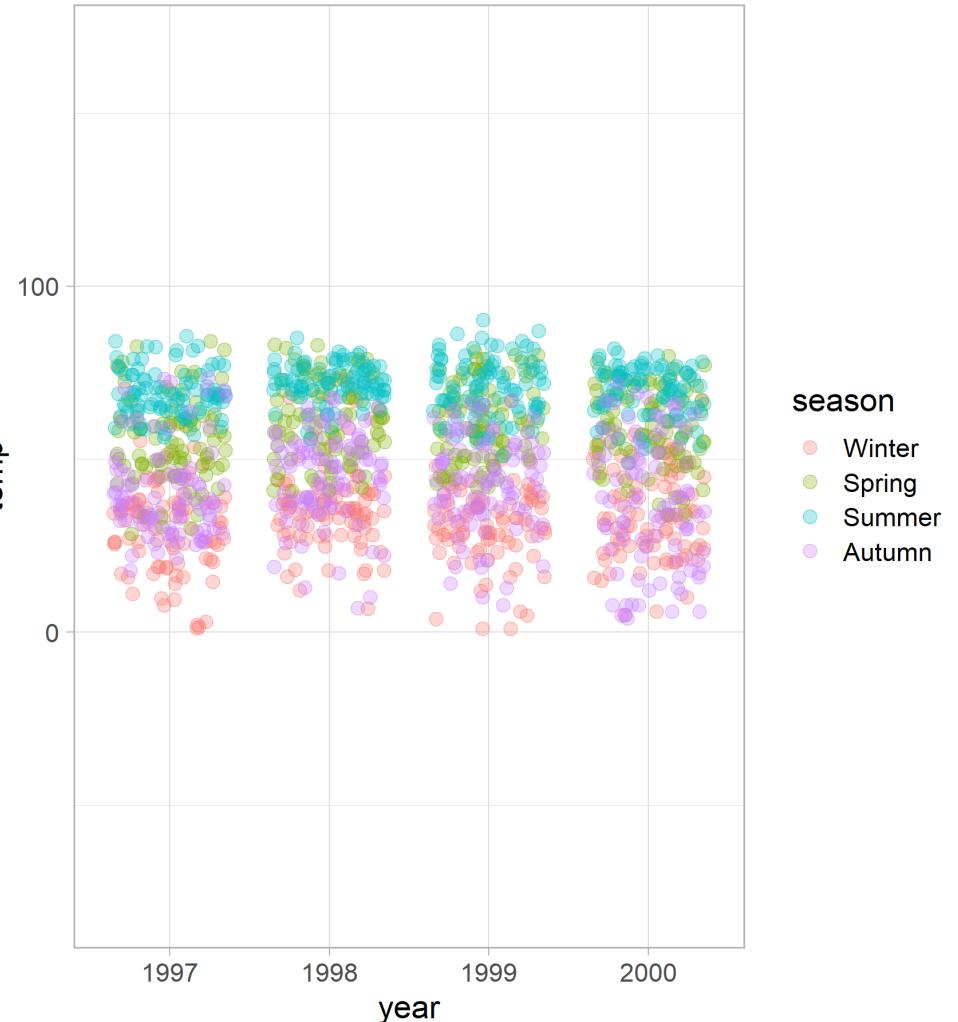
scale_x|y_continuous()

```
g +  
  scale_y_continuous(  
    limits = c(0, NA),  
    expand = c(0, 0)  
)
```



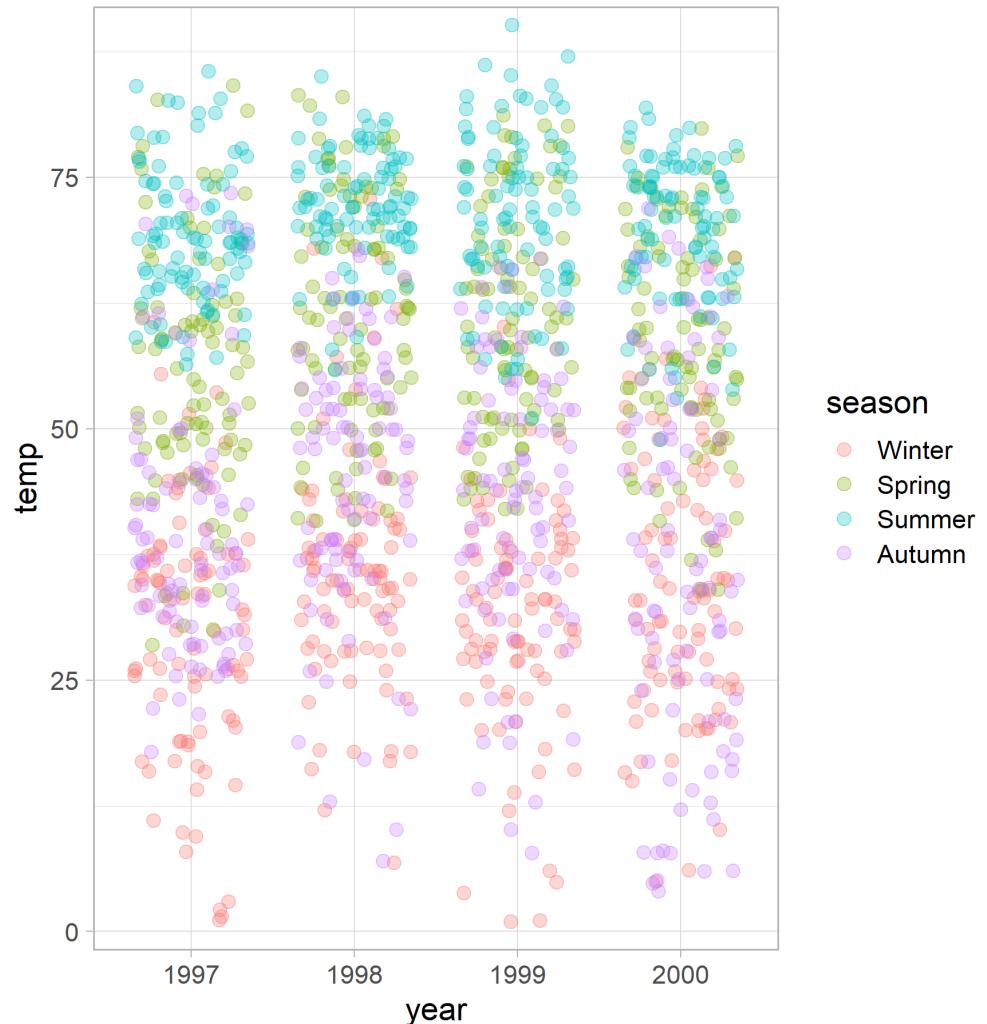
scale_x|y_continuous()

```
g +  
  scale_y_continuous(  
    limits = c(0, NA),  
    expand = c(1, 1)  
)
```



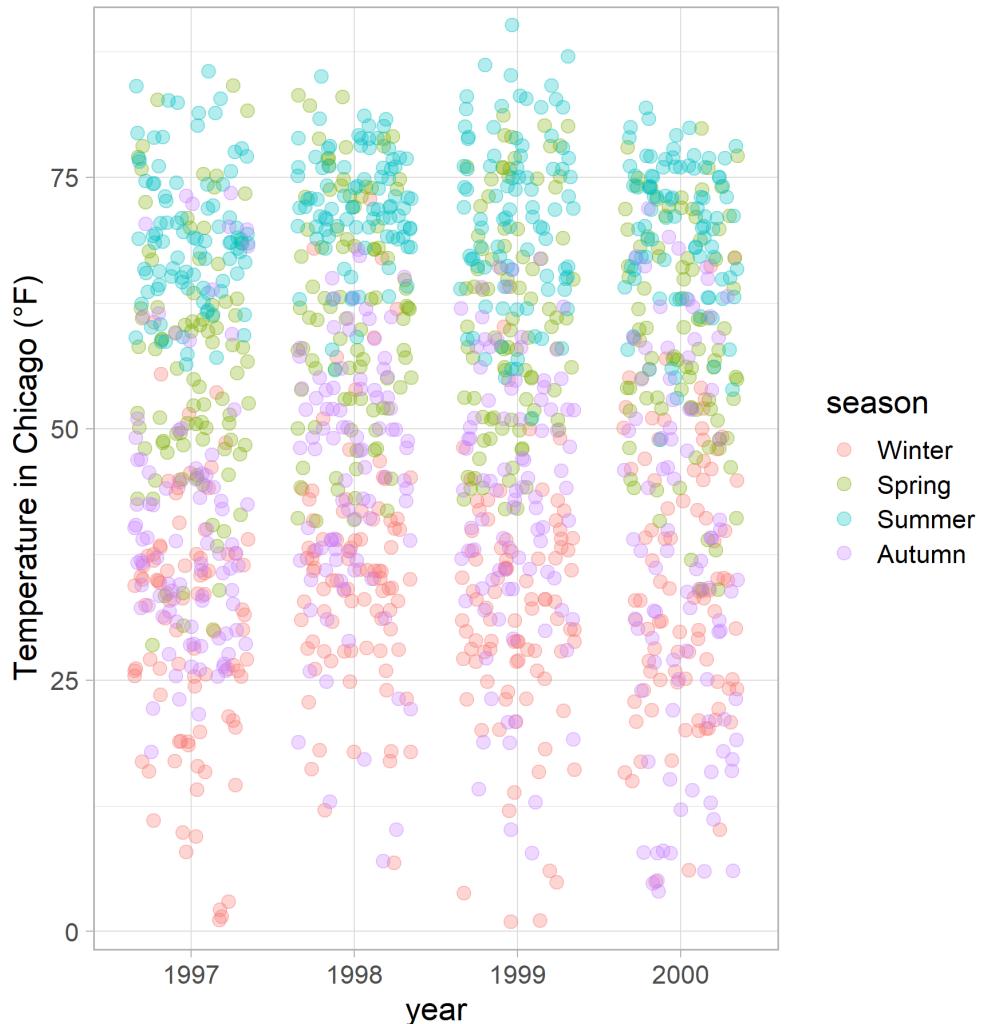
scale_x|y_continuous()

```
g +  
  scale_y_continuous(  
    limits = c(0, NA),  
    expand = c(.02, .02)  
)
```



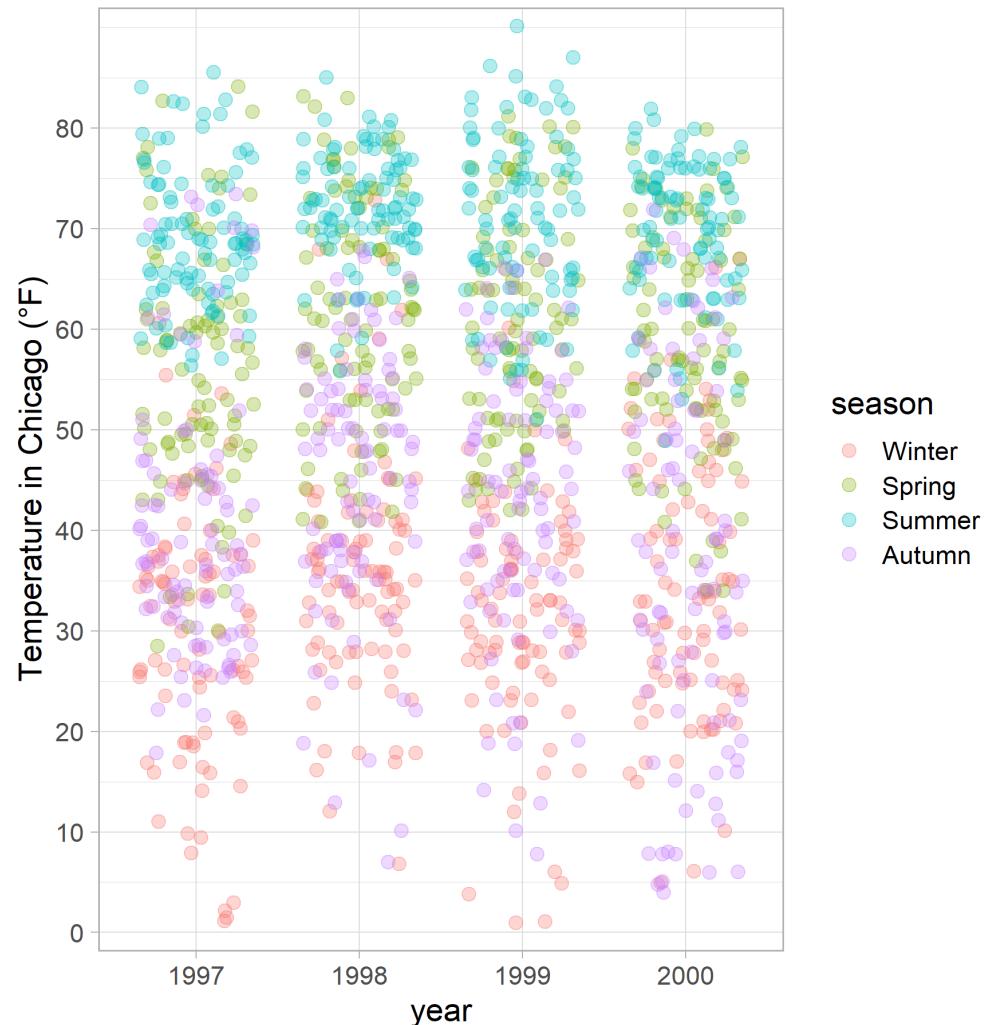
scale_x|y_continuous()

```
g +
  scale_y_continuous(
    name = "Temperature in Chicago (°F)",
    limits = c(0, NA),
    expand = c(.02, .02)
  )
```



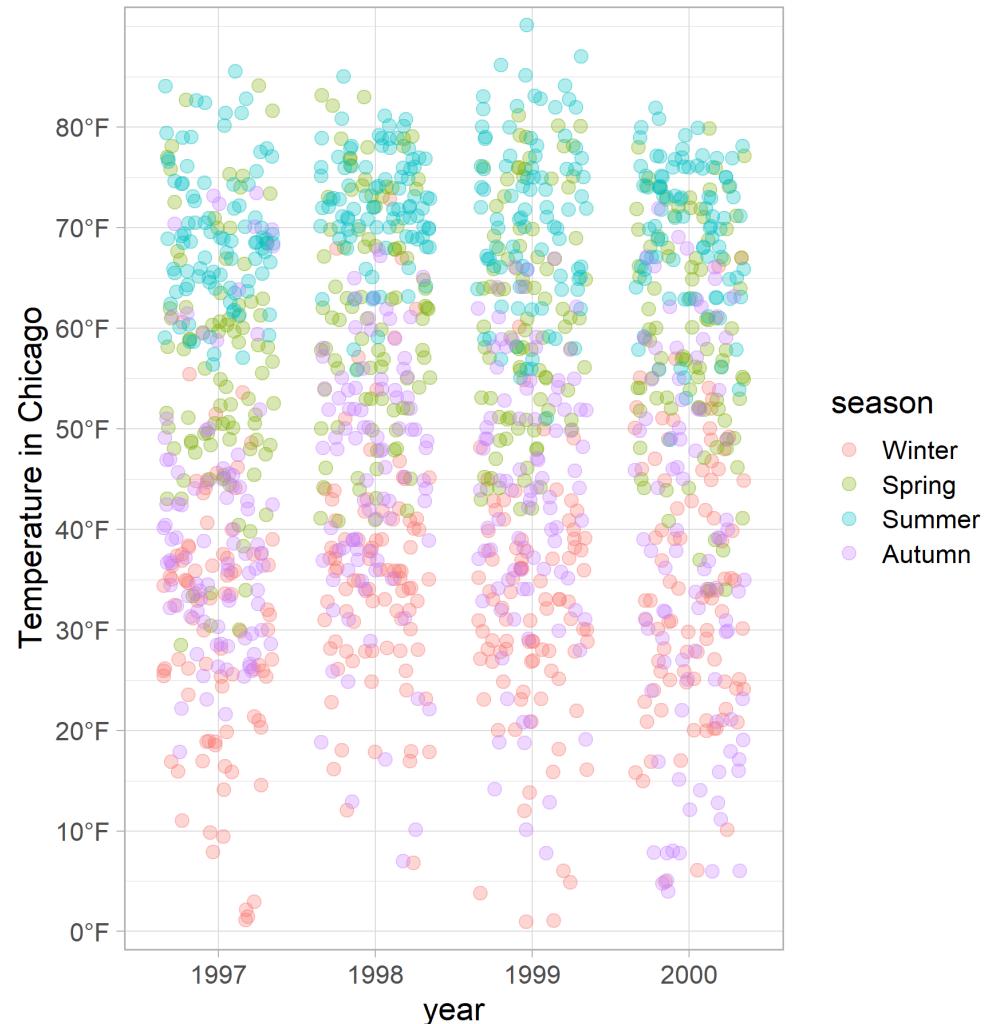
scale_x|y_continuous()

```
g +
  scale_y_continuous(
    name = "Temperature in Chicago (°F)",
    limits = c(0, NA),
    expand = c(.02, .02),
    breaks = seq(0, 80, by = 10)
  )
```



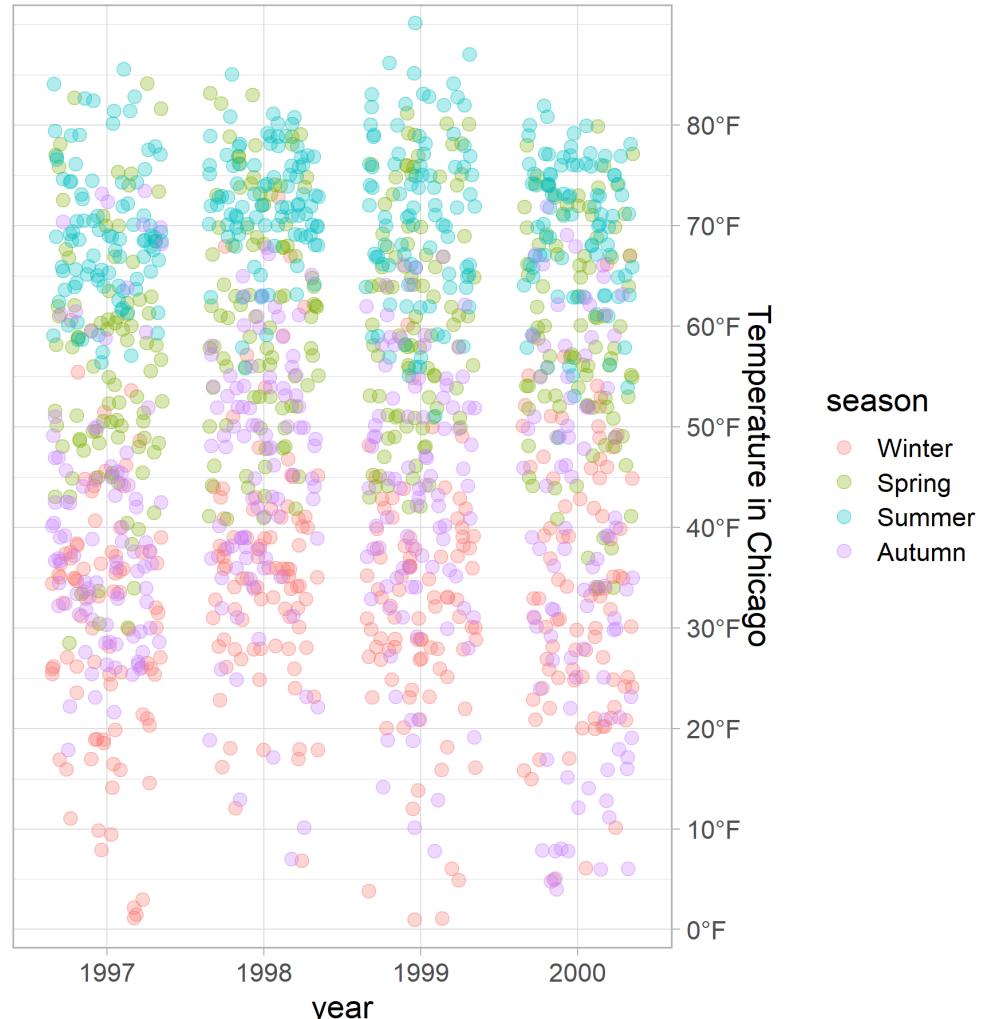
scale_x|y_continuous()

```
g +
  scale_y_continuous(
    name = "Temperature in Chicago",
    limits = c(0, NA),
    expand = c(.02, .02),
    breaks = seq(0, 80, by = 10),
    labels = glue::glue(
      "{seq(0, 80, by = 10)}°F"
    )
  )
```



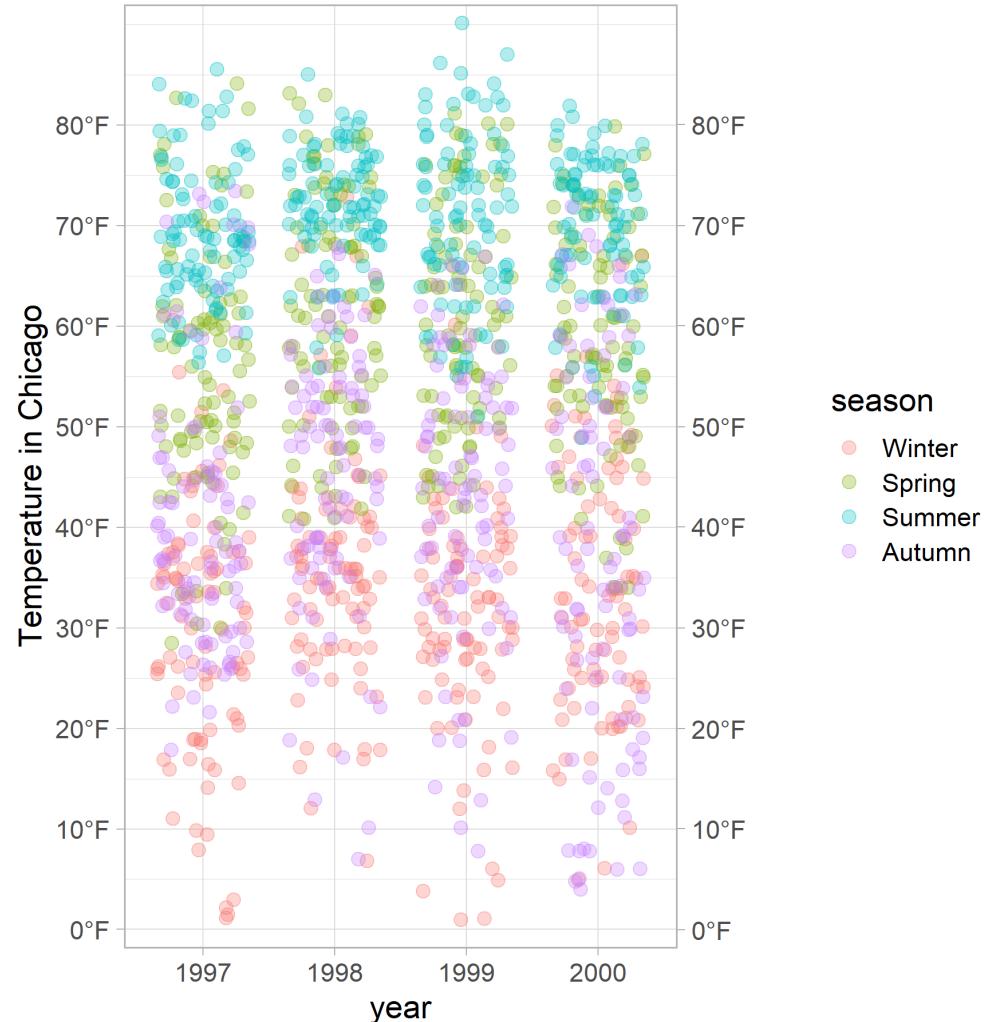
scale_x|y_continuous()

```
g +
  scale_y_continuous(
    name = "Temperature in Chicago",
    limits = c(0, NA),
    expand = c(.02, .02),
    breaks = seq(0, 80, by = 10),
    labels = glue::glue(
      "{seq(0, 80, by = 10)}°F"
    ),
    position = "right"
  )
```



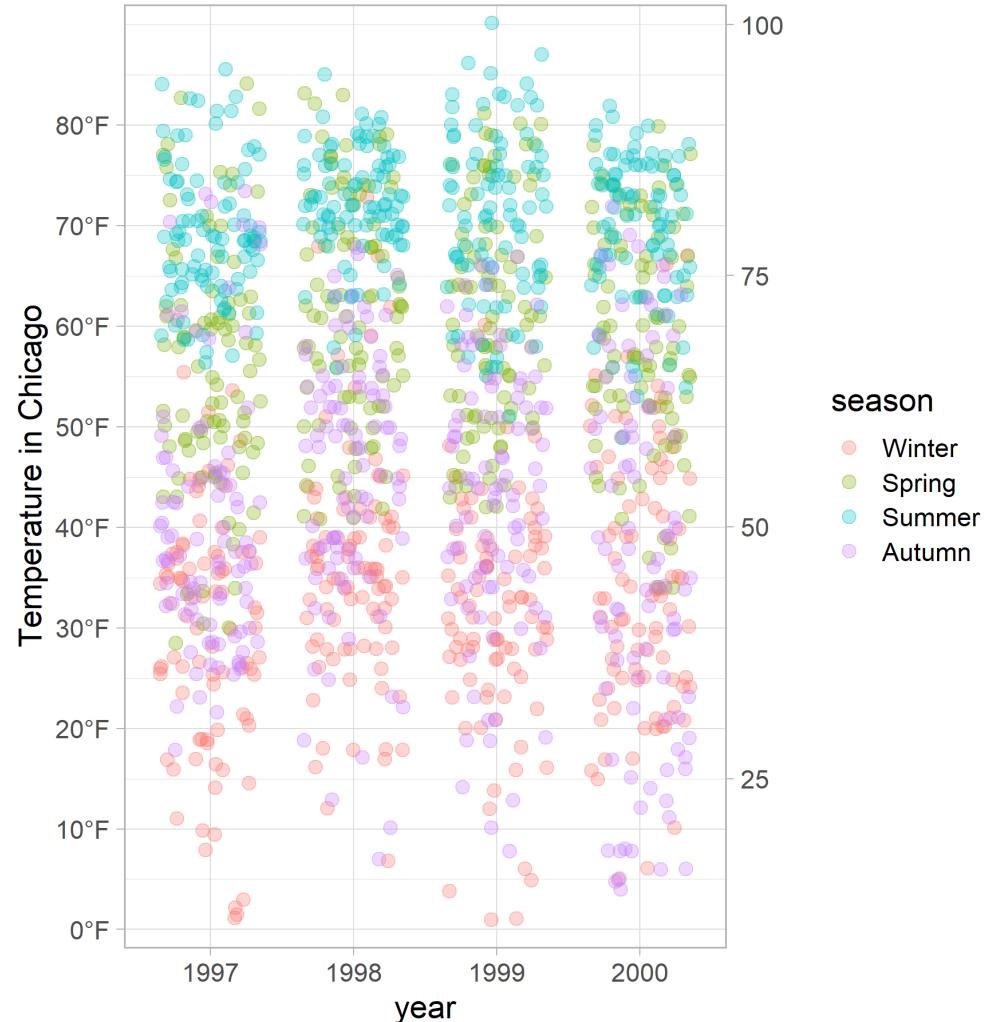
scale_x|y_continuous()

```
g +
  scale_y_continuous(
    name = "Temperature in Chicago",
    limits = c(0, NA),
    expand = c(.02, .02),
    breaks = seq(0, 80, by = 10),
    labels = glue::glue(
      "{seq(0, 80, by = 10)}°F"
    ),
    sec.axis = dup_axis(name = "")
  )
```



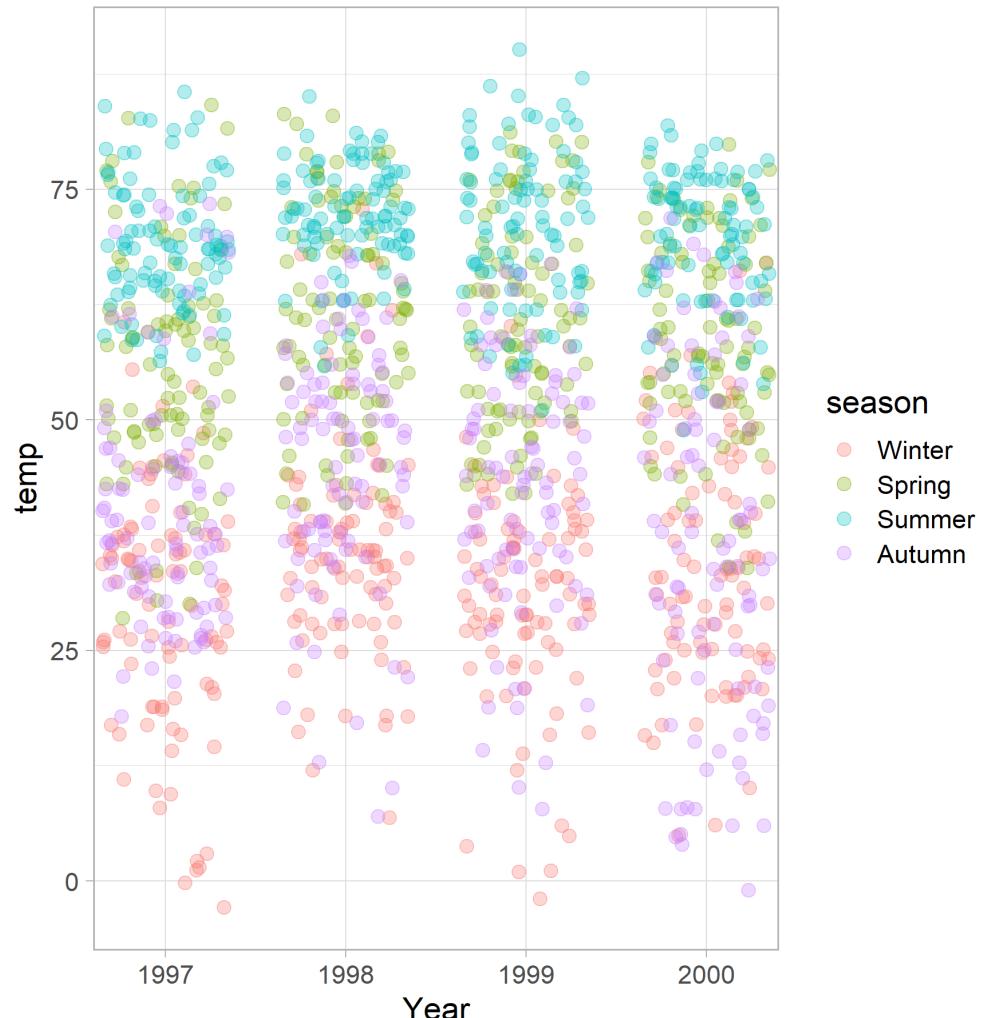
scale_x|y_continuous()

```
g +
  scale_y_continuous(
    name = "Temperature in Chicago",
    limits = c(0, NA),
    expand = c(.02, .02),
    breaks = seq(0, 80, by = 10),
    labels = glue::glue(
      "{seq(0, 80, by = 10)}°F"
    ),
    sec.axis = sec_axis(~ . + 10)
  )
```



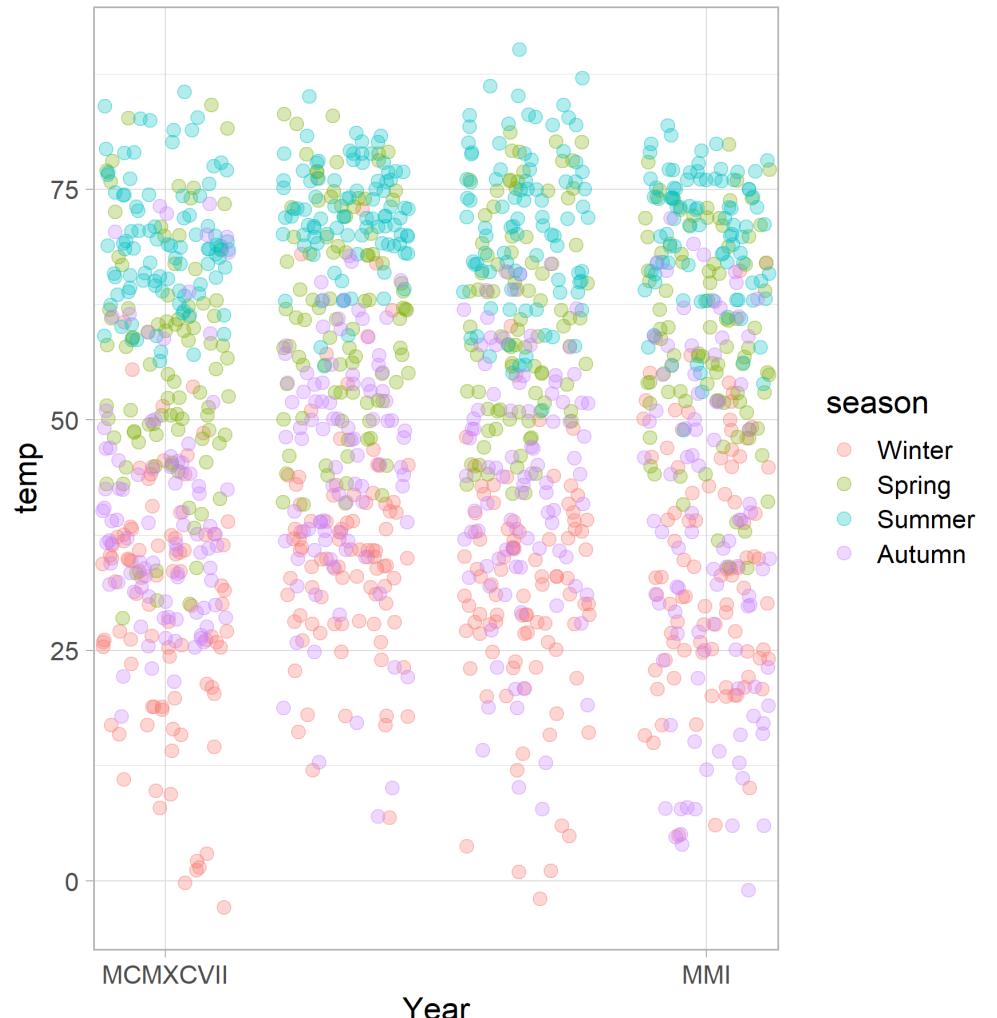
scale_x|y_discrete()

```
g +
  scale_x_discrete(
    name = "Year",
    expand = c(.1, .1)
  )
```



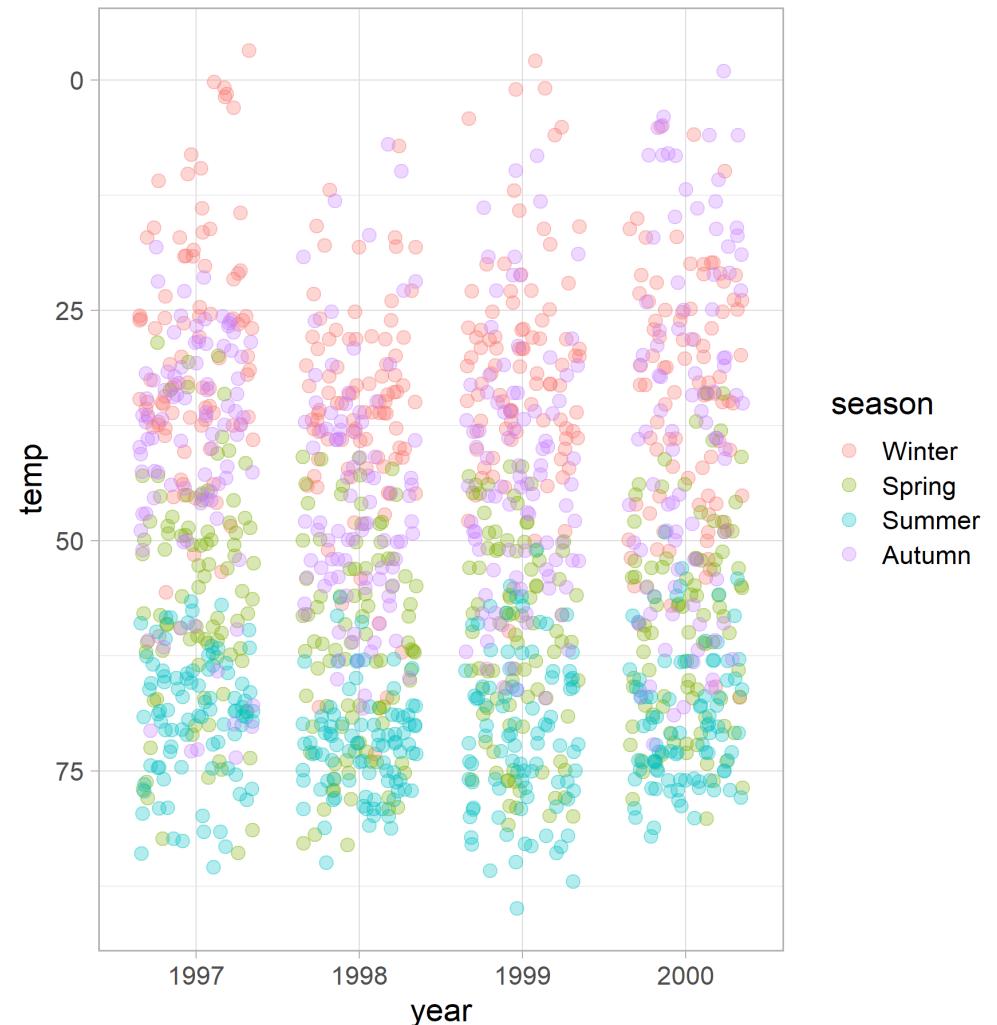
scale_x|y_discrete()

```
g +
  scale_x_discrete(
    name = "Year",
    expand = c(.1, .1),
    breaks = c(1997, 2000),
    labels = c("MCMXCVII", "MMI")
  )
```



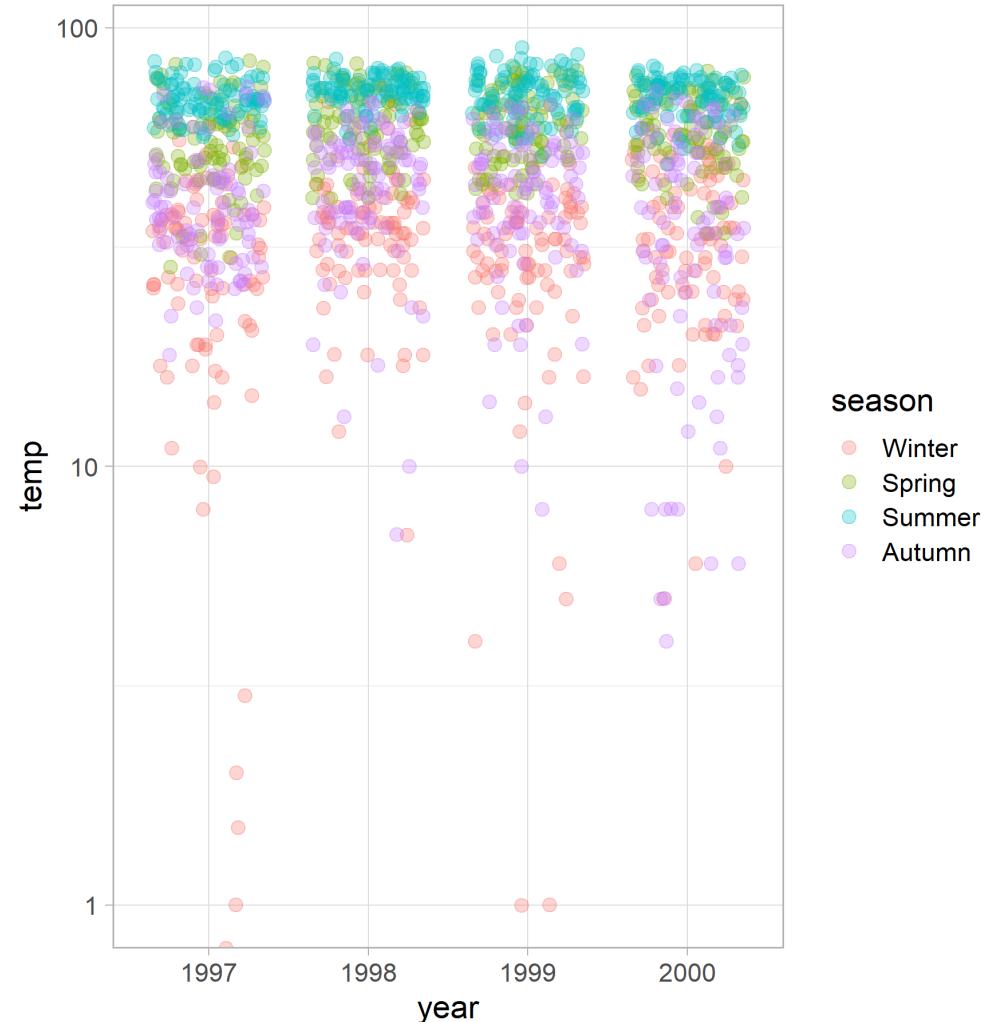
scale_x|y_reverse()

```
g +  
  #scale_x_reverse() +  
  ## doesn't work with factors  
  scale_y_reverse()
```



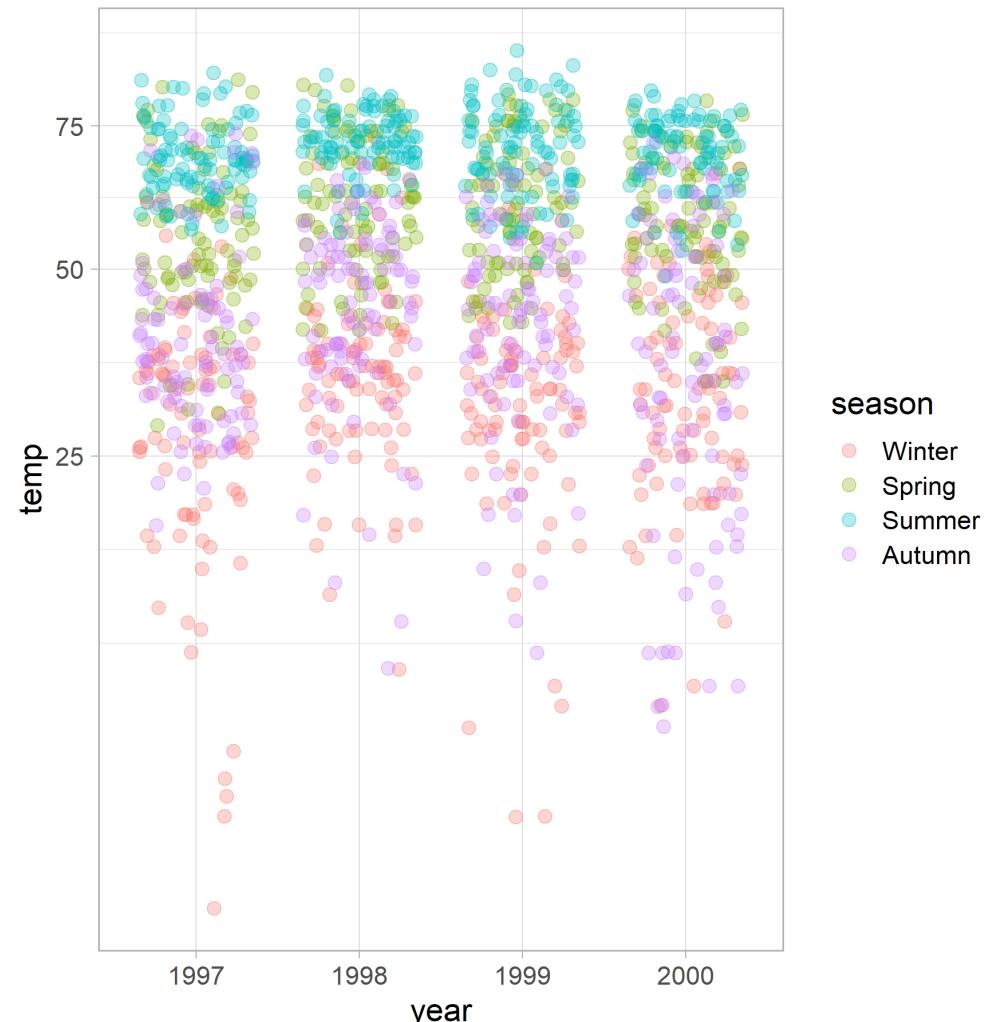
scale_x|y_log10()

```
g +  
  #scale_x_log10() +  
  ## doesn't work with factors  
  scale_y_log10()
```



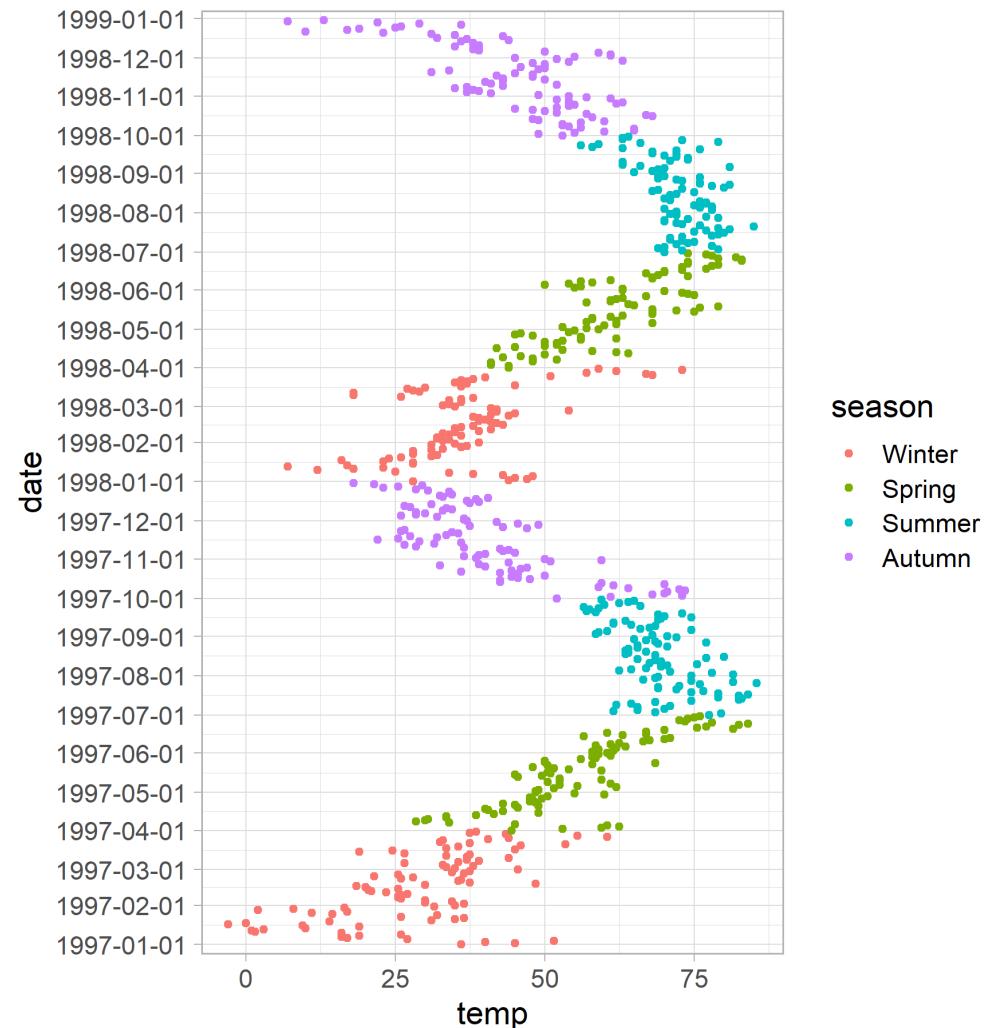
scale_x|y_sqrt()

```
g +
  #scale_x_sqrt() +
  ## doesn't work with factors
  scale_y_sqrt()
```



scale_x|y_date()

```
chic %>%
  filter(year %in% 1997:1998) %>%
  ggplot(aes(temp, date)) +
  geom_point(aes(color = season)) +
  scale_y_date(
    date_breaks = "1 month",
    date_labels = "%Y-%m-%d",
    expand = c(.01, .01)
  )
```

