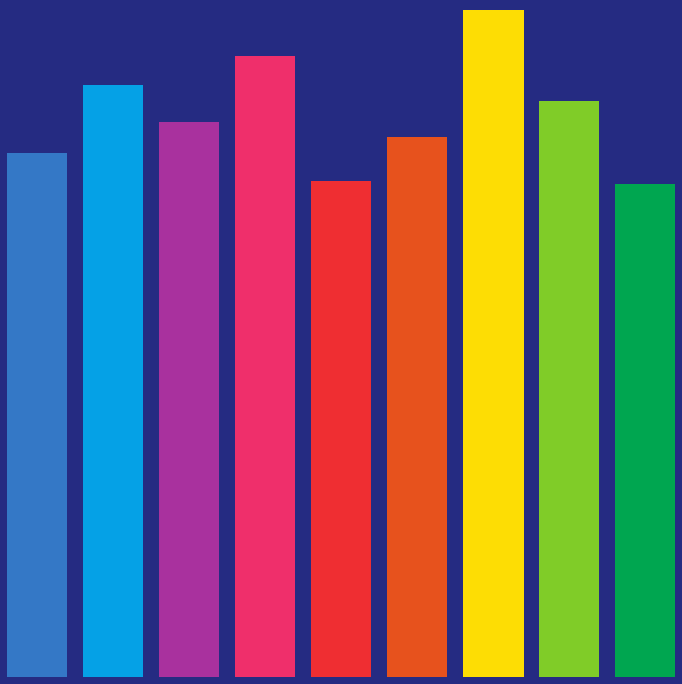
OECDPLOT REFERENCE MANUAL



Mauricio Vargas S.

OECDPLOT REFERENCE MANUAL

MAURICIO VARGAS SEPÚLVEDA

OECDPLOT REFERENCE MANUAL

Mauricio Vargas Sep'ulveda

This version was published on 2023-02-11.

Contents

What to expect from this book

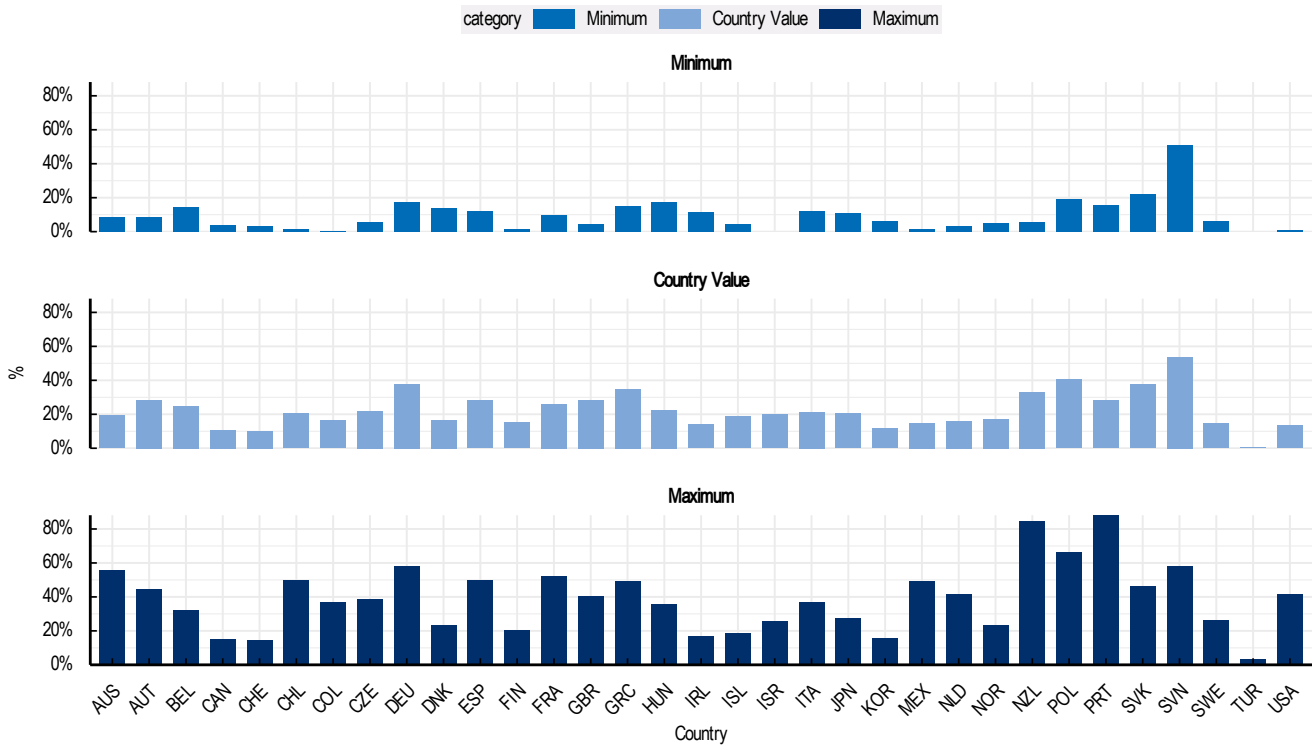
This is a technical book. The book aims to get straight to the point, and the writing style is similar to a recipe with detailed instructions. It is assumed that you know the basics of R and that you want to learn to create beautiful plots according to OECD style rules.

Every chapter is self contained. You can read the whole book or go to a chapter or section of your interest, and we are sure that it will be easy to understand the instructions and reproduce our examples without reading the earlier chapters.

CHAPTER 1

Column plots

We will work towards creating the area plot below. We will take you from a basic bar plot and explain all the customisations we add to the code step-by-step.



1.1. Basic graph

You can use fonts such as Arial Narrow within ggplot2. This package allows that with a dedicated function. The first thing to do is load in the libraries and data, as below:

```
library(oecdplot)
library(ggplot2)

load_oecd_fonts()
```

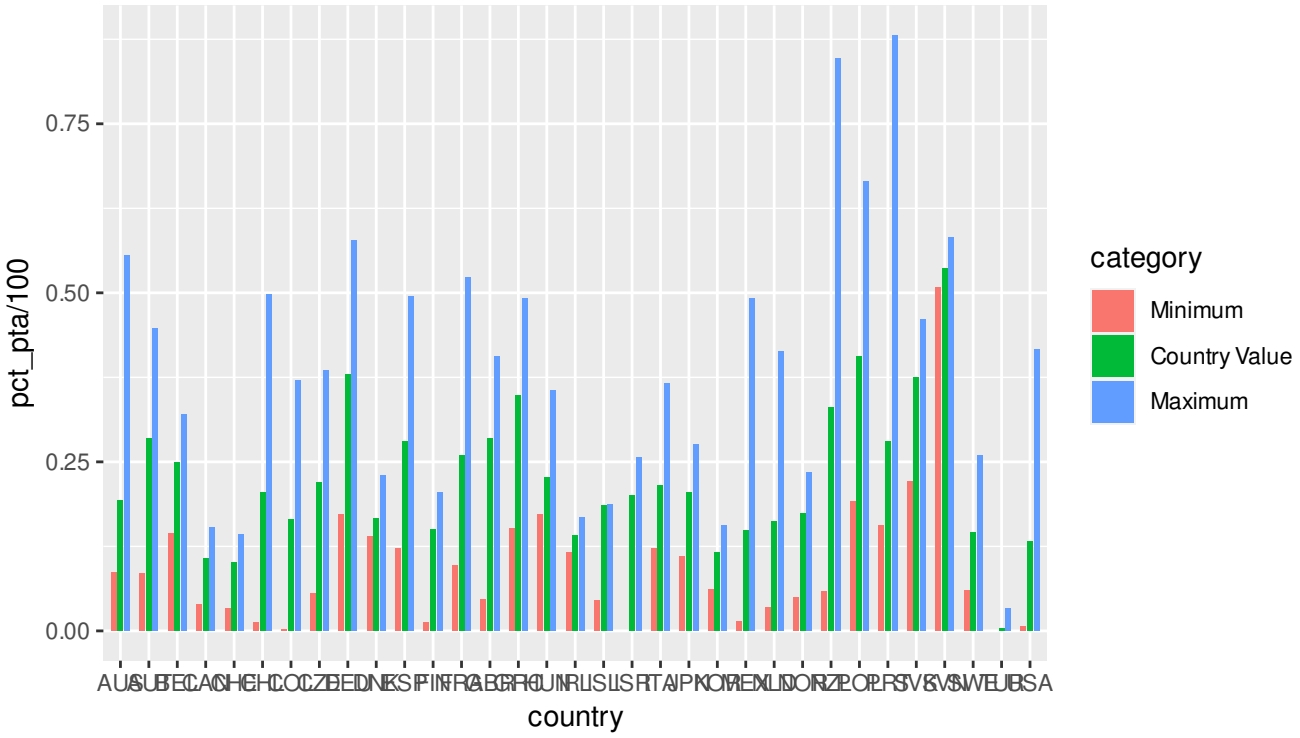
We will be working with the `pta` dataset, which is included in the package.

```
pta
```

```
# A tibble: 99 x 4
   country region         category      pct_pta
   <fct>   <fct>          <fct>           <dbl>
 1 NZL     Auckland       Minimum          5.77
 2 NZL     Country Value  Country Value   33.0
 3 NZL     West Coast     Maximum         84.6
 4 PRT     Centro         Minimum         15.6
 5 PRT     Country Value  Country Value   28.0
 6 PRT     Algarve        Maximum         88.1
 7 CHL     Coquimbo       Minimum          1.3
 8 CHL     Country Value  Country Value   20.5
 9 CHL     Aysén          Maximum         49.8
10 MEX     Guerrero       Minimum          1.47
# ... with 89 more rows
```

To initialise a plot we tell ggplot that `pta` is our data, and specify the variables on each axis. We then instruct ggplot to render this as an bar plot by adding the `geom_col()` function. The `position` argument makes the categories appear side-by-side, instead of stacking them on top of each other and the `width` argument makes the columns thinner.

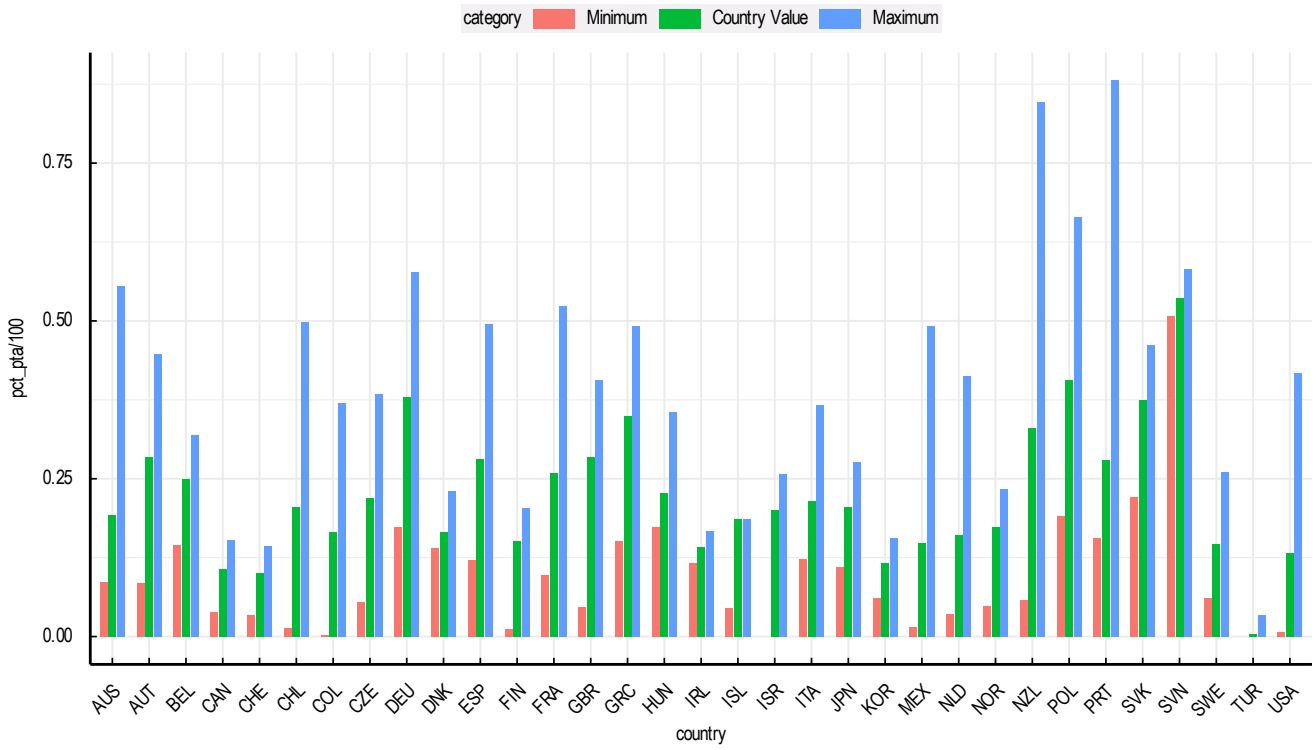
```
p <- ggplot(data = pta, aes(x = country, y = pct_pta / 100, fill = category)) +
  geom_col(position = "dodge2", width = 0.7)
p
```



1.2. Adjusting theme

We can change the overall look of the graph using themes. We'll start using a simple theme customisation by adding `theme_oecd()` after `ggplot()`.

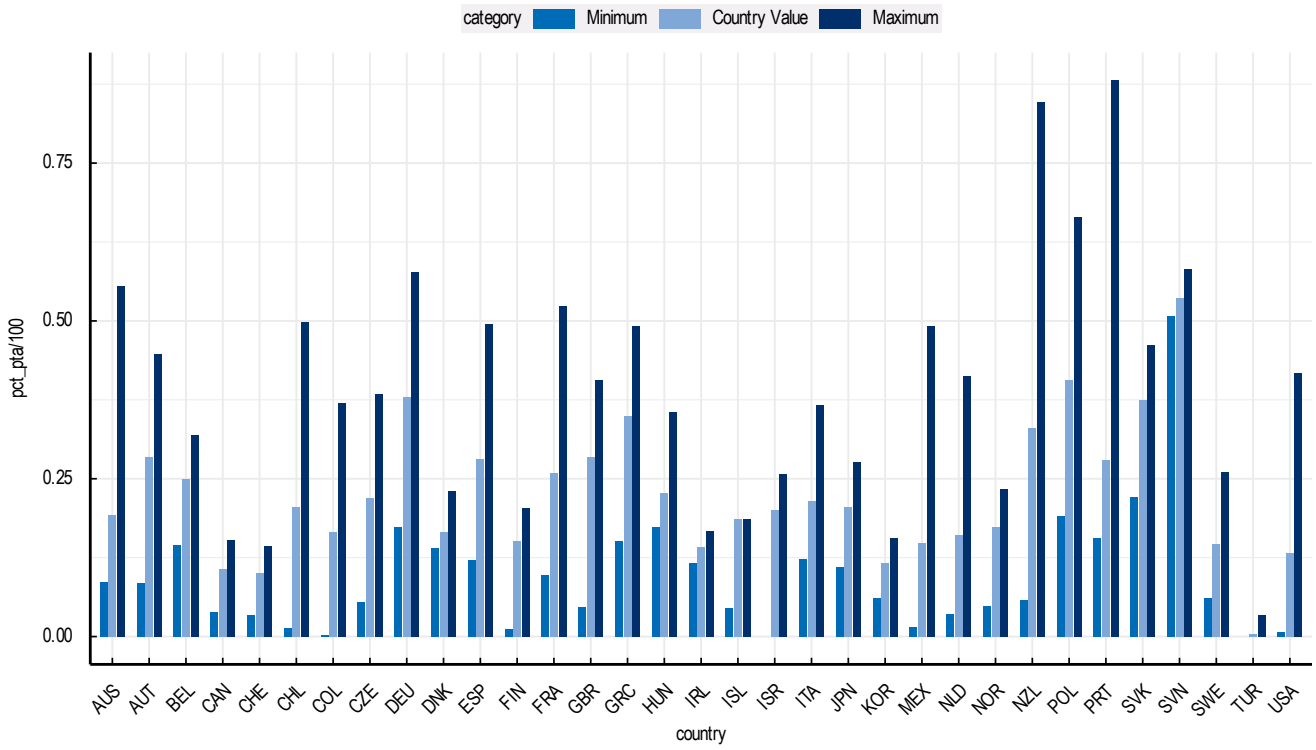
```
p <- p +
  theme_oecd()
p
```



1.3. Adjusting color palette

To change the colours, we use the OECD scales. Note that you can reference the specific colours you'd like to use with specific HEX codes. You can also reference colours by name, with the full list of colours recognised by R here. Here we are using some arguments inside the scale function, in order to show some of the personalization alternatives.

