Strava History Analysis

Author: CARLOS M N

DNI: XXXXXXXX-X

Clear workspace

```
clear all; clc; close all;
```

General variables

```
n_bins = 25;
n_cvpartitions = 10;
def_colors =[ [0, 0.4470, 0.7410];
              [0.8500, 0.3250, 0.0980];
              [0.9290, 0.6940, 0.1250];
              [0.4940, 0.1840, 0.5560];
              [0.4660, 0.6740, 0.1880];
              [0.3010, 0.7450, 0.9330];
              [0.6350, 0.0780, 0.1840]];
```

Initialize variables

Load tables

```
activities_set = readtable('../strava_history/activities.csv')
```

```
Warning: Column headers from the file were modified to make them valid MATLAB identifiers before creating
variable names for the table. The original column headers are saved in the VariableDescriptions property.
Set 'VariableNamingRule' to 'preserve' to use the original column headers as table variable names.
activities_set = 517×84 table
```

...

	ActivityID	ActivityDate	ActivityName
1	737142108	'Oct 7, 2016, 8:56:47 AM'	'Pedalada de mañana'
2	772348809	'Nov 11, 2016, 10:35:51 AM'	'Vuelta en bici a la hora del almuerzo'
3	791388617	'Dec 2, 2016, 10:53:06 AM'	'Vuelta en bici a la hora del almuerzo'
4	795145644	'Dec 6, 2016, 11:48:08 AM'	'Vuelta en bici a la hora del almuerzo'
5	795982182	'Dec 7, 2016, 12:26:46 PM'	'Pedalada de tarde'
6	797672686	'Dec 9, 2016, 12:12:21 PM'	'Pedalada de tarde'
7	1.0366e+09	'Jun 14, 2017, 5:28:42 PM'	'Ciclismo al anochecer'
8	1.0375e+09	'Jun 15, 2017, 6:44:36 AM'	'Ciclismo por la mañana'
9	1.0390e+09	'Jun 16, 2017, 7:09:22 AM'	'Ciclismo por la mañana'
10	1.0611e+09	'Jun 30, 2017, 3:02:40 PM'	'Atletismo por la tarde'
11	1.1021e+09	'Jul 26, 2017, 6:55:16 AM'	'Ciclismo por la mañana'
12	1.1054e+09	'Jul 27, 2017, 7:50:58 AM'	'Ciclismo por la mañana'

	ActivityID	ActivityDate	ActivityName
13	1.1076e+09	'Jul 29, 2017, 5:32:22 PM'	'Ciclismo al anochecer'
14	1.1126e+09	'Aug 1, 2017, 4:09:30 PM'	'Ciclismo al anochecer'

⋮

Transform input data

Ensure that ID is integer (this variable will only be used for mapping to GPX files, not as a predictor)

```
activities_set.ActivityID = uint64(activities_set.ActivityID);
```

Create categorical variables

```
activities_set.ActivityType = categorical(activities_set.ActivityType);
activities_set.ActivityGear = categorical(activities_set.ActivityGear);
activities_set.ActivityGear = fillmissing(activities_set.ActivityGear, "constant", "No Data");
```

Divide Date into day of the year / year

```
aux = split(activities_set.ActivityDate,',');

activities_set.DayofYear = day(datetime(datenum(aux(:,1),'mmmdd'), ConvertFrom='datenum'), 'day
activities_set.Year = uint16(str2num(cell2mat(aux(:,2))));
```

Get paused time

```
activities_set.PausedTimeRatio = activities_set.ElapsedTime ./ activities_set.MovingTime - 1;
```

Get net elevation (difference between starting point and destination)

```
activities_set.ElevationNet = activities_set.ElevationGain - activities_set.ElevationLoss;
```

Compute total weight

```
def_AthleteWeight = 70;
def_BikeWeight = 13;
def_LuggageWeight = 7;

activities_set.TotalWeight = fillmissing(activities_set.AthleteWeight, "constant", def_AthleteW
% Add bike weight
idx = activities_set.ActivityType == "Ride";
activities_set.TotalWeight(idx) = activities_set.TotalWeight(idx) + fillmissing(activities_set.
% Add luggage weight when doing bikepacking
idx = contains(activities_set.ActivityDescription, "Bikepacking");
activities_set.TotalWeight(idx) = activities_set.TotalWeight(idx) + def_LuggageWeight;
```

Convert Speed from [m/s] to [km/h]

```
activities_set.MaxSpeed = 3.6*activities_set.MaxSpeed;
activities_set.AverageSpeed = 3.6*activities_set.AverageSpeed;
```

Get average ascent meters per kilometer

```
activities_set.ElevationGainRatio = activities_set.ElevationGain ./ activities_set.Distance;
```

Compute mean speed from distance and moving time

```
activities_set.MeanSpeed = 3600*activities_set.Distance ./ activities_set.MovingTime;
```

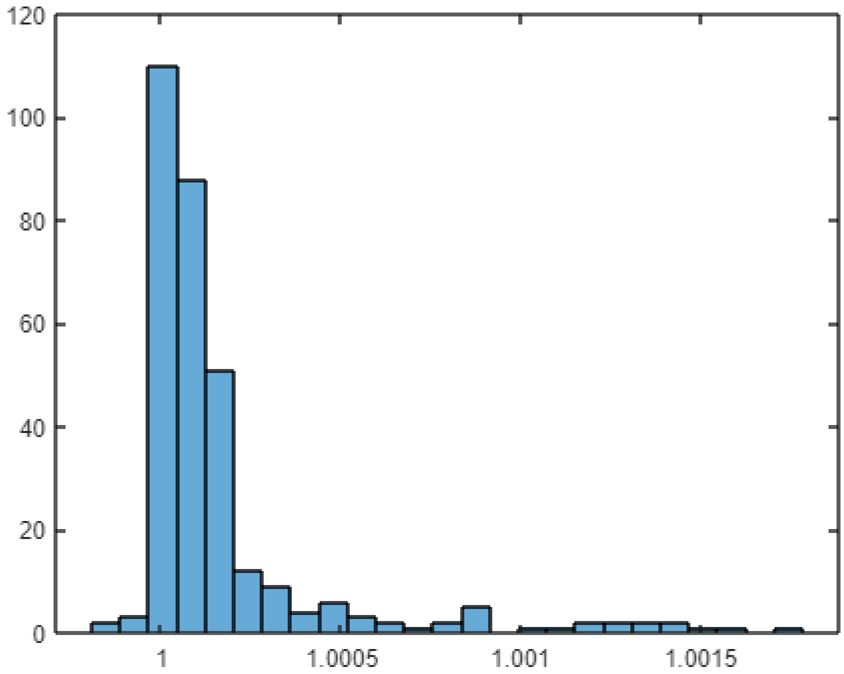
Check correctness

Check computed MeanSpeed is similar to AvgSpeed provided

```
max_test = max(activities_set.AverageSpeed ./ activities_set.MeanSpeed);
min_test = min(activities_set.AverageSpeed ./ activities_set.MeanSpeed);
fprintf(strcat("Max/Min ratio of mean speed: ",num2str(max_test)," / ",num2str(min_test)))
```

```
Max/Min ratio of mean speed: 1.0018 / 0.99983
```

```
histogram(activities_set.AverageSpeed ./ activities_set.MeanSpeed, n_bins)
```

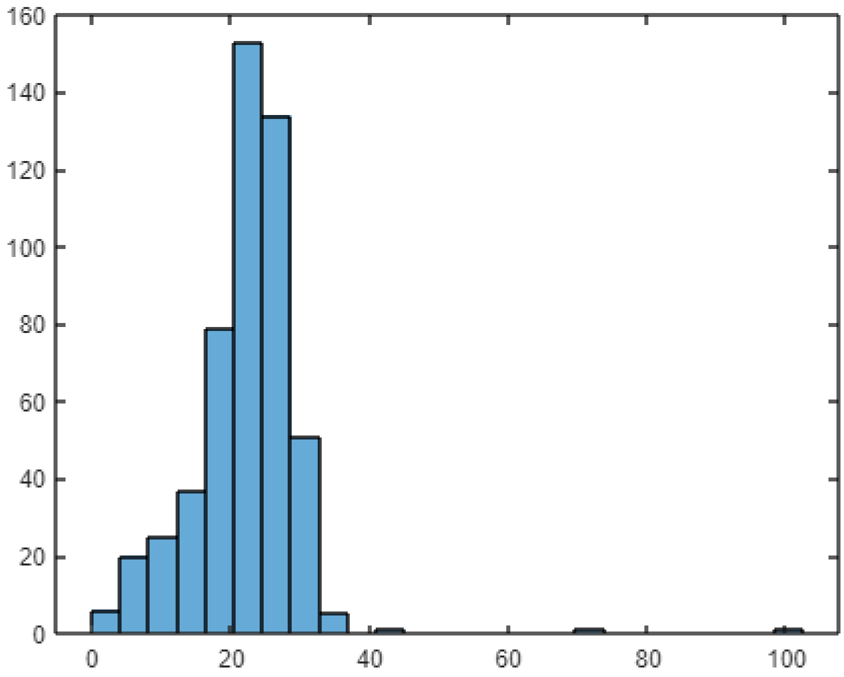


Check reasonable MeanSpeed

```
max_test = max(activities_set.MeanSpeed);
min_test = min(activities_set.MeanSpeed);
fprintf(strcat("Max/Min mean speed: ",num2str(max_test)," / ",num2str(min_test)))
```

```
Max/Min mean speed: 101.4647 / 0.69058
```

```
histogram(activities_set.MeanSpeed, n_bins)
```

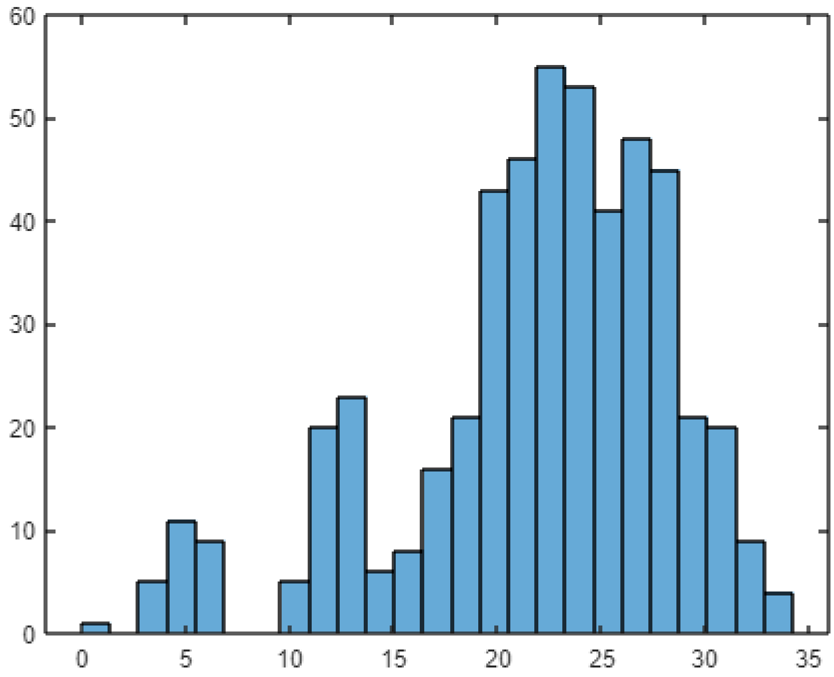


Remove Outliers

```
activities_set(activities_set.MeanSpeed > 40, :) = [];

max_test = max(activities_set.MeanSpeed);
min_test = min(activities_set.MeanSpeed);
fprintf(strcat("Max/Min mean speed: ",num2str(max_test)," / ",num2str(min_test)))
```

```
Max/Min mean speed: 34.1903 / 0.69058
```

```
histogram(activities_set.MeanSpeed, n_bins)
```

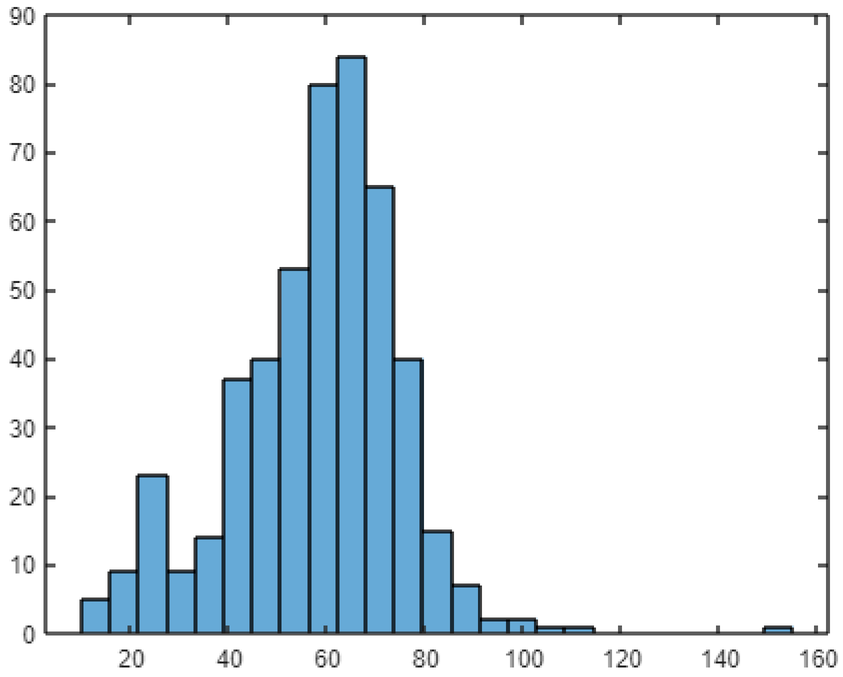


Check reasonable MaxSpeed

```
max_test = max(activities_set.MaxSpeed);
min_test = min(activities_set.MaxSpeed);
fprintf(strcat("Max/Min max speed: ",num2str(max_test)," / ",num2str(min_test)))
```

```
Max/Min max speed: 153 / 11.88
```

```
histogram(activities_set.MaxSpeed, n_bins)
```

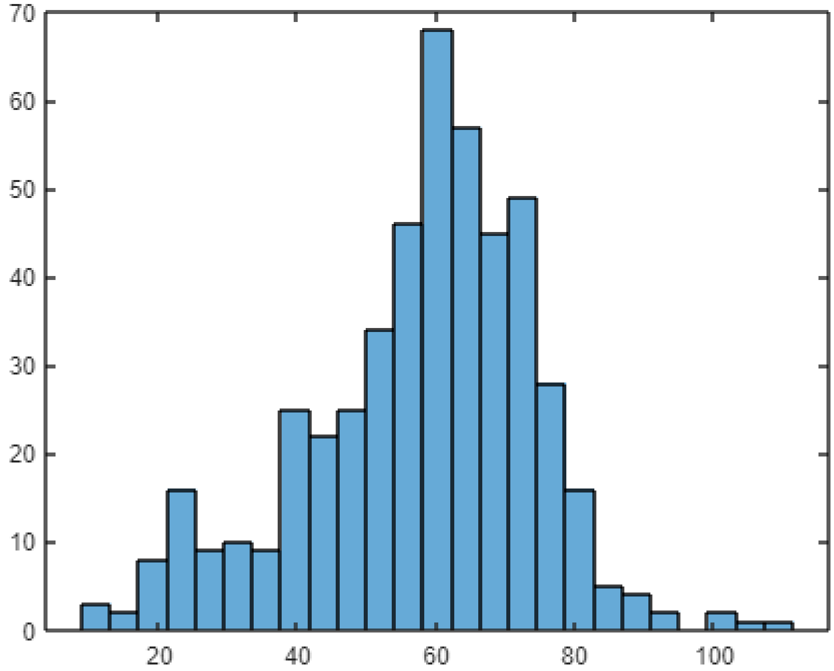


Remove outliers

```
activities_set(activities_set.MaxSpeed > 120, :) = [];

max_test = max(activities_set.MaxSpeed);
min_test = min(activities_set.MaxSpeed);
fprintf(strcat("Max/Min max speed: ",num2str(max_test)," / ",num2str(min_test)))
```

```
Max/Min max speed: 110.52 / 11.88
```

```
histogram(activities_set.MaxSpeed, n_bins)
```

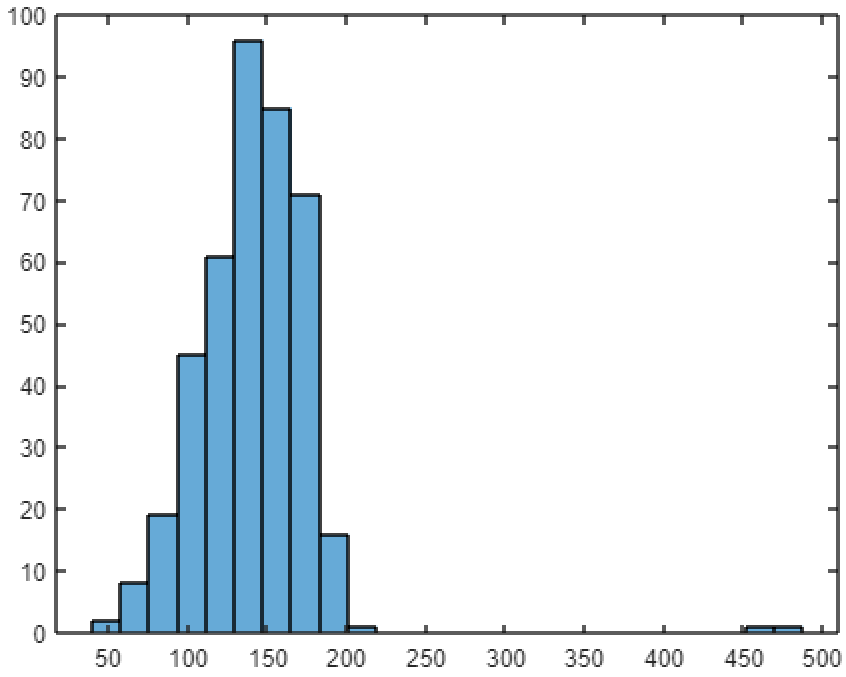


Check reasonable AverageWatts

```
max_test = max(activities_set.AverageWatts);
min_test = min(activities_set.AverageWatts);
fprintf(strcat("Max/Min average Power: ",num2str(max_test)," / ",num2str(min_test)))
```

```
Max/Min average Power: 485.1785 / 44.3771
```

```
histogram(activities_set.AverageWatts, n_bins)
```

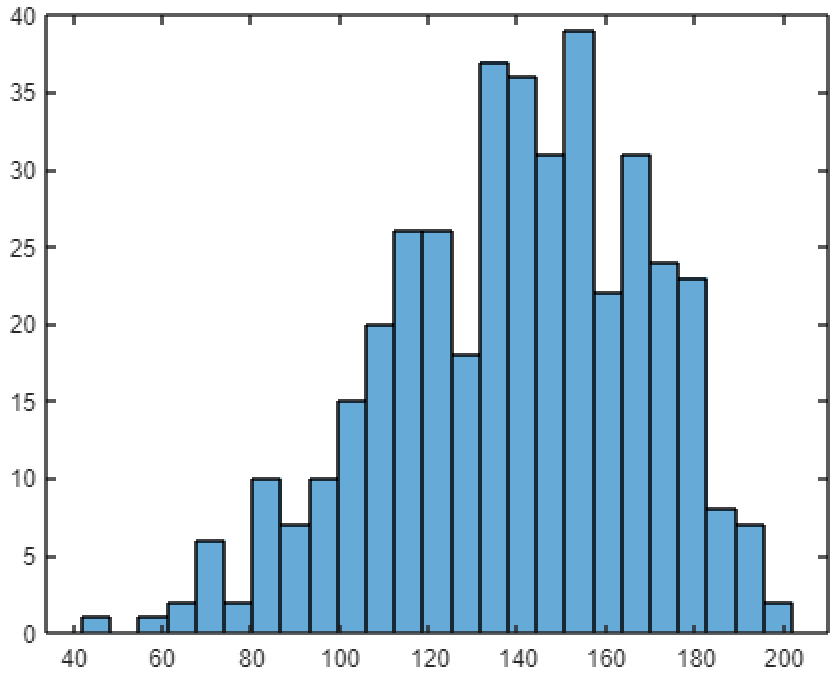


Remove outliers

```
activities_set(activities_set.AverageWatts > 300, :) = [];

max_test = max(activities_set.AverageWatts);
min_test = min(activities_set.AverageWatts);
fprintf(strcat("Max/Min average Power: ",num2str(max_test)," / ",num2str(min_test)))
```

```
Max/Min average Power: 201.5569 / 44.3771
```

```
histogram(activities_set.AverageWatts, n_bins)
```

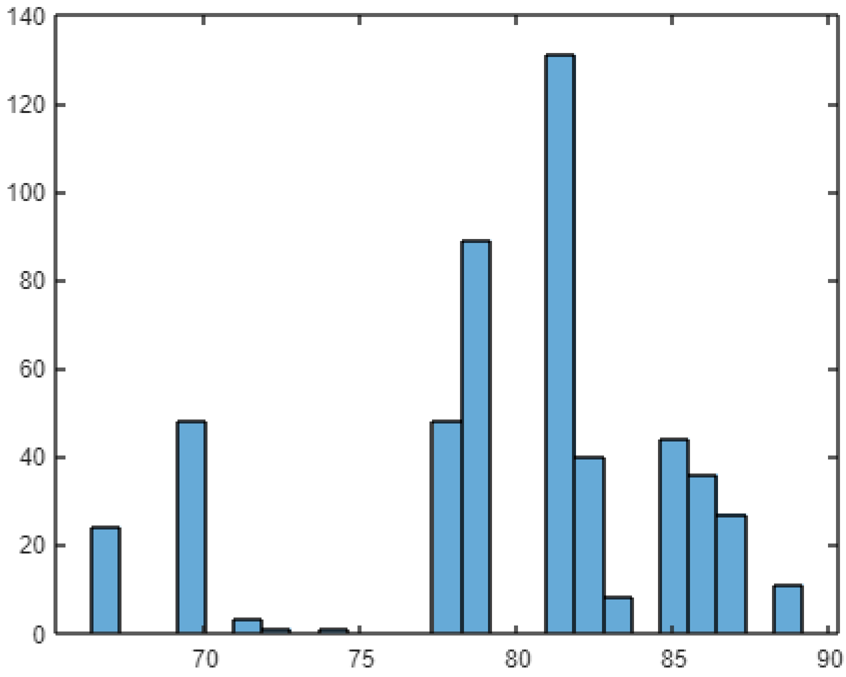


Total weight

```
max_test = max(activities_set.TotalWeight);
min_test = min(activities_set.TotalWeight);
fprintf(strcat("Max/Min total weight: ",num2str(max_test)," / ",num2str(min_test)))
```

```
Max/Min total weight: 89 / 67
```

```
histogram(activities_set.TotalWeight, n_bins)
```

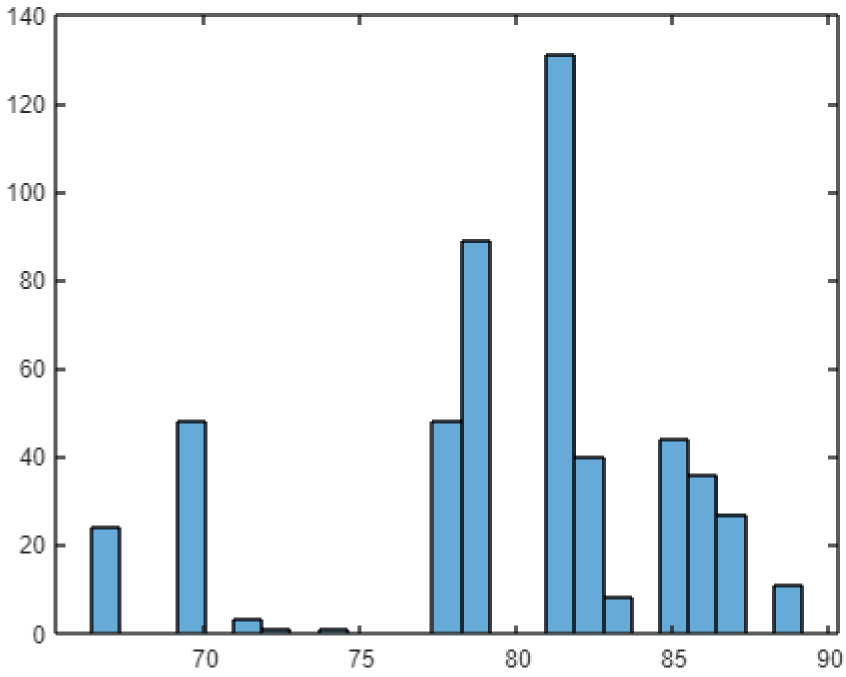


Ammend outliers (no comma in athlete Weight)

```
idx = activities_set.AthleteWeight > 200;
activities_set.TotalWeight(idx) = activities_set.TotalWeight(idx) - 0.9.*activities_set.Athlete

max_test = max(activities_set.TotalWeight);
min_test = min(activities_set.TotalWeight);
fprintf(strcat("Max/Min total weight: ",num2str(max_test)," / ",num2str(min_test)))
```

```
Max/Min total weight: 89 / 67
```

```
histogram(activities_set.TotalWeight, n_bins)
```



Keep only variables of interest

```
vars = ["ActivityID", "Year", "DayofYear", "ActivityType", "MovingTime", "PausedTimeRatio", "Di
        "MeanSpeed", "MaxSpeed", "ElevationGain", "ElevationGainRatio", "ElevationNet", "Elevat
        "TotalWeight", "AverageWatts", "Calories", "ActivityGear"];
dataClean = activities_set(:,vars)
```

```
dataClean = 511×17 table
```

...

