Data visualization with R

Schedule:

Jan 30th: grammar of graphics in ggplot2
Feb 6th: Publication-ready figures
Feb 13th: Complex visualizations

1-3 pm, SEH room 1800

Workshop materials:
www.github.com/collnell/GWU-visual

Contact

Colleen Nell
email: collnell@gwu.edu
website: www.collnell.com
twitter: @collnell
office: SEH 6880

Open RStudio

1. go to www.github.com/collnell/GWU-visual to access workshop materials
2. download the `GWU_pubs.R` script
3. open it in RStudio
4. make sure you have the `ggplot2`, `dplyr`, `reshape2`, & `cowplot` packages installed

```
install.packages(c('reshape2','cowplot'))

library(ggplot)
library(dplyr)
library(reshape2)
library(cowplot)
```

Today's objectives

1. Reproducible workflow for publication figures
2. Tweaking plot theme/appearance
3. Multi-panel plots & facets
4. Saving in high resolution

```r
install.packages(c('reshape2','cowplot'))

library(ggplot)
library(dplyr)
library(reshape2)
library(cowplot)
```

Journal guidelines

- use a consistent style - fonts, colors, theme
- annotate as necesary
- ensure image clarity

Science
PLOS ONE
Oecologia

PRISM data

R package to import PRISM data

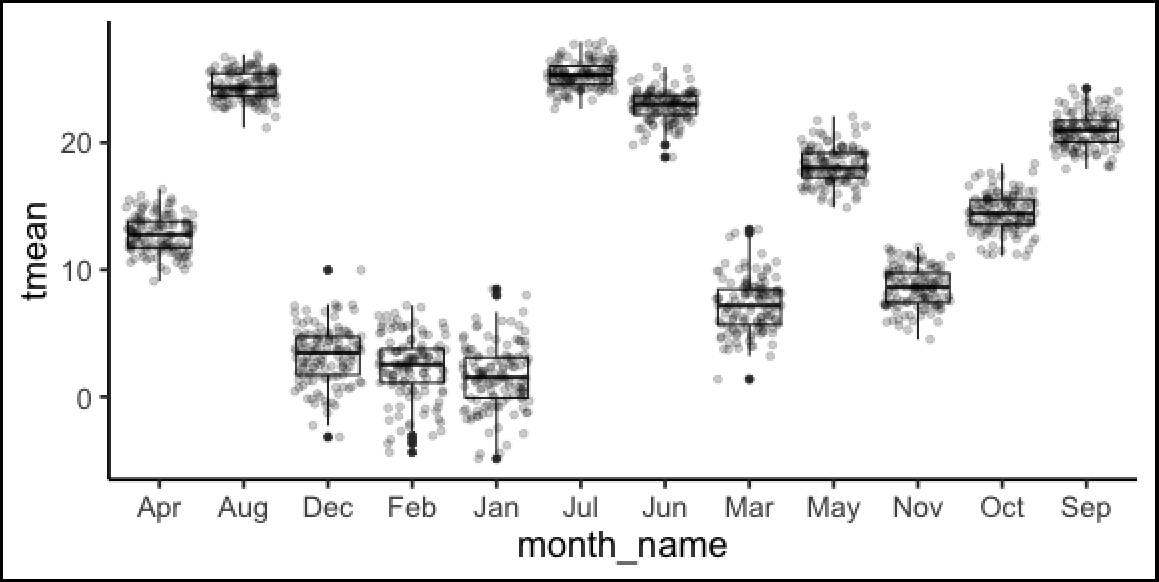
DC monthly temperature mean, min, & max from 1895-present

```
dc<-read.csv('https://raw.githubusercontent.com/collnell/GWU-visual/master/DC_climate.c
head(dc)
```

```
##   year month month_name      tmax       tmin     period       tmean
## 1 1895     1        Jan  4.250909 -4.3827274 historical -0.06590914
## 2 1895     2        Feb  1.610909 -8.3381818 historical -3.36363635
## 3 1895     3        Mar 11.109091  0.5663636 historical  5.83772722
## 4 1895     4        Apr 17.314546  6.2072727 historical 11.76090915
## 5 1895     5        May 22.251818 11.7436364 historical 16.99772731
## 6 1895     6        Jun 29.376364 17.7427271 historical 23.55954534
```

Start with a figure

```
# plot current monthly temperatures, tmean
ggplot(dc, aes(month_name, tmean))+
  geom_boxplot()+
  geom_jitter(alpha=.2)
```



axes

submission guidelines:

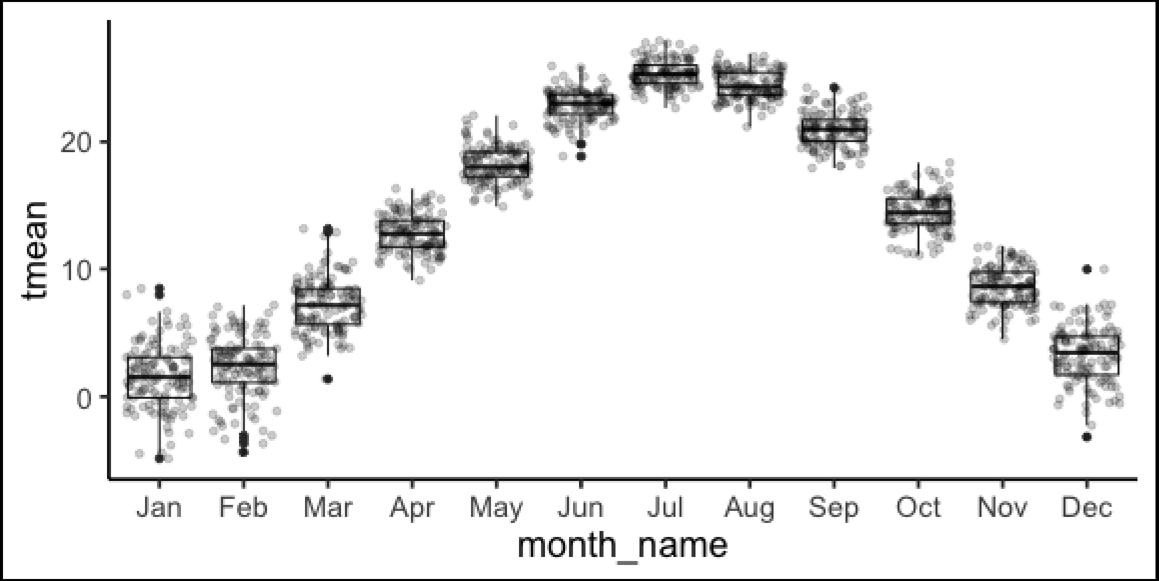
- avoid transformed axes, modify scaling in plot

