Data visualization

ggplot2

Ram Thapa
Moffitt Cancer Center

June 30, 2021

- Data visualization is important in most phases of data analysis workflow i.e. from exploratory data analysis to effectively communicating our results
- Data visualization communicates information much quicker than numerical tables

> The greatest value of a picture is when it forces us to notice what we never expected to see.
>
> -- John Tukey

This quote from John Tukey explains the essence of data visualization

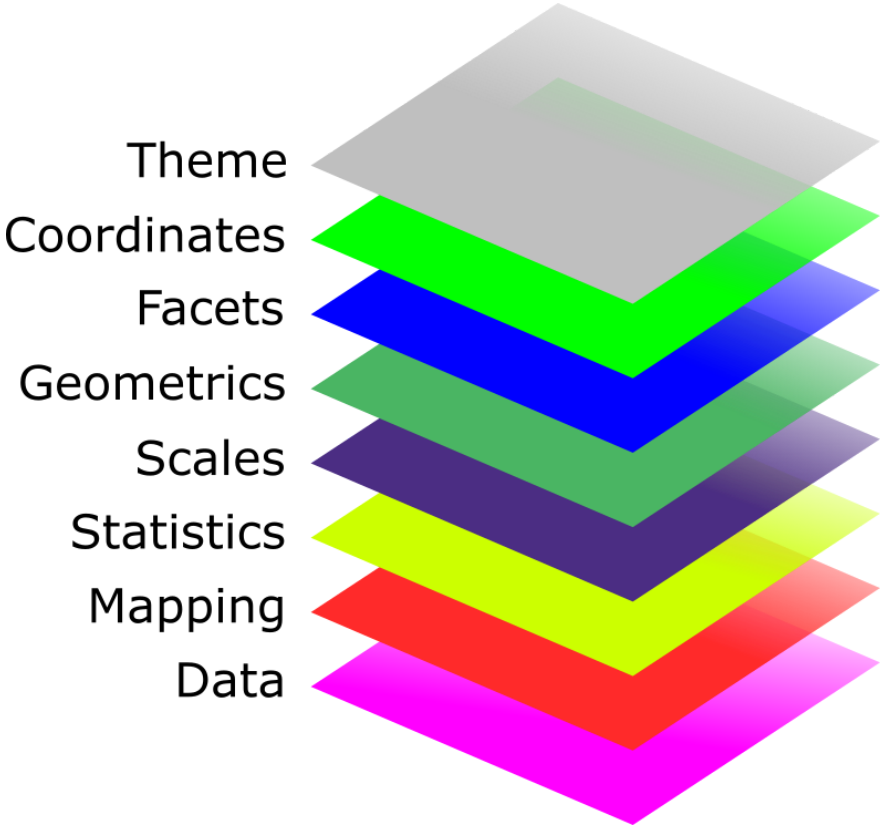
Plotting with {ggplot2} package.

- {ggplot2} is one of the most popular R packages for data visualization. It is a part of {tidyverse} R meta-package
- {ggplot2} is based on a set of principles "Grammar of graphic" and provides a cohesive system for declaratively creating elegant graphics
- Statistical graphic is a mapping from data to aesthetics (such as color, shape and size) represented by geometrics (such as points, lines etc.)

Artwork by @allison_horst

Grammar of graphics

The basic idea is that a statistical graphic can be created in a layered fashion, starting with a layer showing the data then adding layers of graphical objects, annotations and statistical summaries



Data layer

Data to be plotted

{`ggplot2`} prefers data to be in *tidy* format

Key features of tidy data:

- Each column is a variable
- Each row is an observation
- Each value must have its own cell

> *Most of plotting problems in ggplot2 boil down to data wrangling problems*

`ggplot()` is the main plotting function and provides template where plots are constructed in layers. It creates a coordinate system where you can add layers to.

R Code | Plot

```
ggplot(data = tcga_clinical)
```

`ggplot()` is the main plotting function and provides template where plots are constructed in layers. It creates a coordinate system where you can add layers to.

R Code | Plot

Aesthetic mapping layer

Aesthetic layer or `aes()` for short links variables in data to graphical objects (e.g. by describing position, size, color, etc)

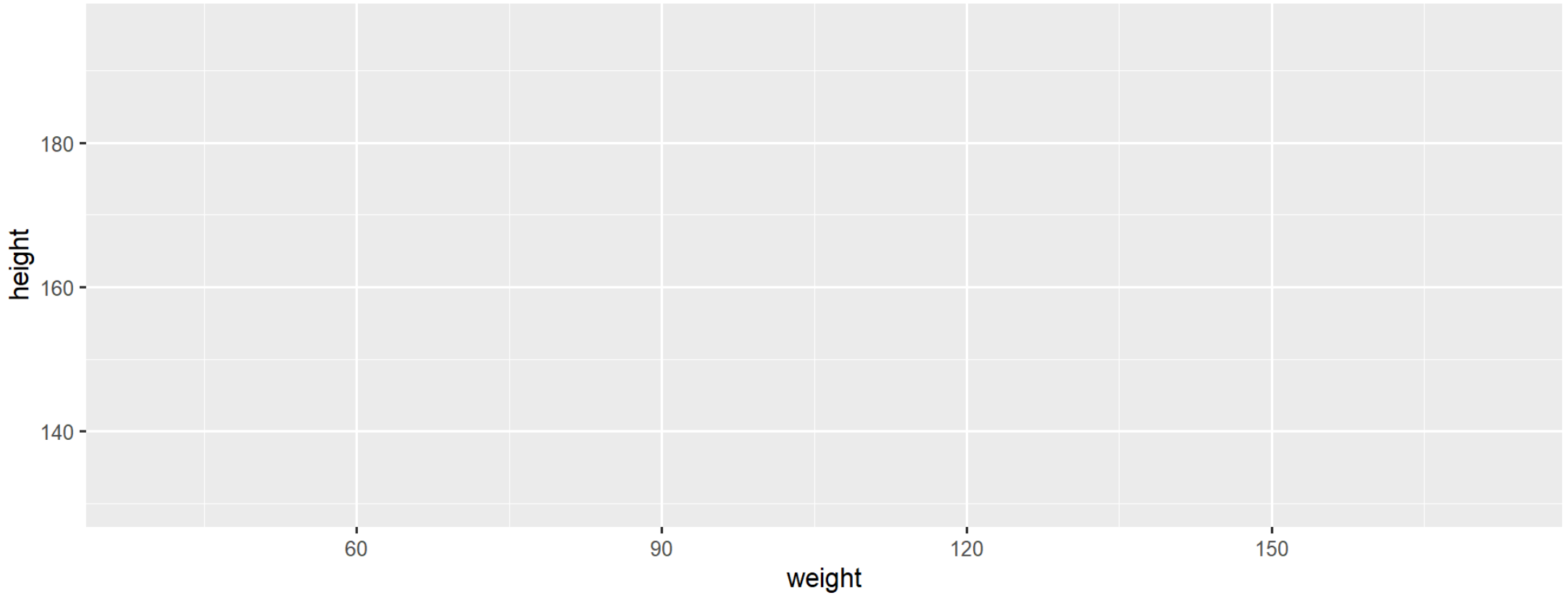
Each aesthetic can be mapped to a variable (or set to a constant value) by associating the name of the aesthetic to the name of the variable inside `aes()`

`ggplot2` automatically assigns a unique level of the aesthetic to each unique value of the variable by a process called scaling, and adds a legend that explains which levels correspond to which values.

R Code Plot

```
ggplot(
  data = tcga_clinical,
  mapping = aes(x = weight, y = height))
```

R Code | Plot



Geometries layer

Geometric object or `geom` defines the visual object and determines the type of graphs

R Code | Plot

```
ggplot(
  data = tcga_clinical,
  mapping = aes(x = weight, y = height)) +
  geom_point()
```

'+' is used to combine ggplot2 elements

Geometries layer

Geometric object or `geom` defines the visual object and determines the type of graphs

R Code | Plot

...revisiting aesthetic layer

Add additional variables to a plot by mapping a specific variable in the data to aesthetics options:

Aesthetic	Description
`fill`	fill color
`color`	color of points, outline of other geoms
`size`	area of point, thickness of line
`shape`	shape
`alpha`	transparency
`linetype`	line dash pattern

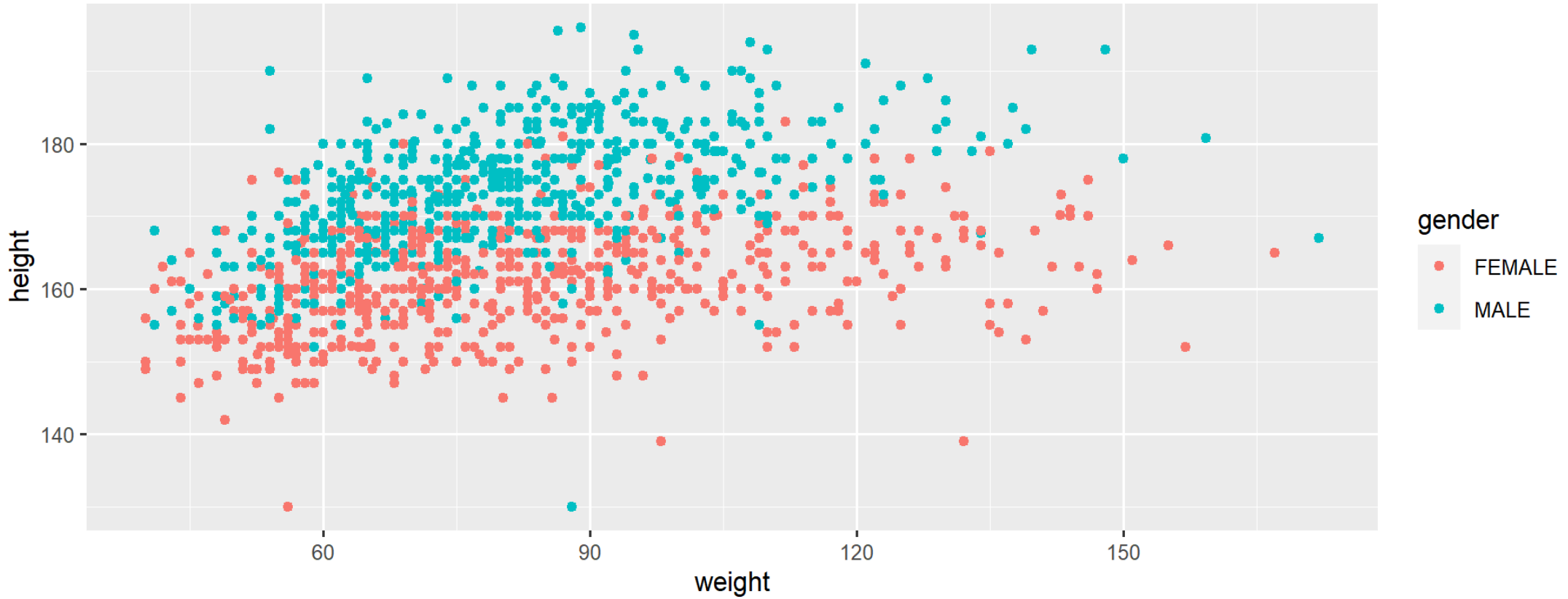
Mapping onto color aesthetic

R Code | Plot

```
ggplot(
  data = tcga_clinical,
  mapping = aes(x = weight,
                y = height,
                color = gender)) +
  geom_point()
```

