# Week 4 - Codebook

October 25, 2024

## 1 Week 4: Visualizing Data

### 1.1 Clear the entire workspace

[1]:
```
rm(list = ls())
```

### 1.2 Load libraries

[3]:
```
ReqdLibs =␣
 ↪c("readxl","ggplot2","ggthemes","dplyr","tidyr","forcats","janitor","IRdisplay")

invisible(lapply(ReqdLibs, library, character.only = TRUE))
```

## 2 A. The Anatomy of a GGPLOT

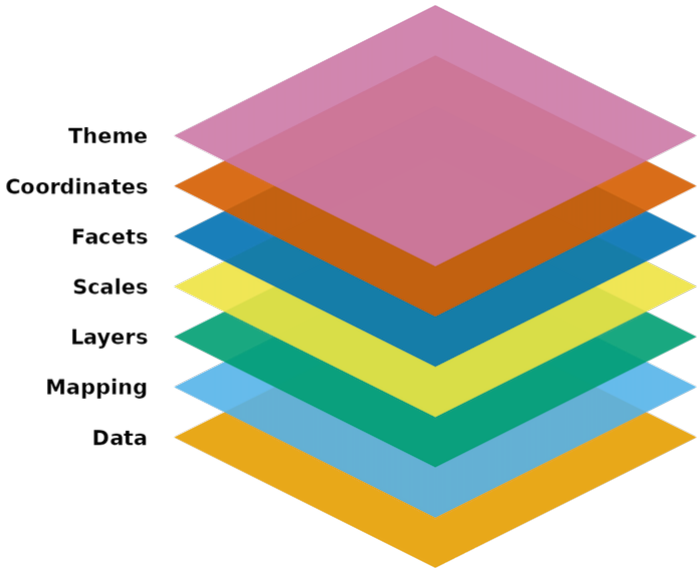



*Image Source*: https://ggplot2.tidyverse.org/articles/ggplot2.html

## 2.1 The Data

This is the foundation of the your whole graphic edifice, which you define through the function `ggplot()`. Make sure it is not wrong (or absent!)

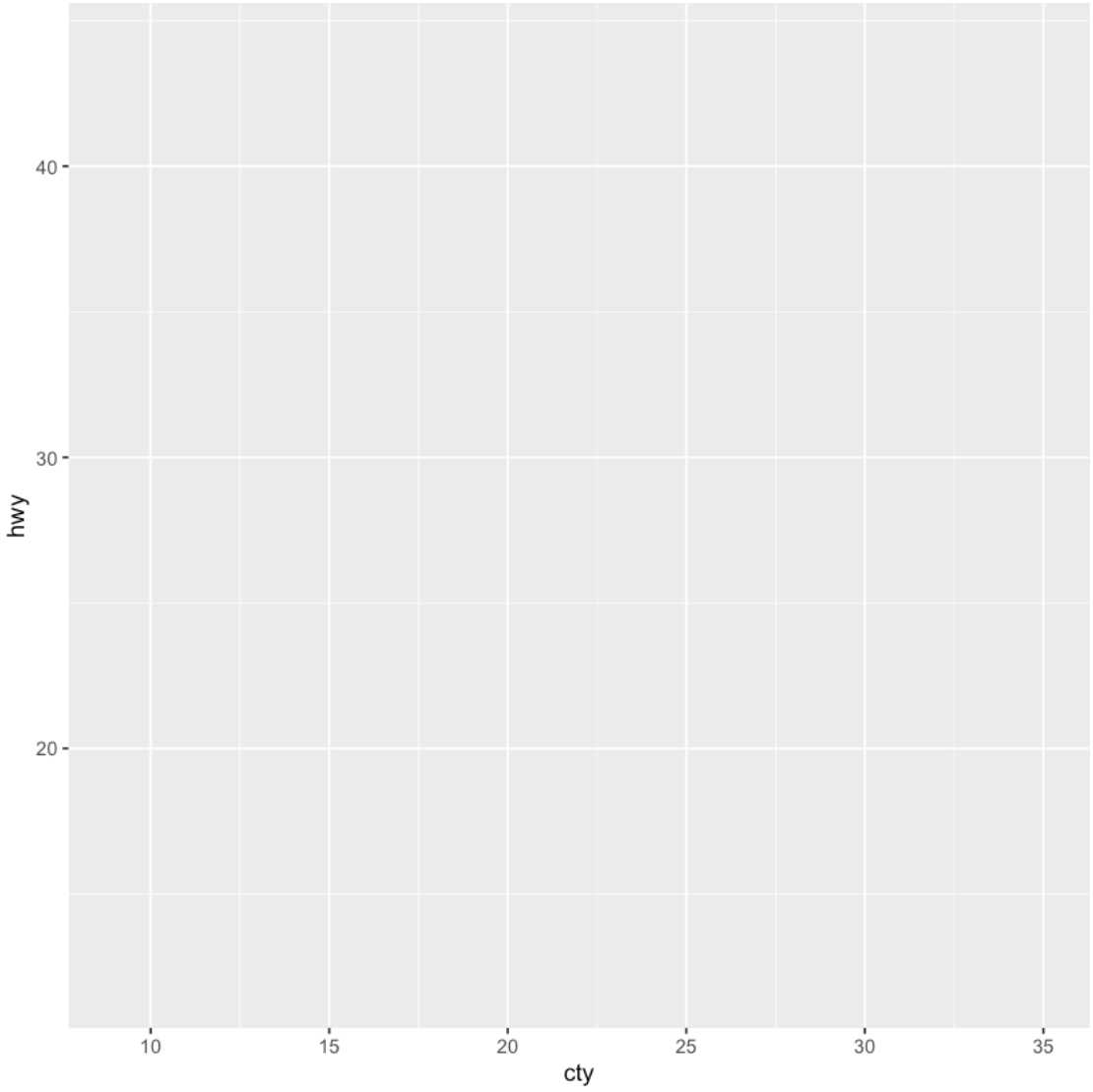
[4]:
```
options(repr.plot.width = 7, repr.plot.height = 7)
ggplot(data = mpg)
```



## 2.2 The Mapping

This is where you define the fundamental descriptors of the graph through the function `aes()`. What you define here are the elements of your graph and those elements are directly linked to **the number of information carrying dimensions of your data** (*Tufte, Chap. 2*, Visual Display of Quantitative Information, $2^{nd}$ Ed).

