Social Sciences Intro to Statistics

Week 2.2 Graphing

Week 2: Apply basic Dplyr functions in R and produce graphs of continuous and categorical variables.

Introduction

Lecture overview:

- Scatterplot, boxplot
- Graphing one continuous variable, one categorical variable, two continuous variable, two categorical variables
- Load and create data frame of generated variables
- Plot distribution, find mean, standard deviation, median

Load packages:

```
library(tidyverse)
library(ggplot2) # superfluous because ggplot2 is part of tidyverse
library(scales) # for formatting labels for axes and legends

library(haven)
library(labelled)
library(RColorBrewer)
```

Resources used to create this lecture:

- https://r4ds.had.co.nz/data-visualisation.html
- https://cfss.uchicago.edu/notes/grammar-of-graphics/#data-and-mapping
- https://codewords.recurse.com/issues/six/telling-stories-with-data-using-the-grammar-of-graphics
- http://r-statistics.co/Complete-Ggplot2-Tutorial-Part1-With-R-Code.html

- https://ggplot2-book.org/

ggplot cheatsheet: It may be helpful to familiarize yourself with this cheatsheet and use it as a reference point for review and practice after going through the lecture.

- https://www.maths.usyd.edu.au/u/UG/SM/STAT3022/r/current/Misc/data-visualization-2.1.pdf

Datasets we will use

We will use two datasets that are part of the `ggplot2` package:

- `mpg`: EPA fuel economy data in 1999 and 2008 for 38 car models that had a new release every year between 1999 and 2008
  - Note: There are no set of variables that uniquely identify observations
- `diamonds`: Prices and attributes of about 54,000 diamonds

```
#?mpg
glimpse(mpg)
#> Rows: 234
#> Columns: 11
#> $ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi", "~
#> $ model        <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro", "~
#> $ displ        <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, 2.~
#> $ year         <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, 200~
#> $ cyl          <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 8, 8, ~
#> $ trans        <chr> "auto(l5)", "manual(m5)", "manual(m6)", "auto(av)", "auto~
#> $ drv          <chr> "f", "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", "4~
#> $ cty          <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17, 1~
#> $ hwy          <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25, 2~
#> $ fl           <chr> "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p~
#> $ class        <chr> "compact", "compact", "compact", "compact", "compact", "c~
```

```
#?diamonds
glimpse(diamonds)
#> Rows: 53,940
#> Columns: 10
#> $ carat   <dbl> 0.23, 0.21, 0.23, 0.29, 0.31, 0.24, 0.24, 0.26, 0.22, 0.23, 0.~
#> $ cut     <ord> Ideal, Premium, Good, Premium, Good, Very Good, Very Good, Ver~
#> $ color   <ord> E, E, E, I, J, J, I, H, E, H, J, J, F, J, E, E, I, J, J, J, I,~
#> $ clarity <ord> SI2, SI1, VS1, VS2, SI2, VVS2, VVS1, SI1, VS2, VS1, SI1, VS1, ~
```

```
#> $ depth   <dbl> 61.5, 59.8, 56.9, 62.4, 63.3, 62.8, 62.3, 61.9, 65.1, 59.4, 64~
#> $ table   <dbl> 55, 61, 65, 58, 58, 57, 57, 55, 61, 61, 55, 56, 61, 54, 62, 58~
#> $ price   <int> 326, 326, 327, 334, 335, 336, 336, 337, 337, 338, 339, 340, 34~
#> $ x       <dbl> 3.95, 3.89, 4.05, 4.20, 4.34, 3.94, 3.95, 4.07, 3.87, 4.00, 4.~
#> $ y       <dbl> 3.98, 3.84, 4.07, 4.23, 4.35, 3.96, 3.98, 4.11, 3.78, 4.05, 4.~
#> $ z       <dbl> 2.43, 2.31, 2.31, 2.63, 2.75, 2.48, 2.47, 2.53, 2.49, 2.39, 2.~
```

We will use public-use data from the *National Center for Education Statistics (NCES) Educational Longitudinal Survey (ELS)* of 2002:

- Follows 10th graders from 2002 until 2012
- Variable `stu_id` uniquely identifies observations

The following RData file contains 2 dataframes, `df_els_stu_allobs` (variables are mostly haven labelled) and `df_els_stu_allobs_fac` (variables are mostly factors). We will be using the `df_els_stu_allobs_fac` because `ggplot` generally expects categorical variables to be factor variables.

```
load(file = url('https://github.com/anyone-can-cook/educ152/raw/main/data/els/output_data/els
els <- df_els_stu_allobs_fac
```

ELS dataset: Overview of variables using `glimpse()`

```
els %>% glimpse()
#> Rows: 16,197
#> Columns: 104
#> $ stu_id        <dbl> 101101, 101102, 101104, 101105, 101106, 101107, 101108, ~
#> $ sch_id        <fct> Suppressed, Suppressed, Suppressed, Suppressed, Suppress~
#> $ strat_id      <dbl> 101, 101, 101, 101, 101, 101, 101, 101, 101, 101, 101, 1~
#> $ psu           <fct> PSU 1, PSU 1, PSU 1, PSU 1, PSU 1, PSU 1, PSU 1, PSU 1, ~
#> $ f3univ        <chr> "1101", "1111", "1111", "1111", "1111", "1111", "1111", ~
#> $ g10cohrt      <fct> Sophomore cohort member, Sophomore cohort member, Sophom~
#> $ f1pared       <fct> "Attended college, no 4-year degree", "Attended college,~
#> $ byincome      <fct> "$50,001-$75,000", "$75,001-$100,000", "$50,001-$75,000"~
#> $ bystexp       <fct> "Attend or complete 2-year college/school", "Obtain PhD,~
#> $ byparasp      <fct> "Graduate from college", "Obtain PhD, MD, or other advan~
#> $ bytxstat      <fct> Both reading and math, Both reading and math, Both readi~
#> $ bypqstat      <fct> Full CATI, Hard copy full questionnaire, Full CATI, Nonr~
#> $ bytxmstd      <dbl+lbl> 52.11, 57.65, 66.44, 44.68, 40.57, 35.04, 50.71, 66.~
#> $ bynels2m      <dbl+lbl> 47.84, 55.30, 66.24, 35.33, 29.97, 24.28, 45.16, 66.~
#> $ bytxrstd      <dbl+lbl> 59.53, 56.70, 64.46, 48.69, 33.53, 28.85, 40.80, 68.~
#> $ bysctrl       <fct> Public, Public, Public, Public, Public, Public, Public, ~
```

```
#> $ byurban       <fct> Urban, Urban, Urban, Urban, Urban, Urban, Urban, Urban, ~
#> $ byregion      <fct> Northeast, Northeast, Northeast, Northeast, Northeast, N~
#> $ byfcomp       <fct> Father and female guardian, Mother and father, Mother an~
#> $ bysibhom      <fct> 0 siblings, Missing, 1 sibling, Nonrespondent, 3 sibling~
#> $ bys34a        <dbl+lbl>  1,  1, -9,  4,  8,  7,  1,  2, -9,  7,  4,  1,  5, ~
#> $ f1sex         <fct> Female, Female, Female, Female, Female, Male, Male, Male~
#> $ f1race        <fct> "Hispanic, race specified", "Asian, Hawaii/Pac. Islander~
#> $ f1stlang      <fct> Yes, No, Yes, Yes, No, No, Yes, Yes, Yes, Yes, Yes, No, ~
#> $ f1homlng      <fct> English, West/South Asian language, English, English, Sp~
#> $ f1mothed      <fct> "Did not finish high school", "Attended college, no 4-ye~
#> $ f1fathed      <fct> "Attended college, no 4-year degree", "Attended college,~
#> $ f1ses1        <fct> -0.25, 0.57, -0.86, -0.81, -1.41, -0.99, 0.27, -0.16, -1~
#> $ f1ses1qu      <fct> Second quartile, Highest quartile, Lowest quartile, Lowe~
#> $ f1stexp       <fct> "High school graduation only", "Obtain PhD, MD, or other~
#> $ f1txmstd      <dbl+lbl> 49.60, 60.64, 64.26, 45.59, 38.79, 32.00, 46.15, -8.~
#> $ f1rgpp2       <fct> 1.51 - 2.00, 2.51 - 3.00, 2.51 - 3.00, 2.51 - 3.00, 2.51~
#> $ f1s24cc       <fct> Item legitimate skip/NA, Item legitimate skip/NA, Item l~
#> $ f1s24bc       <fct> Item legitimate skip/NA, Item legitimate skip/NA, Item l~
#> $ f2everdo      <fct> No available evidence of dropout episode, No available e~
#> $ f2dostat      <fct> No known dropout episode, No known dropout episode, No k~
#> $ f2evratt      <fct> Survey component legitimate skip/NA, Yes, Yes, Yes, Yes,~
#> $ f2ps1         <fct> Survey component legitimate skip/NA, 1, 1, 1, 1, Item le~
#> $ f2ps1lvl      <fct> "Survey component legitimate skip/NA", "Four or more yea~
#> $ f2ps1ctr      <fct> Survey component legitimate skip/NA, Public, Public, Pub~
#> $ f2ps1sec      <fct> "Survey component legitimate skip/NA", "Public, 4-year o~
#> $ f2ps1slc      <fct> Suppressed, Suppressed, Suppressed, Suppressed, Suppress~
#> $ f2rtype       <fct> Survey component legitimate skip/NA, Standard enrollee, ~
#> $ f2c01         <fct> Survey component legitimate skip/NA, Yes, Yes, Yes, Yes,~
#> $ f2c24_p       <fct> Survey component legitimate skip/NA, 0 jobs, 1 job, Item~
#> $ f2c25a        <fct> Survey component legitimate skip/NA, Item legitimate ski~
#> $ f2c29_p       <fct> Survey component legitimate skip/NA, 0 jobs, 1 job, 1 jo~
#> $ f2c30a        <fct> Survey component legitimate skip/NA, Item legitimate ski~
#> $ f3hsstat      <fct> Recvd HS diploma: Fall 2003 - Summer 2004 graduate, Recv~
#> $ f3hscpdr      <dbl+lbl> 2004, 2004, 2004, 2004, 2004, 2004, 2004, 2004, 2004~
#> $ f3edstat      <fct> "Not currently enrolled, but has previous PS attendance"~
#> $ f3a01d        <fct> No, No, No, Yes, Yes, No, No, No, Nonrespondent, Yes, No~
#> $ f3evratt      <fct> Has some postsecondary enrollment, Has some postsecondar~
#> $ f3ps1start    <fct> 2005, 2004, 2004, 2005, 2004, 2006, 2004, 2004, Nonrespo~
#> $ f3ps1lvl      <fct> "At least 2, but less-than-4-year institution", "4-year ~
#> $ f3ps1ctr      <fct> Public, Public, Public, Public, Public, Private for-prof~
#> $ f3ps1sec      <fct> "2-year public", "4-year public", "4-year public", "2-ye~
#> $ f3ps1slc      <fct> Suppressed, Suppressed, Suppressed, Suppressed, Suppress~
```

```
#> $ f3ps1out      <fct> Yes, No, No, No, No, No, No, Yes, Nonrespondent, No, Yes~
#> $ f3ps1retain   <fct> No cred from PS1; no longer attending PS1; did attend an~
#> $ f3pstiming    <fct> Delayed postsecondary enrollment, Immediate postsecondar~
#> $ f3tztranresp  <fct> "Postsecondary attendance reported, transcript responden~
#> $ f3tzcoverage  <fct> "Potentially incomplete coverage: not all transcripts we~
#> $ f3tzrectrans  <dbl+lbl>  2,  2,  2,  3,  1, -4,  1,  2, -8,  1,  2,  2,  1, ~
#> $ f3tzreqtrans  <dbl+lbl>  3,  2,  2,  3,  2, -4,  1,  2, -8,  2,  2,  3,  1, ~
#> $ f3tzschtotal  <dbl+lbl>  3,  4,  2,  3,  2, -4,  1,  2, -8,  2,  3,  3,  1, ~
#> $ f3tzps1sec    <fct> "2-year public", "4-year public", "4-year public", "Less~
#> $ f3tzps1slc    <fct> "Selectivity not classified, 2-year institution", "Highl~
#> $ f3tzps1start  <dbl+lbl> 2005, 2004, 2004, 2008, 2004,   -4, 2004, 2004,   -8~
#> $ f3tzhs2ps1    <fct> 18, 3, 3, 54, 3, Nonrespondent, 3, 3, Survey component l~
#> $ f3tzever2yr   <fct> Yes, Yes, Yes, Yes, Yes, Nonrespondent, Yes, Yes, Survey~
#> $ f3tzever4yr   <fct> Yes, Yes, Yes, No, No, Nonrespondent, No, Yes, Survey co~
#> $ f3tzremtot    <dbl+lbl>  0,  0,  0,  3,  5, -4,  2,  0, -8,  4,  0,  0,  4, ~
#> $ f3tzrempass   <dbl+lbl>  0,  0,  0,  1,  5, -4,  2,  0, -8,  2,  0,  0,  4, ~
#> $ f3tzremengps  <dbl+lbl>  0,  0,  0,  0,  0, -4,  0,  0, -8,  0,  0,  0,  1, ~
#> $ f3tzrementot  <dbl+lbl>  0,  0,  0,  0,  0, -4,  0,  0, -8,  0,  0,  0,  1, ~
#> $ f3tzremmthps  <dbl+lbl>  0,  0,  0,  1,  3, -4,  1,  0, -8,  1,  0,  0,  2, ~
#> $ f3tzremmttot  <dbl+lbl>  0,  0,  0,  3,  3, -4,  1,  0, -8,  3,  0,  0,  2, ~
#> $ f3tzanydegre  <fct> No, No, Yes, Yes, No, Nonrespondent, No, Yes, Survey com~
#> $ f3tzhighdeg   <fct> Item legitimate skip/NA, Item legitimate skip/NA, Bachel~
#> $ f3tzcert1dt   <fct> Item legitimate skip/NA, Item legitimate skip/NA, Item l~
#> $ f3tzcrt1cip2  <fct> "Item legitimate skip/NA", "Item legitimate skip/NA", "I~
#> $ f3tzasoc1dt   <fct> Item legitimate skip/NA, Item legitimate skip/NA, Item l~
#> $ f3tzasc1cip2  <fct> "Item legitimate skip/NA", "Item legitimate skip/NA", "I~
#> $ f3tzbach1dt   <fct> Item legitimate skip/NA, Item legitimate skip/NA, 2008, ~
#> $ f3tzbch1cip2  <fct> "Item legitimate skip/NA", "Item legitimate skip/NA", "V~
#> $ f3stloanamt   <dbl+lbl>    -3,    -3, 40000,    -3, 20000,  5000,    -3, 250~
#> $ f3stloanevr   <fct> No, No, Yes, No, Yes, Yes, No, Yes, Nonrespondent, No, Y~
#> $ f3stloanpay   <dbl+lbl>  -3,  -3, 400,  -3, 260,  50,  -3, 260,  -4,  -3,  -~
#> $ f3ern2011     <dbl+lbl>  4000,  3000, 37000,  1500, 48000, 35000, 17000, 680~
#> $ f3tzpostatt   <dbl+lbl>  56,  51, 132,  90,  50,  -4,  30, 204,  -8,  15, 12~
#> $ f3tzpostern   <dbl+lbl>  44,  51, 132,  72,  50,  -4,  30, 195,  -8,  15, 10~
#> $ f1rmat_p      <fct> 4.0 - 4.99, 5.0 - 5.99, 5.0 - 5.99, 5.0 - 5.99, 5.0 - 5.~
#> $ f3totloan     <dbl> 0, 0, 40000, 0, 20000, 5000, 0, 25000, 0, 0, 45000, 2000~
#> $ f2enroll0405  <fct> NA, yes, yes, no, yes, no, yes, yes, NA, yes, yes, NA, N~
#> $ f2enroll0506  <fct> NA, yes, yes, yes, yes, no, yes, yes, NA, yes, yes, NA, ~
#> $ f2intern0405  <fct> NA, no, no, NA, no, NA, no, no, NA, no, no, NA, NA, no, ~
#> $ f2intern0506  <fct> NA, no, no, no, yes, NA, no, yes, NA, no, no, NA, NA, no~
#> $ parent_income <dbl> 62500, 87500, 62500, 500, 17500, 32500, 62500, 62500, 30~
#> $ f1race_v2     <fct> latinx, api, white, black, latinx, latinx, latinx, white~
```

```
#> $ hs_math_cred  <fct> 4.0 - 4.99, 5.0 - 5.99, 5.0 - 5.99, 5.0 - 5.99, 5.0 - 5.~
#> $ dev_math_01   <fct> no, no, no, yes, yes, NA, yes, no, NA, yes, no, no, yes,~
#> $ dev_math_cat4 <fct> 0 courses, 0 courses, 0 courses, 3+ courses, 3+ courses,~
#> $ dev_math_cat3 <fct> 0 courses, 0 courses, 0 courses, 2+ courses, 2+ courses,~
```

ELS dataset: Overview of variable labels using `var_label()`

```
els %>% var_label()
#> $stu_id
#> [1] "Student ID"
#>
#> $sch_id
#> [1] "School ID"
#>
#> $strat_id
#> [1] "Stratum"
#>
#> $psu
#> [1] "Primary sampling unit"
#>
#> $f3univ
#> [1] "Cross-round sample member status summary (BY to F3)"
#>
#> $g10cohrt
#> [1] "Sophomore cohort member in 2001-2002 school year"
#>
#> $f1pared
#> [1] "F1 parent's highest level of education"
#>
#> $byincome
#> [1] "Total family income from all sources 2001-composite"
#>
#> $bystexp
#> [1] "How far in school student thinks will get-composite"
#>
#> $byparasp
#> [1] "How far in school parent wants 10th-grader to go-composite"
#>
#> $bytxstat
#> [1] "Base year test score status"
#>
#> $bypqstat
```

```
#> [1] "Base year parent questionnaire status"
#>
#> $bytxmstd
#> [1] "Math test standardized score"
#>
#> $bynels2m
#> [1] "ELS-NELS 1992 scale equated sophomore math score"
#>
#> $bytxrstd
#> [1] "Reading test standardized score"
#>
#> $bysctrl
#> [1] "School control"
#>
#> $byurban
#> [1] "School urbanicity"
#>
#> $byregion
#> [1] "Geographic region of school"
#>
#> $byfcomp
#> [1] "Family composition"
#>
#> $bysibhom
#> [1] "BY number of in-home siblings"
#>
#> $bys34a
#> [1] "Hours/week spent on homework in school"
#>
#> $f1sex
#> [1] "F1 sex-composite"
#>
#> $f1race
#> [1] "F1 student's race/ethnicity-composite"
#>
#> $f1stlang
#> [1] "F1 whether English is student's native language-composite"
#>
#> $f1homlng
#> [1] "F1 student's native language-composite"
#>
#> $f1mothed
```

```
#> [1] "F1 mother's highest level of education-composite"
#>
#> $f1fathed
#> [1] "F1 father's highest level of education-composite"
#>
#> $f1ses1
#> [1] "F1 socio-economic status composite, v.1"
#>
#> $f1ses1qu
#> [1] "F1 quartile coding of SES1 variable"
#>
#> $f1stexp
#> [1] "F1 how far in school student thinks will get-composite"
#>
#> $f1txmstd
#> [1] "F1 math standardized score"
#>
#> $f1rgpp2
#> [1] "GPA for all courses taken in the 9th - 12th grades - categorical"
#>
#> $f1s24cc
#> [1] "Participated in Gear Up/other similar program in 11th grade"
#>
#> $f1s24bc
#> [1] "Participated in Upward Bound in 11th grade"
#>
#> $f2everdo
#> [1] "F2 ever dropped out"
#>
#> $f2dostat
#> [1] "F2 dropout status (as of 2006 interview)"
#>
#> $f2evratt
#> [1] "Whether has ever attended a postsecondary institution - composite"
#>
#> $f2ps1
#> [1] "First 'real' postsecondary institution link number"
#>
#> $f2ps1lvl
#> [1] "Level of offering of first postsecondary institution"
#>
#> $f2ps1ctr
```

```
#> [1] "Control of first postsecondary institution"
#>
#> $f2ps1sec
#> [1] "Sector of first postsecondary institution"
#>
#> $f2ps1slc
#> [1] "Institutional selectivity of first attended postsecondary institution"
#>
#> $f2rtype
#> [1] "F2 respondent type"
#>
#> $f2c01
#> [1] "Ever held a job since leaving high school"
#>
#> $f2c24_p
#> [1] "Number of jobs during 2004-2005 school year"
#>
#> $f2c25a
#> [1] "Held internship or co-op job while enrolled in 2004-2005 school year"
#>
#> $f2c29_p
#> [1] "Number of jobs during 2005-2006 school year"
#>
#> $f2c30a
#> [1] "Held internship or co-op job while enrolled in 2005-2006 school year"
#>
#> $f3hsstat
#> [1] "High school completion status (updated version of F2HSSTAT)"
#>
#> $f3hscpdr
#> [1] "High school completion date (updated version of F2HSCPDR)"
#>
#> $f3edstat
#> [1] "Postsecondary enrollment status as of the F3 interview"
#>
#> $f3a01d
#> [1] "Current activities:  Taking courses at a 2- or 4-yr school"
#>
#> $f3evratt
#> [1] "F3 ever attended a postsecondary institution"
#>
#> $f3ps1start
```

```
#> [1] "Year/month first attended a postsecondary institution"
#>
#> $f3ps1lvl
#> [1] "Level of first-attended PS institution"
#>
#> $f3ps1ctr
#> [1] "Control of first-attended PS institution"
#>
#> $f3ps1sec
#> [1] "Sector of first-attended PS institution"
#>
#> $f3ps1slc
#> [1] "Selectivity of first-attended PS institution"
#>
#> $f3ps1out
#> [1] "Whether 1st PS institution was out-of-state"
#>
#> $f3ps1retain
#> [1] "Status relative to first-attended postsecondary institution"
#>
#> $f3pstiming
#> [1] "Timing of first postsecondary enrollment"
#>
#> $f3tztranresp
#> [1] "Transcript: Transcript response status"
#>
#> $f3tzcoverage
#> [1] "Transcript: Overall transcript coverage indicator"
#>
#> $f3tzrectrans
#> [1] "Transcript: Number of transcripts received"
#>
#> $f3tzreqtrans
#> [1] "Transcript: Number of transcripts requested"
#>
#> $f3tzschtotal
#> [1] "Transcript: Total known institutions attended"
#>
#> $f3tzps1sec
#> [1] "Transcript: Sector of first known postsecondary institution"
#>
#> $f3tzps1slc
```

```
#> [1] "Transcript: Institutional selectivity of first known postsecondary institution"
#>
#> $f3tzps1start
#> [1] "Transcript: Date of first known postsecondary attendance"
#>
#> $f3tzhs2ps1
#> [1] "Transcript: Number of months between HS exit and postsecondary entry"
#>
#> $f3tzever2yr
#> [1] "Transcript: Ever attended a known 2-year institution"
#>
#> $f3tzever4yr
#> [1] "Transcript: Ever attended a known 4-year institution"
#>
#> $f3tzremtot
#> [1] "Transcript: Remedial courses: known number taken"
#>
#> $f3tzrempass
#> [1] "Transcript: Remedial courses: known number passed"
#>
#> $f3tzremengps
#> [1] "Transcript: Remedial English courses: known number passed"
#>
#> $f3tzrementot
#> [1] "Transcript: Remedial English courses: known number taken"
#>
#> $f3tzremmthps
#> [1] "Transcript: Remedial mathematics courses: known number passed"
#>
#> $f3tzremmttot
#> [1] "Transcript: Remedial mathematics courses: known number taken"
#>
#> $f3tzanydegre
#> [1] "Transcript: Any known degree attained as of June 2013"
#>
#> $f3tzhighdeg
#> [1] "Transcript: Highest known degree attained as of June 2013"
#>
#> $f3tzcert1dt
#> [1] "Transcript: Date of first known certificate earned"
#>
#> $f3tzcrt1cip2
```

```
#> [1] "Transcript: First known certificate major/field of study: 2-digit CIP"
#>
#> $f3tzasoc1dt
#> [1] "Transcript: Date of first known associate's degree earned"
#>
#> $f3tzasc1cip2
#> [1] "Transcript: First known associate's degree major/field of study: 2-digit CIP"
#>
#> $f3tzbach1dt
#> [1] "Transcript: Date of first known bachelor's degree earned"
#>
#> $f3tzbch1cip2
#> [1] "Transcript: First known bachelor's degree major/field of study: 2-digit CIP"
#>
#> $f3stloanamt
#> [1] "Total amount borrowed in student loans"
#>
#> $f3stloanevr
#> [1] "Whether R took out any student/PSE loans"
#>
#> $f3stloanpay
#> [1] "Amount currently paid monthly toward student loan balance"
#>
#> $f3ern2011
#> [1] "2011 employment income:  R only"
#>
#> $f3tzpostatt
#> [1] "Transcript: Postsecondary career: known credits attempted"
#>
#> $f3tzpostern
#> [1] "Transcript: Postsecondary career: known credits earned"
#>
#> $f1rmat_p
#> [1] "Units in mathematics (SST) - categorical"
#>
#> $f3totloan
#> [1] "total loans taken out to pay for postsecondary education as of f3 (2013)"
#>
#> $f2enroll0405
#> [1] "0/1 (no/yes) enrolled in 2004-05, based on student survey follow-up 2"
#>
#> $f2enroll0506
```

```
#> [1] "0/1 (no/yes) enrolled in 2005-06, based on student survey follow-up 2"
#>
#> $f2intern0405
#> [1] "0/1 (no/yes) held an internship or co-op in 2004-05; NA if not enrolled in postsecond
#>
#> $f2intern0506
#> [1] "0/1 (no/yes) held an internship or co-op in 2005-06; NA if not enrolled in postsecond
#>
#> $parent_income
#> [1] "continuous measure of base year parental household income, calculated from categorica
#>
#> $f1race_v2
#> [1] "categorical measure of race based on variable f1race"
#>
#> $hs_math_cred
#> [1] "Units in mathematics (SST) - categorical"
#>
#> $dev_math_01
#> [1] "dichotomous indicator of whether student took any developmental math courses in posts
#>
#> $dev_math_cat4
#> [1] "four category indicator of whether student took any developmental math courses in pos
#>
#> $dev_math_cat3
#> [1] "three category indicator of whether student took any developmental math courses in po
```