Mapping onto color aesthetic

R Code | Plot



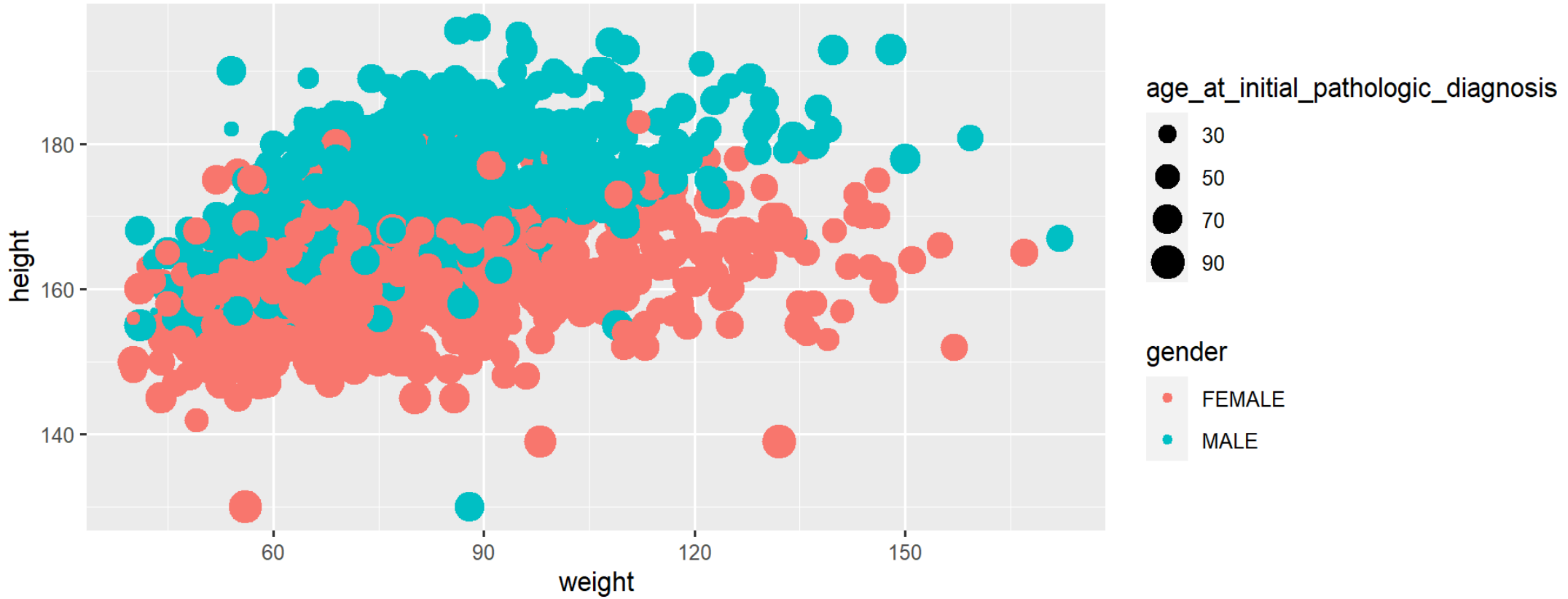
Mapping onto size aesthetic

R Code | Plot

```
ggplot(
  data = tcga_clinical,
  mapping = aes(x = weight,
                y = height,
                color = gender,
                size = age_at_initial_pathologic_diagnosis)) +
  geom_point()
```

Mapping onto size aesthetic

R Code | Plot



geom catalogue

Search:

geom	description
geom_abline(), geom_hline(), geom_vline()	Reference lines: horizontal, vertical, and diagonal
geom_bar(), geom_col()	Bar charts
geom_bin2d()	Heatmap of 2d bin counts
geom_blank()	Draw nothing
geom_boxplot()	A box and whiskers plot (in the style of Tukey)
geom_contour(), geom_contour_filled()	2D contours of a 3D surface

Previous 1 2 3 4 5 6 Next

Statistics layer

Statistical layer allows you to plot statistical values, typically summaries, calculated from the data.

Transforms input variables to displayed values in plot (e.g. count number of observations in each category of bar chart)

R Code | Plot

```
ggplot(
  data = tcga_clinical,
  mapping = aes(x = stage)) +
  geom_bar()
```

Statistics layer

Statistical layer allows you to plot statistical values, typically summaries, calculated from the data.

Transforms input variables to displayed values in plot (e.g. count number of observations in each category of bar chart)

R Code | Plot

Under the hood, data is transformed or new value is calculated by `stat` argument (short for statistical transformation) in `geom_*` functions

```
p1 <- ggplot(
  data = tcga_clinical,
  mapping = aes(x = stage)) +
  geom_bar()
```

```
p2 <- ggplot(
  data = tcga_clinical,
  mapping = aes(x = stage)) +
  geom_bar(stat = "count")
```

stat catalogue

Search:

stat	description
stat_count()	Bar charts
stat_bin_2d()	Heatmap of 2d bin counts
stat_boxplot()	A box and whiskers plot (in the style of Tukey)
stat_contour(), stat_contour_filled()	2D contours of a 3D surface
stat_sum()	Count overlapping points
stat_density()	Smoothed density estimates

Previous 1 2 3 4 Next

Scales layer

Scales control the aesthetic mapping between **data** and **aesthetics** and to control the aesthetic mapping, you can use a scale specification like

```
scale_aesthetics_type
```

Scale takes care of the details of converting data into aesthetics like size, color, position or shape

Scale translates back and forth between variable range and graphical property range

Scale is also responsible for creating a guide (axis or legend) which is needed to provide an inverse mapping, converting aesthetic values back into data values

Every aesthetic in a plot is associated with exactly one scale

Scale can be accessed by using function of form:

- `scale_x_*()`
  - `scale_x_continuous()`,`scale_x_log10()`
- `scale_y_*()`
- `scale_color_*()`
  - `scale_color_discrete()`
- `scale_fill_*()`
- `scale_shape_*()`
- `scale_linetype_*()`
  - `scale_linetype_manual()`
- `scale_size_*()`
- `scale_alpha_*()`

ggplot2 adds a default scale for each of the aesthetics used in the plot

R Code | Plot

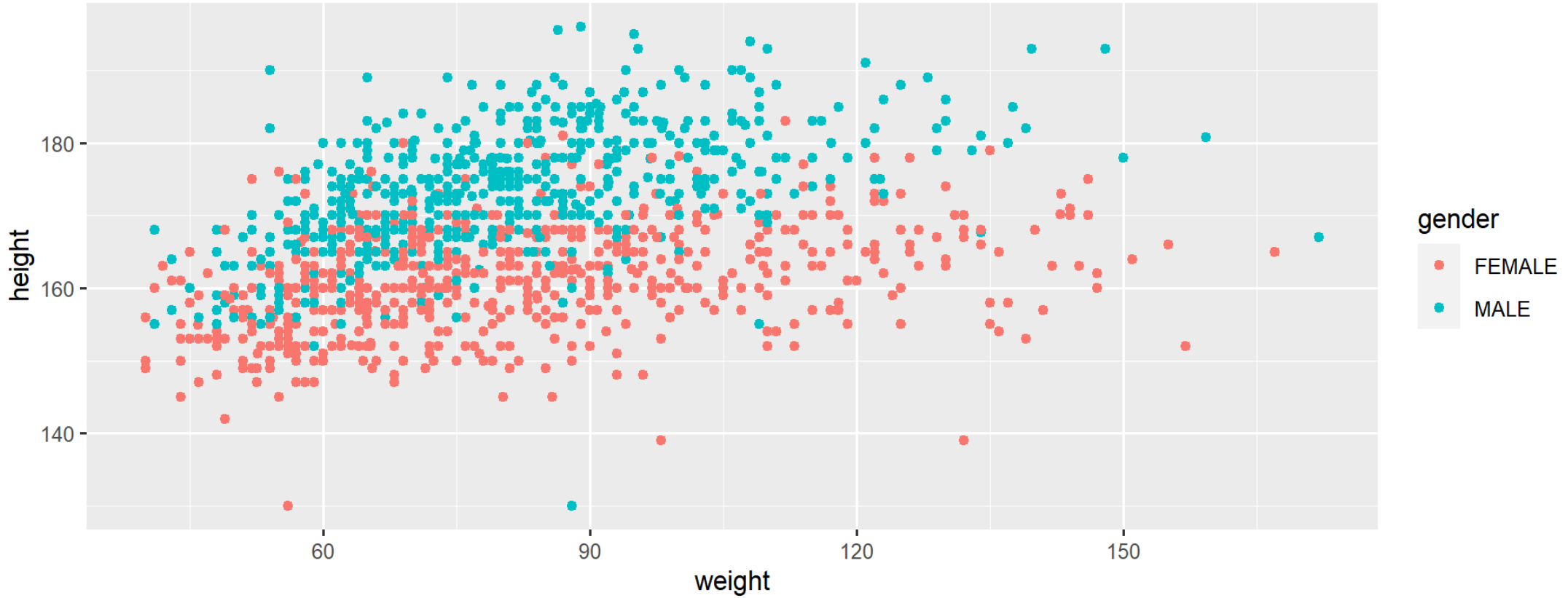
```
ggplot(
  data = tcga_clinical,
  aes(x = weight, y = height, color = gender)) +
  geom_point()
```