`scale_x_discrete()` `scale_x_continuous()` `scale_x_log10()` `scale_x_sqrt()` `scale_x_reverse()`

axes

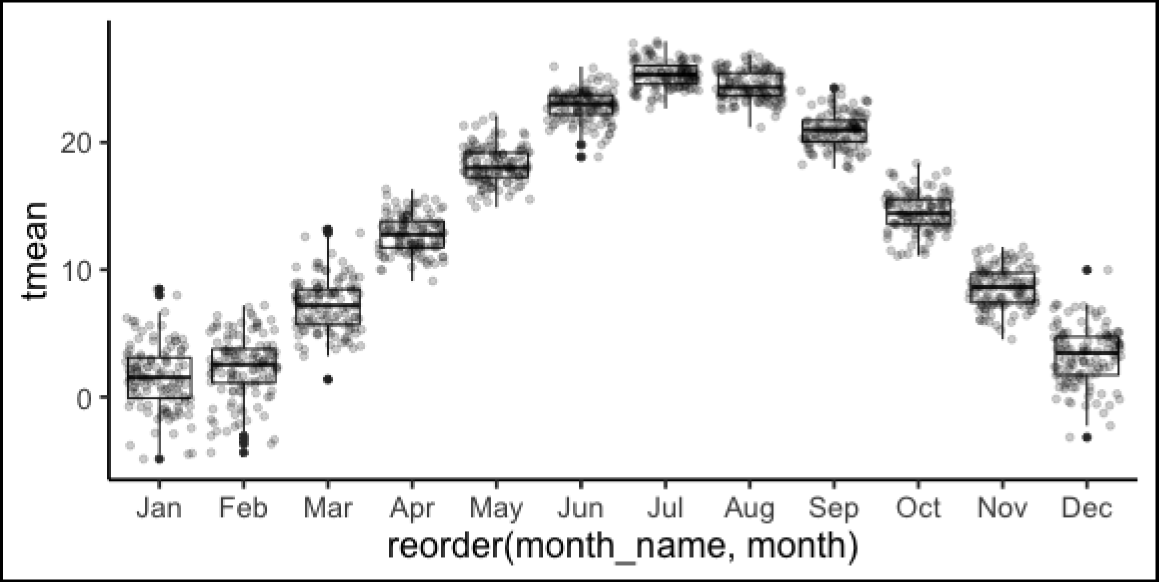
modifying limits, ordering, and number of breaks on axes)

```r
# list of axes in desired order
months<-c('Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec')

ggplot(dc, aes(month_name, tmean))+
  geom_boxplot()+
  geom_jitter(alpha=.2)+
  scale_x_discrete(limits=months)
```



Reorder discrete axis by variable

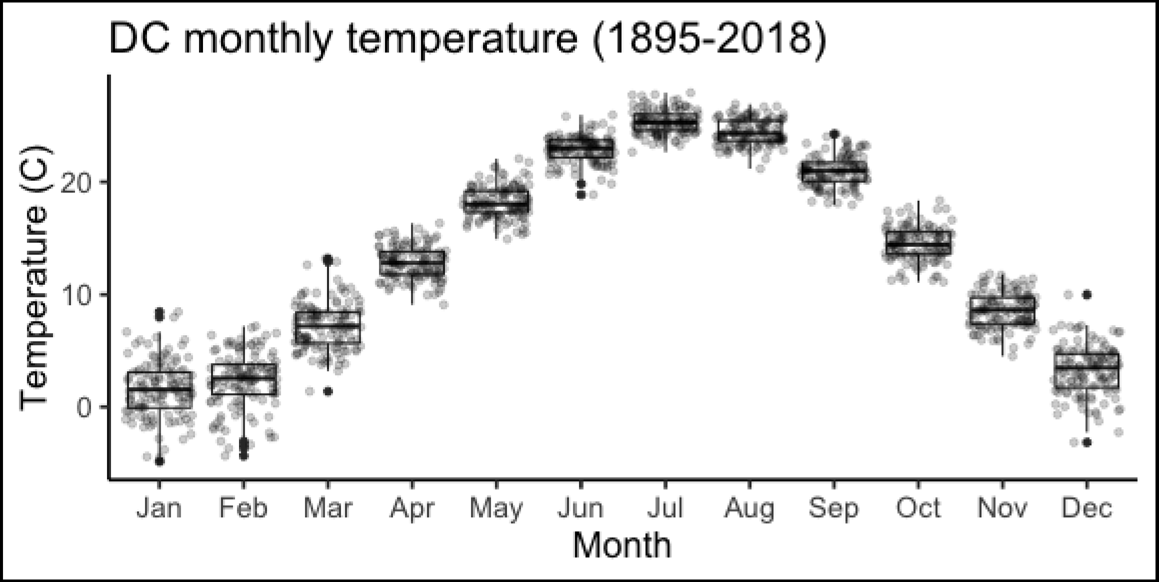
```
plot.tmean<-ggplot(dc, aes(reorder(month_name, month), tmean))+
  geom_boxplot()+
  geom_jitter(alpha=.2)
plot.tmean
```



Add axis labels

```
+ labs(x = ' ', y = ' ', title = ' ')
```

```
plot.tmean+labs(x='Month', y='Temperature (C)', title='DC monthly temperature (1895-201
```



stat_summary

compute summary statistics

```
plot.tmean+stat_summary(color='red') # default is mean_se
plot.tmean+stat_summary(color='red', fun.y='median')
```

```
g<-ggplot(dc, aes(reorder(month_name, month), tmean))
g+stat_summary(geom='bar')
```

stat_summary

compute summary statistics

```
plot.tmean+stat_summary(color='red') # default is mean_se
plot.tmean+stat_summary(color='red', fun.y='median')
```

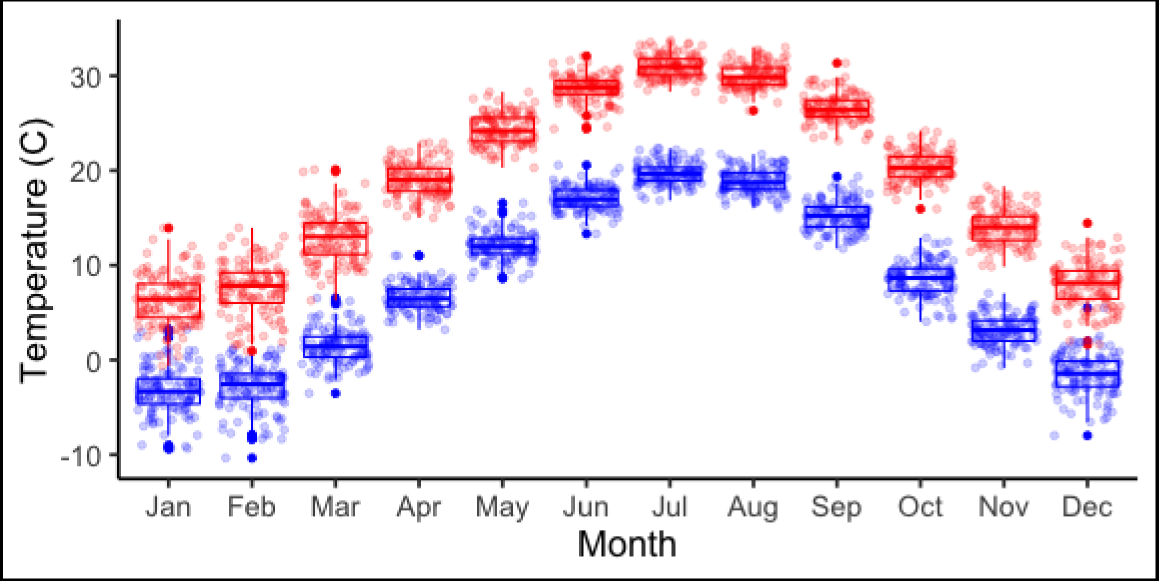
```
g<-ggplot(dc, aes(reorder(month_name, month), tmean))
g+stat_summary(geom='bar')
```

