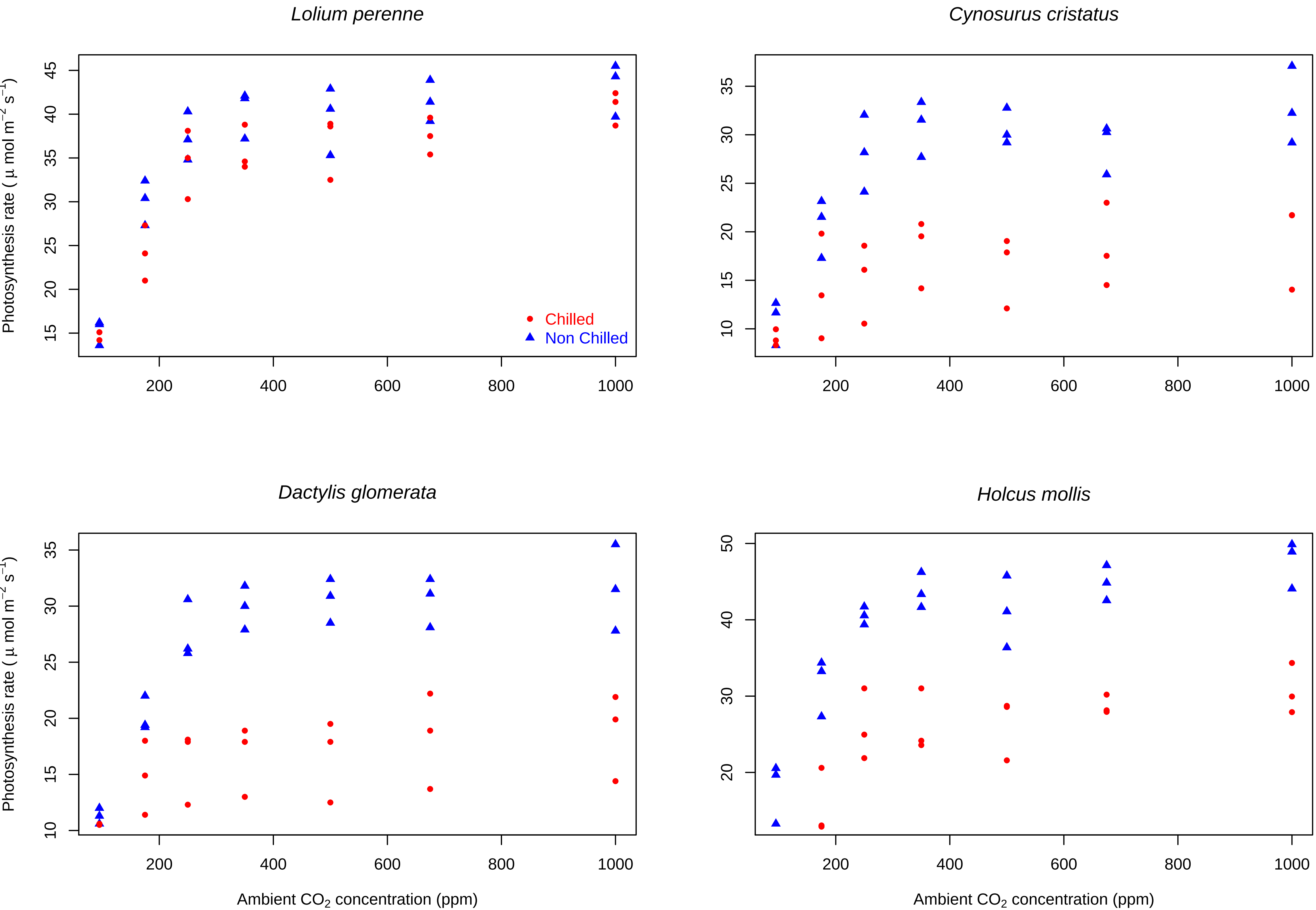
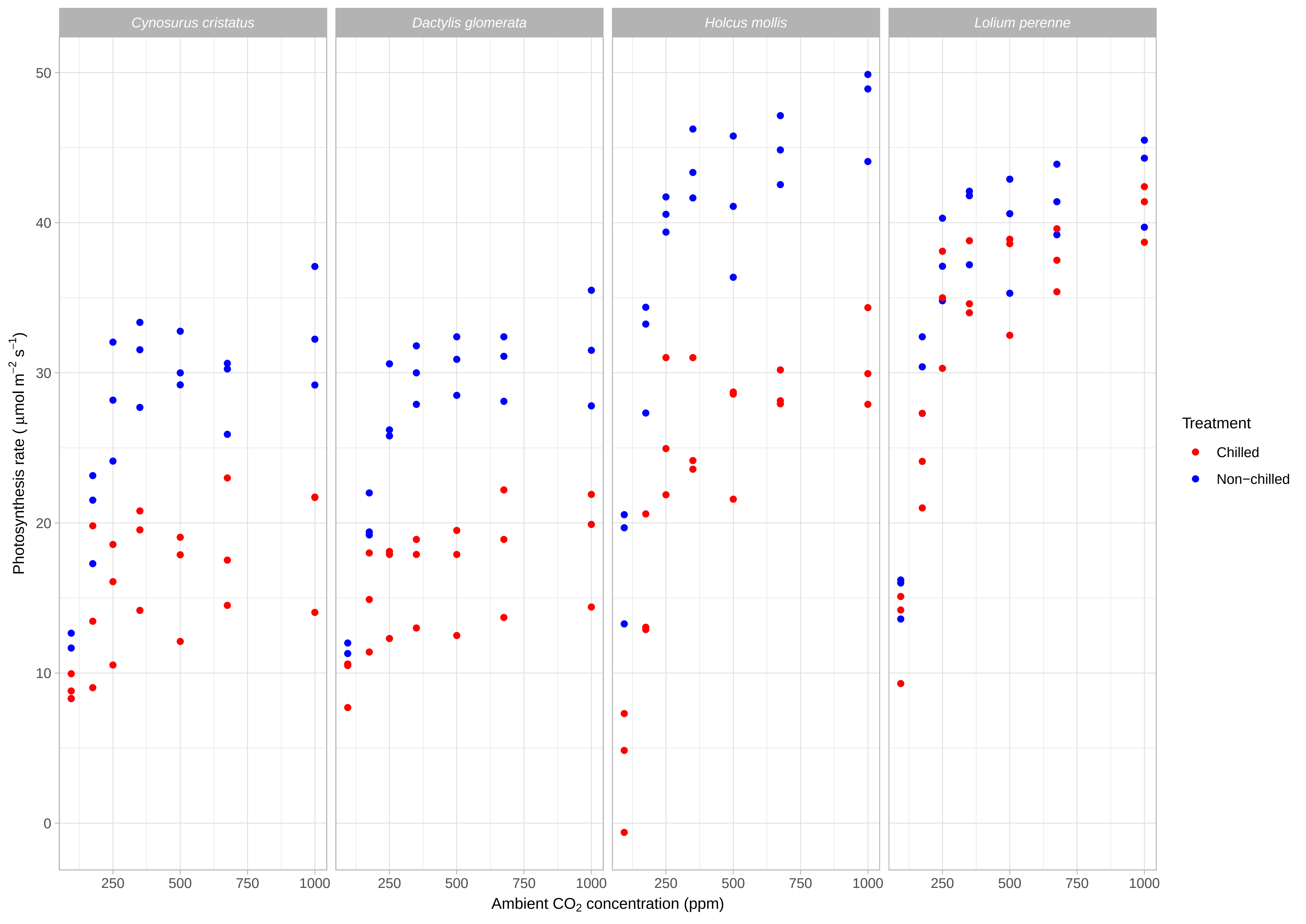
