Saving time with ggplot and cowplot

Théotime Colin

28/05/2020

1) Goals

In this tutorial, you will learn to:

- use in built datasets to formulate reproducible questions
- make your code tidy and reproducible
- use the pipe '%>%' operator from the 'tidyr' library
- use grouping and data handling functions from the 'dplyr' library
- use ggplot2 to make nice figures
- assemble these figures into an elaborate panel

I won't go into too much detail for any of these packages. Instead, this tutorial is designed to give you *just enough* information about how each package work and how to connect them to each other, so that you can get started and start self-learning details that you can apply to your research projects. You probably already know how some of these packages work, but I will explain a few useful tips that can be hard to come across if you don't know they exist.

2) Libraries

I usually start any R script with the command `rm(list=ls())` that erases any stored object from R's memory. Although this appears to be annoying at first because all the packages and datasets will have to be reloaded, it avoids confusion, and helps ensure your script can run smoothly every time R is launched again. It's better to fix the problems from the beginning than to write the whole script and realize there was a bit that depended on another script.

Classicaly, you would load libraries using the `library()` function. Note there is another and sometimes more elegant way to load function from these packages. This can help keep code tidy and remember which function comes from which package. It also avoids loading libraries when only a few of their functions are used in a script. Finally, it avoids creating conflicts between libraries, as they may use the same name for a function that works differently and produces a different output.

A function from a package can be simply loaded in the following way `package::function`: For example, to use the function `gather` from `dplyr`, this works `dplyr::group_by()` and it is identical to using `library(dplyr); group_by()`.

Whenever it is possible, we will use the `package::function` syntax. Here we will call functions from the packages `cowplot` and `dplyr` this way and so we won't load them at the beginning. For some libraries this is impractical. This is notably the case for libraries that have functions that are repeatidly used in a script, or libraries that use specific operators like the `+` in `ggplot2` or the `%>%` from tidyr, that bind different rows of code together, so we will load these two packages directly at the beginning of our script.

```
library(tidyr) # brings a tidy way to write and read code
library(ggplot2) # makes pretty plots
#library(cowplot) # builds panels, functions are called directly
#library(dplyr) # contains useful functions to handle data, functions are called directly
```

3) The 'iris' dataset

R has built-in datasets that are loaded with the environment or with packages, they allow to develop easily reproducible examples of code, bugs, and methods. Let's have a look at the most famous one in ecology, `iris`.

```
head(iris,10)
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1           5.1         3.5          1.4         0.2  setosa
## 2           4.9         3.0          1.4         0.2  setosa
## 3           4.7         3.2          1.3         0.2  setosa
## 4           4.6         3.1          1.5         0.2  setosa
## 5           5.0         3.6          1.4         0.2  setosa
## 6           5.4         3.9          1.7         0.4  setosa
## 7           4.6         3.4          1.4         0.3  setosa
## 8           5.0         3.4          1.5         0.2  setosa
## 9           4.4         2.9          1.4         0.2  setosa
## 10          4.9         3.1          1.5         0.1  setosa
```

We're going to ask a few classic ecology questions to illustrate how to make figures with `ggplot2`, panels with `cowplot`, and handle our data with `tidyr` and `dplyr`:
Do petal widths and petal lengths vary between species?
Do petal lengths vary with petal widths similarly between species?

Pretty figures are worth better than all the p-values in the world, so we'll focus on that during this R user group session. There will be no stats.

4) Do petal widths and petal lengths vary between species?

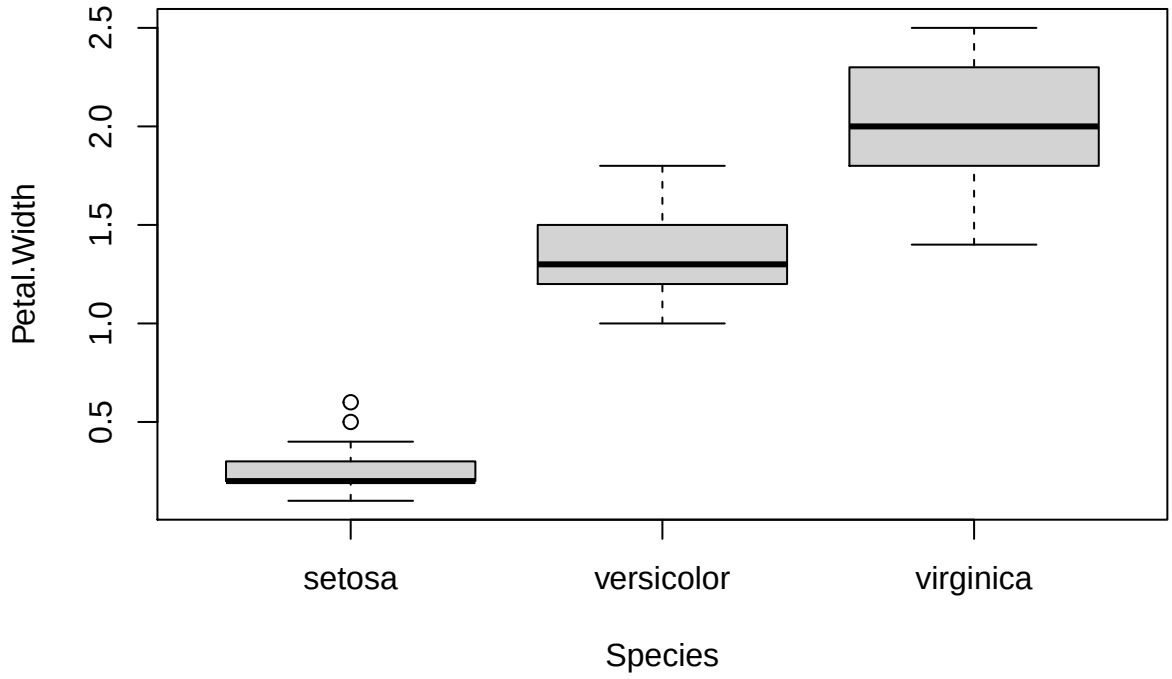
A long time ago in a galaxy far far away, programmers used to make figures in the standard code installed with R, without any additional libraries.

For example, you can make a box and whiskers plot without loading any additional library by using the command `boxplot()`.