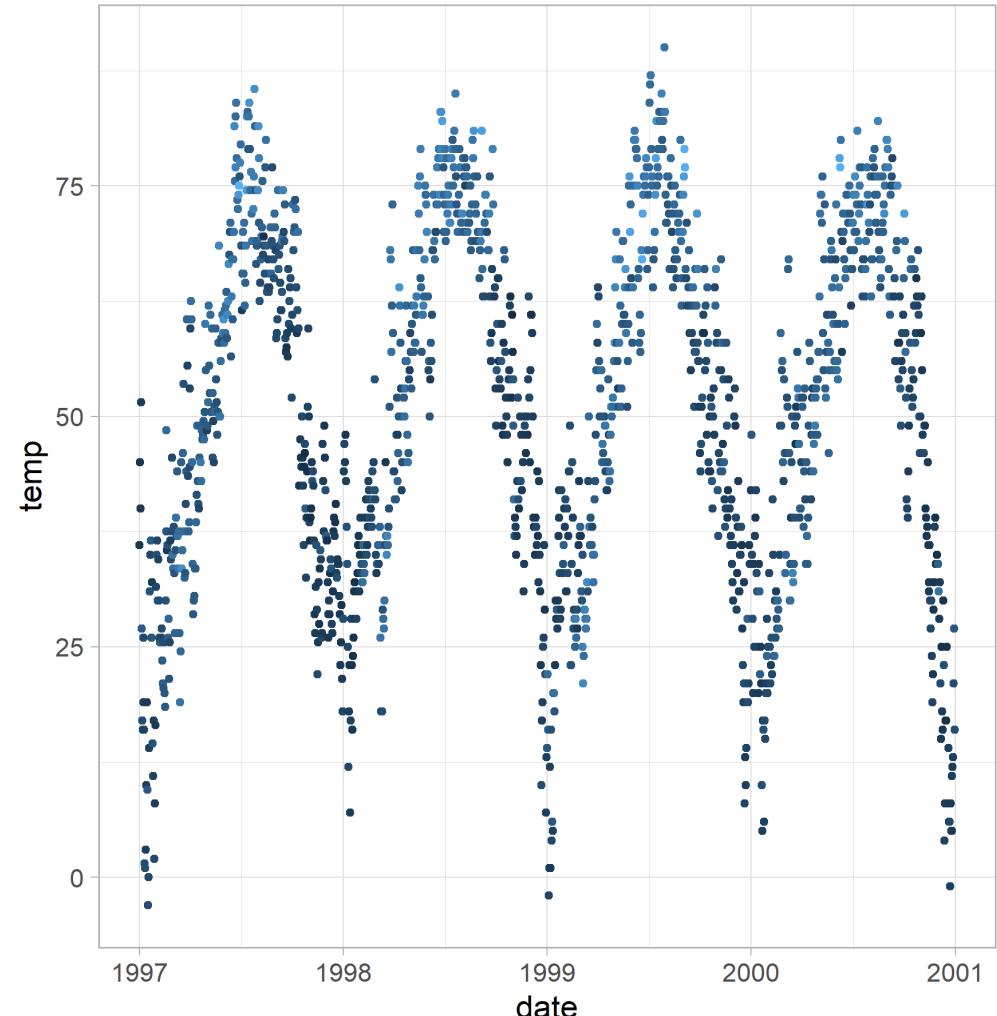


Scales

`scale_color|fill_*`

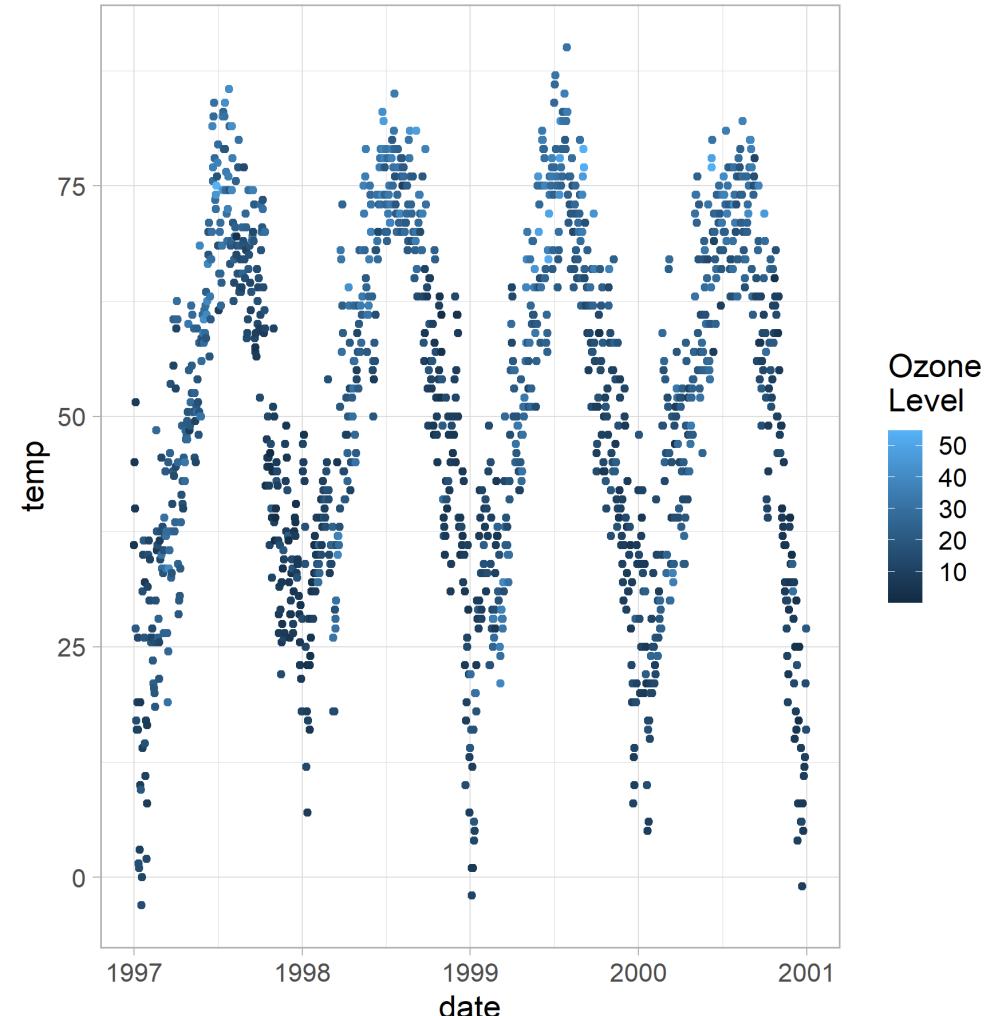
scale_color|fill_continuous()

```
ggplot(chic, aes(date, temp)) +  
  geom_point(aes(color = o3)) +  
  scale_color_continuous(  
    guide = "none"  
)
```



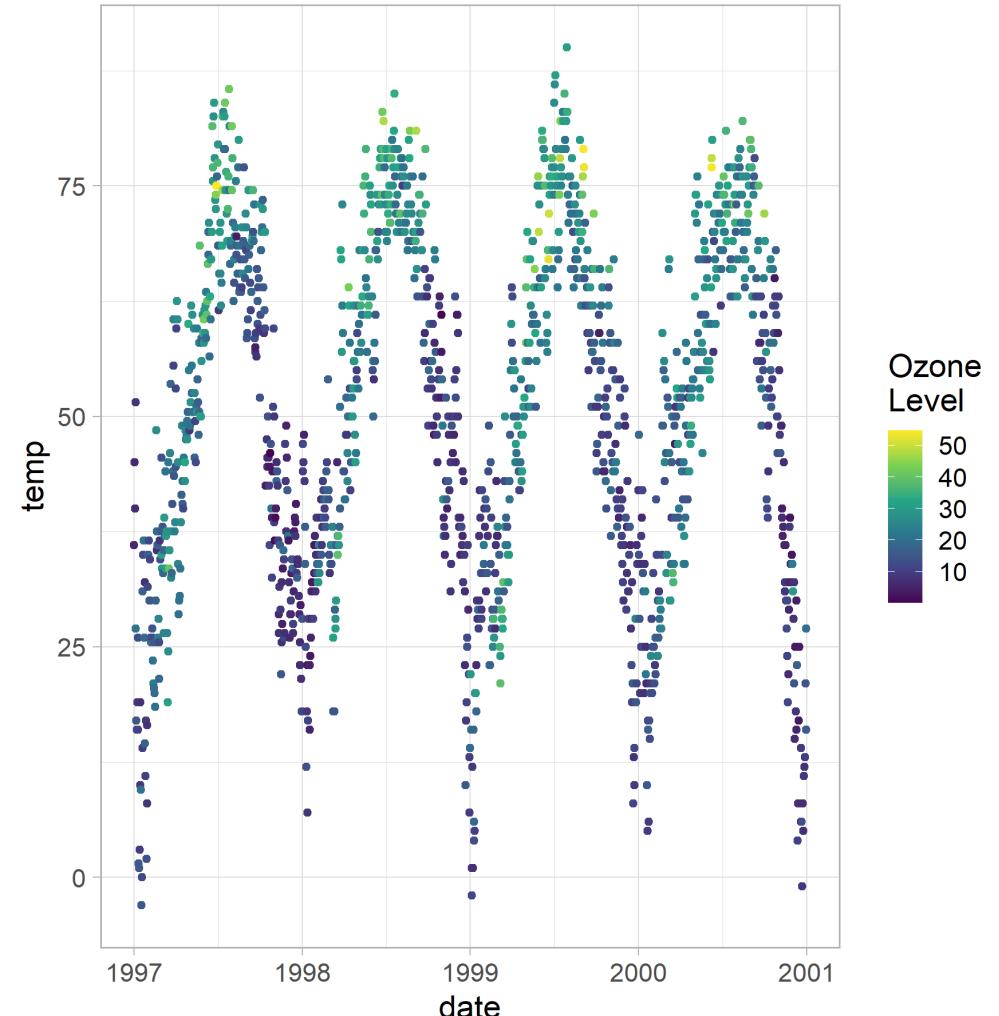
scale_color|fill_continuous()

```
ggplot(chic, aes(date, temp)) +  
  geom_point(aes(color = o3)) +  
  scale_color_continuous(  
    name = "Ozone\nLevel"  
)
```



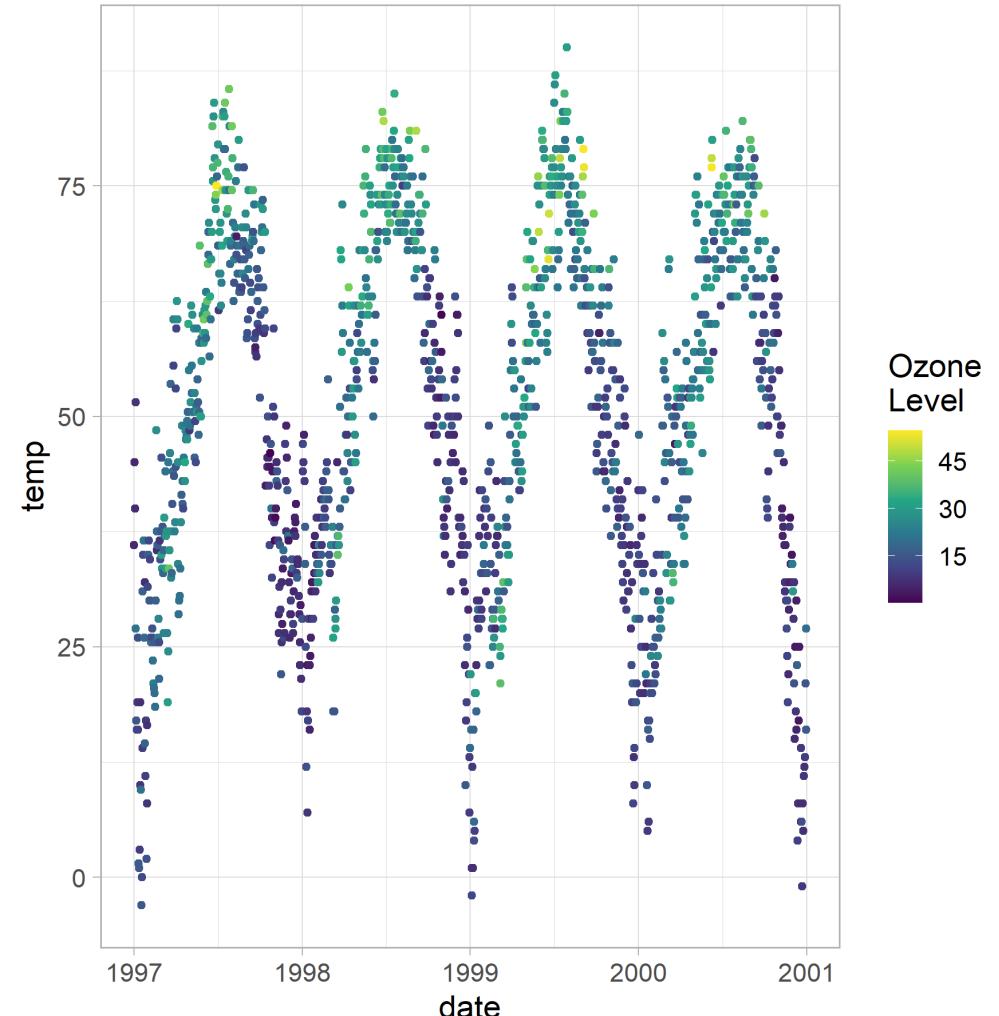
scale_color|fill_continuous()

```
ggplot(chic, aes(date, temp)) +  
  geom_point(aes(color = o3)) +  
  scale_color_continuous(  
    name = "Ozone\nLevel",  
    type = "viridis"  
)
```



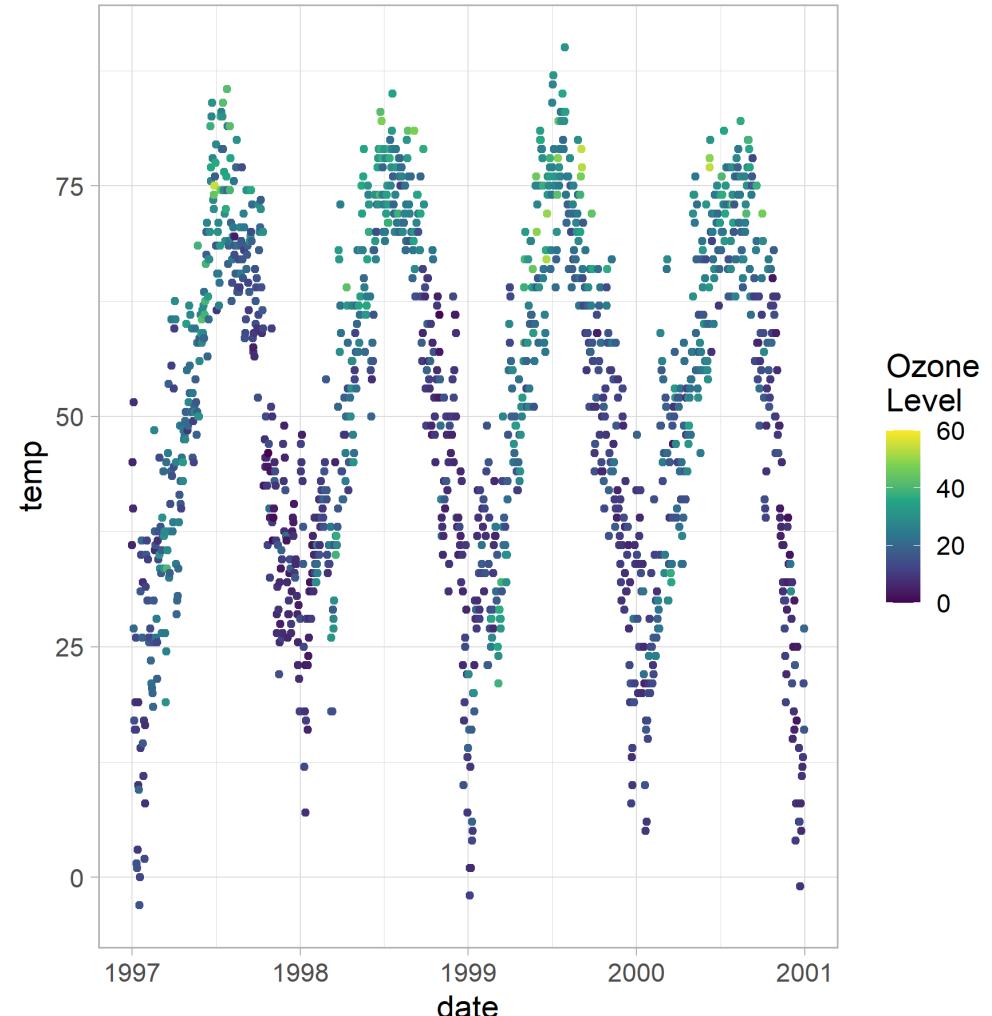
scale_color|fill_continuous()

```
ggplot(chic, aes(date, temp)) +  
  geom_point(aes(color = o3)) +  
  scale_color_continuous(  
    name = "Ozone\nLevel",  
    type = "viridis",  
    breaks = seq(0, 60, length.out = 5)  
)
```



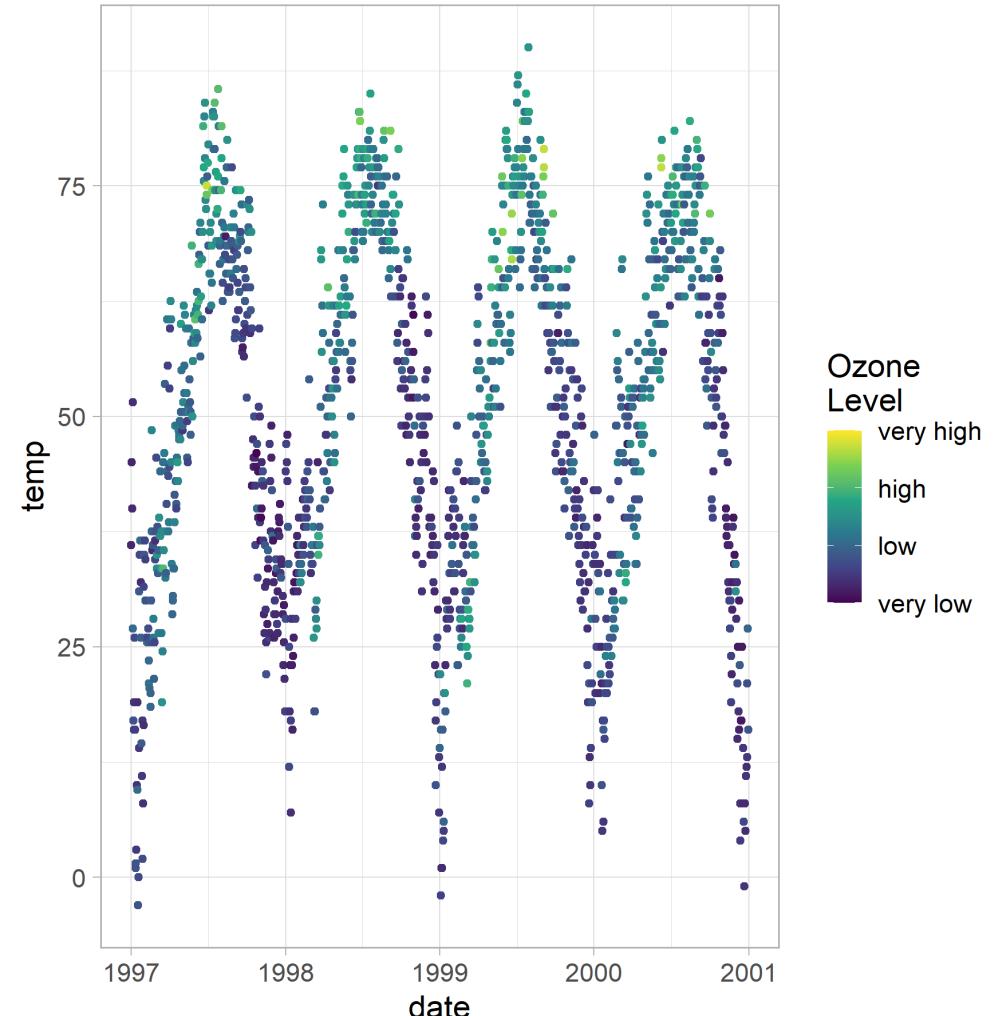
scale_color|fill_continuous()

```
ggplot(chic, aes(date, temp)) +  
  geom_point(aes(color = o3)) +  
  scale_color_continuous(  
    name = "Ozone\\nLevel",  
    type = "viridis",  
    breaks = seq(0, 60, length.out = 4),  
    limits = c(0, 60)  
)
```



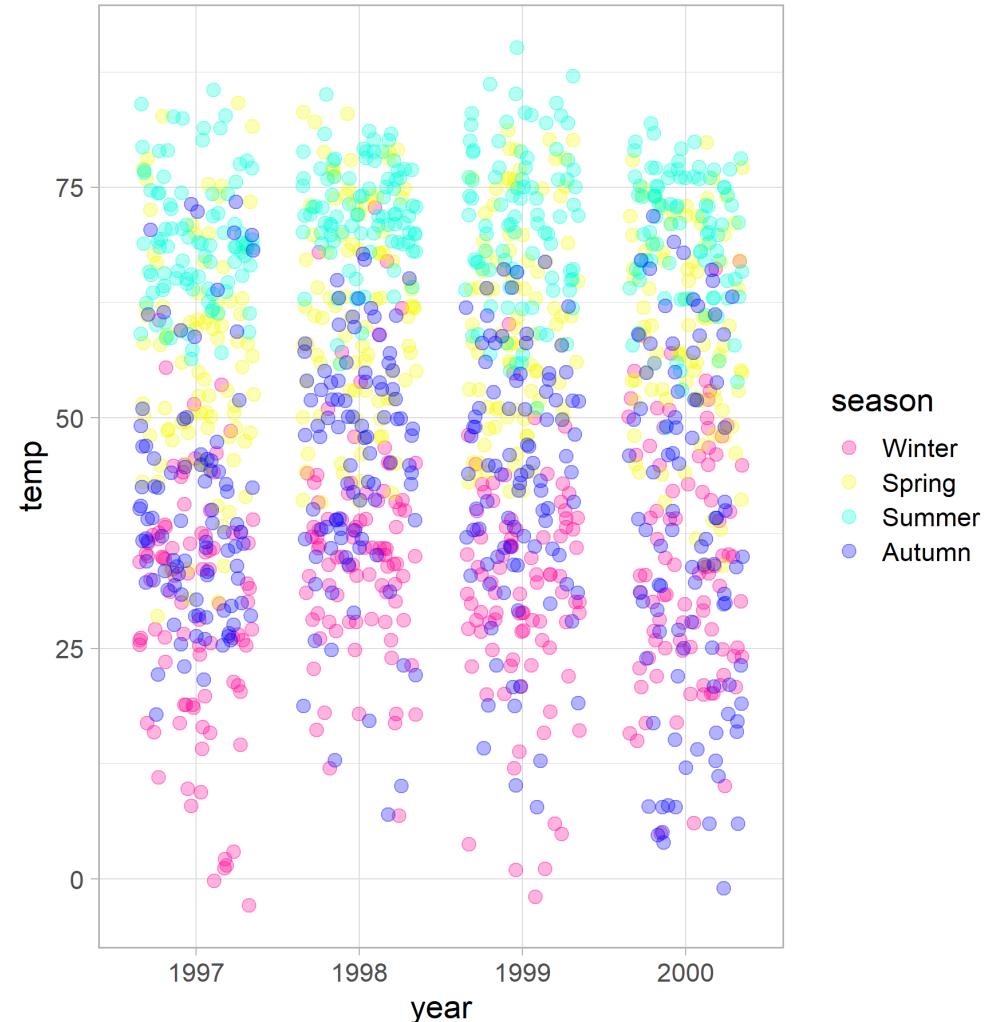
scale_color|fill_continuous()

```
ggplot(chic, aes(date, temp)) +  
  geom_point(aes(color = o3)) +  
  scale_color_continuous(  
    name = "Ozone\\nLevel",  
    type = "viridis",  
    breaks = seq(0, 60, length.out = 4),  
    limits = c(0, 60),  
    labels = c("very low", "low",  
              "high", "very high"))
```



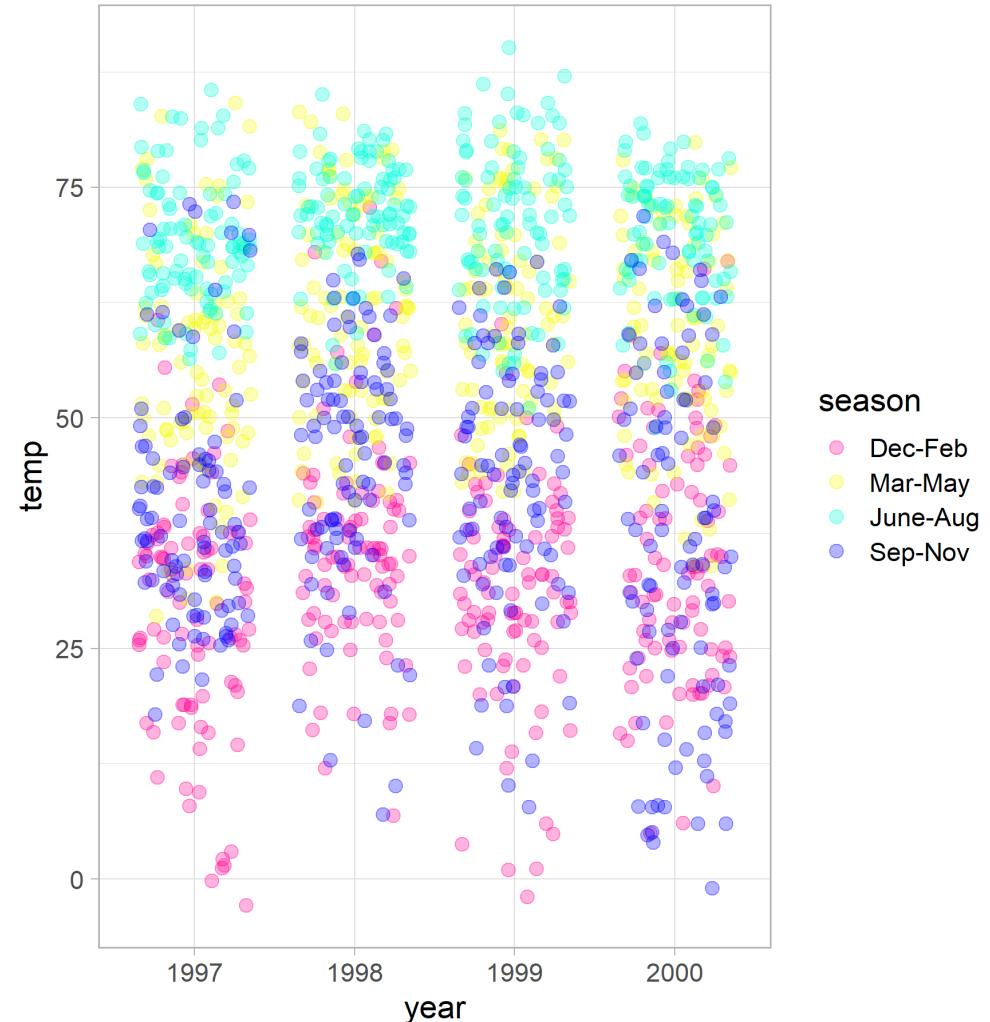
scale_color|fill_discrete()

```
g +
  scale_color_discrete(
    h = c(0, 260), ## hue: [0,360]
    c = 500, ## chroma: [0,x]
    l = 90 ## luminance: [0,100]
  )
```



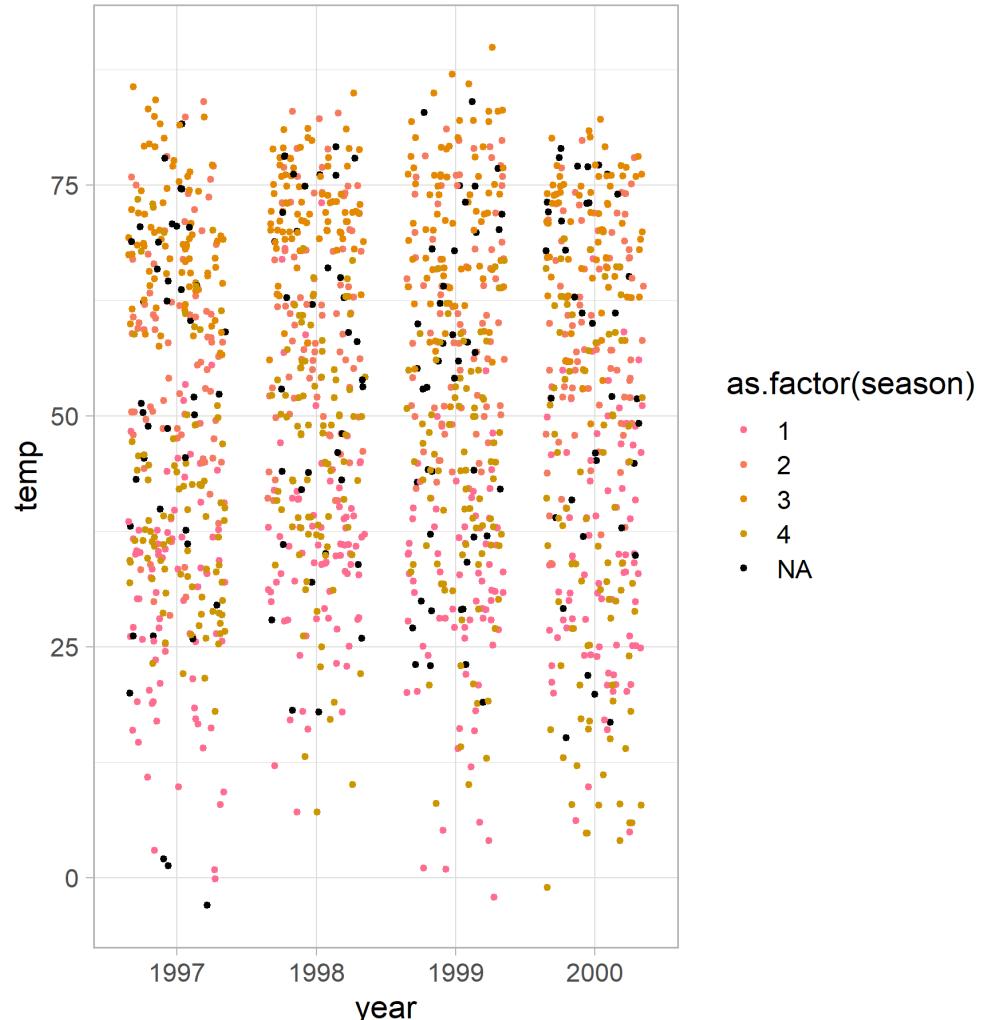
scale_color|fill_discrete()

```
g +
  scale_color_discrete(
    h = c(0, 260), ## hue: [0,360]
    c = 500, ## chroma: [0,x]
    l = 90, ## luminance: [0,100]
    labels = c("Dec-Feb", "Mar-May",
              "June-Aug", "Sep-Nov")
  )
```



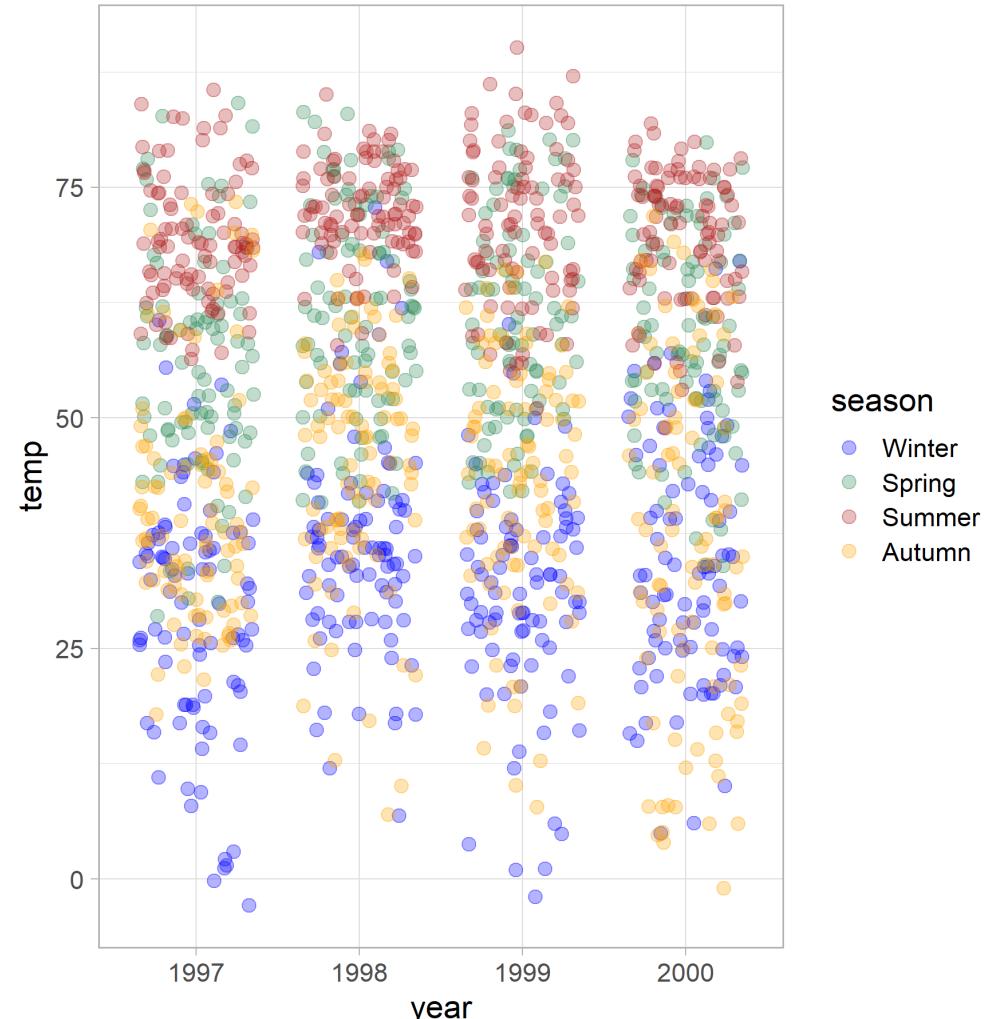
scale_color|fill_discrete()

```
chic %>%
  group_by(date) %>%
  mutate(season = if_else(
    runif(1, min = 0, max = 1) > .9,
    NA_integer_,
    as.integer(season)
  )) %>%
  ggplot(
    aes(
      year, temp,
      color = as.factor(season)
    )) +
  geom_jitter(
    size = 1.3,
    width = .35
  ) +
  scale_color_discrete(
    h = c(0, 60), ## hue: [0,360]
    na.value = "black"
  )
```



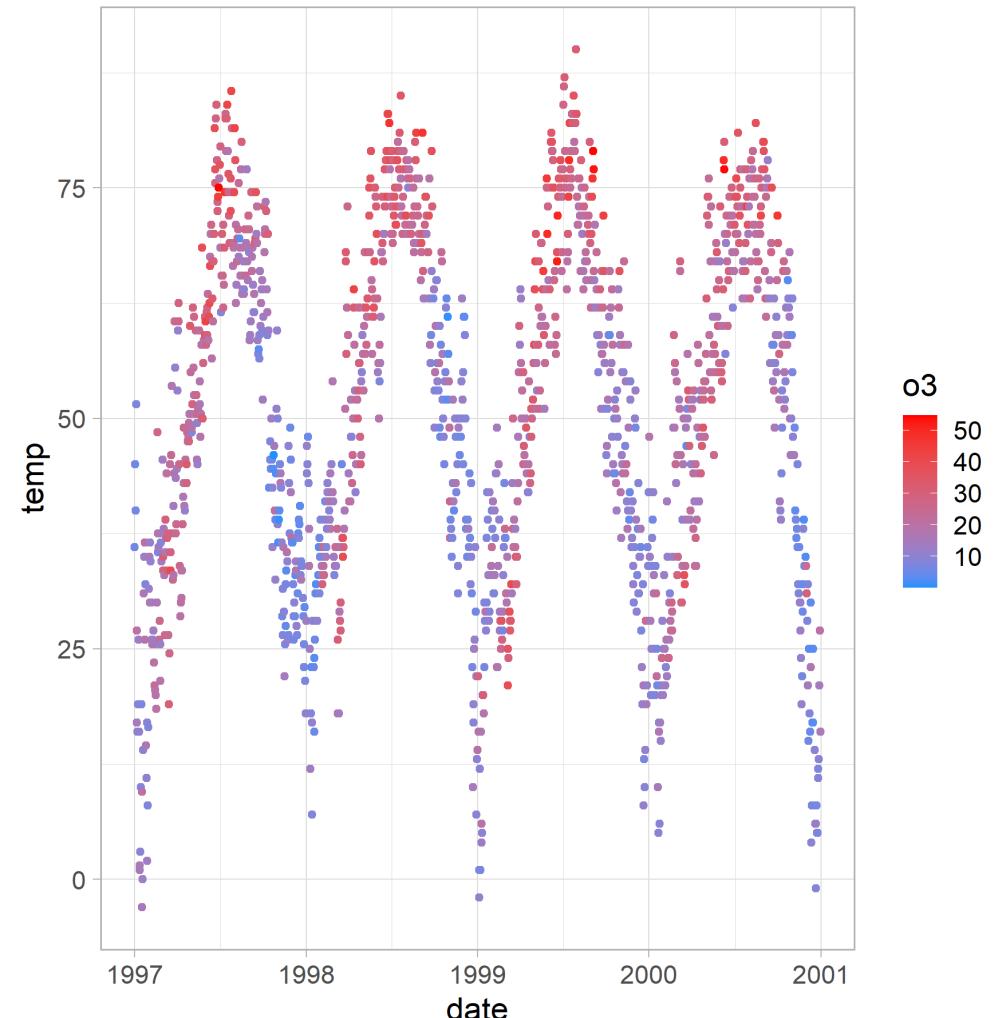
scale_color|fill_manual()

```
g +  
  scale_color_manual(  
    values = c("blue", "seagreen",  
              "firebrick", "orange")  
)
```



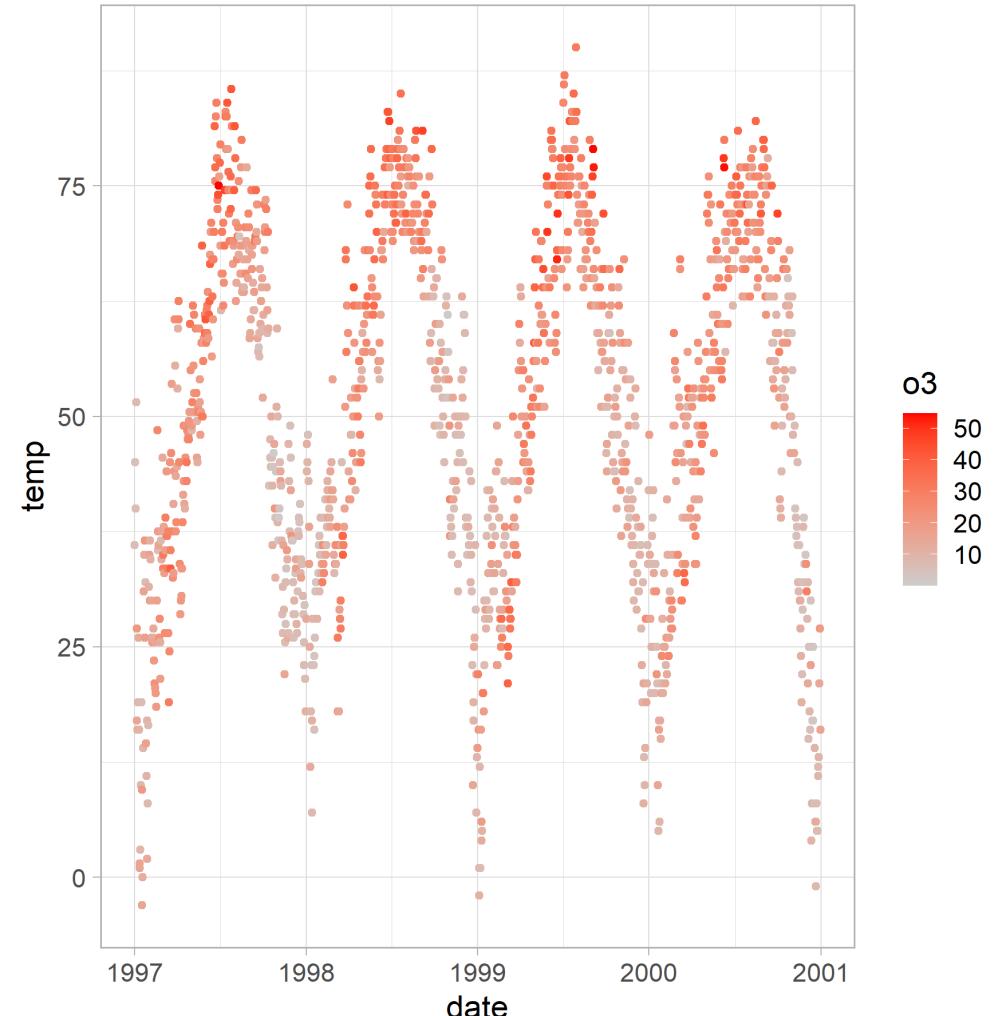
scale_color|fill_gradient()

```
ggplot(chic, aes(date, temp)) +  
  geom_point(aes(color = o3)) +  
  scale_color_gradient(  
    low = "dodgerblue",  
    high = "red"  
)
```



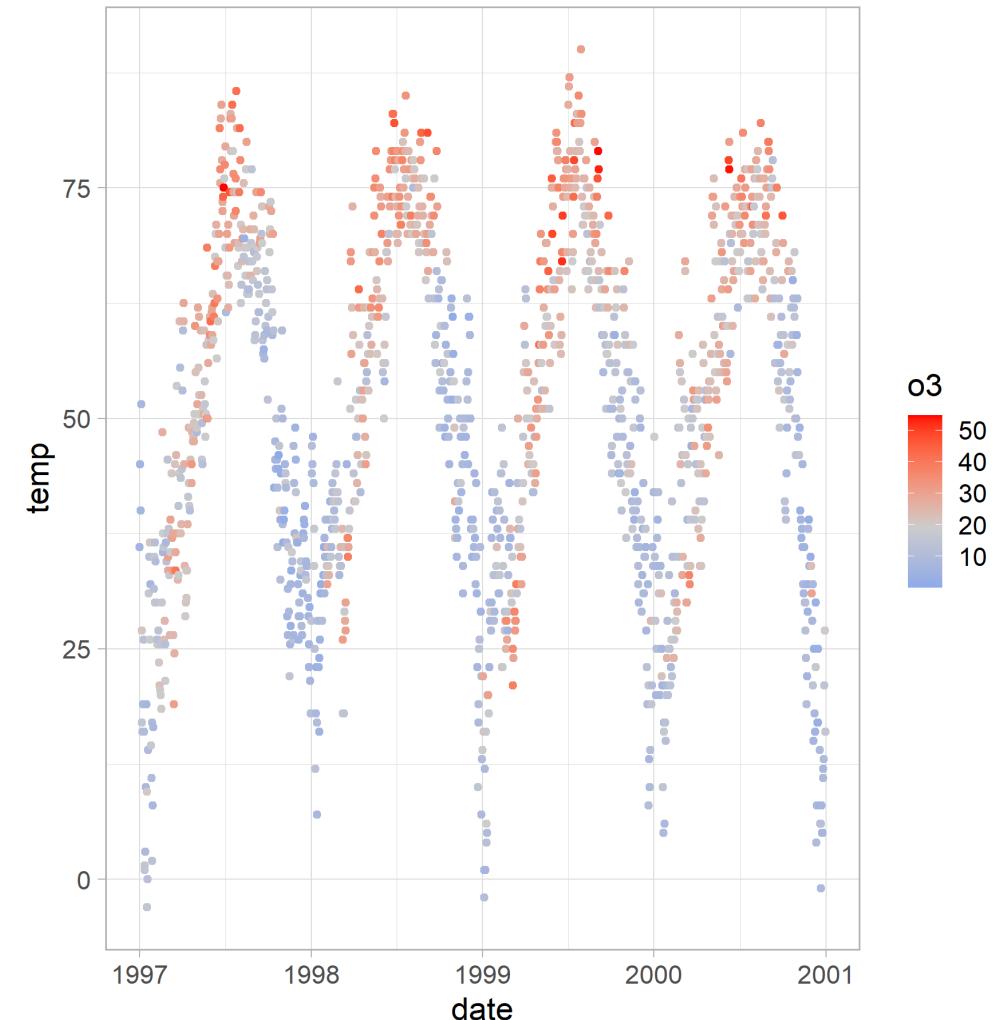
scale_color|fill_gradient2()

```
ggplot(chic, aes(date, temp)) +  
  geom_point(aes(color = o3)) +  
  scale_color_gradient2(  
    low = "dodgerblue",  
    mid = "grey80",  
    high = "red"  
)
```



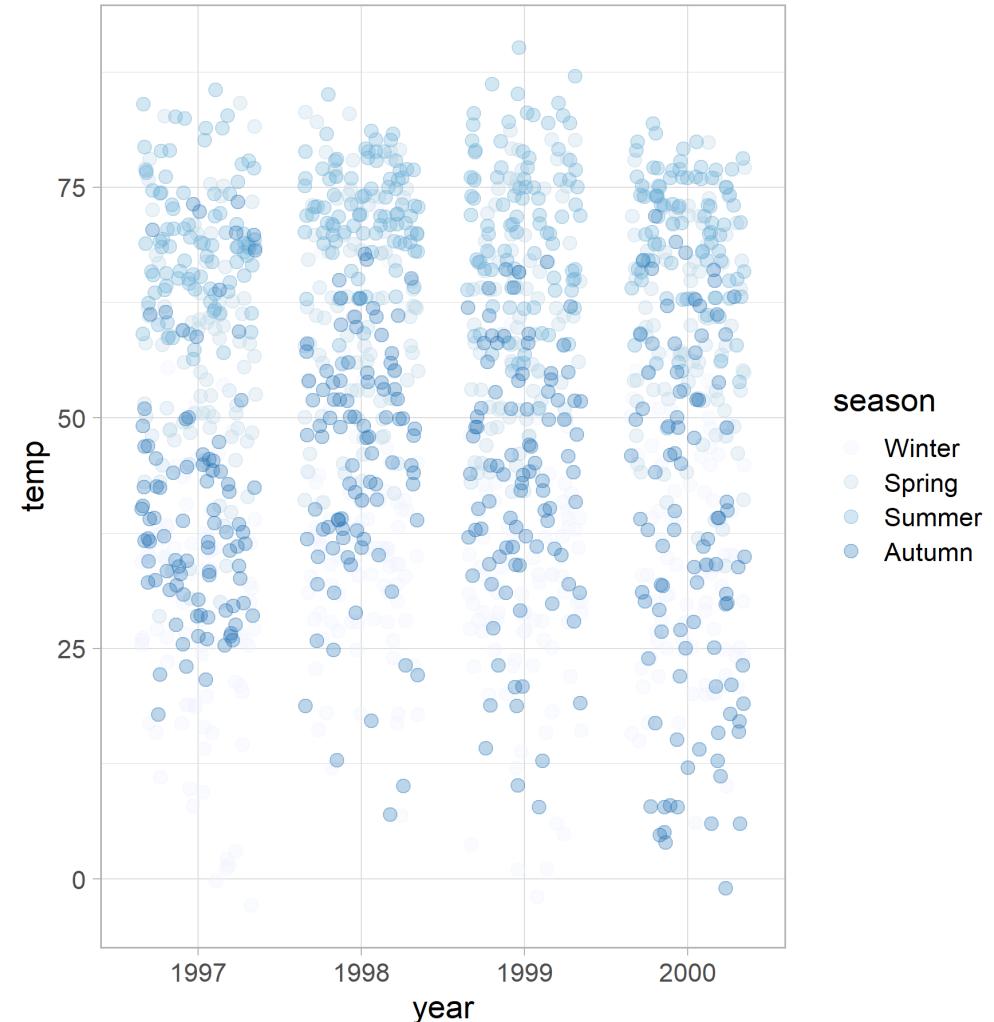
scale_color|fill_gradient2()

```
ggplot(chic, aes(date, temp)) +  
  geom_point(aes(color = o3)) +  
  scale_color_gradient2(  
    low = "dodgerblue",  
    mid = "grey80",  
    high = "red",  
    midpoint = mean(chic$o3)  
)
```



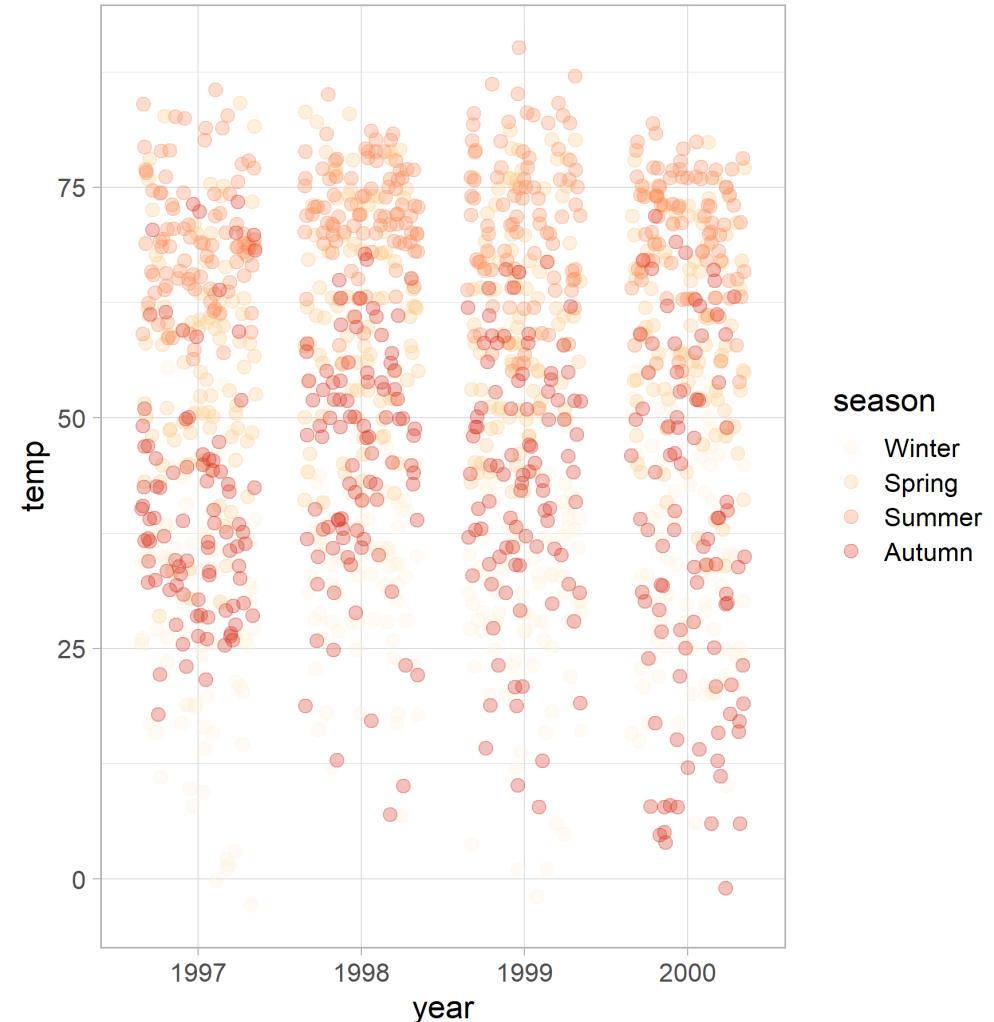
scale_color|fill_brewer()

```
g +  
  scale_color_brewer()
```