Plotting multiple variables

```
# plot min and max temps
ggplot(dc, aes(month_name))+
  geom_boxplot(aes(y=tmin), color='blue')+
  geom_boxplot(aes(y=tmax), color='red')+
  scale_x_discrete(limits=months)+
  labs(x='Month', y='Temperature (C)')
```

a bit tedious...

```
# plot min and max temps
ggplot(dc, aes(month_name))+
  geom_boxplot(aes(y=tmin), color='blue')+
  geom_boxplot(aes(y=tmax), color='red')+
  geom_jitter(aes(y=tmin), color='blue',alpha=.2)+
  geom_jitter(aes(y=tmax), color='red',alpha=.2)+
  scale_x_discrete(limits=months)+
  labs(x='Month', y='Temperature (C)')
```



reshaping data: wide to long format

```
library(reshape2)
head(dc)
```

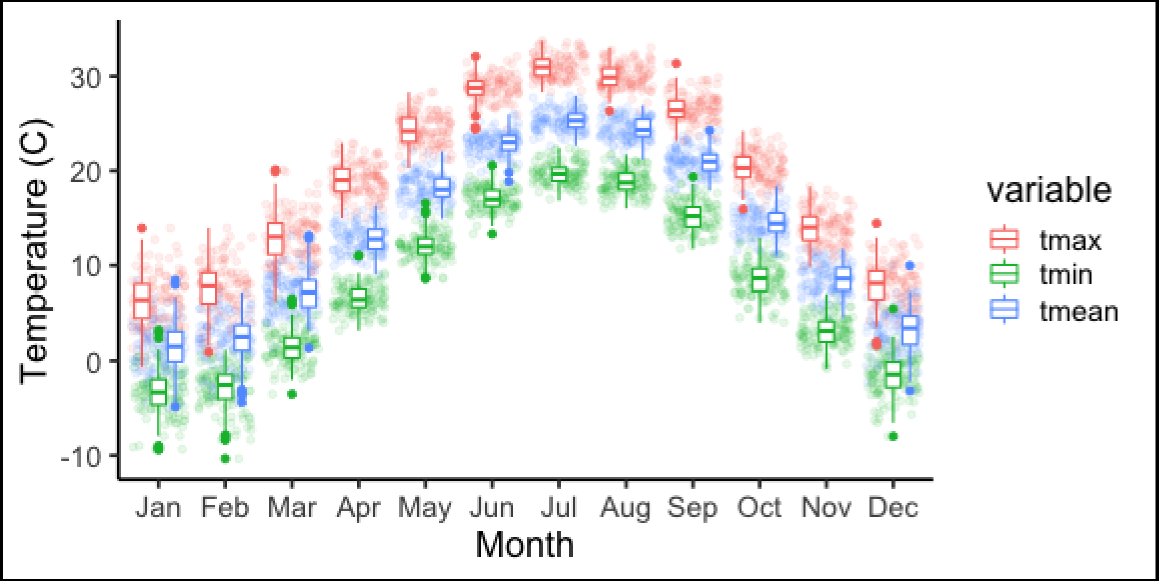
```
##   year month month_name      tmax       tmin     period       tmean
## 1 1895     1        Jan  4.250909 -4.3827274 historical -0.06590914
## 2 1895     2        Feb  1.610909 -8.3381818 historical -3.36363635
## 3 1895     3        Mar 11.109091  0.5663636 historical  5.83772722
## 4 1895     4        Apr 17.314546  6.2072727 historical 11.76090915
## 5 1895     5        May 22.251818 11.7436364 historical 16.99772731
## 6 1895     6        Jun 29.376364 17.7427271 historical 23.55954534
```

```
# id.vars are the grouping variables you would like to keep
# all other variables are 'melted' into 2 columns -
# temp has the former column names & 2 (value) has all the values
dc.melt<-melt(dc, id.vars=c('year','month_name','month','period'), value.name = 'temp')
head(dc.melt)
```

```
##   year month_name month     period variable      temp
## 1 1895        Jan     1 historical     tmax  4.250909
## 2 1895        Feb     2 historical     tmax  1.610909
## 3 1895        Mar     3 historical     tmax 11.109091
## 4 1895        Apr     4 historical     tmax 17.314546
## 5 1895        May     5 historical     tmax 22.251818
## 6 1895        Jun     6 historical     tmax 29.376364
```

assign variable in aes()

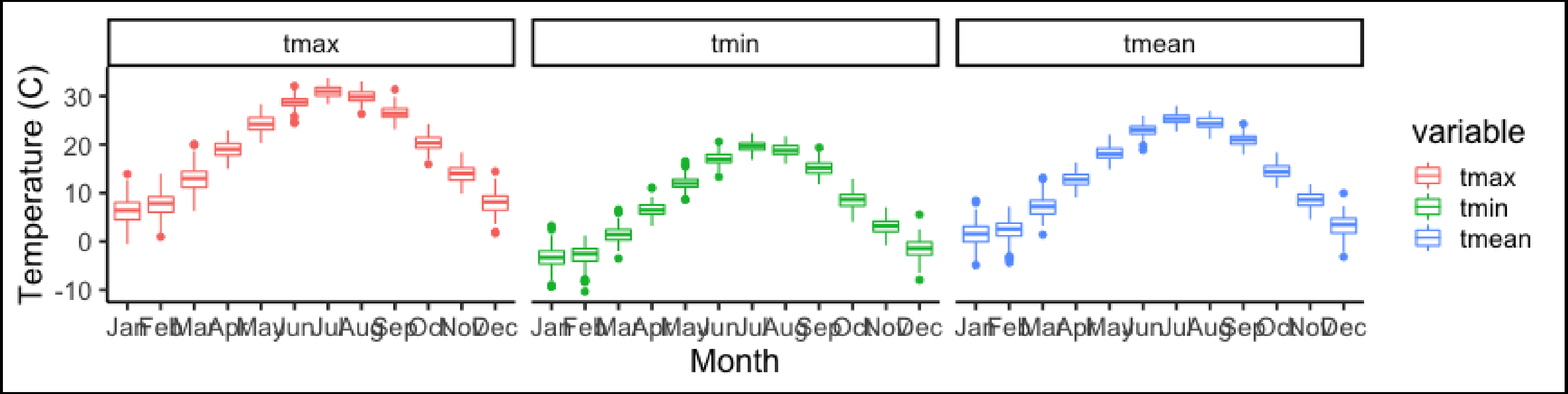
```
g<-ggplot(dc.melt, aes(month_name, temp, color = variable))+
  geom_jitter(alpha=.1)+
  geom_boxplot()+
  scale_x_discrete(limits=months)+
  labs(x='Month', y='Temperature (C)')

g
```