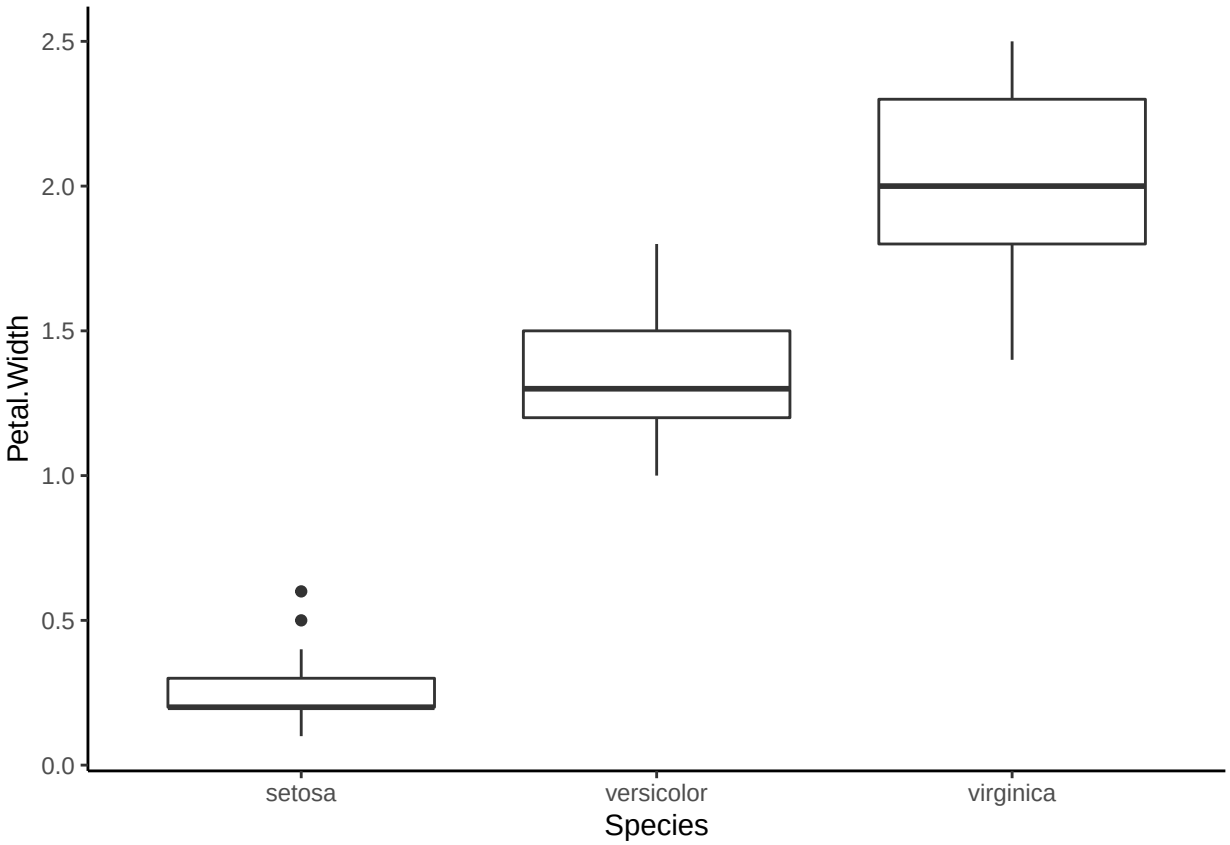
Now almost every body is using the library `ggplot2` which for example contains `geom_boxplot()`, an equivalent to `boxplot()`.

`boxplot()` and `geom_boxplot()` are pretty similar:

```
boxplot(Petal.Width ~ Species, data=iris)
```



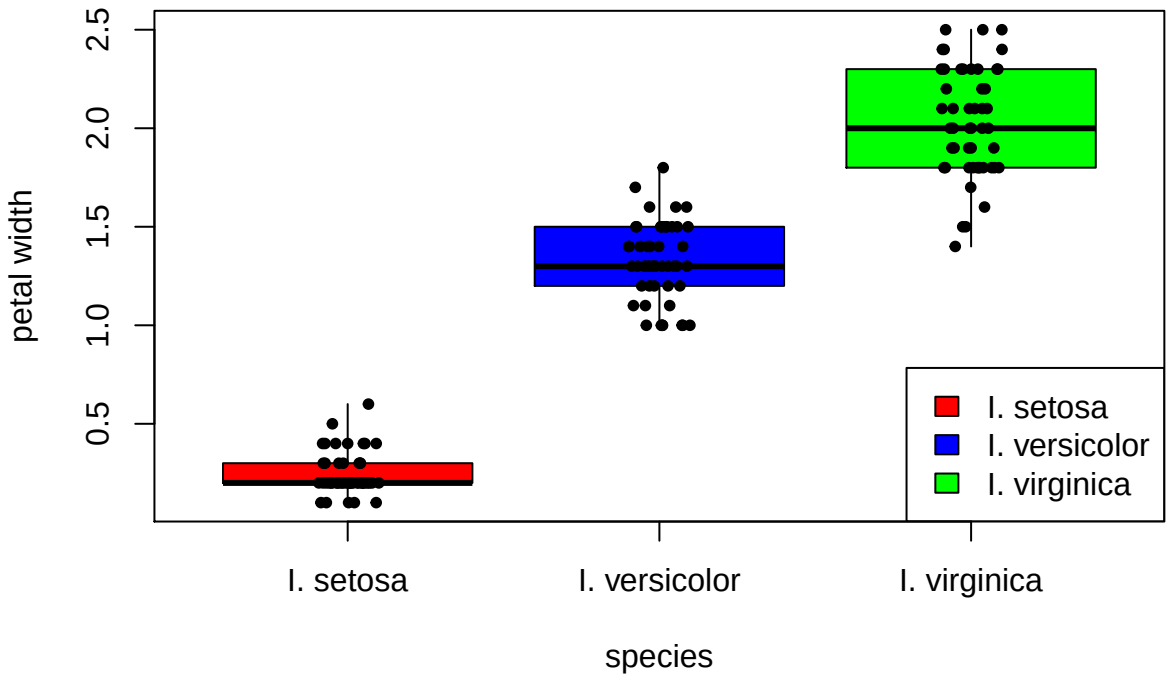

```
ggplot(iris,aes(x=Species, y=Petal.Width))+
  geom_boxplot()+
  theme_classic()
```


Petal.Width
2.5
2.0
1.5
1.0
0.5
0.0
setosa
versicolor
virginica
Species

If you want to make more complex figures, `ggplot2` is more flexible and easier to read than base R.

This is an example if we want to add a legend and colors in base R:

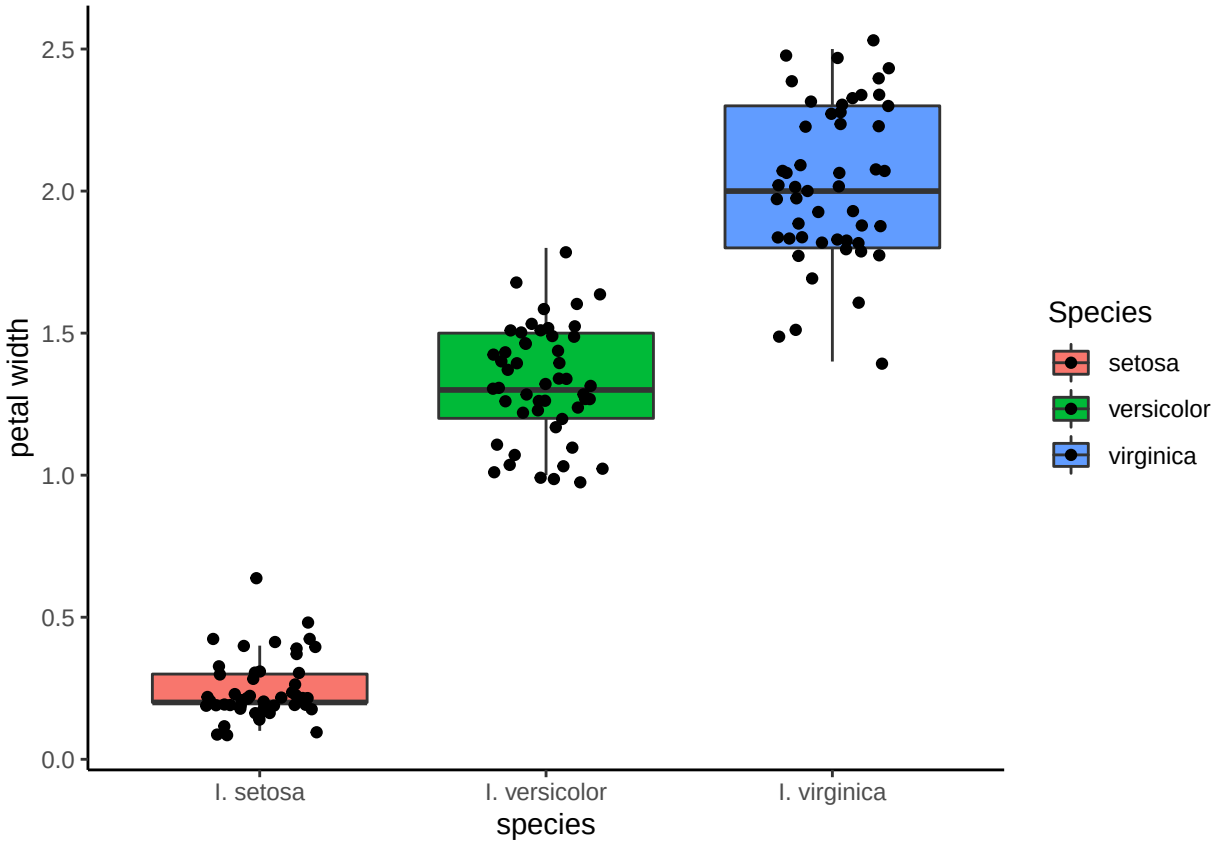
```
boxplot(Petal.Width ~ Species, data=iris,
        range=+Inf,
        staplelty = 0,
        col=c('red','blue','green'),
        xlab='species',
        ylab='petal width',
        names=c("I. setosa", "I. versicolor", "I. virginica"),
        whisklty = 1
)
stripchart(Petal.Width ~ Species, data=iris,
           vertical = TRUE,
           method = "jitter",
           add=T,
           pch = 20,
           group.names=c("I. setosa", "I. versicolor", "I. virginica")
)
legend("bottomright",legend=c("I. setosa", "I. versicolor", "I. virginica"),
       fill=c('red','blue','green')
)
```