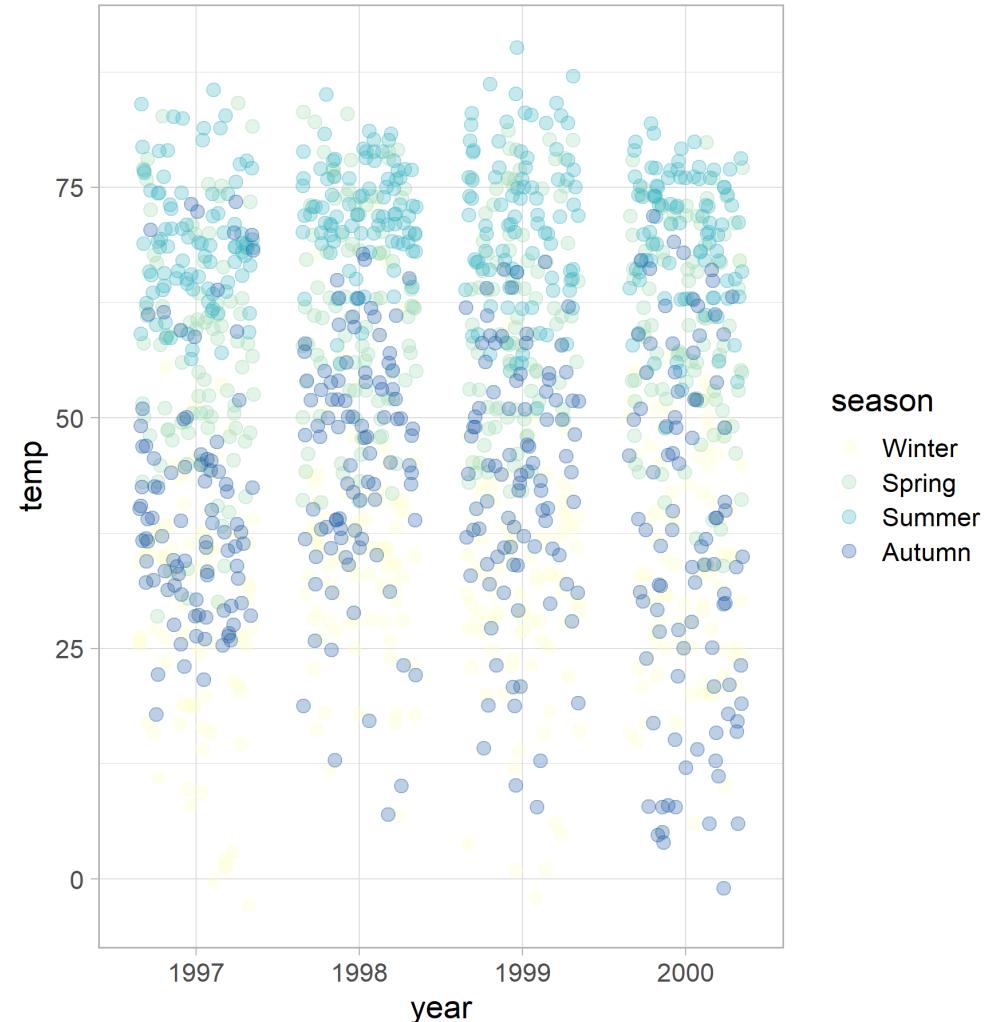
scale_color|fill_brewer()

```
g +  
  scale_color_brewer(  
    palette = 8  
)
```



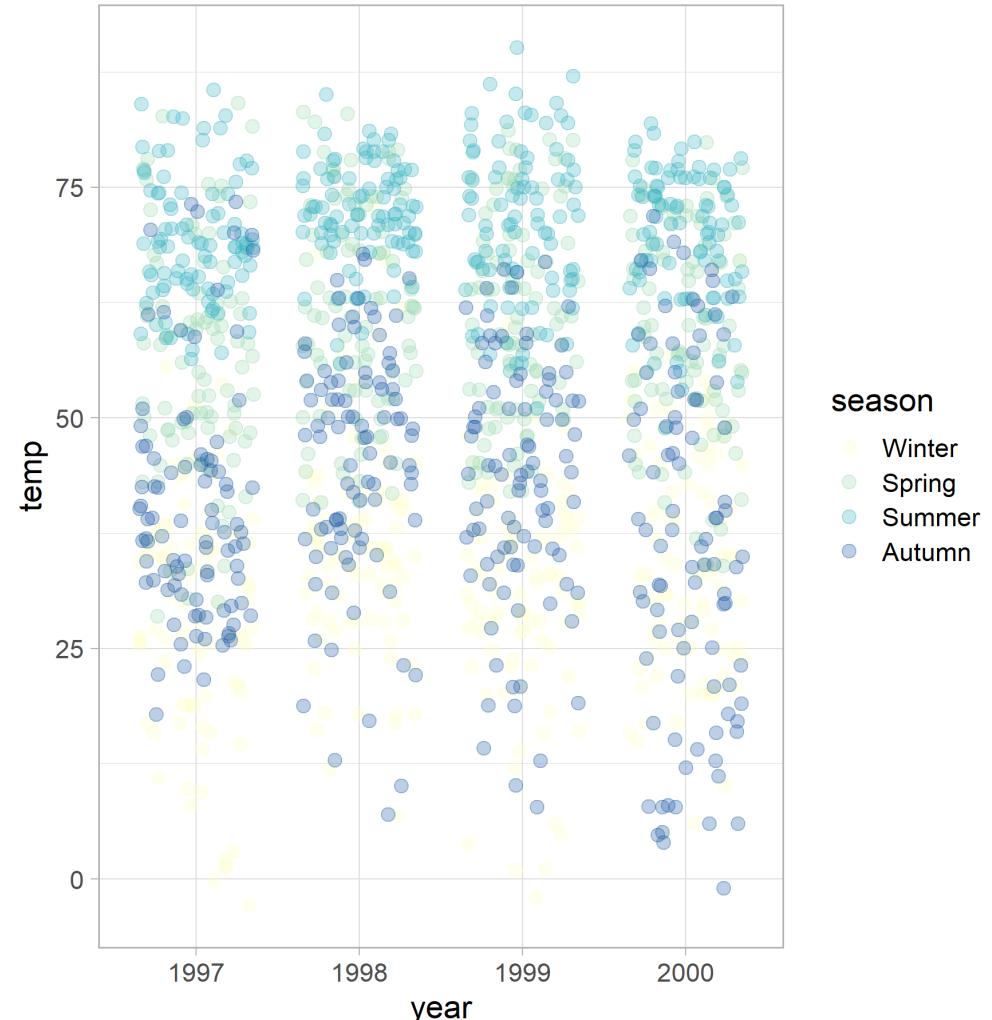
scale_color|fill_brewer()

```
g +
  scale_color_brewer(
    palette = "YlGnBu"
  )
```



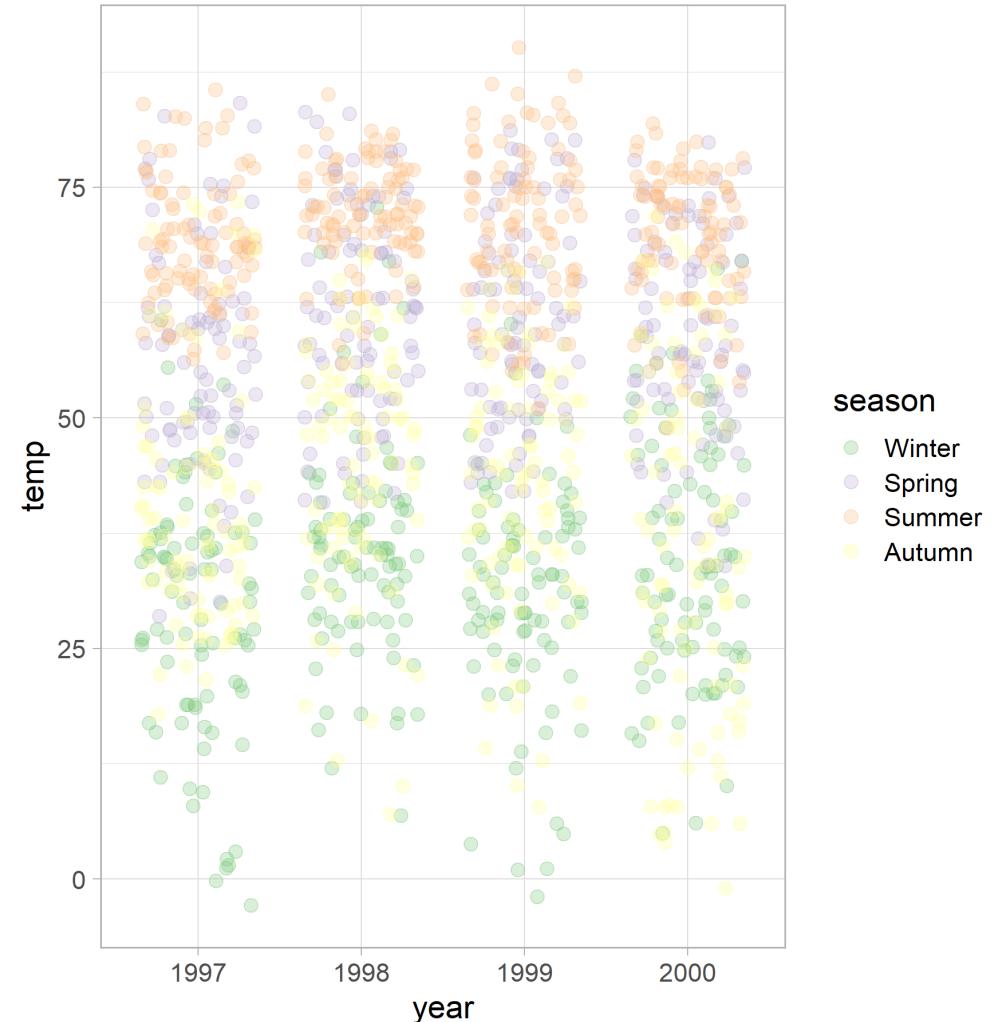
scale_color|fill_brewer()

```
g +
  scale_color_brewer(
    type = "seq", ## the default
    palette = "YlGnBu"
  )
```



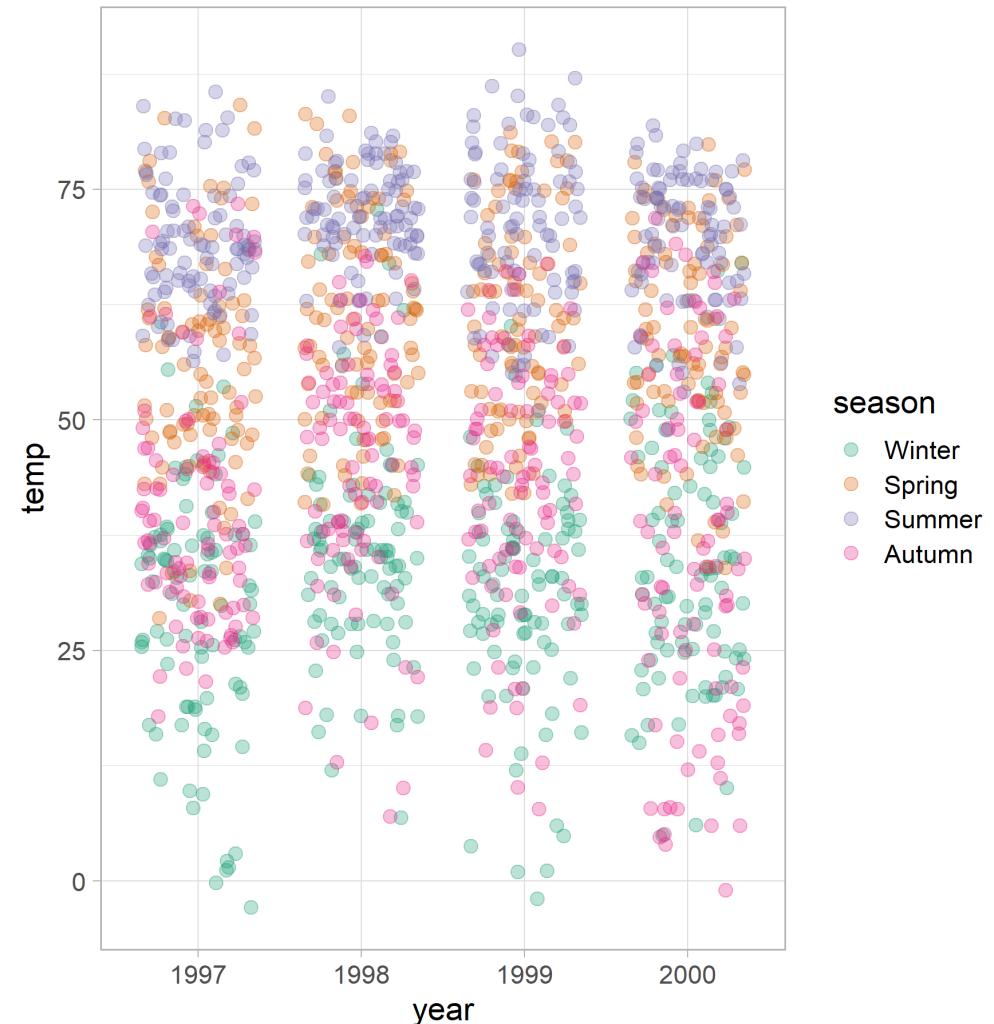
scale_color|fill_brewer()

```
g +  
  scale_color_brewer(  
    type = "qual"  
)
```



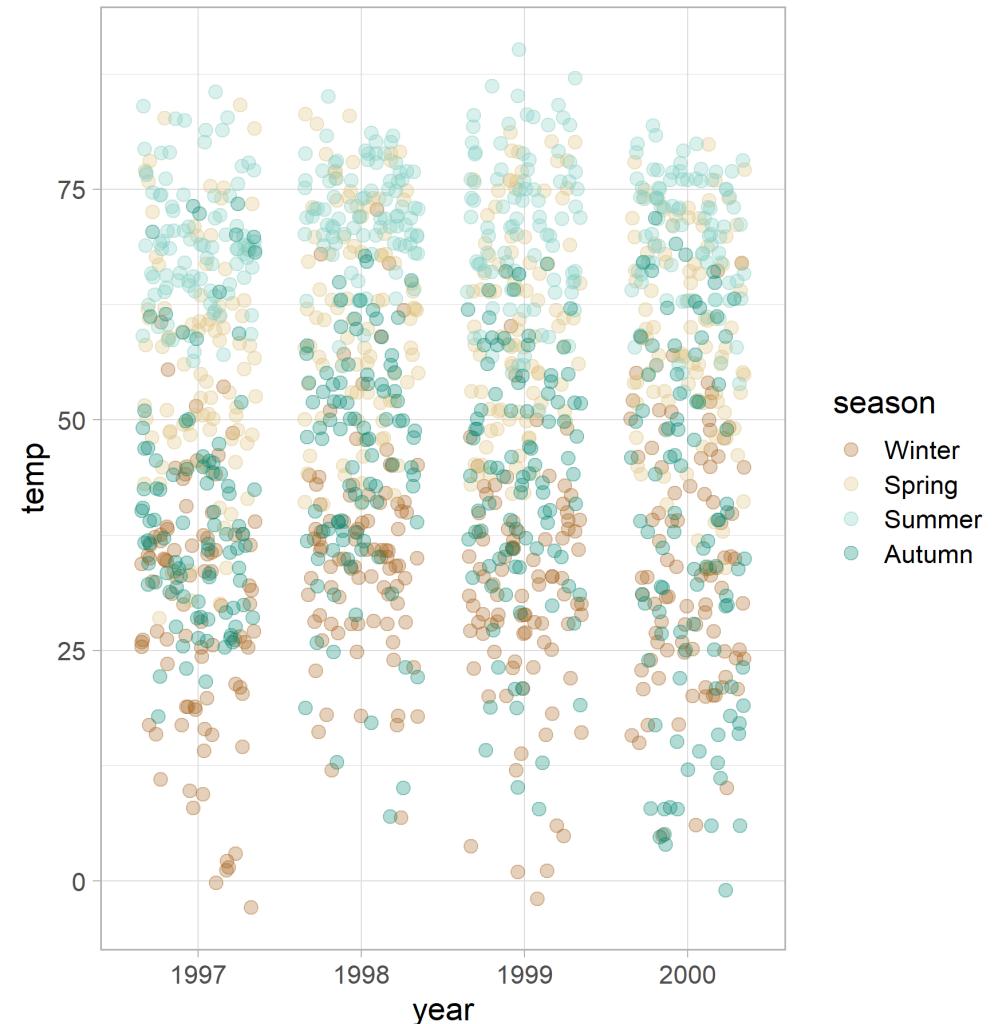
scale_color|fill_brewer()

```
g +
  scale_color_brewer(
    type = "qual",
    palette = "Dark2"
  )
```



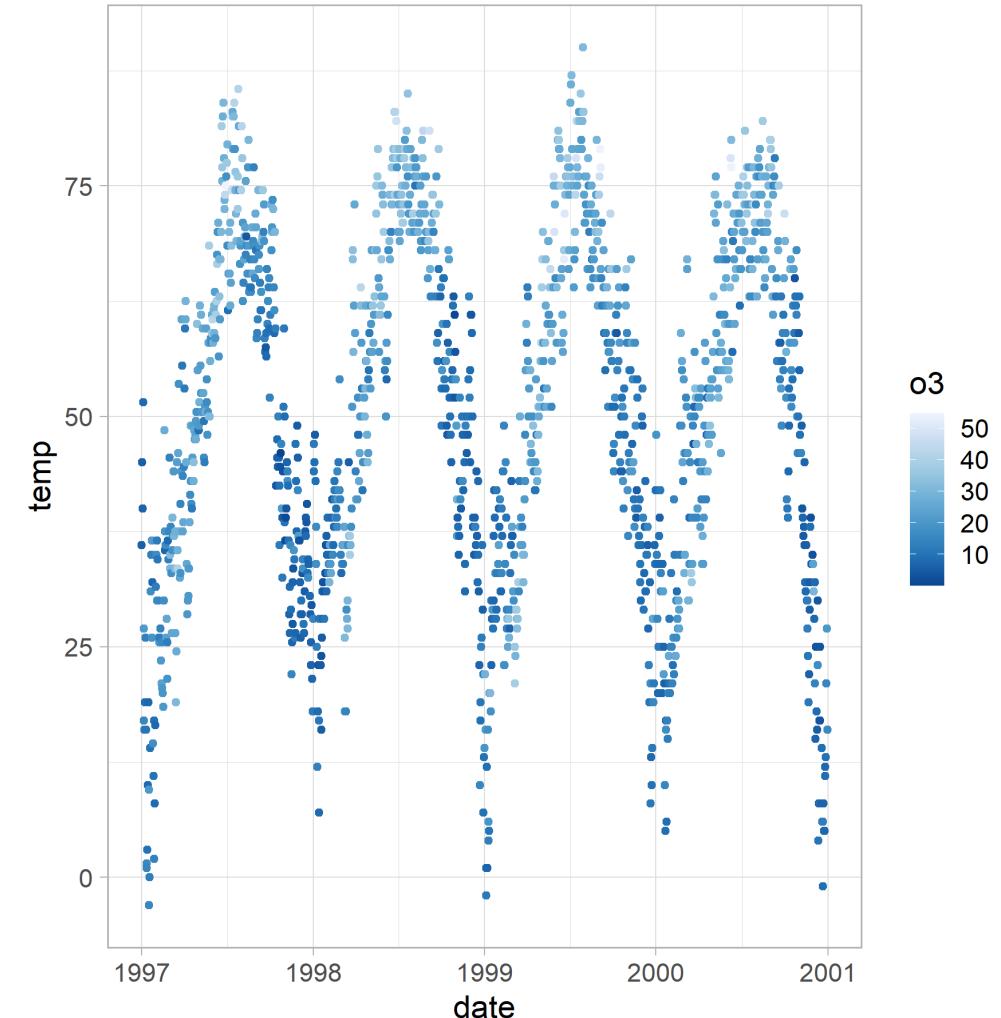
scale_color|fill_brewer()

```
g +  
  scale_color_brewer(  
    type = "div"  
)
```



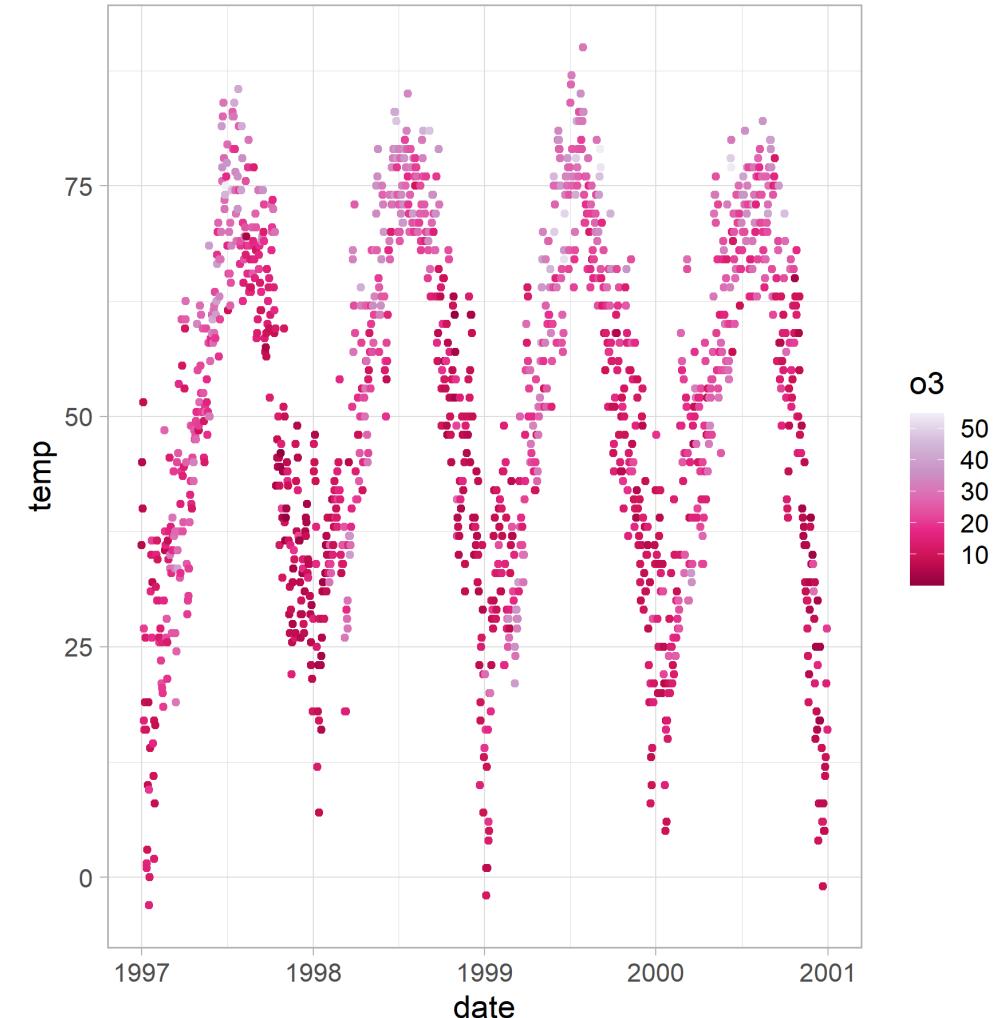
scale_color|fill_distiller()

```
ggplot(chic, aes(date, temp)) +  
  geom_point(aes(color = o3)) +  
  scale_color_distiller()
```



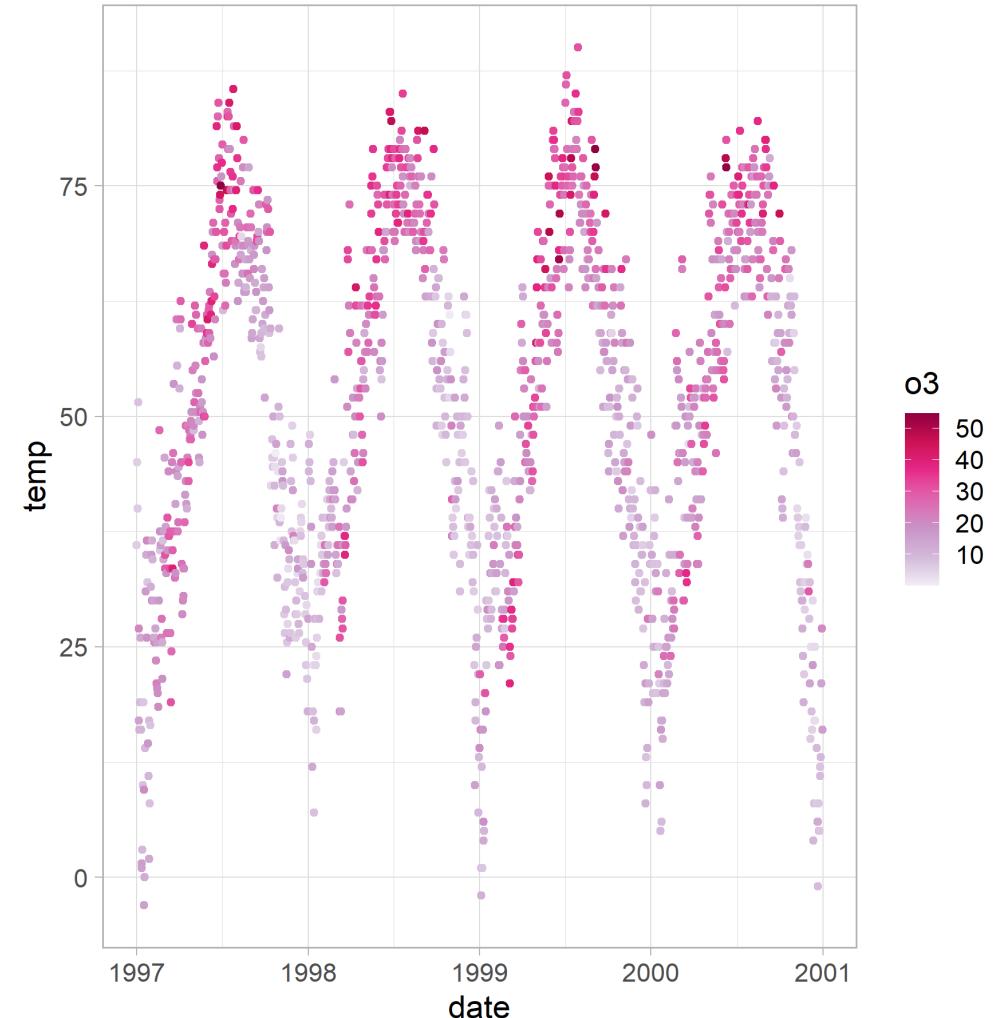
scale_color|fill_distiller()

```
ggplot(chic, aes(date, temp)) +  
  geom_point(aes(color = o3)) +  
  scale_color_distiller(  
    palette = "PuRd"  
)
```



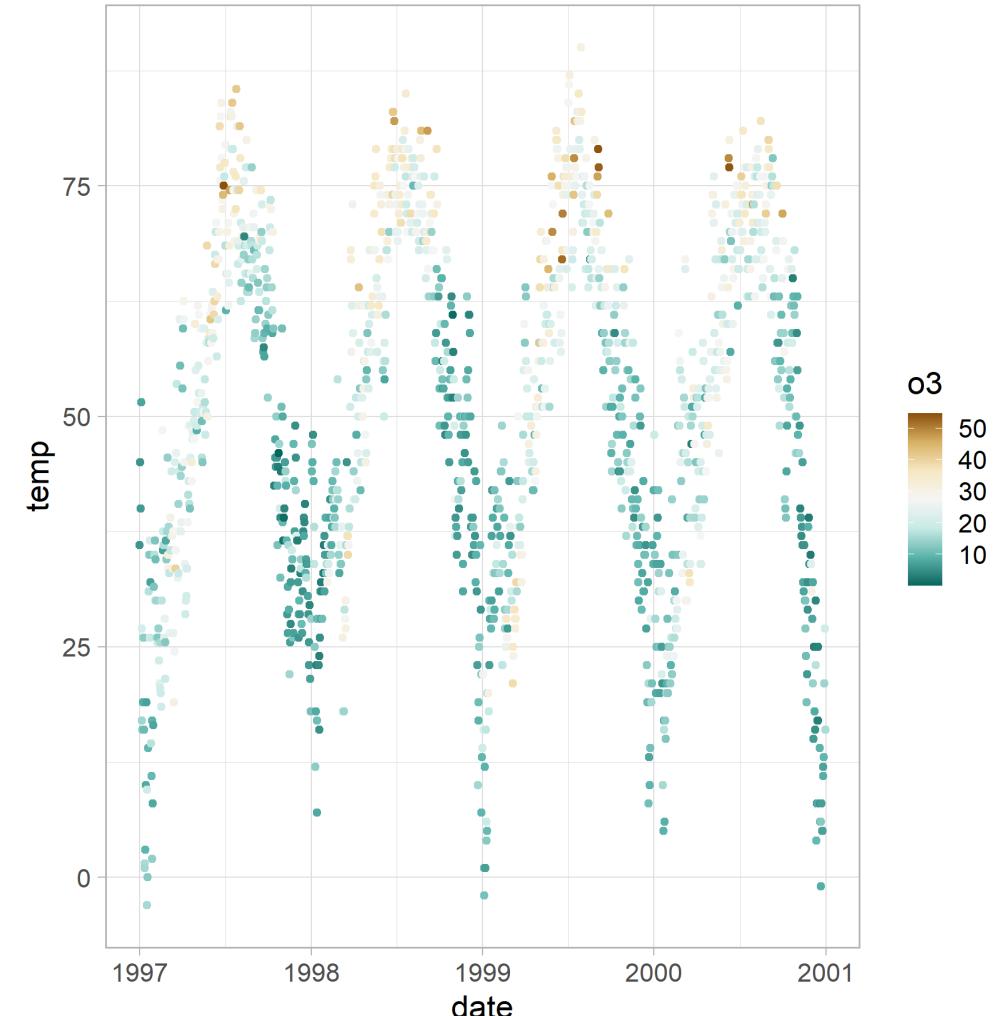
scale_color|fill_distiller()

```
ggplot(chic, aes(date, temp)) +  
  geom_point(aes(color = o3)) +  
  scale_color_distiller(  
    palette = "PuRd",  
    direction = 1  
    ## note that -1 is the default  
)
```



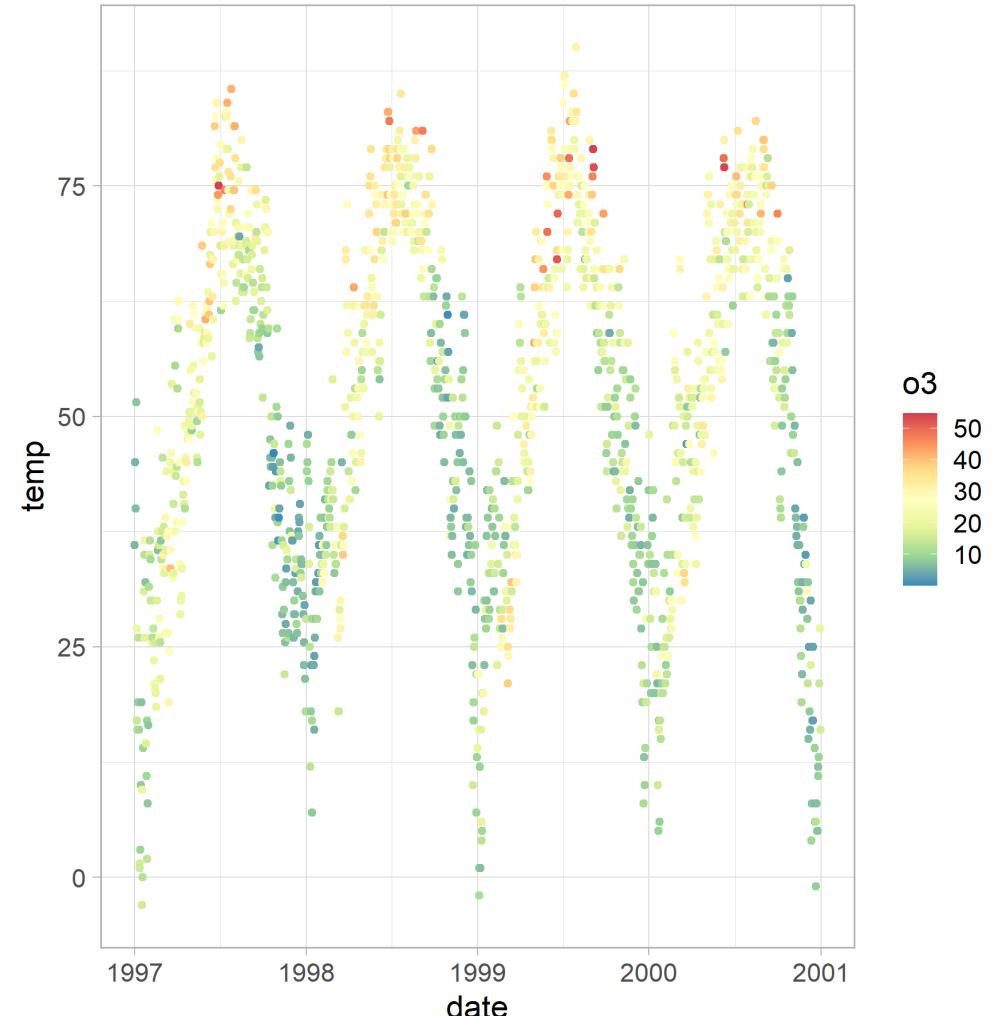
scale_color|fill_distiller()

```
ggplot(chic, aes(date, temp)) +  
  geom_point(aes(color = o3)) +  
  scale_color_distiller(  
    type = "div"  
)
```



scale_color|fill_distiller()

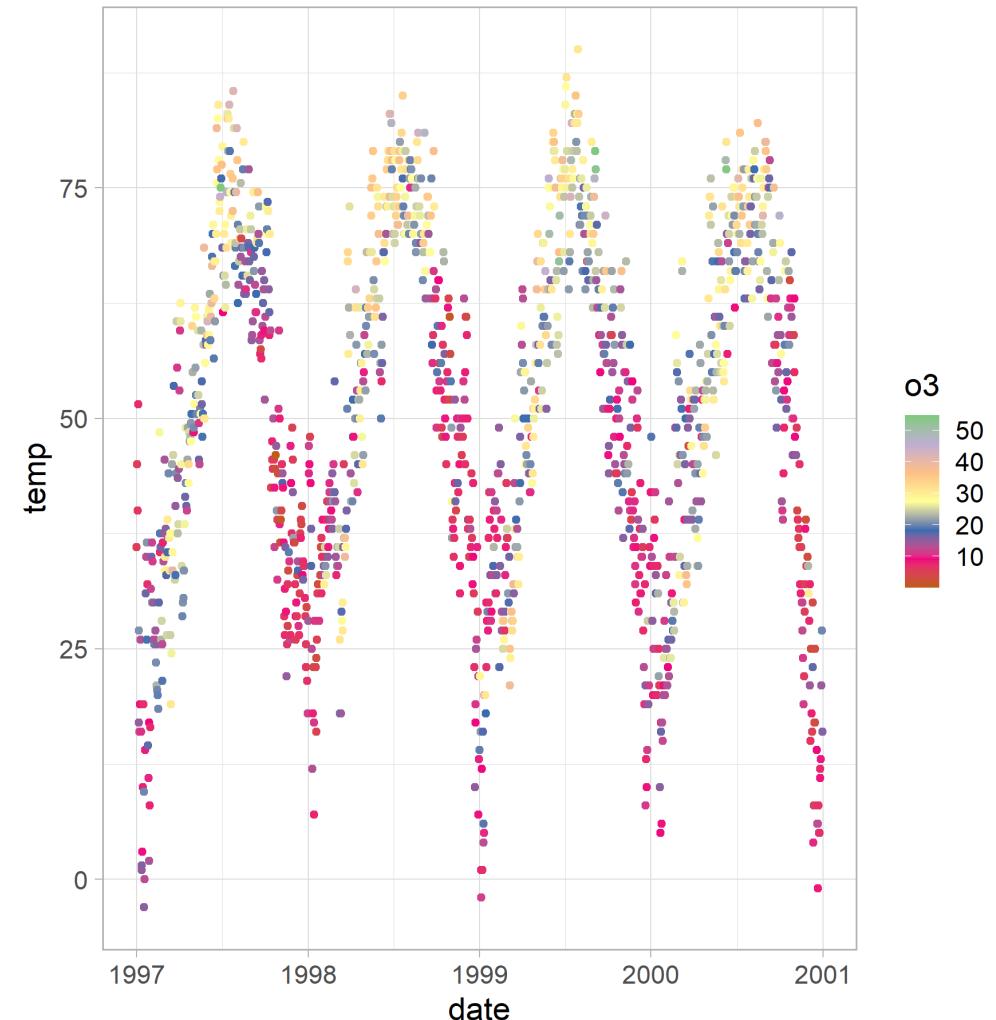
```
ggplot(chic, aes(date, temp)) +  
  geom_point(aes(color = o3)) +  
  scale_color_distiller(  
    type = "div",  
    palette = "Spectral"  
)
```



scale_color|fill_distiller()

```
ggplot(chic, aes(date, temp)) +  
  geom_point(aes(color = o3)) +  
  scale_color_distiller(  
    type = "qual"  
)
```

Warning message: Using a discrete colour palette
in a continuous scale. Consider using type = "seq"
or type = "div" instead.



`scale_color|fill_*`()

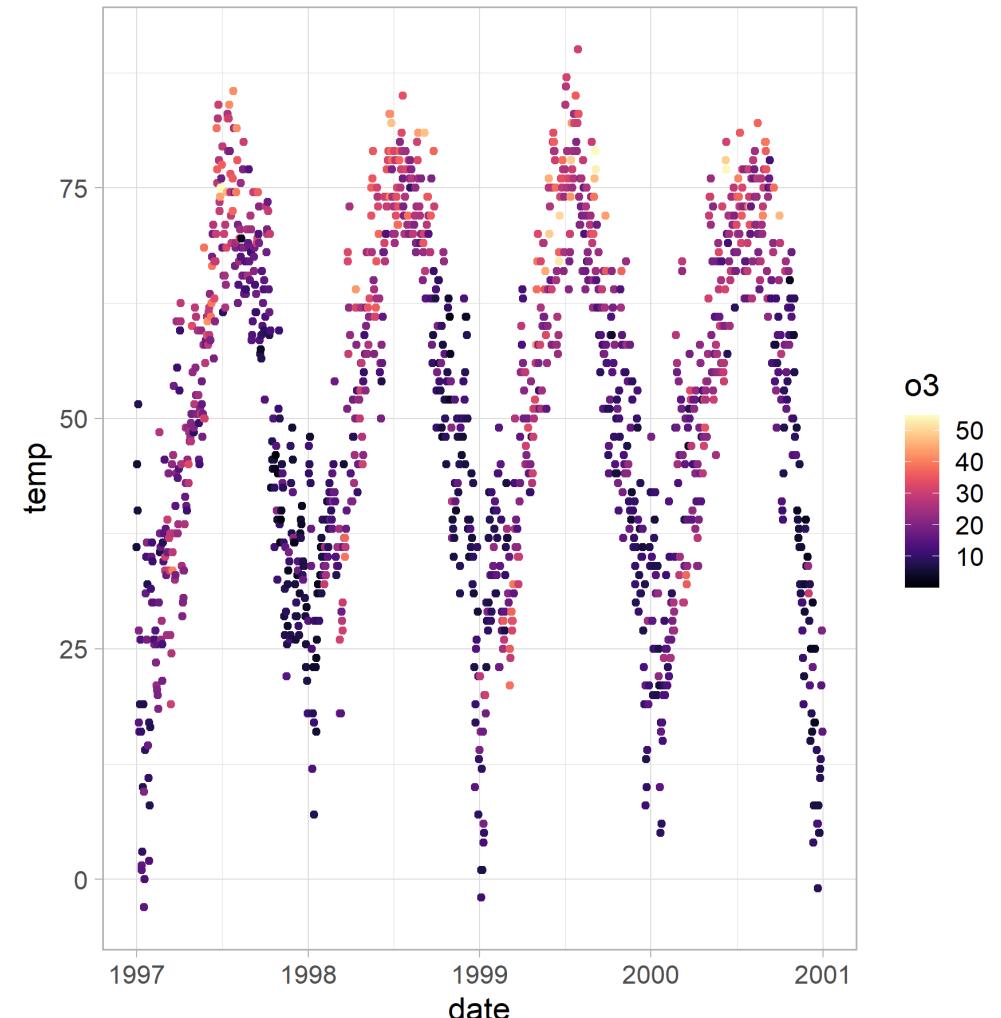
Several packages offer predefined palettes, e.g.:

- `{viridis}` for perceptually uniform palettes
- `{scico}` for more perceptually uniform palettes
- `{rcartocolor}` for map color palettes
- `{ggsci}` for scientific journal and sci-fi themed color
- `{ggthemes}` for colors of popular software & publishers
- `{LaCroixColoR}` for vibrant summery colors

Check the [collection by Emil Hvitfeldt](#) for an extensive list of color palettes available in R!

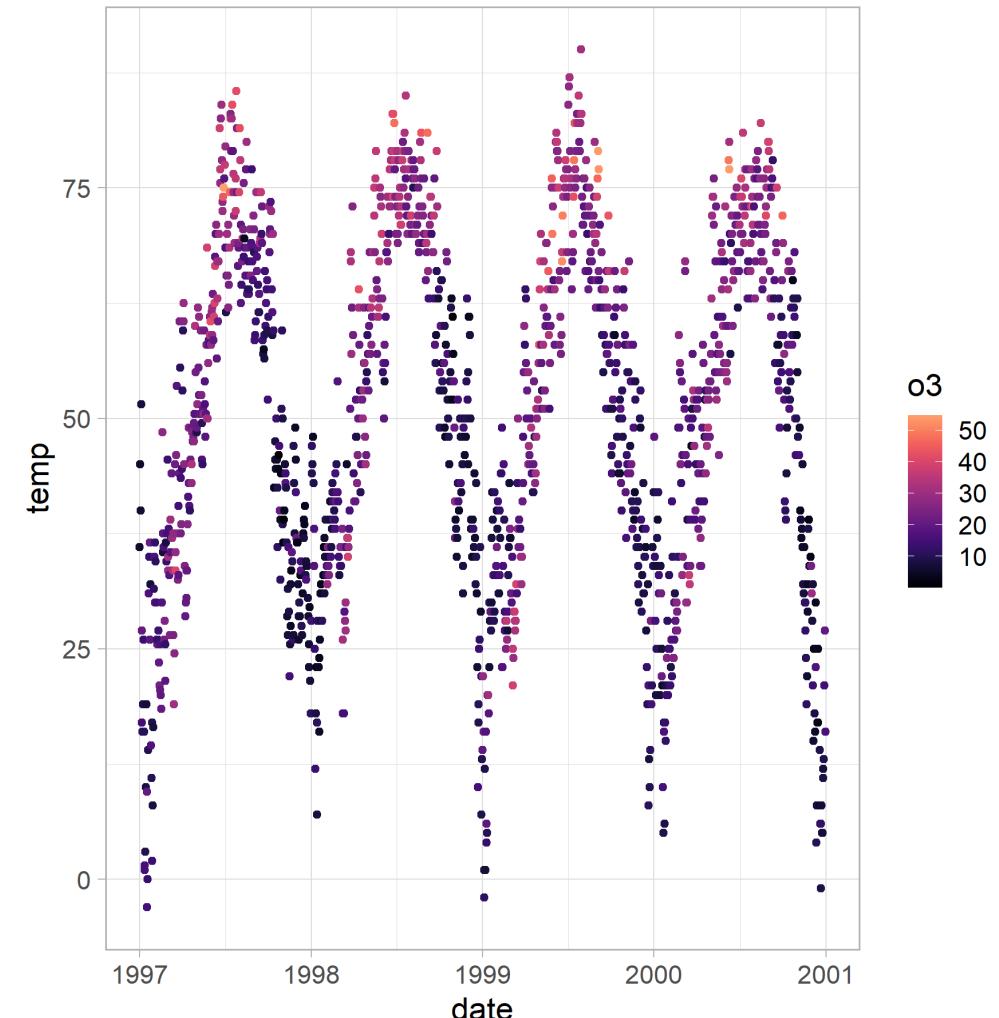
`viridis::scale_color|fill_viridis()`

```
ggplot(chic, aes(date, temp)) +  
  geom_point(aes(color = o3)) +  
  viridis::scale_color_viridis(  
    option = "magma"  
)
```



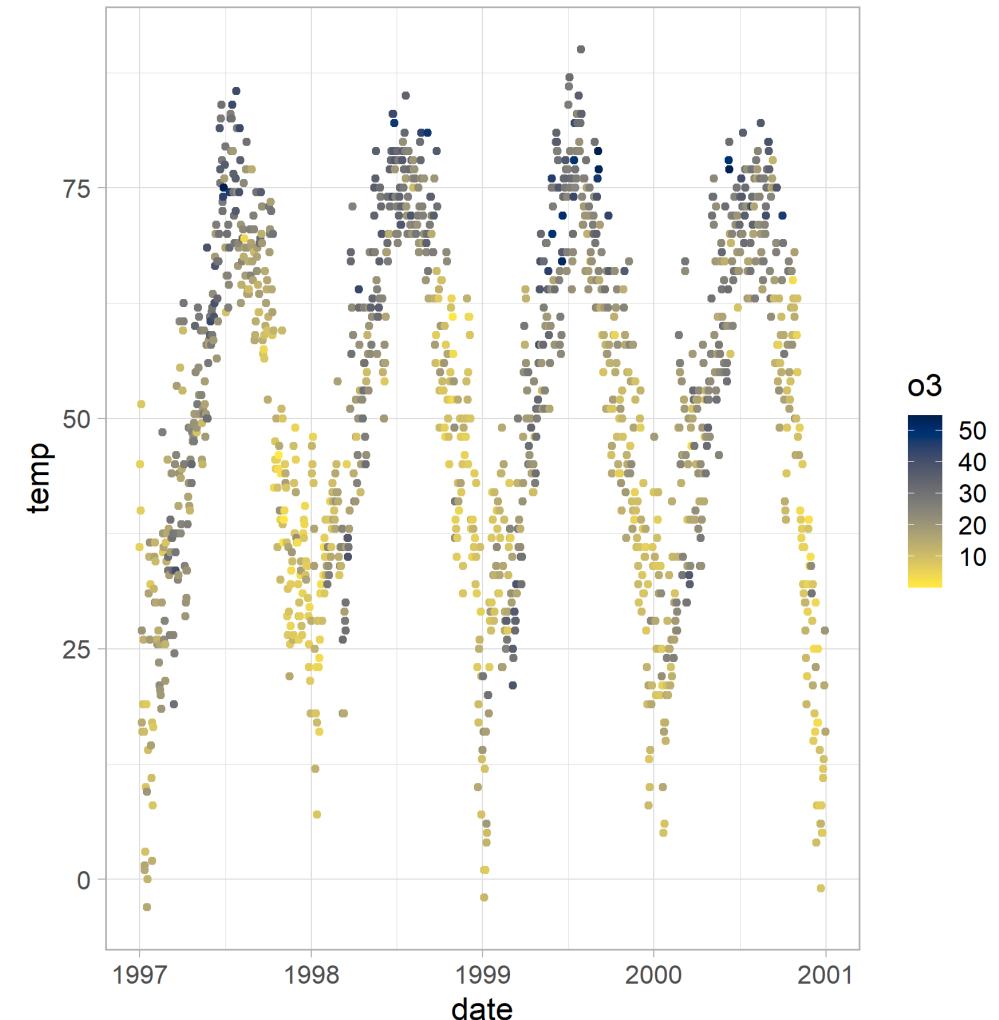
`viridis::scale_color|fill_viridis()`

```
ggplot(chic, aes(date, temp)) +  
  geom_point(aes(color = o3)) +  
  viridis::scale_color_viridis(  
    option = "magma",  
    end = .8  
)
```



