

“Whale wave”: Preparing 2004-2014 data

Eric Keen

February 27, 2016

Contents

1	Intro	1
2	Blocking up the study area	1
3	Sightings	2
3.1	Basic formatting	4
3.2	Assigning geographic blocks	4
3.3	Determine distance into fjord	5
4	Effort	6
4.1	Gitga’at Data	6
4.1.1	Create Effort Maps	6
4.1.2	Draw survey tracklines	8
4.1.3	Interpolate survey tracks	9
4.1.4	Bin effort and sightings into blocks	9
4.2	NCCS data	11
4.2.1	Route maps	12
4.2.2	Interpolate survey tracks	12
4.2.3	Bin effort and sightings into blocks	12
5	Combine datasets	13
6	Dataset summaries	14
6.1	Summary tables	14
6.2	Effort trends	15
7	Useful Functions	16
7.1	Julian Day	16

1 Intro

We are using 10 years of boat-based surveys by the Gitga’at First Nation and North Coast Cetacean Society (NCCS).

2 Blocking up the study area

A seasonal distribution shift is a pattern in both time and space. To observe it, we needed to divide the study area into small geographic strata. We explored several candidate schemes (Fig. 1). The fjord system

is naturally compartmentalized by islands and underwater sills that divide the fjord into discrete channels. These channels were further stratified according to the following considerations. The more numerous the strata, the higher the statistical power and the more successful we are in capturing habitat heterogeneity. However, if strata are too small in area (which would happen if they were too numerous), effort calculations may become less meaningful since many survey routes had to be estimated based on field notes. Moreover, humpback density differences between adjacent strata would be more likely a result of random chance than ecologically relevant patterns. Strata design must strike a balance between statistical power and meaningful results. We aimed for 2-4 boxes per channel per sound. Finally, areas of typically high effort (e.g., thoroughfares, entries to headquarters) were deliberately separated from those of lower effort so that sightings in low-effort areas don't get scaled down by a falsely high effort score, and vice-versa.

These are the data sources for all the various blocking schemes we tried: In the end, we pooled surveys and sightings into 26 geographic strata (Fig. 2, left). GPS coordinates for candidate block polygons are in folder "surveyblocks". The final blocking scheme was prepared in this R file:

```
> source("./surveyblocks/block-delineate.R",echo=FALSE)
```

This R file is used to produce two versions of the blocks: the simple rectangles of each block and the shore polygons those blocks contain. The simple rectangle boundaries for the blocks are output to the "surveyblocks" folder in this file:

```
> head(read.csv("./surveyblocks/tribounds.csv"))
```

	left	right	top	bottom	block	base
1	-129.307	-129.100	52.940	52.500	caas	caa
2	-129.550	-129.307	52.955	52.500	caac	caa
3	-129.580	-129.314	53.040	52.955	caan	caa
4	-129.900	-129.430	53.110	53.040	ests	est
5	-129.900	-129.500	53.170	53.110	estc	est
6	-129.900	-129.530	53.280	53.170	estn	est

The rectangle boundaries in Fig. 1 *left* are what is used to bin the humpback sightings into each geographic block (more on sighting bins below). Note that the boundaries for the rectangles had to be slightly different than the boundaries used to define the shoreline polygons, due to constraints in the shoreline coordinate data (which I drew in the online mapping site www.geojson.io and saved to the file "channel-spolygons.csv", which is stored in the "surveyblocks" folder. I prepared this .csv file using the R file "channelspolyprep.R", in the same folder). The file that contains boundaries for the 26 *polygons* themselves is "tribounds-poly.csv". The above R file reads in this and the "channel-spolygons.csv" files in order to produce the shoreline polygons for each of the 26 sub-blocks. The shoreline polygon coordinates for the blocks are saved to the "surveyblocks" folder in file "triblocks.csv" – it's large (23.2 MB) because the polygon is interpolated.

3 Sightings

Sightings from both platforms were delivered to me in one document: "hw-sightings-nccs-gg.csv". Gitga'at sightings were noted as "GG", NCCS sightings were noted as "NCCS".

```
> head(read.csv("./sightings/hw-sightings-nccs-gg.csv"))[,1:7]
```

	Observer	DATE	SPECIES	Latitude	Longitude	GROUP.SIZE.BEST	GROUP.TYPE
1	GG	April 30, 2005	HW	53.10127	-129.1839	1	A
2	GG	May 12, 2005	HW	53.27220	-129.4065	2	A
3	GG	May 12, 2005	HW	53.27220	-129.4065	2	A
4	GG	May 12, 2005	HW	53.27700	-129.4289	1	A
5	GG	June 6, 2005	HW	53.44972	-128.8999	1	A
6	GG	June 6, 2005	HW	53.44007	-128.9044	1	A