...is equivalent to

```
ggplot(
  data = tcga_clinical,
  mapping = aes(x = weight, y = height, color = gender)) +
  geom_point() +
  scale_x_continuous() +
  scale_y_continuous() +
  scale_color_discrete()
```

ggplot2 adds a default scale for each of the aesthetics used in the plot

R Code | Plot



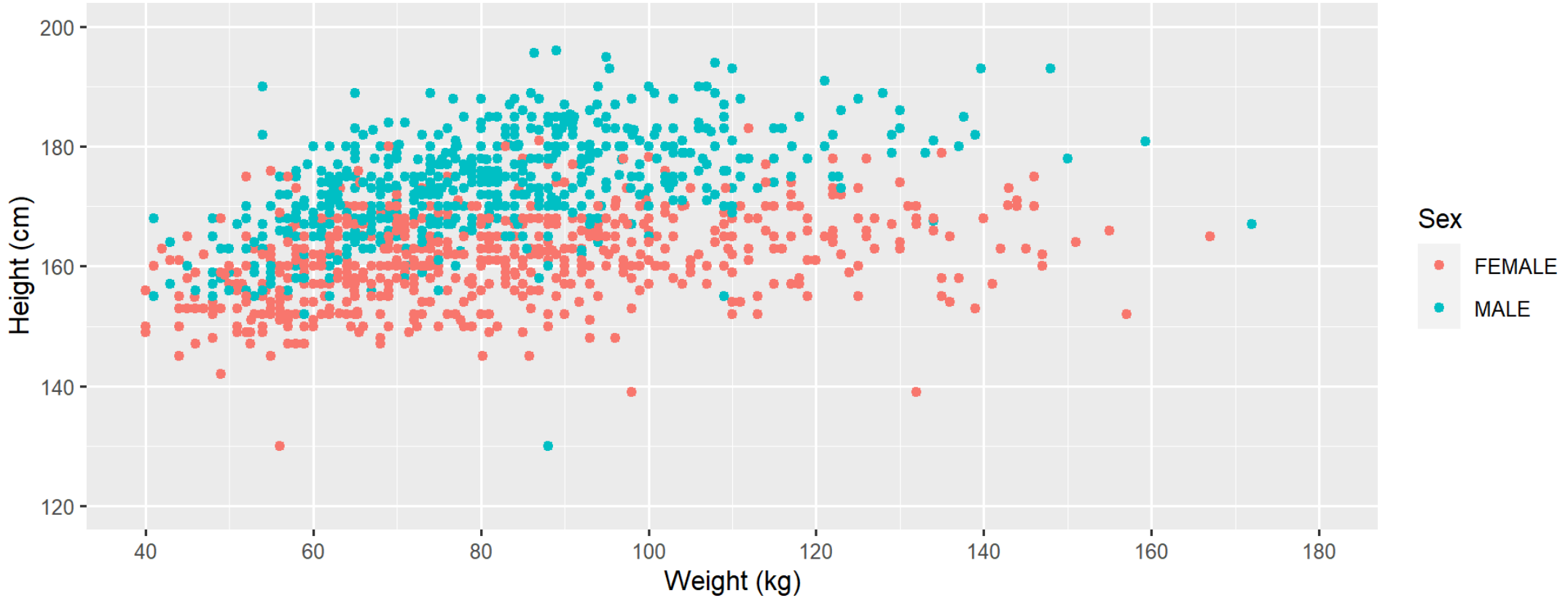
We can override the defaults by adding the `scale` function

R Code | Plot

```
ggplot(
  data = tcga_clinical,
  mapping = aes(x = weight, y = height, color = gender)) +
  geom_point() +
  scale_x_continuous(
    "Weight (kg)", limits = c(40, 180), breaks = seq(40, 180, 20)) +
  scale_y_continuous(
    "Height (cm)", limits = c(120, 200), breaks = seq(120, 200, 20)) +
  scale_color_discrete(
    "Sex", na.value = "red")
```

We can override the defaults by adding the `scale` function

R Code | **Plot**



Facet layer

Facet layer allows us to create subplots within the same graphic object

It splits the data into subsets and displays the same graph for each subset

Before {ggplot2} v3.0.0 the variables used for faceting could only be passed as a formula notation (<variable> ~ <variable>) but now the variables used for faceting can be passed wrapped in vars().

```
ggplot(
  data = tcga_clinical,
  mapping = aes(x = weight, y = height)) +
  geom_point() +
  # facet_grid(gender ~ tumor_tissue_site)
  facet_grid(rows = vars(gender), cols = vars(tumor_tissue_site))
```

There are two types of facetting:

- `facet_wrap()`: wraps a 1D ribbon of panels into 2D

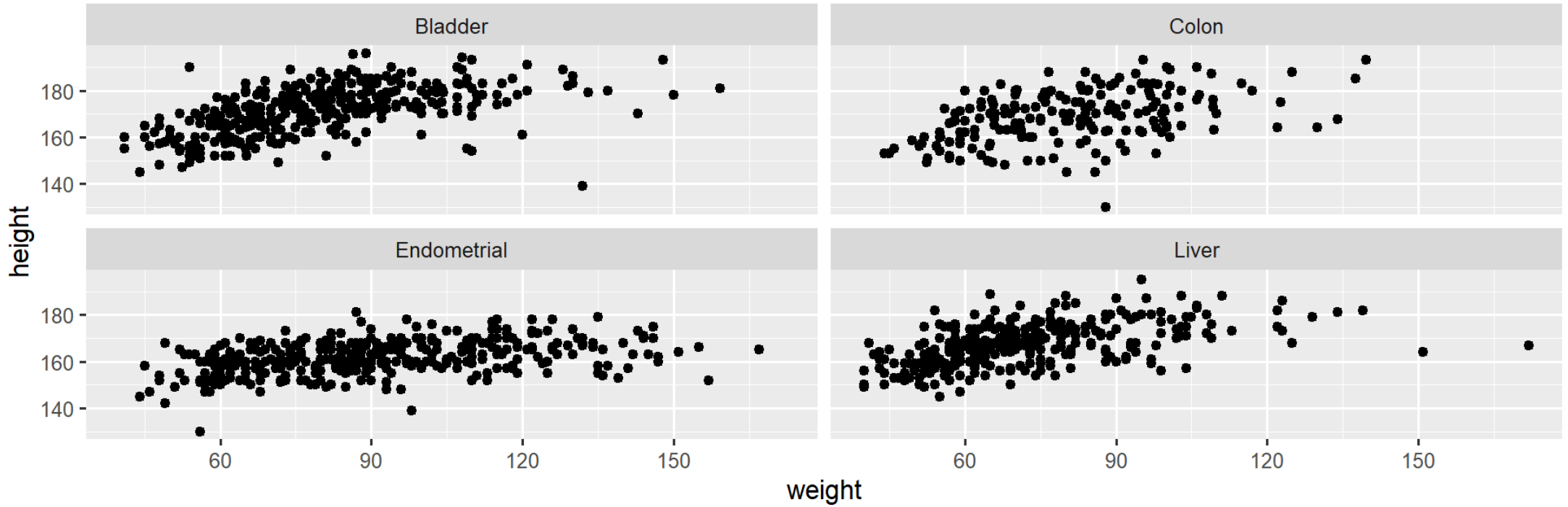
R Code | Plot

```
ggplot(
  data = tcga_clinical,
  mapping = aes(x = weight, y = height)) +
  geom_point() +
  # facet_wrap(~ tumor_tissue_site, ncol = 2)
  facet_wrap(vars(tumor_tissue_site), ncol = 2)
```

There are two types of facetting:

- `facet_wrap()`: wraps a 1D ribbon of panels into 2D

R Code | Plot



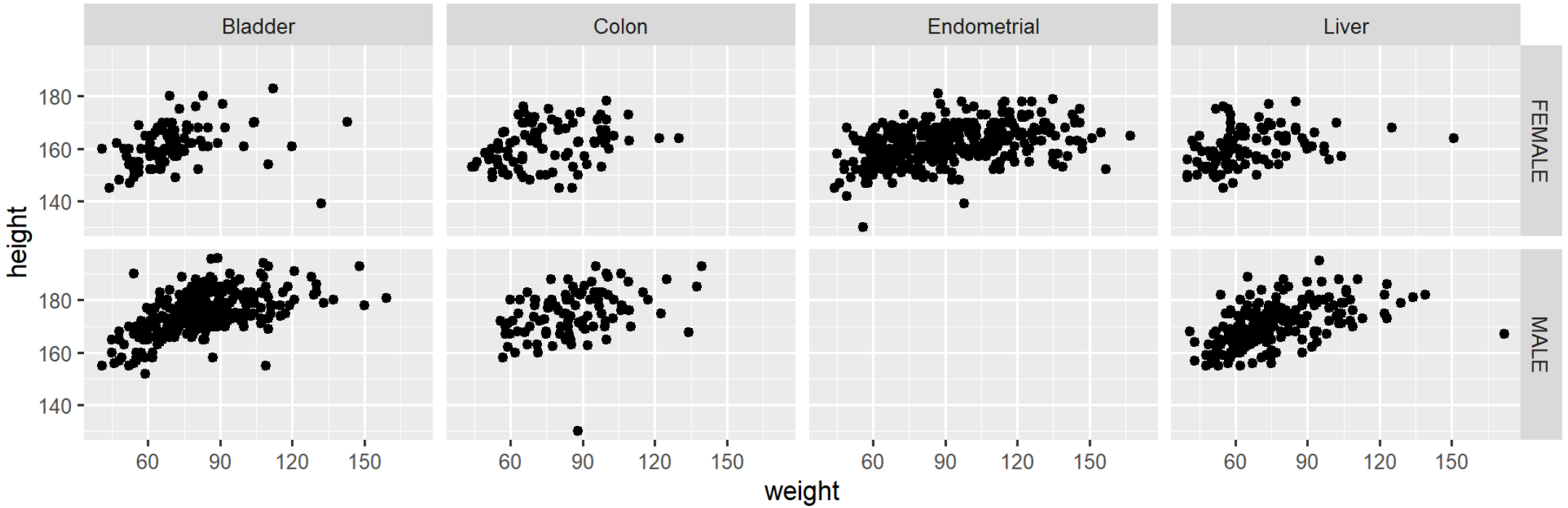
- `facet_grid()`: produces a 2D grid of panels defined by variables which form the rows and columns

R Code | Plot

```
ggplot(
  data = tcga_clinical,
  mapping = aes(x = weight, y = height)) +
  geom_point() +
  # facet_grid(gender ~ tumor_tissue_site)
  facet_grid(rows = vars(gender), cols = vars(tumor_tissue_site))
```

- `facet_grid()`: produces a 2D grid of panels defined by variables which form the rows and columns

R Code | Plot



Theme layer

Controls all non-data plot elements and appearance

Visual elements that are not part of the data:

Type	Modified using
text	`element_text()`
line	`element_line()`
rectangle	`element_rect()`

`theme()` function lets us override the default theme elements by calling above element functions