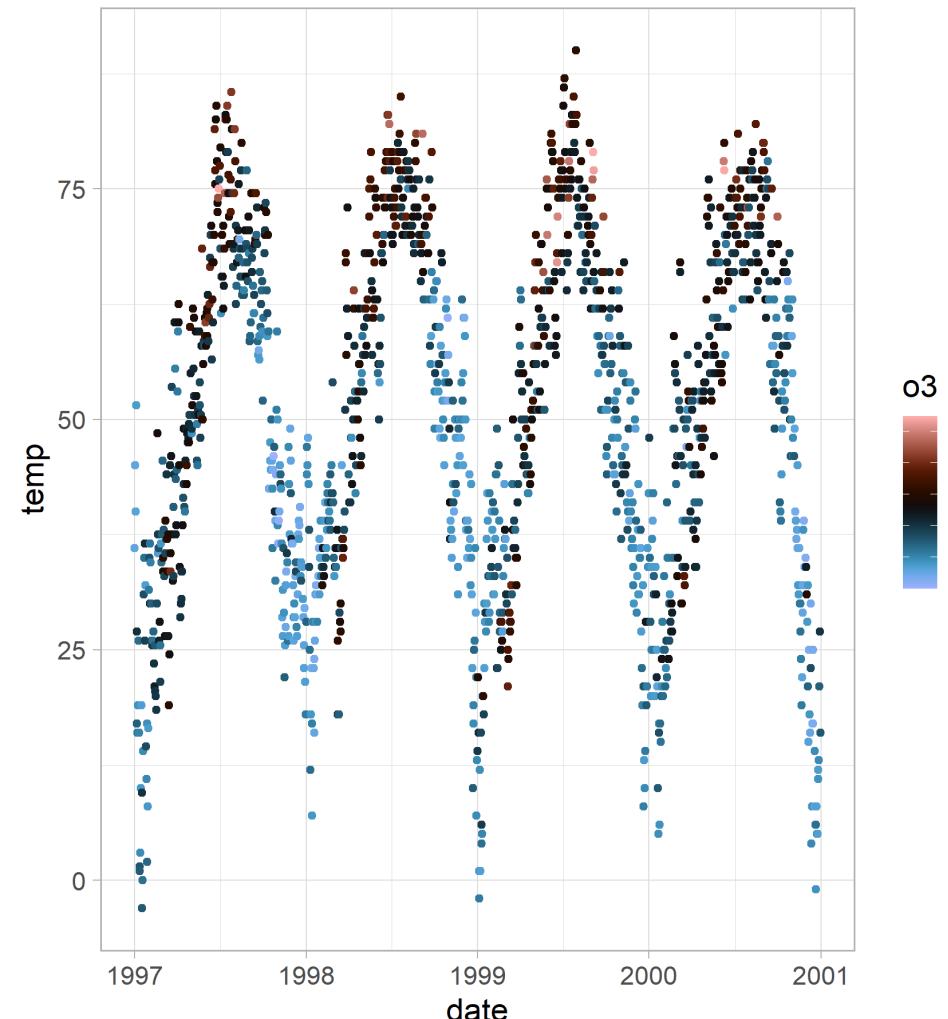
`viridis::scale_color|fill_viridis()`

```
ggplot(chic, aes(date, temp)) +  
  geom_point(aes(color = o3)) +  
  viridis::scale_color_viridis(  
    option = "cividis",  
    direction = -1  
)
```



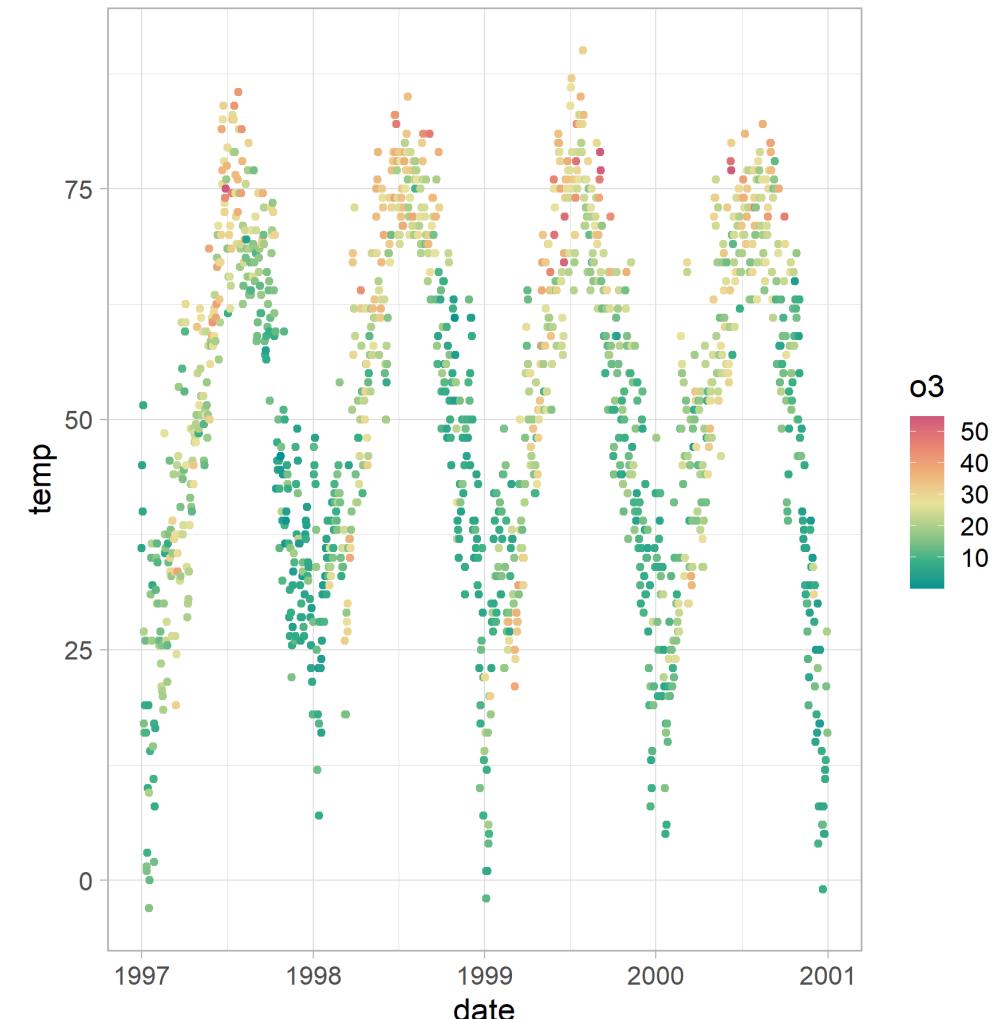
scico::scale_color|fill_scico()

```
ggplot(chic, aes(date, temp)) +  
  geom_point(aes(color = o3)) +  
  scico::scale_color_scico(  
    palette = "berlin"  
)
```



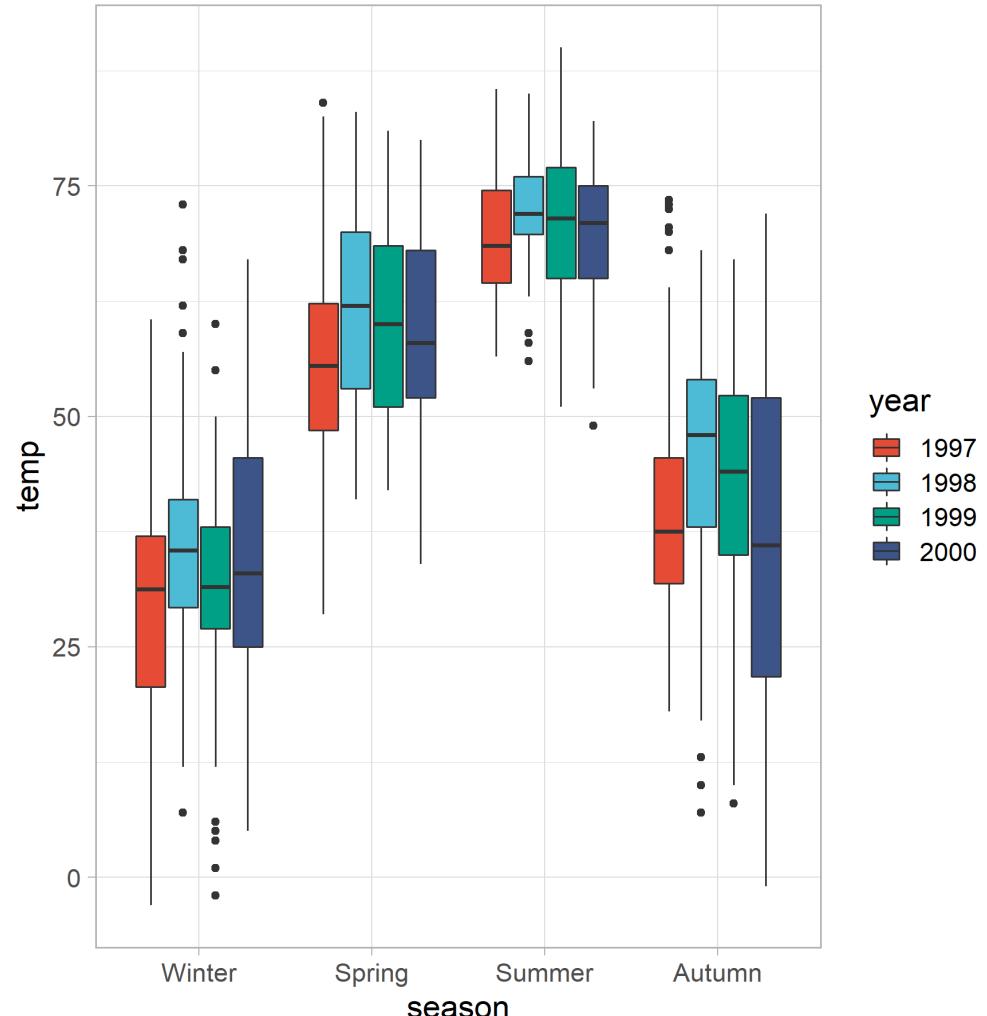
rcartocolor::scale_color|fill_carto_c()

```
ggplot(chic, aes(date, temp)) +  
  geom_point(aes(color = o3)) +  
  rcartocolor::scale_color_carto_c(  
    palette = "Temps"  
)
```



ggsci::scale_color|fill_*

```
ggplot(chic, aes(season, temp)) +  
  geom_boxplot(aes(fill = year)) +  
  ggsci::scale_fill_npg()
```



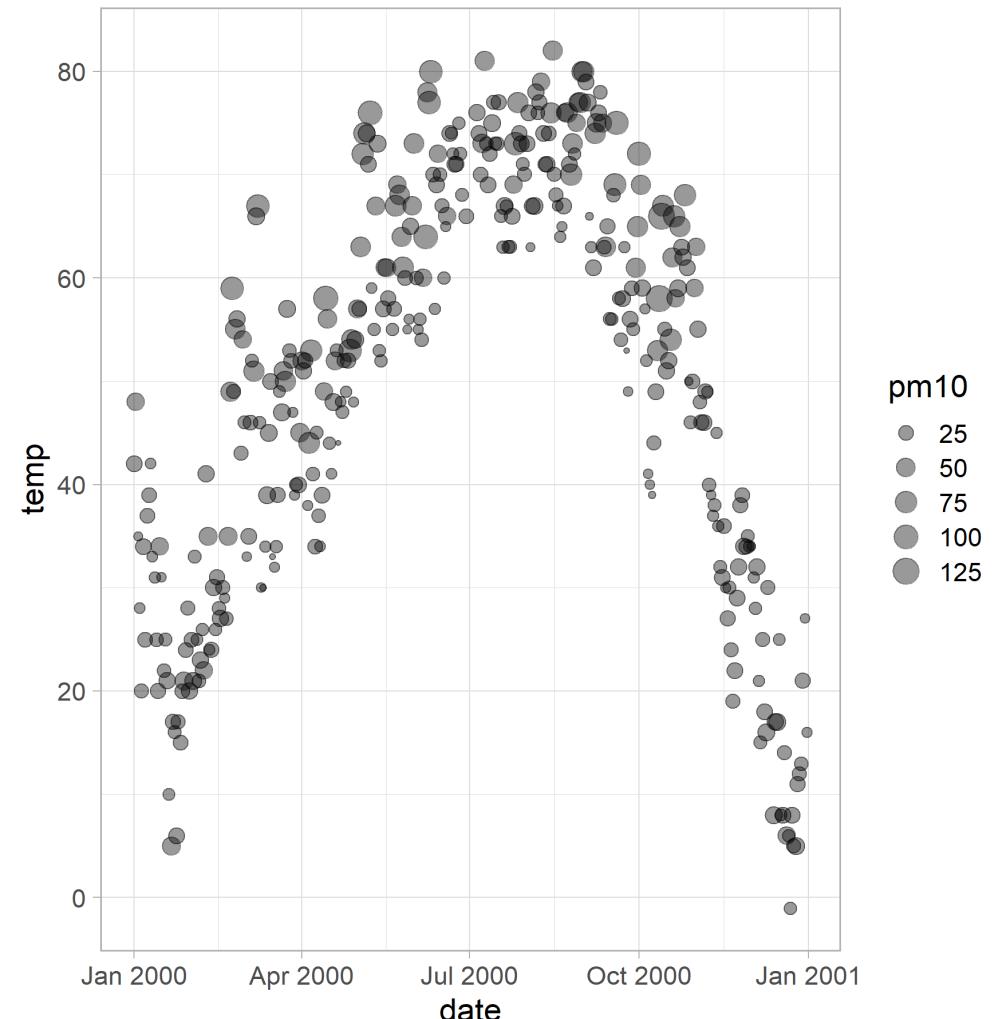
Scales

`scale_size|radius_*`(`)`

scale_size|radius_*

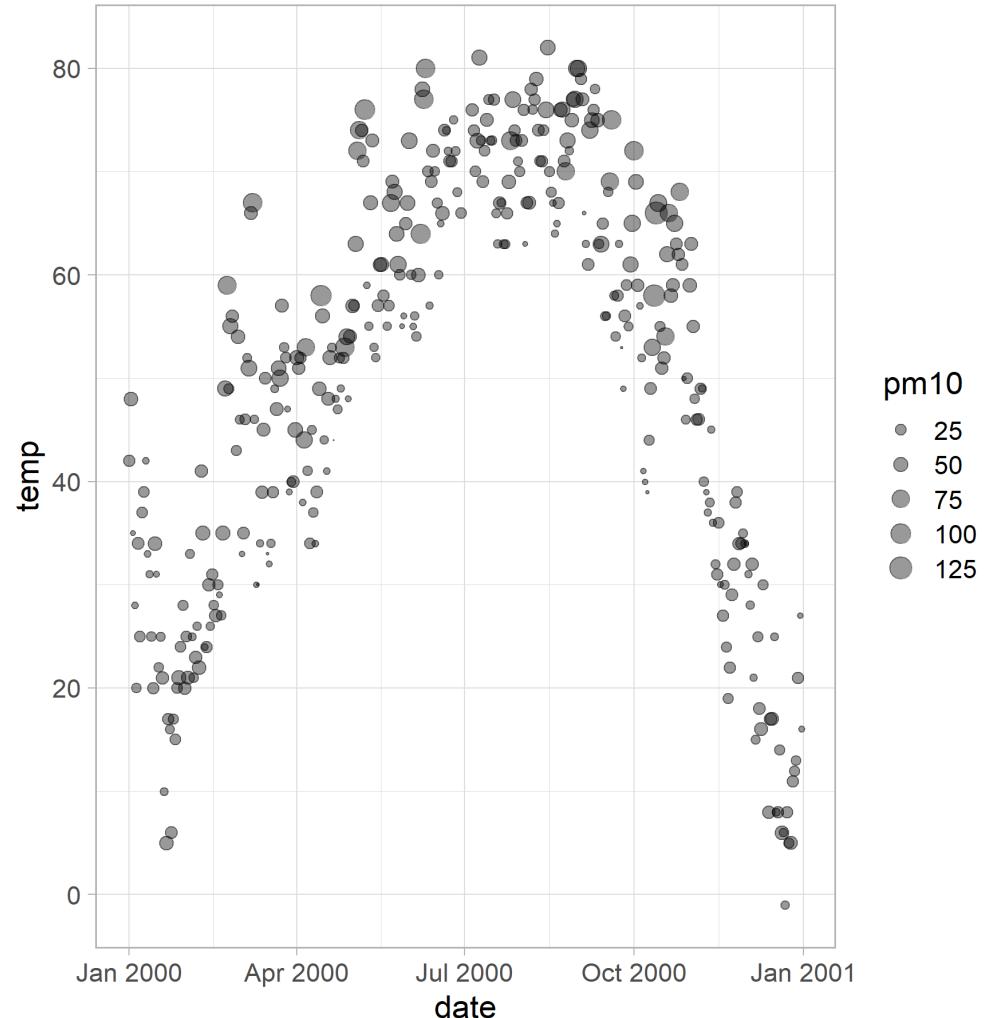
```
chic2k <- filter(chic, year == 2000)

ggplot(chic2k, aes(date, temp)) +
  geom_point(
    aes(size = pm10),
    alpha = .4
  )
```



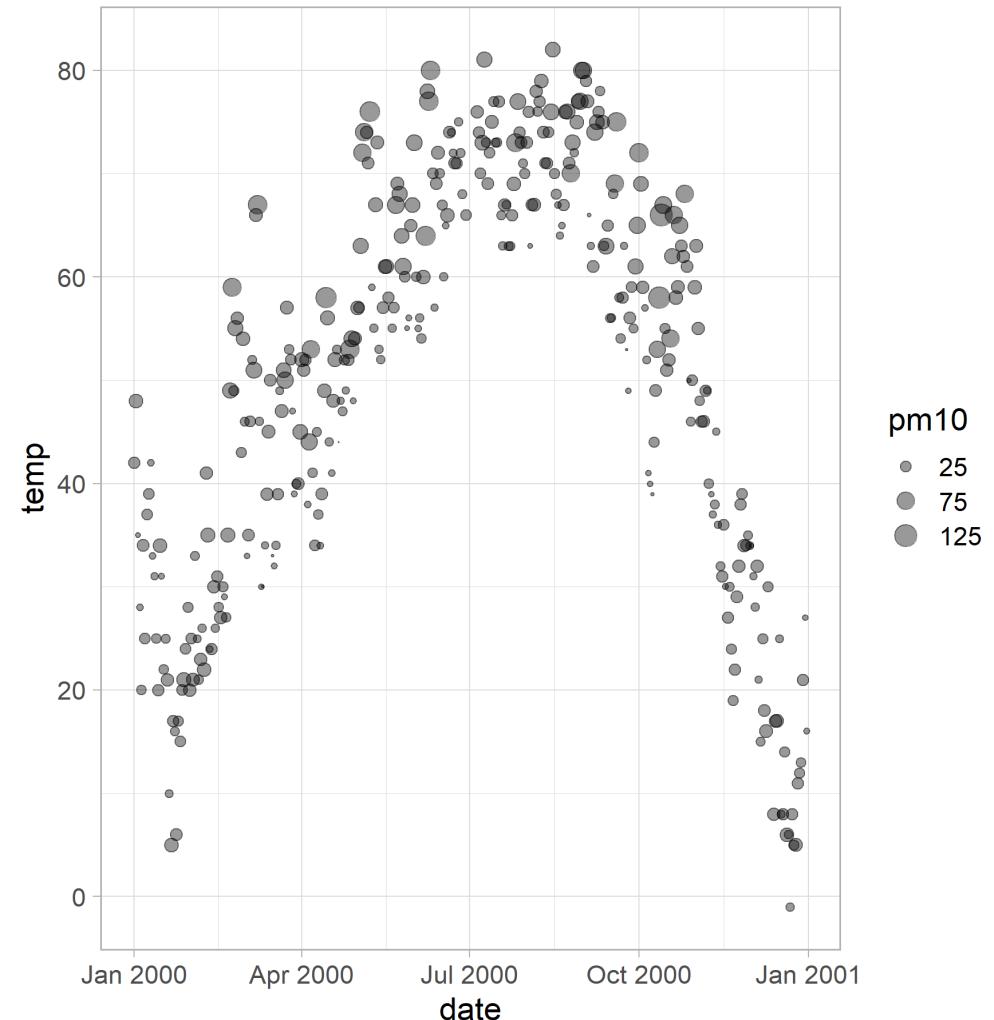
scale_size()

```
ggplot(chic2k, aes(date, temp)) +  
  geom_point(  
    aes(size = pm10),  
    alpha = .4  
) +  
  scale_size(  
    range = c(.1, 5)  
)
```



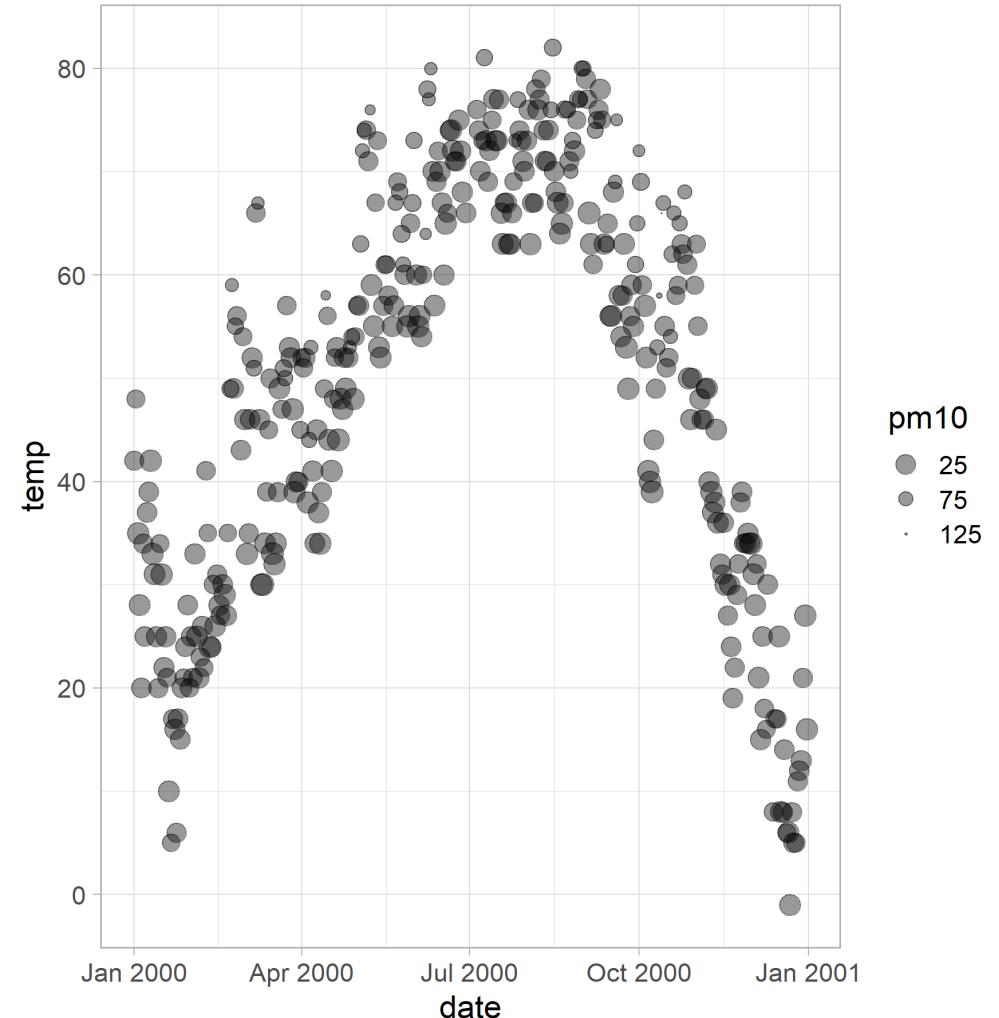
scale_size()

```
ggplot(chic2k, aes(date, temp)) +  
  geom_point(  
    aes(size = pm10),  
    alpha = .4  
) +  
  scale_size(  
    range = c(.1, 5),  
    breaks = c(25, 75, 125)  
)
```



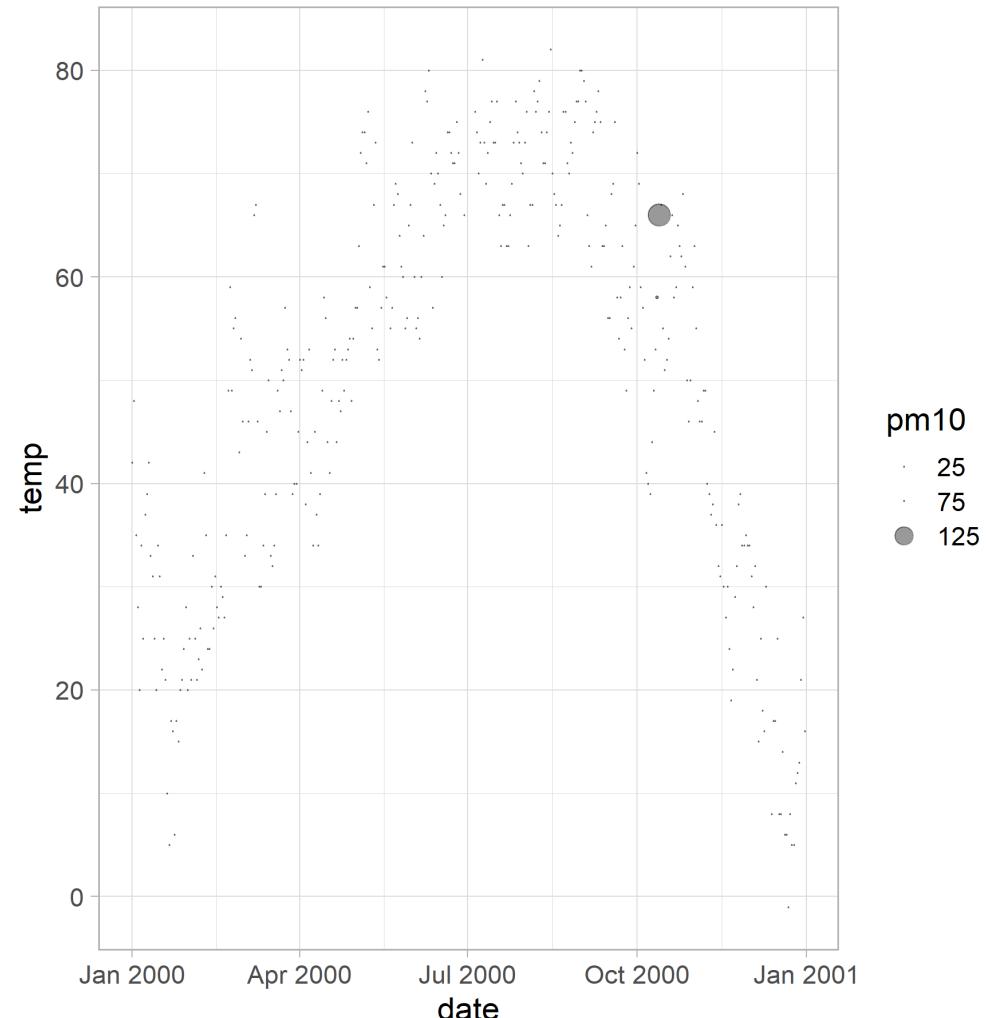
scale_size()

```
ggplot(chic2k, aes(date, temp)) +  
  geom_point(  
    aes(size = pm10),  
    alpha = .4  
  ) +  
  scale_size(  
    range = c(.1, 5),  
    breaks = c(25, 75, 125),  
    trans = "reverse"  
  )
```



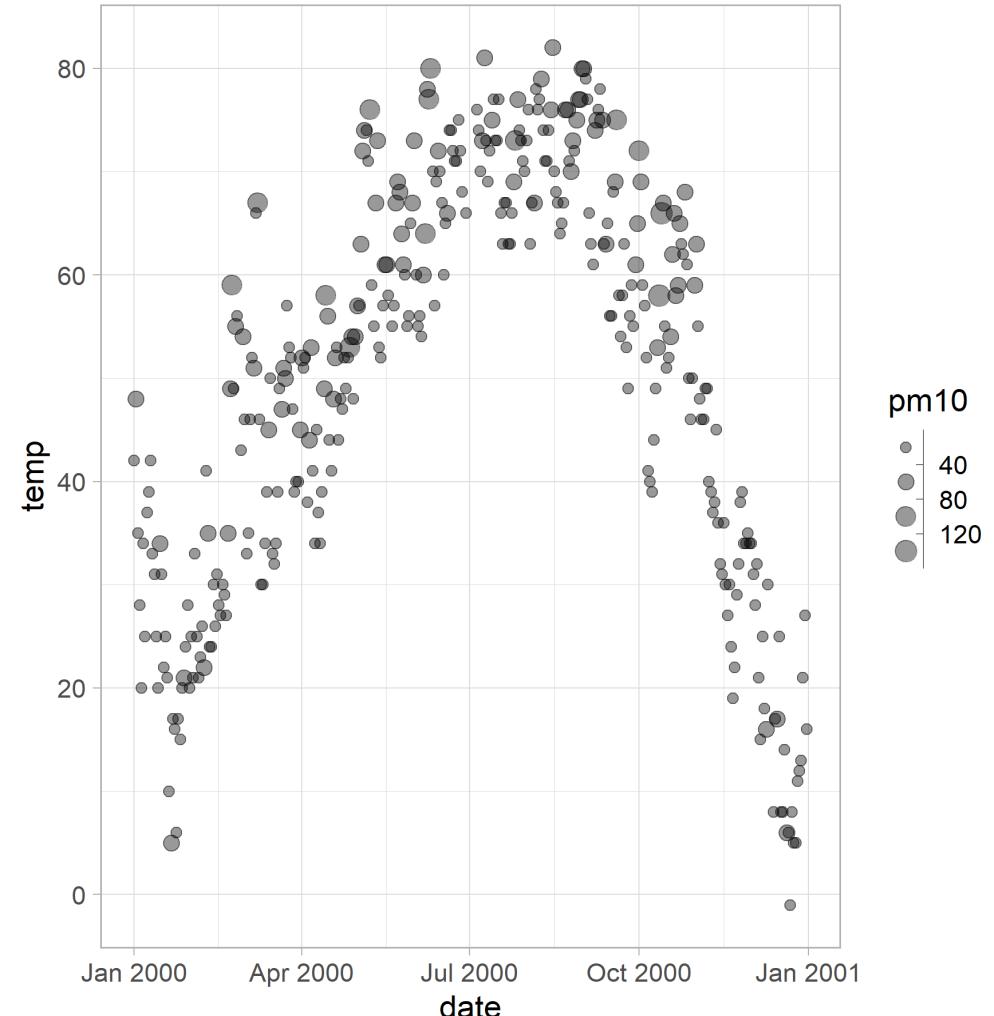
scale_size()

```
ggplot(chic2k, aes(date, temp)) +  
  geom_point(  
    aes(size = pm10),  
    alpha = .4  
  ) +  
  scale_size(  
    range = c(.1, 5),  
    breaks = c(25, 75, 125),  
    trans = "exp"  
  )
```



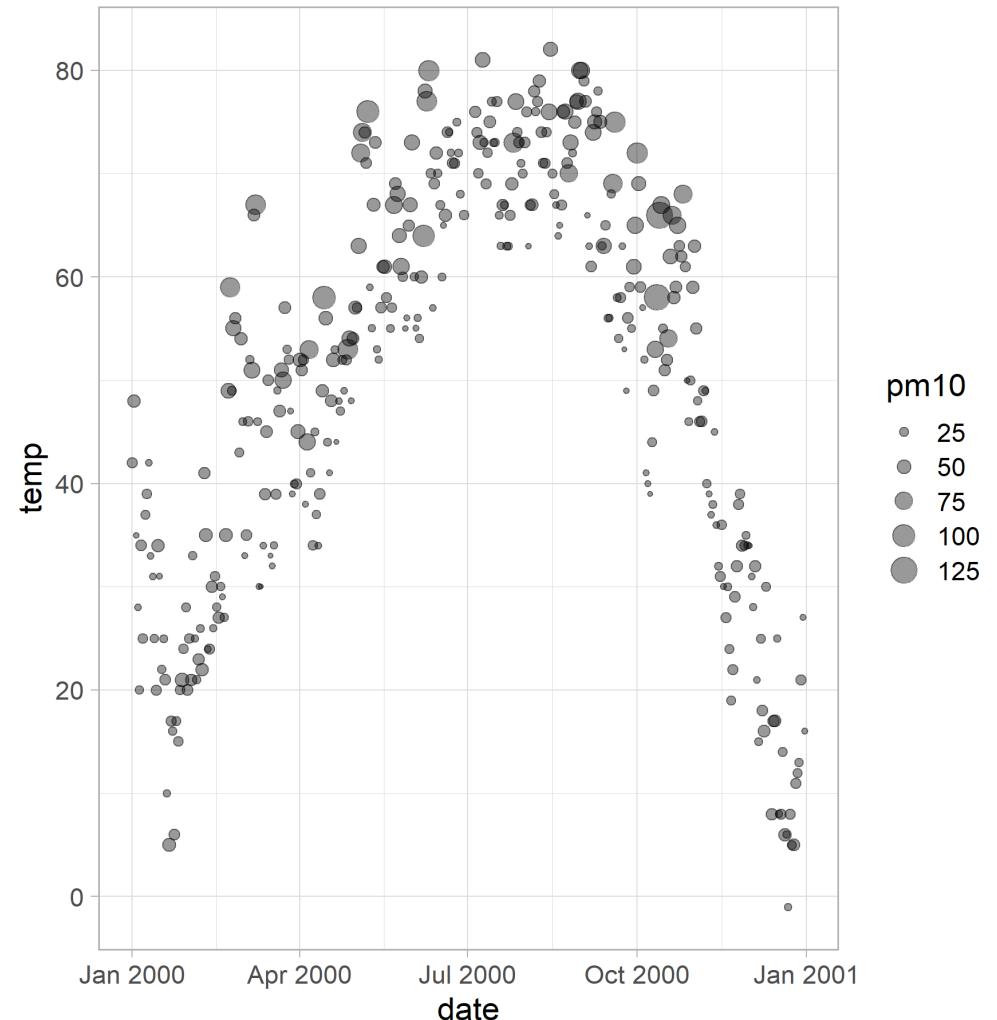
scale_size()

```
ggplot(chic2k, aes(date, temp)) +  
  geom_point(  
    aes(size = pm10),  
    alpha = .4  
  ) +  
  scale_size_binned(  
    n.breaks = 3,  
    range = c(.5, 5)  
  )
```

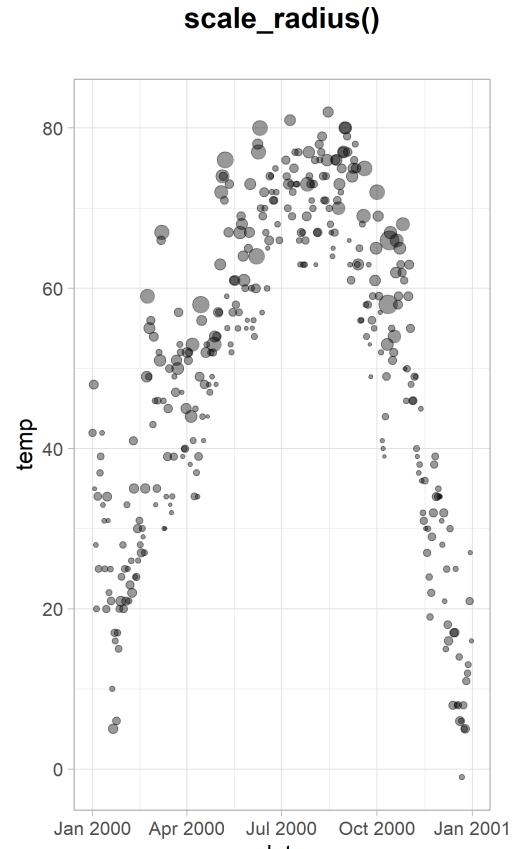
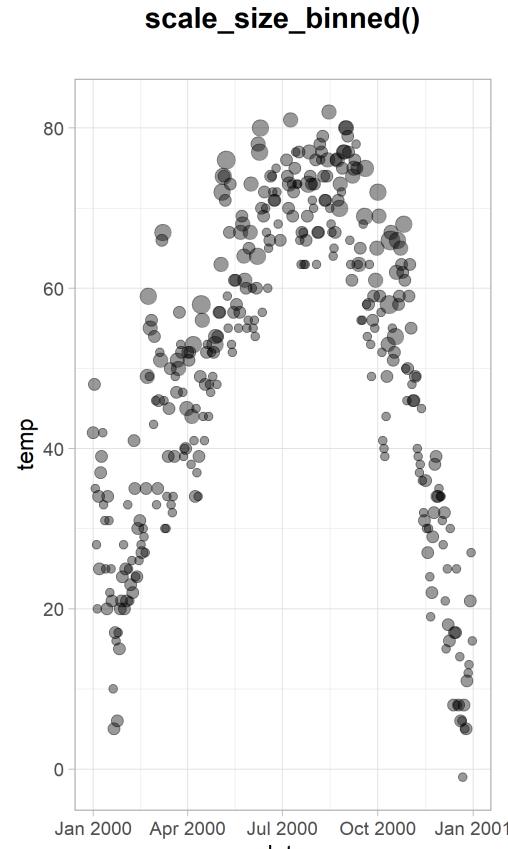
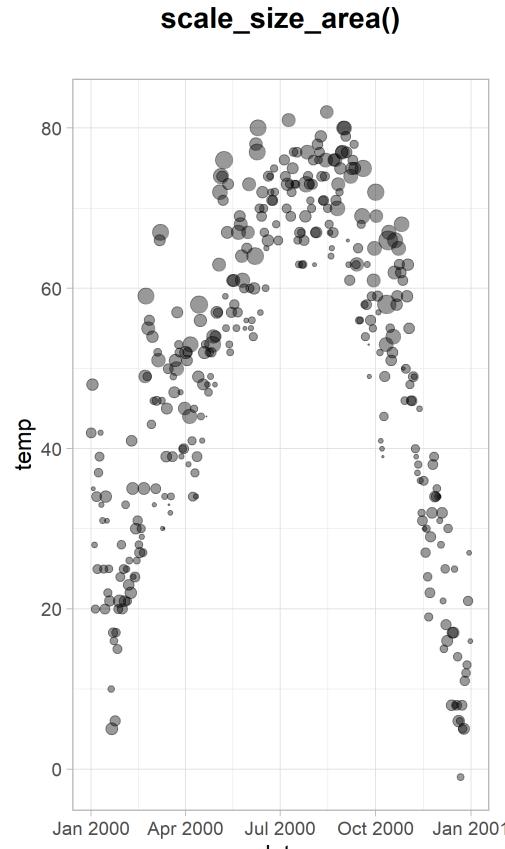
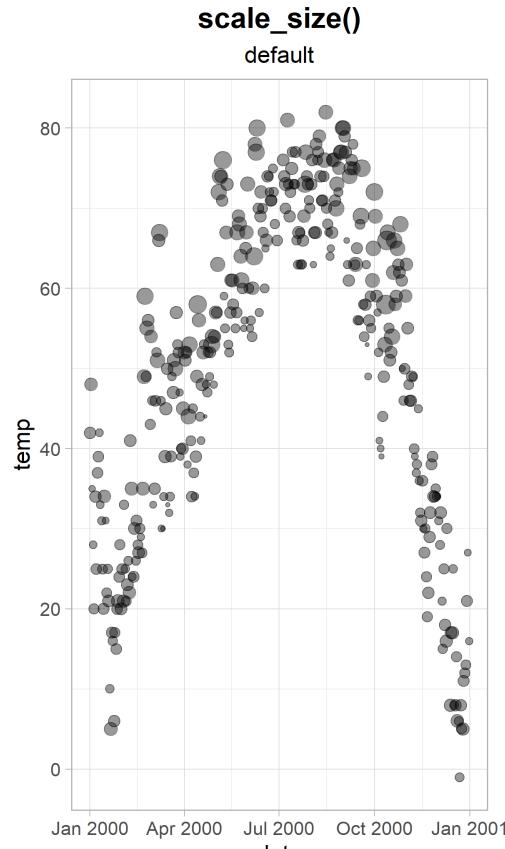


scale_radius()

```
ggplot(chic2k, aes(date, temp)) +  
  geom_point(  
    aes(size = pm10),  
    alpha = .4  
) +  
  scale_radius()
```



scale_size|radius_*

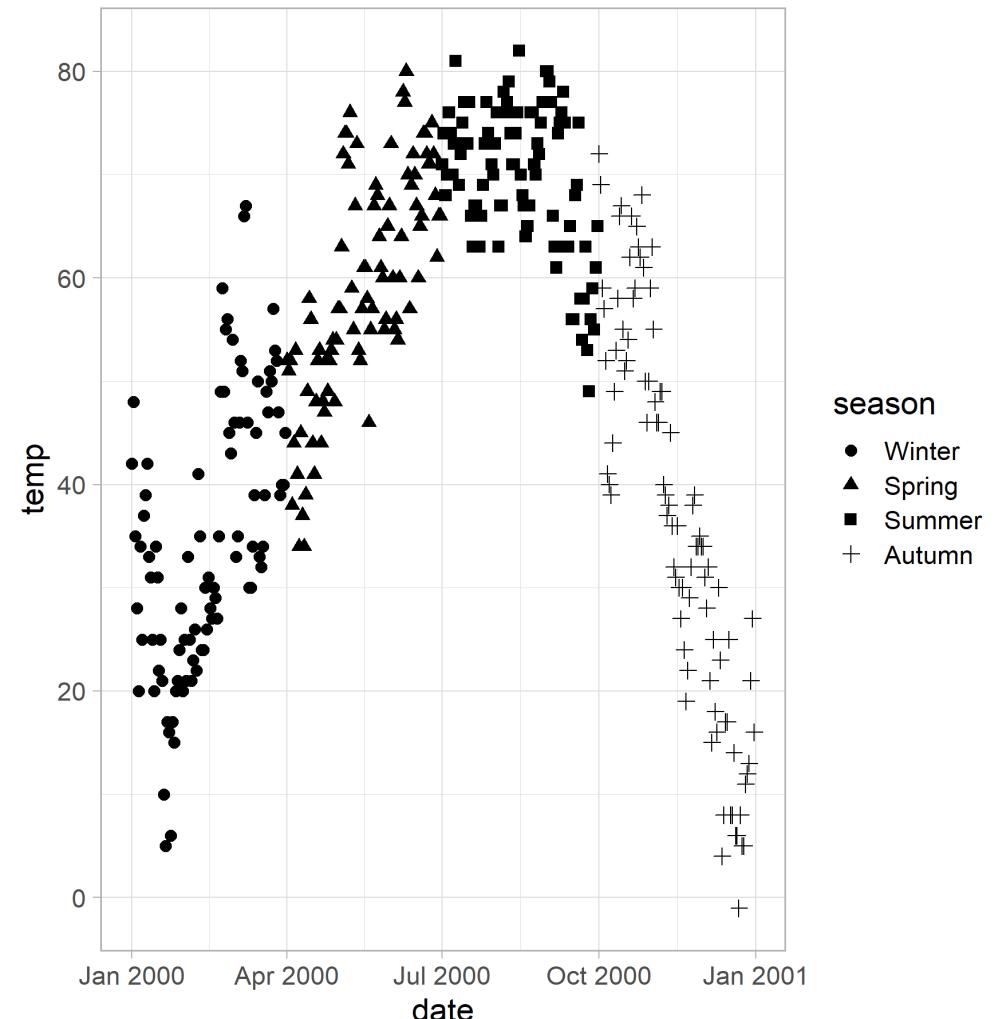


Scales

scale_shape | linetype_*()

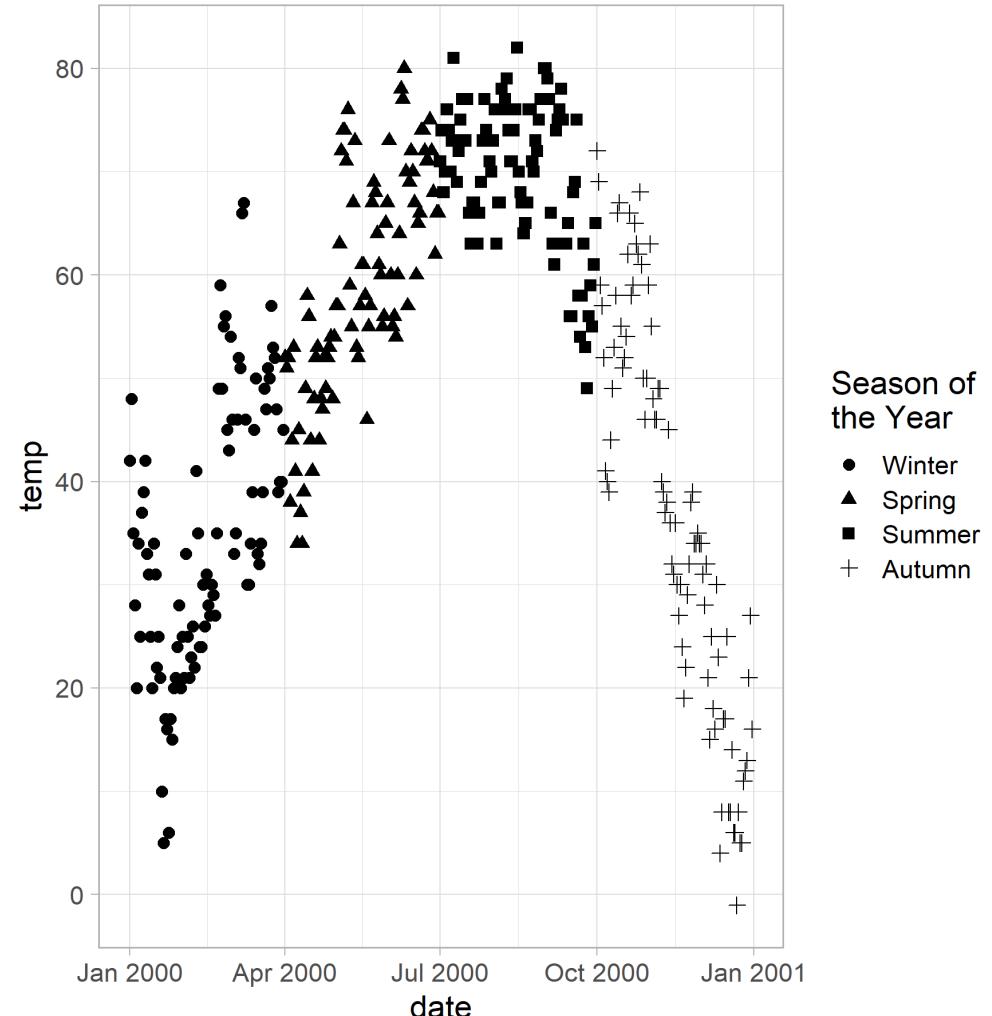
scale_shape_*

```
ggplot(chic2k, aes(date, temp)) +  
  geom_point(  
    aes(shape = season),  
    size = 2.5  
)
```



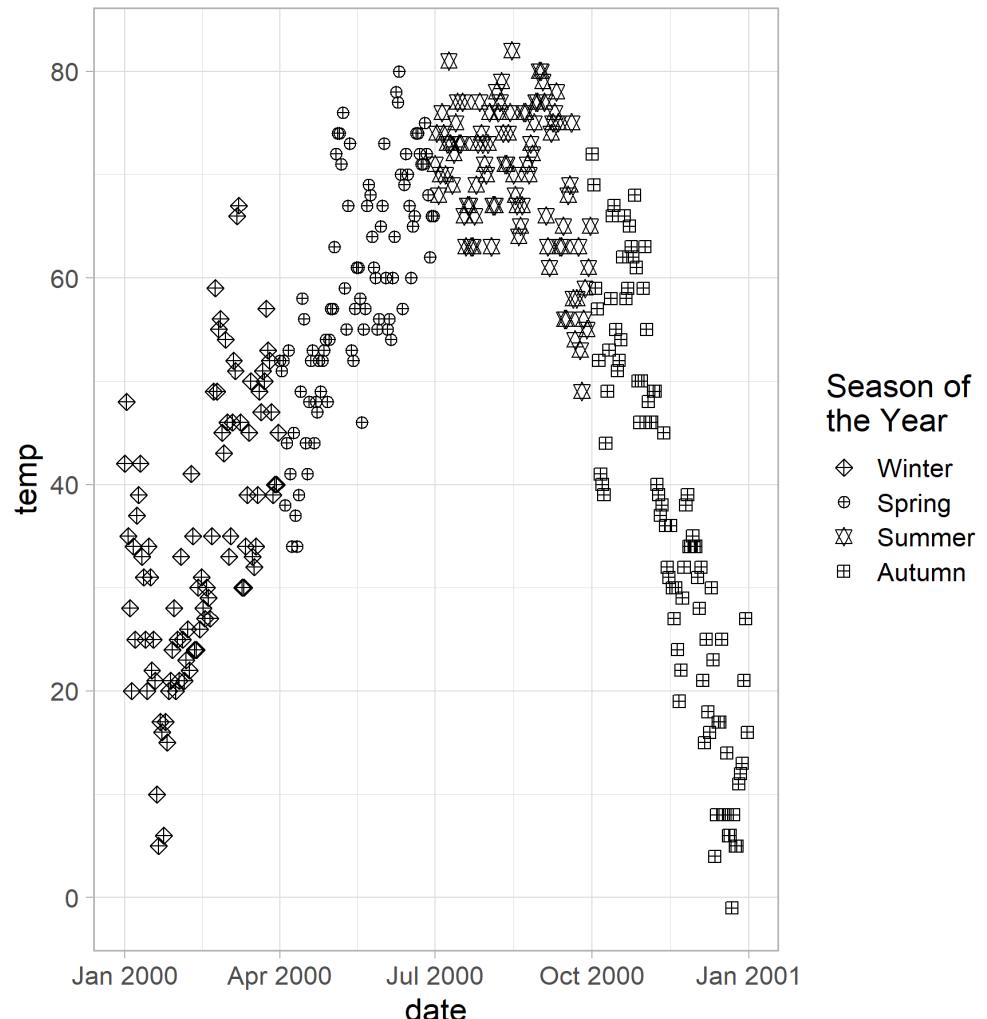
scale_shape()

```
ggplot(chic2k, aes(date, temp)) +  
  geom_point(  
    aes(shape = season),  
    size = 2.5  
  ) +  
  scale_shape(  
    name = "Season of\nthe Year"  
  )
```



scale_shape_manual()

```
ggplot(chic2k, aes(date, temp)) +  
  geom_point(  
    aes(shape = season),  
    size = 2.5  
  ) +  
  scale_shape_manual(  
    name = "Season of\nthe Year",  
    values = 9:12  
  )
```