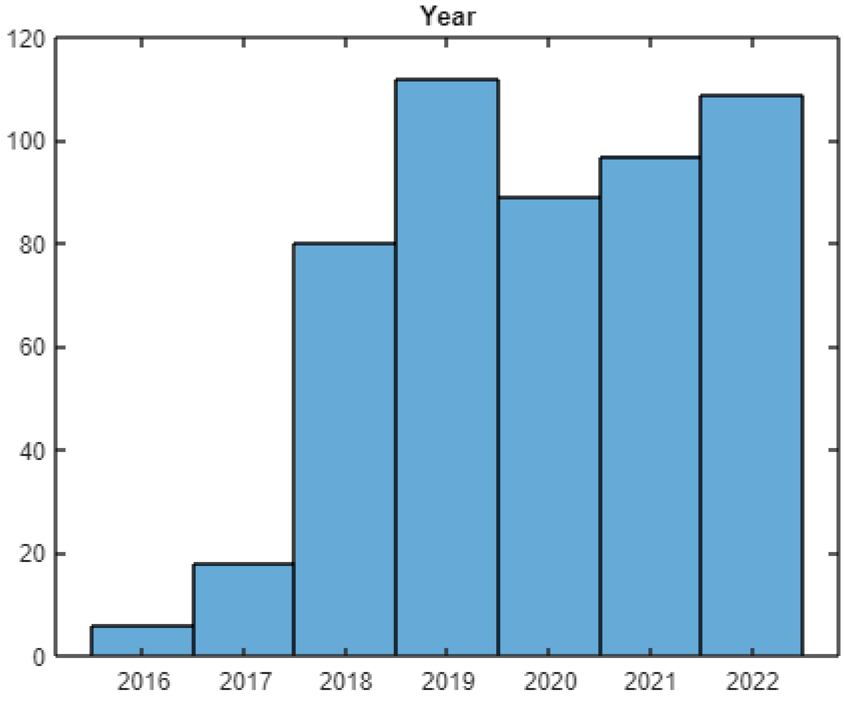
	ActivityID	Year	DayofYear	ActivityType	MovingTime
1	737142108	2016	280	Ride	8603
2	772348809	2016	315	Ride	12492
3	791388617	2016	336	Ride	7889
4	795145644	2016	340	Ride	5642
5	795982182	2016	341	Ride	3875
6	797672686	2016	343	Ride	4273
7	1036607345	2017	165	Ride	2496
8	1037484336	2017	166	Ride	8668
9	1039048647	2017	167	Ride	6462
10	1061085074	2017	181	Run	3950
11	1102095810	2017	207	Ride	9431
12	1105357168	2017	208	Ride	4500

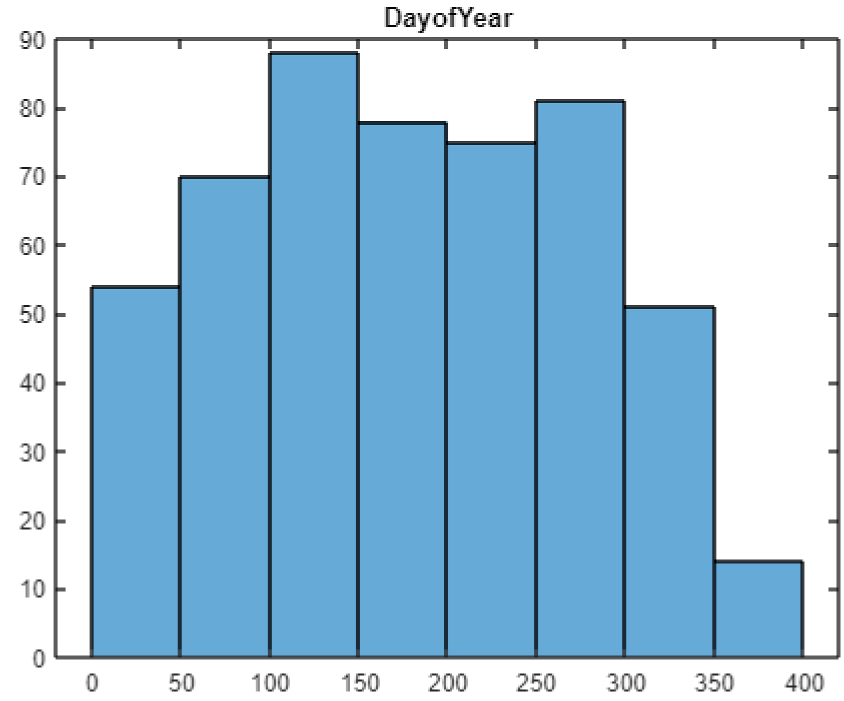
	ActivityID	Year	DayofYear	ActivityType	MovingTime
13	1107611767	2017	210	Ride	4810
14	1112599687	2017	213	Ride	8953

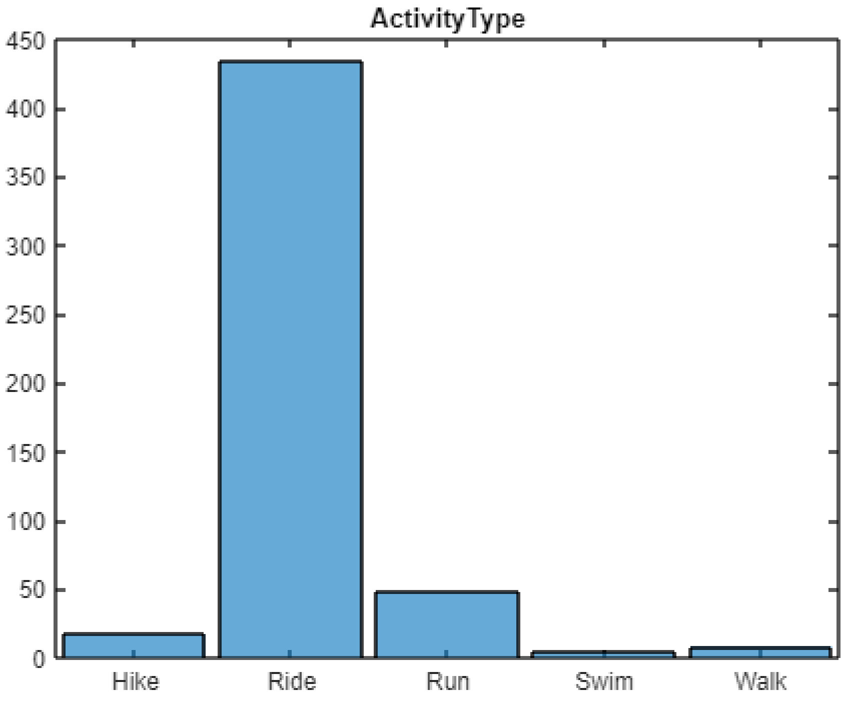
⋮

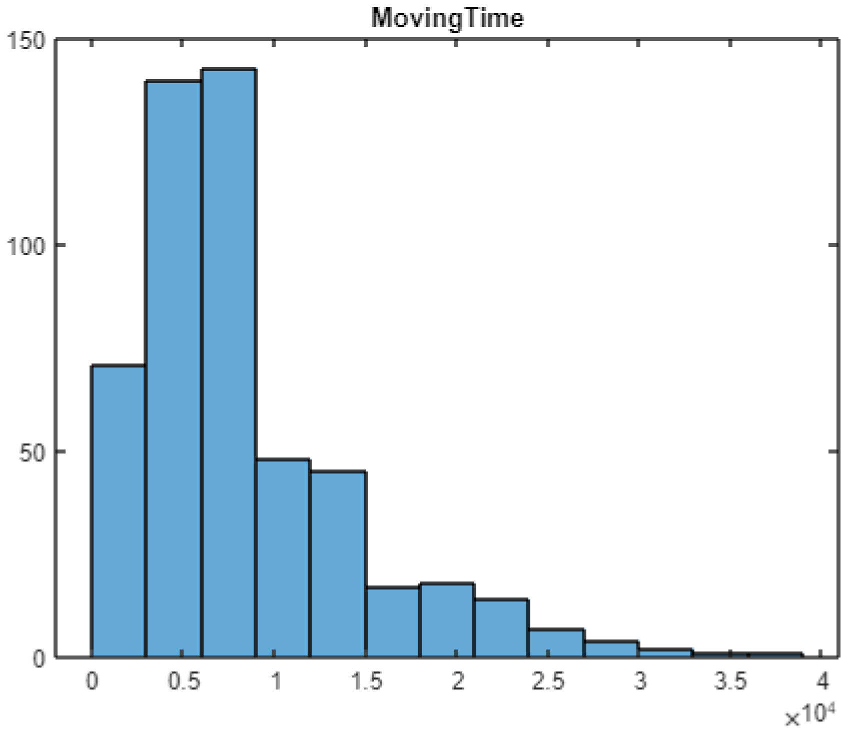
Plot histograms of general data once Cleaned

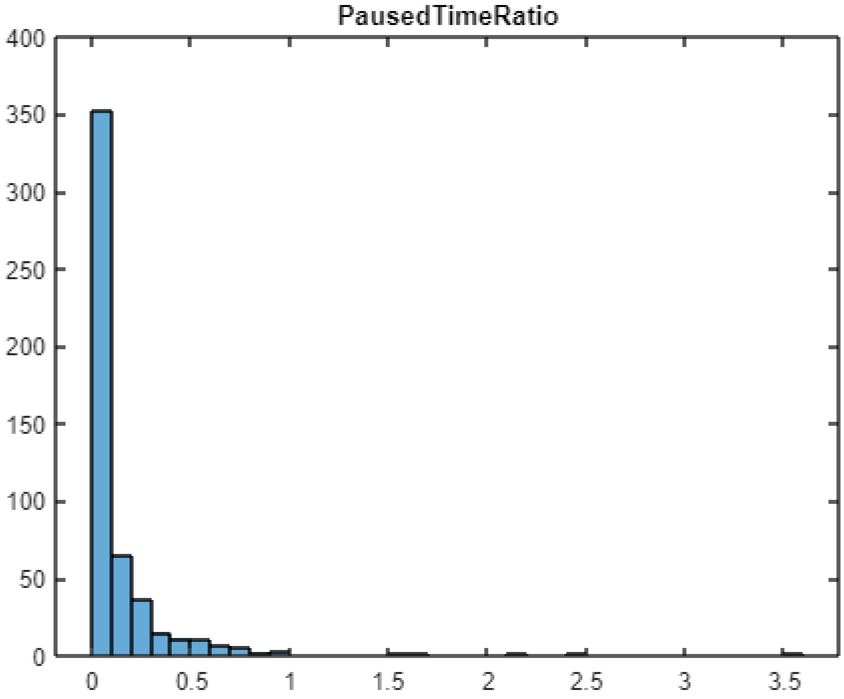
```
for ii = 2:size(vars,2)
    figure
    if iscategorical(table2array(dataClean(:,vars(ii))))
        histogram(table2array(dataClean(:,vars(ii))))
    else
        histogram(table2array(dataClean(:,vars(ii)), n_bins))
    end
    title(vars(ii))
end
```

