Universitat Autònoma de Barcelona, Departament de Física

Treball de fi de grau de Física

On Tropical-Cyclones
A Statistical Analysis in a Warming Environment

Alfredo Hernández Cavieres

Supervised by: Álvaro Corral

Academic tutor: Diego Pavón

July 2017

Dedicated to RMT, Val & the Brotherhood

Abstract

This thesis describes the physical and statistical nature of tropical-cyclones (TC) in an environment of increasing sea surface temperature. The influence of global warming on the intensity of TCs is a rather controversial topic that has already been addressed in many statistical studies. The goal of this text is to replicate the results obtained by Corral et al. in [1] from scratch as a learning process, to revise them using updated hurricanes track and sea surface temperature (SST) databases, and to do some new statistical analyses regarding the influence of the SST on the intensity and duration of TCs.

Given the nature of this study, the methodology is heavily rooted in programming using a language specialised in statistical computing, along with analytical description of the maths and physics behind the TCs and the SSTs. Our results clearly show the effects of global warming on TC occurrence, in accordance with the results previously found by Corral et al.

Contents

1. Introduction

With the intention of quantifying the consequences of global warming, in this thesis we describe the physical and statistical nature of tropical-cyclones in order to introduce a statistical analysis based on the application of the power dissipation index (PDI), defined in § 2.1, to individual tropical-cyclones.

Ultimately, our goal is to replicate the results obtained by Corral et al. in [1] from scratch, following the very process described in the paper, using updated hurricanes track and sea surface temperature (SST) databases to see if the tendency of increasing temperatures implies any significant change in the results found in [1] (from § 2.2 to § 4.1).

Apart from this, in § 4.2, we perform a statistical analysis regarding the influence of the SST, or lack thereof, on the intensity and duration of tropical-cyclones.

As the entirety of the behind-the-scenes calculations, plots, and analyses are done using a programming language specific for statistical computing, for the sake of showing a real representation of the work done in this thesis, we include in the Appendix A all the code developed structured in *base* scripts that include only functions, and a `main_analysis.R` script that calls all the functions to perform all the appropriate computations.

2. Preparing tropical-cyclones data

2.1. Power dissipation index (PDI)

To characterise the intensity of a tropical-cyclone one needs to define a physically relevant measure of released energy. In [2] Emanuel showed that in a tropical-cyclone energy dissipation occurs mostly in the atmospheric surface layer, and that the corresponding dissipation rate per unit area, D, is

$$D \equiv C_D \rho \|\vec{v}\|^3 \tag{1}$$

where C_D is the surface drag coefficient, ρ is the surface air density, $\|\vec{v}\|$ is the magnitude of the surface wind velocity.

Thus, integrated over the surface area covered by a circularly symmetric tropical-cyclone of radius r_0 of lifetime τ, the total power dissipated by the storm, PD, is given by

$$PD \equiv 2\pi \int_0^\tau \int_0^{r_0} C_D \rho \|\vec{v}\|^3 r \,\mathrm{d}r \,\mathrm{d}t \tag{2}$$

However, as stated in [3], since the integral in the expression (3a) will in practice be dominated by high wind speeds, one can approximate the product $C_D\rho$ as a constant and define a simplified power dissipation index (PDI) as

$$PDI \equiv \int_0^\tau v_{max}^3 \,\mathrm{d}t \tag{3a}$$

However, the wind data (as we will discuss in § 2.2.3) is recorded every $\Delta t = 6\,\mathrm{h}$. Therefore, we will discretise the expression for the PDI using the rectangle method:

$$PDI = \sum_t v_t^3 \Delta t \tag{3b}$$

where v_t is the maximum sustained surface wind speed at time t.

Using the expression (3b) we can easily calculate the PDI value associated to any tropical-cyclone, which is exactly what we will do in § 2.3.1.

If one wishes to consult a thorough technical review article on tropical-cyclone as a thermodynamic system, [4] by Emanuel would probably be the best source available.

2.2. Hurricane tracks data sets

2.2.1. Description of the database

Although Corral et al. analyse several ocean basins, we will focus only on the North Atlantic (N. Atl.) and the Northeast Pacific (E. Pac.) Oceans. The reason to do this is the abundance of research on these two basins and the precision of the database provided by the National Hurricane Center (NHC) [5]: the HURDAT [6].

Since both basins directly concern USA territories (especially the N. Atl.), the government's efforts on improving the tracking and prediction technologies, routine satellite imagery has been used since as early as the late 1960s.

A major change between our data sets and the ones used in [1] is that the second-generation hurricane database (HURDAT2), has been developed this decade [7]. The improvements of the revised version are mainly:

(i) Inclusion of non-developing tropical depressions.

(ii) Inclusion of systems that were added to the database after the end of each season.

Also, the ongoing post-storm analysis reviews of the tropical-cyclones have revised several storms [8], particularly important in the 1851–1960 era.

2.2.2. Tropical-cyclones classification

Tropical cyclones are classified into three main groups, based on wind intensity: tropical depressions, tropical storms, and a third group of more intense storms, whose name depends on the region. In particular, in the Northeast Pacific or in the North Atlantic, it is called a hurricane. In table 1 we can see a detailed classification of the tropical-cyclones studied in this text.

Category	1–minute sustained winds	
Tropical depression	≤ 33 kn	(≤ 61 km/h)
Tropical storm	34–63 kn	(63–118 km/h)
Category 1 hurricane	64–82 kn	(119–153 km/h)
Category 2 hurricane	83–95 kn	(154–177 km/h)
Category 3 major hurricane	96–112 kn	(178–208 km/h)
Category 4 major hurricane	113–136 kn	(209–253 km/h)
Category 5 major hurricane	≥ 137 kn	(≥ 254 km/h)

Table 1: Tropical-cyclone classification used by the NHC

2.2.3. Importing raw data

The format of the HURDAT2 data sets is documented at [9, 10]. A record of data is recorded once every 6 hours for each storm (although there are additional records for

certain storms). The record is comprised of the date and time, storm identifier, system status (cf. tropical-cyclone category), latitude and longitude of the centre of the storm, the sustained surface wind speed (in knots) observed in the storm, and several other properties we are not interested in.

The data sets used can be downloaded from `http://www.aoml.noaa.gov/hrd/hurdat/hurdat2-1851-2016-apr2017.txt` (N. Atl.) and `http://www.aoml.noaa.gov/hrd/hurdat/hurdat2-nepac-1949-2016-apr2017.txt` (E. Pac.).

Given the systematic format used by the NHC, it's fairly easy to read the raw data using R (a programming language for statistical computing). First of all, we need to split the data into two lists: (i) metadata (`hurr.meta`) containing the values for the storm ID, name, and number of observations for the storm, and (ii) observations (`hurr.obs`) for each storm record:

Snippet 2.1: Read and clean raw data

```
# Read and split raw data ---------------------------------
tracks.file <- paste0("data/", filename)
hurr.tracks <- readLines(tracks.file)
hurr.tracks <- lapply(hurr.tracks, str_split, pattern = ",",
    simplify = TRUE)

# Clean the raw data ---------------------------------------
# Split the hurr.tracks into meta and observation lists
hurr.lengths <- sapply(hurr.tracks, length)
hurr.meta <- hurr.tracks[hurr.lengths == 4]
hurr.obs <- hurr.tracks[hurr.lengths == 21]
```

Then we can convert the data into data frames[1] and select only the observations we are interested in, and rename the columns to easily manipulate the data later:

Snippet 2.2: Create data frames and select observations

```
# Create and clean meta data frame
hurr.meta <- lapply(hurr.meta, tibble::as_tibble)
hurr.meta <- bind_rows(hurr.meta)

hurr.meta <- hurr.meta %>%
  dplyr::select(-V4) %>%
  rename(storm.id = V1, storm.name = V2, n.obs = V3) %>%
  mutate(storm.name = str_trim(storm.name),
         n.obs = as.numeric(n.obs))

storm.id <- rep(hurr.meta$storm.id, times = hurr.meta$n.obs)

# Create and clean obs data frame
hurr.obs <- lapply(hurr.obs, tibble::as_tibble)
```

[1] The concept of a data frame comes from the world of statistical software used in empirical research; it generally refers to "tabular" data: a data structure representing cases (rows), each of which consists of a number of observations or measurements (columns).

```
15 hurr.obs <- bind_rows(hurr.obs) %>%
16   mutate(storm.id = storm.id) %>%
17   dplyr::select(storm.id, V1, V2, V4:V7) %>%
18   rename(date = V1, time = V2, status = V4, lat = V5, long = V6,
         wind = V7)
```

It is incredibly useful to unite the raw date and time values and convert them into a date-time objects to ease manipulation and calculations. We will also change `status` to give the levels meaningful names that can be easily understood without consulting the documentation:

Snippet 2.3: Change date-time and use meaningful status names

```
1 # Change date and time & unite them
2 hurr.obs <- hurr.obs %>%
3   unite(date.time, date, time) %>%
4   mutate(date.time = ymd_hm(date.time))
5
6 # Meaningful status names
7 storm.levels <- c("TD", "TS", "HU", "EX", "SD", "SS", "LO", "WV",
      "DB")
8 storm.labels <- c("Tropical␣depression", "Tropical␣storm",
      "Hurricane", "Extratropical␣cyclone", "Subtropical␣depression",
      "Subtropical␣storm", "Other␣low", "Tropical␣wave", "Disturbance")
9 hurr.obs <- hurr.obs %>%
10   mutate(status = factor(str_trim(status),
11                          levels = storm.levels,
12                          labels = storm.labels))
```

Ultimately, we want numeric values for the latitude and longitude so that we can use them for mapping and for the SST analysis (§ 4.1). Using regular expressions to separate the numeric and non-numeric parts of these columns, it's easy to do this (we won't enter into details; this can be found in the script A.1, in § A).

As commented above, there are additional records for certain storms (mainly from aircraft reconnaissance data), but to replicate the methodology used in [1], we will just ignore them. We also found that a couple of storms had a middle missing value (`NA` in R, meaning *not available*), so assuming the continuous behaviour of a tropical-cyclone, using Bolzano's Theorem we manually changed them to the corresponding intermediate value:

Snippet 2.4: Clean non-standard data and odd middle values for certain storms

```
1 # Clean non-standard data ----------------------------------
2 # Ignore data outside the delta.t = 6 hours
3 hurr.obs <- hurr.obs %>%
4   filter(hour(date.time) == 00 |
5            hour(date.time) == 06 |
6            hour(date.time) == 12 |
7            hour(date.time) == 18) %>%
8   filter(minute(date.time) == 00)
9
10 # Clean up wind column -------------------------------------
```

```
# Manually change odd middle values for AL191976 & AL111973
hurr.obs <- hurr.obs %>%
  mutate(wind = ifelse(storm.id == "AL191976" & wind == "␣-99", 20,
      wind),
         wind = ifelse(storm.id == "AL111973" & wind == "␣-99", 30,
            wind),
         wind = ifelse(storm.id == "AL111973" & month(date.time) ==
            9 & day(date.time) == 12 & hour(date.time) == 12, NA,
            wind)) %>%
  filter(is.na(wind) != TRUE)
```

One thing to take into consideration is that, even though the use of new technologies improve the best tracks, the uncertainties of the observations are non-trivial. Using aircraft and satellite monitoring data the relative uncertainty of wind speeds falls around 15 % for tropical storms, ~ 10 % for category 1 and 2 hurricanes, and ~ 8 % for major hurricanes [7].

The inability to work with observational error data does not mean, however, that we won't be able to assign an error to our results. In § 2.3.3 we derive the statistical error of the PDI probability density, whilst for the SST analysis in § 4.1 we will use the standard error of the mean.

2.2.4. Resulting data structure

In table 2 we show the structure of the `hurr.natl.obs` data frame to illustrate the variables we use in the study (by all means, `hurr.epac.obs` shares the same structure), as well as the format (data type[2]) of the observational record data.

storm.id <chr>	storm.name <chr>	n.obs <int>	date.time <dttm>	status <fctr>	lat <dbl>	long <dbl>	wind <dbl>	storm.year <dbl>
AL011851	UNNAMED	13	1851-06-25 00:00:00	Hurricane	28.0	-94.8	80	1851
AL011851	UNNAMED	13	1851-06-25 06:00:00	Hurricane	28.0	-95.4	80	1851
AL011851	UNNAMED	13	1851-06-25 12:00:00	Hurricane	28.0	-96.0	80	1851
AL011851	UNNAMED	13	1851-06-25 18:00:00	Hurricane	28.1	-96.5	80	1851
AL011851	UNNAMED	13	1851-06-26 00:00:00	Hurricane	28.2	-97.0	70	1851
AL011851	UNNAMED	13	1851-06-26 06:00:00	Tropical storm	28.3	-97.6	60	1851

Table 2: Excerpt of the `hurr.natl.obs` data frame

2.2.5. Activity windows

Even though recent improvements have been made to the HURDAT2 database, [8, 9, 10], following the methodology of Corral et al., we will intentionally limit this study to the satellite era.

[2]In computer science and computer programming, a data type or simply type is a classification of data which tells the compiler or interpreter how the programmer intends to use the data.

In [11], Webster et al. go into more details about the activity windows for the hurricane tracks data as well as the sea surface temperature used by researchers in the past. In table 3 we can see a summary of the spatial and temporal activity windows we will use for each basin based on the information available in the previously mentioned papers; we also include the amount tropical-cyclones analysed in the text, N, as well as the size of the entire data set, N_{tot}.

Basin	Years	Season	Longitude	Latitude	N	N_{tot}
N. Atl.	1966–2016	June–October	90°W–20°E	5°N–25°N	771	1756
E. Pac.	1966–2016	June–October	120°W–90°W	5°N–20°N	920	1071

Table 3: Spatial and temporal activity windows for each basin

Below we can see a map for each basin (figures 1 and 2) showing all the storms analysed in the text in red lines, and the spatial window highlighted in green. These maps have been generated by using the function `map_region_hurrs()` defined in the script A.1 in § A.

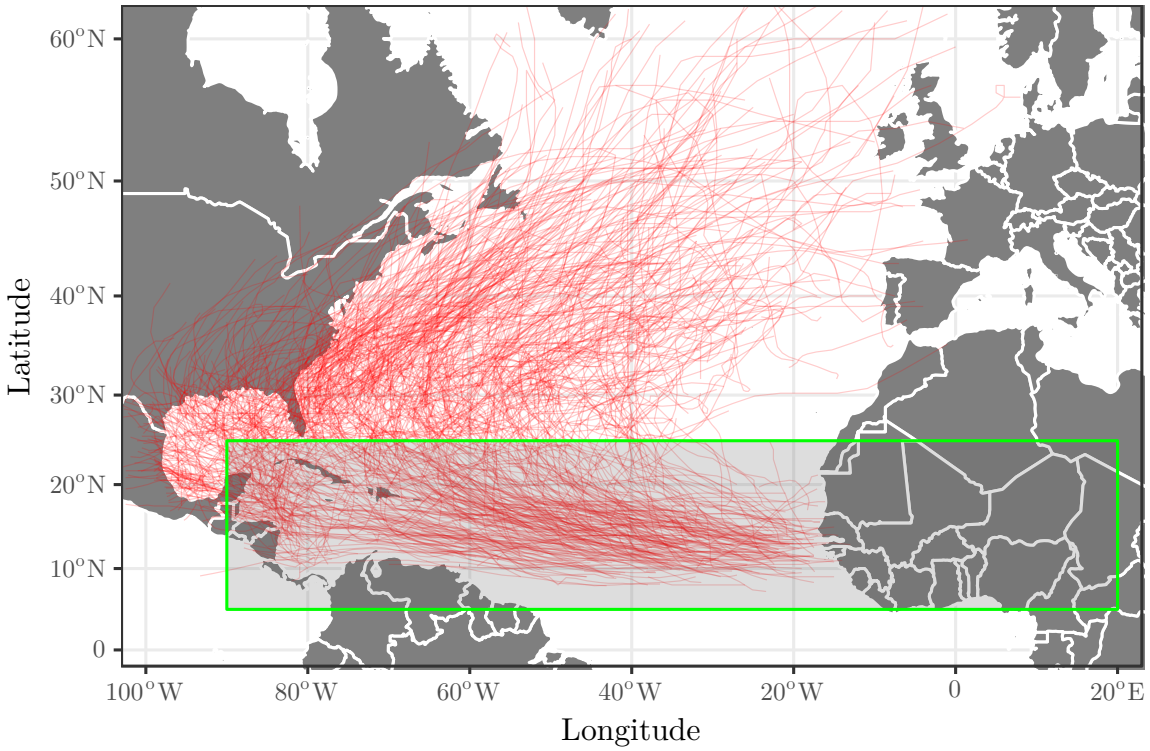

Figure 1: Tropical-cyclones best tracks for the North Atlantic Ocean

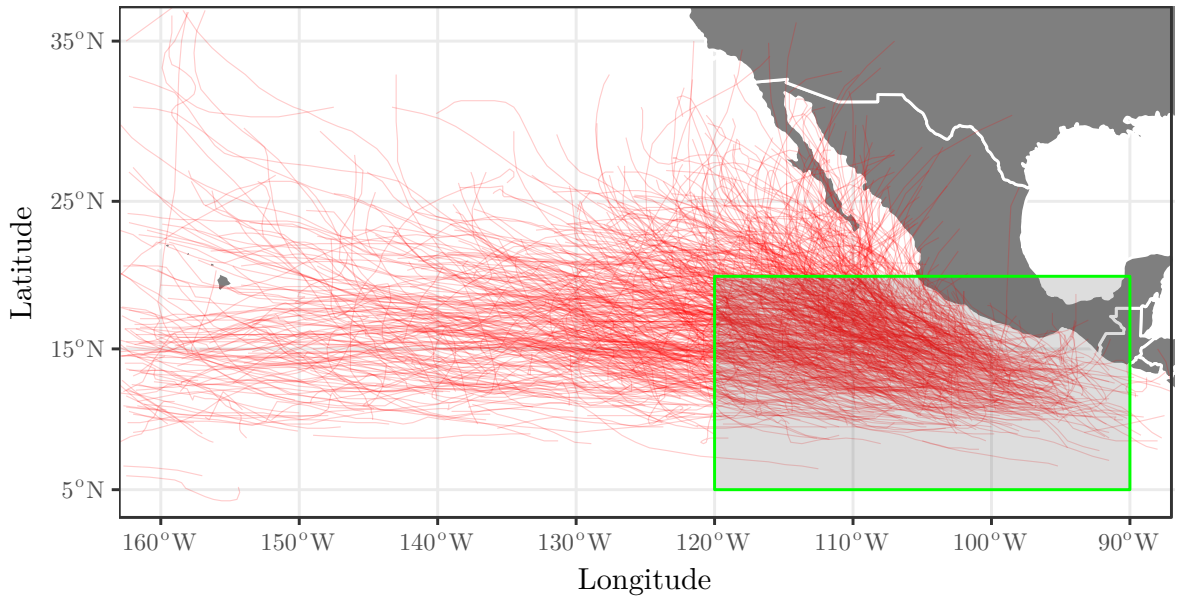

Figure 2: Tropical-cyclones best tracks for the Northeast Pacific Ocean

2.3. PDI analysis

2.3.1. PDI calculation

After reading and cleaning the raw HURDAT2 data sets, calculating the PDI using (3b) is a straightforward process. To accomplish this, we created a function that transforms an observation data frame (`hurr.obs`) into a new PDI data frame (`hurr.obs.pdi`) by grouping all the observation records by storm ID and summarising the product $v_t^3 \Delta t$, and the maximum sustained wind speed during the storm's lifetime (this will allow us to filter tropical-cyclones by category). An important thing to take into account here is that, since the wind speed data is expressed in knots, we need to manually convert them into the International System of Units ($1\,\mathrm{kn} = 0.514\,\mathrm{m\,s^{-1}}$):