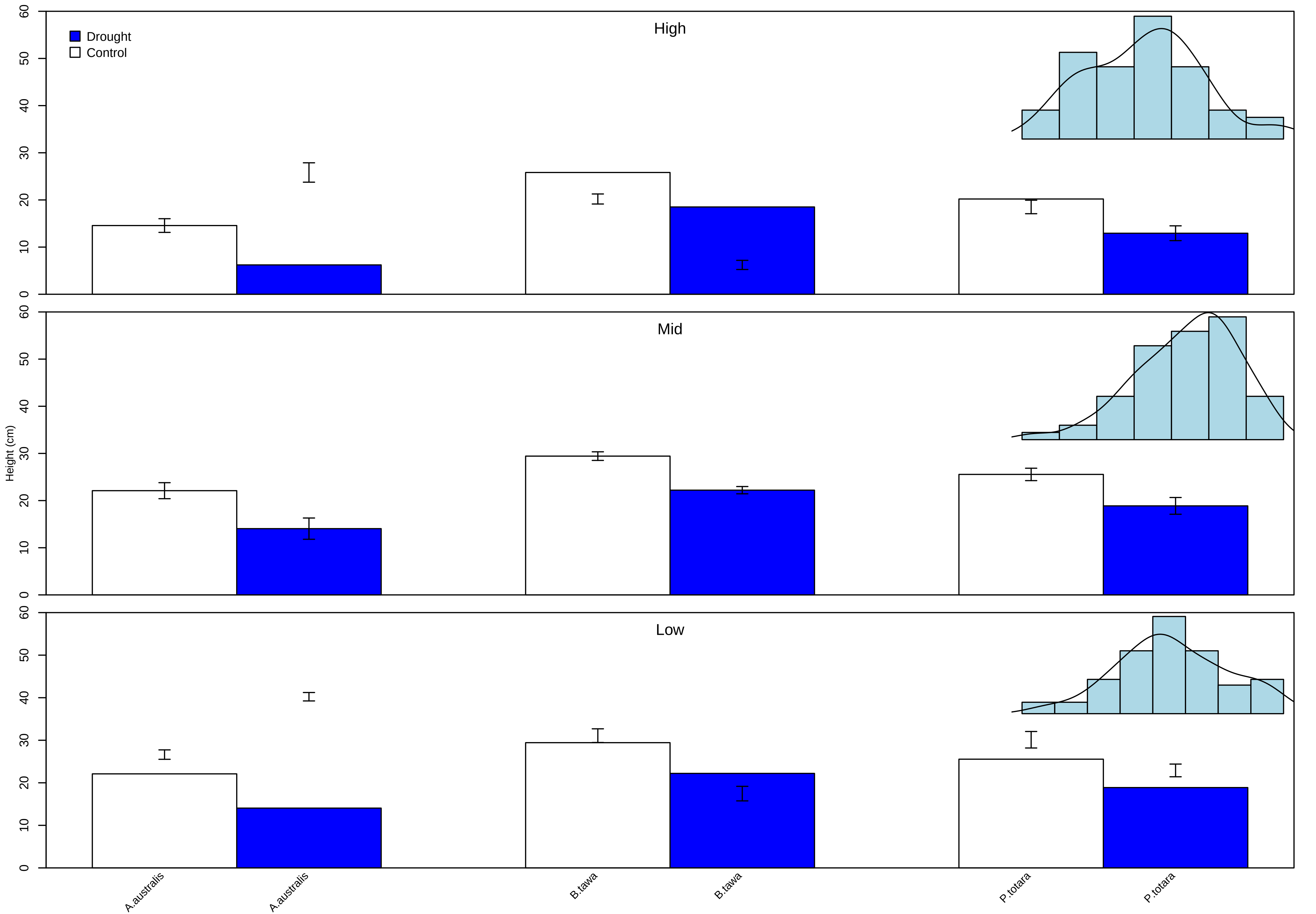
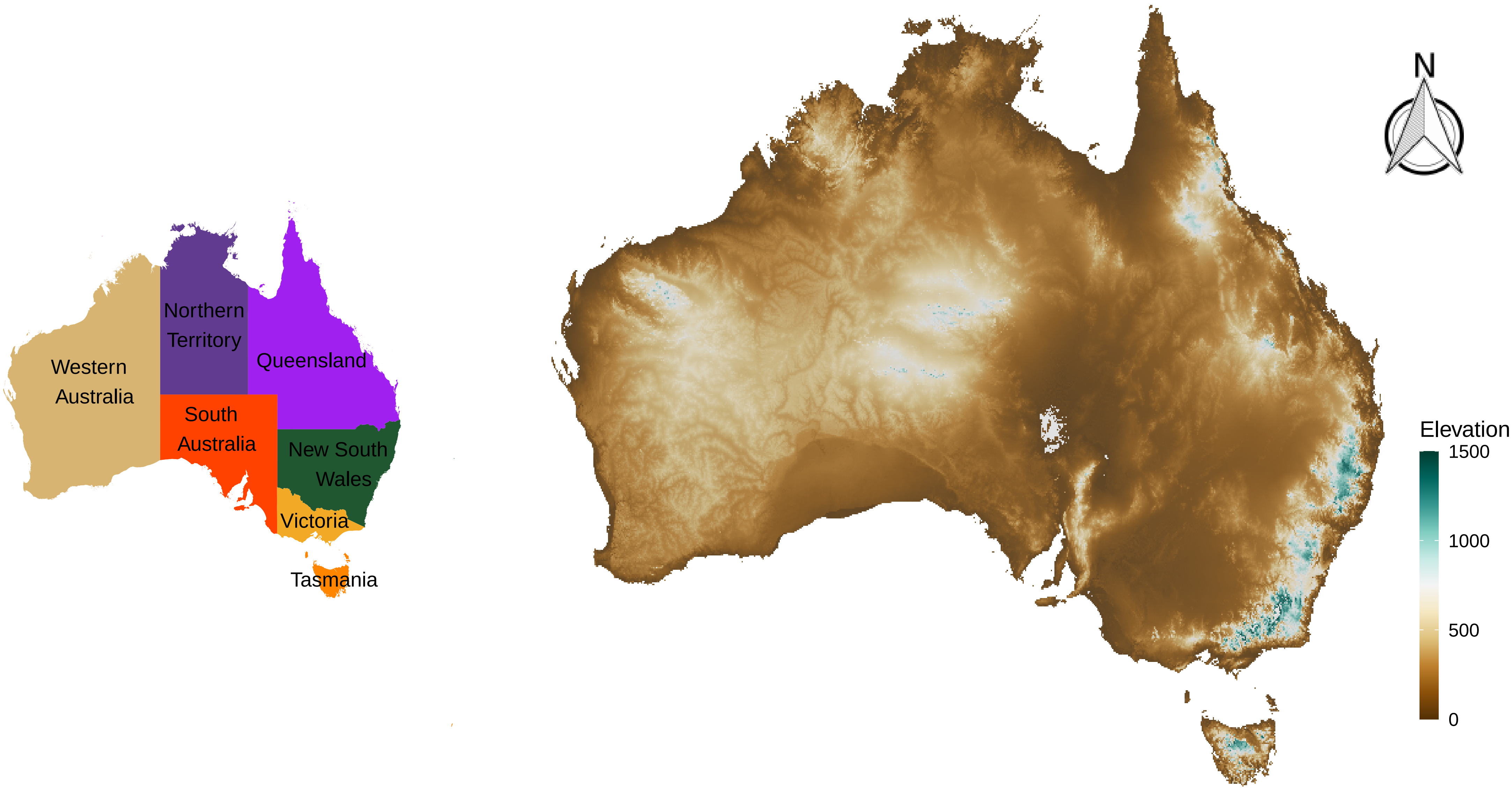