It is just a simple plot but already a nightmare spreading to 18 rows.

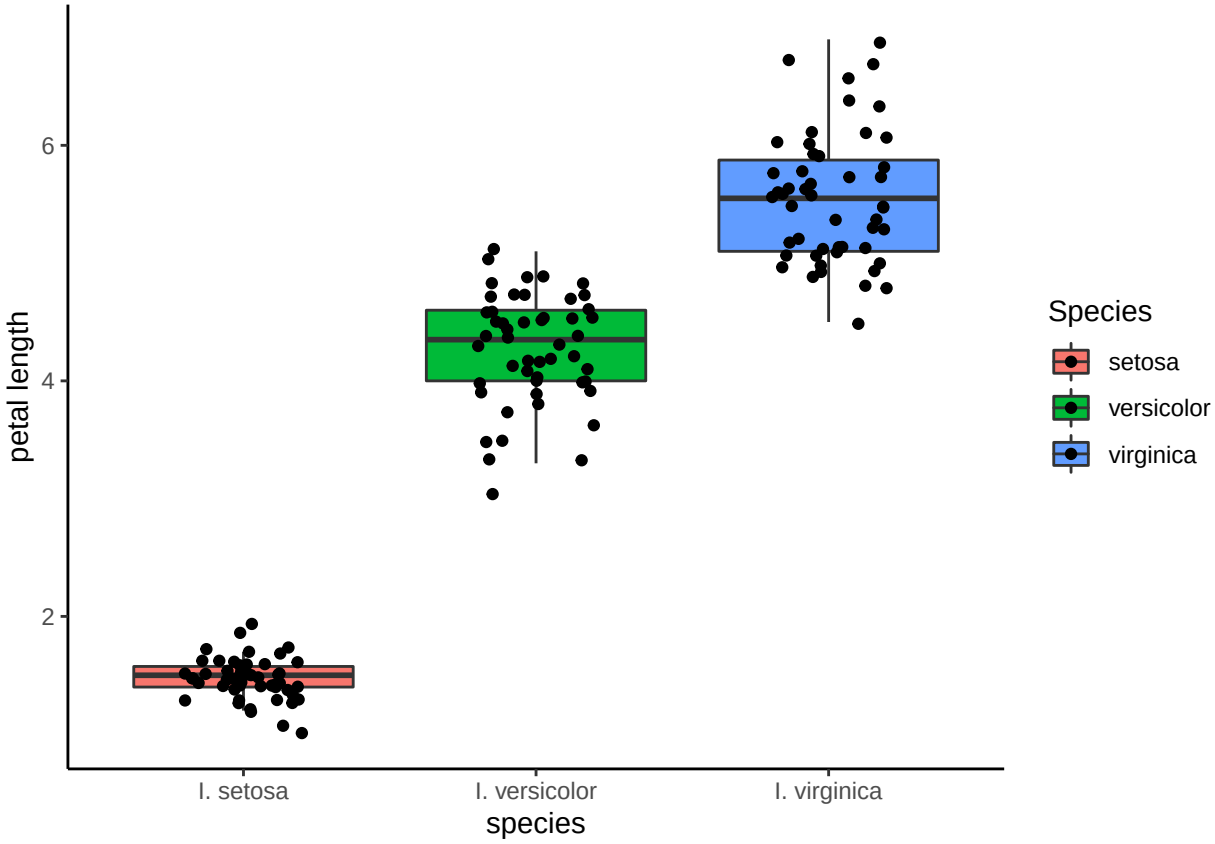
ggplot2 does this in six lines, and the graphics are already nicer:

```
ggplot(iris,aes(x=Species, y=Petal.Width, fill=Species))+
  geom_boxplot(outlier.shape = NA)+
  geom_jitter(width = 0.2)+
  xlab("species")+
  ylab("petal width")+
  scale_x_discrete(labels=c("I. setosa", "I. versicolor", "I. virginica"))+
  theme_classic()
```



And it's so easy to do it for `Petal.Length` too ! only 12 characters to change:

```
ggplot(iris,aes(x=Species, y=Petal.Length, fill=Species))+
  geom_boxplot(outlier.shape = NA)+
  geom_jitter(width = 0.2)+
  xlab("species")+
  ylab("petal length")+
  scale_x_discrete(labels=c("I. setosa", "I. versicolor", "I. virginica"))+
  theme_classic()
```



Note how to use one line per characteristic and the + at the end. This allows to comment any line to come back to it later

Reviewers will probably want you to report group mean +/- sd. You could do it in base R, it's fine for three groups, but can rapidly become annoying.

```
levels(iris$Species)
```

```
## [1] "setosa"     "versicolor" "virginica"
```

```
data.frame(value=c(mean(iris$Petal.Width[iris$Species=='setosa']),
                   mean(iris$Petal.Width[iris$Species=='versicolor']),
                   mean(iris$Petal.Width[iris$Species=='virginica']),
                   sd(iris$Petal.Width[iris$Species=='setosa']),
                   sd(iris$Petal.Width[iris$Species=='versicolor']),
                   sd(iris$Petal.Width[iris$Species=='virginica'])),
           key=c("mean","mean","mean","sd","sd","sd"),
           species=c("setosa","versicolor","virginica","setosa","versicolor","virginica")
)
```

```
##       value  key    species
## 1 0.2460000 mean     setosa
## 2 1.3260000 mean versicolor
## 3 2.0260000 mean  virginica
## 4 0.1053856   sd     setosa
## 5 0.1977527   sd versicolor
## 6 0.2746501   sd  virginica
```

It's long to write and the code is hard to read.

