Data analysis in R applied to Marine Science

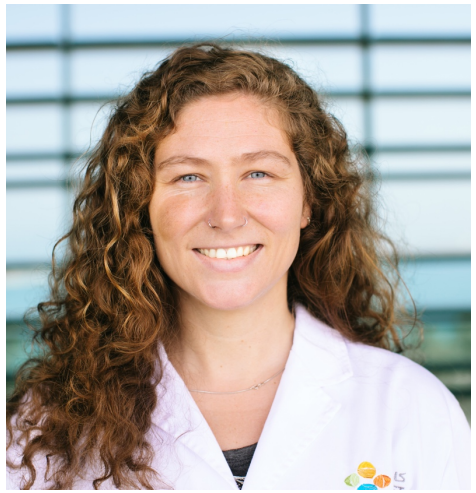
Day 5. Packages for Oceanography
31 January 2019

Dr. Tamara Huete-Stauffer – Dr . Grégoire Michoud – Dr. Daffne López-Sandoval – Dr. Malika Kheireddine

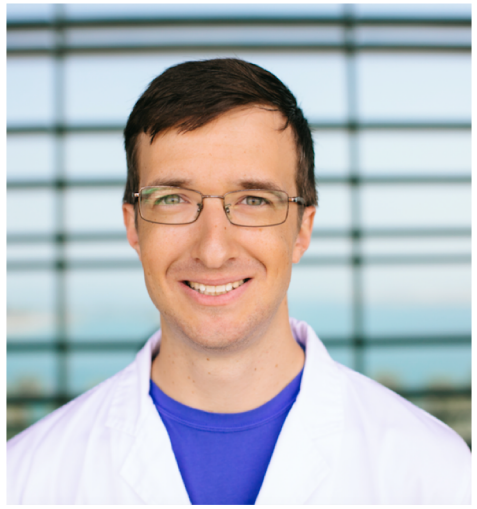
جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

RSRC Red Sea Research Center

Instructors



Tamara Huete-Stauffer
Postdoctoral fellow
Prof. Xelu Morán
Marine Microbial ecology
tamara.huetestauffer@kaust.edu.sa



Grégoire Michoud
Postdoctoral fellow
Prof. Daniele Daffonchio
Microbiology of brine pools
gregoire.michoud@kaust.edu.sa

Daffne López-Sandoval
Postdoctoral Fellow
Prof. Susana Agustí
Phytoplankton ecology
daffne.lopezsandoval@kaust.edu.sa

Marmap package

script: Maps.Rmd
folder: ~/R_Course_2019/Day5/Maps/

1. Marmap

Get high resolution NOAA topography and bathymetry maps

- Install package marmap

```
install.packages("marmap")
```

```
library(marmap)
```

- Download and plot NOAA maps

```
bathy<-getNOAA.bathy(lon1=, lon2=, lat1=, lat2=, resolution=)
```

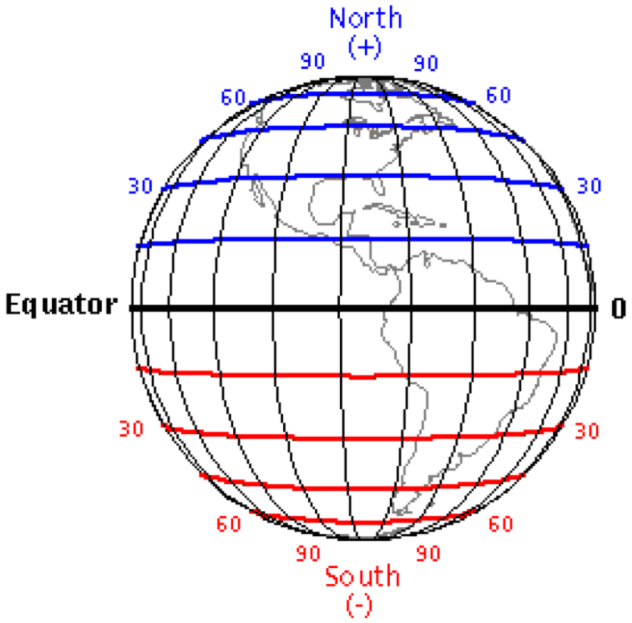
```
plot(bathy)
```

- Overlay any data with lat-long coordinates

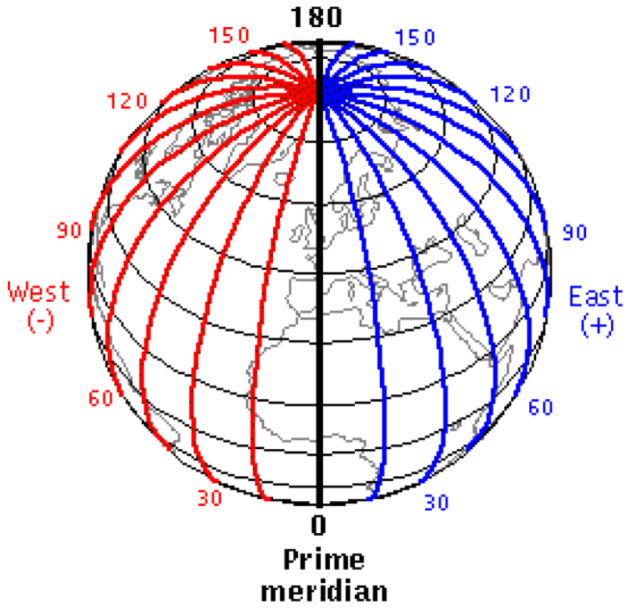
```
points(x=long coord,y=lat coord)
```