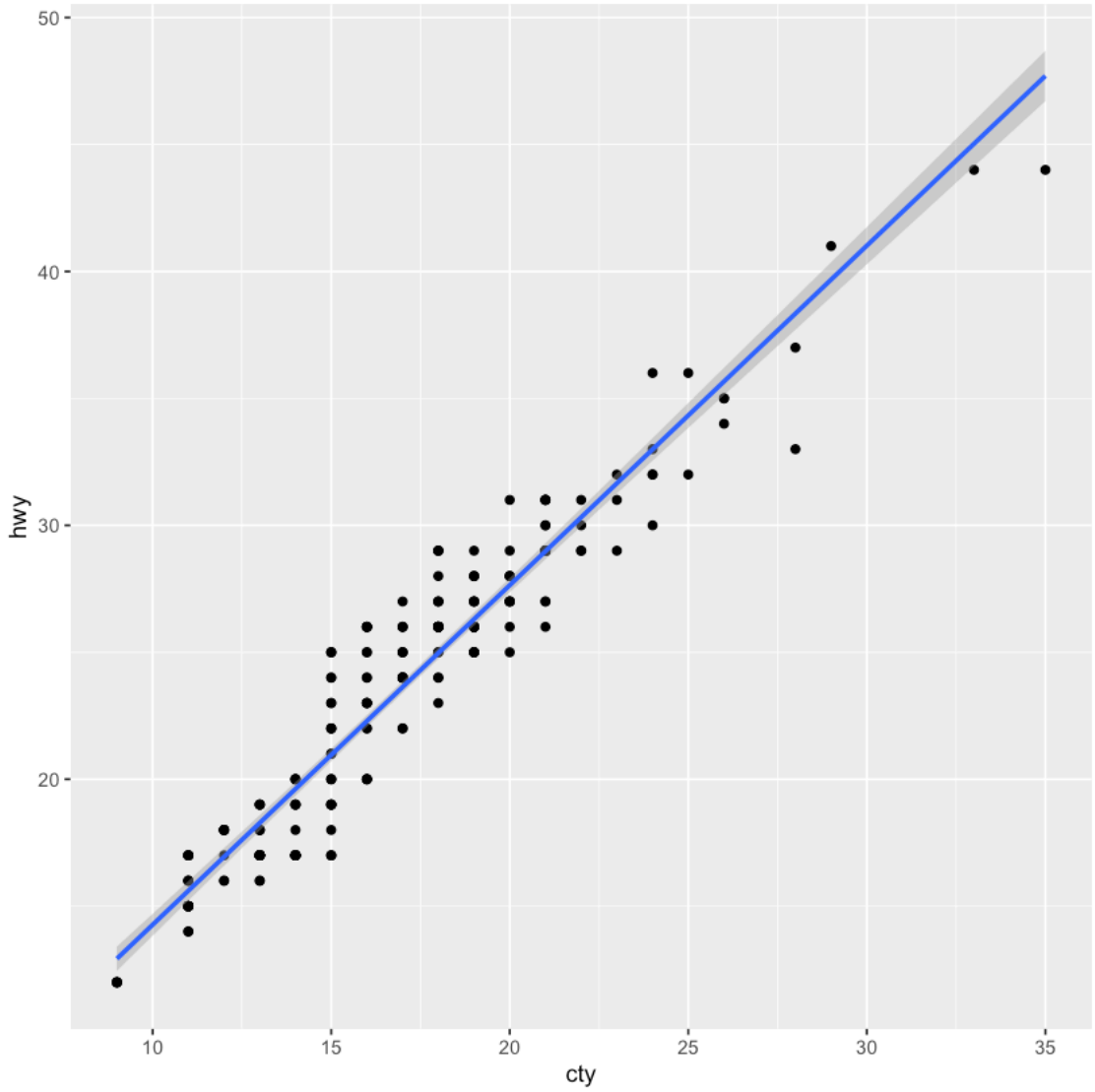
[5]:
```
ggplot(mpg, mapping = aes(x = cty, y = hwy))
```




### 2.3 The Layers

This is where you define the geometric, statistical, or positional elements of the data through the `geom_*()`, `stat_*()`, or `position_*` function respectively. This is where all the action happens and something actually appears on your plot!

