Lec 2 R Functions And The Grammar Of Visualization

Xinan Wang

2023-01-24

Functions

r functions are called like this: function_name(arg1 = val1, arg2 = val2, ...)

```
sum(1,2)
```

```
## [1] 3
```

Create Functions

variable <- function(arg1, arg2, ...) { your expression / algorithm }

```
hello <- function() {
  print("Hello World!")
}

hello()
```

```
## [1] "Hello World!"
```

Name Masking

```
f <- function() {
  x <- 1
  y <- 2
  c(x, y)
}
f()
```

```
## [1] 1 2
```

```
rm(f)
```

If a name isn't defined inside a function, R will look one level up

```r
x <- 2
g <- function() {
  y <- 1
  c(x, y)
}
g()
```

```
## [1] 2 1
```

```r
rm(g)
```

Functions v.s. Variables

```r
n <- function(x)
  x / 2
o <- function() {
  n <- 10
  n(n)
}
o()
```

```
## [1] 5
```

```r
rm(n, o)
```

```r
f <- function(x) {
  f <- function(x) {
    f<- function(x) {
      x ^ 2
    }
    f(x) + 1
  }
  f(x) * 2
}
f(10)
```

```
## [1] 202
```

Every operation is a function call

```r
x <- 1
y <- 2
x+y
```

```
## [1] 3
```

```
`+`(x,y)
```

```
## [1] 3
```

```
for (i in 1:2) print(i)
```

```
## [1] 1
## [1] 2
```

```
`for`(i, 1:2, print(i))
```

```
## [1] 1
## [1] 2
```

Anonymous Functions

function(x)3()

(function(x)3)()

Return Values

```
double.num <- function(x) {
  x * 3
  print("hello")
  x * 2
}
double.num(5)
```

```
## [1] "hello"
```

```
## [1] 10
```

```
double.num <- function(x) {
  return(x * 3)
  print("hello")
  return(3)
}
double.num(5)
```

```
## [1] 15
```

Functions with if and else

```
if.one <- function(x) {
  if(x == 1) {
    print("True")
  }
  else {
    print("False")
  }
}
if.one(1)
```

```
## [1] "True"
```

```
if.one(2)
```

```
## [1] "False"
```

```
if.one <- function(x){
  ifelse(x==1, "TRUE", "FALSE")
}
if.one(1)
```

```
## [1] "TRUE"
```

```
if.one(2)
```

```
## [1] "FALSE"
```

Switch

```
multipleCases <- function(x) {
  switch(x,
         a = "first",
         b = "second",
         z = "last",
         c = "third",
         d = "other")
}
multipleCases("a")
```

```
## [1] "first"
```

...

```
f <- function(a,b,c) {
  data.frame(a,b,c)
}
f(a = 1, b = 2, c = 3)
```

```
##   a b c
## 1 1 2 3
```

```
f <- function(...) {
  data.frame(...)
}
f(a = 1, b = 2, c = 3, d = 5, e = 7)
```

```
##   a b c d e
## 1 1 2 3 5 7
```

For Loops

```
for (i in 1:3) {
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
```

```
fruit <- c("apple", "banana", "pomegranate")

fruitlength <- rep(NA, length(fruit))
fruitlength
```

```
## [1] NA NA NA
```

```
names(fruitlength) <- fruit
fruitlength
```

```
##       apple      banana pomegranate
##          NA          NA          NA
```

```
for (i in fruit) {
  fruitlength[i] <- nchar(i)
}
fruitlength
```

```
##       apple      banana pomegranate
##           5           6          11
```

Apply

apply

Must be used on a matrix, meaning all the elements must be of the same type whether they are character, numeric or logical.

```r
theMatrix <- matrix(1:9, nrow = 3)
theMatrix
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```r
# sum the row
apply(theMatrix, 1, sum)
```

```
## [1] 12 15 18
```

```r
apply(theMatrix, 2, sum)
```

```
## [1]  6 15 24
```

```r
rowSums(theMatrix)
```

```
## [1] 12 15 18
```

```r
colSums(theMatrix)
```

```
## [1]  6 15 24
```

lapply, sapply

Basic grammar

lapply(x, FUN, . . .) sapply(x, FUN, . . .)

lapply works by applying a function to each element of a list and returning the results as a list

```r
theList <- list(A = 1:3, B = 1:5, C = -1:1, D = 2)
lapply(theList, sum)
```

```
## $A
## [1] 6
##
## $B
## [1] 15
##
## $C
## [1] 0
##
## $D
## [1] 2
```

sapply is a user-friendly version and wrapper of lapply by default returning a vector

```r
theList <- list(A = 1:3, B = 1:5, C = -1:1, D = 2)
sapply(theList, sum)
```

```
##  A  B  C  D
##  6 15  0  2
```

mapply

Mapply: applies a function to each elements of multiple lists

```r
firstList <-
  list(A = matrix(1:16, 4),
       B = matrix(1:16, 2),
       C = data.frame(1:5))
secondList <-
  list(A = matrix(1:16, 4),
       B = matrix(1:16, 8),
       C = data.frame(15:1))
mapply(identical, firstList, secondList)
```

```
##     A     B     C
##  TRUE FALSE FALSE
```

```r
simpleFunc <- function(x, y) {
  nrow(x) + nrow(y)
}
mapply(simpleFunc, firstList, secondList)
```

```
##  A  B  C
##  8 10 20
```

Graphics

Basic Plots

- Bar chart (categorical variable vs continuous variable)
- Scatter plot (continuous variable vs continuous variable)
- Line chart (continuous variable vs continuous variable)
- Box plot (categorical variable vs continuous variable)
- Histogram (continuous variable vs continuous variable)