Lolium perenne
Cynosurus cristatus
Dactylis glomerata
Holcus mollis
Photosynthesis rate (μ mol m⁻² s⁻¹)
Ambient CO2 concentration (ppm)
Chilled
Non Chilled
200
400
600
800
1000

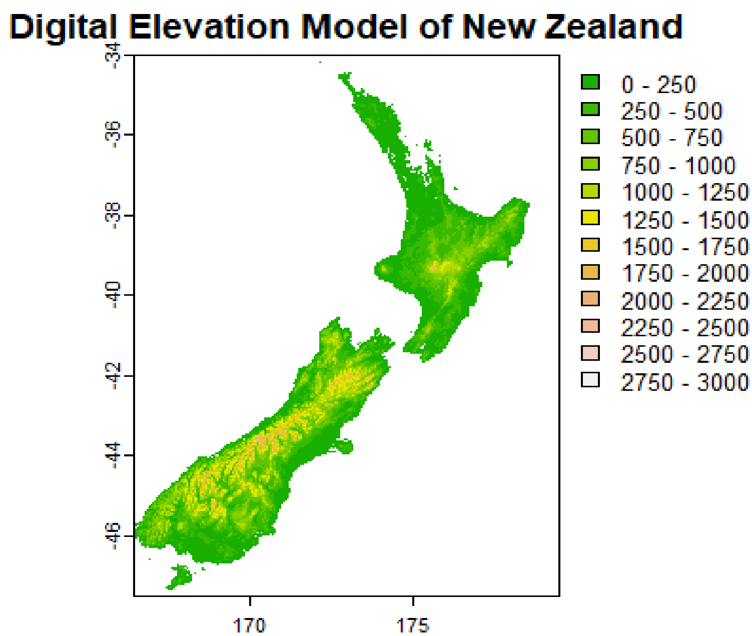

Cynosurus cristatus
Dactylis glomerata
Holcus mollis
Lolium perenne
Photosynthesis rate (µmol m^{-2} s^{-1})
0
10
20
30
40
50
250
500
750
1000
Ambient CO$_2$ concentration (ppm)
Treatment
Chilled
Non−chilled

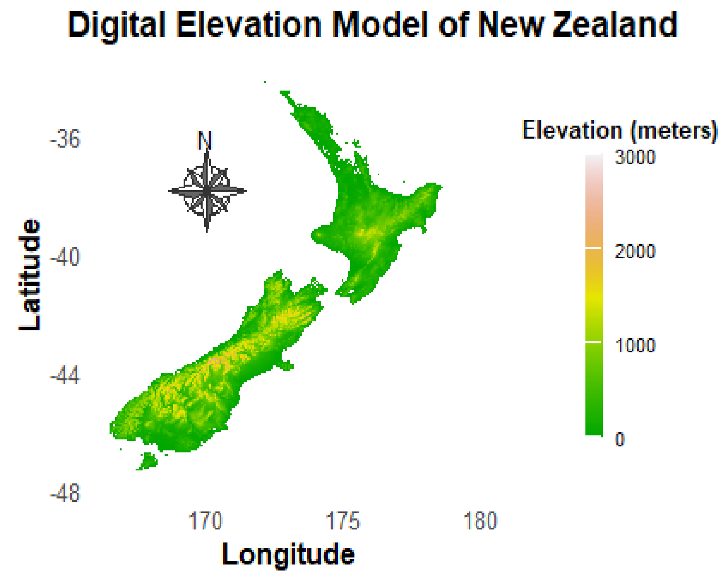


High
Drought
Control
Mid
Low
Height (cm)
0
10
20
30
40
50
60
A.australis
A.australis
B.tawa
B.tawa
P.totara
P.totara



Northern Territory
Western Australia
Queensland
South Australia
New South Wales
Victoria
Tasmania
N
Elevation
1500
1000
500
0


Digital Elevation Model of New Zealand
-34
-36
-38
-40
-42
-44
-46
170
175
0 - 250
250 - 500
500 - 750
750 - 1000
1000 - 1250
1250 - 1500
1500 - 1750
1750 - 2000
2000 - 2250
2250 - 2500
2500 - 2750
2750 - 3000


Digital Elevation Model of New Zealand
N
Elevation (meters)
3000
2000
1000
0
-36
-40
-44
-48
Latitude
170
175
180
Longitude

Graphs R Us - Data Visualisation with R Project

Pawaneet Kaur

Setting up work directory:

getwd () Indicates the current working directory without no arguments by returning a NULL or character string. # This is important for debugging, larger programs to reiterate via the directory contents.

On Windows the returned path use "/" between directory levels as a separator unless in a root directory (of a share or drive on Windows)

setwd() function changes and sets the existing working director navigated via getwd function.

chilling_sensitivity.csv

```
cs = read.csv("chilling_sensitivity.csv") # activating the chilling_sensitivity.csv
dataframe; represented as "cs"
```

drought_elevation.csv

```
elv = read.csv("drought_elevation.csv") # activating the drought_elevation.csv data frame
; displayed as "elv"
```

The "read.csv" reads an excel table format files to generate a data frame.

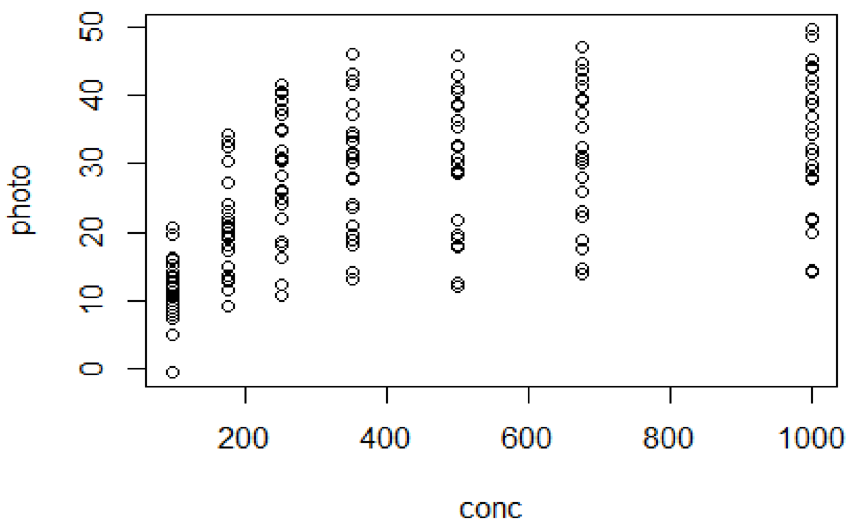
Part 1: Multi-panel scatterplot

1.1 Data Aggregation

```
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

head(cs, n=3) # the x axis independent variable is ambient CO2 concentration (ppm)

##   plant            spec      treat conc photo
## 1     1 Lolium perenne nonchilled   95  16.0
## 2     1 Lolium perenne nonchilled  175  30.4
## 3     1 Lolium perenne nonchilled  250  34.8

#the  while the y axis dependent variable is the  photosynthetic rate (µmol m-2 s -1)
```