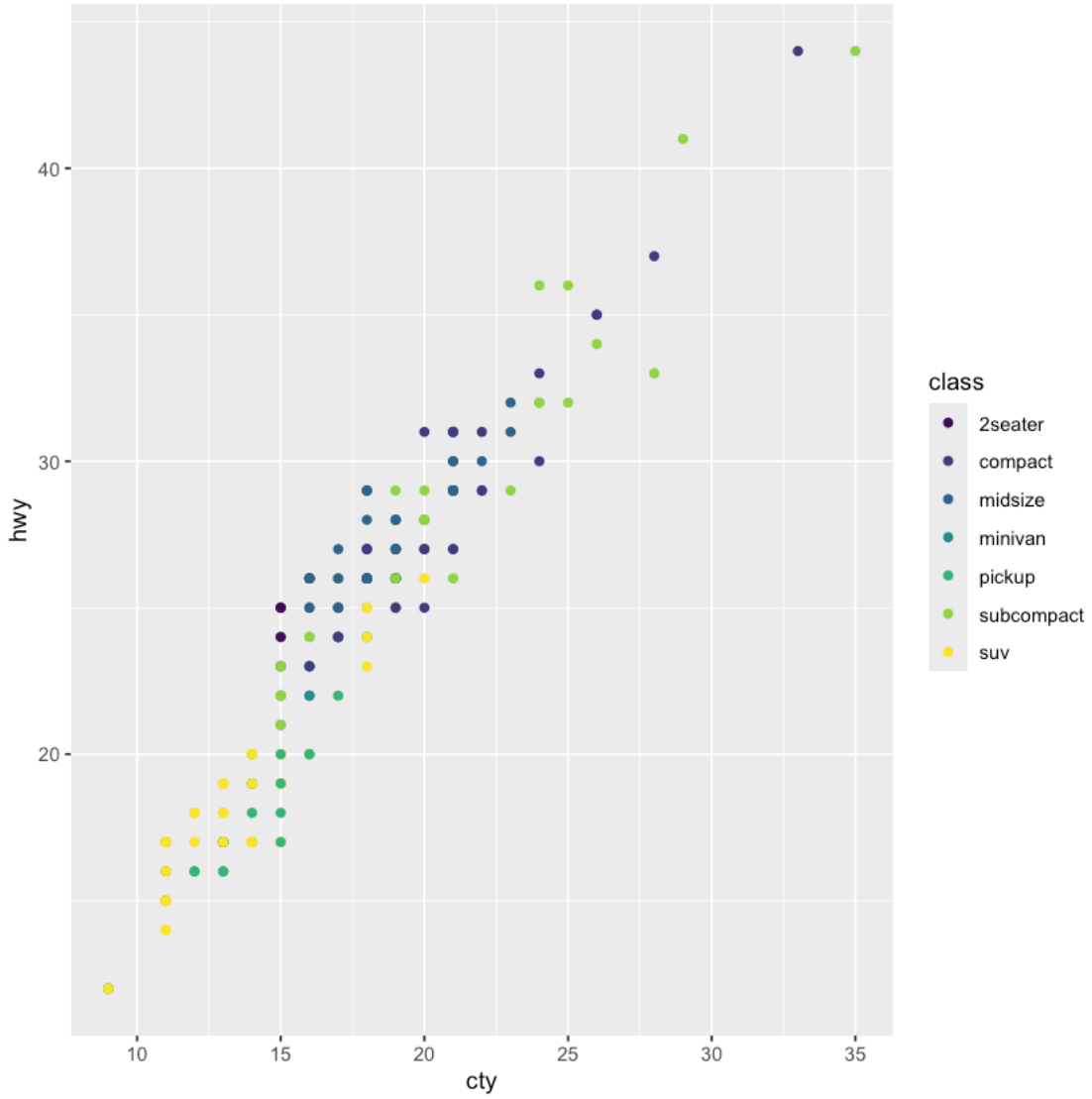
[6]:
```
ggplot(mpg, aes(cty, hwy)) +
  # to create a scatterplot
  geom_point() +
  # to fit and overlay a loess trendline
  geom_smooth(formula = y ~ x, method = "lm")
```




## 2.4 Enhancements

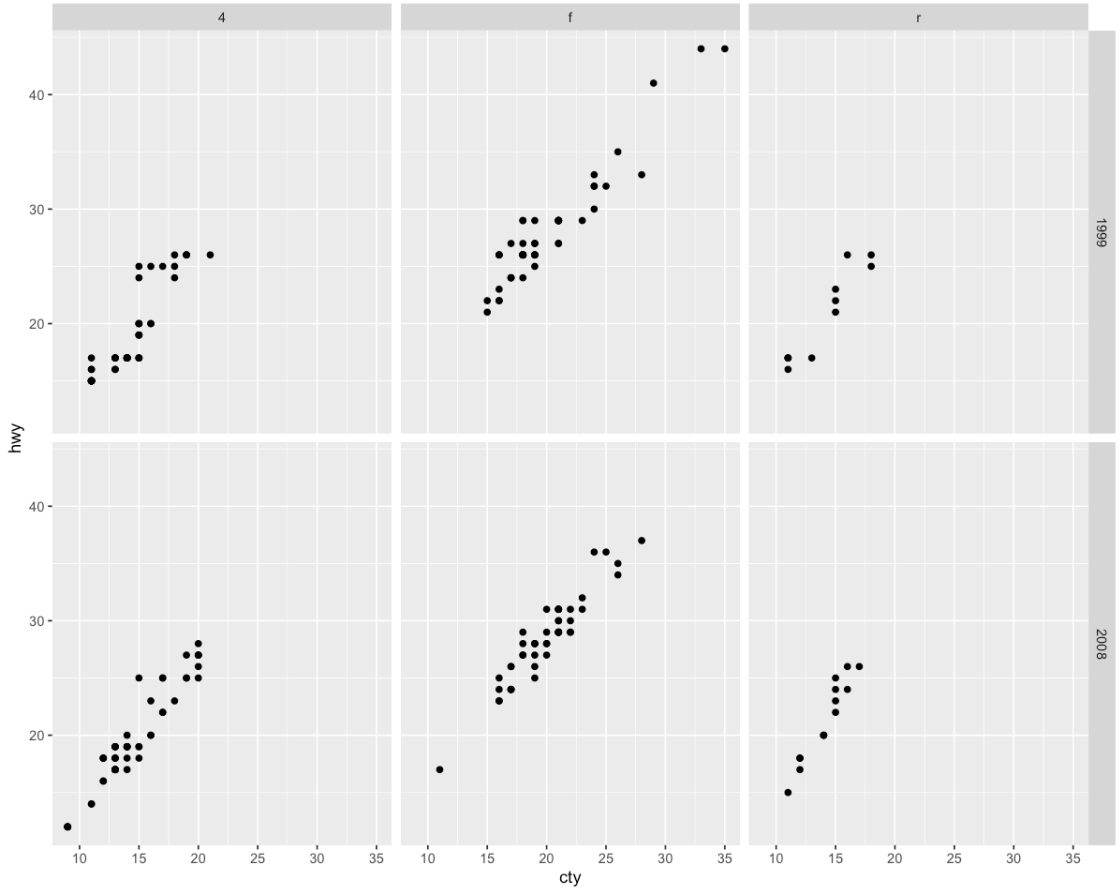
### 2.4.1 Scales

[7]:
```
ggplot(mpg, aes(cty, hwy, colour = class)) +
  geom_point() +
  scale_colour_viridis_d()
```



