*Angela M. Zoss, Ph.D.*

# *Visualization for Data Science with R*

To my family.

I'm so grateful for your support.

# *Contents*

# *List of Tables*

# *List of Figures*

# *Proposal*

**Note: This book is a work in progress, with a full draft expected in April of 2022.**

This book combines instruction on writing R code with building basic graphic design skills in a way that is unusual in data science literature. The book will guide readers through a series of projects, each designed to cover both how visualizations work in R and how visualizations can be designed to have the greatest impact. Far more than a "do this, then this" checklist, this book will focus on building understanding, confidence, and the ability to transfer skills to other tools and design contexts. It will avoid technical jargon that our target audience is unlikely to have encountered before. To accommodate learners who don't have time to work through an entire book, each chapter will operate independently, covering a specific set of tasks that all make sense together as part of a visualization project. For those who would like extra practice, there will be several types of hands-on exercises, from those that are entirely prescribed to those that allow readers to apply new techniques to problems in their own areas.

The book will have solutions (in the form of completed code and sample output) for all exercises. While not a textbook, the book will also include a brief teacher's guide for courses that might want to use one or more chapters to structure lessons in a course. The book will also have a website, including links to Open Access content, solutions, and related resources like video tutorials.

The target audience of this book would be professionals who are having to learn data science techniques on the job, likely at an under-resourced organization or company. These newly minted data professionals may feel comfortable in Excel but have only just started to learn R for processing data. They have never used a programming language to build a visualization before, and even creating charts in Excel has often been a frustrating and mystifying process. They appreciate that R is freely available and are able to get started on a data science project, but the idea of creating publication-quality visualizations using only code is daunting.

Increasingly, programs of study with a focus on preparing students for professional careers in under-resourced fields, like public policy and even management, include courses on data analysis and communication using freely available software. This book, while not a textbook, could easily be used for a semester-long course, titled something like "Practical data visualization for

the modern workforce." A chapter could be covered each week, and larger projects could help learners synthesize chapters into a complete set of analyses and communication materials.

## Why read this book

This book will be:

- Written for non-academics, beginning programmers
- Each chapter stands alone
- Covers pressing modern issues, like accessibility and ethics
- Focuses on freely available software
- Combines hands-on exercises with basic graphic design principles

## Structure of the book

- Chapter 1: Overview of common visualizations and how to read them
- Chapter 2: Building basic visualizations with ggplot2
- Chapter 3: Working with textual data in ggplot2
- Chapter 4: Customizing the design of ggplot2 visualizations
- Chapter 5: Avoiding unethical design practices
- Chapter 6: Building ggplot2 visualizations into print publications
- Chapter 7: Basic accessibility for static visualizations
- Chapter 8: Exploring interactivity in visualizations with plotly and crosstalk
- Chapter 9: Using RMarkdown to build websites for projects
- Chapter 10: Using RMarkdown to build dashboards for projects
- Chapter 11: Basic usability for interactive visualizations
- Chapter 12: Teacher's guide

## Software information and conventions

I used the **knitr** package (Xie, 2015) and the **bookdown** package (Xie, 2021) to compile my book. My R session information is shown below:

```
xfun::session_info()
```

```
## R version 4.1.0 (2021-05-18)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur 10.16
##
## Locale: en_US.UTF-8 / en_US.UTF-8 / en_US.UTF-8 / C / en_US.UTF-
8 / en_US.UTF-8
##
## Package version:
##   base64enc_0.1.3   bookdown_0.23
##   compiler_4.1.0    digest_0.6.27
##   evaluate_0.14     glue_1.4.2
##   graphics_4.1.0    grDevices_4.1.0
##   highr_0.9         htmltools_0.5.1.1
##   jquerylib_0.1.4   jsonlite_1.7.2
##   knitr_1.33        magrittr_2.0.1
##   markdown_1.1      methods_4.1.0
##   mime_0.11         rlang_0.4.11
##   rmarkdown_2.10    rstudioapi_0.13
##   stats_4.1.0       stringi_1.7.3
##   stringr_1.4.0     tinytex_0.33
##   tools_4.1.0       utils_4.1.0
##   xfun_0.25         yaml_2.2.1
```

Package names are in bold text (e.g., **rmarkdown**), and inline code and filenames are formatted in a typewriter font (e.g., `knitr::knit('foo.Rmd')`). Function names are followed by parentheses (e.g., `bookdown::render_book()`).

Angela Zoss

# *About the Author*

**FIGURE 1:** Angela M. Zoss, Ph.D.

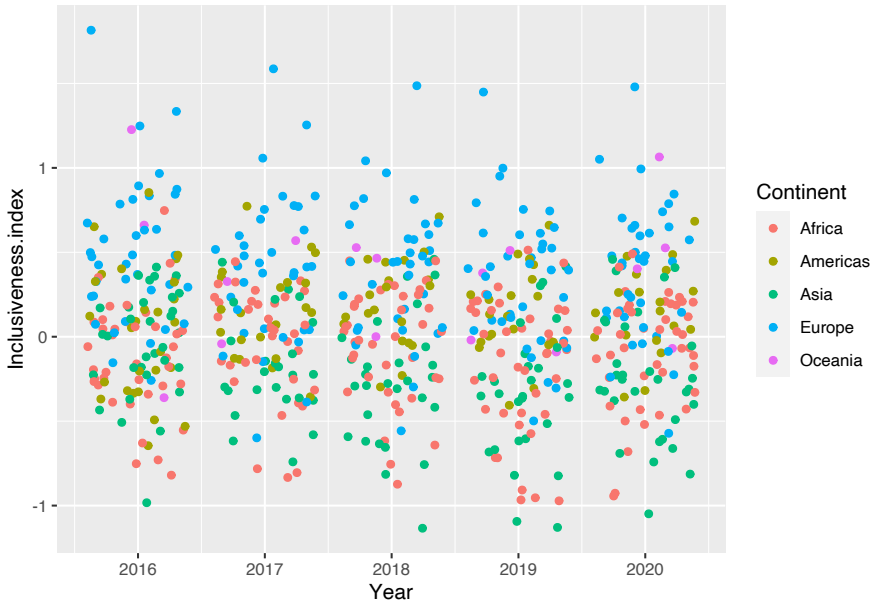
Angela is the Assessment & Data Visualization Analyst[1] in the Assessment & User Experience Department[2] in the Duke University Libraries[3]. She has many years of experience in teaching and training, predominantly focusing on teaching data visualization to university students, faculty, and staff. She is also active in several open source development projects, including FOLIO[4] and Wax[5].

[1] https://library.duke.edu/about/directory/staff/angela.zoss
[2] https://library.duke.edu/about/depts/assessment-user-experience
[3] https://library.duke.edu/
[4] https://github.com/folio-org/
[5] https://github.com/minicomp/wax

# 1

# *Overview of common visualizations and how to read them*

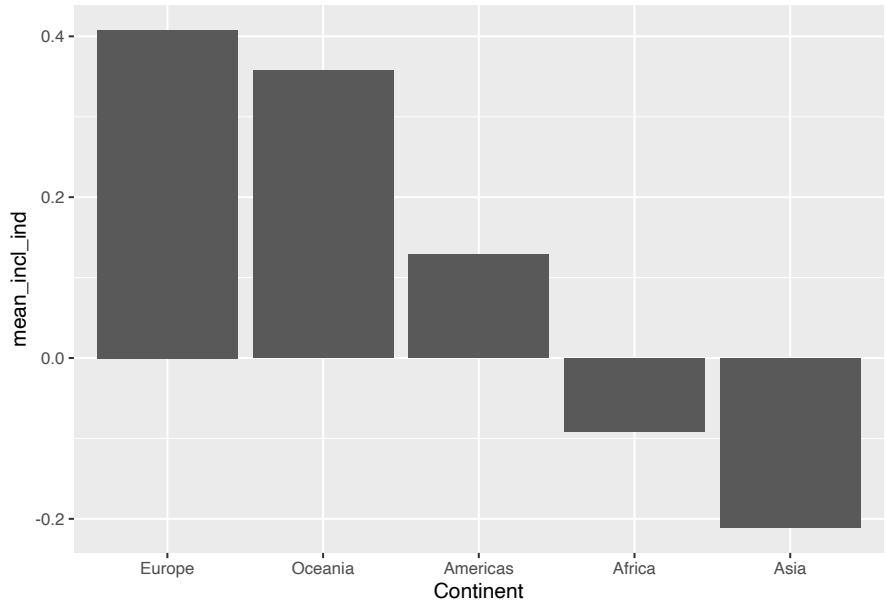
```
inclusiveness_index_annual %>%
  drop_na(Inclusiveness.index) %>%
  ggplot(aes(x = Year, y = Inclusiveness.index, color = Continent)) +
    geom_jitter()
```




## 1.1 Bar Chart

bar chart / stacked bar / dot plot / dumbbell plot

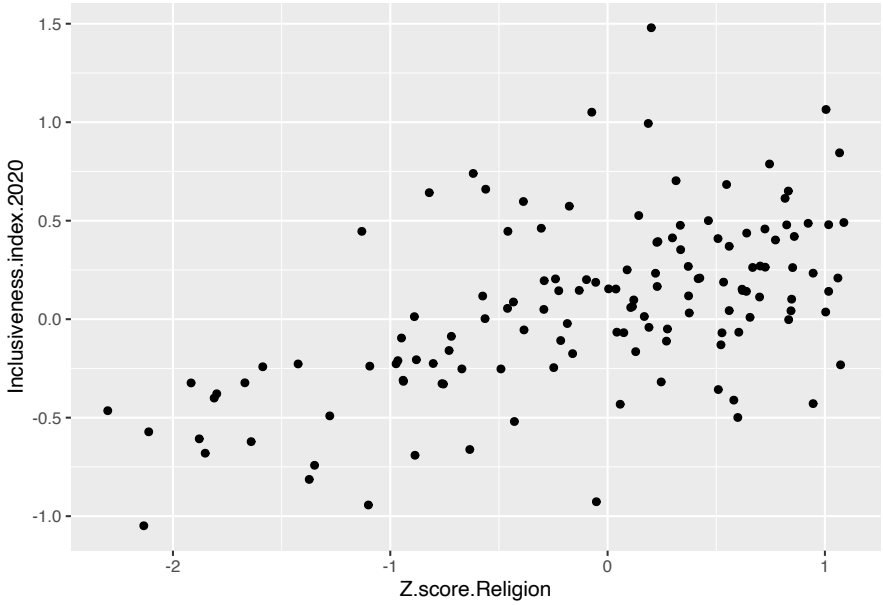
```
inclusiveness_index_annual %>%
  drop_na(Inclusiveness.index) %>%
  group_by(Continent) %>%
  summarize(mean_incl_ind = mean(Inclusiveness.index, na.rm = TRUE)) %>%
  mutate(Continent = Continent %>% as_factor() %>% fct_reorder(mean_incl_ind, .desc = TRUE)) %>%
ggplot(mapping = aes(x = Continent, y = mean_incl_ind)) +
  geom_col()
```




## 1.2 Scatter Plot

scatter plot / scatter plot with color / bubble chart / countour plot

```
inclusiveness_index %>%
  drop_na(Z.score.Religion, Inclusiveness.index.2020) %>%
  ggplot(aes(x = Z.score.Religion,
             y = Inclusiveness.index.2020)) +
    geom_point()
```

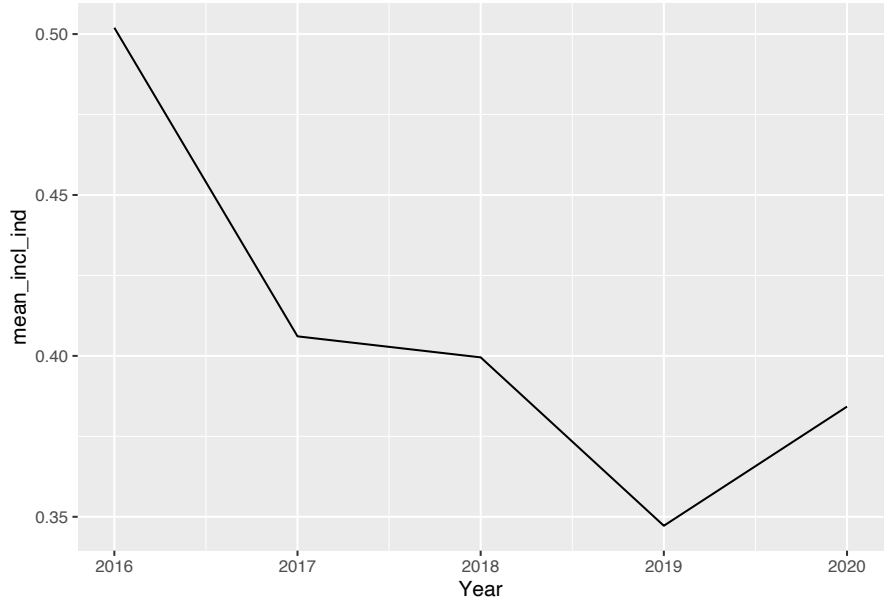



## 1.3 Line Chart

line chart / area chart

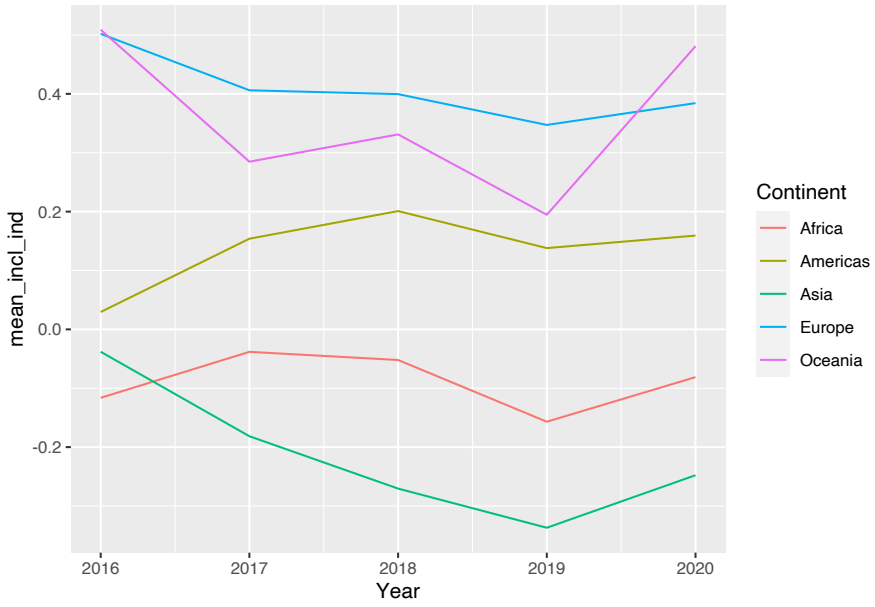
```
inclusiveness_index_annual %>%
  group_by(Year, Continent) %>%
  summarise(mean_incl_ind = mean(Inclusiveness.index, na.rm = TRUE)) %>%
  dplyr::filter(Continent == "Europe") %>%
ggplot(aes(x = Year,
           y = mean_incl_ind)) +
  geom_line()
```

```
## `summarise()` has grouped output by 'Year'. You can override using the `.groups` argument.
```




```
inclusiveness_index_annual %>%
  group_by(Year, Continent) %>%
  summarise(mean_incl_ind = mean(Inclusiveness.index, na.rm = TRUE)) %>%
  ggplot(aes(x = Year,
             y = mean_incl_ind,
             color = Continent)) +
    geom_line()
```