Type/class of variables ggplot expects

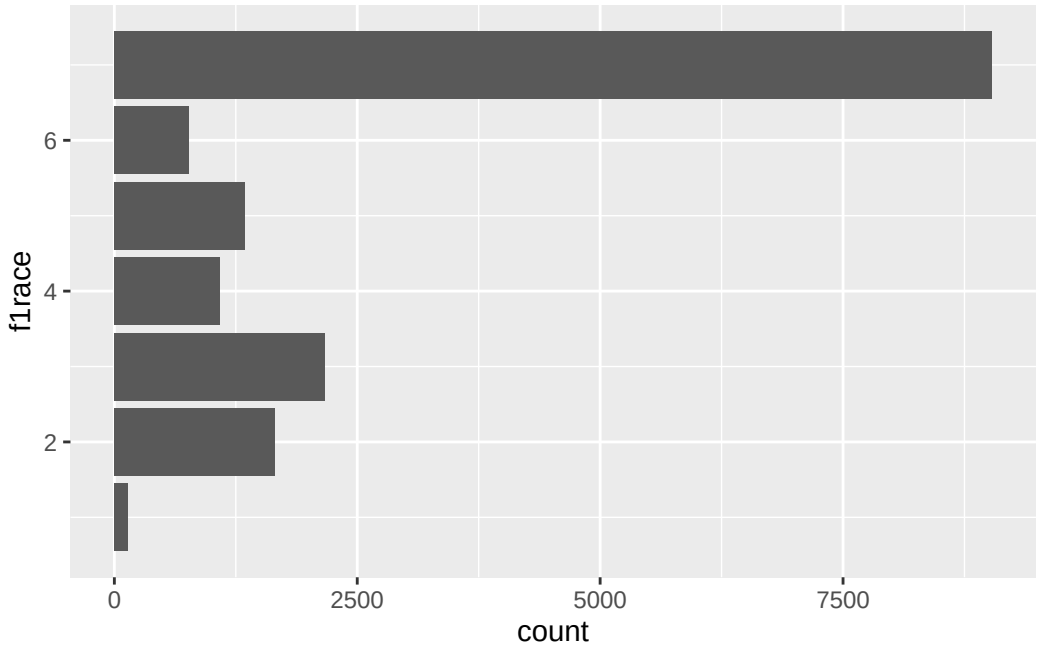
`ggplot` often doesn't work well with variables whose class is haven labelled. For categorical variables, `ggplot` works best with factor class because there is an ordering to the categorical values. We can convert variables to factor class as part of the data manipulation step prior to plotting.

Example: Comparing plotting haven labelled and factor variables

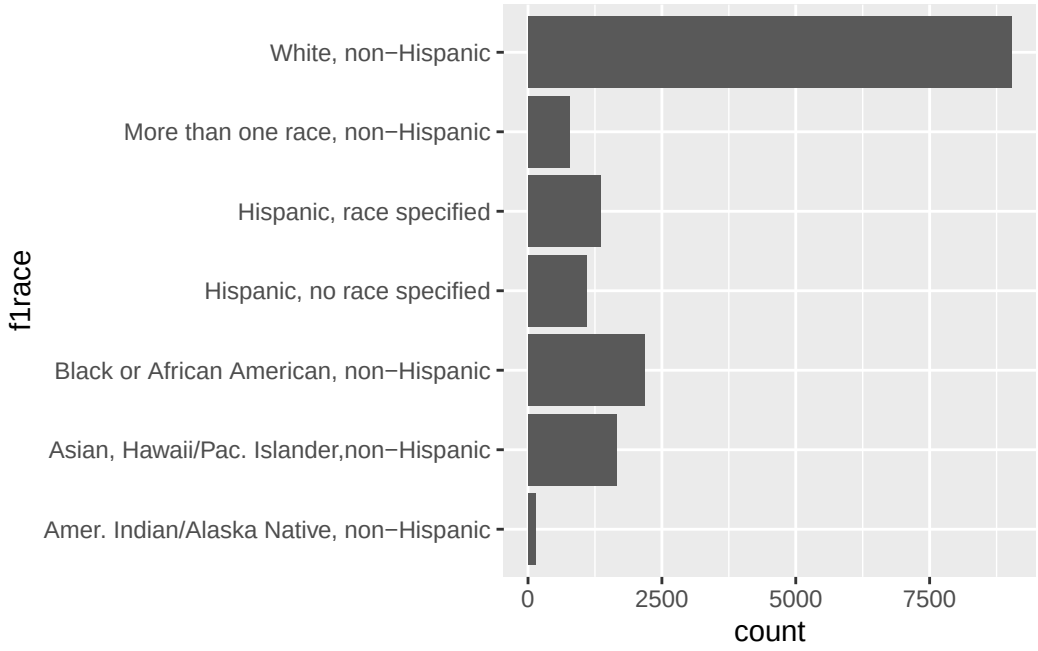
The `f1race` variable has a class of haven labelled in the `df_els_stu_allobs` dataframe and class of factor in the `df_els_stu_allobs_fac` dataframe. For the factor version, the levels should be the values we want the plot to display (e.g., `White, non-Hispanic`, `Black or African American, non-Hispanic`).

```
# Haven labelled
df_els_stu_allobs$f1race %>% str()
```

```
#>  dbl+lbl [1:16197] 5, 2, 7, 3, 4, 4, 4, 7, 4, 3, 3, 4, 3, 2, 2, 3, 3, 4, 7,...
#>  @ label       : chr "F1 student's race/ethnicity-composite"
#>  @ format.stata: chr "%8.0g"
#>  @ labels      : Named num [1:7] 1 2 3 4 5 6 7
#>   ..- attr(*, "names")= chr [1:7] "Amer. Indian/Alaska Native, non-Hispanic" "Asian, Hawa
df_els_stu_allobs %>% ggplot() + geom_bar(aes(y=f1race))
```




```
# Factor
df_els_stu_allobs_fac$f1race %>% str()
#>  Factor w/ 7 levels "Amer. Indian/Alaska Native, non-Hispanic",..: 5 2 7 3 4 4 4 7 4 3 ..
#>  - attr(*, "label")= chr "F1 student's race/ethnicity-composite"
df_els_stu_allobs_fac %>% ggplot() + geom_bar(aes(y=f1race))
```



If we try using `f1race` from `df_els_stu_allobs` (haven labelled) to specify the color of the points in the following scatterplot, we will get an error:

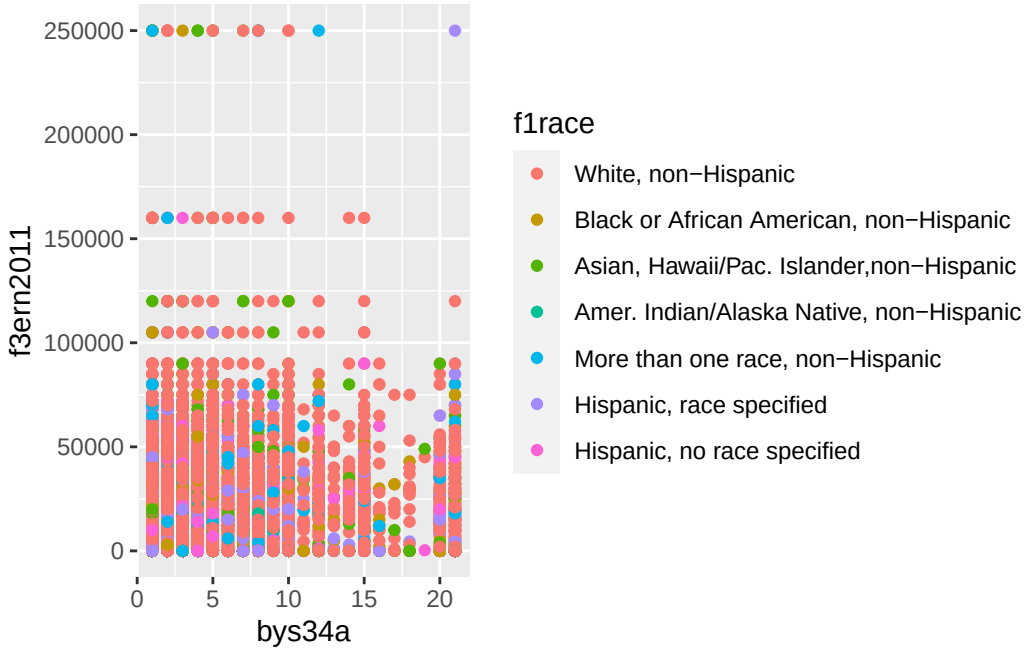
```
# Haven labelled (does not work)
df_els_stu_allobs %>%
  filter(unclass(bys34a) > 0) %>%
  ggplot(mapping = aes(x = bys34a, y = f3ern2011, color = f1race)) +
  geom_point()
```

But using `f1race` from `df_els_stu_allobs_fac` (factor) works:

```
# Factor
df_els_stu_allobs_fac %>%
  filter(unclass(bys34a) > 0) %>%
  ggplot(mapping = aes(x = bys34a, y = f3ern2011, color = f1race)) +
  geom_point()
```

We can also specify the *levels* of a factor variable to control the order in which the values are displayed:

```
df_els_stu_allobs_fac$f1race <- factor(
  df_els_stu_allobs_fac$f1race,
  levels = c('White, non-Hispanic',
             'Black or African American, non-Hispanic',
             'Asian, Hawaii/Pac. Islander,non-Hispanic',
             'Amer. Indian/Alaska Native, non-Hispanic',
             'More than one race, non-Hispanic',
             'Hispanic, race specified',
             'Hispanic, no race specified')
)

df_els_stu_allobs_fac %>%
  filter(unclass(bys34a) > 0) %>%
  ggplot(mapping = aes(x = bys34a, y = f3ern2011, color = f1race)) +
  geom_point()
```



Concepts

Basic definitions:

- Grammar
  - "The fundamental principles or rules of an art or science" (Oxford English dictionary)
- Grammar of graphics [@RN4563]
  - Principles/rules to describe and construct statistical graphics
- Layered grammar of graphics [@RN4561]
  - Principles/rules to describe and construct statistical graphics "based around the idea of building up a graphic from multiple layers of data" [@RN4561, p. 4]
  - The layered grammar of graphics is a "formal system for building plots… based on the insight that you can uniquely describe *any* plot as a combination of" seven paramaters [@RN4564, chapter 3]
- Aesthetics
  - **Aesthetics** are visual elements of the plot (e.g., lines, points, symbols, colors, axes)

  - **Aesthetic mappings** (**mappings**) are visual elements of the plot determined by values of specific variables (e.g., a scatterplot where the color of each point depends on the value of the variable `race`)
  - In the context of visualization using ggplot, all **mappings** are **aesthetic mappings** but not all **aesthetics** are **aesthetic mappings** (i.e., determined by variable values). For example, when creating a scatterplot you may specify that the color of each point be blue.

The seven parameters of the layered grammar of graphics consists of:

- Five layers
  - A dataset (**data**)
  - A set of aesthetic mappings (**mappings**)
  - A statistical transformation (**stat**)
  - A geometric object (**geom**)
  - A position adjustment (**position**)
- A coordinate system (**coord**)
- A faceting scheme (**facets**)

`ggplot2` – part of `tidyverse` – is an R package to create graphics and `ggplot()` is a function within the `ggplot2` package.

> "In practice, you rarely need to supply all seven parameters to make a graph because ggplot2 will provide useful defaults for everything except the data, the mappings, and the geom function." [@RN4564, chapter 3]

Syntax conveying the seven parameters of the layered grammar of graphics:

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

Layers

What does Wickham mean by **layers**? (from "Telling Stories with Data Using the Grammar of Graphics" by Liz Sander)

- In the grammar of a language, words have different parts of speach (e.g., noun, verb, adjective), with each part of speech performing a different role in a sentence
- The layered grammar of graphics decomposes a graphic into different **layers**
  - "These are layers in a literal sense – you can think of them as transparency sheets for an overhead projector, each containing a piece of the graphic, which can be arranged and combined in a variety of ways."

The five layers of the grammar of graphics:

- Dataset (**data**)
- Set of aesthetic mappings (**mappings**)
- Statistical transformation (**stat**)
- Geometric object (**geom**)
- Position adjustment (**position**)

Dataset (data)

Data defines the information to be visualized.

Example: Imagine a dataset where each observation is a student

- The variables of interest are hours per week spent on homework in high school (`bys34a`), earnings in 2011 (`f3ern2011`), and student sex (`f1sex`)

```
els %>% select(stu_id, bys34a, f3ern2011, f1sex) %>% head(10)
#> # A tibble: 10 x 4
#>    stu_id bys34a f3ern2011 f1sex
#>     <dbl>  <dbl>     <dbl> <fct>
#>  1 101101      1      4000 Female
#>  2 101102      1      3000 Female
#>  3 101104     -9     37000 Female
#>  4 101105      4      1500 Female
#>  5 101106      8     48000 Female
#>  6 101107      7     35000 Male
#>  7 101108      1     17000 Male
#>  8 101109      2     68000 Male
#>  9 101110     -9        -4 Male
#> 10 101111      7     42000 Male
```

Set of aesthetic mappings (mappings)

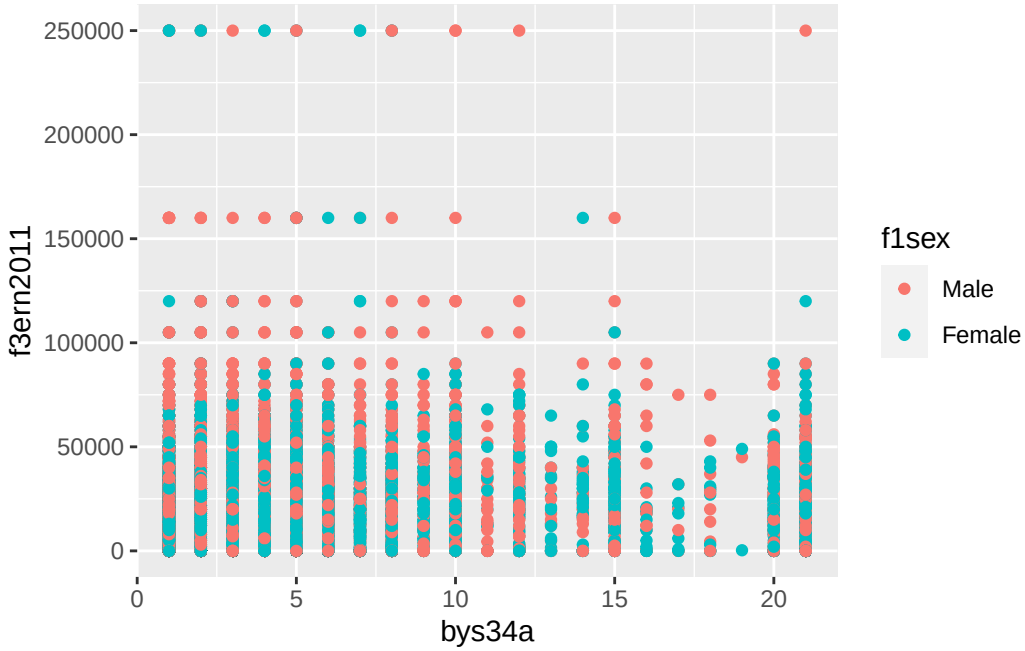
Mapping defines how variables in a dataset are applied (mapped) to a graphic.

Example: Consider the previous dataset

- Map hours/week spent on homework to the x-axis
- Map 2011 income to the y-axis
- Map sex to the color of each point

```
els %>% select(stu_id, bys34a, f3ern2011, f1sex) %>%
  rename(x = bys34a, y = f3ern2011, color = f1sex) %>%
  head(10)
#> # A tibble: 10 x 4
#>    stu_id     x     y color
#>     <dbl> <dbl> <dbl> <fct>
#>  1 101101     1  4000 Female
#>  2 101102     1  3000 Female
#>  3 101104    -9 37000 Female
#>  4 101105     4  1500 Female
#>  5 101106     8 48000 Female
#>  6 101107     7 35000 Male
#>  7 101108     1 17000 Male
#>  8 101109     2 68000 Male
#>  9 101110    -9    -4 Male
#> 10 101111     7 42000 Male
```

```
els %>%
  filter(bys34a > 0) %>%
  ggplot(mapping = aes(x = bys34a, y = f3ern2011, color = f1sex)) +
  geom_point()
```

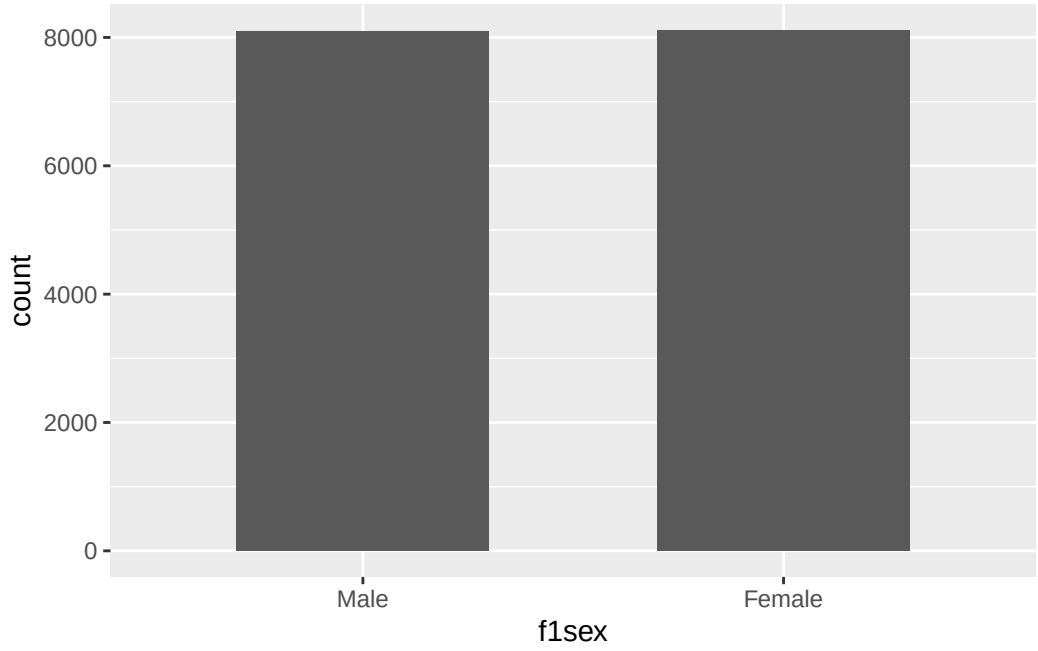


Example: Depending on the type of plot, a different set of mapping may be needed (i.e., not always `x` to one variable and `y` to another)

- Map sex to the x-axis (*by default, bar plots will plot the counts, see next section*)

```
els %>% select(stu_id, f1sex) %>%
  rename(x = f1sex) %>%
  head(10)
#> # A tibble: 10 x 2
#>    stu_id x
#>     <dbl> <fct>
#>  1 101101 Female
#>  2 101102 Female
#>  3 101104 Female
#>  4 101105 Female
#>  5 101106 Female
#>  6 101107 Male
#>  7 101108 Male
#>  8 101109 Male
#>  9 101110 Male
#> 10 101111 Male
```

```r
els %>%
  ggplot(mapping = aes(x = f1sex)) +
  geom_bar(width = 0.6)
```



Statistical transformation (stat)

A **statistical transformation** transforms the underlying data before plotting it. Different types of plots (i.e., `geom_*()`) will use a different transformation by default so we often do not need to explicitly specify it.

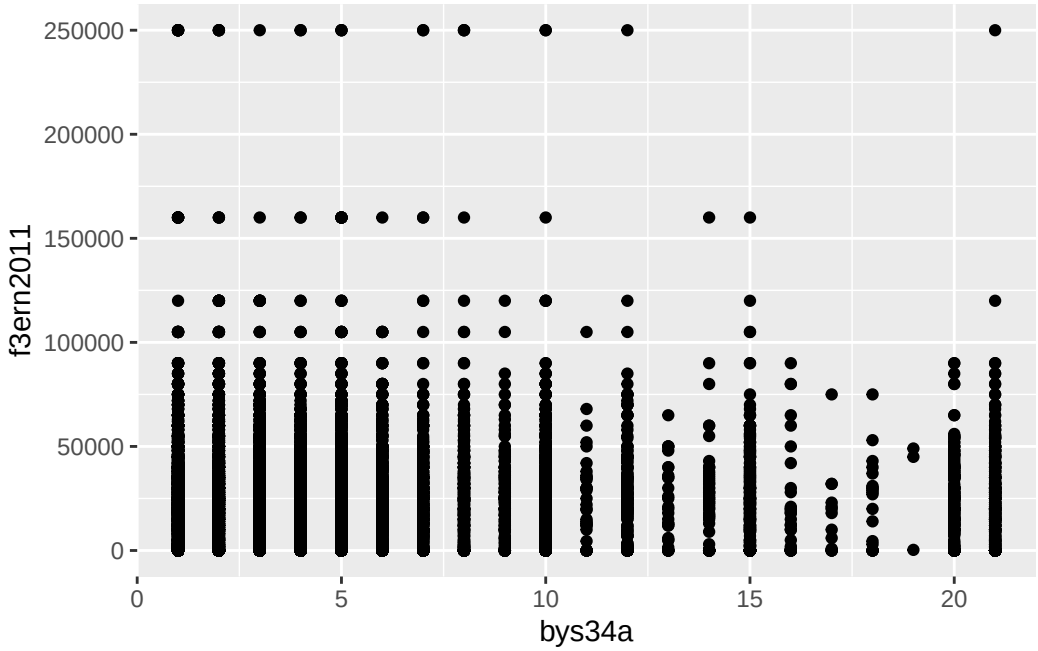
Example: Imagine creating a scatterplot of the relationship between hours/week spent on homework (x-axis) and 2011 income (y-axis)

- When creating a scatterplot we usually do not transform the data prior to plotting
- This is the "identity" transformation (*default for plots like scatterplots*)

```r
els %>% select(stu_id,bys34a,f3ern2011) %>% rename(x=bys34a, y=f3ern2011) %>%
  head(10)
#> # A tibble: 10 x 3
#>    stu_id     x     y
#>     <dbl> <dbl> <dbl>
#>  1 101101     1  4000
```

```
#>  2 101102     1  3000
#>  3 101104    -9 37000
#>  4 101105     4  1500
#>  5 101106     8 48000
#>  6 101107     7 35000
#>  7 101108     1 17000
#>  8 101109     2 68000
#>  9 101110    -9    -4
#> 10 101111     7 42000
```

```
els %>%
  filter(bys34a > 0) %>%
  ggplot(mapping = aes(x = bys34a, y = f3ern2011)) +
  geom_point()
```

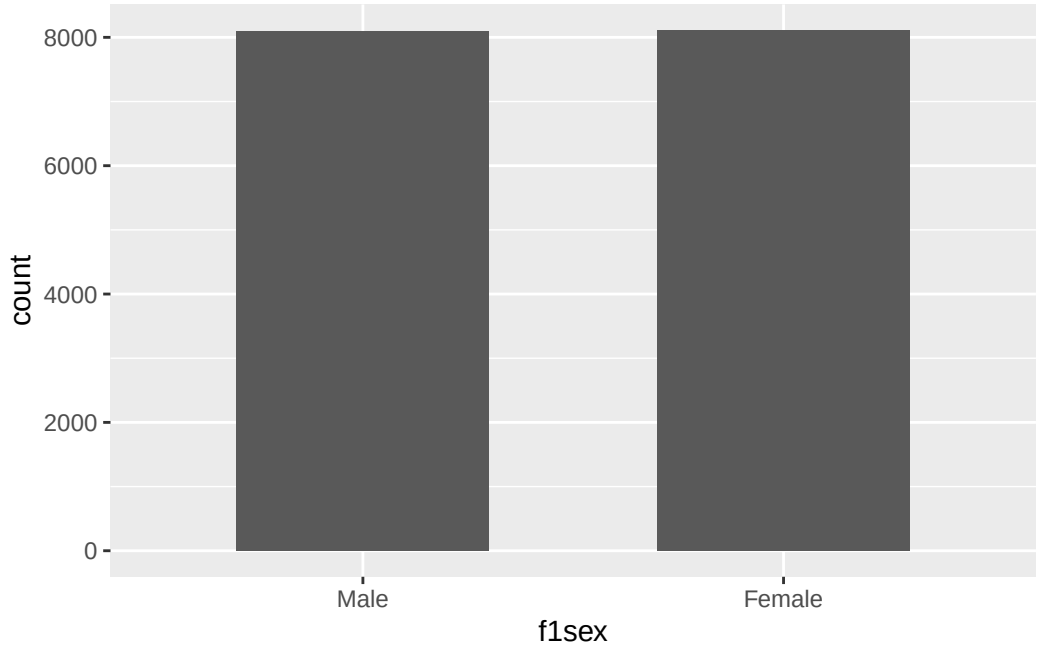


Example: Imagine creating a bar chart of the number of students by sex

- Here, we do not plot the raw data. Rather, we count the number of observations for each sex category.
- This is the "count" transformation (*default for plots like barplots*)

```r
els %>% count(f1sex)
#> # A tibble: 2 x 2
#>   f1sex      n
#>   <fct>  <int>
#> 1 Male    8090
#> 2 Female  8107
```

```r
els %>%
  ggplot(mapping = aes(x = f1sex)) +
  geom_bar(width = 0.6)
```



Geometric objects (geoms)

Graphs visually display data, using **geometric objects** like a point, line, bar, etc.