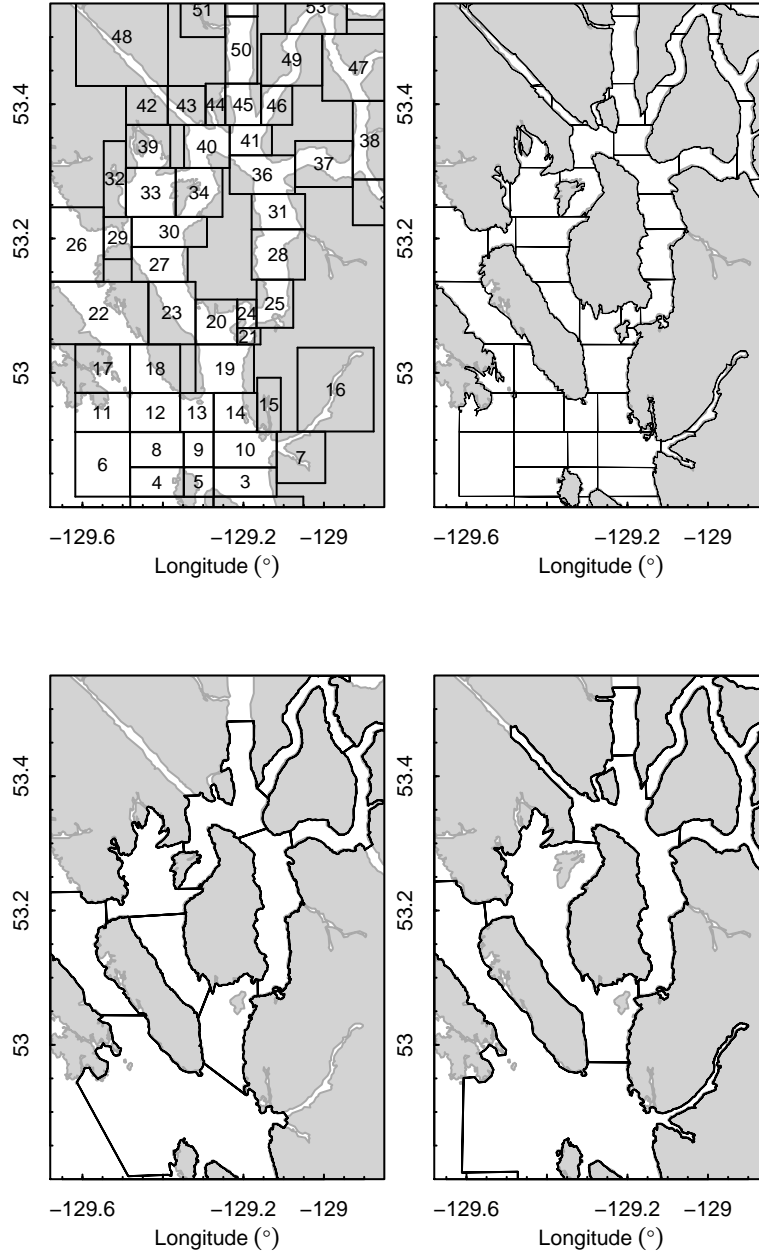


Figure 1: Three schemes considered for stratifying the study area, but not used. The top is a 54-block design. Bottom left is 8 blocks (the 8 main channels in the study area). Bottom right is the pooling of those blocks into 4 “provinces”.

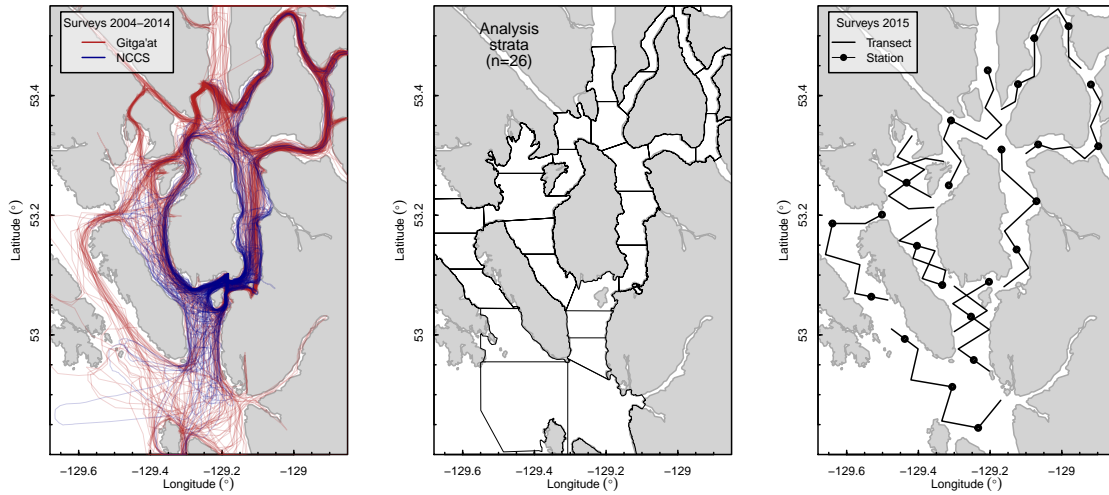


Figure 2: The final blocking scheme used in the HW Wave paper (center), flanked by survey routes. Left: NCCS (blue) and Gitga'at (red). Right: Bangarang.

