Vehicular accident risk analysis for Portland, OR

Ben Stabley

December 14, 2022

Abstract

Risk of vehicle accidents has been modeled using both statistical methods based on observations and probabilistic methods using road characteristics. Here I use a mixed method GIS-based approach to determine risk for streets in Portland, OR for both "normal" and "snow" conditions. A form of suitability analysis is performed using various publicly available GIS datasets. Generally, arterial streets exhibited higher risk scores than neighborhood streets under normal conditions, while the opposite was true of snow conditions. Furthermore, a spatial variation in conditional risk was observed between far east and far west Portland due to the street characteristics of those areas.

Contents

List of Figures

1 Introduction

In Portland, Oregon there are approximately 10,000 to 12,000 vehicle accidents each year [6]. Although many *human factors*, such as intoxication and distractions, contribute to vehicle accidents, this project will not examine human factors. Determining "high risk" sections of road has historically been done using both recorded accident incidents and probabilistic modeling based on physical road characteristics [5]. In this GIS-based approach, I have attempted to combine both methods to rate roads in Portland based on the likelihood that a vehicle will have an accident there. Specifically, I have tried to address these questions:

1. Can a "risk score" for vehicular accidents be assigned to roads in Portland? Where are some of the highest risk segments of roads?
2. How do "snow" conditions affect the assigned normal surface risk scores?

2 Method

2.1 Data

Datasets for this project are shown in Table 1, and most of them come from the City of Portland's "PortlandMaps Open Data" portal [3]. All the PortlandMaps data is in the Web Mercator (EPSG:3857) CRS, but metadata suggests that it was originally in one of the Oregon State Plane CRSs before uploading to the portal. I decided to use this CRS for the project since the bulk of my data was already in it. I gave each dataset a short name that I will use to refer to it throughout this document.

The `streets` layer was the skeleton onto which I attached the rest of the meaty data. It seemed to be a frequently updated and very complete set of centerlines for streets in the entire metro area, so I clipped it to just the city limits of `pdxbound`. Its `TYPE` attribute is a federal code to denote the general "size" of the road, from freeway to forest service road. I used this to remove freeways (e.g. I205) and ramps from the analysis. The geometry of this layer is a line for each *segment* of a road, usually beginning and ending at intersections, so that a single road such as Burnside Blvd is represented by hundreds of short segments. This was ideal for this analysis because I wanted to be able to assign a "risk" in a granular way.

The `speedlimit` layer has speed limits for each road segment. There were a few streets which did not have a speed limit (were `NULL`), such as fire lines in Forest Park or some roads at the airport. I mapped `NULL`$\rightarrow$25 mph.

The high crash network (hcn) is basically list of the top "most dangerous" streets in Portland based on incident statistics gathered by the Portland Dept of Transportation (PBOT) and Oregon Department of Transportation (ODOT). I simply used the `hcn` as binary data: a street is part of the hcn or not.

The dispatched calls for service from the Portland Police Bureau (PPB) is an Excel spreadsheet, and contains information about calls to 911, the police non-emergency number, and police-initiated calls. PPB makes these available for each year since 2012. I am using just the 2021 year because it was the most recent complete year. To remove unnecessary data and make it easier to use, I removed all rows from the spreadsheet where `FinalCallCategory` was not `collision` or `hazard`, then put each of those into their own feature classes by importing the spreadsheet using `OpenDataLon` and `OpenDataLat` for the coordinate. I assumed these were "GPS", and therefore WGS84 or NAD83. I assigned WGS84 because these points are "100-block level", so if they were in fact NAD83 the slight inaccuracy will not affect the analysis.

The traffic speed and volume counts has data spanning almost a decade. I used it all because I basically aggregated it during processing (details below). `PctOverPosted` describes the percent of observed traffic over the posted speed limit, and `ADVolume` describes the average daily volume (number) of vehicles observed at that sample site.

Snow and ice routes is a layer cataloging what PBOT does to prevent and remove snow and ice from roads during one of the rare winter storms. I used only the `Priority` attribute, which is a

Dataset	Short name	Source	Geometry	Attributes used
streets	`streets`	City of Portland [3]	line	shape, `TYPE`
high crash network	`hcn`	City of Portland [3]	line	shape
speed limits	`speedlimit`	City of Portland [3]	line	`SpeedLimit`
snow and ice routes	`snowice`	City of Portland [3]	line	`Priority`
traffic speed counts	`trafficspd`	City of Portland [3]	point	`PctOverPosted`
traffic volume counts	`trafficvol`	City of Portland [3]	point	`ADTVolume`
dispatched calls for service	`collisions`, `hazards`	Portland Police [7]	point	`OpenDataLon`, `OpenDataLat`
USGS_13_n46w123	`DEM`	USGS [9]	raster	values
city boundaries	`pdxbound`	City of Portland [3]	polygon	shape
water area	`water`	Census Bureau [8]	polygon	shape

Table 1: Data sources for the project.