Title of graph, scale, coordinate system, context, visual cues

Components of the plots: • Layers: Dataset; Aesthetic mapping(color, shape, size, etc) Statistical transformation Geometric object(line, bar, dots, etc) Position adjustment • Scale(optional) • Coordinate system • Faceting(optional) • Defaults

ggplot

ggplot(data = ,

```
  mapping = aes(<Mappings>) +

    layer(geom = <GEOM>,

          stat = <STAT>,

          position = <POSITION>) +

    <SCALE_FUNCTION>() +

    <COORDINATE_FUNCTION>() +

    <FACET_FUNCTION>()
```

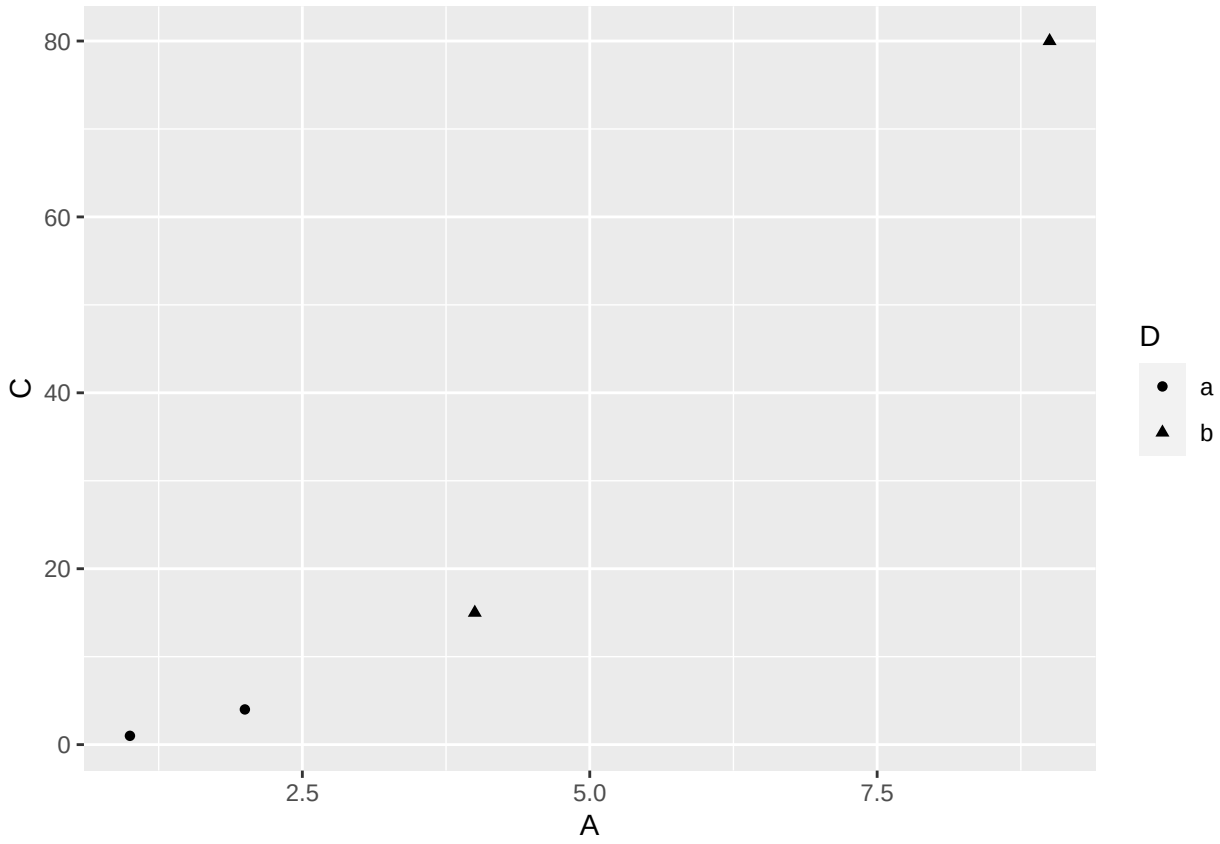
```
df <- data.frame(
  A = c(2, 1, 4, 9),
  B = c(3, 2, 5, 10),
  C = c(4, 1, 15, 80),
  D = c("a", "a", "b", "b")
)
df
```

```
##   A  B  C D
## 1 2  3  4 a
## 2 1  2  1 a
## 3 4  5 15 b
## 4 9 10 80 b
```