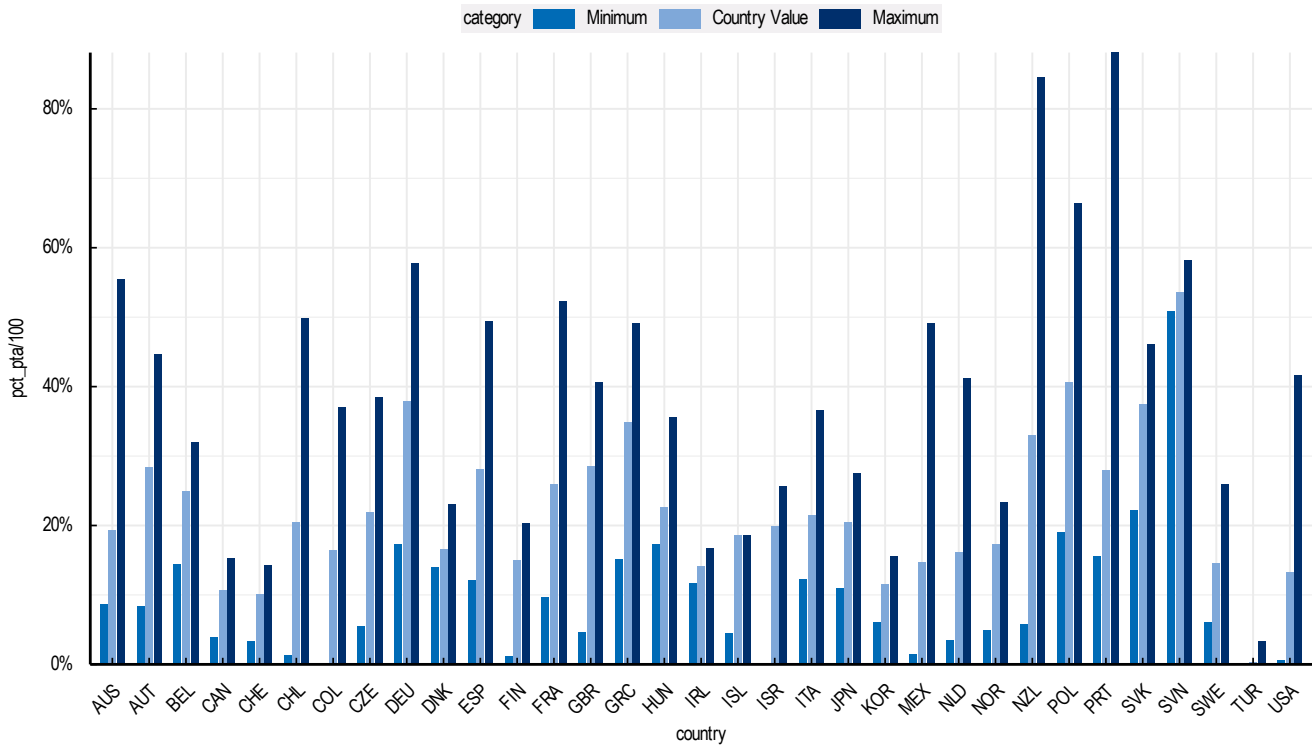
```
p <- p +
  scale_fill_oecd_d(option = "darkblue", direction = -1)
p
```



1.4. Adjusting axis scale

To change the scale to percentage, we can use the `percent()` function from the `scales` package, which comes with `ggplot2`.

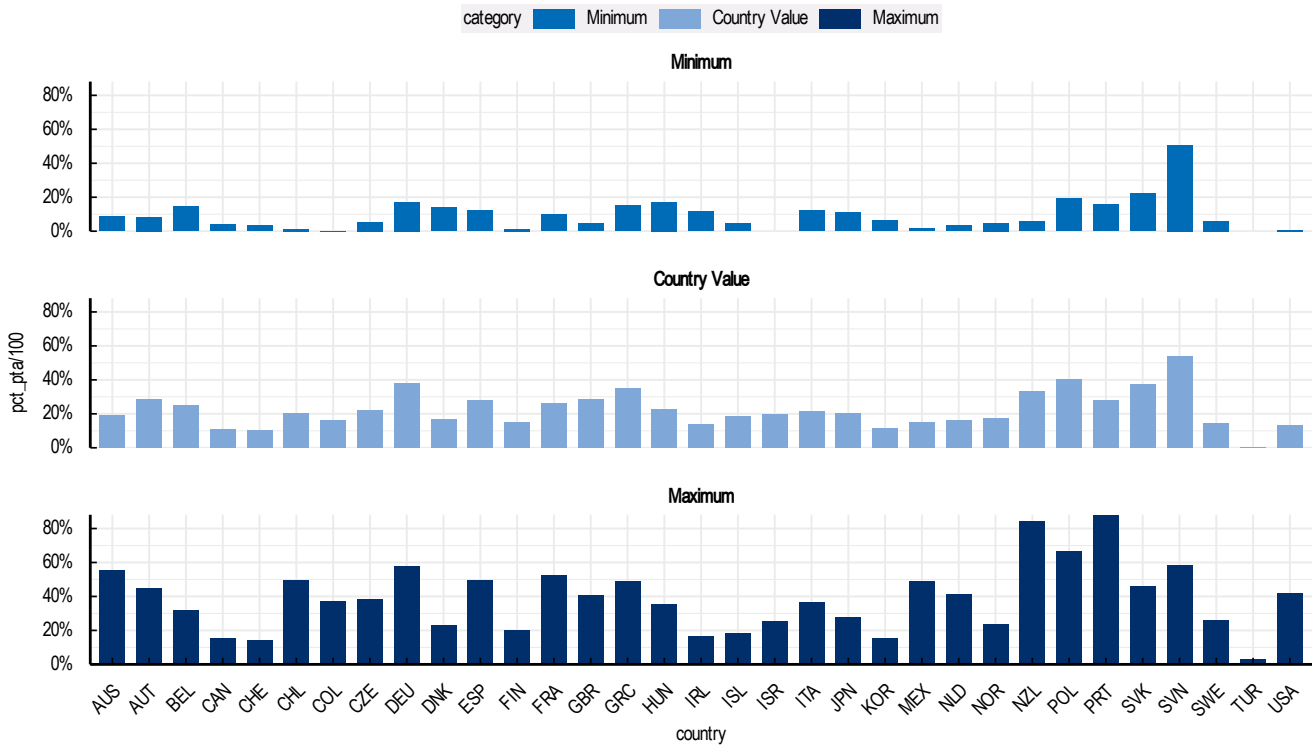
```
p <- p +
  scale_y_continuous(labels = scales::percent, expand = c(0,0))
p
```



1.5. Facetting

In order to divide the resulting plot into a 3-in-1 plot, we use the `facet_wrap()` function. The `ncol` argument is used to arrange the resulting plots in one column, instead of creating a three columns layout in this case.

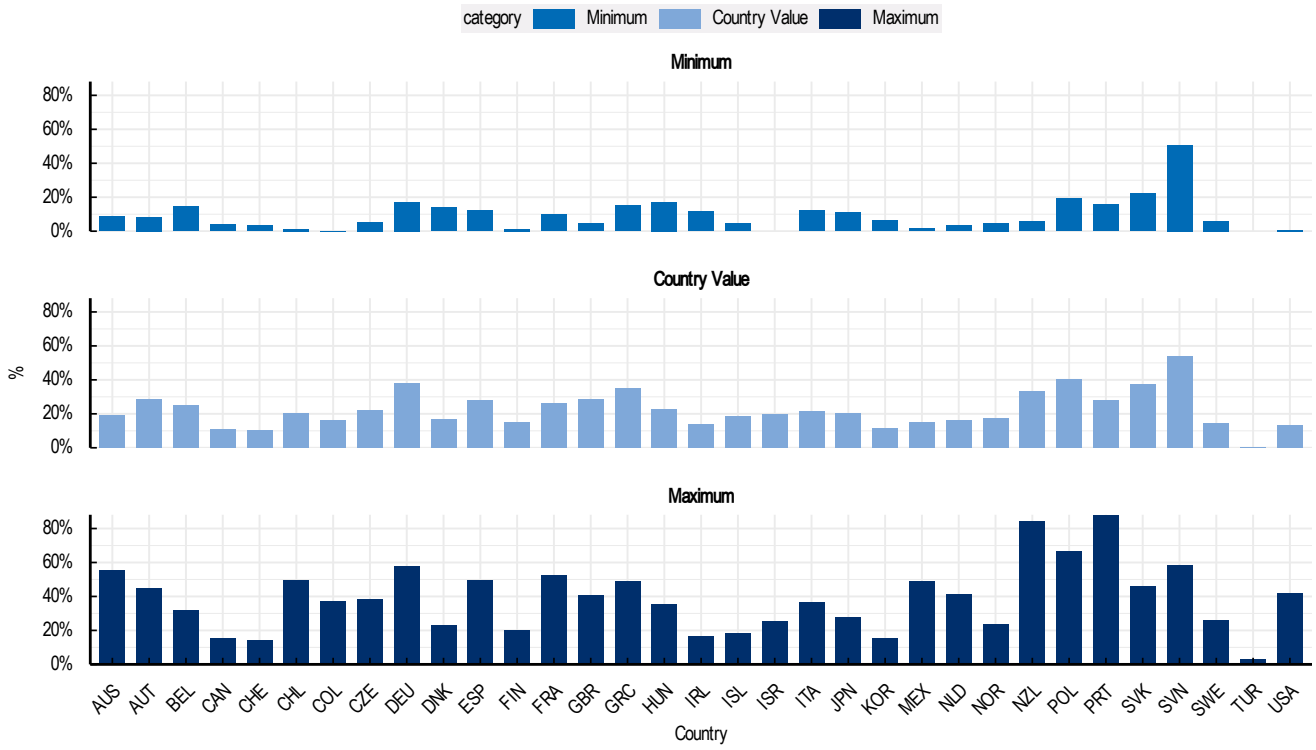
```
p <- p +
  facet_wrap(~category, ncol = 1)
p
```



1.6. Adding labels

To obtain the plot from the start of this section, we use the `labs()` function.

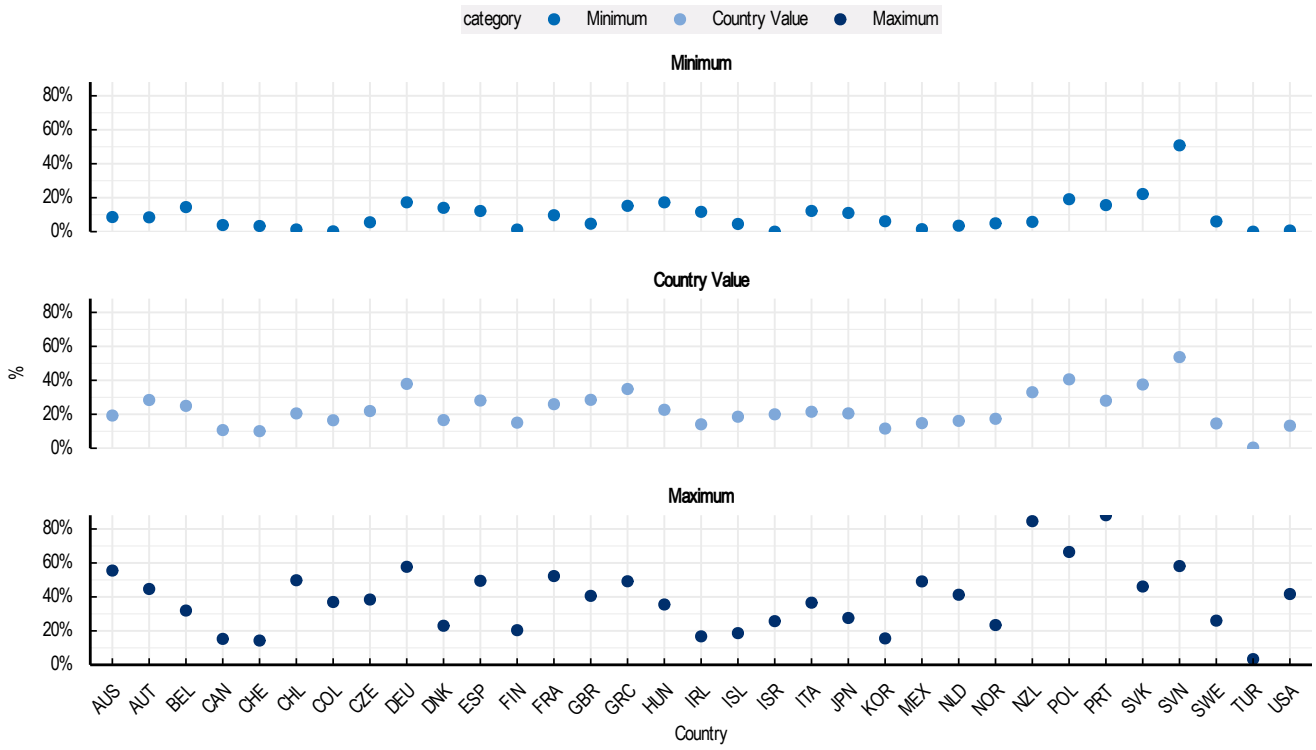
```
p <- p +
  labs(
    x = "Country",
    y = "%"
  )
p
```


category
Minimum
Country Value
Maximum
Minimum
Country Value
Maximum
%
80%
60%
40%
20%
0%
AUS AUT BEL CAN CHE CHL COL CZE DEU DNK ESP FIN FRA GBR GRC HUN IRL ISL ISR ITA JPN KOR MEX NLD NOR NZL POL PRT SVK SVN SWE TUR USA
Country

CHAPTER 2

Scatter plots

We will work towards creating the area plot below. We will take you from a basic bar plot and explain all the customisations we add to the code step-by-step.



2.1. Basic graph

You can use fonts such as Arial Narrow within ggplot2. This package allows that with a dedicated function. The first thing to do is load in the libraries and data, as below:

```
library(oecdplot)
library(ggplot2)

load_oecd_fonts()
```

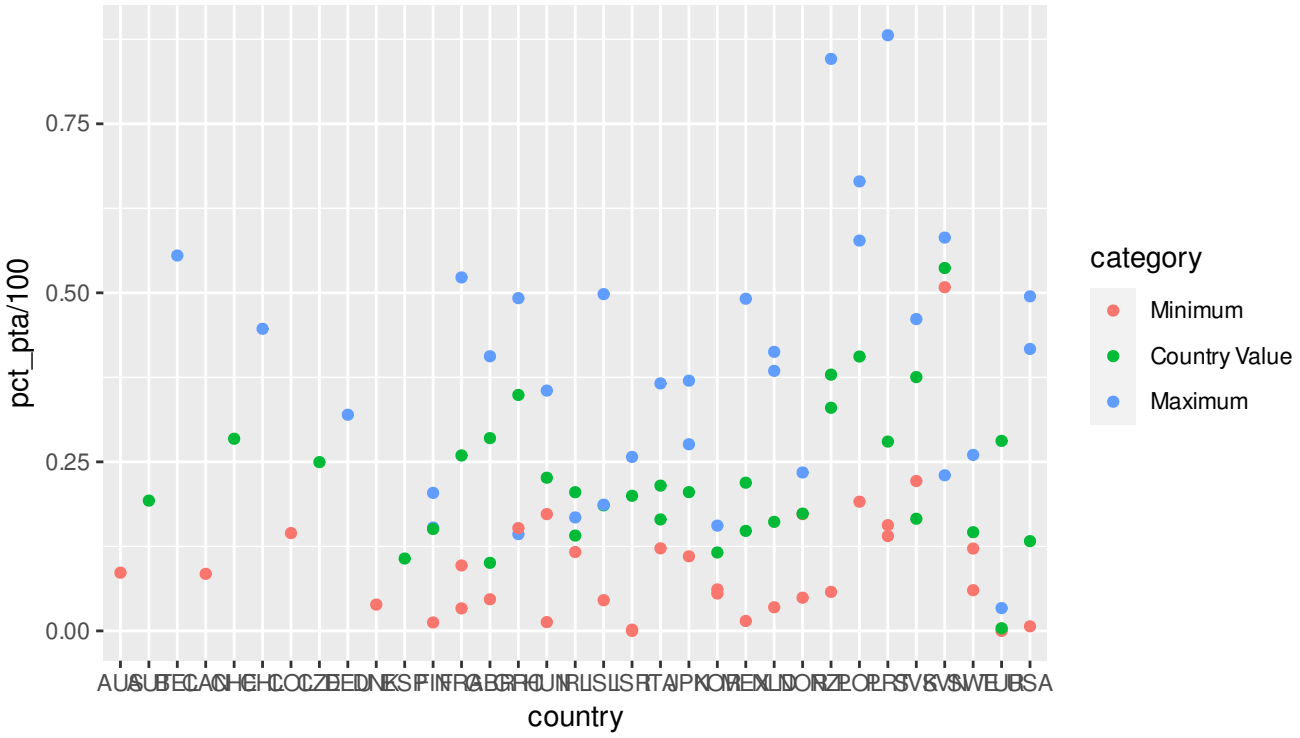
We will be working with the `pta` dataset, which is included in the package.

```
pta
```

```
# A tibble: 99 x 4
   country region        category      pct_pta
   <fct>   <fct>         <fct>           <dbl>
 1 NZL     Auckland      Minimum          5.77
 2 NZL     Country Value Country Value   33.0
 3 NZL     West Coast    Maximum         84.6
 4 PRT     Centro        Minimum         15.6
 5 PRT     Country Value Country Value   28.0
 6 PRT     Algarve       Maximum         88.1
 7 CHL     Coquimbo      Minimum          1.3
 8 CHL     Country Value Country Value   20.5
 9 CHL     Aysén         Maximum         49.8
10 MEX     Guerrero      Minimum          1.47
# ... with 89 more rows
```

To initialise a plot we tell ggplot that `pta` is our data, and specify the variables on each axis. We then instruct ggplot to render this as an bar plot by adding the `geom_col()` function. The `position` argument makes the categories appear side-by-side, instead of stacking them on top of each other.