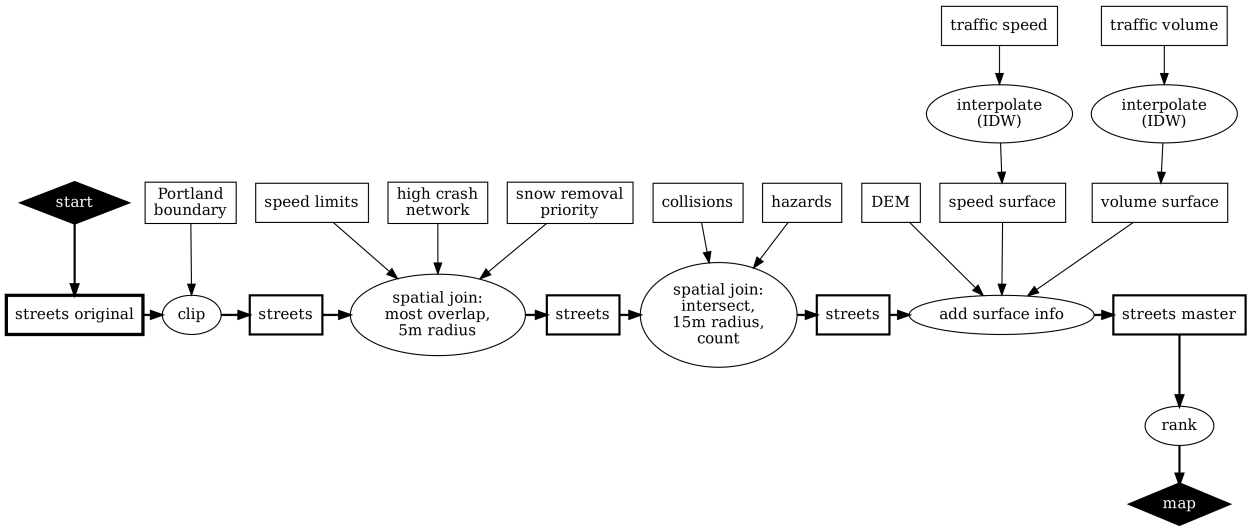

Figure 1: A flowchart of the overall analysis process.

letter grade where "A" is the highest priority and "D" is the lowest. In order to use these in ranking, I mapped A→1, B→2, C→3, D→4, and all other `NULL` values to 5.

The DEM obtained from the USGS National Map covers the standard $1^\circ \times 1^\circ$ "quad" for N 46° W 123°. This was convenient because a single DEM covered my entire study area. I decided to use the $\frac{1}{3}$ arcsec resolution because it seemed a good compromise. $\frac{1}{9}$ arcsec and 1 arcsec DEM were also available for the study area, and although 1 arcsec may have worked well I suspect $\frac{1}{9}$ arcsec might be a bit "noisy" for my purpose. I also clipped the DEM to pdxbound.

The Portland city boundary pdxbound and water layers were primarily used for an aesthetic and simple base map, though I think displaying the Willamette and Columbia rivers also adds important geographical reference for the viewers. pdxbound was also used to clip some other datasets to the study area.

2.2 Process

My essential process was to attach tributes from the various input layers described above to the `streets` skeleton. *Spatial join* was often the most useful way to do this, but I also used *add surface information* to apply values to `streets` from rasters such as `DEM`. A diagram of my general process is shown in Figure 1.

The layers `speedlimit`, `hcn`, and `snowice` are line features representing the centerlines of streets, very similarly to `streets`. Unfortunately, *none* of these feature classes share a common attribute on