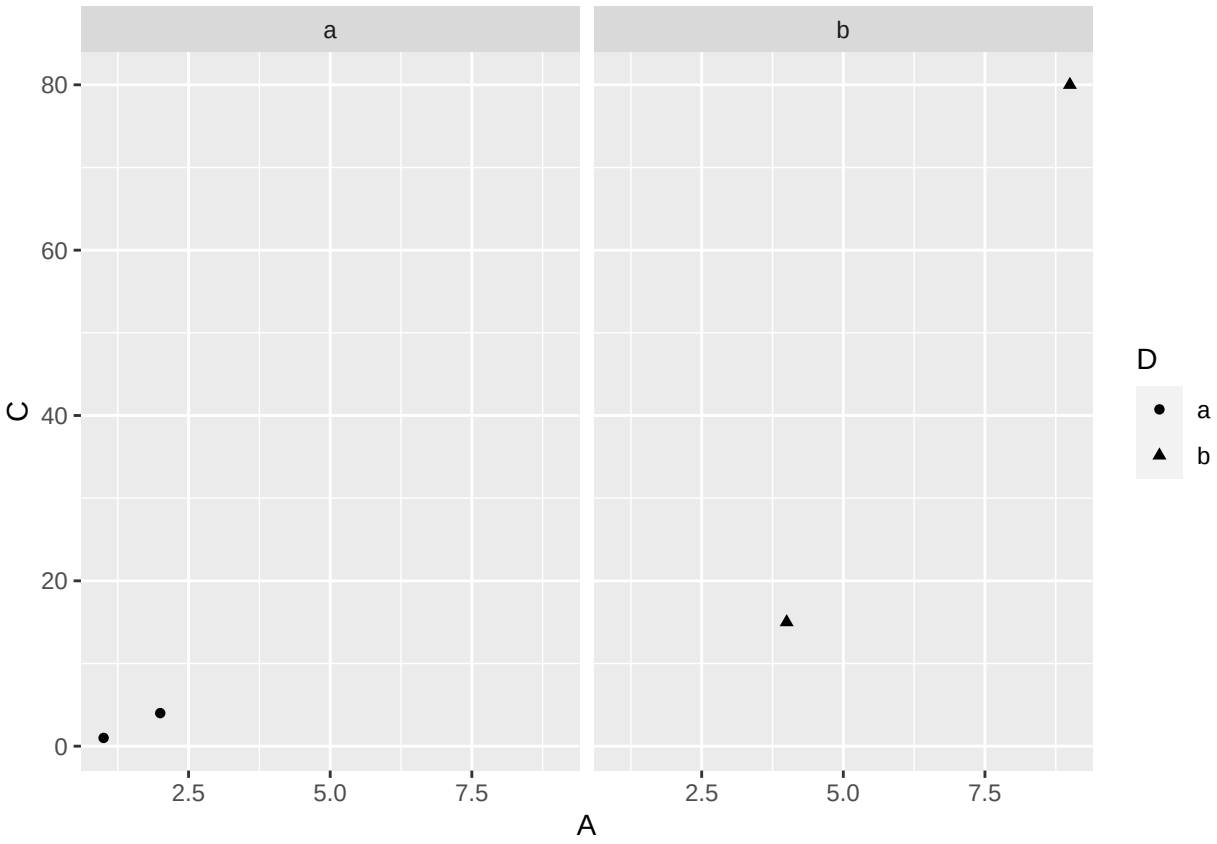
```
library(ggplot2)

df <- data.frame(
  A = c(2, 1, 4, 9),
  B = c(3, 2, 5, 10),
  C = c(4, 1, 15, 80),
  D = c("a", "a", "b", "b")
)

ggplot(data = df,
       mapping = aes(x = A, y = C, shape = D)) +
  layer(geom = "point",
        stat = "identity",
        position = "identity") +
  scale_x_continuous() +
  scale_y_continuous() +
  coord_cartesian() +
  facet_null()
```




```
ggplot(data = df, mapping = aes(x = A, y = C, shape = D)) +
  geom_point() +
  facet_grid( ~ D)
```



Dataset : Fuel economy in cars

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.5.0
## v readr   2.1.3      v forcats 0.5.2
## v purrr   1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
head(mpg, 5)
```

```
## # A tibble: 5 x 11
##   manufacturer model displ  year   cyl trans      drv     cty   hwy fl    class
##   <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
## 1 audi         a4      1.8  1999     4 auto(l5)   f        18    29 p     compa~
## 2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     compa~
## 3 audi         a4      2    2008     4 manual(m6) f        20    31 p     compa~
## 4 audi         a4      2    2008     4 auto(av)   f        21    30 p     compa~
## 5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p     compa~
```

```r
# Interviewing data

mpg[,c("displ","hwy")]
```

```
## # A tibble: 234 x 2
##    displ   hwy
##    <dbl> <int>
##  1   1.8    29
##  2   1.8    29
##  3   2      31
##  4   2      30
##  5   2.8    26
##  6   2.8    26
##  7   3.1    27
##  8   1.8    26
##  9   1.8    25
## 10   2      28
## # ... with 224 more rows
```

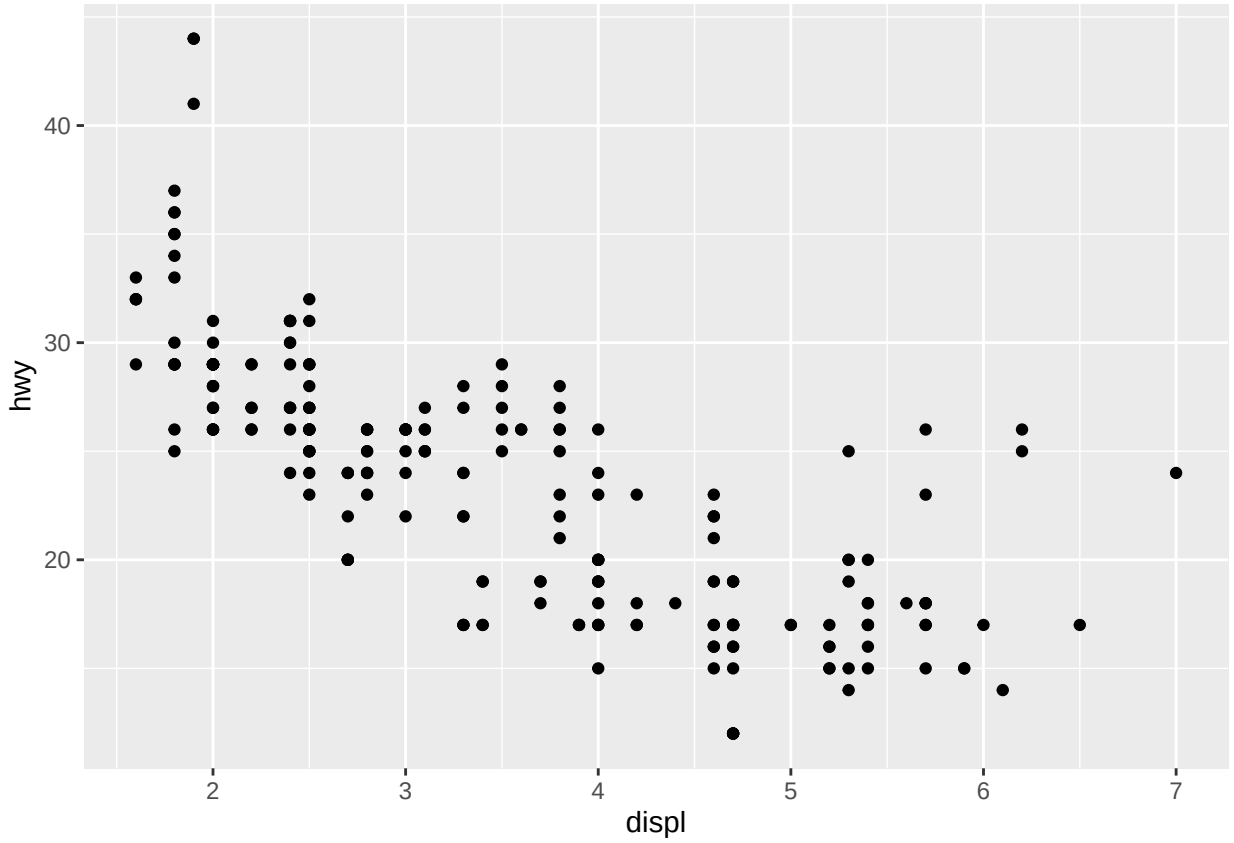
```r
# Creating base