2.Latitude – Longitude grid

y = Latitude



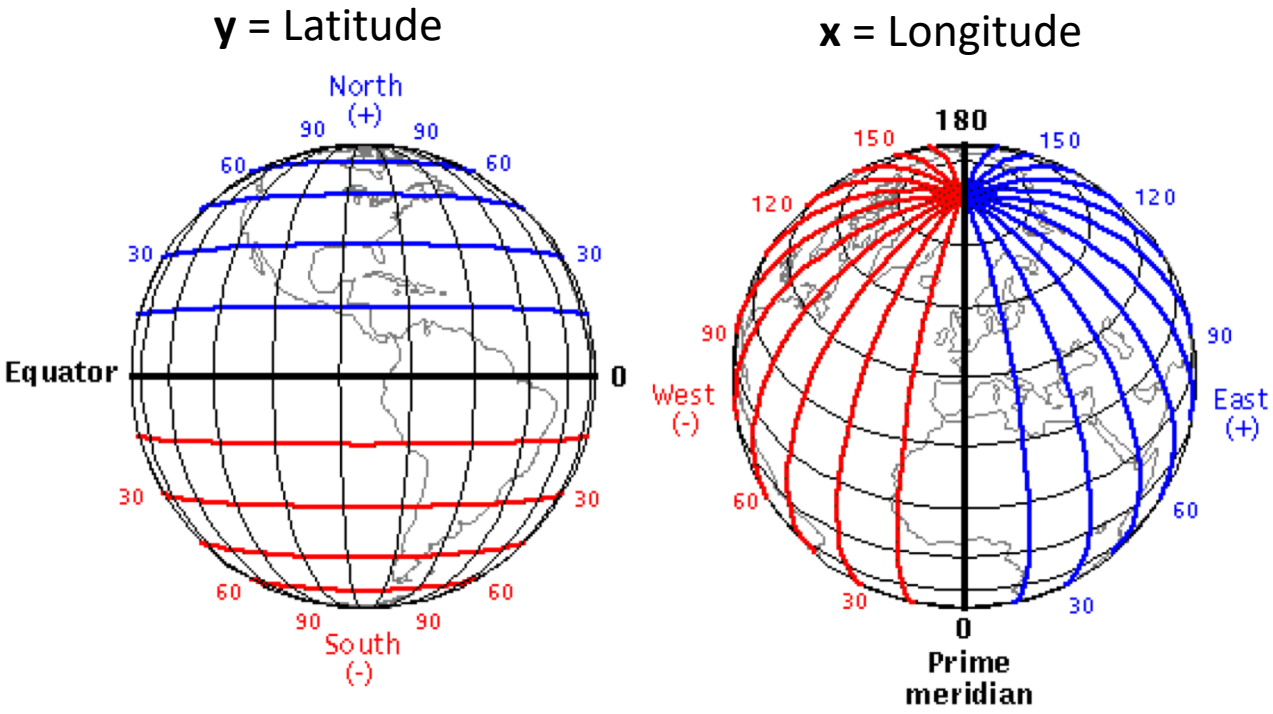
x = Longitude



KAUST
22.309577, 39.104705
latitude longitude

http://www.eylandt.info/earth-globe-map-longitude-latitude-61402/

2.Latitude – Longitude grid



KAUST
22.309577, 39.104705
latitude longitude

Degree Minute Second = 28° 18' 5"

= 28° + 18'/60 + 5"/3600 = **28.30139° Decimal degrees**

marmap uses decimal degrees

http://www.eylandt.info/earth-globe-map-longitude-latitude-61402/

3.Import map and stations

Import the stations files

```
setwd("R_Course_2019/Day5/Maps")
stations <- read.csv("Map_stations.csv")
head(stations)
```

Download the map (Low resolution)

```
PNG_Map_lowres <- getNOAA.bathy(lon1 = , lon2 = , lat1 = ,
lat2 = , resolution = 10)
```

What would you put as coordinates of the Red Sea ?

```
range(stations$lat)
range(stations$long)
```

3.Import map and stations

Import the stations files

```
setwd("R_Course_2019/Day5/Maps")
stations <- read.csv("Map_stations.csv")
head(stations)
```

Download the map (Low resolution)

```
PNG_Map_lowres <- getNOAA.bathy(lon1 = 28, lon2 = 46, lat1
=5, lat2 = 35, resolution = 10)
```

3.Import map and stations

Import the stations files

```
setwd("R_Course_2019/Day5/Maps")
stations <- read.csv("Map_stations.csv")
head(stations)
```

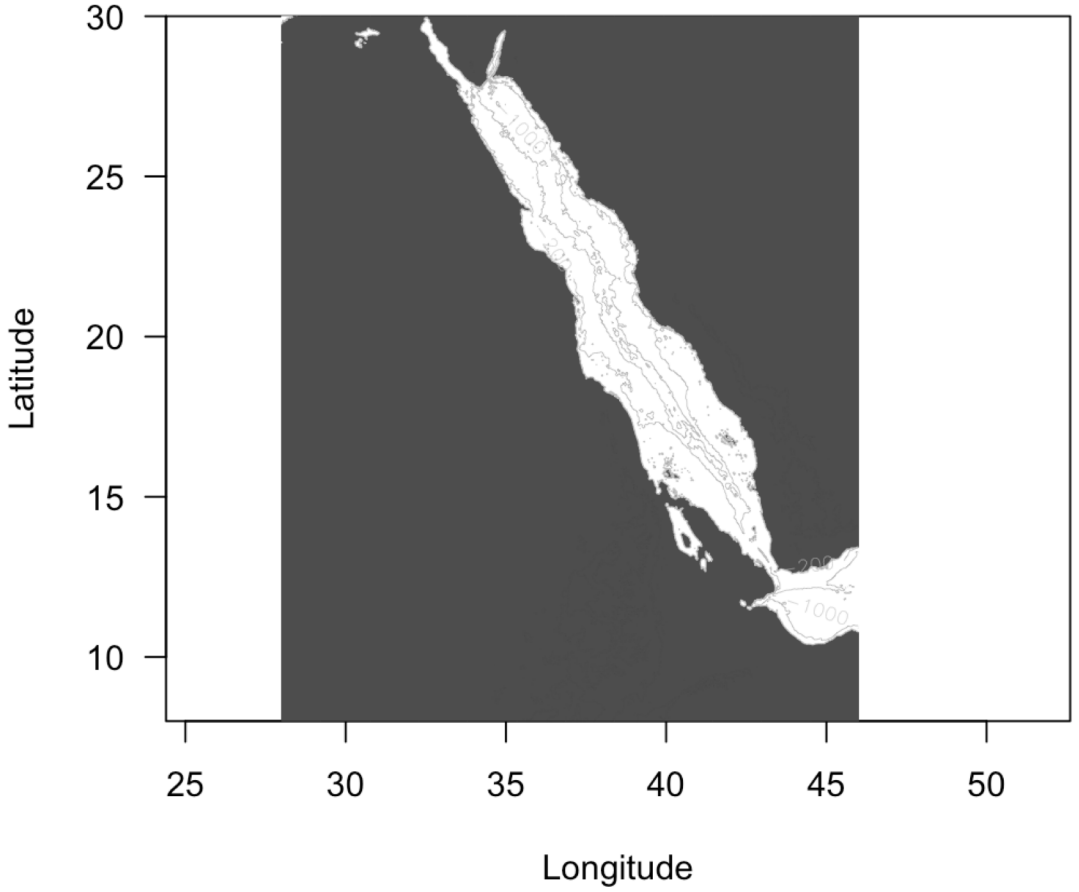
Download the map (Low resolution)

```
PNG_Map_lowres <- getNOAA.bathy(lon1 = 28, lon2 = 46, lat1
=5, lat2 = 35, resolution = 10)
```

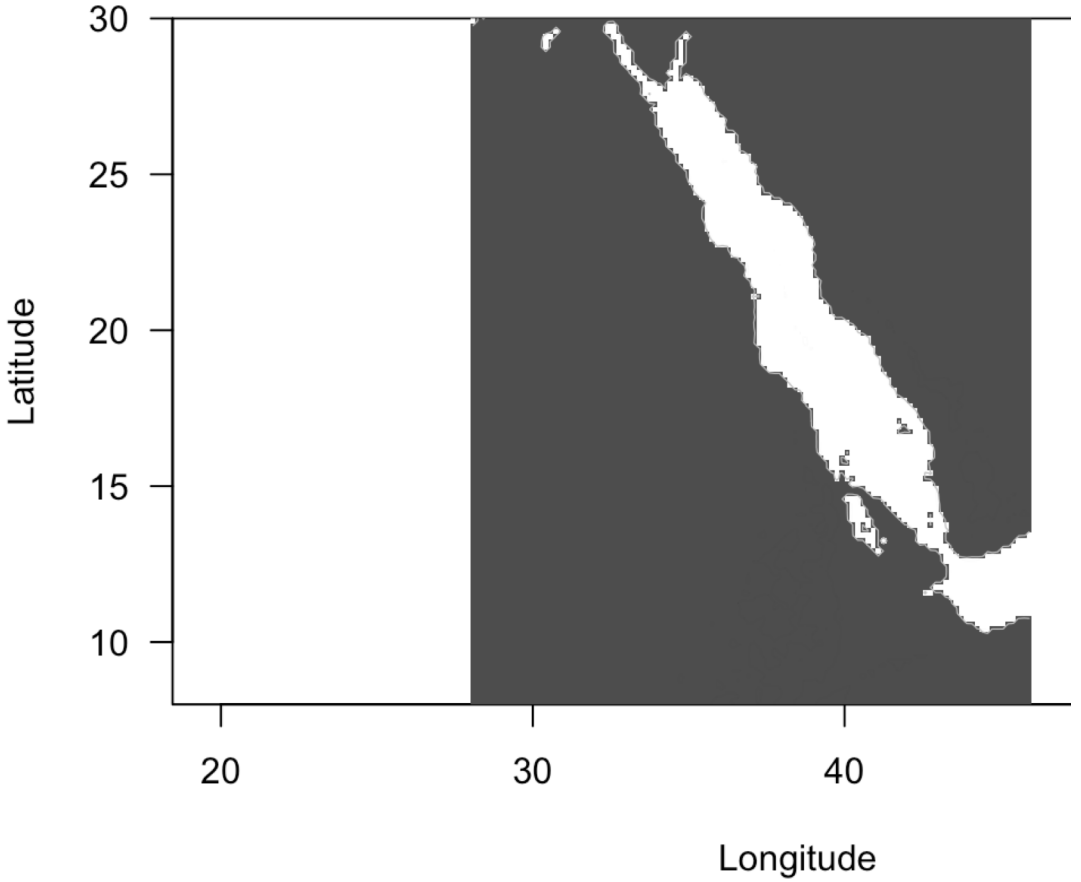
Download the map (High resolution)

```
PNG_Map_highres <- getNOAA.bathy(lon1 = 28, lon2 = 46, lat1
=5, lat2 = 35, resolution = 3)
```

