Technical Report: A Data Analysis Study of Temperature Based on CALCOFI Data.

Summary:

This report presents a stepwise numerical and categorical analysis of the CalCOFI Cast and Bottle dataset to understand and predict ocean temperature in the California Current System. The purpose of the analysis is to build and test a statistical model that **predicts ocean temperature** from measurements collected from the same water samples —especially salinity, dissolved oxygen, ammonia, and nutrients—across many stations and depths, using all other variables. This report is also part of the Capstone project's white paper.

Project Question:

1. Numerical Analysis: Does temperature prediction improve when we use all available ocean measurements together, instead of analyzing temperature against one variable at a time?
2. Categorical Analysis: Do the physical relationships in the ocean shift or behave differently depending on the Time (Season), Place (Geography), or Depth Layer?

My Intervention:

These are the steps I used to analyze the CALCOFI data.

1. Download actual real data (bottle and cast data from CALCOFI) and merge both datasets into one dataset using MSSQL.

2. Clean and merge the real CalCOFI files in R.

3. Analysis of Numerical Analysis using Correlation and multilinear regression vs Ridge regression, split the data into test or train for all the variables based on station id to predict the temperature on the best-fitted model.

4. Perform all the categorical analyses based on ANOVA, bar chart, and maps.

Note: Please read "Capstone Project CodeBook and DataSet Readme instructions- Calcofi Oceanographic Data" and CalCofi DataSets CodeBook to understand the indicators better.

Project Questions:

ID	Question	Answered in Section
Q1	How strongly does temperature change with depth?	Check Result Section
Q2	Is salinity related to temperature?	Check Result Section

Q3	Does depth mostly explain the salinity-temperature relationship?	Check Result Section
Q4	Does adding depth improve prediction accuracy?	Check Result Section
Q5	Do oxygen/ammonia/nutrients improve prediction beyond salinity+depth?	Check Result Section
Q6	Are predictors overlapping too much (multicollinearity)?	Check Result Section
Q7	If overlap is high, should Ridge be used?	Check Result Section
Q8	If we use all available variables, do we predict better?	Check Result Section
Q9	What are the final findings?	Check Result Section
Q10	Does the relationship between temperature and salinity change as we go deeper?	Check Result Section
Q11	Does where the water is located change its thermal behavior?	Check Result Section
Q12	Does the time of year change the "rules" of the ocean?	Check Result Section

MSQL Module I have used:

Module name	Why do we use it
Task	To import files into the data as tables faster.
Queries	Join Queries to join the tables.

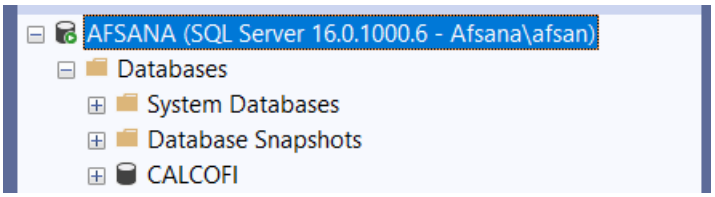
R Packages I have used:

Package	**Why do we use it**	**CRAN link**
tidyverse	Reading CSV, cleaning, plotting, tables	https://cran.r-project.org/package=tidyverse
janitor	Standardize messy column names	https://cran.r-project.org/package=janitor
stringi	Safely convert strange characters to UTF-8	https://cran.r-project.org/package=stringi
rsample	Train/Test split, including grouped split by cruise	https://cran.r-project.org/package=rsample
yardstick	Metrics (R2, RMSE, MAE) to compare models fairly	https://cran.r-project.org/package=yardstick
broom	Tidy model output for reporting	https://cran.r-project.org/package=broom
ppcor	Partial correlation (controls for depth confounding)	https://cran.r-project.org/package=ppcor
car	VIF for multicollinearity	https://cran.r-project.org/package=car
glmnet	Ridge/Lasso regression (stable with overlapping predictors)	https://cran.r-project.org/package=glmnet
Matrix	Sparse model matrices (important when the station has many levels)	https://cran.r-project.org/package=Matrix
knitr	Print tables nicely in the report	https://cran.r-project.org/package=knitr
ggcorrplot	Create the heatmap	https://CRAN.R-project.org/package=ggcorrplot

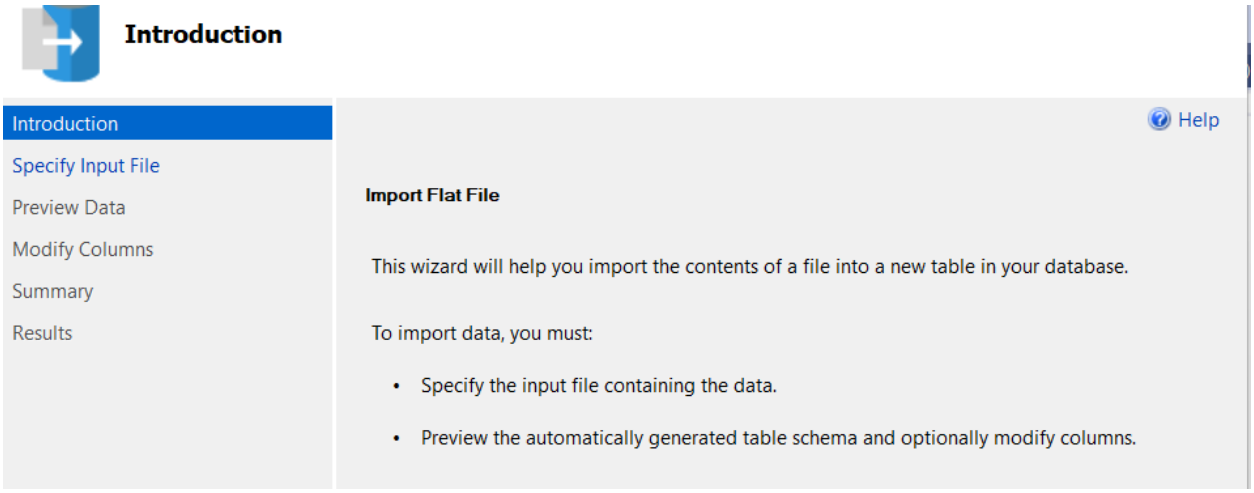
Pre-Analysis Task in MSSQL:

I have used the MSSQL task to perform the initial data merge because, from my personal experience, data import is generally faster for large-scale data retrieval, joining, and aggregation.

1. Create a database of CalCOFI data:
   ```
   CREATE DATABASE [CALCOFI]
   ```



2. Once the database was created, import the actual raw data via the import task in SQL. The image is given below:



3. Once imported, the data looks like it.

   ```
   Select top 10 * from [dbo].[194903-202105_Bottle]
   ```

Results | Messages

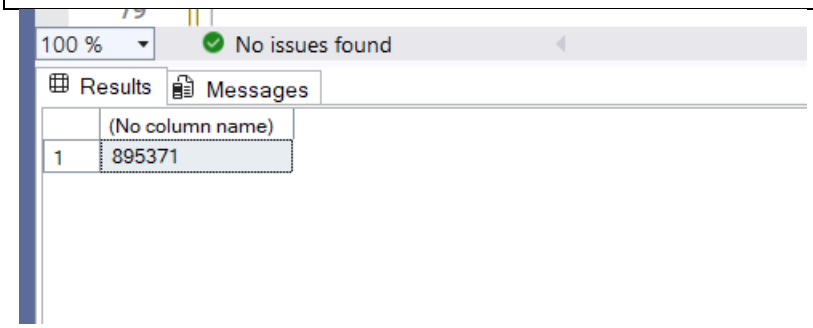
	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2Sat	Oxy_µmol_Kg	BtlNum	RecInd	T_prec	T_qual	S_prec	S_qual	P_qual	O_qual	STHtaq	O2Sa
1	1	1	054.0 056.0	19-4903CR-HY-060-0930-05400560-0000A-3	0	10.5	33.44	NULL	25.649	NULL	NULL	NULL	3	1	NULL	2	NULL	9	9	NULL	9
2	1	2	054.0 056.0	19-4903CR-HY-060-0930-05400560-0008A-3	8	10.46	33.44	NULL	25.656	NULL	NULL	NULL	3	2	NULL	2	NULL	9	9	NULL	9
3	1	3	054.0 056.0	19-4903CR-HY-060-0930-05400560-0010A-7	10	10.46	33.437	NULL	25.654	NULL	NULL	NULL	7	2	NULL	3	NULL	9	9	NULL	9
4	1	4	054.0 056.0	19-4903CR-HY-060-0930-05400560-0019A-3	19	10.45	33.42	NULL	25.643	NULL	NULL	NULL	3	2	NULL	2	NULL	9	9	NULL	9
5	1	5	054.0 056.0	19-4903CR-HY-060-0930-05400560-0020A-7	20	10.45	33.421	NULL	25.643	NULL	NULL	NULL	7	2	NULL	3	NULL	9	9	NULL	9
6	1	6	054.0 056.0	19-4903CR-HY-060-0930-05400560-0030A-7	30	10.45	33.431	NULL	25.651	NULL	NULL	NULL	7	2	NULL	3	NULL	9	9	NULL	9
7	1	7	054.0 056.0	19-4903CR-HY-060-0930-05400560-0039A-3	39	10.45	33.44	NULL	25.658	NULL	NULL	NULL	3	2	NULL	2	NULL	9	9	NULL	9
8	1	8	054.0 056.0	19-4903CR-HY-060-0930-05400560-0050A-7	50	10.24	33.424	NULL	25.682	NULL	NULL	NULL	7	2	NULL	3	NULL	9	9	NULL	9
9	1	9	054.0 056.0	19-4903CR-HY-060-0930-05400560-0058A-3	58	10.06	33.42	NULL	25.71	NULL	NULL	NULL	3	2	NULL	2	NULL	9	9	NULL	9
10	1	10	054.0 056.0	19-4903CR-HY-060-0930-05400560-0075A-7	75	9.86	33.494	NULL	25.801	NULL	NULL	NULL	7	2	NULL	3	NULL	9	9	NULL	9