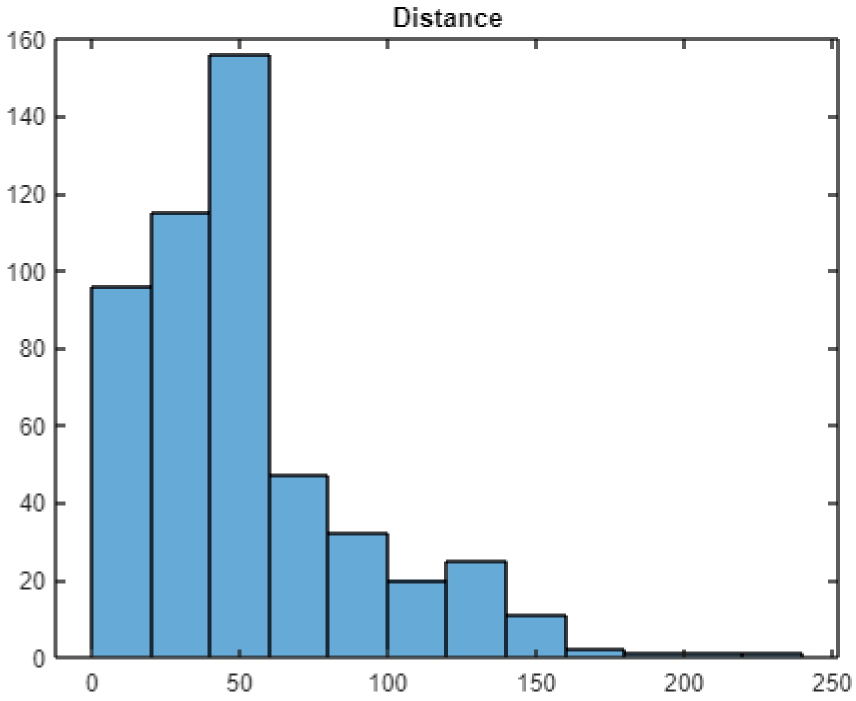


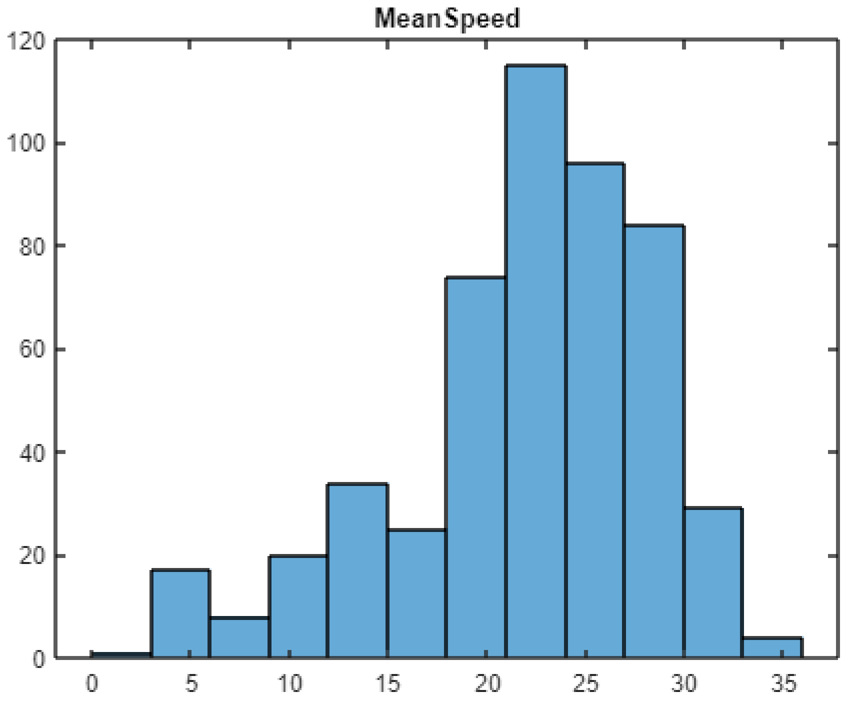

DayofYear
0
10
20
30
40
50
60
70
80
90
0
50
100
150
200
250
300
350
400

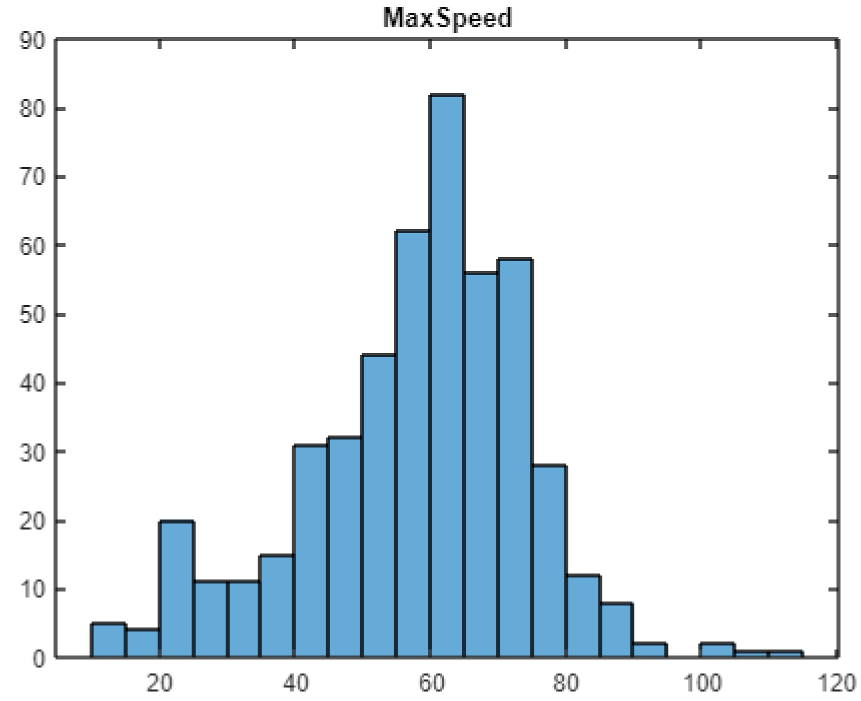

ActivityType
0
50
100
150
200
250
300
350
400
450
Hike
Ride
Run
Swim
Walk

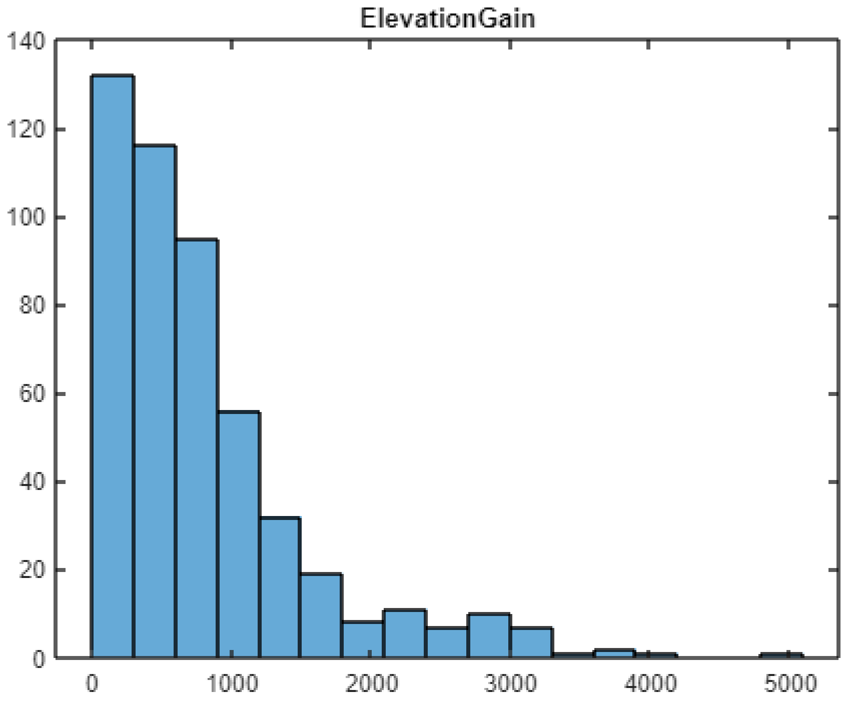

MovingTime
150
100
50
0
0
0.5
1
1.5
2
2.5
3
3.5
4
×10^4

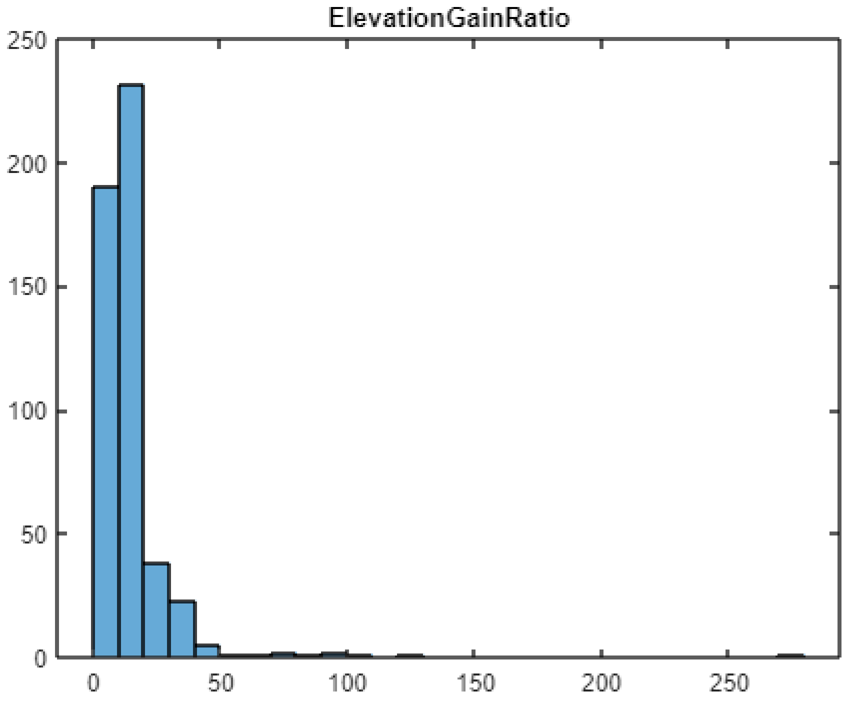

PausedTimeRatio
400
350
300
250
200
150
100
50
0
0
0.5
1
1.5
2
2.5
3
3.5

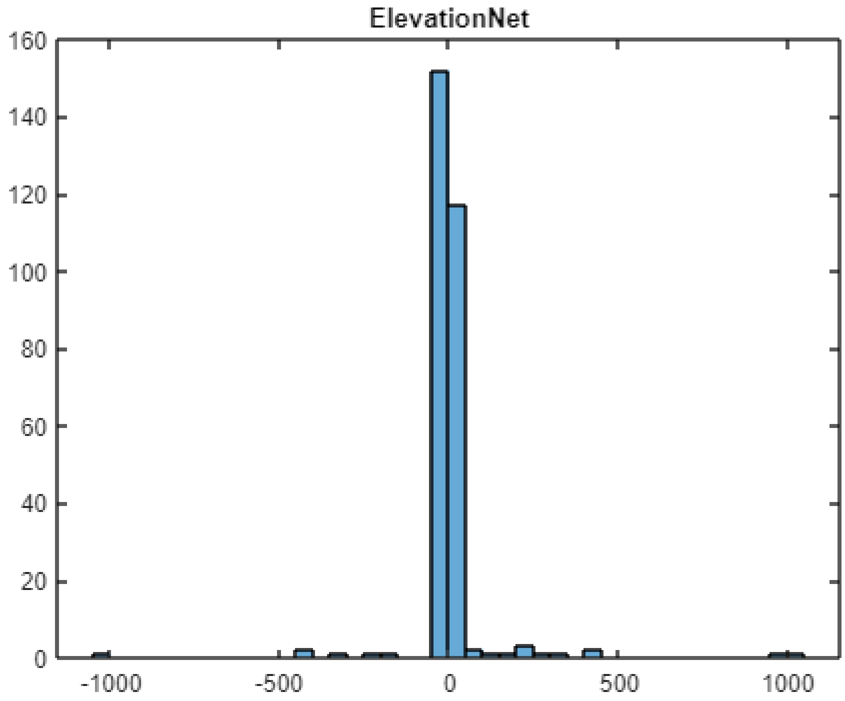

Distance
160
140
120
100
80
60
40
20
0
0
50
100
150
200
250

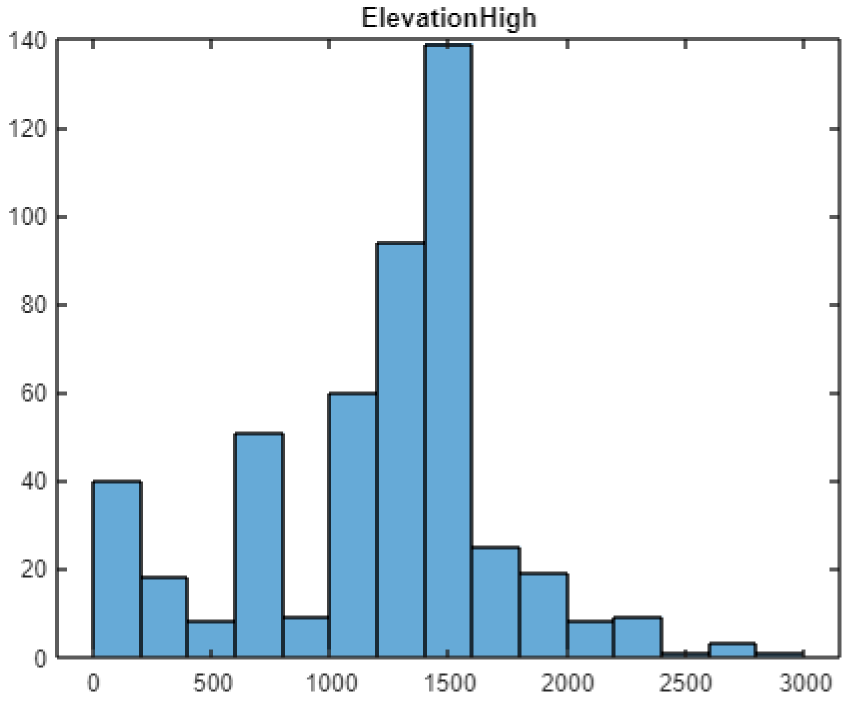

MeanSpeed
120
100
80
60
40
20
0
0
5
10
15
20
25
30
35


MaxSpeed
90
80
70
60
50
40
30
20
10
0
20
40
60
80
100
120

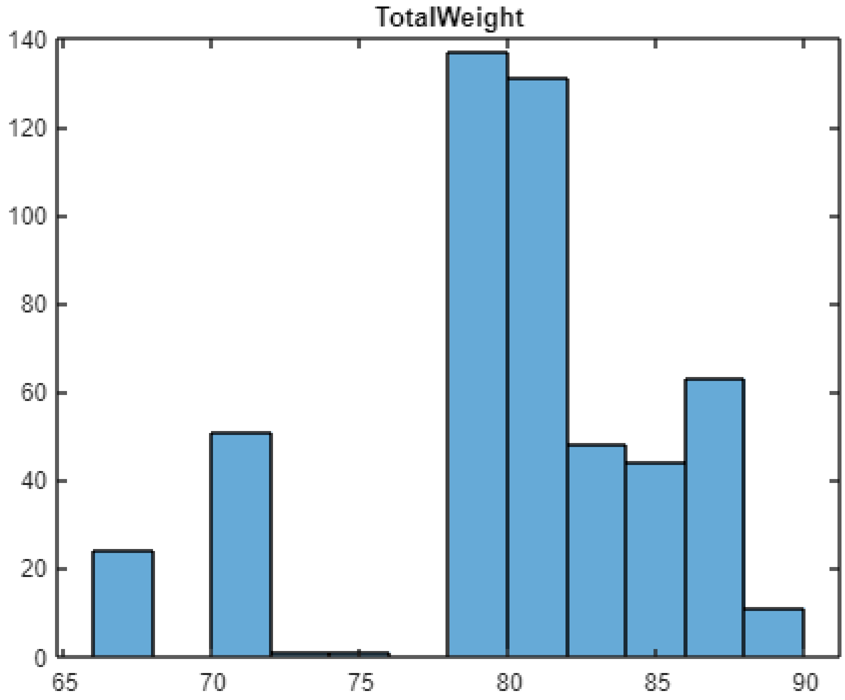

ElevationGain
140
120
100
80
60
40
20
0
0
1000
2000
3000
4000
5000


ElevationGainRatio
250
200
150
100
50
0
0
50
100
150
200
250

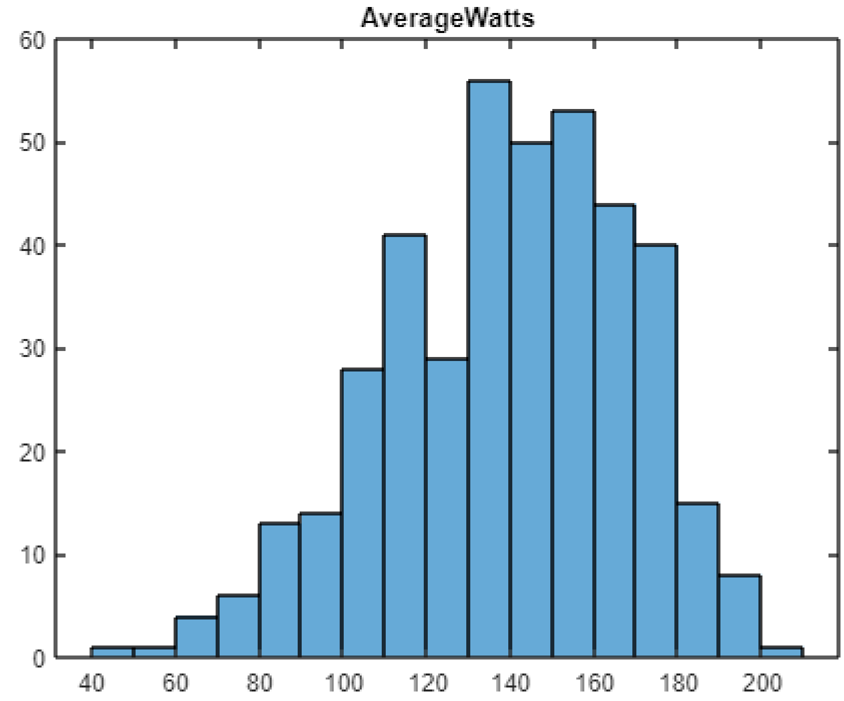

ElevationNet
160
140
120
100
80
60
40
20
0
-1000
-500
0
500
1000

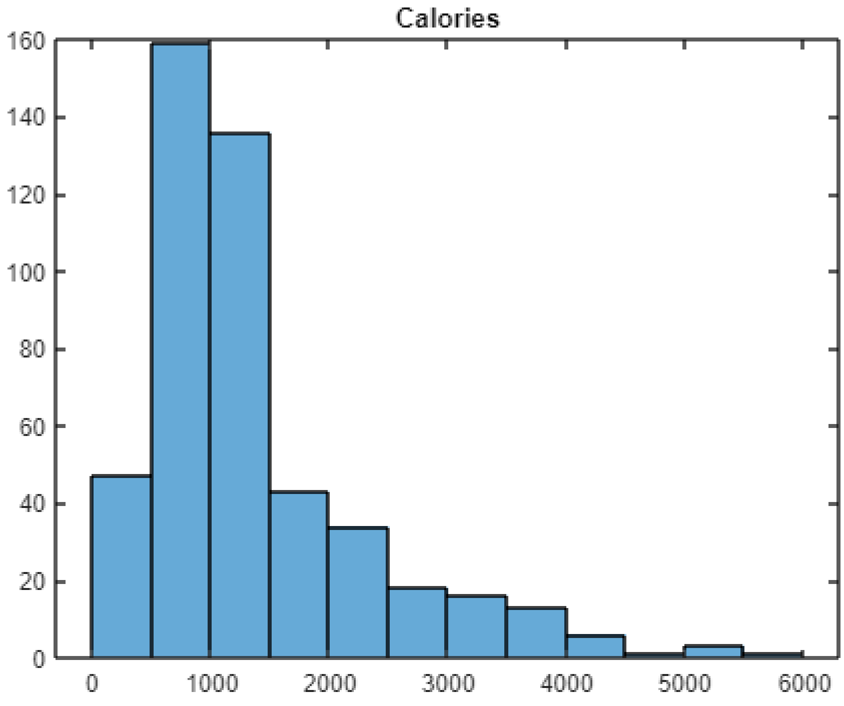


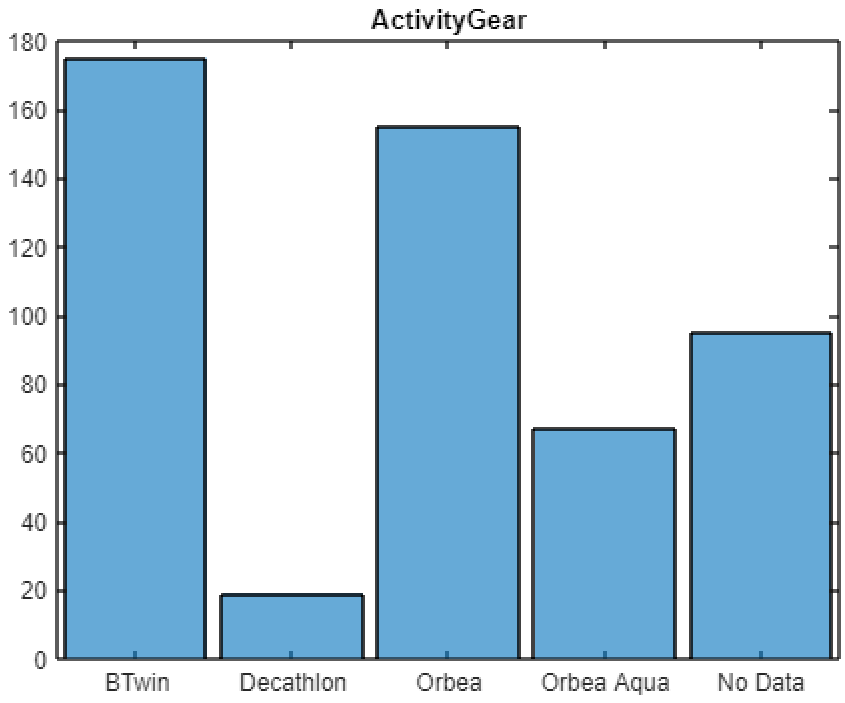
ElevationHigh



TotalWeight


AverageWatts
0
10
20
30
40
50
60
40
60
80
100
120
140
160
180
200


Calories
0
20
40
60
80
100
120
140
160
0
1000
2000
3000
4000
5000
6000



Activity Type filter: keep only bike rides

```
cyclingData = dataClean(dataClean.ActivityType == 'Ride', :);
cyclingData = removevars(cyclingData, "ActivityType")
```

```
cyclingData = 434×16 table
```

...

	ActivityID	Year	DayofYear	MovingTime	PausedTimeRatio	Distance
1	737142108	2016	280	8603	0.0076	40.8600
2	772348809	2016	315	12492	0.1258	51.7300
3	791388617	2016	336	7889	0.0843	36.7600
4	795145644	2016	340	5642	0.0482	31.1900
5	795982182	2016	341	3875	0.1899	22.1200
6	797672686	2016	343	4273	0.0012	26.6200
7	1036607345	2017	165	2496	0.5296	15.6500
8	1037484336	2017	166	8668	0.0388	47.8600
9	1039048647	2017	167	6462	0.0639	34.5900
10	1102095810	2017	207	9431	0.0349	52.1400
11	1105357168	2017	208	4500	0	25.8800
12	1107611767	2017	210	4810	0.3630	27.6600
13	1112599687	2017	213	8953	0.0545	49.3900

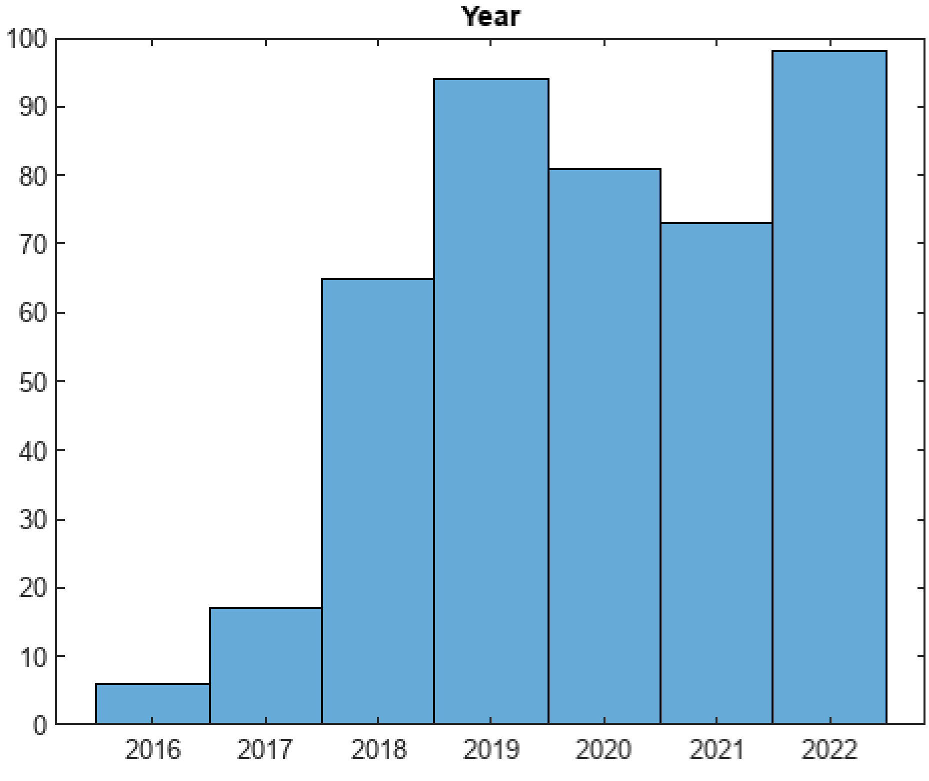
	ActivityID	Year	DayofYear	MovingTime	PausedTimeRatio	Distance
14	1114309758	2017	214	5702	0.0219	37.3100

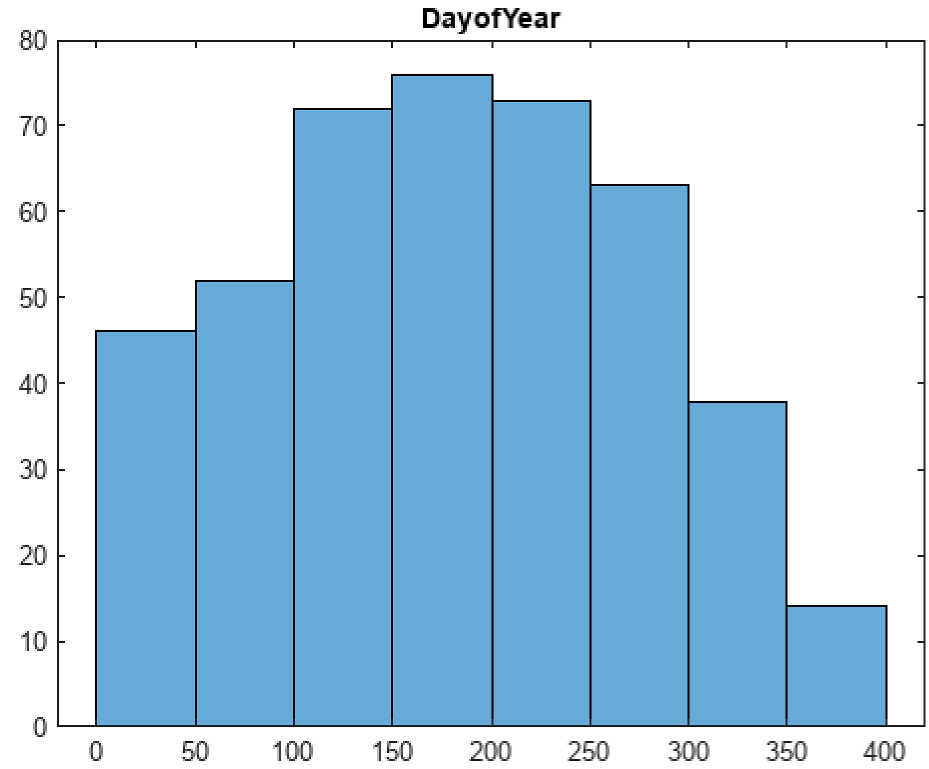
⋮

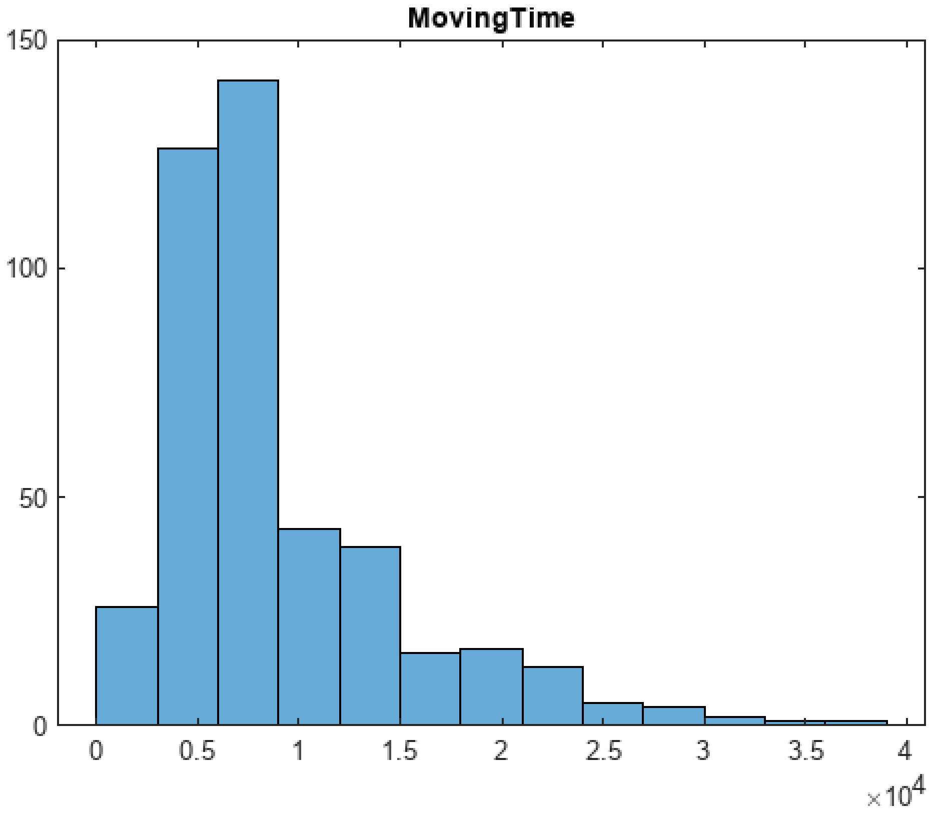
```
vars = ["ActivityID", "Year", "DayofYear", "MovingTime", "PausedTimeRatio", "Distance", ...
        "MeanSpeed", "MaxSpeed", "ElevationGain", "ElevationGainRatio", "ElevationNet", "Elevat
        "TotalWeight", "AverageWatts", "Calories", "ActivityGear"];
```

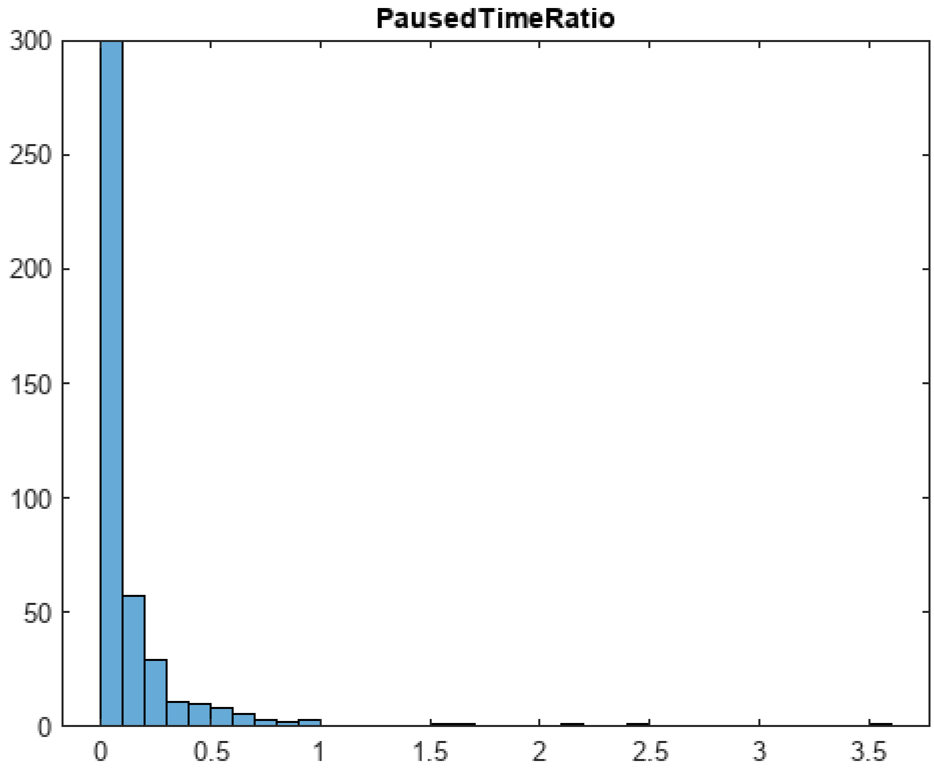
Plot histograms of Cycling activities Data

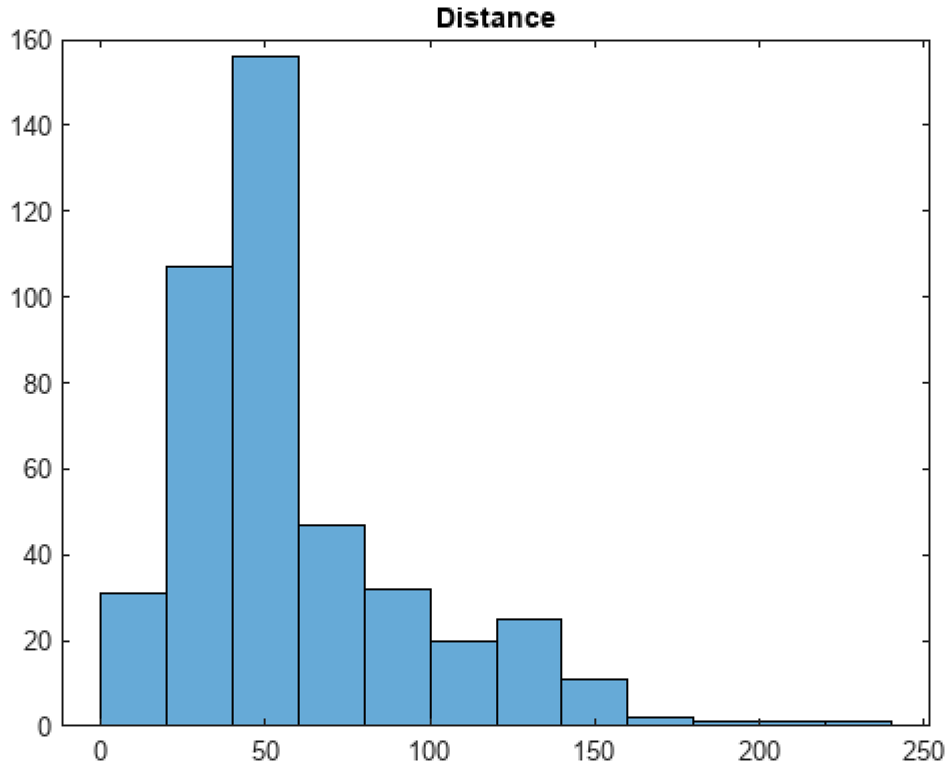
```
for ii = 2:size(vars,2)
    figure
    if iscategorical(table2array(cyclingData(:,vars(ii))))
        histogram(table2array(cyclingData(:,vars(ii))))
    else
        histogram(table2array(cyclingData(:,vars(ii)), n_bins))
    end
    title(vars(ii))
    saveas(gcf,strcat('Figures/hist_',vars(ii),'.png') )
end
```

