UNIVERSIDADE DA CORUÑA

Universidade Vigo

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Análisis de las líneas de autobuses urbanos de A Coruña

Trabajo Fin de Máster

Máster en Técnicas Estadísticas

Curso 2014-2015

Autor: Juliana Díaz López

Director: Rubén Fernández Casal

Análisis de las líneas de autobuses urbanos de A Coruña

Juliana Díaz López

ÍNDICE

1. INTRODUCCIÓN

El presente trabajo de fin de máster es de carácter aplicado y fue tutelado en el Ayuntamiento de A Coruña que se ha mostrado interesado en realizar un análisis estadístico sobre la base de datos de las tarjetas Millennium.

Es importante aclarar que en este trabajo siempre que hablemos de una parada física de autobús, la llamaremos "stop", y reservaremos la palabra "parada" para cuando un autobús para en un stop.

El objetivo inicial de las prácticas era valorar el impacto de las obras en el desplazamiento por transporte público. No fue posible valorar ese impacto porque hasta la presente fecha el ayuntamiento no nos ha cedido las informaciones sobre las obras, cambios de rutas antes y después de las obras, es decir sabemos cuales son las líneas actuales con sus stops pero no sabemos cuales eran las líneas y sus stops antes de empezar las obras.

Debido a este contratiempo, se ha cambiado el objetivo del trabajo, que pasó a ser un análisis exploratorio descriptivo de las bases de datos de los autobuses urbanos de A Coruña.

Otro objetivo era representar en un mapa la densidad demográfica de Coruña (según la dirección de los usuarios de las Tarjetas Millennium), y sobre ese mapa representar la red de autobuses de A Coruña coloreando cada tramo entre dos stops por la cantidad de pasajeros, con el objetivo de identificar si hay zonas que necesitan más autobuses o mas líneas.

Tampoco fue posible hacer ese mapa con la densidad demográfica de A Coruña, porque aun que en el formulario de solicitud de la tarjeta Millennium contenga el campo "dirección", ese dato no está almacenado en la base de datos de las tarjetas. Según los funcionarios del ayuntamiento, al tener el nombre, apellidos y DNI de los usuarios, cuando les hace falta la dirección, acceden a eso por a base de datos de padrón. Hemos solicitado acceso a esa base de datos, pero hasta la presente fecha no nos han concedido.

El ayuntamiento también ha mostrado interés en saber la frecuencia de usuarios por tramo entre dos stops, lo que tampoco fue posible debido a que el usuario utiliza la tarjeta para pagar cuando entra en el autobús (eso no permite saber en que stop ha subido), pero no tenemos como saber donde se ha bajado. Se intentó estimar ese número de usuarios verificando las tarjetas que han utilizado la misma línea dos veces en un mismo día (una vez en la ida y otra en la vuelta), pero se decidió no continuar porque los stops de ida y de vuelta no coinciden, y además muchos usuarios que tienen la opción de utilizar más de una línea para llegar a su destino final.

Por lo tanto hemos seguido con el objetivo de hacer un análisis exploratorio descriptivo de la base de datos de las tarjetas Millennium. También hicimos un análisis descriptivo de esos datos desde el punto de vista de redes y al final hemos representado datos de stops y variables de tramos como por ejemplo la velocidad media.

Para eso el ayuntamiento nos ha concedido el acceso a otras dos bases de datos: tranvías, y datos espaciales.

1.1. UN POCO DE HISTORIA

El 1 de enero de 1903 comienza la historia de la Compañía de Tranvías de La Coruña, se inaugura por todo lo alto, la primera línea de tranvía de mulas que cubría el trayecto de Puerta Real a la Estación de ferrocarriles, con ramales a Riazor y cocheras.

En 1907 una compañía belga se hace con el control de la empresa, pero su gestión no durará mucho. En 1909 la compañía se declara en quiebra y la Compañía de Tranvías recupera el control perdido.

En 1913 se aprueba la electrificación del tranvía y el servicio mejora notablemente. Se alcanzan los 20km/hora y comienza a planificarse el tranvía interurbano entre A Coruña y Sada, que será una realidad en 1922.

Son años de continuas mejoras tecnológicas y ampliaciones de líneas, pero al mismo tiempo son tiempos conflictivos con numerosas protestas obreras y manifestaciones.

En 1936 surgió el trolebús como solución a los constantes problemas. A pesar de que en 1947 se registra el record histórico de viajeros en tranvía, en 1948 comienza a circular el primer trolebús cubriendo la línea entre la Plaza de Pontevedra y Monelos.

Trolebuses y tranvías convivieron juntos hasta principios de los sesenta. En 1961, 18 nuevos trolebuses de dos pisos se incorporan a la flota de la Compañía y desplazan definitivamente a los tranvías.

Los trolebuses dominan el paisaje Coruñés, pero por poco tiempo. En 1965 la Compañía de Tranvías obtiene la primera concesión de autobuses y surge la Línea A que cubrirá el trayecto entre la Plaza de Pontevedra y el Puente del Pasaje.

Los recién llegados autobuses comienzan a sustituir a los trolebuses. En 1978 Autobuses Urbanos de La Coruña y la Compañía de Tranvías se fusionan en una única empresa. Un año más tarde los trolebuses dejan de circular después de 31 años de servicio.

En la década de los 80 comienza a notarse un descenso de los viajeros provocado por el aumento de vehículos privados. En 1989, para paliar los efectos la bajada de usuarios, se llevan a cabo una serie de reformas y reestructuraciones de líneas.

En 1997, culmina una idea que se llevaba años barajando, la recuperación del tranvía para uso turístico. Poco a poco irá ampliando la línea hasta completar gran parte el paseo marítimo.

El centenario de la compañía en el año 2003 trae consigo un nuevo sistema de control por satélite, una mejora tecnológica que supone un gran avance para los usuarios. Un año después se presentan las paradas inteligentes y la Compañía recibe el premio a la mejor empresa de transporte público de España.

En 2013 Tranvías cumple 110 años con una gran flota de autobuses dotados con las últimas tecnologías.

Actualmente la página web de tranvías http://www.tranviascoruna.com/, cuenta con servicios muy interesantes para el usuario, como por ejemplo:

- Líneas y horarios
- Cuando llega mi bus
- Búsqueda de paradas más cercanas al sitio donde está el usuario
- Planificación de ruta integrada con Google Maps

La empresa cuenta también con el aplicación para el móvil itranvias, que es una aplicación oficial gratis para los móviles, que le da al usuario información en tiempo real de líneas, como por ejemplo la predicción de tiempos de llegada del autobús en determinada parada.

Otra tecnología disponible desde junio de 2011 es el pago con tarjetas Tarjeta Coruña Millennium, permiten pagar en los autobuses urbanos de la Compañía de Tranvías de A Coruña sin necesidad de disponer de dinero en metálico. Eso quiere decir que se gana en agilidad y sencillez.

1.1.1. Tarjeta Coruña Millennium

La Tarjeta Coruña Millennium sirve tanto para pagar un viaje en autobús como para pagar la O.R.A., o utilizar los servicios de las bibliotecas municipales. Según el ayuntamiento, en breve, también permitirá realizar trámites administrativos desde el ordenador, reservar instalaciones municipales y sacar entradas para conciertos.

Existen varias modalidades de Tarjeta Millennium:

- **Tarjeta Millennium General**
  Tiene derecho a la tarjeta Millennium General cualquier persona que este empadronada o no en el Ayuntamiento de A Coruña.

- **Tarjeta Bus Millennium Jubilados / Pensionista**
  Tiene derecho a la tarjeta de jubilado la persona que sea mayor de 65 años. Y pensionista quien tiene ingresos que no superen en más del 30% el Salario Mínimo Interprofesional por miembro de la unidad familiar. Y que a la vez sea menor de 65 años. Además, en ambos casos hay que estar empadronado en el Ayuntamiento de A Coruña con una antigüedad mínima de seis meses.

- **Tarjeta Bus Millennium Desempleados**
  Tiene derecho a la tarjeta de desempleados quien cumple los siguientes requisitos:

- Estar empadronado en el Ayuntamiento de A Coruña con una antigüedad mínima de seis meses.
- Estar en situación de desempleo con una antigüedad mínima de tres meses desde la última fecha de alta.

- **Tarjeta Bus Millennium para Discapacitados**
  Tiene derecho a la tarjeta de discapacitado cumple los siguientes requisitos:
  - Estar empadronado en el Ayuntamiento de A Coruña con una antigüedad mínima de seis meses.
  - Tener ingresos que no superen en más del 30% el Salario Mínimo Interprofesional por miembro de la unidad familiar.
  - Tener reconocida por el órgano competente una minusvalía igual o superior al 33%.

- **Tarjeta Bus Millennium Escolar**
  Tiene derecho a la tarjeta escolar quien cumple los siguientes requisitos:
  - Tener entre 3 y 25 años de edad.
  - Tener una renta anual inferior a 40.000 € brutos por unidad familiar, siendo incrementada esta cantidad en 5.000 € por cada hijo a mayores del solicitante.
  - En los supuestos de separación, divorcio, parejas de hecho o parejas que no tengan la consideración de parejas de hecho la renta anual bruta por unidad familiar no podrá superar 24.000 €, incrementada en 3.000 € por hijo a mayores.
  - Estar matriculado en centros de estudios homologados por el Ministerio competente en materia de educación del Estado o por la Conserjería competente en materia de educación de la Xunta de Galicia.
  - Que los estudios cursados tengan 15 o más horas lectivas a la semana, con la excepción de los estudiantes de la Universidad a Distancia (UNED) a los que este último requisito no les es exigible (para la justificación de este punto se presentará un certificado del centro en que se cursen los estudios).

- **Tarjeta Bus Millennium Universitaria**
  Tiene derecho a la tarjeta Universitaria la persona que cumple los requisitos exigidos en cada caso por la Universidad de A Coruña.

Este tipo de pago facilita la fluidez del transporte ya que el tiempo empleado en el pago con este tipo de tarjetas es bastante menor que el que se emplearía si se pagase en metálico. Para incentivar ese tipo de pago las personas que pagan el autobús con ese tipo de tarjeta disfrutan de importantes descuentos en el precio del billete.

Tipo de Tarifa	Precio
Tarifa ordinaria	1,30 €
Bono Bus general	0,85 €
Escolar	0,33 €
Universitario	0,30 €
Desempleado	0,33 €
Jubilado/Pensionista	0,33 €
Discapacitado	0,33 €

Todas la tarjetas cuentan con transbordo gratuito permitido entre líneas (máximo 45 minutos).

¿Cómo funciona?

Primeramente se hace la solicitud de la tarjeta al ayuntamiento, luego hay que cargar la tarjeta con dinero eso se puede hacer en cajeros automáticos o en las oficinal Abanca en ventanilla.

Cuando un usuario sube al autobús, aproxima la tarjeta al lector esperando a que el lector le avise con un pitido de que la transacción se ha realizado.

1.2. BASES DE DATOS

Para realizar ese trabajo el ayuntamiento de A Coruña nos ha permitido acceder a tres bases de datos:

Base de datos Tranvías
Contiene todos los datos sobre el uso de los autobuses, conductores, liquidaciones y operaciones de la empresa de tranvías de Coruña de los últimos años.

Base de datos Tarjeta Coruña Millennium
Contiene todos los datos sobre titulares, solicitudes, movimientos y usos de las de tarjetas Coruña Millennium.

Base de datos GTSF (datos espaciales)
Contiene, entre otras cosas, información geográfica para generar mapas de las rutas de autobuses con sus paradas (stops).

A continuación vamos a describir cada una de las tablas de nuestras 3 bases de datos.

1.2.1. BASE DE DATOS TRANVÍAS

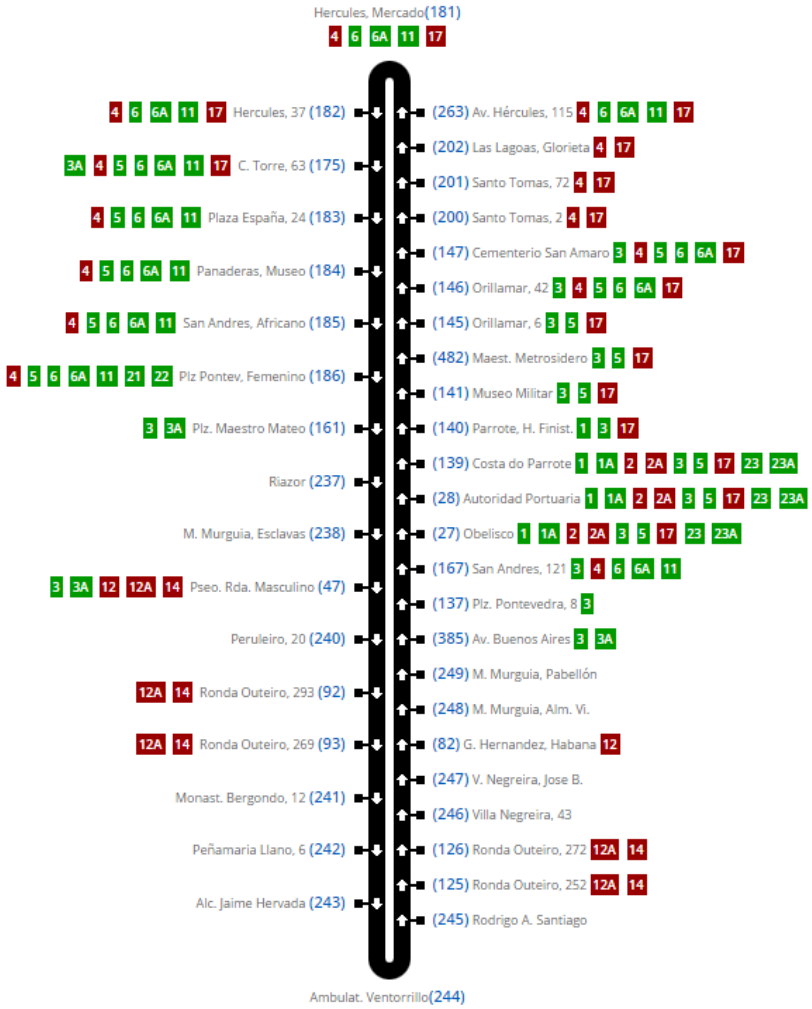
Antes de presentar la base de datos de Tranvías vamos a aclarar el concepto de tres términos utilizados por la compañía de Tranvías.

Viajero/Billete
Cada persona que utiliza un autobús se identifica con un billete/viajero. Cada persona se corresponde con un billete.

Viaje
Un viaje viene definido por conductor, autocar, número de línea, sentido del viaje, servicio y máquina registradora en una fecha dada.

Servicio
Un servicio es la agrupación de viajes con igual número de línea, sentido en una fecha dada.

E**jemplo:**
Para aclara mejor los conceptos descritos, tomaremos como ejemplo la línea 7: Hércules – Ventorrillo.



Cada vez que un autobús sale de Hércules y llega a Ventorrillo, o sale de Ventorrillo y llega a Hércules, eso se considera un viaje.

Lo viajes tiene un sentido, Hércules – Ventorrillo es un viaje de ida, Ventorrillo Hércules es un viaje de vuelta.

Actualmente, para un día laborable, la línea 7 por ejemplo, tiene la siguiente tabla de horarios:

Hora IDA:

Laborais

07:13	07:30	07:47	08:00	08:14	08:28	08:42	08:56
09:10	09:24	09:38	09:52	10:06	10:20	10:34	10:48
11:02	11:16	11:30	11:44	11:58	12:12	12:26	12:40
12:54	13:08	13:22	13:36	13:50	14:04	14:18	14:32
14:46	15:00	15:14	15:28	15:42	15:56	16:10	16:24
16:38	16:52	17:06	17:20	17:34	17:48	18:02	18:16
18:30	18:44	18:58	19:12	19:26	19:40	19:54	20:08
20:22	20:36	20:50	21:04	21:18	21:32	21:46	22:00
22:12	22:24	22:36	23:05	23:25			

Hora VOLTA:

Laborais

06:50	07:05	07:20	07:35	07:45	07:56	08:06	08:18
08:32	08:46	09:00	09:14	09:28	09:42	09:56	10:10
10:24	10:38	10:52	11:06	11:20	11:34	11:48	12:02
12:16	12:30	12:44	12:58	13:12	13:26	13:40	13:54
14:08	14:22	14:36	14:50	15:04	15:18	15:32	15:46
16:00	16:14	16:28	16:42	16:56	17:10	17:24	17:38
17:52	18:06	18:20	18:34	18:48	19:02	19:16	19:30
19:44	19:58	20:12	20:26	20:40	20:54	21:08	21:22
21:36	21:50	22:02	22:18	22:38	23:00		

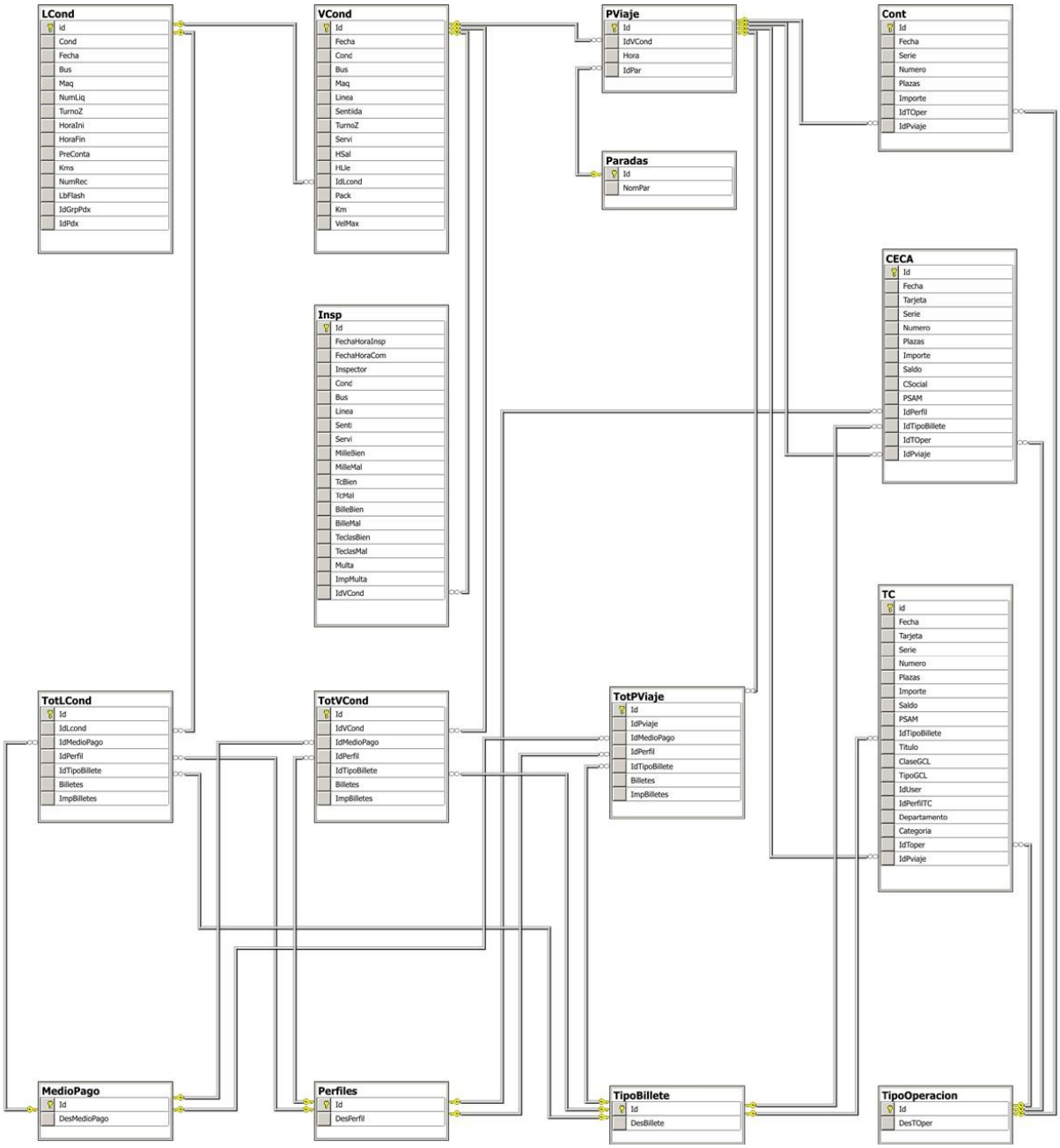
Por lo tanto esa línea en un día laborable hace 69 viajes de ida y 70 viajes de vuelta.

Esos 69 viajes de ida de la línea 7, el lunes, forman un servicio y esos 70 viajes de vuelta de la línea 7, el lunes, forman otro servicio. Y de igual manera con los demás días de la semana. Por lo tanto la línea 7 tiene 2 servicios al día, 14 servicios a la semana.

Aclarados esos conceptos, pasamos a la presentación de la base de datos de Tranvías que está instalada bajo Microsoft SQL Server 2005 con el nombre “Tranvías” y está compuesta de un total de 15 tablas.
Las tablas se dividen en 4 tipos.
- Tablas relativas a Liquidaciones
- Tablas relativas a Viajes.
- Tablas relativas a Operaciones (consumos, cargas, …)
- Tablas relativas a Tipo de Datos (las cuales sólo se usan para consulta de índices).

Se describen solamente las variables de las tablas que consideramos de interés.

Diagrama de Datos



TABLAS RELATIVAS A LIQUIDACIONES

LCond (Liquidación de Conductor)
Cada registro determina el trabajo continuo de un conductor en un determinado autobús y expendedora.

TotLCond (Totales por Liquidación de Conductor)
Detalle de la tabla *LCond*. Grupo de datos que hacen relación a las cantidades de operaciones que se han realizado en una determinada Liquidación de Conductor (*LCond*), indicando el número total de billetes y el importe total de los mismos dependiendo de la forma de pago, perfil y tipo de billete.

TABLAS RELATIVAS A VIAJES

VCond (Viajes de Conductores)
En la tabla VCond se almacenan todos y cada uno de los viajes realizados por los autobuses de la compañía de Tranvías de La Coruña.

Id	Cada índice almacena un viaje
Fecha	Fecha del viaje
Linea	Número de la línea
Senttida	1 para ida y 0 para vuelta
HSal	Hora de salida del servicio
HLle	Hola de llegada del servicio

TotVCond (Totales relacionados con Viajes de Conductores)
En esta tabla se reflejan los totales de operaciones de consumo relacionados con los viajes de los conductores, indicando el número total de billetes y el importe total de los mismos dependiendo de la forma de pago, perfil, y tipo de billete.

PViajes (Paradas)
Conjunto de datos en los que se relaciona el código de la parada con la tabla VCond, pudiendo obtener de esta forma la hora de paso del autobús en una determinada parada y los movimientos realizados en esta.

Id	Cada índice almacena una parada de un viaje dado
IdVCond	Índice correspondiente a la tabla VCond
Hora	Hora de llegada a la parada
IdPar	Código de Parada relacionado con la tabla Paradas

TotPViajes (Paradas)
Mediante esta tabla se obtienen el número total de billetes e importes de los registros de la tabla PViaje dependiendo de la forma de pago, perfil, y tipo de billete usado en la misma.

IdPviaje	Índice correspondiente a la tabla PViajes
IdMedioPago	Índice correspondiente a la tabla MedioPago
IdTipoBillete	Índice correspondiente a la tabla TipoBillete
Billetes	Número de billetes

Insp (Inspecciones)
Grupo de datos relacionados con las inspecciones que se llevan a cabo por los Inspectores.

TABLAS RELATIVAS A OPERACIONES DE CONSUMO

CECA (Pagos solo con tarjetas Millennium).
Esta tabla contiene datos, acerca de los consumos que se realizan con las tarjetas CECA (actuales Tarjetas Millenium y antiguas tarjetas con contactos bancarias), relacionando cada operación que se realiza con el perfil, tipo de billete y operación correspondiente.

Fecha	Fecha y hora de uso de la tarjeta
Tarjeta	Número de la tarjeta que se ha utilizado
IdPerfil	Código relacionado con la tabla Perfiles
IdTipoBillete	Código relacionado con la tabla TipoBillete
IdPviaje	Código relacionado con la tabla PViaje

TC (Operaciones llevadas a cabo con tarjeta de Tranvías de Coruña).
Esta tabla contiene datos, acerca de los consumos que se realizan con las tarjetas TC (tarjetas propietarias de la empresa Tranvías de La Coruña), relacionando cada operación que se realiza con el perfil, tipo de billete y operación correspondiente.

CONT (Operaciones realizadas al Contado)
En esta distribución se presentan los registros de aquellas operaciones cuya forma de pago se realiza en efectivo. Esta tabla en la actualidad, no está disponible.

RECCECA (Operaciones de Recargas en Tarjetas CECA)
Aquí se relacionan las operaciones de Recargas en las Tarjetas CECA con el número de tarjeta.

TABLAS CON ETIQUETAS DESCRIPTIVAS

TipoBillete (Tipo Billete)
Grupo de datos en los que se relaciona el Id de un billete determinado con la descripción del mismo.
1 – Normal
2 – Transbordo
3 – Defecto
4 – Bonificado
5 – Transbordo Xunta

TipoOperacion (Tipo de Operación)
Grupo de datos en los que se relaciona el Id de una operación con la descripción de la misma.

Perfiles (Perfiles)
Grupo de datos en los que se relaciona el Id de un determinado perfil con la descripción del mismo.
1 – General
2 – Escolar
3 – Universitario
4 – Desempleado
5 – Jubilado
6 – Minusválido
7 – INEF
9 – Pensionista
254 – Coruña Card Personal
255 – Coruña Card Familiar

MedioPago (Medio de Pago)
Grupo de datos en los que se relaciona el Id de un determinado medio de pago con la descripción del mismo.

1 – Contado
2 – Bono
3 – TC
4 – TC
5 – CECA
6 – Multas
7 – CECA Metropolitana

Paradas (Paradas)
Grupo de datos en los que se relaciona el Id de una parada, con la descripción de la misma.

Id	Código de las paradas
NomPar	Nombre de la parada
Largo	Descripción de la parada
Informativa	1 si la parada es informativa y 0 si la parada no es informativa
GPSX	Coordenada X indicando la ubicación de la parada
GPSY	Coordenada Y indicando la ubicación de la parada

Buses (Autobuses)
Relación de los autobuses que prestan el servicio.

Id	índice de la tabla
PlatBaja	1 si el autobús es de plataforma baja y 0 si el autobús no es de plataforma baja
Minusvalido	1 si el autobús está preparado para minusválidos y 0 si el autobús no está preparado para minusválidos
Articulado	1 si el autobús es articulado y 0 si el autobús no es articulado

Líneas (Líneas)
Grupo de datos en los que se relacionan las líneas.

