ggplot2 Course, AufLand Conference

Eduard Szöcs

26. November 2015

Institute for Environmental Sciences - University of Koblenz-Landau

INSTITUTE FOR
ENVIRONMENTAL SCIENCES

Intro

Short Intro: Eduard Szöcs

- PhD student @Quantitative Landscape Ecology
- Environmental Sciences & Ecotoxicology
- Research:
  - Statistical Ecology - Eco(toxico)logical Statistics
  - Effects and distribution of pesticides in freshwaters
- R-Programming:
  - R-user for 6years
  - Author/Co-Author of 3 CRAN packages (taxize, webchem, rspear)
  - Other packages on github (restax, esmisc)
  - Minor contribitions to other pkgs (e.g. vegan)

`edild.github.io` @EduardSzoecs

Short Intro: Gunnar Oehmichen

- Diploma in Environmental Sciences
- Diploma Thesis @Quantitative Landscape Ecology
- Topics: Ecology, Renewable Energies and Conservation
- Graduate Assistant @Environmental Sciences Lab: Renewable Energies
  - Automated display of Spatial Information on Renewable Energies in a WebGIS
- R-Programming:
  - R-User for 5 Years
  - Assistant in several Summerschools (@UFZ-Leipzig, @Landau)

Course Outline

- Short intro & course organisation, Software preparation
- An introduction to ggplot2.
  - Basic ggplot2 usage
  - Customization of your plots
  - Free time for specific Q.
- After 4 hours of ggplot2 you should be an expert ;)

Course material: `https://github.com/EDiLD/r_landau_2015`

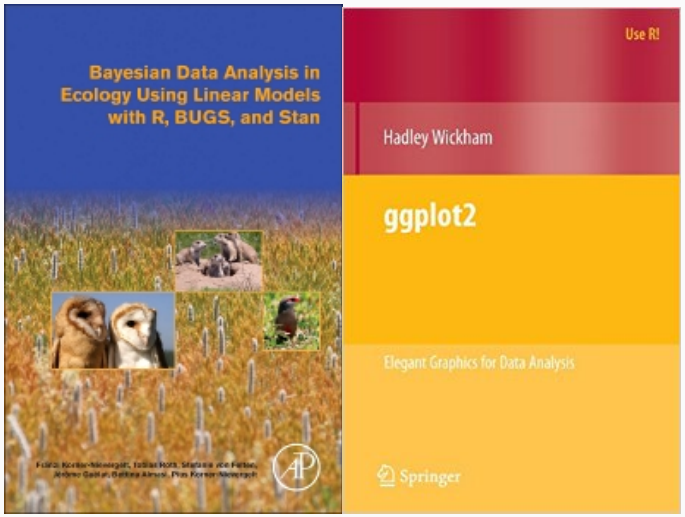
Preliminaries

- Download the course repository.
- No formal R knowledge required to follow.
- (Install RStudio and packages needed).
- Just open the '.R' files in RStudio and execute the script line by line: 'CTRL + ENTER'

Datasets

We will use 2 data sets in this course:

1. Frog monitoring in Swiss (Slides).
2. Diamond prices (Exercises).



Tip:

Don't buy the old ggplot2 book - a new one is coming soon...

```
library(blmeco)
data(frogs)
head(frogs)
```

```
##   count1 count2 elevation year fish waterarea vegetation pondid      x
## 1     16     12       380 2013    0      2500          1 400301 649750
## 2      0      0       565 2009    0       300          1 400411 647350
## 3      0      0       430 2012    0       450          1 400603 650250
## 4      0      0       500 2012    0       348          1 400608 649400
## 5      0      0       450 2012    0       200          1 400701 646700
## 6      0      0       560 2010    0        42          1 400802 646500
##        y
## 1 248850
## 2 255750
## 3 244600
## 4 243850
## 5 240750
## 6 253650
```

```
frogs$fish <- factor(frogs$fish)
frogs$vegetation <- factor(frogs$vegetation)
```

Questions:

What influences frog abundance? Can we predict frog abundance (ala SDM)?

```
library(ggplot2)
data(diamonds)
head(diamonds)
```

```
##   carat       cut color clarity depth table price    x    y    z
## 1  0.23     Ideal     E     SI2  61.5    55   326 3.95 3.98 2.43
## 2  0.21   Premium     E     SI1  59.8    61   326 3.89 3.84 2.31
## 3  0.23      Good     E     VS1  56.9    65   327 4.05 4.07 2.31
## 4  0.29   Premium     I     VS2  62.4    58   334 4.20 4.23 2.63
## 5  0.31      Good     J     SI2  63.3    58   335 4.34 4.35 2.75
## 6  0.24 Very Good     J    VVS2  62.8    57   336 3.94 3.96 2.48
```

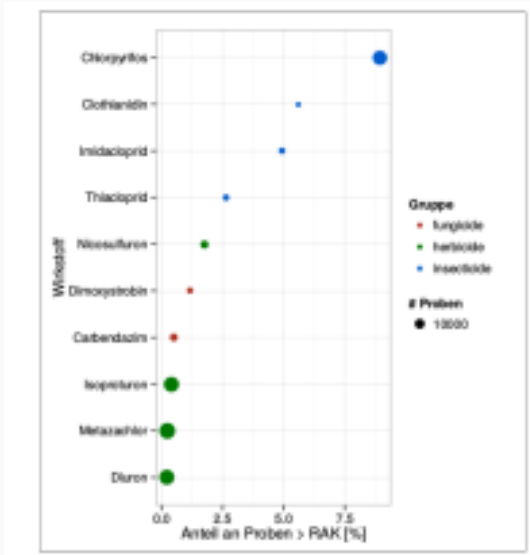
```
?diamonds
```

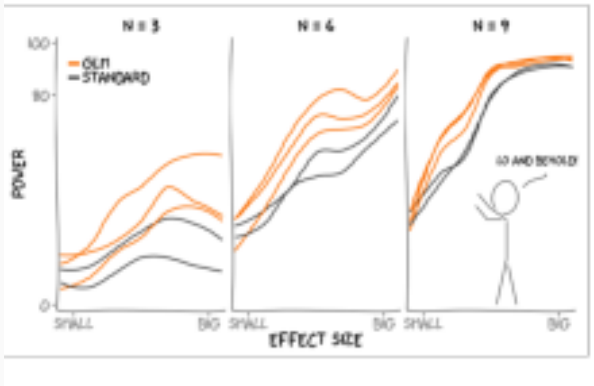
Questions:

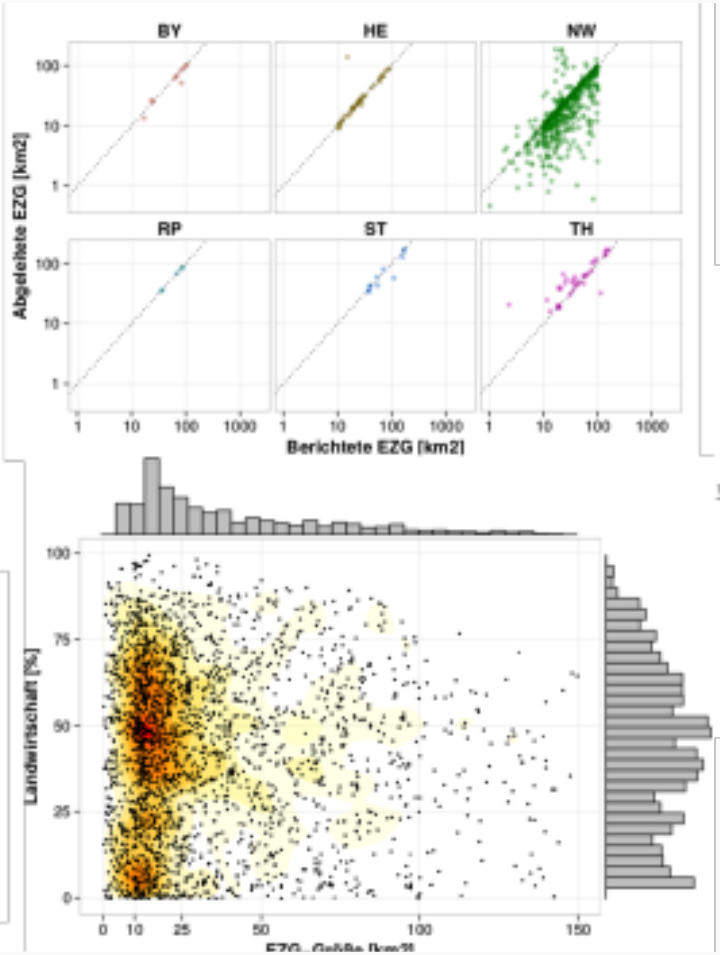
What factors are crucial for the diamond prices?

ggplot2 basics

Recent plots I created with ggplot2

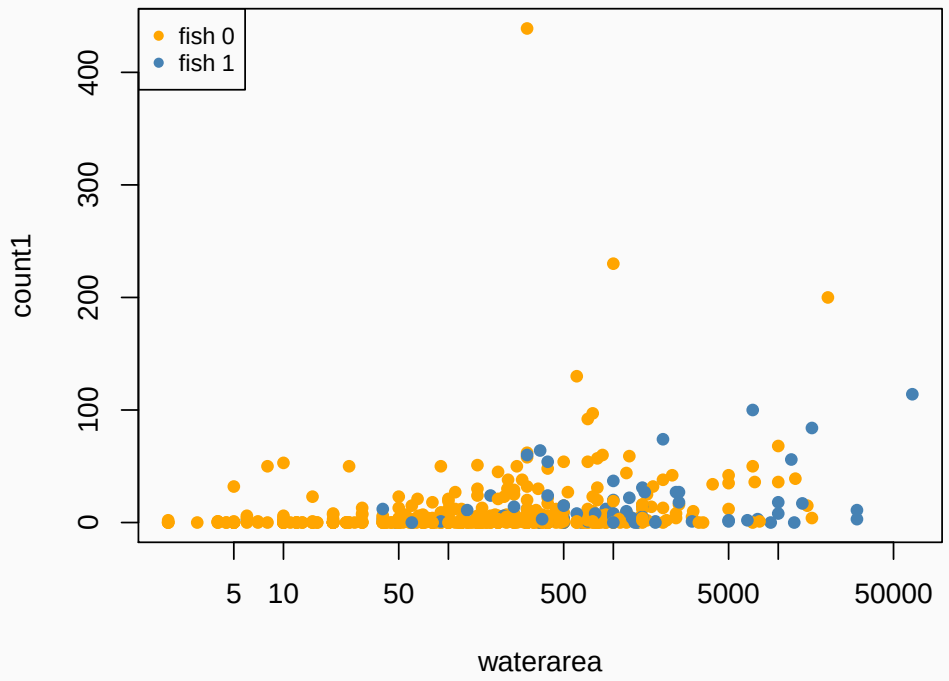






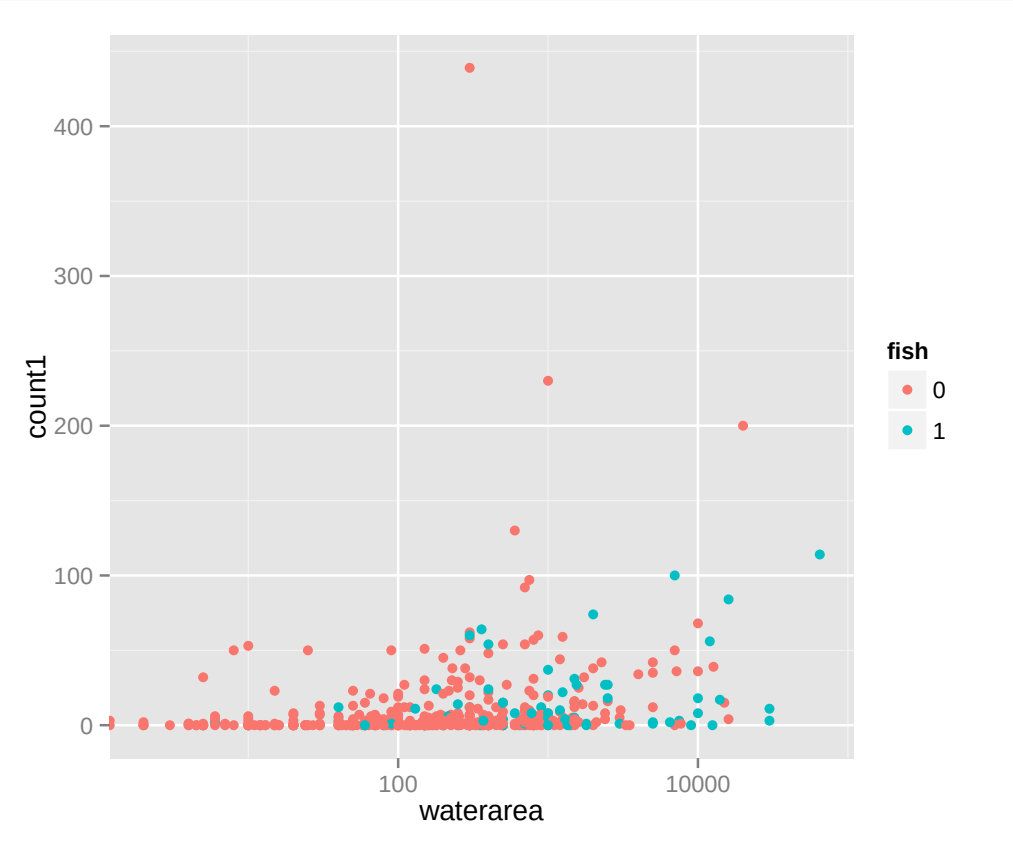
With base R

```
cols <- c('orange', 'steelblue')
plot(count1 ~ waterarea, data = frogs, type = 'n', log = 'x')
with(frogs, points(waterarea, count1, col = cols[fish], pch = 16))
legend('topleft', legend = c('fish 0', 'fish 1'), pch = 16,
       col = cols,
       cex = 0.8)
```



With ggplot2

```
ggplot(frogs) +
  geom_point(aes(x = waterarea, y = count1, col = fish)) +
  scale_x_log10()
```



Why ggplot2?

base graphics

- more work / code
- legends, colors?
- defaults ok
- multipanel plots?
- can do anything
- have to know the *tricks*

ggplot2

- quick-and-dirty and complex
- Automatic legends, colors!
- nice defaults
- easy multipanel plots
- restrictions (e.g. 2nd y-axis)
- intuitive syntax

Why ggplot2?

base graphics

- more work / code
- legends, colors?
- defaults ok
- multipanel plots?
- can do anything
- have to know the *tricks*

ggplot2

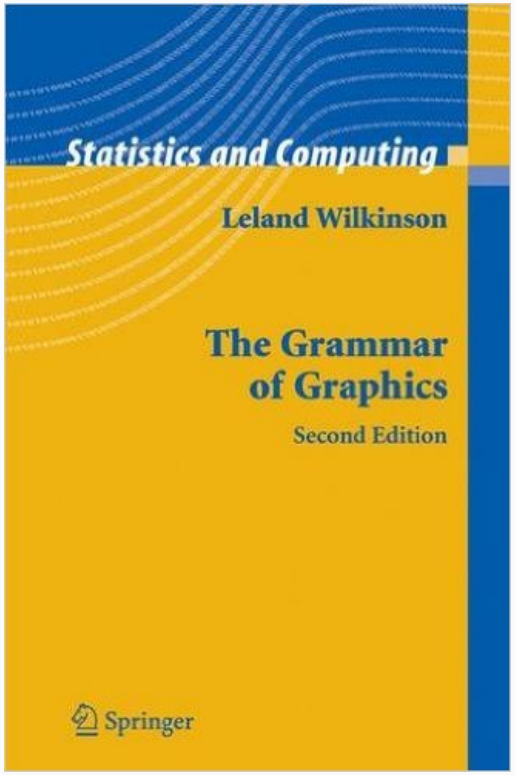
- quick-and-dirty and complex
- Automatic legends, colors!
- nice defaults
- easy multipanel plots
- restrictions (e.g. 2nd y-axis)
- intuitive syntax

Tip:

If a plot is too complicated to draw with ggplot2, reconsider if it's a good representation of your data!

What is ggplot2?

- A graphic system for R
- gg = Grammar of Graphics (Wilkinson 2005)
- *Grammar*: A set of components that define a sentence
- ggplot defines a grammar to create plots
- Consistent, intuitive, easy to learn



```
ggplot(frogs) +
  geom_point(
    aes(x = waterarea, y = count1, col = fish)) +
  scale_x_log10() +
  theme_bw()
```

ggplot() The main function. Can specify the data set and variables **globally**.

geom A geometric object: `geom_point`, `geom_line`, `geom_text`, `geom_violin`, ...

aes aesthetics: maps a variable to the properties of a geom: `shape`, `color`, `fill`, `linetype`, transparency (`alpha`), ...

scale How are data mapped visually (`log`, `date`, `colors`, `sizes`, ...)

theme The general plot apperance (`background`, `axes`, `labels`, ...)

```
ggplot(frogs) +
  geom_point(
    aes(x = waterarea, y = count1, col = fish)) +
  scale_x_log10() +
  theme_bw()
```

ggplot() The main function. Can specify the data set and variables **globally**.

geom A geometric object: `geom_point`, `geom_line`, `geom_text`, `geom_violin`, ...

aes aesthetics: maps a variable to the properties of a geom: `shape`, `color`, `fill`, `linetype`, transparency (`alpha`), ...

scale How are data mapped visually (`log`, `date`, `colors`, `sizes`, ...)

theme The general plot apperance (`background`, `axes`, `labels`, ...)

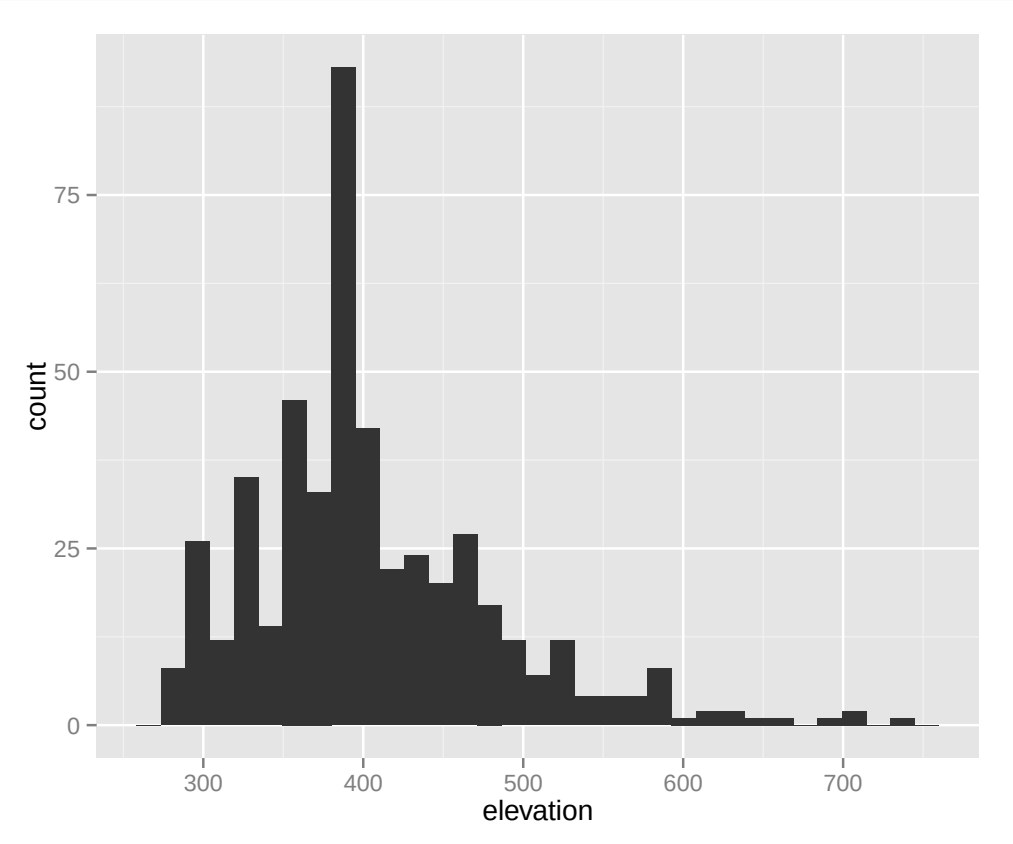
Tip:

Never use `qplot()` (=quick plot)! - You won't learn the grammar...

Univariate Data

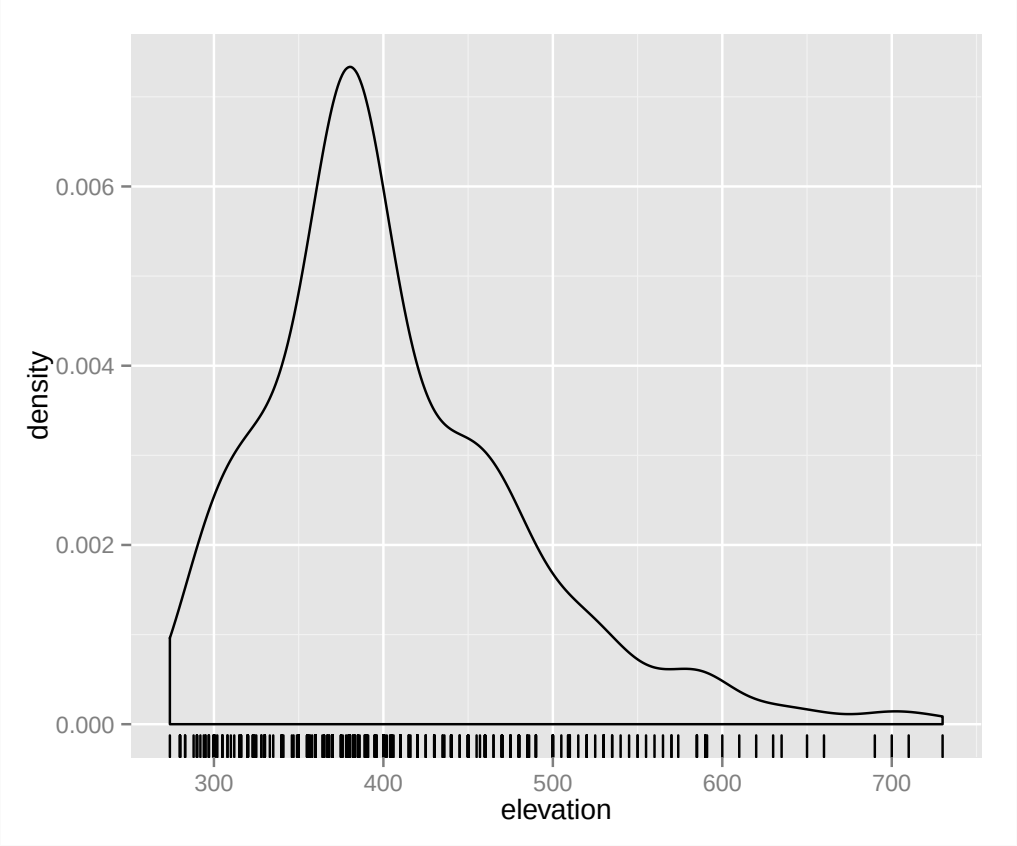
```
ggplot(frogs,                          # use the frogs
       aes(x = elevation))+   # take the variable 'elevation' from the dataset)
  geom_histogram()                     # display a histogramm

## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust
this.
```



We can add multiple geoms of the same variable to the plot

```
ggplot(frogs, aes(x = elevation)) +    # plot the 'elevation' from the frogs data
  geom_density() +                        # display a density
  geom_rug()                              # display a rug
```

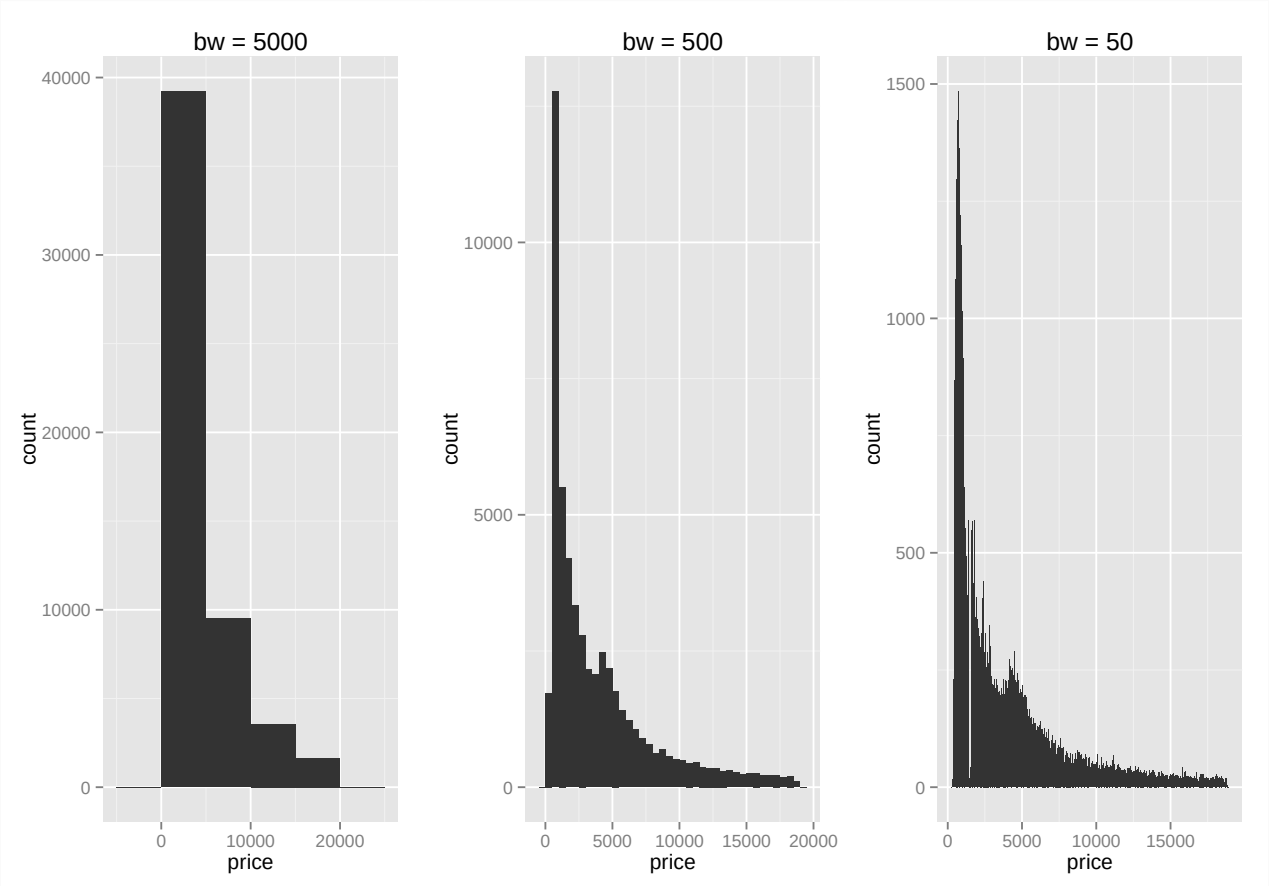


Exercise 1:

Plot a histogramm of diamond prices at different binwidths (50, 500, 5000).

