Useful R codes for ggplot2 & data analysis

Ver. 1

10/20/2021

Rie Sadohara

```
Alt + Shift + arrow ↑↓ … duplicate the current line
Alt + arrow ↑↓         … move the current line
Ctrl + D               … delete the current line
Ctrl + Shift + C       … comment/uncomment the current line
Ctrl + Shift + M       … insert the pipe symbol (%>%) for dplyr
```

Preface

This document assumes you are an intermediate user of RStudio.

This document aims to remind you of how to do specific operations, not necessarily to explain every single line of code.

The codes in this document assume that you have loaded the ggplot2 and dplyr packages and the example dataset ‘mpg’ in ggplot2, and converted some variables in mpg into factors.

To the best of my knowledge the codes are functional, but please know that R and/or package updates and other configuration issues may present problems. Googling your error messages will usually help with troubleshooting. I would be happy to answer your questions, too.

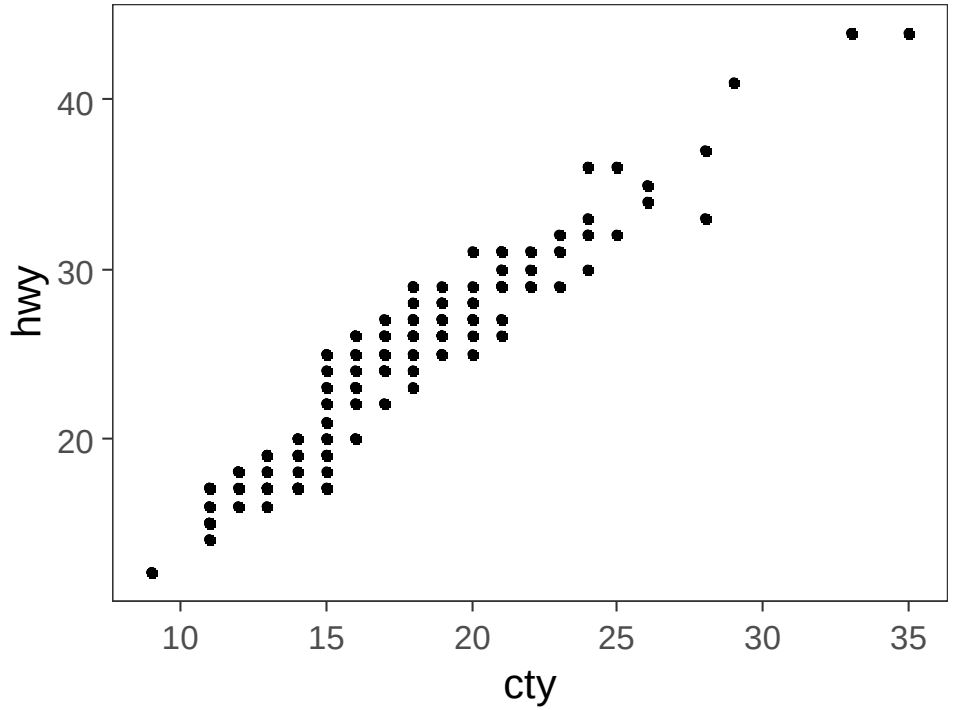
Some codes need additional packages, which are specified in each slide as necessary.

The codes in this presentation are in ‘Useful_R_gg_da_v1.R’.

‘=’ and ‘<-’ means the same

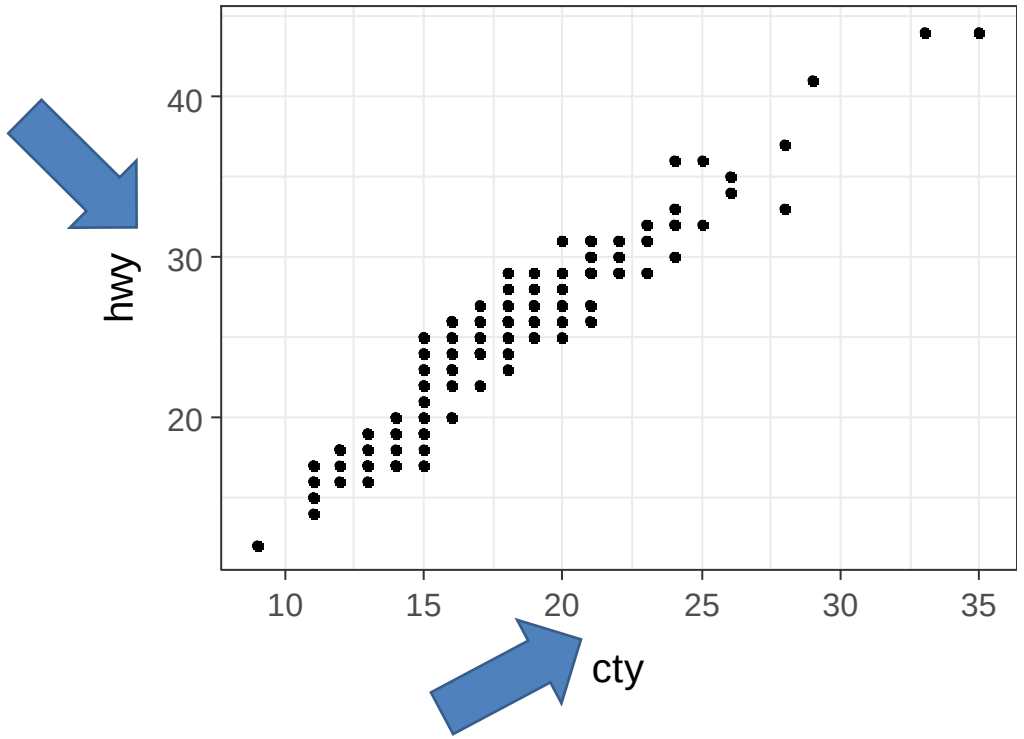
Useful codes for ggplot2

```
mpg %>%
  ggplot(aes( x=cty, y=hwy )) +
  geom_point() +
  theme_bw(base_size=15) +
  theme( panel.grid.major = element_blank(), panel.grid.minor = element_blank() )
```



