Intro to R

Data Visualization

`esquisse` and `ggplot2`

->

->

->

ggplot2

- A package for producing graphics - gg = *Grammar of Graphics*
- Created by Hadley Wickham in 2005
- Belongs to “Tidyverse” family of packages
- *“Make a ggplot”* = Make a plot with the use of ggplot2 package

ggplot2

Based on the idea of:

layering

plot objects are placed on top of each other with `+`

📉

➕

📈

ggplot2

- Pros: extremely powerful/flexible – allows combining multiple plot elements together, allows high customization of a look, many resources online
- Cons: ggplot2-specific “grammar of graphic” of constructing a plot
- ggplot2 gallery

Tidy data

To make graphics using `ggplot2`, our data needs to be in a **tidy** format

Tidy data:

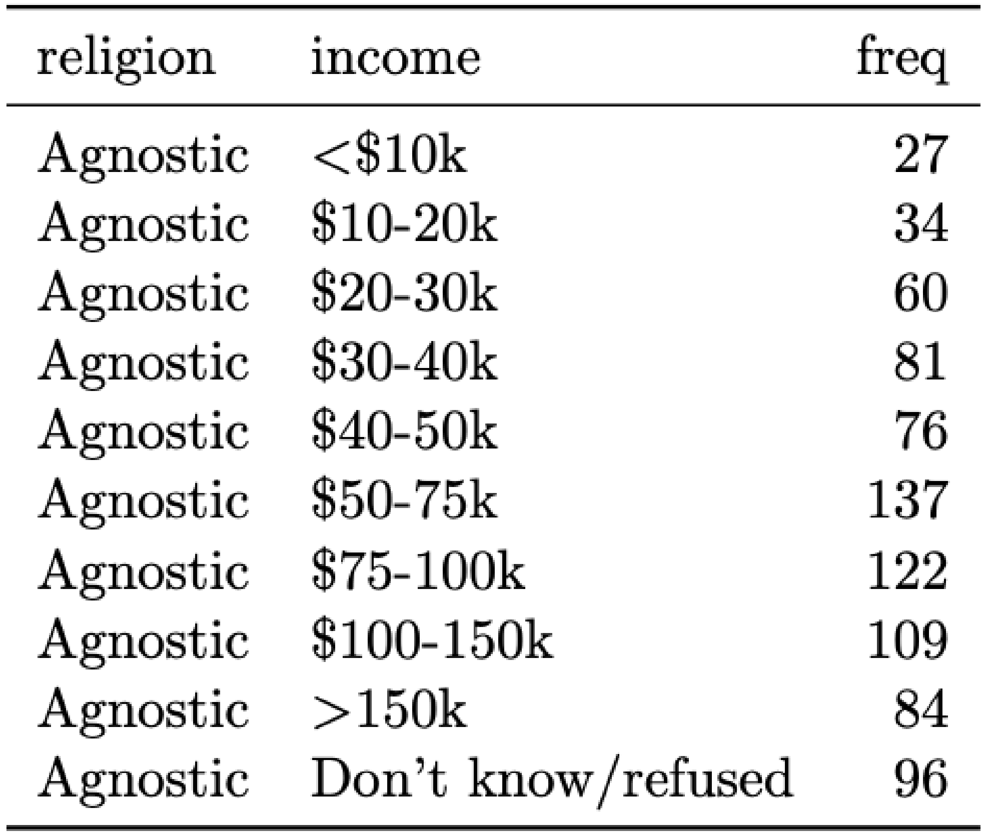
1. Each variable forms a column.
2. Each observation forms a row.

Messy data:

- Column headers are values, not variable names.
- Multiple variables are stored in one column.
- Variables are stored in both rows and columns.

Tidy data: example

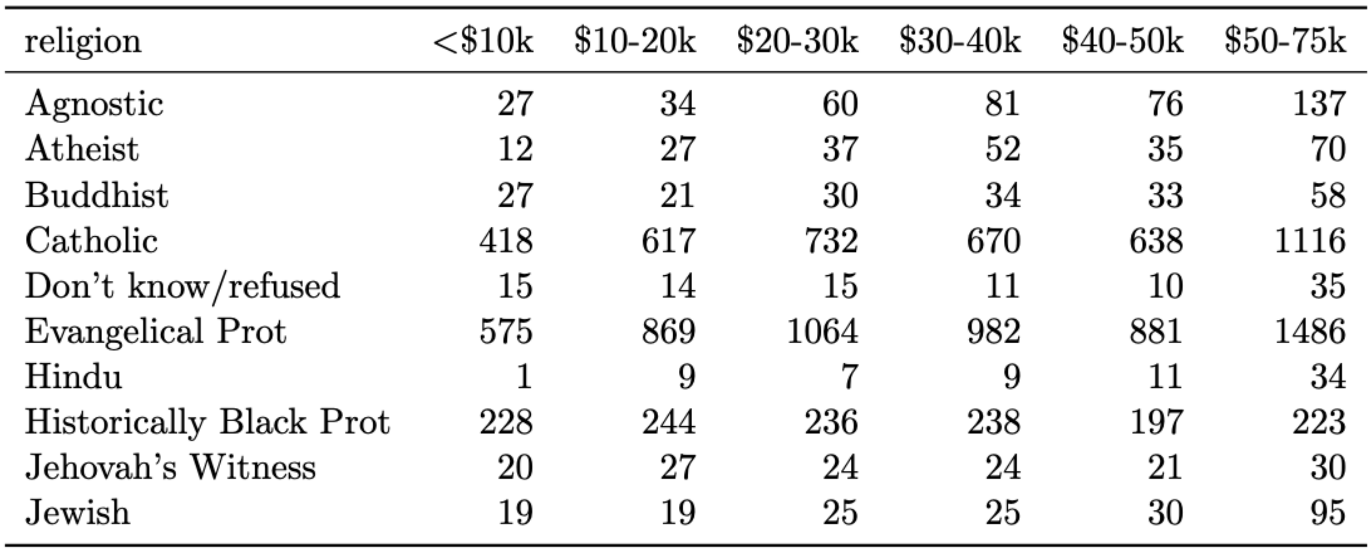
Each variable forms a column. Each observation forms a row.



religion	income	freq
Agnostic	<$10k	27
Agnostic	$10-20k	34
Agnostic	$20-30k	60
Agnostic	$30-40k	81
Agnostic	$40-50k	76
Agnostic	$50-75k	137
Agnostic	$75-100k	122
Agnostic	$100-150k	109
Agnostic	>150k	84
Agnostic	Don't know/refused	96

Messy data: example

Column headers are values, not variable names



religion	<$10k	$10-20k	$20-30k	$30-40k	$40-50k	$50-75k
Agnostic	27	34	60	81	76	137
Atheist	12	27	37	52	35	70
Buddhist	27	21	30	34	33	58
Catholic	418	617	732	670	638	1116
Don't know/refused	15	14	15	11	10	35
Evangelical Prot	575	869	1064	982	881	1486
Hindu	1	9	7	9	11	34
Historically Black Prot	228	244	236	238	197	223
Jehovah's Witness	20	27	24	24	21	30
Jewish	19	19	25	25	30	95

Table 4: The first ten rows of data on income and religion from the Pew Forum. Three columns, `$75-100k`, `$100-150k` and `>150k`, have been omitted

Read more about tidy data and see other examples: Tidy Data tutorial by Hadley Wickham

It's also helpful to have data in long format!!!