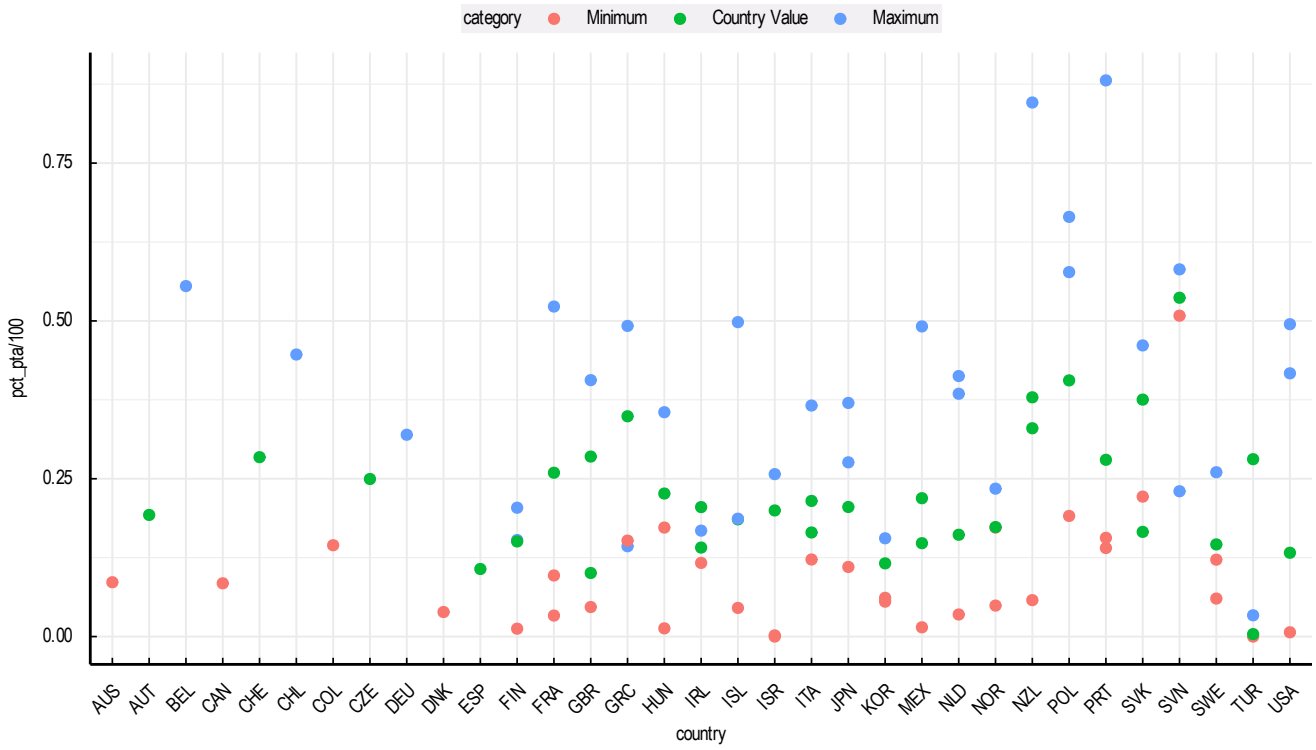
```
p <- ggplot(data = pta, aes(x = country, y = pct_pta / 100, colour = category)) +
  geom_point(position = "dodge2")
p
```



2.2. Adjusting theme

We can change the overall look of the graph using themes. We'll start using a simple theme customisation by adding `theme_oecd()` after `ggplot()`.

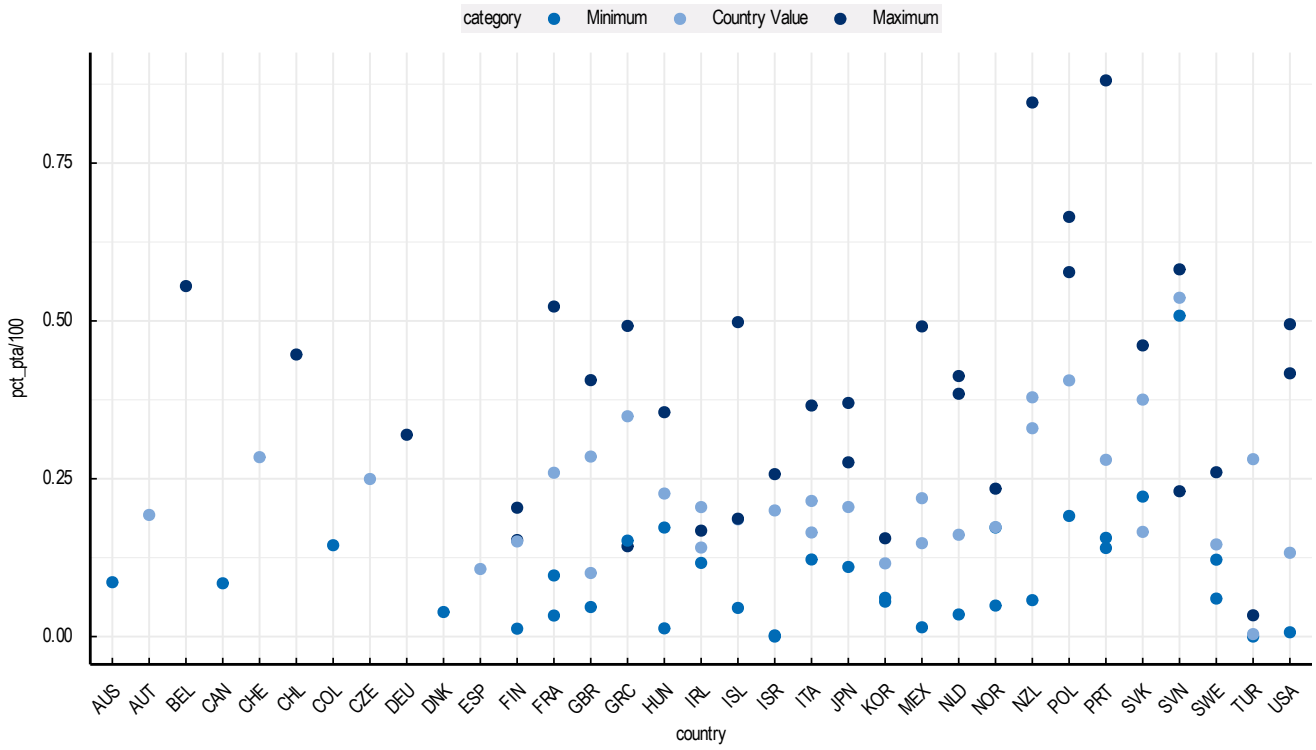
```
p <- p +
  theme_oecd()
p
```



2.3. Adjusting color palette

To change the colours, we use the OECD scales. Note that you can reference the specific colours you'd like to use with specific HEX codes. You can also reference colours by name, with the full list of colours recognised by R here. Here we are using some arguments inside the scale function, in order to show some of the personalization alternatives.

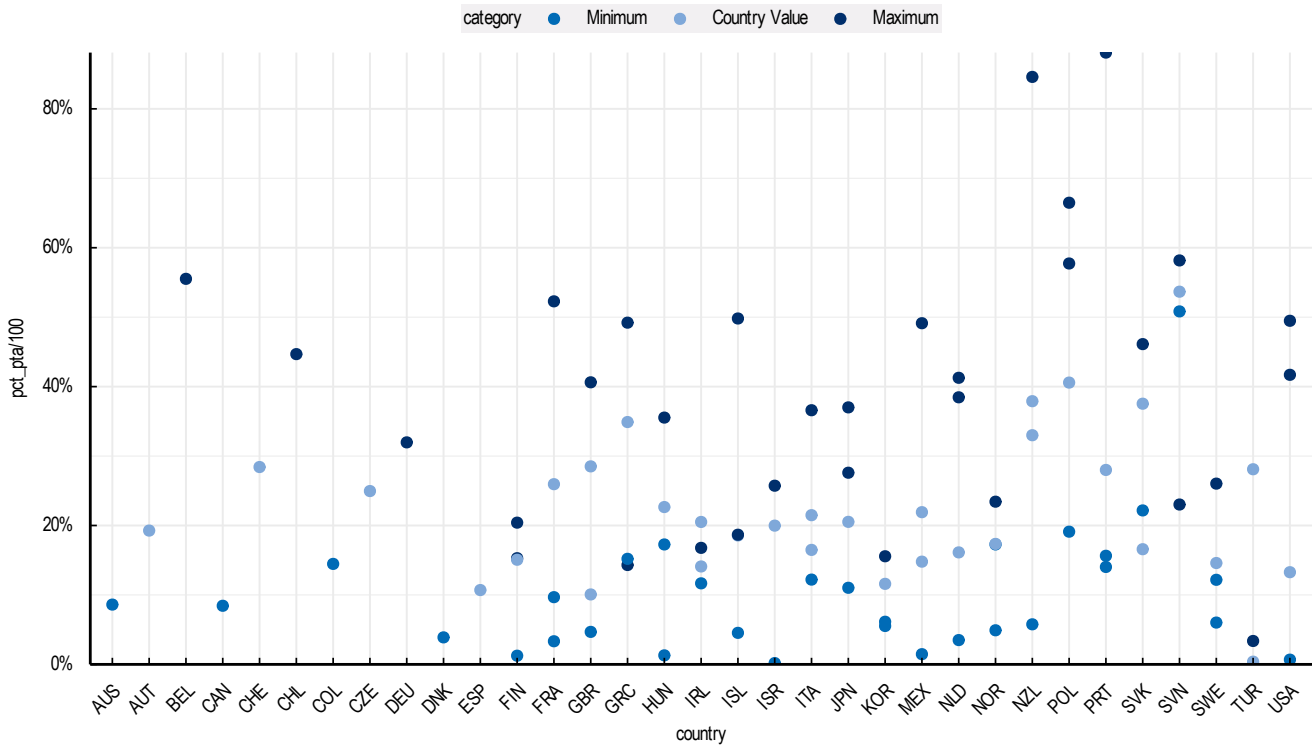
```
p <- p +
  scale_colour_oecd_d(option = "darkblue", direction = -1)
p
```



2.4. Adjusting axis scale

To change the scale to percentage, we can use the `percent()` function from the `scales` package, which comes with `ggplot2`.

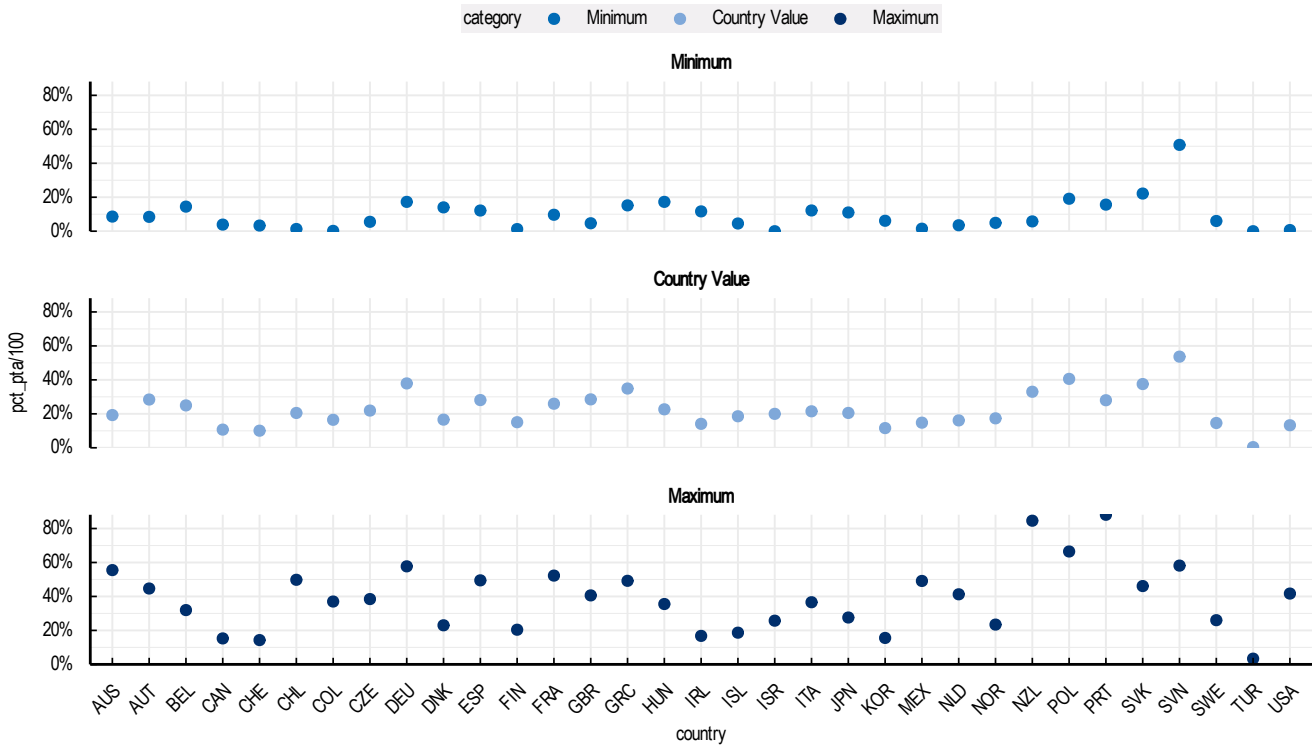
```
p <- p +
  scale_y_continuous(labels = scales::percent, expand = c(0,0))
p
```



2.5. Facetting

In order to divide the resulting plot into a 3-in-1 plot, we use the `facet_wrap()` function. The `ncol` argument is used to arrange the resulting plots in one column, instead of creating a three columns layout in this case.

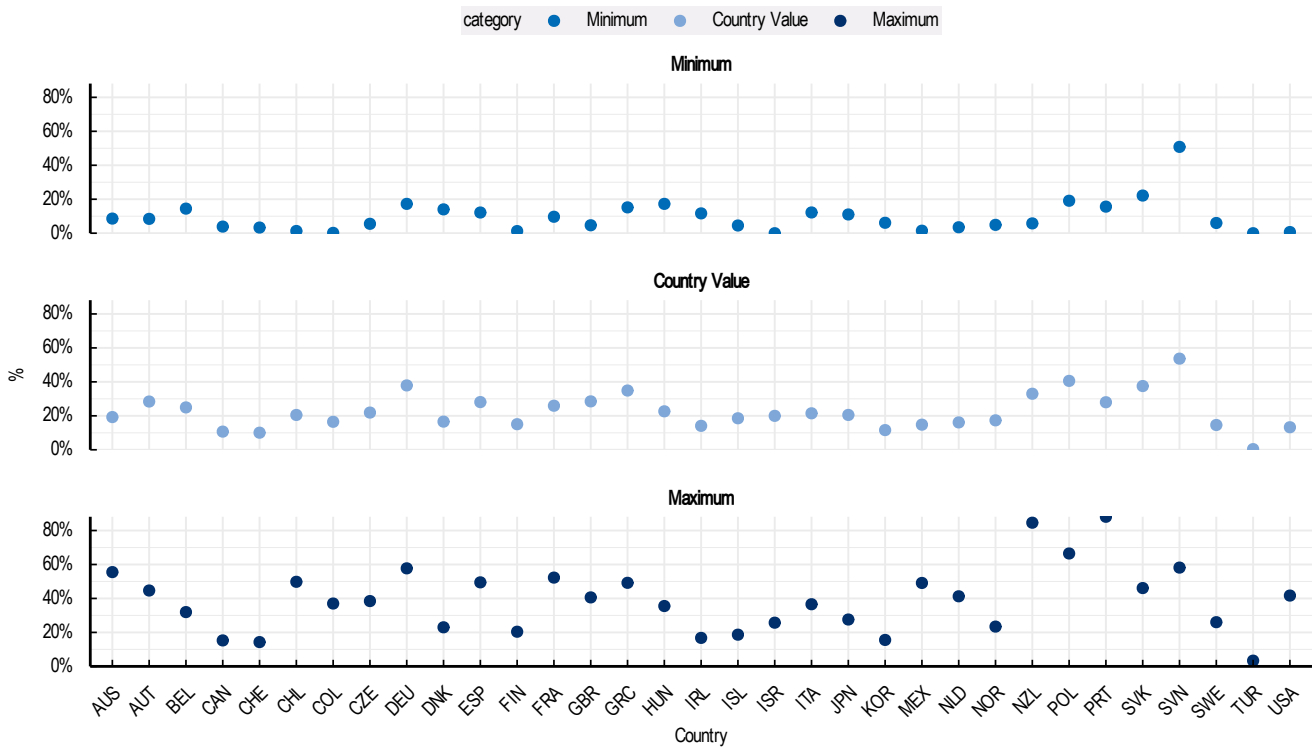
```
p <- p +
  facet_wrap(~category, ncol = 1)
p
```



2.6. Adding labels

To obtain the plot from the start of this section, we use the `labs()` function.

```
p <- p +
  labs(
    x = "Country",
    y = "%"
  )
p
```

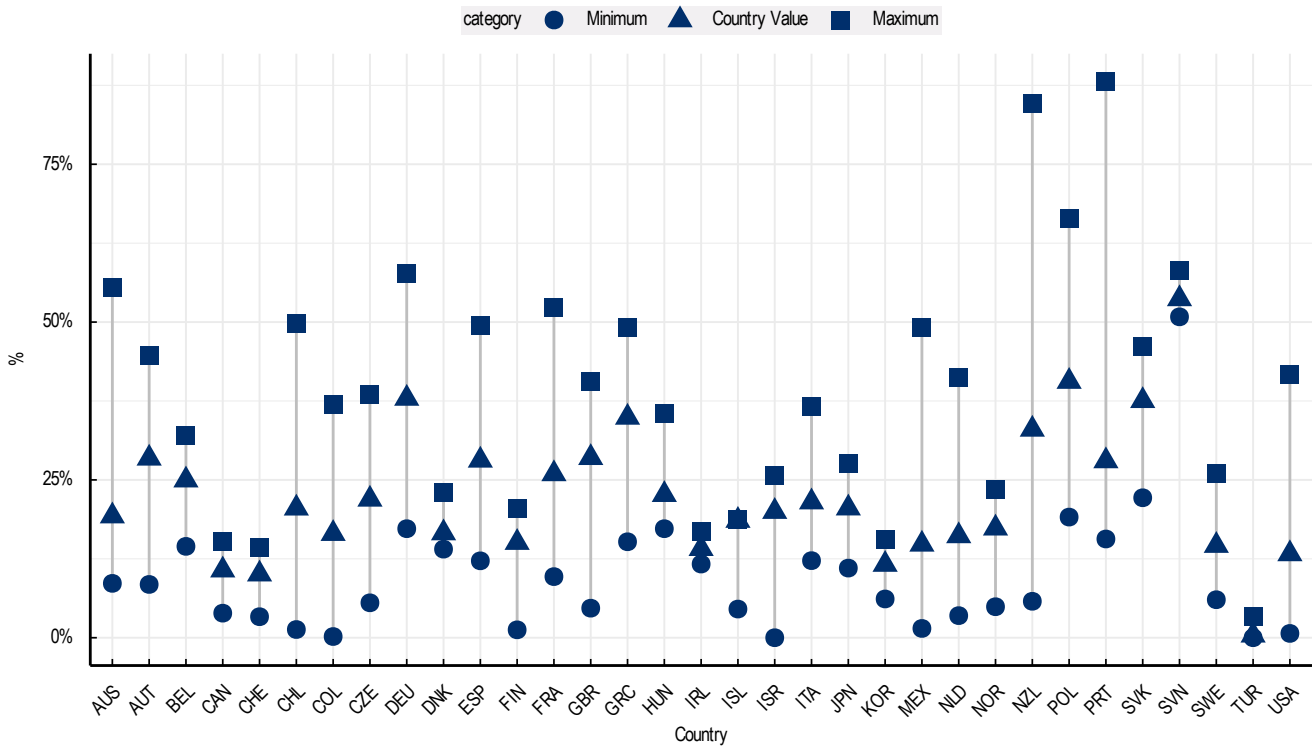

category
Minimum
Country Value
Maximum
Minimum
Country Value
Maximum
%
80%
60%
40%
20%
0%
AUS AUT BEL CAN CHE CHL COL CZE DEU DNK ESP FIN FRA GBR GRC HUN IRL ISL ISR ITA JPN KOR MEX NLD NOR NZL POL PRT SVK SVN SWE TUR USA
Country

CHAPTER 3

Min-max plots

We will work towards creating the area plot below. We will take you from a basic bar plot and explain all the customisations we add to the code step-by-step.