   ```
   Select top 10 * from [dbo].[194903-202105_Cast]
   ```

	Cst_Cnt	Cruise_ID	Cruise	Cruz_Sta	DbSta_ID	Cast_ID	Sta_ID	Quarter	Sta_Code	Distance	Date	Year	Month	Julian_Date	Julian_Day
1	1	1949-03-01-C-31CR	194903	19490305400560	05400560	19-4903CR-HY-060-0930-05400560	054.0 056.0	1	NST	NULL	03/01/1949	1949	3	17958	60
2	2	1949-03-01-C-31CR	194903	19490305200750	05200750	19-4903CR-HY-060-2112-05200750	052.0 075.0	1	NST	NULL	03/01/1949	1949	3	17958	60
3	3	1949-03-01-C-31CR	194903	19490305100850	05100850	19-4903CR-HY-061-0354-05100850	051.0 085.0	1	NST	NULL	03/02/1949	1949	3	17959	61
4	4	1949-03-01-C-31CR	194903	19490305000950	05000950	19-4903CR-HY-061-1042-05000950	050.0 095.0	1	NST	NULL	03/02/1949	1949	3	17959	61
5	5	1949-03-01-C-31CR	194903	19490305001040	05001040	19-4903CR-HY-061-1706-05001040	050.0 104.0	1	NST	NULL	03/02/1949	1949	3	17959	61
6	6	1949-03-01-C-31CR	194903	19490304901140	04901140	19-4903CR-HY-062-0036-04901140	049.0 114.0	1	NST	NULL	03/03/1949	1949	3	17960	62
7	7	1949-03-01-C-31CR	194903	19490305671460	05671460	19-4903CR-HY-063-0506-05671460	056.7 146.0	1	NST	NULL	03/04/1949	1949	3	17961	63
8	8	1949-03-01-C-31CR	194903	19490305671360	05671360	19-4903CR-HY-063-1154-05671360	056.7 136.0	1	NST	NULL	03/04/1949	1949	3	17961	63
9	9	1949-03-01-C-31CR	194903	19490305801270	05801270	19-4903CR-HY-063-1742-05801270	058.0 127.0	1	NST	NULL	03/04/1949	1949	3	17961	63
10	10	1949-03-01-C-31CR	194903	19490305901170	05901170	19-4903CR-HY-063-2354-05901170	059.0 117.0	1	NST	NULL	03/04/1949	1949	3	17961	63

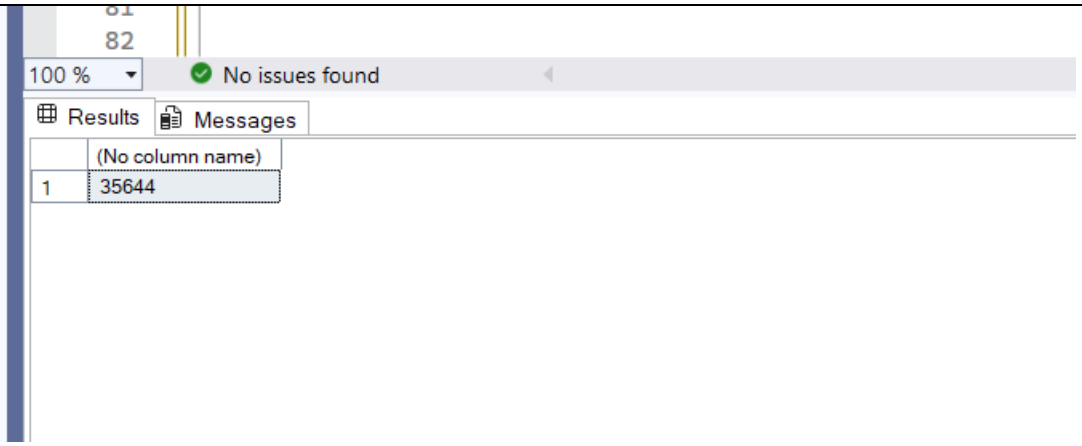
4. Row counts of the records:
   For Bottle: The total number of records is 895371

```
Select count(*) from [dbo].[194903-202105_Bottle]
```



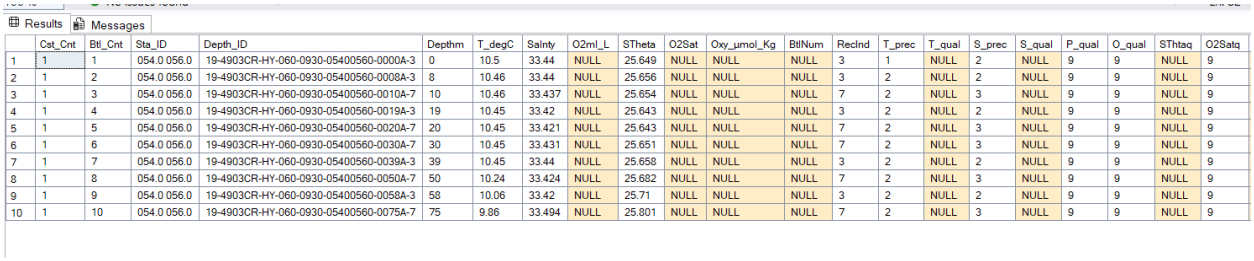
   For Cast: The total number of records is 35644

```
Select count(*) from [dbo].[194903-202105_Cast]
```



5. Join both tables with the column, cst_cnt, to get the merged data. Here are the top 10 records of it.

```
Select TOP 10 T1.*, T2.* from [dbo].[194903-202105_Bottle] T1
Join [dbo].[194903-202105_Cast]  T2 on T1.Cst_Cnt = T2.Cst_Cnt
```



Results | Messages

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2Sat	Oxy_µmol_Kg	BtlNum	RecInd	T_prec	T_qual	S_prec	S_qual	P_qual	O_qual	SThtaq	O2Satq
1	1	1	054.0 056.0	19-4903CR-HY-060-0930-05400560-0000A-3	0	10.5	33.44	NULL	25.649	NULL	NULL	NULL	3	1	NULL	2	NULL	9	9	NULL	9
2	1	2	054.0 056.0	19-4903CR-HY-060-0930-05400560-0008A-3	8	10.46	33.44	NULL	25.656	NULL	NULL	NULL	3	2	NULL	2	NULL	9	9	NULL	9
3	1	3	054.0 056.0	19-4903CR-HY-060-0930-05400560-0010A-7	10	10.46	33.437	NULL	25.654	NULL	NULL	NULL	7	2	NULL	3	NULL	9	9	NULL	9
4	1	4	054.0 056.0	19-4903CR-HY-060-0930-05400560-0019A-3	19	10.45	33.42	NULL	25.643	NULL	NULL	NULL	3	2	NULL	2	NULL	9	9	NULL	9
5	1	5	054.0 056.0	19-4903CR-HY-060-0930-05400560-0020A-7	20	10.45	33.421	NULL	25.643	NULL	NULL	NULL	7	2	NULL	3	NULL	9	9	NULL	9
6	1	6	054.0 056.0	19-4903CR-HY-060-0930-05400560-0030A-7	30	10.45	33.431	NULL	25.651	NULL	NULL	NULL	7	2	NULL	3	NULL	9	9	NULL	9
7	1	7	054.0 056.0	19-4903CR-HY-060-0930-05400560-0039A-3	39	10.45	33.44	NULL	25.658	NULL	NULL	NULL	3	2	NULL	2	NULL	9	9	NULL	9
8	1	8	054.0 056.0	19-4903CR-HY-060-0930-05400560-0050A-7	50	10.24	33.424	NULL	25.682	NULL	NULL	NULL	7	2	NULL	3	NULL	9	9	NULL	9
9	1	9	054.0 056.0	19-4903CR-HY-060-0930-05400560-0058A-3	58	10.06	33.42	NULL	25.71	NULL	NULL	NULL	3	2	NULL	2	NULL	9	9	NULL	9
10	1	10	054.0 056.0	19-4903CR-HY-060-0930-05400560-0075A-7	75	9.86	33.494	NULL	25.801	NULL	NULL	NULL	7	2	NULL	3	NULL	9	9	NULL	9

6. After that, I have renamed the fields with prefixes as “Bottle-###” or “Cast-###.”

Data Analysis Part in R:

I used R for statistical analysis, as I have prior experience with it from previous coursework, and, coming from an applied mathematics background, I truly love using it for this project. It is easy to do statistical analysis in R. Here is the step I have used:

1. Import packages and set libraries to do the analysis. I don’t need to install it since I have already installed it. Anyone can install it by using the command *install.packages("Package name")* in R.

```
library(tidyverse)
library(janitor)
library(stringi)
library(rsample)
library(yardstick)
library(broom)
library(ppcor)
library(car)
library(glmnet)
library(Matrix)
library(knitr)
library(readxl)
library(dplyr)
library(stringr)
library(readr)
library(leaflet)
```

2. Get the path and set the path as a variable.
   2.1 Read the file path.

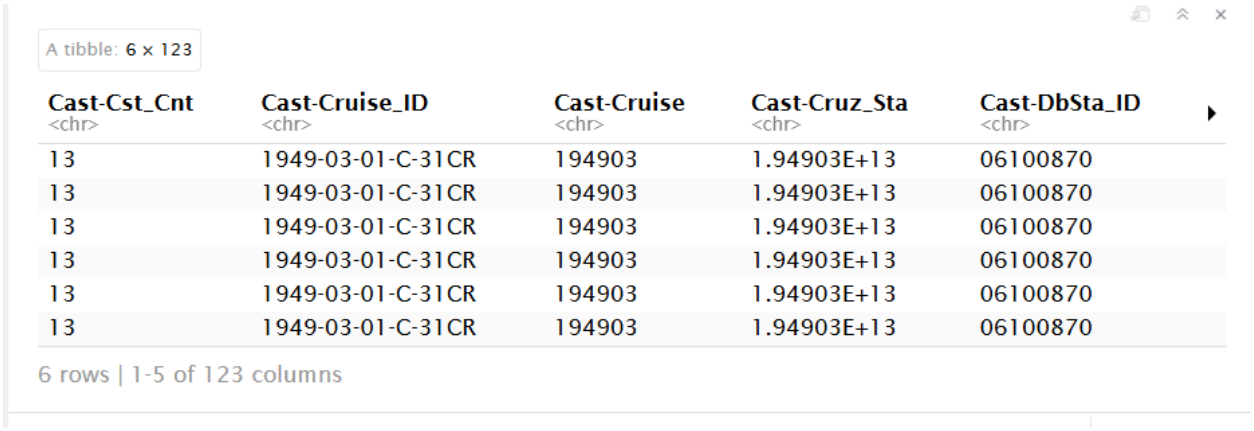
```
#Set the File Path
data_file <-
"C:/Users/afsan/OneDrive/Desktop/Thesis/MERGED_CalCOFI_DATA_CAST_BOTTLE.csv"
```

Since I had an encoding issue reading the file in R, I want to dive deeper into it by checking the field names. I have found out that R cannot read the data, for example, Bottle-pH2. So, I have renamed the name in R. If I rename it manually, it will be tedious, so I renamed the file name in R.