How does this affect the preception of the data?

Exercise

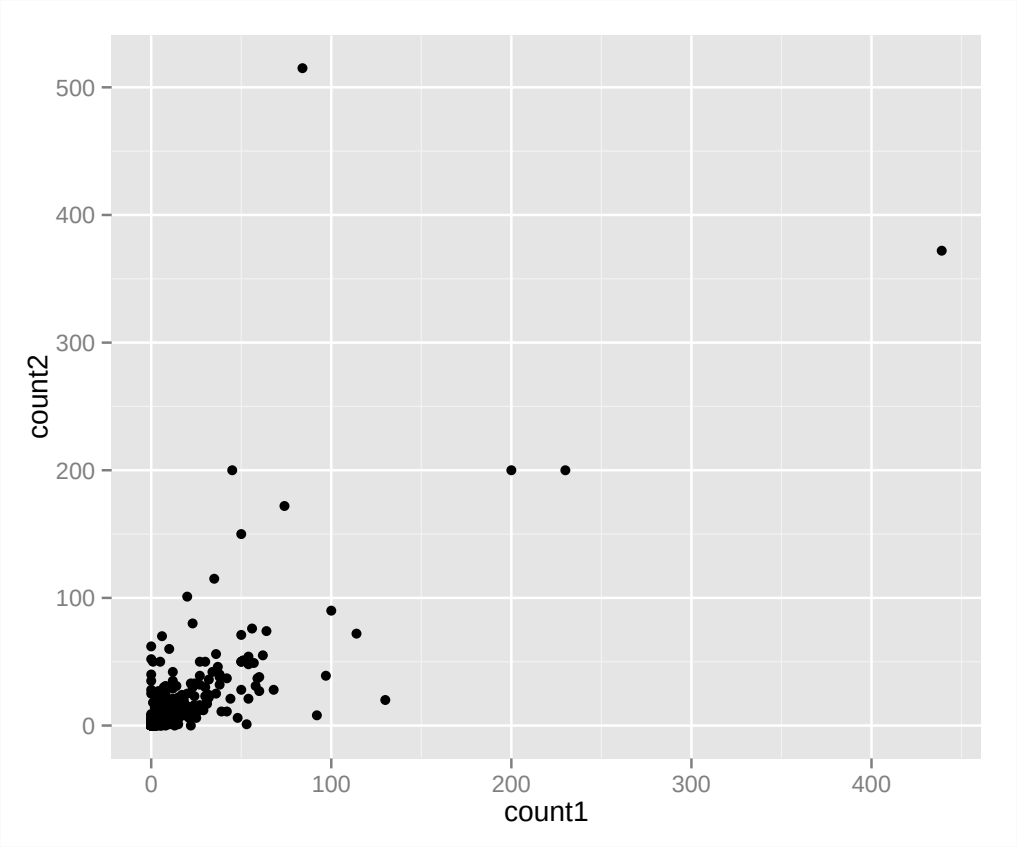

bw = 5000
bw = 500
bw = 50
count
price
0
10000
20000
30000
40000
5000
15000
500
1000
1500

Bivariate data
(cont. x cont.)

GEOM: POINT (=CONT. X CONT.)

The mother of all plots.

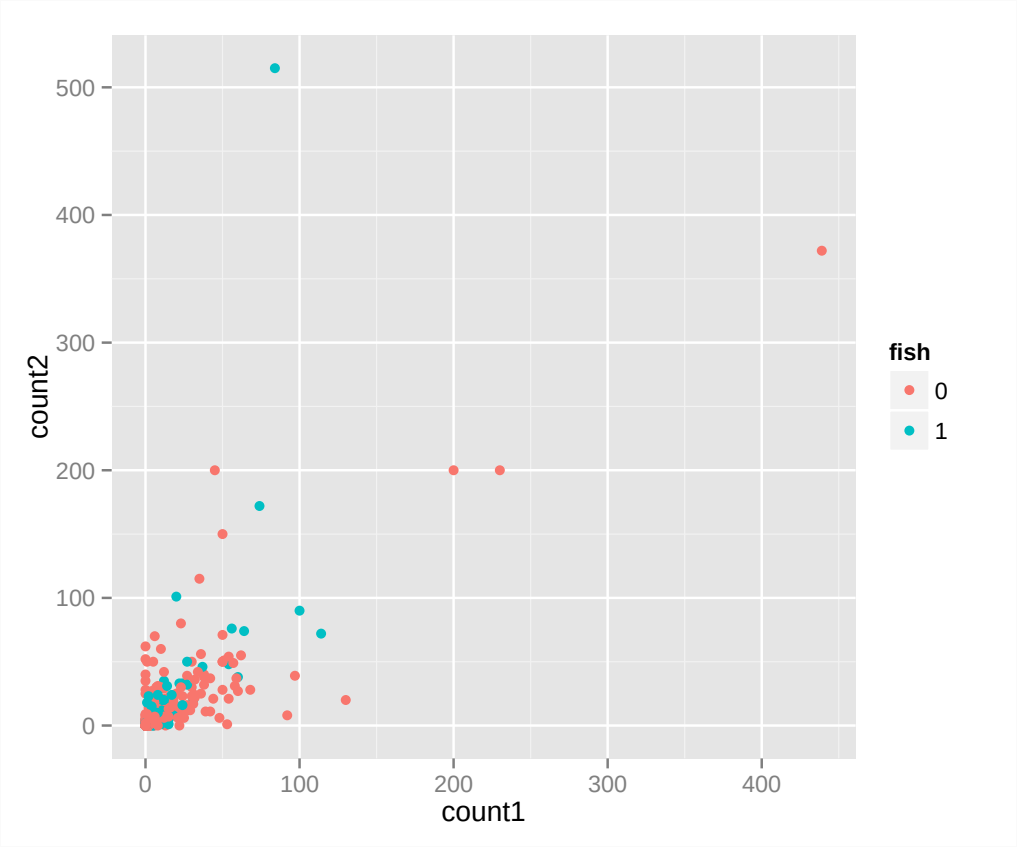
```
ggplot(frogs, aes(x = count1, y = count2)) +
  geom_point()
```



AESTHETIC: COLOR (DISCRETE)

Differentiate between fish and no-fish:

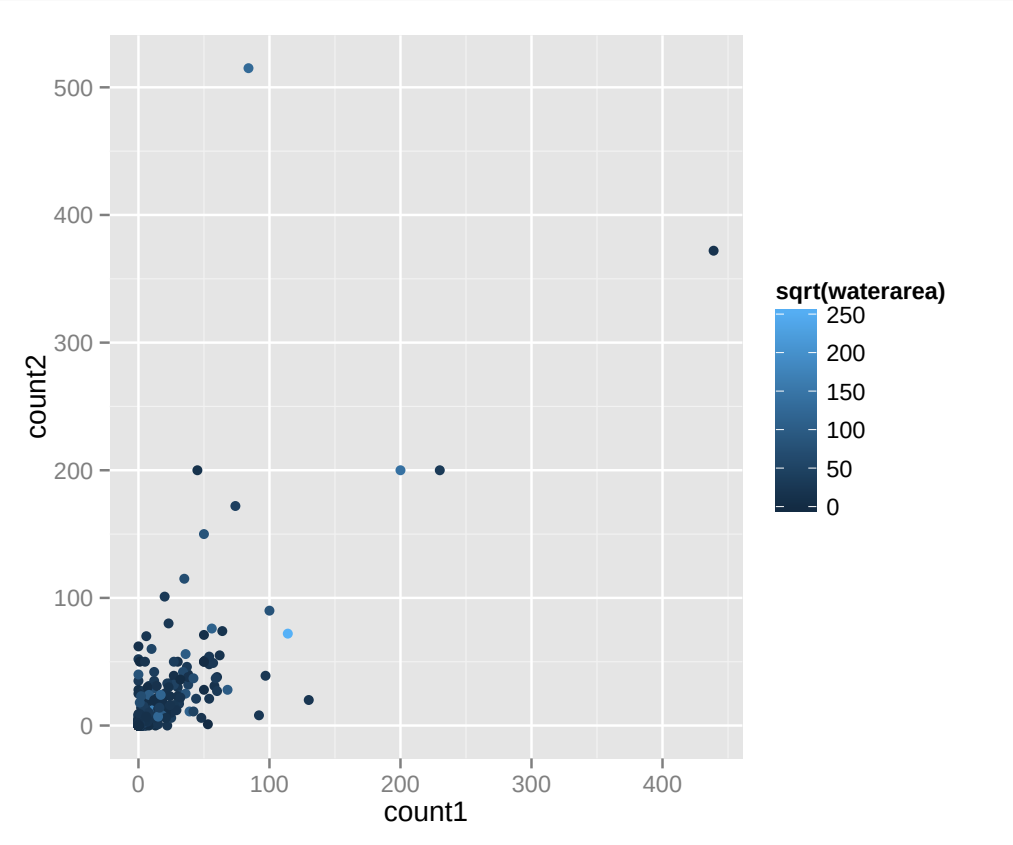
```
ggplot(frogs, aes(x = count1, y = count2, col = fish)) +
  geom_point()
```



AESTHETIC: COLOR (CONTINUOUS)

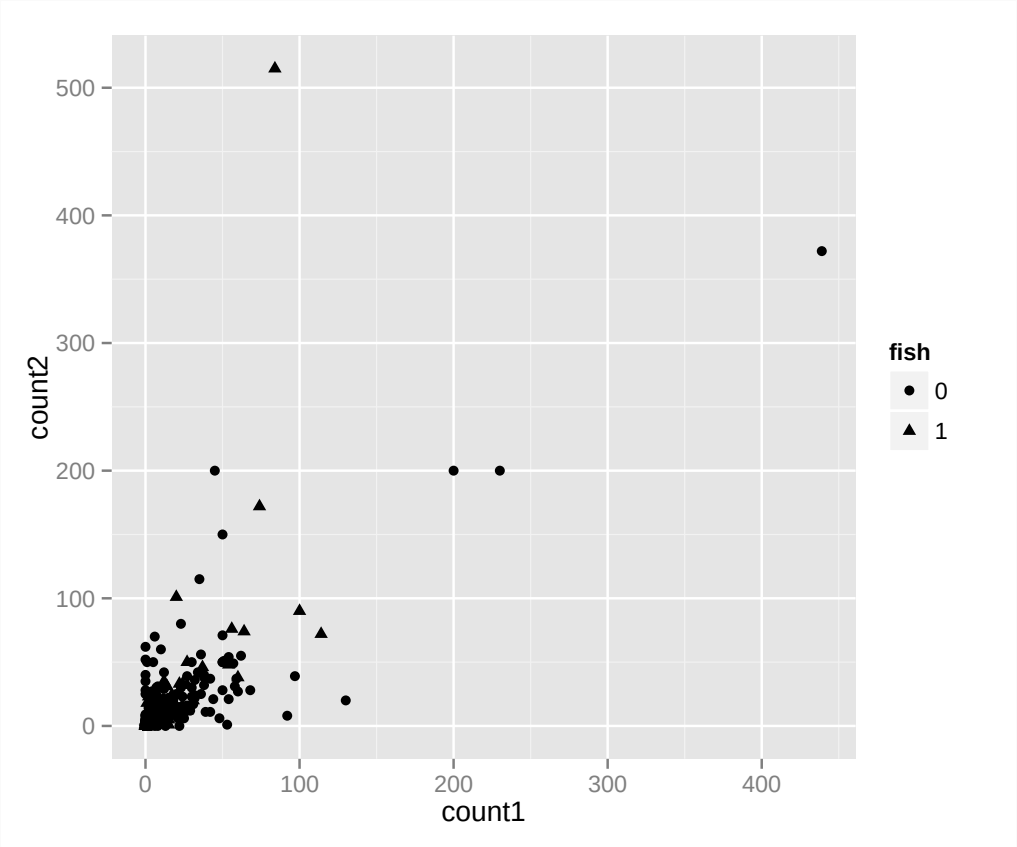
Continuous color scale:

```
ggplot(frogs, aes(x = count1, y = count2, col = sqrt(waterarea))) +
  geom_point()
```



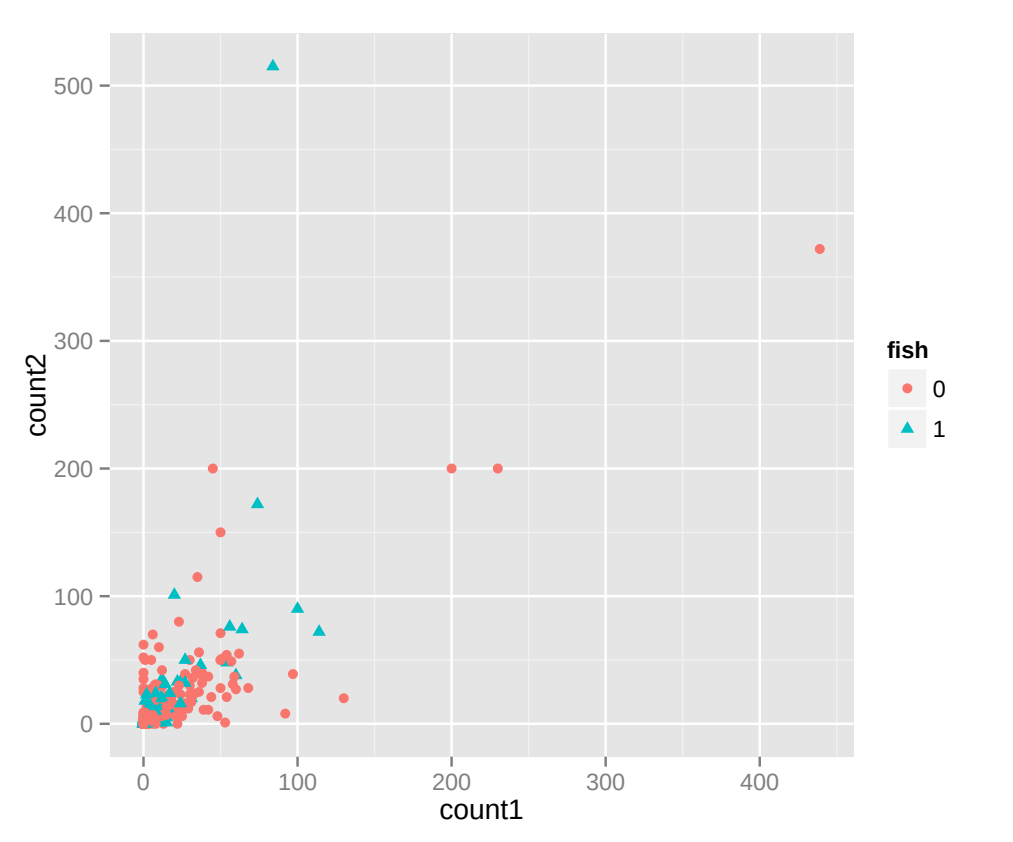
Can also differentiate by shape:

```
ggplot(frogs, aes(x = count1, y = count2, shape = fish)) +
  geom_point()
```



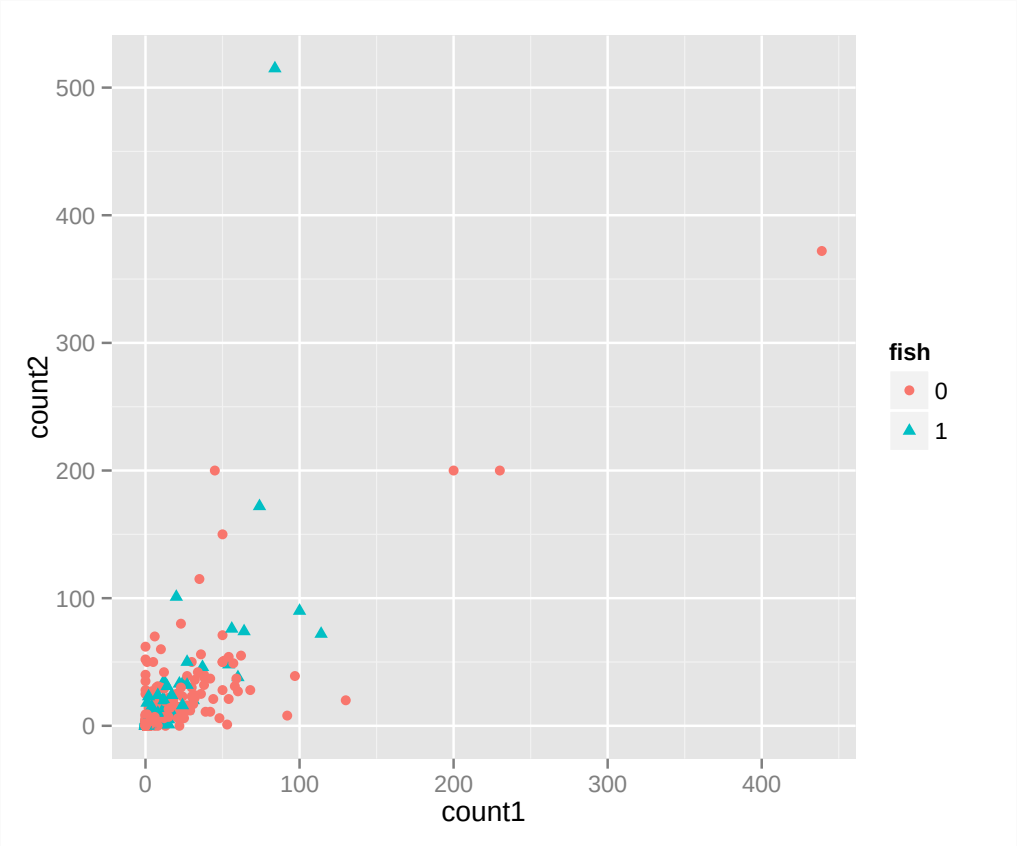
Can also differentiate by shape:

```
ggplot(frogs, aes(x = count1, y = count2, shape = fish, col = fish)) +
  geom_point()
```



Can also differentiate by shape:

```
ggplot(frogs, aes(x = count1, y = count2, shape = fish, col = fish)) +
  geom_point()
```

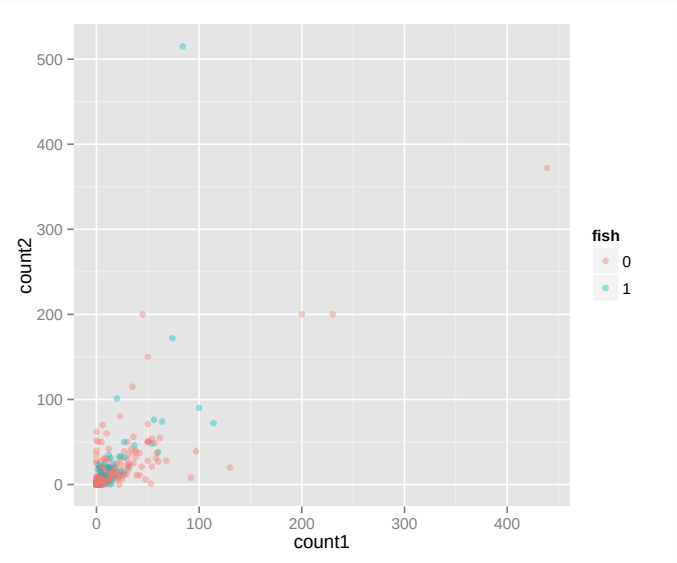


Tip:

Do not use redundant aesthetics.

AESTHETICS: ALPHA (=TRANSPARENCY)

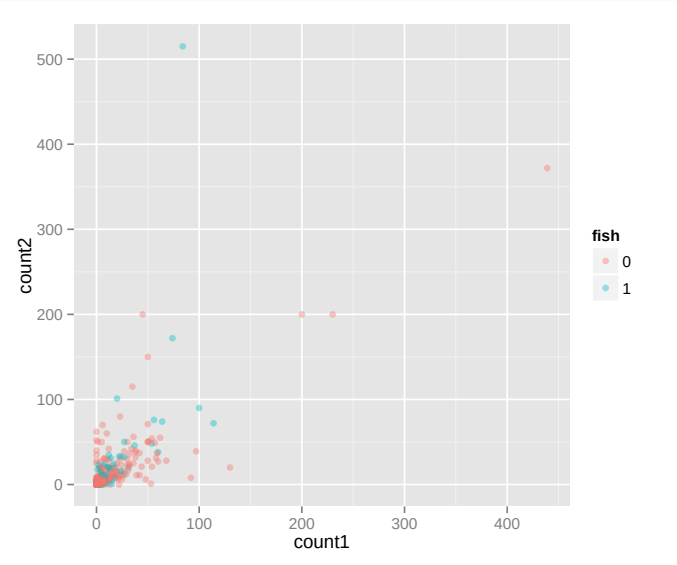
```
ggplot(frogs) +
  geom_point(aes(x = count1, y = count2, col = fish), alpha = 0.4)
```



Q: alpha is not within aes(), why?

AESTHETICS: ALPHA (=TRANSPARENCY)

```
ggplot(frogs) +
  geom_point(aes(x = count1, y = count2, col = fish), alpha = 0.4)
```

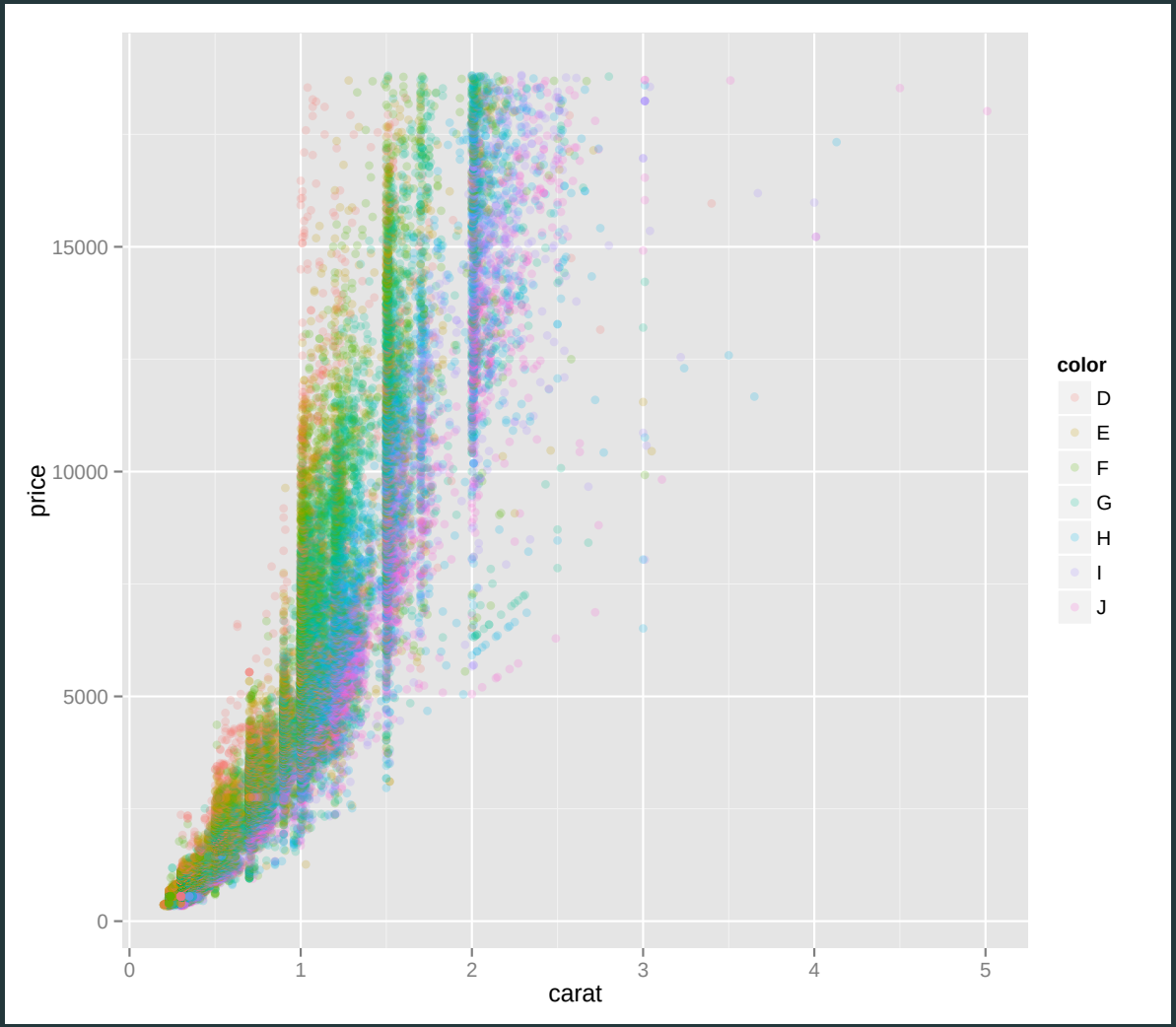


Q: alpha is not within aes(), why?

A:

Within `aes()` the aesthetic changes with the specified variable (e.g. `col = fish`). Outside of `aes` we set it to a specific value.