3.1 Basic formatting

In the following R file I did some basic formatting, standardizing GPS and date formats and preparing columns for mapping in R. The NCCS sightings were already formatted such that each sighting occurred on a single row. However, Gitga'at sightings before 2007 had a single row for every individual in the sighting. So I reduced Gitga'at data to a single row for every sighting, ensuring that the group size column was accurate.

```
> source("./sightings/hwsits-nccs-gggw-format.R", echo=FALSE)
```

That R file produced an updated sightings file:

```
> head(read.csv("./sightings/hwsits-formatted.csv"))
```

	EID	X	Y	platform	date	sp	grp	dem	id	comm	cex
1	1	-129.1839167	53.10127	GG	2005-04-30	HW	1	A			0.4
2	2	-129.4065167	53.27220	GG	2005-05-12	HW	2	A			0.6
3	3	-129.4289	53.27700	GG	2005-05-12	HW	1	A			0.4
4	4	-128.8999333	53.44972	GG	2005-06-06	HW	1	A			0.4
5	5	-128.90445	53.44007	GG	2005-06-06	HW	1	A			0.4
6	6	-129.2584833	53.02140	GG	2005-06-06	HW	1	A			0.4

	pcol	doy	doycol
1	firebrick	120	#E80000
2	firebrick	132	#FF1500
3	firebrick	132	#FF1500
4	firebrick	157	#FF7200
5	firebrick	157	#FF7200
6	firebrick	157	#FF7200

3.2 Assigning geographic blocks

I determined which geographic block each sighting fell into using this R script:

```
> source("./sightings/hwsits-assignblocks.R")
```

This R script relies on a .csv that lists the GPS coordinates of the blocks:

```
> head(read.csv("./surveyblocks/tribounds.csv"))
```

	left	right	top	bottom	block	base
1	-129.307	-129.100	52.940	52.500	caas	caa
2	-129.550	-129.307	52.955	52.500	caac	caa
3	-129.580	-129.314	53.040	52.955	caan	caa
4	-129.900	-129.430	53.110	53.040	ests	est
5	-129.900	-129.500	53.170	53.110	estc	est
6	-129.900	-129.530	53.280	53.170	estn	est

And outputs an updated sightings file with a column for “block”:

```
> head(read.csv("./sightings/hwsits-nccs-gggw.csv"))
```

	X.1	EID	X	Y	platform	date	sp	grp	dem	id	comm	cex
1	1	1	-129.1839167	53.10127	GG	2005-04-30	HW	1	A			0.4
2	2	2	-129.4065167	53.27220	GG	2005-05-12	HW	2	A			0.6
3	3	3	-129.4289	53.27700	GG	2005-05-12	HW	1	A			0.4
4	4	4	-128.8999333	53.44972	GG	2005-06-06	HW	1	A			0.4
5	5	5	-128.90445	53.44007	GG	2005-06-06	HW	1	A			0.4
6	6	6	-129.2584833	53.02140	GG	2005-06-06	HW	1	A			0.4

	pcol	doy	doycol	block
1	firebrick	120	#E80000	cmpn
2	firebrick	132	#FF1500	squz
3	firebrick	132	#FF1500	squz
4	firebrick	157	#FF7200	mckn
5	firebrick	157	#FF7200	mckn
6	firebrick	157	#FF7200	cmpc

3.3 Determine distance into fjord

I passed this data set to another computer to calculate the “swimming distance” of each humpback sighting from the ocean. I did this using the function “routeKFS” in the new R package “bangarang.” For each sighting, swimming distance was calculated to 4 points spaced across the entrance to Caamano Sound. The minimum of these 4 swimming distances was considered the group’s “distance into the fjord”. This processing took several days. I brought the resulting file back to this file structure and saved it here:

```
> head(read.csv("./sightings/hw-od-ggnccs.csv"))
```

	EID	X	Y	platform	date	sp	grp	dem	id	comm	cex
1	1	-129.1839167	53.10127	GG	2005-04-30	HW	1	A			0.4
2	2	-129.4065167	53.27220	GG	2005-05-12	HW	2	A			0.6
3	3	-129.4289	53.27700	GG	2005-05-12	HW	1	A			0.4
4	4	-128.8999333	53.44972	GG	2005-06-06	HW	1	A			0.4
5	5	-128.90445	53.44007	GG	2005-06-06	HW	1	A			0.4
6	6	-129.2584833	53.02140	GG	2005-06-06	HW	1	A			0.4

	pcol	doy	doycol	OD
1	firebrick	120	#E80000	18.7986
2	firebrick	132	#FF1500	25.6842
3	firebrick	132	#FF1500	25.4446
4	firebrick	157	#FF7200	46.7086
5	firebrick	157	#FF7200	47.0544
6	firebrick	157	#FF7200	12.9802

4 Effort

4.1 Gitga'at Data

Gitga'at survey route data came to me in the form of a spreadsheet with time- and GPS-stamped effort/event updates that were logged manually throughout a survey:

```
> head(read.csv("./effortprocessing/effort-gg-05-15.csv"))
```

	X24. Jan.05	X	B	X14.18	X53	X25.334	X129	X14.397	X3	X1	N	X.1	EX
1	24-Jan-05	X		14:24	53	24.013	129	12.393	3	1	N		EX
2	24-Jan-05	O		14:34	53	23.879	129	12.341	3	1	N		EX
3	24-Jan-05	X		15:25	53	23.766	128	59.992	3	1	N		EX
4	24-Jan-05	O		15:33	53	32.582	128	0.248	3	1	N		EX
5	24-Jan-05	W		15:56	53	26.504	128	55.522	0	0	N		EX
6	24-Jan-05	X		16:08	53	28.218	128	50.224	0	1	N		EX

```

X.2
1 HYDROPHONE CHECK NO WHALES HEARD
2 RESTART AFTER HYDROPHONE CHECK
3 HYDROPHONE CHECK @ ? DEVESTATION
4 RESTART- 2 SEALIONS SEEN
5 ENTER BISHOP BAY
6 BISHOP BAY STOP

```

4.1.1 Create Effort Maps

Based on these manual notes, I needed to infer and interpolate the route taken during each survey. To do so, I first did some basic processing in this R file:

```
> source("./effortprocessing/effort-gg-prep.R")
```

