Taking control of graphics using R

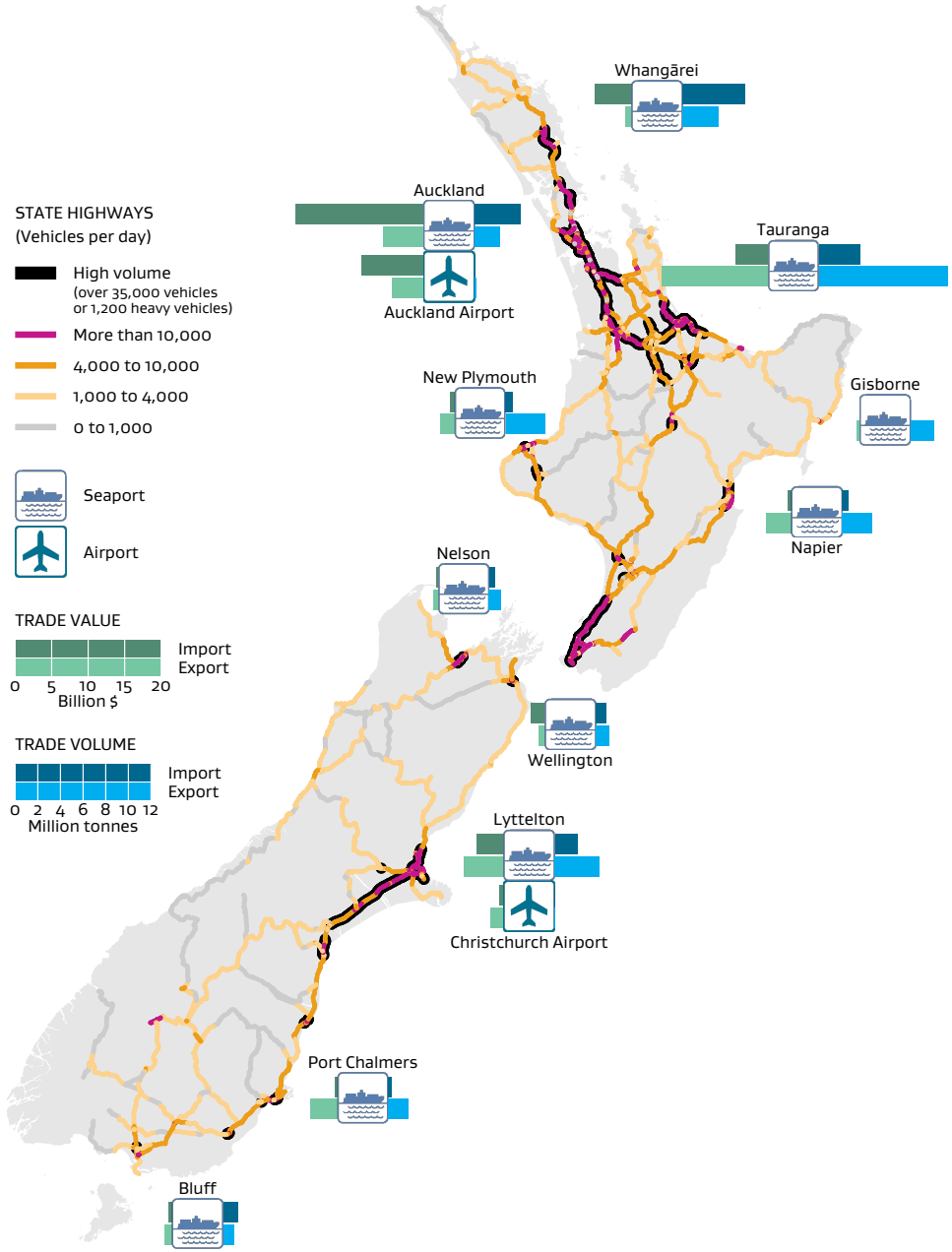
Yvan Richard

Wellington R-Users Group
17 March 2014

Outline

1. Data preparation
2. Plotting
3. Subplots

34. Major New Zealand transport network connections 2013



Sources: New Zealand Transport Agency
Statistics New Zealand

Organisation

Separation of data preparation and plot generation

Link the whole process using a makefile:

```
all: plot_dir/the_plot.png

plot_dir/the_plot.png: data_dir/the_data.rdata \
                       plot_dir/create_plot.r
        cd plot_dir && Rscript create_plot.r

data_dir/the_data.rdata: data_dir/prepare_data.r \
                         data_dir/raw_data.csv
        cd data_dir && Rscript prepare_data.r
```