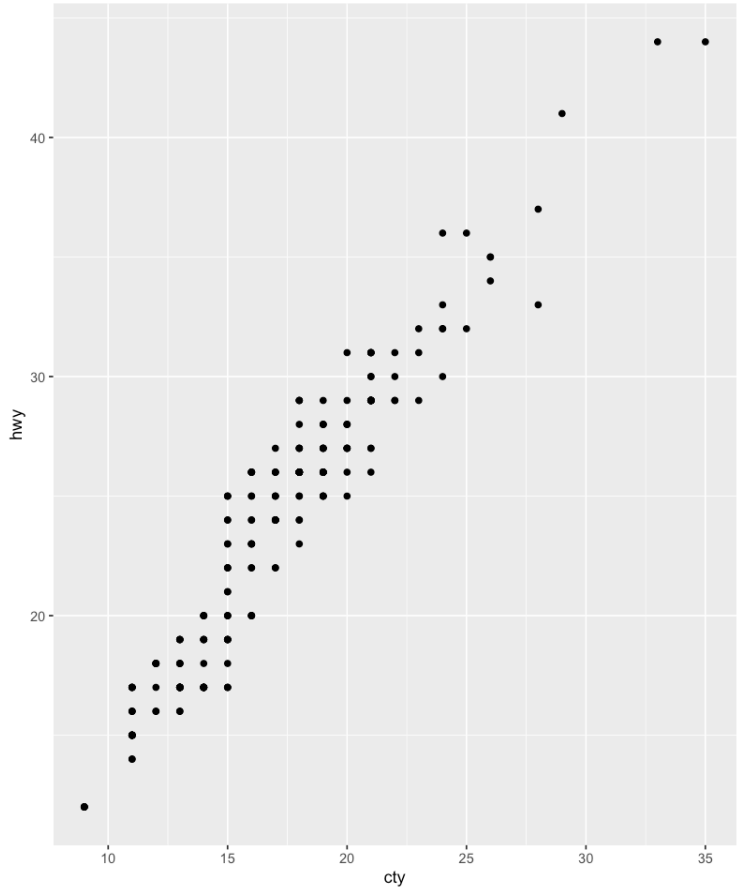

### 2.4.2 Facets

[8]:
```
options(repr.plot.width = 10, repr.plot.height = 8)
ggplot(mpg, aes(cty, hwy)) +
  geom_point() +
  facet_grid(year ~ drv)
```