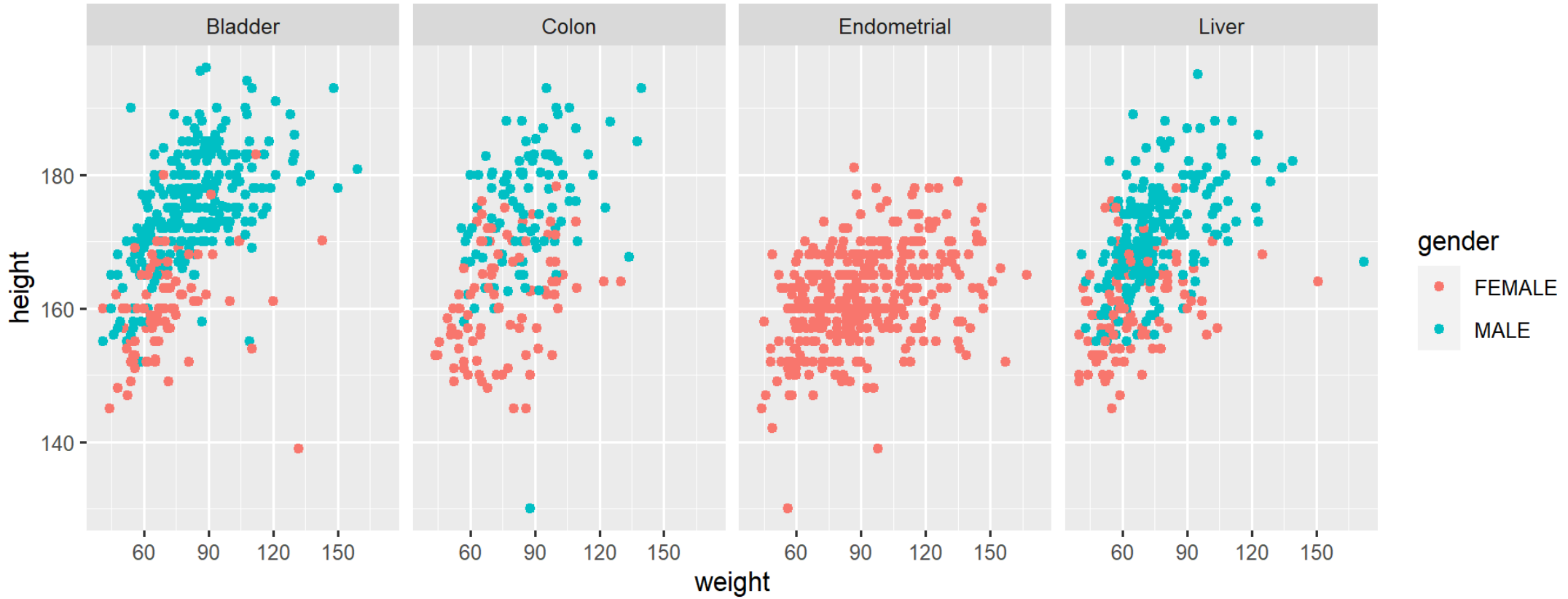
Default theme

R Code | Plot

```
ggplot(
  data = tcga_clinical,
  mapping = aes(x = weight, y = height, color = gender)) +
  geom_point() +
  facet_wrap(~ tumor_tissue_site, nrow = 1)
```

Default theme

R Code | Plot



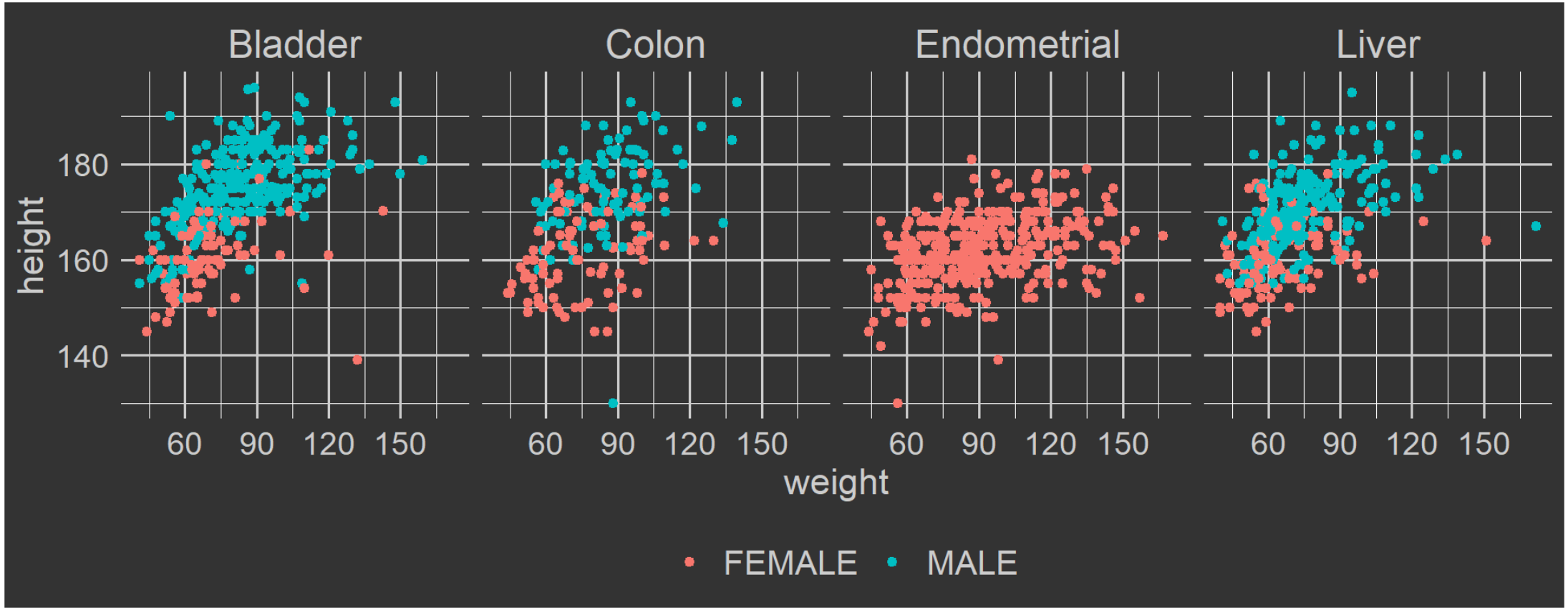
Changing theme elements

R Code | Plot

```
ggplot(
  data = tcga_clinical,
  mapping = aes(x = weight, y = height, color = gender)) +
  geom_point() +
  facet_wrap(~ tumor_tissue_site, nrow = 1) +
  theme(axis.title = element_text(size = 15, color = "gray80"),
        axis.text = element_text(size = 13, color = "gray80"),
        strip.text = element_text(size = 16, color = "gray80"),
        strip.background = element_blank(),
        legend.background = element_blank(),
        legend.key = element_blank(),
        legend.text = element_text(size = 14, color = "gray80"),
        legend.title = element_blank(),
        legend.position = "bottom",
        panel.grid.major = element_line(color = "gray80"),
        panel.background = element_rect(fill = "gray20"),
        plot.background = element_rect(fill = "gray20"))
```