Snippet 2.5: Function to calculate a PDI data frame

```
get_pdis <- function(hurr.obs){
  hurr.obs.pdi <- hurr.obs %>%
    group_by(storm.id, storm.name, n.obs) %>%
    summarise(storm.pdi = sum(conv_unit(wind, "knot", "m_per_sec")^3
       * conv_unit(6, "hr", "sec")),
              max.wind = max(wind)) %>%
    mutate(storm.duration = n.obs * conv_unit(6, "hr", "sec")) %>%
    mutate(storm.year = substring(storm.id, 5, 9)) %>%
    filter(storm.pdi != "NA") %>%
    filter(storm.pdi != 0)
  hurr.obs.pdi <- hurr.obs.pdi[c("storm.id", "storm.name", "n.obs",
      "storm.duration", "storm.pdi", "max.wind", "storm.year")]
  return(hurr.obs.pdi)
}
```

An important aspect to consider about the hurricane season of the tropical-cyclones is that they do not always start and end in the same year (e.g., the hurricane season for the Southwest Pacific Ocean is December–April [11]). Although this is not our case (as depicted in table 3), for the sake of reproducibility and completeness, we read the storm year directly from the storm ID instead of the `date.time`; doing this would result in having two entries for such cases if we did `group_by(storm.id, storm.name, n.obs, date.time)`.

2.3.2. Resulting data structure

In table 4 we show the structure of the `hurr.natl.pdi` data frame to illustrate the variables we use in the study, as well as the data type of the observational PDI data. Naturally, `hurr.epac.pdi` has the same data structure.

storm.id <chr>	storm.name <chr>	n.obs <int>	storm.duration <dbl>	storm.pdi <dbl>	max.wind <dbl>	storm.year <chr>
AL112016	JULIA	33	712800	3984450402	45	2016
AL122016	KARL	54	1166400	10872230381	60	2016
AL132016	LISA	30	648000	4626653083	45	2016
AL142016	MATTHEW	47	1015200	176267909828	145	2016
AL152016	NICOLE	62	1339200	60328453629	120	2016
AL162016	OTTO	36	777600	17968073735	100	2016

Table 4: Excerpt of the `hurr.natl.pdi` data frame

One of the things we can do right out of the gate after reading and processing the HURDAT2 data sets is to focus into one storm, calculate its PDI, and compare it to the results found by Corral et al. to make sure we are on the right track and haven't made any mistakes thus far. In particular, Hurricane Katrina (2005) is selected in [1] as example to illustrate the calculation of the PDI, so we will use it as well to do our own calculations.

First of all, we need a function to get the PDI value of any given storm specifying its name and year of occurrence:

Snippet 2.6: Function to get PDI of a single storm

```
get_pdi <- function(hurr.obs.pdi, storm, year){
  wanted.pdi.df <- hurr.obs.pdi %>%
    filter(storm.name == toupper(storm)) %>%
    filter(storm.year == year)
  wanted.pdi.df$storm.pdi
}
```

To visually illustrate the physical meaning of the PDI (an integral, at the end of the day), it's really helpful to plot the individual wind speed records of the storm against time. One of the advantages of using R, is that we can trivially use different colours to show the storm status at very points of its lifetime:

Snippet 2.7: Function to track a single storm

```
track_storm <- function(hurr.obs = hurr.all.obs, storm, year){
  ggplot(hurr.obs %>%
           filter(storm.name == toupper(storm)) %>%
           filter(storm.year == year),
         aes(x = date.time, y = wind)) +
    geom_line(linetype = "dotted") +
    geom_point(aes(colour = status)) +
    labs(title = bquote(.(storm) ~ profile ~ .(paste0("(", year,
       "),") ) ~ PDI == .(scientific(get_pdi(hurr.all.pdi, storm,
       year), digits = 3)) ~ m^3 ~s^-2),
         x = "Time␣(days)", y = "Wind␣speed␣(kt)", colour = "Status")
}
```

In figure 3 we can see the result of calling `track_storm(hurr.natl.obs, "Katrina", 2005)`.

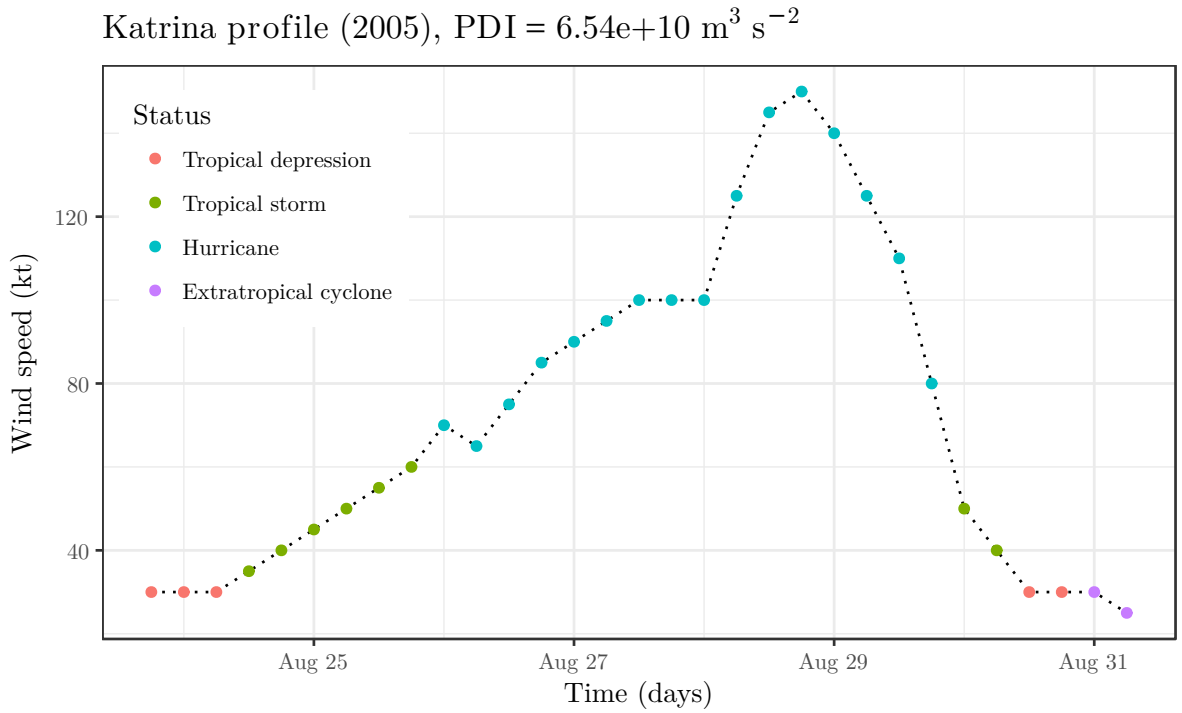

Figure 3: Katrina (`AL122005`) sustained surface wind speed profile

The results found here are exactly the same found by Corral et al. In fact, Katrina is a good example of why removing additional records (as stated in § 2.2.3) is essential to replicate their methodology and results. If we did the calculations by dynamically calculating the Δt to take into account all the revised recorded data for each storm, which was our initial implementation, we would get $PDI = 6.62 \times 10^{10}\,\mathrm{m^3\,s^{-2}}$ instead of $PDI = 6.54 \times 10^{10}\,\mathrm{m^3\,s^{-2}}$. And this is not even the most extreme case; Katrina only has 3 additional records, whereas some storms have considerably more.

Since some storms are `UNNAMED`, we have also created the `track_storm_by_id()` and `get_pdi_by_id()` functions that do basically the same that standard functions defined above, but only require the storm ID. These two functions can be found in the script A.2, in § A).

One may wonder why we do not just stick to these `*_by_id()` functions. The reason to define two separate functions is that in any realistic scenario, tracking storms or getting their PDI by their name and year of occurrence is way faster and more intuitive for a researcher not familiarised with the Automated Tropical Cyclone Forecast (ATCF) cyclone number assignment system; in such cases `*_by_id()` would only act as a fallback.

2.3.3. Probability density

Since the PDI is a variable quantity, it is convenient to work with its probability density, $D(PDI)$. Let us recall that a probability density (in this case of tropical-cyclones PDIs) is defined as the probability that the value of the PDI is in a narrow interval,

$$[PDI,\, PDI + \mathrm{d}PDI)$$

normalized to the total number of occurrences, N, and divided by the length of the interval, $\mathrm{d}PDI$, so that $\int_0^\infty D(PDI)\,\mathrm{d}PDI \equiv 1$. That being so, the $D(PDI)$ is mathematically given by

$$D(PDI) \equiv \frac{P[PDI \leq \text{value} < PDI + \mathrm{d}PDI]}{\mathrm{d}PDI} \approx \frac{n(PDI)}{N\,\mathrm{d}PDI} \tag{4}$$

In [12] Hergarten states that for variables distributed across a wide range of scales it is convenient to use a logarithmic binning instead of a linear binning, meaning that the bin widths increase accordingly to the scale of the variable. The main advantage of doing this is that the problem of low numbers of objects per bin at large sizes is less severe. Statistically, using a logarithmic binning means that for the k-th interval,

$$\mathrm{d}PDI = c^{k-1}(c-1)m \tag{5}$$

where c is a factor related to the size of each bin, and m is an absolute minimum PDI value. In this way, the value of $D(PDI)$ is associated to the whole interval

$$PDI_{min} = c^{k-1}m \leq PDI < PDI_{max} = c^k m \tag{6}$$

Corral et al. have taken $c = \sqrt[5]{10} = 1.58$, which corresponds to 5 intervals per decade, and $m = 10^8\,\mathrm{m^3\,s^{-2}}$. We have, however, taken the actual absolute minimum value for each basin, which is $m = 1.84 \times 10^8\,\mathrm{m^3\,s^{-2}}$ for both of them. It's important to note here that we have removed two clear outlying storms after doing a quick analysis of the PDI data frames of both basins:

Basin	Storm ID	PDI ($\mathrm{m^3\,s^{-2}}$)
North Atlantic	`AL171988`	1.25×10^8
Northeast Pacific	`EP231989`	2.35×10^7

Table 5: Outlying storms judging by their PDI value

Since we want to plot the $D(PDI)$, it's useful to choose a single point of the interval, $PDI^\star$, to be representative of the probability density. The exact calculation of $PDI^\star$ is fairly complex, as it would require to know the mathematical form of the probability distribution beforehand. We know, however, that geophysical systems are generally

described by a Pareto power law distribution [12]. Therefore, a reasonable estimation of $PDI^\star$ is the geometric mean of the limits of the interval:

$$PDI^\star = \sqrt{PDI_{min} PDI_{max}} \tag{7}$$

The error bars for each interval are estimated from the standard deviation of the probability density:

$$\epsilon(PDI) \equiv \epsilon_{rel}(PDI) D(PDI) \tag{8a}$$

where the relative error follows a binomial distribution:

$$\epsilon_{rel}(PDI) = \sqrt{\frac{1-P}{n}} \approx \frac{1}{\sqrt{n}} \tag{8b}$$

The calculation of the PDI probability density looks quite complex, but it's really not that complicated to do using R. The process consists in creating an empty data frame containing the values of PDI_{min} and PDI_{max} for each individual bin following (5), being `pdi.bin` $\equiv \mathrm{d}PDI$ the size of each bin. The computation of $PDI^\star$, `ndpdi` $= n$, and `length(hurr.obs.pdi$storm.pdi)` $= N$ is straightforward. Once we have calculated those, the values of `dpdi` $= D(PDI)$ and `pdi.error` $= \epsilon(PDI)$ are calculated using (4) and (8a), respectively:

Snippet 2.8: Function to calculate the $D(PDI)$ in a range of years

```
get_dpdi <- function(hurr.obs.pdi, years){
  hurr.obs.pdi <- hurr.obs.pdi %>%
    filter(storm.year %in% years)
  c <- 10^(1/5)
  m <- min(hurr.obs.pdi$storm.pdi)
  dpdi.df <- data.frame()
  for (i in 1:20) {
    dpdi.df <- rbind(dpdi.df, c(c^(i-1)*m, c^(i)*m))
  }
  colnames(dpdi.df) <- c("pdi.min", "pdi.max")
  dpdi.df <- dpdi.df %>%
    mutate(pdi.star = sqrt(pdi.min * pdi.max),
           pdi.bin = pdi.max - pdi.min) %>%
    rowwise() %>%
    mutate(ndpdi = sum(hurr.obs.pdi$storm.pdi >= pdi.min &
                       hurr.obs.pdi$storm.pdi <= pdi.max),
           dpdi = ndpdi/(length(hurr.obs.pdi$storm.pdi)*pdi.bin),
           pdi.error = dpdi / sqrt(ndpdi) ) %>%
    filter(dpdi != 0)
  return(dpdi.df)
}
```

Note that since we do not know the absolute maximum PDI value in advance, we create a few more bins than necessary, giving us the following range:

$$PDI \in [m,\, c^{20} m) = [1.84 \times 10^8,\, 1.84 \times 10^{12})\,\mathrm{m^3\,s^{-2}}$$

After doing the calculations previously described, we just need to delete the rows (bins) with no value for n (or $D(PDI)$ to be more precise). It's important to notice that this short routine won't affect the compile times in any significant way.

The `plot_dpdi()` function that allows us to plot the $D(PDI)$ against the PDI using this logarithmic binning is defined in the script A.2, in § A). In figures 4 and 5 we can see probability density analysis for the N. Atl. and E. Pac. basins for the 1966-2016 period.

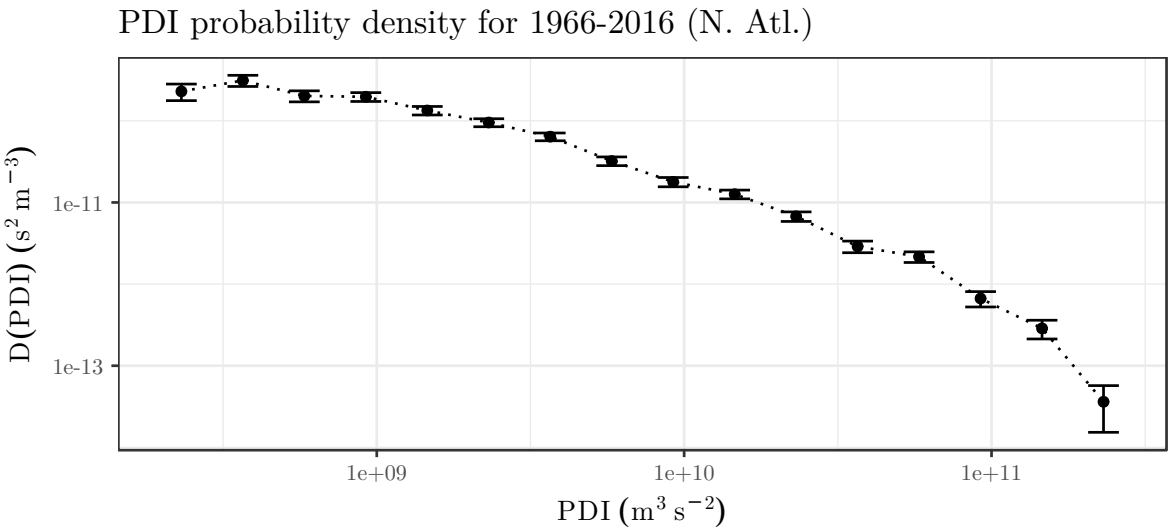

Figure 4: $D(PDI)$ distribution for the North Atlantic Ocean

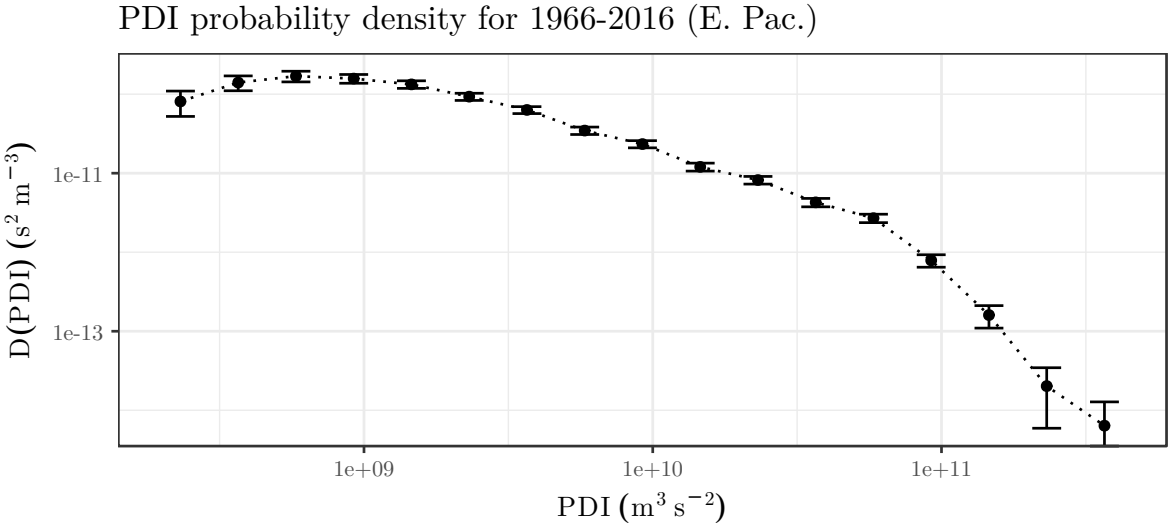

Figure 5: $D(PDI)$ distribution for the Northeast Pacific Ocean

As we can see, both distributions are essentially the same, being the end of the tail the only remarkable difference. It is apparent how the tail has a longer range when the size

of the region is increased (the North Atlantic Ocean is about three times bigger than the Northeast Pacific Ocean). In [1] Corral et al. go into a fair amount of detail on the effect of the finite size of basins, both from a physical and a statistical point of view, showing that the finite size delimits the evolution and lifetime of hurricanes.