Making data to plot

```r
set.seed(3)
var_1 <- seq(from = 1, to = 30)
var_2 <- rnorm(30)
my_data = tibble(var_1, var_2)
my_data
```

```
# A tibble: 30 × 2
   var_1   var_2
   <int>   <dbl>
 1     1 -0.962
 2     2 -0.293
 3     3  0.259
 4     4 -1.15
 5     5  0.196
 6     6  0.0301
 7     7  0.0854
 8     8  1.12
 9     9 -1.22
10    10  1.27
# … with 20 more rows
```

First plot with `ggplot2` package

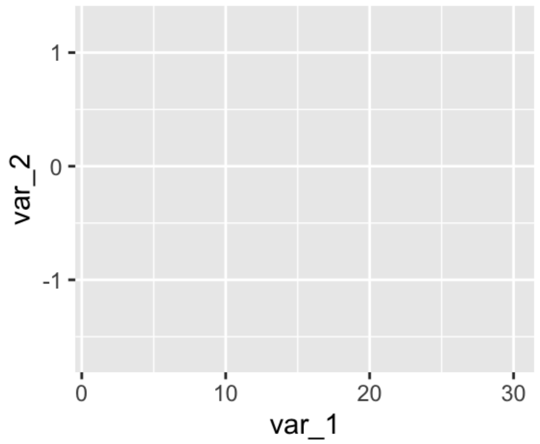
First layer of code with `ggplot2` package

Will set up the plot - it will be empty!

- **Aesthetic mapping** `(mapping= aes(x= , y =))` describes how variables in our data are mapped to elements of the plot

```
library(ggplot2) # don't forget to load ggplot2
# This is not code but shows the general format
ggplot({data_to plot},  mapping = aes(x = {var in data to plot},
                                      y = {var in data to plot}))
```

```
ggplot(my_data, mapping = aes(x = var_1, y = var_2))
```



Next layer code with `ggplot2` package

There are many to choose from, to list just a few:

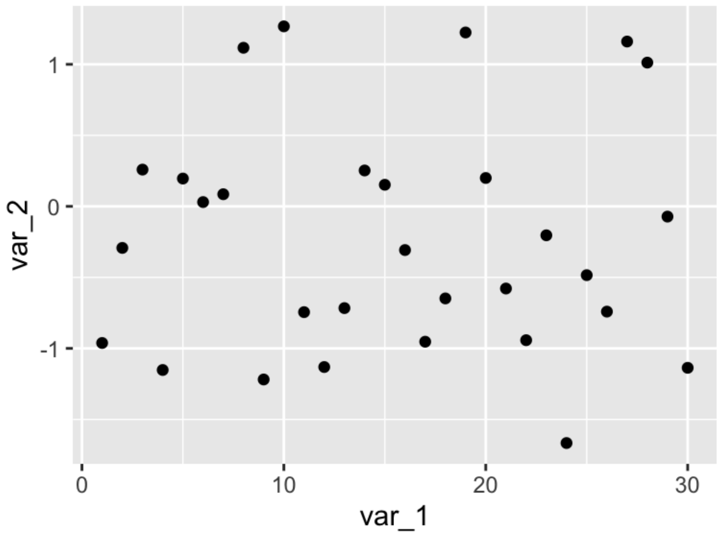
- `geom_point()` – points (we have seen)
- `geom_line()` – lines to connect observations
- `geom_boxplot()`
- `geom_histogram()`
- `geom_bar()`
- `geom_col()`
- `geom_errorbar()`
- `geom_density()`
- `geom_tile()` – blocks filled with color

Next layer code with `ggplot2` package

Need the + sign to add the next layer to specify the type of plot

```
ggplot({data_to plot},  mapping = aes(x = {var in data to plot},
                                      y = {var in data to plot})) +
  geom_{type of plot}</div>
```

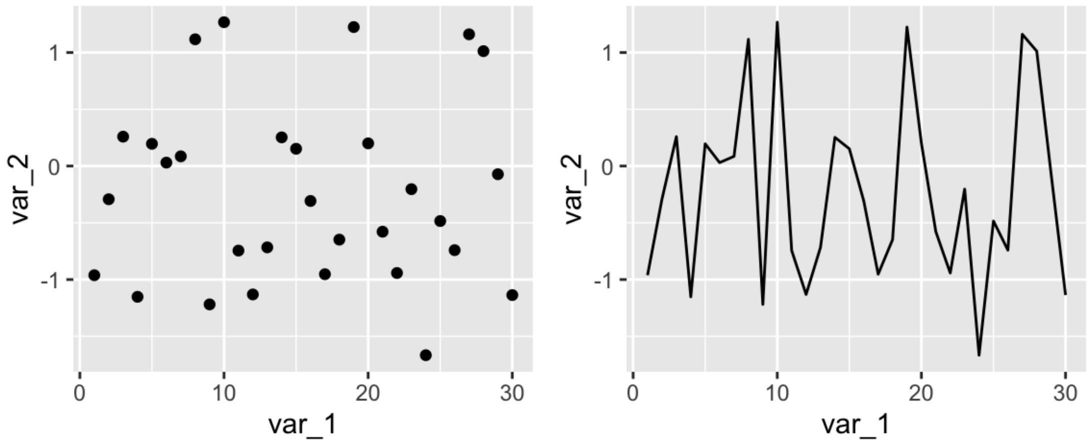
```
ggplot(my_data, mapping = aes(x = var_1, y = var_2)) +
  geom_point()
```



Read as: *add points to the plot (use data as provided by the aesthetic mapping)*

Specifying plot layers: examples

```r
plt1 <-
  ggplot(my_data, aes(x = var_1, y = var_2)) +
  geom_point()

plt2 <-
  ggplot(my_data, aes(x = var_1, y = var_2)) +
  geom_line()

plt1; plt2 # to have 2 plots printed next to each other on a slide
```