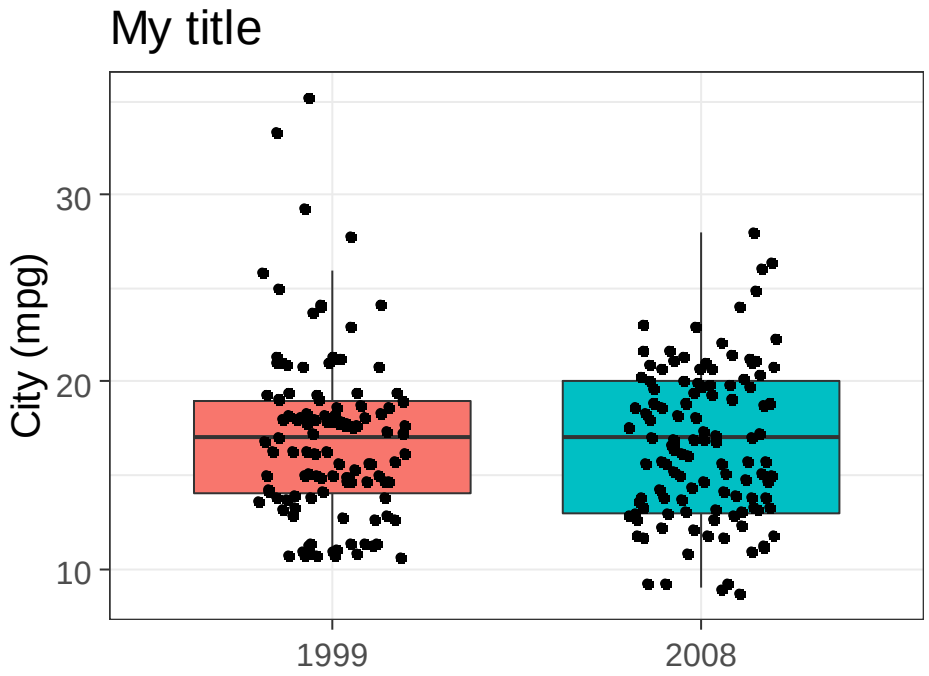
Remove the inner grid of your plot

```
mpg %>%
  ggplot(aes( x=cty, y=hwy )) +
  geom_point() +
  theme_bw(base_size=15) +
  theme(axis.title.x = element_text(margin=margin(t = 10, r = 0, b = 0, l = 0) ) ) +
  theme(axis.title.y = element_text(margin=margin(t = 0, r = 10, b = 0, l = 0) ) )
```



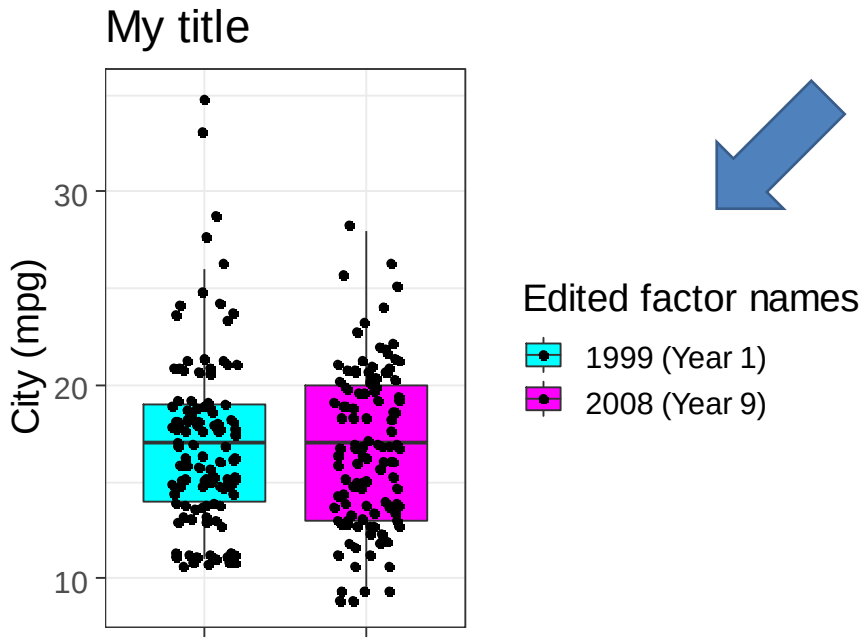
Increase margin between axis title and axis.

```
mpg %>%
  filter( !is.na(year_f) ) %>%  # Pick up only those that are not NA as X axis.
  ggplot( aes(x=    year_f ,
              y=    cty ,
              fill= year_f ) ) +
  theme_bw(base_size = 15) +
  geom_boxplot(outlier.shape = NA) +  # Hide outliers if using jitter to avoid duplicated dots.
  geom_jitter(color="black", width=0.2) +
  ggtitle("My title" ) +
  theme(legend.position = "none") +
  labs(x=element_blank(), y="City (mpg)")
```



