Data analysis in R applied to Marine Science

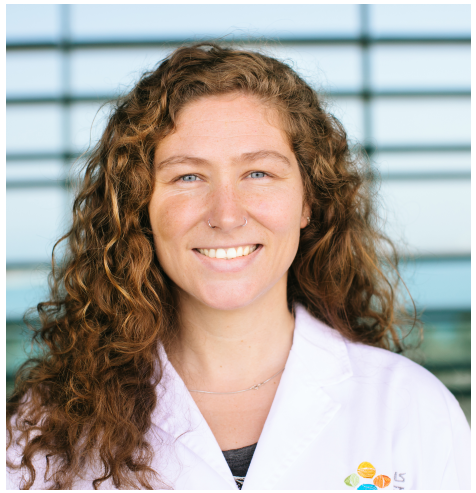
Day 2. Advanced plots, labels, dates
28 January 2019

Tamara Huete-Stauffer - Grégoire Michoud - Daffne López-Sandoval - Malika Kheireddine

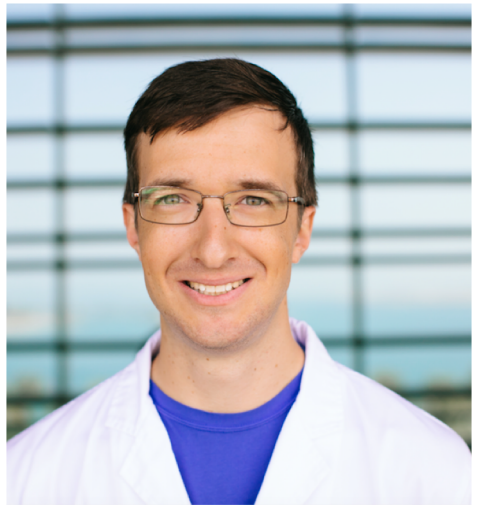
جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

RSRC Red Sea Research Center

Instructors



Tamara Huete-Stauffer
Postdoctoral fellow
Prof. Xelu Morán
Marine microbial ecology
tamara.huetestauffer@kaust.edu.sa



Grégoire Michoud
Postdoctoral fellow
Prof. Daniele Daffonchio
Microbiology of brine pools
gregoire.michoud@kaust.edu.sa

Daffne López-Sandoval
Postdoctoral Fellow
Prof. Susana Agustí
Phytoplankton ecology
daffne.lopezsandoval@kaust.edu.sa

Advanced plots

folder: Day2_Plots/GGplotCourse/Exercise

Preliminary information

- We are going to use the dataset **mtcars** (Motor Trend Car Road Tests)

```
data("mtcars")
str(mtcars)
?mtcars
```

Load
View structure
Get help/information about the dataset

Preliminary information

- We are going to use the dataset **mtcars** (Motor Trend Car Road Tests)

```
data("mtcars")
str(mtcars)
?mtcars
```

Load
View structure (all numeric)
Get help/information about the dataset

- We are going to create factors to facilitate the plotting

```
mtcars$gear <- factor(mtcars$gear,
levels=c(3,4,5),
labels=c("3 gears","4 gears","5 gears"))
```

Data
Unique values
Names/labels

Preliminary information

Create factors for columns

- **am** (Transmission)

- **cyl** (Number of cylinders)

Preliminary information

Create factors for columns

- **am** (Transmission)

```
mtcars$am <- factor(mtcars$am,levels=c(0,1),
labels=c("Automatic","Manual"))
```

- **cyl** (Number of cylinders)

```
mtcars$cyl <- factor(mtcars$cyl,levels=c(4,6,8),
labels=c("4cyl","6cyl","8cyl"))
```

Plotting systems in R

- Base plotting system ("fast and easy" but ugly)
- Lattice package
- **ggplot2 package (not so fast and easy but highly customable and effective effective for creating publication quality graphics.)**

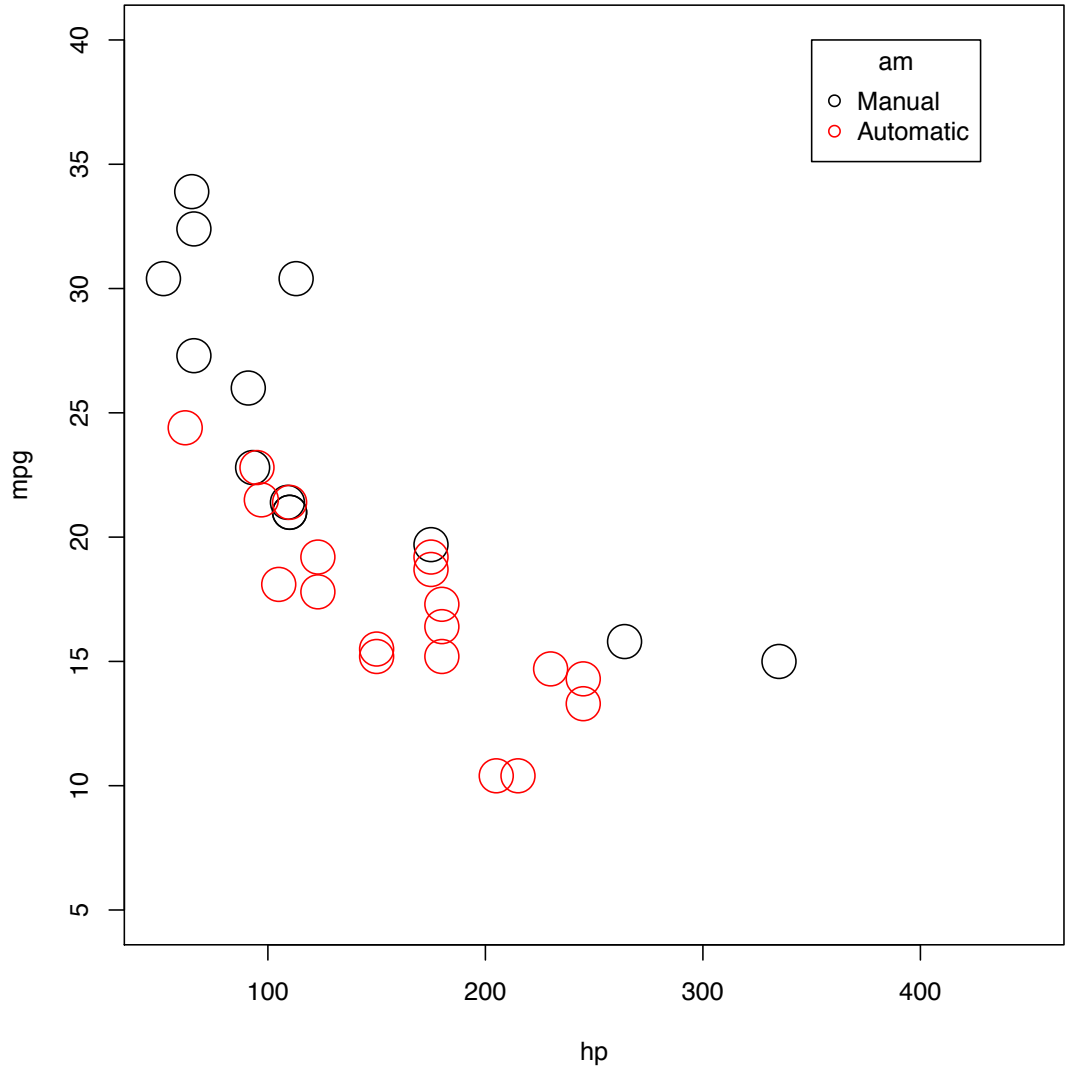
```
install.packages("ggplot2")
library("ggplot2")
```

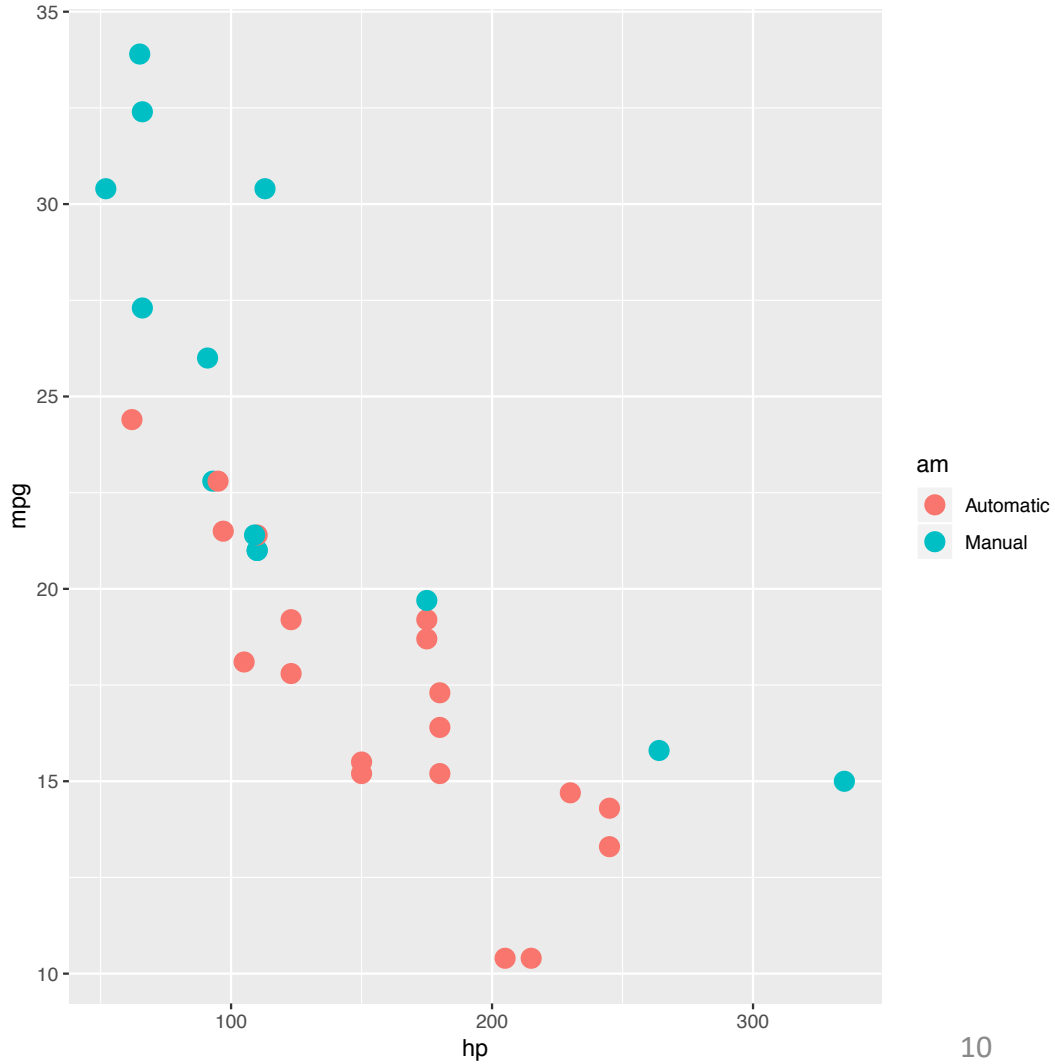
Plotting systems in R

- Base plotting system ("fast and easy" but ugly)
- Lattice package
- **ggplot2 package (not so fast and easy but highly customable and effective effective for creating publication quality graphics.)**