ggplot(data = mpg)
```



```r
# Creating plot
```

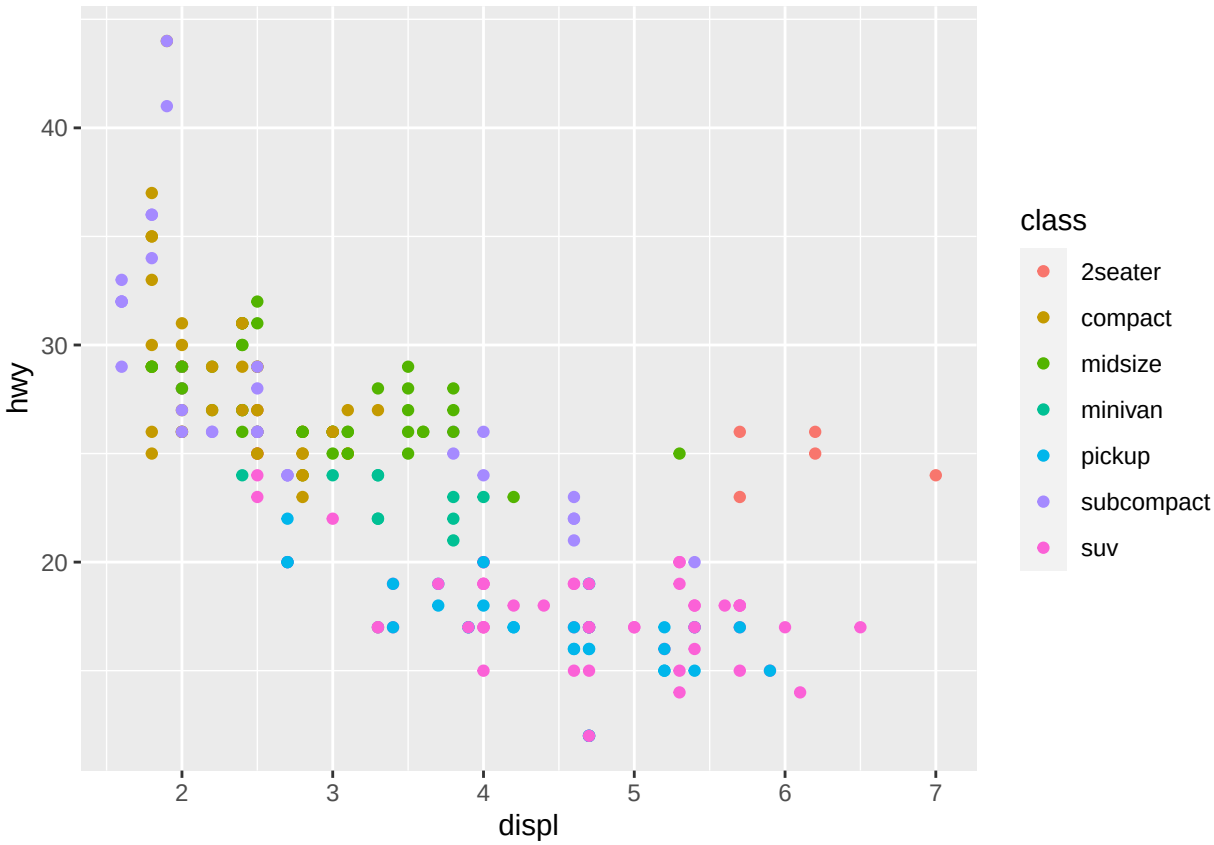
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```



Aesthetic Mappings

Aesthetic mappings: We can change levels of size, shape, color, fill, alpha etc. inside of aes()

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



Aesthetic mappings: outside of aes()

- The name of a color as character string
- The size of a point in mm
- The shape of a point as a number

geom_() functions

- geom_bar: bar chart
- geom_histogram: histogram
- geom_point: scatterplot
- geom_qq: quantile-quantile plot
- geom_boxplot: boxplot
- geom_line: line chart