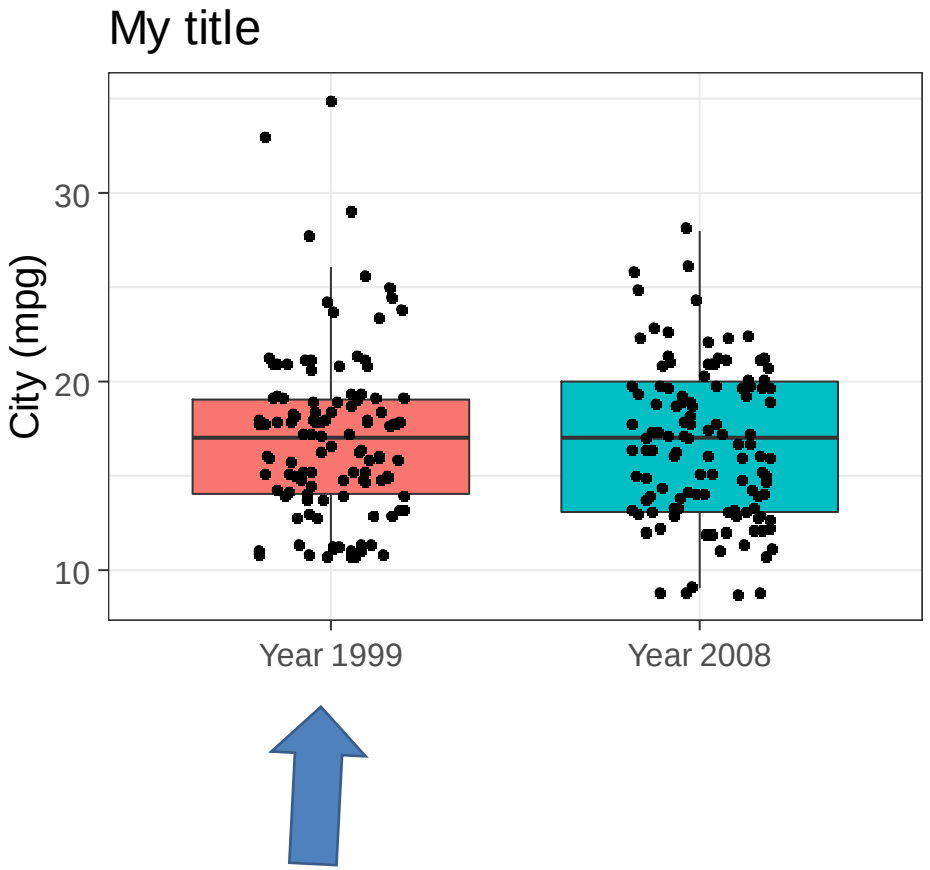
Hide the entire legend

```
mpg %>%
  filter( !is.na(year_f) ) %>%  # Pick up only values that are not NA.
  ggplot( aes(x=    year_f ,
              y=    cty ,
              fill= year_f ) ) +
  theme_bw(base_size = 15) +
  geom_boxplot(outlier.shape = NA) +  # Hide outliers if using jitter to avoid duplicated dots.
  geom_jitter(color="black", width=0.2) +
  ggtitle("My title" ) +
  labs(x=element_blank(), y="City (mpg)", fill = "Edited factor names") +
  theme(axis.text.x = element_blank()) +
  scale_fill_manual(values = colornames,
                    labels = c("1999 (Year 1)", "2008 (Year 9)") )
```



Change the legend title and text.

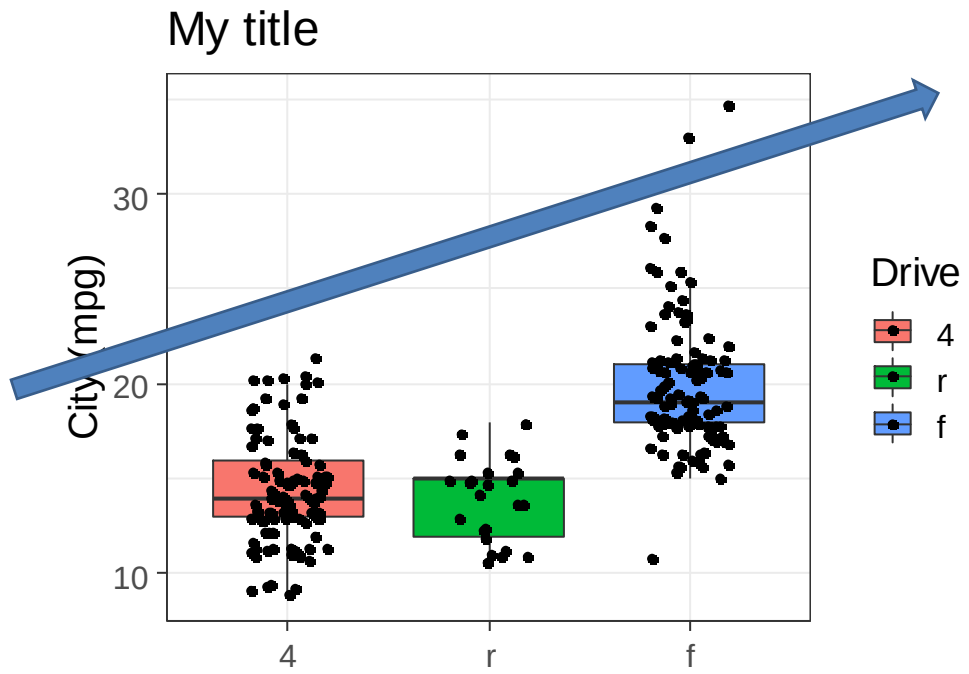
```
mpg %>%
 filter( !is.na(year_f) ) %>%  # Pick up only those that are not NA as X axis.
 ggplot( aes(x=    year_f ,
             y=    cty ,
             fill= year_f ) ) +
 theme_bw(base_size = 15) +
 geom_boxplot(outlier.shape = NA) +  # Hide outliers if using jitter to avoid duplicated dots.
 geom_jitter(color="black", width=0.2) +
 ggtitle("My title" ) +
 theme(legend.position = "none") +
 labs(x=element_blank(), y="City (mpg)") +
 scale_x_discrete(labels = c("Year 1999", "Year 2008"))
```



Change the factor names on the x axis of the plot.