Map resolution



High resolution, resolution = 3 minute



Low resolution, resolution = 20 minutes

4.Plot map

Plot Map

```
plot(PNG_Map_highres, image = TRUE, land = TRUE,
xlim=c(32,45), ylim=c(8,30), n=5, lwd = 0.01, bpal =
list(c(0, max(PNG_Map_highres), gray(.3)),
c(min(PNG_Map_highres),0, "white")),las=1)
```

Add coastline

```
plot(PNG_Map_highres, deep=0, shallow=0, lwd = 0.6,
add=T,col="gray")
```

Add isobath at 200 m

```
plot(PNG_Map_highres, deep=-200, shallow=-200, lwd = 0.4,
drawlabels=T, add=T,col="gray")
```

Exercise

Add isobath at 1000 m

Add isobath at 2000 m

Exercise

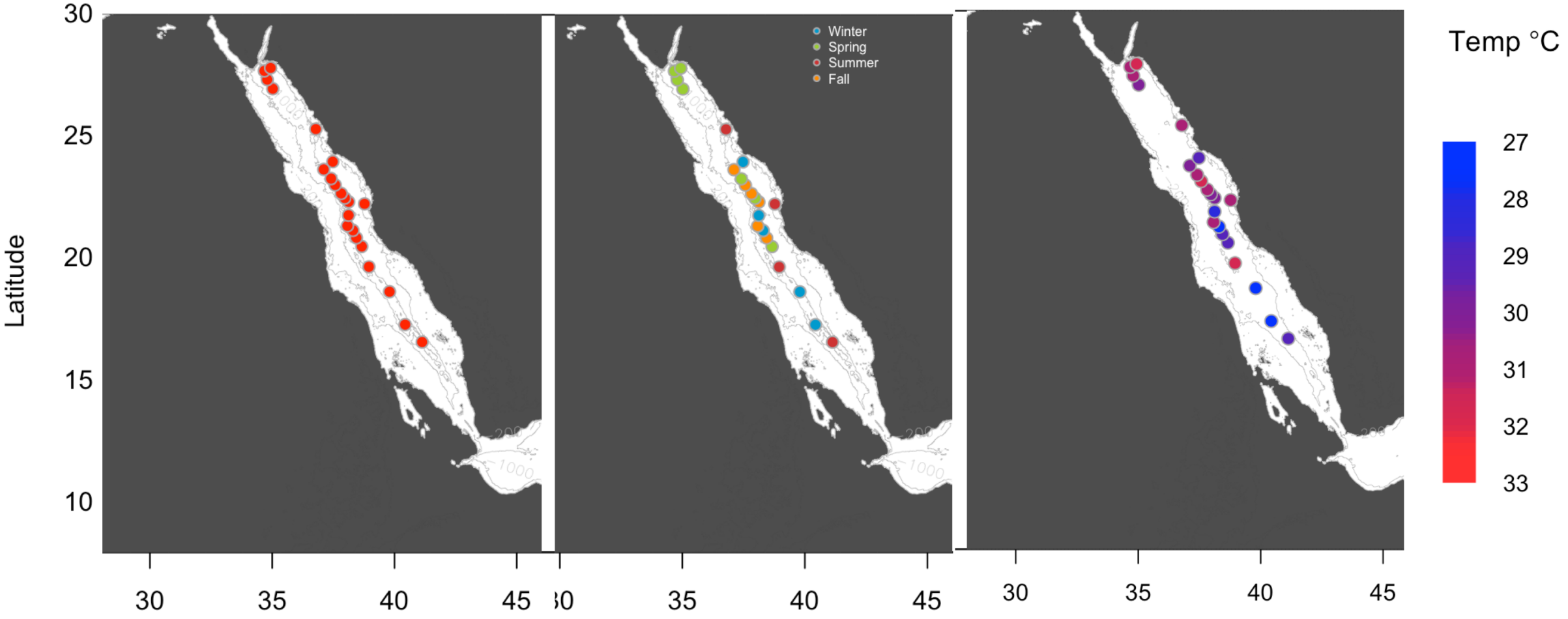
Add isobath at 1000 m

```
plot(PNG_Map_highres, deep=-1000, shallow=-1000, lwd =
0.4, drawlabels=T, add=T,col="gray")
```

Add isobath at 2000 m

```
plot(PNG_Map_highres, deep=-2000, shallow=-2000, lwd =
0.4, drawlabels=T, add=T,col="gray")
```

4. Color points

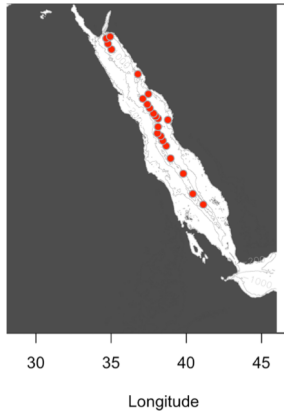


All same color

By categorical variable

By continuous variable

a) All stations the same color




```
plot(PNG_Map_highres,.......)
```

Load PNG map and all the isobaths

```
points(stations$long,stations$lat, bg='red', cex=1, pch=21, col="darkgray")
```

Use points to add the stations

Choose any color for your stations

b) Stations colored by categorical variable


```
stations$season <- factor(stations$season,levels=c(
"Winter","Spring","Summer","Fall" ))
```

```
colseason <-
c("deepskyblue3","yellowgreen","brown3","darkorange")
```

order levels and choose colors

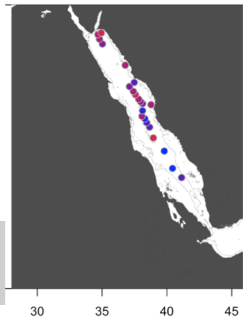
```
plot(PNG_Map_highres,.......)
```

```
points(stations$long,stations$lat, bg= colseason[stations$season] , cex =
1, pch=21, col= "darkgray" )
```

color by season variable

Use points to add the stations

c) Stations colored by continuous variable




```
range(stations$SST)
```

```
length(seq(27,33,by=1))
```

```
stations$Col <- rbPal(7)[as.numeric(cut(stations$SST,breaks =
seq(27,33,by=1),include.lowest=TRUE))]
```

get the range of the variable and make braks

```
plot(PNG_Map_highres,.......)
```

Load PNG map and all the isobaths

```
points(stations$long, stations$lat, bg=stations$Col, cex = 1.2, pch=21,
col="darkgray")
```