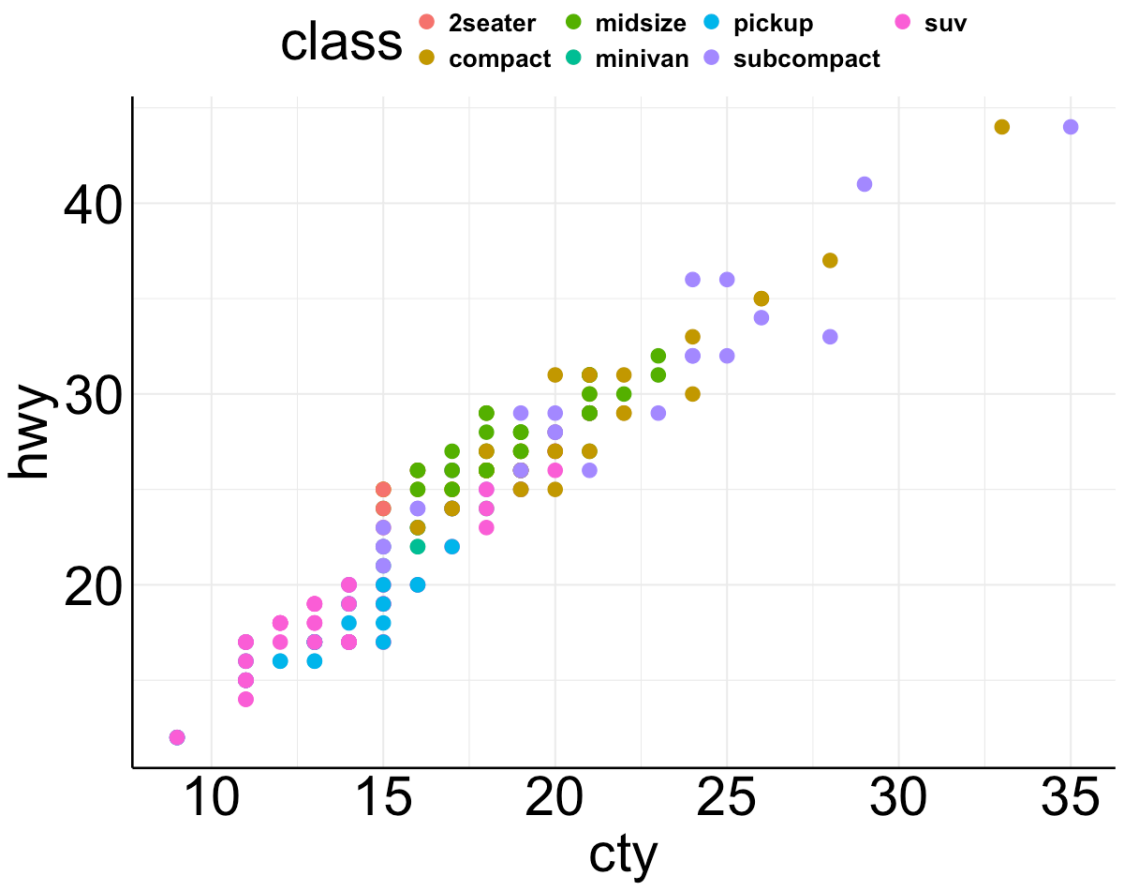

### 2.4.3 Coordinates

[9]:
```
ggplot(mpg, aes(cty, hwy)) +
  geom_point() +
  coord_fixed()
```



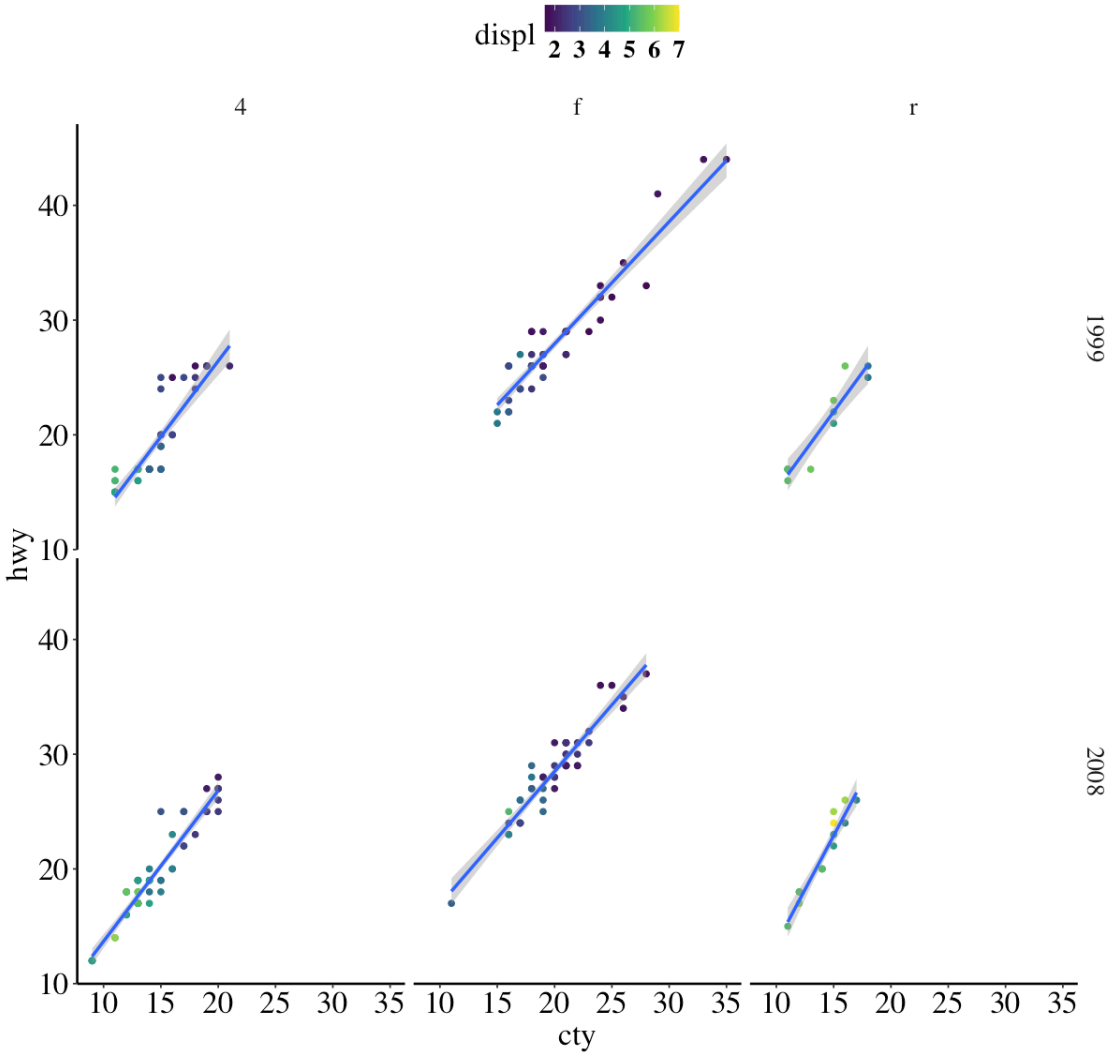

### 2.4.4 Theme

[10]:
```
ggplot(mpg, aes(cty, hwy, colour = class)) +
  geom_point(size = 4) +
  theme_minimal() +
  theme(
    legend.position = "top",
    legend.text=element_text(size=16,face="bold"),
    axis.line = element_line(linewidth = 0.75),
    axis.text = element_text(colour = "black",size=35),
    text = element_text(colour = "black",size=35)
  )
```




### 2.5 All together

[11]:
```
options(repr.plot.width = 10, repr.plot.height = 10)
ggplot(mpg, aes(cty, hwy)) +
  geom_point(mapping = aes(colour = displ)) +
  geom_smooth(formula = y ~ x, method = "lm") +
  scale_colour_viridis_c() +
  facet_grid(year ~ drv) +
  coord_fixed() +
  theme_tufte() +
  theme(
    legend.position = "top",
    legend.text=element_text(size=16,face="bold"),
    axis.line = element_line(linewidth = 0.75),
    axis.text = element_text(colour = "black",size=20),
    text = element_text(colour = "black",size=20)
  )
```




# 3 B. Reproducing a figure using GGPLOT

## 3.1 Import data

[12]:
```
dat.raw = read.csv(file = "rawdata.csv", header = TRUE)
head(dat.raw)
dim(dat.raw)
```

A data.frame: 6 × 8

| | X <int> | t <int> | R <dbl> | VT <dbl> | VE <dbl> | VO2 <dbl> | Sub <chr> | trial <chr> |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 0.9855204 | 0.7548531 | 13.97876 | 408.6987 | Sub1 | rest |
| 2 | 2 | 5 | 0.9628962 | 0.5538989 | 11.62026 | 304.7868 | Sub1 | rest |
| 3 | 3 | 8 | 0.9976182 | 0.7058896 | 12.87337 | 347.7043 | Sub1 | rest |
| 4 | 4 | 10 | 0.9754142 | 0.7823950 | 19.97604 | 614.8135 | Sub1 | rest |
| 5 | 5 | 13 | 0.9758078 | 0.4957548 | 10.89571 | 253.6431 | Sub1 | rest |
| 6 | 6 | 16 | 0.9630747 | 0.7874953 | 14.90527 | 437.0967 | Sub1 | rest |