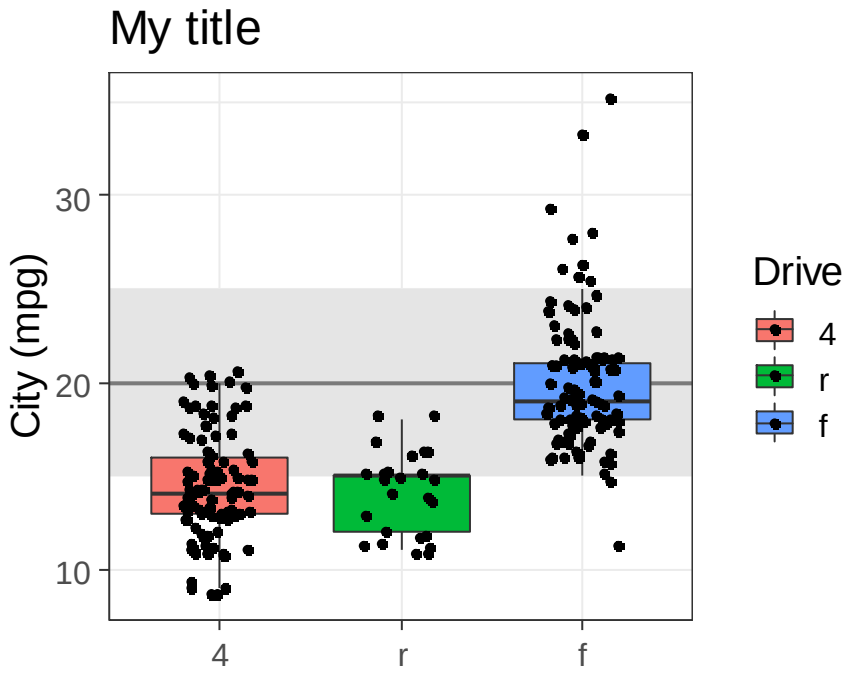
```
mpg %>%
    ggplot( aes(x= fct_reorder(drv_f, cty, .desc = F),
                y = cty,
                fill = fct_reorder(drv_f, cty, .desc = F) ) ) +
    theme_bw(base_size = 15) +
    geom_boxplot(outlier.shape = NA) +
    geom_jitter(color="black", width=0.2) +
    ggtitle("My title" ) +
    labs(x=element_blank(), y="City (mpg)", fill = "Drive")
```

Need ‘forcats’ package



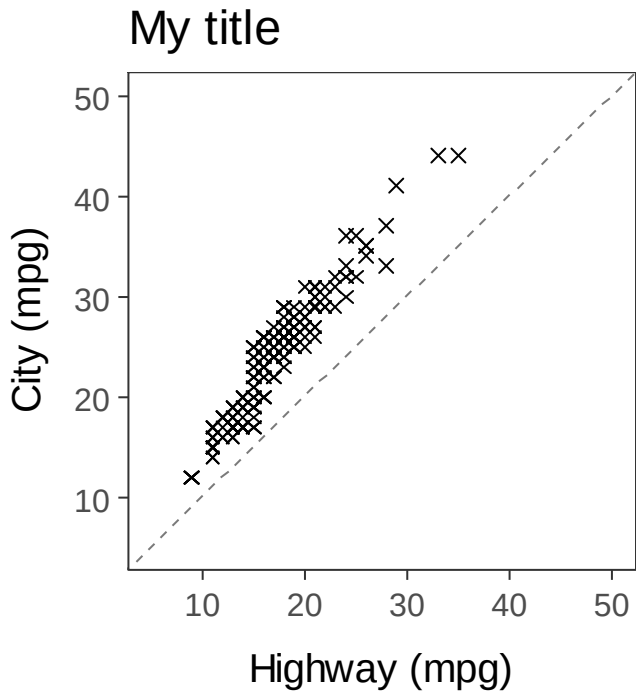
Sort by median

```
mpg %>%
   ggplot( aes(x= fct_reorder(drv_f, cty, .desc = F),
               y = cty,
               fill = fct_reorder(drv_f, cty, .desc = F))) +
   geom_rect( aes(xmin = -Inf, xmax = Inf, ymin = 15, ymax = 25), fill = "grey90” ) +
   geom_hline(yintercept = 20, color="grey48", size=1, linetype=1) +
   theme_bw(base_size = 15) +
   geom_boxplot(outlier.shape = NA) +  # Hide outliers if using jitter to avoid duplicated dots.
   geom_jitter(color="black", width=0.2) +
   ggtitle("My title" ) +
   labs(x=element_blank(), y="City (mpg)", fill = "Drive")
```



Add a rectangular shape and a horizontal line to your chart

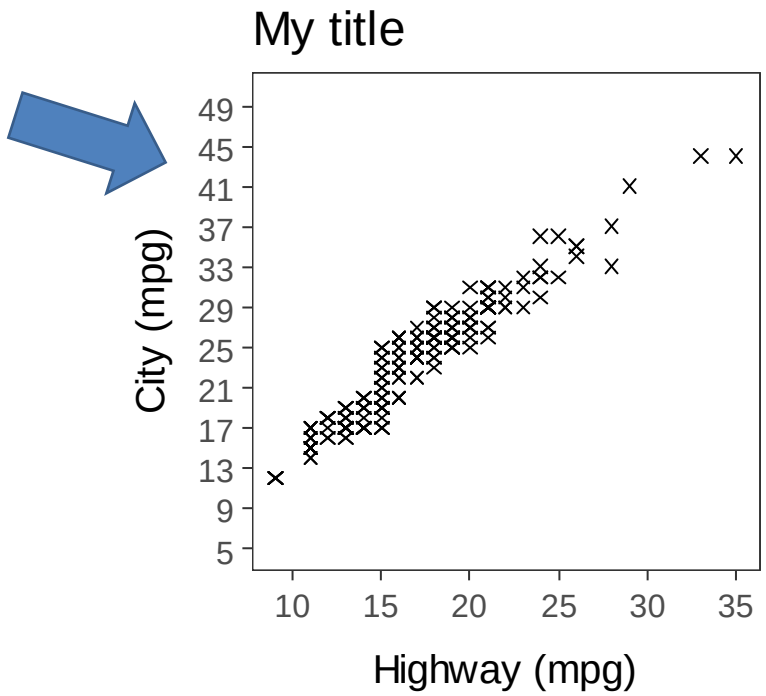
```
mpg %>%
   ggplot( aes(x = cty, y = hwy) ) +
   geom_abline(slope = 1, intercept = 0, color="grey48", linetype = 2) +
   theme_bw(base_size = 15) +
   geom_point(size = 2, na.rm = F, shape = 4) +
   ggtitle("My title" ) +
   labs(x = "Highway (mpg)", y="City (mpg)") +
   xlim(5, 50) + ylim(5, 50) +
   theme( panel.grid.major = element_blank(), panel.grid.minor = element_blank()) +
   theme(axis.title.x = element_text(margin = margin(t = 10, r = 0, b = 0, l = 0) ) ) +
   theme(axis.title.y = element_text(margin = margin(t = 0, r = 10, b = 0, l = 0) ) )
```