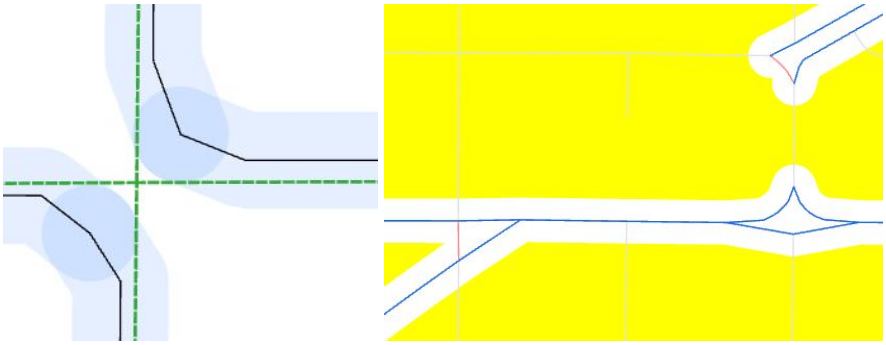

Figure 2: Left: example of a mismatch of `streets` (black) and `speedlimit` (dotted green). Right: example of negative buffer (yellow) used to select the side streets (light gray) adjacent to `hcn`.

which I could perform a standard join. Furthermore, these layers seem to have been created from older or different line features than `streets`, so the features do not coincide in some places where roads have been physically modified since the creation of these 3 layers and updated in `streets`. (See Figure 2 left.) This made a simpler spatial join fail in places. To deal with this, I used a spatial join with a 5m search radius where the join was on the feature with "most overlap." This did a fairly good job of joining my input layers to `streets` in a one-to-one fashion since those layers mostly coincided spatially with `streets`.

An unintended side effect of the 5m search radius was that features in `streets` that were adjacent to a join feature would gain attributes, even though they did not have an corresponding feature in the joining feature class, because they very slightly overlapped with the 5m pseudo-buffer. For example, Burnside Blvd is in `hcn`, but all the smaller side streets are not but they joined with Burnside Blvd. Fortunately, I was able to deal with that by creating a "negative buffer" and using it to select incorrectly joined features (Figure 2 right). For example, I created a 5m dissolved buffer on `hcn`, used that erase from `pdxbound`, then use the modified `pdxbound` to select intersecting features of `streets`. Once selected, I just assigned all those selected features, which were incorrectly assigned "in hcn," to "not in hcn." I did the same thing for `snowice`, but `speedlimit` did not require this step because it had every street (dense) while the other two did not (sparse).

The layers `collisions` and `hazards` were much easier to handle than those described above. I wanted to assign to each street segment the total number of incidents that occurred along the segment. I spatially joined each layer to `streets` using a slightly larger 15m buffer, joining any feature that intersected, and aggregating using "count" representing incidents per year.

Because the layers `trafficspd` and `trafficvol` are samples of traffic speed and volume and I want to be able to assign their attributes to all features in `streets`, I could not use the same method as `collisions` and `hazards`. Instead I decided to interpolate the samples and create new rasters with data in previously "empty" areas. I chose to use the Inverse Distance Weighted (IDW) interpolation method. I call the resultant raster layers a "surface" in Figure 1.

The `DEM` is already a "surface" and required no further processing beyond clipping to the `pdxbound` extents.

In order to apply information from `trafficspd`, `trafficvol`, and `DEM` "surfaces", I used the *Add Surface Information* tool. This reads a value from the input raster at each vertex of the input feature, and can perform a number of aggregation methods. For `trafficspd` and `trafficvol` I calculated only the mean value, that is, for example, "the mean percent traffic over the posted speed limit on this segment of road." For `DEM` I calculated both mean value (elevation) and mean slope.

These 9 variables are illustrated in the map in Figure 3. Once I had all the variables for my risk model for each segment in `streets`, I reclassifed each variable into an integer in $[1, 5]$. For most variables I simply used the class breakpoints determined using Jenk's Natural Breaks, except for `hcn`, `speedlimit`, `slope`, `elevation`, and `priority` (from `snowice`). For these, I manually assigned breakpoints, though they were essentially equivalent the Equal Interval method due to the natural classes formed by the data. The full details of the breakpoints and reclassification step is

shown in code listing 7.

Using the variables `hcn`, `speedlimit`, `collisions`, `hazards`, `trafficspd`, and `trafficvol`, I calculated the "normal risk" score. Each variable received an *importance rank* in $[1, 3]$, 1 being lowest importance and 3 being highest importance. The same ranking could be assigned to multiple variables. I would have liked to inform each variable's assigned importance rank with supporting research, but due to time constraints I did not, and simply assigned importance ranks using my personal intuition as a driver.

These importance ranks were used to calculate weighting in the final normal risk score. I used the following formula to calculate weights in the interval $(0, 1)$ for each i^{th} factor.

$$w_i = \frac{r_i}{\sum_{k=1}^{n} r_k} \tag{1}$$

Where w_i is the weight, r_i is the rank, and n is the total number of factors. This is numerically equivalent to one given by Bolstad [1]. Each variable's reclassified value in $[1, 5]$ was multipled by its weight in $(0, 1)$, and then these were summed to give the final normal risk score in $[1, 5]$. See code listing 7 for complete code details.