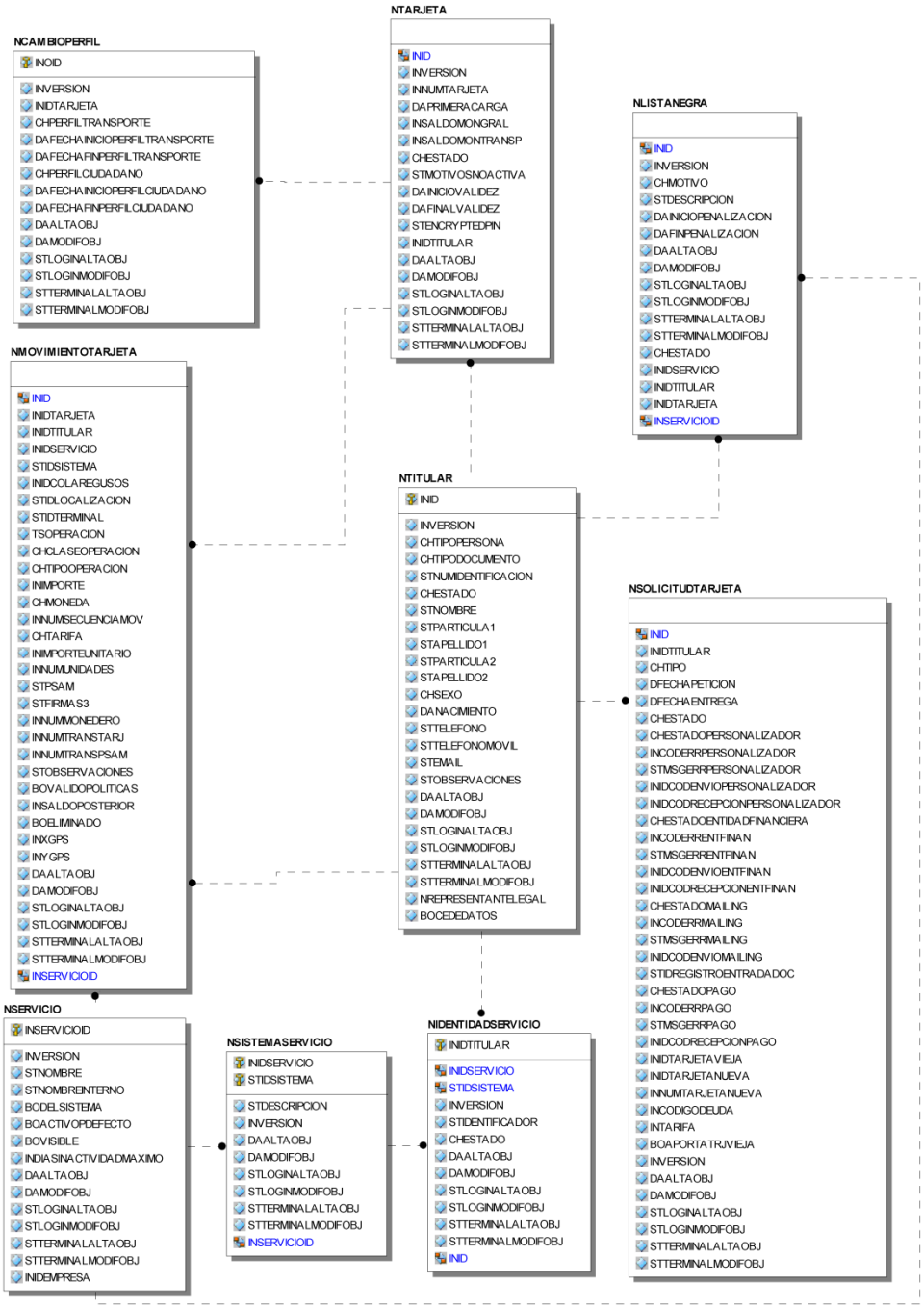
Id	Índice de tabla (no es único)
Nombre	Descripción de la línea
LineaComercial	Código de línea conocido por los usuarios

1.2.2. Base de Datos Millennium

Pasamos a la presentación de la base de datos de las Tarjetas Coruña Millennium que está compuesta de un total de 8 tablas.

Se describen solamente las variables de las tablas que consideramos de interés.

Diagrama de Datos



NCAMBIOPERFIL

En esta tabla se registran cada uno de los cambios de perfil que se realizan sobre las tarjetas.

NIDENTIDADSERVICIO

Identidades de cada titular en los distintos sistemas de cada servicio (NSERVICIO). Por ejemplo, Titular X en el sistema GESAURO dentro del servicio DEPORTES tiene el identificador YYYYYY.

NLISTANEGRA

En esta tabla se consignan las sanciones a las tarjetas o a titulares derivadas de un mal uso de la tarjeta ciudadana.

NMOVIMIENTOTARJETA

En esta tabla se registran todos los movimientos realizados con las tarjetas ciudadanas por los titulares.

NSERVICIO
Concepto equiparable a "servicio municipal" o a "organismo dentro de cada servicio municipal" según se defina y que se utiliza para relacionar los usos de las tarjetas, políticas de fidelización, etc.

NSOLICITUDTARJETA
Almacena las solicitudes de tarjetas realizadas por titulares.

NTARJETA
Almacena los datos de las tarjetas de cada titular.

NTITULAR
Almacena los datos de titulares de tarjetas ciudadanas. Es la única de las tablas descritas anteriormente que contiene datos sobre los titulares de las tarjetas.

Id	Índice de la tabla
CHTIPODOCUMENTO	Tipo documento (NIF, CIF, ...)
STNUMIDENTIFICACION	El número único del documento acreditativo
CHESTADO	Alta, Baja, ...
STNOMBRE	Nombre de pila
STAPELLIDO1	Primer apellido
STAPELLIDO2	Segundo apellido
CHSEXO	Género (Hombre, Mujer, Desconocido)
DANACIMIENTO	Fecha Nacimiento
STTELEFONO	Teléfono
STEMAIL	Email

1.2.3. DATOS GTSF

Nuestra base de datos espaciales está en formato GTSF, por lo tanto comenzaremos explicando que son datos en formato GTSF.

Las agencias de transporte público son expertas en tender redes de transporte extensas y complejas, pero es posible que estas empresas tengan menos experiencia en cuanto a crear aplicaciones informáticas. Los programadores de software son expertos en crear este tipo de aplicaciones, pero es posible que estos programadores tengan menos experiencia sobre transporte público.

Para aprovechar las habilidades de ambos grupos se han creado formatos comunes para intercambiar información de transporte, como por ejemplo el GTSF (General Transit Feed Specification). Este formato permite a las empresas de transporte público publicar sus datos y a los programadores crear aplicaciones que consuman esos datos de manera interoperable.

La especificación general de bases de datos de transporte público, conocidas como GTFS, define un formato común para los horarios de transporte público y la información geográfica asociada a ellos.

Una base de datos GTFS se compone de una serie de archivos de texto normalmente recopilados en un archivo ZIP. Cada archivo modela un aspecto particular de información de tránsito: paradas, rutas, viajes y otros datos relacionados con los horarios.

Vamos a explicar los tipos de archivos que comprende un GTFS y vamos a describir los campos usados en todos esos archivos, en donde los 6 primeros archivos son obligatorios y los demás son opcionales.

Nombre archivo	Descripción
agency.txt	Una o varias empresas de transporte público que proporcionan los datos.
stops.txt	Ubicaciones concretas en donde los vehículos recogieron o dejaron pasajeros.
routes.txt	Rutas de transporte público. Una ruta es un grupo de viajes que se muestran a los usuarios como servicio independiente.
trips.txt	Viajes para cada ruta. Un viaje es una secuencia de dos o más paradas que se produce en una hora específica.
stop_times.txt	Horarios a los que un vehículo llega a una parada concreta y sale de ella en cada viaje.
calendar.txt	Fechas de los ID de servicio a través de un horario semanal. Se especifica cuando comienza y finaliza un servicio, al igual que los días de la semana en que el servicio está disponible.
calendar_dates.txt	Excepciones de los ID de servicio definidas en el archivo calendar.txt. Si el archivo calendar_dates.txt incluye TODAS las fechas de servicio, se puede especificar este archivo en lugar de calendar.txt.
fare_attributes.txt	Información sobre tarifas correspondientes a las rutas de una organización de transporte público.
fare_rules.txt	Reglas de aplicación de la información sobre tarifas correspondientes a las rutas de una organización de transporte público.
shapes.txt	Reglas para el trazado de las líneas en un mapa que representen las rutas de una organización de transporte público.
frequencies.txt	Tiempo entre viajes para las rutas cuya frecuencia de servicio es variable.

transfers.txt	Reglas para establecer conexiones en los puntos de transbordo entre rutas.
feed_info.txt	Información adicional sobre la base de datos en sí, incluida la información sobre el editor, la versión y el vencimiento.

Pasamos a la presentación de nuestra base de datos de bus urbano (formato GTSF) que está compuesta de un total de 8 tablas.

En ese trabajo utilizaremos solamente ciertas variables de determinadas tablas de las descritas anteriormente, debido a que las tablas que contienen información sobre fechas y horas contienen un corto periodo de tiempo, una vez que el objetivo esa base de datos es solamente generar la información necesaria para generar los planos que necesitamos. La información sobre fechas y horas de los viajes y paradas vamos a obtener de la base de datos tranvías.

Se describen solamente las variables de las tablas que consideramos de interés.

AGENCY.TXT
Información sobre la empresa de transporte público que proporciona los datos.

CALENDAR.TXT
Fechas de los ID de servicio a través de un horario semanal. Se especifica cuando comienza y finaliza un servicio, al igual que los días de la semana en los que el servicio está disponible.

CALENDAR_DATES.TXT
Excepciones de los ID de servicio definidas en el archivo calendar.txt. Si el archivo calendar_dates.txt incluye TODAS las fechas de servicio.

ROUTES.TXT
Información sobre las rutas.

route_id	ID que identifica una ruta de forma exclusiva
agency_id	Identifica una empresa para la ruta especificada
route_short_name	Nombre comercial de la ruta
route_long_name	Nombre completo de una ruta
route_type	Tipo de transporte público utilizado en una ruta (en este caso el código será 3 pues únicamente vamos a estudiar autobuses y es el código con el que se corresponden)

TRIPS.TXT
Información sobre los viajes para cada ruta.

route_id	ID que identifica una ruta de forma exclusiva
service_id	Identifica de forma exclusiva un conjunto de fechas en el que el servicio se encuentra disponible en una o más rutas
trip_id	ID que identifica un viaje de forma exclusiva
trip_headsign	Identifica el destino del viaje para los pasajeros
direction_id	Indica la dirección de un viaje: 1 para ida y 0 para vuelta
shape_id	Permite definir la forma en que se debe trazar una línea en el mapa para representar un viaje.

SHAPES.TXT
Reglas para el trazado de las líneas en un mapa que representen las rutas.

shape_id	ID que identifica exclusivamente a una forma
shape_pt_lat	Asocia la latitud de un punto de una forma con un ID de forma
shape_pt_lon	Asocia la longitud de un punto de una forma con un ID de forma
shape_pt_sequence	Asocia la latitud y la longitud de un punto de una forma al orden secuencial que tienen a lo largo de la forma

STOPS.TXT
Ubicación de las paradas.

stop_id	ID que identifica de forma exclusiva a una parada
stop_code	Identifica de forma exclusiva la parada
stop_name	El nombre de una parada
stop_desc	Descripción de una parada
stop_lat	Latitud de una parada
stop_lon	Longitud de una parada

STOP_TIMES.TXT
Horarios a los que un vehículo llega a una parada concreta y sale de ella en cada viaje.

1.3. ESTRUCTURA

En este trabajo el acceso y la manipulación de los datos fue igual o más laboriosa que el análisis estadístico posterior porque hubo que dedicar muchas horas a aprender a utilizar distintas herramientas para poder acceder, ordenar y seleccionar los datos. Por lo tanto dividiremos el trabajo en dos partes.

En primer lugar nos dedicaremos a presentar las distintas herramientas que hemos utilizado, y de paso haremos pequeños resúmenes de cómo utilizar algunas de ellas.

A continuación en la segunda parte haremos un análisis descriptivo clásico de los datos, un análisis exploratorio descriptivo considerando que esos datos forman una red y por último vamos a representar datos de stops y variables de tramos como por ejemplo la velocidad media.

Con las bases de datos presentadas estaríamos ante un gran volumen de datos, por ello nos vamos a centrar solamente en un mes del año, quedando con aproximadamente 20Mb (comprimidos en un archivo .RData). Hemos escogido el mes de Febrero de 2014 para el estudio pues podemos considerarlo un mes "normal". Con este término queremos hacer referencia a que es un mes con no demasiadas festividades como podrían ser los meses de Diciembre y Enero por las navidades o los meses de Julio y Agosto debido a las vacaciones.

Vamos a preparar los datos cruzando las tablas de las bases de datos de Tranvías y Tarjetas Millennium, donde nos quedamos solamente con algunas variables de interés en el mes de Febrero de 2014.

2. HERRAMIENTAS

En ese capítulo vamos a presentar todas las herramientas que usaremos para desarrollar este trabajo.

2.1. SQL

Tenemos 2 bases de datos externas y muy grandes:

- Base de datos Tranvías (servidor SQL), que tiene 15 tablas.
- Base de datos Tarjetas Millennium (servidor Oracle), que tiene 8 tablas distintas.

En primer lugar hubo que obtener los permisos de acceso de lectura y la configuración de dichas bases de datos. Una vez obtenidos, se buscó una manera de acceder a las bases de datos anteriores para poder explorarlas y por último encontrar la manera de cargar las tablas de interés en R, programa que utilizaremos en nuestro análisis.

2.1.1. CONEXIÓN A GESTORES DE BASES DE DATOS

Los gestores de base de datos son programas que permiten crear y administrar ficheros accediendo a ellos de forma cómoda, ordinariamente a través de variantes dialectales de SQL.

Por lo tanto comenzaremos viendo que es SQL y como se utiliza.

2.1.2. LENGUAJE DE CONSULTA ESTRUCTURADO

El lenguaje de consulta estructurado o SQL (por sus siglas en inglés **Structured Query Language**) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas. Una de sus características es el manejo del álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar de forma sencilla información de interés de bases de datos, así como hacer cambios en ella.

El SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales y permite así gran variedad de operaciones.

Componentes del SQL

El lenguaje SQL está compuesto por comandos, cláusulas, operadores y funciones de agregado. Estos elementos se combinan en las instrucciones para crear, actualizar y manipular las bases de datos.

Comandos

- **DLL (Data Definition Language)** que permiten crear y definir nuevas bases de datos, campos e índices.
- **DML (Data Manipulation Language)** que permiten generar consultas para ordenar, filtrar y extraer datos de la base de datos.
- **DCL (Data Control Language)** que se encargan de definir las permisos sobre los datos

En ese trabajo nos vamos a centrar en los comandos DML, porque la base de datos Tranvías utilizada por el ayuntamiento está en SQL y necesitamos generar consultas para ordenar, filtrar y extraer datos de la base de datos.
Las consultas de selección se utilizan para indicar al motor de datos que devuelva información de las bases de datos, esta información es devuelta en forma de conjunto de registros.

Haremos un pequeño resumen con las principales sentencias de consultas en SQL.

Consultas Básicas

La sintaxis básica de una consulta de selección es la siguiente:

```
SELECT Campos FROM Tabla;
```

En donde campos es la lista de campos que se deseen recuperar y tabla es el origen de los mismos, por ejemplo:

```
SELECT Fecha, Tarjeta FROM CECA;
```

Ordenar los registros

Adicionalmente se puede especificar el orden en que se desean recuperar los registros de las tablas mediante la cláusula ORDER BY Lista de Campos. En donde Lista de campos representa los campos a ordenar. Ejemplo:

```
SELECT IdPerfil, Fecha, Tarjeta FROM CECA ORDER BY Fecha;
```

Esta consulta devuelve los campos IdPerfil, Fecha, Tarjeta de la tabla CECA ordenados por el campo Fecha.

*Se pueden ordenar los registros por más de un campo, como por ejemplo:

```
SELECT IdPerfil, Fecha, Tarjeta FROM CECA ORDER BY Fecha, Tarjeta;
```

*Incluso se puede especificar el orden de los registros: ascendente mediante la cláusula (ASC -se toma este valor por defecto) ó descendente (DESC).

```
SELECT IdPerfil, Fecha, Tarjeta FROM CECA ORDER BY Fecha ASC, Tarjeta DESC;
```

Consultas con predicado

El predicado se incluye entre la cláusula y el primer nombre del campo a recuperar, los posibles predicados son:

Predicado	Descripción
ALL	Devuelve todos los campos de la tabla.
TOP	Devuelve un determinado número de registros de la tabla.
DISTINCT	Omite los registros cuyos campos seleccionados coincidan totalmente.
DISTINCTROW	Omite los registros duplicados basándose en la totalidad del registro y no sólo en los campos seleccionados.

- **ALL**

Si no se incluye ninguno de los predicados se asume ALL. No es conveniente utilizar de este predicado porque tarda, es mucho más rápido indicar el listado de campos deseados.

```
SELECT ALL FROM CECA;

SELECT * FROM CECA;
```

- **TOP**

Devuelve un cierto número de registros que entran entre el principio o el final de un rango especificado por una cláusula ORDER BY. Supongamos que queremos recuperar los nombres de los 25 primeros estudiantes del curso 1994:

```
SELECT TOP 25 IdPViaje, MedioPago FROM TotPViaje ORDER BY Billetes DESC;
```

Si no se incluye la cláusula ORDER BY, la consulta devolverá un conjunto arbitrario de 25 registros de la tabla TotPViaje .El predicado TOP no elige entre valores iguales. En el ejemplo anterior, si el número medio de billetes del número 25 y la 26 son iguales, la consulta devolverá 26 registros.

Se puede utilizar la palabra reservada PERCENT para devolver un cierto porcentaje de registros que caen al principio o al final de un rango especificado por la cláusula ORDER BY. Supongamos que en lugar de los 25 primeros que más pagan con tarjetas deseamos el 10 por ciento que más paga con tarjetas:

```
SELECT TOP 10 PERCENT IdPViaje, MedioPago FROM TotPViaje ORDER BY Billetes DESC;
```

- **DISTINCT**

Omite los registros que contienen datos duplicados en los campos seleccionados. Para que los valores de cada campo listado en la instrucción SELECT se incluyan en la consulta deben ser únicos.

Por ejemplo, si existen varios registros de pagos con la misma tarjeta en la tabla CECA, la siguiente instrucción SQL devuelve un único registro para cada tarjeta:

```
SELECT DISTINCT Tarjeta FROM CECA;
```

Con otras palabras el predicado DISTINCT devuelve aquellos registros cuyos campos indicados en la cláusula SELECT posean un contenido diferente. El resultado de una consulta que utiliza DISTINCT no es actualizable y no refleja los cambios subsiguientes realizados por otros usuarios.

- **DISTINCTROW**

Devuelve los registros diferentes de una tabla; a diferencia del predicado anterior que sólo se fijaba en el contenido de los campos seleccionados, éste lo hace en el contenido del registro completo independientemente de los campos indicados en la cláusula SELECT.

```
SELECT DISTINCTROW Tarjeta FROM CECA;
```

Criterios de selección

Se presentarán las posibilidades de filtrar los registros con el fin de recuperar solamente aquellos que cumplan condiciones preestablecidas.

Hay que recalcar dos detalles de gran importancia. El primero de ellos es que cada vez que se desee establecer una condición referida a un campo de texto la condición de búsqueda debe ir entre comillas simples; y la segunda hace referencia a las fechas. Las fechas se deben escribir siempre en formato mm-dd-aa en donde mm representa el mes, dd el día y aa el año, hay que prestar atención a los separadores -no sirve la separación habitual de la barra (/), hay que utilizar el guión (-) y además la fecha debe ir entre almohadillas (#). Por ejemplo si deseamos referirnos al día 3 de Septiembre de 1995 deberemos hacerlo de la siguiente forma: #09-03-95# ó #9-3-95#.

Operadores Lógicos

Los operadores lógicos poseen la siguiente sintaxis:

<expresión1> operador <expresión2>

En donde expresión1 y expresión2 son las condiciones a evaluar.

Para ese ejemplo, supongamos que tenemos una tabla Empleados con las variables: edad, sueldo, estado, provincia.

```
SELECT * FROM Empleados WHERE (Edad > 25 AND Edad < 50) OR Sueldo = 100;

SELECT * FROM Empleados WHERE NOT Estado = 'Soltero';

SELECT * FROM Empleados WHERE (Sueldo > 100 AND Sueldo < 500) OR Provincia =
'Madrid' AND Estado = 'Casado');
```

Intervalos de Valores

Para indicar que deseamos recuperar los registros según el intervalo de valores de un campo emplearemos el operador Between cuya sintaxis es:

campo [Not] Between valor1 And valor2

(la condición Not es opcional)

En este caso la consulta devolvería los registros que contengan en "campo" un valor incluido en el intervalo valor1, valor2 (ambos inclusive).
Si anteponemos la condición Not devolverá aquellos valores no incluidos en el intervalo.

Para este ejemplo, vamos a imaginar que tenemos una tabla Pedidos con las variables: código postal, estado, provincia, número del pedido.

```
SELECT * FROM Pedidos WHERE CodPostal Between 28000 And 28999;
```

Devuelve todas las variables de la tabla pedidos, realizados entre los códigos postales 28000 y 28999.
Otra manera de hacer la consulta sería:

```
SELECT IIf (CodPostal Between 28000 And 28999, 'Provincial', 'Nacional') FROM
Pedidos;
```

Esta sentencia devuelve de la tabla Pedidos el valor 'Provincial' si el código se encuentra en el intervalo y 'Nacional' en el caso contrario.

El operador IN

Este operador devuelve aquellos registros cuyo campo indicado coincide con alguno de los de una lista. Su sintaxis es:

expresión [Not] In(valor1, valor2, . . .)

```
SELECT * FROM Pedidos WHERE Provincia In ('Madrid', 'Barcelona', 'Sevilla');
```

La cláusula WHERE

La cláusula WHERE puede usarse para determinar qué registros de las tablas enumeradas en la cláusula FROM aparecerán en los resultados de la instrucción SELECT. Después de escribir esta cláusula se deben especificar las condiciones expuestas anteriormente. Si no se emplea esta cláusula, la consulta devolverá todas las filas de la tabla. WHERE es opcional, pero cuando aparece debe ir a continuación de FROM. Ejemplos:

```
SELECT Apellidos, Salario FROM Empleados WHERE Salario > 21000;

SELECT * FROM Pedidos WHERE Fecha_Envio = #5/10/94#;
```

```
SELECT Apellidos, Nombre FROM Empleados WHERE Apellidos = 'King';

SELECT Apellidos, Nombre FROM Empleados WHERE Apellidos Like 'S*';

SELECT Apellidos, Salario FROM Empleados WHERE Salario Between 200 And 300;

SELECT Apellidos, Salario FROM Empleados WHERE Apellidos Between 'Lon' And
'Tol';

SELECT Id_Pedido, Fecha_Pedido FROM Pedidos WHERE Fecha_Pedido Between #1-1-94#
And  #30-6-94#;

SELECT Apellidos, Nombre, Ciudad FROM Empleados WHERE Ciudad In ('Sevilla', 'Los
Angeles', 'Barcelona');
```

Consultas de unión internas

La sentencia SQL JOIN se utiliza para relacionar varias tablas. Nos permitirá obtener un listado de los campos que tienen coincidencias en ambas tablas. Su sintaxis es:

```
SELECT campos FROM tabla1 JOIN tabla2 ON tabla1.campo1=tabla2.campo2
```

Donde:

- campos, son las variables que necesitamos.
- tabla1, tabla2, son los nombres de las tablas que contienen los registros.
- campo1, campo2, son los nombres de las variables que tienen que ser iguales en las dos tablas. Si no son numéricos, los campos deben ser del mismo tipo y contener el mismo tipo de datos, pero no tienen porque tener el mismo nombre.

Por ejemplo, si tenemos una tabla clientes y una tabla acciones, y queremos un listado de las variables nombre, teléfono y cantidad de la tabla clientes, en donde el valor de la variable id (de la tabla cliente) sea igual al valor de la variable rtb (de la tabla acciones):

```
SELECT nombre, telefono, cantidad FROM clientes JOIN acciones ON
cliente.id=acciones.rtb
```

2.2. R

R **es un entorno y un lenguaje** para el análisis estadístico de datos. R es un dialecto del lenguaje S, desarrollado en los Laboratorios Bell por John Chambers et al., por lo que también es conocido como ”GNU S”. R es un Software libre bajo las condiciones de licencia GPL de GNU y está implementado en los sistemas operativos más populares como MacOS X, Linux, Ubuntu y Windows, permite al usuario crear y modificar nuevas funciones y incluye numerosos complementos (paquetes) para aplicaciones estadísticas concretas.

2.2.1. PAQUETE RODBC

Hay varios paquetes que permiten conectar R a gestores de bases de datos, en la misma o en otra máquina diferente: son RPgSQL (para PostgreSQL), ROracle (para Oracle), RMySQL (para MySQL) y RODBC (para cualquier origen de datos ODBC), entre otros.

Como ya hemos comentado, tenemos 2 bases de datos externas (una en SQL y otra en Oracle), por lo tanto vamos a utilizar el paquete RODBC. Ese paquete emplea como lenguaje de interrogación SQL y tiene facilidad para poblar una dataframe realizando una consulta a una base de datos externa, permite tratar ficheros muy grandes de los que sólo se importan las observaciones/variables que interesan. Fue escrito originalmente por Michael Lapsley (Escuela de Medicina de St. George, Universidad de Londres) en 1999, pero desapareció en 2002, y fue rescatado por Brian Ripley en enero de 2003.

Ahora que ya sabemos cómo funcionan las consultas SQL, veamos como utilizar el paquete RODBC.

Establecimiento de una conexión

Para establecer el canal de comunicación entre R y una Base de Datos se utiliza la función “odbcConnect”, como se muestra a continuación:

```
library(RODBC)

ch <- odbcConnect( “dsn_de_la_base_de_datos")
```

Para facilitar la comprensión utilizaremos en los ejemplos nuestra bases de datos Tranvías que tiene como DNS: sqlTranvias32.

Vamos a crear un canal de comunicación “tranvias”, que iremos utilizar a lo largo de este apartado en los ejemplos.

```
library(RODBC)

tranvias <- odbcConnect("sqlTranvias32")
```

Cuando terminemos de hacer las consultas necesarias debemos cerrar este canal utilizando la función “close”:

```
close(tranvias)
```

Lectura de una base de datos

Una vez que hemos establecido un canal de comunicación entre R y el servidor podemos leer desde R esa base de datos.

Para ver que tablas son accesibles desde nuestra conexión, vamos a utilizar

```
sqltables (tranvias)
```

Puede que aparezcan muchas tablas, incluso tablas que no tenemos permiso para acceder porque son tablas internas que no son tablas de consulta de las bases de datos. Podríamos restringir el alcance de la búsqueda de tablas

```
sqlTables(ch, tableType = "TABLE")

sqlTables(ch, schema = "some pattern")

sqlTables(ch, tableName = "some pattern")
```

Como ya tenemos una descripción de la base de datos y ya sabemos el nombre de las tablas, no nos hace falta hacer esa búsqueda.