What we want to focus on, however, is the causal relationship between increasing hurricane intensity and increasing sea surface temperature (SST) proposed by Trenberth in [13]. In the text Trenberth states that higher SSTs are associated with increased water vapour in the lower troposphere; both higher SSTs and increased water vapour tend to increase the energy available for atmospheric convection and the energy available to tropical-cyclones as a consequence.

The hypothesis is, therefore, that separating the PDI data by low-SST and high-SST years, we should get two similar $D(PDI)$ distributions with one major difference: high-SST years should have a longer tail on account of having more available energy from the sea. For this separation (or classification) process we will follow the methodology used by Corral et al. in [1], which is a variation of the methodology proposed by Webster et al. in [11].

3. Preparing sea surface temperature (SST) data

3.1. SST data set

3.1.1. Description of the database

There are several sea surface temperature (SST) databases, with different time-steps (e.g., daily, weekly, monthly, and so on), domains (i.e., global or specific regions), and data resolutions; each used for different analyses of climatological nature [14, 15].

The Met Office [16] Hadley Centre's sea ice and sea surface temperature database, HadISST1 [17], is a unique combination of monthly globally complete fields of SST and sea ice concentration on a latitude-longitude grid from 1871. In figure 6 we can see a sample from the data set to illustrate the grid structure. This map has been generated using the function `map_global_sst()` defined in the script A.3 in § A.

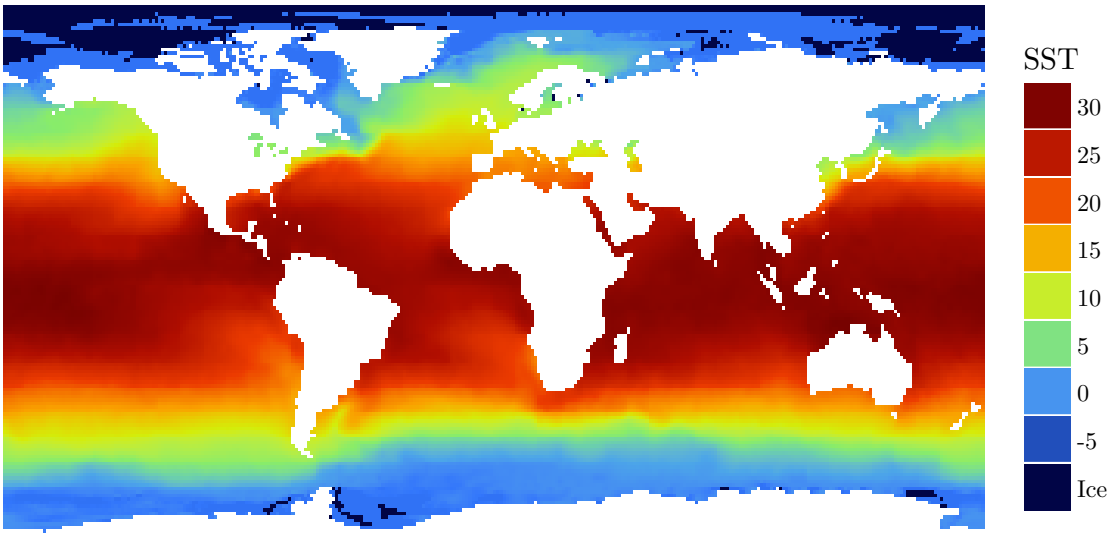

Figure 6: Global SST (in °C) map from December 2015

Although there is a revised HadISST.2 database [18], we will use the HadISST1 database, as it is the one used in Corral et al.'s, Webster et al.'s and several other authors's climate analyses.

The main reason to do so, however, is that HadISST.2 contains more ocean grid boxes and introduces a different method to calculate the monthly temperatures, making it quite incompatible with HadISST1 (as opposed to the revised HURDAT2 database, which is just an improved version of the old database).

3.1.2. Importing and working with raw data

The format of the HadISST1 database is documented at [19]. The particularities of the this database (in ASCII format) are the following:

- Temperatures are stored as °C × 100 (all values are integers).
- 100% sea-ice-covered grid-boxes are flagged as `-1000`.
- Land squares are set to `-32768`.

Reading ASCII data, however, can be particularly slow and tedious, as there are several data sets per decade (1870–2003 era) and per year (2004–present era). Thankfully, the HadISST data are also available in netCDF format, which is constructed using raster data instead of just text.

A raster brick consists of a matrix of cells (or pixels) organised into a grid where each cell contains a value representing information, such as temperature in our case. Each matrix can also be comprised of layers (as illustrated in figure 7); in the HadISST1 database, each matrix layer represents a different month.

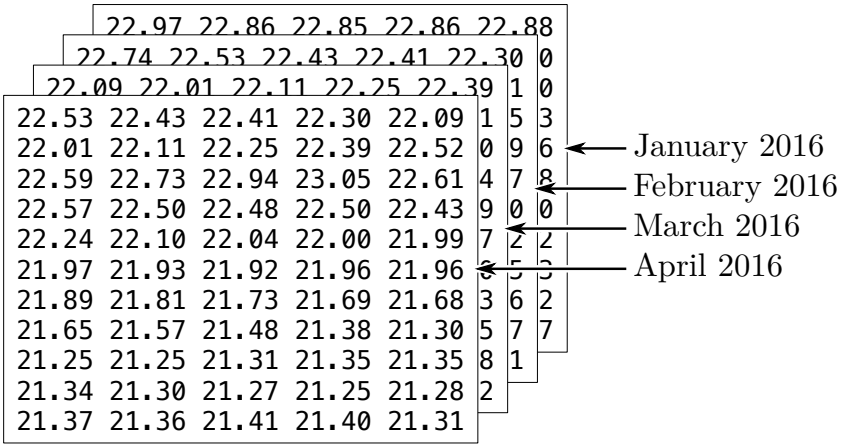

Figure 7: Simple diagram of the internal structure of a raster brick

The data set used can be downloaded from `http://www.metoffice.gov.uk/hadobs/hadisst/data/HadISST_sst.nc.gz`.

Using the `raster` library in R, we can easily load the HadISST1 raster data:

Snippet 3.1: Function to load HasISST data set in netCDF format

```
1 load_hadsst <- function(file = "./HadISST_sst.nc") {
2   b <- brick(file)
3   NAvalue(b) <- -32768 # Land
4   return(b)
5 }
```

Just by doing this we see the clear advantages of using the netCDF format: instead of loading the entire data set, we create a *formal class* raster brick that contains the

information about the structure of the `HadISST_sst.nc` file, whilst pointing to it. This allows us then to work with the data dynamically (e.g., if we wanted to get the value of an individual SST cell, we would only need the memory necessary to store the actual cell). If we used the ASCII format instead, just reading all 114 307 200 individual SST cells using `data.table` (one of the fastest libraries to read RAW data in R) would have taken a few hours; and this doesn't take into account cleaning and organising the data afterwards.

Note that we do not take into account the 100% sea-ice-covered grid-boxes (mainly in the Arctic Ocean and small regions of the Antarctic Ocean), as the activity windows are restricted to the tropical regions.

Now all we need to do is to define a function to read the data from the raster data and get the mean annual (or seasonal to be more precise) SST from a specified region. To do this we crop the raster brick to our desired spatial activity window, and then do the arithmetic mean to get a representative SST value for each year:

Snippet 3.2: Function to get mean SSTs data frame from a specific spatial and temporal activity window

```
get_mean_ssts <- function(x = hadsst.raster, years, range = 6:10,
                          coords = c("180W", "180E", "90S", "90N")){
  coords <- morph_coords(coords)
  area <- extent(as.numeric(coords))
  nms <- names(x)
  x <- crop(x, area)

  months <- c("01", "02", "03", "04", "05", "06", "07", "08", "09",
     "10", "11", "12")
  xMeans <- vector(length = length(years), mode = 'list')
  for (ix in 1:length(years)){
    xMeans[[ix]] <- mean(x[[c(sapply(range,function(x)
       grep(paste0(years[ix],'.',months[x]),nms)))]], na.rm = T)
  }
  mean.brick <- do.call(brick,xMeans)
  mean.brick <- lapply(1:nlayers(mean.brick),function(ix)
     mean(as.matrix(mean.brick[[ix]]), na.rm = T))

  mean.df <- unlist(mean.brick)
  mean.df <- data.frame(sst = mean.df)
  mean.df <- classify_ssts(mean.df, years)
  return(mean.df)
}
```

Notice that in the snippet 3.2 we have used the function `classify_ssts()` (snippet 3.3). This function calculates the global basin SST, mathematically defined as

$$\langle \mathrm{SST} \rangle = \sum_{y} \frac{\mathrm{SST}(y)}{Y} \tag{9}$$

where $\mathrm{SST}(y)$ is the mean SST of the year y, and Y is the total number of years studied; as usual, the standard error of this mean, is defined as

$$SE_{\mathrm{SST}} = \frac{1}{\sqrt{Y}} \sqrt{\frac{1}{Y-1} \sum_{y} \{\mathrm{SST}(y) - \langle \mathrm{SST} \rangle\}^2} \tag{10}$$

The function then classifies each year in low-SST and high-SST years depending on whether they are lower or greater than $\langle \mathrm{SST} \rangle$.

Snippet 3.3: Function to normalise the SSTs and classify years in low or high SST

```
classify_ssts <- function(data.df, years){
  mean.sst <- mean(data.df$sst)
  data.df <- data.df %>%
    mutate(year = as.numeric(substring(rownames(data.df), 1)) +
        years[1] - 1,
           year = ymd(paste(year, "01", "01", sep = "-")),
           sst.norm = sst/mean.sst,
           sst.class = ifelse(sst.norm >= 1, "high", "low"))
  data.df <- data.df[c("year", "sst", "sst.norm", "sst.class")]
  return(data.df)
}
```

As we have stated before, using netCDF is incredibly fast: reading the raster data and computing the necessary calculations merely needs a compile time of 2.639 s for the North Atlantic Ocean, and 1.003 s for the Northeast Pacific Ocean.

We commented in § 2.2.1 that the hurricane seasons do not always start and end in the same year for some basins. If we wanted to do deal with these basins, we would need to define an alternative `get_mean_ssts()` function that takes two month ranges (`first.range` and `second.range`) for input (instead of just `range`), and calculate the mean SST accordingly:

Snippet 3.4: Loop to calculate the mean SST in seasons not contained in the same year

```
  for (ix in 2:length(years){
    xMeans[[ix-1]] <- mean(x[[c(sapply(first.range,function(x)
        grep(paste0(years[ix-1],'.',months[x]), nms)),
        sapply(second.range,function(x)
        grep(paste0(years[ix],'.',months[x]), nms)))]], na.rm = T)
  }
```

3.1.3. Resulting data structure

In table 6 we show the structure of the `ssts.natl` data frame to illustrate the variables we used, as well as the data type of the SST data. Naturally, `ssts.epac` has the same data structure.

year <date>	sst <dbl>	sst.norm <dbl>	sst.class <chr>
1966-01-01	27.47	0.9979934	low
1967-01-01	27.19	0.9879054	low
1968-01-01	27.34	0.9933687	low
1969-01-01	27.75	1.0083072	high
1970-01-01	27.36	0.9940200	low
1971-01-01	27.04	0.9825272	low

Table 6: Excerpt of the `ssts.natl` data frame

3.2. SST ANALYSIS

In figures 8 and 9 we can see the result of doing the analysis described in § 3.1.2 using the function `plot_annual_sst()` defined in the script A.3 in § A. The spatial and temporal activity windows used are the ones described in table 3.

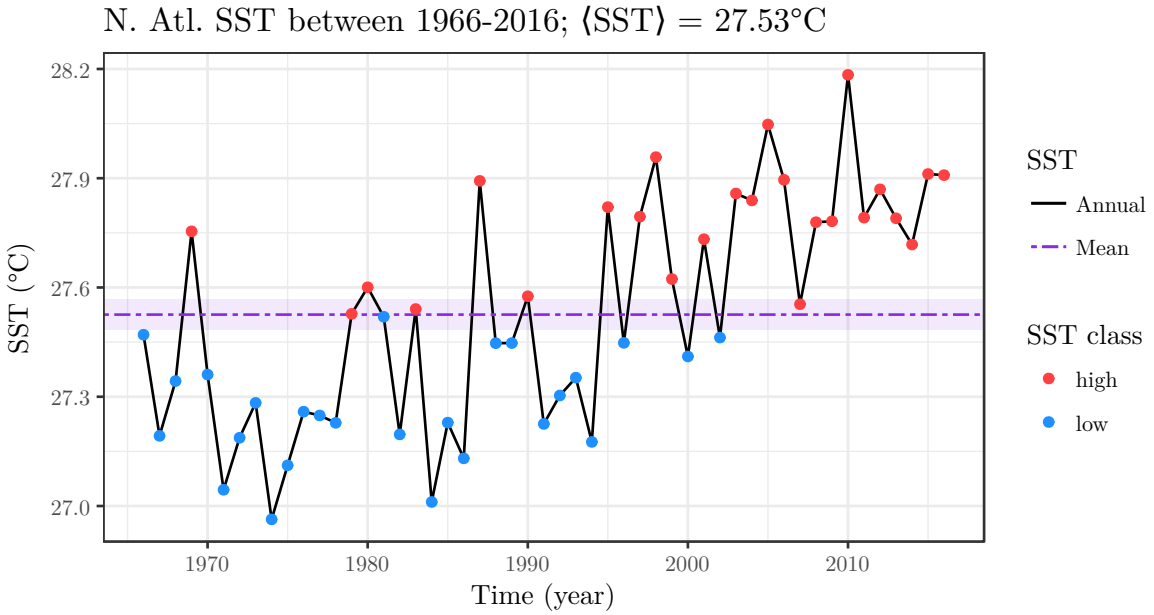

Figure 8: SST analysis for the North Atlantic Ocean

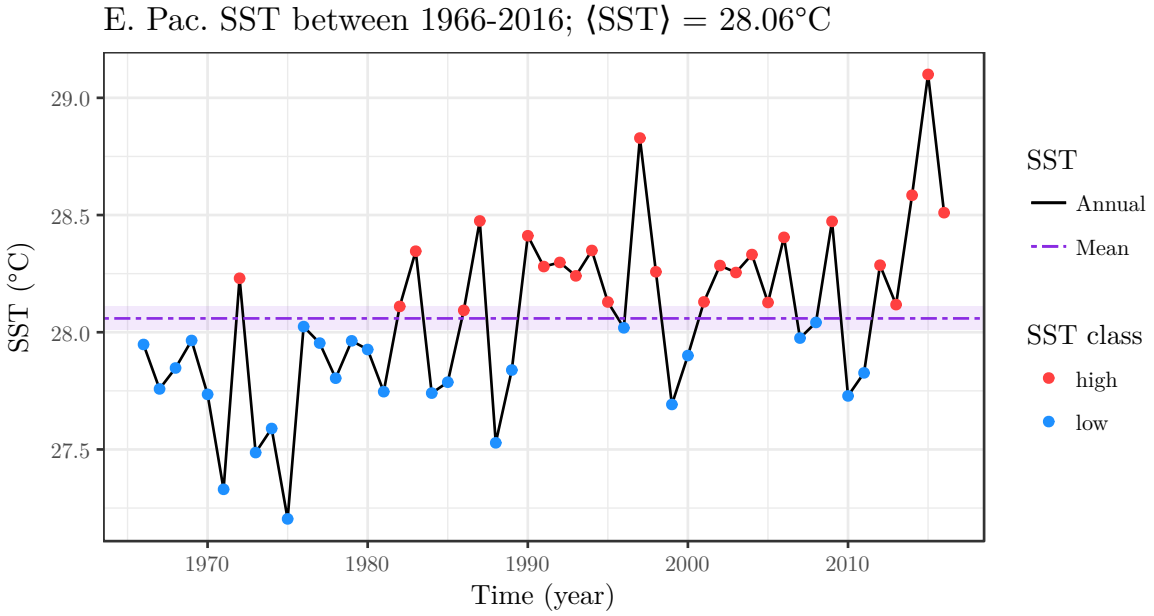

Figure 9: SST analysis for the Northeast Pacific Ocean

The first thing we notice is the general tendency of increasing sea surface temperatures after the 1970s (being the year 1975 a reasonable estimation of the *start* of global warming; see figure 10), and an alarming increase in the rate of warming after the 2000s.

