```
library(knitr)
knitr::opts_chunk$set(fig.show = "hide",
                      fig.path = 'atlas/',
                      fig.width = 11,
                      fig.height = 7,
                      tidy = FALSE,
                      #message = FALSE,
                      out.width = "11in",
                      out.height = "7in")
```

```
library(ggplot2)
library(scales)
library(grattan)
```

```
##
## Attaching package:  'grattan'
##
## The following object is masked from 'package:datasets':
##
##     Orange
```

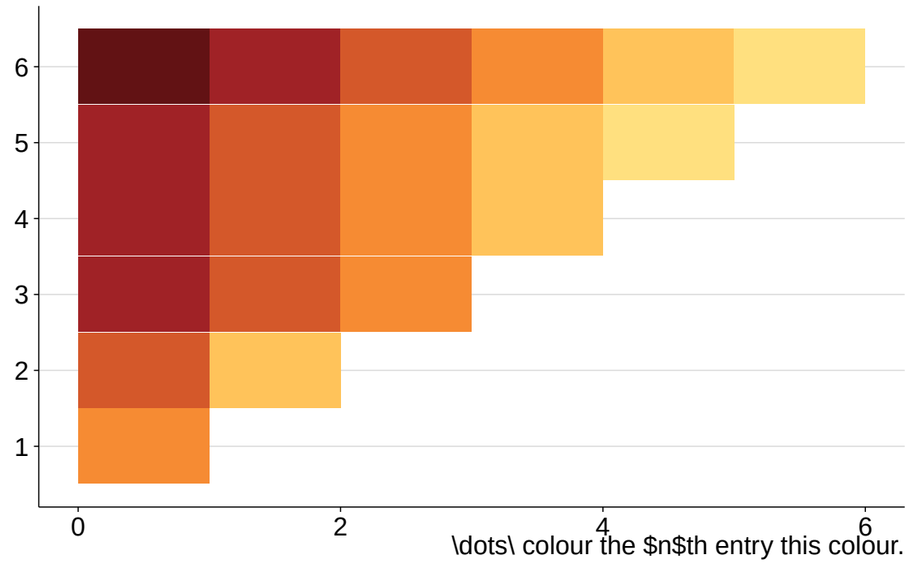
```
library(tidyr)
library(dplyr)
```

```
##
## Attaching package:  'dplyr'
##
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(magrittr)
```

```
##
## Attaching package:  'magrittr'
##
## The following object is masked from 'package:tidyr':
##
##     extract
```

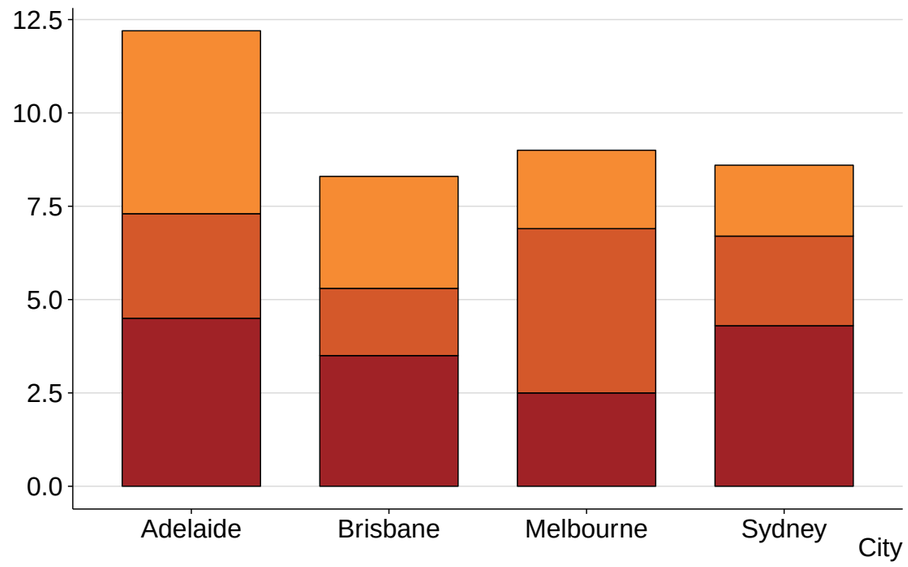
```r
p <- grplot(NULL)
for (y in 1:6){
  for (x in 1:6){
    p <- p + annotate("rect",
                      xmin = x - 1, xmax = x,
                      ymin = y - 0.99, ymax = y,
                      fill = gpal(y)[x])
  }
}
p +
  scale_y_continuous("With this number of categories",
                     breaks = 0:6 + 0.5, labels = 1:7, limits = c(0,6)) +
  scale_x_continuous("\\dots\\ colour the $n$th entry this colour.")