This R file outputs an reformatted effort spreadsheet...

```
> head(read.csv("./effortprocessing/eff-gg.csv")) # simple reformatting
```

	PID	POS	X	Y	survey	date	ev	crew
1	1	1	-129.23995	53.4222333333333	1	2005-01-24 14:18:00	B	
2	1	2	-129.20655	53.4002166666667	1	2005-01-24 14:24:00	X	
3	1	3	-129.205683333333	53.3979833333333	1	2005-01-24 14:34:00	O	
4	1	4	-128.999866666667	53.3961	1	2005-01-24 15:25:00	X	
5	1	5	-128.004133333333	53.5430333333333	1	2005-01-24 15:33:00	O	
6	1	6	-128.925366666667	53.4417333333333	1	2005-01-24 15:56:00	W	

```

bft swell prec vis qual notes raw.notes
1 3 1 N EX
2 3 1 N EX HYDROPHONE CHECK NO WHALES HEARD
3 3 1 N EX RESTART AFTER HYDROPHONE CHECK
4 3 1 N EX HYDROPHONE CHECK @ ? DEVESTATION
5 3 1 N EX RESTART- 2 SEALIONS SEEN
6 0 0 N EX ENTER BISHOP BAY

```

...and also outputs maps of each survey date on the spreadsheet. These maps contain the locations of HW sightings from that date as well as the locations of manual effort updates, color-coded by time of day so that I could look at the map and get a sense for where the survey began and the order in which it completed its route (example in Fig. 3). These figures were saved to folder "effortprocessing>ggroutes-01-effmap".

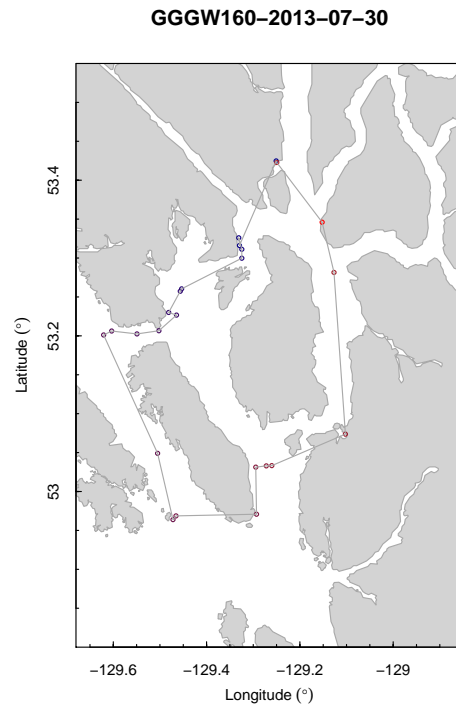


Figure 3: Map of manual effort entries from a Gitga'at survey. I used these maps to infer and interpolate tracklines for each survey. Effort update locations are colored in a blue-to-red gradient from early in the day to later in the day.

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "stroke": "#555555",
        "stroke-width": 2,
        "stroke-opacity": 1,
        "id": 1
      },
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [
            -129.25277709960938,
            53.42017241083368
          ],
          [
            -129.22119140625,
            53.40871323837144
          ]
        ]
      }
    }
  ]
}
```

Figure 4: Screenshot of the output format of trackline drawings from www.geojson.io.

4.1.2 Draw survey tracklines

I then took these effort maps and used them to approximate each survey's trackline. Most of these were straightforward, despite many typo's in the GPS entries for the effort updates. I adopted a shortest probable route approach. While there could have been some error in these tracklines, I doubt that they were significant.

I drew the tracklines in the online mapping app www.geojson.io. This site saved the tracks as a kind of kmz object, but in order for me to use them in R I just saved them as .txt files. An example of the geojson output is in Fig. 4. I saved these drawn tracklines to "effortprocessing>ggroutes-02-raw". I then had to convert this geospatial formatting to basic spreadsheet format for me to use for mapping and analysis in R. To do that I used this R script:

```
> source("./effortprocessing/gg-raw2csv.R")
```

This R file saved all the routes from the same year into a single .csv, in "effortprocessing>ggroutes-03-csv". Here's an example of one of those files:

```
> head(read.csv("./effortprocessing/ggroutes-03-csv/ggroutes-2010.csv"))
```

	PID	POS	X	Y	YEAR	date
1	20100423	1	-129.2899	53.27917	2010	2010-04-23
2	20100423	2	-129.3063	53.29067	2010	2010-04-23
3	20100423	3	-129.3626	53.29067	2010	2010-04-23
4	20100423	4	-129.3983	53.28164	2010	2010-04-23
5	20100423	5	-129.4148	53.25125	2010	2010-04-23
6	20100423	6	-129.4450	53.21672	2010	2010-04-23

The column "PID" in these files is a unique survey ID number for Gitga'at surveys.

One of the dates of trackline drawings, 2008, needed extra formatting. I did that formatting with the following R file and output the results into the same folder as the other annual .csv's: "effortprocessing>ggroutes-03-csv".