Scatterplot of the raw data

```
plot( photo ~ conc, data = cs)
```



Standard error function

```
se <- function(x, na.rm = FALSE)# a.rm = FALSE removes blank cells
{
  if(na.rm == TRUE)
  {
    sqrt(var(x, na.rm = T)/length(na.omit(x)))
  }
  else
  {
    sqrt(var(x)/length(x))
  }
}


pcs <- group_by(.data = cs, treat, spec,conc) %>% summarise(p_avv = mean(photo)
    , pos = mean(photo) + se(photo), neg = mean(photo) - se(photo))

## `summarise()` regrouping output by 'treat', 'spec' (override with `.groups` argument)

head(pcs,n=3)

## # A tibble: 3 x 6
## # Groups:   treat, spec [1]
##   treat   spec                 conc p_avv   pos   neg
##   <chr>   <chr>               <int> <dbl> <dbl> <dbl>
## 1 chilled Cynosurus cristatus    95  9.02  9.51  8.54
## 2 chilled Cynosurus cristatus   175 14.1  17.2  11.0
## 3 chilled Cynosurus cristatus   250 15.1  17.4  12.7
```

calculation without concentration

```
pcsnc <- group_by(.data = cs, treat, spec) %>% summarise(p_avvnc = mean(photo)
, pos = mean(photo) + se(photo), neg = mean(photo) - se(photo))
```

Wide formatting our photosynthesis rate data

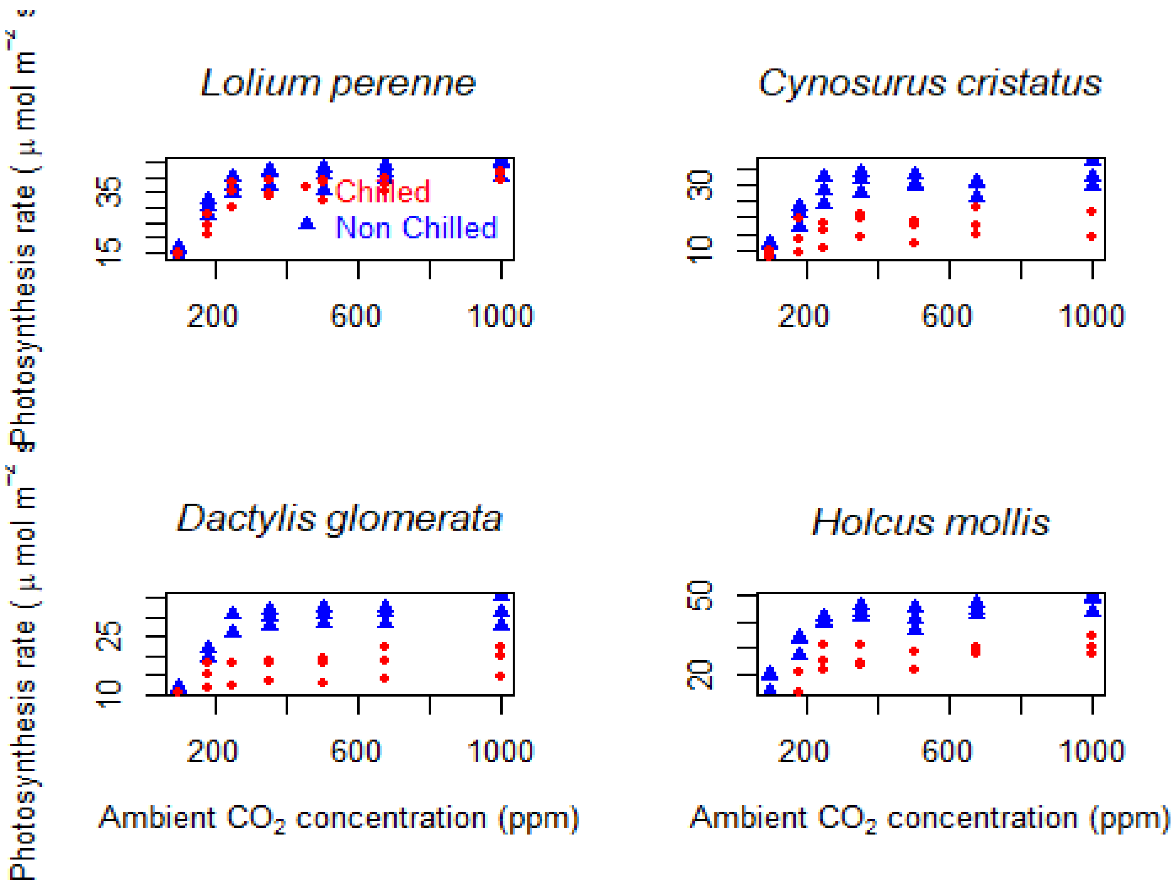
```
library(reshape2) # required to run acast function
wides= acast(data = pcsnc, formula = treat ~ spec, value.var = "p_avvnc")
wides
```

1.2 Multi-panel scatterplot in a traditional graphics system

I have not used the dplyr package.

```
# four-panel Layout

par(mfrow=c(2, 2)) # 2 rows and 2 columns

# Dots/points in blue indicate nonchilled and red as chilled

##### species #####

## Lolium perenne

plot(cs$photo[cs$spec=="Lolium perenne"][cs$treat=="nonchilled"] ~
       cs$conc[cs$spec=="Lolium perenne"] [cs$treat=="nonchilled"]
     , col= "blue", pch= 17, ylab= expression(paste("Photosynthesis rate ( ", mu," mol m"^-2, " s"^-1, ")"))
     ,xlab='', main = substitute(paste(italic('Lolium perenne'))))
points(cs$photo[cs$spec=="Lolium perenne"][cs$treat=="chilled"] ~
         cs$conc[cs$spec=="Lolium perenne"][cs$treat=="chilled"]
       , col= "red",pch= 20)

# Legend

l <- legend( "bottomright"
             , inset = c(0,0)
             , cex = 1
             , bty = "n"
             , legend = c("Chilled", "Non Chilled")
             , text.col = c("red", "blue")
             , pt.bg = c("red","blue")
             , pch = c(20,17)
             , col = c("red","blue"))

## Cynosurus cristatus

plot(cs$photo[cs$spec=="Cynosurus cristatus"] [cs$treat=="nonchilled"] ~
       cs$conc[cs$spec=="Cynosurus cristatus"] [cs$treat=="nonchilled"]
     , col= "blue", pch= 17, ylab='',xlab='', main = substitute(paste(italic('Cynosurus cristatus'))))

points(cs$photo[cs$spec=="Cynosurus cristatus"][cs$treat=="chilled"] ~
         cs$conc[cs$spec=="Cynosurus cristatus"][cs$treat=="chilled"]
       , col= "red",pch= 20)

## Dactylis glomerata

plot(cs$photo[cs$spec=="Dactylis glomerata"] [cs$treat=="nonchilled"] ~
       cs$conc[cs$spec=="Dactylis glomerata"] [cs$treat=="nonchilled"]
     , col= "blue", pch= 17, ylab= expression(paste("Photosynthesis rate ( ",  mu," mol m"^-2, " s"^-1, ")"))
     ,xlab= expression(paste("Ambient CO"[2], " concentration (ppm)")), main = substitute(paste(italic('Dacty
```

```
lis glomerata'))))
points(cs$photo[cs$spec=="Dactylis glomerata"] [cs$treat=="chilled"] ~
         cs$conc[cs$spec=="Dactylis glomerata"] [cs$treat=="chilled"]
       , col= "red",pch= 20)

## Holcus mollis

plot(cs$photo[cs$spec=="Holcus mollis"] [cs$treat=="nonchilled"] ~
       cs$conc[cs$spec=="Holcus mollis"] [cs$treat=="nonchilled"]
     , col= "blue", pch= 17, ylab='',xlab= expression(paste("Ambient CO"[2], " concentration (ppm)")), main =
substitute(paste(italic('Holcus mollis'))))
points(cs$photo[cs$spec=="Holcus mollis"] [cs$treat=="chilled"] ~
         cs$conc[cs$spec=="Holcus mollis"] [cs$treat=="chilled"]
       , col= "red",pch= 20)
```