The key to understanding ggplot2 is thinking about a figure in layers

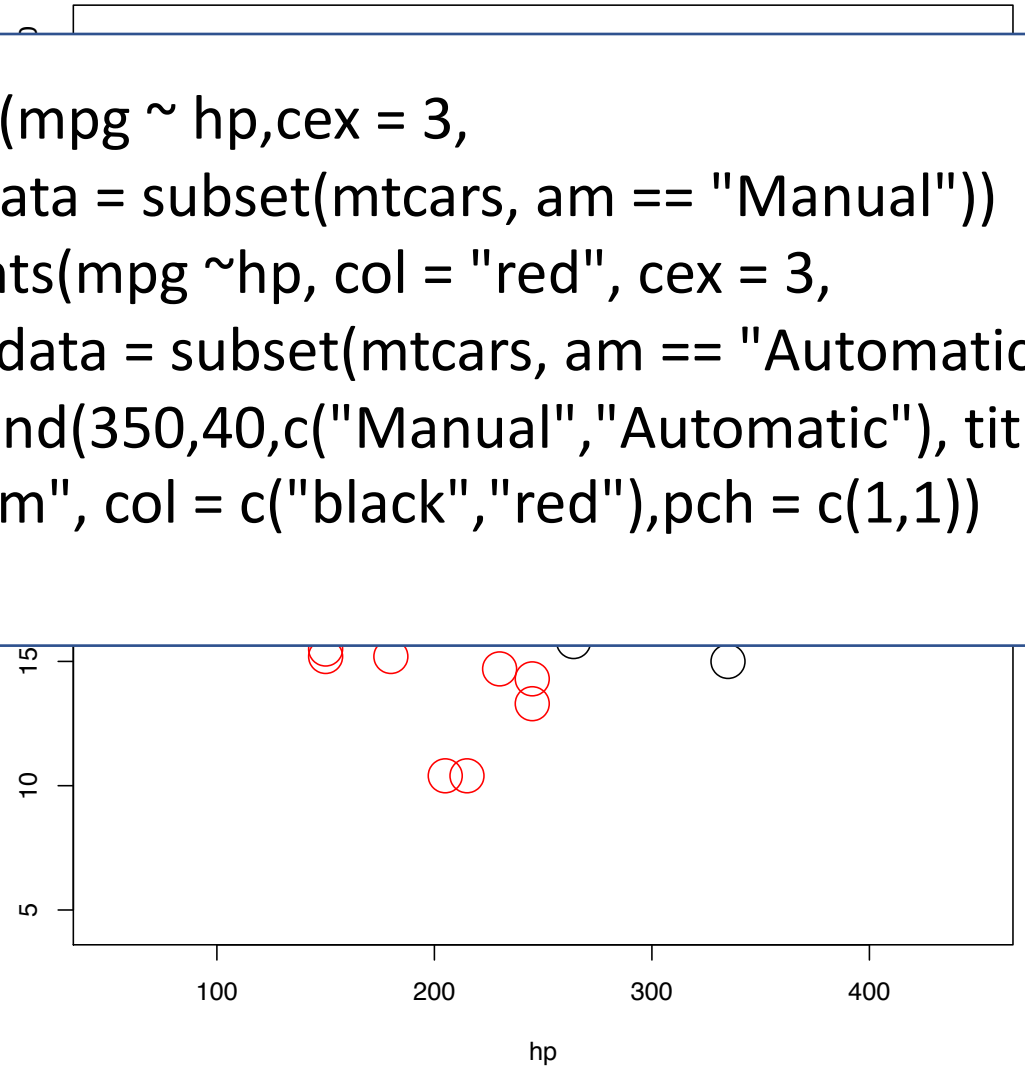
Example between base and ggplot



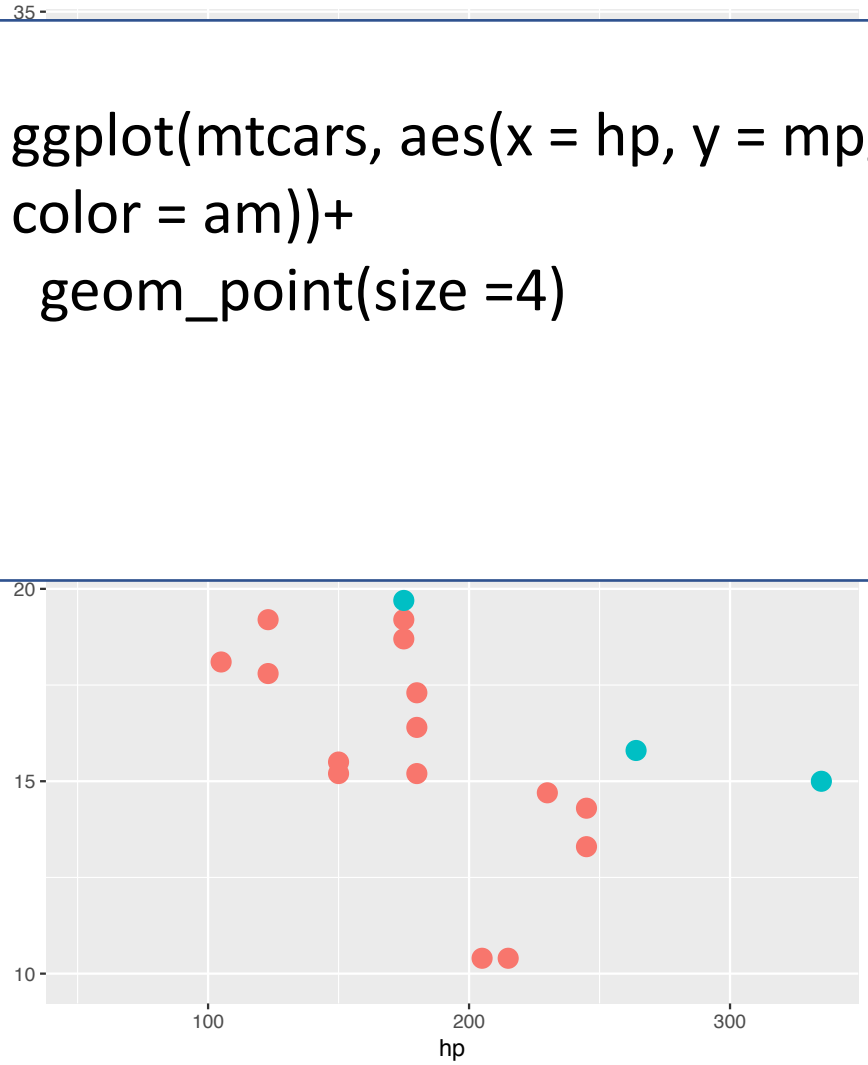


Example between base and ggplot

```
plot(mpg ~ hp,cex = 3,
    data = subset(mtcars, am == "Manual"))
points(mpg ~hp, col = "red", cex = 3,
      data = subset(mtcars, am == "Automatic") )
legend(350,40,c("Manual","Automatic"), title
= "am", col = c("black","red"),pch = c(1,1))
```




```
ggplot(mtcars, aes(x = hp, y = mpg,
color = am))+
  geom_point(size =4)
```



Grammar of graphics or Layers

Any plot can be expressed from the same set of components: a **data** set, a **coordinate system**, and a set of **geoms**–the visual representation of data points.

- Data : Self Explanatory (More or less)
- Coordinate system : how variables in the **data** map to *aesthetic* properties of the figure, aes()
- Geoms :
  - **geom_point**
  - **geom_bar**
  - **geom_box**
  - …

Data

Wide Format

ID	T	P.1	P.2	P.3
1	24.3	10.2	5.5	2.1
2	23.4	10.4	5.7	2.8
3	22.1	10.5	5.9	3.1
4	19.9	10.2	5.2	2.4

Long format

ID	Channel	T	P
1	1	24.3	10.2
2	1	23.4	10.4
3	1	22.1	10.5
4	1	19.9	10.2
1	2	24.3	5.5
2	2	23.4	5.7
3	2	22.1	5.9
4	2	19.9	5.2
1	3	24.3	2.1
2	3	23.4	2.8
3	3	22.1	3.1
4	3	19.9	2.4

The data needs to be in long format !!!

Reshape2 package functions :

melt (Wide to long)

dcast (Long to wide)

Aesthetic or aes()

- Something you can see !
- Examples :
  - Position
  - Color / Fill
  - Shape
  - Size
  - Linetype
- For each geom, different mapping

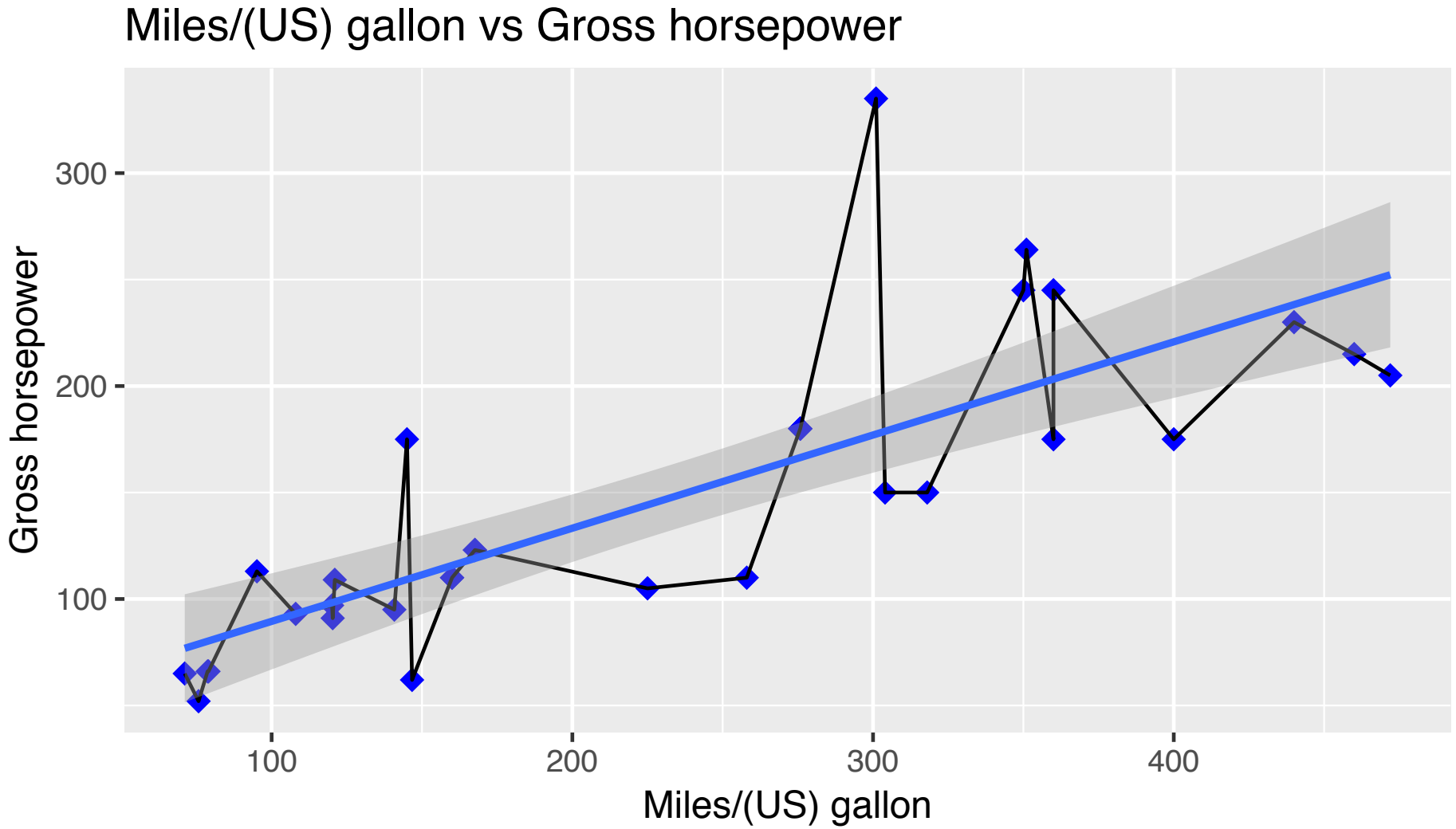
Geometic Objects (geom)