This plot is build by adding *layers* of different geoms, using the `geom_line()` and `geom_point()` functions. The first one add the vertical grey line behind the dots, and the second adds the overlaying points. Defining the shape parameter here adds a visual aid to make it easier to differentiate the different values.



3.1. Basic graph

You can use fonts such as Arial Narrow within ggplot2. This package allows that with a dedicated function. The first thing to do is load in the libraries and data, as below:

```
library(oecdplot)
library(ggplot2)

load_oecd_fonts()
```

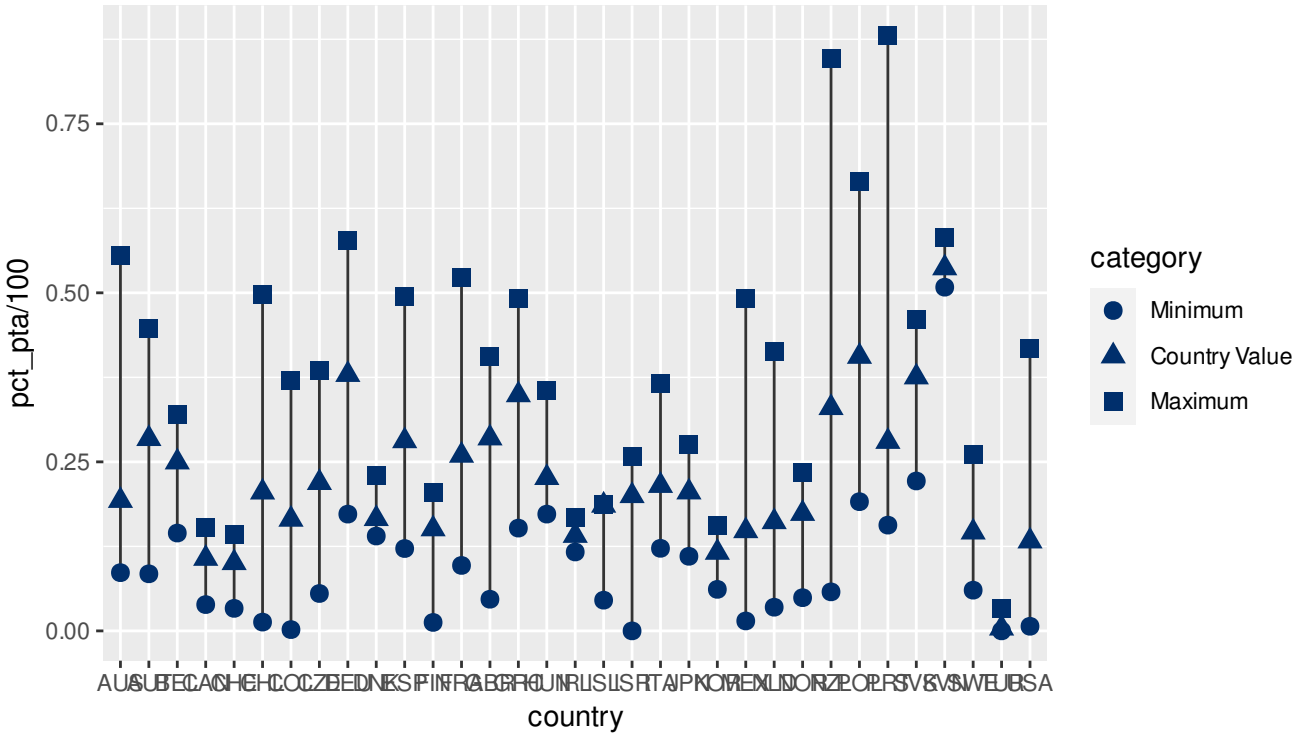
We will be working with the pta dataset, which is included in the package.

```
pta
```

```
# A tibble: 99 x 4
   country region         category      pct_pta
   <fct>   <fct>          <fct>           <dbl>
 1 NZL     Auckland       Minimum          5.77
 2 NZL     Country Value  Country Value   33.0
 3 NZL     West Coast     Maximum         84.6
 4 PRT     Centro         Minimum         15.6
 5 PRT     Country Value  Country Value   28.0
 6 PRT     Algarve        Maximum         88.1
 7 CHL     Coquimbo       Minimum          1.3
 8 CHL     Country Value  Country Value   20.5
 9 CHL     Aysén          Maximum         49.8
10 MEX     Guerrero       Minimum          1.47
# ... with 89 more rows
```

To initialise a plot we tell ggplot that pta is our data, and specify the variables on each axis. We then instruct ggplot to render this as an bar plot by adding the geom_line() and geom_point() functions. The colour argument is outside aes() in order to use the same colour for each figure.

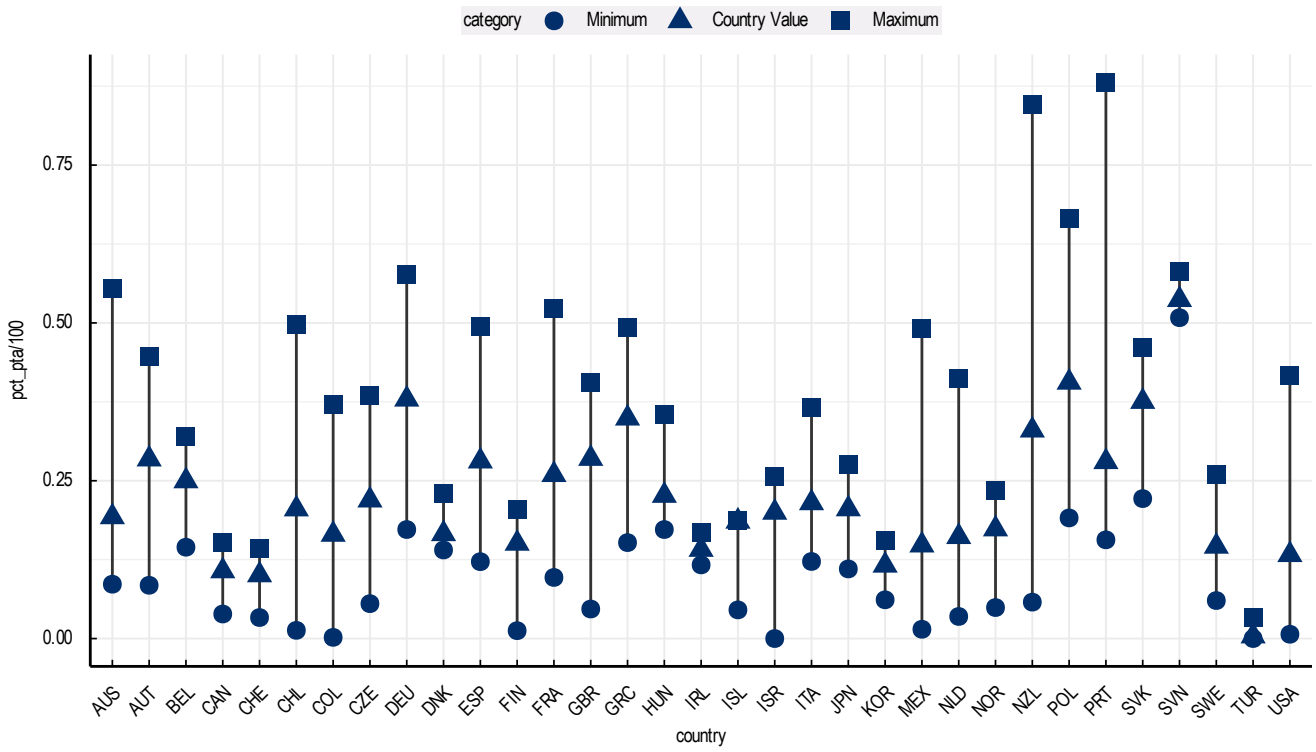
```
p <- ggplot(pta, aes(x = country, y = pct_pta / 100)) +
  geom_line(colour = "grey20") +
  geom_point(
    aes(shape = category),
    colour = oecd_clrs(n = 1, option = "darkblue"),
    size = 3
  )
p
```



3.2. Adjusting theme

We can change the overall look of the graph using themes. We'll start using a simple theme customisation by adding `theme_oecd()` after `ggplot()`.

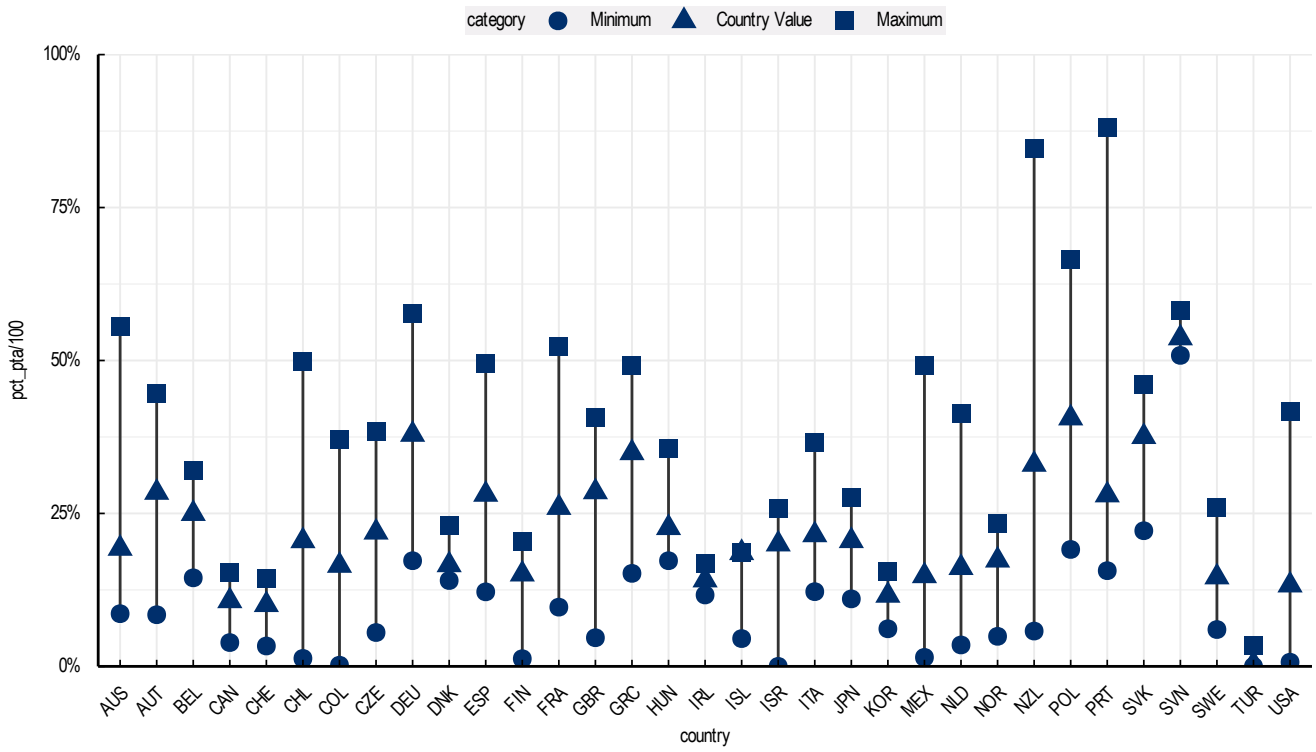
```
p <- p +
  theme_oecd()
p
```



3.3. Adjusting axis scale

To change the scale to percentage, we can use the `percent()` function from the `scales` package, which comes with ggplot2.

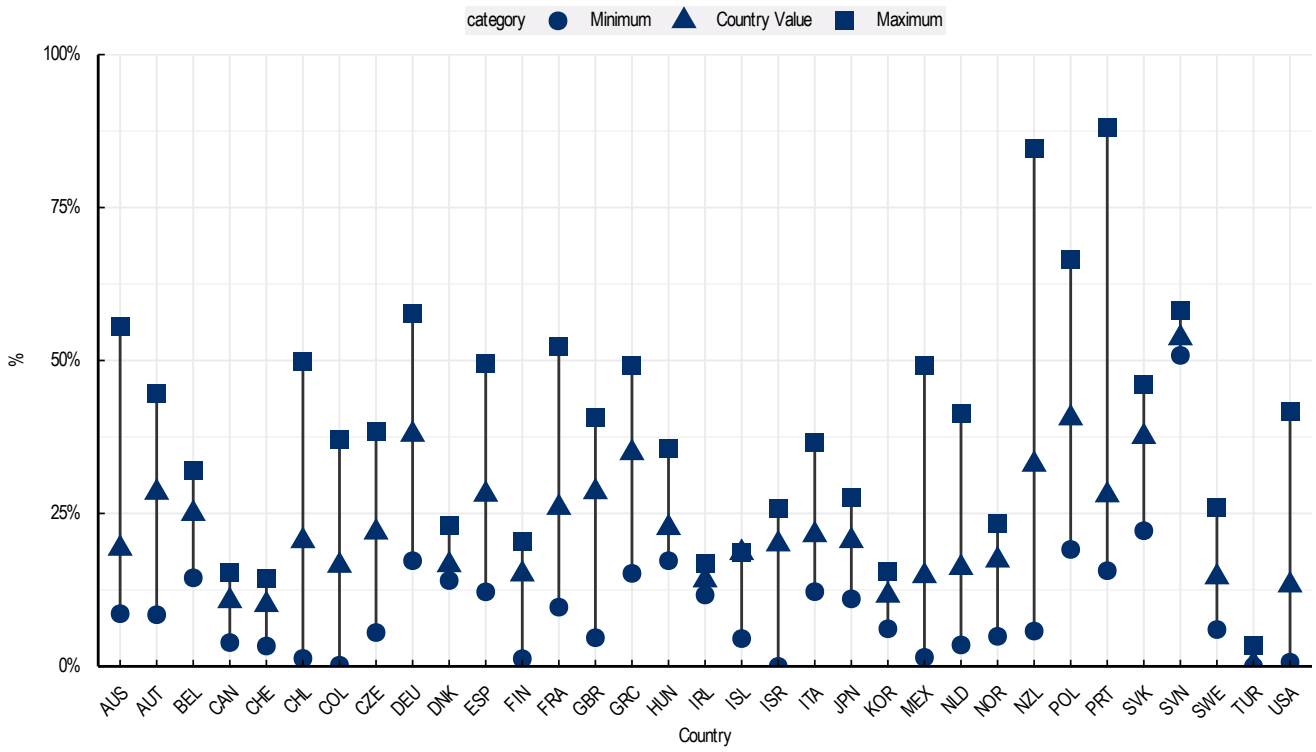
```
p <- p +
  scale_y_continuous(
    labels = scales::percent,
    expand = c(0,0),
    limits = c(0,1)
  )
p
```



3.4. Adding labels

To obtain the plot from the start of this section, we use the `labs()` function.

```
p <- p +
  labs(
    x = "Country",
    y = "%"
  )
p
```


category
Minimum
Country Value
Maximum
100%
75%
50%
25%
0%
%
AUS
AUT
BEL
CAN
CHE
CHL
COL
CZE
DEU
DNK
ESP
FIN
FRA
GBR
GRC
HUN
IRL
ISL
ISR
ITA
JPN
KOR
MEX
NLD
NOR
NZL
POL
PRT
SVK
SVN
SWE
TUR
USA
Country

CHAPTER 4

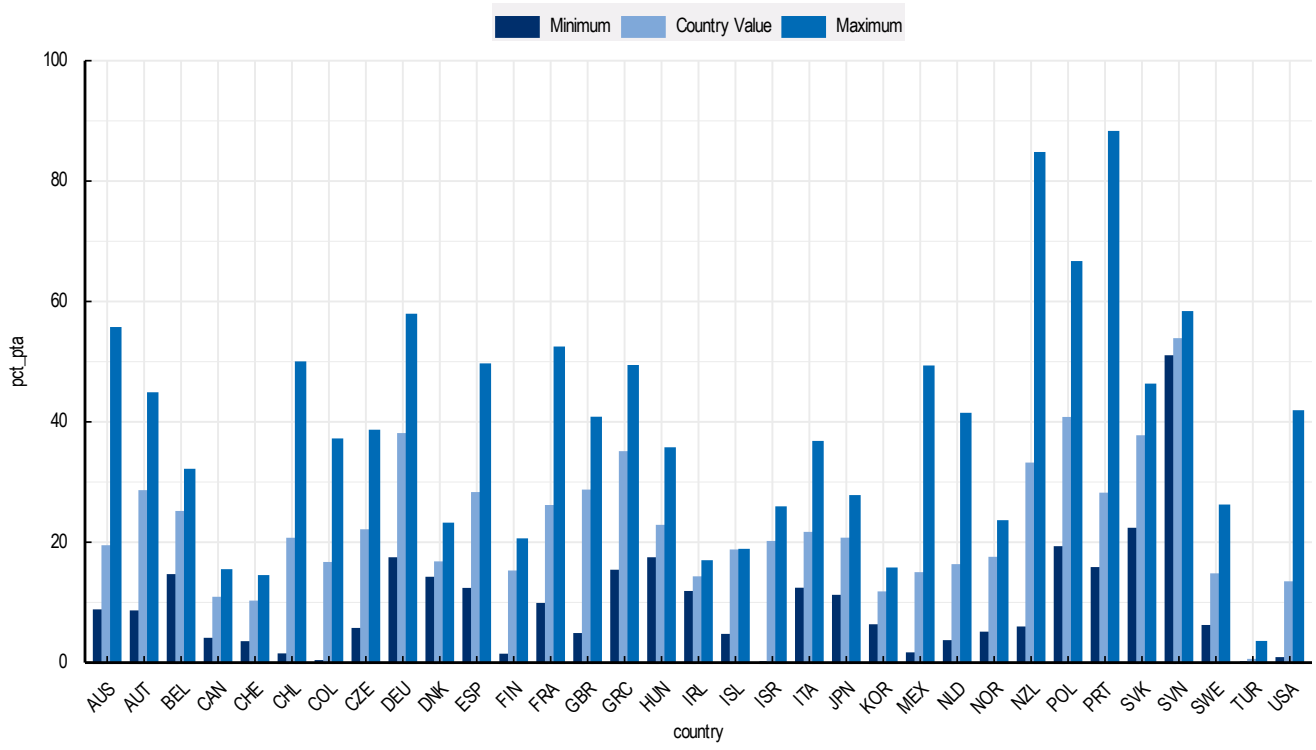
Shortcuts

We will work towards creating different plots by using shortcut functions within this package.