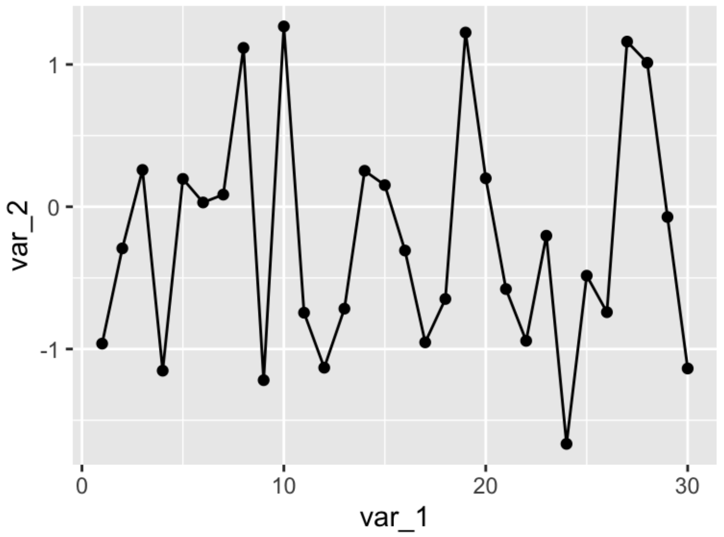


Also check out the `patchwork` package

Specifying plot layers: combining multiple layers

Layer a plot on top of another plot with `+`

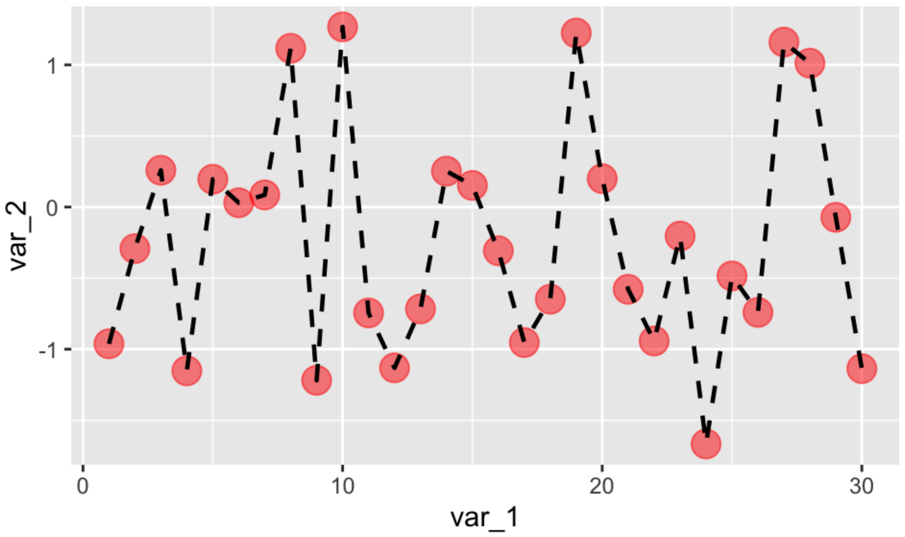
```
ggplot(my_data, aes(x = var_1, y = var_2)) +
  geom_point() +
  geom_line()
```



Customize the look of the plot

You can change look of each layer separately.

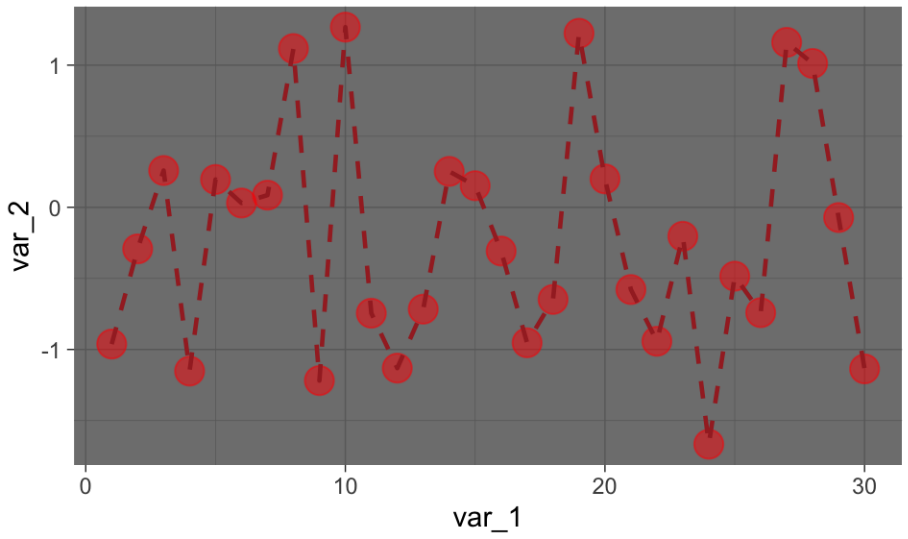
```
ggplot(my_data, aes(x = var_1, y = var_2)) +
  geom_point(size = 5, color = "red", alpha = 0.5) +
  geom_line(size = 0.8, color = "black", linetype = 2)
```



Customize the look of the plot

You can change the look of whole plot using `theme_*()` functions.

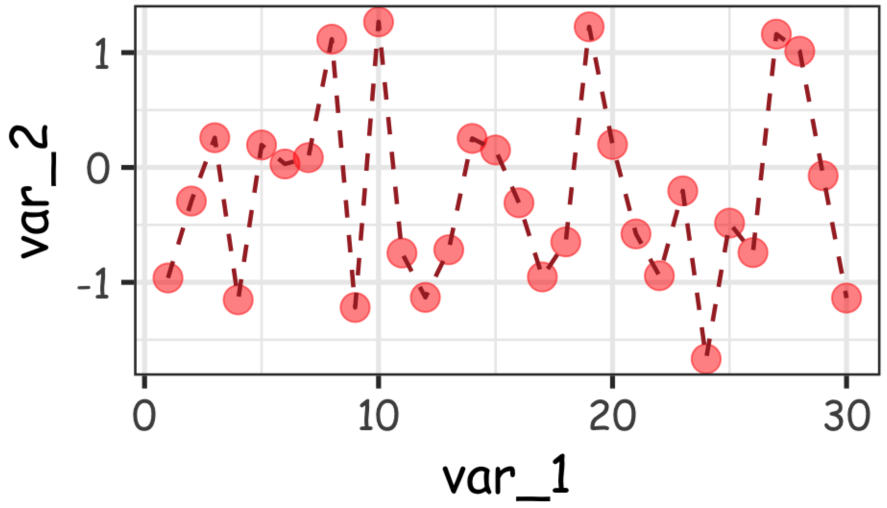
```
ggplot(my_data, aes(x = var_1, y = var_2)) +
  geom_point(size = 5, color = "red", alpha = 0.5) +
  geom_line(size = 0.8, color = "brown", linetype = 2) +
  theme_dark()
```



Customize the look of the plot

You can change the look of whole plot - **specific elements, too** - like changing font and font size - or even more fonts