```
geom_line
```

```
## function (mapping = NULL, data = NULL, stat = "identity", position = "identity",
##     na.rm = FALSE, orientation = NA, show.legend = NA, inherit.aes = TRUE,
##     ...)
## {
##     layer(data = data, mapping = mapping, stat = stat, geom = GeomLine,
##         position = position, show.legend = show.legend, inherit.aes = inherit.aes,
```

```
##          params = list2(na.rm = na.rm, orientation = orientation,
##              ...))
## }
## <bytecode: 0x125d5cf70>
## <environment: namespace:ggplot2>
```

- Use geom_ function to make variables visible on the screen
- Use stat_ function and define geom shape as an argument inside geom_.
- Or use geom_ and define statistical transformation as an argument inside stat_.

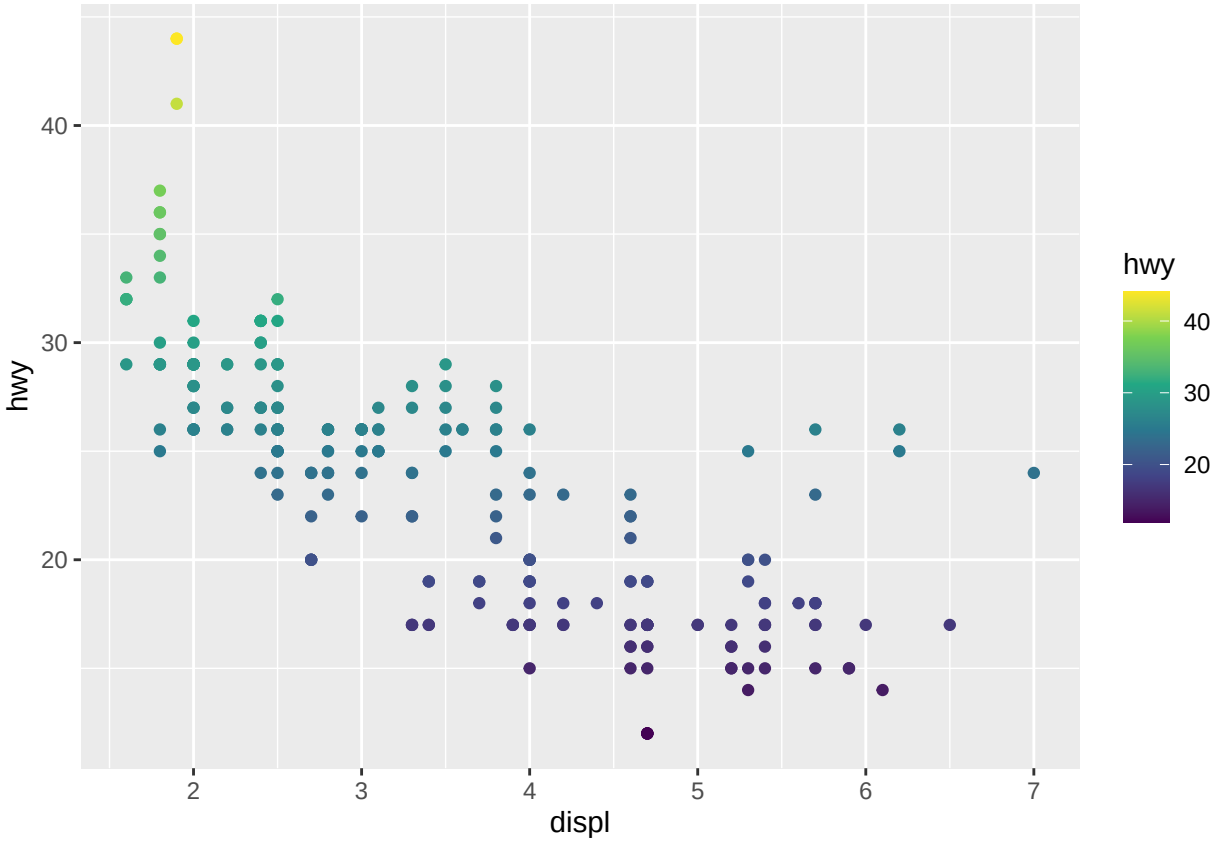
Common geom with position adjustments:

geom_point, geom_jitter

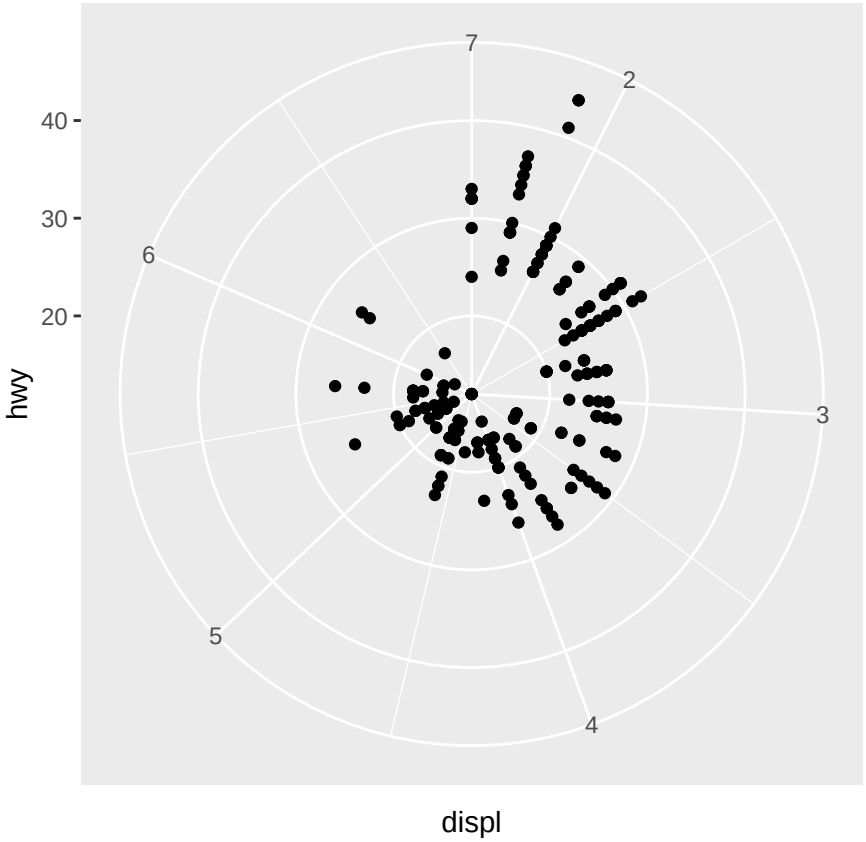
Scale syntax

labs() xlab() ylab() ggtitle() lim() xlim() ylim() scale_colour_brewer() scale_colour_continuous()

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = hwy)) +
  scale_colour_continuous(type = "viridis")
```