Lectura de una tabla

Para consultar las tablas de las bases de datos el paquete RODBC utiliza el lenguaje de consulta SQL "adaptado", con adaptado hacemos referencia a que utiliza la misma lógica que hemos visto en las consultas sql pero combinadas con funciones de R.

Dada la tabla dbo.Buses de la Base de Datos esta puede ser leída desde R como un data frame con la función "sqlQuery" como se ilustra a continuación:

```
buses <- sqlQuery(tranvias, "select * from dbo.Buses")
```

Es muy importante recordar que nuestras bases de datos son muy grandes y por ese motivo habrá tablas que no podremos descargar completas en R porque no tenemos memoria suficiente para eso. En dichos casos optaremos por restringir la consulta a las variables de interés o realizar la consulta bajo determinadas condiciones (filtros).

Crear un dataframe con las variables de interés de una tabla

En la base de datos tranvias queremos las variables Minusvalidos y Articulado de la tabla dbo.Buses

```
minus <- sqlQuery(tranvias, paste("select Minusvalido, Articulado from
dbo.Buses"))
```

Crear un dataframe con las variables de interés de una tabla, con una condición

En la base de datos tranvias queremos las variables Id y Minusvalidos de la tabla dbo.Buses, donde la variable Platbaja sea = 1.

```
minus <- sqlQuery(tranvias, paste("select Id, Minusvalido from dbo.Buses",
"where Platbaja = 1"))
```

Si no quisiéramos solo esas dos variables y quisiéramos todas las variables de esa tabla

```
minus <- sqlQuery(tranvias, paste("select *  from dbo.Buses","where Platbaja =
1"))
```

Crear un dataframe con las variables de interés de una tabla, con dos o más condiciones

Para utilizar dos o más condiciones tenemos que utilizar "and" entre ellas. Por ejemplo, en la base de datos tranvías queremos las variables Id y Minusvalidos de la tabla dbo.Buses, donde la variable Platbaja sea = 1 y Articulado sea =1.

```
minus <- sqlQuery(tranvias, paste("select Id,Minusvalido from dbo.Buses","where
Platbaja = 1 and Articulado = 1"))
```

Filtrar por fechas

Cuando la condición de la consulta es una fecha hay que tener cuidado con la sintaxis, por ejemplo, en la base de datos tranvías queremos las variables Tarjeta y Importe de la tabla dbo.CECA, en el mes de febrero de 2014. O sea la variable Fecha tiene que estar entre 01/02/2014 y 28/02/2014.

```
ceca <- sqlQuery(tranvias, paste("select Tarjeta, Importe from dbo.CECA", "where
Fecha between '20140201' and '20140228'"))
```

Ordenar consulta

Si quisiéramos la misma consulta que hicimos anteriormente pero ordenada por fecha, solamente tendríamos que añadir "order by".

```
ceca <- sqlQuery(tranvias, paste("select Tarjeta, Importe from dbo.CECA", "where
Fecha between '20140201' and '20140228'", "order by Fecha"))
```

Crear un dataframe con todas las variables de DOS tablas distintas

En algunas situaciones nos podría interesar consultar datos de 2 tablas distintas de la base de datos a la vez. Por ejemplo, en la base de datos tranvias quieremos todas las variables de la tabla dbo.Buses y dbo.Insp

```
buses.insp <- sqlQuery(tranvias,("select * from dbo.Buses, dbo.Insp"))
```

Crear una dataframe con las variables de interés de DOS tablas distintas

En otras situaciones nos podría interesar consultar solamente algunas variables de 2 tablas distintas de la base de datos a la vez. Por ejemplo en la base de datos tranvías queremos consultar las tablas dbo.VCond y dbo.PViaje. Pero nos interesan solamente algunas variables de cada tabla:

dbo.VCond: Fecha, Bus

dbo.PViaje: Hora

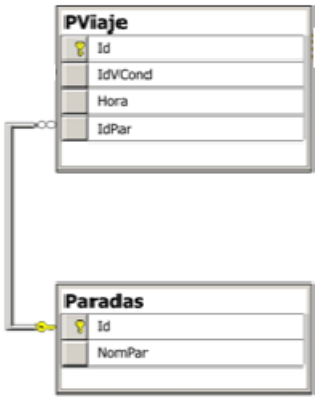
```
bus <- sqlQuery(tranvias, paste("select Fecha, Bus, Hora from dbo.VCond,
dbo.PViaje"))
```

La sintaxis anterior solo funciona porque esas tres variables que hemos elegido no se repiten en las dos tablas. Si quisiéramos por ejemplo agregar a la variable Id de la tabla dbo.VCond, tendríamos que indicar en que tabla esta dicha variable, porque la tabla dbo.PViaje también tiene una variable que se llama Id. Por lo tanto en ese caso la sintaxis sería:

```
bus <- sqlQuery(tranvias, paste("select Fecha, Bus, Hora, dbo.VCond.Id from
dbo.VCond, dbo.PViaje"))
```

Crear un dataframe con las variables de interés con DOS tablas RELACIONADAS

Las tablas de las bases de datos normalmente están relacionadas entre sí, por ejemplo en el diagrama de datos de nuestra base de datos tranvías podemos observar que las tablas PViaje y Pardas, por ejemplo, están relacionadas.



Queremos las variables Hora, IdPar, NomPar, donde IdPar(PViaje)=Id(Paradas):

```
Par <- sqlQuery(tranvias, paste("select Hora, IdPar, NomPar from dbo.PViaje join
dbo.Parada on dbo.PViaje.IdPar=dbo.Paradas.Id")
```

A esto se podrían añadir condiciones, como por ejemplo que IdVCond este entre 11599911 y 11669012.

```
Par <- sqlQuery(tranvias, paste("select Hora, IdPar, NomPar from dbo.PViaje join
dbo.Parada on dbo.PViaje.IdPar=dbo.Paradas.Id", "where IdVCond between
'11599911' and '11669012'")
```

Crear una dataframe con las variables de interés con TRES o más tablas RELACIONADAS

De la misma manera que se hace con dos tablas relacionadas, se puede hacer con más tablas, vamos a utilizar un ejemplo con 4 tablas relacionadas para ilustrar.

Queremos:

BD: tranvías

tablas: dbo.VCond, dbo.PViaje, dbo.CECA, TotPViaje

Variables interes en cada tabla:

VCond: Fecha,Linea,Sentiida

PViaje: IdPar, Hora

CECA: Tarjeta, IdPerfil, IdTipoBillete

TotPViaje: IdMedioPago

Relaciones:

dbo.VCond.Id=dbo.PViaje.IdVCond

dbo.PViaje.Id=dbo.CECA.IdPviaje

dbo.PViaje.Id=dbo.TotPViaje.IdPViaje

Condiciones:

Mes febrero de 2014

Ordenar por Fecha

```
datos.tot <- sqlQuery(tranvias, paste("select

dbo.VCond.Fecha, dbo.VCond.Linea, dbo.VCond.Sentiida,

dbo.PViaje.IdPar, dbo.PViaje.Hora,

dbo.CECA.Tarjeta, dbo.CECA.IdPerfil, dbo.CECA.IdTipoBillete,

dbo.TotPViaje.IdMedioPago

from dbo.PViaje join dbo.VCond on dbo.VCond.Id = dbo.PViaje.IdVCond

join dbo.CECA on dbo.PViaje.Id = dbo.CECA.IdPviaje

join dbo.TotPViaje on dbo.PViaje.Id = dbo.TotPViaje.IdPViaje",

"where dbo.CECA.Fecha between '20140201' and '20140228'",

"order by Fecha"))
```

2.2.2. PAQUETE SP

El paquete SP (Edzer J. Pebesma and Roger Bivand 2009) del R provee métodos y objetos para manipular datos espaciales en R. Permite trabajar con las estructuras espaciales básicas junto con sus atributos como son: puntos, líneas, polígonos y raster. Su conjunto de métodos permiten, de un modo discreto pero potente, procesar los datos, hacer sobre-posición, desplegar gráficamente los resultados de simulaciones y aplicar funciones de otros paquetes. Actualmente muchos paquetes de R, creados para análisis espacial, importan y exportan objetos con clase tipo SP.

Los creadores de este paquete fueron Edzer J. Pebesma y Roger S. Bivand. La idea nace durante la "pre-conference spatial data Workshops" (DCS 2003) al ver que existían muchos paquetes que servían para realizar análisis estadístico espacial pero carecían de uniformidad y compatibilidad para tratar los datos y objetos espaciales. Luego desde mayo de 2005, cuando se lanzó el paquete sp en el CRAN con su conjunto de funciones y familias de clases coherente para soportar objetos de representación espacial, se ha generado gran interés y desarrollos sp – orientados en análisis espacial con R.

2.2.2.1. TIPOS DE DATOS ESPACIALES

Los principales tipos de datos espaciales son: líneas, polígonos, puntos y raster. A continuación se hace una breve descripción de cada uno de estos elementos.

Líneas y polígonos

También conocidos como representación vector. Se usan para describir principalmente áreas y líneas. Estos datos se basan en la representación arco-nodo, es decir, curvas no interceptadas llamadas arcos con puntos o nodos en sus extremos. Se almacenan como una serie de puntos (x,y) o (x,y,z). Los puntos a lo largo de una curva se llaman vértices. Dos pares consecutivos de (x,y), definen un segmento de arco. A partir de los arcos se pueden formar elementos espaciales como líneas (p.e., caminos ríos) o polígonos (p.e., fincas o áreas de bosques, divisiones políticas de municipios). Cada elemento formado de este modo recibe un número que se emplea para relacionar el dato geométrico con atributos descriptivos en una base de datos. Para el análisis espacial estas estructuras en R, normalmente se importan de alguna fuente externa ya construidos y geo-referenciados a un sistema de coordenadas y las bases de datos se almacenan como data.frame.

Puntos

Los puntos son un caso especial de datos tipo vector. Son elementos independientes definidos por sus coordenadas y atributos (p.e., ciudades, casas). También pueden ser muestras de un campo continuo (p.e., elevación, precipitación), a menudo distribuido irregularmente. Los atributos pueden ser numéricos o de texto.

Rejilla

Es un tipo de almacenamiento de información espacial en una matriz regular. La representación puede ser como una malla de puntos, en cuyo caso se pretende representar un campo continuo y se denominan grids. También puede ser una malla con celdas (con área constante), conocida como raster propiamente. Estos últimos muy empleados para representar imágenes. Bien sea en puntos o celdas cada elemento puede

tener uno o múltiples atributos a través de una base de datos relacionada (un data.frame). Los datos se ubican por su posición en filas-columnas. Las celdas tienen lados iguales y la longitud de un segmento de lado define la resolución. La resolución controla el grado de detalle espacial que se captura en este tipo de representaciones. La ventaja de la forma grids es que cuando existen mucho faltantes, estos no se almacenan, es decir no necesariamente debe estar completo. La desventaja es que las coordenadas para cada punto se almacenan, lo cuan pude ser complicado para grids grandes. Este tipo de grids tienen un índice que permiten rápidamente hacer la transformación a la forma raster. Solo la selección de datos por filas y columnas funciona para la forma completa del grid (en sp para la clase de objetos SpatialGridDataFrame).

2.2.2.2. CLASES Y ATRIBUTOS DE LOS OBJETOS ESPACIALES

En el paquete sp cada uno de estos elementos espaciales representa una "clase" de dato. Cada una de estas clases puede tener atributos (o información) adicional; en cuyo caso, dependiendo de la forma de su almacenamiento, a la clase del objeto se le adiciona un nombre (lo que se le conoce como extensión). Los atributos adicionales se pueden almacenar como data.frame. En la siguiente tabla se resumen las clases de objetos disponibles en el paquete sp.

Tipo de dato	Clase	Atributos
Puntos	SpatialPoints	No
	SpatialPointsDataFrame	Si, data.frame
Lineas	Line	No
	Lines	No
	SpatialLines	No
	SpatialLinesDataFrame	Si, data.frame
Poligonos	Polygon	No
	Polygons	No
	SpatialPlygons	No
	SpatialPolygonsDataFrame	Si, data.frame
Grid o rejilla incompleta	SpatialPixel	No
	SpatialPixelDataFrame	Si, data.frame
Rejilla completa o Raster	SpatialGrid	No
	SpatialGridDataFrame	Si, data.frame

No nos vamos a centrar en hacer resúmenes con las funciones de este paquete porque no lo vamos a utilizar el nuestro análisis estadístico, solamente lo vamos a utilizar para crear el mapa de la red de transportes de Coruña que está en la base de datos espaciales.

A partir de la base de datos GTSF se construyeron los objetos sp, por ejemplo, el siguiente código crea un objeto espacial *lines* de un viaje (o sea un sentido de una linea) de forma que cada tramo es un objeto *lines*.

Suponiendo que IdPar son los códigos de stops y en index_par están las posiciones (los índices) de los stops en las líneas que describen la ruta (shape_lines), al finalizar la variable geo contiene el objeto espacial correspondiente al viaje.

```
    origen <- IdPar[-length(IdPar)]

    destino <- IdPar[-1]
```

```
    IdTramo <- origen*1000 + destino


    geo <- vector("list", length(IdTramo))

    distancia <- numeric(length(IdTramo))

    ini <- index_par[-length(index_par)]

    fin <- index_par[-1]


    for (i in seq_len(ntramos)) {

        geo[[i]] <- Lines(Line(shape_lines[ini[i]:fin[i], c("shape_pt_lon",
"shape_pt_lat")]), IdTramo[i])

        distancia[i] <- LinesLength(geo[[i]], longlat = TRUE)

    }


    geo <- SpatialLines(geo, proj4string = CRS("+proj=longlat +ellps=WGS84"))
```

Esto permite también calcular fácilmente las distancias entre dos stops, o sea la distancia de cada tramo.

2.2.3. PAQUETE DPLYR

En principio no teníamos pensado en utilizar ese paquete pero durante el análisis resultó interesante su uso, porque con el paquete dplyr resulta mucho más sencillo y rápido manejar datos con R. Además, una de las características más interesantes de dplyr es que el código escrito usando este paquete es mucho más legible y fácil de entender que el habitual, porque se puede utilizar sintaxis en cadena, en lugar de utilizar la sintaxis anidada.

Otra ventaja, es que utiliza comandos que coinciden con las acciones más comunes que se realizan sobre un conjunto de datos:

Filter: seleccionar filas.

Select: seleccionar columnas.

Arrange: ordenar.

Mutate: añadir nuevas variables.

Summarise: resumir mediante alguna medida numérica.

Para llevar a cabo acciones con dicho paquete debemos tener en cuenta algunas características comunes:

- El primer argumento siempre es un *data.frame.*

- El resto de argumentos indican lo que queremos hacer con el *data.frame*.
- El resultado siempre tiene también la estructura de *data.frame (*realmente una extensión)

El dplyr también puede ser utilizado para conectar R a gestores de bases de datos realizando una consulta a una base de datos externa, permite tratar ficheros muy grandes de los que sólo se importan las observaciones/variables que interesan. En ese trabajo no hemos utilizado ese paquete para acceder a las bases de datos porque como hemos comentado anteriormente en principio no teníamos pensado en utilizar ese paquete, y cuando nos resultó interesante ya teníamos los datos utilizando el paquete RODBC citado en la sección 2.2.1.

Haremos un pequeño resumen con las principales sentencias de dplyr. Para eso utilizaremos las tablas pViaje y vCond (como ya hemos comentado antes, en la tabla VCond se almacenan cada uno de los viajes realizados por los autobuses. Y en la tabla pViaje se relaciona el código de la parada con la tabla VCond, pudiendo obtener de esta forma la hora de paso del autobús en una determinada parada y los movimientos realizados en esta).

```
library(dplyr)

head(pViaje)
```

```
         Id  IdVCond IdPar Hora2 Linea Sentiida      Fecha nceca notros
1 113802424 11607527    76    22     4   vuelta 2014-02-04     1      0
2 113802425 11607527   194    22     4   vuelta 2014-02-04     0      0
3 113802426 11607527   195    22     4   vuelta 2014-02-04     0      0
4 113802427 11607527   196    22     4   vuelta 2014-02-04     1      0
5 113802428 11607527   197    22     4   vuelta 2014-02-04     1      2
6 113802429 11607527   167    22     4   vuelta 2014-02-04     0      0
```

```
head(vCond)
```

```
        Id      Fecha Linea Sentiida
1 11599911 2014-02-01  2100        1
2 11599912 2014-02-01  2100        1
3 11599913 2014-02-01  2100        0
4 11599914 2014-02-01  2100        1
5 11599915 2014-02-01  2100        0
6 11599916 2014-02-01  2100        1
```

UNIR TABLAS

Los datos están ahora repartidos en 2 tablas: pViajes y vCond. Vamos a agrupar estos en una sola tabla.

Las columnas que se corresponden en las dos tablas son: Id (de la tablas vCond) = IdVCond (de la tabla pViaje).

Primeramente tenemos que cambiar el nombre de una de esas dos columnas de manera que las dos tengan el mismo nombre. Por lo tanto vamos a cambiar el nombre de la columna Id de la tabla vCond a IdVCond.

```
names(vCond)[1] <- "IdVCond"

head(vCond)
```

```
       Fecha Linea Sentiida  IdVCond
1 2014-02-01  2100         1 11599911
2 2014-02-01  2100         1 11599912
3 2014-02-01  2100         0 11599913
4 2014-02-01  2100         1 11599914
5 2014-02-01  2100         0 11599915
6 2014-02-01  2100         1 11599916
```

Ahora para unir las dos tablas, vamos a utilizar la función `left_join` y luego le tenemos que decir el data.frame.x (vCond), data.frame.y (pViaje) y el nombre de la columna que relaciona las dos tablas:

```
pViaje_vCond <- left_join(vCond, pViaje, by = "IdVCond")

head(pViaje_vCond)
```

```
     Fecha.x Linea.x Sentiida.x  IdVCond        Id IdPar Hora2 Linea.y Sentiida.y    Fecha.y nceca notros
1 2014-02-01    2100          1 11599911 113649543   424    07      21        ida 2014-02-01     0      0
2 2014-02-01    2100          1 11599911 113649544   526    07      21        ida 2014-02-01     0      0
3 2014-02-01    2100          1 11599912 113649545   526    07      21        ida 2014-02-01     0      0
4 2014-02-01    2100          1 11599912 113649546     5    07      21        ida 2014-02-01     0      0
5 2014-02-01    2100          1 11599912 113649547     6    07      21        ida 2014-02-01     0      0
6 2014-02-01    2100          1 11599912 113649548     7    07      21        ida 2014-02-01     0      0
```

La función `left_join` devuelve todas las filas de los dos data frames y todas las columnas de los dos data frames. Las filas del data.frame.x sin correspondencia en data.frame.y tendrán valores de NA en las nuevas columnas. Si hay varias coincidencias entre x e y, se devuelven todas las correspondencias.

También se puede unir tablas utilizando las funciones:

`inner_join`

Devuelve solamente las filas del data.frame.x donde hay valores coincidentes en data.frame.y, y todas las columnas de data.frame.x y data.frame.y. Si hay varias coincidencias entre x e y, se devuelven todas correspondencias.

`semi_join`

Devuelve todas las filas del data.frame.x donde hay valores coincidentes en el data.frame.y, manteniendo sólo las columnas del data.frame.x.

Difiere de `inner_join` debido a que `inner_join` devuelve una fila del data.frame.x para cada fila a juego del data.frame.y, y `semi_join` no duplica filas de x.

`anti_join`

Devuelve todas las filas del data.frame.x en las que no hay correspondencia con y, manteniendo sólo las columnas de x.

OPERACIONES CON CASOS

- select(): seleccionar columnas por nombre

En la tabla pViaje, queremos seleccionar solamente las variables Fecha, Linea, Sentiida, IdPar.

```
reduced <- select(pViaje, Fecha,Linea,Sentiida,IdPar)

head(reduced)
```

```
       Fecha Linea Sentiida IdPar
1 2014-02-04     4   vuelta    76
2 2014-02-04     4   vuelta   194
3 2014-02-04     4   vuelta   195
4 2014-02-04     4   vuelta   196
5 2014-02-04     4   vuelta   197
6 2014-02-04     4   vuelta   167
```

Toda las variables menos Fecha.

```
select(pViaje, -Fecha)
```

Las columnas entre IdPar y Sentiida.

```
select(pViaje, IdPar:Sentiida)
```

La columnas cuyo el nombre contenga `'id'`

```
select(pViaje, contains('id'))
```

Las columnas que empiezan por `'n'`

```
select(pViaje, starts_with('n'))
```

También podemos utilizar la función `select` para renombrar nuestras columnas:

```
pViaje <- select(pViaje, Sentido = Sentiida, IdStops = IdPar)
```

- Filter(): suprimir las filas que no respectan una condición

En la tabla pViaje queremos sólo la línea 21.

```
filtro<-filter(pViaje, Linea == "21")

head(filtro)
```

```
         Id  IdVCond IdPar Hora2 Linea Sentiida      Fecha nceca notros
1 113649543 11599911   424    07    21      ida 2014-02-01     0      0
2 113649544 11599911   526    07    21      ida 2014-02-01     0      0
3 113649545 11599912   526    07    21      ida 2014-02-01     0      0
4 113649546 11599912     5    07    21      ida 2014-02-01     0      0
5 113649547 11599912     6    07    21      ida 2014-02-01     0      0
6 113649548 11599912     7    07    21      ida 2014-02-01     0      0
```

En la tabla pViaje queremos sólo la línea 21 y 14.

```
filter(pViaje, Linea=='21' | Linea=='14')
```

o

```
filter(pViaje, Linea %in% c("21", "14"))
```

Solo la línea 2100 con fecha de 03/02/2014.

```
filter(pViaje, Linea=='2100' | Fecha=='2014-02-03')
```

Paradas en los stops con id menor que 5.

```
filter(pViaje, IdPar < 5)
```

Solo la línea 21, paradas en los stops con id menor que 5.

```
filter(pViaje, Linea == "21" & IdPar < 5)
```

Solo la línea 21, paradas en los stops con id menor que 5.

```
filter(pViaje, Linea == "21" & IdPar < 5)
```

Para selecionar registros por posición, usar `slice()`:

```
slice(pViaje, 30:45)
```

- arrange(): ordenar filas

En la tabla pViaje queremos ordenar por IdPar.

```
arrange(pViaje, IdPar)
```

	Id	IdVCond	IdPar	Hora2	Linea	Sentiida	Fecha	nceca	notros
1	113728794	11603766	1	07	23A	vuelta	2014-02-03	0	0
2	113728795	11603767	1	07	23A	ida	2014-02-03	3	0
3	113728843	11603768	1	08	23A	vuelta	2014-02-03	0	0
4	113728844	11603769	1	08	23A	ida	2014-02-03	1	0
5	113728894	11603770	1	09	23A	vuelta	2014-02-03	0	0
6	113728895	11603771	1	09	23A	ida	2014-02-03	0	1

En la tabla pViaje queremos ordenar por IdPar en orden decreciente.

```
arrange(pViaje, desc(IdPar))
```

o

```
arrange(pViaje, -IdPar)
```

En la tabla pViaje queremos ordenar por IdPar y Hora

```
arrange(pViaje, IdPar,Hora)
```

SINTAXIS EN CADENA

Como ya habíamos comentado anteriormente, una de las grandes ventajas de este paquete es que se puede utilizar sintaxis en cadena, en lugar de utilizar la sintaxis anidada.

El paquete dplyr utiliza el operador **%>%** , conocido en inglés como *pipe operator*, que nos permite evitar paréntesis anidados. De esta manera, podemos leer las instrucciones de arriba a abajo y de izquierda a derecha, en lugar de leer desde el interior del paréntesis anidado hacia afuera.

Primero se escribe el nombre del dataframe y luego las acciones en el orden en que se realizan separadas por el operador %>%. Por ejemplo, si queremos seleccionar las variables que contienen Id, seleccionar los IdVCond > 11603771 y ordenarlos de menor a mayor:

```
pViaje %>%

  select(contains('Id'))  %>%

  filter(IdVCond > 11603771)   %>%

  arrange(IdVCond)
```

Podemos comparar el código anterior con el código anidado habitual:

```
arrange(filter(select(pViaje,contains('Id')), IdVCond > 116003771),IdVCond)
```

Una vez cargado dplyr, la sintaxis encadenada puede utilizarse con cualquier comando de R, así que es posible que dplyr sea un paquete muy útil.

Veamos un ejemplo en el que se calcula la distancia euclídea entre dos vectores:

```
x1 <- 1:6; x2 <- 7:12

# Sintaxis anidada

sqrt(sum((x1-x2)^2))

# Sintaxis en cadena

(x1-x2)^2 %>% sum() %>% sqrt()
```

- mutate(): Añadir nuevas variables

Vamos a crear nuevas variables que son función de las ya existentes.

En la tabla pViaje queremos una nueva variable (ntot) que sea la suma de nceca y notros

```
pViaje %>%

  mutate(ntot = nceca+notros)
```

```
         Id  IdVCond IdPar Hora2 Linea Sentiida      Fecha nceca notros ntot
1 113802424 11607527    76    22     4   vuelta 2014-02-04     1      0    1
2 113802425 11607527   194    22     4   vuelta 2014-02-04     0      0    0
```

```
3 113802426 11607527   195    22     4   vuelta 2014-02-04     0      0     0
4 113802427 11607527   196    22     4   vuelta 2014-02-04     1      0     1
5 113802428 11607527   197    22     4   vuelta 2014-02-04     1      2     3
6 113802429 11607527   167    22     4   vuelta 2014-02-04     0      0     0
```

- group_by() + summarise(): Resumir (subconjuntos de) variables:

Usamos `summarise()` para aplicar comandos a variables. Normalmente se usa en combinación con `group_by()` de manera que se calculen estadísticos para subgrupos de observaciones. En el siguiente ejemplo se calcula la media de pagos con tarjetas (nceca) para los buses de cada una de las líneas:

```
pViaje %>%

  group_by(Linea) %>%

  summarise(mean(nceca))
```

```
   Linea mean(nceca)
1      1   0.7808785
2      2   0.8770339
3      3   0.7186227
4     3A   0.7006061
5      4   1.2197998
6      5   0.9856928
7      6   1.1699590
8     6A   0.8450106
9      7   0.9336591
10    2A   0.8604987
..   ...         ...
```

Si quisiéramos hacer lo mismo para varias variables a la vez, utilizaríamos la función `summarise_each()`.

- n():contar el número de registros.

La función `n()` permite dentro `summarise()`, `mutate()` y `filter()` contar el número de registros.

Por ejemplo queremos la cantidad de veces que cada línea ha parado en los stops.

```
pViaje %>%  group_by(Linea) %>%

    summarize(Freq = n())
```

```
   Linea  Freq
1      1 61993
2      2 33188
3      3 34356
4     3A 59557
5      4 95405
6      5 69895
..   ...   ...
```

2.2.4. PAQUETE IGRAPH

Es una librería de R para el análisis de redes que cuenta con funciones para crear y manipular grafos grandes (por ejemplo, del orden de millones de vértices y aristas) con facilidad.