Use points to add the stations an color according to temperature

OCE package

folder: Day5/OcePackage

OCE

Process and plot Oceanographic data files

- Install package oce and ocedata

```
install.packages(c("oce","ocedata")
```

```
library(oce)
Library(ocedata)
```

- Example : plot CTD data

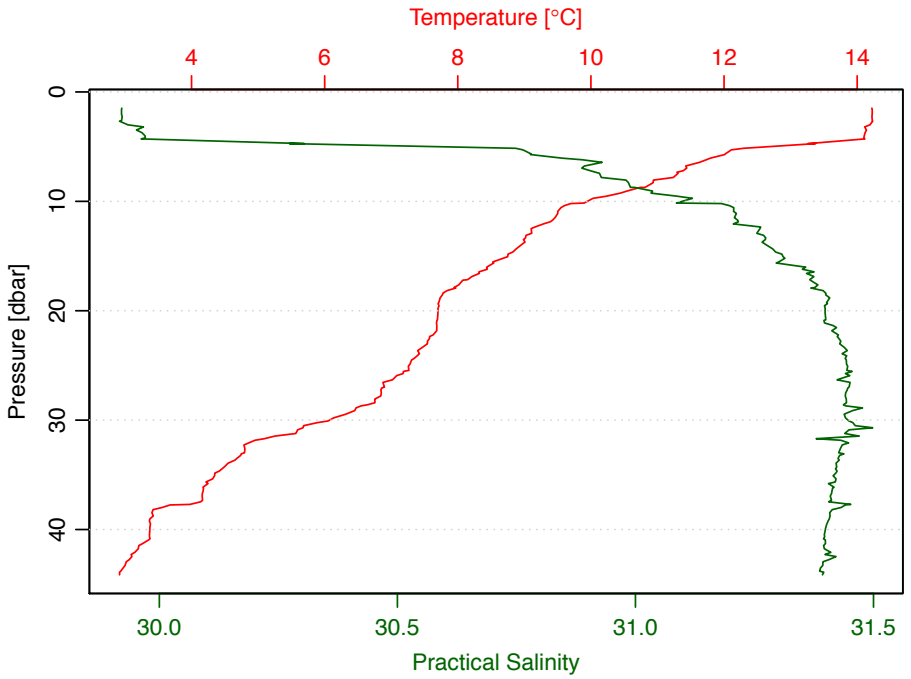
```
data(ctd)
```

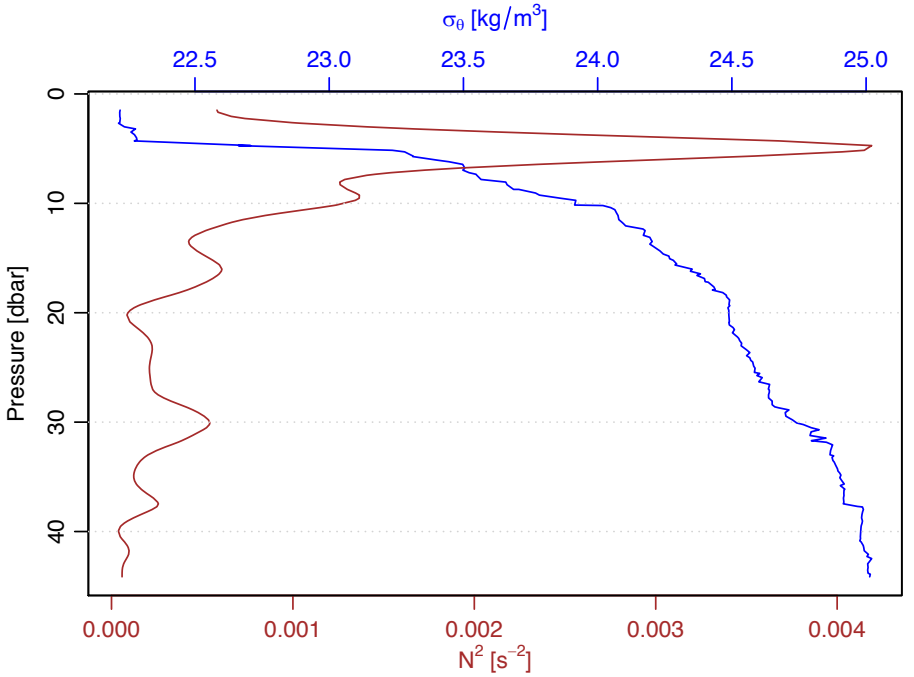
```
plot(ctd, colCoastline="lightgray")
```

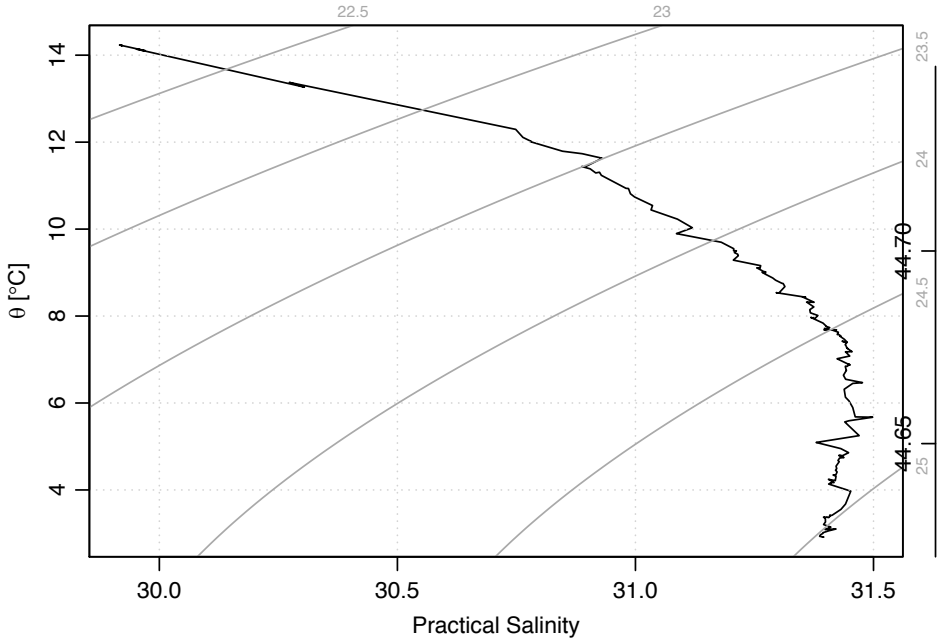
A whole book was written about this package !! :

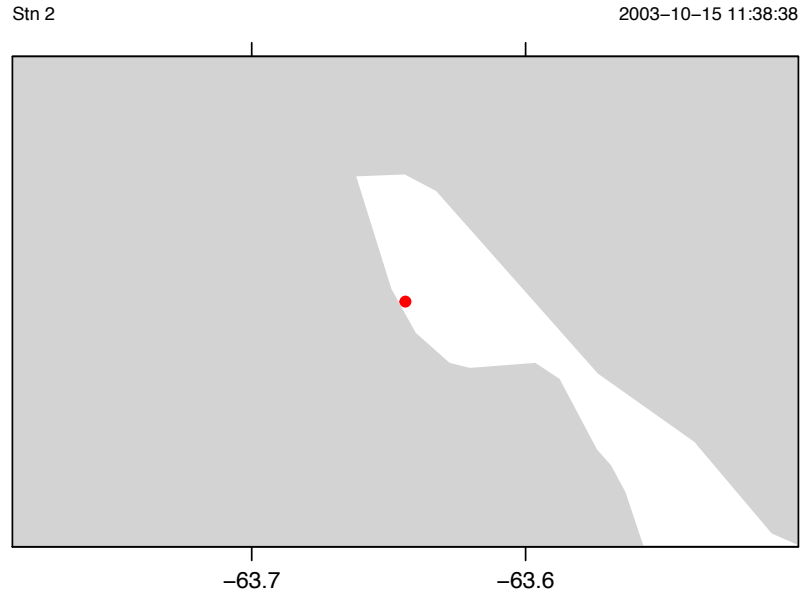
Oceanographic Analysis with R, Dan E. Kelley, Springer

A - Profiles of salinity and temperature
B - Density and the square of buoyancy frequency
C - TS diagram
D - Coastline diagram indicating the station location.