4.1. Column plot

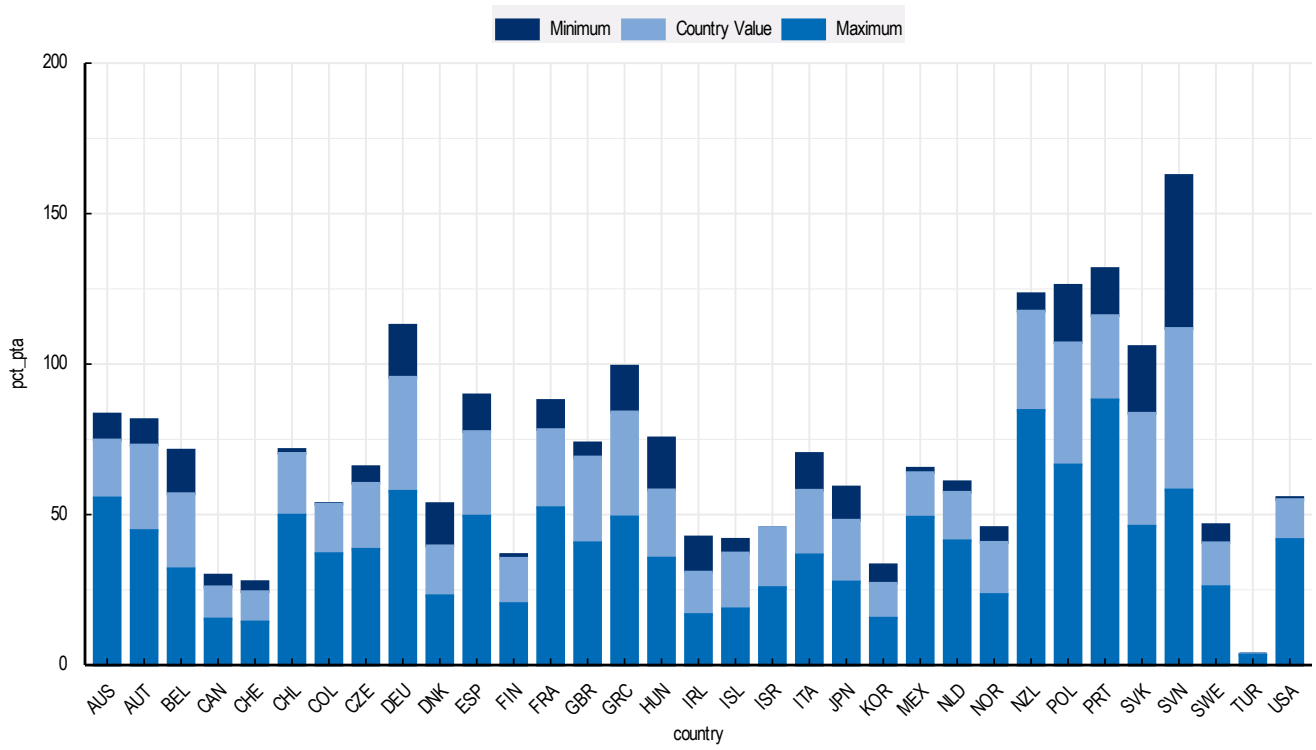
Simple column plot:

```
oecd_col(data = pta, x = country, y = pct_pta,  colour = category)
```



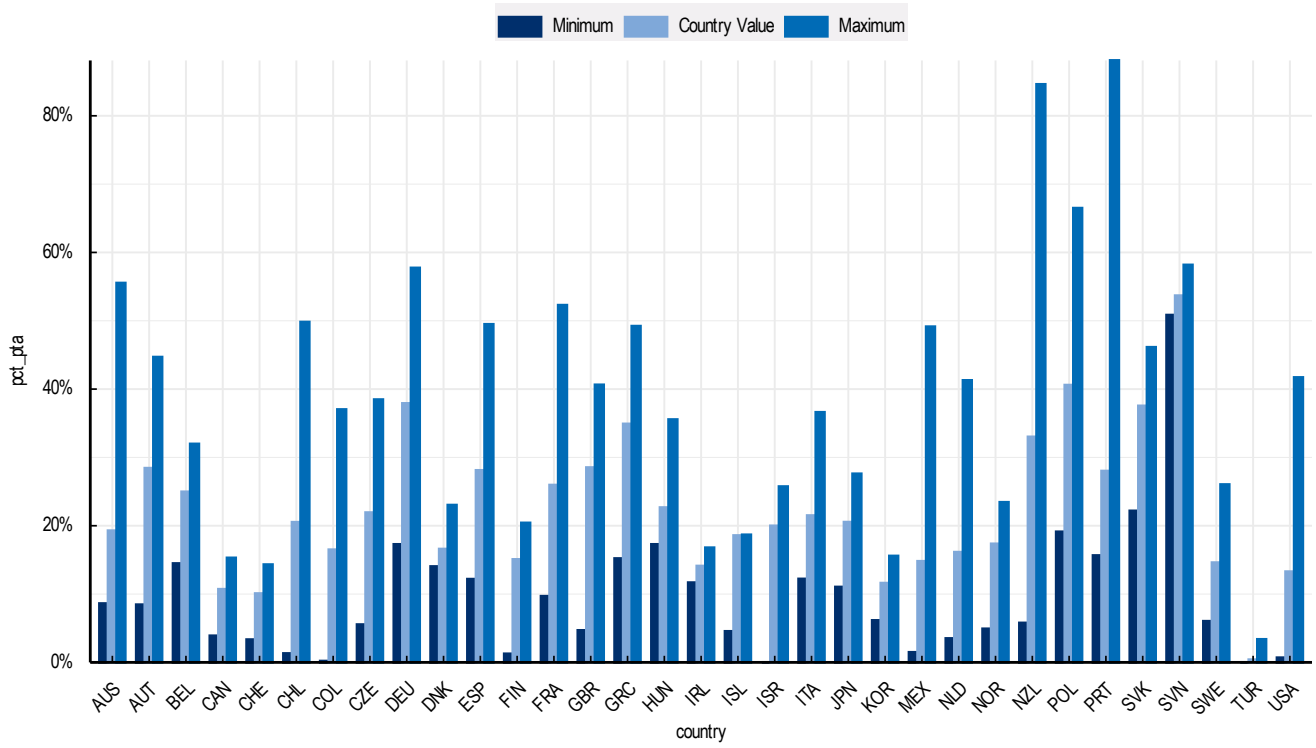
Stacked column plot:

```
oecd_col(data = pta, x = country, y = pct_pta, colour = category, stacked = T)
```



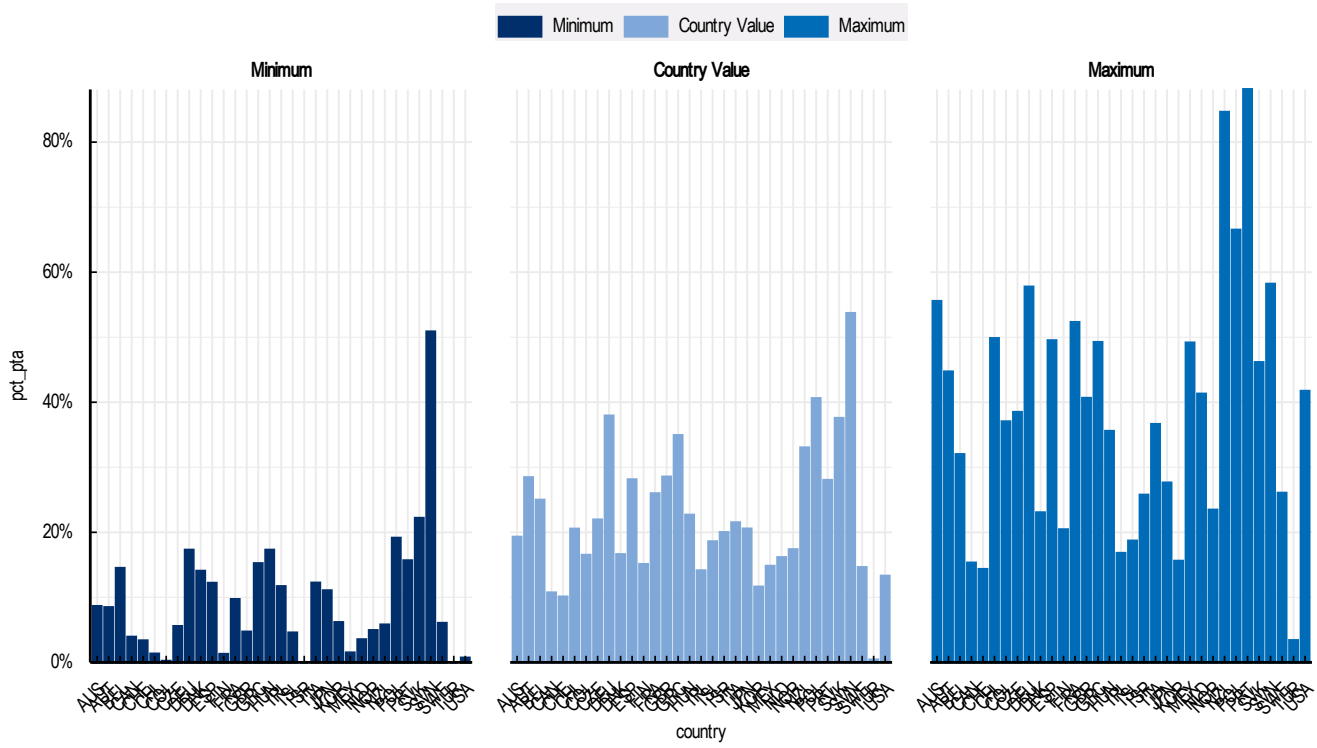
Stacked column plot with y-axis as percentage:

```
pta2 <- pta
pta2$pct_pta <- pta2$pct_pta / 100
oecd_col(data = pta2, x = country, y = pct_pta, colour = category) +
  scale_y_continuous(labels = scales::percent, expand = c(0,0))
```



Faceted column plot with y-axis as percentage:

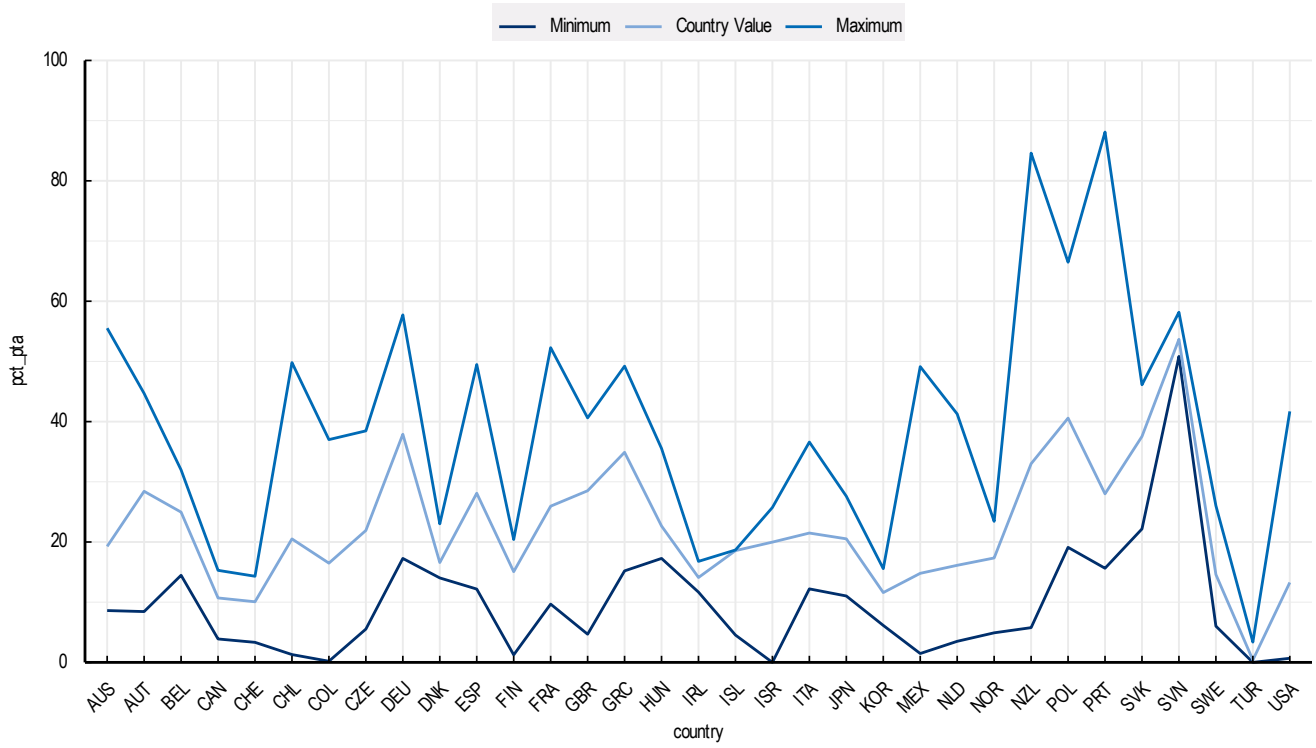
```
pta2 <- pta
pta2$pct_pta <- pta2$pct_pta / 100

oecd_col(data = pta2, x = country, y = pct_pta, colour = category,
         facet = category) +
  scale_y_continuous(labels = scales::percent, expand = c(0,0))
```



4.2. Line plot

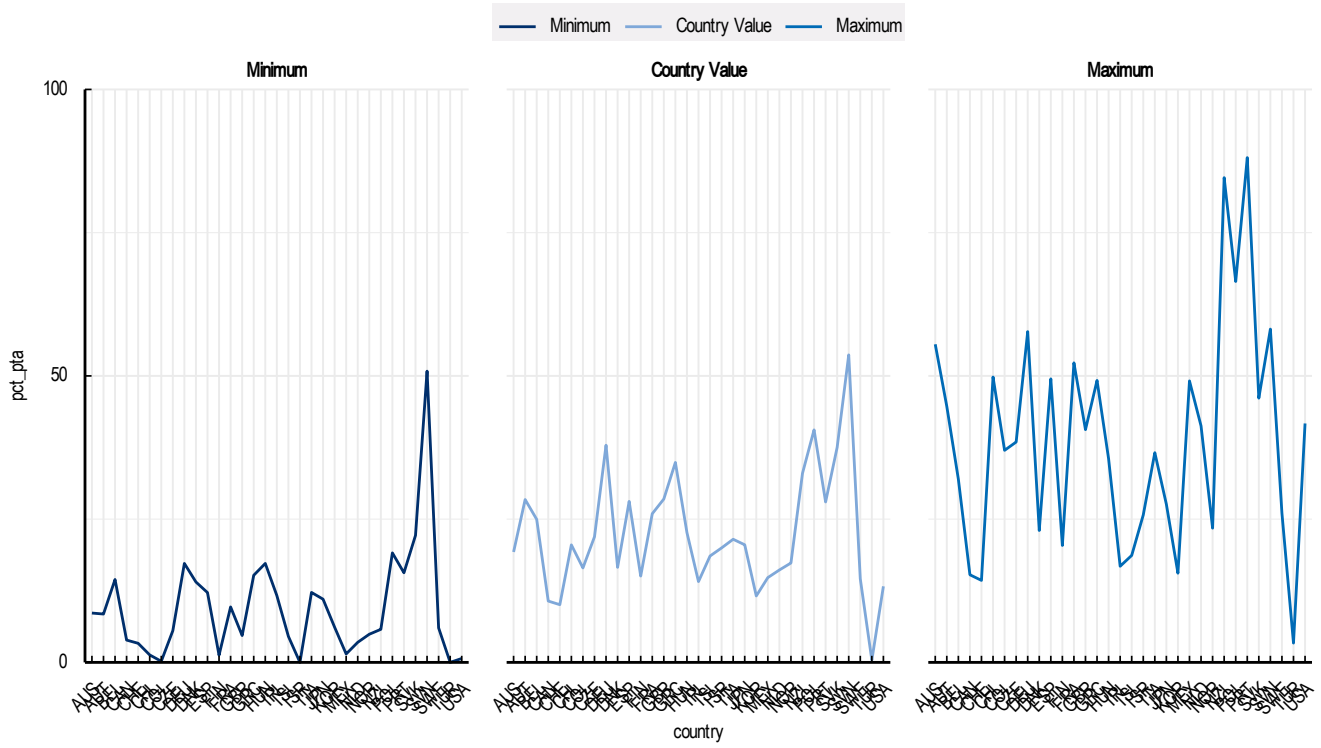
Simple column plot:

```
oecd_line(pta, x = country, y = pct_pta, colour = category, group = category)
```