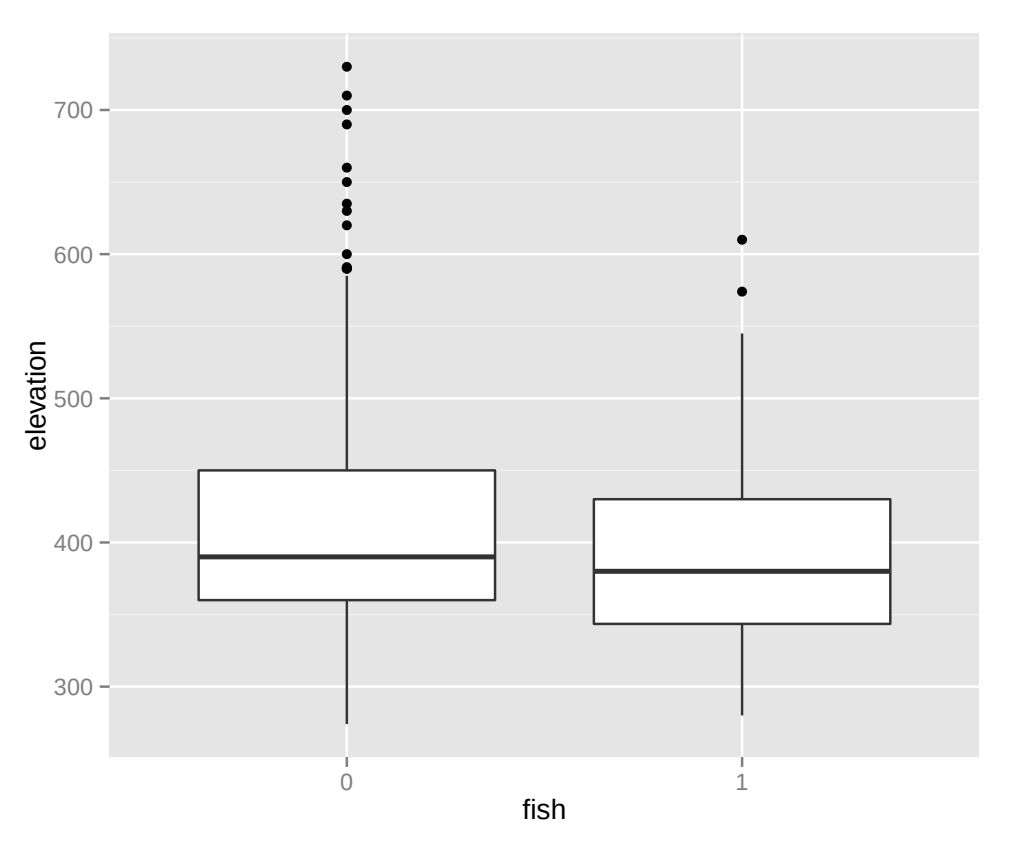
Exercise 2:
Create this plot:



Bivariate data
(discrete x continuous)

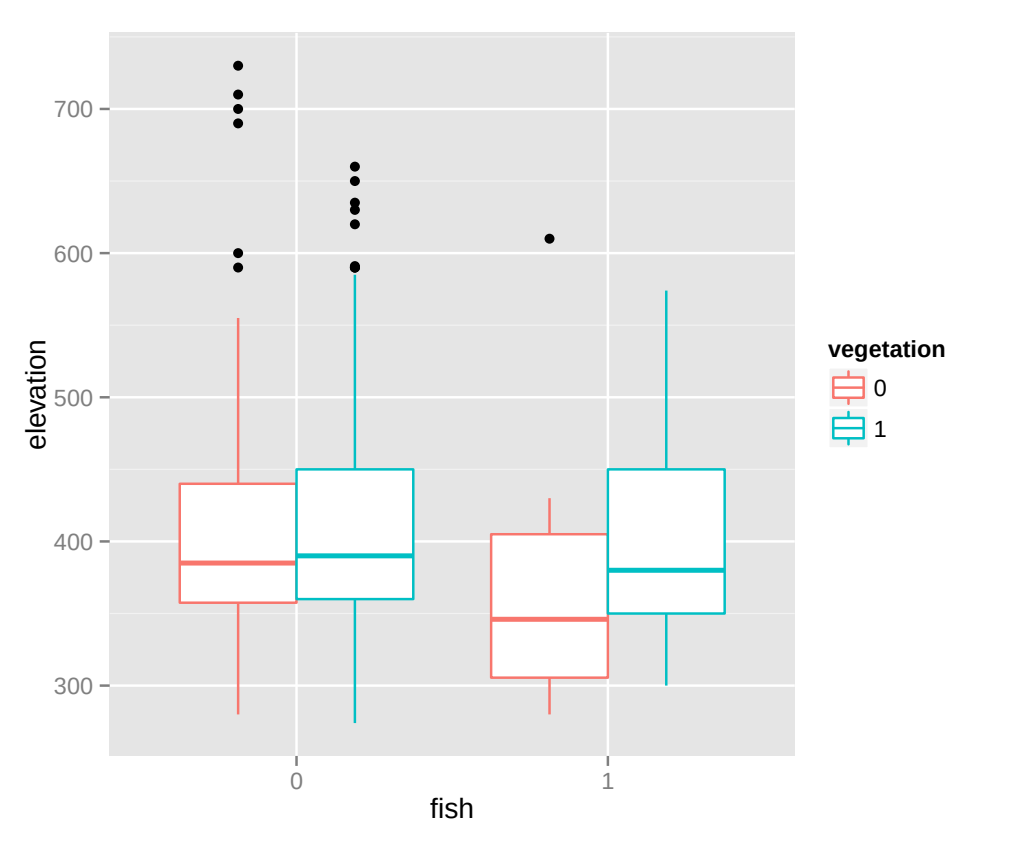
- More then 45 years (Tukey 1970)
- For discrete scales (here fish)

```
ggplot(frogs, aes(x = fish, y = elevation)) +
  geom_boxplot()
```



AESTHETIC: COLOR (DISCRETE)

```
ggplot(frogs, aes(x = fish, y = elevation, col = vegetation)) +
  geom_boxplot()
```

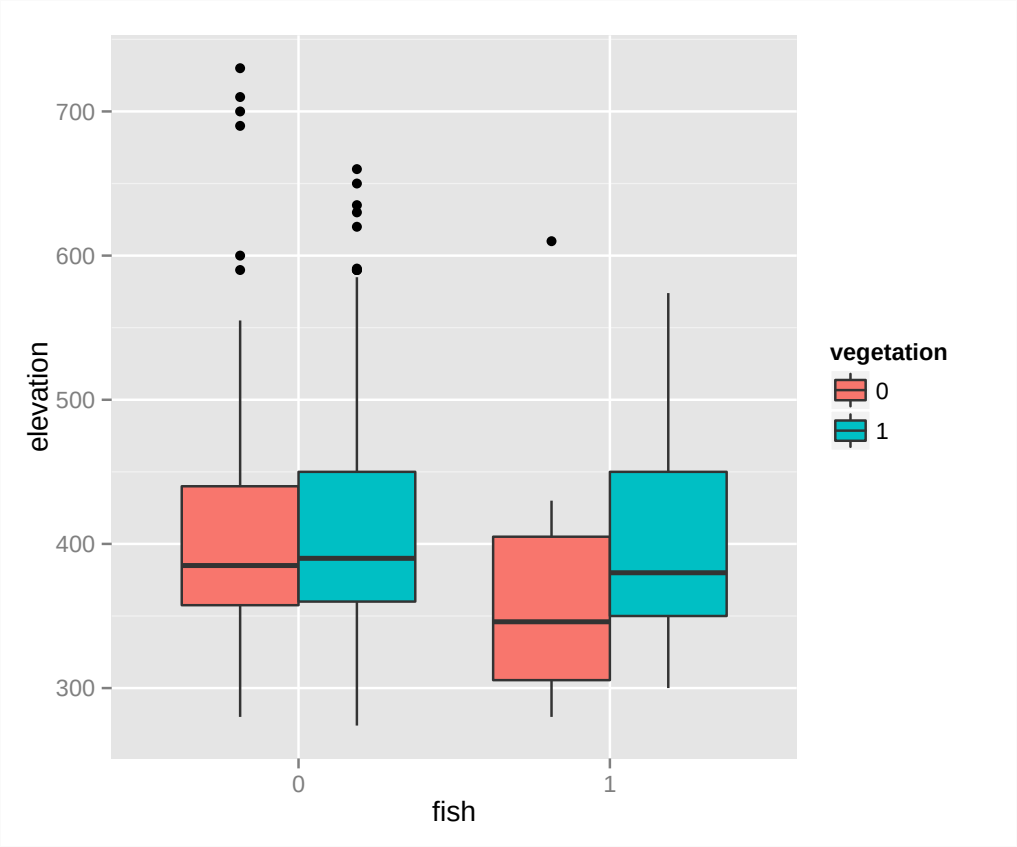


Tip:

Nice for factorial designs (e.g. easily spot interactions).

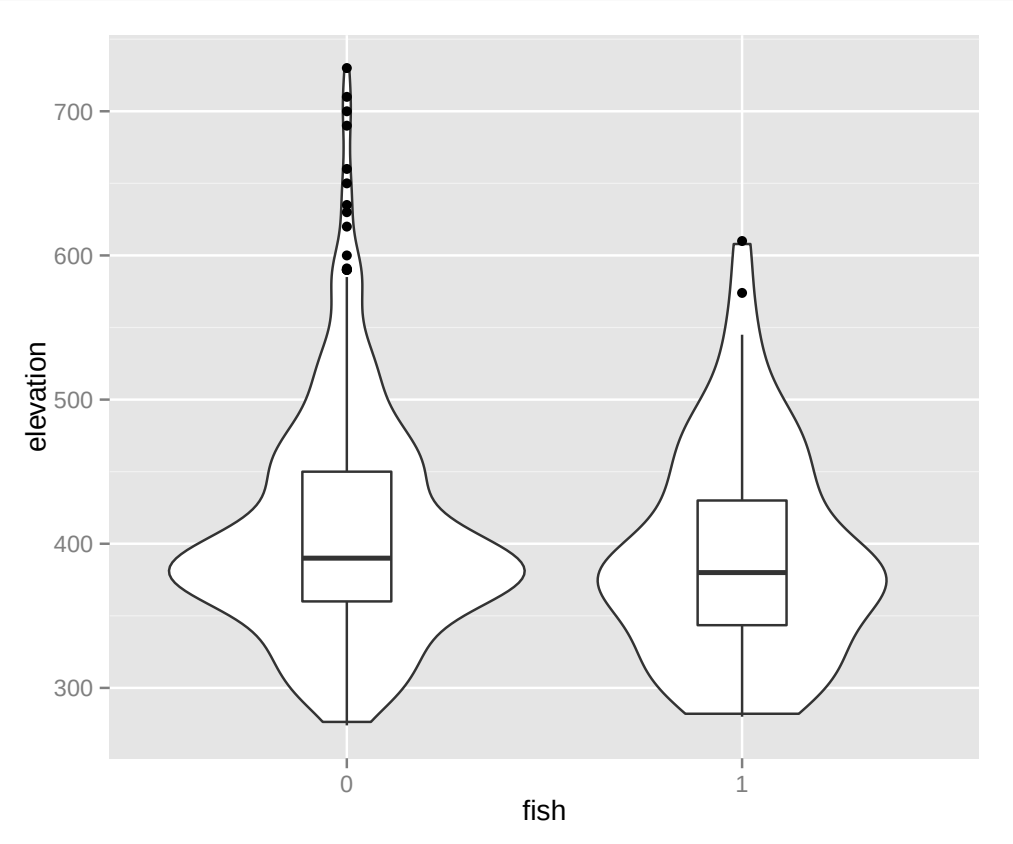
AESTHETIC: FILL (DISCRETE)

```
ggplot(frogs, aes(x = fish, y = elevation, fill = vegetation)) +
  geom_boxplot()
```



The Violin plot is a powerful alterative/addition:

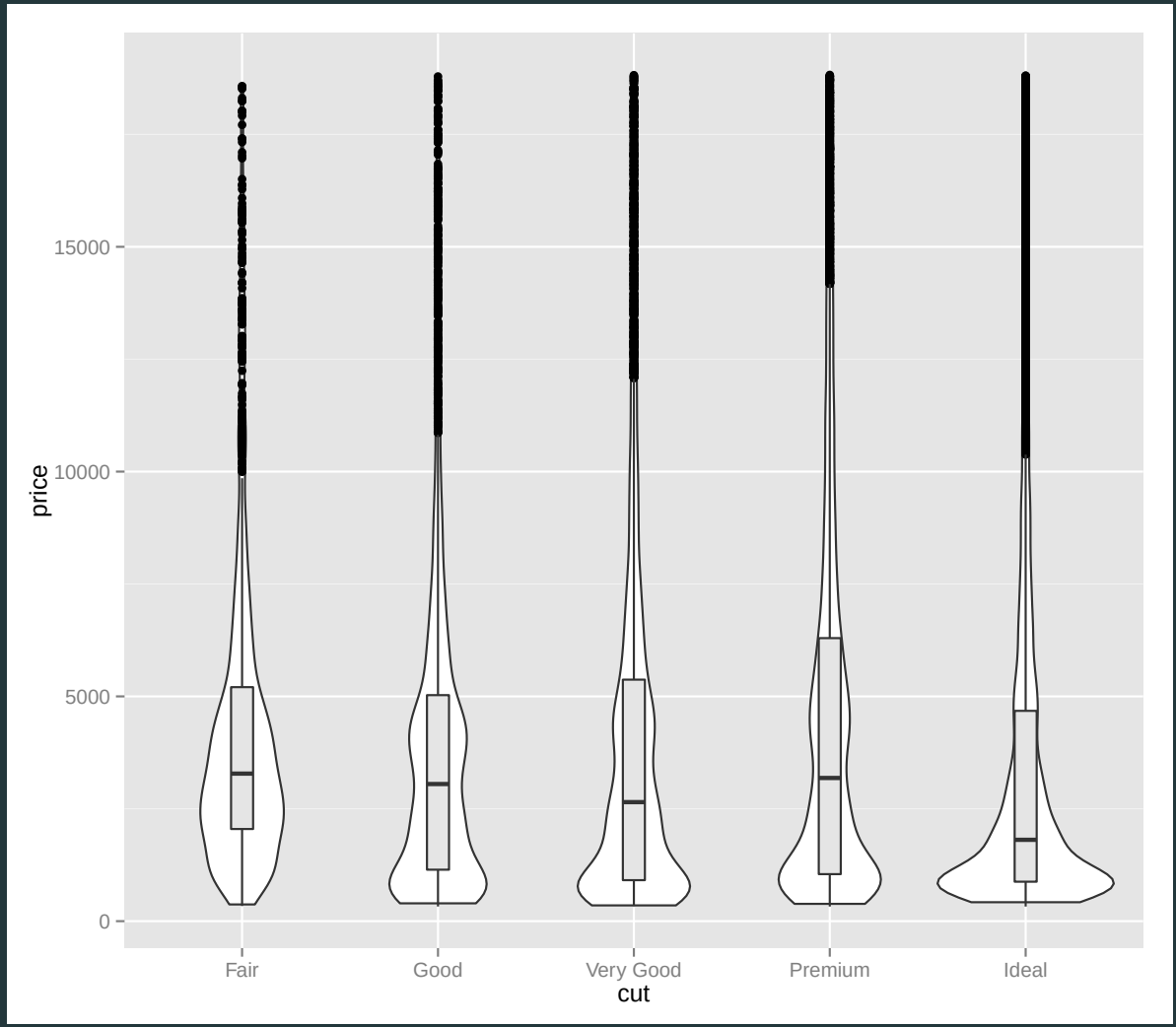
```
ggplot(frogs, aes(x = fish, y = elevation)) +
  geom_violin() +
  geom_boxplot(width = 0.3)
```



Tip:

The order of geoms is important! Uppermost layer is the last geom specified.

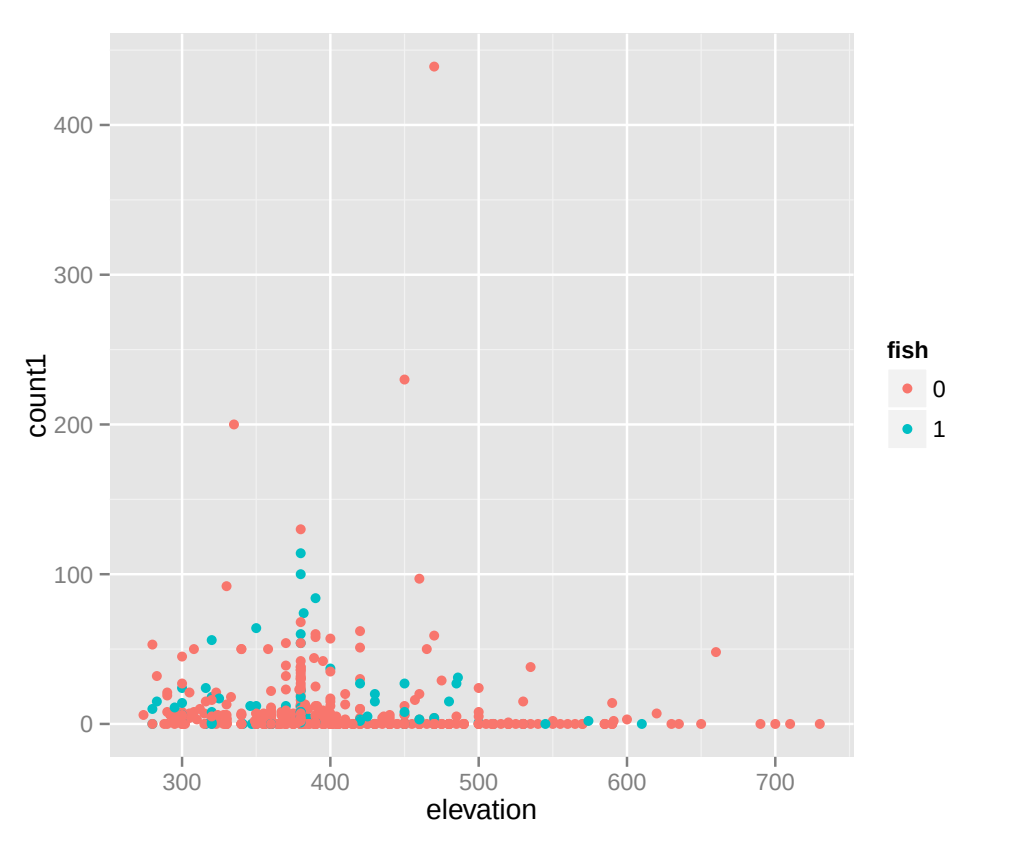
Exercise 3:
Create this plot:



Multivariate data
(Facets)

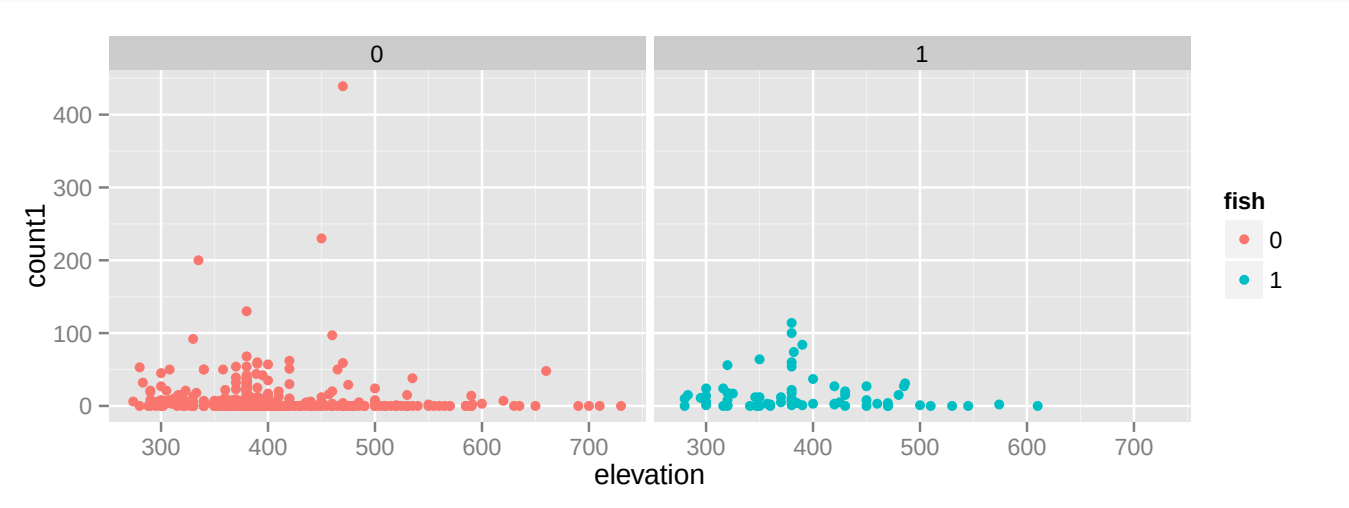
The different relationships between *fish* and *no fish* are hard to spot:

```
ggplot(frogs, aes(x = elevation, y = count1, col = fish)) +
  geom_point()
```



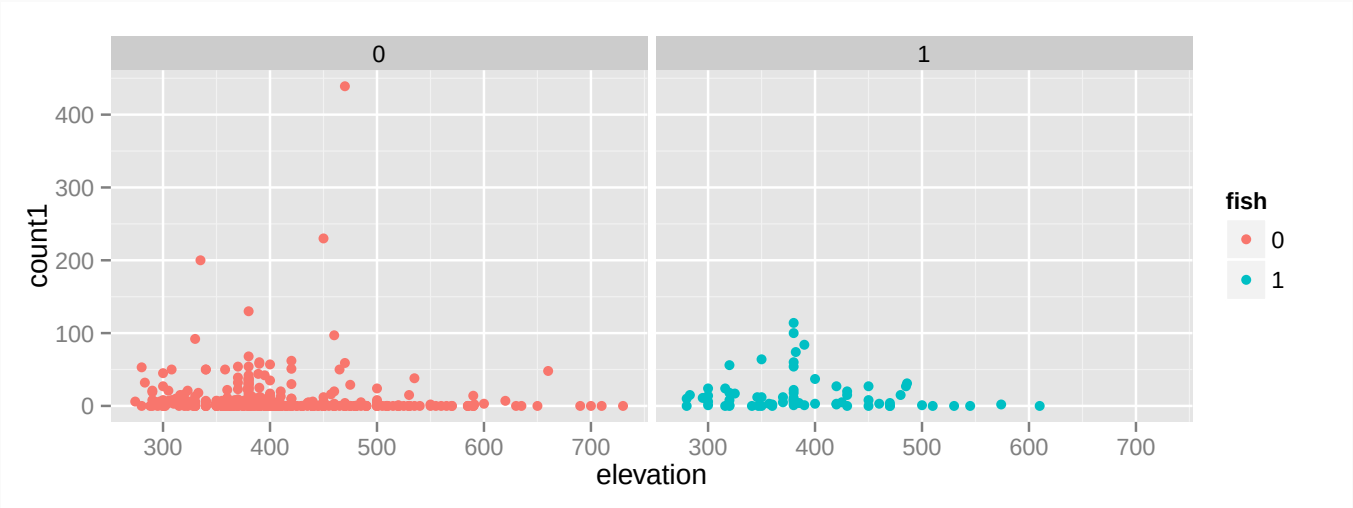
facets: Split the plots by groups

```
ggplot(frogs, aes(x = elevation, y = count1, col = fish)) +
  geom_point() +
  facet_wrap(~fish)
```



facets: Split the plots by groups

```
ggplot(frogs, aes(x = elevation, y = count1, col = fish)) +
  geom_point() +
  facet_wrap(~fish)
```



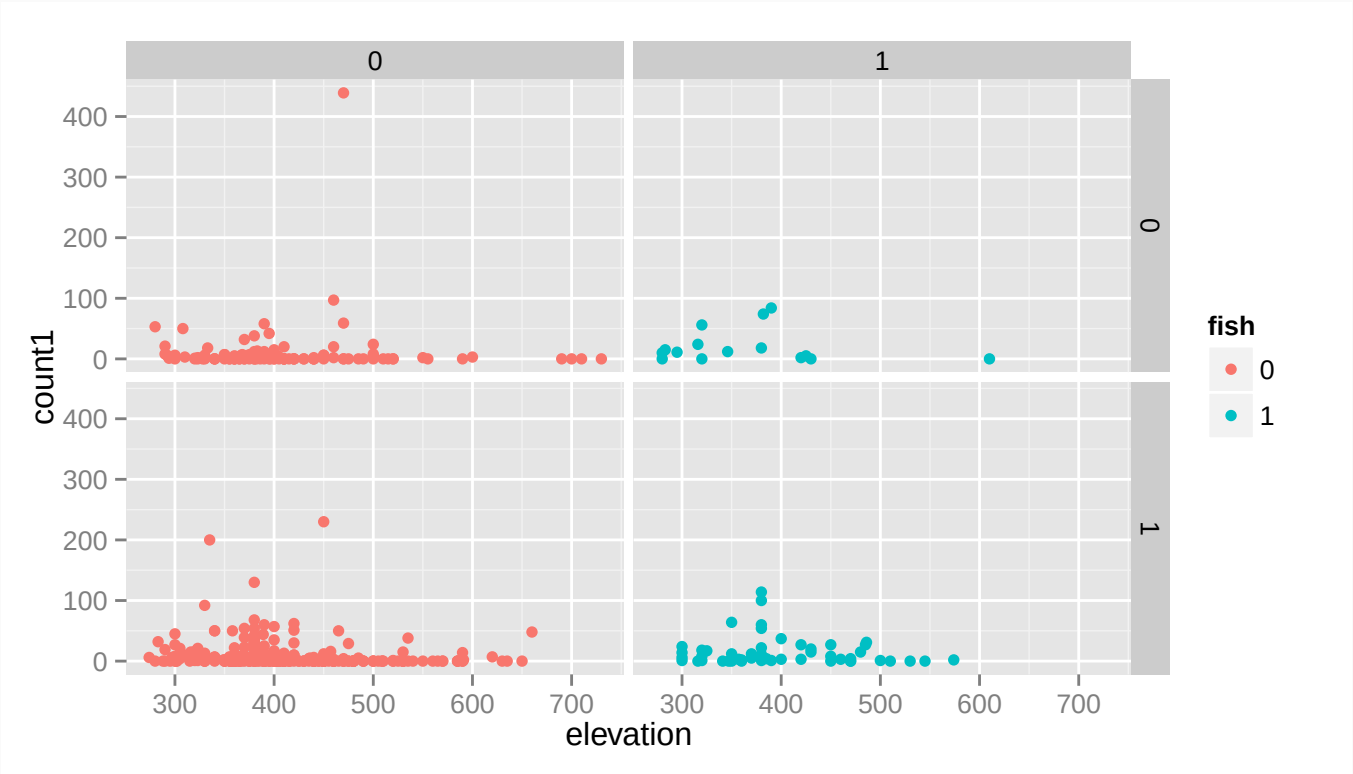
Tip:

The color here is redundant!

facets: Split the plots by groups

- `facet_wrap()` wraps into a rectagular layout
- `facet_grid(rows ~cols)` wraps into rows and columns