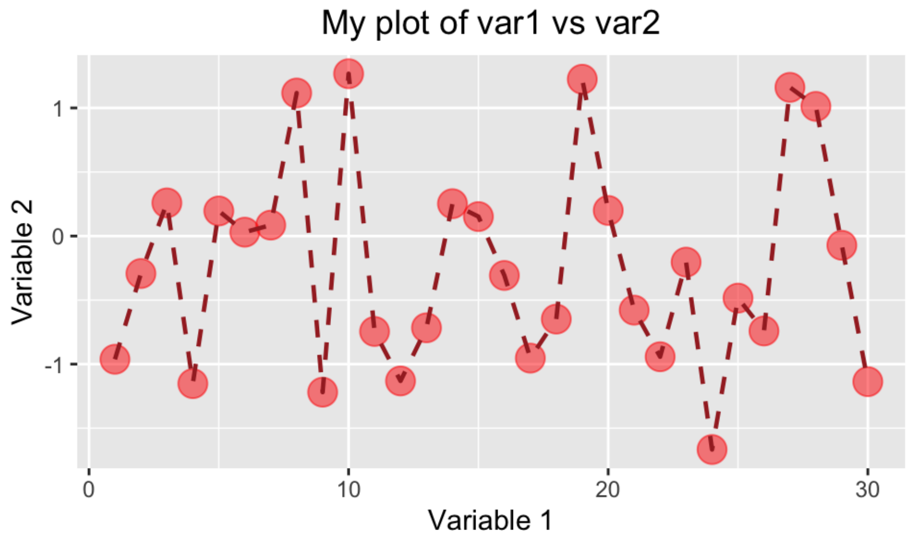
```
ggplot(my_data, aes(x = var_1, y = var_2)) +
  geom_point(size = 5, color = "red", alpha = 0.5) +
  geom_line(size = 0.8, color = "brown", linetype = 2) +
  theme_bw(base_size = 20, base_family = "Comic Sans MS")
```



Adding labels

The `labs()` function can help you add or modify titles on your plot.

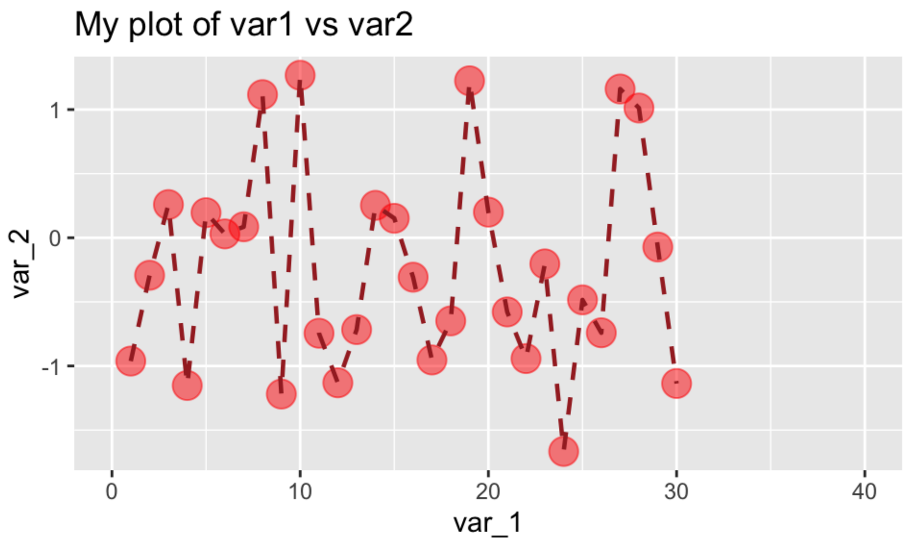
```
ggplot(my_data, aes(x = var_1, y = var_2)) +
  geom_point(size = 5, color = "red", alpha = 0.5) +
  geom_line(size = 0.8, color = "brown", linetype = 2) +
  labs(title = "My plot of var1 vs var2",
            x = "Variable 1",
            y = "Variable 2") +
  theme(plot.title = element_text(hjust = 0.5))
```



Changing axis

`xlim()` and `ylim()` can specify the limits for each axis

```
ggplot(my_data, aes(x = var_1, y = var_2)) +
  geom_point(size = 5, color = "red", alpha = 0.5) +
  geom_line(size = 0.8, color = "brown", linetype = 2) +
  labs(title = "My plot of var1 vs var2") +
  xlim(0,40)
```



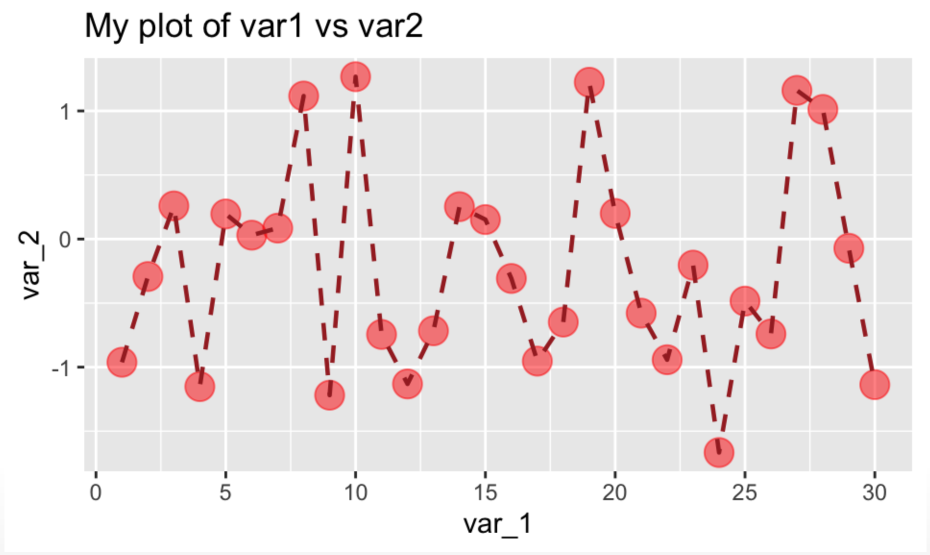
Changing axis

`scale_x_continuous()` and `scale_y_continuous()` can change how the axis is plotted. Can use the `breaks` argument to specify how you want the axis ticks to be.

```
seq(from = 0, to = 30, by = 5)
```

```
[1]  0  5 10 15 20 25 30
```

```
ggplot(my_data, aes(x = var_1, y = var_2)) +
  geom_point(size = 5, color = "red", alpha = 0.5) +
  geom_line(size = 0.8, color = "brown", linetype = 2) +
  labs(title = "My plot of var1 vs var2") +
  scale_x_continuous(breaks = seq(from = 0, to = 30, by = 5))
```