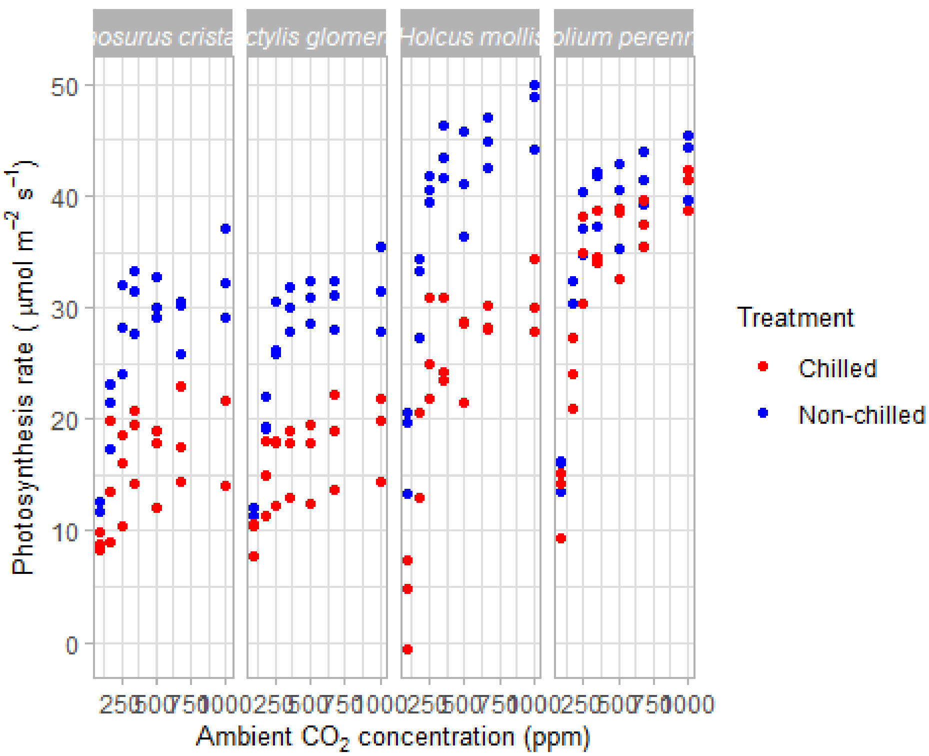
1.3 Multi-panel scatterplot in ggplot2

A multi-panel scatterplot in ggplot2

```
library(ggplot2)
library(dplyr)
```

Photosynthesis rate is higher on non-chilled treatment

```
qplot(conc,photo, data = cs, color= treat, facets=.~ spec) + labs(x = expression(paste("Ambient CO"[2], " con
centration (ppm)"))
    , y= expression(paste("Photosynthesis rate ( ",  mu,"mol m"^-2, " s"^-1, ")"))) +
  theme_light() +
  scale_colour_manual(labels= c("Chilled", "Non-chilled"), values= c("red","blue")) +
  labs(colour="Treatment") +
  theme(strip.text = element_text(face = "italic")) +
  theme(legend.title =element_text(size=10)) +
  theme(axis.title.x = element_text(size=10)) +
  theme(axis.title.y = element_text(size=10))
```



Part 2: Multi-panel barplot with insets

```
# Generating  aggregating and dataframe  data

de =read.csv("drought_elevation.csv")
head(de)

##            spec elevation treat height
## 1 A. australis       low  ctrl   24.9
```

```
## 2 A. australis       low  ctrl   25.3
## 3 A. australis       low  ctrl   25.7
## 4 A. australis       low  ctrl   33.0
## 5 A. australis       low  ctrl   31.2
## 6 A. australis       low  ctrl   25.5
```

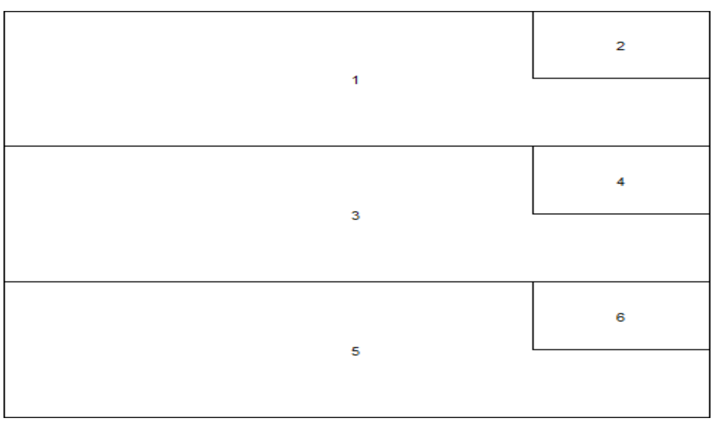
```
str(de)
```

```
## 'data.frame':    180 obs. of  4 variables:
##  $ spec     : chr  "A. australis" "A. australis" "A. australis" "A. australis" ...
##  $ elevation: chr  "low" "low" "low" "low" ...
##  $ treat    : chr  "ctrl" "ctrl" "ctrl" "ctrl" ...
##  $ height   : num  24.9 25.3 25.7 33 31.2 25.5 20.8 28.6 26.8 24.5 ...
```

```
## Creating standard error (se) function

se <- function(x, na.rm = FALSE)
{
  if(na.rm == TRUE)
  {
    sqrt(var(x, na.rm = T)/length(na.omit(x)))
  }
  else
  {
    sqrt(var(x)/length(x))
  }
}
```

Generating the Multi-panel bar plot

```
## multi-panel layout

layout(widths = c(0.9, 0.3),rbind(c(1, 2),
              c(1, 1),
              c(3, 4),
              c(3, 3),
              c(5, 6),
              c(5, 5)))
layout.show(n = 6)
```