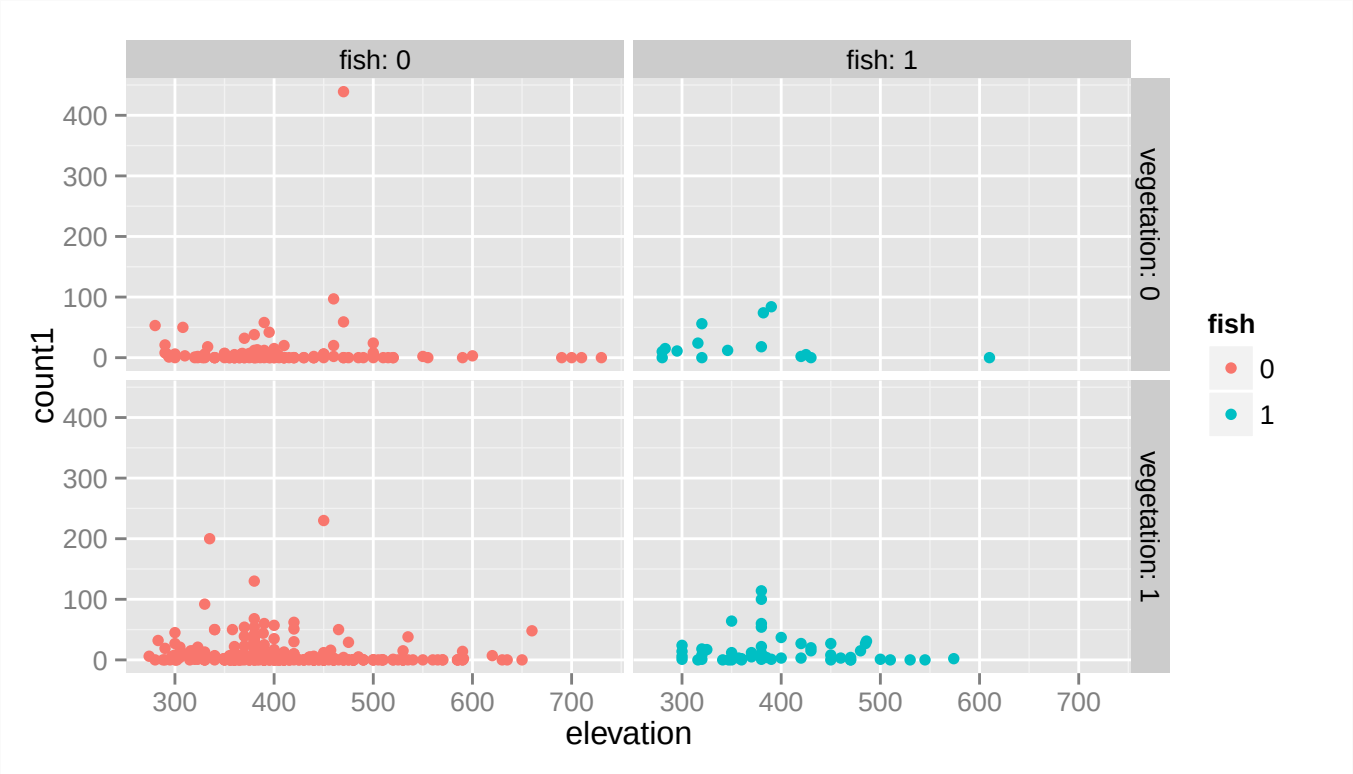
```
ggplot(frogs, aes(x = elevation, y = count1, col = fish)) +
  geom_point() +
  facet_grid(vegetation~fish)
```



facets: Split the plots by groups

- add text to strip to identify variables with `labeller = label_both`

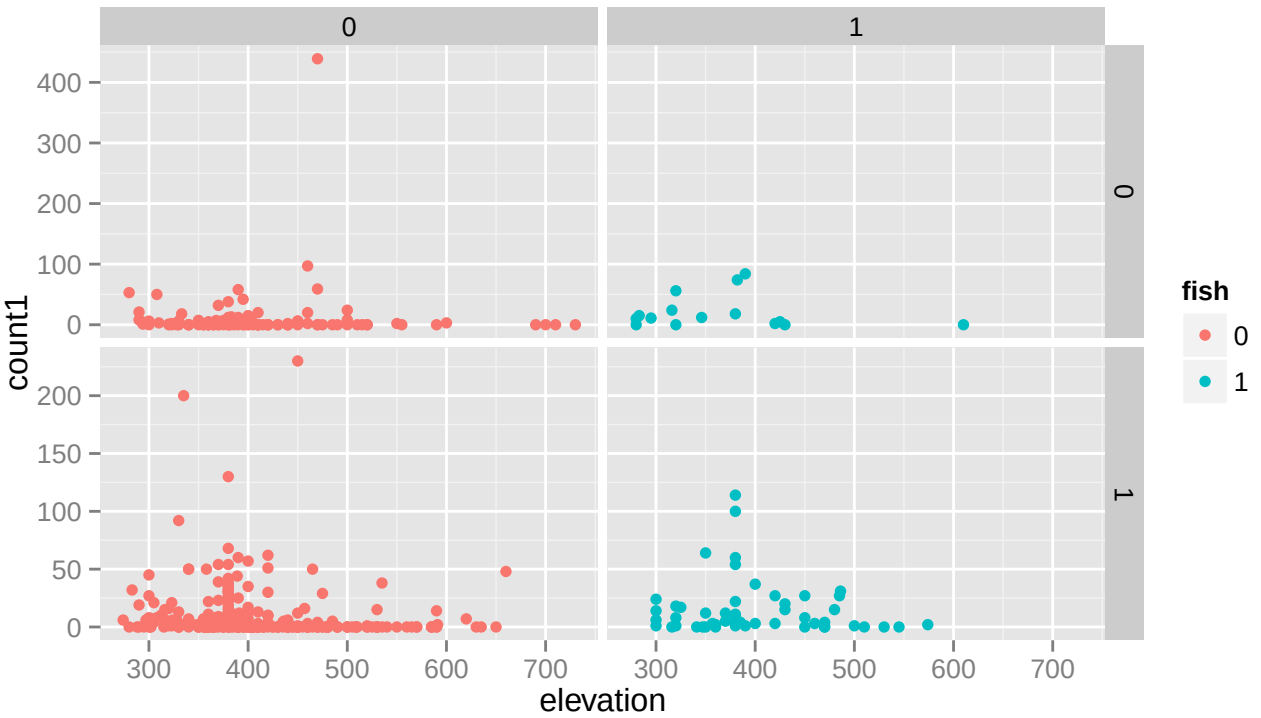
```
ggplot(frogs, aes(x = elevation, y = count1, col = fish)) +
  geom_point() +
  facet_grid(vegetation~fish, labeller = label_both)
```



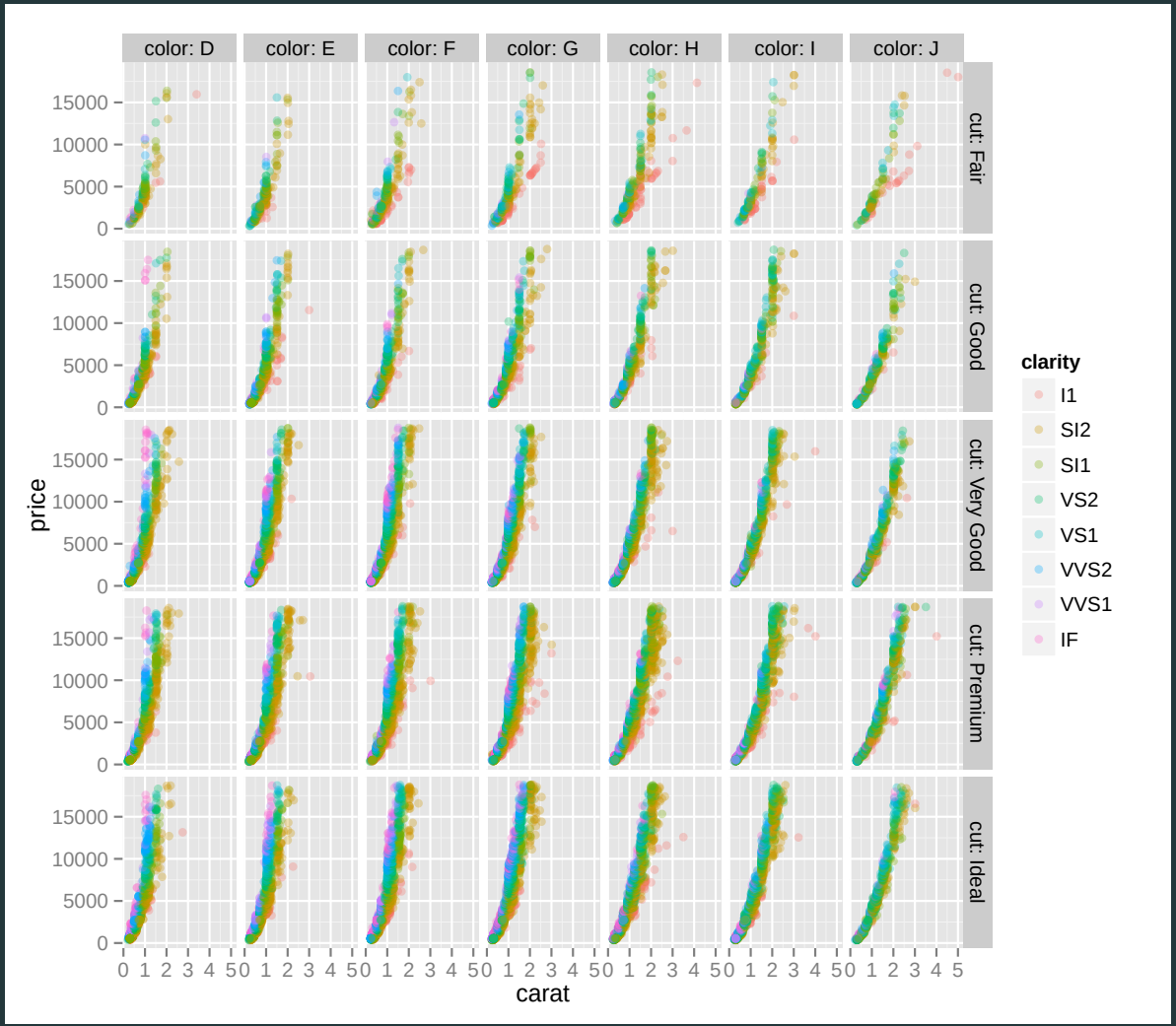
facets: Split the plots by groups

- Both rows have the same scale.
- use `scales = "free" or "frex_x", "free_y"` to adjust

```
ggplot(frogs, aes(x = elevation, y = count1, col = fish)) +
  geom_point() +
  facet_grid(vegetation~fish,
             scales = 'free_y')
```



Exercise 4:
Create this plot:

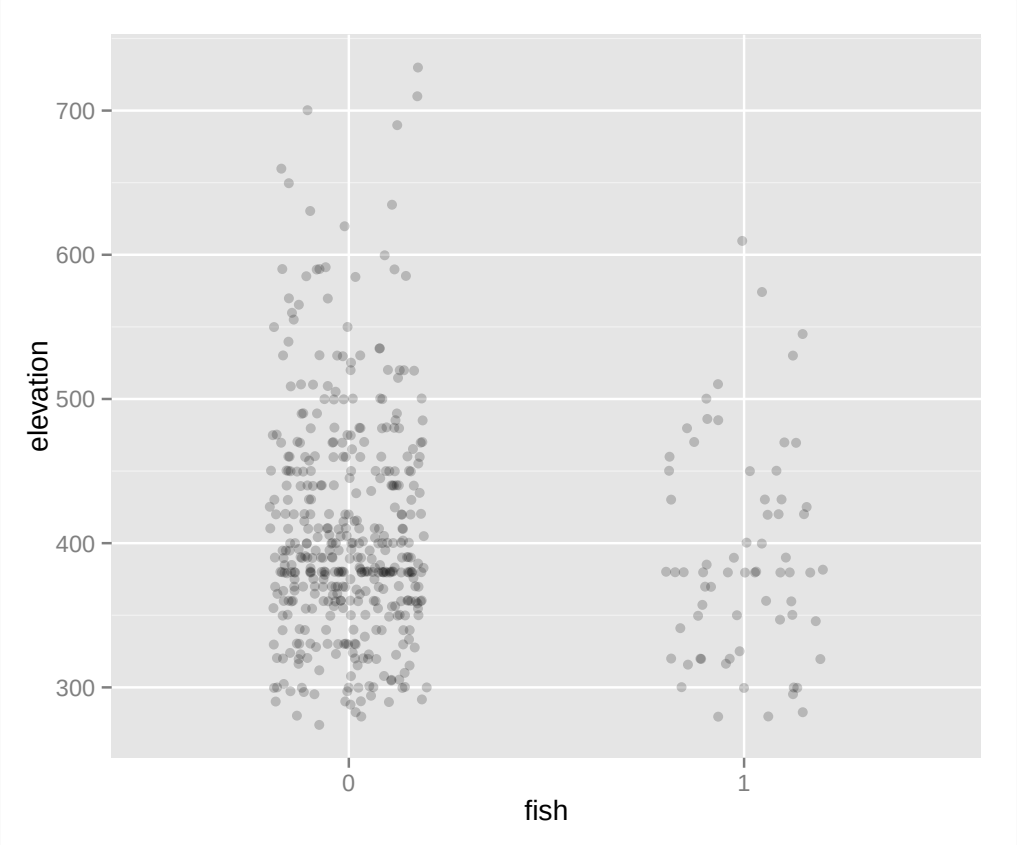


Aggregated data
(Visualize errors)

Aggregate data

- Often we want to show aggregates of our data (e.g. means and CIs)

```
ggplot(frogs, aes(x = fish, y = elevation)) +
  geom_point(position = position_jitter(width = 0.2), alpha = 0.2)
```



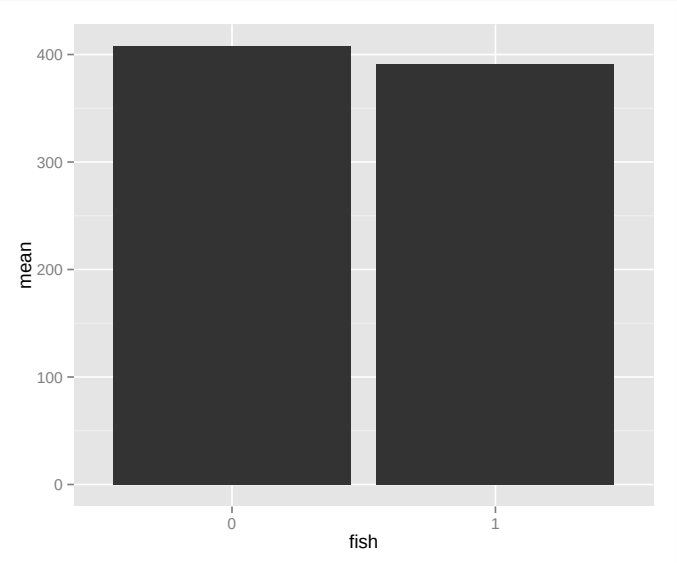
Aggregate data

- Aggregate first, then plot!
- A lot of possibilities to aggregate (`plyr`, `dplyr`, `data.table`, `sqldf`, `ave()`, `doBy`, ...)
- I use a `plyr` solution here.

```
require(plyr)
mean_ci <- ddply(frogs, .(fish), summarise,
      mean = mean(elevation),
      err = qnorm(0.975) * sd(elevation) / sqrt(length(elevation)))
mean_ci
```

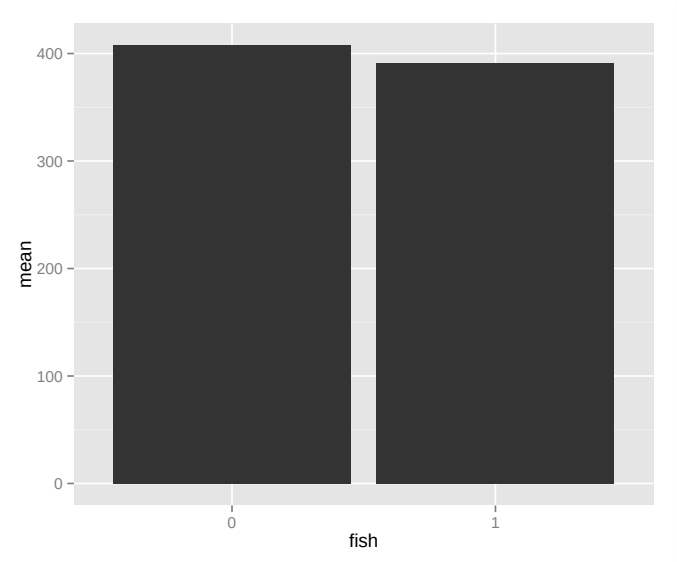
```
##   fish     mean       err
## 1    0 407.5990  7.509866
## 2    1 391.6119 17.560849
```

```
ggplot(mean_ci, aes(x = fish, y = mean)) +
  geom_bar(stat = 'identity')
```



- We need to set `stat = 'identity'` so that the heights of the bars represent the mean.
- Otherwise `geom_bar` tries to transform the data (height = number of cases per group)

```
ggplot(mean_ci, aes(x = fish, y = mean)) +
  geom_bar(stat = 'identity')
```

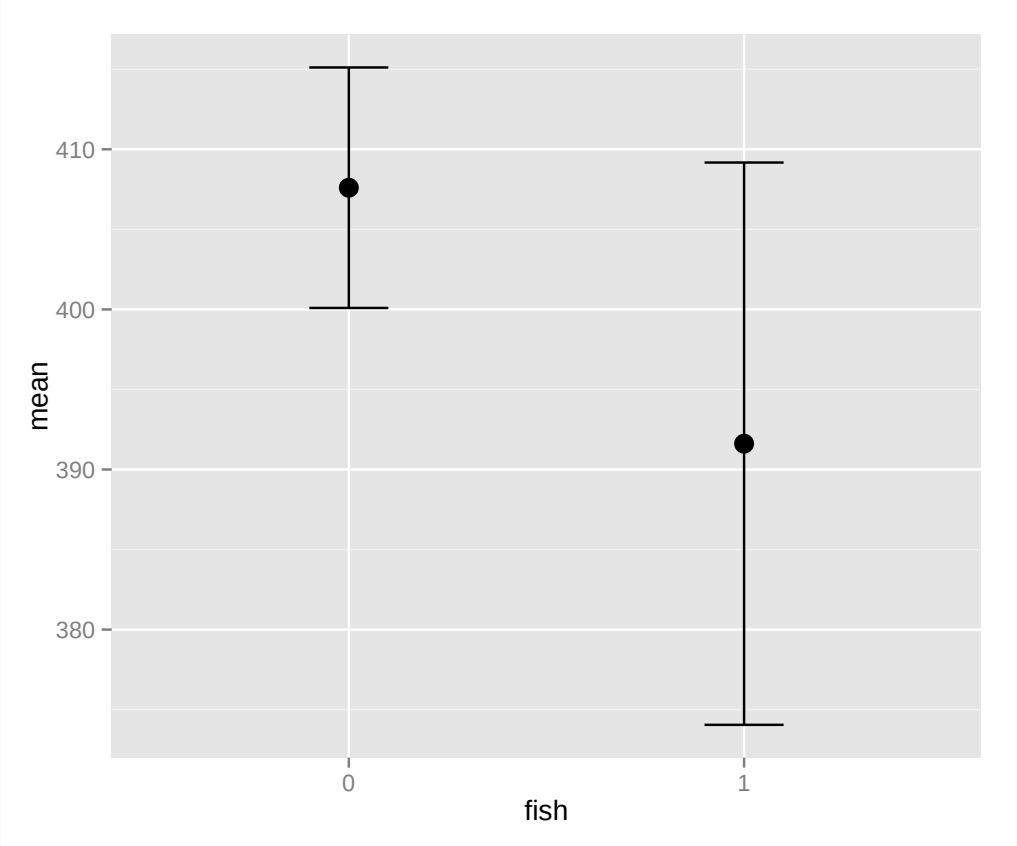


- We need to set `stat = 'identity'` so that the heights of the bars represent the mean.
- Otherwise `geom_bar` tries to transform the data (height = number of cases per group)

Tip:

Don't use bars to display means. Bad data-to-ink ratio (Tufte)!

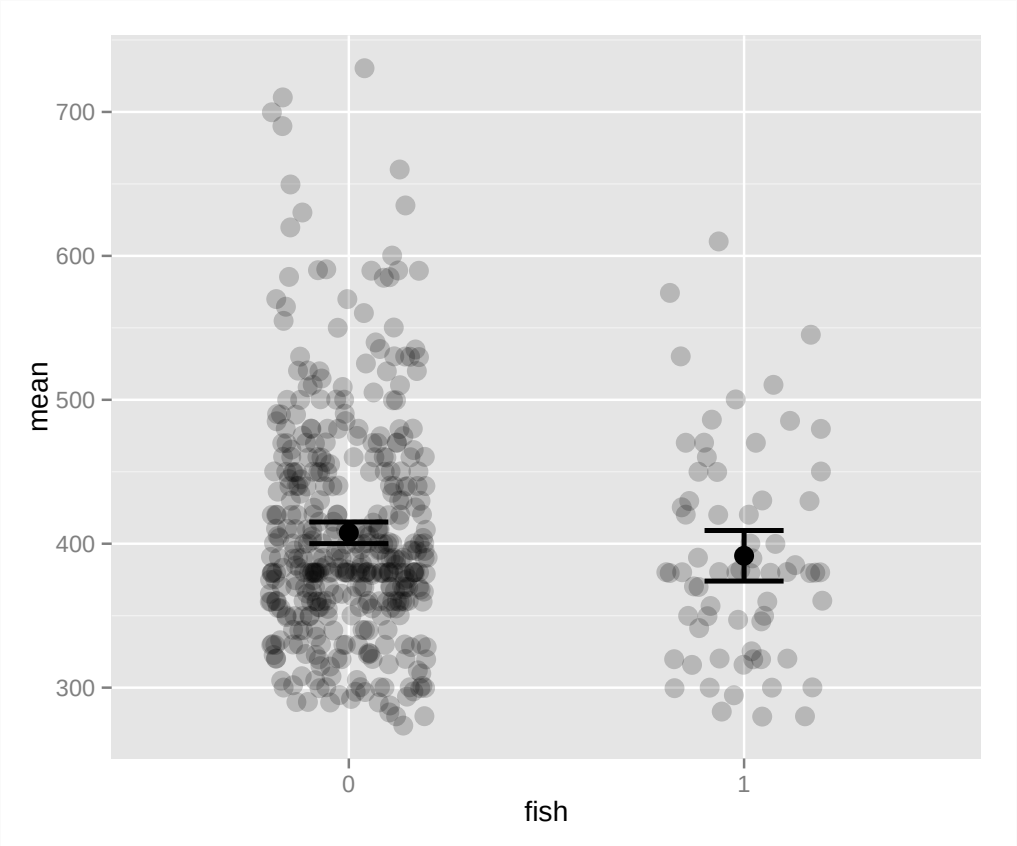
```
ggplot(data = mean_ci, aes(x = fish, y = mean)) +
  geom_point(size = 4) +
  geom_errorbar(aes(ymin = mean - err, ymax = mean + err), width = 0.2)
```



- points are enough to represent the means
- `width = 0.2` fo smaller whisker (default = 1)

Combining different datasets into on plot

```
ggplot() +
  geom_point(data = mean_ci, aes(x = fish, y = mean), size = 4) +
  geom_errorbar(data = mean_ci, aes(x = fish, y = mean,
                                    ymin = mean - err, ymax = mean + err),
                width = 0.2, size = 1) +
  geom_point(data = frogs, aes(x = fish, y = elevation),
             position = position_jitter(width = 0.2),
             alpha = 0.2, size = 4)
```



Combining different datasets into one plot

```
ggplot() +
  geom_point(data = mean_ci, aes(x = fish, y = mean), size = 4) +
  geom_errorbar(data = mean_ci, aes(x = fish, y = mean,
                                    ymin = mean - err, ymax = mean + err),
                width = 0.2, size = 1) +
  geom_point(data = frogs, aes(x = fish, y = elevation),
             position = position_jitter(width = 0.2),
             alpha = 0.2, size = 4)
```

- Because we have multiple datasets, we cannot pass them globally in `ggplot()`
- specify data and aesthetics for each layer / geom
- `geom_violin` + `geom_jitter` are a good combination

```
ggplot() +
  geom_point(data = mean_ci, aes(x = fish, y = mean), size = 4) +
  geom_errorbar(data = mean_ci, aes(x = fish, y = mean,
                                    ymin = mean - err, ymax = mean + err),
                width = 0.2, size = 1) +
  geom_point(data = frogs, aes(x = fish, y = elevation),
             position = position_jitter(width = 0.2),
             alpha = 0.2, size = 4)
```

- Because we have multiple datasets, we cannot pass them globally in `ggplot()`
- specify data and aesthetics for each layer / geom
- `geom_violin` + `geom_jitter` are a good combination

Tip:

Plotting the raw data is a good idea (spot errors, make data available, etc...)