Data

Oce makes heavy use of the R notion of *generic functions*, so that a single function call works across a wide range of data types/objects :

Class	Details
`adp`	Acoustic Doppler profiler, in RDI-Teledyne , Nortek or Sontek format
`adv`	Acoustic Doppler velocimeter, in Nortek or Sontek format
`amsr`	AMSR satellite data
`argo`	Argo float data
`bremen`	Data format used at Bremen
`cm`	Current meter, in Interocean format
`coastline`	Coastline shape, in mapgen, shapefile and other formats
`ctd`	CTD, in Seabird `*.cnv`, WOCE exchange, ODF or Ruskin format
`echosounder`	Biosonics scientific echosounder
`g1sst`	Global 1km SST satellite/model data
`gps`	Location data
`ladp`	Lowered Acoustic Doppler profiler
`landsat`	Landsat satellite data
`lisst`	Laser in situ scattering and transmissometry
`lobo`	Land/Ocean biogeochemistry Observatory
`met`	Meteorological data.
`oce`	Base of all classes in the oce package
`odf`	Data format used by Department of Fisheries and Oceans , Canada
`rsk`	RBR logging devices, e.g. temperature-depth recorders
`satellite`	Base of `amsr`, `g1sst` and `landsat` classes
`sealevel`	Sea-level elevation, in MEDS or Hawaii format
`section`	Section data
`tidem`	Tidal-model data
`topo`	Earth topography, in NOAA format
`windrose`	Wind rose data

Example : Map

- Import internal datasets

```
data("coastlineWorldMedium")
data(endeavour)
```

- Plot the cruise map of the H.M.S. Endeavour cruise in 1768-1771

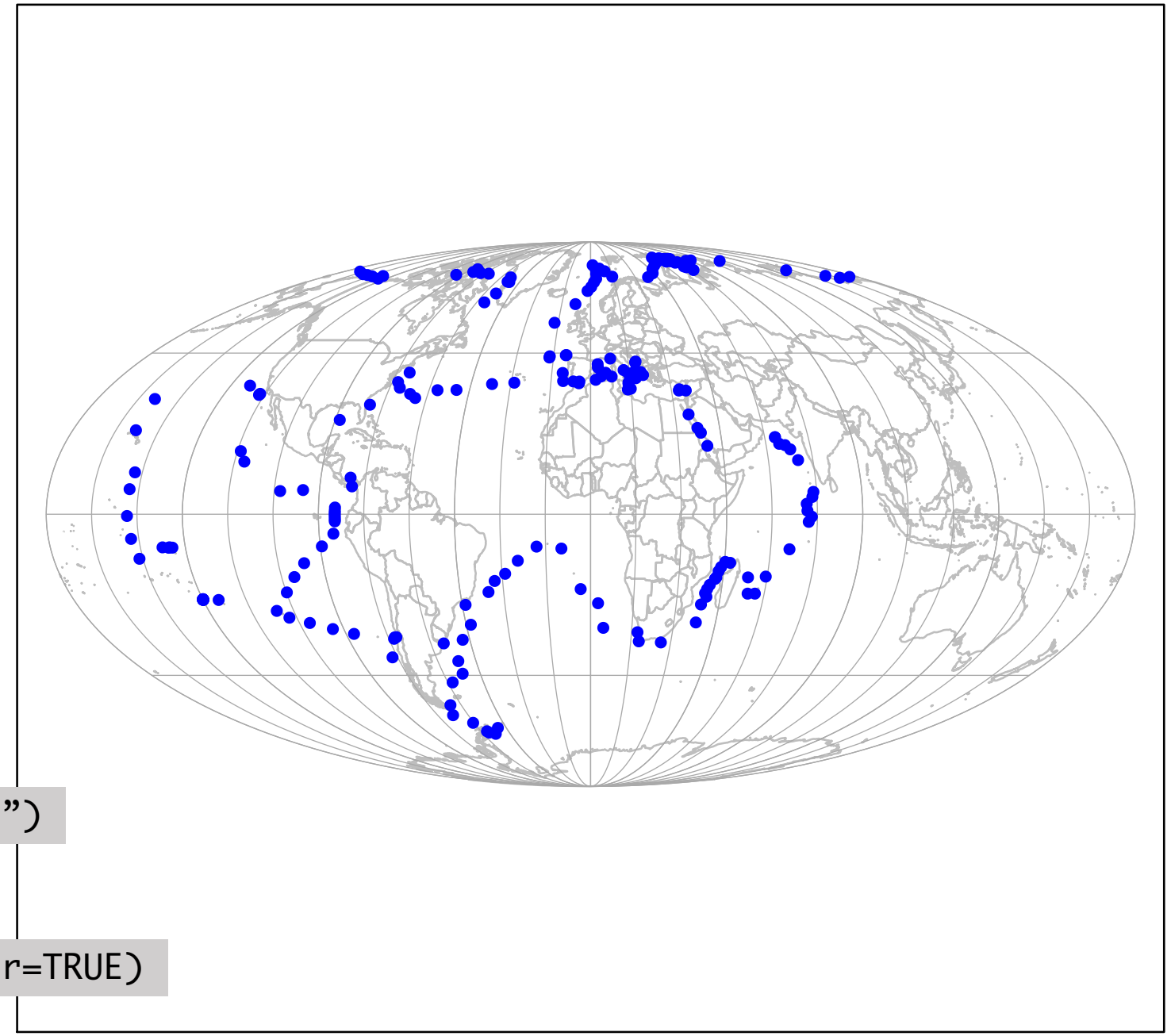
```
mapPlot(coastlineWorldMedium, type='l', col='gray’)

mapPoints(endeavour$longitude, endeavour$latitude,
pch=20, col='red')
```

Exercise

- Use the file

TARA_stations.tab

to obtain this figure



```
read.csv("TARA_stations.tab", sep="\t")
```

or

```
read.table("TARA_stations.tab", header=TRUE)
```

Exercise

```
Tara <- read.table("TARA_stations.tab", sep = "\t", header
= T)
mapPlot(coastlineWorldMedium,type='l', col='gray’)

mapPoints(Tara$Longitude, Tara$Latitude, pch=20,
col='blue')
```

Example : CTD file

- Seabird CTD file are .cnv files : NR1_cast.cnv

```
cast <- read.ctd.sbe("NR1_cast.cnv")
cast <- ctdTrim(cast,method="downcast")
plot(cast)
```

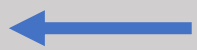
Example : CTD file

- Seabird CTD file are .cnv files : NR1_cast.cnv

```
cast <- read.ctd.sbe("NR1_cast.cnv")
cast <- ctdTrim(cast,method="downcast")
plot(cast)
```