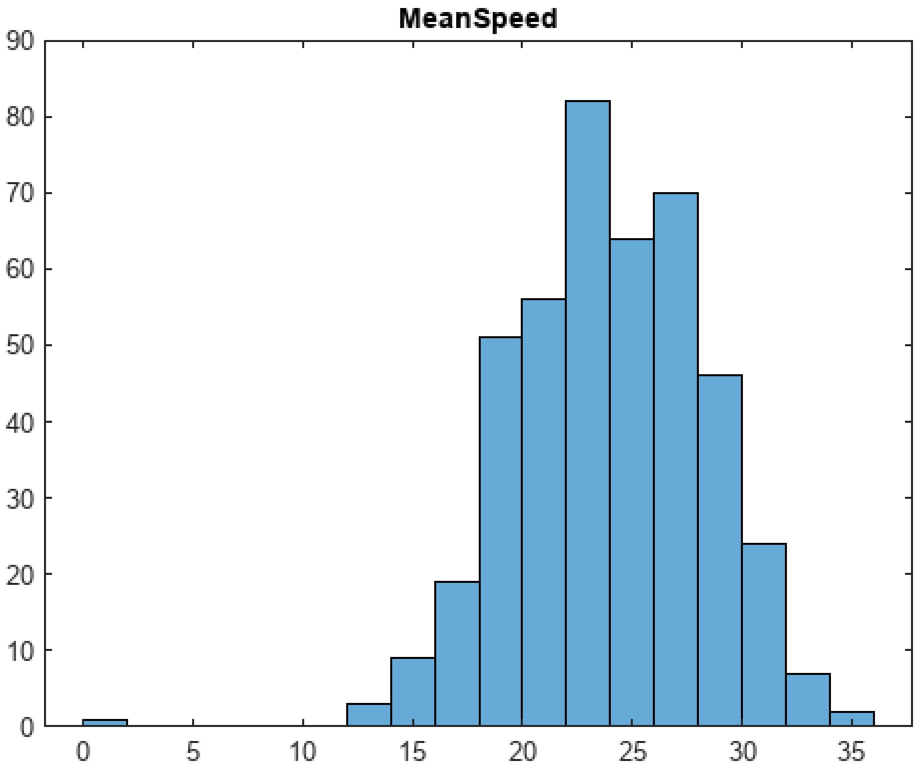


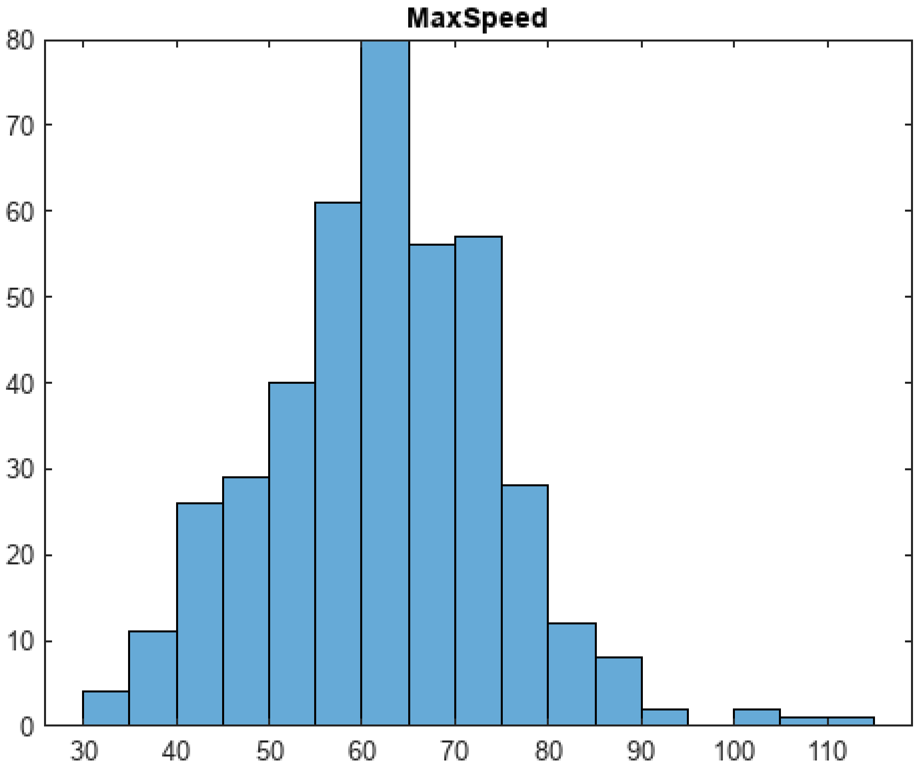

DayofYear
0
10
20
30
40
50
60
70
80
0
50
100
150
200
250
300
350
400

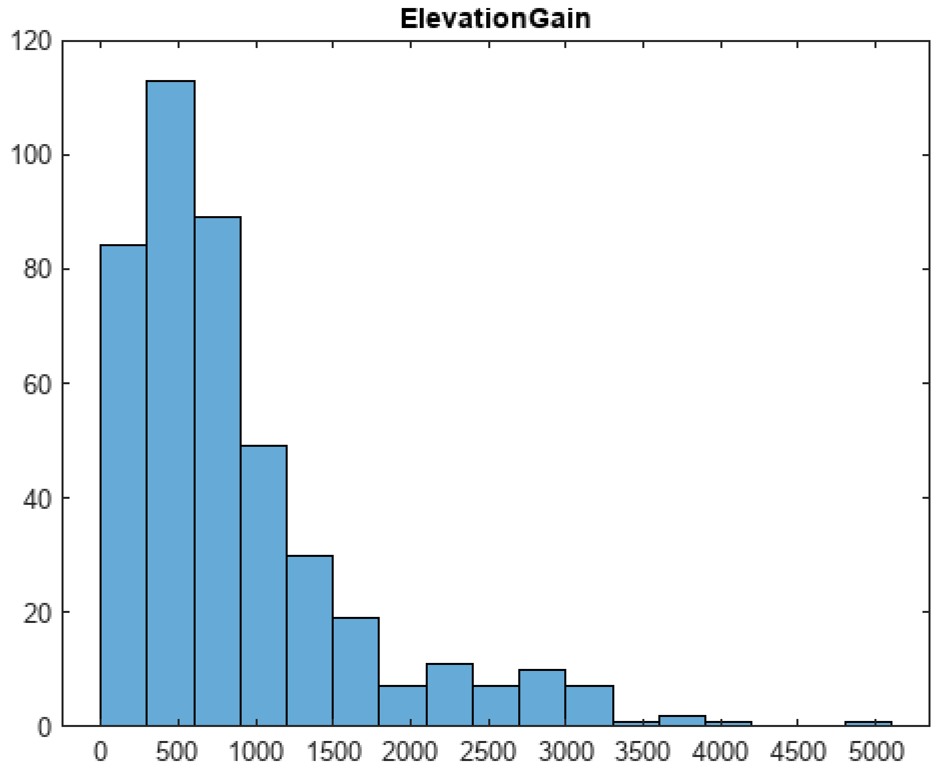

MovingTime
0
50
100
150
0
0.5
1
1.5
2
2.5
3
3.5
4
$\times 10^4$

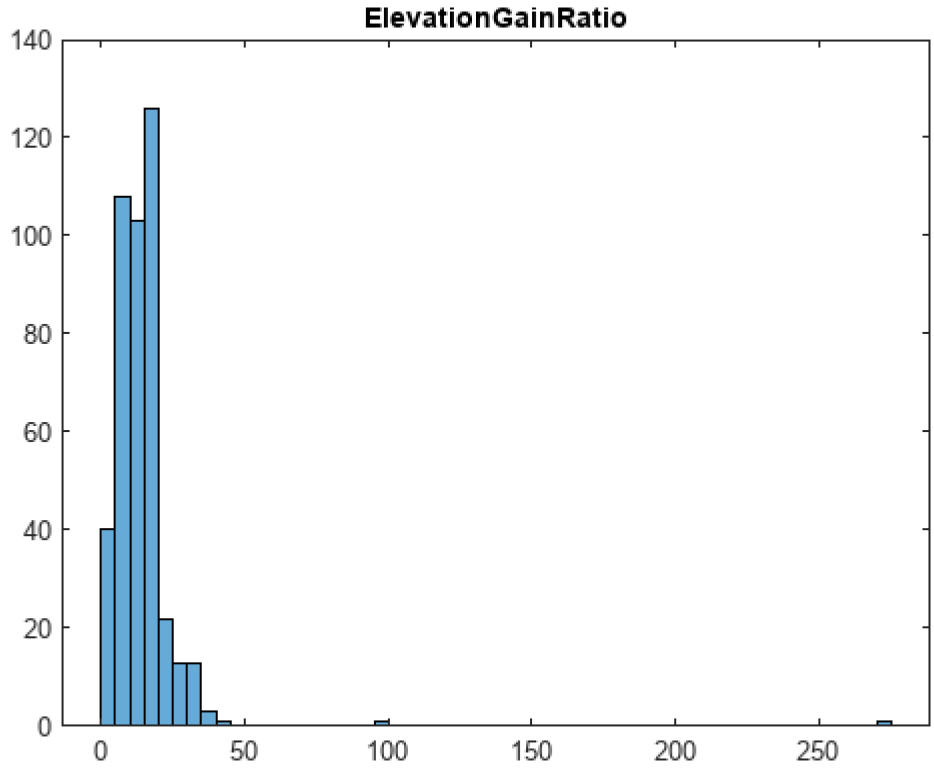

PausedTimeRatio
300
250
200
150
100
50
0
0
0.5
1
1.5
2
2.5
3
3.5

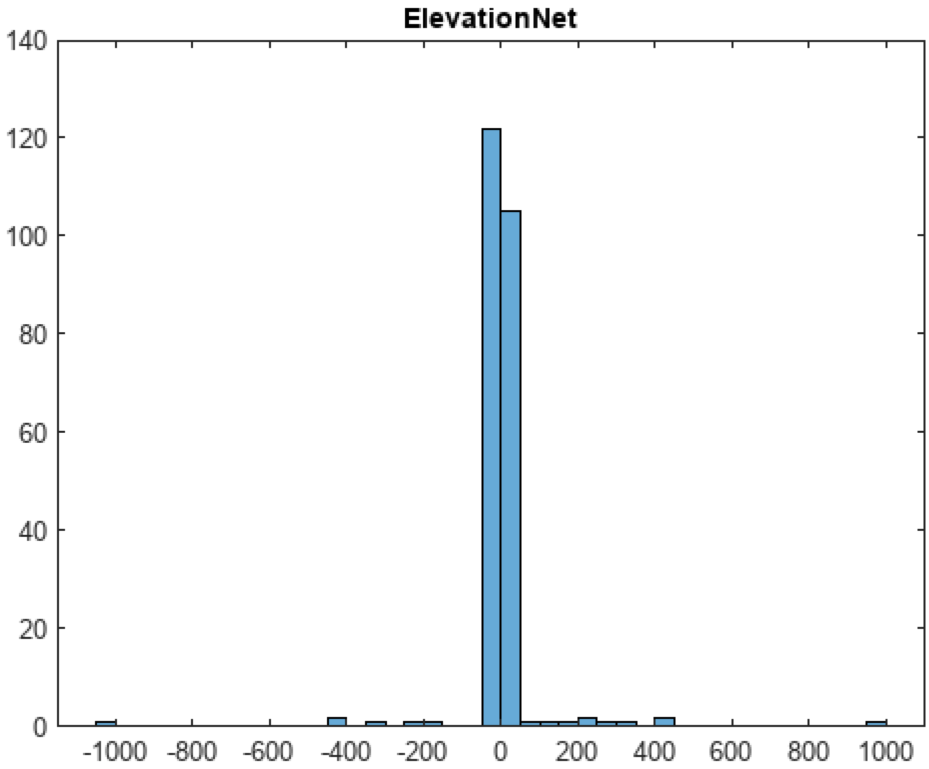

Distance
160
140
120
100
80
60
40
20
0
0
50
100
150
200
250

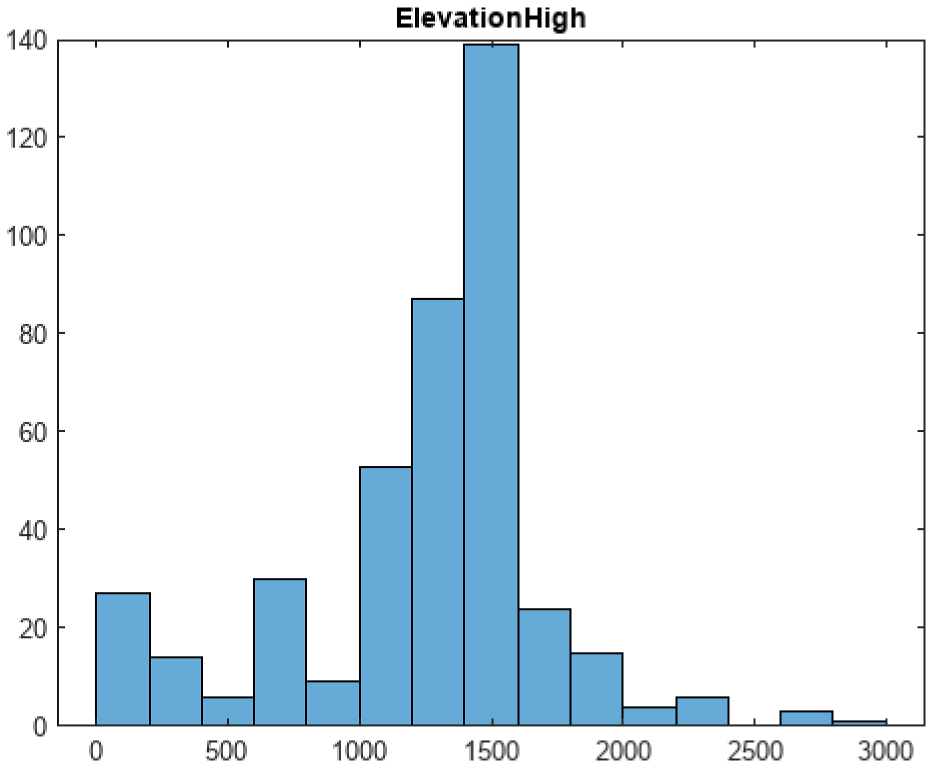

MeanSpeed
90
80
70
60
50
40
30
20
10
0
0
5
10
15
20
25
30
35

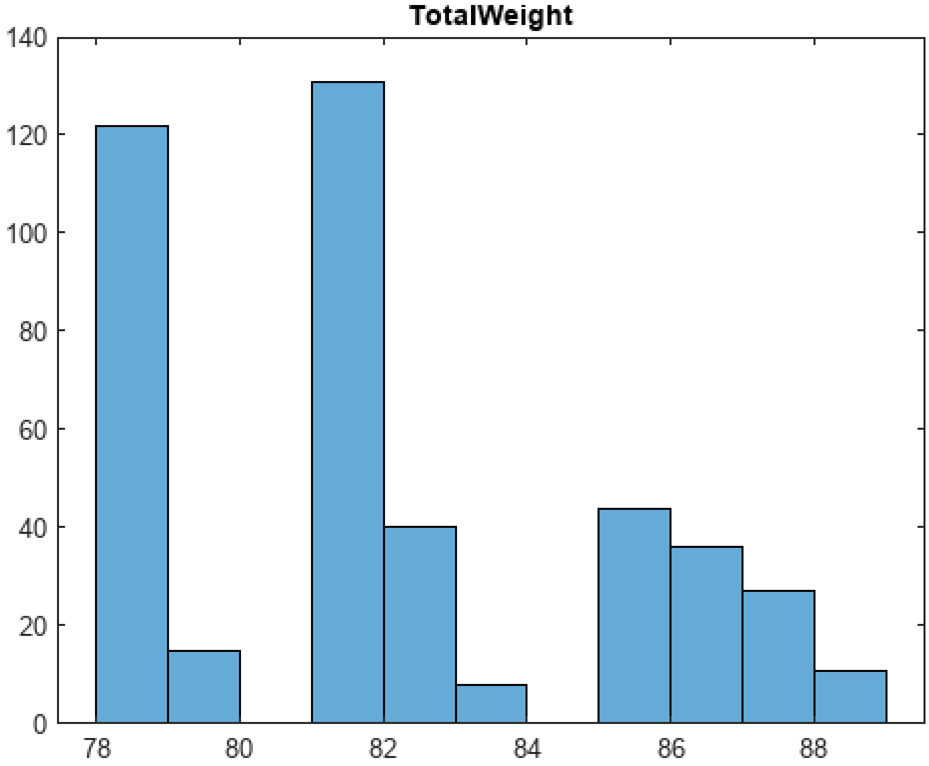

MaxSpeed
80
70
60
50
40
30
20
10
0
30
40
50
60
70
80
90
100
110

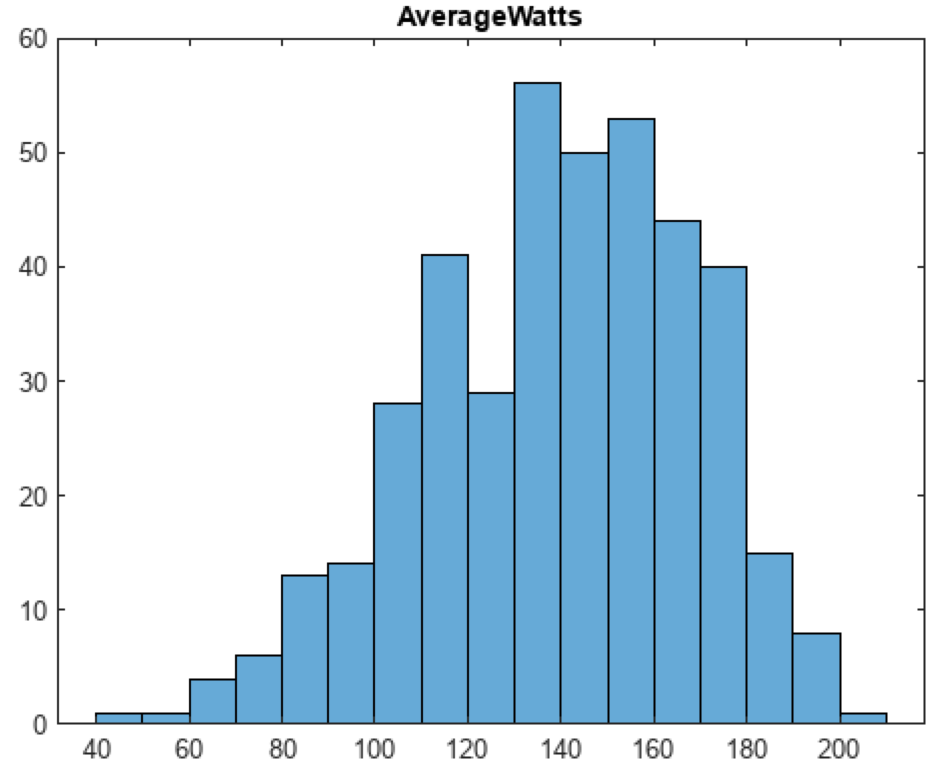

ElevationGain
120
100
80
60
40
20
0
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000

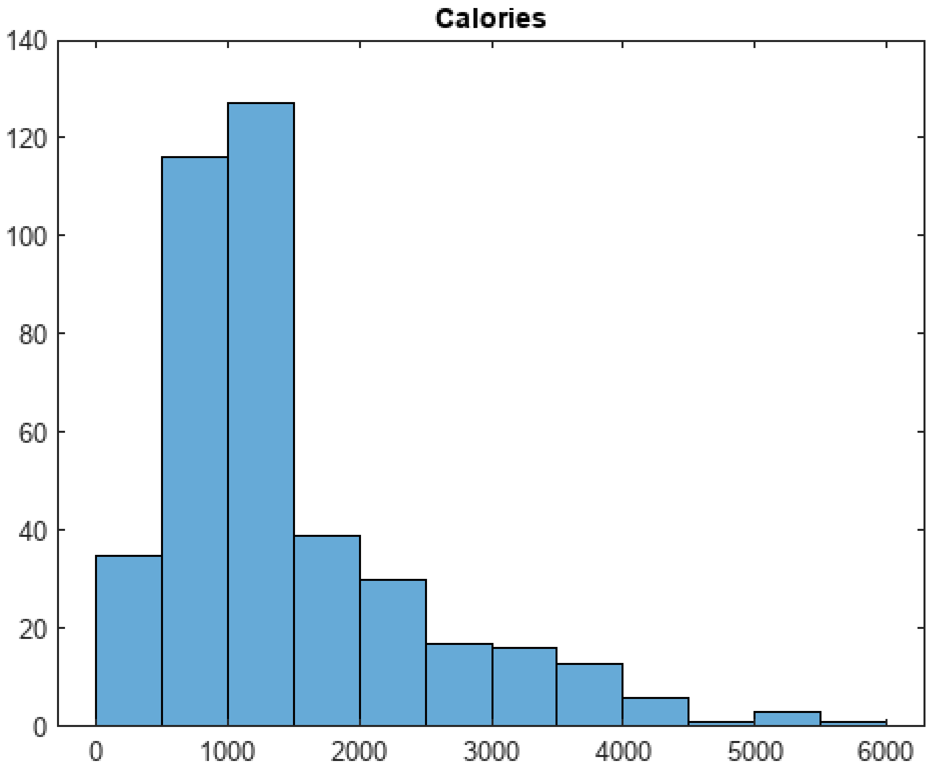

ElevationGainRatio
140
120
100
80
60
40
20
0
0
50
100
150
200
250

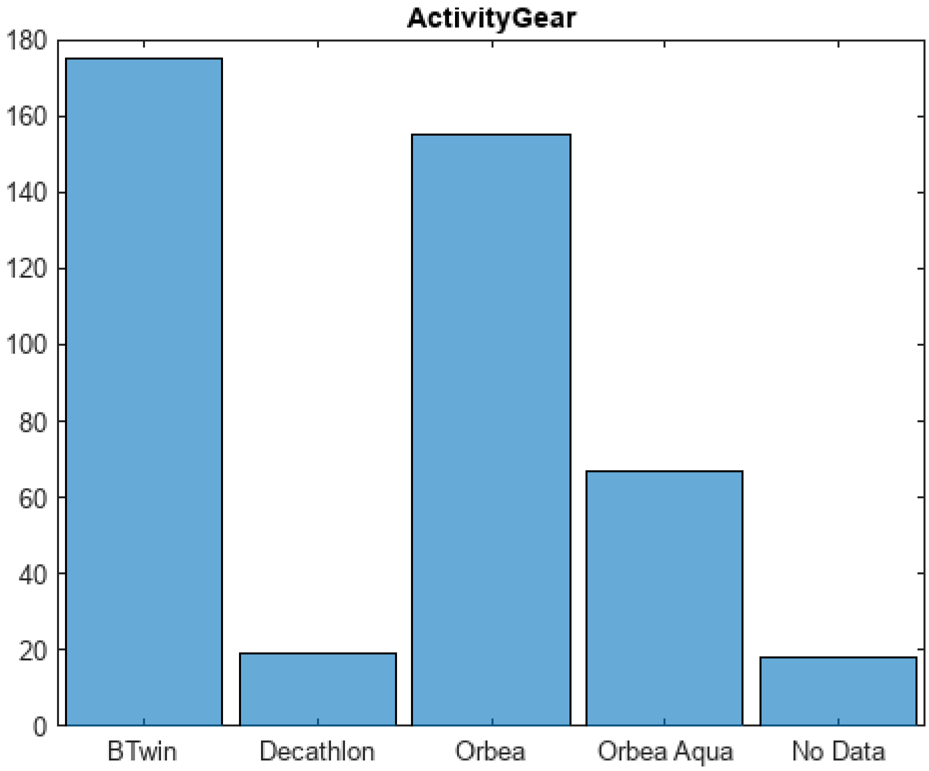

ElevationNet
140
120
100
80
60
40
20
0
-1000
-800
-600
-400
-200
0
200
400
600
800
1000


ElevationHigh
140
120
100
80
60
40
20
0
0
500
1000
1500
2000
2500
3000

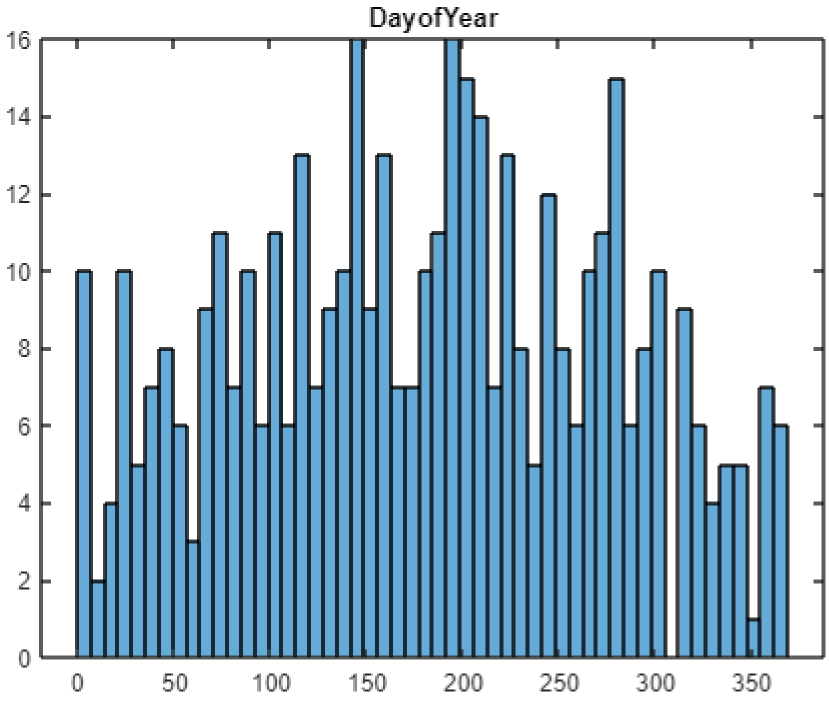

TotalWeight
140
120
100
80
60
40
20
0
78
80
82
84
86
88


AverageWatts
60
50
40
30
20
10
0
40
60
80
100
120
140
160
180
200





```
histogram(cyclingData.DayofYear,52)
title("DayofYear")
saveas(gcf,strcat('Figures/hist_DayofYear.png') )
```

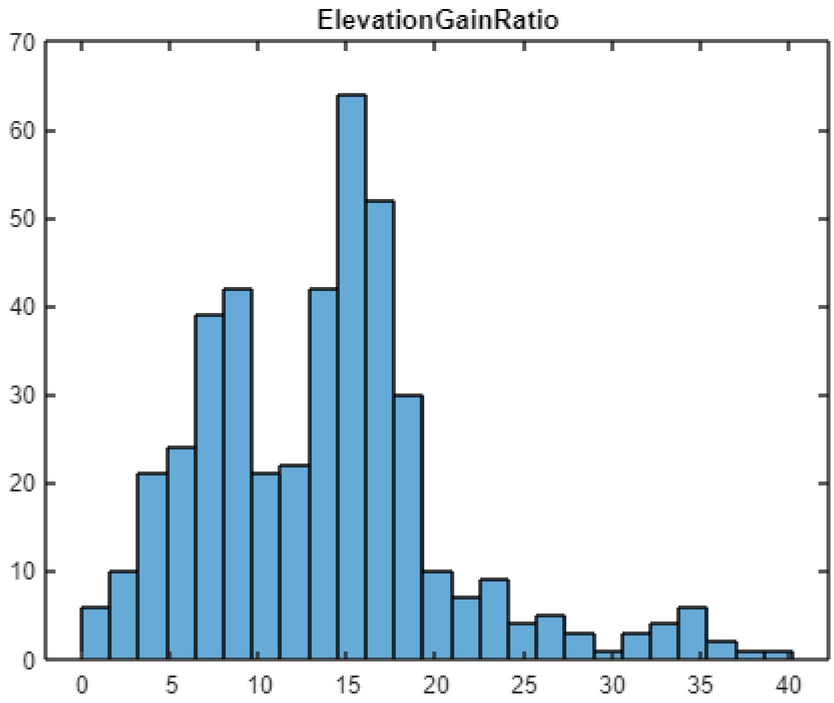


Remove outliers of ElevationGainRatio

```
cyclingData(cyclingData.ElevationGainRatio > 60, :) = [];

max_test = max(cyclingData.ElevationGainRatio);
min_test = min(cyclingData.ElevationGainRatio);
fprintf(strcat("Max/Min Elevation/Distance ratio [m/km]: ",num2str(max_test)," / ",num2str(min_
```

```
Max/Min Elevation/Distance ratio [m/km]: 40.1396 / 0
```

```
histogram(cyclingData.ElevationGainRatio, n_bins)
title("ElevationGainRatio")
saveas(gcf,strcat('Figures/hist_ElevationGainRatio.png') )
```

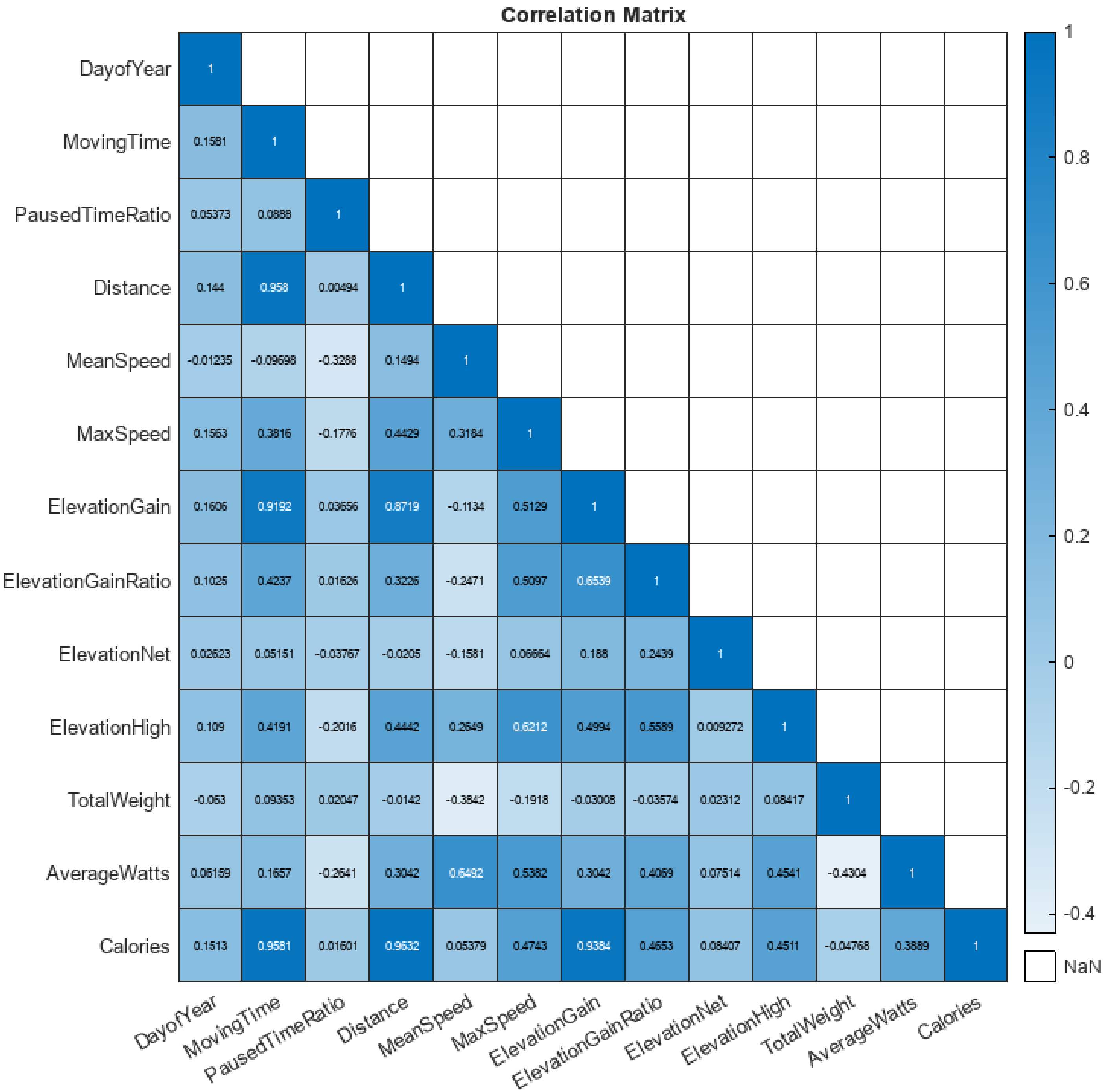


Save to csv the table to be analysed

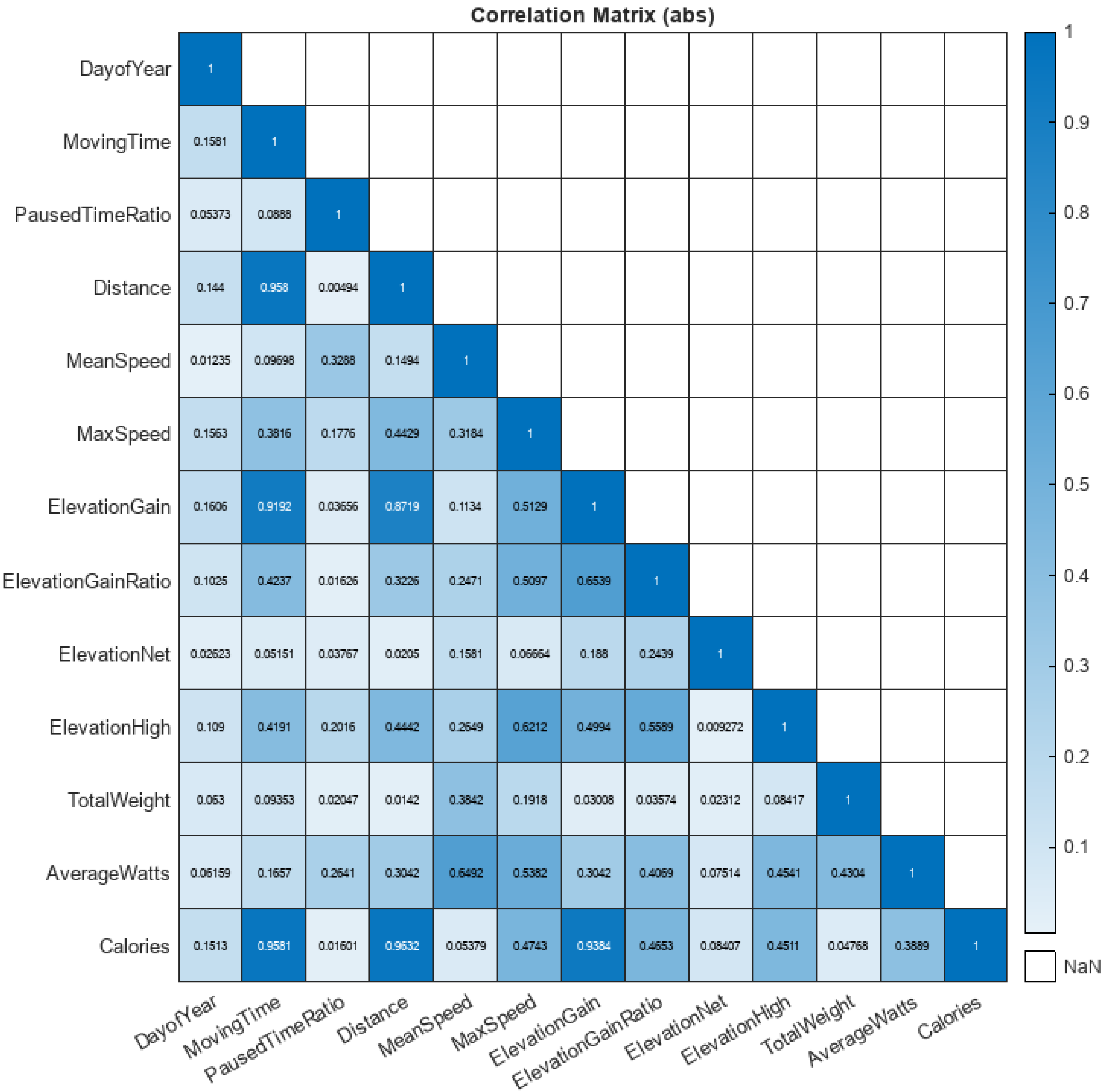
```
writetable(cyclingData,'./data/cyclingData_processed.csv')
```