Changing theme elements

R Code | Plot



We can avoid the `theme()` function by using built-in themes

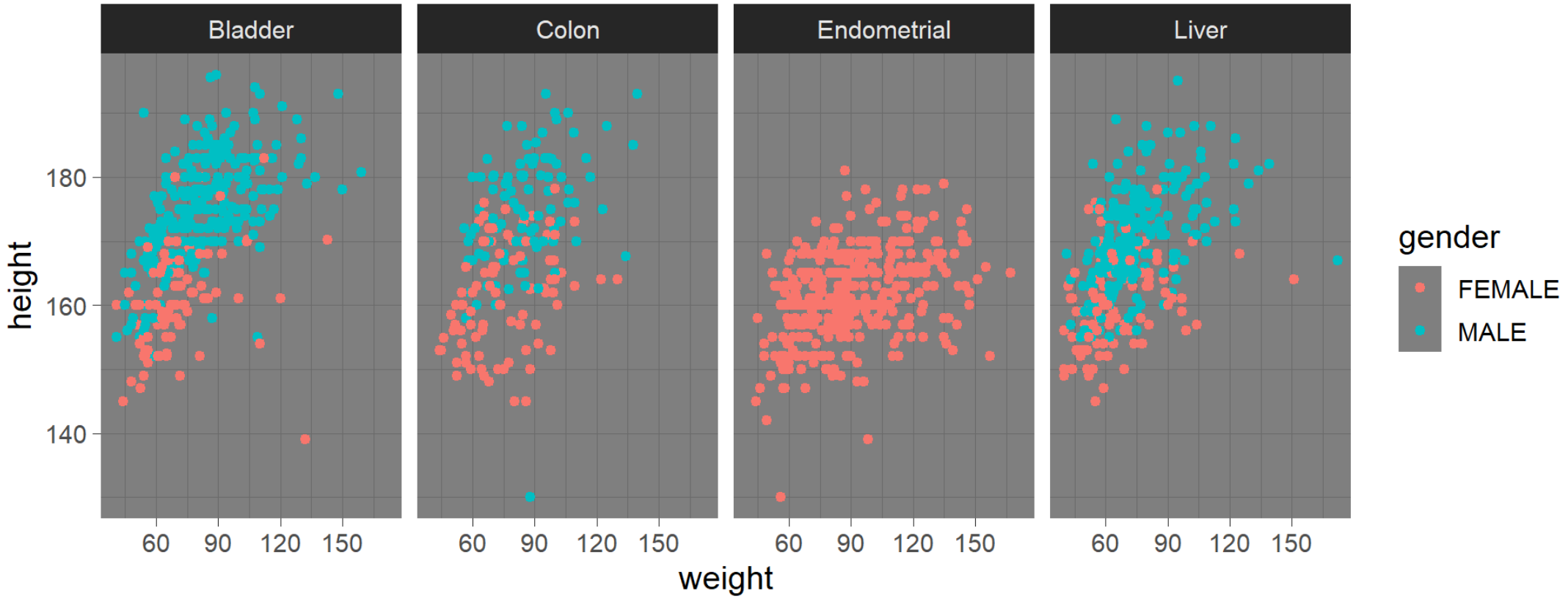
R Code | Plot

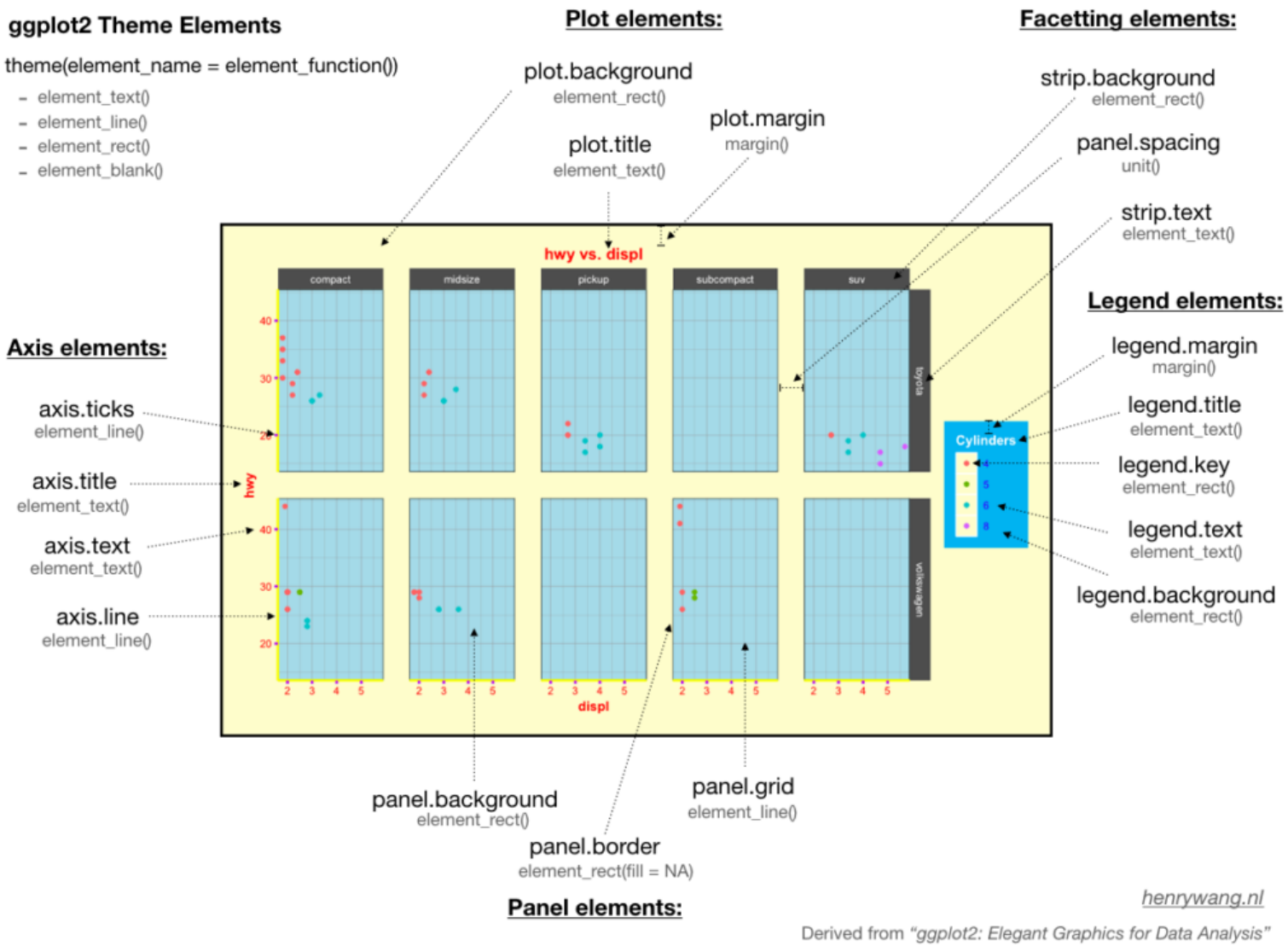
```
ggplot(
  data = tcga_clinical,
  mapping = aes(x = weight, y = height, color = gender)) +
  geom_point() +
  facet_wrap(~ tumor_tissue_site, nrow = 1) +
  theme_dark(base_size = 13)
```

Set a default theme for the rest of your plots using `theme_set()` at the top of your script (e.g. `theme_set(theme_dark())`)

We can avoid the `theme()` function by using built-in themes

R Code | Plot





Source: https://henrywang.nl/ggplot2-theme-elements-demonstration/

Global vs local aesthetic mapping

Global

inside `ggplot()` call

```
ggplot(
  data = tcga_clinical,
  mapping = aes(x = weight,
                y = height,
                color = gender))
  geom_point()
```

Local

inside `geom_*()` layer

```
ggplot(
  data = tcga_clinical) +
  geom_point(
    mapping = aes(x = weight,
                  y = height,
                  color = gender)
```

if there is only one layer in the plot, the way aesthetics are specified doesn't make any difference

However, when we start adding more layers, the distinction becomes more important

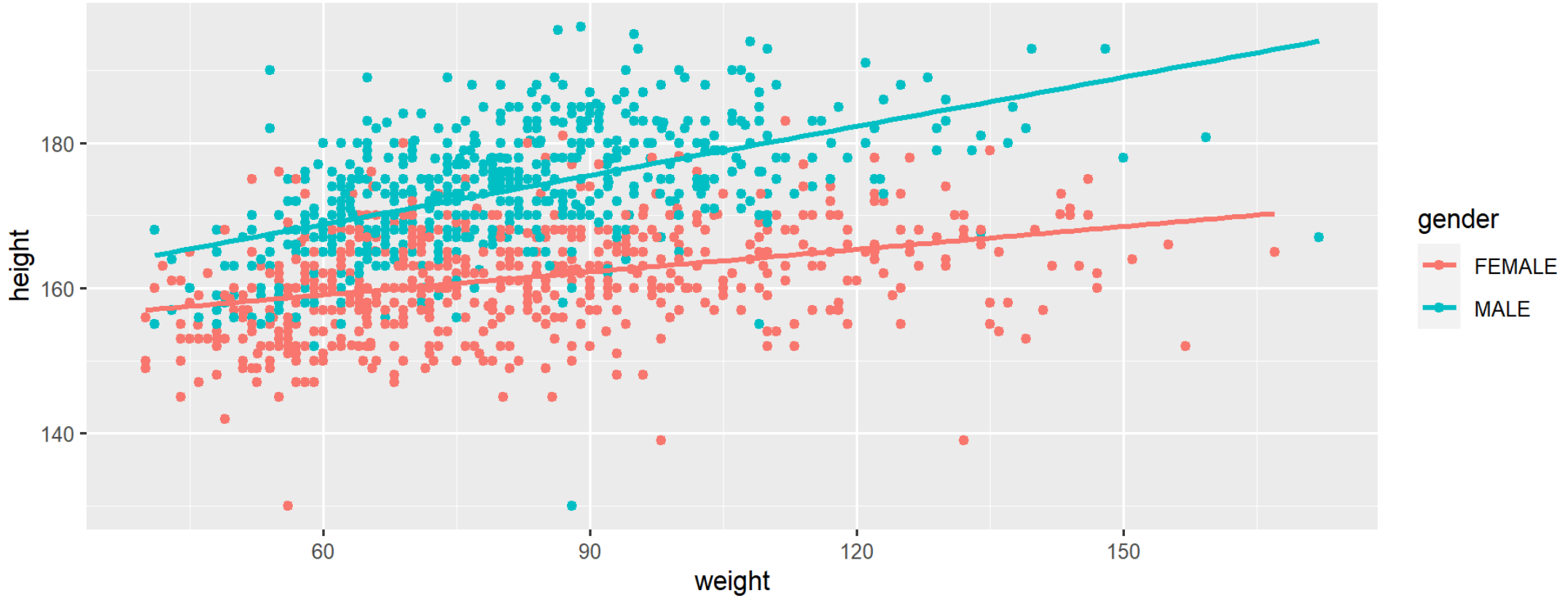
R Code | Plot

```
ggplot(
  data = tcga_clinical,
  mapping = aes(x = weight,
                y = height,
                color = gender)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)
```

`ggplot2` treats aesthetics defined in `ggplot()` call as global mappings and applies them to each geom in the graph

However, when we start adding more layers, the distinction becomes more important

R Code | Plot



.. and `ggplot2` treats aesthetics defined in `geom_*()` function as local mappings and use them to extend or overwrite the global mappings for that layer only

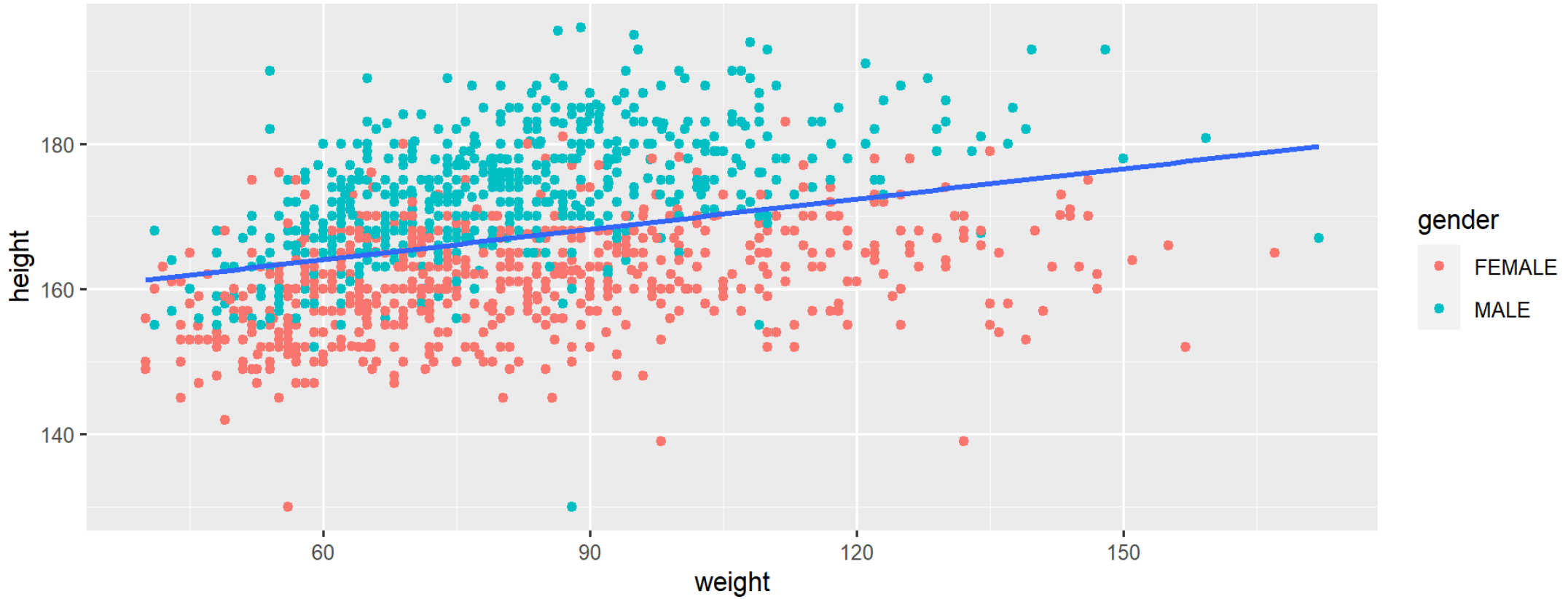
R Code | Plot

```
ggplot(
  data = tcga_clinical,
  mapping = aes(x = weight,
                y = height)) +
  geom_point(aes(color = gender)) +
  geom_smooth(method = "lm", se = FALSE)
```

Since `color` aesthetic is defined inside `geom_point()` only, this mapping is applied to `point` geom but not to `smoother` geom

.. and `ggplot2` treats aesthetics defined in `geom_*()` function as local mappings and use them to extend or overwrite the global mappings for that layer only

R Code | **Plot**



Setting vs mapping aesthetics

Aesthetic property can also be set to a single value (constant) by specifying it in the layer parameters.