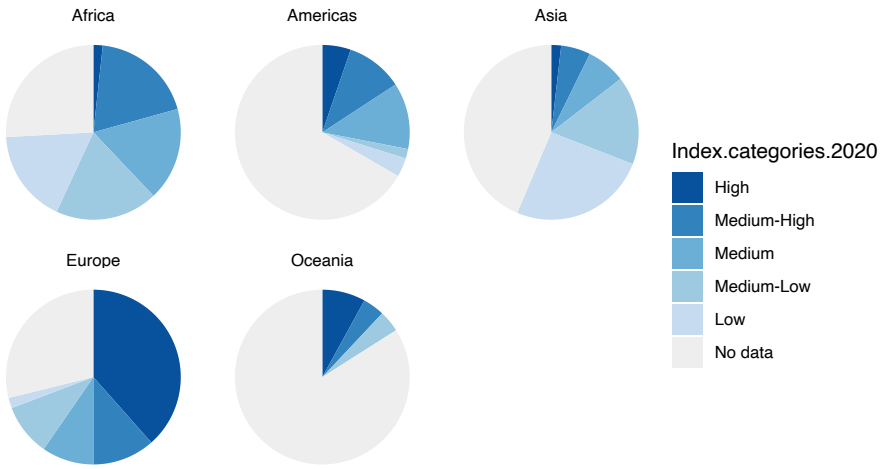
```
## `summarise()` has grouped output by 'Year'. You can override using the `.groups` argument.
```




## 1.4 Pie Chart

pie chart / donut chart

```
# library(RColorBrewer)
# brewer.pal(5, "Blues")

inclusiveness_index %>%
  #dplyr::filter(Continent == "Americas") %>%
  ggplot(aes(y = "", fill = Index.categories.2020)) +
    geom_bar(position=position_fill()) +
    coord_polar(direction = -1) +
    #scale_fill_brewer(type="seq", palette = 1, direction = -1) +
    scale_fill_manual(values = c("#08519C","#3182BD","#6BAED6","#9ECAE1","#C6DBEF","#EEEEEE")) +
    theme_void() +
    facet_wrap(vars(Continent))
```




## 1.5 Heat Map

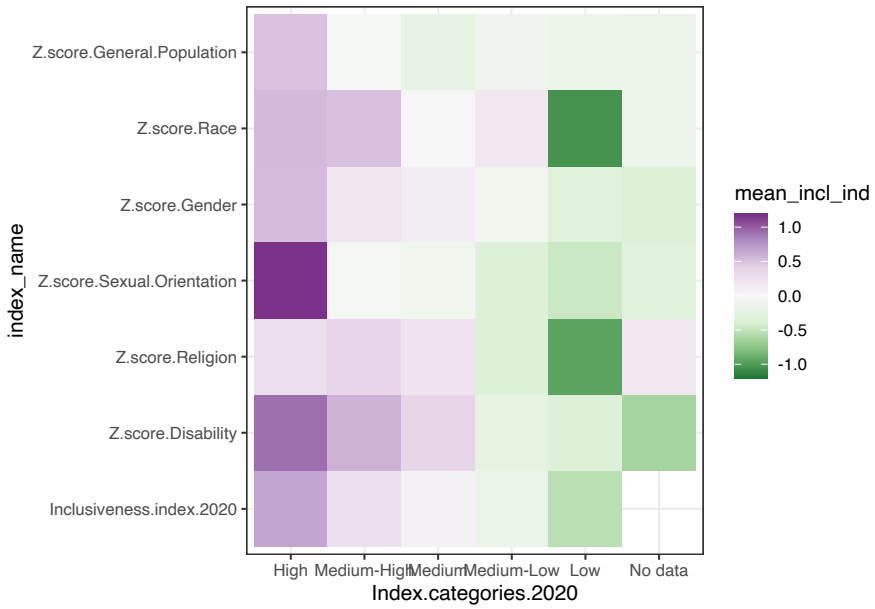
heat map / matrix / circles with color and size

```
inclusiveness_index %>%
  pivot_longer(cols = c(Inclusiveness.index.2020, starts_with("Z")),
               names_to = "index_name",
               values_to = "index_values",
               values_drop_na = TRUE) %>%
  group_by(Index.categories.2020, index_name) %>%
  summarise(mean_incl_ind = mean(index_values)) %>%
  mutate(index_name = index_name %>%
           as_factor() %>%
           fct_relevel(c("Z.score.General.Population", "Z.score.Race", "Z.score.Gender",
                         "Z.score.Sexual.Orientation", "Z.score.Religion",
                         "Z.score.Disability", "Inclusiveness.index.2020")) %>%
           fct_rev()) %>%
  #View() %>%
  ggplot(aes(y = index_name, x = Index.categories.2020, fill = mean_incl_ind)) +
```

```
    geom_tile() +
    scale_fill_distiller(type="div", palette = 3,
                         limits=c(-1.2,1.2),
                         direction = -1) +
  theme_bw()
```

```
## `summarise()` has grouped output by 'Index.categories.2020'. You can override using the `.groups` ar
```

```
## Warning: Unknown levels in `f`:
## Inclusiveness.index.2020
```



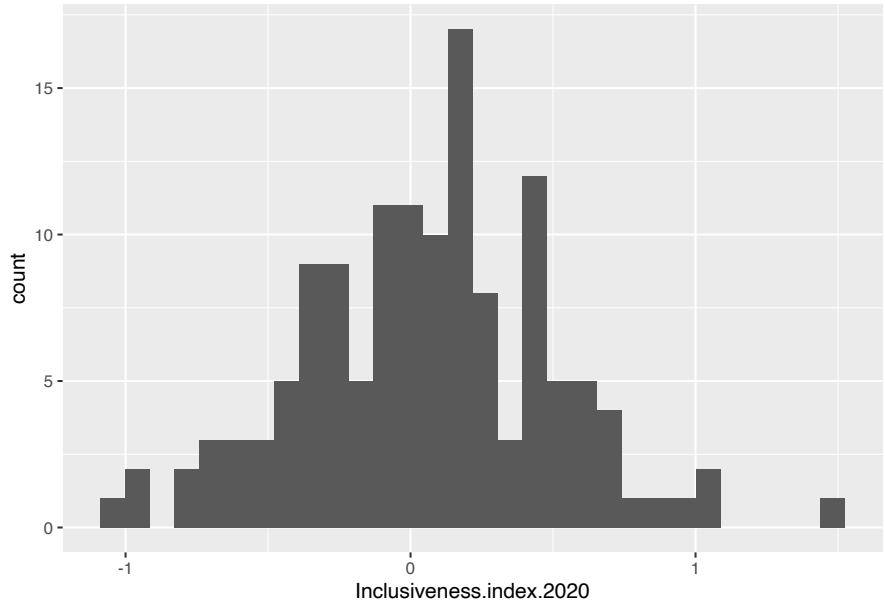

## 1.6 Histogram

histogram / density

```
inclusiveness_index %>%
  drop_na(Inclusiveness.index.2020) %>%
```

```
  ggplot(aes(x = Inclusiveness.index.2020)) +
    geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value
## with `binwidth`.
```

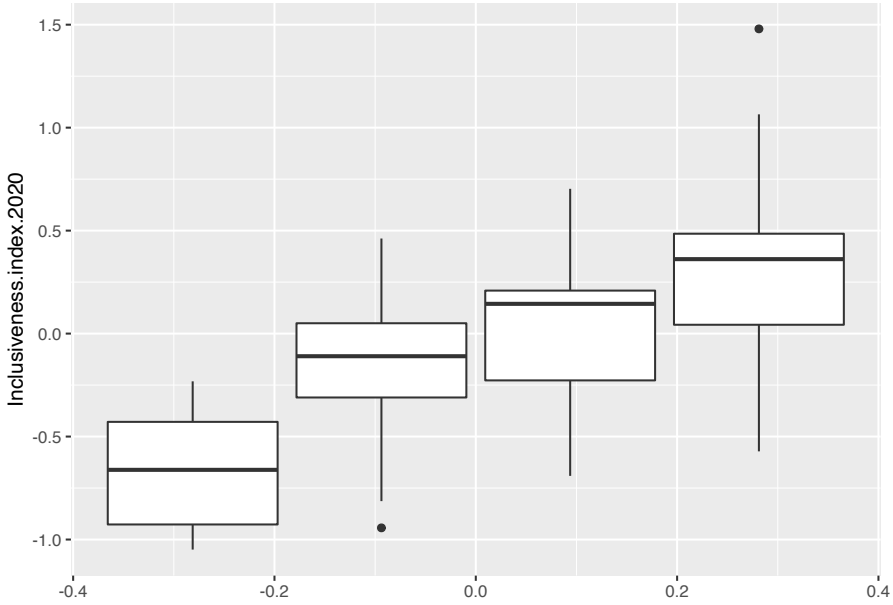



## 1.7 Box Plot

box plot / violin plot / bee swarm

```
ggplot(inclusiveness_index,
       aes(group = Z.score.Disability,
           y = Inclusiveness.index.2020)) +
  geom_boxplot()
```

```
## Warning: Removed 113 rows containing non-finite values
## (stat_boxplot).
```

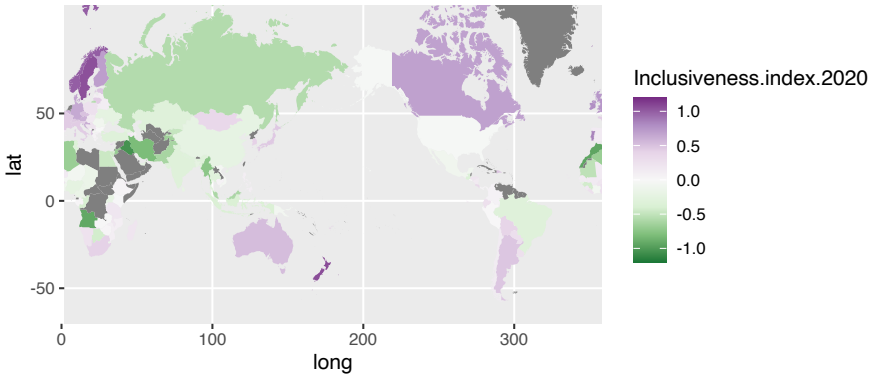



## 1.8 Maps

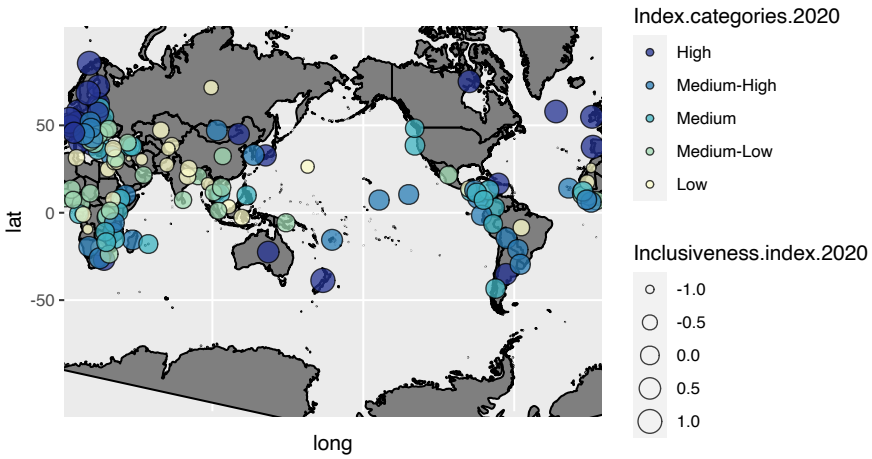
choropleth / proportional symbol map

```
map_incl_ind <- inclusiveness_index %>%
  mutate(Country = case_when(
    Country == "Antigua and Barbuda" ~ "Antigua",
    Country == "Bonaire, Saint Eustatius and Saba" ~ "Bonaire",
    Country == "Cote d'Ivoire" ~ "Ivory Coast",
    Country == "East Timor" ~ "Timor-Leste",
    Country == "Palestina" ~ "Palestine",
    Country == "Saint Kitts and Nevis" ~ "Saint Kitts",
    Country == "Saint Vincent and the Grenadines" ~ "Saint Vincent",
    Country == "Saint-Barthalemy" ~ "Saint Barthelemy",
    Country == "Saint-Martin" ~ "Saint Martin",
    Country == "Trinidad and Tobago" ~ "Trinidad",
    Country == "United Kingdom" ~ "UK",
    Country == "United States" ~ "USA",
    Country == "Vatican City" ~ "Vatican",
```

```
    TRUE ~ Country
  )) %>%
  bind_rows(
    inclusiveness_index %>% dplyr::filter(
      Country %in% c("Antigua and Barbuda", "Bonaire, Saint Eustatius and Saba",
                     "Saint Kitts and Nevis", "Saint Vincent and the Grenadines",
                     "Trinidad and Tobago")
    ) %>% mutate(Country = case_when(
      Country == "Antigua and Barbuda" ~ "Barbuda",
      Country == "Bonaire, Saint Eustatius and Saba" ~ "Sint Eustatius",
      Country == "Saint Kitts and Nevis" ~ "Nevis",
      Country == "Saint Vincent and the Grenadines" ~ "Grenadines",
      Country == "Trinidad and Tobago" ~ "Tobago",
      TRUE ~ Country
      )
    ),
    inclusiveness_index %>% dplyr::filter(
      Country %in% c("Bonaire, Saint Eustatius and Saba")
    ) %>% mutate(Country = case_when(
      Country == "Bonaire, Saint Eustatius and Saba" ~ "Saba",
      TRUE ~ Country
      )
    )
  )

map_incl_ind %>%
  left_join(map_data("world2"), by = c("Country" = "region")) %>%
  ggplot() +
    geom_polygon(aes(x = long,
                     y = lat,
                     group=group,
                     fill=Inclusiveness.index.2020)) +
    coord_map() +
    scale_fill_distiller(type="div", palette = 3,
                         limits=c(-1.2,1.2),
                         direction = -1)
```