MAPS

Easy maps with ggmap

- `ggmap` can retrieve maps from different sources (Google Maps, OpenStreetMap, StamenMaps, and CloudMade Maps).
- Needs coordinates in lat/lon format
- easy, consistent and modular framework for spatial graphics
- Workflow: Prepare data > retrive background map > add layers

1) Prepare data: Transform coordinates to lat/lon

- Coordinates in frogs dataset are in CH1903 / LV03 format
- Transform using **sp** package to lat / lon

```
require(sp)
frogs_sp <- frogs
coordinates(frogs_sp) <- ~x+y
proj4string(frogs_sp) <- CRS("+init=epsg:21781")
# transform coordinates to WGS84 (lat/lon)
frogs_84 <- spTransform(frogs_sp, CRS("+init=epsg:4326"))
```

```
head(coordinates(frogs_sp))
```

```
##        x      y
## 1 649750 248850
## 2 647350 255750
## 3 650250 244600
## 4 649400 243850
## 5 646700 240750
## 6 646500 253650
```

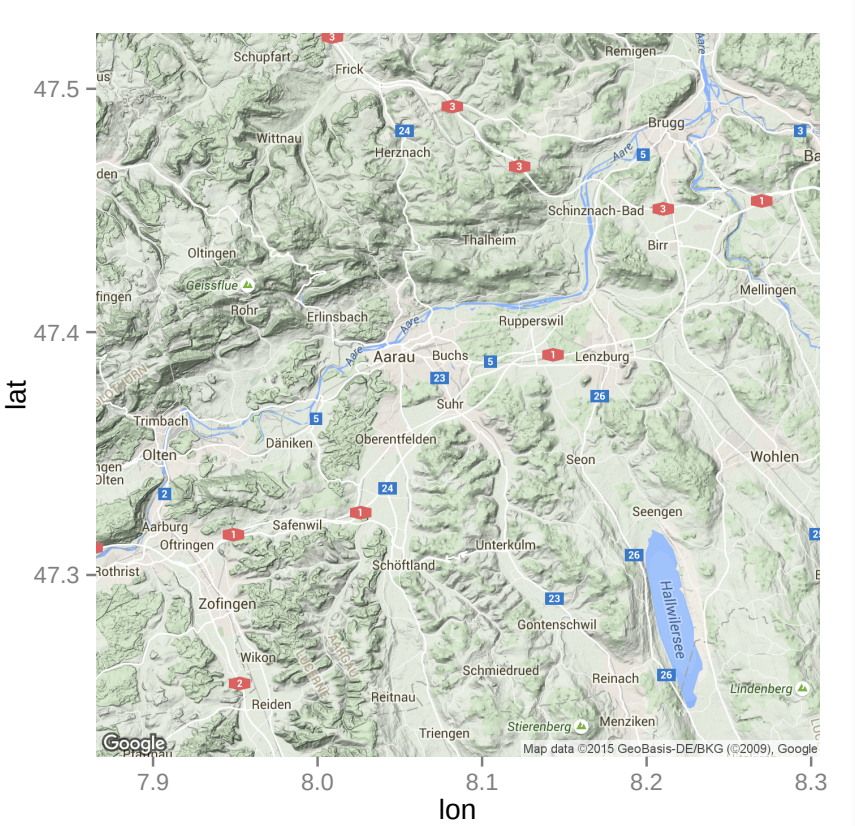
```
head(coordinates(frogs_84))
```

```
##          x        y
## 1 8.097525 47.38860
## 2 8.066471 47.45083
## 3 8.103670 47.35034
## 4 8.092338 47.34366
## 5 8.056288 47.31597
```

2) Retrieve backgound map

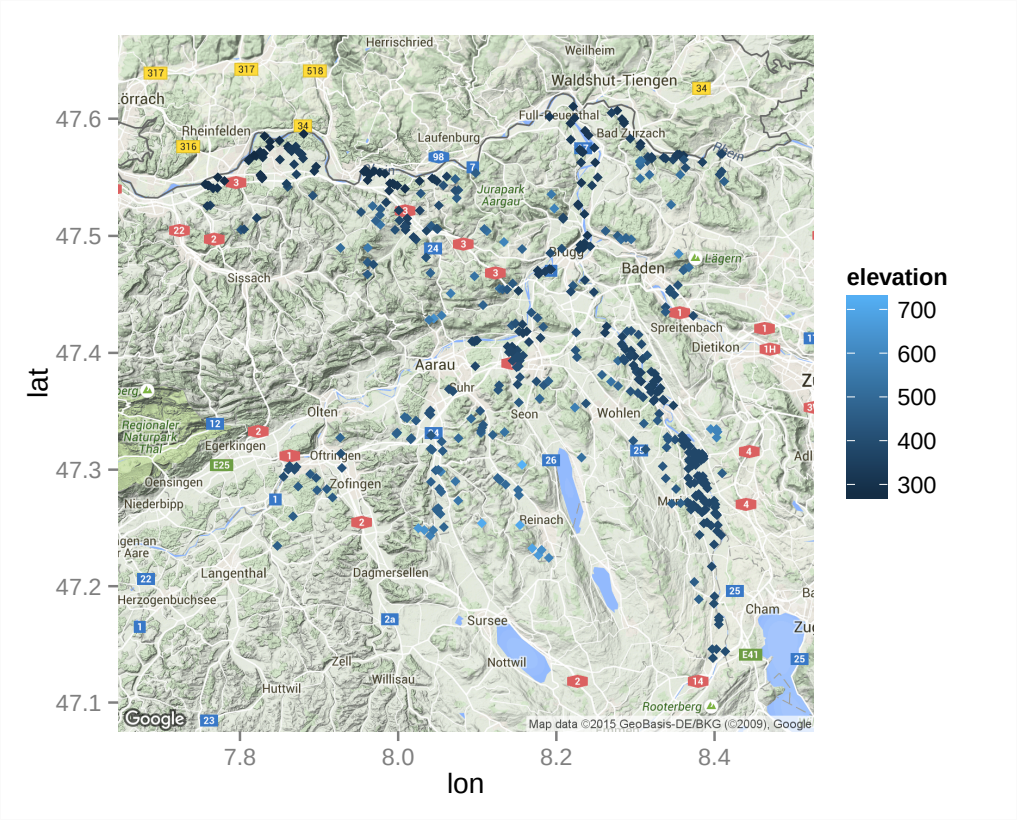
- `get_map()` needs a location of which to retrive the map
- I use here the bounding box of the coordinates, but adress or a coordinate work also
- other useful options are *zoom* and *maptype*

```
require(ggmap)
ggmap(get_map(frogs_84@bbox, source = 'google'), zoom = 10)
```



3) Add layers

```
ggmap(get_map(frogs_84@bbox, source = 'google', zoom = 10)) +
  geom_point(data = data.frame(frogs_84),
             aes(x = x, y = y, col = elevation),
             size = 2,
             shape = 18)
```



Break!

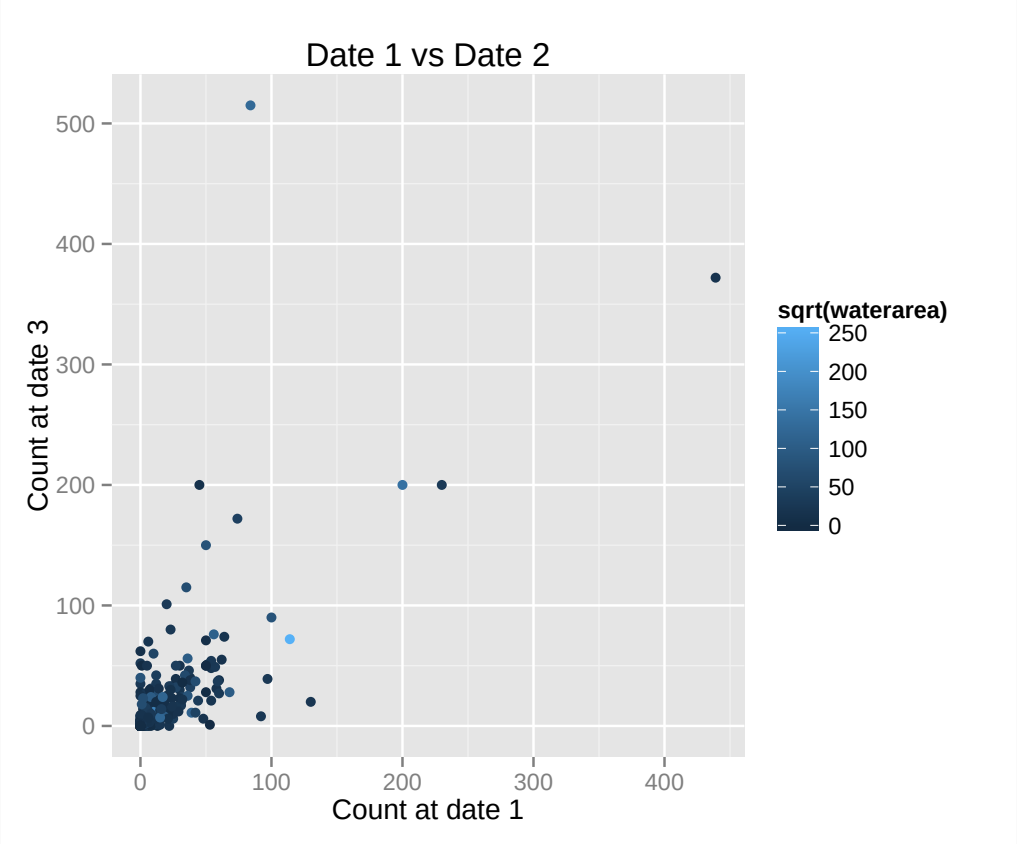
Now we had enough of the basics

Next part: How to customize your plots.

Customization and Prettification

- `labs()` can be used to set axis labels & title

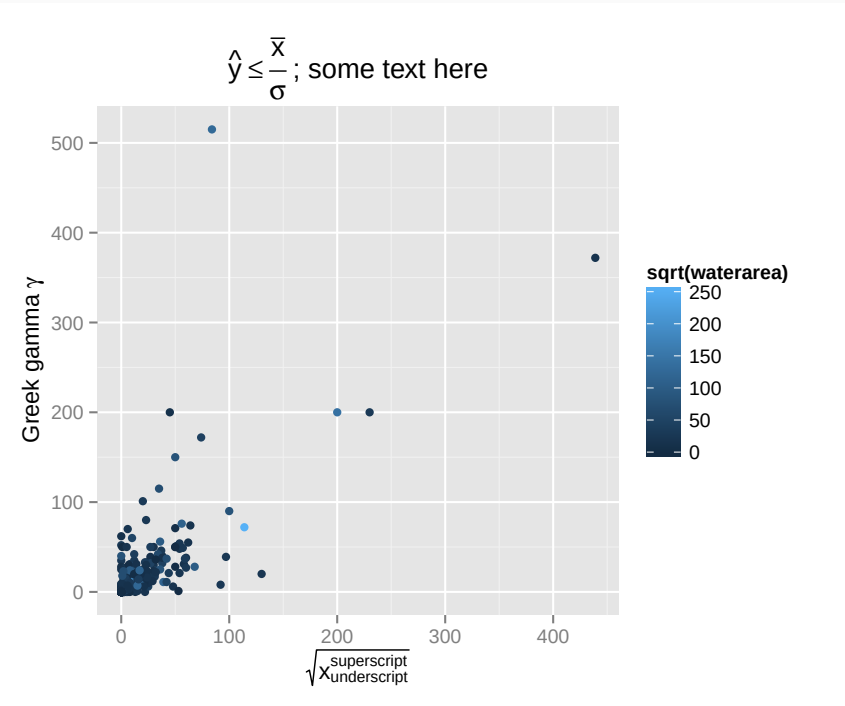
```
ggplot(frogs, aes(x = count1, y = count2, col = sqrt(waterarea))) +
  geom_point() +
 labs(x = 'Count at date 1', y = 'Count at date 3', title = 'Date 1 vs Date 2')
```



Working with labels: plotmath

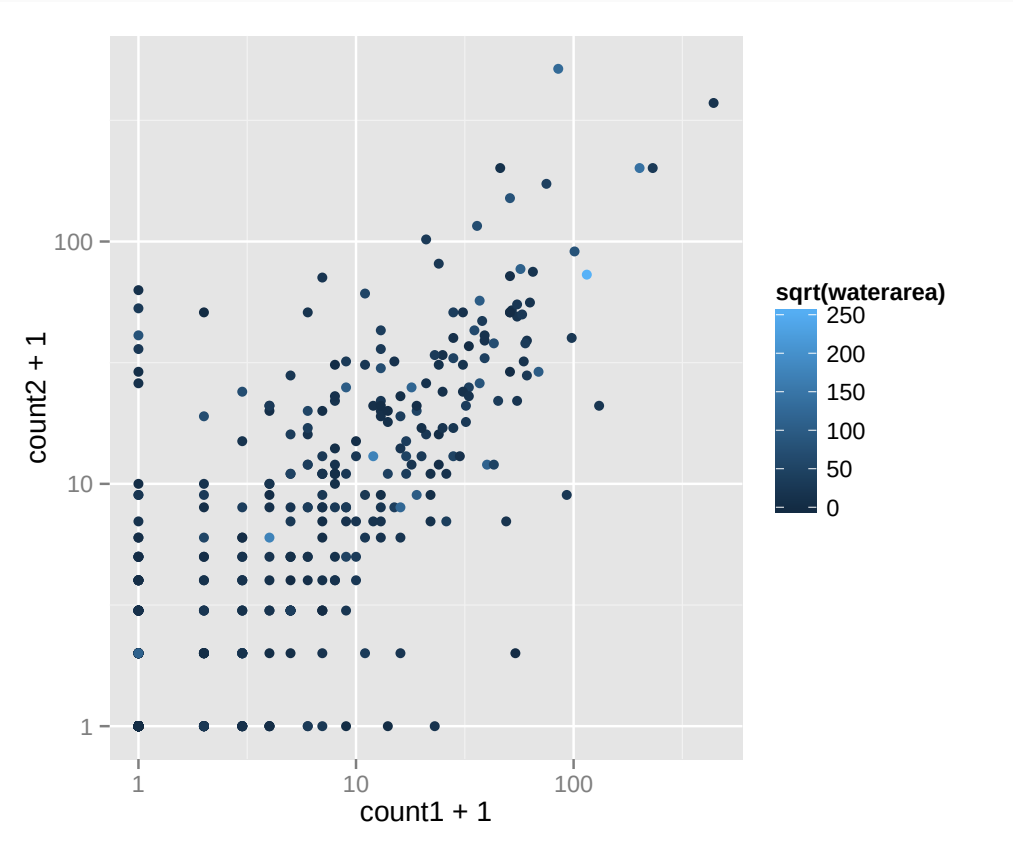
- For mathematical annotation use `plotmath` `?plotmath`
- wrap into `expression()`
- separate text with ~

```
ggplot(frogs, aes(x = count1, y = count2, col = sqrt(waterarea))) +
  geom_point() +
   labs(x = expression(sqrt(x[underscript]^superscript)),
        y = expression('Greek gamma'~gamma),
    title = expression(hat(y) <= frac(bar(x), sigma)~'; some text here'))
```



- The distribution of counts is highly skewed
- try a double log scale

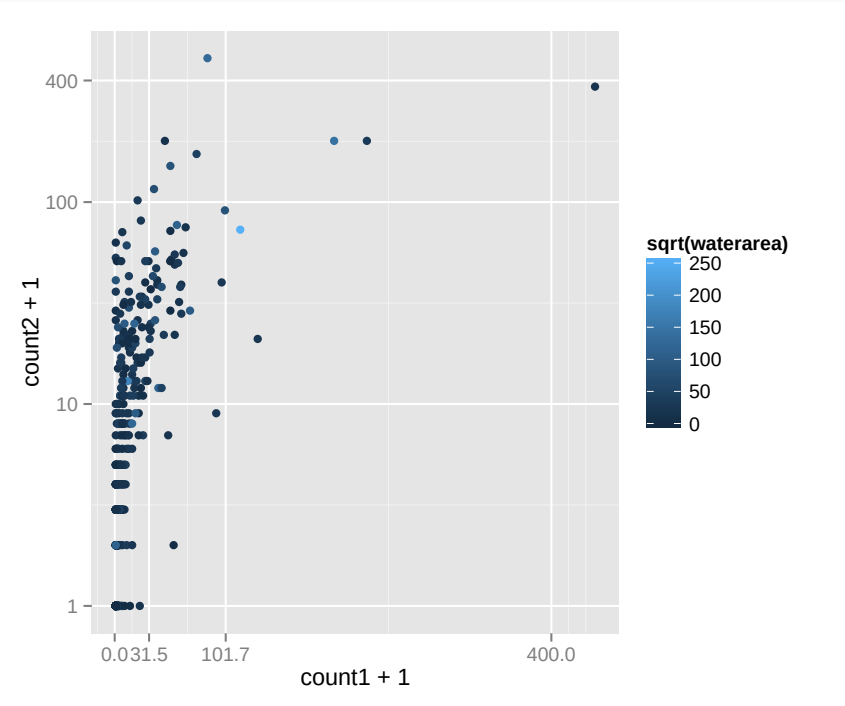
```
ggplot(frogs, aes(x = count1 + 1, y = count2 + 1, col = sqrt(waterarea))) +
  geom_point() +
  scale_x_log10() +
  scale_y_log10()
```



```
ggplot(frogs, aes(x = count1 + 1, y = count2 + 1, col = sqrt(waterarea))) +
  geom_point() +
  scale_x_log10() +
  scale_y_log10()
```

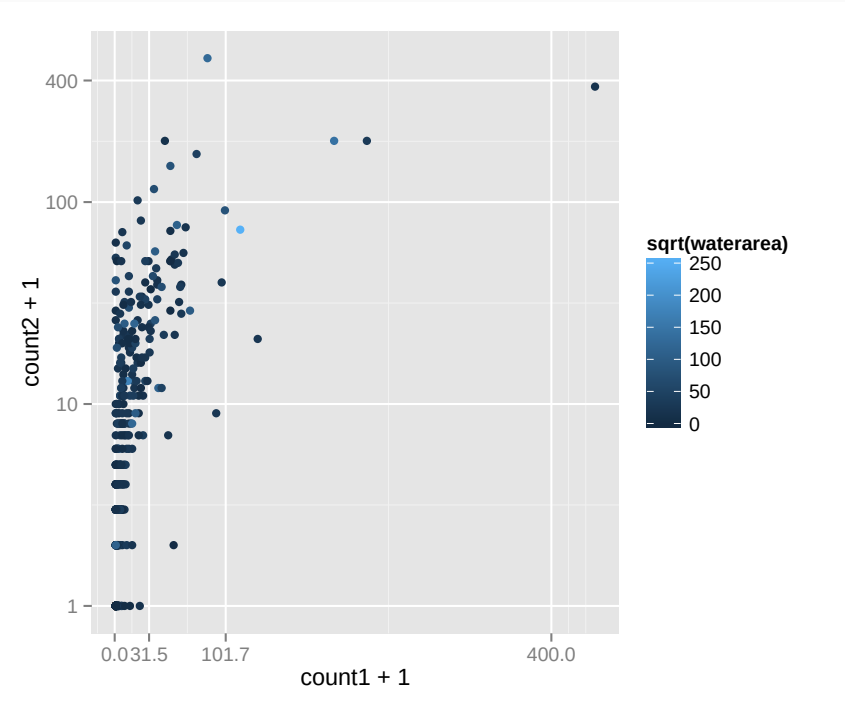
- General form: scale_aesthetic_type
- type of scale (e.g. `log10, grey, continuous etc.`)
- data -> scale -> aesthetic

```
ggplot(frogs, aes(x = count1 + 1, y = count2 + 1, col = sqrt(waterarea))) +
  geom_point() +
  scale_x_continuous(breaks = c(0, 31.5, 101.7, 400)) +
  scale_y_log10(breaks = c(1, 10, 100, 400))
```



- use the `breaks=` argument to specify where ticks shoud be drawn.

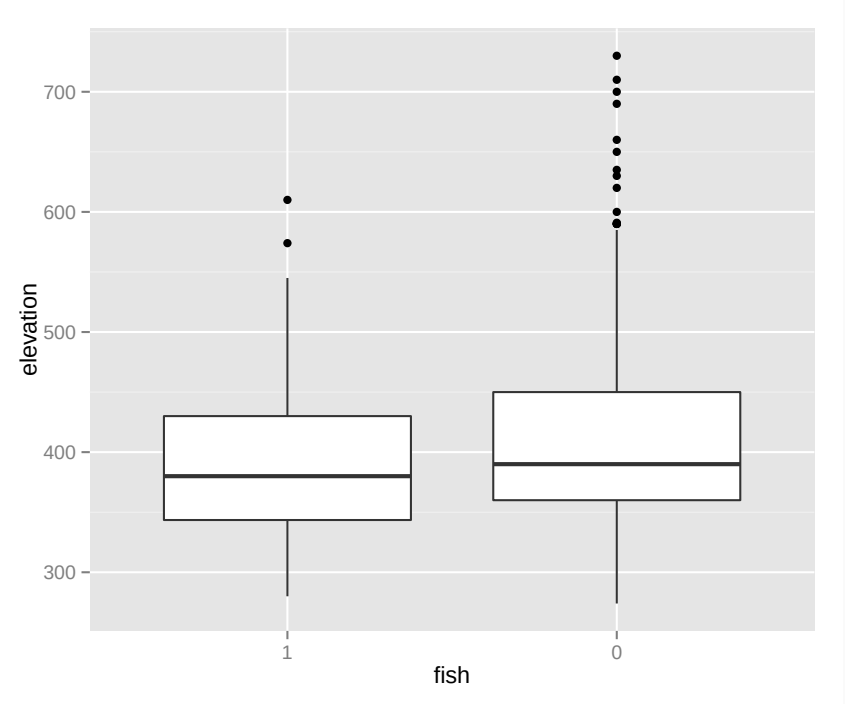
```
ggplot(frogs, aes(x = count1 + 1, y = count2 + 1, col = sqrt(waterarea))) +
  geom_point() +
  scale_x_continuous(breaks = c(0, 31.5, 101.7, 400)) +
  scale_y_log10(breaks = c(1, 10, 100, 400))
```



- use the `breaks=` argument to specify where ticks shoud be drawn.
- Simplify the axes ticks to min / max only for presentations.

SCALES: CHANGE ORDER DISCRETE AXES

```
ggplot(frogs, aes(x = fish, y = elevation)) +
   geom_boxplot() +
  scale_x_discrete(limits = c("1", "0"))
```

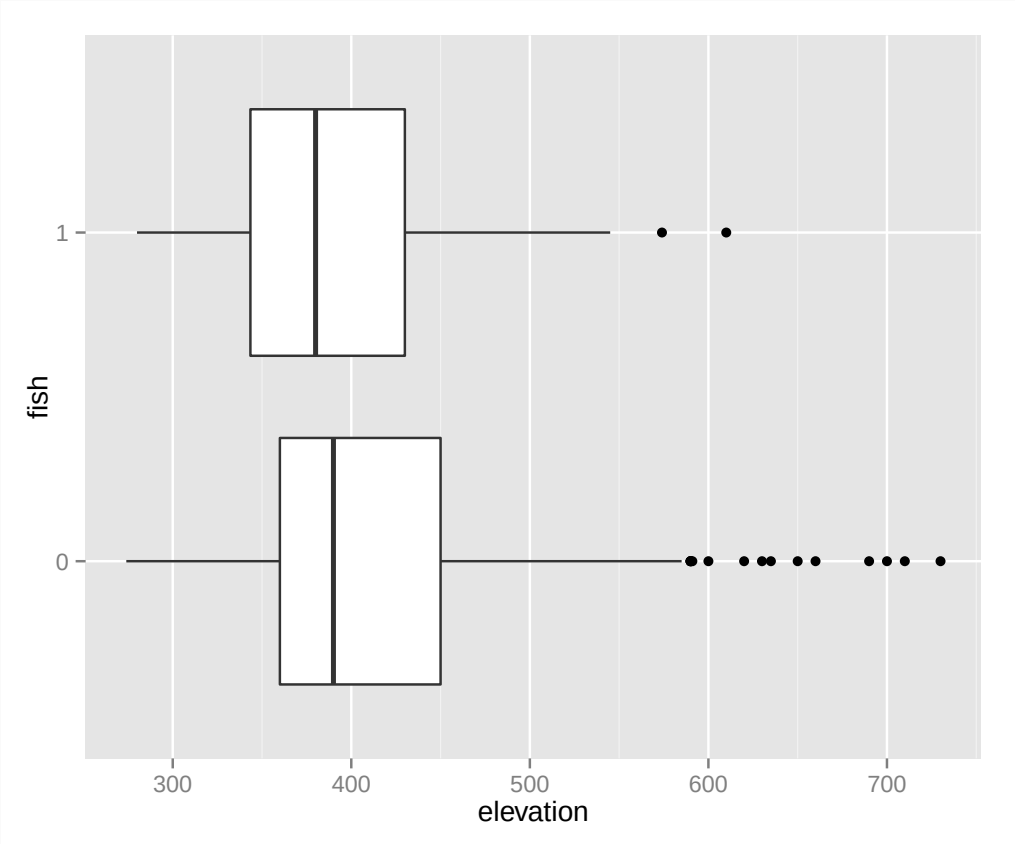



```
# or to generally reverse
# scale_x_discrete(limits = rev(levels(frogs$fish)))
```

- use the `limits=` argument to specify the order

AXES: SWAP X AND Y

```
ggplot(frogs, aes(x = fish, y = elevation)) +
   geom_boxplot() +
  coord_flip()
```

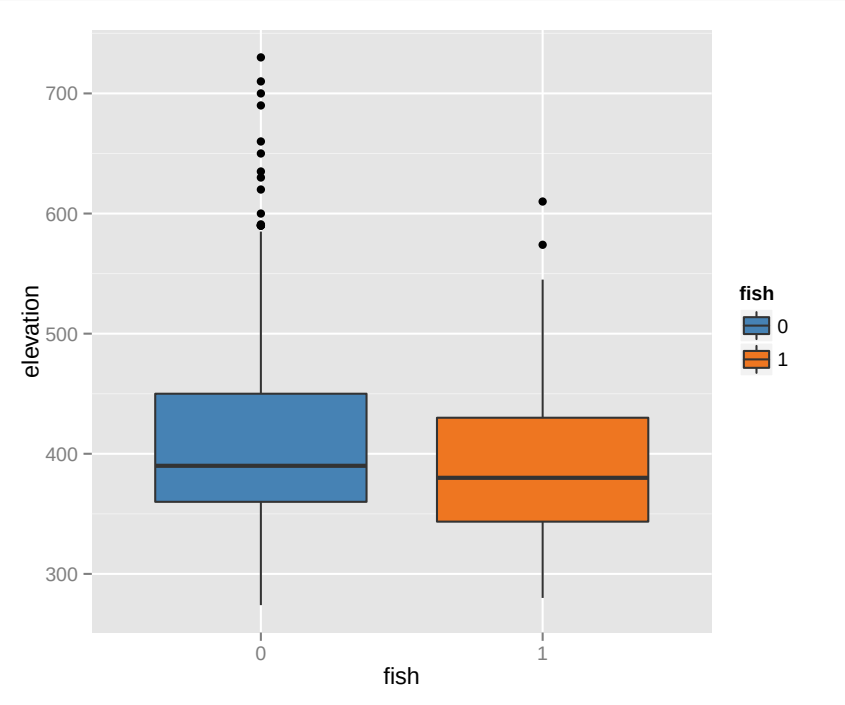


- Sometimes useful to compare the order & magnitude (see slide 12)

SCALES: COLOR (MANUAL)

- Manually select colors from built-in colors
- see `?colors` and `demo("colors")` for possible values