```



```
p <- grplot(NULL)
x <- 0
for (gray in c(217,174,130,87,43,0)/256){
  x <- x + 1
  p <- p + annotate("rect", xmin = x - 1, xmax = x, ymin = 0.5, ymax = 1.5,
                    fill = rgb(gray, gray, gray)) +
    coord_equal() +
    theme_void()
}
p
```

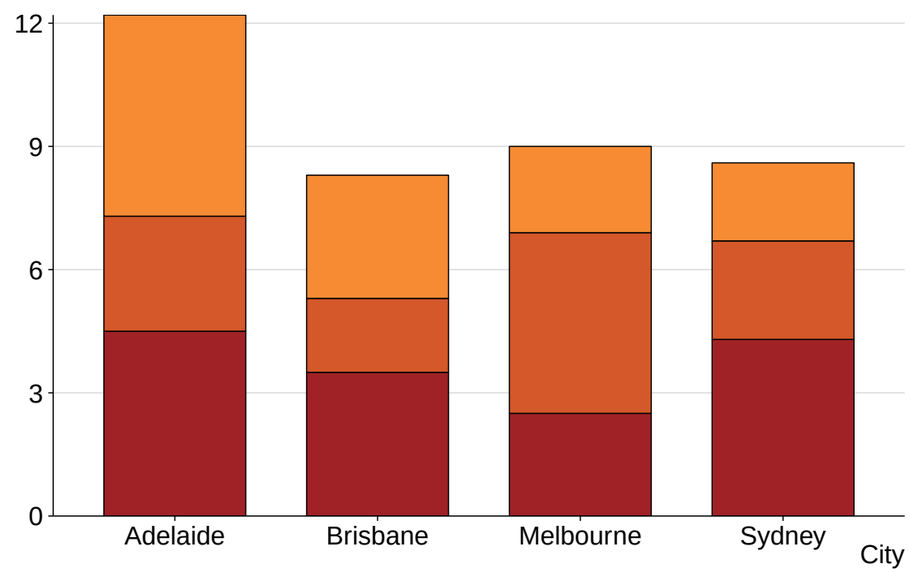
0.1 Bar charts

```
read.table(text ="