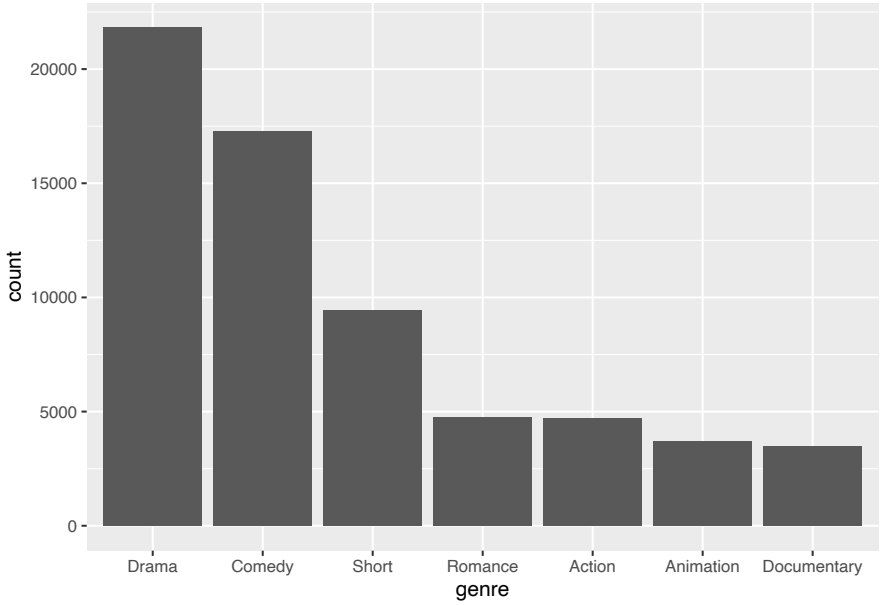

```
map_incl_ind %>%
  drop_na(Inclusiveness.index.2020) %>%
  left_join(
    map_data("world2") %>%
      group_by(region) %>%
      summarise(x = mean(long),
                y = mean(lat)),
    by = c("Country" = "region")) %>%
  ggplot() +
    geom_polygon(
      data = map_data("world2"),
      aes(x = long,
          y = lat,
          group=group),
      fill = "grey50",
      color = "black") +
    coord_map() +
    geom_point(aes(x = x, y = y,
                   fill = Index.categories.2020,
                   size = Inclusiveness.index.2020),
               shape = 21,
               alpha = .75) +
    scale_fill_brewer(palette = "YlGnBu", direction = -1)
```




## 1.9 Movies

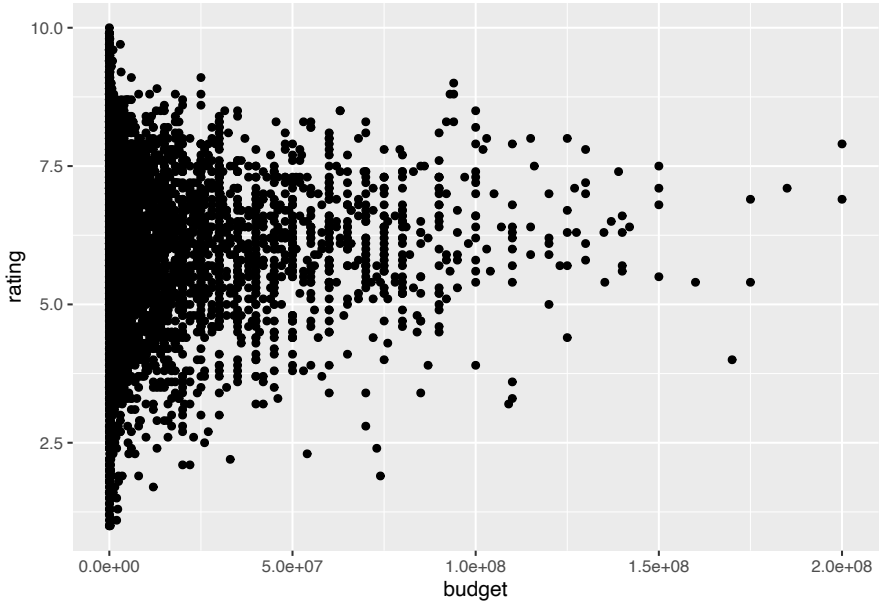
```
library(ggplot2movies) # add to index if you use this instead

# Movies were selected for inclusion if they had a known length and had
# been rated by at least one imdb user.
movies <- movies
```

```
movies %>%
  pivot_longer(cols = c(Action, Animation, Comedy, Drama, Documentary, Romance, Short), names_to =
  mutate(value = as.logical(value)) %>%
  dplyr::filter(value) %>%
  mutate(genre = genre %>%
           as_factor() %>%
           fct_infreq) %>%
  ggplot(aes(x=genre)) +
    geom_bar()
```



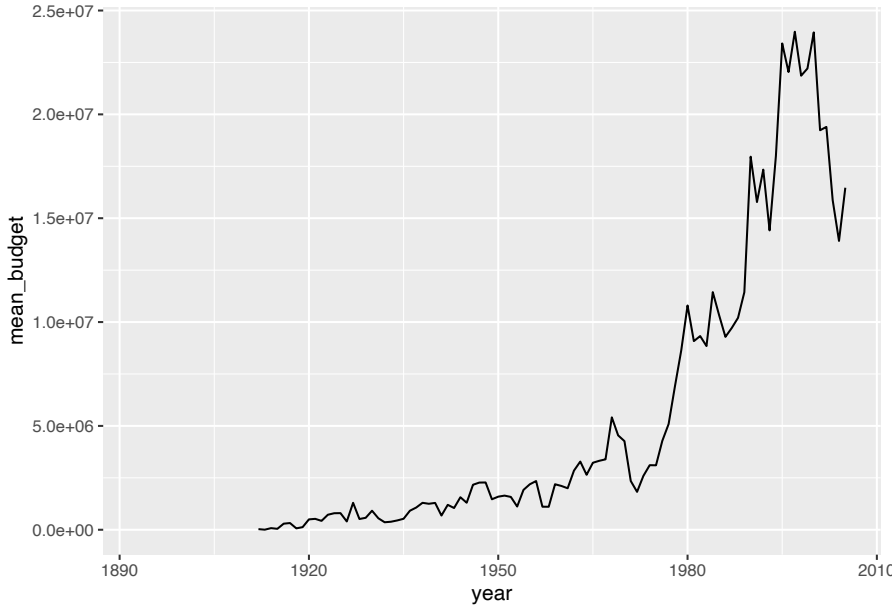

```
ggplot(movies, aes(budget, rating)) +
  geom_point()
```

```
## Warning: Removed 53573 rows containing missing values
## (geom_point).
```

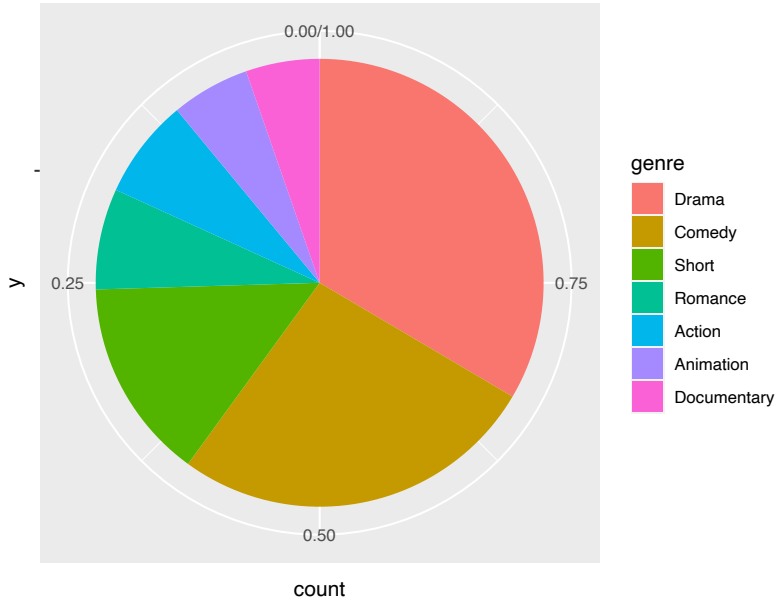



```
movies %>%
  group_by(year) %>%
  summarise(mean_budget = mean(budget, na.rm=T)) %>%
  ggplot(aes(x=year, y=mean_budget)) +
    geom_line()
```

```
## Warning: Removed 10 row(s) containing missing values
## (geom_path).
```

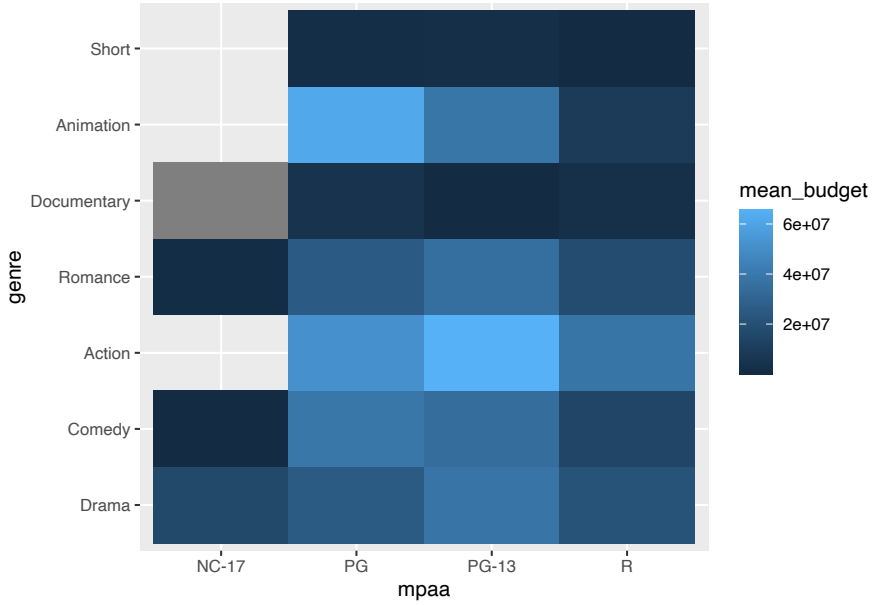



```
movies %>%
  pivot_longer(cols = c(Action, Animation, Comedy, Drama, Documentary, Romance, Short), names_to =
  mutate(value = as.logical(value)) %>%
  dplyr::filter(value) %>%
  mutate(genre = genre %>%
           as_factor() %>%
           fct_infreq) %>%
  ggplot(aes(y="", fill=genre)) +
    geom_bar(position = position_fill()) +
    coord_polar(direction = -1)
```



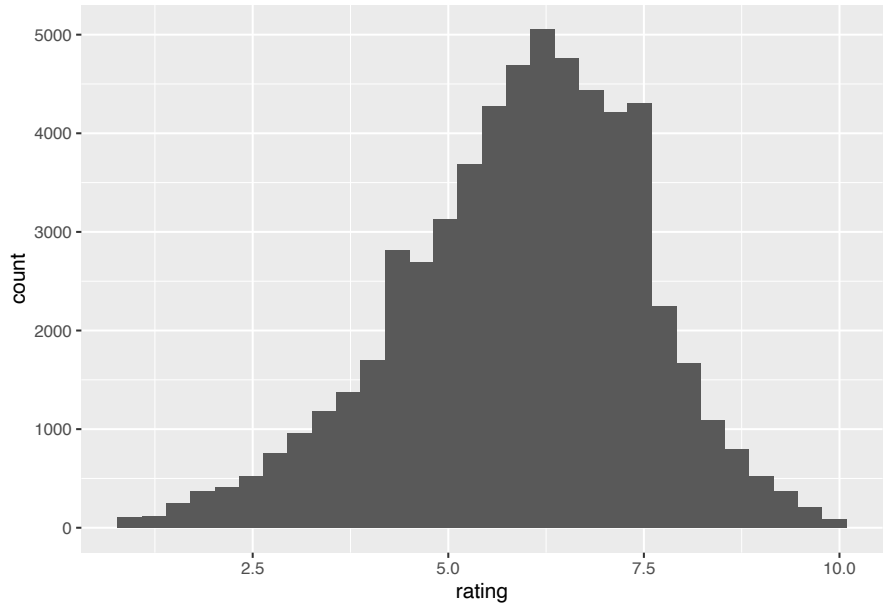

```
movies %>%
  pivot_longer(cols = c(Action, Animation, Comedy, Drama, Documentary, Romance, Short), names_to =
  mutate(value = as.logical(value)) %>%
  dplyr::filter(value, mpaa != "") %>%
  mutate(genre = genre %>%
           as_factor() %>%
           fct_infreq) %>%
  group_by(mpaa, genre) %>%
  summarize(n = n(), mean_budget = mean(budget, na.rm=T)) %>%
  ggplot(aes(x=mpaa, y=genre, fill = mean_budget)) +
    geom_tile()
```

```
## `summarise()` has grouped output by 'mpaa'. You can override using the `.groups` argument.
```