- Tabular data from another CTD company

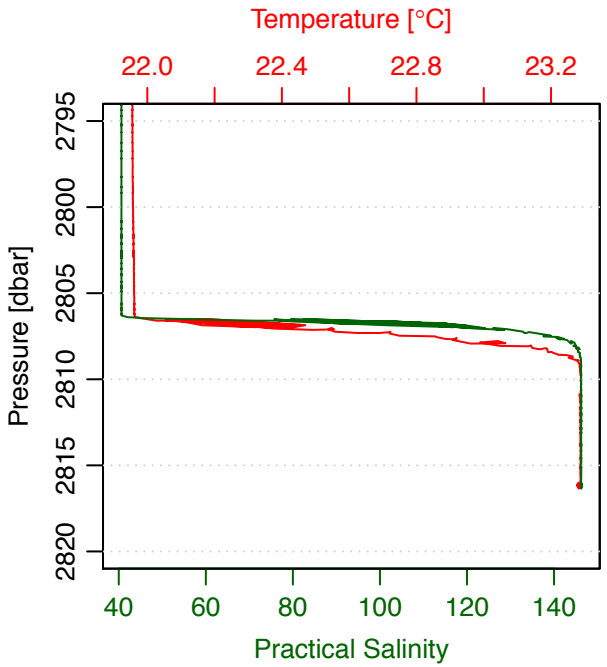
```
castBrine <- read.table("Cast_Brine.txt", header = T)
brine <- as.ctd(castBrine)
plot(brine)
```

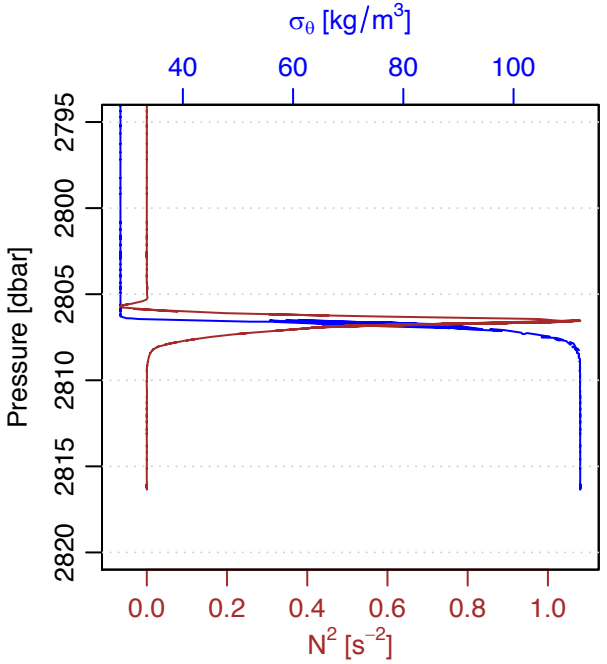


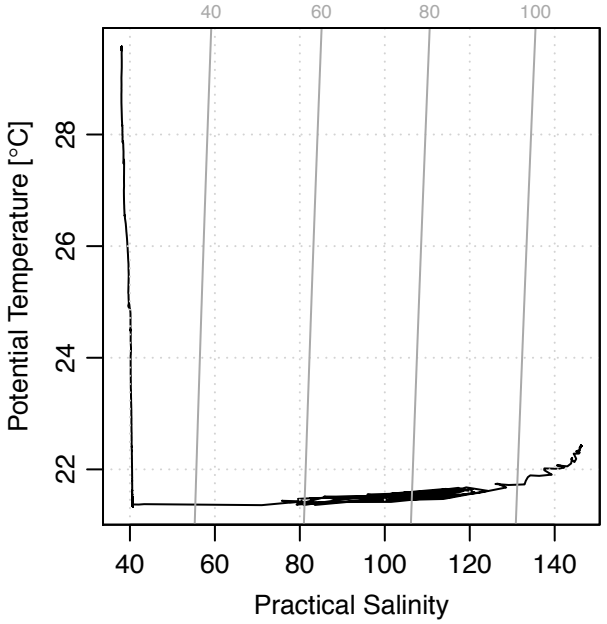
Which parameters were extracted ?

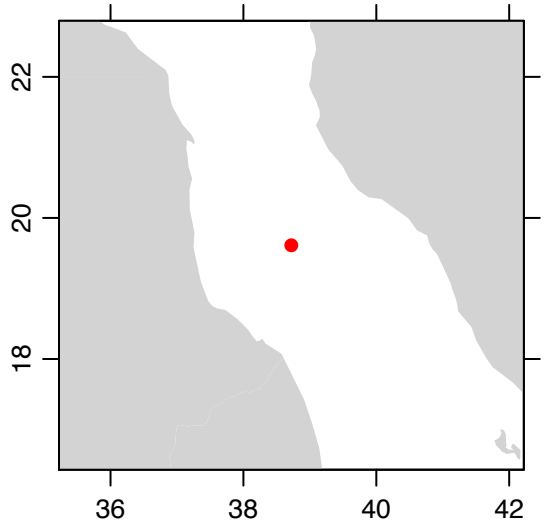
Exercise

- From the previous ctd object, get this figure :





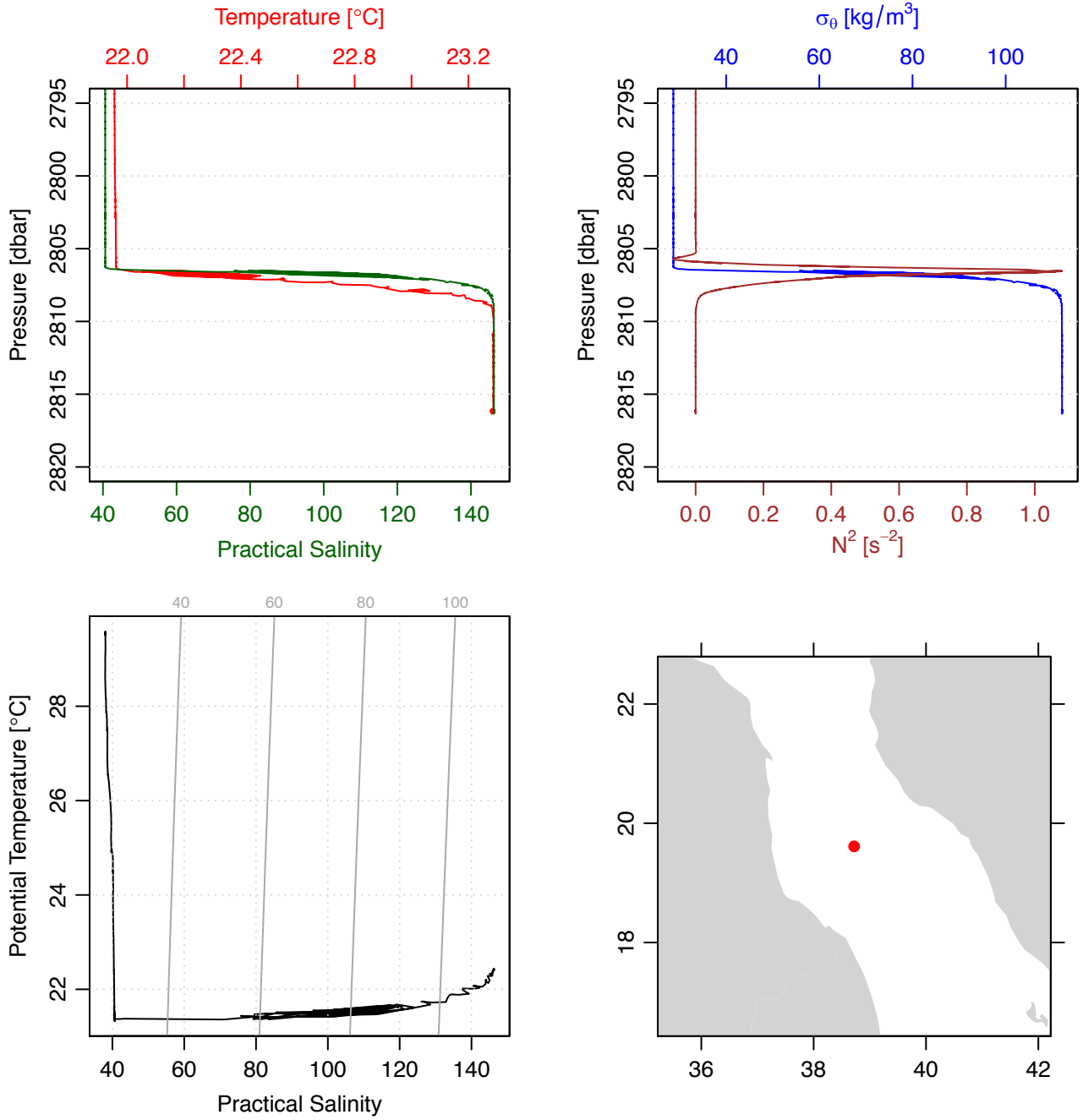




Exercise

- From the previous ctd object, get this figure :

```
plot(brine,
plim=c(2820,2795))
```