Haremos un pequeño resumen con las principales sentencias de igraph.

CREAR GRAFOS A MANO

Para crear grafos pequeños podemos utilizar la función `graph.formula`, vamos a crear por ejemplo un grafo con 7 vértices y 10 arcos.

```
library(igraph)

grafo <- graph.formula(1-2, 1-3, 2-3, 2-4, 3-5, 4-5, 4-6, 4-7, 5-6, 6-7)
```

Podemos visualizar los vértices y arcos del objeto grafo que hemos creado.

```
V(grafo)
```

```
+ 7/7 vertices, named:
[1] 1 2 3 4 5 6 7
```

```
E(grafo)
```

```
+ 10/10 edges (vertex names):
 [1] 1--2 1--3 2--3 2--4 3--5 4--5 4--6 4--7 5--6 6--7
```

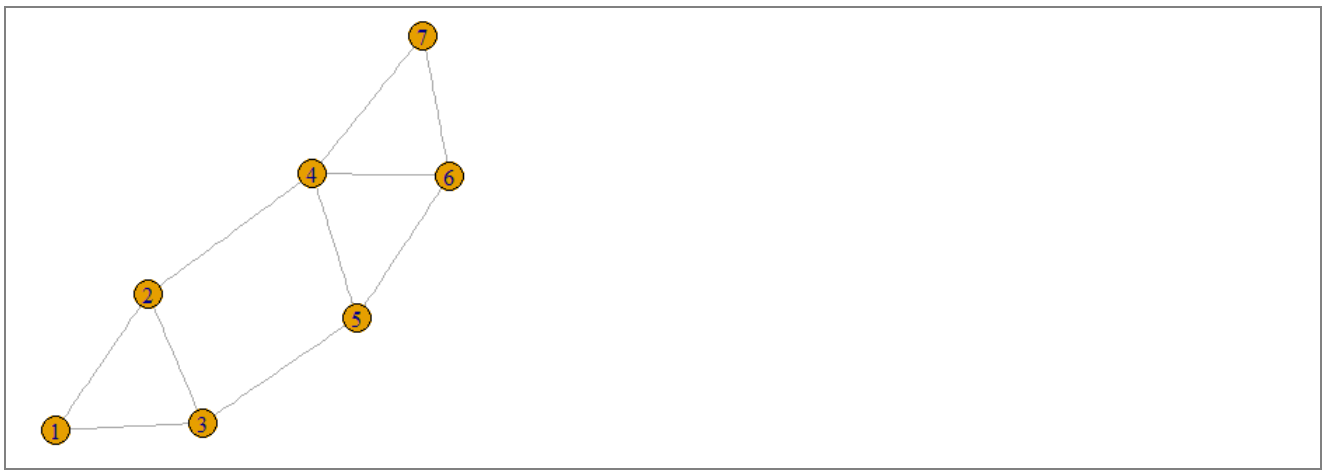
También podríamos obtener esa información utilizando la función `str`.

```
str(grafo)
```

```
IGRAPH UN-- 7 10 --
+ attr: name (v/c)
+ edges (vertex names):
1 -- 2, 3
2 -- 1, 3, 4
3 -- 1, 2, 5
4 -- 2, 5, 6, 7
5 -- 3, 4, 6
6 -- 4, 5, 7
7 -- 4, 6
```

Ahora vamos a visualizar nuestro grafo:

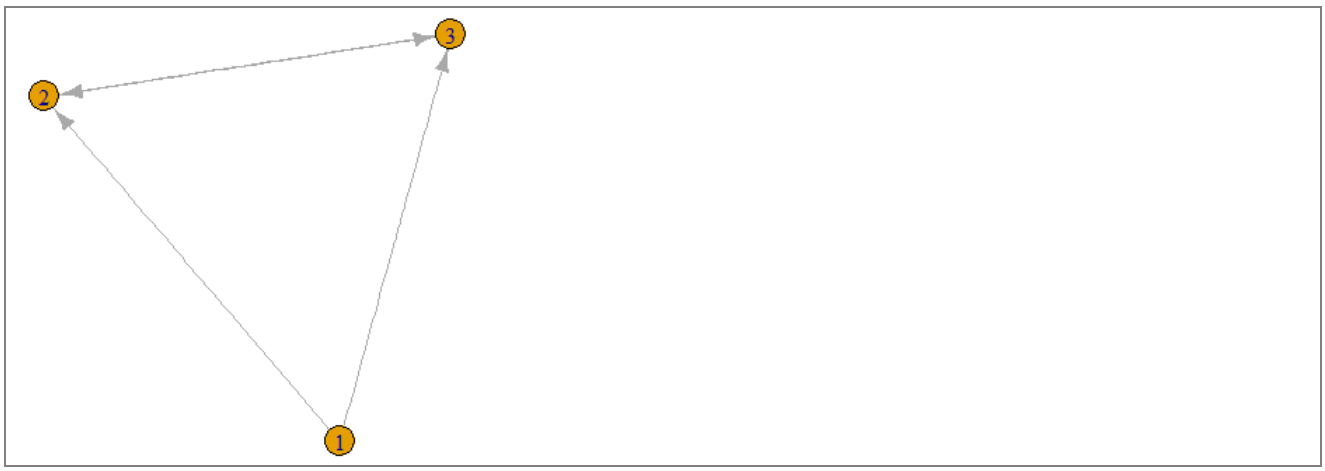
```
plot(grafo)
```



Si quisiéremos un grafo dirigido, al crear el grafo tendríamos que indicar con un "-" en que nodo empieza el arco y con un "+" en que nodo termina el arco.

```
grafo.dirigido <- graph.formula(1-+2, 1-+3, 2++3)

plot(grafo.dirigido)
```



Hemos etiquetado nuestros vértices de la forma convencional, que es por números, pero también podríamos hacerlo con nombres por ejemplo.

```
grafo.orientado  <- graph.formula(Sam-+Mary, Sam-+Tom, Mary++Tom)

str(grafo.orientado)
```

```
IGRAPH DN-- 3 4 --
+ attr: name (v/c)
+ edges (vertex names):
[1] Sam ->Mary Sam ->Tom  Mary->Tom  Tom ->Mary
```

Otra opción, sería simplemente cambiar los números que estaban en los vértices por etiquetas.

```
V(grafo.orientado)$name <- c("Sam", "Mary", "Tom")
```

Podemos estar interesados en analizar subgrafos, por ejemplo en nuestro primer grafo podríamos estar interesados en el subgrafo formado solamente por los vértices del 1 al 5.

```
grafo2 <- induced.subgraph(grafo, 1:5)
```

```
str(grafo2)
```

```
IGRAPH UN-- 5 6 --
+ attr: name (v/c)
+ edges (vertex names):
[1] 1--2 1--3 2--3 2--4 3--5 4--5
```

Otra opción seria excluir los vértices 6 y 7 del grafo inicial.

```
subgrafo <- grafo - vertices(c(6,7))
```

De la misma manera podríamos recuperar los vértices y aristas que hemos borrado.

```
grafo <- subgrafo + vertices(c(6,7))

grafo <- subgrafo + edges(c(4,6),c(4,7),c(5,6),c(6,7))
```

También podemos unir dos subgrafos.

```
h1 <- subgrafo

h2 <- graph.formula(4-6, 4-7, 5-6, 6-7)

grafo <- graph.union(h1,h2)
```

Con frecuencia también necesitamos agregar a nuestros grafos más información sobre los vértices y aristas. Por ejemplo en nuestro ejemplo de grafo orientado donde hemos puesto etiquetas a los vértices.

```
V(grafo.orientado)$name
```

```
[1] "Sam"  "Mary" "Tom"
```

Hemos llamado los vértices de "Sam", "Mary", "Tom" y nos podría interesar poner en el grafo por ejemplo el sexo de cada uno.

```
V(grafo.orientado)$gender <- c("M","F","M")
```

Podemos también cambiar al color de los vértices.

```
V(grafo)$color <- "red"
```

También podemos darle un nombre a nuestro grafo.

```
grafo$name <- "Ejemplo"
```

CREAR GRAFOS UTILIZANDO DATA FRAMES

Normalmente tenemos muchos vértices y arcos, y nuestros datos de red suelen estar almacenados en tablas de datos que contienen no solo los vértices y nodos como también información adicional sobre esos, lo que torna inviable crear un grafo a mano.

En R es conveniente tener dos data frames (uno de vértices y otro de arcos) con toda esa información.

En el data frame de los vértices la primera columna debe contener los nombres de los vértice, mientras que cada una de las otras columnas contienen informaciones sobre esos vértices.

Del mismo modo, las dos primeras columnas del data frame de los arcos debe contener la lista de arcos que definen nuestro grafo, mientras que cada una de las otras columnas contienen informaciones sobre esos arcos.

Como ejemplo vamos a utilizar nuestros datos de tranvías, queremos representar con un grafo nuestras líneas de autobuses en donde cada vértice sea un stop y cada arco sea el un tramo entre dos stops. De esa manera tenemos dos data frames, uno con los datos de los stops (vértices), y otro con los datos de los tramos.

```
load('stops.RData')

head(stops)
```

```
   IdPar Freq   mean_tot Billetes_tot stop_lat  stop_lon
21    23 1584 0.23926768          379 43.35964 -8.403302
22    24 1589 0.31214600          496 43.36200 -8.405097
23    25 1589 0.10509755          167 43.36442 -8.404943
24    26 1568 0.13329082          209 43.36651 -8.404457
25    27 1568 0.08482143          133 43.36871 -8.401460
26    28 1568 0.02806122           44 43.36972 -8.399210
```

En el data frame de vértices tenemos en la primera columna el Id del Stop (nombre de nuestros vértices), y luego tenemos informaciones sobre cada stop, en febrero de 2014, como cantidad de veces que han parado en ese stop, media de viajeros, número total de billetes, latitud y la longitud.

```
library(igraph)

load('tramos_red.RData')

head(tramos_red)
```

```
  origen destino IdTramo distancia    tiempo velocidad linea CodSent
1     68      69   68069 0.2228437 0.9000000  14.85625     1       0
2     69      70   69070 0.2883592 0.7000000  24.71650     1       0
3     70      71   70071 0.1692824 0.5000000  20.31389     1       0
4     71      72   71072 0.2473788 1.1000000  13.49339     1       0
5     72      73   72073 0.2545900 0.8333333  18.33048     4       0
6     73      74   73074 0.2578972 0.8333333  18.56860     4       0
```

En el data frame de arcos tenemos en la primera columna el stop de origen de ese arco, en la segunda columna el stop de destino de ese arco, y otras informaciones sobre cada arco como un id de tramo, distancia, tiempo de recorrido, velocidad media, línea y código de sentido.

Vamos a crear la red con la función `graph.data.frame`.

```
v.bus <- stops

a.bus <- tramos_red

tranvias <- graph.data.frame(a.bus,directed="TRUE",vertices=v.bus)

tranvias$name <- "Tranvias Coruna"
```

Visualización de un grafo

Podemos dibujar grafos con varias disposiciones distintas:

```
igraph.options(vertex.size=3,vertex.label=NA,edge.color="gray80",edge.arrow.size
=0.5)
```

Disposición de círculo

```
plot.igraph(tranvias,layout=layout.circle)
```

Método de Kamada y Kawai (apropiado para redes pequeñas)

```
plot.igraph(tranvias,layout=layout.kamada.kawai)
```

Método de Fruchterman and Reingold (apropiada para redes de tamaño medio)

```
plot.igraph(tranvias,layout=layout.fruchterman.reingold)
```

Método DrL (apropiado para redes muy grandes)

```
plot.igraph(tranvias,layout=layout.drl)
```

Y si no sabemos que grafo es el más adecuado podemos utilizar la opción layout.auto que determina de manera automática que procedimiento es el más adecuado.

```
plot.igraph(tranvias,layout=layout.auto)
```

Análisis descriptivo de un grafo

Dado un grafo lo primero que nos interesa es analizar la estructura de esa red, el paquete `igraph` nos permite acceder a las características principales de ese grafo.

Verificar número de nodos.

```
vcount(tranvias)
```

```
[1] 450
```

Verificar el número de aristas.

```
ecount(tranvias)
```

```
[1] 1092
```

Podemos verificar si ese grafo es o no un multigrafo.

```
is.simple(tranvias)
```

También se puede verificar los nodos que son adyacentes, por ejemplo si queremos saber con que paradas está conectada directamente la parada 3.

```
neighbors(tranvias, 3)
```

O verificar el grado de un nodo.

```
degree(tranvias)
```

O en el caso de un grafo orientado, podemos también, verificar cuantas aristas entran (mode="in"), y cuantas aristas salen (mode="out") de cada nodo.

```
degree(tranvias, mode = "in")

degree(tranvias, mode = "out")
```

Podemos verificar si nuestro grafo está o no conectado.

```
s.connected(tranvias)
```

Acceder a los atributos de los nodos.

```
head(V(tranvias)$Freq_par)

head(V(tranvias)$mean_nceca)

head(V(tranvias)$mean_tot)

head(V(tranvias)$Desemp)
```

Acceder a los atributos de los arcos

```
head(E(tranvias)$distancia)

head(E(tranvias)$tiempo)

head(E(tranvias)$velocidad)
```

Nos podemos plantear preguntas sobre los nodos, como por ejemplo ¿En qué stop se suben mas pasajeros?

```
max(V(tranvias)$billetes_tot)

which.max(V(tranvias)$billetes_tot)

V(tranvias)$name[which(V(tranvias)$billetes_tot=="50671")]
```

O aun plantear preguntas sobre los arcos, como por ejemplo, ¿Cuál es el tramo con mayor distancia?

```
max(E(tranvias)$distancia)

sum(E(tranvias)$distancia==max(max(E(tranvias)$distancia)))

which(E(tranvias)$distancia==max(max(E(tranvias)$distancia)))

E(tranvias)[which(E(tranvias)$distancia==max(max(E(tranvias)$distancia)))]
```

2.3. MAPAS

Existen muchos paquetes para integrar mapas de ciudades en R, entre ellos están OpenStreetMap (Fellows, 2012), plotGoogleMaps (Milan Kilibarda, 2015), RgoogleMaps (Loecher, 2012), y ggmap (Kahle andWickham, 2012).

En este trabajo decidimos utilizar el paquete Osmar que proporciona infraestructura para acceder a los datos de OpenStreetMap de diferentes fuentes, permitiendo trabajar con los datos OSM en el lenguaje R y convertir los datos en objetos basados en los paquetes existentes de R. También vamos a utilizar, el paquete RgoogleMaps. Vamos a explicar cómo utilizarlos.

2.3.1. OPENSTREETMAP

OpenStreetMap (también conocido como **OSM**) es un proyecto colaborativo para crear mapas libres y editables. Los mapas se crean utilizando información geográfica capturada con dispositivos GPS móviles, orto fotografías y otras fuentes libres. Esta cartografía, tanto las imágenes creadas como los datos vectoriales almacenados en su base de datos, se distribuye bajo licencia abierta Open Database License (ODbL).

Los primeros datos del mapa fueron recopilados desde cero por voluntarios mediante un sistemático trabajo de campo a través de dispositivos GPS de mano y ordenadores portátiles o grabadoras de voz, información que posteriormente se incorporaban a la base de datos de OpenStreetMap.

Más recientemente la disponibilidad de fotografías aéreas y otras fuentes de datos comerciales y públicas ha aumentado considerablemente la velocidad de este trabajo, permitiendo que la recogida de información tenga una mayor precisión.

La recogida de información en campo es realizado por voluntarios, que consideran la contribución al proyecto un adictivo hobby. Aprovechando sus desplazamientos a pie, en bicicleta o en automóvil y utilizando un dispositivo GPS, van capturando las trazas y waypoints, utilizando además, para registrar la información asociada a esas trazas o puntos de interés, bloc de notas, grabadora de voz o una cámara de fotos digital.

La existencia o liberación de datos públicos de instituciones gubernamentales con un tipo de licencia compatible con la de OpenStreetMap ha permitido importar esa información geográfica en la base de datos del proyecto. En España, por ejemplo, el Instituto Geográfico Nacional (IGN), organismo público encargado de la creación, mantenimiento y comercialización de la cartografía oficial en el país, ha modificado en abril de 2008 la licencia de utilización de sus datos, liberando parte de estos de forma gratuita para cualquier tipo de uso.

Algunas empresas privadas, han donado datos al proyecto bajo licencias adecuadas para este fin. En particular, los datos provenientes de la compañía holandesa *Automotive Navigation Data* (AND), la cual donó la cobertura completa para los Países Bajos y las principales carreteras de China e India.

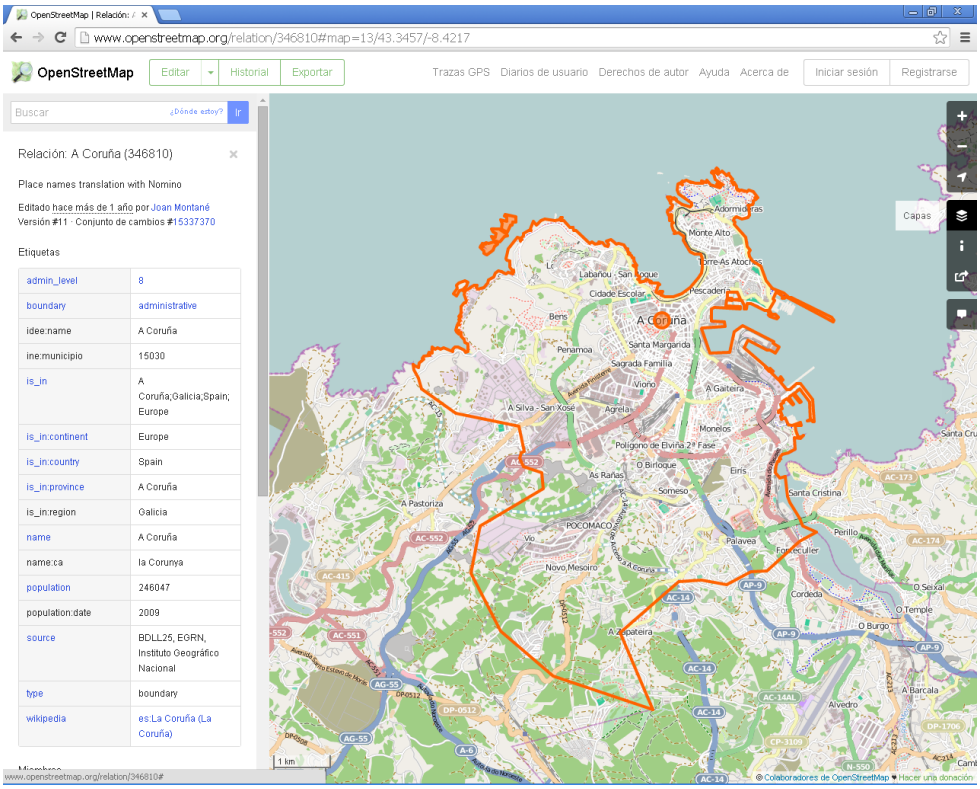
OpenStreetMap utiliza una estructura de datos topológica. Los datos se almacenan en el datum WGS84 lat/lon (EPSG:4326) de proyección de Mercator. Los datos primitivos o elementos básicos de la cartografía de OSM son:

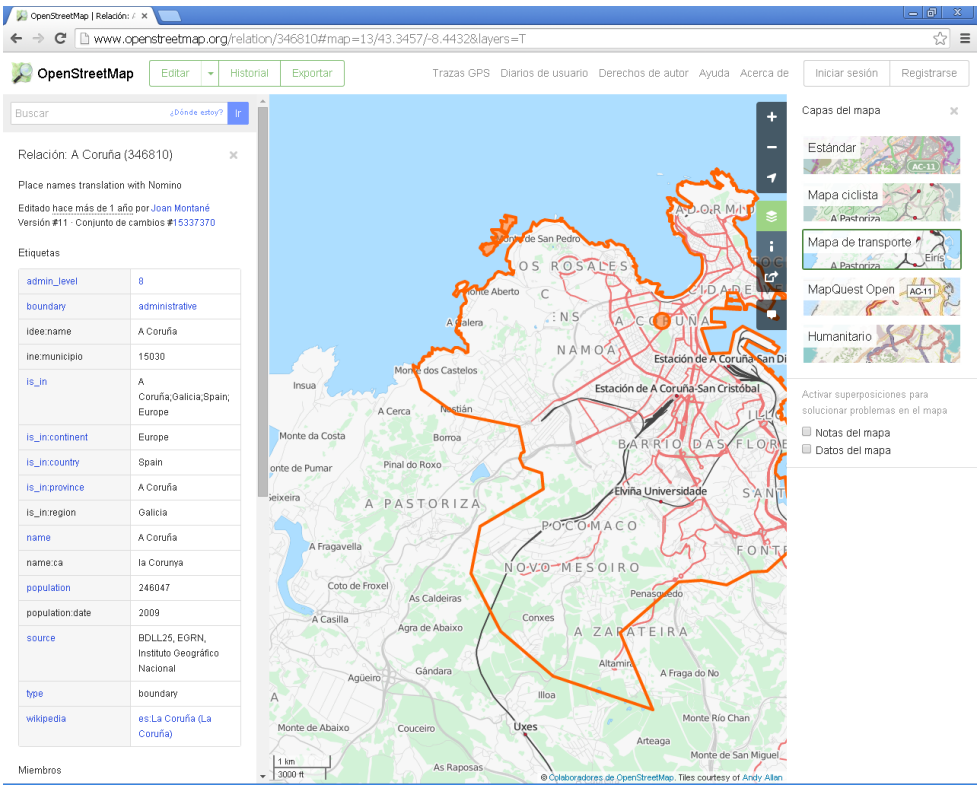
- Los **nodos** (*nodes*): son puntos que recogen una posición geográfica dada.
- Las **vías** (*ways*): son una lista ordenada de nodos que representa una poli línea o un polígono (cuando una poli línea empieza y finaliza en el mismo punto).
- Las **relaciones** (*relations*): son grupos de nodos, vías y otras relaciones a las que se pueden asignar determinadas propiedades comunes. Por ejemplo, todas aquellas vías que forman parte del Camino de Santiago.
- Las **etiquetas** (*tags*): se pueden asignar a nodos, caminos o relaciones y constan de una clave (*key*) y de un valor (*value*). Por ejemplo:highway=trunk

La ontología de las características del mapa (principalmente el significado de las etiquetas) se mantiene mediante a una wiki (sitio web cuyas páginas pueden ser editadas por múltiples voluntarios a través del navegador web. L os usuarios pueden crear, modificar o eliminar un mismo texto que comparten).

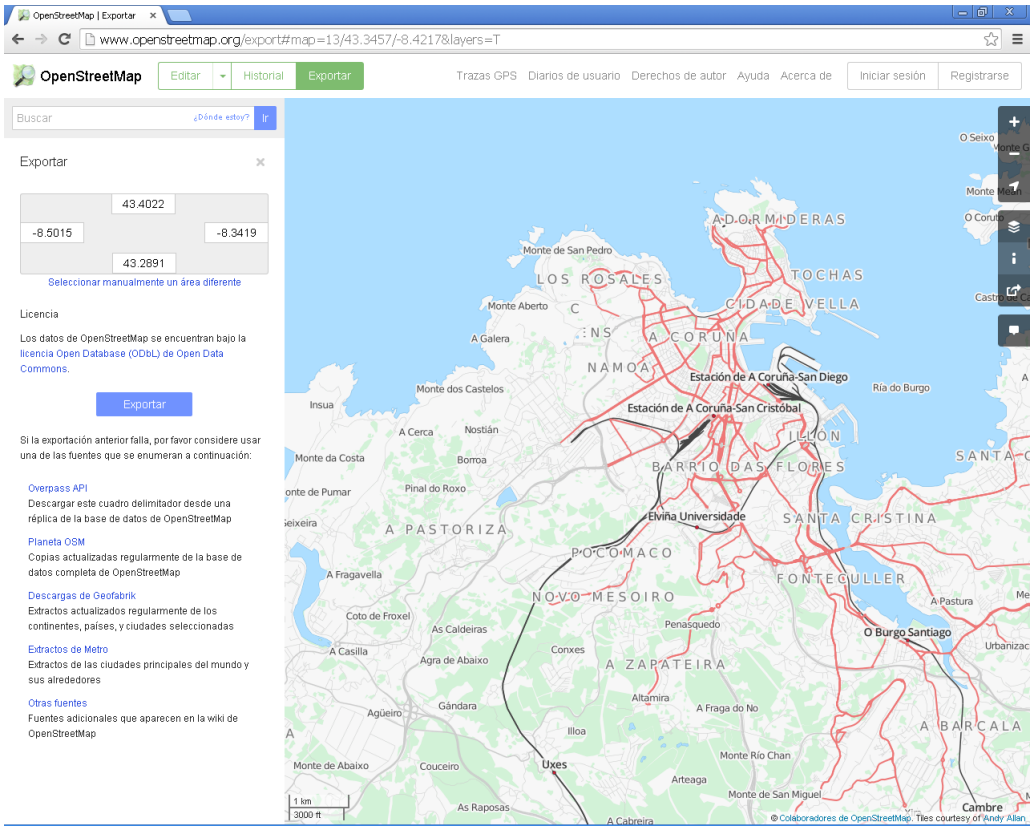
A partir de los datos del proyecto OpenStreetMap no sólo se pueden producir mapas de carreteras, sino también para la creación de mapas de senderismo, mapas de vías ciclables, mapas náuticos, mapas de estaciones de esquí, etc. También se usan en aplicaciones para el cálculo de las rutas óptimas para vehículos y peatones. Gracias a su licencia abierta los datos brutos son de libre acceso para el desarrollo de otras aplicaciones.

Estamos interesado en obtener el mapa de transporte público de Coruña, para eso vamos a abrir la página web http://www.openstreetmap.org/, buscamos la ciudad de A Coruña, y en el menú que se encuentra a la derecha seleccionamos la opción “Capas”, y luego “Mapas de transporte públicos”.





Luego en el menú superior seleccionamos exportar y nos quedamos con las coordenadas que nos aparecen en pantalla.



2.3.1.1. INTEGRAR OPENSTREETMAP Y R

El paquete osmar integra el OpenStreetMap en R, y mientras que los demás paquetes proporcionan acceso a los datos ya presentados (por ejemplo, imágenes de mapa de bits), Osmar permite el uso de los datos OSM en bruto.