Several sources:

- GIS shapefiles (NZ coastline, road networks)
- Spreadsheet with port locations and position tweaks (manual)
- Spreadsheets with import/export values and volumes

Data - shapefiles

```
library(rgdal)
nz <- readOGR("gis_folder", "nz-coastline",
              p4s="+init=epsg:4326")  ## lat/long projection
## Project to NZTM (in metres)
nz <- spTransform(nz, CRS("+init=epsg:2193"))
## Saves as .shp if needed
writeOGR(nz, ".", "nz-coastline-nztm", "ESRI Shapefile")

## Fortify - Converts spatial object to a data frame of
##  x and y coordinates usable by ggplot
library(ggplot2)
nz <- fortify(nz)
save(nz, file='nz-coastline-nztm.rdata')
```

Data - spreadsheets

	1	2	3	4	5	6
1	location	x	y	offset_x	offset_y	name_dir
2	Auckland Airport	174.763332	-36.84846	-150000	-45000	B
3	Auckland	174.763332	-36.84846	-150000	15000	T
4	Christchurch Airport	172.636225	-43.532054	130000	-40000	B
5	Lyttelton	172.636225	-43.532054	130000	20000	T
6	Dunedin Airport	170.502798	-45.87876	0	0	T
7	Port Chalmers	170.502798	-45.87876	100000	0	T
8	Gisborne	178.017649	-38.662334	80000	0	T
9	Greymouth	171.210762	-42.450392	0	0	T
10	Hamilton Airport	175.279253	-37.787001	0	0	T
11	Invercargill Airport	168.353773	-46.413187	0	0	T
12	Bluff	168.353773	-46.413187	70000	-80000	T
13	Napier	176.912018	-39.492844	100000	-20000	B
14	Nelson	173.283965	-41.270632	0	80000	T
15	New Plymouth	174.075228	-39.055625	-50000	40000	T

```
ports_loc <- read.csv('ports-locations.csv', as.is=T)

library(sp)
coordinates(ports_loc) <- ~ x + y
proj4string(ports_loc) <- "+init=epsg:4326"
ports_loc <- as.data.frame(spTransform(ports_loc, CRS("+init=epsg:2193")))
```

Data - preparation

Normalise data to have only one row per "point"

Not:

	1	2	3	4	5
1	Region	Year_2008	Year_2009	Year_2010	Year_2011
2	Auckland	23548	21543	23512	26128
3	Wellington	10256	11254	13985	12984

But:

	1	2	3
1	Region	year	value
2	Auckland	Year_2008	23548
3	Wellington	Year_2008	10256
4	Auckland	Year_2009	21543
5	Wellington	Year_2009	11254
6	Auckland	Year_2010	23512
7	Wellington	Year_2010	13985
8	Auckland	Year_2011	26128
9	Wellington	Year_2011	12984

Obtained using

```
library(reshape2)
melt(data, id.vars='Region', variable.name='year')
```