2.2 Reading data from the data file

```
df_raw <- readr::read_csv(
  data_file,
  col_types = readr::cols(.default = readr::col_character()),
  locale = readr::locale(encoding = "Latin1"),
  show_col_types = FALSE
)

head(df_raw)
```



A tibble: 6 × 123

Cast-Cst_Cnt <chr>	Cast-Cruise_ID <chr>	Cast-Cruise <chr>	Cast-Cruz_Sta <chr>	Cast-DbSta_ID <chr>
13	1949-03-01-C-31CR	194903	1.94903E+13	06100870
13	1949-03-01-C-31CR	194903	1.94903E+13	06100870
13	1949-03-01-C-31CR	194903	1.94903E+13	06100870
13	1949-03-01-C-31CR	194903	1.94903E+13	06100870
13	1949-03-01-C-31CR	194903	1.94903E+13	06100870
13	1949-03-01-C-31CR	194903	1.94903E+13	06100870

6 rows | 1-5 of 123 columns

3. Column Name Standardization & Encoding Issue Fix.

3.1. Rename column names

```
# Fix encoding before cleaning names

names(df_raw) <- stringi::stri_enc_toutf8(names(df_raw))
```

3.2. Clean Column names to snake_case

```
# Clean names to snake_case
df_raw <- janitor::clean_names(df_raw)

names(df_raw)
```

```
[1] "cast_cst_cnt"          "cast_cruise_id"
 [3] "cast_cruise"           "cast_cruz_sta"
 [5] "cast_db_sta_id"        "cast_cast_id"
```

[7] "cast_sta_id"	"cast_quarter"
[9] "cast_sta_code"	"cast_distance"
[11] "cast_date"	"cast_year"
[13] "cast_month"	"cast_julian_date"
[15] "cast_julian_day"	"cast_time"
[17] "cast_lat_dec"	"cast_lat_deg"
[19] "cast_lat_min"	"cast_lat_hem"
[21] "cast_lon_dec"	"cast_lon_deg"
[23] "cast_lon_min"	"cast_lon_hem"
[25] "cast_rpt_line"	"cast_st_line"
[27] "cast_ac_line"	"cast_rpt_sta"
[29] "cast_st_station"	"cast_ac_sta"
[31] "cast_bottom_d"	"cast_secchi"
[33] "cast_forel_u"	"cast_ship_name"
[35] "cast_ship_code"	"cast_data_type"
[37] "cast_order_occ"	"cast_event_num"
[39] "cast_cruz_leg"	"cast_orig_sta_id"
[41] "cast_data_or"	"cast_cruz_num"
[43] "cast_int_chl"	"cast_int_c14"
[45] "cast_inc_str"	"cast_inc_end"
[47] "cast_pst_lan"	"cast_civil_t"
[49] "cast_time_zone"	"cast_wave_dir"
[51] "cast_wave_ht"	"cast_wave_prd"
[53] "cast_wind_dir"	"cast_wind_spd"
[55] "cast_barometer"	"cast_dry_t"
[57] "cast_wet_t"	"cast_wea"
[59] "cast_cloud_typ"	"cast_cloud_amt"
[61] "cast_visibility"	"bottle_cst_cnt"
[63] "bottle_btl_cnt"	"bottle_sta_id"
[65] "bottle_depth_id"	"bottle_depthm"
[67] "bottle_t_deg_c"	"bottle_salnty"
[69] "bottle_o2ml_l"	"bottle_s_theta"
[71] "bottle_o2sat"	"bottle_oxy_mol_kg"
[73] "bottle_btl_num"	"bottle_rec_ind"
[75] "bottle_t_prec"	"bottle_t_qual"
[77] "bottle_s_prec"	"bottle_s_qual"
[79] "bottle_p_qual"	"bottle_o_qual"
[81] "bottle_s_thtaq"	"bottle_o2satq"
[83] "bottle_chlor_a"	"bottle_chlqua"
[85] "bottle_phaeop"	"bottle_phaqua"
[87] "bottle_po4u_m"	"bottle_po4q"
[89] "bottle_si_o3u_m"	"bottle_si_o3qu"
[91] "bottle_no2u_m"	"bottle_no2q"
[93] "bottle_no3u_m"	"bottle_no3q"

[95] "bottle_nh3u_m"	"bottle_nh3q"
[97] "bottle_c14as1"	"bottle_c14a1p"
[99] "bottle_c14a1q"	"bottle_c14as2"
[101]"bottle_c14a2p"	"bottle_c14a2q"
[103]"bottle_dark_as"	"bottle_dark_ap"
[105]"bottle_dark_aq"	"bottle_mean_as"
[107]"bottle_mean_ap"	"bottle_mean_aq"
[109]"bottle_inc_tim"	"bottle_light_p"
[111]"bottle_r_depth"	"bottle_r_temp"
[113]"bottle_r_sal"	"bottle_r_dynht"
[115]"bottle_r_nuts"	"bottle_r_oxy_mol_kg"
[117]"bottle_dic1"	"bottle_dic2"
[119]"bottle_ta1"	"bottle_ta2"
[121]"bottle_p_h1"	"bottle_p_h2"
[123]"bottle_dic_quality_comment"	

3.2 Clean field names to the proper format so that R can read the columns.

```
# Clean names
df_raw <- df_raw %>%
  mutate(across(where(is.character), ~ stringi::stri_enc_toutf8(.x)))
head(df_raw)
```

A tibble: 6 × 123

cast_cst_cnt <chr>	cast_cruise_id <chr>	cast_cruise <chr>	cast_cruz_sta <chr>	cast_db_sta_id <chr>
13	1949-03-01-C-31CR	194903	1.94903E+13	06100870
13	1949-03-01-C-31CR	194903	1.94903E+13	06100870
13	1949-03-01-C-31CR	194903	1.94903E+13	06100870
13	1949-03-01-C-31CR	194903	1.94903E+13	06100870
13	1949-03-01-C-31CR	194903	1.94903E+13	06100870
13	1949-03-01-C-31CR	194903	1.94903E+13	06100870

Right now, I have fixed the encoding issue. I want to read the *CALCOFI DataSets Codebook*.

In this, I am leveraging the fact that I have filtered out some unnecessary fields, which are entitled to Quality Code, and keeping only the records where Quality Code is blank(“Data is acceptable”), 4(“Measurement is zeroed out because it falls below the detection limit”), 6(“Value is derived from a CTD sensor”), or 9(“Data is missing”). I have dropped 8 because it is entitled to “Technician suspects the measurement.”

4.1 Read Calcofi DataSets Codebook.xlsx dataset.

```
df_excel <- read_excel("C:/Users/afsan/OneDrive/Desktop/Thesis/CalCofi DataSets CodeBook.xlsx", sheet = "Sheet1")

head(df_excel)
```

A tibble: 6 × 4

Cast\Bottle Fields <chr>	Units <chr>
[Cast-Cst_Cnt]	N/A
[Cast-Cruise_ID]	N/A
[Cast-Cruise]	N/A
[Cast-Cruz_Sta]	N/A
[Cast-DbSta_ID]	N/A
[Cast-Cast_ID]	N/A

6 rows | 1-2 of 4 columns

4.2 Get Quality Code column names from *df_excel*

```
#Check the names

names(df_excel)
```

```
[1] "Cast\\Bottle Fields"    "Units"
[3] "Description"            "CalCOFi Datset Example"
```

4.3 Extract Quality Code Field Names

```
quality_cols <- df_excel %>%
 filter(str_detect(Description, "Quality Code")) %>%
 pull(`Cast\\Bottle Fields`) %>%
 gsub("^\\[|\\]$", "", .) %>%      # remove brackets
 janitor::make_clean_names()       # match df_raw cleaned names


quality_cols
```

```
 [1] "bottle_t_qual"  "bottle_s_qual"  "bottle_p_qual"
 [4] "bottle_o_qual"  "bottle_s_thtaq" "bottle_o2satq"
 [7] "bottle_chlqua"  "bottle_phaqua"  "bottle_po4q"
[10] "bottle_si_o3qu" "bottle_no2q"    "bottle_no3q"
[13] "bottle_nh3q"    "bottle_c14a1q"  "bottle_c14a2q"
[16] "bottle_dark_aq" "bottle_mean_aq"
```

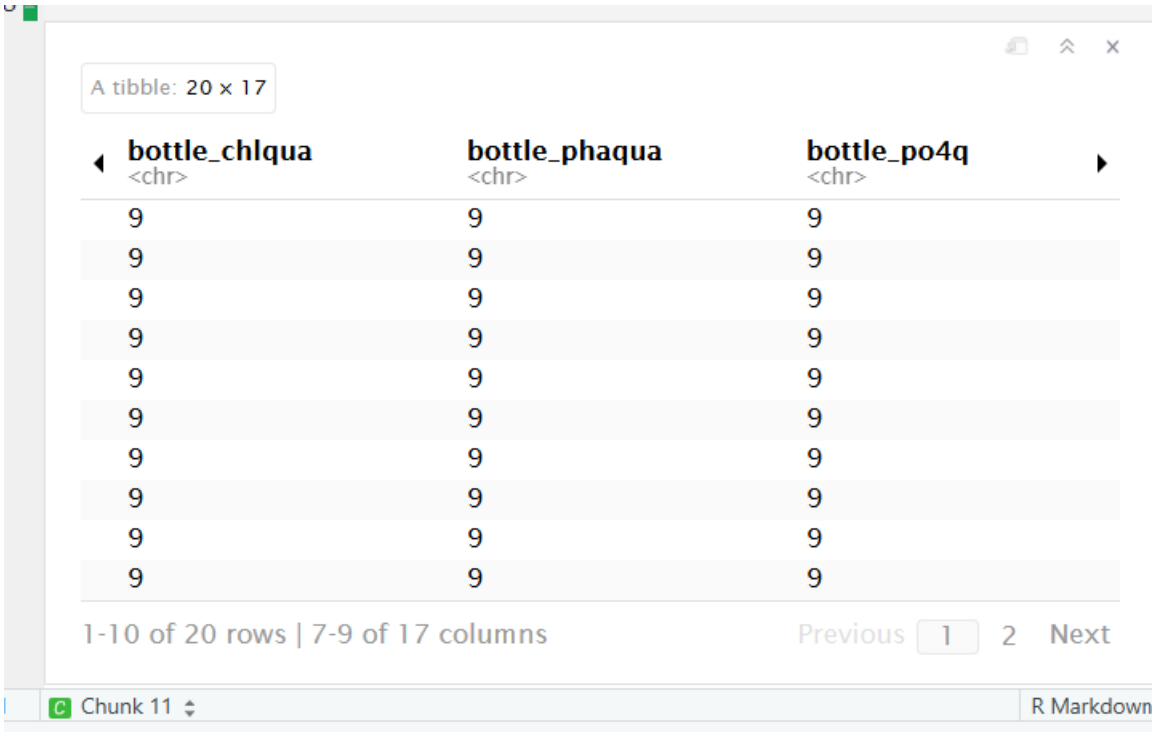
4.4 Keep Only Rows Where "Quality Code" is “Blank” or “4” OR “6” OR “9.”

```
df_filtered <- df_raw %>%
 filter(
   if_all(
    all_of(quality_cols),
    ~ is.na(.x) | .x %in% c(4, 6, 9)))
nrow(df_filtered)
nrow(df_raw)
```

```
[1] 868668
[1] 895371
```

4.4 Check only quality columns (to confirm filter worked)

```
df_filtered %>%
  dplyr::select(all_of(quality_cols)) %>%
  head(20)
```