```
> source("./effortprocessing/gg-2008POSfix.R")
```

For quality control, I produced maps of every survey route I drew after it had been converted from the raw .kmz format. The following R file saved these maps to the folder "effortprocessing>ggroutes-04-routemap" (Example map in Fig. 5).

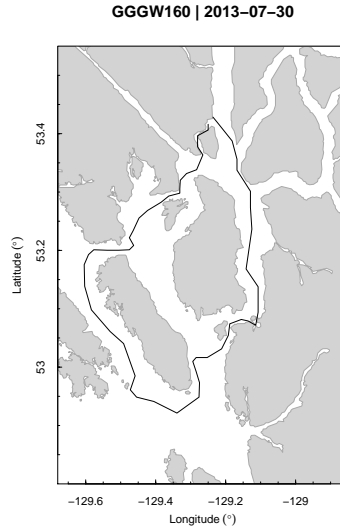


Figure 5: Map of Gitga'at survey route. A map was produced for every drawn trackline, for quality control.

```
> source("./effortprocessing/gg-mapeachroute.R")
```

I also produced trackline maps for each year of Gitga'at effort with the following R file. These annual maps were saved to "effortprocessing>ggroutes-05-yearmap" (e.g., Fig. 6).

```
> source("./effortprocessing/gg-mapeachyr.R")
```

4.1.3 Interpolate survey tracks

In order to carry out the steps coming up, I had to interpolate the survey tracks I drew so that tracks were made up of closely-spaced GPS coordinates. I performed that interpolation with this R script:

```
> source("./effortprocessing/gg-interpolate.R")
```

Interpolated routes were saved to "effortprocessing>ggroutes-06-interp" and files had the same format as before interpolation, e.g.:

```
> head(read.csv("./effortprocessing/ggroutes-06-interp/GGGW20110804.csv"))
```

	PID	POS	X	Y	YEAR	date
1	20110804	1	-129.3105	52.83762	2011	2011-08-04
2	20110804	2	-129.3106	52.83789	2011	2011-08-04
3	20110804	3	-129.3106	52.83816	2011	2011-08-04
4	20110804	4	-129.3107	52.83843	2011	2011-08-04
5	20110804	5	-129.3108	52.83870	2011	2011-08-04
6	20110804	6	-129.3109	52.83897	2011	2011-08-04

4.1.4 Bin effort and sightings into blocks

For each Gitga'at survey, I determined the trackline distance covered in each geographic block. At the same time, I determined the number of humpbacks seen in each geographic block during that survey. To safeguard against typos, I did not allow humpback sightings to be counted if no effort occurred in a block.

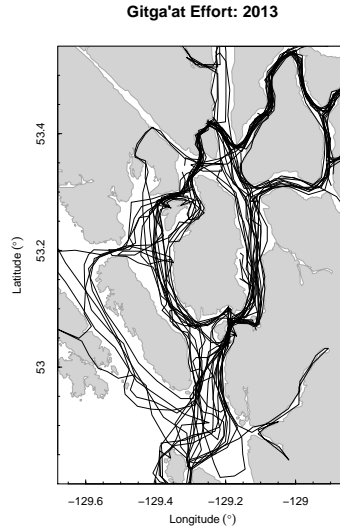


Figure 6: Map of Gitga'at survey routes for a single year. A map like this was produced for every year, 2004-2014.

```
> source("./effortprocessing/ggroutes-blocked.R")
```

This R script generates two files: one for effort and one for sightings. In each file the format is the same: every row is a survey. Every column is a geographic block bearing its namesake (e.g., "caas", "sqs", etc.). For the effort file, each cell value is the kilometers travelled during survey ROW in block COLUMN. For the sightings file, each cell value is the total number of humpbacks seen in that survey-block. These files are saved to the folder "effortprocessing".

```
> read.csv("./effortprocessing/BLOCKED-EFF-GGGW.csv")[1:10,10:20]
```

	cmpn	squs	squc	squn	squz	whas	whac	whan
1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	2.818173
2	0.000000	0.000000	0.000000	15.439295	11.726540	0.000000	0.000000	0.000000
3	10.367538	4.381576	3.299353	2.140752	0.000000	4.632058	5.375143	5.300075
4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	5.038048
5	6.832208	5.633719	3.318982	2.358006	0.000000	3.969878	5.428098	4.847253
6	6.843654	4.526121	3.290562	1.808788	0.000000	7.443981	5.393006	11.643410
7	6.653376	5.437817	3.362968	2.250237	0.000000	6.967915	5.377403	5.412559
8	8.198692	0.000000	0.000000	0.000000	0.000000	5.367075	5.381164	4.945517
9	4.710135	5.061577	3.403121	5.831715	3.640155	4.943512	5.413605	5.084520
10	6.597495	5.045847	3.335596	4.890555	5.920248	4.755966	5.358978	5.126841
	wris	wric	wrin					
1	0.000000	0.000000	3.428755					
2	1.209621	18.820860	6.019021					
3	8.982716	6.304389	3.252436					
4	0.000000	0.000000	11.676257					
5	5.758177	6.109676	4.440564					
6	0.903586	0.000000	16.454101					
7	5.644485	5.604766	3.136996					
8	0.000000	0.000000	2.922859					
9	1.591752	6.902900	2.849401					
10	0.000000	6.180760	3.573229					