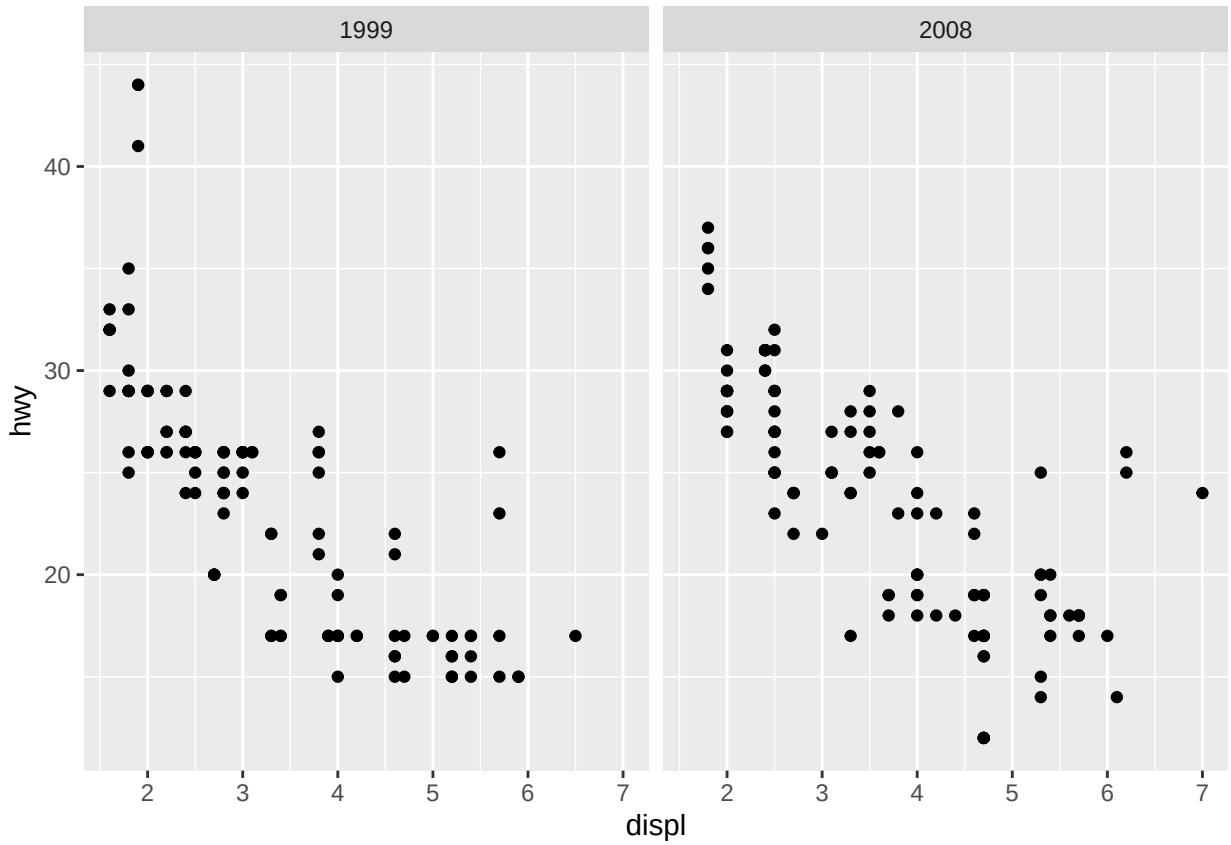

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  coord_polar()
```



Faceting

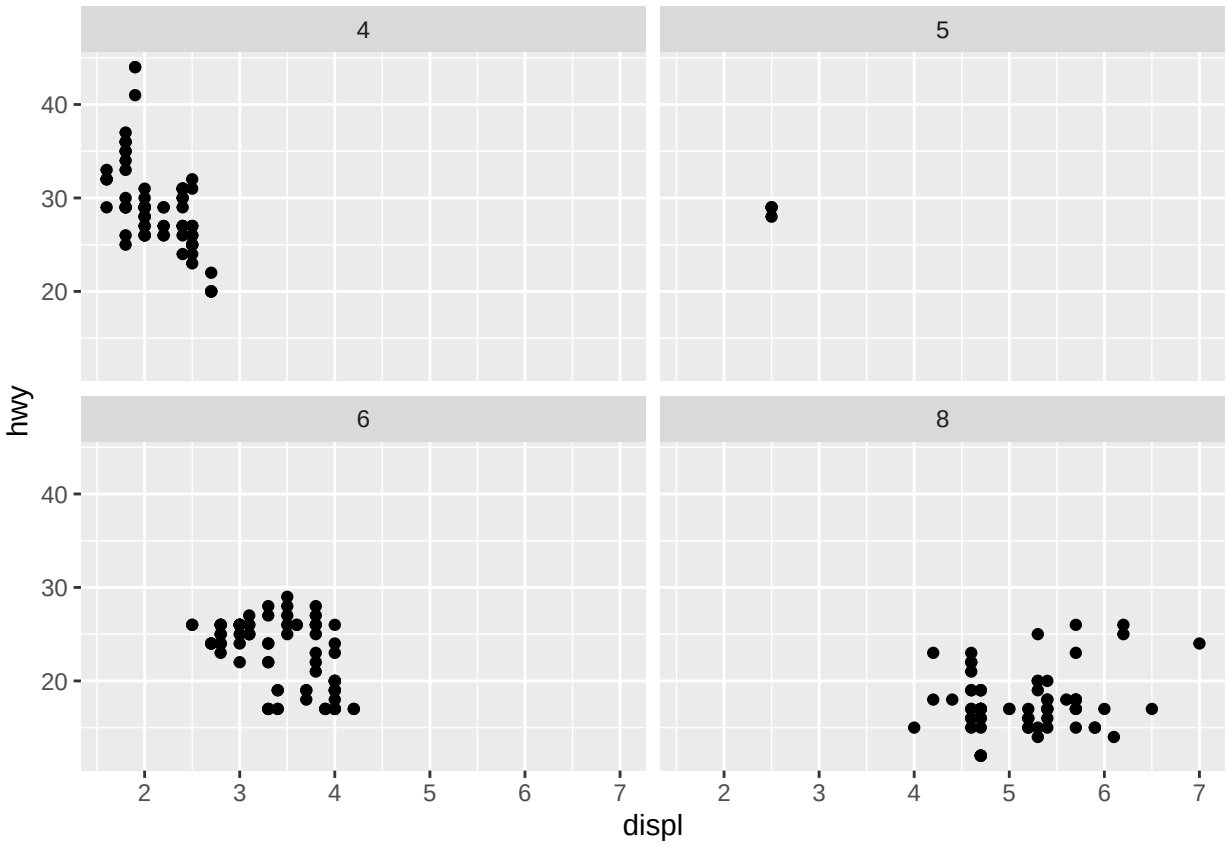
grid:

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(.~year)
```



wrap:

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



A more complicated embedded grammer

```
troops <- read.table("minard-troops.txt", header = T)
cities <- read.table("minard-cities.txt", header = T)

head(troops)
```

```