`tidyr` and `dplyr` allow you to get this info quickly. First we transfer the dataset from the *wide* to the *long* format. Then we create groups, here we want to group our data per `Species` and `type`, and for each of these we want to summarise the group values into means, sd and sample size:

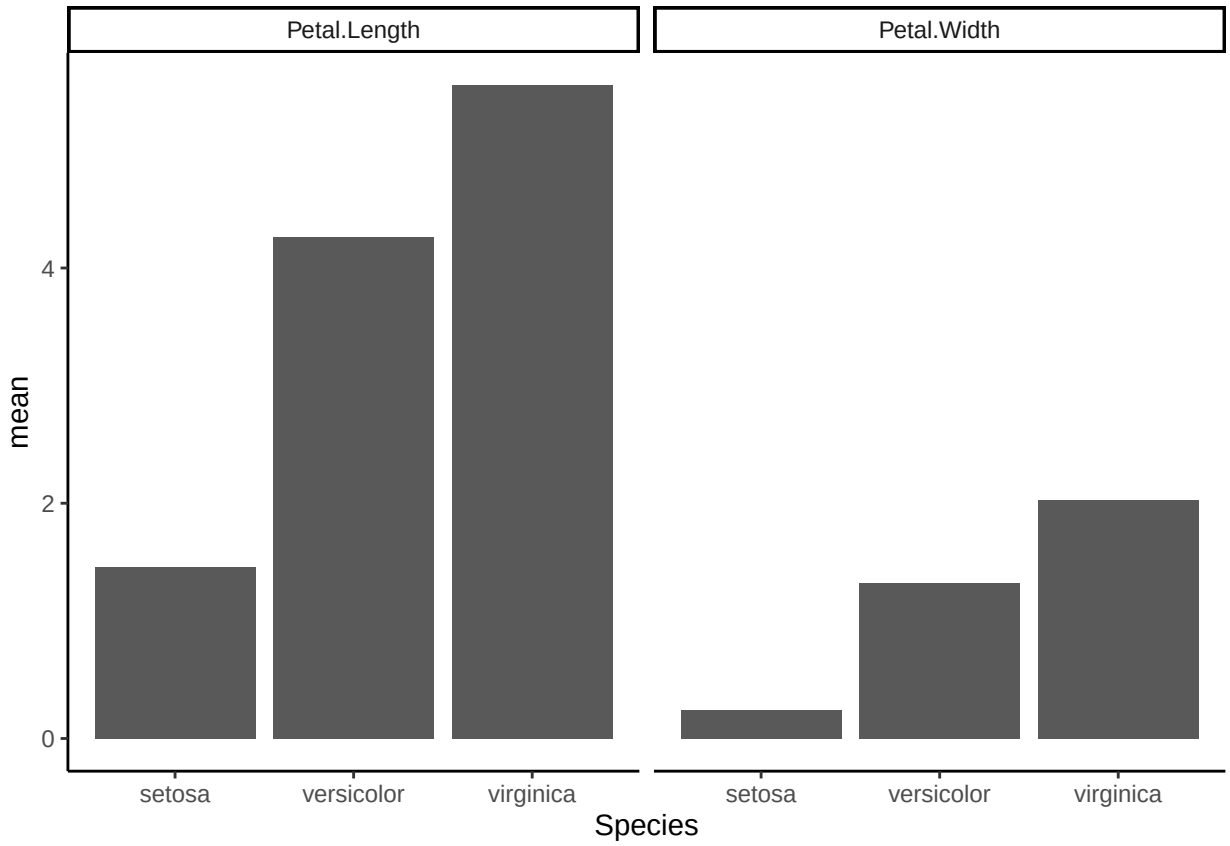
```
iris %>%
  gather(type,measurement,Petal.Length:Petal.Width) %>% #from wide to long format
  dplyr::group_by(Species,type) %>%
  dplyr::summarise(mean=mean(measurement),
            sd=sd(measurement),
            n=dplyr::n())
```

```
## # A tibble: 6 x 5
## # Groups:   Species [3]
##   Species    type          mean    sd     n
##   <fct>      <chr>        <dbl> <dbl> <int>
## 1 setosa     Petal.Length 1.46  0.174    50
## 2 setosa     Petal.Width  0.246 0.105    50
## 3 versicolor Petal.Length 4.26  0.470    50
## 4 versicolor Petal.Width  1.33  0.198    50
## 5 virginica  Petal.Length 5.55  0.552    50
## 6 virginica  Petal.Width  2.03  0.275    50
```

`tidyr` makes the code much easier to read and much shorter. We could also have made a loop that subsets the dataset for each group and calculate each of these and them stores them back into objects, but this takes up a lot more memory and processing power, and it slows down your computer a lot for large datasets. `tidyr` is a nice way to do these operations faster.

You can also make a plot directly from the output of `tidyr` without saving the output of tidyr into a new table, so it doesn't store any object in R's memory (if you have memory issues that can be very useful):

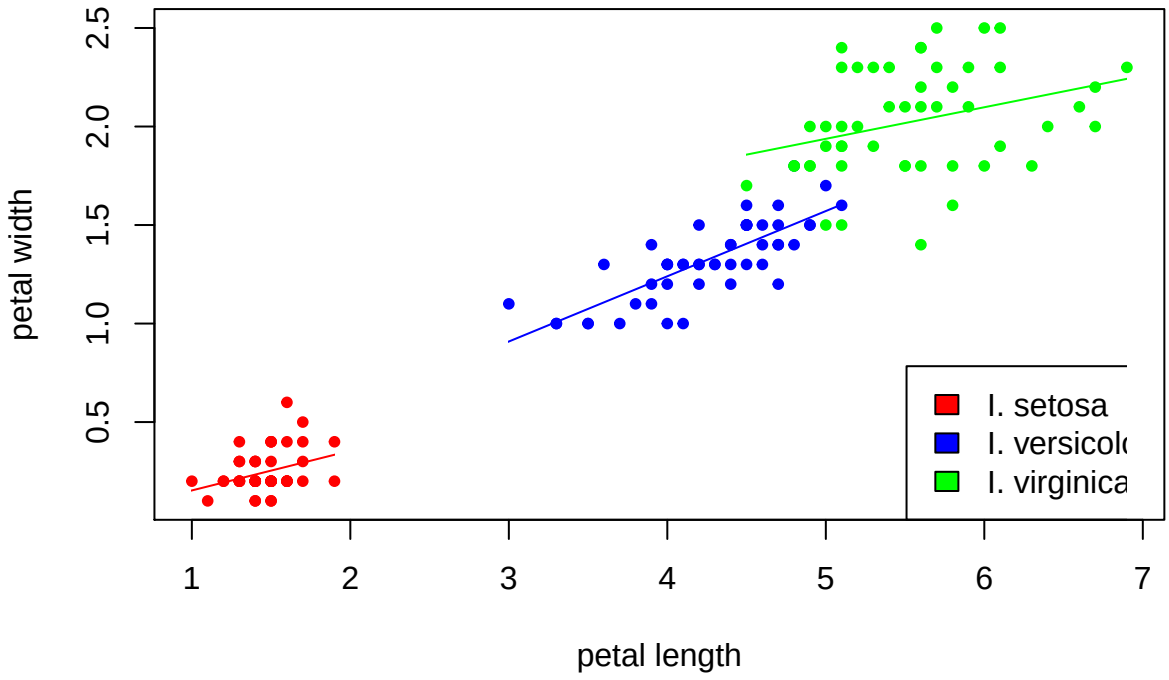
```
iris %>%
  gather(type,measurement,Petal.Length:Petal.Width) %>% #from wide to long format
  dplyr::group_by(Species,type) %>%
  dplyr::summarise(mean=mean(measurement),
            sd=sd(measurement),
            n=dplyr::n()) %>%
  ggplot(aes(x=Species,y=mean))+
  geom_bar(stat = "identity")+
  facet_wrap(~type)+
  theme_classic()
```



5) Do petal lengths vary with petal widths similarly between species?