The same process was performed for the "snow risk score", which used the variables `slope`, `elevation`, `priority`, and the previously calculated normal risk score (`regularRisk`).

Finally, I calculated a "relative risk score" by subtracting snow risk from normal risk. I also normalized each risk score before subtraction.

$$relative = 5 \times \left(\frac{normal - min(normal)}{max(normal) - min(normal)} - \frac{snow - min(snow)}{max(snow) - min(snow)}\right) \tag{2}$$

This was useful to see the change in risk when road surface conditions changed. Roads where normal risk "dominated" would have positive relative risk scores, and roads where snow risk "dominated" would have negative relative risk scores. Roads with little difference in risk would score near zero.

3 Results

Maps illustrating the final results for normal risk, snow risk, and relative risk are shown in Figure 4, Figure 5, and Figure 6.

General trends I see in these maps is that larger arterial streets have more normal risk than neighborhood streets. I think this makes logical sense because arterial streets often have higher speeds and more overall traffic volume, and therefore one would expect the chance of an accident to increase. Furthermore, I also could see a slightly higher prevalence of high normal risk roads on the east side of Portland, specifically east of SE 82$^{\text{nd}}$ Ave. My personal speculation about this, based on living near this area, is that there is a higher proportion of arterial streets vs. neighborhood streets. Recent PBOT road projects to address the safety of east Portland streets seem to confirm my speculation [2].

In the snow risk scores, I notice that neighborhood streets appear to have more or similar risk compared to arterial streets. One explanation is that PBOT plows and deices the major roads during a snow storm, but often leaves neighborhood streets unplowed [4]. Also, streets in hilly areas, such as west and southeast Portland and near Mt Tabor) exhibited higher snow risk, but this was expected due to the use and weighing of `slope` in the model.

The relative risk map highlights the above trends very well.

4 Conclusion

I was able to use available data to calculate a risk score for roads in Portland, and was able to also calculate a snow risk score. I was also able to quantify the change between normal risk and snow

risk using the calculated relative risk score.

Although work like this has already been done by PBOT and ODOT when the high crash network data was compiled, the HCN is not very "granular" in that an entire street is "dangerous" or not without taking into consideration that some parts of a road have different characteristics. Perhaps a more granular model could help planners prioritize safety projects differently.

5 Future Work

I think more work could be done refining the model used in this analysis, particularly examining what data and variables are used and their weights in the model. For example, by using the HCN and police dispatches for collisions, I wondered if perhaps I am "double counting" because the HCN is partially based on observed collisions and that collisions themselves are are a direct measurement of the risk of vehicular accidents that I attempted to quantify. Also, I think a literature review on which to base variable inclusion and weighting is necessary if the process and model described here are to be anything more than an exercise in GIS analysis.

I also think there is potential to add additional factors to the model. I had considered including variables such as road width, number of lanes, one-way vs two-way direction of travel, and roadside hazards, but did not due to lack of data and lack of time. Other factors to include may be the complexity of intersections and road curvature. A temporal analysis may also produce interesting insights.

References

[1] Paul Bolstad. *GIS Fundamentals. A First Text on Geographic Information Systems*. 6th ed. XanEdu, 2019. ISBN: 978-1-59399-552-2.

[2] City of Portland. *News release: Transportation projects in East Portland will deliver safer routes and better employment connections for thousands of Portlanders.* Aug. 8, 2022. URL: https://www.portland.gov/transportation/news/2022/8/8/news-release-transportation-projects-east-portland-will-deliver-safer (visited on 12/12/2022).

[3] City of Portland. *PortlandMaps Open Data*. 2022. URL: https://gis-pdx.opendata.arcgis.com/ (visited on 10/30/2022).

[4] City of Portland. *Winter Weather Basics and FAQ*. URL: https://www.portland.gov/transportation/weather/winter-weather-basics-and-faq (visited on 12/12/2022).

[5] Ezra Hauer. "Identification of Sites with Promise". In: *Transportation Research Record* 1542 (Jan. 1996), pp. 54–60. DOI: 10.3141/1542-09.

[6] Portland Dept. of Transportation. *How Crash Data Works*. Jan. 1, 2022. URL: https://www.portland.gov/transportation/vision-zero/crash-data (visited on 10/30/2022).