Comenzamos con la definición de la fuente de datos. Actualmente se utilizan dos fuentes para obtener los mapas: HTTP-API (mapas online) y archivos del planeta (hay que descargar). Vamos a utilizar la API del proyecto OSM con la url URL default http: / / api.openstreetmap.org/api/0.6 /:

Primeramente vamos a instalar y cargar el paquete:

```
install.packages('osmar')

library(osmar)
```

Indicamos donde están los datos:

```
src <- osmsource_api()
```

Con la función corner_bbox especificamos las coordenadas del área que queremos representar, considerando que por HTTP-API sólo podemos manejar 50.000 nodos, si necesitáramos mas nodos tendríamos que obtener los datos descargando los archivos del planeta. Vamos a utilizar las coordenadas de Coruña obtenidas anteriormente en la página web de OpenStreeMap.

```
bb <- corner_bbox(-8.42, 43.35,-8.39,43.38)
```

La función get_osm nos permite obtener datos de OSM como objeto osmar, vamos a utilizar dicha función para obtener los datos con las coordenadas que hemos especificado, y guardar el archivo.

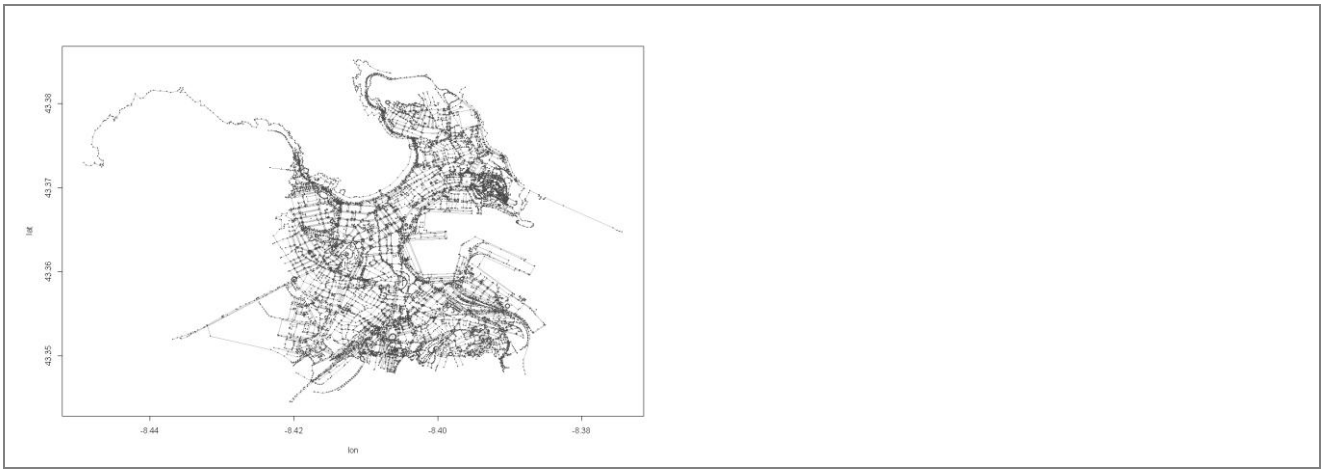
```
coru <- get_osm(bb, source = src)

save(coru, file = 'osm_coru.RData')
```

Ahora vamos a leer y visualizar el archivo del mapa con las coordenadas que hemos guardado

```
load('osm_coru.RData')

plot(coru)
```



Del mapa obtenido nos interesa la estructura: nodos, vías, relaciones y etiquetas.

```
str(Coru)
```

```
List of 3
 $ nodes    :List of 2
  ..$ attrs:'data.frame':       36615 obs. of  9 variables:
  .. ..$ id        : num [1:36615] 1.87e+09 4.71e+08 4.71e+08 2.77e+08 4.71e+08
...
  .. ..$ visible   : Factor w/ 1 level "true": 1 1 1 1 1 1 1 1 1 1 ...
  .. ..$ timestamp: POSIXlt[1:36615], format: "2012-08-20 13:26:44" "2014-11-14
22:44:43" "2012-08-20 13:26:45" "2012-08-20 13:26:46" ...
  .. ..$ version   : num [1:36615] 1 3 2 4 2 2 2 3 3 4 ...
  .. ..$ changeset: num [1:36615] 12796180 26788493 12796180 12796180 12796180
...
  .. ..$ user      : Factor w/ 138 levels "(Í¡Â° ÍœÊ- Í¡Â°)",..: 91 4 91 91 91 91
91 4 4 4 ...
  .. ..$ uid       : Factor w/ 138 levels "103188","1060112",..: 122 48 122 122
122 122 122 48 48 48 ...
  .. ..$ lat       : num [1:36615] 43.4 43.4 43.4 43.4 43.4 ...
  .. ..$ lon       : num [1:36615] -8.41 -8.41 -8.41 -8.41 -8.41 ...
  ..$ tags :'data.frame':       47247 obs. of  3 variables:
  .. ..$ id: num [1:47247] 2.77e+08 2.77e+08 2.77e+08 2.77e+08 4.71e+08 ...
  .. ..$ k : Factor w/ 173 levels "addr:city","addr:country",..: 21 46 67 98 46
27 158 158 158 158 ...
  .. ..$ v : Factor w/ 1893 levels "-8.41000483888889",..: 1886 643 1138 1747
851 1794 1408 1408 1408 1408 ...
  ..- attr(*, "class")= chr [1:3] "nodes" "osmar_element" "list"
 $ ways     :List of 3
  ..$ attrs:'data.frame':       5639 obs. of  7 variables:
  .. ..$ id        : num [1:5639] 39441964 4884907 39443333 39443332 39443331 ...
  .. ..$ visible   : Factor w/ 1 level "true": 1 1 1 1 1 1 1 1 1 1 ...
  .. ..$ timestamp: POSIXlt[1:5639], format: "2014-08-27 21:29:54" "2012-07-27
10:40:00" "2010-08-24 12:16:01" "2010-08-24 12:16:02" ...
  .. ..$ version   : num [1:5639] 4 9 2 3 2 2 3 8 3 2 ...
  .. ..$ changeset: num [1:5639] 25062426 12509658 5579230 5579230 5579230 ...
  .. ..$ user      : Factor w/ 81 levels "(Í¡Â° ÍœÊ- Í¡Â°)",..: 33 54 44 44 44 44
44 3 3 8 ...
  .. ..$ uid       : Factor w/ 81 levels "109925","121415",..: 4 70 46 46 46 46
46 23 23 72 ...
  ..$ tags :'data.frame':       14213 obs. of  3 variables:
  .. ..$ id: num [1:14213] 39441964 39441964 39441964 4884907 4884907 ...
  .. ..$ k : Factor w/ 109 levels "access","addr:city",..: 42 57 73 69 94 73 57
42 70 73 ...
  .. ..$ v : Factor w/ 1575 levels "-1","-2","+34  981 182 057",..: 1332 489
1572 493 1013 1572 1385 821 882 1572 ...
  ..$ refs :'data.frame':       49213 obs. of  2 variables:
  .. ..$ id : num [1:49213] 39441964 39441964 39441964 4884907 4884907 ...
  .. ..$ ref: num [1:49213] 4.72e+08 3.05e+09 4.72e+08 3.16e+07 3.16e+07 ...
  ..- attr(*, "class")= chr [1:3] "ways" "osmar_element" "list"
 $ relations:List of 3
  ..$ attrs:'data.frame':       194 obs. of  7 variables:
  .. ..$ id        : num [1:194] 936941 2274185 2274188 2274187 3897952 ...
  .. ..$ visible   : Factor w/ 1 level "true": 1 1 1 1 1 1 1 1 1 1 ...
  .. ..$ timestamp: POSIXlt[1:194], format: "2010-05-25 22:16:15" "2013-09-10
15:01:48" "2013-09-10 15:01:48" "2013-09-10 15:01:48" ...
  .. ..$ version   : num [1:194] 1 2 2 2 2 2 1 1 1 1 ...
  .. ..$ changeset: num [1:194] 4808815 17768834 17768834 17768834 25021850 ...
  .. ..$ user      : Factor w/ 19 levels "(Í¡Â° ÍœÊ- Í¡Â°)",..: 3 19 19 19 2 2 2
2 7 7 ...
  .. ..$ uid       : Factor w/ 19 levels "121415","1241595",..: 10 7 7 7 8 8 8 8
19 19 ...
```

```
  ..$ tags :'data.frame':       939 obs. of  3 variables:
  .. ..$ id: num [1:939] 936941 936941 2274185 2274185 2274185 ...
  .. ..$ k : Factor w/ 315 levels "addr:city","addr:country",..: 301 309 47 298
309 47 298 309 47 298 ...
  .. ..$ v : Factor w/ 383 levels "+34 981 20 92 51",..: 256 313 314 370 301 315
370 301 92 370 ...
  ..$ refs :'data.frame':       7759 obs. of  4 variables:
  .. ..$ id  : num [1:7759] 936941 936941 936941 2274185 2274185 ...
  .. ..$ type: Factor w/ 3 levels "node","relation",..: 3 1 3 1 1 1 1 1 1 1 ...
  .. ..$ ref : num [1:7759] 6.02e+07 3.92e+08 6.02e+07 4.62e+08 1.82e+09 ...
  .. ..$ role: Factor w/ 14 levels "","admin_centre",..: 5 14 13 11 10 11 10 10
11 14 ...
  ..- attr(*, "class")= chr [1:3] "relations" "osmar_element" "list"
 - attr(*, "class")= chr [1:2] "osmar" "list"
```

Podemos observar que dentro del mapa de Coruña obtenido tenemos: nodos, vías y relaciones. Donde las relaciones son formadas por vías, que a su vez son formadas por nodos. Observamos también que cada nodo, vía y relación tiene un id que identifica ese elemento del mapa de forma única. Necesitamos los ids correspondientes a las rutas de autobuses para eso vamos a generar una lista con todas la relaciones de autobuses.

Como habíamos visto, uno de los elementos básicos de la cartografía de OSM son las **etiquetas** (*tags*) que se pueden asignar a nodos, caminos o relaciones. Esas etiquetas tienen de una clave (*key*) y un valor (*value*).

Vamos a utilizar la función find para buscar todos las relaciones del mapa que contengan una tag con valor igual a bus (v=bus), y lo guardaremos en bus_ids.

```
bus_ids <- find(coru, relation(tags(v == "bus")))

str(bus_ids)
```

```
num [1:32] 2222368 2266403 2269737 2219074 2269904 ...
```

Observamos que hemos obtenido 32 ids, lo que quiere decir que tenemos 32 líneas de autobuses identificadas en el mapa. Como no estamos seguros que todas la líneas están completas en ese trozo del mapa que tenemos, vamos a generar una lista de relaciones completas para cada una de las ids. Y luego vamos a guardar esa lista.

Podemos obtener la relación completa para cada una de las ids:

```
tmp<-get_osm(relation(2268886),full=TRUE

tmp$relations$tags

str(tmp)
```

Podemos obtener la relación completa para todas las ids obtenidas a la vez:

```
bus <- lapply(bus_ids, function(i) get_osm(relation(i), full = TRUE))

save(bus, file = 'osm_coru_bus.RData')
```

Vamos a abrir la lista que hemos generado y verificar que información tenemos sobre cada línea de bus.

```
load('osm_coru_bus.RData')

bus
```

```
[[1]]
osmar object
409 nodes, 67 ways, 1 relations

[[2]]
osmar object
445 nodes, 66 ways, 1 relations

[[3]]
osmar object
349 nodes, 47 ways, 1 relations

.
.
.
```

Nos quedamos con la primera línea de buses que R nos devuelve y vemos las relaciones y etiquetas que tenemos en esa línea.

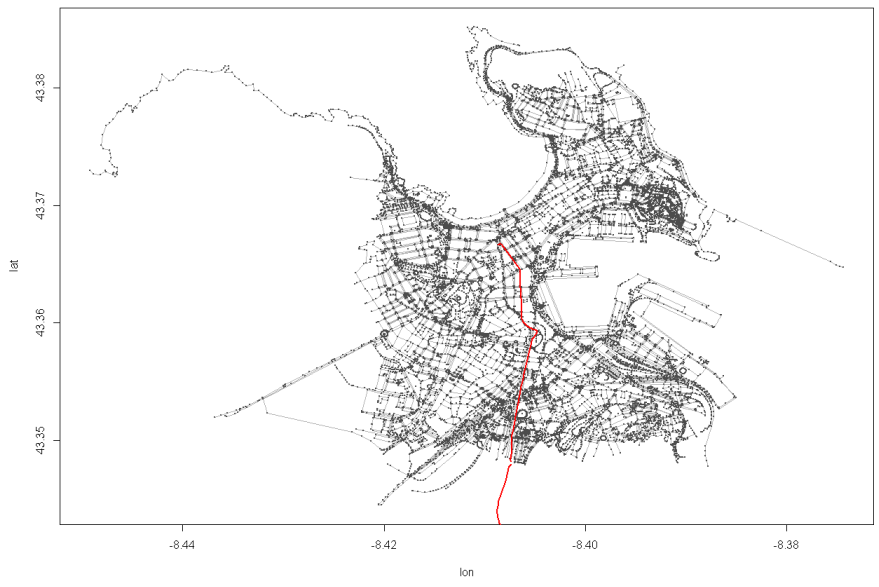
```
bus[[1]]$relations$tags
```

```
       id        k                                                  v
1 2222368     name Bus 11: HÃ©rcules => Ãrea comercial Marineda
2 2222368 operator     CompaÃ±Ã-a de TranvÃ-as de A CoruÃ±a, S.A
3 2222368      ref                                                 11
4 2222368    route                                               bus
5 2222368     type                                             route
```

Vamos a utilizar la función as_sp para convertir en un objeto de la clase sp, y luego vamos a dibujar ese objeto (en este caso nuestra línea de autobús E - Universidad) sobre el mapa de Coruña.

```
spl <- as_sp(bus[[1]], "lines")

plot(spl, lw=2, col = 'red', add = TRUE)
```



Y de esa manera podríamos obtener un plano de Coruña con todas sus líneas de buses en R.

En este trabajo hemos optado por utilizar la base de datos espaciales que nos ha proporcionado ayuntamiento de A Coruña, porque en el OpenStreetMap hemos encontrado 32 líneas de autobuses y actualmente la ciudad cuenta con 24. Eso probablemente se debe a los cambios de las líneas por obras que empezaron recientemente.

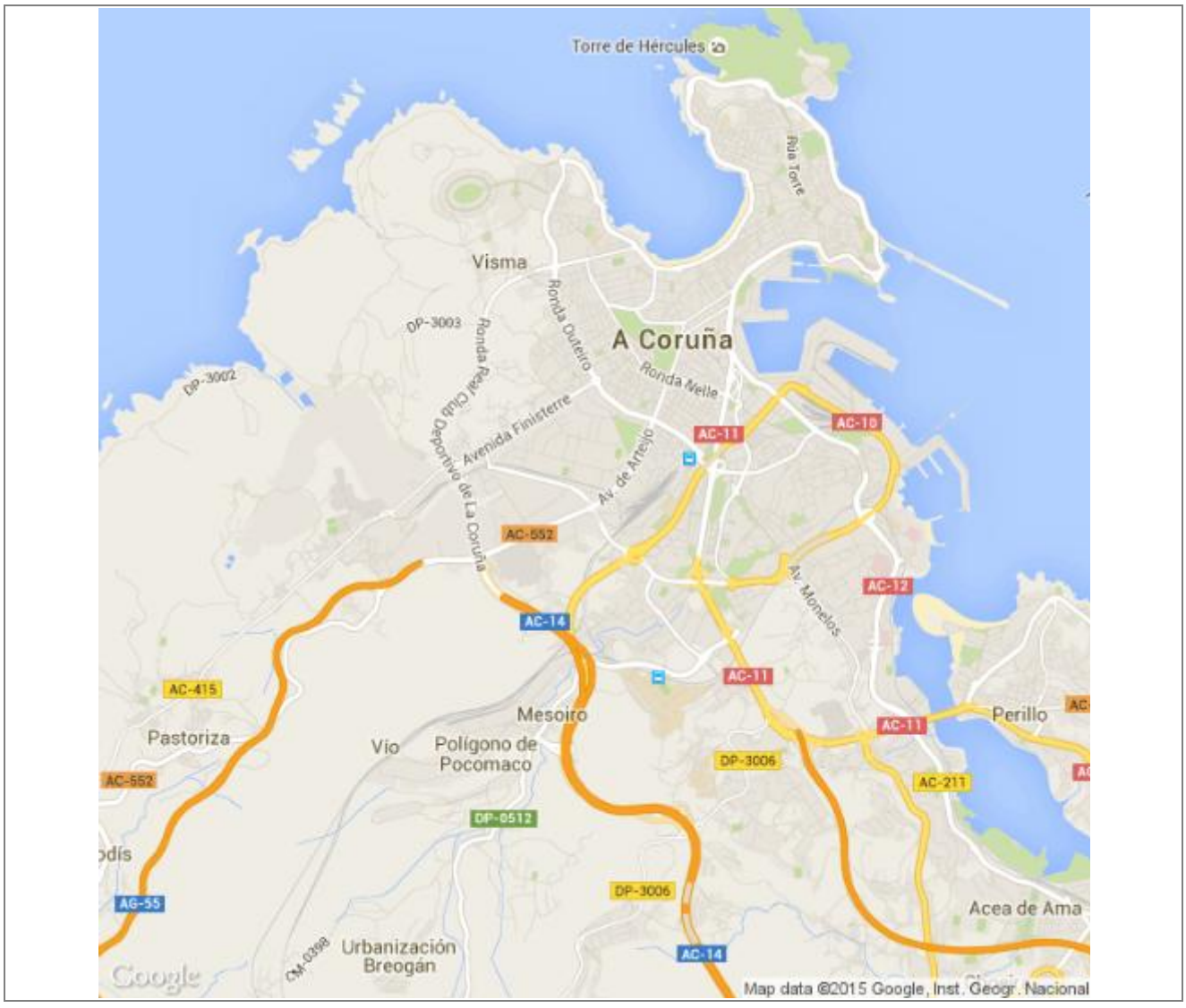
2.3.2. RGOOGLEMAPS

El paquete RgoogleMaps hemos utilizar solamente para dibujar el mapa de la ciudad de A Coruña en R.

```
library("RgoogleMaps")

 PlotOnStaticMap(GetMap(center = c(43.3490, -8.42), zoom =13, destfile =
"Coru.png", maptype = "mobile"), axes = TRUE)

#El zoom va de 0 a 19
```



Para verificar la latitud y longitud se pueden utilizar las páginas web:
http://tierra.tutiempo.net/Espana/La-Coruna-SP024636.html
http://www.coordenadas-gps.com/

Si queremos un imagen de satélite:

```
PlotOnStaticMap (GetMap (center = c(43.3490,-8.42), zoom = 13, destfile =
"Coru_satelite.png", maptype = "satellite"), axes = TRUE)
```

3. ANÁLISIS DE DATOS

Se comenzará realizando un pequeño análisis descriptivo que muestre las características principales de los datos de la base de datos tranvías.

Primeramente nos interesa conocer cuantas y cuales líneas de buses tenemos disponibles por la Ciudad en la actualidad, para eso utilizamos la tabla líneas. En la Tabla 1 observamos que hay 24 líneas.

1	Abente y Lago-Pablo Iglesias
2	Abente y Lago-Os Castros
3	Los Rosales-Adormideras
3A	Los Rosales-Adormideras
4	Hercules-B.Flores
5	Adormideras-Espacio Coruña
6	Hercules-San Jose
6A	Hercules-Bens
7	Hercules-Ventorrillo
2A	Abente y Lago-Hospital de Oza
11	Las Lagoas-Area C. Marineda
12	Los Rosales-Residencia
14	Los Rosales-Plz.Pablo Iglesias
12A	Ciudad Escolar-Residencia
17	Hercules-Residencia
B (BÚHO)	Obelisco - Los Castros
1A	Abente y Lago-Pasaje
20	P.Pontevedra-Pasaje
21	Juana Vega-Urbaniza. Mesoiro
22	P.Pontevedra-Pasaje
23	Juana de Vega-Breogan
23A	Abente y Lago-Breogan
24	P.Pontevedra-Valaire
E (Especial Universidad)	Juan Flórez, 10 - Campus Zapateira, Filología

Tabla 1. Líneas de autobuses actualmente en la ciudad de A Coruña

Ahora haremos resúmenes numéricos, tablas de frecuencia simples y gráficos para algunas variables de interés.

Obtenemos el siguiente resumen numérico de la tabla “Buses”:

Planta baja	Minusválido	Articulado
Si: 96	Si: 72	Si: 19
No: 36	No: 60	No: 113

Tabla 2. Resumen numérico de la tabla Buses

En la Tabla 2 se puede ver que de los 132 buses que tiene la compañía de tranvías, el 73% tiene planta baja, el 55% es para minusválidos y el 14% es articulado. Y están distribuidos de la siguiente manera entre las líneas:

	Líneas																						
	1	2	3	3A	4	5	6	6A	7	2A	11	12	14	12A	17	B	1A	20	21	22	23	23A	24	E
Articulado																								
Planta Baja																								
Minusvalidos																								

Tabla 3. Tipos de autobuses disponibles en cada línea. En amarillo tenemos buses articulados, en morado tenemos buses con planta baja y en verde tenemos buses para minusválidos.

En la Tabla 3 podemos observar que no tenemos autobuses para minusválidos en las líneas Búho y Especial Universidad, y que pocas líneas tienen autobuses articulados.

Como ya habíamos comentado anteriormente, en este trabajo nos vamos a centrar solamente en algunas variables de interés, en el mes de Febrero de 2014.

Otro detalle importante es que tenemos 2 tipos de información distinta en nuestra base de datos Tranvías, por un lado tenemos los pagos con tarjeta que nos permiten saber que perfil de usuario ha utilizado el servicio de tranvías, y por otro lado tenemos los otros tipos de pagos que no nos permiten saber nada sobre el usuario.

3.1. ANÁLISIS DESCRIPTIVO DE LOS DATOS DE VIAJES

En este apartado haremos un análisis descriptivo de los datos de todos los tipos de pagos.

	Hora	IdPar	Linea	Sentiida	Fecha	nceca	notros
1	2014-02-04 22:23:23	76	4	vuelta	2014-02-04	1	0
2	2014-02-04 22:24:23	194	4	vuelta	2014-02-04	0	0
3	2014-02-04 22:26:03	195	4	vuelta	2014-02-04	0	0
4	2014-02-04 22:26:31	196	4	vuelta	2014-02-04	1	0
5	2014-02-04 22:27:43	197	4	vuelta	2014-02-04	1	2
6	2014-02-04 22:28:35	167	4	vuelta	2014-02-04	0	0

Tabla 4. Variables de interés, datos todos los tipos de pagos, Febrero 2014

Es importante resaltar que cada fila de nuestra tabla representa una parada que un autobús hizo en un stop en el mes de Febrero de 2014. Como ya habíamos comentado anteriormente, siempre que hablemos de una parada física de autobús le llamaremos “stop”, y dejaremos la palabra “parada” para cuando un autobús para en un stop.

Obtenemos los siguientes gráficos de nuestra tabla de datos.

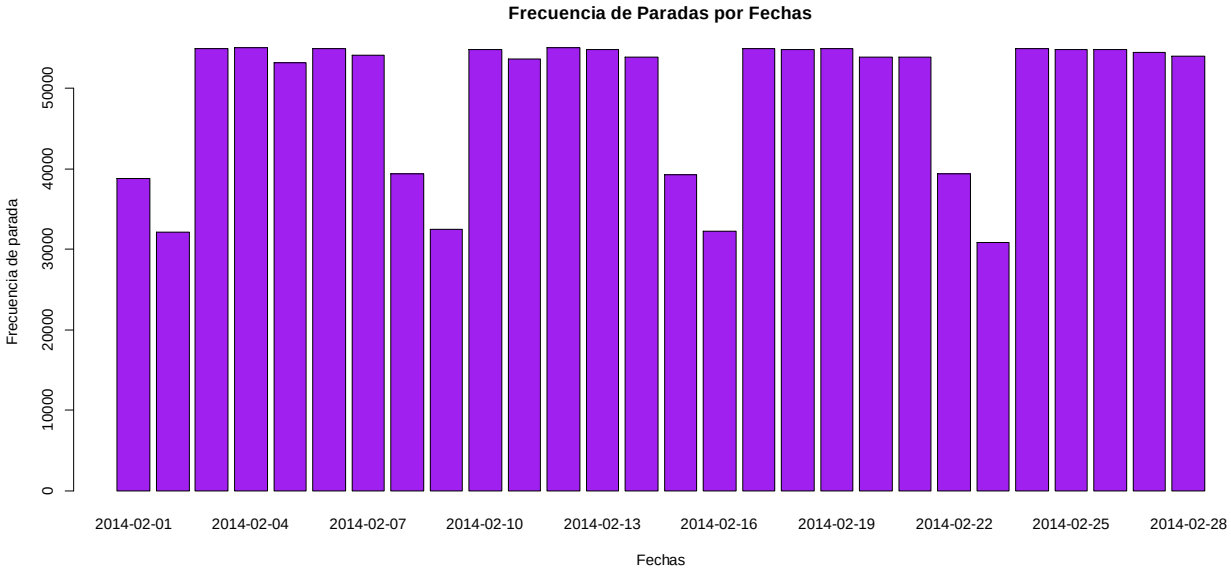


Figura 1. Gráficos de frecuencia de paradas por fechas

En la figura anterior podemos observar que tenemos un gráfico bastante homogéneo y con mucho sentido una vez que está muy claro que como de lunes a viernes se utilizan más buses que en los fines de semana hay mas paradas en los días laborables, y que los domingos son los días en que menos hay paradas, porque es el día de la semana que menos se utilizan autobuses.

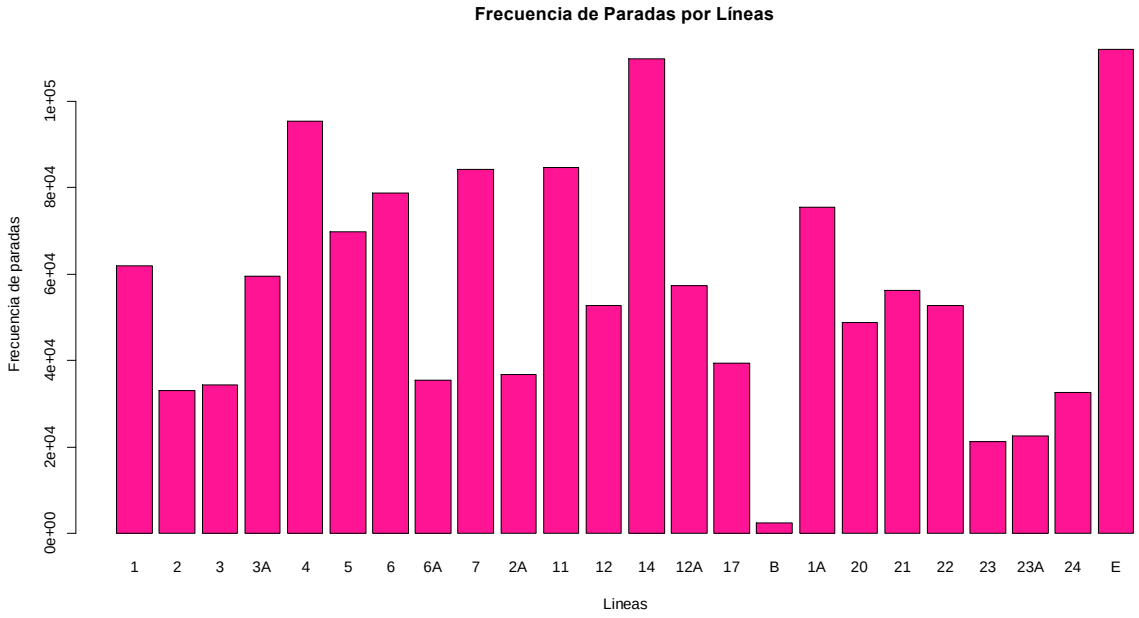


Figura 2. Gráficos de frecuencia de paradas por líneas

Podemos observar que las líneas 14 y Especial Universidad son las líneas con más paradas, pero como no está claro cuál de las dos tiene una frecuencia mayor, podemos hacer un resumen numérico (Figura 3) de esa variable que nos enseña que las líneas con más paradas son las E, 14 y 4 respectivamente. También se nota en el gráfico que la línea BÚHO es la que tiene menos paradas, seguida por las líneas 23 y 23A.

```
     Linea
E.Univ :111956
14     :109781
4      : 95405
11     : 84632
7      : 84322
(Other):871939
NA's   : 16000
```

Figura 3. Resumen numérico de la variable Linea que muestra el número total de paradas realizas en el período de tiempo de estudio.