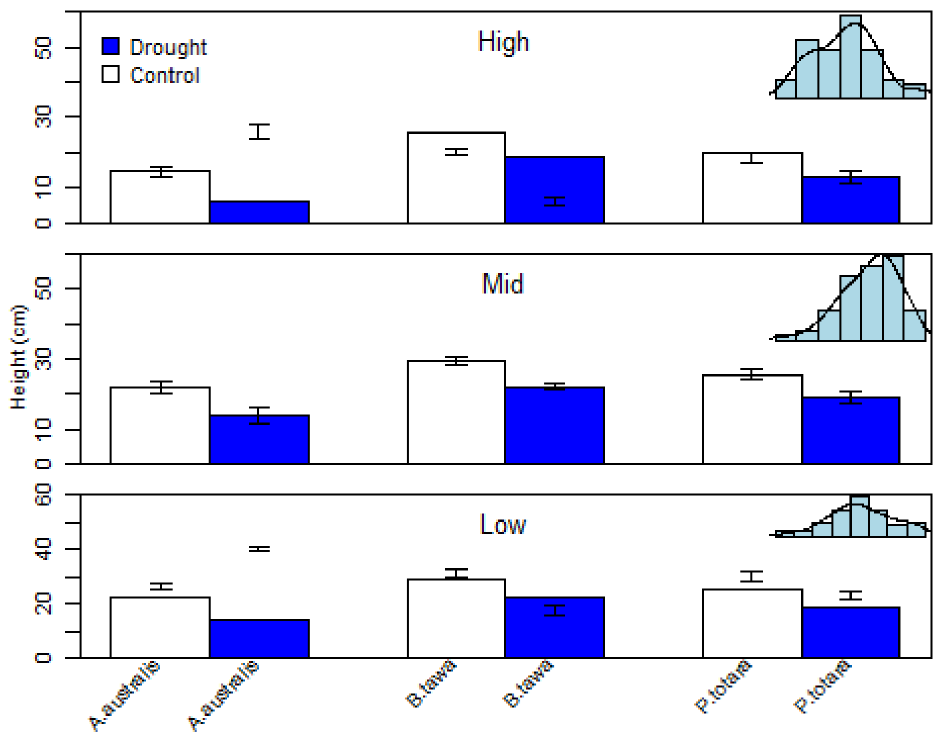



```
oper <- par(oma = c(1,1,1,1),mar = c(1,1,1,1))

## top panel ##

# Aggregating high elevation

de4 <- de[!de$elevation %in% c("mid","low"), ]
de4 <- droplevels(de4)
levels(de4$elevation) # selecting high elevation
de5 <- group_by(.data = de4, spec, treat ) %>%
  summarise(vertical_mean = mean(height, na.rm = T),
            neg = mean(height, na.rm = T) - se(height, na.rm = T),
             pos= mean(height, na.rm = T) + se(height, na.rm = T))
de5 <- de5[order(de5$treat), ]
```

```r
# Panel layout

par(mar = c(.24, 2.1, .95, .55))

# Wide format data transformation

format_wide <- acast(data= de5, treat ~ spec, value.var = "vertical_mean")

# High elevation barplot

barplot= barplot(format_wide, col = c("white", "blue"), ylim = c(0, 60), axisnames = F,  axes = T,
  beside = T) # barplot keeps the midpoints bars coordinates; creating texts/error bars
##--------- ---------##------ ------##
## Extra

# Legend
legend(x = 0.8, y = 58, legend = c("Drought","Control"), bty = "n",
       pch = 22,
       pt.bg = c("blue", "white"), pt.cex = 2, cex = 1, y.intersp =
         1.1, xpd = NA)
##--------- ---------##------ ------##

# High elevation title

mtext("High", outer = F, line = -1.7,side = 3, cex = 0.8)

# Error bars

arrows(x0 = barplot,x1 = barplot, y1 = de5$neg, y0 = de5$pos, angle = 90, code = 3,
       length = 0.05 )

# Adding a box to this panel

box()


## Plotting a density histogram for the elevation "low" in the inset


hist(de$height[de$elevation== "high"], main = NA,col = "lightblue",axes = F, freq = F,las = 1 )
densitys <- density(de$height[de$elevation== "high"])
lines(densitys)


##---------------------------------------##---------------------------------------------##

## middle panel ##


#  data subsetting; storing "mid" by ignoring high and low factor levels


de2 = de[!de$elevation %in% c("high","low"), ]
de2 <- droplevels(de2) # remove unwanted factor levels
levels(de2$elevation) # generating "mid"


de3 <- group_by(.data = de2, spec ,treat ) %>%
  summarise(vertical_mean = mean(height, na.rm = TRUE),
            neg = mean(height, na.rm = TRUE) - se(height, na.rm = TRUE),
            pos = mean(height, na.rm = TRUE) + se(height, na.rm = TRUE))
de3

# middle panel Margin
```

```
par(mar = c(.24, 2.1, .95, .55))


format_wide <- acast(de3, treat ~ spec, value.var = "vertical_mean") # install.packages(reshape2) | libaray('
reshape2')

# Creating a barplot for the elevation "mid"

barplot= barplot(format_wide, col = c("white", "blue"), ylim = c(0, 60), axisnames = F, beside = T
                 , axes = T) # barplot keeps the midpoints bars coordinates; creating texts/error bars
# Panel box
box()

# Mid text

mtext("Mid", side = 3, line = -1.7, cex = 0.8, outer = F)# Wide format data transformation

##--------- ---------##------ ------##
## Extra

mtext("Height (cm)",  line = 2, cex = 0.6, side = 2)

##--------- ---------##------ ------##

# Creating error bars

arrows(x0 = barplot, x1 = barplot, y0= de3$pos, y1 = de3$neg , angle = 90, code = 3, length = 0.05)

## Low elevation Density histogram

hist(de$height[de$elevation== "mid"], col = "lightblue", main = NA, las =
1, axes = F, freq = F)
densitys <- density(de$height[de$elevation== "mid"])
lines(densitys)




##----------------------------------------##-------------------------------------------------##
# bottom  panel


# Low elevation

l = de[!de$elevation %in% c("high", "mid"), ] # "[]" allow data subsetting

levels(l$elevation) # generating Low
l= droplevels(l) #  unexploited factor levels are dropped

# species, summary, standard error, treatment in subsets generate object group data.

library(dplyr) #  data summary package loaded via library ()

de1= group_by(.data = l, treat, spec) %>%
  summarise(vertical_mean = mean(height, na.rm = TRUE),
            neg = mean(height, na.rm = TRUE) - se(height, na.rm = TRUE),
            pos = mean(height, na.rm = TRUE) + se(height, na.rm = TRUE))
de1= de1[order(de1$treat), ]
de1

# Wide format data transformation

library(reshape2)

#343434343434format_wide = acast(se1, treat ~ spec, value.var = "vertical_mean")
```

```r
# Margin control

par(mar = c(2.1, 2.1, .95, .55))



# Creating a low elevation barplot for three species

barplot= barplot(format_wide, beside = T, ylim = c(0, 60), axes = T, axisnames
                 = F, col = c("white", "blue")) # 'bp' stores the coordinates of the midpoints of the bars, w
hich you can now use to add error bars or text.

# Top panel text

mtext("Low", outer = F, cex = 0.8,line = -1.7, side = 3, font = 1)

# Error bars

arrows(x0 = barplot, x1 = barplot, y0= de1$pos, y1 = de1$neg , length = 0.05, angle = 90, code = 3)


##--------- ---------##------ ------##
## Extra

# species x-axis, displaying Podocarpus totara as P.totara,
# Beilschmiedia tawa as B.tawa and Agathis australis as A.australis,
text(barplot, -2,
     c("A.australis","A.australis","B.tawa","B.tawa","P.totara","P.totara"
     ),
     srt = 45, adj = c(1, 0), xpd = NA, cex = 0.9)
##--------- ---------##------ ------##

#  box panel
box()

hist(de$height[de$elevation== "low"], col = "lightblue", main = NA, las =
       2, axes = F, freq = F)


## low elevation density histogram

densitys <- density(de$height[de$elevation== "low"])
lines(densitys)
```