Exploratory Data Analysis

```
numerical_vars = vars(vars~="ActivityID" &vars~="Year" & vars~="ActivityGear");

X = cyclingData(:,numerical_vars);
xnames = X.Properties.VariableNames;


r = corr(table2array(X),'rows', 'complete');
isupper = logical(triu(ones(size(r)),1));
r(isupper) = NaN;
figure('Position', [0 0 1200 1200])
h = heatmap(r,'MissingDataColor','w');
h.XDisplayLabels = xnames;
h.YDisplayLabels = xnames;
title("Correlation Matrix")
saveas(gcf,strcat('Figures/corr_matrix.png') )
```

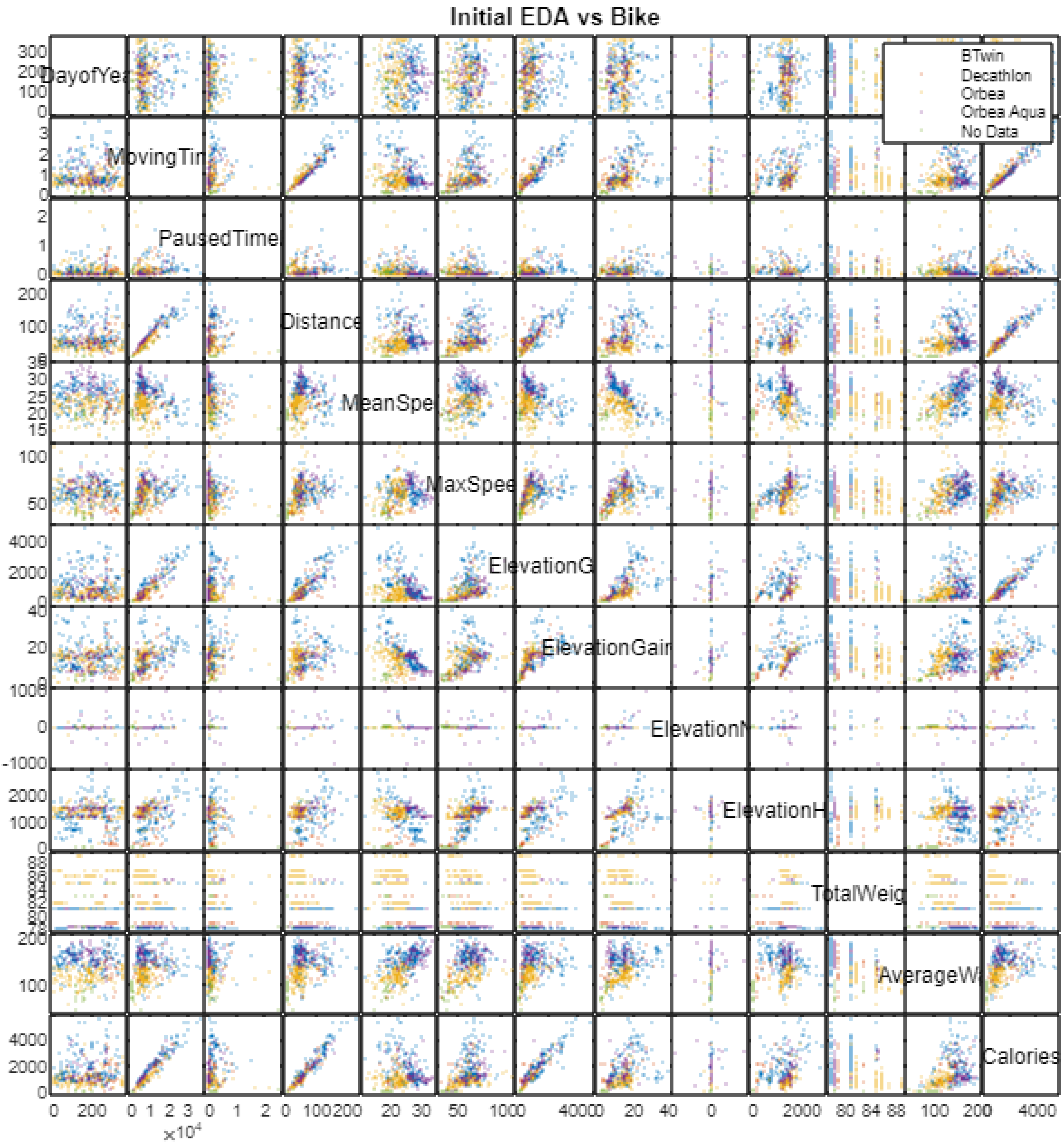


```
figure('Position', [0 0 1200 1200])
h = heatmap(abs(r),'MissingDataColor','w');
h.XDisplayLabels = xnames;
h.YDisplayLabels = xnames;
title("Correlation Matrix (abs)")
saveas(gcf,strcat('Figures/corr_matrix_abs.png') )
```

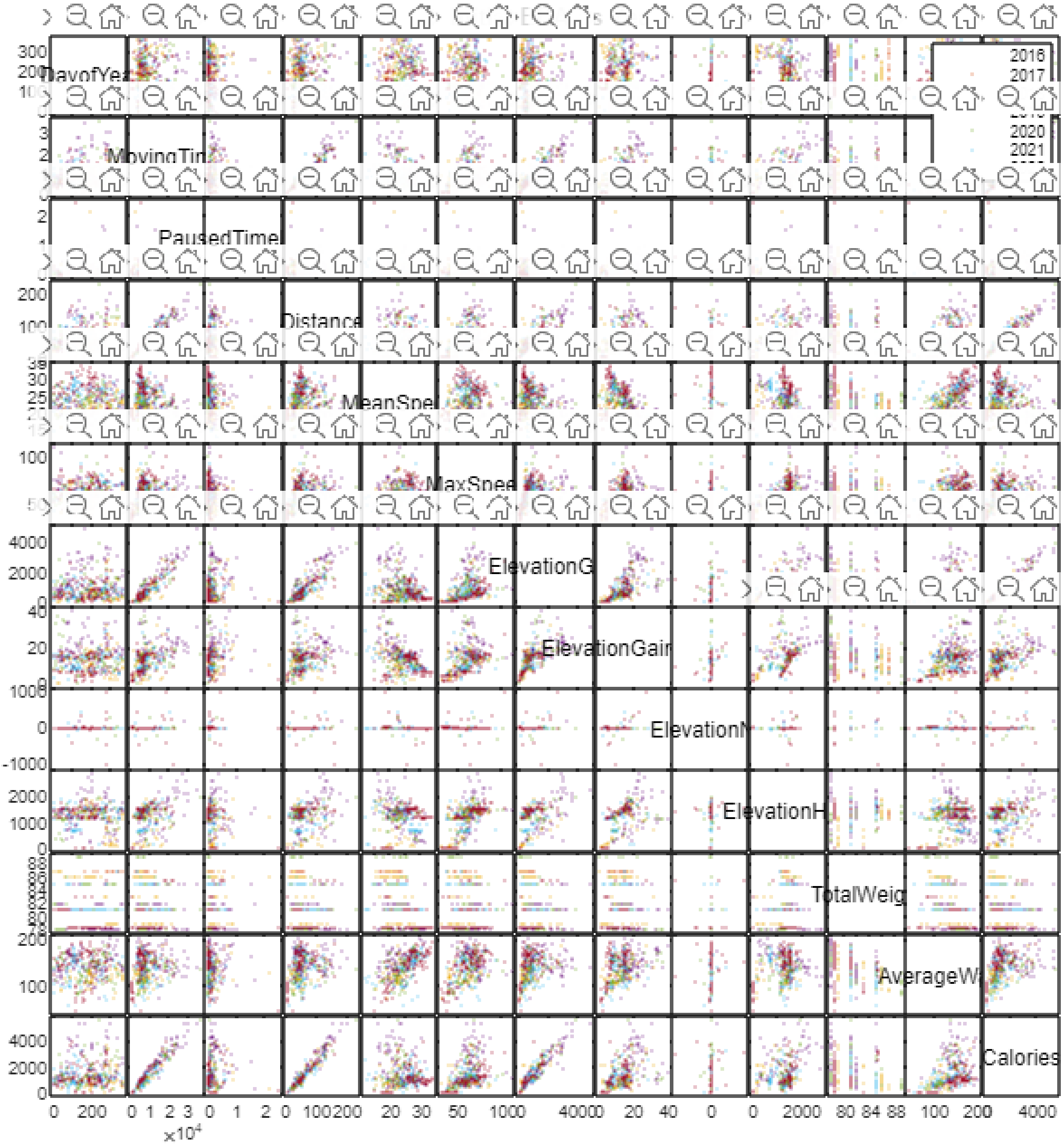


	DayofYear	MovingTime	PausedTimeRatio	Distance	MeanSpeed	MaxSpeed	ElevationGain	ElevationGainRatio	ElevationNet	ElevationHigh	TotalWeight	AverageWatts	Calories
DayofYear	1												
MovingTime	0.1581	1											
PausedTimeRatio	0.05373	0.0888	1										
Distance	0.144	0.958	0.00494	1									
MeanSpeed	0.01235	0.09698	0.3288	0.1494	1								
MaxSpeed	0.1563	0.3816	0.1776	0.4429	0.3184	1							
ElevationGain	0.1606	0.9192	0.03656	0.8719	0.1134	0.5129	1						
ElevationGainRatio	0.1025	0.4237	0.01626	0.3226	0.2471	0.5097	0.6539	1					
ElevationNet	0.02623	0.05151	0.03767	0.0205	0.1581	0.06664	0.188	0.2439	1				
ElevationHigh	0.109	0.4191	0.2016	0.4442	0.2649	0.6212	0.4994	0.5589	0.009272	1			
TotalWeight	0.063	0.09353	0.02047	0.0142	0.3842	0.1918	0.03008	0.03574	0.02312	0.08417	1		
AverageWatts	0.06159	0.1657	0.2641	0.3042	0.6492	0.5382	0.3042	0.4069	0.07514	0.4541	0.4304	1	
Calories	0.1513	0.9581	0.01601	0.9632	0.05379	0.4743	0.9384	0.4653	0.08407	0.4511	0.04768	0.3889	1

```
categories = unique(cyclingData.ActivityGear);
color = lines(size(categories));
figure('Position', [0 0 1200 1200])
[h, ax] = gplotmatrix(table2array(X),[],cyclingData.ActivityGear,color,[],[],[],'variable',xnar
title('Initial EDA vs Bike')
saveas(gcf,strcat('Figures/gplot_bike.png') )
```



```
categories = unique(cyclingData.Year);
color = lines(size(categories));
figure('Position', [0 0 1200 1200])
[h, ax] = gplotmatrix(table2array(X),[],cyclingData.Year,color,[],[],[],'variable',xnames);
title('Initial EDA vs Year')
saveas(gcf,strcat('Figures/gplot_year.png') )
```

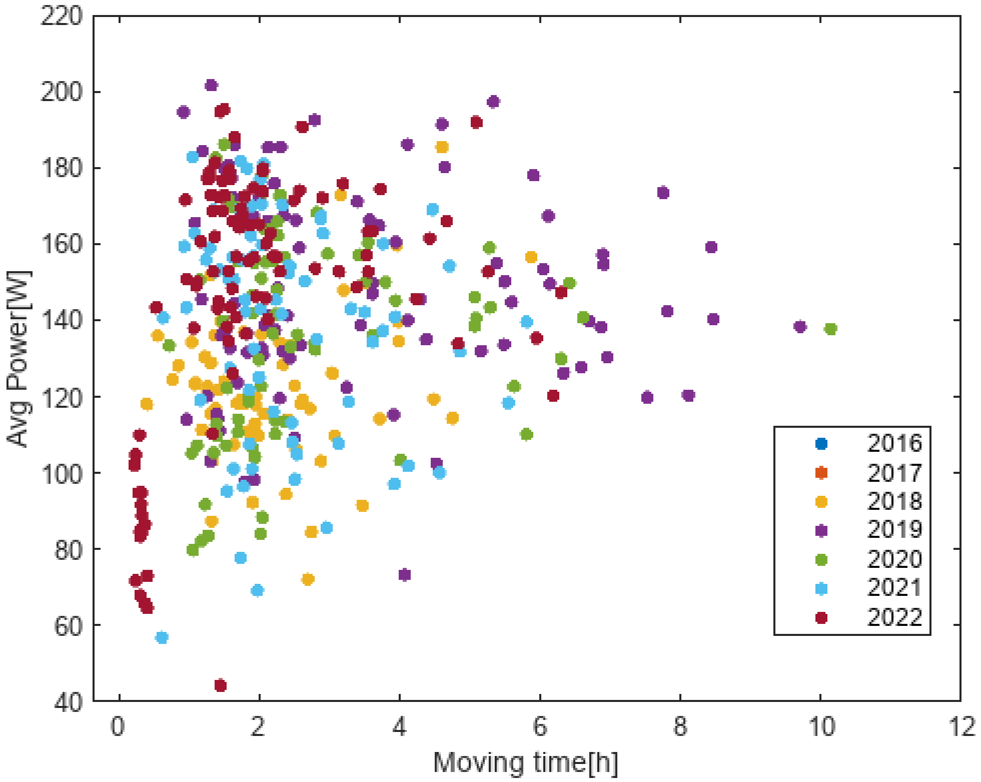


Power and Speed

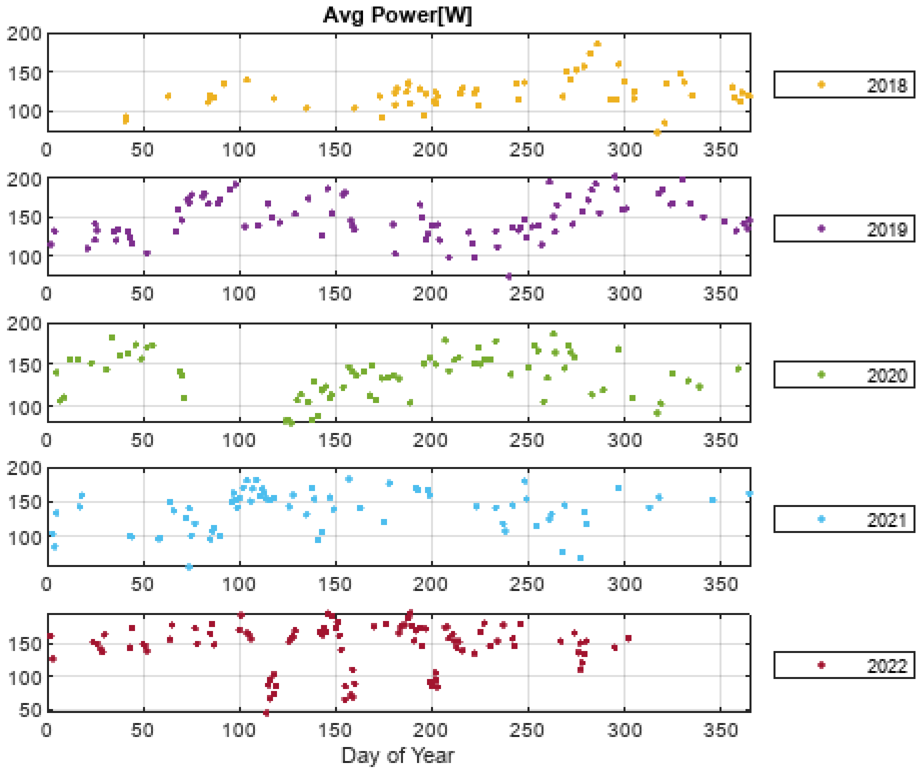
```
figure
gscatter(cyclingData.MovingTime./3600, cyclingData.AverageWatts, cyclingData.Year)
legend()
xlabel("Moving time[h]")
ylabel("Avg Power[W]")
```

```
Warning: Some output might be missing due to a network interruption. To get the missing output, rerun the
script.
```

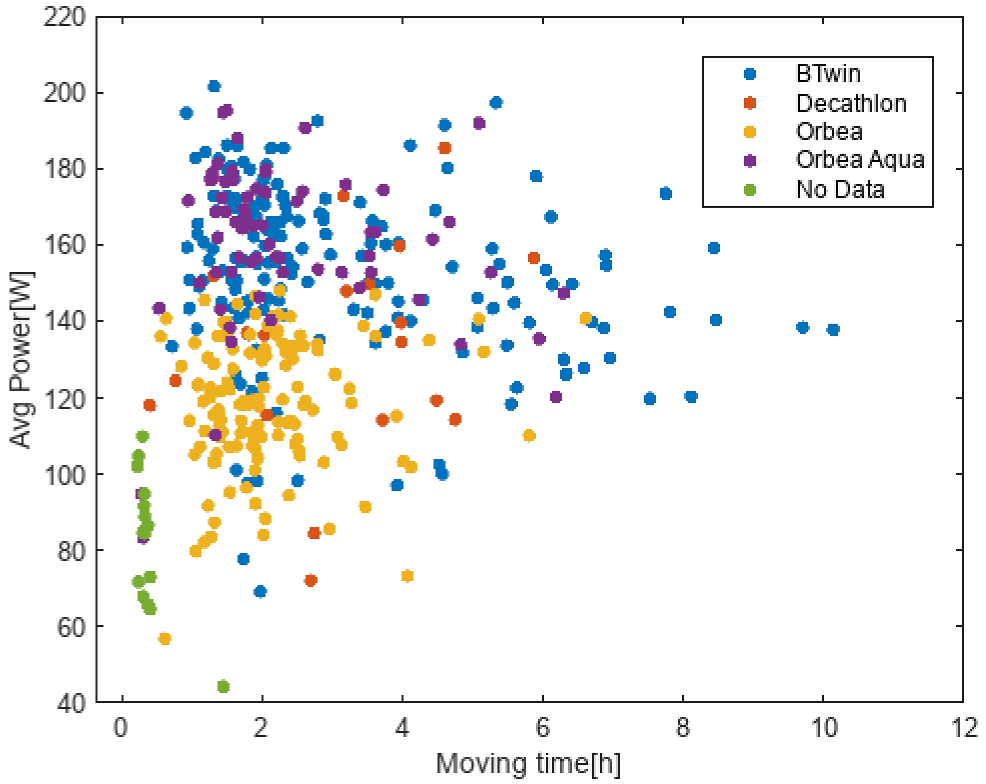
```
saveas(gcf,strcat('Figures/scatter_Power_Time_Year.png') )
```



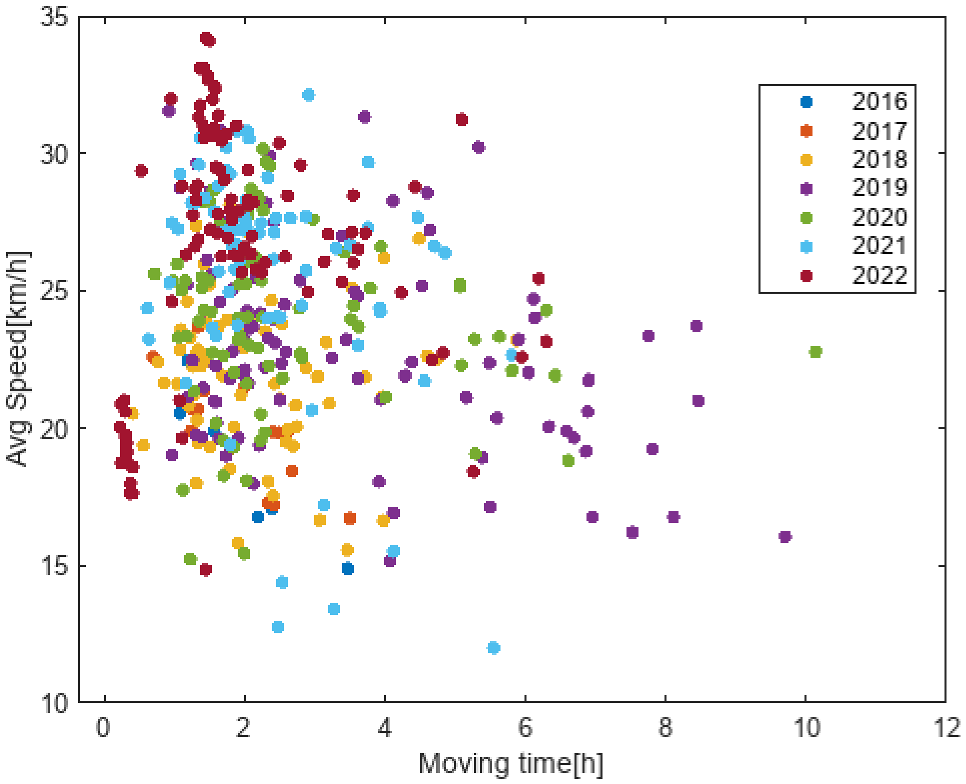
```
first = true;
figure
for year = 2018:2022
    subplot(5,1,year-2017)
    plot(cyclingData.DayofYear(cyclingData.Year==year,:), cyclingData.AverageWatts(cyclingData.
    legend(num2str(year), Location="eastoutside")
    xlim([0 366])
    grid()
    if first
        title("Avg Power[W]")
        first=false;
    end
end
xlabel("Day of Year")
saveas(gcf,strcat('Figures/scatter_Power_Day_Year.png') )
```



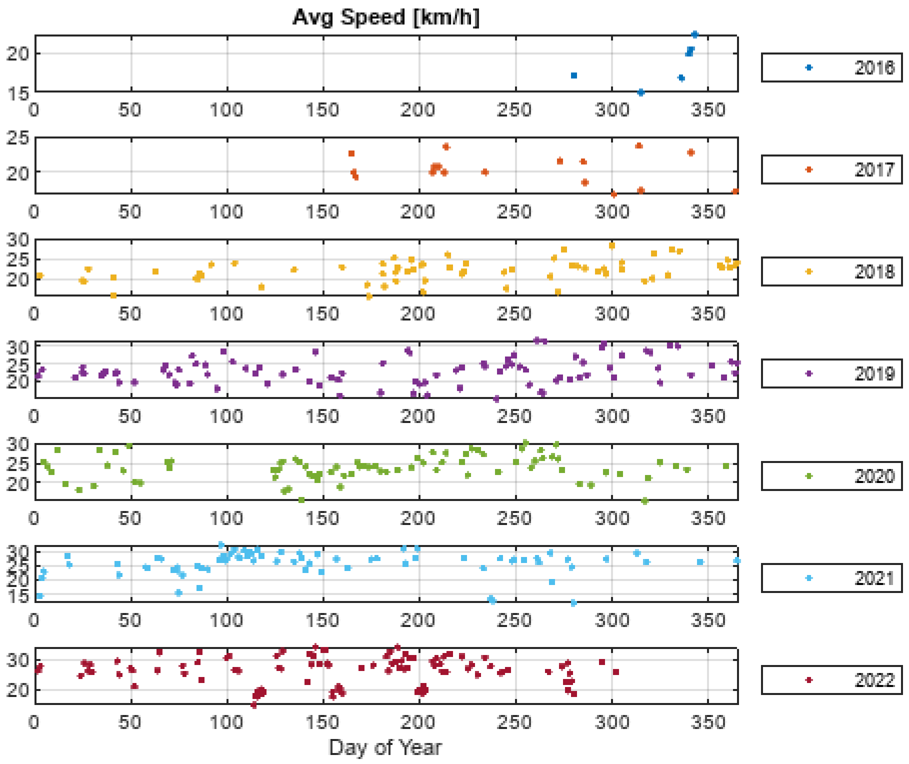
```
figure
gscatter(cyclingData.MovingTime./3600, cyclingData.AverageWatts, cyclingData.ActivityGear)
legend()
xlabel("Moving time[h]")
ylabel("Avg Power[W]")
saveas(gcf,strcat('Figures/scatter_Power_Time_Bike.png') )
```