City  1994 1995 1996
Sydney 4.3 2.4 1.9
Melbourne 2.5 4.4 2.1
Brisbane 3.5 1.8 3
Adelaide 4.5 2.8 4.9
", header = TRUE,
check.names = FALSE) %>%
  # First we need to 'tidy' our data from
  # 'wide' (which it is now) to 'long'.
  # ggplot (and hence grplot) requires every
  # row to refer to a single data point ---
  # i.e. a single bar
  tidyr::gather(Year, value, -City) %>%
  # Year needs to be factor(Year) so that the colours
  # are distinctive -- leaving it makes the colors be
  # elements on a range 1994-1996
  grplot(aes(x = City, y = value, fill = factor(Year))) +
  # set stat = "identity" so that the height of the bar
  # represents values, not counts of entries
  geom_bar(stat = "identity", width = 0.7)
```



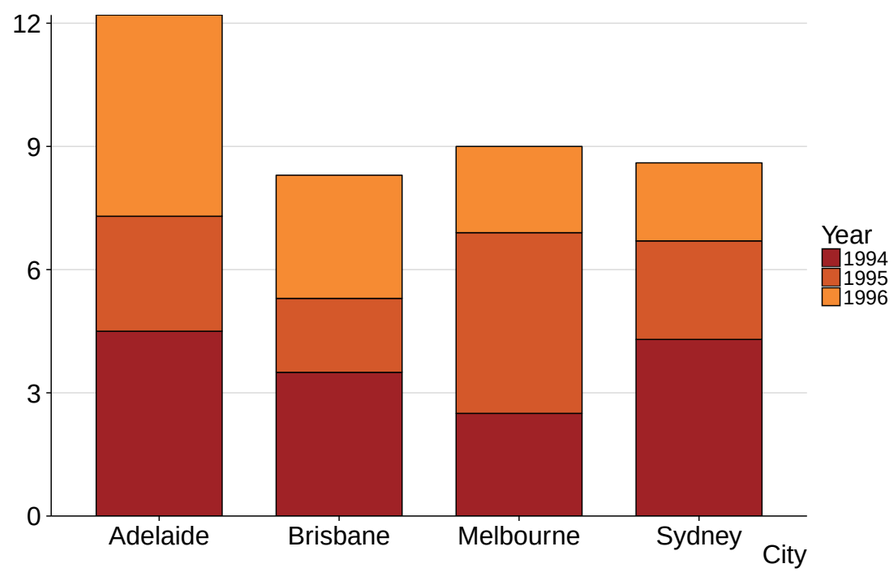
0.1.1 Axes flush with data

```
read.table(text ="
City  1994 1995 1996
Sydney 4.3 2.4 1.9
Melbourne 2.5 4.4 2.1
Brisbane 3.5 1.8 3
Adelaide 4.5 2.8 4.9
", header = TRUE,
check.names = FALSE) %>%
  # First we need to 'tidy' our data from
  # 'wide' (which it is now) to 'long'.
  # ggplot (and hence grplot) requires every
  # row to refer to a single data point ---
  # i.e. a single bar
  tidyr::gather(Year, value, -City) %>%
  # Year needs to be factor(Year) so that the colours
  # are distinctive -- leaving it makes the colors be
  # elements on a range 1994-1996
  grplot(aes(x = City, y = value, fill = factor(Year))) +
  # set stat = "identity" so that the height of the bar
  # represents values, not counts of entries
  geom_bar(stat = "identity", width = 0.7) +


  # To adjust the spacing between the axes, use expand = c(0,0)
  scale_y_continuous(expand = c(0,0))
```



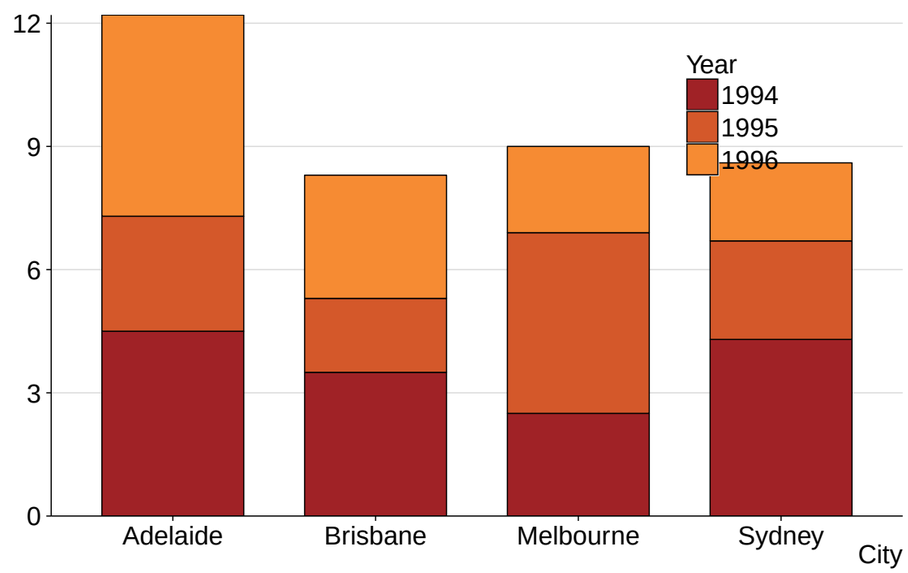
0.1.2 Add legend

```
read.table(text ="