1. 15062 2. 8

## 3.2 Transform data

### 3.2.1 Reordering `Sub` factor

so that it is sorted numerically rather than alphabetically

[13]:
```
# reorder the Sub factor so that it is sorted numerically rather than␣
 ↪alphabetically

# fct_reorder is in the forcats package which does not rely on 'exact' names␣
 ↪for reordering
# gsub or global substitution funcn that looks for a specific text pattern␣
 ↪(regex) and replaces it
# (kind of like ctrl F & replace)
dat.raw$Sub <- fct_reorder(dat.raw$Sub, as.numeric(gsub("Sub", "", dat.
 ↪raw$Sub)))
levels(dat.raw$Sub)
```

1. 'Sub1' 2. 'Sub2' 3. 'Sub3' 4. 'Sub4' 5. 'Sub5' 6. 'Sub6' 7. 'Sub7' 8. 'Sub8' 9. 'Sub9' 10. 'Sub10' 11. 'Sub11' 12. 'Sub12' 13. 'Sub13'

### 3.2.2 Trial = experimental manipulations!

All participants did 1 single `trial` walking under different conditions. Hence, we have 6 trials + rest. ***But***, `trial` as a variable is pretty uninformative. We want to understand what these trials mean so that we can plot how some response, $\boldsymbol{y}$ ($VO_2$ etc) changed as a result of the independently manipulated variable, $\boldsymbol{x}$ (incline or speed).

| | |
|---|---|
| **Trial 1** | Level, 1.3 m/s |
| **Trial 2** | Level, 0.8 m/s |
| **Trial 3** | Uphill, 1.3 m/s |
| **Trial 4** | Uphill, 0.8 m/s |
| **Trial 5** | Downhill, 1.3 m/s |
| **Trial 6** | Downhill, 0.8 m/s |

[14]:
```

```

```
# based on above, we define trial number (conditions) that match the incline␣
 ↪and speeds
# these are 'condition sets' which we will call when we apply conditional␣
 ↪logics next.

# inclines
lev = c("1","2")
uph = c("3","4")
dwh = c("5","6")

# speeds
fs = c("1","3","5")
sl = c("2","4","6")
```

[15]:
```
dat.raw %>%
# so we can refer to those trial conditions more succinctly
# let's separate these characters into workable parts
separate(trial,into=c("prefix","num"), sep = " ",fill="right",remove = FALSE)␣
 ↪%>%

# rest is coded differently from them, so we fill out the new variable type␣
 ↪with "rest"
mutate(num = if_else(prefix == "rest", prefix, num)) %>%

# now we are ready to define some new informative variables from our␣
 ↪non-informative variable "type"
mutate(cond = if_else(num == "rest", num, "walk"),

       incline = case_when(num %in% lev ~ "level",
                           num %in% uph ~ "uphill",
                           num %in% dwh ~ "downhill",
                           num == "rest" ~ "rest",
                           TRUE ~ NA_character_),

       speed = case_when(num %in% fs ~ 1.3,
                         num %in% sl ~ 0.8,
                         cond == "rest" ~ 0,
                           TRUE ~ NA_real_)) %>%
# remove unwanted variables
select(!c("X","prefix","num")) %>%

# assign to a new 'defined' data frame
{.->> dat.def}

tail(dat.def)
```

A data.frame: 6 × 10

| | t <int> | R <dbl> | VT <dbl> | VE <dbl> | VO2 <dbl> | Sub <fct> | trial <chr> | cond <chr> |
|---|---|---|---|---|---|---|---|---|
| 15057 | 426 | 0.8729754 | 0.6416436 | 14.473163 | 509.72379 | Sub9 | trial 6 | walk |
| 15058 | 429 | 0.9632060 | 0.5324928 | 11.055213 | 361.50629 | Sub9 | trial 6 | walk |
| 15059 | 431 | 0.9126303 | 0.8946287 | 23.963267 | 873.85555 | Sub9 | trial 6 | walk |
| 15060 | 434 | 0.8497735 | 0.5273923 | 14.582274 | 494.51991 | Sub9 | trial 6 | walk |
| 15061 | 435 | 0.8142049 | 0.4304827 | 14.675545 | 394.51961 | Sub9 | trial 6 | walk |
| 15062 | 437 | 0.4978951 | 0.1407739 | 4.590455 | 35.72638 | Sub9 | trial 6 | walk |

### 3.2.3 Trimming time `t`

From the Koelewijn et al paper: "The first 30 seconds of the resting trial, and the first three minutes of each walking trial were disregarded. The rate of oxygen consumption, $VO_2$ in mL/min/kg, and respiratory quotient, $R$ were averaged over time."

[16]:
```
rmv.rest = 30
rmv.walk = 3*60

dat.def %>%
filter(case_when(cond == "rest" ~ t > rmv.rest,
                 cond == "walk" ~ t > rmv.walk,
                 TRUE ~ FALSE)) %>%
group_by(Sub,cond,incline,speed) %>%
summarize_if(is.double, ~ mean(., na.rm = TRUE)) %>%

{.->>dat.summ}

head(dat.summ,7)
```

A grouped_df: 7 × 8

| Sub <fct> | cond <chr> | incline <chr> | speed <dbl> | R <dbl> | VT <dbl> | VE <dbl> | VO2 <dbl> |
|---|---|---|---|---|---|---|---|
| Sub1 | rest | rest | 0.0 | 1.0537845 | 0.7031141 | 12.16978 | 328.9456 |
| Sub1 | walk | downhill | 0.8 | 0.9019424 | 0.7859465 | 20.33910 | 728.5032 |
| Sub1 | walk | downhill | 1.3 | 0.8661119 | 0.8409442 | 25.83893 | 1005.2468 |
| Sub1 | walk | level | 0.8 | 0.8727268 | 1.0036957 | 26.00091 | 992.9288 |
| Sub1 | walk | level | 1.3 | 0.8340541 | 1.0104406 | 32.61279 | 1368.0726 |
| Sub1 | walk | uphill | 0.8 | 0.8661413 | 1.2505480 | 40.73454 | 1687.3811 |
| Sub1 | walk | uphill | 1.3 | 0.8315770 | 1.4630620 | 58.84244 | 2540.5691 |