A tibble: 20 × 17

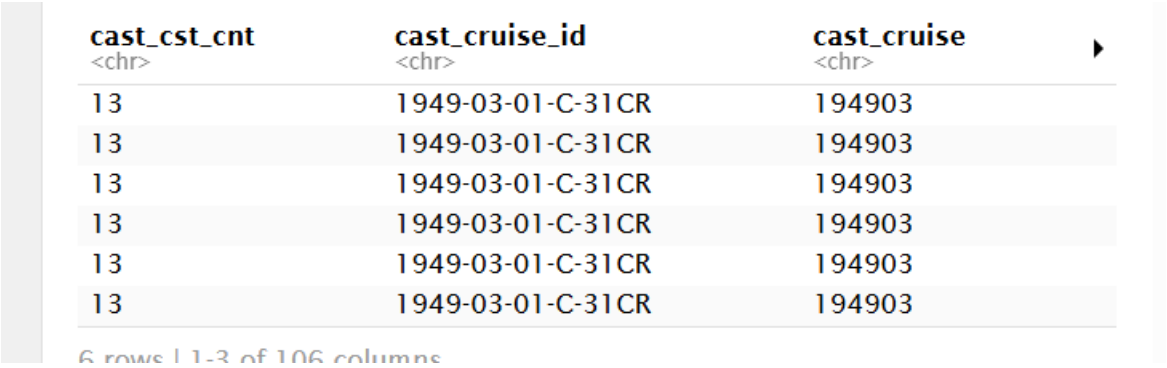
bottle_chlqua <chr>	bottle_phaqua <chr>	bottle_po4q <chr>
9	9	9
9	9	9
9	9	9
9	9	9
9	9	9
9	9	9
9	9	9
9	9	9
9	9	9
9	9	9

1-10 of 20 rows | 7-9 of 17 columns Previous 1 2 Next

Chunk 11 R Markdown

4.5. Remove Quality Columns

```
df_final <- df_filtered %>%
  dplyr::select(-dplyr::all_of(quality_cols))
nrow(df_final)
head(df_final)
```



cast_cst_cnt <chr>	cast_cruise_id <chr>	cast_cruise <chr>
13	1949-03-01-C-31CR	194903
13	1949-03-01-C-31CR	194903
13	1949-03-01-C-31CR	194903
13	1949-03-01-C-31CR	194903
13	1949-03-01-C-31CR	194903
13	1949-03-01-C-31CR	194903

6 rows | 1-3 of 106 columns

4.5 Get rid of bottle_rec_ind since it is also "Quality Code," and bottle_t_prec is “the level of reproducibility or repeatability of temperature measurements.”

```
df_final <- df_final %>%
  dplyr::select(
```

```
  -bottle_rec_ind,
  -bottle_t_prec
 )

# Confirm they are gone
c("bottle_rec_ind", "bottle_t_prec") %in% names(df_final)
```

```r
# Get rid of temperature-related quality fields
df_final <- df_final %>%
  dplyr::select(
    -bottle_rec_ind,
    -bottle_t_prec
  )

# Confirm they are gone
c("bottle_rec_ind", "bottle_t_prec") %in% names(df_final)
```

```
[1] FALSE FALSE
```

5. Correlation and Predictive Analysis of the temperature based on numerical variables (Response vs Predictors)

5.1 Define Response Variable, *bottle_t_deg_c*

```
response <- "bottle_t_deg_c"
response
```

```
[1] "bottle_t_deg_c"
```

5.2. Split into train data, which captures 80% of the data.

```
sp <- group_initial_split(df_final, group = cast_sta_id, prop = 0.8)

sp
```

```
<Training/Testing/Total>
<694963/173705/868668>
```

The data was split by station ID so that the model was trained on the geographic locations and tested on stations different from those used for training. This ensures that the model evaluates temperature prediction across spatial variation rather than within the same station.

5.3. Here is the train dataset, and the following is given to see the top 10 records.

```
train <- training(sp)

head(train)
```

A tibble: 6 × 105

cast_cst_cnt <chr>	cast_cruise_id <chr>	cast_cruise <chr>	cast_cruz_sta <chr>	cast_db_sta_id <chr>
233	1949-05-02-C-31HO	194905	1.94905E+13	03000740
233	1949-05-02-C-31HO	194905	1.94905E+13	03000740
233	1949-05-02-C-31HO	194905	1.94905E+13	03000740
233	1949-05-02-C-31HO	194905	1.94905E+13	03000740
233	1949-05-02-C-31HO	194905	1.94905E+13	03000740
233	1949-05-02-C-31HO	194905	1.94905E+13	03000740

6 rows | 1-5 of 105 columns

5.4. Here is the test dataset, and the following is given to see the top 10 records.

```
test  <- testing(sp)

head(test)
```

cast_cst_cnt <chr>	cast_cruise_id <chr>	cast_cruise <chr>	cast_cruz_sta <chr>	cast_db_sta_id <chr>
15	1949-03-01-C-31CR	194903	1.94903E+13	06200680
15	1949-03-01-C-31CR	194903	1.94903E+13	06200680
15	1949-03-01-C-31CR	194903	1.94903E+13	06200680
15	1949-03-01-C-31CR	194903	1.94903E+13	06200680
15	1949-03-01-C-31CR	194903	1.94903E+13	06200680
15	1949-03-01-C-31CR	194903	1.94903E+13	06200680

6 rows | 1-5 of 105 columns

5.5. Here are the numeric datasets since they were initially denoted as character variables.

```
train_num <- train %>%

  mutate(across(everything(),

                ~ suppressWarnings(as.numeric(.x))))

head(train_num)
```

cast_cst_cnt <dbl>	cast_cruise_id <dbl>	cast_cruise <dbl>	cast_cruz_sta <dbl>	cast_db_sta_id <dbl>
13	NA	194903	1.94903e+13	6100870
13	NA	194903	1.94903e+13	6100870
13	NA	194903	1.94903e+13	6100870
13	NA	194903	1.94903e+13	6100870
13	NA	194903	1.94903e+13	6100870
13	NA	194903	1.94903e+13	6100870

6 rows | 1-5 of 105 columns

```
test_num <- test %>%
 mutate(across(everything(),
            ~ suppressWarnings(as.numeric(.x))))
head(test_num)
```

A tibble: 6 × 105

cast_cst_cnt <dbl>	cast_cruise_id <dbl>	cast_cruise <dbl>	cast_cruz_sta <dbl>	cast_db_sta_id <dbl>
15	NA	194903	1.94903e+13	6200680
15	NA	194903	1.94903e+13	6200680
15	NA	194903	1.94903e+13	6200680
15	NA	194903	1.94903e+13	6200680
15	NA	194903	1.94903e+13	6200680
15	NA	194903	1.94903e+13	6200680

6 rows | 1-5 of 105 columns

5.6 Keep Only Numeric Variables

```
train_num_d <- train_num %>% dplyr::select(where(is.numeric))
nrow(train_num)
nrow(train_num_d)
```

```
[1] 694963
[1] 694963
```

```
test_num_d <- test_num %>% dplyr::select(where(is.numeric))
nrow(test_num)
nrow(test_num_d)
```

```
[1] 173705
[1] 173705
```

It tells us that once we remove non-numeric variables and keep only numeric values, the same results will be obtained.

5.7. Check the data type and descriptive statistics

```
#check the dimensions

dim(train_num_d)

summary(train_num_d$bottle_t_deg_c)
```

```
[1] 694963    104
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   1.44    7.72   10.10   10.81   13.86   99.00    8792
```

It is just telling us that there are 104 variables, 694973 records, with an average value of 10.81, and that the temperature ranges from 1.44 to 99.00 degrees Celsius.

5.8. If the sum of the train_num_d is 0. It means the data hasn't been stored as numeric.

```
sum(complete.cases(train_num_d))
```

```
[1] 0
```

5.9 Remove Very Sparse Columns

Instead of requiring complete cases across all numeric variables, we retain only variables with less than 80% missingness.

```
missing_rate <- colMeans(is.na(train_num_d))

head(missing_rate)
```

```
cast_cst_cnt cast_cruise_id    cast_cruise  cast_cruz_sta cast_db_sta_id
           0              1              0              0              0
cast_cast_id
           1
```

When I see a value of 0, it means that the column has no missing data and is filled with numeric values. When I see a value of 1, it means the column is completely missing (all NA). In my case, columns like cast_cruise_id and cast_cast_id became all NAs because I converted everything to numeric, and since they were originally text, R treated them as missing values. So the "1" does not mean the original data was empty — it means the conversion caused the data to become missing.

5.10. Keep Columns With <80 % Missing

```
good_cols <- names(missing_rate[missing_rate < 0.8])

train_filtered <- train_num_d[, good_cols]

test_filtered  <- test_num_d[, good_cols]

dim(train_filtered)
```

```
[1] 694963     63
```

There are 694,963 records and 63 variables, as I limited the variables based on the proportion of missingness. Variables are limited; I have dropped variables with 80% or more missing records.

5.11 Remove Rows with Missing Values

```
train_clean <- train_filtered[complete.cases(train_filtered), ]

test_clean  <- test_filtered[complete.cases(test_filtered), ]

nrow(train_clean)

nrow(test_clean)
```

```
[1] 141499
[1] 34074
```

5.12. Correlation Analysis

```
numeric_vars <- train_clean %>%
  dplyr::select(where(is.numeric))

cor_matrix <- cor(numeric_vars, use = "complete.obs")

#  Save Full Correlation Matrix
write.csv(
  cor_matrix,
  "calcofi_full_correlation_matrix.csv",
  row.names = TRUE
)

# Extract correlations with temperature
temp_cor <- cor_matrix["bottle_t_deg_c", ]

# Convert to tidy table
```

```
cor_table <- tibble(
  variable = names(temp_cor),
  correlation_with_temperature = round(as.numeric(temp_cor), 3)
) %>%
  arrange(desc(abs(correlation_with_temperature)))

# Save Temperature Correlation Table
write.csv(
  cor_table,
  "calcofi_temperature_correlations.csv",
  row.names = FALSE
)

cor_table
```

Here is the csv file for the correlation table with Temperature.

Correlation with the Temperature of All the Variables

variable	correlation_with_temperature
bottle_t_deg_c	1
bottle_r_temp	1
bottle_s_theta	-0.974
bottle_r_sal	0.974
bottle_po4u_m	-0.887
bottle_no3u_m	-0.882
bottle_o2sat	0.857
bottle_si_o3u_m	-0.833
bottle_depthm	-0.774
bottle_r_depth	-0.774
bottle_oxy_mmol_kg	0.773
bottle_r_oxy_mmol_kg	0.773
bottle_o2ml_l	0.772
bottle_r_dynht	-0.702
bottle_salnty	-0.547
cast_lat_dec	-0.247
cast_lat_deg	-0.245
cast_distance	-0.233
cast_dry_t	0.226
cast_wet_t	0.194

cast_rpt_sta	0.189
cast_st_station	0.189
cast_ac_sta	0.189
cast_db_sta_id	0.145
cast_rpt_line	0.145
cast_int_chl	-0.145
cast_st_line	0.144
cast_ac_line	0.144
bottle_chlor_a	0.14
cast_bottom_d	0.128
cast_month	0.118
cast_julian_day	0.115
cast_quarter	0.114
cast_lon_dec	-0.114
cast_lon_deg	-0.11
bottle_no2u_m	-0.08
cast_cruz_num	0.076
cast_julian_date	-0.05
cast_order_occ	-0.048
cast_cruise	-0.047
cast_cruz_sta	-0.047
cast_year	-0.047
cast_event_num	-0.047
cast_date	-0.044
bottle_btl_cnt	-0.044
cast_cst_cnt	-0.043
bottle_cst_cnt	-0.043
cast_lat_min	0.042
cast_cloud_typ	0.028
cast_wave_prd	-0.027
cast_visibility	0.027
bottle_phaeop	0.026
cast_wind_spd	-0.024
cast_wea	-0.019
cast_barometer	0.012
cast_lon_min	0.01
cast_wind_dir	-0.007
cast_cloud_amt	0.007
cast_wave_dir	-0.004
cast_wave_ht	0.003
bottle_s_prec	-0.001
cast_time	0
cast_data_or	NA

The above correlation results show that temperature is very strongly linked to nutrients and oxygen. Phosphate (r = -0.887), nitrate (r = -0.882), and silicate (r = -0.833) all have strong negative correlations with temperature. This means that when the temperature decreases, nutrient levels increase. In simple terms, colder water contains more nutrients, while warmer water contains fewer nutrients.

