GGPLOT2 CONTINUED…

Code Club - 19th May 2022

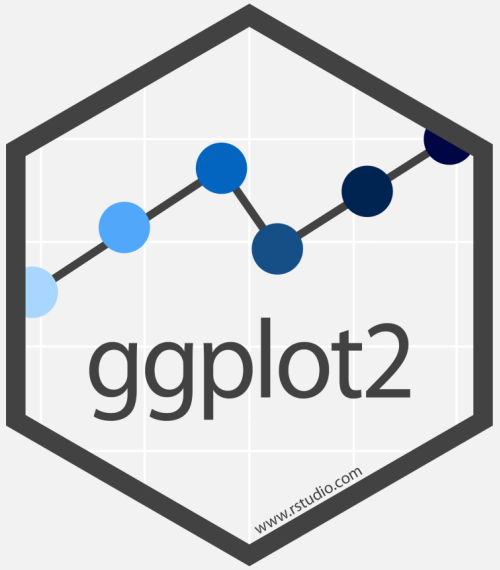
Olivia Johnson

OUTLINE FOR TODAY

- Quick recap of last week
- A new dataset
- Plotting times series data
- Practise plotting!

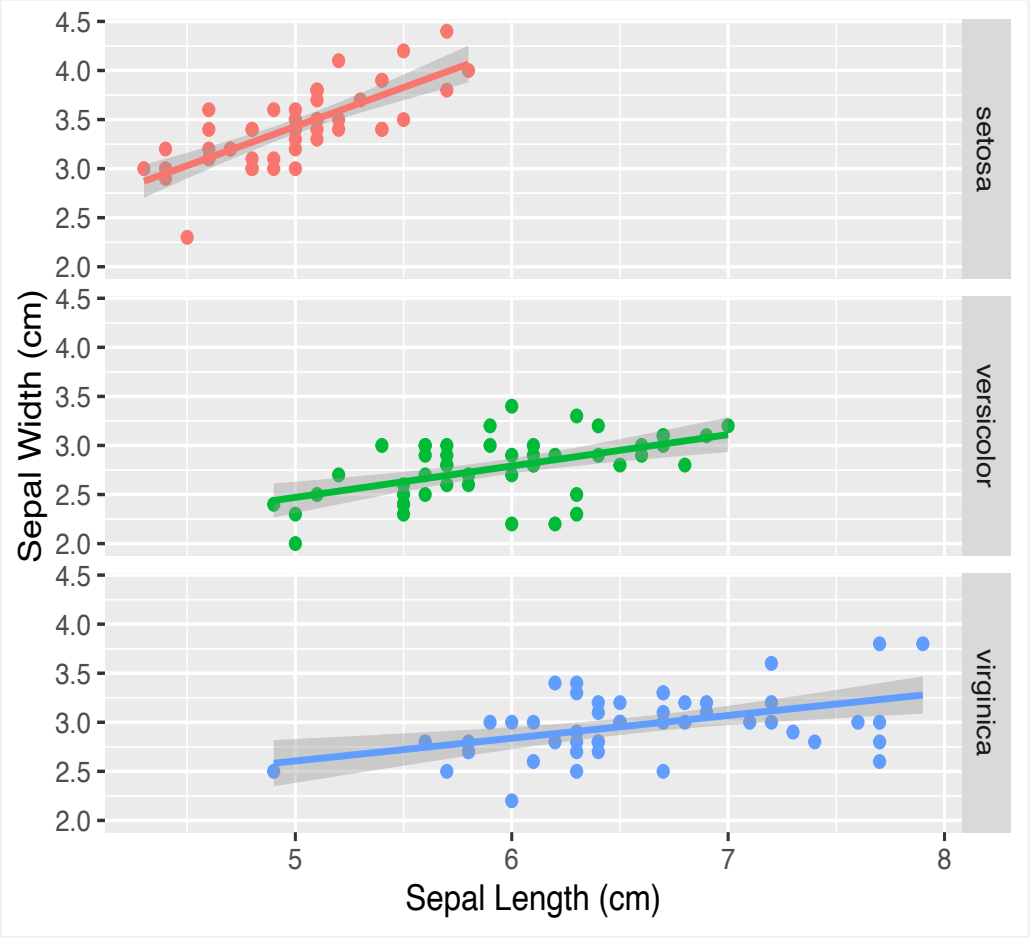
QUICK RECAP

- ggplot basic structure
  - `ggplot(data = <DATA>, mapping = aes(<MAPPINGS>)) + <GEOM_FUNCTION>()`
- Mappings
  - `x, y, col etc`
- Geom_functions
  - `geom_point, geom_violin, geom_boxplot, geom_smooth`
- Facets
  - Using `facet_wrap` and `facet_grid`



QUICK RECAP

- Tidying plot
  - `labs()` – to change axis labels
  - `theme(legend.position="none")` – to remove legend key
- Saving plots
  - `ggsave("plot.jpg", plot = plot, width = 5, height = 5)`



```
> soy
# A tibble: 9,897 x 6
   entity code   year human_food animal_feed processed
   <chr>  <chr> <dbl>      <dbl>        <dbl>     <dbl>
 1 Africa NA     1961      33000         6000     14000
 2 Africa NA     1962      43000         7000     17000
 3 Africa NA     1963      31000         7000      5000
 4 Africa NA     1964      43000         6000     14000
 5 Africa NA     1965      34000         6000     12000
 6 Africa NA     1966      41000         6000      2000
 7 Africa NA     1967      47000         6000      4000
 8 Africa NA     1968      50000         7000      3000
 9 Africa NA     1969      52000         6000      6000
10 Africa NA     1970      52000         6000      8000
# … with 9,887 more rows
```

TODAY'S DATASET

Soybean production and use by year and country, from tidytuesdays
https://github.com/rfordatascience/tidytuesday/tree/master/data/2021/2021-04-06

```
> soy
# A tibble: 9,897 x 6
   entity code   year human_food animal_feed processed
   <chr>  <chr> <dbl>      <dbl>       <dbl>     <dbl>
 1 Africa NA     1961      33000        6000     14000
 2 Africa NA     1962      43000        7000     17000
 3 Africa NA     1963      31000        7000      5000
 4 Africa NA     1964      43000        6000     14000
 5 Africa NA     1965      34000        6000     12000
 6 Africa NA     1966      41000        6000      2000
 7 Africa NA     1967      47000        6000      4000
 8 Africa NA     1968      50000        7000      3000
 9 Africa NA     1969      52000        6000      6000
10 Africa NA     1970      52000        6000      8000
# … with 9,887 more rows
```

WHAT CAN WE PLOT

TIME SERIES DATA

- Today's data can be plotted over time!
- Most obvious option is a line graph.
- `Use geom_line() function`

TIME SERIES DATA

- Today's data can be plotted over time!
- Most obvious option is a line graph.
- `Use geom_line() function.`