City  1994 1995 1996
Sydney 4.3 2.4 1.9
Melbourne 2.5 4.4 2.1
Brisbane 3.5 1.8 3
Adelaide 4.5 2.8 4.9
", header = TRUE,
check.names = FALSE) %>%
  # First we need to 'tidy' our data from
  # 'wide' (which it is now) to 'long'.
  # ggplot (and hence grplot) requires every
  # row to refer to a single data point ---
  # i.e. a single bar
  tidyr::gather(Year, value, -City) %>%
  # Year needs to be factor(Year) so that the colours
  # are distinctive -- leaving it makes the colors be
  # elements on a range 1994-1996
  grplot(aes(x = City, y = value, fill = factor(Year))) +
  # set stat = "identity" so that the height of the bar
  # represents values, not counts of entries
  geom_bar(stat = "identity", width = 0.7) +


  # To adjust the spacing between the axes, use expand = c(0,0)
  scale_y_continuous(expand = c(0,0)) +

  # Include a legend:
  # Without the following, we get `factor(Year)` as the
  # legend title. This sets the legend title.
  guides(fill = guide_legend("Year")) +
  theme(legend.position = "right")
```



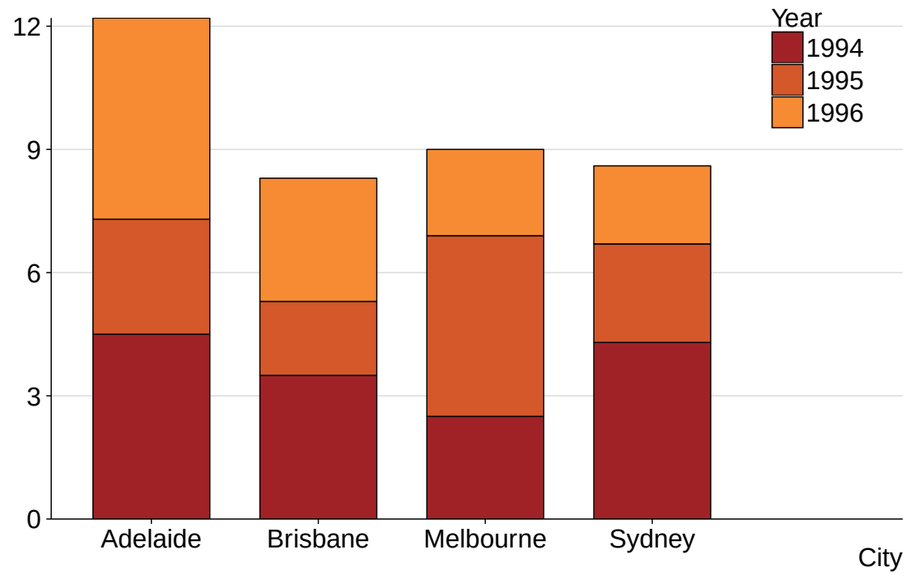
0.1.3 Add legend over plot

```
read.table(text ="