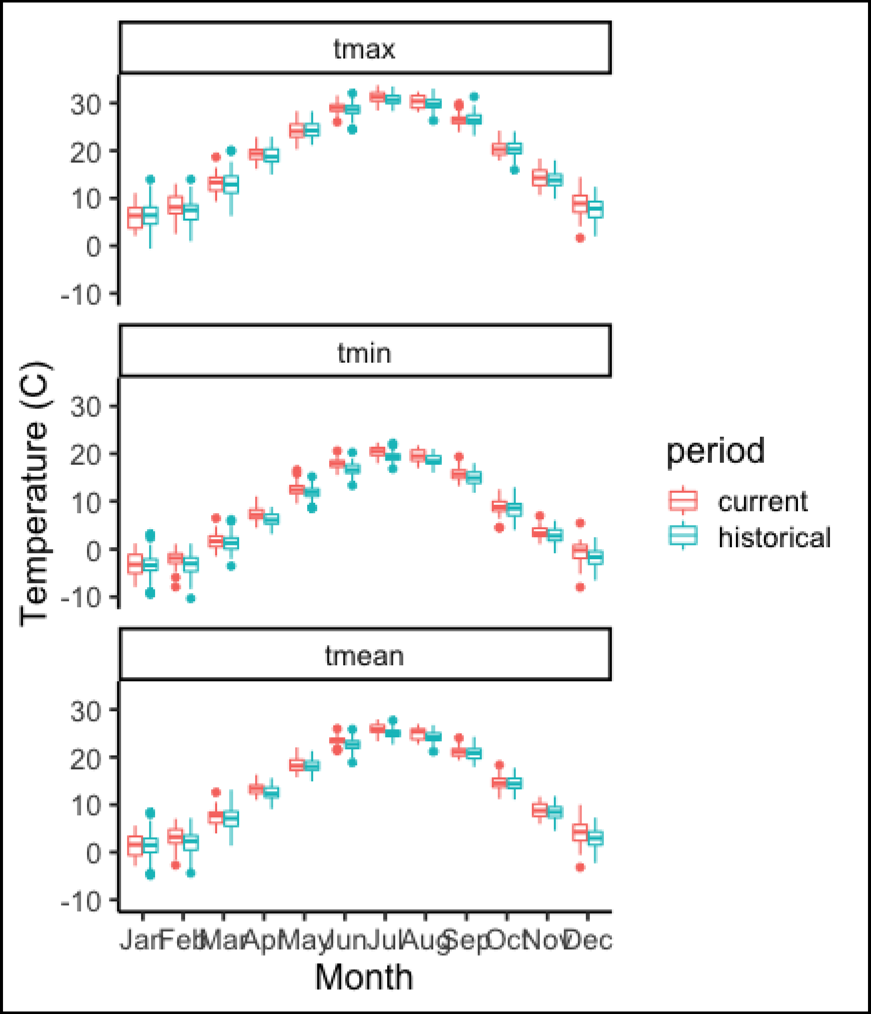
facets

facets

```
ggplot(dc.melt, aes(month_name, temp, color = variable))+
  geom_boxplot()+
  scale_x_discrete(limits=months)+
  labs(x='Month', y='Temperature (C)')+
  facet_wrap(~variable)
```



Recreate this plot:



set color scales

`scale_fill_manual()`, `scale_fill_discrete()`, `scale_fill_gradient()`
`scale_color_manual()`, `scale_color_discrete()`, `scale_color_gradient()`

Colors:
Named colors
Hexadecimal color codes
i want hue

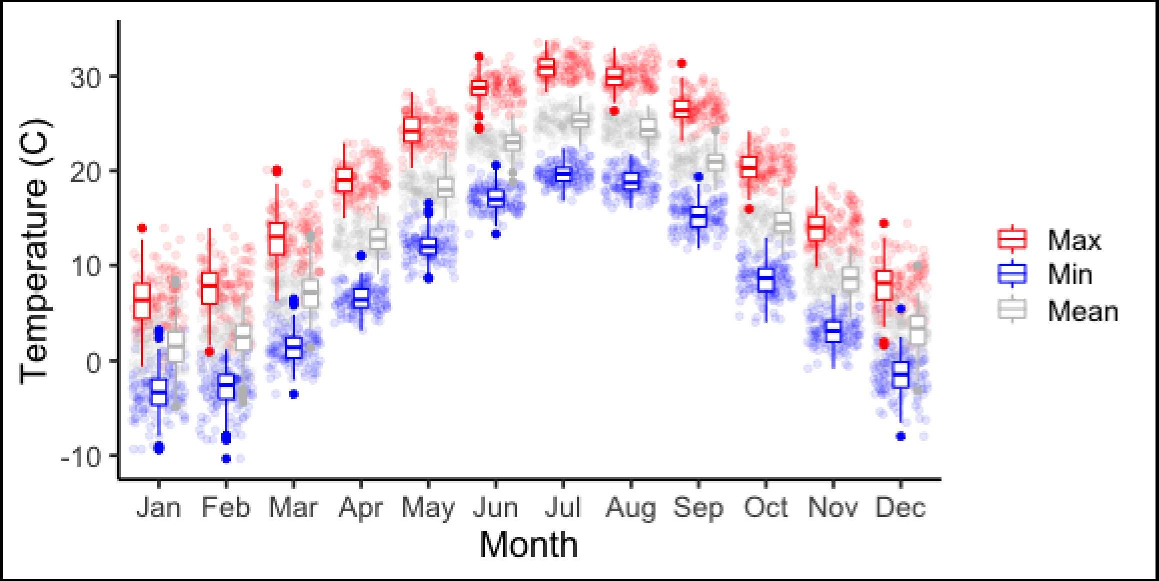
journal considerations:

- avoid excessive use of color, use shapes and fill when possible
- may charge extra to print in color (Oecologia $1150 per article)

set discrete color scale

```
g<-ggplot(dc.melt, aes(month_name, temp, color = variable))+
  geom_jitter(alpha=.1)+
  geom_boxplot()+
  scale_x_discrete(limits=months)+
  labs(x='Month', y='Temperature (C)')
```

```
g+scale_color_manual(labels=c('Max','Min','Mean'),values=c('red','blue','grey'), name='
```



color scale packages

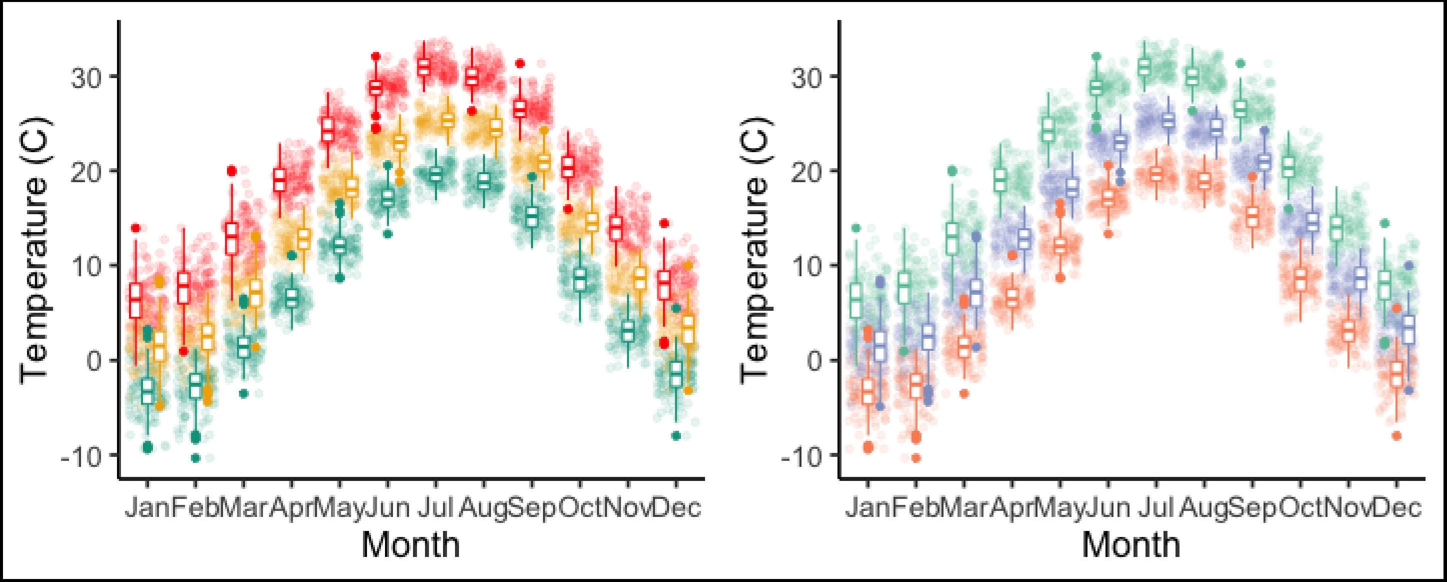
viridis
colormap
RColorBrewer
wesanderson
LACroixColoR

```
library(RColorBrewer)
display.brewer.all()
```

```
library(wesanderson)
names(wes_palettes)
```