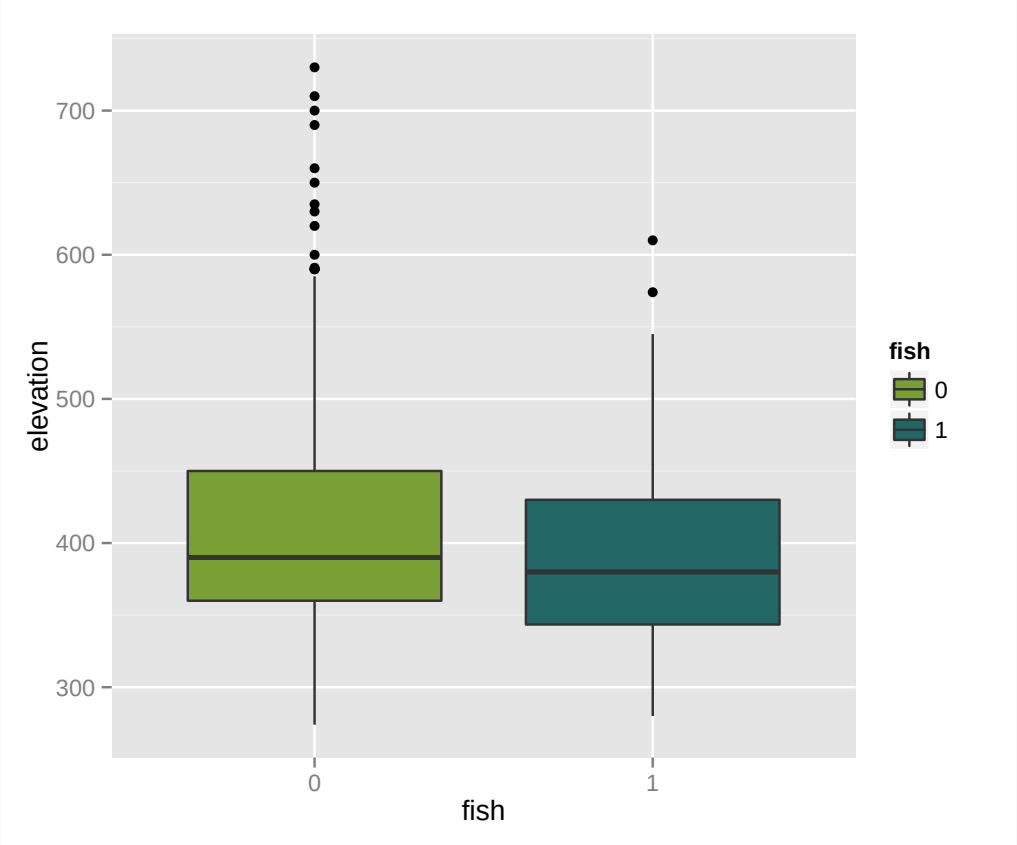
```
ggplot(frogs, aes(x = fish, y = elevation, fill = fish)) +
   geom_boxplot() +
  scale_fill_manual(values = c('steelblue', 'chocolate2'))
```



SCALES: COLOR (MANUAL)

- Manually select colors from Hex code
- a lot of generators on the web (e.g. `paletton.com`)

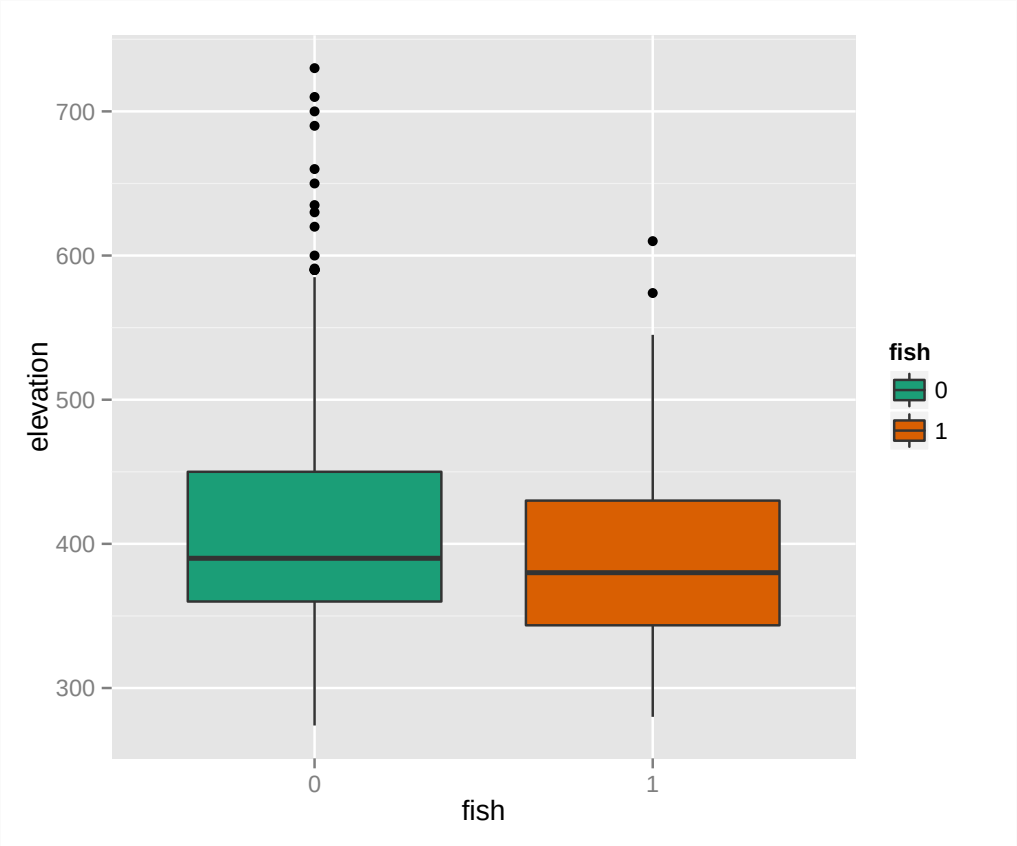
```
ggplot(frogs, aes(x = fish, y = elevation, fill = fish)) +
   geom_boxplot() +
  scale_fill_manual(values = c('#7A9F35', '#226666'))
```



SCALES: COLOR (PALETTES)

- A palette is a specific order of colors
- Select from a predefined palette
- e.g. `http://colorbrewer2.org/`

```
ggplot(frogs, aes(x = fish, y = elevation, fill = fish)) +
   geom_boxplot() +
  scale_fill_brewer(palette = 'Dark2')
```



- The palettes can be classified into 3 categories of data:

  sequential change of a unique color. Use for qualitative data from low to high.

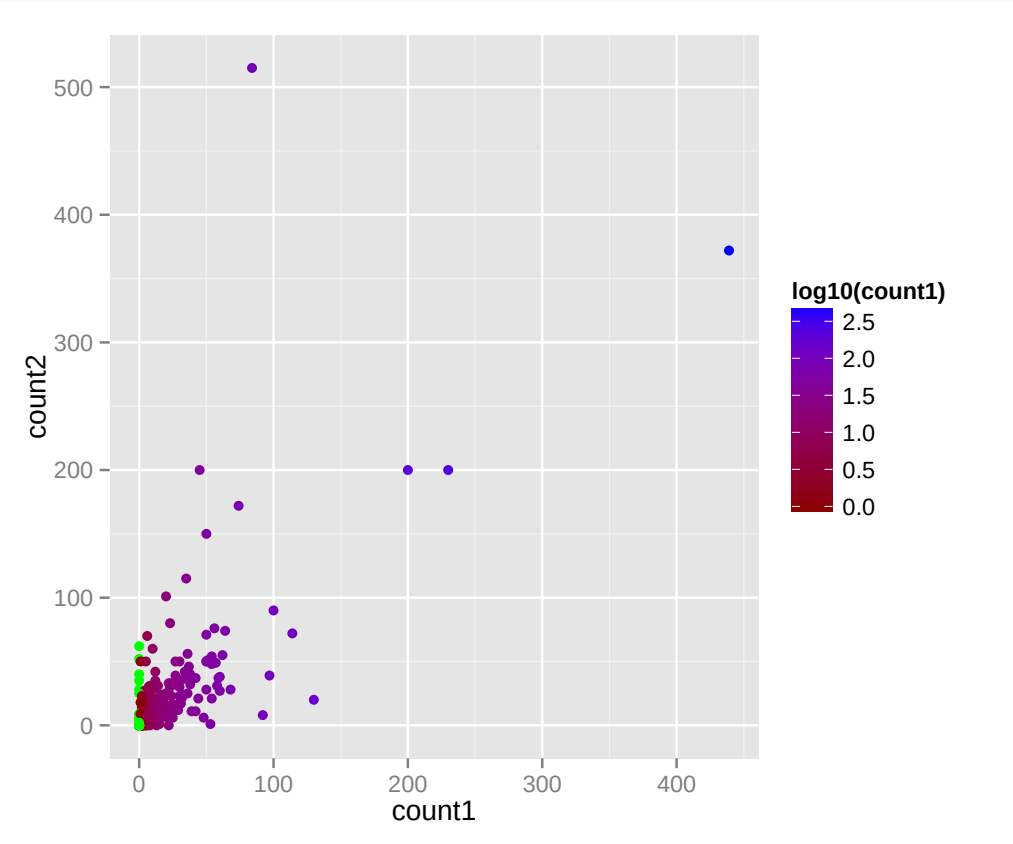
  diverging change from one color to another. Used to show deviations from the mean.

  qualitative maximze difference between colors. Used for categorical data.

- Colorblind safe palettes (8% of men are colorblind)

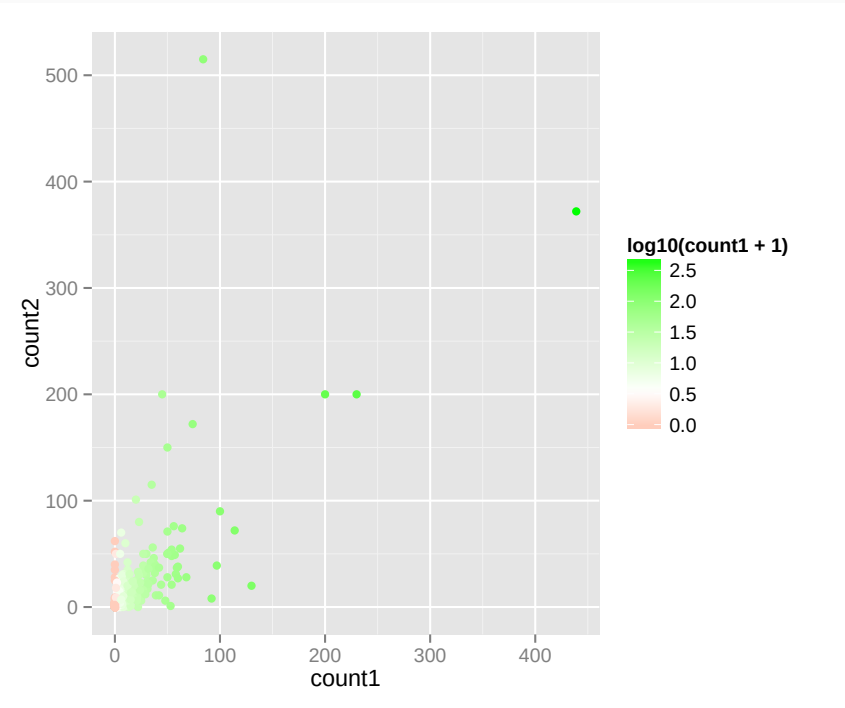
- for continuous variable use a gradient (no palette)
- Change the gradient for a continuous variable

```
ggplot(frogs, aes(x = count1, y = count2, col = log10(count1))) +
  geom_point() +
  scale_color_gradient(low = 'darkred', high = 'blue', na.value = 'green')
```

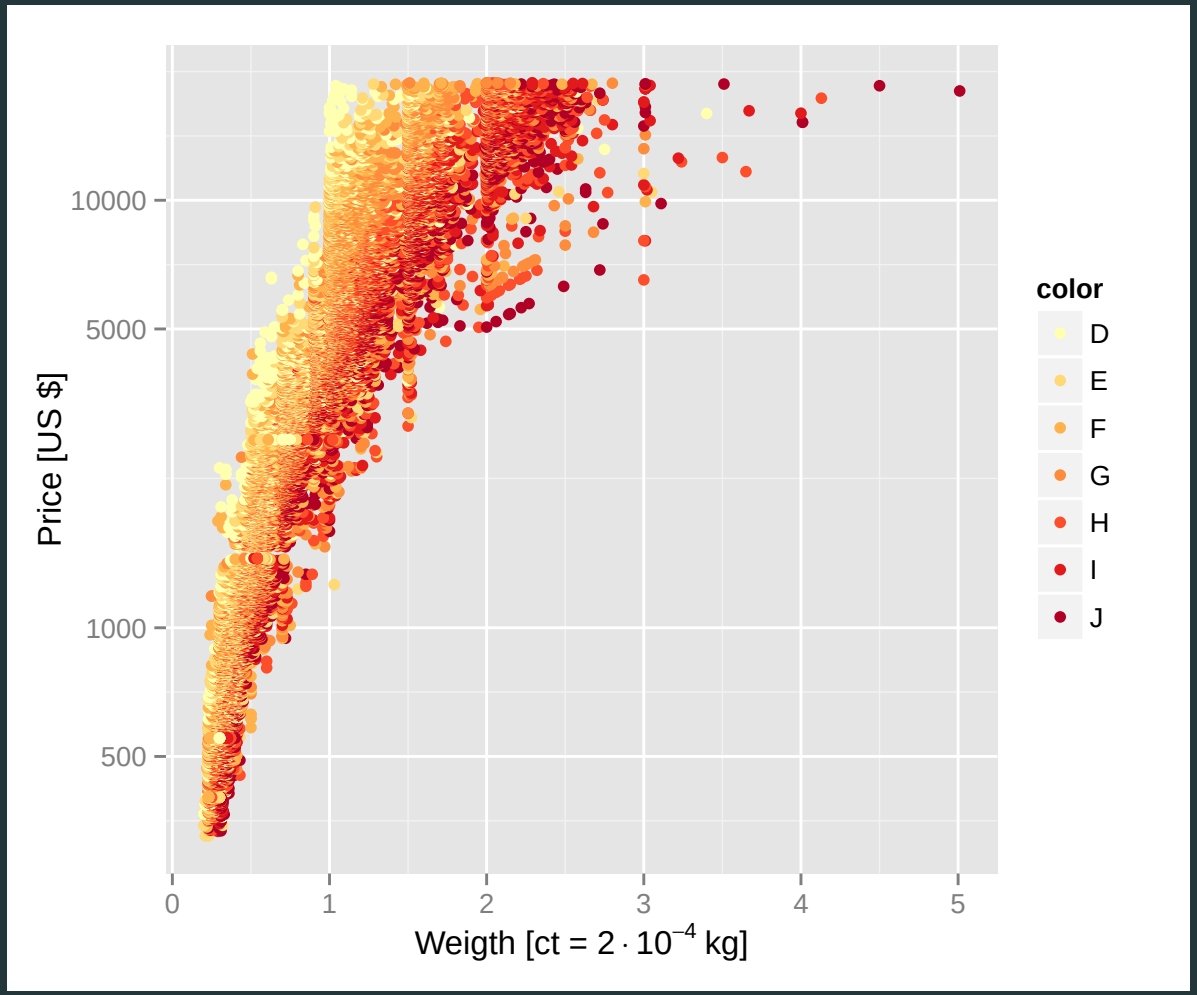


- `scale_*_gradient2` for a diverging gradient
- `midpoint` defaults to 0, changed to mean here

```
midp <- mean(log10(frogs$count1+1))
ggplot(frogs, aes(x = count1, y = count2, col = log10(count1+1))) +
  geom_point() +
  scale_color_gradient2(low = 'red', mid = 'white', high = 'green',
                        midpoint = midp, space = 'Lab')
```



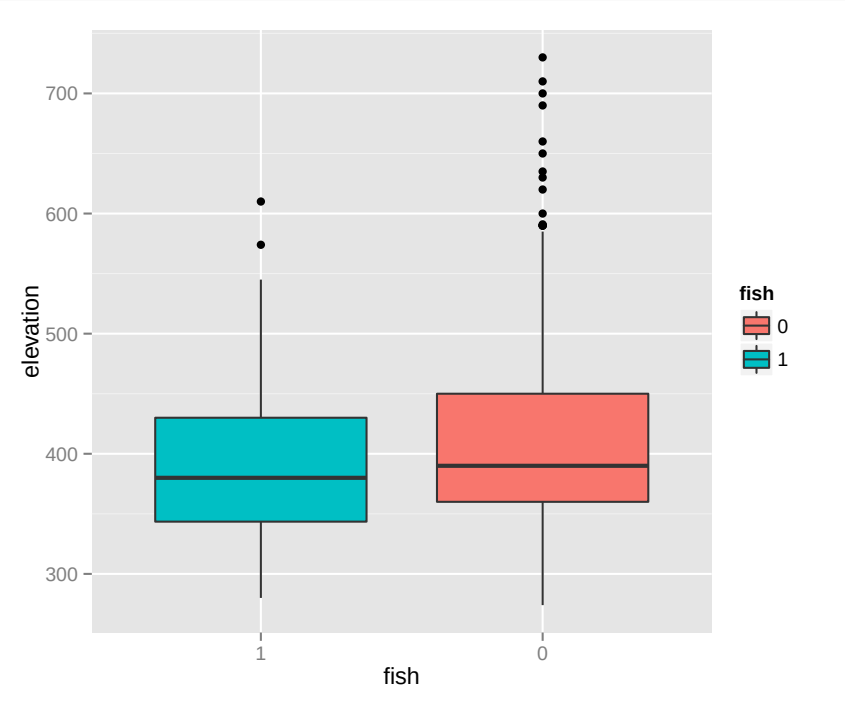
Exercise 5:
Create this plot:



Change order on x-axis

- used `scale_x_discrete(limits = c("1", "0"))` to change order on x-axis
- how to change order in legend?

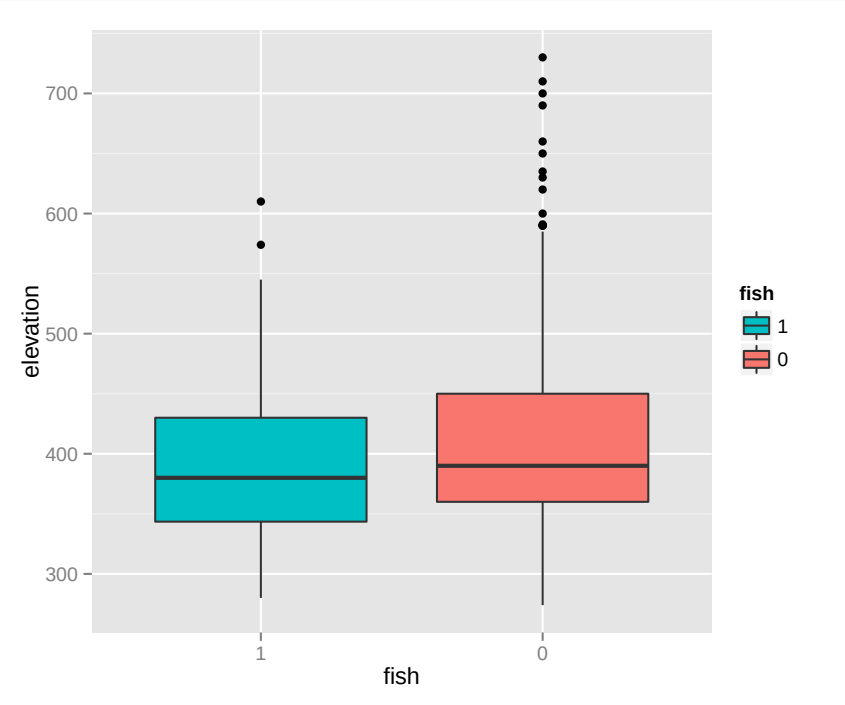
```
ggplot(frogs, aes(x = fish, y = elevation, fill = fish)) +
geom_boxplot() +
scale_x_discrete(limits = c("1", "0"))
```



Working with legends: Change order in legend

- use scale for fill aesthetic.
- use `breaks = <levels>`.

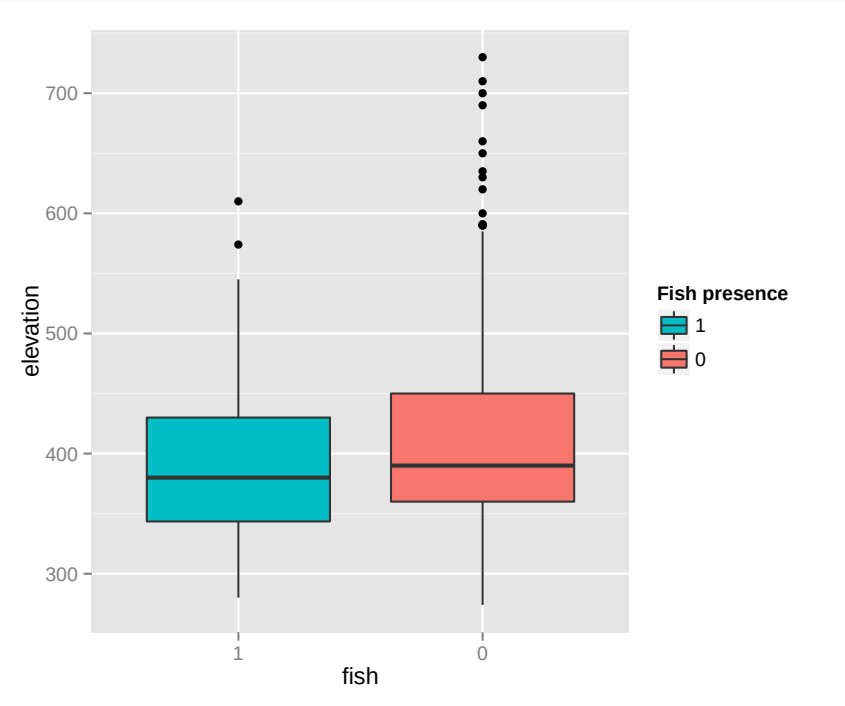
```
ggplot(frogs, aes(x = fish, y = elevation, fill = fish)) +
geom_boxplot() +
scale_x_discrete(limits = c('1', '0')) +
scale_fill_discrete(breaks = c('1', '0'))
```



Working with legends: Change legend title

- use `name = <legendtitle>`

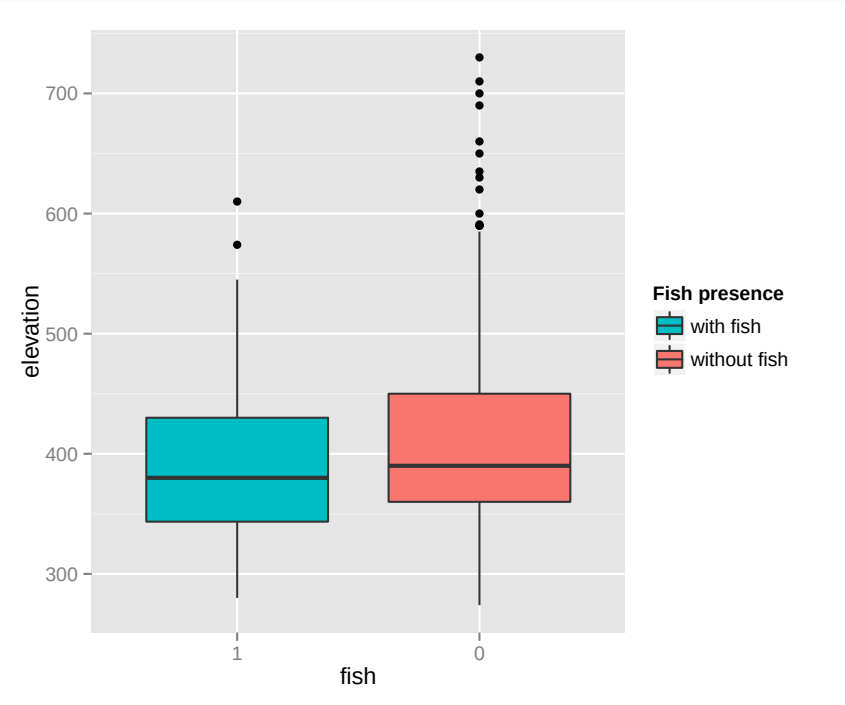
```
ggplot(frogs, aes(x = fish, y = elevation, fill = fish)) +
geom_boxplot() +
scale_x_discrete(limits = c('1', '0')) +
scale_fill_discrete(name = 'Fish presence', breaks = c('1', '0'))
```



Working with legends: Change legend labels

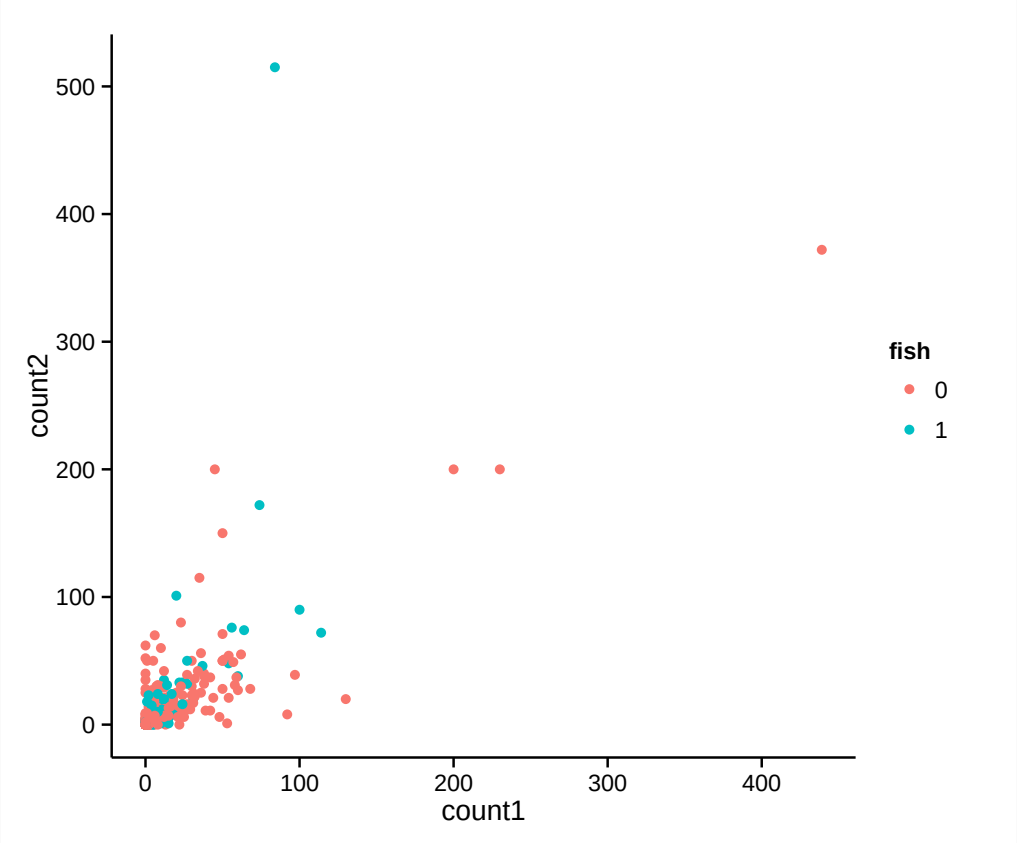
- use `name = <legendtitle>`

```
ggplot(frogs, aes(x = fish, y = elevation, fill = fish)) +
geom_boxplot() +
scale_x_discrete(limits = c('1', '0')) +
scale_fill_discrete(name = 'Fish presence', breaks = c('1', '0'), la-
bels = c('with fish', 'without fish'))
```