Spatial objects

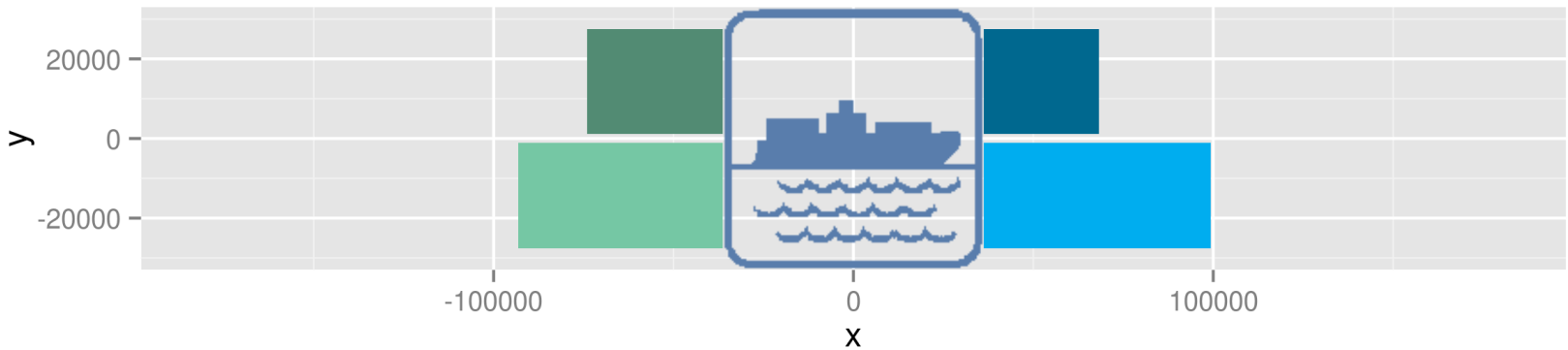
Plot NZ coastline, and road networks:

```
p <- qplot(xlims, ylims, geom = "blank") +
    geom_polygon(aes(x=long, y=lat, group=id), fill=colour_nz, colour=NA,
                 data=nz) +
    geom_path(aes(x=long, y=lat, group=group), data=hightraff,
              size=2.5*size_large_roads, colour='black') +
    geom_path(aes(x=long, y=lat, group=group, colour=traff), data=traff,
              size=size_large_roads) +
    scale_colour_manual(values = routes_cols, guide='none') +
    coord_fixed(ratio=1, xlim=xlims, ylim=ylims) +
    theme_blank()
```

`xlims, ylims, colour_nz, size_large_roads` are variables defined at the beginning of script.

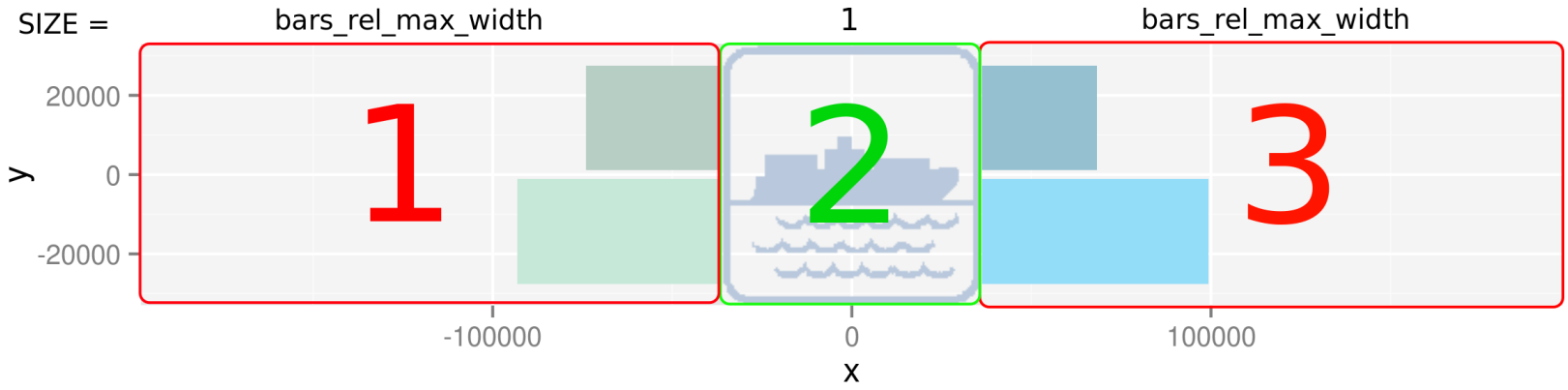
`routes_cols` is a named vector relating discrete values of route traffic to their colour

Without theme



Viewports

Three viewports




```
pushViewport(
  viewport(layout = grid.layout(1, 3,
    widths = unit(c(bars_rel_max_width, 1, bars_rel_max_width),
    rep('grobwidth',3), list(plane_grob, plane_grob, plane_grob)),
    heights = grobHeight(plane_grob))))
```