discrete color scales

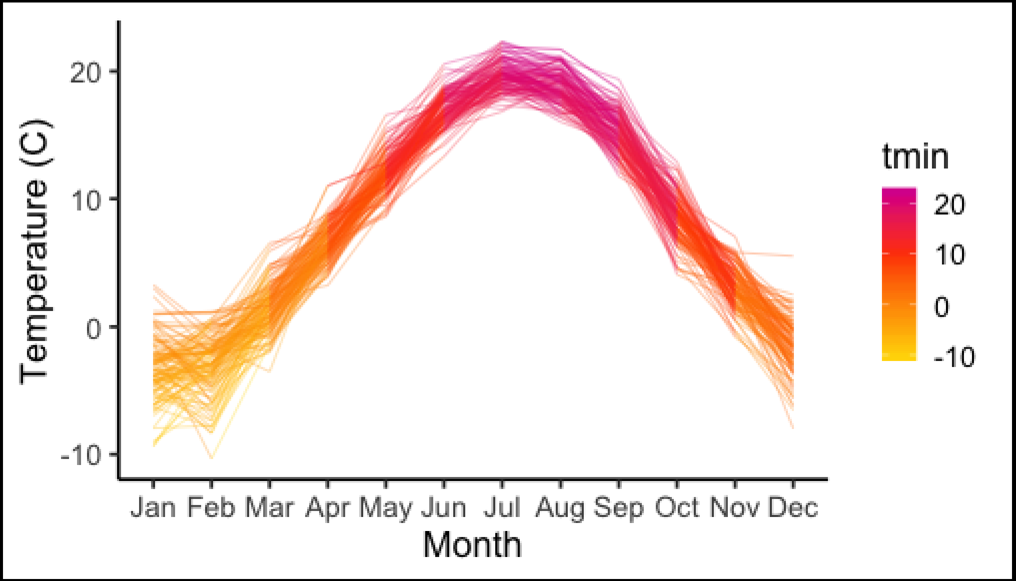
```
library(wesanderson)

g+scale_color_manual(values=wes_palette('Darjeeling1'))
g+scale_color_brewer(palette='Set2')
```



set gradient color scale

`scale_color_gradient()`- 2 color gradient
`scale_color_gradientn()` - with n colors
`scale_color_gradient2(low = , mid= , high = )` - diverging color gradient

```
g<-ggplot(dc, aes(month_name, tmin))+
  geom_line(aes(color=tmin, group=year), alpha=.3)+ # what happens when change to color
  scale_x_discrete(limits=months)+
  labs(x='Month', y='Temperature (C)')

g+scale_color_gradient2(low='gold',mid='orangered',high='purple', midpoint=10)
```



colorblind friendly

```r
library(viridis)
```

```
## Loading required package: viridisLite
```

```r
g+scale_color_viridis(direction=-1)
```

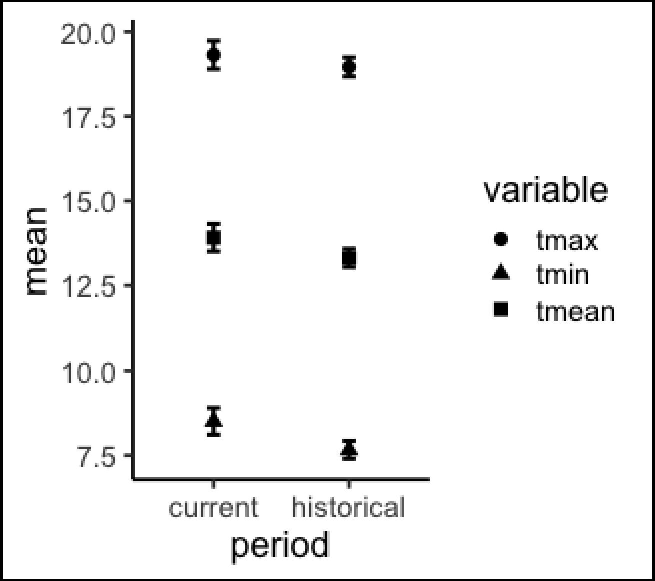
shape scales

Oecologia: Make symbols intuitive (e.g., if you manipulated light levels use “•” for shaded plots and “o” for open plots). Preferred datapoint symbols are circles, triangles, squares, diamonds. Avoid using symbols for datapoints such as “*”, “+”, “-”, letters, words, etc. We prefer only open or filled symbols

```r
se<-function(x) sd(x, na.rm=TRUE)/sqrt(length(x))# standard error

## mean temp by period
temps<-dc.melt%>%
  group_by(period, variable)%>%
  summarize(mean = mean(temp), se = se(temp))
```

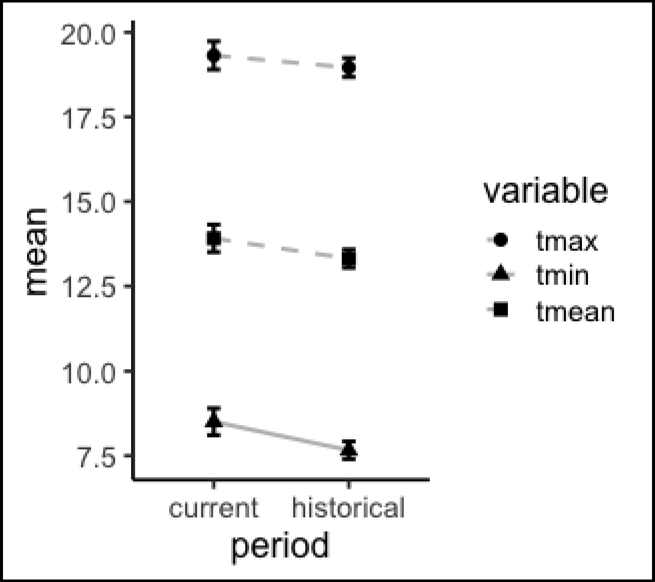
mean and error

```
ggplot(temps, aes(period, mean, shape=variable))+
  geom_point(size=3)+
  geom_errorbar(aes(ymin=mean-se, ymax=mean+se), width=0.1, size=1)
```