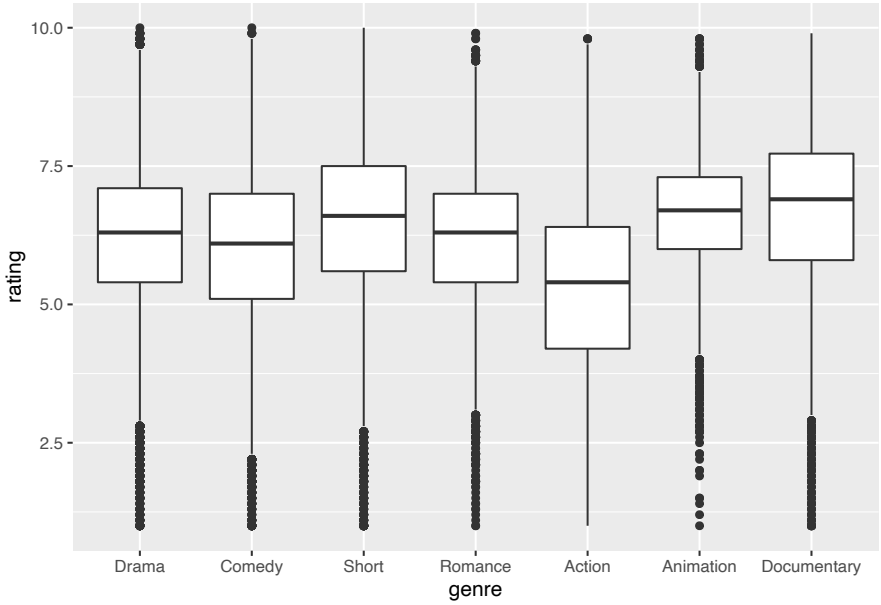



```
ggplot(movies, aes(x=rating)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value
## with `binwidth`.
```




```
movies %>%
  pivot_longer(cols = c(Action, Animation, Comedy, Drama, Documentary, Romance, Short), names_to =
  mutate(value = as.logical(value)) %>%
  dplyr::filter(value) %>%
  mutate(genre = genre %>%
           as_factor() %>%
           fct_infreq) %>%
  ggplot(aes(x=genre, y=rating)) +
    geom_boxplot()
```

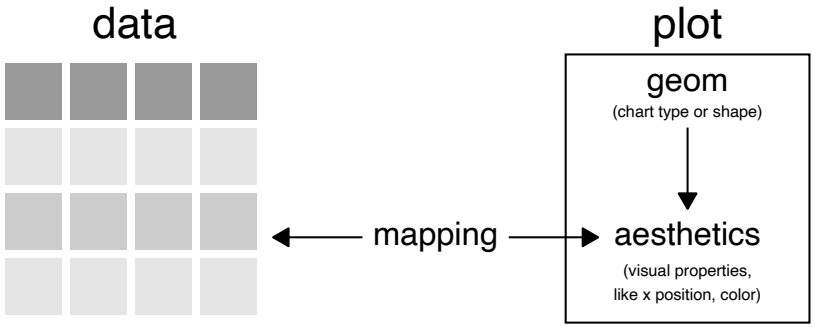



(No maps, but maybe that's okay)

# 2

# *Building basic visualizations with ggplot2*

## 2.1 Basic ggplot2 syntax




```
# start with the main plot function and set the data frame
ggplot(data = data_frame) +
# next add a geometry layer and create the mapping between data frame and aesthetics
  geom_...(mapping = aes(...))

# to create the mapping, compose a list of aesthetics assignments
# using the following format:
#   aesthetics = data_field
```

# 3

# *Working with textual data in ggplot2*

*sample text*

Cleaning data: use duke_enrollment (either by status or school) to talk about factors. Have Semester, which is really a time-based variable. Need to combine with Year to get the real sequence of enrollment.

# 4

# *Customizing the design of ggplot2 visualizations*

*sample text*

We talk about the *FOO* method in this chapter.

# 5

# *Avoiding unethical design practices*

*sample text*

We talk about the *FOO* method in this chapter.

# 6

# *Building ggplot2 visualizations into print publications*

*sample text*

We talk about the *FOO* method in this chapter.

# 7

# *Basic accessibility for static visualizations*

## 7.1 Low Vision

- Large text
  - "output-examples" file[1]
- High color contrast
  - Both marks/text on background and labels on marks
  - Check with savonliquide package[2]

## 7.2 Color Vision Deficiency

### 7.2.1 Dual encoding (never just color)

- Line color – also vary line type
- Point color – also vary point shape
- https://www.youtube.com/watch?v=mbi_JVC1arM

### 7.2.2 Color palettes

- colorspace package[3]

[1] https://github.com/amzoss/RVis-2Day/blob/master/Day%201/templates/output-examples.md

[2] https://github.com/feddelegrand7/savonliquide

[3] http://colorspace.r-forge.r-project.org/index.html

## 7.3 Alternative Text for Screen Readers

In R, R Markdown:

- fig.alt[4] in code chunk (new, just for HTML output)
- fig.cap[5] in code chunk as backup
- embedded images: write alt text between square brackets
- New: ggplot2 v3.3.4 adds alt option in labs()[6], with plans to propagate to Rmd, Shiny

Writing good alt text for visualizations[7]

Longer descriptions: savonliquide package[8]

## 7.4 Converting graphics to sound, touch, text

- sonify package
- tactileR package
- BrailleR package[9]
  - Note: set plot title, subtitle, caption using labs()

Accessible Data Science for the Blind Using R[10]

## 7.5 Accessibility Resources

- savonliquide package[11]

[4] https://blog.rstudio.com/2021/04/20/knitr-fig-alt/
[5] https://bookdown.org/yihui/rmarkdown/r-code.html
[6] https://ggplot2.tidyverse.org/reference/labs.html
[7] https://nightingaledvs.com/writing-alt-text-for-data-visualization/
[8] https://github.com/feddelegrand7/savonliquide
[9] https://r-resources.massey.ac.nz/BrailleRInAction/GGPlot.html
[10] https://jooyoungseo.com/post/ds4blind/
[11] https://github.com/feddelegrand7/savonliquide

- Making betteR figures: Accessibility and Universal Design[12]
- Highlights from the DVS accessibility fireside chat[13]

[12] https://bookdown.org/ybrandvain/Applied-Biostats/betteRfigs.html#accessibility-and-universal-design

[13] https://nightingaledvs.com/highlights-from-the-dvs-accessibility-fireside-chat/

# 8

# *Exploring interactivity in visualizations with plotly and crosstalk*

*sample text*

We talk about the *FOO* method in this chapter.

# 9

# *Using RMarkdown to build websites for projects*

*sample text*

We talk about the *FOO* method in this chapter.

# 10

# *Using RMarkdown to build dashboards for projects*

*sample text*

We talk about the *FOO* method in this chapter.

# 11

# *Basic usability for interactive visualizations*

*sample text*

We talk about the *FOO* method in this chapter.

# 12

# *Teacher's guide*

*sample text*

We talk about the *FOO* method in this chapter.

# A

# *Datasets*

## Duke Enrollment

Duke enrollment[1]

Sample of Duke Enrollment By School dataset, Table A.1.

## A.1 Bar Chart

Figure A.1.

[1] https://doi.org/10.7924/r4db82p1j

**TABLE A.1:** A sample from the Duke Enrollment By School dataset.

| Year | Semester | Origin | Region | Sex | School | Count |
|---|---|---|---|---|---|---|
| 1970 | Fall | Alabama | United States | Female | Trinity | 11 |
| 1970 | Fall | Alabama | United States | Female | Graduate | 7 |
| 1970 | Fall | Alabama | United States | Female | Divinity | 1 |
| 1970 | Fall | Alabama | United States | Female | Law | 1 |
| 1970 | Fall | Alaska | United States | Female | Trinity | 1 |
| 1970 | Fall | Alaska | United States | Female | Graduate | 1 |

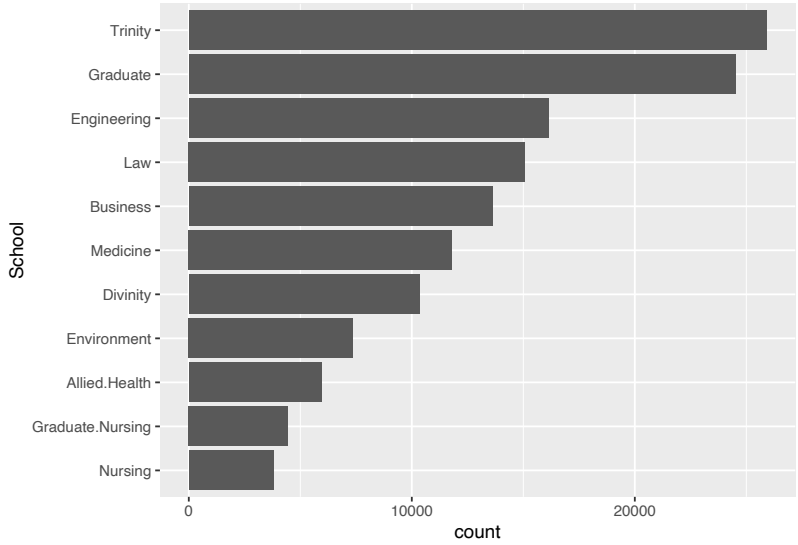



**FIGURE A.1:** Total Duke Enrollment by School

## Coral Resilience Data

Protecting coral reefs[2]

Figure A.2.

```
## Warning: Removed 1 rows containing missing values
## (geom_point).
```

## Git Experience

A Behavioral Approach to Understanding the Git Experience[3]

[2] https://doi.org/10.7924/G8348HFP
[3] https://osf.io/57tb8/

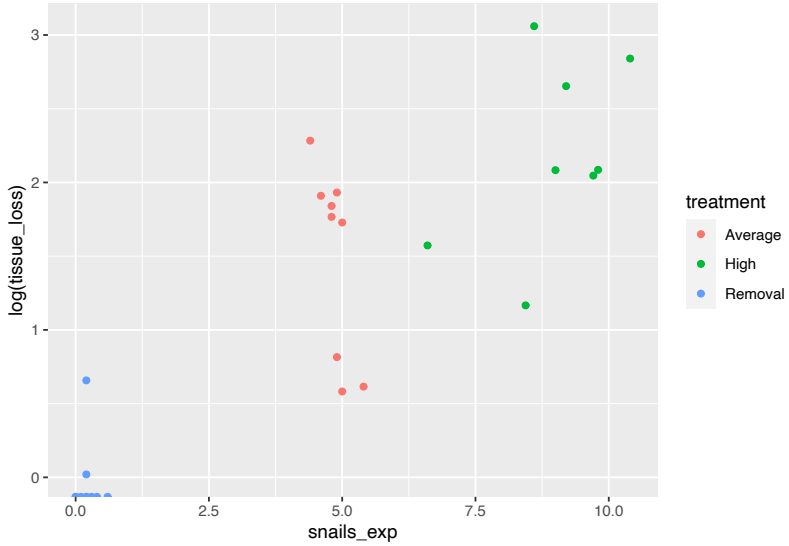
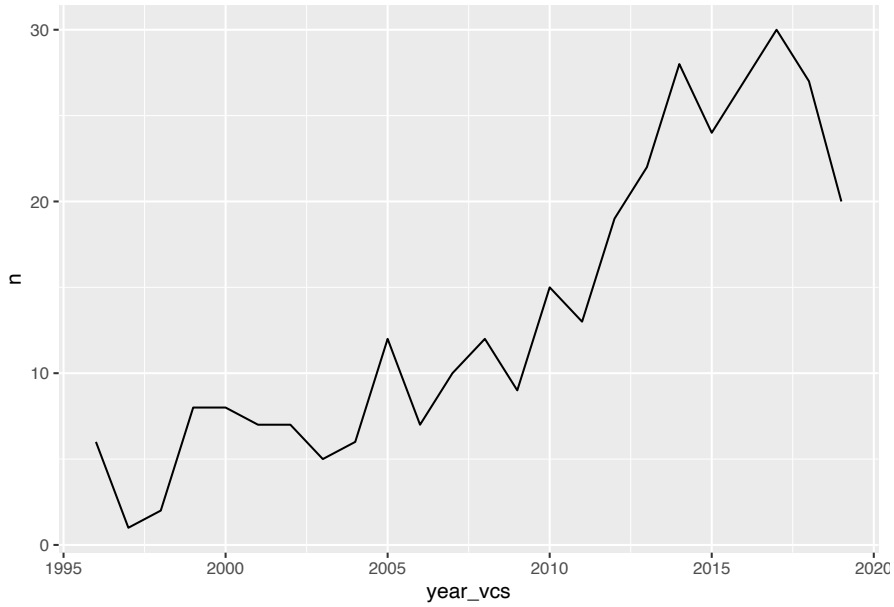