Oxygen saturation also has a strong positive correlation with temperature (r = 0.857). This shows that warmer surface waters tend to have higher oxygen saturation values.

These high correlation values (greater than 0.8 in magnitude) indicate strong linear relationships. This confirms that temperature is closely tied to vertical ocean structure, with deeper, colder waters nutrient-rich and surface waters warmer and less nutrient-rich. One note I want to add *is bottle_t_deg_c,* which is the actual water temperature; bottle_r_temp is the reported temperature to the system, which is the same input. I am choosing *bottle_t_deg_c* as the response variable and will not use *bottle_r_temp* or the variables below that have strong or moderate correlations with temperature to perform the predictive analysis.

Variables	Correlation With Temperature	Relationship
bottle_t_deg_c	1	Very positive, strong relationship
bottle_r_temp	1	Very positive, strong relationship
bottle_s_theta	-0.974	Very negative, strong relationship
bottle_r_sal	0.974	Very negative, strong relationship
bottle_po4u_m	-0.887	Very negative, strong relationship
bottle_no3u_m	-0.882	Very negative, strong relationship
bottle_o2sat	0.857	Very positive, strong relationship
bottle_si_o3u_m	-0.833	Very negative, strong relationship
bottle_depthm	-0.774	Very negative, strong relationship
bottle_r_depth	-0.774	Very negative, strong relationship
bottle_oxy_mmol_kg	0.773	Very positive, strong relationship
bottle_r_oxy_mmol_kg	0.773	Very positive, strong relationship
bottle_o2ml_l	0.772	Very positive, strong relationship
bottle_r_dynht	-0.702	Very negative, strong relationship
bottle_salnty	-0.547	Moderate negative relationship

5.12. Select Only Strong Predictors

```
selected_vars <- c(
  "bottle_depthm",
  "bottle_salnty",
```

```
  "bottle_po4u_m",
  "bottle_no3u_m",
  "bottle_si_o3u_m",
  "bottle_o2sat"
)
train_selected <- train_clean[, c(response, selected_vars)]
test_selected  <- test_clean[, c(response, selected_vars)]
```

5.13. Select Only Strong Predictors for Linear Regression

```
selected_vars <- c(
  "bottle_depthm",
  "bottle_salnty",
  "bottle_po4u_m",
  "bottle_no3u_m",
  "bottle_si_o3u_m",
  "bottle_o2sat"
)

train_selected <- train_clean[, c(response, selected_vars)]
test_selected  <- test_clean[, c(response, selected_vars)]
```

5.14 Remove Missing Rows

```
train_selected <- na.omit(train_selected)
test_selected  <- na.omit(test_selected)
```

5.15 Linear Regression

```
m_lm <- lm(bottle_t_deg_c ~ ., data = train_selected)
summary(m_lm)
```

```
Call:
lm(formula = bottle_t_deg_c ~ ., data = train_selected)

Residuals:
     Min       1Q   Median       3Q      Max
-10.5224  -0.6704  -0.0760   0.5055   8.3422

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    -5.361e+01  9.673e-01  -55.42   <2e-16 ***
bottle_depthm  -1.415e-02  1.391e-04 -101.76   <2e-16 ***
bottle_salnty   2.263e+00  2.809e-02   80.55   <2e-16 ***
bottle_po4u_m  -5.496e+00  7.338e-02  -74.90   <2e-16 ***
bottle_no3u_m  -1.313e-01  4.589e-03  -28.62   <2e-16 ***
bottle_si_o3u_m  1.350e-01  2.166e-03   62.35   <2e-16 ***
bottle_o2sat   -4.041e-02  1.032e-03  -39.16   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.156 on 62184 degrees of freedom
Multiple R-squared:  0.8563,    Adjusted R-squared:  0.8562
F-statistic: 6.173e+04 on 6 and 62184 DF,  p-value: < 2.2e-16
```

Adjusted R-squared is 0.8562. The model is highly effective, explaining approximately 85.6% of the variation in water temperature based on physical and chemical properties of the water samples. This indicates that the variables included (depth, salinity, oxygen, etc.) account for nearly 86% of the temperature changes. This is considered a very strong predictive model. Residual Standard Error is 1.156
On average, the model's temperature predictions are off by only 1.16°C. The analysis identified six key factors that significantly influence water temperature. All factors listed in the linear regression analysis are statistically significant (p-value < 2e-16), meaning their impact is very real.

Based on the data, the predictive equation (Linear Regression) to estimate temperature is:

Temperature = -53.61 - (0.01415 * bottle_depthm) + (2.263 * bottle_salnty) - (5.496 * bottle_po4u_m) - (0.1313 * bottle_no3u_m) + (0.135 * bottle_si_o3u_m) - (0.04041 * bottle_o2sat)

However, since predictors like depth and oxygen may be linked together (**multicollinearity**), they may influence or overlap with each other. So I want to check the relationship between them.

5.14. Generate a correlation matrix for the predictors

```
cor_matrix <- cor(train_selected[, selected_vars])
# Save it as a CSV file
write.csv(cor_matrix, "correlation_matrix.csv")
round(cor_matrix, 2) # Rounding to 2 decimals
```

```
                bottle_depthm bottle_salnty bottle_po4u_m bottle_no3u_m bottle_si_o3u_m
bottle_depthm            1.00          0.61          0.78          0.79            0.80
bottle_salnty            0.61          1.00          0.76          0.77            0.81
bottle_po4u_m            0.78          0.76          1.00          0.99            0.97
bottle_no3u_m            0.79          0.77          0.99          1.00            0.98
bottle_si_o3u_m          0.80          0.81          0.97          0.98            1.00
bottle_o2sat            -0.79         -0.77         -0.98         -0.98           -0.96
                bottle_o2sat
bottle_depthm          -0.79
bottle_salnty          -0.77
bottle_po4u_m          -0.98
bottle_no3u_m          -0.98
bottle_si_o3u_m        -0.96
bottle_o2sat            1.00
```

Here is the table:

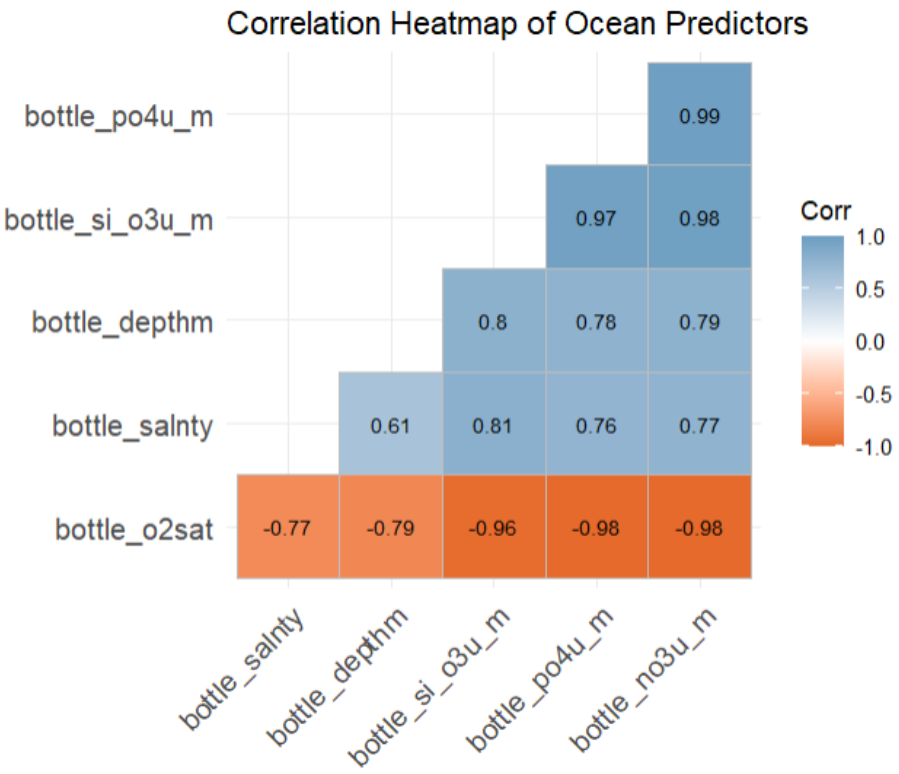
	bottle_depthm	bottle_salnty	bottle_po4u_m	bottle_no3u_m	bottle_si_o3u_m	bottle_o2sat
bottle_depthm	1	0.607801456	0.78193768	0.79390634	0.802234441	-0.792446364
bottle_salnty	0.607801456	1	0.758882501	0.773589036	0.808990045	-0.774098236
bottle_po4u_m	0.78193768	0.758882501	1	0.994294056	0.973348469	-0.982438548
bottle_no3u_m	0.79390634	0.773589036	0.994294056	1	0.975909206	-0.980479738
bottle_si_o3u_m	0.802234441	0.808990045	0.973348469	0.975909206	1	-0.96478152
bottle_o2sat	-0.792446364	-0.774098236	-0.982438548	-0.980479738	-0.96478152	1

I want to show it as a heat map.

5.15. Create the heat map.

```
ggcorrplot(cor_matrix,
           hc.order = TRUE,       # Groups similar variables together
           type = "lower",        # Shows only the bottom half
           lab = TRUE,            # Adds the actual numbers to the squares
```

```
        lab_size = 3,          # Size of the numbers
        colors = c("#E46726", "white", "#6D9EC1"), # Red for negative, Blue for
positive
        title = "Correlation Heatmap of Ocean Predictors",
        ggtheme = theme_minimal())
```



The above correlation analysis reveals a very strong interdependence between all oceanographic predictors. The nutrient variables (Phosphate, Nitrate, and Silicate) exhibit near-perfect positive correlations ranging from 0.97 to 0.99, suggesting they are highly likely to provide redundant information. Furthermore, these nutrients show a nearly perfect inverse relationship with Oxygen Saturation (approx. -0.98). These patterns are primarily driven by Water Depth, with greater depth consistently associated with higher nutrient concentrations and lower oxygen levels. Due to this high level of multicollinearity, we should consider selecting a single representative variable—such as Nitrate or Depth to avoid statistical instability. So, I want to do ridge regression.

In our oceanographic data, ridge regression may be the solution to the "redundancy" problem we identified in our correlation matrix. Standard linear regression (OLS) breaks down when variables like nutrients and depth are highly correlated. Ridge Regression

"connects" to our analysis by allowing us to keep all those variables while keeping the model stable.