linetype

```
ggplot(temps, aes(period, mean, shape=variable))+
  geom_line(aes(group=variable, linetype=variable),color='gray', size=1)+
  geom_point(size=3)+
  geom_errorbar(aes(ymin=mean-se, ymax=mean+se), width=0.1, size=1)+
  scale_linetype_manual(values=c('dashed','solid','dashed'))
```



barplot

```
ggplot(temps, aes(variable, mean, fill=period))+
  geom_bar(stat='identity')
```

```
plot.bar<-ggplot(temps, aes(variable, mean, fill=period))+
  geom_bar(stat='identity', position=position_dodge(1), color='black')+
  geom_errorbar(aes(ymin=mean-se, ymax=mean+se), width=.1, position=position_dodge(1))+
  scale_fill_manual(values=c('grey','white'))
```

built-in themes

```
g+theme_classic()
g+theme_bw()
g+theme_minimal()
g+theme_gray()

# set theme for R session - applies to all plots automatically
theme_set(theme_classic(base_size=18)) # scale font sizes
```

theme elements

```
theme_classic
```

```
## function (base_size = 11, base_family = "", base_line_size = base_size/22,
##     base_rect_size = base_size/22)
## {
##     theme_bw(base_size = base_size, base_family = base_family,
##         base_line_size = base_line_size, base_rect_size = base_rect_size) %+replace%
##         theme(panel.border = element_blank(), panel.grid.major = element_blank(),
##             panel.grid.minor = element_blank(), axis.line = element_line(colour = "black"
##                 size = rel(1)), legend.key = element_blank(),
##             strip.background = element_rect(fill = "white", colour = "black",
##                 size = rel(2)), complete = TRUE)
## }
## <bytecode: 0x7f8325ed0408>
## <environment: namespace:ggplot2>
```

submission guidelines

- sans serif fonts (e.g. arial or helvetica), avoid serif (Times New Roman)
- consistent font sizes, minimal variation
- keep key within borders of figure

theme elements

The function `theme()` is used to control non-data parts of the graph including:

Line elements: axis lines, minor and major grid lines, plot panel border, axis ticks background color
`axis.line`, `axis.line.x`, `axis.line.y`, `plot.grid.major`, `panel.border`
Text elements: plot title, axis titles, legend title and text, axis tick mark labels
`axis.title`, `axis.title.x`
Rectangle elements: plot background, panel background, legend background
`panel.border`

There is a specific function to modify each of these three elements:
`element_line()` to modify the line elements of the theme
`element_text()` to modify the text elements
`element_rect()` to change the appearance of the rectangle elements
`element_blank()` to remove theme element

theme elements

```
# rotate x-axis labels
g+theme(axis.text.x = element_text(angle=90))

# modify gridlines
g+theme(panel.grid.major=element_line(color='grey', linetype='dotted'))

# add box around plot
g+theme(panel.border = element_rect(color='black', fill=NA, size=2))

# remove elements
g+theme(axis.line = element_blank(), axis.text=element_blank(), axis.ticks=element_blan
```

legends

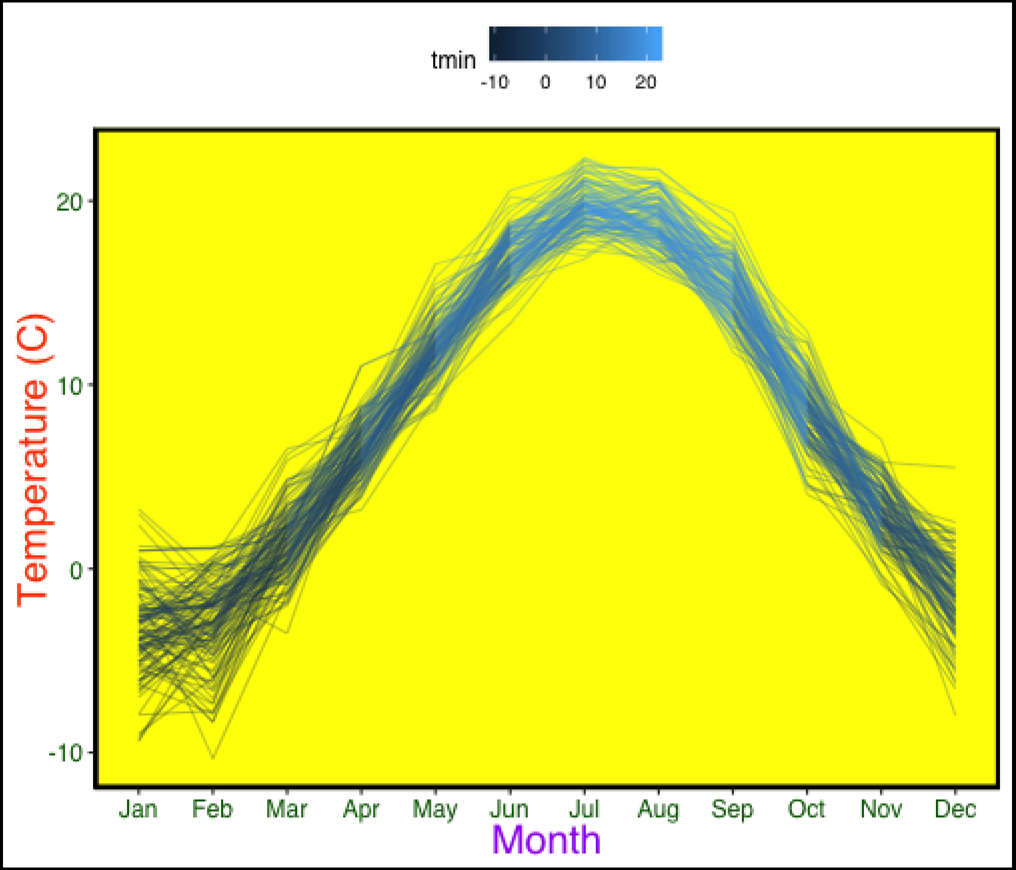
```
g<-g+scale_color_viridis()
g+theme(legend.position='none')# top, bottom, left, right
g+theme(legend.position = c(0.2,.9), legend.direction='horizontal', legend.background=e
```

Custom themes

```r
theme_example <- function (base_size = 12, base_family = "sans") {
  theme_classic(base_size = base_size, base_family = base_family) %+replace%
    theme(axis.text = element_text(colour = "darkgreen"),
          axis.title.x = element_text(colour = "purple", size=20),
          axis.title.y = element_text(colour = "orangered", angle = 90, size=20),
          panel.background = element_rect(fill = "yellow", size=2),
          panel.grid.minor.y = element_blank(),
          panel.grid.minor.x = element_blank(),
          panel.grid.major = element_blank(),
          plot.background = element_blank(),
          legend.position = 'top'
  )
}
```