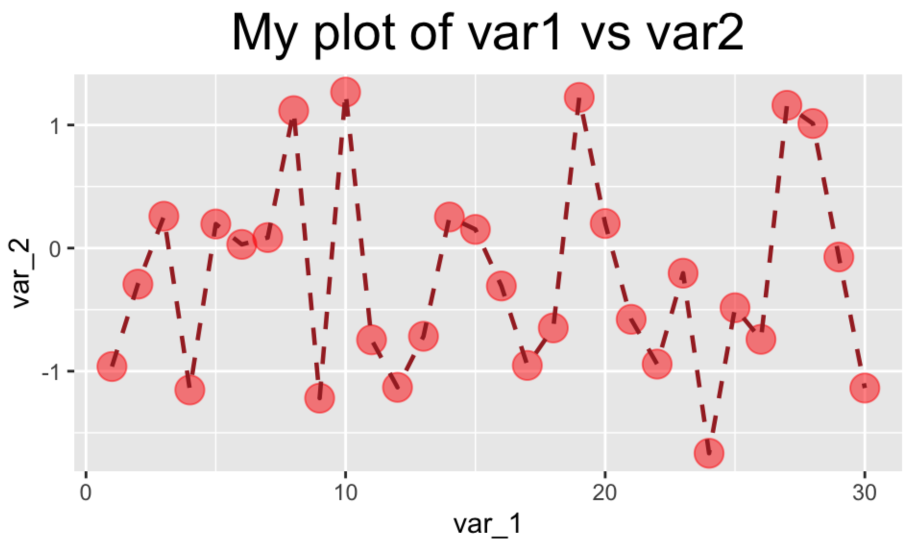
Lab 1

Lab document:
http://jhudatascience.org//intro_to_r/Data_Visualization/lab/Data_Visualization_Lab.Rm

theme() function

The `theme()` function can help you modify various elements of your plot. Here we will adjust the horizontal placement of the plot title.

```
ggplot(my_data, aes(x = var_1, y = var_2)) +
  geom_point(size = 5, color = "red", alpha = 0.5) +
  geom_line(size = 0.8, color = "brown", linetype = 2) +
  labs(title = "My plot of var1 vs var2") +
  theme(plot.title = element_text(hjust = 0.5, size = 20))
```



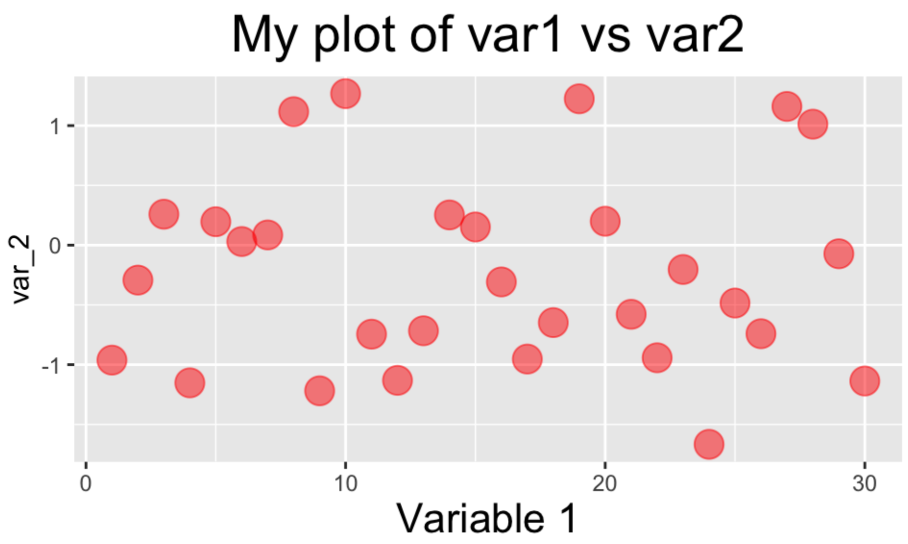
theme() function

The `theme()` function always takes:

1. an object to change (use ?theme() to see - plot.title, axis.title, axis.ticks etc.)
2. the aspect you are changing about this: `element_text()`, `element_line()`, `element_rect()`, `element_blank()`
3. what you are changing:
   - text: size, color, fill, face, alpha, angle
   - position: “top”, “bottom”, “right”, “left”, “none”
   - rectangle: size, color, fill, linetype
   - line: size, color, linetype,

theme() function

```r
ggplot(my_data, aes(x = var_1, y = var_2)) +
  geom_point(size = 5, color = "red", alpha = 0.5) +
  labs(title = "My plot of var1 vs var2", x = "Variable 1") +
  theme(plot.title = element_text(hjust = 0.5, size = 20),
        axis.title.x = element_text(size = 16))
```



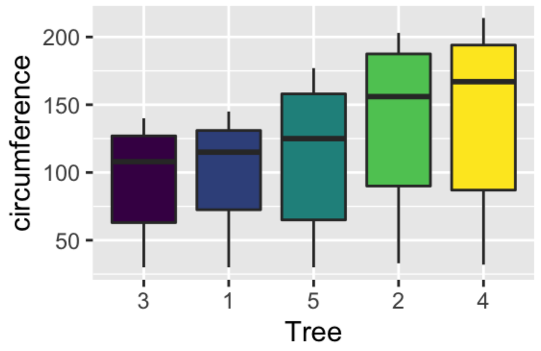
theme() function

```
head(Orange, 3)
```

```
  Tree age circumference
1    1 118            30
2    1 484            58
3    1 664            87
```

If specifying position - use: “top”, “bottom”, “right”, “left”, “none”

```
ggplot(Orange, aes(x = Tree, y = circumference, fill = Tree)) +
  geom_boxplot() +
  theme(legend.position = "none")
```



Can make your own theme to use on plots!

Guide on how to: https://rpubs.com/mclaire19/ggplot2-custom-themes

Group and/or color by variable's values

First, we will generate some data frame for the purpose of demonstration.

- 20 different items (e.g. products in a store)
- of 2 different categories (e.g. pasta, rice)
- 100 price values collected over time for each item

```r
# create 4 vectors: 2x character class and 2x numeric class
item_categ <- as.vector(sapply(1:20, function(i) rep(sample(c("pasta", "rice")
item_ID  <- rep(seq(from = 1, to = 20), each = 100)
item_ID <- paste0("ID_", item_ID)
observation_time  <- rep(seq(from = 1, to = 100), times = 20)
item_price <- as.vector(replicate(20, cumsum(rnorm(100))))
item_price <- item_price + abs(min(item_price)) + 1

# use 4 vectors to create data frame with 4 columns
df  <- data.frame(item_ID, item_categ, observation_time, item_price)
```

Group and/or color by variable's values

```
head(df, 3)
```

```
  item_ID item_categ observation_time item_price
1    ID_1      pasta                1   21.68561
2    ID_1      pasta                2   21.37515
3    ID_1      pasta                3   23.07403
```

```
tail(df, 3)
```

```
     item_ID item_categ observation_time item_price
1998   ID_20      pasta               98   34.97370
1999   ID_20      pasta               99   35.34058
2000   ID_20      pasta              100   35.47118
```