- Each geometric object in a graph is called a "geom" – they are defined as "visual marks that represent data points" (ggplot cheatsheet)
- "A geom is the geometrical object that a plot uses to represent data" [@RN4564, chapter 3]
- "People often describe plots by the type of geom that the plot uses. For example, bar charts use bar geoms, line charts use line geoms, boxplots use boxplot geoms" [@RN4564, chapter 3]

- **Aesthetics** are "visual attributes of the geom" (e.g., color, fill, shape, position) (Grammar of Graphics)
  - Each geom can only display certain aesthetics
  - For example, a "point geom" can only include the aesthetics position, color, shape, and size
- We can plot the same underlying data using different geoms (e.g., bar chart vs. pie chart)
- A single graph can layer multiple geoms (e.g., scatterplot with a "line of best fit" layered on top)

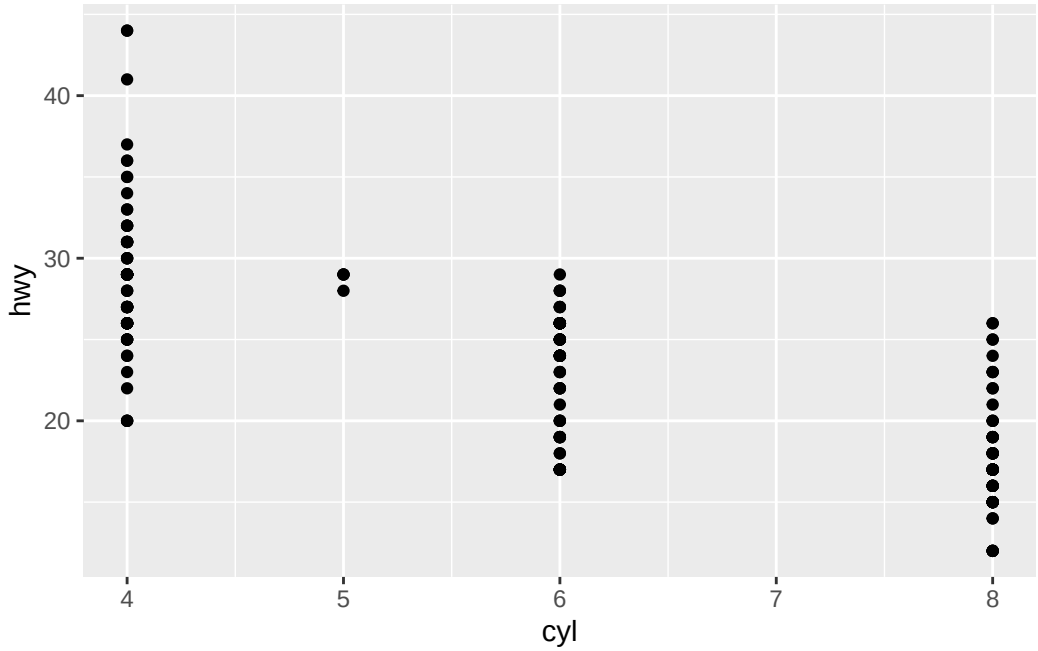
Position adjustment (position)

Position adjustment adjusts the position of visual elements in the plot so that these visual elements do not overlap with one another in ways that make the plot difficult to interpret.

Example: The dataset `mpg` (included in the `ggplot2` package) contains variables for the specifications of different cars, with 234 observations

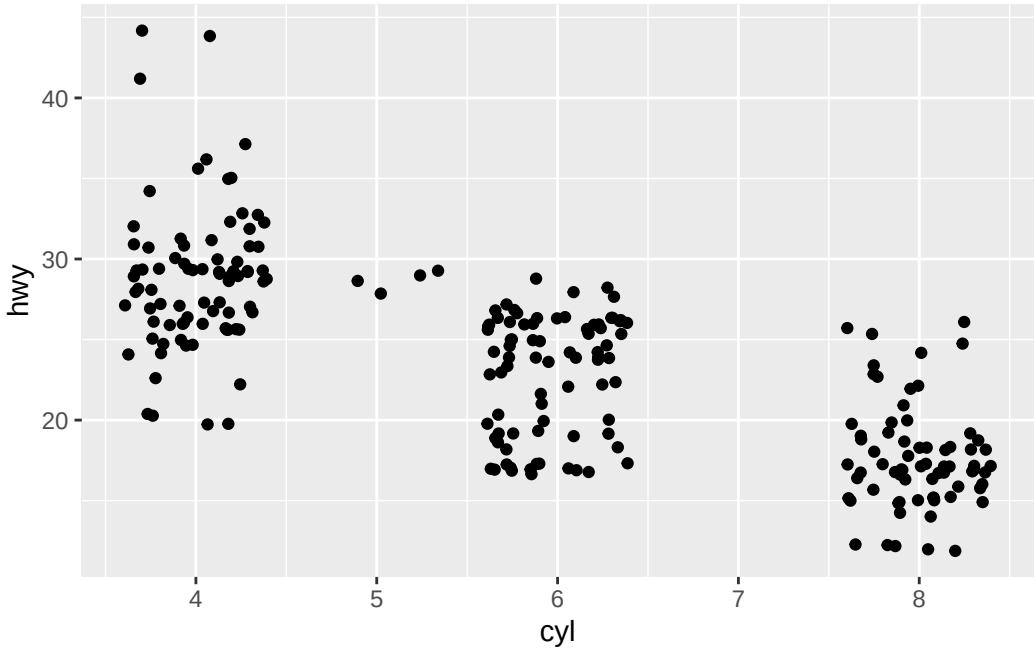
- Create a scatterplot of the relationship between number of cylinders in the engine (x-axis) and highway miles-per-gallon (y-axis)
- Below plot is difficult to interpet because many points overlap with one another

```
ggplot(data = mpg, mapping = aes(x = cyl, y = hwy)) +
  geom_point()
```



- The `jitter` position adjustment "adds a small amount of random variation to the location of each point" (from `?geom_jitter`)

```
ggplot(data = mpg, mapping = aes(x = cyl, y = hwy)) +
  geom_point(position = "jitter")
```



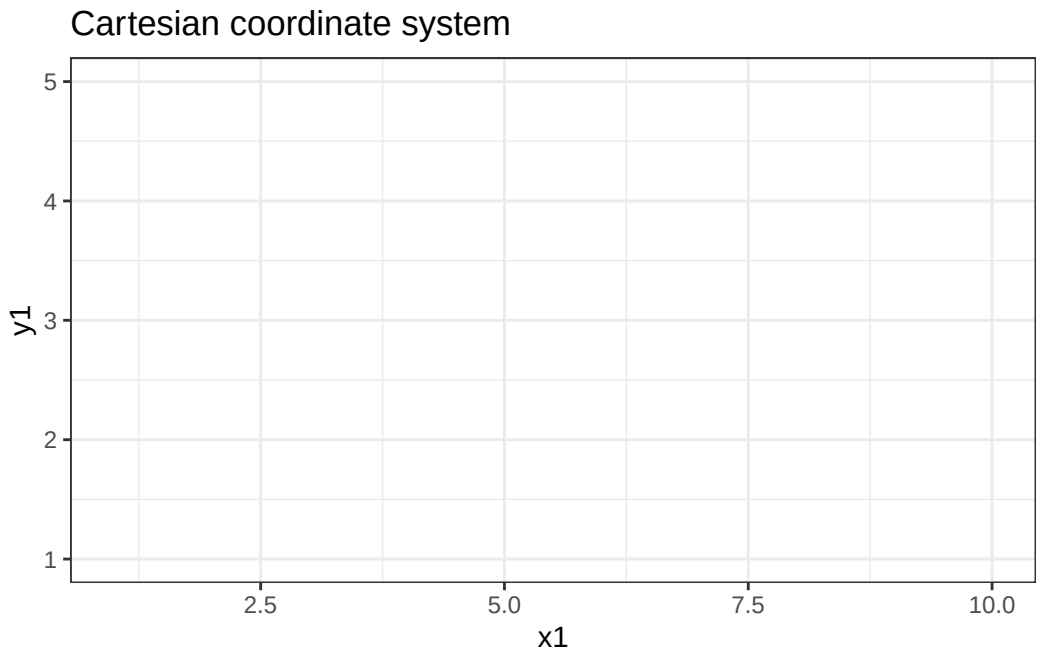
Coordinate system (coord)

"A **coordinate system** maps the position of objects onto the plane of the plot, and controls how the axes and grid lines are drawn. Plots typically use two coordinates (x,y), but could use any number of coordinates." (Grammar of Graphics)

Example: Cartesian coordinate system

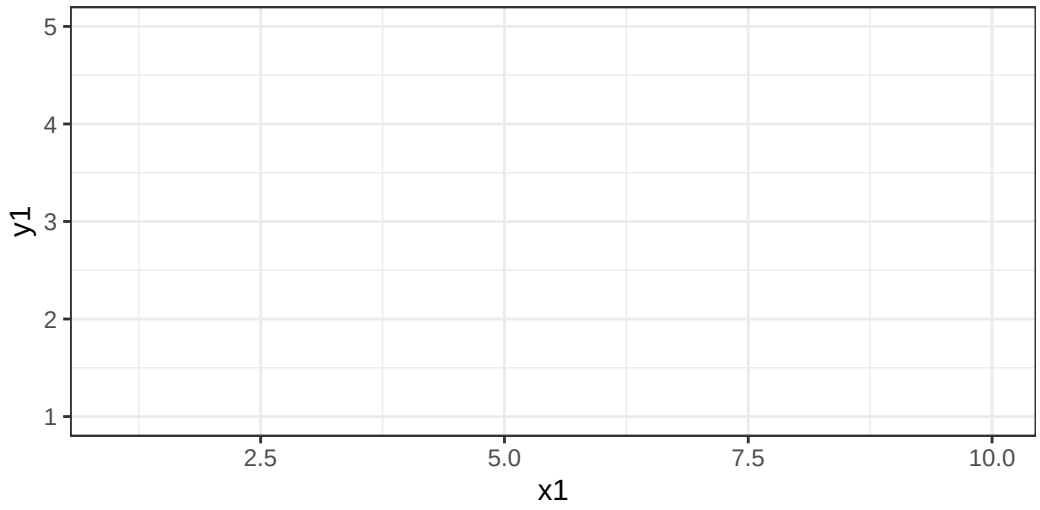
- Most plots use the Cartesian coordinate system

```
x1 <- c(1, 10)
y1 <- c(1, 5)
p <- qplot(x = x1, y = y1, geom = "blank", xlab = NULL, ylab = NULL) +
  theme_bw()

p +
  ggtitle(label = "Cartesian coordinate system")
```



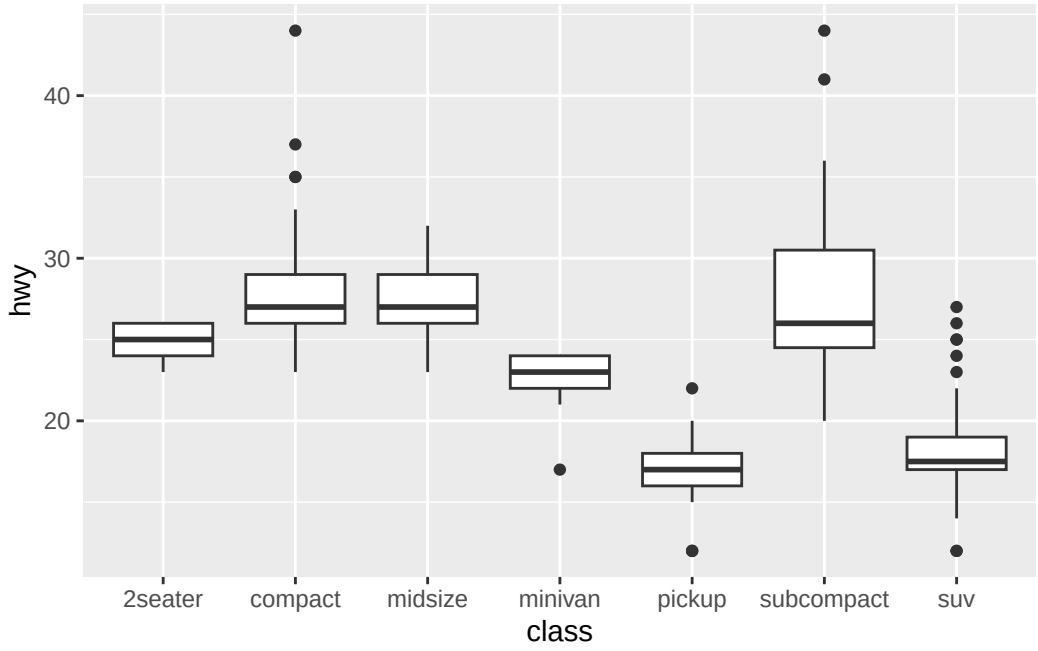
- Use `coord_fixed()` to fix the scaling of the coordinate system

```
p +
  coord_fixed()
```

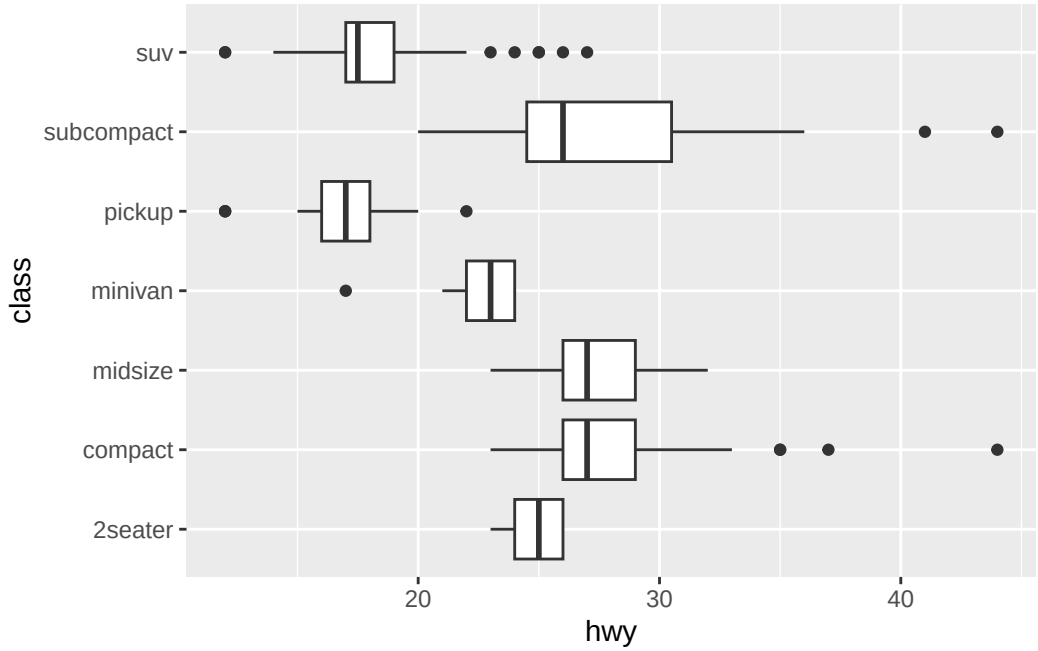


- When using the default Cartesian coordinate system, a common task is to flip the x and y axis using `coord_flip()`. (From R for Data Science)

```
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
  geom_boxplot()
```

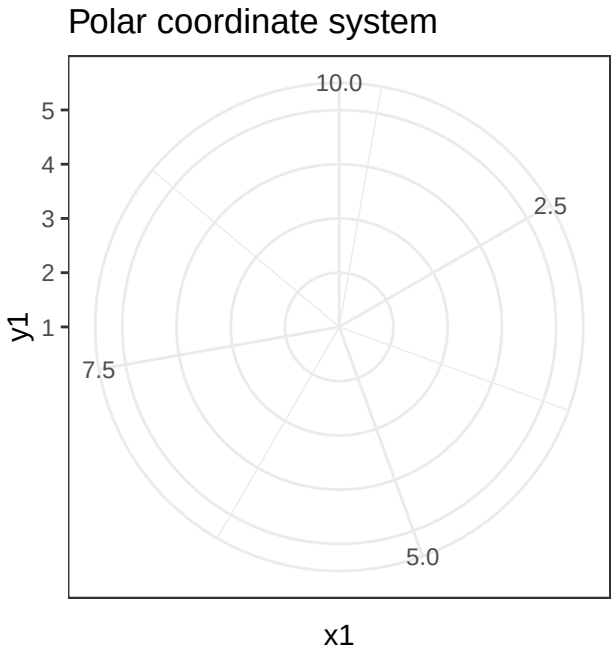



```
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
  geom_boxplot() +
  coord_flip()
```



Example: Polar coordinate system

```
p +
  coord_polar() +
  ggtitle(label = "Polar coordinate system")
```

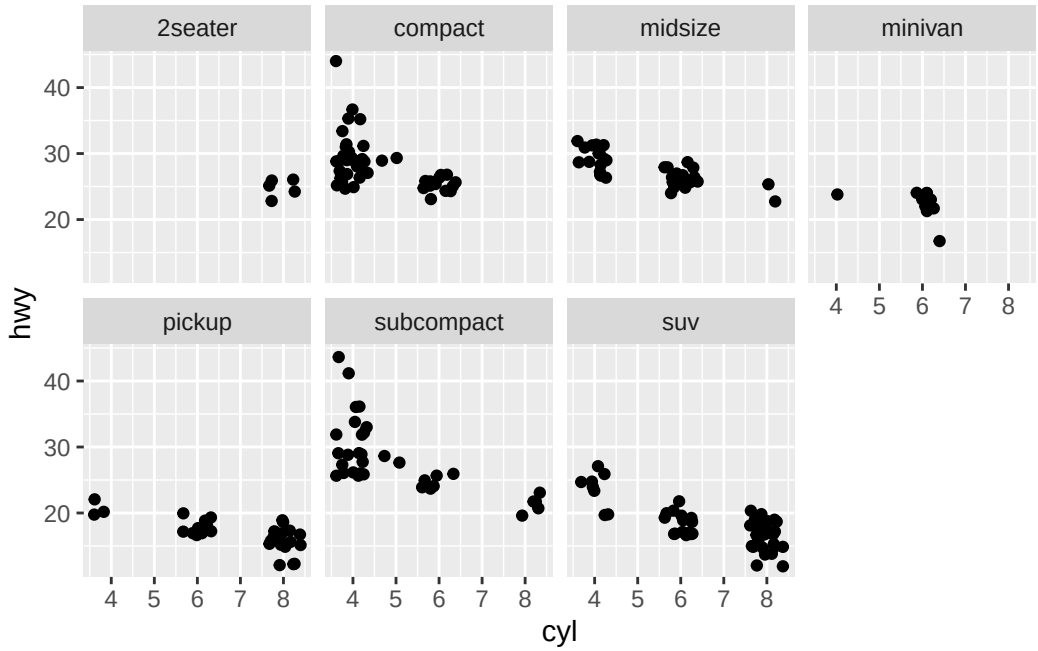


Faceting scheme (facets)

Facets are subplots that display one subset of the data. They are most commonly used to create "small multiples"

Example: Imagine creating a scatterplot of the relationship between number of cylinders in the engine (x-axis) and highway miles-per-gallon (y-axis), with separate subplots for car `class` (e.g., midsize, minivan, pickup, suv)

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = cyl, y = hwy), position = "jitter") +
  facet_wrap(~ class, nrow = 2)
```



Creating graphs using `ggplot`

`ggplot()` and `aes()` functions

Show help pages for package `ggplot2`:

```
help(package = ggplot2)
```

The `ggplot()` function:

```
?ggplot

# SYNTAX AND DEFAULT VALUES
ggplot(data = NULL, mapping = aes())
```

- Description (from help file)
  - "`ggplot()` initializes a ggplot object. It can be used to declare the input data frame for a graphic and to specify the set of plot aesthetics intended to be common throughout all subsequent layers unless specifically overridden"
- Arguments

  - `data`: Dataset to use for plot. If not specified in `ggplot()` function, must be supplied in each layer added to the plot.
  - `mapping`: Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot.

The `aes()` function (often called within the `ggplot()` function):

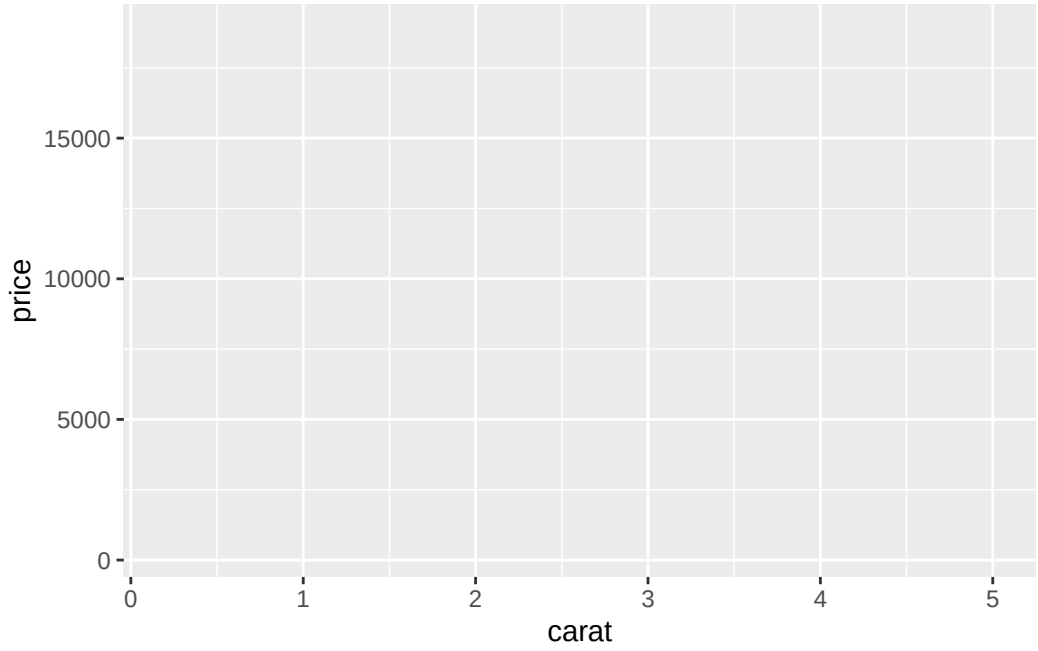
```
?aes