```
> read.csv("./effortprocessing/BLOCKED-SIT-GGGW.csv")[130:140,10:20]
```

	cmpn	squs	squc	squn	squz	whas	whac	whan	wris	wric	wrin
130	0	0	0	0	0	0	0	1	0	0	0
131	0	0	0	0	0	3	0	0	0	0	0
132	9	1	0	0	0	0	0	5	0	0	0
133	0	0	0	0	0	0	0	0	0	0	0
134	0	0	0	0	0	0	0	0	0	0	0
135	0	0	0	0	0	0	0	0	0	0	0
136	0	0	0	0	0	0	0	2	0	0	0
137	0	0	0	0	0	0	0	0	0	0	0
138	0	0	0	0	0	0	0	0	0	0	0
139	0	0	0	0	0	0	0	0	0	0	0
140	0	0	0	0	0	0	0	2	0	0	0

4.2 NCCS data

For 2008-2012, NCCS survey route data came to me in the form of GIS tracklines drawn by collaborators at the World Wildlife Fund (WWF) and formatted into .kmz by Kim-Ly Thompson. WWF drew these tracklines much like I drew those for the Gitga'at: they inferred and interpolated survey routes using detailed notes from NCCS archives in addition to sighting locations. These .kmz files were stored in folder ">effortprocessing>nccsroutes-02-raw". There is a file for each year.

In order to re-format 2008-2012 kmz's into standard spreadsheet style, I had to write customized R routines specific to each year due to slight inconsistencies in date and note formatting. The .kmz's are saved to .txt files in this folder: ">effortprocessing>nccsroutes-02-txt".

```
> source("./effortprocessing/nccs-raw2csv-2008.R")
> source("./effortprocessing/nccs-raw2csv-2009.R")
> source("./effortprocessing/nccs-raw2csv-2010.R")
> source("./effortprocessing/nccs-raw2csv-2011.R")
> source("./effortprocessing/nccs-raw2csv-2012.R")
```

Stage 2 of the .kmz to .csv conversion took place in this R file:

```
> source("./effortprocessing/nccs-raw2csv-corrected.R")
```

This R script saves .csv's of each year's collection of routes to ">effortprocessing>nccsroutes-03-csv". It also prints the annual maps of tracks in "> effortprocessing > nccsroutes-05-yearmap" (e.g., Fig 7). After this automatic processing, some of the files did not look quite right, so I went in and double-checked and corrected my processing mistakes with this R script:

```
> source("./effortprocessing/nccs-correct-csvs")
```

This R script saves finalized, corrected route .csvs for 2008-2012 to ">effortprocessing>nccsroutes-04-correct".

For 2006-7 and 2013-4, NCCS survey route data came to me as written descriptions based on NCCS archived field logs. Janie wrote these logs into an Excel spreadsheet for each year, which I stored in "> effortprocessing > nccsroutes-01-xl-06071314". I used these spreadsheets to draw tracklines in the online GIS, as I did for the Gitga'at tracks. These .kmz tracklines are stored in "> effortprocessing > nccsroutes-02-raw-06071314". These were converted to .csv format and stored in "> effortprocessing > nccsroutes-03-csv-06071314". I made minor corrections and stored corrected files in "> effortprocessing > nccsroutes-04-correct-06071314". I then combined the two folders of different sets of years into a single folder containing all the NCCS years of data in a single place: "> effortprocessing > nccsroutes-05-all". Here are the R files used in these steps:

```
> source("./effortprocessing/nccs-0607-prep.R")
> source("./effortprocessing/nccs-raw2csv-06071314.R")
```

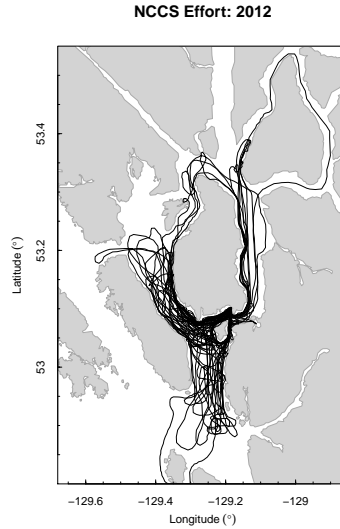


Figure 7: Map of NCCS survey routes for a single year. A map like this was produced for every year, 2004-2014.

4.2.1 Route maps

For quality control, I produced maps of every NCCS survey route. The following R file saved these maps to the folder "effortprocessing>nccsroutes-05-routemap" (Example map in Fig. 8).

```
> source("./effortprocessing/nccs-mapeachroute.R")
```

4.2.2 Interpolate survey tracks

```
> source("./effortprocessing/nccs-interpolate.R")
```

Interpolated routes were saved to "effortprocessing>nccs-06-interp" and files had the same format as before interpolation, e.g.:

```
> head(read.csv("./effortprocessing/nccsroutes-06-interp/NCCS20120810.csv"))
```

	PID	POS	X	Y	YEAR	date
1	20120810	1	-129.1840	53.10168	2012	2012-08-10
2	20120810	2	-129.1841	53.10167	2012	2012-08-10
3	20120810	3	-129.1841	53.10165	2012	2012-08-10
4	20120810	4	-129.1841	53.10163	2012	2012-08-10
5	20120810	5	-129.1842	53.10161	2012	2012-08-10
6	20120810	6	-129.1842	53.10159	2012	2012-08-10

4.2.3 Bin effort and sightings into blocks

This is the dataset, finalized, before being pooled with Gitga'at data for randomization analyses.