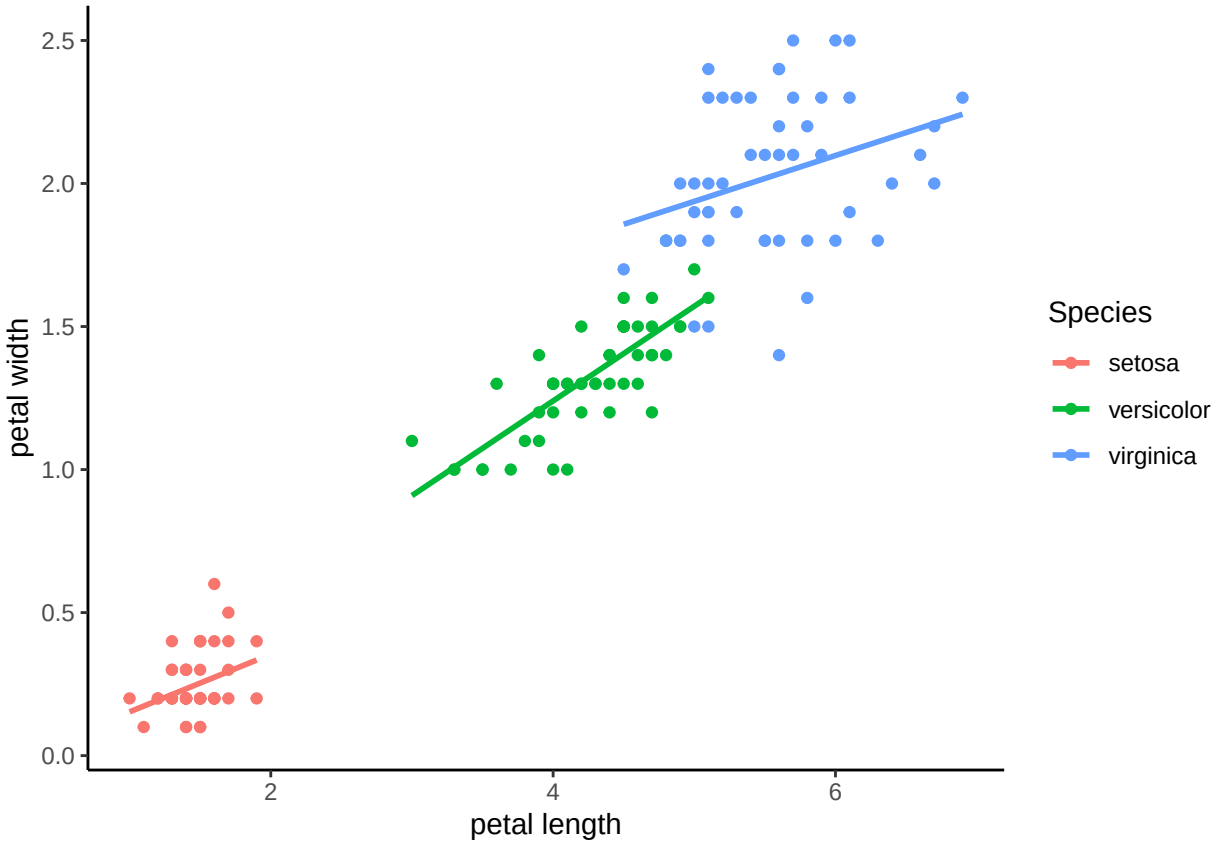
We can make this figure in base R, but this code is atrocious, and imagine if we had more grouping variables, we would have to write it all again.

```
plot(Petal.Width~Petal.Length,data=iris,pch=20,
     xlab='petal length',
     ylab='petal width',
     col=c('red','blue','green')[iris$Species])
clip(min(iris$Petal.Length[iris$Species=="setosa"]),
     max(iris$Petal.Length[iris$Species=="setosa"]), -100, +100)
abline(lm(Petal.Width~Petal.Length, data=iris[iris$Species=="setosa",]),col="red")
clip(min(iris$Petal.Length[iris$Species=="versicolor"]),
     max(iris$Petal.Length[iris$Species=="versicolor"]), -100, +100)
abline(lm(Petal.Width~Petal.Length, data=iris[iris$Species=="versicolor",]), col="blue")
clip(min(iris$Petal.Length[iris$Species=="virginica"]),
     max(iris$Petal.Length[iris$Species=="virginica"]), -100, +100)
abline(lm(Petal.Width~Petal.Length, data=iris[iris$Species=="virginica",]), col="green")
legend("bottomright",legend=c("I. setosa", "I. versicolor", "I. virginica"),
       fill=c('red','blue','green'))
```



With ggplot this takes 6 lines of code:

```
ggplot(iris,aes(x=Petal.Length,y=Petal.Width,colour=Species, group=Species))+
  geom_point()+
  geom_smooth(method = "lm", fill = NA)+
  xlab("petal length")+
  ylab("petal width")+
  theme_classic()
```



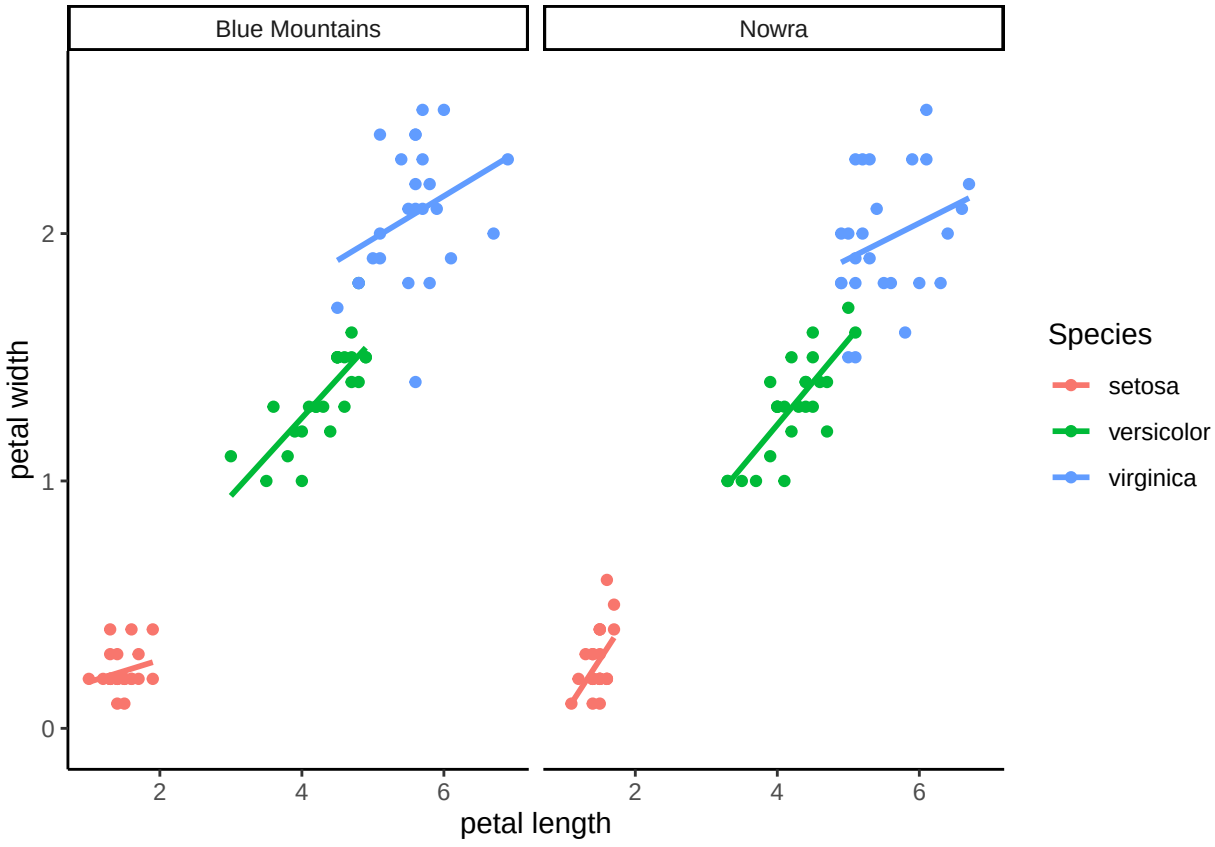
It is also much easier to make this figure if we had two different locations. We can fabricate another column with two locations to demonstrate this:

```
iris<-iris %>%
  dplyr::mutate(location=rep(c("Blue Mountains","Nowra"),dim(iris)[1]/2)) #this code creates a new colu

iris %>%
  head()
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species       location
## 1          5.1         3.5          1.4         0.2  setosa Blue Mountains
## 2          4.9         3.0          1.4         0.2  setosa          Nowra
## 3          4.7         3.2          1.3         0.2  setosa Blue Mountains
## 4          4.6         3.1          1.5         0.2  setosa          Nowra
## 5          5.0         3.6          1.4         0.2  setosa Blue Mountains
## 6          5.4         3.9          1.7         0.4  setosa          Nowra
```

```
ggplot(iris,aes(x=Petal.Length, y=Petal.Width, colour=Species, group=Species))+
  geom_point()+
  geom_smooth(method = "lm", fill = NA)+
  xlab("petal length")+
  ylab("petal width")+
  facet_wrap(~location)+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        strip.background = element_rect(colour="white", fill="white"))+
  theme_classic()
```