City  1994 1995 1996
Sydney 4.3 2.4 1.9
Melbourne 2.5 4.4 2.1
Brisbane 3.5 1.8 3
Adelaide 4.5 2.8 4.9
", header = TRUE,
check.names = FALSE) %>%
  # First we need to 'tidy' our data from
  # 'wide' (which it is now) to 'long'.
  # ggplot (and hence grplot) requires every
  # row to refer to a single data point ---
  # i.e. a single bar
  tidyr::gather(Year, value, -City) %>%
  # Year needs to be factor(Year) so that the colours
  # are distinctive -- leaving it makes the colors be
  # elements on a range 1994-1996
  grplot(aes(x = City, y = value, fill = factor(Year))) +
  # set stat = "identity" so that the height of the bar
  # represents values, not counts of entries
  geom_bar(stat = "identity", width = 0.7) +


  # To adjust the spacing between the axes, use expand = c(0,0)
  scale_y_continuous(expand = c(0,0)) +

  # Include a legend:
  # Without the following, we get `factor(Year)` as the
  # legend title. This sets the legend title.
  guides(fill = guide_legend("Year")) +
  # you can also adjust the legend position ranged
  # c(0,0) = southwest corner
  # c(1,0) = southeast corner
  # c(0,1) = northwest corner
  # c(1,1) = northeast corner
  # ranged between
  theme(legend.position = c(0.8, 0.8),
        # play with unit(<width> , "lines")
        legend.text = element_text(size = 23),
        legend.key.size = unit(2, "lines"))
```



0.1.4 Adding space for legend

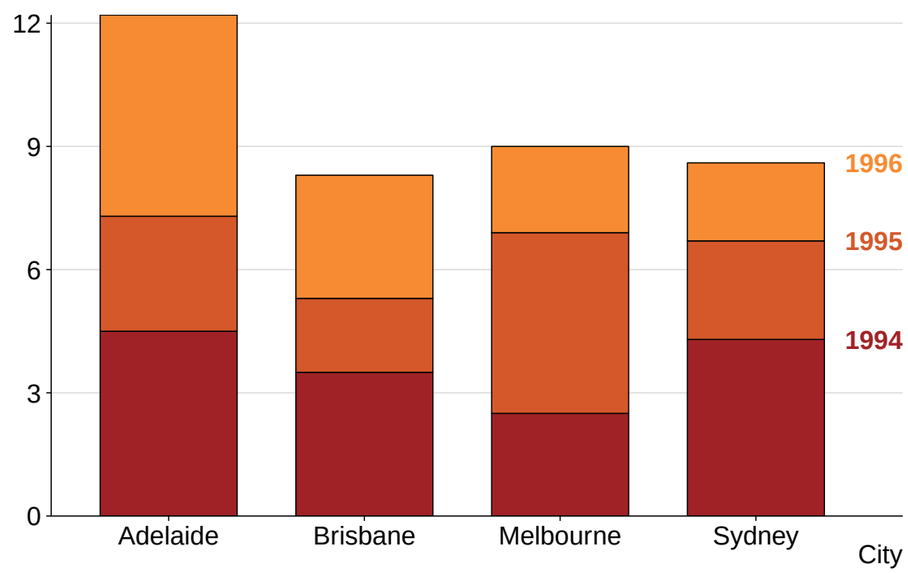
```
read.table(text ="
City  1994 1995 1996
Sydney 4.3 2.4 1.9
Melbourne 2.5 4.4 2.1
Brisbane 3.5 1.8 3
Adelaide 4.5 2.8 4.9
", header = TRUE,
check.names = FALSE) %>%
  # First we need to 'tidy' our data from
  # 'wide' (which it is now) to 'long'.
  # ggplot (and hence grplot) requires every
  # row to refer to a single data point ---
  # i.e. a single bar
  tidyr::gather(Year, value, -City) %>%
  # Year needs to be factor(Year) so that the colours
  # are distinctive -- leaving it makes the colors be
  # elements on a range 1994-1996
  grplot(aes(x = City, y = value, fill = factor(Year))) +
  # set stat = "identity" so that the height of the bar
  # represents values, not counts of entries
  geom_bar(stat = "identity", width = 0.7) +


  # To adjust the spacing between the axes, use expand = c(0,0)
  scale_y_continuous(expand = c(0,0)) +