##   long  lat survivors direction group
## 1 24.0 54.9    340000         A     1
## 2 24.5 55.0    340000         A     1
## 3 25.5 54.5    340000         A     1
## 4 26.0 54.7    320000         A     1
## 5 27.0 54.8    300000         A     1
## 6 28.0 54.9    280000         A     1
```

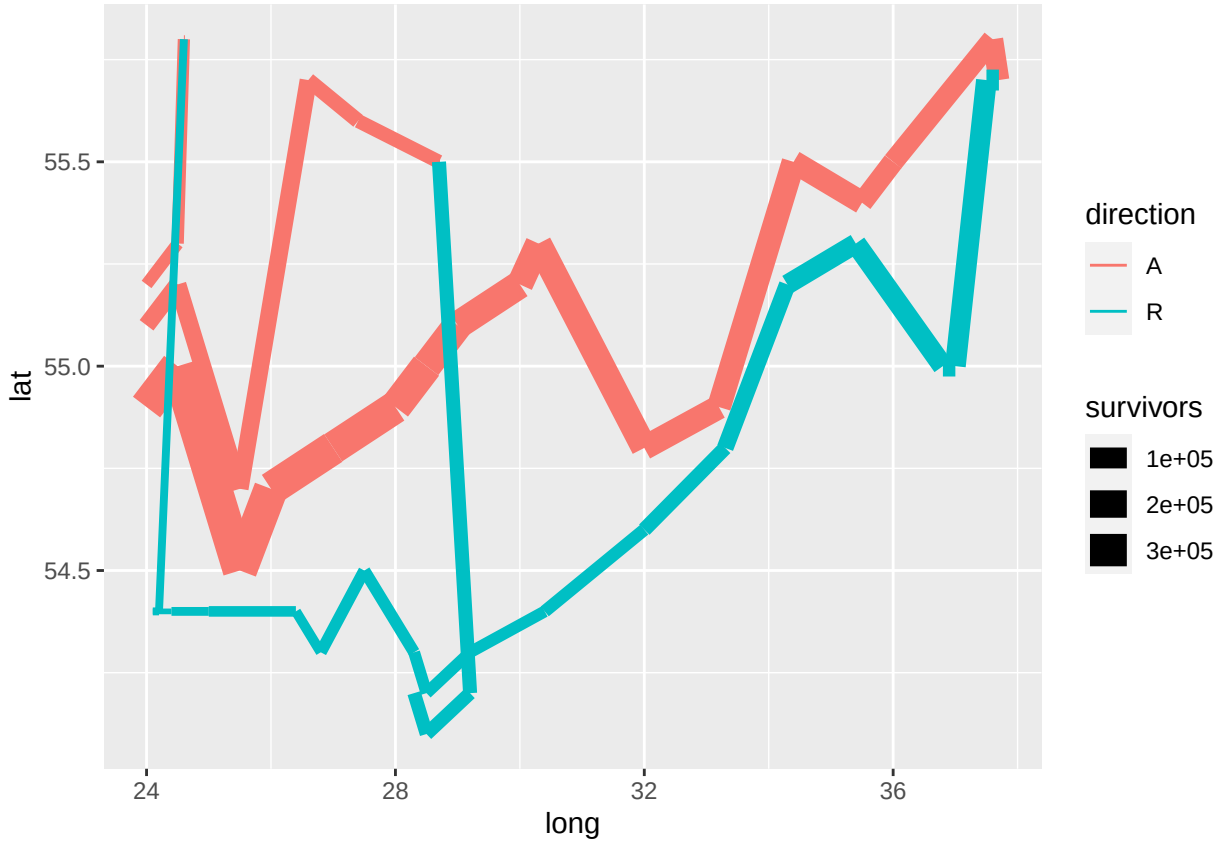
```
head(cities)
```

```
##   long  lat      city
## 1 24.0 55.0     Kowno
## 2 25.3 54.7     Wilna
## 3 26.4 54.4  Smorgoni
## 4 26.8 54.3 Moiodexno
## 5 27.7 55.2 Gloubokoe
## 6 27.6 53.9     Minsk
```