And again we can summarise the values for each group, like in question I. This time we will save it in the object "results" and export this result as a csv file, to include in our manuscript.

```
results<-iris %>%
  gather(type,measurement,Petal.Length:Petal.Width) %>%
  dplyr::group_by(Species,type,location) %>%
  dplyr::summarise(mean=mean(measurement),
            sd=sd(measurement),
            n=dplyr::n())

write.csv(results,"table 1 - mean - sd - n.csv",row.names=F) #the file "table 1 - mean - sd - n.csv" ha
```

6) Making panels with cowplot

We want a panel with:

- the two boxplots side by side
- the scatter plot under that
- one common legend
- panel names A B C

First we need to store our plots into objects, and the two first boxplots shouldn't have a legend.

Of course in a real script we wouldn't rewrite that, but here we do just to show how compact the final code is. The *whole* code is below:

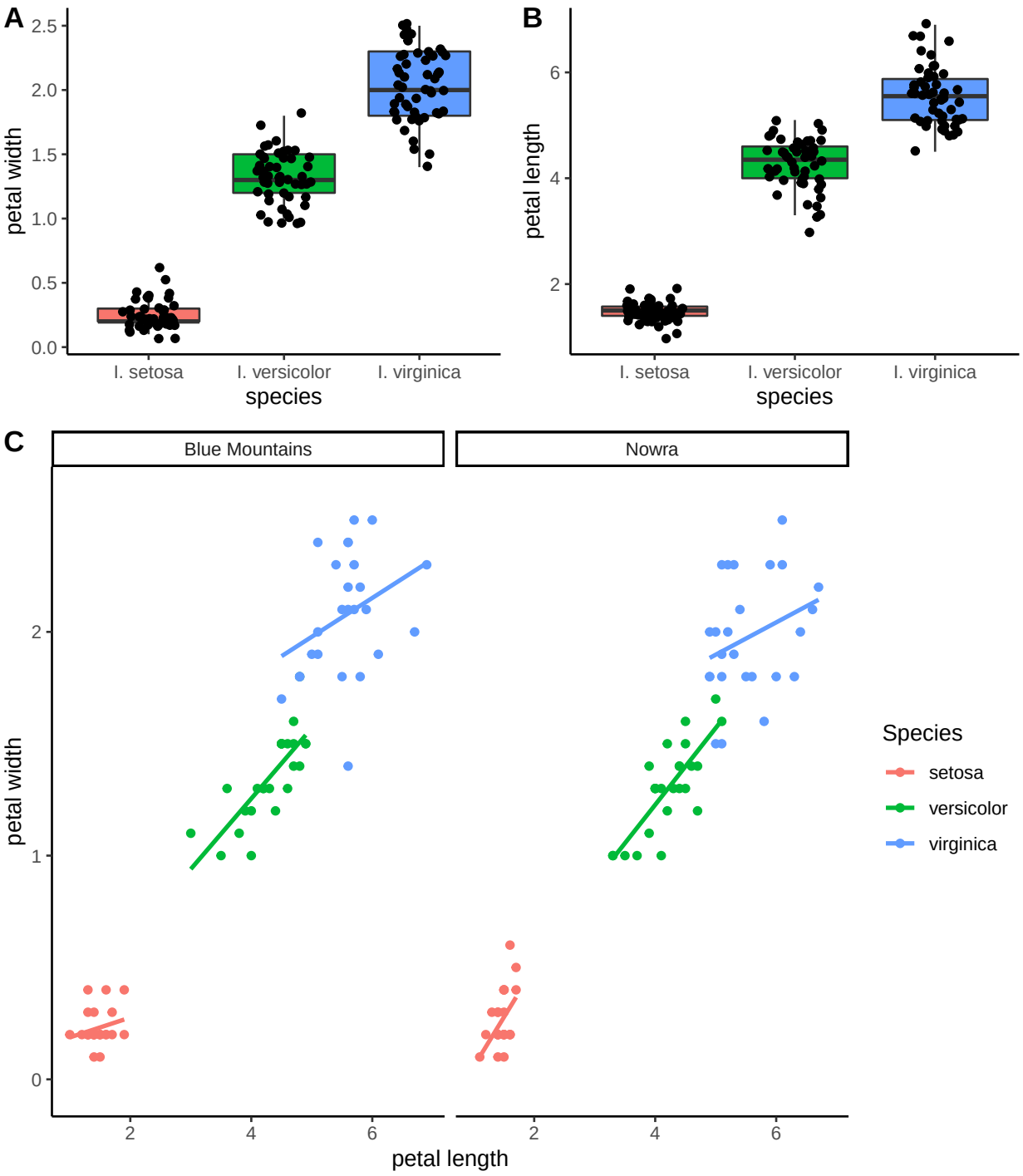
```
#our first boxplot
boxplot.width <- ggplot(iris,aes(x=Species, y=Petal.Width, fill=Species))+
  geom_boxplot(outlier.shape = NA)+
  geom_jitter(width = 0.2)+
  xlab("species")+
  ylab("petal width")+
  scale_x_discrete(labels=c("I. setosa", "I. versicolor", "I. virginica"))+
  theme_classic()

#our second boxplot
boxplot.length <- ggplot(iris,aes(x=Species, y=Petal.Length, fill=Species))+
  geom_boxplot(outlier.shape = NA)+
  geom_jitter(width = 0.2)+
  xlab("species")+
  ylab("petal length")+
  scale_x_discrete(labels=c("I. setosa", "I. versicolor", "I. virginica"))+
  theme_classic()

#our scatterplot
scatterplot <- ggplot(iris,aes(x=Petal.Length,y=Petal.Width,colour=Species, group=Species))+
  geom_point()+
  geom_smooth(method = "lm", fill = NA)+  # if you remove the fill = NA, you get a 95% confidence inter
  xlab("petal length")+
  ylab("petal width")+
  facet_wrap(~location)+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        strip.background = element_rect(colour="white", fill="white"))+
  theme_classic()

#now we create a first panel, with the two boxplots
top<-cowplot::plot_grid(boxplot.width + theme(legend.position="none"),
                boxplot.length + theme(legend.position="none"),
                labels = c('A', 'B'),
                align = 'h')

#here we add the scatterplot to the panel
cowplot::plot_grid(top,scatterplot,labels=c('','C'),ncol=1, rel_heights=c(1,1.8))
```




```
# we'll save this as a high quality compressed TIFF image that any journal will accept:
ggsave("Figures/Figure 1.tiff",
       compression="lzw", #make sure to always include this, or your file will be heavy
       width=220,height=200,units="mm")
```