  # Include a legend:
  # Without the following, we get `factor(Year)` as the
  # legend title. This sets the legend title.
  guides(fill = guide_legend("Year")) +
  # you can also adjust the legend position ranged
  # c(0,0) = southwest corner
  # c(1,0) = southeast corner
  # c(0,1) = northwest corner
  # c(1,1) = northeast corner
  # ranged between
  theme(legend.position = c(0.9, 0.9),
        # play with unit(<width> , "lines")
        legend.text = element_text(size = 23),
        legend.key.size = unit(2, "lines")) +
  #
  # you can use blank annotations to expand the axis
  annotate("blank",
           x = 5.5,
           y = NA_real_)
```



0.1.5 Coloured text as legend

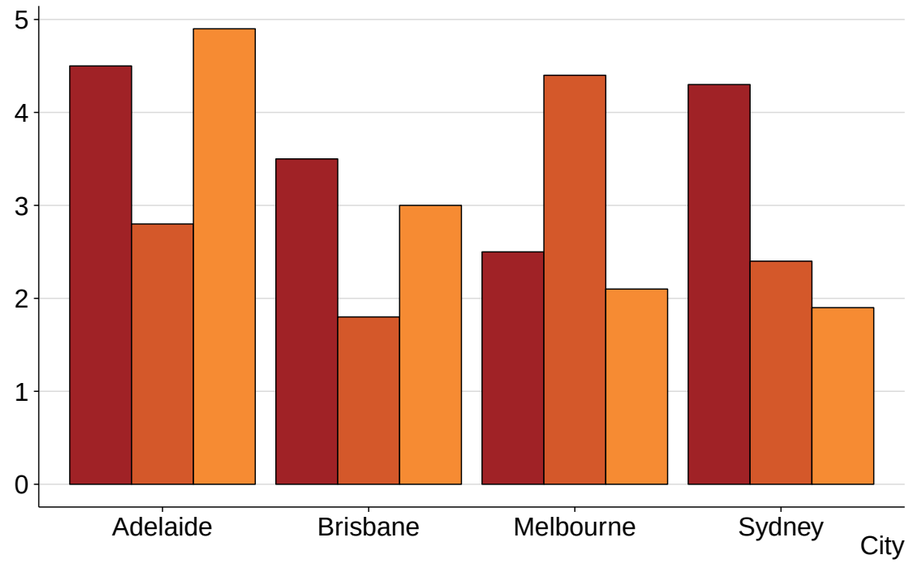
```
read.table(text ="
City  1994 1995 1996
Sydney 4.3 2.4 1.9
Melbourne 2.5 4.4 2.1
Brisbane 3.5 1.8 3
Adelaide 4.5 2.8 4.9
", header = TRUE,
check.names = FALSE) %>%
  # First we need to 'tidy' our data from
  # 'wide' (which it is now) to 'long'.
  # ggplot (and hence grplot) requires every
  # row to refer to a single data point ---
  # i.e. a single bar
  tidyr::gather(Year, value, -City) %>%
  # Year needs to be factor(Year) so that the colours
  # are distinctive -- leaving it makes the colors be
  # elements on a range 1994-1996
  #
  # add a label variable. Exploit the fact that if any
  # variable is NA then the text will not be plotted.
  mutate(text.label = ifelse(as.numeric(City) == max(as.numeric(City)),
                             as.character(Year),
                             NA_character_),
         text.x = as.numeric(factor(City)) + 0.75) %>%
  group_by(City) %>%
  mutate(text.y = cumsum(value)) %>%
  grplot(aes(x = City, y = value, fill = factor(Year))) +
  # set stat = "identity" so that the height of the bar
  # represents values, not counts of entries
  geom_bar(stat = "identity", width = 0.7) +


  # To adjust the spacing between the axes, use expand = c(0,0)
  scale_y_continuous(expand = c(0,0)) +

  # Alternatively, we can use geom_text to place text
  geom_text(aes(x = text.x, y = text.y,
                label = text.label, color = factor(Year)),
            size = 23/(14/5),
            hjust = 1,
            fontface = "bold")
```