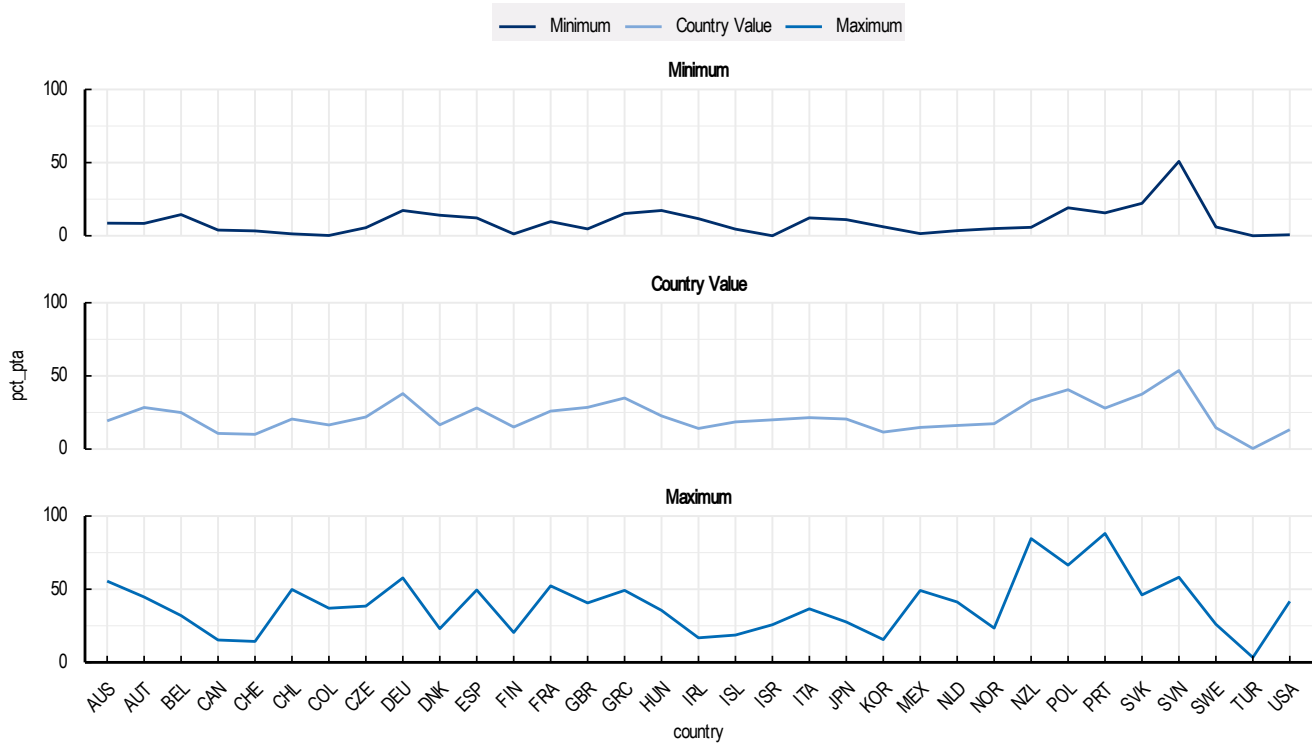
Faceted line plot:

```
oecd_line(pta,
  x = country, y = pct_pta, colour = category, group = category,
  facet = category
)
```



Faceted line plot with custom ordering:

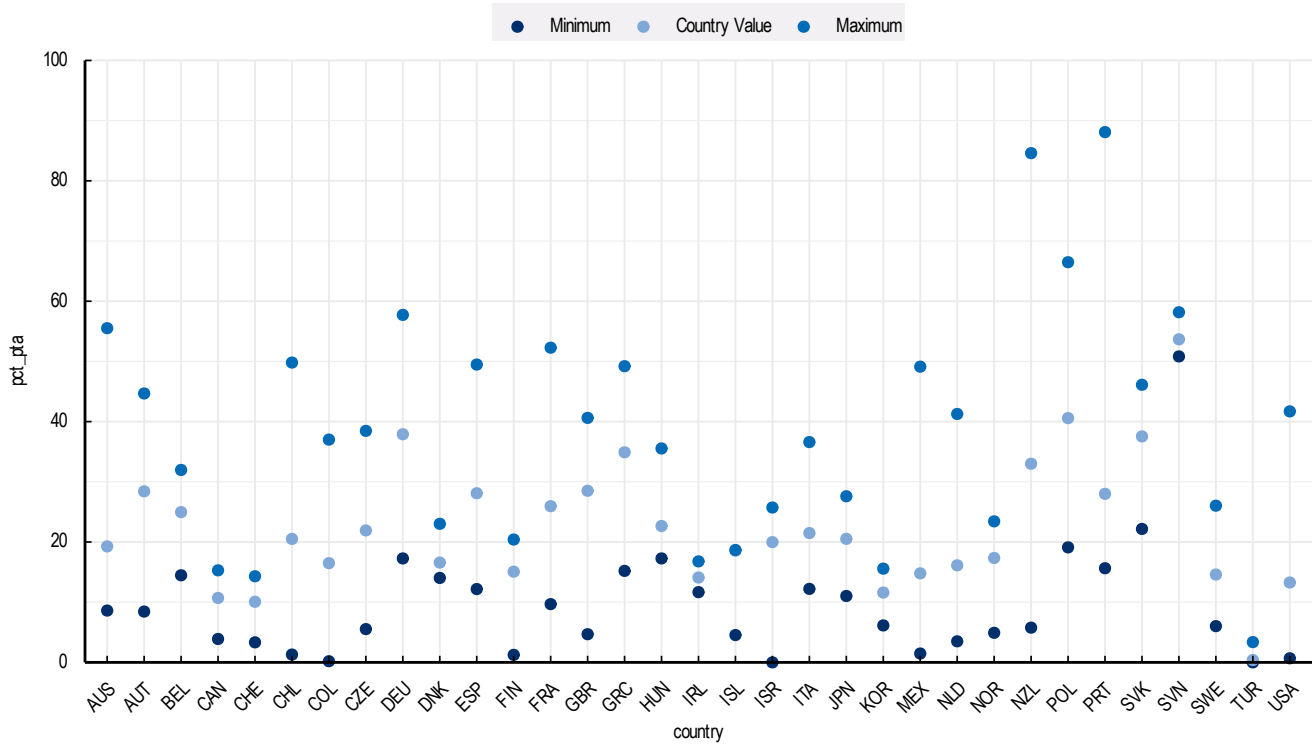
```
oecd_line(pta,
  x = country, y = pct_pta, colour = category, group = category,
  facet = category, facet_ncol = 1
)
```



4.3. Scatterplot

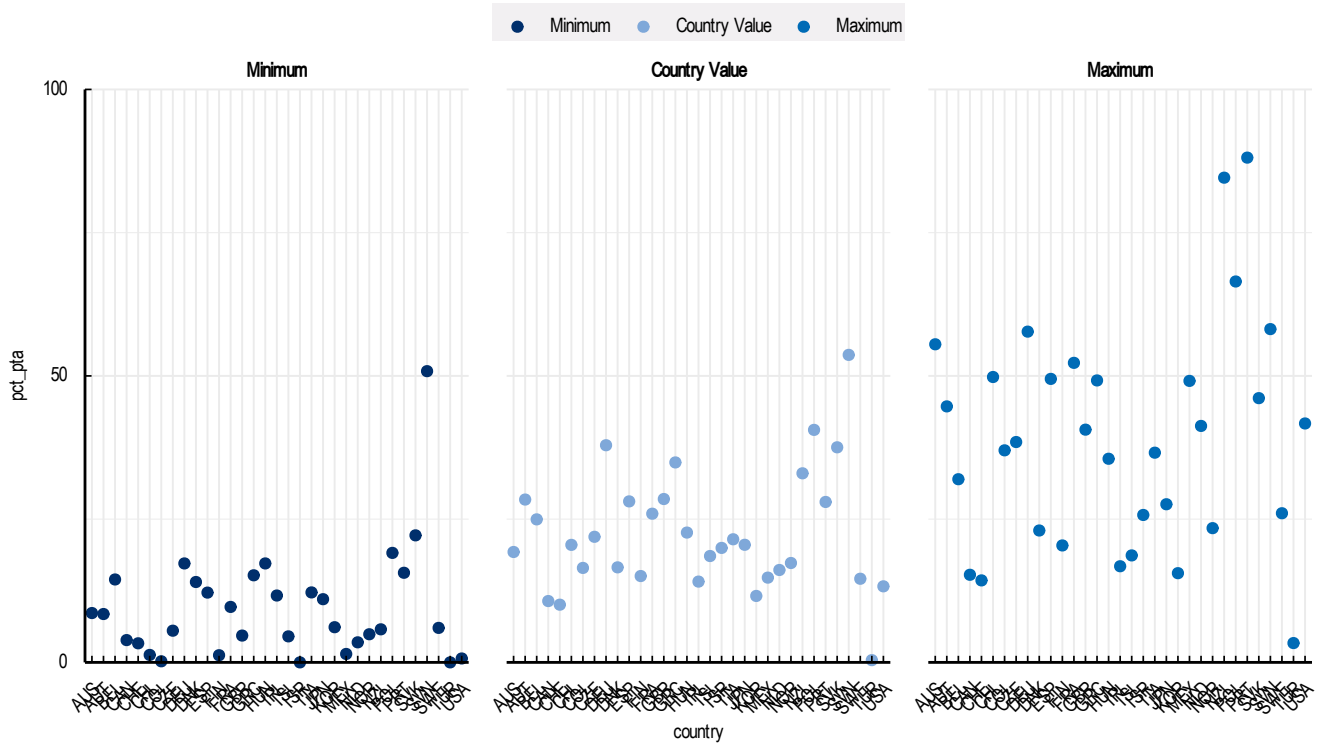
Simple scatterplot:

```
oecd_point(pta, x = country, y = pct_pta, colour = category, group = category)
```



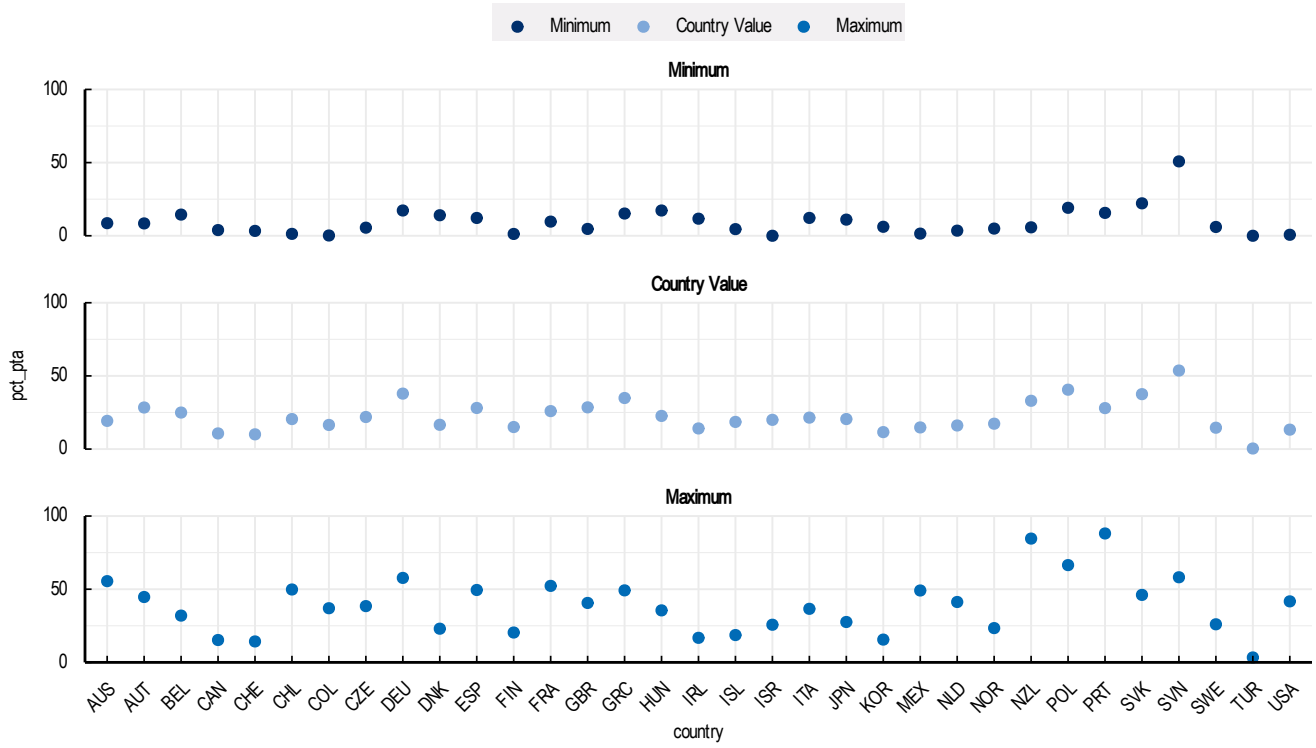
Faceted scatterplot:

```
oecd_point(pta,
  x = country, y = pct_pta, colour = category, group = category,
  facet = category
)
```



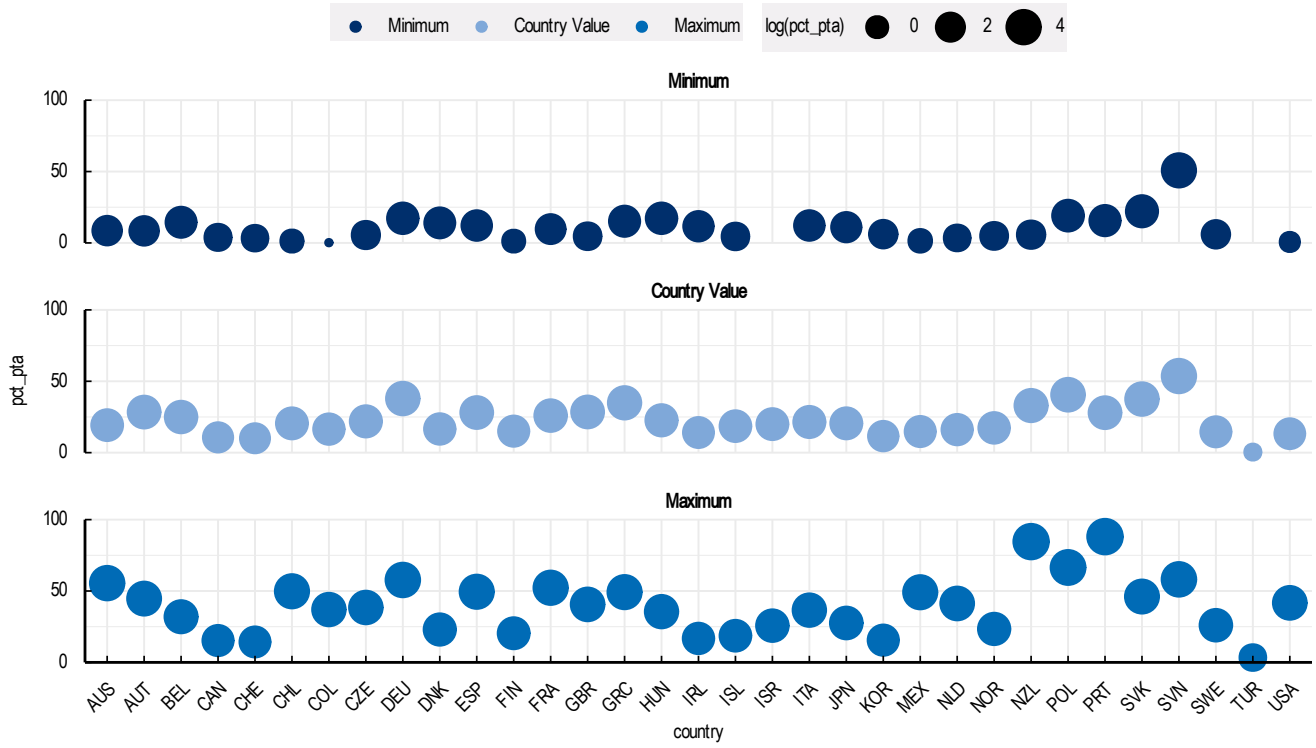
Faceted line plot with custom ordering:

```
oecd_point(pta,
  x = country, y = pct_pta, colour = category, group = category,
  facet = category, facet_ncol = 1
)
```



Options specific to scatterplots:

```
oecd_point(pta,
  x = country, y = pct_pta, colour = category, group = category,
  facet = category, facet_ncol = 1, size = log(pct_pta)
)
```