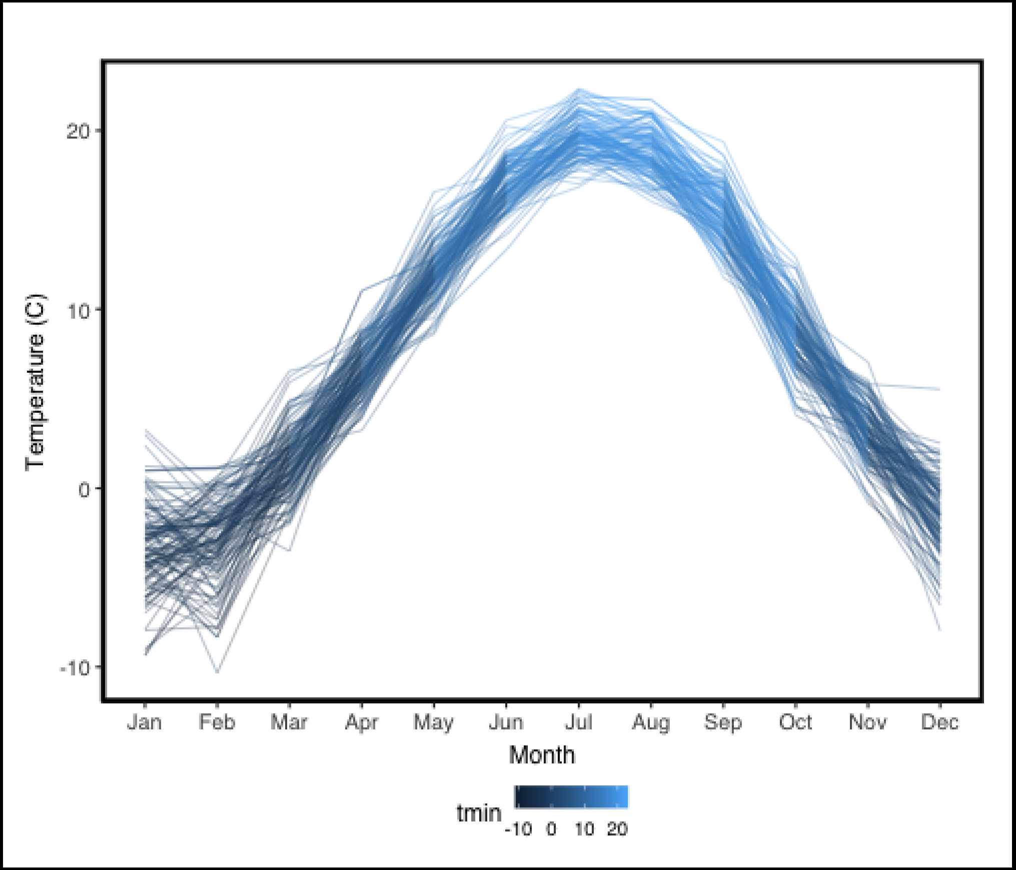
Custom themes

```
g+theme_example()
```



Reusing themes

```
source('theme_custom.R')

g+theme_mooney()
```



break

calculate annual mean, min, and max temperatures

```
head(dc)
```

```
##   year month month_name      tmax       tmin     period       tmean
## 1 1895     1        Jan  4.250909 -4.3827274 historical -0.06590914
## 2 1895     2        Feb  1.610909 -8.3381818 historical -3.36363635
## 3 1895     3        Mar 11.109091  0.5663636 historical  5.83772722
## 4 1895     4        Apr 17.314546  6.2072727 historical 11.76090915
## 5 1895     5        May 22.251818 11.7436364 historical 16.99772731
## 6 1895     6        Jun 29.376364 17.7427271 historical 23.55954534
```

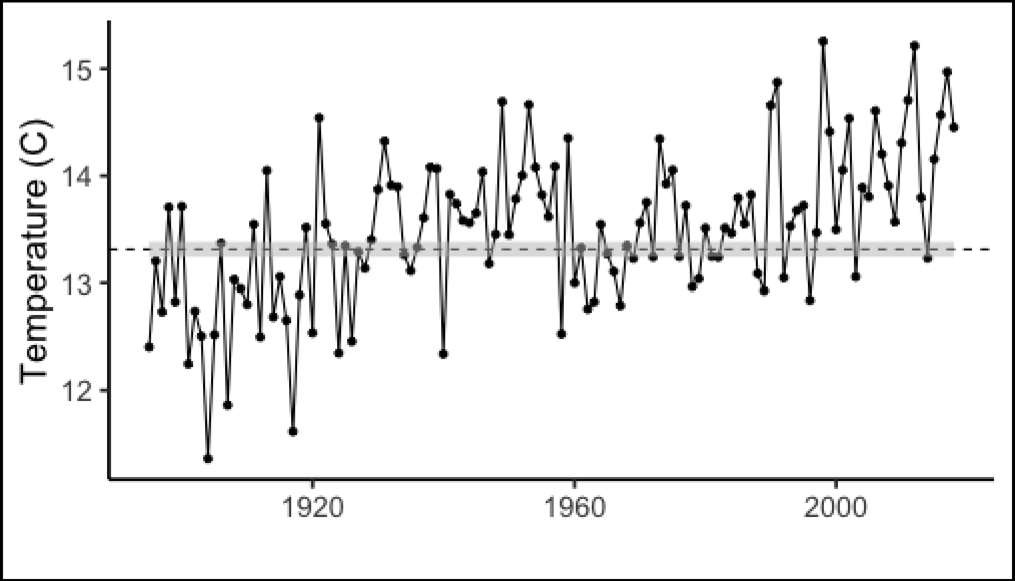
```
dc.yr<-dc%>%
  group_by(period, year)%>%
  summarize(mean = mean(tmean), max=max(tmax), min=min(tmin))

# calculate historical mean
dc.past<-dc.yr%>%filter(period == 'historical')
hist.mean<-mean(dc.past$mean)
hist.se<-se(dc.past$mean)
```

annotating plots

geom_text(), geom_label, annotate(), geom_hline(), geom_vline(), geom_ribbon()