In table 7 we can see the evolution of ⟨SST⟩ comparing three time periods starting from the year 1966: (i) years before the global warming, (ii) years studied in Corral et al.'s original text, and (iii) years studied in this text. As shown, the increase in the sea surface temperatures is quite noticeable.

Basin	1966–1975 ⟨SST⟩	1966–2007 ⟨SST⟩	1966–2016 ⟨SST⟩
N. Atl.	27.27 °C	27.45 °C	27.52 °C
E. Pac.	27.71 °C	28.01 °C	28.06 °C

Table 7: Change of the seasonal mean sea surface temperature

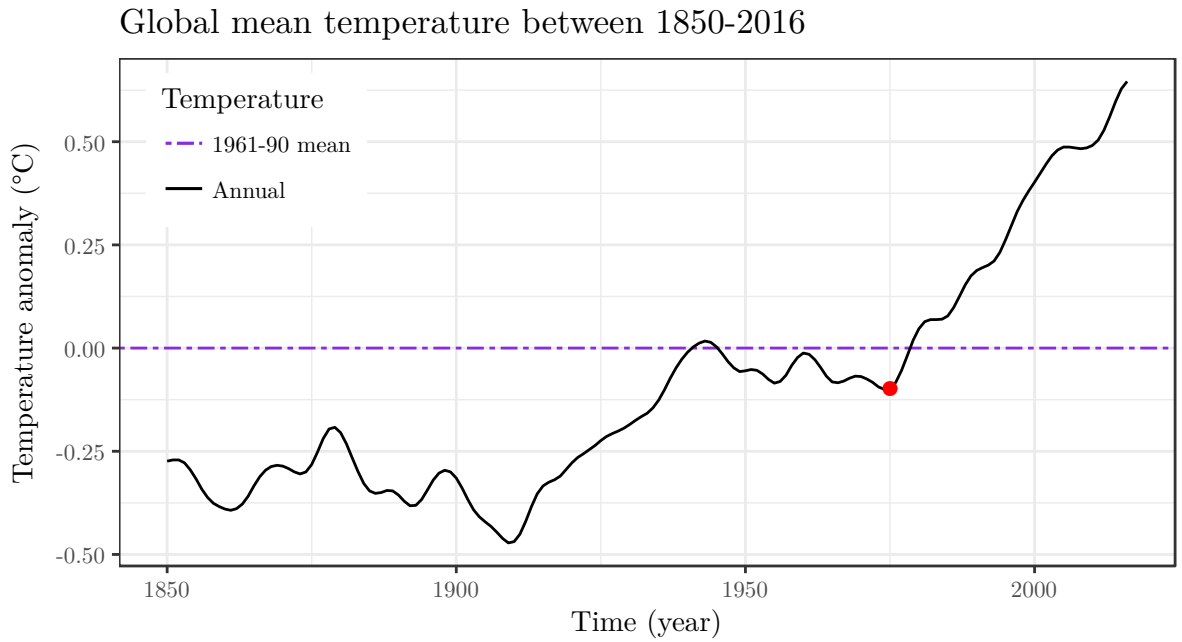

Figure 10: Global surface temperature record for the last 166 years

The time series seen in figure 10, a reproduction of the global time series from [20], has been created using the data provided by the Met Office's HadCRUT4 database. The data set can be downloaded from `http://www.metoffice.gov.uk/hadobs/hadcrut4/data/current/time_series/HadCRUT.4.5.0.0.monthly_ns_avg.txt`.

In table 8 we can see the list of years obtained in each one of the basins studied separated by SST class. With this, we have every piece needed to answer the hypothesis raised in § 2.3.3.

Basin	SST class	List of years
N. Atl.	Low	1966 1967 1968 1970 1971 1972 1973 1974 1975 1976 1977 1978 1981 1982 1984 1985 1986 1988 1989 1991 1992 1993 1994 1996 2000 2002
	High	1969 1979 1980 1983 1987 1990 1995 1997 1998 1999 2001 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016
E. Pac.	Low	1966 1967 1968 1969 1970 1971 1973 1974 1975 1976 1977 1978 1979 1980 1981 1984 1985 1988 1989 1996 1999 2000 2007 2008 2010 2011
	High	1972 1982 1983 1986 1987 1990 1991 1992 1993 1994 1995 1997 1998 2001 2002 2003 2004 2005 2006 2009 2012 2013 2014 2015 2016

Table 8: Separation of the years as a function of the value of the SST

4. Analysis

4.1. Separation by SST

Although in the context of statistics the primary goal of time series analysis is forecasting (as in the ones in figures 8, 9, and 10), we think it's useful to do a time series of the PDI of each basin (figures 11 and 12) to have more intuitive understanding of the data and to detect any outlying storms we might have missed in § 2.3.3.

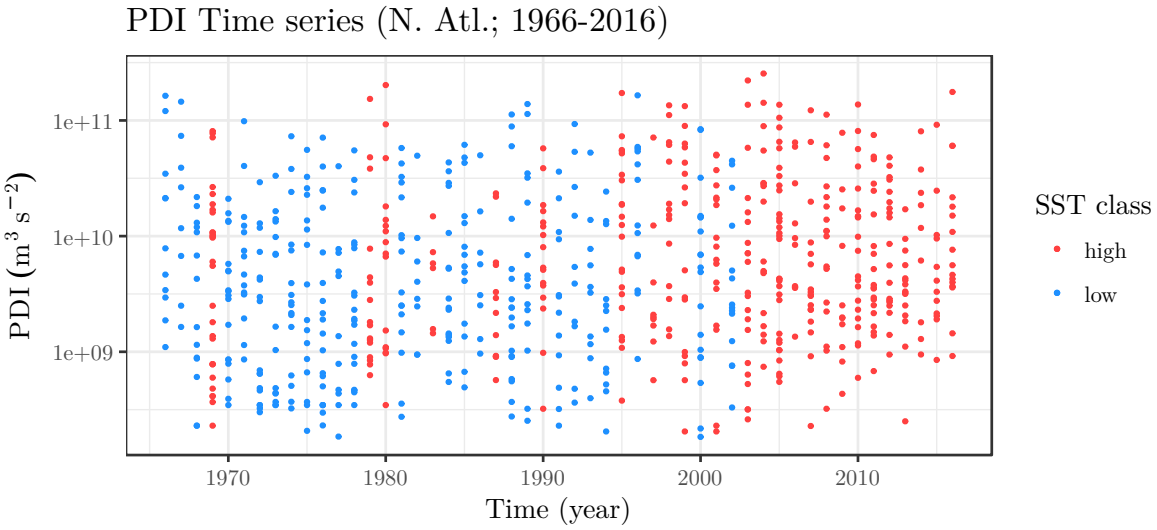

Figure 11: PDI time series for the Northeast Pacific Ocean

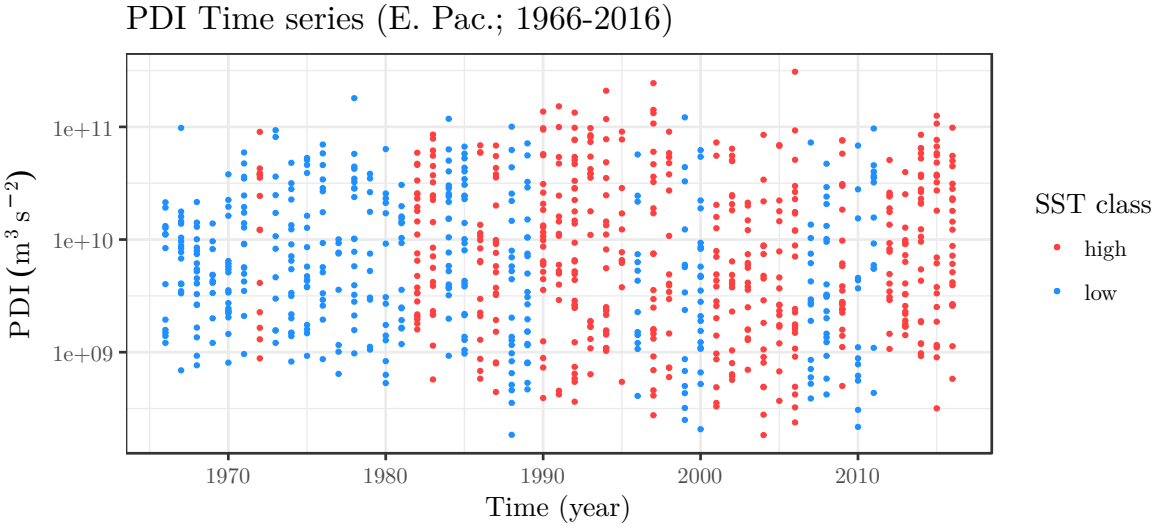

Figure 12: PDI time series for the Northeast Pacific Ocean

As we can see, the data seems quite robust and has no outlying storms (one make an argument about `CP012006` (2006) in the E. Pac. being an outlier, but the PDI gap between it and the closest storm by PDI value is quite small when compared to the outliers detected in § 2.3.3). It's also noticeable the fact that high-SST years have several more storms than low-SST years with a PDI greater than $10^{11}\,\mathrm{m^3\,s^{-2}}$, which already seems to confirm the hypothesis brought up in § 2.3.3. In table 9 we can see a numerical summary of the separation effect on each basin.

Basin	N (high-SST)	N (low-SST)
N. Atl.	406	365
E. Pac.	599	421

Table 9: Summary of tropical-cyclones of each basin separated by SST class

Without further ado, let's examine the PDI density probability distributions presented in figures 13 and 14. For these figures we used the `plot_dpdi_by_sst_class()` function defined in the script A.4, in § A), which is nothing more than a slightly tweaked version of the `plot_dpdi()` function.

In essence we see what we predicted in § 2.3.3: the distribution of high-SST and low-SST years have the same shape, but years with high SST are characterized by tropical-cyclones with larger PDI values. As the PDI integrates the cube of the velocity over the storm life-time, larger PDI values can result from longer lifetimes, larger wind speeds or both. We will try to answer this in the next section.

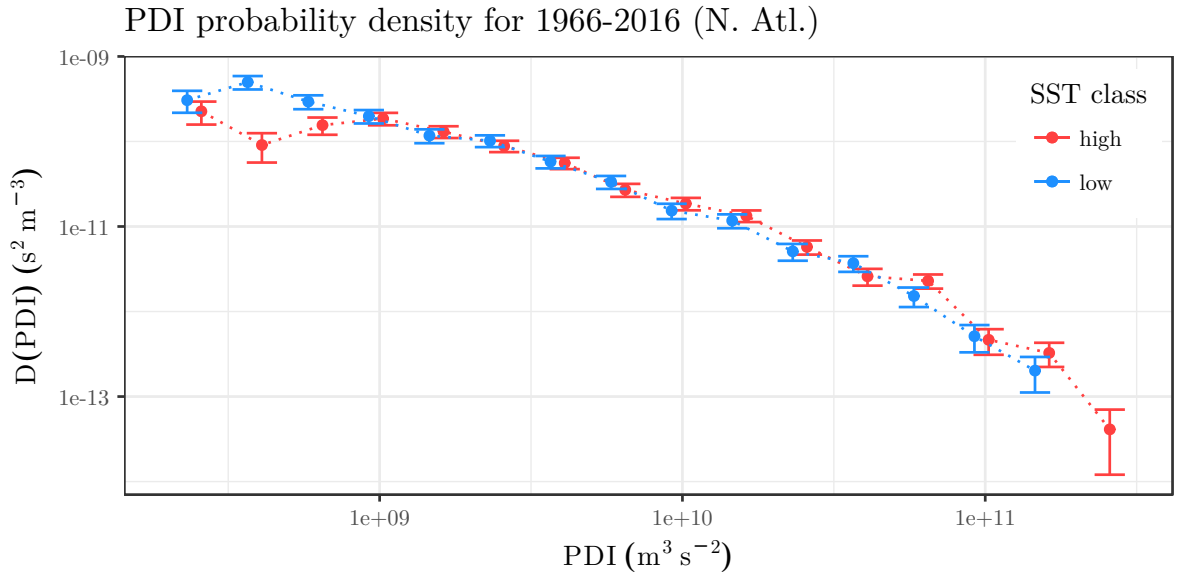

Figure 13: $D(PDI)$ distributions calculated separately for years with high or low SST for the North Atlantic Ocean

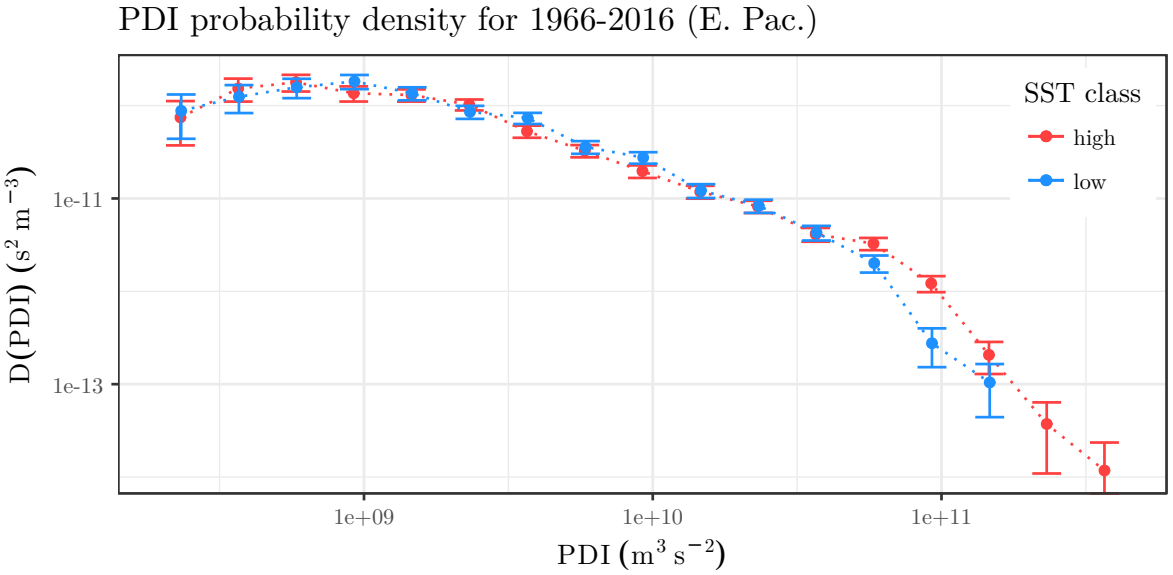

Figure 14: $D(PDI)$ distributions calculated separately for years with high or low SST for the Northeast Pacific Ocean

4.2. PDI correlations

4.2.1. Statistical description

As we mentioned in § 4.1, we want to find a correlation between the PDI and the duration of the storms, or their wind speeds. Instead of using the whole tropical-cyclones data set as is, we will separate developing from non-developing systems.

Tropical-cyclones that surpass the 33 kn wind speed threshold are called developing systems, whilst the ones that do not do so are called non-developing systems. In terms of hydrodynamics, the major difference between developing and non-developing systems is that the developing have a distinct area of low height or low pressure centred on the system, i.e., they form cyclones [21].

This means that even though all tropical-cyclones behave thermodynamically in the same way throughout their lifetime (the PDI calculation is valid for all their life span), the wind speeds evolve rather differently depending on the development status of the storm. For this reason, we will play on the cautious side and study developing and non-developing systems separately.

The most common method to find the best fit is the least squares regression. This method consists in minimising the square of the deviations $y_i - \hat{y}_i$ where (x_i, y_i) are the data points $(i = 1, \ldots, n)$ and $(\hat{x}_i, \hat{y}_i)$ are the expected (x, y) points according to equation

$$y = a + bx \tag{11}$$

More specifically, we have to minimise the sum of the product of the deviation, measured in absolute values:

$$D^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - a - bx_i)^2 \tag{12a}$$

The minimum is reached when

$$\frac{\partial D^2}{\partial a} = -2 \sum_{i=1}^{n} (y_i - a - bx_i) = 0 \tag{12b}$$

$$\frac{\partial D^2}{\partial b} = -2 \sum_{i=1}^{n} x_i (y_i - a - bx_i) = 0 \tag{12c}$$

From equation (12b), we can show that $a = \bar{y} - b\bar{x}$, where $\bar{x}$ and $\bar{y}$ are the mean of x and y:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i, \quad \bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i \tag{13}$$

Replacing a by $\bar{y} - m\bar{x}$ in the equation (12c), we find that the slope is

$$b = \frac{s_{x,y}}{s_x^2} \tag{14}$$

where s_x^2 and $s_{x,y}$ are the variance of x and the covariance of x and y, respectively:

$$s_x^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{15a}$$

$$s_{x,y} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) \tag{15b}$$

The equation of our fit is therefore

$$y = \frac{s_{x,y}}{s_x^2}(x - \bar{x}) + \bar{y} \tag{16}$$

which passes through the point $(\bar{x}, \bar{y})$.

The goodness of a fit can be assessed by measuring the correlation coefficient. This coefficient is given by

$$r = \frac{s_{x,y}}{s_x s_y}, \quad -1 \leq r \leq 1 \tag{17}$$

where $s_x = \sqrt{s_x^2}$ and $s_y = \sqrt{s_y^2}$ are the standard deviations of x and y.

This linear fit model can be interpreted as a causal relationship between x and y. Importantly, causality in this context means the direction of causality runs from x to y and not the other way round.

It's important to notice that the relationships between the PDI and the other variables are of non-linear nature. This naturally means that our regressions need to follow a so-called log–log model:

$$\log\eta = a + b\log\epsilon, \quad \text{where } \log\eta \equiv y, \quad \log\epsilon \equiv x \tag{11 bis}$$

We do not know exactly how the PDI and the duration of the storms, or their wind speeds are correlated; we just suspect there is a correlation. For this reason, we will not only compute the $y(x)$ fit for each data set, but also $x(y) = \alpha + \beta y$, which is calculated exactly as above just interchanging the role of x and y, and calculate the slope and intercept of the equivalent $x(y)^{-1}$ fit respectively as

$$a' = -\frac{\alpha}{\beta}, \quad b' = \frac{1}{\beta} \tag{18}$$

For the determination of the standard errors s_a and s_b, we will rely on the R `stats` library and the usual error propagation methods.