### 3.2.4 Calculating new variables, $W$ & $C_{meas}$

...from constants and measured variables

$$W = \frac{4.184}{60}(3.972 + 1.078R)\dot{V}O_2$$

[17]:
```
# VO2 in our data table is absolute VO2, to normalize it by the weight of each
↪subject,
# we need weight data from the demographic excel file!
# read excel file

demo=read_excel("SubjectInfo.xlsx")

tail(demo)
```

A tibble: 6 × 8

| Subject No <chr> | Age <dbl> | Reported Weight (kg) <dbl> | Reported Length (cm) <dbl> | Gender <chr> | Level Slow <dbl> | L |
|---|---|---|---|---|---|---|
| Sub8 | 21 | 57 | 173 | F | 599.2760 | 59 |
| Sub9 | 25 | 58 | 170 | M | 575.0346 | 58 |
| Sub10 | 20 | 66 | 170 | F | 673.0147 | 67 |
| Sub11 | 20 | 76 | 188 | M | 783.6798 | 78 |
| Sub12 | 33 | 70 | 178 | M | 715.1903 | 71 |
| Sub13 | 21 | 56 | 169 | F | 661.9267 | 65 |

[18]:
```
# TIP! you could use clean_names function from the janitor package
# to clean var names so that they don't contain spaces and parentheses..
# also makes it easyt to use dplyr to mutate later #uncomment below

demo.clean = demo %>% clean_names()
dat.summdemo = merge(dat.summ,demo.clean,by.x = 'Sub', by.y = 'subject_no')

head(demo.clean)
```

A tibble: 6 × 8

| subject_no <chr> | age <dbl> | reported_weight_kg <dbl> | reported_length_cm <dbl> | gender <chr> | level_slow <dbl> | level_ <dbl |
|---|---|---|---|---|---|---|
| Sub1 | 26 | 86 | 185 | M | 885.5529 | 891.5 |
| Sub2 | 28 | 77 | 178 | F | 767.7686 | 760.2 |
| Sub3 | 21 | 52 | 170 | M | 530.6408 | 558.2 |
| Sub4 | 25 | 73 | 168 | M | NA | NA |
| Sub5 | 34 | 86 | 173 | M | 878.6303 | 898.5 |
| Sub6 | 19 | 54 | 160 | F | 553.4936 | 558.2 |

### 3.2.5 Normalizing $VO_2$ by weight

[19]:
```
dat.summdemo %>%
mutate(adjVO2 = VO2/weight_from_force_plates_kg) %>%
mutate(W = 4.184/60 * (3.972 + 1.078 * R) * adjVO2) %>%
{.->>dat.calc1}

head(dat.calc1)
```

A data.frame: 6 × 17

| | Sub <fct> | cond <chr> | incline <chr> | speed <dbl> | R <dbl> | VT <dbl> | VE <dbl> | VO2 <dbl> | a |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Sub1 | rest | rest | 0.0 | 1.0537845 | 0.7031141 | 12.16978 | 328.9456 | 2 |
| 2 | Sub1 | walk | downhill | 0.8 | 0.9019424 | 0.7859465 | 20.33910 | 728.5032 | 2 |
| 3 | Sub1 | walk | downhill | 1.3 | 0.8661119 | 0.8409442 | 25.83893 | 1005.2468 | 2 |
| 4 | Sub1 | walk | level | 0.8 | 0.8727268 | 1.0036957 | 26.00091 | 992.9288 | 2 |
| 5 | Sub1 | walk | level | 1.3 | 0.8340541 | 1.0104406 | 32.61279 | 1368.0726 | 2 |
| 6 | Sub1 | walk | uphill | 0.8 | 0.8661413 | 1.2505480 | 40.73454 | 1687.3811 | 2 |

### 3.2.6 Now calculating $C_{meas}$ :

From the Koelewijn et al paper: "The resting trial was subtracted from each walking trial. The metabolic rate was divided by walking speed to find the measured metabolic cost in J/kg/m:"

$$C_{meas} = \frac{W}{v}$$

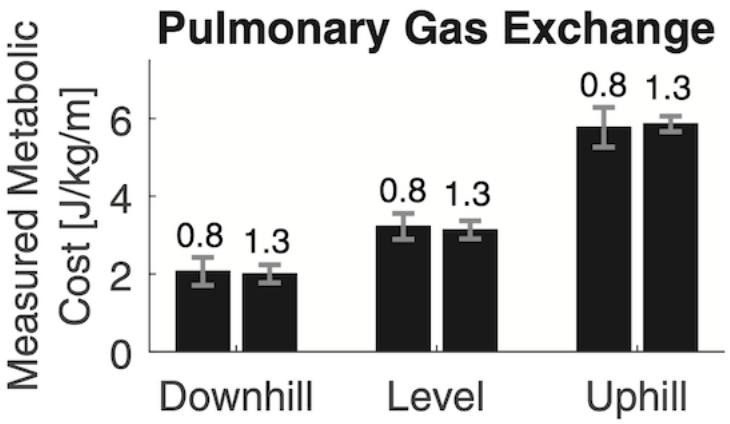
[20]:
```
# extracting the rest trials only so we can merge it as a column next
dat.calc1 %>%
filter(cond=="rest") %>%
select(Sub,W) %>%
{.->>dat.rest}

# now merging it with the original
merge(dat.calc1,dat.rest,by='Sub',suffixes = c('','_rest')) %>%
filter(cond!="rest") %>%
mutate(W_adj = W - W_rest) %>%
mutate(C_meas = W_adj/speed) %>%
{.->>dat.calc2}

head(dat.calc2)
```

A data.frame: 6 × 20

| | Sub <fct> | cond <chr> | incline <chr> | speed <dbl> | R <dbl> | VT <dbl> | VE <dbl> | VO2 <dbl> | a |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Sub1 | walk | downhill | 0.8 | 0.9019424 | 0.7859465 | 20.33910 | 728.5032 | 2 |
| 2 | Sub1 | walk | downhill | 1.3 | 0.8661119 | 0.8409442 | 25.83893 | 1005.2468 | 2 |
| 3 | Sub1 | walk | level | 1.3 | 0.8340541 | 1.0104406 | 32.61279 | 1368.0726 | 2 |
| 4 | Sub1 | walk | uphill | 0.8 | 0.8661413 | 1.2505480 | 40.73454 | 1687.3811 | 2 |
| 5 | Sub1 | walk | uphill | 1.3 | 0.8315770 | 1.4630620 | 58.84244 | 2540.5691 | 2 |
| 6 | Sub1 | walk | level | 0.8 | 0.8727268 | 1.0036957 | 26.00091 | 992.9288 | 2 |