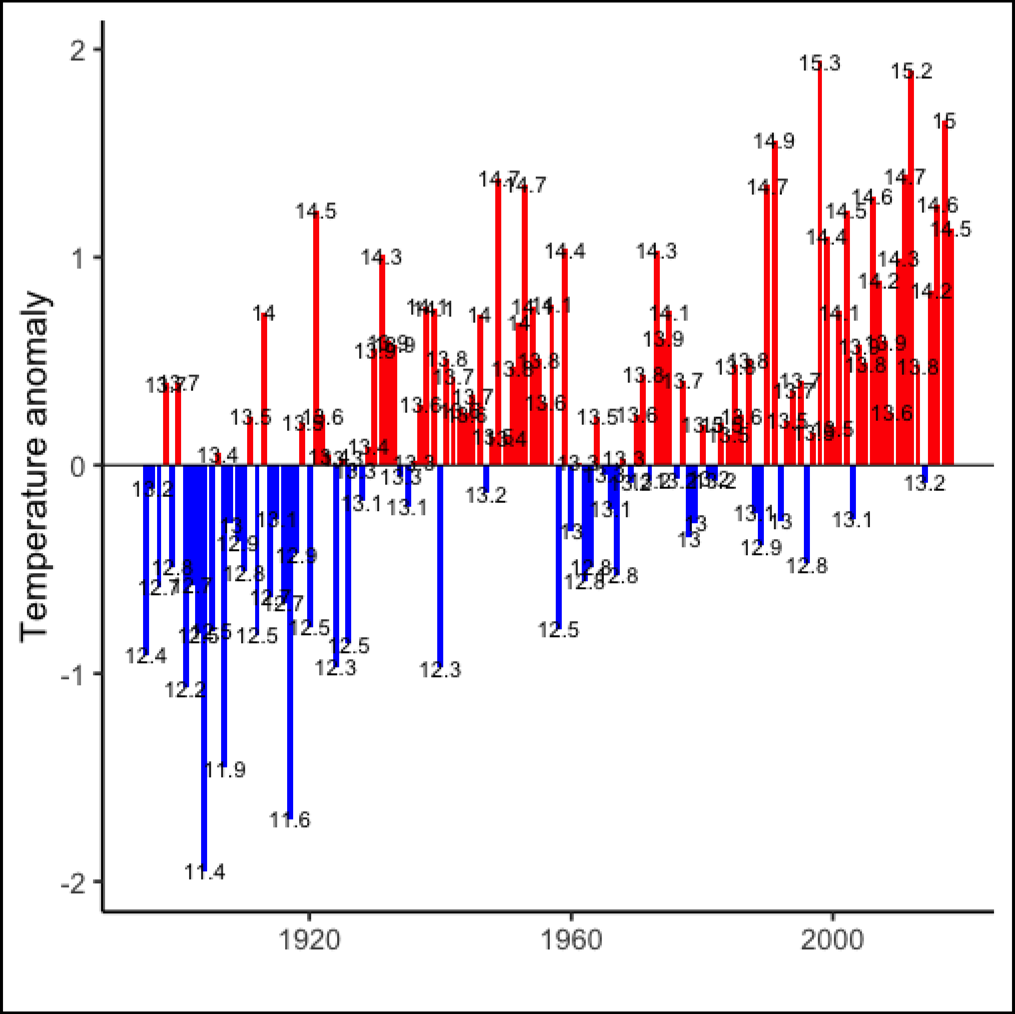
```
timeline<-ggplot(dc.yr, aes(year, mean))+
  geom_line()+
  geom_hline(yintercept =hist.mean, linetype = 'dashed')+
  geom_point()+
  labs(x='', y='Temperature (C)')+
  theme(legend.position='none')
```

```
timeline+geom_ribbon(aes(ymin=hist.mean-hist.se, ymax=hist.mean+hist.se), fill='gray',
```



```r
temp.time<-ggplot(dc.yr, aes(year, anomaly, fill=anom_color))+
  geom_bar(stat='identity')+
  geom_hline(yintercept = 0)+
  scale_fill_manual(values=c('red','blue'))+
  theme(legend.position='none')+
  labs(x='', y='Temperature anomaly')

temp.time+
  annotate(geom='text', y=-1, x=2010, label = 'historical mean', size=5)+
  annotate('segment', x=2010, xend=2011, y=-.9, yend=0, arrow=arrow(), size=1)
```

```
temp.time+geom_text(aes(label=round(mean,1)))
```



Multiple plots

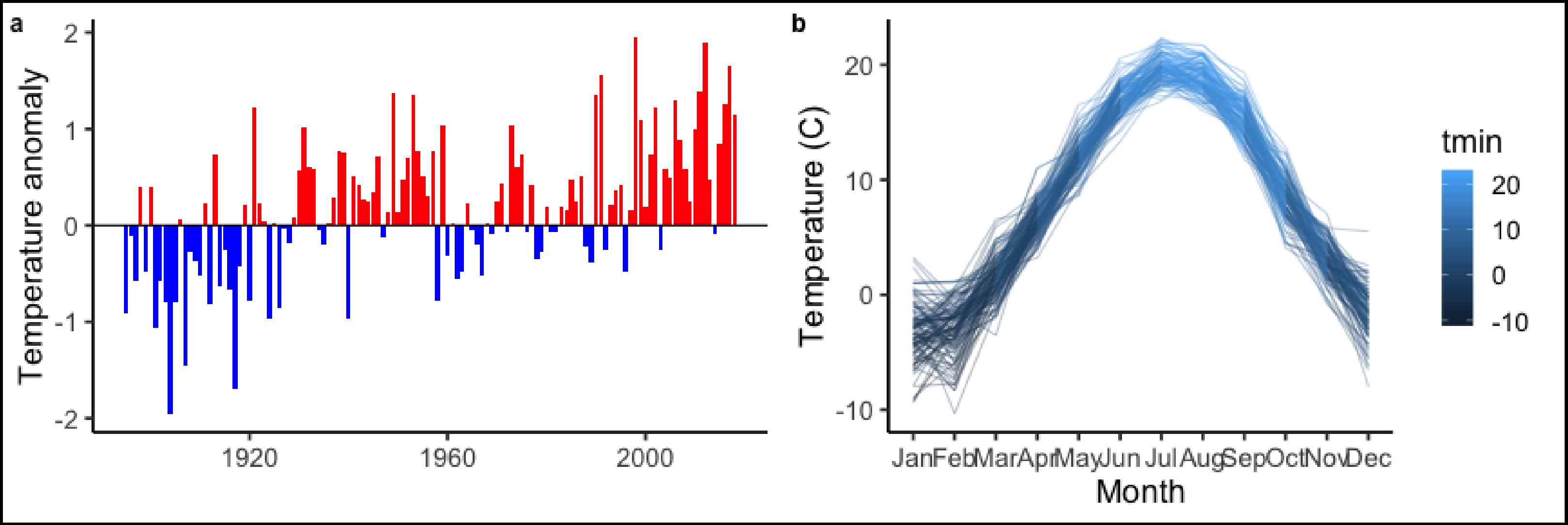
submission guidelines:

- multi-paneled figures should be grouped on a single page
- label each panel in figure
- arrange panels to share common axes if possible

Multiple plots

arguments: nrow, ncol, rel_widths, rel_heights, align, labels, label_size

```
library(cowplot)

plot_grid(temp.time, g, nrow=1, ncol=2, labels=c('a','b'))
```



Saving

Size of Figures: Figures in Oecologia are usually published in one column **width (84 mm)** although they may be reproduced as **1.5 or 2 column widths**. Check that all lines and lettering within the figures are legible and **do not appear pixilated** at the expected final size. All lines in the final size of figures should be at least 0.1 mm (0.3 pt) wide. Text in the final size of figures should be 8 to 12 pt font. Submitted figures should not exceed the print area of **174 X 234 mm (approx. 7 X 9.4 inches)**.

Science - The width of figures, when printed, will usually be 5.5 cm (2.25 inches or 1 column) or 12.0 cm (4.75 inches or 2 columns). Bar graphs, simple line graphs, and gels may be reduced to a smaller width. Symbols and lettering should be large enough to be legible after reduction [a reduced size of about 7 points (2 mm) high, and not smaller than 5 points]. Avoid wide variation in type size within a single figure. In laying out information in a figure, the objective is to maximize the space given to presentation of the data. Avoid wasted white space and clutter.

size + quality + format

Vector vs raster

saving

ggsave defaults to the last plot displayed

```
## pdf
temp.time
ggsave("cool_fig.pdf", device='pdf', width = 6, height = 4, units='in', dpi=300)

#png
ggsave("cool_fig.png", device='png', width = 6, height = 4, units='in', dpi=300)
```

ggpubr()

Publication-ready plots with ggpubr

```r
library(ggpubr)
```