5.14 Ridge Regression

```
#  Prepare the data (X = predictors, y = target variable, temperature)

x <- as.matrix(train_selected[, selected_vars])

y <- train_selected$bottle_t_deg_c

# 2. Find the best penalty value (lambda) using cross-validation

cv_model <- cv.glmnet(x, y, alpha = 0) # alpha=0 means Ridge

best_lambda <- cv_model$lambda.min

# 3. Fit the final Ridge model

ridge_model <- glmnet(x, y, alpha = 0, lambda = best_lambda)

# 4. See which ocean variables are most important after the shrinkage

coefficient (ridge_model)
```

```
7 x 1 sparse Matrix of class "dgCMatrix"
                         s0
(Intercept)     -63.468151235
bottle_depthm    -0.010852472
bottle_salnty     2.339193144
bottle_po4u_m    -1.657054146
bottle_no3u_m    -0.094967823
bottle_si_o3u_m  -0.009852528
bottle_o2sat      0.017423267
```

Now I want to do R- square of the ridge regression,

5.16. R- squared for Ridge Regression

```
# 1. Get the model's predictions the data
predictions <- predict(ridge_model, s = best_lambda, newx = x)

# 2. Calculate the Sum of Squares (The math for R-squared)
actual <- y
rss <- sum((predictions - actual)^2)  # Residual Sum of Squares
tss <- sum((actual - mean(actual))^2) # Total Sum of Squares

# 3. The Final R-squared
```

```
r_squared <- 1 - (rss / tss)
print(paste("The R-squared value is:", round(r_squared, 4)))
```

```
[1] "The R-squared value is: 0.832"
```

The above-the-Ridge analysis shows how to handle high multicollinearity (high overlap) in the ocean data. It performs coefficient shrinkage, which pushes redundant variables like Nitrate and Silicate nearly to zero, so they don't “double count” the same information. This process identifies Salinity (+2.34) and Phosphate (-1.66) as the most influential predictors, creating a much more stable and reliable model for Temperature than standard regression. It is also telling that the model explains 83% of the variance in the data.

The final Ridge regression equation for the model is:

Temperature = -63.4682 - (0.0109 * bottle_depthm) + (2.3392 * bottle_salnty) - (1.6571 * bottle_po4u_m) - (0.0950 * bottle_no3u_m) - (0.0099 * bottle_si_o3u_m) + (0.0174 * bottle_o2sat)

However, I want to compare who is better for this data model based on the analysis.

5.17. Compare Linear Regression Vs Ridge Regression

```
# Linear Regression Prediction
# Using the test data model
pred_lm <- predict(m_lm, newdata = test_selected)

# Ridge Regression Prediction
# Ridge needs the data as a matrix
x_test <- as.matrix(test_selected[, selected_vars])
pred_ridge <- as.numeric(predict(ridge_model, newx = x_test, s = best_lambda))

# The Comparison
results_comparison <- data.frame(
  Actual = test_selected$bottle_t_deg_c,
  Linear_Error = abs(test_selected$bottle_t_deg_c - pred_lm),
  Ridge_Error  = abs(test_selected$bottle_t_deg_c - pred_ridge)
)

# Calculate Average Error (MAE)
```

```
# If Ridge_MAE is lower, Ridge is better.
summary_metrics <- results_comparison %>%
  summarise(
    Linear_MAE = mean(Linear_Error),
    Ridge_MAE  = mean(Ridge_Error)
  )

print(summary_metrics)
```

```
  Linear_MAE Ridge_MAE
1  0.8995836 0.9624537
```

In our analysis of the CalCOFI dataset, we compared Standard Linear Regression and Ridge Regression to see which model generalizes better to unseen test data. The results show that the Standard Linear Regression is more accurate, achieving a Mean Absolute Error (MAE) of 0.899, compared to the Ridge model's MAE of 0.962. This 0.06°C difference indicates that the regularization penalty in the Ridge model was unnecessary, as the relationship between ocean chemistry and temperature is very stable in this dataset. Because the standard model has a lower error rate on the test set, the Standard Linear Regression equation is the correct model to use for this study, as it captures the strongest predictive signals without the added bias of the Ridge approach. The right equation for this project is given below:

Temperature = -53.61 - (0.01415 * bottle_depthm) + (2.263 * bottle_salnty) - (5.496 * bottle_po4u_m) - (0.1313 * bottle_no3u_m) + (0.135 * bottle_si_o3u_m) - (0.04041 * bottle_o2sat)

6. Scatter Plot of the Linear Regression

```
# Prepare the data
plot_lm_final <- data.frame(
  Actual = test_selected$bottle_t_deg_c,
  Predicted = as.numeric(pred_lm)
)

# Create the Plot
ggplot(plot_lm_final, aes(x = Actual, y = Predicted)) +
  # Points
  geom_point(alpha = 0.3, color = "#6D9EC1", size = 1.2) +
```

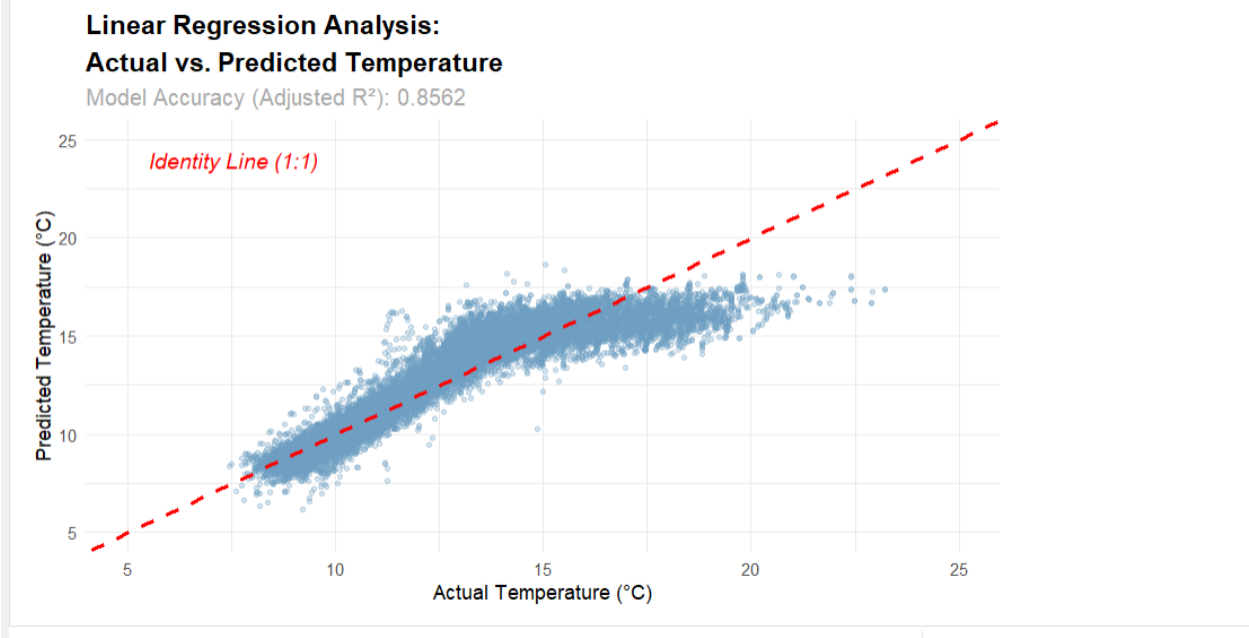
```
# The 1:1 Identity Line
geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "red", size = 1) +

# This allows the plot to stretch and fill the RStudio window
scale_x_continuous(limits = c(5, 25)) +
scale_y_continuous(limits = c(5, 25)) +

# Better label placement (Top-Left)
annotate("text", x = 5.5, y = 24, label = "Identity Line (1:1)",
         color = "red", fontface = "italic", size = 4, hjust = 0) +

#  Wrap title so it doesn't cut off
labs(
  title = "Linear Regression Analysis:\nActual vs. Predicted Temperature",
  subtitle = "Model Accuracy (Adjusted R\u00B2): 0.8562",
  x = "Actual Temperature (\u00B0C)",
  y = "Predicted Temperature (\u00B0C)"
) +

theme_minimal() +
theme(
  plot.title = element_text(face = "bold", size = 14, lineheight = 1.2),
  plot.subtitle = element_text(size = 12, color = "darkgray"),
  axis.title = element_text(size = 11)
)
```



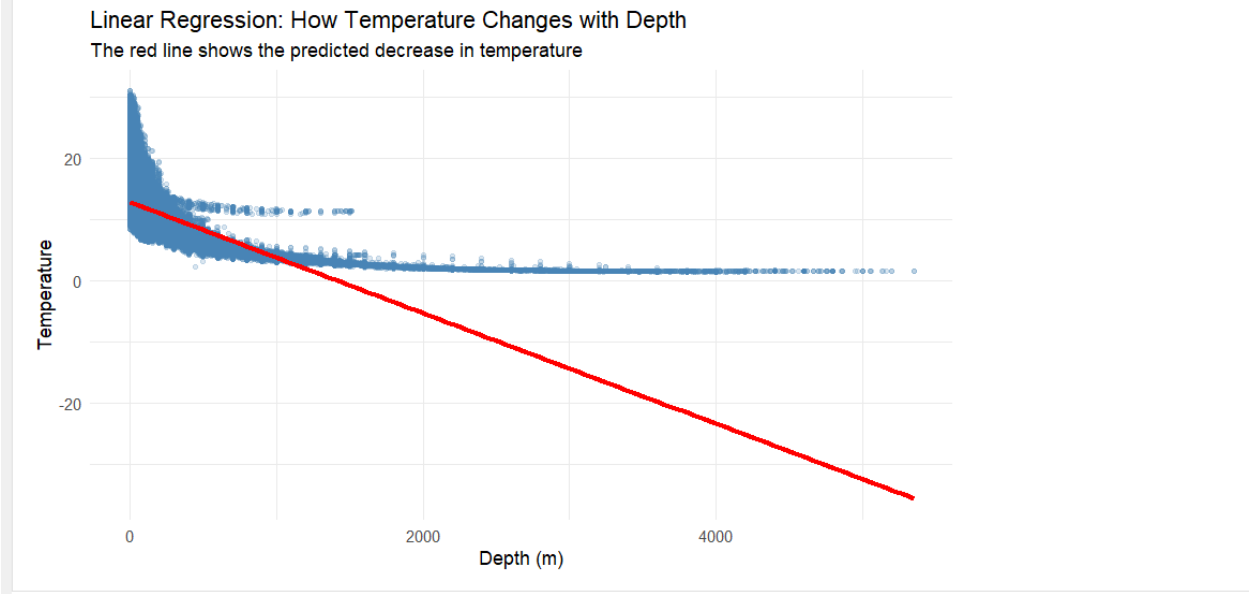
The scatter plot shows how well the computer model predicts ocean temperatures compared to the real-world measurements. The red dashed line represents a good prediction where the "Actual" and "Predicted" temperatures are exactly the same. For cooler temperatures between 5°C and 15°C, the blue dots stay very close to this line, which means the model is doing a great job. However, once the water exceeds 15°C, the blue dots flatten out and fall below the red line. This tells us that the model has estimated a lower temperature than the actual temperature. Even though the model has a strong accuracy score of 0.8562, this graph proves that it is much more reliable for cold water than for warm water.