Add a y=x reference line in your chart

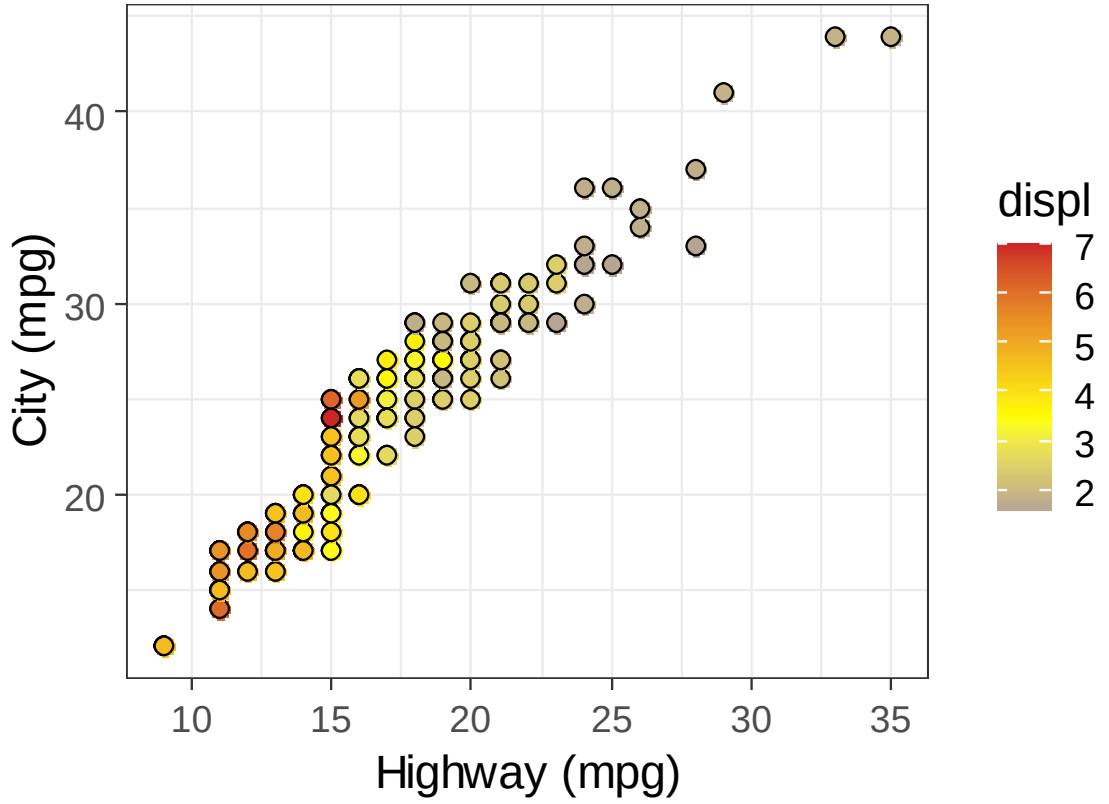
```
mpg %>%
    ggplot( aes(x = cty, y = hwy) ) +
    theme_bw(base_size = 15) +
    geom_point(size = 2, na.rm = F, shape = 4) +
    ggtitle("My title" ) +
    labs(x = "Highway (mpg)", y="City (mpg)") +
    scale_y_continuous(limits = c(5, 50), breaks= round( seq(from=5, to=50, by=4), 0 ) ) +
    theme( panel.grid.major = element_blank(), panel.grid.minor = element_blank()) +
    theme(axis.title.x = element_text(margin = margin(t = 10, r = 0, b = 0, l = 0) ) ) +
    theme(axis.title.y = element_text(margin = margin(t = 0, r = 10, b = 0, l = 0) ) )
```

Y axis goes from 5 to 50, and the interval to show is from 5, 5+4x1, 5+4x2, 5+4x3, …, 50 (or the cloest to 50).



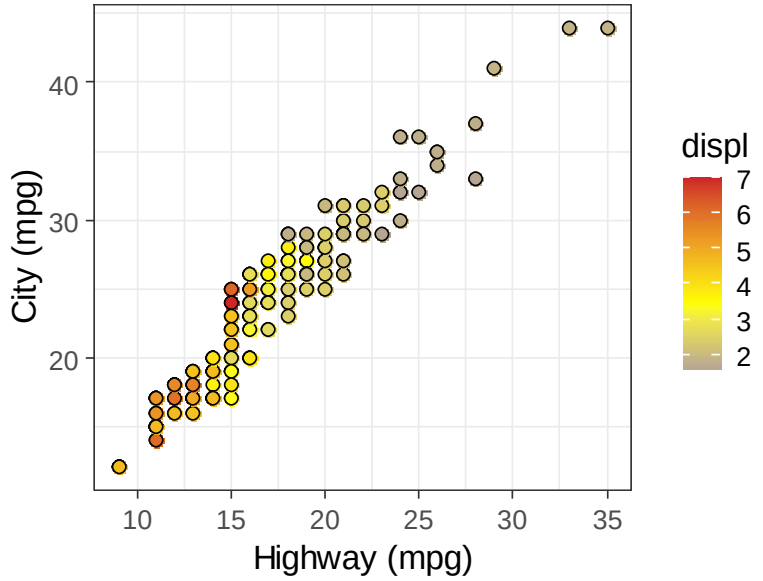
Set *axis limits* with custom breaks

```
# Calculate the mean value of the variable to be used for coloring
mid_displ = mean(mpg$displ, na.rm=T)

mpg %>%
  ggplot( aes(x=cty, y=hwy) ) +
  geom_point(size=3, na.rm=T, shape=21, aes(fill=displ), color="black") +
  scale_fill_gradient2(midpoint=mid_displ, low="royalblue3", mid="yellow", high="firebrick3" ) +
  theme_bw(base_size = 15) +
  labs(x="Highway (mpg)", y="City (mpg)")
```