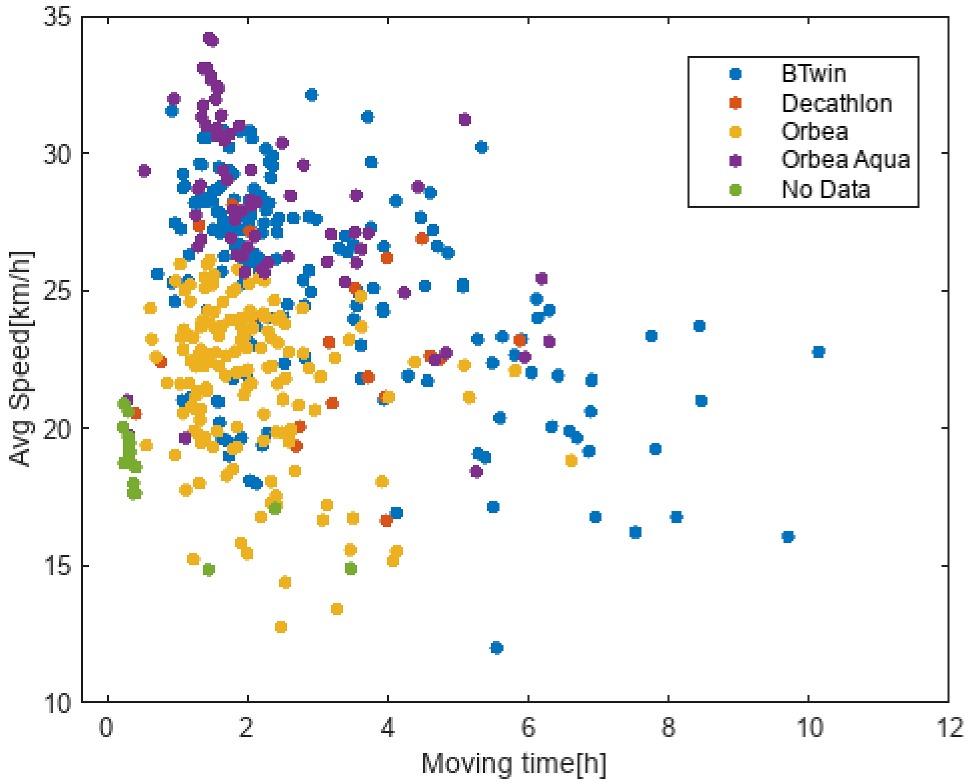
```
figure
gscatter(cyclingData.MovingTime./3600, cyclingData.MeanSpeed, cyclingData.Year)
legend()
xlabel("Moving time[h]")
ylabel("Avg Speed[km/h]")
saveas(gcf,strcat('Figures/scatter_Speed_Time_Year.png') )
```



```
first = true;
figure
for year = 2016:2022
    subplot(7,1,year-2015)
    plot(cyclingData.DayofYear(cyclingData.Year==year,:), cyclingData.MeanSpeed(cyclingData.Yea
    legend(num2str(year), Location="eastoutside")
    xlim([0 366])
    grid()
    if first
        title("Avg Speed [km/h]")
        first=false;
    end
end
xlabel("Day of Year")
saveas(gcf,strcat('Figures/scatter_Speed_Day_Year.png') )
```

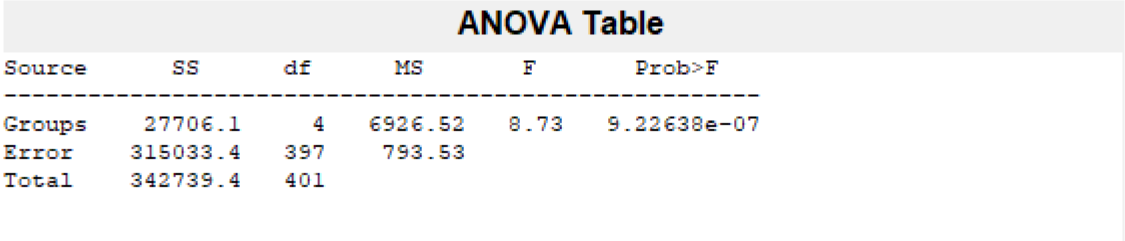


```
figure
gscatter(cyclingData.MovingTime./3600, cyclingData.MeanSpeed, cyclingData.ActivityGear)
legend()
xlabel("Moving time[h]")
ylabel("Avg Speed[km/h]")
saveas(gcf,strcat('Figures/scatter_Speed_Time_Bike.png') )
```



ANOVA test on Power and Speed

```
[p, tbl, stats] = anova1(cyclingData.AverageWatts, cyclingData.Year)
```



ANOVA Table

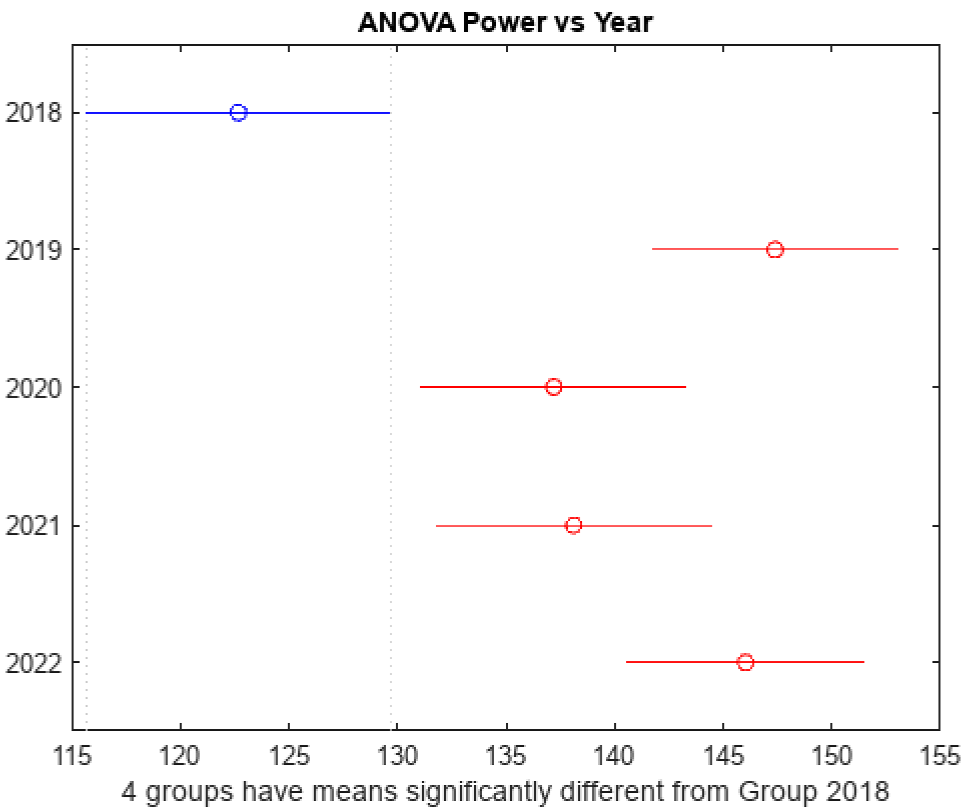
```
Source         SS       df       MS        F        Prob>F
---------------------------------------------------------
Groups     27706.1      4    6926.52    8.73    9.22638e-07
Error     315033.4    397     793.53
Total     342739.4    401
```

```
p = 9.2264e-07
tbl = 4×6 cell
```

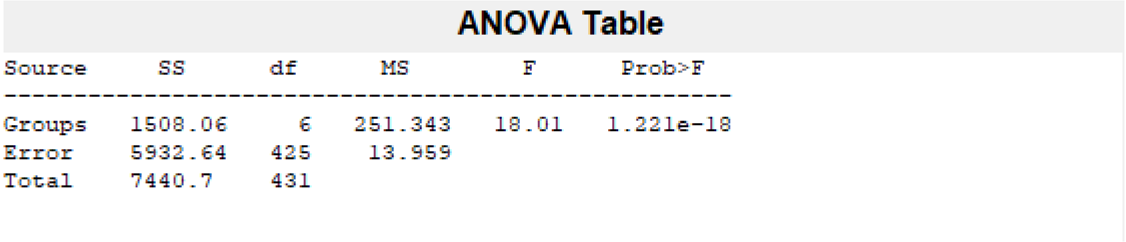
	1	2	3	4	5	6
1	'Source'	'SS'	'df'	'MS'	'F'	'Prob>F'
2	'Groups'	2.7706e+04	4	6.9265e+03	8.7287	9.2264e-07
3	'Error'	3.1503e+05	397	793.5349	[]	[]
4	'Total'	3.4274e+05	401	[]	[]	[]

```
stats = struct with fields:
    gnames: {5×1 cell}
         n: [61 92 78 73 98]
    source: 'anova1'
     means: [122.6784 147.3893 137.2023 138.1264 146.0298]
        df: 397
         s: 28.1698
```

```
multcompare(stats);
title("ANOVA Power vs Year")
```



```
saveas(gcf,strcat('Figures/ANOVA_Power_Year.png') )

[p, tbl, stats] = anova1(cyclingData.MeanSpeed, cyclingData.Year)
```



```
ANOVA Table
Source      SS       df       MS         F       Prob>F
-------------------------------------------------------
Groups   1508.06      6    251.343    18.01    1.221e-18
Error    5932.64    425     13.959
Total    7440.7     431
```

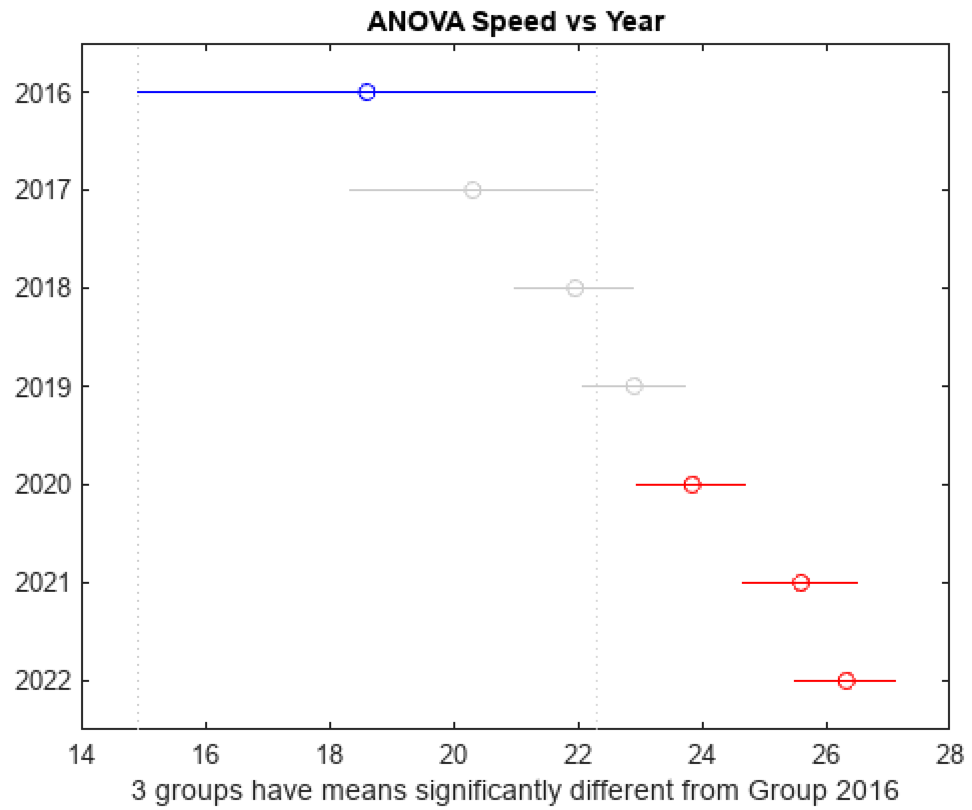
```
p = 1.2210e-18
tbl = 4×6 cell
```

	1	2	3	4	5	6
1	'Source'	'SS'	'df'	'MS'	'F'	'Prob>F'
2	'Groups'	1.5081e+03	6	251.3427	18.0056	1.2210e-18
3	'Error'	5.9326e+03	425	13.9592	[ ]	[ ]
4	'Total'	7.4407e+03	431	[ ]	[ ]	[ ]

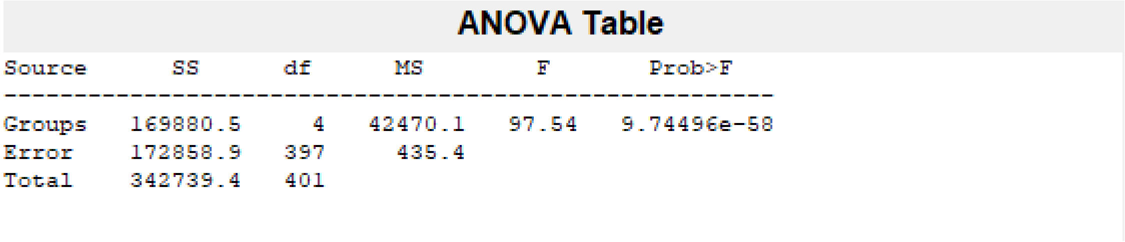
```
stats = struct with fields:
    gnames: {7×1 cell}
         n: [6 17 65 93 80 73 98]
    source: 'anova1'
     means: [18.6100 20.3152 21.9577 22.9115 23.8483 25.5979 26.3331]
```

```
        df: 425
         s: 3.7362
```

```
multcompare(stats);
title("ANOVA Speed vs Year")
saveas(gcf,strcat('Figures/ANOVA_Speed_Year.png') )
```



```

[p, tbl, stats] = anova1(cyclingData.AverageWatts, cyclingData.ActivityGear)
```



ANOVA Table

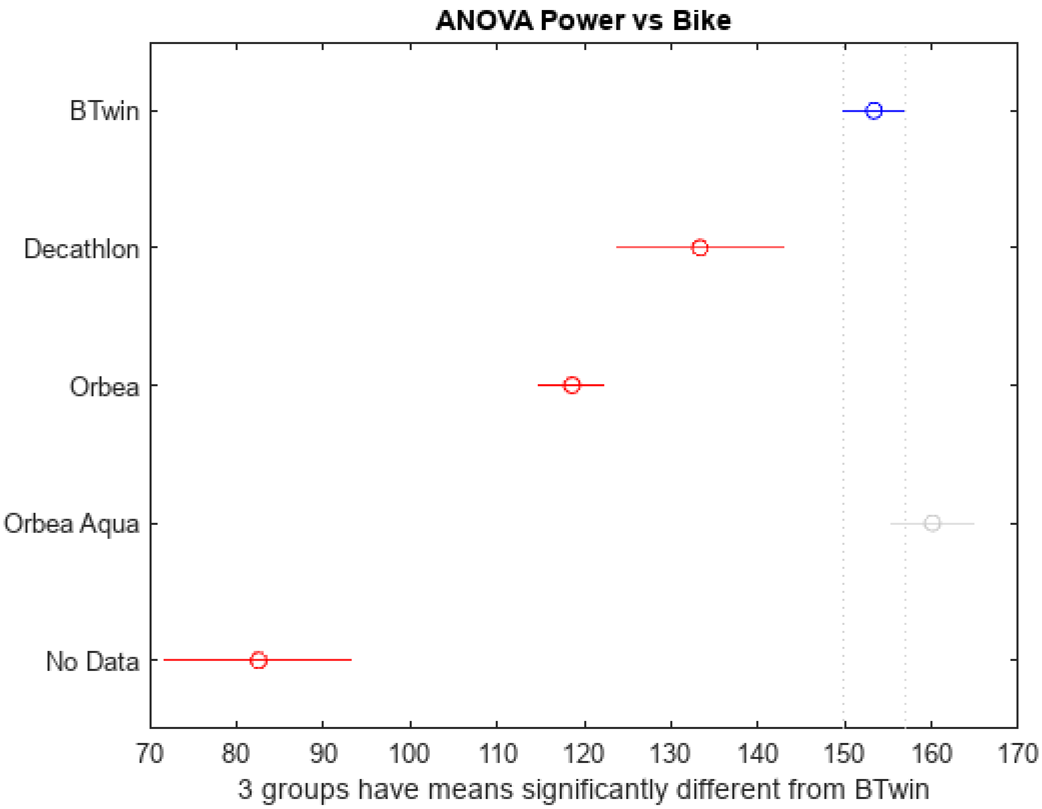
```
Source        SS       df       MS         F        Prob>F
------------------------------------------------------------
Groups    169880.5      4    42470.1    97.54    9.74496e-58
Error     172858.9    397      435.4
Total     342739.4    401
```

```
p = 9.7450e-58
tbl = 4×6 cell
```

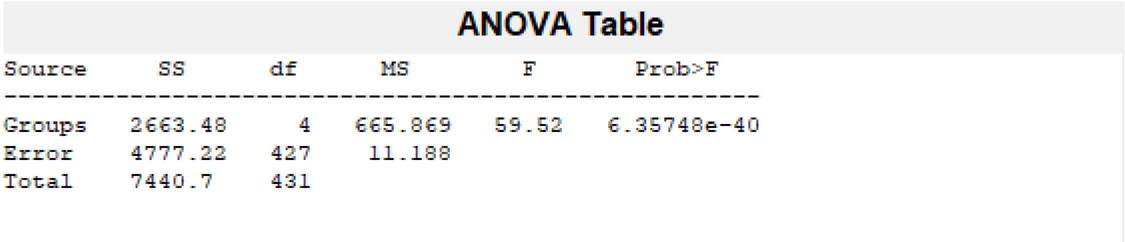
	1	2	3	4	5	6
1	'Source'	'SS'	'df'	'MS'	'F'	'Prob>F'
2	'Groups'	1.6988e+05	4	4.2470e+04	97.5399	9.7450e-58
3	'Error'	1.7286e+05	397	435.4129	[]	[]
4	'Total'	3.4274e+05	401	[]	[]	[]

```
stats = struct with fields:
    gnames: {5×1 cell}
         n: [172 19 128 67 16]
    source: 'anova1'
     means: [153.3552 133.3402 118.6049 160.0902 82.5680]
        df: 397
         s: 20.8665
```

```
multcompare(stats);
title("ANOVA Power vs Bike")
saveas(gcf,strcat('Figures/ANOVA_Power_Bike.png') )
```



```
[p, tbl, stats] = anova1(cyclingData.MeanSpeed, cyclingData.ActivityGear)
```



ANOVA Table

```
Source      SS       df      MS         F       Prob>F
-----------------------------------------------------
Groups    2663.48      4   665.869    59.52   6.35748e-40
Error     4777.22    427    11.188
Total     7440.7     431
```

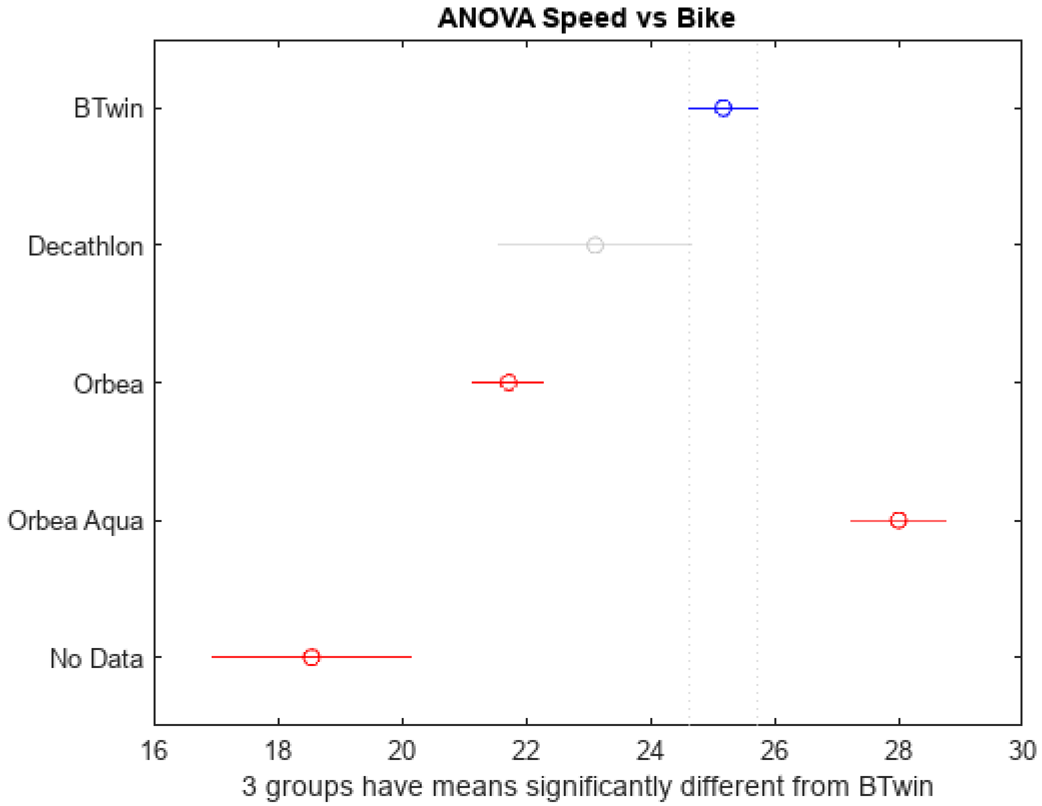
```
p = 6.3575e-40
tbl = 4×6 cell
```

	1	2	3	4	5	6
1	'Source'	'SS'	'df'	'MS'	'F'	'Prob>F'
2	'Groups'	2.6635e+03	4	665.8692	59.5171	6.3575e-40
3	'Error'	4.7772e+03	427	11.1879	[ ]	[ ]

	1	2	3	4	5	6
4	'Total'	7.4407e+03	431	[ ]	[ ]	[ ]

```
stats = struct with fields:
    gnames: {5×1 cell}
         n: [173 19 155 67 18]
    source: 'anova1'
     means: [25.1768 23.1137 21.7259 28.0008 18.5484]
        df: 427
         s: 3.3448
```

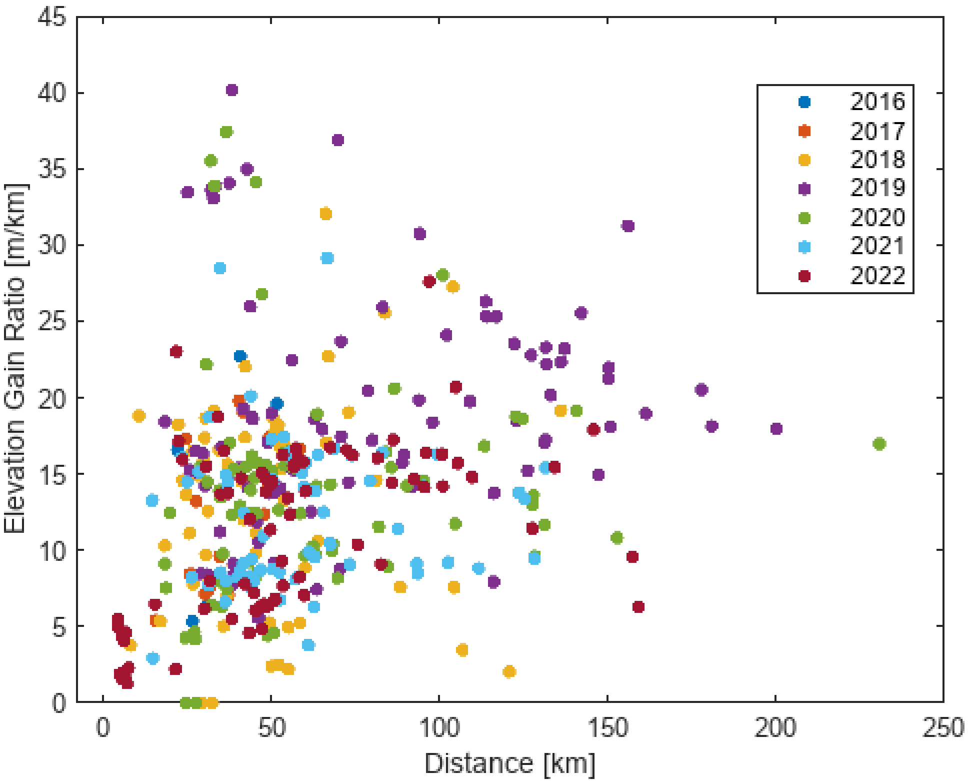
```
multcompare(stats);
title("ANOVA Speed vs Bike")
```



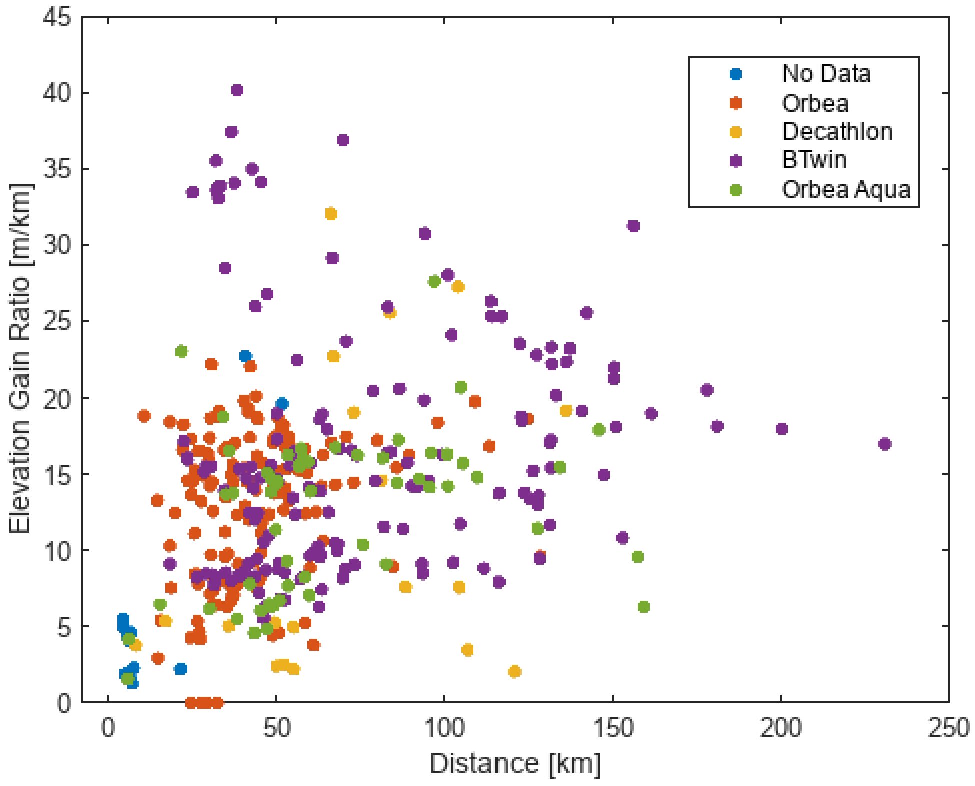
```
saveas(gcf,strcat('Figures/ANOVA_Speed_Bike.png') )
```

Explanation of differences between Power and Speed

```
figure
gscatter(cyclingData.Distance, cyclingData.ElevationGainRatio, cyclingData.Year)
legend()
xlabel("Distance [km]")
ylabel("Elevation Gain Ratio [m/km]")
saveas(gcf,strcat('Figures/scatter_hilly_year.png'))
```



```
figure
gscatter(cyclingData.Distance, cyclingData.ElevationGainRatio, cyclingData.ActivityGear)
legend()
xlabel("Distance [km]")
ylabel("Elevation Gain Ratio [m/km]")
saveas(gcf,strcat('Figures/scatter_hilly_bike.png'))
```



Regresion models

The aim is reproducing the Average Speed (directly related to **Moving Time**) and Average Watts (dorectly related to **Calories spent**)

```
% Check functional relation between variables
corr(cyclingData.Calories, cyclingData.MovingTime.*cyclingData.AverageWatts, 'rows','complete'
```

```
ans = 1.0000
```

Remove ID (not used), Moving Time, Calories (directly related to output) and ElevationGain (redundant) spent from tabel for regression

```
regression_vars = vars(vars~="ActivityID" & vars~="MovingTime" & vars~="Calories" & vars~="Elev
                       & vars~="MeanSpeed" & vars~="AverageWatts");

X = cyclingData(:,regression_vars)
```

X = 432×10 table

...