The hypothesis is that the SST does not directly affect the maximum wind speed of a tropical-cyclone: storms of equal duration should, in theory, have the same wind speed and PDI. Once the cyclone is activated, the wind speed should not depend on the SST.

In essence, our methodology will consist in comparing the low-SST and high-SST fits obtained for each data set. Ideally the equations for both fits should be statistically compatible for developing systems. There is no clear basis to say this about non-developing systems, but we will study them anyhow.

4.2.2. Discussion of the results

PDI vs storm duration

In figures 15 and 16 we can see the PDI vs storm duration regression analyses for the North Atlantic Ocean (developing and non-developing systems, respectively); in figures 17 and 18 we can see the same analyses for the Northeast Pacific Ocean instead. These figures have been obtained by using the `plot_pdi_scatter_by_status()` function defined in the script A.4, in § A.

A detailed of the statistical coefficients of the regression models we have computed is given in table 10.

PDI vs wind speed

Regarding the correlation between the PDI and the wind speed, we use the maximum surface wind speed (in $\mathrm{m\,s^{-1}}$) sustained over the entire lifetime of the storm as a representative wind speed value of it. Note that we only study developing systems, as the non-developing systems data sets are statistically scarce (the only possible winds speeds are 20, 25 and 30 kn).

In figures 19 and 20 we can see the PDI vs maximum wind speed regression analyses for the North Atlantic Ocean and the Northeast Pacific Ocean. Note that we have added jitter (a small amount of random variation to the location of each point) to improve the data visualisation, as there are lots of overlapping points. These figures have been obtained by using the `plot_pdi_scatter_wind()` function defined in the script A.4, in § A.

The summary of the statistical coefficients of these regressions can be found in table 11.

Discussion

The first thing to notice about the results is that, as it ought to be, the two regression lines, $y(x)$ and $x(y)^{-1}$, for each fitted data set cut at the $(\bar{x}, \bar{y})$ point.

For the comparison of the low-SST and high-SST regression lines, we will use a coverage factor $k = 2$ to ensure a level of confidence associated with the results of 95.45 %. What we see is that, as we suspected, the SST has no clear influence on the evolution of the tropical-cyclone once it's activated; given a regression model (i.e., $y(x)$ or $x(y)^{-1}$) the two data sets are statistically compatible. This is true for both the PDI vs storm duration and the PDI vs maximum wind speed regressions.

The major difference between the two studied correlations is that the determination coefficient, r^2, is slightly better for the PDI vs maximum wind speed regression. This should not be too surprising, as there is an explicit physical dependence of the PDI on the sustained surface wind speed, as discussed in § 2.1. If anything, we should be surprised by the fact that the maximum wind speed is such a good representative value of the whole storm.

What we did definitely not expect is that our hypothesis would also be valid for non-developing systems; the correlations in the data are even better than for the developing systems data sets ($r^2 \sim 0.9$ vs $r^2 \sim 0.6$), but that could be associated to the low number of storms, resulting in less spread in the data. Having said that, it's a remarkable result.

To finish, we would like to discuss the difference between the $y(x)$ and $x(y)^{-1}$ fits. What one would expect from a perfect correlation is that the $y(x)$ and $x(y)^{-1}$ functions describe the same exact linear equation. By all means, nonetheless, in real-life data the determination coefficient will hardly be $r^2 \equiv 1$.

The mathematical relationship between $y(x)$ and $x(y)^{-1}$, with respective slopes of b and b', clearly explains this behaviour:

$$b' = \frac{b}{r^2} \tag{19}$$

In our results this is quite unmistakable, the limit $y(x) = x(y)^{-1}$ is truer for higher r^2 coefficients. This is undoubtedly reflected upon the regression lines themselves: the low-SST PDI vs maximum wind speed regression for the E. Pac. (figure 20) has the best correlation, whilst both PDI vs storm duration regressions for E. Pac.'s developing systems (figure 17) have the worst.

Basin	System status	$f(x)$	SST class	a	b	r^2
N. Atl.	Developing	$y(x)$	Low	5.94 ± 0.18	1.83 ± 0.08	0.67
			High	5.91 ± 0.17	1.83 ± 0.08	0.62
		$x(y)^{-1}$	Low	3.9 ± 0.5	2.74 ± 0.12	0.67
			High	3.4 ± 0.4	2.97 ± 0.13	0.62
	Non-developing	$y(x)$	Low	6.76 ± 0.07	1.09 ± 0.04	0.88
			High	7.02 ± 0.08	0.98 ± 0.04	0.89
		$x(y)^{-1}$	Low	6.5 ± 0.4	1.24 ± 0.04	0.88
			High	6.8 ± 0.5	1.09 ± 0.05	0.89
E. Pac.	Developing	$y(x)$	Low	6.07 ± 0.18	1.78 ± 0.08	0.59
			High	5.36 ± 0.18	2.10 ± 0.08	0.60
		$x(y)^{-1}$	Low	3.5 ± 0.4	3.00 ± 0.13	0.59
			High	2.2 ± 0.4	3.53 ± 0.14	0.60
	Non-developing	$y(x)$	Low	7.05 ± 0.11	0.96 ± 0.06	0.87
			High	7.15 ± 0.09	0.91 ± 0.05	0.85
		$x(y)^{-1}$	Low	6.8 ± 0.7	1.09 ± 0.07	0.87
			High	6.9 ± 0.7	1.07 ± 0.06	0.85

Table 10: Summary of the PDI vs duration linear regressions

Basin	$f(x)$	SST class	a	b	r^2
N. Atl.	$y(x)$	Low	4.64 ± 0.14	3.43 ± 0.09	0.85
		High	4.90 ± 0.11	3.28 ± 0.07	0.86
	$x(y)^{-1}$	Low	3.7 ± 0.3	4.04 ± 0.11	0.85
		High	4.10 ± 0.23	3.80 ± 0.08	0.86
E. Pac.	$y(x)$	Low	5.07 ± 0.09	3.14 ± 0.06	0.87
		High	4.94 ± 0.07	3.23 ± 0.04	0.93
	$x(y)^{-1}$	Low	4.37 ± 0.21	3.60 ± 0.07	0.87
		High	4.54 ± 0.15	3.48 ± 0.05	0.93

Table 11: Summary of the PDI vs maximum wind speed linear regressions for developing systems

4.2.3. PDI vs storm duration scatterplots

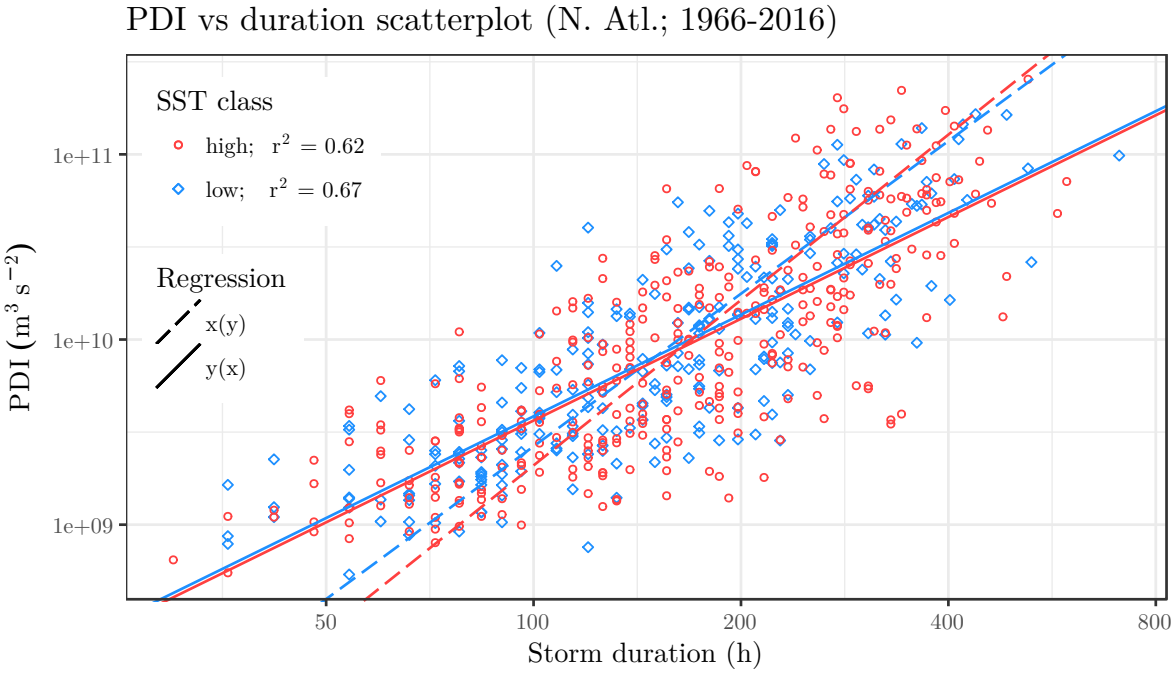

Figure 15: *PDI* vs duration analysis for developing systems (N. Atl.)

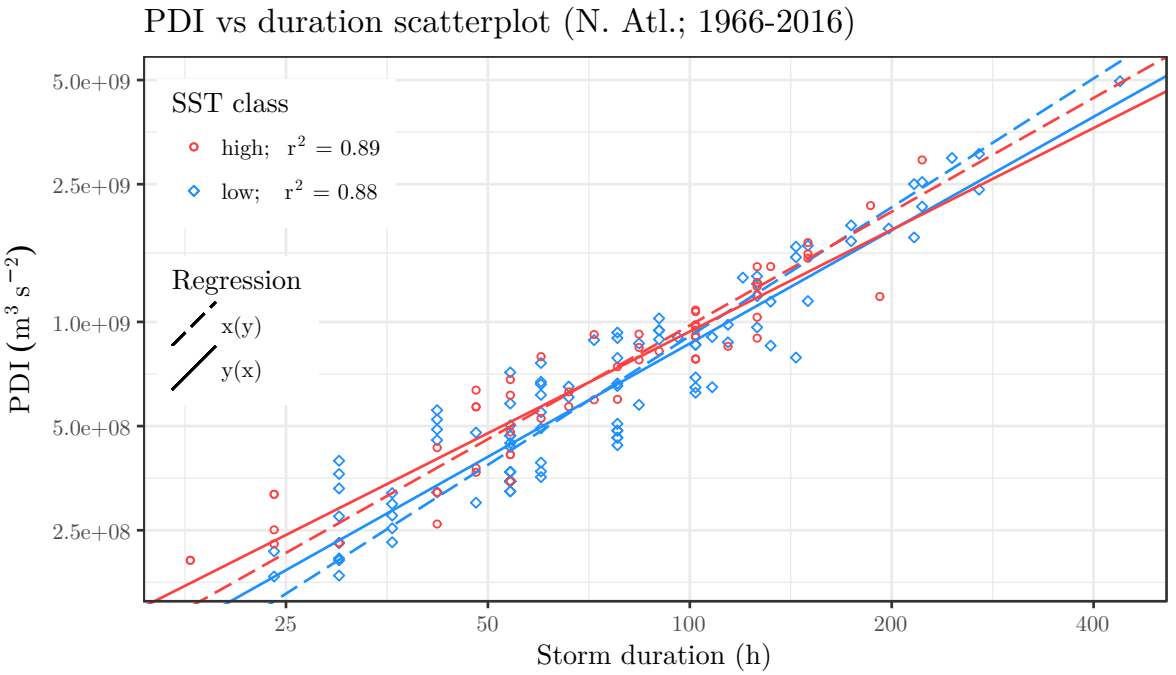

Figure 16: *PDI* vs duration analysis for non-developing systems (N. Atl.)

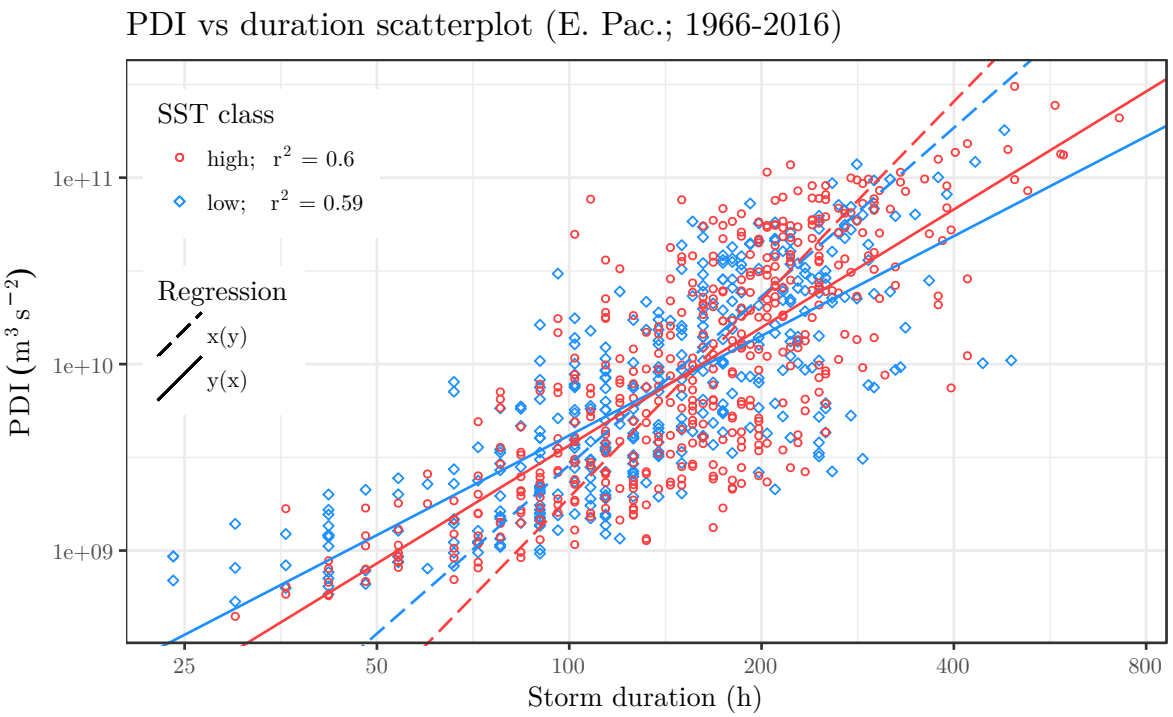

Figure 17: *PDI* vs duration analysis for developing systems (E. Pac.)

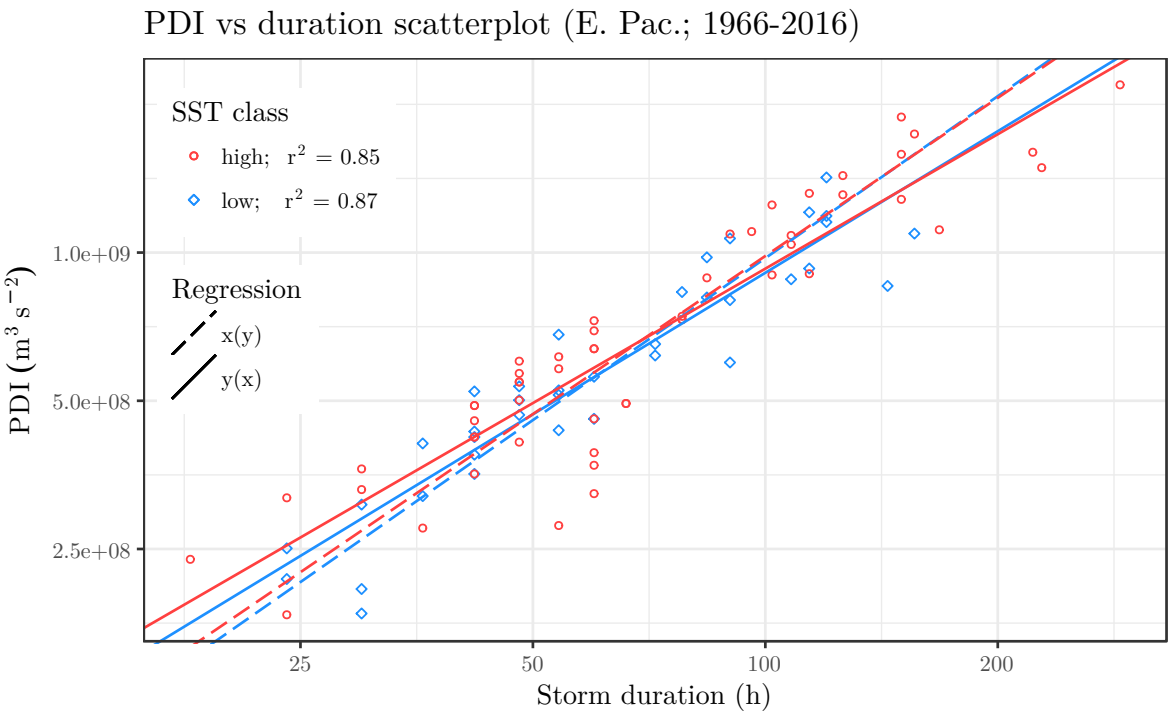

Figure 18: *PDI* vs duration analysis for non-developing systems (E. Pac.)