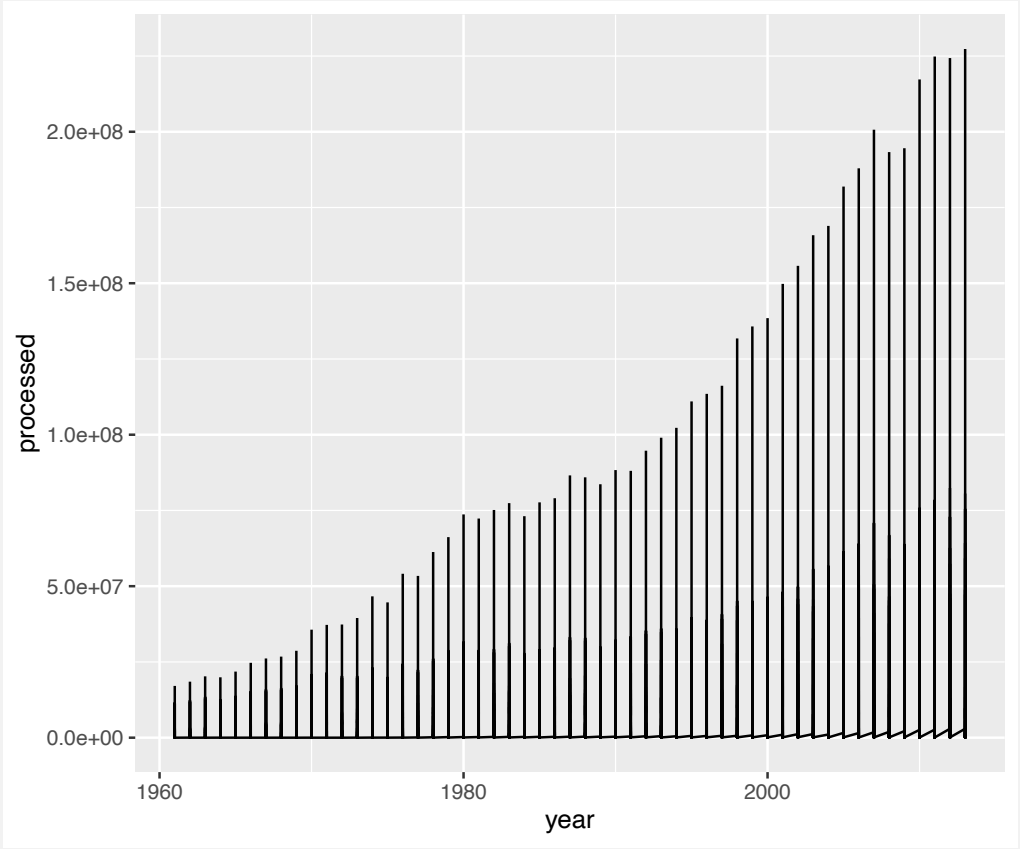
```
ggplot(soy, aes(x=year, y=processed))+
  geom_line()
```

TIME SERIES DATA

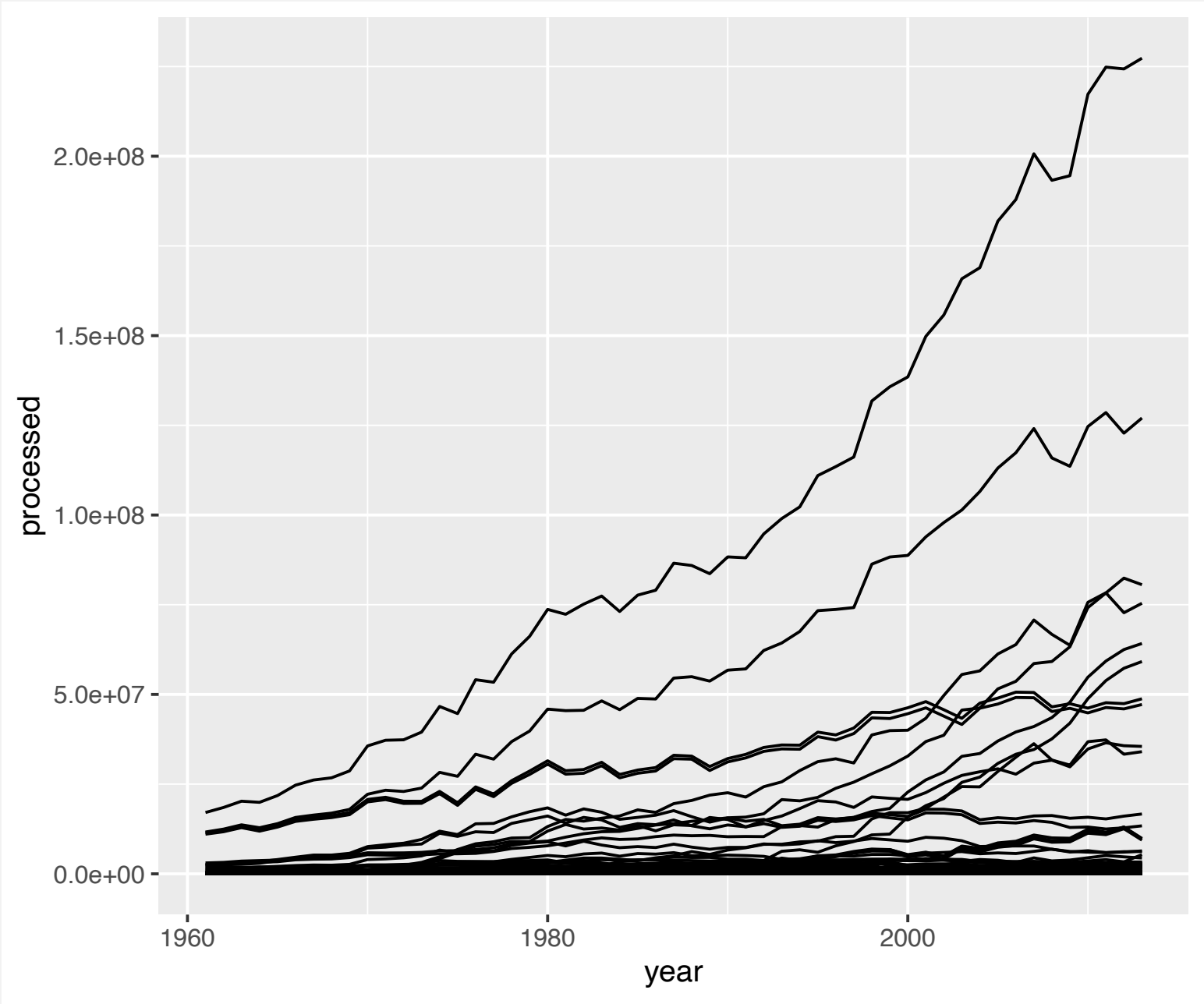
- Today's data can be plotted over time!
- Most obvious option is a line graph.
- `Use geom_line() function.`

```
ggplot(soy, aes(x=year, y=processed))+
   geom_line()
```