Additionaly, we should report the version of R and the packages attached, and cite them. I usually copy the output of this as a comment at the end of my scripts.

```
sessionInfo()
```

```
## R version 4.0.0 (2020-04-24)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Catalina 10.15.4
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_AU.UTF-8/en_AU.UTF-8/en_AU.UTF-8/C/en_AU.UTF-8/en_AU.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] ggplot2_3.3.0 tidyr_1.1.0   formatR_1.7
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.4.6     pillar_1.4.4     compiler_4.0.0   tools_4.0.0
##  [5] digest_0.6.25    lattice_0.20-41  evaluate_0.14    lifecycle_0.2.0
##  [9] tibble_3.0.1     gtable_0.3.0     nlme_3.1-147     mgcv_1.8-31
## [13] pkgconfig_2.0.3  rlang_0.4.6      Matrix_1.2-18    cli_2.0.2
## [17] yaml_2.2.1       xfun_0.14        withr_2.2.0      dplyr_0.8.5
## [21] stringr_1.4.0    knitr_1.28       vctrs_0.3.0      cowplot_1.0.0
## [25] grid_4.0.0       tidyselect_1.1.0 glue_1.4.1       R6_2.4.1
## [29] fansi_0.4.1      rmarkdown_2.1    purrr_0.3.4      farver_2.0.3
## [33] magrittr_1.5     scales_1.1.1     ellipsis_0.3.1   htmltools_0.4.0
## [37] splines_4.0.0    assertthat_0.2.1 colorspace_1.4-1 labeling_0.3
## [41] utf8_1.1.4       stringi_1.4.6    munsell_0.5.0    crayon_1.3.4
```

```
citation()
```

```
##
## To cite R in publications use:
##
##   R Core Team (2020). R: A language and environment for statistical
##   computing. R Foundation for Statistical Computing, Vienna, Austria.
##   URL https://www.R-project.org/.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {R: A Language and Environment for Statistical Computing},
##     author = {{R Core Team}},
##     organization = {R Foundation for Statistical Computing},
##     address = {Vienna, Austria},
##     year = {2020},
##     url = {https://www.R-project.org/},
##   }
##
## We have invested a lot of time and effort in creating R, please cite it
```

```
## when using it for data analysis. See also 'citation("pkgname")' for
## citing R packages.
```

```
citation("ggplot2")
```

```
##
## To cite ggplot2 in publications, please use:
##
##   H. Wickham. ggplot2: Elegant Graphics for Data Analysis.
##   Springer-Verlag New York, 2016.
##
## A BibTeX entry for LaTeX users is
##
##   @Book{,
##     author = {Hadley Wickham},
##     title = {ggplot2: Elegant Graphics for Data Analysis},
##     publisher = {Springer-Verlag New York},
##     year = {2016},
##     isbn = {978-3-319-24277-4},
##     url = {https://ggplot2.tidyverse.org},
##   }
```

```
citation("cowplot")
```

```
##
## To cite package 'cowplot' in publications use:
##
##   Claus O. Wilke (2019). cowplot: Streamlined Plot Theme and Plot
##   Annotations for 'ggplot2'. R package version 1.0.0.
##   https://CRAN.R-project.org/package=cowplot
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {cowplot: Streamlined Plot Theme and Plot Annotations for 'ggplot2'},
##     author = {Claus O. Wilke},
##     year = {2019},
##     note = {R package version 1.0.0},
##     url = {https://CRAN.R-project.org/package=cowplot},
##   }
```

```
citation("tidyr")
```

```
##
## To cite package 'tidyr' in publications use:
##
##   Hadley Wickham and Lionel Henry (2020). tidyr: Tidy Messy Data. R
##   package version 1.1.0. https://CRAN.R-project.org/package=tidyr
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {tidyr: Tidy Messy Data},
##     author = {Hadley Wickham and Lionel Henry},
##     year = {2020},
##     note = {R package version 1.1.0},
```

```
##     url = {https://CRAN.R-project.org/package=tidyr},
##   }
```

```
citation("dplyr")
```

```
##
## To cite package 'dplyr' in publications use:
##
##   Hadley Wickham, Romain François, Lionel Henry and Kirill Müller
##   (2020). dplyr: A Grammar of Data Manipulation. R package version
##   0.8.5. https://CRAN.R-project.org/package=dplyr
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {dplyr: A Grammar of Data Manipulation},
##     author = {Hadley Wickham and Romain François and Lionel Henry and Kirill Müller},
##     year = {2020},
##     note = {R package version 0.8.5},
##     url = {https://CRAN.R-project.org/package=dplyr},
##   }
```