[7] Portland Police Bureau. *Police Dispatched Calls Dashboard*. 2022. URL: https://www.portland.gov/police/open-data/police-dispatched-calls (visited on 10/30/2022).

[8] US Bureau of the Census. *TIGER/Line Shapefile*. 2022. URL: https://www.census.gov/geographies/mapping-files/time-series/geo/tiger-line-file.2020.html (visited on 10/30/2022).

[9] US Geological Survey. *The National Map Viewer*. 2022. URL: https://apps.nationalmap.gov/viewer/ (visited on 10/30/2022).

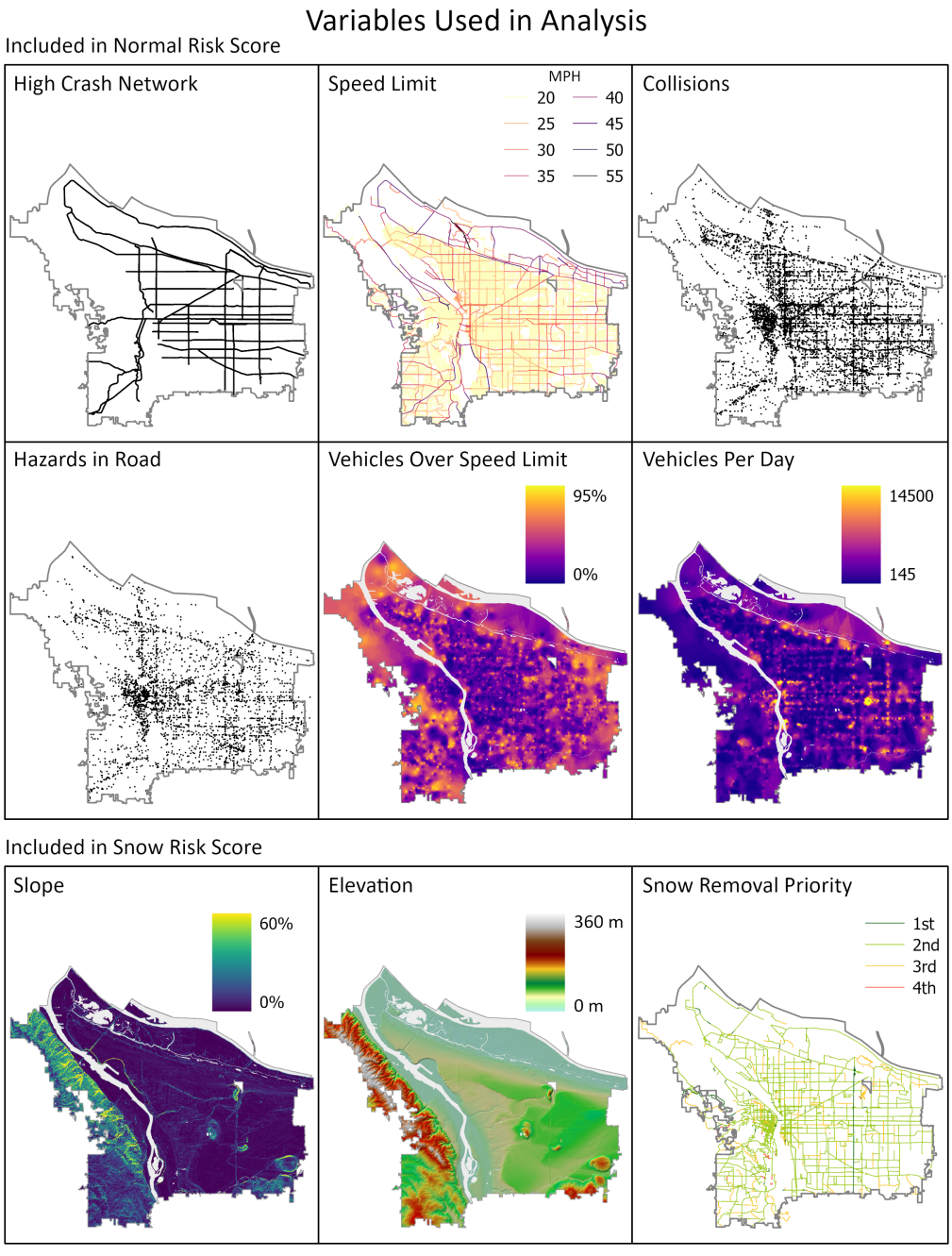

Figure 3: Variables used. Full size

Normal Conditions Risk
of Vehicular Accidents, Portland, OR

A "normal risk score" for general non-snowy conditions was calculated for every non-freeway street in Portland. Scores ranged from 1 to 5. Six factors were analyzed for each segment of road: if it was or was not part of the designated "high crash network", the posted speed limit, the number of reported collisions in 2021, the number of reported hazards-in-road in 2021, the average percent of vehicles exceeding the speed limit, and the average daily traffic volume.