	Year	DayofYear	PausedTimeRatio	Distance	MaxSpeed
1	2016	280	0.0076	40.8600	64.4400
2	2016	315	0.1258	51.7300	47.8800

	Year	DayofYear	PausedTimeRatio	Distance	MaxSpeed
3	2016	336	0.0843	36.7600	57.2400
4	2016	340	0.0482	31.1900	39.9600
5	2016	341	0.1899	22.1200	54.3600
6	2016	343	0.0012	26.6200	39.2400
7	2017	165	0.5296	15.6500	40.6800
8	2017	166	0.0388	47.8600	59.0400
9	2017	167	0.0639	34.5900	46.8000
10	2017	207	0.0349	52.1400	48.2400
11	2017	208	0	25.8800	NaN
12	2017	210	0.3630	27.6600	70.9200
13	2017	213	0.0545	49.3900	64.8000
14	2017	214	0.0219	37.3100	60.4800

⋮

```
xnames = X.Properties.VariableNames;
ySpeed = cyclingData(:,"MeanSpeed");
yPower = cyclingData(:,"AverageWatts");
idx_pow = ~isnan(yPower{:,:});

% Create indexes for cross-validation
idx = floor(1 + n_cvpartitions*rand(size(X,1),1));
err = zeros(n_cvpartitions,1);
```

Fill NaN values so methods don´t fail

```
X.MaxSpeed = fillmissing(X.MaxSpeed, "constant", mean(X.MaxSpeed,"omitnan"));
% X.ElevationGain = fillmissing(X.ElevationGain, "constant", 0);
X.ElevationGainRatio = fillmissing(X.ElevationGainRatio, "constant", 0);
X.ElevationNet = fillmissing(X.ElevationNet, "constant", 0);
X.ElevationHigh = fillmissing(X.ElevationHigh, "constant", 1000);
```

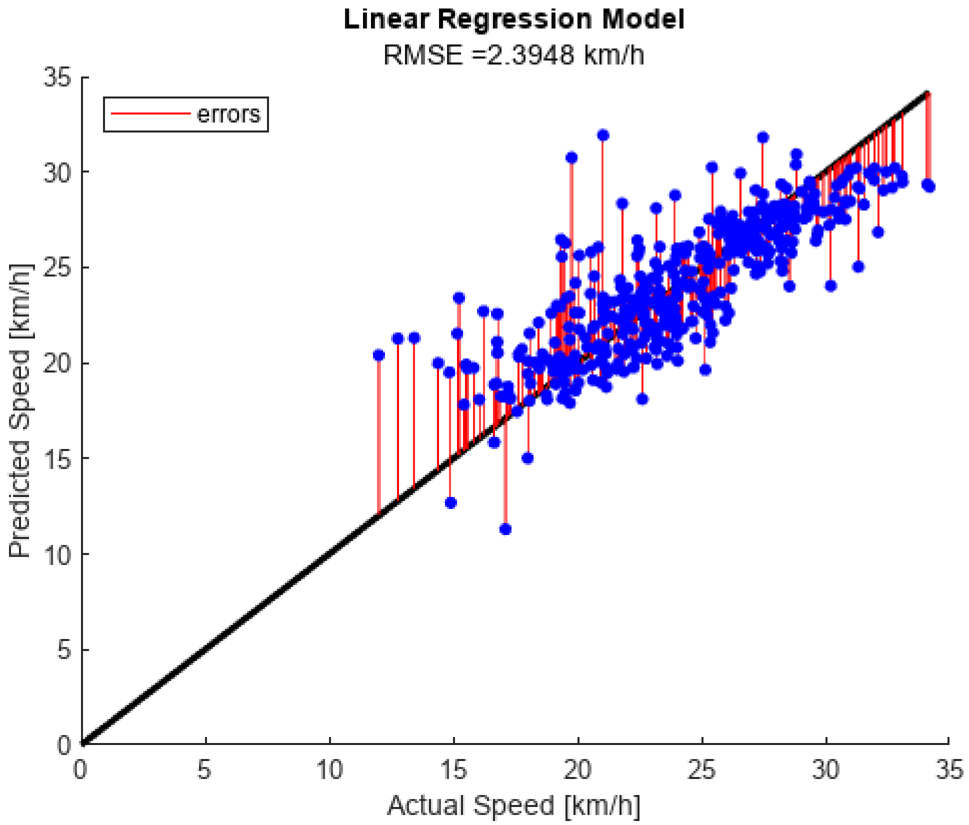
Linear regression model

```
yPredict = zeros(size(ySpeed,1),1);
coeffs = zeros(1+size(X,2)+size(unique(X(:,1)),1)-3+size(unique(X(:,end)),1)-1,n_cvpartitions)
% Manual cross validation
for ii=1:n_cvpartitions
    mdl = fitlm([X(idx~=ii,:), ySpeed(idx~=ii,:)], "CategoricalVars", [1, size(X,2)]);
    yPredict(idx==ii,:) = predict(mdl, X(idx==ii,:));
    err(ii) = sqrt(mean( (table2array(ySpeed(idx==ii,:)) - yPredict(idx==ii,:)).^2 ));
    coeffs(:,ii) = mdl.Coefficients.Estimate;
end
```

```
rmse_Speed = mean(err)
```

```
rmse_Speed = 2.3948
```

```

y_plot = table2array(ySpeed);
figure
hold on
plot([0, max(table2array(ySpeed))], [0, max(table2array(ySpeed))], 'k', LineWidth=2.5)
for ii = 1:size(y_plot)
    plot([y_plot(ii), y_plot(ii)], [y_plot(ii), yPredict(ii)], 'r-', LineWidth=0.15)
end
plot(y_plot, yPredict, 'b.', MarkerSize=16)
xlabel('Actual Speed [km/h]')
ylabel('Predicted Speed [km/h]')
legend('','errors', 'Location', 'northwest')
title('Linear Regression Model')
subtitle(strcat('RMSE = ', num2str(rmse_Speed), ' km/h'))
hold off
saveas(gcf,strcat('Figures/Regression_Linear_Speed.png') )
```



```
% Formula coeffs
mdl.Coefficients
```

```
ans = 19×4 table
```

	Estimate	SE	tStat	pValue
1 (Intercept)	37.5415	6.3647	5.8984	0

	Estimate	SE	tStat	pValue
2 Year_2017	-1.6744	1.4427	-1.1606	0.2466
3 Year_2018	-0.6428	1.3117	-0.4900	0.6244
4 Year_2019	-0.5332	1.3433	-0.3969	0.6917
5 Year_2020	-0.2590	1.3136	-0.1972	0.8438
6 Year_2021	-0.3070	1.3410	-0.2289	0.8191
7 Year_2022	-0.8198	1.4336	-0.5719	0.5678
8 DayofYear	-0.0001	0.0015	-0.0462	0.9632
9 PausedTimeRatio	-0.1545	0.6363	-0.2429	0.8082
10 Distance	-0.0108	0.0046	-2.3533	0.0191
11 MaxSpeed	0.0723	0.0128	5.6558	0
12 ElevationGainRatio	-0.3528	0.0237	-14.9170	0
13 ElevationNet	-0.0023	0.0014	-1.7242	0.0855
14 ElevationHigh	-0.0004	0.0004	-0.9233	0.3565

⋮

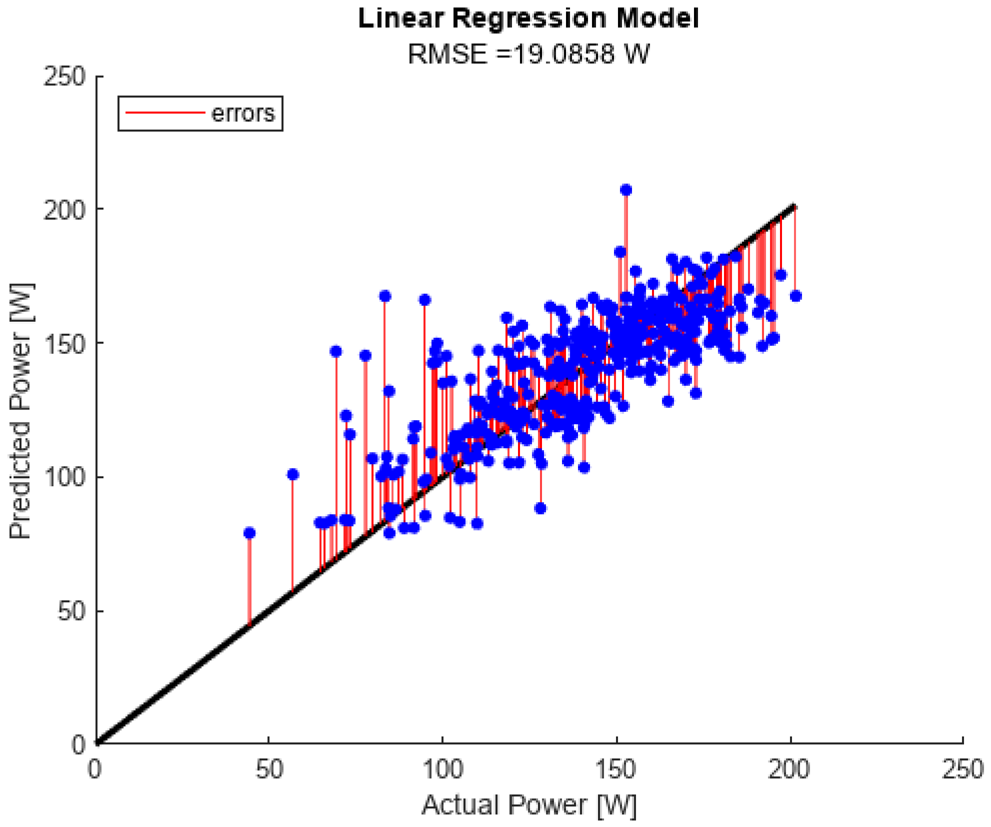
```
mean(coeffs,2)
```

```
ans = 19×1
   34.4070
   -1.5064
   -0.3659
   -0.1002
    0.0213
   -0.1046
   -0.6613
    0.0003
   -0.7204
   -0.0111
      ⋮
```

```
yPredict = zeros(size(yPower,1),1);
coeffs = zeros(1+size(X,2)+size(unique(X(:,1)),1)-5+size(unique(X(:,end)),1)-1,n_cvpartitions)
% Manual cross validation
for ii=1:n_cvpartitions
    mdl = fitlm([X(idx~=ii & idx_pow,:), yPower(idx~=ii & idx_pow,:)], "CategoricalVars", [1, s
    yPredict(idx==ii & idx_pow,:) = predict(mdl, X(idx==ii & idx_pow,:));
    err(ii) = sqrt(mean( (table2array(yPower(idx==ii & idx_pow,:)) - yPredict(idx==ii & idx_pow
    coeffs(:,ii) = mdl.Coefficients.Estimate;
end
rmse_Power = mean(err)
```

```
rmse_Power = 19.0858
```

```
y_plot = table2array(yPower(idx_pow,:));
yPredict = yPredict(idx_pow,:);
figure
hold on
plot([0, max(table2array(yPower))], [0, max(table2array(yPower))], 'k', LineWidth=2.5)
for ii = 1:size(y_plot)
    plot([y_plot(ii), y_plot(ii)], [y_plot(ii), yPredict(ii)], 'r-', LineWidth=0.15)
end
plot(y_plot, yPredict, 'b.', MarkerSize=16)
xlabel('Actual Power [W]')
ylabel('Predicted Power [W]')
legend('','errors', 'Location', 'northwest')
title('Linear Regression Model')
subtitle(strcat('RMSE = ', num2str(rmse_Power), ' W'))
hold off
saveas(gcf,strcat('Figures/Regression_Linear_Power.png') )
```



```
% Formula coeffs
mdl.Coefficients
```

```
ans = 17×4 table
```

	Estimate	SE	tStat	pValue
1 (Intercept)	172.4806	51.5511	3.3458	0.0009
2 Year_2019	-0.8106	4.4970	-0.1803	0.8571
3 Year_2020	0.9786	4.2530	0.2301	0.8182

	Estimate	SE	tStat	pValue
4 Year_2021	-7.0534	4.7141	-1.4962	0.1355
5 Year_2022	2.5230	7.3219	0.3446	0.7306
6 DayofYear	0.0035	0.0122	0.2913	0.7710
7 PausedTimeRatio	-1.7360	5.1101	-0.3397	0.7343
8 Distance	-0.0149	0.0368	-0.4040	0.6865
9 MaxSpeed	0.5054	0.1028	4.9169	0
10 ElevationGainRatio	1.0823	0.1998	5.4167	0
11 ElevationNet	0.0027	0.0108	0.2458	0.8060
12 ElevationHigh	-0.0166	0.0033	-5.0088	0
13 TotalWeight	-0.8974	0.5923	-1.5152	0.1306
14 ActivityGear_Decathlon	4.0468	7.5725	0.5344	0.5934

⋮

```
mean(coeffs,2)
```

```
ans = 17×1
  163.2850
    0.4259
    0.4541
   -6.8489
    2.3664
    0.0035
   -6.4054
   -0.0167
    0.4956
    1.0862
      ⋮
```

Superlinear model

```
yPredict = zeros(size(yPower,1),1);
% Manual cross validation
for ii=1:n_cvpartitions
    mdl = fitlm([X(idx~=ii,:), ySpeed(idx~=ii,:)], 'interactions', "CategoricalVars", [1, size(
    yPredict(idx==ii,:) = predict(mdl, X(idx==ii,:));
    err(ii) = sqrt(mean( (table2array(ySpeed(idx==ii,:)) - yPredict(idx==ii,:)).^2 ));
end
```

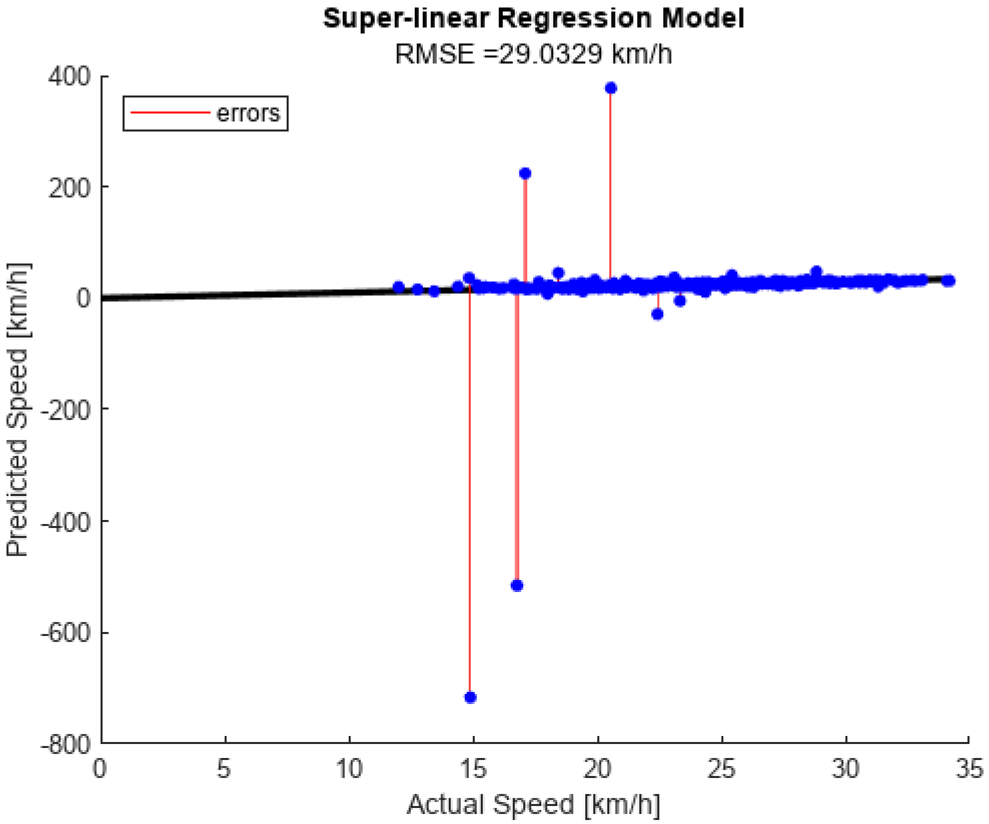
```
Warning: Regression design matrix is rank deficient to within machine precision.
Warning: Regression design matrix is rank deficient to within machine precision.
Warning: Regression design matrix is rank deficient to within machine precision.
Warning: Regression design matrix is rank deficient to within machine precision.
Warning: Regression design matrix is rank deficient to within machine precision.
Warning: Regression design matrix is rank deficient to within machine precision.
Warning: Regression design matrix is rank deficient to within machine precision.
Warning: Regression design matrix is rank deficient to within machine precision.
Warning: Regression design matrix is rank deficient to within machine precision.
```

```
Warning: Regression design matrix is rank deficient to within machine precision.
```

```
rmse_Speed = mean(err)
```

```
rmse_Speed = 29.0329
```

```

y_plot = table2array(ySpeed);
figure
hold on
plot([0, max(table2array(ySpeed))], [0, max(table2array(ySpeed))], 'k', LineWidth=2.5)
for ii = 1:size(y_plot)
    plot([y_plot(ii), y_plot(ii)], [y_plot(ii), yPredict(ii)], 'r-', LineWidth=0.15)
end
plot(y_plot, yPredict, 'b.', MarkerSize=16)
xlabel('Actual Speed [km/h]')
ylabel('Predicted Speed [km/h]')
legend('','errors', 'Location', 'northwest')
title('Super-linear Regression Model')
subtitle(strcat('RMSE = ', num2str(rmse_Speed), ' km/h'))
hold off
saveas(gcf,strcat('Figures/Regression_SupLinear_Speed.png') )
```



```

yPredict = zeros(size(yPower,1),1);
% Manual cross validation
for ii=1:n_cvpartitions
```

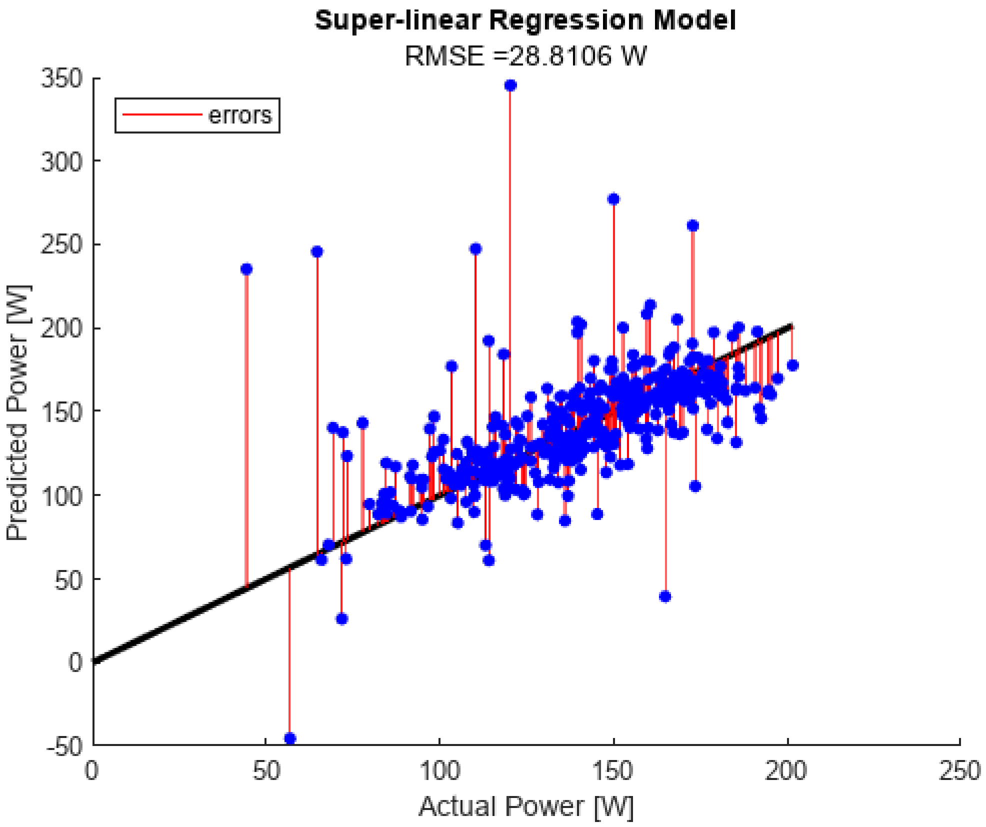
```
    mdl = fitlm([X(idx~=ii & idx_pow,:), yPower(idx~=ii & idx_pow,:)], 'interactions', "Categor
    yPredict(idx==ii & idx_pow,:) = predict(mdl, X(idx==ii & idx_pow,:));
    err(ii) = sqrt(mean( (table2array(yPower(idx==ii & idx_pow,:)) - yPredict(idx==ii & idx_pow
end
```

```
Warning: Regression design matrix is rank deficient to within machine precision.
Warning: Regression design matrix is rank deficient to within machine precision.
Warning: Regression design matrix is rank deficient to within machine precision.
Warning: Regression design matrix is rank deficient to within machine precision.
Warning: Regression design matrix is rank deficient to within machine precision.
Warning: Regression design matrix is rank deficient to within machine precision.
Warning: Regression design matrix is rank deficient to within machine precision.
Warning: Regression design matrix is rank deficient to within machine precision.
Warning: Regression design matrix is rank deficient to within machine precision.
Warning: Regression design matrix is rank deficient to within machine precision.
```

```
rmse_Power = mean(err)
```

```
rmse_Power = 28.8106
```

```

y_plot = table2array(yPower(idx_pow,:));
yPredict = yPredict(idx_pow,:);
figure
hold on
plot([0, max(table2array(yPower))], [0, max(table2array(yPower))], 'k', LineWidth=2.5)
for ii = 1:size(y_plot)
    plot([y_plot(ii), y_plot(ii)], [y_plot(ii), yPredict(ii)], 'r-', LineWidth=0.15)
end
plot(y_plot, yPredict, 'b.', MarkerSize=16)
xlabel('Actual Power [W]')
ylabel('Predicted Power [W]')
legend('','errors', 'Location', 'northwest')
title('Super-linear Regression Model')
subtitle(strcat('RMSE = ', num2str(rmse_Power), ' W'))
hold off
saveas(gcf,strcat('Figures/Regression_SupLinear_Power.png') )
```



Non-parametric Regression Models

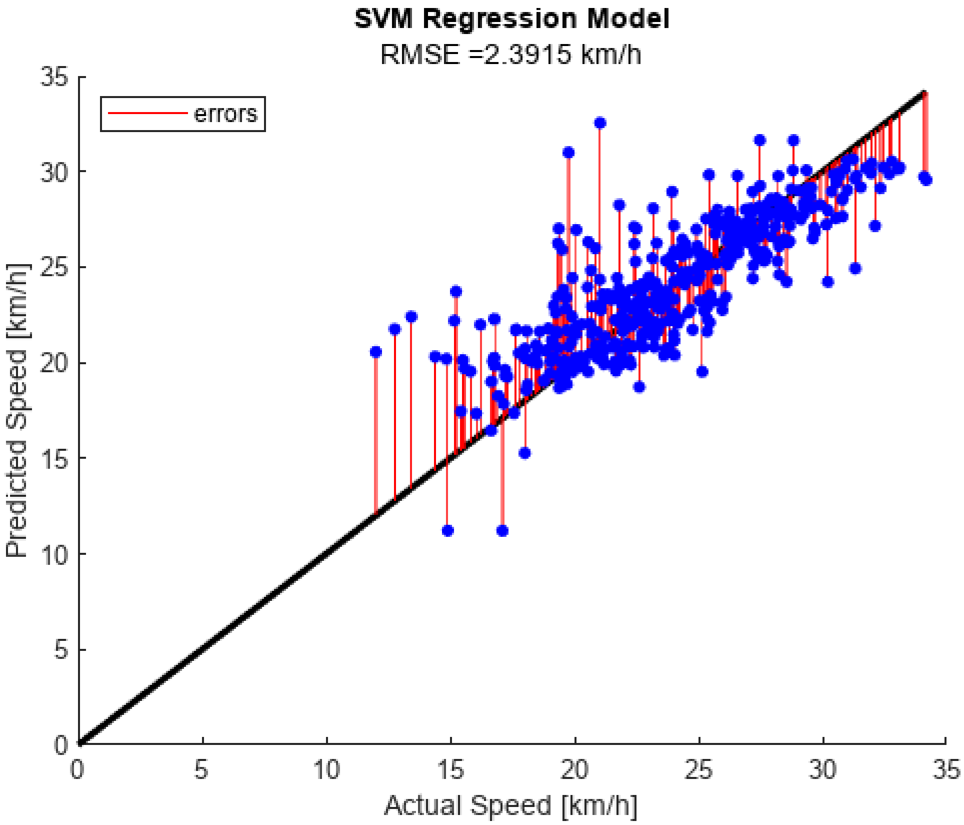
Support Vector Machine

```
yPredict = zeros(size(yPower,1),1);
% Manual cross validation
for ii=1:n_cvpartitions
    mdl = fitrsvm( X(idx~=ii,:), ySpeed(idx~=ii,:), "Standardize", true);
    yPredict(idx==ii,:) = predict(mdl, X(idx==ii,:));
    err(ii) = sqrt(mean( (table2array(ySpeed(idx==ii,:)) - yPredict(idx==ii,:)).^2 ));
end
rmse_Speed = mean(err)
```

```
rmse_Speed = 2.3915
```

```
y_plot = table2array(ySpeed);
figure
hold on
plot([0, max(table2array(ySpeed))], [0, max(table2array(ySpeed))], 'k', LineWidth=2.5)
for ii = 1:size(y_plot)
    plot([y_plot(ii), y_plot(ii)], [y_plot(ii), yPredict(ii)], 'r-', LineWidth=0.15)
end
plot(y_plot, yPredict, 'b.', MarkerSize=16)
xlabel('Actual Speed [km/h]')
ylabel('Predicted Speed [km/h]')
```

```
legend('','errors', 'Location', 'northwest')
title('SVM Regression Model')
subtitle(strcat('RMSE = ', num2str(rmse_Speed), ' km/h'))
hold off
saveas(gcf,strcat('Figures/Regression_SVM_Speed.png') )
```