- Geometric objects are the actual marks we put on a plot
- Examples include:
  - Points
  - Lines
  - Boxplot
- A plot must have at least one geom; there is no upper limit
- Add a geom to a plot using the + operator

Example (p1)

```
ggplot(mtcars, aes(x = disp,y = hp))+

  geom_point(size = 3, color = "blue", shape = 18)+

 geom_smooth(method = "lm")+

  labs(title="Miles/(US) gallon vs Gross horsepower",
       x="Miles/(US) gallon", y="Gross horsepower")
```

Example (p1)


Miles/(US) gallon vs Gross horsepower
Gross horsepower
300
200
100
100
200
300
400
Miles/(US) gallon

Example (p1)

Data

Aesthetic

```
ggplot(mtcars, aes(x = disp,y = hp))+

  geom_point(size = 3, color = "blue", shape = 18)+

 geom_smooth(method = "lm")+

  labs(title="Miles/(US) gallon vs Gross horsepower",
       x="Miles/(US) gallon", y="Gross horsepower")
```

Example (p1)

Data

Aesthetic

```
ggplot(mtcars, aes(x = disp,y = hp))+
  geom_point(size = 3, color = "blue", shape = 18)+
  geom_smooth(method = "lm")+
  labs(title="Miles/(US) gallon vs Gross horsepower",
       x="Miles/(US) gallon", y="Gross horsepower")
```

Geometic Object

Geometic Object

Example (p1)

Data

Aesthetic

```
ggplot(mtcars, aes(x = disp,y = hp))+
  geom_point(size = 3, color = "blue", shape = 18)+
  geom_smooth(method = "lm")+
  labs(title="Miles/(US) gallon vs Gross horsepower",
       x="Miles/(US) gallon", y="Gross horsepower")
```

Geometic Object

Geometic Object

Labels

Example (p1)

Ggplot object can be saved

```
p1 <- ggplot(mtcars, aes(x = disp,y = hp))+
  geom_point(size = 3, color = "blue", shape = 18)+
  geom_smooth(method = "lm")+
  labs(title="Miles/(US) gallon vs Gross horsepower",
       x="Miles/(US) gallon", y="Gross horsepower")
p1
```

Example (p1)

Ggplot object can be saved

```
p1 <- ggplot(mtcars, aes(x = disp,y = hp))+
  geom_point(size = 3, color = "blue", shape = 18)+
  geom_smooth(method = "lm")+
  labs(title="Miles/(US) gallon vs Gross horsepower",
       x="Miles/(US) gallon", y="Gross horsepower")
```

Use the ggsave() command to save the plot

```
ggsave("plot1.pdf",p1)
```

Will be saved based on the extension (e.g. pdf)

Aesthetic Mapping VS Assignment

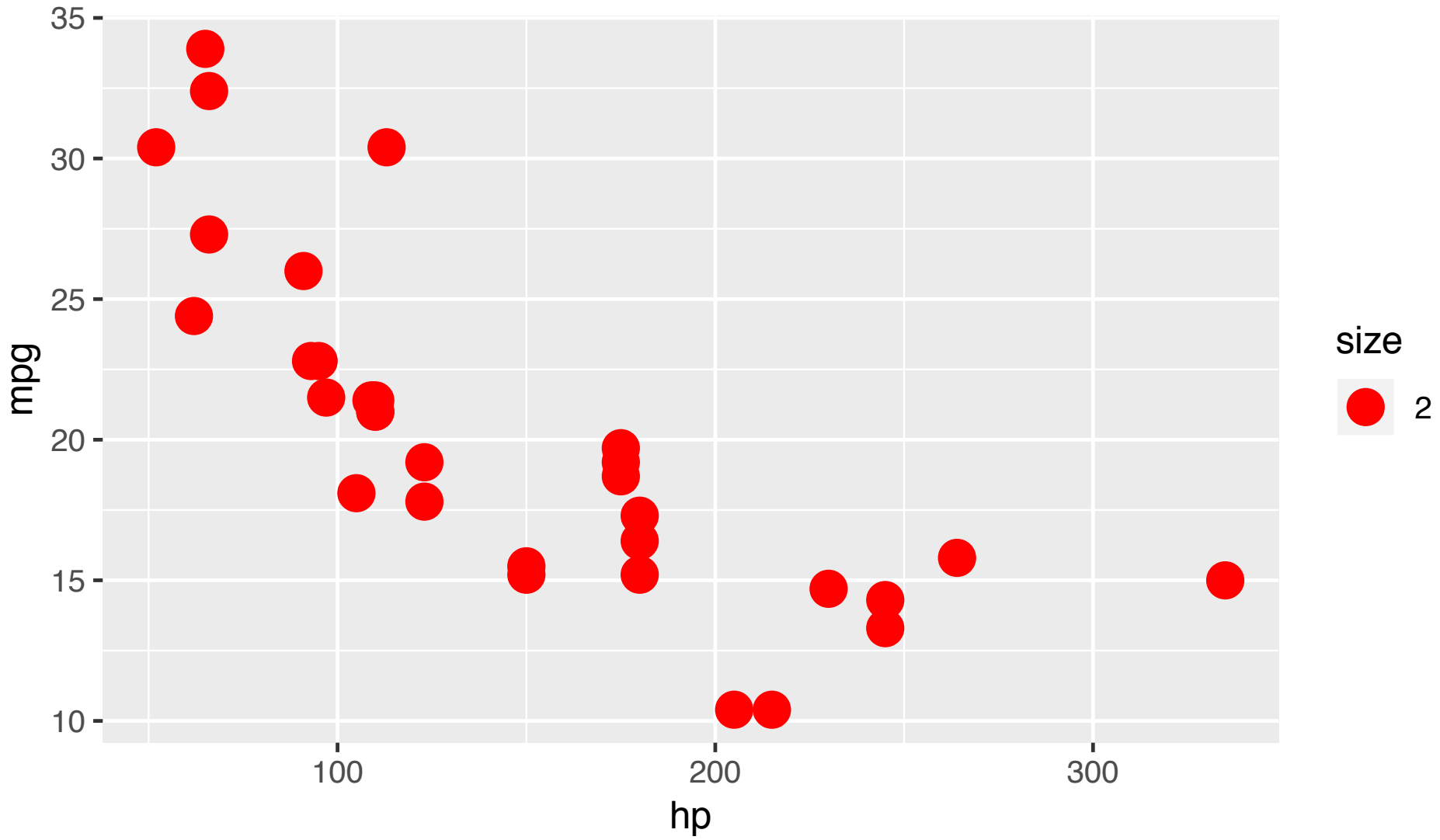
- Note that variables are mapped to aesthetics with the aes() function, while fixed aesthetics are set outside the aes() call
- This sometimes leads to confusion, as in this example:

```
ggplot(mtcars, aes(x = hp, y = mpg)) +
 geom_point(aes(size = 2), color="red")
```

Incorrect! 2 is not a variable

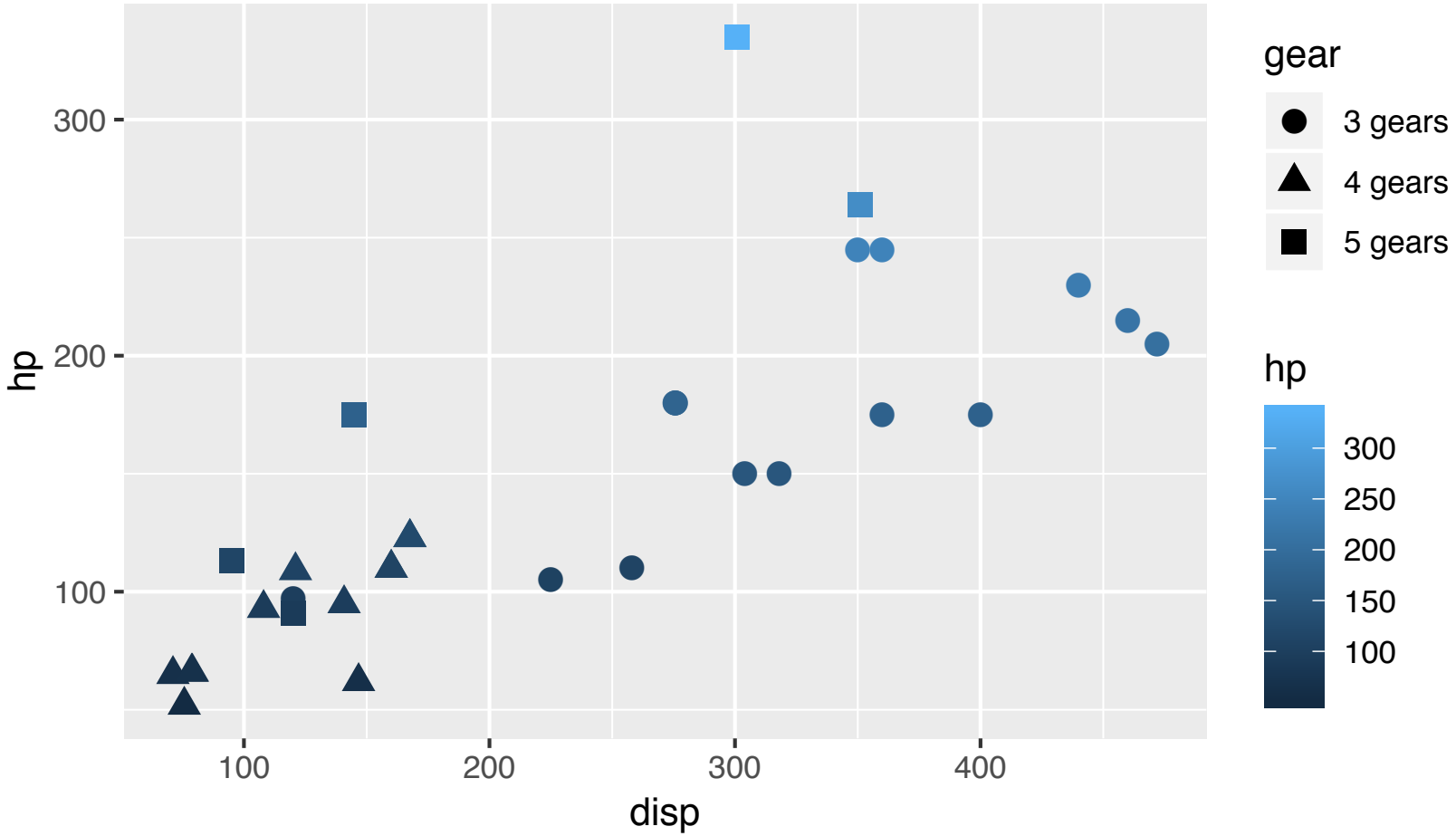
This is fine -- all points red

Aesthetic Mapping VS Assignment



Another example

```
ggplot(mtcars, aes(disp,hp, color = hp, shape = gear))+
  geom_point(size = 3)
```