- **map** an aesthetic to a variable inside `aes()`

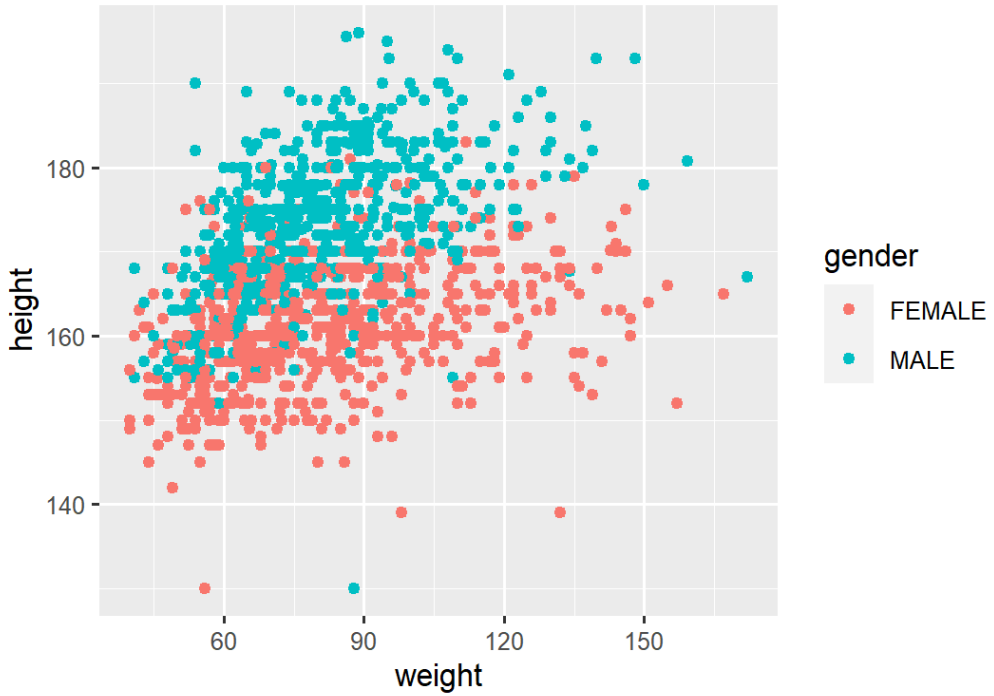
```
... +
geom_point(aes(color = gender))
```

- **set** an aesthetic to a constant outside `aes()`

```
... +
geom_point(color = "blue")
```

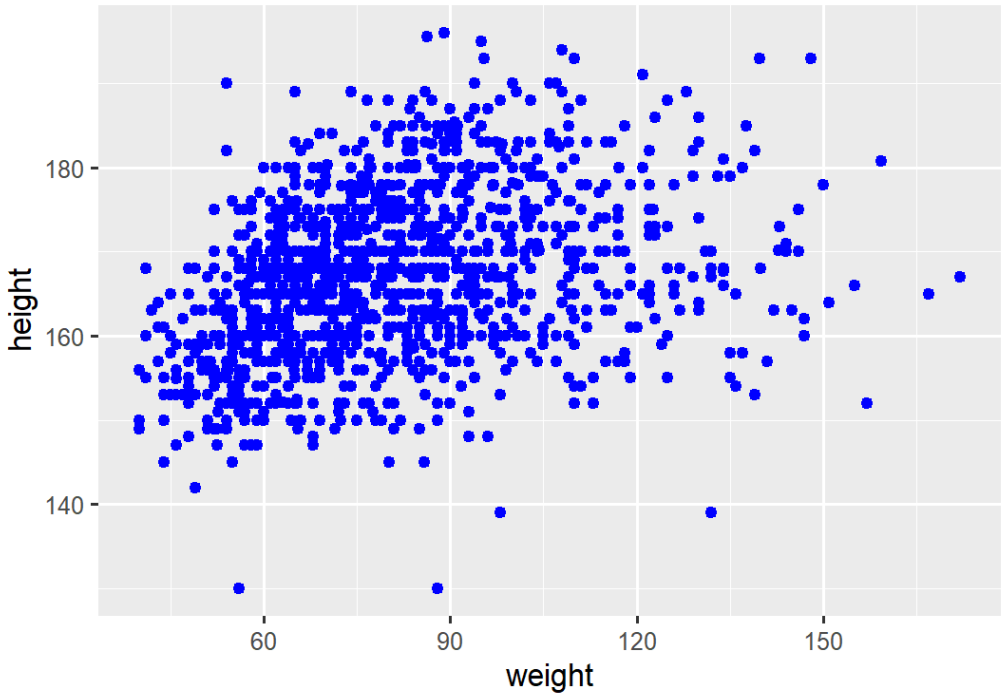
Mapping

```
ggplot(
  data = tcga_clinical,
  mapping = aes(x = weight,
                y = height)) +
  geom_point(aes(color = gender))
```



Setting

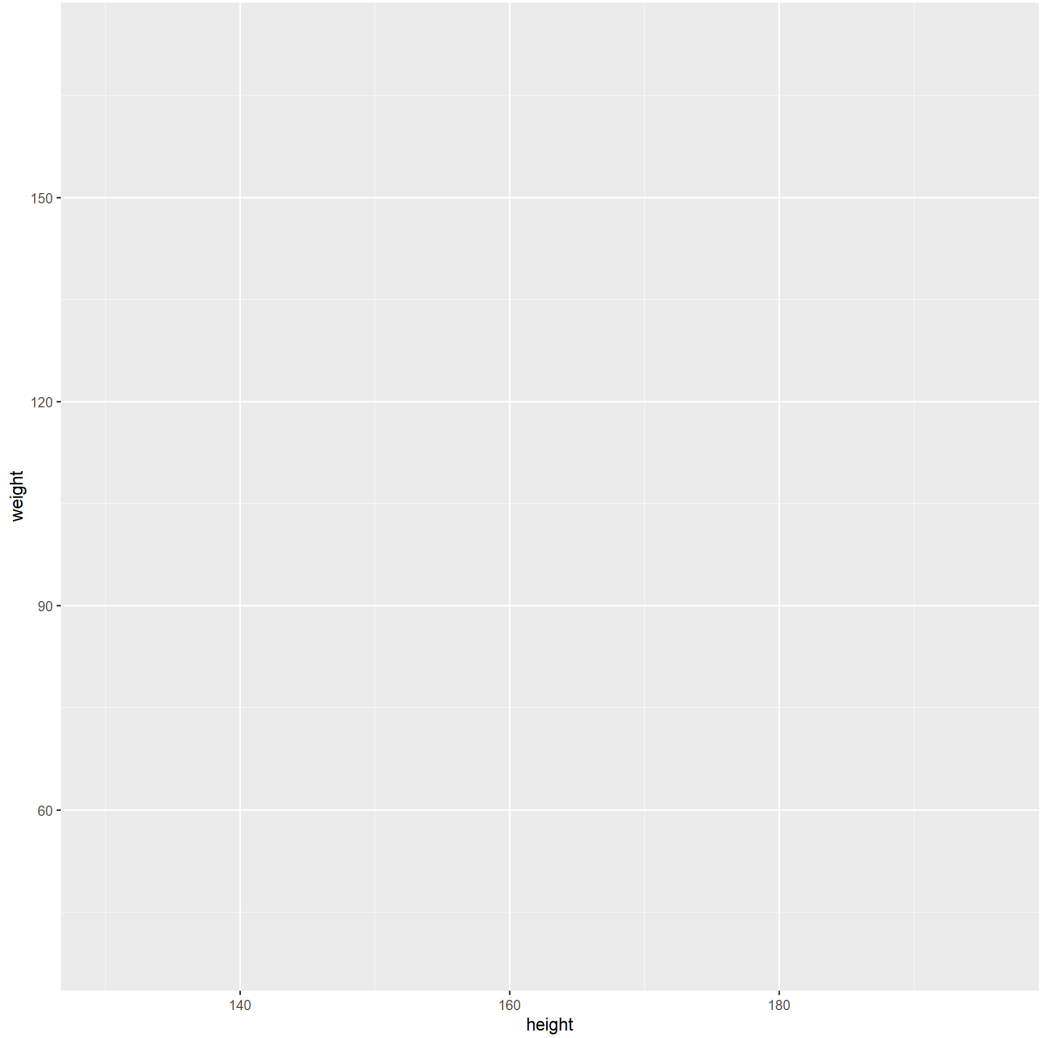
```
ggplot(
  data = tcga_clinical,
  mapping = aes(x = weight,
                y = height)) +
  geom_point(color = "blue")
```



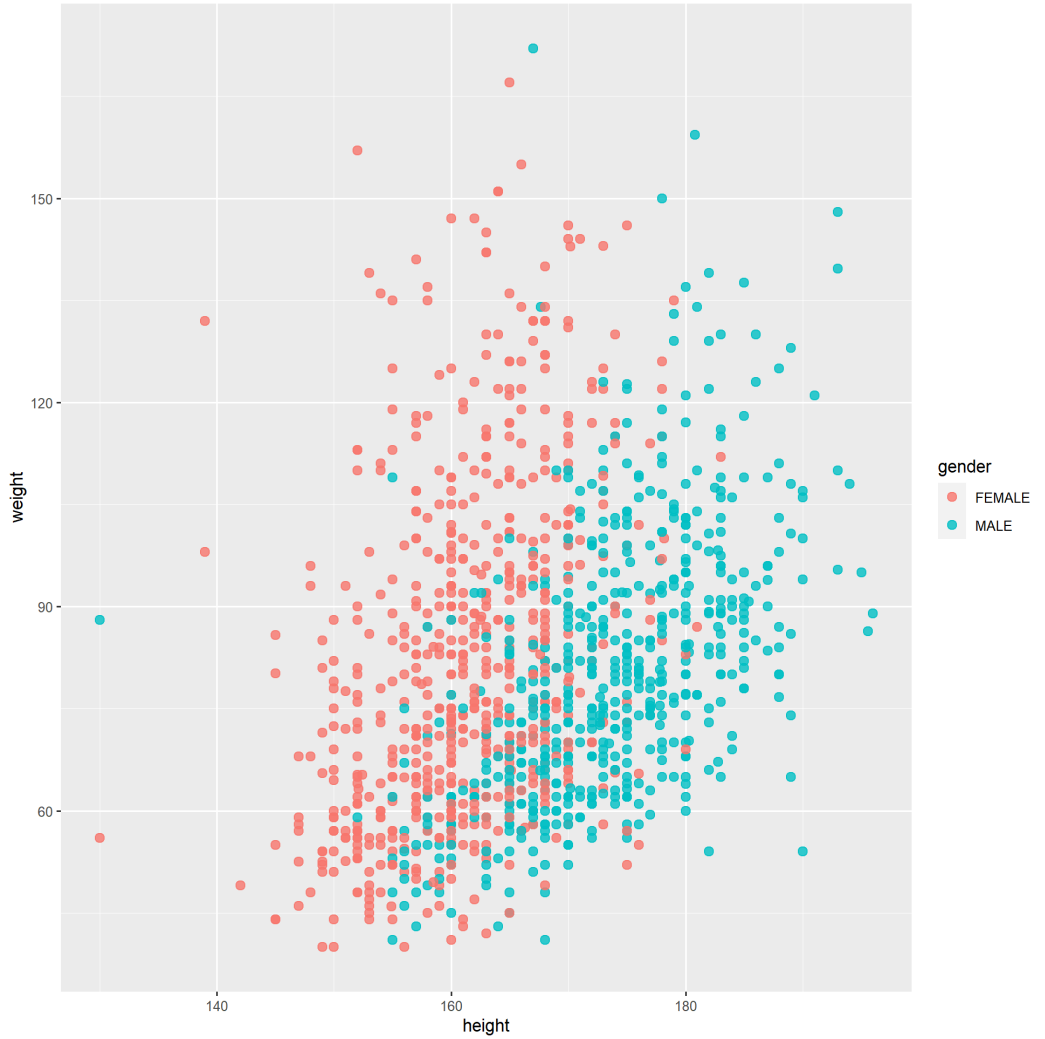
ggplot2 codes step-by-step

```
ggplot(
  data = tcga_clinical)
```