```
> source("./effortprocessing/nccsroutes-blocked.R")
```

Output files:

```
> read.csv("./effortprocessing/BLOCKED-EFF-NCCS.csv") [232:238,19:25]
```

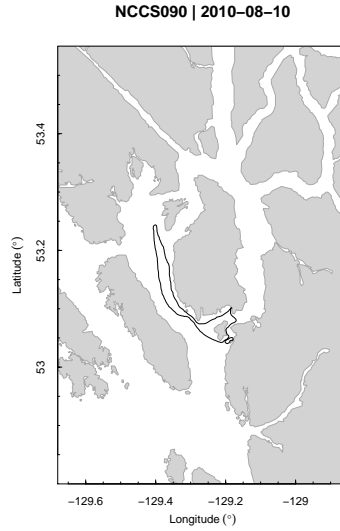


Figure 8: Map of NCCS survey route. A map was produced for every drawn trackline, for quality control.

	wric	wrin	wriz	mcks	mckc	mckn	vers
232	0.000000	0.000000	0	0	0	0	0.000000
233	0.000000	0.000000	0	0	0	0	0.000000
234	2.956580	1.192965	0	0	0	0	0.000000
235	0.000000	0.000000	0	0	0	0	0.000000
236	0.000000	0.000000	0	0	0	0	0.000000
237	0.000000	0.000000	0	0	0	0	0.000000
238	4.069036	9.515946	0	0	0	0	5.583714

```
> read.csv("./effortprocessing/BLOCKED-SIT-NCCS.csv")[232:238,9:15]
```

	cmpc	cmpn	squs	squc	squn	squz	whas
232	0	0	3	0	0	0	0
233	1	0	0	0	0	0	0
234	0	1	4	0	0	0	0
235	0	0	0	0	0	0	13
236	0	5	0	0	0	0	8
237	0	20	0	0	0	0	0
238	0	8	0	0	11	0	8

5 Combine datasets

The blocked datasets of effort and sightings from each platform were then combined into annual datasets, to verify that the Whale Wave happened each year.

```
> source("./effortprocessing/annual-density-gggw-nccs.R")
> list.files(path="./effortprocessing/blocked-years")
```

The blocked datasets of effort and sightings from each platform were then combined into a single pair of datasets: one for effort, one for sightings. These combined datasets were then sent over to the Randomization routines (which has its own Workflow document).

```
> source("./effortprocessing/time-block-gggw-n-nccs.R")
```

The effort dataset is sent to the Randomization folder as "eff.csv". Sightings are sent as "sit.csv".

6 Dataset summaries

6.1 Summary tables

As a final step of data exploration, I summarized each platform's datasets to get effort and sightings for each month in each year.

```
> read.csv("./summaries/GGEFF.csv")
```

	jan	feb	mar	apr	may	jun	jul
1	152.79186	137.28835	260.87018	68.54015	87.3412	NA	157.75213
2	NA	NA	NA	66.93207	338.2228	205.1683	96.51056
3	NA	92.34324	78.99687	123.87401	NA	87.6419	85.84184
4	NA	NA	NA	234.79139	301.5380	286.2967	109.25110
5	98.58730	97.60326	183.67721	NA	294.9939	267.8296	297.45539
6	NA	NA	NA	127.50266	229.7437	NA	NA
7	NA	NA	NA	NA	NA	104.5961	NA
8	156.66120	136.54866	367.58387	NA	181.0465	249.1895	297.36068
9	NA	NA	NA	92.88131	480.7049	387.0473	508.29471
10	74.88167	166.13006	67.33715	207.10329	123.1943	175.7076	115.95184

	aug	sep	oct	nov	dec
1	35.42301	204.33078	119.47572	212.75059	NA
2	90.21588	91.21409	85.41996	85.92574	94.41888
3	125.15043	52.37885	NA	222.45858	NA
4	146.40727	209.13292	81.51740	163.47742	93.82772
5	237.25814	274.42338	92.24673	106.97487	NA
6	NA	NA	NA	93.29250	97.98659
7	436.27833	100.17194	133.89832	NA	135.93077
8	261.81398	99.49391	254.01698	57.98203	55.36880
9	244.68383	253.37531	374.68167	NA	44.86272
10	NA	NA	199.09619	226.26214	NA

```
> read.csv("./summaries/GGSIT.csv")
```

	jan	feb	mar	apr	may	jun	jul	aug	sep	oct	nov	dec
1	0	0	0	1	3	0	5	3	29	40	18	0
2	0	0	0	0	2	8	2	15	10	52	18	1
3	0	0	0	0	0	6	2	2	3	0	0	0
4	0	0	0	1	0	21	27	7	38	44	42	28
5	0	0	0	0	16	23	37	42	33	13	14	0
6	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	18	8	0	0	1
8	8	0	0	0	5	22	0	28	34	76	0	0
9	0	0	0	0	14	15	87	38	95	98	0	0
10	9	0	0	2	0	3	1	0	0	73	43	0

```
> read.csv("./summaries/NCEFF.csv")
```

	jan	feb	mar	apr	may	jun	jul	aug	sep	oct
1	NA	NA	NA	NA	NA	77.28826	212.29446	266.9994	345.5021	234.35485
2	NA	NA	NA	NA	NA	71.49463	238.61114	238.9816	262.0202	NA
3	NA	NA	NA	NA	NA	41.13418	151.67275	286.0436	510.4157	365.78853
4	NA	NA	NA	NA	NA	53.03536	201.09021	156.6545	214.5746	40.54116
5	NA	NA	NA	NA	NA	21.80330	187.33928	378.3397	240.8327	NA
6	NA	NA	NA	NA	NA	NA	69.92881	261.3773	218.4841	284.55140
7	NA	NA	NA	NA	9.231399	218.12253	97.22002	522.0730	424.8517	178.78377