Each of these factors were classified into intermediate values, multiplied by an importance weighting, and summed. The weights were as follows, where 3 is most important:

high crash network: 3	reported hazards: 1
posted speed limit: 1	percent speeders: 3
reported collisions: 3	traffic volume: 2

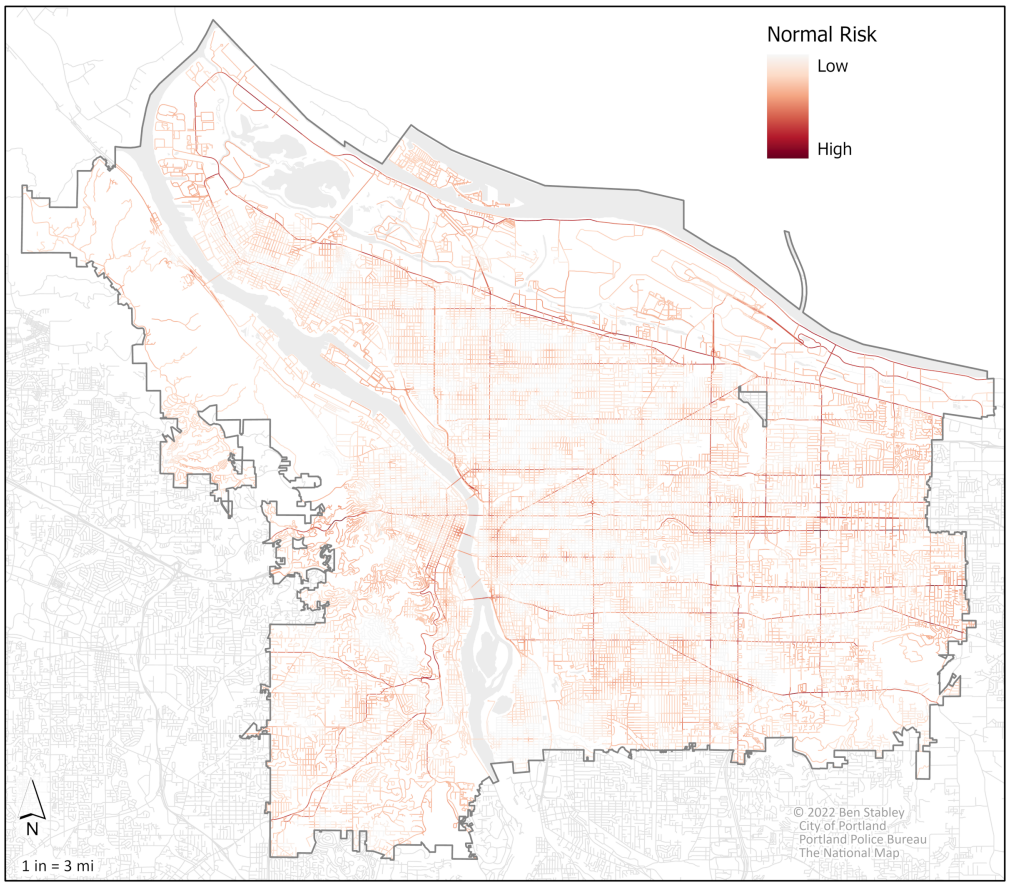

Figure 4: Normal risk map. Full size

Snowy Conditions Risk
of Vehicular Accidents, Portland, OR

An additional "snow risk score" ranging from 1 to 5 was calculated for every non-freeway street in Portland. The normal risk score and 3 additional factors were included in the analysis for each segment of road: the average slope, the average elevation, and the priority for snow removal. As with the normal risk score, each of these factors were classified into intermediate values, multiplied by a weight, and summed. The weights were as follows, where 3 is the most important:

normal risk score: 2
average slope: 3
average elevation: 1
snow removal priority: 2