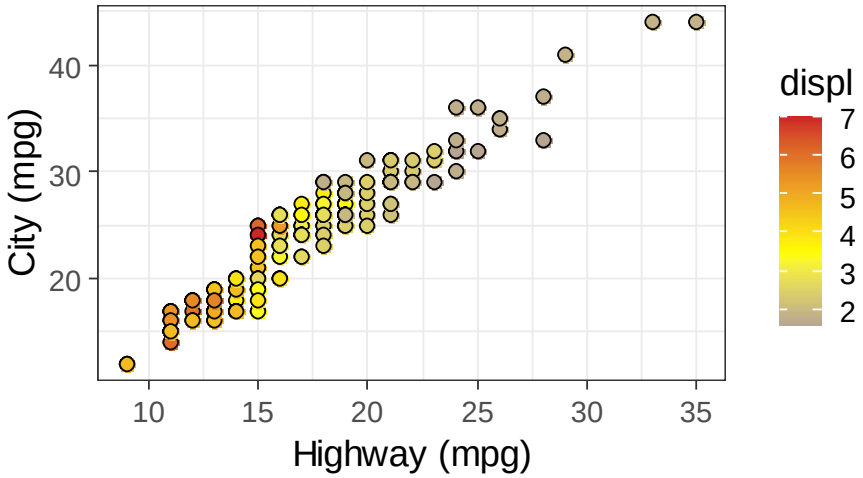


Scatter plot, black outlined circles filled with gradient color of another variable

```
mpg %>%
    ggplot(aes(x=cty, y=hwy ) ) +
    geom_point(size=3, na.rm=T, shape=21,
               aes(fill=displ), color="black") +
    scale_fill_gradient2(midpoint=mid_displ,
                         low="royalblue3",
                         mid="yellow",
                         high="firebrick3" ) +
    theme_bw(base_size = 15) +
    labs(x="Highway (mpg)", y="City (mpg)") +
    theme(aspect.ratio = 0.9)
```




```
    theme(aspect.ratio = 0.6)
```

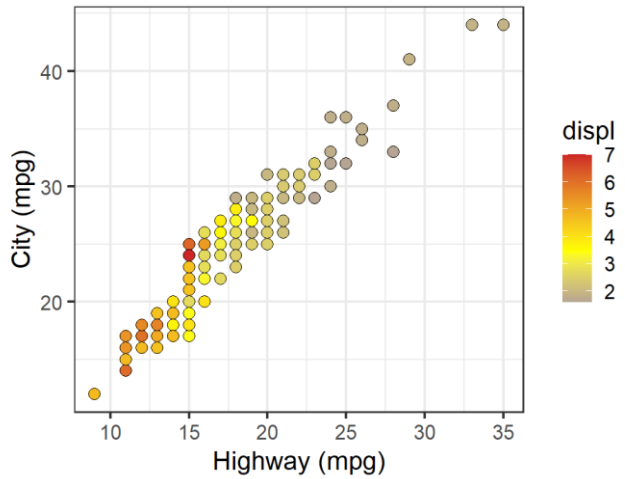


Fix the aspect ratio of a plot regardless of the plot window size of your R studio

Read the next slide as well

Specify aspect ratio in ggplot code **AND** Specify plot size in ggsave code.

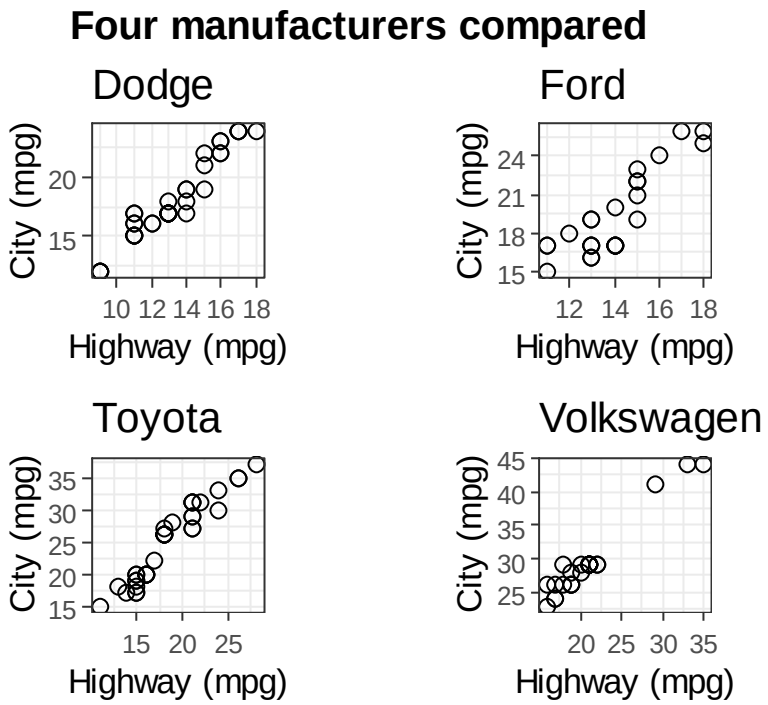
```
myplot1 = mpg %>%
   ggplot(aes(x=cty, y=hwy ) ) +
   geom_point(size=3, na.rm=T, shape=21, aes(fill=displ), color="black") +
   scale_fill_gradient2(midpoint=middispl, low="royalblue3", mid="yellow", high="firebrick3" ) +
   theme_bw(base_size = 15) +
   labs(x="Highway (mpg)", y="City (mpg)") +
   theme(aspect.ratio=0.9)


# Save your chart as a tiff file in your working directory.
ggsave("MYPLOTNAME.tif",  myplot, device='tiff', width=7, height=4, dpi=200)

# Show the current working directory
getwd()
```