To share your whole code and allow your results to be reproduced by anybody at anytime, you can save this output and share it online

```
dput(iris)
```

```
## structure(list(Sepal.Length = c(5.1, 4.9, 4.7, 4.6, 5, 5.4, 4.6,
## 5, 4.4, 4.9, 5.4, 4.8, 4.8, 4.3, 5.8, 5.7, 5.4, 5.1, 5.7, 5.1,
## 5.4, 5.1, 4.6, 5.1, 4.8, 5, 5, 5.2, 5.2, 4.7, 4.8, 5.4, 5.2,
## 5.5, 4.9, 5, 5.5, 4.9, 4.4, 5.1, 5, 4.5, 4.4, 5, 5.1, 4.8, 5.1,
## 4.6, 5.3, 5, 7, 6.4, 6.9, 5.5, 6.5, 5.7, 6.3, 4.9, 6.6, 5.2,
## 5, 5.9, 6, 6.1, 5.6, 6.7, 5.6, 5.8, 6.2, 5.6, 5.9, 6.1, 6.3,
## 6.1, 6.4, 6.6, 6.8, 6.7, 6, 5.7, 5.5, 5.5, 5.8, 6, 5.4, 6, 6.7,
## 6.3, 5.6, 5.5, 5.5, 6.1, 5.8, 5, 5.6, 5.7, 5.7, 6.2, 5.1, 5.7,
## 6.3, 5.8, 7.1, 6.3, 6.5, 7.6, 4.9, 7.3, 6.7, 7.2, 6.5, 6.4, 6.8,
## 5.7, 5.8, 6.4, 6.5, 7.7, 7.7, 6, 6.9, 5.6, 7.7, 6.3, 6.7, 7.2,
## 6.2, 6.1, 6.4, 7.2, 7.4, 7.9, 6.4, 6.3, 6.1, 7.7, 6.3, 6.4, 6,
## 6.9, 6.7, 6.9, 5.8, 6.8, 6.7, 6.7, 6.3, 6.5, 6.2, 5.9), Sepal.Width = c(3.5,
## 3, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4, 2.9, 3.1, 3.7, 3.4, 3, 3, 4,
## 4.4, 3.9, 3.5, 3.8, 3.8, 3.4, 3.7, 3.6, 3.3, 3.4, 3, 3.4, 3.5,
## 3.4, 3.2, 3.1, 3.4, 4.1, 4.2, 3.1, 3.2, 3.5, 3.6, 3, 3.4, 3.5,
## 2.3, 3.2, 3.5, 3.8, 3, 3.8, 3.2, 3.7, 3.3, 3.2, 3.2, 3.1, 2.3,
## 2.8, 2.8, 3.3, 2.4, 2.9, 2.7, 2, 3, 2.2, 2.9, 2.9, 3.1, 3, 2.7,
## 2.2, 2.5, 3.2, 2.8, 2.5, 2.8, 2.9, 3, 2.8, 3, 2.9, 2.6, 2.4,
## 2.4, 2.7, 2.7, 3, 3.4, 3.1, 2.3, 3, 2.5, 2.6, 3, 2.6, 2.3, 2.7,
## 3, 2.9, 2.9, 2.5, 2.8, 3.3, 2.7, 3, 2.9, 3, 3, 2.5, 2.9, 2.5,
## 3.6, 3.2, 2.7, 3, 2.5, 2.8, 3.2, 3, 3.8, 2.6, 2.2, 3.2, 2.8,
## 2.8, 2.7, 3.3, 3.2, 2.8, 3, 2.8, 3, 2.8, 3.8, 2.8, 2.8, 2.6,
## 3, 3.4, 3.1, 3, 3.1, 3.1, 3.1, 2.7, 3.2, 3.3, 3, 2.5, 3, 3.4,
## 3), Petal.Length = c(1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5,
## 1.4, 1.5, 1.5, 1.6, 1.4, 1.1, 1.2, 1.5, 1.3, 1.4, 1.7, 1.5, 1.7,
## 1.5, 1, 1.7, 1.9, 1.6, 1.6, 1.5, 1.4, 1.6, 1.6, 1.5, 1.5, 1.4,
## 1.5, 1.2, 1.3, 1.4, 1.3, 1.5, 1.3, 1.3, 1.3, 1.6, 1.9, 1.4, 1.6,
## 1.4, 1.5, 1.4, 4.7, 4.5, 4.9, 4, 4.6, 4.5, 4.7, 3.3, 4.6, 3.9,
## 3.5, 4.2, 4, 4.7, 3.6, 4.4, 4.5, 4.1, 4.5, 3.9, 4.8, 4, 4.9,
```

```
## 4.7, 4.3, 4.4, 4.8, 5, 4.5, 3.5, 3.8, 3.7, 3.9, 5.1, 4.5, 4.5,
## 4.7, 4.4, 4.1, 4, 4.4, 4.6, 4, 3.3, 4.2, 4.2, 4.2, 4.3, 3, 4.1,
## 6, 5.1, 5.9, 5.6, 5.8, 6.6, 4.5, 6.3, 5.8, 6.1, 5.1, 5.3, 5.5,
## 5, 5.1, 5.3, 5.5, 6.7, 6.9, 5, 5.7, 4.9, 6.7, 4.9, 5.7, 6, 4.8,
## 4.9, 5.6, 5.8, 6.1, 6.4, 5.6, 5.1, 5.6, 6.1, 5.6, 5.5, 4.8, 5.4,
## 5.6, 5.1, 5.1, 5.9, 5.7, 5.2, 5, 5.2, 5.4, 5.1), Petal.Width = c(0.2,
## 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2, 0.2, 0.1, 0.1,
## 0.2, 0.4, 0.4, 0.3, 0.3, 0.3, 0.2, 0.4, 0.2, 0.5, 0.2, 0.2, 0.4,
## 0.2, 0.2, 0.2, 0.2, 0.4, 0.1, 0.2, 0.2, 0.2, 0.2, 0.1, 0.2, 0.2,
## 0.3, 0.3, 0.2, 0.6, 0.4, 0.3, 0.2, 0.2, 0.2, 0.2, 1.4, 1.5, 1.5,
## 1.3, 1.5, 1.3, 1.6, 1, 1.3, 1.4, 1, 1.5, 1, 1.4, 1.3, 1.4, 1.5,
## 1, 1.5, 1.1, 1.8, 1.3, 1.5, 1.2, 1.3, 1.4, 1.4, 1.7, 1.5, 1,
## 1.1, 1, 1.2, 1.6, 1.5, 1.6, 1.5, 1.3, 1.3, 1.3, 1.2, 1.4, 1.2,
## 1, 1.3, 1.2, 1.3, 1.3, 1.1, 1.3, 2.5, 1.9, 2.1, 1.8, 2.2, 2.1,
## 1.7, 1.8, 1.8, 2.5, 2, 1.9, 2.1, 2, 2.4, 2.3, 1.8, 2.2, 2.3,
## 1.5, 2.3, 2, 2, 1.8, 2.1, 1.8, 1.8, 1.8, 2.1, 1.6, 1.9, 2, 2.2,
## 1.5, 1.4, 2.3, 2.4, 1.8, 1.8, 2.1, 2.4, 2.3, 1.9, 2.3, 2.5, 2.3,
## 1.9, 2, 2.3, 1.8), Species = structure(c(1L, 1L, 1L, 1L, 1L,
## 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L,
## 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L,
## 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 2L, 2L, 2L,
## 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L,
## 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L,
## 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 3L,
## 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L,
## 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L,
## 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L,
## 3L), .Label = c("setosa", "versicolor", "virginica"), class = "factor"),
##     location = c("Blue Mountains", "Nowra", "Blue Mountains",
##     "Nowra", "Blue Mountains", "Nowra", "Blue Mountains", "Nowra",
##     "Blue Mountains", "Nowra", "Blue Mountains", "Nowra", "Blue Mountains",
##     "Nowra", "Blue Mountains", "Nowra", "Blue Mountains", "Nowra",
##     "Blue Mountains", "Nowra", "Blue Mountains", "Nowra", "Blue Mountains",
##     "Nowra", "Blue Mountains", "Nowra", "Blue Mountains", "Nowra",
##     "Blue Mountains", "Nowra", "Blue Mountains", "Nowra", "Blue Mountains",
##     "Nowra", "Blue Mountains", "Nowra", "Blue Mountains", "Nowra",
##     "Blue Mountains", "Nowra", "Blue Mountains", "Nowra", "Blue Mountains",
##     "Nowra", "Blue Mountains", "Nowra", "Blue Mountains", "Nowra",
##     "Blue Mountains", "Nowra", "Blue Mountains", "Nowra", "Blue Mountains",
##     "Nowra", "Blue Mountains", "Nowra", "Blue Mountains", "Nowra",
##     "Blue Mountains", "Nowra", "Blue Mountains", "Nowra", "Blue Mountains",
##     "Nowra", "Blue Mountains", "Nowra", "Blue Mountains", "Nowra",
##     "Blue Mountains", "Nowra", "Blue Mountains", "Nowra", "Blue Mountains",
##     "Nowra", "Blue Mountains", "Nowra", "Blue Mountains", "Nowra",
##     "Blue Mountains", "Nowra", "Blue Mountains", "Nowra", "Blue Mountains",
##     "Nowra", "Blue Mountains", "Nowra", "Blue Mountains", "Nowra",
##     "Blue Mountains", "Nowra", "Blue Mountains", "Nowra", "Blue Mountains",
##     "Nowra", "Blue Mountains", "Nowra", "Blue Mountains", "Nowra",
```

```
##     "Blue Mountains", "Nowra", "Blue Mountains", "Nowra", "Blue Mountains",
##     "Nowra", "Blue Mountains", "Nowra", "Blue Mountains", "Nowra",
##     "Blue Mountains", "Nowra", "Blue Mountains", "Nowra", "Blue Mountains",
##     "Nowra", "Blue Mountains", "Nowra", "Blue Mountains", "Nowra",
##     "Blue Mountains", "Nowra")), class = "data.frame", row.names = c(NA,
## -150L))
```