Tidyverse

folder: Day5/Tidyverse

Presentation

Coherent system of packages for data manipulation, exploration and visualization that share a common design philosophy

Packages :

- ggplot2
- dplyr
- tidyr
- readr
- purrr
- tibble
- stringr
- forcats

```
Library(tidyverse)
```

Presentation

Coherent system of packages for data manipulation, exploration and visualization that share a common design philosophy

Packages :

- ggplot2
- dplyr
- tidyr
- readr
- purrr
- tibble
- stringr
- forcats

```
Library(tidyverse)
```

readr : Import data

Import data

```
flights <- read_tsv("flights.txt" )
flights
```

Not a dataframe but a **tibble**

Tibbles are data.frames that are lazy and surly:

- They do less (Don’t change variable names or types)
- They complain more (When a variable does not exist)

Goal : **Confront problems earlier**

readr : Import data

Import data

```
flights <- read_tsv("flights.txt" )
flights
```

```
# A tibble: 336,776 x 19
    year month   day dep_time sched_dep_time dep_delay arr_time
   <dbl> <dbl> <dbl>    <dbl>          <dbl>     <dbl>    <dbl>
 1  2013     1     1      517            515         2      830
 2  2013     1     1      533            529         4      850
 3  2013     1     1      542            540         2      923
 4  2013     1     1      544            545        -1     1004
```

dplyR

Grammar of data manipulation, providing a consistent set of verbs that help you solve the most common data manipulation challenges:

- `mutate()` adds new variables that are functions of existing variables
- `select()` picks variables based on their names.
- `filter()` picks cases based on their values.
- `summarise()` reduces multiple values down to a single summary.
- `arrange()` changes the ordering of the rows.

dplyR : operations

filter() : filter base on value

```
flightsJanuary <- filter(flights, month == 1)
```

arrange() : re-order data

```
flightsOrderDelay <- arrange(flights, desc(dep_delay))
```

select()

```
flightsDates <- select(flights, year, month, day)
```

dplyR : operations

mutate()/transmute()

add new columns that are functions of existing columns

```
flightsGain <- transmute(flights,
      gain = arr_delay - dep_delay,
      gain_per_hour = gain / (air_time / 60)
)
```

summarise() : collapses a data frame to a single row

```
summarise(flights,
  delay = mean(dep_delay, na.rm = TRUE)
)
```

dplyR : grouping everything

```
flights %>%
  group_by(year, month, day) %>%
  select(arr_delay, dep_delay) %>%
  summarise(
    arr = mean(arr_delay, na.rm = TRUE),
    dep = mean(dep_delay, na.rm = TRUE)
  ) %>%
  filter(arr > 30 | dep > 30)
```

dplyR : grouping everything

Pipe

```
flights %>%
  group_by(year, month, day) %>%
  select(arr_delay, dep_delay) %>%
  summarise(
    arr = mean(arr_delay, na.rm = TRUE),
    dep = mean(dep_delay, na.rm = TRUE)
  ) %>%
  filter(arr > 30 | dep > 30)
```