Ahora vamos a verificar cuales son los stops más utilizados.

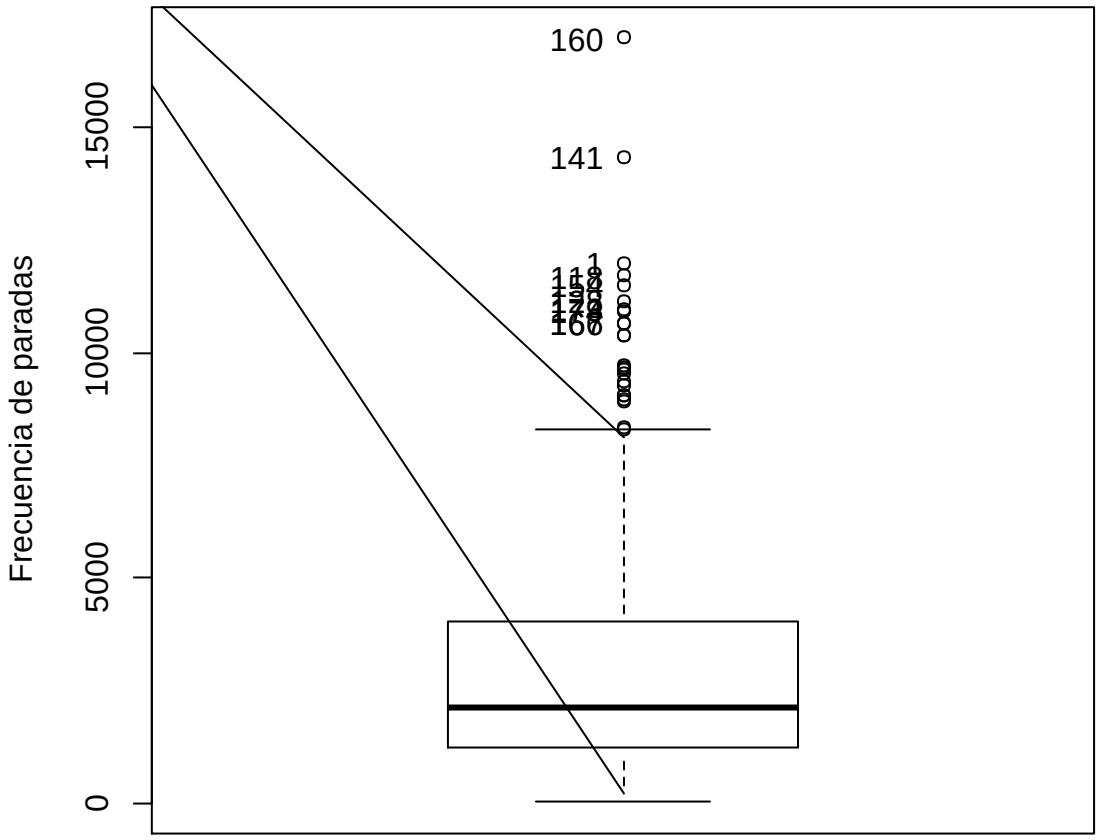


Figura 4. Gráfico de cajas de frecuencia de paradas en cada Stop

En la Figura 4 podemos observar claramente que el Stop 160 (Juan Florez, 10) es con diferencia donde los autobuses hacen más paradas, seguido por el 141 (Museo Militar).

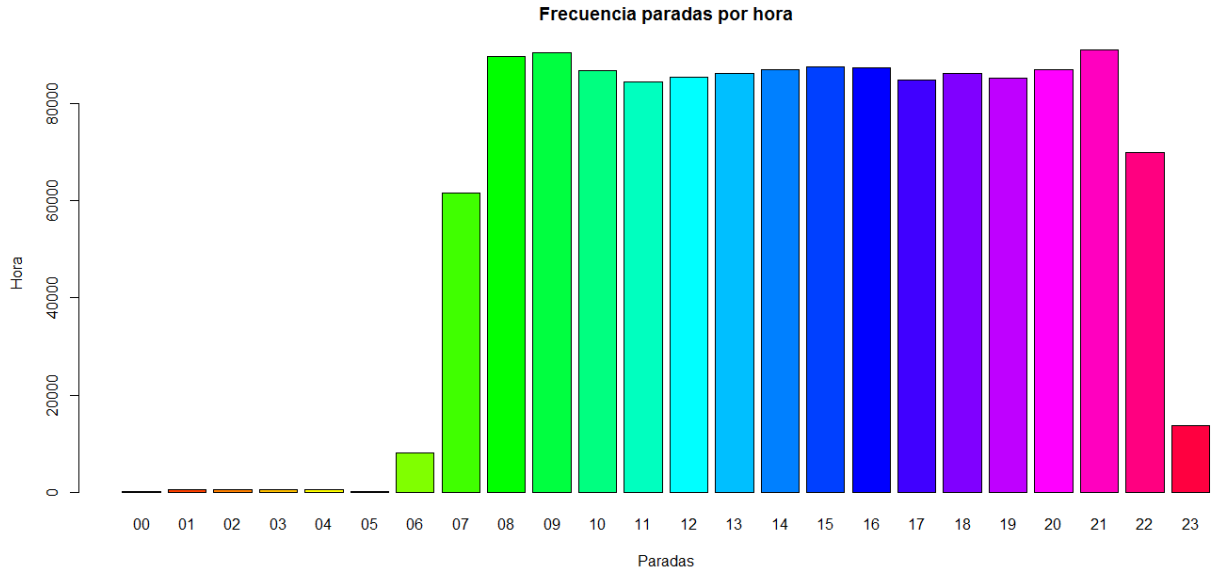


Figura 5. Gráficos de frecuencia de paradas por hora

En la Figura 5 podemos observar que como ya se esperaba de 00:00 a las 6:00 no hay prácticamente paradas ya que el único autobús disponible a esa hora es el Búho, y el número de usuarios es muy pequeño. También se nota que de 8 de la mañana hasta las 22 de la noche es el horario con más usuarios, siendo a las 9 de la mañana y a las 21 de la noche los horarios con más paradas.

3.2. ANÁLISIS DESCRIPTIVO DATOS TARJETA MILLENNIUM

Ahora que ya hemos estudiado la frecuencia de paradas en función de líneas, horarios y stops nos centraremos en estudiar el método de pago del billete de autobús. Como ya hemos comentado por un lado tenemos los pagos con tarjeta Millennium que nos permiten saber que perfil de usuario ha utilizado el servicio de tranvías, y por otro lado tenemos los otros tipos de pagos que no nos permiten saber nada sobre el usuario.

Para entender mejor porque tenemos un interés especial en los pagos con tarjetas, nos vamos a fijar en el número de viajeros que han utilizado los autobuses en Febrero 2014.

Total de billetes	Pagos con tarjeta Milenium	Otros tipos de pagos
1.633.179	1.343.991	289.188

Tabla 5. Total de billetes/viajeros Febrero 2014

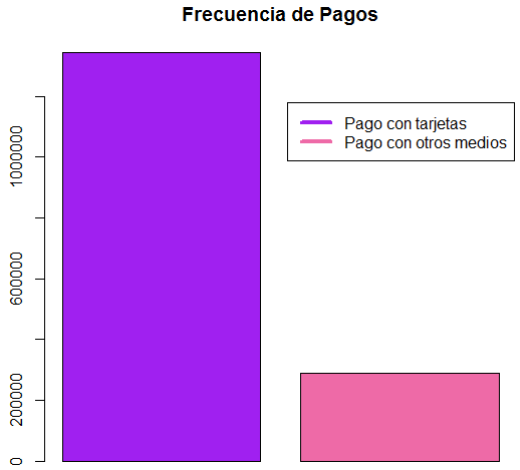


Figura 6. Gráficos de frecuencia por tipos de pagos

Podemos observar en la Figura 6 que los pagos con tarjeta Millennium representan un 82% del total, mientras que apenas el otro 18% se divide entre los otros tipos de pagos. Este dato hace que tengamos un interés especial en los pagos con Tarjeta Millennium, ya que la tarjeta contiene datos de tipos de perfil de usuario, parada de subida y línea, posibles transbordos, etc. Por lo tanto en este apartado haremos un análisis descriptivo de los pagos solamente con tarjeta Millennium (CECA).

	Fecha	Tarjeta	IdPerfil	IdTipoBillete	IdPar	Linea	Sentiida	nceca
1	2014-02-01	9.8701e+15	Desempleado	Normal	20	14	vuelta	5
2	2014-02-01	9.8701e+15	Pensionista	Normal	20	14	vuelta	5
3	2014-02-01	9.8701e+15	Jubilado	Normal	9	14	vuelta	3
4	2014-02-01	9.8701e+15	General	Normal	9	14	vuelta	3
5	2014-02-01	9.8701e+15	General	Normal	9	14	vuelta	3
6	2014-02-01	9.8701e+15	General	Transbordo	119	14	vuelta	1

Tabla 6. Variables de interés, datos pagos con tarjeta Millenium, Febrero 2014

Es importante que nos recordemos que cada fila de nuestra tabla representa una operación con tarjeta en el mes de Febrero de 2014.

Obtenemos los siguientes gráficos de nuestra tabla de datos.

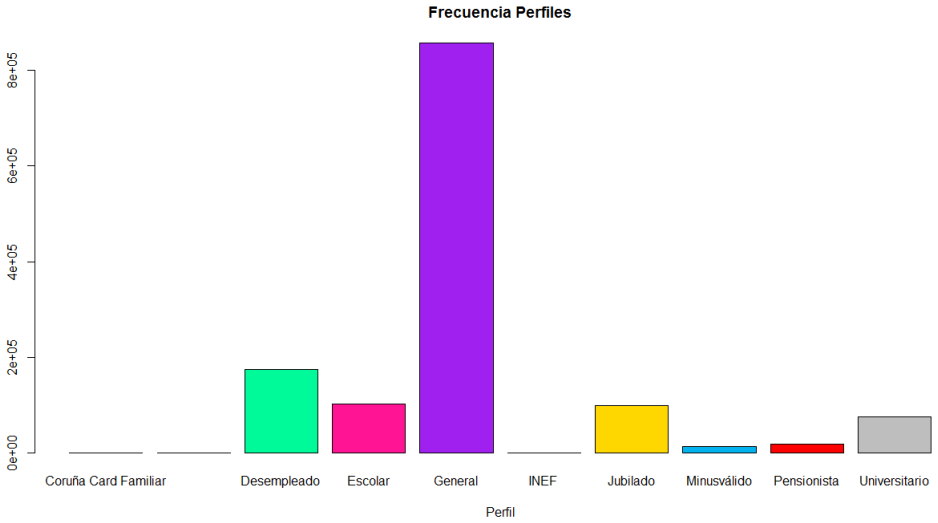


Figura 7. Gráficos de frecuencia por perfil de tarjeta Millennium

En la Figura 7 tenemos que la mayor parte de los usuarios de tarjetas Millennium son los que tienen un perfil general, seguidos de los desempleados y escolares respectivamente.

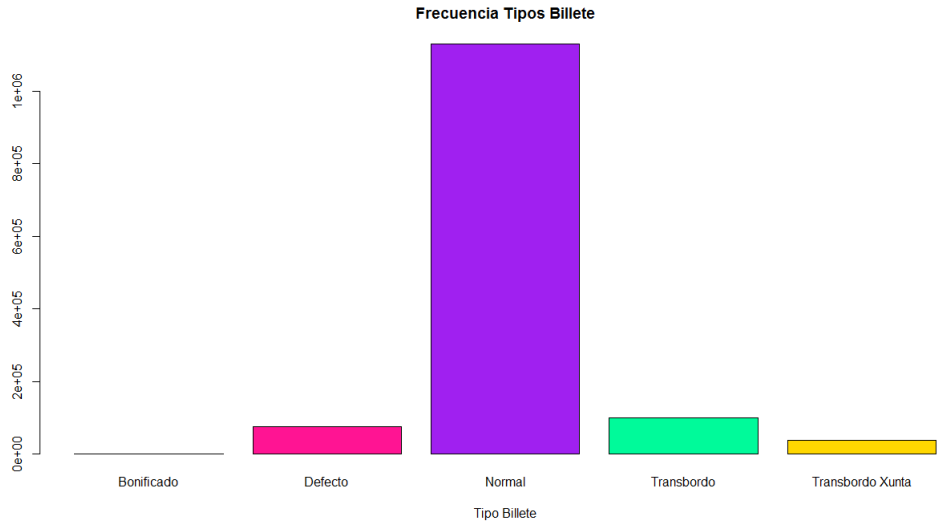


Figura 8. Gráficos de frecuencia por tipos billetes pagos tarjeta Millennium

La mayor parte de los usuarios hacen viajes normales, seguidos por los usuarios que hacen transbordo.

En este punto realizaremos análisis bivariante, para eso utilizaremos tablas cruzadas, también llamadas tablas de contingencia. Son tablas construidas en base a dos variables: una variable fila y una variable columna. A partir de esta disposición de los datos se pueden establecer relaciones entre estas dos variables.

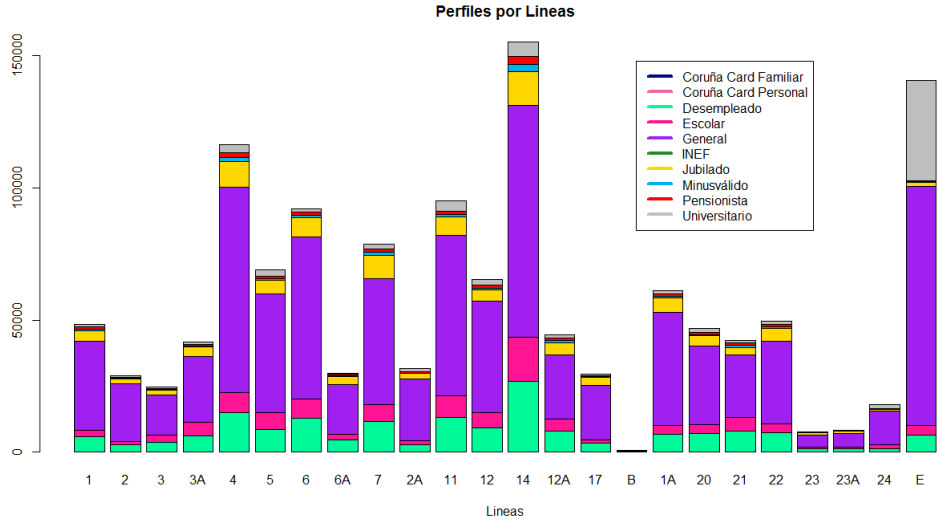


Figura 9. Gráficos de frecuencia de paradas por perfiles y línea

En la Figura 9 Perfiles por línea podemos observar que la línea 14 es la línea con más usuarios desempleados, jubilados, escolares, minusválidos y pensionistas, esto puede deberse a que es la línea con más usuarios. La línea Especial Universidad tiene el mayor número de universitarios, como ya era de esperar.

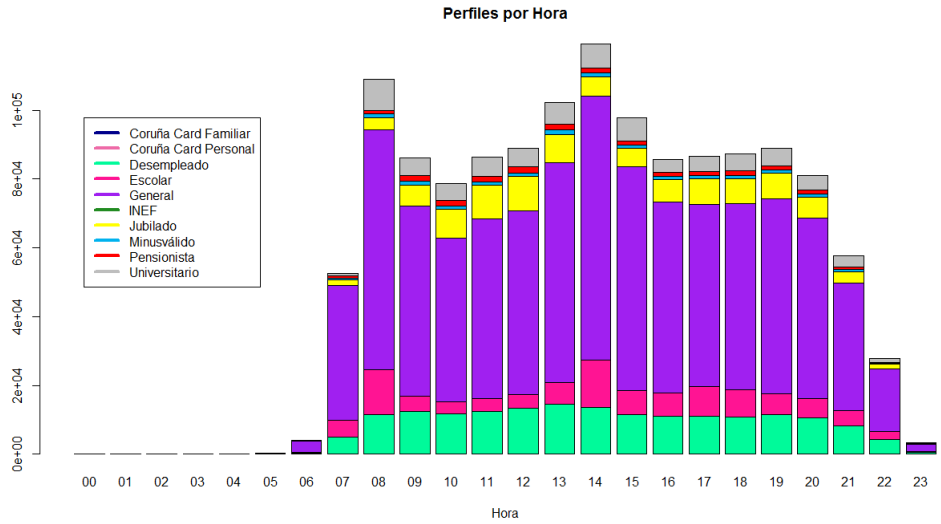


Figura 10. Gráficos de frecuencia de paradas por perfiles y hora

En la Figura 10 observamos que los desempleados utilizan más los buses de 13:00 a 14:00, mientras que los universitarios utilizan más de 08:00 a 09:00, los jubilados entre las 11:00 y las 13:00 y los escolares de 08:00 a 9:00 y 14:00 a 15:00.

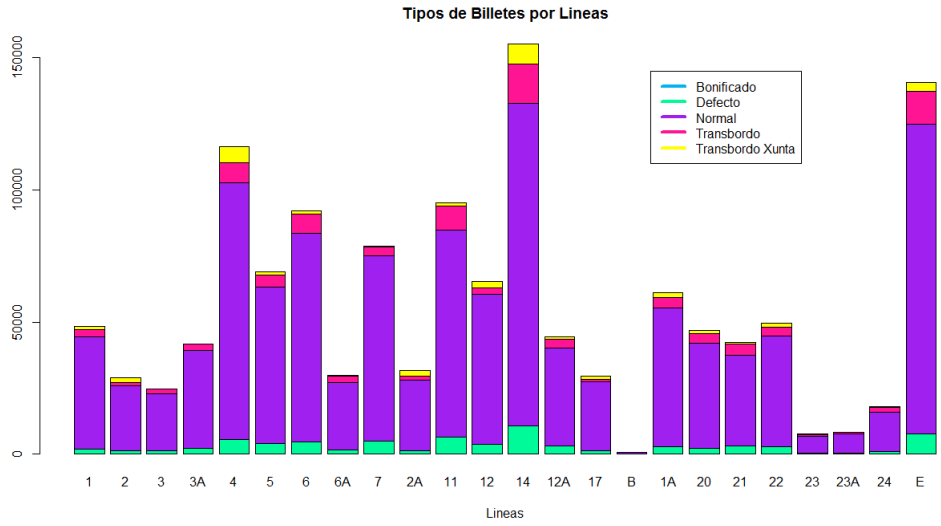


Figura 11. Gráficos de frecuencia de paradas por Tipos de billetes y líneas

En el grafico anterior, Tipos de billetes por líneas, observamos que la línea 14 es la línea con más transbordos y defectos seguida de la línea Especial Universidad.

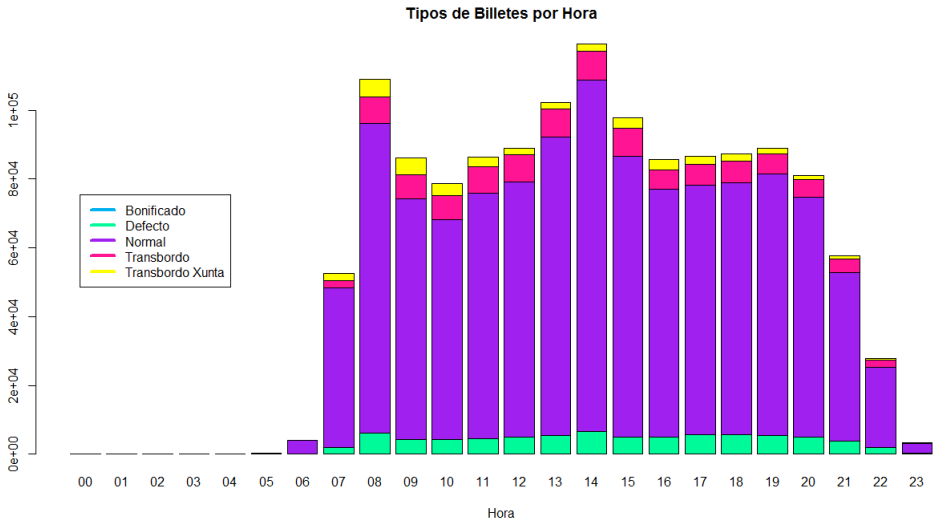


Figura 12. Gráficos de frecuencia de paradas por Tipos de billetes y hora

En el grafico Tipos de billetes por hora observamos que los horarios con menos transbordos son de 07:00 a 08:00 y de 22:00 a 23:00.

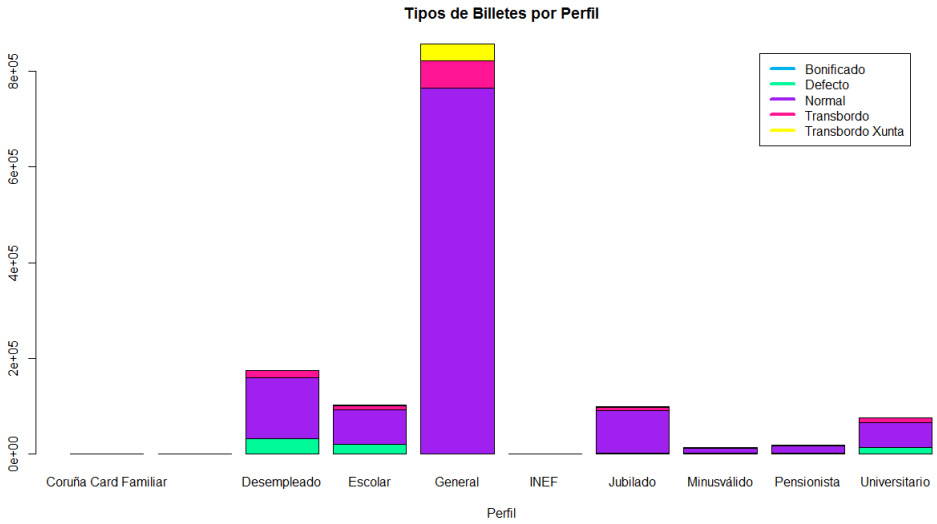


Figura 13. Gráficos de frecuencia de paradas por Tipos de billetes y perfil

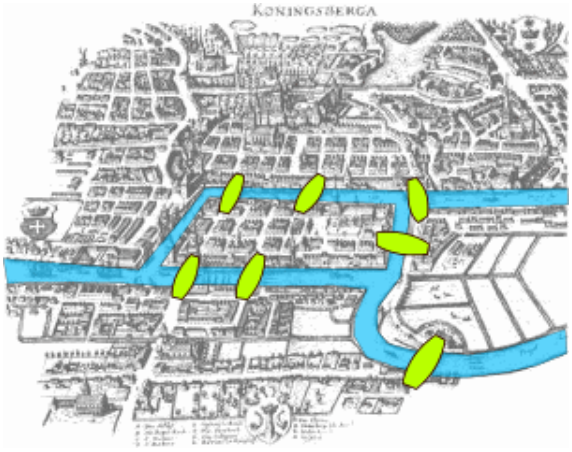
En la Figura 13 podemos observar que dentro del el perfil de billetes general (más utilizado), la mayoría de los billetes son del tipo normal, seguido del tipo transbordo y transbordo Xunta. La mayor parte de los billetes con defecto son del perfil desempleado, seguido por escolar y universitarios.

3.3. ANÁLISIS DE DATOS DE RED

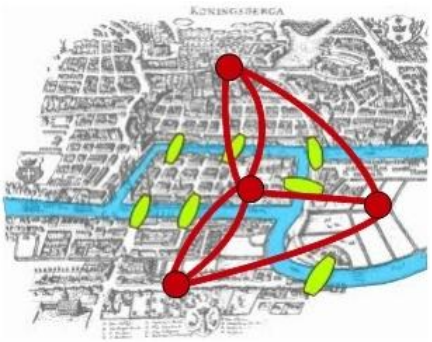
El análisis de datos de redes es, probablemente, una de las zonas de mayor interés actualmente, porque ahora más que nunca, las redes forman parte no solo hacen parte del ámbito de la investigación y la metodología, sino también de la vida cotidiana a través de Internet, redes sociales, redes de transporte, redes eléctricas y hasta redes biológicas de neuronas y redes de salud pública. En consecuencia, hay una necesidad crítica del análisis de los datos contenidos en esas redes, que luego se puede utilizar con diferentes objetivos.

La representación matemática de las redes son los grafos.

Se puede considerar que el origen de la teoría de grafos fue en 1736, año en el que Euler publicó una obra en el que resolvía el problema de los puentes de Königsberg, ciudad alemana de la Prusia Central en esos años, y hoy **Kaliningrado** perteneciente a Rusia por donde pasa el **río Pregel** que se bifurcaba en varios ramales formando dos islas antes de seguir su curso de nuevo. Para cruzar de una isla a otra el pueblo contaba con 7 puentes y el problema era encontrar una ruta por la que cruzando una sola vez cada puente se pudiese regresar al punto de partida.



Para resolver el problema, Euler, sustituyó cada uno de los trozos de tierra firme por un punto (nodo) y cada puente por un trazo (arcos), dando lugar a un esquema simplificado que actualmente conocemos como un grafo. Euler probó que no había camino alguno con esas propiedades.



Con frecuencia tenemos informaciones adicionales sobre los nodos y arcos, como por ejemplo la distancia de un nodo a otro, o la localización geográfica de cada nodo. Lamamos a esa información adicional atributos.

Vamos a representar y analizar la red de autobuses de A Coruña mediante un grafo, donde cada stop es un nodo y el recorrido entre dos stops es un arco. Por lo tanto tendremos 450 nodos y 1082 aristas. Nuestro grafo será dirigido y las direcciones de los de los arcos vienen dadas por en sentido (ida y vuelta) de cada línea de autobuses.