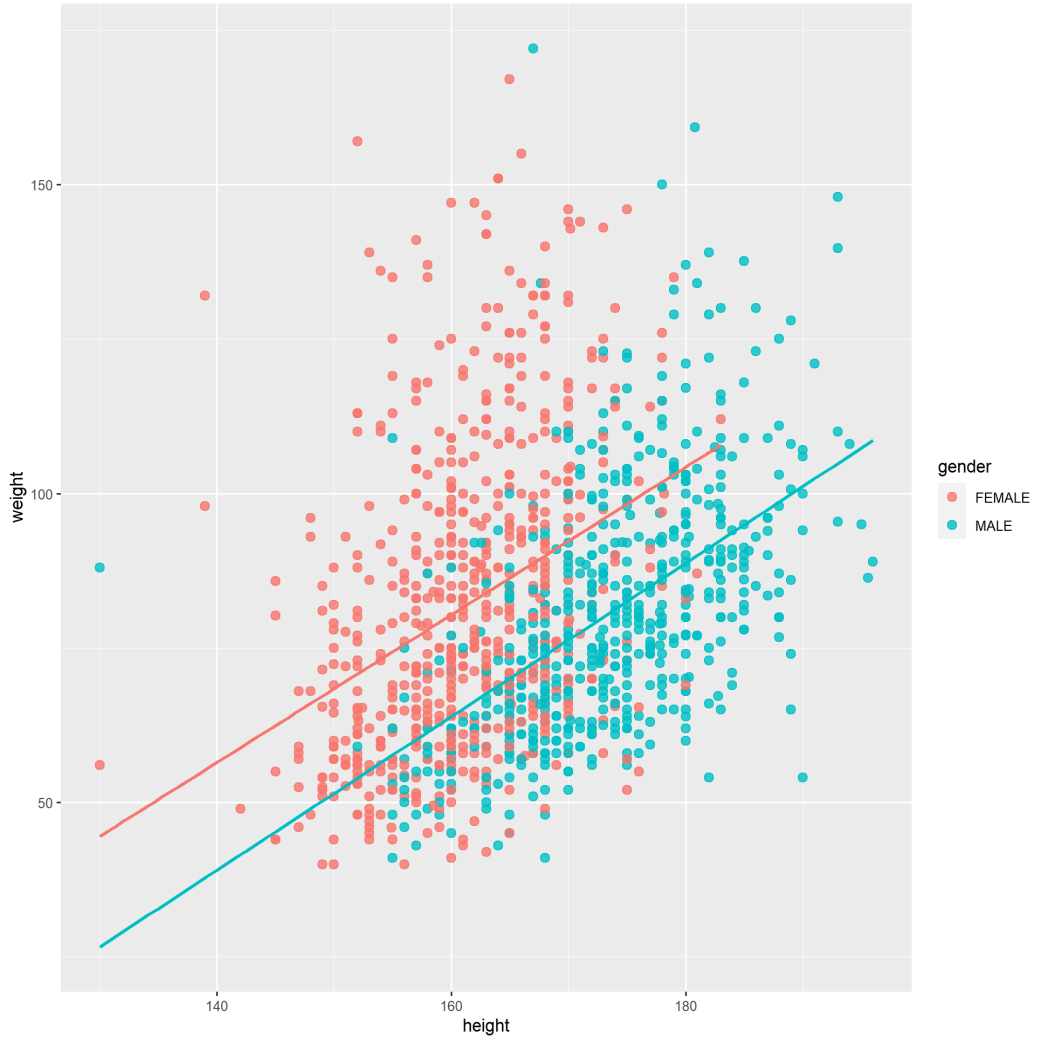
```
ggplot(
  data = tcga_clinical) +
  aes(
    x = height, y = weight,
    color = gender)
```



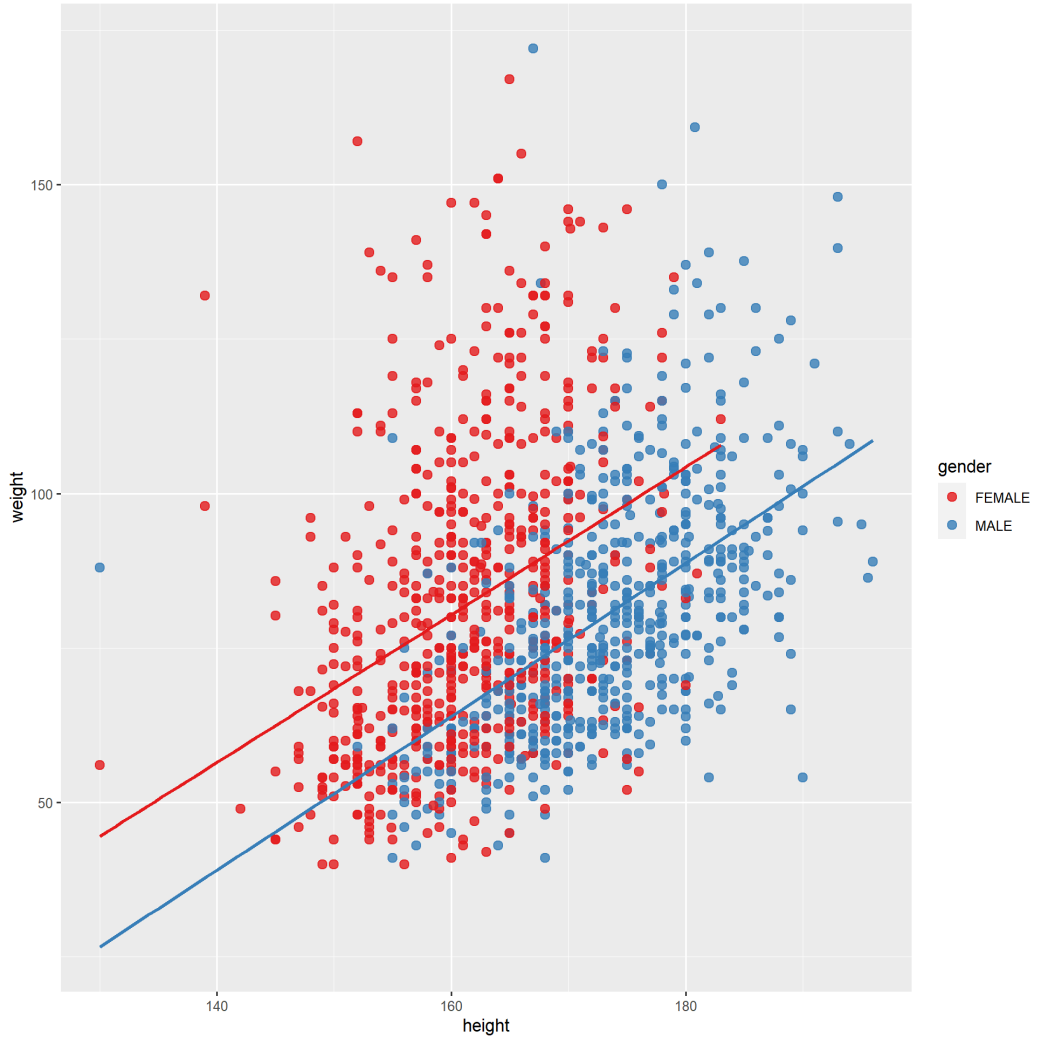
```
ggplot(
  data = tcga_clinical) +
  aes(
    x = height, y = weight,
    color = gender) +
  geom_point(
    size = 2.5, alpha = 0.8)
```



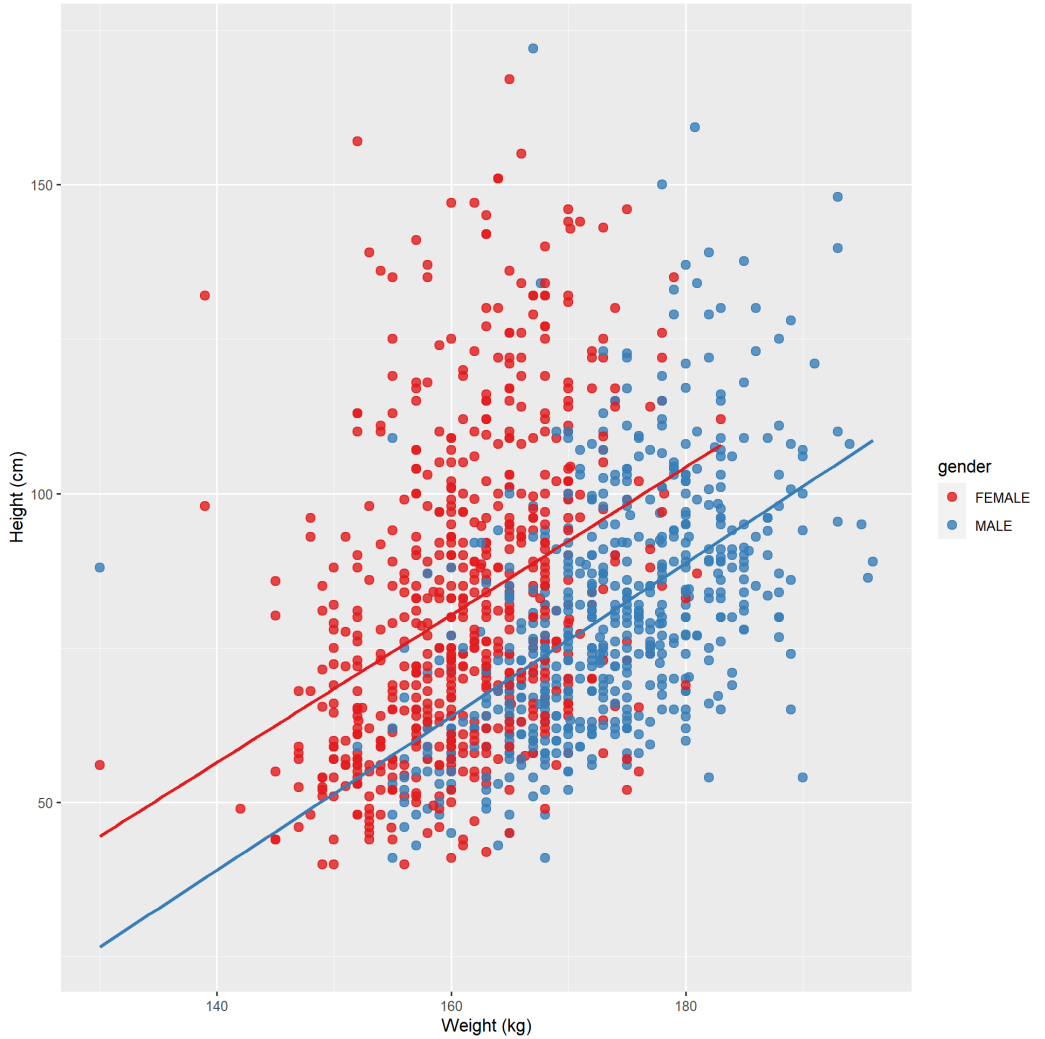
```
ggplot(
  data = tcga_clinical) +
  aes(
    x = height, y = weight,
    color = gender) +
  geom_point(
    size = 2.5, alpha = 0.8) +
  geom_smooth(
    method = "lm", se = FALSE,
    show.legend = FALSE)
```