Group by day of the year

Select 2 columns

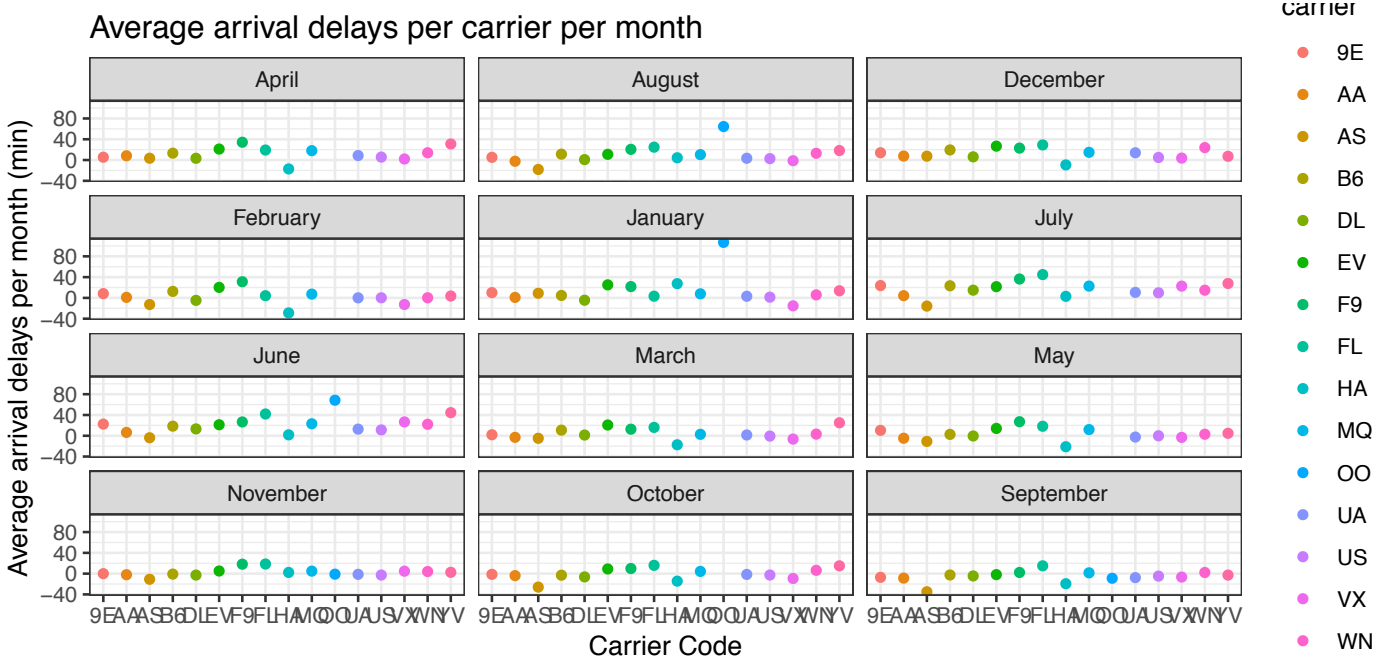
Calculate the mean of both columns

Keep only when delays are above 30 min

dplyR : Exercise

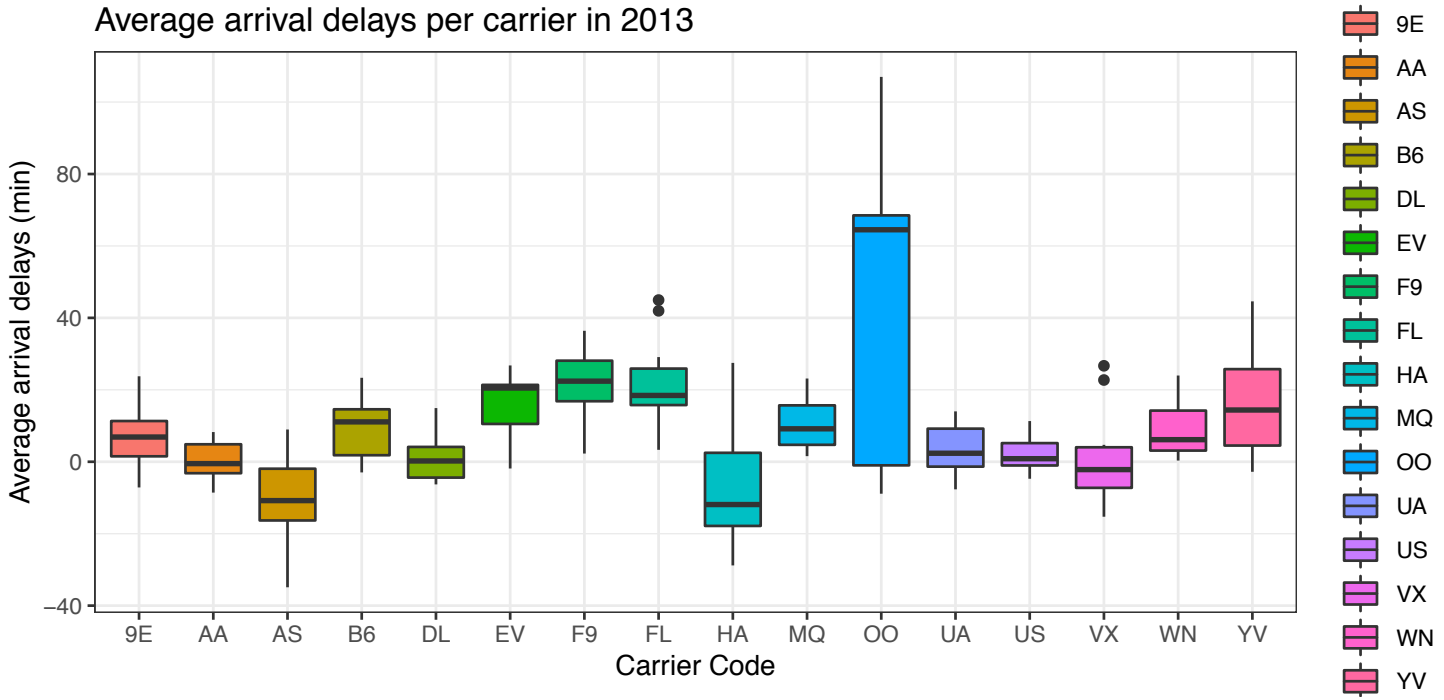
Plot the average arrival delay per airline carrier for every month (geom_point)

Plot the average arrival delay per airline carrier for the year of 2013 as a boxplot



Plot the average arrival delay per airline carrier for every month (geom_point)

Plot the average arrival delay per airline carrier for the year of 2013 (geom_boxplot)



dplyR : How to get the data

```
data <- flights %>%
  mutate()%>%
  group_by( ) %>%
  select() %>%
  summarise()
```

monthText

arrMean

Please create this 2 variables inside the data

dplyR : How to get the data

```
data <- flights %>%
  mutate(
    monthText = months(time_hour)
  )%>%
  group_by() %>%
  select() %>%
  summarise()
```

dplyR : How to get the data

```
data <- flights %>%
  mutate(
    monthText = months(time_hour)
  )%>%
  group_by(carrier, monthDay) %>%
  select(carrier,monthDay, arr_delay) %>%
  summarise(
    arrMean = mean(arr_delay, na.rm = TRUE)
  )
```

dplyR : plot the data

```
ggplot(data, aes(carrier,arrMean, color = carrier))+
  geom_point()+
  theme_bw()+
  facet_wrap(~monthText, ncol = 3)+
  labs(x = "Carrier Code", y = "Average arrival delays per
month (min)",
       title = "Average arrival delays per carrier per
month")
```

```
ggplot(a, aes(carrier,arrMean, fill= carrier))+
  geom_boxplot()+
  theme_bw()+
labs(x = "Carrier Code", y = "Average arrival delays (min)",
     title = "Average arrival delays per carrier in 2013")
```

plot3D package

script: TS_diagram.R
folder: Day5/plot3D

TS diagram

1. Install and load package plot3D

```
install.packages("plot3D")
```

```
library(plot3D)
```

2. Load data (in Rdata format)

```
load("Temp_data.RData")
```

```
load("Dens_data.RData")
```

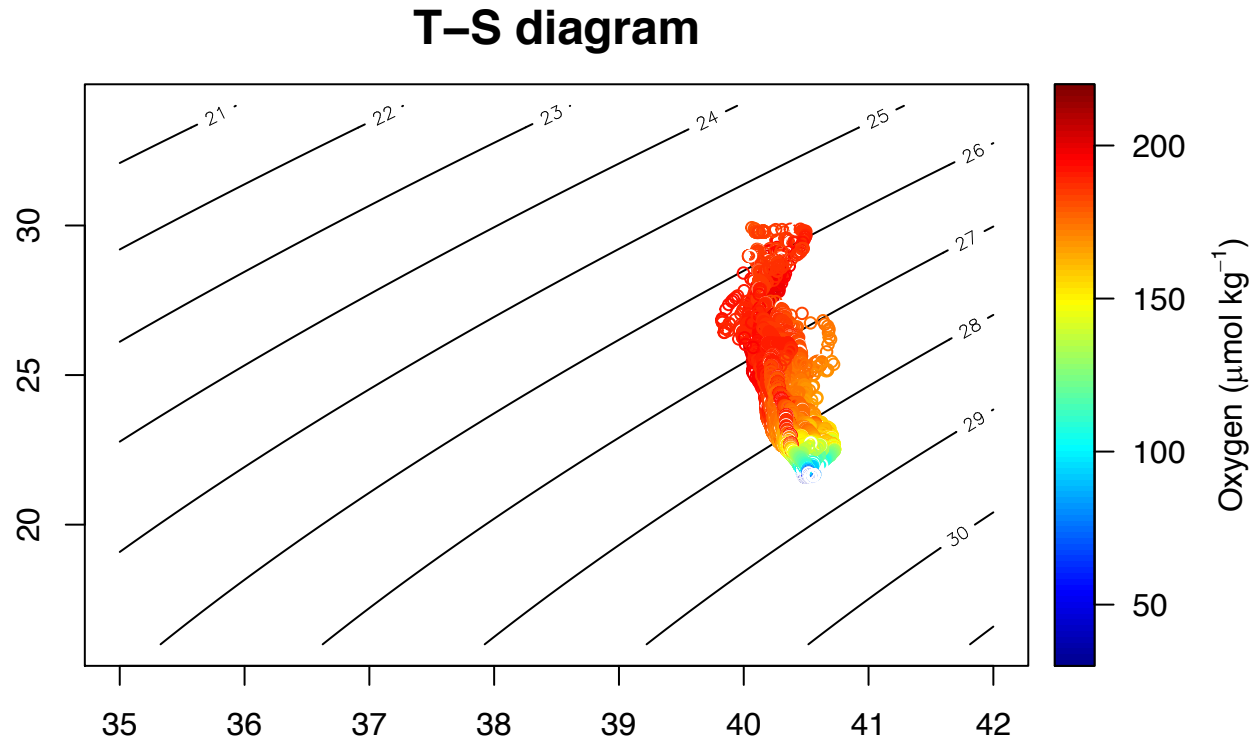
```
load("Sal_data.RData")
```

```
load("Oxy_data.RData")
```

3. Create a matrix of density values with each combination of temperature and salinity values

- Use the density function from the script (line 34)

4. Plot the density contour and overlay Oxygen values



*Example 4 from the script (line 80)