Save charts with a consistent size

```
library(ggpubr)
Four_manufacturers = ggarrange(myplot_dodge, myplot_ford, myplot_toyota, myplot_volks,
                               ncol=2, nrow=2)
Four_manufacturers

# It may look crowded on the plot preview window, but you can enlarge it to view it better.

# Add a title to the combined figure.
annotate_figure(Four_manufacturers, top=text_grob("Four manufacturers compared",
                                                  face="bold",  size = 15) )
```

Need 'ggpubr' package



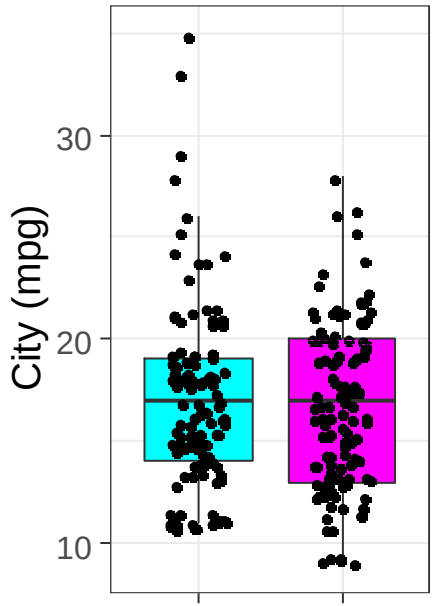
Organize multiple plots in one chart

```
library(scales)

And then use muted() function for colors.

scale_fill_manual( values = c(muted("cyan"), muted("magenta")) )
```

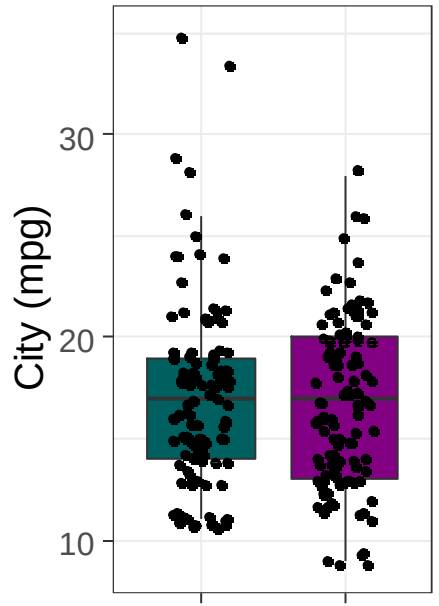
Need ‘scales’ package

Make colors less vivid and easier to the eyes

cyan & magenta



Muted cyan & magenta



Useful codes for data analysis

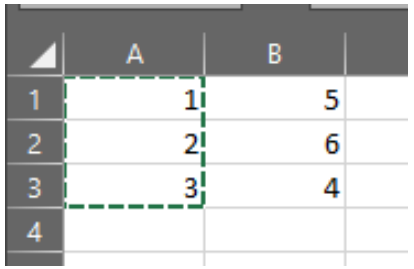
Columns = variables

Rows
= observations
= samples

	Variable 1	Variable 2	Variable 3	Variable 4	Variable 5	…
Observation 1	8.5	2.7	6.9	1.1	2.6	…
Observation 2	4.6	1.1	7.2	14.7	6.2	…
Observation 3	3.9	-1.8	3.6	22.3	0.6	…
Observation 4	1.5	4.4	NA	67.7	9.8	…
Observation 5	2.0	-0.6	9.2	90.0	6.9	…
Observation 6	4.2	4.4	6.1	104.9	4.3	…
Observation 7	NA	1.9	2.0	127.8	1.7	…
Observation 8	4.3	-4.1	6.1	122.3	8.1	…
Observation 9	9.6	-3.0	8.4	86.5	8.6	…
Observation 10	0.9	0.6	7.6	177.0	1.7	…
…	…	…	…	…	…	…

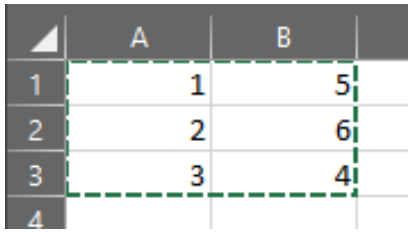
A dataframe in R is like Excel spreadsheet

```
# Copy (Ctrl + V) one column of cells in Excel, then,
MYDATAFRAME = read.table(file="clipboard", sep=",")
```



	A	B
1	1	5
2	2	6
3	3	4
4		

```
# Copy (Ctrl + V) more than one columns of cells in Excel, then,
MYDATAFRAME = read.table(file="clipboard", sep="\t")
```