# SYNTAX
aes(x, y, ...)
```

- Description (from help file)
  - "Aesthetic mappings describe how variables in the data are mapped to visual properties (aesthetics) of geoms. Aesthetic mappings can be set in `ggplot()` and in individual layers."
- Arguments
  - `x, y, ...`: List of name value pairs giving aesthetics to map to variables
    - The names for `x` and `y` aesthetics are typically omitted because they are so common
    - All other aesthetics must be named

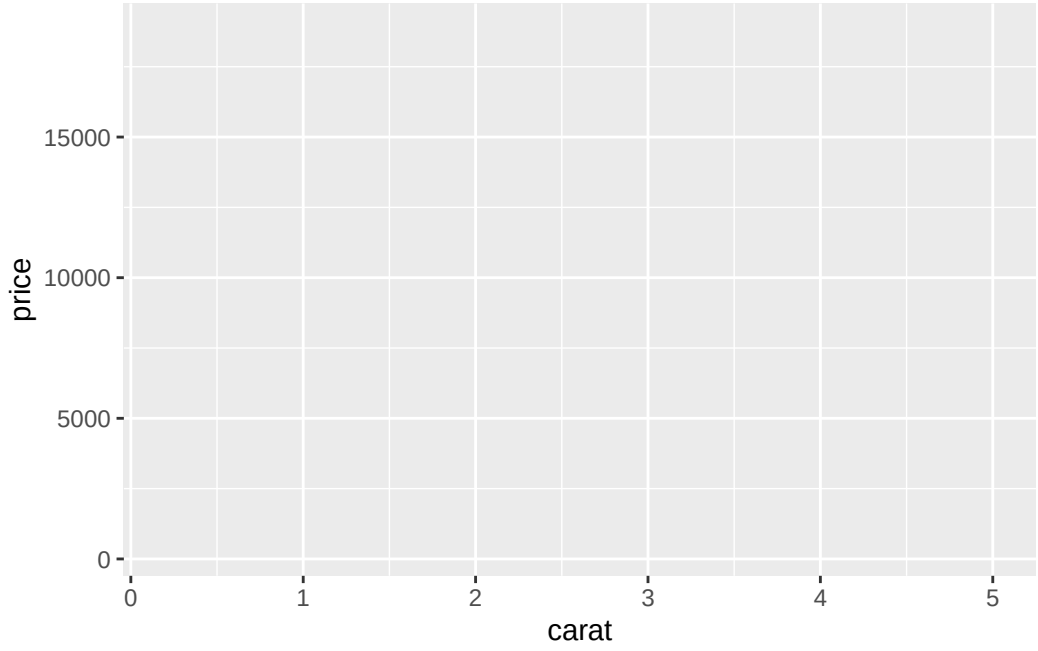
Example: Putting `ggplot()` and `aes()` together

- Specifying `ggplot()` and `aes()` without specifying a geom layer (e.g., `geom_point()`) creates a blank ggplot:
  - (The two lines of code below are functionally identical)

```
ggplot(data = diamonds, aes(x = carat, y = price))
```

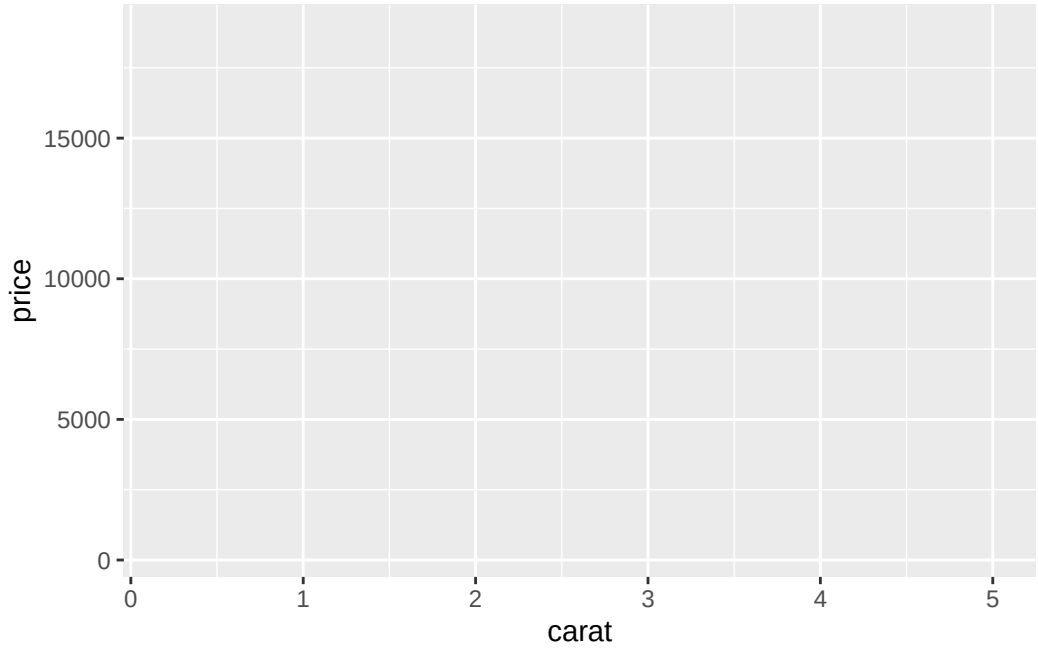



```
ggplot(data = diamonds, mapping = aes(x = carat, y = price))
```



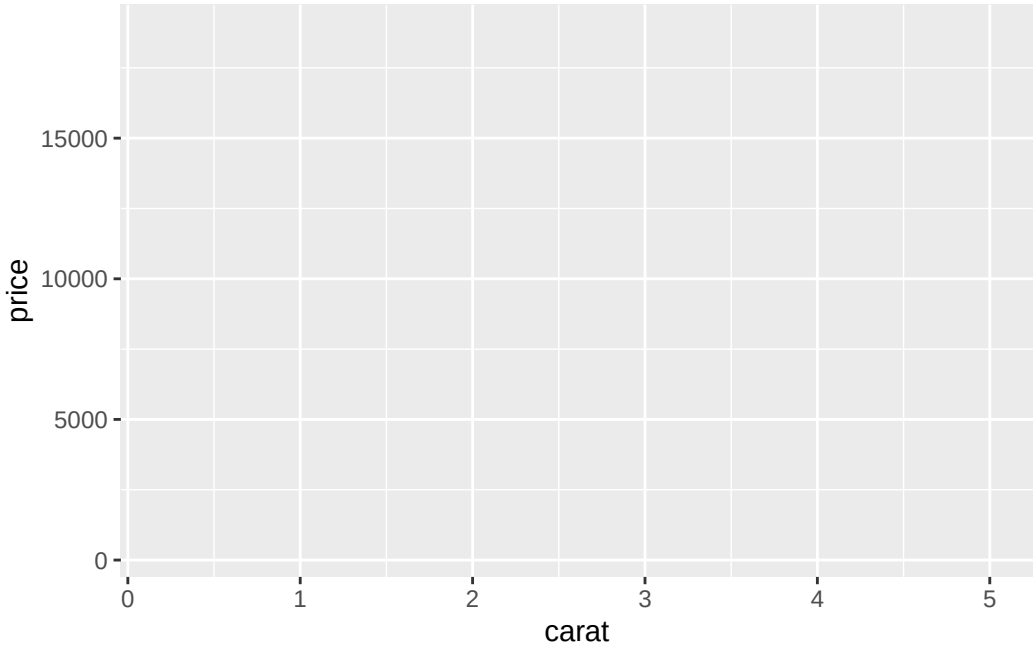
- Alternatively, we can use pipes with the dataframe we want to plot, which allows us to omit the first `data` argument of `ggplot()`:

```
class(diamonds)
#> [1] "tbl_df"     "tbl"        "data.frame"
diamonds %>% ggplot(mapping = aes(x = carat, y = price))
```



- We can also create a ggplot object and assign it to a variable for later use:

```
diam_ggplot <- ggplot(data = diamonds, aes(x = carat, y = price))

diam_ggplot # blank ggplot
```



- Investigate ggplot object:

```
typeof(diam_ggplot)
#> [1] "list"
class(diam_ggplot)
#> [1] "gg"     "ggplot"

str(diam_ggplot)
#> List of 9
#>  $ data       : tibble [53,940 x 10] (S3: tbl_df/tbl/data.frame)
#>   ..$ carat  : num [1:53940] 0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
#>   ..$ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...
#>   ..$ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...
#>   ..$ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 5 ...
#>   ..$ depth  : num [1:53940] 61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
#>   ..$ table  : num [1:53940] 55 61 65 58 58 57 57 55 61 61 ...
#>   ..$ price  : int [1:53940] 326 326 327 334 335 336 336 337 337 338 ...
#>   ..$ x      : num [1:53940] 3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
#>   ..$ y      : num [1:53940] 3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
#>   ..$ z      : num [1:53940] 2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
#>  $ layers     : list()
#>  $ scales     :Classes 'ScalesList', 'ggproto', 'gg' <ggproto object: Class ScalesList, gg
#>     add: function
```

```
#>     clone: function
#>     find: function
#>     get_scales: function
#>     has_scale: function
#>     input: function
#>     n: function
#>     non_position_scales: function
#>     scales: NULL
#>     super:  <ggproto object: Class ScalesList, gg>
#>  $ mapping    :List of 2
#>   ..$ x: language ~carat
#>   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
#>   ..$ y: language ~price
#>   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
#>   ..- attr(*, "class")= chr "uneval"
#>  $ theme      : list()
#>  $ coordinates:Classes 'CoordCartesian', 'Coord', 'ggproto', 'gg' <ggproto object: Class C
#>     aspect: function
#>     backtransform_range: function
#>     clip: on
#>     default: TRUE
#>     distance: function
#>     expand: TRUE
#>     is_free: function
#>     is_linear: function
#>     labels: function
#>     limits: list
#>     modify_scales: function
#>     range: function
#>     render_axis_h: function
#>     render_axis_v: function
#>     render_bg: function
#>     render_fg: function
#>     setup_data: function
#>     setup_layout: function
#>     setup_panel_guides: function
#>     setup_panel_params: function
#>     setup_params: function
#>     train_panel_guides: function
#>     transform: function
#>     super:  <ggproto object: Class CoordCartesian, Coord, gg>
#>  $ facet      :Classes 'FacetNull', 'Facet', 'ggproto', 'gg' <ggproto object: Class FacetN
```

```
#>     compute_layout: function
#>     draw_back: function
#>     draw_front: function
#>     draw_labels: function
#>     draw_panels: function
#>     finish_data: function
#>     init_scales: function
#>     map_data: function
#>     params: list
#>     setup_data: function
#>     setup_params: function
#>     shrink: TRUE
#>     train_scales: function
#>     vars: function
#>     super:  <ggproto object: Class FacetNull, Facet, gg>
#>  $ plot_env   :<environment: R_GlobalEnv>
#>  $ labels     :List of 2
#>   ..$ x: chr "carat"
#>   ..$ y: chr "price"
#>  - attr(*, "class")= chr [1:2] "gg" "ggplot"
```

- Attributes of ggplot object:

```
attributes(diam_ggplot)
#> $names
#> [1] "data"        "layers"      "scales"      "mapping"     "theme"
#> [6] "coordinates" "facet"       "plot_env"    "labels"
#>
#> $class
#> [1] "gg"     "ggplot"

diam_ggplot$mapping
#> Aesthetic mapping:
#> * `x` -> `carat`
#> * `y` -> `price`

diam_ggplot$labels
#> $x
#> [1] "carat"
#>
#> $y
#> [1] "price"
```

Adding geometric layers

Adding a **geometric layer** to a ggplot object dictates how observations are displayed in the plot.

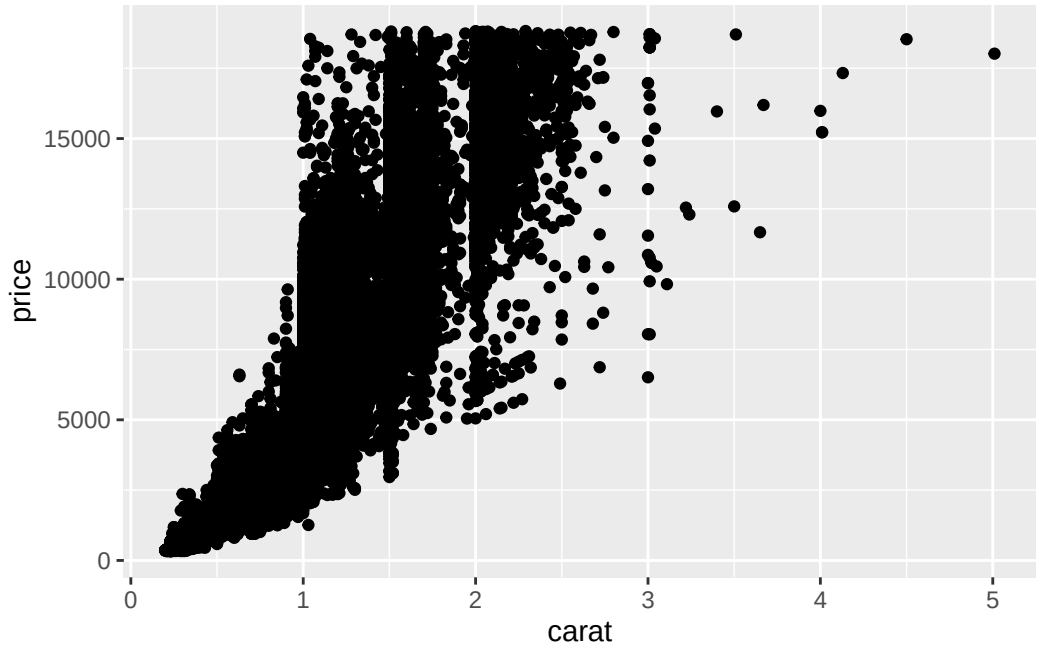
- Geometric layers are specified using "geom functions"
- There are many different geom functions:
  - `geom_point()`: creates a scatterplot
  - `geom_bar()`: creates a bar chart
  - etc.

Scatterplots using `geom_point()`

Scatterplots are most useful for showing the relationship between two continuous variables.

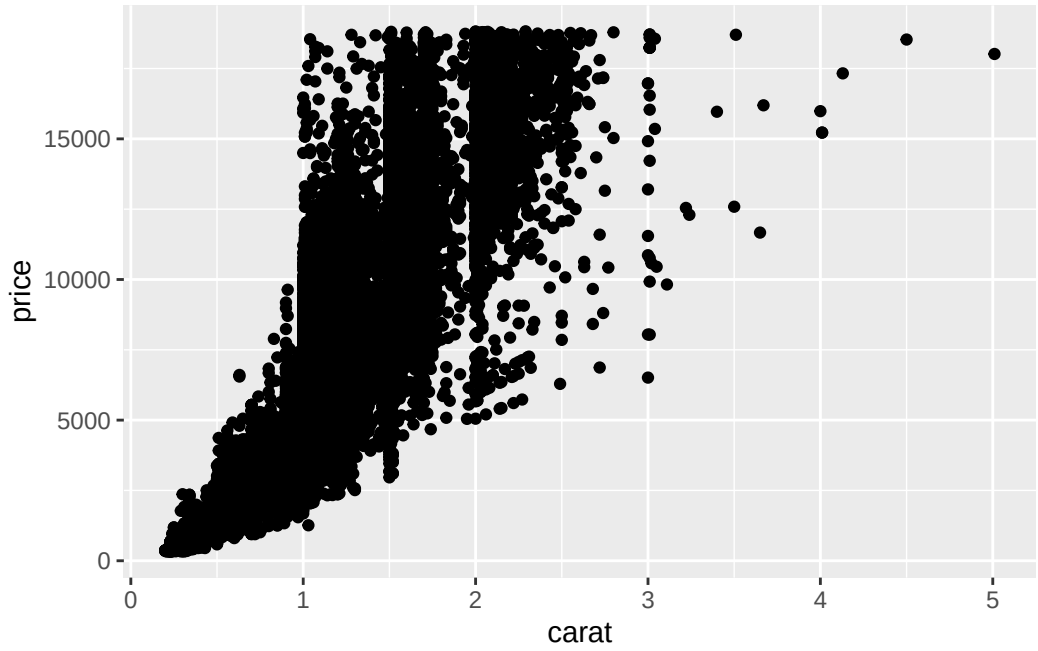
Example: Scatterplot of the relationship between `carat` and `price`, using the `diamonds` dataset

```
#ggplot(data = diamonds, aes(x = carat, y = price)) + geom_point()
ggplot(data = diamonds, mapping = aes(x = carat, y = price)) + geom_point()
```



- If we already created and assigned a ggplot object, we can use that object to create the plot:

```
diam_ggplot + geom_point()
```



Example: Scatterplot of the hours/week spent on homework (`bys34a`) and 2011 earnings (`f3ern2011`), using the `els` dataset

- First, let's investigate the underlying variables:

```
els %>% select(bys34a,f3ern2011) %>%
  summarize_all(.funs = list(~ mean(., na.rm = TRUE), ~ min(., na.rm = TRUE), ~ max(., na.rm
#> # A tibble: 1 x 6
#>   bys34a_mean f3ern2011_mean bys34a_min f3ern2011_min bys34a_max f3ern2011_max
#>         <dbl>          <dbl>      <dbl>         <dbl>      <dbl>         <dbl>
#> 1        3.65         21276.         -9            -8         21        250000
```

- Investigate values less than zero:

```
els %>% select(bys34a) %>% filter(bys34a<0) %>% count(bys34a)
#> # A tibble: 4 x 2
#>   bys34a     n
#>    <dbl> <int>
#> 1     -9   572
#> 2     -8   305
#> 3     -4   648
```

```
#> 4     -1     2

els %>% select(f3ern2011) %>% filter(f3ern2011<0) %>% count(f3ern2011)
#> # A tibble: 2 x 2
#>   f3ern2011     n
#>       <dbl> <int>
#> 1        -8   459
#> 2        -4  2488
```

- Create version of variables that replace values less than zero with `NA`:

```
els_v2 <- els %>%
  mutate(
    hw_time = if_else(bys34a<0,NA_real_,as.numeric(bys34a)),
    earn2011 = if_else(f3ern2011<0,NA_real_,as.numeric(f3ern2011)),
  )

#check
els_v2 %>% filter(bys34a<0) %>% count(bys34a, hw_time)
#> # A tibble: 4 x 3
#>   bys34a hw_time     n
#>    <dbl>   <dbl> <int>
#> 1     -9      NA   572
#> 2     -8      NA   305
#> 3     -4      NA   648
#> 4     -1      NA     2
els_v2 %>% filter(f3ern2011<0) %>% count(f3ern2011, earn2011)
#> # A tibble: 2 x 3
#>   f3ern2011 earn2011     n
#>       <dbl>    <dbl> <int>
#> 1        -8       NA   459
#> 2        -4       NA  2488
```

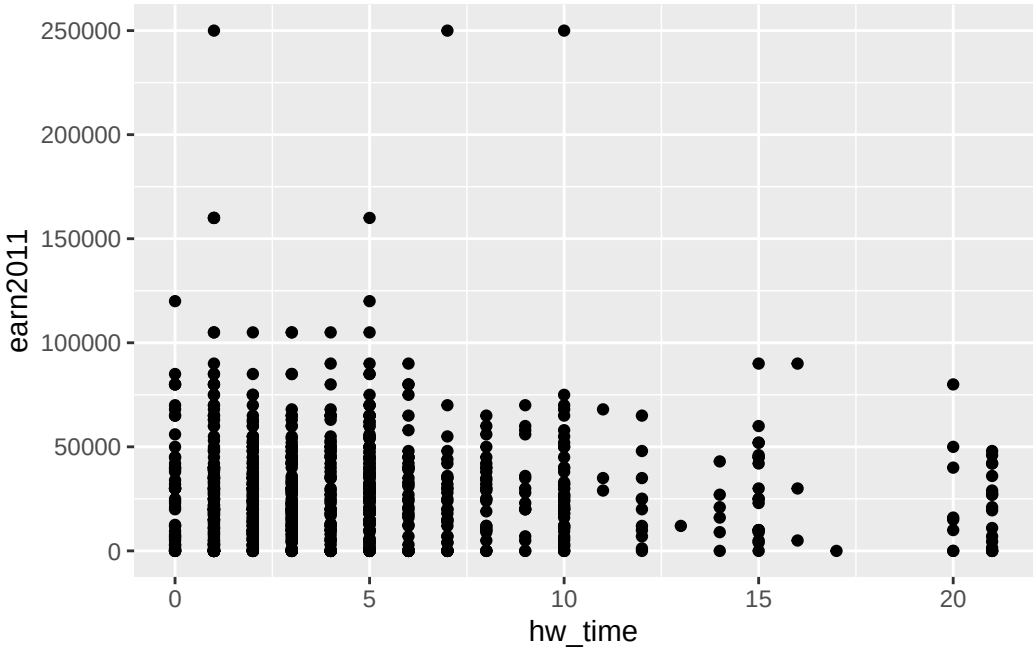
- To avoid scatterplot with too many points, create a dataframe consisting of students whose parents have a PhD or first professional degree:

```
els_v2 %>% count(f1pared)
#> # A tibble: 8 x 2
#>   f1pared                                    n
#>   <fct>                                  <int>
#> 1 Did not finish high school              1020
#> 2 Graduated from high school or GED       3250
```

```
#> 3 Attended 2-year school, no degree         1736
#> 4 Graduated from 2-year school              1660
#> 5 Attended college, no 4-year degree        1820
#> 6 Graduated from college                    3663
#> 7 Completed Master's degree or equivalent   1927
#> 8 Completed PhD, MD, other advanced degree  1121

els_v2$f1pared %>% class()
#> [1] "factor"
els_v2$f1pared %>% attributes()
#> $levels
#> [1] "Did not finish high school"
#> [2] "Graduated from high school or GED"
#> [3] "Attended 2-year school, no degree"
#> [4] "Graduated from 2-year school"
#> [5] "Attended college, no 4-year degree"
#> [6] "Graduated from college"
#> [7] "Completed Master's degree or equivalent"
#> [8] "Completed PhD, MD, other advanced degree"
#>
#> $class
#> [1] "factor"
#>
#> $label
#> [1] "F1 parent's highest level of education"

# els_v2$f1pared is a factor class variable so we filter based on factor levels
els_parphd <- els_v2 %>% filter(f1pared=="Completed PhD, MD, other advanced degree")
```

- Plot the scatterplot:

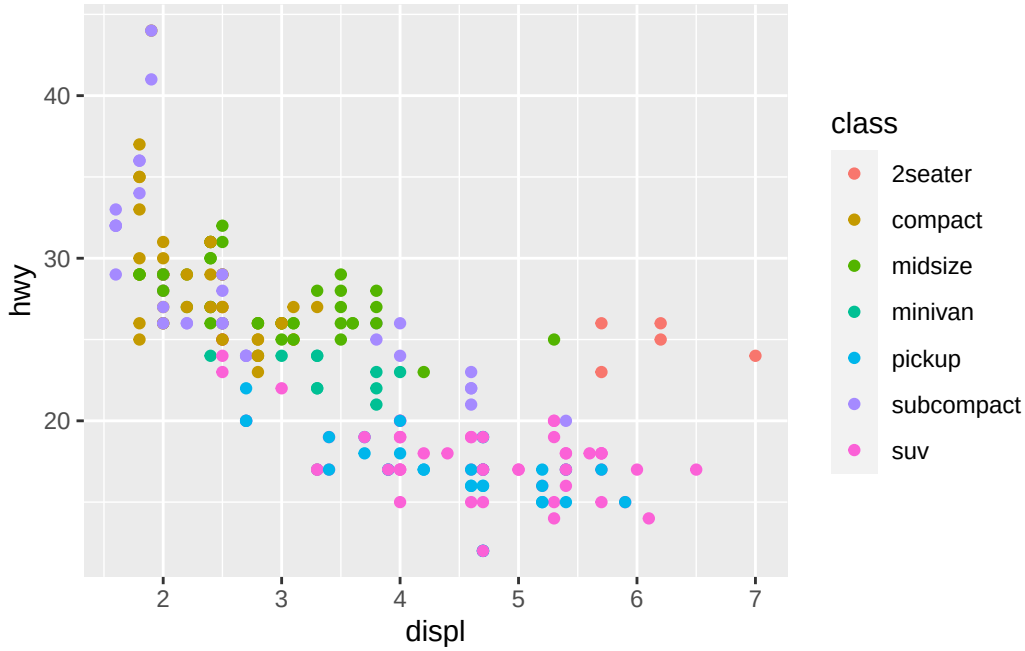
```
ggplot(data= els_parphd, aes(x = hw_time, y = earn2011)) + geom_point()
```



Example: Scatterplot of the relationship between engine displacement (`displ`) and highway miles-per-gallon (`hwy`), using the `mpg` dataset

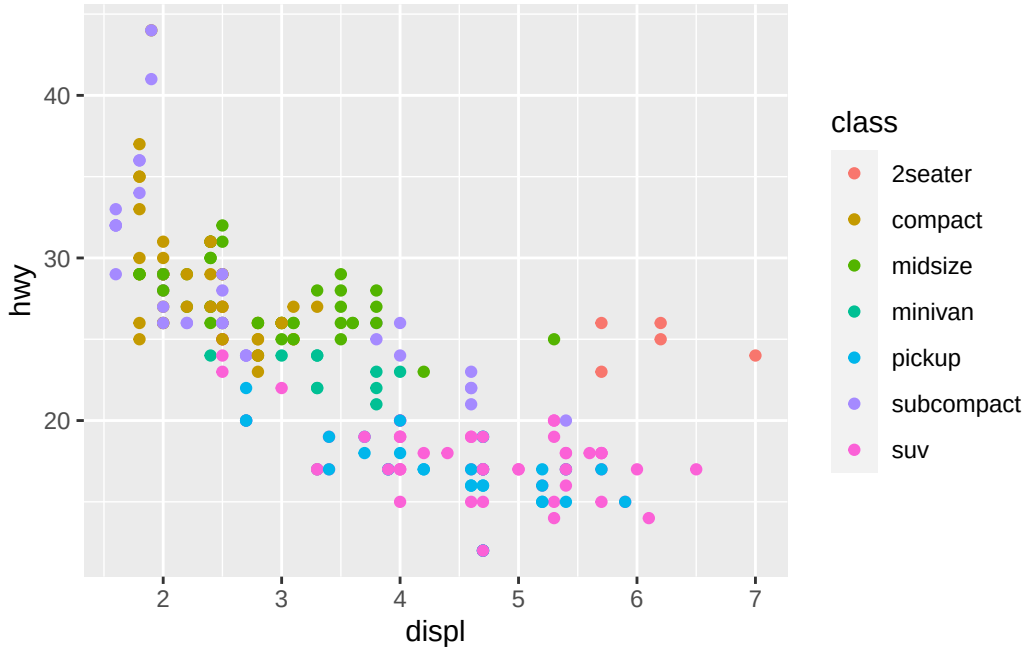
- Color of points determined by type of car (`class`):