**FIGURE A.2:** Log of tissue loss by snail density

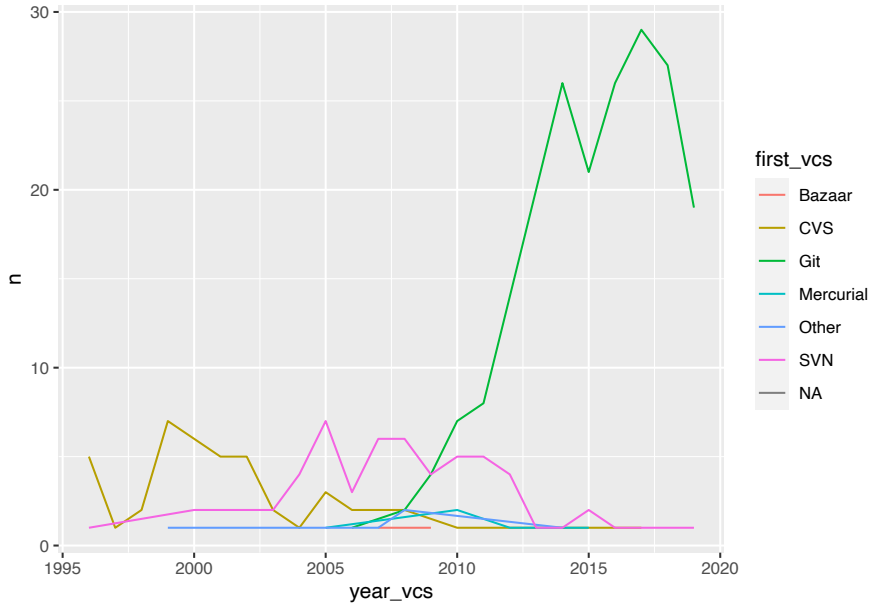
```
# won't work without aggregation
#ggplot(git_experience, aes(x=year_vcs)) +
#  geom_line()

ggplot(git_experience %>% count(year_vcs), aes(x=year_vcs, y=n)) +
  geom_line()
```

```
## Warning: Removed 1 row(s) containing missing values
## (geom_path).
```




```
ggplot(git_experience %>% count(year_vcs, first_vcs), aes(x=year_vcs, y=n, color=first_vcs)) +
  geom_line()
```

```
## Warning: Removed 3 row(s) containing missing values
## (geom_path).
```

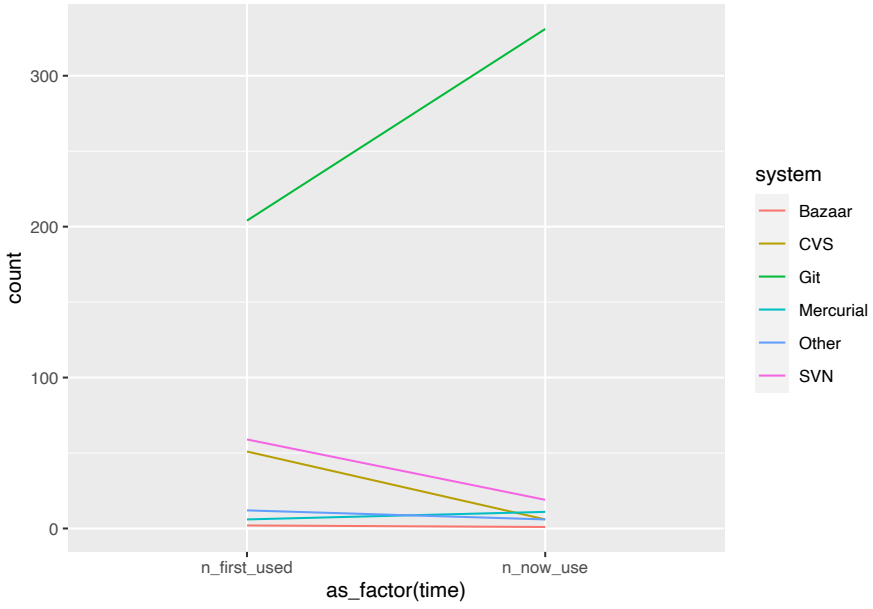



```
# First use vs. now use

first_used <- git_experience %>%
  count(first_vcs) %>%
  rename("n_first_used" = "n", "system" = "first_vcs")

now_use <- git_experience %>%
  pivot_longer(cols=c(first_bazaar,first_cvs,first_git,
                      first_hg,first_monotone,first_svn, first_other),
               names_to = "system") %>%
  dplyr::filter(value == TRUE) %>%
  count(system) %>%
  rename("n_now_use" = "n") %>%
  mutate(system = case_when(
    system == "first_bazaar" ~ "Bazaar",
    system == "first_cvs" ~ "CVS",
    system ==  "first_git" ~ "Git",
    system ==  "first_hg" ~ "Mercurial",
    system ==  "first_monotone" ~ "Monotone",
    system ==  "first_svn" ~ "SVN",
    system ==  "first_other" ~ "Other"
  ))
```

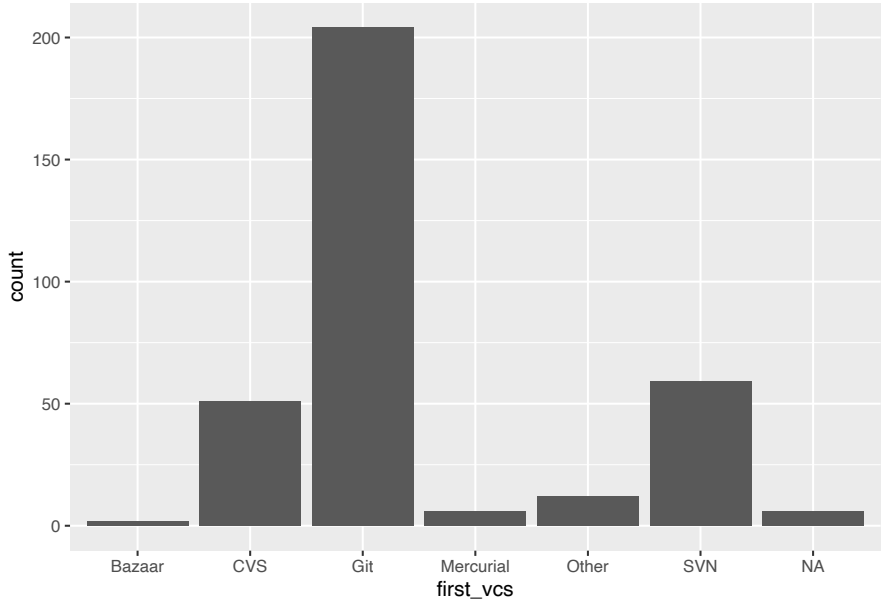
```
combined <- first_used %>% left_join(now_use) %>%
  pivot_longer(cols = c(n_first_used, n_now_use),
               names_to = "time",
               values_to = "count") %>%
  drop_na()
```

```
## Joining, by = "system"
```

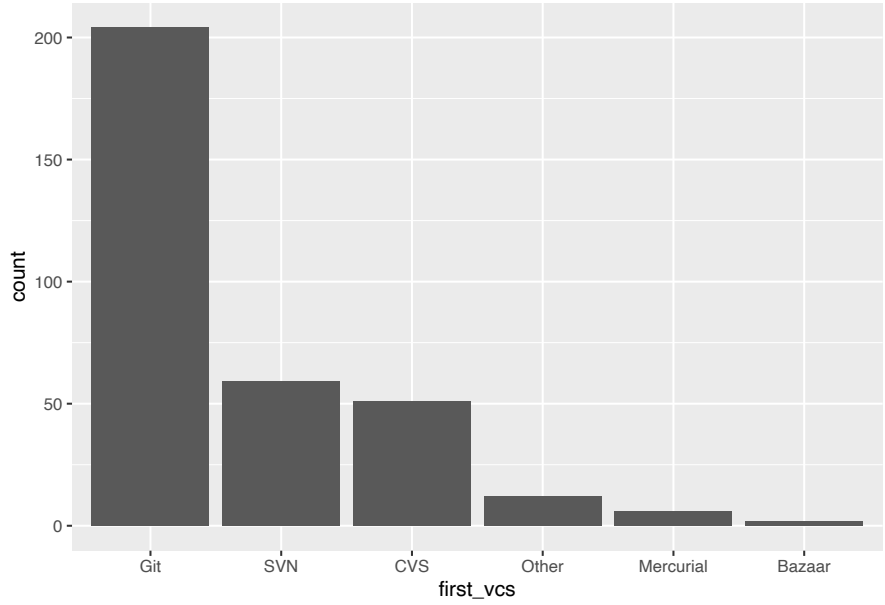
```
ggplot(combined, aes(x=as_factor(time), y=count, group=system, color=system)) +
  geom_line()
```



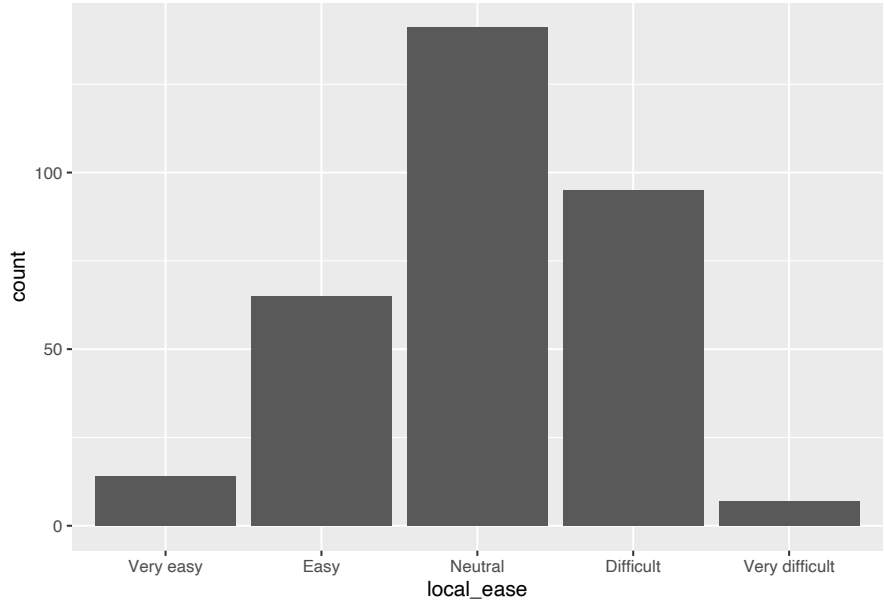

```
ggplot(git_experience, aes(x=first_vcs)) +
  geom_bar()
```



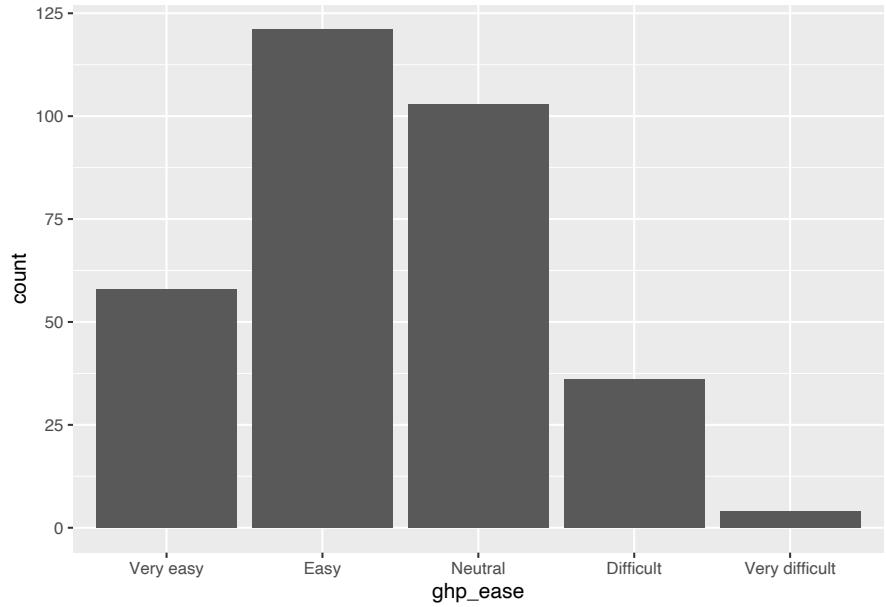

```
ggplot(git_experience %>%
         drop_na(first_vcs) %>%
         mutate(first_vcs = as_factor(first_vcs) %>%
             fct_infreq()),
         aes(x=first_vcs)) +
  geom_bar()
```



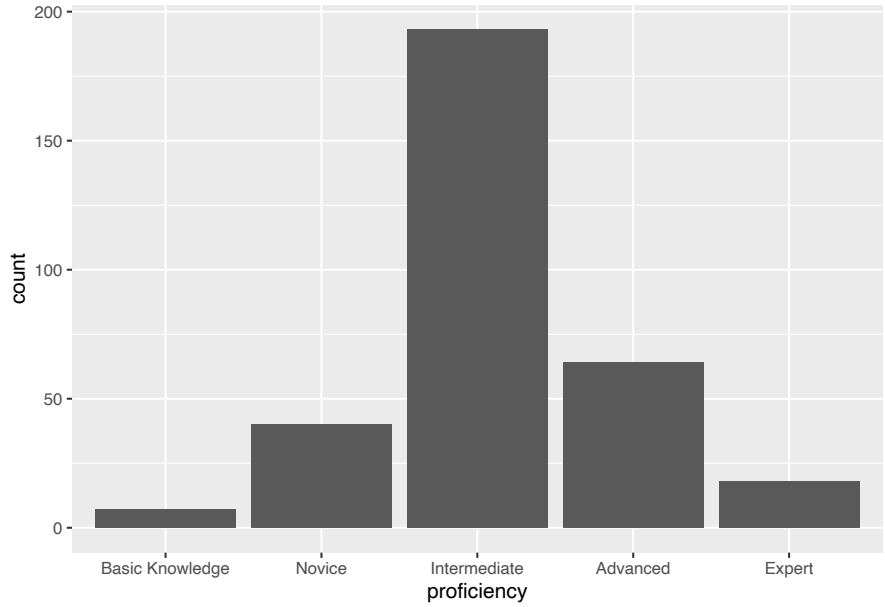

```
#How difficult was it for you to learn how to use git on your local computer?
# 1 - Very easy (1)
# 2 - Easy (2)
# 3 - Neutral (Neither easy nor difficult) (3)
# 4 - Difficult (4)
# 5 - Very difficult (5)

ggplot(git_experience %>%
         drop_na(local_ease) %>%
         mutate(local_ease = as_factor(local_ease) %>%
             fct_recode("Very easy"="1",
                        "Easy"="2", "Neutral"="3",
                        "Difficult"="4", "Very difficult"="5")),
       aes(local_ease)) +
  geom_bar()
```