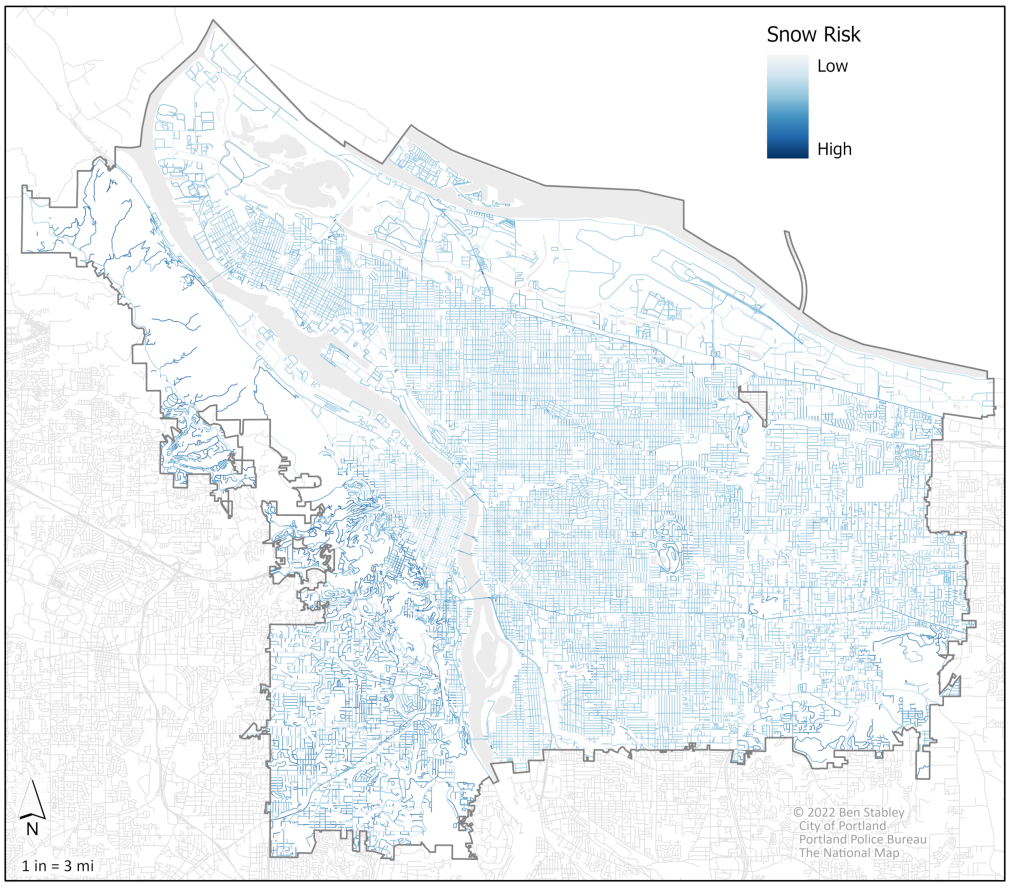

Figure 5: Snow risk map. Full size

Relative Conditional Risk
of Vehicular Accidents, Portland, OR

Here, the snow risk score was subtracted from the normal risk score. This relative conditional risk score emphasizes streets where the risk score changes greatly depending on the condition of the road. Extreme positive or negative scores stand out in red or blue respectively, while streets where the risk changes very little between normal and snowy conditions score near zero, shown in white.

Of interest are two areas on the west and east sides of Portland. The western area is characterized by mostly neighborhood streets on hilly terrain with low snow removal priority. The eastern area has more higher speed roads which are also higher snow removal priority and the terrain is mostly flat. Accordingly, the west side generally has more accident risk in snow, and the east side has more risk in normal conditions.