- If multiple variables need to use colour or group to separate them

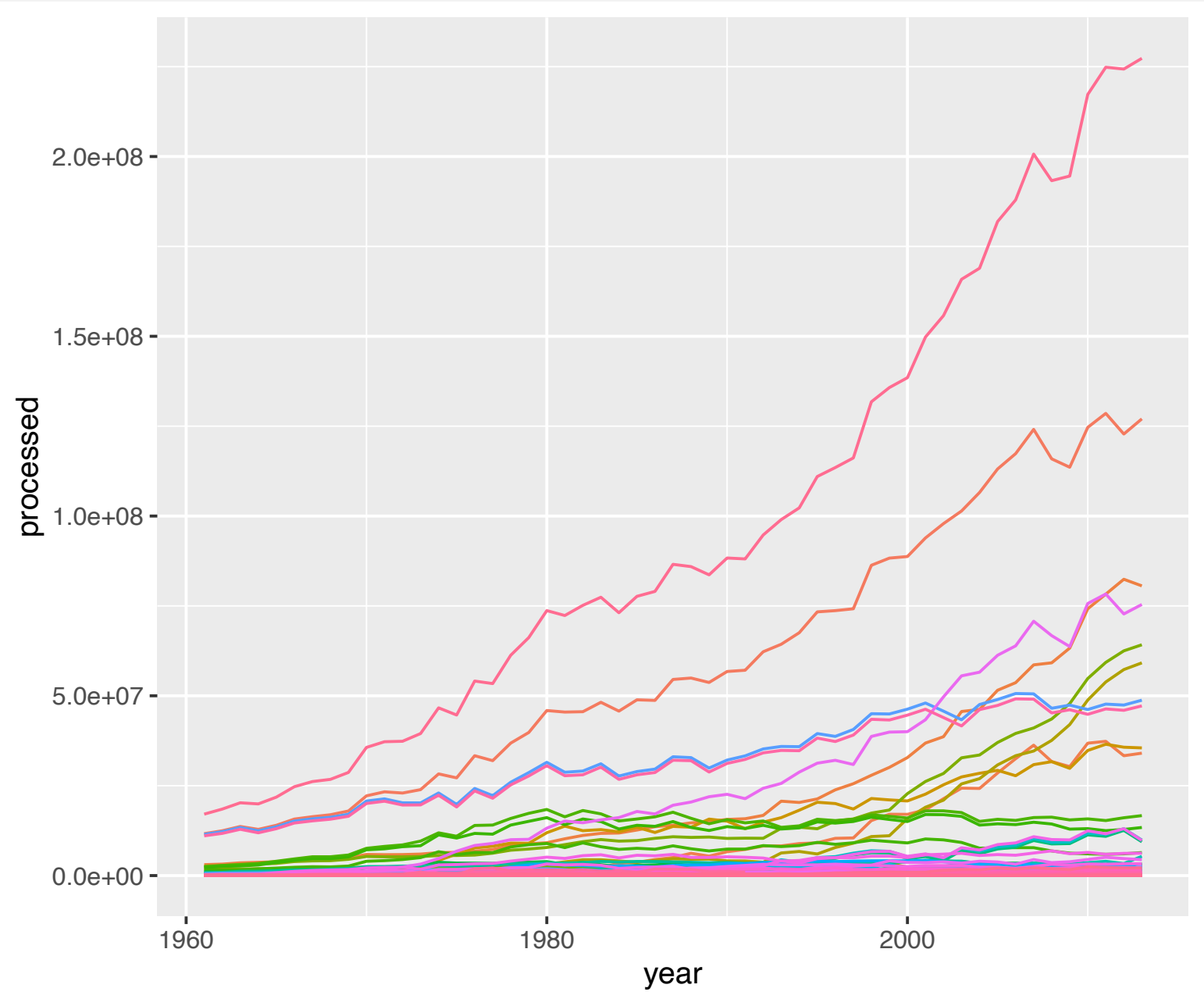


```
ggplot(soy, aes(x= year,
y=processed, group= entity))+
  geom_line()
```



```
ggplot(soy, aes(x= year,
y=processed, colour= entity))+
  geom_line()

** too many colour variable, need
to remove legend
```

```
ggplot(soy, aes(x= year,
y=processed, colour= entity))+
  geom_line()+
theme(legend.position="none")
```



ADDING MORE VARIABLES

- Want to add animal and human food amounts
- Plot will be too busy with all the lines
- Subset to a portion of the data, in this case the total amounts for the world

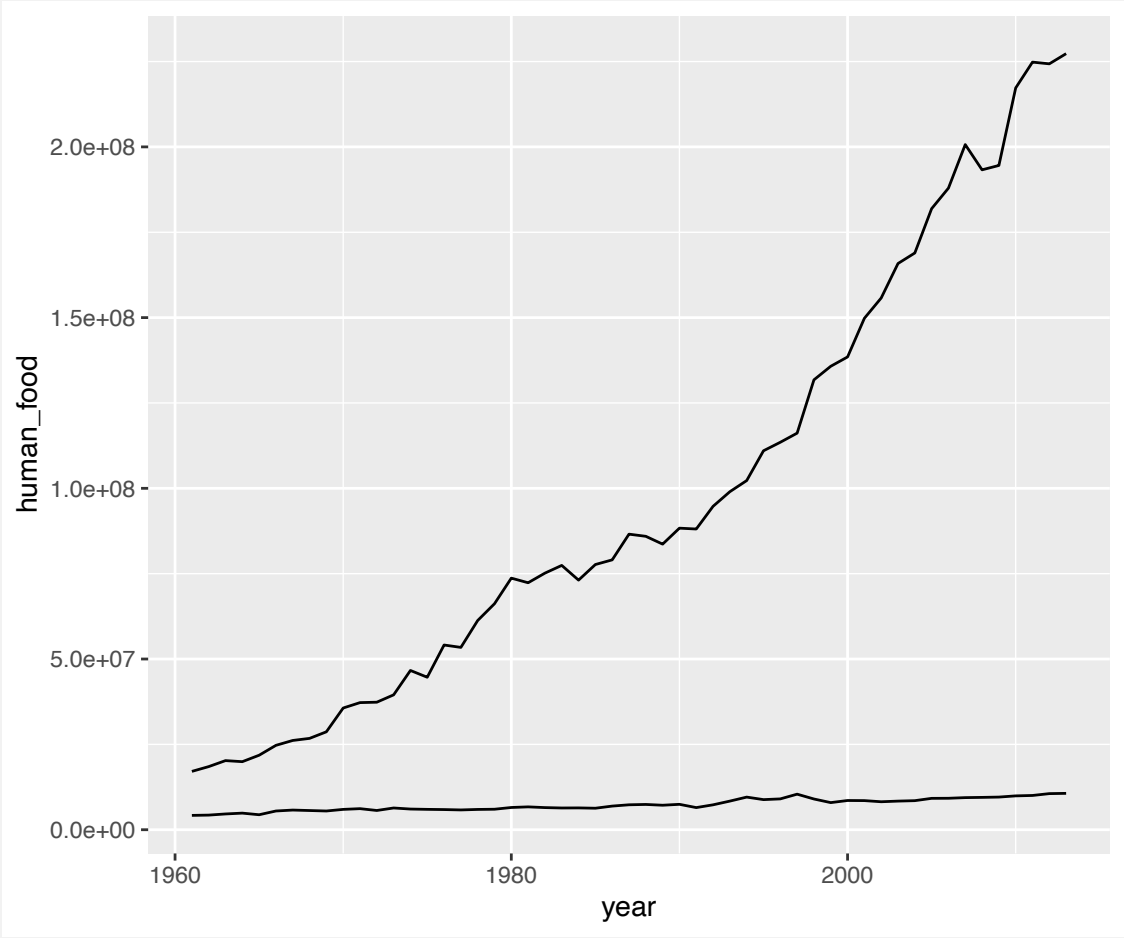
```
df <- soy %>% filter(entity=="World")
```

- Leaves us with a much more manageable dataset

```
> df
# A tibble: 53 x 6
   entity code      year human_food animal_feed processed
   <chr>  <chr>    <dbl>      <dbl>       <dbl>     <dbl>
 1 World  OWID_WRL  1961    4202000      368000  17086000
 2 World  OWID_WRL  1962    4281000      440000  18487000
 3 World  OWID_WRL  1963    4619000      498000  20240000
 4 World  OWID_WRL  1964    4857000      487000  19927000
 5 World  OWID_WRL  1965    4391000      531000  21814000
 6 World  OWID_WRL  1966    5487000      759000  24715000
 7 World  OWID_WRL  1967    5758000      690000  26147000
 8 World  OWID_WRL  1968    5620000      825000  26756000
 9 World  OWID_WRL  1969    5485000      756000  28679000
10 World  OWID_WRL  1970    5958000      607000  35646000
# … with 43 more rows
```