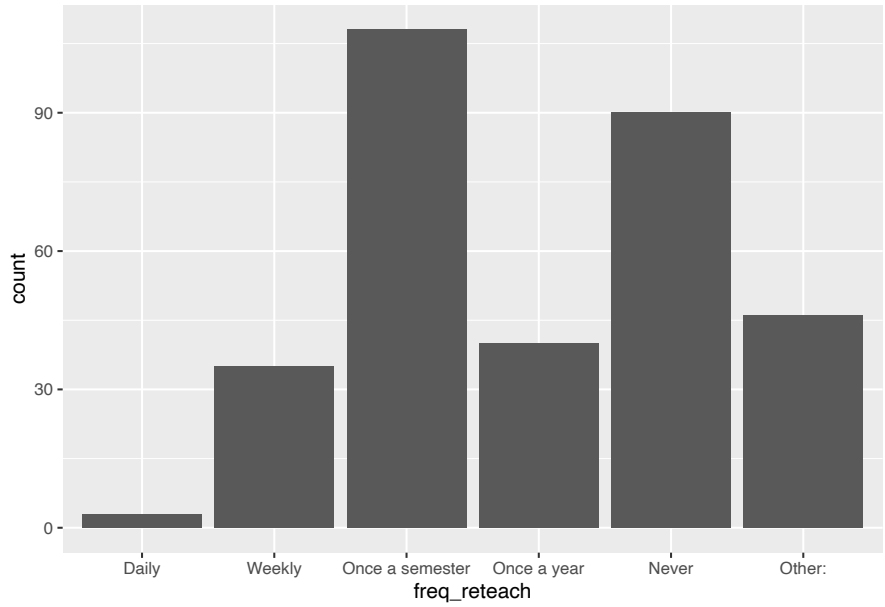

```
#How difficult was it for you to learn how to use the git hosting platform (e.g. GitLab, GitHub, e
# 1 - Very easy (1)
# 2 - Easy (2)
# 3 - Neutral (Neither easy nor difficult) (3)
# 4 - Difficult (4)
# 5 - Very difficult (5)
ggplot(git_experience %>%
         drop_na(ghp_ease) %>%
         mutate(ghp_ease = as_factor(ghp_ease) %>%
             fct_recode("Very easy"="1",
                        "Easy"="2", "Neutral"="3",
                        "Difficult"="4", "Very difficult"="5")),
       aes(ghp_ease)) +
  geom_bar()
```



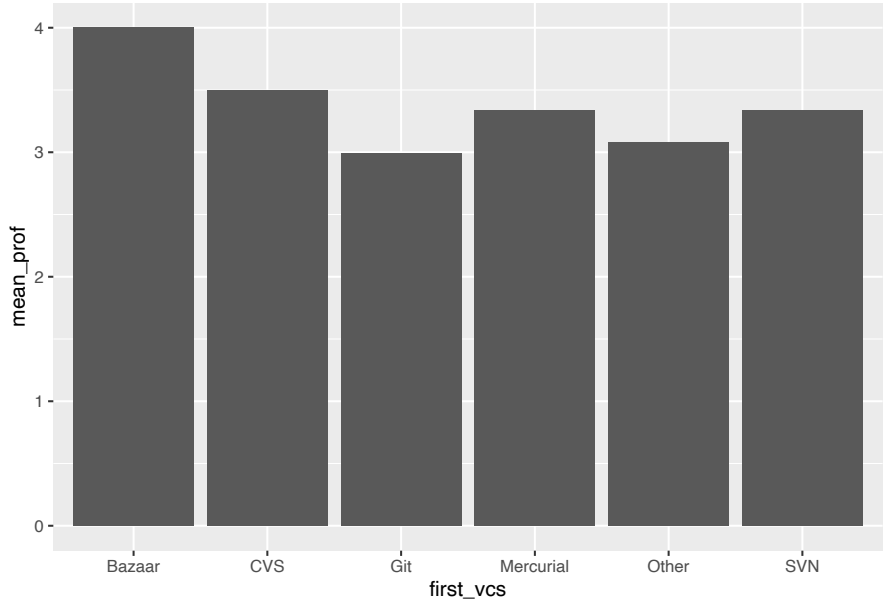

```
# How proficient do you think you are with git?
# 1 - Fundamental Awareness (basic knowledge) (1)
# 2 - Novice (limited experience) (2)
# 3 - Intermediate (practical application) (3)
# 4 - Advanced (applied theory) (4)
# 5 - Expert (recognized authority) (5)
ggplot(git_experience %>%
         drop_na(proficiency) %>%
         mutate(proficiency = as_factor(proficiency) %>%
             fct_recode("Basic Knowledge"="1",
                        "Novice"="2", "Intermediate"="3",
                        "Advanced"="4", "Expert"="5")),
       aes(proficiency)) +
  geom_bar()
```



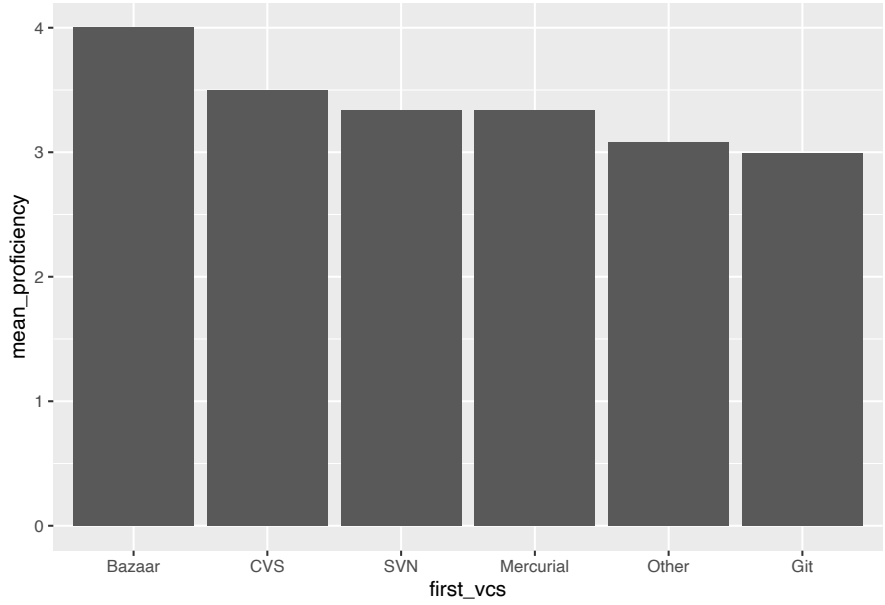

```
# How frequently do you have to reteach yourself git?
# Daily (1)
# Weekly (2)
# Once a semester (3)
# Once a year (4)
# Never (5)
# Other: (6)
ggplot(git_experience %>%
         drop_na(freq_reteach) %>%
         mutate(freq_reteach = as_factor(freq_reteach) %>%
             fct_relevel("Daily", "Weekly", "Once a semester",
                         "Once a year", "Never", "Other:")),
       aes(freq_reteach)) +
  geom_bar()
```




```
git_experience %>% drop_na(first_vcs) %>%
  group_by(first_vcs) %>%
  summarise(count = n(), mean_prof = mean(proficiency, na.rm=TRUE)) %>%
  ggplot(aes(x=first_vcs, y=mean_prof)) +
    geom_col()
```

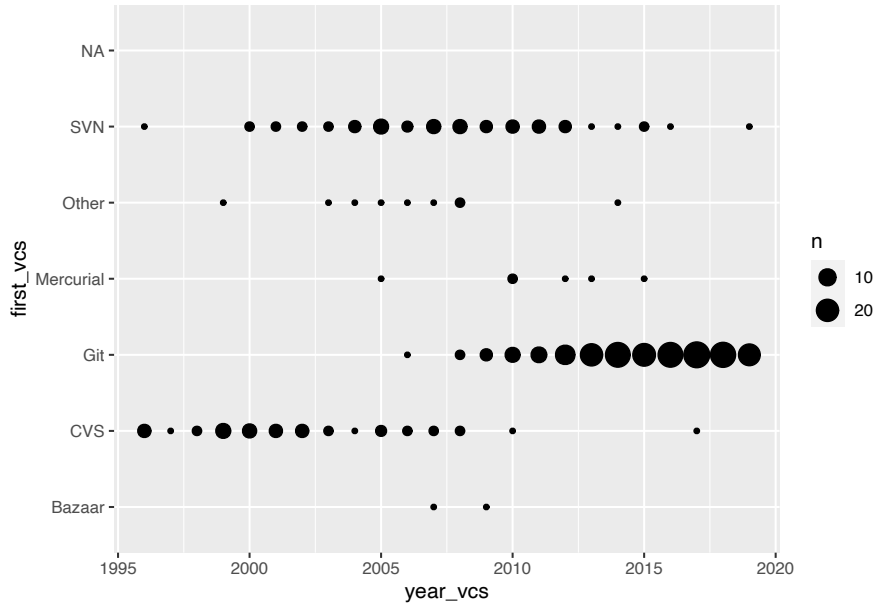



```
git_experience %>%
  drop_na(first_vcs) %>%
  group_by(first_vcs) %>%
  summarise(count = n(), mean_proficiency = mean(proficiency, na.rm=TRUE)) %>%
  mutate(first_vcs = as_factor(first_vcs) %>%
           fct_reorder(mean_proficiency) %>% fct_rev()) %>%
  ggplot(aes(x=first_vcs, y=mean_proficiency)) +
    geom_col()
```



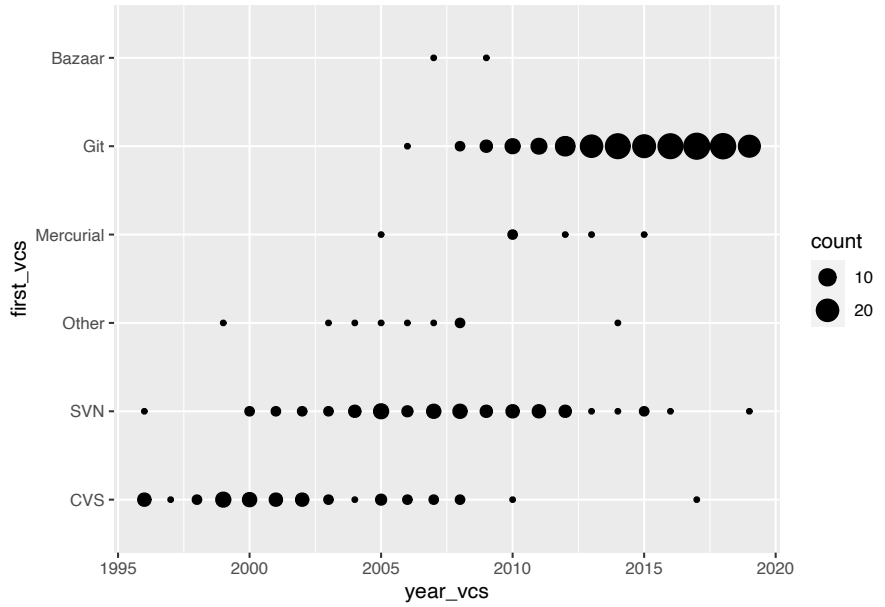

```
ggplot(git_experience, aes(x=year_vcs, y=first_vcs)) +
  geom_count()
```

```
## Warning: Removed 15 rows containing non-finite values
## (stat_sum).
```



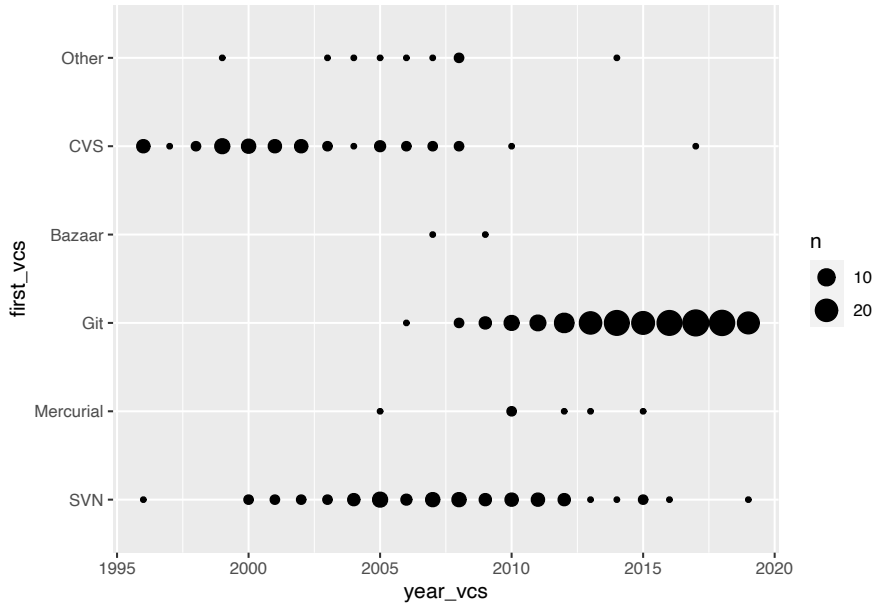

```
git_experience %>%
  drop_na(year_vcs, first_vcs) %>%
  group_by(year_vcs, first_vcs) %>%
  summarize(min_year = min(year_vcs),
            count=n()) %>%
  mutate(first_vcs = as_factor(first_vcs) %>%
           fct_reorder(min_year)) %>%
  ggplot(aes(x=year_vcs, y=first_vcs, size=count)) +
  geom_point()
```

```
## `summarise()` has grouped output by 'year_vcs'. You can override using the `.groups` argument.
```