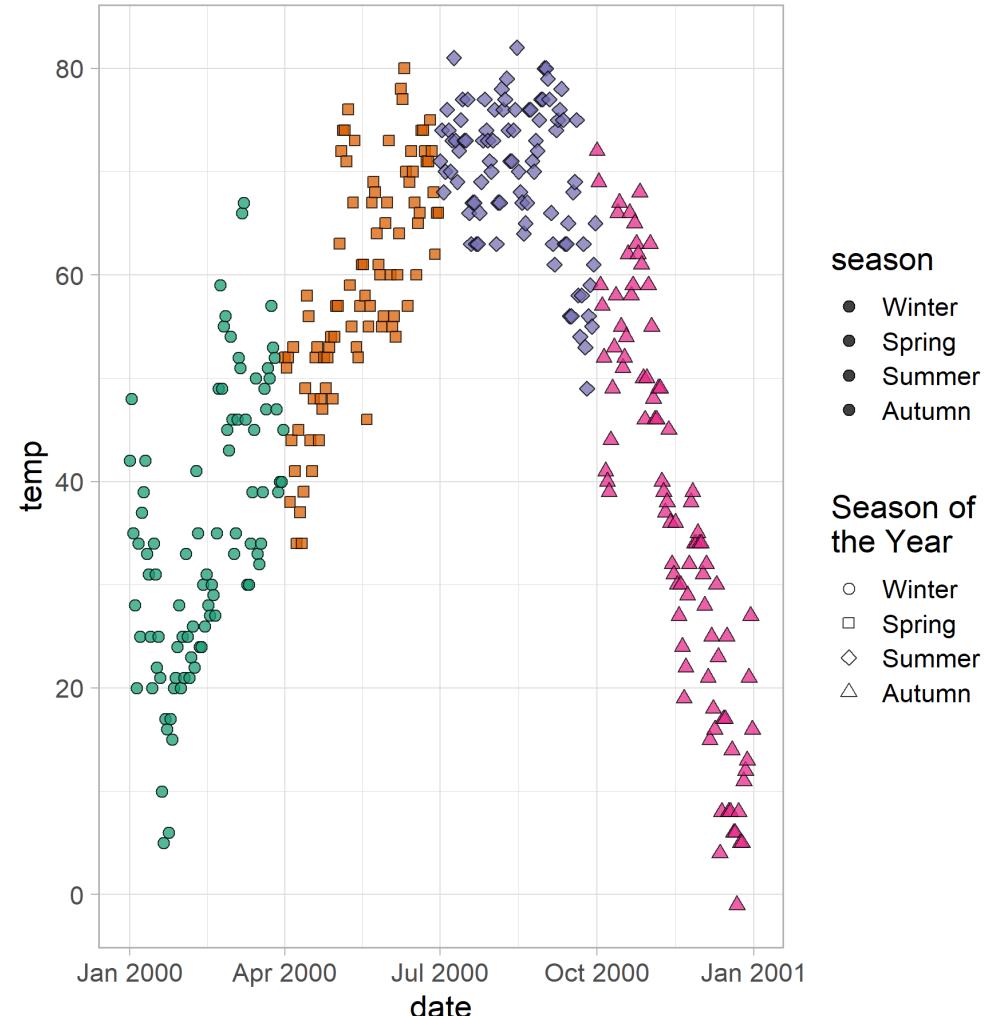
scale_shape_*()

0	1	2	3	4
□	○	△	+	×
5	6	7	8	9
◇	▽	⊗	*	◇
10	11	12	13	14
⊕	⊗	田	⊗	田
15	16	17	18	19
■	●	▲	◆	●
20	21	22	23	24
●	●	■	◆	▲

Source: ggplot2.tidyverse.org/articles/ggplot2-specs.html

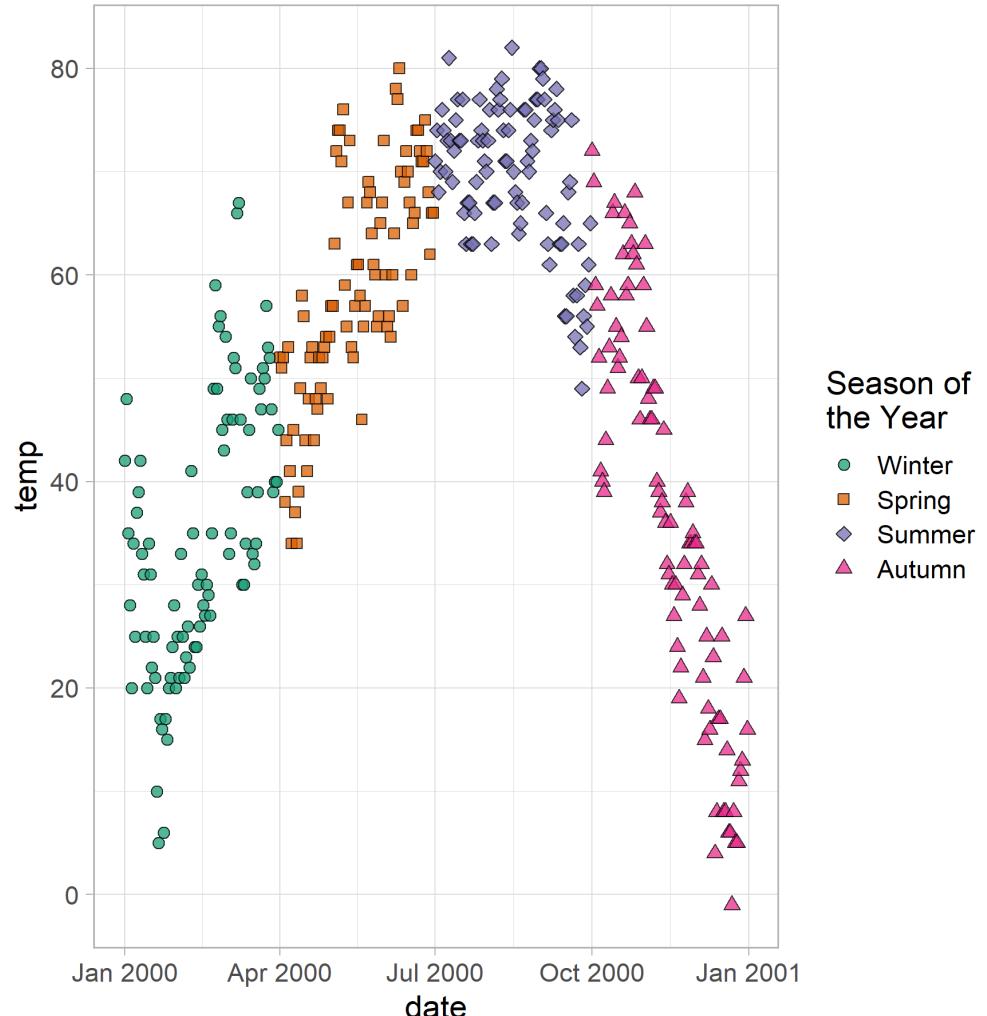
scale_shape_manual()

```
ggplot(chic2k, aes(date, temp)) +  
  geom_point(  
    aes(  
      shape = season,  
      fill = season  
    ),  
    size = 2.5,  
    alpha = .75  
  ) +  
  scale_shape_manual(  
    name = "Season of\nthe Year",  
    values = 21:24  
  ) +  
  scale_fill_brewer(  
    palette = "Dark2"  
  )
```

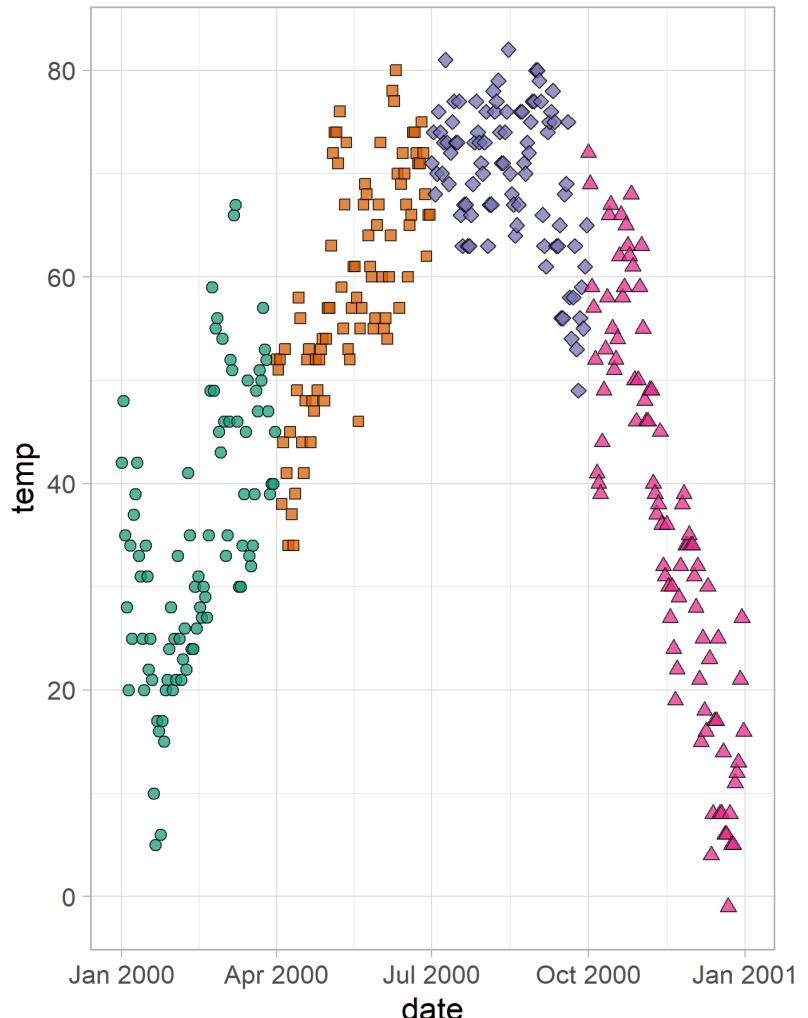


scale_shape_manual()

```
ggplot(chic2k, aes(date, temp)) +  
  geom_point(  
    aes(  
      shape = season,  
      fill = season  
    ),  
    size = 2.5,  
    alpha = .75  
  ) +  
  scale_shape_manual(  
    name = "Season of\nthe Year",  
    values = 21:24  
  ) +  
  scale_fill_brewer(  
    palette = "Dark2",  
    name = "Season of\nthe Year"  
  )
```



scale_shape_manual()

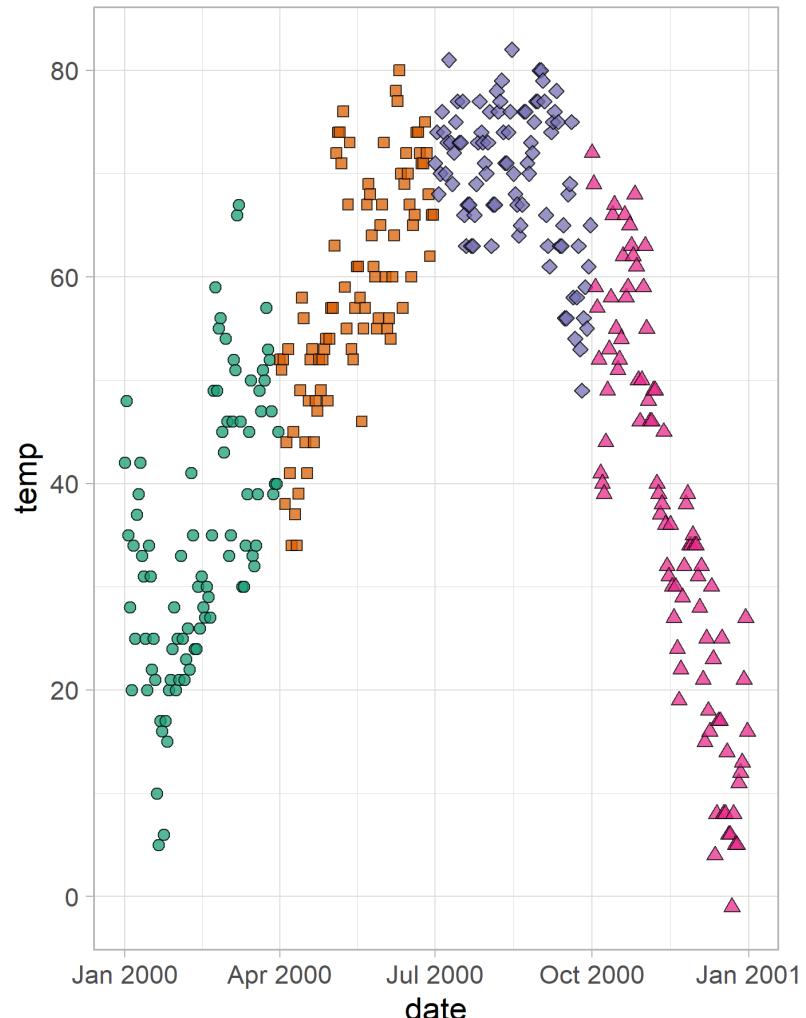


season

- Winter
- Spring
- Summer
- Autumn

Season of the Year

- Winter
- Spring
- ◊ Summer
- △ Autumn

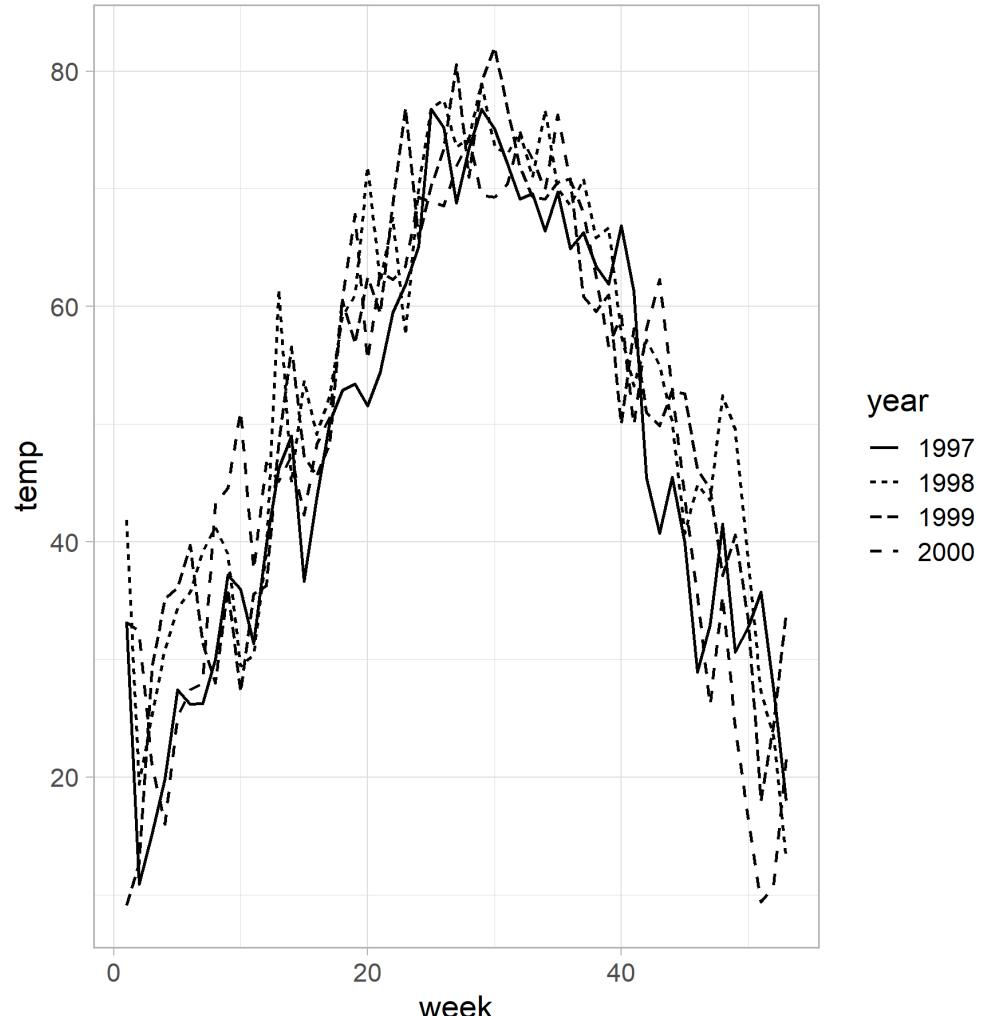


Season of the Year

- Winter
- Spring
- Summer
- Autumn

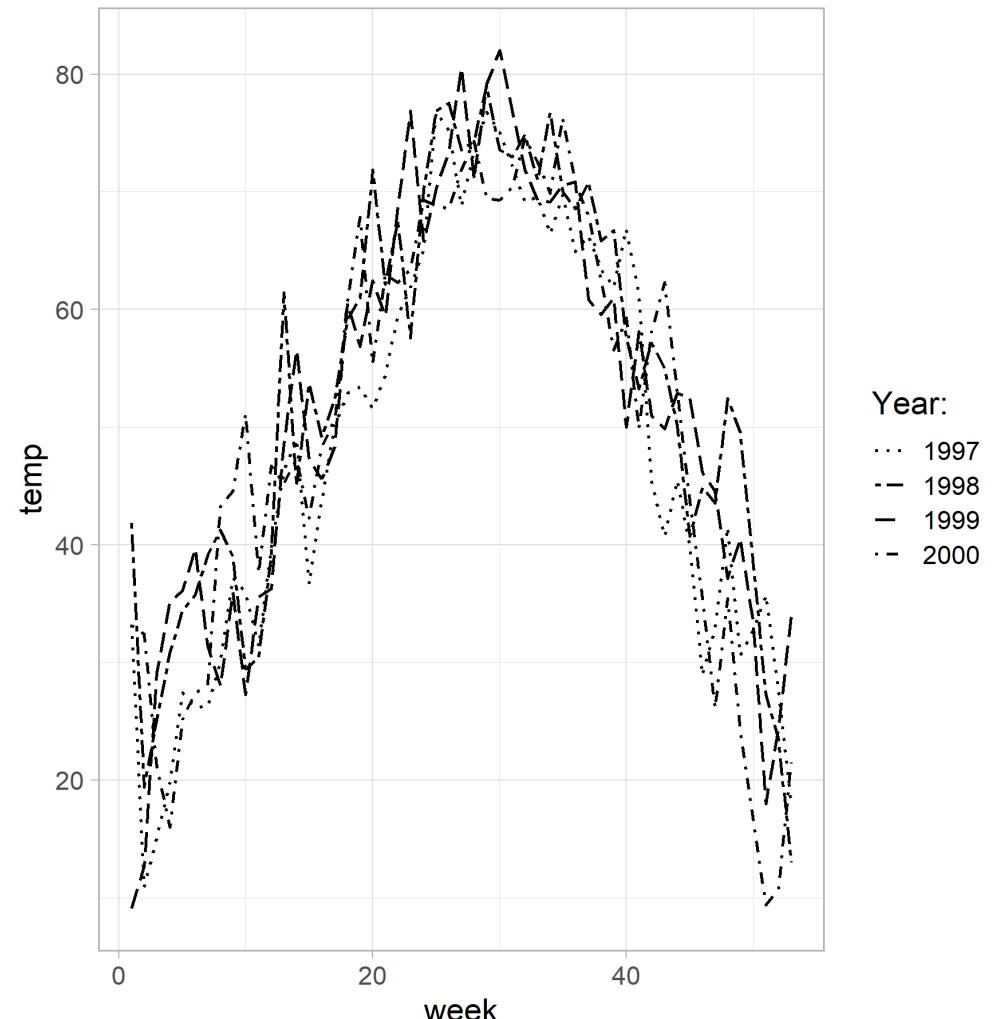
scale_linetype_*

```
chic_wy <-  
  chic %>%  
  mutate(  
    week = lubridate::week(date)  
  ) %>%  
  group_by(week, year) %>%  
  summarize(temp = mean(temp))  
  
ggplot(chic_wy, aes(week, temp)) +  
  geom_line(  
    aes(linetype = year),  
    size = .7  
)
```



scale_linetype_manual()

```
ggplot(chic_wy, aes(week, temp)) +  
  geom_line(  
    aes(linetype = year),  
    size = .7  
  ) +  
  scale_linetype_manual(  
    name = "Year:",  
    values = c("dotted",  
              "twodash",  
              "longdash",  
              "dotdash")  
  )
```

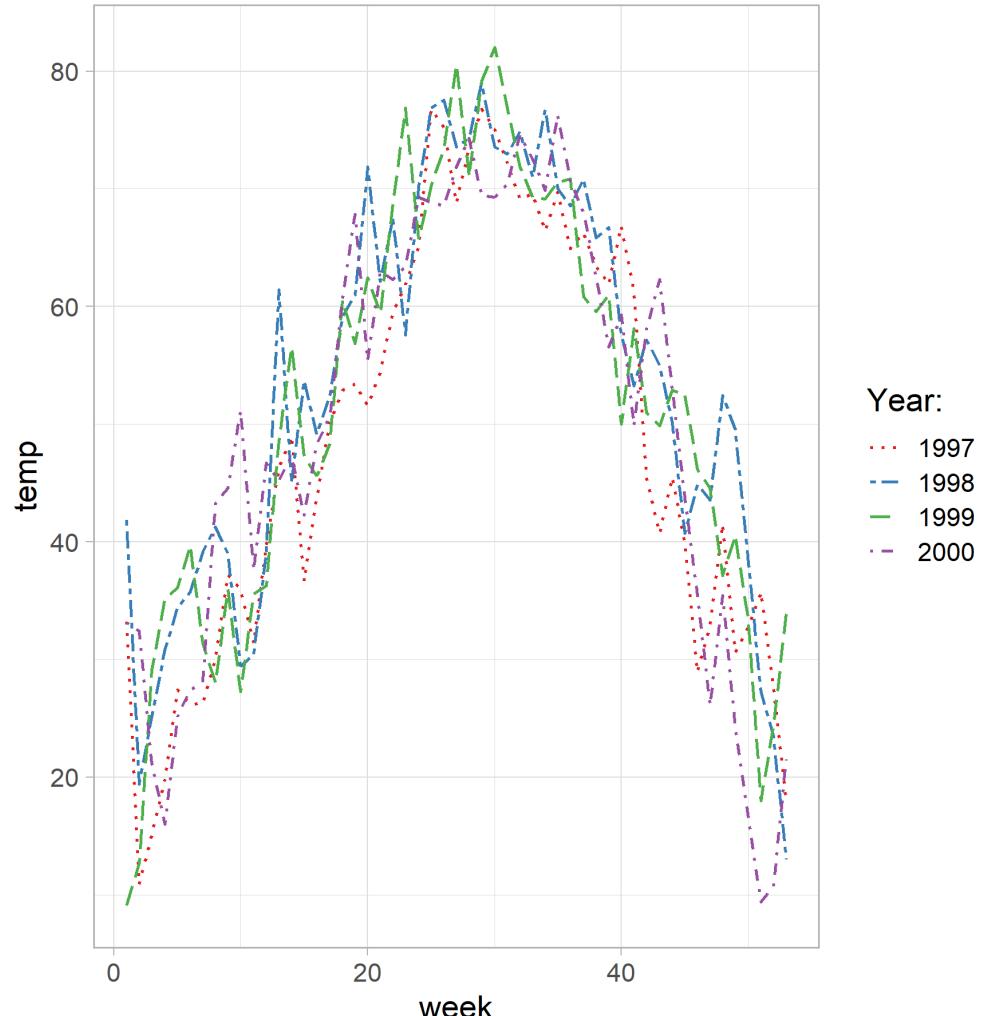


scale_linetype_*

—	lty=1 or 'solid'
- - - - -	lty=2 or 'dashed'
.....	lty=3 or 'dotted'
- . - . -	lty=4 or 'dotdash'
- - - - -	lty=5 or 'longdash'
- - - - -	lty=6 or 'twodash'

scale_linetype_manual()

```
ggplot(chic_wy, aes(week, temp)) +  
  geom_line(  
    aes(  
      linetype = year,  
      color = year  
    ),  
    size = .7  
  ) +  
  scale_linetype_manual(  
    name = "Year:",  
    values = c("dotted",  
              "twodash",  
              "longdash",  
              "dotdash")  
  ) +  
  scale_color_brewer(  
    name = "Year:",  
    palette = "Set1",  
  )
```



Scales

`scale_alpha_*`

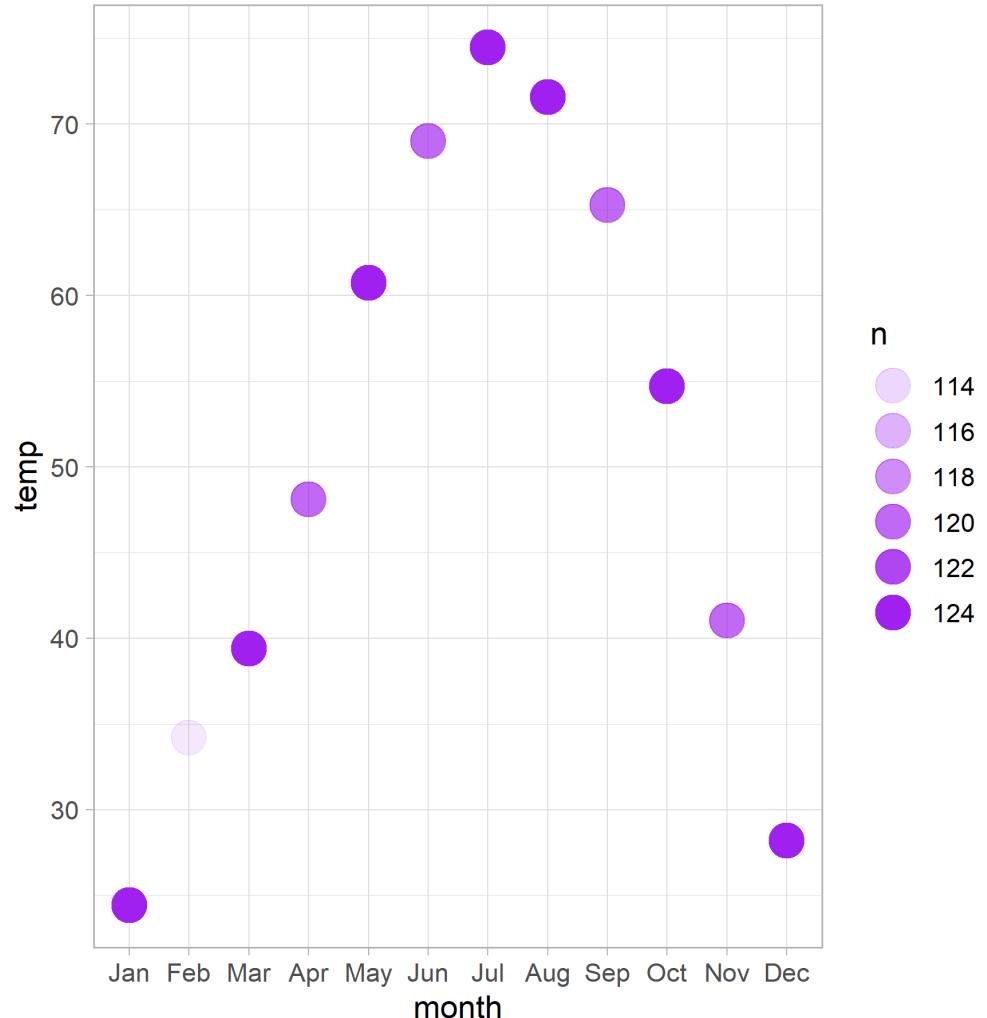
scale_alpha_*

```
chic_month <-  
  chic %>%  
  mutate(  
    month = lubridate::month(  
      date,  
      label = TRUE  
    )  
  ) %>%  
  group_by(month) %>%  
  summarize(  
    temp = mean(temp),  
    n = n()  
  ) %>%  
  ungroup()  
  
chic_month
```

```
## # A tibble: 12 x 3  
##   month   temp     n  
##   <ord>   <dbl> <int>  
## 1 Jan     24.4   124  
## 2 Feb     34.2   113  
## 3 Mar     39.4   124  
## 4 Apr     48.1   120  
## 5 May     60.7   124  
## 6 Jun     69.0   120  
## 7 Jul     74.5   124  
## 8 Aug     71.6   124  
## 9 Sep     65.3   120  
## 10 Oct    54.7   124  
## 11 Nov    41.0   120  
## 12 Dec    28.2   124
```

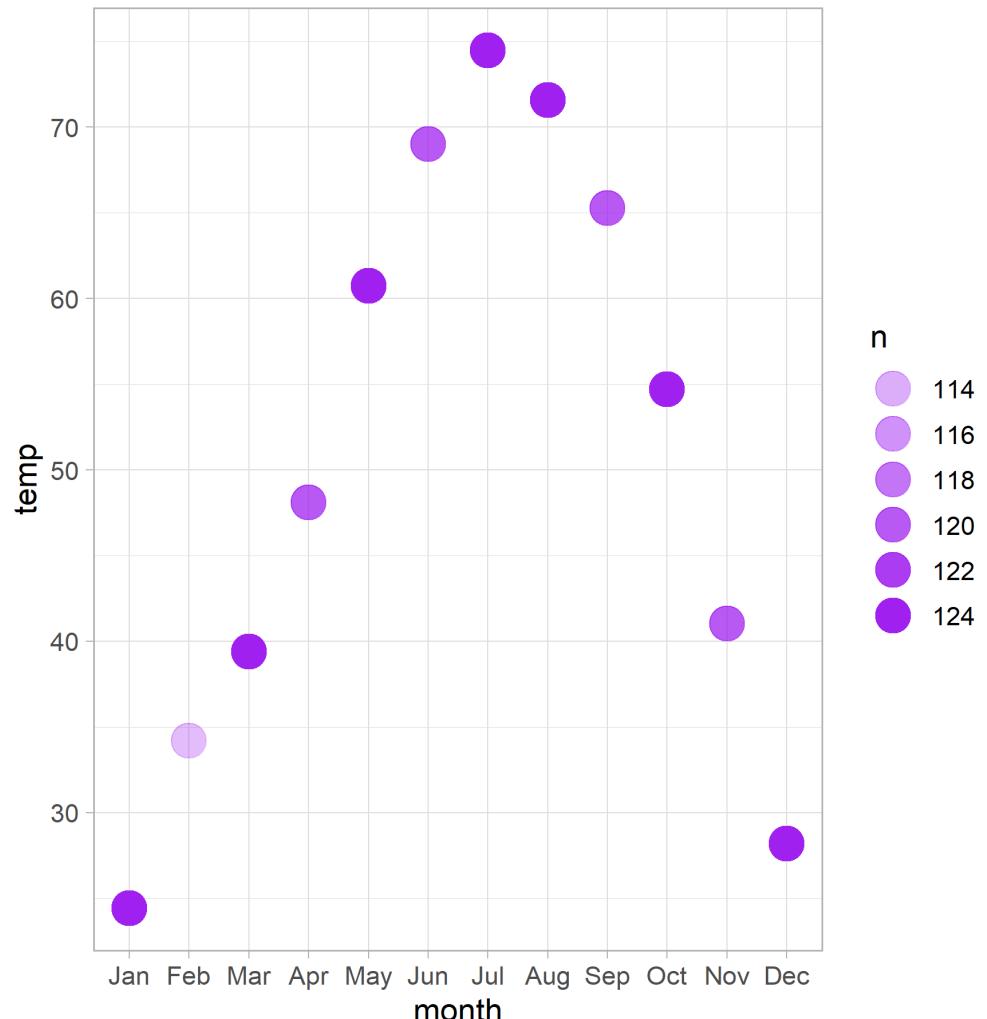
scale_alpha_*

```
ggplot(chic_month, aes(month, temp)) +  
  geom_point(  
    aes(alpha = n),  
    size = 8,  
    color = "purple"  
)
```



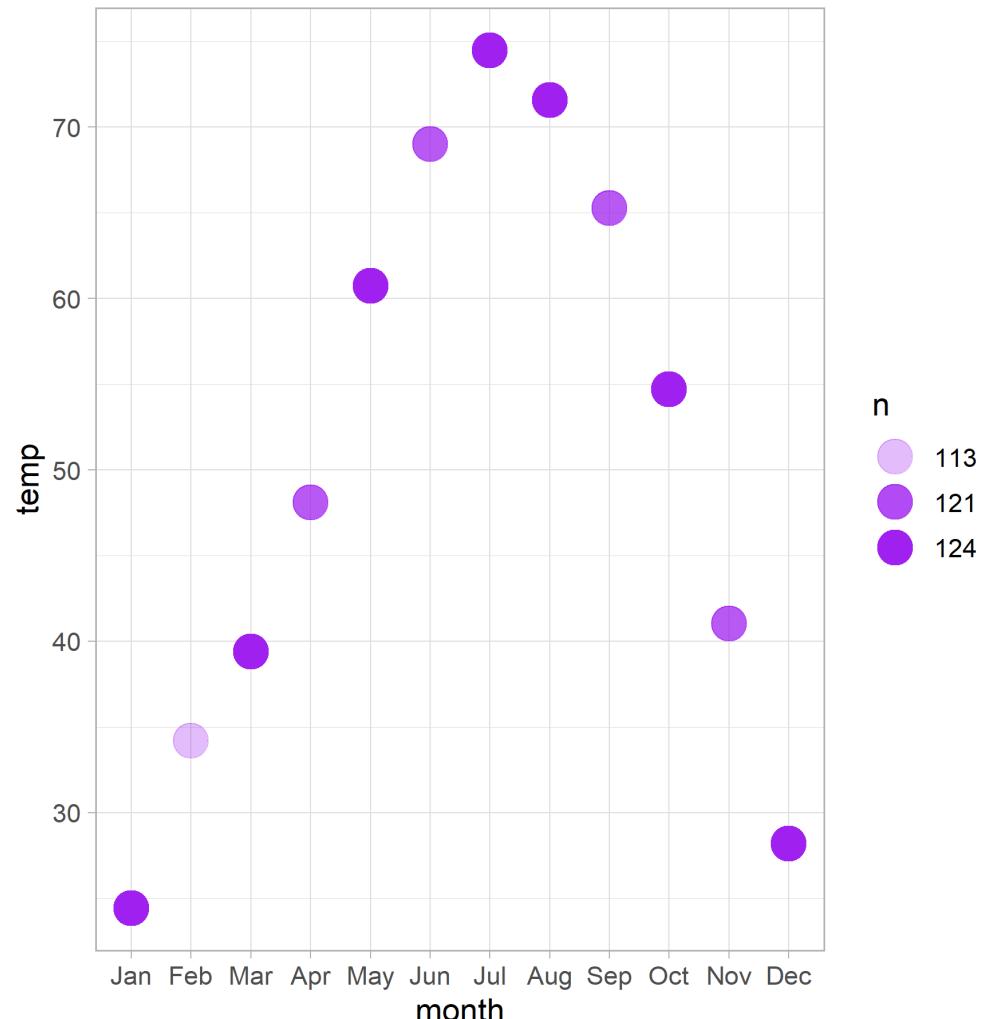
scale_alpha()

```
ggplot(chic_month, aes(month, temp)) +  
  geom_point(  
    aes(alpha = n),  
    size = 8,  
    color = "purple"  
  ) +  
  scale_alpha(  
    range = c(.3, 1)  
  )
```



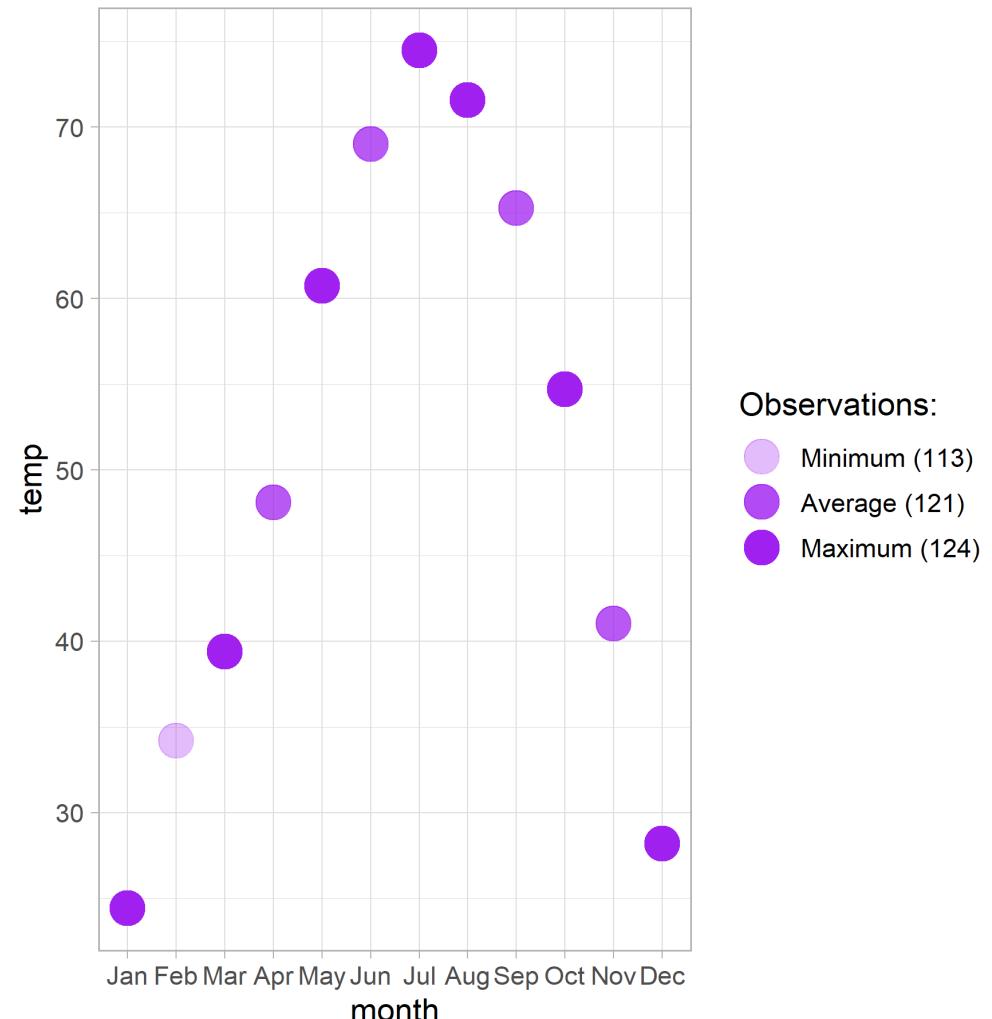
scale_alpha()

```
ggplot(chic_month, aes(month, temp)) +  
  geom_point(  
    aes(alpha = n),  
    size = 8,  
    color = "purple"  
  ) +  
  scale_alpha(  
    range = c(.3, 1),  
    breaks = c(  
      min(chic_month$n),  
      floor(mean(chic_month$n)),  
      max(chic_month$n)  
    )  
  )
```



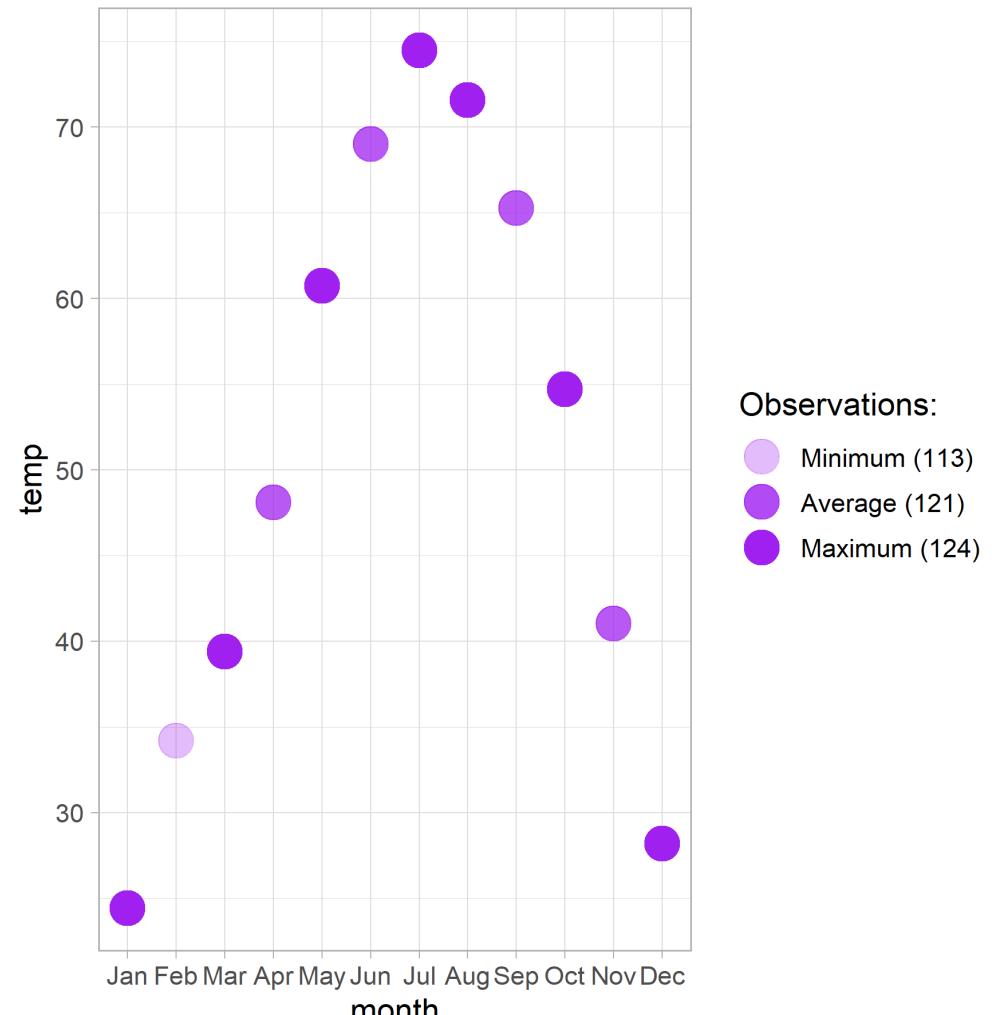
scale_alpha()

```
ggplot(chic_month, aes(month, temp)) +  
  geom_point(  
    aes(alpha = n),  
    size = 8,  
    color = "purple"  
  ) +  
  scale_alpha(  
    name = "Observations:",  
    range = c(.3, 1),  
    breaks = c(  
      min(chic_month$n),  
      floor(mean(chic_month$n)),  
      max(chic_month$n)  
    ),  
    labels = c(  
      "Minimum (113)",  
      "Average (121)",  
      "Maximum (124)"  
    )  
  )
```



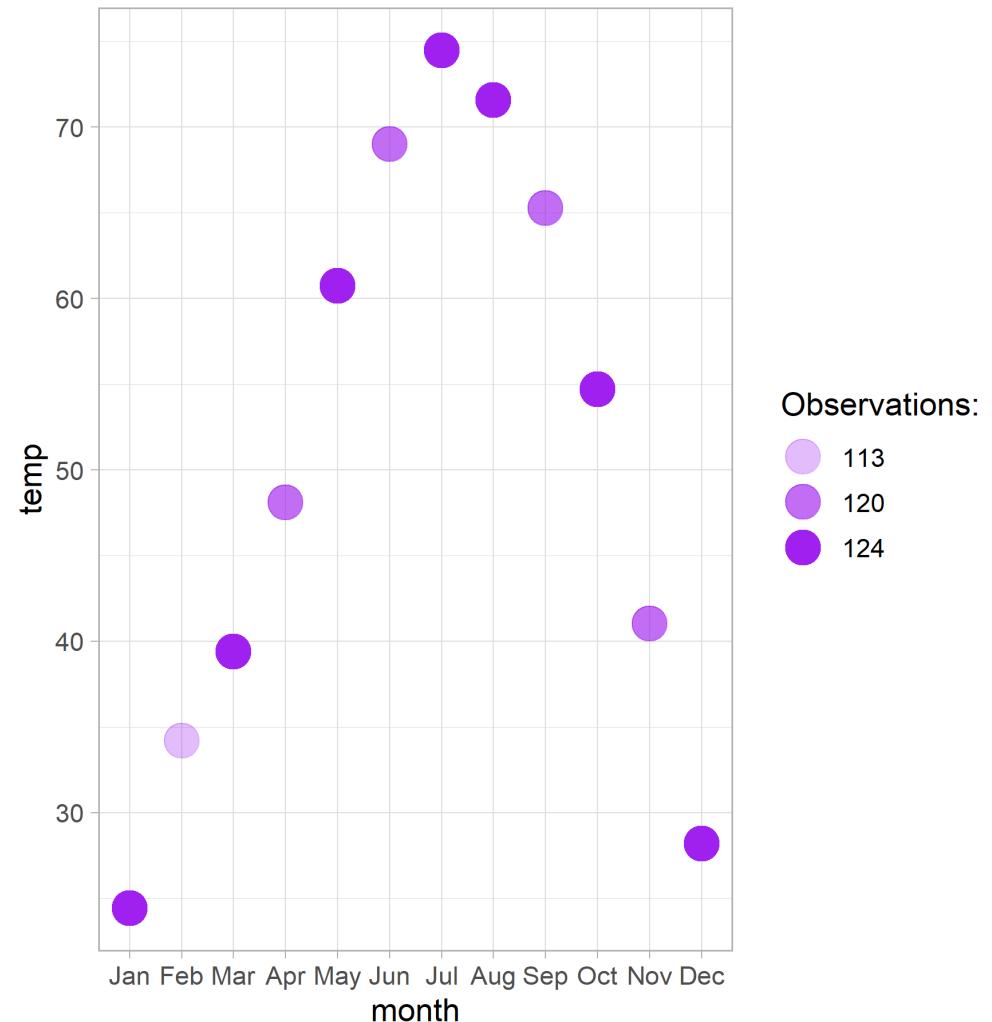
scale_alpha_continuous()

```
ggplot(chic_month, aes(month, temp)) +  
  geom_point(  
    aes(alpha = n),  
    size = 8,  
    color = "purple"  
  ) +  
  scale_alpha_continuous(  
    name = "Observations:",  
    range = c(.3, 1),  
    breaks = c(  
      min(chic_month$n),  
      floor(mean(chic_month$n)),  
      max(chic_month$n)  
    ),  
    labels = c(  
      "Minimum (113)",  
      "Average (121)",  
      "Maximum (124)"  
    )  
  )
```



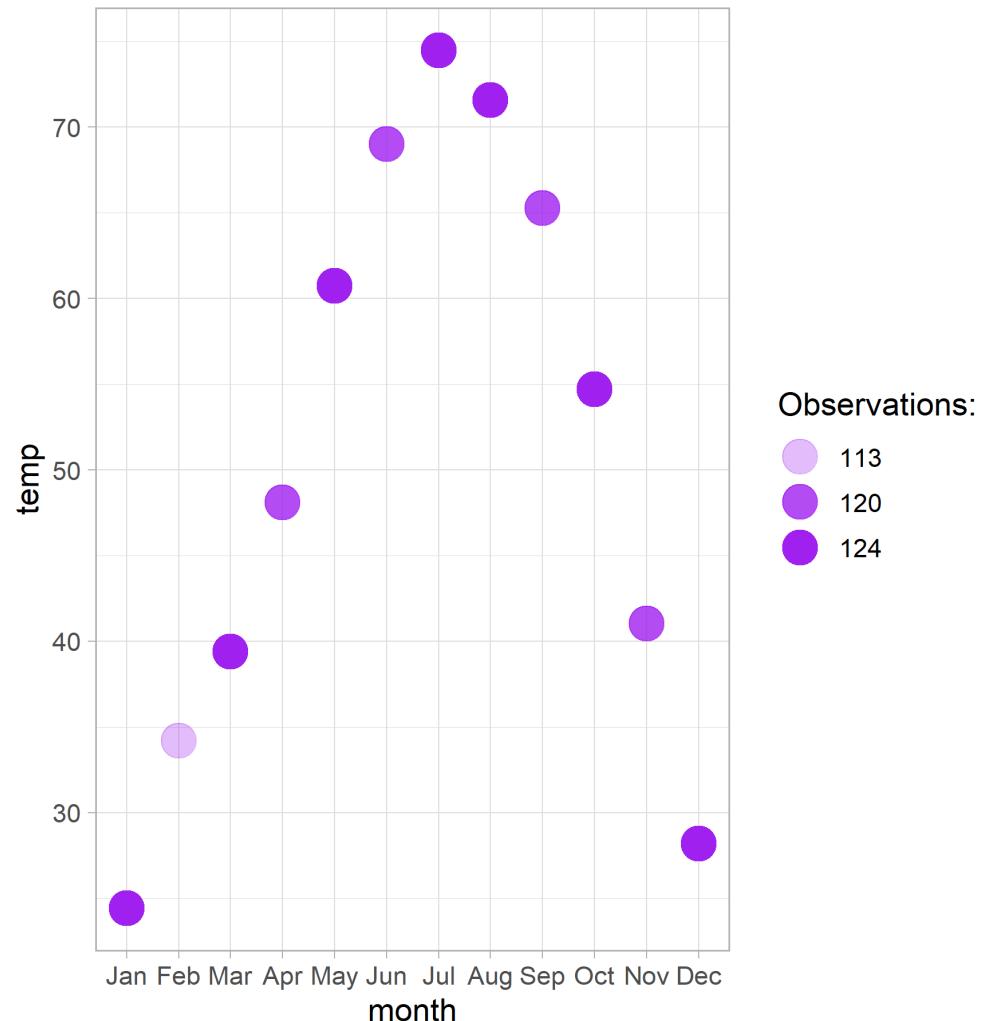
scale_alpha_discrete()

```
ggplot(chic_month, aes(month, temp)) +  
  geom_point(  
    aes(alpha = as.factor(n)),  
    size = 8,  
    color = "purple"  
  ) +  
  scale_alpha_discrete(  
    name = "Observations:",  
    range = c(.3, 1)  
  )
```



scale_alpha_discrete()

```
ggplot(chic_month, aes(month, temp)) +  
  geom_point(  
    aes(alpha = as.factor(n)),  
    size = 8,  
    color = "purple"  
  ) +  
  scale_alpha_manual(  
    name = "Observations:",  
    values = c(.3, .8, 1)  
  )
```

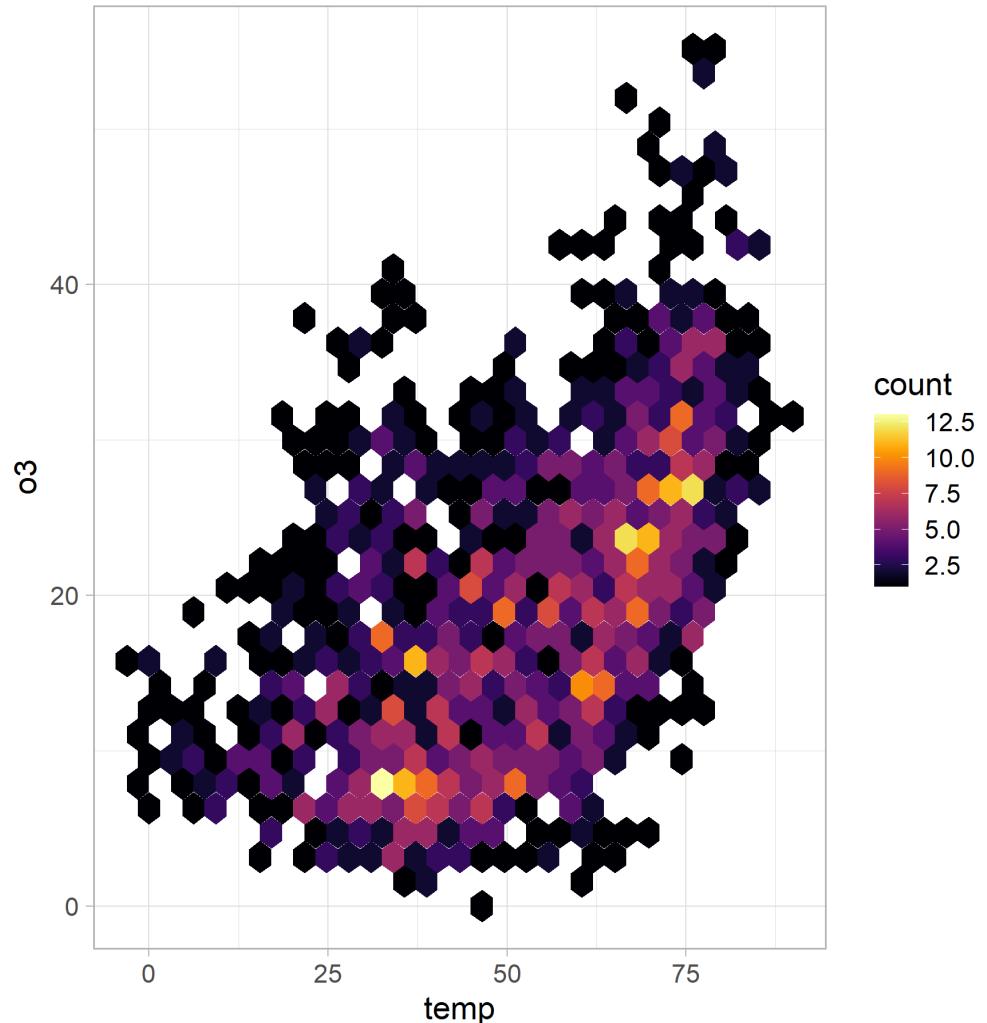


Your Turn!