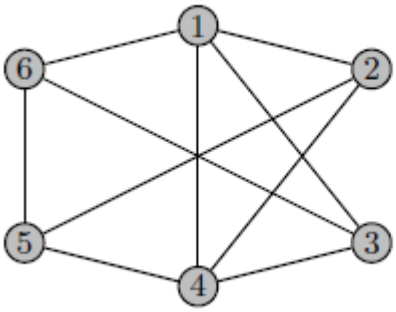
En este trabajo no vamos a entrar en los detalles de la teoría de los grafos, apenas daremos algunas definiciones necesarias para entender conceptos que vamos a necesitar para los análisis que vamos a hacer. El objetivo a continuación es explorar características y propiedades estructurales de nuestra red.

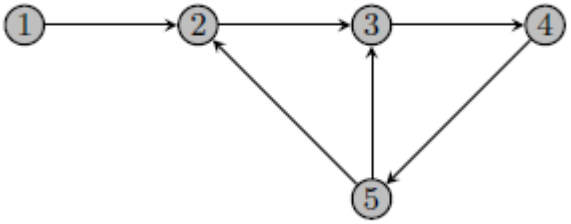
3.3.1. DEFINICIONES:

Grafo es un par $G = (V,A)$ donde V es el conjunto de vértices o nodos y $A \subset V \times V$ es el conjunto de arcos o aristas.

Una *red* es un grafo con uno o más números asociados con cada arco o nodo. Estos números pueden representar distancias, costes, fiabilidades u otros parámetros de interés.

Un grafo puede ser orientado o no orientado.





Un grafo *orientado* o *dirigido* es aquel en el que los arcos son pares ordenados; el arco (*i, j*) empieza en el nodo *i* y termina en el nodo *j*. En este caso (*i, j*) ≠ (*j, i*).

En un grafo *no orientado*, (*i, j*) y (*j, i*) representan el mismo arco. En principio, no consideraremos grafos en los que hay lazos, es decir, arcos de la forma (*i, i*).

Dos vértices i y j son adyacentes si existe una arista que los conecte.

Un *subgrafo* de un grafo *G* es un grafo $G`= (V´, A`)$ que tiene todos sus vértices y aristas en *G*, es decir, $V´ \subseteq, V \quad A´ \subseteq, A.$

El *grado* de un nodo cualquiera de un grafo *G* es el número de aristas incidentes con él. En un grafo dirigido se distingue entre grado interior o de entrada de un vértice (número de aristas que llegan a él) y grado exterior o de salida (número de aristas que salen de él).

Un grafo es *completo* si todas las aristas posibles están presentes. En el caso de un grafo orientado el máximo número de aristas es $n(n-1)$ (no permitimos lazos, por eso no es n^2) y, en el caso de un grafo no orientado, tendremos la mitad, $n(n-1)/2$.

Se dice que un grafo está conectado si existe como mínimo un camino entre cualquier par de vértices distintos, o sea, si no hay nodos aislados.

3.3.2. VISUALIZAR REDES

Hay muchas maneras de representar un grafo, en general, se utilizan técnicas de visualización de datos de red, de manera a conseguir un grafo que visualmente represente de la mejor manera posible las informaciones que tenemos sobre esa red. Entre los métodos para visualizar redes están:

- Método del Círculo que dispone los nodos en un círculo y los une con líneas que representan los arcos.

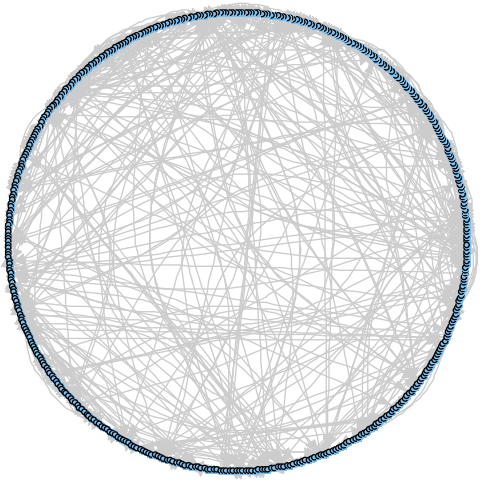


Figura 15. Gráfico de la red de autobuses de A Coruña utilizando el Método del Círculo.

- Método Kamada y Kawai que dispone los nodos con la misma longitud y de manera que haya pocos cruces entre los arcos.

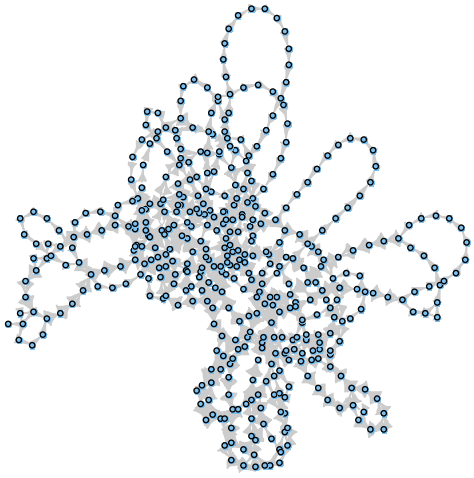


Figura 16. Gráfico de la red de autobuses de A Coruña utilizando el Método Kamada y Kawai.

- Método de Fruchterman y Reingold que hace algo similar al método de Kamada y Kawai pero utiliza una función distinta para optimizar.

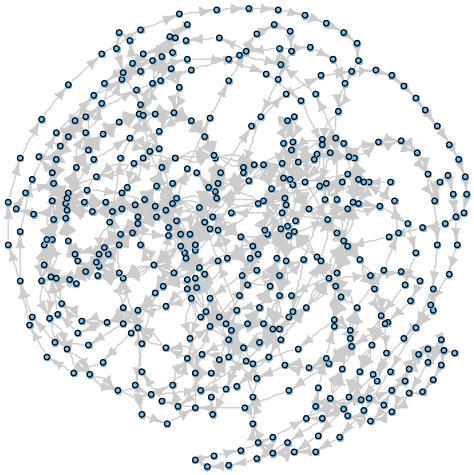


Figura 17. Gráfico de la red de autobuses de A Coruña utilizando el Método de Fruchteman y Reingold.

- Método DRL que agrupa los nodos. Ese método es utilizado para visualizar redes grandes.

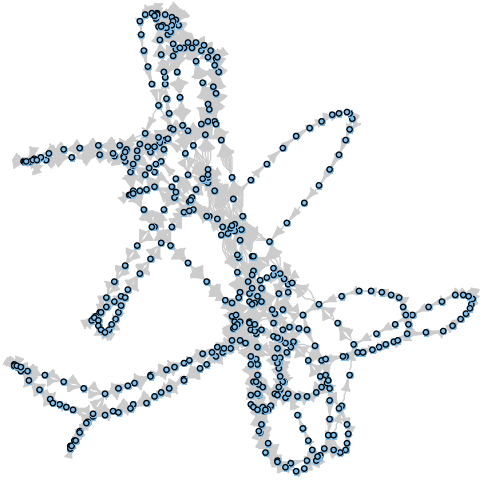


Figura 18. Gráfico de la red de autobuses de A Coruña utilizando el Método DRL.

El paquete igraph de R en la función plot.igraph tiene una opción layout.auto, que determina de manera automática que procedimiento es el más adecuado para dibujar un grafo. En el caso de nuestra red, según esa función, el procedimiento más adecuado para dibujar nuestro grafo sería el método de Fruchterman y Reingold.

Se puede encontrar más información sobre esos métodos en el capítulo 3 del libro “Statistical analysis of network data with R”. En este trabajo no vamos a profundizar en este tema, porque consideramos preferible utilizar como grafo el mapa de la red de transporte de autobuses que hemos construido a través de objetos espaciales con el paquete *sp* de R, sobre la base de datos GTSF cedida por el ayuntamiento de A Coruña.

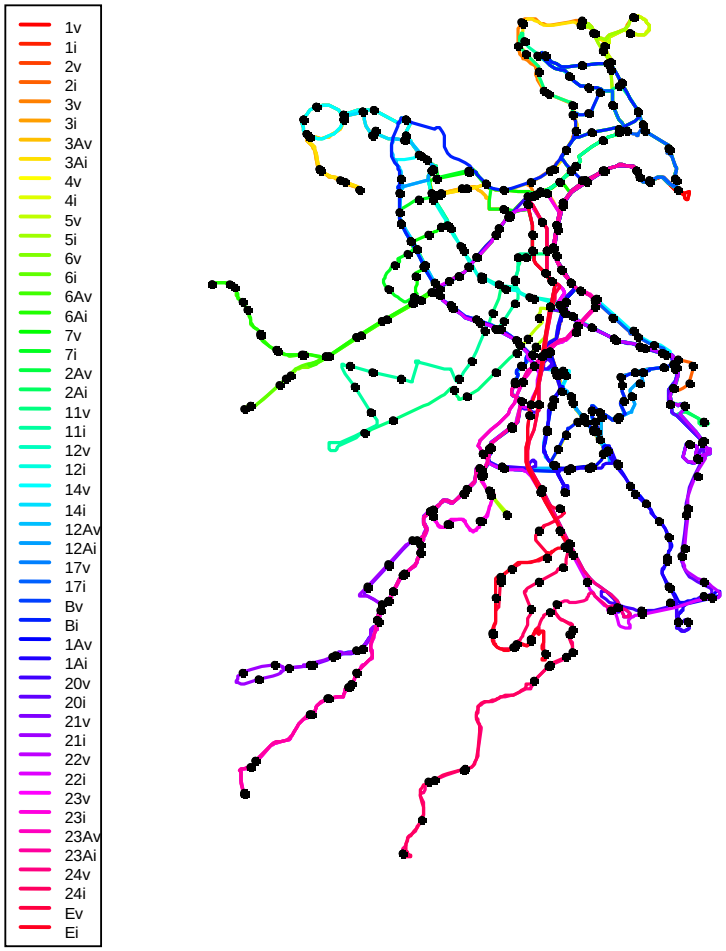


Figura 19. Grafo de la red construido a través de objetos espaciales utilizando la base de datos GTSF

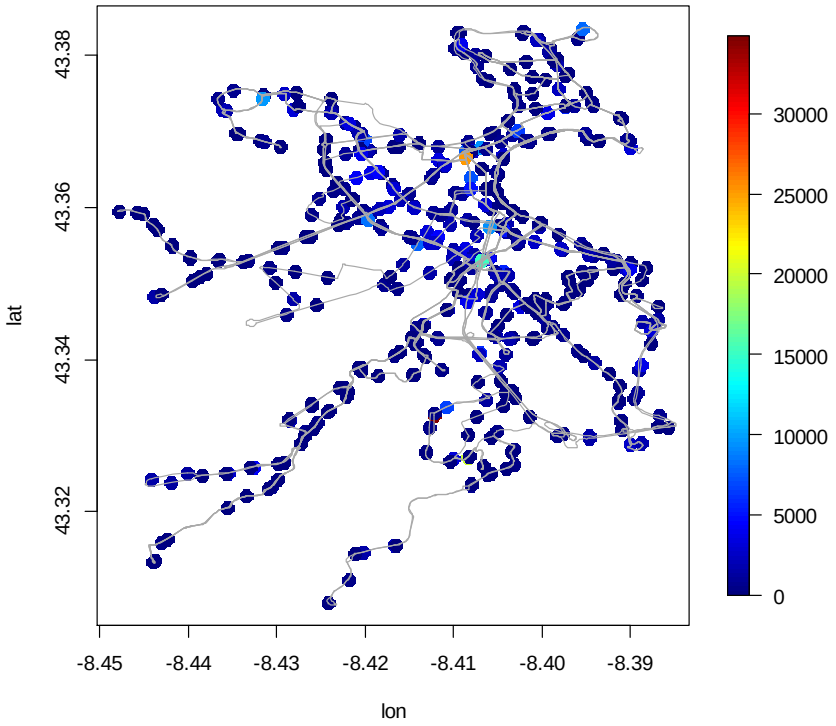


Figura 20. Grafo con los nodos de la red (stops) coloreados según el total de billetes.

3.3.3. Análisis Descriptivo de las características de la red

Lo primero que se debe hacer es estudiar la conectividad de la red, para eso vamos a verificar si nuestra red contiene o no subredes (o subgrafos) y si hay nodos aislados o si la red está conectada. Como ya esperábamos nuestra red está conectada. Como tenemos la red de autobuses urbanos de A Coruña, no tendría sentido tener stops aislados, o sea stops por donde no pasan las líneas de autobuses.

Vamos a dividir el análisis en dos partes, en principio vamos a estudiar toda la red, y luego estudiaremos las sub redes formadas por cada una de las 3 líneas de autobuses más utilizadas.

Sabemos que en un mismo stop paran autobuses de líneas distintas. Al estudiar el grafo de toda la red tendremos los atributos de los nodos sin considerar las líneas y al estudiar las sub redes (cada línea de autobuses), tendremos los atributos de nodos considerando solamente esa línea.

- Atributos de los stops (nodos):
    - Frecuencia de autobuses en que han parado en ese stop,
    - Total de usuarios que han pagado con la tarjeta Millennium
    - Total de usuarios que han utilizado medios de pagos distintos a la tarjeta Millennium
    - Total de viajeros que han subido a los buses en ese stop
    - Total de usuarios que han pagado con la tarjeta con perfil desempleados
    - Total de usuarios que han pagado con la tarjeta con perfil escolar
    - Total de usuarios que han pagado con la tarjeta con perfil general
    - Total de usuarios que han pagado con la tarjeta con perfil jubilado
    - Total de usuarios que han pagado con la tarjeta con perfil pensionista
    - Total de usuarios que han pagado con la tarjeta con perfil universitario
    - Total de pagos tipo “defecto” con tarjeta
    - Total de pagos tipo “normal” con tarjeta
    - Total de pagos tipo “transbordo” con tarjeta

- Atributos de los recorridos entre los stops (arcos):
    - Distancia (km)
    - Tiempo (minutos)
    - velocidad media (Km/h)

Al utilizar los atributos de la base de datos tranvías y construir la red a partir de los datos GTSF aparecieron algunos problemas, como por ejemplo stops que existían en una base de datos y no existían en la otra, tramos entre stops de 5m (probablemente es el mismo stop que fue desplazado), entre otro. Debido a este contratiempo se decidió eliminar posibles datos atípicos y para eso, como criterio general, en las variables relativas a tramos en las que aparentemente ocurría esto, se eliminaron las observaciones menores que la mediana menos 1.5 veces el recorrido intercuartílico (empleando la función boxplot de R).

Se pueden plantear muchas de las preguntas sobre un vértice de una red, pero lo más importante es tratar de entender su “importancia” dentro la red.

Muchas medidas de centralidad fueron propuestas través de los años para cuantificar tal "importancia". Pero posiblemente la medida de centralidad más utilizada es el grado de los vértices. Aquí centraremos nuestro análisis en torno a tres tipos clásicos de centralidad vértice: cercanía, intermediación y status. Para

eso vamos a utilizar la distancia geodésica entre dos nodos que es la longitud del camino más corto que los une. Es importante resaltar que cuando un grafo es dirigido tenemos que distinguir la distancia de i a j y la distancia de j a i.

Medidas de centralidad por cercanía

Medidas de centralidad por Cercanía utilizan idea de que un nodo es "central" si es adyacente a muchos otros nodos. El enfoque estándar, es que la centralidad varía inversamente con la total distancia de un nodo de todos los demás.

Medidas de centralidad por intermediación

En las medidas de centralidad por intermediación un nodo es "importante" si sus aristas son las que más se utilizan en los caminos más cortos entre los demás nodos.

Medidas de centralidad por status

Medidas de centralidad por status, utilizan la idea de que cuantos más vecinos centrales tenga un nodo, más central será ese nodo.

Aplicando a nuestra red estas tres medidas de centralidad obtenemos que el stop más “importante” por cercanía considerando los grados interiores o de entrada de los nodos es el 9 (Caballeros Estación), por intermediación es el 129 (San Pedro Mezonzo) y por status es el 7(Primo de Rivera, 1).

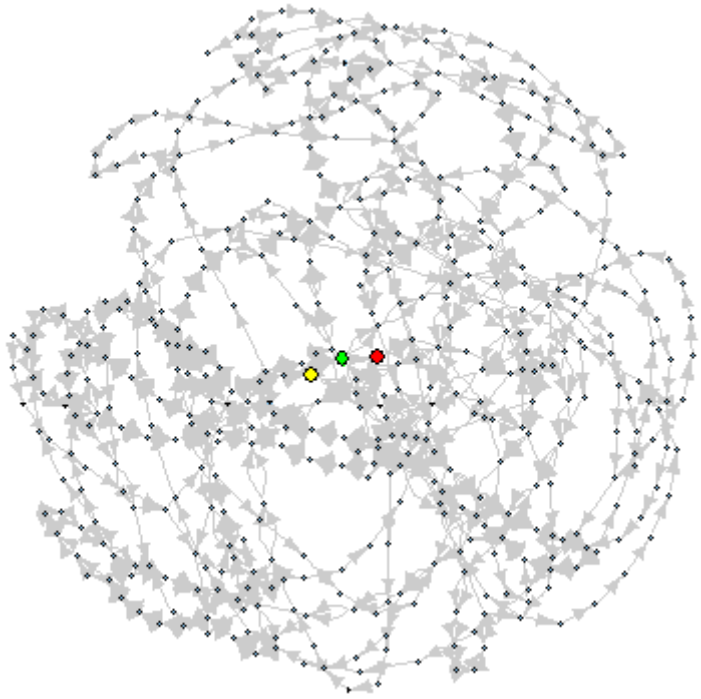


Figura 21. Gráfico de la red de autobuses de A Coruña con los puntos centrales: 9 (Caballeros Estación) en rojo, 129(San Pedro Mezonzo) en verde y 7 (Primo de Rivera, 1) en amarillo.

Empleando el paquete plotGoogleMaps, podemos representar estos stops en un mapa de google interactivo.

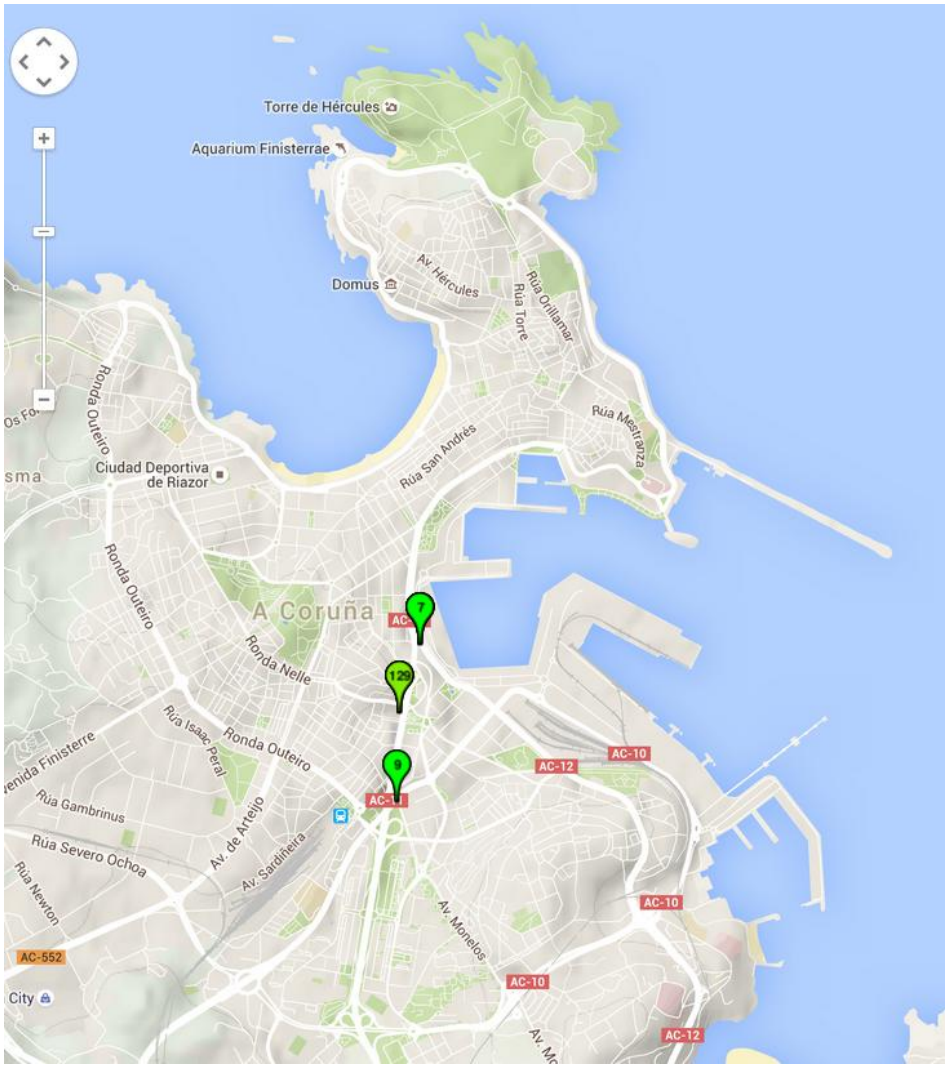


Figura 22. Mapa de A Coruña con los puntos centrales de la red: 9 (Caballeros Estación), 129 (San Pedro Mezonzo) y 7 (Primo de Rivera, 1).

Igual que para los vértices, también se pueden plantear preguntas sobre los arcos, y tratar de entender su "importancia" dentro la red. Una manera de medir esa "importancia" es por centralidad por intermediación, en donde un arco es "importante" si es el que más veces aparece en los caminos más cortos entre los nodos.

En el análisis de redes, un gran número de preguntas se reducen a cuestiones sobre la cohesión de la red. Hay muchas maneras de definir la cohesión de la red, podemos definirla, dependiendo del contexto de la pregunta.

Por ejemplo, en una red social, al hacerse amigo de un determinado actor esa persona tende a ser amigo de los otros amigos de ese actor? ¿Qué proteínas en una célula parecen trabajar estrechamente juntas? ¿Las páginas web tienden a separar su estructura con respecto a los distintos tipos de contenido?

Lo primero que hicimos en nuestro análisis descriptivo fue estudiar la conectividad de la red verificamos si nuestra red contiene o no sub redes (o subgrafos) y si había nodos aislados o si la red estaba conectada. La

conectividad es una posible caracterización de la cohesión de la red. Por ejemplo, nos podemos preguntar si dos stops que tienen conexión con un tercero tienen a su vez conexión entre ellos. También nos podemos preguntar si hay grupos de stops aislados respecto de otros.

Podemos medir la cohesión de una red por densidad o por detección de comunidades.

La densidad de una red es el cociente entre los arcos existentes y los arcos posibles, es siempre un número entre 0 y 1 que mide como de cerca está la red de tener tantos ejes como sea posible.

Detección de comunidades, es mirar si los nodos de la red forman grupos entre ellos, se buscan grupos disjuntos cuyos nodos tengan una cierta cohesión con respecto a las relaciones entre ellos. Más concretamente, buscamos grupos en los cuales los nodos son más intensamente conectados entre sí que con el resto da rede, y están relativamente bien separados del resto de nodos en otros grupos.

Existen numerosas técnicas que se han propuesto para el problema de detección de comunidades, que difieren principalmente en la forma en que evalúan la calidad de los agrupamientos propuestos y los algoritmos por los que buscan optimizar esa calidad. Esos algoritmos a su vez pueden ser divididos en varios tipos de métodos.

Sin embargo, la mayoría de ellos presentan dos problemas importantes: están diseñados para redes no dirigidas; y tienen un coste computacional muy alto para redes muy grandes.

En la librería igraph podemos encontrar hasta ocho algoritmos diferentes. Entre ellos están:

1) Algoritmo de Newman-Girvan (Newman and Girvan, 2004);

Como hemos visto, la cercanía por intermediación de un eje mide el número de caminos más cortos que pasan por dicho eje. La idea principal de ese algoritmo está basada en esperar que ejes que conecten comunidades diferentes tengan un alto valor de la intermediación ya que todos los caminos más cortos deben pasar por dicho eje. Por lo tanto, si eliminamos el eje con mayor cercanía por intermediación dividiremos la red en dos sub-redes, que a su vez se pueden dividir de manera similar. Es un algoritmo de tipo jerárquico y podemos obtener un dendograma. Sin embargo, computacionalmente es bastante costoso.

2) Algoritmo de Pons-Latapy (Pons and Latapy, 2005)

La idea principal de ese algoritmo está basada en encontrar grupos que estén densamente conectados mediante el uso de paseos aleatorios. Se espera que los nodos que recorren paseos aleatorios estén dentro de la misma comunidad.

Este algoritmo ignora la dirección de los ejes al buscar los grupos. Sin embargo, es extremadamente rápido incluso en redes de un tamaño muy grande. El problema está en la elección de la longitud de los paseos aleatorios utilizados. La elección habitual es 4, pero si aumentamos este valor, el número de comunidades detectadas se puede reducir bastante.

Aplicando a nuestra red el Algoritmo de Newman-Girvan, obtenemos que nuestros arcos más centrales son:

52 -> 53 (Ronda Nelle, Caixa -> Ronda Nelle, 9)
53 -> 129 (Ronda Nelle, 9 -> San Pedro Mezonzo)
26 -> 437 (Plz. Mina -> Obelisco)
437 -> 554 (Obelisco -> P. Maritimo, Cancela)
382 -> 73 (S. Madariaga,Mercado -> Caballeros, 33)

Ahora nos plantearemos algunas preguntas descriptivas sobre la red.

¿Qué stop tiene menor frecuencia de paradas?	16 (Abegondo, 4) y 17 (Castrillón, 39)
¿Qué stop tiene una mayor frecuencia de paradas?	181 (Hercules, Mercado)
¿En qué stop se suben más pasajeros?	186 (Plz Pontev, Femenino)
¿En qué stop se suben mas pasajeros que utilizan otros tipos de pago que no la tarjeta?	186 (Plz Pontev, Femenino)
¿En qué stop se suben más pasajeros que pagan con la tarjeta millennium?	186 (Plz Pontev, Femenino)
¿En qué stop se suben más pasajeros que están haciendo transbordo?	160 (Juan Florez, 10)
¿En qué stop se suben más universitarios?	460 (Intercampus,Gta.Elvi)
¿Cuál es el segundo stop en qué se suben más universitarios?	462 (Maestranza, 5)
¿En qué stop se suben más desempleados?	186 (Plz Pontev, Femenino)
¿En qué stop se suben más jubilados?	186 (Plz Pontev, Femenino)
¿En qué stop se suben más minusválidos?	396 (Ramón y Cajal, 30)
¿En qué stop se suben más pensionistas?	185 (San Andres, Africano)
¿En qué stop se utilizan más tarjetas con defecto?	460 (Intercampus,Gta.Elvi)
¿Cuál es el tramo con menor distancia?	90 -> 164, con 0.07483969 km (E. Gonzalez, 54 -> Portiño)
¿Cuál es el tramo con menor tiempo?	167 -> 168, 0.2 min (San Andres, 121 -> San Andres, 93)
¿Cuál es el tramo con menor velocidad media?	550 -> 212, con 0.02475532 km/h (Area C. Marineda -> C. Carballo, Iveco)
¿Cuál es el tramo con mayor distancia?	431 -> 432, con 1.38 km (Nueva York, Casino -> Nuev York, Rialta)
¿Cuál es el tramo con mayor tiempo?	550 -> 212, con 12.46 min (Area C. Marineda -> C. Carballo, Iveco)
¿Cuál es el tramo con mayor velocidad media?	279 -> 281, 70 km/h (A. Molina, Elviña -> A. Molina, M. Grande)

3.4. ANÁLISIS DE DESCRIPTIVO DE LAS LÍNEAS

De acuerdo con nuestro análisis descriptivo clásico anterior, sabemos que las líneas más utilizadas son 14, E, 4, respectivamente. Vamos a estudiar el subgrafo que representa cada una de estas líneas, nos vamos a plantear algunas preguntas sobre cada una de ellas. También haremos un gráfico (análisis descriptivo espacial) para cada una de estas tres líneas coloreando los arcos de acuerdo con una de las variables asociadas a los tramos. Como ejemplo emplearemos la velocidad media del tramo, lo que permitiría identificar que tramos son los más lentos de cada línea.