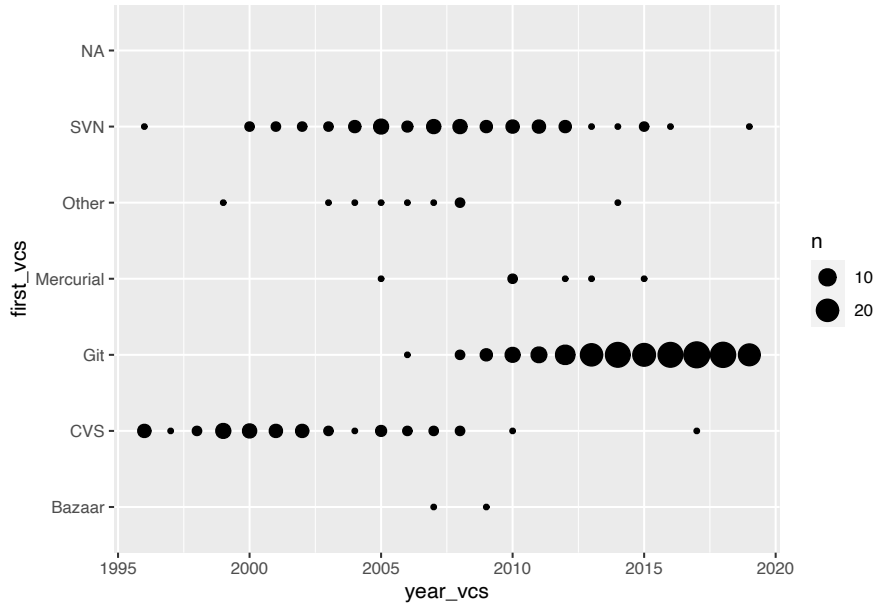

```
git_experience %>%
  drop_na(first_vcs) %>%
  mutate(first_vcs = as_factor(first_vcs) %>%
           fct_reorder(year_vcs, .fun=min)) %>%
  ggplot(aes(x=year_vcs, y=first_vcs)) +
  geom_count()
```

```
## Warning: Removed 9 rows containing non-finite values
## (stat_sum).
```



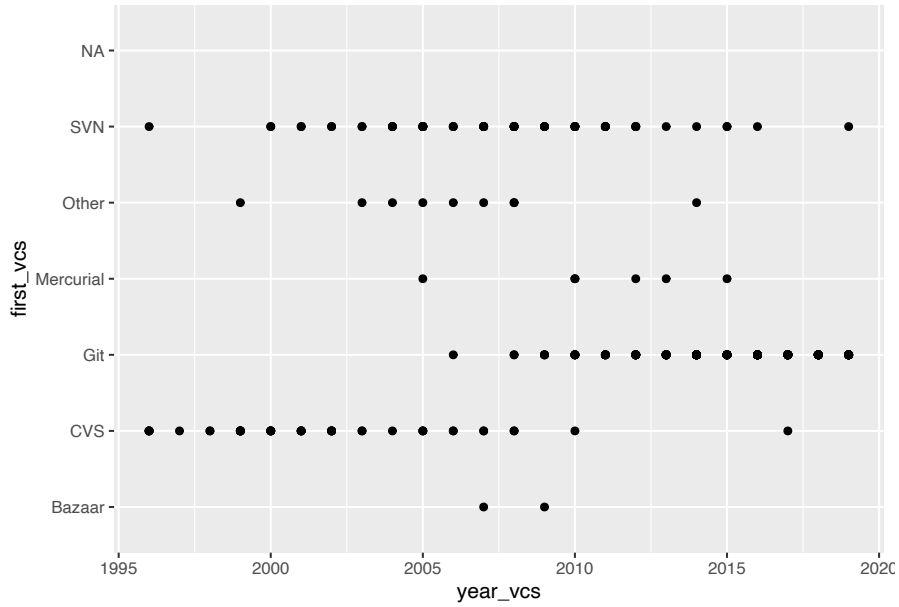

```
ggplot(git_experience %>% count(year_vcs, first_vcs),
       aes(x=year_vcs, y=first_vcs, size=n)) +
  geom_point()
```

```
## Warning: Removed 3 rows containing missing values
## (geom_point).
```



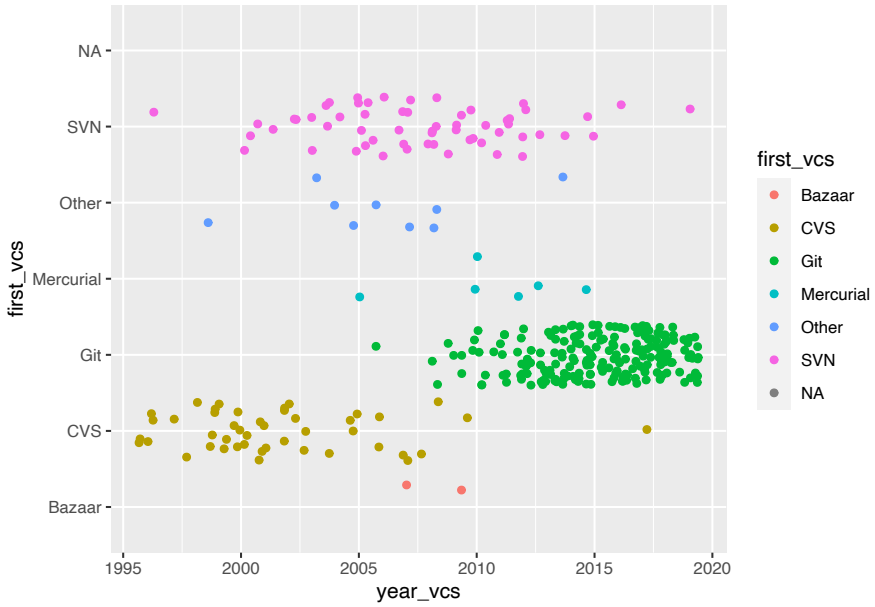

```
ggplot(git_experience, aes(x=year_vcs, y=first_vcs)) +
  geom_point()
```

```
## Warning: Removed 15 rows containing missing values
## (geom_point).
```



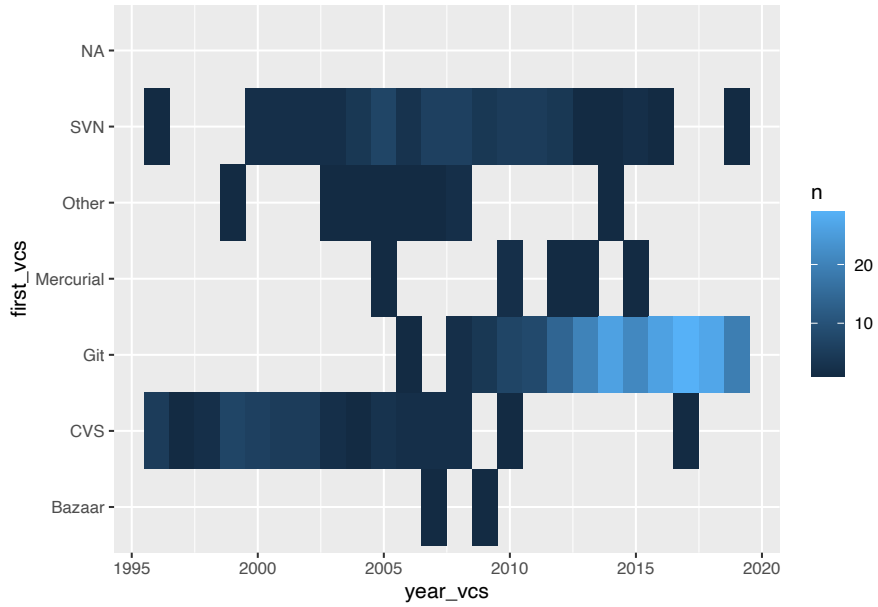

```
ggplot(git_experience, aes(x=year_vcs, y=first_vcs, color=first_vcs)) +
  geom_jitter()
```

```
## Warning: Removed 15 rows containing missing values
## (geom_point).
```

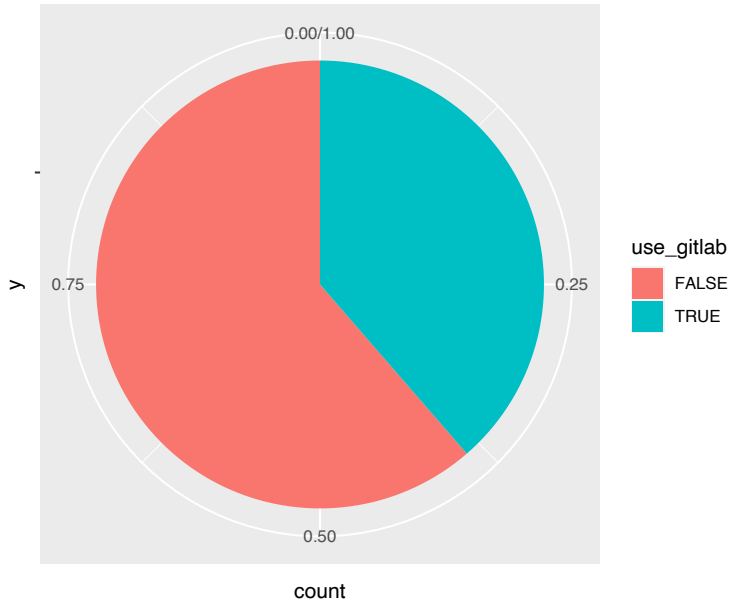



```
ggplot(git_experience %>% count(year_vcs, first_vcs),
       aes(x=year_vcs, y=first_vcs, fill=n)) +
  geom_tile()
```

```
## Warning: Removed 3 rows containing missing values
## (geom_tile).
```




```
ggplot(git_experience %>% drop_na(use_gitlab), aes(y="",fill=use_gitlab)) +
  geom_bar(position=position_fill()) +
  coord_polar()
```

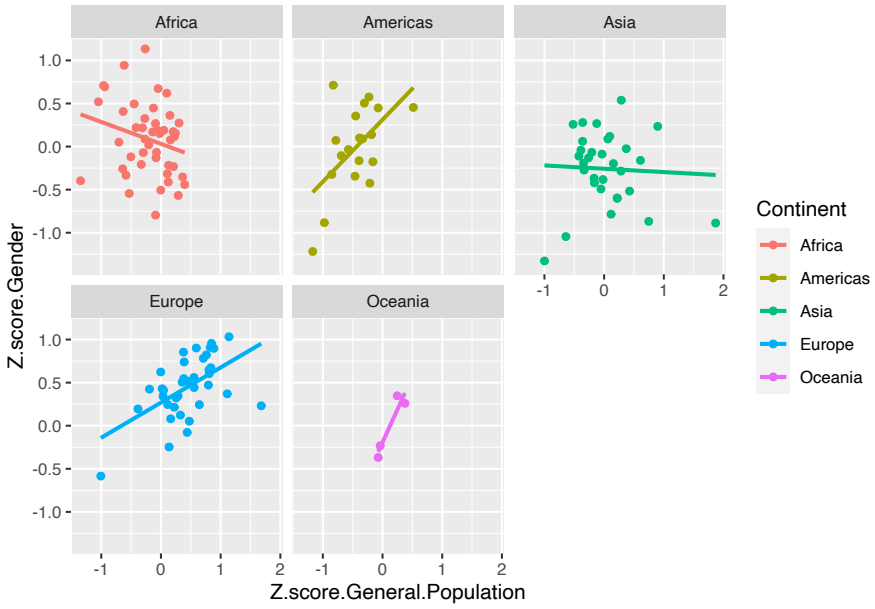


## Inclusiveness Index

Inclusiveness Index[4]

```
ggplot(inclusiveness_index,
       aes(x = Z.score.General.Population,
           y = Z.score.Gender,
           color = Continent)) +
  geom_point() +
  geom_smooth(method="lm", se=FALSE) +
  facet_wrap(vars(Continent))
```

```
## `geom_smooth()` using formula 'y ~ x'

## Warning: Removed 111 rows containing non-finite values
## (stat_smooth).

## Warning: Removed 111 rows containing missing values
## (geom_point).
```




[4] https://belonging.berkeley.edu/inclusivenessindex/data-and-resources

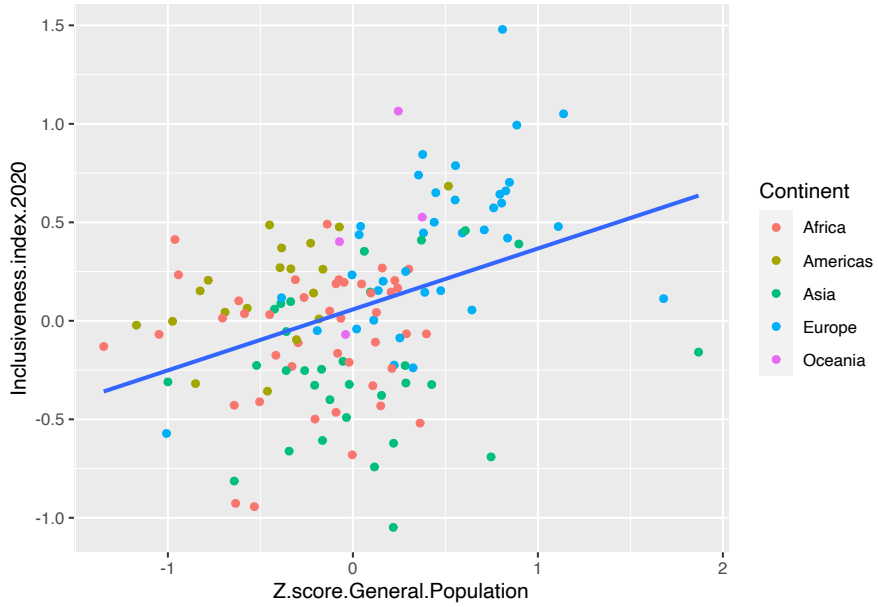
```
ggplot(inclusiveness_index,
       aes(x = Z.score.General.Population,
           y = Z.score.Sexual.Orientation)) +
  geom_point(aes(color = Continent)) +
  geom_smooth(method="lm", se=FALSE)
```

```
## `geom_smooth()` using formula 'y ~ x'

## Warning: Removed 109 rows containing non-finite values
## (stat_smooth).

## Warning: Removed 109 rows containing missing values
## (geom_point).
```


```
ggplot(inclusiveness_index,
       aes(x = Z.score.General.Population,
           y = Inclusiveness.index.2020)) +
  geom_point(aes(color = Continent)) +
  geom_smooth(method="lm", se=FALSE)
```