- Visualize the relationship of temperature and ozone in Chicago using a hexagonal binning and a (non-green) viridis color palette.
- Afterwards, make the axis title more descriptive and change the legend title to "Count:".
- Now change the x axis so it shows the temperature every 10°F in a range of -10°F to 90°F.
- Turn the labels on the y axis into two string labels (e.g. "low" and "high").
- Explore some other color palettes that are available in other packages, pick one you like and change the color palette of the plot.
- Check if your final plot is color-blindness safe either online, e.g. color-blindness.com/coblis-color-blindness-simulator or using the package `colorblindr` (*Note: You need to install again some development versions of packages, follow the instructions on <https://github.com/clauswilke/colorblindr>*)

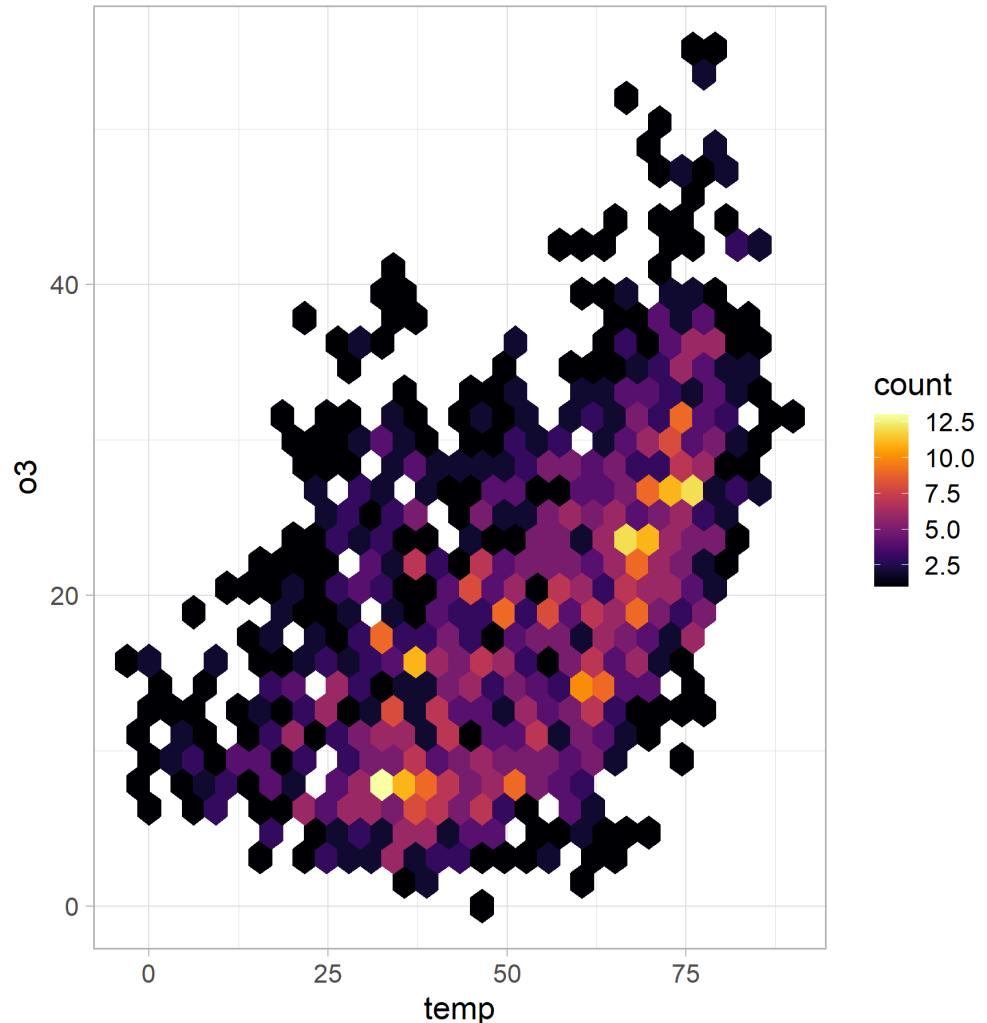
Your Turn: Viridis-Hexagons

```
ggplot(chic, aes(temp, o3)) +  
  geom_hex() +  
  viridis::scale_fill_viridis(  
    option = "inferno"  
)
```



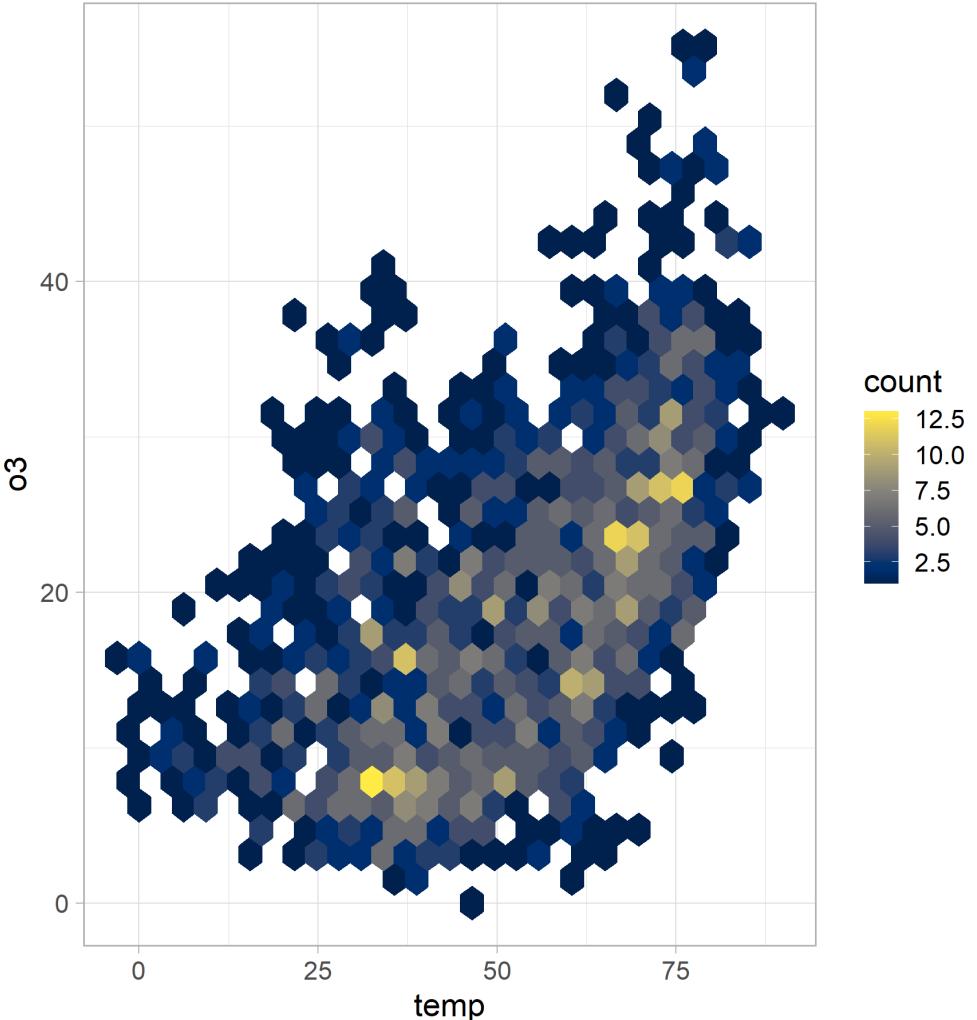
Your Turn: Viridis-Hexagons

```
ggplot(chic, aes(temp, o3)) +  
  geom_hex(  
    aes(  
      #fill = ..count.., ## the default  
      color = ..count..  
    )  
  ) +  
  viridis::scale_fill_viridis(  
    option = "inferno"  
  ) +  
  viridis::scale_color_viridis(  
    option = "inferno"  
  )
```



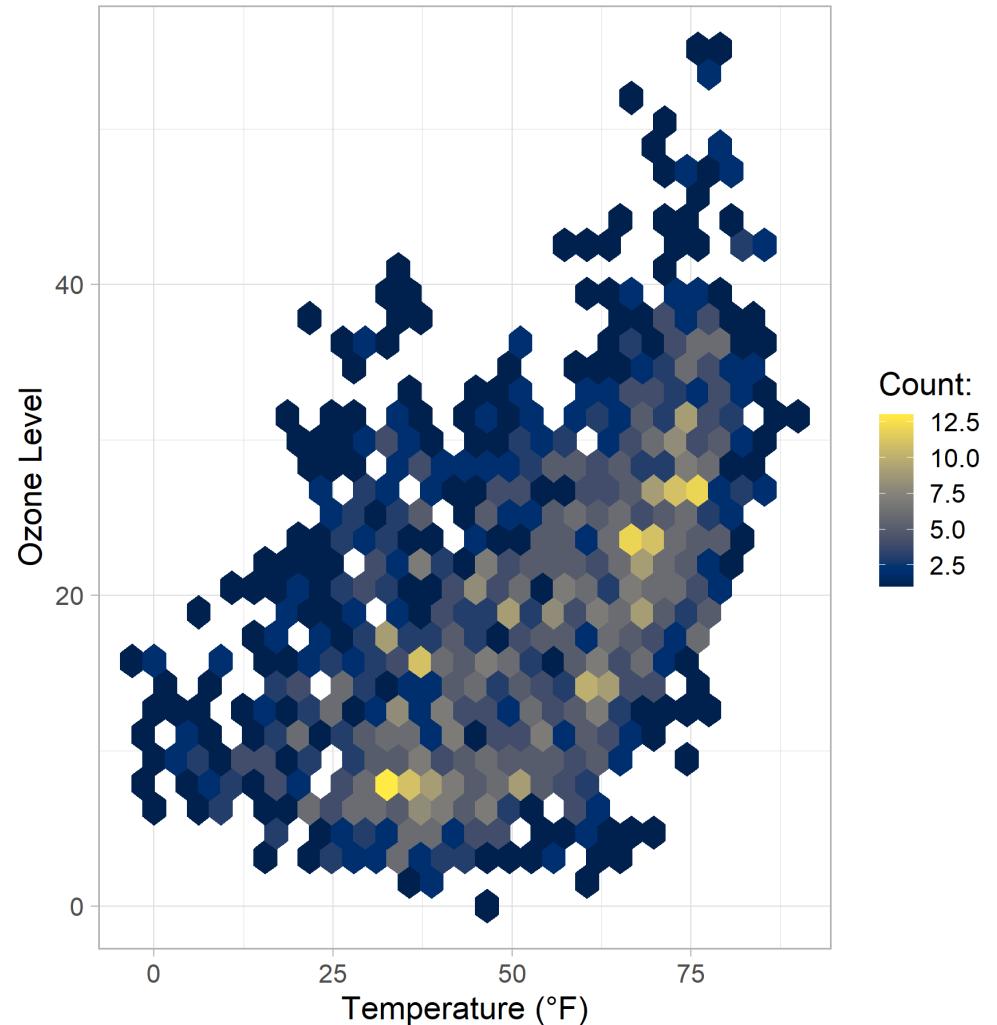
Your Turn: Viridis-Hexagons

```
ggplot(chic, aes(temp, o3)) +  
  geom_hex(  
    aes(  
      #fill = ..count.., ## the default  
      color = ..count..  
    )  
  ) +  
  viridis::scale_fill_viridis(  
    option = "cividis"  
  ) +  
  viridis::scale_color_viridis(  
    option = "cividis"  
  )
```



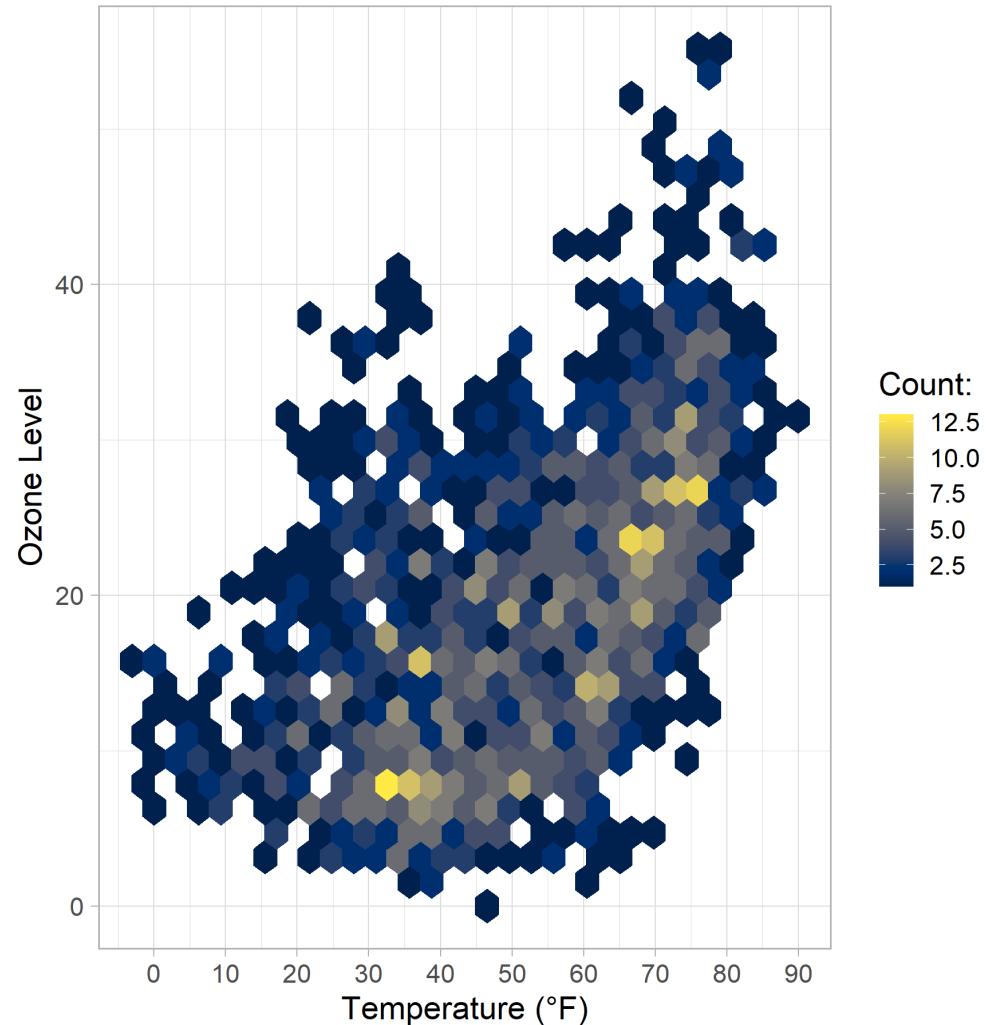
Your Turn: Viridis-Hexagons

```
ggplot(chic, aes(temp, o3)) +  
  geom_hex(aes(color = ..count..)) +  
  viridis::scale_fill_viridis(  
    option = "cividis",  
    name = "Count:"  
) +  
  viridis::scale_color_viridis(  
    option = "cividis",  
    name = "Count:"  
) +  
  labs(  
    x = "Temperature (°F)",  
    y = "Ozone Level"  
)
```



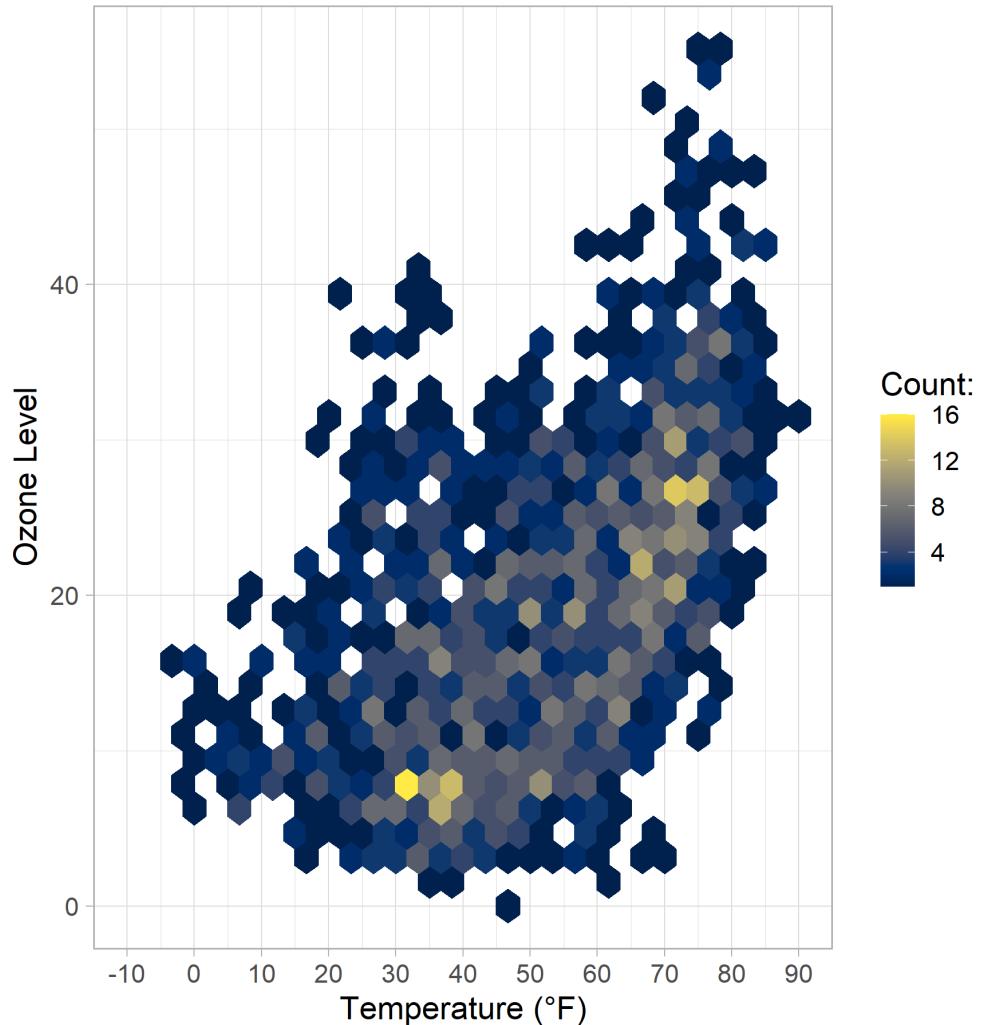
Your Turn: Viridis-Hexagons

```
ggplot(chic, aes(temp, o3)) +  
  geom_hex(aes(color = ..count..)) +  
  scale_x_continuous(  
    breaks = seq(-10, 90, by = 10)  
  ) +  
  viridis::scale_fill_viridis(  
    option = "cividis",  
    name = "Count:"  
  ) +  
  viridis::scale_color_viridis(  
    option = "cividis",  
    name = "Count:"  
  ) +  
  labs(  
    x = "Temperature (°F)",  
    y = "Ozone Level"  
  )
```



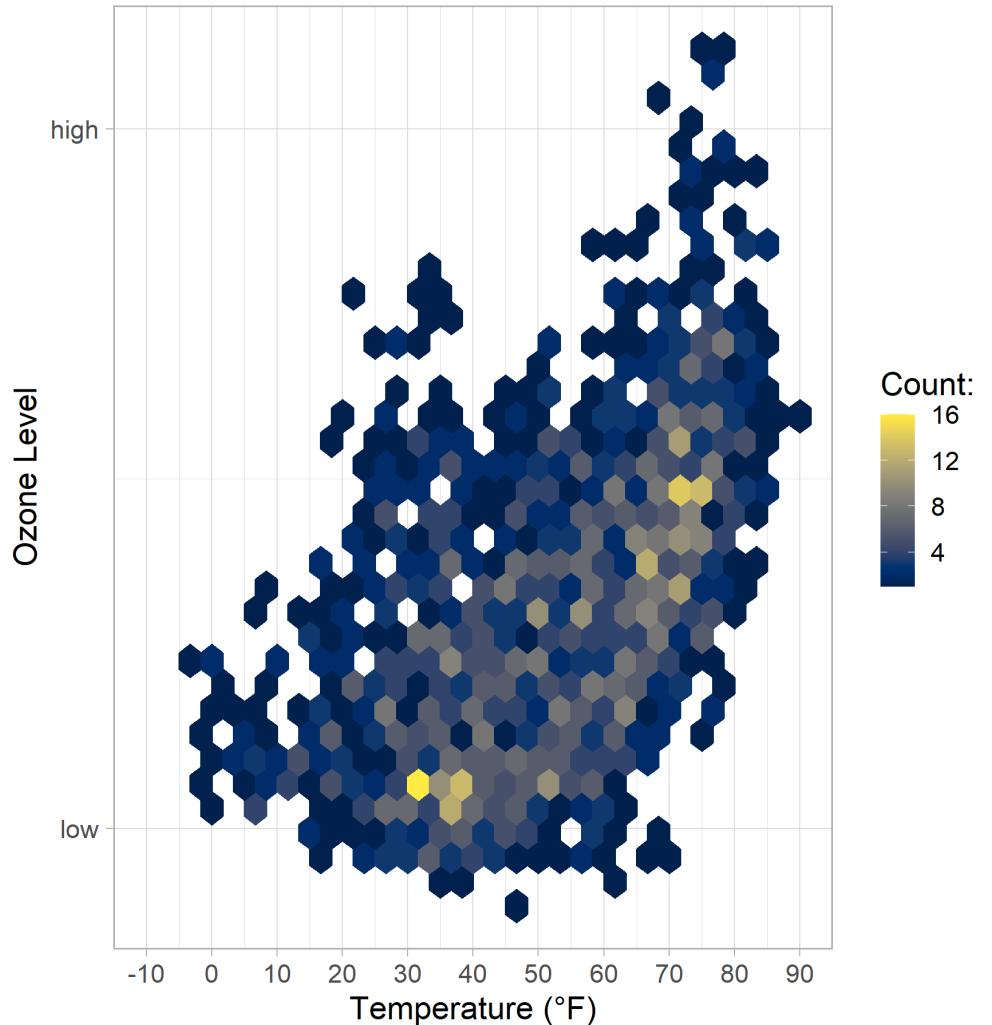
Your Turn: Viridis-Hexagons

```
ggplot(chic, aes(temp, o3)) +  
  geom_hex(aes(color = ..count..)) +  
  scale_x_continuous(  
    breaks = seq(-10, 90, by = 10),  
    limits = c(-10, NA)  
  ) +  
  viridis::scale_fill_viridis(  
    option = "cividis",  
    name = "Count:"  
  ) +  
  viridis::scale_color_viridis(  
    option = "cividis",  
    name = "Count:"  
  ) +  
  labs(  
    x = "Temperature (°F)",  
    y = "Ozone Level"  
  )
```



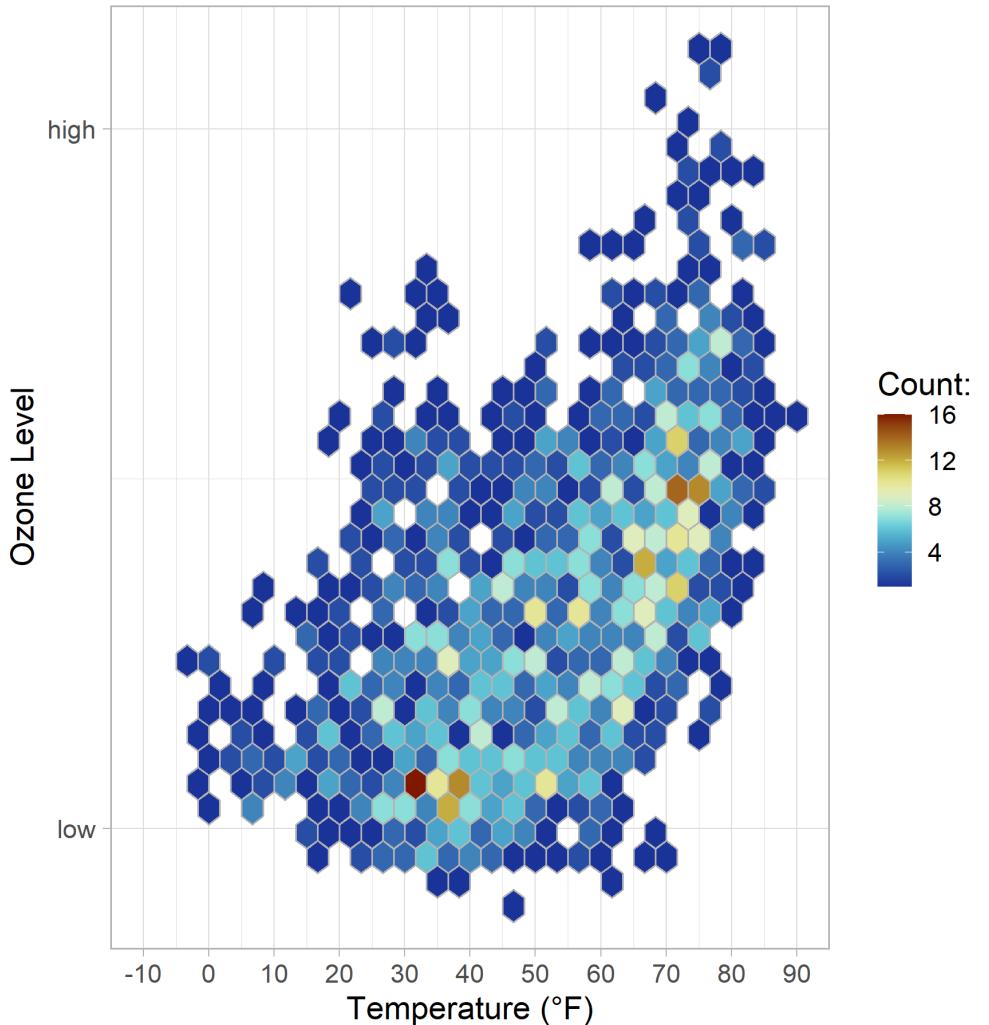
Your Turn: Viridis-Hexagons

```
ggplot(chic, aes(temp, o3)) +  
  geom_hex(aes(color = ..count..)) +  
  scale_x_continuous(  
    breaks = seq(-10, 90, by = 10),  
    limits = c(-10, NA)  
  ) +  
  scale_y_continuous(  
    breaks = c(5, 50),  
    labels = c("low", "high")  
  ) +  
  viridis::scale_fill_viridis(  
    option = "cividis",  
    name = "Count:"  
  ) +  
  viridis::scale_color_viridis(  
    option = "cividis",  
    name = "Count:"  
  ) +  
  labs(  
    x = "Temperature (°F)",  
    y = "Ozone Level"  
  )
```



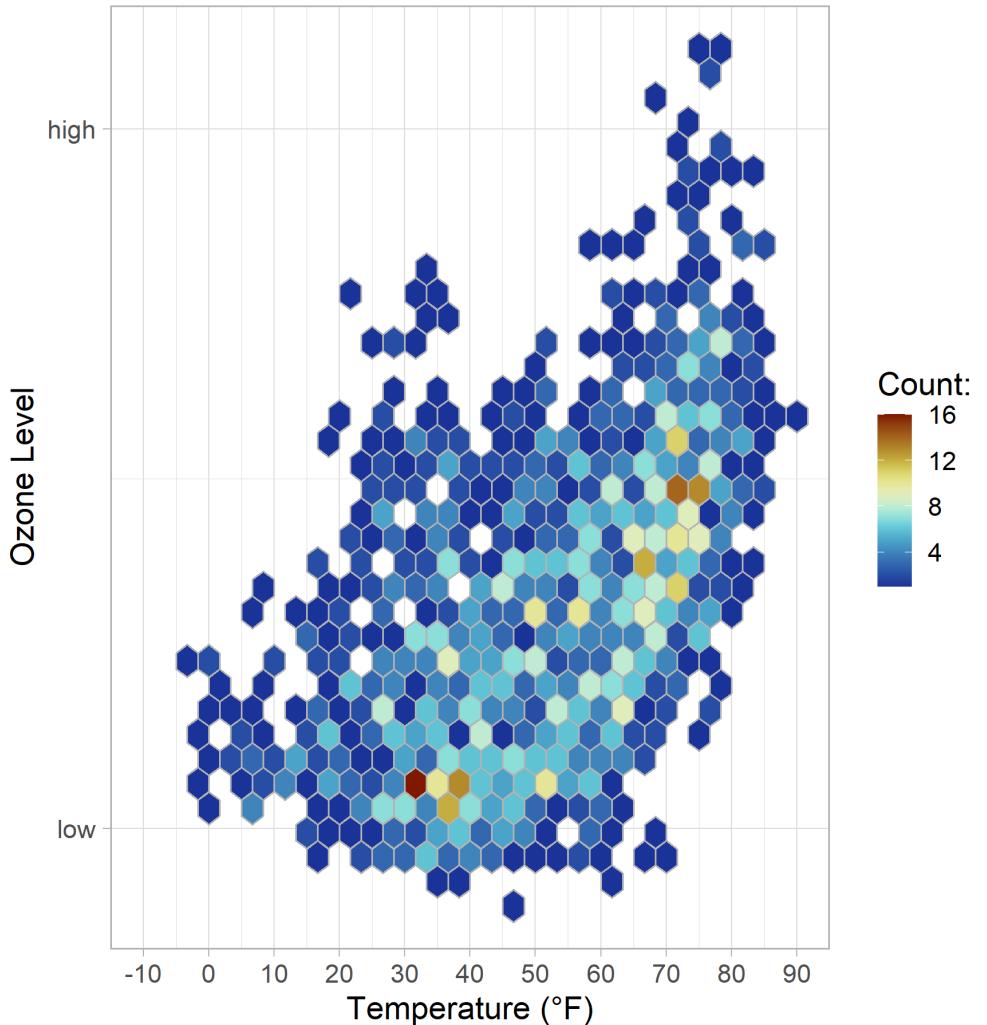
Your Turn: Viridis-Hexagons

```
ggplot(chic, aes(temp, o3)) +  
  geom_hex(color = "grey70") +  
  scale_x_continuous(  
    breaks = seq(-10, 90, by = 10),  
    limits = c(-10, NA)  
  ) +  
  scale_y_continuous(  
    breaks = c(5, 50),  
    labels = c("low", "high")  
  ) +  
  scico::scale_fill_scico(  
    palette = "roma",  
    name = "Count:",  
    direction = -1  
  ) +  
  labs(  
    x = "Temperature (°F)",  
    y = "Ozone Level"  
  )
```



Your Turn: Viridis-Hexagons

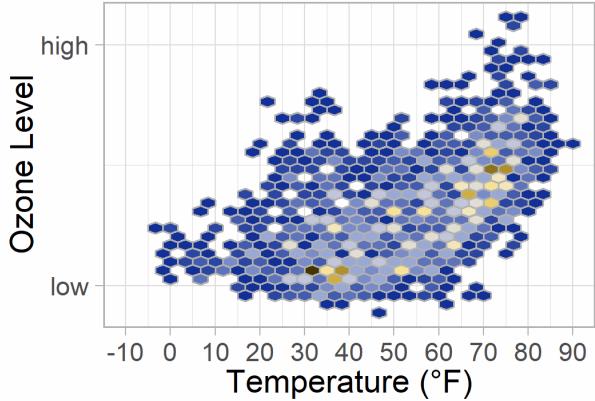
```
(g <- ggplot(chic, aes(temp, o3)) +  
  geom_hex(color = "grey70") +  
  scale_x_continuous(  
    breaks = seq(-10, 90, by = 10),  
    limits = c(-10, NA)  
  ) +  
  scale_y_continuous(  
    breaks = c(5, 50),  
    labels = c("low", "high")  
  ) +  
  scico::scale_fill_scico(  
    palette = "roma",  
    name = "Count:",  
    direction = -1  
  ) +  
  labs(  
    x = "Temperature (°F)",  
    y = "Ozone Level"  
)
```



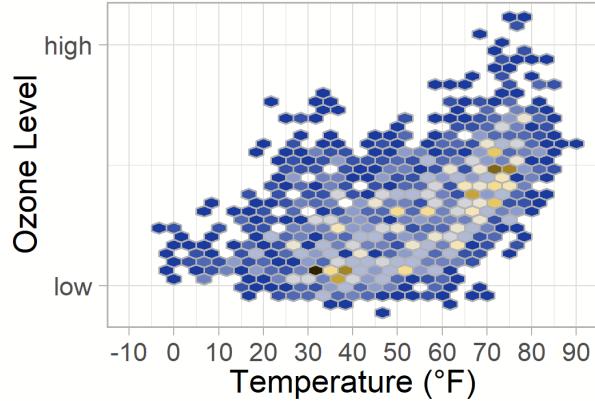
Your Turn: Viridis-Hexagons

```
colorblindr:::cvd_grid(g)
```

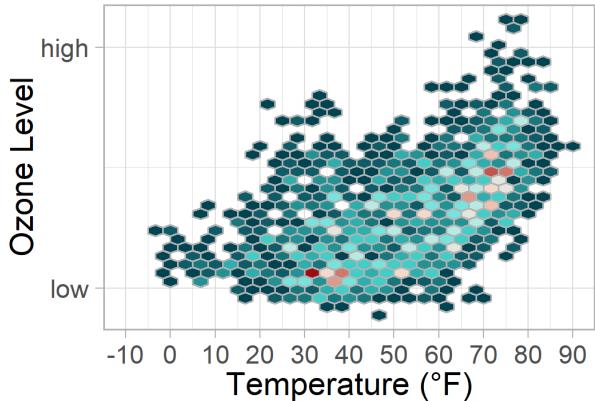
Deutanomaly



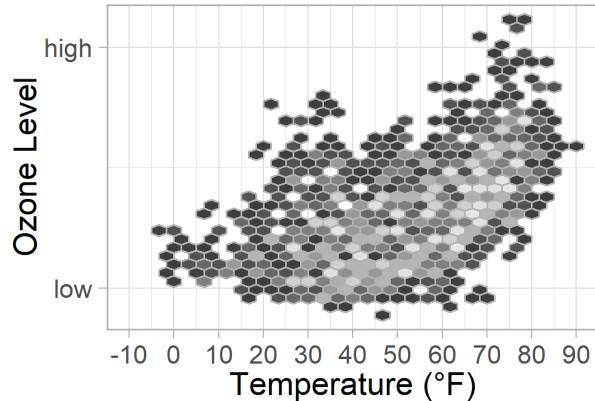
Protanomaly



Tritanomaly



Desaturated



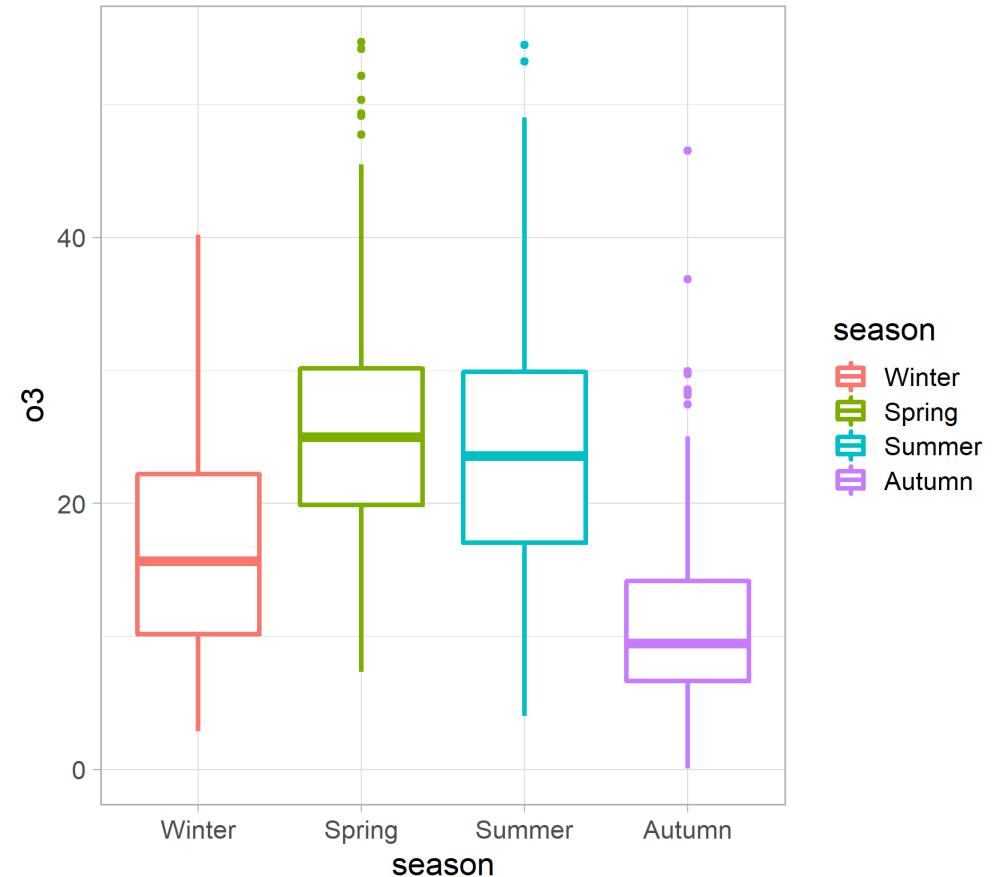
Scales

Modify aesthetics with
after_scale()

after_scale()

Since ggplot 3.0.0 it is possible to modify aesthetics after the data has been transformed and mapped:

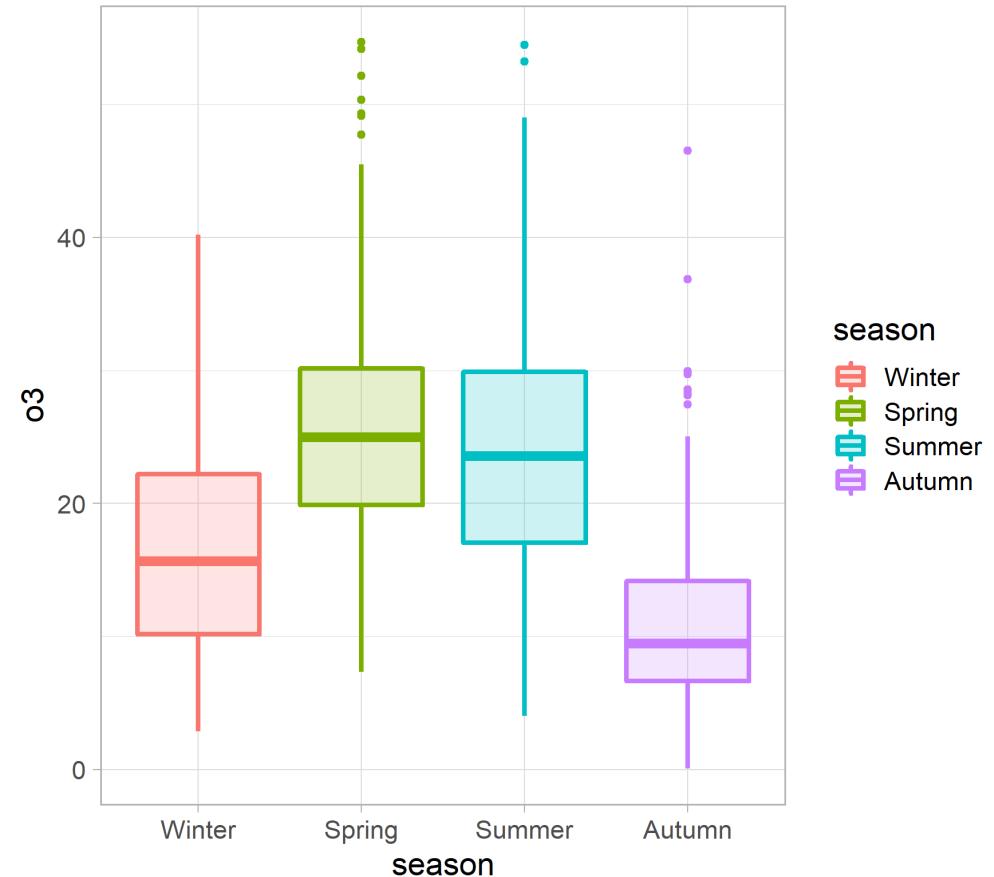
```
ggplot(chic, aes(season, o3)) +  
  geom_boxplot(  
    aes(  
      color = season  
    ),  
    size = 1.2  
  )
```



after_scale()

Since ggplot 3.0.0 it is possible to modify aesthetics after the data has been transformed and mapped:

```
ggplot(chic, aes(season, o3)) +  
  geom_boxplot(  
    aes(  
      color = season,  
      fill = after_scale(alpha(color, .2))  
    ),  
    size = 1.2  
  )
```



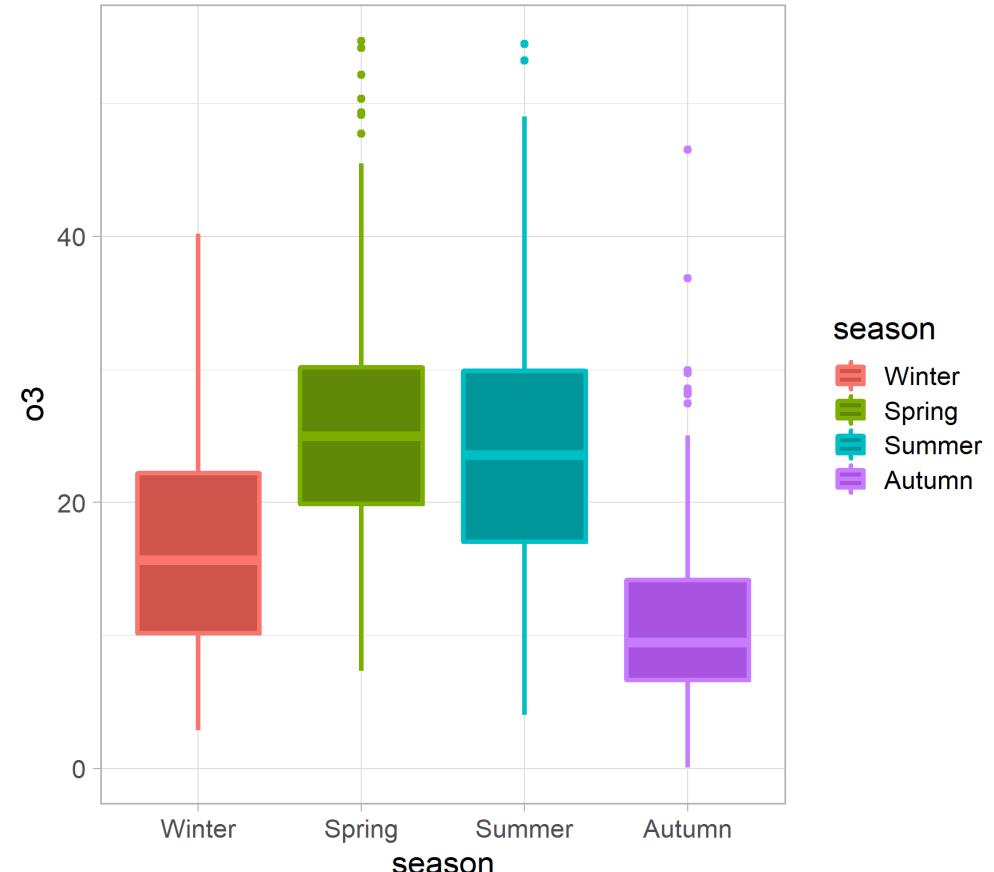
after_scale()

Since ggplot 3.0.0 it is possible to modify aesthetics after the data has been transformed and mapped:

```
library(colorspace)

ggplot(chic, aes(season, o3)) +
  geom_boxplot(
    aes(
      color = season,
      fill = after_scale(darken(color, .2),
      ),
      size = 1.2
    )
  )
```

The `{colorspace}` package provides useful functions for `after_scale`—namely `darken()`, `lighten()`, and `desaturate()`.