7. Explore the regression of Temperature based on depth.

```
# 1. Ensure we use the cleaned data (removing those 100°C errors)
ocean_data_filtered <- train %>%
  mutate(
    temp = as.numeric(bottle_t_deg_c),
    depth = as.numeric(bottle_depthm)
  ) %>%
  filter(temp < 35, !is.na(temp), !is.na(depth))

# 2. Plot Depth vs. Temperature with a Linear Regression line
ggplot(ocean_data_filtered, aes(x = depth, y = temp)) +
  # Add the raw data points (using alpha to handle the thousands of dots)
  geom_point(alpha = 0.2, color = "steelblue", size = 1) +
  # Add the Linear Regression Line in Red
```

```
  geom_smooth(method = "lm", color = "red", size = 1.2) +
  labs(title = "Linear Regression: How Temperature Changes with Depth",
       subtitle = "The red line shows the predicted decrease in temperature",
       x = "Depth (m)",
       y = "Temperature") +
  theme_minimal()
```



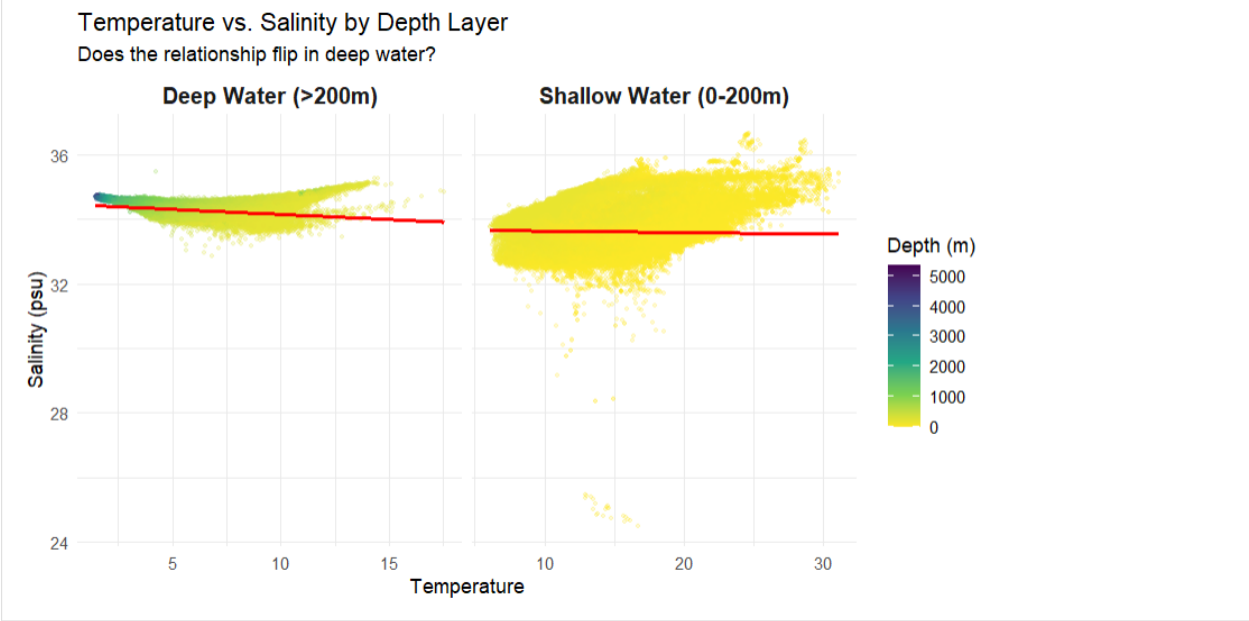
This image shows underfitting, where a simple straight line (the red model) is too rigid to follow the natural curve of the ocean data (the blue dots). While the actual temperature drops quickly and then levels off nearby. The linear model assumes a constant drop, leading it to predict "impossible" freezing temperatures in the deep sea. Analytically, this visual mismatch shows that a single-variable linear model is too simple for this dataset, which helps explain why the final report uses a multilinear approach with 6 variables to achieve 85.6% accuracy.

8. Explain Temperature Based on Salinity and depth.

```
# 1. Prepare the data with "Depth sizes"
ocean_data_faceted <- train %>%
 mutate(
   temp = as.numeric(bottle_t_deg_c),
   sal = as.numeric(bottle_salnty),
   depth = as.numeric(bottle_depthm),
   # Create a category to split the data at 200 meters
```

```
  depth_group = ifelse(depth <= 200, "Shallow Water (0-200m)", "Deep Water (>200m)")
 ) %>%
 # Filter out errors and missing values
 filter(temp < 35, !is.na(temp), !is.na(sal), !is.na(depth_group))

# 2. Create the Faceted Plot
ggplot(ocean_data_faceted, aes(x = temp, y = sal, color = depth)) +
 geom_point(alpha = 0.2, size = 0.8) +
 # Add a trend line to EACH panel
 geom_smooth(method = "lm", color = "red", se = FALSE) +
 # This line creates the two side-by-side windows
 facet_wrap(~ depth_group, scales = "free_x") +
 scale_color_viridis_c(direction = -1, name = "Depth (m)") +
 labs(title = "Temperature vs. Salinity by Depth Layer",
      subtitle = "Does the relationship flip in deep water?",
      x = "Temperature",
      y = "Salinity (psu)") +
 theme_minimal() +
 theme(strip.text = element_text(face = "bold", size = 12)) # Make panel titles bold
```



The above shows that in the deep ocean (>200 m), the water system is closed and governed strictly by the laws of density. This filters out environmental noise and forces a high correlation between variables; water must be cold and salty to remain heavy enough to stay

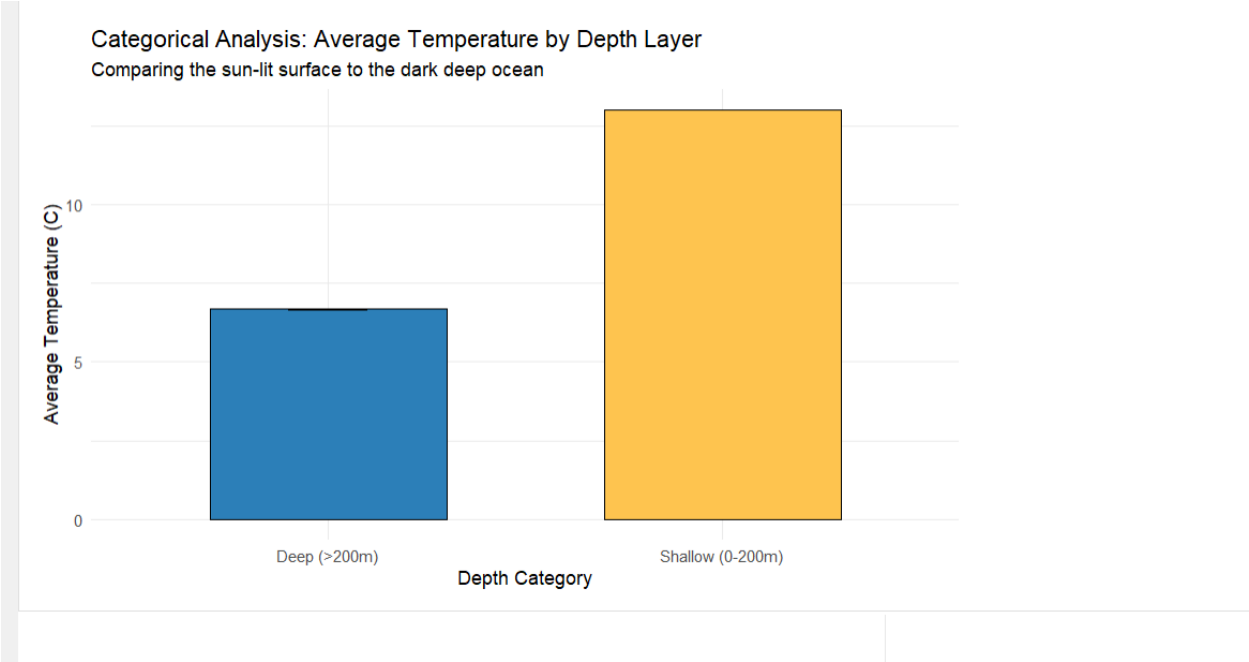
at depth. The shallow water of the ocean is an open system dominated by atmospheric "noise" (sun and rain). This decouples temperature and salinity, resulting in the high-entropy, disorganized data cloud seen in the above plot. The relationship doesn't "flip”; it stabilizes. The analysis proves that depth acts as a quality-control filter, moving the ocean from chaos to order.

9. Categorical Analysis: In this section, we move beyond simple numbers to see if the relationship between Temperature and Salinity changes depending on the **Category** of water. By splitting our data into groups such as 'Shallow vs. Deep' or 'Summer vs. Winter,' we can determine whether a single linear model is sufficient or whether the ocean comprises multiple distinct 'chemical worlds.' We use **ANOVA** to statistically validate these group differences.

```
# 1. Prepare the categorical data
depth_summary <- train %>%
  mutate(
    temp = as.numeric(bottle_t_deg_c),
    depth_group = ifelse(as.numeric(bottle_depthm) <= 200, "Shallow (0-200m)", "Deep
(>200m)")
  ) %>%
  filter(!is.na(temp), !is.na(depth_group), temp < 40) %>%
  group_by(depth_group) %>%
  summarise(
    mean_temp = mean(temp),
    se = sd(temp) / sqrt(n()) # Standard Error for the bars
  )

# 2. Create the Bar Chart
ggplot(depth_summary, aes(x = depth_group, y = mean_temp, fill = depth_group)) +
  geom_bar(stat = "identity", color = "black", width = 0.6) +
  geom_errorbar(aes(ymin = mean_temp - se, ymax = mean_temp + se), width = 0.2) +
  scale_fill_manual(values = c("Deep (>200m)" = "#2c7fb8", "Shallow (0-200m)" = "#fec44f"))
+
  labs(title = "Categorical Analysis: Average Temperature by Depth Layer",
       subtitle = "Comparing the sun-lit surface to the dark deep ocean",
       x = "Depth Category",
```

```
    y = "Average Temperature (C)") +
  theme_minimal() +
  theme(legend.position = "none")
```



The data confirms that water temperature decreases with depth. The surface layer is nearly twice as warm as the deep ocean layer shown in this analysis. So I started to dive deeper and wanted to conduct a categorical analysis.