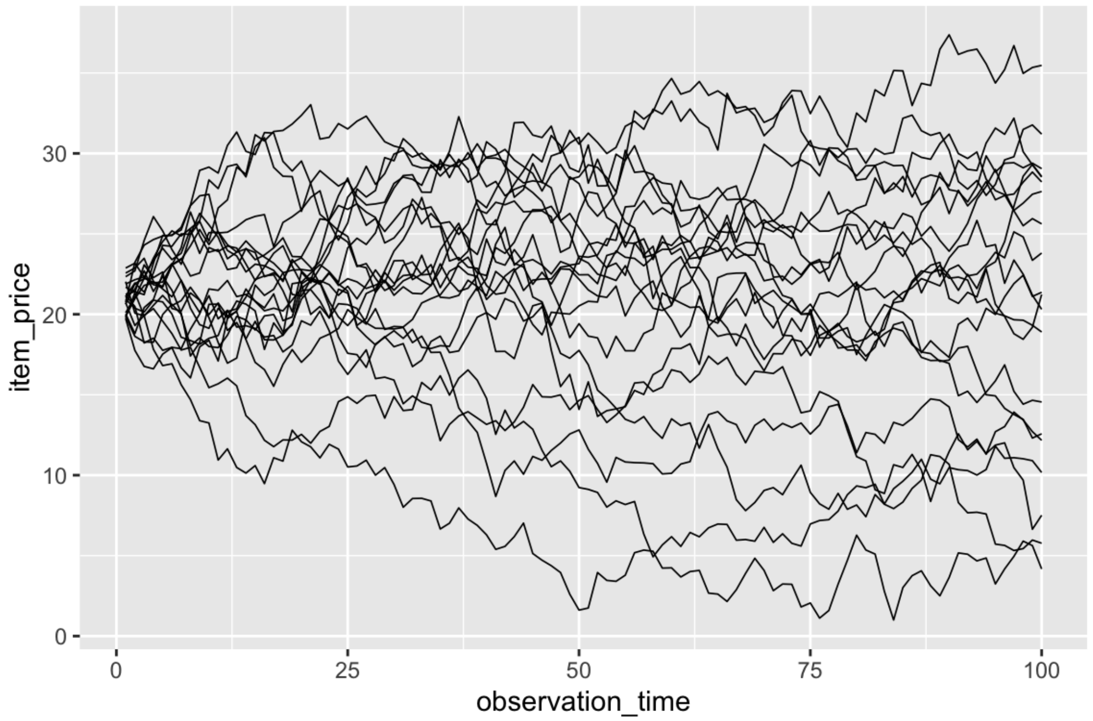
```
str(df)
```

```
'data.frame':   2000 obs. of  4 variables:
 $ item_ID         : chr  "ID_1" "ID_1" "ID_1" "ID_1" ...
 $ item_categ      : chr  "pasta" "pasta" "pasta" "pasta" ...
 $ observation_time: int  1 2 3 4 5 6 7 8 9 10 ...
 $ item_price      : num  21.7 21.4 23.1 22.3 22.6 ...
```

Group and/or color by variable's values

You can use `group` element in a mapping to indicate that each `item_ID` will have a separate price line (more generally: a separate layer element)

```
ggplot(df, aes(x = observation_time, y = item_price, group = item_ID)) +
  geom_line(size = 0.3)
```

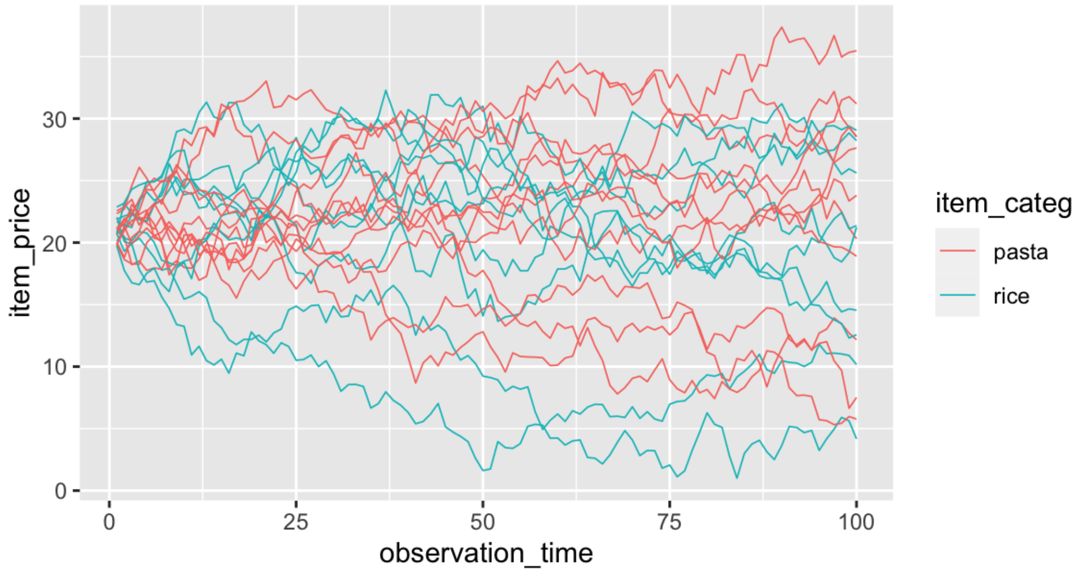


Group and/or color by variable's values

You can use `color` element in a mapping to indicate that each `item_categ` will have different color used.

Colors palette is selected by default (and can be modified). Legend position, legend title etc. have a default look (and can be modified).

```
ggplot(df, aes(x = observation_time, y = item_price, group = item_ID,
               color = item_categ)) +
  geom_line(size = 0.3)
```

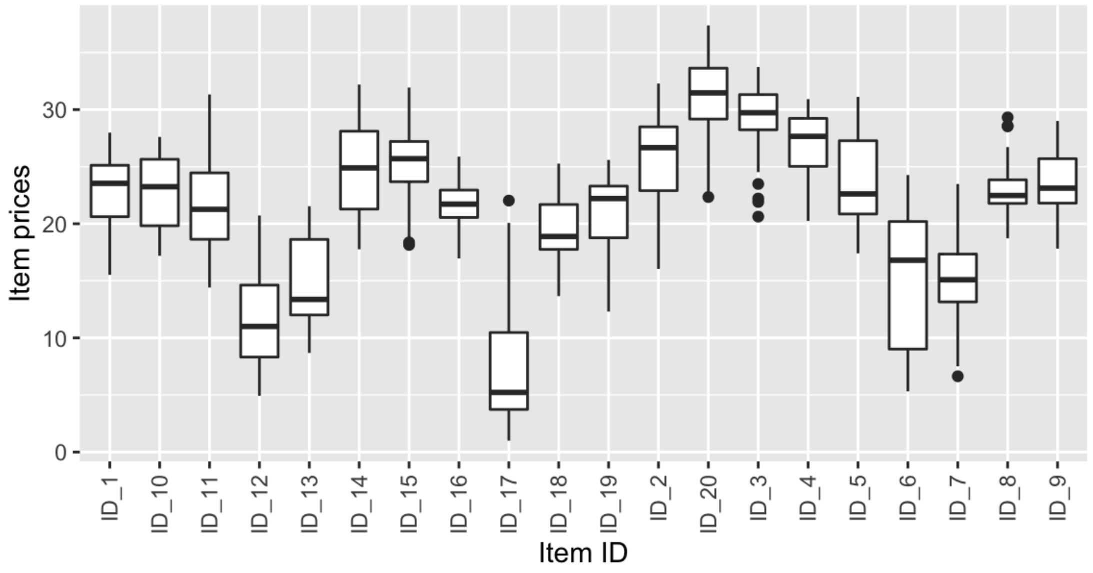


Group and/or color by variable's values

Here, we use boxplot instead of lines.