	A	B
1	1	5
2	2	6
3	3	4
4		

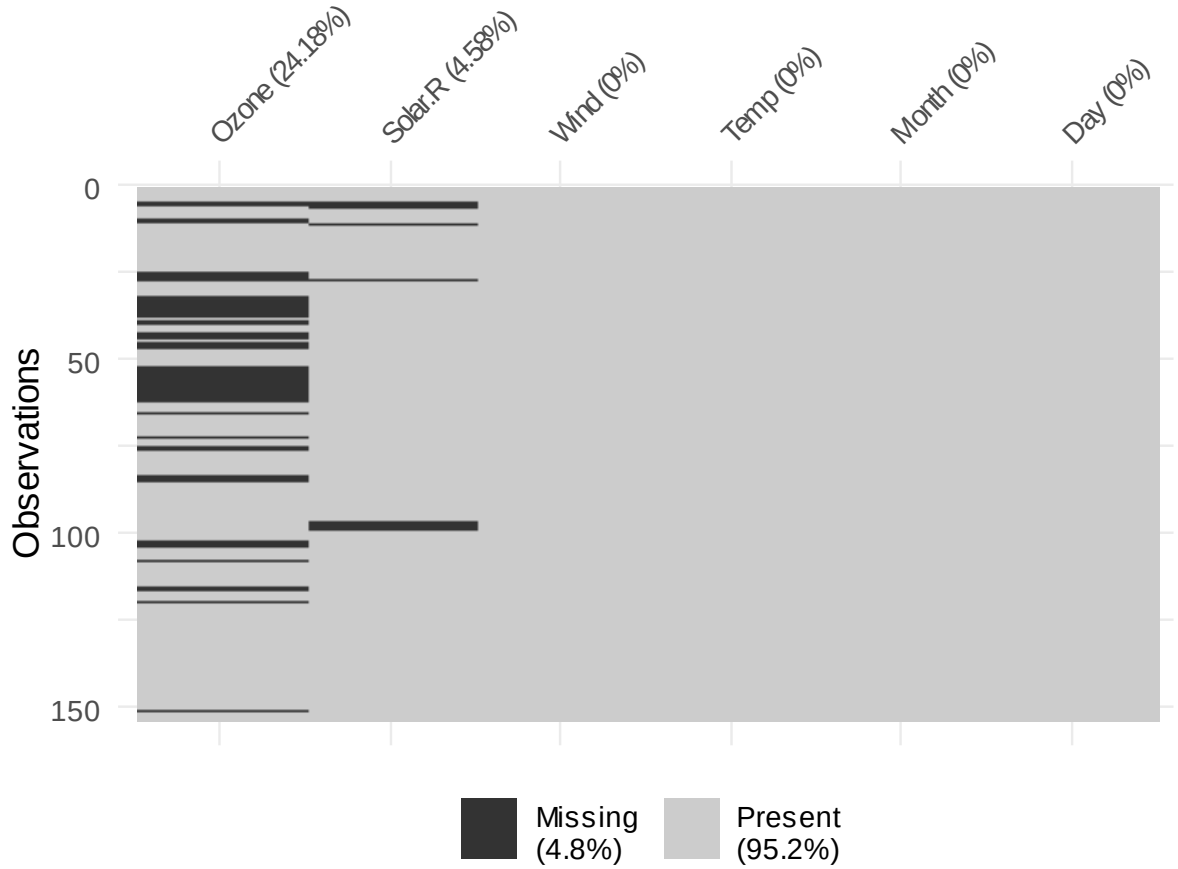
Copy excel data and make it into a dataframe in R

```
# Change the name of column 4.

colnames(MYDATAFRAME)[4] <- “MYNEWCOLUMNNAME"
```

Rename a column

```
data(airquality)  # Load 'airquality' dataset which has some missing data
head(airquality, 10)  # View the first 10 rows in airquality

library(naniar)  # Load naniar package to use vis_miss function
vis_miss(airquality)
```

```
> head(airquality, 10)
   Ozone Solar.R Wind Temp Month Day
1     41     190  7.4   67     5   1
2     36     118  8.0   72     5   2
3     12     149 12.6   74     5   3
4     18     313 11.5   62     5   4
5     NA      NA 14.3   56     5   5
6     28      NA 14.9   66     5   6
7     23     299  8.6   65     5   7
8     19      99 13.8   59     5   8
9      8      19 20.1   61     5   9
10    NA     194  8.6   69     5  10
```



Visualize missing data in each variable

Need 'naniar' package

```
MYDATAFRAME %>% filter( is.na(COLUMN_NAME) )
```

```
MYDATAFRAME %>% filter( !is.na(COLUMN_NAME) )
```

Filter IN or OUT NA values.

```
MYDATAFRAME[ is.na(MYDATAFRAME) ] <- 0
```

Note that MYDATAFRAME will be overwritten!

Replace NAs with zero in a dataframe

```
MYDATAFRAME[ order(MYDATAFRAME$MYCOLUMN), ]

OR

# with the dplyr package
MYDATAFRAME %>% arrange(MYCOLUMN)
```

Sort rows by a variable

```
MYDATAFRAME$MYVARIABLE <- factor(MYDATAFRAME$MYVARIABLE,
                                 levels=c("Level 4", "Level 2", "Level 1", "Level 3" ))

# Check the order of the factors (as well as frequency)
table(MYDATAFRAME$MYVARIABLE)

# Sort according to the new order: the last item will be on top on a scatter plot.
MYDATAFRAME_s <- MYDATAFRAME[order(MYDATAFRAME$MYVARIABLE, decreasing = F), ]
```

Specify the order of factors to be displayed/plotted, instead of the default alphabetical/numerical order.

```
# Take two random rows from MYDATAFRAME.
MYDATAFRAME[ sample(nrow(MYDATAFRAME), 2), ]
```

Take random sample rows from a dataframe.

```
# Make a 4x3 dataframe with random numbers.
  m1 <- matrix( C<-(1:12), nrow=4, ncol=3 )
  m1

# Add each row
  apply(m1, 1, sum)

# Add each column
  apply(m1, 2, sum)
```

Sum each column or row of a matrix

```
# Get column totals
CTotal = colSums( t(MYDATAFRAME) )

# Add column totals
MYDATAFRAMEcc = cbind(MYDATAFRAME, CTotal)

# Add row totals
MYDATAFRAMEccrr = rbind(MYDATAFRAMEcc, colSums(MYDATAFRAMEcc))

# Change the last rowname
rownames(MYDATAFRAMEccrr)[length(rownames(MYDATAFRAMEccrr))] <- “RTotal"
MYDATAFRAMEccrr
```

Sum each column or row of a dataframe

```
MYDATAFRAME$mean <- rowMeans(MYDATAFRAME[, c('MYCOLUMN1', ' MYCOLUMN2')],
na.rm=TRUE)

 # na.rm=TRUE option ignores NA and gives average with existing data.
```

Take average of multiple columns

```
# Convert 2nd-4th columns to numeric
MYDATAFRAME[, c(2:4)] = sapply( MYDATAFRAME[, c(2:4)], as.numeric )
```

Convert column(s) of a dataframe to numeric.

```
# Calculate variation for each row. MARGIN=1 means apply to rows.
# apply(MYDATAFRAME, MARGIN=1, var)

# Calculate variation for each column. MARGIN=2 means apply to columns.
# apply(MYDATAFRAME, MARGIN=2, var)
```

Apply a function to all columns or rows.

```
# Replace ‘Original text’ with ‘Replaced text’.
# also applies to a part of text.

MYDATAFRAME$MYVARIABLE <- gsub('Original text',
                               'Replaced text',
                               MYDATAFRAME$MYVARIABLE)
```

Replace character strings in a data frame