```
ggplot(data = mpg, aes(x = displ, y = hwy, color = class)) +
  geom_point()
```



- Alternatively, the `color` aesthetic can be specified within `geom_point()`:

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(color = class))
```



The `geom_point()` function:

```
?geom_point

# SYNTAX AND DEFAULT VALUES
geom_point(mapping = NULL, data = NULL, stat = "identity",
           position = "identity", ..., na.rm = FALSE, show.legend = NA,
           inherit.aes = TRUE)
```

- note that the default statistical transformation is `stat = "identity"`
  - that is, we simply plot values of `x` and `y` on the Cartesian coordinate system; we don't perform some kind of statistical transformation before plotting
- The `mapping` argument determines Aesthetics, like this:
  - `geom_point(mapping = aes(...))`
- Aesthetics previously stated from `ggplot(mapping = aes(...))` will be carried forward unless you explicitly change them within `geom_point(mapping = aes(...))`
- For any geometric layer function (e.g., `geom_point`, `geom_bar`, `geom_boxplot`), the help file will tell you which aesthetics the function will accept
  - Aesthetics: `geom_point()` understands (i.e., accepts) the following aesthetics (required aesthetics in **bold**)
    - `x`, `y`, `alpha`, `colour`, `fill`, `group`, `shape`, `size`, `stroke`
  - Note: Other geom functions (e.g., `geom_bar()`) accepts a different set of aesthetics

Student Task: Using the `els_parphd` dataset, create a scatterplot of the relationship between hours/week spent on homework (`hw_time`) on the x-axis and 2011 earnings (`earn2011`) on the y-axis, with the color of points determined by sex (`f1sex`)

Solution

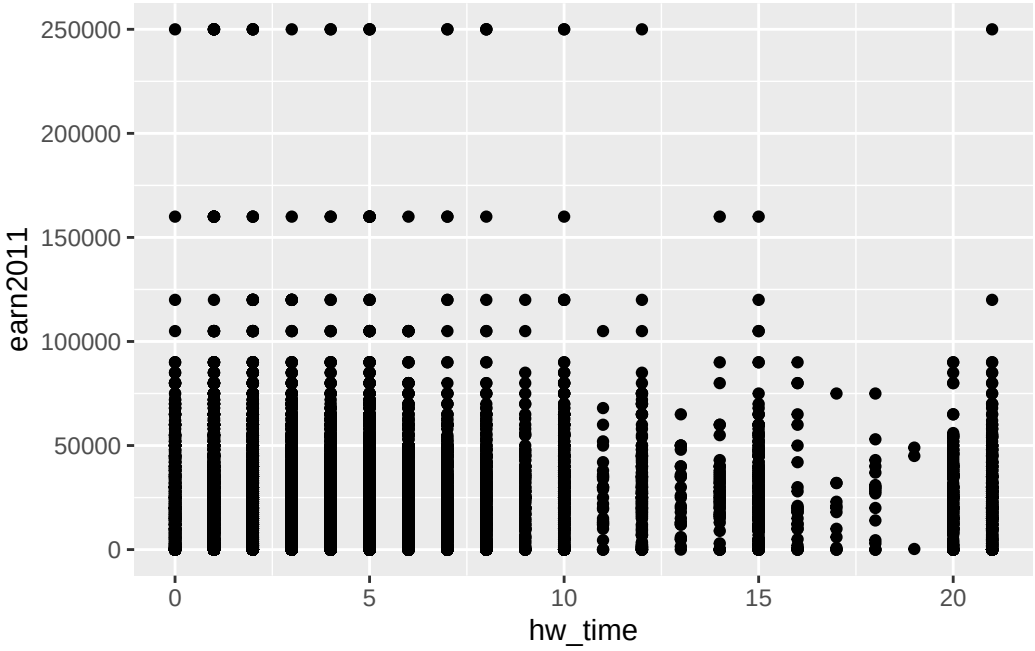
```
ggplot(data= els_parphd, aes(x = hw_time, y = earn2011, color = f1sex)) + geom_point()
```

Smoothed prediction lines using `geom_smooth()`

Why use `geom_smooth()`?

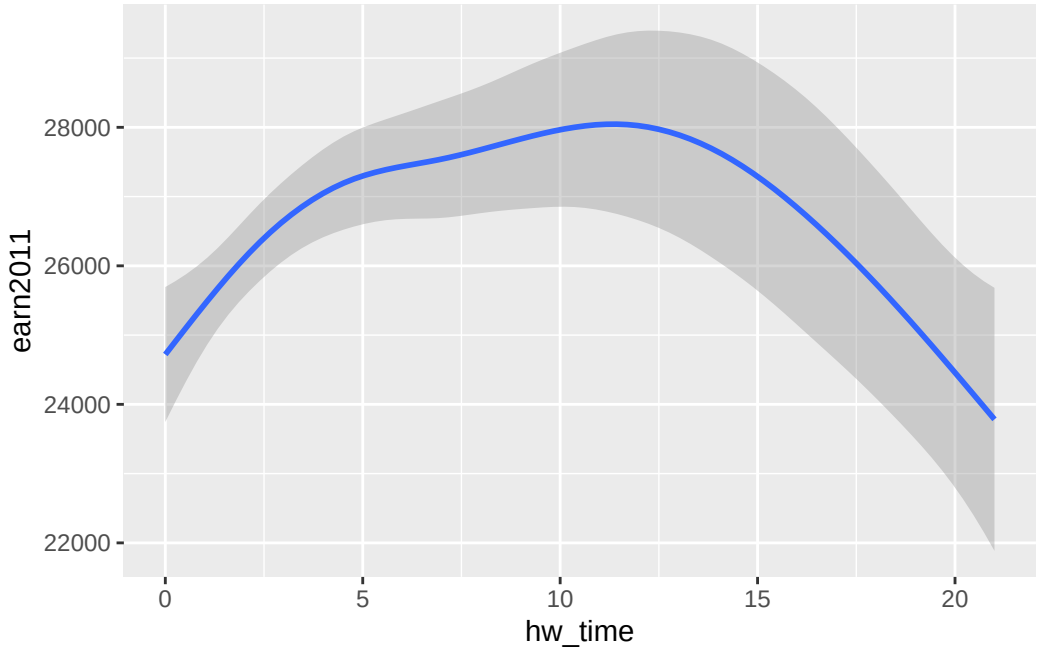
- The biggest problem with scatterplots is "overplotting." That is, when you plot many observations, points may be plotted on top of one another and it becomes difficult to visually discern the relationship:

```
ggplot(data = els_v2, aes(x = hw_time, y = earn2011)) + geom_point()
```



- Instead, using `geom_smooth()` creates smoothed prediction lines with shaded confidence intervals:

```
ggplot(data = els_v2, aes(x = hw_time, y = earn2011)) + geom_smooth()
```



The `geom_smooth()` function:

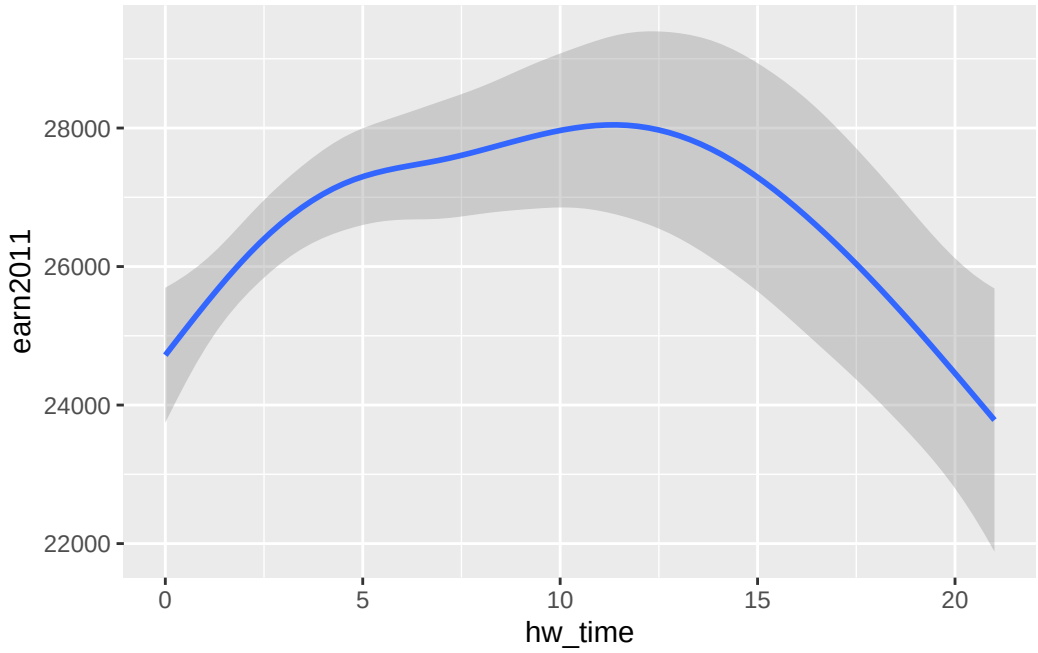
```
?geom_smooth

# SYNTAX AND DEFAULT VALUES
geom_smooth(mapping = NULL, data = NULL, stat = "smooth",
            position = "identity", ..., method = "auto", formula = y ~ x,
            se = TRUE, na.rm = FALSE, show.legend = NA, inherit.aes = TRUE)
```

- Arguments
  - Note default "statistical transformation" (`stat`), as compared to that of `geom_point()`:
    - `stat = "smooth"` for `geom_smooth()`
    - `stat = "identity"` for `geom_point()`
- Aesthetics: `geom_smooth()` accepts the following aesthetics (required aesthetics in **bold**)
  - **`x`**, **`y`**, `alpha`, `colour`, `fill`, `group`, `linetype`, `size`, `weight`, `ymax`, `ymin`

Example: Smoothed prediction lines for hours/week spent on homework (`bys34a`) versus 2011 earnings (`f3ern2011`), using the `els` dataset

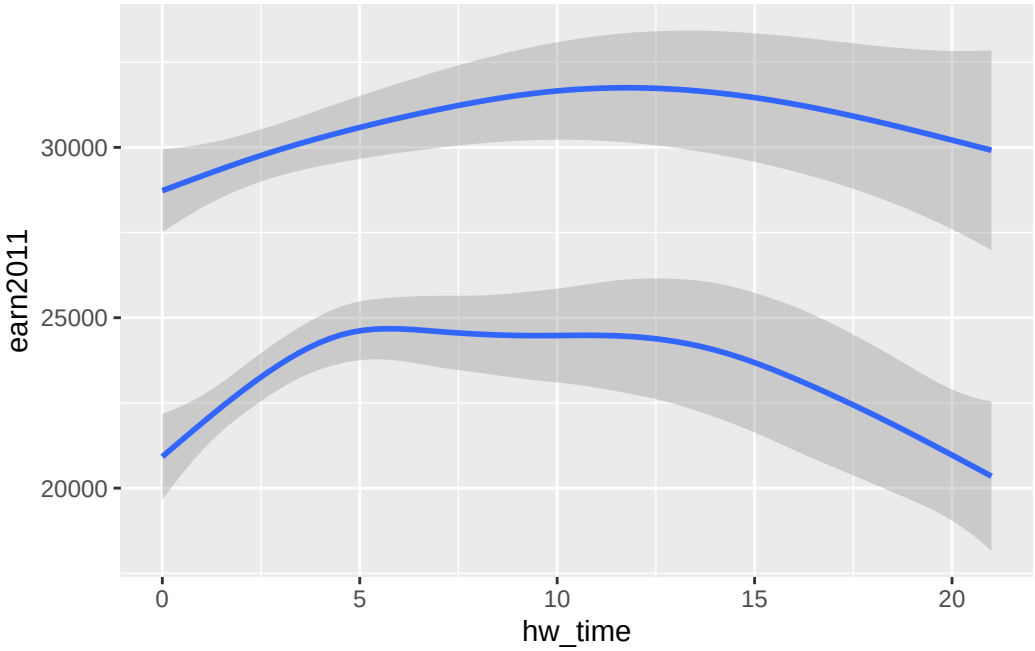
- This code produces same plot as above, when aesthetics were specified in `ggplot()`:

```
ggplot(data=els_v2) + geom_smooth(mapping = aes(x = hw_time, y = earn2011))
```



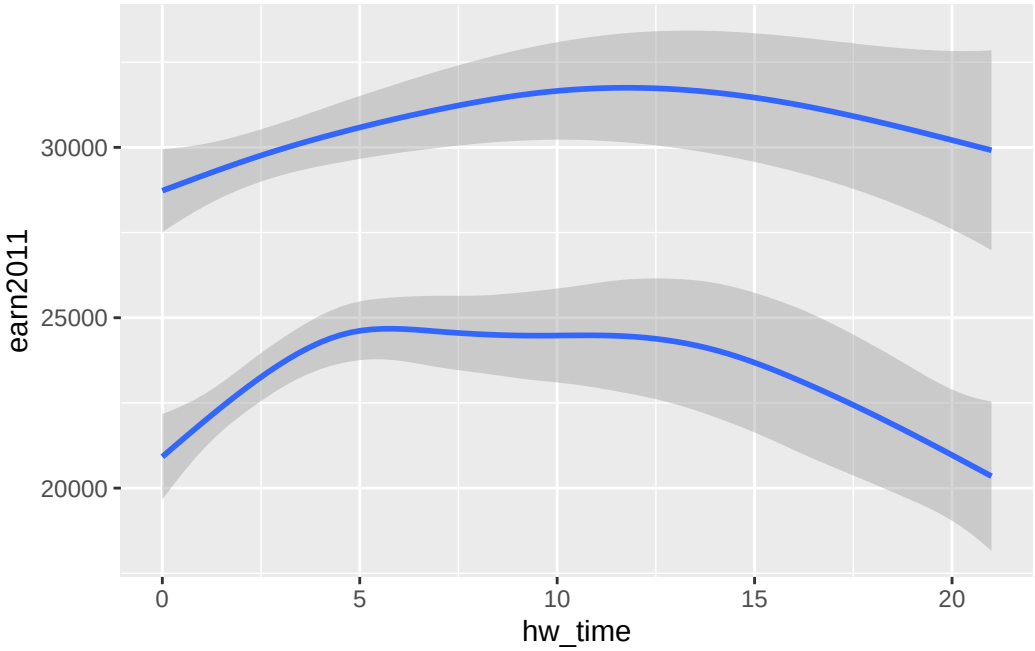
- Use `group` aesthetic to create separate prediction lines by sex (`f1sex`):

```
#ggplot(data=els_v2, aes(x = hw_time, y = earn2011, group=as_factor(f1sex))) + geom_smooth()
ggplot(data=els_v2) + geom_smooth(mapping = aes(x = hw_time, y = earn2011, group=f1sex))
```



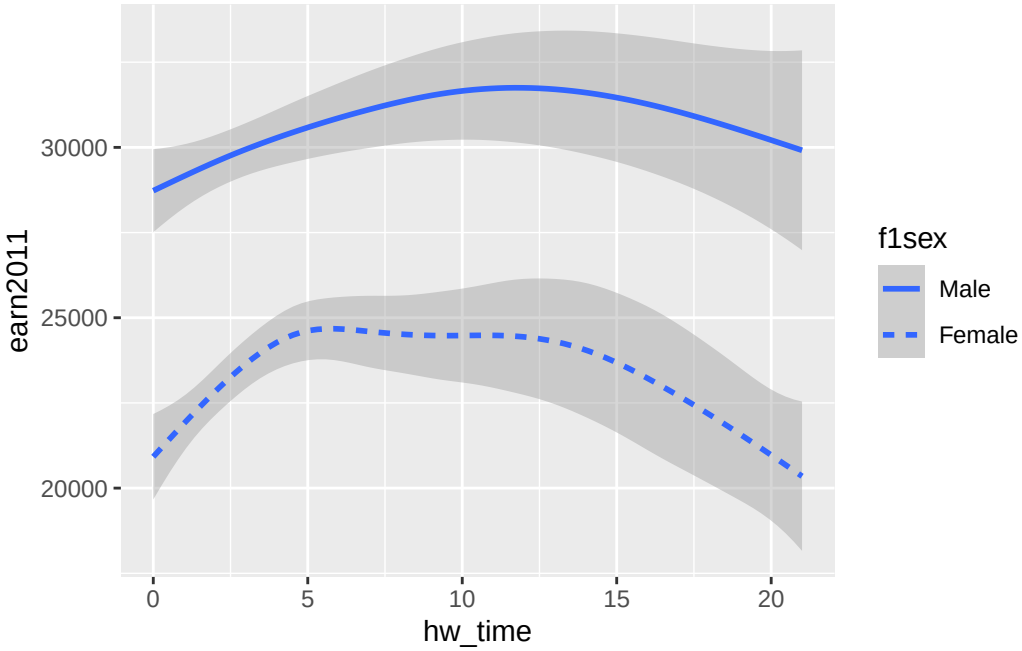
- Alternatively, we could produce the same plot by specifying all aesthetics – including the `group` aesthetic – within the `ggplot()` function rather than the `geom_smooth()` function:

```
ggplot(data = els_v2, aes(x = hw_time, y = earn2011, group=f1sex)) + geom_smooth()
```



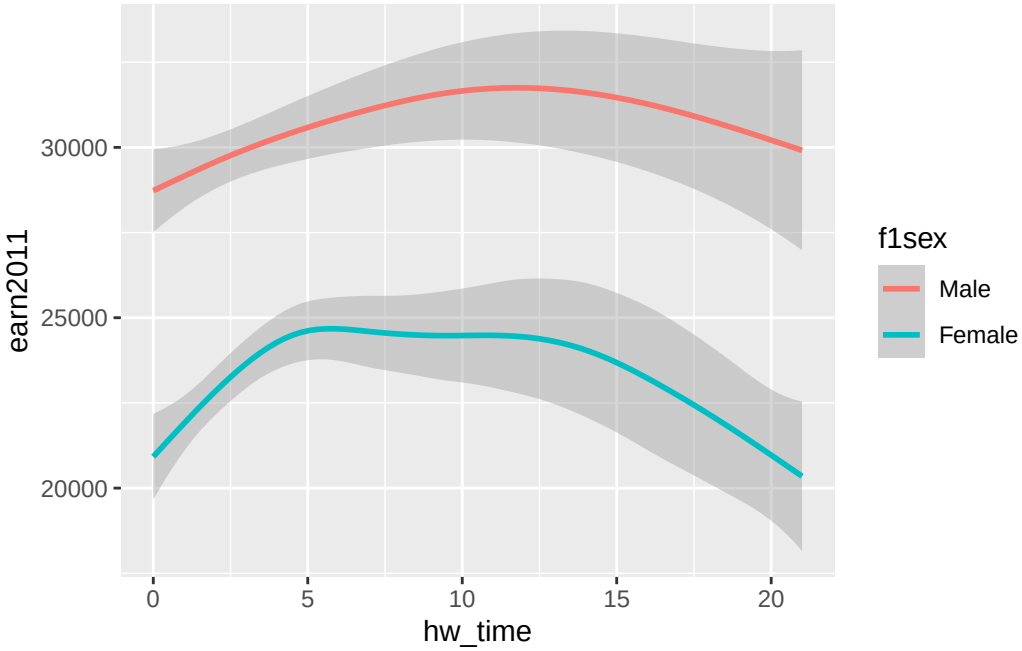
- Use `linetype` aesthetic to create separate prediction lines (with different line styles) by sex (`f1sex`):

```
#ggplot(data=els_v2, aes(x = hw_time, y = earn2011, linetype=as_factor(f1sex))) + geom_smoothl
ggplot(data=els_v2) + geom_smooth(mapping = aes(x = hw_time, y = earn2011, linetype=f1sex))
```



- Use `color` aesthetic to create separate prediction lines (with different colors) by sex (`f1sex`):

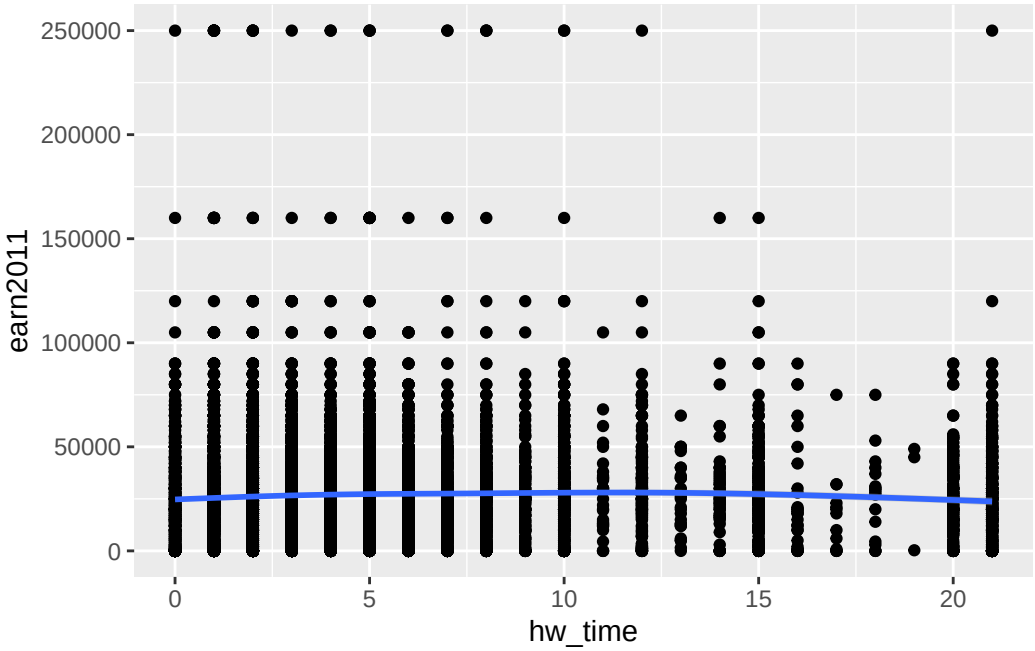
```
#ggplot(data=els_v2, aes(x = hw_time, y = earn2011, color=as_factor(f1sex))) + geom_smooth()
ggplot(data=els_v2) + geom_smooth(mapping = aes(x = hw_time, y = earn2011, color=f1sex))
```



Plotting multiple geom layers

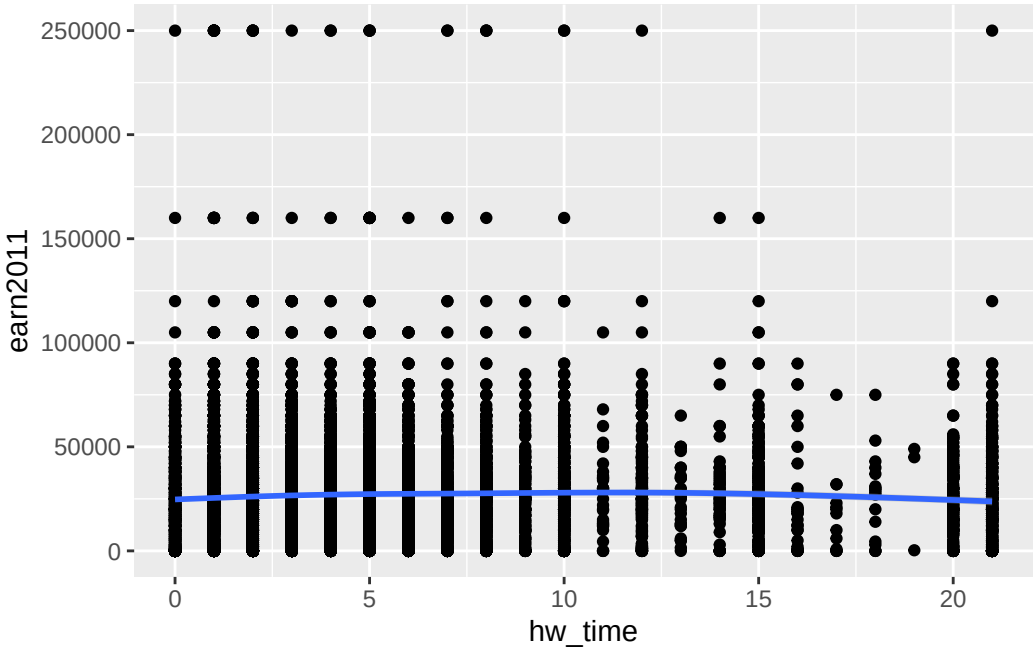
Example: Layer smoothed prediction lines (`geom_smooth()`) on top of scatterplot (`geom_point()`)

```
ggplot(data= els_v2) +
  geom_point(mapping = aes(x = hw_time, y = earn2011)) +
  geom_smooth(mapping = aes(x = hw_time, y = earn2011))
```