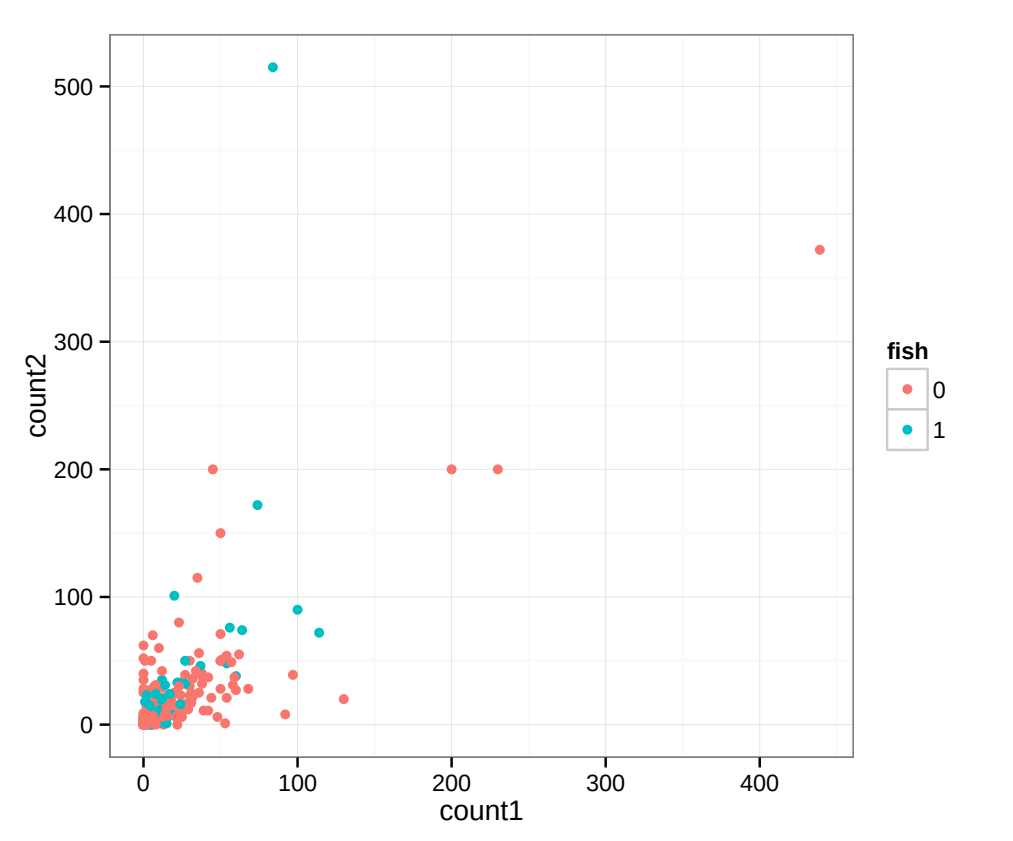


Changing the appearance of the plots: Themes

```
ggplot(frogs, aes(x = count1, y = count2, col = fish)) +
  geom_point() +
  theme_classic()
```

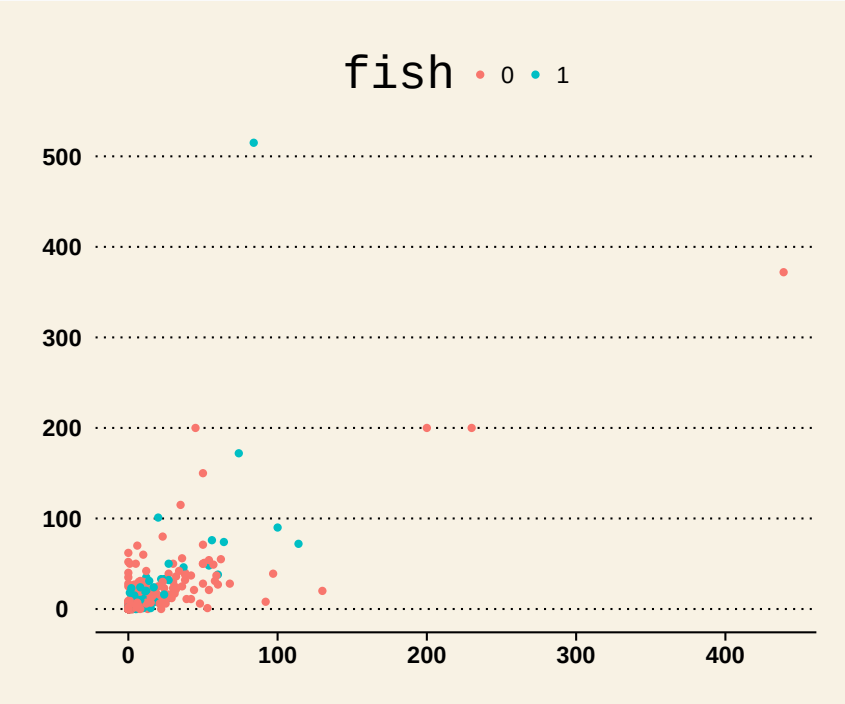


```
ggplot(frogs, aes(x = count1, y = count2, col = fish)) +
  geom_point() +
  theme_bw()
```



- ggplot2 comes with the themes: `bw`, `classic`, `gray`, `light` & `minimal`
- see `ggthemes` package for more `https://github.com/jrnold/ggthemes`

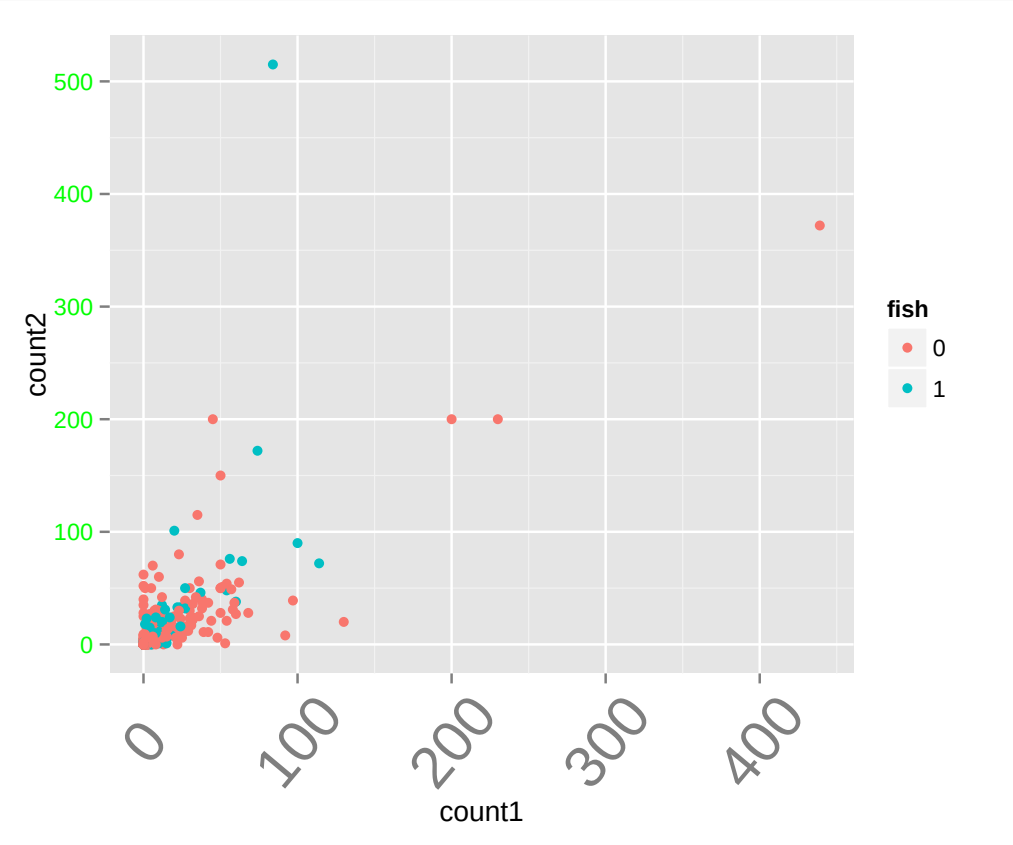
```
require(ggthemes)
ggplot(frogs, aes(x = count1, y = count2, col = fish)) +
  geom_point() +
  theme_wsj()
```



- Allows very fine control over your plots
- If done once, can be reused
- Every theme consists of several elements
- see `?theme` for a complete list
- we go through the most important elements one by one
- Each element consists of
  - element_text
  - element_line
  - element_rect
  - element_blank

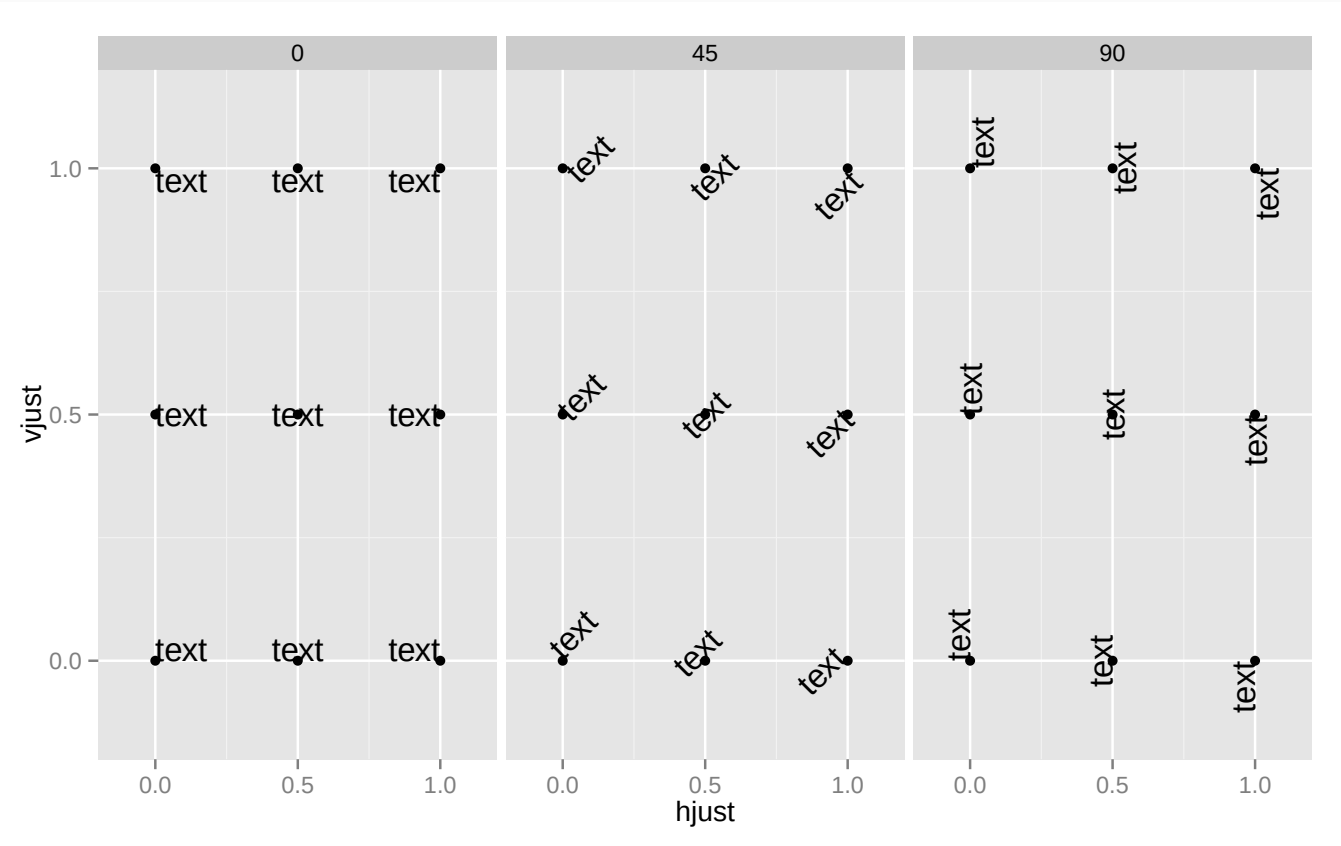
- each element can be controlled with `element_*`
- `text, line, rect, blank (=omit element)`

```
ggplot(frogs, aes(x = count1, y = count2, col = fish)) +
  geom_point() +
  theme(axis.text.x = element_text(angle = 50, size = 25, vjust = 0.5),
        axis.text.y = element_text(colour = 'green'))
```



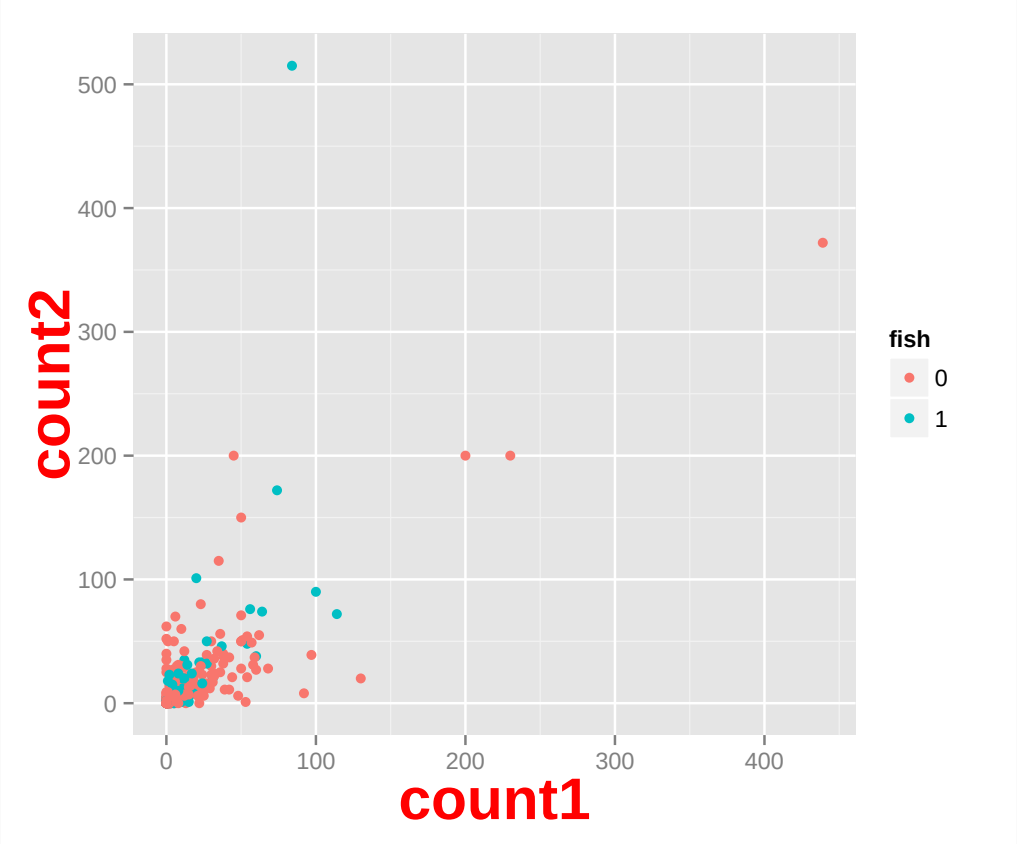
hjust, vjust, angle?!

- controls the vertical and horizontal justification
- 0 = left-justified; 1 = right-justified



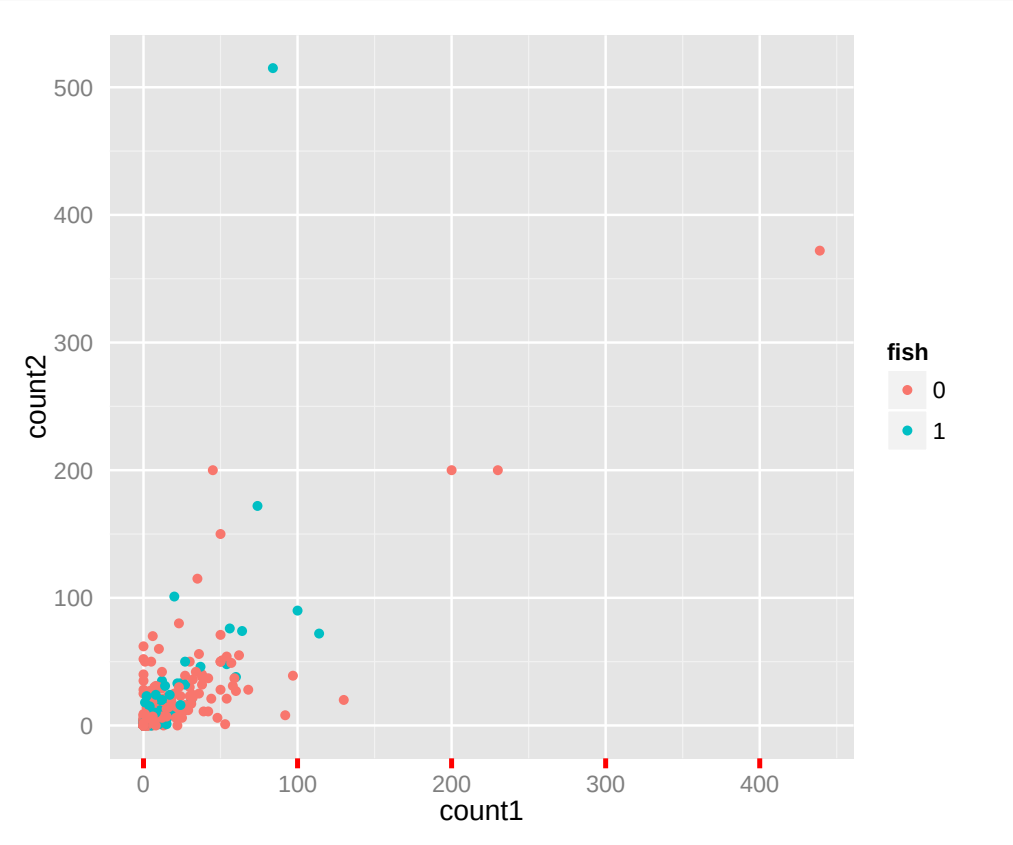
- if we omit the .x / .y the element is applied to both axes

```
ggplot(frogs, aes(x = count1, y = count2, col = fish)) +
  geom_point() +
  theme(axis.title = element_text(face = 'bold', colour = 'red', size = 25))
```



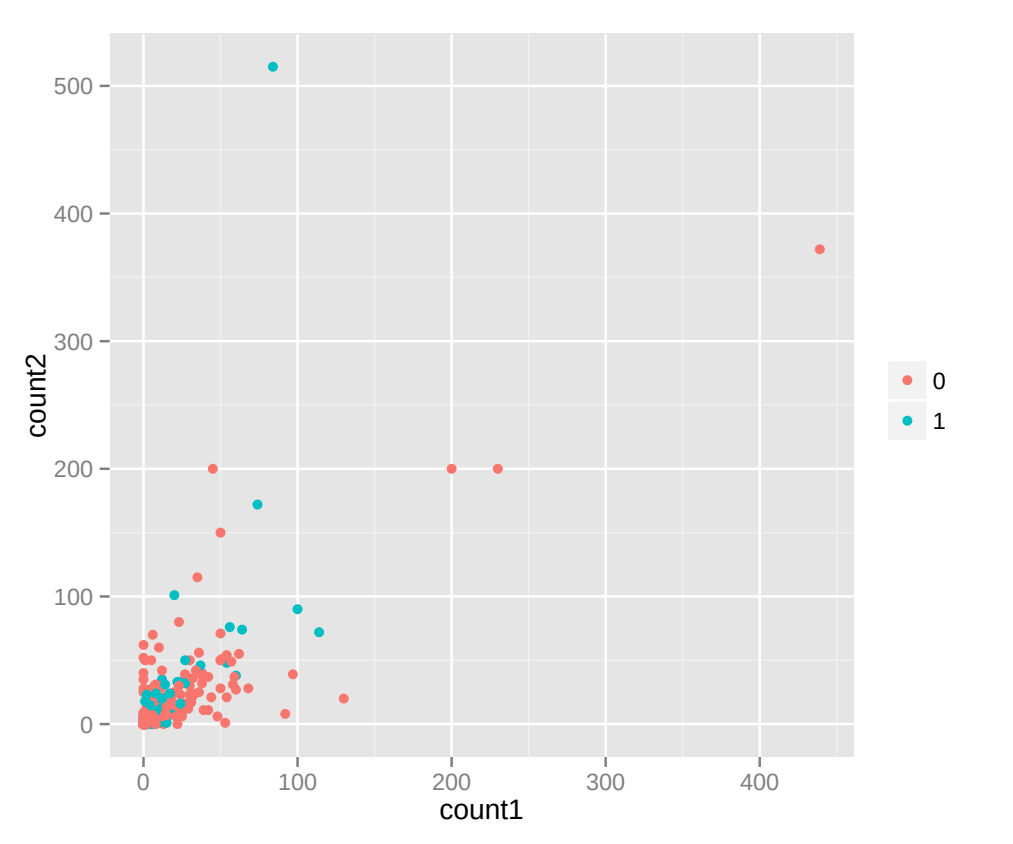
- `element_line()` for line elements.

```
ggplot(frogs, aes(x = count1, y = count2, col = fish)) +
  geom_point() +
  theme(axis.ticks.x = element_line(colour = 'red', size = 1),
        axis.ticks.y = element_blank())
```



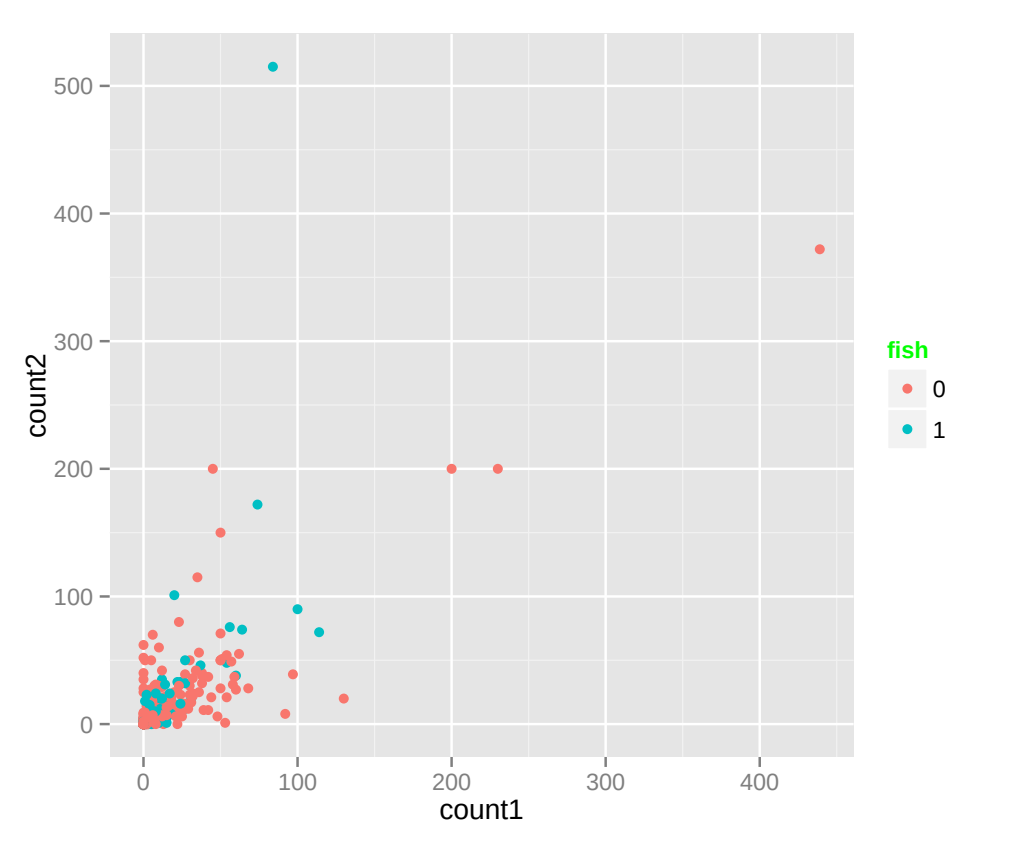
- `element_blank()` removes the element from the theme.

```
ggplot(frogs, aes(x = count1, y = count2, col = fish)) +
  geom_point() +
  theme(legend.title=element_blank())
```