Exercise 1

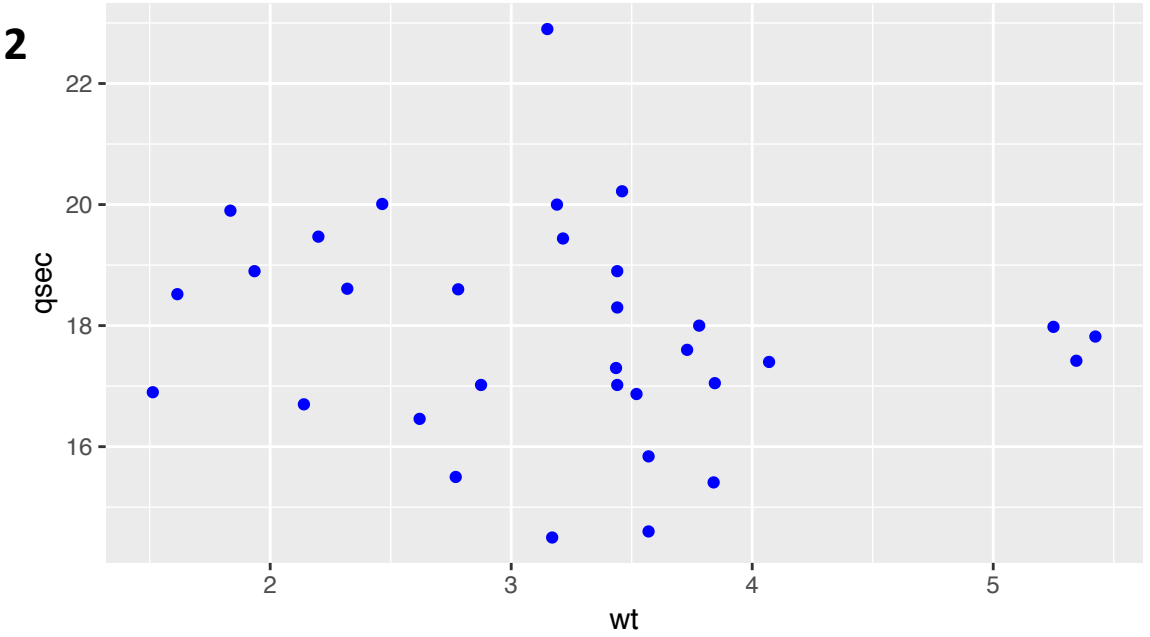
1. Create a scatter plot with Weight on the x axis and 1/4 mile time on the y axis using the mtcars dataset

2. Color all the points in the blue

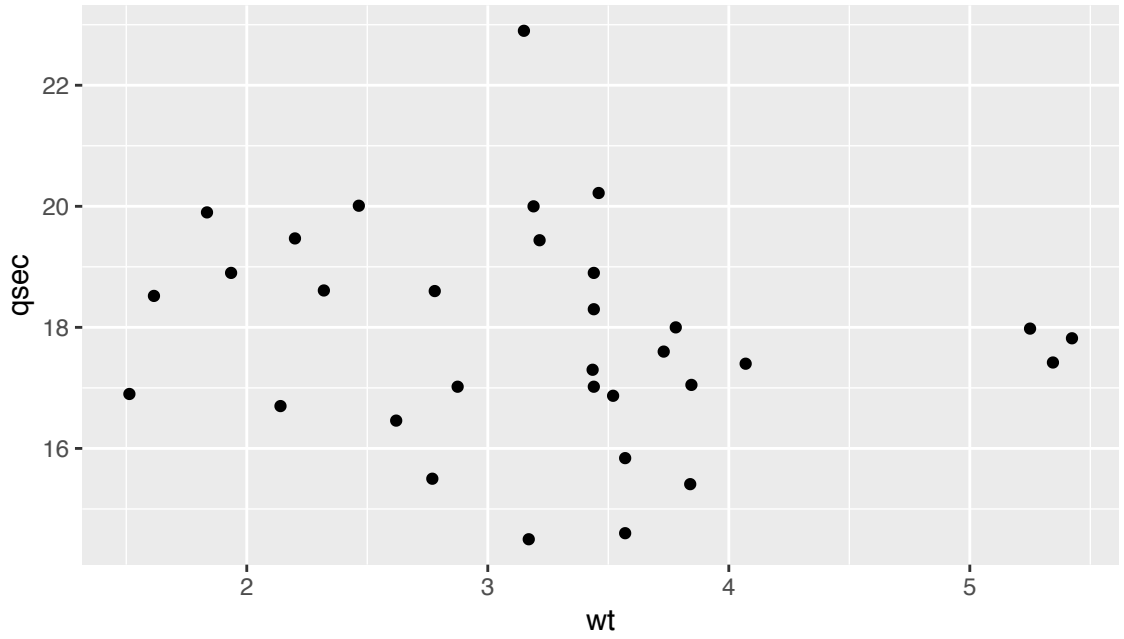
3. Color the points in the previous plot according to the number of cylinders

Exercise 1

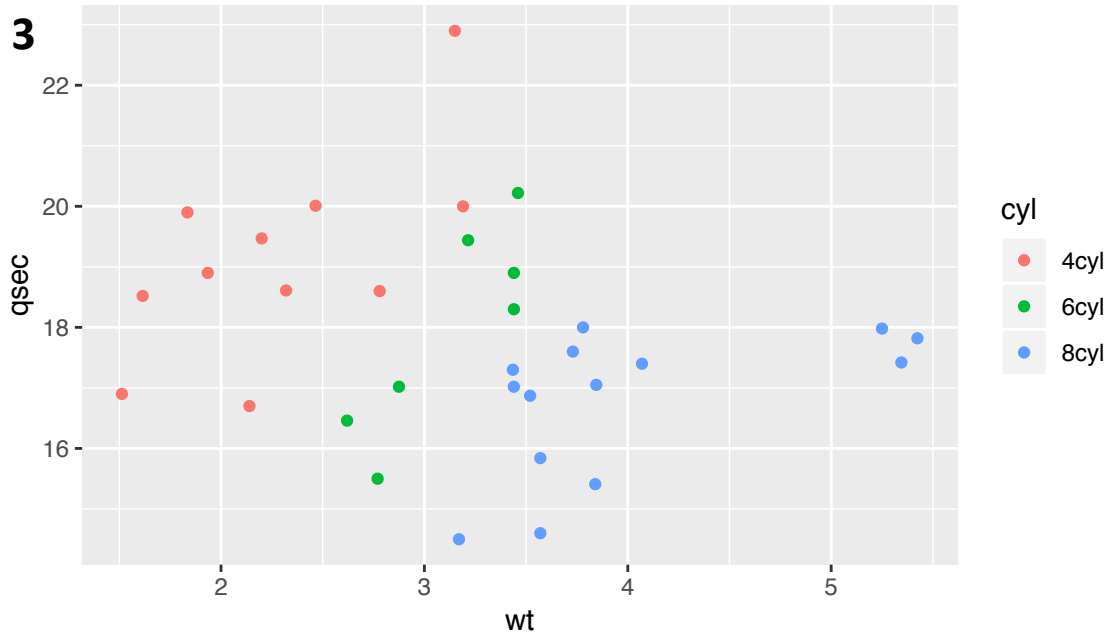
2



1



3



Exercise 1

1. Create a scatter plot with Weight on the x axis and 1/4 mile time on the y axis using the mtcars dataset

```
ggplot(mtcars, aes(wt, qsec))+ geom_point()
```

2. Color all the points in the blue

```
ggplot(mtcars, aes(wt, qsec))+ geom_point(color = "blue")
```

3. Color the points in the previous plot according to the number of cylinders

```
ggplot(mtcars, aes(wt, qsec))+
geom_point(aes(color = cyl))
```

Facetting (p4)

Faceting is ggplot2 parlance for small multiples. The idea is to create separate graphs for subsets of data.

ggplot2 offers two functions for creating small multiples:

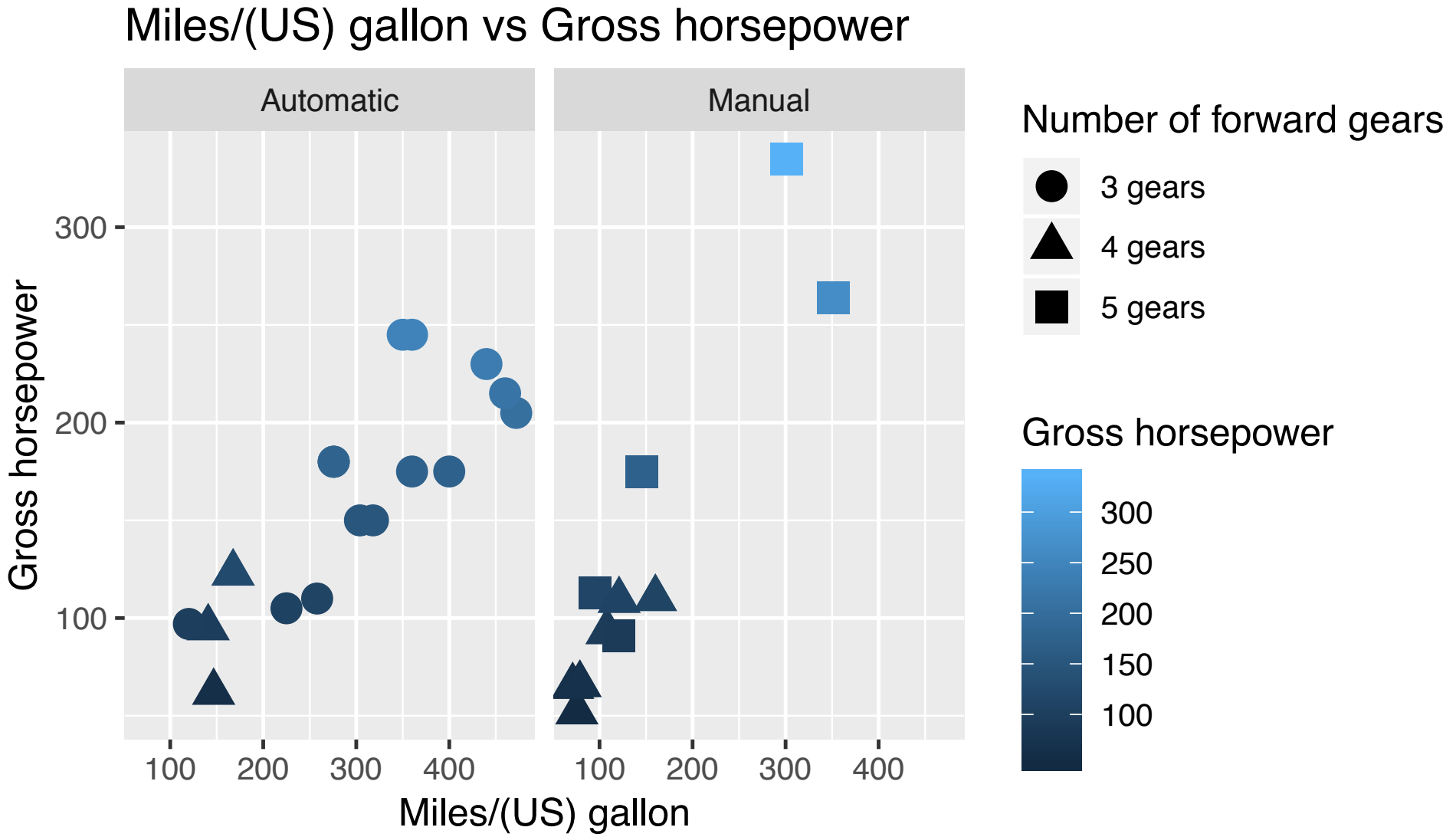
1. facet_wrap(): define subsets as the levels of a single grouping variable
2. facet_grid(): define subsets as the crossing of two grouping variables

Facilitates comparison among plots, not just of geoms within a plot

Facetting (p4)

```
p4 <- ggplot(mtcars, aes(disp,hp, color = hp, shape = gear))+
  geom_point(size = 4)+
  labs(title="Miles/(US) gallon vs Gross horsepower",
       x="Miles/(US) gallon", y="Gross horsepower",
       color = "Gross horsepower",
       shape = "Number of forward gears")+
  facet_wrap(~am)

p4
```

Facetting (p4)



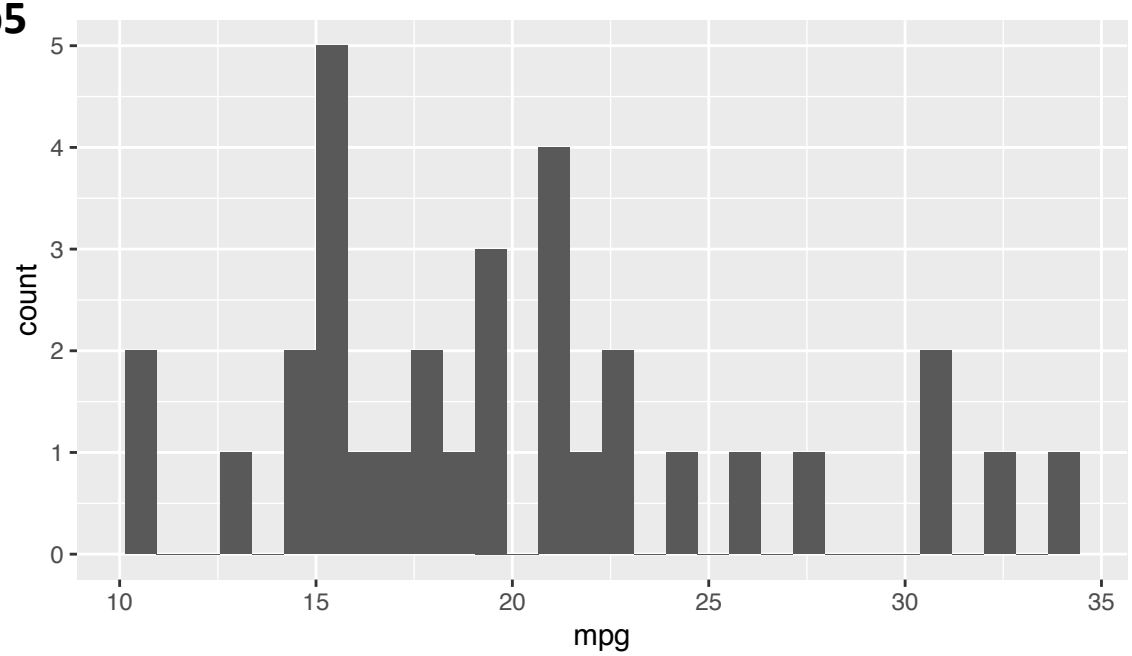
Statistical Transformations (p5, p6, p7)

- Some plot types (such as scatterplots) do not require transformations–each point is plotted at x and y coordinates equal to the original value
- Other plots, such as boxplots, histograms, prediction lines etc. require statistical transformations
  - For a boxplot the y values must be transformed to the median and 1.5(IQR)
  - For a smoother smother the y values must be transformed into predicted values
- Each geom has a default statistic, but these can be changed

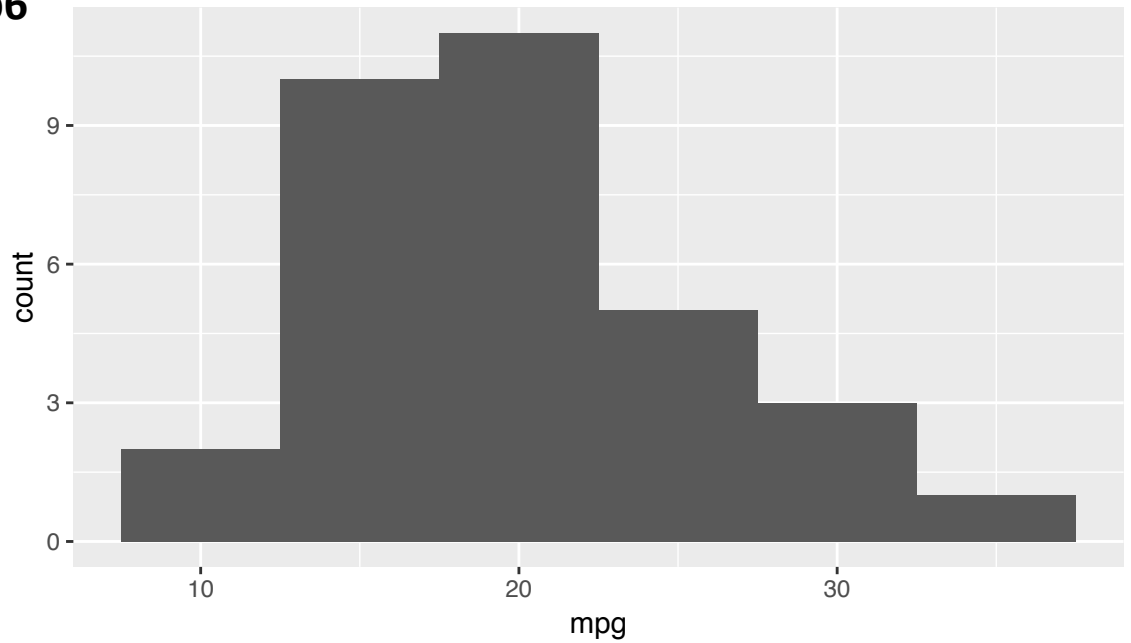
Statistical Transformations (p5, p6, p7)

```
p5 <- ggplot(mtcars, aes(x = mpg)) +
     geom_histogram()
p5

p6 <- ggplot(mtcars, aes(x = mpg)) +
     geom_histogram(stat = "bin", binwidth =5)
p6

p7 <- ggplot(mtcars, aes(x = gear, y = mpg))+
  geom_bar(stat = "identity", position = position_dodge())
p7
```

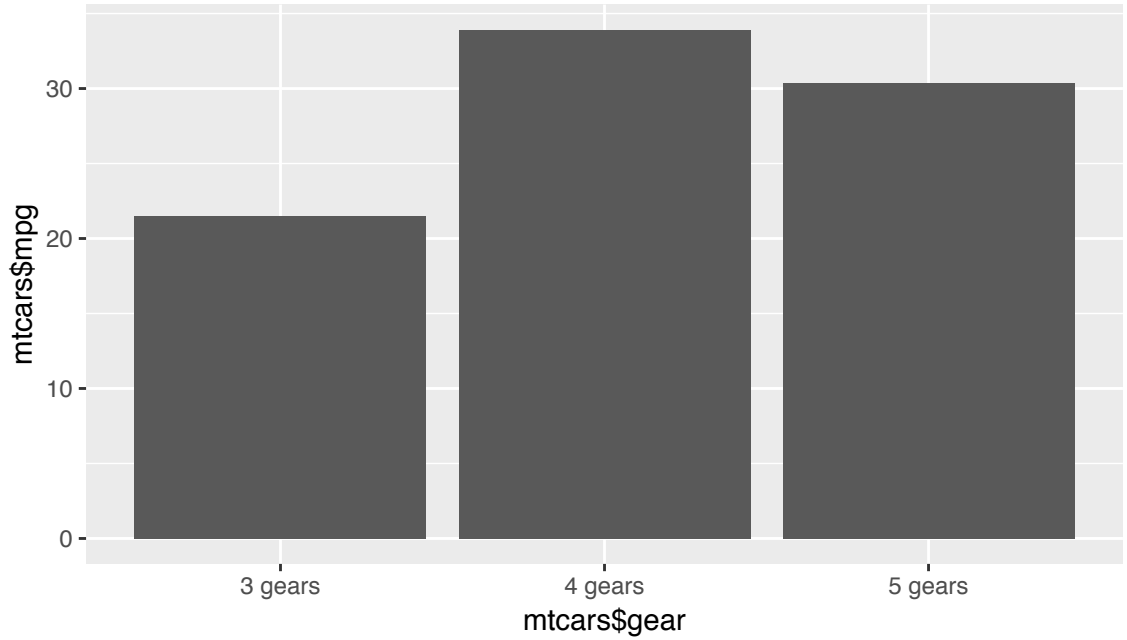
p5


count
mpg
0
1
2
3
4
5
10
15
20
25
30
35

p6


count
mpg
0
3
6
9
10
20
30

p7


mtcars$mpg
mtcars$gear
0
10
20
30
3 gears
4 gears
5 gears

Exercise 2

- Create a boxplot of mpg by gear
- Overlay points on top of the boxplot
- Create a scatter plot of weight vs. horsepower
- Overlay a **linear** regression line on top of the scatter plot (check geom_smooth)