```
## Warning:  Removed 9 rows containing missing values (geom_text).
```



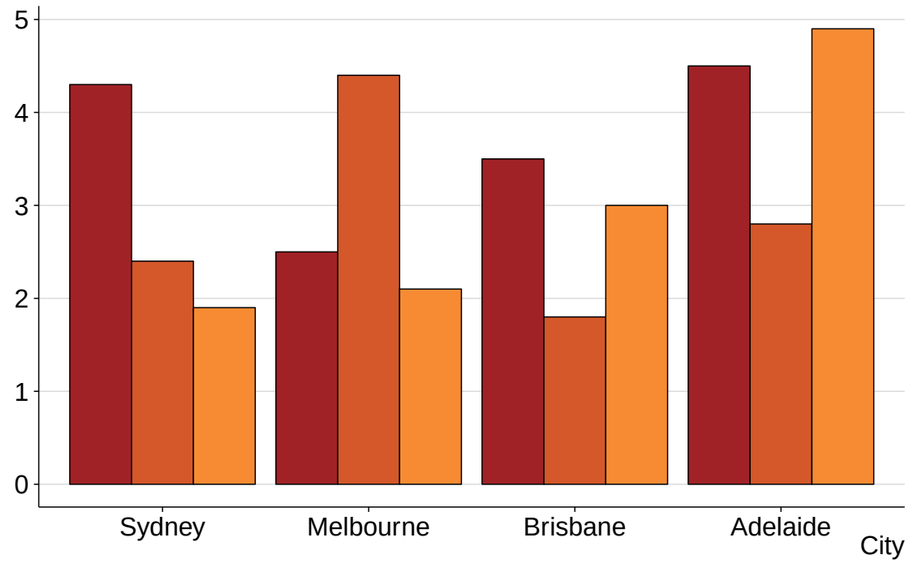
0.1.6 "Dodged" bar charts

```
read.table(text ="
City  1994 1995 1996
Sydney 4.3 2.4 1.9
Melbourne 2.5 4.4 2.1
Brisbane 3.5 1.8 3
Adelaide 4.5 2.8 4.9
", header = TRUE,
check.names = FALSE) %>%
  # First we need to 'tidy' our data from
  # 'wide' (which it is now) to 'long'.
  # ggplot (and hence grplot) requires every
  # row to refer to a single data point ---
  # i.e. a single bar
  tidyr::gather(Year, value, -City) %>%
  # Year needs to be factor(Year) so that the colours
  # are distinctive -- leaving it makes the colors be
  # elements on a range 1994-1996
  grplot(aes(x = City, y = value, fill = factor(Year))) +
  # set stat = "identity" so that the height of the bar
  # represents values, not counts of entries
  geom_bar(stat = "identity", width = 0.9,
           # position = "dodge"
           position = "dodge")
```



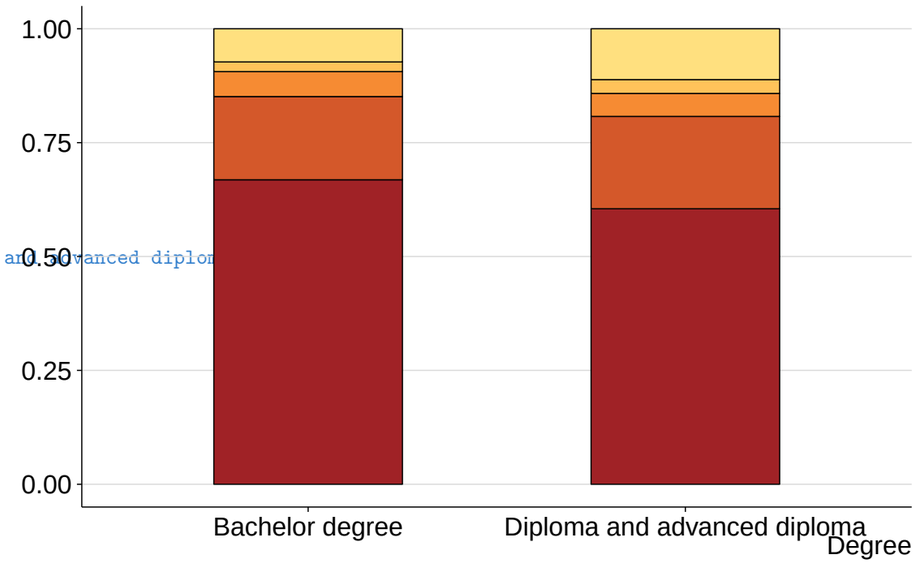
0.1.7 Order bars

```
read.table(text ="
City  1994 1995 1996
Sydney 4.3 2.4 1.9
Melbourne 2.5 4.4 2.1
Brisbane 3.5 1.8 3
Adelaide 4.5 2.8 4.9
", header = TRUE,
check.names = FALSE) %>%



  tidyr::gather(Year, value, -City) %>%
  #
  #
  #
  # use levels IMMEDIATELY before grplot to reorder bars.
  mutate(City = factor(City,
                       levels = c("Sydney",
                                  "Melbourne",
                                  "Brisbane",
                                  "Adelaide"))) %>%

  grplot(aes(x = City, y = value, fill = factor(Year))) +
  # set stat = "identity" so that the height of the bar
  # represents values, not counts of entries
  geom_bar(stat = "identity", width = 0.9,
           # position = "dodge"
           position = "dodge")
```



0.1.8 Stacked (filled) bar charts

```
read.table(text = "
Degree  Full-time Part-time Away from work Unemployed Not in LF