3.4.1. LÍNEA 14

¿Qué stop tiene menor frecuencia de paradas?	90 (E. Gonzalez, 54)
¿Qué stop tiene una mayor frecuencia de paradas?	42 (E. Gonlez, M. Azaña)
¿En qué stop se suben más pasajeros?	9 (Caballeros Estación)
¿En qué stop se suben más pasajeros que utilizan otros tipos de pago que no la tarjeta?	119 (EEFF, Glorieta)
¿En qué stop se suben más pasajeros que pagan con la tarjeta millennium?	9 (Caballeros Estación)
¿En qué stop se suben más pasajeros que están haciendo transbordo?	9 (Caballeros Estación)
¿En qué stop se suben más universitarios?	9 (Caballeros Estación)
¿En qué stop se suben más desempleados?	20 (Os Castros)
¿En qué stop se suben más jubilados?	20 (Os Castros)
¿En qué stop se suben más minusválidos?	42 (E. Gonlez, M. Azaña)
¿En qué stop se suben más pensionistas?	45 (G. Canaria, Colombia)
¿En qué stop se utilizan más tarjetas con defecto?	20 (Os Castros)
¿Cuál es el tramo con menor distancia?	55 -> 20, con 0.09069689 km (C.Pasaje, Av.Caidos -> Os Castros)
¿Cuál es el tramo con menor tiempo?	264 -> 265, con 0.5666667 min (Rd. Out. J. Planells -> Rda. Out. Viaducto)
¿Cuál es el tramo con menor velocidad media?	55 -> 20, con 0.15 km/h (C.Pasaje, Av.Caidos -> Os Castros)

¿Cuál es el tramo con mayor distancia?	20 -> 21, con 0.6 km (Os Castros -> Glorieta Av Ejército)
¿Cuál es el tramo con mayor tiempo?	20 -> 21, con 2.36 min (Os Castros-> Glorieta Av Ejército)
¿Cuál es el tramo con mayor velocidad media?	21 -> 268, con 38.8 km/h (Glorieta Av Ejército -> R. y Cajal, Casa Mar)

Empleando el paquete plotGoogleMaps, podemos representar los resultados en un mapa de google interactivo que se mostrará en el navegador (ver figura 23 como ejemplo), sin embargo decidimos emplear los gráficos estándar de R porque facilitan la preparación de informes.

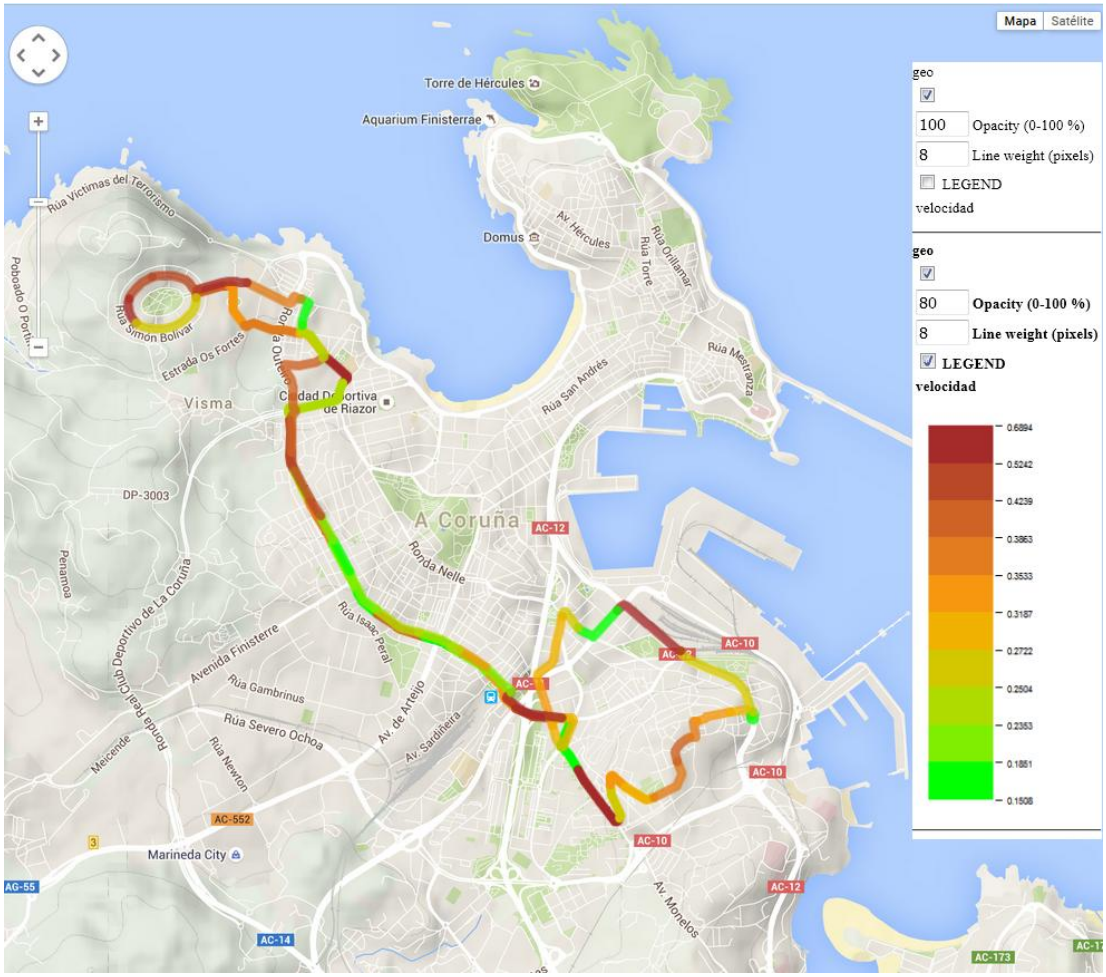


Figura 23. Mapa de google de A Coruña con la Línea 14 (ida y vuelta) con los tramos coloreados de acuerdo con la velocidad media de los autobuses.

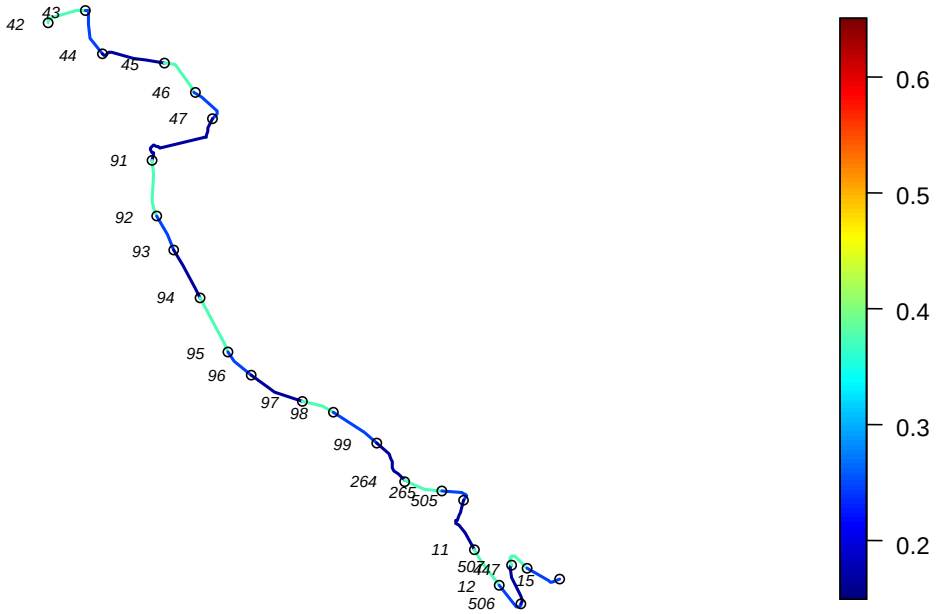


Figura 24. Subgrafo de Línea 14 (ida) de la red de autobuses de A Coruña. Cada arco entre dos stops esta coloreado de acuerdo con la velocidad media de los autobuses en ese tramo.

Podemos observar en la Figura 24, que en la ida de la Línea 14 tenemos 15 tramos (azul oscuros) en que la velocidad media de los autobuses es de menos de 20km/h, llegando como mucho a una velocidad media de 40 km/h en unos pocos tramos.

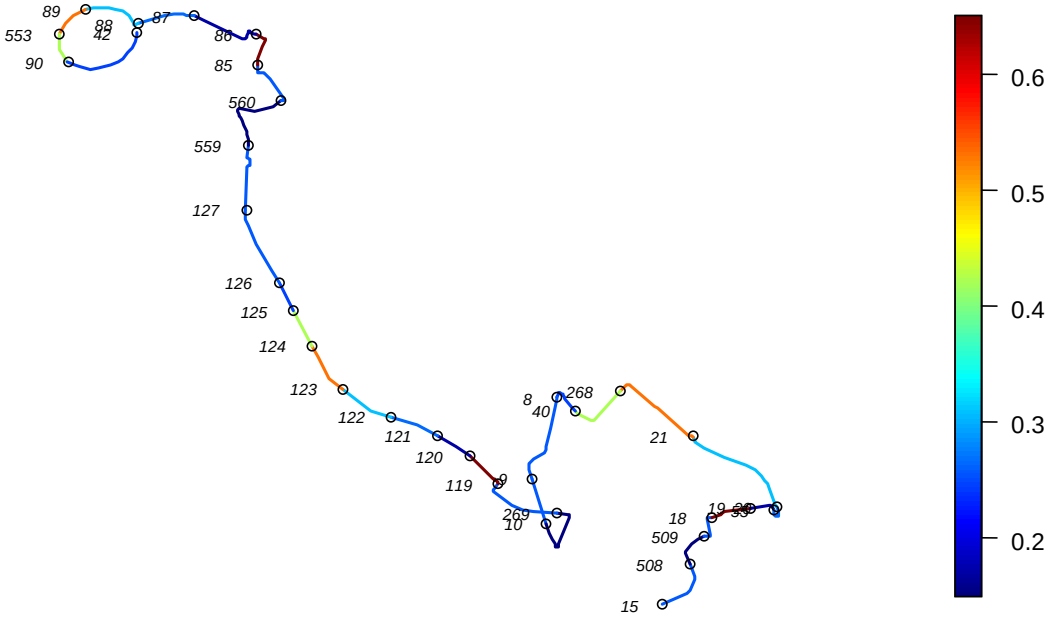


Figura 25. Subgrafo de Línea 14 (vuelta) de la red de autobuses de A Coruña. Cada arco entre dos stops esta coloreado de acuerdo con la velocidad media de los autobuses en ese tramo.

Podemos observar en la Figura 25, que en la mayor parte de los tramos de la vuelta de la línea 14 los autobuses tiene una velocidad media bastante aceptable, entre 20 y 40 km/h, llegando hasta 60km/h en algunos tramos.

3.4.2. LÍNEA E (ESPECIAL UNIVERSIDAD)

¿Qué stop tiene menor frecuencia de paradas?	558 (Arquitectura Interc.)
¿Qué stop tiene una mayor frecuencia de paradas?	160 (Juan Florez, 10)
¿En qué stop se suben más pasajeros?	460 (Intercampus,Gta.Elvi)
¿En qué stop se suben más pasajeros que utilizan otros tipos de pago que no la tarjeta?	460 (Intercampus,Gta.Elvi)
¿En qué stop se suben más pasajeros que pagan con la tarjeta millennium?	460 (Intercampus,Gta.Elvi)
¿En qué stop se suben más pasajeros que están haciendo transbordo?	160 (Juan Florez, 10)
¿En qué stop se suben más universitarios?	460 (Intercampus,Gta.Elvi)
¿En qué stop se suben más desempleados?	160 (Juan Florez, 10)
¿En qué stop se suben más jubilados?	160 (Juan Florez, 10)
¿En qué stop se suben más minusválidos?	160 (Juan Florez, 10)
¿En qué stop se suben más pensionistas?	160 (Juan Florez, 10)
¿En qué stop se utilizan más tarjetas con defecto?	460 (Intercampus,Gta.Elvi)
¿Cuál es el tramo con menor distancia?	462 -> 456, con 0.1669502 km (Zapateira, Aparejad. -> Intercampus, Arquit.)
¿Cuál es el tramo con menor tiempo?	287 -> 394, con 0.3333333 min (A. Molina, SEAT -> C. Campus Elviña)
¿Cuál es el tramo con menor velocidad media?	356 -> 462, con 0.06714432 km/h (Zapateira, Filología -> Zapateira, Aparejad.)
¿Cuál es el tramo con mayor distancia?	394 -> 348, con 0.7401622 km (C. Campus Elviña -> Campus Elviña, Glor.)
¿Cuál es el tramo con mayor tiempo?	356 -> 462, con 7.533333 min (Zapateira, Filología -> Zapateira, Aparejad.)
¿Cuál es el tramo con mayor velocidad media?	287 -> 394, con 0.6433591 km/h (A. Molina, SEAT -> C. Campus Elviña)

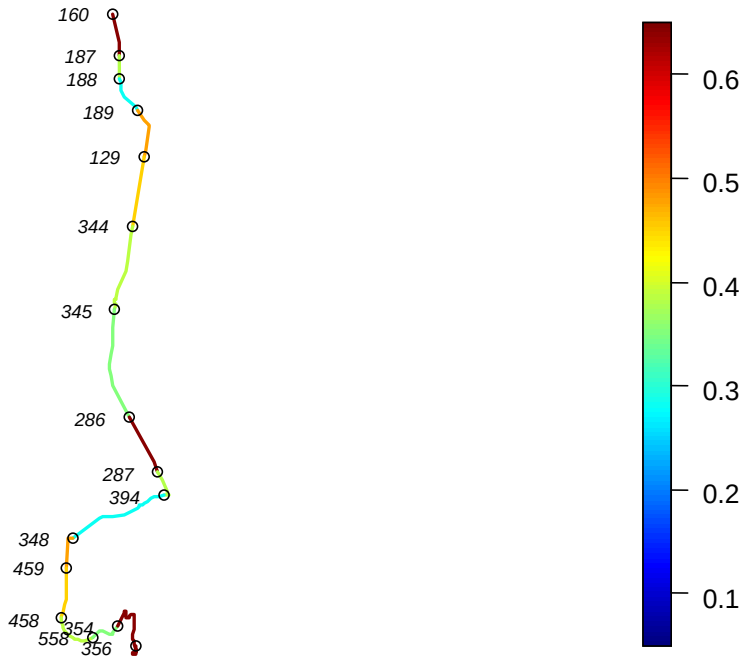


Figura 26. Subgrafo de Línea E – Especial Universidad (ida) de la red de autobuses de A Coruña. Cada arco entre dos stops esta coloreado de acuerdo con la velocidad media de los autobuses en ese tramo.

La Figura 26 nos muestra que la ida de la Línea especial Universidad, no tiene tramos de lentitud, toda la ruta tiene una velocidad media entre 30 y 65km/h.

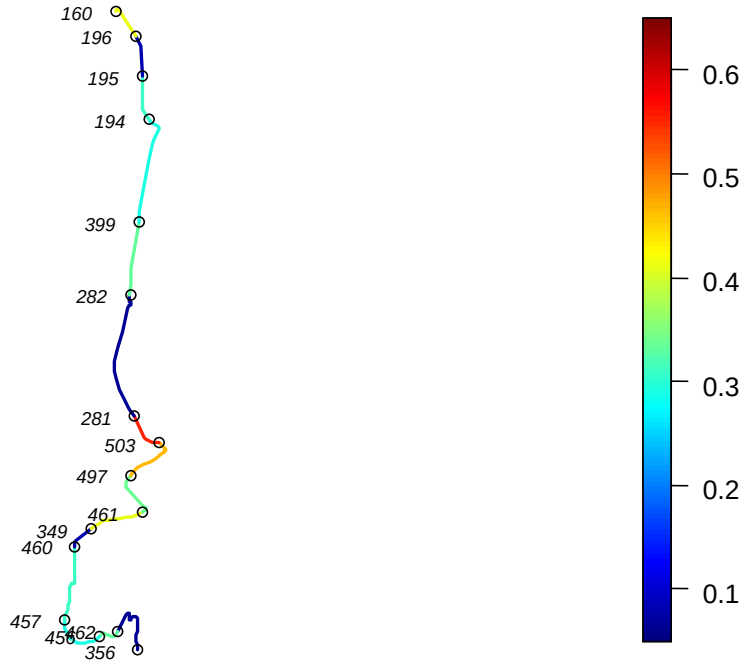


Figura 27. Subgrafo de Línea E – Especial Universidad (vuelta) de la red de autobuses de A Coruña. Cada arco entre dos stops esta coloreado de acuerdo con la velocidad media de los autobuses en ese tramo.

En la figura 27 podemos ver que la vuelta de la línea Especial Universidad tiene muchos tramos con una velocidad media entre 10 y 30 km/h.

3.4.3. LÍNEA 4

¿Qué stop tiene menor frecuencia de paradas?	184 (Panaderas, Museo)
¿Qué stop tiene una mayor frecuencia de paradas?	181 (Hércules, Mercado)
¿En qué stop se suben más pasajeros?	74 (Caballeros, E. Bus)
¿En qué stop se suben más pasajeros que utilizan otros tipos de pago que no la tarjeta?	74 (Caballeros, E. Bus)
¿En qué stop se suben más pasajeros que pagan con la tarjeta millennium?	74 (Caballeros, E. Bus)
¿En qué stop se suben más pasajeros que están haciendo transbordo?	197 (Plaza Pontevedra, 11)
¿En qué stop se suben más universitarios?	129 (San Pedro Mezonzo)
¿En Qué stop se suben más desempleados?	186 (Plz Pontev, Femenino)
¿En Qué stop se suben más jubilados?	186 (Plz Pontev, Femenino)
¿En Qué stop se suben más minusválidos?	185 (San Andres, Africano)
¿En Qué stop se suben más pensionistas?	185 (San Andres, Africano)
¿En Qué stop se utilizan más tarjetas con defecto?	186 (Plz Pontev, Femenino)
¿Cuál es el tramo con menor distancia?	12 -> 445, con 0.1186436 km (Av. Monelos, Bloq 2 -> Av. Monelos,Bloque 6)
¿Cuál es el tramo con menor tiempo?	167 -> 168, con 0.2 min (San Andres, 121 -> San Andres, 93)
¿Cuál es el tramo con menor velocidad media?	181 -> 182, con 1.469849 km/h (Hercules, Mercado -> Hercules, 37)
¿Cuál es el tramo con mayor distancia?	129 -> 9, con 0.6026605 km (San Pedro Mezonzo -> Caballeros Estación)
¿Cuál es el tramo con mayor tiempo?	181 -> 182, con 2.9 min (Hercules, Mercado -> Hercules, 37)
¿Cuál es el tramo con mayor velocidad media?	167 -> 168, con 43.16918 km/h (San Andres, 121 -> San Andres, 93)

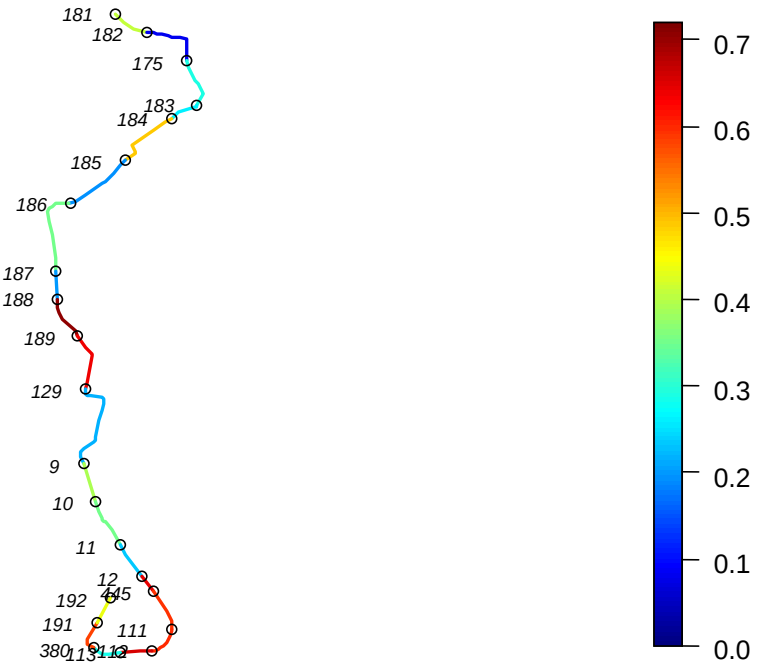


Figura 28. Subgrafo de Línea 4 (ida) de la red de autobuses de A Coruña. Cada arco entre dos stops esta coloreado de acuerdo con la velocidad media de los autobuses en ese tramo.

En la ida de la línea 4 tenemos la mayor parte de los tramos con una velocidad media entre 20 y 50km/h, legando a una velocidad máxima de 70km/h.

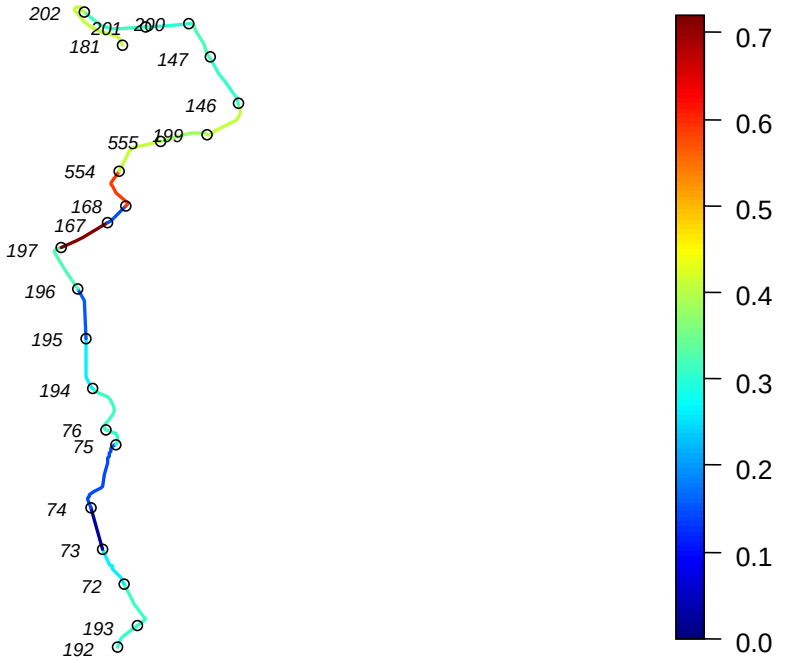


Figura 29. Subgrafo de Línea 4 (vuelta) de la red de autobuses de A Coruña. Cada arco entre dos stops esta coloreado de acuerdo con la velocidad media de los autobuses en ese tramo.

En la Figura 29 observamos que la vuelta de la línea 4 es muy similar a la ida, y tenemos la mayor parte de los tramos con una velocidad media entre 20 y 50km/h, legando a una velocidad máxima de 70km/h.

CONSIDERACIONES FINALES

Los análisis exploratorios clásicos y de red del presente trabajo nos ayudan a conocer mejor las bases de datos y el tipo de información que pueden aportar. El presente trabajo está siempre abierto a extensiones y a la aplicación de otras herramientas, instrumentos y otras bases de datos, que contribuyan a ampliar los resultados aquí expuestos. Futuramente se podrían constituir líneas de investigación como:

- valorar el impacto de las obras en el desplazamiento por transporte público, si disponemos de informaciones sobre las obras y los cambios de las rutas.

- Estimar la frecuencia de usuarios por tramo entre dos stops, por muestreo haciendo encuestas a los viajeros.

- representar en un mapa la densidad demográfica de Coruña (según la dirección de los usuarios de las Tarjetas Millennium), y sobre ese mapa representar la red de autobuses de A Coruña coloreando cada tramo entre dos stops por la cantidad de pasajeros, con el objetivo de identificar si hay zonas que necesitan más autobuses o mas líneas. Para eso necesitaremos acceso a la base de datos de padrón de manera a obtener la dirección de los usuarios de la tarjeta Millennium.

- Se podría extender el estudio que hemos hecho de los subgrafos las tres líneas de autobuses urbanos más utilizadas en A Coruña a todas las líneas. Podríamos aun colorear los subgrafos utilizando otros parámetros de interés como por ejemplo hemos utilizado la velocidad media.

- También se podría volver a realizar ese trabajo, pero utilizando solamente la base de datos GTSF. Para eso necesitaríamos los datos de tranvías completos en la base de datos GTSF.

- Se puede aun preparar un paquete de R para el análisis de datos GTSF con una interfaz web.

BIBLIOGRAFIA

Ripley, B. (2014). ODBC Connectivity. Reference manual RODBC.

Csardi, G. y otros. (2015).Reference manual: Network Analysis and Visualization.

García, P. C. SQL Fácil. (2012). Marcombo.

SÁNCHEZ, L. S. (2013). Google Transit. Trabajo de Fin de Grado en Ingeniería Informática.

Wickham, H. and Francois, R. A Grammar of Data Manipulation. Reference manual dplyr.

Kolaczyk, E. D. and Csárdi, G. (2014). Statistical analysis of network data with R. Springer.

Rodríguez M. L. C. y Freire, S.M.L. (2014/2015). Apuntes de la asignatura Modelos Interactivos de la Investigación Operativa.

Díaz, J. G. (2013/2014). Apuntes de la asignatura Programación Lineal y Entera, modelos de optimización en redes.

Schlesinger, T. and. Eugster, M. J. A . (2015). Reference manual: OpenStreetMap and R.

Pebesma, E. and Bivand, R. (2015) . Reference manual: Classes and Methods for Spatial Data.

Pardiñas, R. J. (2013/2014). Apuntes de la asignatura Análisis Exploratorio de Datos.

Material concedido por el ayuntamiento de A Coruña.