Note how aesthetic mappings are defined now: `aes(x = item_ID, y = item_price)`.

```r
ggplot(df, aes(x = item_ID, y = item_price)) +
  geom_boxplot() +
  labs(x = "Item ID", y = "Item prices") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```

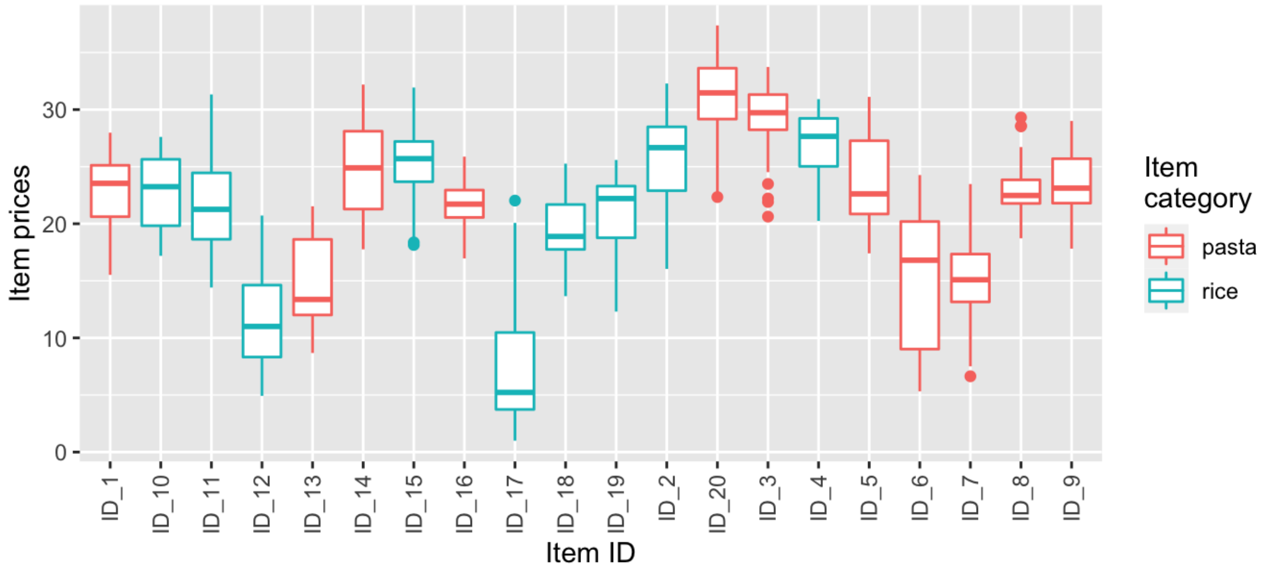


Group and/or color by variable's values

We use `color` element in mappings to indicate that each `item_categ` will have different **color of boxplot box** used.

We also use `color = "Item\ncategory"` to change name of legend.

```
ggplot(df, aes(x = item_ID, y = item_price, color = item_categ)) +
  geom_boxplot() +
  labs(x = "Item ID", y = "Item prices", color = "Item\ncategory") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```

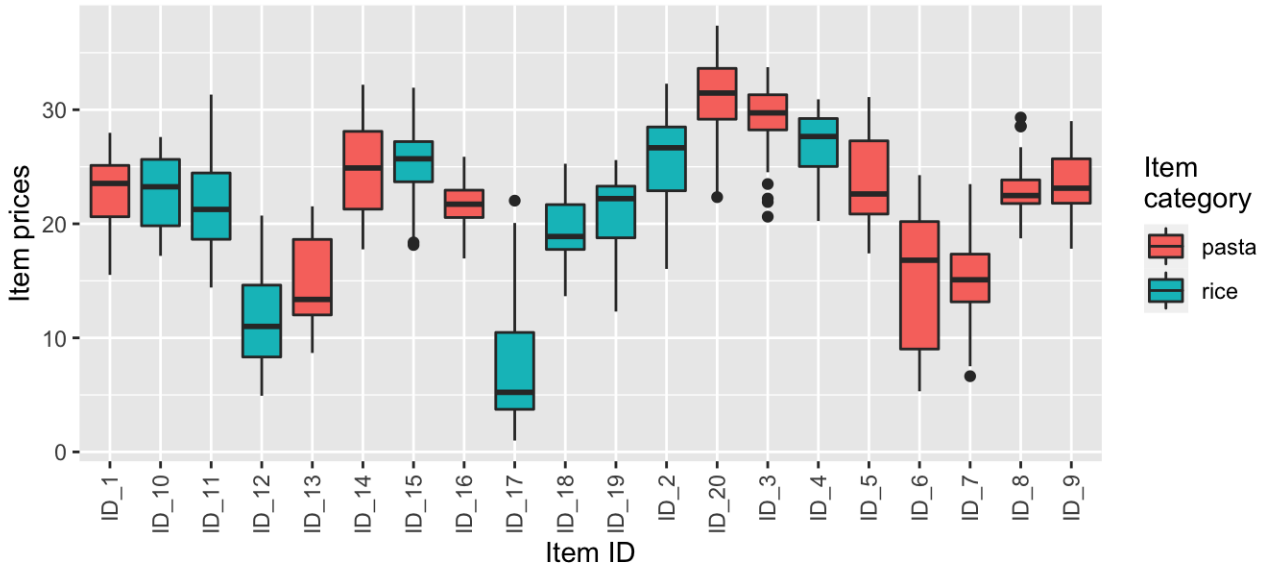


Group and/or color by variable's values

We use `fill` element in mappings to indicate that each `item_categ` will have different **color of boxplot filling** used.

We also use `fill = "Item\ncategory"` to change name of legend.

```
ggplot(df, aes(x = item_ID, y = item_price, fill = item_categ)) +
  geom_boxplot() +
  labs(x = "Item ID", y = "Item prices", fill = "Item\ncategory") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```



Group and/or color by variable's values

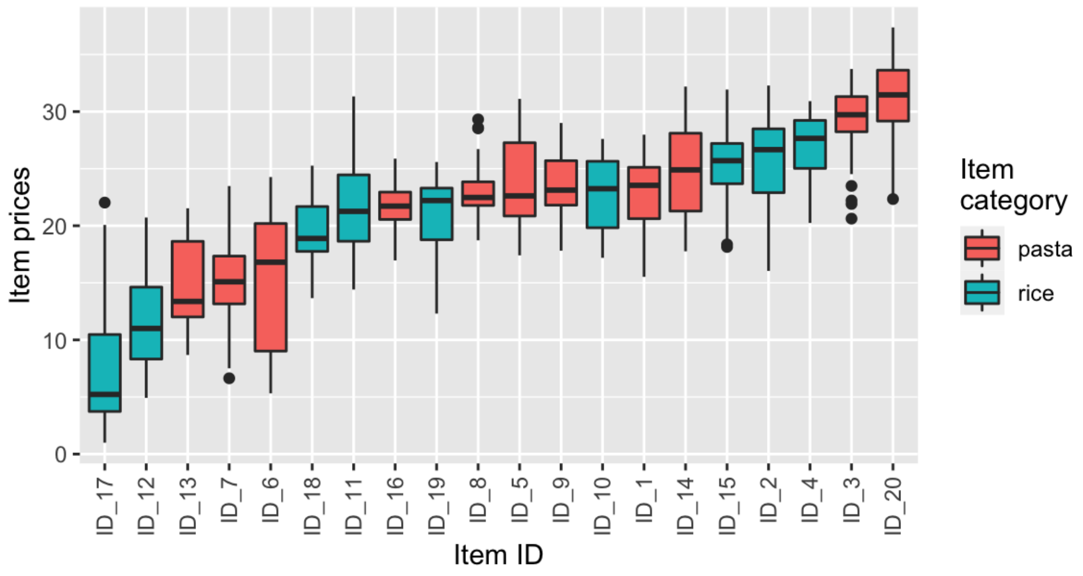
Let's make some tweaks to item ID (`item_ID`) ordering by creating a factor version with a certain order of factor levels