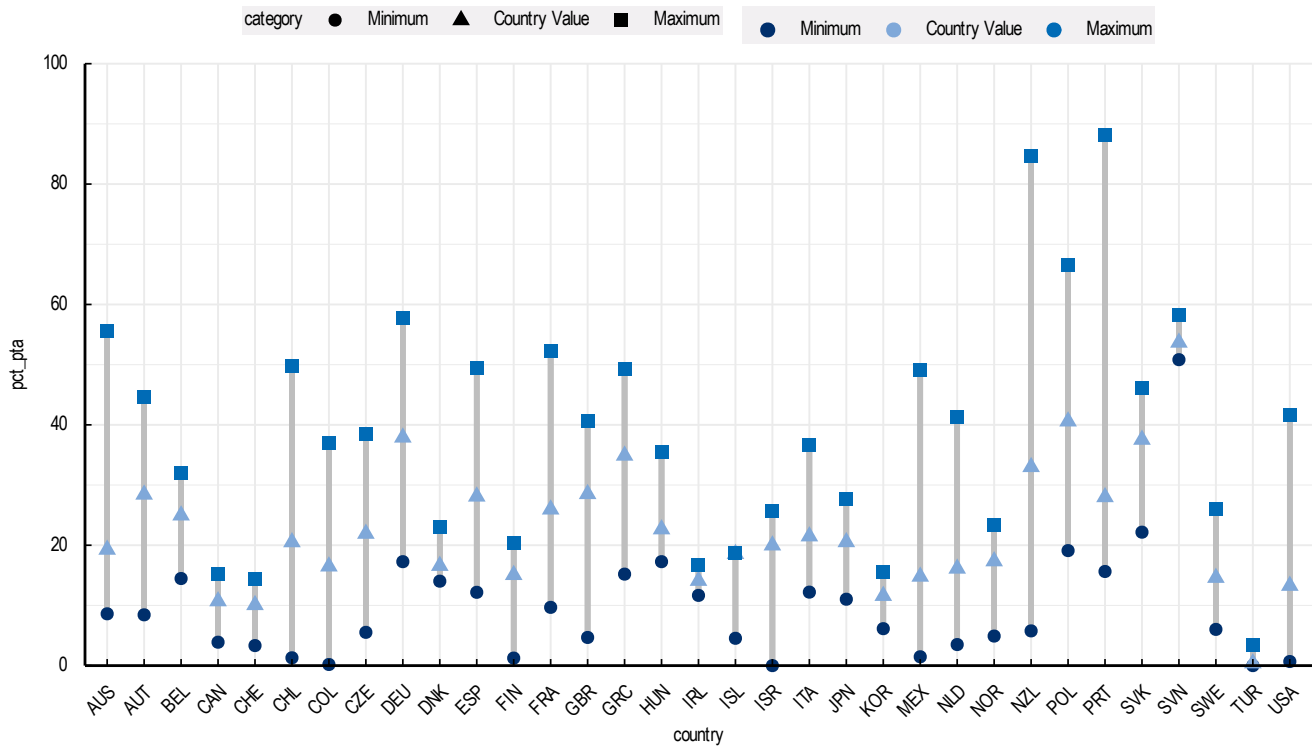

Minimum
Country Value
Maximum
log(pct_pta)
0
2
4
Minimum
Country Value
Maximum
100
50
0
pct_pta
AUS AUT BEL CAN CHE CHL COL CZE DEU DNK ESP FIN FRA GBR GRC HUN IRL ISL ISR ITA JPN KOR MEX NLD NOR NZL POL PRT SVK SVN SWE TUR USA
country

CHAPTER 5

Max-min plot

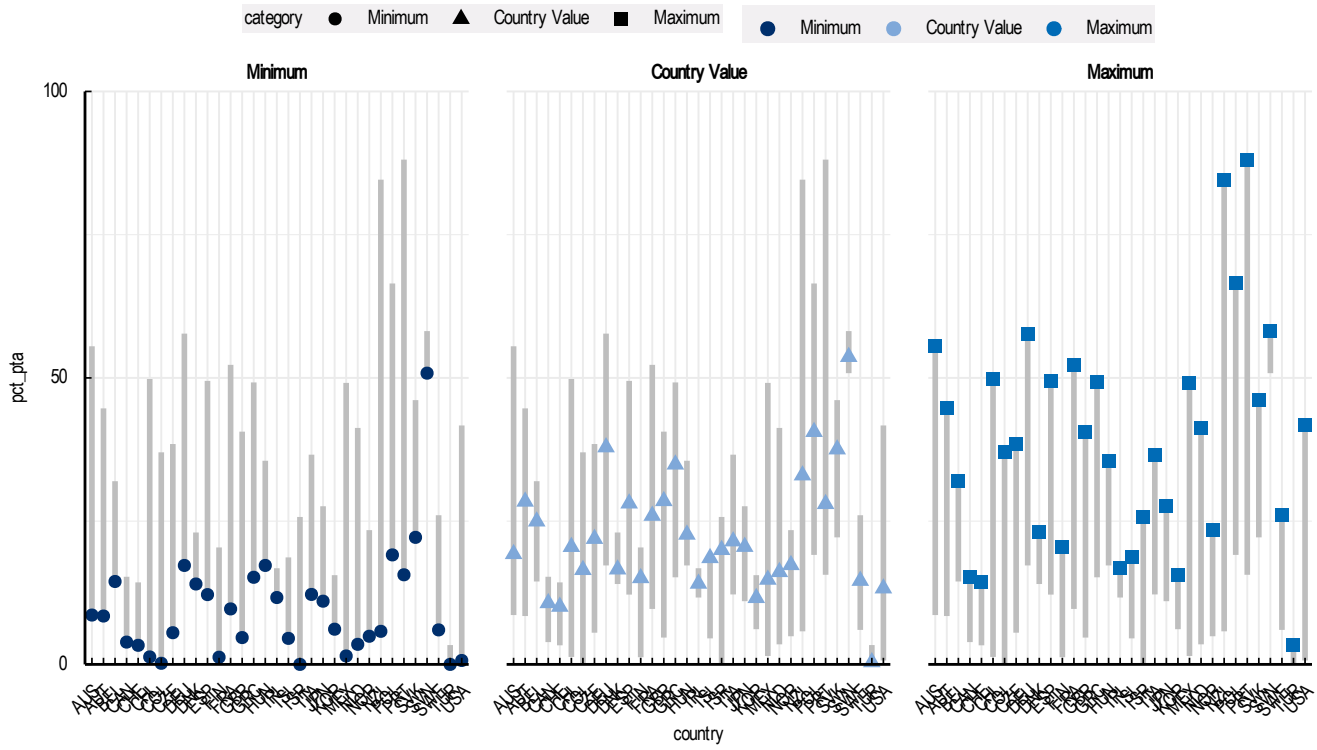
Simple max-min plot:

```
oecd_maxmin(pta, x = country, y = pct_pta, colour = category, group = category)
```



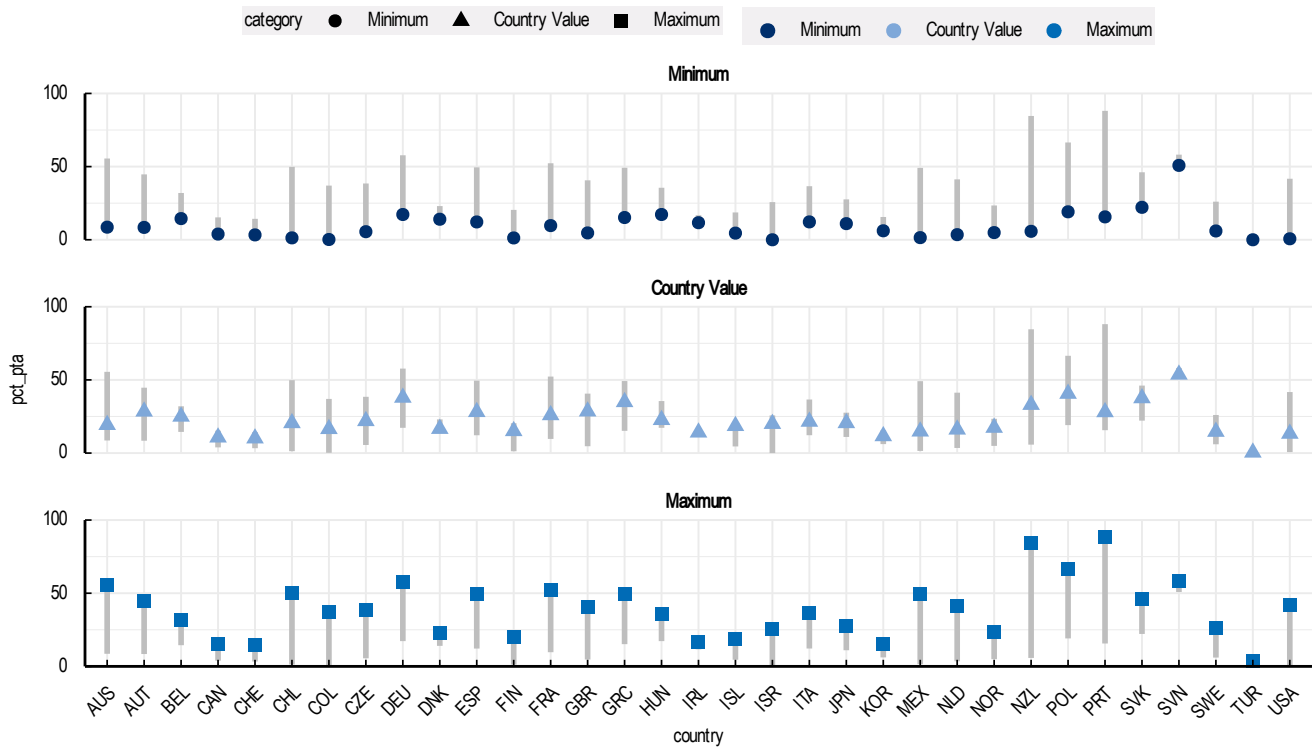
Faceted max-min plot:

```
oecd_maxmin(pta,
  x = country, y = pct_pta, colour = category, group = category,
  facet = category
)
```



Faceted max-min plot with custom ordering:

```
oecd_maxmin(pta,
  x = country, y = pct_pta, colour = category, group = category,
  facet = category, facet_ncol = 1
)
```


category
Minimum
Country Value
Maximum
Minimum
Country Value
Maximum
Minimum
Country Value
Maximum
pct_pta
100
50
0
AUS
AUT
BEL
CAN
CHE
CHL
COL
CZE
DEU
DNK
ESP
FIN
FRA
GBR
GRC
HUN
IRL
ISL
ISR
ITA
JPN
KOR
MEX
NLD
NOR
NZL
POL
PRT
SVK
SVN
SWE
TUR
USA
country

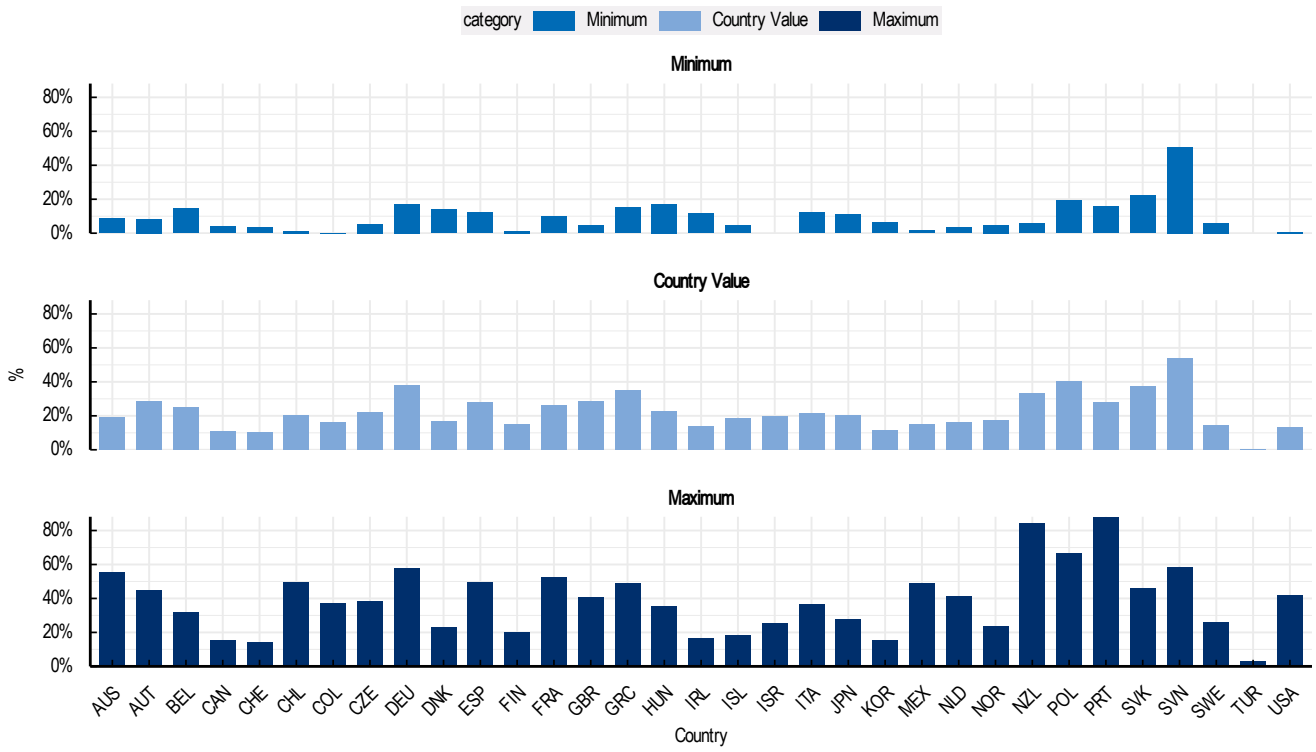
CHAPTER 6

What If?

6.1. Exporting plots for official publication

Suppose you have the next plot.

```
library(ggplot2)
library(oecdplot)

load_oecd_fonts()

ggplot(data = pta, aes(x = country, y = pct_pta / 100, fill = category)) +
  geom_col(position = "dodge2", width = 0.7) +
  theme_oecd() +
  scale_fill_oecd_d(option = "darkblue", direction = -1) +
  scale_y_continuous(labels = scales::percent, expand = c(0,0)) +
  facet_wrap(~category, ncol = 1) +
  labs(
    x = "Country",
    y = "%"
  )
```



We would only need to use the `save_oecd_chart()` function which already provides PAC-compatible aesthetic elements including dimensions and formats.

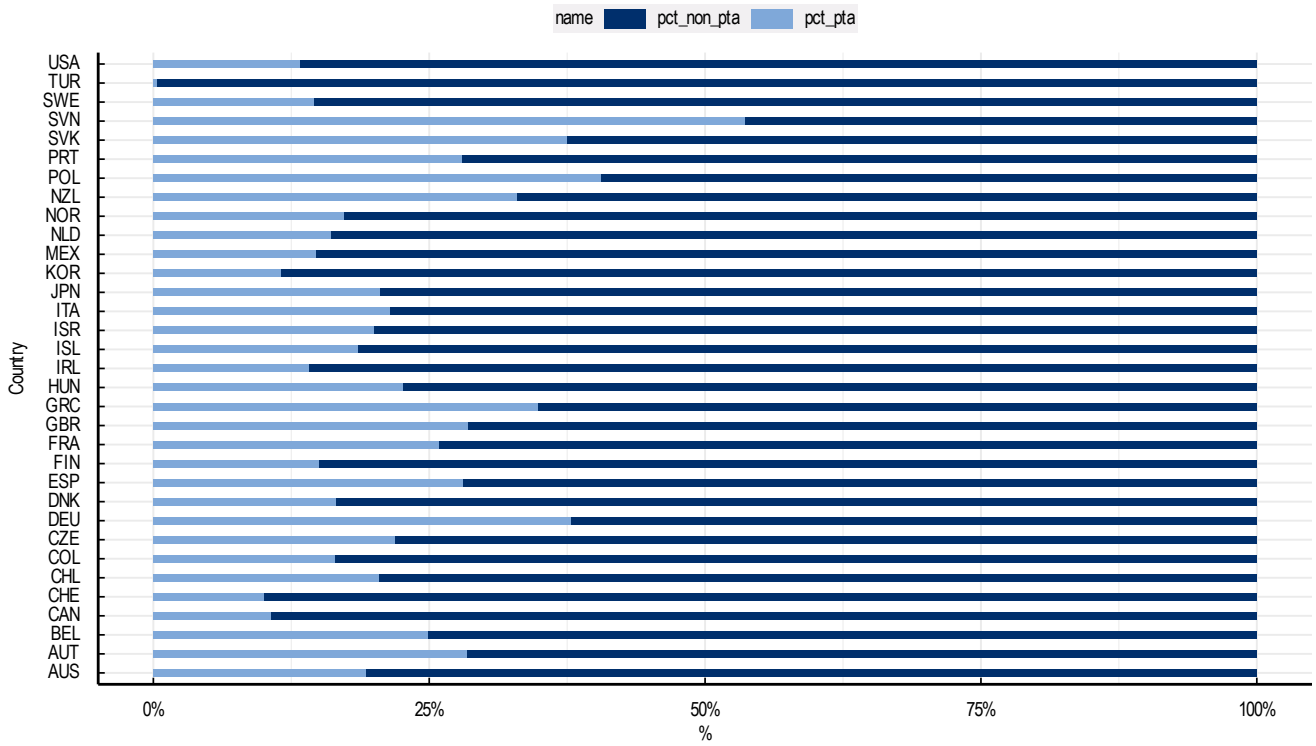
```
save_oecd_chart(
  file_name = "Figure 1",
  plot = last_plot(),
  folder = "my_folder/",
  size = "1/2",
  plot_title = "Figure 1. Regional disparities in protected terrestrial areas, 2017",
  plot_note = "Protected terrestrial areas refers to all protected areas recorded
   in the World Database on Protected Areas (WDPA)",
  plot_source = "OECD calculations based World Database on Protected Areas (WDPA).
   OECD (2020), OECD Regional Statistics (database)",
  statlink_create = TRUE,
  statlink_dataframe = pta,
  statlink_filename = "pta",
  box = FALSE,
  custom_size = NULL,
  format = "emf"
)
```

In the code above we can also pass `format = c("png","emf")` to create two files at the same time.