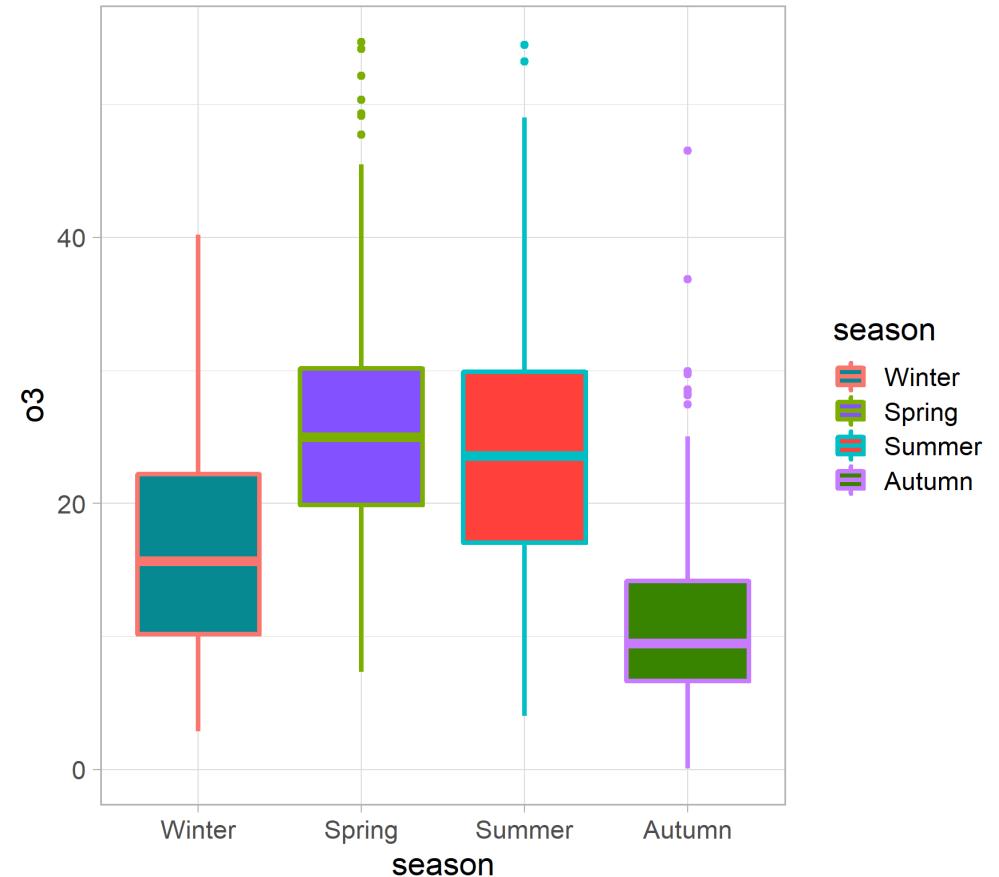


after_scale()

Since ggplot 3.0.0 it is possible to modify aesthetics after the data has been transformed and mapped:

```
library(ggdark)

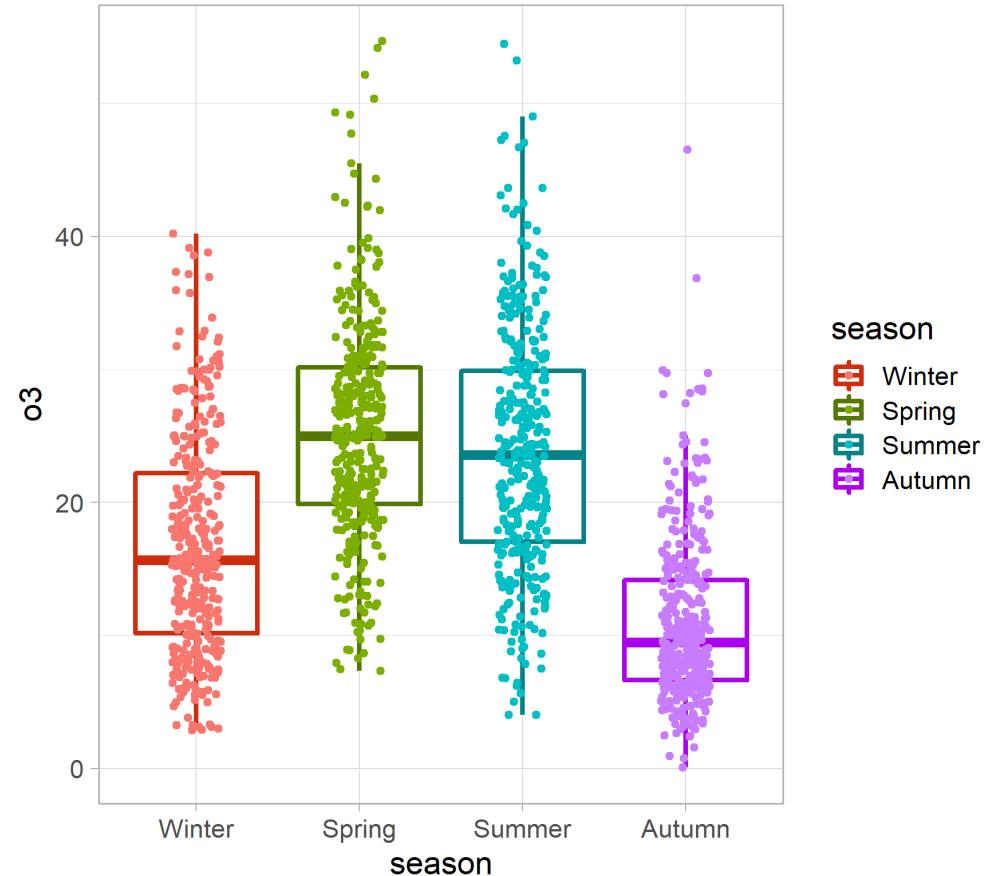
ggplot(chic, aes(season, o3)) +
  geom_boxplot(
    aes(
      color = season,
      fill = after_scale(invert_color(color)),
      size = 1.2
    )
  )
```



after_scale()

Note that one can also overwrite the same aesthetic (note that you have to put the aesthetic inside the geom):

```
ggplot(chic, aes(season, o3)) +  
  geom_boxplot(  
    aes(  
      color = season,  
      color = after_scale(darken(color, .1),  
      ),  
      size = 1.2,  
      outlier.shape = NA  
    ) +  
  geom_jitter(  
    aes(color = season),  
    width = .15  
  )
```



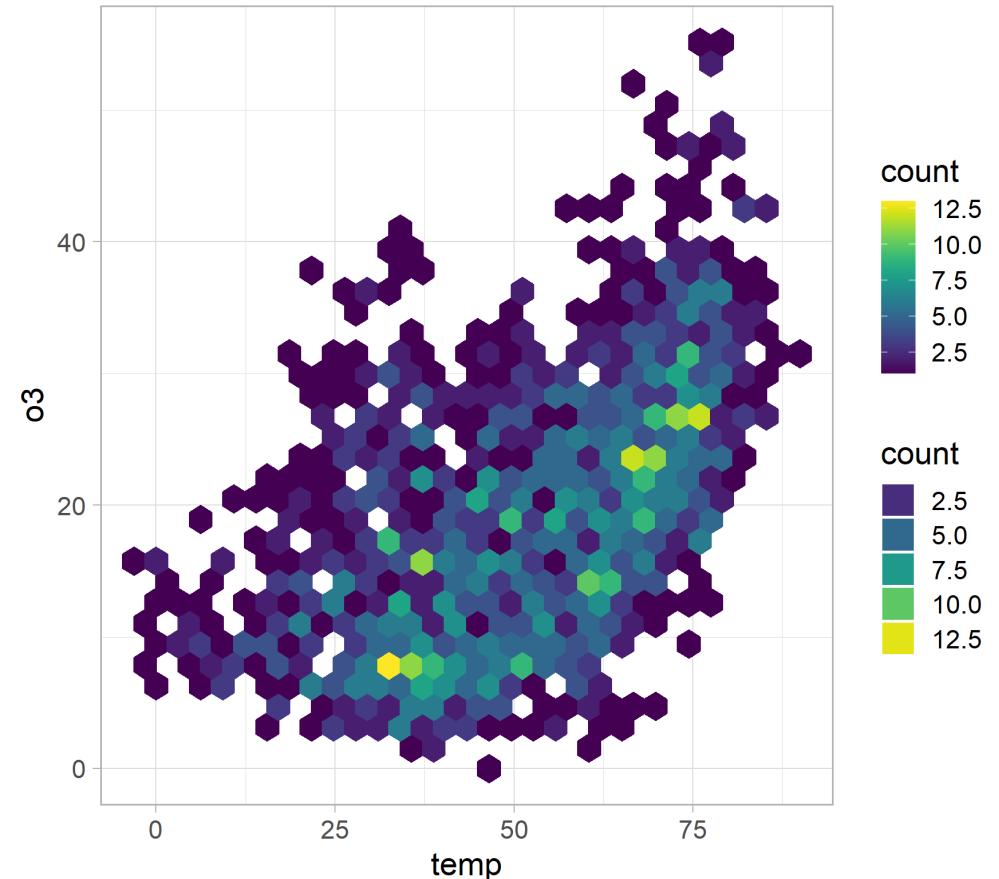
Scales

Control Legends with `guides()`

scale_*(guide)

You already know the `guide` argument of the `scale_*`() function to control legends:

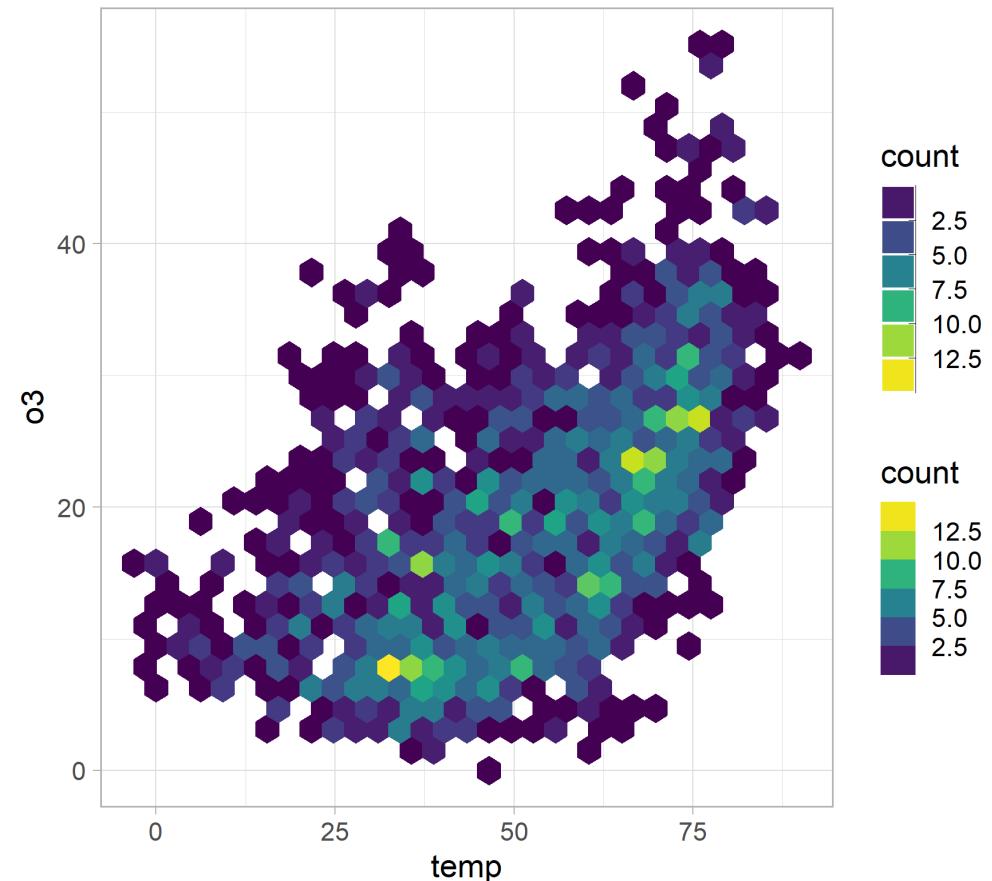
```
ggplot(chic, aes(temp, o3)) +  
  geom_hex(aes(color = ..count..)) +  
  viridis::scale_color_viridis(  
    guide = "colorbar"  
) +  
  viridis::scale_fill_viridis(  
    guide = "legend"  
)
```



scale_*(guide)

You already know the `guide` argument of the `scale_*`() function to control legends:

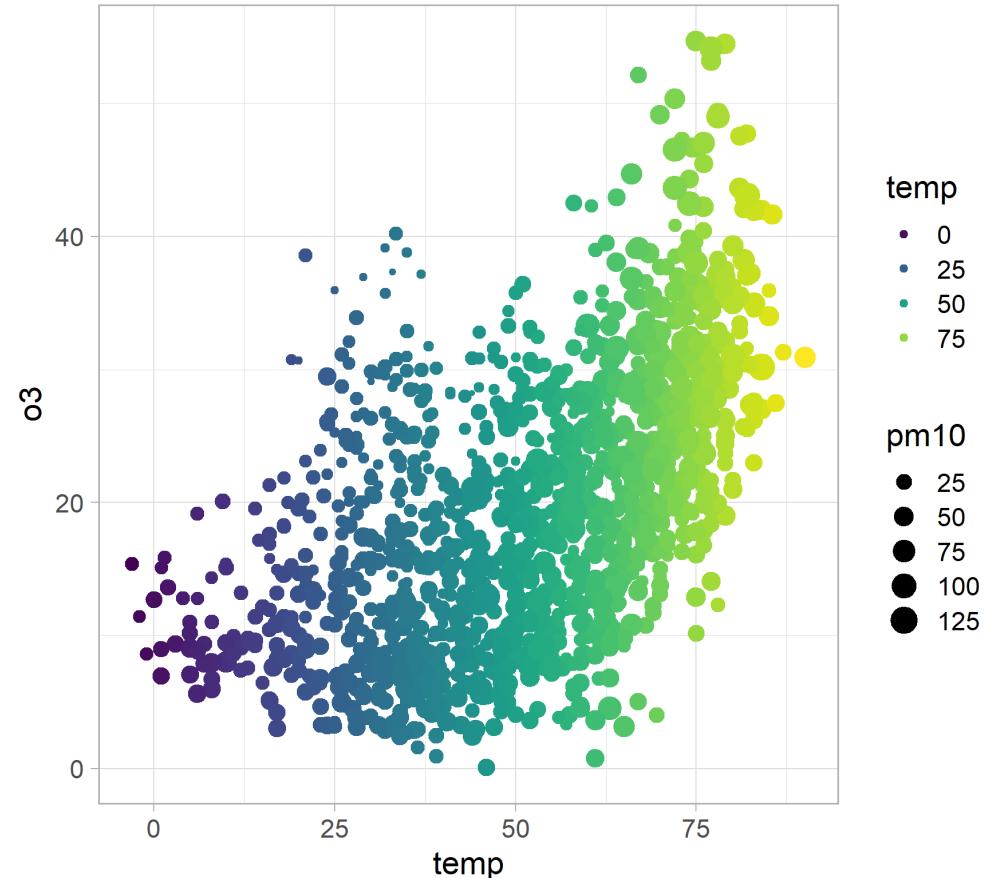
```
ggplot(chic, aes(temp, o3)) +  
  geom_hex(aes(color = ..count..)) +  
  viridis::scale_color_viridis(  
    guide = "colorsteps"  
) +  
  viridis::scale_fill_viridis(  
    guide = "bins"  
)
```



guides()

The `guides()` function allows you to do the same:

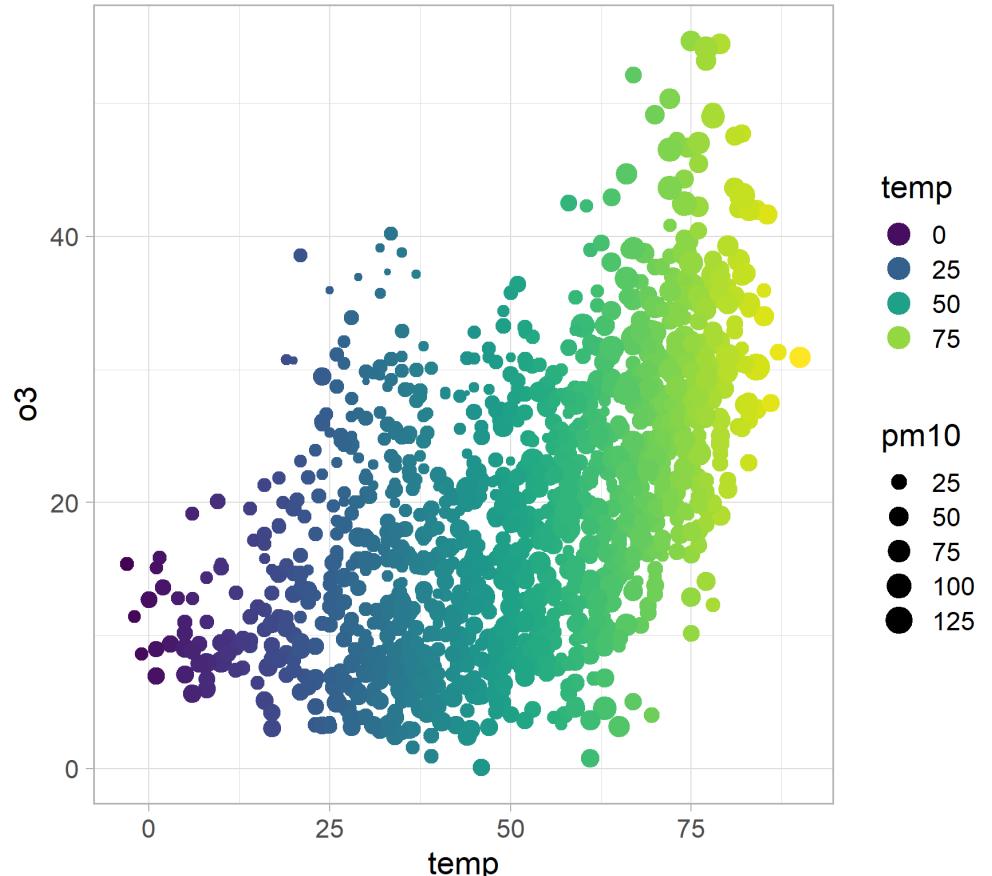
```
ggplot(chic, aes(temp, o3)) +  
  geom_point(  
    aes(  
      color = temp,  
      size = pm10  
    )  
  ) +  
  viridis::scale_color_viridis() +  
  guides(  
    color = "legend"  
  )
```



guides()

The `guides()` function allows for even more control of the legends such as aesthetics:

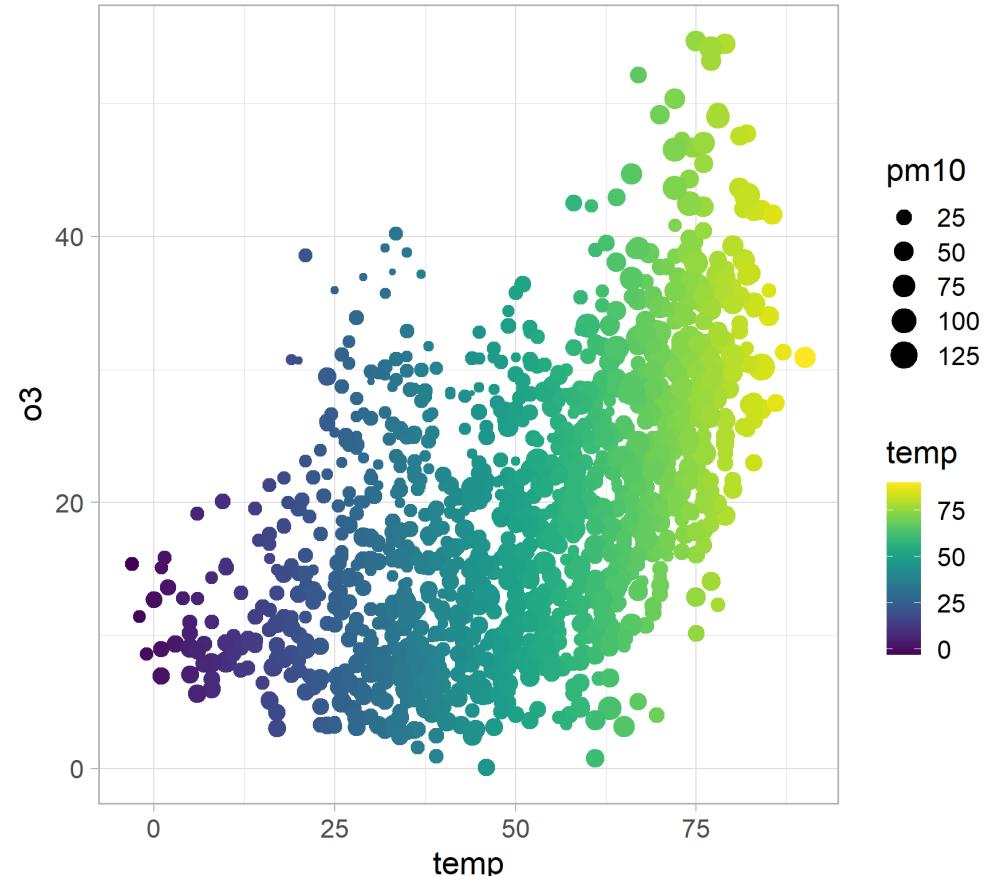
```
ggplot(chic, aes(temp, o3)) +  
  geom_point(  
    aes(  
      color = temp,  
      size = pm10  
    )  
  ) +  
  viridis::scale_color_viridis() +  
  guides(  
    color = guide_legend(  
      override.aes = list(size = 5)  
    )  
  )
```



guides()

The `guides()` function allows for even more control of the legends such as the order:

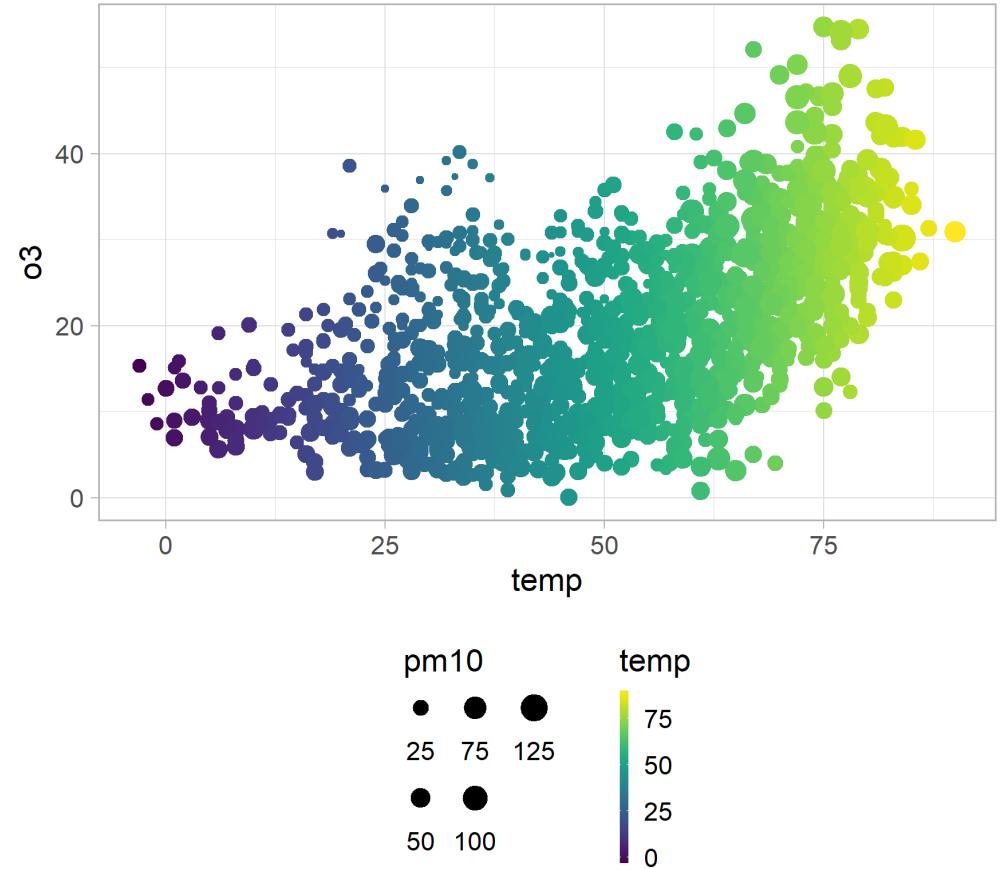
```
ggplot(chic, aes(temp, o3)) +  
  geom_point(  
    aes(  
      color = temp,  
      size = pm10  
    )  
  ) +  
  viridis::scale_color_viridis() +  
  guides(  
    size = guide_legend(order = 1),  
    color = guide_colorbar(order = 2)  
  )
```



guides()

The `guides()` function allows for even more control of the legends any many more details:

```
ggplot(chic, aes(temp, o3)) +  
  geom_point(  
    aes(color = temp, size = pm10)  
  ) +  
  viridis::scale_color_viridis() +  
  guides(  
    color = guide_colorbar(  
      direction = "vertical",  
      barwidth = unit(0.3, "lines")  
    ),  
    size = guide_legend(  
      nrow = 2,  
      bins = 6,  
      title.position = "top",  
      label.position = "bottom"  
    )  
  ) +  
  theme(legend.position = "bottom")
```

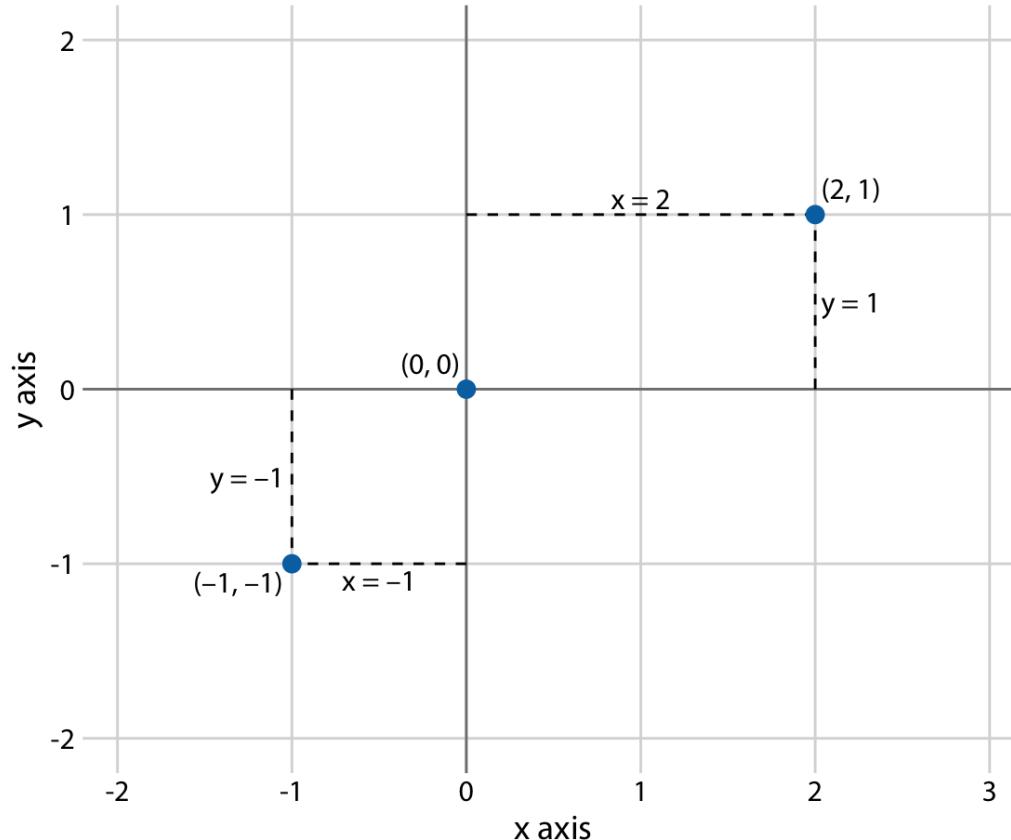


Coordinate System

`coord_*`(`)`

coord_*

Coordinate systems produce a 2d position based on the relative geometric arrangement of the two position aesthetics (usually `x` and `y`).



Source: "Fundamentals of Data Visualization" by Claus Wilke

`coord_*`()

Coordinate systems produce a 2d position based on the relative geometric arrangement of the two position aesthetics (usually `x` and `y`).

The meaning of the position aesthetics depends on the coordinate system used:

- Linear coordinate systems that preserve the shape of geoms.
- Non-linear coordinate systems that likely change the shapes.

`coord_*`()

Coordinate systems produce a 2d position based on the relative geometric arrangement of the two position aesthetics (usually `x` and `y`).

The meaning of the position aesthetics depends on the coordinate system used:

- Linear coordinate systems that preserve the shape of geoms:
 - `coord_cartesian()` and `coord_fixed()`:
the default with two fixed perpendicular oriented axes
 - `coord_flip()`:
a Cartesian coordinate system with flipped axes
 - `coord_fixed()`:
a Cartesian coordinate system with a fixed aspect ratio

`coord_*`()

Coordinate systems produce a 2d position based on the relative geometric arrangement of the two position aesthetics (usually `x` and `y`).

The meaning of the position aesthetics depends on the coordinate system used:

- Linear coordinate systems that preserve the shape of geoms:
 - `coord_cartesian()` and `coord_fixed()`:
the default with two fixed perpendicular oriented axes
 - `coord_flip()`:
a Cartesian coordinate system with flipped axes
 - `coord_fixed()`:
a Cartesian coordinate system with a fixed aspect ratio
- Non-linear coordinate systems that likely change the shapes:
 - `coord_map()` and `coord_sf()`:
map projections
 - `coord_polar()`:
a polar coordinate system
 - `coord_trans()`:
arbitrary transformations to x and y positions

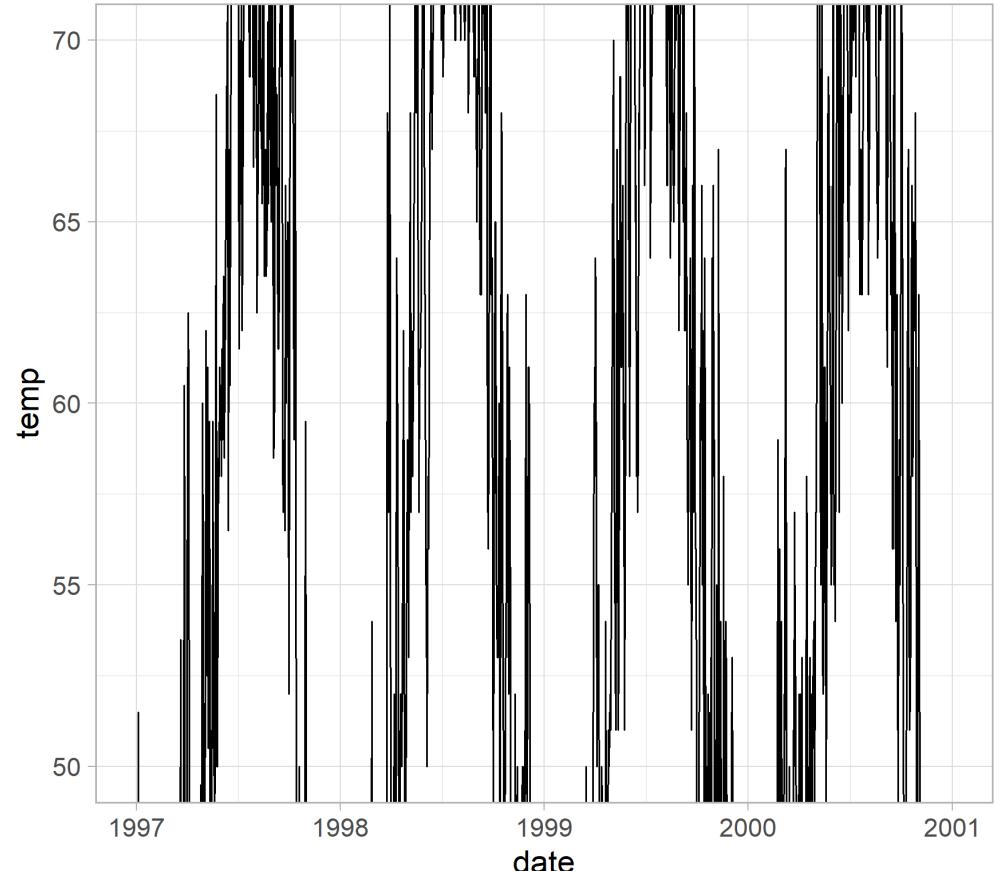
Coordinate System

`coord_cartesian()`

coord_cartesian()

coord_*() functions allow you to zoom in a plot:

```
ggplot(chic, aes(date, temp)) +  
  geom_line() +  
  coord_cartesian(  
    ylim = c(50, 70)  
  )
```

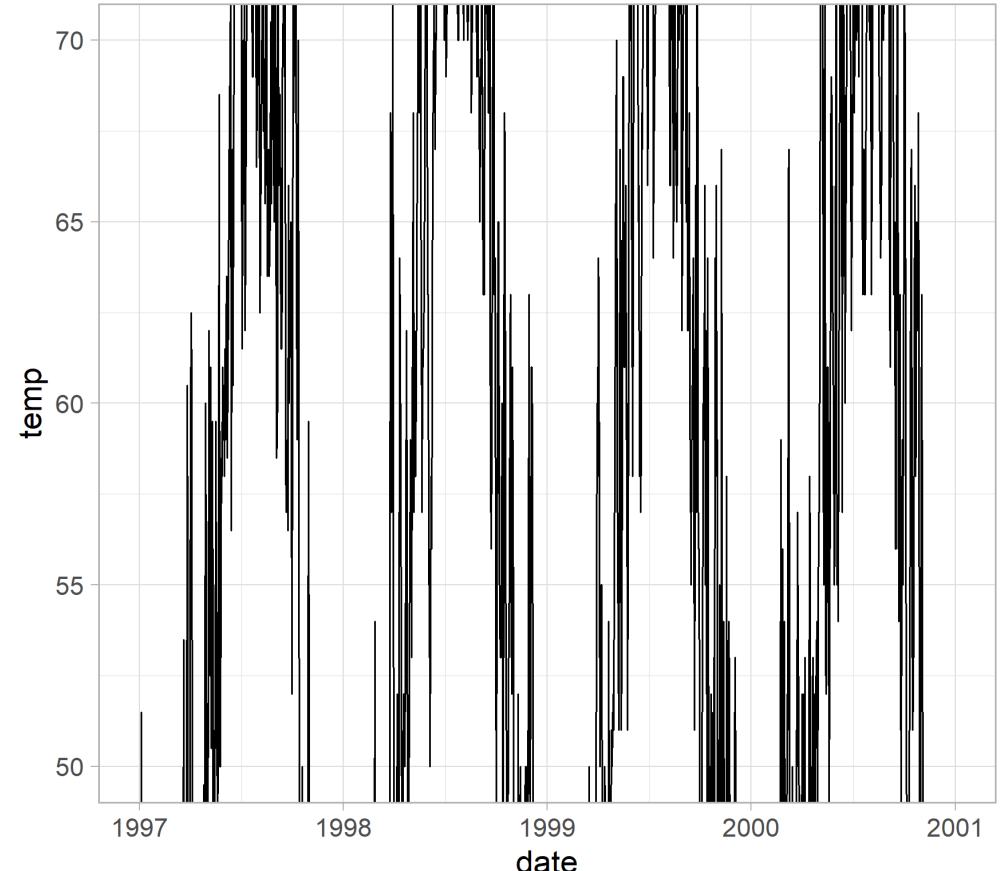


coord_cartesian()

coord_*() functions allow you to zoom in a plot:

```
ggplot(chic, aes(date, temp)) +  
  geom_line() +  
  coord_cartesian(  
    ylim = c(50, 70)  
  )
```

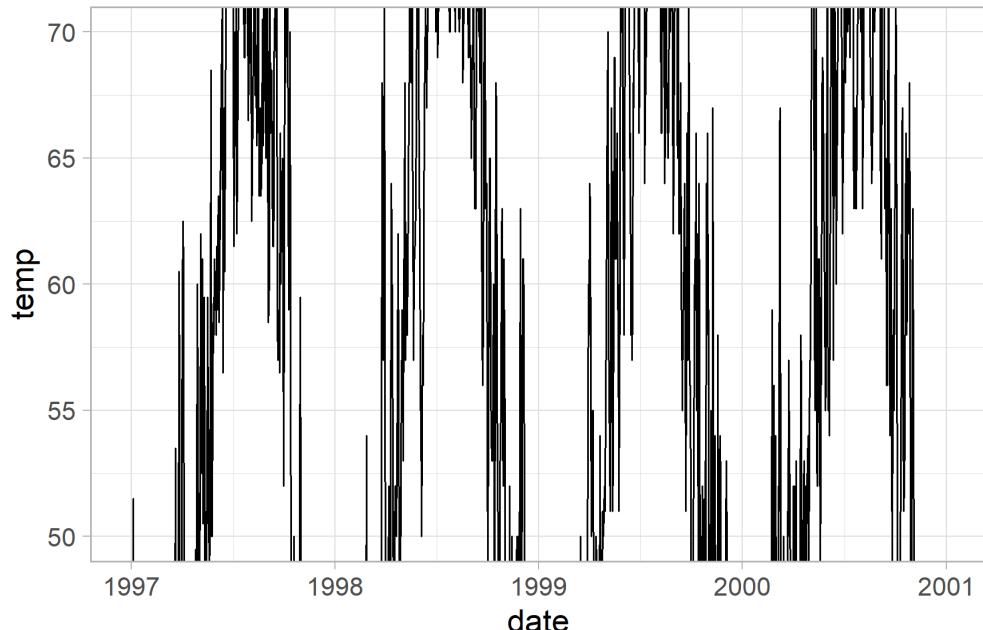
Note that data points outside the plot area are not removed!



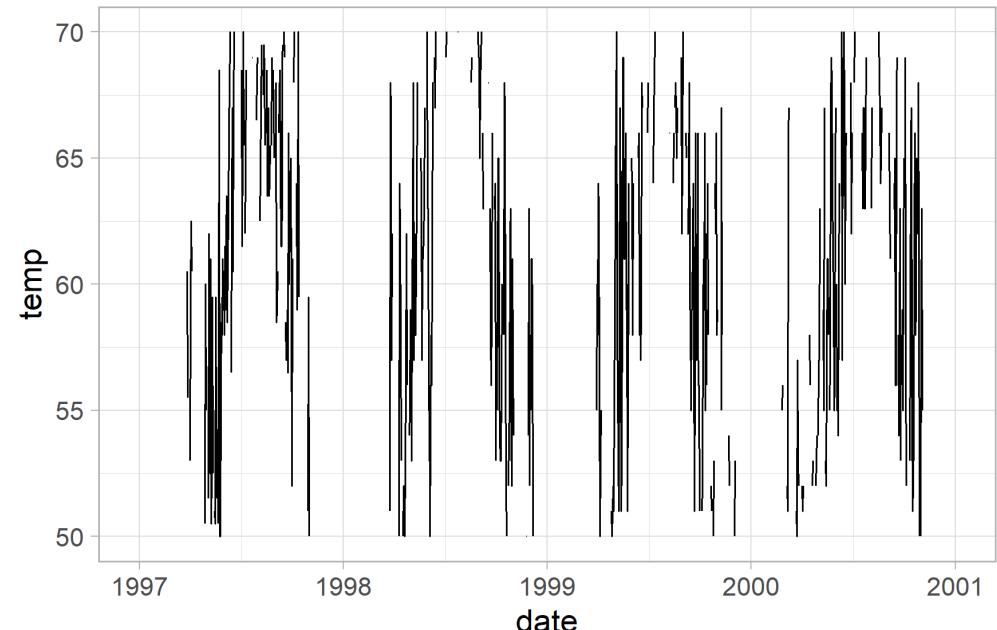
coord_cartesian()

In case you want to remove those data points, use `scale_y_continuous(limits = c(min, max))`.

```
ggplot(chic, aes(date, temp)) +  
  geom_line() +  
  coord_cartesian(ylim = c(50, 70))
```



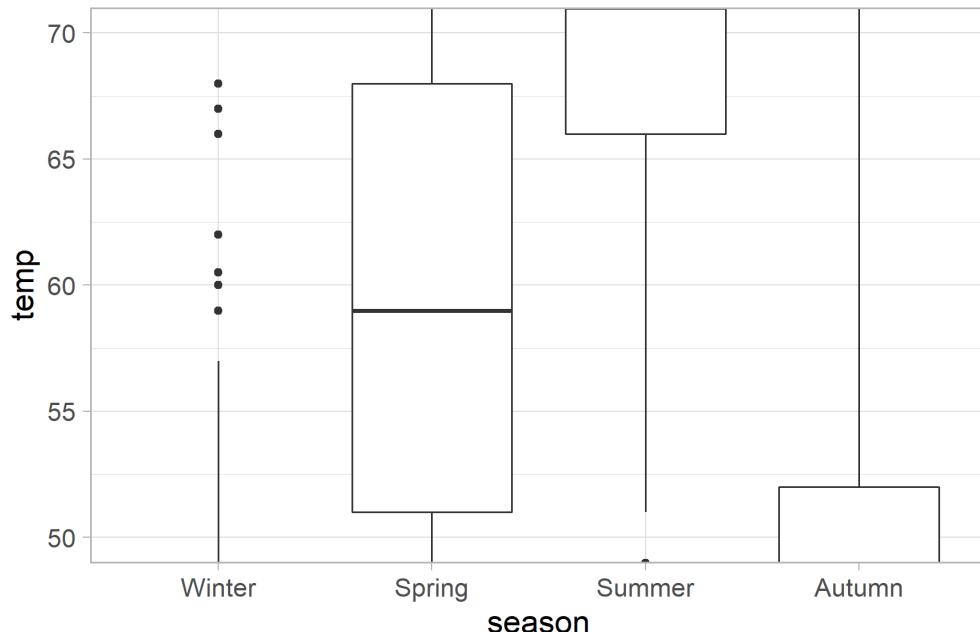
```
ggplot(chic, aes(date, temp)) +  
  geom_line() +  
  scale_y_continuous(limits = c(50, 70))
```



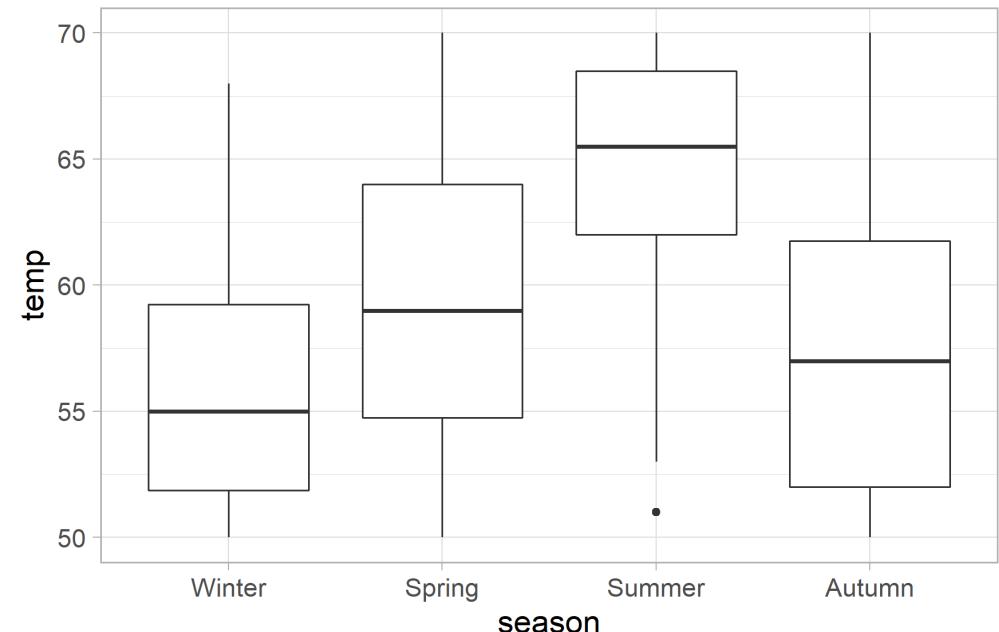
coord_cartesian()

In case you want to remove those data points, use `scale_y_continuous(limits = c(min, max))`.

```
ggplot(chic, aes(season, temp)) +  
  geom_boxplot() +  
  coord_cartesian(ylim = c(50, 70))
```



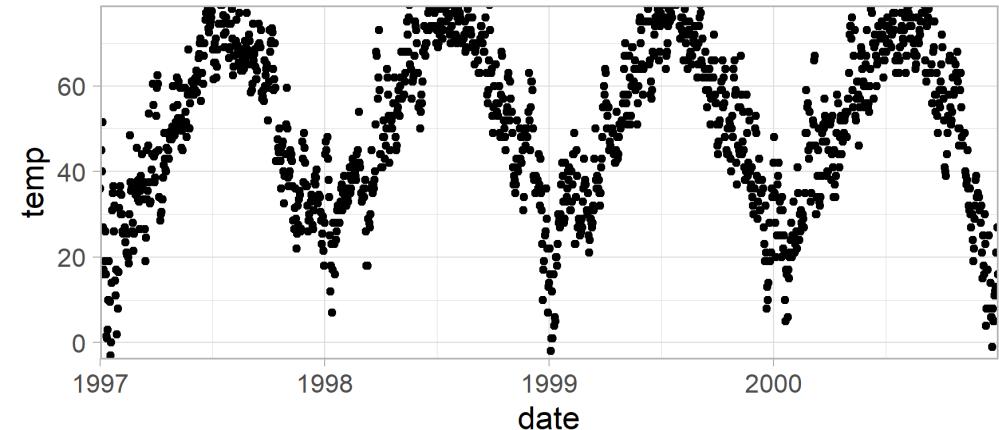
```
ggplot(chic, aes(season, temp)) +  
  geom_boxplot() +  
  scale_y_continuous(limits = c(50, 70))
```



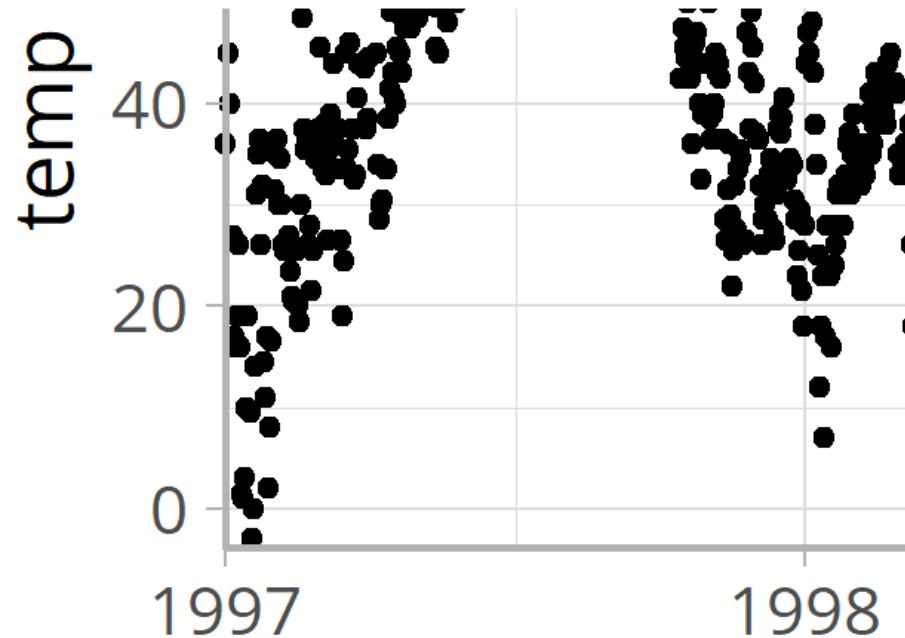
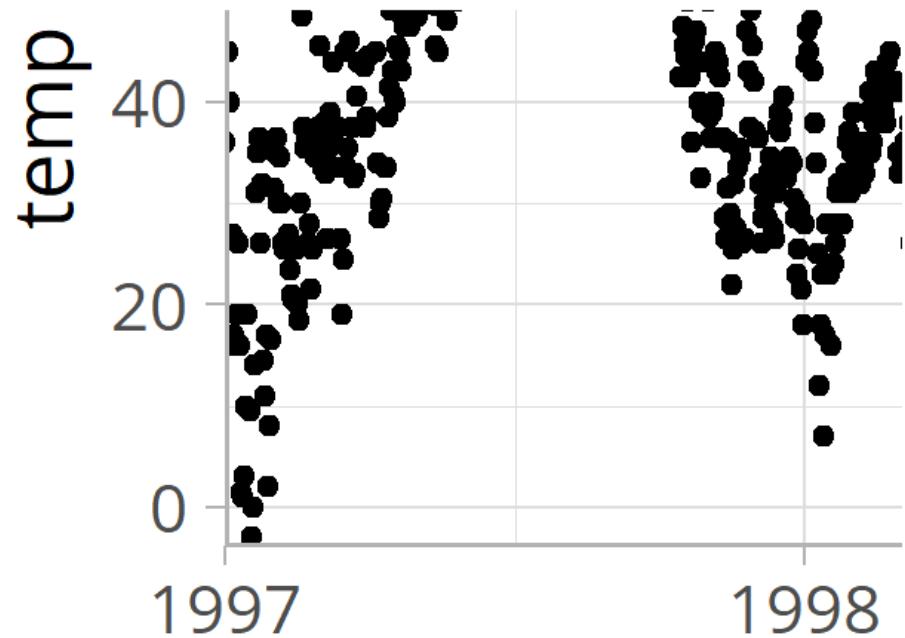
coord_cartesian()

coord_*() is also used to *prevent clipping* of points and the correct alignment of axis ticks:

```
ggplot(chic, aes(date, temp)) +  
  geom_point() +  
  coord_cartesian(  
    ylim = c(0, 75)  
  )
```