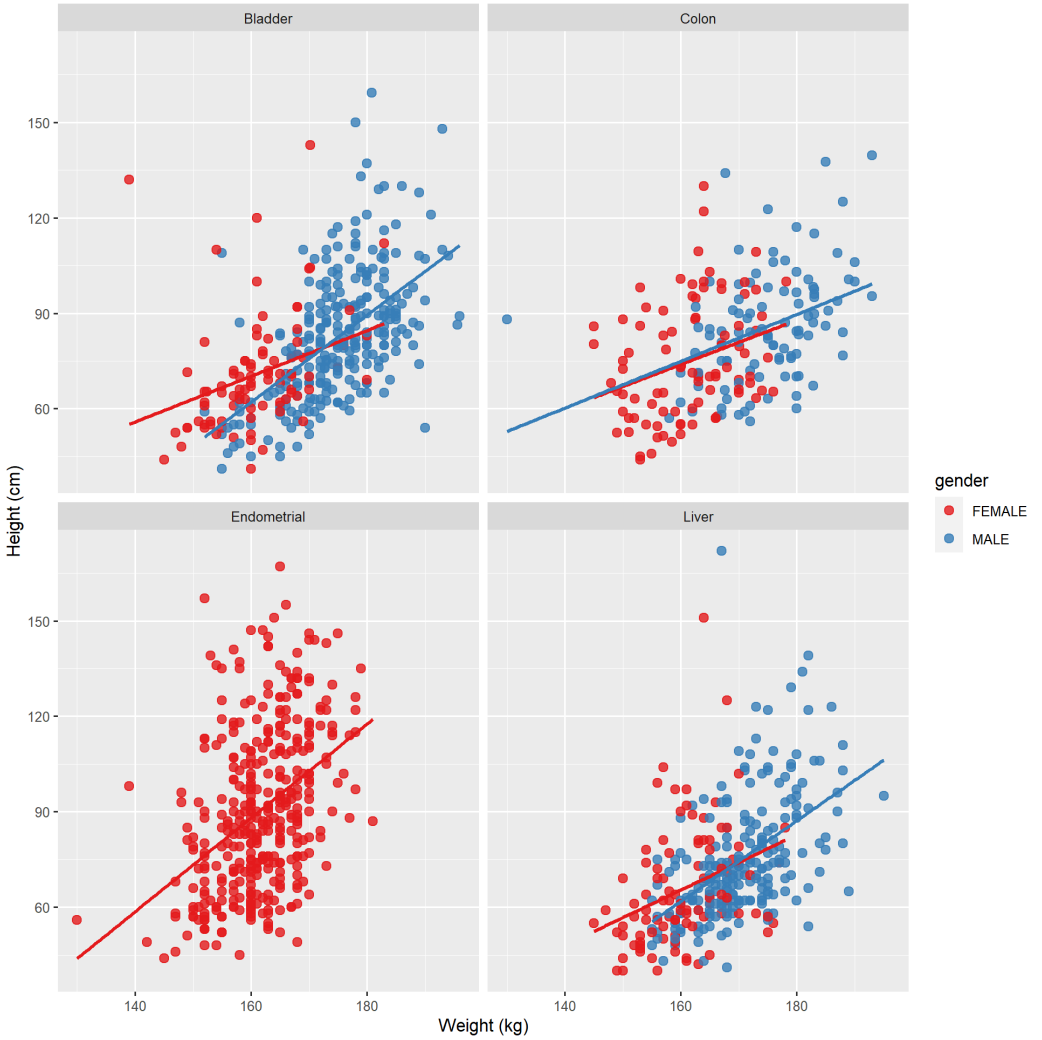
```
ggplot(
  data = tcga_clinical) +
  aes(
    x = height, y = weight,
    color = gender) +
  geom_point(
    size = 2.5, alpha = 0.8) +
  geom_smooth(
    method = "lm", se = FALSE,
    show.legend = FALSE) +
  scale_color_brewer(
    palette = "Set1")
```



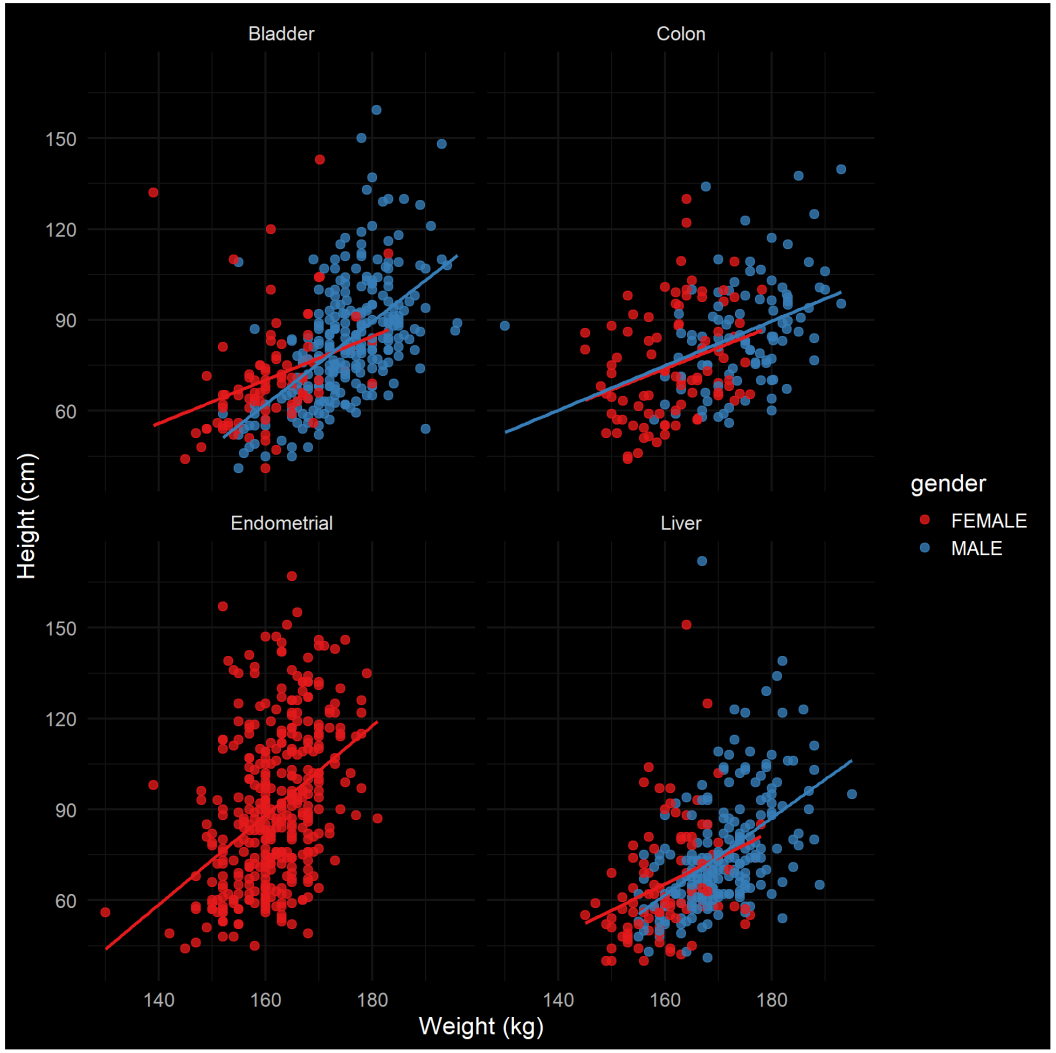
```
ggplot(
  data = tcga_clinical) +
  aes(
    x = height, y = weight,
    color = gender) +
  geom_point(
    size = 2.5, alpha = 0.8) +
  geom_smooth(
    method = "lm", se = FALSE,
    show.legend = FALSE) +
  scale_color_brewer(
    palette = "Set1") +
  labs(
    x = "Weight (kg)",
    y = "Height (cm)")
```



```
ggplot(
  data = tcga_clinical) +
  aes(
    x = height, y = weight,
    color = gender) +
  geom_point(
    size = 2.5, alpha = 0.8) +
  geom_smooth(
    method = "lm", se = FALSE,
    show.legend = FALSE) +
  scale_color_brewer(
    palette = "Set1") +
  labs(
    x = "Weight (kg)",
    y = "Height (cm)") +
  facet_wrap(~ tumor_tissue_site)
```



```
ggplot(
  data = tcga_clinical) +
  aes(
    x = height, y = weight,
    color = gender) +
  geom_point(
    size = 2.5, alpha = 0.8) +
  geom_smooth(
    method = "lm", se = FALSE,
    show.legend = FALSE) +
  scale_color_brewer(
    palette = "Set1") +
  labs(
    x = "Weight (kg)",
    y = "Height (cm)") +
  facet_wrap(~ tumor_tissue_site)
  dark_theme_minimal(
    base_size = 15)
```



Useful resources

- ggplot2 cheatsheet from RStudio
- ggplot2 documentation
- ggplot2: Elegant Graphics for Data Analysis, 3rd edition
- Data visualization chapter from R for Data Science book
- Gallery of code examples of graphs made with `{ggplot2}` package