6.2. Trailing and leading spaces in the plotting area

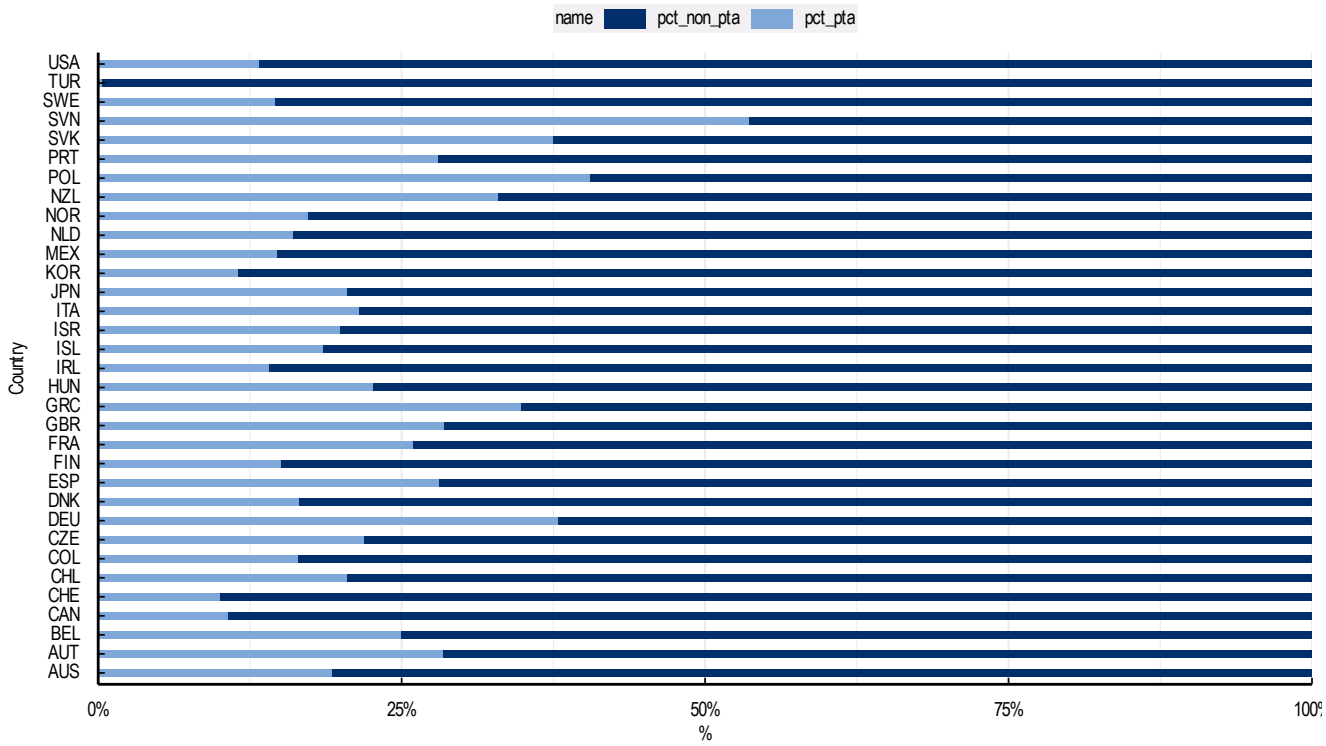
Consider the next plot, where we already applied the OECD theme and formatted the labels as percentage.

```
library(dplyr)
library(tidyr)
library(ggplot2)
library(oecdplot)

load_oecd_fonts()

pta2 <- pta %>%
  filter(category == "Country Value") %>%
  mutate(pct_non_pta = 100 - pct_pta) %>%
  pivot_longer(pct_pta:pct_non_pta) %>%
  mutate(value = value / 100)

pta2_plot <- pta2 %>%
  ggplot(aes(x = value, y = country)) +
  geom_col(aes(fill = name), width = 0.4) +
  scale_fill_oecd_d() +
  labs(y = "Country", x = "%") +
  theme_oecd(base_x_axis_angle = 0)+
  scale_x_continuous(
    labels = scales::percent_format(accuracy = 1)
  )

pta2_plot
```



One option to remove the whitespace in the plot is to use the `expand` parameter, in order to start from the (0,0) coordinate.

```
pta2_plot +
  scale_x_continuous(
    expand = c(0,0),
    labels = scales::percent_format(accuracy = 1)
  )
```



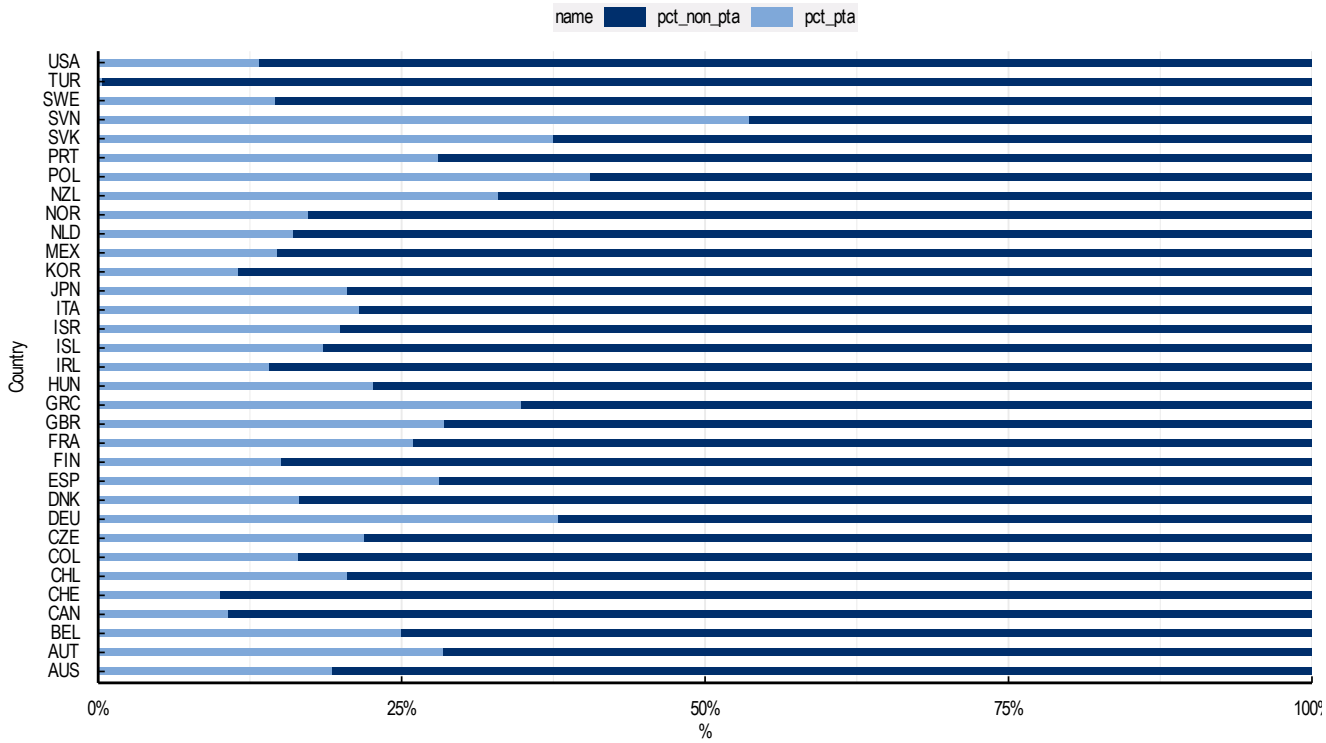
This plot has a problem, the "100%" label is outside the margins, which is covered in then next what-if.

6.3. The x-axis contains off-margin labels

Consider the next plot, where we added different adjustments, such as the x-axis labels rotation.

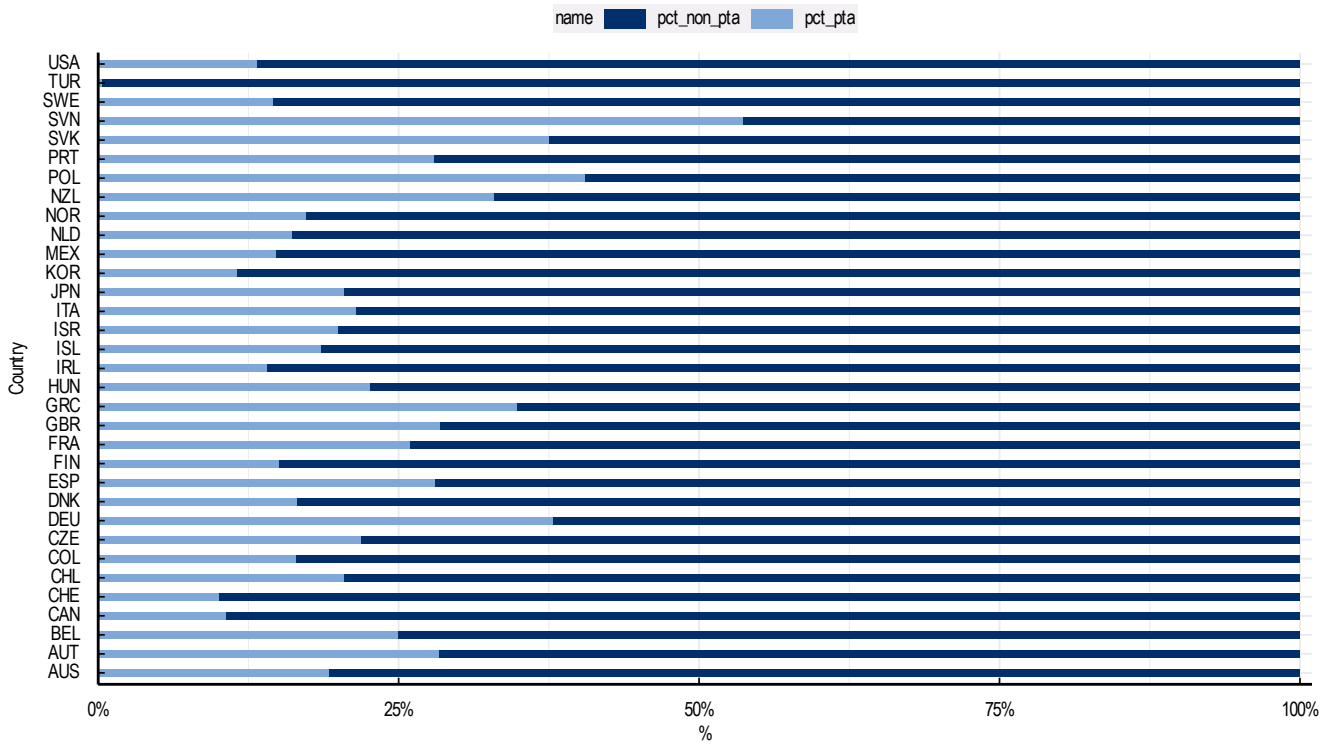
```
library(dplyr)
library(tidyr)
library(ggplot2)
library(oecdplot)

load_oecd_fonts()

pta2 <- pta %>%
  filter(category == "Country Value") %>%
  mutate(pct_non_pta = 100 - pct_pta) %>%
  pivot_longer(pct_pta:pct_non_pta) %>%
  mutate(value = value / 100)

pta2_plot <- pta2 %>%
  ggplot(aes(x = value, y = country)) +
  geom_col(aes(fill = name), width = 0.4) +
  scale_fill_oecd_d() +
```

```
  labs(y = "Country", x = "%") +
  theme_oecd(base_x_axis_angle = 0)+
  scale_x_continuous(
    expand = c(0,0),
    labels = scales::percent_format(accuracy = 1)
  )

pta2_plot
```



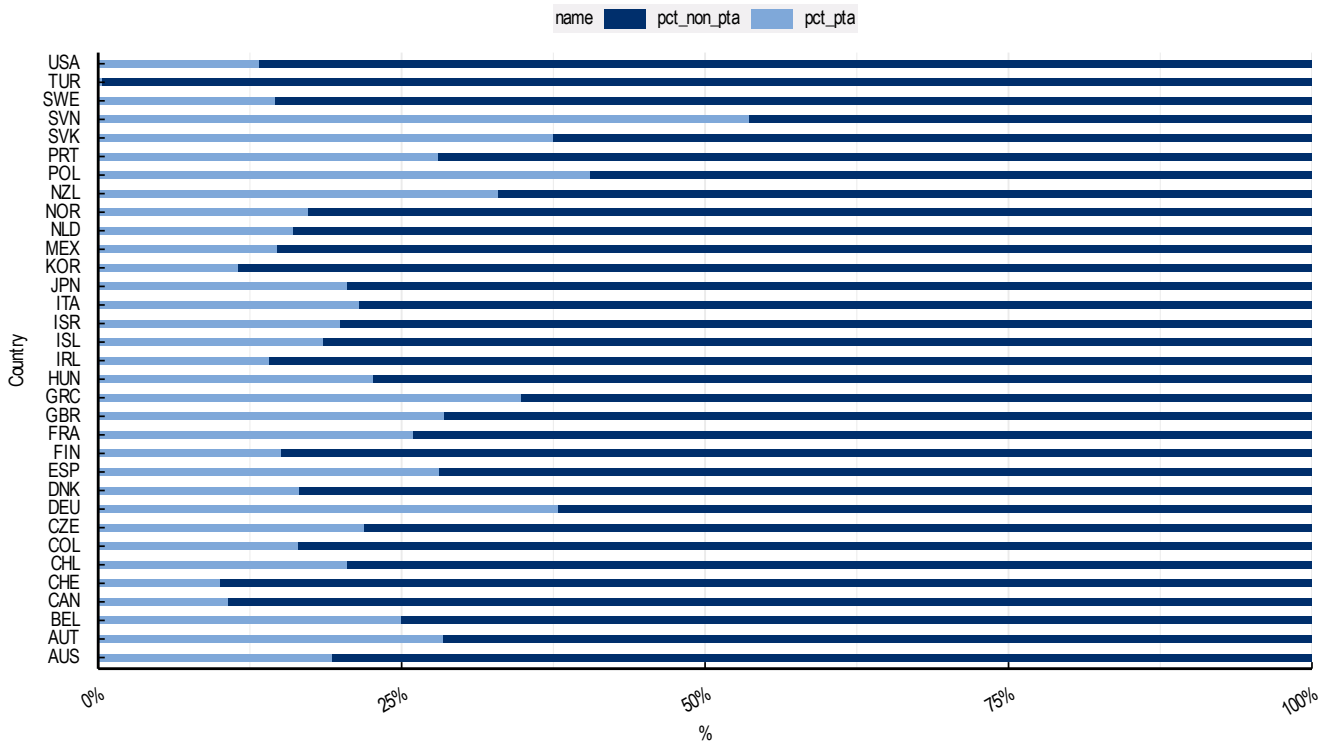
In this case, we can alter the x-axis scale options to define the range to go from 0 to 1.01 (or 1.02-1.05).

```
pta2_plot +
  scale_x_continuous(
    expand = c(0,0),
    limits = c(0,1.01),
    labels = scales::percent_format(accuracy = 1)
  )
```



An alternative solution could be rotating the labels.

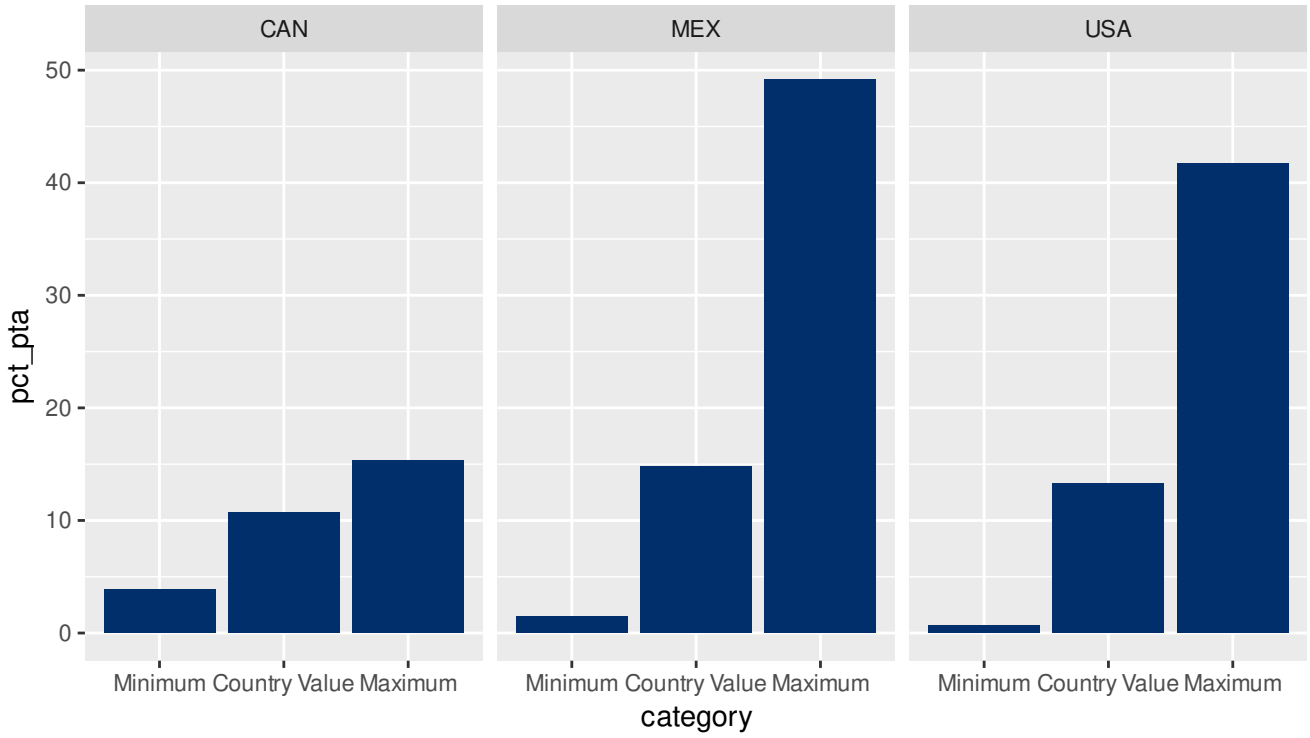
```
pta2_plot +
  theme_oecd(base_x_axis_angle = 30)
```



6.4. Too many lines/columns

One option is to use ggplot2's `facet_wrap()`, as in the next example.

```
ggplot(pta %>% filter(country %in% c("CAN", "USA", "MEX"))) +
  geom_col(aes(x = category, y = pct_pta), fill = oecd_clrs()[1]) +
  facet_wrap(~country)
```


CAN
MEX
USA
pct_pta
50
40
30
20
10
0
Minimum Country Value Maximum
Minimum Country Value Maximum
Minimum Country Value Maximum
category