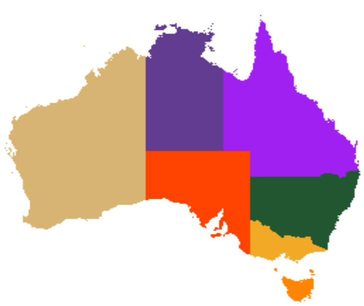
Part 3 : Multi-panel map

```
# Packages

library(ggplot2)
library(ggmap)

## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.

## Please cite ggmap if you use it! See citation("ggmap") for details.

library(ggthemes)
library(RStoolbox)
library(raster)

## Loading required package: sp

##
## Attaching package: 'raster'

## The following object is masked from 'package:dplyr':
##
##     select

library(ggsn)

## Loading required package: grid

##
## Attaching package: 'ggsn'

## The following object is masked from 'package:raster':
##
##     scalebar

library(mapdata)
```

```
## Loading required package: maps

library(broom)
library(rmapshaper)

## Registered S3 method overwritten by 'geojsonlint':
##   method         from
##   print.location dplyr

library(colorRamps)
library(RColorBrewer)

#install.packages(c('ggmap','ggthemes','ggplot2', 'ggmap', 'ggthemes', 'RStoolbox', 'ggsn', 'raster', 'mapdat
a', 'broom', 'rmapshaper', 'colorRamps','RColorBrewer'))

#install.packages(c('RStoolbox', 'ggsn', 'mapdata', 'rmapshaper', 'colorRamps',type = "binary"))
```

Australia States

```
# Australia States

aus = getData(name = "GADM", country= "Australia", level= 1)
nrow(tidy(aus)) # 1426114 polygon areas

## Regions defined for each Polygons

## [1] 1426114

# Adding Australia State colours

cols= colorRampPalette(colors= c("red", "orange", "grey", "darkgreen", "purple"))

# creating a state map of Australia with distinct colours

t = ggplot(data= aus, aes(x= long, y= lat,map_id= id, group= group)) +
  theme_map() +
  coord_quickmap() +
  geom_polygon(size= 0.5,aes(fill= id)) +
  guides(fill= F) +
  scale_fill_manual(values = cols(n= 11))

## Regions defined for each Polygons

  t
```



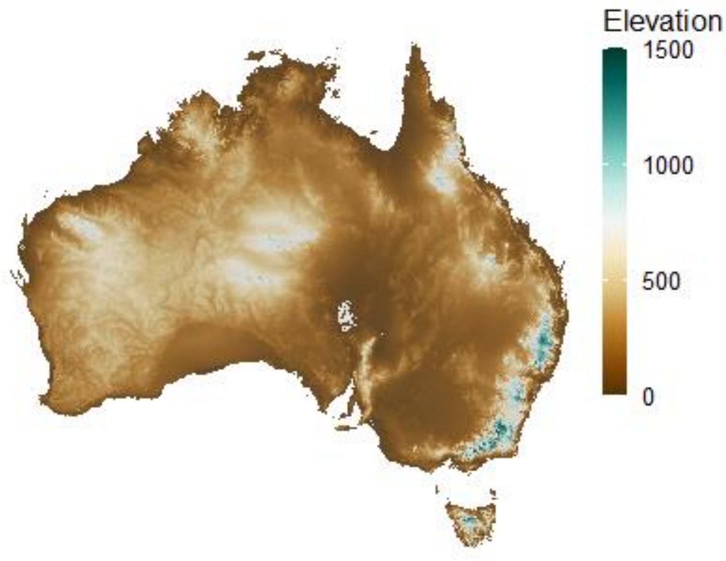
```
#  creating labels or texts of states on the Australia map

states= t + annotate(geom = "text", x = 122, y = -24.8485, label = "Western \n Australia") +
  annotate(geom = "text", x = 133.5, y = -19.9485, label = "Northern\nTerritory") +
  annotate(geom = "text", x = 134.5, y = -28.9485, label = "South \n Australia") +
  annotate(geom = "text", x = 144.5, y = -23, label = "Queensland") +
  annotate(geom = "text", x = 147.5, y = -32, label = "New South \n Wales") +
```

```
  annotate(geom = "text", x = 144.8, y = -36.9, label = "Victoria") +
  annotate(geom = "text", x = 146.8, y = -42, label = "Tasmania")

states
```

Digital elevation models (DEM) of Australia

```
## Download altitude dataset
oz_alt= getData(name = "alt", country = "AUS")

## Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO", prefer_proj =
## prefer_proj): Discarded datum Unknown based on WGS84 ellipsoid in CRS definition

# selects australia via country = "AUS" and elevation from name = "alt"
oz_alt

## class      : RasterLayer
## dimensions : 5496, 5568, 30601728  (nrow, ncol, ncell)
## resolution : 0.008333333, 0.008333333  (x, y)
## extent     : 112.8, 159.2, -54.9, -9.1  (xmin, xmax, ymin, ymax)
## crs        : +proj=longlat +datum=WGS84 +no_defs
## source     : C:/Users/Pawaneet/Desktop/Uni/AUT/pawaneet_R_Us_Graph/project/AUS_msk_alt.grd
## names      : AUS_msk_alt
## values     : -43, 2143  (min, max)

# Raster with elevation data can calculate the aspect,
#any remaining terrain traits and slope of an area
# (note: coordinate reference system (long/lat) need to be in meters).

oz_terr= terrain(opt = c("slope", "aspect"), x = oz_alt[[1]])

oz_terr$slope # indiates australia geograpical information

## class      : RasterLayer
## dimensions : 5496, 5568, 30601728  (nrow, ncol, ncell)
## resolution : 0.008333333, 0.008333333  (x, y)
## extent     : 112.8, 159.2, -54.9, -9.1  (xmin, xmax, ymin, ymax)
## crs        : +proj=longlat +datum=WGS84 +no_defs
## source     : memory
## names      : slope
## values     : 0, 0.4283984  (min, max)

## Compute the hill shade from the slope and aspect layer

oz_hillshade= hillShade(slope = oz_terr$slope, aspect = oz_terr$aspect)
```