Diploma and advanced diploma 60.48452522 20.27784088 5.035534013 3.029462697
11.17263719
Bachelor degree 66.85055608 18.26301611 5.480946866 2.121800946 7.283680001
", header = TRUE, sep = "\t") %>%
  tidyr::gather(Status, value, -Degree) %>%
  mutate(Degree = factor(Degree, levels = c("Bachelor degree", "dummy", "Diploma and advanced diplom
  grplot(aes(x = Degree,
             y = value, fill = Status)) +
  geom_bar(stat = "identity", position = "fill", width = 0.5)
```



0.1.9 New colors

```
read.table(text = "
Degree  Full-time Part-time Away from work Unemployed Not in LF
Diploma and advanced diploma 60.48452522 20.27784088 5.035534013 3.029462697
11.17263719
Bachelor degree 66.85055608 18.26301611 5.480946866 2.121800946 7.283680001
", header = TRUE, sep = "\t") %>%
  tidyr::gather(Status, value, -Degree) %>%
  mutate(Degree = factor(Degree,
                         levels = c("Bachelor degree",
                                    "dummy",
                                    "Diploma and advanced diploma"))) %>%
  grplot(aes(x = Degree,
             y = value, fill = Status)) +
  geom_bar(stat = "identity", position = "fill", width = 0.5) +
  scale_fill_manual(values = c(gpal(6)[1:4], "black")) +
  scale_x_discrete(expand = c(0.05,0.05)) +
  # get rid of the x-axis title (and the space allocated thereto)
  theme(axis.title = element_blank()) +
  # make data flush with plot background
  # and the axis "%"
  scale_y_continuous(expand = c(0,0),
                     label = percent) +
  annotate("text",
           x = 1.5,
           # Probably best to manually position text here!
           y = c(0.4, 0.75, 0.85, 0.9, 0.95),
           color = c(gpal(6)[1:4], "black"),
           fontface = "bold",
           size = 23/(14/5),
           label = c("Full-time", "Part-time", "Away from work", "Unemployed", "Not in LF"))

## Scale for 'fill' is already present.  Adding another scale for 'fill',
## which will replace the existing scale.
```