4.2.4. PDI vs wind speed scatterplots

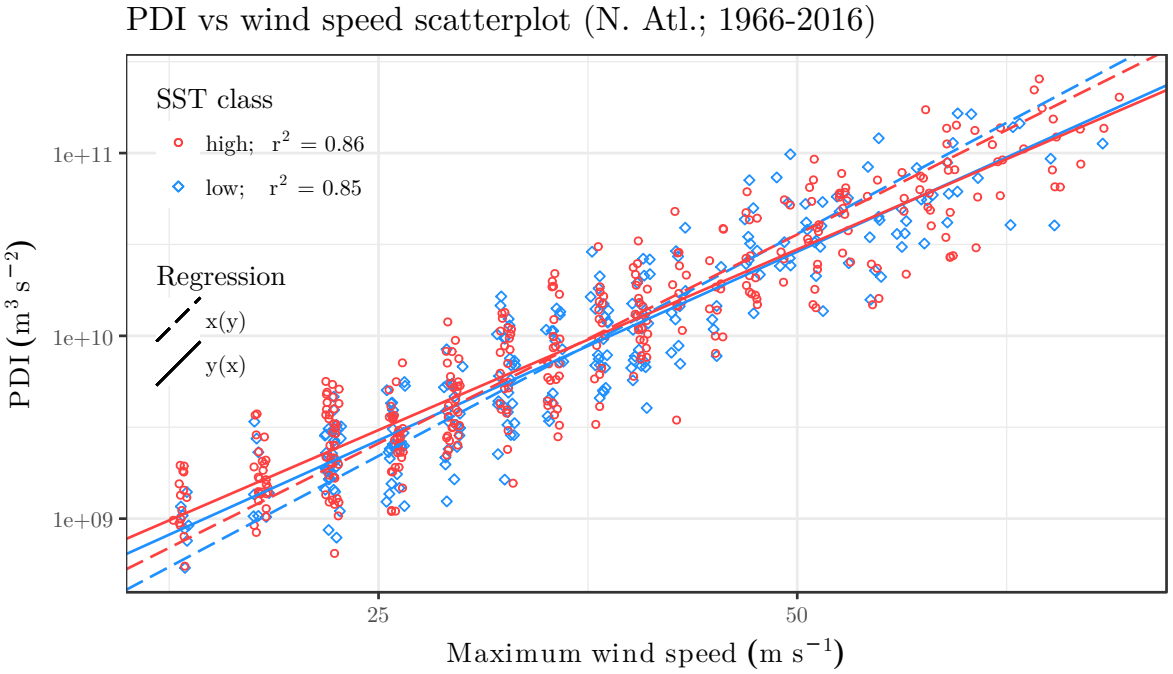

Figure 19: *PDI* vs maximum wind speed analysis for developing systems (N. Atl.)

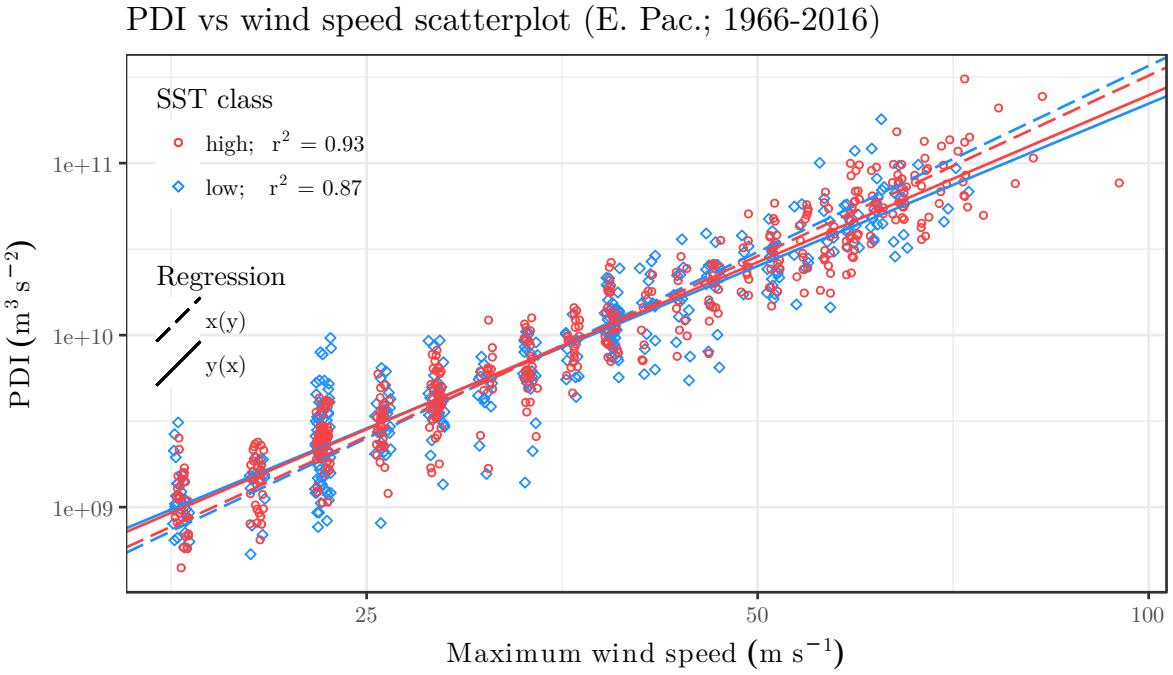

Figure 20: *PDI* vs maximum wind speed analysis for developing systems (E. Pac.)

5. Conclusions

Even though a big part of this text is a statistical analysis that, as yet, no other author has addressed, reproducibility has been the main focus of this thesis. This in and of itself has presented a reasonable amount of challenges; reproducing an advanced scientific study can be as hard as coming up with a state-of-the-art study on your own. That being said, the whole process has been truly satisfying as a scientist.

The results we have found in § 4.1 are a corroboration of what most climate scientists agree on: global warming is a reality with consequences that affect us all; whilst the results found in § 4.2 give an interesting insight on the statistical properties of tropical-cyclones and could well be a starting point of a comprehensive study using multivariate statistics.

The main limitation of this research has probably been the amount of basins analysed. Even though our results are statistically robust as such, it would be quite precipitate to generalise these rather localised results without performing an statistical analysis for each of the omitted basins in this text.

A. Developed code

The code developed for this thesis is licensed under the GNU General Public License v3.0 and can be downloaded from `https://github.com/aldomann/tropical-cyclones`.

Script A.1: `hurdat2_base.R` - Base code to study the PDI of hurricane data from the National Hurricane Center (HURDAT)

```
1  library(tidyverse)
2  library(stringr) # To split lines
3  library(lubridate) # Use dates
4
5  # Get hurricane observations data frame --------------------
6
7  get_hurr_obs <- function(filename){
8    # Read and split raw data ----------------------------------
9
10   # tracks.url <- paste0("http://www.aoml.noaa.gov/hrd/hurdat/",
       "hurdat2-nepac-1949-2016-apr2017.txt")
11   # tracks.url <- paste0("http://www.aoml.noaa.gov/hrd/hurdat/",
       "hurdat2-1851-2016-apr2017.txt")
12   tracks.file <- paste0("data/", filename)
13   hurr.tracks <- readLines(tracks.file)
14   hurr.tracks <- lapply(hurr.tracks, str_split, pattern = ",",
       simplify = TRUE)
15
16   # Clean the raw data ---------------------------------------
17
18   # Split the hurr.tracks into meta and observation lists
19   hurr.lengths <- sapply(hurr.tracks, length)
20   hurr.meta <- hurr.tracks[hurr.lengths == 4]
21   hurr.obs <- hurr.tracks[hurr.lengths == 21]
22
23   # Create and clean meta data frame
24   hurr.meta <- lapply(hurr.meta, tibble::as_tibble)
25   hurr.meta <- bind_rows(hurr.meta)
26
27   hurr.meta <- hurr.meta %>%
28     dplyr::select(-V4) %>%
29     rename(storm.id = V1, storm.name = V2, n.obs = V3) %>%
30     mutate(storm.name = str_trim(storm.name),
31            n.obs = as.numeric(n.obs))
32
33   storm.id <- rep(hurr.meta$storm.id, times = hurr.meta$n.obs)
34
35   # Create and clean obs data frame
36   hurr.obs <- lapply(hurr.obs, tibble::as_tibble)
37   hurr.obs <- bind_rows(hurr.obs) %>%
38     mutate(storm.id = storm.id) %>%
39     dplyr::select(storm.id, V1, V2, V4:V7) %>%
40     rename(date = V1, time = V2, status = V4, lat = V5, long = V6,
         wind = V7)
41
```

```
42  # Change date and time & unite them
43  hurr.obs <- hurr.obs %>%
44    unite(date.time, date, time) %>%
45    mutate(date.time = ymd_hm(date.time))
46
47  # Meaningful status names
48  storm.levels <- c("TD", "TS", "HU", "EX", "SD", "SS", "LO", "WV",
       "DB")
49  storm.labels <- c("Tropical␣depression", "Tropical␣storm",
       "Hurricane", "Extratropical␣cyclone", "Subtropical␣
       depression", "Subtropical␣storm", "Other␣low", "Tropical␣
       wave", "Disturbance")
50  hurr.obs <- hurr.obs %>%
51    mutate(status = factor(str_trim(status),
52                           levels = storm.levels,
53                           labels = storm.labels))
54
55  # Morph coordinates
56  morph_long <- function(long){
57    long <- ifelse(str_extract(long, "[A-Z]") == "W",
58                   - as.numeric(str_extract(long, "[^A-Z]+")),
59                     as.numeric(str_extract(long, "[^A-Z]+")) )
60    return(long)
61  }
62  morph_lat <- function(lat){
63    lat <- ifelse(str_extract(lat, "[A-Z]") == "S",
64                  - as.numeric(str_extract(lat, "[^A-Z]+")),
65                    as.numeric(str_extract(lat, "[^A-Z]+")) )
66    return(lat)
67  }
68
69  # Split the numeric coordinates from their directions
70  hurr.obs <- hurr.obs %>%
71    mutate(lat.num = morph_lat(lat),
72           long.num = morph_long(long))
73
74  # Clean non-standard data ---------------------------------
75
76  # Ignore data outside the delta.t = 6 hours
77  hurr.obs <- hurr.obs %>%
78    filter(hour(date.time) == 00 |
79             hour(date.time) == 06 |
80             hour(date.time) == 12 |
81             hour(date.time) == 18) %>%
82    filter(minute(date.time) == 00)
83
84  # Clean up wind column ------------------------------------
85
86  # Manually change odd middle values for AL191976 & AL111973
87  hurr.obs <- hurr.obs %>%
88    mutate(wind = ifelse(storm.id == "AL191976" & wind == "␣-99",
       20, wind),
```

```
89            wind = ifelse(storm.id == "AL111973" & wind == "␣-99",
                 30, wind),
90            wind = ifelse(storm.id == "AL111973" & month(date.time)
                 == 9 & day(date.time) == 12 & hour(date.time) == 12,
                 NA, wind)) %>%
91      filter(is.na(wind) != TRUE)
92
93    # Clean and reformat the wind column
94    hurr.obs <- hurr.obs %>%
95      mutate(wind = ifelse(wind == "␣-99", NA, as.numeric(wind)))
96
97    # Add useful info to data frame ----------------------------
98
99    # Add storm.name and storm.year to hurr.obs
100   hurr.obs <- hurr.obs %>%
101     left_join(hurr.meta, by = "storm.id") %>%
102     mutate(storm.year = year(date.time))
103
104   # Recalculate n.obs
105   hurr.obs <- hurr.obs %>%
106     group_by(storm.id) %>%
107     mutate(n.obs = length(wind))
108
109   # Rearrange hurr.obs data frame columns
110   hurr.obs <- hurr.obs[c("storm.id", "storm.name", "n.obs",
        "date.time", "status", "lat.num", "long.num", "wind",
        "storm.year")]
111   return(hurr.obs)
112 }
```

Script A.2: `hurdat2_pdi_base.R` - Base code to study the PDI probability density (DPDI)

```
1  source("hurdat2_base.R")
2
3  library(measurements)
4  library(scales)
5
6  # Calculate the PDI ----------------------------------------
7
8  # Create data frame with PDI and year of the storm
9  get_pdis <- function(hurr.obs){
10   hurr.obs.pdi <- hurr.obs %>%
11     group_by(storm.id, storm.name, n.obs) %>%
12     summarise(storm.pdi = sum(conv_unit(wind, "knot", "m_per_sec")^3
          * conv_unit(6, "hr", "sec")),
13               max.wind = max(wind)) %>%
14     mutate(storm.duration = n.obs * conv_unit(6, "hr", "sec")) %>%
15     mutate(storm.year = substring(storm.id, 5, 9)) %>%
16     filter(storm.pdi != "NA") %>%
17     filter(storm.pdi != 0)
18   hurr.obs.pdi <- hurr.obs.pdi[c("storm.id", "storm.name", "n.obs",
       "storm.duration", "storm.pdi", "max.wind", "storm.year")]
```

```
19   return(hurr.obs.pdi)
20 }
21
22 # Get PDI of a single storm by name
23 get_pdi <- function(hurr.obs.pdi, storm, year){
24   wanted.pdi.df <- hurr.obs.pdi %>%
25     filter(storm.name == toupper(storm)) %>%
26     filter(storm.year == year)
27   wanted.pdi.df$storm.pdi
28 }
29
30 # Get PDI of a single storm by id
31 get_pdi_by_id <- function(hurr.obs.pdi, id){
32   wanted.pdi.df <- hurr.obs.pdi %>%
33     filter(storm.id == id)
34   wanted.pdi.df$storm.pdi
35 }
36
37 # Calculate and plot the DPDI ------------------------------
38
39 # Function to calculate the DPDI in a range of years
40 get_dpdi <- function(hurr.obs.pdi, years){
41   hurr.obs.pdi <- hurr.obs.pdi %>%
42     filter(storm.year %in% years)
43
44   c <- 10^(1/4)
45   m <- min(hurr.obs.pdi$storm.pdi)
46
47   dpdi.df <- data.frame()
48   for (i in 1:20) {
49     dpdi.df <- rbind(dpdi.df, c(c^(i-1)*m, c^(i)*m))
50   }
51   colnames(dpdi.df) <- c("pdi.min", "pdi.max")
52   dpdi.df <- dpdi.df %>%
53     mutate(pdi.star = sqrt(pdi.min * pdi.max),
54            pdi.bin = pdi.max - pdi.min) %>%
55     rowwise() %>%
56     mutate(ndpdi = sum(hurr.obs.pdi$storm.pdi >= pdi.min &
          hurr.obs.pdi$storm.pdi <= pdi.max),
57            dpdi = ndpdi/(length(hurr.obs.pdi$storm.pdi)*pdi.bin),
58            pdi.error = dpdi / sqrt(ndpdi) ) %>%
59     filter(dpdi != 0)
60
61   return(dpdi.df)
62 }
63
64 # Function to plot the DPDI data frame
65 plot_dpdi <- function(hurr.obs.pdi, years){
66   dpdi.df <- get_dpdi(hurr.obs.pdi, years)
67   ggplot(dpdi.df, aes(x = pdi.star, y = dpdi, ymin = dpdi-pdi.error,
        ymax = dpdi+pdi.error)) +
68     geom_line(linetype = "dotted") +
```

```
69      geom_point() +
70      geom_errorbar(width = 0.1) +
71      scale_x_log10() +
72      scale_y_log10() +
73      labs(title = paste0("PDI␣probability␣density␣for␣", years[1],
           "-", years[length(years)],
74                          "␣(", attr(hurr.obs.pdi, "title"), ")"),
75           x = bquote(PDI~ (m^3 ~s^-2)), y = bquote(D(PDI)~(s^2~m^-3)))
76  }
77
78  # Track a storm ------------------------------------------
79
80  # Track of a single storm by name
81  track_storm <- function(hurr.obs = hurr.all.obs, storm, year){
82    ggplot(hurr.obs %>%
83             filter(storm.name == toupper(storm)) %>%
84             filter(storm.year == year),
85           aes(x = date.time, y = wind)) +
86      geom_line(linetype = "dotted") +
87      geom_point(aes(colour = status)) +
88      labs(title = paste0(storm, "␣profile␣", "(", year, ")",
89                          ",␣PDI␣=␣", scientific(get_pdi(hurr.all.pdi,
                               storm, year), digits = 3), "␣m^3/s^2"),
90           x = "Time␣(days)", y = "Velocity␣(kt)", colour = "Status")
91  }
92
93  # Track of a single storm by id
94  track_storm_by_id <- function(hurr.obs = hurr.all.obs, id){
95    wanted.storm <- hurr.obs %>%
96      filter(storm.id == id)
97    storm <- wanted.storm$storm.name
98    year <- wanted.storm$storm.year
99
100   ggplot(wanted.storm, aes(x = date.time, y = wind)) +
101     geom_line(linetype="dotted") +
102     geom_point(aes(colour = status)) +
103     labs(title = paste0(storm, "␣profile␣", "(", year, ")",
104                         ",␣PDI␣=␣",
                              scientific(get_pdi_by_id(hurr.all.pdi,
                              id), digits = 3), "␣m^3/s^2"),
105          x = "Time␣(days)", y = "Velocity␣(kt)", colour = "Status")
106 }
```