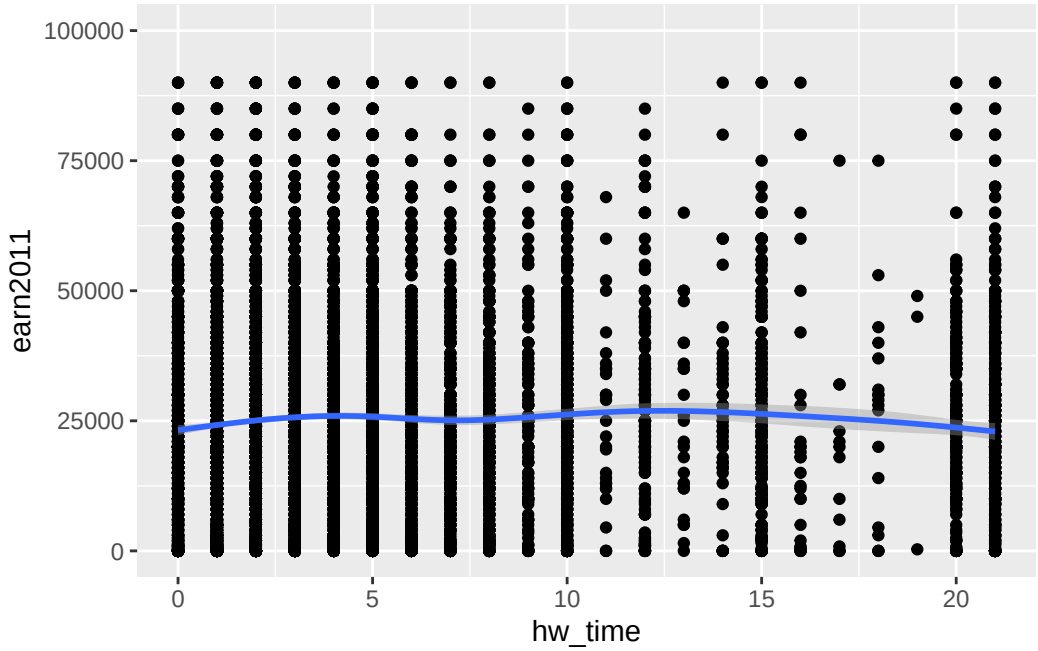
- Equivalently, the same plot can be created using this syntax:

```
ggplot(data= els_v2, aes(x = hw_time, y = earn2011)) +
  geom_point() +
  geom_smooth()
```



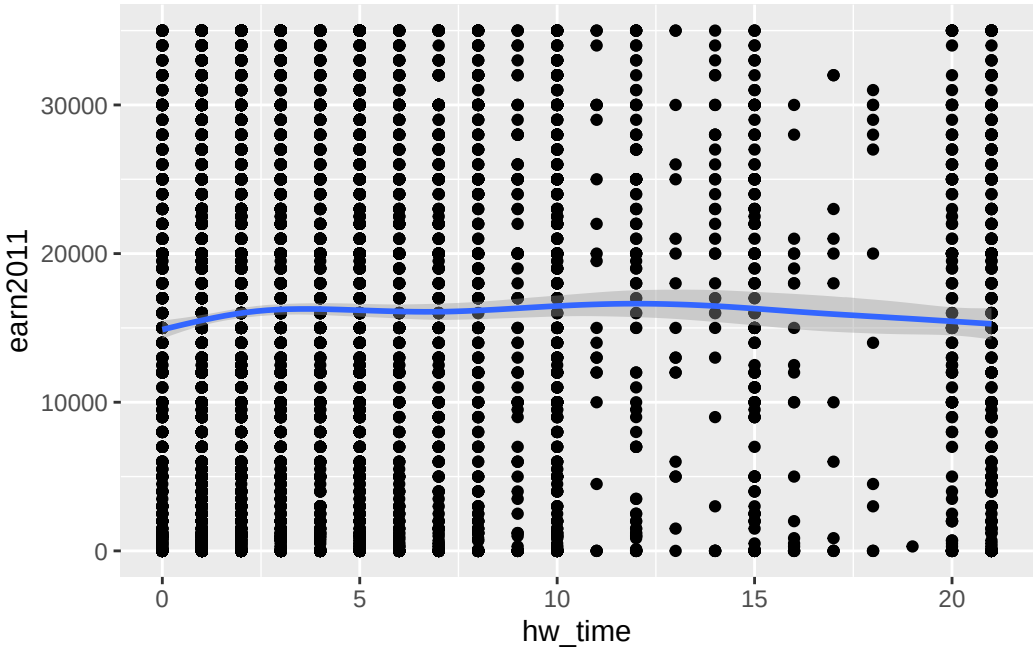
- Adjust x-axis and y-axis limits by using `+ xlim()` and `+ ylim()`:

```
ggplot(data= els_v2, aes(x = hw_time, y = earn2011)) +
  geom_point() +
  geom_smooth() +
  xlim(c(0,21)) + ylim(c(0,100000))
```



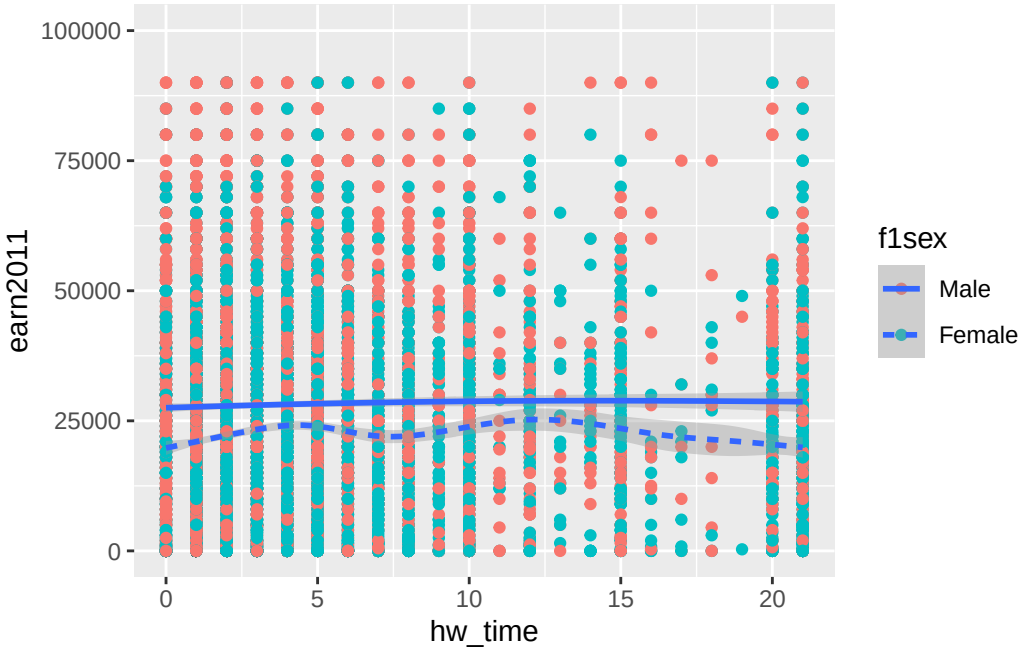
- Let's try smaller y-axis limits:
  - **Note**. Observations with income values above the y-axis limit are removed from the plot. This includes being removed from the calculation of the smooted prediction line

```
ggplot(data= els_v2, aes(x = hw_time, y = earn2011)) +
  geom_point() +
  geom_smooth() +
  xlim(c(0,21)) + ylim(c(0,35000))
```



- Layer smoothed prediction lines with different line types by sex (`f1sex`) on top of scatterplot with different point colors by sex:

```
ggplot(data= els_v2) +
  geom_point(mapping = aes(x = hw_time, y = earn2011, color = f1sex)) +
  geom_smooth(mapping = aes(x = hw_time, y = earn2011, linetype = f1sex)) +
  xlim(c(0,21)) + ylim(c(0,100000))
```



Bar charts using `geom_bar()` and `geom_col()`

Bar charts are used to plot a single, discrete variable.

- X-axis typically represents a categorical variable (e.g,. race, sex, institutional type)
  - Each value of the categorical variable is a "group"
- Y-axis often represents the number of cases in a group (or the proportion of cases in a group)
  - But height of bar could also represent mean value for a group or some other summary statistic (e.g., min, max, std)

Two geom functions to create bar charts:

- `geom_bar()`: The height of each bar represents the number of cases (i.e., observations) in the group
  - Statistical transformation = "count"
    - Y-value for a group is the number of cases in the group
  - Use `geom_bar()` when using (for example) student-level data and you don't want to summarize student-level data prior to creating the chart
- `geom_col()`: The height of each bar represents the value of some variable for the group

  - Statistical transformation = "identity"
    - Y-value for a group is the value of a variable in the dataframe
  - Use `geom_col()` when you have already created an object of summary statistics (e.g., counts, mean value, etc.)

The `geom_bar()` and `geom_col()` functions:

```
?geom_bar

# SYNTAX AND DEFAULT VALUES
geom_bar(mapping = NULL, data = NULL, stat = "count",
         position = "stack", ..., width = NULL, binwidth = NULL,
         na.rm = FALSE, show.legend = NA, inherit.aes = TRUE)


?geom_col

# SYNTAX AND DEFAULT VALUES
geom_col(mapping = NULL, data = NULL, position = "stack", ...,
         width = NULL, na.rm = FALSE, show.legend = NA,
         inherit.aes = TRUE)
```

- Both `geom_bar` and `geom_col` accept the following aesthetics:
  - x, y, alpha, colour, fill, group, linetype, size

Example: Bar chart with the variable `cut` (e.g., "Fair," "Good," "Ideal") as x-axis and number of diamonds as y-axis, using the `diamonds` dataset

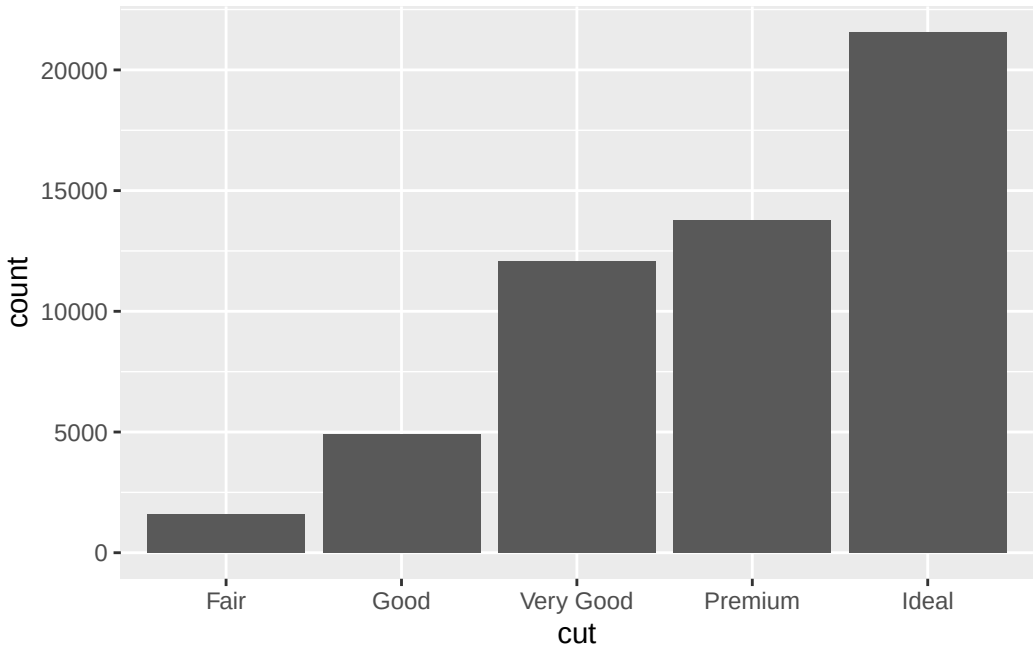
- Essentially, you are being asked to create a bar chart from the following frequency count:

```
diamonds %>% count(cut)
#> # A tibble: 5 x 2
#>   cut           n
#>   <ord>     <int>
#> 1 Fair       1610
#> 2 Good       4906
#> 3 Very Good 12082
#> 4 Premium   13791
#> 5 Ideal     21551
```

Method 1: Create bar chart using `geom_bar()`

- note: `geom_bar()` uses `stat = "count"`

```
ggplot(data = diamonds, aes(x = cut)) +
  geom_bar()
```



Method 2: Create bar chart using `geom_col()`

By contrast, help file says `geom_col()` uses " uses `stat_identity()`: it leaves the data as is."

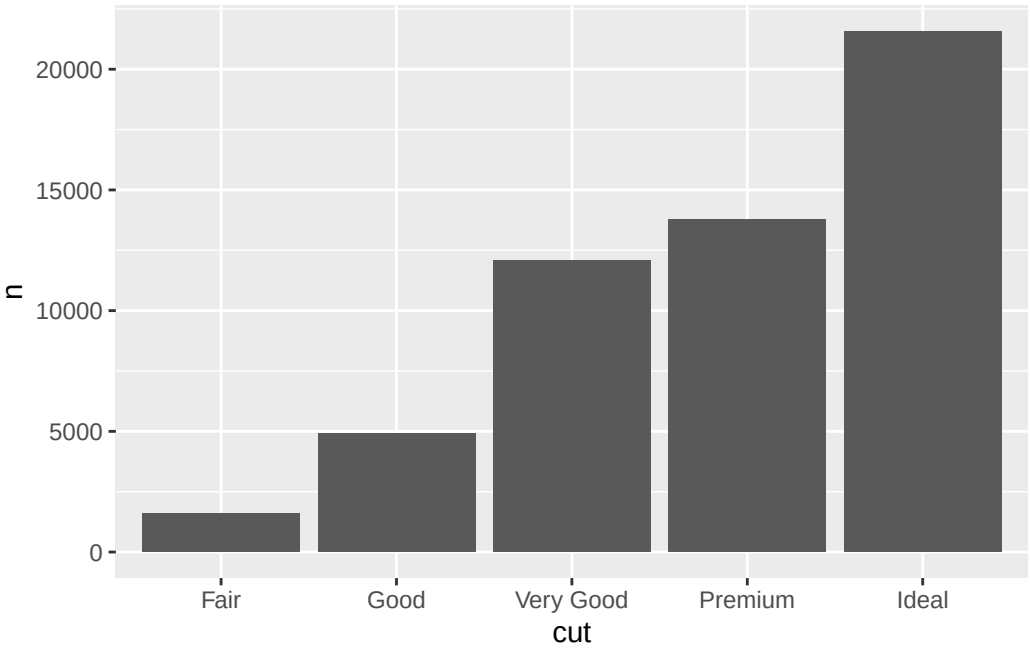
- So before we create chart using `geom_col()` we create an object that contains the frequency count for the variable `cut`:

```
cut_count <- diamonds %>% count(cut)
cut_count
#> # A tibble: 5 x 2
#>   cut           n
#>   <ord>     <int>
#> 1 Fair       1610
#> 2 Good       4906
#> 3 Very Good 12082
#> 4 Premium   13791
#> 5 Ideal     21551

cut_count %>% str()
```

```
#> tibble [5 x 2] (S3: tbl_df/tbl/data.frame)
#>  $ cut: Ord.factor w/ 5 levels "Fair"<"Good"<..: 1 2 3 4 5
#>  $ n  : int [1:5] 1610 4906 12082 13791 21551
```

- Note that the object `cut_count` is just a data frame with two variables:
  - `cut` is a factor variable with five levels
  - `n` is an integer variable, showing the number of observations for each level of `cut`
- Next, use `ggplot() + geom_col` to plot the data from the object `cut_count`:

```
ggplot(data = cut_count, aes(x = cut, y=n)) +
  geom_col()
```

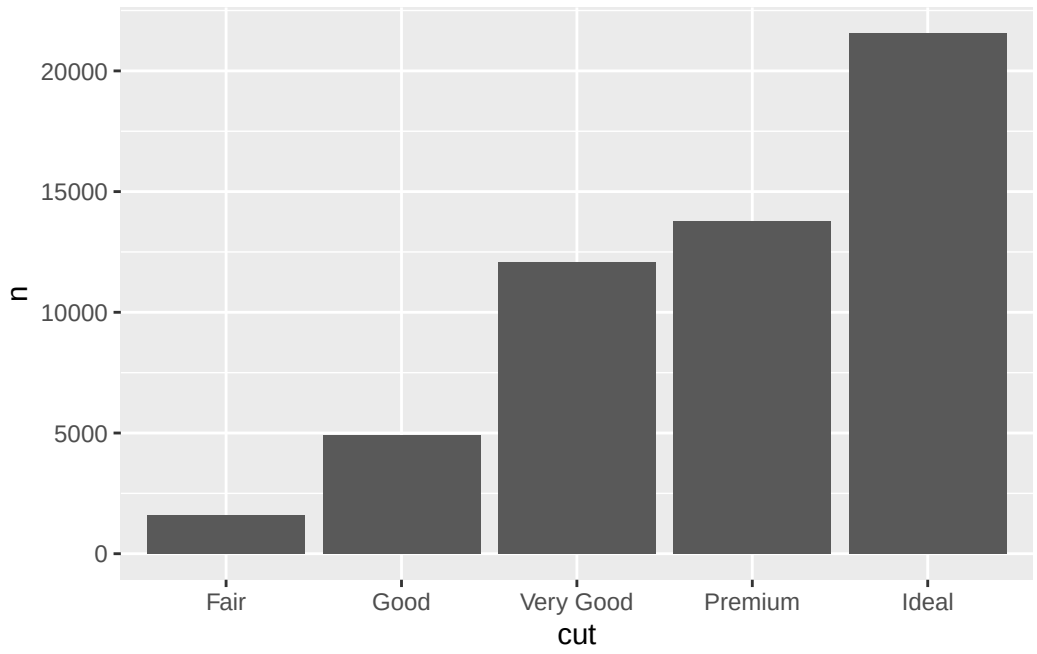


- note: if we didn't specify the aesthetic `y=n` we would get an error because `y` is a required aesthetic for `geom_col()`

```
ggplot(data = cut_count, aes(x = cut)) +
  geom_col()
```

- Alternatively, we can use pipes to create the plot without creating a separate `cut_count` object first:

```
#diamonds %>% count(cut) %>% str()
diamonds %>% count(cut) %>% str()
#> tibble [5 x 2] (S3: tbl_df/tbl/data.frame)
#>  $ cut: Ord.factor w/ 5 levels "Fair"<"Good"<..: 1 2 3 4 5
#>  $ n  : int [1:5] 1610 4906 12082 13791 21551
diamonds %>% count(cut) %>% ggplot(aes(x= cut, y=n)) +
  geom_col()
```



Student Task: Using the `els_v2` dataset, create a bar chart with the variable "ever attended postsecondary education" (`f2evratt`) as x-axis and number of students as y-axis

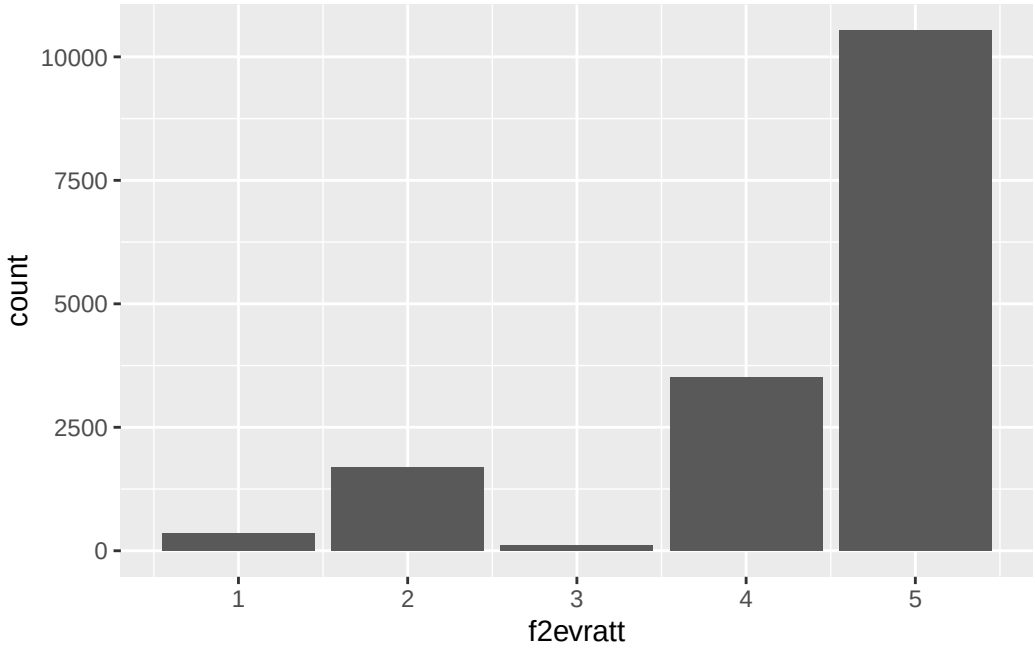
Solution

- Essentially, you are being asked to create a bar chart from the following frequency count:

```
els_v2 %>% count(f2evratt)
#> # A tibble: 5 x 2
#>   f2evratt     n
#>      <int> <int>
#> 1        1   359
#> 2        2  1691
#> 3        3   108
#> 4        4  3505
#> 5        5 10534
```

Method 1: Create bar chart using `geom_bar()`

```
ggplot(data = els_v2, aes(x = f2evratt)) +
  geom_bar()
```



- Additionally, we can use pipes to filter values of `f2evratt` before plotting:

```
els_v2 %>% filter(f2evratt %in% c('No','Yes')) %>% ggplot(aes(x = f2evratt)) +
  geom_bar()
```

Method 2: Create bar chart using `geom_col()`

```
els_v2 %>%
  # filter to remove missing values
  filter(f2evratt %in% c('No','Yes')) %>%
  # use count() to create summary statistics object
  count(f2evratt) %>%
  # plot summary statistic object
  ggplot(aes(x=f2evratt, y=n)) + geom_col()
```

Small multiples using faceting

Facets divide a plot into subplots based on the values of one or more discrete variables. They are most commonly used to create "small multiples"

Two functions to split your plots into facets:

- `facet_grid()`: Display subplots in grid format, where rows and columns are determined by the faceting variable(s)
  - `facet_grid()` is most useful when you have two discrete variables, and all combinations of the variables exist in the data
- `facet_wrap()`: Display all subplots side-by-side, but can be wrapped to fill multiple rows
  - `facet_wrap()` generally has better use of screen space, and you can specify the number of plots in each row or column

The `facet_grid()` and `facet_wrap()` functions:

```
?facet_grid

# SYNTAX AND DEFAULT VALUES
```

```
facet_grid(rows = NULL, cols = NULL, scales = "fixed",
  space = "fixed", shrink = TRUE, labeller = "label_value",
  as.table = TRUE, switch = NULL, drop = TRUE, margins = FALSE,
  facets = NULL)

?facet_wrap

# SYNTAX AND DEFAULT VALUES
facet_wrap(facets, nrow = NULL, ncol = NULL, scales = "fixed",
  shrink = TRUE, labeller = "label_value", as.table = TRUE,
  switch = NULL, drop = TRUE, dir = "h", strip.position = "top")
```

Specifying which variable(s) to facet your plot on:

- `facet_grid()`
  - Since `facet_grid()` arranges subplots in a grid format, we need to specify how we define the rows and columns
  - One way to do this is passing in the `rows` and `cols` arguments, which should be variables quoted by `vars()`
    - `facet_grid(rows = vars(<var_1>), cols = vars(<var_2>))`: facet into both rows and columns
    - `facet_grid(rows = vars(<var_1>))`: facet into rows only
    - `facet_grid(cols = vars(<var_1>))`: facet into columns only
  - Alternatively, we can pass in a *formula*, which has the syntax `<row_var> ~ <col_var>`
    - `facet_grid(<var_1> ~ <var_2>)`: facet into both rows and columns
    - `facet_grid(<var_1> ~ .)`: facet into rows only
    - `facet_grid(. ~ <var_1>)`: facet into columns only
- `facet_wrap()`
  - `facet_wrap()` also accepts a *formula* for its `facets` argument
    - `facet_wrap(~ <var_1>)`: facet by one variable
    - `facet_wrap(<var_1> ~ <var_2>)`: facet on the combination of two variables