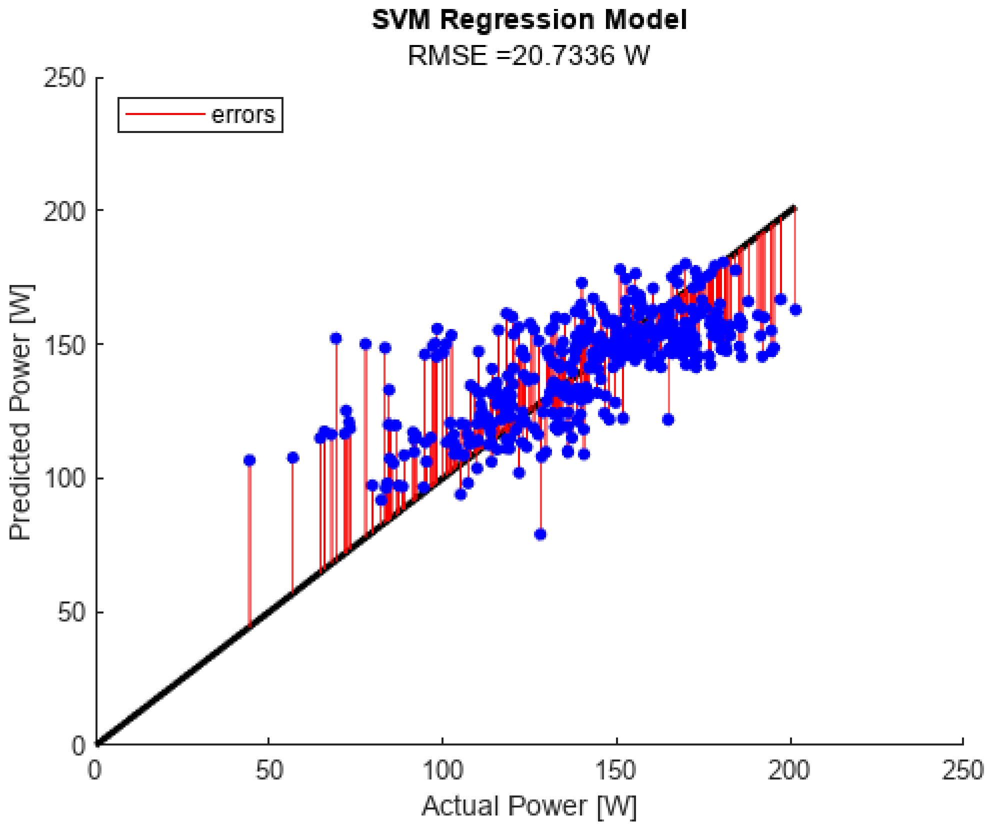


```
yPredict = zeros(size(yPower,1),1);
% Manual cross validation
for ii=1:n_cvpartitions
    mdl = fitrsvm( X(idx~=ii & idx_pow,:), yPower(idx~=ii & idx_pow,:), "Standardize", true);
    yPredict(idx==ii & idx_pow,:) = predict(mdl, X(idx==ii & idx_pow,:));
    err(ii) = sqrt(mean( (table2array(yPower(idx==ii & idx_pow,:)) - yPredict(idx==ii & idx_pow
end
rmse_Power = mean(err)
```

```
rmse_Power = 20.7336
```

```
y_plot = table2array(yPower(idx_pow,:));
yPredict = yPredict(idx_pow,:);
figure
hold on
plot([0, max(table2array(yPower))], [0, max(table2array(yPower))], 'k', LineWidth=2.5)
for ii = 1:size(y_plot)
    plot([y_plot(ii), y_plot(ii)], [y_plot(ii), yPredict(ii)], 'r-', LineWidth=0.15)
end
```

```
plot(y_plot, yPredict, 'b.', MarkerSize=16)
xlabel('Actual Power [W]')
ylabel('Predicted Power [W]')
legend('','errors', 'Location', 'northwest')
title('SVM Regression Model')
subtitle(strcat('RMSE = ', num2str(rmse_Power), ' W'))
hold off
saveas(gcf,strcat('Figures/Regression_SVM_Power.png') )
```



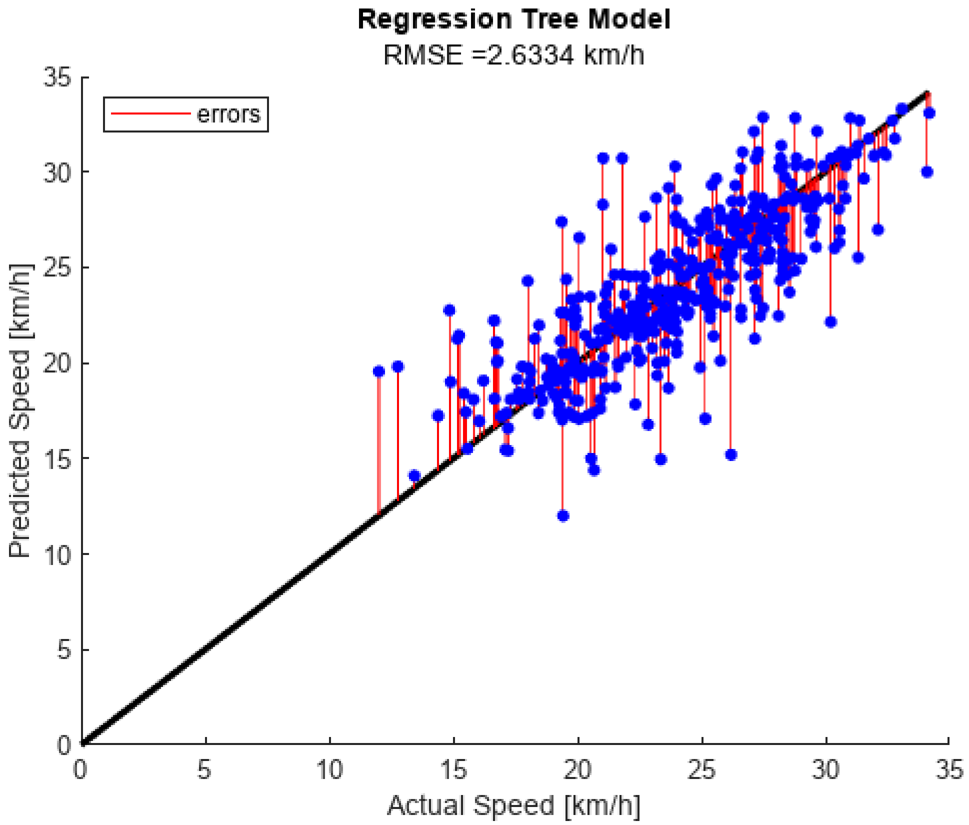
Regression Tree

```
mdl = fitrtree(X, ySpeed, "CategoricalPredictors", [1, size(X,2)], 'Kfold', n_cvpartitions);
yPredict = kfoldPredict(mdl);
rmse_Speed = sqrt(kfoldLoss(mdl))
```

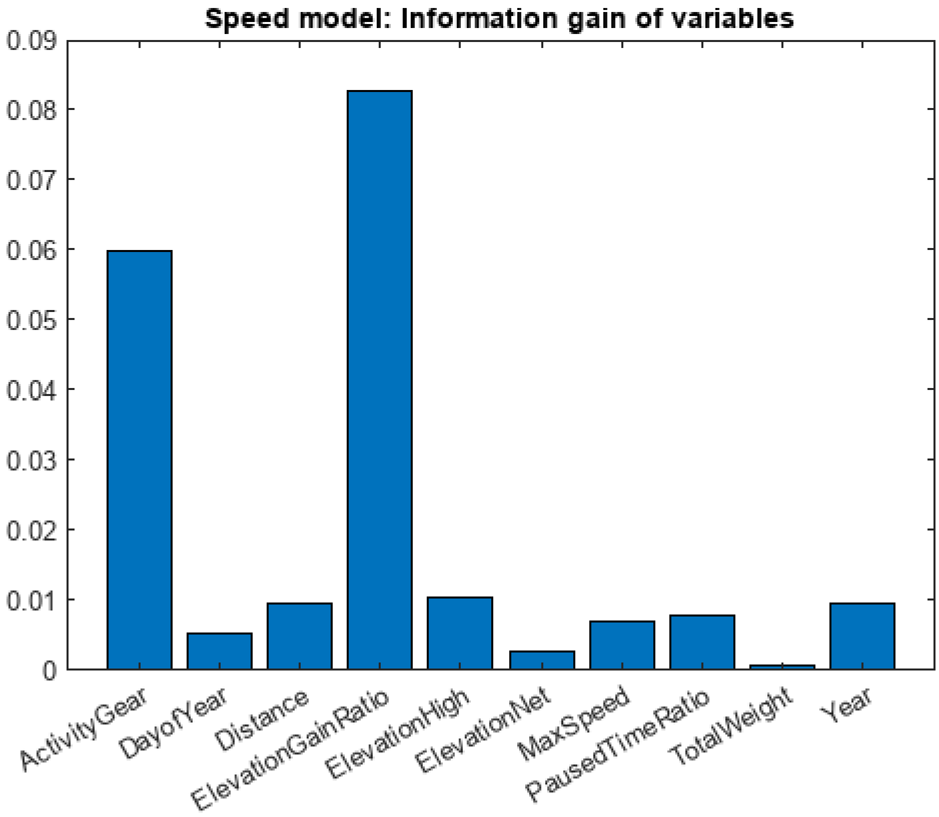
```
rmse_Speed = 2.6334
```

```

y_plot = table2array(ySpeed);
figure
hold on
plot([0, max(table2array(ySpeed))], [0, max(table2array(ySpeed))], 'k', LineWidth=2.5)
for ii = 1:size(y_plot)
    plot([y_plot(ii), y_plot(ii)], [y_plot(ii), yPredict(ii)], 'r-', LineWidth=0.15)
end
plot(y_plot, yPredict, 'b.', MarkerSize=16)
xlabel('Actual Speed [km/h]')
```

```
ylabel('Predicted Speed [km/h]')
legend('','errors', 'Location', 'northwest')
title('Regression Tree Model')
subtitle(strcat('RMSE = ', num2str(rmse_Speed), ' km/h'))
hold off
saveas(gcf,strcat('Figures/Regression_Tree_Speed.png') )
```



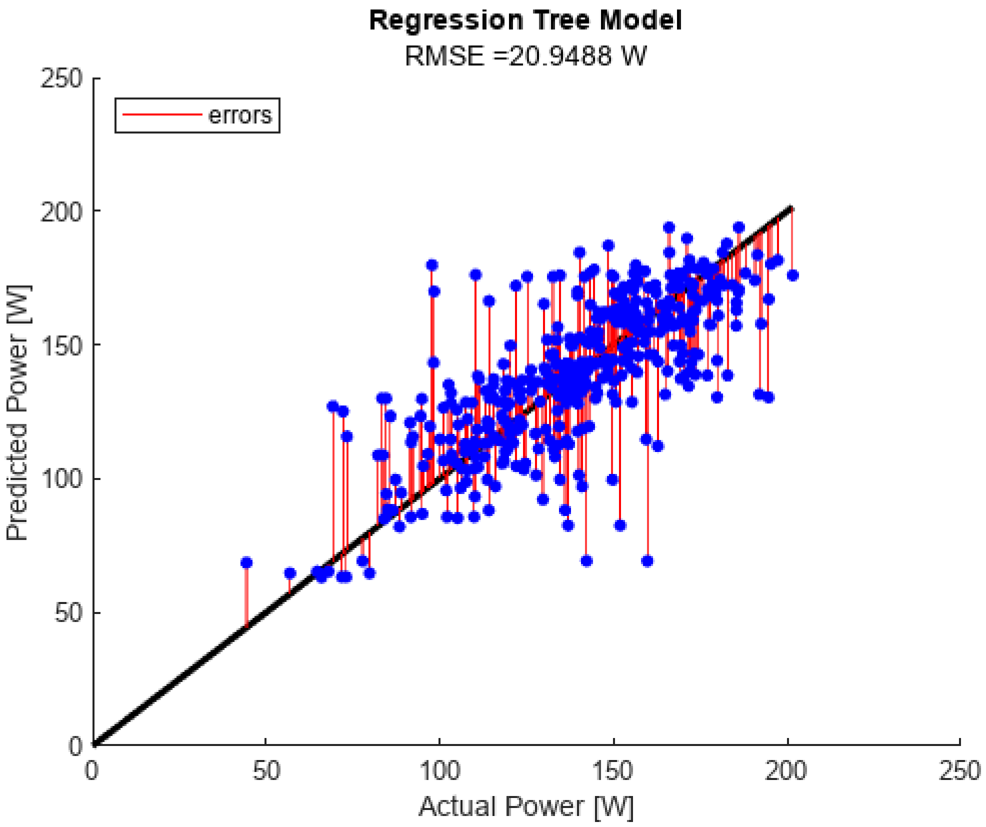
```
importance = zeros(mdl.KFold,size(mdl.X,2));
for ii = 1:mdl.KFold
    importance(ii,:) = predictorImportance(mdl.Trained{ii,1});
end
figure
bar(categorical(X.Properties.VariableNames), mean(importance,1))
title('Speed model: Information gain of variables')
saveas(gcf,strcat('Figures/Regression_Tree_Speed_varImportance.png') )
```



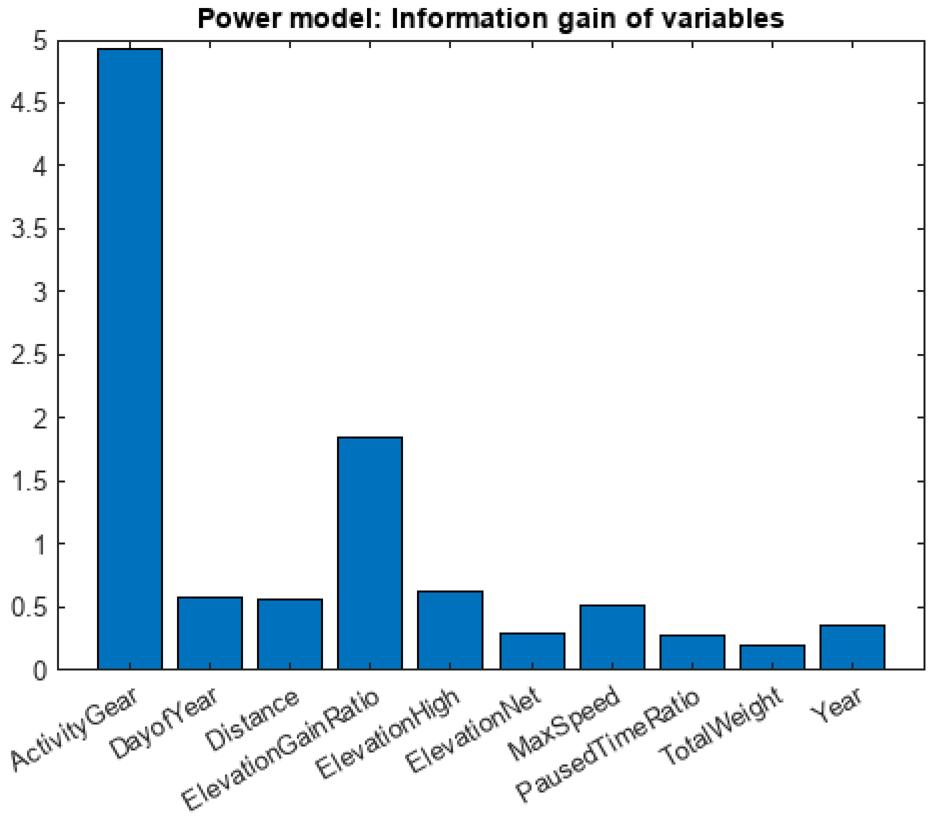
```
mdl = fitrtree(X(idx_pow,:), yPower(idx_pow,:), "CategoricalPredictors", [1, size(X,2)], 'Kfol
yPredict = kfoldPredict(mdl);
rmse_Power = sqrt(kfoldLoss(mdl))
```

```
rmse_Power = 20.9488
```

```
y_plot = table2array(yPower(idx_pow,:));
figure
hold on
plot([0, max(table2array(yPower))], [0, max(table2array(yPower))], 'k', LineWidth=2.5)
for ii = 1:size(y_plot)
    plot([y_plot(ii), y_plot(ii)], [y_plot(ii), yPredict(ii)], 'r-', LineWidth=0.15)
end
plot(y_plot, yPredict, 'b.', MarkerSize=16)
xlabel('Actual Power [W]')
ylabel('Predicted Power [W]')
legend('','errors', 'Location', 'northwest')
title('Regression Tree Model')
subtitle(strcat('RMSE = ', num2str(rmse_Power), ' W'))
hold off
saveas(gcf,strcat('Figures/Regression_Tree_Power.png') )
```



```
importance = zeros(mdl.KFold,size(mdl.X,2));
for ii = 1:mdl.KFold
    importance(ii,:) = predictorImportance(mdl.Trained{ii,1});
end
figure
bar(categorical(X.Properties.VariableNames), mean(importance,1))
title('Power model: Information gain of variables')
saveas(gcf,strcat('Figures/Regression_Tree_Power_varImportance.png') )
```

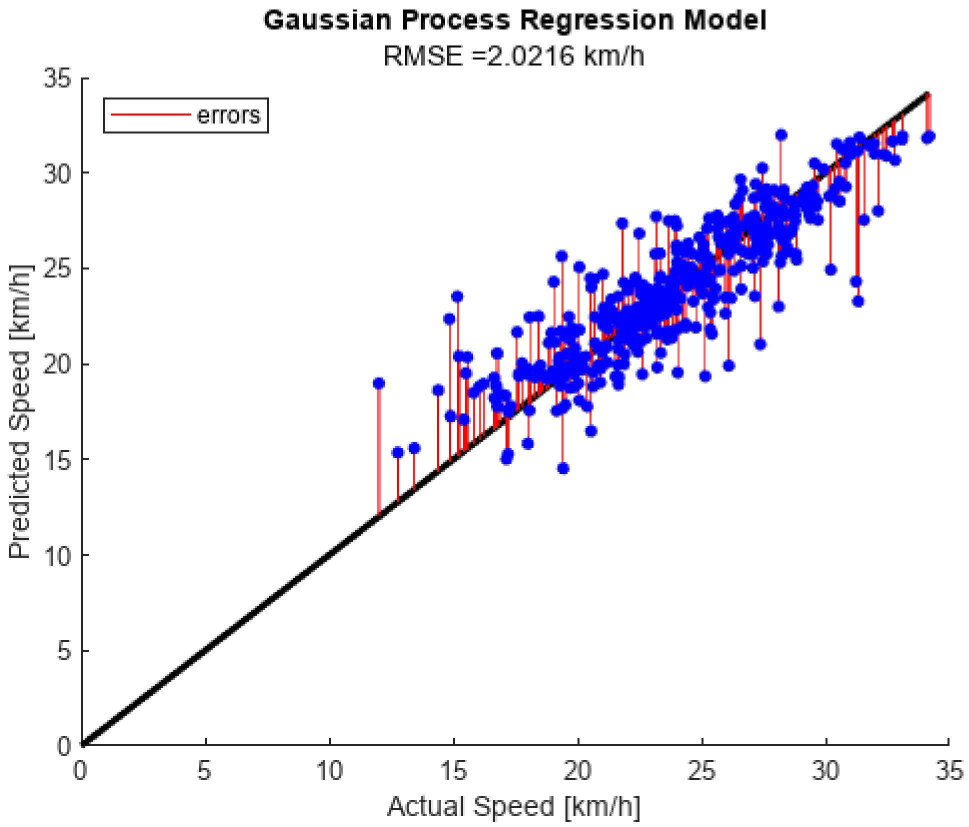


Gaussian Process Regression

```
mdl = fitrgp(X, ySpeed, "CategoricalPredictors", [1, size(X,2)], 'Kfold', n_cvpartitions, Stanc
yPredict = kfoldPredict(mdl);
rmse_Speed = sqrt(kfoldLoss(mdl))
```

```
rmse_Speed = 2.0216
```

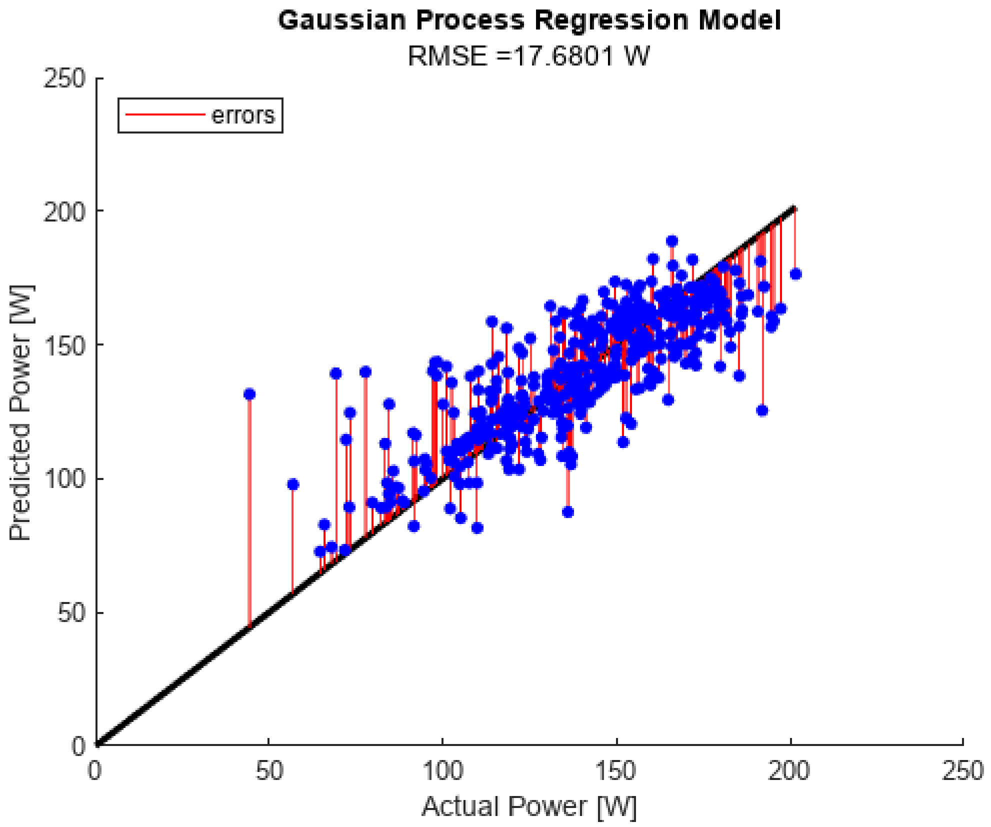
```

y_plot = table2array(ySpeed);
figure
hold on
plot([0, max(table2array(ySpeed))], [0, max(table2array(ySpeed))], 'k', LineWidth=2.5)
for ii = 1:size(y_plot)
    plot([y_plot(ii), y_plot(ii)], [y_plot(ii), yPredict(ii)], 'r-', LineWidth=0.15)
end
plot(y_plot, yPredict, 'b.', MarkerSize=16)
xlabel('Actual Speed [km/h]')
ylabel('Predicted Speed [km/h]')
legend('','errors', 'Location', 'northwest')
title('Gaussian Process Regression Model')
subtitle(strcat('RMSE = ', num2str(rmse_Speed), ' km/h'))
hold off
saveas(gcf,strcat('Figures/Regression_Gaussian_Speed.png') )
```



```
mdl = fitrgp(X(idx_pow,:), yPower(idx_pow,:), "CategoricalPredictors", [1, size(X,2)], 'Kfold'
yPredict = kfoldPredict(mdl);
rmse_Power = sqrt(kfoldLoss(mdl))
```

```
rmse_Power = 17.6801
```

```
y_plot = table2array(yPower(idx_pow,:));
figure
hold on
plot([0, max(table2array(yPower))], [0, max(table2array(yPower))], 'k', LineWidth=2.5)
for ii = 1:size(y_plot)
    plot([y_plot(ii), y_plot(ii)], [y_plot(ii), yPredict(ii)], 'r-', LineWidth=0.15)
end
plot(y_plot, yPredict, 'b.', MarkerSize=16)
xlabel('Actual Power [W]')
ylabel('Predicted Power [W]')
legend('','errors', 'Location', 'northwest')
title('Gaussian Process Regression Model')
subtitle(strcat('RMSE = ', num2str(rmse_Power), ' W'))
hold off
saveas(gcf,strcat('Figures/Regression_Gaussian_Power.png') )
```


Gaussian Process Regression Model
RMSE =17.6801 W
errors
Predicted Power [W]
250
200
150
100
50
0
0
50
100
150
200
250
Actual Power [W]