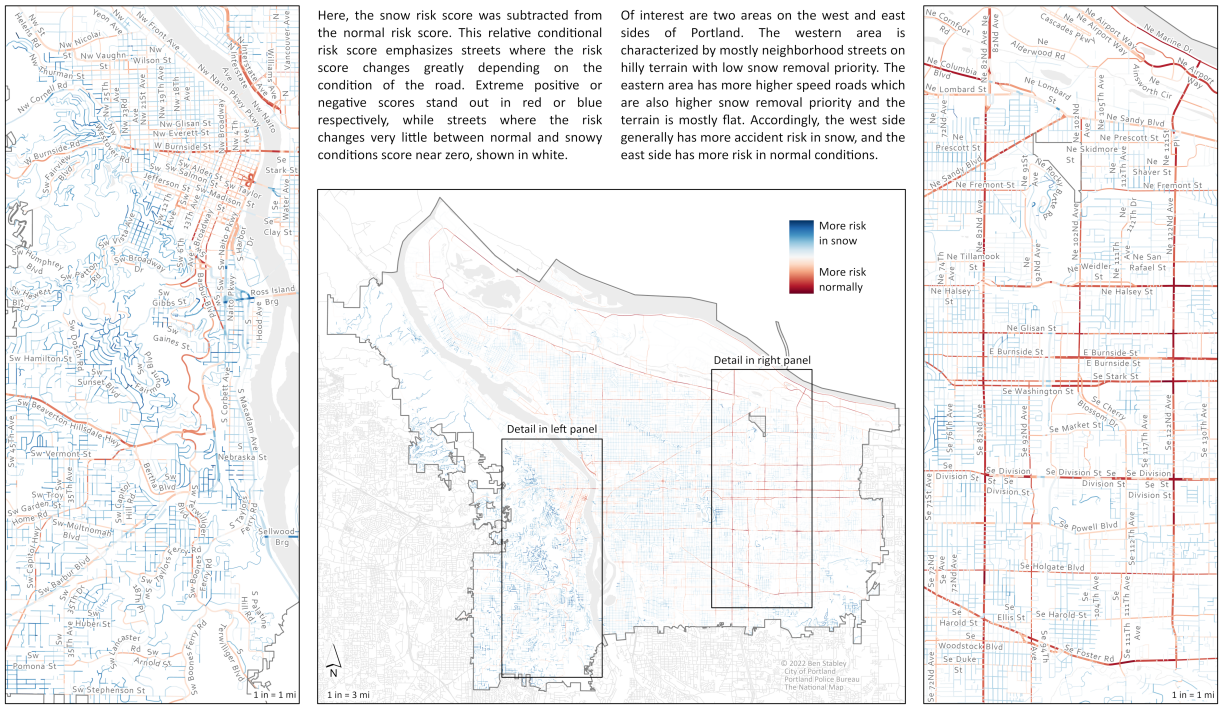

Figure 6: Relative risk map. Full size

Figure 7: The code used to perform reclassification and ranking calculations for the normal and snow risk scores.

```
class Ranker:
    """stores weight, classes, and null info for a field, and
    performs rank classification for a field's value."""
    def __init__(this, weight: int, classes: dict, NULL=None):
        this.weight = weight
        this.classes = classes
        this.NULL = NULL

    def rank(this, x):
        """compare x to each classes's max value, and
        return the rank of the first matching class."""
        if x is None:
            x = this.NULL
        for value, rank in this.classes.items():
            if x <= value:
                return rank
        return None


def classifier(fields: dict):
    """
        Performs calculation.
        run with:
            regular:
        classifier({"hcn": !hcn!, "speedlimit": !SpeedLimitNum!, "collision": !NumCollision!, "hazard":
             !NumHazard!, "pctSpeeding": !PctSpeedMean!, "trafficVol": !TrafficVolMean!})
            snow/winter:
        classifier({"regularRisk": !RiskScore!, "slope": !Avg_Slope!, "elevation": !ElevMean!, "priority":
             !PriorityNum!})
    """

    # weights: 1=low, 3=high
    # classes: key is max value for that class. must be ordered low->high.
    # NULL:    value with which to replace <null> (None) values
    ranker = {
        # regular criteria
        "hcn":          Ranker(weight=3, classes={0: 1, 1: 5}),
        "speedlimit":   Ranker(weight=1, classes={20: 1, 30: 2, 40: 3, 50: 4, 60: 5}, NULL=25),
        "collision":    Ranker(weight=3, classes={2: 1, 7: 2, 14: 3, 24: 4, 47: 5}, NULL=0),
        "hazard":       Ranker(weight=1, classes={1: 1, 4: 2, 7: 3, 13: 4, 18: 5}, NULL=0),
        "pctSpeeding":  Ranker(weight=3, classes={14: 1, 26: 2, 40: 3, 58: 4, 100: 5}),
        "trafficVol":   Ranker(weight=2, classes={1387: 1, 3076: 2, 5444: 3, 9090: 4, 26611: 5}),
        # winter/snow criteria
        "regularRisk":  Ranker(weight=2, classes={1: 1, 2: 2, 3: 3, 4: 4, 5: 5}),
        "slope":        Ranker(weight=3, classes={3: 1, 6: 2, 12: 3, 20: 4, 50: 5}),
        "elevation":    Ranker(weight=1, classes={100: 1, 200: 2, 300: 3, 350: 4, 400: 5}),
        "priority":     Ranker(weight=2, classes={1: 1, 2: 2, 3: 3, 4: 4, 5: 5})
    }

    # sum only weights of used fields
    sumweights = sum([ranker[f].weight for f in fields])
    # sum weighted rank of each field
    return sum([ranker[name].rank(value) * (ranker[name].weight / sumweights) for (name, value) in
         fields.items()])
```