Subplots - viewports

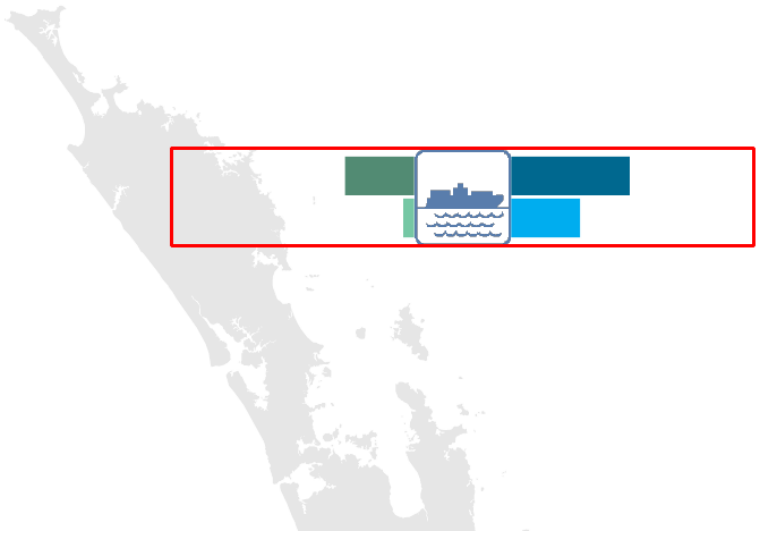
One of the two bar plots:

```
bars_vol <- ggplot(vol, aes(x=import_export, y=value, fill=import_export)) +
    geom_bar(stat='identity', width=1-gap_between_bars) +
    scale_fill_manual(guide='none', values=c(import=colour_bar_imp_vol,
                                             export=colour_bar_exp_vol)) +
   coord_flip(ylim=c(0, max.vol)) +  # make bars horizontal and set limits
    scale_y_continuous(expand=c(0,0)) +  # remove gaps
    theme_nothing()
```

Add it to viewport 3:

```
pushViewport(viewport(layout.pos.row = 1, layout.pos.col = 3))
grid.draw(ggplotGrob(bars_vol))
upViewport()
```

Adding subplot to main plot



```
subplot <- grid.grabExpr({
              pushViewport(...)
              grid.draw(...)
              ... })

subplot_width <- (2*bars_rel_max_width + 1)*subplot_size

p1 <- p + annotation_custom(subplot,
           xmin = loc_x - subplot_width/2, xmax = loc_x + subplot_width/2,
            ymin = loc_y - subplot_size/2, ymax = loc_y + subplot_size/2)
```

Adding grid objects in user coordinates

Example: add a reference scale for the bar plots (a rectangle the width of the longest bar)

```
g <- ggplot(...)
print(g)

downViewport('panel.3-4-3-4')
gb <- ggplot_build(g)
pushViewport(dataViewport(xscale=gb$panel$ranges[[1]]$x.range,
                          yscale=gb$panel$ranges[[1]]$y.range,
                          clip='off'))

grid.rect(x = unit(x_coord, 'native'),
          y = unit(y_coord, 'native'),
          width = unit(subplot_size * bars_rel_max_width, 'native'),
          height = unit(0.5, 'cm'))
```

Adding grid objects in user coordinates

Grid offers many primitive shapes for adding to plots

- grid.rect(), grid.lines(), grid.text(), grid.arrows(), ...

Different units can be conveniently combined (i.e. for adding spacing):

```
grid.text(x = unit(coord_x, 'native') + unit(3, 'mm'), ...)
```

Styling - ggplot

Use `theme`

```
ggplot(...) +
  theme(axis.ticks.length = unit(1.5, "mm"),
        panel.background = element_blank())
```

Or replace geoms with your own function to use consistent formats:

```
title_text <- function(...)
    geom_text(..., size=10, family='Helvetica',lineheight=0.8)

ggplot(...) + title_text('This is a title')
```

Styling - grid

Use gpar

```
grid.rect(gp = gpar(col='red', fill=NA, lwd=2))
```

Also make use of functions for consistency

```
gpar_title <- function(...)
    gpar(fontsize=10, fontfamily='Helvetica', lineheight=0.8, ...)

grid.text('This is a title', gp=gpar_title())
```

Benefits

Yes, it is not very easy, but:

- Scripts are easy to re-use and adapt
- Possibilities almost infinite
- Transparent
- Robust link between data and plots
- Easy to update with new data
- Try doing this in Excel...