Exercise 2

- Create a boxplot of mpg by gear

```
ggplot(mtcars, aes(gear, mpg))+ geom_boxplot()
```

- Overlay points on top of the boxplot

```
ggplot(mtcars, aes(gear, mpg))+ geom_boxplot()+ geom_point()
```

- Create a scatter plot of weight vs. horsepower

```
ggplot(mtcars, aes(wt, hp))+ geom_point()
```

- Overlay a **linear** regression line on top of the scatter plot (check geom_smooth)

```
ggplot(mtcars, aes(wt, hp))+ geom_point()+
geom_smooth(method = "lm")
```

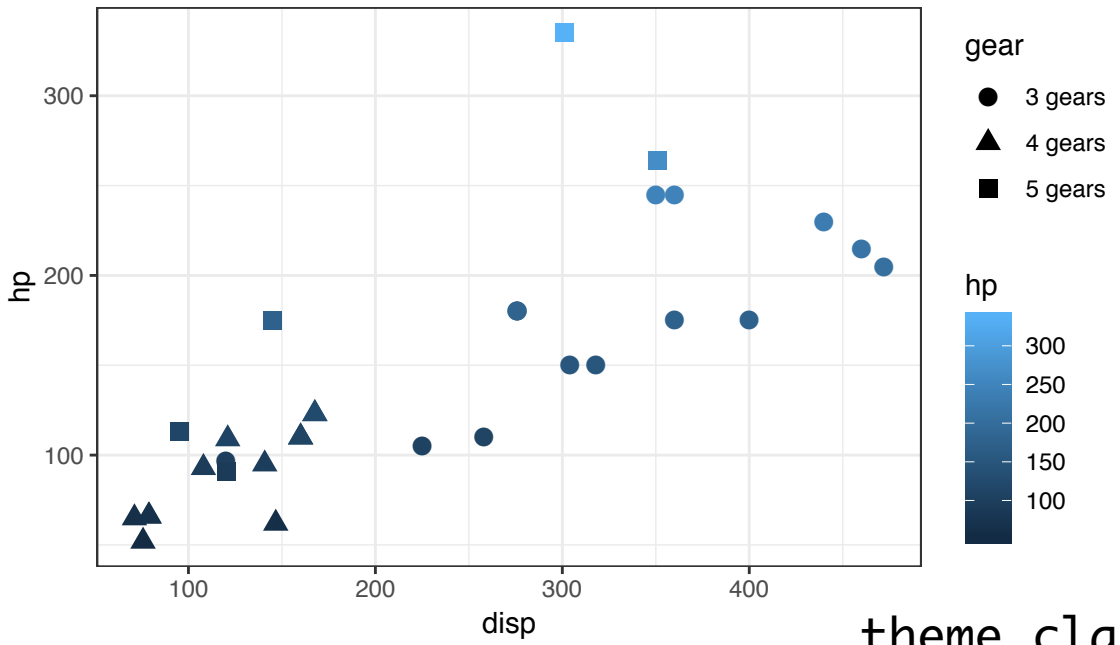
Theme (p8)

- The ggplot2 theme system handles non-data plot elements such as
  - Axis labels
  - Plot background
  - Facet label background
  - Legend
  - appearance
- Several built-in themes:
  - theme_gray() (default)
  - theme_bw()
  - theme_classic()
  - theme_dark()

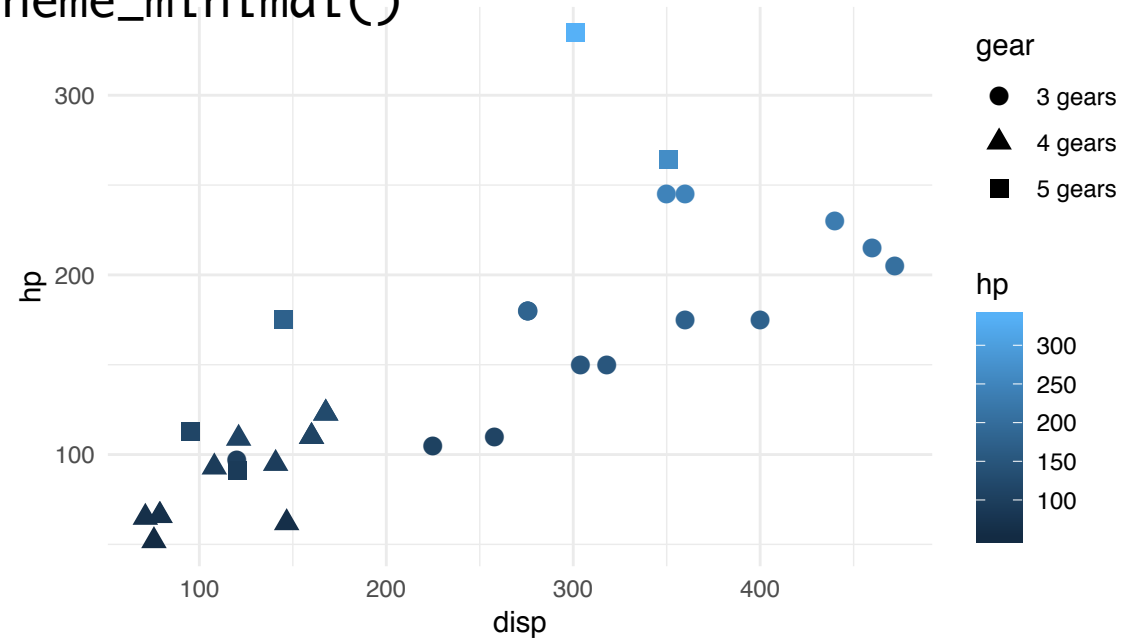
Theme (p8)

```
p3
p8 <- p3 + theme_bw()
p8 <- p3 + theme_minimal()
p8 <- p3 + theme_classic()
```

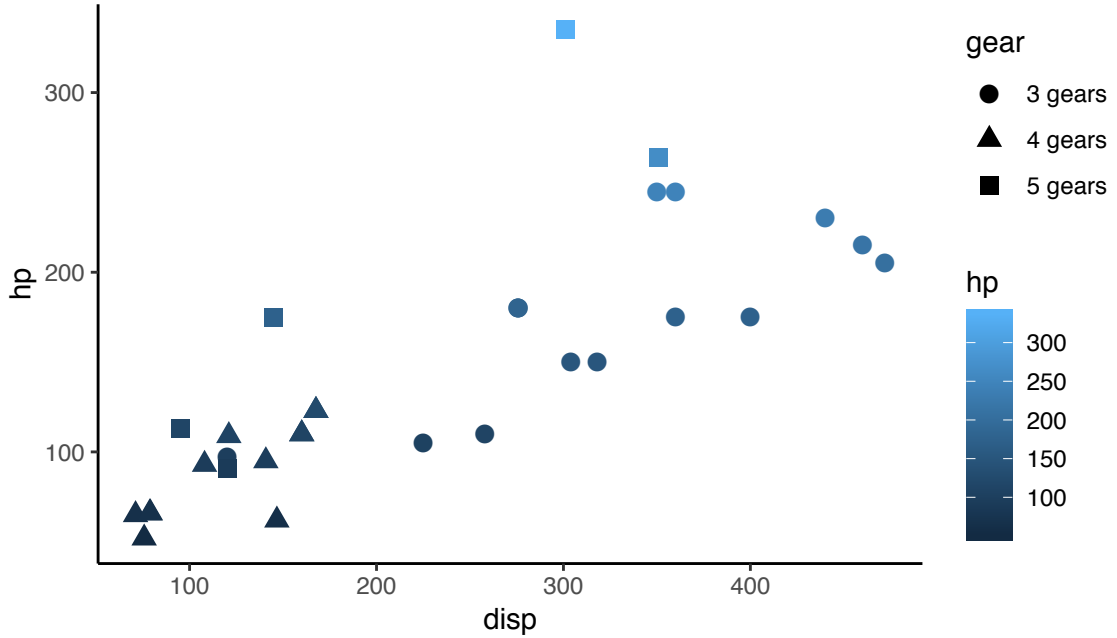
theme_bw()



theme_minimal()



theme_classic()

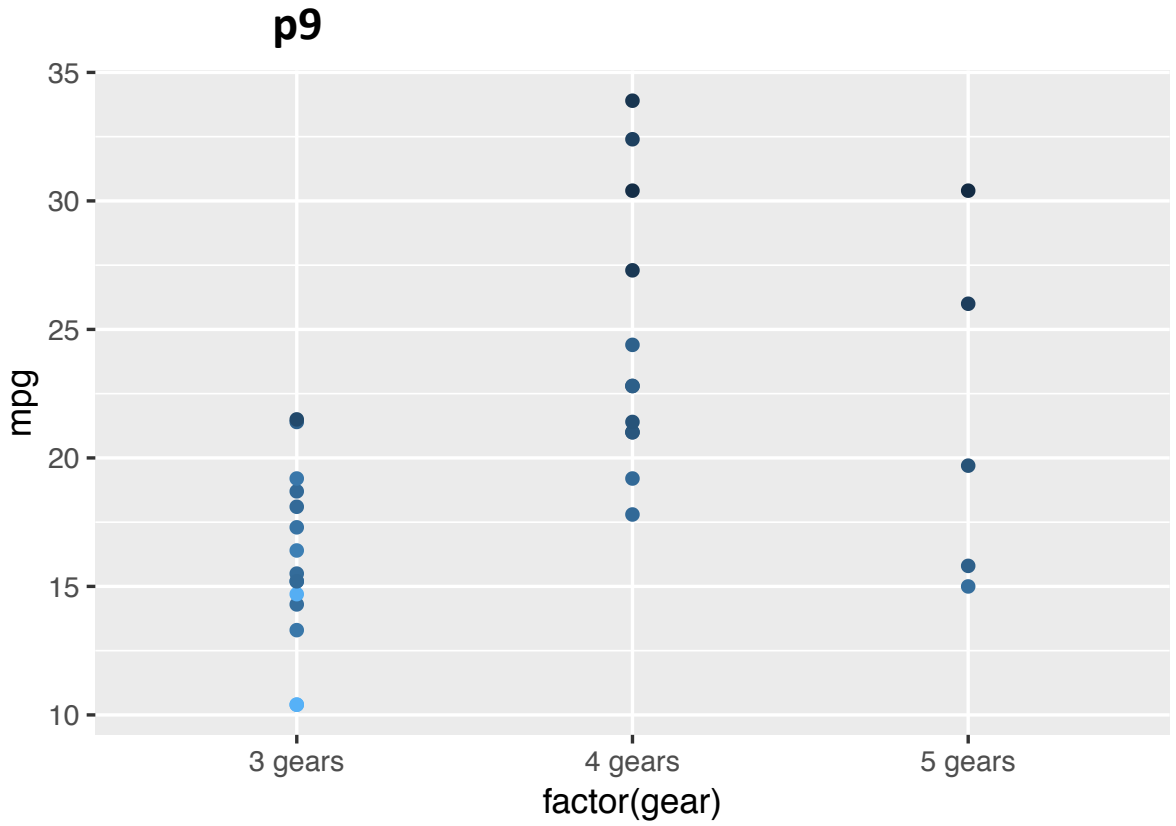


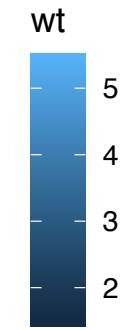
Scales (p9, p10)