10. ANOVA Test

```
# 1. Prepare the data for the test
anova_data <- train %>%
  mutate(
    temp = as.numeric(bottle_t_deg_c),
    depth_group = ifelse(as.numeric(bottle_depthm) <= 200, "Shallow", "Deep")
  ) %>%
  filter(!is.na(temp), !is.na(depth_group))

# 2. Run the ANOVA (Comparing Temperature by Depth Group)
temp_anova <- aov(temp ~ depth_group, data = anova_data)

# 3. View the results
summary(temp_anova)
```

```
                Df  Sum Sq Mean Sq F value Pr(>F)
depth_group      1 6311414 6311414  715208 <2e-16 ***
Residuals   685931 6053053       9
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

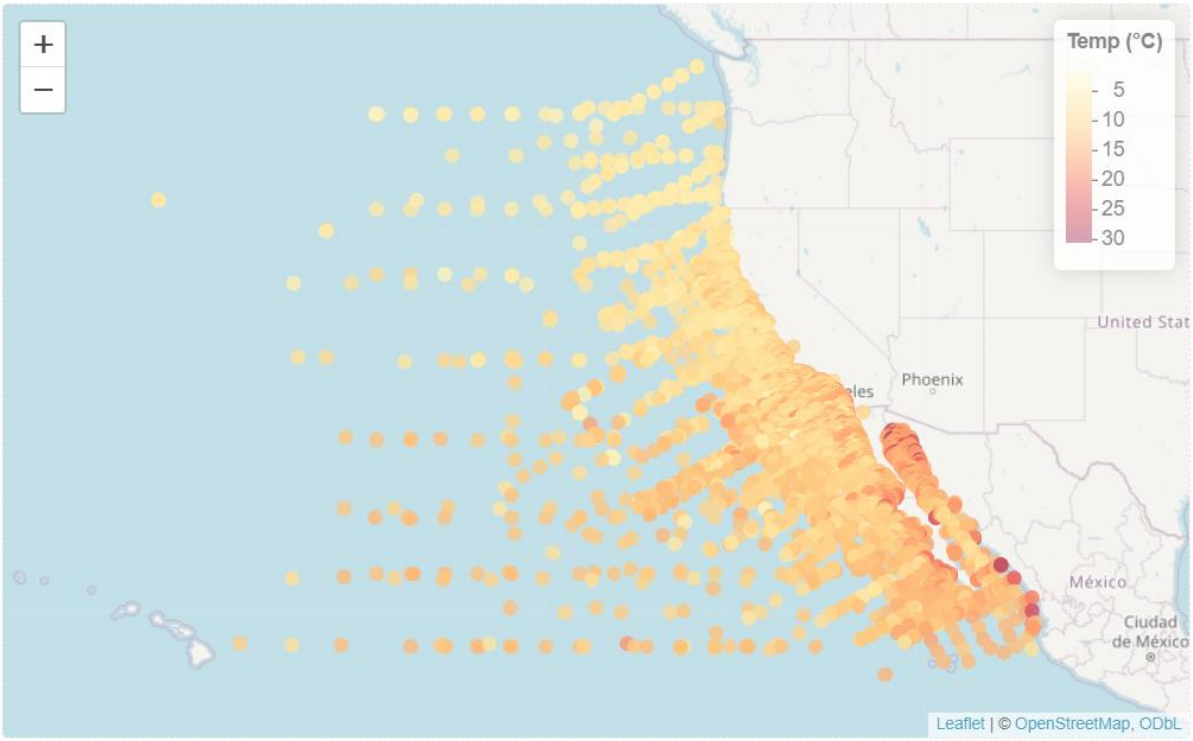
The ANOVA test provides the final mathematical proof for this study. With a p-value of nearly zero and a massive F-value of 715,208, we can conclude that the transition at 200m is not a coincidence. This analysis confirms that the ocean is physically divided into two separate worlds: a chaotic surface and a highly organized depth.

11. Geographic Analysis:

```
# 1. Clean the data and prepare for the map
map_data <- train %>%
  mutate(
    lat = as.numeric(cast_lat_dec),
    lon = as.numeric(cast_lon_dec),
    temp = as.numeric(bottle_t_deg_c)
  ) %>%
  # Filter out missing coordinates and extreme temperature outliers
  filter(!is.na(lat), !is.na(lon), !is.na(temp), temp < 40) %>%
  group_by(cast_sta_id, lat, lon) %>%
  summarise(avg_temp = mean(temp), .groups = "drop")

# 2. Create a color palette (Blue for cold, Red for hot)
pal <- colorNumeric(palette = "YlOrRd", domain = map_data$avg_temp)

# 3. Build the Leaflet map
leaflet(map_data) %>%
  addTiles() %>%  # Adds the standard street map background
  addCircleMarkers(
    lng = ~lon, lat = ~lat,
    radius = 5,
    color = ~pal(avg_temp),
    stroke = FALSE, fillOpacity = 0.8,
    # This popup appears when  we click a dot
    popup = ~paste0("<b>Station ID:</b> ", cast_sta_id,
                    "<br><b>Avg Temp:</b> ", round(avg_temp, 2), "\u00B0C")
  ) %>%
  addLegend(pal = pal, values = ~avg_temp, title = "Temp (\u00B0C)")
```



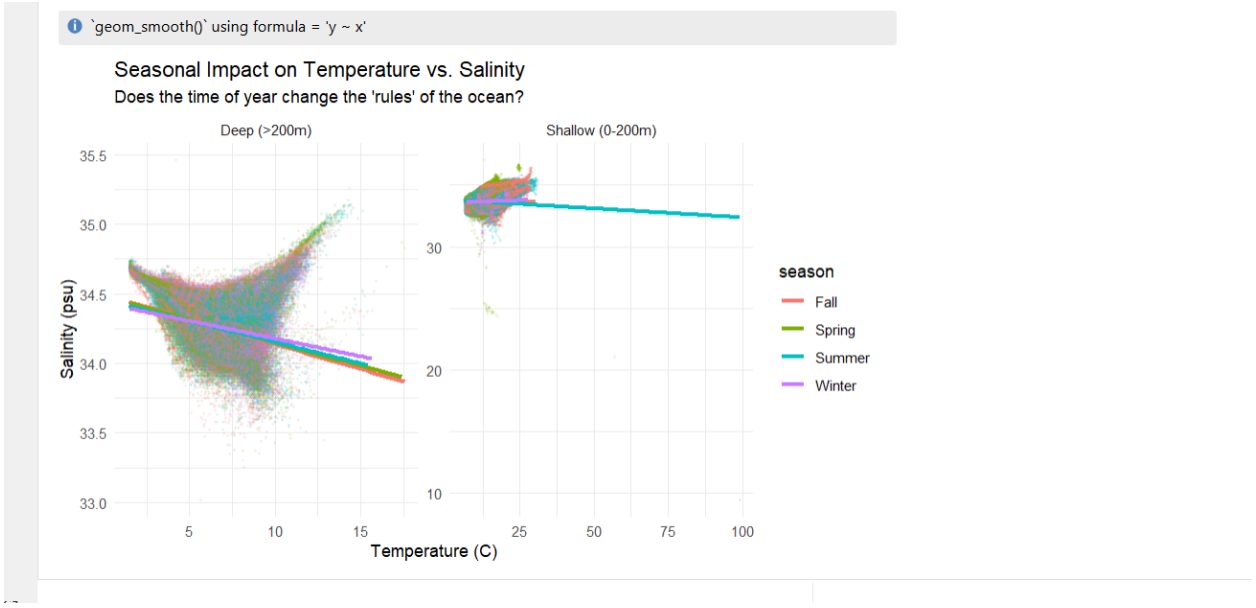
The above images show that Water in the South starts with higher temperatures (25-20 degrees Celsius), which creates more "noise" in the data than in the cooler North (10-15 degrees Celsius). The **Leaflet map** shows a clear thermal gradient from North to South.

12. Seasonal Impact:

```
# Time Seasonal
# 1. Create the Season column and filter for depth layers
seasonal_data <- train %>%
  mutate(
    # Convert types for calculation
    temp = as.numeric(bottle_t_deg_c),
    sal = as.numeric(bottle_salnty),
    month = as.numeric(cast_month),
    depth = as.numeric(bottle_depthm),
    # Assign seasons based on month number
    season = case_when(
      month %in% c(12, 1, 2) ~ "Winter",
      month %in% c(3, 4, 5) ~ "Spring",
      month %in% c(6, 7, 8) ~ "Summer",
      month %in% c(9, 10, 11) ~ "Fall"
```

```
  ),
  depth_group = ifelse(depth <= 200, "Shallow (0-200m)", "Deep (>200m)")
 ) %>%
 filter(!is.na(season), !is.na(temp), !is.na(sal))

# 2. Create the Visual Comparison
ggplot(seasonal_data, aes(x = temp, y = sal, color = season)) +
 geom_point(alpha = 0.1, size = 0.5) +
 geom_smooth(method = "lm", se = FALSE, size = 1.2) +
 facet_wrap(~ depth_group, scales = "free") +
 labs(title = "Seasonal Impact on Temperature vs. Salinity",
      subtitle = "Does the time of year change the 'rules' of the ocean?",
      x = "Temperature (C)", y = "Salinity (psu)") +
 theme_minimal()
```



The deep ocean shows how the trend lines for all four seasons (Fall, Spring, Summer, Winter) nearly overlap. This proves that the deep ocean is **thermally isolated**. It doesn't "know" what month it is. Whether it's the middle of Summer or the dead of Winter, the relationship between temperature and salinity stays the same. The **Shallow (0-200m)** data clusters for each season are more distinct. We can see the colors (seasons) shifting along the temperature scale.

13. Result:

Here is the actual analysis Result given below:

ID	Question	Answer
Q1	How strongly does temperature change with depth?	There is a strong negative correlation (r = -0.774), indicating that temperature decreases significantly with depth.
Q2	Is salinity related to temperature?	Yes, the analysis identifies a moderate negative relationship (r = -0.547) between salinity and temperature.
Q3	Does depth mostly explain the salinity-temperature relationship?	Yes. Oceanographic patterns are primarily driven by water depth, which in turn influences temperature, nutrient levels, and oxygen levels.
Q4	Does adding depth improve prediction accuracy?	Yes. Depth is identified as one of the six key factors with a statistically significant impact (p-value < 2e-16).
Q5	Do oxygen/ammonia/nutrients improve prediction beyond salinity+depth?	Yes. Including these chemical properties allows the model to explain approximately 85.6% of the variation in temperature.
Q6	Are predictors overlapping too much (multicollinearity)?	Yes. Nutrient variables exhibit near-perfect positive correlations ranging from 0.97 to 0.99, indicating high redundancy.
Q7	If overlap is high, should Ridge be used?	Ridge was tested to handle redundancy, but Standard Linear Regression was chosen as it achieved a lower MAE (0.899).
Q8	If we use all available variables, do we predict better?	Not necessarily all variables can predict better. The datasets have many missing records. After data mining, we extracted 6 variables. Using 6 variables together creates a very strong predictive model with an Adjusted R-squared of 0.8562.
Q9	What are the final findings?	Standard Linear Regression is the most accurate model (MAE: 0.899), though it tends to under-predict at temperatures above 15°C.
Q10	Does the relationship between temperature and salinity change as we go deeper?	Depth acts as a "quality-control filter" that moves the ocean from chaos to order.
Q11	Does where the water is located change its thermal behavior?	Geography determines the "starting state" of the water mass.

Q12	Does the time of year change the "rules" of the ocean?	Only at the surface. The deep ocean is "timeless." Seasonal Plot shows that trend lines for all four seasons nearly overlap in the deep ocean, proving it is thermally isolated. In the shallow water, however, the data clusters shift significantly as the seasons change