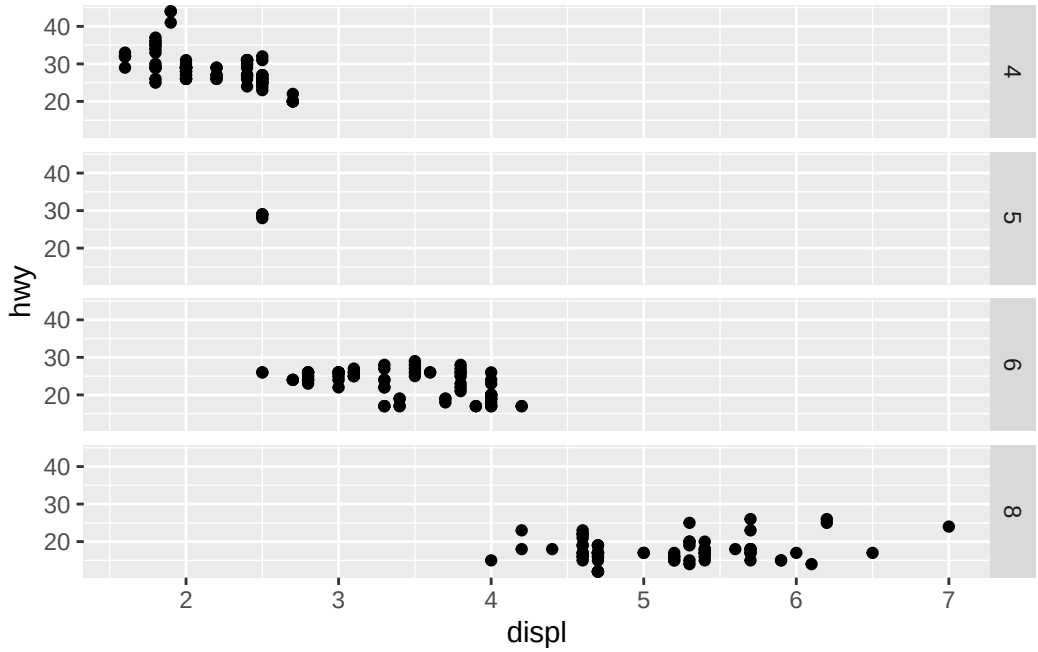
Faceting by one variable

Example: Scatterplot of the relationship between engine displacement (`displ`) and highway miles-per-gallon (`hwy`), faceted by number of cylinders (`cyl`), from the `mpg` dataset

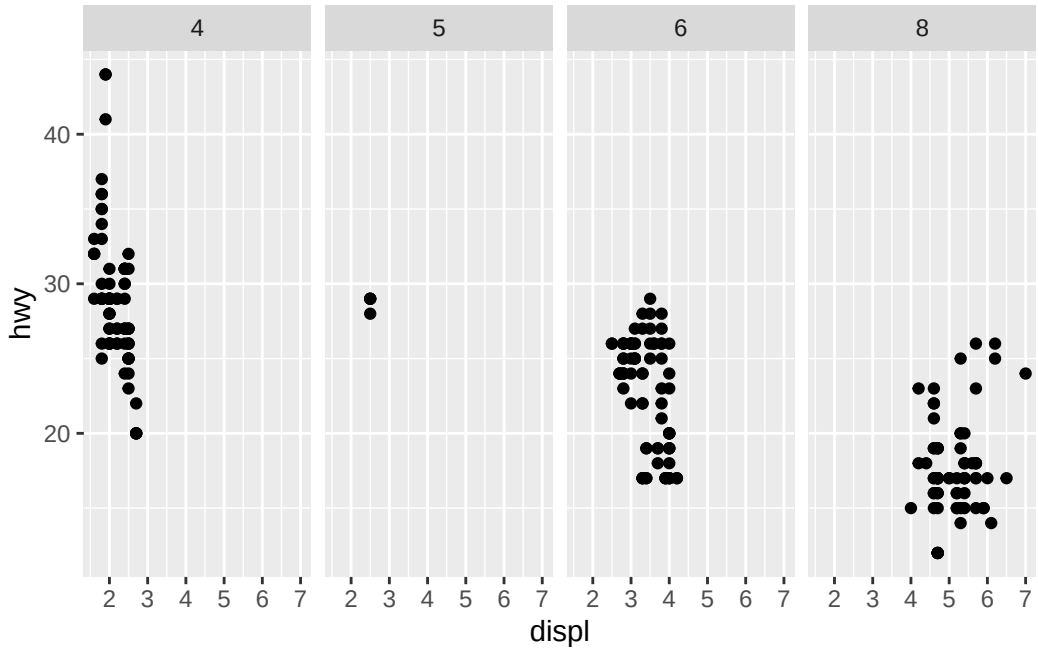
Method 1: Faceting using `facet_grid()`

- For one variable, you can choose to facet into rows or columns:

```
# Facet into rows
ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  facet_grid(rows = vars(cyl))
```

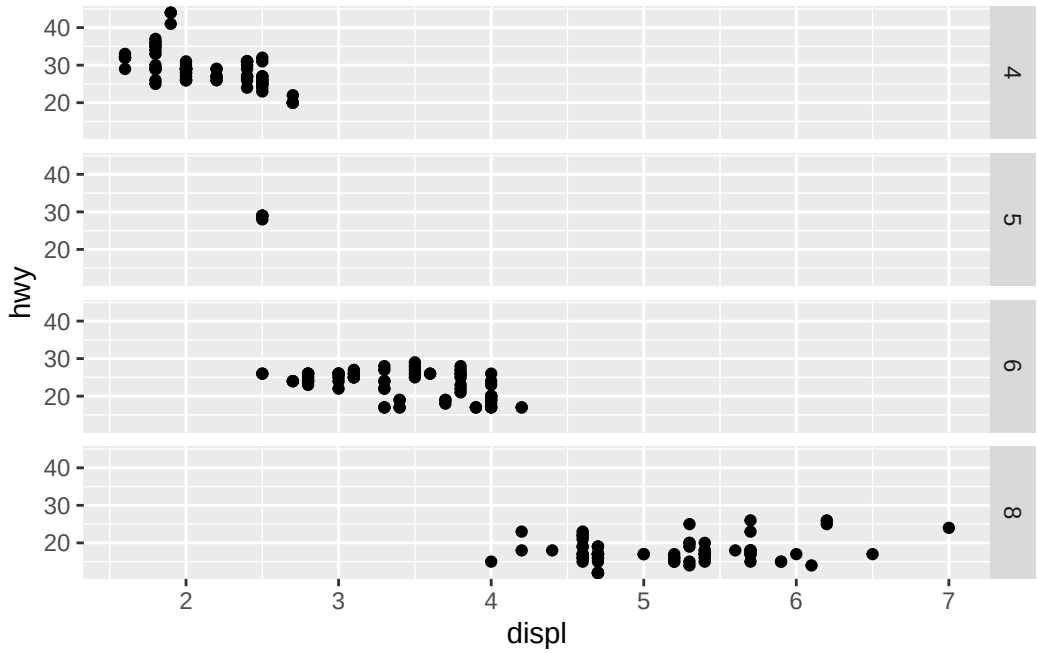



```
# Facet into columns
ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  facet_grid(cols = vars(cyl))
```

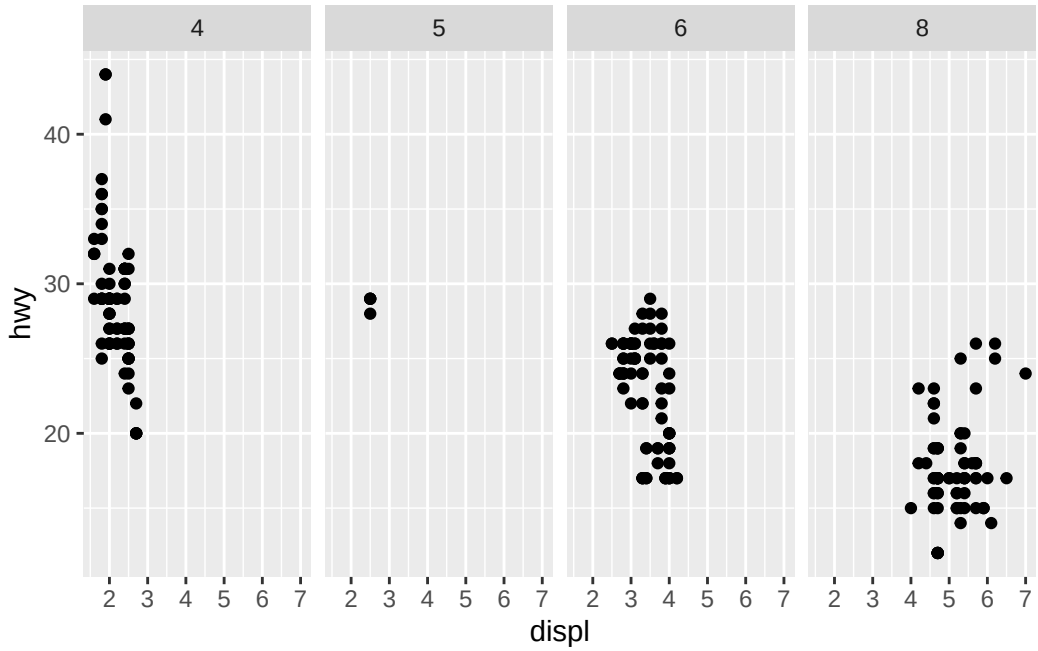


- Alternatively, we could specify the input as a *formula* to get the same results:

```
# Facet into rows
ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  facet_grid(cyl ~ .)
```



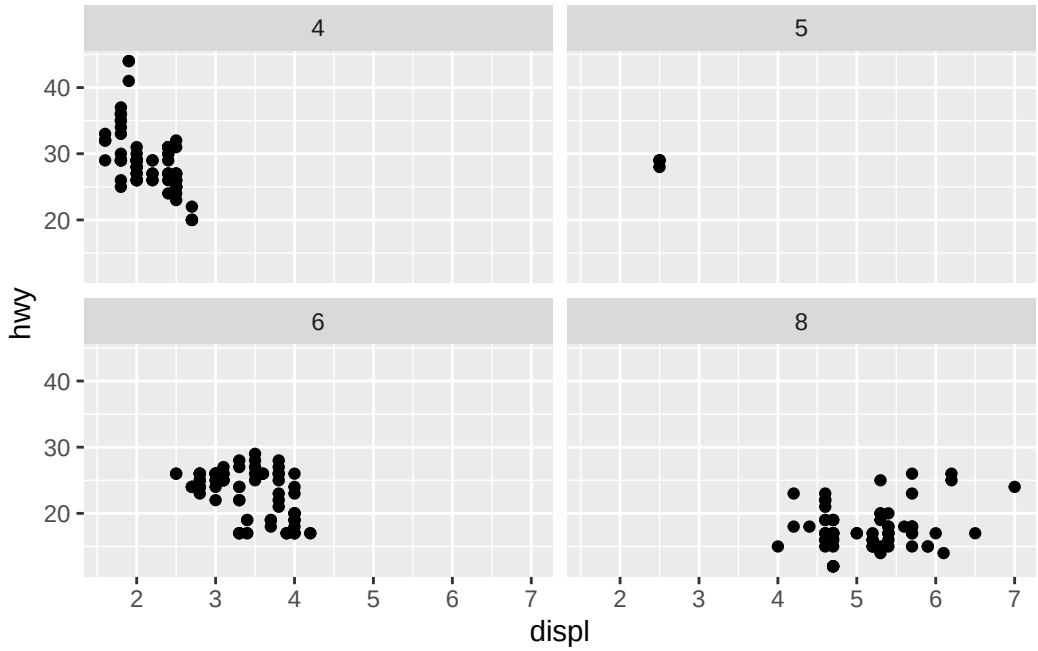

```
# Facet into columns
ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  facet_grid(. ~ cyl)
```



Method 2: Faceting using `facet_wrap()`

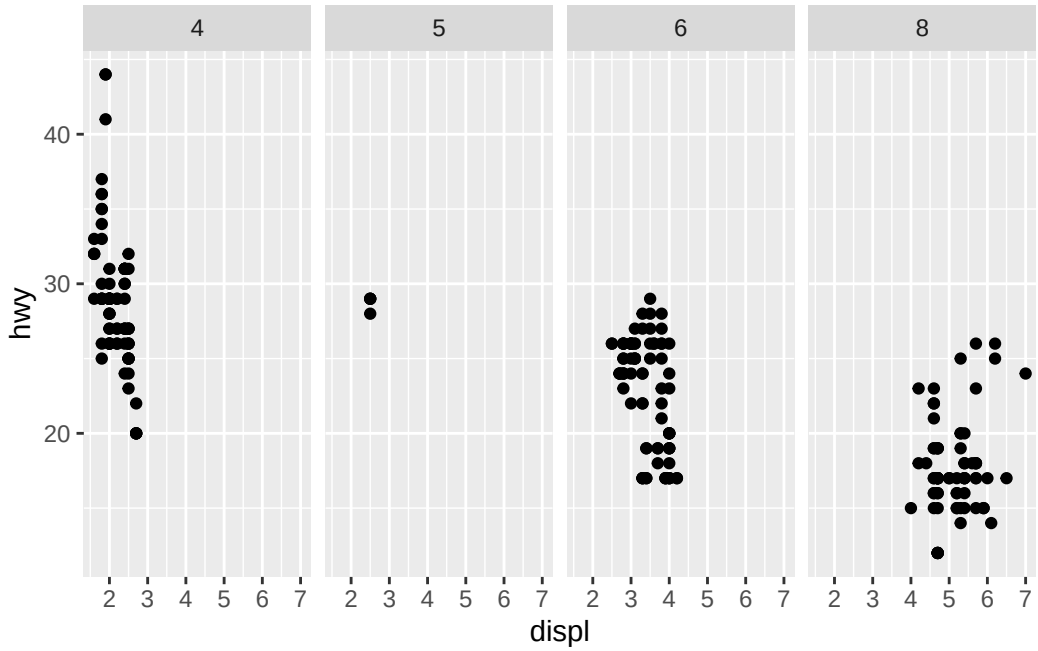
- Unlike `facet_grid()`, `facet_wrap()` is not restricted to either rows or columns:

```
ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  facet_wrap(~ cyl)
```

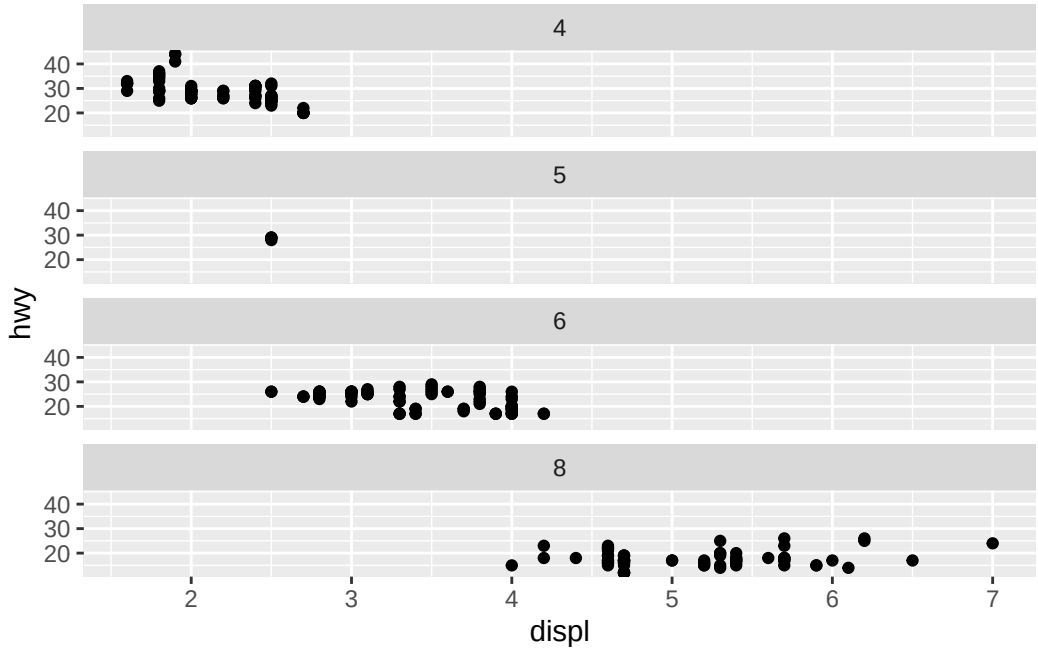


- But we are free to set the number of rows or columns if we wanted:

```
ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  facet_wrap(~ cyl, nrow = 1)
```




```
ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  facet_wrap(~ cyl, ncol = 1)
```



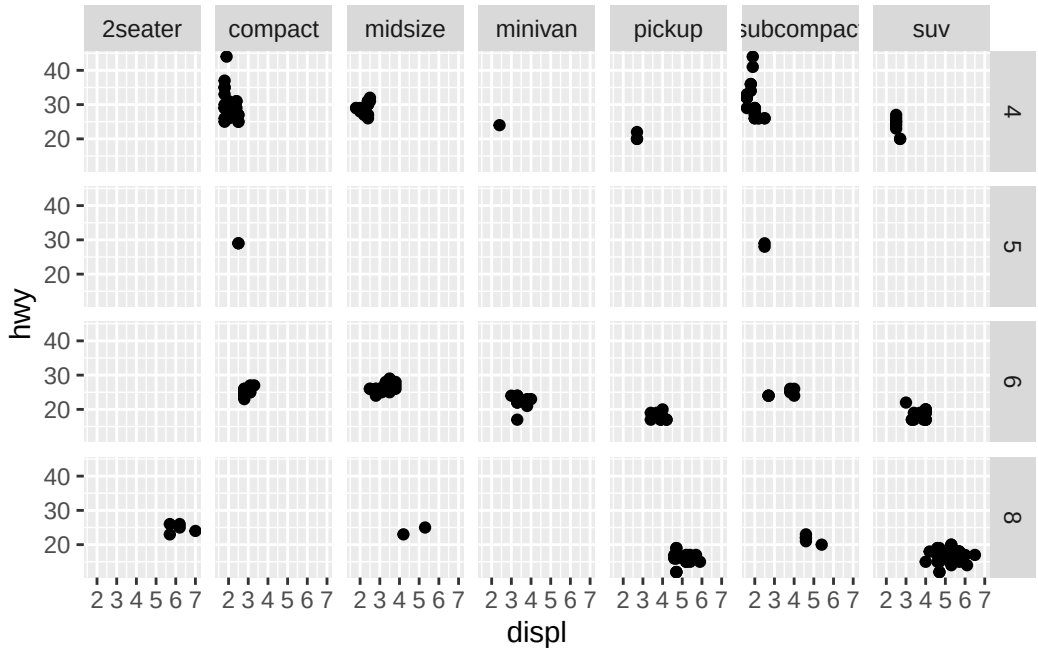
Faceting by two variables

Example: Scatterplot of the relationship between engine displacement (`displ`) and highway miles-per-gallon (`hwy`), faceted by number of cylinders (`cyl`) and type of car (`class`), from the `mpg` dataset

Method 1: Faceting using `facet_grid()`

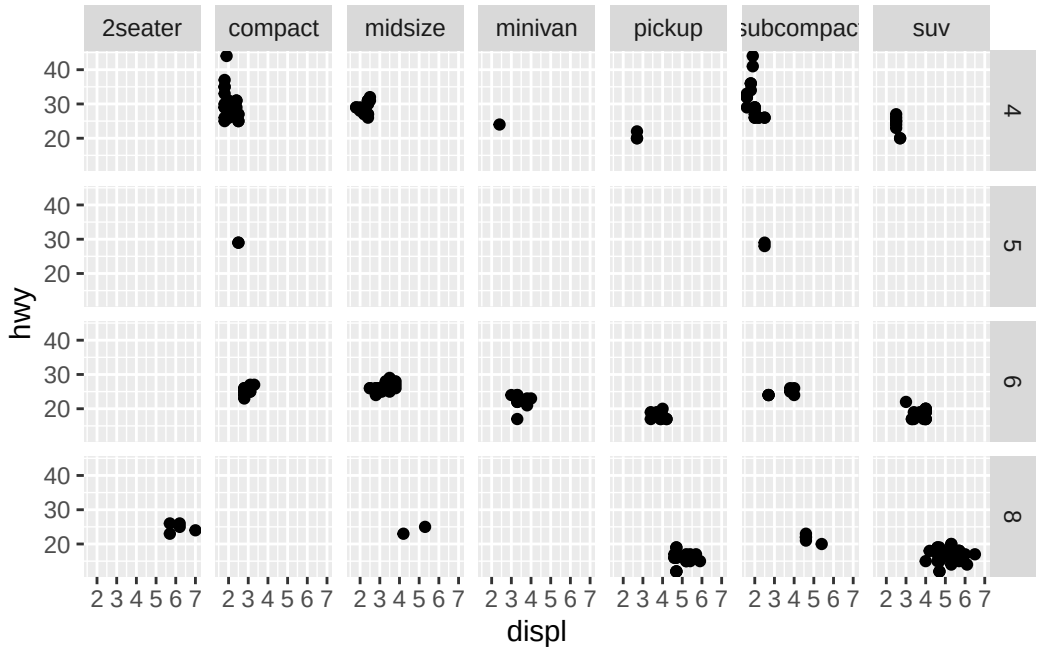
- For example, we can make the rows based on `cyl` and the columns based on `class`:

```
ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  facet_grid(rows = vars(cyl), cols = vars(class))
```



- Alternatively, we could specify the input as a *formula* to get the same results:

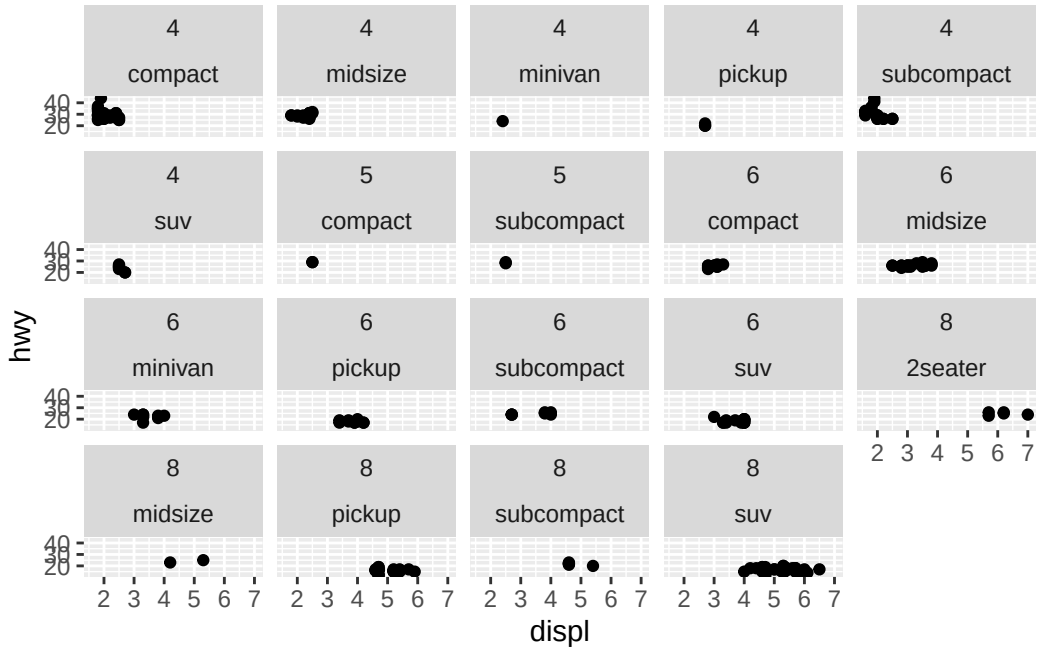
```
ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  facet_grid(cyl ~ class)
```



Method 2: Faceting using `facet_wrap()`

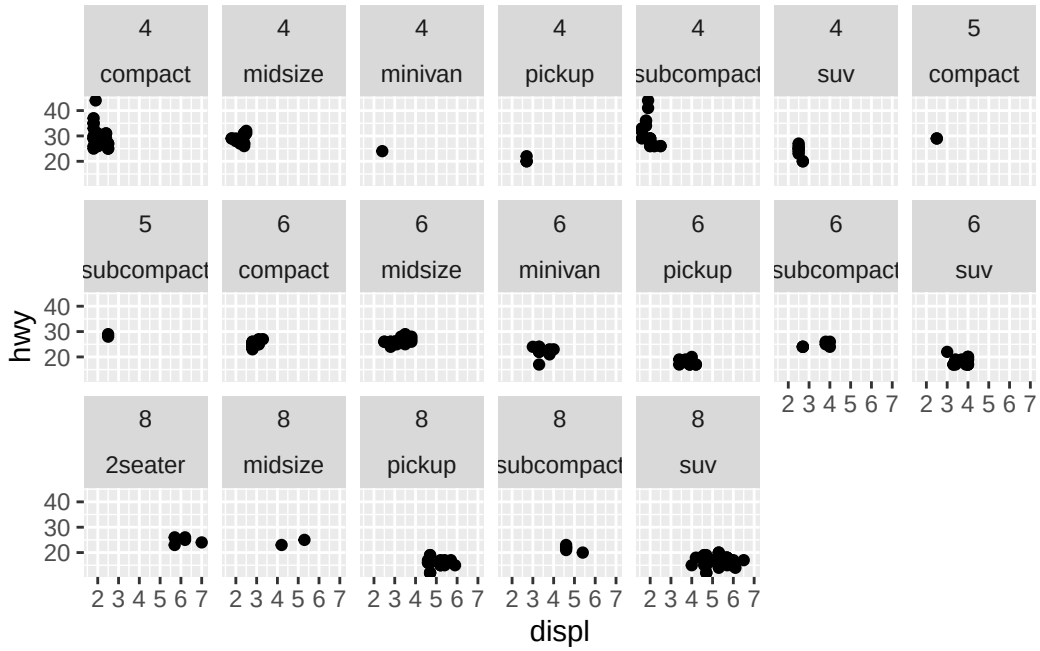
- Since `facet_wrap()` is not defined by rows and columns, it omits any subplots that do not display any data:

```
ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  facet_wrap(cyl ~ class)
```