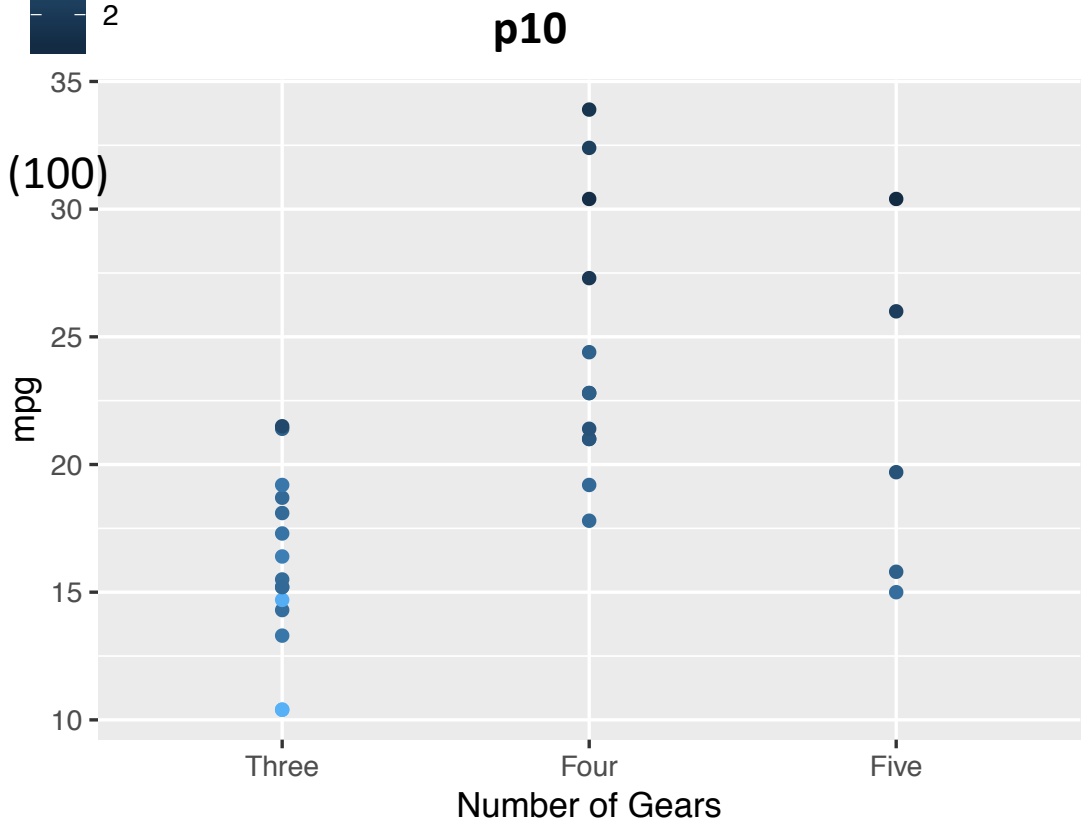
- In ggplot2 scales include
  - position
  - color and fill
  - size
  - Shape
  - line type
- Modified with scale_<aesthetic>_<type> (**92 !!**)
- Arguments :
  - name: the first argument gives the axis or legend title
  - limits: the minimum and maximum of the scale
  - breaks: the points along the scale where labels should appear
  - labels: the labels that appear at each break
  - Others…

Scales (p9, p10)

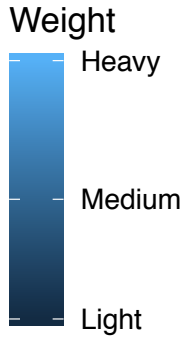
```
p9 <- ggplot(mtcars, aes(x = gear, y = mpg))+
  geom_point(aes(color = wt))
p9

p10 <- p9 +
  scale_x_discrete("Number of Gears",
                   breaks = c("3 gears", "4 gears", "5 gears"),
                   labels = c("Three", "Four", "Five"))+
  scale_color_continuous("Weight", breaks = with(mtcars,
c(min(wt), median(wt), max(wt))), labels = c("Light", "Medium",
"Heavy"))
p10
```


p9
35
30
25
20
15
10
mpg
3 gears
4 gears
5 gears
factor(gear)


wt
5
4
3
2


p10
35
30
25
20
15
10
mpg
Three
Four
Five
Number of Gears

(100)


Weight
Heavy
Medium
Light

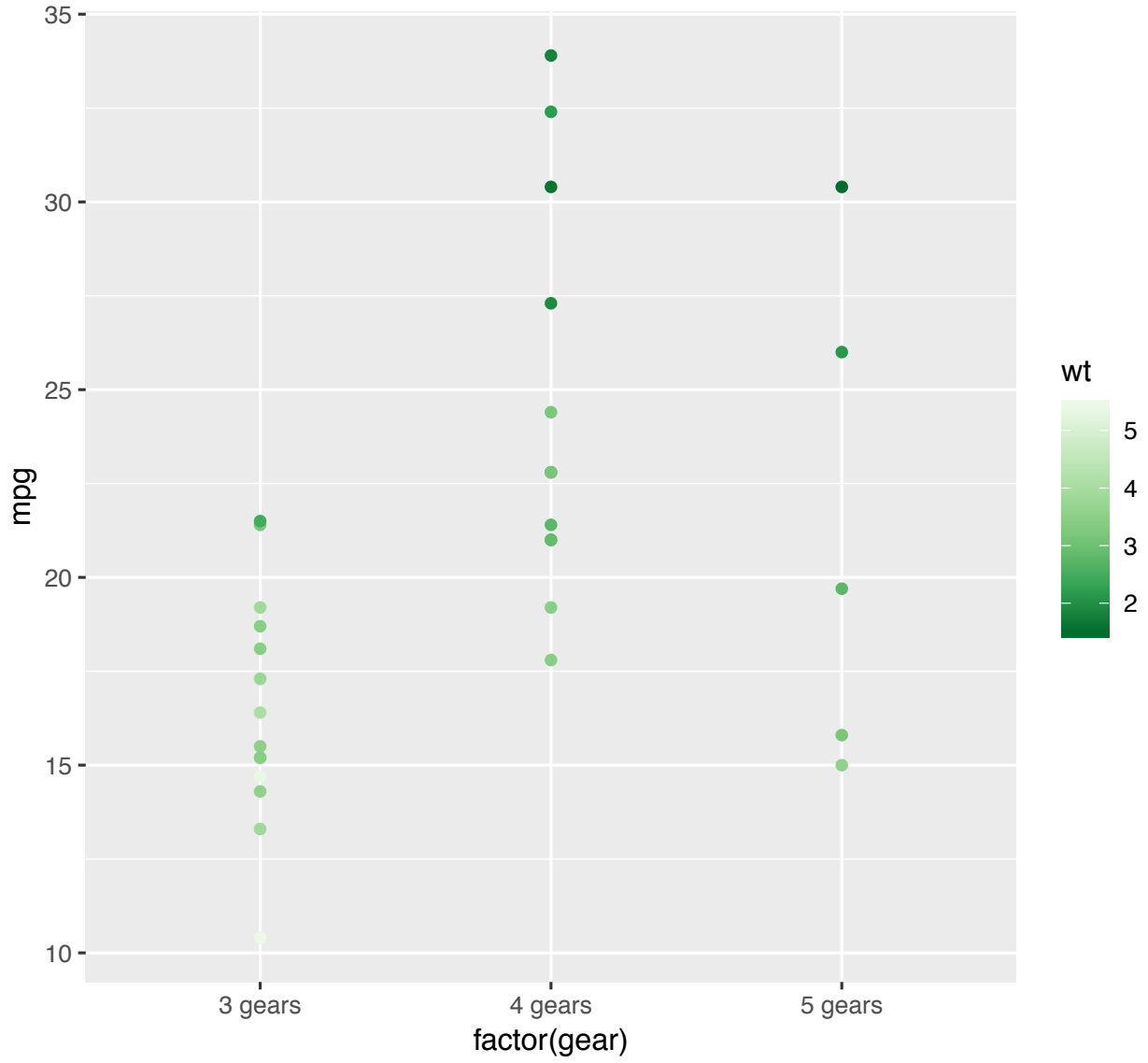
Scales (p11)

scale_colour_brewer – discrete data
scale_colour_distiller – continuous data

```
p11 <- p9 +
     scale_colour_distiller(palette = "Greens")
p11
```