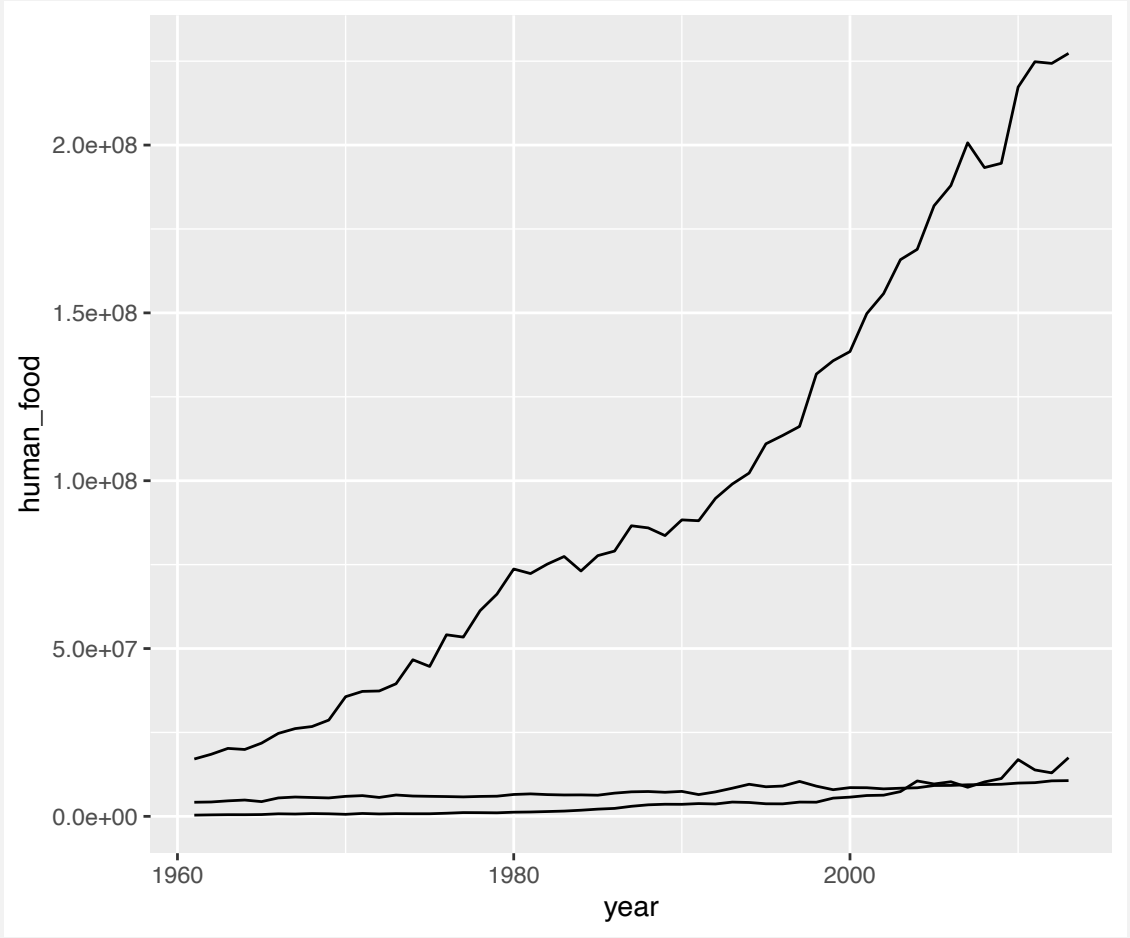
TO ADD ADDITIONAL VARIABLES, ADD MORE GEOM LAYERS

```
ggplot(soy, aes(x= year))+

  geom_line(aes(y= processed))+

  geom_line(aes(y= human_food))
```



TO ADD ADDITIONAL VARIABLES, ADD MORE GEOM LAYERS

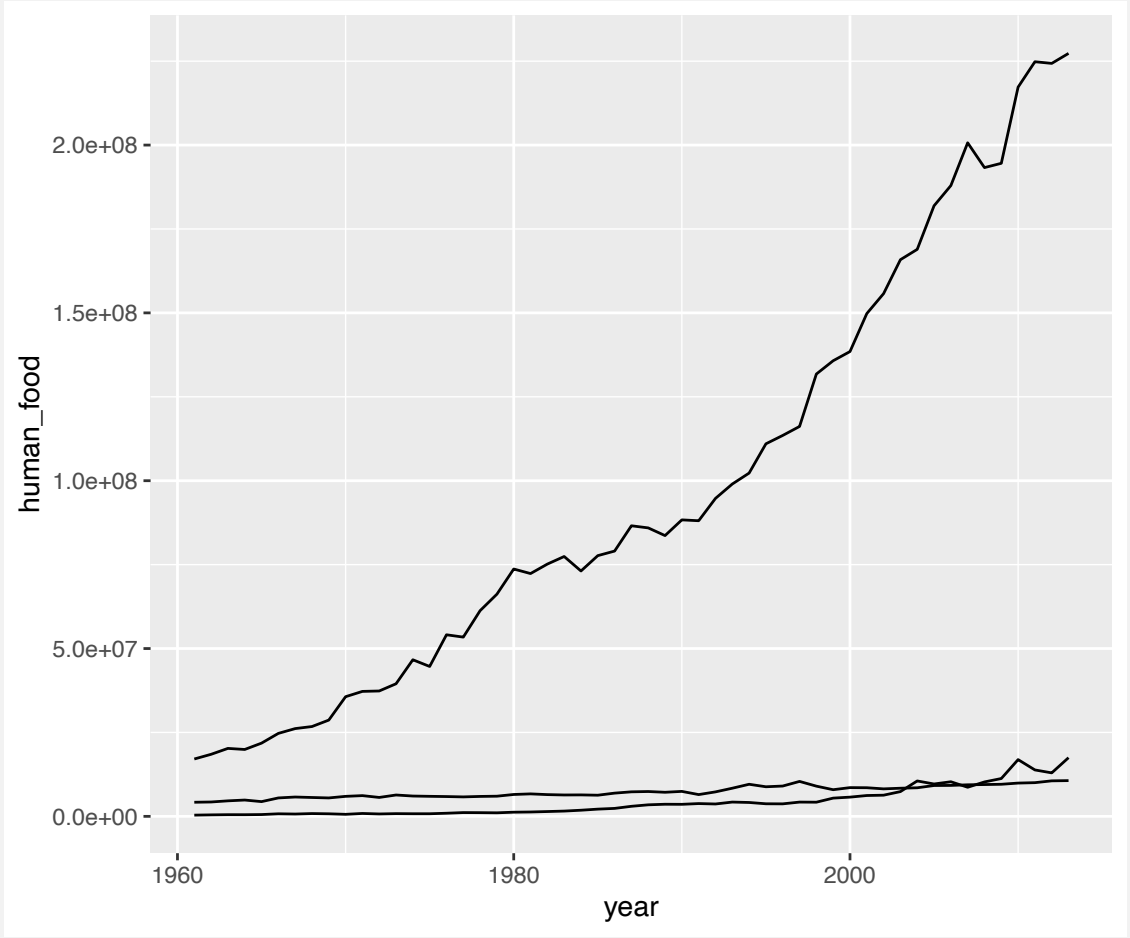
```
ggplot(soy, aes(x= year))+
  geom_line(aes(y= processed))+
  geom_line(aes(y= human_food))+
  geom_line(aes(y= animal_feed))
```



TO ADD ADDITIONAL VARIABLES, ADD MORE GEOM LAYERS

```
ggplot(soy, aes(x= year))+
  geom_line(aes(y= processed))+
  geom_line(aes(y= human_food))+
  geom_line(aes(y= animal_feed))
```

This is difficult to differentiate between the values. However, we can add colour and labels to improve this.