```

8 NA NA NA NA NA 46.93809 202.33223 336.5371 435.9064 484.05472
9 NA NA NA NA 49.843181 195.90455 199.35302 512.1246 269.5746 217.90693
      nov dec
1      NA NA
2      NA NA
3      NA NA
4 6.833462 NA
5      NA NA
6      NA NA
7      NA NA
8      NA NA
9      NA NA

```

```
> read.csv("./summaries/NCSIT.csv")
```

```

      jan feb mar apr may jun jul aug sep oct nov dec
1    0    0    0    0    0  10  11  84  55  59    0    0
2    0    0    0    0    0    8  34  53  37    0    0    0
3    0    0    0    0    0  23  54 125 232 160    0    0
4    0    0    0    0    0  20  74  86  94  52    9    0
5    0    0    0    0    0    3  31 152 113    0    0    0
6    0    0    0    0    0    0 145 326 282 164    0    0
7    0    0    0    0    9  42  24 110  95  44    0    0
8    0    0    0    0    0  21  73  83 207 332    0    0
9    0    0    0    0    4 107  33 221 157  49    0    0

```

6.2 Effort trends

Effort trends year-to-year and month-to-month were explored using these files.

```

> # R file that made the below .csv's:
> # source("./summaries/density-trends-04-14.R")
>
> # Monthly Trends
> read.csv("./summaries/monthly-effort.csv",stringsAsFactors=FALSE,header=TRUE)

```

	month	gggw	nccs
1	J	482.9220	0.000000
2	F	629.9136	0.000000
3	M	958.4653	0.000000
4	A	921.6249	0.000000
5	M	2036.7853	59.074580
6	J	1763.4771	725.720916
7	J	1668.4183	1559.841915
8	A	1577.2309	2959.130765
9	S	1284.5212	2922.162172
10	O	1340.3530	1805.981359
11	N	1169.1239	6.833462
12	D	522.3955	0.000000

```

> # Annual Trends
> read.csv("./summaries/trends-junnov-all.csv",stringsAsFactors=FALSE,header=TRUE)

```

	yr	platform	effort	sits	density	color	pch
1	2005	gggw	817.0734	98	0.11994026	firebrick	16
2	2006	gggw	992.6774	107	0.10778930	firebrick	16

3	2007	gggw	573.4716	13	0.02266895	firebrick	16
4	2008	gggw	1297.6208	179	0.13794476	firebrick	16
5	2009	gggw	1571.1820	178	0.11329050	firebrick	16
6	2010	gggw	323.0362	0	0.00000000	firebrick	16
7	2011	gggw	774.9447	26	0.03355078	firebrick	16
8	2012	gggw	1400.9036	165	0.11778112	firebrick	16
9	2013	gggw	2248.7878	347	0.15430536	firebrick	16
10	2014	gggw	840.2120	120	0.14282109	firebrick	16
11	2005	nccs	0.0000	0	NA	dark blue	1
12	2006	nccs	1136.4391	219	0.19270720	dark blue	1
13	2007	nccs	811.1075	132	0.16274044	dark blue	1
14	2008	nccs	1355.0548	594	0.43835865	dark blue	1
15	2009	nccs	672.7293	335	0.49797144	dark blue	1
16	2010	nccs	828.3150	299	0.36097378	dark blue	1
17	2011	nccs	834.3416	917	1.09907024	dark blue	1
18	2012	nccs	1450.2824	324	0.22340476	dark blue	1
19	2013	nccs	1505.7685	716	0.47550469	dark blue	1
20	2014	nccs	1444.7068	571	0.39523589	dark blue	1

7 Useful Functions

7.1 Julian Day

Pools a dataframe of blocked surveys into monthly bins of effort/sightings.

```
> source("../effortprocessing/julian.R")
> julian.split

function (gg, n, platform, pool.funk)
{
  gg$X <- NULL
  breaks <- floor(seq(0, 365, length = n + 1))
  platform <- rep(platform, times = n)
  blockmeandf <- data.frame()
  for (i in 2:length(breaks)) {
    ggsub <- subset(gg, gg$julian > breaks[i - 1] & gg$julian <
      breaks[i])
    ggsub$julian <- NULL
    blockmeans <- rep(NA, times = 54)
    if (nrow(ggsub) != 0) {
      for (j in 1:54) {
        if (pool.funk == "mean") {
          blockmeans[j] <- mean(as.numeric(ggsub[, j]),
            na.rm = TRUE)
        }
        if (pool.funk == "sum") {
          blockmeans[j] <- sum(as.numeric(ggsub[, j]),
            na.rm = TRUE)
        }
      }
    }
  }
  blockmeandf <- rbind(blockmeandf, blockmeans)
}
```

```

    final <- blockmeandf
    return(final)
}
> jul.pool
function (gg, nc, bg, n, pool.funk = "mean")
{
  ggief.mn <- julian.split(gg, n = n, platform = "GGW", pool.funk = pool.funk)
  ncief.mn <- julian.split(nc, n = n, platform = "NCCS", pool.funk = pool.funk)
  bgief.mn <- julian.split(bg, n = n, platform = "BANG", pool.funk = pool.funk)
  combdf <- rbind(gg, nc, bg)
  combief.mn <- julian.split(combdf, n = n, platform = "COMB",
    pool.funk = pool.funk)
  return(list(ggief.mn, ncief.mn, bgief.mn, combief.mn))
}
> setwd(curdir)

```