Script A.3: `hadisst_base.R` - Base code to study the SST data from the Hadley Centre (HadISST)

```
1  library(tidyverse)
2  library(stringr)
3  library(raster)
4  library(rasterVis)
5  library(scales)
6  library(lubridate)
7
```

```
# SST manipulation functions ------------------------------

load_hadsst <- function(file = "./HadISST_sst.nc") {
  b <- brick(file)
  NAvalue(b) <- -32768 # Land
  return(b)
}

# Transform basin coordinates into numbers
morph_coords <- function(coords){
  coords[1] <- ifelse(str_extract(coords[1], "[A-Z]") == "W",
                      - as.numeric(str_extract(coords[1], "[^A-Z]+")),
                        as.numeric(str_extract(coords[1], "[^A-Z]+"))
                          )
  coords[2] <- ifelse(str_extract(coords[2], "[A-Z]") == "W",
                      - as.numeric(str_extract(coords[2], "[^A-Z]+")),
                        as.numeric(str_extract(coords[2], "[^A-Z]+"))
                          )
  coords[3] <- ifelse(str_extract(coords[3], "[A-Z]") == "S",
                      - as.numeric(str_extract(coords[3], "[^A-Z]+")),
                        as.numeric(str_extract(coords[3], "[^A-Z]+"))
                          )
  coords[4] <- ifelse(str_extract(coords[4], "[A-Z]") == "S",
                      - as.numeric(str_extract(coords[4], "[^A-Z]+")),
                        as.numeric(str_extract(coords[4], "[^A-Z]+"))
                          )
  return(coords)
}

# Get mean SSTs data frame filtering by spatial and temporal window
    of activity
get_mean_ssts <- function(x = hadsst.raster, years, range = 6:10,
                          coords = c("180W", "180E", "90S", "90N")){
  coords <- morph_coords(coords)
  aoi <- extent(as.numeric(coords))
  nms <- names(x)
  x <- crop(x, aoi)

  months <-
     c("01","02","03","04","05","06","07","08","09","10","11","12")
  xMeans <- vector(length = length(years), mode = 'list')
  for (ix in 1:length(years)){
    xMeans[[ix]] <- mean(x[[c(sapply(range,function(x)
       grep(paste0(years[ix],'.',months[x]),nms)))]], na.rm = T)
  }
  mean.brick <- do.call(brick,xMeans)
  mean.brick <- lapply(1:nlayers(mean.brick),function(ix)
     mean(as.matrix(mean.brick[[ix]]), na.rm = T))

  mean.df <- unlist(mean.brick)
  mean.df <- data.frame(sst = mean.df)
  mean.df <- normalise_ssts(mean.df, years)
```

```
52   return(mean.df)
53 }
54
55 # Normalise SSTs and divide by class
56 normalise_ssts <- function(data.df, years){
57   mean.sst <- mean(data.df$sst)
58   data.df <- data.df %>%
59     mutate(year = as.numeric(substring(rownames(data.df), 1)) +
         years[1] - 1,
60            year = ymd(paste(year, "01", "01", sep = "-")),
61            sst.norm = sst/mean.sst,
62            sst.class = ifelse(sst.norm >= 1, "high", "low"))
63   data.df <- data.df[c("year", "sst", "sst.norm", "sst.class")]
64   return(data.df)
65 }
66
67 get_low_years <- function(data.df) {
68   low.years <- year(data.df[data.df$sst.class == "low", ]$year)
69   return(low.years)
70 }
71
72 get_high_years <- function(data.df) {
73   low.years <- year(data.df[data.df$sst.class == "high", ]$year)
74   return(low.years)
75 }
76
77 # Data visualisation functions ----------------------------
78
79 # SST map of a single time layer
80 map_global_sst <- function(x = hadsst.raster, month, year){
81   time.layer = (year - 1870) * 12 + month
82   gplot(raster(hadsst.raster, layer = time.layer)) +
83     geom_tile(aes(fill = value)) +
84     scale_fill_gradientn(colours = c("#760200", "#b10e00",
         "#ec3b00", "#f9a100", "#d3ed0a", "#88ec6a", "#6cd2a8",
         "#4ca6e8", "#3276fb", "#214FBB", "#010546"),
85                          values = rescale(c(35, 30, 25, 20, 15, 10,
                               7.5, 5, 2.5, 0, -1000)),
86                          breaks=c(35, 30, 25, 20, 15, 10, 5,  0, -5,
                               -1000),
87                          labels=c(35, 30, 25, 20, 15, 10, 5,  0, -5,
                               "Ice"),
88                          guide="legend", na.value = "white") +
89     labs(fill = "SST") +
90     theme(plot.margin=margin(c(0,10,0,0)),
91           legend.margin=margin(c(1,1,5,-15)),
92           axis.line=element_blank(),
93           axis.text.x=element_blank(),
94           axis.text.y=element_blank(),
95           axis.ticks=element_blank(),
96           axis.title.x=element_blank(),
97           axis.title.y=element_blank(),
```

```
98            panel.background=element_blank())
99 }
100
101 # Plot SST time series
102 plot_annual_sst_norm <- function(data.df){
103   title <- attr(data.df, "title")
104   years.str <- paste0(year(data.df$year[1]), "-",
        year(data.df$year[length(data.df$year)]))
105
106   ggplot(data.df, aes(x = year, y = sst.norm)) +
107     geom_line(aes(linetype = "Annual"), colour = "black") +
108     geom_hline(aes(yintercept = 1, linetype = "Mean"), colour =
          "blueviolet") +
109     scale_linetype_manual(values = c("solid", "twodash")) +
110     geom_point(aes(colour = sst.class)) +
111     scale_colour_manual(values = c("brown1", "dodgerblue1")) +
112     labs(title = paste0(title, "␣SST␣between␣", years.str),
113          x = "Time␣(year)", y = "SST/<SST>",
114          linetype = "SST", colour = "SST␣class") +
115     guides(linetype = guide_legend(override.aes = list(colour =
          c("black", "blueviolet"))))
116 }
117
118 plot_annual_sst <- function(data.df){
119   mean.sst <- mean(data.df$sst)
120   mean.sd.sst <- sd(data.df$sst)/sqrt(length(data.df$sst))
121
122   title <- attr(data.df, "title")
123   years.str <- paste0(year(data.df$year[1]), "-",
        year(data.df$year[length(data.df$year)]))
124
125   ggplot(data.df, aes(x = year, y = sst)) +
126     annotate("rect", fill = "blueviolet", alpha = 0.1,
127              xmin = as.Date(-Inf, origin = data.df$year[1]),
128              xmax = as.Date(Inf, origin = data.df$year[1]),
129              ymin = mean.sst - mean.sd.sst, ymax = mean.sst +
                     mean.sd.sst) +
130     geom_line(aes(linetype = "Annual"), colour = "black") +
131     geom_hline(aes(yintercept = mean.sst, linetype = "Mean"), colour
          = "blueviolet") +
132     scale_linetype_manual(values = c("solid", "twodash")) +
133     geom_point(aes(colour = sst.class)) +
134     scale_colour_manual(values = c("brown1", "dodgerblue1")) +
135     labs(title = paste0(title, "␣SST␣between␣", years.str, ";␣<SST>␣
          =␣", format(round(mean.sst, 2), nsmall = 2), "C"),
136          x = "Time␣(year)", y = "SST␣(C)",
137          linetype = "SST", colour = "SST␣class") +
138     guides(linetype = guide_legend(override.aes = list(colour =
          c("black", "blueviolet"))))
139 }
```

Script A.4: `analysis_base.R` - Base code with several functions needed in `main_analysis.R`

```
library(maps)
library(ggalt)

# Basins maps functions ------------------------------------

# Map showing the hurricanes in specified window
# SRC: https://stackoverflow.com/a/33302968/988432
scale_x_longitude <- function(xmin = -180, xmax = 180, step = 1,
    xtra.lim = 1.5, ...) {
  xbreaks <- seq(xmin,xmax,step)
  xlabels <- unlist(lapply(xbreaks, function(x) ifelse(x < 0,
      parse(text=paste0(-x,"^o", "*W")), ifelse(x > 0,
      parse(text=paste0(x,"^o", "*E")), x))))
  return(scale_x_continuous("Longitude", breaks = xbreaks, labels =
      xlabels, expand = c(0, 0), limits = c(xmin-xtra.lim,
      xmax+xtra.lim), ...))
}

scale_y_latitude <- function(ymin = -90, ymax = 90, step = 0.5,
    xtra.lim = 1.5, ...) {
  ybreaks <- seq(ymin,ymax,step)
  ylabels <- unlist(lapply(ybreaks, function(x) ifelse(x < 0,
      parse(text=paste0(-x,"^o", "*S")), ifelse(x > 0,
      parse(text=paste0(x,"^o", "*N")), x))))
  return(scale_y_continuous("Latitude", breaks = ybreaks, labels =
      ylabels, expand = c(0, 0), limits = c(ymin-xtra.lim,
      ymax+xtra.lim), ...))
}

# Install legacy version of ggalt (see
    https://github.com/hrbrmstr/ggalt/issues/33)
# devtools::install_github("rplzzz/ggalt", ref = "ggp221")
map_region_hurrs <- function(hurr.obs, years, coords, rect.coords,
    steps = c(5,5), xtra.lims = c(1.5,1.5)){
  coords <- morph_coords(coords)
  rect.coords <- morph_coords(rect.coords)
  hurr.obs <- hurr.obs %>%
    filter(storm.year %in% years)
  world_map <- map_data("world")
  world_map <- subset(world_map, region!="Antarctica")

  ggplot(data = world_map, aes(x = long, y = lat, group = group)) +
    geom_cartogram(map = world_map, aes(map_id = region), colour =
        "white", fill = "grey50") +
    scale_x_longitude(xmin = as.numeric(coords[1]),
                      xmax = as.numeric(coords[2]),
                      step = steps[1], xtra.lim = xtra.lims[1]) +
    scale_y_latitude(ymin = as.numeric(coords[3]),
                     ymax = as.numeric(coords[4]),
                     step = steps[2], xtra.lim = xtra.lims[2]) +
```

```
38      coord_proj("+proj=merc") +
39      geom_path(data = hurr.obs, aes(x = long, y = lat, group =
           storm.id), colour = "red", alpha = 0.2, size = 0.2) +
40      annotate("rect", xmin = as.integer(rect.coords[1]),
41               xmax = as.integer(rect.coords[2]),
42               ymin = as.integer(rect.coords[3]),
43               ymax = as.integer(rect.coords[4]),
44               colour = "green", alpha = 0.2)
45  }
46
47  map_region_hurrs_small <- function(hurr.obs, years, coords, steps =
       c(5,5), xtra.lims = c(1.5,1.5)){
48    coords <- morph_coords(coords)
49    hurr.obs <- hurr.obs %>%
50      filter(storm.year %in% years)
51    world_map <- map_data("world")
52    world_map <- subset(world_map, region!="Antarctica")
53
54    ggplot(data = world_map, aes(x = long, y = lat, group = group)) +
55      geom_cartogram(map = world_map, aes(map_id = region)) +
56      scale_x_longitude(xmin = as.numeric(coords[1]),
57                        xmax = as.numeric(coords[2]),
58                        step = steps[1], xtra.lim = xtra.lims[1]) +
59      scale_y_latitude(ymin = as.numeric(coords[3]),
60                       ymax = as.numeric(coords[4]),
61                       step = steps[2], xtra.lim = xtra.lims[2]) +
62      coord_proj("+proj=merc") +
63      geom_path(data = hurr.obs, aes(x = long, y = lat, group =
           storm.id), colour = "red", alpha = 0.2, size = 0.2)
64  }
65
66  # PDI visualisation functions ------------------------------
67
68  # Plot the DPDI data frame
69  plot_dpdi_by_sst_class <- function(hurr.obs.pdi, ssts.df){
70    years <-
         year(ssts.df$year[1]):year(ssts.df$year[length(ssts.df$year)])
71
72    high.years <- get_high_years(ssts.df)
73    low.years <- get_low_years(ssts.df)
74    dpdi.high.df <- get_dpdi(hurr.obs.pdi, high.years)
75    dpdi.low.df <- get_dpdi(hurr.obs.pdi, low.years)
76
77    ggplot() +
78      aes(x = pdi.star, y = dpdi, ymin = dpdi-pdi.error, ymax =
           dpdi+pdi.error) +
79      geom_line(data = dpdi.high.df, aes(colour = "high"), linetype =
           "dotted") +
80      geom_point(data = dpdi.high.df, aes(colour = "high")) +
81      geom_errorbar(data = dpdi.high.df, aes(colour = "high"), width =
           0.1) +
82      geom_line(data = dpdi.low.df, aes(colour = "low"), linetype =
```

```
        "dotted") +
83    geom_point(data = dpdi.low.df, aes(colour = "low")) +
84    geom_errorbar(data = dpdi.low.df, aes(colour = "low"), width =
        0.1) +
85    scale_colour_manual(values = c("brown1", "dodgerblue1")) +
86    scale_x_log10() +
87    scale_y_log10() +
88    labs(title = paste0("PDI␣probability␣density␣for␣",
89                        years[1], "-", years[length(years)],
90                        "␣(", attr(ssts.df, "title"), ")"),
91         x = bquote(PDI~ (m^3 ~s^-2)),
92         y = bquote(D(PDI)~(s^2~m^-3)),
93         colour = "SST␣class")
94 }
95
96 # Plot PDI time series
97 plot_pdi_tempseries <- function(hurr.pdi, ssts){
98   hurr.high.pdi <- hurr.pdi %>% filter(storm.year %in%
       get_high_years(ssts))
99   hurr.low.pdi <- hurr.pdi %>% filter(storm.year %in%
       get_low_years(ssts))
100  years.str <- paste0(year(ssts$year[1]), "-",
       year(ssts$year[length(ssts$year)]))
101
102  ggplot() +
103    aes(x = as.Date(paste(storm.year, "01", "01", sep = "-")), y =
       storm.pdi, group = 1) +
104    geom_point(data = hurr.high.pdi, aes(colour = "high"), size =
       0.5) +
105    geom_point(data = hurr.low.pdi, aes(colour = "low"), size = 0.5)
       +
106    scale_colour_manual(values = c("brown1", "dodgerblue1")) +
107    scale_y_log10() +
108    labs(title = paste0("PDI␣Time␣series", "␣(", attr(ssts,
       "title"), ";␣", years.str, ")"),
109         x = "Time␣(year)",
110         y = bquote(PDI~ (m^3 ~s^-2)),
111         colour = "SST␣class" )
112 }
113
114 # Scatterplot functions ------------------------------------
115
116 # PDI scatterplots
117 plot_pdi_scatter_by_class <- function(hurr.pdi, ssts, class){
118   if(class == "high"){
119     hurr.pdi <- hurr.pdi %>%
120       filter(storm.year %in% get_high_years(ssts))
121   } else if(class == "low"){
122     hurr.pdi <- hurr.pdi %>%
123       filter(storm.year %in% get_low_years(ssts))
124   }
125   hurr.ds.pdi <- hurr.pdi %>%
```

```
126     filter(max.wind > 33)
127   hurr.nds.pdi <- hurr.pdi %>%
128     filter(max.wind <= 33)
129
130   lm.ds.y <- lm(log10(storm.pdi) ~ log10(conv_unit(storm.duration,
          "sec", "hr")), data = hurr.ds.pdi)
131   lm.nds.y <- lm(log10(storm.pdi) ~ log10(conv_unit(storm.duration,
          "sec", "hr")), data = hurr.nds.pdi)
132   lm.ds.x <- lm(log10(conv_unit(storm.duration, "sec", "hr")) ~
          log10(storm.pdi), data = hurr.ds.pdi)
133   lm.nds.x <- lm(log10(conv_unit(storm.duration, "sec", "hr")) ~
          log10(storm.pdi), data = hurr.nds.pdi)
134
135   years.str <- paste0(year(ssts$year[1]), "-",
          year(ssts$year[length(ssts$year)]))
136
137   gg <- ggplot() +
138     aes(x = conv_unit(storm.duration, "sec", "hr"), y = storm.pdi) +
139     geom_point(data = hurr.ds.pdi,
140                aes(colour = "developing"), shape = 1, size = 1) +
141     geom_point(data = hurr.nds.pdi,
142                aes(colour = "non-developing"), shape = 5, size = 1) +
143     geom_abline(aes(slope = coef(lm.nds.y)[[2]],
144                     intercept = coef(lm.nds.y)[[1]],
145                     colour = "non-developing", linetype = "y(x)")) +
146     geom_abline(aes(slope = 1/coef(lm.nds.x)[[2]],
147                     intercept =
                            -coef(lm.nds.x)[[1]]/coef(lm.nds.x)[[2]],
148                     colour = "non-developing", linetype = "x(y)")) +
149     geom_abline(aes(slope = coef(lm.ds.y)[[2]],
150                     intercept = coef(lm.ds.y)[[1]],
151                     colour = "developing", linetype = "y(x)")) +
152     geom_abline(aes(slope = 1/coef(lm.ds.x)[[2]],
153                     intercept =
                            -coef(lm.ds.x)[[1]]/coef(lm.ds.x)[[2]],
154                     colour = "developing", linetype = "x(y)")) +
155     scale_x_log10(breaks = c(25, 50, 100, 200, 400, 800)) +
156     scale_y_log10() +
157     labs(title = paste0("PDI␣vs␣duration␣scatterplot", "␣(",
            attr(ssts, "title"), ";␣", years.str ,")") ,
158          x = "Storm␣duration␣(h)", y = bquote(PDI~ (m^3 ~s^-2)),
159          colour = "System␣status", linetype = "Regression") +
160     guides(colour = guide_legend(order = 1, override.aes =
            list(linetype = c(0,0), shape = c(1,5))),
161              linetype = guide_legend(order = 2, override.aes =
                    list(colour = c("black", "black"))))
162
163   if(class == "high"){
164     gg +
165       scale_colour_manual(labels = c(bquote(.(paste0("developing;␣
            ")) ~ r^2 ~ .(paste0("=␣",
            format(summary(lm.ds.y)$r.squared, digits = 2)))),
```

```
            bquote(.(paste0("non-developing;␣")) ~ r^2 ~ .(paste0("=␣
            ", format(summary(lm.nds.y)$r.squared, digits = 2)))),
166                         values = c("developing" = "brown1",
                               "non-developing" = "darkviolet")) +
167         scale_linetype_manual(values = c("x(y)" = "longdash", "y(x)" =
              "solid"))
168     } else if(class == "low"){
169       gg +
170         scale_colour_manual(labels = c(bquote(.(paste0("developing;␣
              ")) ~ r^2 ~ .(paste0("=␣",
              format(summary(lm.ds.y)$r.squared, digits = 2)))),
              bquote(.(paste0("non-developing;␣")) ~ r^2 ~ .(paste0("=␣
              ", format(summary(lm.nds.y)$r.squared, digits = 2)))),
171                         values = c("developing" = "dodgerblue1",
                               "non-developing" = "purple")) +
172         scale_linetype_manual(values = c("x(y)" = "longdash", "y(x)" =
              "solid"))
173     }
174 }
175
176 plot_pdi_scatter_by_status <- function(hurr.pdi, ssts, status){
177     if(status == "ds"){
178       hurr.pdi <- hurr.pdi %>%
179         filter(max.wind > 33)
180
181     } else if(status == "nds"){
182       hurr.pdi <- hurr.pdi %>%
183         filter(max.wind <= 33)
184     }
185     hurr.high.pdi <- hurr.pdi %>%
186       filter(storm.year %in% get_high_years(ssts))
187     hurr.low.pdi <- hurr.pdi %>%
188       filter(storm.year %in% get_low_years(ssts))
189
190     lm.high.y <- lm(log10(storm.pdi) ~ log10(conv_unit(storm.duration,
          "sec", "hr")), data = hurr.high.pdi)
191     lm.low.y <- lm(log10(storm.pdi) ~ log10(conv_unit(storm.duration,
          "sec", "hr")), data = hurr.low.pdi)
192     lm.high.x <- lm(log10(conv_unit(storm.duration, "sec", "hr")) ~
          log10(storm.pdi), data = hurr.high.pdi)
193     lm.low.x <- lm(log10(conv_unit(storm.duration, "sec", "hr")) ~
          log10(storm.pdi), data = hurr.low.pdi)
194
195     years.str <- paste0(year(ssts$year[1]), "-",
          year(ssts$year[length(ssts$year)]))
196
197     gg <- ggplot() +
198       aes(x = conv_unit(storm.duration, "sec", "hr"), y = storm.pdi) +
199       geom_point(data = hurr.low.pdi, aes(colour = "low"), shape = 5,
            size = 1) +
200       geom_point(data = hurr.high.pdi, aes(colour = "high"), shape =
            1, size = 1) +
```

```
201     geom_abline(aes(slope = coef(lm.low.y)[[2]],
202                     intercept = coef(lm.low.y)[[1]],
203                     colour = "low", linetype = "y(x)")) +
204     geom_abline(aes(slope = 1/coef(lm.low.x)[[2]],
205                     intercept =
                           -coef(lm.low.x)[[1]]/coef(lm.low.x)[[2]],
206                     colour = "low", linetype = "x(y)")) +
207     geom_abline(aes(slope = coef(lm.high.y)[[2]],
208                     intercept = coef(lm.high.y)[[1]],
209                     colour = "high", linetype = "y(x)")) +
210     geom_abline(aes(slope = 1/coef(lm.high.x)[[2]],
211                     intercept =
                           -coef(lm.high.x)[[1]]/coef(lm.high.x)[[2]],
212                     colour = "high", linetype = "x(y)")) +
213     scale_x_log10(breaks = c(25, 50, 100, 200, 400, 800)) +
214     scale_y_log10() +
215     guides(colour = guide_legend(order = 1, override.aes =
           list(linetype = c(0,0), shape = c(1,5))),
216            linetype = guide_legend(order = 2, override.aes =
                  list(colour = c("black", "black")))) +
217     scale_colour_manual(labels = c(bquote(.(paste0("high;␣")) ~ r^2
           ~ .(paste0("=␣", format(summary(lm.high.y)$r.squared, digits
           = 2)))), bquote(.(paste0("low;␣␣")) ~ r^2 ~ .(paste0("=␣",
           format(summary(lm.low.y)$r.squared, digits = 2))))),
218                         values = c("high" = "brown1", "low" =
                               "dodgerblue1")) +
219     scale_linetype_manual(values = c("x(y)" = "longdash", "y(x)" =
           "solid")) +
220     labs(title = paste0("PDI␣vs␣duration␣scatterplot", "␣(",
           attr(ssts, "title"), ";␣", years.str ,")") ,
221          x = "Storm␣duration␣(h)", y = bquote(PDI~ (m^3 ~s^-2)),
222          colour = "SST␣class", linetype = "Regression")
223
224   if(status == "ds"){
225     gg
226   } else if(status == "nds"){
227     gg +
228       scale_y_log10(breaks = c(0.25*10^9, 0.5*10^9, 10^9,
             0.25*10^10, 0.5*10^10))
229   }
230 }
231
232 plot_pdi_scatter_wind <- function(hurr.pdi, ssts){
233   hurr.pdi <- hurr.pdi %>%
234     filter(max.wind > 33)
235   hurr.high.pdi <- hurr.pdi %>%
236     filter(storm.year %in% get_high_years(ssts))
237   hurr.low.pdi <- hurr.pdi %>%
238     filter(storm.year %in% get_low_years(ssts))
239
240   lm.high.y <- lm(log10(storm.pdi) ~ log10(conv_unit(max.wind,
        "knot", "m_per_sec")), data = hurr.high.pdi)
```

```
241   lm.low.y <- lm(log10(storm.pdi) ~ log10(conv_unit(max.wind,
          "knot", "m_per_sec")), data = hurr.low.pdi)
242   lm.high.x <- lm(log10(conv_unit(max.wind, "knot", "m_per_sec")) ~
          log10(storm.pdi), data = hurr.high.pdi)
243   lm.low.x <- lm(log10(conv_unit(max.wind, "knot", "m_per_sec")) ~
          log10(storm.pdi), data = hurr.low.pdi)
244
245   years.str <- paste0(year(ssts$year[1]), "-",
          year(ssts$year[length(ssts$year)]))
246
247   ggplot() +
248     aes(x = conv_unit(max.wind, "knot", "m_per_sec"), y = storm.pdi)
          +
249     geom_point(data = hurr.low.pdi, aes(colour = "low"), shape = 5,
          size = 1, position = "jitter") +
250     geom_point(data = hurr.high.pdi, aes(colour = "high"), shape =
          1, size = 1, position = "jitter") +
251     geom_abline(aes(slope = coef(lm.low.y)[[2]],
252                     intercept = coef(lm.low.y)[[1]],
253                     colour = "low", linetype = "y(x)")) +
254     geom_abline(aes(slope = 1/coef(lm.low.x)[[2]],
255                     intercept =
                            -coef(lm.low.x)[[1]]/coef(lm.low.x)[[2]],
256                     colour = "low", linetype = "x(y)")) +
257     geom_abline(aes(slope = coef(lm.high.y)[[2]],
258                     intercept = coef(lm.high.y)[[1]],
259                     colour = "high", linetype = "y(x)")) +
260     geom_abline(aes(slope = 1/coef(lm.high.x)[[2]],
261                     intercept =
                            -coef(lm.high.x)[[1]]/coef(lm.high.x)[[2]],
262                     colour = "high", linetype = "x(y)")) +
263     scale_x_log10(breaks = c(25, 50, 100, 200, 400, 800)) +
264     scale_y_log10() +
265     guides(colour = guide_legend(order = 1, override.aes =
          list(linetype = c(0,0), shape = c(1,5))),
266              linetype = guide_legend(order = 2, override.aes =
                    list(colour = c("black", "black")))) +
267     scale_colour_manual(labels = c(bquote(.(paste0("high;␣")) ~ r^2
          ~ .(paste0("=␣", format(summary(lm.high.y)$r.squared, digits
          = 2)))), bquote(.(paste0("low;␣␣")) ~ r^2 ~ .(paste0("=␣",
          format(summary(lm.low.y)$r.squared, digits = 2))))),
268                         values = c("high" = "brown1", "low" =
                               "dodgerblue1")) +
269     scale_linetype_manual(values = c("x(y)" = "longdash", "y(x)" =
          "solid")) +
270     labs(title = paste0("PDI␣vs␣wind␣speed␣scatterplot", "␣(",
          attr(ssts, "title"), ";␣", years.str ,")") ,
271          x = bquote(Maximum~wind~speed~ (m~s^-1)), y = bquote(PDI~
                (m^3 ~s^-2)),
272          colour = "SST␣class", linetype = "Regression")
273 }
```