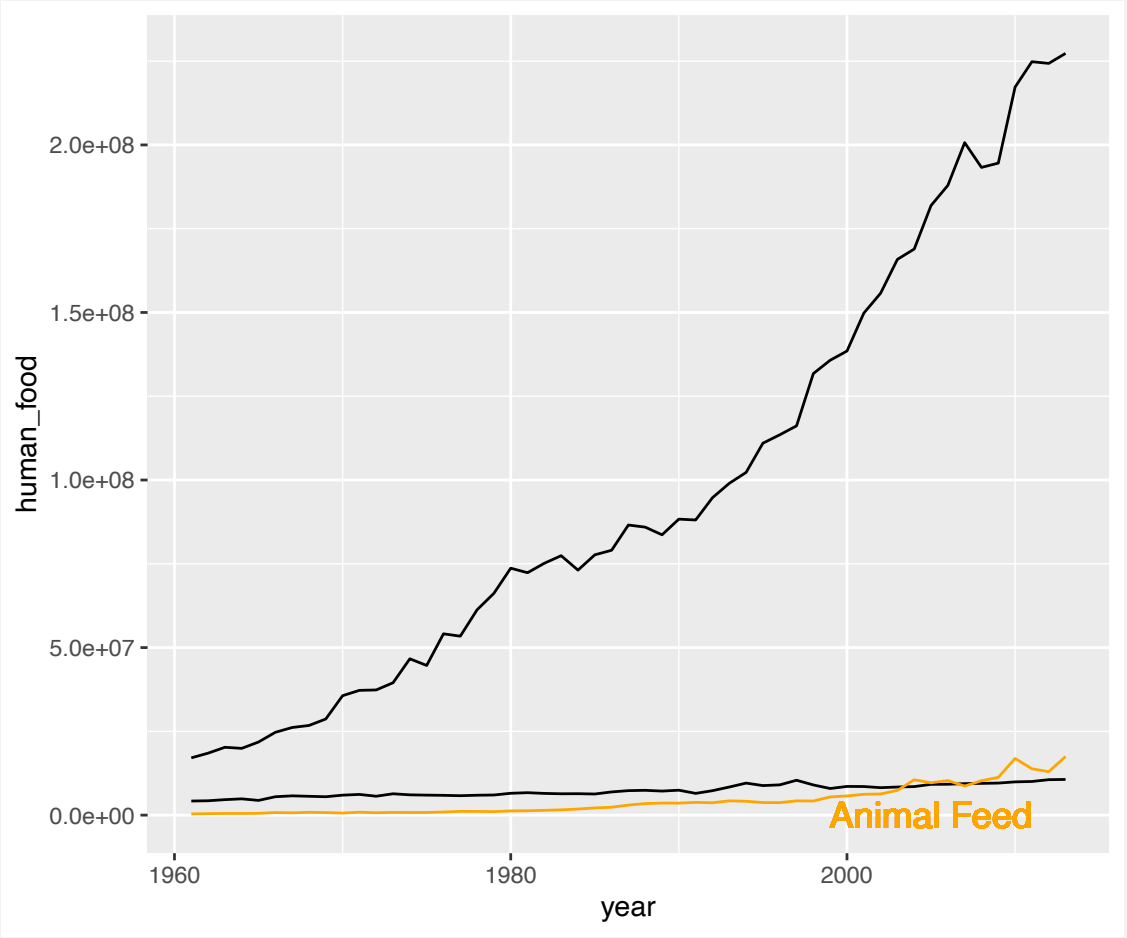
COLOUR AND LABEL LINES

To colour, outside of aes in geom_function add in colour.

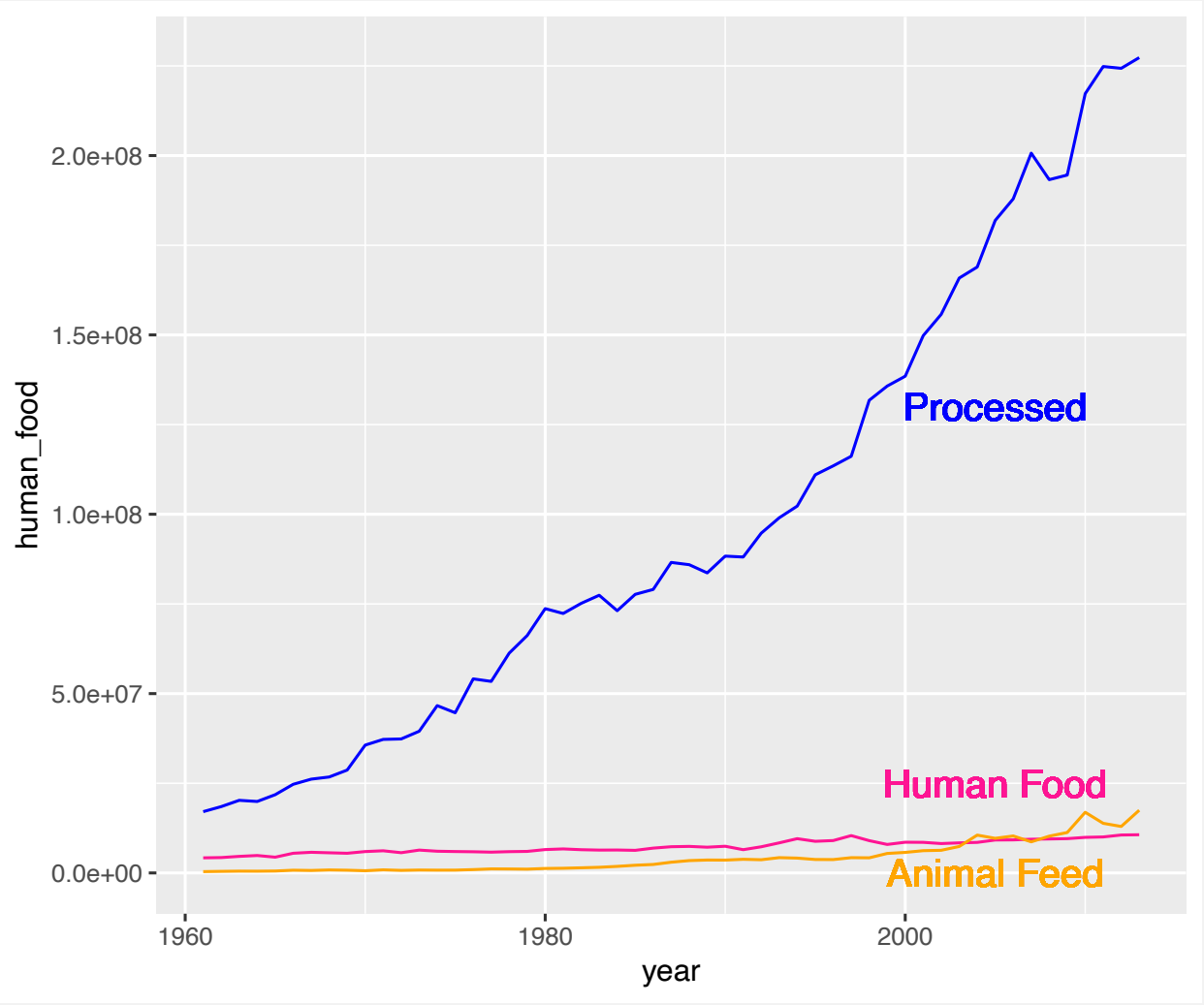
```
geom_line(aes(y= animal_feed), col = “orange”)
```

To add label, use geom_text().

```
geom_text(aes(2005, 1000, label = "Animal Feed"), col="orange", size=5)
```