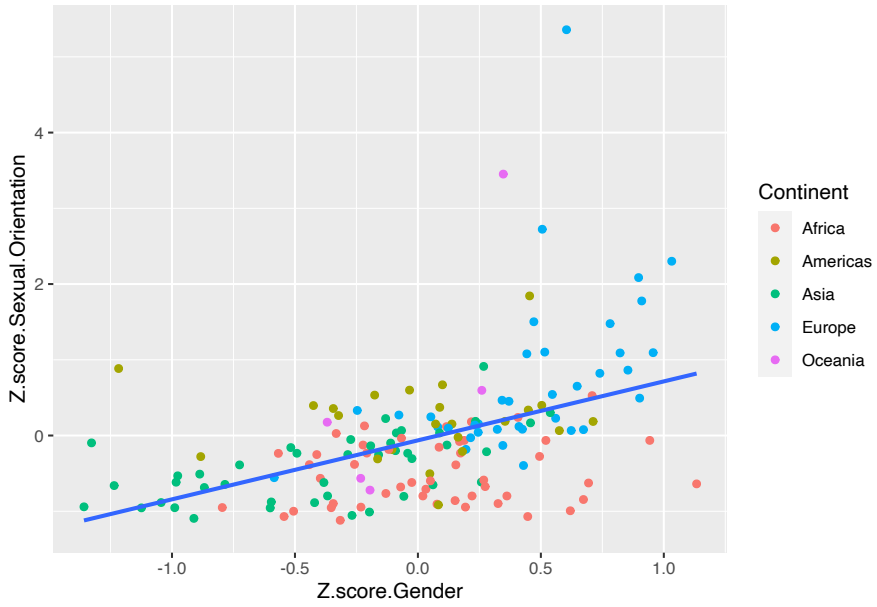
```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 113 rows containing non-finite values
## (stat_smooth).

## Warning: Removed 113 rows containing missing values
## (geom_point).
```



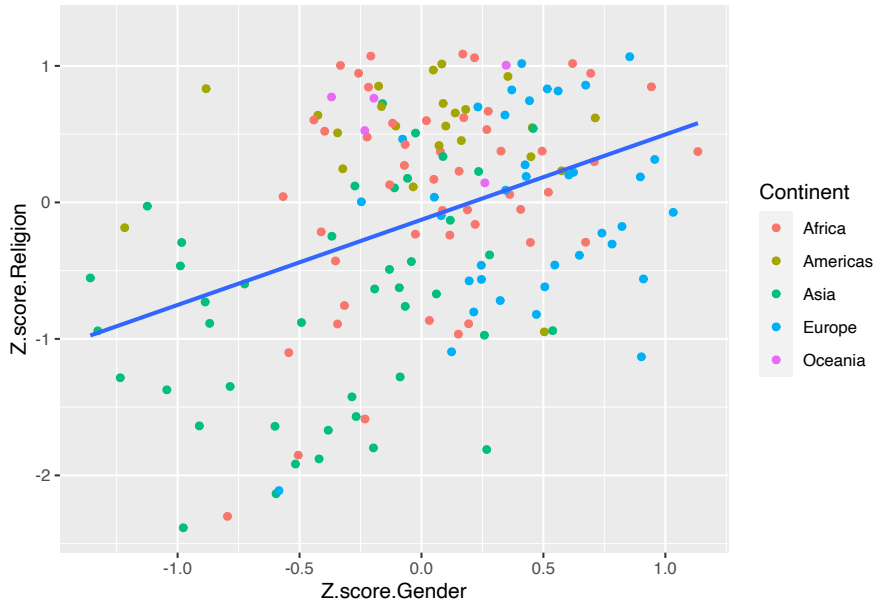

```
ggplot(inclusiveness_index,
       aes(x = Z.score.Gender,
           y = Z.score.Sexual.Orientation)) +
  geom_point(aes(color = Continent)) +
  geom_smooth(method="lm", se=FALSE)
```

```
## `geom_smooth()` using formula 'y ~ x'

## Warning: Removed 90 rows containing non-finite values
## (stat_smooth).

## Warning: Removed 90 rows containing missing values
## (geom_point).
```



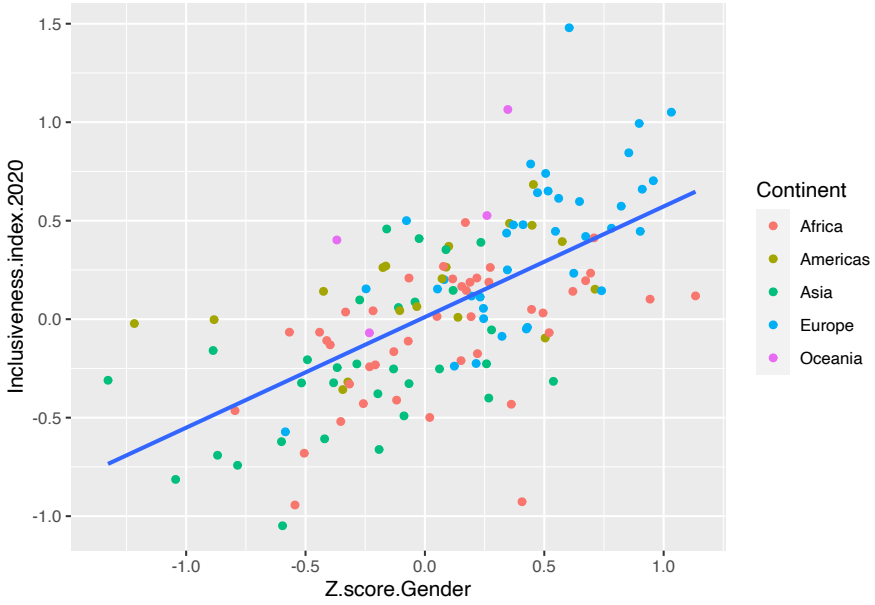

```
ggplot(inclusiveness_index,
       aes(x = Z.score.Gender,
           y = Z.score.Religion)) +
  geom_point(aes(color = Continent)) +
  geom_smooth(method="lm", se=FALSE)
```

```
## `geom_smooth()` using formula 'y ~ x'

## Warning: Removed 90 rows containing non-finite values
## (stat_smooth).

## Warning: Removed 90 rows containing missing values
## (geom_point).
```



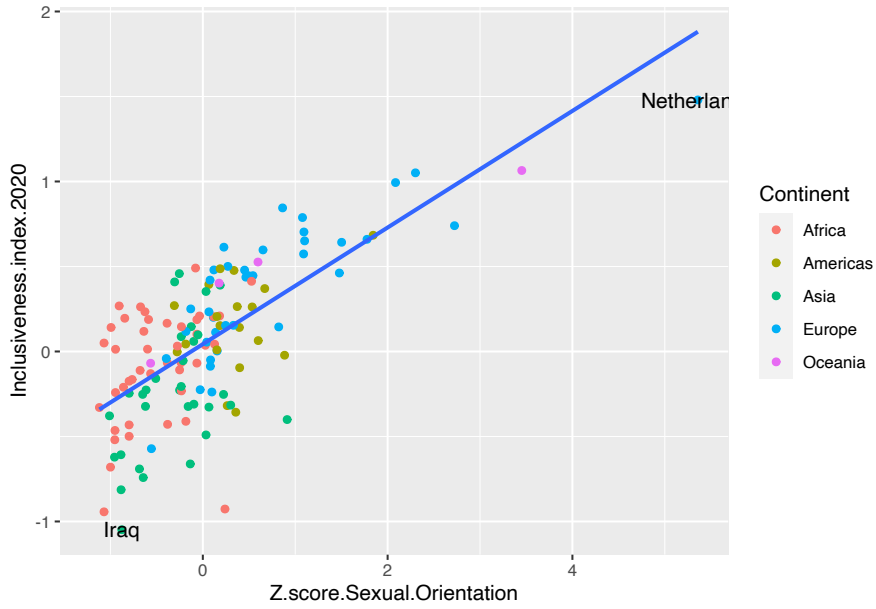

```
ggplot(inclusiveness_index,
       aes(x = Z.score.Gender,
           y = Inclusiveness.index.2020)) +
  geom_point(aes(color = Continent)) +
  geom_smooth(method="lm", se=FALSE)
```

```
## `geom_smooth()` using formula 'y ~ x'

## Warning: Removed 113 rows containing non-finite values
## (stat_smooth).

## Warning: Removed 113 rows containing missing values
## (geom_point).
```



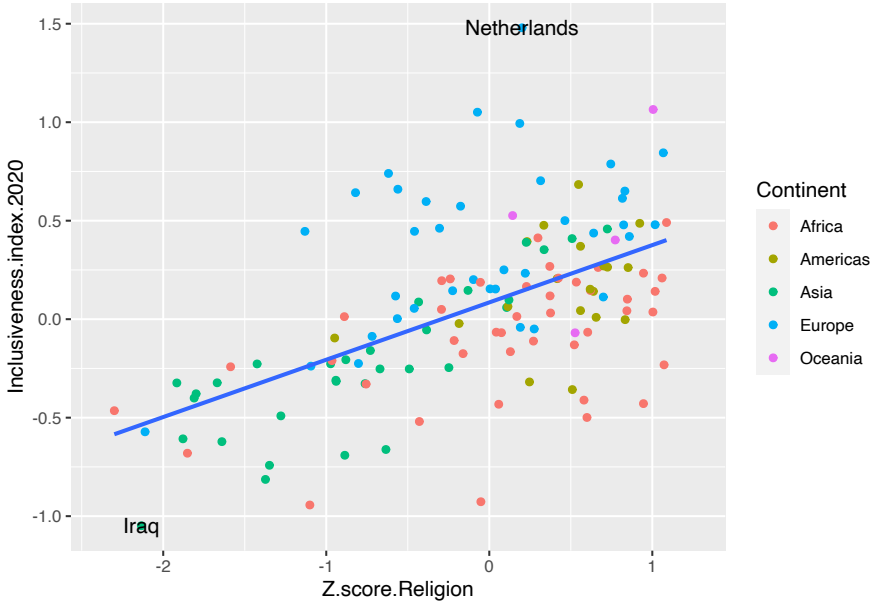

```
ggplot(inclusiveness_index,
       aes(x = Z.score.Sexual.Orientation,
           y = Inclusiveness.index.2020)) +
  geom_point(aes(color = Continent)) +
  geom_text(data = inclusiveness_index %>%
              dplyr::filter(Inclusiveness.index.2020 == max(Inclusiveness.index.2020, na.rm=TRUE)
  geom_smooth(method="lm", se=FALSE)
```

```
## `geom_smooth()` using formula 'y ~ x'

## Warning: Removed 113 rows containing non-finite values
## (stat_smooth).

## Warning: Removed 113 rows containing missing values
## (geom_point).
```



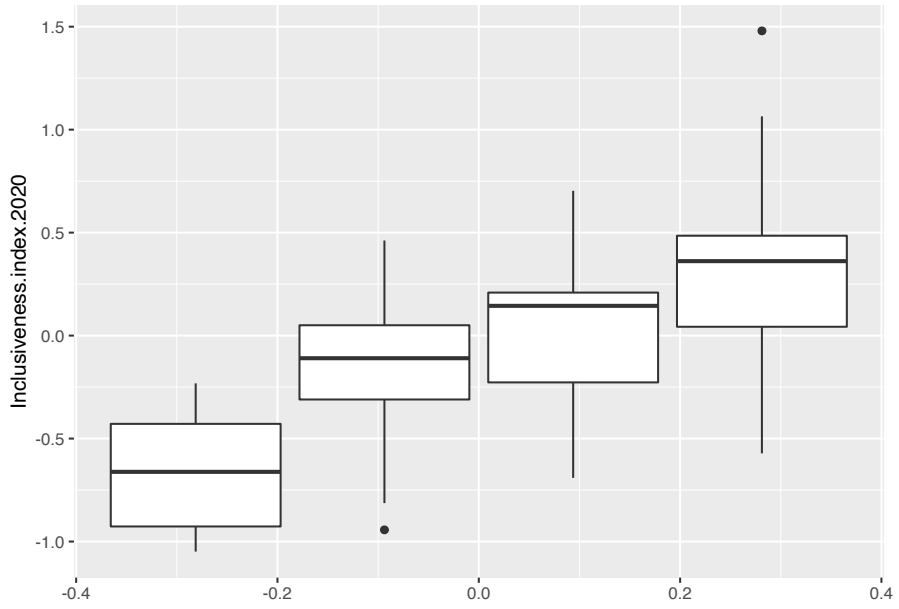

```
### USE THIS ONE? ###
ggplot(inclusiveness_index,
       aes(x = Z.score.Religion,
           y = Inclusiveness.index.2020)) +
  geom_point(aes(color = Continent)) +
  geom_text(data = inclusiveness_index %>%
              dplyr::filter(Inclusiveness.index.2020 == max(Inclusiveness.index.2020, na.rm=TRUE)
  geom_smooth(method="lm", se=FALSE)
```

```
## `geom_smooth()` using formula 'y ~ x'

## Warning: Removed 113 rows containing non-finite values
## (stat_smooth).

## Warning: Removed 113 rows containing missing values
## (geom_point).
```




```
ggplot(inclusiveness_index,
       aes(group = Z.score.Disability,
           y = Inclusiveness.index.2020)) +
  geom_boxplot()
```

```
## Warning: Removed 113 rows containing non-finite values
## (stat_boxplot).
```




## Candidate Demographics

Candidate Demographics[5]

Includes State, Candidate Name, Candidate Party, Office Name, White/Non-White, Race, Gender, Race/Gender Category, Office Level; 4 years (2012, 2014, 2016, 2018), over 40k records

## Affinity Spending

Affinity Spending[6]

[5] https://wholeads.us/research/rising-tide-ballot-demographics/
[6] https://github.com/OpportunityInsights/EconomicTracker

# *Bibliography*

Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.

Xie, Y. (2021). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.23.

# *Index*