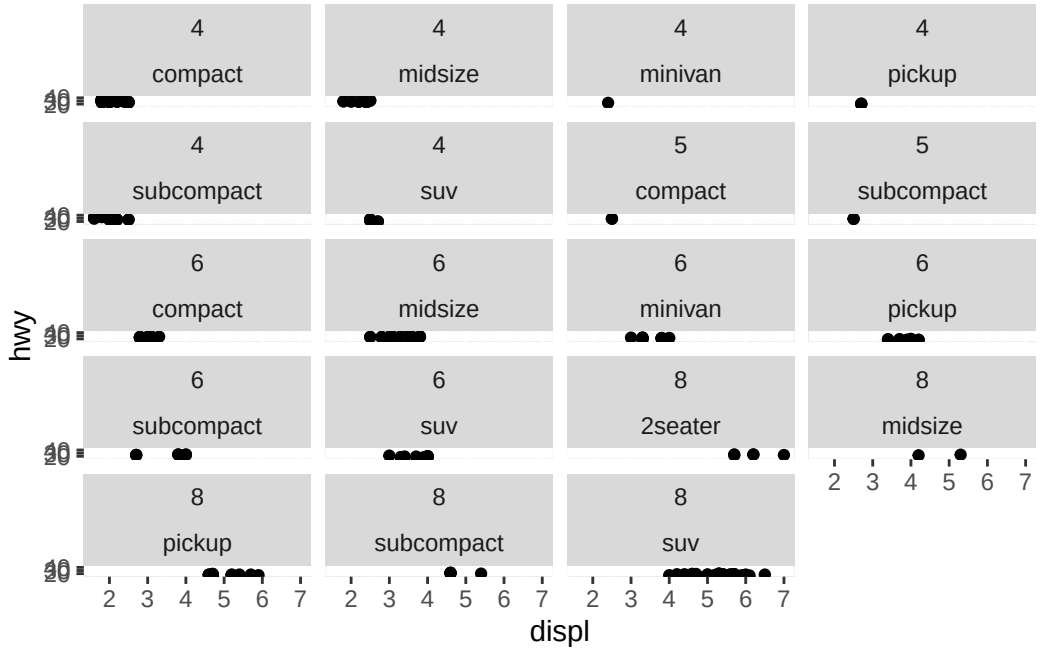


- We are also free to choose the number of rows or columns to display:

```
ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  facet_wrap(cyl ~ class, nrow = 3)
```



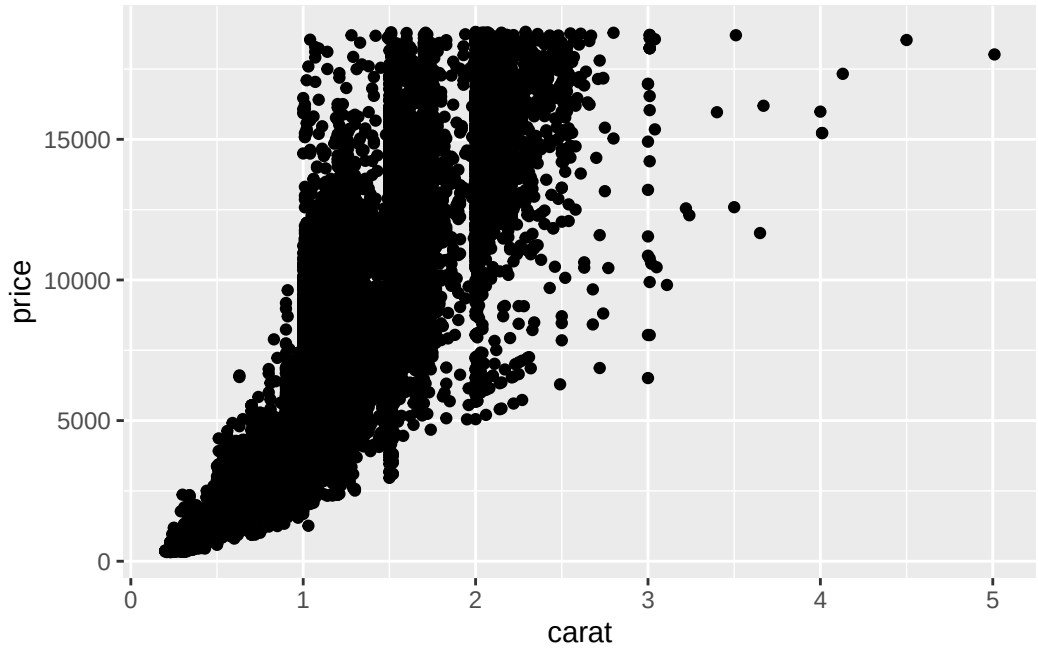

```
ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  facet_wrap(cyl ~ class, ncol = 4)
```



Customization

There are many ways to customize the display of our plot. For this section, we will build upon this scatterplot we saw earlier:

```
diamonds %>%
  ggplot(mapping = aes(x = carat, y = price)) +
  geom_point()
```



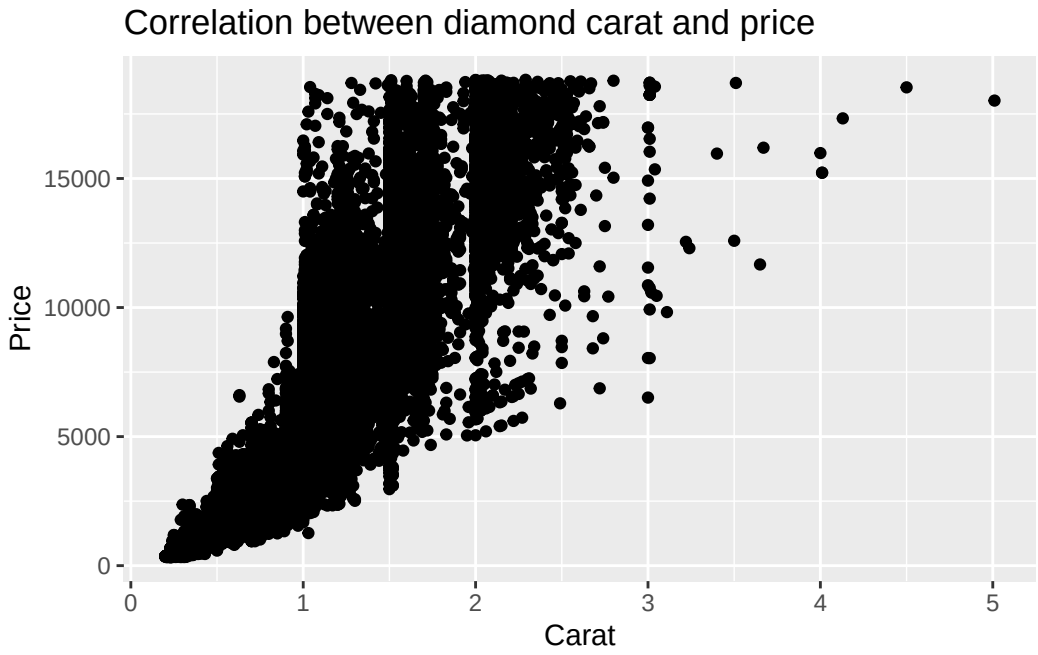
Labels

Functions to add title and axis labels:

- `ggtitle()`: Add title of graph
- `xlab()`: Add x-axis label
- `ylab()`: Add y-axis label

Example: Adding title and axis labels

```
diamonds %>%
  ggplot(mapping = aes(x = carat, y = price)) +
  geom_point() +
  ggtitle('Correlation between diamond carat and price') +
  xlab('Carat') + ylab('Price')
```



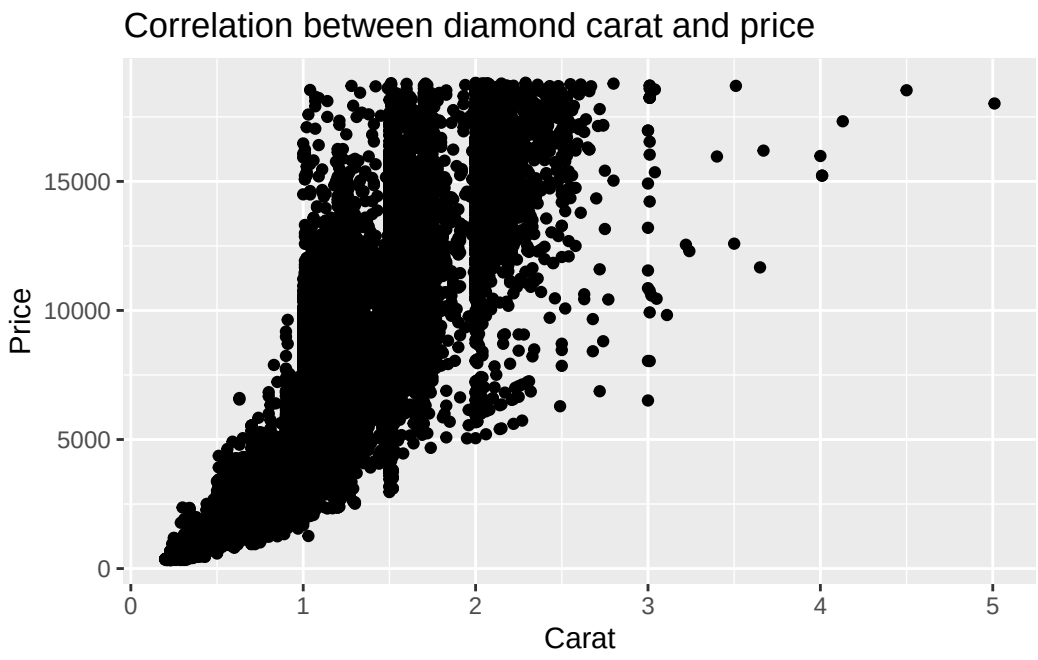
Alternatively, the `labs()` function can be used to add all titles and labels:

```
?labs

# SYNTAX AND DEFAULT VALUES
labs(
  ...,
  title = waiver(),
  subtitle = waiver(),
  caption = waiver(),
  tag = waiver()
)
```

Note: A **waiver** is a "flag" object, similar to `NULL`, that indicates the calling function should just use the default value. It is used in certain functions to distinguish between displaying nothing (`NULL`) and displaying a default value calculated elsewhere (`waiver()`).

Example: Adding title and axis labels using `labs()`

```
diamonds %>%
  ggplot(mapping = aes(x = carat, y = price)) +
  geom_point() +
  labs(title = 'Correlation between diamond carat and price', x = 'Carat', y = 'Price')
```



Scales

The **`scale_x_continuous()`** function is used to control the display of the x-axis when x is a continuous variable.

Similarly, the **`scale_y_continuous()`** function is to control the display of the y-axis when y is a continuous variable.

```
?scale_x_continuous
?scale_y_continuous

# SYNTAX AND DEFAULT VALUES
scale_x_continuous(
  name = waiver(),
  breaks = waiver(),
```

```
  minor_breaks = waiver(),
  n.breaks = NULL,
  labels = waiver(),
  limits = NULL,
  expand = waiver(),
  oob = censor,
  na.value = NA_real_,
  trans = "identity",
  guide = waiver(),
  position = "bottom",
  sec.axis = waiver()
)

scale_y_continuous(
  name = waiver(),
  breaks = waiver(),
  minor_breaks = waiver(),
  n.breaks = NULL,
  labels = waiver(),
  limits = NULL,
  expand = waiver(),
  oob = censor,
  na.value = NA_real_,
  trans = "identity",
  guide = waiver(),
  position = "left",
  sec.axis = waiver()
)
```

- Description (from help file)
  - "`scale_x_continuous()` and `scale_y_continuous()` are the default scales for continuous x and y aesthetics."
- Arguments
  - `name`: The name of the scale. Used as the axis or legend title.
  - `labels`: Custom labelling of the scales (i.e., ticks)
  - `limits`: Limits of the scale (i.e., min/max values)
  - `position`: The position of the axis. (`'left'` or `'right'` for y axes, `'top'` or `'bottom'` for x axes)

To force "decimal display" of numbers (rather than scientific notation), we can use the `label_number()` function like this:

- `scale_y_continuous(labels = label_number(...))`

The `label_number()` function:

```
?label_number

# SYNTAX AND DEFAULT VALUES
label_number(
  accuracy = NULL,
  scale = 1,
  prefix = "",
  suffix = "",
  big.mark = " ",
  decimal.mark = ".",
  trim = TRUE,
  ...
)
```

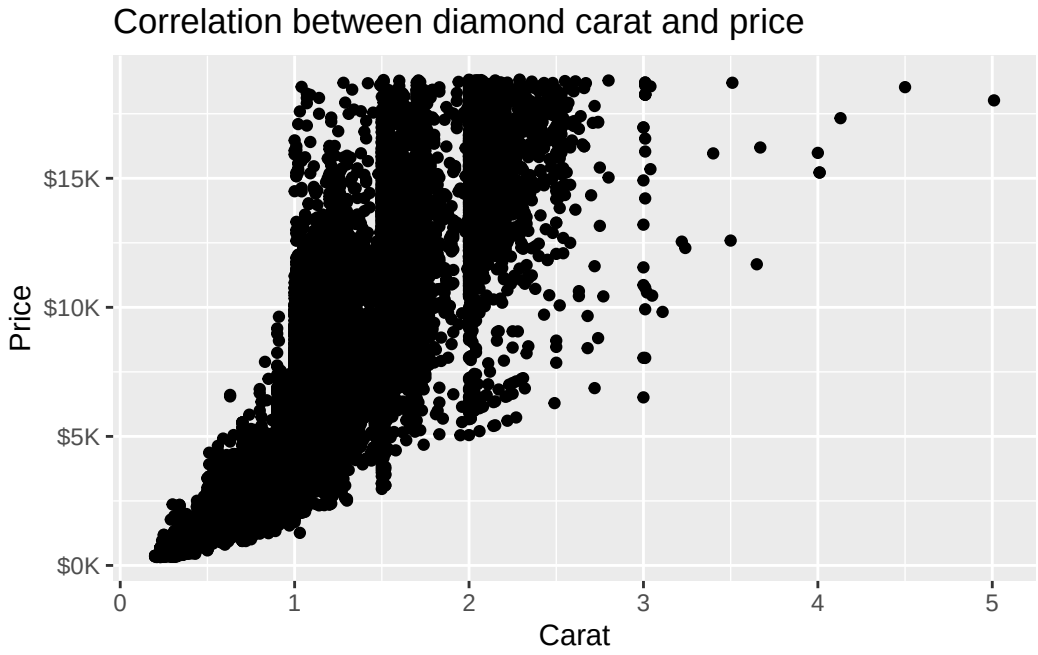
- Description (from help file)
  - "Use `label_number()` force decimal display of numbers (i.e. don't use scientific notation)"
- Arguments
  - `accuracy`: A number to round to (e.g. use 0.01 to show 2 decimal places of precision)
  - `scale`: A scaling factor (e.g., x will be multiplied by scale before formatting)
  - `prefix`: Symbols to display before value
  - `suffix`: Symbols to display after value

Example: Formatting numbers on the y-axis

We can use `scale_y_continuous()`, in conjunction with `label_number()` from the `scales` package, to format the numbers on the y-axis:

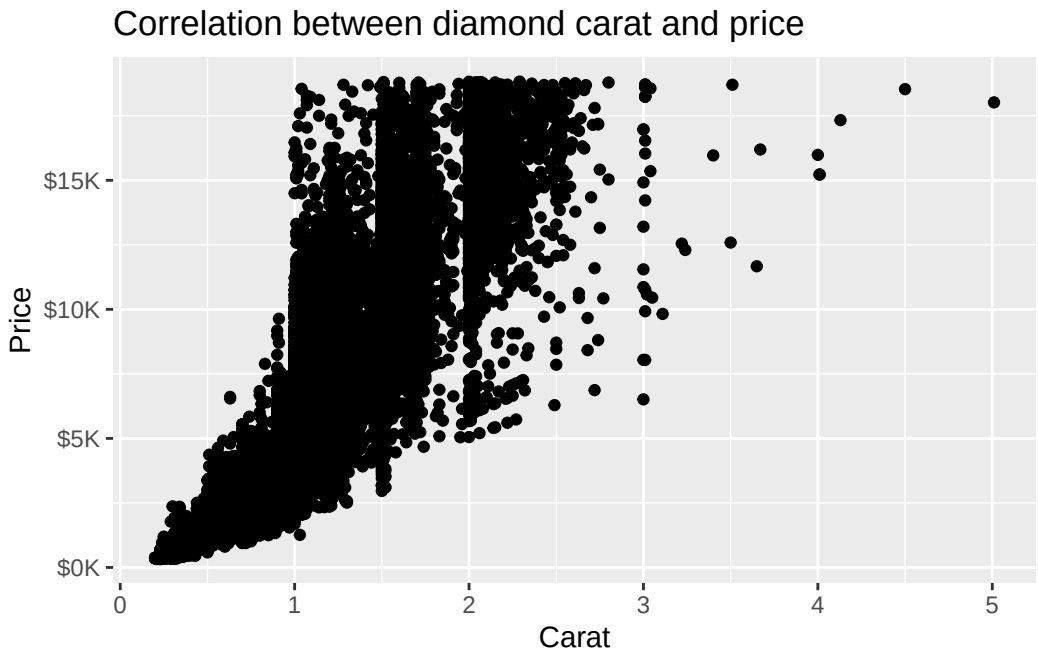
- Use `prefix` to add `$` before the number
- Use `suffix` to add `K` after the number
- Use `scale` of `1e-3` to divide number by `1000`
- Use `accuracy` of `1` to round number to the ones digit

```
diamonds %>%
  ggplot(mapping = aes(x = carat, y = price)) +
  geom_point() +
  ggtitle('Correlation between diamond carat and price') +
  xlab('Carat') + ylab('Price') +
  scale_y_continuous(labels = label_number(prefix = '$', suffix = 'K', scale = 1e-3, accuracy
```



- Alternatively, we could specify scale like this: `scale = .001`:

```
diamonds %>%
  ggplot(mapping = aes(x = carat, y = price)) +
  geom_point() +
  ggtitle('Correlation between diamond carat and price') +
  xlab('Carat') + ylab('Price') +
  scale_y_continuous(labels = label_number(prefix = '$', suffix = 'K', scale = .001, accuracy
```



Colors

Below, we briefly overview some core concepts when using colors. This will be helpful when creating graphs.

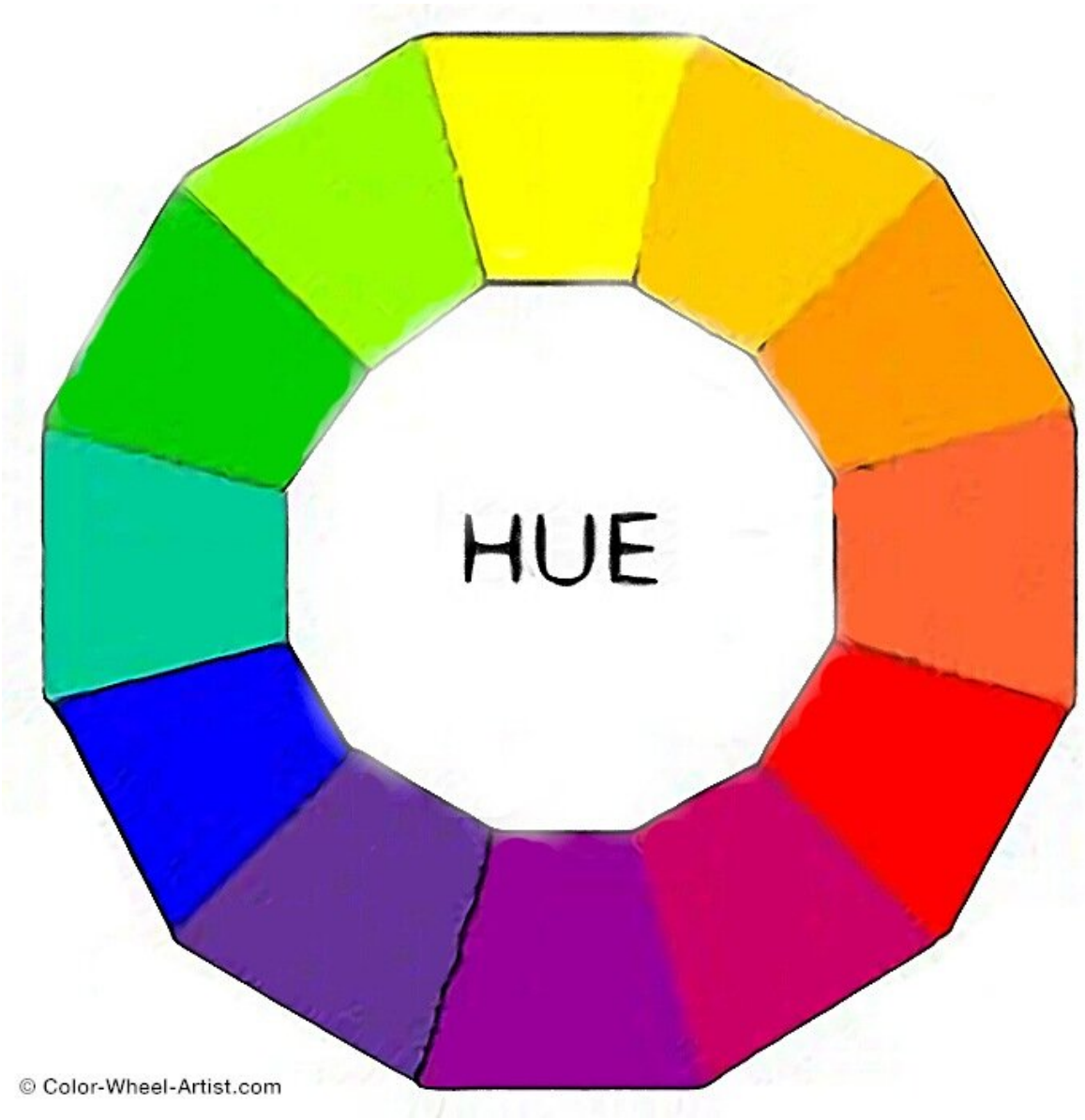
HCL Color Space

The HCL color space is based on how the human perception works. "When using HCL, you can directly control the color (hue), the colorness (chroma) and the luminance (brightness)." (Why HCL from hcl wizard)

Hue

Hue refers to a specific color we see from one of the six Primary and Secondary colors in the Color Family– Yellow, Orange, Red, Violent, Blue, or Green. This means that Black, Gray, and White are not Hues.

- Example: Burgundy has a dominant hue of red.



Color

Color and Hue are used interchangeably, however, Color is used to describe hues, tints, shades, and tones.

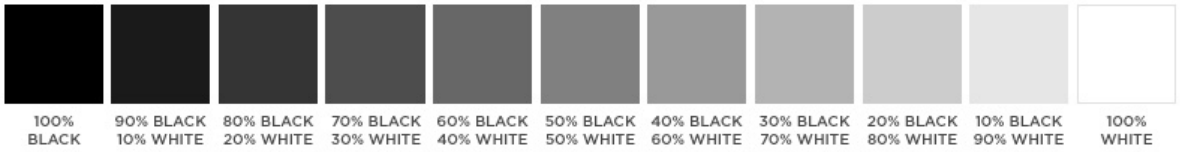
- Example: Black, Gray, and White are colors.

Value

The value of a color is determined by how close the color is to Black or White. The closer a color is to Black, the darker the value and the closer a color is to White, the lighter the

value.

- Example: It is helpful to use a grayscale to determine the value of a color.



- "All colors fall somewhere on the value scale between black and white" (Hue, Value, Chroma Explained from sensationalcolor.com)

Chroma

- Chroma expresses the "purity" of a color. "Mixing a pure hue with black, white, gray, or any other color reduces its purity and lowers the chroma." (Hue, Value, Chroma Explained from sensationalcolor.com)
- Example: The image below displays the changes of a "pure" hue when adding the color white or black to it. As the red squares on the far left column are blended with white, they become lighter in value and lower in chroma and as the squares are blended with black, they become darker in value and lower in chroma.