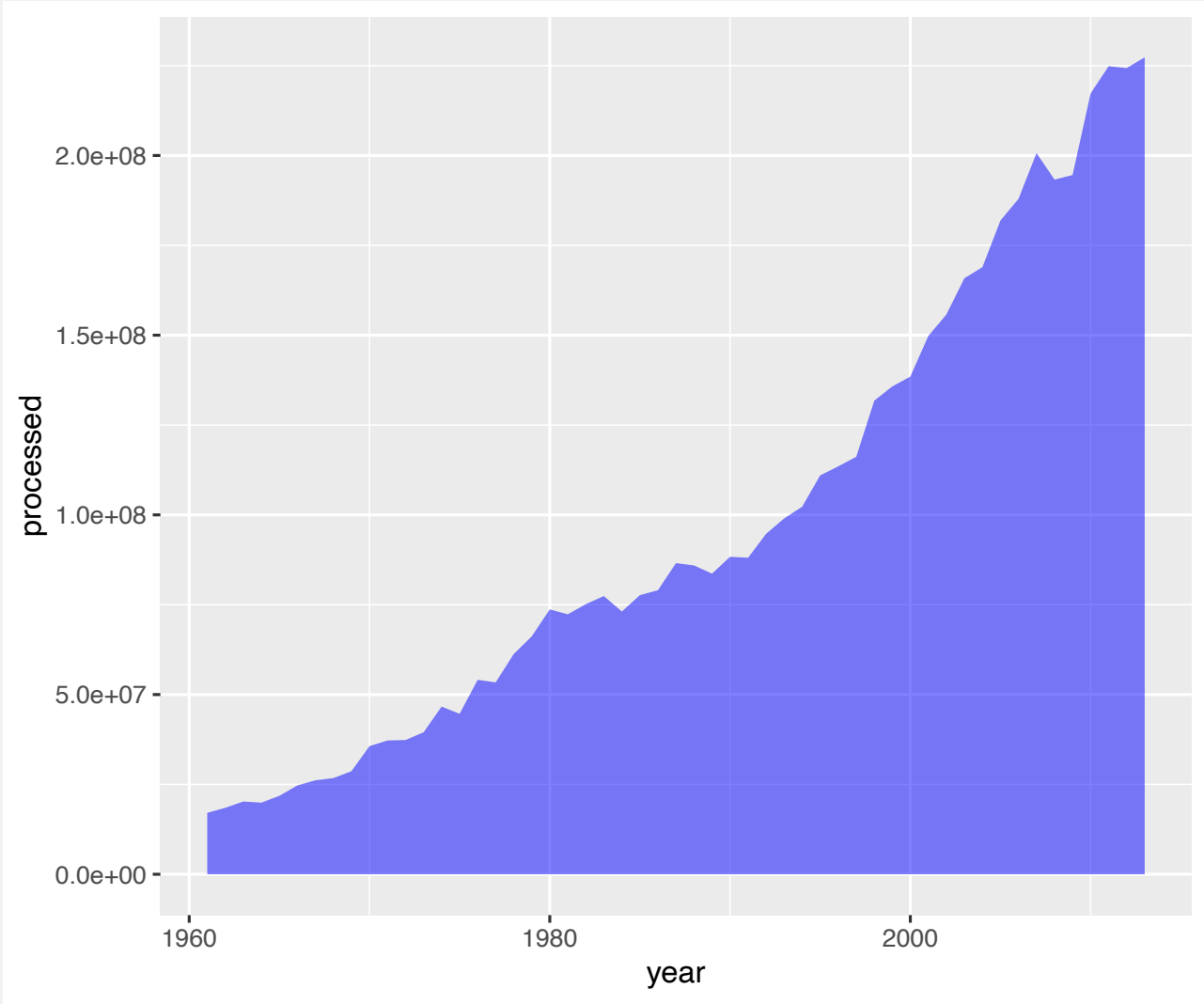
```
ggplot(df, aes(x=year))+
  geom_line(aes(y=human_food), col =
"deeppink")+
  geom_line(aes(y=animal_feed), col="orange")+
  geom_line(aes(y=processed), col="blue")+
  geom_text(aes(2005, 1000, label = "Animal
Feed"), col="orange",  size=5)+
  geom_text(aes(2005, 25000000, label = "Human
Food"), col="deeppink",  size=5)+
  geom_text(aes(2005, 130000000, label =
"Processed"), col="blue",  size=5)
```



AREA PLOTS

- Can use geom_area instead to show filled region.

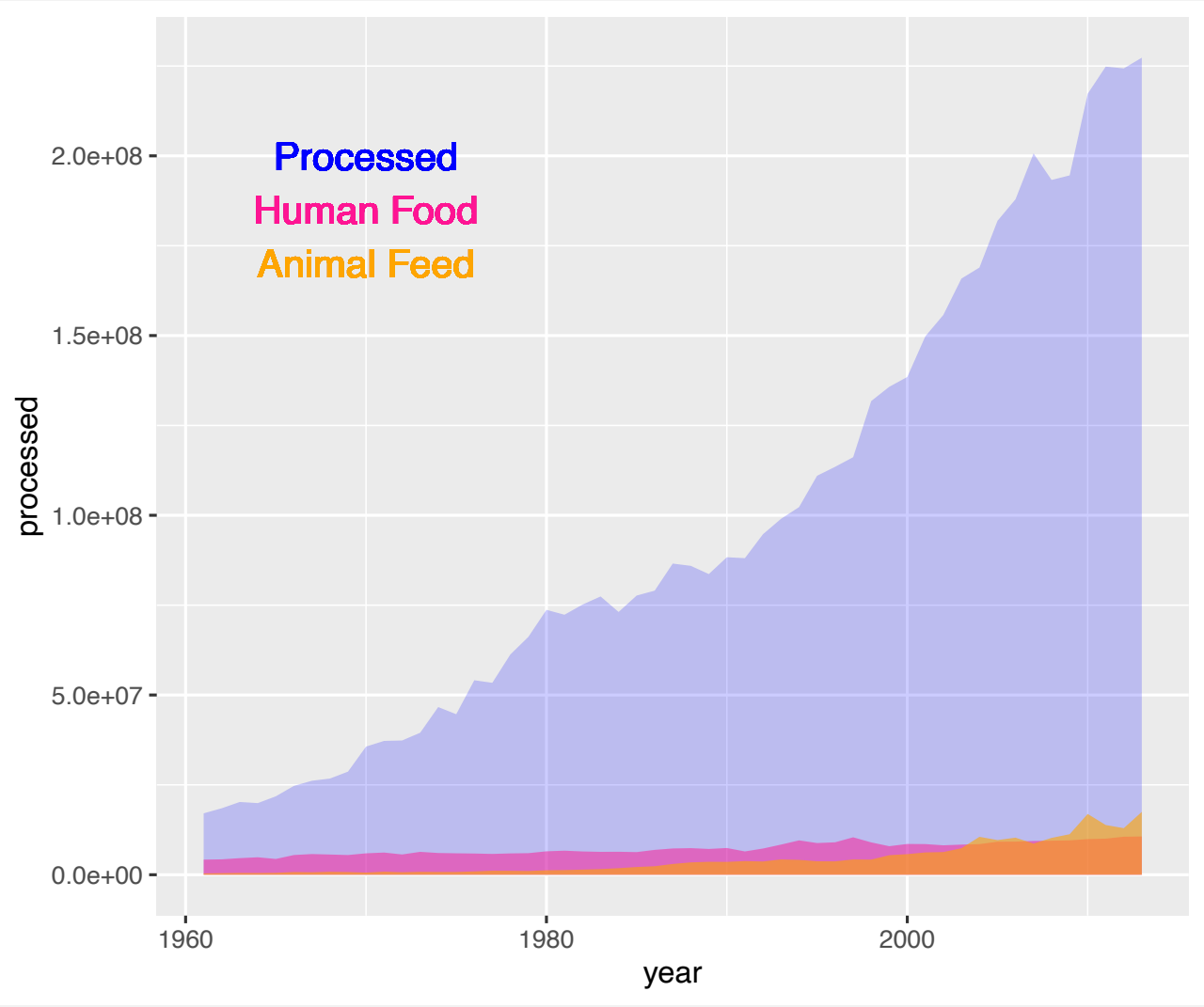
```
ggplot(df, aes(x=year))+
  geom_area(aes(y=processed), fill = "blue", alpha = 0.5)
```