```
item_ID_levels <-
  df %>%
  group_by(item_ID) %>%
  summarise(item_price_median = median(item_price)) %>%
  arrange(item_price_median) %>%
  pull(item_ID)

df <-
  df %>%
  mutate(item_ID_factor = factor(item_ID, levels = item_ID_levels))
```

Group and/or color by variable's values

Same as 2 slides ago, but we replaced `item_ID` with `item_ID_factor` in mappings definition (`aes()`).

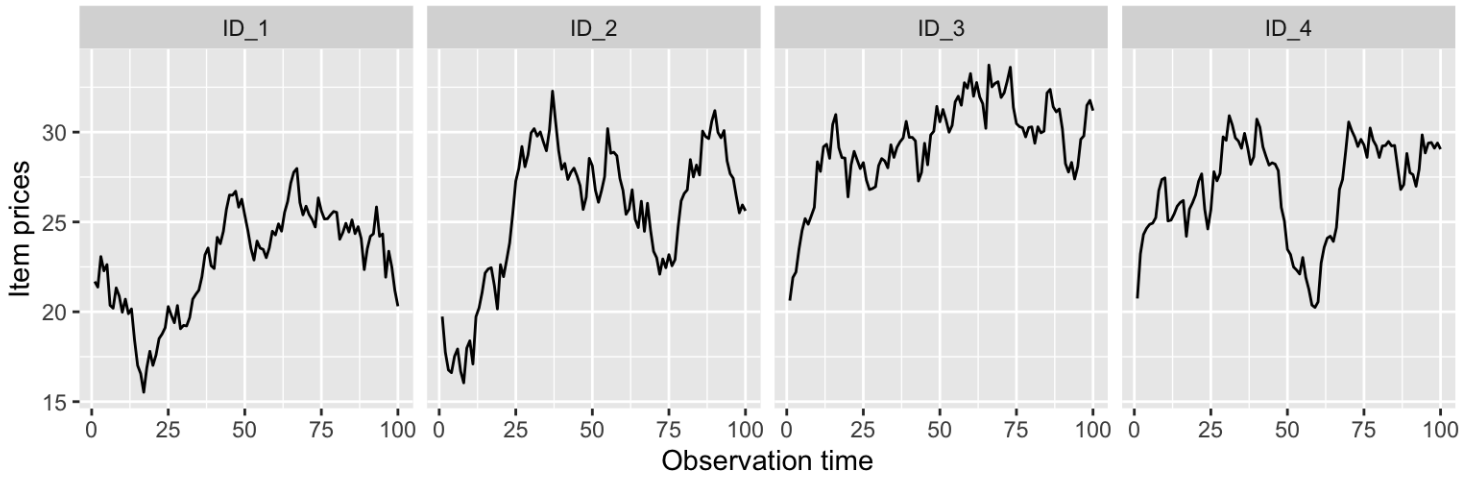
```r
ggplot(df, aes(x = item_ID_factor, y = item_price, fill = item_categ)) +
  geom_boxplot() +
  labs(x = "Item ID", y = "Item prices", fill = "Item\ncategory") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```



Split plot into panels by variable's values

We define data subset to keep only 4 (out of 20) item IDs. We use `facet_grid(. ~ item_ID)` to split the plot into panels where each product ID has a separate panel

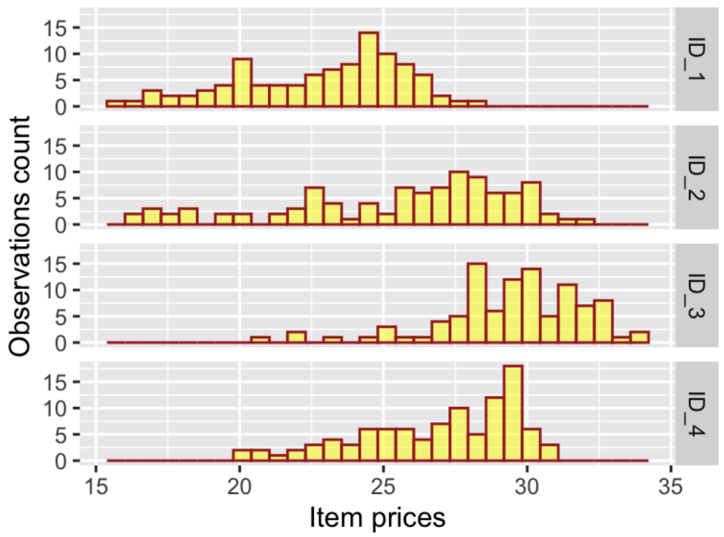
```
df_subset <- df %>%
  filter(item_ID %in% c("ID_1", "ID_2", "ID_3", "ID_4"))

ggplot(df_subset, aes(x = observation_time, y = item_price)) +
  geom_line() +
  labs(x = "Observation time", y = "Item prices") +
  facet_grid(. ~ item_ID)
```



Split plot into panels by variable's values

We define data subset to keep only 4 (out of 20) item IDs. We use `facet_grid(item_ID ~ .)` to split the plot into panels where each product ID has a separate panel

```
ggplot(df_subset, aes(x = item_price)) +
  geom_histogram(fill = "yellow", color = "brown", alpha = 0.5) +
  labs(x = "Item prices", y = "Observations count") +
  facet_grid(item_ID ~ .)
```



Saving a ggplot to file

A few options:

- RStudio > Plots > Export > Save as image / Save as PDF
- RStudio > Plots > Zoom > [right mouse click on the plot] > Save image as
- In the code

```r
plot_FINAL <-
  ggplot(df_subset, aes(x = item_price)) +
  geom_histogram(fill = "yellow", color = "brown", alpha = 0.5) +
  labs(x = "Item prices", y = "Observations count") +
  facet_grid(item_ID ~ .)

ggsave(filename = "very_important_plot.png",  # will save in working directory
       plot = plot_FINAL,
       width = 6, height = 3.5)                # by default in inch
```

Lab 2

Lab document: http://jhudatascience.org//intro_to_r/Data_Visualization/lab/Data_Visualization_Lab.Rm