Scales (p11)

```
p11 <- p9 +
scale_colour_distiller(palette
= "Greens")
p11
```

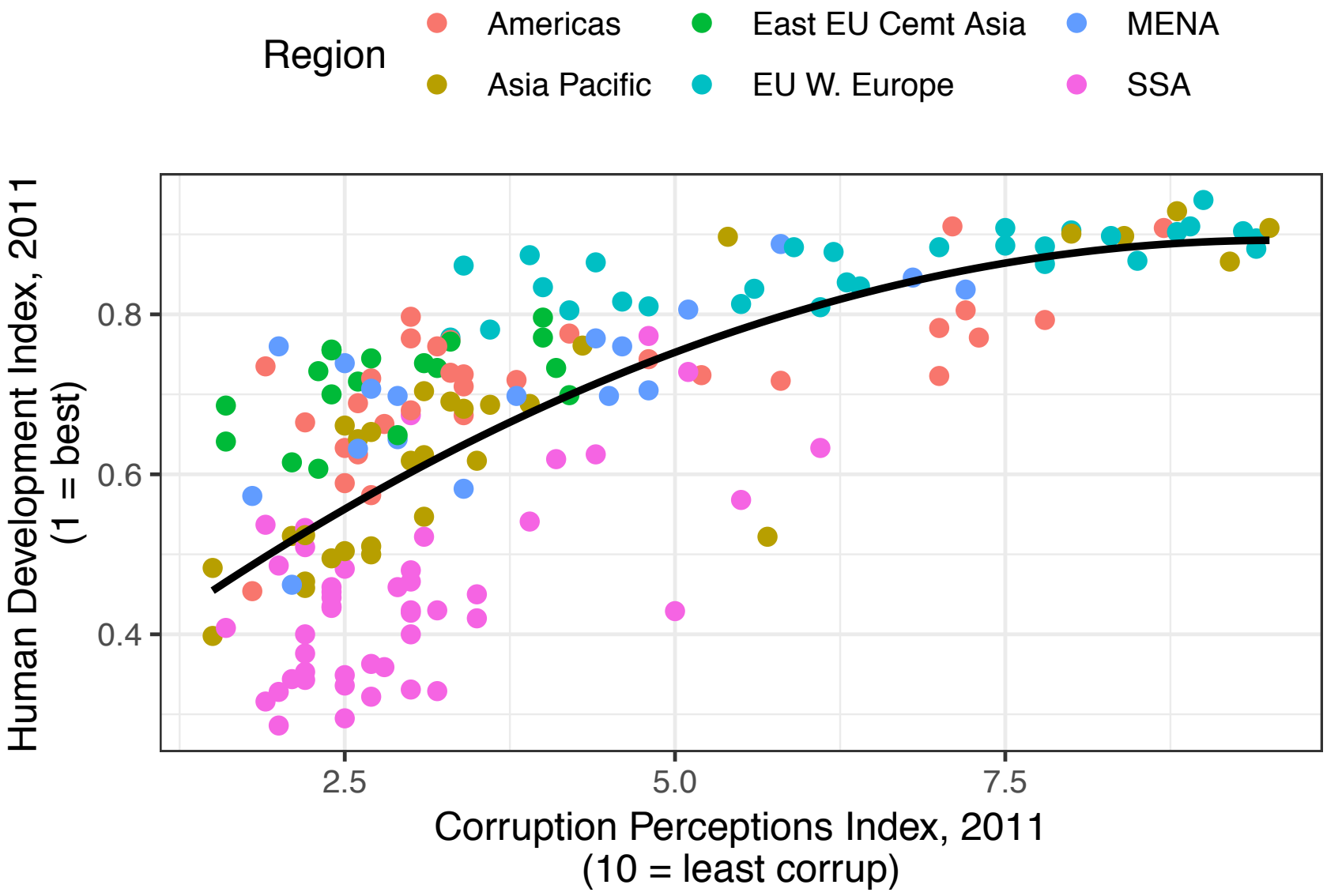


Exercise 3

From the file "EconomistData.csv", recreate this figure !!

The formula for the regression curve is :

y ~ poly(x, 2)



Exercise 3

```
ggplot(data, aes(x = …, y = …, color = …))+
  geom_point(…)+
  geom_smooth(method = "lm", ….)+
  theme_bw()+
  labs(x = …, y = …)
  theme(legend.position = "...",
legend.direction = "...")
```

Exercise 3 : Solution

```
ggplot(data, aes(x = CPI, y = HDI, color = Region))+
  geom_point(shape = 19, size = 2)+
  geom_smooth(method = "lm",
              color = "black",
              formula = y ~ poly(x, 2),
              se = FALSE)+
  theme_bw()+
  labs(x = "Corruption Perceptions Index, 2011\n(10 = least
corrup)", y = "Human Development Index, 2011\n(1 = best)")+
  theme(legend.position = "top", legend.direction =
"horizontal")
```

LABELS

script: Labels.R
folder: Day2_Plots/Dates_Labels

Fancy labels

- combine `paste()` and `expression()`
- typical expressions:
  - superindex (^): cells ml^{-1} → `expression(paste("cells ",ml^-1))`
  - subindex ([]): $\log_{10}$ Abundance → `expression(paste(log[10]," Abundance"))`
  - Greek letters (mu, pi): diameter µm → `expression(paste("diameter ",mu,"m"))`
  - Degree C (degree): Temperature (°C) → `expression(paste("Temperature ",degree~C))`
  - Italics (italic()): Synechococcus → `expression(italic("Synechococcus"))`
  - bold (bold()): **nmol** → `expression(bold ("nmol"))`

- Plot(1:10, xlab=expression(……..))

Exercises

- Write the following labels:
  - Oxygen (µmol kg^{-1}) mu ^
  - log_{10} b_{bp} (700) (m^{-1}) []
  - log_{10} chl *a* (mg m^{-3}) [] italic ^
  - Density (kg m^{-3}) ^
  - `plot(1:10, xlab=expression(paste(……)))`

```
expression(paste("cells ",ml^-1))
```

Exercises

- Write the following labels:
  - Oxygen (µmol kg^{-1})

```
expression(paste("Oxygen", " ", "(" ,mu, "mol", " ", kg^-1,")"))
```

  - $\log_{10} b_{bp}$ (700) (m^{-1})

```
expression(paste(log[10]," ", b[bp]," ", "(700)" , " ", "(" , m^-1, ")"))
```

  - $\log_{10}$ chl *a* (mg m^{-3})

```
expression(paste(log[10], " ", "chl ", italics(a), " ", "(" ,"mg", " ", m^-3, ")"))
```

  - Density (kg m^{-3})

```
expression(paste(Density , " ", "(","kg", " ", m^-3, ")"))
```

DATES AND TIMES

script: Dates.R
files: dates.txt
folder: Day2_Plots/Dates_Labels

Dates

- Special class format: Date
- Standard format for dates: (17 June 2011)
  - 2011-6-17
  - 2011/06/17

```
as.Date("2011-6-17")
as.Date("2011/6/17")
```

```
[1] "2011-06-17"
[1] "2011-06-17"
```

- For other date structures, we need to provide the format information

```
as.Date("6/17/2011")
```

```
Error in charToDate(x) :
  character string is not in a
standard unambiguous format
```

Dates

- For other date structures, we need to provide the format information

```
as.Date("6/17/2011","%m/%d/%Y")
as.Date("17/6/2011","%d/%m/%Y")
as.Date("17/6/11","%d/%m/%y")
as.Date(”17JUN-11","%d%b-%y")
as.Date("June 17, 2011","%B %d,
 %Y")
```

Code	Value
%d	Day of the month (decimal number)
%m	Month (decimal number)
%b	Month (abbreviated)
%B	Month (full name)
%y	Year (2 digit)
%Y	Year (4 digit)

Dates and times

- Special class format: **POSIXct** and POSIXlt
- Standard input:
  - 2005/6/24
  - 2005-06-24 11:25
  - 2005/6/24 11:25:00

```
as.POSIXct("2005/6/24 11:25:00")
```

Dates and times

- For other date/time formats:

```
strptime("24/Jun/2005:07:51:00",format="%d/%b/%Y:%H:%M:%S")
```

```
as.POSIXct(strptime("24/Jun/2005:07:51:00",format="%d/%b/%Y:%H:%M:%S"))
```

- specify time zone

```
as.POSIXct("2005/06/24 11:25:00",tz= "Asia/Riyadh")
```

POSIXct and POSIXlt formats

Code	Meaning	Code	Meaning
%a	Abbreviated weekday	%A	Full weekday
%b	Abbreviated month	%B	Full month
%c	Locale-specific date and time	%d	Decimal date
%H	Decimal hours (24 hour)	%I	Decimal hours (12 hour)
%j	Decimal day of the year	%m	Decimal month
%M	Decimal minute	%p	Locale-specific AM/PM
%S	Decimal second	%U	Decimal week of the year (starting on Sunday)
%w	Decimal Weekday (0=Sunday)	%W	Decimal week of the year (starting on Monday)
%x	Locale-specific Date	%X	Locale-specific Time
%y	2-digit year	%Y	4-digit year
%z	Offset from GMT	%Z	Time zone (character)

Operations with date/time objects

- Get weekdays from dates

```
weekdays(as.Date("28-Jan-2019", format="%d-%b-%Y"))
```

```
[1] Monday
```

- Calculate the difference in time units between 2 dates

```
difftime("2019-01-20", "2017-07-07", units="weeks")
```

```
difftime("2019-01-20", "2017-07-07", units="days")
```

- Extract values from the date/time object

```
format(as.POSIXct("2019-01-20"),"%Y")
```

```
format(as.POSIXct("2019-01-20"),"%B")
```

Exercise 1

1. Read the *dates.txt* table and look at the data structure (read.table())

2. Look at the format of the column **date1**

3. Create a new column called **Date1** that has the **date1** column in POSIXct format

4. * to create a new column:

```
dates$Date1 <-
```

Exercise 1

1. Read the *dates.txt* table and look at the data structure (read.table())

```
dates<-read.table("dates.txt", header=T)
```

```
head(dates)
```

2. Look at the format of the column **date1**

```
dates$date1
```

3. Create a new column called **Date1** that has the **date1** column in POSIXct format

```
dates$Date1 <- as.POSIXct(strptime(dates$date1,
format="%m/%d/%y"))
```

```
dates$Date1
```

Exercise 2

1. Look at the format of the column **date2**

2. Create a new column called **Date2** that has the **date2** column in POSIXct format

Exercise 2

1. Look at the format of the column **date2**

```
head(dates)
```

```
dates$date2
```

2. Create a new column called **Date2** that has the **date2** column in POSIXct format

```
dates$Date2 <- as.POSIXct(strptime(dates$date1,
format="%m.%d.%Y"))
```

```
dates$Date2
```

Exercise 3

1. Look at the format of the columns **date3_jd** and **date3_year**

2. Create a new column called **date3_jdy** that combines both columns (date3_jd, date3_year). Use the function paste()

3. Create a new column called Date3 with the date3_jdy column in POSIXct format

Exercise 3

1. Look at the format of the columns **date3_jd** and **date3_year**

```
head(dates)
```

2. Create a new column called **date3_jdy** that combines both columns (date3_jd, date3_year). Use the function paste()

```
dates$date3_jdy <- paste(dates$date3_jd, dates$date3_year)
```

3. Create a new column called Date3 with the date3_jdy column in POSIXct format

```
dates$Date3 <- as.POSIXct(strptime(dates$date3_jdy,
format="%j %Y"))
```

```
dates$Date3
```

Section plot

script: Section_plots_Exercise.R
folder: Day2_Plots/SectionPlot

”Real Data” : What you need to know

- You have a table with 20 CTD profile one after the other, determined by the column profile
- You need to use/load the packages : ggplot2, scales and Rcolorbrewer
- The date column will need to be converted in a format usable by ggplot

- The goal : Make the 3 plots that are in the exercise folder

A little help

You may need the following command of ggplot2

- scale_x_datetime
- scale_colour_gradientn
- scale_y_continuous
- brewer_pal

A reminder : ?scale_x_datetime gives you an help on the command

Don’t forget to convert the date column to a good format

Solution

```
dat$date <- as.POSIXct(strptime(dat$date, format="%m/%d/%Y"))

str(dat$date)
```

Solution

```
ggplot(dat)+
  geom_point(aes(x = date, y = -depth, colour=salinity), size=2)+

  labs(colour="Salinity")+

  theme_bw()
```

Solution

```
ggplot(dat)+
  geom_point(aes(x = date, y = -depth, colour=salinity), size=2)+

  scale_colour_gradientn(colours=brewer_pal(palette = "OrRd")(9))+

  scale_y_continuous(limits=c(-800,0),"Depth (m)")+

  labs(colour="Salinity")+

  theme_bw()+

  scale_x_datetime("Time",breaks = date_breaks("1 week"),date_labels
= "%d/%m")
```

Solution

```
ggplot(dat)+
  geom_point(aes(x = profile, y = -depth, colour=chlorophyll), size=2)+

  scale_colour_gradientn(colours=brewer_pal(palette="PiYG")(9))+

  scale_y_continuous(limits=c(-1000,0),"Depth (m)")+

  labs(colour=expression(paste("chla", " ", "(" ,"mg", " ", m^-3, ")")))+

  theme_bw()
```