AREA PLOTS

- Can use geom_area instead to show filled region.

```
ggplot(df, aes(x=year))+
  geom_area(aes(y=processed), fill = "blue", alpha = 0.2)+
  geom_area(aes(y=human_food), fill = "deeppink", alpha = 0.5)+
  geom_area(aes(y=animal_feed), fill = "orange", alpha = 0.6)+
  geom_text(aes(1970,170000000,label = "Animal Feed"),col="orange",size=5)+
  geom_text(aes(1970,185000000,label = "Human Food"),col="deeppink",size=5)+
  geom_text(aes(1970,200000000,label = "Processed"),col="blue",size=5)
```



TIDY AXES

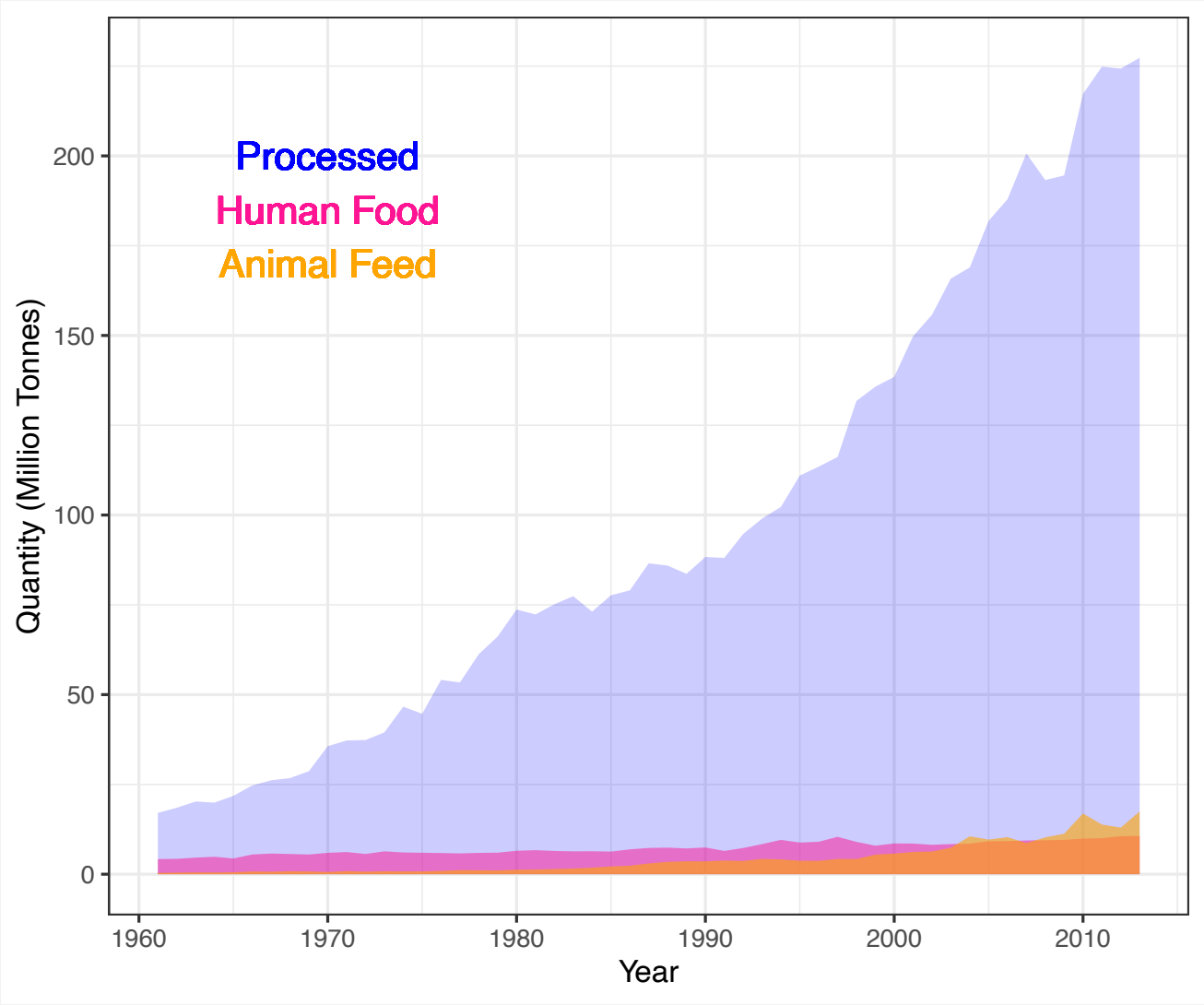
- x-axis increments are a bit sparse can use `scale_x_continuous()` to fix this
  - Can also use this to change axis labels, add limits to axis
  - Can also use `scale_x_*` fuction to transform axis. For example log10 make logarithmic scale, reverse, or replace x with y and change y axis.

```
scale_x_continuous(breaks = c(1960, 1970, 1980, 1990, 2000, 2010))
```

- Use `theme_bw()` to make white background of plot
- Change scale of y axis, do this in aes() but could also mutate dataset