## 3.3 Visualizing metabolic cost $C_{meas}$

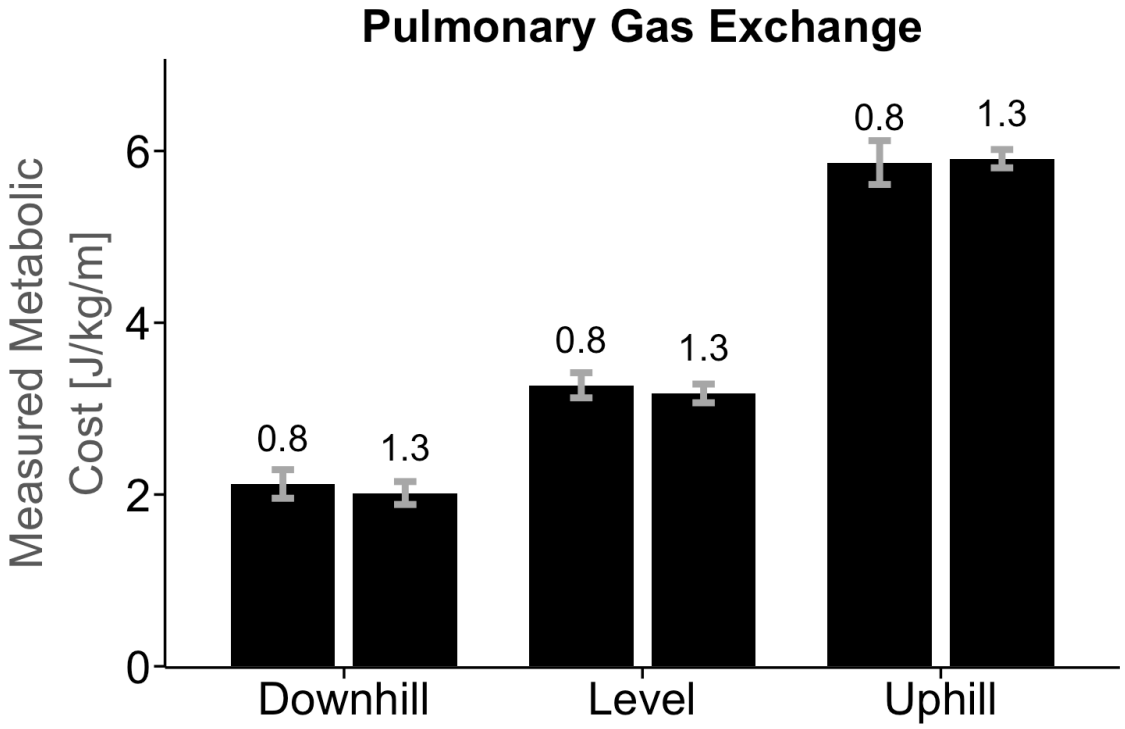
Finally after all that we have the variable we want to plot. Let's now reproduce the bottom-right panel of Figure 5 from the paper. It's shown below for reference:




[22]:
```
thm = theme(
          legend.text=element_text(size=16,face="bold"),
          legend.position = "top",
          legend.title=element_text(size=16,face="bold"),
          title =element_text(size=14, face='bold'),
          text = element_text(colour = "black",size=18),
          plot.title = element_text(colour = "black",size = 35, face = "bold",␣
 ↪hjust = 0.5),
          axis.ticks.length = unit(0.3,"cm"),
          axis.line = element_line(colour = "black",size=1),
          axis.ticks = element_line(colour = "black",size=1),
          axis.text = element_text(colour = "black",size=35),
          axis.title.y =element_text(size=35, colour = "grey35", face =␣
 ↪"plain",
                                     lineheight = 1.1, margin = margin(r = 10)))
```

[24]:
```
options(repr.plot.width = 12, repr.plot.height = 8)

# FUNCTION CALL
ggplot(dat.calc2, aes(x = incline,y = C_meas, group = speed, label = speed)) +

# LAYERS THAT SUMMARIZE WHILE PLOTTING! - this is one of the most powerful␣
 ↪features of ggplot
stat_summary(geom = "bar", fun.y = mean, col = NA, fill = "black", width = 0.7,␣
 ↪na.rm = TRUE,
             position=position_dodge(width = 0.82, preserve = 'single')) +
stat_summary(geom = "errorbar",fun.data = mean_se, width = 0.15, lwd=2.5,␣
 ↪col="darkgray", na.rm = TRUE,
             position=position_dodge(0.82, preserve = 'single')) +
stat_summary(geom = "text", fun.y = mean, position = position_dodge(0.82,␣
 ↪preserve = 'total'), na.rm = TRUE,
             vjust = -1.25, size = 10) +
```

```r
# AXIS LIMITS
coord_cartesian(ylim = c(0.307,6.75)) +

# LABELS
scale_x_discrete(labels = c("Downhill", "Level", "Uphill")) +  # capitalize␣
 ↪label initials :/
labs(title = "Pulmonary Gas Exchange", x = "", y = "Measured Metabolic\nCost [J/
 ↪kg/m]") +
theme_classic() + thm
```




## 4 Additional Resources

1) GGPLOT book: https://ggplot2-book.org/introduction.html#what-is-the-grammar-of-graphics
2) R Graphics Cookbook: https://r-graphics.org

**Try it yourself!** 1) Can you plot the graph above as a boxplot where the color is determined by `speed`? * What aspects of the graph would you be changing? 2) Can you overlay individual data points on top of that new boxplot? 3) Can you plot a scatterplot that shows `Age` on the x against metabolic cost `C_meas` on the y? * Then, facet it by `incline` and `speed`?

## 5 The End