Script A.5: `main_analysis.R` - Code to study the PDI dependence with the SST

```
# Source base code ----------------------------------------
source("hadisst_base.R")
source("hurdat2_pdi_base.R")
source("analysis_base.R")

# Create PDI data frame -----------------------------------

# Get hurricanes data frames
hurr.natl.obs <- get_hurr_obs("hurdat2-1851-2016-apr2017.txt")
hurr.epac.obs <- get_hurr_obs("hurdat2-nepac-1949-2016-apr2017.txt")
hurr.all.obs <- rbind(hurr.natl.obs, hurr.epac.obs)

# Create data frame with PDI and year of the storm
hurr.natl.pdi <- get_pdis(hurr.natl.obs) %>%
  filter(storm.id != "AL171988")
attr(hurr.natl.pdi, "title") <- "N. Atl."
hurr.epac.pdi <- get_pdis(hurr.epac.obs) %>%
  filter(storm.id != "EP231989")
attr(hurr.epac.pdi, "title") <- "E. Pac."
hurr.all.pdi <- get_pdis(hurr.all.obs)

# Create SST data frames ----------------------------------

hadsst.raster <- load_hadsst(file = "data/HadISST_sst.nc")

# Windows of activity
years.natl <- 1966:2016
coords.natl <- c("90W", "20E", "5N", "25N")
coords.natl.map <- c("100W", "20E", "0N", "60N")

years.epac <- 1966:2016
coords.epac <- c("120W", "90W", "5N", "20N")
coords.epac.map <- c("160W", "90W", "5N", "35N")

# Construct SST data frames
ssts.natl <- get_mean_ssts(years = years.natl, coords = coords.natl)
attr(ssts.natl, "title") <- "N. Atl."

ssts.epac <- get_mean_ssts(years = years.epac, coords = coords.epac)
attr(ssts.epac, "title") <- "E. Pac."

# Get list of low & high SST years
get_low_years(ssts.natl)
get_high_years(ssts.natl)
get_low_years(ssts.epac)
get_high_years(ssts.epac)

# Get number of years per SST class
table(ssts.natl$sst.class)
table(ssts.epac$sst.class)

```

```
52 # Get number of storms per SST class
53 length((hurr.natl.pdi %>% filter(storm.year %in%
       get_high_years(ssts.natl)))$storm.pdi)
54 length((hurr.natl.pdi %>% filter(storm.year %in%
       get_low_years(ssts.natl)))$storm.pdi)
55 length((hurr.epac.pdi %>% filter(storm.year %in%
       get_high_years(ssts.epac)))$storm.pdi)
56 length((hurr.epac.pdi %>% filter(storm.year %in%
       get_low_years(ssts.epac)))$storm.pdi)
57
58 # Basins maps -------------------------------------------
59
60 # Maps of the basins (full)
61 map_region_hurrs(hurr.natl.obs, years.natl, coords.natl.map,
       coords.natl, steps = c(20, 10), xtra.lims = c(3,2))cairo_pdf)
62 map_region_hurrs(hurr.epac.obs, years.epac, coords.epac.map,
       coords.epac, steps = c(10, 10), xtra.lims = c(3,2))cairo_pdf)
63
64 # SST map of a raster layer
65 map_global_sst(hadsst.raster, 12, 2015)
66
67 # Plot SSTs and PDIs ------------------------------------
68
69 # Plot annual SSTs
70 plot_annual_sst(ssts.natl)
71 plot_annual_sst(ssts.epac)
72
73 # PDI time series
74 plot_pdi_tempseries(hurr.natl.pdi, ssts.natl)
75 plot_pdi_tempseries(hurr.epac.pdi, ssts.epac)
76
77 # DPDI plots
78 plot_dpdi(hurr.natl.pdi, years.natl)
79 plot_dpdi(hurr.epac.pdi, years.epac)
80
81 # DPDI plots by SST class
82 plot_dpdi_by_sst_class(hurr.natl.pdi, ssts.natl)
83 plot_dpdi_by_sst_class(hurr.epac.pdi, ssts.epac)
84
85 # Scatterplots analysis ---------------------------------
86
87 # PDI scatterplots (duration)
88 plot_pdi_scatter_by_status(hurr.natl.pdi, ssts.natl, "ds")
89 plot_pdi_scatter_by_status(hurr.natl.pdi, ssts.natl, "nds")
90 plot_pdi_scatter_by_status(hurr.epac.pdi, ssts.epac, "ds")
91 plot_pdi_scatter_by_status(hurr.epac.pdi, ssts.epac, "nds")
92
93 # PDI scatterplots (wind)
94 plot_pdi_scatter_wind(hurr.natl.pdi, ssts.natl)
95 plot_pdi_scatter_wind(hurr.epac.pdi, ssts.epac)
```

Script A.6: `hadcrut4_analysis.R` - Code to study the global average temperature (from HadCRUT4)

```
library(tidyverse)
library(lubridate) # Use dates

# Read and clean raw data ----------------------------------

# Read the raw data
temps.data <-
    read.table("data/HadCRUT.4.5.0.0.annual_ns_avg_smooth.txt")

# Clean the raw data
temps.data <- temps.data %>%
  dplyr::select(V1, V2) %>%
  rename(year = V1, temp = V2) %>%
  mutate(year = ymd(paste(year, "01", "01", sep = "-")))

# Data visualisation ---------------------------------------

# Find absolute minimum
filter(temps.data, temp == min((temps.data %>% filter(year(year)
    %in% 1966:2016))$temp))

# Plot global mean temperature anomalies
plot_global_temperature <- function(data.df){
  years.str <- paste0(year(data.df$year[1]), "-",
      year(data.df$year[length(data.df$year)]))

  ggplot(data.df) +
    aes(x = year, y = temp) +
    geom_hline(aes(yintercept = 0, linetype = "1961-90 mean"),
        colour = "blueviolet") +
    geom_line(aes(linetype = "Annual"), colour = "black") +
    geom_point(data=temps.data[126, ], aes(x = year, y = temp),
        colour="red", size=2) +
    scale_linetype_manual(values = c("twodash", "solid")) +
    labs(title = paste0("Global mean temperature between ",
        years.str),
         x = "Time (year)", y = "Temperature anomaly (C)", linetype
             = "Temperature") +
    guides(linetype = guide_legend(override.aes = list(colour =
        c("blueviolet", "black"))))
}

plot_global_temperature(temps.data)
```

References

[1] Á. Corral, A. Ossó and J. E. Llebot. Scaling of tropical-cyclone dissipation. *Nature Physics*, 6(9):693–696, 2010. ISSN: 1745-2473. URL: http://www.nature.com/doifinder/10.1038/nphys1725.

[2] K. A. Emanuel. An Air-Sea Interaction Theory for Tropical Cyclones. Part I: Steady-State Maintenance. *Journal of the Atmospheric Sciences*, 43(6):585–605, 1986. ISSN: 0022-4928. URL: https://doi.org/10.1175/1520-0469(1986)043%3C0585:AASITF%3E2.0.CO;2.

[3] K. A. Emanuel. Increasing destructiveness of tropical cyclones over the past 30 years. *Nature*, 436(7051):686–688, 2005. ISSN: 0028-0836. URL: http://www.nature.com/doifinder/10.1038/nature03906.

[4] K. A. Emanuel. Tropical Cyclones. *Annual Review of Earth and Planetary Sciences*, 31(1):75–104, 2003. ISSN: 0084-6597. URL: http://www.annualreviews.org/doi/10.1146/annurev.earth.31.100901.141259.

[5] National Hurricane Center. URL: http://www.nhc.noaa.gov/.

[6] HURDAT Re-Analysis Project. URL: http://www.aoml.noaa.gov/hrd/hurdat/Data_Storm.html.

[7] C. Landsea and J. Franklin. Atlantic Hurricane Database Uncertainty and Presentation of a New Database Format. *Monthly Weather Review*, 141(10):3576–3592, 2013. ISSN: 0027-0644. URL: http://journals.ametsoc.org/doi/abs/10.1175/MWR-D-12-00254.1.

[8] North Atlantic Hurricane Basin (1851–2016) Comparison of Original and Revised HURDAT. URL: http://www.aoml.noaa.gov/hrd/hurdat/comparison_table.html.

[9] C. Landsea, J. Franklin and J. Bevin. The revised Atlantic hurricane database (HURDAT2), 2014. URL: http://www.nhc.noaa.gov/data/hurdat/hurdat2-format-atlantic.pdf.

[10] C. Landsea et al. The revised Northeast and North Central Pacific hurricane database (HURDAT2), 2016. URL: http://www.nhc.noaa.gov/data/hurdat/hurdat2-format-nencpac.pdf.

[11] P. J. Webster et al. Changes in tropical cyclone number, duration, and intensity in a warming environment. *Science (New York, N.Y.)*, 309(5742):1844–6, 2005. ISSN: 1095-9203. URL: http://www.ncbi.nlm.nih.gov/pubmed/16166514.

[12] S. Hergarten. *Self-Organized Criticality in Earth Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002. ISBN: 978-3-642-07790-6. URL: http://link.springer.com/10.1007/978-3-662-04390-5.

[13] K. Trenberth. Uncertainty in Hurricanes and Global Warming. *Science*, 308(5729):1753–1754, 2005. URL: http://science.sciencemag.org/content/sci/308/5729/1753.full.pdf.

[14] NCAR Climate Data Guide Content with Tag: SST - sea surface temperature. URL: https://climatedataguide.ucar.edu/variables/ocean/sst-sea-surface-temperature.

[15] N. A. Rayner. Global analyses of sea surface temperature, sea ice, and night marine air temperature since the late nineteenth century. *Journal of Geophysical Research*, 108(D14):4407, 2003. ISSN: 0148-0227. URL: http://doi.wiley.com/10.1029/2002JD002670.

[16] Weather and climate change - Met Office. URL: http://www.metoffice.gov.uk/.

[17] Hadley Centre Sea Ice and Sea Surface Temperature data set (HadISST). URL: http://www.metoffice.gov.uk/hadobs/hadisst/.

[18] Hadley Centre Sea Ice and Sea Surface Temperature data set (HadISST.2). URL: http://www.metoffice.gov.uk/hadobs/hadisst2/.

[19] HadISST data format instructions. URL: http://www.metoffice.gov.uk/hadobs/hadisst/data/Read_instructions_sst.txt.

[20] HadCRUT4 Diagnostics. URL: http://www.metoffice.gov.uk/hadobs/hadcrut4/diagnostics.html.

[21] J. L. McBride. *Observational analysis of tropical cyclone formation*. Colorado State University, 1979. URL: http://nla.gov.au/nla.cat-vn1301003.

[22] C. Domingo. *Lecture notes in "Iniciació a la Física Experimental"*. Departament de Física (UAB), 2012.