Clipping

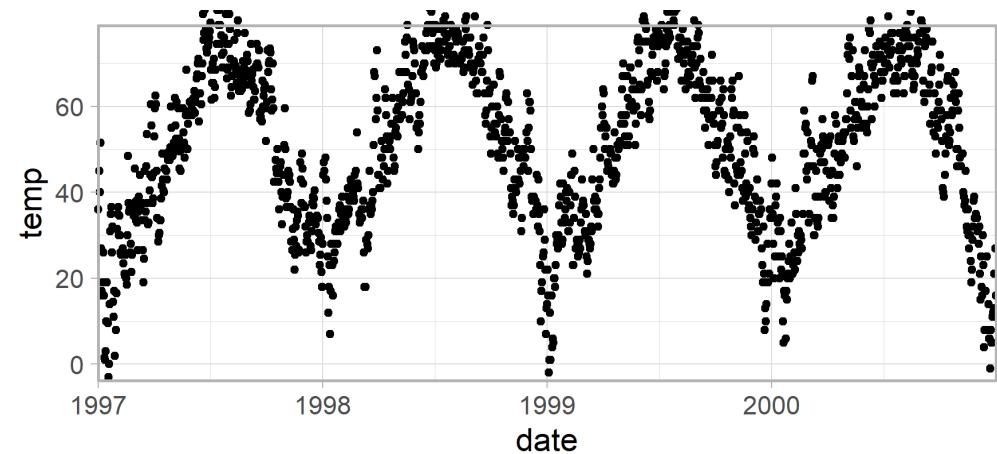
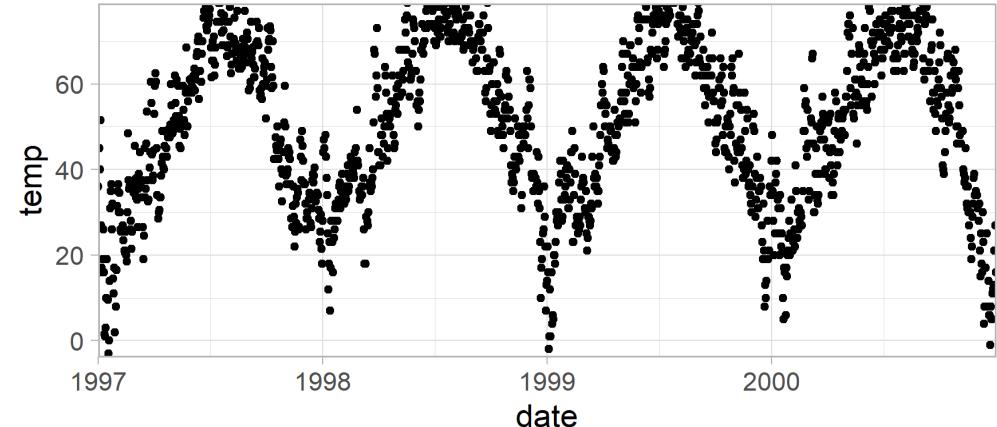


coord_cartesian()

coord_*() is also used to *prevent clipping* of points and correct alignment of axis ticks:

```
ggplot(chic, aes(date, temp)) +  
  geom_point() +  
  coord_cartesian(  
    ylim = c(0, 75)  
)
```

```
ggplot(chic, aes(date, temp)) +  
  geom_point() +  
  coord_cartesian(  
    ylim = c(0, 75),  
    clip = "off"  
)
```



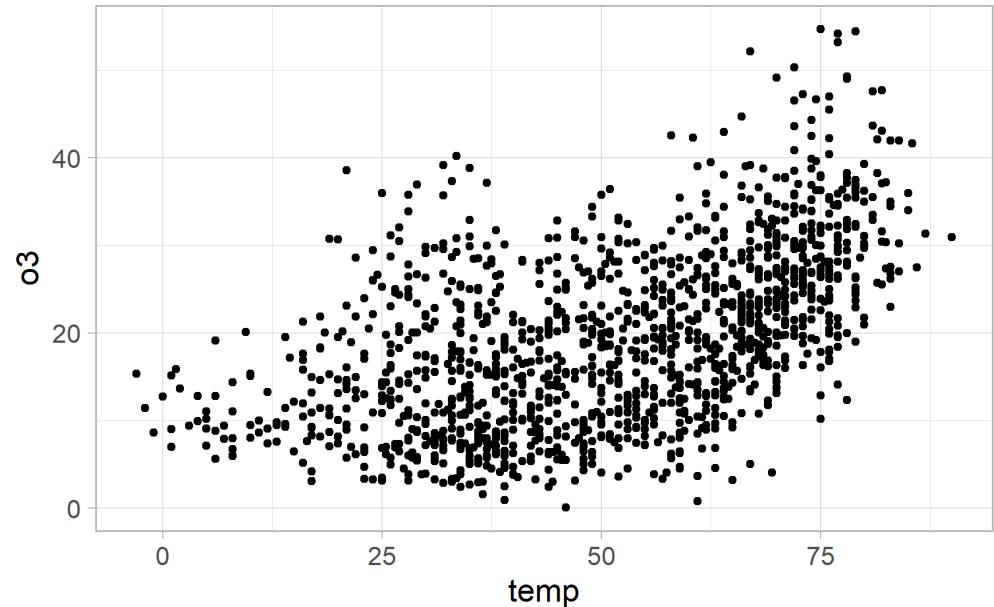
Coordinate System

`coord_fixed()`

coord_fixed()

`coord_fixed()` forces a specified ratio as physical representation of units on the axes:

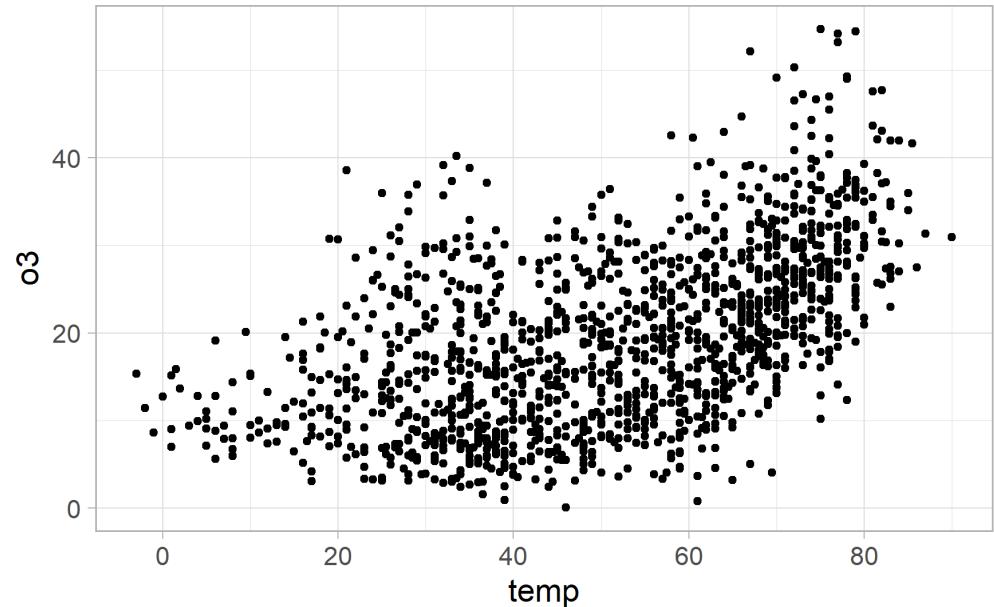
```
ggplot(chic, aes(temp, o3)) +  
  geom_point() +  
  coord_fixed()
```



coord_fixed()

`coord_fixed()` forces a specified ratio as physical representation of units on the axes:

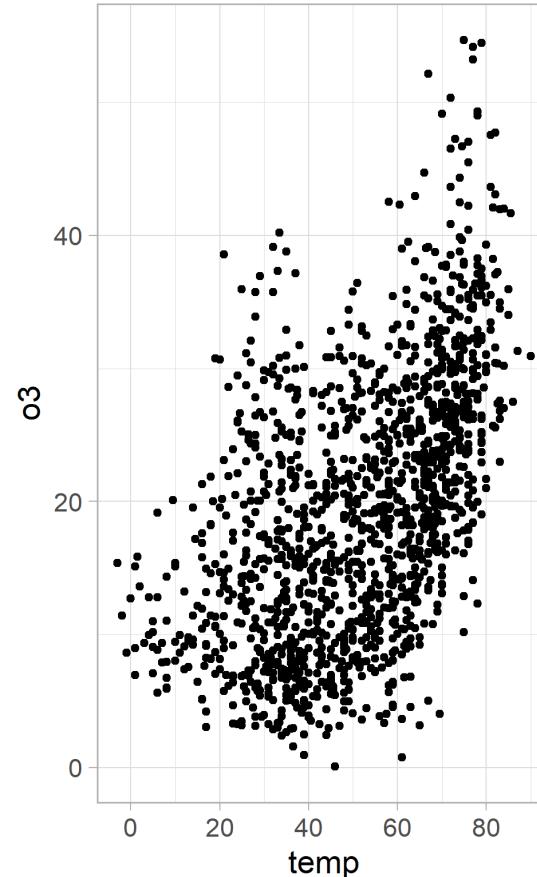
```
ggplot(chic, aes(temp, o3)) +  
  geom_point() +  
  coord_fixed() +  
  scale_x_continuous(  
    breaks = seq(0, 100, by = 20)  
)
```



coord_fixed()

`coord_fixed()` forces a specified ratio as physical representation of units on the axes:

```
ggplot(chic, aes(temp, o3)) +  
  geom_point() +  
  coord_fixed(  
    ratio = 3  
) +  
  scale_x_continuous(  
    breaks = seq(0, 100, by = 20)  
)
```



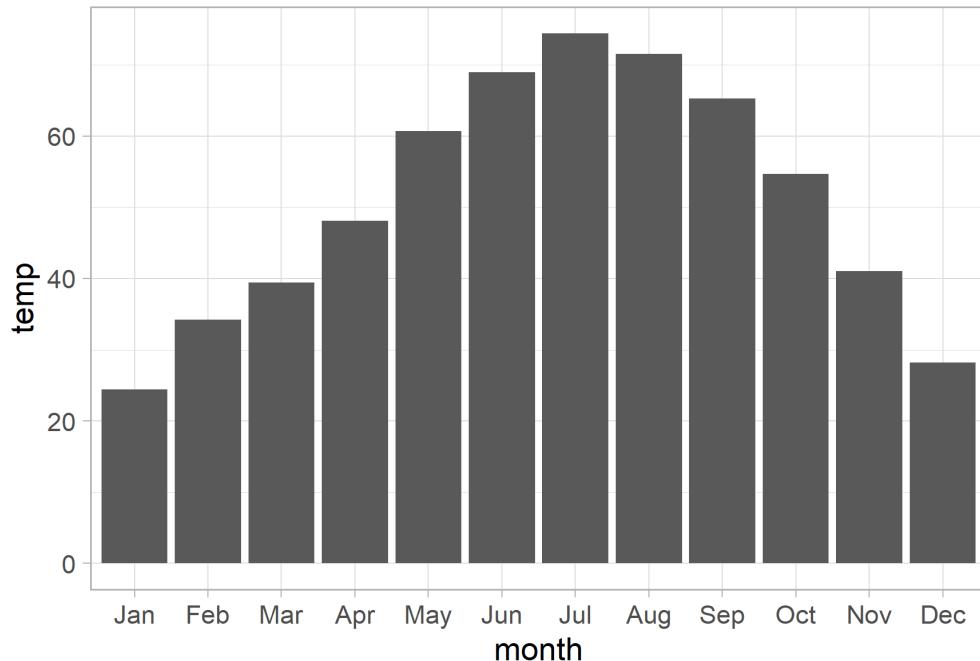
Coordinate System

`coord_flip()`

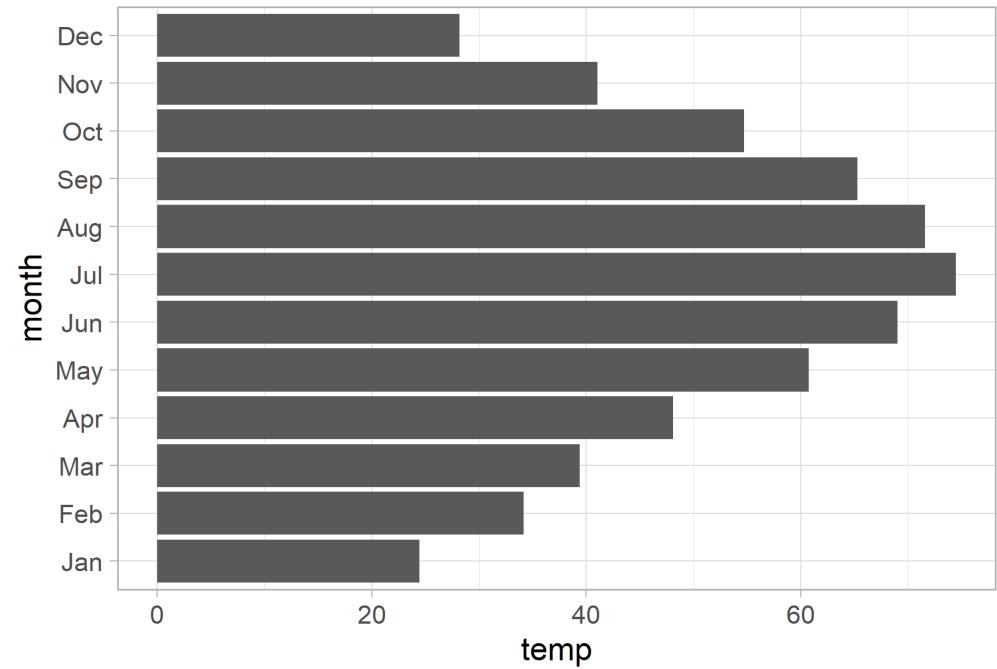
coord_flip()

`coord_flip()` allows you to flip a Cartesian coordinate system:

```
ggplot(chic_month, aes(month, temp)) +  
  geom_col() +  
  coord_cartesian()
```



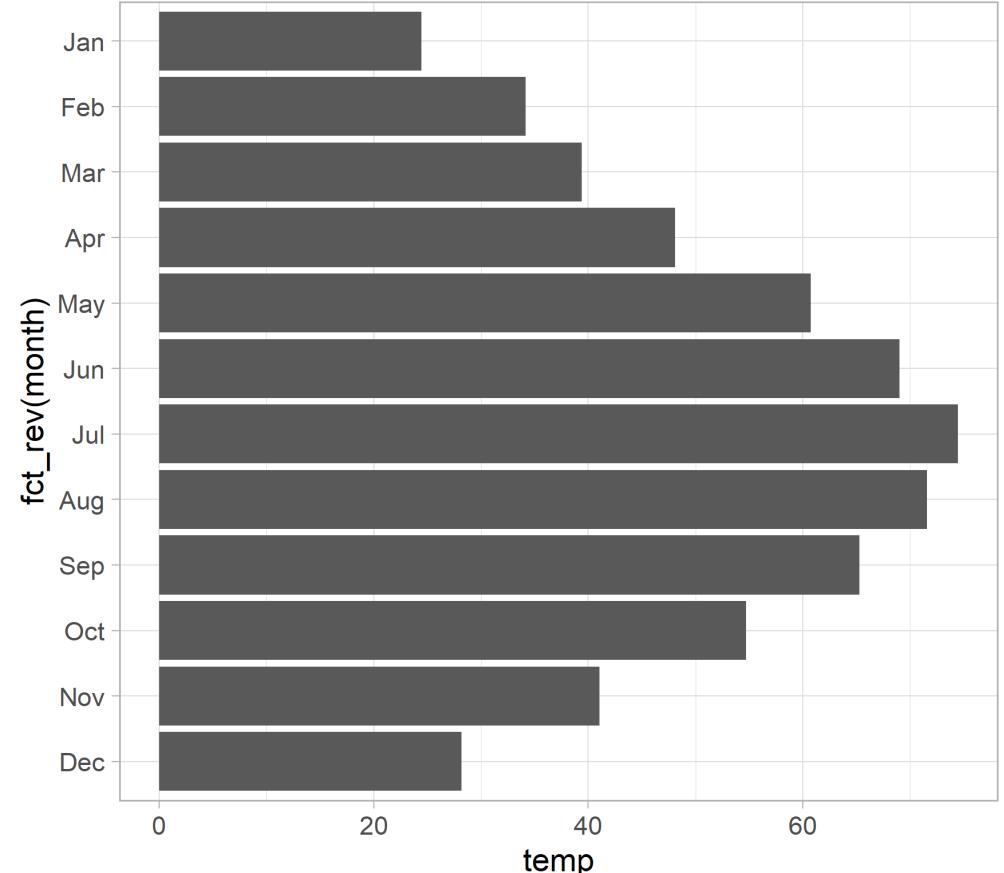
```
ggplot(chic_month, aes(month, temp)) +  
  geom_col() +  
  coord_flip()
```



coord_flip()

`coord_flip()` allows you to flip a Cartesian coordinate system:

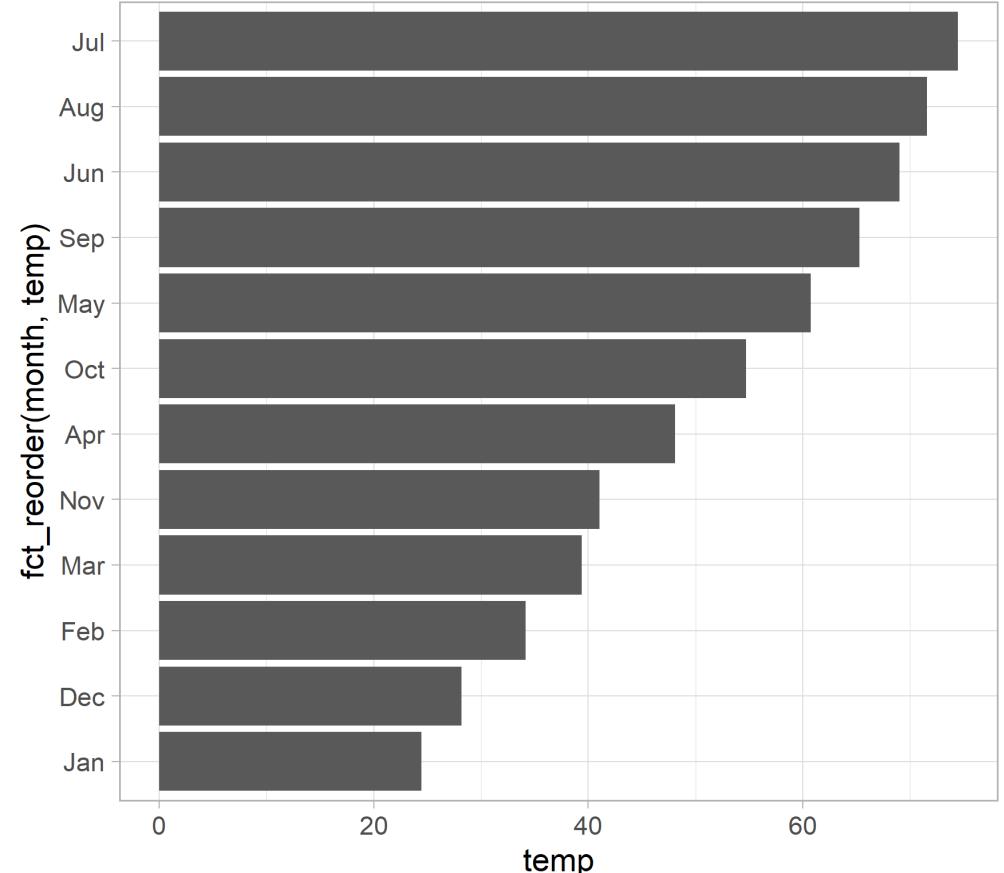
```
ggplot(  
  chic_month,  
  aes(  
    fct_rev(month),  
    temp  
  )) +  
  geom_col() +  
  coord_flip()
```



coord_flip()

`coord_flip()` allows you to flip a Cartesian coordinate system:

```
ggplot(  
  chic_month,  
  aes(  
    fct_reorder(month, temp),  
    temp  
  )) +  
  geom_col() +  
  coord_flip()
```



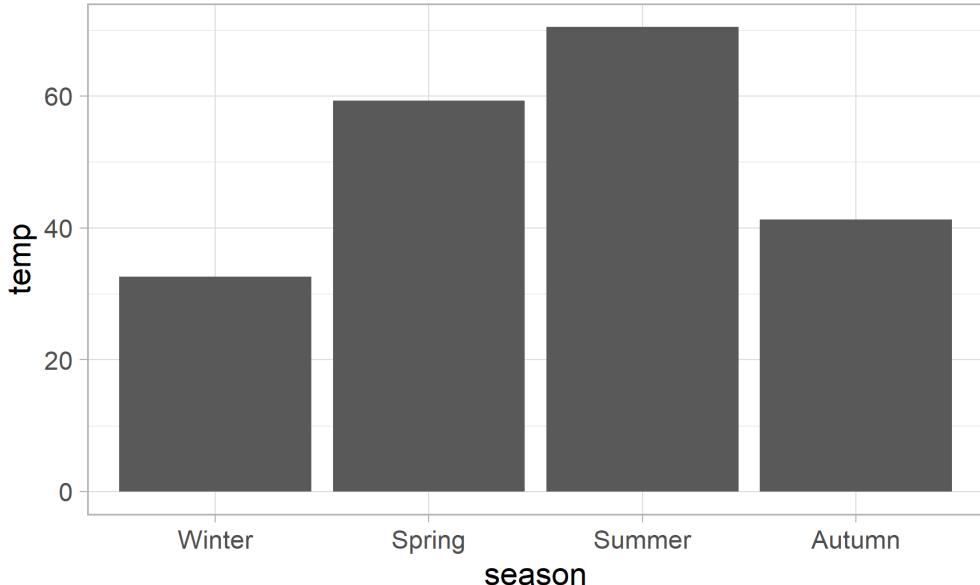
Coordinate System

`coord_polar()`

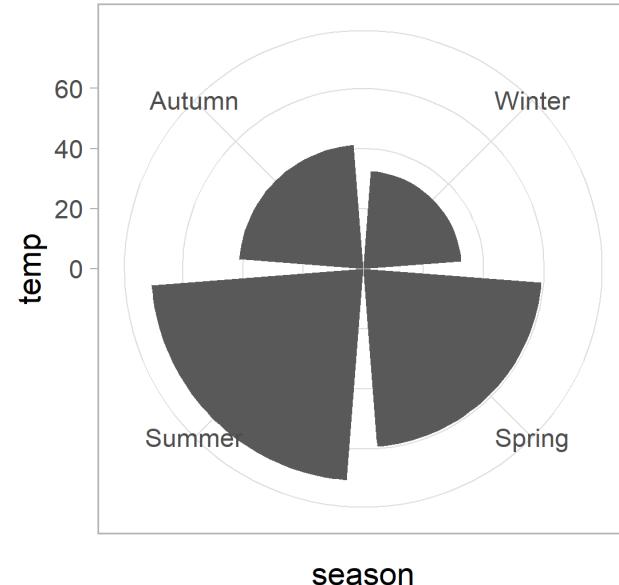
coord_polar()

You can easily transform a rectangular coordinate system into a polar one with `coord_polar()`:

```
ggplot(chic, aes(season, temp)) +  
  stat_summary(fun.y = mean,  
              geom = "col") +  
  coord_cartesian()
```



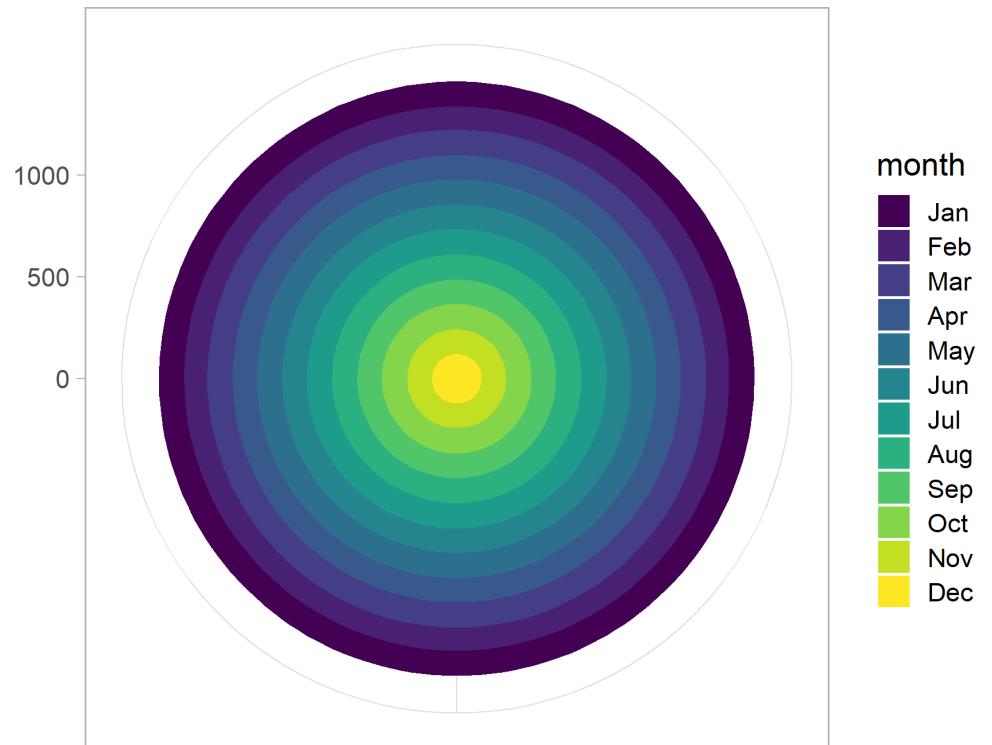
```
ggplot(chic, aes(season, temp)) +  
  stat_summary(fun.y = mean,  
              geom = "col") +  
  coord_polar()
```



coord_polar()

This way, we can create pie charts:

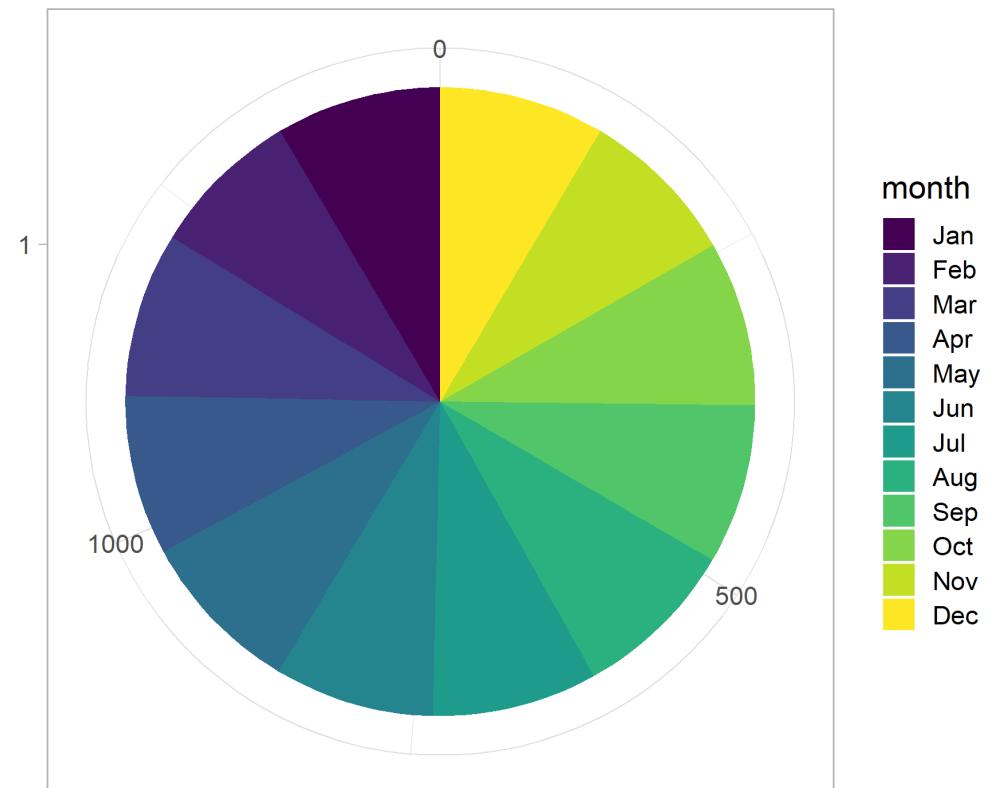
```
ggplot(  
  chic_month,  
  aes(  
    factor(1), n,  
    fill = month  
  )) +  
  geom_col() +  
  coord_polar() +  
  scale_x_discrete(  
    name = NULL,  
    expand = c(0, 0)  
  ) +  
  scale_y_continuous(  
    name = NULL,  
    expand = c(0, 0)  
  )
```



coord_polar()

This way, we can create pie charts:

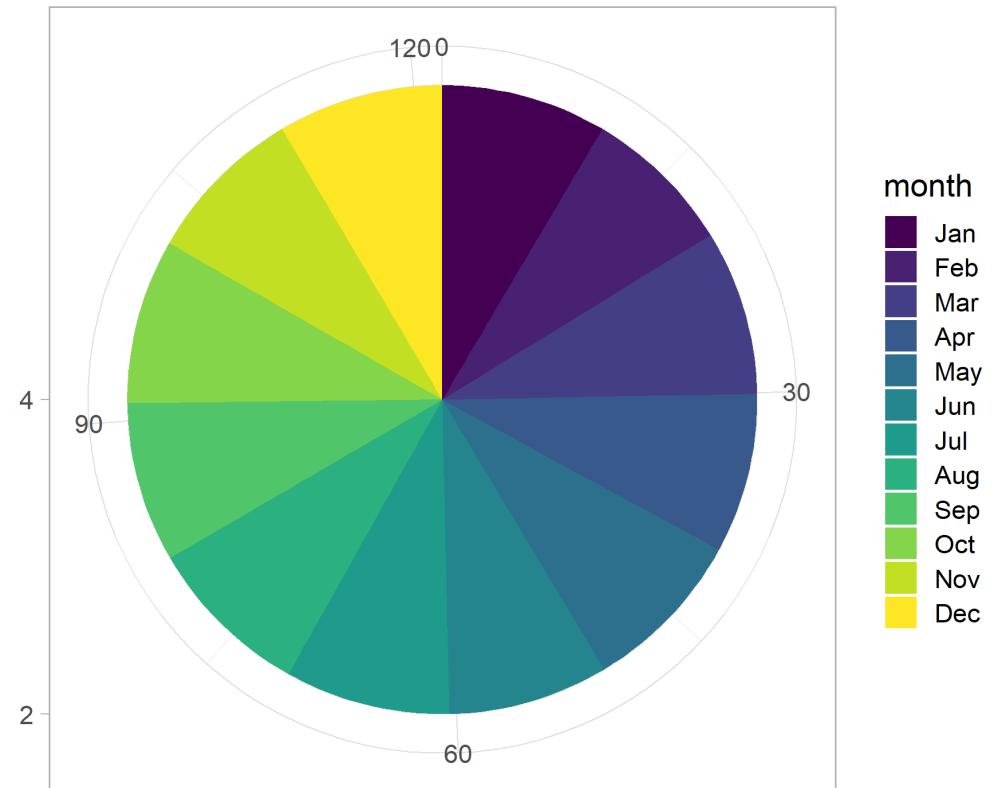
```
ggplot(  
  chic_month,  
  aes(  
    factor(1), n,  
    fill = month  
  )) +  
  geom_col() +  
  coord_polar(theta = "y") +  
  scale_x_discrete(  
    name = NULL,  
    expand = c(0, 0)  
  ) +  
  scale_y_continuous(  
    name = NULL,  
    expand = c(0, 0)  
  )
```



coord_polar()

... and doughnut charts:

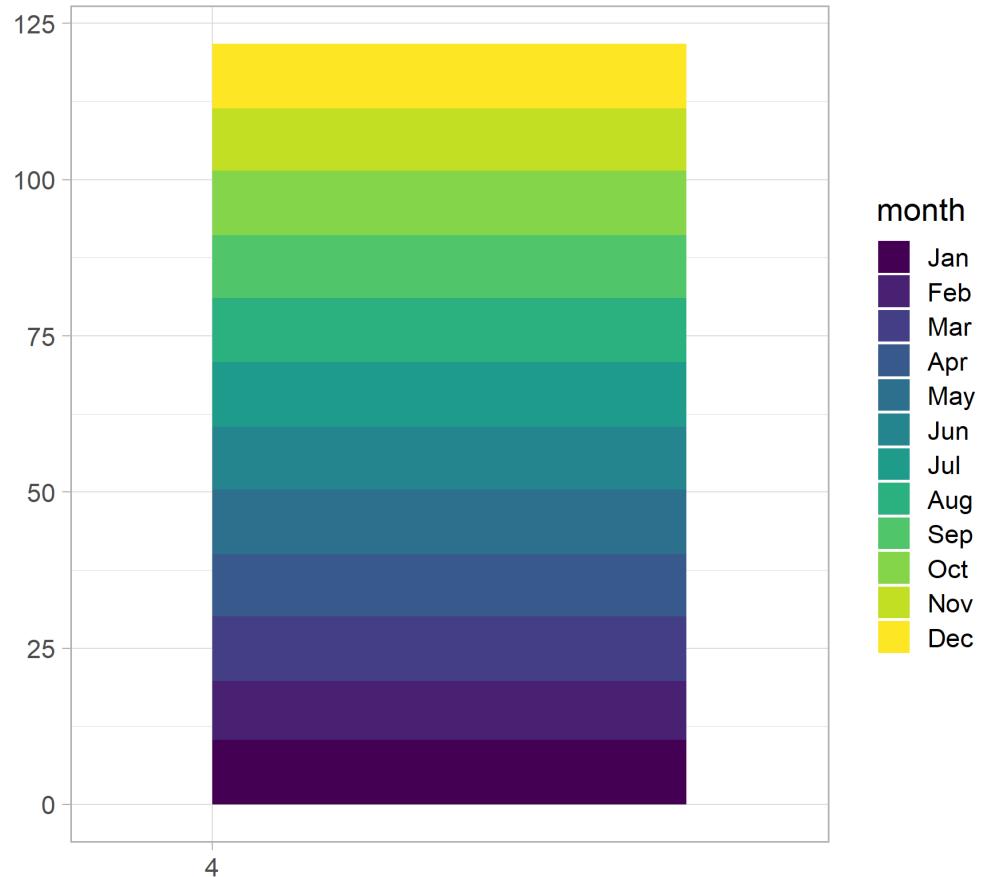
```
chic_month %>%
  mutate(
    perc = n / n(),
    ymax = cumsum(perc),
    ymin = c(0, head(ymax, n = -1)))
) %>%
  ggplot(
    aes(
      xmin = 4, xmax = 6,
      ymin = ymin, ymax = ymax,
      fill = month
    )) +
  geom_rect() +
  coord_polar(theta = "y") +
  scale_x_discrete(
    name = NULL,
    limits = c(2, 4)
)
```



coord_polar()

Our doughnut charts looks like this in a Cartesian coordinate system:

```
chic_month %>%
  mutate(
    perc = n / n(),
    ymax = cumsum(perc),
    ymin = c(0, head(ymax, n = -1)))
) %>%
  ggplot(
    aes(
      xmin = 4, xmax = 6,
      ymin = ymin, ymax = ymax,
      fill = month
    )) +
  geom_rect() +
  #coord_polar(theta = "y") +
  scale_x_discrete(
    name = NULL,
    limits = c(2, 4)
  )
```



Coordinate System

`coord_map()` and `coord_sf()`

`coord_map()` and `coord_sf()`

Maps often come as polygons:

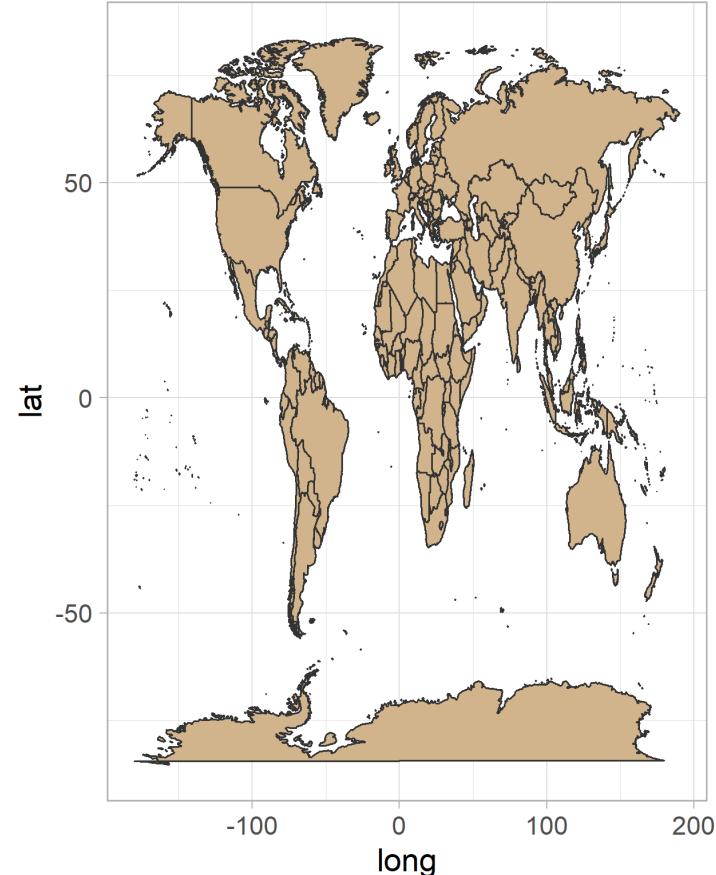
```
world <- map_data("world")

head(world, 15)
##       long      lat group order   region subregion
## 1 -69.89912 12.45200     1     1    Aruba    <NA>
## 2 -69.89571 12.42300     1     2    Aruba    <NA>
## 3 -69.94219 12.43853     1     3    Aruba    <NA>
## 4 -70.00415 12.50049     1     4    Aruba    <NA>
## 5 -70.06612 12.54697     1     5    Aruba    <NA>
## 6 -70.05088 12.59707     1     6    Aruba    <NA>
## 7 -70.03511 12.61411     1     7    Aruba    <NA>
## 8 -69.97314 12.56763     1     8    Aruba    <NA>
## 9 -69.91181 12.48047     1     9    Aruba    <NA>
## 10 -69.89912 12.45200    1    10    Aruba    <NA>
## 12  74.89131 37.23164    2    12 Afghanistan <NA>
## 13  74.84023 37.22505    2    13 Afghanistan <NA>
## 14  74.76738 37.24917    2    14 Afghanistan <NA>
## 15  74.73896 37.28564    2    15 Afghanistan <NA>
## 16  74.72666 37.29072    2    16 Afghanistan <NA>
```

`coord_map()` and `coord_sf()`

Maps often come as polygons and can be plotted via `geom_polygon()`:

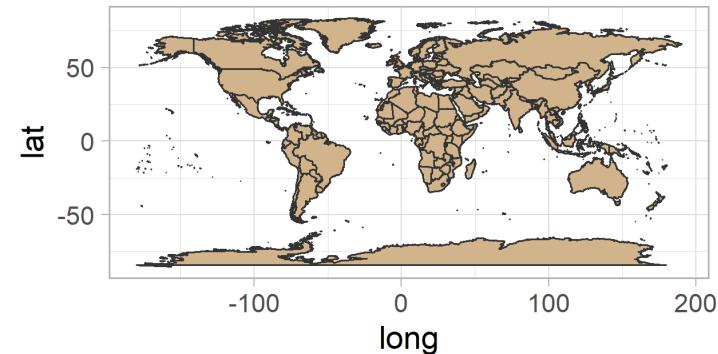
```
(g_world <-
  ggplot(
    world,
    aes(
      long, lat,
      group = group
    )) +
  geom_polygon(
    fill = "tan",
    color = "grey20"
  )
)
```



`coord_map()` and `coord_sf()`

We can fix that by using `coord_fixed()`:

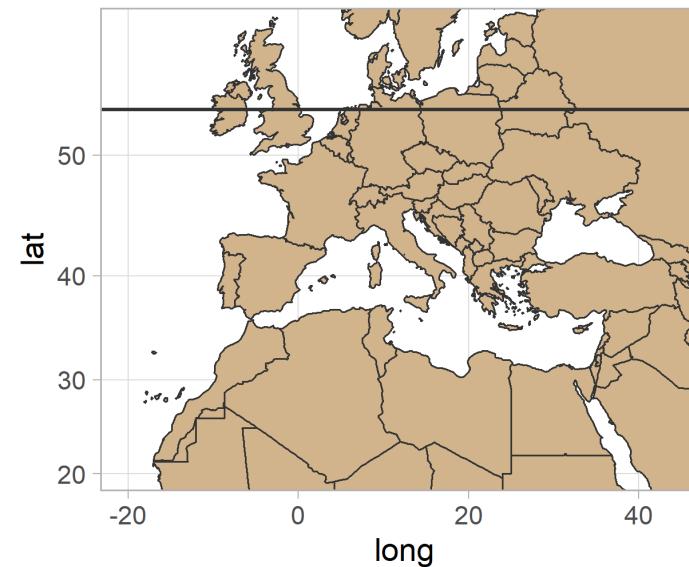
```
g_world +  
  coord_fixed()
```



`coord_map()` and `coord_sf()`

We can fix that by using `coord_fixed()`:

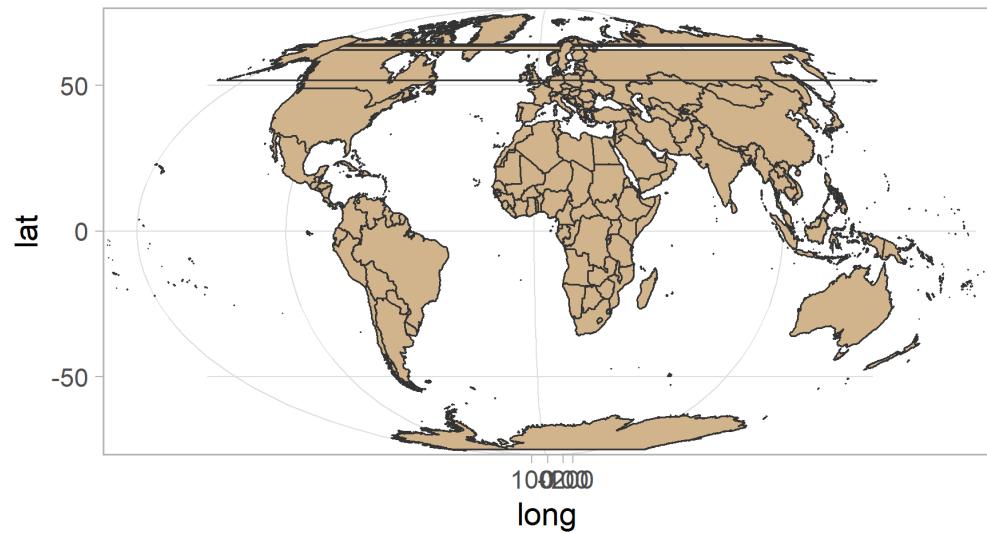
```
g_world +  
  coord_map(  
    xlim = c(-20, 45),  
    ylim = c(20, 58)  
  )
```



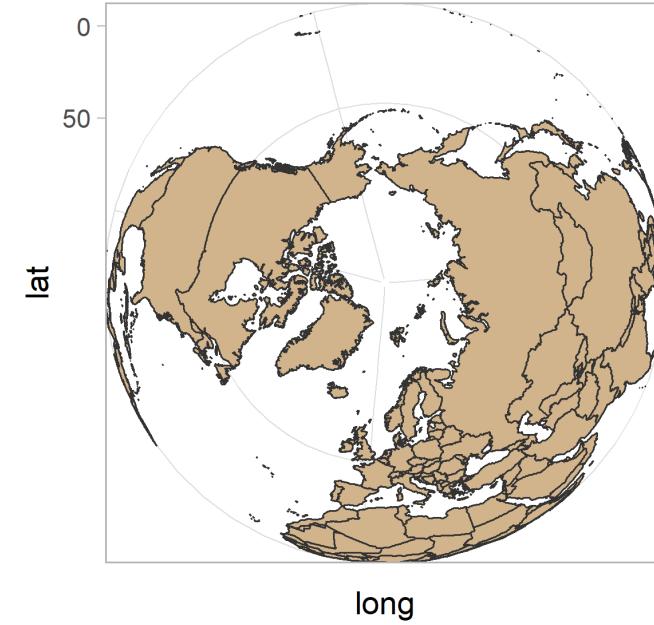
`coord_map()` and `coord_sf()`

Since maps are displaying spherical data, it's better to project the data via `coord_map()`:

```
g_world +  
  coord_map("mollweide")
```



```
g_world +  
  coord_map("ortho")
```



coord_map() and coord_sf()

The **sf** package (**simple features**) is the new standard in geospatial computation:

```
library(sf)

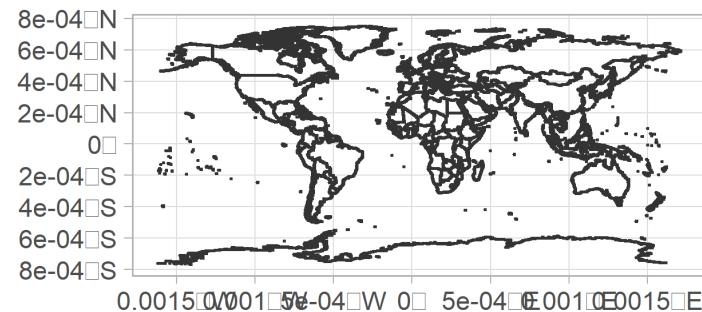
sf_world <- st_as_sf(world, coords = c("long", "lat"), crs = 3786)

head(sf_world)
## Simple feature collection with 6 features and 4 fields
## geometry type: POINT
## dimension: XY
## bbox:           xmin: -70.06612 ymin: 12.423 xmax: -69.89571 ymax: 12.59707
## projected CRS: World Equidistant Cylindrical (Sphere)
##   group order region subregion               geometry
## 1     1      1 Aruba    <NA>  POINT (-69.89912 12.452)
## 2     1      2 Aruba    <NA>  POINT (-69.89571 12.423)
## 3     1      3 Aruba    <NA>  POINT (-69.94219 12.43853)
## 4     1      4 Aruba    <NA>  POINT (-70.00415 12.50049)
## 5     1      5 Aruba    <NA>  POINT (-70.06612 12.54697)
## 6     1      6 Aruba    <NA>  POINT (-70.05088 12.59707)
```

`coord_map()` and `coord_sf()`

The `sf` package is included in `ggplot2` via `geom_sf()` and `coord_sf()`:

```
ggplot(sf_world) +  
  geom_sf(  
    fill = "tan",  
    color = "grey20",  
    size = .3  
)
```

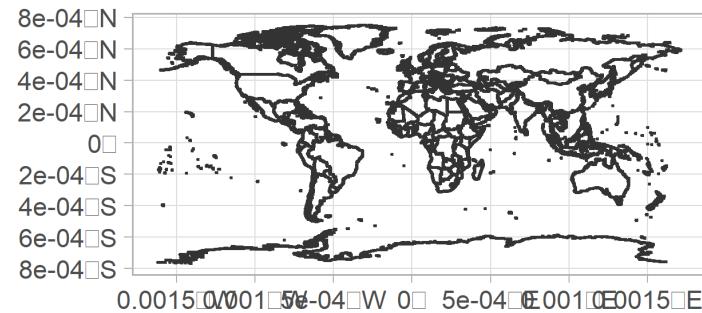


`coord_map()` and `coord_sf()`

The `sf` package is included in `ggplot2` via `geom_sf()` and `coord_sf()`:

```
ggplot(sf_world) +  
  geom_sf(  
    fill = "tan",  
    color = "grey20",  
    size = .3  
)
```

The `fill` argument doesn't work here, since it turned our polygons into points...



`coord_map()` and `coord_sf()`

The `sf` package is included in `ggplot2` via `geom_sf()` and `coord_sf()`:

```
ggplot(sf_world) +  
  geom_sf(  
    fill = "tan",  
    color = "grey20",  
    size = .3  
  ) +  
  coord_sf(  
    xlim = c(-20, 45),  
    ylim = c(20, 58)  
  )
```

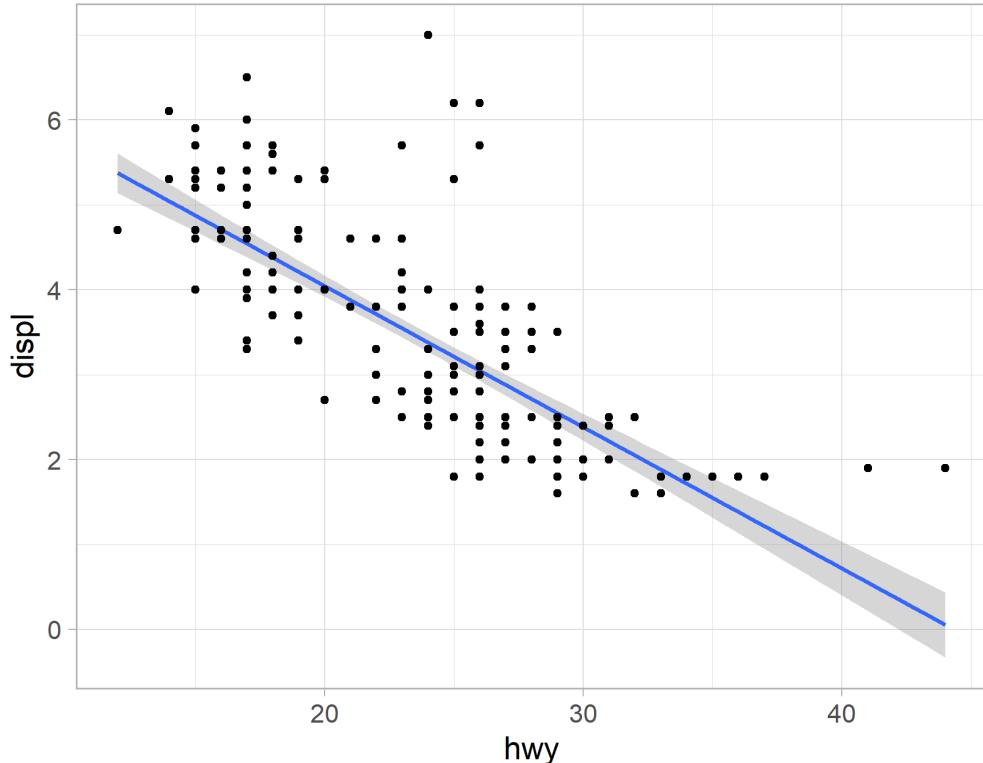


Your Turn!

- For a data set of your choice
 - create a scatter plot with a linear fitting and test how `coord_fixed()` changes the perception of the relationship
 - create a pie or doughnut chart for one variable in that data set

Your Turn: Fixed Aspect Ratio

```
ggplot(mpg, aes(hwy, displ)) +  
  geom_smooth(method = "lm") +  
  geom_point()
```



```
ggplot(mpg, aes(hwy, displ)) +  
  geom_smooth(method = "lm") +  
  geom_point() +  
  coord_fixed()
```