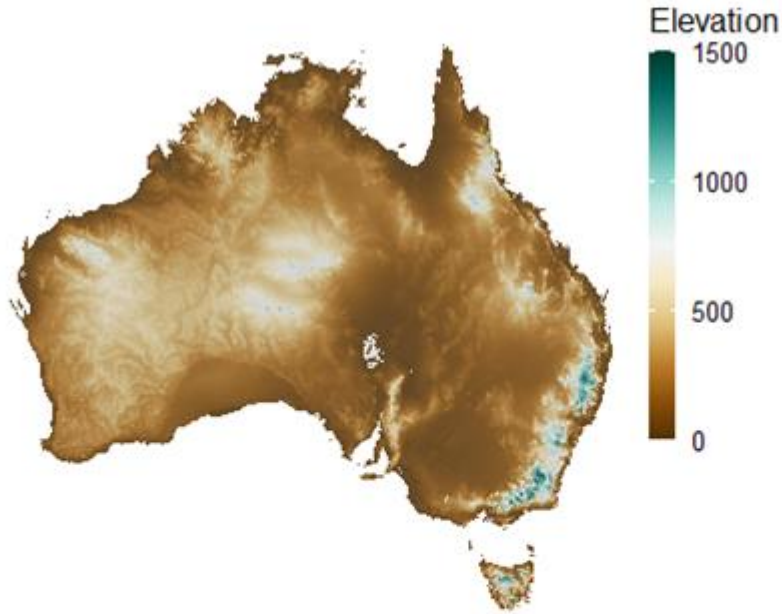
```r
## DEM map of NZ

## Crop the map by finding the studied site


s= extent(c(xmin =  86.842428, xmax = 158.068268, ymin = -49.887759, ymax =-10.240108)) # ss is the studied s
ite

## the hill shade raster is cropped via reference ss ( studied site)

oz_hillshade2= crop(x = oz_hillshade, y = s)

## Same for the elevations

oz_alt2= crop(x = oz_alt, y = s)

## Colour palette for the DEM

library(RColorBrewer)


bbg <- colorRampPalette(colors = brewer.pal(n = 11, name = "BrBG"))


RdYlBu <- colorRampPalette(colors = rev(brewer.pal(n = 11, "RdYlBu")))

# creating a DEM (note use help file or tab to find the arguments functions, values, usage, details)

oz.elv=  ggR(oz_hillshade2) + ggR(oz_alt2, geom_raster = T, ggLayer = T, alpha = 0.75, ggObj = T) +


  scale_fill_gradientn(colours = bbg(n = 1500),

                      na.value = "transparent", limits = c(0, 1500), name = "Elevation",

                        expand = c(2, 1),

                             guide = guide_colourbar(nbin = 2000, barwidth = 0.7,

                                    barheight = 10,

                                    title.theme = element_text(size = 12),

                                    label.theme = element_text(size = 10),

                                    draw.ulim = F)) +

                                    coord_quickmap()  +

                                    theme_map() +

theme(legend.position = c(0.9, 0.4),legend.background = element_rect(colour = NA, fill = NA))

## Coordinate system already present. Adding a new coordinate system, which will replace the existing one.

oz.elv
```



Adding the maps together

```
#install. packages('cowplot')
library(cowplot)

##
## Attaching package: 'cowplot'

## The following object is masked from 'package:ggthemes':
##
##     theme_map

## The following object is masked from 'package:ggmap':
##
##     theme_nothing

library(gridExtra)# generates grid.arrange() function

##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine

library(cowplot)
library(gridExtra)

grid.arrange (oz.elv,states, layout_matrix=
                rbind(c(2, 2, 1, 1),
                     c(2, 2, 1, 1),
                      c(2, 2, 1, 1),
                      c(2, 2, 1, 1)),
              heights = c(10, 10, 0, 0), widths = c(1, 1, 2, 2))
```


Western
Australia
Northern
Territory
Queensland
South
Australia
New South
Wales
Victoria
Tasmania


Elevation
1500
1000
500
0

Digital Elevation Models of New Zealand

Define the bounding box coordinates for New Zealand

```
nz_boundary <- st_as_sf(data.frame(
  x = c(166.5, 179.5, 179.5, 166.5, 166.5),
  y = c(-47.5, -47.5, -34.0, -34.0, -47.5)
), coords = c("x", "y"), crs = 4326)
```

Download DEM data for New Zealand using elevatr

```
dem_data <- get_elev_raster(locations = nz_boundary, z = 5, clip = "bbox")
```

Convert DEM data to a terra raster object and crop to the New Zealand boundary

```
dem_raster <- rast(dem_data)
dem_cropped <- crop(dem_raster, vect(nz_boundary))
```

Convert the DEM raster to a data frame for plotting with ggplot

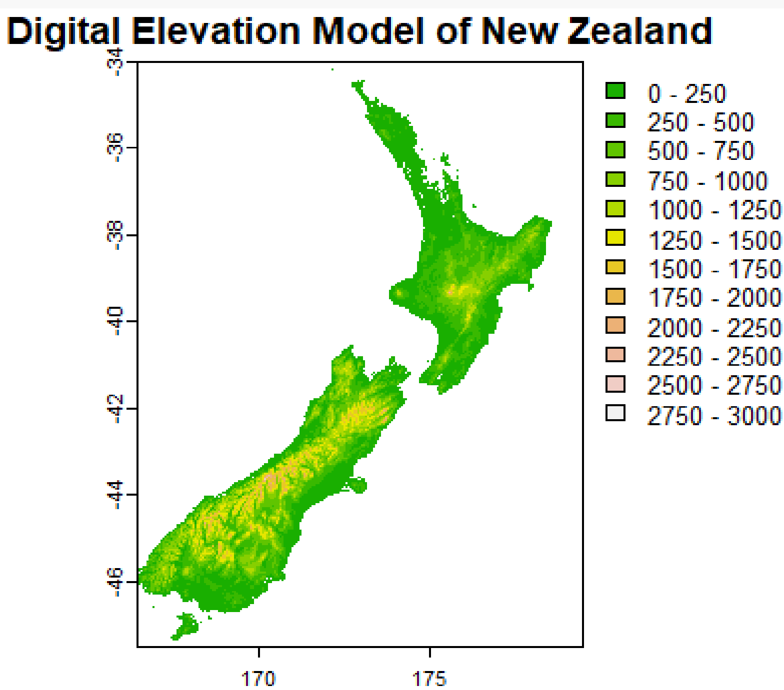
```
dem_df <- terra::as.data.frame(dem_cropped, xy = TRUE)
```

Check the structure of dem_df to ensure the correct column name for elevation

```
elevation_col <- names(dem_df)[3]  # Assuming the third column is the elevation data
```

Plot the DEM using base plotting functions with a different color palette and enhanced scale

```
plot(dem_raster, col = terrain.colors(100), border = NA,
     breaks = seq(0, 3000, by = 250), main = "Digital Elevation Model of New Zealand")
```



Plotting code with north arrow (left side different style)

```
ggplot(dem_df, aes(x = x, y = y, fill = !!sym(elevation_col))) +
  geom_raster() +
  scale_fill_gradientn(colors = terrain.colors(100), na.value = "transparent",
                       breaks = c(0, 1000, 2000, 3000),
                       labels = function(x) ifelse(x < 0, paste0("-", abs(x)), x),
                       limits = c(0, 3000),
                       guide = guide_colorbar(barwidth = 0.5, barheight = 8,
                                              title.position = "top", title.hjust = 0.5,
                                              label.position = "right")) +
  labs(title = "Digital Elevation Model of New Zealand",
       x = "Longitude", y = "Latitude",
       fill = "Elevation (meters)") +
  theme_minimal() +
  theme(axis.text = element_text(size = 10),
        axis.title = element_text(size = 12, face = "bold"),
        plot.title = element_text(size = 14, face = "bold"),
        legend.title = element_text(size = 10, face = "bold"),
        legend.text = element_text(size = 8),
        legend.position = "right", # Adjust legend position
        legend.background = element_blank(),
        plot.margin = margin(20, 20, 40, 20),
        plot.background = element_rect(fill = "white", color = NA)) +
  coord_fixed() +
  # Improved north arrow and text placement
  annotation_north_arrow(location = "tl", which_north = "true",
                         pad_x = unit(0.4, "in"), pad_y = unit(0.4, "in"),
                         style = north_arrow_nautical(
                           fill = c("grey40", "white"),
                           line_col = "grey20",
                           text_family = "ArcherPro Book"
                         )) +
  theme(panel.border = element_blank(), panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(), axis.line = element_blank(),
        plot.title.position = "plot",
        plot.title = element_text(hjust = 0.5))
```