FINAL PLOT

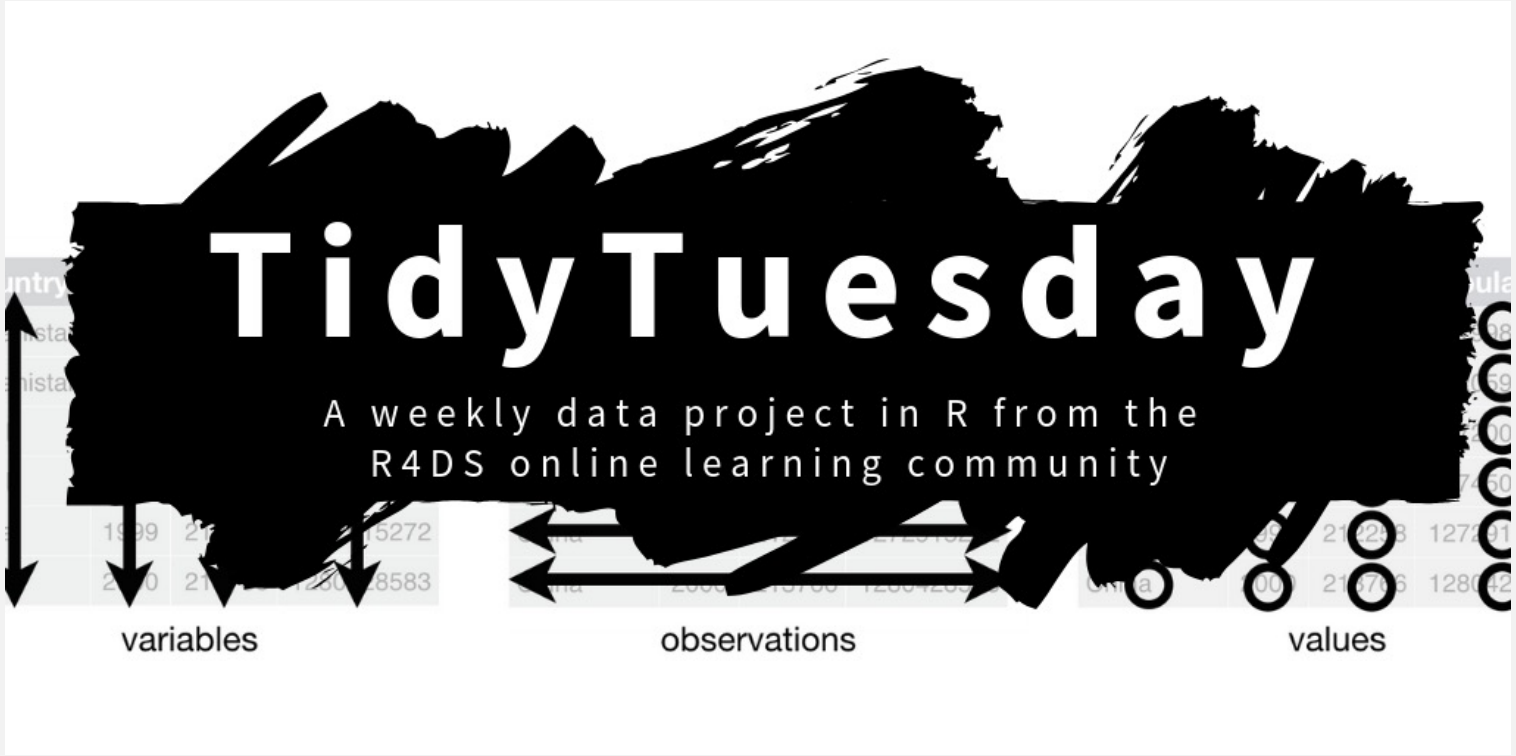
```
ggplot(df, aes(x=year))+
  geom_area(aes(y=processed/1e6), fill = "blue", alpha
= 0.2)+
  geom_area(aes(y=human_food/1e6), fill = "deeppink",
alpha = 0.5)+
  geom_area(aes(y=animal_feed/1e6), fill = "orange",
alpha = 0.6)+
  scale_x_continuous(breaks = c(1960, 1970, 1980, 1990,
2000, 2010))+
  geom_text(aes(1970,170,label = "Animal
Feed"),col="orange",size=5)+
  geom_text(aes(1970,185,label = "Human
Food"),col="deeppink",size=5)+
  geom_text(aes(1970,200,label =
"Processed"),col="blue",size=5)+
  labs(x= "Year", y="Quantity (Million Tonnes)")+
  theme_bw()
```



SUMMARY

- Used geom_line and geom_area to show time series data
- Filtered new dataset
- Utilised colour and alpha options
- Add annotation to plot using geom_text
- Edited axis using scale_x_*

TIDYTUESDAY



Datasets and the plots people have made from them
https://shiny.rstudio.com/gallery/tidy-tuesday.html

https://github.com/rfordatascience/tidytuesday

Data Visualization with ggplot2 : : CHEAT SHEET

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and geoms—visual marks that represent data points.

To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.

Complete the template below to build a graph.

required

```
ggplot (data = <DATA>) +
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
 stat = <STAT> , position = <POSITION>) +
<COORDINATE_FUNCTION>+
<FACET_FUNCTION>+
<SCALE_FUNCTION>+
<THEME_FUNCTION>
```

Not required, sensible defaults supplied

ggplot(data = mpg, **aes**(x = cty, y = hwy)) Begins a plot that you finish by adding layers to. Add one geom function per layer.

aesthetic mappings · data · geom

qplot(x = cty, y = hwy, data = mpg, geom = "point") Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

last_plot() Returns the last plot

ggsave("plot.png", width = 5, height = 5) Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))

a + geom_blank()
(Useful for expanding limits)

b + geom_curve(aes(yend = lat + 1, xend=long+1,curvature=z)) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(lineend="butt", linejoin="round", linemitre=1)
x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(group = group))
x, y, alpha, color, fill, group, linetype, size

b + geom_rect(aes(xmin = long, ymin=lat, xmax= long + 1, ymax = lat + 1)) - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

a + geom_ribbon(aes(ymin=unemploy - 900, ymax=unemploy + 900)) - x, ymax, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

b + geom_abline(aes(intercept=0, slope=1))
b + geom_hline(aes(yintercept = lat))
b + geom_vline(aes(xintercept = long))

b + geom_segment(aes(yend=lat+1, xend=long+1))
b + geom_spoke(aes(angle = 1:1155, radius = 1))

ONE VARIABLE continuous

c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)

c + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian")
x, y, alpha, color, fill, group, linetype, size, weight

c + geom_dotplot()
x, y, alpha, color, fill

c + geom_freqpoly() x, y, alpha, color, group, linetype, size

c + geom_histogram(binwidth = 5) x, y, alpha, color, fill, linetype, size, weight

c2 + geom_qq(aes(sample = hwy)) x, y, alpha, color, fill, linetype, size, weight

discrete

d <- ggplot(mpg, aes(fl))

d + geom_bar()
x, alpha, color, fill, linetype, size, weight

TWO VARIABLES

continuous x , continuous y

e <- ggplot(mpg, aes(cty, hwy))

e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE) x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

e + geom_jitter(height = 2, width = 2)
x, y, alpha, color, fill, shape, size

e + geom_point(), x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile(), x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl"), x, y, alpha, color, linetype, size

e + geom_smooth(method = lm), x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE), x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

discrete x , continuous y

f <- ggplot(mpg, aes(class, hwy))

f + geom_col(), x, y, alpha, color, fill, group, linetype, size

f + geom_boxplot(), x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

f + geom_dotplot(binaxis = "y", stackdir = "center"), x, y, alpha, color, fill, group

f + geom_violin(scale = "area"), x, y, alpha, color, fill, group, linetype, size, weight

discrete x , discrete y

g <- ggplot(diamonds, aes(cut, color))

g + geom_count(), x, y, alpha, color, fill, shape, size, stroke

continuous bivariate distribution

h <- ggplot(diamonds, aes(carat, price))

h + geom_bin2d(binwidth = c(0.25, 500))
x, y, alpha, color, fill, linetype, size, weight

h + geom_density2d()
x, y, alpha, colour, group, linetype, size

h + geom_hex()
x, y, alpha, colour, fill, size

continuous function

i <- ggplot(economics, aes(date, unemploy))

i + geom_area()
x, y, alpha, color, fill, linetype, size

i + geom_line()
x, y, alpha, color, group, linetype, size

i + geom_step(direction = "hv")
x, y, alpha, color, group, linetype, size

visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))

j + geom_crossbar(fatten = 2)
x, y, ymax, ymin, alpha, color, fill, group, linetype, size

j + geom_errorbar(), x, ymax, ymin, alpha, color, group, linetype, size, width (also **geom_errorbarh()**)

j + geom_linerange()
x, ymin, ymax, alpha, color, group, linetype, size

j + geom_pointrange()
x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps

data <- data.frame(murder = USArrests$Murder, state = tolower(rownames(USArrests)))
map <- map_data("state")
k <- ggplot(data, aes(fill = murder))

k + geom_map(aes(map_id = state), map = map) **+ expand_limits**(x = map$long, y = map$lat), map_id, alpha, color, fill, linetype, size

THREE VARIABLES

seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))l <- ggplot(seals, aes(long, lat))

l + geom_contour(aes(z = z))
x, y, z, alpha, colour, group, linetype, size, weight

l + geom_raster(aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE)
x, y, alpha, fill

l + geom_tile(aes(fill = z)), x, y, alpha, color, fill, linetype, size, width

Cheat sheets available online

https://www.rstudio.com/resources/cheatsheets/