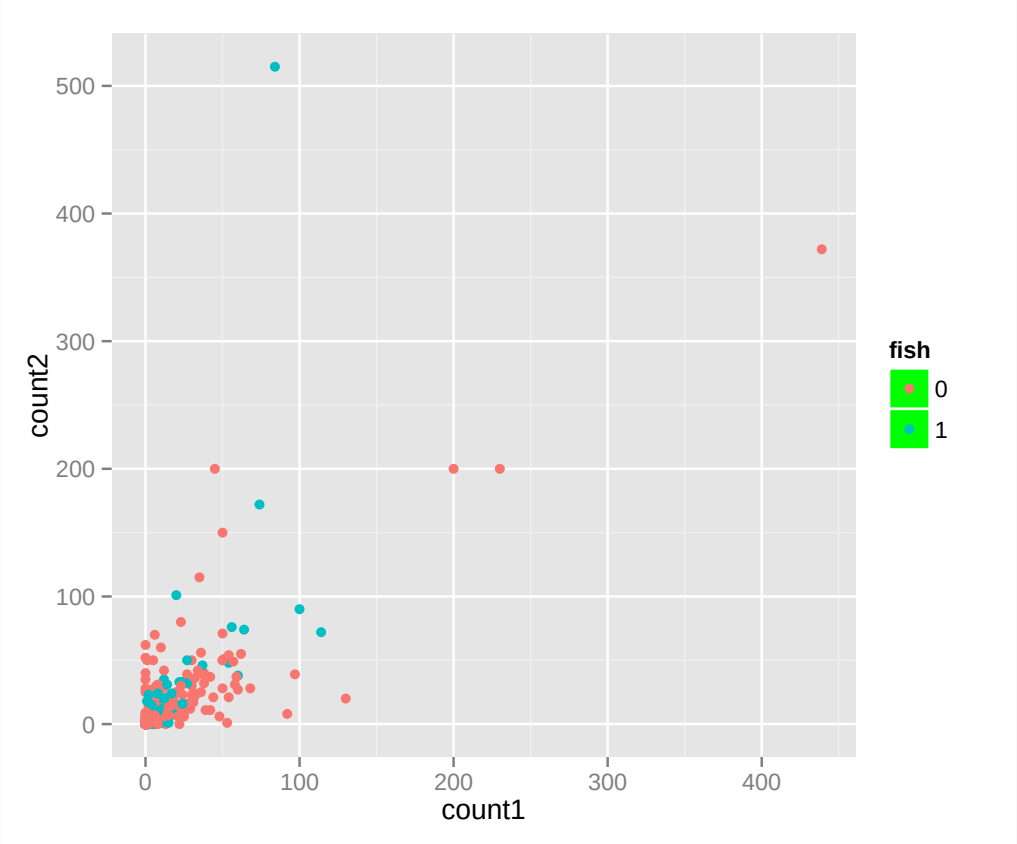
- `element_blank()` removes the element from the theme.

```
ggplot(frogs, aes(x = count1, y = count2, col = fish)) +
  geom_point() +
  theme(legend.title=element_text(colour = 'green'))
```



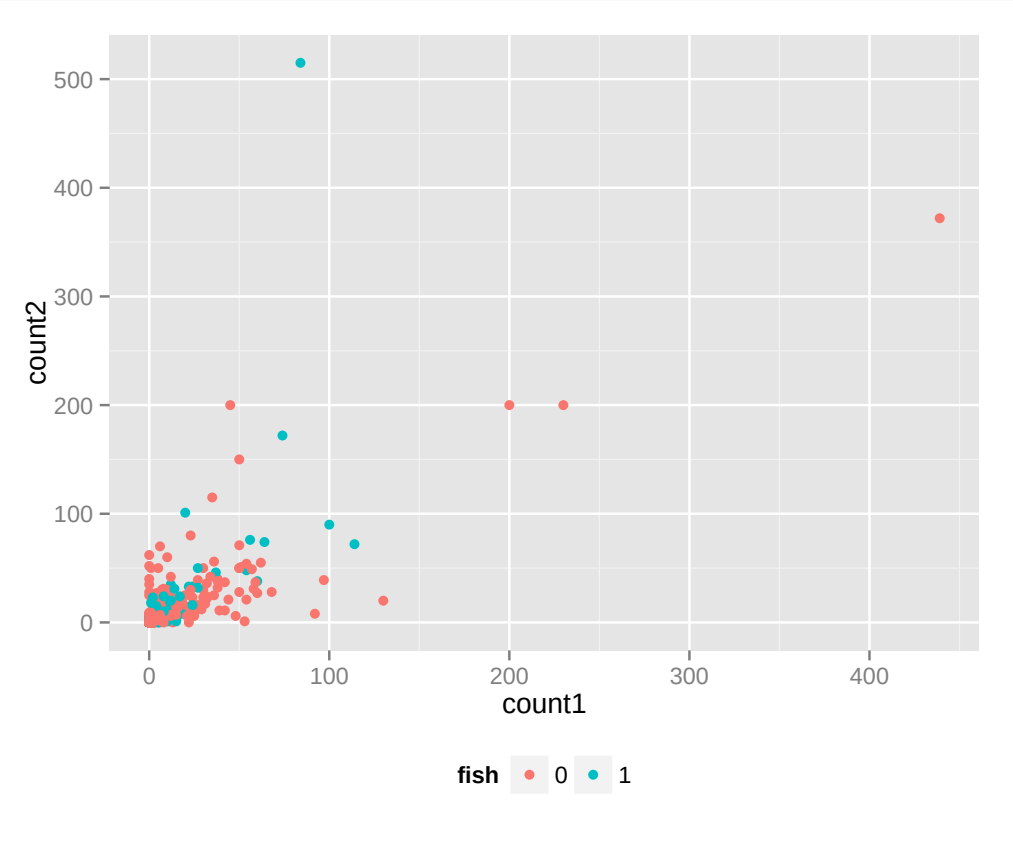
- `element_rect()` for rectangular elements.

```
ggplot(frogs, aes(x = count1, y = count2, col = fish)) +
  geom_point() +
  theme(legend.key = element_rect(fill = 'green'))
```



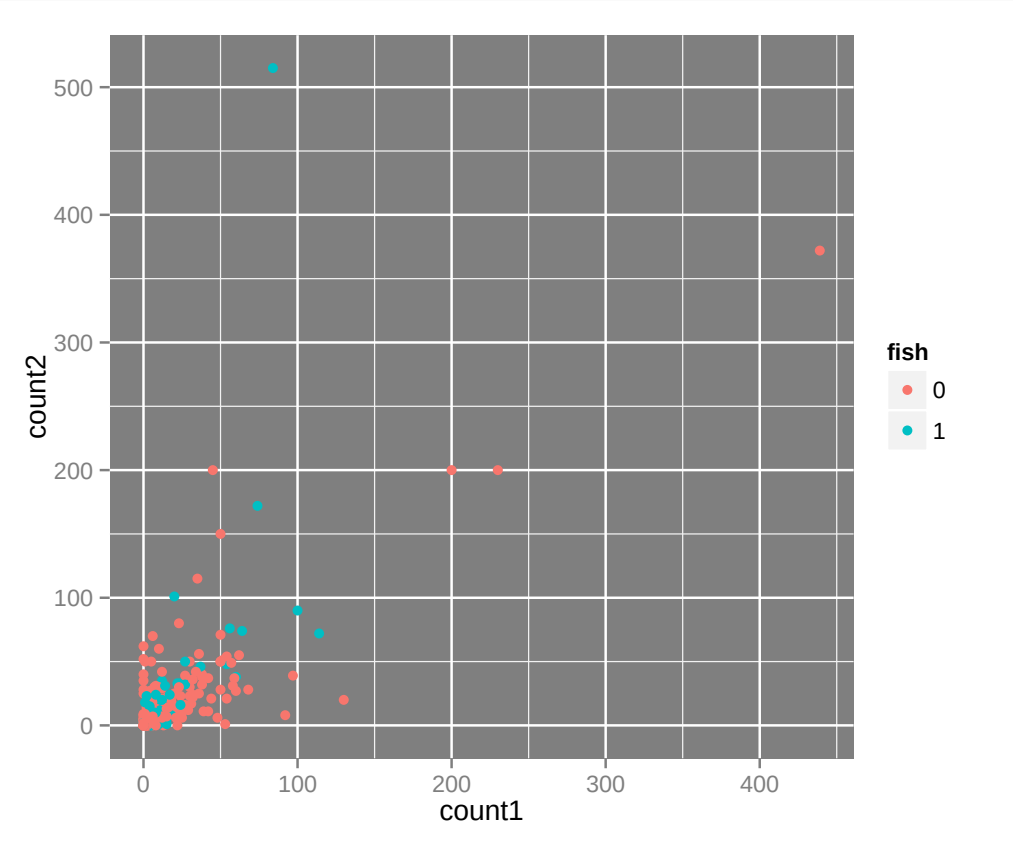
- position can be "none", "left", "right", "bottom", "top"
- also check legend.direction ("horizontal" or "vertical")

```
ggplot(frogs, aes(x = count1, y = count2, col = fish)) +
  geom_point() +
  theme(legend.position = 'bottom')
```

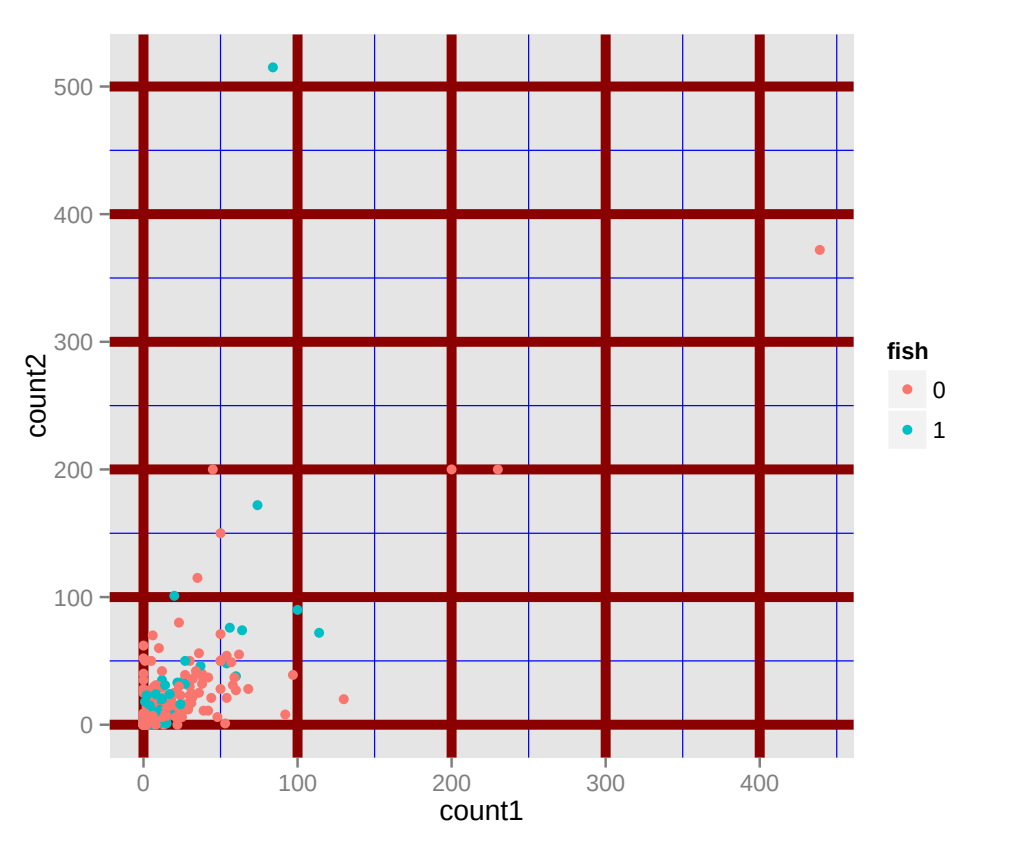


Themes: panel background

```
ggplot(frogs, aes(x = count1, y = count2, col = fish)) +
  geom_point() +
  theme(panel.background = element_rect(fill = 'grey50'))
```

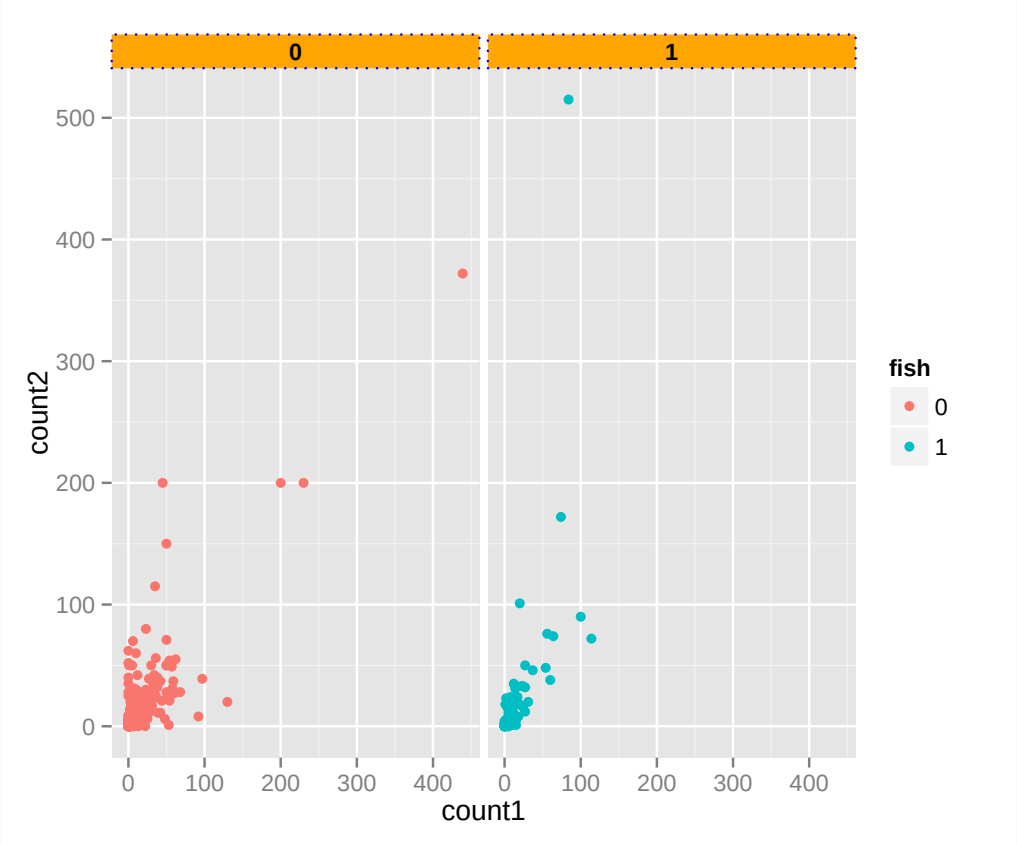


```
ggplot(frogs, aes(x = count1, y = count2, col = fish)) +
  geom_point() +
  theme(panel.grid.major = element_line(colour = 'darkred', size = 2),
        panel.grid.minor = element_line(color = 'blue'))
```



Themes: strips (facets)

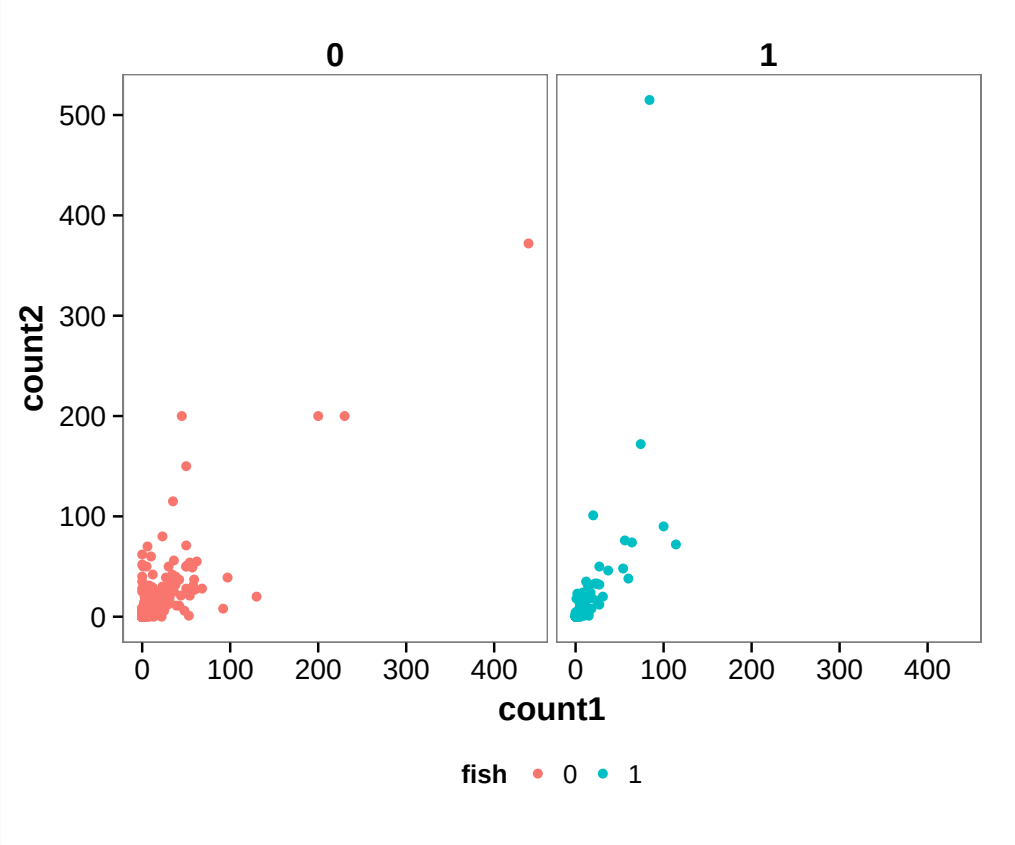
```
ggplot(frogs, aes(x = count1, y = count2, col = fish)) +
  geom_point() +
  facet_wrap(~fish) +
 theme(strip.background = element_rect(colour = 'blue', fill = 'orange', line-
type = 'dotted'),
       strip.text = element_text(face = 'bold'))
```



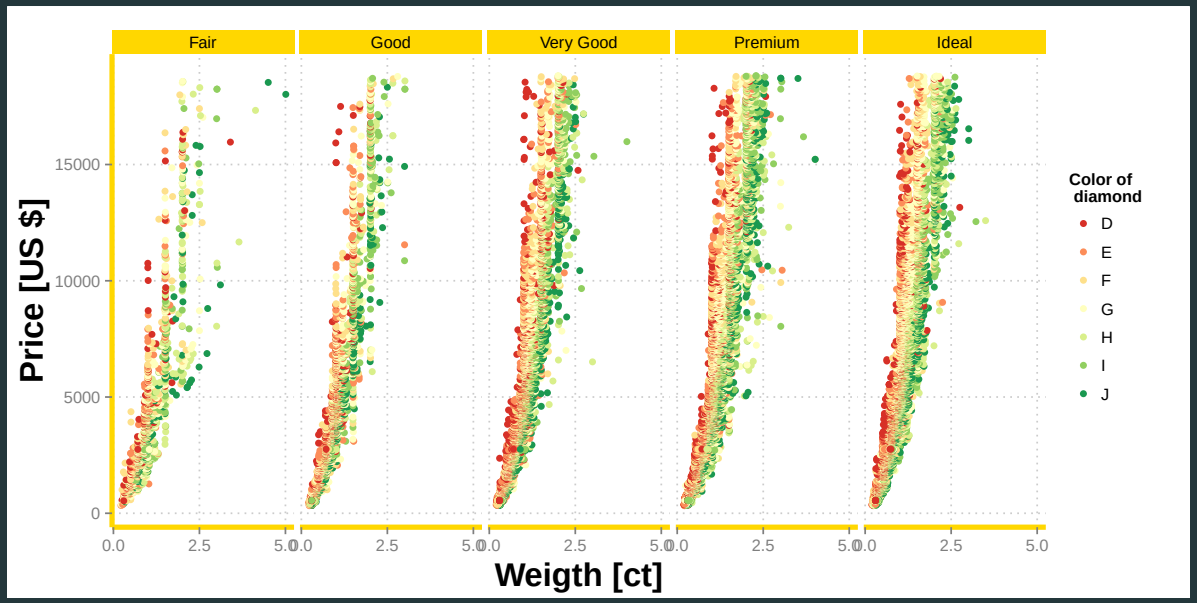
- my (often used) theme is based on `theme_bw()`
- bold and bigger fonts, cleaner strips
- safe your theme as object (or better in a separate .R file) and add it to the plot to apply it

```
edis_theme <- theme_bw(base_size = 12, base_family = "Helvetica") +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        text = element_text(size = 14),
        axis.text = element_text(size = 12),
        axis.title.x = element_text(size = 14,face = "bold", vjust = 0),
        axis.title.y = element_text(size = 14,face = "bold", vjust = 1),
        legend.position = "bottom",
        legend.key = element_blank(),
        strip.background = element_blank(),
        strip.text = element_text(size = 14, face = 'bold'))
```

```
ggplot(frogs, aes(x = count1, y = count2, col = fish)) +
  geom_point() +
  facet_wrap(~fish) +
  edis_theme
```



Exercise 6:
Create this plot:



Exporting your plots

Export your plot to a file

- save your plot as object
- use `ggsave()` to save the object

```
p <- ggplot(frogs, aes(x = count1, y = count2, col = fish)) +
  geom_point()
```

```
# export to different formats
ggsave('myplot.pdf', p)
ggsave('myplot.png', p)
ggsave('myplot.svg', p)
```

Use the vector format!

- use the vector format for graphics without loss
- either .pdf (no post-manipulation) or .svg (easy post-manipulation)
- For MS Office use high-resolution png:

```
ggsave('myplot.png', p, width = 7, heigh = 6, dpi = 300)
```

Use the vector format!

- My workflow:
  1. save as .svg
  2. (optional) manipulate in inkscape
  3. convert to needed format using inkscape (bash script)

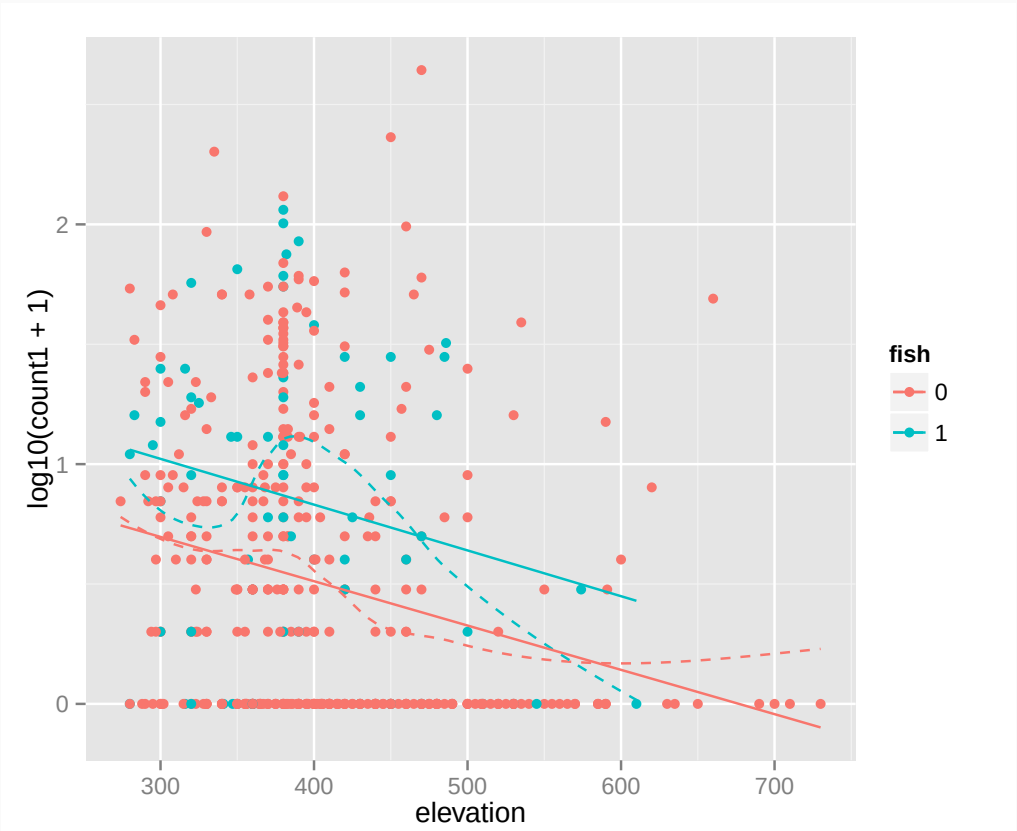
```
ggsave('myplot.svg', p, width = 7, heigh = 6)
```

Inkscape Demo.

Misc stuff

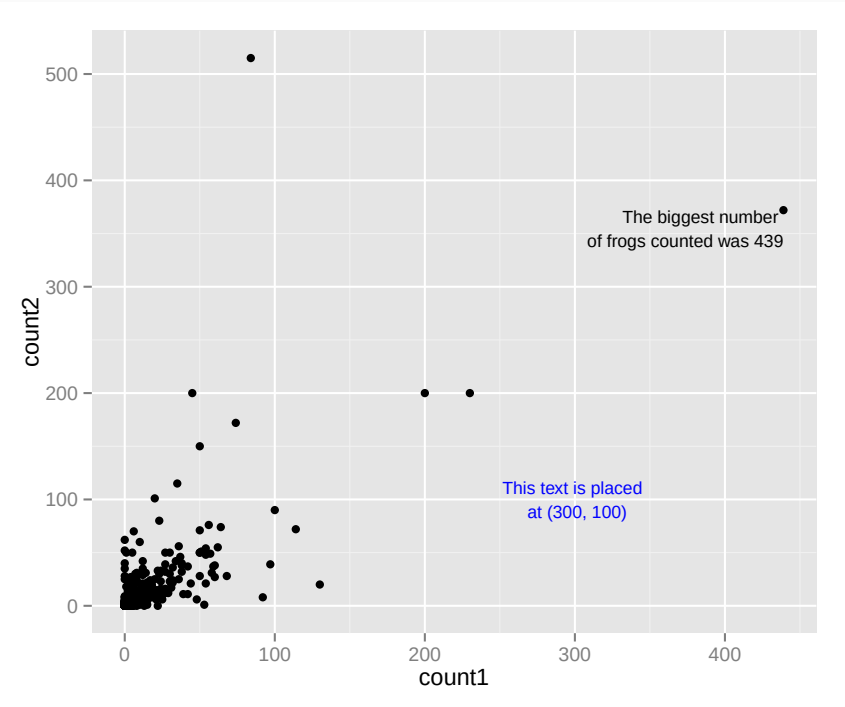
- `geom_smooth` is great to quickly visualize relationships
- I do not recommend the built-in model fitting functions
- better separate modelling from plotting

```
ggplot(frogs, aes(y = log10(count1  + 1), x = elevation, col = fish)) +
  geom_point() +
  geom_smooth(se = FALSE, linetype = 'dashed') +
  geom_smooth(se = FALSE, method = 'lm')
```



Annotate the plots

```
ggplot(frogs, aes(x = count1, y =count2)) +
  geom_point() +
  annotate('text',
           x = max(frogs$count1),
           y = frogs$count2[which.max(frogs$count1)],
          label = paste0('The biggest number \n of frogs counted was ',
                         max(frogs$count1)),
          vjust = 1, hjust = 1, size = 3) +
  annotate('text', x = 300, y = 100,
           label = 'This text is placed \n at (300, 100)',
           size = 3, col = 'blue')
```




THANKS FOR YOUR ATTENTION!
& A SUCCESSFUL CONFERENCE