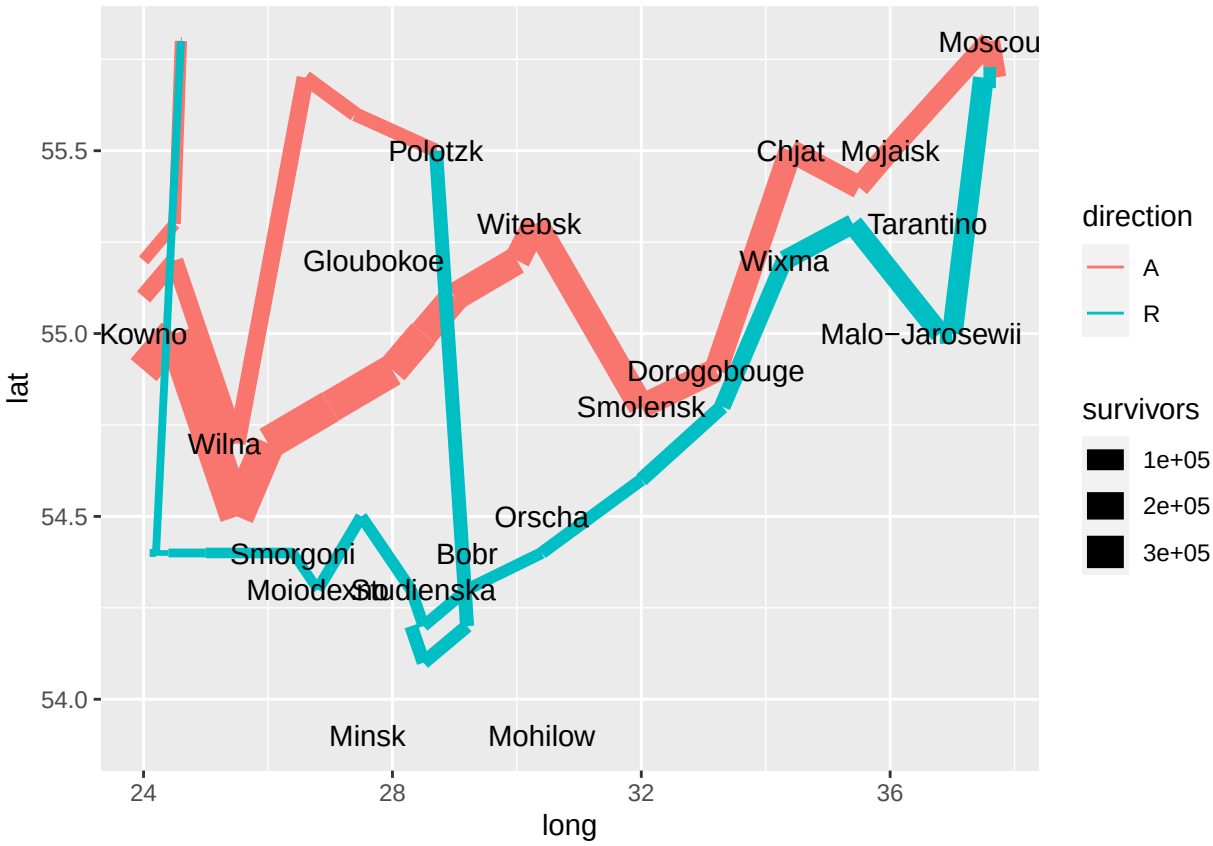
```
plot_troops <- ggplot(troops, aes(long, lat)) +
  geom_path(aes(size = survivors, colour = direction, group = group))
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
```

```
plot_troops
```




```
plot_both <- plot_troops +
  geom_text(aes(label = city), size = 4, data = cities)
plot_both
```




```
library(mapproj)
```

```
## Loading required package: maps
```

```
##
## Attaching package: 'maps'
```

```
## The following object is masked from 'package:purrr':
##
##     map
```

```
plot_polished <- plot_both +
  scale_size(
    range = c(0, 12),
    # breaks = c(10000, 20000, 30000),
    # labels = c("10,000", "20,000", "30,000")
  ) +
  scale_color_manual(values = c("tan", "grey50")) +
  coord_map() +
  labs(title = "Map of Napoleon's Russian campaign of 1812") +
  theme_void() +
  theme(legend.position = "none")
plot_polished
```

Map of Napoleon's Russian campaign of 1812